MATEMATIKA PRO VYSOKÉ ŠKOLY TECHNICKÉ

MVŠT

SEŠIT
XXXI

Jiří Adámek

Kódování

SNTL

RNDr. Jiří Adámek, CSc.

MATEMATIKA PRO VYSOKÉ ŠKOLY TECHNICKÉ

Kódování

Praha 1989

SNTL - Nakladatelství technické literatury

Obsah

Úvod

Teorie kódování se zabývá konstrukcemi kódů, zaměřenými na různé cíle (např. odstraňování chyb, zrychlení přenosu dat, utajení informace, atd.) a studiem vlastností kódů. V této knize se soustřeďujeme na algebraickou teorii kódů, které samočinně objevují nebo i opravují chyby, tzv. bezpečnostních kódů. Jde o aplikaci lineární algebry: uvažované kódy jsou lineární prostory a opravy chyb se provádějí vynásobením určitou maticí. Algebraické pojmy, které bezpečnostní kódy využívají, podrobně vyložíme, ale předpokládáme, že čtenáři jsou známy základy lineární algebry (lineární prostor, dimenze, součin matic apod.). V první kapitole se kniha zabývá kódováním bez šumu a zejména Huffmanovou konstrukcí nejkratšího kódu. Kapitoly II až VII jsou věnovány bezpečnostním kódům. Poslední kapitola přináší základní informaci o šifrování, tj. kódování zaměřeném na utajení informace. Uvádíme zde slavnou metodu šifrování pomocí velkých prvočísel.

Teorie kódování vznikla ve čtyřicátých letech pracemi Shannona, věnovanými teorii informace, a Hamminga s Golayem, kteří konstruovali první lineární kódy. Hammingovy kódy, opravující jednoduché chyby, a Golayův kód, opravující trojnásobné chyby, dodnes patří mezi nejdůležitější kódy jak z praktického, tak i z teoretického hlediska. V padesátých letech objevili Reed a Muller první třídu kódů, opravující libovolný předepsaný počet chyb. Tyto kódy navíc mají elegantní a snadno proveditelnou metodu dekódování, tj. opravování předepsaných chyb. Později našli Bose, Chaudhuri a Hocquenghem další nesmírně důležité kódy, nazývané BCH kódy, které opravují libovolný počet chyb a mají lepší parametry než Reedovy-Mullerovy kódy. Mají však náročnější dekódování.

V dnešní době je teorie kódování velmi rozvětvená. Sahá od inženýrských aspektů, týkajících se implementace konkrétních kódů, až po aspekty čistě teoretické, související s řadou matematických disciplín: s kombinatorikou, konečnou geometrií, teorií grup, atd. Naše kniha se soustřeďuje na prakticky významné kódy; další informace může čtenář najít v literatuře, kterou uvádíme na konci knihy. Charakter knihy je matematický: Všechny pojmy přesně definujeme a všechna tvrzení podrobně dokazujeme.

Praha, leden 1986 *Autor*

I. KÓDOVÁNÍ BEZ ŠUMU

1. Kódování a dekódování

1.1. Při přenosu zpráv musíme často nahrazovat znaky $a_1, a_2, \ldots, a_r$, kterými jsou zprávy zapsány, binárními symboly 0 a 1. Pokud je počet r původních (tzv. zdrojových) znaků větší než 2, musíme ovšem používat slova, vytvořená z binárních symbolů. Pojem *slova* v množině B znamená konečnou (a neprázdnou) posloupnost prvků této množiny, tedy posloupnost $b_1 b_2 \ldots b_k$, kde $b_i \in B$ pro $i = 1, 2, \ldots, k$. Posloupnosti zapisujeme bez oddělovacích znaků; podrobněji mluvíme o slově délky k. Například slova délky 1 ztotožňujeme s prvky množiny B.

Množinu všech slov označujeme B^*. Zavádíme na ní operaci *skládání slov*:

$$\boldsymbol{b} \mid \boldsymbol{c} = b_1 b_2 \ldots b_k c_1 c_2 \ldots c_n$$

pro libovolná slova $\boldsymbol{b} = b_1 b_2 \ldots b_k$ a $\boldsymbol{c} = c_1 c_2 \ldots c_n$. Například

$$00110 = 00 \mid 110 = 0 \mid 0 \mid 1 \mid 1 \mid 0 .$$

Všimněte si, že v n-prvkové množině B máme n slov délky 1, n^2 slov délky 2, n^3 slov délky 3, atd.

Příkladem nahrazení znaků binárními slovy je tento kód:

0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

Jestliže použijeme tento kód, potom numerickou zprávu

1984

přepíšeme takto:

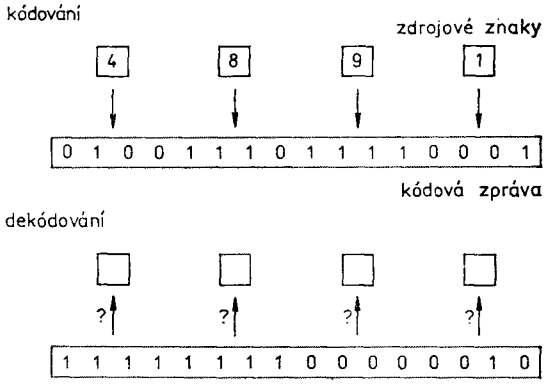
0001111111100100 .

Jednotlivá binární slova od sebe neoddělujeme (čárkami ani mezerami), protože

potom by už nešlo o binární kód; třetím znakem by byla čárka nebo mezera. Například Morseova abeceda je ternární kód, protože používá znaky tečka, čárka a mezera.

Uvedená tabulka je příkladem kódování. Zdrojové znaky 0, 1, ..., 9 kódujeme binárními slovy délky 4.

Obecně je *kódování* předpis, který každému prvku konečné množiny A přiřazuje právě jedno slovo v konečné množině B. Stručněji, kódování je zobrazení $K: A \to B^*$. Množina A se nazývá *zdrojová abeceda* a její prvky *zdrojové znaky*, množina B se nazývá *kódová abeceda* a její prvky *kódové znaky*. Nejdůležitější je případ *binárního kódování*, které má dva kódové znaky (0 a 1). Množinu všech kódových slov $K(a)$, kde a jsou zdrojové znaky, nazýváme stručně *kód*. (Pojmy „kód" a „kódování" není obvykle nutné rozlišovat.) Význam mají jen *prostá* kódování, tj. taková, kdy různým zdrojovým znakům odpovídají vždy různá kódová slova.



Obr. 1

1.2. Příklad. Máme binárně zakódovat informaci o teplotě vody, pro kterou nastávají čtyři možnosti: ledová, studená, vlažná a teplá. Zdrojová abeceda má tedy 4 prvky, takže stačí použít binární slova délky 2. Pokud ale předem víme, že voda je většinou ledová, bude výhodnější použít kódování, které přiřadí prvku „ledová" co nejkratší slovo. Například toto kódování:

ledová	0
studená	01
vlažná	011
teplá	111

Zprávu „ledová, ledová, studená, ledová" zakódujeme nyní takto:

00010.

Umíte dekódovat zprávu

01111111 ?

1.3. Každé kódování zdrojových znaků $K: A \to B^*$ můžeme rozšířit na *kódování zdrojových zpráv*, tedy slov v abecedě A: položme

$$K^*(a_1 a_2 \dots a_n) = K(a_1) \mid K(a_2) \mid \dots \mid K(a_n) .$$

To znamená, že zprávu $a_1 a_2 \dots a_n$ kódujeme znak po znaku. Tím vznikne zobrazení $K^*: A^* \to B^*$, které každé zdrojové zprávě přiřazuje její kódovou zprávu. Například v uvedeném příkladu numerického kódování jsme měli $K^*(1984) = \mathrm{K}(1) \mid \mathrm{K}(9) \mid \mid \mathrm{K}(8) \mid \mathrm{K}(4) = 0001111111100100$.

1.4. Definice. Řekneme, že kódování $K: A \to B^*$ je *jednoznačně dekódovatelné*, jestliže ze znalosti zakódované zprávy $K^*(a_1 a_2 \dots a_n)$ můžeme vždy jednoznačně určit zdrojovou zprávu $a_1 a_2 \dots a_n$; stručněji, jestliže je kódování zpráv $K^*: A^* \to B^*$ prostým zobrazením.

1.5. Příklady

1.5.1. *Blokové kódování* (délky n) je takové prosté kódování, při kterém všechna kódová slova mají stejnou délku (a sice n). Každé blokové kódování je jednoznačně dekódovatelné: pokud známe zprávu $K^*(a_1 a_2 \dots a_m)$, potom prvních n znaků tvoří kód zdrojového znaku a_1, dalších n znaků tvoří kód znaku a_2, atd.

Kódování znaků 0, 1, ..., 9, uvedené v odst. 1.1, je blokové (délky 4).

1.5.2. *Prefixové kódování. Prefixem* (tj. předponou) slova $b_1 b_2 \dots b_k$ se rozumí každé ze slov b_1, $b_1 b_2$, ..., $b_1 b_2 \dots b_k$. Kódování se nazývá *prefixové*, jestliže je prosté a žádné kódové slovo není prefixem jiného kódového slova. Prefixové kódování je vždy jednoznačně dekódovatelné: pokud známe zprávu $K^*(a_1 a_2 \dots a_m)$, potom v ní najdeme nejmenší počet znaků (zleva), které tvoří kódové slovo, a ty jsou kódem znaku a_1. Dekódované znaky umažeme a zase hledáme nejmenší počet znaků, tvořících kódové slovo, to je kód znaku a_2, atd.

Každé blokové kódování je ovšem prefixové. Zde je příklad ternárního kódování, které není blokové, ale je prefixové:

A	0
B	1
C	20
D	21
E	220
F	221

Zprávu

1122112211

dekódujeme zleva znak po znaku:

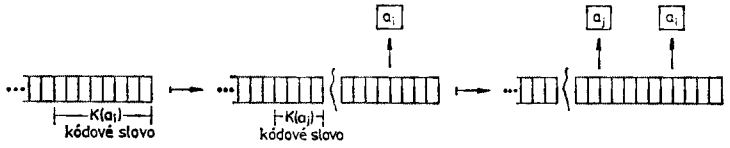
BBFBFB .

1.5.3. Kódování teploty (1.2) není prefixové. (Kód znaku „ledová" je prefixem kódu znaku „studená".) Přesto jde o jednoznačně dekódovatelný kód: zprávy dekódujeme odzadu. Je-li na konci zakódované zprávy nula, je poslední znak zdrojové zprávy „ledová". Je-li na konci právě jedna jednička, je poslední znak „studená", atd.

1.6. Poznámka. Prefixové kódy mají velký význam. Jsou to jediné kódy, které lze dekódovat znak po znaku. To znamená, že při dekódování zprávy $K^*(a_1a_2a_3\ldots)$ nemusíme znát celý výsledek, ale v okamžiku přijetí začátku zprávy, tvořeného slovem $K(a_1)$, můžeme již určit zdrojový znak a_1. (Například kód teploty vody (1.2) nemůžeme dekódovat znak po znaku. Dokonce velmi dlouhou zprávu tohoto typu:

011111111111 ...

můžeme dekódovat teprve po jejím úplném skončení – potom teprve budeme s to určit, zda první přijaté slovo je 0, 01 nebo 011.)



dekódování prefixového kódu

Obr. 2

1.7. Úlohy

1.7.1. Ověřte přesně, že kód je prefixový, právě když jej můžeme dekódovat znak po znaku.

1.7.2. Ověřte, že jednoznačně dekódovatelný kód je vždy prostý.

1.7.3. Je kód

A	01
B	10
C	0100
D	001
E	110

prostý? Je jednoznačně dekódovatelný? Návod: zkuste 0100110.

2. Konstrukce prefixových kódů

Ukážeme, že při konstrukci kódů stačí znát délky kódových slov, která chceme použít. Pomocí známých délek snadno sestavíme prefixový kód. Jak máme délky volit, určuje tzv. Kraftova nerovnost. Příjemné překvapení: prefixové kódy, které se

nejlépe dekódují, jsou stejně obecné jako vůbec všechny jednoznačně dekódovatelné kódy. Tento výsledek, tzv. McMillanova věta, ukazuje, že se můžeme omezit jen na prefixové kódy.

2.1. Příklad. Chceme sestrojit binární prefixový kód cifer 0, 1, ..., 9, vhodný pro zprávy, ve kterých se často vyskytuje 0 a 1, ale zřídka se vyskytuje 8 a 9. To znamená, že znakům 0 a 1 chceme přiřadit slova co nejkratší. Délka těchto slov ovšem nemůže být 1, protože pro další znaky bychom nenašli kódová slova. Kód totiž má být prefixový a každé binární slovo má prefix buď 0, nebo 1. Zvolme tedy slova délky 2 pro znaky 0 a 1. Pro znaky 2, 3, ..., 7 bychom chtěli zvolit slova délky 3 a pro znaky 8 a 9 slova délky 4. Je to možné?

Zvolme kódy

0 00

1 01 .

Potom žádné další kódové slovo nemůže začínat nulou (jinak by mělo buď prefix 00, nebo prefix 01). Počet všech slov délky tři, která začínají jedničkou, je však pouze $2^2 = 4$. Nemůžeme jimi tedy zakódovat čísla 2 až 7. Stejný výsledek jistě dostaneme, kdykoli zakódujeme znaky 0 a 1 jinými slovy délky 2.

Snížíme-li své požadavky tak, že znaky 2 až 9 budeme kódovat slovy délky 4, potom takový kód sestavíme. Stačí volit libovolná slova délky 4, začínající jedničkou: jejich počet je $2^3 = 8$. Zde je výsledný kód:

0 00

1 01

2 1000

3 1100

4 1010

5 1001

6 1110

7 1101

8 1011

9 1111

2.2. Poznámka. Tak jako v předchozím příkladu, při konstrukci prefixového kódu vždy stačí znát požadované délky $d_1, d_2, \ldots, d_r$ kódových slov pro jednotlivé zdrojové znaky $a_1, a_2, \ldots, a_r$. Předpokládejme pro jednoduchost, že zdrojové znaky jsme uspořádali tak, že délky tvoří neklesající posloupnost, $d_1 \leqq d_2 \leqq \ldots \leqq d_r$. Zvolíme libovolné slovo délky d_1 a to označíme jako $K(a_1)$. Dále zvolíme libovolné slovo délky d_2, které nemá prefix $K(a_1)$, a to označíme jako $K(a_2)$, atd. Je ovšem otázka, jak smíme volit délky kódových slov.

V případě binárního kódu existuje 2^{d_1} slov délky d_1 a mezi nimi volíme slovo $K(a_1)$. Při volbě slova $K(a_2)$ se musíme vyhnout všem slovům délky d_2 s prefixem $K(a_1)$; jejich počet je $2^{d_2-d_1}$ (protože každé takové slovo vznikne z prefixu $K(a_1)$ doplněním libovolného slova délky $d_2 - d_1$). Platí $2^{d_2-d_1} < 2^{d_2}$, a proto můžeme volit $K(a_2)$ jako některé zbývající slovo délky d_2. Při volbě slova $K(a_3)$ mohou už vzniknout potíže. Musíme se vyhnout všem slovům s prefixem $K(a_1)$, jejichž počet je $2^{d_3-d_1}$, a všem slovům s prefixem $K(a_2)$, těch je $2^{d_3-d_2}$. Potřebujeme, aby celkový počet 2^{d_3} slov délky d_3 byl alespoň o 1 větší než počet slov, kterým se vyhýbáme:

$$2^{d_3-d_1} + 2^{d_3-d_2} + 1 \leqq 2^{d_3},$$

anebo, což je totéž (po vynásobení členem 2^{-d_3}),

$$2^{-d_1} + 2^{-d_2} + 2^{-d_3} \leqq 1.$$

Podobně pro čtyři a více slov. Abychom mohli dojít až do konce, je tedy nutné (i postačující), aby platilo

$$2^{-d_1} + 2^{-d_2} + \ldots + 2^{-d_r} \leqq 1.$$

Tato podmínka je nutná nejen v případě prefixových kódů, ale i v případě jednoznačně dekódovatelných kódů, jak plyne z McMillanovy věty, kterou dokážeme později. Uvedený postup zformulujeme obecně:

2.3. Věta. *Při kódování n znaky můžeme sestavit prefixový kód s délkami kódových slov $d_1, d_2, \ldots, d_r$, právě když platí tzv. Kraftova nerovnost*

(2.3.1) $$n^{-d_1} + n^{-d_2} + \ldots + n^{-d_r} \leqq 1.$$

Důkaz. Jestliže platí Kraftova nerovnost, ukážeme, že existuje prefixový kód $K(a_1), K(a_2), \ldots, K(a_i)$ s délkami slov $d_1, \ldots, d_i$ pro každé $i = 1, \ldots, r$. Postupujeme matematickou indukcí. Můžeme jistě předpokládat, že zdrojové znaky jsou seřazeny tak, že $d_1 \leqq d_2 \leqq \ldots \leqq d_r$. Pro $i = 1$ zvolíme libovolné slovo $K(a_1)$ délky d_1. V indukčním kroku předpokládáme, že máme prefixový kód $K(a_1), \ldots, K(a_{i-1})$. Při volbě slova $K(a_i)$ se musíme vyhnout všem slovům délky d_i s některými z prefixů $K(a_1), \ldots, K(a_{i-1})$. Počet těchto „zakázaných" slov je

$$n^{d_i-d_1} + n^{d_i-d_2} + \ldots + n^{d_i-d_{i-1}}.$$

Protože platí Kraftova nerovnost, máme

$$n^{-d_1} + n^{-d_2} + \ldots + n^{-d_{i-1}} + n^{-d_i} \leqq 1,$$

a tedy

$$n^{d_i-d_1} + n^{d_i-d_2} + \ldots + n^{d_i-d_{i-1}} + 1 \leqq n^{d_i}.$$

Odtud plyne, že mezi všemi n^{d_i} slovy délky d_i lze vybrat alespoň jedno slovo, které není „zakázané". To označíme $K(a_i)$.

Naopak z existence prefixového kódu plyne Kraftova nerovnost. Čtenář může tento fakt snadno dokázat podobnou úvahou jako v binárním případě, nebo jej může odvodit z McMillanovy věty, kterou dokážeme dále.

2.4. Poznámka. Kraftova nerovnost ovšem nezaručuje, že daný kód je prefixový — zaručuje jen, že existuje prefixový kód se stejnými délkami kódových slov.

Například v 1.2 jsme měli binární kód, kde $d_1 = 1$, $d_2 = 2$ a $d_3 = d_4 = 3$. Platí sice

$$2^{-1} + 2^{-2} + 2 \cdot 2^{-3} = \frac{1}{2} + \frac{1}{4} + \frac{2}{8} = 1 ,$$

ale kód sám není prefixový. Zde je prefixový kód se stejnými délkami slov:

ledová	0
studená	10
vlažná	110
teplá	111

2.5. Příklad. Vraťme se k příkladu 2.1 kódování cifer. Původně jsme chtěli kódovat s délkami $d_0 = d_1 = 2$, $d_8 = d_9 = 4$ a ostatní délky $d_i = 3$ $(i = 2, \ldots, 7)$. Platí ovšem

$$2\frac{1}{2^2} + 2\frac{1}{2^4} + 6\frac{1}{2^3} = \frac{12}{8} > 1 ,$$

takže takové prefixové kódování neexistuje. Jestliže však připustíme délku 5 pro nejméně časté znaky 8 a 9, potom na tyto znaky „spotřebujeme" jen $2\,(1/2^5) = 1/16$ a na znaky 0 a 1 „spotřebujeme" $2\,(1/2^2) = 1/2$, takže do jedničky zbývá $7/16$. Potom můžeme znak 2 kódovat slovem délky 3 a ostatní znaky 3, 4, 5, 6, 7 slovy délky 4: platí

$$2\frac{1}{2^5} + 2\frac{1}{2^2} + \frac{1}{2^3} + \frac{5}{2^4} = 1 .$$

Zde je takový kód:

0	00
1	01
2	100
3	1010
4	1011
5	1100
6	1101
7	1110
8	11110
9	11111

2.6. Úlohy

2.6.1. Existuje binární prefixové kódování číslic 0, 1, ... 9, které používá jen slov délky nejvýše 5 a číslice 9 má kód 0?

2.6.2. Najděte ternání (tj. tří-znakový) prefixovaný kód znaků A, B, C, D, E, F s předepsanými délkami kódových slov 2, 2, 1, 1, 2, 2.

2.6.3. Kolik znaků musí mít kódová abeceda k zakódování všech 26 písmen (bez háčků) naší abecedy, jestliže kódová slova mají mít délku nejvýše 2 a pro 5 samohlásek mají mít dokonce délku 1?

2.7. McMillanova věta. *Pro každé jednoznačně dekódovatelné kódování platí Kraftova nerovnost* (2.3.1).

Poznámka. To znamená (podle věty 2.3), že k takovému kódování můžeme sestrojit prefixové kódování se stejnými délkami slov.

Důkaz. Je dáno jednoznačně dekódovatelné kódování K s délkami kódových slov $d_1 \leqq d_2 \leqq \ldots \leqq d_r$. Chceme dokázat, že číslo $c = n^{-d_1} + n^{-d_2} + \ldots + n^{-d_r}$ je nejvýše rovno 1. To ověříme úpravou jeho k-té mocniny ($k = 1, 2, 3, \ldots$). Ta je ovšem součtem všech možných k-členných součinů čísel n^{-d_i}:

$$c^k = (n^{-d_1} + n^{-d_2} + \ldots + n^{-d_r})^k = \sum_{i_1,\ldots,i_k=1}^{r} n^{-(d_{i_1}+d_{i_2}+\ldots+d_{i_k})}.$$

V tomto součtu uděláme pořádek: zapíšeme jej jako $s_1 n^{-1} + s_2 n^{-2} + \ldots$, kde s_j je počet těch sčítanců, pro které $d_{i_1} + d_{i_2} + \ldots + d_{i_k} = j$, tj. sčítanců tvaru n^{-j}. Největší takové číslo j je ovšem rovno $d_r + d_r + \ldots + d_r = kd_r$, takže dostáváme tvar

$$c^k = s_1 n^{-1} + s_2 n^{-2} + \ldots + s_{kd_r} n^{-kd_r}.$$

Pro každý sčítanec $n^{-(d_{i_1}+d_{i_2}+\ldots+d_{i_k})}$ sestavme odpovídající zdrojovou zprávu $a_{i_1} a_{i_2} \ldots a_{i_k}$. Její kód $K(a_{i_1}) K(a_{i_2}) \ldots K(a_{i_k})$ je slovo délky $d_{i_k} + d_{i_2} + \ldots + d_{i_k} = j$. Protože je kódování K jednoznačně dekódovatelné, každým dvěma různým zprávám odpovídají dvě různá slova délky j, to znamená, že počet s_j těchto zpráv nemůže překročit počet n^j všech slov délky j, tj. $s_j \leqq n^j$. Platí tedy

$$\begin{aligned} c^k &\leqq n^1 n^{-1} + n^2 n^{-2} + \ldots + n^{kd_r} n^{-kd_r} = \\ &= 1 + 1 + \ldots + 1 = \\ &= kd_r. \end{aligned}$$

Vidíme, že číslo c splňuje podmínku $c^k/k \leqq d_r$ pro $k = 1, 2, 3, \ldots$. Odtud plyne, že $c \leqq 1$ (protože pro každé číslo $c > 1$ platí $\lim_{k\to\infty} c^k/k = \infty$).

2.8. Úlohy

2.8.1. Ověřte, zda kódování K_i je jednoznačně dekódovatelné. Pokud je, najděte prefixové kódování se stejnými délkami slov.

	K_1	K_2	K_3
A	00	a	0
B	10	ac	1
C	001	b	210
D	101	bc	211
E	011	ca	212
F	111	cb	1010

2.8.2. Jestliže prefixové kódování splňuje ostrou Kraftovu nerovnost, tj. $n^{-d_1} + \ldots + n^{-d_r} < 1$, existuje takové slovo kódové abecedy, které není kódem žádné zprávy. Dokažte to! Návod: Zdrojovou abecedu můžeme rozšířit o 1 znak a najít rozšíření kódování.

2.8.3. Která z uvedených kódování znaků a, b, c, d, e, f jsou jednoznačně dekódovatelná a která jsou prefixová?

	K_1	K_2	K_3	K_4
a	0	xxx	0	0
b	01	xxy	001	1
c	011	xyx	111	20
d	0111	yxx	110	21
e	01111	xyy	101	220
f	011111	yyx	011	221

3. Nejkratší kód

V předchozím článku jsme viděli příklady konstrukcí prefixových kódů, založených na tom, že častějším znakům jsme se snažili přiřadit kratší kódová slova než méně častým. V tomto článku jdeme o krok dál: předpokládáme, že jsme zjistili přesnou frekvenci znaků zdrojové abecedy $a_1, a_2, \ldots, a_r$. To znamená, že pro každý znak a_i známe pravděpodobnost p_i toho, že na náhodně zvoleném místě zdrojové zprávy je zapsáno a_i. Platí ovšem $p_1 + p_2 + \ldots + p_r = 1$ a $0 \leqq p_i \leqq 1$. Naším cílem je najít nejkratší kód, přesněji, prefixový kód, jehož průměrná délka slova je nejmenší možná. (Podle McMillanovy věty 2.7 jde o nejkratší kód nejen mezi prefixovými kódy, ale i mezi všemi kódy, které jsou jednoznačně dekódovatelné.)

3.1. Označme délky kódových slov daného kódování $d_1, d_2, \ldots, d_r$ a odpovídající pravděpodobnosti $p_1, p_2, \ldots, p_r$. Z teorie pravděpodobnosti víme, že *průměrná* (nebo *střední*) *délka kódového slova* je číslo

$$\bar{d} = d_1p_1 + d_2p_2 + \ldots + d_rp_r .$$

To znamená, že na zakódování dlouhé zprávy o m zdrojových znacích spotřebujeme zhruba $m\bar{d}$ kódových znaků. Například kódujeme znaky *, +, !, ? a víme, že * se vyskytuje dvakrát častěji než ostatní znaky. Zvolíme toto binární kódování:

znak	*	+	!	?
pravděpodobnost výskytu	0,4	0,2	0,2	0,2
kód	0	10	110	111

Jeho průměrná délka slova je

$$\bar{d} = 1 \,.\, 0{,}4 + 2 \,.\, 0{,}2 + 3 \,.\, 0{,}2 + 3 \,.\, 0{,}2 = 2 .$$

Stejnou průměrnou délku slova má i toto kódování:

*	+	!	?
00	01	10	11

Je zřejmé, že žádné prefixové binární kódování nemá průměrnou délku menší než 2. Našli jsme tedy nejkratší kód.

3.2. Definice. *Nejkratším n-znakovým kódováním* zdrojové abecedy $a_1, a_2, \ldots, a_r$ s pravděpodobnostmi výskytu $p_1, p_2, \ldots, p_r$ se rozumí prefixové kódování této abecedy pomocí n znaků, které má nejmenší možnou průměrnou délku slova $\bar{d}$.

Například uvedená abeceda *, +, !, ? má nejkratší binární kódy průměrné délky 2. Její nejkratší ternání kód je tento:

*	+	!	?
0	1	20	21

Průměrná délka slova je nyní

$$\bar{d} = 1 \,.\, 0{,}4 + 1 \,.\, 0{,}2 + 2 \,.\, 0{,}2 + 2 \,.\, 0{,}2 = 1{,}4 \,.$$

3.3. Huffmanova konstrukce binárního kódu. Nejkratší kód zkonstruoval O. Huffman (čti hafmen) v r. 1952. Probereme nejprve případ binárního kódování. Zdrojové znaky uspořádáme podle pravděpodobnosti, tj. tak, aby platilo $p_1 \geqq p_2 \geqq \ldots \geqq p_r$. Pokud máme jen dva zdrojové znaky ($r = 2$), je nejkratší kód zřejmý:

znak	a_1	a_2
kód K	0	1

V případě tří znaků ($r = 3$) pracujeme s tzv. redukovanou abecedou a_1 a $a_{2,3}$, která má pravděpodobnosti výskytu p_1 a $p_{2,3} = p_2 + p_3$. Nejkratší kód redukované abecedy je tento:

znak	a_1	$a_{2,3}$
kód K	0	1

Rozdělením slova 1 na dvě slova (10 a 11) dostaneme nejkratší kód původní abecedy:

znak	a_1	a_2	a_3
kód	0	10	11 .

Podobně postupujeme v případě libovolného počtu r zdrojových znaků. Definujeme *redukovanou abecedu* $a_1, a_2, \ldots, a_{r-2}$ a $a_{r-1,r}$ s pravděpodobnostmi výskytu $p_1, p_2, \ldots, p_{r-2}$ a $p_{r-1,r} = p_{r-1} + p_r$. Uspořádáme znovu její znaky podle pravděpodobnosti a najdeme její nejkratší kód K. Potom původní abeceda má nejkratší kód, vzniklý rozdělením slova $K(a_{r-1,r})$ na dvě:

(3.3.1)

znak	a_1	a_2	...	a_{r-2}	a_{r-1}	a_r
kód	$K(a_1)$	$K(a_2)$	...	$K(a_{r-2})$	$K(a_{r-1,r})\,\vert\,0$	$K(a_{r-1,r})\,\vert\,1$.

Dříve než dokážeme, že (3.3.1) je skutečně nejkratším kódem, ilustrujeme metodu redukcí na příkladu. Redukce provádíme postupně, až dojdeme ke dvěma znakům.

3.4. Příklad. Sestrojíme nejkratší kód abecedy 1, 2, 3, A, B, ve které se znak A vyskytuje třikrát častěji než numerické znaky a znak B čtyřikrát častěji. Odhadněte nejkratší kód!

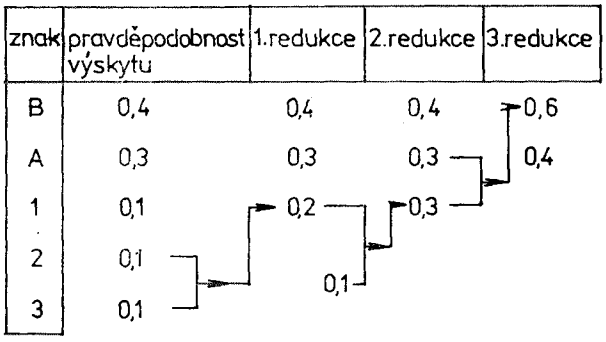
V Huffmanově konstrukci tedy danou zdrojovou abecedu:

znak	B	A	1	2	3
pravděpodobnost výskytu	0,4	0,3	0,1	0,1	0,1

zredukujeme a seřadíme podle pravděpodobností:

znak	B	A	$a_{2,3}$	1
pravděpodobnost výskytu	0,4	0,3	0,2	0,1

Podobně provádíme další redukce. Proces redukcí a seřazování schematicky znázorníme takto:



Obr. 3

Zpětným postupem dostáváme tyto nejkratší kódy:

0,6 ... 0	0,4 ... 1	0,4 ... 1	0,4 ... 1
0,4 ... 1	0,3 ... 00	0,3 ... 00	0,3 ... 00
	0,3 ... 01	0,2 ... 010	0,1 ... 011
		0,1 ... 011	0,1 ... 0100
			0,1 ... 0101

a výsledný kód K je tento:

B ... 1
A ... 00
1 ... 011
2 ... 0100
3 ... 0101

Jeho průměrná délka slova je

$$\bar{d} = 0{,}4 + 2 \,.\, 0{,}3 + (3 + 4 + 4 \;.\, 0{,}1 = 2{,}1 \,.$$

(Byl váš odhad stejně úspěšný?)

3.5. Tvrzení. Je-li K nejkratší kód redukované abecedy, je kód (3.3.1) nejkratším kódem původní abecedy.

Důkaz. I. Nejdříve ukážeme, že pro každou abecedu $a_1, a_2, \ldots, a_r$, seřazenou podle pravděpodobností (viz 3.3), lze sestrojit nejkratší kód K^* takový, že slova $K^*(a_{r-1})$ a $K^*(a_r)$ se liší jen v posledním znaku.

Zvolme libovolný nejkratší kód K'. Délky d'_i kódových slov $K'(a_i)$ neklesají, tj. $d'_1 \leqq d'_2 \leqq \ldots \leqq d'_r$. (Kdyby totiž pro některé $i < j$ platilo $d'_i > d'_j$, mohli bychom přehodit i-té a j-té kódové slovo. Protože platí $p_i \geqq p_j$, vzniklý kód by měl menší průměrnou délku slova než kód K', a to není možné.) Označme K'' kód, vzniklý z kódu K' odstraněním posledního znaku slova $K'(a_r)$. Protože nový kód má kratší průměrnou délku slova než (nejkratší) kód K', nemůže být K'' prefixovým kódem. To znamená, že slovo $K''(a_r)$ je prefixem některého slova $K''(a_i) = K'(a_i)$, $i < r$. Potom platí $d'_r - 1 < d'_i$ a zároveň $d'_i \leqq d'_r$ (protože $i < r$), takže $d'_i = d'_r$. Jestliže $i = r - 1$, stačí položit $K^* = K'$: slova $K'(a_{r-1})$ a $K'(a_r)$ mají stejnou délku a stejný prefix po odstranění posledního znaku. Jestliže $i < r - 1$, označme K^* kód, vzniklý z kódu K' přehozením i-tého slova s $(r-1)$-ním slovem. Protože $d'_i = d'_{r-1} = d'_r$, je K^* nejkratší kód s požadovanou vlastností.

II. Dokážeme, že kód $\tilde{K}$, definovaný v (3.3.1), je nejkratší. Je zřejmé, že $\tilde{K}$ je prefixový kód (protože K je prefixový kód redukované abecedy). Podle I. existuje nejkratší kód K^* takový, že

(3.5.1) $\quad K^*(a_{r-1}) = b_1 \ldots b_m 0 \quad \text{a} \quad K^*(a_r) = b_1 \ldots b_m 1 .$

Z něj sestrojíme kód $\tilde{K}^*$ redukované abecedy:

znak	a_1	a_2	$\ldots$	a_{r-2}	$a_{r-1,r}$
kód $\tilde{K}^*$	$K^*(a_1)$	$K^*(a_2)$	$\ldots$	$K^*(a_{r-2})$	$b_1 \ldots b_m$.

Označme d^*_i délky slov $K^*(a_i)$, platí tedy $d^*_{r-1} = d^*_r = m + 1$ (3.5.1). Proto má kód K^* průměrnou délku slova

$$\bar{d}(K^*) = d^*_1 p_1 + d^*_2 p_2 + \ldots + d^*_{r-2} p_{r-2} + (m+1)(p_{r-1} + p_r) .$$

Kód $\tilde{K}^*$ má tuto průměrnou délku slova:

$$\bar{d}(\tilde{K}^*) = d^*_1 p_1 + d^*_2 p_2 + \ldots + d^*_{r-2} p_{r-2} + m(p_{r-1} + p_r) ,$$

takže platí

$$\bar{d}(K^*) - \bar{d}(\tilde{K}^*) = p_{r-1} + p_r .$$

Podobně se ověří

$$\bar{d}(\tilde{K}) - \bar{d}(K) = p_{r-1} + p_r .$$

Protože je K nejkratším kódem redukované abecedy, platí $\bar{d}(K) \leqq \bar{d}(\tilde{K}^*)$, a tedy

$$\bar{d}(\tilde{K}) = \bar{d}(K) + p_{r-1} + p_r \leqq \bar{d}(\tilde{K}^*) + p_{r-1} + p_r = \bar{d}(K^*) .$$

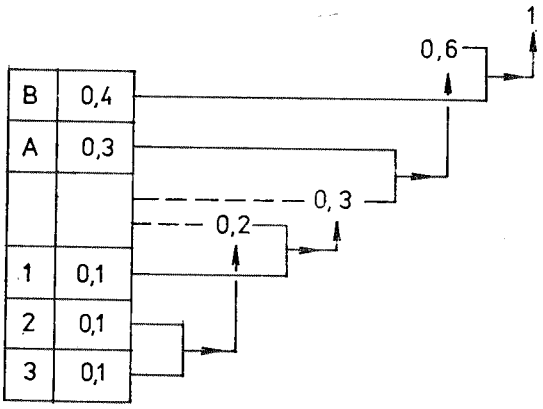
Na druhé straně, protože je K^* nejkratší kód, platí $\bar{d}(\tilde{K}) \geqq \bar{d}(K^*)$, a tedy $\bar{d}(\tilde{K}) = \bar{d}(K^*)$. To znamená, že i $\tilde{K}$ je nejkratší kód.

3.6. Označení. Při provádění Huffmanovy konstrukce je výhodné redukci pouze naznačovat: sečteme dvě poslední, nejmenší pravděpodobnosti a výsledek

zařadíme mezi předchozí pravděpodobnosti (naznačeno přerušovanou vodorovnou čarou). Opět sečteme dvě poslední dosud nesečtené pravděpodobnosti a zařadíme. Tak postupujeme, dokud nedosáhneme součet 1. Potom začneme přiřazovat binární slova: poslední dva sčítanci dostanou kódová slova 0 a 1, a kdykoli má pravděpodobnost kódové slovo $b_1 \dots b_m$ a je součtem dvou sčítanců, dostanou tito sčítanci kódová slova $b_1 \dots b_m 0$ a $b_1 \dots b_m 1$.

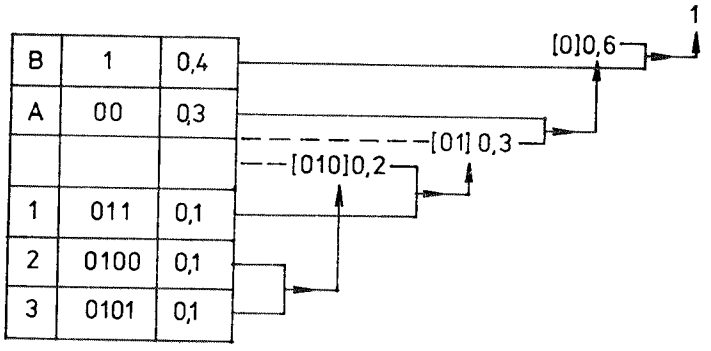
V tomto označení provedeme znovu příklad 3.4.

3.7. Příklad (*Huffmanova konstrukce kódu*). Provedení redukce:



Obr. 4

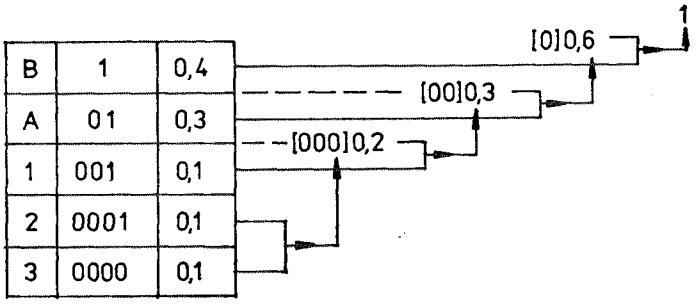
Přiřazení kódových slov:



Obr. 5

Všimněte si, že Huffmanova konstrukce není jednoznačná: podle toho, kam pravděpodobnost zařadíme, dostaneme různé kódy. Také při zpětném přiřazení kódových

slov máme určitou libovůli. Ovšem všechny kódy, které zkonstruujeme, mají stejnou hodnotu $\bar{d}$. Zde je jiný kód pro tutéž abecedu:



Obr. 6

3.8. Úlohy

3.8.1. Najděte odhadem nejkratší kód této abecedy:

znak	x	y	z	t	v
pravděpodobnost	$\frac{1}{10}$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{40}$	$\frac{1}{8}$.

Přesvědčte se o tom, že je skutečně nejkratší.

3.8.2. Pro uvedenou zdrojovou abecedu najděte Huffmanovou konstrukcí dva kódy takové, že jeden má jen slova délky $\leqq 4$, zatímco druhý má i slovo délky 5:

znak	s_1	s_2	s_3	s_4	s_5	s_6
pravděpodobnost	0,04	0,4	0,06	0,3	0,1	0,1 .

3.9. Huffmanova konstrukce obecného kódu. V případě n-znakové kódové abecedy se nejkratší kód najde analogicky jako v binárním případě: provádíme postupné redukce sečtením n nejmenších pravděpodobností. Výjimku tvoří první redukce, kde sčítáme někdy méně než n členů. Předpokládáme zase, že zdrojová abeceda $a_1, \ldots, a_r$ je uspořádána od nejpravděpodobnějšího znaku k nejméně pravděpodobnému.

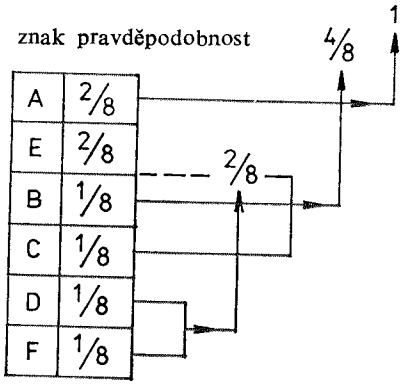
Než přistoupíme k obecnému případu, podívejme se na ternární kódování, kde kódové znaky jsou 0, 1 a 2. Pokud $r = 3$, je nejkratší kód zřejmý:

znak	a_1	a_2	a_3
kód	0	1	2 .

Pokud $r = 4$ nebo 5, provedeme redukci: v případě $r = 4$ je redukovaná abeceda $a_1, a_2, a_{3,4}$, kde $p_{3,4} = p_3 + p_4$ (jen dva sčítanci!), a v případě $r = 5$ je redukovaná abeceda $a_1, a_2, a_{3,4,5}$, kde $p_{3,4,5} = p_3 + p_4 + p_5$. Je zřejmé, že u první redukce se vždy musíme zamyslet, zda sčítat dva nebo tři sčítance. Další redukce ale probíhají

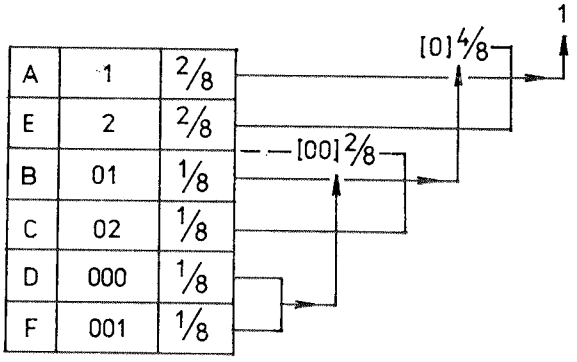
hladce: vždy sčítáme tři poslední nesečtené pravděpodobnosti. První redukce má dva sčítance pro $r = 4, 6, 8, \ldots$ (a ovšem i pro $r = 2$), tedy pro sudá čísla r; a má tři sčítance pro r lichá.

3.10. Příklad. Určíme nejkratší ternární kód abecedy A, B, C, D, E, F, ve které jsou samohlásky dvakrát četnější než souhlásky. Provedeme redukce (první sčítání má dva sčítance, protože kódujeme sudý počet znaků):



Obr. 7

Kódová slova přiřazujeme tak, že v každém rozvětvení „přivěsíme" znaky 0, 1, 2 na konec kódového slova:



Obr. 8

Průměrná délka slova tohoto kódu K je

$$\bar{d} = 1 \cdot \tfrac{2}{8} + 1 \cdot \tfrac{2}{8} + (2 + 2 + 3 + 3)\,\tfrac{1}{8} = \tfrac{14}{8} = 1{,}75\,.$$

3.11. Úloha. Najděte nejkratší ternární kód abecedy v 3.8.2.

3.12. Huffmanova konstrukce *n*-znakového kódu. Kódujeme zdrojovou abecedu $a_1, a_2, \ldots, a_r$, seřazenou podle pravděpodobností, n-znakovou kódovou abecedou $b_1, b_2, \ldots, b_n$. Pokud $r \leqq n$, je nejkratší kód zřejmý:

znak	a_1	a_2	$\ldots$	a_r
kód	b_1	b_2	$\ldots$	b_r.

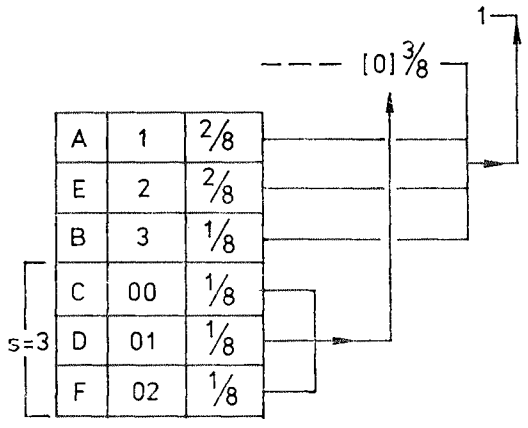
Pokud $r > n$, provedeme redukci zdrojové abecedy. Sečteme posledních s pravděpodobností, kde $s = 2, 3, \ldots, n$ je (jediné) číslo takové, že $n - 1$ je dělitelem čísla $r - s$. Vznikne *redukovaná abeceda* $a_1, a_2, \ldots, a_{r-s}, a^*$ s pravděpodobnostmi $p_1, p_2, \ldots, p_{r-s}$ a $p^* = p_{r-s+1} + p_{r-s+2} + \ldots + p_r$. Tu seřadíme podle pravděpodobností a určíme její nejkratší kód K. Potom má původní abeceda tento nejkratší kód:

znak	a_1	a_2	$\ldots$	a_{r-s}	a_{r-s+1}	a_{r-s+2}	$\ldots$	a_r
kód	$K(a_1)$	$K(a_2)$	$\ldots$	$K(a_{r-s})$	$K(a^*) \mid b_1$	$K(a^*) \mid b_2$	$\ldots$	$K(a^*) \mid b_s$.

3.13. Poznámka. Určení čísla s můžeme mechanicky provádět tak, že v seznamu zdrojových symbolů zatrhneme skupiny po $n - 1$ symbolech, až zbyde poslední skupina s počtem $s = 2, 3, \ldots, n$ symbolů; jejich pravděpodobnosti sečteme. Číslo s určujeme jen u první redukce, protože u všech dalších je $s = n$, tedy sčítáme vždy n posledních pravděpodobností.

Důkaz, že je tento algoritmus správný, je analogický důkazu tvrzení 3.5.

3.14. Příklad. Hledáme nejkratší kód abecedy z příkl. 3.10 s kódovými znaky 0, 1, 2 a 3:



Obr. 9

3.15. Úlohy

3.15.1. Vysvětlete, proč u dalších redukcí není třeba v Huffmanově konstrukci zjišťovat počet sčítanců. Návod: Proberte nejdříve ternární případ.

3.15.2. Kolik znaků musí mít kódová abeceda, jestliže potřebujeme najít prefixový kód průměrné délky slova $\leqq 1{,}6$ pro tuto zdrojovou abecedu:

znak	A	B	C	D	E	F	G	H	I	J	K
pravděpodobnost	0,22	0,15	0,12	0,1	0,1	0,08	0,06	0,05	0,05	0,04	0,03.

II. BEZPEČNOSTNÍ KÓDY

Při skutečném používání kódů je třeba počítat s tím, že během přenosu kódových znaků dojde k chybě, způsobené šumem. Tomu můžeme čelit tím, že používáme tzv. *bezpečnostní* (nebo detekční či opravné) *kódy*. Zatímco Huffmanova konstrukce má za cíl snížit redundanci (tj. nadbytečně přenášenou informaci) na minimum, v případě bezpečnostních kódů naopak uměle zvyšujeme redundanci, abychom zabezpečili informaci před šumem. Šum se může projevit dvěma způsoby:

1. záměnou vyslaného znaku za jiný znak,
2. poruchou synchronizace, tj. vytvořením znaku, ačkoliv nebyl žádný vyslán, nebo naopak pohlcením vyslaného znaku.

Šum druhého typu (který je méně obvyklý) nebudeme uvažovat. Budeme tedy konstruovat kódy, které jsou schopny objevit a případně i opravit chyby, vzniklé záměnou znaků. Pracujeme stále s blokovými kódy (1.5.1). Délku kódového slova budeme vždy označovat n.

Jak může kód objevovat a opravovat chyby? Tak, že obsahuje určitou *redundanci*: některé znaky kódového slova nenesou žádnou informaci, ale slouží ke kontrole. Například čeština je určitým kódem s dost velkou redundancí. Jestliže např. vyšleme české slovo a přijmeme slovo „opakkvání", potom můžeme provést opravu a napsat „opakování". To je totiž jediné české slovo, které má od přijatého slova Hammingovu vzdálenost (tj. počet odlišných znaků) rovnu 1. Při přijetí slova „opakkvánr" chybu objevíme, tj. vidíme, že nejde o české slovo, ale nemůžeme ji opravit. Existuje totiž víc než jedno české slovo s Hammingovou vzdáleností 2, např.

opakování a opakováno ,

ale žádné české slovo nemá Hammingovu vzdálenost 1.

V případě umělých kódů však obecně nemáme možnost rozhodnutí, zda došlo k chybě a k jaké. Tuto možnost ale můžeme vhodnou volbou kódu vytvořit, jestliže vyčleníme některé znaky kódových slov ke kontrole správnosti. Často užívaná technika je tzv. *kontrola parity*: k binárnímu kódovému slovu $a_1a_2 \ldots a_n$ připojíme znak a_{n+1} tak, aby celé slovo $a_1a_2 \ldots a_{n+1}$ obsahovalo sudý počet jedniček. Jestliže při přenosu rozšířeného slova dojde k jediné chybě, poznáme, že přijaté slovo nebylo skutečně vysláno. O tomto kódu říkáme, že *objevuje* jednoduché chyby, tj. zjistí, že došlo k chybě (ale neví kde), pokud byl poškozen jenom jeden vyslaný znak. Kódy objevující chyby jsou výhodné tehdy, když máme možnost zprávu opakovat. Pokud tuto možnost nemáme, budeme spíš používat kódy, které *opravují chyby*, tedy nejen poznají, že došlo k chybě, ale také vědí kde. Například opakuje-li se v kódu každý znak třikrát po sobě, potom tento kód opravuje jednoduché chyby:

vyšleme aaa a jednoduchá chyba ovlivní jediný znak, takže přijmeme např. aba – pak víme, že vysláno bylo aaa. Kódy opravující chyby jsou ovšem náročnější než ty, které chyby jen objevují.

Je jasné, že čím větší je redundance kódu, tím je tento kód méně účinný. Chceme tedy sestrojit kódy, které by opravily nebo objevily co největší počet chyb a přitom měly rozumně nízkou redundanci. Několik vtipných konstrukcí takových kódů poznáme v příštích kapitolách. Nejdříve se budeme zabývat obecnými rysy bezpečnostních kódů.

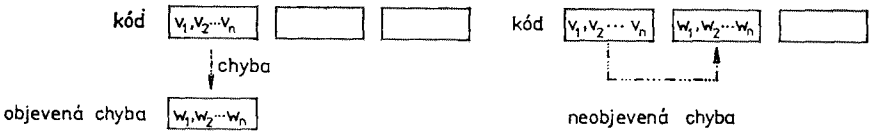
4. Objevování chyb

V celém zbytku knihy pracujeme s blokovým kódem K v nějaké konečné abecedě. Označme T tuto abecedu a

$$T^n = \{t_1 t_2 \dots t_n \mid t_i \in T \text{ pro } i = 1, 2, \dots, n\}$$

množinu všech slov délky n. Pokud délka kódu K (tj. délka všech slov v K) je rovna n, potom $K \subseteq T^n$ a slova množiny T^n dělíme na *kódová slova* (v K) a *nekódová slova* (v $T^n - K$). Předpokládáme, že vysíláme kódová slova a přijímáme slova z množiny T^n. Pokud jsme přijali nekódové slovo, říkáme, že jsme *objevili chybu*. Pokud je přijaté slovo kódové, pak buď nedošlo k chybě, nebo jsme ji neobjevili.

Mluvíme o *t-násobné chybě* ($t = 1, 2, 3, \dots$), jestliže počet chybných míst je nejvýše t. Tedy např. ze slova 1111 může dvojnásobná chyba vytvořit slovo 0110, ale i slovo 1110. Kód K *objevuje t-násobné chyby*, jestliže při vyslání kódového slova a vzniku t-násobné chyby je přijaté slovo vždy nekódové. Podrobněji, pro každé slovo $v_1 v_2 \dots v_n$ kódu K a každé jiné slovo $w_1 w_2 \dots w_n$ takové, že pro nejvýše t indexů i platí $v_i \neq w_i$, je $w_1 w_2 \dots w_n$ nekódové slovo.



Obr. 10

4.1. Příklad: Kód „dva z pěti". Jde o binární kód délky 5, tedy $K \subseteq \{0, 1\}^5$, který sestává ze všech slov s právě dvěma jedničkami. Počet kódových slov je $\binom{5}{2} = 10$, a proto tento kód můžeme použít např. ke kódování cifer:

1	11000	6	00101
2	10100	7	00011
3	10010	8	00110
4	10001	9	01100
5	01001	0	01010

Tento kód objevuje jednoduché chyby ($t = 1$): při jednoduché chybě má přijaté slovo buď jednu, nebo tři jedničky. Dvojnásobné chyby však neobjevuje. Například z kódu čísla 4 vytvoří chyba na prvních dvou místech kód čísla 5.

4.2. Definice. *Hammingovou vzdáleností* dvou slov $v_1v_2 \dots v_n$ a $w_1w_2 \dots w_n$ nazýváme počet odlišných znaků těchto slov, tj. velikost množiny

$$\{i \mid i = 1, 2, \dots, n, v_i \neq w_i\} .$$

Minimální vzdáleností blokového kódu K nazýváme nejmenší Hammingovu vzdálenost dvou různých kódových slov.

4.3. Příklad. Kód „dva z pěti" má minimální vzdálenost 2: slova 11000 a 10100 mají Hammingovu vzdálenost 2 a žádná dvě různá kódová slova nemají Hammingovu vzdálenost 1.

4.4. Úloha. Ověřte, že Hammingova vzdálenost je *metrikou* na množině T^n všech slov délky n. To znamená, že pro každá slova $\boldsymbol{u}$, $\boldsymbol{v}$ a $\boldsymbol{w}$ v T^n splňuje Hammingova vzdálenost $d(\boldsymbol{v}, \boldsymbol{w})$ tyto podmínky:

(i) $d(\boldsymbol{v}, \boldsymbol{w}) \geqq 0$ a rovnost nastává, právě když $\boldsymbol{v} = \boldsymbol{w}$,

(ii) $d(\boldsymbol{v}, \boldsymbol{w}) = d(\boldsymbol{w}, \boldsymbol{v})$,

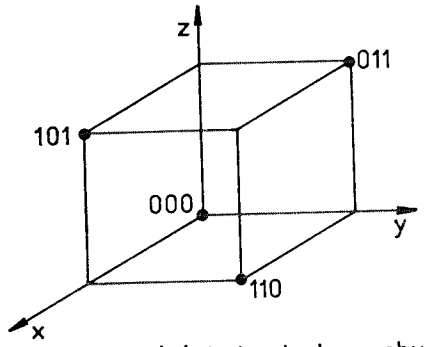
(iii) $d(\boldsymbol{v}, \boldsymbol{w}) \leqq d(\boldsymbol{v}, \boldsymbol{u}) + d(\boldsymbol{u}, \boldsymbol{w})$.

4.5. Pozorování. Blokový kód minimální vzdálenosti d objevuje t-násobné chyby pro všechna $t < d$, ale není schopen objevit všechny d-násobné chyby. To znamená, že pokud v kódovém slově změníme t znaků, potom

a) nevznikne kódové slovo, jestliže $t \leqq d - 1$;

b) může vzniknout kódové slovo, jestliže $t = d$.

Skutečně, když při změně kódového slova $\boldsymbol{v}$ v t znacích vznikne slovo $\boldsymbol{w}$, je Hammingova vzdálenost slova $\boldsymbol{w}$ od slova $\boldsymbol{v}$ rovna t. Pokud $t \leqq d - 1$, znamená to, že buď $\boldsymbol{v} = \boldsymbol{w}$ (a tedy nedošlo k žádné chybě), nebo $\boldsymbol{w}$ není kódové slovo. Naopak, pro $t = d$ najdeme dvě kódová slova $\boldsymbol{v}$ a $\boldsymbol{w}$ Hammingovy vzdálenosti d, a potom $\boldsymbol{w}$ mohlo vzniknout d-násobnou chybou ze slova $\boldsymbol{v}$.



kód objevující jednoduchou chybu (Hammingova vzdálenost 2)

Obr. 11

4.6. Příklad: Kód celkové kontroly parity. Jde o nejobvyklejší bezpečnostní kód vůbec: k binárnímu slovu přidáme jeden kontrolní znak tak, aby výsledné slovo mělo sudou paritu (tj. sudý počet jedniček). Vznikne binární kód délky n všech slov sudé parity. Například pro $n = 4$ je to tento kód (se třemi informačními znaky a jedním kontrolním):

0000, 1100, 1010, 0110, 1001, 0101, 0011, 1111 .

Jeho minimální vzdálenost je 2, takže kód objeví jednoduché chyby. Ty mění sudou paritu v lichou. Žádnou chybu, která změní dva znaky, tento kód neobjeví.

4.7. Příklad: Opakovací kód. V případě velkého šumu se můžeme rozhodnout, že každý znak abecedy T budeme n-krát opakovat. Vznikne opakovací kód, jehož jediná slova jsou $vv \ldots v$ $(v \in T)$.

Například opakovací ternární kód délky 5 má kódová slova 00000, 11111 a 22222. Pokud při přenosu nastane čtyřnásobná chyba, vznikne slovo, které není kódové. Tento kód tedy objevuje čtyřnásobné chyby.

Obecně je minimální vzdálenost opakovacího kódu délky n rovna n, takže kód objeví $(n - 1)$-násobné chyby.

4.8. Úloha. Určete, kolik chyb objeví (a jakým způsobem) binární kód všech slov $a_1a_2 \ldots a_7$, kde a_3 je kontrolní znak parity znaků a_1, a_2 (tedy slovo $a_1a_2a_3$ má vždy sudou paritu), a_6 je kontrolní znak parity znaků a_4, a_5, zatímco a_7 je celková kontrola parity (takže celé slovo má sudou paritu).

5. Opravování chyb

Ukážeme, že kód je schopen opravovat t-násobné chyby, právě když má minimální vzdálenost alespoň $2t + 1$. Jiná je ovšem otázka, jak se opravy realizují. Přijaté slovo většinou nemůžeme porovnávat se všemi kódovými slovy, protože to by bylo příliš zdlouhavé. V dalších kapitolách poznáme důležité třídy kódů s rychlým opravováním chyb.

Říkáme, že kód K *opravuje t-násobné chyby*, jestliže při vyslání kódového slova $\boldsymbol{v}$ a při t-násobné chybě má přijaté slovo $\boldsymbol{w}$ Hammingovu vzdálenost $d(\boldsymbol{v}, \boldsymbol{w})$ menší, než je jeho vzdálenost od libovolného jiného kódového slova (takže vyslané slovo správně určíme jako to, které má k přijatému slovu nejblíže). Jinými slovy, pro každé slovo $\boldsymbol{v} \in K$ a každé slovo $\boldsymbol{w} \in T^n$ takové, že $d(\boldsymbol{v}, \boldsymbol{w}) \leqq t$, platí:

$$d(\boldsymbol{v}, \boldsymbol{w}) < d(\boldsymbol{x}, \boldsymbol{w}) \quad \text{pro každé slovo} \quad \boldsymbol{x} \in K, \quad \boldsymbol{x} \neq \boldsymbol{v}.$$

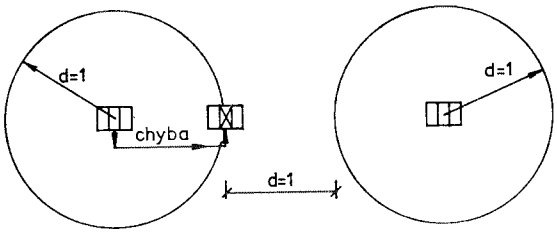
5.1. Pozorování. Blokový kód minimální vzdálenosti d opravuje t-násobné chyby pro všechna

$$t < \frac{d}{2},$$

ale není schopen opravit všechny chyby násobnosti $d/2$ nebo větší. To znamená, že pokud v kódovém slově $\boldsymbol{v}$ změníme t znaků, potom

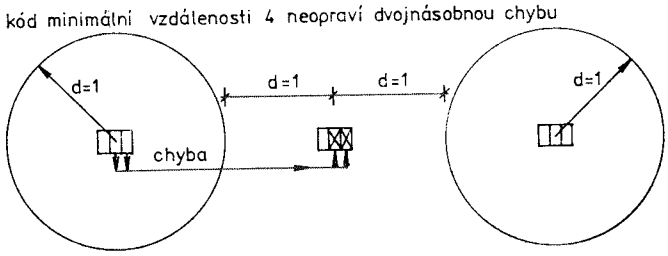
a) vznikne slovo, jehož Hammingova vzdálenost od slova $\boldsymbol{v}$ je menší než od jiných kódových slov, jestliže $t < d/2$;

b) může vzniknout slovo se stejnou nebo menší Hammingovou vzdáleností od jiných kódových slov, jestliže $t \geqq d/2$.



kód minimální vzdálenosti 3 opraví jednoduchou chybu

kód minimální vzdálenosti 4 neopraví dvojnásobnou chybu



Obr. 12

Skutečně, pokud $t < d/2$, má přijaté slovo $\boldsymbol{w}$ Hammingovu vzdálenost $d(\boldsymbol{v}, \boldsymbol{w}) = t < d/2$ od slova $\boldsymbol{v}$. Přitom pro každé jiné kódové slovo $\boldsymbol{x} \neq \boldsymbol{v}$ platí $d(\boldsymbol{v}, \boldsymbol{x}) \geqq d$, takže z nerovnosti $d(\boldsymbol{v}, \boldsymbol{x}) \leqq d(\boldsymbol{v}, \boldsymbol{w}) + d(\boldsymbol{w}, \boldsymbol{x})$ (viz 4.4) plyne

$$d(\boldsymbol{x}, \boldsymbol{w}) \geqq d(\boldsymbol{x}, \boldsymbol{v}) - d(\boldsymbol{v}, \boldsymbol{w}) \geqq d - d(\boldsymbol{v}, \boldsymbol{w}) \geqq d/2 .$$

Naopak, pokud $t \geqq d/2$, najdeme kódová slova $\boldsymbol{v}$ a $\boldsymbol{v}'$ s Hamingovou vzdáleností d. To znamená, že tato slova se liší v některých d souřadnicích $(i_1, i_2, \ldots, i_d)$ a platí $v_i = v_i'$ pro všechna $i \neq i_1, i_2, \ldots, i_d$. Změňme každou druhou ze souřadnic $v_{i_1}, \ldots, v_{i_d}$ na hodnotu ve slově $\boldsymbol{v}'$. Tím vytvoříme slovo $\boldsymbol{w} = w_1 w_2 \ldots w_n$, kde

$$w_i = v_i', \quad \text{pokud} \quad i = i_2, i_4, i_6, \ldots,$$
$$w_i = v_i \quad \text{pro ostatní} \quad i = 1, \ldots, n .$$

Jestliže d je sudé číslo, změnili jsme $d/2$ souřadnic (tj. Hammingova vzdálenost slova $\boldsymbol{w}$ od slova $\boldsymbol{v}$ je $d/2$) a tuto chybu nemůžeme opravit, protože také Hammingova vzdálenost slova $\boldsymbol{w}$ od kódového slova $\boldsymbol{v}'$ je $d/2$ (liší se jen v souřadnicích $i_1, i_3, \ldots, i_{d/2-1}$). Jestliže d je liché číslo, změnili jsme $(d + 1)/2$ souřadnic, tedy nejmenší počet t

souřadnic takový, že $t \geqq (d/2)$. A tuto chybu nemůžeme opravit, protože Hammingova vzdálenost slov $\boldsymbol{w}$ a $\boldsymbol{v}'$ je dokonce menší: $(d - 1)/2$.

5.2. Příklady

5.2.1. Opakovací kód délky 5 opravuje dvojnásobné chyby. Pokud vyšleme slovo *vvvvv* a dojde k chybě na dvou místech, většina znaků zůstane nezměna. Máme tedy dekódování, opravující dvojnásobné chyby: každé slovo $w_1w_2w_3w_4w_5$, ve kterém většinu znaků tvoří znak v, dekódujeme jako slovo *vvvvv*.

Obecně, opakovací kód délky n opravuje t-násobné chyby pro všechna $t < n/2$.

5.2.2. Kód celkové kontroly parity 4.6 (který má minimální vzdálenost 2) neopravuje ani jednoduché chyby: pokud zjistíme lichou paritu, víme, že došlo k chybě, ale nemáme žádnou informaci o tom, kde chyba nastala.

5.2.3. *Kódy dvourozměrné kontroly parity.* Jde o prakticky velmi významné kódy, běžně používané v samočinných počítačích. Informační znaky zapíšeme do matice typu (p, q). Potom ke každému řádku přidáme jeden symbol kontroly parity řádku, podobně ke každému sloupci přidáme znak kontroly parity sloupce a nakonec znak „kontroly kontrol" volený tak, aby i parita výsledné matice byla sudá. Například pro $p = 7$ a $q = 3$ máme $7 \cdot 3$ informačních znaků a $8 \cdot 4 = 32$ všech znaků, takže jde o kód délky 32. Tento kód se nazývá ASCII. Příklad kódového slova:

Kód ASCII:

101	0 ← kontrola parity řádku
000	0
001	1
010	1
111	1
111	1
000	0
110	0 ← celková kontrola parity

↑ kontrola parity sloupce

Kód dvourozměrné kontroly parity opraví jednoduché chyby: taková chyba změní paritu jediného řádku (řekněme i-tého) a jediného sloupce (řekněme j-tého). Potom chybným je znak na místě (i, j).

5.3. Úlohy

5.3.1. Určete způsob, jak objevit trojnásobné chyby při použití kódu dvourozměrné kontroly parity. Opravuje tento kód dvojnásobné chyby?

5.3.2. Kolik chyb opraví kód z úlohy 4.8?

6. Dekódování

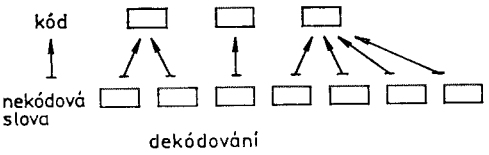
Mluvili jsme o schopnosti kódu odstraňovat chyby, ale ne o tom, jak se opravy skutečně provádějí. To podrobněji probereme u důležitých tříd kódů v dalších kapitolách, teď jen uvedeme některé obecné vlastnosti.

Jestliže na vstup přicházejí kódová slova kódu $K \subseteq T^n$, mohou se na výstupu díky šumu objevit v podstatě libovolná slova množiny T^n. Na to se připravíme tím, že stanovíme, jak dekódovat, tj. jak slovům $w_1 w_2 \dots w_n \in T^n$ (na výstupu) přiřazovat kódová slova $\delta(w_1 w_2 \dots w_n) \in K$, o kterých pak můžeme předpokládat, že byla vyslána. Přesněji, *dekódování* je libovolné zobrazení

$$\delta: T^n \to K,$$

které každému slovu délky n přiřazuje kódové slovo a přitom přijatá kódová slova nemění, tj.

$$\delta(v_1 \dots v_n) = v_1 \dots v_n \quad \text{pro každé slovo} \quad v_1 \dots v_n \in K.$$



Obr. 13

6.1. Příklad: Dekódování opakovacího binárního kódu délky 6. Jestliže přijmeme slovo, kde většina znaků (alespoň 4) je shodná, víme, jak dekódovat:

$$\delta(100100) = 000000.$$

U slov se třemi nulami a jedničkami se však předem žádná volba nenabízí. Můžeme se například řídit prvním znakem. To znamená, že definujeme dekódování

$$\delta: \{0, 1\}^6 \to \{000000, 111111\}$$

předpisem

$$\delta(w_1 w_2 w_3 w_4 w_5 w_6) = \begin{cases} 000000, & \text{je-li } w_i = 0 \text{ pro alespoň 4 indexy } i, \text{ nebo} \\ & w_1 = 0 \text{ a } w_i = 0 \text{ pro 2 indexy } i > 1; \\ 111111 & \text{jinak.} \end{cases}$$

6.2. Poznámka. Konkrétní výběr dekódování může rozšířit množinu typů chyb, kterou jsme schopni opravovat. Například v předchozím příkladě opravuje kód nejen dvojnásobné chyby, ale i ty trojnásobné chyby, které nemění první znak.

Někdy se omezujeme jen na *částečné dekódování*, tj. parciální zobrazení z množiny T^n do množiny K. Například „rozumné" částečné dekódování opakovacího kódu délky 6 je definováno vztahem

$$\delta(w_1 w_2 w_3 w_4 w_5 w_6) = \begin{cases} 000000, & \text{pokud ve slově jsou 4 nuly,} \\ 111111, & \text{pokud ve slově jsou 4 jedničky} \end{cases}$$

(a tedy není definováno pro slova se třemi nulami a třemi jedničkami).

6.3. Příklad: Částečné dekódování kódu dvourozměrné kontroly parity (viz 5.2.3). Pro každou binární matici $\boldsymbol{W}$ uvažovaného typu definujeme $\delta(\boldsymbol{W})$ jen tehdy, když všechny řádky až na případně jediný, i-tý, mají sudou paritu. Potom $\delta(\boldsymbol{W})$ je matice, vzniklá z matice $\boldsymbol{W}$ změnou těch prvků $w_{i,j}$, pro které j-tý sloupec má lichou paritu v matici $\boldsymbol{W}$. Například

$$\begin{bmatrix}0&0&1&1\\1&0&0&0\\1&0&1&0\\0&0&0&0\\0&0&0&0\\1&1&0&0\\0&0&1&1\end{bmatrix} \mapsto \begin{bmatrix}0&0&1&1\\0&1&1&0\\1&0&1&0\\0&0&0&0\\0&0&0&0\\1&1&0&0\\0&0&1&1\end{bmatrix}.$$

6.4. Úloha. Ověřte, že v předchozím příkladě je matice $\delta(\boldsymbol{W})$ kódové slovo, pokud je definována. (Tedy každá matice, která má všechny sloupce a všechny řádky kromě i-tého sudé parity, má také i-tý řádek sudé parity.)

7. Informační znaky

U některých kódů můžeme znaky rozdělit na *informační*, které lze libovolně zvolit (a tím kódovat informaci), a *kontrolní*, které jsou úplně určeny volbou informačních znaků. Například kód celkové kontroly parity délky 5 má 4 informační znaky a jeden (např. poslední), který je těmito čtyřmi určen, a je tedy kontrolní. Opakovací kód má jediný informační znak, ostatní znaky jsou kontrolní.

Jestliže můžeme z n znaků kódového slova libovolně volit k informačních znaků, máme předpis, který libovolné slovo délky k promění v kódové slovo délky n. Takový předpis se nazývá kódování informačních znaků:

7.1. Definice. Buď $K \subseteq T^n$ blokový kód délky n. Jestliže existuje prosté zobrazení φ množiny všech slov délky k na množinu všech kódových slov, $\varphi\colon T^k \to K$, řekneme, že kód K má k *informačních znaků* a $n - k$ *kontrolních znaků*. Zobrazení φ se nazývá *kódování informačních znaků*.

