

1) Časová složitost algoritmů

def. časová náročnost algoritmu je počet kroků výpočtu
krok výpočtu = elementární operace proveditelná
v konstantním čase

prostorová náročnost algoritmu je rozsah pracovní paměti
použité při běhu algoritmu

velikost vstupu:

- délka řetězce (# znaků)
- počet prvků posloupnosti
- hodnota čísla na vstupu (př. Euklidův algoritmus)
- počet vrcholů / hran grafu
- řád matice
- délka čísla v dvojkové soustavě ($\lfloor L \log_2 N \rfloor + 1$)

časová složitost = funkce, která pro \forall velikost vstupu N
vrátí délku nejdélšího výpočtu přes
všechny vstupy délky N

prostorová složitost = \exists vrátí největší prostorovou náročnost
přes výpočty nad všemi vstupy
délky N

- v nejhorsím případě \rightarrow maximální délka výpočtu
- v nejlepším případě \rightarrow minimální délka výpočtu
- v průměrném případě \rightarrow průměr z délek výpočtů nad
všemi vstupy velikosti n

Asymptotická notace

O ... horní odhad

Ω ... dolní odhad

$f(n)$... fce složitosti pro vstup velikosti n

$f(n)$ je třídy $O(g(n)) \equiv$

$$\exists c > 0 \ \exists n_0 \ \forall n > n_0 : 0 \leq f(n) \leq c \cdot g(n)$$

$f(n)$ je třídy $\Omega(g(n)) \equiv$

$$\exists c > 0 \ \exists n_0 \ \forall n > n_0 : 0 \leq c \cdot g(n) \leq f(n)$$

$f(n)$ je třídy $\Theta(g(n)) \equiv f(n) \in O(g(n)) \wedge f(n) \in \Omega(g(n))$

1) formule $f(n)$

2) ponechám pouze nejrychleji rostoucí člen $f(n)$

3) seškrtnu multiplikaci konstanty

2) Třídy složitosti

def. rozhodovací problem je $f: \{0,1\}^* \rightarrow \{0,1\}$

vstup: konečná posl. 0 a 1

výstup: ano / ne

def. P je třída (množina) rozhodovacích problemů, které jsou řešitelné v polynomálním čase

↓

problem $L \in P \Leftrightarrow \exists$ algoritmus A a polynom f t.ž.

\forall vstup x A dležíne f(|x|) a

vydaří výsledek $A(x) = L(x)$

def. NP je třída rozhodovacích problemů t.ž.

$L \in NP \Leftrightarrow \exists$ problem K $\in P$ a polynom g t.ž.

$\forall x : L(x) = 1 \Leftrightarrow$ pro nějaký řetězec y

délky nejvýš $|g(x)|$

$K(x,y) = 1$

y = polynomálně dlouhá nařízená

def. Problem je NP-těžký je-li na něj převoditelný

každý problem $\in NP$

Problem je NP-úplný je-li NP-těžký a leží v NP

def. A,B rozhodovací problemy

A je převoditelný na B ($A \rightarrow B$) \Leftrightarrow

$\exists f: \{0,1\}^* \rightarrow \{0,1\}^*$ t.ž. $\forall x \in \{0,1\}^* : A(x) = B(f(x))$

a lze ji spočítat v čase polynomálním k |x|

f = převod / redukce

L: Pokud $A \rightarrow B$ a B je řešitelný v polynomialním čase,
pak je A také řešitelný v polynomialním čase

NP-úplné problémy:

- SAT \rightarrow splnitelnost logických formulí v CNF
- 3SAT \rightarrow $\neg\neg\ldots$, kde klausule obsahuje ≤ 3 literály
- 3,3SAT \rightarrow $\neg\neg\ldots$, kde proměnná se vyskytuje max 3x
- nezávislá množina v grafu velikosti k
- klika v grafu velikosti k
- obarvitelnost grafu $k \geq 3$ barvami
- Hamiltonovská cesta
- Hamiltonovská kružnice
- podmnožina množiny $\subseteq \mathbb{N}$ s daným součtem
- rozdělení množiny na 2 se stejným součtem
- batoh \rightarrow ceny a váhy předmětů, kapacita batohu,
 \exists podmnožiny předmětů t.ž. vaha nepřesahuje
kapacitu a cena je \geq zadanému číslu

Převody mezi problémy:

- klika \leftrightarrow nez. množina

klika = nezávislá množina v doplňku grafu

- 3,3-SAT \rightarrow 3-SAT \rightarrow SAT
převod identitou ($3,3\text{-SAT} \Leftrightarrow \text{SAT}$)

- SAT \rightarrow 3-SAT

klauzule $(l_1 \vee \dots \vee l_k)$ pro $k > 3$

$$\rightarrow (z \vee l_1 \vee l_2) \wedge (\neg z \vee l_3 \vee \dots \vee l_k)$$

- nová proměnná z
 opakuj: dokud nemáme 3-SAT

složitost: $O(k)$ za krok, $O(k)$ kroků na klauzuli
 n klauzuli délky $k \rightarrow O(n^3)$

- 3-SAT \rightarrow 3,3-SAT

proměnná $x \in S$ $t > 3$ výskytů

\rightarrow nahradíme proměnnými x_1, \dots, x_t

\rightarrow zajistíme stejné ohodnocení:

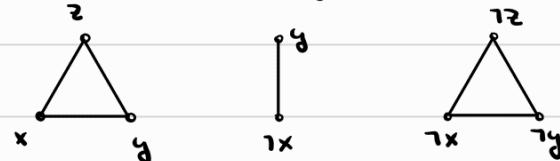
$$x_1 \rightarrow x_2, x_2 \rightarrow x_3, \dots, x_{t-1} \rightarrow x_t, x_t \rightarrow x_1$$

$$(x_1 \rightarrow x_2 = \neg x_1 \vee x_2)$$

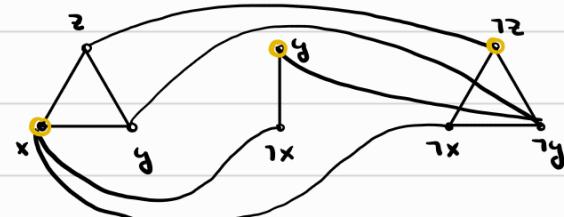
- 3-SAT \rightarrow nezávislá množina

- vytvoření grafu z klauzule v CNF

$$(x \vee y \vee z) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y \vee \neg z)$$



+ hrany muzí proměnnými a jejich negacemi



př. nezávislá množina

formule je splnitelná \Leftrightarrow \exists nezávislá mn. velikosti $\geq k$
 $(k = \# \text{ klauzul})$

◦ nezávislá množina \rightarrow SAT

- graf s vrcholy $1, \dots, n$

1. sada $\left\{ \begin{array}{l} \text{- proměnné: } x_1, \dots, x_n \text{ t.ž.} \\ x_i = 1 \Leftrightarrow \text{vrchol } i \text{ patří nezávislé množině} \\ \forall \{i, j\} \in E : \\ \text{klausule } \neg(x_i \wedge x_j) = \neg x_i \vee \neg x_j \end{array} \right.$

2. sada $\left\{ \begin{array}{l} \text{- zajistění velikosti nezávislé množiny} \\ \rightarrow \text{vrcholy nezávislé množiny očíslováme} \\ \text{- proměnné } p_{ij} \text{ pro } 1 \leq i \leq k, 1 \leq j \leq n \\ p_{ij} = 1 \Leftrightarrow j \text{ je } i\text{-tý vrchol nezávislé množiny} \\ (p_{ij} \rightarrow \neg p_{i'j}) \text{ pro } 1 \leq i \neq i' \leq k, 1 \leq j \leq n \\ \rightarrow j \text{ se neopakuje} \\ (p_{ij} \rightarrow \neg p_{ij'}) \text{ pro } 1 \leq i \leq k, 1 \leq j \neq j' \leq n \\ \rightarrow \max 1 i\text{-tý prvek} \\ (p_{i1} \vee p_{i2} \vee \dots \vee p_{in}) \quad 1 \leq i \leq k \\ \rightarrow \text{alespoň 1 } i\text{-tý prvek} \end{array} \right.$

$(p_{ij} \rightarrow x_j) \rightarrow$ propojení obou sad

3) Metoda rozděl a panuj

Rekurzivní dělení:

- problém rozdělíme na menší podproblemy
- použijeme algoritmus na $\#$ podproblemu
- dělíme, dokud nedojde do elementárního stavu

př. $\text{fib}(n) \rightarrow$ Fibonacciho číslo $F_n = F_{n-1} + F_{n-2}$

if $n \leq 1$:

 return n

else:

 return $\text{fib}(n-1) + \text{fib}(n-2)$

výpočet časové složitosti:

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \Theta(n^c)$$

$a = \#$ podproblémů

$b = \text{faktor zmenšení}$

$c = \text{lokální práce}$

Kuchařkova věta:

$$\text{Rekurence } T(n) = a \cdot T\left(\frac{n}{b}\right) + \Theta(n^c), \quad T(1) = 1$$

pro $a \geq 1, b > 1, c \geq 0$

má řešení:

- 1) $\Theta(n^c \log n)$ pro $\frac{a}{b^c} = 1$
- 2) $\Theta(n^c)$ pro $\frac{a}{b^c} < 1$
- 3) $\Theta(n^{\log_b a})$ pro $\frac{a}{b^c} > 1$

Merge Sort

vstup: posloupnost a_1, \dots, a_n

1. Pokud $n = 1$:

2. return a_1

3. $x_1, \dots, x_{\lfloor \frac{n}{2} \rfloor} \leftarrow \text{Merge Sort } (a_1, \dots, a_{\lfloor \frac{n}{2} \rfloor})$

4. $y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor} \leftarrow \text{Merge Sort } (a_{\lfloor \frac{n}{2} \rfloor + 1}, \dots, a_n)$

5. $b_1, \dots, b_n \leftarrow \text{Merge } (x_1, \dots, x_{\lfloor \frac{n}{2} \rfloor}, y_1, \dots, y_{\lfloor \frac{n}{2} \rfloor})$

↳ slévání 2 sestříděných posloupností

výstup: sestříděná posloupnost b_1, \dots, b_n

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n) \rightarrow a=2, b=2, c=1 \rightarrow \Theta(n \log n)$$

Násobení dluhých čísel

čísla $X, Y \rightarrow n$ ciferná, BÚNO n je mocnina 2

$$X = A \cdot 10^{\frac{n}{2}} + B \quad \begin{array}{|c|c|} \hline & \frac{n}{2} \text{ cífer} & \frac{n}{2} \text{ cífer} \\ \hline A & & B \\ \hline & n \text{ cífer} & \\ \hline \end{array}$$

$$Y = C \cdot 10^{\frac{n}{2}} + D$$

$$X \cdot Y = AC \cdot 10^n + ((A+B)(C+D) - AC - BD) \cdot 10^{\frac{n}{2}} + BD$$

vstup: n -cíferná čísla X, Y

1. Pokud $n \leq 1$: return $X \cdot Y$

2. $k = \lfloor \frac{n}{2} \rfloor$

3. $A \leftarrow \lfloor \frac{X}{10^k} \rfloor, B \leftarrow X \bmod 10^k$

4. $C \leftarrow \lfloor \frac{Y}{10^k} \rfloor, D \leftarrow Y \bmod 10^k$

5. $P \leftarrow \text{NA'SOB}(A, C)$

6. $Q \leftarrow \text{NA'SOB}(B, D)$

7. $R \leftarrow \text{NA'SOB}(A+B, C+D)$

8. $Z \leftarrow P \cdot 10^{2k} + (R-P-Q) \cdot 10^k + Q$

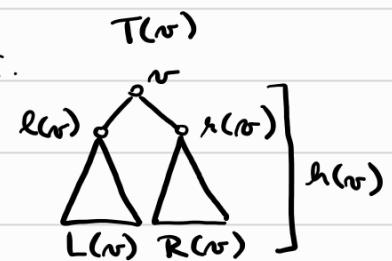
výstup: $Z = X \cdot Y$

$$T(n) = 3 \cdot T\left(\frac{n}{2}\right) + \Theta(n) \rightarrow \frac{a}{b^c} = \frac{3}{2^1} > 1 \rightarrow \Theta(n^{\log_2 3}) \approx \Theta(n^{1.59})$$

4) Binární vyhledávací stromy

def. Binární strom je zakořeněný strom t.ž.

