

1) Dynamické programování

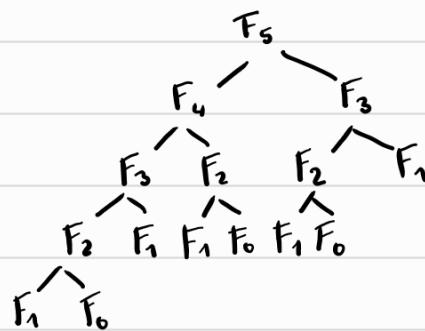
- rozklad na podproblemy → rekurze
- využítí toho, že se podproblemy můžou opakovat

Fibonacciho čísla

vstup: n

1. Pokud $n \leq 1$: vrátím n
2. vrátím $\text{Fib}(n-1) + \text{Fib}(n-2)$

výstup: F_n



→ např. F_2 vypočítáváme 3x

↓

můžu si uložit již získané hodnoty do pole

1. $T[n]$ definováno → return $T(n)$
2. $n \leq 1$: $T[n] \leftarrow n$
3. jinak : $T[n] \leftarrow \text{Fib}(n-1) + \text{Fib}(n-2)$
4. return $T(n)$

! Ize bez rekurze vyplňovat postupně $0, \dots, n$

1. $T[0] \leftarrow 0, T[1] \leftarrow 1$
2. Pro $k=2, \dots, n$
3. $T[k] \leftarrow T[k-1] + T[k-2]$
4. return $T[n]$

Princip dynamického programování

- začnu s exponenciálním rekursivním algoritmem
- odhadem opakování výpočty stejných podproblemů
- vytvořím paměť na ukládání výsledků podproblemů
 - kešování (keš = tabulka na průběžné výsledky)
 - (pro zrychlování rekurze = memoizace)
- rekurzi nahradím vyplňováním keše cyklem ve správném pořadí

Nejdleší rostoucí podposloupnost

posloupnost: x_1, \dots, x_n

NRP(i) \leftarrow délka NRP vybrane $\geq x_i \dots x_n$

1. $d \leftarrow 1$
2. Pro $j = i+1, \dots, n$
3. Pokud $x_i < x_j$:
4. $d \leftarrow \max(d, NRP(j)+1)$
5. return d

→ exponenciální složitost,

pro rostoucí posl. projde všechn 2^n možnosti

přidám keš a budu ji vyplňovat od konce:

$x_0 \leftarrow -\infty$

1. Pro $i = n, n-1, \dots, 0$

2. $d \leftarrow 1$

3. Pro $j = i+1, \dots, n$:

4. Pokud $x_i < x_j$:

5. $d \leftarrow \max(P[j]+1, d)$

6. $P[i] \leftarrow d$

7. vrátim $P[0]$

jde zrychlit na
 $O(n \log n)$
↓
 $O(n \log n)$

→ $O(n^2)$

Editační vzdálenost

def. editační vzdálenost řetězů α, β

= minimální délka editačních operací, které $\neq \alpha$

↳ užela již β

$$\hookrightarrow L(\alpha, \beta) = L(\alpha_1 \dots \alpha_n, \beta_1 \dots \beta_m)$$

editační operace: změna / přidání / smazání 1 znaku

→ BÚNO prováděm operace zleva doprava

Edit(i, j) : $\rightarrow L(\alpha_i \dots \alpha_n, \beta_j \dots \beta_m)$

1. Pokud $i > n$: vrátím $m-j+1$

Pokud $j > m$: vrátím $n-i+1$

2. $l_v \leftarrow 1 + \text{Edit}(i, j+1)$

3. $l_s \leftarrow 1 + \text{Edit}(i+1, j)$

4. $l_z \leftarrow \text{Edit}(i+1, j+1)$

5. Pokud $\alpha_i \neq \beta_j$: $l_z \leftarrow l_z + 1$

6. vrátím $\min(l_v, l_s, l_z)$

→ je exponenciální (např. pro $x=y=a \dots a$)

→ $\Theta(n \cdot m)$ různých volání ($i=1, \dots, n+1$ $j=1, \dots, m+1$)

keš: dvourozměrné pole T

↳ pole závisí na polích pod ním a napravo od

→ vyplňuji zcela, zprava

potemník	
8	3 4
6	4 . . . 6
4	3 . . . 5
2	2 . . . 4
1	1 . . . 3
k	0 . . . k
	8 7 6 . . . 1 0

1. Pro $j = 1, \dots, m+1$: $T[i, m+1] \leftarrow n-i+1$

Pro $i = 1, \dots, n+1$: $T[n+1, i] \leftarrow m-j+1$

2. Pro $i = n, \dots, 1$:

3. Pro $j = m, \dots, 1$

4. Pokud $\alpha_i = \beta_i \cdot \delta < 0$

5. Jinak: $\delta \leftarrow 1$

6. $T[i, j] \leftarrow \min(\delta + T[i+1, j+1], 1 + T[i, j+1], 1 + T[i+1, j])$

výstup $T[1, 1] = L(\alpha, \beta)$

složitost $\Theta(n \cdot m)$

2) Grafové algoritmy

def. tranzitivní uzávěr orientovaného grafu s $V = \{1, \dots, n\}$
je matice 0 a 1 $T \in \{0,1\}^{n \times n}$ t.ž.

