# Obsah

# 1 Homomorfismus a isomorfismus grafů, souvislost, stromy, kostry, minimální kostra

**Obecně zde bude řeč o neorientovaných grafech.**

### Homomorfismus grafu

Homomorfismus grafu G do grafu G' je zobrazení $f : U(G) \to U(G')$ takové, že pro každou hranu $\{x, y\}$ grafu G **je** $f(x), f(y)$ hranou grafu G'. Jinými slovy graf může být schován v jiném.

### Isomorfismus grafu

Isomorfismus grafu G do grafu G' je bijekce[1] $f : U(G) \to U(G')$ pro kterou platí, že $\{x, y\}$ je hranou G, **právě když** $f(x), f(y)$ je hranou G'. Jinými slovy grafy jsou shodné až na označení vrcholů.

### Souvislost grafu

**Souvislý graf** Graf $G$ je souvislý pokud pro každou dvojici vrcholů grafu $x, y$ existuje cesta z bodu $x$ do bodu $y$. Jinak je nesouvislý.
Pozn.: Graf $G$ je také souvislý jsou-li stupně všech vrcholů alespoň $\frac{n}{2}$.

**Most** Hrana $\{x, y\}$ je mostem, jestliže neleží na žádné kružnici.
Pozn.: Graf s mostem má alespoň 2 uzly lichého stupně

**Artikulace** Má alespoň 2 hrany nepatřící téže kružnici.
Pozn.: Most má dvě artikulace.

**Blok** Blok grafu je maximální souvislý podgraf bez artikulace.
Pozn.: Žádné dva bloky nemají společnou hranu, protože by jinak nebyly maximální.
Pozn.: Pro každý graf G je blokový graf B(G) stromem.

**Hranový řez** Hranový řez B **mezi uzly** $x$ **a** $y$ (v souvislém grafu) je **minimální** množina hran taková, že každá cesta z $x$ do $y$ obsahuje alespoň 1 hranu z B.
Pozn.: Most je vlastně jednoprvkový hranový řez.
Pozn.: Odstraněním B se graf stane nesouvislým.

**Hranový stupeň** $\mathbf{h_G(x, y)}$ je počet prvků **nejmenšího** hranového řezu mezi $x$ a $y$.
Pozn.: Matice hranové souvislosti - matice s prvky $h_G(i, j)$.
Pozn.: Graf je hranově k-souvyslý, jestliže $h(G) \leq k$.

[1] vzájemně jednoznačné (prosté a na)

**Uzlový řez** Uzlový řez **A mezi uzly** $x$ **a** $y$ (v souvislém grafu) je **minimální** množina uzlů taková, že každá cesta z $x$ do $y$ obsahuje alespoň 1 hranu z **A**.
Pozn.: Odstraněním A se graf stane nesouvislým.
Pozn.: Artikulace je jednoprvkový uzlový řez.

**Uzlový stupeň** $\mathbf{u_G(x,y)}$ je počet prvků **nejmenšího** uzlového řezu mezi $x$ a $y$. Ale pokud jsou $x$ a $y$ sousední, tak $u_G(x,y) = |U(G)| - 1$
Pozn.: Graf je (uzlově[2]) k-souvyslý, jestliže $u(G) \leq k$.

**Další věty o k-souvislosti** Pro každý graf G platí - $\mathbf{u(G)} \leq \mathbf{h(G)} \leq \delta\mathbf{(G)}$.
[FORD, FULKERSON] Graf je hranově k-souvislý mezi uzly $a \neq b \Leftrightarrow$ je v něm k hranově disjunktních cest vedoucích mezi těmito uzly. (Algoritmus na toky.)
[MENGER] Graf je uzlově k-souvislý mezi nesousedními uzly $\Leftrightarrow$ je v něm k uzlově disjunktních cest vedoucích mezi těmito uzly. (Algoritmus s rozštěpenim vrcholu na zdroj a stok.)

### Stromy

Strom je souvislý graf, který neobsahuje žádnou kružnici. List stromu $T$ je libovolný vrchol se stupněm jedna.

### Kostra

Kostra grafu $G$, je souvislý podgraf bez kružnic, který obsahuje všechny vrcholy grafu $G$.

### Minimální kostra

Kostra $T$ grafu $G$ je minimální, pokud má minimální váhu.

Pro hledání minimálních koster se používá tzv. hladových algoritmů. Tyto algoritmy vybírají v každém kroku lokální optimum a nikdy se nevrací zpět. Vlastností těchto algoritmů je, že jsou rychlé, ale většinou neposkytují globální optimum. My ale nehledáme nejmenší kostru ale minimální, takže nám to stačí ...

**Algoritmus 1 (vstup = G, výstup = minimální kostra):** Dokud existuje nějaká kružnice, tak z ní odstraníme hranu s nejhorším ohodnocením.

**Algoritmus 2 (vstup = G, výstup = minimální kostra):** Dokud nemáme kostru, tak mezi všemi hranami najdi hranu s nejmenším ohodnocením, tu přidej do naší budoucí kostry, pokud nevytvoří cyklus. Jinak ji zahoď.

Složitost algoritmu je $O(m \cdot n^2)$, protože algoritmus proběhne v $(n-1)$ krocích (v každém kroku přidáme jednu hranu), pro každou hranu $m$ v každém kroku musíme zjistit jestli nevznikne kružnice a to si vyžádá dalších $n$ kroků.

[2]Slovo „uzlově“ se často vynechává.

# 2 Metrika grafu, minimální cesta, distanční matice grafu

**Obecně zde bude řeč o orientovaných grafech.**

### Vzdálenost v grafu

Vzdálenost $d_G(u, v)$ dvou vrcholů $u$ a $v$ v grafu G je dána **délkou**[3] nejkratší cesty[4] mezi $u$ a $v$. Pokud neexistuje, je vzdálenost $d_G(u, v) = \infty$.

Pozn: Z té definice se nechá vykoukat, že platí:
$d_G(u, v) \geq 0$, $(d_G(u, u) = 0)$,
$d_G(u, v) = d_G(v, u)$ (pro neorientovaný graf),
$d_G(u, z) \leq d_G(u, v) + d_G(v, z)$ (Trojúhelníková nerovnost).

### Metrika grafu

Metrikou grafu myslíme soubor vzdáleností mezi všemi dvojicemi vrcholů grafu. Jinak řečeno, **metrikou grafu G je distanční matice** D, ve které prvky $d_{ij}$ udávají vzdálenost mezi vrcholy $i$ a $j$.