• vrchol má nejvýš 2 syny



def. Binární vyhledávací strom je

binární strom t.ž. $r \in V$ obsahuje klíč $k(r) \in \mathcal{U}$

a platí: $\forall l \in L(r) : k(l) < k(r)$

$\forall r \in R(r) : k(r) < k(r)$

(klíče jsou unikatní)

operace:

Show / Enumerate :

$O(n)$

rekurzivně 1. $L(r)$

2. r

3. $R(r)$

Find(x):

$x < r \rightarrow L(r)$

$x > r \rightarrow R(r)$

$x = r \rightarrow \checkmark$

Insert(x):

1. neúspěšný Find

2. přidám x jako potomka listu, ve kterém skončím

Delete(x):

list \rightarrow smažu

vrchol s 1 synem \rightarrow kontrakce hrany

vrchol se 2 syny \rightarrow prohodím s listem \rightarrow smažu list
(min $R(x)$ nebo max $L(x)$)

$\hookrightarrow O(h(T(r))) \rightarrow O(n)$

def. Strom je dokonale vyvážený $\Leftrightarrow | |L(v)| - |R(v)| | \leq 1 \forall v$

def. AVL strom je hloubkově vyvážený binární vyhledávací strom, tedy $\forall v: | h(L(v)) - h(R(v)) | \leq 1$

↳ hloubka $\Theta(\log n)$

(v nevyváženém BVS $O(n)$)

operace s rotacemi \rightarrow průběžná kontrola vyvážení

5) Třídění

Bubble Sort

- postupně procházím pole
- pokud $P[i] > P[i+1]$ → procházím
 - „probublavaím“ největší prvek na konec
 - na konci i -tého prochodu polem je posledních i prvků správně umístěných

vstup: pole P

1. pokračuj \leftarrow true
2. while pokračuj :
 3. pokračuj \leftarrow false
 4. pro $i = 0, \dots, n-2$:
 5. if $P[i] > P[i+1]$
 6. pokračuj \leftarrow true
 7. $P[i], P[i+1] \leftarrow P[i+1], P[i]$

výstup: seřiděné pole P

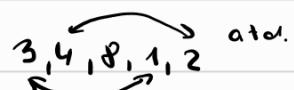
čas: $O(n^2)$

Selection Sort

vstup: pole P

1. Pro $i = 0, \dots, n-1$
2. $m \leftarrow i$
3. Pro $j = i+1, \dots, n-1$
4. Pokud $P[j] < P[m]$:
5. $m \leftarrow j$
6. $P[i], P[m] \leftarrow P[m], P[i]$

→ prohazuje s „lokálním“ minimum

 3, 4, 8, 1, 2 atol

Insert Sort

vstup: pole P

1. Pro $i = 0, \dots, n-1$:
2. $x \leftarrow P[i]$
3. $j \leftarrow i$
4. Dokud $j \geq 0 \wedge P[j-1] > x$:
5. $P[j] \leftarrow P[j-1]$
6. $j \leftarrow j-1$
7. $P[j] \leftarrow x$

výstup: seřiděné pole P

→ vkládám na správné místo dolera

3, 5, 4, 8
 ^

QuickSort:

• nahodný výber pivotovacího pruhu p

→

L	S	P
\nearrow $< p$	\nearrow $= p$	\searrow $> p$

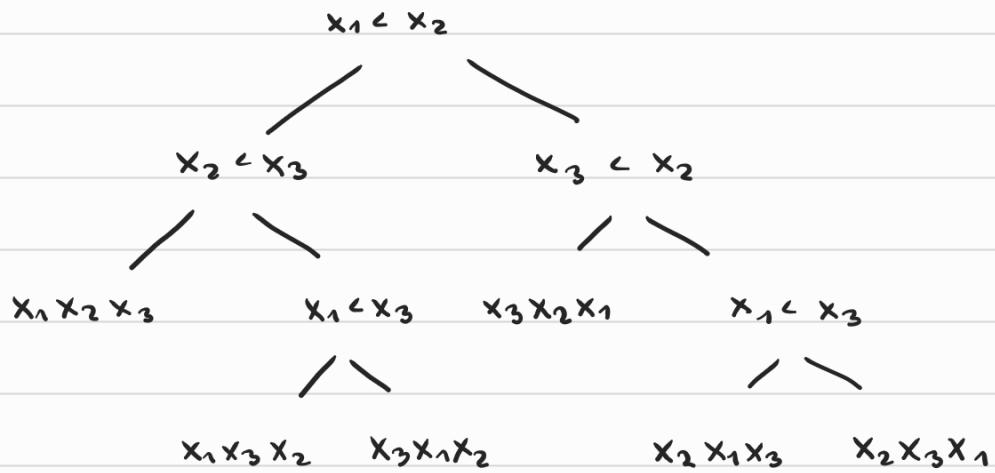
vstup: posloupnost $X = x_1, \dots, x_n$