$T_{uv} = 1 \Leftrightarrow \exists v \in V \text{ cesta z } u \text{ do } v$

↳ matici dosažitelnosti pro orientované grafy

def. matici sousednosti grafu G s $V = \{1, \dots, n\}$
je matice 0 a 1 $A \in \{0,1\}^{n \times n}$ t.ž.

$A_{uv} = 1 \Leftrightarrow \exists v \in V \text{ hrana z } u \text{ do } v$

mocnina matice A^k :

$A_{uv}^k = 1 \Leftrightarrow \exists v \in V \text{ cesta z } u \text{ do } v \text{ délky } k$

def. matici dosažitelnosti grafu G s $V = \{1, \dots, n\}$

je matice $A^* \in \{0,1\}^*$ t.ž.

$A_{ij}^* = 1 \Leftrightarrow \exists v \in V \text{ cesta z } i \text{ do } j$

získání A^* v $O(n \log n)$ krocích:

→ analogicky tranzitivní uzávěr T

Strassenovo násobení matic

→ rozděl a pánej rekursivní algoritmus

$$X \cdot Y = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \cdot \begin{pmatrix} P & Q \\ R & S \end{pmatrix} = \begin{pmatrix} AP + BR & AQ + BS \\ CP + DR & CQ + DS \end{pmatrix}$$

$$\rightarrow T(n) = 8 \cdot T\left(\frac{n}{2}\right) + \Theta(n^2) \rightarrow \Theta(n^{\log_2 8}) = \Theta(n^3)$$

$$X \cdot Y = \begin{pmatrix} T_1 + T_4 - T_5 + T_7 & T_3 + T_5 \\ T_2 + T_4 & T_1 - T_2 + T_3 + T_6 \end{pmatrix}$$

$$T_1 = (A+D)(P+S)$$

$$T_5 = (A+B) \cdot S$$

$$T_2 = (C+D) \cdot P$$

$$T_6 = (C-A)(P+Q)$$

$$T_3 = A \cdot (Q-S)$$

$$T_7 = (B-D) \cdot (R+S)$$

$$T_4 = D \cdot (R-P)$$

$$T(n) = 7 \cdot T\left(\frac{n}{2}\right) + \Theta(n^2) \rightarrow \Theta(n^{\log_2 7}) \approx \Theta(n^{2.8})$$

Floydov - Warshallov algoritmus

→ konstrukce matice vzdáleností

→ algoritmus založený na dynamickém programování

D_{ij}^k := délka nejkratší cesty z i do j, jejíž vnitřní
vrcholy leží v množině $\{1, \dots, k\}$

→ D_{ij}^0 = délka hranu ij nebo ∞ když hranu neexistuje

→ D_{ij}^n = délka nejkratší cesty i j v G

vstup: matice délek hran D^0

1. Pro $k = 0, \dots, n-1$:

2. Pro $i = 1, \dots, n$:

3. Pro $j = 1, \dots, n$:

4. $D_{ij}^k \leftarrow \min(D_{ij}^k, D_{ik}^k + D_{kj}^k)$

výstup: matice vzdáleností D^n

časová i prostorová složitost $\Theta(n^3)$

→ prostorová lze zlepšit na $\Theta(n^2)$ tím, že budeme

přepisovat pozice 1 matice

def. Orientovaný graf je **slabě souvislý**, pokud zrušením orientace hran získáme souvislý neorientovaný graf

def. Orientovaný graf je **silně souvislý**, pokud $\forall u, v \in V \exists$ orientovaná cesta uv i vu

def. \leftrightarrow je binární relace na vrcholech G t.z.

$x \leftrightarrow y \Leftrightarrow \exists$ orientovaná cesta xy i yx .

relace \leftrightarrow je jistě ekvivalence

komponenta silné souvislosti grafu je podgraf

indukovaný ekvivalencími třídou relace \leftrightarrow

def. **graf komponent** $C(G)$ má za vrcholy komponenty

silné souvislosti grafu G

$\Rightarrow C_i$ do C_j vede hrana $\Leftrightarrow \forall v \in G \exists$ hrana uv t.z.
 $u \in C_i, v \in C_j$

\rightarrow v podstatě kontrakce komponent do vrcholu a
odstranění našobných hran

\rightarrow jistě acyklický graf (DAG)

def. komponenta je **zdrojová**, když do ní nevede hrana

komponenta je **stoková**, když z ní nevede hrana

$$G^T := (V(G), \{uv \mid u, v \in E(G)\})$$

\rightarrow graf, který vznikne otočením orientace všech hran

$C(G)$ je DAG \rightarrow má topologické uspořadání

\rightarrow použijí DFS a seřadí vrcholy podle klesajícího $out(v)$
nejvyšší $out =$ stoková komponenta

vstup: Orientovaný graf G

1. sestrojím G^T $O(n+m)$
 2. $Z \leftarrow$ prázdný zásobník
 3. $\forall v : \text{komp}(v) \leftarrow$ nedefinováno
 4. opakovane' DFS v G^T než projdu cele' G^T $O(n+m)$
→ při opuštění přidám do zásobníku
5. Postupně odebíram ze Z , $\forall v :$
6. Pokud $\text{komp}(v) =$ nedefinováno :
7. pustím DFS(v) v G , ustupují do vrcholu
s komp = nedefinováno :
 $\text{komp} \leftarrow v$

$O(nm)$

výstup: $\forall v \text{ komp}(v)$

čas $\Theta(n+m)$

def. sit' = (G, z, s, c)

G = orientovaný graf (symetrický)

$z, s \in V$ = zdroj a stok

$c: E \rightarrow \mathbb{R}_0^+$ = kapacity hran

def. tok = $f: E \rightarrow \mathbb{R}_0^+$ t.ž.