7.2. Příklady

7.2.1. Opakovací kód délky 5 má 1 informační znak a 4 kontrolní. Kódování informačních znaků

$\varphi\colon T \to K$

definujeme předpisem

$\varphi(v) = vvvv$.

7.2.2. *„Koktavý kód"*, ve kterém se každý znak dvakrát opakuje, má polovinu kontrolních a polovinu informačních znaků. Například v délce 6 je kódování informačních znaků zobrazení

$$\varphi\colon T^3 \to K\,,$$

definované předpisem

$$\varphi(v_1v_2v_3) = v_1v_1v_2v_2v_3v_3\,.$$

7.2.3. Kód „dva z pěti" (viz 4.1) nemá oddělené kontrolní a informační znaky. Neexistuje vůbec žádné prosté zobrazení z množiny $\{0, 1\}^k$ na množinu kódových slov, protože počet kódových slov je 10 a to není mocnina dvojky.

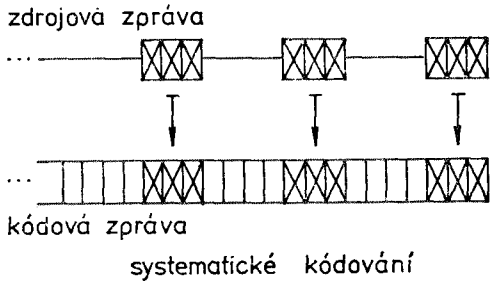
7.3. Poznámka. Pojem dekódování, o kterém jsme mluvili v minulém článku, není inverzní postup ke kódování informačních znaků. V případě bezpečnostních kódů vznikají dva typy dekódování: dekódování informačních znaků (tj. inverzní zobrazení $\varphi^{-1}\colon K \to T^k$ k zobrazení φ) a dekódování, odstraňující chyby (tj. zobrazení $\delta\colon T^n \to K$). Pokud mluvíme stručně o dekódování, máme vždy na mysli druhý pojem.

Nejjednodušší kódování informačních znaků je to, že je napíšeme jako prvních k znaků každého slova. Takové kódování se nazývá systematické, stejně jako kódy, pro které existuje:

7.4. Definice. Blokový kód $K \subseteq T^n$ se nazývá *systematický*, jestliže existuje číslo $k < n$ takové, že pro každé slovo $v_1 \dots v_k$ v T^k existuje právě jedno kódové slovo $v_1 \dots v_k v_{k+1} \dots v_n$. Kódování $\varphi\colon T^k \to K$, definované předpisem

$$\varphi(v_1 \dots v_k) = v_1v_2 \dots v_kv_{k+1} \dots v_n\,,$$

se nazývá *systematické*.



Obr. 14

7.5. Příklady

7.5.1. Opakovací kód je systematický ($k = 1$).

7.5.2. Kód celkové kontroly parity je systematický ($k = n - 1$).

7.5.3. „Koktavý" kód z předchozího příkladu není systematický.

7.6. Pozorování. Minimální vzdálenost d systematického kódu nemůže překročit počet $n - k$ kontrolních znaků o více než jeden. Stručněji,

$$d \leqq n - k + 1 .$$

To ukazuje, s jakým problémem se potýkáme při výběru vhodného kódu: na jedné straně chceme mít velkou kapacitu přenosu informace, tedy velkou hodnotu k, a na druhé straně chceme opravovat velký počet chyb, což vyžaduje velkou hodnotu d. Musíme ovšem volit kompromis.

Důkaz nerovnosti je snadný. Označme K daný kód. Zvolme libovolně $k - 1$ znaků $v_1 v_2 \ldots v_{k-1}$ a označme $K_0 \subseteq K$ množinu těch kódových slov, která mají prefix (1.5.2) $v_1 v_2 \ldots v_{k-1}$. Potom minimální vzdálenost d_0 kódu K_0 splňuje $d_0 \leqq n - (k - 1)$, protože libovolná dvě slova mají alespoň $k - 1$ společných znaků. Platí ovšem $d \leqq d_0$ (protože $K_0 \subseteq K$) a to jsme měli dokázat.

7.7. Úloha. Najděte dva typy kódů, pro které platí rovnost $d = n - k + 1$. Najděte kód, pro který $d + 1000 < n - k + 1$. [Kódy celkové kontroly parity a opakovací kódy na jedné straně, „koktavé" kódy na druhé straně.]

7.8. Závěr

Při výběru bezpečnostního kódu se snažíme o to, aby počet k informačních znaků byl velký (tedy, aby byla malá redundance) a také o to, aby minimální vzdálenost d kódu byla velká (tedy, aby kód byl schopen objevovat a opravovat hodně chyb). Protože tyto dva požadavky jsou rozporné, hledáme vhodný kompromis. Kromě toho je důležitá nejen možnost opravování chyb, ale i způsob, jak se opravy skutečně realizují. Proto se zajímáme o speciální třídy kódů, které při „rozumném" počtu informačních znaků mají jednoduchý algoritmus oprav „rozumného" počtu chyb. Takové třídy (např. Hammingovy kódy, Reedovy-Mullerovy kódy a BCH kódy) poznáme v dalších kapitolách.

Při volbě kódu bereme často v úvahu ještě jeden důležitý parametr: *informační poměr* R,

$$R = \frac{k}{n},$$

tedy poměr počtu informačních znaků ku počtu všech znaků. Jestliže potřebujeme přenášet zprávy rychlostí 100 znaků za sekundu a máme k dispozici kanál, který přenáší 200 znaků za sekundu, můžeme použít jen takové bezpečnostní kódy, které mají informační poměr $R \geqq 1/2$. Často jsou dána i jiná omezení technického rázu, např. omezení délky kódu n. Abychom takovým požadavkům vyhověli, je třeba známé kódy různým způsobem upravovat. Úpravám kódů se budeme věnovat v příštích kapitolách.

III. LINEÁRNÍ KÓDY

8. Binární lineární kódy

Fundamentální myšlenkou teorie bezpečnostních kódů je zavedení algebraických operací sčítání a násobení na abecedě T, takže T se stane tělesem a T^n n-rozměrným lineárním prostorem. Potom je blokový kód K délky n lineární, jestliže tvoří lineární podprostor prostoru T^n. Pro tyto kódy můžeme použít mocný algebraický aparát, který zjednodušuje jak popis kódu, tak i dekódování. Přitom běžné kódy jsou lineární, jak ukážeme.

V této kapitole se omezíme na binární kódy, tedy $T = \{0, 1\}$. Operace logického součtu a logického součinu, určené těmito tabulkami

+	0	1
0	0	1
1	1	0

.	0	1
0	0	0
1	0	1

tvoří na množině $\{0, 1\}$ těleso (jak přesně dokážeme v další kapitole). Všimněte si, že násobení se shoduje s obvyklým násobením čísel, zatímco pro sčítání platí

$$1 + 1 = 0,$$

takže $1 = -1$ (tj. odečítání je zde totéž co přičítání). Běžné binární kódy můžeme vyjádřit jako množinu všech řešení soustavy lineárních rovnic, používajících logický součet.

8.1. Příklady vyjádření binárního kódu rovnicemi

8.1.1. Kód celkové kontroly parity (4.6) je popsán jedinou rovnicí,

$$v_1 + v_2 + \ldots + v_n = 0.$$

Jestliže totiž slovo $v_1 v_2 \ldots v_n$ obsahuje sudý počet jedniček, pak tuto rovnici splňuje (protože $1 + 1 = 0$, takže i $1 + 1 + 1 + 1 = 0$, atd.). Naopak, pokud slovo $v_1 v_2 \ldots v_n$ obsahuje lichý počet jedniček, je $v_1 + v_2 + \ldots + v_n = 1$.

8.1.2. Opakovací kód (4.7) je popsán soustavou rovnic

$$\begin{aligned} v_1 + v_2 &= 0, \\ v_1 + v_3 &= 0, \\ &\vdots \\ v_1 + v_n &= 0. \end{aligned}$$

První rovnice totiž znamená, že $v_1 = v_2$ $(= -v_2)$, druhá $v_1 = v_3$, atd. Jediná řešení této soustavy rovnic jsou tedy slova $000\ldots0$ a $111\ldots1$.

8.1.3. „Koktavý" kód 7.2.2 je popsán třemi rovnicemi

$$v_1 + v_2 = 0,$$
$$v_3 + v_4 = 0,$$
$$v_5 + v_6 = 0.$$

8.2. Z lineární algebry víme, že všechna řešení homogenní soustavy lineárních rovnic o n neznámých tvoří lineární prostor, totiž podprostor prostoru T^n. To znamená, že součet $\boldsymbol{v} + \boldsymbol{w} = (v_1 + w_1, v_2 + w_2, \ldots, v_n + w_n)$ dvou řešení tvoří řešení a skalární násobek $t\boldsymbol{v} = (tv_1, tv_2, \ldots, tv_n)$ řešení $\boldsymbol{v}$ je také řešením. V případě $T = \{0, 1\}$ druhá podmínka nic neříká, ale první je důležitá. Vidíme, že kód celkové kontroly parity, opakovací kód i „koktavý" kód mají tu vlastnost, že součet dvou kódových slov je kódové slovo. Takové binární kódy nazýváme lineárními. Jsou to ty, které tvoří lineární podprostor prostoru $\{0, 1\}^n$.

Předpokládáme, že čtenář ví, co je to lineární (nebo vektorový) prostor nad tělesem T, a proto tyto pojmy v příští kapitole již jen stručně připomeneme. Potíž může být v tom, že zatímco zde pracujeme jen s konečnými tělesy, čtenář je možná zvyklý pracovat s reálnými nebo komplexními lineárními prostory (kde T je těleso reálných nebo komplexních čísel). Naštěstí je lineární algebra nad obecným tělesem naprosto analogická případu těchto dvou základních těles. Jednu výjimku tvoří ovšem fakt, že konečně-rozměrný vektorový prostor (nad konečným tělesem) je sám konečný. To jsme viděli v předchozích příkladech: množina všech řešení dané soustavy lineárních rovnic tam byla konečná, což může na první pohled překvapit ve srovnání s reálnými lineárními prostory. Druhou výjimku představuje definice polynomu — to podrobně vyložíme v V. kapitole.

8.3. Definice. Binární kód K se nazývá *lineární*, jestliže je podprostorem lineárního prostoru $\{0, 1\}^n$, tj. jestliže součet dvou kódových slov je kódové slovo. Je-li K podprostorem dimenze k, mluvíme o *lineárním* (n, k)*-kódu*.

8.4. Dimenze k je ovšem počet prvků libovolné báze. Například celý prostor $\{0, 1\}^n$ má dimenzi n, protože má bázi

$$\boldsymbol{e}_1 = 100\ldots0,$$
$$\boldsymbol{e}_2 = 010\ldots0,$$
$$\vdots$$
$$\boldsymbol{e}_n = 000\ldots1.$$

Kód celkové kontroly parity je $(n, n-1)$-kód: jeho bází je např.

$$\boldsymbol{b}_1 = 1000\ldots001,$$
$$\boldsymbol{b}_2 = 0100\ldots001,$$
$$\boldsymbol{b}_3 = 0010\ldots001,$$
$$\vdots$$
$$\boldsymbol{b}_{n-1} = 0000\ldots011.$$

Je totiž zřejmé, že a) každé slovo $\boldsymbol{b}_i$ má dvě jedničky, a tedy je kódovým slovem,

b) tato slova jsou lineárně nezávislá (po odtržení poslední jedničky tvoří totiž bázi prostoru $\{0, 1\}^{n-1}$) a c) každé slovo $v_1v_2 \dots v_n$ sudé parity je součtem těchto slov, totiž

$$v_1v_2 \dots v_n = \sum_{\substack{i=1 \\ v_i=1}}^{n-1} \boldsymbol{b}_i .$$

8.5. Počet informačních znaků. Každý lineární (n, k)-kód má k informačních, a tedy $n - k$ kontrolních, znaků. Zvolíme-li totiž bázi

$$\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_k$$

lineárního kódu K, potom každé kódové slovo $\boldsymbol{v}$ je tvaru

$$\boldsymbol{v} = u_1\boldsymbol{b}_1 + u_2\boldsymbol{b}_2 + \dots + u_k\boldsymbol{b}_k$$

pro právě jednu k-tici $u_1u_2 \dots u_k$. Dostáváme tedy kódování informačních znaků (7.1)

$$\varphi: \{0, 1\}^k \to K ,$$

definované předpisem

$$\varphi(u_1u_2 \dots u_k) = u_1\boldsymbol{b}_1 + u_2\boldsymbol{b}_2 + \dots + u_k\boldsymbol{b}_k .$$

8.6. Příklady

8.6.1. Kód celkové kontroly parity je $(n, n - 1)$-kód, a má tedy jediný kontrolní znak.

8.6.2. Opakovací kód je $(n, 1)$-kód; jeho bázi tvoří jediné slovo 111...11. Má tedy jediný informační znak.

8.6.3. „Koktavý" kód (7.2.2) je (6,3)-kód s bází

110000 001100 000011 .

8.6.4. Kód „dva z pěti" (4.1) není lineární např. proto, že nulové slovo 00000 není kódové.

8.7. Úloha. Pro kódy z 4.8 a 5.2.3 ověřte, že jsou lineární. Najděte jejich dimenzi a popište je rovnicemi.

8.8. Kontrolní matice kódu. Je-li binární kód popsán soustavou homogenních lineárních rovnic, potom matice $\boldsymbol{H}$ této soustavy se nazývá *kontrolní maticí* kódu. To znamená, že $\boldsymbol{H}$ je kontrolní maticí lineárního kódu délky n, jestliže platí: slovo $v_1v_2 \dots v_n$ je kódové, právě když splňuje soustavu rovnic

$$\boldsymbol{H}\begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} .$$

Jestliže označíme $\boldsymbol{v} = v_1 v_2 \ldots v_n$ řádkový vektor, platí tedy

$\boldsymbol{v} \in K$, právě když $\boldsymbol{H}\boldsymbol{v}^{\mathrm{T}} = \boldsymbol{o}^{\mathrm{T}}$

(kde index T označuje transpozici, která zde z řádku dělá sloupec).

Z příkladu 8.1 vidíme, že opakovací kód délky 5 má tuto kontrolní matici

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

V celé knize budeme zapisovat matice (stejně jako slova) bez oddělování prvků čárkami, prvky matice budeme oddělovat pouze mezerami.

Kód celkové kontroly parity délky 5 má kontrolní matici

$[1\ 1\ 1\ 1\ 1]$.

8.9. Závěr. Řadu binárních kódů lze považovat za lineární podprostory prostoru $\{0, 1\}^n$. Tyto kódy můžeme stručně popsat buď jejich bází, nebo jejich kontrolní maticí. První popis je výhodnější u kódů nižších dimenzí, druhý u kódů vyšších dimenzí – viz kódy opakovací, s bází $\{111\ldots1\}$, a kódy celkové kontroly parity s kontrolní maticí

$[1\ 1\ 1\ \ldots\ 1]$.

Dimenze kódu je rovna počtu informačních znaků.

Ukazuje se, že lineární struktura kódu hraje roli i při popisu dekódování. Než k tomu přistoupíme, věnujeme další článek tělesům, abychom mohli uvažovat i jiné kódy než jen binární.

9. Tělesa

Připomeneme základní algebraické struktury, se kterými budeme stále pracovat: grupy, tělesa a okruhy. Těleso je, zhruba řečeno, algebraická struktura se sčítáním a násobením, splňujícími všechny vlastnosti „prototypu" těles – množiny reálných čísel. To znamená, že počet axiómů, které těleso splňuje, je velký: tak velký, abychom mohli v každém tělese bezpečně počítat a nemuseli se zamýšlet nad tím, jaké úpravy ještě smíme udělat. Horší je ověřování, že určitá algebraická struktura všechny tyto axiómy splňuje. Naštěstí umíme popsat všechna konečná tělesa (a v teorii kódování hrají roli jen konečná tělesa). To znamená, že axiómy tělesa není třeba ověřovat, stačí znát jednoduchý popis všech těles. Konečné těleso je v podstatě určeno počtem svých prvků a to je mocnina prvočísla. V tomto článku se seznámíme s tělesy Z_p o p prvcích, kde p je prvočíslo. V dalších článcích, věnovaných Galoisovým tělesům, poznáme, jak vypadají tělesa s počtem prvků p^n.

9.1. Grupa. Grupa je množina G spolu s operací, přiřazující každým dvěma prvkům a a b množiny G prvek ab tak, že platí:

(i) ab je prvek množiny G;
(ii) *asociativní zákon*: $(ab)c = a(bc)$ (pro všechna $a, b, c \in G$);
(iii) existuje *neutrální prvek*, tj. prvek 1 množiny G takový, že $1a = a$ a $a1 = a$ (pro všechna $a \in G$);
(iv) každý prvek $a \in G$ má *inverzní prvek*, tj. prvek $a^{-1} \in G$ takový, že $aa^{-1} = 1$ a $a^{-1}a = 1$.

Jestliže je grupa *komutativní*, tj. splňuje $ab = ba$ (pro všechna $a, b \in G$), zapisuje se obvykle aditivně. To znamená, že operace grupy se označuje $a + b$ (místo ab) a nazývá se *sčítání*, neutrální prvek se označuje 0 a inverzní prvek k prvku a se označuje $-a$ a nazývá se *opačný prvek*.

Příkladem konečné grupy je množina $\{0, 1\}^n$ všech binárních slov délky n (s operací $+$, kterou jsme uvažovali v minulém článku). Neutrálním prvkem je $000\ldots0$. Každé slovo $v_1\ldots v_n$ je inverzní samo k sobě, protože $(v_1\ldots v_n) + (v_1\ldots v_n) = 00\ldots0$.

Podmínka inverzního prvku znamená, že v grupě můžeme dělit:

$$a : b = ab^{-1}.$$

Prvek $a : b$ má totiž tu vlastnost, že jeho součin s prvkem b je $(ab^{-1})\, b = a$. V aditivním zápisu to znamená, že v komutativní grupě můžeme odečítat:

$$a - b = a + (-b).$$

9.2. Těleso. Těleso je množina T spolu se dvěma operacemi $+$ a $.$ takovými, že platí:

(i) operace $+$ vytváří na množině T komutativní grupu s neutrálním prvkem 0;
(ii) operace $.$ vytváří na množině $T - \{0\}$ všech nenulových prvků komutativní grupu s neutrálním prvkem 1;
(iii) *distributivní zákon*: $a(b + c) = ab + ac$ (pro všechna $a, b, c \in T$).

Všimněte si, že druhá podmínka určuje, že $ab \neq 0$ kdykoli $a \neq 0$ a $b \neq 0$, ale neříká nic o součinech s nulou. Platí ovšem $0 \,.\, a = 0$ pro všechna $a \in T$. (Důkaz: rovnici $a = a + 0$ násobíme prvkem a, tj. $a \,.\, a = (a + 0)\, a = a \,.\, a + 0 \,.\, a$, a odečteme $a \,.\, a$).

Axiómy tělesa jsme tedy stručně vyjádřili pomocí pojmu grupy. Je ale dobře si uvědomit, kolik podmínek skutečně klademe na těleso:

9.3. Těleso je množina T spolu s operacemi, přiřazujícími každým dvěma prvkům a a b množiny T prvky $a + b$ a ab tak, že platí:

(T1) $a + b$ a ab jsou prvky množiny T;
(T2) asociativní zákony: $(a + b) + c = a + (b + c)$ a $(ab)\, c = a(bc)$ (pro všechna $a, b, c \in T$);
(T3) komutativní zákony: $a + b = b + a$ a $ab = ba$ (pro všechna $a, b \in T$);
(T4) distributivní zákon: $a(b + c) = ab + ac$ (pro všechna $a, b, c \in T$);
(T5) existují neutrální prvky 0 a 1, tj. prvky množiny T takové, že $a + 0 = a$ a $a1 = a$ (pro všechna $a \in T$);
(T6) každý prvek a množiny T má opačný prvek, tj. prvek $-a$ takový, že $a - a = 0$;

(T7) každý prvek $a \neq 0$ množiny T má inverzní prvek, tj. prvek a^{-1} takový, že $aa^{-1} = 1$.

V tělese tedy máme operace sčítání, odčítání, násobení a dělení (nenulovými prvky), splňující asociativní, komutativní a distributivní zákony. Proto můžeme provádět všechny algebraické úpravy, na které jsme zvyklí z tělesa reálných čísel.

9.4. Tělesa Z_p. Pro každé prvočíslo p definujeme na množině

$$\{0, 1, 2, \ldots, p-1\}$$

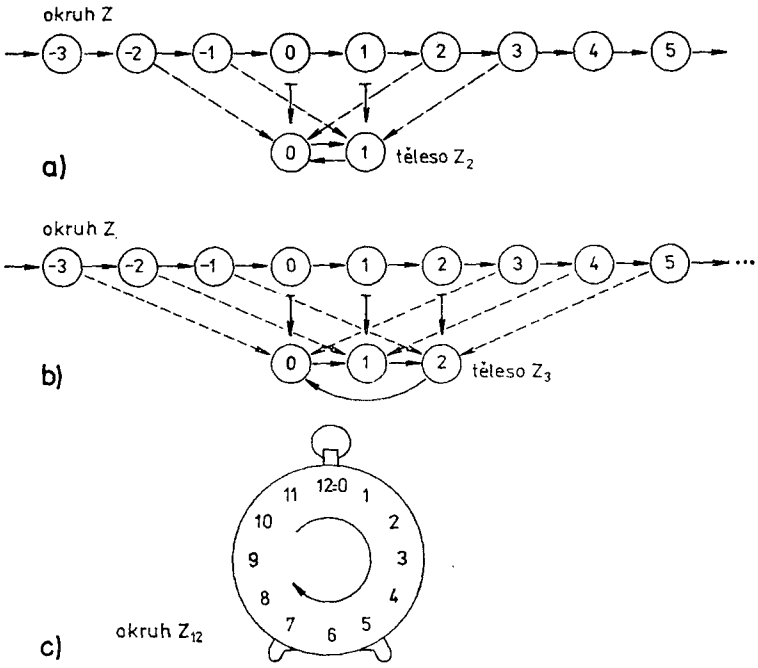
operace $\oplus$ a $\otimes$: je to obvyklé sčítání a násobení čísel až na to, že pokud výsledek „překročí" naši množinu (tj. je větší než $p-1$), odečteme od něj číslo p. Podrobněji, pro každá dvě čísla $a, b = 0, 1, \ldots, p-1$ položíme

$$a \oplus b = \begin{cases} a + b, & \text{pokud } a + b \leqq p - 1, \\ a + b - p, & \text{pokud } a + b \geqq p. \end{cases}$$

Při násobení odečítání opakujeme, je-li to nutné, tedy odečítáme číslo p k-krát:

$$a \otimes b = ab - kp, \quad k = 0, 1, 2, \ldots,$$

kde k volíme tak, aby výsledek $ab - kp$ byl některým z čísel $0, 1, \ldots, p-1$. (Takové k je jediné.)



Obr. 15

Například Z_2 má prvky 0 a 1, $\otimes$ je obvyklé násobení čísel, ale $1 \oplus 1 = 1 + 1 - 2 = 0$. Operace $\oplus$ a $\otimes$ jsou tedy shodné s operacemi, zavedenými

v minulém článku. (Obecně označujeme operace v tělese Z_p symboly + a ., pokud nemůže dojít k omylu.) V Z_3 máme prvky 0, 1 a 2. Pro sčítání platí $1 \oplus 2 = 0$ a $2 \oplus 2 = 1$. Pro násobení, $2 \otimes 2 = 1$. Dostáváme tyto tabulky:

$\oplus$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\otimes$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

9.5. Úloha. Napište tabulky sčítání a násobení tělesa Z_5.

9.6. Věta. *Pro každé prvočíslo p je Z_p tělesem.*

Důkaz. Musíme ověřit podmínky, formulované v 9.3. Platí (T1), protože tak jsme právě upravili obvyklé operace + a × (sčítání a násobení čísel). Dále platí (T2), (T3) a (T4), tedy úprava operací „nepokazila" asociativní, komutativní a distributivní zákony (platné pro obvyklé operace s celými čísly). Ověřme to např. v případě komutativního zákona: víme, že $a + b = b + a$. Přitom buďto platí $a \oplus b = a + b$, a potom ovšem také $b \oplus a = b + a$, a tím jsme ověřili, že $a \oplus b = b \oplus a$; anebo $a \oplus b = a + b - p$. V tomto případě je $a + b \geqq p$, a tedy i $b + a \geqq p$, takže $b \oplus a = b + a - p$, a tím jsme ověřili, že opět $a \oplus b = b \oplus a$. Analogická je situace se všemi podmínkami (T2), (T3) a (T4). Zřejmě platí (T5): neutrální prvky jsou 0 (pro $\oplus$) a 1 (pro $\otimes$).

Ověřme (T6). Opačným prvkem k 0 je ovšem 0, protože $0 + 0 = 0$. Pro každý jiný prvek $a = 1, 2, \ldots, p - 1$ platí $p - a = 1, 2, \ldots, p - 1$ a dále

$$a \oplus (p - a) = a + p - a - p = 0 .$$

Tedy opačným prvkem libovolného prvku $a \neq 0$ je prvek $p - a$.

Ověřme (T7). Inverzním prvkem k 1 je ovšem 1, protože $1 \otimes 1 = 1$. Další inverzní prvky budeme hledat matematickou indukcí: jestliže každé z čísel $1, 2, \ldots, a - 1$ má inverzní prvek, najdeme inverzní prvek i k prvku a. Dělíme-li číslo p číslem a, dostaneme tvar

$$p = aq + r ,$$

kde q je podíl a r ($< a$) je zbytek. Protože je p prvočíslo a platí $1 < a < p$, není a dělitelem čísla p, takže $r \neq 0$. Podle indukčního předpokladu má číslo r inverzní prvek, tedy existuje $s = 1, \ldots, p - 1$ takové, že

$$r \otimes s = 1 .$$

Protože $aq = p - r$, je $a \otimes q$ opačným prvkem k prvku r, a tedy

$$a \otimes q \otimes s = -r \otimes s = -1 .$$

Opačný prvek k prvku $q \otimes s$ (tj. číslo $p - q \otimes s$) je hledaným inverzním prvkem:

$$a \otimes (-q \otimes s) = -a \otimes q \otimes s = 1 .$$

9.7. Okruhy. Dokázali jsme, že Z_p je tělesem pro každé prvočíslo p; a co když p není prvočíslo? Je Z_6 těleso?

Operace $\oplus$ a $\otimes$ můžeme ovšem zavést na množině $\{0, 1, \ldots, p-1\}$ i v případě, že p není prvočíslo. Například pro Z_6 dostáváme rovnost

$$2 \otimes 3 = 6 - 6 = 0,$$

a proto není Z_6 tělesem: prvek 2 ani prvek 3 nemá inverzní prvek. (Kdyby totiž některý prvek x splňoval $x \otimes 2 = 1$, potom by z rovnosti $2 \otimes 3 = 0$ plynulo $3 = x \otimes 2 \otimes 3 = 0$, a to je spor.) Axióm (T7) tedy v Z_p neplatí, pokud není p prvočíslem: žádné číslo $2, 3, \ldots, p-1$, které dělí číslo p, nemá inverzní prvek. Ostatní axiómy však platí. O tom se můžete přesvědčit, jestliže projdete důkaz předchozí věty. Algebraická struktura, splňující axiómy (T1) až (T6), se nazývá *okruh* (podrobněji: komutativní okruh s jednotkou). Příkladem je okruh Z celých čísel (který není tělesem, protože žádný prvek, kromě 1 a -1, nemá inverzní prvek). Každé těleso je ovšem okruhem.

9.7.1. Úloha. Ověřte, že v okruhu Z_6 mají prvky 1 a 5 inverzní prvky, ale 2, 3 a 4 nemají inverzní prvky. Obecněji dokažte, že v okruhu Z_p má číslo $x = 2, 3, \ldots, p-1$ inverzní prvek, právě když je x nesoudělné s číslem p.

Návod. Je-li x nesoudělné, postupujte jako v důkazu věty 9.6. Pokud je soudělné, platí $x = ay$ a $p = az$, takže $a \otimes z = 0$, a proto nemá prvek $a \otimes y$ inverzní prvek.

9.8. Faktorový okruh modulo p. Okruhy Z_p můžeme také popsat jako faktorové okruhy okruhu Z celých čísel. K tomu potřebujeme zavést pojem třídy modulo p. Například pro $p = 2$ je třídou čísla a modulo 2 množina všech čísel x, které se od a liší o násobek dvojky, tedy čísel $x = a + 2t$. Třídy označujeme lomenými závorkami:

$$[a] = \{a + 2t \mid t \text{ je libovolné celé číslo}\}.$$

Všimněte si, že třídy modulo 2 jsou vlastně jen dvě:

$$[0] = \{2t \mid t \in Z\}$$

je množina všech sudých čísel (a platí $[0] = [2] = [-2] = [4] = \ldots$) a

$$[1] = \{1 + 2t \mid t \in Z\}$$

je množina všech lichých čísel. Obecněji, zvolíme-li v okruhu T libovolný prvek p, potom *třídou modulo p* daného prvku a v T nazýváme množinu všech prvků, které se od a liší o násobek prvku p, tj.

$$[a] = \{a + pt \mid t \text{ je libovolný prvek okruhu } T\}.$$

Snadno ověříte, že pro každé dva prvky a a a' platí buď

(i) p je dělitelem rozdílu $a - a'$, a potom $[a] = [a']$,

nebo

(ii) p není dělitelem rozdílu $a - a'$, a potom třída $[a]$ nemá žádný společný prvek se třídou $[a']$.

Definujeme *faktorový okruh* okruhu T modulo p: jeho prvky jsou právě všechny různé třídy $[a]$, kde $a \in T$. Třídy sčítáme a násobíme takto:

$$[a] + [a'] = [a + a'], \quad [a][a'] = [aa'].$$

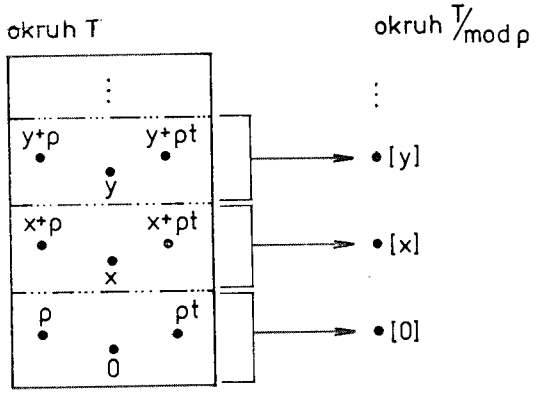
Ověření axiómů (T1) až (T6) je snadné. Faktorový okruh označujeme

$$T/\mathrm{mod}\ p\,.$$

Například okruh

$$Z/\mathrm{mod}\ 2$$

má dva prvky: třídu $[0]$ všech sudých čísel a třídu $[1]$ všech lichých čísel. Platí např. $[1] + [1] = [2] = [0]$. Vidíme, že až na psaní lomených závorek jde právě o okruh (dokonce těleso) Z_2. Obecněji, okruh $Z/\mathrm{mod}\ p$ pro každé číslo $p = 1, 2, 3, \ldots$ má prvky $[0], [1], \ldots, [p-1]$. (Platí totiž $[p] = [0], [p+1] = [1]$, atd. Dále $[-1] = [p-1], [-2] = [p-2]$, atd.) Až na zápis lomených závorek jde o okruh Z_p.



Obr. 16

9.8.1. Úloha. Ověřte podrobně, že okruh $Z/\mathrm{mod}\ 3$ má právě tři prvky a napište tabulku sčítání a násobení v tomto okruhu. Porovnejte s tabulkou tělesa Z_3 (9.4). Obecněji ověřte, že okruhy Z_p a $Z/\mathrm{mod}\ p$ jsou shodné až na označení prvků (i místo $[i]$).

9.8.2. Označení. Jestliže dva prvky a a a' mají stejnou třídu modulo p, říkám e, že a je *kongruentní s* a' *modulo* p a píšeme

$$a \equiv a' \ (\mathrm{mod}\ p)\,.$$

Například

$$0 \equiv 6 \ (\mathrm{mod}\ 2)$$

a obecněji, pro dvě čísla a a a' platí $a \equiv a' \ (\mathrm{mod}\ 2)$, právě když jsou obě sudá nebo obě lichá. Platí

$$a \equiv a' \ (\mathrm{mod}\ p)\,, \quad \text{právě když rozdíl } a - a' \text{ je násobkem prvku } p\,.$$

9.8.3. Úloha. Ověřte, že $T/\mathrm{mod}\ 0$ je (až na označení) okruh shodný s okruhem T. Naproti tomu, je-li **R** těleso reálných čísel, má okruh $\mathbf{R}/\mathrm{mod}\ 2$ jediný prvek. Platí to pro každý faktorový okruh tělesa **R**? A pro každý faktorový okruh každého tělesa? [Rozlišujte 0 a prvky $p \neq 0$!]

9.9. Lineární prostory. Připomeňme ještě pojem lineárního prostoru nad daným tělesem T. To je množina L spolu s operacemi $+$ (sčítání) a $.$ (skalární násobení) takovými, že

(i) operace $+$ vytváří na množině L komutativní grupu;

(ii) operace $.$ každému prvku x v L a každému skaláru t v T přiřazuje prvek tx v L;

(iii) pro libovolné prvky x a y v L a libovolné skaláry s, t v T platí

(L1) $t(x + y) = tx + ty$;

(L2) $(st)\,x = s(tx)$;

(L3) $(s + t)\,x = sx + tx$;

(L4) $1x = x$.

Například lineární prostor nad tělesem $Z_2 = \{0, 1\}$ je totéž jako komutativní grupa L, splňující $x + x = 0$ pro všechna $x \in L$. Skutečně, operace $.$ je zde triviální: platí $1x = x$ (podle posledního axiómu) a $0x = 0$ (protože $0x = (1 - 1)\,x = x - x = 0$). Z (L3) plyne $0 = (1 + 1)\,x = x + x$.

Důležitým příkladem lineárního prostoru je prostor $L = T^n$ všech slov $t_1 t_2 \ldots t_n$ délky n v abecedě T. Sčítání a skalární násobení provádíme po složkách. Každý konečně-dimenzionální lineární prostor L je izomorfní s prostorem T^n (kde n je dimenze prostoru L), což znamená, že algebraická struktura prostorů L a T^n je shodná až na případné označení prvků.

9.10. Závěr. Pro každé prvočíslo $p = 2, 3, 5, 7, \ldots$ existuje těleso Z_p o p prvcích. Můžeme je popsat jako faktorový okruh okruhu Z celých čísel (pak má prvky $[0], [1], \ldots, [p - 1]$) anebo přímo (s označením prvků $0, 1, \ldots, p - 1$). Později ukážeme, že každé těleso o p prvcích je, až na označení prvků, shodné s tělesem Z_p. Kromě toho uvidíme, že existují tělesa o p^n prvcích, kde p je prvočíslo a $n = 1, 2, 3, \ldots$.

V teorii lineárních kódů vycházíme z toho, že na abecedě zpráv jsou dány operace tělesa. To znamená, že abeceda může mít

$$2, 3, 4, 5, \quad 7, 8, 9, \quad 11, \quad 13 \ldots$$

prvků, ale nemůže mít $6, 10, 12, \ldots$ prvků (protože tato čísla nejsou mocninami prvočísel). Toto omezení není příliš tragické; jednak proto, že zdaleka nejvýznamnější kódy jsou kódy binární, a jednak proto, že při vyšším počtu znaků abecedy se můžeme dohodnout, že některé nepoužijeme. (Například z 12-znakové abecedy vynecháme jeden znak.)

10. Generující matice

Ukázali jsme, že řada binárních kódů tvoří lineární podprostory prostoru Z_2^n. Nyní se budeme zabývat lineárními kódy nad libovolnou abecedou T, na které je dána struktura tělesa. Například ternárními kódy, tj. kódy nad tělesem Z_3.

Pro každé těleso T tvoří všechna slova délky n lineární prostor T^n: součet dvou slov $v_1v_2 \dots v_n + w_1w_2 \dots w_n$ je slovo $u_1u_2 \dots u_n$, kde $u_i = v_i + w_i$, a násobek slova skalárem, $t(v_1v_2 \dots v_n)$ pro $t \in T$, je slovo $u_1u_2 \dots u_n$, kde $u_i = tv_i$. Lineární (n, k)-kódy zavádíme jako podprostory tohoto lineárního prostoru. Ty můžeme zadat jejich generující maticí (neboli bází) anebo kontrolní maticí, jak uvedeme v čl. 11. Dimenze k je rovna počtu informačních znaků. Uvidíme úzkou souvislost mezi generující maticí a kódováním informačních znaků.

Mezi lineárními kódy jsou dva extrémní případy (které jsou nepoužitelné). Prvním je nulový prostor (dimenze $k = 0$), který obsahuje jen nulové slovo, a nemůže tedy přenášet žádnou informaci. Další je celý prostor (dimenze $k = n$), který nemá žádnou redundanci, a tedy nemůže sloužit k zabezpečení informace. Tyto kódy nazýváme triviálními.

10.1. Definice. *Lineárním kódem* rozumíme lineární podprostor K prostoru T^n (kde T je konečné těleso). Podrobněji mluvíme o lineárním (n, k)-*kódu*, je-li k dimenze podprostoru K. Pokud $k = 0$ nebo $k = n$, nazýváme K *triviálním kódem*.

10.2. Jinými slovy, blokový kód K délky n v abecedě T je lineární, jestliže pro každá dvě kódová slova $v_1v_2 \dots v_n$ a $w_1w_2 \dots w_n$ je i součet $(v_1v_2 \dots v_n) + (w_1w_2 \dots w_n)$ kódovým slovem a dále násobky $t(v_1v_2 \dots v_n)$ jsou kódová slova. Kód je *q-znakový*, jestliže abeceda T má počet znaků q.

Všimněte si, že lineární (n, k)-kód má celkem q^k kódových slov, protože kódová slova můžeme vyjádřit jejich k souřadnicemi ve zvolené bázi a každá souřadnice má q možných hodnot.

10.3. Generující matice. Lineární (n, k)-kód s $k \neq 0$ je určen svou (libovolnou) bází $\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_k$. Napíšeme-li těchto k slov pod sebe, vznikne matice

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \dots \\ \boldsymbol{b}_k \end{bmatrix}$$

o k řádcích a n sloupcích. Ta se nazývá *generující matice* daného kódu.

Jinými slovy, matice $\boldsymbol{G}$ typu (k, n) je generující maticí lineárního kódu, jestliže

a) každý její řádek je kódovým slovem,

b) každé kódové slovo je lineární kombinací řádků,

c) řádky jsou lineárně nezávislé, takže hodnost matice $\boldsymbol{G}$ je rovna k.

10.4. Příklady

10.4.1. Binární kód celkové kontroly parity (4.6) má v délce 4 tuto generující matici:

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Každý řádek této matice má sudou paritu a naopak, slova sudé parity jsou právě součty řádků matice $\boldsymbol{G}$ (viz 8.4).

10.4.2. Označme K ternární kód (s abecedou $Z_3 = \{0, 1, 2\}$) délky šest, ve kterém třetí znak slouží ke kontrole prvních dvou: $a_3 = a_1 + a_2$, a šestý znak slouží ke kontrole čtvrtého a pátého znaku: $a_6 = a_4 + a_5$. Jde tedy o kód, popsaný rovnicemi

$$a_1 + a_2 + 2a_3 = 0\,,$$
$$a_4 + a_5 + 2a_6 = 0\,.$$

(V tělese Z_3 platí $1 + 2 = 0$, takže $-1 = 2$. To znamená, že odečítání je shodné s přičítáním dvojnásobku). Tento kód má 4 informační znaky: a_1, a_2, a_4 a a_5. Jde tedy o (6, 4)-kód. Jeho generující matici získáme např. tak, že tyto čtyři znaky necháme probíhat hodnotami 1000, 0100, 0010 a 0001:

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix}.$$

10.4.3. Opakovací 5-znakový kód (v abecedě Z_5) délky 6 má generující matici

$$\boldsymbol{G} = (1\ 1\ 1\ 1\ 1\ 1)\,.$$

Každé slovo tohoto kódu je skalárním násobkem slova 111111.

10.5. Kódování informačních znaků. Každý lineární (n, k)-kód má k informačních znaků a $n - k$ znaků kontrolních. Jestliže totiž zvolíme generující matici $\boldsymbol{G}$, tj. bázi kódu $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k$, je každé kódové slovo určeno svými k souřadnicemi v této bázi. To znamená, že zobrazení

$$\varphi\colon T^k \to K\,,$$

definované předpisem

$$\varphi(u_1u_2 \ldots u_k) = u_1\boldsymbol{b}_1 + u_2\boldsymbol{b}_2 + \ldots + u_k\boldsymbol{b}_k \quad (u_1u_2 \ldots u_k \in T^k)$$

nebo maticově

$$\varphi(\boldsymbol{u}) = \boldsymbol{u}\boldsymbol{G} \quad (\boldsymbol{u} \in T^k)\,,$$

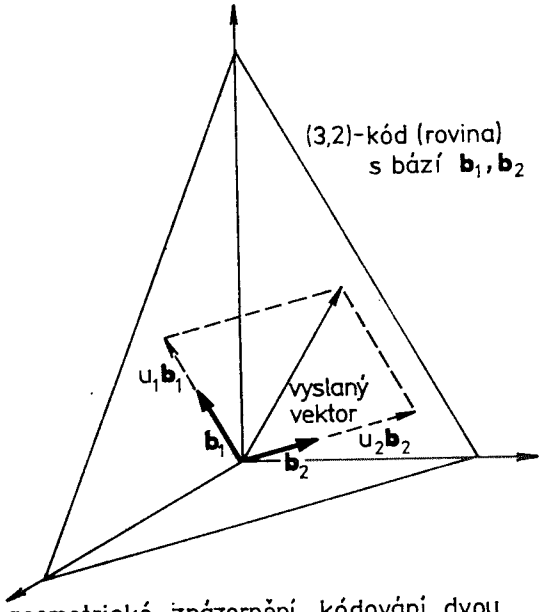
je prosté zobrazení množiny všech slov délky k na množinu všech kódových slov. Je to tedy kódování informačních znaků, viz 7.1. (Zobrazení φ je ovšem navíc lineární. Každá generující matice tedy definuje lineární kódování informačních znaků. Naopak, každé lineární kódování φ je určeno generující maticí: její řádky jsou $\varphi(100\ldots0)$, $\varphi(010\ldots0)$, ..., $\varphi(000\ldots1)$.)

Například ternární kód 10.4.2 má kódování informačních znaků $u_1u_2u_3u_4$ (v Z_3^4) definované předpisem

$$\varphi(u_1u_2u_3u_4) = [u_1\ u_2\ u_3\ u_4] \begin{bmatrix} 1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix} = [u_1\ u_2\ a\ u_3\ u_4\ b]\,,$$

kde $a = 2u_1 + 2u_2$ a $b = 2u_3 + 2u_4$.

Kódování informačních znaků opakovacího kódů je zobrazení
$\varphi(u) = u(111\ldots11) = uuu\ldots uu$.



Obr. 17

10.6. Poznámka. Použili jsme zde jednoduché pravidlo pro násobení matice $\boldsymbol{A}$ řádkem $\boldsymbol{u} = u_1u_2\ldots u_k$: označíme-li řádky matice $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_k$, platí

$$\boldsymbol{uA} = [u_1\ u_2\ \ldots\ u_k]\begin{bmatrix}\boldsymbol{a}_1\\ \boldsymbol{a}_2\\ \ldots\\ \boldsymbol{a}_k\end{bmatrix} = u_1\boldsymbol{a}_1 + u_2\boldsymbol{a}_2 + \ldots + u_k\boldsymbol{a}_k\,.$$

Podobně při násobení matice $\boldsymbol{B}$ sloupcem: označíme-li sloupce matice $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n$, platí

$$\boldsymbol{B}\begin{bmatrix}v_1\\ v_2\\ \ldots\\ v_n\end{bmatrix} = v_1\boldsymbol{b}_1 + v_2\boldsymbol{b}_2 + \ldots + v_n\boldsymbol{b}_n\,.$$

10.7. Systematické kódy. Systematické kódy (viz 7.4) jsou ty lineární kódy, které mají generující matici tvaru

$$\boldsymbol{G} = [\boldsymbol{E} \mid \boldsymbol{B}]\,,$$

kde $\boldsymbol{E}$ je jednotková matice řádu k. Potom totiž platí

$$[u_1\ u_2\ \dots\ u_k]\begin{bmatrix} 1\ 0\ 0\ \dots\ 0\ b_{11}\ \dots\ b_{1n-k} \\ 0\ 1\ 0\ \dots\ 0\ b_{21}\ \dots\ b_{2n-k} \\ \dots\dots\dots\dots\dots\dots\dots \\ 0\ 0\ 0\ \dots\ 1\ b_{k1}\ \dots\ b_{kn-k} \end{bmatrix} = [u_1\ u_2\ \dots\ u_k\ v_{k+1}\ \dots\ v_n],$$

kde $\boldsymbol{v} = v_{k+1} \dots v_n$ je vektor $\boldsymbol{v} = \boldsymbol{uB}$. Například kód celkové kontroly parity je systematický a také opakovací kód je systematický.

Ukážeme, že každý lineární kód, který není systematický, můžeme na systematický kód upravit pouhou permutací pořadí znaků v kódových slovech. Přesněji: říkáme, že dva blokové kódy K a K' délky n jsou *ekvivalentní*, jestliže existuje permutace $[\pi_1, \pi_2, \dots, \pi_n]$ čísel $1, 2, \dots, n$ taková, že platí

$$v_1 v_2 \dots v_n \in K, \quad \text{právě když} \quad v_{\pi_1} v_{\pi_2} \dots v_{\pi_n} \in K'$$

(pro všechna slova $v_1 v_2 \dots v_n$ v T^n).

10.8. Příklad. Označme K' ternární $(6, 4)$-kód všech slov $a_1 a_2 a_3 a_4 a_5 a_6$ takových, že a_5 je součtem prvních dvou znaků a a_6 je součtem dalších dvou znaků:

$$a_5 = a_1 + a_2,$$
$$a_6 = a_3 + a_4.$$

Tento kód je ekvivalentní s kódem K v 10.4.2: stačí v kódu K prvek a_3 přemístit na páté místo a prvky a_4, a_5 přesunout o místo kupředu. Přesněji, pro permutaci

$$\pi = [1, 2, 4, 5, 3, 6]$$

čísel $1, \dots, 6$ platí:

$$a_1 a_2 a_3 a_4 a_5 a_6 \in K, \quad \text{právě když} \quad a_{\pi_1} a_{\pi_2} a_{\pi_3} a_{\pi_4} a_{\pi_5} a_{\pi_6} \in K'.$$

10.9. Věta. *Každý lineární kód je ekvivalentní se systematickým lineárním kódem.*

Důkaz. Pro každý lineární (n, k)-kód K můžeme najít generující matici $\boldsymbol{G}$ a ta má hodnost k, protože její řádky (tj. báze prostoru K) jsou lineárně nezávislé. Odtud plyne, že matice $\boldsymbol{G}$ má k lineárně nezávislých sloupců. (To je např. dokázáno v knize [9], tvrzení 2.12.5.) Jestliže dokonce první k sloupce jsou lineárně nezávislé, můžeme matici $\boldsymbol{G}$ elementárními úpravami řádků převést na matici $\boldsymbol{G}_1 = [\boldsymbol{E} \mid \boldsymbol{B}]$. Přitom $\boldsymbol{G}_1$ je opět generující maticí kódu K: každý její řádek $\boldsymbol{v}$ jsme získali lineární kombinací řádků matice $\boldsymbol{G}$, takže $\boldsymbol{v}$ je kódové slovo, a přitom má matice $\boldsymbol{G}_1$ hodnost $k = \dim K$.

Jestliže však jsou první k sloupce lineárně závislé, provedeme takovou permutaci $\pi_1, \pi_2, \dots, \pi_n$ sloupců matice $\boldsymbol{G}$, která ji převede na matici $\boldsymbol{G}'$ s prvními k sloupci lineárně nezávislými. Elementárními úpravami řádků převedeme matici $\boldsymbol{G}'$ na tvar $\boldsymbol{G}_1' = [\boldsymbol{E} \mid \boldsymbol{B}]$. Lineární kód K', generovaný maticí $\boldsymbol{G}'$ (nebo $\boldsymbol{G}_1'$), je systematický. Přitom vznikl z kódu K permutací $[\pi_1, \pi_2, \dots, \pi_n]$ pořadí znaků kódových slov.

10.10. Příklady. Důkaz předchozí věty dává návod, jak hledat ekvivalentní systematický kód: v generující matici přehodíme sloupce tak, aby první k sloupce byly lineárně nezávislé.

10.10.1. Koktavý kód (7.2.2) má generující matici

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Není systematický. Přehodíme-li druhý a pátý sloupce, tj. provedeme-li permutaci $\pi = [1, 5, 3, 4, 2, 6]$, dostaneme generující matici

$$\boldsymbol{G}' = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

systematického kódu K', ekvivalentního s koktavým kódem. Kódová slova kódu K' jsou všechna slova

$$u_1 u_2 u_3 u_3 u_1 u_2 .$$

Přehodíme-li druhý a třetí řádek matice $\boldsymbol{G}'$, dostaneme jinou generující matici kódu K':

$$\boldsymbol{G}_1' = [\boldsymbol{E} \mid \boldsymbol{B}] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right].$$

10.10.2. V abecedě Z_5 je dána generující matice

$$\boldsymbol{G} = \begin{bmatrix} 1 & 4 & 1 & 1 & 1 \\ 2 & 4 & 0 & 0 & 1 \\ 0 & 2 & 1 & 1 & 0 \end{bmatrix}.$$

Je výsledný kód systematický? To zjistíme elementárními úpravami:

$$\begin{bmatrix} 1 & 4 & 1 & 1 & 1 \\ 2 & 4 & 0 & 0 & 1 \\ 0 & 2 & 1 & 1 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & 4 & 1 & 1 & 1 \\ 0 & 1 & 3 & 3 & 4 \\ 0 & 2 & 1 & 1 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & 4 & 1 & 1 & 1 \\ 0 & 1 & 3 & 3 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Kód není systematický, ale permutací $[1, 2, 5, 4, 3]$, která zaměňuje třetí a pátý sloupec, získáme ekvivalentní systematický kód. Provádíme elementární úpravy (odspoda) na nové matici $\boldsymbol{G}'$:

$$\boldsymbol{G}' = \begin{bmatrix} 1 & 4 & 1 & 1 & 1 \\ 0 & 1 & 4 & 3 & 3 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & 4 & 0 & 1 & 1 \\ 0 & 1 & 0 & 3 & 3 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \sim \left[\begin{array}{ccc|cc} 1 & 0 & 0 & 4 & 4 \\ 0 & 1 & 0 & 3 & 3 \\ 0 & 0 & 1 & 0 & 0 \end{array}\right].$$

Vidíme, že výsledný systematický kód sestává ze všech slov

$$u_1 u_2 u_3 v v ,$$

kde $v = 4u_1 + 3u_2$.

10.11. Úlohy. Najděte systematický kód, ekvivalentní s daným lineárním kódem.

10.11.1. K: 00000, 11100, 11110, 11101, 11111, 00010, 00001 a 00011.

10.11.2. Ternární kód s generující maticí

$$\begin{bmatrix} 2 & 2 & 0 & 1 & 1 \\ 2 & 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 0 & 0 \end{bmatrix}.$$

10.12. Hammingova váha. Hammingova váha slova $v_1 v_2 \ldots v_n$ je počet nenulových znaků ve slově. Označujeme ji $\|\boldsymbol{v}\|$:

$\|v_1 v_2 \ldots v_n\|$ = velikost množiny $\{i \mid v_i \neq 0\}$.

Například

$\|10021\| = 3$.

10.13. Pozorování. Každý binární lineární kód obsahuje buď jen slova sudé váhy, nebo má stejný počet slov sudé i liché váhy.

Skutečně, jestliže kód K obsahuje $p > 0$ slov liché váhy a q slov sudé váhy, ukážeme, že $p = q$. Zvolme slovo $\boldsymbol{v}$ liché váhy. Každému kódovému slovu $\boldsymbol{w}$ liché váhy odpovídá kódové slovo $\boldsymbol{v} + \boldsymbol{w}$ sudé váhy (a přitom z $\boldsymbol{w} \neq \boldsymbol{w}'$ plyne $\boldsymbol{v} + \boldsymbol{w} \neq \boldsymbol{v} + \boldsymbol{w}'$), takže platí $p \geqq q$. Analogicky, každému kódovému slovu $\boldsymbol{u}$ sudé váhy odpovídá kódové slovo $\boldsymbol{v} + \boldsymbol{u}$ liché váhy, takže $q \geqq p$.

10.14. Úlohy

10.14.1. Kolik slov sudé váhy a kolik slov liché váhy má binární kód s generující maticí

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} ?$$

10.14.2. Ověřte, že pro každý binární lineární kód a každé číslo $i = 1, 2, \ldots, n$ je počet kódových slov $\boldsymbol{v}$ s vlastností $v_i = 1$ buď nulový, nebo polovina počtu všech kódových slov.

11. Kontrolní matice

Lineární kód můžeme zadat jeho kontrolní maticí, jak jsme to viděli u binárních kódů (8.8). Jde o významný popis jak při objevování, tak při opravování chyb. Kontrolní matice generuje tzv. duální kód.

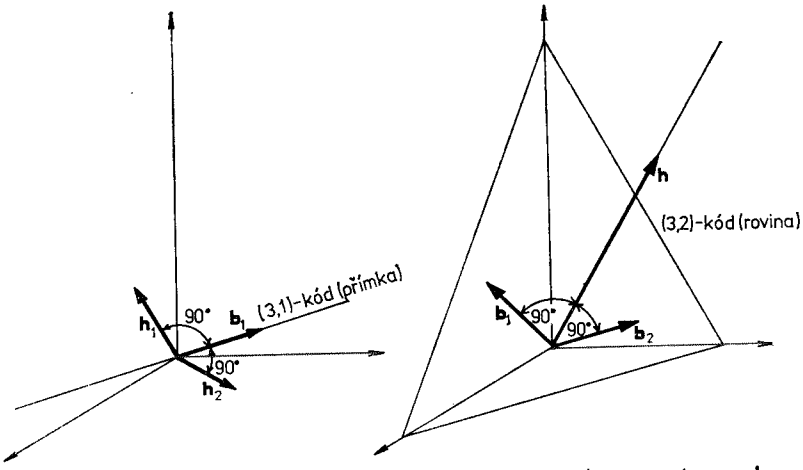
11.1. Definice. *Kontrolní matice* lineárního kódu K je taková matice $\boldsymbol{H}$ z prvků abecedy T, pro kterou platí: slovo $\boldsymbol{v} = v_1 v_2 \ldots v_n$ (v T^n) je kódové, právě když splňuje

podmínku

$$\boldsymbol{H}\begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}.$$

Stručněji:

$\boldsymbol{v} \in K$, právě když $\boldsymbol{H}\boldsymbol{v}^{\mathrm{T}} = \boldsymbol{o}^{\mathrm{T}}$.



a) kontrolní matice má dva sloupce $\mathbf{h}_1$ a $\mathbf{h}_2$

b) kontrolní matice má jeden sloupec $\mathbf{h}$

Obr. 18

11.2. Příklady

11.2.1. Kód celkové kontroly parity je popsán rovnicí $v_1 + v_2 + \dots + v_n = 0$, a tedy má kontrolní matici

$$\boldsymbol{H} = [1\ 1\ \dots\ 1].$$

11.2.2. Určíme kontrolní matici ternárního kódu (v abecedě Z_3) s generující maticí

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Je-li $\boldsymbol{v} = v_1v_2v_3v_4v_5$ kódové slovo, potom každý řádek $h_1h_2h_3h_4h_5$ kontrolní matice splňuje $h_1v_1 + h_2v_2 + h_3v_3 + h_4v_4 + h_5v_5 = 0$. Aplikujeme tuto podmínku na

řádky $\boldsymbol{v}$ matice $\boldsymbol{G}$ a dostaneme soustavu rovnic

$$\begin{aligned} h_1 + h_2 + h_3 + h_4 + h_5 &= 0\,, \\ h_2 + h_3 + h_4 + h_5 &= 0\,, \\ h_1 + h_2 \qquad\qquad\quad &= 0\,. \end{aligned}$$

Její řešení tvoří právě všechna slova $00h_3h_4h_5$, kde $h_3 = -h_4 - h_5$. Dimenze prostoru řešení je tedy 2 (protože dva parametry, např. h_4 a h_5, volíme libovolně) a kontrolní matice má dva řádky. Volíme $h_4 = 1$, $h_5 = 0$ a potom $h_4 = 0$, $h_5 = 1$:

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 0 & 1 \end{bmatrix}.$$

To znamená, že kód je popsán rovnicemi

$$\begin{aligned} 2v_3 + v_4 &= 0\,, \\ 2v_3 + v_5 &= 0 \end{aligned}$$

neboli $v_3 = v_4 = v_5$.

Pro systematické kódy máme jednodušší způsob určení kontrolní matice. Ten je založen na tom, že můžeme najít generující matici, která má v levé části jednotkovou matici $\boldsymbol{E}$, tedy matici tvaru $\boldsymbol{G} = [\boldsymbol{E} \mid \boldsymbol{B}]$, viz (10.7):

11.3. Věta. *Lineární kód s generující maticí* $\boldsymbol{G} = [\boldsymbol{E} \mid \boldsymbol{B}]$ *má kontrolní matici* $\boldsymbol{H} = [-\boldsymbol{B}^{\mathrm{T}} \mid \boldsymbol{E}']$,
kde $\boldsymbol{B}^{\mathrm{T}}$ *je transponová matice* $\boldsymbol{B}$ *a* $\boldsymbol{E}'$ *je jednotková matice.*

Důkaz. Buď K lineární (n, k)-kód s generující maticí $\boldsymbol{G} = [\boldsymbol{E} \mid \boldsymbol{B}]$. Máme ověřit, že K je lineární prostor, shodný s prostorem K_0 všech řešení $\boldsymbol{v}$ rovnice $\boldsymbol{H}\boldsymbol{v}^{\mathrm{T}} = \boldsymbol{o}^{\mathrm{T}}$. Platí

$$\begin{aligned} \boldsymbol{H}\boldsymbol{G}^{\mathrm{T}} &= [-\boldsymbol{B}^{\mathrm{T}} \mid \boldsymbol{E}'] \begin{bmatrix} \boldsymbol{E} \\ \hline \boldsymbol{B}^{\mathrm{T}} \end{bmatrix} = \\ &= -\boldsymbol{B}^{\mathrm{T}}\boldsymbol{E} + \boldsymbol{E}'\boldsymbol{B}^{\mathrm{T}} = \\ &= -\boldsymbol{B}^{\mathrm{T}} + \boldsymbol{B}^{\mathrm{T}} = \boldsymbol{0}\,. \end{aligned}$$

To znamená, že pro každý řádek $\boldsymbol{v}$ matice $\boldsymbol{G}$ platí $\boldsymbol{G}\boldsymbol{v}^{\mathrm{T}} = \boldsymbol{o}^{\mathrm{T}}$. Jinými slovy, prostor K_0 obsahuje celou bázi prostoru K, takže K je podprostorem prostoru K_0.

Nyní stačí ověřit, že prostory K a K_0 mají stejnou dimenzi. Matice $\boldsymbol{G}$, a tedy i matice $\boldsymbol{B}$, má k řádků. Jednotková matice $\boldsymbol{E}'$ je řádu $n - k$ (protože z n sloupců matice $\boldsymbol{H}$ tvoří k sloupců matici $-\boldsymbol{B}^{\mathrm{T}}$), takže matice $\boldsymbol{H}$ má hodnost $h(\boldsymbol{H}) = n - k$. Odtud plyne, že prostor K_0 má dimenzi $n - h(\boldsymbol{H}) = k$, takže $K = K_0$.

11.4. Příklady

11.4.1. Ternární kód z příkl. 11.2.2 je systematický, protože je popsán rovnicemi $v_3 = v_4 = v_5$. Můžeme zvolit tuto generující matici:

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Potom kontrolní matice je

$$H = \left[\begin{array}{ccc|cc} 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccccc} 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 0 & 1 \end{array}\right].$$

11.4.2. Určíme kontrolní matici 5-znakového kódu (v abecedě Z_5) s generující maticí

$$G = \left[\begin{array}{cccccc} 3 & 2 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right].$$

Snažíme se najít generující matici tvaru $(E \mid B)$. K tomu použijeme elementární úpravy: první řádek dělíme třemi (tj. násobíme dvěma, protože v Z_5 platí $3 \cdot 2 = 1$) a odečteme od druhého řádku:

$$G \sim \left[\begin{array}{cccccc} 1 & 4 & 2 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \sim \left[\begin{array}{cccccc} 1 & 4 & 2 & 0 & 0 & 0 \\ 0 & 2 & 4 & 1 & 1 & 1 \end{array}\right].$$

Potom druhý řádek dělíme dvěma (tj. násobíme třemi) a čtyřnásobek odečteme od prvého řádku:

$$G \sim \left[\begin{array}{cccccc} 1 & 0 & 4 & 2 & 2 & 2 \\ 0 & 1 & 2 & 3 & 3 & 3 \end{array}\right].$$

Vidíme, že daný kód je systematický: má generující matici

$$\left[\begin{array}{cc|cccc} 1 & 0 & 4 & 2 & 2 & 2 \\ 0 & 1 & 2 & 3 & 3 & 3 \end{array}\right],$$

a tedy kontrolní matici

$$H = \left[\begin{array}{cccccc} -4 & -2 & 1 & 0 & 0 & 0 \\ -2 & -3 & 0 & 1 & 0 & 0 \\ -2 & -3 & 0 & 0 & 1 & 0 \\ -2 & -3 & 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{cccccc} 1 & 3 & 1 & 0 & 0 & 0 \\ 3 & 2 & 0 & 1 & 0 & 0 \\ 3 & 2 & 0 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 0 & 1 \end{array}\right].$$

11.4.3. „Koktavý" kód 7.2.2 není systematický. Můžeme však přejít k ekvivalentnímu systematickému kódu K' (10.10.1), který má generující matici

$$G_1' = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right],$$

a tedy kontrolní matici

$$H' = \left[\begin{array}{ccc|ccc} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right].$$

Přitom vznikl kód K' z koktavého kódu přehozením druhého a pátého sloupce. Jestliže provedeme inverzní permutaci, tj. přehodíme druhý a pátý sloupec zpět, dostaneme kontrolní matici koktavého kódu:

$$H = \left[\begin{array}{cccccc} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right].$$

Přesvědčte se přímo o tom, že platí $\boldsymbol{H}\boldsymbol{v}^{\mathrm{T}} = \boldsymbol{o}^{\mathrm{T}}$, právě když $\boldsymbol{v}$ je slovo koktavého kódu.

11.5. Duální kód. Na prostoru T^n definujeme, jako obvykle, *skalární součin* předpisem

$$\boldsymbol{u} * \boldsymbol{v} = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n$$

pro libovolná dvě slova $\boldsymbol{u} = u_1 \ldots u_n$ a $\boldsymbol{v} = v_1 \ldots v_n$. Například v Z_2^5 platí

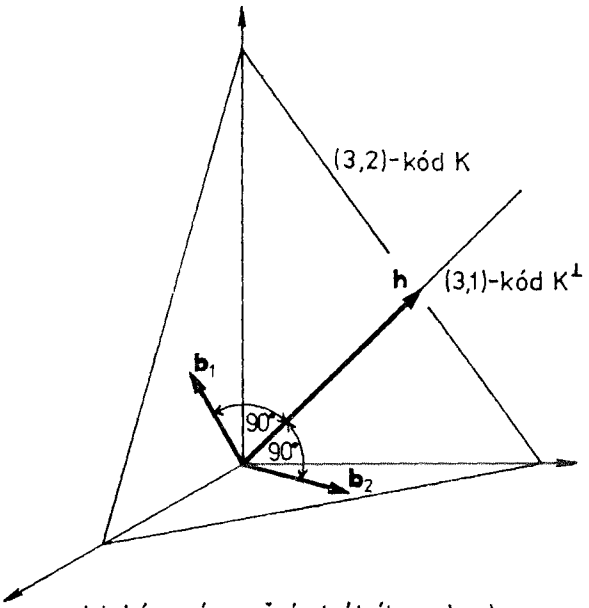
$$11010 * 01011 = 1 + 1 = 0\,,$$

zatímco v Z_3^5 platí

$$11010 * 01011 = 1 + 1 = 2\,.$$

Duální kód lineárního kódu $K \subseteq T^n$ definujeme jako kód $K^{\perp} \subseteq T^n$ všech slov $u_1 u_2 \ldots u_n$ v T^n, která mají s každým slovem v K skalární součin roven nule. Stručněji:

$\boldsymbol{u} \in K^{\perp}$, právě když $\boldsymbol{u} * \boldsymbol{v} = 0$ pro každé $\boldsymbol{v} \in K$.



Obr. 19

11.6. Příklad. Duální kód opakovacího kódu délky 5 je kód všech slov $u_1 u_2 u_3 u_4 u_5$ takových, že

$$u_1 + u_2 + u_3 + u_4 + u_5 = 0\,.$$

(Každé takové slovo má skalární součin se slovem 11111 roven 0, a tedy i jeho skalární součin se slovy *tttit* je roven 0.) Vidíme, že duální kód opakovacího kódu je kód celkové kontroly parity.

11.7. Tvrzení. Duální kód $K^{\perp}$ lineárního (n, k)-kódu K má $n - k$ informačních znaků, a je tedy $(n, n - k)$-kódem. Generující matice kódu K je kontrolní maticí kódu $K^{\perp}$ (a naopak).

Důkaz. Duální kód $K^{\perp}$ je množina všech slov $x_1x_2 \ldots x_n$, která jsou řešeními soustavy lineárních rovnic

$$v_1x_1 + v_2x_2 + \ldots + v_nx_n = 0 \quad (v_1v_2 \ldots v_n \in K).$$

Jak známo, je množina všech řešení homogenní soustavy lineárních rovnic lineárním prostorem dimenze $n - k$, kde k je hodnost matice soustavy. V našem případě je hodnost matice rovna dimenzi prostoru K. To znamená, že $K^{\perp}$ je lineární $(n, n - k)$-kód.

Buď $\boldsymbol{G}$ generující matice kódu K. Pro každé slovo $w_1w_2 \ldots w_n$ kódu K jsou skalární součiny s řádky matice $\boldsymbol{G}$ rovny 0, takže platí

$$\boldsymbol{G}\begin{bmatrix} w_1 \\ w_2 \\ \ldots \\ w_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \ldots \\ 0 \end{bmatrix}.$$

To znamená, že kód K je podprostorem prostoru L všech slov $\boldsymbol{w}$, splňujících $\boldsymbol{G}\boldsymbol{w}^{\mathrm{T}} = \boldsymbol{o}^{\mathrm{T}}$. Protože dimenze prostorů $K^{\perp}$ i L je rovna $n - k$ (kde k je hodnost matice $\boldsymbol{G}$), platí $K^{\perp} = L$. Tyto úvahy platí i naopak, protože duálním kódem kódu $K^{\perp}$ je kód K.

11.8. Příklad. Hledáme duální kód binárního kódu

K: 0000, 1011, 0111, 1100.

Jeho generující matice je např.

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

a kontrolní matice je tedy (podle 11.3)

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}.$$

To znamená, že $\boldsymbol{H}$ je generující matice duálního kódu:

$K^{\perp}$: 0000, 1110, 1101, 0011.

11.9. Poznámka. Pojem duálního kódu úzce souvisí s pojmem ortogonálního doplňku (reálného) vektorového prostoru. Je zde významný rozdíl: kódy K a $K^{\perp}$ mohou mít netriviální společná kódová slova. Dokonce existují *samoduální kódy*, tj. kódy K takové, že $K = K^{\perp}$ (a tedy generující matice je zároveň maticí kontrolní). Například „koktavý" kód (7.2.2) je samoduální. Jeho generující matice

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

je zároveň maticí kontrolní: vznikne z kontrolní matice $\boldsymbol{H}$ v 11.4.3 přehozením řádků.

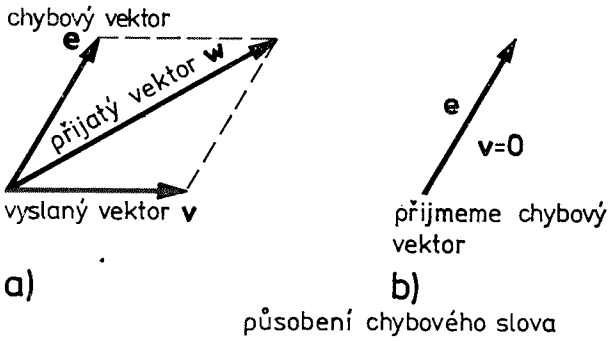
11.10. Úloha. Najděte duální kódy kódů z 10.11.

12. Objevování chyb

O objevování t-násobných chyb jsme mluvili v čl. 4. U lineárních kódů můžeme podrobněji probírat různé typy chyb zavedením chybového slova. Vyšleme kódové slovo $v_1v_2\ldots v_n$ a přijemem slovo $w_1w_2\ldots w_n$. Potom *chybovým slovem* nazýváme slovo

$$e_1e_2\ldots e_n = (w_1w_2\ldots w_n) - (v_1v_2\ldots v_n)\,.$$

Tedy i naopak: chybové slovo $e_1e_2\ldots e_n$, vytvořené šumem, zkomolí slovo $v_1v_2\ldots v_n$ na slovo $(e_1e_2\ldots e_n) + (v_1v_2\ldots v_n)$. Lineární kód K *objevuje chybové slovo* $e_1e_2\ldots e_n$, jestliže pro každé kódové slovo $v_1v_2\ldots v_n$ platí: slovo $(e_1e_2\ldots e_n) + (v_1v_2\ldots v_n)$ není kódové. Speciálně, kód objevuje t-násobné chyby, právě když objevuje všechna chybová slova Hammingovy váhy $\leqq t$.



Obr. 20

12.1. Příklad: Binární kód celkové kontroly parity délky 5. Tento kód objevuje jednoduché chyby, ale neobjevuje dvojnásobné chyby (viz 4.6). Objevuje však i všechny chyby, které zkreslí 3 znaky nebo 5 znaků. Jestliže má totiž chybové slovo $e_1e_2e_3e_4e_5$ lichou paritu, potom je kód objeví: pro každé kódové slovo $v_1v_2v_3v_4v_5$ (sudé parity) má součet $(e_1e_2e_3e_4e_5) + (v_1v_2v_3v_4v_5)$ paritu lichou. Obecněji:

12.2. Pozorování. Lineární kód objevuje právě ta chybová slova, která nejsou slovy kódovými.

Pokud totiž slovo $e_1e_2\ldots e_n$ není kódové a slovo $v_1v_2\ldots v_n$ je kódové, potom slovo $w_1w_2\ldots w_n = (e_1e_2\ldots e_n) + (v_1v_2\ldots v_n)$ není kódové: kdyby bylo, plynulo by z linearity kódu, že i slovo $(w_1w_2\ldots w_n) - (v_1v_2\ldots v_n) = e_1e_2\ldots e_n$ je kódové. Naopak, pokud je chybové slovo $e_1e_2\ldots e_n$ kódové, pak při vyslání nulového slova $000\ldots0$ přijmeme slovo $(e_1e_2\ldots e_n) + (000\ldots0) = e_1e_2\ldots e_n$ a tuto chybu neobjevíme.