### Distanční matice ohodnoceného grafu

Hrany takového grafu jsou ohodnoceny funkcí $w : H(\vec{G}) \to (0, \infty)$. w nazýváme matice $w$-vzdáleností. w-distanční matici grafu G značíme $D^w_G$ a její prvky jsou:

$$d^w_{i,j} = d^w_G(i, j) \text{ , kde } i, j \in U(\vec{G})$$

### Algoritmus výpočtu $D^w$ - Floydovým algoritmem

Výpočet distanční matice se provádí Floydovým algoritmem[5] v čase $O(n^3)$.

- Na počátku nechť **d[i,j]** $= 0$, nebo $w_{ij}$, nebo $\infty$ pokud hrana mezi $i$ a $j$ není.
- Po každém kroku $t \geq 0$ nechť **d[i,j]** udává délku nejkratší cesty mezi $i$ a $j$, která jde pouze přes vnitřní vrcholy z množiny $0, 1, 2, \ldots, t - 1$.
- Při přechodu z $t$ na následující krok $t + 1$ upravujeme vzdálenost pro každou dvojici vrcholů – jsou vždy pouze dvě možnosti:
  - Buď je cesta délky **d[i,j]** z předchozího kroku stále nejlepší (tj. nově povolený vrchol t nám nepomůže),

[3] Tzn. že je rovna délce té cesty.
[4] Původně tu bylo sledu. Ale podle me je to jedno a cesta se mi libi vic. :)
[5] To je ten jak vypočítá $n$ matic takovym upravenym násobenim

  - **nebo** cestu vylepšíme spojením přes nově povolený vrchol t, čímž získáme menší vzdálenost **d[i,t]+d[t,j]**.

- Po $|U(G)|$ krocích obdržíme $D^w$.

### Dijkstrův algoritmus

Ten použijeme, pokud nám stačí znám vzdálenost jen mezi dvěma body. Složitost $O(n^2)$.
Pozn.: Dijkstrův algoritmus nalezne 1 řádek distanční matice. Chceme-li matici celou, tak stačí výpočet n krát opakovat.[6]

### Nejkratší cesta

Cesta z $u$ do $v$ optimalizovaná vzhledem k počtu hran. Značíme ji $d_G(u, v)$. Pokud cesta neexistuje je $d_G(u, v) = \infty$.
Pozn.: Alg. prohledávání do šířky.

### Minimální cesta

Cesta P z $u$ do $v$ optimalizovaná vzhledem k w délce cesty ($w(P) = \sum_{k \in H(G)} w(k)$).

# 3 Silná souvislost, kvazikomponenty, kondenzace, acyklické grafy, kritická cesta.

### Silná souvislost

Orientovaný graf $\vec{G}$ je **silně** souvislý, pokud pro každou jeho dvojici uzlů $x$,$y$ existuje orientovaný sled z $x$ do $y$ i z $y$,$x$.

nebo: Souvislý orientovaný graf $\overrightarrow{G}$ je silně souvislý, právě když každá jeho hrana leží alespoň v jednom cyklu.

### Kvazikomponenty

Kvazikomponenta grafu $\vec{G}$ je jeho maximální silně souvislý podgraf.[7]

### Kondenzace

Kondenzace grafu $\vec{G}$ je graf $\vec{G}_c$ jehož vrcholy jsou kvazikomponenty. Hrany mezi kvazikomponentami se sloučí do jedné pokud je jich více v jednom směru.

[6] to má pak složitost $O(n^3)$.
[7] Zbytek může být třebas „jen“ (nesilně) souvislý.

### Acyklické grafy

Orientovaný graf $\vec{G}$ je acyklický, pokud neobsahuje žádný cyklus.
Pozn: Každý acyklický graf má vstupní a výstupní uzel.

Algoritmus (vstup = graf $\vec{G}$, úkol = je graf acyklický?, výstup = ano/ne): Problém je řešitelný v polynomiálním čase.

### Kritická cesta

Kritická cesta je cesta grafem přes uzly, které se nesmějí zpozdit.

# 4 Rozložitelnost a slabá rozložitelnost matic.

### Rozložitelnost matic

Čtvercová matice $A$ je rozložitelná, lze-li ji zapsat ve tvaru $A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$, kde $A_{11}$ a $A_{22}$ jsou čtvercové matice **řádu alespoň 1** a **0** je nulová matice, nebo lze-li ji do tohoto tvaru převést **stejnou** řádkovou a sloupcovou permutací (simultánní permutace). V opačném případě je nerozložitelná.
Pozn.: Pokud je orientovaný graf $\vec{G}$ silně souvislý pak je matice $W(\vec{G})$ nerozložitelná.
Pozn.: Algoritmus pro zjištění rozložitelnosti matice, který by pracoval na zkoušení různých permutací by měl složitost $O(n!)$.
Pozn.: Ale my to umíme rychleji. Najdeme kvazikomponenty, ty acyklicky očíslujeme a podle toho pak přeházíme řádky matice A. Takto vzniklé podmatice $A_{ij}$ jsou již dále nerozložitelné.

### Slabá rozložitelnost matic

Řekneme, že čtvercová matice A je slabě rozložitelná, jestliže existují permutacní matice P a Q tak, že $PAQ = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$, kde $A_{11}$ a $A_{22}$ jsou čtvercové matice **řádu alespoň 1** a **0** je nulová matice. Čtvercová matice, která není slabě rozložitelná, se nazývá úplně nerozložitelná.
Pozn.: Jak je vidět P a Q jsou různé matice. Takže jinými slovy stači najít tu novou matici libovolným přeházením řádků a sloupců.
Pozn.: Zda je matice slabě rozložitelná zjistíme z bigrafu, kde budeme hledat **stabilní množinu** (Co je to stabilní množina viz dále.).

**Bigraf** je orientovaný graf, s dvěmi disjunktními neprázdných množinami uzlů $U_1, U_2$, kde pro každou hranu $u, v \in H(\vec{G})$ platí, že $u \in U_1$ a $v \in U_2$.

**Bigraf** je acyklický graf, jehož každý uzel je buď vstupní nebo výstupní.

Množina $V$(to je ta vlevo - řádek) v bigrafu $\vec{G}$ je stabilní, pokud $|V| \geq |W|$ a kde:

$$V \in U_1, 0 \neq V \neq U_1 \quad \text{a} \quad W \in U_2.$$

Pozn.: Takže do množiny $W$(sloupce) patří všechny vrcholy do kterých vede hrana z $V$(řádky).

Přečíslujeme množinu $U_1$ a to tak, že **nejprve** očíslujeme vrcholy **mimo** množinu $V$ a **pak uvnitř** množiny. Získáme tak předpis pro permutaci řádků. Obdobně to provedeme z množinou $U_2$ a $W$ a získáme předpis pro permutace sloupců.