1) $\forall e \in E : f(e) \leq c(e)$ → přebytek

2) $\forall v \in E, v \neq z, s : f^\Delta(v) = 0$

def. přítok $f^+(v) := \sum_{u \in E} f(uv)$

odtok $f^-(v) := \sum_{u \in E} f(vu)$

přebytek $f^\Delta(v) := f^+(v) - f^-(v)$

velikost toku $|f| := f^\Delta(s) = -f^\Delta(z)$

def. rezerva hraný

$$r(uv) := \underbrace{c(uv) - f(uv)}_{\text{po směru}} + \underbrace{f(vu)}_{\text{proti směru}}$$

def. čistý tok f^* k toku f : $f^*(uv) := f(uv) - f(vu)$

$$\star f^*(v) = \sum_{u \in E} f^*(uv) = 0 \quad \text{pro } \forall v \neq z, s$$

def. sit' rezerv $R(S, f)$ k siti $S = (V, E, z, s, c)$ a toku f

$$R(S, f) = (V, E, z, s, r)$$

↳ rezerva místo kapacity

↓

$$f \text{ tok}, g \text{ tok} \vee R(S, f) \rightarrow f' = f + g \text{ tok } \vee S \\ (\text{bez sestroyit } \vee O(m))$$

def. tok g je blokující $\equiv \forall P$ cesta $z-s \quad \exists e \in P: g(e) = c(e)$

def. siti je pročistěná (vrstevnatá) $\equiv \forall v, e$ leží na nejkratších
cestách $z-s$

Dinicův algoritmus

1. $f \leftarrow 0$

2. Opakujeme:

3. $R \leftarrow$ sit' rezerv $R(S, f)$, smažme hraný s $r(e) = 0$

for

4. Pročistíme R

5. $l \leftarrow$ délka nejkratší cesty $z-s \vee R$

pokud $l = +\infty$: skončím

6. $g \leftarrow$ blokující tok $\vee R$

7. zlepším f pomocí g

čistění sítě

$O(m)$

1. BFS ze zdroje \rightarrow rozdělení vrcholů do vrstev
2. Smazu vrstvy za stokem
3. Smazu hranu zpět a uvnitř vrstvy
4. F fronta $\leftarrow \{v \neq s \mid \deg_{out}(v) = 0\}$
5. Dokud $F \neq \emptyset$
6. $v \leftarrow F.pop()$
7. smazu v a hranu do něj
8. nejaky $\deg(v) = 0 \rightarrow F.push(v)$

blokující tok

$O(n \cdot m)$

1. $g \leftarrow 0$
 2. Dokud \exists cesta $P \neq \emptyset$ do $s \rightarrow$ nejvýš m iterací
 3. $\epsilon \leftarrow \min_{e \in P} c(e) - g(e)$
 4. $\forall e \in P: g(e) \leftarrow g(e) + \epsilon$
 5. Pokud $g(e) = c(e):$
smazu $e \in R$
 6. Dokončení sítě \rightarrow vše složitost $O(m)$
- $O(n)$
 \hookrightarrow k vrcholu jakékoli
odbočce zí hranu

1 fáze $O(nm)$
fází $O(n)$ } $O(n^2m)$

def. vlna v síti je fct $f: E \rightarrow \mathbb{R}_0^+$ t.z.

$$1) \forall e: f(e) \leq c(e)$$

$$2) \forall v \neq s, t: f^*(v) \geq 0$$

převedení přebytku $\geq u$ do v , $f^*(u) > 0$, $r(u, v) > 0$

$$\epsilon \leftarrow \min(f^*(u), r(u, v))$$

$$f^*(uv) \leftarrow f^*(uv) + \epsilon$$

výška $h: V \rightarrow \mathbb{N}$

! 3. podmínka převedení: $h(u) > h(v)$

Goldbergův algoritmus

1. $\forall v : h(v) \leftarrow 0, h(z) \leftarrow n$
2. $\forall e : f(e) \leftarrow 0, \forall z_u \in E : f^*(z_u) \leftarrow c(z_u)$
3. Dokud $\exists u \neq z, s, f^*(u) > 0$:
4. Pokud $\exists u,v \in E, r(u,v) > 0 \wedge h(u) > h(v)$
5. Převedeme po uv
6. Jinak :
7. $h(u) \leftarrow h(u) + 1$

↪ f je vlna

def. převedení je

nasycené, když $r(u,v)$ klesne na 0

nenasycené, když $f^*(u)$ klesne na 0

	# operací	čas na operaci
zvednutí	$O(n^2)$	$O(n)$
nasycené převedení	$O(nm)$	$O(1)$
nenasycené převedení	$O(n^2m)$	$O(1)$

$$\rightarrow O(n^2m)$$

3) Algoritmy vyhledávání v textu

abeceda = Σ = konečná množina znaků

Σ^* = množina všech řetězců nad Σ

$\alpha, \beta, \gamma, \dots$ = řetězce

x, y, z, \dots = znaky = jednoznačkové řetězce

$|\alpha|$ = délka řetězce = # jeho znaků

ϵ = prázdný řetězec $\rightarrow |\epsilon| = 0$

zřetězení = $\alpha\beta$ = slenčí α a β za sebe

$\alpha[i]$ = i-tý znak řetězce (indexované od 0)