12.3. Poznámka. Pro lineární kód K platí: minimální vzdálenost d (4.2) je rovna minimální váze (10.12) nenulového kódového slova. Stručněji:

$$d = \min_{\mathbf{v}\in K-\{\mathbf{o}\}} \|\mathbf{v}\|\,.$$

Jestliže totiž kódová slova $\mathbf{w}$ a $\mathbf{w}'$ mají Hammingovu vzdálenost d, potom kódové slovo $\mathbf{w} - \mathbf{w}'$ má Hammingovu váhu d. Naopak, jestliže kódové slovo $\mathbf{v} \neq \mathbf{o}$ má Hammingovu váhu d, potom Hammingova vzdálenost slov $\mathbf{v}$ a $\mathbf{o}$ je rovna d.

Odtud plyne, že lineární kód objevuje t-násobné chyby, právě když každé nenulové kódové slovo má Hammingovu váhu větší než t, viz 4.5.

12.4. Syndrom. K objevování chyb můžeme využít kontrolní matici. Je-li $\boldsymbol{H}$ kontrolní matice lineárního kódu délky n, pak pro každé slovo $\mathbf{v} = v_1 v_2 \ldots v_n$ definujeme slovo $\mathbf{s} = s_1 \ldots s_m$ předpisem

$$\boldsymbol{H} \begin{bmatrix} v_1 \\ v_2 \\ \ldots \\ v_n \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \ldots \\ s_m \end{bmatrix}.$$

Slovo $\mathbf{s}$ se nazývá *syndrom slova* $\mathbf{v}$. Je-li syndrom nenulový,

$$\mathbf{s} \neq \mathbf{o},$$

víme, že došlo k chybě. V případě nulového syndromu předpokládáme, že nedošlo k chybě (protože slovo $v_1 v_2 \ldots v_n$ je kódové).

Důležitou vlastností syndromu je to, že *přijaté slovo má stejný syndrom jako chybové slovo.* Jestliže totiž vyšleme kódové slovo $\mathbf{v}$ a přijmeme slovo $\mathbf{w} = \mathbf{v} + \mathbf{e}$ (kde $\mathbf{e}$ je chybové slovo), platí

(12.4.1) $\quad \boldsymbol{H}\mathbf{w}^{\mathrm{T}} = \boldsymbol{H}\mathbf{e}^{\mathrm{T}}.$

To plyne z faktu, že $\boldsymbol{H}\mathbf{v}^{\mathrm{T}} = \mathbf{o}^{\mathrm{T}}$, takže

$$\boldsymbol{H}\mathbf{w}^{\mathrm{T}} = \boldsymbol{H}(\mathbf{v}^{\mathrm{T}} + \mathbf{e}^{\mathrm{T}}) = \mathbf{o}^{\mathrm{T}} + \boldsymbol{H}\mathbf{e}^{\mathrm{T}}.$$

12.5. Příklady

12.5.1. Kód celkové kontroly parity má kontrolní matici $\boldsymbol{H} = [1\ 1\ 1\ \ldots\ 1]$. Syndromem slova $v_1 v_2 \ldots v_n$ je tedy součet $v_1 + v_2 + \ldots + v_n$.

12.5.2. „Koktavý" kód má kontrolní matici (11.4.3)

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Syndrom slova $v_1 v_2 v_3 v_4 v_5 v_6$ dostaneme součinem s maticí $\boldsymbol{H}$:

$$\boldsymbol{H} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} v_3 + v_4 \\ \\ v_1 + v_2 \\ \\ v_5 + v_6 \end{bmatrix}.$$

To znamená, že syndrom je $s_1 s_2 s_3$, kde $s_1 = v_3 + v_4$, $s_2 = v_1 + v_2$ a $s_3 = v_5 + v_6$.

12.6. Tvrzení. Lineární kód objevuje t-násobné chyby, právě když každých t sloupců jeho kontrolní matice je lineárně nezávislých.

Důkaz. Zvolme kódové slovo $\boldsymbol{v} \neq \boldsymbol{o}$ minimální váhy d. To znamená, že $\boldsymbol{v} = v_1 v_2 \ldots v_n$, kde $v_{i_1}, v_{i_2}, \ldots, v_{i_d}$ jsou nenulové složky a $v_i = 0$ pro $i \neq i_1, i_2, \ldots, i_d$. Sloupce kontrolní matice $\boldsymbol{H}$ označme $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n$. Podle (10.6) platí

$$\boldsymbol{o}^{\mathrm{T}} = \boldsymbol{H}\boldsymbol{v}^{\mathrm{T}} = v_{i_1}\boldsymbol{h}_{i_1} + v_{i_2}\boldsymbol{h}_{i_2} + \ldots + v_{i_d}\boldsymbol{h}_{i_d}.$$

Sloupce $\boldsymbol{h}_{i_1}, \ldots, \boldsymbol{h}_{i_d}$ jsou tedy lineárně závislé.

Vidíme, že pro každé kódové slovo $\boldsymbol{v} \neq \boldsymbol{o}$ váhy d máme d lineárně závislých sloupců matice $\boldsymbol{H}$. Jestliže tedy každých t sloupců je lineárně nezávislých, platí $t < d$, takže kód objevuje t-násobné chyby (4.5.).

Naopak, jestliže kód objevuje t-násobné chyby, ověříme, že libovolné sloupce $\boldsymbol{h}_{i_1}, \ldots, \boldsymbol{h}_{i_t}$ jsou lineárně nezávislé. Pro každou lineární kombinaci $w_{i_1}\boldsymbol{h}_{i_1} + w_{i_2}\boldsymbol{h}_{i_2} + \ldots + w_{i_t}\boldsymbol{h}_{i_t} = \boldsymbol{o}^{\mathrm{T}}$ máme ukázat, že koeficienty $w_{i_1}, \ldots, w_{i_t}$ jsou samé nuly. Definujeme slovo $w_1 w_2 \ldots w_n$ tak, že položíme $w_i = 0$ pro všechna $i = 1, \ldots, n$ kromě $i_1, i_2, \ldots, i_d$. Podle 10.6 platí

$$\boldsymbol{H}\boldsymbol{w}^{\mathrm{T}} = w_{i_1}\boldsymbol{h}_{i_1} + \ldots + w_{i_t}\boldsymbol{h}_{i_t} = \boldsymbol{o}^{\mathrm{T}},$$

takže $\boldsymbol{w}$ je kódové slovo. Jeho váha je $\leqq t$, ale podle 4.5 a 12.3 je váha nenulového kódového slova $> t$. To znamená, že $\boldsymbol{w} = \boldsymbol{o}$, a to jsme měli ukázat.

12.7. Příklad. Lineární kód objevuje jednoduché chyby, právě když žádný sloupec kontrolní matice není nulový, tj. právě když žádné slovo $0 \ldots 00100 \ldots 0$ váhy 1 není kódové.

12.8. Úloha. Ověřte, že lineární kód objevuje dvojnásobné chyby, právě když žádný sloupec kontrolní matice není skalárním násobkem jiného sloupce.

13. Opravování chyb

Podobně jako v případě objevování chyb, můžeme u lineárních kódů přesně stanovit, jaké typy chyb (tj. chybová slova) daný kód opravuje. Vlastně ne daný kód, ale dané dekódování. Připomeňme, že dekódování je předpis δ, který každému (přijatému) slovu $w_1 w_2 \ldots w_n$ přiřazuje kódové slovo $\delta(w_1 w_2 \ldots w_n)$, viz čl. 6. Jestliže vyšleme kódové slovo $\boldsymbol{v} = v_1 v_2 \ldots v_m$ a dojde k chybě vyjádřené slovem $\boldsymbol{e} = e_1 e_2 \ldots e_n$, pak přijmeme slovo $\boldsymbol{e} + \boldsymbol{v}$. Pokud

$$\delta(\boldsymbol{e} + \boldsymbol{v}) = \boldsymbol{v},$$

dekódovali jsme správně. To znamená, že *lineární kód při dekódování δ opravuje právě ta chybová slova* $\boldsymbol{e}$, *která splňují*

(*) $\delta(\boldsymbol{e} + \boldsymbol{v}) = \boldsymbol{v}$ (pro každé kódové slovo $\boldsymbol{v}$).

Pro každý lineární kód uvedeme tzv. standardní dekódování. Je to optimální postup z hlediska počtu opravovaných chybových slov, ale je pomalý. Proto se

u významných tříd kódů, které poznáme v dalších kapitolách, používá rychlejší dekódování, i když má slabší vlastnost: opraví sice všechny t-násobné chyby pro $t < d/2$, kde d je minimální vzdálenost (viz 5.1), ale neopravuje chybová slova násobnosti $\geqq d/2$, která je s to opravit standardní dekódování. Jestliže tedy máme k dispozici „dost" času, je standardní dekódování nejlepší.

13.1. Třídy slov podle kódu. Používáme lineární kód K délky n v abecedě T. Potom je K podgrupa aditivní grupy T^n, tj. součet i rozdíl dvou kódových slov je kódové slovo. U každého slova $\mathbf{e} = e_1 e_2 \ldots e_n$ v T^n mluvíme o jeho *třídě* podle kódu K (srovnejte s 9.8): to je množina

$$\mathbf{e} + K = \{\mathbf{e} + \mathbf{v} \mid \mathbf{v} \in K\} .$$

Například kód K sám je třídou:

$$K = \mathbf{o} + K ;$$

pro každé kódové slovo $\mathbf{v}$ ovšem platí $\mathbf{v} + K = K$, takže $\mathbf{o}$ a $\mathbf{v}$ mají stejnou třídu.

Pro libovolná slova $\mathbf{e}$ a $\mathbf{e}'$ v T^n platí:

(i) *Jestliže je* $\mathbf{e} - \mathbf{e}'$ *kódové slovo, jsou třídy slov* $\mathbf{e}$ *a* $\mathbf{e}'$ *stejné, tj.* $\mathbf{e} + K = \mathbf{e}' + K$;

(ii) *Jestliže není* $\mathbf{e} - \mathbf{e}'$ *kódové slovo, jsou třídy disjunktní, tj. žádné slovo neleží v* $\mathbf{e} + K$ *i v* $\mathbf{e}' + K$.

(iii) *Počet slov každé třídy* $\mathbf{e} + K$ *je roven počtu kódových slov. Jestliže je* K (n, k)*-kódem v* q*-prvkové abecedě, je počet všech tříd roven* q^{n-k}.

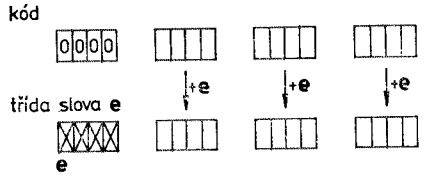
Důkaz. (i) Označme $\mathbf{v}_0 = \mathbf{e} - \mathbf{e}'$. Pro každé slovo $\mathbf{w} = \mathbf{e} + \mathbf{v}$ třídy $\mathbf{e} + K$ jsou $\mathbf{v}$ a $\mathbf{v}_0$ kódová slova, takže i $\mathbf{v} + \mathbf{v}_0$ je kódové slovo. Odtud plyne

$$\mathbf{w} = \mathbf{e} + \mathbf{v} = \mathbf{e}' + \mathbf{v}_0 + \mathbf{v} \in \mathbf{e}' + K .$$

Podobně se ověří, že každé slovo třídy $\mathbf{e}' + K$ leží i ve třídě $\mathbf{e} + K$.

(ii) Kdyby měly třídy $\mathbf{e} + K$ a $\mathbf{e}' + K$ společné slovo, bylo by toto slovo tvaru $\mathbf{e} + \mathbf{v} = \mathbf{e}' + \mathbf{v}'$, kde $\mathbf{v}$ a $\mathbf{v}'$ jsou kódová slova. Potom $\mathbf{e} - \mathbf{e}' = \mathbf{v}' - \mathbf{v}$ je také kódové slovo, a to je spor.

(iii) Vyjmenujme všechna kódová slova: $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{q^k}$ (jejich počet je q^k, viz 10.2). Pak $\mathbf{e} + K$ je množina všech slov $\mathbf{e} + \mathbf{v}_1, \ldots, \mathbf{e} + \mathbf{v}_{q^k}$. Ta jsou navzájem různá, protože z rovnosti $\mathbf{e} + \mathbf{v}_i = \mathbf{e} + \mathbf{v}_j$ odečtením slova $\mathbf{e}$ dostáváme $\mathbf{v}_i = \mathbf{v}_j$ (a tedy $i = j$). Proto má třída $\mathbf{e} + K$ právě q^k prvků. Podle (i) a (ii) se žádné dvě různé třídy nepřekrývají. To znamená, že celkový počet q^n slov množiny T^n se rozděluje mezi stejně velké třídy velikosti q^k, takže počet tříd je $q^n : q^k = q^{n-k}$.



Obr. 21

13.2. Příklad: Binární (4,3)-kód celkové kontroly parity. Celý kód K sestává z těchto slov:

$$K = \{0000,\ 1001,\ 0101,\ 0011,\ 1111,\ 1100,\ 0110,\ 1010\}\ .$$

Pro každé kódové slovo $\mathbf{e}$ je ovšem $\mathbf{e} + K = K$. Zvolme nekódové slovo, např. 1000. Platí

$$1000 + K = \{1000,\ 0001,\ 1101,\ 1011,\ 0111,\ 0100,\ 1110,\ 0010\}\ .$$

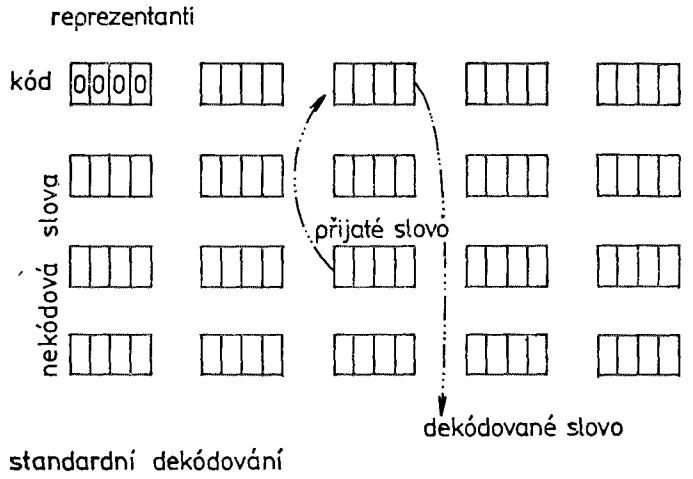
Vidíme, že $1000 + K$ je třída všech slov liché parity. Celkový počet tříd je $2^{4-3} = 2$.

13.3. Standardní dekódování. Definujeme dekódování lineárního kódu $K \subseteq T^n$. Z každé třídy podle K vybereme jedno slovo, tzv. *reprezentanta* třídy, tak, aby jeho váha byla co nejmenší. Pak každé (přijaté) slovo $\boldsymbol{w}$ v T^n dekódujeme jako $\boldsymbol{v} = \boldsymbol{w} - \mathbf{e}$, kde (chybové) slovo $\mathbf{e}$ je reprezentantem třídy slova $\boldsymbol{w}$. Stručněji:

$$\delta(\boldsymbol{w}) = \boldsymbol{w} - [\text{reprezentant třídy } \boldsymbol{w} + K]\ .$$

13.4. Příklad. Binární (4,3)-kód celkové kontroly parity má dvě třídy: K, všechna kódová slova, a $\overline{K}$, všechna slova liché parity. Reprezentantem první třídy je nutně 0000 (protože má nejmenší váhu). Reprezentantem druhé třídy může být libovolné slovo váhy 1. Zvolme např. 0001. Pak standardní dekódování nemění slova $\boldsymbol{v}$ sudé parity, $\delta(\boldsymbol{v}) = \boldsymbol{v} - 0000 = \boldsymbol{v}$, a slova liché parity změní opravou posledního znaku, $\delta(\boldsymbol{v}) = \boldsymbol{v} - 0001$. Například $\delta(1011) = 1010$.

V tomto případě standardní dekódování opravuje právě tu chybu, která neovlivní první tři znaky.



Obr. 22

13.5. Standardní dekódování můžeme dobře znázornit graficky tzv. *Slepianovým standardním rozmístěním.* To je tabulka, jejíž řádky jsou třídy a na prvním místě stojí reprezentant třídy. V prvním řádku stojí samotný kód, jehož reprezentan-

tem je ovšem nulové slovo (to má minimální váhu). Je to tedy řádek

$$\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3\ \ldots\ \mathbf{v}_{q^k},$$

kde $\mathbf{v}_1 = 000\ldots0$; q^k je počet kódových slov (n, k)-kódu v q-prvkové abecedě. Pro druhý řádek zvolíme reprezentanta: je to libovolné slovo $\mathbf{e}$ co nejmenší váhy, které není slovem kódovým. Tak vznikne druhý řádek

$$\mathbf{e} + \mathbf{v}_1 \quad \mathbf{e} + \mathbf{v}_2 \quad \mathbf{e} + \mathbf{v}_3 \quad \ldots \quad \mathbf{e} + \mathbf{v}_{q^k}.$$

Pro třetí řádek zvolíme reprezentanta $\mathbf{e}'$: slovo co nejmenší váhy, které nestojí v prvních dvou řádcích, atd.

Vznikne tabulka o q^{n-k} řádcích, která obsahuje vůbec všechna slova prostoru T^n. Jestliže přijmeme slovo $w_1w_2\ldots w_n$, vyhledáme je v tabulkce a dekódujeme jako to kódové slovo $v_1v_2\ldots v_n$, které stojí v tomtéž sloupci. (To totiž znamená, že slovo $\mathbf{w} = w_1w_2\ldots w_n$ vzniklo ze slova $\mathbf{v} = v_1v_2\ldots v_n$ přičtením reprezentata $\mathbf{e}$ třídy $\mathbf{w} + K = \mathbf{e} + K$. Tedy $\mathbf{v} = \mathbf{w} - \mathbf{e} = \delta(\mathbf{w})$.)

13.6. Příklad. Binární (4, 3)-kód celkové kontroly parity má toto standardní dekódování:

	reprezentant							
kód	0000	1001	0101	0011	1111	1100	0110	1010
	0001	1000	0100	0010	1110	1101	0111	1011

Jestliže přijmeme slovo 1101, najdeme je v tabulce a dekódujeme jako to kódové slovo, v jehož sloupci přijaté slovo leží:

$\delta(1101) = 1100$.

Jiné standardní dekódování získáme jiným výběrem reprezentanta:

	reprezentant							
kód	0000	1001	0101	0011	1111	1100	0110	1010
	1000	0001	1101	1011	0111	0100	1110	0010

Toto dekódování opraví každou chybu, ovlivňující jen první znak.

13.7. Příklad: Ternární kód s generující maticí

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

Vytváříme tabulku tak, že v prvém řádku vyjmenujeme všech 3^2 kódových slov, pak zvolíme reprezentanta (minimální váhy) druhé třídy, který není kódovým slovem, a nakonec reprezentanta (opět minimální váhy) třetí třídy, který neleží v prvních dvou řádcích. Počet řádků je $3^{3-2} = 3$.

	reprezentant								
kód	000	010	101	020	202	111	222	121	212
	100	110	201	120	002	211	022	221	012
	200	210	001	220	102	011	122	021	112

Které chyby opraví tento kód? Ty, které odpovídají chybovým vektorům 100 a 200, tedy všechny chyby, ovlivňující jen první znak.

Například vyšleme slovo 121. Jestliže dojde k chybě na prvním znaku, přijmeme buď 221, nebo 021 — to jsou slova, ležící ve sloupci kódového slova 121, takže je správně dekódujeme. Ale při chybě na druhém znaku přijmeme např. 111 a to je kódové slovo, takže dekódujeme nesprávně.

13.8. Pozorování. Standardní dekódování opravuje právě ta chybová slova, která jsme zvolili jako reprezentanty tříd.

Skutečně, pokud slovo $\mathbf{e}$ je reprezentantem své třídy, pak platí podmínka (*): vektor $\mathbf{e} + \mathbf{v}$, kde $\mathbf{v}$ je kódové slovo, leží ve třídě $\mathbf{e} + K$, takže

$$\delta(\mathbf{e} + \mathbf{v}) = \mathbf{e} + \mathbf{v} - \mathbf{e} = \mathbf{v}\,.$$

Naopak, jestliže slovo $\mathbf{e}$ není reprezentantem, tedy jiné slovo $\mathbf{e}'$ reprezentuje třídu $\mathbf{e} + K = \mathbf{e}' + K$, potom při vyslání kódového slova $\mathbf{v} = \mathbf{o}$ a přijetí slova $\mathbf{e} + \mathbf{v} = \mathbf{e}$ dekódujeme

$$\delta(\mathbf{e}) = \mathbf{e} - \mathbf{e}'\,.$$

Protože $\mathbf{e} - \mathbf{e}' \neq \mathbf{o}$, je toto dekódování chybné.

13.9. Tvrzení. Každé standardní dekódování δ je optimální v tom smyslu, že žádné jiné dekódování neopravuje větší množinu chybových slov než δ.

Poznámka. Podrobněji: Ověříme, že pokud dekódování δ^* opravuje všechny ty chyby, které opravuje δ (tj. všechny chyby, vytvářející slova reprezentantů třídy), pak neopravuje žádné další chybové slovo.

Důkaz. Zvolme tedy chybové slovo $\mathbf{e}$, které není reprezentantem své třídy. To znamená, že zvolený reprezentant $\mathbf{e}'$ třídy $\mathbf{e} + K$ je různý od $\mathbf{e}$. Slovo $\mathbf{v} = \mathbf{e} - \mathbf{e}'$ je kódové a nenulové. Jestliže vyšleme slovo $\mathbf{v}$ a dojde k chybě, vytvářející slovo $\mathbf{e}'$, přijmeme slovo $\mathbf{v} + \mathbf{e}' = \mathbf{e}$. Tuto chybu opraví dekódování δ, a tedy i dekódování δ^*, takže platí

$$\delta^*(\mathbf{e}) = \mathbf{v}\,.$$

Naproti tomu, když vyšleme kódové slovo $\mathbf{o}$ a dojde k chybě, vytvářející slovo $\mathbf{e}$, přijmeme slovo $\mathbf{o} + \mathbf{e} = \mathbf{e}$ a tuto chybu dekódování δ^* neopraví: $\delta^*(\mathbf{e}) \neq \mathbf{o}$.

13.10. Úloha. Jestliže minimální vzdálenost lineárního kódu je d, ověřte, že standardní dekódování opraví t-násobné chyby pro všechna $t < \frac{1}{2}d$. Návod: pokud má slovo $\mathbf{e}$ váhu t, pak ve třídě $\mathbf{e} + K$ všechna ostatní slova mají váhu $\geqq \frac{1}{2}d$, protože váha slova $\mathbf{e} - \mathbf{e}'(\in K)$ je $\geqq d$ pro $\mathbf{e}' \in \mathbf{e} + K$.

13.11. Dekódování pomocí syndromů. Standardní dekódování je sice optimální, ale zdlouhavé. Například u kódu 13.6 se od nás vyžaduje, abychom po přijetí každého slova prohlédli tabulku o 27 slovech. Běžně užívané binární kódy mají délku 64, takže při standardním dekódování musíme prohledat 2^{64} $(> 10^{19})$ slov. Postup dekódování můžeme naštěstí podstatně urychlit (i když stále ne dostatečně pro velké kódy) pomocí kontrolní matice $\boldsymbol{H}$ daného kódu a syndromů (12.4).

Všimněte si, že dvě slova $\mathbf{e}$ a $\mathbf{e}'$ leží ve stejné třídě, právě když mají stejné syndromy. Stručně:

$$\mathbf{e} + K = \mathbf{e}' + K, \quad \text{právě když} \quad \boldsymbol{H}\mathbf{e}^{\mathrm{T}} = \boldsymbol{H}(\mathbf{e}')^{\mathrm{T}}.$$

Skutečně, rovnost $\mathbf{e} + K = \mathbf{e}' + K$ znamená právě totéž, co $(\mathbf{e} - \mathbf{e}') \in K$, viz 13.1. A platí $(\mathbf{e} - \mathbf{e}') \in K$, právě když $\boldsymbol{H}(\mathbf{e} - \mathbf{e}')^{\mathrm{T}} = \mathbf{o}^{\mathrm{T}}$, tj. $\boldsymbol{H}\mathbf{e}^{\mathrm{T}} = \boldsymbol{H}(\mathbf{e}')^{\mathrm{T}}$.

Místo celé dekódovací tabulky tedy stačí znát seznam reprezentantů tříd a jejich syndromů:

reprezentant	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$	...	$\mathbf{e}_m$
syndrom	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_3$	...	$\mathbf{s}_m$.

Jestliže přijmeme vektor $\mathbf{w}$, zjistíme jeho syndrom. Je to některé z daných slov $\mathbf{s}_i$ a my položíme

$$\delta(\mathbf{w}) = \mathbf{w} - \mathbf{e}_i .$$

Například kód 13.6 má kontrolní matici

$$\boldsymbol{H} = [1\ 0\ 2] .$$

Dekódovací tabulku zjednodušíme takto:

reprezentant	000	100	200
syndrom	0	1	2 .

Jestliže vyšleme slovo 121 a přijmeme slovo 221, vypočteme syndrom

$$\boldsymbol{H}\mathbf{w}^{\mathrm{T}} = [1\ 0\ 2] \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} = 1 ,$$

a dekódujeme tedy

$$\delta(221) = 221 - 100 = 121 .$$

13.12. Poznámka. Při sestavování standardního dekódování se můžeme řídit syndromy: jestliže jsme nalezli reprezentanty $\mathbf{e}_1, \ldots, \mathbf{e}_m$, potom dalším reprezentantem je libovolné slovo $\mathbf{e}_{m+1}$, jehož syndrom je odlišný od předchozích m syndromů. Přesněji řečeno, ne libovolné slovo, ale slovo co nejnižší váhy s touto vlastností.

13.13. Příklad: Hammingův (7, 4)-kód. Ukážeme příklad kódu, který má velmi rychlé standardní dekódování. Je to binární kód, definovaný kontrolní maticí, jejíž sloupce jsou binární rozvoje čísel 1, 2, ..., 7:

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} .$$

Tato matice má hodnost 3, takže počet informačních znaků je $7 - 3 = 4$. Hledáme reprezentanty tříd. Prvním z nich je slovo $\mathbf{e}_0 = 0000000$ a to má syndrom 000. Další hledáme mezi slovy váhy 1. Slovo $\mathbf{e}_1 = 1000000$ má syndrom 001, a tedy neleží ve třídě $\mathbf{e}_0 + K$. Zvolíme je tedy za reprezentanta. Dále slovo $\mathbf{e}_2 = 0100000$ má syndrom 010, odlišný od předchozích syndromů, takže toto slovo můžeme zvolit za dalšího reprezentanta. Postupně zjistíme, že slova $\mathbf{e}_i$ s jedinou jedničkou na i-tém

místě mají navzájem různé syndromy: syndromem je i-tý sloupec matice $\boldsymbol{H}$ (tj. číslo i v binárním rozvoji). Protože těchto slov, včetně $\mathbf{e}_0$, je osm a protože (7, 4)-kód má počet tříd $2^{7-4} = 8$, našli jsme všechny reprezentanty:

reprezentant	$\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$	$\mathbf{e}_6$	$\mathbf{e}_7$
syndrom	000	001	010	011	100	101	110	111.

Dekódování je jednoduché: přijaté slovo $\boldsymbol{w}$ má syndrom $\boldsymbol{s}$, který je rozvojem některého čísla $i = 0, 1, \ldots, 7$. Potom opravíme i-tý znak (tj. vytvoříme slovo $\boldsymbol{w} - \mathbf{e}_i$). Například pro $\boldsymbol{w} = 1010111$ máme

$$\boldsymbol{H}\boldsymbol{w}^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

Opravíme tedy šestý znak (6 je binárně 110): $\boldsymbol{v} = 1010101$. Toto dekódování je správné, právě když nastala jednoduchá chyba, viz 13.8.

13.14. Závěr. Lineární kódy mají standardní dekódování, založené na určení reprezentantů všech tříd a jejich syndromů. Je to dekódování optimální (co do počtu opravených chyb), ale zdlouhavé: při přijetí každého slova bychom měli projít tabulku všech q^{n-k} syndromů. Proto hledáme kódy s účinnějším dekódováním. Příkladem je Hammingův (7, 4)-kód. Další článek věnujeme Hammingovým kódům obecně.

14. Hammingovy kódy — perfektní kódy pro jednoduché opravy

Hammingovy kódy tvoří významnou třídu kódů, které opravují jednoduché chyby. Tyto kódy se neobyčejně snadno dekódují a jsou perfektní, tj. mají nejmenší myslitelnou redundanci. Hammingův binární kód s m kontrolními znaky ($m = 2, 3, 4, \ldots$) má délku $2^m - 1$, takže dostáváme binární (3, 2)-kód, (7, 4)-kód, (15, 11)-kód atd. Informační poměr (7.8) roste rychle k 1, např. pro kód délky $2^6 - 1 = 63$ je informační poměr

$$\frac{63 - 6}{63} > 0{,}9\,.$$

Rozšířením Hammingova kódu o jeden kontrolní znak dostaneme kód délky 2^m, který opravuje jednoduché chyby a objevuje trojnásobné chyby. Myšlenka, která vede k Hammingovým kódům, je vyjádřena v následujícím tvrzení.

14.1. Tvrzení. Binární lineární kód opravuje jednoduché chyby, právě když sloupce jeho kontrolní matice jsou nenulové a navzájem různé.

Důkaz. Víme, že kód K opravuje jednoduché chyby, právě když jeho minimální vzdálenost je alespoň 3 (5.1), tj. právě když každé kódové slovo $\boldsymbol{v} \neq \boldsymbol{o}$ má váhu $\|\boldsymbol{v}\| \geqq 3$ (12.3).

Jestliže má kontrolní matice $\boldsymbol{H}$ nulový sloupec (i-tý), platí $\boldsymbol{H}\mathbf{e}_i^{\mathrm{T}} = \boldsymbol{o}^{\mathrm{T}}$, kde $\mathbf{e}_i = 00\ldots010\ldots0$ je slovo s jedničkou na i-tém místě. Potom je $\mathbf{e}_i$ kódové slovo váhy 1. Podobně, jestliže má matice $\boldsymbol{H}$ dva shodné sloupce (i-tý a j-tý), platí $\boldsymbol{H}\mathbf{e}_{ij}^{\mathrm{T}} = \boldsymbol{o}^{\mathrm{T}}$, kde $\mathbf{e}_{ij}$ má dvě jedničky, na i-tém a j-tém místě. Potom je $\mathbf{e}_{ij}$ kódové slovo váhy 2. Ani v jednom z těchto případů neopravuje kód K jednoduché chyby.

Předpokládejme naopak, že matice $\boldsymbol{H}$ má sloupce nenulové a po dvou různé. Ověříme, že žádné slovo váhy 1 nebo 2 není kódové. Slovo váhy 1 je vždy tvaru $\mathbf{e}_i$. Protože $\boldsymbol{H}\mathbf{e}_i^{\mathrm{T}}$ ($\neq \boldsymbol{o}^{\mathrm{T}}$) je i-tý sloupec matice $\boldsymbol{H}$, není $\mathbf{e}_i$ kódovým slovem. Slovo váhy 2 je vždy tvaru $\mathbf{e}_{ij}$. Protože je $\boldsymbol{H}\mathbf{e}_{ij}^{\mathrm{T}}$ součtem i-tého a j-tého sloupce, platí $\boldsymbol{H}\mathbf{e}_{ij}^{\mathrm{T}} \neq \boldsymbol{o}^{\mathrm{T}}$, a tedy $\mathbf{e}_{ij}$ není kódové slovo.

14.2. Úloha. Obecněji ověřte, že q-znakový lineární kód opravuje jednoduché chyby, právě když žádný sloupec kontrolní matice není (skalárním) násobkem jiného sloupce.

14.3. Pozorování. Předchozí tvrzení dává návod, jak sestrojit binární kód, který opravuje jednoduché chyby a má při daném počtu m kontrolních znaků co nejmenší redundanci (tj. co nejvíce informačních znaků anebo, což je totéž, co největší délku): Za kontrolní matici volíme takovou matici, která jako sloupce obsahuje všechna nenulová slova délky m.

14.4. Definice. Binární kód se nazývá *Hammingův*, jestliže má kontrolní matici, jejíž sloupce jsou všechna nenulová slova dané délky a žádné z nich se neopakuje.

14.5. Příklady. Na uspořádání sloupců kontrolní matice nezáleží. Použijeme zde uspořádání takové, aby sloupce byly binárními rozvoji čísel 1, 2, 3, ... (Jiné důležité uspořádání poznáme v teorii cyklických kódů.)

14.5.1. Pro $m = 2$ máme kontrolní matici

$$\boldsymbol{H} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

Jde o (3, 2)-kód, popsaný rovnicemi $v_2 + v_3 = 0$ a $v_1 + v_3 = 0$, tedy opakovací kód délky 3. Ten ovšem opravuje jednoduché chyby.

14.5.2. Nechť $m = 3$. Jde o (7, 4)-kód z příkl. 13.13: jeho kontrolní matice je

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Abychom našli jeho generující matici, přemístíme sloupce matice $\boldsymbol{H}$ tak, aby výsledná matice byla tvaru $[\boldsymbol{B}^{\mathrm{T}} \mid \boldsymbol{E}]$, kde $\boldsymbol{E}$ je jednotková matice: prohodíme první a poslední sloupec, dále druhý a předposlední a nakonec čtvrtý s pátým. Vznikne matice

$$\boldsymbol{H}' = \left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right].$$

To je matice ekvivalentního Hammingova kódu, který má (podle 11.3) generující matici

$$\boldsymbol{G}' = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right].$$

Generující matici původního kódu získáme zpětným přehozením sloupců:

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

Z hlediska kódování informačních znaků je výhodnější tvar matice $\boldsymbol{G}'$, protože jde o systematický Hammingův kód. Z hlediska dekódování je, jak uvidíme, výhodnější tvar matice $\boldsymbol{G}$, přesněji, odpovídající matice $\boldsymbol{H}$ (protože generující matici vůbec nemusíme znát).

14.5.3. Nechť $m = 4$. Jde o (15, 11)-kód s touto kontrolní maticí:

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

14.6. Dekódování. Binární Hammingův kód nejen opravuje jednoduché chyby, ale opravuje je snadno. Předpokládejme, že sloupce matice $\boldsymbol{H}$ jsou uspořádány tak, že tvoří binární rozvoje čísel $1, 2, \ldots, 2^m - 1$. Přijmeme vektor $\boldsymbol{w}$ a zjistíme jeho syndrom $\boldsymbol{s}$, kde $\boldsymbol{s}^{\mathrm{T}} = \boldsymbol{H}\boldsymbol{w}^{\mathrm{T}}$. Slovo $\boldsymbol{s}$ je binárním zápisem čísla i a my změníme i-tý znak přijatého slova (pokud $\boldsymbol{s} \neq \boldsymbol{o}$, v opačném případě je $\boldsymbol{w}$ kódovým slovem). Stručněji,

$$\delta(\boldsymbol{w}) = \begin{cases} \boldsymbol{w}, & \text{pokud } \boldsymbol{s} = \boldsymbol{o}, \\ \boldsymbol{w} - \boldsymbol{e}_i, & \text{pokud } \boldsymbol{s} \text{ je binárním rozvojem čísla } i, \end{cases}$$

kde $\boldsymbol{e}_i$ je slovo s jednou jedničkou na i-tém místě.

14.7. Tvrzení. Uvedené dekódování je správné v případě jednoduché chyby. Podrobněji: jestliže se dané slovo $\boldsymbol{w}$ liší od některého kódového slova $\boldsymbol{v}$ nejvýše v jednom znaku, platí $\delta(\boldsymbol{w}) = \boldsymbol{v}$.

Důkaz. Pokud se slova $\boldsymbol{v}$ a $\boldsymbol{w}$ neliší v žádném znaku, platí $\boldsymbol{s}^{\mathrm{T}} = \boldsymbol{H}\boldsymbol{v}^{\mathrm{T}} = \boldsymbol{o}^{\mathrm{T}}$ a správně dekódujeme $\delta(\boldsymbol{w}) = \boldsymbol{w} = \boldsymbol{v}$. Pokud se liší v jediném, i-tém znaku, platí $\boldsymbol{w} = \boldsymbol{v} + \boldsymbol{e}_i$, a tedy $\boldsymbol{s}^{\mathrm{T}} = \boldsymbol{H}\boldsymbol{w}^{\mathrm{T}} = \boldsymbol{H}\boldsymbol{v}^{\mathrm{T}} + \boldsymbol{H}\boldsymbol{e}_i^{\mathrm{T}} = \boldsymbol{H}\boldsymbol{e}_i^{\mathrm{T}}$. Protože $\boldsymbol{H}\boldsymbol{e}_i^{\mathrm{T}}$ je i-tý sloupec matice $\boldsymbol{H}$ a ten je rozvojem čísla i, dekódovali jsme opět správně.

14.8. Příklad. Při použití Hammingova (7, 4)-kódu dekódujeme slovo 1110111. Syndrom je

$$\boldsymbol{H}\begin{bmatrix}1\\1\\1\\0\\1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\0\\0\end{bmatrix},$$

tedy binární zápis čísla 4. Opravíme čtvrtý znak: 1111111.

14.9. Úloha. Při použití Hammingova (15, 11)-kódu dekódujte slovo 001001000000001 .

14.10. Perfektní kódy. Hammingovy kódy opravují jednoduché chyby a mají navíc nejmenší myslitelnou redundanci. Například binární kód K délky 7, opravující jednoduché chyby, musí mít standardní dekódování s reprezentanty $\boldsymbol{o}, \boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_7$ (viz 13.8). Jestliže nemá žádné další reprezentanty, je ovšem Hammingovým (7, 4)-kódem. Jestliže má více reprezentantů tříd, je (7, k)-kódem pro některé $k < 4$, a tedy má zbytečně velkou redundanci.

Obecněji, lineární kód opravující t-násobné chyby musí mít za reprezentanty tříd všechna slova váhy $\leqq t$. Ty kódy, které žádné další reprezentanty nemají, se nazývají perfektní:

14.11. Definice. Lineární kód je *perfektní* pro t-násobné opravy, jestliže množina všech slov váhy $\leqq t$ tvoří systém reprezentantů jeho tříd.

14.12. Poznámka. Opakovací kód délky $2t + 1$ je perfektní pro t-násobné opravy. To snadno ověříte. Kromě Hammingových kódů ($t = 1$) a opakovacích kódů již existuje jen překvapivě málo perfektních kódů. To přesně zformulujeme v kapitole věnované Golayově kódu.

14.13. Tvrzení. Hammingovy binární kódy jsou perfektní kódy pro jednoduché opravy. Tato vlastnost je charakterizuje, tj. každý perfektní binární kód pro jednoduché opravy je Hammingův kód.

Poznámka. Odtud plyne, že každý lineární binární $(2^m - 1, 2^m - m - 1)$-kód minimální vzdálenosti $\geqq 3$ je Hammingův kód.

Důkaz. Hammingův kód K délky $2^m - 1$ má m kontrolních znaků, a tedy 2^m tříd (13.1). Jaké mají reprezentanty: třída K má reprezentanta $\mathbf{e}_0 = 00...0$. Další třídy mají reprezentanty váhy 1, protože reprezentant musí mít minimální váhu a víme, že každá dvě slova $\mathbf{e}_i$ a $\mathbf{e}_j$, kde $i \neq j$, leží v různých třídách. (Kdyby ležela ve stejné třídě, bylo by $\mathbf{e}_i - \mathbf{e}_j$ kódové slovo váhy 2, a takové neexistuje.) Máme tedy 2^m reprezentantů $\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{2^m-1}$. Protože jejich počet je shodný s počtem tříd, jsou to jediní reprezentanti.

Naopak, jestliže K je binární kód délky n, který je perfektní pro jednoduché opravy, ukážeme, že K je Hammingův kód. Označme m počet kontrolních znaků. Potom má kód K kontrolní matici $\boldsymbol{H}$ typu (m, n). Podle 14.1 jsou každé dva sloupce matice $\boldsymbol{H}$ různé a nenulové, takže celkový počet sloupců nemůže překročit číslo $2^m - 1$, tj.

$$n \leqq 2^m - 1 .$$

Na druhé straně je počet tříd kódu K (což je číslo 2^m, viz 13.1) roven počtu slov váhy 0 nebo 1, a to je $n + 1$. Z toho plyne $n = 2^m - 1$, a matice $\boldsymbol{H}$ tedy obsahuje jako sloupce všechna nenulová slova délky m. Proto je K Hammingův kód.

14.14. Rozšířený Hammingův kód. To je binární kód, vzniklý rozšířením Hammingova kódu o znak celkové kontroly parity. Je to tedy $(2^m, 2^m - m - 1)$-kód všech slov $v_1 v_2 \ldots v_{2^m}$ takových, že $v_1 v_2 \ldots v_{2^m-1}$ je kódové slovo Hammingova kódu a $v_{2^m} = v_1 + v_2 + \ldots + v_{2^m-1}$.

Minimální váha Hammingova kódu je 3 (ověřte!), a tedy minimální váha rozšířeného Hammingova kódu je 4. Tento kód opravuje jednoduché chyby a objevuje trojnásobné chyby.

14.15. Úlohy

14.15.1. Napište kontrolní matici rozšířeného Hammingova (8, 4)-kódu. Dekodujte slovo 10111101. Je jisté, že jste dekódovali správně?

14.15.2. Ověřte, že rozšířený Hammingův (8, 4)-kód je samoduální (11.9), tj. kontrolní matice je maticí generující. (Návod: protože dimenze obou kódů je 4, stačí ověřit, že skalární součin libovolných dvou řádků je roven 0.)

14.15.3. Napište kontrolní matici rozšířeného Hammingova (16, 11)-kódu. Proč není tento kód samoduální?

14.16. Víceznakové Hammingovy kódy. Úloha 14.2 dává návod, je definovat Hammingův kód v abecedě T. Je to kód s kontrolní maticí $\boldsymbol{H}$ takovou, že (i) žádný sloupec není skalárním násobkem jiného sloupce, ale (ii) každé nenulové slovo příslušné délky je skalárním násobkem sloupce matice $\boldsymbol{H}$.

Matici $\boldsymbol{H}$ můžeme např. sestavit ze všech slov dané délky, která mají první nenulový znak roven 1.

14.17. Příklad: Ternární Hammingovy kódy. Pro $m = 2$ máme kontrolní matici

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix}.$$

Jde tedy o (4, 2)-kód. Pro $m = 3$ je kontrolní matice

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 2 & 2 & 1 & 2 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 2 & 2 & 0 & 1 & 2 & 2 \end{bmatrix}$$

a vzniká (13, 3)-kód.

14.18. Úlohy

14.18.1. Při použití ternárního Hammingova (4, 2)-kódu dekódujte slova 1220 a 1101.

14.18.2. Ověřte, že ternární Hammingův kód s m kontrolními znaky má délku $(3^m - 1)/2$.

14.18.3. Zobecněte na q-znakové kódy. Dokažte, že jsou perfektní.

15. Golayův kód — perfektní kód pro trojnásobné opravy

Mezi nejvýznamnější binární kódy patří Golayův (čti golejův) kód G_{23}, který má délku 23, z toho 12 informačních a 11 kontrolních znaků. Tento kód je perfektní pro opravy trojnásobných chyb. Hluboký výsledek teorie kódování je ten, že kromě Hammingových a opakovacích kódů je G_{23} jediný perfektní binární kód.

Kód G_{23} zavedeme pomocí generující matice. Budeme také pracovat s rozšířeným Golayovým kódem G_{24}. Ukážeme jednoduchý postup dekódování těchto kódů. Zmíníme se o ternárním Golayově kódu G_{11} a o perfektnosti q-znakových kódů.

15.1. Definice kódů G_{23} a G_{24}. *Golayovy kódy* zavedeme jako systematické binární kódy délky 23 a 24. Levou polovinu generující matice tedy tvoří matice jednotková. Pravá polovina sestává ze čtvercové matice $\boldsymbol{B}$ řádu 11, doplněné o řádek 11...1 v případě G_{23}:

$$G_{23} = \left[\begin{array}{c|c} \boldsymbol{E} & \boldsymbol{B} \\ \hline & 11\ldots11 \end{array}\right]$$

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}
r_0	1												1	1		1	1	1				1		1
r_1		1												1	1		1	1	1				1	1
r_2			1										1		1	1		1	1	1				1
r_3				1										1		1	1		1	1	1			1
r_4					1										1		1	1		1	1	1		1
r_5						1										1		1	1		1	1	1	1
r_6							1						1				1		1	1		1	1	1
r_7								1					1	1				1		1	1		1	1
r_8									1				1	1	1				1		1	1		1
r_9										1				1	1	1				1		1	1	1
r_{10}											1		1		1	1	1				1		1	1
r_{11}												1	1	1	1	1	1	1	1	1	1	1	1	

Generující matice G_{24} rozšířeného Golayova kódu

a ještě o sloupec 11...10 v případě G_{24}:

$$\boldsymbol{G}_{24} = \left[\begin{array}{c|c|c} & & 1 \\ \boldsymbol{E} & \boldsymbol{B} & 1 \\ & & \dots \\ & & 1 \\ \hline & 11\dots11 & 0 \end{array}\right].$$

Matice $\boldsymbol{B}$ vznikne cyklickými posuvy svého prvého řádku, což je slovo

11011100010.

(Toto slovo má jedničku na místě $i = 0, 1, \dots, 10$, právě když je i čtvercem modulo 11, tj. pro $0^2, 1^2, 2^2, 3^2, 4^2 \equiv 5$ a $5^2 \equiv 3$.) Zde je celá matice:

$$\boldsymbol{B} = \begin{array}{c} \begin{array}{ccccccccccc} \boldsymbol{b}_0 & \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_3 & \boldsymbol{b}_4 & \boldsymbol{b}_5 & \boldsymbol{b}_6 & \boldsymbol{b}_7 & \boldsymbol{b}_8 & \boldsymbol{b}_9 & \boldsymbol{b}_{10} \end{array} \\ \left[\begin{array}{ccccccccccc} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right] \end{array}.$$

Kód G_{24} vznikne tak, že ke kódu G_{23} přidáme celkovou kontrolu parity. Celá matice $\boldsymbol{G}_{24}$ (s vynecháním zápisu nul kvůli přehlednosti) je uvedena na str. 71.

15.2. Vlastnosti kódu G_{24}

15.2.1. Počet informačních znaků je 12, protože levou část matice $\boldsymbol{G}_{24}$ tvoří jednotková matice, takže hodnost matice $\boldsymbol{G}_{24}$ je 12.

15.2.2. Kód G_{24} je samoduální, tj. $G_{24}^{\perp} = G_{24}$ (11.9). Abychom to ověřili, stačí ukázat, že skalární součin dvou libovolných řádků matice $\boldsymbol{G}_{24}$ je nulový, tj.

$\boldsymbol{r}_i * \boldsymbol{r}_j = 0$.

Potom totiž lineární prostor G_{24} je podprostorem duálu $G_{24}^{\perp}$, a protože dimenze duálu je $24 - 12 = 12$ (stejná jako dimenze prostoru G_{24}), plyne odtud $G_{24}^{\perp} = G_{24}$.

Pokud $i = j$, všimneme si, že každý řádek $\boldsymbol{r}_i$ matice $\boldsymbol{G}_{24}$ má sudou váhu: 8 pro $i = 0, \dots, 10$ a 12 pro $i = 11$. To znamená, že součin $\boldsymbol{r}_i * \boldsymbol{r}_i$ je roven součtu 8 nebo 12 jedniček, a to je 0. Pro $i \neq j$ ověříme, že řádky $\boldsymbol{r}_i$ a $\boldsymbol{r}_j$ mají sudý počet společných jedniček – potom $\boldsymbol{r}_i * \boldsymbol{r}_j = 0$. Pokud $i = 11$ nebo $j = 11$, je počet společných jedniček 6. V případě $i \neq 11 \neq j$ nemáme žádnou společnou jedničku v levé části (tj. ve sloupcích $\boldsymbol{a}_0, \dots, \boldsymbol{a}_{11}$) a v pravé části je jedna společná jednička v posledním sloupci. Stačí tedy ukázat, že dva různé řádky matice $\boldsymbol{B}$ mají lichý počet společných jedniček:

15.2.3. Každé dva různé řádky matice $\boldsymbol{B}$ mají právě 3 společné jedničky. To ověříte mechanicky: stačí se omezit na případ prvého řádku a některého z 10 zbývajících řádků.

15.2.4. Kód G_{24} neporuší tato permutace sloupců:

$$a_0 \leftrightarrow b_0\,, \quad a_{11} \leftrightarrow b_{11} \quad \text{a} \quad a_i \leftrightarrow b_{11-i} \quad \text{pro} \quad i = 1, 2, \ldots, 10\,.$$

Přesněji, pro každé kódové slovo

$$\boldsymbol{w} = w_0 w_1 w_2 \ldots w_{10} w_{11} w_0' w_1' w_2' \ldots w_{10}' w_{11}'$$

je také následující slovo kódovým slovem:

$$\boldsymbol{w}^* = w_0' w_{10}' w_9' \ldots w_2' w_1' w_{11}' w_0 w_{10} w_9 \ldots w_2 w_1 w_{11}\,.$$

Jistě stačí, když toto tvrzení ověříme pro řádky matice $\boldsymbol{G}_{24}$. Mechanicky se přesvědčíte o tom, že platí

$$\boldsymbol{r}_0^* = 101000111011\ 100000000000 =$$
$$= \boldsymbol{r}_0 + \boldsymbol{r}_2 + \boldsymbol{r}_6 + \boldsymbol{r}_7 + \boldsymbol{r}_8 + \boldsymbol{r}_{10} + \boldsymbol{r}_{11}\,.$$

Je tedy $\boldsymbol{r}_0^*$ kódové slovo. Další řádky $\boldsymbol{r}_i$, $i \neq 11$, vzniknou cyklickým posuvem levé i pravé části řádku $\boldsymbol{r}_0^*$ a odtud je zřejmé, že $\boldsymbol{r}_i^*$ jsou kódová slova. Nakonec

$$\boldsymbol{r}_{11}^* = 111\ldots110\,00\ldots001 = \boldsymbol{r}_0 + \boldsymbol{r}_1 + \ldots + \boldsymbol{r}_9 + \boldsymbol{r}_{10}\,.$$

15.2.5. Minimální vzdálenost je 8.

Zvolme nenulové kódové slovo $\boldsymbol{w}$ a ověřme, že $\|\boldsymbol{w}\| \geqq 8$, viz 12.3. Protože je kód G_{24} samoduální, je $\|\boldsymbol{w}\|$ sudé číslo. Kdyby platilo $\|\boldsymbol{w}\| = 2$, mělo by slovo $\boldsymbol{w}$ v levé části buď jedinou jedničku, ale potom $\boldsymbol{w} = \boldsymbol{r}_i$ a $\|\boldsymbol{w}\| = 8$, nebo by zde mělo dvě jedničky (na místech i a j). Potom $\boldsymbol{w} = \boldsymbol{r}_i + \boldsymbol{r}_j$, ale z 15.2.3 plyne $\|\boldsymbol{r}_i + \boldsymbol{r}_j\| \geqq 8$. Kdyby platilo $\|\boldsymbol{w}\| = 4$, potom by buď slovo $\boldsymbol{w}$ nebo slovo $\boldsymbol{w}^*$ (viz 15.2.4) mělo v levé části jen dvě jedničky, a protože $\|\boldsymbol{w}\| = \|\boldsymbol{w}^*\|$, plynulo by odtud $\|\boldsymbol{w}\| = 8$.

Zbývá vyloučit případ $\|\boldsymbol{w}\| = 6$. Jediná situace, kterou nevyřeší stejná úvaha jako v případě váhy 4, je ta, že slovo $\boldsymbol{w}$ má 3 jedničky vlevo a 3 vpravo, tedy $\boldsymbol{w} = \boldsymbol{r}_i + \boldsymbol{r}_j + \boldsymbol{r}_k$. Rozborem možností, kolik jedniček mají společně slova $\boldsymbol{r}_i$ a $\boldsymbol{r}_j + \boldsymbol{r}_k$, se (dosti pracně) ověří, že pro libovolná i, j a k platí $\|\boldsymbol{r}_i + \boldsymbol{r}_j + \boldsymbol{r}_k\| \geqq 8$.

15.3. Důsledek. Golayův kód G_{23} je (23, 12)-kód, který je perfektní (14.11) pro trojnásobné opravy.

Skutečně, protože má rozšířený kód minimální vzdálenost 8, kód G_{23} má minimální vzdálenost $\geqq 7$, takže opravuje trojnásobné chyby (5.1). Ve standardním dekódování jsou tedy slova váhy $\leqq 3$ reprezentanty tříd (13.7). Počet těchto slov je

$$\binom{23}{0} + \binom{23}{1} + \binom{23}{2} + \binom{23}{3} = 2\,048 = 2^{11}\,.$$

Počet tříd (23, 12)-kódu je také 2^{11} (13.1). To znamená, že reprezentanty jsou právě všechna slova váhy $\leqq 3$.

15.4. Věta (Tietäväinenova a Van Lintova).
Jediné netriviální perfektní binární kódy jsou tyto:

a) *Hammingovy kódy pro jednoduché chyby,*

b) *Golayův kód pro trojnásobné chyby a kódy s ním ekvivalentní,*

c) *opakovací kódy délky* $2t + 1$ *(pro* t*-násobné chyby), kde* $t = 1, 2, 3, \ldots$.

Krásný kombinatorický důkaz této věty přesahuje bohužel rámec naší knihy. Čtenář jej může najít v knize [11].

Obdobný výsledek platí pro ternární kódy: Existuje perfektní *Golayův ternární* $(11, 6)$*-kód* opravující trojnásobné chyby. Jeho generující matice je tvaru

$$G_{11} = \left[\begin{array}{c|c} \boldsymbol{E} & \begin{matrix} \boldsymbol{B} \\ \hline 11111 \end{matrix} \end{array}\right],$$

kde $\boldsymbol{E}$ je jednotková matice řádu 6 a $\boldsymbol{B}$ je matice vzniklá cyklickými posuvy slova 01221. Kromě tohoto kódu (a kódů ekvivalentních) jsou jediné perfektní netriviální ternární kódy opakovací kódy a Hammingovy kódy.

V případě abecedy o více než třech znacích je situace ještě jednodušší: jediné perfektní netriviální kódy jsou Hammingovy a opakovací kódy.

15.5. Dekódování Golayova kódu. Ukážeme jednoduchou dekódovací metodu při použití rozšířeného Golayova kódu G_{24}. Naším cílem je při přijetí vektoru $\boldsymbol{w}$ správně určit vyslaný vektor $\boldsymbol{v}$ za předpokladu, že došlo nejvýše ke třem chybám. Přesněji, kdykoli ke slovu $\boldsymbol{w}$ existuje kódové slovo $\boldsymbol{v}$ o vzdálenosti $\leqq 3$ (tj. $\|\boldsymbol{w} - \boldsymbol{v}\| \leqq 3$), máme toto slovo $\boldsymbol{v}$ najít.

Protože je kód G_{24} samoduální (viz 15.2.2), je jeho kontrolní maticí matice $\boldsymbol{G}_{24}$. Označme $\tilde{\boldsymbol{B}}$ matici $\boldsymbol{B}$ s přidaným posledním řádkem a posledním sloupcem tvaru $11\ldots110$. Tedy

$$\boldsymbol{G}_{24} = [\boldsymbol{E} \mid \tilde{\boldsymbol{B}}] .$$

15.5.1. První krok dekódování: výpočet syndromu $\boldsymbol{s}$. Platí

$$\boldsymbol{s}^{\mathrm{T}} = \boldsymbol{H}\boldsymbol{w}^{\mathrm{T}} = (s_0 s_1 \ldots s_{11})^{\mathrm{T}} .$$

Chybový vektor $\mathbf{e} = \boldsymbol{v} - \boldsymbol{w}$ splňuje ovšem vztah (12.4.1),

$$\boldsymbol{s}^{\mathrm{T}} = \boldsymbol{H}\mathbf{e}^{\mathrm{T}} = (e_0 e_1 \ldots e_{11})^{\mathrm{T}} + e_{12}\boldsymbol{b}_0 + e_{13}\boldsymbol{b}_1 + \ldots + e_{23}\boldsymbol{b}_{11}$$

(kde $\boldsymbol{b}_i$ jsou sloupce pravé části matice $\boldsymbol{G}_{24}$, v levé části je jednotková matice). Platí tedy

$$(15.5.1) \qquad (e_0 e_1 \ldots e_{11})^{\mathrm{T}} + \sum_{i=0}^{11} e_{12+i}\boldsymbol{b}_i = (s_0 s_1 \ldots s_{11})^{\mathrm{T}} .$$

15.5.2. Druhý krok dekódování: výpočet pomocného slova $\mathbf{t}$. Definujeme slovo $\mathbf{t} = t_0 t_1 \ldots t_{11}$ předpisem

$$\mathbf{t} = \mathbf{s}\tilde{\boldsymbol{B}} .$$

Potom pro řádky $\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{11}$ matice $\tilde{\boldsymbol{B}}$ platí:

$$(15.5.2) \qquad e_{12}e_{13}\ldots e_{23} + \sum_{i=0}^{11} e_i\mathbf{c}_i = t_0 t_1 \ldots t_{11} .$$

Důkaz. Nejprve ověříme, že $\tilde{\boldsymbol{B}}^{\mathrm{T}}\tilde{\boldsymbol{B}}$ je jednotková matice. Protože je kód G_{24} samoduální, je matice $\boldsymbol{G}_{24}$ generující i kontrolní maticí, takže součin $\boldsymbol{G}_{24}\boldsymbol{G}_{24}^{\mathrm{T}}$ je

nulovou maticí. Platí tedy

$$\mathbf{0} = \mathbf{G}_{24}\mathbf{G}_{24}^{\mathrm{T}} = [\mathbf{E} \mid \widetilde{\mathbf{B}}] \begin{bmatrix} \mathbf{E} \\ \hline \widetilde{\mathbf{B}}^{\mathrm{T}} \end{bmatrix} = \mathbf{E} + \widetilde{\mathbf{B}}\widetilde{\mathbf{B}}^{\mathrm{T}}.$$

Odtud $\widetilde{\mathbf{B}}\widetilde{\mathbf{B}}^{\mathrm{T}} = -\mathbf{E} = \mathbf{E}$, a tedy i $\widetilde{\mathbf{B}}^{\mathrm{T}}\widetilde{\mathbf{B}} = \mathbf{E}$. Platí

$$\mathbf{t}^{\mathrm{T}} = \widetilde{\mathbf{B}}^{\mathrm{T}}\mathbf{s}^{\mathrm{T}} = \widetilde{\mathbf{B}}^{\mathrm{T}}\mathbf{G}_{24}\mathbf{e}^{\mathrm{T}} = \widetilde{\mathbf{B}}^{\mathrm{T}}[\mathbf{E} \mid \widetilde{\mathbf{B}}]\,\mathbf{e}^{\mathrm{T}} = [\widetilde{\mathbf{B}}^{\mathrm{T}} \mid \mathbf{E}]\,\mathbf{e}^{\mathrm{T}},$$

a to je právě dokazovaná rovnost.

15.5.3. Třetí krok dekódování: určení chybového slova $\mathbf{e}$. Prozkoumáním váhy 26 slov: $\mathbf{s}$, $\mathbf{t}$, $\mathbf{s} + \mathbf{b}_i^{\mathrm{T}}$ a $\mathbf{t} + \mathbf{c}_i$ pro $i = 0, 1, \ldots, 11$ jsme schopni určit slovo $\mathbf{e}$, o kterém dokážeme, že je rovno skutečnému chybovému slovu $\hat{\mathbf{e}} = \mathbf{w} - \mathbf{v}$. Předpokládáme ovšem stále, že došlo k trojnásobné chybě, tj. $\|\hat{\mathbf{e}}\| \leqq 3$. Dekódování je založeno na tom, že buď v pravé nebo v levé polovině slova $\mathbf{w}$ došlo k nejvýše jedné chybě (jinak $\|\hat{\mathbf{e}}\| \geqq 2 + 2$).

a) Kdykoli platí $\|\mathbf{s}\| \leqq 3$, položme

$$\mathbf{e} = \mathbf{s} \mid \mathbf{o} = s_0 s_1 s_2 \ldots s_{10} s_{11} 000 \ldots 00 .$$

Tento předpis je správný, pokud v pravé polovině slova nedošlo k žádné chybě: potom $\hat{e}_i = 0$ pro $i = 12, \ldots, 23$ a podle (15.5.1) platí $\hat{e}_0\hat{e}_1 \ldots \hat{e}_{11} = s_0 s_1 \ldots s_{11}$, takže $\hat{\mathbf{e}} = \mathbf{s} \mid \mathbf{o} = \mathbf{e}$.

Během rozboru dalších případů ukážeme, že pokud v pravé polovině slova došlo k chybě, je $\|\mathbf{s}\| > 3$.

b) Kdykoli platí $\|\mathbf{s} + \mathbf{b}_i^{\mathrm{T}}\| \leqq 2$ (pro některé $i = 0, \ldots, 11$), položme

$$\mathbf{e} = (\mathbf{s} + \mathbf{b}_i^{\mathrm{T}}) \mid \mathbf{d}_i ,$$

kde $\mathbf{d}_i$ je slovo délky 12 s jedinou jedničkou na i-tém místě. Tento předpis je správný, pokud v pravé polovině slova došlo k jediné chybě, tj. $\hat{e}_{12+i} = 1$ pro právě jedno $i = 0, 1, \ldots, 11$. Potom pravou polovinu slova $\hat{\mathbf{e}}$ tvoří slovo $\mathbf{d}_i$. Pro levou polovinu platí (15.5.1), $\hat{e}_0\hat{e}_1 \ldots \hat{e}_{11} = (s_0 s_1 \ldots s_{11}) + \mathbf{b}_i^{\mathrm{T}}$, takže $\hat{\mathbf{e}} = (\mathbf{s} + \mathbf{b}_i^{\mathrm{T}}) \mid \mathbf{d}_i = \mathbf{e}$. Navíc máme v případě jedné chyby v pravé polovině nerovnost $\|\mathbf{s}\| > 3$. (To plyne z faktu, že každý sloupec $\mathbf{b}_i$ matice $\mathbf{B}$ má váhu 7 nebo 11, takže slovo $\mathbf{s} + \mathbf{b}_i^{\mathrm{T}}$ má váhu alespoň $7 - \|\mathbf{s}\|$ a platí $7 - \|\mathbf{s}\| < \|(\mathbf{s} + \mathbf{b}_i^{\mathrm{T}}) \mid \mathbf{d}_i\| = \|\hat{\mathbf{e}}\| \leqq 3$.)

Během rozboru dalších případů ukážeme, že pokud v pravé polovině slova došlo k 2 nebo 3 chybám (tj. pokud v levé polovině došlo k nejvýše jedné chybě), je $\|\mathbf{s} + \mathbf{b}_i^{\mathrm{T}}\| > 2$ pro všechna i.

c) Kdykoli platí $\|\mathbf{t}\| \leqq 3$, položme

$$\mathbf{e} = \mathbf{o} \mid \mathbf{t} = 000 \ldots 00\ t_0 t_1 \ldots t_{11} .$$

Tento předpis je správný, pokud v levé polovině slova nedošlo k chybě — viz (15.5.2). Navíc v případě jedné chyby v pravé polovině platí podle (15.5.1) $\mathbf{s} = \mathbf{b}_i^{\mathrm{T}}$, a tedy $\|\mathbf{s}\| \geqq 7$. V případě dvou chyb v pravé polovině platí $\mathbf{s} = \mathbf{b}_k^{\mathrm{T}} + \mathbf{b}_j^{\mathrm{T}}$ $(k \neq j)$, a tedy $\|\mathbf{s}\| \geqq 3$ (snadno se přesvědčíte, že součet dvou sloupců matice $\widetilde{\mathbf{B}}$ má váhu alespoň 6 — viz podobný rozbor v (15.2.2)). Dále pro každé i je $\|\mathbf{s} + \mathbf{b}_i^{\mathrm{T}}\| = \|\mathbf{b}_k^{\mathrm{T}} + \mathbf{b}_j^{\mathrm{T}} + \mathbf{b}_i^{\mathrm{T}}\| \geqq$ $\geqq 4$, o tom se lze také mechanicky přesvědčit.

Pokud v levé polovině slova došlo k chybě, platí $\|\mathbf{t}\| > 3$. Důkaz je analogický důkazu nerovnosti $\|\mathbf{s}\| > 3$ pro případ pravé poloviny.

d) Kdykoli platí $\|\mathbf{t} + \mathbf{c}_i\| \leqq 2$ (pro některé $i = 0, \ldots, 11$), položme

$$\mathbf{e} = \mathbf{d}_i \mid (\mathbf{t} + \mathbf{c}_i) .$$

Tento předpis je správný, pokud v levé polovině slova došlo k jediné chybě, tj. $\hat{\mathbf{e}}_i = 1$ pro právě jedno $i = 0, \ldots, 11$ – viz (15.5.2). Navíc v případě jedné chyby v pravé polovině platí podle (15.5.1) $\mathbf{s} = \mathbf{d}_i + \mathbf{b}_j$, a tedy $\|\mathbf{s}\| \geqq 7 - 1 = 6$. V případě dvou chyb v pravé polovině platí $\mathbf{s} = \mathbf{d}_i + \mathbf{b}_k^{\mathrm{T}} + \mathbf{b}_j^{\mathrm{T}}$ $(k \neq j)$, a tedy $\|\mathbf{s}\| \geqq 3$ a $\|\mathbf{s} + \mathbf{b}_i^{\mathrm{T}}\| \geqq \|\mathbf{b}_k^{\mathrm{T}} + \mathbf{b}_j^{\mathrm{T}} + \mathbf{b}_i^{\mathrm{T}}\| - 1 \geqq 3$.

Pokud v levé polovině slova došlo k více než jedné chybě, platí $\|\mathbf{t} + \mathbf{c}_i\| > 2$. Důkaz je analogický důkazu nerovnosti $\|\mathbf{s} + \mathbf{b}_i^{\mathrm{T}}\| > 2$ pro případ pravé poloviny.

15.5.4. Příklad. Dekódujeme slovo

000000000000000111000101 ,

tedy slovo $\mathbf{w}$ takové, že $w_i = 1$ právě pro $i = 15, 16, 17, 21$ a 23. Platí

$$\mathbf{s} = (\mathbf{b}_3 + \mathbf{b}_4 + \mathbf{b}_5 + \mathbf{b}_9 + \mathbf{b}_{11})^{\mathrm{T}} = 111100100110 ,$$

$$\mathbf{t} = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3 + \mathbf{c}_6 + \mathbf{c}_9 + \mathbf{c}_{10} = 000111000101 .$$

Mechanicky zjistíme, že $\|\mathbf{s} + \mathbf{b}_i^{\mathrm{T}}\| > 2$ pro všechna i. Platí však

$$\mathbf{t} + \mathbf{c}_0 = 110000000000 ,$$

a to je slovo váhy 2, takže

$$\mathbf{e} = \mathbf{d}_0 \mid (\mathbf{t} + \mathbf{c}_0) = 100000000000 \mid 110000000000 .$$

Pokud došlo k trojnásobné chybě, jsou chybná místa w_0, w_{12} a w_{13} a vyslán byl vektor

$$\mathbf{v} = \mathbf{w} + \mathbf{e} = 100000000000 \mid 110111000101 .$$

Všimněte si, že výsledek je prvním řádkem $\mathbf{r}_0$ matice $\mathbf{G}_{24}$, takže jde o kódové slovo. Protože se liší od slova $\mathbf{w}$ ve třech znacích, dekódovali jsme správně.

15.5.5. Úloha. Dekódujte slovo 111000000000 | 000000000000 .

16. Konstrukce kódů

V tomto článku stručně probereme konstrukce, upravující daný lineární kód. Takové konstrukce jsou důležité zejména tehdy, když některé parametry kódu jsou předem určeny a my musíme upravit známý „dobrý" kód tak, aby daným parametrům vyhovoval. (Například může z technických důvodů být stanovena délka kódového slova nebo informační poměr.) Uvádíme i anglický název konstrukcí. Omezujeme se pro přehlednost na binární lineární kódy.

16.1. Rozšíření kódu [extension]. K danému kódu přidáme znak celkové kontroly parity. To znamená, že z (n, k)-kódu K vytváříme $(n + 1, k)$-kód K všech slov

$$v_1 v_2 \ldots v_n v_{n+1}$$

takových, že platí

$$v_1 v_2 \ldots v_n \in K \quad \text{a} \quad v_{n+1} = v_1 + v_2 + \ldots + v_n .$$

Pokud má kód K kontrolní matici $\boldsymbol{H}$, má rozšířený kód K^* kontrolní matici

$$\boldsymbol{H}^* = \left[\begin{array}{ccc|c} & & & 0 \\ & \boldsymbol{H} & & 0 \\ & & & \ldots \\ & & & 0 \\ \hline 1 & 1 \ \ldots & 1 & 1 \end{array}\right].$$

Všechny původní kontrolní rovnice totiž zůstávají, ovšem bez proměnné v_{n+1}, a navíc máme rovnici $v_1 + v_2 + \ldots + v_n + v_{n+1} = 0$.

16.1.1. Příklad. Rozšířený Hammingův (8, 4)-kód (14.13) má kontrolní matici

$$\boldsymbol{H}^* = \left[\begin{array}{ccccccc|c} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right].$$

16.1.2. Pozorování. Jestliže má binární kód K minimální vzdálenost d, má rozšířený kód minimální vzdálenost

$d + 1$, pokud je d liché číslo,
d, pokud je d sudé číslo.

Důkaz. Buď $v_1 v_2 \ldots v_n$ nenulové kódové slovo minimální váhy d (viz 12.3). Je-li d sudé číslo, je $v_1 v_2 \ldots v_n 0$ slovo rozšířeného kódu. Je-li d liché číslo, je $v_1 v_2 \ldots v_n 1$ slovo rozšířeného kódu. V obou případech jde o nenulová slova s nejmenší váhou.

16.2. Zúžení kódu [puncturing]. Jde o vypuštění některé souřadnice ze všech kódových slov. Pro každý (n, k)-kód K a každé $i = 1, \ldots, n$ máme tedy zúžený kód všech slov $v_1 v_2 \ldots v_{i-1} v_{i+1} \ldots v_n$ vzniklých ze slov kódu K. To je buď $(n - 1, k)$-kód nebo $(n - 1, k - 1)$-kód. Jeho minimální vzdálenost je buď d (tj. minimální vzdálenost původního kódu K) nebo $d - 1$.