# 5 Generická hodnost matice

K výpočtu generické hodnosti matice potřebujeme znát strukturální matici. To je taková matice, u které je dána pouze struktura nenulových a nulových prvků a hodnoty prvků nejsou známy. Pro strukturální matici pak nastávají pouze dvě možnosti:

1. Determinant je nenulový a A je regulární pro jakékoliv náhodně zvolené hodnoty nenulových prvků. (s pravděpodobností 1)

2. Determinant je nulový a matice A je singulární pro jakékoliv náhodně zvolené hodnoty nenulových prvků.

Pozn.: Strukturální matice A je genericky regulární právě když její bigraf $B(A)$ má perfektní párování.
Pozn.: Počet hran největšího párování v bigrafu $\vec{B}$ se nazývá párovací číslo bigrafu $\vec{B}$ a značí se $\nu(\vec{B})$.
Pozn.: Nejvyšší přirozené číslo $k$, pro které v matici A existuje genericky regulární podmatice řádu $k$ se nazývá generická hodnost matice A a značí se $gh(A)$.
Pozn.: Máme-li A strukturální matici pak $gh(A) = v(\vec{B}(A))$. (Generická hodnost je rovna největšímu párování bigrafu matice A) Pozn.: Hledání generické hodnosti matice, lze převést na úkol hledání maximálního toku a v tom případě je $gh(A) =$ maximálnímu toku v grafu.

# 6 Síť, tok, existence toku v síti

**Síť**

Síť je orientovaný graf s kladným ohodnocením hran a s reálným ohodnocením uzlů.

### Tok

Tok v ohodnocené síti $\vec{G}$ je nezáporné hranové ohodnocení a musí splňovat dvě podmínky:

1. Ohodnocení každého uzlu = součtu toku, které vychází - součtu toku, které vchází do uzlu.
2. Pro každou hranu platí, že $0 \leq x_{ij} \leq r_{ij}$. (Tok není přesycen)

Pozn.: Hodnotě ohodnoceného uzlu $a_i$ se říká **intenzita uzlu**.
Pozn.: Hodnotě ohodnocené hrany $r_{ij}$ se říká **propustnost hrany**.
Pozn.: Je-li $a_i < 0$ pak se uzel nazývá **stok** a je-li $a_i > 0$ pak se uzel nazývá **zdroj**. V případě, že $a_i = 0$ jde o **neutrální uzel**.

### Tok množiny

Nechť $A \in U(\vec{G})$, pak platí:

$$a(A) = x(A, \overline{A}) - x(\overline{A}, A)$$

### Existence toku v síti

V síti existuje tok, jestliže $a(G) = 0$ a pro každou množinu $A \in U(G)$ je $a(A) \leq r(A, \overline{A})$.

# 7 Maximální tok v síti, Ford-Fulkersonova věta

### Tok z $a$ do $b$

Pro síť $\vec{G}$ s jedním zdrojem, nechť $(A, \overline{A})$ je řez sítě $\vec{G}$, oddělující $a$ a $b$ a $x$ je tok v $\vec{G}$, pak platí:

1. $|x| = x(A, \overline{A}) - x(\overline{A}, a)$ (Velikost toku = tok řezem oddělujícího a a b)
2. $|x| \leq r(A, \overline{A}))$ (tok $\leq$ max tok řezem)

### Maximální tok v síti

Pro síť $\vec{G}$ s jedním zdrojem a jedním stokem řekneme, že tok $|x|$je maximální jestliže pro každý tok $x'$ platí, že $|x'| \leq |x|$.
Pozn.: Prostě že neexistuje žádný větší.

### Rezervní polocesta

Polocesta je rezervní, právě když neobsahuje žádnou souhlasnou nasycenou hranu ani nesouhlasná nulová hranu.
Pozn.: Pokud pak v grafu existuje rezervní polocesta ze zdroje do stoku, není tok maximální.

### Ford-Fulkersonova věta

Mějme graf s jedním zdrojem a jedním stokem. Velikost maximálního toku je rovna propustnosti minimálního řezu[8], oddělujícího zdroj a stok.

Z toho plyne, že každá hrana, která vede z řezu musí být nasycená a každá hrana která vede do řezu musí být nulová, protože kdyby ne, pak by existovala rezervní polocesta a tok by nebyl maximální.

Pro sestrojení minimálního řezu budujeme množinu $R$ a to tak, že do ní přidáme zdroj a pak pokud vedou od vrcholu v množině $R$ nenasycené souhlasné hrany, přidáme vrcholy na konci hran do množiny $R$, nebo také přidáme vrcholy do kterých vedou nenulové nesouhlasné hrany.

### Algoritmus Ford-Fulkersonův

1. Výchozí tok je pro každou hranu nulový $x_{ij} = 0$
2. Pokud existuje nějaká rezervní polocesta, upravíme podle ní tok $x$:
   - $x_{ij} + \Theta$ pokud $(i, j)$ je souhlasná hrana polocesty.
   - $x_{ij} - \Theta$ pokud $(i, j)$ je nesouhlasná hrana polocesty.
   - $x_{ij}$ pokud $(i, j)$ neleží na cestě.
3. Pokud rezervní polocesta neexistuje máme maximální tok.

Pozn.: Pro použitelnost algoritmu se musí zajistit aby byl konečný. Toho se docílí celočíselnými propustnostmi hran.
Pozn.: Složitost algoritmu je závislá nejen na počtu uzlů a hran, ale také na ohodnocení hran.

### Algoritmus Edmonds-Karpův

Je to modifikace F-F algoritmu s rozdílem, že volí nejkratší možnou rezervní polocestu. Výběr se může provést např. pomocí prohledávání grafu do šířky.
Pozn.: Složitost algoritmu je $O(m^2 n)$.

# 8 Míry souvislosti grafu

Řeč je tu o neorientovaných grafech.

### Mosty

Hrana $\{x, y\}$ je mostem, jestliže neleží na žádné kružnici.
Pozn.: Jestliže má souvislý graf most, pak má alespoň 2 uzly lichého stupně.
Pozn.: Pokud tuto hranu z grafu odstraníme, stane se ze souvislého grafu nesouvislý.

[8] Řez s nejmenší propustností ze věch řezů.

### Artikulace

Uzel $x$ je artikulace grafu, jestliže existují 2 hrany z uzly $x$, které nepatří současně téže kružnici.
Pozn.: Každý uzel mostu je artikulace.
Pozn.: Pokud nemá graf artikulaci nemůže mít ani most.
Pozn.: Graf, který nemá artikulace je (alespoň) 2 souvislý.

### Bloky

Blok grafu je maximální souvislý podgraf bez artikulace.
Pozn.: Dva různé bloky nemají žádnou společnou hranu, nebo jsou si rovny.
Pozn.: Pro každý graf G je blokový graf B(G) stromem.
Pozn.: Každý most je blokem grafu o jedné hraně.
Pozn.: Každé dvě hrany bloku, které nejsou mostem, leží na jedné kružnici.
Pozn.: Dva různé bloky nemají žádnou společnou hranu.