$\alpha[i:j]$ = podřetězec $\alpha[i] \dots \alpha[j-1]$

$\alpha[:j]$ = prefix (předpona)

$\alpha[i:]$ = suffix (přípona)

problem:

vstup: seno Š $\rightarrow S := |\mathbf{\check{S}}|$

ježka Z $\rightarrow J := |\mathbf{\check{Z}}|$

výstup: $\{i \mid \mathbf{\check{S}}[i:i+J] = \mathbf{\check{Z}}\}$

\rightarrow index na kterých v Š začíná Z

Knuthův - Morrisův - Prattův algoritmus

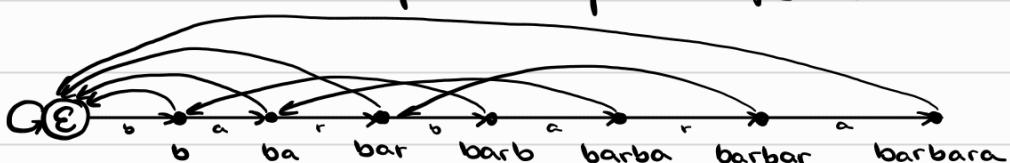
vyhledávací automat:

orientovaný graf:

verholy = stavy automatu = prefiry jenky

hrany \rightarrow dopředne = prodloužení prefixu

zpětné = podle zpětné fce



KMP:

Krok (i, x):

1. Dokud $\pi[i] \neq x$
2. je-li $i = 0$: vrátíme 0
3. $i \leftarrow \pi[i]$ (stav do kterého z i vede zpětná hrana)
4. vrátíme $i + 1$

Hledaj (G):

1. $i \leftarrow 0$
2. Pro $j = 0, \dots, |G| - 1$
3. $i \leftarrow \text{Krok}(i, G[j])$
4. Pokud $i = |\pi|$:
5. nalezený výskyt na $j - |\pi| + 1$

složitost Hledaj: $\Theta(s)$

dopředních hran $\leq s$

zpětných h. \leq # dopředních hran $\leq s$

Konstrukce automatu:

1. $Z[0] \leftarrow 0, Z[1] \rightarrow 0$
2. $i \leftarrow 0$
3. Pro $j = 1, \dots, s$: ?? indexy ??
4. $i \leftarrow \text{Krok}(i, Z[j-1])$
5. $Z[j] \leftarrow i$

čas $\Theta(s)$

\rightarrow celkový čas algoritmu $\Theta(s + s)$

lze zobecnit pro více jehel $\rightarrow \Theta(s + \sum_j l_j + v)$

\hookrightarrow #výskytů

Aho-Corasickova

= zábezpečení KMP

jetby z_1, \dots, z_n

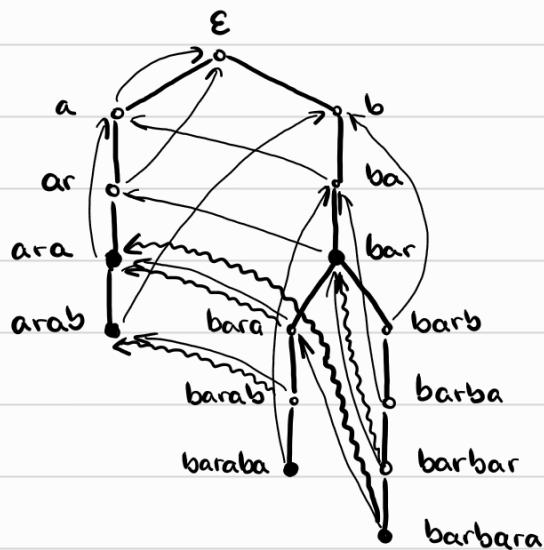
seno 6

stavy = prefixy všech jeho

hraný mezi stavy:

- dopředné → prodloužení prefixu jehly o 1 znak
 $\alpha \rightarrow \alpha x$
 - zpětné → nejdelší vlastní suffix α , který je stavem
 - zkratkové → nejbližší koncový stav po zpětných h.
(usnadňují vypisování jehel)

př. jehly: Ara, bar, arab, barbara, baraba



reprézentace automatu:

- stary očíslujeme ($0 = E$ (kořen))
 - Slovo (i) := která jehla končí v i
 - Zpět (i) := kam vede zpětná h.
 - Zkratka (i) := kam vede zkratka
 - Dopředu (i, x) := dopředná hrana pro znak x

i = stay

Krok(s, x):

1. Dokud $Dopředn(s, x) = \emptyset$
2. pokud $s = 0$: vrátíme s
3. $s \leftarrow Zpět(s)$
4. vrátíme $Dopředn(s, x)$

Hledej(G):

1. $s \leftarrow kořen$
 2. Pro $i = 0, \dots, S-1$
 3. $s \leftarrow Krok(s, G[i])$
 4. $t \leftarrow s$
 5. Dokud $t = \emptyset$
 6. Pokud $Slovo(t) \neq \emptyset$
 7. hlaším výskyt
 8. $t \leftarrow Zkratka(t)$
-] # výskytů