Například zúžením rozšířeného Hammingova (8, 4)-kódu K^* vznikne Hammingův (7, 4)-kód K ($i = 8$). Dalším zúžením ($i = 7$) vznikne lineární (6, 3)-kód $\hat{K}$ s touto kontrolní maticí:

$$\hat{\boldsymbol{H}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

16.3. Zvětšení kódu [augmentation]. Obecně je to postup přidání nových slov ke kódovým slovům. Konkrétněji se zvětšením kódu K rozumí obohacení o slovo $\mathbf{1} = 111\ldots1$, a tedy i slova $\mathbf{v} + \mathbf{1}$, kde $\mathbf{v} \in K$. (Všimněte si, že $\mathbf{v} + \mathbf{1}$ je „opačné" slovo ke slovu $\mathbf{v}$: má jedničky právě tam, kde má slovo $\mathbf{v}$ nuly.)

16.3.1. Příklad. Zvětšením (3, 2)-kódu s generující maticí

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

vznikne tento kód:

$$\begin{array}{ll} 0\,0\,0 & 1\,1\,1 \\ 1\,0\,1 & 0\,1\,0 \\ 1\,1\,0 & 0\,0\,1 \\ 0\,1\,1 & 1\,0\,0 \end{array} \qquad (16.3.1)$$

16.3.2. Úloha. Je-li K binární (n, k)-kód takový, že $\mathbf{1}$ není kódové slovo, ověřte, že zvětšený kód je $(n, k + 1)$-kódem s minimální vzdáleností

$$\min(d, n - d^{\perp}),$$

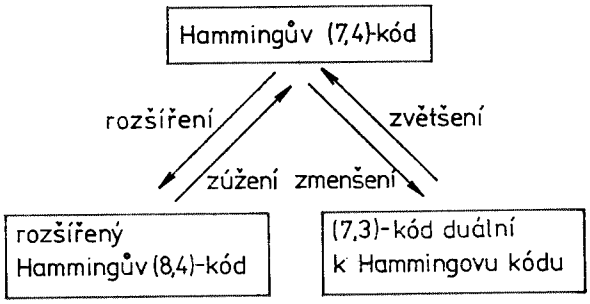
kde $d^{\perp}$ je minimální vzdálenost duálního kódu (11.5).

16.4. Zmenšení kódu [expurgation]. Je to obecně postup odstranění některých kódových slov. Konkrétněji se zmenšením kódu K rozumí odstranění všech slov liché váhy. Tím vznikne „poloviční" kód (viz 10.13), pokud kód K vůbec slova liché váhy obsahuje.

Například zmenšením kódu (16.3.1) vznikne kód s generující maticí $\mathbf{G}$.

Zmenšením Hammingova (7, 4)-kódu vznikne (7, 3)-kód všech slov sudé váhy. Jeho generující matici $\tilde{\mathbf{G}}$ obdržíme např. z matice $\mathbf{G}$ v 14.5.2 tak, že poslední řádek přičteme k řádku druhému i třetímu a pak jej vynecháme:

$$\tilde{\mathbf{G}} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$



úpravy Hammingova (7,4)-kódu

Obr. 23

16.5. Úlohy

16.5.1. Ověřte, že zmenšený Hammingův $(7, 3)$-kód je duální k Hammingovu $(7, 4)$-kódu.

16.5.2. Jaký kód vznikne zvětšením kódu s generující maticí $\tilde{\mathbf{G}}$? Jaký kód vznikne rozšířením tohoto kódu?

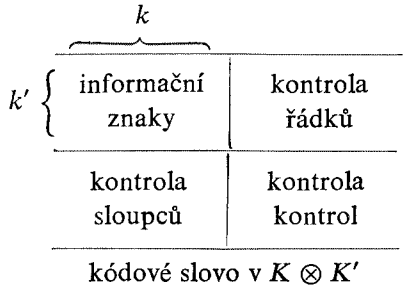
16.6. Direktní součin [direct product]. Direktní součin dvou kódů K (délky n) a K' (délky n') je kód $K \otimes K'$ délky nn'. Pro snadný popis budeme slova délky nn' psát do matic: prvních n znaků tvoří první řádek, druhých n druhý řádek, atd.

Kód $K \otimes K'$ sestává ze všech matic typu (n', n), které v každém řádku mají slovo kódu K a v každém sloupci mají slovo kódu K'. Například, pro každá dvě slova $\mathbf{v} \in K$ a $\mathbf{w} \in K$ můžeme sestavit slovo $\mathbf{v} \otimes \mathbf{w}$ kódu $K \otimes K'$ takto: matice $\mathbf{v} \otimes \mathbf{w}$ má i-tý řádek roven $\mathbf{v}$, pokud $w_i = 1$, a roven $\mathbf{o}$, pokud $w_i = 0$. (Ověřte, že potom j-tý sloupec matice je roven $\mathbf{w}$, pokud $v_j = 1$, a je roven $\mathbf{o}$, pokud $v_j = 0$.)

16.6.1. Příklad. K je kód celkové kontroly parity délky 3 a K' je opakovací kód délky 3. Pak $K \otimes K'$ je tento kód:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

16.6.2. Pozorování. Direktní součin $K \otimes K'$ je (nn', kk')-kód (kde k a k' je počet informačních znaků kódů K a K'). To nejlépe ověříme u systematických kódů (7.4), kde kódová slova direktního součinu mají tuto strukturu:



kódové slovo v $K \otimes K'$

Jestliže kódy K a K' nejsou systematické, použijeme 10.9.

16.6.3. Příklad. Direktní součin kódů celkové kontroly parity je kód dvourozměrné kontroly parity (5.2.3). Například z $(4, 3)$-kódů a $(8, 7)$-kódu vznikne $(32, 21)$-kód.

16.6.4. Tvrzení. Minimální vzdálenost kódu $K \otimes K'$ je součinem minimálních vzdáleností kódů K a K'.

Důkaz. Zvolme slovo $\mathbf{v} \in K$ minimální váhy $d > 0$ a slovo $\mathbf{w} \in K'$ minimální váhy $d' > 0$. Naším úkolem je ověřit, že minimální váha kódu $K \otimes K'$ je dd' (12.3). Slovo $\mathbf{v} \otimes \mathbf{w}$ má váhu dd', takže minimální váha kódu $K \otimes K'$ není větší než dd'. Ale menší také není: zvolme libovolnou nenulovou matici (a_{ij}) v kódu $K \otimes K'$. Existuje tedy nenulový prvek matice. Pokud je v i-tém řádku, je i-tý řádek nenulové slovo kódu K, takže má váhu alespoň d. Zvolme tedy indexy $j_1, j_2, \ldots, j_d$ takové, že platí $a_{ij_m} = 1$ pro $m = 1, 2, \ldots, d$. Pak j_m-tý sloupec matice (a_{ij}) je nenulové kódové slovo v K', a tedy má váhu alespoň d'. To znamená, že váha celé matice je alespoň dd'.

16.7. Direktní součet [direct sum]. Direktní součet kódů K a K' je kód $K \mid K'$ všech slov

$$\mathbf{v} \mid \mathbf{w} = v_1 v_2 \ldots v_n w_1 w_2 \ldots w_m \quad (\mathbf{v} \in K \text{ a } \mathbf{w} \in K').$$

Je zřejmé, že vznikne $(n + n', k + k')$-kód s minimální vzdáleností $\min\{d, d'\}$, kde d a d' označují minimální vzdálenosti kódů K a K'.

Důležitější konstrukce je *modifikovaný direktní součet*: je to kód $K \oplus K'$ všech slov

$$\mathbf{v} \mid (\mathbf{v} + \mathbf{w}) = v_1 v_2 \ldots v_n u_1 u_2 \ldots u_n,$$

kde $\mathbf{v} \in K$, $\mathbf{w} \in K'$ a $u_1 u_2 \ldots u_n = \mathbf{v} + \mathbf{w}$. To je $(n + n', k + k')$-kód s minimální vzdáleností

(16.7.1) $\min\{2d, d'\}$.

Skutečně, jestliže má nenulové slovo $\mathbf{v} \mid (\mathbf{v} + \mathbf{w})$ minimální váhu d_0, pak buď $\mathbf{w} \neq \mathbf{o}$ a odtud jistě plyne $\mathbf{v} = \mathbf{o}$, takže

$$d_0 = \|\mathbf{o} \mid \mathbf{w}\| = \|\mathbf{w}\| = d',$$

anebo $\mathbf{w} = \mathbf{o}$ a pak

$$d_0 = \|\mathbf{v} \mid \mathbf{v}\| = 2d.$$

16.7.1. Příklady. a) Je-li K opakovací kód délky 4 a K' je „koktavý" kód (7.2.2) délky 4, pak $K \mid K'$ je tento kód:

00000000 00001100 00000011 00001111
11111111 11110011 11111100 11110000.

b) Důležitý příklad (jak uvidíme v teorii Reedových-Mullerových kódů) je $K \oplus K'$, kde K je kód celkové kontroly parity a K' je opakovací kód, oba délky 4. Je to kód všech slov $\mathbf{v} \mid \mathbf{v}$ a $\mathbf{v} \mid \bar{\mathbf{v}}$, kde $\mathbf{v}$ je slovo sudé parity a $\bar{\mathbf{v}}$ je „opačné" slovo $\bar{\mathbf{v}} = \mathbf{v} + \mathbf{1}$:

$\mathbf{v} \mid \mathbf{v}$: 00000000 11001100 10101010 10011001 01100110
01010101 00110011 11111111,

$\mathbf{v} \mid \bar{\mathbf{v}}$: 00001111 11000011 10100101 10010110 01101001
01011010 00111100 11110000.

Vzniká (8, 4)-kód minimální vzdálenosti 4. A to je rozšířený Hammingův kód: zúžením kódu vznikne totiž kód délky 7 a minimální vzdálenosti 3 a ten je perfektní pro jednoduché chyby, takže je Hammingovým kódem (14.13).

IV. REEDOVY-MULLEROVY KÓDY— —KÓDY SE SNADNÝM DEKÓDOVÁNÍM

Reedovy-Mullerovy (čti rídovy-malerovy) kódy jsou nejstarší známou třídou kódů opravujících volitelný počet chyb. Byly objeveny v roce 1954 a jejich význam spočívá ve velmi jednoduché dekódovací metodě, která se také snadno implementuje. Ve srovnání s BCH kódy, kterým věnujeme kap. VII., mají slabší parametry. Přesto jsou prakticky významné. Například kód $R(1,5)$ použil kosmický koráb Mariner 9 při vysílání fotografií z Marsu.

Definice Reedových-Mullerových kódů je založena na boolovských funkcích. Ty probereme nejdříve a potom ukážeme základní vlastnosti Reedových-Mullerových kódů: počet informačních znaků, minimální vzdálenost a postupné vytváření od nejjednodušších k složitějším. Poslední článek věnujeme dekódování.

17. Boolovské funkce

Boolovské funkce jsou funkce, nabývající jen hodnot 0 a 1 při proměnných, které jsou také jen 0 nebo 1. Podrobněji, *boolovská funkce m* proměnných je předpis f, který každé m-tici $x_1, x_2, \ldots, x_m$ nul a jedniček přiřazuje hodnotu $f(x_1, x_2, \ldots, x_m) = 0$ nebo 1; f je tedy zobrazení z množiny Z_2^m do množiny Z_2. Tato zobrazení zapisujeme buď pomocí pravdivostní tabulky jako slova délky 2^m, nebo jako boolovské polynomy.

17.1. Pravdivostní tabulka. Pravdivostní tabulka je tabulka, ve které vypíšeme všechny možné kombinace hodnot $x_1, x_2, \ldots, x_m$ a u každé kombinace uvedeme hodnotu funkce f. Zápis kombinací proměnných provádíme systematicky tak, že sloupce tvoří čísla $0, 1, \ldots, 2^m - 1$ v binárním zápisu (odshora). Příklad boolovské funkce f dvou proměnných:

x_1	0 1 0 1
x_2	0 0 1 1
f	1 1 0 1.

Příklad boolovské funkce g tří proměnných:

x_1	0 1 0 1 0 1 0 1
x_2	0 0 1 1 0 0 1 1
x_3	0 0 0 0 1 1 1 1
g	1 1 0 1 1 1 0 1.

Tak jako v těchto dvou příkladech, budeme libovolnou boolovskou funkci $f(x_1, x_2, \ldots, x_m)$ chápat jako binární slovo délky 2^m, $f_0 f_1 \ldots f_{2^m-1}$, kde pro každé číslo $j = 0, 1, \ldots, 2^m - 1$ je

(17.1.1) $f_j = f(j_1, j_2, \ldots, j_m)$, pokud j má binární rozvoj $j_m j_{m-1} \ldots j_1$.

Například $f_0 = f(0, 0, \ldots, 0)$, $f_1 = f(1, 0, 0, \ldots, 0)$, $f_3 = f(1, 1, 0, \ldots, 0)$, atd. Všimněte si, že ve vzorci (17.1.1) bereme binární rozvoj čísla j pozpátku!

Mezi boolovské funkce patří ovšem i konstantní funkce $\mathbf{o} = 000\ldots0$ a $\mathbf{1} = 111\ldots1$. Dále každá z m proměnných x_i je sama boolovskou funkcí, $f(x_1, x_2, \ldots, x_m) = x_i$. Např. pro $m = 2$ je

$x_1 = 0101$ a $x_2 = 0011$.

17.1.1. Úloha. Ověřte, že proměnná x_i, zapsaná (jako boolovská funkce) ve tvaru binárního slova, je pravidelným střídáním skupin 2^{i-1} nul a 2^{i-1} jedniček, kde první znak je 0.

17.2. Logické operace. Na boolovských funkcích m proměnných zavádíme obvyklé logické operace:

název	označení	= 1, právě když
f a g	fg	$f = 1$ a $g = 1$;
f vel g	$f + g$	$f = 1$ nebo $g = 1$, ale ne oboje;
f nebo g	$f + g + fg$	$f = 1$ nebo $g = 1$ nebo $f = g = 1$;
negace f	$\bar{f}$	$f = 0$.

Například pro $f = 1101$ a $g = 0011$ platí:

$fg = 0001$,
$f + g = 1110$,
$f + g + fg = 1111$,
$\bar{f} = 0010$.

Mezi logickými operacemi platí jednoduché vztahy, např.

(17.2.1) $\bar{f} = f + 1$

a

(17.2.2) $ff = f$.

Boolovské funkce lze považovat za prvky lineárního prostoru Z_2^n, kde $n = 2^m$, a logický součet se shoduje se sčítáním v tomto prostoru:

(17.2.3) $f + g = h$, právě když $f_j + g_j = h_j$ $(j = 0, \ldots, 2^m - 1)$.

Navíc máme ještě operaci logického součinu, který splňuje

(17.2.4) $fg = h$, právě když $f_j g_j = h_j$ $(j = 0, \ldots, 2^m - 1)$.

To plyne ze vztahu (17.1.1):

$h_j = h(j_1, \ldots, j_m) = f(j_1, \ldots, j_m)\, g(j_1, \ldots, j_m) = f_j g_j$.

17.2.1. Pozorování. Jestliže má boolovská funkce tří proměnných $f(x_1, x_2, x_3)$ binární zápis $f_0 f_1 \dots f_7$, potom první polovina tohoto slova (tj. slovo $f_0 f_1 f_2 f_3$) je binárním zápisem boolovské funkce dvou proměnných $f(x_1, x_2, 0)$ a druhá polovina (tj. slovo $f_4 f_5 f_6 f_7$) je zápisem funkce $f(x_1, x_2, 1)$. Například pro funkci

$$\begin{aligned} f(x_1, x_2, x_3) &= x_2 + x_3 \\ &= 00110011 + 00001111 \\ &= 0011 \mid 1100 \end{aligned}$$

platí

$$f(x_1, x_2, 0) = x_2 = 0011$$

a

$$f(x_1, x_2, 1) = x_2 + 1 = 1100 .$$

(To plyne z pozorování pravdivostní tabulky v 17.1: první čtyři symboly odpovídají hodnotě $x_3 = 0$ a druhé čtyři hodnotě $x_3 = 1$.)

Obecně, boolovská funkce $f(x_1, x_2, \dots, x_m) = f_0 f_1 \dots f_{2^m-1}$ dává dvě boolovské funkce $m - 1$ proměnných:

$$f(x_1, x_2, \dots, x_{m-1}, 0) = f_0 f_1 \dots f_{2^{m-1}-1}$$

(první polovina slova) a

$$f(x_1, x_2, \dots, x_{m-1}, 1) = f_{2^{m-1}} f_{2^{m-1}+1} \dots f_{2^m-1}$$

(druhá polovina). Vyplývá to ze vztahu (17.1.1): pokud číslo $j = 0, 1, \dots, 2^m - 1$ leží v první polovině, tj. $j < \frac{1}{2}(2^m - 1)$, platí $j_m = 0$ a $f_j = f(j_1, j_2, \dots, j_{m-1}, 0)$. Pokud leží ve druhé polovině, platí $j_m = 1$ a $f_j = f(j_1, j_2, \dots, j_{m-1}, 1)$.

17.3. Boolovské polynomy. Jiný způsob, jak reprezentujeme boolovskou funkci, je pomocí součtů a součinů funkcí x_i a 1. Například reprezentujme funkci $f = 1101$. Abychom si to usnadnili, přejdeme k negaci $\bar{f} = 0010$. Ta je rovna 1, právě když $x_1 = 0$ a $x_2 = 1$, takže

$$\bar{f} = \bar{x}_1 x_2 = (1 + x_1) x_2 = x_2 + x_1 x_2 .$$

Odtud plyne

$$f = 1 + \bar{f} = 1 + x_2 + x_1 x_2 .$$

Tomuto tvaru říkáme *boolovský polynom* (dvou proměnných). V boolovském polynomu se ovšem nevyskytují mocnitelé vyšší než 1, protože podle (17.2.2) je $x_i^2 = x_i$ (a tedy $x_i^n = x_i$). Každý boolovský polynom m proměnných je tedy součtem členů

$$x_1^{i_1} x_2^{i_2} \dots x_m^{i_m} ,$$

kde mocnitelé $i_1, \dots, i_m$ jsou buď nula (u těch proměnných, které se ve výrazu nevyskytují), nebo jedna. Například pro $m = 3$ máme

$$\begin{aligned} x_1^1 x_2^0 x_3^0 &= x_1 , \\ x_1^0 x_2^1 x_3^1 &= x_2 x_3 , \\ x_1^0 x_2^0 x_3^0 &= 1 , \end{aligned}$$

atd.

Polynomy můžeme obecně zapsat ve tvaru

$$(*)\quad f(x_1, x_2, \ldots, x_m) = \sum_{i=0}^{2^m-1} q_i\, x_1^{i_1} x_2^{i_2} \ldots x_m^{i_m},$$

kde koeficient q_i je buď nula nebo 1 a číslo i má binární rozvoj $i_m \ldots i_2 i_1$ (který čtením pozpátku určuje mocnitele). Například polynom $f = x_2 + x_1 x_2$ je tvaru

$$f = q_0 + q_1 x_1 + q_2 x_1 + q_3 x_1 x_2,$$

kde $q_0 = q_1 = 0$ a $q_2 = q_3 = 1$.

Pro funkce tří proměnných má vztah (*) tvar

$$f(x_1, x_2, x_3) = q_0 + q_1 x_1 + q_2 x_2 + q_3 x_1 x_2 + q_4 x_3 + q_5 x_1 x_3 + q_6 x_2 x_3 + q_7 x_1 x_2 x_3$$

a např. funkce $f = 00001111$, neboli $f = x_3$, má jediný nenulový koeficient q_4.

Obecný postup jak z daného binárního slova délky 2^m vytvořit polynom m proměnných vyplývá z 17.2.1 a z následujícího tvrzení.

17.3.1. Tvrzení. Pro každou boolovskou funkci $m + 1$ proměnných

$$f(x_1, x_2, \ldots, x_m, x_{m+1})$$

platí

$$f = f(x_1, x_2, \ldots, x_m, 0) + [f(x_1, x_2, \ldots, x_m, 0) + f(x_1, x_2, \ldots, x_m, 1)]\, x_{m+1}.$$

Důkaz. Stačí ověřit, že obě strany se sobě rovnají jak pro $x_{m+1} = 0$, tak pro $x_{m+1} = 1$. To je snadné.

17.3.2. Příklad. Boolovskou funkci $f = 0111$ (dvou proměnných) převedeme na boolovský polynom: podle 17.2.1 a 17.3.1 je

$$f = 01 + (01 + 11)\, x_2 = 01 + 01 x_2.$$

Ovšem stejným postupem vidíme, že $01 = 0 + (0 + 1)\, x_1 = x_1$ a $10 = 1 + (1 + 0)\, x_1 = 1 + x_1$, takže

$$f = x_1 + (1 + x_1)\, x_2 = x_1 + x_2 + x_1 x_2.$$

Podobně pro $f = 11000111$ platí

$$f = 1100 + 1011 x_3 = x_1 + x_2 + x_3 + x_1 x_3 + x_1 x_2 x_3.$$

17.4. Množiny $M(i)$. Pro každé číslo $i = 0, 1, \ldots, 2^m - 1$ označujeme $M(i)$ množinu těch čísel $j \leqq i$, která ve svém binárním rozvoji má jedničky jen tam, kde je má číslo i. V binárním zápisu tedy

$$M(i_m \ldots i_2 i_1) = \{j_m \ldots j_2 j_1 \mid \text{z } j_k = 1 \text{ plyne } i_k = 1 \text{ pro } k = 1, 2, \ldots, m\}.$$

Například

$$M(0) = \{0\},\quad M(1) = \{0, 1\},\quad M(2) = \{0, 2\},\quad M(3) = \{0, 1, 2, 3\}.$$

Všimněte si, že platí

$$M(2^i) = \{0, 2^i\}.$$

17.5. Věta. *Každá boolovská funkce* $f = f_0 f_1 \dots f_{2^m-1}$ *je polynomem*

$$(17.5.1)\quad f = \sum_{i=0}^{2^m-1} q_i\, x_1^{i_1} x_2^{i_2} \dots x_m^{i_m}$$

(kde číslo i *má binární rozvoj* $i_m \dots i_1$*) s koeficienty*

$$(17.5.2)\quad q_i = \sum_{j \in M(i)} f_j .$$

Důkaz. Postupujeme matematickou indukcí. Pro $m = 1$ máme ověřit, že platí

$$f(x_1) = q_0 1 + q_1 x_1 = f_0 + (f_0 + f_1)\, x_1 .$$

To snadno ověříte dosazením $x_1 = 0$ a $x_1 = 1$.

V indukčním kroku je dána funkce $f(x_1, \dots, x_{m+1})$. Použijeme indukční předpoklad na funkci $f(x_1, \dots, x_m, 0)$, která má m proměnných:

$$f(x_1, \dots, x_m, 0) = \sum_{i=0}^{2^m-1} q_i'\, x_1^{i_1} \dots x_m^{i_m} ,$$

kde koeficienty q_i' jsou podle (17.5.2) a (17.1.1) dány vztahem

$$q_i' = \sum_{j \in M(i)} f(j_1, \dots, j_m, 0) = \sum_{j \in M(i)} f_j = q_i .$$

(Využíváním faktu, že pokud m-ciferný binární rozvoj čísla j je $j_m \dots j_1$, potom jeho $(m+1)$-ciferný rozvoj je $0 j_m \dots j_1$.) Analogicky pro funkci $f(x_1, \dots, x_m, 1)$ platí

$$f(x_1, \dots, x_m, 1) = \sum_{i=0}^{2^m-1} q_i''\, x_1^{i_1} \dots x_m^{i_m} ,$$

kde

$$q_i'' = \sum_{j \in M(i)} f(j_1, \dots, j_m, 1) = \sum_{j \in M(i)} f_{j+2^m}$$

(protože číslo $j + 2^m$ má binární rozvoj $1 j_m \dots j_1$).

Nyní využijeme 17.3.1: platí tedy

$$f(x_1, \dots, x_m, x_{m+1}) = \sum_{i=1}^{2^m-1} q_i\, x_1^{i_1} \dots x_m^{i_m} + \sum_{i=1}^{2^m-1} (q_i + q_i'')\, x_1^{i_1} \dots x_m^{i_m} x_{m+1} .$$

Všechna čísla $1, \dots, 2^{m+1} - 1$ se dělí na čísla

a) $i \leqq 2^m - 1$ s binárním rozvojem $0 i_m \dots i_1$. Koeficient polynomu f u členu $x_i^{i_1} \dots x_m^{i_m} = x_1^{i_1} \dots x_m^{i_m} x_{m+1}^{i_{m+1}}$ je q_i;

b) $i \geqq 2^m$, kde $i \leqq 2^m - 1$, s binárním rozvojem $1 i_m \dots i_1$. Koeficient polynomu f u členu $x_1^{i_1} \dots x_m^{i_m} x_{m+1}^1$ je $q_i + q_i''$. Prvky množiny $M(i + 2^m)$ jsou jednak čísla $j \leqq 2^m - 1$, která leží v $M(i)$, a jednak čísla $j + 2^m$ pro $j \in M(i)$, takže

$$q_{i+2^m} = \sum_{j \in M(i+2^m)} f_j = \sum_{j \in M(i)} f_j + \sum_{j \in M(i)} f_{j+2^m} = q_i + q_i'' .$$

Pro každé $k = 1, \dots, 2^{m+1} - 1$ je tedy koeficient u členu $x_1^{k_1} \dots x_{m+1}^{k_{m+1}}$ roven q_k.

17.6. Příklad. Pro boolovskou funkci $f = 1101$ máme

$$q_0 = f_0 = 1 ,$$
$$q_1 = f_0 + f_1 = 0 ,$$

$q_2 = f_0 + f_2 = 1$,
$q_3 = f_0 + f_1 + f_2 + f_3 = 1$,

takže

$$f(x_1, x_2) = 1 + x_2 + x_1x_2 .$$

To se shoduje se 17.3.

17.7. Úlohy

17.7.1. Napište boolovskou funkci 11011101 jako polynom.

17.7.2. Vyjádřete boolovský polynom $1 + x_1 + x_1x_2 + x_1x_2x_3$ jako binární slovo.

17.8. Poznámka. Boolovské polynomy o jediném sčítanci, tj. funkce

1,
$x_1, x_2, \ldots, x_m$,
x_ix_j pro $i, j = 1, \ldots, m$,
$\vdots$
$x_1x_2 \ldots x_m$

tvoří bázi lineárního prostoru Z_2^n pro $n = 2^m$. Podle věty 17.5 je totiž každé slovo v Z_2^n součtem některých z těchto funkcí. Lineární nezávislost plyne z faktu, že počet těchto funkcí je roven dimenzi celého prostoru: jejich počet je totiž

$$1 + m + \binom{m}{2} + \ldots + \binom{m}{m},$$

a to je číslo $n = 2^m$ (podle binomické věty, aplikované na $(1 + 1)^m$).

18. Vlastnosti Reedových-Mullerových kódů

18.1. Chceme definovat speciální binární kódy délky $n = 2^m$. Kódová slova tedy můžeme považovat za boolovské polynomy m proměnných (17.5). Definujeme *stupeň boolovského polynomu* f jako největší počet proměnných, které se vyskytují v některém sčítanci tohoto polynomu. Přesněji, stupeň polynomu

$$f = \sum q_i x_1^{i_1} x_2^{i_2} \ldots x_m^{i_m}$$

je maximální váha slova $i_1i_2 \ldots i_m$ takového, že $q_i = 1$. Například polynom $1 + x_1 + x_3$ má stupeň 1 a polynom $x_1x_2x_3$ má stupeň 3. Musíme ještě definovat stupeň nulového polynomu: to je -1.

18.2. Definice. *Reedovým-Mullerovým kódem stupně r a délky 2^m* se nazývá množina $R(r, m)$ všech boolovských polynomů m proměnných stupně nejvýše r.

18.3. Příklad: Všechny Reedovy-Mullerovy kódy délky 4

0	0 0 0 0	$R(-1, 2)$
1	1 1 1 1	$R(0, 2)$
x_1	0 1 0 1	
x_2	0 0 1 1	
$x_1 + x_2$	0 1 1 0	
$1 + x_1$	1 0 1 0	
$1 + x_2$	1 1 0 0	
$1 + x_1 + x_2$	1 0 0 1	$R(1, 2)$
x_1x_2	0 0 0 1	
$1 + x_1x_2$	1 1 1 0	
$x_1 + x_1x_2$	0 1 0 0	
$x_2 + x_1x_2$	0 0 1 0	
$x_1 + x_2 + x_1x_2$	0 1 1 1	
$1 + x_1 + x_1x_2$	1 0 1 1	
$1 + x_2 + x_1x_2$	1 1 0 1	
$1 + x_1 + x_2 + x_1x_2$	1 0 0 0	$R(2, 2)$

Vidíme, že $R(-1, 2) = \{\mathbf{0}\}$ a $R(0, 2)$ je opakovací kód. Dále $R(1, 2)$ je $(4, 3)$-kód, který je tedy kódem celkové kontroly parity. Nakonec $R(2, 2) = Z_2^4$.

18.4. Generující matice. Kód $R(r, m)$ je ovšem lineární, protože součet dvou polynomů stupně $\leqq r$ je také polynom stupně $\leqq r$. Řádky generující matice tvoří všechny součiny $1, x_i, x_{i_1}x_{i_2}, \ldots, x_{i_1}x_{i_2} \ldots x_{i_s}$ pro $s \leqq r$ (to plyne ze 17.8).

Například kód $R(2, 3)$ (boolovských polynomů stupně $\leqq 2$) má tuto generující matici:

$$\boldsymbol{G} = \begin{array}{|cccccccc|l} \multicolumn{8}{c}{} & \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & x_1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & x_2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & x_3 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & x_1x_2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & x_1x_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & x_2x_3 \\ \hline \end{array}.$$

Všimněte si, že první čtyři řádky této matice tvoří generující matici kódu $R(1, 3)$, a první řádek je generující matice opakovacího kódu $R(0, 3)$.

18.5. Kódování. Při kódování v $R(r, m)$ jsou informačními znaky hodnoty $q_i\,(=0, 1)$ pro všechna čísla $i = 0, 1, \ldots, 2^m - 1$, jejichž binární rozvoj má nejvýše m jedniček. Pak vyšleme polynom $(*)$ (viz 17.3), kde klademe $q_i = 0$ jakmile má i v binárním rozvoji váhu $\geqq m$.

Počet informačních znaků kódu $R(r, m)$ je číslo

$$k = \binom{m}{0} + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{r}.$$

To plyne z faktu, že počet všech součinů $x_{i_1}x_{i_2}\ldots x_{i_s}$ (pro $i_1, i_2, \ldots, i_s$ navzájem různá) je $\binom{m}{s}$ pro $s = 0, 1, \ldots, r$. Odtud plyne:

$R(m, m) = Z_2^n$ pro $n = 2^m = \sum_{i=0}^{m} \binom{m}{i}$;

$R(m-1, m)$ je $(2^m, 2^m - 1)$-kód, tj. kód celkové kontroly parity (4.6);

$R(m-2, m)$ je $(2^m, 2^m - m - 1)$-kód, tj. rozšířený Hammingův kód (14.14);

$R(1, m)$ je $(2^m, m+1)$-kód, duální k $R(m-2, m)$;

$R(0, m)$ je $(2^m, 1)$-kód, tj. opakovací kód;

$R(-1, m) = \{\mathbf{0}\}$.

Všimněte si, že kódy $R(m, m)$ a $R(-1, m)$ jsou navzájem duální (11.5), a také kódy $R(m-1, m)$ a $R(0, m)$, stejně jako $R(m-2, m)$ a $R(1, m)$. To dokážeme obecně.

18.6. Tvrzení. Duálním kódem k Reedovu-Mullerovu kódu $R(r, m)$ je Reedův-Mullerův kód $R(m-r-1, m)$.

Důkaz. I. Pro každé dvě boolovské funkce $f \in R(r, m)$ a $g \in R(m-r-1, m)$ dokážeme, že platí

$$f * g = 0.$$

Podle definice skalárního součinu a vztahu (17.2.4) platí

$$f * g = \sum_{i=0}^{n-1} f_i g_i = \sum_{i=0}^{n-1} (fg)_i.$$

Máme tedy ověřit, že slovo fg má sudou váhu (takže poslední součet je roven 0). Protože je f boolovský polynom stupně $\leqq r$ a g je boolovský polynom stupně $\leqq m-r-1$, je součin fg boolovský polynom stupně $\leqq m-1$. Ten je součtem některých funkcí $x_{i_1}x_{i_2}\ldots x_{i_s}$ $(s \leqq m-1)$ a stačí dokázat, že každá z těchto funkcí je slovem sudé váhy. Podle (17.1.1) je j-té písmeno tohoto slova rovno $j_{i_1}j_{i_2}\ldots j_{i_s}$, a to je 1, právě když číslo j má v binárním rozvoji jedničky na místech $i_1, i_2, \ldots, i_s$. Počet všech takových čísel j je ovšem 2^{m-s}: v binárním rozvoji máme s povinných jedniček a zbylých $m-s$ míst volíme libovolně. Protože $s \leqq m-1$, je 2^{m-s} sudé číslo.

II. Vidíme, že kód $R(r-m-1, m)$ je částí duálního kódu $R(r, m)^{\perp}$. Abychom dokázali rovnost, stačí ověřit, že oba tyto prostory mají stejnou dimenzi (tj. počet informačních znaků). Dimenze prostoru $R(r-m-1, m)$ je podle 18.5 rovna

$$\sum_{i=0}^{m-r-1} \binom{m}{i} = \sum_{i=0}^{m-r-1} \binom{m}{m-i} = \sum_{j=r+1}^{m} \binom{m}{j}.$$

Zde jsme využili známý vzorec $\binom{m}{i} = \binom{m}{m-i}$ a přešli ke sčítacímu indexu $j = m - i$. Dimenze prostoru Z_2^n je

$$n = 2^m = \sum_{i=0}^{m} \binom{m}{i} = \sum_{i=0}^{r} \binom{m}{i} + \sum_{j=r+1}^{m} \binom{m}{j},$$

a to je součet dimenzí prostorů $R(r, m)$ a $R(r - m - 1, m)$. Odtud plyne

$$\dim R(r, m)^{\perp} = n - \dim R(r, m) = \dim R(r - m - 1, m).$$

18.7. Příklad. Kód $R(1, 3)$ je samoduální (protože $3 - 1 - 1 = 1$). Má tuto generující matici:

$$\boldsymbol{G} = \begin{bmatrix} 1\,1\,1\,1\,1\,1\,1\,1 \\ 0\,1\,0\,1\,0\,1\,0\,1 \\ 0\,0\,1\,1\,0\,0\,1\,1 \\ 0\,0\,0\,0\,1\,1\,1\,1 \end{bmatrix} \begin{matrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{matrix}.$$

Počet informačních znaků je $\binom{3}{0} + \binom{3}{1} = 4$. Jde o binární $(8, 4)$-kód, který je rozšířeným Hammingovým kódem (14.14).

18.8. Úloha. Napište generující matici kódů $R(0, 3)$, $R(2,3)$ a $R(3, 3)$. Určete duální kódy.

18.9. Věta o vytváření. *Reedovy-Mullerovy kódy můžeme postupně vytvářet modifikovanými součty* (16.7):

$$R(r, m + 1) = R(r, m) \oplus R(r - 1, m) \quad (m > r > 0).$$

Důkaz. Použijeme tvrzení 17.3.1. Pro každé slovo $\boldsymbol{f}$ v kódu $R(r, m + 1)$ platí: polynom

$$g(x_1, \ldots, x_m) = f(x_1, \ldots, x_m, 0)$$

je stupně $\leqq r$, a tedy je kódovým slovem v $R(r, m)$, a polynom

$$h(x_1, \ldots, x_m) = f(x_1, \ldots, x_m, 0) + f(x_1, \ldots, x_m, 1)$$

je stupně $\leqq r - 1$ (ověřte matematickou indukcí!), a tedy je kódovým slovem v $R(r - 1, m)$. Podle 17.3.1 je

$$\boldsymbol{f} = \boldsymbol{g} + \boldsymbol{h}\, x_{m+1},$$

a to znamená, že pro slova $\boldsymbol{f}$ (délky 2^{m+1}) a $\boldsymbol{g}$ a $\boldsymbol{h}$ (délky 2^m) platí

$$\boldsymbol{f} = \boldsymbol{g} \,|\, (\boldsymbol{g} + \boldsymbol{h}).$$

Pro písmeno f_j v levé polovině slova $\boldsymbol{f}$ (tj. $j \leqq 2^m - 1$) máme totiž binární rozvoj $0 j_m \ldots j_2 j_1$, a tedy, podle (17.1.1),

$$f_j = f(j_1, j_2, \ldots, j_m, 0) = g_j.$$

Pro f_j v pravé polovině máme $j = 2^m + i$ a platí

$$\begin{aligned} f_j &= f(i_1, i_2, \ldots, i_m, 1) = \\ &= h(i_1, i_2, \ldots, i_m) + g(i_1, i_2, \ldots, i_m) = \\ &= h_i + g_i. \end{aligned}$$

Naopak, každé slovo $f = g \mid (g + h)$, kde g je stupně $\leqq r$ a h stupně $\leqq r - 1$, dává polynom

$$f = g + hx_{m+1}$$

stupně $\leqq r$.

18.10. Důsledek. Minimální vzdálenost kódu $R(r, m)$ je $d = 2^{m-r}$.

To je snadné cvičení matematické indukce za použití vztahu (16.7.1). Například kód $R(1, 5)$ je tedy (32, 6)-kód s minimální vzdáleností 16. Je to kód s nízkým informačním poměrem:

$$\frac{k}{n} < 0{,}2\,,$$

ale se schopností oprav 7-násobných chyb. Tento kód využil kosmický koráb Mariner 9 při vysílání fotografií z Marsu v roce 1972. Každý bod fotografie byl ohodnocen $2^6 = 64$ stupni světlosti a těchto 6 informačních znaků bylo zakódováno do slova délky 32, vyslaného na Zem.

18.11. Zúžený Reedův-Mullerův kód. Označme $R(r, m)^*$ kód, který vznikne z kódu $R(r, m)$ vynecháním prvního písmena každého slova. To je zúžený Reedův-Mullerův kód (16.2). Například generující matice kódu $R(1, 3)^*$ je podle 18.7 tato:

$$G^* = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Vynecháme-li z této matice řádek 1111111, vznikne matice, jejíž sloupce jsou navzájem různá slova délky 3 s počtem $2^3 - 1$, tedy vznikne kontrolní matice Hammingova kódu. Vidíme, že kód $R(1, 3)^*$ je zvětšením (16.3) kódu, duálního k Hammingovu (7, 4)-kódu.

18.12. Úloha. Duální kód Hammingova kódu délky $2^m - 1$ se nazývá *simplexový kód* délky $2^m - 1$. Ověřte, že zvětšením tohoto kódu vznikne kód $R(1, m)^*$. Návod: viz 14.13.

18.13. Závěr. Reedův-Mullerův kód $R(r, m)$ (kde $-1 \leqq r \leqq m$ a $m = 1, 2, 3, \ldots$) je binární kód délky 2^m. Počet jeho informačních znaků je $k = \sum_{i=0}^{r} \binom{m}{i}$. Jeho minimální vzdálenost je $d = 2^{m-r}$, takže kód objevuje všechny chyby násobnosti $< 2^{m-r}$ a opravuje všechny chyby násobnosti $< 2^{m-r-1}$. Kódování jsme popsali v 18.5 a dekódování probereme v následujícím článku.

Zúžením vznikne kód $R(r, m)^*$ délky $2^m - 1$ se stejným počtem informačních znaků a s minimální vzdáleností $2^{m-r} - 1$. Tento kód má lepší informační poměr než $R(r, m)$ a je cyklický (viz další část); v dodatku A2 uvádíme parametry všech zúžených Reedových-Mullerových kódů délek $\leqq 127$.

19. Dekódování Reedových-Mullerových kódů

Reedovy-Mullerovy kódy jsou důležité zejména pro svou jednoduchou a snadno implementovanou metodu dekódování. Je založena na většinové logice: pro hledanou hodnotu najdeme soustavu rovnic a rozhodneme se buď pro 0, nebo pro 1 tak, aby většina těchto rovnic platila.

Uvedeme nejprve dekódování kódů $R(0, m)$ a $R(1, m)$. Jsou to dva případy důležité samy o sobě a ilustrují obecný postup. Čtenář ovšem může přejít přímo k dekódování kódů $R(r, m)$.

19.1. Opakovací kódy $R(0, m)$. Dekódování provádíme „hlasováním". Přijali jsme slovo $\boldsymbol{w} = w_0 w_1 \ldots w_{n-1}$, $n = 2^m$, a hledáme první složku v_0 vysланého vektoru $\boldsymbol{v}$ tak, že položíme

$$\begin{aligned} v_0 &= w_0 \,, \\ v_0 &= w_1 \,, \\ &\vdots \\ v_0 &= w_{n-1} \,. \end{aligned}$$

Jestliže platí většina těchto rovnic pro $v_0 = 0$, rozhodneme, že skutečně $v_0 = 0$; podobně je $v_0 = 1$, jestliže tak rozhodla většina rovnic. V nerozhodném případě buď necháme v_0 nedefinováno, anebo (což je obvyklejší, ale nemění počet opravených chyb) položíme $v_0 = w_0$. Nakonec položíme $\boldsymbol{v} = v_0 v_0 \ldots v_0$.

Pokud nastalo méně než $n/2$ chyb, je většina z uvedených rovnic platná a vysланé slovo je skutečně $\boldsymbol{v}$. To jsme chtěli docílit: minimální vzdálenost kódu je n a my opravíme $n/2 - 1$ chyb.

19.2. Kódy prvního řádu $R(1, m)$. Ty mají minimální vzdálenost $d = 2^{m-1}$. Pro každý hledaný koeficient vysланého slova $\boldsymbol{v}$ najdeme 2^{m-1} rovnic a rozhodneme se pro tu hodnotu, pro kterou hlasuje většina.

Každé kódové slovo $\boldsymbol{v}$ je polynomem nejvýše prvního stupně, tj.

$$\boldsymbol{v} = q_0 1 + q_1 x_1 + q_2 x_2 + q_4 x_3 + q_8 x_4 + \ldots + q_{2^{m-1}} x_m \,.$$

To znamená, že všechny koeficienty q_i, kde i není 0 nebo mocnina dvojky, jsou nulové. Budeme určovat postupně koeficienty $q_1, q_2, \ldots, q_{2^m-1}$ (pomocí složek přijatého slova $\boldsymbol{w}$). Koeficient q_0 si necháme nakonec.

Pro koeficient q_1 chceme sestavit 2^{m-1} rovnic. Protože $M(1) = \{0, 1\}$, je jedna rovnice podle (17.5.2)

$$q_1 = w_0 + w_1 \,.$$

(Podrobněji: platí rovnice $q_1 = v_0 + v_1$, a tedy, pokud nenastala chyba na místě v_0 nebo v_1, i rovnice $q_1 = w_0 + w_1$.) Další rovnici najdeme tak, že si všimneme, že platí

$$q_3 = 0 \,,$$

protože náš polynom je prvního stupně, a neobsahuje tedy člen $x_1 x_2$. Podle (17.5.2) to znamená, že

$$0 = w_0 + w_1 + w_2 + w_3 = q_1 + w_2 + w_3 \,,$$

a další rovnice je tedy

$$q_1 = w_2 + w_3 .$$

Podobně

$$q_5 = 0 = w_0 + w_1 + w_4 + w_5 = q_1 + w_4 + w_5 ,$$

a tedy

$$q_1 = w_4 + w_5 .$$

Zjišťujeme, že pro každé sudé číslo $s = 0, \ldots, 2^m - 1$ platí

$$q_1 = w_s + w_{s+1} .$$

Počet všech sudých čísel $s \leqq 2^m - 1$ je 2^{m-1}, takže jsme našli 2^{m-1} rovnic. Protože pravé strany dvou různých rovnic neobsahují společné sčítance, žádná chyba nezkazí víc než jednu rovnici. O hodnotě koeficientu q_1 rozhodneme hlasováním a v nerozhodném případě (který není důležitý) se řídíme první rovnicí. Pokud nastalo méně než $(1/2)\, 2^{m-1}$ chyb, určili jsme koeficient q_1 správně.

Podobně postupujeme u všech koeficientů q_i, kde $i = 2^t$ $(t = 0, 1, \ldots, m - 1)$, a tedy $M_i = \{0, i\}$. Jedna rovnice je podle (17.5.2)

$$q_i = w_0 + w_i .$$

Další rovnici získáme tak, že si uvědomíme, že koeficient u členu $x_t x_u$ je nulový pro každé $u \neq t$. To znamená, že pro číslo $s = 2^u$ platí

$$q_{i+s} = 0 .$$

(Binární rozvoj čísla $i + s = 2^t + 2^u$ má jen dvě jedničky: na místech t a u.) Platí tedy

$$0 = w_0 + w_i + w_s + w_{i+s} ,$$

a odtud

$$q_i = w_s + w_{i+s} .$$

Stejnou úvahu můžeme provést pro každé číslo $s \neq 0$ takové, že nemá jedničku na místě t v binárním rozvoji: protože koeficient q_{i+s} patří ke členu stupně alespoň 2, je $q_{i+s} = 0$, a tedy $q_i = w_s + w_{i+s}$ (poslední rovnici přesně odvodíme v 19.3.1). Jak zapsat taková čísla s? Všimněte si, že číslo $2^m - 1$ má binární rozvoj $11\ldots11$, a tedy číslo $2^m - 1 - i$ má všude jedničky kromě místa t. Uvažujeme tedy všechna čísla s z množiny $M(2^m - i - 1) = M(n - i - 1)$. Dostáváme rovnice

$$(19.2.1) \qquad q_i = w_s + w_{i+s} \quad (\text{pro } s \in M(n - i - 1)) .$$

O hodnotě koeficientu q_i rozhodne většina rovnic, v nerozhodném případě první rovnice.

Když jsme určili všechny koeficienty $q_1, q_2, \ldots, q_{2^m-1}$, odečteme příslušné slovo od slova $\mathbf{w}$:

$$\mathbf{w}' = \mathbf{w} - (q_1 x_1 + q_2 x_2 + \ldots + q_{2^m-1} x_m) .$$

Tím získáme polynom $\mathbf{w}'$ stupně 0 (pokud nedošlo k chybě), tedy slovo opakovacího kódu. To zase dekódujeme hlasováním, abychom určili q_0:

$$q_0 = w'_0 = w'_1 = \ldots = w'_{n-1} .$$

Na závěr položíme

$$\boldsymbol{v} = q_0\,1 + q_1x_1 + \ldots + q_{2^{m-1}}x_m\,.$$

19.2.1. Příklad. Používáme kód $R(1, 3)$ a chceme dekódovat slovo $\boldsymbol{w} = 00010101$. Platí

$$\boldsymbol{v} = q_0\,1 + q_1x_1 + q_2x_2 + q_4x_3\,.$$

K určení koeficientu q_1 máme rovnice

$$\begin{aligned} q_1 &= w_0 + w_1 \quad (s = 0)\\ &= w_2 + w_3 \quad (s = 2)\\ &= w_4 + w_5 \quad (s = 4)\\ &= w_6 + w_7 \quad (s = 6)\,. \end{aligned}$$

V našem případě jsou všechny hodnoty kromě prvé rovny 1, a tedy

$$q_1 = 1\,.$$

Podobně pro q_2 máme $M(8 - 2 - 1) = M(5) = \{0, 1, 4, 5\}$, a tedy

$$\begin{aligned} q_2 &= w_0 + w_2 \quad (s = 0)\\ &= w_1 + w_3 \quad (s = 1)\\ &= w_4 + w_6 \quad (s = 4)\\ &= w_5 + w_7 \quad (s = 5)\,. \end{aligned}$$

V našem případě jsou všechny hodnoty kromě druhé rovny 0, a tedy

$$q_2 = 0\,.$$

Dále pro q_4 je $M(3) = \{0, 1, 2, 3\}$ a platí

$$\begin{aligned} q_4 &= w_0 + w_4 \quad (s = 0)\\ &= w_1 + w_5 \quad (s = 1)\\ &= w_2 + w_6 \quad (s = 2)\\ &= w_3 + w_7 \quad (s = 3) \end{aligned}$$

a většina určuje

$$q_4 = 0\,.$$

Zbývá určit koeficient q_0. Od přijatého slova odečteme q_1x_1 (protože $q_2 = q_4 = 0$). Platí $x_1 = 01010101$ (viz 17.1.1), a tedy

$$\boldsymbol{w}' = \boldsymbol{w} - q_1x_1 = 00010101 - 01010101 = 01000000\,.$$

Odtud hlasováním dostáváme

$$q_0 = 0\,.$$

Výsledný vektor je

$$\boldsymbol{v} = q_1x_1 = 01010101\,.$$

Jestliže nedošlo k více než jedné chybě, víme, že při přijetí slova $\boldsymbol{w} = 00010101$ bylo vysláno slovo $\boldsymbol{v} = 01010101$.

19.3. Obecné kódy $R(r, m)$. Každé kódové slovo $\boldsymbol{v}$ je boolovským polynomem

$$\boldsymbol{v} = \sum_{i=0}^{n-1} q_i\, x_m^{i_m} \ldots x_2^{i_2}x_1^{i_1}\,.$$

Protože je $\mathbf{v}$ polynomem stupně $\leqq r$, platí ovšem

$$q_i = 0\,, \quad \text{kdykoli} \quad \|i\| > r\,,$$

kde Hammingovou vahou $\|i\|$ čísla $i = 0, \ldots, 2^m - 1$ rozumíme Hammingovu váhu jeho binárního rozvoje $i_m \ldots i_2 i_1$. Dekódování budeme provádět tak, že nejdříve určíme ty koeficienty q_i, pro které $\|i\| = r$. Potom tyto členy můžeme od polynomu $\mathbf{v}$ odečíst a vytvořit polynom

$$\mathbf{v}' = \mathbf{v} - \sum_{\substack{i=0 \\ \|i\|=r}}^{n-1} q_i\, x_m^{i_m} \ldots x_2^{i_2} x_1^{i_1}\,.$$

Protože je $\mathbf{v}'$ polynomem stupně $\leqq r - 1$, a tedy kódovým slovem kódu $R(r-1, m)$, aplikujeme dále stejný postup (se změnou $r \mapsto r - 1$ a $\mathbf{v} \mapsto \mathbf{v}'$). Postup dekódování bude tedy objasněn, jakmile uvedeme, jak určit koeficienty s vahou indexu rovnou r.

Kód $R(r, m)$ má minimální váhu 2^{m-r}. Naším cílem je sestavit 2^{m-r} rovnic pro každý koeficient q_i, kde $\|i\| = r$, tak, aby jedna chyba nezkazila víc než jednu rovnici. Potom můžeme určit q_i z těchto rovnic „hlasováním": jestliže došlo k méně než $(1/2)\, 2^{m-r}$ chybám, většina rovnic zůstává v platnosti a určuje správnou hodnotu q_i. Nejprve uvedeme přesné znění rovnic a potom podrobně popíšeme dekódovací algoritmus, který z nich vychází.

19.3.1. Tvrzení. Je-li $\mathbf{v}$ kódové slovo kódu $R(r, m)$ a $i = 0, \ldots, 2^m - 1$ je číslo Hammingovy váhy r, potom pro koeficient q_i polynomu $\mathbf{v}$ platí tyto rovnice:

$$q_i = \sum_{j \in M(i)} v_{j+s} \quad \text{pro všechna} \quad s \in M(n - i - 1)\,. \tag{19.3.1}$$

Důkaz. Postupujeme matematickou indukcí podle Hammingovy váhy čísla s. Pokud $\|s\| = 0$, tj. $s = 0$, jde o rovnici (17.5.2). V indukčním kroku vezmeme nenulové číslo $s \in M(n - i - 1)$ Hammingovy váhy $h + 1$. Podle (17.5.2) platí

$$q_{i+s} = \sum_{j \in M(i+s)} v_j\,.$$

Číslo $n - 1 = 2^m - 1$ má binární rozvoj $11 \ldots 11$, takže číslo $n - i - 1$ má binární rozvoj, který je „negací" čísla i: nuly jsou tam, kde má číslo i jedničky, a naopak. Pro každé číslo $s = 0, 1, \ldots, 2^m - 1$ tedy podmínka

$$s \in M(n - i - 1)$$

znamená, že v binárních rozvojích nemají čísla i a s žádné společné jedničky. Pokud $s \neq 0$, je tedy $\|i + s\| > r$. Odtud plyne

$$q_{i+s} = 0\,,$$

protože polynom $\mathbf{v}$ je stupně $\leqq r$. Dále každé číslo $j \in M(i + s)$ je tvaru $j = j' + s'$, kde

$$j' \in M(i) \quad \text{a} \quad s' \in M(s)\,.$$

Číslo s' získáme z čísla j tak, že všechny jedničky v binárním rozvoji, které se shodují s jedničkami čísla i, změníme v nuly; podobně j'. Platí tedy

$$q_{i+s} = 0 = \sum_{s' \in M(s)} \sum_{j' \in M(i)} v_{j'+s'}\,.$$

Místo j' můžeme ovšem psát v tomto vzorci j. Každé číslo $s' \in M(s)$ kromě samotného s má ovšem váhu $\leqq h$ a podle indukčního předpokladu tedy platí $q_i = \sum_{j \in M(i)} v_{j+s'}$ pro $s' \neq s$, takže

$$0 = \sum_{j \in M(i)} v_{j+s} + q_i + q_i + \ldots + q_i ,$$

kde počet sčítanců q_i je roven počtu čísel $s' \in M(s)$, $s' \neq s$. Množina $M(s)$, včetně s, má 2^{h+1} prvků: víme, že číslo s má $h + 1$ jedniček v binárním rozvoji a na těchto $h + 1$ místech můžeme volit 0 nebo 1 (a na zbylých místech jsou ovšem nuly). Počet sčítanců q_i je tedy $2^{h+1} - 1$ (liché číslo), a protože sčítáme v Z_2, plyne odtud

$$0 = \sum_{j \in M(i)} v_{j+s} + q_i .$$

To jsme měli dokázat.

19.3.2. Poznámka. Počet rovnic (19.3.1) je $d = 2^{m-r}$ a žádná chyba neovlivní více než jednu rovnici.

Skutečně, váha čísla i je r a číslo $n - i - 1$ (které má v binárním rozvoji nuly tam, kde má číslo i jedničky, a naopak) má tedy váhu $m - r$. To znamená, že počet čísel s v $M(2^m - i - 1)$ je 2^{m-r}: číslo s vznikne tak, že na $m - r$ místech, kde jsou jedničky čísla $n - i - 1$, volíme 0 nebo 1, a na ostatních místech jsou nuly.

Dále, pro dvě různá čísla s a s' v $M(n - i - 1)$ jsou sčítance rovnic (19.3.1) navzájem různé, tj. platí

$$j + s \neq j' + s' \quad \text{pro všechna } j, j' \in M(i) .$$

Jestliže totiž v binárním rozvoji čísla $j + s$ anulujeme všechna místa, na kterých má číslo i jedničku, vznikne číslo s. Kdyby platilo $j + s = j' + s'$, odvodili bychom tedy $s = s'$.

Na základě těchto rovnic můžeme snadno navrhnout dekódovací algoritmus.

19.3.3. *První krok dékódování*: určení koeficientů s indexem váhy r.

Používáme kód $R(r, m)$, takže $d = 2^{m-r}$ (viz 18.10) a přijali jsme slovo $\boldsymbol{w} = w_0 w_1 \ldots w_{n-1}$ $(n = 2^m)$. Naším cílem je určit kódové slovo ve tvaru

$$\boldsymbol{v} = \sum_{\substack{i=0 \\ \|i\| \leqq r}}^{n-1} q_i x_m^{i_m} \ldots x_2^{i_2} x_1^{i_1} .$$

V prvém kroku určíme ty koeficienty q_i, pro které $\|i\| = r$. Jestliže většina z d výrazů

$$\sum_{j \in M(i)} w_{j+s} \quad \text{pro} \quad s \in M(n - i - 1)$$

je rovna 1, položíme $q_i = 1$; je-li většina rovna 0, položíme $q_i = 0$. V nerozhodném případě položíme $q_i = \sum_{j \in M(i)} w_j$ a ohlásíme, že počet chyb je alespoň $d/2$.

19.3.4. *Další kroky dekódování*: určení koeficientů s indexy váhy

$r - 1, r - 2, \ldots, 0$.

K určení koeficientů q_i, kde $\|i\| = r - 1$, použijeme koeficienty q_j, které jsme

určili, a položíme

$$\boldsymbol{w}' = \boldsymbol{w} - \sum_{\|j\|=r} q_j x_m^{j_m} \dots x_2^{j_2} x_1^{j_1} .$$

(Polynom, který odečítáme, ovšem vyjádříme ve tvaru slova.) Nyní aplikujeme předchozí postup na slovo $\boldsymbol{w}'$. Jestliže většina z $2^{m-(r-1)} = 2d$ výrazů

$$\sum_{j \in M(i)} w'_{j+s} \quad \text{pro} \quad s \in M(n-i-1)$$

je rovna 1, položíme $q_i = 1$; je-li většina rovna 0, položíme $q_i = 0$. V nerozhodném případě se řídíme rovnicí pro $s = 0$ a ohlásíme, že počet chyb je alespoň $d/2$.

Tak určíme postupně všechny koeficienty q_i a tím vektor $\boldsymbol{v}$.

19.3.5. Tvrzení. Jestliže došlo při přijetí slova $\boldsymbol{w}$ k chybě násobnosti $< d/2$, tj. jestliže existuje kódové slovo $\boldsymbol{v} \in R(r, m)$ takové, že $\|\boldsymbol{w} - \boldsymbol{v}\| < 1/2\, 2^{m-r}$, vede předchozí postup ke správnému určení kódového slova $\boldsymbol{v}$.

Poznámka. Z důkazu tvrzení bude zřejmé, že také hlášení o počtu chyb $\geqq d/2$ jsou správná.

Důkaz. Označme $\hat{q}_i$ nalezené koeficienty a q_i skutečné koeficienty polynomu $\boldsymbol{v}$. Pokud $\|i\| = r$, platí d rovnic (19.3.1), a protože jedna chyba neovlivní více než jednu rovnici (viz 19.3.2), platí více než $d/2$ (tj. většina) z rovnic

$$q_i = \sum_{j \in M(i)} w_{j+s} \quad \text{pro} \quad s \in M(n-i-1) .$$

Z těchto rovnic jsme hlasováním určili $\hat{q}_i$, takže $q_i = \hat{q}_i$.

Označme $\boldsymbol{v}' = \boldsymbol{v} - \sum q_i x_1^{i_1} \dots x_m^{i_m}$, kde sčítáme přes čísla i váhy r. Potom slova $\boldsymbol{v}'$ a $\boldsymbol{w}'$ mají stejný vzájemný vztah jako slova $\boldsymbol{v}$ a $\boldsymbol{w}$, pokud snížíme r na $r-1$. Můžeme tedy použít stejnou úvahu k důkazu platnosti $q_i = \hat{q}_i$ pro váhu i rovnou $r-1$ (a $r-2, \dots, 0$). Zde máme k dispozici dokonce $2d$ rovnic (a více), takže počet chyb menší než $d/2$ ovlivní jen menšinu z nich.

19.3.6. Příklad. Při použití kódu $R(2, 3)$ dekódujeme slovo $\boldsymbol{w} = 11111010$. Každé kódové slovo je tvaru

$$\boldsymbol{v} = q_0 1 + (q_1 x_1 + q_2 x_2 + q_4 x_3) + (q_3 x_1 x_2 + q_5 x_1 x_3 + q_6 x_2 x_3) .$$

První krok: určení koeficientů q_3, q_5 a q_6.

Pro q_3 máme $M(3) = \{0, 1, 2, 3\}$ a $M(8-3-1) = \{0, 4\}$, takže

$$\begin{aligned} q_3 &= w_0 + w_1 + w_2 + w_3 \quad (s = 0) \\ &= w_4 + w_5 + w_6 + w_7 \quad (s = 4) . \end{aligned}$$

Obě hodnoty jsou 0, a tedy $q_3 = 0$. Podobně určíme $q_5 = 1$ a $q_6 = 0$. Podle 17.1.1 platí $x_1 = 01010101$ a $x_3 = 00001111$, takže položíme

$$\boldsymbol{w}' = \boldsymbol{w} - q_5 x_1 x_3 = 11111010 - 00000101 = 11111111 .$$

Druhý krok: určení koeficientů q_1, q_2 a q_4.

Pro q_1 máme $M(1) = \{0, 1\}$ a $M(8 - 1 - 1) = \{0, 2, 4, 6\}$, takže

$$\begin{aligned} q_1 &= w_0 + w_1 \quad (s = 0) \\ &= w_2 + w_3 \quad (s = 2) \\ &= w_4 + w_5 \quad (s = 4) \\ &= w_6 + w_7 \quad (s = 6) . \end{aligned}$$

Všechny čtyři výrazy jsou 0, a tedy $q_1 = 0$. Podobně určíme $q_2 = 0$ a $q_4 = 0$. Položíme

$$\boldsymbol{w}'' = \boldsymbol{w}' - \mathbf{o} = 11111111 ,$$

a tedy

$$q_0 = 0 .$$

Závěr:

$$\boldsymbol{v} = x_1x_3 + 1 = 11111010 ,$$

V tomto případě $\boldsymbol{w} = \boldsymbol{v}$. (To však mnoho neznamená, protože kód $R(2, 3)$ má minimální vzdálenost jen 1.)

19.4. Úloha. Při použití kódu $R(2, 4)$ dekódujte slovo 1111111011111111.

V. CYKLICKÉ KÓDY

Síla algebraického popisu se v teorii kódování plně projevuje u třídy cyklických kódů. Jejich definice je (vzhledem k dalekosáhlým důsledkům) překvapivě jednoduchá: lineární kód je *cyklický*, jestliže s každým slovem $v_0v_1 \ldots v_{n-1}$ obsahuje i jeho cyklický posun $v_{n-1}v_0v_1 \ldots v_{n-2}$. Indexy zde píšeme od 0 (místo od 1) z důvodů, které budou za chvíli zřejmé.

Kódová slova můžeme zapisovat jako polynomy, protože každý polynom $v_0 + v_1x + \ldots + v_{n-1}x^{n-1}$ je jen zápisem slova $v_0v_1 \ldots v_{n-1}$. Například místo slova 2010100 (v ternárním kódu) píšeme polynom $2 + x^2 + x^4$. Cyklickým posuvem vznikne slovo 0201010, tedy polynom

$$2x + x^3 + x^5 = x(2 + x^2 + x^4).$$

To je x-násobek původního polynomu. Dalším cyklickým posunem vznikne slovo 0020101, tedy polynom

$$2x^2 + x^4 + x^6 = x^2(2 + x^2 + x^4),$$

atd. Ukazuje se, že cyklické kódy mají tu vlastnost, že pro každé kódové slovo, zapsané jako polynom $v(x)$, jsou i všechny násobky $p(x)\,v(x)$ kódová slova. A odtud plyne, jak dokážeme, že dokonce existuje jediné kódové slovo, tzv. generující polynom, které vytváří celý kód: kódová slova jsou právě všechny násobky generujícího polynomu.

To, že od slov (řekněme délky 7) přecházíme k polynomům (stupně $\leqq 6$) znamená, že používáme přepis

$$v_0v_1v_2v_3v_4v_5v_6 \leftrightarrow v_0 + v_1x + v_2x^2 + v_3x^3 + v_4x^4 + v_5x^5 + v_6x^6 .$$

Proto indexy od 0 do 6. Tento přepis nemění operaci sčítání: v lineárním prostoru T^7 sčítáme stejně, jak jsme zvyklí sčítat polynomy stupně $\leqq 6$. Ale navíc zde máme operaci násobení polynomů, takže algebraická struktura se velmi obohatila: místo pouhého násobení vektoru skalárem zde můžeme násobit dva vektory (ve smyslu obvyklého součinu dvou polynomů). Je tu ovšem háček. Co když součin dvou polynomů má stupeň vyšší než 6? V předchozím příkladu jsme cyklicky posunuli vektor 2010100 dvakrát a dostali jsme vektor 0020101, tj. polynom $x^2(2 + x^2 + x^4)$. Další posun dává vektor

$$1002010 \leftrightarrow 1 + 2x^3 + x^5 ,$$

a ten už bohužel není součinem $x^3(2 + x^2 + x^4)$ prostě proto, že tento součin obsahuje člen x^7. Ovšem při cyklickém posunu se koeficient u x^7 „octne" na nultém místě vektoru: místo polynomu $x^3(2 + x^2 + x^4) = 2x^3 + x^5 + x^7$ máme polynom

$1 + 2x^3 + x^5$. Jak je to možné? Prostě tak, že platí

$$x^7 = 1\,.$$

Je to podobný postup, jako při vytváření tělesa Z_7 (viz 9.8): na prvky $0, 1, \ldots, 6$ se můžeme dívat jako na obyčejná čísla (a tak je sčítáme i násobíme), až na to, že pokládáme

$$7 = 0\,,$$

a odtud odvodíme všechny další rovnosti ($8 = 1$, $9 = 2$, atd.).

Cyklické kódy délky n tedy budeme popisovat jako polynomy stupně $< n$, které sčítáme, jak je u polynomů obvyklé, a také obvyklým způsobem násobíme, až na to, že pokládáme $x^n = 1$ (a tedy $x^{n+1} = x$, $x^{n+2} = x^2$, atd.). Tento algebraický aparát popíšeme v čl. 20. Pomocí něj zavedeme generující a kontrolní polynom cyklického kódu a ukážeme, jak se cyklické kódy vytvářejí.

20. Okruhy polynomů

V tomto článku nejprve připomeneme základní fakta o polynomech a potom definujeme faktorové okruhy $T/q(x)$, kde T je těleso a $q(x)$ je polynom nad ním. Pokud volíme $q(x) = x^n - 1$, vznikne okruh polynomů, ve kterém platí $x^n = 1$, tedy okruh, ve kterém budeme pracovat s cyklickými kódy délky n. Význam faktorových okruhů se také ukáže v kap. VI, věnované konstrukci konečných těles.

20.1. Polynomy. Slova $a_0 a_1 \ldots a_n$ nad tělesem T můžeme formálně zapsat ve tvaru

$$a_0 + a_1 x + \ldots + a_n x^n\,,$$

který se nazývá *polynom proměnné* x nad tělesem T. To znamená, že a_i, tzv. *koeficient u i-té mocniny*, je prvek tělesa T. Při tomto zápisu vynecháváme ty členy $a_i x^i$, pro které $a_i = 0$. Speciálně, *nulový polynom* 0 je zápisem nulového slova $000\ldots0$. Důsledkem vynechání nulových členů je to, že vlastně neznáme délku slova (kterou můžeme měnit doplňováním nul).

Stupeň polynomu $a(x) = a_0 + a_1 x + \ldots + a_n x^n$ je největší číslo $k = \operatorname{st} a(x)$ takové, že $a_k \neq 0$; stupeň nulového polynomu pokládáme roven -1.

20.2. Příklady

20.2.1. Polynomy nad tělesem $Z_2 = \{0, 1\}$ mají jen koeficienty 0 nebo 1. Například $a(x) = 1 + x + x^2 + x^3$ je polynom třetího stupně a $b(x) = 1 + x^4$ je polynom čtvrtého stupně. Všimněte si, že oba definují tytéž funkce na Z_2: pro $x = 0$ mají oba polynomy hodnotu 1, pro $x = 1$ hodnotu 0.

20.2.2. Polynomy nad tělesem Z_3 mají koeficienty 0, 1 nebo 2. Předchozí polynomy $a(x)$ a $b(x)$ definují nad Z_3 různé funkce: $a(1) = 1$ a $b(1) = 2$. Polynom $c(x) = 1 + 2x + x^2$ druhého stupně definuje tutéž funkci jako polynom $a(x)$. Polynom $d(x) = 2$ je stupně 0 a polynom $e(x) = 0$ je stupně -1.

20.3. Polynomy a funkce. Každý polynom $a(x)$ definuje funkci z tělesa T do sebe: pro každý prvek t tělesa vypočteme hodnotu $a(t) = a_0 + a_1 t + \ldots + a_n t^n$. Nesmíte však ztotožňovat polynomy a funkce! Například jsme viděli, že dva různé polynomy $1 + x + x^2 + x^3$ a $1 + x^4$ (tj. slova 11110 a 10001) definují nad tělesem Z_2 tutéž funkci. V tom je situace odlišná od případu reálných polynomů (T = těleso reálných čísel), kde každý polynom je určen jak svými koeficienty, tak svými funkčními hodnotami, a proto se polynomy považují za funkce.

Kořenem polynomu $a(x)$ nad tělesem T nazýváme každý prvek t takový, že $a(t) = 0$.

20.4. Sčítání a násobení. Sčítání polynomů provádíme podle mocnin. To znamená, že *součet polynomů* $a(x) = a_0 + a_1 x + \ldots + a_n x^n$ a $b(x) = b_0 + b_1 x + \ldots + b_n x^n$ je polynom $c(x) = a(x) + b(x)$ s koeficienty

$$c_i = a_i + b_i \quad \text{(v tělese } T\text{)}.$$

Všimněte si ještě jednou rozdílu mezi polynomy a funkcemi. Pro funkce $a(x)$ a $b(x)$ je jasné, co znamená součet $a(x) + b(x)$: při každém dosazení za x hodnoty sečteme. Zvolíme-li $a(x) = 1 + x + x^2 + x^3$ a $b(x) = 1 + x^4$ (nad Z_2), vznikne nulová funkce $a(x) + b(x)$, protože $a(0) + b(0) = 1 + 1 = 0$ a $a(1) + b(1) = 0 + 0 = 0$. Přesto je $a(x) + b(x) = x + x^2$ polynom stupně 2, a nikoli nulový polynom.

Násobení polynomů má také svůj obvyklý význam: součin $c(x) = a(x)\,b(x)$ má koeficient u i-té mocniny

$$c_i = a_0 b_i + a_1 b_{i-1} + \ldots + a_i b_0 \quad \text{(v tělese } T\text{)}.$$

Například nad tělesem Z_2 platí

$$(1 + x + x^2 + x^3) + (1 + x + x^4) = x^2 + x^3 + x^4$$

a

$$(1 + x + x^2 + x^3)(1 + x + x^4) = 1 + x^5 + x^6 + x^7.$$

Nad tělesem Z_3 platí

$$(1 + x + x^2 + x^3) + (1 + x + x^4) = 2 + 2x + x^2 + x^3 + x^4$$

a

$$(1 + x + x^2 + x^3)(1 + x + x^4) = 1 + 2x + 2x^2 + 2x^3 + 2x^4 + x^5 + x^6 + x^7.$$

20.5. Úlohy

20.5.1. Ověřte, že množina všech polynomů nad tělesem T tvoří okruh. Ten se označuje $T[x]$.

20.5.2. Ověřte, že pro libovolné dva nenulové polynomy $a(x)$ a $b(x)$ platí: $\operatorname{st} a(x)\,b(x) = \operatorname{st} a(x) + \operatorname{st} b(x)$. Odvoďte, že pokud $a(x)\,b(x) = 0$, potom buď $a(x) = 0$, nebo $b(x) = 0$. Návod: Buď $n = \operatorname{st} a(x)$ a $m = \operatorname{st} b(x)$, pak $c_{n+m} = a_0 b_{n+m} + \ldots + a_{n+m} b_0$, ale členy $a_i b_j$ jsou nulové pro $i > n$ ($a_i = 0$) nebo $j > m$ ($b_j = 0$), takže $c_{n+m} = a_n b_m$.