### Blokový graf

Blokový graf se sestaví z grafu $G$ tak, že jeho uzly jsou jednotlivé bloky a artikulace grafu $G$. A jeho hrany vedou mezi novými uzly utvořenými z bloků a artikulací.
Pozn.: Každý takto sestavený blokový graf je stromem pro souvislý graf $G$.

### Hranový řez

Hranový řez B **mezi uzly** $x$ **a** $y$ (v souvislém grafu) je **minimální** množina hran taková, že každá cesta z $x$ do $y$ obsahuje alespoň 1 hranu z B.
Pozn.: Most je vlastně jednoprvkový hranový řez.
Pozn.: Odstraněním B se graf stane nesouvislým.
Pozn.: Pokud množinu B odstraníme bude $x$ a $y$ ležet v různých komponentách.

### Hranový stupeň souvislosti $\mathbf{h_G(x, y)}$

Hranový stupeň souvislosti je počet prvků **nejmenšího** hranového řezu mezi $x$ a $y$.
Pozn.: Matice hranové souvislosti - matice s prvky $h_G(i, j)$.
Pozn.: Graf je hranově k-souvyslý, jestliže $h(G) \geq k$.

### Uzlový řez

Uzlový řez **A mezi uzly** $x$ **a** $y$ (v souvislém grafu) je **minimální** množina uzlů taková, že každá cesta z $x$ do $y$ obsahuje alespoň 1 uzel z **A**.
Pozn.: Odstraněním A se graf stane nesouvislým.
Pozn.: Pokud množinu A odstraníme bude $x$ a $y$ ležet v různých komponentách.
Pozn.: Artikulace je jednoprvkový uzlový řez.

### Uzlový stupeň souvislosti $\mathbf{u_G(x, y)}$

Uzlový stupeň $\mathbf{u_G(x, y)}$ je počet prvků **nejmenšího** uzlového řezu mezi $x$ a $y$. Jsou-li však $x$ a $y$ sousedy (neexistuje uzlový řez) pak $u_G(x, y) = |U(G)| - 1$.
Pozn.: Graf je (uzlově[9]) k-souvyslý, jestliže $u(G) \geq k$.

### k–souvislost

Graf je hranově k-souvyslý, jestliže $h(G) \geq k$.
Graf je (uzlově[10]) k-souvyslý, jestliže $u(G) \geq k$.
Pozn.: Pro každý graf G platí - $\mathbf{u(G)} \leq \mathbf{h(G)} \leq \delta(\mathbf{G})$, kde $\delta(x)$ je nejmenší stupeň vrcholu v grafu $G$.
Pozn.: V každém grafu $G$ platí $h(G) \leq \frac{2|H(G)|}{|U(G)|}$.

### Charakterizační věty k–souvislých grafů

**FORD, FULKERSON** Graf je hranově k-souvislý mezi uzly $a \neq b \Leftrightarrow$ je v něm $k$ hranově disjunktních cest vedoucích mezi $a$ a $b$.
Pozn.: Řeší se algoritmem na toky. Hrany ohodnoceny jedničkou.
Pozn.: Algoritmus(Vstup = neorientovaný G, Otázka: je $G$ hranově k-souvislý mezi $a$ a $b$?, Výstup: Ano/Ne) Složitost algoritmu je $O(n^3)$.

**MENGER** Graf je uzlově k-souvislý mezi nesousedními uzly $\Leftrightarrow$ je v něm k uzlově disjunktních cest vedoucích mezi těmito uzly.
Pozn.: Řeší se opět algoritmem na toky s rozštěpenim vrcholu na zdroj a stok. Mezi zdrojem a stokem propustnost 1, jinde $\infty$. Složitost algoritmu je $O(n^3)$.
Pozn.: Vidíme zde převod uzlové souvislosti na hranovou.

# 9 Algoritmy prohledávání a jejich použití

### Algoritmus prohledávání grafu

Je základem mnoha dalších algoritmů jako například prohledávání do hloubky a do šířky.
Pozn.: Komponenty lze nalézt v $O(m + n)$ (Pustíme-li prohledávání grafu na nějaký uzel, tak proleze pouze jednu komponentu. Koukneme jaké uzly nasel, uděláme $G-$closed a pokud ještě něco zbylo, tak je to další komponenta...)

**open** Seznam uzlů, které ještě nebyly prozkoumány.

**closed** Seznam uzlů, které jsou prozkoumány.

[9] Slovo „uzlově“ se často vynechává.
[10] Slovo „uzlově“ se často vynechává.

**Algoritmus** by se dal rozdělit do následujících kroků:

1. Vytvoříme seznam **open** a **closed**. Do seznamu **open** vložíme uzel $\mathbf{n_0}$ a určíme pro něj ohodnocení $\mathbf{f_0}$.

2. Je-li seznam **open** prázdný, řešení neexistuje, a končíme prohledávání.

3. Ze seznamu **open** vybereme uzel $\mathbf{n_i}$ s extrémním ohodnocením. Uzel $\mathbf{n_i}$ vložíme do seznamu **closed** a smažeme jej v seznamu **open**.

4. Provedeme expanzi uzlu $\mathbf{n_i}$. Pokud nemá žádné sousedy, tak se vrátíme na bod **2**.

5. Je-li mezi vygenerovanými následovníky uzlu $\mathbf{n_i}$ alespoň jeden cílový uzel, pak vybereme ten, který má extrémní ohodnocení. Ve vygenerovaném stromu řešení najdeme cestu od kořene stromu k cílovému uzlu a jako řešení poskytneme posloupnost elementárních operátorů (produkčních pravidel) prováděných podél cesty. Ukončíme prohledávání.

6. Pro každého bezprostředního následovníka uzlu $\mathbf{n_i}$, kterého označíme $\mathbf{n_j}$, vypočteme jeho ohodnocení $\mathbf{f_j}$ a pokusíme se najít stejný stav v seznamu **open**, nebo **closed**, který označíme $\mathbf{n_s}$ a jeho ohodnocení $\mathbf{f_s}$.
   Pokud takový stav v **open** ani v **closed** neexistuje, pak $\mathbf{n_j}$ zařaď do seznamu **open**. (uzel se ve vygenerovaném stromu řešení dosud nevyskytuje).
   Pokud $\mathbf{f_j} > (<) \mathbf{f_s}$ a $\mathbf{n_s}$ není předchůdcem $\mathbf{n_j}$, tak pak zařaď $\mathbf{n_j}$ do seznamu **open** a zruš uzel $\mathbf{n_s}$. Pokud byl zrušen nějaký uzel v seznamu **closed**, pak musejí být rovněž zrušeni všichni jeho následovníci. Je přitom lhostejno zda se nachází v seznamu **open** nebo **closed**.

