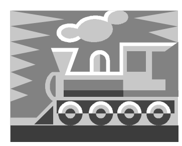
Algoritmické strategie III

I.Kolingerová

Obsah:

5. Dynamické programování

5. Dynamické programování



- Trochu jako D&C, ale jde zdola nahoru, každý podproblém řeší jen 1x
- V tabulce uložena nejlepší dílčí řešení, kombinují se pro získání větších a větších řešení
- Dynamika - pro problémy, kde čas výpočtu a pořadí operací důležité
- Tak trochu kombinace brutální síly a žravé strategie
- Minimalizace úsilí důsledným využíváním dílčích výsledků

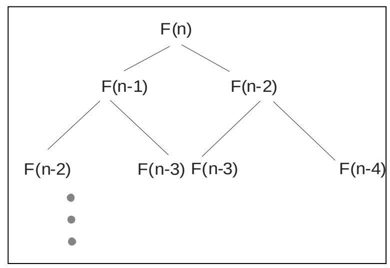
2

Př.1: Rozdíl mezi D&C a DP

Spočtěte n-té Fibonacciho číslo F(n):

F(0)=F(1)=1, F(n)=F(n-1) + F(n-2) pro $n \geq 2$

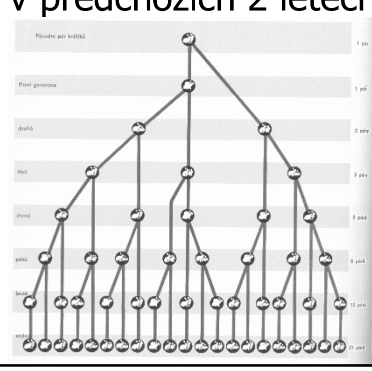
- DP spočte: F(0),F(1),F(2)=F(1)+F(0),...F(n)=F(n-1)+F(n-2) => O(n) pro DP
- D&C spočte:



= > O(2^n) pro D&C

Odbočka: Fibonacciho čísla

- Definoval italský matematik Fibonacci ve 13.st. jako model růstu králičí populace
- Počet párů králíků narozených v daném roce = počtu párů králíků v předchozích 2 letech, na zač. 1 pár



Př.2: Nejdelší rostoucí posloupnost

- Čísla nemusí tvořit souvislý úsek (pokud souvislé – tzv. rostoucí běh)
- Aplikace: pattern matching, permutace
- S=(9,5,2,8,7,3,1,6,4)
- Nejdelší rostoucí posloupnost: buď (2,3,4) nebo (2,3,6), délka 3
- Nejdelší rostoucí běh: (2,8) nebo (1,6), délka 2
- Nalezení běhu: O(*n*)
- Nalezení posloupnosti: Užití DP

Nejdelší rostoucí posloupnost

posloupnost s_i	9	5	2	8	7	3	1	6	4
délka l_i									
předchůdce p_i									

- Nejdelší rost. posl. - připojením čísla *n* k té nejdelší už existující rost. posl., která končí číslem menším než *n*

$$l_i = \max_{0<j<i} l_j + 1 \; kde \; s_j < s_i$$
$$l_0 = 0$$

Největší délka:
$$\max_{1 \le i \le n} l_i$$

Nejdelší rostoucí posloupnost

DP pro posloupnost

posloupnost s_i	9	5	2	8	7	3	1	6	4
délka l_i	1	1	1						
předchůdce p_i	-	-	-						

- Ad 5: exist. posloupnosti: 9, exist. posl. s posledním členem < 5: -, takže vyplníme délku 1, předchůdce 0
- Ad 2: totéž

7

Nejdelší rostoucí posloupnost

DP pro posloupnost

posloupnost s_i	9	5	2	8	7	3	1	6	4
délka l_i	1	1	1	2	2	2			
předchůdce p_i	-	-	-	2	2	2			

- Ad 8: exist. posloupnosti: 9; 5; 2; exist. posl. s posl. členem < 8: 5; 2, takže vyplníme délku 2, předchůdce 5 nebo 2
- Ad 7, 3: obdobně