20.5.3. Ověřte, že každým nenulovým polynomem $a(x)$ lze krátit, tj. z rovnice $a(x)\,b(x) = a(x)\,c(x)$ plyne $b(x) = c(x)$. Návod: $a(x)\,[b(x) - c(x)]$ je nulový polynom.

20.6. Dělení. Dělením polynomu $a(x)$ polynomem $b(x)$ nad tělesem T rozumíme určení *podílu* $q(x)$ a *zbytku* $r(x)$. To jsou polynomy nad tělesem T definované touto podmínkou:

(20.6.1) $a(x) = q(x)\,b(x) + r(x)$ a $\operatorname{st} r(x) < \operatorname{st} b(x)$.

Pro každý polynom $a(x)$ a každý nenulový polynom $b(x)$ můžeme dělit takto. Pokud $\operatorname{st} a(x) < \operatorname{st} b(x)$, jsme hotovi:

$$q(x) = 0 \quad \text{a} \quad r(x) = a(x).$$

Pokud $\operatorname{st} a(x) \geqq \operatorname{st} b(x)$, najdeme nejvyšší nenulový koeficient a_n polynomu $a(x)$ a nejvyšší nenulový koeficient b_m polynomu $b(x)$. Jejich podíl, který označíme

$$c_{n-m} = a_n b_m^{-1},$$

je koeficientem mocniny x^{n-m} podílu $q(x)$. Položme

$$\tilde{a}(x) = a(x) - c_{n-m}x^{n-m}\,b(x).$$

Potom stupeň polynomu $\tilde{a}(x)$ je jistě nižší než stupeň polynomu $a(x)$. Jestliže nyní $\operatorname{st} \tilde{a}(x) < \operatorname{st} b(x)$, jsme hotovi:

$$q(x) = c_{n-m}x^{n-m} \quad \text{a} \quad r(x) = \tilde{a}(x).$$

V opačném případě postupujeme stejným způsobem: dělíme nejvyšší nenulový koeficient polynomu $\tilde{a}(x)$ prvkem b_m atd.

20.7. Příklad. Dělíme

$$(x^3 + x + 1) : (x + 1)$$

nad tělesem Z_2:

$$\begin{array}{l} (x^3 + x + 1) : (x + 1) = x^2 + x \\ \underline{-\,(x^3 + x^2)} \\ \quad x^2 + x + 1 \\ \underline{-\,(x^2 + x)} \\ \qquad\quad 1. \end{array}$$

Je tedy $q(x) = x^2 + x$ a $r(x) = 1$.

Provedeme podíl nad tělesem Z_3:

$$\begin{array}{l} (x^3 + x + 1) : (x + 1) = x^2 + 2x + 2 \\ \underline{-\,(x^3 + x^2)} \\ \quad 2x^2 + x + 1 \\ \underline{-\,(2x^2 + 2x)} \\ \qquad\quad 2x + 1 \\ \underline{\qquad -\,(2x + 2)} \\ \qquad\qquad\quad 2. \end{array}$$

Zde platí $q(x) = x^2 + 2x + 2$ a $r(x) = 2$.

20.8. Úloha. Tento podíl proveďte nad tělesem Z_5. Ověřte dosazením do (20.6.1).

20.9. Okruh polynomů modulo $q(x)$. Pro každý polynom $q(x)$ stupně $n \geqq 1$ nad tělesem T definujeme okruh

$$T/q(x)\,,$$

tzv. *okruh polynomů modulo* $q(x)$. Prvky okruhu $T/q(x)$ jsou všechny polynomy nad T stupně nejvýše $n - 1$. Jejich proměnnou označujeme (kvůli odlišení) jinak než x – např. z. Operací sčítání v okruhu $T/q(x)$ je obvyklé sčítání polynomů:

$$a(z) + b(z)$$

je polynom stupně $< n$, pokud je $a(z)$ i $b(z)$ polynom stupně $< n$. Násobení $\otimes$ definujeme v okruhu $T/q(x)$ tak, že obvyklý součin dělíme polynomem $q(z)$ a všímáme si jen zbytku:

$a(z) \otimes b(z)$ *je zbytek po dělení polynomu* $a(z)\,b(z)$ *polynomem* $q(z)$.

(Dělení je ovšem opakovaným odečítáním: od součinu $a(z)\,b(z)$ odečítáme násobky polynomu $q(z)$ tak dlouho, až vznikne polynom stupně $< n$, a to je $a(z) \otimes b(z)$). Násobení v $T/q(x)$ je tedy určeno vztahem

(20.9.1) $\quad q(z) = 0\,.$

20.10. Příklady

20.10.1. Okruh $Z_2/(x^2 + 1)$. Jeho prvky jsou polynomy stupně $\leqq 1$ a ty jsou čtyři: 0, 1, z a $z + 1$. Sčítání je obvyklé sčítání polynomů (nad tělesem Z_2):

$+$	0	1	z	$z+1$
0	0	1	z	$z+1$
1	1	0	$z+1$	z
z	z	$z+1$	0	1
$z+1$	$z+1$	z	1	0

.

Násobení je určeno odečítáním polynomu $z^2 + 1$ (tj. vztahem $z^2 + 1 = 0$):

$$z \otimes z = z^2 - (z^2 + 1) = 1\,,$$
$$z \otimes (z + 1) = (z^2 + z) - (z^2 + 1) = z + 1\,,$$
$$(z + 1) \otimes (z + 1) = (z^2 + 1) - (z^2 + 1) = 0\,.$$

Zde je celá tabulka:

$\otimes$	0	1	z	$z+1$
0	0	0	0	0
1	0	1	z	$z+1$
z	0	z	1	$z+1$
$z+1$	0	$z+1$	$z+1$	0

.

20.10.2. Okruh $Z_2/(x^2 + x + 1)$. Tento okruh má stejné prvky a stejné sčítání jako předchozí okruh, ale má jiné násobení:

$$z \otimes z = z^2 - (z^2 + z + 1) = z + 1$$
$$z \otimes (z + 1) = (z^2 + z) - (z^2 + z + 1) = 1$$
$$(z + 1) \otimes (z + 1) = (z^2 + 1) - (z^2 + z + 1) = z\,.$$

Zde je celá tabulka:

$\otimes$	0	1	z	$1 + z$
0	0	0	0	0
1	0	1	z	$1 + z$
z	0	z	$1 + z$	1
$1 + z$	0	$1 + z$	1	z

20.10.3. $R/(x^2 + 1)$, kde R je těleso reálných čísel. Prvky jsou reálné polynomy stupně $\leqq 1$, jejich proměnná se však tradičně označuje i (ne z):

$$a + bi \quad (a, b \in R)\,.$$

Sčítání je ovšem dáno předpisem

(20.10.1) $\quad (a + bi) + (a' + b'\mathrm{i}) = (a + a') + (b + b')\,\mathrm{i}\,.$

Násobení je určeno odečítáním polynomu $\mathrm{i}^2 + 1$, takže např.

(20.10.2) $\quad \mathrm{i} \otimes \mathrm{i} = \mathrm{i}^2 - (\mathrm{i}^2 + 1) = -1\,.$

V okruhu $R/(x^2 + 1)$ tedy platí $\mathrm{i}^2 = -1$. Tím je násobení určeno:

$$(a + bi) \otimes (a' + b'\mathrm{i}) = aa' + ab'\mathrm{i} + ba'\mathrm{i} + bb'(-1) =$$
$$= (aa' - bb') + (ab' + a'b)\,\mathrm{i}\,.$$

Závěr:

$R/(x^2 + 1)$ je těleso komplexních čísel.

Konečně „správná" definice! Komplexní čísla se obvykle zavádějí tak, že se prohlásí, že bychom rádi odmocňovali číslo -1, a tak položíme $\mathrm{i} = \sqrt{-1}$. To je trochu příliš magické na to, aby to byla matematická definice. Ve snaze o upřesnění se někdy komplexní čísla zavádějí zcela formálně: jako množina všech dvojic reálných čísel (a, b) nebo $a + bi$, na které zavedeme sčítání (20.10.1) a násobení (20.10.2). To je trochu příliš formální na to, aby se objasnilo, o co jde. Komplexní čísla však můžeme zavést jako rozšíření reálných čísel o symbol i, splňující $\mathrm{i}^2 + 1 = 0$, tedy jako okruh $R/(x^2 + 1)$.

20.11. Úloha. Napište tabulku sčítání a násobení v okruhu $Z_3/(x^2 + 1)$ a $Z_2/(x^3 + 1)$. Které z uvedených příkladů okruhů $T/q(x)$ jsou tělesa?

20.12. Poznámka. Jiný způsob zavedení okruhů $T/q(x)$ je pomocí faktorových okruhů, viz 9.8. Polynom $q(x)$ je prvkem okruhu $T[x]$ všech polynomů nad tělesem T (20.5.1). Pro každý polynom $a(x)$ je třída $[a(x)]$ množinou polynomů tvaru $a(x) + t(x)\,q(x)$. Tyto polynomy mají ovšem stejný zbytek po dělení polynomem $q(x)$ jako polynom $a(x)$. Označme tento zbytek $r(x)$, potom tedy platí

$$[a(x)] = [r(x)]\,.$$

Každou třídu polynomů modulo $q(x)$ lze tedy psát ve tvaru $[r(x)]$, kde $r(x)$ je polynom stupně $<n$ (protože jde o zbytek po dělení polynomem $q(x)$ stupně n). Nadto žádné dva polynomy $r(x) \neq r'(x)$ stupně $<n$ neleží ve stejné třídě. (Polynom $q(x)$ nemůže dělit rozdíl $r(x) - r'(x)$, protože ten je stupně $<n$.) Jestliže místo označení $[r(x)]$ použijeme označení $r(z)$, dostáváme okruh $T/q(x)$ jako faktorový okruh $T[x]$ modulo $q(x)$: sčítání v obou okruzích je obyčejné sčítání polynomů a při násobení můžeme přejít ke zbytku po dělení polynomem $q(x)$.

20.13. Okruh $T^{(n)}$. Na množině všech slov délky n v tělese T můžeme zavést strukturu sčítání a násobení tak, že slova

$$a_0 a_1 \ldots a_{n-1}$$

zapíšeme jako polynomy

$$a_0 + a_1 z + \ldots + a_{n-1} z^{n-1},$$

které jsou prvky okruhu polynomů

$$T/(x^n - 1).$$

To znamená, že sčítání dvou slov má stejný význam jako v lineárním prostoru T^n: polynomy se sčítají stejně jako slova, tvořená jejich koeficienty. Násobení je určeno vztahem

$$z^n = 1,$$

přesněji, součin slov $a_0 a_1 \ldots a_{n-1}$ a $b_0 b_1 \ldots b_{n-1}$ dostaneme jako zbytek po dělení

$$(a_0 + a_1 z + \ldots + a_{n-1} z^{n-1})(b_0 + b_1 z + \ldots + b_{n-1} z^{n-1}) : (z^n - 1).$$

Pokud první polynom je stupně 0, tedy $a_0 00 \ldots 0$, jde o násobení skalárem a_0, které známe z lineárního prostoru T^n. Tento okruh označujeme

$$T^{(n)} = T/(x^n - 1).$$

Je obohacením algebraické struktury lineárního prostoru T^n.

20.14. Úloha. Popište součin v okruhu $Z_2^{(2)}$ a $Z_3^{(2)}$.

21. Cyklické kódy a generující polynomy

Cyklické kódy v abecedě T jsou lineární kódy (v prostoru T^n), uzavřené na cyklický posun. Budeme s nimi pracovat jako s množinami polynomů (v okruhu $T^{(n)}$). Potom dokážeme, že existuje tzv. generující polynom, jehož násobky tvoří daný kód. Generující polynom tedy popisuje kód stejně dobře jako generující matice, a přitom je to popis stručnější a umožňující snadné kódování informačních znaků.

21.1. Cyklické kódy. Připomeňme, že lineární kód $K \subseteq T^n$ se nazývá *cyklický*, jestliže pro každé kódové slovo $v_0 v_1 \ldots v_{n-1}$ je také $v_{n-1} v_0 v_1 \ldots v_{n-2}$ kódovým slovem.

Kódová slova budeme zapisovat ve tvaru polynomů stupně $< n$, takže místo $v_0 v_1 \dots v_{n-1}$ budeme psát

$$v(z) = v_0 + v_1 z + \dots + v_{n-1} z^{n-1} .$$

Potom kód K můžeme považovat za podmnožinu okruhu $T^{(n)}$ (viz 20.13). Protože platí $z^n = 1$, cyklický posuv odpovídá násobení prvkem z:

$$z\, v(z) = v_0 z + v_1 z^2 + \dots + v_{n-2} z^{n-1} + v_{n-1} z^n = \\ = v_{n-1} + v_0 z + v_1 z^2 + \dots + v_{n-2} z^{n-1} .$$

To znamená, že pro cyklický kód K platí: je-li $v(z)$ kódové slovo, je také $z\, v(z)$ kódové slovo, a tedy i $z^2 v(z)$, $z^3 v(z)$, ... jsou kódová slova. Obecněji:

21.2. Pozorování. Každý cyklický kód K je *ideálem* okruhu $T^{(n)}$, což znamená, že pro libovolný polynom $v(z)$ v K jsou i všechny násobky $q(z)\, v(z)$, kde $q(z) \in T^{(n)}$, polynomy v kódu K.

Skutečně, součin

$$q(z)\, v(z) = q_0\, v(z) + q_1 z\, v(z) + \dots + q_{n-1} z^{n-1}\, v(z)$$

e lineární kombinací polynomů $v(z)$, $z\, v(z)$, ..., $z^{n-1} v(z)$, které jsou kódovými slovy. Protože je kód K lineární, je lineární kombinace kódových slov také kódovým slovem.

21.3. Příklad: Kód celkové kontroly parity délky 4. To je cyklický kód, který sestává z těchto polynomů:

$$0,\ 1 + z,\ 1 + z^2,\ 1 + z^3,\ z + z^2,\ z + z^3,\ z^2 + z^3,\\ 1 + z + z^2 + z^3 .$$

To jsou totiž všechny polynomy se sudým počtem sčítanců. Každý součin polynomu $q(z)$ s některým kódovým slovem má ovšem opět sudý počet sčítanců, a je tedy kódovým slovem.

Všimněte si, že nenulový kódový polynom nejnižšího stupně je polynom $1 + z$. Ostatní kódové polynomy jsou právě všechny násobky polynomu $1 + z$ v okruhu $Z_2^{(4)}$:

$$0 = 0 \,.\, (1 + z) \qquad z + z^2 = z(1 + z)$$
$$1 + z = 1 \,.\, (1 + z) \qquad z + z^3 = (z + z^2)(1 + z)$$
$$1 + z^2 = (1 + z)(1 + z) \qquad z^2 + z^3 = z^2(1 + z)$$
$$1 + z^3 = (1 + z + z^2)(1 + z) \qquad 1 + z + z^2 + z^3 = (1 + z^2)(1 + z) .$$

Ukážeme, že toto platí obecně.

Připomínáme, že triviální lineární kód v T^n má dimenzi 0 nebo n, ostatní lineární kódy jsou netriviální.

21.4. Věta. *Každý netriviální cyklický (n, k)-kód K obsahuje polynom $g(z)$ stupně $n - k$. Ten má tyto vlastnosti:*

(i) *Kód K sestává ze všech násobků polynomu $g(z)$ v $T^{(n)}$, tj.*
$K = \{q(z)\, g(z) \mid q(z) \in T^{(n)}\}$;

(ii) *Polynomy* $g(z), z\,g(z), \ldots, z^{k-1}\,g(z)$ *tvoří bázi kódu* K;
(iii) *Polynom* $g(x)$ *dělí polynom* $x^n - 1$ *beze zbytku.*

21.4.1. Poznámka. Polynom stupně $n - k$ je v kódu K v podstatě jediný: ostatní polynomy, kromě skalárních násobků $k\,g(z)$, $k \in T$, mají vyšší stupeň — to plyne z (ii). Všimněte si, že v podmínce (iii) se vyskytuje proměnná x, tedy podmínka se týká okruhu polynomů $T[x]$. Předchozí podmínky se týkají okruhu $T^{(n)}$ (ve kterém $z^n - 1 = 0$, takže podmínka (iii) zde nemá smysl).

Důkaz. Zvolme nenulový polynom $g(z)$ v K co nejnižšího stupně s. Pro každé kódové slovo $v(z)$ ověříme, že existuje polynom $q(x)$ stupně $< n - s$, pro který v okruhu $T[x]$ platí

(21.4.1) $\quad v(x) = q(x)\,g(x)$.

Jestliže totiž polynom $v(x)$ dělíme polynomem $g(x)$, dostaneme

$$v(x) = q(x)\,g(x) + r(x) \quad \text{a} \quad \operatorname{st} r(x) < s\,.$$

Tato rovnost platí i po dosazení proměnné z za x, takže

$$v(z) = q(z)\,g(z) + r(z)\,.$$

Protože $v(z)$ i $q(z)\,g(z)$ jsou kódová slova (viz 21.2), je i $r(z) = v(z) - q(z)\,g(z)$ kódové slovo. Protože $\operatorname{st} r(z) < s$, ale s byl nejnižší stupeň nenulového kódového polynomu, vidíme, že $r(z) = 0$. Pro polynom $r(x)$ v okruhu $T[x]$ platí: polynom $r(z)$ je nulový prvek okruhu $T^{(n)} = T/(x^n - 1)$, právě když je polynom $x^n - 1$ dělitelem polynomu $r(x)$. Platí ale $\operatorname{st} r(x) < s < n$, takže $r(x) = 0$. Tím jsme ověřili (21.4.1). Polynom $q(x)$ má stupeň nejvýše $n - s - 1$ (protože $\operatorname{st} q(x) + s = \operatorname{st} q(x) + \operatorname{st} g(x) = \operatorname{st} v(x) < n$).

Ověříme (i). Právě jsme dokázali, že kód K je podmnožinou množiny $\{q(z)\,g(z) \mid q(z) \in T^{(n)}\}$. Podle 21.2 je i nadmnožinou této množiny.

Ověříme (ii). Každé kódové slovo je tvaru $q(z)\,g(z)$, kde $\operatorname{st} q(z) \leqq n - s - 1$, takže je lineární kombinací kódových slov

$$g(z), z\,g(z), \ldots, z^{n-s-1}\,g(z)\,.$$

Dokážeme, že tato slova jsou lineárně nezávislá, tj. že v každé nulové lineární kombinaci

$$0 = q_0\,g(z) + q_1 z\,g(z) + \ldots + q_{n-s-1} z^{n-s-1}\,g(z) = q(z)\,g(z)$$

je polynom $q(z)$ nulový. Protože platí $q(z)\,g(z) = 0$ v okruhu $T^{(n)}$, je polynom $q(x)\,g(x)$ dělitelný polynomem $x^n - 1$. Platí však $\operatorname{st} q(x)\,g(x) \leqq \operatorname{st} q(z) + \operatorname{st} g(z) = n - s - 1 + s = n - 1$, takže je součin $q(x)\,g(x)$ nulový polynom v $T[x]$. Protože polynom $g(x)$ je nenulový, plyne odtud, že $q(x)$ je nulový polynom, viz 20.5.2.

Vidíme, že dimenze k kódu K je rovna počtu $n - s$ vektorů dané báze:

$$k = n - s \quad \text{a} \quad s = n - k\,.$$

Tím jsme ověřili i to, že zvolený polynom $g(z)$ je stupně $n - k$.

Ověříme (iii). Dělením polynomu $x^n - 1$ polynomem $g(x)$ dostaneme tvar $x^n - 1 = q(x)\,g(x) + r(x)$, kde st $r(x) < s$. Stejně jako nahoře je $r(z) = (z^n - 1) - q(z)\,g(z) = -q(z)\,g(z)$ kódové slovo, a proto je $r(x)$ nulový polynom.

21.5. Definice. *Generující polynom* cyklického (n, k)-kódu je polynom stupně $n - k$ v tomto kódu (tj. polynom $g(z)$ v předchozí větě).

21.6. Pozorování. Generující matice cyklického kódu vznikne cyklickými posuvy koeficientů $g_0 g_1 \ldots g_{n-k}$ cyklického polynomu (až do vyčerpání nul napravo):

$$\boldsymbol{G} = \begin{bmatrix} g_0 & g_1 & g_2 & g_3 & \ldots & g_{n-k} & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_{n-k-1} & g_{n-k} & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & \ldots & g_{n-k-2} & g_{n-k-1} & g_{n-k} & 0 & \ldots & 0 \\ \multicolumn{11}{c}{\dotfill} \\ 0 & 0 & \ldots & 0 & g_0 & & g_1 & g_2 & g_3 & \ldots & g_{n-k} \end{bmatrix}.$$

($k - 1$ nad prvními n sloupci; $k - 1$ pod prvními k sloupci)

To plyne z podmínky (ii) předchozí věty: první řádek matice $\boldsymbol{G}$ je zápisem polynomu $g(z)$, druhý řádek je zápisem polynomu $z\,g(z)$, atd., až poslední odpovídá polynomu $z^{k-1}\,g(z)$.

21.7. Příklady

21.7.1. Kód celkové kontroly parity (21.3) má generující polynom $1 + z$, a tedy generující matici

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

21.7.2. Kód s generující maticí

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

má tato kódová slova: 0000, 1111, 0101 a 1010. Vidíme, že je cyklický. Polynom stupně $4 - 2 = 2$ je 1010, tj. $1 + z^2$. Ten dává jinou generující matici

$$\boldsymbol{G}' = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

21.7.3. Hammingův (7, 4)-kód. Kód z příkl. 14.5.2 není cyklický. Například cyklickým posunem kódového slova 1101001 (prvního řádku matice $\boldsymbol{G}$) dostaneme slovo 1110100, které není kódové (protože jeho syndrom je 101). Protože však sloupce kontrolní matice Hammingova kódu můžeme psát v libovolném pořadí, ukážeme nyní, jak je uspořádat tak, aby vznikl cyklický kód. Tyto sloupce tvoří

všechna slova délky 3 kromě 000. Místo slov budeme psát polynomy stupně $\leqq 2$, řekněme v proměnné t. Chceme tedy najít pořadí, v jakém zapsat všechny nenulové polynomy $a + bt + ct^2$ jako sloupce

$$\begin{bmatrix} c \\ b \\ a \end{bmatrix}$$

matice $\boldsymbol{H}$ tak, aby vznikl cyklický kód. K tomu použijeme okruh $Z_2/q(x)$ (viz 20.9), kde $q(x)$ je polynom třetího stupně, takže prvky okruhu jsou právě naše polynomy stupně $\leqq 2$. Vhodná volba, jak se ukazuje, je polynom

$$q(x) = x^3 + x + 1 .$$

To znamená, že proměnná t splňuje

$$t^3 + t + 1 = 0 .$$

Proč je tato volba vhodná? Protože mocninami t^i vyjádříme nyní všechny naše polynomy. Platí totiž $t^3 = t + 1$, a tedy $t^4 = t^2 + t$ atd. Zde jsou všechny mocniny:

$$\begin{aligned} t^0 &= 1 , \\ t^1 &= t , \\ t^2 &= t^2 , \\ t^3 &= t + 1 , \\ t^4 &= t^2 + t , \\ t^5 &= t^2 + t + 1 , \\ t^6 &= t^2 + 1 . \end{aligned}$$

Kontrolní matici $\boldsymbol{H}$ nyní uspořádáme podle těchto mocnin:

$$\boldsymbol{H} = \begin{bmatrix} 1 & t & t^2 & t^3 & t^4 & t^5 & t^6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} .$$

Kód s touto kontrolní maticí sestává ze všech polynomů

$$v(z) = v_0 + v_1 z + \ldots + v_6 z^6 ,$$

pro které platí

$$\boldsymbol{H} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_6 \end{bmatrix} = \begin{bmatrix} 1 & t & t^2 & \ldots & t^6 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_6 \end{bmatrix} = v_0 + v_1 t + v_2 t^2 + \ldots + v_6 t^6 = 0 ,$$

stručněji:

$$v(t) = 0 \quad \text{v okruhu} \quad Z_2/(x^3 + x + 1) .$$

A tento kód je ovšem cyklický. Pokud je totiž $v(z)$ kódové slovo, je i $z\, v(z)$ kódové slovo, protože $t\, v(t) = t \,.\, 0 = 0$. Přitom matice $\boldsymbol{H}$ má jako sloupce všechna nenulová slova délky 3, takže jde o Hammingův (7, 4)-kód.

Generující polynom je (jediný) kódový polynom stupně $7 - 4 = 3$. Protože proměnná t splňuje $t^3 + t + 1 = 0$, vidíme, že

$$g(z) = z^3 + z + 1$$

je generující polynom. Generující matice je tedy

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

21.8. Úlohy

21.8.1. Najděte generující polynom opakovacího kódu.

21.8.2. Najděte generující polynom ternárního kódu s generující maticí

$$\boldsymbol{G} = \begin{bmatrix} 1 & 2 & 0 & 0 & 2 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 2 & 2 & 1 & 0 & 0 \end{bmatrix}.$$

21.9. Kódování informačních znaků. Generující matice

$$\boldsymbol{G} = \begin{bmatrix} g(z) \\ z\,g(z) \\ \vdots \\ z^{k-1}\,g(z) \end{bmatrix}$$

dává návod, jak kódovat informační znaky $u_0u_1 \ldots u_{k-1}$: platí

$$(u_0u_1 \ldots u_{k-1})\,\boldsymbol{G} = (u_0 + u_1z + \ldots + u_{k-1}z^{k-1})\,g(z)\,.$$

To znamená, že z informačních znaků vytvoříme polynom stupně $<k$ a tím násobíme polynom $g(z)$.

21.10. Příklady

21.10.1. Kód celkové kontroly parity délky 4 má generující polynom $1 + z$ (viz 21.3). Informační znaky $u_0u_1u_2$ kódujeme takto:

$$(u_0 + u_1z + u_2z^2)(1 + z) = u_0 + (u_0 + u_1)\,z + (u_1 + u_2)\,z^2 + u_2z^3\,.$$

Například slovo 110 dává polynom $1 + z^2$, tj. slovo 1010.

21.10.2. Generující polynom Hammingova (7, 4)-kódu je $z^3 + z + 1$ (21.7.3). Informační znaky 1001 zakódujeme takto:

$$(1 + z^3)(z^3 + z + 1) = 1 + z + z^4 + z^6\,,$$

takže vyšleme slovo 1100101.

21.11. Systematické kódování. Cyklický kód, jako každý lineární kód, je ekvivalentní se systematickým kódem (10.9). V případě cyklických kódů je to snadné: každé slovo čteme pozpátku. (Jinými slovy, polynomy zapisujeme od nejvyšší mocniny k nejnižší.) Pro tento kód máme systematické kódování, založené na dělení generu-

jícím polynomem. Z informačních znaků $u_0u_1 \ldots u_{k-1}$ vytvoříme polynom

$$u(z) = u_0z^{n-1} + u_1z^{n-2} + \ldots + u_{k-1}z^{n-k}$$

a ten dělíme polynomem $g(z)$:

$$u(z) = q(z)\,g(z) + r(z), \quad \text{kde} \quad \text{st}\, r(z) < n - k.$$

Odečtením zbytku $r(z)$ vznikne tedy kódové slovo

$$q(z)\,g(z) = u(z) - r(z).$$

Označme $-r(z) = r_kz^{n-k-1} + r_{k+1}z^{n-k-2} + \ldots + r_{n-1}z + r_n$, pak vyšleme kódové slovo

$$u_0u_1 \ldots u_{k-1}r_k \ldots r_n.$$

21.12. Příklad. Při systematickém kódování Hammingova (7, 4)-kódu zakódujeme informační znaky 1001 tak, že polynom $u(z) = z^6 + z^3$ dělíme generujícím polynomem $g(z) = z^3 + z + 1$:

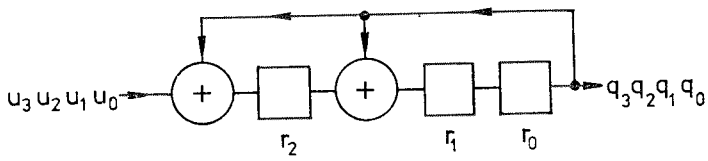
$$\begin{array}{l} (z^6 + z^3) : (z^3 + z + 1) = z^3 + z \\ \underline{-\,(z^6 + z^4 + z^3)} \\ \quad z^4 \\ \quad \underline{-\,(z^4 + z^2 + z)} \\ \qquad z^2 + z\,. \end{array}$$

Vyšleme kódové slovo

$$u(z) - r(z) = (z^6 + z^3) - (z^2 + z) = z^6 + z^3 + z^2 + z,$$

tj. 1001110 (při zápisu od nejvyšší mocniny k nejnižší).

21.13. Poznámka. Důležitým rysem binárních cyklických kódů je snadná realizace kódování informačních znaků. Násobení i dělení daným binárním polynomem se totiž snadno vytváří číslicovými obvody. Například při systematickém kódování Hammingova (7, 4)-kódu dělíme polynomem $z^3 + z + 1$. To můžeme realizovat číslicovým obvodem s dvěma binárními sčítačkami a třemi posuvnými registry:



Obr. 24

Vstupují koeficienty polynomu $u(z)$ a vystupují koeficienty podílu $q(z) = q_0z^3 + q_1z^2 + q_2z + q_3$. Po vystoupení posledního z nich v registrech zůstávají koeficienty zbytku $r(z) = r_0z^2 + r_1z + r_2$.

22. Kontrolní polynomy

Tak jako roli generující matice převzal generující polynom $g(x)$, také kontrolní matici cyklického kódu můžeme nahradit jediným polynomem. Je to tzv. *kontrolní polynom*

$$h(x) = (x^n - 1) : g(x) .$$

Tato definice má smysl, protože polynom $g(x)$ dělí polynom $x^n - 1$ beze zbytku (21.4). Ukážeme, že kontrolní matici můžeme sestavit podle tohoto polynomu. Ten je dále generujícím polynomem duálního kódu, čteného pozpátku. Stupněm kontrolního polynomu je počet informačních znaků.

22.1. Příklad. Kód celkové kontroly parity délky 4 má generující polynom $1 + z$ (21.3), a tedy kontrolní polynom je

$$h(x) = (x^4 - 1) : (x + 1) = x^3 + x^2 + x + 1 .$$

Všimněte si, že platí $(1 + z)\, h(z) = z^4 - 1 = 0$, takže pro každé slovo $v(z)$ sudé parity platí $v(z)\, h(z) = 0$, protože $v(z)$ je násobkem polynomu $1 + z$. Naopak, kdykoli platí $v(z)\, h(z) = 0$, má slovo $v(z)$ sudou paritu.

22.2. Tvrzení. Cyklický kód s kontrolním polynomem $h(x)$ sestává právě z těch polynomů $v(z)$ v okruhu $T^{(n)}$, pro které platí

$$v(z)\, h(z) = 0 \quad \text{v} \quad T^{(n)} .$$

Důkaz. Protože $g(x)\, h(x) = x^n - 1$, platí

$$g(z)\, h(z) = z^n - 1 = 0 \quad \text{v} \quad T^{(n)} .$$

Každé kódové slovo $v(z)$ je tvaru $v(z) = q(z)\, g(z)$, viz 21.4. Proto

$$v(z)\, h(z) = q(z)\, g(z)\, h(z) = 0 \quad \text{v} \quad T^{(n)} .$$

Naopak, jestliže polynom $v(z)$ v $T^{(n)}$ splňuje rovnost $v(z)\, h(z) = 0$, dokážeme, že je násobkem polynomu $g(z)$, a tedy kódovým slovem. Dělíme polynom $v(x)$ polynomem $g(x)$:

$$v(x) = q(x)\, g(x) + r(x) , \quad \text{st}\, r(x) < n - k .$$

Pak platí

$$0 = v(z)\, h(z) = q(z)\, g(z)\, h(z) + r(z)\, h(z) = r(z)\, h(z) .$$

To znamená, že součin $r(x)\, h(x)$ je dělitelný polynomem $x^n - 1$. Protože $\text{st}\, r(x)\, h(x) < \text{st}\, g(x)\, h(x) = n$, je polynom $r(x)\, h(x)$ nulový, a tedy je $r(x)$ nulový polynom. Odtud $v(z) = q(z)\, g(z)$.

22.3. Pozorování. Kontrolní matici cyklického (n, k)-kódu získáme cyklickými posuvy koeficientů polynomu $h(x) = h_0 + h_1 x + \ldots + h_k x^k$, čteného od nej-

vyšší mocniny:

$$H = \begin{bmatrix} \overbrace{0 \ 0 \ \ldots \ 0 \ 0}^{n-k+1} & h_k & h_{k-1} & \ldots & h_3 & h_2 & h_1 & h_0 \\ 0 \ 0 \ \ldots \ 0 \ h_k & h_{k-1} & h_{k-2} & \ldots & h_2 & h_1 & h_0 & 0 \\ 0 \ 0 \ \ldots \ h_k \ h_{k-1} & h_{k-2} & h_{k-3} & \ldots & h_1 & h_0 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ h_k \ h_{k-1} \ \ldots \ h_3 \ h_2 & h_1 & h_0 & 0 & & \ldots & & 0 \end{bmatrix}.$$

Skutečně, uvedená matice má $n - k$ řádků, které jsou lineárně nezávislé (protože $h_k \neq 0$). Stačí tedy ověřit, že každé kódové slovo, tj. každý polynom $v(z)$ takový, že $v(z)\,h(z) = 0$, splňuje $\boldsymbol{H}\boldsymbol{v}^{\mathrm{T}} = \boldsymbol{o}^{\mathrm{T}}$. První znak součinu $\boldsymbol{H}\boldsymbol{v}^{\mathrm{T}}$ je

$$v_{n-k-1}h_k + v_{n-k}h_{k-1} + \ldots + h_1 v_{n-2} + h_0 v_{n-1}\,.$$

To je nula, protože je to koeficient polynomu $v(z)\,h(z)$ u mocniny z^{n-1}. Podobně je druhý znak koeficientem polynomu $v(z)\,h(z)$ u mocniny z^{n-2}, atd.

22.4. Příklady

22.4.1. Kód celkové kontroly parity délky 4 má kontrolní polynom $h(x) = x^3 + x^2 + x + 1$, a tedy kontrolní matici

$$\boldsymbol{H} = [1 \ 1 \ 1 \ 1]\,.$$

22.4.2. Hammingův (7, 4)-kód má kontrolní polynom

$$(x^7 - 1) : (x^3 + x + 1) = x^4 + x^2 + x + 1\,.$$

Jeho koeficienty (čtené pozpátku) dávají tuto kontrolní matici:

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

To je skutečně kontrolní matice: vznikne z matice $\boldsymbol{H}$ v 21.7.3 přičtením prvního řádku ke třetímu.

22.5. Poznámka. Každý cyklický kód rozkládá polynom $x^n - 1$ v součin $g(x)\,h(x)$. Naopak, každý takový součin definuje cyklický kód (všech násobků polynomu $g(z)$). Například popíšeme všechny binární cyklické kódy délky 5. Platí

$$x^5 - 1 = (x + 1)(x^4 + x^3 + x^2 + x + 1)\,.$$

Polynom $x^4 + x^3 + x^2 + x + 1$ už dále nemůžeme rozložit, tj. není součinem $p(x)\,q(x)$ polynomů stupně <4. (To plyne z faktu, že 0 ani 1 není kořenem polynomu $x^4 + x^3 + x^2 + x + 1$, takže ani polynomy $p(x)$ a $q(x)$ nemohou mít kořen. Jediný polynom stupně 2 and Z_2, který nemá kořeny, je $x^2 + x + 1$ a ani ten nedělí polynom $x^4 + x^3 + x^2 + x + 1$.) Odtud plyne, že existují právě dva netriviální cyklické binární kódy délky 5:

opakovací kód: $g(x) = x + 1$ a $h(x) = x^4 + x^3 + x^2 + x + 1$,
kód celkové kontroly parity: $g(x) = x^4 + x^3 + x^2 + x + 1$ a
$h(x) = x + 1$.

Naproti tomu v tělese Z_5 platí

$x^5 - 1 = (x - 1)^5$,

takže zde máme čtyři cyklické kódy:

$g(x) = x - 1$ a $h(x) = (x - 1)^4$, $(5, 4)$-kód kontroly parity
$g(x) = (x - 1)^2$ a $h(x) = (x - 1)^3$, $(5, 3)$-kód
$g(x) = (x - 1)^3$ a $h(x) = (x - 1)^2$, $(5, 2)$-kód
$g(x) = (x - 1)^4$ a $h(x) = x - 1$, opakovací kód.

22.6. Úloha. Najděte všechny ternární cyklické kódy délky 5 a všechny binární cyklické kódy délky 6.

22.7. Duální kód. Duální kód (11.5) cyklického kódu je zřejmě také cyklický. Je-li $h(x)$ kontrolní polynom cyklického kódu K, pak je $h(z)$ generující polynom kódu $K^\perp$, čteného pozpátku (tj. od nejvyšší mocniny k nejnižší).

Skutečně, duální kód $K^\perp$ daného (n, k)-kódu K je cyklický $(n, n - k)$-kód. To znamená, že jeho generujícím polynomem je libovolný kódový polynom stupně $n - (n - k) = k$, a to je stupeň polynomu $h(z)$. Stačí tedy ověřit, že slovo $h_{n-1}\, h_{n-2} \ldots h_1 h_0$, vytvořené z koeficientů polynomu $h(z)$, leží v duálním kódu $K^\perp$. Skalární součin tohoto slova s libovolným kódovým slovem $v_0 v_1 \ldots v_{n-1}$ kódu K je roven $v_0 h_{n-1} + v_1 h_{n-2} + \ldots + v_{n-1} h_0$, a to je koeficient u mocniny z^{n-1} polynomu $v(z)\, h(z)$. Protože platí $v(z)\, h(z) = 0$, je tento koeficient nulový.

22.8. Příklad. Hammingův $(7, 4)$-kód má kontrolní polynom $h(x) = x^4 + x^2 + x + 1$. Čteme-li pozpátku, dostaneme generující matici duálního kódu

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Při „přímém" čtení to je právě matice $\boldsymbol{H}$ v 22.4.2.

22.9. Úloha. Určete duální kód k cyklickému kódu délky 5 nad Z_5 s generujícím polynomem $x^3 + 2x^2 + 3x + 2$.

VI. KONEČNÁ TĚLESA A POLYNOMY

Tato kapitola je čistě algebraická: ukážeme, jaká je struktura konečných těles a jak se pracuje s polynomy a jejich kořeny. Čtenář může tuto kapitolu vynechat, ale budeme se na ni odvolávat všude tam, kde jsou cyklické kódy určeny kořeny polynomů. Na tom je např. založena definice BCH kódů, nejdůležitější třídy cyklických kódů.

Vysvětlíme, jaké vlastnosti mají kořeny polynomů a jak se pomocí polynomů rozšiřuje dané těleso. Odvodíme, že pro každé prvočíslo p existují tělesa (tzv. Galoisova) $GF(p^n)$ o p^n prvcích ($n = 1, 2, 3, \ldots$), která rozšiřují těleso Z_p. Pro cyklické kódy má základní význam otázka, ve kterém tělese je možno rozložit polynom $x^n - 1$ na lineární součinitele – tuto otázku podrobně zodpovíme.

23. Kořeny polynomů a ireducibilita

Tak jako v případě reálných polynomů, má polynom $f(x)$ nad konečným tělesem kořen a, právě když je dělitelný kořenovým činitelem $x - a$. Důležitý je pojem ireducibilního (tj. nerozložitelného) polynomu, tedy polynomu, který není součinem žádných dvou polynomů nižšího stupně. Jde o obdobu pojmu prvočísla: každý polynom je součinem ireducibilních polynomů. Významnou vlastností ireducibilních polynomů je to, že jejich faktorové okruhy (20.9) jsou tělesa.

23.1. Kořen polynomu. Kořenem polynomu $f(x)$ nad tělesem T se nazývá takový prvek $a \in T$, pro který platí $f(a) = 0$.

Například polynom $x^2 + 1$ má v tělese Z_2 kořen 1. V tělese Z_3 žádný kořen nemá.

23.2. Tvrzení. Polynom $f(x)$ má kořen a, právě když je dělitelný polynomem $x - a$ (tzv. *kořenovým činitelem*).

Důkaz. Provedeme dělení polynomu $f(x)$ polynomem $x - a$ a dostaneme tvar

$$f(x) = q(x)(x - a) + r,$$

kde r je stupně $\leqq 0$ (tedy konstanta).

Je-li a kořen polynomu $f(x)$, dosadíme do rovnice a zjistíme, že $0 = q(a) \cdot 0 + r$, takže $r = 0$ a $f(x) = q(x)(x - a)$.

Naopak, je-li $r = 0$, platí $f(a) = q(a) \cdot 0 = 0$.

23.3. Důsledek. Polynom $f(x)$ s navzájem různými kořeny $a_1, a_2, \ldots, a_k$ je dělitelný polynomem $(x - a_1)(x - a_2) \ldots (x - a_k)$.

To plyne z předchozí věty matematickou indukcí: podíl $f(x)/(x - a_1) \ldots$ $\ldots (x - a_{k-1})$ má kořen a_k, a je tedy dělitelný polynomem $x - a_k$.

23.4. Důsledek. Polynom stupně $n \geqq 0$ má nejvýše n kořenů.

Skutečně, polynom s kořeny $a_1, \ldots, a_k$ je dělitelný polynomem $(x - a_1)(x - a_2) \ldots (x - a_k)$, a tedy je buď stupně $n \geqq k$, nebo $n = -1$.

23.5. Příklad. Rozložme polynom $x^3 + 1$ nad tělesem Z_2 v součin co nejjednodušších polynomů.

Protože je 1 kořen, je polynom $x^3 + 1$ dělitelný kořenovým činitelem $x + 1$:

$$x^3 + 1 = (x + 1)(x^2 + x + 1).$$

Polynom $x^2 + x + 1$ už dál rozložit nemůžeme: nemá kořeny, a tedy není součinem lineárních polynomů.

23.6. Úlohy

23.6.1. Rozložte polynom $x^3 + 1$ nad tělesy Z_3 a Z_5.

23.6.2. *Kořen* a polynomu $f(x)$ je *n-násobný*, jestliže je polynom $(x - a)^n$ dělitelem polynomu $f(x)$, ale polynom $(x - a)^{n+1}$ není jeho dělitelem. Určete všechny kořeny, včetně násobností, polynomu $x^5 + x$ nad tělesem Z_2 a nad tělesem Z_3.

23.6.3. *Formální derivace* polynomu $f(x) = \sum_{n=0}^{k} f_n x^n$ je polynom $f'(x) = \sum_{n=0}^{k} n f_n x^{n-1}$, kde $n = 1 + 1 + \ldots + 1$ s n sčítanci. Dokažte, že platí obvyklý vzorec pro derivaci součinu: $[f(x)\, g(x)]' = f'(x)\, g(x) + f(x)\, g'(x)$.

Ověřte, že kořen a polynomu $f(x)$ je jednoduchý, kdykoli $f'(a) \neq 0$. Návod: Pokud $f(x) = (x - a)^2\, g(x)$, je derivace polynomu $f(x)$ dělitelná polynomem $x - a$.

23.7. Definice. Polynom $f(x)$ je *ireducibilní* nad tělesem T, jestliže není součinem $f(x) = a(x)\, b(x)$, kde $a(x)$ a $b(x)$ jsou polynomy nad T nižšího stupně než $\mathrm{st}\, f(x)$.

Například každý lineární polynom je ireducibilní. Polynom $x^2 + 1$ je rozložitelný nad tělesem Z_2: platí $x^2 + 1 = (x + 1)(x + 1)$. Nad tělesem Z_3 je však ireducibilní. To plyne z faktu, že nad tímto tělesem nemá kořeny:

23.8. Tvrzení. Ireducibilní polynom stupně > 1 nemá žádné kořeny. Naopak, polynom stupně 2 nebo 3, který nemá kořeny, je ireducibilní.

Důkaz. Jestliže má ireducibilní polynom $f(x)$ kořen a, platí $f(x) = q(x)(x - a)$ (viz 23.2), kde $\mathrm{st}\, q(x) = \mathrm{st}\, f(x) - 1$. Z definice ireducibility plyne, že $\mathrm{st}\, f(x) = \mathrm{st}\,(x - a) = 1$.

Jestliže polynom $f(x)$ stupně 2 nebo 3 nemá kořeny, je ireducibilní. To plyne z faktu, že každý polynom stupně 1 má kořen, a pokud $f(x) = a(x)\, b(x)$, kde oba polynomy $a(x)$ i $b(x)$ mají stupeň menší než $\mathrm{st}\, f(x)$, potom jeden z těchto polynomů je stupně 1.

23.9. Příklady

23.9.1. Polynom $(x^2 + x + 1)^2$ není ireducibilní, protože je součinem dvou (stejných) polynomů stupně 2. Přitom nad tělesem Z_2 nemá tento polynom kořeny. Pro polynomy stupně $\geqq 4$ tedy nestačí zjišťovat kořeny k určení ireducibility.

23.9.2. Je polynom $x^4 + x^3 + 1$ ireducibilní nad Z_2? Snadno zjistíme, že nemá kořeny, takže není dělitelný lineárním polynomem. Jediný rozklad, který přichází v úvahu, je tedy $a(x)\, b(x)$, kde $a(x)$ i $b(x)$ jsou polynomy stupně 2. Navíc tyto polynomy nemohou mít nad Z_2 kořen. Takový polynom existuje jen jeden: $x^2 + x + 1$. Pro $a(x) = b(x) = x^2 + x + 1$ je však $a(x)\, b(x) = x^4 + x^2 + 1$. Dokázali jsme tedy, že polynom $x^4 + x^3 + 1$ je ireducibilní nad Z_2.

23.10. Úlohy

23.10.1. Je polynom $x^4 + x^3 + 1$ ireducibilní nad tělesem Z_3?

23.10.2. Ověřte, že polynom $x^2 + 1$ je ireducibilní nad tělesem R reálných čísel. Jak vypadají všechny ireducibilní polynomy stupně 2 nad R? A nad tělesem komplexních čísel?

23.10.3. Ověřte, že žádný reálný polynom stupně $\geqq 3$ není ireducibilní (nad R). Návod: s každým komplexním kořenem a je i komplexní sdružené číslo $\bar{a}$ kořenem a polynom $(x - a)(x - \bar{a})$ je reálný.

23.10.4. Dokažte, že každý polynom (nad libovolným tělesem) je součinem ireducibilních polynomů. Uvědomte si, co to znamená pro reálné polynomy. Návod: matematická indukce podle $\mathrm{st}\, f(x)$. Buď je polynom $f(x)$ ireducibilní, nebo $f(x) = a(x)\, b(x)$ a na polynomy $a(x)$ a $b(x)$ použijeme indukční krok.

23.10.5. *Normovaný* polynom je polynom, jehož koeficient u nejvyšší mocniny je 1. (Nad tělesem Z_2 je každý nenulový polynom normovaný, ale nad tělesem Z_3 již ne.) Ověřte, že každý nenulový polynom $f(x)$ je tvaru $f(x) = k\tilde{f}(x)$, kde $k \neq 0$ je konstanta a $\tilde{f}(x)$ je normovaný polynom. Ověřte, že každý normovaný polynom je součinem ireducibilních normovaných polynomů.

23.11. Věta. *Pro každý ireducibilní polynom $q(x)$ nad tělesem T je okruh polynomů $T/q(x)$ (20.9) tělesem.*

Důkaz. Prvky okruhu $T/q(x)$ jsou polynomy $a(z)$ stupně $k < \text{st } q(x)$. Naším úkolem je dokázat, že pokud $k \neq -1$ (tj. pokud $a(z)$ není 0), má prvek $a(z)$ inverzní prvek v $T/q(x)$. Postupujeme matematickou indukcí podle k.

Pokud $k = 0$, je $a(z) = a_0$ $(\neq 0)$ a inverzním prvkem je konstantní polynom a_0^{-1}.

Pro $k + 1$ předpokládáme, že všechny nenulové polynomy stupně $\leqq k$ mají inverzní prvky. Dělíme polynom $q(x)$ polynomem $a(x)$ a dostaneme podíl $s(x)$ a zbytek $r(x)$:

(23.11.1) $q(x) = s(x)\, a(x) + r(x)$, $\text{st } r(x) \leqq k$.

Protože polynom $q(x)$ je ireducibilní, nemůže platit $q(x) = s(x)\, a(x)$. (Víme, že stupeň polynomu $a(x)$ je menší než polynomu $q(x)$, ale větší než 0, takže platí $\text{st } s(x) < \text{st } q(x)$.) To znamená, že $r(x)$ je nenulový polynom. Jeho stupeň je menší než st $q(x)$, takže i $r(z)$ je nenulový prvek v okruhu $T/q(x)$. Podle indukčního předpokladu existuje polynom $p(z)$, inverzní k tomuto prvku:

$$p(z)\, r(z) = 1 \quad \text{v} \quad T/q(x).$$

V proměnné x to znamená, že zbytek po dělení polynomu $p(x)\, r(x)$ polynomem $q(x)$ je 1. Označme $t(x)$ podíl tohoto dělení, pak

$$p(x)\, r(x) = t(x)\, q(x) + 1.$$

Vynásobme rovnici (23.11.1) členem $p(x)$:

$$p(x)\, q(x) = p(x)\, s(x)\, a(x) + t(x)\, q(x) + 1.$$

Dosadíme proměnnou z, pro kterou platí $q(z) = 0$:

$$0 = p(z)\, s(z)\, a(z) + 1.$$

To znamená, že prvek $-p(z)\, s(z)$ v okruhu $T/q(x)$ je inverzní k prvku $a(z)$, protože $-p(z)\, s(z)\, a(z) = 1$.

Protože má každý nenulový prvek okruhu $T/q(x)$ inverzní prvek, je tento okruh tělesem.

23.12. Příklady

23.12.1. Polynom $x^2 + 1$ je ireducibilní nad tělesem R reálných čísel. Proto je okruh $R/(x^2 + 1)$ (komplexních čísel) tělesem.

23.12.2. Polynom $x^2 + x + 1$ je ireducibilní nad tělesem Z_2, protože nemá kořeny. Proto je okruh $Z_2/(x^2 + x + 1)$ tělesem. Skutečně, v tabulce násobení 20.10.2 v každém nenulovém řádku najdete jedničku, takže každý nenulový prvek má inverzní prvek.

23.12.3. Polynom $x^2 + 1$ je rozložitelný nad tělesem Z_2: platí $x^2 + 1 = (x + 1)^2$. Proto nepřekvapuje, že okruh $Z_2/(x^2 + 1)$ není tělesem. Skutečně,

v řádku prvku $z + 1$ v tabulce násobení 20.10.1 není jednička, takže tento prvek nemá inverzní prvek.

23.13. Poznámka. Těleso $T/q(x)$, kde $q(x)$ je ireducibilní polynom, se nazývá *algebraické rozšíření* tělesa T. Těleso reálných čísel má jen jediné algebraické rozšíření – těleso komplexních čísel. Naproti tomu konečná tělesa mají libovolně velká algebraická rozšíření, jak uvidíme později.

Obecněji se *rozšířením* tělesa T rozumí těleso T^*, které původní těleso obsahuje ($T \subseteq T^*$) a všechny algebraické operace tělesa T dávají stejné výsledky, i když je provádíme v tělese T^*. Například těleso $T/q(x)$ obsahuje prvky $t \in T$ (považované za konstantní polynomy) a součet $t + t'$ má pro $t, t' \in T$ stejný význam v tělesech T i $T/q(x)$. Totéž platí pro rozdíl, součin a podíl.

Každý polynom nad tělesem T je ovšem i polynomem nad rozšířeným tělesem T^*.

23.14. Úlohy. Pro libovolný ireducibilní polynom $q(x)$ stupně m nad tělesem T dokažte, že platí:

23.14.1. Prvek z tělesa $T/q(x)$ není kořenem žádného nenulového polynomu $f(x)$ nad T stupně $< m$. Návod: $f(z)$ je prvek tělesa $T/q(x)$.

23.14.2. Polynom $f(x)$ nad T má kořen z v tělese $T/q(x)$, právě když je násobkem polynomu $q(x)$. Návod: na zbytek po dělení $f(x) : q(x)$ použijte předchozí cvičení.

23.14.3. Jestliže polynom $q(x)$ dělí součin polynomů $f(x)\, g(x)$, potom dělí buď polynom $f(x)$, nebo polynom $g(x)$. Návod: v tělese $T/q(x)$ platí $f(z)\, g(z) = 0$, a tedy $f(z) = 0$ nebo $g(z) = 0$.

23.14.4. Buď $\tilde{q}(x)$ ireducibilní polynom, který není skalárním násobkem polynomu $q(x)$, tj. $\tilde{q}(x) \neq t\, q(x)$ pro $t \in T$. Každý polynom $f(x)$, dělitelný polynomy $q(x)$ a $\tilde{q}(x)$, je dělitelný i jejich součinem $q(x)\, \tilde{q}(x)$. Návod: v tělese $T/q(x)$ je $f(z) = 0$, přitom $f(x) = h(x)\, \tilde{q}(x)$, a protože $\tilde{q}(z) \neq 0$ (podle 23.14.2), je $h(z) = 0$. Podle 23.14.2 je $h(x) = k(x)\, q(x)$.

23.14.5. Zobecněte předchozí cvičení na libovolné ireducibilní polynomy $q_1(x), \ldots, q_k(x)$: pokud žádný není skalárním násobkem jiného, pak každý polynom, dělitelný těmito polynomy, je dělitelný i jejich součinem. Návod: matematická indukce a 23.14.3.

23.14.6. Dokažte větu o rozkladu: každý polynom $f(x)$ nad tělesem T je tvaru $f(x) = t\, q_1(x)\, q_2(x) \ldots q_k(x)$, kde $t \in T$ a $q_i(x)$ jsou normované ireducibilní polynomy, určené až na pořadí jednoznačně. Návod: existence plyne z 23.10.4 a 23.10.5, jednoznačnost z 23.14.5.

23.15. Definice. Těleso

$Z_p/q(x)$,

kde p je prvočíslo a $q(x)$ je ireducibilní polynom stupně n nad Z_p, se nazývá *Galoisovo* (čti galoázovo) a označuje se

$GF(p^n)$.

23.16. Příklady

23.16.1. $GF(4)$ je těleso $Z_2/(x^2 + x + 1)$. Víme totiž, že polynom $x^2 + x + 1$ (který nemá nad Z_2 kořeny) je ireducibilní. Tabulka sčítání a násobení tělesa $GF(4)$ je uvedena v 20.10.

23.16.2. $GF(16)$ je těleso $Z_2/(x^4 + x^3 + 1)$.

23.16.3. $GF(9)$ je těleso $Z_3/(x^2 + 1)$ a také těleso $Z_3/(x^2 + x + 2)$. Později uvidíme, že jde jen o dvě různé reprezentace stejného tělesa.

23.17. Úloha. Ověřte, že pokud polynom $q(x)$ není ireducibilní nad tělesem T, není $T/q(x)$ těleso. Návod: v každém tělese z $ab = 0$ plyne $a = 0$ nebo $b = 0$.

23.18. Poznámka. Každé těleso Z_p, kde p je prvočíslo, je Galoisovo těleso: $Z_p = Z_p/(x - 1)$. Další Galoisova tělesa jsou uvedena v dodatku A1.

Později ukážeme, že

a) pro každou mocninu prvočísla p^n existuje těleso $GF(p^n)$, tj. existuje ireducibilní polynom $q(x)$ stupně n nad tělesem Z_p (v ostrém kontrastu s tělesem reálných čísel, viz 23.10.3!),

b) na výběru polynomu $q(x)$ nezáleží,

c) jiná konečná tělesa než Galoisova tělesa neexistují.

24. Řád a primitivní prvky

Prvek $a \neq 0$ má v tělese *řád* r, jestliže je r nejmenší číslo $1, 2, 3, \ldots$ takové, že $a^r = 1$. (Zde $a^r = aa \ldots a$ je součin s r činiteli, definujeme také $a^0 = 1$.) Pokud takové číslo r neexistuje, je řád prvku a roven ∞. Například v tělese reálných čísel mají všechna čísla řád ∞ kromě 1 $(r = 1)$, -1 $(r = 2)$ a 0 (řád není definován).

V konečném tělese se řád ∞ nevyskytuje:

24.1. Tvrzení. Každý prvek $a \neq 0$ konečného tělesa má konečný řád r. Prvky $1, a, a^2, \ldots, a^{r-1}$ jsou navzájem různé a platí:

$a^n = 1$, právě když je n násobkem čísla r $(n = 1, 2, 3, \ldots)$.

Poznámka. Platí tedy $a^n = a^m$ $(n > m)$, právě když číslo r dělí rozdíl $n - m$.

Důkaz. Prvky a^n, kde n je libovolné přirozené číslo, nemohou být v konečném tělese navzájem různé. Zvolme nejmenší číslo $r = 1, 2, 3, \ldots$ takové, že $a^{n+r} = a^n$ (pro některé n). Tuto rovnici dělíme prvkem a^n a dostaneme $a^r = 1$. Je tedy r řádem prvku a a prvky $1, a, a^2, \ldots, a^{r-1}$ jsou navzájem různé.

Pro každý násobek $n = kr$ čísla r platí $a^n = (a^r)^k = 1^k = 1$. Naopak, pokud $a^n = 1$, ověříme, že číslo n je násobkem čísla r. Provedeme celočíselné dělení $n : r$ a dostaneme podíl q a zbytek r_0 tak, že platí

$$n = qr + r_0 \quad \text{a} \quad r_0 < r .$$

Máme dokázat, že $r_0 = 0$, a to plyne z faktu, že r_0 je menší než řád r a přitom $a^{r_0} = 1$:

$$1 = a^n = a^{qr} a^{r_0} = (a^r)^q a^{r_0} = 1^q a^{r_0} = a^{r_0} .$$

Nakonec, z rovnice $a^n = a^m$ plyne $a^{n-m} = 1$, takže číslo r dělí rozdíl $n - m$.

24.2. Příklad. V tělese Z_5 mají prvky tyto řády:

prvek	0	1	2	3	4
řád	–	1	4	4	2 .

24.3. Poznámka. Úkol rozložit polynom $x^n - 1$ (důležitý pro cyklické kódy) je snadný v každém tělese, které obsahuje prvek a řádu n. Platí totiž

$$x^n - 1 = (x - 1)(x - a)(x - a^2) \ldots (x - a^{n-1}) .$$

Skutečně, mocniny a^i jsou navzájem různé kořeny polynomu $x^n - 1$ (protože $(a^i)^n - 1 = (a^n)^i - 1 = 1 - 1 = 0$), a tedy polynom na pravé straně dělí polynom $x^n - 1$ (23.3). Protože oba polynomy jsou stupně n a u mocniny x^n mají koeficient 1, platí uvedená rovnost.

24.4. Definice. Prvek a tělesa T je *primitivní*, jestliže každý nenulový prvek v T je jeho mocninou (a^i pro $i = 1, 2, 3, \ldots$).

24.5. Příklady

24.5.1. Číslo 2 je primitivní prvek tělesa Z_3: platí $2 = 2^1$ a $1 = 2^2$. Řád čísla 2 je 2.

24.5.2. Číslo 2 je primitivní i v tělese Z_5: platí $2 = 2^1$, $3 = 2^3$, $4 = 2^2$ a $1 = 2^4$. Řád čísla 2 je 4.

24.5.3. Prvek z je primitivní v tělese $GF(4) = Z_2/(x^2 + x + 1)$. Platí $z^2 = 1 + z$ a $z^3 = 1$. Řád prvku z je 3.

24.6. Úlohy

24.6.1. Ověřte, že v n-prvkovém tělese je prvek primitivní, právě když je řádu $n - 1$. Návod: viz 24.1.

24.6.2. Najděte primitivní prvek tělesa Z_7 a určete řády všech prvků tohoto tělesa.

24.7. Věta. *Každé konečné těleso má primitivní prvek.*

Důkaz. Zvolme prvek a co nejvyššího řádu r. Naším úkolem je ověřit, že $r = n - 1$, kde n je počet prvků tělesa (viz 24.6.1). Protože prvky $1, a, \ldots, a^{r-1}$ jsou navzájem různé, platí $r \leqq n - 1$ (= počet všech nenulových prvků). Na druhé straně, nerovnost $r \geqq n - 1$ dokážeme tak, že ověříme, že každý nenulový prvek je kořenem polynomu $x^r - 1$ (viz 23.4). Skutečně, pro libovolný nenulový prvek b ukážeme, že jeho řád s je dělitelem čísla r, takže $b^r = 1$ (viz 24.1). Stačí ovšem ukázat, že každý člen prvočíselného rozkladu čísla s je dělitelem čísla r. Buď tedy p prvočíslo a zapišme čísla s a r ve tvaru

$$s = p^i s' \quad \text{a} \quad r = p^j r' ,$$

kde s' a r' jsou čísla nesoudělná s prvočíslem p (a kde j může mít hodnotu 0). Naším úkolem je ověřit, že $i \leqq j$. K tomu stačí najít prvek c řádu $p^i r'$: z volby čísla r plyne $p^i r' \leqq r = p^j r'$, takže $i \leqq j$. Takovým prvkem je např.

$$c = a^{p^j} b^{s'} .$$

Platí totiž

$$c^{p^i r'} = a^{p^j p^i r'} b^{s' p^i r'} = (a^r)^{p^i} (b^s)^{r'} = 1 .$$

Dále z rovnice $c^m = 1$ odvodíme, že číslo m je dělitelné čísly p^i a r' – potom je m dělitelné i součinem těchto (nesoudělných) čísel, takže $p^i r'$ je řád prvku c. Platí

$$1 = (c^m)^{r'} = a^{p^j m r'} b^{s' m r'} = (a^r)^m b^{s' m r'} = b^{s' m r'} ,$$

takže číslo $s'mr'$ je (podle 24.1) dělitelné číslem $s = p^i s'$. Odtud plyne, že číslo m je dělitelné číslem p^i. Nakonec,

$$1 = (c^m)^{p^i} = a^{p^j m p^i} b^{s' m p^i} = a^{p^j m p^i} (b^s)^m = a^{p^j m p^i} ,$$

takže číslo $p^j m p^i$ je dělitelné číslem $r = p^j r'$. Odtud plyne, že číslo m je dělitelné číslem r'.

24.8. Důsledek (Fermatova věta). V n-prvkovém tělese T je každý prvek kořenem polynomu $x^n - x$. Platí tedy

$$x^n - x = \prod_{a \in T} (x - a) \tag{24.8.1}$$

a

$$x^{n-1} - 1 = \prod_{a \in T - \{0\}} (x - a) . \tag{24.8.2}$$

Skutečně, buď b primitivní prvek tělesa. Všechny nenulové prvky jsou $b, b^2, \ldots, b^{n-1}$, a protože $b^{n-1} = 1$, jsou všechny tyto prvky kořeny polynomu $x^{n-1} - 1$. Vztah (24.8.2) plyne z 23.3 a vynásobením obou stran členem x dostaneme (24.8.1).

24.9. Důsledek. V n-prvkovém tělese existuje prvek řádu m, právě když je m dělitelem čísla $n - 1$.

Primitivní prvek b je totiž řádu $n - 1$, takže pokud $n - 1 = mk$, je mocnina b^k řádu m (platí $(b^k)^m = b^{n-1} = 1$). Naopak, jestliže má daný prvek řád m, vyjádříme jej ve tvaru b^k. Platí $(b^k)^m = 1$, takže součin km dělí číslo $n - 1$, a proto i m dělí číslo $n - 1$.

24.10. Příklad. V tělese Z_5 platí

$$x^5 - x = x(x - 1)(x - 2)(x - 3)(x - 4).$$

Primitivní prvky jsou čísla 2 a 3, ty jsou řádu 4. Číslo $4 = 2^2$ je řádu 2. Číslo 1 je řádu 1. Řád čísla 0 není definován.

24.11. Tabulky Galoisových těles. Výhodný způsob určení Galoisova tělesa $GF(p^n) = Z_p/q(x)$ je ten, že zvolíme primitivní prvek a a vyjádříme všechny nenulové prvky (tj. nenulové polynomy proměnné z stupně $<n$) ve tvaru a^i. Potom je součet v $GF(p^n)$ obyčejný součet polynomů a součin je dán vztahem $a^i a^j = a^{i+j}$. Takovým způsobem jsou zadána „malá" Galoisova tělesa v dodatku A1.

Například těleso

$$GF(9) = Z_3/(x^2 + 1)$$

má primitivní prvek $1 + z$. (Prvek z není primitivní: $z^4 = 1$.) Platí

$$\begin{aligned} (1+z)^1 &= 1 + z & (1+z)^5 &= 2 + 2z \\ (1+z)^2 &= 2z & (1+z)^6 &= z \\ (1+z)^3 &= 1 + 2z & (1+z)^7 &= 2 + z \\ (1+z)^4 &= 2 & (1+z)^8 &= 1. \end{aligned}$$

Nyní můžeme snadno násobit v tělese $GF(9)$, např.

$$(2 + 2z)(1 + 2z) = (1 + z)^{5+3} = 1.$$

25. Charakteristika tělesa

Charakteristika tělesa T je prvočíslo p takové, že těleso T je rozšířením tělesa Z_p. Potom i polynomy nad Z_p se stávají polynomy nad tělesem T. Tyto polynomy mají důležitou vlastnost: s každým kořenem a jsou i a^p, a^{2p}, a^{3p} ... kořeny. Všechna tato fakta směřují k popisu konečných těles a k rozkladu polynomu $x^n - 1$.

25.1. Charakteristika tělesa. Charakteristikou tělesa T se nazývá nejmenší číslo p takové, že součet $1 + 1 + \ldots + 1$ (s p sčítanci) je roven 0. Pokud takové číslo neexistuje, má těleso charakteristiku ∞.

Těleso Z_2 má charakteristiku 2 a stejnou charakteristiku mají jeho rozšíření $GF(2^n) = Z_2/q(x)$. Obecněji, Galoisovo těleso $GF(p^n)$ má charakteristiku p. Těleso reálných čísel má ovšem charakteristiku ∞.

25.2. Tvrzení. Charakteristika každého konečného tělesa je prvočíslo.

Důkaz. Označme $a_n = 1 + 1 + \ldots + 1$ (n sčítanců). Tyto prvky nemohou být v konečném tělese navzájem různé pro $n = 1, 2, 3, \ldots$. Zvolme nejmenší číslo p takové, že $a_{n+p} = a_n$ (pro některé n). Platí

$$a_p = a_{n+p} - a_n = 0,$$

a tedy p je charakteristika tělesa T.

Abychom ověřili, že p je prvočíslo, ukážeme, že kdykoli $p = uv$ a $u < p$, platí $p = v$ (a $u = 1$). Skutečně,

$$0 = a_p = a_u + a_u + \ldots + a_u \quad (v \text{ sčítanců}),$$

a protože z $u < p$ plyne $a_u \neq 0$, můžeme tuto rovnost dělit prvkem a_u a dostaneme $0 = 1 + 1 + \ldots + 1 = a_v$. Potom $v = p$.

25.3. Poznámka. Každé těleso T charakteristiky p je rozšířením tělesa Z_p. Prvky

$$0, 1, 2 = 1 + 1, 3 = 1 + 1 + 1, \ldots, p - 1 = 1 + 1 + \ldots + 1$$

tělesa T jsou totiž navzájem různé, ale $(p - 1) + 1 = 0$. Odtud plyne, že sčítání těchto prvků v tělese T je stejné jako v tělese Z_p: platí $i + j = i + j - p$, protože $p = 0$. Podobně je tomu s dalšími operacemi: odečítáním, násobením a dělením.

Naopak ovšem každé rozšíření tělesa Z_p je charakteristiky p.

25.4. Úlohy. Buď T těleso charakteristiky p.

25.4.1. Pro každý prvek a v Z_p (tj. každé celé číslo tělesa T) platí $a^p = a$ (viz 24.8). Ověřte, že naopak z rovnosti $a^p = a$ plyne, že a je celé číslo. Návod: 23.4.

25.4.2. Odvoďte, že řád žádného prvku není dělitelný číslem p. Návod: 24.1.

25.5. Věta. *V tělese charakteristiky p platí*

$$(a + b)^p = a^p + b^p$$

pro libovolné prvky a a b.

Důkaz. Jde o snadnou aplikaci *binomické věty*, kterou ovšem musíme napřed dokázat. Zavedeme obvyklým způsobem faktoriály:

$$0! = 1 \quad \text{a} \quad k! = k(k - 1)(k - 2) \ldots . 2 . 1 .$$

Platí ovšem $p! = 0$ (protože $p = 0$), ale $k! \neq 0$ pro všechna $k < p$. Můžeme tedy definovat kombinační čísla

$$\binom{n}{k} = \frac{n!}{(n - k)!\, k!}, \quad 1 \leqq k < n \leqq p$$

a dále $\binom{n}{0} = \binom{n}{n} = 1$ $(n = 0, 1, \ldots, p)$.

Snadno ověříte, že platí

$$\binom{n}{k} + \binom{n}{k-1} = \binom{n+1}{k}.$$

Binomická věta říká, že

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$$

pro $n = 0, 1, 2, \ldots$. Dokážeme ji matematickou indukcí. Pro $n = 0$ jsou obě strany rovny 1. Indukční krok ověříme úpravou a změnou indexu (k změníme na $k-1$):

$$\begin{aligned}
(a+b)^{n+1} &= (a+b)^n a + (a+b)^n b = \\
&= \sum_{k=0}^{n} \binom{n}{k} a^{k+1} b^{n-k} + \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k+1} = \\
&= \sum_{k=1}^{n+1} \binom{n}{k-1} a^k b^{n-k+1} + \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k+1} = \\
&= \binom{n}{n} a^{n+1} + \sum_{k=1}^{n} \left[\binom{n}{k-1} + \binom{n}{k} \right] a^k b^{n-k+1} + \binom{n}{0} b^{n+1} = \\
&= a^{n+1} + \sum_{k=1}^{n} \binom{n+1}{k} a^k b^{n+1-k} + b^{n+1} = \\
&= \sum_{k=0}^{n+1} \binom{n+1}{k} a^k b^{n+1-k}.
\end{aligned}$$

Pokud $n = p$, platí $\binom{p}{k} = 0$ pro všechna k kromě 0 a p, takže

$$(a+b)^p = a^p + b^p.$$

25.6. Úlohy. Ověřte, že v tělese charakteristiky p platí:

25.6.1. $(a_1 + a_2 + \ldots + a_n)^p = a_1^p + a_2^p + \ldots + a_n^p$. Návod: matematickou indukcí podle n.

25.6.2. $(a+b)^{p^m} = a^{p^m} + b^{p^m}$. Návod: matematickou indukcí podle m.