7. Pokračuj krokem **2**.

### Prohledávání do šířky (breadth first search)

Jedná se o speciální případ základního algoritmu hledání v grafu, v němž platí:

- ohodnocení uzlu = hloubka uzlu
- V kroku 3, ve výše popsaném algoritmu hledání v grafu, se bere **minimum**.

Pozn.: Tento algoritmus vždy nalezne nejkratší cestu k cíli (pokud nějaká cesta existuje).
Pozn.: Složitost algoritmu je $O(m + n)$. Přidání všech vrcholů do $N$ trvá $O(n)$, vložení a vybrání z fronty je $O(1)$, sousedé vrcholů jsou procházeni jenom jednou a jejich délka je $O(m)$.

### Prohledávání do hloubky (depth first search)

Jedná se o speciální případ základního algoritmu hledání v grafu, v němž platí:

- ohodnocení uzlu = hloubka uzlu
- V kroku 3, ve výše popsaném algoritmu hledání v grafu, se bere **maximum**.
- V kroku 3, ve výše popsaném algoritmu hledání v grafu, je nutno přidat test, zda již bylo dosaženo zadané maximální hloubky. Pokud ano, vracíme se ke kroku 2.

Pozn.: Využití najde například při hledání artikulací a bloků grafu, zjišťování k–souvislostí, generování hamiltonovských cest a cyklů, hledání komponent aj.
Pozn.: Složitost algoritmu je $O(m+n)$.

### Artikulace a bloky grafu

Bactracking má tu vlastnost, že v místě artikulace vleze do větve, celou ji projde a pak ji opustí zase v místě artikulace.

### Algoritmus Backtracking

1. uzly se očíslují v pořadí nalezení $p(x)$.
2. pro každý uzel $x$ se vypočte dolní číslo $r(x)$ a to tak:
   (a) při objevení nového uzlu je $r(x) = p(x)$
   (b) při nalezení chordy $xy$ položíme $r(x) = \min p(y), r(x)$
   (c) při zpětném kroku z $x$ do $y$ položíme $r(y) = \min r(y), r(x)$

Pozn.: Složitost algoritmu je $O(m+n)$.
Pozn.: Vyhledá artikulace v $O(m+n)$.
Pozn.: Uzel x **je artikulace právě když** existuje jeho následovník, ze kterého se nelze dostat do s nižším $p(x)$ než má uzel $x$.
Pozn.: Pokud chceme najít bloky, pak když při zpětném kroku nalezneme artikulaci $x$, dáme na výstup všechny prozkoumané hrany mezi prvním (dopředným) a druhým (zpětným) průchodem uzlu $x$.
Pozn.: Hledání artikulace lze řešit naivním způsobem a to tak, že se vždy odebere jeden uzel a zkoumá se, jestli vzrostl počet komponent. Takový algoritmus má složitost $O(n(n+m))$.
Pozn.: Existuje i modifikace pro nalezení kvazikomponent, resp. kondenzace. (To je sice možné i z distanční matice $O(n^3)$, ale toto by mělo být rychlejší)
Pozn.: Existuje i modifikace 3-komponent(maximální 3-souvislý podgraf), biartikulace (dvouuzlové řezy) s $O(n+m)$. Pro vyšší k–souvislosti už modifikace backtrackingu známy nejsou.

### Hranová souvislost grafu

Obecně lze získat $h_G(x, y)$, převedením na úlohu maximálního toku a řešit v čase $O(n^3)$. Pokud tedy chceme získat hranovou souvislost grafu, pak musíme provést tento postup pro všechny dvojice uzlů, takže $n^2$. Pak celková složitost je $O(n^5)$. Nejlepší známé vylepšení snižují odhad na $O(n^{4,5})$.

### Uzlová souvislost grafu

Obecně lze získat $u_G(x, y)$, štěpení uzlů a převedením úlohy na maximální tok a ten řešit v čase $O(n^3)$. To zas provedeme pro všechny dvojice uzlů $n^2$, takže pak celková složitost je $O(n^5)$. Podobně jako u hranové souvislosti lze získat řešení v $O(n^{4,5})$.
Pozn.: Oba tyto postupy jsou nejlepší známé na zjišťování k–souvislostí pro $k \geq 4$. Pro $k \leq 4$ lze použít upravený backtracking.

# 10 Eulerovské a hamiltonovské grafy, postačující podmínky hamiltonovskosti

### Eulerovský graf

Eulerovský graf je souvislý graf $G$, v kterém existuje eulerovský tah.
Pozn.: Euleroský tah je sled, při kterém navštívíme každou hranu grafu právě jednou.
Pozn.: Každý uzel eulerovského grafu je obsažen, alespoň na jedné kružnici v grafu.
Pozn.: Pokud je tah uzavřený pak každý uzel grafu je sudého stupně.
Pozn.: Pokud je graf otevřený pak dva uzly v grafu jsou lichého stupně a to právě ty, kde tah začíná a končí.
Pozn.: Složitost algoritmu je $O(m + n)$.

### Hamiltonovský graf

Je takový graf, který obsahuje hamiltonovskou kružnici. Hamiltonovská kružnice je tah, při jehož sestavování projdeme všechny vrcholy grafu právě jednou a vrátíme se do vrcholu, z kterého jsme vyšli.
**Algoritmus:** Vstup: Graf $G$.
Úloha: Ham. kružnice/cesta?
Výstup: Ano/Ne.

Modifikuje se bactracking a to tak:

1. neberou se v úvahu chordy
2. pokud $|D| = n$ pak testujeme existenci hrany z koncového do počátečního uzlu:
   (a) Ne: hamiltonovkská cesta

(b) Ano: hamiltonovská kružnice

3. při zpětném kroku se díváme, zda neexistuje následník s vyšším číslem, pokud ano pokračujeme zase vpřed

Pozn.: Hledání hamiltonovských kružnic v grafu patří mezi NP-úplné problémy. Pro vyhledávání se používají heuristiky, které **mohou** v polynomiálním čase dát buď odpověď ano nebo nevím.

### Postačující podmínky hamiltonovsti

[**Chvátal**] $G$ je k–souvislý($k \geq 2$) a $\alpha(G) \leq k$, pak je hemiltonovský.

[**ORE**] Jestliže pro všechny dvojice nesousedních uzlů $x, y$ je $d(x) + d(y) \geq n$, pak je graf hamiltonovský.

[**DIRAC**] Pokud je $\delta(G) \geq \frac{n}{2}$ pak je graf hamiltonovský.