8

Nejdelší rostoucí posloupnost

DP pro posloupnost

posloupnost s_i	9	5	2	8	7	3	1	6	4
délka l_i	1	1	1	2	2	2	1	3	
předchůdce p_i	-	-	-	2	2	2	-	3	

- Ad 1: exist. posl. s posl. členem < 1: -, takže vyplníme délku 1, předchůdce -
- Ad 6: exist. posl. s posl. členem < 6: 5; 2,3; 1, takže vyplníme délku 3, předchůdce 3
- Ad 4: obdobně

9

Nejdelší rostoucí posloupnost

DP pro posloupnost

posloupnost s_i	9	5	2	8	7	3	1	6	4
délka l_i	1	1	1	2	2	2	1	3	3
předchůdce p_i	-	-	-	2	2	2	-	3	3

- Rekonstrukce posloupnosti: přes předchůdce (z 4 nebo 6 se vrátím na 3, odtud na 2 ...)
- O(n^2) až O(n log n) s použitím datové struktury slovník apod.

10

Př.3: Problém dělení

DP pro dělení

3 pracovníci mají najít informaci skrytou v některé z knih na polici. Chceme rovnoměrnou práci pro všechny tři - rozdělit polici na 3 části, knihy nepřesouvat a nedělit.

Pokud knihy stejně silné, "trivka":

100 100 100 | 100 100 100 | 100 100 100

Pokud ne: 100 200 300 | 400 500 600 | 700 800 900

☺ :-| ☹

Nejrovnoměrnější možné dělení:

100 200 300 400 500 | 600 700 | 800 900

:-| ☺ ☹

11

Problém lineárního dělení

DP pro lineární dělení

Input: Dané seskupení S nezáporných čísel $\{s_1,s_2,\ldots,s_n\}$ a celé číslo k

Output: Dělení S na k nebo méně intervalů, aby max. suma v každém intervalu byla minimální

Heuristika: spočítat průměr. velikost dělení $\boxed{\sum_{i=1}^{n} si/k}$ a pak se pokusit vložit oddělovače, aby blízkost průměru; na někt. vstupech selže - nevyhodnocuje systematicky všechny možnosti

DP: kam dám poslední zarážku

12

Problém lineárního dělení

DP pro lineární dělení

- DP řešení : kam dám poslední zarážku?
- Mezi i-tý a (i+1). prvek
- Cena: max z: (1) ceny posledního dělení $\boxed{\sum_{j=i+1}^{n} sj}$
  (2) nejvyšší ceny doleva od i
- Nejvyšší cena doleva od i: DP pro $\{s_1,s_2,\ldots,s_i\}$ => rekurze

13

Problém lineárního dělení

DP pro lineární dělení

- Formálně: M[n,k] - min. možná cena přes všechna dělení $\{s_1,s_2,\ldots,s_n\}$ do *k* intervalů, kde cena dělení - největší suma prvků v jedné části

O(kn^3)

$$M[n,k] = \min_{i=1}^{n-1} \max(M[i,k-1], \sum_{j=i+1}^{n} s_j)$$

$$kde$$

$$M[1,k] = s_1 \ \forall k > 0$$

$$M[n,1] = \sum_{i=1}^{n} s_i$$

14

Problém lineárního dělení

[100 200 300 400 500 600 700 800 900], *n*=9, *k*=3

s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9

Pokud budeme počítat M[9,3] odzdola přesně podle vztahů bez snahy o úsporu:

M[2,1]=suma s1, s2
M[2,2]=min (max (M[1,1], s2)) - min jen pro i=1
M[2,2]=max(100,200)=200
M[2,3]=max(M[1,2],s2)=max(100,200)=200
M[3,1]=max(suma s1 až s3, s1 až s2 nebo s1)
= suma s1 až s3
M[3,2]=min (max (M[i,1],suma s_{i+1} až s3) pro i=1,2
=min(max(100,500),max(300,300))=300
M[3,3]=min(max(M[1,2],s2+s3),max(M[2,2],s3))
=min(max(300,500),max(200,300))=300

Problém lineárního dělení

To bychom se ale nadřeli!