$\Theta(S + \# výskytů)$

Vytvoření automatu:

1. vybudoji trii pro $l_1, \dots, l_n \rightarrow$ dopředné hrany, $r =$ kořen
2. $Zpět(r), Zkratka(r) \leftarrow \emptyset$
3. $\forall s, syn\ kořene : Zpět(s) \leftarrow r, Zkratka(s) \leftarrow \emptyset$
4. $F \leftarrow fronta se syny kořene$
5. Dokud $F \neq \emptyset$:
6. $v \leftarrow Dequeue(F)$
7. pro $\forall s, syn\ v$:
8. $Krok(Zpět(v), pism. na hraně vs) \rightarrow Zpět(s)$
9. $F.Enqueue(s)$
10. Pokud $Slovo(Zpět(s)) \neq \emptyset : Zkratka(s) \leftarrow Zpět(s)$
11. Jinak: $Zkratka(s) \leftarrow Zkratka(Zpět(s))$

4) Algebraické algoritmy

polynom $P(x) := \sum_{i=0}^{n-1} p_i \cdot x^i$, velikost polynomu = n

$\deg(P) = \max \{i \mid p_i \neq 0\}$
 $= -1$ pro nulový polynom

násobení polynomů:

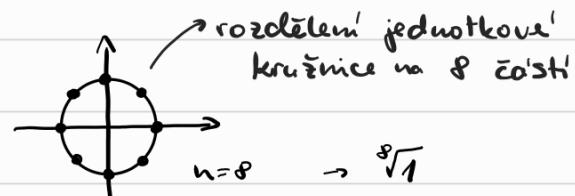
$$P(x) \cdot Q(x) = \sum_{j=0}^{n-1} P_j x^j \cdot \sum_{k=0}^{m-1} Q_k x^k = \sum_{j,k} P_j Q_k x^{j+k} = \\ = \sum_{t=0}^{n+m-2} \left(\sum_{i=0}^t P_i Q_{t-i} \right) x^t$$

$$\deg(P \cdot Q) = \deg P + \deg Q$$

\rightarrow vynásobení triviálně $O(n \cdot m)$

def. číslo ω je primitivní n-tá odmocnina $\geq 1 \equiv$
 $\equiv \omega^n = 1 \wedge \omega^1, \dots, \omega^{n-1} \neq 1$

př. $e^{\frac{2\pi i}{n}}$, $e^{-\frac{2\pi i}{n}}$
 \downarrow
 ω , $\omega^{-1} = \bar{\omega}$



n-tých odmocnin ≥ 1 je max n

def. diskrétní Fourierova transformace (DFT)

je $F: \mathbb{C}^n \rightarrow \mathbb{C}^n$ t.z. (lineární zobrazení)

$$\vec{x} \rightarrow \vec{y} \text{ kde } V_j: y_j = \sum_{k=0}^{n-1} x_k \cdot \omega^{jk}$$

linearní kombinace

\hookrightarrow jde zapsat jako násobení matic

$$1 \rightarrow \omega = \sqrt[n]{1}$$

$$F(p_0, \dots, p_{n-1}) = (P(\omega^0), \dots, P(\omega^{n-1}))$$

$$F(\vec{x}) = \Omega \cdot \vec{x} \rightarrow \text{maticové vyjádření}$$

$$\Omega_{jk} = \omega^{jk}$$

Rychlá Fourierova transformace (FFT)

→ rekursivní algoritmus na výhodnocení polynomu

vstup: $n = 2^k$, ω primitivní $\sqrt[n]{1}$
 (p_0, \dots, p_{n-1}) koeficienty polynomu
výstup: (y_0, \dots, y_{n-1}) t.ž. $y_j = P(\omega^j)$

1. Pokud $n=1$: vrátím $y_0 = p_0$
2. $(s_0, \dots, s_{\frac{n}{2}-1}) \leftarrow \text{FFT}(\frac{n}{2}, \omega^2, (p_0, p_2, \dots, p_{n-2}))$
 $(l_0, \dots, l_{\frac{n}{2}-1}) \leftarrow \text{FFT}(\frac{n}{2}, \omega^2, (p_1, p_3, \dots, p_{n-1}))$
↳ rozdělení na sude' a liche'
3. Pro $j = 0, \dots, \frac{n}{2}-1$:
 $y_j \leftarrow s_j + \omega^j \cdot l_j$
 $y_{\frac{n}{2}+j} \leftarrow s_j - \omega^j \cdot l_j$

časová složitost $\Theta(n \log n)$

Násobení polynomů

vstup: P, Q polynomy velikosti n

BÚNO horních $\frac{n}{2}$ koeficientů 0 → PQ velikosti n

1. nařazíme různá čísla x_0, \dots, x_{n-1}

2. $(P(x_0), \dots, P(x_{n-1}))$] $O(n^2)$
 $(Q(x_0), \dots, Q(x_{n-1}))$

3. $\forall i \in \{0, \dots, n-1\}: R(x_i) \leftarrow P(x_i) \cdot Q(x_i)$

4. spočítání koeficientů R z x_i a $R(x_i)$
(FFT)