[**ORE lemma**] Máme-li dva nesousední uzly $\{x, y\}$ takové, že $d(x) + d(y) \geq n$. Pak je graf G je hamiltonovský, právě když je $G + \{x, y\}$ hamiltonovský.

Pozn.: Je-li uzávěr grafu $G$ (se 3 a více uzly), úplný graf, pak G je hamiltonovský. Pokud tedy graf nemá úplný uzávěr pak s jistotou nenalezneme ham. kružnici. (Uzávěr je graf, který vznikne přidáním věch hran splňující Oreho lemma)

### Pósovo heuristika

Snaží se o přímočaré prodlužování cesty, pokud neexistuje hrana z koncového uzlu do volného pak vezme hranu do nějakého uzlu na cestě a provede modifikaci. Pokud se nenalezne řešení přečíslují se uzly a začíná se znovu. Nutná je ochrana proti zacyklení.
Pozn.: Viz obrázek ve skriptech na straně 35.

### Užití Oreho lemma (heuristika)

Nejdříve vytvoříme z grafu $G$ úplný uzávěr grafu rekurentním přidáváním hran podle Oreho lemmatu a testujeme stále splnění lemmatu. Pak uhádneme hamiltonovskou kružnici. Pokud kružnice leží na původních hranách pak je hotovo. Pokud ne odstraníme naposledy přidanou hranu $u, v$. Pokud porušíme kružnici, získáme ham. cestu $P$ z $u$ do $v$, pak uzly této cesty postupně očíslujeme $x_1, x_2 \ldots$.

Do množiny $M$, přidáme vrcholy $x_i$ pokud z $u$ do $x_{i+1}$ vede hrana a do množiny $N$ přidáme vrcholy $x_i$ pokud z $x_i$ do $v$ vede hrana. Průnik množin $M \cap N = x_i$ a zrušíme hranu $\{x_i, x_{i+1}\}$. Pak utvoříme hrany $\{u, x_{i+1}\}$ a $\{x_i, v\}$.
Pozn.: Viz obrázek ve skriptech na straně 35.

# 11 Nezávislost, dominance, klikovost a jádro v neorientovaných grafec

### Nezávislost

Množina $A$, která je podmnožinou všech uzlů grafu $G$, je **nezávislá** (vnitřně stabilní), pokud žádné dva uzly z $A$ nejsou spojeny hranou.

**Nezávislost grafu** $\alpha(G)$ je pak **největší** počet prvků nezávislé množiny grafu.
Pozn.: Vlastní podmnožina nezávislé množiny je také nezávislá. Proto se má cenu ptát na největší nezávislou množinu.
Pozn.: Zatím co nalezení maximální nezávislé množiny je polynomiální problém(zvolíme si jakýkoliv uzel, odstraníme jej i se sousedy a pak postup opakujeme), tak nalezení největší nezávislé množiny je *NP-těžký* problém.
Pozn.: Množina $A$ je nezávislá právě když $G - A$ je množina $B$ pokrytí grafu $G$.

**Pokrytí grafu** $G$ je množina uzlů $B$, pro kterou platí, že $\{x, y\} \in H(G)$ je $x \in B$ nebo $y \in B$ (mohou být i oba).
Pozn.: Každá nadmnožina B je také pokrytí. Má proto smysl hledat nejmenší pokrytí.
Pozn.: Pak pokrývací číslo $\beta(G)$ je nejmenší pokrytí grafu $G$. Je pak jasné, že $\alpha(G)+\beta(G) = |U(G)|$.

### Klika

Klika je podmnožina uzlů, která tvoří maximální úplný podgraf grafu $G$. Klikovost grafu $\omega(G)$ je pak klika s největším počtem uzlů v grafu $G$.
Pozn.: Pomocí kliky a doplňku grafu, lze také určit maximální nezávislou množinu. Doplněk grafu obsahuje všechny vrcholy původního grafu a všechny hrany, které původní graf **neměl**. Pokud $A$ je maximální nezávislá $\overline{G}$, pak žádné dva uzly v $A$ nejsou spojeny hranou, právě když v $G$ jsou každé dva její uzly spojeny hranou. Z toho pak plyne, že $\alpha(G) = \omega(\overline{G})$.

### Dominantnost

Množina $B$ se nazývá dominantní, pokud každý uzel, který není v množině $B$, má alespoň souseda v množině $B$. Pak číslo dominance $\gamma(G)$ je nejmenší počet prvků v dominantní množině $B$.
Pozn.: Protože každá nadmnožina je také dominantní, tak se ptáme na nejmenší a minimální.
Pozn.: Nezávyslá a dominantní $\Rightarrow$ maximální nezávislá množina.
Pozn.: Maximální nezávislá množina $\Rightarrow$ minimální dominantní. Obráceně to však neplatí viz obrázek ve skriptech na straně 40.
Pozn.: Je tedy jasné, že $\alpha(G) \geq \gamma(G)$.

### Jádro neorientovaného grafu

Jádro neorientovaného grafu je množina uzlů, která je současně nezávislá i dominantní. To je vlastně každá maximální nezávislá množina, protože je zároveň i minimálně dominantní. Každý **neorientovaný** (pro orientovaný to neplatí) graf má jádro.

# 12 Jádro v orientovaném grafu

Definice nezávislé množiny pro orientované grafy je stejná jako pro neorientované. Pak jádro orientovaného grafu je množina $C$, která musí splňovat tyto dvě podmínky:

1. $C$ je nezávislá množina
2. z každého uzlu mimo jádro, existuje hrana do jádra

Pozn.: Každý orientovaný graf bez cyklů liché délky má jádro. (Hledáme přes kvazikomponenty.)

### Acyklický graf

Každý acyklický graf má jádro. Důkaz, lze provézt použitím tzv. Grundyho funkce.

### Algoritmus:

1. Provedeme acyklické číslování a poslední uzel ohodnotíme 0.
2. Postupujem zpět podle acyklického číslování a v každém uzlu se koukneme na ohodnocení všech následníků a vybereme nejmenší možné číslo z přirozené posloupnosti 0. . . .N, které ještě žádný následník nemá.
3. Všechny uzly, které mají ohodnocení 0, tvoří jádro grafu

### Orientovaný graf bez cyklů liché délky

Každý orientovaný graf, bez cyklů liché délky, má jádro.