=> Vyplňuje se do tabulky, mezivýsledky najdeme v tabulce v předchozích řádcích a sloupcích

Problém lineárního dělení

DP pro lineární dělení

[100 200 300 400 500 600 700 800 900], *n*=9, *k*=3

s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9

1. Spočteme částečné součty

P [0,100,300,600,1000,1500,2100,2800,3600,4500]

2. Vyplníme poč. podm.

M[9,3]:

	1	2	3
1	100	100	100
2	300		
3	600		
4	1000		
5	1500		
6	2100		
7	2800		
8	3600		
9	4500		

17

Problém lineárního dělení

DP pro lineární dělení

3. Vyplňujeme max z 2 hodnot:
   - z čísla v minulém sloupci a z rozdílu mezi částečnými součty

Co to znamená: rozdíl částečných součtů v 1. sloupci dá potřebnou dílčí sumu (cena posledního úseku), číslo v minulém sloupci pak „rekurzivní dílčí výsledek" M[i,k-1]

- pro všechny předchozí řádky, z toho minimum

Co to znamená: chceme nejlevnější možné dělení

18

Problém lineárního dělení

DP pro lineární dělení

	1	2	3
1	100	100	100
2	300	200	
3	600		
4	1000		
5	1500		
6	2100		
7	2800		
8	3600		
9	4500		

Vyplňujeme max
z 2 hodnot:
- z čísla v minulém sloupci a z rozdílu mezi částečnými součty –

Pro všechny předchozí řádky,
Z toho minimum

max (M[1,1],p2-p1)=
max (100,200)=200

19

Problém lineárního dělení

DP pro lineární dělení

	1	2	3
1	100	100	100
2	300	200	200
3	600		
4	1000		
5	1500		
6	2100		
7	2800		
8	3600		
9	4500		

Vyplňujeme max
z 2 hodnot:
- z čísla v minulém sloupci a z rozdílu mezi částečnými součty –

Pro všechny předchozí řádky,
Z toho minimum

max (M[1,2],p2-p1)=
max (100,200)=200

20

Problém lineárního dělení

DP pro lineární dělení

	1	2	3
1	100	100	100
2	300	200	200
3	600	300	
4	1000		
5	1500		
6	2100		
7	2800		
8	3600		
9	4500		

Vyplňujeme max z 2 hodnot:
- z čísla v minulém sloupci a z rozdílu mezi částečnými součty –

Pro všechny předchozí řádky,
Z toho minimum

min (
max (M[1,1],p3-p1),
max (M[2,1],p3-p2)) =
min (500,300)=300

21

Problém lineárního dělení

DP pro lineární dělení

	1	2	3
1	100	100	100
2	300	200	200
3	600	300	300
4	1000		
5	1500		
6	2100		
7	2800		
8	3600		
9	4500		

Vyplňujeme max z 2 hodnot:
- z čísla v minulém sloupci a z rozdílu mezi částečnými součty –

Pro všechny předchozí řádky,
Z toho minimum

min (
max (M[1,2],p3-p1),
max (M[2,2],p3-p2))=
min (500,300)=300

22

Problém lineárního dělení

DP pro lineární dělení

Celkem:

	1	2	3
1	100	100	100
2	300	200	200
3	600	300	300
4	1000	600	400
5	1500	900	600
6	2100	1100	900
7	2800	1500	1100
8	3600	2100	1500
9	4500	2400	**1700**

Z toho nejde zjistit,
jak jsme dělili,
jen cena nejdražšího úseku
- uvidíme dál, co s tím

23

Problém lineárního dělení

DP pro lineární dělení

Kromě matice M je třeba plnit matici D – indexy použitých oddělovačů "i", z nich rekurzivně od konce se najde řešení