→ krok 2 s DFT trvá $\Theta(n \log n)$

5) RSA

def. eulerova fce $\varphi(n) = \#\text{ čísel } k \in \{1, \dots, n-1\} \text{ t.ž. } \text{NSD}(k, n) = 1$

$$n = p_1^{k_1} \cdot \dots \cdot p_m^{k_m} \quad \text{pručitelný rozklad}$$
$$\varphi(n) = p_1^{k_1-1} (p_1-1) \cdot \dots \cdot p_m^{k_m-1} (p_m-1)$$

Eulerova věta: $a^{\varphi(n)} \equiv 1 \pmod{n} \quad \text{NSD}(a, n) = 1$

Malá Fermatova věta: $a^{p-1} \equiv 1 \pmod{p} \quad p \nmid a$

p, q nesoudělná pručísla

$$N = p \cdot q, \quad \varphi(N) = (p-1)(q-1)$$

e nesoudělné s $\varphi(N)$

$$d: d \cdot e \equiv 1 \pmod{\varphi(N)}$$

→ d spočítám Euklidovým algoritmem

veřejný klíč: (N, e)

soukromý klíč: (d, p, q)

zpráva: x t.ž. $0 < x < N, \quad \text{NSD}(x, N) = 1$

zašifrování: $y = x^e \pmod{N}$

dešifrování: $x = y^d \pmod{N}$

dk. $y^d = (x^e)^d \equiv x^{ed} \equiv x^1 \equiv x \pmod{N}$

P

$$ed \equiv 1 \pmod{\varphi(N)}$$

6) Aproximační algoritmy

- pro problémy, pro které je těžké najít přesné řešení
 - pro maximizační problémy:
chci alespoň α -násobek optima pro $0 < \alpha < 1$

poměrová chyba = $\frac{c'}{c^*}$ pro max. problém, $\frac{c^*}{c'}$ pro min. problém

relativní chyba = $\frac{|c' - c^*|}{c^*}$, c^* = optimum, c' = přípustné řešení

def. **aproximační schéma** pro optimalizační úlohu X je
algoritmus A t.ž. $vstup(A) =$ zadání k úlohy X , $\epsilon > 0$,
který pro libovolné permutace e pracuje jako approximační
algoritmus pro X s relativní chybou e

Problém obchoduho cestujícího

vstup: $V = \{1, \dots, n\}$
 $d: V \times V \rightarrow \mathbb{R}_0^+$ (vzdálenosti)

} vstup, graf

výstup: $\tilde{\sigma}$ permutace na V

cíl: minimalizace $\sum_{i=1}^{n-1} d(\sigma(i), \sigma(i+1)) + d(\sigma(n), \sigma(1))$

↳ předpoklad: d je metrika

tzn. symetrie, Δ nerovnost, $d(u, v) = 0 \Leftrightarrow u = v$

1. $T =$ minimální kostra

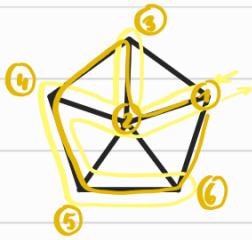
2. $E = T \cup T$

↳ disjunktní sjednocení

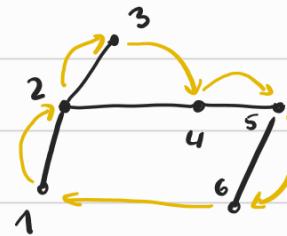
↳ nahrazení ℓ hran ℓ kopie $\ell \rightarrow 2n - \ell$ hran

3. DFS → tah přes ℓ vrcholy

4. vynechávám opakující se vrcholy



→ Kostra



nalezená kružnice není delší než dvojnásobek optima



2-aproximační algoritmus

Lemma: $d(T) \leq \text{OPT}$

Dle.

bud' C nějaké přípustné řešení (např. OPT)



$D = C$ po odebrání 1 hrany

D je kostra : $d(T) \leq d(D) \leq d(C)$

↳ minimality T

□

Věta: Kostrový algoritmus je 2-aproximační

Dle.

$d(C) \leq d(E) (=) 2 \cdot d(T) \leq 2 \cdot \text{OPT}$

↳ def. $E = T \cup T$

□

Problém batohu

vstup: kapacita batohu H

ceny c_1, \dots, c_n
vahy w_1, \dots, w_n

} n předmětů

výstup: $\max \left\{ \sum_{j \in S} c_j \mid \sum_{j \in S} w_j \leq H \right\}$

myšlenka: kvantování cen

vyberu c_{\max} z cen c_i a nějaké $M \in \mathbb{N}$, $M < c_{\max}$

$f_i : c_i \rightarrow c_i \cdot \frac{M}{c_{\max}} + \text{zaokrouhlení}$

1. Odstraním předměty s $w_i > H$
2. Spočítám $c_{\max} = \max_i c_i$, $M = \lceil \frac{n}{\varepsilon} \rceil$
3. $\forall i: \hat{c}_i \leftarrow \lfloor c_i \cdot \frac{M}{c_{\max}} \rfloor$
4. dynamickým programováním vyřeším problem
pro upravené ceny (r_i a H zůstávají stejné)

$$\rightarrow \text{čas } O(n \hat{c}), \hat{c} \leq n \cdot M = O\left(\frac{n^2}{\varepsilon}\right)$$

$$\rightarrow O\left(\frac{n^3}{\varepsilon}\right)$$

↓

$(1-\varepsilon)$ -approximace pro libovolné $\varepsilon > 0$

7) Paralelní třídění pomocí komparátorových sítí

konečná abeceda Σ

hradlo : $f: \Sigma^k \rightarrow \Sigma$ $\xrightarrow{k} \boxed{f} \rightarrow$

unární \rightarrow konstanty (0,1), unární \rightarrow identita, negace

binární \rightarrow AND, OR, XOR, ...