### Algoritmus:

1. V grafu vyhledáme kvazikomponenty a provedeme kondenzaci.
2. Vezmeme výstupní kvazikomponentu $G_0$ a provedeme následující:
   (a) zvolíme v ní libovolný bod $x$
   (b) pokud existuje z bodu $y$ do $x$ sled sudé délky dáme $y$ do množiny $S$ (samozřejmě tam patří i sled délky 0, takže i $x$)

3. Je v grafu ještě další kvazikomponenta:
   (a) Ano – $G = G - G_0$ pokračuj 2
   (b) Ne – množina S obsahuje jádro grafu

# 13 Barevnost grafu, chromatické číslo grafu

**k–obarvitelnost a chromatické číslo grafu**

Graf se nazývá k–obarvitelný, pokud lze uzly obarvit $k$ barvami tak, aby žádné dva sousední uzly neměli stejnou barvu.
Chromatické číslo (obarvitelnost) grafu $\chi(G)$ je pak nejmenší počet barev, kterými lze graf obarvit.
Pozn.: Každý graf je $n$–obarvitelný.
Pozn.: Pokud graf obsahuje úplný podgraf $K_k$. Pak $\chi(G) \geq k$. Úplný graf má barvitelnost $k$, proto když je podgrafem jiného, tak ten graf, který ho obsahuje, musí mít minimálně stejnou barvitelnost.
Pozn.: Horní odhad obarvitelnosti je $\chi(G) \leq \Delta(G) + 1$, kde $\Delta(G)$ je maximální stupeň uzlu v grafu. Musí to tak být protože sousedy takového uzlu obarvíme $\Delta(G)$ barvami a na uzel musíme pak použít další barvu.

[**Books**] $\chi(G) \leq \Delta(G)$ až na 2 výjimky:

1. G má komponentu $K_{(\Delta(G)+1)}$
2. $\Delta(G) = 2$ a G má komponentu kružnici liché délky.

Pozn.: $\chi(G) = 2$ právě když neobsahuje lichou kružnici.
Pozn.: $\chi(G) \leq 4$ pokud je graf rovinný.
Pozn.: Zjistit zda je graf obarvitelný 3 barvami patří už mezi NP-úplné problémy. Určení barevnosti grafu je ekvivalentní s rozkladem grafu na minimální počet nezávislých množin. Každá nezávislá množina totiž představuje jednu barvu.

Pro graf s chromatickým číslem $\chi(G)$ a nezávislostí $\alpha(G)$ platí:

1. $\chi(G)\alpha(G) \geq |U(G)|$, protože při optimálním obarvení je jedna barva nezávislou množinou o nejvíce $\alpha(G)$.
2. $\chi(G) + \alpha(G) \leq |U(G)| + 1$, protože nezávislou množinu obarvíme jednou barvou a zbylé uzly každý jinou barvou, ale pravděpodobně existuje lepší obarvitelnost proto $\leq$.

**Heuristický algoritmus 1:**

1. Hledej v $G$ nezávislou množinu
2. obarvi ji jednou barvou a vyhoď
3. ve zbytku grafu pokračuj bodem jedna

**Heuristický algoritmus 2:**

1. vezmi jeden uzel
2. koukni na sousedy a obarvi ho nejnižší barvou, kterou nemá žádný soused
3. pokračuj v bodu 1

### Rovinné grafy

Rovinné grafy jsou takové, které lze nakreslit v rovině tak, že žádná z hran grafu se nekříží. Každý rovinný graf má $\chi(G) \leq 4$.

### Hranové barvení

Dobré hranové obarvení je takové, že každá hrana, která má společný uzel, má jinou barvu. Chromatický index $\chi'(G)$ je nejmenší číslo $k$, pro které je graf obarvitelný.

U uzlového obarvení neexistuje dolní omezující funkce, u hranového je to jiné. Pro každý graf platí, $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$. První nerovnost je jasná, z uzlu jde tolik hran kolik je jeho stupeň a proto musí být i tolik barev.
Pozn.: Lze dokázat, že $\chi'(K_{2n-1}) = \chi'(K_{2n}) = 2n - 1$.

# 14 Třída problémů P, NP

### Třída P

Třída P je třída všech jazyků (problémů), pro něž existuje polynom $p$ takový, že udává časovou složitost jazyka nejvýše $p$.

Jde teda o všechny problémy, které jsou řešitelné v polynomiálním čase na základě výpočtu rozhodovacího algoritmu. Pokud je problém optimalizačního charakteru, musí se na rozhodovací problém převést.

Abstrakce pro rozhodovací algoritmus je deterministický přijímací počítač, jehož každý krok je jednoznačný a výstupem je pak akceptování a nebo odmítnutí slova na vstupu.

### Třída NP

Třída $NP$ je třída všech jazyků $J$ takových, že pro ně existuje nedeterministický počítač, který pracuje v omezeném polynomiálním čase a jazyk $J$ přijímá.

Jde tedy o třídu problémů u nichž uhádnu řešení a pak jeho správnost dokážu ověřit v polynomiálním čase.

Oproti deterministickému počítači je zde navíc příkaz CHOOSE J1, J2(vyber si J1 nebo J2). Deterministický počítač je tedy speciální případ nedeterministického a proto $P \subset NP$.

Nedeterministický počítač pracuje v polynomiálním omezeném čase, jestliže existuje polinom $p$ takový, že libovolné slovo délky $n$ zpracuje v čase nejvýše $p(n)$.

# 15 Splnitelnost logických formulí, polynomialita problému 2-SAT

### Splnitelnost logických formulí

Vstup: Logická formule v proměnných v KNF. Úkol: Zjistit zda je formule splnitelná.

Pokud máme logickou formuli, můžeme se pokusit řešení uhádnout. V polynomiálním čase pak můžeme ověřit, zda je formule pro uhádnuté řešení splnitelná. Pokud však řešení neuhádneme, nemůžeme jednoznačně říct, že formule je nesplnitelná pokud neprojdeme všechny možnosti $2^n$.

Neexistuje efektivní algoritmus, který by nám dal jednoznačné řešení v polynomiálním čase. Tato vlastnost je typická i pro další úlohy podobného typu, např. hamiltonovskost, existence kliky velikosti $k$, k–obarvitelnost, aj...

### 2-SAT

2-SAT patří do třídy P. Důkaz je algoritmus, který v polynomiálním čase, dokáže rozhodnout zda je booleovská formule typu 2-SAT splnitelná.

Vstup: Formule $f$, jejíž každá klauzule obsahuje právě dva literály a je KNF
Úkol: Je formule $f$ splnitelná?
Výstup: Ano/Ne

1. Sestavíme orientovaný graf $\vec{G}$ a to tak, že každá klauzule bude obsahovat v grafu dvě orientované hrany podle předpisu $(\overline{a}, b), (\overline{b}, a)$.