	1	2	3
1	100	100	100
2	300	200	200
3	600	300	300
4	1000	600	400
5	1500	900	600
6	2100	1100	900
7	2800	1500	1100
8	3600	2100	1500
9	4500	2400	1700

	1	2	3
1			
2		1	1
3		2	2
4		3	3
5		3	4
6		4	5
7		5	6
8		5	6
9		6	7

24

Problém lineárního dělení

DP pro lineární dělení

A nyní trochu přesněji:

25

Problém lineárního dělení

DP pro lineární dělení

Step 1: // Spočítat částečné součty $p[k]=\sum_{i=1}^{k} s_i$

```
Step 1: // Spočítat částečné součty
        p[0] = 0
        for i := 1 to n do p[i] = p[i-1] + si
Step 2: // Inicializovat hraniční podmínky
        for i := 1 to n do M[i,1] = p[i]
        for j := 1 to k do M[1,j] = s1
Step 3: // Hlavní část - vyhodnocení rekurence
        for i := 2 to n do
            for j := 2 to k do begin
                M[i,j] := ∞
                for x := 1 to i-1 do begin
                    s = max (M[x,j-1],p[i]-p[x])
                    if (M[i,j] > s) then
                        begin M[i,j] = s, D[i,j] = x end
                end  end
```

pro rekonstrukci optim. dělení - polohy oddělovačů

26

Problém lineárního dělení

DP pro lineární dělení

Step 4: Rekonstrukce dělení z výsledné matice

procedure ReconstructPartition (S,D,n,k)

if k=1 **then** tisk 1.dělení {s_1,s_2,...,s_n}
else
begin
ReconstructPartition (S,D,D[n,k],k-1)
tisk k-tého dělení {$s_{D[n,k]+1}$,...,s_n}
end

27

Problém lineárního dělení

DP pro lineární dělení

Řešení: D[9,3] -> D[7,2] -> D[5,1] => tisk $s_1..s_5$, return, tisk $s_6..s_7$, return, tisk $s_8..s_9$

	1	2	3
1	100	100	100
2	300	200	200
3	600	300	300
4	1000	600	400
5	1500	900	600
6	2100	1100	900
7	2800	1500	1100
8	3600	2100	1500
9	4500	2400	1700

	1	2	3
1			
2		1	1
3		2	2
4		3	3
5		3	4
6		4	5
7		5	6
8		5	6
9		6	7

28

Problém lineárního dělení

DP pro lineární dělení

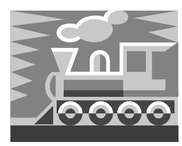
Pokud dělíme [100 100 100 100 100 100 100 100 100], n=9, k=3

M[9,3]:

	1	2	3
1	100	100	100
2	200	100	100
3	300	200	100
4	400	200	200
5	500	300	200
6	600	300	200
7	700	400	300
8	800	400	300
9	900	500	300

	1	2	3
1			
2		1	1
3		1	2
4		2	2
5		2	3
6		3	4
7		3	4
8		4	5
9		4	6

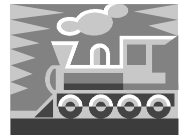
Použitelnost DP



- Problém vhodný pro DP: princip optimality - podstatný je stav, ne způsob, jakým se do něj došlo
- Důležitá je cena operací, ne samotné operace
- Většina kombinatorických problémů
- Největší omezení: počet dílčích řešení, která je nutno sledovat - potřebujeme "zastavovací místa", implicitní pořadí prvků - pokud není splněno, exponenciální počet možných dílčích řešení - nedostupné množství paměti

30

Typický problém pro DP



- Dekomponovatelný do posloupnosti rozhodnutí, přijatých v různých etapách
- Každá etapa má určitý počet možných stavů
- Rozhodnutí nás převádí ze stavu v jedné etapě do stavu v další etapě
- Nejlepší posloupnost rozhodnutí v určité etapě je nezávislá na rozhodnutích v dřívějších etapách
- Cena přechodu mezi stavy v různých etapách je jasně definovaná
- Existuje rekurentní vztah pro výběr nejlepšího rozhodnutí

31