hradlová síť \rightarrow kombinační / booleovský obvod

- vrcholy: vstup, výstup, hradla

hradlům přiřazujeme fci a aritu

- hrany: vstup ($\deg^{in}=0$), výstup ($\deg^{in}=1, \deg^{out}=0$),

hradlo : $\deg^{in} = \text{arita}$, $\deg^{out} > 0$

(vstupní hrany jsou očíslované)

\hookrightarrow DAG

- výpočet probíhá v taktích

$t=0$: výstup vydají vstupní porty a unární hradla

$t=i$: výstup vydají hradla, jejichž všechny vstupy kž jsou definované

\rightarrow vstupy N_0, N_1, \dots

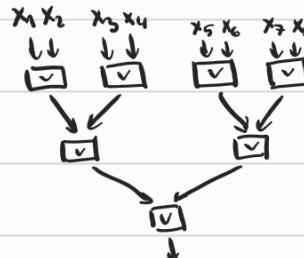
N_i : vrcholy, které vydají výstup poprvé
v čase i

čas = # vstupů = # taktů výpočtu

prostor = # hradel

př. vstup : x_1, \dots, x_n

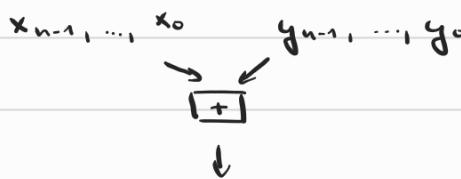
výstup : $x_1 \vee \dots \vee x_n$



hloubka $\Theta(\log n)$

velikost $\Theta(n)$

Sčítání



c_i = přenos do i -tého řádku $\approx (i-1)$.

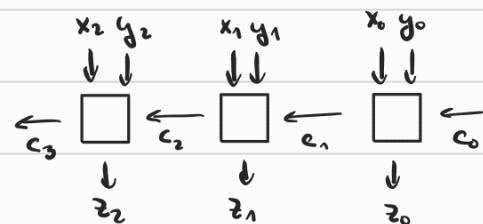
$$c_0 = 0$$

$$z_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = \text{maj}(x_i, y_i, c_i) = (x_i \cdot y_i) \vee (y_i \cdot c_i) \vee (c_i \cdot x_i)$$

\hookrightarrow majorita

sekvenční výpočet



\hookrightarrow hloubka i velikost $\Theta(n)$



souvislost c_i in a c_{i+1}

a) identita <

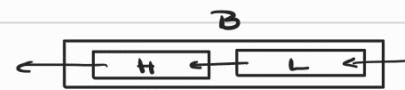
b) negace (nenastane)

c) konstanta 0

d) konstanta 1

1-bit blok $c_{i+1} \leftarrow$

0	0	1
0	<	1
1	<	1



H	0	0	1	<
1	1	1	1	1
<	0	1	<	

a) $(p, q) = (1, *)$

$$p = x \oplus y$$

$$p_B = p_H \& p_L$$

c) $(p, q) = (0, 0)$

$$q_F = x$$

$$q_B = (p_H \& q_L) \vee (\neg p_H \& q_H)$$

d) $(p, q) = (0, 1)$

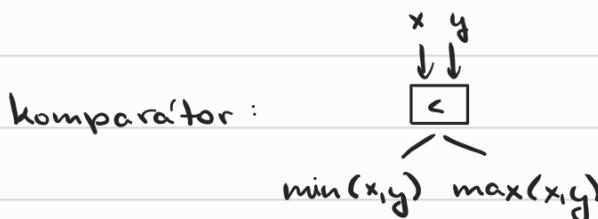
1. výpočet chování kanonických bloků

hloubka $\Theta(\log n)$, velikost $\Theta(n)$

2. zahrnutování přenosů

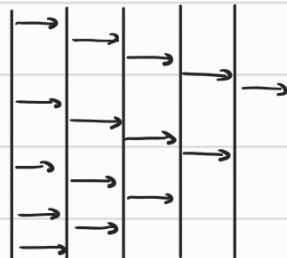
hloubka $\Theta(\log n)$, velikost $\Theta(n)$

Třídič komparátorová síť

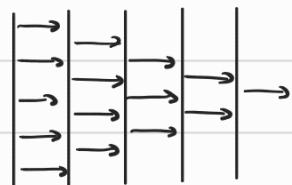


s malou újimou na obecnosti předp. že síť se nevětuje

Bubble Sort



po taktach :



hloubka $\Theta(n)$

velikost $\Theta(n^2)$

def. posloupnost x_0, \dots, x_{n-1} je bitomická =

= $\exists i, j : x_i < x_{i+1} < \dots < x_{i+j} > x_{i+j+1} > \dots > x_{i+n-1}$