25.6.3. $(x+a)^p = x^p + a^p$ (ve smyslu násobení polynomů). Návod: důkaz je zcela analogický důkazu rovnosti $(a+b)^p = a^p + b^p$. Uvědomte si však, že nestačí dosadit $x = b$ a použít předchozí věty.

25.6.4. $(a_0 + a_1 x + a_2 x^2 + \ldots + a_k x^k)^p = a_0^p + a_1^p x^p + a_2^p x^{2p} + \ldots + a_k^p x^{kp}$ (ve smyslu násobení polynomů). Návod: matematickou indukcí podle k.

25.7. Důsledek. Buď T těleso charakteristiky p. Jestliže polynom $f(x)$ nad tělesem Z_p má kořen a v tělese T, potom jsou i $a^p, a^{2p}, a^{3p} \ldots$ kořeny polynomu $f(x)$.

Skutečně, pro koeficienty f_i polynomu $f(x)$ platí $f_i^p = f_i$ (25.4.1), takže

$$f(a^p) = f_0^p + f_1^p a^p + f_2^p a^{2p} + \ldots + f_k^p a^{kp} =$$
$$= (f_0 + f_1 a + f_2 a^2 + \ldots + f_k a^k)^p =$$
$$= 0^p .$$

25.8. Rozklad polynomu $x^n - 1$

V teorii cyklických kódů potřebujeme rozložit polynom $x^n - 1$ v součin lineárních polynomů. Jestliže kódovou abecedou je Z_p, kde p je prvočíslo, stačí najít prvek řádu n v některém rozšíření tělesa Z_p – viz 24.3. To znamená, že hledáme mocnitele m tak, aby v tělese $GF(p^m)$ existoval prvek řádu n. To není možné, pokud je p dělitelem čísla n, viz 25.4.2. Budeme tedy pracovat jen s čísly n, nedělitelnými číslem p.

Obecněji můžeme předpokládat, že kódová abeceda je $GF(q)$, tedy že jde o q-znakový kód, kde q je mocnina prvočísla. Pro číslo n, které je nesoudělné s q, hledáme mocninu q^m takovou, že těleso $GF(q^m)$ obsahuje prvek řádu n. Ukážeme, že taková mocnina vždy existuje a je ji možno určit pomocí Eulerovy funkce.

25.8.1. Eulerova funkce $\varphi(n)$. Pro každé číslo $n = 1, 2, 3, \ldots$ definujeme

$\varphi(n)$ = počet čísel $1, 2, \ldots, n - 1$, nesoudělných s číslem n.

Například $\varphi(3) = 2$, protože 1 i 2 jsou čísla nesoudělná s číslem 3. Obecněji:

$\varphi(p) = p - 1$ pro každé prvočíslo p

(protože všechna uvažovaná čísla jsou nesoudělná s číslem p). Ještě obecněji:

(25.8.1) $\varphi(p^k) = p^k - p^{k-1}$ pro každé prvočíslo p a každé $k = 1, 2, 3, \ldots$.

(Jediná čísla $i < p^k$, soudělná s číslem p^k, jsou čísla $i = pj$, kde $j = 1, 2, \ldots, p^{k-1} - 1$, a jejich počet je p^{k-1}.) Dále uveďme, že pro dvě různá prvočísla p a q platí

(25.8.2) $\varphi(pq) = (p - 1)(q - 1)$.

(Jediná soudělná čísla s pq jsou pi pro $i = 1, \ldots, q - 1$ a iq pro $i = 1, \ldots, p - 1$, takže počet nesoudělných čísel je $pq - (q - 1) - (p - 1) - 1 = (p - 1)(q - 1)$.) Odtud plyne tato tabulka:

n	1	2	3	4	5	6	7	8	9	10	11
$\varphi(n)$	0	1	2	2	4	2	6	4	6	4	10

25.8.2. Fermatova-Eulerova věta. *Pro každá dvě nesoudělná čísla q a n platí*

$$q^{\varphi(n)} \equiv 1 \pmod{n},$$

tj. *číslo n je dělitelem čísla $q^{\varphi(n)} - 1$.*

Důkaz. Budeme pracovat v okruhu $Z/\mathrm{mod}\ n$ tříd celých čísel modulo n (9.8). Každá třída

$$[a] = \{a + kn \mid k \text{ celé číslo}\}$$

čísla a, nesoudělného s číslem n, sestává ze samých čísel, která jsou s n nesoudělná. (Pokud totiž čísla n a $x = a + kn$ mají společného dělitele d, potom d dělí i číslo kn,

a tedy i rozdíl $x - kn = a$, takže $d = 1$.) Přitom některý prvek třídy $[a]$ leží mezi 1 a $n - 1$. (Je to zbytek po dělení čísla a číslem n – ten nemůže být roven 0.)

Podle definice Eulerovy funkce existuje právě $\varphi(n)$ čísel $a_1, a_2, \ldots, a_{\varphi(n)}$ od 1 do $n - 1$, nesoudělných s číslem n. To znamená, že existuje právě $\varphi(n)$ tříd $[a_1], [a_2], \ldots, [a_{\varphi(n)}]$ čísel, nesoudělných s číslem n. Ovšem každé z čísel $qa_1, qa_2, \ldots, qa_{\varphi(n)}$ je také s n nesoudělné a navíc jsou jejich třídy navzájem různé. Jestliže totiž $[qa_i] = [qa_j]$, potom rozdíl $qa_i - qa_j = q(a_i - a_j)$ je dělitelný číslem n, takže i číslo $|a_i - a_j|$ je dělitelné číslem n, ale $|a_i - a_j| = 0, 1, \ldots, n - 1$, takže $a_i - a_j = 0$ (tj. $i = j$). Odtud plyne, že třídy $[qa_1], [qa_2], \ldots, [qa_{\varphi(n)}]$ jsou až na pořadí shodné s třídami $[a_1], [a_2], \ldots, [a_{\varphi(n)}]$. Proto součiny těchto prvků v okruhu $Z/\mathrm{mod}\ n$ jsou také shodné:

$$[a_1][a_2]\ldots[a_{\varphi(n)}] = [qa_1][qa_2]\ldots[qa_{\varphi(n)}] .$$

Číslo $a = a_1 a_2 \ldots a_{\varphi(n)}$ je ovšem nesoudělné s číslem n a podle poslední rovnice platí

$$[a] = [q^{\varphi(n)} a] ,$$

takže rozdíl $q^{\varphi(n)}a - a = (q^{\varphi(n)} - 1)\, a$ je dělitelný číslem n. Odtud plyne, že číslo $q^{\varphi(n)} - 1$ je dělitelné číslem n, a to jsme měli dokázat.

25.8.3. Důsledek. Buď q mocnina prvočísla a buď n číslo, nesoudělné s číslem q. Potom v tělese $GF(q^{\varphi(n)})$ existuje prvek a řádu n a platí

$$x^n - 1 = (x - 1)(x - a)\ldots(x - a^{n-1}) \quad \text{v} \quad GF(q^{\varphi(n)}) .$$

Skutečně, položme $m = \varphi(n)$. Podle předchozí věty je číslo n dělitelem čísla $q^m - 1$, takže hledaný prvek a existuje podle 24.9. Potom platí uvedený rozklad polynomu $x^n - 1$ podle 24.3.

Často můžeme volit menšího mocnitele m než číslo $\varphi(n)$. Nejmenší mocnitel je vždy dělitelem čísla $\varphi(n)$.

25.8.4. Příklad. Chceme rozložit polynom $x^7 - 1$ nad tělesem, rozšiřujícím těleso Z_2. Protože jsou čísla 2 a 7 nesoudělná, a protože $\varphi(7) = 6$, víme, že nad tělesem $GF(2^6) = GF(64)$ existuje rozklad v lineární činitele. Ale i v tělese $GF(2^3)$ takový rozklad najdeme. Primitivní prvek tělesa $GF(2^3)$ je řádu 7. Například zvolme prvek z tělesa $Z_2/(x^3 + x + 1)$ (viz dodatek A1). Platí

$$x^7 - 1 = (x - 1)(x - z)(x - z^2)(x - z^3)(x - z^4)(x - z^5)(x - z^6) .$$

O tom se můžete přesvědčit roznásobením.

25.8.5. Úloha. Najděte nejmenší rozšíření tělesa (a) Z_2, (b) Z_3 a (c) $GF(4)$, ve kterém má polynom $x^9 - 1$ rozklad v součin lineárních činitelů. V tělese $GF(4)$ (viz A1) takový rozklad skutečně napište. Návod: v tělese $GF(64)$ je prvek z řádu 63, a tedy prvek z^7 je řádu 9.

25.9. Rozšíření tělesa $GF(q)$. Víme, že Galoisovo těleso $GF(p^m)$ je rozšířením tělesa Z_p (pomocí ireducibilního polynomu stupně m). Obecněji, pro každou mocninu

prvočísla $q = p^s$ je Galoisovo těleso $GF(q^m)$ rozšířením tělesa $GF(q)$, jinými slovy, $GF(p^{ms})$ je rozšířením tělesa $GF(p^s)$:

25.9.1. Tvrzení. V tělese $GF(q^m)$, kde q je mocninou prvočísla, tvoří všechny kořeny polynomu $x^q - x$ těleso o q prvcích, jehož rozšířením je těleso $GF(q^m)$.

Důkaz. V tělese $GF(q^m)$ existuje prvek b řádu $q - 1$, protože číslo $q - 1$ dělí číslo $q^m - 1 = (q - 1)(q^{m-1} + q^{m-2} + \ldots + 1)$ (viz 24.9). Platí $b^{q-1} = 1$, a tedy $b^q = b$. Prvky

$$0, 1, b, b^2, \ldots, b^{q-2}$$

jsou navzájem různé a každý z nich je kořenem polynomu $x^q - x$. (Pro každé i je totiž $(b^i)^q - b^i = (b^q)^i - b^i = b^i - b^i = 0$.) To znamená, že to jsou právě všechny kořeny polynomu $x^q - x$ (23.4). Máme dokázat, že množina

$$T = \{0, 1, b, \ldots, b^{q-2}\}$$

je tělesem, jehož rozšířením je těleso $GF(q^m)$. Jinými slovy, operace tělesa T jsou shodné s operacemi tělesa $GF(q^m)$. Stačí tedy ukázat, že součet, rozdíl, součin a podíl prvků množiny T opět leží v množině T.

Pro $a_1, a_2 \in T$ platí $a_1^q = a_1$ a $a_2^q = a_2$. Podle 25.6.2 platí

$$(a_1 + a_2)^q = a_1^q + a_2^q = a_1 + a_2 ,$$

takže $a_1 + a_2 \in T$; analogicky $a_1 - a_2 \in T$, protože $(-a_2)^q = -a_2^q$. Součet a podíl dvou kořenů polynomu $x^q = x$, je zřejmě také kořenem. Proto je T těleso.

25.9.2. Důsledek. Rozšířeními tělesa $GF(q^k)$ jsou všechna tělesa $GF(q^{km})$, $m = 1, 2, 3, \ldots$.

To plyne z předchozího tvrzení, které aplikujeme na q^k (což je ovšem mocnina prvočísla) místo na q.

25.9.3. Příklad. V tělese $GF(16)$ (viz dodatek A1) jsou kořeny rovnice $x^4 - x = 0$ právě prvky

$$0, 1, z^5 \text{ a } z^{10} .$$

Ty tvoří těleso $GF(4)$.

26. Minimální polynomy

Každý prvek a konečného tělesa je kořenem některého polynomu s celočíselnými koeficienty — např. polynomu $x^n - x$ (24.8). Minimální polynom prvku a je celočíselný polynom nejnižšího možného stupně s kořenem a. Ukážeme, jaké jsou základní vlastnosti minimálních polynomů a jak se tyto polynomy hledají. Pomocí minimálních polynomů dokážeme, že každé konečné těleso je Galoisovo těleso.

26.1. Definice. Buď T těleso charakteristiky p. *Minimálním polynomem* prvku $a \in T$ nazýváme nenulový polynom $f(x)$ nad tělesem Z_p minimálního stupně takový, že $f(a) = 0$.

26.2. Příklad. Těleso

$$GF(4) = Z_2/(x^2 + x + 1)$$

je charakteristiky 2. Minimální polynom prvku z je polynom $f(x) = x^2 + x + 1$ (nad Z_2). Platí totiž $f(z) = z^2 + z + 1 = 0$ a pro každý polynom $g(x)$ stupně 1 nad Z_2 (tj. pro polynomy x a $x + 1$) je $g(z) \neq 0$.

Prvek $z^2 = 1 + z$ má také minimální polynom $f(x)$, protože $f(z^2) = z^4 + 1 = z + z^2 + 1 = 0$ a přitom $g(z^2) \neq 0$ pro každý polynom $g(x)$ stupně 1.

Prvky samotného tělesa Z_2 mají ovšem minimální polynomy stupně 1: minimální polynom prvku 0 je x a minimální polynom prvku 1 je $x - 1$.

26.3. Tvrzení. Minimální polynom prvku a je dělitelem každého polynomu nad tělesem Z_p s kořenem a.

Důkaz. Buď $f(x)$ minimální polynom a $g(x)$ libovolný polynom nad tělesem Z_p takový, že $g(a) = 0$. Dělíme polynom $g(x)$ polynomem $f(x)$ nad tělesem Z_p:

$$g(x) = q(x) f(x) + r(x), \quad \text{st}\, r(x) < \text{st}\, f(x).$$

Dosazením prvku a dostáváme $0 = 0 + r(a)$. To znamená, že $r(x)$ je polynom nad tělesem Z_p, který má kořen a a přitom je nižšího stupně než minimální polynom – takový polynom musí být nulový. Platí tedy $g(x) = q(x) f(x)$.

26.4. Důsledek. Minimální polynom je (až na konstantní násobek) jediný. Podrobněji, každý prvek má právě jeden normovaný (23.10.5) minimální polynom.

To plyne z faktu, že pokud polynom $f(x)$ je dělitelem polynomu $g(x)$ a také naopak, $g(x)$ je dělitelem polynomu $f(x)$, potom existuje konstanta k taková, že $f(x) = k\, g(x)$ (a to $k = f_n g_n^{-1}$, kde $n = \text{st}\, f(x) = \text{st}\, g(x)$). Jestliže jsou oba polynomy normované, platí $k = 1$.

26.5. Příklad. Hledáme minimální polynom prvku $1 + z$ v tělese $GF(9) = Z_3/(x^2 + 1)$, viz 24.11. Je zřejmé, že minimální polynom není stupně 1 (pak by musel být tvaru $x - (1 + z)$, ale to není polynom nad tělesem Z_3). Všimneme si, že $(1 + z)^2 = 2z$, takže

$$(1 + z)^2 - 2(1 + z) + 2 = 0.$$

Minimálním polynomem je tedy $f(x) = x^2 - 2x + 2$. Také jeho konstantní násobek $2f(x) = 2x^2 - x + 1$ je minimální polynom. Další minimální polynomy prvek $1 + z$ nemá.

26.6. Poznámka. Každý minimální polynom je ireducibilní nad tělesem Z_p. Jestliže je totiž $f(x) = p(x)\, q(x)$ minimální polynom prvku a, pak z rovnice

$p(a)\,q(a) = 0$ plyne, že jeden z polynomů $p(x)$ a $q(x)$ má kořen a, a ten pak má stejný stupeň jako polynom $f(x)$.

Také naopak: pokud má ireducibilní polynom $f(x)$ nad tělesem Z_p kořen a v některém rozšíření tělesa Z_p, je $f(x)$ minimálním polynomem prvku a. To plyne z 26.3.

26.7. Úloha. Dokažte, že pro každý prvek n-prvkového tělesa je minimální polynom dělitelem polynomu $x^n - x$ (nad tělesem Z_p). Návod: Fermatova věta a 26.3.

26.8. Pomocí minimálních polynomů ukážeme, že každé konečné těleso je izomorfní s některým Galoisovým tělesem. Dvě tělesa T a T' jsou *izomorfní*, jestliže existuje vzájemně jednoznačné zobrazení

$$t\colon T \to T'$$

(tj. zobrazení prosté a na), zachovávající algebraické operace:

$$\begin{aligned} t(a+b) &= t(a) + t(b)\,,\\ t(a-b) &= t(a) - t(b)\,,\\ t(ab) &= t(a)\,t(b)\,,\\ t(a/b) &= t(a)/t(b)\,, \quad \text{pokud} \quad b \neq 0\,. \end{aligned}$$

Takové zobrazení se nazývá *izomorfismus*. Tělesa T a T' jsou potom v podstatě totožná: jestliže prvek a tělesa T „přejmenujeme" na prvek $t(a)$ tělesa T', potom toto přejmenování nemění algebraickou strukturu. Příklad izomorfismu jsme viděli v (9.8): Těleso Z_2 (s prvky 0 a 1) je izomorfní s tělesem $Z/\mathrm{mod}\,2$ (s prvky $[0]$ a $[1]$). Izomorfismus $t\colon Z_2 \to Z/\mathrm{mod}\,2$ jen přejmenovává prvky přidáním závorek.

26.8.1. Poznámka. Vzájemně jednoznačné zobrazení mezi tělesy, $t\colon T \to T'$, je izomorfismus právě když pro všechna a a b v T platí

$$(26.8.1) \qquad \begin{aligned} &t(a+b) = t(a) + t(b)\,, \quad t(ab) = t(a)\,t(b)\,,\\ &t(0) = 0 \quad \text{a} \quad t(1) = 1\,. \end{aligned}$$

Skutečně, jestliže zobrazení t splňuje tuto podmínku, pak zachovává i operaci odečítání : platí

$$t(b) + t(-b) = t(b + (-b)) = t(0) = 0\,,$$

takže $t(-b) = -t(b)$, a odtud plyne

$$t(a - b) = t(a + (-b)) = t(a) - t(b)\,;$$

podobně pro $b \neq 0$ platí

$$t(1/b)\,t(b) = t(1/b\,.\,b) = t(1) = 1\,,$$

takže $t(1/b) = 1/t(b)$, a odtud plyne

$$t(a/b) = t(a\,.\,1/b) = t(a)\,.\,1/t(b) = t(a)/t(b)\,.$$

Naopak, každý izomorfismus splňuje

$$t(0) = t(1 - 1) = t(1) - t(1) = 0$$

a

$$t(1) = t(1/1) = t(1)/t(1) = 1\,.$$

Proto se izomorfismus obvykle definuje (stručnější) podmínkou (26.8.1).

26.9. Věta o konečných tělesech. Každé konečné těleso je izomorfní s Galoisovým tělesem.

Poznámka. Podrobněji, každé konečné těleso T charakteristiky p je izomorfní s tělesem $Z_p/q(x)$, kde $q(x)$ je minimální polynom primitivního prvku tělesa T.

Důkaz. Buď T konečné těleso charakteristiky p. Zvolme primitivní prvek a tělesa T (24.7) a označme $q(x)$ jeho normovaný minimální polynom (26.4). Prvky tělesa $Z_p/q(x)$ jsou polynomy $f(z)$ nad tělesem Z_p stupně $<n$, kde $n = \operatorname{st} q(x)$. Každému takovému polynomu přiřaďme jeho hodnotu $f(a)$ v tělese T. Tak vznikne zobrazení

$$t\colon Z_p/q(x) \to T,$$

definované předpisem

$$t(f(z)) = f(a)\,.$$

Dokážeme, že t je izomorfismus.

I. Zobrazení t je prosté. Jinými slovy, pro každé dva polynomy $f(z)$ a $\tilde{f}(z)$ stupně $<n$ nad Z_p takové, že $f(a) = \tilde{f}(a)$, platí $f(z) = \tilde{f}(z)$. Polynom $f(z) - \tilde{f}(z)$ má totiž kořen a a přitom je nižšího stupně než n (= stupeň minimálního polynomu $q(x)$), takže je $f(z) - \tilde{f}(z)$ nulový polynom.

II. t je zobrazení na. Nulový polynom 0 splňuje $t(0) = 0$, takže stačí ověřit, že každý nenulový prvek tělesa T, tj. prvek tvaru a^i, leží v oboru hodnot zobrazení t. To je zřejmé, pokud $i < n$: platí $t(z^i) = a^i$. Pro $i \geqq n$ využijeme toho, že (normovaný) polynom $q(x)$ má kořen a. Platí

$$q(a) = q_0 + q_1 a + \ldots + q_{n-1}a^{n-1} + a^n = 0\,,$$

takže všechny mocniny a^i, kde $i \geqq n$, můžeme vyjádřit pomocí mocnin $1, a, a^2, \ldots$ $\ldots, a^{n-1}$, a tak najít polynom $f(z)$ stupně $<n$ takový, že $f(a) = a^i$. Například

$$a^n = -q_0 - q_1 a - \ldots - q_{n-1}a^{n-1}\,,$$

a tedy

$$t(-q_0 - q_1 z - \ldots - q_{n-1}z^{n-1}) = a^n\,.$$

Podobně

$$\begin{aligned} a^{n+1} &= -q_0 a - q_1 a^2 - \ldots - q_{n-2}a^{n-1} - q_{n-1}a^n = \\ &= -q_0 a - q_1 a^2 - \ldots - q_{n-2}a^{n-1} - q_{n-1}(-q_0 - q_1 a - \ldots \\ &\quad \ldots - q_{n-1}a^{n-1})\,, \end{aligned}$$

atd.

III. Zobrazení t zachovává sčítání, protože

$$t(f(z) + \tilde{f}(z)) = f(a) + \tilde{f}(a) = t(f(z)) + t(\tilde{f}(z))\,.$$

IV. Zobrazení t zachovává násobení. Skutečně, je-li $h(z) = f(z)\tilde{f}(z)$ součin v tělese $Z_p/q(x)$, je polynom $h(x)$ zbytkem po dělení obyčejného součinu $k(x) = f(x)\tilde{f}(x)$ polynomem $q(x)$. Protože $q(a) = 0$, plyne odtud $h(a) = k(a)$. Proto platí

$$\begin{aligned} t(f(z)\tilde{f}(z)) &= t(h(z)) = \\ &= h(a) = \\ &= k(a) = \\ &= f(a)\tilde{f}(a) = \\ &= t(f(z))\, t(\tilde{f}(z)) . \end{aligned}$$

V. Platí $t(0) = 0$ a $t(1) = 1$.

26.10. Důsledek. Počet prvků každého konečného tělesa je mocninou prvočísla.

Žádné 6-prvkové těleso tedy neexistuje. Později dokážeme i opačné tvrzení: každá mocnina prvočísla je počtem prvků některého tělesa.

26.11. Věta. *Buď T konečné těleso charakteristiky p. Potom minimálním polynomem prvku $a \in T$ je polynom*

$$f(x) = (x - a)(x - a^p)(x - a^{p^2}) \dots (x - a^{p^m}),$$

kde m je nejmenší číslo takové, že $a = a^{p^{m+1}}$.

Důkaz. I. Nejprve si uvědomíme, že takové číslo m existuje. Počet prvků tělesa T je (podle poznámky 26.9) p^n, takže platí $a^{p^n} = a$ (24.8). Číslo m tedy najdeme mezi čísly $0, 1, 2, \dots, n - 1$.

II. Ověříme, že uvedený polynom $f(x)$ má koeficienty v tělese Z_p. Stačí dokázat, že koeficienty f_i tohoto polynomu $f(x) = f_0 + f_1 x + \dots + f_k x^k$ splňují $f_i^p = f_i$, takže $f_i \in Z_p$ (25.4.1). Podle 25.6.4 platí

$$[f(x)]^p = f_0^p + f_1^p x^p + \dots + f_k^p x^{kp} .$$

Polynom $[f(x)]^p$ má tedy u mocniny x^{ip} koeficient f_i^p $(i = 0, \dots, k)$. Na druhé straně, protože

$$(x - a^i)^p = x^p - a^{ip}$$

(25.6.3), platí

$$\begin{aligned} [f(x)]^p &= (x - a)^p (x - a^p)^p \dots (x - a^{p^{m-1}})^p (x - a^{p^m})^p = \\ &= (x^p - a^p)(x^p - a^{p^2}) \dots (x^p - a^{p^m})(x^p - a^{p^{m+1}}) . \end{aligned}$$

Z volby čísla m plyne, že poslední součinitel je $x^p - a$, takže

$$[f(x)]^p = (x^p - a)(x^p - a^p)(x^p - a^{p^2}) \dots (x^p - a^{p^m}) = f(x^p) .$$

Polynom $f(x^p)$ má u mocniny x^{ip} koeficient f_i. Platí tedy $f_i^p = f_i$.

III. Polynom $f(x)$ je minimální. Platí ovšem $f(a) = 0$. Buď $g(x)$ polynom nad tělesem Z_p s kořenem a. Potom také $a^p, a^{p^2}, \dots, a^{p^m}$ jsou kořeny polynomu $g(x)$ (25.7), takže polynom $f(x)$ je dělitelem polynomu $g(x)$ (23.3).

26.12. Příklad. Minimální polynom prvku $a = z^2 + z$ v tělese $GF(16)$ (viz dodatek A1). Platí $z^2 + z = z^5$. Prvky z^5 a z^{10} jsou různé, ale $z^{20} = z^5 = a$. Proto je minimální polynom

$$f(x) = (x - z^5)(x - z^{10}) = x^2 + x + 1 .$$

Prvek z (a tedy i z^2, z^4 a z^8) má minimální polynom $x^4 + x + 1$. Prvek z^3 (a tedy i z^6, z^{12} a $z^{24} = z^9$) má minimální polynom

$$f(x) = (x - z^3)(x - z^6)(x - z^{12})(x - z^9) = x^4 + x^3 + x^2 + x + 1 .$$

26.13. Úlohy

26.13.1. Určete minimální polynomy všech prvků tělesa $GF(8)$.

26.13.2. Určete minimální polynomy všech prvků tělesa $GF(9)$.

26.14. Poznámka. Minimální polynomy můžeme obecněji zavést nad každým tělesem T_0, jehož rozšířením (23.13) je dané těleso T. *Minimální polynom* prvku $a \in T$ *nad tělesem* T_0 je nenulový polynom co nejnižšího stupně nad T_0 s kořenem a. Protože a je kořenem polynomu $x^{p^n} - x$, který má koeficienty v T_0, minimální polynom vždy existuje. Je dělitelem všech polynomů nad T_0 s kořenem a.

Například těleso $GF(16)$ je rozšířením tělesa

$$GF(4) = \{0, 1, z^5, z^{10}\}$$

(25.9.3). Minimální polynom prvku $a \in GF(16)$ nad tělesem $GF(4)$ je tedy polynom s koeficienty 0, 1, z^5 nebo z^{10}. Například minimální polynom prvku z^5 nad tělesem $GF(4)$ je polynom lineární: $x - z^5$. Minimální polynom prvku z^2 nad tělesem $GF(4)$ je

$$f(x) = x^2 + x + z^{10} .$$

Platí totiž $f(z^2) = z^4 + z^2 + z^{10} = 0$ a žádný lineární polynom nad tělesem $GF(4)$ nemá kořen z^2. Srovnejte s 26.12.

27. Konečná tělesa

Víme již, že v podstatě jediná konečná tělesa jsou Galoisova tělesa (26.9). Zbývá dokázat, že Galoisovo těleso $Z_p/q(x)$ nezávisí na výběru ireducibilního polynomu $q(x)$ (ale jen na jeho stupni) a že existují ireducibilní polynomy všech možných stupňů.

27.1. Věta o jednoznačnosti. *Každá dvě konečná tělesa o stejném počtu prvků jsou izomorfní.*

Důkaz. Nechť T a T' jsou konečná tělesa o stejném počtu prvků. Podle 26.9 existuje izomorfismus

$$t: Z_p/q(x) \to T ,$$

kde p je charakteristika tělesa T a $q(x)$ je minimální polynom (stupně n) některého prvku tělesa T. Tento polynom je dělitelem polynomu $x^{p^n} - x$ (26.7).

Těleso T' má stejný počet prvků jako T, tedy p^n. Má charakteristiku p (podle poznámky 26.9, aplikované na těleso T'). Nad tělesem T' máme tedy rozklad na lineární činitele

$$x^{p^n} - x = \prod_{b \in T'} (x - b)$$

(24.8), a protože polynom $q(x)$ je dělitelem polynomu $x^{p^n} - x$, je také součinem lineárních činitelů nad tělesem T'. Zvolme kořen b polynomu $q(x)$ v tělese T'. Pro každý polynom $f(z)$ stupně $< n$ nad Z_p označme

$$s(f(z)) = f(b)$$

hodnotu polynomu v bodě b. Tím vzniká zobrazení

$$s: Z_p/q(x) \to T' .$$

Dokážeme, že s je izomorfismus (analogicky jako v důkazu věty 26.9).

Zobrazení s je prosté: Polynom $q(x)$ je minimálním polynomem prvku b (26.6), takže pro každé dva polynomy $f(z)$ a $\tilde{f}(z)$ stupně $< n$ takové, že $f(b) = \tilde{f}(b)$ platí $f(z) = \tilde{f}(z)$. Polynom $f(z) - \tilde{f}(z)$ má totiž kořen b a je stupně nejvýše st $q(x) - 1$; je to tedy nulový polynom. Protože obě množiny $Z_p/q(x)$ a T' mají stejný počet prvků, je zobrazení s vzájemně jednoznačné.

Zobrazení s zachovává sčítání:

$$s(f(z) + \tilde{f}(z)) = f(b) + \tilde{f}(b) = s(f(z)) + s(\tilde{f}(z)) .$$

Zobrazení s zachovává násobení. Označme $f(z)\tilde{f}(z) = h(z)$ součin v tělese $Z_p/q(x)$, tj. $h(x)$ je zbytek po dělení obyčejného součinu $k(x) = f(x)\tilde{f}(x)$ nad Z_p polynomem $q(x)$. Protože $q(b) = 0$, plyne odtud $h(b) = k(b)$. Je tedy

$$s(f(z)\tilde{f}(z)) = h(b) = k(b) = f(b)\tilde{f}(b) = s(f(z))\, s(\tilde{f}(z)) .$$

Nakonec $s(0) = 0$ a $s(1) = 1$ — to je zřejmé.

Ověřili jsme, že s je izomorfismus. Odtud snadno plyne, že i inverzní zobrazení s^{-1} (definované předpisem: $s^{-1}(y) = x$, právě když $s(x) = y$) je izomorfismem

$$s^{-1}: T' \to Z_p/q(x) .$$

Zobrazení složené ze dvou izomorfismů

$$ts^{-1}: T' \to T$$

je rovněž izomorfismem.

27.2. Příklad. Polynomy $x^2 + 1$ a $x^2 + x + 2$ jsou oba ireducibilní nad tělesem Z_3 (protože ani jeden nemá kořeny nad Z_3). Tělesa $Z_3/(x^2 + 1)$, viz 24.11, a $Z_3/(x^2 + x + 2)$, viz A1, jsou tedy izomorfní — a obě označujeme $GF(9)$.

Skutečně, zobrazení

$$s: Z_3/(x^2 + 1) \to Z_3/(x^2 + x + 2)$$

definované předpisem

$$s(f_0 + f_1 z) = f_0 + f_1 + f_1 z$$

je izomorfismus. Snadno se o tom přesvědčíte při použití popisu těles.

27.3. Věta o existenci. *Pro každé prvočíslo p existují ireducibilní polynomy všech stupňů $n = 1, 2, 3, \ldots$ nad tělesem Z_p, a existují tedy Galoisova tělesa $GF(p^n)$.*

Důkaz. Postup důkazu bude opačný: sestrojíme těleso T o p^n prvcích. Podle 26.9 je minimální polynom primitivního prvku tělesa T hledaným ireducibilním polynomem stupně n.

I. Těleso T^*: sestrojíme nejdříve těleso T^* charakteristiky p, ve kterém se polynom

$$x^{p^n} - x$$

rozkládá v součin lineárních činitelů.

Definujeme postupně tělesa $T_0^* \subseteq T_1^* \subseteq T_2^* \ldots$ tak, že některé z nich má hledaný rozklad. Položme nejdříve

$$T_0^* = Z_p .$$

Polynom $x^{p^n} - x$ rozložíme nad tímto tělesem v součin normovaných ireducibilních činitelů (23.14.6):

$$x^{p^n} - x = a_1(x)\, a_2(x) \ldots a_{k_0}(x) \quad \text{nad} \quad T_0^* .$$

Jestliže platí $k_0 = p^n$, jsou všechny tyto členy nutně lineární a jsme hotovi, $T = T_0^*$. Naopak v případě, že $k_0 < p^n$, zvolíme polynom $a_i(x)$ stupně $\geqq 2$ a položíme

$$T_1^* = T_0^*/a_i(x) .$$

Polynom $x^{p^n} - x$ rozložíme nad tímto tělesem opět v součin normovaných ireducibilních polynomů

$$x^{p^n} - x = b_1(x)\, b_2(x) \ldots b_{k_1}(x) \quad \text{nad} \quad T_1^* .$$

Každý lineární činitel z minulého rozkladu se opakuje i zde. Pokud totiž $a_j(x) = x - c$, je c kořenem polynomu $x^{p^n} - x$ nad tělesem T_0^* (a tedy i nad tělesem T_1^*), takže $b_{j'}(x) = x - c$ pro některé j'. Navíc prvek z tělesa $T_0^*/a_i(x)$ splňuje $a_i(z) = 0$, a tedy je z novým kořenem polynomu $x^{p^n} - x$ nad tělesem T_1^*. Odtud plyne, že počet k_1 činitelů je větší než počet k_0. Jestliže nyní platí $k_1 = p^n$, jsou všichni činitelé posledního rozkladu lineární a jsme hotovi, $T = T_1^*$. V opačném případě zvolíme polynom $b_j(x)$ stupně $\geqq 2$ a položíme

$$T_2^* = T_1^*/b_j(x) .$$

V tomto tělese má polynom $x^{p^n} - x$ rozklad na $k_2 > k_1$ ireducibilních polynomů, atd. Protože počet k_i činitelů rozkladu v tělese T_i^* nemůže růst nad číslo p^n, je zřejmé, že v některém tělese T_i^* má polynom $x^{p^n} - x$ rozklad na lineární činitele. Položíme $T^* = T_i^*$.

II. Těleso T: označme T množinu všech kořenů polynomu $x^{p^n} - x$ v tělese T^*. Tedy $T = \{a \in T^* \mid a^{p^n} = a\}$. Při algebraických operacích, definovaných jako v tělese T^*, je množina T tělesem — důkaz je analogický jako v 25.9.1. Těleso T má p^n prvků, protože polynom $x^{p^n} - x$ se rozkládá nad tělesem T^* v součin lineárních faktorů, které jsou navzájem různé. (Jinak by měl polynom $x^{p^n} - x$ vícenásobný kořen v tělese T, viz 23.6.2, ale to není možné podle 23.6.3: derivace $(x^{p^n} - x)' = p^n x^{p^n - 1} - 1 = -1$ nemá žádný kořen.)

28. Generující kořeny cyklického kódu

Významný způsob určení cyklického kódu je stanovení společných kořenů všech polynomů tohoto kódu. Kořeny leží v určitém rozšíření kódové abecedy. Ukážeme, že nezáleží na tom, ve kterém rozšíření pracujeme a dále stanovíme, jak se z generujících kořenů vytvoří generující polynom a kontrolní matice.

28.1. Binární kódy. Probereme nejprve případ binárního cyklického kódu. Kódovou abecedou je tedy těleso Z_2. Každé těleso T charakteristiky 2 (tj. takové, že platí $1 + 1 = 0$) je rozšířením kódové abecedy: prvky 0 a 1 tělesa T tvoří těleso Z_2.

Chceme stručně charakterizovat binární cyklické kódy délky n. Důležitý je předpoklad, že číslo n je liché. Potom můžeme najít rozšíření T abecedy Z_2 s prvkem α řádu n (25.8.3). V tělese T rozložíme polynom $x^n - 1$ na lineární činitele:

$$x^n - 1 = (x - 1)(x - \alpha)(x - \alpha^2) \dots (x - \alpha^{n-1}).$$

Přitom jsou mocniny $1, \alpha, \dots, \alpha^{n-1}$ navzájem různé (24.1). Těleso T najdeme jako $T = GF(2^m)$, kde m je (nejmenší) číslo takové, že n je dělitelem čísla $2^m - 1$. Toto číslo m lze najít mezi děliteli čísla $\varphi(n)$, viz 25.8. Příklady:

n	3	5	7	9	11	13	15	17	19
m	2	4	3	6	10	11	4	8	18.

Například v tělese $GF(8)$ zvolme $\alpha = z$ (viz tabulku A1). Protože má prvek α řád 7, platí

(28.1.1) $$x^7 - 1 = (x - 1)(x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)(x - \alpha^5)(x - \alpha^6).$$

Každý binární cyklický kód K délky n je charakterizován svým generujícím polynomem $g(x)$, který je dělitelem polynomu $x^n - 1$ (21.4). V rozšířeném tělese T má tedy i polynom $g(x)$ rozklad

$$g(x) = (x - \alpha^{i_1})(x - \alpha^{i_2}) \dots (x - \alpha^{i_s}).$$

Potom pro každý polynom $v(z)$ okruhu $Z_2^{(n)}$ platí: $v(z)$ je kódové slovo, právě když má kořeny $\alpha^{i_1}, \alpha^{i_2}, \dots, \alpha^{i_s}$. Stručněji:

(28.1.2) $$v(z) \in K \Leftrightarrow v(\alpha^{i_k}) = 0 \quad \text{pro} \quad k = 1, 2, \dots, s.$$

Každé kódové slovo je totiž tvaru $v(z) = q(z)\, g(z)$ (21.4), a tedy $v(\alpha^{i_k}) = q(\alpha^{i_k}) \cdot 0 = 0$. Naopak, jestliže platí $v(\alpha^{i_k}) = 0$, potom je polynom $v(x)$ dělitelný polynomem $g(x)$ (23.3), a tedy $v(z) \in K$. Kód K je tedy úplně charakterizován kořeny

$$\alpha^{i_1}, \alpha^{i_2}, \dots, \alpha^{i_s}.$$

Často není třeba znát všechny kořeny polynomu $g(x)$, stačí jen některé.

28.1.2. Definice. Říkáme, že binární cyklický kód K délky n má *generující kořeny* $\alpha^{i_1}, \alpha^{i_2}, \dots, \alpha^{i_s}$, jestliže α je prvek řádu n (v některém rozšíření tělesa Z_2) a pro každý polynom $v(z)$ v okruhu $Z_2^{(n)}$ platí vztah 28.1.2.

28.2. Příklady

28.2.1. Hammingův (7, 4)-kód (21.7.3). Tento kód má délku 7 a generující polynom $g(x) = x^3 + x + 1$.

V tělese $GF(8)$ máme rozklad (28.1.1). Přitom prvek $\alpha(= z)$ je kořenem polynomu $g(x)$. Odtud plyne, že také α^2 a α^4 jsou jeho kořeny (25.7), a tedy

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^4) \quad \text{nad} \quad GF(8).$$

Vidíme, že α, α^2 a α^4 jsou generující kořeny Hammingova kódu. Také sám kořen α je generující kořen: má-li polynom $v(z)$ v $Z_2^{(7)}$ kořen α, má i kořeny α^2 a α^4, a tedy je dělitelný polynomem $g(z)$ (23.3).

28.2.2. Kód celkové kontroly parity (4.6). Jeho jediný generující kořen je 1. Polynom $v(z)$ má totiž kořen 1, právě když $v_0 + v_1 + \ldots + v_{n-1} = 1$, tj. právě když má slovo $v_0 v_1 \ldots v_{n-1}$ sudou paritu.

28.2.3. Určíme binární cyklický kód délky 15 s generujícími kořeny α, α^2 a α^5 (kde α je prvek řádu 15). Zvolíme rozšíření $GF(16)$ kódové abecedy Z_2 (viz dodatek A1): primitivní prvek $\alpha = z$ je řádu 15. Každý polynom nad tělesem Z_2 s kořenem α je, podle 26.3, dělitelný minimálním polynomem prvku α, a to je polynom

$$M_1(x) = x^4 + x + 1\,,$$

viz 26.12. Kořen α^2 není důležitý (protože každý polynom s kořenem α má i kořen α^2, viz 25.7). Minimální polynom prvku α^5 je

$$M_5(x) = x^2 + x + 1\,.$$

Každé kódové slovo je polynom $v(z)$ v $Z_2^{(15)}$, dělitelný polynomy $M_1(x)$ a $M_5(x)$, a tedy dělitelný jejich součinem

$$g(x) = M_1(x)\, M_5(x)\,,$$

viz 23.14.4 a 26.3. Polynom $g(x)$ je generujícím polynomem našeho kódu: protože má kořeny α, α^2 a α^5, je kódovým slovem a každé kódové slovo je násobkem polynomu $g(x)$. Generující kořeny tedy zadaly cyklický (15, 9)-kód s generujícím polynomem

$$g(x) = (x^4 + x + 1)(x^2 + x + 1)\,.$$

28.3. Generující polynom. Jestliže známe generující kořeny kódu, je generující polynom součinem minimálních polynomů generujících kořenů (jak jsme to viděli v předchozím příkladě). Podrobněji, buď K binární cyklický kód délky n s generujícími kořeny $\alpha^{i_1}, \alpha^{i_2}, \ldots, \alpha^{i_s}$. Označme $M_i(x)$ minimální polynom prvku α^i.

a) Jestliže jsou minimální polynomy $M_{i_1}(x), \ldots, M_{i_s}(x)$ po dvou různé, má kód K generující polynom

$$g(x) = M_{i_1}(x)\, M_{i_2}(x) \ldots M_{i_s}(x)\,.$$

Je totiž zřejmé, že polynom $g(x)$ má kořeny $\alpha^{i_1}, \ldots, \alpha^{i_s}$, takže je kódovým slovem. Přitom každé kódové slovo je dělitelné polynomem $g(x)$ (viz 23.14.5 a 26.3).

b) Jestliže prvky α^{i_p} a α^{i_q} mají stejný minimální polynom, můžeme jeden z nich vyškrtnout a opět zbydou generující kořeny kódu K.

Každý polynom s kořenem α^{i_p} má totiž i kořen α^{i_q} (a naopak).

28.3.1. Příklad. Hammingovy kódy (14.4) jsou binární cyklické kódy délky $2^m - 1$ s jediným generujícím kořenem α. Jestliže totiž zvolíme primitivní prvek α v Galoisově tělese $GF(2^m)$, pak minimální polynom prvku α je polynom $q(x)$ stupně m (viz 26.11). Ten je generujícím polynomem kódu, takže jde o kód s m kontrolními znaky, tj. $(2^m - 1, 2^m - m - 1)$-kód a ten je Hammingovým kódem, viz 14.13, protože neobsahuje slova váhy 1 nebo 2. (Slovo, $v(z) = z^i + z^j$ váhy 2 nesplňuje $v(\alpha) = 0$, protože prvek α je řádu $2^m - 1$, a tedy $\alpha^i \neq \alpha^j$, viz 24.1.)

28.4. Kontrolní matice. Jestliže pracujeme v rozšíření $T = GF(2^m)$ kódové abecedy Z_2, potom prvky tohoto rozšíření jsou polynomy $a(z)$ stupně $< m$ nad tělesem Z_2, které můžeme chápat jako sloupce jejich koeficientů:

$$\begin{bmatrix} a_{m-1} \\ a_{m-2} \\ \vdots \\ a_1 \\ a_0 \end{bmatrix}.$$

V tomto smyslu se každá matice o k řádcích nad tělesem T stává binární maticí o km řádcích. Například nad tělesem $GF(8)$ volíme $\alpha = z$ jako v 28.2.1 a vytvoříme jednořádkovou matici

$$[1 \ \alpha \ \alpha^2 \ \alpha^3 \ \alpha^4 \ \alpha^5 \ \alpha^6].$$

S použitím tabulky tělesa $GF(8)$ (viz A1) můžeme přepsat tuto matici do binárního tvaru. Například $\alpha = z$ je tento sloupec

$$\alpha = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

a $\alpha^3 = 1 + z$ je tento sloupec

$$\alpha^3 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

atd. Vzniká tedy tato binární matice:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

To je kontrolní matice Hammingova (7, 4)-kódu, viz 21.7.3. Obecně:

28.4.1. Tvrzení. Binární cyklický kód délky n s generujícími kořeny $\alpha^{i_1}, \alpha^{i_2}, \ldots, \alpha^{i_s}$ má tuto kontrolní matici:

$$(28.4.1) \qquad \boldsymbol{H} = \begin{bmatrix} 1 & \alpha^{i_1} & (\alpha^{i_1})^2 & (\alpha^{i_1})^3 & \ldots & (\alpha^{i_1})^{n-1} \\ 1 & \alpha^{i_2} & (\alpha^{i_2})^2 & (\alpha^{i_2})^3 & \ldots & (\alpha^{i_2})^{n-1} \\ & & \ldots\ldots\ldots\ldots\ldots\ldots & & & \\ 1 & \alpha^{i_s} & (\alpha^{i_s})^2 & (\alpha^{i_s})^3 & \ldots & (\alpha^{i_s})^{n-1} \end{bmatrix}.$$

Důkaz. Pro každý polynom $v(z)$ v $Z_2^{(n)}$ platí

$$\boldsymbol{H}\begin{bmatrix} v_0 \\ v_1 \\ \dots \\ v_{n-1} \end{bmatrix} = \begin{bmatrix} v_0 + v_1\alpha^{i_1} + v_2(\alpha^{i_1})^2 + \dots + v_{n-1}(\alpha^{i_1})^{n-1} \\ v_0 + v_1\alpha^{i_2} + v_2(\alpha^{i_2})^2 + \dots + v_{n-1}(\alpha^{i_2})^{n-1} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ v_0 + v_1\alpha^{i_s} + v_2(\alpha^{i_s})^2 + \dots + v_{n-1}(\alpha^{i_s})^{n-1} \end{bmatrix} = \begin{bmatrix} v(\alpha^{i_1}) \\ v(\alpha^{i_2}) \\ \dots \\ v(\alpha^{i_s}) \end{bmatrix}.$$

Vidíme, že kódová slova jsou právě ta, pro která $\boldsymbol{H}\mathbf{v}^{\mathrm{T}}$ je nulový sloupec. To znamená, že $\boldsymbol{H}$ je kontrolní matice.

28.4.2. Příklad. Kód z příkladu 28.2.3 má tuto kontrolní matici

$$\boldsymbol{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha^9 & \alpha^{10} & \alpha^{11} & \alpha^{12} & \alpha^{13} & \alpha^{14} \\ 1 & \alpha^5 & \alpha^{10} & 1 & \alpha^5 & \alpha^{10} & 1 & \alpha^5 & \alpha^{10} & 1 & \alpha^5 & \alpha^{10} & 1 & \alpha^5 & \alpha^{10} \end{bmatrix}$$

zapsanou v tělese $GF(16)$. S použitím tabulky tělesa (viz A1), kde $\alpha = z$, dostáváme binární tvar této matice

$$\boldsymbol{H} = \begin{bmatrix} 0&0&0&1&0&0&1&1&0&1&0&1&1&1&1 \\ 0&0&1&0&0&1&1&0&1&0&1&1&1&1&0 \\ 0&1&0&0&1&1&0&1&0&1&1&1&1&0&0 \\ 1&0&0&0&1&0&0&1&1&0&1&0&1&1&1 \\ \dots&\dots&\dots&\dots&\dots&\dots&\dots&\dots&\dots&\dots&\dots&\dots&\dots&\dots&\dots \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \\ 0&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \\ 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \end{bmatrix}.$$

Z této matice můžeme jistě vynechat pátý a šestý řádek. Tím dostaneme matici typu (6, 15) a to odpovídá šesti kontrolním znakům kódu.

28.5. Obecné kódy. Obecně pracujeme s q-znakovým cyklickým kódem, kde q je mocnina prvočísla. Budeme předpokládat, že délka kódu n je nesoudělná s číslem q (pro jiné cyklické kódy generující kořeny nezavádíme). Najdeme rozšíření $T = GF(q^m)$ kódové abecedy s prvkem α řádu n, a tedy s rozkladem

$$x^n - 1 = (x - 1)(x - \alpha^2)\dots(x - \alpha^{n-1}),$$

viz 25.8.3. (Mocnitele m můžeme vždy najít mezi děliteli čísla $\varphi(n)$.)

28.6. Definice. *Generujícími kořeny* cyklického kódu K délky n v abecedě T nazýváme prvky $\alpha^{i_1}, \alpha^{i_2}, \dots, \alpha^{i_s}$ takové, že α je prvek n-tého řádu v některém rozšíření abecedy T a pro každý polynom $v(z)$ v $T^{(n)}$ platí:

$v(z)$ je kódové slovo, právě když $v(\alpha^i) = 0$ pro všechna $i = i_1, i_2, \dots, i_s$.

28.7. Poznámka. Analogicky jako v binárním případě je generující polynom součinem všech navzájem různých minimálních polynomů generujících kořenů. Zde ovšem minimálním polynomem se rozumí polynom nad tělesem $T_0 = GF(q)$, viz 26.14.

28.8. Příklady

28.8.1. Určíme generující kořeny ternárního cyklického kódu délky 8 s generujícím polynomem $g(x) = x^3 + x^2 + 2x + 2$. V tělese $T = GF(9)$ má prvek $\alpha = z$ řád 8, viz A1. Snadno ověříte, že 1 a α jsou kořeny polynomu $g(x)$, takže i α^3 je kořenem, viz 25.7. Protože je $g(x)$ polynomem třetího stupně, známe všechny jeho kořeny nad tělesem $GF(9)$:

$$g(x) = (x - 1)(x - \alpha)(x - \alpha^3).$$

(O této rovnosti se můžete přesvědčit roznásobením.)

Vidíme, že 1 a α jsou generující kořeny daného kódu. Každý polynom s kořeny 1 a α má i kořen α^3, a je tedy násobkem polynomu $g(x)$.

28.8.2. Určíme 4-znakový cyklický kód délky 5 s generujícím kořenem α. Zde kódová abeceda je $T_0 = GF(4)$. Těleso $T = GF(4^2)$ je jejím rozšířením (25.9). Primitivní prvek z tělesa $GF(4^2)$, uvedeného v dodatku A1, je řádu 15. Položíme $t = z^5$ a dostaneme čtyři prvky

$$0,\ 1,\ t \quad \text{a} \quad s = 1 + t = z^{10},$$

které tvoří těleso $GF(4)$. Navíc má prvek $\alpha = z^3$ řád 5.

Máme tedy kódovou abecedu $T_0 = \{0, 1, t, s\}$ a její rozšíření $T = GF(16)$ s prvkem α řádu 5. Všechna slova našeho kódu mají kořen α (a tedy i kořeny α^2, α^4 a α^8, viz 25.7). Jsou tedy dělitelná polynomem

$$(x - \alpha)(x - \alpha^2)(x - x^4)(x - \alpha^8) = x^4 + x^3 + x^2 + x + 1.$$

Poslední rovnost ověříte roznásobením. Vidíme, že generující polynom $g(x) = x^4 + x^3 + x^2 + x + 1$ je stupně 4, takže jde o opakovací kód (s jediným informačním znakem).

28.9. Kontrolní matice. I v obecném případě je kontrolní matice dána vzorcem (28.4.1), ve kterém prvky α^i tělesa $T = GF(q^m)$ zase chápeme jako sloupce délky m nad tělesem $T_0 = GF(q)$. Důkaz je úplně analogický případu $q = 2$.

Dosud jsme se nezabývali otázkou, jak důležitá je volba konkrétního rozšíření kódové abecedy a konkrétního prvku α pro vlastnosti cyklického kódu. Ukazuje se, že vůbec ne:

28.10. Věta o nezávislosti. *Cyklický kód K, určený generujícími kořeny $\alpha^{i_1}, \alpha^{i_2}, \ldots, \alpha^{i_s}$, je až na ekvivalenci nezávislý na výběru prvku α (řádu n v rozšíření kódové abecedy). Je-li tedy β prvek řádu n v (jiném) rozšíření, potom je kód K ekvivalentní s cyklickým kódem, určeným generujícími kořeny $\beta^{i_1}, \beta^{i_2}, \ldots, \beta^{i_s}$.*

Důkaz. Jestliže je kódová abeceda T_0 tělesem charakteristiky p, potom každé její rozšíření je Galoisovo těleso $GF(p^i)$ pro některé číslo i, viz 26.9. Prvek α leží v určitém rozšíření $GF(p^i)$ a prvek β leží v rozšíření $GF(p^j)$. Těleso $T = GF(p^{ij})$ je společným rozšířením obou těchto těles (25.9.1), takže můžeme předpokládat, že α i β leží ve společném rozšíření T kódové abecedy T_0.

V tělese T máme rozklad $x^n - 1 = (x - 1)(x - \alpha) \dots (x - \alpha^{n-1})$, viz 24.3, takže všechny kořeny polynomu $x^n - 1$ jsou tvaru α^t. Protože $\beta^n - 1 = 0$, platí tedy $\beta = \alpha^t$ pro některé číslo $t = 0, 1, \dots, n - 1$. Položme $\pi_i = it$, kde součin provádíme v okruhu Z_n, pro $i = 0, 1, \dots, n - 1$. Čísla $\pi_0, \pi_1, \dots, \pi_{n-1}$ jsou navzájem různá. Jestliže totiž platí $\pi_i = \pi_j$, tj. $it \equiv jt \pmod{n}$, existuje číslo r takové, že $it = jt + rn$, a potom

$$\beta^i = \alpha^{it} = \alpha^{jt}\alpha^{rn} = \alpha^{jt}(\alpha^n)^r = \alpha^{jt} = \beta^j .$$

Odtud plyne $i = j$, viz 24.1. To znamená, že jsme vytvořili permutaci $\pi = [\pi_0, \pi_1, \dots, \pi_{n-1}]$ čísel $0, 1, \dots, n - 1$. Platí $\alpha^{\pi_i} = \beta^i$.

Kód K je ekvivalentní s kódem K', zadaným generujícími kořeny $\beta^{i_1}, \beta^{i_2}, \dots, \beta^{i_s}$. Platí totiž

$$v_0 v_1 \dots v_{n-1} \in K , \quad \text{právě když} \quad v_{\pi_0} v_{\pi_1} \dots v_{\pi_{n-1}} \in K' .$$

Skutečně, polynom $v(z) = \sum v_j z^j$ je kódovým slovem kódu K, právě když splňuje podmínku

$$v(\alpha^{i_t}) = \sum_{j=0}^{n-1} v_j (\alpha^j)^{i_t} = 0 \quad (t = 1, 2, \dots, s) .$$

Místo sčítacího indexu j můžeme použít index π_j. Protože $\alpha^{\pi_j} = \beta^j$, dostáváme ekvivalentní podmínku

$$\sum_{j=0}^{n-1} v_{\pi_j} (\beta^j)^{i_t} = 0 \quad (t = 1, 2, \dots, s) .$$

Vidíme, že polynom s koeficienty $v_0 v_1 \dots v_{n-1}$ má kořeny α^{i_t}, právě když má polynom s koeficienty $v_{\pi_0} v_{\pi_1} \dots v_{\pi_{n-1}}$ kořeny β^{i_t}. To jsme měli dokázat.

VII. BCH KÓDY—OBECNÉ KÓDY S DOBRÝMI PARAMETRY

V této kapitole se věnujeme nejdůležitější třídě bezpečnostních kódů. Tyto kódy objevili koncem padesátých let R. C. Bose a D. K. Ray-Chaudhuri a nezávisle na nich A. Hocquenghem; podle iniciál autorů se nazývají BCH kódy. Mezi jejich významné vlastnosti patří velká volitelnost parametrů, dobrý vztah mezi počtem informačních znaků a počtem opravovaných chyb a detailně vypracované dekódovací metody.

Uvedeme nejdříve binární BCH kódy opravující dvojnásobné chyby. Dále se budeme zabývat obecnými binárními BCH kódy, včetně maticové metody dekódování. Je to snadno pochopitelná metoda, která však není praktická při vyšším počtu plánovaných oprav. Proto uvádíme (i pro víceznakové BCH kódy) další metodu, založenou na Euklidově algoritmu.

Ve srovnání s Reedovými-Mullerovými kódy mají BCH kódy o něco náročnější dekódování, zato ale mají lepší parametry. To se projevuje nevýrazně při malých délkách (viz dodatek A2), ale např. BCH kódy délky 255, opravující 31 chyb, mají 55 informačních znaků, zatímco odpovídající Reedův-Mullerův kód má jen 37 informačních znaků.

29. BCH kód délky 15

29.1. Pro opravy jednoduchých chyb jsme vybaveni perfektními Hammingovými kódy (14.13) s jedním generujícím kořenem (28.3.1). Například Hammingův (15, 11)-kód je dán svou kontrolní maticí

$$\boldsymbol{H} = [1 \ \alpha \ \alpha^2 \ \ldots \ \alpha^{14}],$$

kde α je primitivní prvek tělesa $GF(16)$, viz 28.4.

BCH kódy (binární) pro dvojnásobné opravy jsou zobecněním Hammingových kódů: zatímco pro jednu chybu má matice $\boldsymbol{H}$ jeden řádek (nad tělesem $GF(16)$), pro dvě chyby má dva řádky:

$$\boldsymbol{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \ldots & \alpha^{14} \\ 1 & \alpha^3 & (\alpha^3)^2 & \ldots & (\alpha^3)^{14} \end{bmatrix}. \tag{29.1.1}$$

(S použitím tabulky tělesa $GF(16)$ v dodatku A1 bychom mohli přepsat tuto matici do binárního tvaru s osmi řádky.)

Ukážeme, jak kód s touto kontrolní maticí opravuje dvojnásobné chyby. Přijali jsme slovo

$$\boldsymbol{w} = w_0 w_1 \ldots w_{14},$$

které se od vyslaného slova liší na i-tém a j-tém místě. Chybové slovo je tedy

$$\mathbf{e} = 00\ldots010\ldots010\ldots00,$$

nebo ve tvaru polynomu

$$e(z) = z^i + z^j.$$

Známe syndrom

$$\begin{pmatrix} s_1 \\ s_3 \end{pmatrix} = \boldsymbol{H}\boldsymbol{w}^{\mathrm{T}} = \boldsymbol{H}\boldsymbol{e}^{\mathrm{T}} = \begin{pmatrix} \alpha^i + \alpha^j \\ \alpha^{3i} + \alpha^{3j} \end{pmatrix}.$$

Chceme tedy určit čísla i a j taková, že platí

$$s_1 = \alpha^i + \alpha^j,$$
$$s_3 = \alpha^{3i} + \alpha^{3j}.$$

Z první rovnice dostáváme

$$\begin{aligned} s_1^3 &= (\alpha^i + \alpha^j)^3 = \\ &= \alpha^{3i} + \alpha^{2i}\alpha^j + \alpha^i\alpha^{2j} + \alpha^{3j} = \\ &= s_3 + \alpha^i\alpha^j(\alpha^i + \alpha^j) = \\ &= s_3 + s_1\alpha^i\alpha^j. \end{aligned}$$

Známe tedy součet

$$\alpha^i + \alpha^j = s_1$$

i součin

$$\alpha^i\alpha^j = \frac{s_1^3 - s_3}{s_1} = s_1^2 + s_3 s_1^{-1}$$

neznámých veličin α^i a α^j. Tyto veličiny jsou potom kořeny kvadratické rovnice

(29.1.2) $$\lambda^2 + s_1\lambda + (s_1^2 + s_3 s_1^{-1}) = 0.$$

Po sestavení této rovnice dosazujeme postupně $\lambda = 1, \alpha, \alpha^2, \ldots$. Tím získáme kořeny α^i a α^j a opravíme tedy w_i a w_j. (Obvyklý vzorec pro řešení kvadratických rovnic nad konečným tělesem neplatí!)

29.1.1. Úloha. Ověřte, že kvadratická rovnice (29.1.2) určuje opravy i v případě jediné chyby: opravíme právě to místo w_i, pro které je prvek α^i kořenem rovnice. Návod: pro $e(z) = z^i$ má rovnice tvar $\lambda^2 + \alpha^i\lambda = 0$.

29.2. Příklad. Přijali jsme slovo

$$\boldsymbol{w} = 000000111000000 = z^6 + z^7 + z^8.$$

Jeho syndrom (s použitím tabulky tělesa $GF(16)$ v dodatku A1, kde $\alpha = z$) je

$$\begin{bmatrix} s_1 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1 & \alpha & \alpha^3 & \ldots & \alpha^{15} \\ 1 & \alpha^3 & \alpha^6 & \ldots & \alpha^{45} \end{bmatrix} \boldsymbol{w}^{\mathrm{T}} = \begin{bmatrix} \alpha^6 + \alpha^7 + \alpha^8 \\ \alpha^{18} + \alpha^{21} + \alpha^{24} \end{bmatrix} = \begin{bmatrix} \alpha \\ \alpha^{11} \end{bmatrix}.$$

Odtud

$$s_1 = \alpha,$$
$$s_3 = \alpha^{11}$$

a

$$s_1^2 + s_3 s_1^{-1} = \alpha^2 + \alpha^{10} = \alpha^4.$$

Dostáváme kvadratickou rovnici

$$\lambda^2 + \lambda\alpha + \alpha^4 = 0 .$$

Její kořeny najdeme postupným dosazováním za λ:

$$\lambda_1 = 1 = \alpha^0 \quad \text{a} \quad \lambda_2 = \alpha^4 .$$

Opravíme místa w_0 a w_4. Vysláno bylo slovo

$$\mathbf{v} = 100010111000000 .$$

(Zkouška: syndrom slova $\mathbf{v}$ je nulový, takže $\mathbf{v}$ je kódové slovo. Od přijatého slova má Hammingovu vzdálenost 2. Jak uvidíme v dalším článku, má náš kód minimální vzdálenost 5, takže nalezené slovo $\mathbf{v}$ je jediné kódové slovo, které se od přijatého slova $\mathbf{w}$ liší jen ve dvou znacích.)

29.2.1. Úloha. Při přijetí slova $1 + z + z^5 + z^{14}$ ověřte, že rovnice (29.1.2) nemá řešení. Odvoďte, že došlo alespoň ke třem chybám.

29.3. Generující polynom uvažovaného kódu najdeme podle 28.3: minimální polynom prvku $\alpha(=z)$ je podle 26.12

$$M_1(x) = x^4 + x + 1$$

a minimální polynom prvku α^3 je

$$M_3(x) = (x - \alpha^3)(x - \alpha^6)(x - \alpha^9)(x - \alpha^{12}) = x^4 + x^3 + x^2 + x + 1 .$$

Odtud

$$g(x) = M_1(x) M_3(x) = 1 + x^4 + x^6 + x^7 + x^8 .$$

Kód má tedy 8 kontrolních znaků a je (15, 7)-kódem. Nyní definujeme BCH kódy pro dvojnásobné opravy obecně.

30. BCH kódy pro dvojnásobné opravy

30.1. Definice. *Binárním* BCH *kódem pro dvojnásobné opravy* se nazývá kód liché délky $n \geqq 5$ s generujícími kořeny

$$\alpha \quad \text{a} \quad \alpha^3$$

(kde α je prvek řádu n v některém tělese $GF(2^m)$, viz 28.1.2).

30.2. Poznámka. BCH kód je tedy určen kontrolní maticí

$$\boldsymbol{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{n-1} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \dots & \alpha^{3(n-1)} \end{bmatrix},$$

viz 28.4. V případě délky $n = 2^m - 1$ můžeme volit za α primitivní prvek tělesa $GF(2^m)$. Ukážeme v příštím odstavci, že počet kontrolních znaků (tj. stupeň polynomu $g(x)$) je pak roven $2m$ pro všechna $m = 3, 4, 5, \dots$. Odtud podle 28.3 plyne, že generující polynom je součinem minimálních polynomů prvků α a α^3,

$$g(x) = M_1(x) M_3(x) .$$

30.3. Dekódování. Přijali jsme slovo $\boldsymbol{w}$ a předpokládáme, že došlo nejvýše k dvěma chybám. Vypočteme syndrom

$$\begin{bmatrix} s_1 \\ s_3 \end{bmatrix} = \boldsymbol{H}\boldsymbol{w}^{\mathrm{T}} = \begin{bmatrix} w(\alpha) \\ w(\alpha^3) \end{bmatrix}.$$

a) Jestliže $s_1 = s_2 = 0$, bylo vysláno slovo $\boldsymbol{w}$. Nulový syndrom mají totiž jen kódová slova.

b) Jestliže $s_1 \neq 0$, sestavíme kvadratickou rovnici

$$\lambda^2 + s_1\lambda + (s_1^2 + s_3 s_1^{-1}) = 0 . \tag{30.3.1}$$

Pokud má tato rovnice nenulové kořeny α^i a α^j, opravíme znaky w_i a w_j. (Protože $s_1 \neq 0$, nemá rovnice dvojnásobný kořen: platí $(\lambda - c)^2 = \lambda^2 + c^2$.)

Pokud má rovnice kořen 0, tj. pokud

$$s_3 = s_1^3 ,$$

došlo k jediné chybě. Opravíme w_i, kde $\alpha^i = s_i$. (Jinými slovy, najdeme kořeny 0 a α^i kvadratické rovnice a nulový kořen ignorujeme.)

c) Jestliže $s_1 = 0 \neq s_3$, není předpoklad o dvojnásobné chybě správný, tj. došlo alespoň ke třem chybám. Totéž platí v případě, že rovnice (30.3.1) nemá řešení.

Důkaz. Pokud došlo ke dvěma chybám, má chybový vektor tvar $e(z) = z^i + z^j$ a platí

$$\begin{bmatrix} s_1 \\ s_3 \end{bmatrix} = \boldsymbol{H}\boldsymbol{e}^{\mathrm{T}} = \begin{bmatrix} \alpha^i + \alpha^j \\ \alpha^{3i} + \alpha^{3j} \end{bmatrix}.$$

Odtud odvodíme, stejně jako v 29.1, že α^i a α^j jsou kořeny uvedené kvadratické rovnice. Protože $i \neq j$, platí ovšem $s_1 = \alpha^i + \alpha^j \neq 0$. Oba kořeny kvadratické rovnice jsou nenulové (a tedy $s_3 \neq s_1^3$).

Pokud došlo k jediné chybě, má chybový vektor tvar $e(z) = z^i$ a platí

$$\begin{bmatrix} s_1 \\ s_3 \end{bmatrix} = \begin{bmatrix} \alpha^i \\ \alpha^{3i} \end{bmatrix}.$$

To znamená, že $s_1 = \alpha^i$ a $s_3 = s_1^3$.

Ani v jednom z těchto případů není $s_1 = 0$, takže pokud $s_1 = 0 \neq s_3$, došlo alespoň ke třem chybám.

30.4. Příklad: BCH kód délky 9

Nejprve určíme těleso s prvkem řádu 9. Použijeme Eulerovu funkci (25.8): $\varphi(9) = 6$, takže $GF(2^6) = GF(64)$ je takové těleso. Skutečně, primitivní prvek a je řádu $63 = 7 \cdot 9$, takže prvek $\alpha = a^7$ je řádu 9.

Prvek α má minimální polynom $M_1(x)$ stupně 6. To plyne z 26.11 pomocí tabulky mocnin v tělese $GF(64)$:

α	α^2	α^4	α^8	α^{16}	α^{32}	$(\alpha^{64} = \alpha)$
a^7	a^{14}	a^{28}	a^{56}	$a^{112} = a^{49}$	$a^{98} = a^{35}$	$(y^{70} = a^7)$.

Prvek α^3 má minimální polynom $M_3(x)$ stupně 2:

$\alpha^3 \quad \alpha^6 \quad (\alpha^{12} = \alpha)$
$a^{21} \quad a^{42} \quad (a^{84} = a^{21})$.

Generující polynom BCH kódu,

$$g(x) = M_1(x)\, M_3(x) = (x - \alpha)(x - \alpha^2) \ldots (x - \alpha^{32})(x - \alpha^3)(x - \alpha^6),$$

je tedy stupně 8. Je dělitelem polynomu $x^9 - 1$ (21.4) a podíl $(x^9 - 1)$: $g(x)$ je ovšem polynom stupně 1. Je to tedy polynom $x + 1$ (protože jinak by to byl polynom x, který však polynom $x^9 - 1$ nedělí). Tedy

$$x^9 - 1 = g(x)(x - 1),$$

a odtud

$$g(x) = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8.$$

Generující matice je tedy

$$\boldsymbol{G} = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1].$$

Závěr: BCH kód pro dvojnásobné chyby délky 9 je opakovací kód.

30.5. Úlohy

30.5.1. Ověřte, že BCH kód pro dvojnásobné opravy délky 7 je opakovací kód. Návod: lépe než postupovat jako v 30.4 je použít tělesa $GF(8)$.

30.5.2. Při použití BCH kódu délky 15 dekódujte $w(z) = z^{11} + z^{14}$.

30.5.3. Určete generující polynom BCH kódu pro dvojnásobné opravy délky 31.

30.5.4. V jakém tělese leží generující kořeny BCH kódu délky 11?

30.6. Závěr

Zobecněním perfektních Hammingových kódů, které opravují jednoduché chyby a mají kontrolní matici s jedním řádkem (nad tělesem $GF(2^m)$), jsou BCH kódy pro dvojnásobné opravy, které mají kontrolní matici se dvěma řádky.

Pro tyto kódy máme rychlé dekódování, založené na řešení kvadratické rovnice (30.3.1). Tímto způsobem opravíme všechny dvojnásobné chyby. Protože však BCH kódy nejsou perfektní, jde pouze o částečné dekódování: BCH kódy jsou schopny opravit i některé trojnásobné chyby, které již nevystihne uvažovaná dekódovací metoda. Například BCH kód délky 15 má 10 kontrolních znaků, takže při standardním (pomalém) dekódování (13.3) je schopen opravit $2^{10} = 1024$ chybových slov. Z toho počtu jsou chybová slova váhy $\leqq 2$ jen malou částí: je jich $1 + 15 + \binom{15}{2} = 121$. (A BCH kód délky 9 je opakovací kód (30.4), který dokonce opravuje čtyřnásobné chyby!) Úplný dekódovací algoritmus je uveden v Berlekampově monografii [5].

V následujících článcích se budeme zabývat BCH kódy pro t-násobné opravy. Vždy se přitom omezíme na částečné dekódování, které opraví t-násobné chyby, ale nevšímá si dalších opravitelných chyb.

31. Binární BCH kódy

Definujeme nyní obecné BCH kódy s plánovanou vzdáleností d a ukážeme, že minimální vzdálenost je alespoň d. To znamená, že pro $d = 2t + 1$ kódy opravují t-násobné chyby. Případ $d = 5$ jsme probrali v minulém článku. Ukážeme, jaké mají BCH kódy vlastnosti, kromě dekódování, kterému věnujeme další články. Obecnější definici BCH kódů (i pro víceznakové abecedy) uvedeme později.

31.1. Definice. *Binární* BCH *kód s plánovanou vzdáleností* $d = 2, 3, 4, \ldots$ je kód liché délky $n \geqq d$ s generujícími kořeny

$$\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{d-1}$$

(kde α je prvek n-tého řádu v tělese $GF(2^m)$, viz 28.1.2).