2. Následně vyhledáme kvazikomponenty a provedeme kondenzaci

3. Provedeme acyklické číslování a sestavíme splňující přiřazení tak, že postupujeme podle acyklického číslování pozadu a literálům přiřazujem hodnoty:

   (a) $x_i = 1$ , pokud $x_i \in U(G_C)$

   (b) $x_i = 0$ , pokud $\overline{x_i} \in U(G_C)$

4. Pro splňující přiřazení $t$ pak formule nabývá hodnoty 1 a je splnitelná.

Proto, abychom zjistili zda je formule splnitelná nám stačí zjistit, že žádná kvazikomponenta neobsahuje žádnou proměnou a její negaci zároveň.

Algoritmus pracuje v polynomiálním čase, protože vyhledání kvazikomponent, kondenzace a acyklického číslování lze provést v polynomiálním čase.

# 16 NP-úplné problémy

Problém je NP-úplný, jestliže náleží do třídy NP a kterýkoliv jiný problém z NP lze na něj polynomiálně převést. NP úplné problémy jsou nejtěžší problémy v NP, protože můžeme, kterýkoliv jazyk $J' \triangleleft J$, a přitom $J$ je alespoň tak těžký jak $J'$, které náleží NP. NP-úplné problémy jsou co do obtížnosti ekvivalentní.

Platí, že $P = NP = NPC$ nebo $P \cap NPC = 0$.

# 17 NP-úplnost problému splnitelnosti logických formulí.

To, že je SAT v NPC se těžko dokazuje, pokud je třída NPC prázdná. Naštěstí zde ale máme Cookovu věta, že $SAT \in NPC$ a důkaz této věty, zařazuje problém SAT jako první problém do množiny NPC.

Důkaz je složen z dvou hlavních částí. Nejprve se musí dokázat, že SAT je v NP, což je poměrně snadné, protože při určování hodnot literálů, používáme nedeterministický přístup, kdy dáváme literátu na výběr buď z hodnotu 0 nebo 1. Správnost řešení určíme v polynomiálním čase, který je úměrný délce formule.

Zbývá tedy dokázat, že každý jazyk $J$ z NP je na SAT převoditelný v polynomiálním čase. Úkolem je tedy pro každé slovo jazyka $J$ nalézt formuli v KNF takovou, že bude splnitelná, právě když bude akceptováno slovo jazyka $J$. Následuje důkaz na dvě stránky, který fakt nedám.

Poznámka: s nedeterminismem se ve formulích vyrovnáme pomoci de Morgana, tj. $(d \Rightarrow d_{L1} \vee d_{L2}) \Leftrightarrow (\overline{d} \vee d_{L1} \vee d_{L2})$.

# 18 Problém 3-splnitelnosti logických formulí

Vstup: Logická formule v KNF, jejíž každá klauzule má právě tři literály.
Úloha: Je úloha splnitelná?
$3 - SAT \in NPC$
To, že je 3-SAT v NP je zřejmé, protože je to speciální případ SAT.
To, že je NPC dokážeme tím, když $SAT \triangleleft 3 - SAT$. Tím, že redukuje SAT na 3-SAT dokážeme, že je to alespoň tak těžký problém jako SAT. Úpravami, můžeme dosáhnout toho,aby libovolná KNF obsahovala klauzule pouze o nejvýše 3 literálech.

# 19 Nezávislost grafu a její NP-úplnost

Vstup: Graf $G$ na $n$ uzlech a přirozené číslo $k \leq n$.
Úkol: Zjistit existenci nezávislé množiny velikosti alespoň $k$ v $G$.
$IND \in NPC$

To, že je IND v NP je zřejmé, protože nedeterminismus je v rozhodnutí, že uzel patří nebo nepatří do množiny nezávislých uzlů.

Pro dokázání, že IND je NPC, je potřeba ukázat $SAT \triangleleft IND$. Tedy, že v $G$ existuje nezávislá množina, právě tehdy pokud formule je splnitelná.

Sestrojíme tedy graf $G$ tak, že počet vrcholů budou odpovídat počtu literálů a pro hrany budou platit následující dvě pravidla:

1. vrcholy odpovídající jedné klauzuli budou tvořit úplný souvislý podgraf grafu $G$.
2. Vrcholy mezi klauzulemi budou spojeny hranou pouze mezi literály s vlastností $\{x, \overline{x}\}$

Pro takto sestavený graf budou pak odpovídat množiny nezávislých vrcholů, řádkám v pravdivostní tabulce, kde formule nabývá hodnotu jedna a je tedy splněna. Všechny uzly v nezávislé množině budou mít tedy hodnotu jedna.

# 20 Barevnost grafu a její NP-úplnost

Vstup: Graf $G$ na $n$ uzlech a přirozené číslo $k \leq n$.
Úkol: Zjistit zda je $G$ k–obarvitelný.
$COL \in NPC$

To, že je COL v NP je zřejmé, protože nedeterminismus je v rozhodnutí, jestli uzel obarvit tou nebo onou barvou.

Pro $NPC \in COV$ ukážeme, že $3 - SAT \triangleleft COV$. viz. prednaska.

# 21 Souhrn toho největšího know-how

**Věta 1.:** Uz nemam sil na prepinani ceske a anglicke klavesnice ⇒ nasleduje vynechavani hacku a carek. :-)

**Seznam vedatorskych zkratek**

$h_G(x, y)$ Hranový stupeň - počet prvků nejmenšího hranového řezu mezi x a y.

$h(G)$ nejmensi cislo z cisel $h_G(x, y)$.

$u_G(x, y)$ Uzlový stupeň - počet prvků nejmenšího uzlového řezu mezi x a y.

$u(G)$ nejmensi cislo z cisel $u_G(x, y)$.

$d_G(u, v)$ vzdalenost mezi u, v

$D$ matice vsech $d_G(u, v)$

$d_G^w(u, v)$ vazena vzdalenost mezi u, v

$D^w$ matice vsech $d_G^w(u, v)$

$w(P)$ minimalni cesta

$gh(A)$ genericka hodnost matice

$v(\vec{B}(A))$ párování bigrafu matice

$r_{ij}$ max mozny tok po hrane - nebo tak neco

$\Theta$ rezerva polocesty

$\delta(G)$ nejmensi stupen uzlu v grafu G

$\Delta(G)$ nejvetsi stupen uzlu v grafu G

$\alpha(G)$ největší nezavisla mnozina - nejsou spojeny hranou

$\beta(G)$ nejmensi pokryti - kazda hrana ma jeden konec v mnozine pokryti

$\omega(G)$ Klikovost grafu - nejvetsi uplny podgraf

$\gamma(G)$ nejmensi dominantnost - bud je v mnozine a nebo do ni ma hranu
nejvetsi nezavisla mnozina je i minimalni dominantni.

$\chi(G)$ Chromatické **číslo** - pocet barev na **vrcholy**

$\chi'(G)$ Chromaticky **index** - pocet barev na **uzly**