↪ indexování mod n



separátor S_n

→ hloubka $O(1)$

velikost $\Theta(n)$

bitomická posloupnost

S_n

$\frac{n}{2}$

$\frac{n}{2}$

bitomické posloupnosti

bitomická třídička B_n

→ řada separátorů

↪ $\Theta(\log n)$ hladin

→ hloubka $\Theta(\log n)$, velikost $\Theta(n \log n)$

bitomická

B_n

↓

n rostoucí

slečvačka M_n

hloubka $\Theta(\log n)$

velikost $\Theta(n \log n)$

\rightarrow lze z nich postavit paralelní Merge Sort



rostoucí

\rightarrow

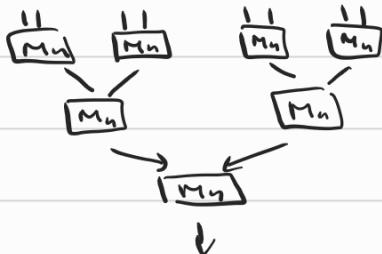


$2n$

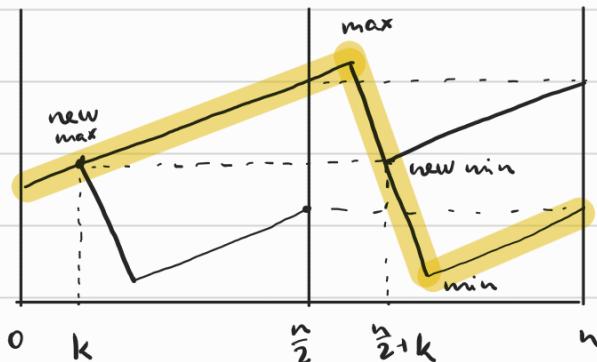
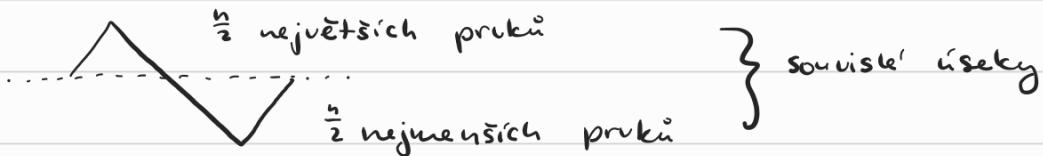
třídička T_n

hloubka $\Theta(\log^2 n)$

velikost $\Theta(n \log^2 n)$



separátor: porovnává průký i a $\frac{n}{2}+i$



pro $i < k$ komparátor neprohazuje
pro $i \geq k$ prohazuje
(BÚNO k v l. polovině)

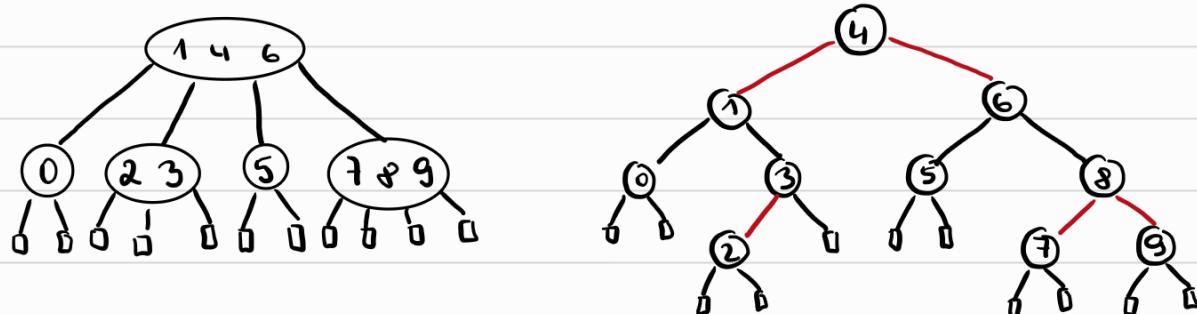
8) Červeno-černé stromy a jejich vyvažování

def. LLRB (left leaning red-black tree)

je BVS s vnitřními vrcholy, jehož hranы jsou obarvenы červené a černé t.ž.

- 1) nejsou 2 červené hranы bezprostředně nad sebou
- 2) pokud z vrcholu vede 1 červená hrana, pak vede doleva
- 3) hranы vedoucí do listů jsou černé
- 4) na A cestě kořen-list leží stejný # černých hran

↔ bijekce mezi $(2,4)$ -stromy a LLRB



list



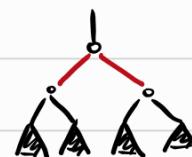
2-vrchol



3-vrchol



4-vrchol



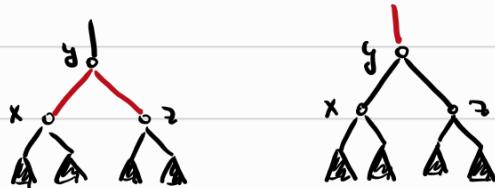
Operace:

• rotace červené hranы



zachová černá axiomy, ale může rozbít červené

- přebarvení 4-vrcholu:



zachová černé axiomu, ale může rozbit červené

Insert:

- při hledání směrem dolů přebarvují 4-vrcholy
→ rozbiti červené axiomu
- nahradím list ⚡ → 
- směrem nahoru opravují červené axiomu rotací hrany



$\Theta(\log n)$