31.2. Poznámky

31.2.1. Sudé mocniny mezi generujícími kořeny jsou zbytečné: polynom $v(z) \in Z_2^{(n)}$ má s každým kořenem α^i také kořeny $\alpha^{2i}, \alpha^{4i}, \ldots$ (25.7). Odtud plyne, že BCH kódy s plánovanou vzdáleností $d = 2t$ nebo $d = 2t + 1$ jsou totožné a mají generující kořeny

$$\alpha, \alpha^3, \alpha^5, \ldots, \alpha^{2t-1}.$$

Proto budeme dále vždy předpokládat, že plánovaná vzdálenost d je liché číslo.

31.2.2. BCH kód s plánovanou vzdáleností d lze také definovat jeho kontrolní maticí (28.4):

$$\boldsymbol{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \ldots & \alpha^{n-1} \\ 1 & \alpha^3 & (\alpha^3)^2 & (\alpha^3)^3 & \ldots & (\alpha^3)^{n-1} \\ 1 & \alpha^5 & (\alpha^5)^2 & (\alpha^5)^3 & \ldots & (\alpha^5)^{n-1} \\ \multicolumn{6}{c}{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} \\ 1 & \alpha^{d-2} & (\alpha^{d-2})^2 & (\alpha^{d-2})^3 & \ldots & (\alpha^{d-2})^{n-1} \end{bmatrix}.$$

31.2.3. Počet kontrolních znaků BCH kódu je číslo

(33.2.1) $\qquad n - k \leqq \frac{1}{2}m(d-1)$.

Kontrolní matice má totiž $\frac{1}{2}(d-1)$ řádků nad tělesem $GF(2^m)$, a tedy $\frac{1}{2}m(d-1)$ řádků nad tělesem Z_2. Jestliže jsou tyto řádky lineárně nezávislé, platí $n - k = \frac{1}{2}m(d-1)$. Jestliže jsou závislé, je $n - k$ menší.

31.3. Příklady

31.3.1. $d = 3$. Jde o binární kód s generujícím kořenem α. Pokud $n = 2^m - 1$, je α primitivní prvek tělesa $GF(2^m)$, takže BCH kód je Hammingův $(2^m - 1, 2^m - m - 1)$-kód, viz 28.3.1. Nerovnost (31.2.1) je zde rovností: $n - k = m$.

31.3.2. $d = 5$. Jde o BCH kódy pro dvojnásobné opravy, které jsme probírali v minulém článku. Ukážeme později, že pro $n = 2^m - 1$ je nerovnost (31.2.1) rovností.

31.3.3. $d = 7$. Určíme BCH kód plánované vzdálenosti 7 a délky 15. Ten má generující kořeny

$$\alpha, \alpha^3 \text{ a } \alpha^5,$$

kde α je primitivní prvek tělesa $GF(16)$. Platí

$$M_1(x) = x^4 + x + 1,$$
$$M_3(x) = x^4 + x^3 + x^2 + x + 1$$

a

$$M_5(x) = x^2 + x + 1$$

(viz 26.12). Generující polynom je tedy

$$g(x) = M_1(x)\, M_3(x)\, M_5(x) =$$
$$= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1).$$

Jde o kód s 10 kontrolními znaky (takže zde nerovnost (31.2.1) není rovností). Tento (15, 5)-kód opravuje trojnásobné chyby, jak plyne z následující věty.

31.4. Lemma (Vandermondův determinant). V každém tělese pro libovolné prvky $a_1, a_2, \ldots, a_n$ platí

$$\det \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ a_1 & a_2 & a_3 & \ldots & a_n \\ a_1^2 & a_2^2 & a_3^2 & \ldots & a_n^2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_1^{n-1} & a_2^{n-1} & a_3^{n-1} & \ldots & a_n^{n-1} \end{bmatrix} = \prod_{i>j} (a_i - a_j).$$

Tento determinant, nazývaný Vandermondův, je tedy součinem všech rozdílů prvků $a_i - a_j$, kde $1 \leqq i < j \leqq n$. Odtud plyne, že determinant je nenulový, právě když jsou prvky $a_1, a_2, \ldots, a_n$ po dvou různé.

Důkaz. Pro $n = 2$ je tvrzení zřejmé:

$$\det \begin{bmatrix} 1 & 1 \\ a_1 & a_2 \end{bmatrix} = a_2 - a_1 .$$

Tento determinant je nenulový, právě když $a_1 \neq a_2$. Pro vyšší n lze snadno převést Vandermondův determinant na nižší řád. Ukážeme to v případě $n = 3$ (postup pro $n = 4, 5, \ldots$ je analogický). Provedeme úpravu matice tak, že první řádek opíšeme

a od každého nižšího řádku odečteme a_1-násobek předchozího řádku:

$$\det\begin{bmatrix}1 & 1 & 1\\ a_1 & a_2 & a_3\\ a_1^2 & a_2^2 & a_3^2\end{bmatrix} = \det\begin{bmatrix}1 & 1 & 1\\ 0 & a_2 - a_1 & a_3 - a_1\\ 0 & a_2^2 - a_1 a_2 & a_3^2 - a_1 a_3\end{bmatrix} =$$

$$= \det\begin{bmatrix}a_2 - a_1 & a_3 - a_1\\ a_2(a_2 - a_1) & a_3(a_3 - a_1)\end{bmatrix}.$$

Vytkneme $a_2 - a_1$ z prvního sloupce a $a_3 - a_1$ z druhého:

$$\det\begin{bmatrix}1 & 1 & 1\\ a_1 & a_2 & a_3\\ a_1^2 & a_2^2 & a_3^2\end{bmatrix} = (a_2 - a_1)(a_3 - a_1)\det\begin{bmatrix}1 & 1\\ a_2 & a_3\end{bmatrix}.$$

Poslední determinant je Vandermondův determinant řádu 2, takže determinant, který počítáme, je roven

$$(a_2 - a_1)(a_3 - a_1)(a_3 - a_2).$$

Ten je nenulový, právě když prvky a_1, a_2 a a_3 jsou navzájem různé.

31.5. Věta. *Binární* BCH *kód s plánovanou vzdáleností d má minimální vzdálenost d nebo větší.*

Důkaz. Podle 12.3 stačí dokázat, že kromě nulového slova žádné kódové slovo $\boldsymbol{v}$ nemá váhu $< d$. Zvolme tedy kódové slovo $\boldsymbol{v}$ váhy $\leqq d - 1$ a dokažme, že $\boldsymbol{v} = \mathbf{o}$. Můžeme najít $d - 1$ souřadnic $i_1, i_2, \ldots, i_{d-1}$ tak, že platí

$$v_i = 0 \quad \text{pro všechna} \quad i \neq i_1, i_2, \ldots, i_{d-1}$$

(a $v_i = 0$ nebo 1 pro $i = i_1, \ldots, i_{d-1}$). Použijeme kontrolní matici tvaru

$$\boldsymbol{H} = \begin{bmatrix}1 & \alpha & \alpha^2 & \alpha^3 & \ldots & \alpha^{n-1}\\ 1 & \alpha^2 & (\alpha^2)^2 & (\alpha^2)^3 & \ldots & (\alpha^2)^{n-1}\\ 1 & \alpha^3 & (\alpha^3)^2 & (\alpha^3)^3 & \ldots & (\alpha^3)^{n-1}\\ & \ldots & \ldots & \ldots & \ldots & \\ 1 & \alpha^{d-1} & (\alpha^{d-1})^2 & (\alpha^{d-1})^3 & \ldots & (\alpha^{d-1})^{n-1}\end{bmatrix}$$

(viz 28.4). Protože $\boldsymbol{v}$ je kódové slovo, platí $\boldsymbol{H}\boldsymbol{v}^{\mathrm{T}} = \mathbf{o}^{\mathrm{T}}$, takže

$$\boldsymbol{H}\boldsymbol{v}^{\mathrm{T}} = \begin{bmatrix}v_{i_1}\alpha^{i_1} + v_{i_2}\alpha^{i_2} + \ldots + v_{i_{d-1}}\alpha^{i_{d-1}}\\ v_{i_1}(\alpha^2)^{i_1} + v_{i_2}(\alpha^2)^{i_2} + \ldots + v_{i_{d-1}}(\alpha^2)^{i_{d-1}}\\ v_{i_1}(\alpha^3)^{i_1} + v_{i_2}(\alpha^3)^{i_2} + \ldots + v_{i_{d-1}}(\alpha^3)^{i_{d-1}}\\ \ldots\\ v_{i_1}(\alpha^{d-1})^{i_1} + v_{i_2}(\alpha^{d-1})^{i_2} + \ldots + v_{i_{d-1}}(\alpha^{d-1})^{i_{d-1}}\end{bmatrix} =$$

$$= \begin{bmatrix}\alpha^{i_1} & \alpha^{i_2} & \ldots & \alpha^{i_{d-1}}\\ (\alpha^2)^{i_1} & (\alpha^2)^{i_2} & \ldots & (\alpha^2)^{i_{d-1}}\\ (\alpha^3)^{i_1} & (\alpha^3)^{i_2} & \ldots & (\alpha^3)^{i_{d-1}}\\ \ldots & \ldots & \ldots & \\ (\alpha^{d-1})^{i_1} & (\alpha^{d-1})^{i_2} & \ldots & (\alpha^{d-1})^{i_{d-1}}\end{bmatrix}\begin{bmatrix}v_{i_1}\\ v_{i_2}\\ v_{i_3}\\ \ldots\\ v_{i_{d-1}}\end{bmatrix} = \begin{bmatrix}0\\ 0\\ 0\\ \ldots\\ 0\end{bmatrix}.$$

Dostali jsme homogenní soustavu lineárních rovnic pro neznámé $v_{i_1}, v_{i_2}, \ldots, v_{i_{d-1}}$. Abychom dokázali, že tyto neznámé jsou všechny rovny 0, stačí, když ověříme,

že matice $\boldsymbol{M}$ této soustavy má nenulový determinant. Vytkneme z prvního sloupce α^{i_1}, ze druhého α^{i_2}, atd.:

$$\det \boldsymbol{M} = \alpha^{(i_1+i_2+\ldots+i_{d-1})} \det \begin{vmatrix} 1 & 1 & \ldots & 1 \\ \alpha^{i_1} & \alpha^{i_2} & \ldots & \alpha^{i_{d-1}} \\ \alpha^{2i_1} & \alpha^{2i_2} & \ldots & \alpha^{2i_{d-1}} \\ \ldots & \ldots & \ldots & \ldots \\ \alpha^{(d-2)i_1} & \alpha^{(d-2)i_2} & \ldots & \alpha^{(d-2)i_{d-1}} \end{vmatrix}.$$

Poslední determinant je Vandermondův determinant prvků $\alpha^{i_1}, \alpha^{i_2}, \ldots, \alpha^{i_{d-1}}$. Protože $i_1, i_2, \ldots, i_{d-1}$ jsou navzájem různá čísla menší než n (= řád prvku α), jsou mocniny $\alpha^{i_1}, \ldots, \alpha^{i_{d-1}}$ navzájem různé (24.1). Z toho plyne, podle předchozího lemmatu, že $\det \boldsymbol{M} \neq 0$, a tedy $\boldsymbol{v} = \boldsymbol{o}$.

31.6. Poznámka. BCH kód s plánovanou vzdáleností d může mít i větší skutečnou minimální vzdálenost. Například BCH kód délky 9 s plánovanou vzdáleností 5 je opakovací kód (30.4), a tedy má skutečnou minimální vzdálenost 9. Je otevřeným problémem, jak určit minimální vzdálenost BCH kódu, nebo alespoň jak určit, že pro danou dvojici čísel n a d je minimální vzdálenost BCH kódu délky n rovna plánované vzdálenosti d.

Další otevřený problém je určení počtu informačních a kontrolních znaků BCH kódů. Pro kódy délek $n = 3, 7, 15, 31, \ldots$ (tvaru $2^m - 1$) ukážeme, že je to snadné, pokud plánovaná vzdálenost nepřekročí číslo $\sqrt{(n+1)} + 2$.

31.7. Lemma. Pro každé číslo $n = 2^m - 1$ a každé liché číslo $i \leqq \sqrt{(n+1)}$ platí: $i \not\equiv 2^s i \pmod{n}$ pro všechna $s = 1, 2, \ldots, m-1$.

Důkaz. Pracujeme s binárním rozvojem čísla $i = i_0 + 2i_1 + \ldots + 2^{m-1} i_{m-1}$, který zapisujeme jako binární slovo $i_{m-1}\, i_{m-2} \ldots i_1\, i_0$. Potom cyklický posun doleva znamená násobení dvěma modulo n. Podrobněji, pro číslo $j = i_{m-2}\, i_{m-3} \ldots \ldots i_1\, i_0\, i_{m-1}$ platí $2i \equiv j \pmod{n}$. Důkaz:

$$\begin{aligned} j &= i_{m-1} + 2i_0 + 4i_1 + \ldots + 2^{m-1} i_{m-2} = \\ &= i_{m-1} + 2i - 2^m i_{m-1} = \\ &= 2i - (2^m - 1)\, i_{m-1}, \end{aligned}$$

takže

$$j = 2i - n\, i_{m-1} \equiv 2i \pmod{n}.$$

Cyklický posun o dvě místa doleva dává $4i \pmod{n}$ atd., obecně při posunu o s míst dostáváme číslo $2^s i \pmod{n}$.

Stačí tedy dokázat, že cyklické posuvy o s míst doleva vytvářejí ze slova $i_{m-1} \ldots i_0$ navzájem různá slova (pro $s = 0, 1, \ldots, m-1$). Protože i je liché číslo, z nerovnosti $i \leqq \sqrt{(n+1)} = 2^{m/2}$ plyne $i < 2^{[(m+1)/2]}$, kde $[(m+1)/2]$ označuje celou část čísla $(m+1)/2$ (tj. $m/2$ pro m sudé a $(m+1)/2$ pro m liché). To znamená, že binární rozvoj čísla i má nejvýše $[(m+1)/2]$ platných cifer, takže má alespoň

$[(m-1)/2]$ nul na konci:

$$i_{m-1}\, i_{m-2} \dots i_1\, i_0 = \underbrace{000\dots000}_{[(m-1)/2]}\, XXX \dots X\, 1\,, \quad X = 1 \text{ nebo } 0\,.$$

Při cyklickém posuvu o s míst doleva poznáme z výsledného slova číslo s: je to jediná poloha jedničky, za kterou následuje alespoň $[(m-1)/2]$ nul (směrem doprava, ale v cyklickém smyslu).

31.8. Věta. *Binární* BCH *kód délky* $n = 2^m - 1$ *a plánované vzdálenosti* $d \leqq \sqrt{(n+1)} + 2$ *má* $n - k = \frac{1}{2}m(d-1)$ *kontrolních znaků a jeho generující polynom je*

$$g(x) = M_1(x)\, M_3(x) \dots M_{d-2}(x)\,.$$

Důkaz. Ověříme, že polynomy $M_1(x), M_3(x), \dots, M_{d-2}(x)$ jsou navzájem různé a mají stupeň m. Potom uvedený součin je generujícím polynomem $g(x)$ podle 28.3 a jeho stupeň (tj. počet $n-k$ informačních znaků) je m-násobkem počtu všech lichých čísel $1, 3, 5, \dots, d-2$, tj. $n - k = \frac{1}{2}m(d-1)$.

Pro každé liché číslo $i \leqq d - 2 \leqq \sqrt{(n+1)}$ platí

$$M_i(x) = (x - \alpha^i)\,(x - [\alpha^i]^2)\,(x - [\alpha^i]^4) \dots (x - [\alpha^i]^{2^{m-1}})\,.$$

To plyne z 26.11: protože je α prvek řádu $n = 2^m - 1$, platí $\alpha^{2^m} = \alpha$, a tedy i $[\alpha^i]^{2^m} = \alpha^i$. Přitom je 2^m nejmenší takový mocnitel, protože podle předchozího lemmatu je $i\, 2^s \not\equiv i \pmod n$ pro $s = 1, 2, \dots, m-1$, a tedy $\alpha^{i2^s} \neq \alpha^i$ (viz 24.1).

Vidíme, že polynomy $M_i(x)$ jsou stupně m. Ještě dokážeme, že tyto polynomy jsou navzájem různé. Jestliže totiž pro dvě čísla $i, j = 1, 3, 5, \dots, d-2$ platí $M_i(x) = M_j(x)$, máme $[\alpha^i]^{2^s} = \alpha^j$ pro některé $s = 0, 1, \dots, m-1$, a tedy $i\, 2^s \equiv j \pmod n$, viz 24.1. Z poslední rovnosti plyne $i = j$, důkaz je zcela analogický důkazu předchozího lemmatu.

31.9. Důsledek. BCH kódy pro dvojnásobné opravy délek $7, 15, 31, \dots$ mají generující polynom $g(x) = M_1(x)\, M_3(x)$ a jsou to $(2^m - 1, 2^m - 2m - 1)$-kódy.

Skutečně, pro $n = 2^m - 1$ je nerovnost $5 = d < \sqrt{(n+1)} + 2$ splněna, pokud $m = 4, 5, 6, \dots$, a případ $m = 3$ (tj. $n = 7$) je snadný, protože jde o opakovací $(7, 1)$-kód (30.5.1).

31.10. Příklady

31.10.1. Plánovaná vzdálenost $d = 7$. BCH kód pro trojnásobné opravy je $(2^m - 1, 2^m - 3m - 1)$-kód s generujícím polynomem

$$g(x) = M_1(x)\, M_3(x)\, M_5(x)\,.$$

Podmínkou je

$$7 \leqq \sqrt{(2^m)} + 2\,,$$

tj. $m = 5, 6, 7, \dots$. Pro $m = 4$ tvrzení neplatí: minimální polynomy prvků α, α^3 a α^5

jsou sice navzájem různé, ale polynom $M_5(x)$ není stupně 4, viz 31.3.3, takže zde $m - k = 10 < \frac{1}{2}m(d-1)$.

31.10.2. Plánovaná vzdálenost $d = 9$. BCH kód pro čtyřnásobné opravy je $(2^m - 1, 2^m - 4m - 1)$-kód s generujícím polynomem

$$g(x) = M_1(x)\, M_3(x)\, M_5(x)\, M_7(x)\,.$$

Podmínkou je

$$9 = \sqrt{(2^m)} + 2\,,$$

tedy opět pro všechna $m = 5, 6, 7, \ldots$. V případě $m = 5$ vzniká $(31, 11)$-kód minimální vzdálenosti $\geqq 9$. Ovšem protože

$$(\alpha^5)^8 = \alpha^{40} = \alpha^9\,,$$

platí $M_9(x) = M_5(x)$, takže stejný je BCH kód plánované vzdálenosti 11! To znamená, že tento $(31, 11)$-kód opravuje pětinásobné chyby. (A dále to znamená, že pro $d = 11$ a $n = 31$ není splněna podmínka předchozí věty — skutečně, $11 > \sqrt{(32)} + 2$.)

31.11. Kódování informačních znaků. BCH kód plánované vzdálenosti d má generující polynom $g(x)$, který je součinem všech navzájem různých polynomů mezi $M_1(x), M_3(x), \ldots, M_{d-2}(x)$. Informační znaky potom kódujeme násobením polynomem $g(x)$ nebo systematicky (při čtení pozpátku) dělením tímto polynomem, viz 21.9–21.11.

Například použijeme BCH kód délky 15 s plánovanou vzdáleností 7. Generující polynom je (podle 31.3.3)

$$g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1) = \\ = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1\,.$$

Informační znaky 01001 zakódujeme takto:

$$g(x)(x + x^4) = x^{14} + x^{12} + x^{11} + x^8 + x^4 + x^3 + x^2 + x\,,$$

a vyšleme tedy slovo 011110001001101.

31.12. Úloha. Najděte kontrolní polynom $(15, 5)$-kódu z posledního příkladu. Zakódujte informační znaky 01001 systematicky.

32. Dekódování BCH kódů I: maticová metoda

BCH kód s plánovanou vzdáleností $2t + 1$ je schopen opravit t-násobné chyby. Ukážeme, jak se tyto opravy provádějí. Najdeme tedy částečné dekódování (6.2) $\delta: Z_2^{(n)} \to K$ takové, že kdykoli při vyslání kódového slova $\boldsymbol{v}$ dojde k t-násobné chybě, potom přijaté slovo $\boldsymbol{w}$ splňuje $\delta(\boldsymbol{w}) = \boldsymbol{v}$. Stačí ovšem najít chybové slovo

$$\boldsymbol{e} = \boldsymbol{w} - \boldsymbol{v}$$

a položit $\delta(\boldsymbol{w}) = \boldsymbol{w} - \boldsymbol{e}$.

Pro každé slovo $\mathbf{w} = w_0 w_1 \ldots w_{n-1}$ tedy nastávají dvě možnosti:

a) Došlo k t-násobné chybě, tedy existuje slovo **e** váhy $\leqq t$ takové, že $\mathbf{w} - \mathbf{e}$ je kódové slovo. Pak z toho, že minimální vzdálenost kódu je alespoň $2t + 1$, plyne, že takové slovo **e** existuje jen jedno. Naše dekódování správně určí chybové slovo **e**.

b) Došlo k chybě více než t-násobné. Tuto situaci v některých případech naše dekódování objeví (viz krok c) dekódování BCH kódu pro dvojnásobné opravy v 30.3). To znamená, že místo určení slova **e** ohlásíme, že došlo k velké chybě. V některých případech ovšem velkou chybu ani neobjevíme a naše dekódování skončí určením nesprávného slova **e** (váhy $\leqq t$).

Předpokládejme, že došlo k t-násobné chybě. Hledané chybové slovo **e** má váhu p ($\leqq t$), a tedy existují souřadnice $e_{i_1}, e_{i_2}, \ldots, e_{i_p}$ rovné 1, zatímco ostatní souřadnice jsou nulové. Určíme polynom $f(x)$, tzv. lokátor chyb, jehož kořeny jsou $\alpha^{-i_1}, \alpha^{-i_2}, \ldots, \alpha^{-i_p}$. Prozkoumáním kořenů tohoto polynomu potom zjistíme, která místa přijatého slova máme opravit: opravíme w_i, právě když $f(\alpha^{-i}) = 0$.

Dekódování má tři kroky: určení syndromu přijatého slova, určení koeficientů lokátoru chyb $f(x)$ a opravení chyb nalezením kořenů polynomu $f(x)$. Než přistoupíme k dekódování, uvedeme několik informací o mocninných řadách nad tělesem.

32.1. Mocninné řady

Podobně jako je polynom formální zápis konečného slova, je *mocninná řada* formální zápis nekonečného slova. To znamená, že pro každou nekonečnou posloupnost

$$a_0\ a_1\ a_2\ a_3\ \ldots$$

prvků tělesa T používáme zápis

$$a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots$$

nebo

$$\sum_{k=0}^{\infty} a_k x^k .$$

Pozor! Tak jako polynomy nejsou funkce, ani mocninné řady nejsou funkce, a tedy nás nezajímá nic takového jako konvergence řad. (Ta ani není v konečných tělesech definována.)

V mocninné řadě vynecháváme nulové sčítance ($a_k x^k$, kde $a_k = 0$). Tedy i každý polynom je mocninnou řadou, např.

$$1 + x^2 = \sum_{k=0}^{\infty} a_k x^k ,$$

kde

$$a_0 = a_2 = 1 \quad \text{a} \quad a_k = 0 \quad \text{pro} \quad k \neq 0, 2 .$$

32.1.1. *Sčítání* mocninných řad definujeme jako u polynomů:

$$\sum_{k=0}^{\infty} a_k x^k + \sum_{k=0}^{\infty} b_k x^k = \sum_{k=0}^{\infty} (a_k + b_k)\, x^k .$$

(Například nad Z_3 platí $(1111\ldots) + (020202\ldots) = (101010\ldots)$, tedy

$$\sum_{k=0}^{\infty} x^k + \sum_{k=0}^{\infty} 2x^{2k} = \sum_{k=0}^{\infty} x^{2k+1}\,.)$$

Analogický vzorec platí i pro 3, 4, ..., p sčítanců, tedy

$$\sum_{n=1}^{p} \sum_{k=0}^{\infty} a_{nk} x^k = \sum_{k=0}^{\infty} \Big(\sum_{n=1}^{p} a_{nk}\Big) x^k\,. \tag{32.1.1}$$

32.1.2. *Násobení* je také analogické jako u polynomů:

$$\Big[\sum_{k=0}^{\infty} a_k x^k\Big]\Big[\sum_{k=0}^{\infty} b_k x^k\Big] = \sum_{k=0}^{\infty}\Big[\sum_{i=0}^{k} a_i b_{k-i}\Big] x^k\,,$$

tj. vzniká mocninná řada s koeficienty

$$c_k = a_0 b_k + a_1 b_{k-1} + \ldots + a_{k-1} b_1 + a_k b_0\,.$$

(Například nad tělesem Z_3 platí $(111111\ldots)\,(020202\ldots) = (120120\ldots)$, tj.

$$\Big[\sum_{k=0}^{\infty} x^k\Big]\Big[\sum_{k=0}^{\infty} 2x^{2k}\Big] = \Big[\sum_{k=0}^{\infty} x^{3k} + 2x^{3k+1}\Big]\,.)$$

32.1.3. Lemma (*Součet geometrické řady*). Pro každý prvek a tělesa je součin mocninné řady $\sum_{k=0}^{\infty} a^k x^k$ a polynomu $1 - ax$ roven jedné. Symbolicky:

$$\sum_{k=0}^{\infty} (ax)^k = \frac{1}{1 - ax}\,.$$

Důkaz. Polynom $1 - ax$ má koeficienty $1, -a, 0, 0, 0, \ldots$ a podle definice součinu mocninných řad je

$$(1 - ax)\sum_{k=0}^{\infty} a^k x^k = \sum_{k=0}^{\infty} c_k x^k\,,$$

kde

$$\begin{aligned} c_0 &= 1\,.\,a^0 = 1\,,\\ c_1 &= 1\,.\,a^1 - a\,.\,a^0 = 0\,,\\ c_2 &= 1\,.\,a^2 - a\,.\,a^1 = 0\,,\\ &\vdots \end{aligned}$$

32.1.4. Úloha. Ověřte, že obecněji platí známý vzorec

$$\sum_{k=m}^{\infty} (ax)^k = \frac{(ax)^m}{1 - ax}\,.$$

32.2. *První krok dekódování*: určení syndromu. Pracujeme s binárním BCH kódem délky n a plánované vzdálenosti $2t + 1$. To znamená, že máme prvek α řádu n (v některém rozšíření abecedy Z_2) a kód má generující kořeny $\alpha, \alpha^2, \ldots, \alpha^{2t}$. Použijeme kontrolní matici $\boldsymbol{H}$ ve tvaru (28.4.1). Pro každé přijaté slovo $\boldsymbol{w} = w_0 w_1 \ldots w_{n-1}$ vypočteme syndrom

$$\mathbf{s}^{\mathrm{T}} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \dots & (\alpha^2)^{n-1} \\ 1 & \alpha^3 & (\alpha^3)^2 & \dots & (\alpha^3)^{n-1} \\ & \dots & \dots & \dots & \\ 1 & \alpha^{2t} & (\alpha^{2t})^2 & \dots & (\alpha^{2t})^{n-1} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_{n-1} \end{bmatrix} =$$

$$= \begin{bmatrix} w_0 + w_1\alpha + w_2\alpha^2 + \dots + w_{n-1}\alpha^{n-1} \\ w_0 + w_1\alpha^2 + w_2(\alpha^2)^2 + \dots + w_{n-1}(\alpha^2)^{n-1} \\ w_0 + w_1\alpha^3 + w_2(\alpha^3)^2 + \dots + w_{n-1}(\alpha^3)^{n-1} \\ \dots \\ w_0 + w_1\alpha^{2t} + w_2(\alpha^{2t})^2 + \dots + w_{n-1}(\alpha^{2t})^{n-1} \end{bmatrix} = \begin{bmatrix} w(\alpha) \\ w(\alpha^2) \\ w(\alpha^3) \\ \dots \\ w(\alpha^{2t}) \end{bmatrix}.$$

Platí tedy

(32.2.1) $\quad s_k = w(\alpha^k)$ pro $k = 1, 2, \dots, 2t$.

Pokud $\mathbf{s} = \mathbf{o}$, ohlásíme, že nedošlo k chybě, tedy $\mathbf{v} = \mathbf{w}$. Pokud $\mathbf{s} \neq \mathbf{o}$, pokračujeme v dekódování.

32.2.1. Poznámka. Chybové slovo $\mathbf{e}$ má stejný syndrom (viz (12.4.1)), takže platí $s_k = e(\alpha^k)$. Víme, že počet p chybných míst (neboli váha slova $\mathbf{e}$) je nejvýše roven t. Označme $e_{i_1}, e_{i_2}, \dots, e_{i_p}$ ty složky slova $\mathbf{e}$, které jsou rovny 1. To znamená, že $e(x) = x^{i_1} + x^{i_2} + \dots + x^{i_p}$.

Označme dále

(32.2.2) $\quad a_n = \alpha^{i_n}$ pro $n = 1, 2, \dots, p$.

Potom platí

(32.2.3) $\quad s_k = a_1^k + a_2^k + \dots + a_p^k = \sum_{n=1}^{p} a_n^k$

pro každé $k = 1, 2, \dots, 2t$. (Je totiž $s_k = e(\alpha^k) = \alpha^{ki_1} + \dots + \alpha^{ki_p}$.)

32.3. *Druhý krok dekódování*: určení lokátoru chyb. *Lokátorem chyb* se nazývá polynom

(32.3.1) $\quad f(x) = (1 - a_1x)(1 - a_2x) \dots (1 - a_px)$.

Jakmile se podaří určit tento polynom a jeho kořeny (což jsou právě prvky $a_n^{-1} = \alpha^{-i_n}$ pro $n = 1, 2, \dots, p$), máme určeno chybové slovo $\mathbf{e}$:

(32.3.2) $\quad e_i = \begin{cases} 1, & \text{pokud } f(\alpha^{-i}) = 0, \\ 0 & \text{jinak}. \end{cases}$

Polynom $f(x) = f_0 + f_1x + \dots + f_px^p$ má první koeficient $f_0 = 1$ (protože $f(0) = 1$) a ostatní koeficienty určíme sestavením soustavy lineárních rovnic. K jejímu odvození použijeme mocninnou řadu

$$S(x) = \sum_{k=1}^{\infty} s_k x^k,$$

kde koeficienty jsou dány vzorcem (32.2.3). (To znamená, že známe jen koeficienty $s_1, s_2, \dots, s_{2t}$. Další koeficienty zatím neznáme — ukáže se, že je také nebudeme potřebovat.)

32.3.1. Pozorování. Pro formální derivaci (23.6.3) polynomu $f(x)$ platí

$$x f'(x) = f(x)\, S(x)\,.$$

Důkaz provedeme snadnou úpravou obou stran. Především ze vztahů (32.2.3) a (32.1.1) plyne, že

$$S(x) = \sum_{k=1}^{\infty} \sum_{n=1}^{p} (a_n x)^k = \sum_{n=1}^{p} \sum_{k=1}^{\infty} (a_n x)^k\,. \tag{32.3.3}$$

Vnitřní suma je geometrická řada a ta je rovna $a_n x/(1 - a_n x)$, viz 32.1.4. Odtud plyne

$$f(x)\, S(x) = \tag{32.3.4}$$
$$= (1 - a_1 x)(1 - a_2 x) \dots (1 - a_p x) \left[\frac{a_1 x}{1 - a_1 x} + \frac{a_2 x}{1 - a_2 x} + \dots + \frac{a_p x}{1 - a_p x} \right] =$$
$$= \sum_{n=1}^{p} a_n x (1 - a_1 x)(1 - a_2 x) \dots (1 - a_{n-1} x)(1 - a_{n+1} x) \dots (1 - a_p x)\,.$$

Abychom upravili levou stranu, použijeme vzorec pro derivaci součinu $(p(x)\, q(x))' = p'(x)\, q(x) + p(x)\, q'(x)$, který naštěstí také platí pro formální derivace polynomů. Je tedy

$$f'(x) = [(1 - a_1 x) \dots (1 - a_p x)]' =$$
$$= \sum_{n=1}^{p} (-a_n)(1 - a_1 x)(1 - a_2 x) \dots (1 - a_{n-1} x)(1 - a_{n+1} x) \dots (1 - a_p x)\,.$$

Protože $-a_n = a_n$, vidíme, že polynom $x f'(x)$ se shoduje s výrazem, na který jsme upravili součin $f(x)\, S(x)$.

32.3.2. Důsledek. Pro koeficienty $f_1, f_2, \dots, f_p$ lokátoru chyb platí tyto rovnice:

$$\begin{array}{llllll} f_1 & & & & & = s_1\,, \\ s_2 f_1 & + s_1 f_2 & + f_3 & & & = s_3\,, \\ s_4 f_1 & + s_3 f_2 & + s_2 f_3 & + s_1 f_4 + f_5 & & = s_5\,, \\ \dots & & & & & \\ s_{2p-2} f_1 & + s_{2p-3} f_2 & + s_{2p-4} f_3 & + \dots & + s_p f_{p-1} + s_{p-1} f_p & = s_{2p-1}\,. \end{array} \tag{32.3.5}$$

Stručněji,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ s_2 & s_1 & 1 & 0 & 0 & \dots & 0 & 0 \\ s_4 & s_3 & s_2 & s_1 & 1 & \dots & 0 & 0 \\ \dots & & & & & & & \\ s_{2p-2} & s_{2p-1} & & & & \dots & s_p & s_{p-1} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \dots \\ f_p \end{bmatrix} = \begin{bmatrix} s_1 \\ s_3 \\ s_5 \\ \dots \\ s_{2p-1} \end{bmatrix}.$$

Stačí si totiž uvědomit, že derivací polynomu $f(x) = f_0 + f_1 x + f_2 x^2 + \dots + f_p x^p$ je polynom

$$f'(x) = f_1 + 2 f_2 x + 3 f_3 x^2 + \dots + p f_p x^{p-1} =$$
$$= f_1 + f_3 x^2 + f_5 x^4 + \dots + f_r x^{r-1} \quad (r = p \text{ nebo } p - 1)\,.$$

Porovnejme koeficient u mocniny x polynomů $x f'(x)$ a $S(x) f(x)$:

$$f_1 = s_1 f_0 + s_0 f_1\,.$$

Protože $s_0 = 0$ (mocninná řada $S(x)$ začíná až od $k = 1$) a $f_0 = 1$, dostáváme první rovnici soustavy (32.3.5). Podobně porovnáme koeficienty u mocniny x^3:

$$f_3 = s_3 + s_2 f_1 + s_1 f_2$$

a to je druhá rovnice, atd. Poslední rovnici soustavy získáme porovnáním koeficientů u mocniny x^{2p-1}: protože $f_k = 0$ pro $k > p$, platí

$$0 = s_{2p-1} + s_{2p-2} f_1 + s_{2p-3} f_2 + \ldots + s_{p-1} f_p \,.$$

32.3.3. Poznámka. Můžeme pokračovat i dále: porovnáním koeficientů u mocniny x^{2p+1} dostáváme rovnici

$$s_{2p} f_1 + s_{2p-1} f_2 + \ldots + s_{p+1} f_p = s_{2p+1} \,. \tag{32.3.6}$$

Tu však můžeme použít, jen když $p \leqq t - 1$, protože pro $p = t$ neznáme levou stranu. Naproti tomu v soustavě (32.3.5) se vyskytují jen známé koeficienty syndromu $s_1 s_2 \ldots s_{2t}$.

Našli jsme tedy soustavu rovnic, jejíž řešení nám dává hledané koeficienty lokátoru chyb. Potíž je však v tom, že neznáme číslo p, a tedy neumíme soustavu sestavit. Navíc je zde otázka, zda soustava (32.3.5) má jediné řešení, tedy zda je determinant její matice nenulový. Oba tyto problémy řeší následující tvrzení.

32.3.4. Tvrzení. Pro každé číslo $q = 1, 2, \ldots, t + 1$ platí: matice

$$\boldsymbol{M}_q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ s_2 & s_1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ s_4 & s_3 & s_2 & s_1 & 1 & \ldots & 0 & 0 \\ \multicolumn{8}{c}{\dotfill} \\ s_{2q-2} & s_{2q-3} & \ldots & & & & s_q & s_{q-1} \end{bmatrix}$$

má determinant roven nule, pokud $q > p + 1$, ale matice $\boldsymbol{M}_p$ i $\boldsymbol{M}_{p+1}$ mají nenulový determinant (kde p je skutečný počet chyb).

Důkaz 1. Matice $\boldsymbol{M}_{p+2}$ má nulový determinant – to dokážeme tím, že najdeme soustavu homogenních lineárních rovnic s maticí soustavy $\boldsymbol{M}_{p+2}$, která má netriviální řešení. Zde je:

$$\begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ s_2 & s_1 & 1 & \ldots & 0 \\ s_4 & s_3 & s_2 & \ldots & 0 \\ \multicolumn{5}{c}{\dotfill} \\ s_{2p+1} & s_{2p} & s_{2p-1} & \ldots & s_p \\ s_{2p+2} & s_{2p+1} & s_{2p} & \ldots & s_{p+1} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ f_1 \\ f_2 \\ \ldots \\ f_p \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \ldots \\ 0 \end{bmatrix}$$

neboli

(1. rovnice)	0	$= 0$
(2. rovnice)	$s_1 + f_1$	$= 0$
(3. rovnice)	$s_3 + s_2 f_1 + s_1 f_2 + f_3$	$= 0$
....		
(poslední rovnice)	$s_{2p+1} + s_{2p} f_1 + s_{2p-1} f_2 + \ldots + s_{p+1} f_p$	$= 0$.

To jsou totiž rovnice (32.3.5) i s pokračováním (32.3.6). Podobně při dalších pokračováních zjistíme, že platí

$$\boldsymbol{M}_q \begin{bmatrix} 0 \\ 1 \\ f_1 \\ f_2 \\ \dots \\ f_p \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ 0 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} \quad \text{kdykoli} \quad q > p + 2 .$$

Proto všechny matice $\boldsymbol{M}_q$, $q \geqq p + 2$, mají nulový determinant.

2. Abychom ověřili, že matice $\boldsymbol{M}_p$ a $\boldsymbol{M}_{p+1}$ mají nenulový determinant, zavedeme funkce p proměnných, analogické vzorci (32.2.3):

$$s_k(x_1, x_2, \dots, x_p) = x_1^k + x_2^k + \dots + x_p^k ,$$

takže syndromy jsou nyní hodnotami $s_k(a_1, a_2, \dots, a_p)$. Pro matici funkcí

$$\boldsymbol{M}_p(x_1, x_2, \dots, x_n) = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ s_2 & s_1 & 1 & \dots & 0 \\ s_4 & s_3 & s_2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ s_{2p-2} & s_{2p-1} & s_{2p} & \dots & s_{p-1} \end{bmatrix},$$

kde s_k zde znamená funkci $s_k(x_1, \dots, x_p)$, platí

(32.3.7) $$\det \boldsymbol{M}_p(x_1, x_2, \dots, x_n) = \prod_{i<j} (x_i + x_j) .$$

Např. pro $p = 2$

$$\det \begin{bmatrix} 1 & 0 \\ x_1^2 + x_2^2 & x_1 + x_2 \end{bmatrix} = x_1 + x_2 ,$$

pro $p = 3$

$$\det \begin{bmatrix} 1 & 0 & 0 \\ x_1^2 + x_2^2 + x_3^2 & x_1 + x_2 + x_3 & 1 \\ x_1^4 + x_2^4 + x_3^4 & x_1^3 + x_2^3 + x_3^3 & x_1^2 + x_2^2 + x_3^2 \end{bmatrix} =$$

$$= \det \begin{bmatrix} x_1 + x_2 + x_3 & 1 \\ x_1^3 + x_2^3 + x_3^3 & x_1^2 + x_2^2 + x_3^2 \end{bmatrix} =$$

$$= (x_1 + x_2 + x_3)(x_1^2 + x_2^2 + x_3^2) + (x_1^3 + x_2^3 + x_3^3) =$$

$$= x_1 x_2^2 + x_1 x_3^2 + x_2 x_1^2 + x_2 x_3^2 + x_3 x_1^2 + x_3 x_2^2 =$$

$$= (x_1 + x_2)(x_1 + x_3)(x_2 + x_3) .$$

Obecně ponecháme rovnost (32.3.7) bez důkazu. Čtenář jej může najít např. v [7].

Platí tedy, po dosazení $x_i = a_i$,

$$\det \boldsymbol{M}_p = \prod_{i<j} (a_i + a_j) \neq 0 ,$$

protože prvky $a_1, \ldots, a_p$ jsou navzájem různé. Nakonec pro matici $\boldsymbol{M}_{p+1}(x_1, x_2, \ldots, x_p)$ řádu $p + 1$ (funkcí p proměnných) platí

$$\det \boldsymbol{M}_{p+1}(x_1, x_2, \ldots, x_p) = x_1 x_2 \ldots x_p \prod_{i<j} (x_i + x_j),$$

a tedy po dosazení

$$\det \boldsymbol{M}_{p+1} = a_1 a_2 \ldots a_p \prod_{i<j} (a_i + a_j) \neq 0.$$

32.3.5. Shrnutí. Koeficienty lokátoru chyb najdeme řešením soustavy

$$\boldsymbol{M}_q \begin{bmatrix} f_1 \\ f_2 \\ \ldots \\ f_q \end{bmatrix} = \begin{bmatrix} s_1 \\ s_3 \\ \ldots \\ s_{2q-1} \end{bmatrix},$$

kde q je největší z čísel $1, 2, \ldots, t$ takové, že $\det \boldsymbol{M}_q \neq 0$. To znamená, že nejprve položíme $q = t$.

a) Je-li $\det \boldsymbol{M}_t \neq 0$, řešíme uvedenou soustavu a dostaneme koeficienty lokátoru chyb. (Skutečně, podle tvrzení 32.3.4 je $t \leqq p + 1$, takže buď došlo k t chybám, nebo k $t - 1$ chybám. V prvém případě jde o soustavu (32.3.5) a ve druhém ji rozšíříme o rovnici (32.3.6).)

b) Je-li $\det \boldsymbol{M}_t = 0$, položíme $q = t - 2$ a postupujeme analogicky. Tedy pokud $\det \boldsymbol{M}_{t-2} \neq 0$, řešíme uvedenou soustavu, a pokud $\det \boldsymbol{M}_{t-2} = 0$, položíme $q = t - 4$ atd.

c) Jestliže všechny determinanty vyšly nulové (a syndrom je nenulový), ohlásíme, že došlo k velké chybě.

32.3.6. Příklad: BCH kódy pro trojnásobné opravy. Platí

$$\det \boldsymbol{M}_3 = \det \begin{bmatrix} 1 & 0 & 0 \\ s_2 & s_1 & 1 \\ s_4 & s_3 & s_2 \end{bmatrix} = s_1 s_2 + s_3 .$$

a) Jestliže $s_1 s_2 \neq s_3$, došlo k 3 nebo 2 chybám. Lokátorem chyb je polynom $f(x) = 1 + f_1 x + f_2 x^2 + f_3 x^3$, kde

$$\begin{aligned} f_1 &= s_1 , \\ s_2 f_1 + s_1 f_2 + f_3 &= s_3 , \\ s_4 f_1 + s_3 f_2 + s_2 f_3 &= s_5 . \end{aligned}$$

b) Jestliže $s_1 s_2 = s_3$, došlo k 1 chybě a lokátorem chyb je polynom $f(x) = 1 + f_1 x$, kde $f_1 = s_1$, tedy polynom $1 + s_1 x$.

Například použijeme (15, 5)-kód z 31.3.3, kde $\alpha = z$ je prvek tělesa $GF(16)$ v dodatku A1. Přijali jsme slovo

$$\boldsymbol{w} = 101000000010000 ,$$

tj. $w(z) = 1 + z^2 + z^{10}$. Určíme syndrom

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{bmatrix} = \begin{bmatrix} 1 + \alpha^2 + \alpha^{10} \\ 1 + \alpha^4 + \alpha^5 \\ 1 + \alpha^6 + 1 \\ 1 + \alpha^8 + \alpha^{10} \\ 1 + \alpha^{10} + \alpha^5 \\ 1 + \alpha^{12} + 1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \alpha^2 \\ \alpha^6 \\ \alpha^4 \\ 0 \\ \alpha^{12} \end{bmatrix}.$$

Protože $s_1 s_2 \neq s_3$, řešíme soustavu

$$\begin{aligned} f_1 &= \alpha, \\ \alpha^2 f_1 + \alpha f_2 + f_3 &= \alpha^6, \\ \alpha^4 f_1 + \alpha^6 f_2 + \alpha^2 f_3 &= 0. \end{aligned}$$

Řešení je $f_1 = \alpha, f_2 = \alpha^6, f_3 = \alpha^{12}$, takže

$$f(x) = 1 + \alpha x + \alpha^6 x^2 + \alpha^{12} x^3.$$

32.3.7. Úloha. Ověřte, že u BCH kódů pro dvojnásobné opravy se určení lokátoru chyb provádí jako v 30.6.

32.4. *Třetí krok*: provedení opravy. Jestliže známe lokátor chyb $f(x)$, potom určíme jeho kořeny postupným dosazováním prvků $1, \alpha^{-1}, \alpha^{-2}, \ldots$. Opravíme i-té místo slova $\boldsymbol{w}$, právě když $f(\alpha^{-i}) = 0$, viz vztah (32.3.2).

Například při použití (15, 5)-kódu a přijetí slova 101000000010000 jsme určili lokátor chyb $f(x) = 1 + \alpha x + \alpha^6 x^2 + \alpha^{12} x^3$ (32.3.6). Jeho kořeny jsou $1, \alpha^{-2}$ a α^{-10}, takže opravíme w_0, w_2 a w_{10}. Vysláno bylo slovo $\boldsymbol{v} = 000000000000000$.

32.5. Úloha. Při použití (15, 5)-kódu dekódujte toto slovo:
100000111000000.

33. BCH kódy a Reedovy-Solomonovy kódy

Nyní uvedeme BCH kódy nad q-znakovou abecedou $GF(q)$, kde q je mocnina prvočísla. I pro $q = 2$ je definice obecnější než ta, se kterou jsme dosud pracovali. Vlastnosti q-znakových BCH kódů jsou analogické vlastnostem binárních BCH kódů: Volíme vzdálenost d a ukážeme, že skutečná minimální vzdálenost je $\geqq d$. Můžeme tedy opravit t-násobné chyby pro $d = 2t + 1$. Metodu dekódování uvedeme v další kapitole.

Zajímavý speciální případ jsou Reedovy-Solomonovy kódy, což jsou BCH kódy délky $q - 1$. Jsou důležité jednak proto, že mají největší myslitelnou minimální vzdálenost, a jednak proto, že pomocí nich lze sestrojit dobré binární kódy.

33.1. Definice. q-*znakový* BCH *kód s plánovanou vzdáleností* $d = 2, 3, 4, \ldots$ je kód délky $n \geqq d$, n nesoudělné s q, který má generující kořeny

$\alpha^j, \alpha^{j+1}, \alpha^{j+2}, \ldots, \alpha^{j+d-2}$ (pro některé $j = 0, 1, \ldots, n - d + 1$),
kde α je prvek n-tého řádu v tělese $GF(q^m)$.

33.2. Poznámky

33.2.1. Pokud $j = 1$, tedy generující kořeny jsou $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{d-1}$, mluvíme o BCH *kódu v užším smyslu*. V předchozích článcích jsme se tedy zabývali jen binárními BCH kódy v užším smyslu.

33.2.2. Protože předpokládáme nesoudělnost čísel n a q, prvek α vždy existuje, např. v tělese $GF(q^{\varphi(n)})$ (25.8.3).

33.2.3. Kontrolní matice BCH kódu je

$$\boldsymbol{H} = \begin{bmatrix} 1 & \alpha^j & (\alpha^j)^2 & (\alpha^j)^3 & \ldots & (\alpha^j)^{n-1} \\ 1 & \alpha^{j+1} & (\alpha^{j+1})^2 & (\alpha^{j+1})^3 & \ldots & (\alpha^{j+1})^{n-1} \\ \multicolumn{6}{c}{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} \\ 1 & \alpha^{j+d-2} & (\alpha^{j+d-2})^2 & (\alpha^{j+d-2})^3 & \ldots & (\alpha^{j+d-2})^{n-1} \end{bmatrix}.$$

Viz 28.9. Důkaz následující věty, založený na použití této kontrolní matice, je analogický důkazu v 31.5.

33.3. Věta. BCH *kód s plánovanou vzdáleností d má minimální vzdálenost d, nebo větší.*

33.4. Příklady

33.4.1. Binární BCH kód délky 7 s generujícími kořeny

1 a α.

Zvolme α primitivní prvek tělesa $GF(8)$ (viz dodatek A1). Potom α má minimální polynom $M_1(x) = x^3 + x + 1$ a 1 má minimální polynom $M_0(x) = x - 1 = x + 1$. Generující polynom kódu je

$$g(x) = (x + 1)(x^3 + x + 1).$$

To je zmenšený (16.4) Hammingův kód: kódová slova jsou ta slova $v(z)$, pro která platí $v(1) = 0$ (tedy $v_0 v_1 \ldots v_{n-1}$ má sudou paritu) a $v(\alpha) = 0$ (tedy $v_0 v_1 \ldots v_{n-1}$ patří k Hammingovu kódu, viz 28.3.1).

33.4.2. Ternární BCH kód délky 8 s generujícími kořeny

$\alpha^2, \alpha^3, \alpha^4, \alpha^5$.

Zvolme α primitivní prvek tělesa $GF(9)$ (viz dodatek A1). Potom prvek α, a tedy i α^3, má minimální polynom $x^2 + x + 2$. Prvek α^2 má minimální polynom (podle 26.11)

$$M_2(x) = (x - \alpha^2)(x - \alpha^6) = x^2 + 1.$$

Prvek $\alpha^4 = 2$ má minimální polynom $M_4(x) = x - 2 = x + 2$ a prvek α^5 má

minimální polynom

$$M_5(x) = (x - \alpha^5)(x - \alpha^7) = x^2 + 2x + 2\,.$$

Generující polynom kódu je tedy

$$g(x) = (x^2 + 1)(x^2 + x + 2)(x + 2)(x^2 + 2x + 2)\,.$$

Zjišťujeme, že jen jeden znak je informační, takže jde o opakovací kód.

33.4.3. Ternární BCH kód délky 8 s generujícími kořeny 1, α, α^2 a α^3. Zvolme opět α primitivní prvek v $GF(9)$. Generující polynom je nyní

$$g(x) = M_0(x)\,M_1(x)\,M_2(x) = (x - 1)(x^2 + x + 2)(x^2 + 1)\,,$$

takže jde o (8, 3)-kód opravující dvojnásobné chyby.

33.4.4. 4-znakový BCH kód délky 3 s generujícím kořenem α^2. Kódovou abecedu volme ve tvaru

$$GF(4) = \{0, 1, z, t\}\,,$$

kde

$$t = z^2 = 1 + z$$

(viz dodatek A1). Protože prvek z má řád 3, můžeme volit $\alpha = z$. Minimální polynom prvku $\alpha^2 = t$ je ovšem $x - t\,(= x + t)$, takže

$$g(x) = x + t\,.$$

Jde o (3, 2)-kód s generující maticí

$$\begin{bmatrix} t & 1 & 0 \\ 0 & t & 1 \end{bmatrix}.$$

33.5. Reedovy-Solomonovy kódy

BCH kód délky $q - 1$, kde q je počet kódových znaků, se nazývá *Reedův-Solomonův kód*. Tyto kódy jsou významné jen u velkých abeced (a nejsou nikdy binární). Jsou důležité např. proto, že pomocí nich lze konstruovat velmi účinné binární kódy.

Například kód v posledním příkladu je Reedův-Solomonův (3, 2)-kód. 7-znakový Reedův-Solomonův kód plánované vzdálenosti 3 s generujícími kořeny α, α^2 a α^3 můžeme určit tak, že volíme

$$\alpha = 3 \in Z_7\,,$$

primitivní prvek tělesa Z_7. Potom generující polynom je

$$g(x) = (x - 3)(x - 2)(x - 6) = x^3 + 2x^2 + x + 6\,,$$

takže jde o (6, 3)-kód s generující maticí

$$\begin{bmatrix} 6 & 1 & 2 & 1 & 0 & 0 \\ 0 & 6 & 1 & 2 & 1 & 0 \\ 0 & 0 & 6 & 1 & 2 & 1 \end{bmatrix}.$$

33.5.1. Tvrzení. Reedův-Solomonův (n, k)-kód s generujícími kořeny $\alpha^i, \alpha^{i+1}, \ldots, \alpha^{i+d-2}$ má generující polynom

$$g(x) = (x - \alpha^i)(x - \alpha^{i+1}) \dots (x - \alpha^{i+d-2})$$

a jeho minimální vzdálenost je

$$d = n - k + 1 .$$

Je to největší možná minimální vzdálenost lineárního (n, k)-kódu.

Důkaz. Protože počet q kódových znaků je roven $n + 1$, má primitivní prvek α tělesa $CF(q)$ řád n. To znamená, že uvedený polynom $g(x) = (x - \alpha^i) \dots (x - \alpha^{i+d-2})$ je polynom nad tělesem $GF(q)$. Je zřejmé, že polynom nad tělesem $GF(q)$ je dělitelný polynomem $g(x)$, právě když má kořeny $\alpha^i, \dots, \alpha^{i+d-2}$ (viz 23.3). Odtud plyne, že $g(x)$ je generující polynom kódu.

Minimální vzdálenost libovolného lineárního (n, k)-kódu je nejvýše $n - k + 1$ (7.6). Minimální vzdálenost Reedova-Solomonova kódu je větší nebo rovna plánované vzdálenosti d, $d = n - k + 1$. Odtud plyne, že skutečná minimální vzdálenost nemůže být větší než d, a je tedy rovna d.

33.5.2. Příklad: 8-znakový Reedův-Solomonův kód minimální vzdálenosti 5. Označme $\alpha = z$ primitivní prvek tělesa

$$GF(8) = Z_2/(x^3 + x + 1)$$

(viz dodatek A1). Kód nad abecedou $GF(8)$ s generujícími kořeny

$$\alpha, \alpha^2, \alpha^3 \text{ a } \alpha^4$$

má generující polynom

$$g(x) = (x + z)(x + z^2)(x + z^3)(x + z^4) = x^4 + z^3x^3 + z^5x^2 + z .$$

Generující matice je tedy

$$\mathbf{G} = \begin{bmatrix} z & 0 & z^5 & z^3 & 1 & 0 & 0 \\ 0 & z & 0 & z^5 & z^3 & 1 & 0 \\ 0 & 0 & z & 0 & z^5 & z^3 & 1 \end{bmatrix} .$$

33.5.3. Vytváření dobrých binárních kódů. V Reedově-Solomonově (n, k)-kódu, kde $n = 2^m - 1$, můžeme každý znak nahradit binárním slovem délky $m + 1$. Kódová abeceda má totiž $n + 1 = 2^m$ znaků, a tedy tvoří těleso

$$GF(2^m) = Z_2/f(x) .$$

Každý znak a je polynomem $a_0 + a_1z + \dots + a_{m-1}z^{m-1}$ nad tělesem Z_2, který nahradíme slovem délky $m + 1$ tak, že přidáme kontrolu parity:

$$a \leftrightarrow a_0\, a_1 \dots a_{m-1}\, a_m ,$$

kde

$$a_m = a_0 + a_1 + \dots + a_{m-1} \quad (\text{nad } Z_2) .$$

Vznikne binární (n^*, k^*)-kód, kde

$$n^* = n(m + 1) = (2^m - 1)(m + 1)$$

a

$$k^* = mk .$$

Každý řádek $\mathbf{v}$ generující matice nahradíme „binárně přeloženými" řádky

$\mathbf{v}, z\mathbf{v}, z^2\mathbf{v}, \ldots, z^{m-1}\mathbf{v}$.

Například abecedu $GF(4) = Z_2/(x^2 + x + 1)$ (viz A1) překládáme takto:

$0 \leftrightarrow 000 \qquad z \leftrightarrow 011$

$1 \leftrightarrow 101 \qquad t \leftrightarrow 110$.

4-znakový (3, 2)-kód v 33.4.4 přeložíme jako binární (9, 4)-kód s touto generující maticí:

$$\begin{bmatrix} 1\,1\,0 & 1\,0\,1 & 0\,0\,0 \\ 1\,0\,1 & 0\,1\,1 & 0\,0\,0 \\ 0\,0\,0 & 1\,1\,0 & 1\,1\,0 \\ 0\,0\,0 & 1\,0\,1 & 0\,1\,1 \end{bmatrix}.$$

Takto vytvořené binární kódy mají velmi dobré parametry. Minimální vzdálenost d binárního kódu je alespoň dvojnásobkem minimální vzdálenosti Reedova-Solomonova kódu,

$$d^* \geqq 2\,(2^m - k + 1).$$

Například z Reedova-Solomonova kódu délky 15 a minimální vzdálenosti 6 vznikne binární (75, 40)-kód minimální vzdálenosti 12. To je kód opravující 5násobné chyby. Pro srovnání: BCH kód délky 63, opravující 5násobné chyby, má 36 informačních znaků (viz dodatek A2).

Binární kódy, které takto vytvoříme, jsou lineární, ale obvykle ne cyklické.

33.6. Shlukové chyby. Zatím jsme se zabývali objevováním a opravováním chyb, které se v kódových slovech vyskytují zcela náhodně. Při některých aplikacích se však chyby vyskytují zpravidla ve shlucích, tj., v blízkosti chybného znaku lze očekávat další chybné znaky. Tak tomu bývá při porušení magnetického disku nebo při poklesu napětí během vysílání apod. Binární kódy, vytvořené z Reedových-Solomonových kódů, se dobře využívají k opravám shlukových chyb.

33.6.1. Definice. *Shlukem chyb délky b* se nazývá slovo $\mathbf{e} = e_1 e_2 \ldots e_n$, jehož všechny nenulové složky tvoří část, ležící mezi b po sobě následujícími znaky. Tedy slovo tvaru

$$\mathbf{e} = 000\ldots0e_i e_{i+1}\ldots e_{i+b-1}00\ldots000.$$

33.6.2. Binární lineární kód *objevuje shluk chyb* délky b, jestliže žádné nenulové kódové slovo není shlukem délky b. To totiž znamená, že při vyslání kódového slova $\mathbf{v}$ a přijetí slova $\mathbf{v} + \mathbf{e}$, kde $\mathbf{e} \neq \mathbf{o}$ je shluk chyb délky b, není $\mathbf{v} + \mathbf{e}$ kódové slovo. (Kdyby $\mathbf{v} + \mathbf{e}$ bylo kódové slovo, potom i $\mathbf{e} = (\mathbf{v} + \mathbf{e}) - \mathbf{v}$ by bylo kódové slovo.) Při přijetí slova $\mathbf{v} + \mathbf{e}$ tedy víme, že došlo k chybě.

Například cyklický Hammingův (7, 4)-kód, viz 21.7.3, objevuje shluk chyb délky 3. To je totiž buďto slovo

$$\mathbf{e} = e_0 e_1 e_2 0000 \leftrightarrow e_0 + e_1 z + e_2 z^2,$$

anebo (cyklický) posun takového slova. Přitom polynom prvního nebo druhého stupně není kódovým slovem, takže ani jeho cyklický posun není kódovým slovem. Obecněji:

33.6.3. Tvrzení. Cyklický (n, k)-kód objevuje shluky chyb délky $n - k$, ale neobjevuje (každý) shluk chyb délky $n - k + 1$.

Důkaz. Generující polynom $g(z)$ je stupně $n - k$ a je to tedy slovo tvaru $g_0 g_1 \ldots g_{n-k} 00 \ldots 0$. To je shluk chyb délky $n - k + 1$, který je kódovým slovem. Odtud plyne, že shluk chyb délky $n - k + 1$ kód obecně neobjevuje.

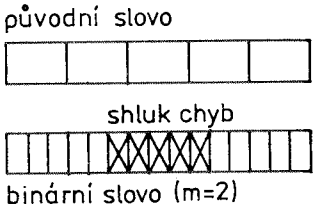
Každý shluk chyb délky $n - k$ je posunem slova typu

$$e_0 e_1 \ldots e_{n-k-1} 00 \ldots 0 \leftrightarrow e(z),$$

kde $e(z)$ je polynom stupně menšího než $g(z)$. Je-li $e(z) \neq 0$, není tedy $e(z)$ kódovým slovem cyklického kódu. Proto ani žádný cyklický posun slova $e(z)$ není kódovým slovem. Obyčejný posun (o nejvýše k míst vpravo) je speciálním případem cyklického posunu. To znamená, že kód neobsahuje nenulové shluky chyb délky $n - k$.

33.6.4. Binární kód *opravuje shluk chyb* délky b, jestliže při vyslání slova $\mathbf{v}$ a přijetí slova $\mathbf{v} + \mathbf{e}$, kde $\mathbf{e}$ je shluk chyb délky b, poznáme, že bylo vysláno slovo $\mathbf{v}$. Kód opravuje dva shluky délky b, jestliže při vyslání slova $\mathbf{v}$ a přijetí slova $\mathbf{v} + \mathbf{e}_1 + \mathbf{e}_2$, kde $\mathbf{e}_1$ a $\mathbf{e}_2$ jsou shluky délky b, poznáme, že bylo vysláno slovo $\mathbf{v}$, atd.

Binární kódy, vytvořené z Reedových-Solomonových kódů, jsou vhodné pro opravy shluků chyb. Buď K Reedův-Solomonův (n, k)-kód, $n = 2^{m-1}$. Binární kód K^*, který z kódu K vytvoříme (viz 33.5.3), opravuje shluk chyb délky $m + 2$. Takový shluk totiž poškodí nejvýše dva znaky původního slova: každému znaku původního slova odpovídá $m + 1$ znaků binárního slova, takže k poškození tří znaků kódu K musíme vytvořit shluk chyb délky alespoň $(m + 1) + 2 = m + 3$.



Obr. 25

Jestliže tedy Reedův-Solomonův kód má plánovanou vzdálenost d, potom binární kód K^* opravuje t shluků chyb délky $m + 2$ kdykoli $t < d/4$. (Tím se totiž poškodí nejvýše $2t$ znaků původního slova a platí $2t < d/2$.) Například binární (75, 40)-kód v 33.5.3, vzniklý z Reedova-Solomonova (15, 10)-kódu, opravuje shluk chyb délky 6 (protože $m = 4$) a objevuje shluk chyb délky 17 (protože takový shluk způsobí

v původním slově shluk chyb délky 5, a ten Reedův-Solomonův kód objeví podle (33.6.3)). Tento binární kód má minimální vzdálenost 12, takže opraví 5 chyb a objeví 11 chyb.

Z Reedova-Solomonova (15, 7)-kódu vznikne binární (75, 28)-kód. Ten je schopen opravit 2 shluky chyb délky 6 (protože $m = 4$ a $d = 6$).

33.6.5. Úloha. Ověřte, že binární kód, vzniklý z Reedova-Solomonova kódu plánované vzdálenosti d opraví t shluků chyb délky $2m + 3$ kdykoli $t < d/6$. Např. uvedený (75, 28)-kód opraví shluk chyb délky 11. (Návod: shluk chyb délky $2m + 3 = 2(m + 1) + 1$ poškodí nejvýše tři znaky původního kódu.)

34. Dekódování BCH kódů II: Euklidův algoritmus

Maticová metoda dekódování BCH kódů (kap. 32) není vhodná pro větší plánované vzdálenosti. Při této metodě počítáme velké determinanty (řádu $t, t - 1, \ldots, p + 1$, kde p je skutečný počet chyb, viz 32.3.5) a to je zdlouhavé, pokud $t \geqq 4$. Proto uvádíme rychlejší metodu dekódování, založenou na Euklidově algoritmu hledání největšího společného dělitele polynomů (který nejdříve připomeneme).

Stejně jako u maticové metody, budeme určovat lokátor chyb $f(x)$, tedy polynom, který má kořen α^{-i}, právě když i-té místo je chybné. U binárních kódů už nemusíme znát nic dalšího. V případě větší kódové abecedy je ovšem ještě třeba určit, jak máme opravit i-té místo. K tomu slouží další polynom, tzv. evaluátor chyb $c(x)$.

34.1. Euklidův algoritmus

Euklides našel (před více než dvěma tisíci lety) algoritmus k určení největšího společného dělitele dvou celých čísel. Stejnou metodou můžeme určit *největšího společného dělitele* dvou polynomů $a_0(x)$ a $a_1(x)$. To je podle definice polynom co nejvyššího stupně, který dělí polynomy $a_0(x)$ i $a_1(x)$. Pokud je $f(x)$ největším společným dělitelem, je jím také každý skalární násobek $tf(x)$, $t \neq 0$. (Někteří autoři proto požadují normovanost, 23.10.5, aby této nejednoznačnosti zabránili.)

34.1.1. Příklad. Na tělesem Z_3 mají polynomy $x^2 + 2$ a $x^2 + x$ největšího společného dělitele $x + 1$. Platí totiž

$$x^2 + 2 = (x + 1)(x + 2) \quad \text{a} \quad x^2 + x = (x + 1)x .$$

Polynom $2x + 2$ také dělí oba polynomy, takže je rovněž největším společným dělitelem.

34.1.2. Popis algoritmu. Jsou dány nenulové polynomy $a_0(x)$ a $a_1(x)$ nad tělesem T.

1. krok. Dělíme polynom $a_0(x)$ polynomem $a_1(x)$ (viz 20.6):

$$a_0(x) = q_1(x)\, a_1(x) + a_2(x), \quad \text{st } a_2(x) < \text{st } a_1(x).$$

Pokud vyšlo toto dělení beze zbytku, tj. $a_2(x) = 0$, jsme hotovi: největší společný dělitel je polynom $a_1(x)$. Pokud $a_2(x) \neq 0$, přejdeme k dalšímu kroku.

k-tý krok. Dělíme polynom $a_{k-1}(x)$ polynomem $a_k(x)$:

$$a_{k-1}(x) = q_k(x)\, a_k(x) + a_{k+1}(x), \quad \text{st } a_{k+1}(x) < \text{st } a_k(x).$$

Pokud jde o dělení beze zbytku, tj. $a_{k+1}(x) = 0$, jsme hotovi: největší společný dělitel je polynom $a_k(x)$. Pokud $a_{k+1}(x) \neq 0$, přejdeme k dalšímu kroku.

Největší společný dělitel je tedy první dělitel, který dělí beze zbytku. Protože stupně polynomů klesají, $\text{st } a_1(x) > \text{st } a_2(x) > \text{st } a_3(x) > \ldots > \text{st } a_k(x)$ (pokud $\text{st } a_k(x) \neq -1$), takový dělitel existuje.

34.1.3. Příklad: Největší společný dělitel polynomů $x^2 + x$ a $x^2 + 2$ nad tělesem Z_3.

1. krok:

$$\underline{(x^2 + x)} : (x^2 + 2) = 1 = q_1(x),$$
$$x + 1$$

2. krok:

$$\underline{(x^2 + 2)} : (x + 1) = x = q_2(x).$$
$$0$$

Největší společný dělitel je polynom $x + 1$.

34.1.4. Poznámka. V Euklidově algoritmu můžeme každý krok vyjádřit jako lineární kombinaci polynomů $a_0(x)$ a $a_1(x)$:

(34.1.1) $$a_k(x) = (-1)^k [v_k(x)\, a_0(x) - u_k(x)\, a_1(x)],$$

kde polynomy $u_k(x)$ a $v_k(x)$ definujeme rekurentně:

$$u_0(x) = 0, \qquad v_0(x) = 1,$$
$$u_1(x) = 1, \qquad v_1(x) = 0,$$
$$u_{k+1}(x) = q_k(x)\, u_k(x) + u_{k-1}(x), \qquad v_{k+1}(x) = q_k(x)\, v_k(x) + v_{k-1}(x).$$

O tom se snadno přesvědčíte matematickou indukcí.

Například pro polynomy $x^2 + x$ a $x^2 + 2$ (nad tělesem Z_3) z předchozího příkladu máme

$$u_0(x) = 0, \qquad v_0(x) = 1,$$
$$u_1(x) = 1, \qquad v_1(x) = 0,$$
$$u_2(x) = 1, \qquad v_2(x) = 1,$$
$$u_3(x) = x + 1, \qquad v_3(x) = x.$$

Platí

$$a_0(x) = x^2 + x = (-1)^0 [(x^2 + x) - 0],$$
$$a_1(x) = x^2 + 2 = -1[0 - (x^2 + 2)],$$
$$a_2(x) = x + 1 = (-1)^2[(x^2 + x) - (x^2 + 2)],$$
$$a_3(x) = 0 = (-1)^3 [x(x^2 + x) - (x + 1)(x^2 + 2)].$$

34.1.5. Důsledek. Dva polynomy $a_0(x)$ a $a_1(x)$ jsou nesoudělné, tj. mají největší společný dělitel 1, právě když existují polynomy $p(x)$ a $q(x)$, pro které platí

$$a_0(x)\, p(x) + a_1(x)\, q(x) = 1 .$$

Skutečně, z uvedené rovnice plyne, že každý společný dělitel polynomů $a_0(x)$ a $a_1(x)$ dělí i polynom 1 (a je tedy stupně 0). Naopak, pokud jsou polynomy $a_0(x)$ a $a_1(x)$ nesoudělné, je některý polynom $a_k(x)$ stupně 0, $a_k(x) = b\ (\neq 0)$ a z rovnice (34.1.1) plyne uvedená rovnice pro $p(x) = b^{-1}(-1)^k v_k(x)$ a $q(x) = b^{-1}(-1)^{k+1} u_k(x)$.

34.2. Úlohy

34.2.1. Ověřte, že pokud jsou polynomy $a_0(x)$ a $a_1(x)$ nesoudělné, potom každý polynom $f(x)$ je jejich kombinací, tj. je tvaru $f(x) = a_0(x)\, p(x) + a_1(x)\, q(x)$. Návod: rovnost v 34.1.5 násobte členem $f(x)$.

34.2.2. Najděte největší společný dělitel polynomů $x^2 + x + 1$ a $x + 1$ nad tělesem Z_2. Napište polynom $x^2 + 1$ jako kombinaci těchto dvou polynomů. Jak se změní tato úloha nad tělesem Z_3?

34.2.3. Polynomy v 34.1.4 splňují pro všechna $k = 0, 1, 2, \ldots$ vztah

$$u_k(x)\, v_{k+1}(x) - v_k(x)\, u_{k+1}(x) = \pm 1 .$$

Ověřte to matematickou indukcí a odvoďte, že polynomy $u_k(x)$ a $v_k(x)$ jsou nesoudělné (pro každé $k = 0, 1, 2, \ldots$). Návod: 34.1.5.

34.2.4. Ověřte, že v Euklidově algoritmu stupně polynomů splňují

$$\operatorname{st} q_k(x) = \operatorname{st} a_{k-1}(x) - \operatorname{st} a_k(x)$$

a

$$\operatorname{st} u_k(x) = \operatorname{st} a_0(x) - \operatorname{st} a_{k-1}(x) .$$

34.3. Předpoklady o dekódování

Pracujeme s q-znakovým BCH kódem délky n (nesoudělné s číslem q), který opravuje t-násobné chyby. Jeho plánovaná vzdálenost je $2t + 1$ a má tedy generující kořeny

$$\alpha^j, \alpha^{j+1}, \ldots, \alpha^{j+2t-1} ,$$

kde α je prvek řádu n v některém tělese $GF(q^m)$.

Přijali jsme slovo $\mathbf{w}$ a předpokládáme, že došlo k t nebo méně chybám. Hledáme chybové slovo $\mathbf{e}$, tedy slovo váhy $p \leqq t$ takové, že $\mathbf{w} - \mathbf{e}$ je kódové slovo. Výsledkem dekódování je buď určení slova $\mathbf{e}$, nebo ohlášení, že došlo k velkému počtu chyb (tj. k počtu $> t$), takže slovo $\mathbf{e}$ neexistuje.

Označme $e_{i_1}, e_{i_2}, \ldots, e_{i_p}$ všechny nenulové složky hledaného slova $\mathbf{e}$. Pro každé $k = 1, 2, \ldots, p$ položme

(34.3.1) $\quad a_k = \alpha^{i_k}$ a $b_k = e_{i_k} \alpha^{j i_k} = e_{i_k} a_k^j$.

K určení slova $\mathbf{e}$ stačí určit prvky $a_1, \ldots, a_p$ a $b_1, \ldots, b_p$, protože platí $b_k = e_{i_k} a_k^j$,

takže

$$e_i = \begin{cases} b_k a_k^{-j}, & \text{pokud } i = i_k, \\ 0, & \text{pokud } i \neq i_1, \ldots, i_p. \end{cases}$$

Jako v binárním případě definujeme *lokátor chyb* předpisem

(34.3.2) $$f(x) = (1 - a_1 x)(1 - a_2 x) \ldots (1 - a_p x).$$

Jeho kořeny jsou $a_1^{-1}, a_2^{-1}, \ldots, a_p^{-1}$ a ty určují polohu chyb. (Opravit máme složku w_i, právě když α^{-i} je kořen lokátoru chyb.) V případě víceznakových kódů zavádíme další polynom, *evaluátor chyb* $c(x)$, jako součet polynomů

$$b_k(1 - a_1 x)(1 - a_2 x) \ldots (1 - a_{k-1} x)(1 - a_{k+1} x) \ldots (1 - a_p x),$$

které můžeme psát ve tvaru $b_k f(k)/(1 - a_k x)$. Definujeme tedy

(34.3.3) $$c(x) = \sum_{k=1}^{p} b_k \frac{f(x)}{1 - a_k x} = \sum_{k=1}^{p} b_k \prod_{\substack{m=1 \\ m \neq k}}^{p} (1 - a_m x).$$

Protože platí

(34.3.4) $$c(a_k^{-1}) = b_k \prod_{\substack{m=1 \\ m \neq k}}^{p} (1 - a_m a_k^{-1}) \neq 0,$$

můžeme ze znalosti polynomů $f(x)$ a $c(x)$ určit prvky

$$b_k = c(a_k^{-1}) \Big[\prod_{\substack{m=1 \\ m \neq k}}^{p} (1 - a_m a_k^{-1}) \Big]^{-1}.$$

Tím určíme chybové slovo:

(34.3.5) $$e_{i_k} = c(a_k^{-1}) \Big[\prod_{\substack{m=1 \\ m \neq k}}^{p} (1 - a_m a_k^{-1}) \Big]^{-1} a_k^{-j}.$$

34.4. První krok dekódování: určení syndromu. Syndrom $\boldsymbol{s}$ přijatého slova $\boldsymbol{w}$ $(= w_0 + w_1 z + \ldots + w_{n-1} z^{n-1})$ je podle 33.2.3 určen vztahem

$$\boldsymbol{s}^{\mathrm{T}} = \begin{bmatrix} 1 & \alpha^j & \alpha^{2j} & \ldots & \alpha^{(n-1)j} \\ 1 & \alpha^{j+1} & \alpha^{2(j+1)} & \ldots & \alpha^{(n-1)(j+1)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & \alpha^{j+2t-1} & \alpha^{2(j+2t-1)} & \ldots & \alpha^{(n-1)(j+2t-1)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \ldots \\ w_{n-1} \end{bmatrix} =$$

$$= \begin{bmatrix} w(\alpha^j) \\ w(\alpha^{j+1}) \\ \ldots \\ w(\alpha^{j+2t-1}) \end{bmatrix}.$$

Syndrom zapisujeme jako polynom $s(x) = s_0 + s_1 x + \ldots + s_{2t-1} x^{2t-1}$, kde $s_i = w(\alpha^{j+i})$.

34.4.1. Příklad. Při použití ternárního BCH kódu délky 8 s generujícími kořeny 1, α, α^2 a α^3 (33.4.3) dekódujeme slovo

$$\boldsymbol{w} = 00120000 = z^2 + 2z^3.$$

Vypočteme syndrom (v tělese $GF(9)$, viz dodatek A1):

$$s_0 = w(1) = 0\,,$$
$$s_1 = w(\alpha) = 2 = \alpha^4\,,$$
$$s_2 = w(\alpha^2) = 2\alpha = \alpha^5\,,$$
$$s_3 = w(\alpha^3) = [w(\alpha)]^3 = \alpha^4\,.$$

Platí tedy

$$s(x) = \alpha^4 x + \alpha^5 x^2 + \alpha^4 x^3\,.$$

34.4.2. Poznámka. a) Pro hledané prvky a_k a b_k platí

$$s_i = b_1 a_1^i + b_2 a_2^i + \ldots + b_p a_p^i \quad (i = 0, 1, 2, \ldots, 2t - 1)\,.$$

To plyne z faktu, že

$$\boldsymbol{s}^{\mathrm{T}} = \boldsymbol{H}\boldsymbol{w}^{\mathrm{T}} = \boldsymbol{H}\boldsymbol{e}^{\mathrm{T}} = \begin{bmatrix} e(\alpha^j) \\ e(\alpha^{j+1}) \\ \ldots \\ e(\alpha^{j+2t-1}) \end{bmatrix}.$$

Odtud

$$\begin{aligned} s_i &= e(\alpha^{i+j}) = e_{i_1}(\alpha^{i+j})^{i_1} + \ldots + e_{i_p}(\alpha^{i+j})^{i_p} = \\ &= e_{i_1}\alpha^{ji_1}(\alpha^{i_1})^i + \ldots + e_{i_p}\alpha^{ji_p}(\alpha^{i_p})^i = \\ &= b_1 a_1^i + \ldots + b_p a_p^i\,. \end{aligned}$$

b) Pro některé číslo $i = 0, 1, \ldots, t - 1$ platí $s_i \neq 0$, jinými slovy, polynom $s(x) = s_0 + s_1 x + \ldots + s_{2t-1}x^{2t-1}$ není beze zbytku dělitelný polynomem x^t. Kdyby totiž platilo $s_0 = s_1 = \ldots = s_{t-1} = 0$, potom bychom z uvedeného vztahu pro s_i získali tuto soustavu rovnic:

$$\begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ a_1 & a_2 & a_3 & \ldots & a_p \\ a_1^2 & a_2^2 & a_3^2 & \ldots & a_p^2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_1^{p-1} & a_2^{p-1} & a_3^{p-1} & \ldots & a_p^{p-1} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \ldots \\ b_p \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \ldots \\ 0 \end{bmatrix}.$$

To není možné, protože matice soustavy má Vandermondův determinant (31.4), různý od nuly, a prvky b_j jsou nenulové.

34.5. *Druhý krok dekódování*: určení lokátoru a evaluátoru chyb.

a) Je-li syndrom nulový polynom, prohlásíme, že nedošlo k žádné chybě, tj. $\boldsymbol{e} = \boldsymbol{o}$.

b) Pokud $s(x) \neq 0$, provádíme Euklidův algoritmus s polynomy

$$a_0(x) = x^{2t} \quad \text{a} \quad a_1(x) = s(x)\,.$$

Jestliže stupeň všech polynomů $a_k(x)$ je alespoň t, prohlásíme, že došlo k více než t chybám. V opačném případě najdeme první index k takový, že polynom $a_k(x)$ je stupně menšího než t, takže

$$\mathrm{st}\, a_k(x) < t \quad \text{a} \quad \mathrm{st}\, a_{k-1}(x) \geqq t\,.$$

Pokud polynom $u_k(x)$ (viz 34.1.4) splňuje $u_k(0) = 0$, prohlásíme, že došlo k více

než t chybám. V opačném případě položíme $d = [u_k(0)]^{-1}$ a definujeme polynomy

$$f(x) = d\, u_k(x)$$

a

$$c(x) = (-1)^{k+1}\, d\, a_k(x)\,.$$

34.5.1. Věta. *Při t-násobné chybě předchozí postup správně určuje lokátor chyb $f(x)$ i evaluátor chyb $c(x)$.*

Poznámka. Tato věta zaručuje, že i naše hlášení o velkém počtu chyb jsou správná: není-li možné předchozím postupem určit $f(x)$ a $c(x)$, pak došlo k více než t chybám.

Důkaz. Označme $f(x)$ a $c(x)$ skutečný lokátor a evaluátor chyb. Dokážeme, že v Euklidově algoritmu

a) existuje polynom $a_k(x)$ stupně $< t$;

b) pro nejnižší index k takový, že $\operatorname{st} a_k(x) < t$ existuje konstanta d taková, že

$$f(x) = d\, u_k(x) \quad \text{a} \quad c(x) = (-1)^{k+1}\, d\, a_k(x)\,.$$

Z definice polynomu $f(x)$ (34.3.2) plyne $f(0) = 1$, takže potom $d\, u_k(0) = 1$, a tedy $u_k(0) \neq 0$ a

$$d = [u_k(0)]^{-1}\,.$$

Tím dokážeme, že polynomy $f(x)$ a $c(x)$ jsou totožné s těmi, které určil předchozí postup.

a) Volme za $a_k(x)$ poslední polynom Euklidova algoritmu, tj. největší společný dělitel polynomů $s(x)$ a x^{2t}. Potom $a_k(x) = x^i$ pro některé i (protože polynom x^{2t} nemá jiné dělitele) a podle bodu b) v 34.4.2 platí $i < t$.

b) Označme k ten index, pro který platí

$$\operatorname{st} a_k(x) < t \quad \text{a} \quad \operatorname{st} a_{k-1}(x) \geqq t\,.$$

Dokážeme, že platí $f(x) = d\, u_k(x)$ a $c(x) = (-1)^{k+1}\, d\, a_k(x)$ (pro některý prvek d). Nejprve ověříme některé společné vlastnosti polynomů $f(x)$ a $\hat{f}(x) = u_k(x)$ a polynomů $c(x)$ a $\hat{c}(x) = (-1)^{k+1}\, a_k(x)$.

1. Polynomy $f(x)$ a $\hat{f}(x)$ mají stupeň nejvýše t. Platí $\operatorname{st} f(x) = p \leqq t$ (viz (34.3.2)). Podle 34.2.4 je

$$\operatorname{st} u_k(x) = 2t - \operatorname{st} a_{k-1}(x) \leqq 2t - t = t\,.$$

Polynomy $c(x)$ a $\hat{c}(x)$ mají stupeň menší než t. Platí $\operatorname{st} c(x) = p - 1 \leqq t - 1$ (viz (34.3.3)) a $\operatorname{st} a_k(x) < t$.

2. Polynomy $c(x)$ a $f(x)\, s(x)$ mají stejné koeficienty u mocnin x^i pro $i < 2t$; také polynomy $\hat{c}(x)$ a $\hat{f}(x)\, s(x)$ mají stejné koeficienty u těchto mocnin. Jinými slovy, existují polynomy $h(x)$ a $\hat{h}(x)$ takové, že

$$c(x) = f(x)\, s(x) + x^{2t}\, h(x) \tag{34.5.1}$$

a

$$\hat{c}(x) = \hat{f}(x)\, s(x) + x^{2t}\, \hat{h}(x)\,. \tag{34.5.2}$$

Druhá rovnost plyne z (34.1.1): protože $\hat{c}(x) = (-1)^{k+1} a_k(x)$, platí

(34.5.3) $\quad \hat{c}(x) = -[v_k(x)\, x^{2t} - u_k(x)\, s(x)] = \hat{f}(x)\, s(x) + x^{2t}(-v_k(x))$.

První tvrzení ověříme pomocí vzorce pro geometrickou řadu (32.1.3): Podle definice evaluátoru (34.3.3) je

$$c(x) = \sum_{k=1}^{p} b_k \frac{f(x)}{1 - a_k x} =$$

$$= \sum_{k=1}^{p} b_k \sum_{m=0}^{\infty} f(x)\,(a_k x)^m =$$

$$= f(x) \sum_{m=0}^{\infty} \Big(\sum_{k=1}^{p} b_k a_k^m \Big) x^m =$$

$$= f(x) \sum_{m=0}^{\infty} (b_1 a_1^m + b_2 a_2^m + \ldots + b_p a_p^m)\, x^m .$$

Porovnáním s 34.4.2 vidíme, že mocninná řada má stejné koeficienty pro $m = 0, 1, \ldots, 2t - 1$ jako polynom $s(x)$. Proto mají polynomy $c(x)$ a $f(x)\, s(x)$ stejné koeficienty u mocnin x^i pro $i < 2t$.

3. Platí

$$f(x)\, \hat{c}(x) = \hat{f}(x)\, c(x) .$$

Polynomy na obou stranách jsou stupně menšího než $2t$, viz 1. Stačí tedy ověřit, že každý z nich má stejné koeficienty u mocnin x^i, $i < 2t$, jako polynom $f(x)\, \hat{f}(x)\, s(x)$. Vynásobme rovnici (34.5.2) členem $f(x)$:

$$f(x)\, \hat{c}(x) = f(x)\, \hat{f}(x)\, s(x) + x^{2t}\, \hat{h}(x)\, f(x) .$$

Podobně vynásobením rovnice (34.5.1) členem $\hat{f}(x)$ dostáváme

$$\hat{f}(x)\, c(x) = \hat{f}(x) f(x)\, s(x) + x^{2t}\, h(x) \hat{f}(x) .$$

Odtud

$$f(x)\, \hat{c}(x) = \hat{f}(x)\, c(x) + x^{2t}[\hat{h}(x) f(x) - h(x) \hat{f}(x)] .$$

4. Ověříme, že existuje polynom $d(x)$ takový, že

$$\hat{f}(x) = d(x) f(x) \quad \text{a} \quad \hat{c}(x) = d(x)\, c(x) .$$

Nejdříve si všimněme, že polynom $f(x)$ je součinem kořenových činitelů prvků a_i^{-1} pro $i = 1, 2, \ldots, p$ (viz 34.3.2)) a žádný z těchto prvků není kořenem polynomu $c(x)$, podle (34.3.4). Podle 3. bodu je ovšem

$$\hat{f}(a_i^{-1})\, c(a_i^{-1}) = f(a_i^{-1})\, \hat{c}(a_i^{-1}) = 0 ,$$

takže

$$\hat{f}(a_i^{-1}) = 0 \quad \text{pro} \quad i = 1, 2, \ldots, p .$$

Odtud plyne, že polynom $\hat{f}(x)$ je dělitelný polynomem $f(x)$ (23.3). Označme $d(x)$ jejich podíl, potom $\hat{f}(x) = d(x) f(x)$. Podle 3. bodu dále

$$f(x)\, \hat{c}(x) = d(x) f(x)\, c(x) ,$$

takže po zkrácení členem $f(x)$ (20.5.3) dostáváme $\hat{c}(x) = d(x)\, c(x)$.

5. Dokončení důkazu. Stačí nyní ověřit, že polynom $d(x)$ je stupně 0, tedy $d(x) = d_0 \neq 0$ (a položit $d = d_0^{-1}$). Protože polynomy $u_k(x)$ a $v_k(x)$ jsou nesoudělné (34.2.3), stačí dokázat, že jsou oba dělitelné polynomem $d(x)$.

Platí $u_k(x) = \hat{f}(x) = d(x) f(x)$, takže polynom $u_k(x)$ je dělitelný polynomem $d(x)$.

Z rovnosti (34.5.3) dostáváme

$$x^{2t} v_k(x) = \hat{f}(x) s(x) - \hat{c}(x) .$$

Podle předchozích částí důkazu platí

$$\begin{aligned} \hat{c}(x) &= d(x) c(x) = \\ &= d(x) (f(x) s(x) + x^{2t} h(x)) = \\ &= \hat{f}(x) s(x) + x^{2t} h(x) d(x) , \end{aligned}$$

takže

$$x^{2t} v_k(x) = \hat{f}(x) s(x) - \hat{f}(x) s(x) - x^{2t} h(x) d(x) = x^{2t}(-h(x)) d(x) .$$

Zkrácením polynomem x^{2t} dostáváme $v_k(x) = -h(x) d(x)$, takže polynom $v_k(x)$ je dělitelný polynomem $d(x)$.

34.5.2. Příklad (pokračování z 34.4.1). Lokátor a evaluátor chyb najdeme prováděním Euklidova algoritmu s polynomy

$$a_0(x) = x^4 \quad \text{a} \quad a_1(x) = \alpha^4 x + \alpha^5 x^2 + \alpha^4 x^3 .$$

Postupujeme tak dlouho, až najdeme polynom $a_k(x)$ stupně <2.

1. dělení:

$$\begin{array}{l} x^4 : (\alpha^4 x^3 + \alpha^5 x^2 + \alpha^4 x) = \alpha^4 x + \alpha \\ \underline{-(x^4 + \alpha x^3 + x^2)} \\ \qquad \alpha^5 x^3 + \alpha^4 x^2 \\ \quad \underline{-(\alpha^5 x^3 + \alpha^6 x^2 + \alpha^5 x)} \\ \qquad\qquad \alpha^5 x^2 + \alpha x . \end{array}$$

To znamená, že platí

$$\begin{aligned} a_2(x) &= \alpha^5 x^2 + \alpha x , \\ q_1(x) &= \alpha^4 x + \alpha , \\ u_2(x) &= q_1(x) = \alpha^4 x + \alpha . \end{aligned}$$

2. dělení:

$$\begin{array}{l} (\alpha^4 x^3 + \alpha^5 x^2 + \alpha^4 x) : (\alpha^5 x^2 + \alpha x) = \alpha^7 x + \alpha^6 \\ \underline{-(\alpha^4 x^3 + \quad x^2)} \\ \qquad\quad \alpha^3 x^2 + \alpha^4 x \\ \qquad \underline{-(\alpha^3 x^2 + \alpha^7 x)} \\ \qquad\qquad\quad \alpha^2 x . \end{array}$$

To znamená, že platí

$$\begin{aligned} a_3(x) &= \alpha^2 x , \\ q_2(x) &= \alpha^7 x + \alpha^6 , \\ u_3(x) &= q_2(x) u_2(x) + 1 = \alpha^3 x^2 + \alpha^3 x + \alpha^6 . \end{aligned}$$

Protože $a_3(x)$ je první polynom stupně <2, Euklidův algoritmus uzavřeme. Položíme

$$d = [u_3(0)]^{-1} = \alpha^{-6} ,$$

a tedy lokátorem chyb je polynom $f(x) = \alpha^{-6} u_3(x) = \alpha^5 x^2 + \alpha^5 x + 1$ a evaluátorem chyb je polynom $c(x) = \alpha^{-6}(-1)^4 a_3(x) = \alpha^4 x$.

34.6. *Třetí krok dekódování*: provedení opravy

Nejdříve určíme všechny kořeny lokátoru chyb $f(x)$ postupným dosazováním do rovnice $f(\alpha^{-i}) = 0\,(i = 0, 1, 2, \ldots)$. Když jsme našli všechny kořeny $\alpha^{-i_1}, \alpha^{-i_2}, \ldots$ $\ldots, \alpha^{-i_p}$, víme, že opravit máme složky $w_{i_2}, w_{i_2}, \ldots, w_{i_p}$ přijatého slova $\mathbf{w}$. Položme pro všechna $k = 1, 2, \ldots, p$

$$a_k = \alpha^{i_k}$$

a

$$e_{i_k} = c(a_k^{-1}) \prod_{\substack{m=1 \\ m \neq k}}^{p} [(1 - a_m a_k^{-1})]^{-1} a_k^{-j},$$

viz (34.3.5). Opravu znaku w_{i_k} provedeme odečtením znaku e_{i_k}. Tak vznikne kódové slovo $\mathbf{w} - \mathbf{e}$.

34.6.1. Příklad (dokončení z 34.5.2). Lokátor chyb $\alpha^5 x^2 + \alpha^5 x + 1$ má kořeny

$$\alpha^{-3} \text{ a } \alpha^{-2},$$

takže slovo 00120000 má chybné složky w_2 a w_3. Položíme

$$a_1 = \alpha^3 \text{ a } a_2 = \alpha^2.$$

V našem případě je $j = 0$, takže

$$e_3 = c(\alpha^{-3})(1 - \alpha^2\alpha^{-3})^{-1} a_1^0 = \alpha^4\alpha^{-3}(1 - \alpha^{-1})^{-1} = \alpha^4 = 2$$

a

$$e_2 = c(\alpha^{-2})(1 - \alpha^3\alpha^{-2})^{-1} a_2^0 = \alpha^4\alpha^{-2}(1 - \alpha)^{-1} = 1.$$

Opravíme složku $w_3 = 2$ odečtením prvku $e_3 = 2$ a složku $w_2 = 1$ odečtením prvku $e_2 = 1$. Výsledné slovo je

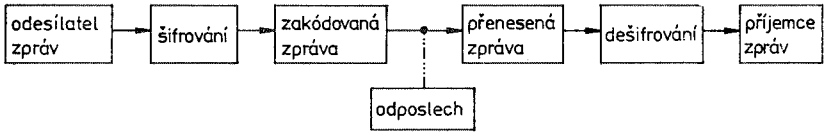
00000000.

To je určitě kódové slovo, takže pokud došlo k dvojnásobné chybě, dekódovali jsme správně.

VIII. KÓDOVÁNÍ TAJNÝCH ZPRÁV

Poslední kapitolu věnujeme kódům, zaměřeným na utajení informace. Některé takové kódy nejsou asi veřejnosti přístupné. Kódy, které budeme probírat, však byly publikovány v matematických časopisech. Přitom je utajení informace velmi účinné.

Při šifrování, tj. kódování za účelem utajení informace, předpokládáme, že kromě odesílatele a příjemce zpráv je na přenos napojen ještě odposlech (obr. 26). Nebudeme se zde zabývat šumem, protože šifrování můžeme samozřejmě kombinovat s bezpečnostním kódováním, abychom šum odstranili.



Obr. 26

35. Nekvalitní odposlech

Nejlépe se bojuje proti odposlechu, který je mnohem méně kvalitní než plánovaný přenos zpráv. V tomto článku budeme pro jednoduchost předpokládat, že zprávy vysíláme kanálem bez šumu, zatímco odposlech je zatížen určitým šumem. Pracujeme s binárními kódy.

35.1. Šifrování pomocí celkové kontroly parity. Myšlenka šifrování je prostá: místo znaku 0 vyšleme náhodné binární slovo sudé parity a místo znaku 1 náhodné slovo liché parity. Například volíme-li délku šifrovacích slov 5, potom zprávu 011 zašifrujeme třeba takto:

1001001101011111 .

Dešifrování je snadné: zjistíme paritu prvních pěti znaků, potom dalších pěti znaků atd. Naproti tomu šum, kterým je zatížen odposlech, výrazně sníží množství informace, která je odposlechnuta. Je-li např. spolehlivost odposlechu 90%, tj. jedna desetina znaků se při odposlechu zkreslí, potom pravděpodobnost, že v pětiznakovém slově byl zkreslen lichý počet znaků je větší než 1/3. To znamená, že víc než třetina odposlechnutých znaků je dekódována chybně.

Jestliže očekáváme malý šum odposlechu, volíme delší slova. Například při šifrování slovy délky 31 a při odposlechu se spolehlivostí 98% opět více než třetina znaků bude odposlechnuta chybně.

Nevýhodou této metody je její malá efektivnost. Například v posledním případě na 31 šifrovacích znaků připadá jediný informační znak. Naštěstí lze efektivitu zvýšit použitím vhodného lineárního kódu.

35.2. Šifrování pomocí lineárního kódu. K šifrování zpráv můžeme použít libovolný binární lineární (n, k)-kód K. Označme $\boldsymbol{H}$ kontrolní matici. Každé binární slovo $\boldsymbol{w}$ délky n má syndrom (12.4) délky $n - k$ (= počet řádků matice $\boldsymbol{H}$). Zvolme reprezentanty tříd kódu (13.3) $\mathbf{e}_1, \ldots, \mathbf{e}_m$, kde $m = 2^{n-k}$, potom každé slovo $\boldsymbol{s}$ délky $n - k$ je syndromem některého reprezentanta, tj. $\boldsymbol{s}^{\mathrm{T}} = \boldsymbol{H}\mathbf{e}_i^{\mathrm{T}}$.

Zprávu šifrujeme po skupinách $n - k$ znaků tak, že pro každé slovo $\boldsymbol{s}$ délky $n - k$ určíme reprezentanta $\mathbf{e}_i$ se syndromem $\boldsymbol{s}$ a potom vyšleme náhodně vybrané slovo třídy $\mathbf{e}_i + K$. Dešifrování (bez šumu) je snadné: pro každé přijaté slovo délky n zjistíme jeho syndrom délky $n - k$. Informační poměr je $(n - k)/n$.

35.3. Příklad: Hammingův (7, 4)-kód (13.13). Chceme zašifrovat zprávu

01011011011 ... .

Šifrujeme po skupinách $7 - 4 = 3$ znaků. První skupina 010 je syndromem reprezentanta $\mathbf{e}_2 = 0100000$ (viz 13.13). Vyšleme tedy libovolné ze $2^4 = 16$ slov třídy $\mathbf{e}_2 + K$. Například v generující matici $\boldsymbol{G}$ (viz 14.5.2) sečteme prvé dva řádky a dostaneme kódové slovo $\boldsymbol{k} = 1000011$, takže vyšleme

$$\mathbf{e}_2 + \boldsymbol{k} = 1100011\,.$$

Dešifrujeme určením syndromu:

$$\boldsymbol{H}\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix},$$

takže vysláno bylo 010. Podobně postupujeme se druhou skupinou 110, atd.

Informační poměr je 3/7. Při odposlechu zatíženém šumem má každé slovo, přijaté s jednoduchou chybou, syndrom odlišný od vyslaného slova. Je-li tedy spolehlivost odposlechu 90%, je většina odposlechnutých trojic chybná.

36. Kvalitní odposlech

36.1. Jednorázový klíč. Předpokládáme nyní, že odposlech je stejně kvalitní jako vlastní přenos. Potom můžeme stále ještě bezpečně šifrovat jednorázovým klíčem, tj. náhodným binárním slovem velké délky n, které předem bezpečně doru-

číme jak odesílateli zpráv, tak adresátovi. Tento klíč lze potom použít pro jeden přenos zprávy délky n: Je-li $\boldsymbol{k} = k_1 k_2 \ldots k_n$ klíč a $\boldsymbol{a} = a_1 a_2 \ldots a_n$ je zdrojová zpráva, vyšleme součet

$$\boldsymbol{a} + \boldsymbol{k} = (a_1 + k_1)(a_2 + k_2) \ldots (a_n + k_n) .$$

Dešifrujeme odečtením klíče: $(\boldsymbol{a} + \boldsymbol{k}) - \boldsymbol{k} = \boldsymbol{a}$. Jestliže však někdo odposlechne zprávu $\boldsymbol{a} + \boldsymbol{k}$ a přitom nezná klíč $\boldsymbol{k}$, nedozvěděl se nic: Klíčem může být totiž libovolné z 2^n slov a potom (pro dané slovo $\boldsymbol{a}$) také součet $\boldsymbol{a} + \boldsymbol{k}$ může být libovolné z 2^n slov.

Nevýhodou tohoto jinak skvělého postupu je, že klíč smíme použít jen jednou. U dvou zpráv $\boldsymbol{a} + \boldsymbol{k}$ a $\boldsymbol{b} + \boldsymbol{k}$ by totiž odposlech zjistil hodnotu $(\boldsymbol{a} + \boldsymbol{k}) + (\boldsymbol{b} + \boldsymbol{k}) = \boldsymbol{a} + \boldsymbol{b}$, a to už je cenná informace. Přitom musíme klíč dodat mimo přenosový kanál, aby nemohl být odposlechnut, a délka klíče musí být větší nebo rovna délce vysílané zprávy. Proto se snažíme najít postup, který by ze „zárodku klíče" vytvořil klíč velký. Přesněji, z náhodného binárního slova $\boldsymbol{k}_0$ chceme vytvořit mnohem větší slovo $\boldsymbol{k}$ tak, aby

a) $\boldsymbol{k}$ bylo slovo téměř náhodné a

b) odesílatel i příjemce se shodli na způsobu vytváření.

Náhodným slovem se při tom rozumí slovo, které např. vytvoříme při házení mince (geometricky dokonalé) a zápisu 0 = panna, 1 = orel. Slova s vlastnostmi podobnými náhodným slovům se nazývají *pseudonáhodná*. Ukážeme, jak pomocí Reedových-Mullerových kódů lze z náhodného slova délky m vytvořit pseudonáhodné slovo délky $2^m - 1$. Tedy např. zárodek klíče délky $m = 10$ dává pseudonáhodný klíč délky $> 1\,000$.

36.2. Pseudonáhodná slova kódu $\mathscr{R}(1, m)^*$. Slova zúženého Reedova-Mullerova kódu $R(1, m)^*$ (viz 18.11) jsou pseudonáhodná. Budeme pro zjednodušení pracovat se simplexovým kódem $\mathscr{S}_m$, jehož zvětšením je kód $R(1, m)^*$ (18.12). Stačí jistě ukázat pseudonáhodný charakter slov v kódu $\mathscr{S}_m$ — potom i „opačné" slovo $1111 \ldots 11 + \boldsymbol{v}$ je pseudonáhodné (a celý kód $R(1, m)^*$ je tvořen slovy $\boldsymbol{v}$ a $\boldsymbol{1} + \boldsymbol{v}$ pro $\boldsymbol{v} \in \mathscr{S}_m$). *Simplexový kód* $\mathscr{S}_m$ je podle definice duálním kódem k Hammingovu kódu. To znamená, podle 28.3.1, že libovolný ireducibilní polynom $h(x)$ stupně m (nad tělesem Z_2) definuje simplexový kód $\mathscr{S}_m$ tak, že $h(x)$ je jeho kontrolní polynom. Označme $h(x) = h_0 + h_1 x + \ldots + h_m x^m$. Potom simplexový kód $\mathscr{S}_m$ tedy sestává ze všech slov $\boldsymbol{v} = v_0 v_1 \ldots v_{n-1}$, kde $n = 2^m - 1$, pro která platí

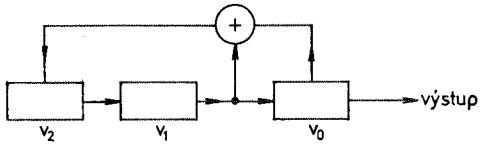
$$(36.2.1) \qquad h_0 v_i + h_1 v_{i+1} + \ldots + h_m v_{i+m} = 0 \quad \text{pro} \quad 0, 1, \ldots, n - 1$$

(kde sčítání v indexech je modulo n). Simplexový kód je (n, m)-kód.

36.2.1. Příklad: Simplexový kód $\mathscr{S}_3$ a Reedův-Mullerův kód $R(1, 3)^*$. Polynom $h(x) = x^3 + x + 1$ je ireducibilní nad tělesem Z_2. Definuje simplexový kód všech slov $v_0 v_1 v_2 v_3 v_4 v_5 v_6$, pro která platí rekurzívní vztah $v_i + v_{i+1} + v_{i+3} = 0$, tj.

$$(36.2.2) \qquad v_{i+3} = v_i + v_{i+1} .$$

Tento kód lze realizovat číslicovým obvodem o třech registrech a binární sčítačce (obr. 27).



Obr. 27

Při libovolných počátečních hodnotách $v_0v_1v_2$ (zprava doleva) vznikne na výstupu nekonečné slovo, splňující rekurzívní vztah (36.2.2). To znamená, že každý segment délky 7 je kódovým slovem kódu $\mathscr{S}_3$: Volíme-li 000, dostaneme ovšem nulové slovo. Jestliže však zvolíme libovolné nenulové slovo $v_0v_1v_2 \neq 000$, vznikne periodické slovo s periodou 7. Protože 7 je počet všech nenulových slov kódu $\mathscr{S}_3$ (což je (7, 3)-kód s počtem slov $2^3 = 8$), vidíme, že celý kód $\mathscr{S}_3$, kromě nulového slova, získáme jako segmenty nekonečného výstupního slova. Například pro $v_0v_1v_2 = 101$ je výstup

10111001011100 1011100...

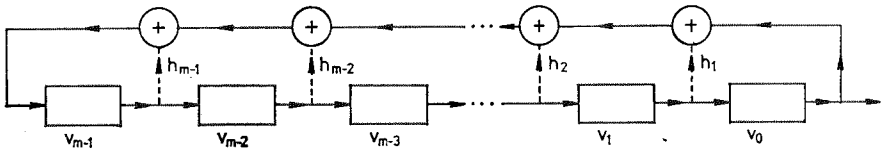
a segmenty délky 7 od prvního znaku, druhého znaku, atd., tvoří celou množinu $\mathscr{S}_3 - \{\mathbf{o}\}$.

KÓD $R(1, 3)$*

KÓD $\mathscr{S}_3$	
0 0 0 0 0 0 0	1 1 1 1 1 1 1
1 0 1 1 1 0 0	0 1 0 0 0 1 1
0 1 0 1 1 1 0	1 0 1 0 0 0 1
0 0 1 0 1 1 1	1 1 0 1 0 0 0
1 0 0 1 0 1 1	0 1 1 0 1 0 0
1 1 0 0 1 0 1	0 0 1 1 0 1 0
1 1 1 0 0 1 0	0 0 0 1 1 0 1
0 1 1 1 0 0 1	1 0 0 0 1 1 0

Ze zárodku klíče, např. 101, vytvoříme klíč 1011100.

36.2.2. Vytváření simplexového kódu $\mathscr{S}_m$. Zvolme ireducibilní polynom $h(x) = h_0 + h_1x + \ldots + h_mx^m$ stupně m nad tělesem Z_2. Simplexový kód $\mathscr{S}_m$ lze vytvořit číslicovým obvodem (obr. 28).



Obr. 28

Čárkovaný spoj s označením h_i je obyčejný spoj pro $h_i = 1$ a naopak spoj chybí pro $h_i = 0$. (Poslední spoj vpravo není čárkovaný, protože platí $h_0 = 1$. Jinak by měl totiž polynom $h(x)$ kořen 0 a nebyl by ireducibilní, viz 23.2). Při libovolném počátečním obsahu registrů $v_0 v_1 \ldots v_{m-1}$ splňuje (nekonečné) výstupní slovo $v_0 v_1 v_2 \ldots$ rekurzi

$$v_{i+m} = v_{i+m-1} h_{m-1} + v_{i+m-2} h_{m-2} + \ldots + v_{i+1} h_1 + v_i .$$

(Viz (36.2.1).) Z toho plyne, že každý segment délky n je kódovým slovem kódu $\mathscr{S}_m$. Navíc má výstupní posloupnost při libovolné volbě nenulového slova $v_0 v_1 \ldots v_{m-1}$ tyto vlastnosti:

a) Perioda výstupního slova je n (tj. číslo $2^m - 1$, podstatně větší než číslo m). To znamená, že segmenty délky n postupně nabývají všech hodnot nenulových slov kódu $\mathscr{S}_m$ dříve, než se začnou opakovat.

b) Každý segment délky n obsahuje v podstatě stejný počet nul a jedniček: počet nul je $(n - 1)/2$ a počet jedniček je $(n + 1)/2$.

c) Autokorelační funkce má nízké hodnoty. Pro každou binární posloupnost s periodou n se definuje *autokorelační funkce* předpisem

$$\varrho(t) = \frac{1}{n} \sum_{i=0}^{n-1} (-1)^{v_i} (-1)^{v_{i+t}} \quad (t = 0, \pm 1, \pm 2, \ldots) .$$

Pro výstupní posloupnosti v uvedeném obvodu je autokorelační funkce tato:

$$\varrho(0) = 1 \quad \text{a} \quad \varrho(t) = -1/n \quad \text{pro} \quad t = 1, 2, \ldots, (n - 1) .$$

Vidíme, že hodnoty $\varrho(1), \ldots, \varrho(n - 1)$ jsou velmi malé.

Tyto vlastnosti, které uvádíme bez důkazu, nás opravňují k tomu označit výstupní posloupnost za pseudonáhodnou. Jestliže se tedy odesílatel i příjemce zpráv dohodnou na společném polynomu $h(x)$ stupně m, potom se zárodku klíče délky m mohou oba současně vytvářet klíč délky $2^m - 1$.

37. Šifrování s veřejně přístupným klíčem

Skvělou metodu šifrování navrhli v roce 1977 Rivest, Shamir a Adelman. Při této metodě se zveřejní způsob, jak šifrovat zprávy, a přesto může dešifrovat jen adresát. Metoda je založena na teorii čísel a zejména na dvou jejích rysech: a) existuje rychlý algoritmus k nalezení náhodného pročísla a b) neexistuje (přesněji: není dosud znám) rychlý algoritmus rozkladu velkého čísla v součin prvočísel.

Později byly nalezeny další metody s veřejným šifrováním a my se ještě zmíníme o metodě, založené na zavazadlovém problému (knapsack problem), jejímiž autory jsou Merkle a Hellman.

37.1. Metoda založená na velkých prvočíslech. Zvolíme velké číslo n, které je součinem prvočísel p a q, a za zdrojovou abecedu bereme čísla $0, 1, \ldots, n - 1$ (případně jejich binární rozvoje, abychom zůstali v rámci binárního kódování).

Číslo n zveřejníme, ale čísla p a q utajíme. Dále zvolíme čísla i (pro šifrování) a j (pro dešifrování) tak, aby jejich součin ij měl zbytek 1 po dělení číslem

$$\varphi(n) = (p - 1)(q - 1) .$$

Platí tedy

(37.1.1) $$ij \equiv 1 \pmod{\varphi(n)} .$$

Číslo i zveřejníme, číslo j utajíme.

Šifrujeme takto: zdrojový znak $x = 0, 1, \ldots, n - 1$ umocníme na i-tou a zredukujeme modulo n, tedy najdeme zbytek $y = 0, 1, \ldots, n - 1$ po dělení mocniny x^i číslem n:

(37.1.2) $$y \equiv x^i \pmod{n} .$$

Vyšleme číslo y. Dešifrování je analogické: přijatý znak y umocníme na j-tou a zredukujeme modulo n. Dokážeme, že platí

(37.1.3) $$x \equiv y^j \pmod{n} ,$$

a to znamená, že vyslaný znak x správně dekódujeme.

Síla metody spočívá v tom, že po zveřejnění čísel n a i ví každý, kdo nám chce poslat zprávu, jak ji zašifrovat. Ovšem číslo j utajíme (a čísla p a q také), takže zprávu nemůže nikdo jiný dešifrovat. Teď se může čtenář pozastavit nad tím, že rovnice $ij \equiv 1 \pmod{\varphi(n)}$ dává snadný návod, jak pomocí čísel n a i určit číslo j. Dává, ale k tomu musíme znát číslo $\varphi(n) = (p - 1)(q - 1)$, které nelze zjistit jinak než rozkladem čísla n, tj. nalezením prvočísel p a q (viz úlohu 37.4.2). Jestliže zvolíme náhodná prvočísla p a q, např. s padesáti ciframi (a to je za použití dnešní výpočetní techniky možné), bude n číslo 100-ciferné. V dnešní době neexistuje metoda, která by v rozumném čase určila prvočíselný rozklad náhodného 100 ciferného čísla.

Uvedeme příklad s malými čísly, který ilustruje šifrovací metodu, ale je ovšem nepoužitelný pro skutečné šifrování (kvůli snadnému rozkladu čísla n).

37.2. Příklad. Šifrujeme binární zprávy tak, že je čteme vždy po pěticích znaků a každou pětici interpretujeme jako binárně zapsané číslo $0, 1, \ldots, 31 (= 2^5 - 1)$. V uvedené metodě zvolíme $n = 33 = 3 . 11$ (takže znak 32 zůstane nevyužit). Je tedy $\varphi(n) = 2 . 10 = 20$. Zvolme $i = 3$ a $j = 7$. Platí $21 \equiv 1 \pmod{20}$. Šifrujeme tedy třetí mocninou a dešifrujeme sedmou.

Chceme např. zašifrovat zprávu

1111000101 .

První pětice 11110 je binární zápis čísla $x = 30$. Platí

$$x^3 = 27\,000 = 33 . 818 + 6 ,$$

takže $y = 6$. Vyšleme tedy binární zápis čísla 6 : 00110. Druhá pětice 00101 je binární zápis čísla $x = 5$ a platí

$$x^3 = 125 = 33 . 3 + 26 ,$$

takže $y = 26$ neboli 11010. Zašifrovaná zpráva zní takto:

0011011010 .

Při dešifrování bereme první pětici, tedy $y = 6$: platí

$$y^7 = 279\,936 = 33 \,.\, 8\,482 + 30 ,$$

takže správně dekódujeme 30 (neboli 11110). Pro druhou pětici podobně ověříme $26^7 \equiv 5 \pmod{33}$, zde s trochu delším výpočtem.

37.3. Důkaz správnosti metody. Naším úkolem je ověřit, že rovnice (37.1.3) je skutečně důsledkem rovnic (37.1.1) a (37.1.2). Připomeňme, že $\varphi(n) = (p - 1)(q - 1)$ je hodnota Eulerovy funkce (25.8.1). Protože $ij \equiv 1 \pmod{\varphi(n)}$, existuje celé číslo r takové, že $ij = 1 + r\,\varphi(n)$. Platí $y \equiv x^i \pmod{n}$, a tedy $y^j \equiv x^{ij} \pmod{n}$, takže naším úkolem je dokázat rovnost

$$x \equiv x^{1+r\varphi(n)} \pmod{n} .$$

Pokud jsou čísla x a n nesoudělná, můžeme použít Fermatovu-Eulerovu větu (25.8.2): platí $x^{\varphi(n)} \equiv 1 \pmod{n}$, a tedy $x^{r\varphi(n)} = (x^{\varphi(n)})^r \equiv 1^r \pmod{n}$, takže

$$x^{1+r\varphi(n)} = x\, x^{r\varphi(n)} \equiv x \pmod{n} .$$

Pokud jsou čísla x a n $(= pq$, kde p a q jsou prvočísla) soudělná, potom číslo x je buď tvaru $x = pa$, nebo tvaru $x = aq$, kde a je celé číslo. Předpokládejme, že $x = pa$ a dále $a \neq 0$. (Pro $a = 0$ máme totiž $x = 0 = x^{1+r\varphi(n)}$.) Potom $a = 1, 2, \ldots, q - 1$ je číslo nesoudělné s číslem q, a tedy i x je číslo nesoudělné s číslem q, takže platí $x^{\varphi(q)} \equiv 1 \pmod{q}$. Přitom $\varphi(q) = q - 1$ (viz 25.8), takže $x^{q-1} \equiv 1 \pmod{q}$ a odtud plyne

$$x^{r\varphi(n)} = x^{r(p-1)(q-1)} = (x^{q-1})^{r(p-1)} \equiv 1^{r(p-1)} = 1 \pmod{q} .$$

To znamená. že platí $x^{r\varphi(n)} = 1 + sq$, kde s je celé číslo. Vynásobíme poslední rovnost číslem x:

$$x^{r\varphi(n)+1} = x + sqx ,$$

a protože číslo $n = pq$ je dělitelem čísla $sqx = sqpa$, plyne odtud

$$x^{r\varphi(n)+1} \equiv x \pmod{n} .$$

37.4. Provádění metody. Jestliže jsme určili čísla $n = pq$ a čísla i a j, je šifrování i dešifrování zpráv otázkou umocnění daného čísla a provedení redukce modulo n. To lze provádět na počítači dost rychle i v případě nesmírně velkých hodnot n, řekněme 100-ciferných. Síla metody spočívá v tom, že ani po zveřejnění 100-ciferného čísla n není v dnešní době znám způsob, jak určit jeho prvočíselný rozklad. Nepomůže ani to, že víme, kolik členů tento rozklad má, pokud prvočísla p a q zvolíme náhodně. Jak ale takovou náhodnou volbu provést?

Z teorie čísel plyne, že pokud umíme zjišťovat prvočíselnost i velkých (řekněme 50-ciferných) čísel, můžeme najít náhodné prvočíslo metodou postupné volby. To znamená, že vytvoříme náhodné 50-ciferné číslo a zjistíme, zda je prvočíslem. Pokud ne, vytvoříme další náhodné číslo, atd. Průměrný počet voleb, potřebný k nalezení prvočísla, je

$$\tfrac{1}{2} \ln 10^{50} \doteq 58 .$$

Stačí tedy najít tak rychlý postup vyšetření prvočíselnosti 50-ciferných čísel, abychom jej byli s to mnohokrát (řádově ve stovkách) opakovat. K tomu slouží heuristické metody, které ověří v několika desítkách sekund, že pravděpodobnost prvočíselnosti je téměř 1. Uvedeme zde takovou metodu, tzv. Rabinův algoritmus (podrobněji viz Kučera, L.: Kombinatorické algoritmy. Praha, SNTL 1983).

Buď n prvočíslo. Podle Fermatovy-Eulerovy věty (25.8.2) platí pro všechna čísla $i = 2, 3, \ldots, n-1$

$$i^{n-1} \equiv 1 \pmod{n}.$$

Naopak, není-li n prvočíslo, máme rozklad $n = ij$, kde $i \neq 1 \neq j$, a potom $i^{n-1} \not\equiv 1 \pmod{n}$. (Kdyby totiž rozdíl $i^{n-1} - 1$ byl dělitelný číslem $n = ij$, byl by dělitelný i číslem i, a to je spor: zbytek po dělení $(i^{n-1} - 1) : i$ je vždy 1.) Příklady:

$$\begin{aligned}
(n = 3)\quad & 2^2 = 4 \equiv 1 \pmod{3} \\
(n = 5)\quad & 2^4 = 16 \equiv 1 \pmod{5} \\
& 3^4 = 81 \equiv 1 \pmod{5} \\
& 4^4 = 256 \equiv 1 \pmod{5},
\end{aligned}$$

ale

$$(n = 4)\quad 2^3 = 8 \not\equiv 1 \pmod{4}.$$

Abychom ověřili, zda náhodné číslo n je prvočíslem, stačí zjistit, zda platí $i^{n-1} \equiv 1 \pmod{n}$ pro všechna $i = 2, 3, \ldots, n-2$. To ale nelze realizovat v případě velkých čísel n. Proto využijeme další vlastnosti prvočísel n: každé číslo větší než 1, které je s číslem n soudělné, je jeho násobkem. Označme nyní pro každé číslo n symbolem K_n množinu všech celých čísel tvaru $(n-1)/2^k$. (Tato množina je obvykle malá, např. $K_{1001} = \{500, 250, 125\}$. Pro žádné číslo $n \leqq 10\,000$ nemá víc než 13 prvků.) Lze dokázat, že jakmile n není prvočíslo, pak **většina** čísel $i = 2, 3, \ldots, n-1$ buď nesplňuje vztah $i^{n-1} \equiv 1 \pmod{n}$, anebo má mocninu i^k $(k \in K_n)$, která je sice soudělná s číslem n, ale není jeho násobkem. To znamená, že pro většinu čísel $i = 2, 3, \ldots, n-1$ platí

(37.4.1) $i^{n-1} \not\equiv 1 \pmod{n}$ nebo existuje $k \in K_n$ takové, že $i^k \not\equiv 0 \pmod{n}$, a přitom je číslo i^k s číslem n soudělné.

Odtud plyne tento postup testování prvočíselnosti čísla n:

1. Zvolme náhodně číslo $i = 2, 3, \ldots, n-1$. Ověřme, zda platí (37.4.1). Pokud ano, vidíme, že n není prvočíslo. Pokud ne, je více než 50% naděje, že n je prvočíslo.

2. Zvolme nezávisle další náhodné číslo i a ověřme (37.4.1). Pokud (37.4.1) platí, není n prvočíslo. Pokud opět neplatí, je pravděpodobnost, že n je prvočíslem, větší než $1 - (\frac{1}{2})^2 = 75\%$.

Po deseti nezávislých volbách buď zjistíme, že n není prvočíslo (pokud (*) platí alespoň jednou), nebo máme naději, že n je prvočíslo, větší než $1 - (\frac{1}{2})^{10} > 99{,}9\%$.

Po určení dvou velkých náhodných prvočísel p a q položíme $n = pq$ a zvolíme náhodné číslo $i = 1, 2, \ldots, n$, které je nesoudělné s číslem $\varphi(n)$. K němu potom existuje inverzní prvek v okruhu $Z_{\varphi(n)}$ (9.7.1), tj. číslo j takové, že $ij = 1 \pmod{\varphi(n)}$.

Číslo j snadno určíme ze známých hodnot $\varphi(n)$ a i. Naopak lze ukázat, že pokud známe jen hodnoty n a i, je určení čísla j stejně náročné jako určení prvočíselného rozkladu čísla n (viz 37.4.2).

37.4.1. Závěr. Pomocí počítače lze určit dvě náhodná 50-ciferná čísla p a q, o kterých můžeme s vysokou pravděpodobností očekávat, že jsou to prvočísla. Po zveřejnění čísla $n = pq$ a čísla i, pro které existuje j s vlastností (37.1.1), může kdokoli zašifrovat zprávu. K dešifrování však musí určit číslo $\varphi(n)$ a k tomu potřebuje najít prvočíselný rozklad 100-ciferného čísla n (viz 37.4.2). Dokud se nenajde rychlý algoritmus pro prvočíselný rozklad, je kódovací metoda bezpečná.

37.4.2. Cvičení. Ověřte, že určení čísla $\varphi(n)$ je stejně těžké, jako určení prvočíselného rozkladu čísla n: ze znalosti čísel $n = pq$ a $\varphi(n) = pq - p - q + 1$ určíme součin i součet hledaných hodnot p a q a odtud i tyto hodnoty.

37.5. Metoda založená na zavazadlovém problému

Autory jiné metody šifrování s veřejným klíčem jsou Merkle a Hellman (v r. 1978). Je založena na obtížnosti řešení tzv. *zavazadlového problému* (knapsack problem), který zní takto: máme sbalit n předmětů, jejichž objemy $a_1, a_2, \ldots, a_n$ známe. Přinesli jsme zavazadlo s objemem S a ptáme se, je-li možné toto zavazadlo plně využít. Tedy je-li možné číslo S zapsat jako součet některých z daných čísel. To můžeme formulovat jako hledání binárního slova $\boldsymbol{x} = x_1x_2\ldots x_n$ takového, že platí

$$S = x_1a_1 + x_2a_2 + \ldots + x_na_n = \boldsymbol{x} * \boldsymbol{a}\,.$$

Obtížnost řešení zavazadlového problému ovšem závisí na volbě vektoru čísel $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$. Každá taková volba dává možnost šifrování zpráv: binární zprávy rozdělíme po n-ticích a místo slova $\boldsymbol{x} = x_1x_2\ldots x_n$ vyšleme číslo $S = \boldsymbol{x} * \boldsymbol{a}$. Při dešifrování stojíme před úkolem vyřešit zavazadlový problém dané hodnoty S. Jestliže je zavazadlový problém lehký, nestojí tato metoda za nic — dešifrovat může kdokoli. Jestliže je zavazadlový problém těžký, nestojí metoda také za nic — dešifrovat nemůže nikdo, ani adresát. Merkle a Hellman navrhli způsob, jak lehký problém (který zná jen správný příjemce zpráv) proměnit v těžký (který lze zveřejnit).

Než metodu obecně popíšeme, uvedeme příklad.

37.5.1. Příklad. K zašifrování zpráv délky 6 lze použít tento snadný zavazadlový problém:

$$\boldsymbol{a}' = (32, 16, 8, 4, 2, 1)\,.$$

Pro dané číslo S totiž rovnost $S = \boldsymbol{x} * \boldsymbol{a}'$ znamená, že slovo $x_1x_2x_3x_4x_5x_6$ je binárním rozvojem čísla S. Jestliže např. zašifrovaná zpráva zní $S = 58$, potom původní zpráva je 111010 (tj. 58 binárně).

Nyní problém zesložitíme tím, že každou složku vektoru $\boldsymbol{a}'$ vynásobíme číslem $v = 14$ a zredukujeme modulo číslo $m = 139$. Nový vektor $\boldsymbol{a}$ má tedy složky $a_i =$

$= 0, 1, \ldots, 138$ takové, že $a_i \equiv 14a'_i \pmod{139}$. Například $a_1 \equiv 14 \cdot 32 \pmod{139}$, a protože platí $14 \cdot 32 = 448 = 139 \cdot 3 + 31$, je $a_1 = 31$. Podobně postupujeme pro další složky a dostáváme vektor

$$\boldsymbol{a} = (31, 85, 112, 56, 28, 14).$$

Zprávu $\boldsymbol{x} = x_1x_2x_3x_4x_5x_6$ zašifrujeme číslem $S = \boldsymbol{x} * \boldsymbol{a}$. Dešifrování je už trochu složitější. Uhodnete, která zpráva má šifru $S = 172$?

Naštěstí můžeme tento složitější zavazadlový problém převést na triviální problém $\boldsymbol{a}'$. Číslo $v = 14$ má totiž inverzní prvek modulo 139: pro číslo $w = 10$ platí $vw \equiv 1 \pmod{139}$. Přijatou šifru S vynásobíme číslem 10 a zredukujeme modulo 139. Pro tuto novou hodnotu S' stačí vyřešit zavazadlový problém $\boldsymbol{a}'$ (tj. najít binární rozvoj čísla S'). Například pro $S = 172$ máme $10\ S = 1\,720 = 139 \cdot 12 + 52$, takže $S' = 52$ nebo binárně 11010. Původní zpráva tedy byla $\boldsymbol{x} = 11010$. Zkouška: platí $\boldsymbol{x} * \boldsymbol{a} = 31 + 85 + 56 = 172$.

37.5.2. Poznámka. V předchozím příkladu jsme uvedli snadný zavazadlový problém $\boldsymbol{a}'$. Obecněji platí: *snadný je každý zavazadlový problém* $\boldsymbol{a}' = (a'_1, a'_2, \ldots, a'_n)$, *který splňuje tuto podmínku*:

$$(37.5.1) \qquad a'_n < a'_{n-1},\ a'_n + a'_{n-1} < a'_{n-2}, \ldots, \sum_{i=2}^{n} a'_i < a'_1 .$$

Skutečně, pokud $S = \boldsymbol{x} * \boldsymbol{a}'$, potom slovo $\boldsymbol{x}$ určíme takto:

$x_1 = 1$, právě když $S \geqq a'_1$,
$x_2 = 1$, právě když $S - x_1a'_1 \geqq a'_2$,
$x_3 = 1$, právě když $S - x_1a'_1 - x_2a'_2 \geqq a'_3$,

atd. (Ověřte!)

37.5.3. Popis metody. K šifrování zpráv použijeme obtížný zavazadlový problém $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$, který zveřejníme. Potom se binární slovo $\boldsymbol{x} = x_1x_2 \ldots x_n$ šifruje číslem $S = \boldsymbol{x} * \boldsymbol{a}$. Problém $\boldsymbol{a}$ sestrojíme tak, že nejprve zvolíme libovolný snadný zavazadlový problém $\boldsymbol{a}' = (a'_1, a'_2, \ldots, a'_n)$. (Například libovolný vektor čísel, splňující podmínku (37.5.1).) Dále zvolíme číslo $m > a'_1 + a'_2 + \ldots + a'_n$ a číslo v, které je s číslem m nesoudělné. Zveřejníme vektor $\boldsymbol{a}$, vzniklý násobením vektoru $\boldsymbol{a}'$ číslem v a redukcí modulo m;

$$(37.5.2) \qquad a_i \equiv va'_i \pmod{m} \quad \text{a} \quad a_i = 0, 1, \ldots, m - 1$$

pro všechna $i = 1, 2, \ldots, n$. Čísla m a v utajíme.

Přijatou šifru S dešifrujeme takto: Najdeme inverzní prvek čísla v modulo m, tedy takové číslo w, pro které platí $vw \equiv 1 \pmod{m}$, viz 9.7.1. Tímto číslem násobíme šifru S a zredukujeme modulo m:

$$(37.5.3) \quad S' \equiv wS \pmod{m} \quad \text{a} \quad S' = 0, 1, \ldots, m - 1 .$$

Potom vyřešíme snadný zavazadlový problém, tj. najdeme slovo $\boldsymbol{x}$ takové, že $S' = \boldsymbol{x} * \boldsymbol{a}'$. Zašifrovaná zpráva je $\boldsymbol{x}$.

37.5.4. Důkaz správnosti metody. Naším úkolem je ověřit, že pro každé číslo $S = \boldsymbol{x} * \boldsymbol{a}$ z rovnice $S' = wS \pmod m$ plyne rovnice $S' = \boldsymbol{x} * \boldsymbol{a}'$ (pro $S' = 0, 1, \ldots, m - 1$). Platí

$$\begin{aligned} S' &\equiv w(\boldsymbol{x} * \boldsymbol{a}) \pmod m \equiv \\ &\equiv wv(\boldsymbol{x} * \boldsymbol{a}') \pmod m \equiv \\ &\equiv \boldsymbol{x} * \boldsymbol{a}' \pmod m . \end{aligned}$$

Dále $\boldsymbol{x} * \boldsymbol{a}' \leqq a'_1 + a'_2 + \ldots + a'_n < m$ a také $S' < m$, takže ze vztahu $S' \equiv \equiv \boldsymbol{x} * \boldsymbol{a}' \pmod m$ již plyne $S' = \boldsymbol{x} * \boldsymbol{a}'$.

37.5.5. Zesílení metody. Jak účinná je tato metoda šifrování? Na tuto otázku není úplně jasná odpověď. Záleží na tom, jak obtížný je zveřejněný zavazadlový problém. Pro zesílení účinnosti můžeme metodu iterovat: začneme snadným zavazadlovým problémem $\boldsymbol{a}^0$ a čísly m^0 a v^0. Násobením hodnotou v^0 a redukcí modulo m^0 dostáváme zavazadlový problém $\boldsymbol{a}^1$. Ten již není snadný pro ostatní, ale je snadný pro nás! Proto můžeme vzít $\boldsymbol{a}^1$ za základ metody, zvolit čísla m^1 a v^1 a vytvořit nový zavazadlový problém $\boldsymbol{a}^2$, atd. Čím delší iteraci uděláme, tím více bude mít výsledný zavazadlový problém charakter náhodného vektoru a tím těžší bude jeho řešení.

DODATKY

A1. Galoisova tělesa

Uvádíme tabulky Galoisových těles. Tělesa o p prvcích, kde p je prvočíslo, neuvádíme: jsou to tělesa Z_p (9.6). Uvádíme tělesa

$GF(p^n) = Z_p/q(x)$, p prvočíslo a $n > 1$,

kde $q(x)$ je ireducibilní polynom stupně n nad Z_p (23.15). Každý prvek vyjádříme jako polynom proměnné z stupně $<n$ a zároveň (kromě 0) jako mocninu prvku z. Tím umožňujeme snadné sčítání (obyčejné sčítání polynomů) i násobení ($z^i z^j = z^{i+j}$). Volíme takové polynomy $q(x)$, že prvek z je primitivním prvkem tělesa $GF(p^n)$.

$GF(4) = Z_2/(x^2 + x + 1)$

$$
\begin{aligned}
0 &\\
1 &= z^0 = z^3\\
z &= z^1\\
1 + z &= z^2
\end{aligned}
$$

$GF(8) = Z_2/(x^3 + x + 1)$

$$
\begin{aligned}
0 &\\
1 &= z^0 = z^7\\
z &= z^1\\
z^2 &= z^2\\
1 + z &= z^3\\
z + z^2 &= z^4\\
1 + z + z^2 &= z^5\\
1 + z^2 &= z^6
\end{aligned}
$$

$GF(9) = Z_3/(x^2 + x + 2)$

$$
\begin{aligned}
0 &\\
1 &= z^0 = z^8\\
z &= z^1\\
1 + 2z &= z^2\\
2 + 2z &= z^3\\
2 &= z^4\\
2z &= z^5\\
2 + z &= z^6\\
1 + z &= z^7
\end{aligned}
$$

$GF(16) = Z_2/(x^4 + x + 1)$

$$
\begin{aligned}
0 &\\
1 &= z^0 = z^{15}\\
z &= z^1\\
z^2 &= z^2\\
z^3 &= z^3\\
1 + z &= z^4\\
z + z^2 &= z^5\\
z^2 + z^3 &= z^6\\
1 + z + z^3 &= z^7\\
1 + z^2 &= z^8\\
z + z^3 &= z^9\\
1 + z + z^2 &= z^{10}\\
z + z^2 + z^3 &= z^{11}\\
1 + z + z^2 + z^3 &= z^{12}\\
1 + z^2 + z^3 &= z^{13}\\
1 + z^3 &= z^{14}
\end{aligned}
$$

$GF(25) = Z_5/(x^2 + x + 2)$

$$
\begin{aligned}
0 &\\
1 &= z^0 = z^{24}\\
z &= z^1\\
4z + 3 &= z^2\\
4z + 2 &= z^3\\
3z + 2 &= z^4\\
4z + 4 &= z^5\\
2 &= z^6\\
2z &= z^7\\
3z + 1 &= z^8\\
3z + 4 &= z^9\\
z + 4 &= z^{10}\\
3z + 3 &= z^{11}\\
4 &= z^{12}\\
4z &= z^{13}\\
z + 2 &= z^{14}\\
z + 3 &= z^{15}\\
2z + 3 &= z^{16}\\
z + 1 &= z^{17}\\
3 &= z^{18}\\
3z &= z^{19}\\
2z + 4 &= z^{20}\\
2z + 1 &= z^{21}\\
4z + 1 &= z^{22}\\
2z + 2 &= z^{23}
\end{aligned}
$$

$GF(27) = Z_3/(x^3 + 2x + 1)$

$$
\begin{aligned}
0 &\\
1 &= z^0 = z^{26}\\
z &= z^1\\
z^2 &= z^2\\
z + 2 &= z^3\\
z^2 + 2z &= z^4\\
2z^2 + z + 2 &= z^5\\
z^2 + z + 1 &= z^6\\
z^2 + 2z + 2 &= z^7\\
2z^2 + 2 &= z^8\\
z + 1 &= z^9\\
z^2 + z &= z^{10}\\
z^2 + z + 2 &= z^{11}\\
z + z^2 &= z^{12}\\
2 &= z^{13}\\
2z &= z^{14}\\
2z^2 &= z^{15}\\
2z + 1 &= z^{16}\\
2z^2 + z &= z^{17}\\
z^2 + 2z + 1 &= z^{18}\\
2z^2 + 2z + 2 &= z^{19}\\
2z^2 + z + 1 &= z^{20}\\
z^2 + 1 &= z^{21}\\
2z + 2 &= z^{22}\\
2z^2 + 2z &= z^{23}\\
2z^2 + 2z + 1 &= z^{24}\\
2z^2 + 1 &= z^{25}
\end{aligned}
$$

$GF(32) = Z_2/(x^5 + x^2 + 1)$

$$
\begin{aligned}
0 \\
1 &= z^0 = z^{31} \\
z &= z \\
z^2 &= z^2 \\
z^3 &= z^3 \\
z^4 &= z^4 \\
z^2 + 1 &= z^5 \\
z^3 + z &= z^6 \\
z^4 + z^2 &= z^7 \\
1 + z^2 + z^3 &= z^8 \\
z + z^3 + z^4 &= z^9 \\
1 + z^4 &= z^{10} \\
1 + z + z^2 &= z^{11} \\
z + z^2 + z^3 &= z^{12} \\
z^2 + z^3 + z^4 &= z^{13} \\
1 + z^2 + z^3 + z^4 &= z^{14}
\end{aligned}
\qquad
\begin{aligned}
1 + z + z^2 + z^3 + z^4 &= z^{15} \\
1 + z + z^3 + z^4 &= z^{16} \\
1 + z + z^4 &= z^{17} \\
1 + z &= z^{18} \\
z + z^2 &= z^{19} \\
z^2 + z^3 &= z^{20} \\
z^3 + z^4 &= z^{21} \\
1 + z^2 + z^4 &= z^{22} \\
1 + z + z^2 + z^3 &= z^{23} \\
z + z^2 + z^3 + z^4 &= z^{24} \\
1 + z^3 + z^4 &= z^{25} \\
1 + z + z^2 + z^4 &= z^{26} \\
1 + z + z^3 &= z^{27} \\
z + z^2 + z^4 &= z^{28} \\
1 + z^3 &= z^{29} \\
z + z^4 &= z^{30}
\end{aligned}
$$

A2. Přehled BCH kódů a Reedových-Mullerových kódů

Uvádíme všechny existující binární kódy délky

$n = 2^m - 1 = 7, 15, 31, 63$ a 127

mezi BCH kódy a zůženými Reedovými-Mullerovými kódy. Udáváme délku (n), minimální vzdálenost (d), počet informačních znaků (k) a u kódů $R(r, m)^*$ ještě řád $r = 0, 1, \ldots, m - 2$. Až na jednu výjimku je plánovaná vzdálenost BCH kódů shodná s údajem d.

Délka	Minimální vzdálenost	BCH: inf. znaky	$R(r, m)^*$: inf. znaky	
7	3	4	4	$(r = 1)$
	7	1	1	$(r = 0)$
15	3	11	11	$(r = 2)$
	5	7	–	–
	7	5	5	$(r = 1)$
	15	1	1	$(r = 0)$
31	3	26	26	$(r = 3)$
	5	21	–	–
	7	16	16	$(r = 2)$
	11	11	–	–
	15	6	6	$(r = 1)$
	31	1	1	$(r = 0)$

Délka	Minimální vzdálenost	BCH: inf. znaky	$R(r, m)$*: inf. znaky	
63	3	57	57	$(r = 4)$
	5	51	–	–
	7	45	42	$(r = 3)$
	9	39	–	–
	11	36	–	–
	13	30	–	–
	15	24	22	$(r = 2)$
	21	18	–	–
	23	16	–	–
	27	10	–	–
	31	7	7	$(r = 1)$
	63	1	1	$(r = 0)$
127	3	120	120	$(r = 5)$
	5	113	–	
	7	106	99	$(r = 4)$
	9	99	–	
	11	92	–	
	13	85	–	
	15	78	64	$(r = 3)$
	19	71	–	
	21	64	–	
	23	57	–	
	27	27	–	
	31*)	43		
	31	36	29	$(r = 2)$
	43	29	–	
	47	22	–	
	55	15	–	
	63	8	8	$(r = 1)$
	127	1	1	$(r = 0)$

*) plánovaná vzdálenost je 29

Doporučená literatura a odkazy

Teorie kódování úzce souvisí s dalšími obory, o kterých jsme se nemohli zmínit. Je to především teorie informace. Velmi podrobnou monografií je kniha

[1] GALLAGER, R. G.: Information theory and reliable communication. New York, J. Wiley 1968. (*Ruský překlad:* Těorija informacji i naděžnaja svjaz. Moskva, Sovetskoje radio 1974.)

V první kapitole jsem se opíral o čtivou učebnici

[2] ABRAMSON, N.: Information theory and coding. New York, McGraw-Hill 1963.

Z novějších publikací je třeba jmenovat

[3] MACELIECE, R. J. The theory of information and coding. Reading, Addison-Wesley 1978.

Souvislosti teorie kódování s kombinatorikou a geometrií najde čtenář ve fundamentální monografii

[4] MACWILLIAMS, F. J. - SLOANE, N. J. A.: The theory of error-correcting codes. Amsterdam—New York—Oxford, North Holland 1981 (3. vydání).

Tato monografie, ze které jsem vycházel v kapitolách II až VII, obsahuje podrobné informace o bezpečnostních kódech z nejrůznějších hledisek a přináší také obsáhlou bibliografii. K dalším významným monografiím, věnovaným bezpečnostním kódům, patří

[5] BERLEHAMP, E. R.: Algebraic coding theory. New York, McGraw-Hill 1968. (*Ruský překlad:* Algebrajičeskaja těorija kodirovanija. Moskva, Mir 1971.)

a

[6] PETERSON, W. W.: Error-correcting codes. Cambridge, Mass., MIT Press 1961. (*Ruský překlad:* Kody, ispravlajuščije ošibky. Moskva, Mir 1964.)

Souvislosti bezpečnostních kódů s konečnou algebrou probírá skriptum matematicko-fyzikální fakulty KU v Praze

[7] BICAN, L. - KEPKA, T. - NĚMEC, P.: Úvod do teorie konečných těles a lineárních kódů. Praha, SPN 1982.

Základy lineární algebry potřebné pro pochopení naší knihy najde čtenář např. v knihách

[8] MACLANE, S. - BIRKHOFF, G.: Algebra. Bratislava, Alfa 1973

a

[9] DEMLOVÁ, M. - NAGY, J.: Algebra (Matematika pro vysoké školy technické III). Praha, SNTL 1982.

Konkrétní aplikace v práci počítačů probírá kniha

[10] MARTIN, J.: Security, accuracy and privacy in computer systems. New Jersey, Prentice Hall 1973.

V kap. VIII jsem vycházel z přehledového článku

[11] SLOANE, N. J. A.: Error-correcting codes and cryptography, v knize Mathematical Garden (ed. D. A. Klarner), California, Wadsworth 1981. (*Ruský překlad*: Matematičeskij cvetnik. Moskva, Mir 1983.)

Z novějších publikací, věnovaných bezpečnostním kódům, bych čtenáři doporučil tyto dvě:

[12] BLAHUT, R. E.: Theory and practice of error control codes. Reading, Addison-Wesley 1983,

[13] LIN, SHU-CASTELLO, D. J.: Error control coding. Englewood Cliffs N. J., Prentice Hall 1983.

Rejstřík

MATEMATIKA PRO VYSOKÉ ŠKOLY TECHNICKÉ

SEŠIT XXXI

MVŠT

RNDr. Jiří Adámek, CSc.

Kódování

DT 519.72 (075.8)
519.725 (075.8)

Vydalo SNTL - Nakladatelství technické literatury,
n. p., Spálená 51, 113 02 Praha 1
v roce 1989
jako svou 10670. publikaci
Redakce teoretické literatury
Odpovědná redaktorka
RNDr. Helena Havelková
Obálku navrhl Vladislav Jacák
Grafická úprava a technická redakce
Šárka Panošová
Vytiskla Polygrafia, závod 6 — Prometheus,
Praha 8, Tř. Rudé armády 171
192 stran, 28 obrázků
Typové číslo L11-C3-V-41/17995. Vydání první
Náklad 5000 prodejných výtisků. 13,84 AA, 14,16 VA

03/2

Cena brožovaného výtisku Kčs 14,—

104/21, 822

Publikace je určena hlavně posluchačům vysokých škol technických. Přivítají ji také absolventi těchto škol a technici z praxe, kteří pracují v oborech souvisejících s přenosem dat.

04—005—89 Kčs 14,—