1. Pokud $n \leq 1$: return $Y = X$
 2. $p \leftarrow$ některý z pruhů X
 3. $L \leftarrow$ pruhy $< p$
 4. $P \leftarrow$ pruhy $> p$
 5. $S \leftarrow$ pruhy $= p$
 6. $L \leftarrow \text{QuickSort}(L)$
 7. $P \leftarrow \text{QuickSort}(P)$
 8. $Y \leftarrow L, S, P$
- } rekurzivní seřidění

výstup: seřiděná posloupnost $Y = X$

čas $O(n \log n)$ s nahodnou volbou pivota

V: $\frac{1}{n}$ deterministický algoritmus v porovnávacím modelu, který trží n -prvkovou posloupnost, použije v nejhorším případě $\Omega(n \log n)$ porovnání
 \downarrow
hloubka porovnávacího stromu



deterministický = # krok je jednoznačně určen výsledky předchozích kroků

6) Grafové algoritmy

BFS = prohledávání do šířky

vstup: $G = (V, E)$, počáteční vrchol $v_0 \in V$

1. Pro \forall vrchol:

2. $stav(v) \leftarrow \text{nenalezený}$

3. $D(v) \leftarrow \emptyset$ // délka cesty z v_0 do v

4. $P(v) \leftarrow \emptyset$ // předchůdce v na min. cestě $v_0 \rightarrow v$

5. $stav(v_0) \leftarrow \text{otevřený}$

6. $D(v_0) \leftarrow 0$

7. $Q = \text{nova fronta}, Q.push(v_0)$

8. Dokud je Q neprázdná:

9. $v \leftarrow Q.pop()$

10. Pro \forall souseda w vrcholu v :

11. Pokud $stav(w) = \text{nenalezený}$

12. $stav(w) \leftarrow \text{otevřený}$

13. $D(w) \leftarrow D(v) + 1$

14. $P(w) \leftarrow v$

15. $Q.push(w)$

16. $stav(v) \leftarrow \text{uzavřený}$

- hledání nejkratších cest v grafu

čas $O(n+m)$

paměť $\Theta(n+m)$

DFS = prohledávání do hloubky

DFS:

vstup: $G = (V, E)$, počáteční vrchol $v_0 \in V$

1. Pro t vrchol v :

2. stav(v) \leftarrow nenašený

3. in(v) \leftarrow undefined // čas otevření

4. out(v) \leftarrow undefined // čas uzavření

5. $T \leftarrow 0$ // počítadlo kroků

6. DFS2(v_0)

DFS2(v):

1. stav(v) \leftarrow otevřený

2. $T \leftarrow T + 1$

3. in(v) $\leftarrow T$

4. Pro všechny sousedy w vrcholu v :

5. Pokud stav(w) = nenašený

6. DFS2(w)

7. stav(v) \leftarrow uzavřený

8. $T \leftarrow T + 1$

9. out(v) $\leftarrow T$

čas $O(n+m)$

paměť $\Theta(n+m)$

- kontrola existence cyklu
- hledání komponent souvislosti

def. DAG = acyklický orientovaný graf

def. topologické uspořádání je lineární uspořádání \leq na $V(G)$

t.z. $\forall xy \in E(G) : x \leq y$

↪ orientovaná hrana $x \rightarrow y$

tzn. \exists číselnou řadu $v_1, \dots, v_n : v_i, v_j \in V \Rightarrow i < j \Rightarrow v_i \leq v_j$

* cyklus nemá topologické uspořádání

V: Graf má topologické uspořádání \Leftrightarrow je DAG

def. zdroj je vrchol grafu t.z. $\deg^{\text{in}}(v) = 0$

↪ * DAG má zdroj

↳ stačí odstraňovat zdroje

V: pořadí, v němž DFS opouští vrcholy je opačné topologickému

topologické seřízení:

A) primitivní algoritmus: odstraňování zdrojů $\Theta(n^2)$

popř. stoků od konce

přidání hran $\Theta(n \cdot m)$

$\Rightarrow \Theta(n(m+n))$

B) pomocí DFS: viz. věta

$\Theta(m \cdot n)$

využití např. pro seřazení provedení nauzájen
závislých činností

def. ohodnocení hran jsou hodnoty $w(e)$, kde
 $w: E \rightarrow \mathbb{R}$ je váhuval funkce

ohodnocený graf = graf s váhuval funkcií

délka cesty $p = v_0 v_1 \dots v_n : w(p) = \sum_{i=1}^n w(v_{i-1}, v_i)$

vzdálenost $u, v : d(u, v) = \min \{w(p) : p \text{ cesta z } u \text{ do } v\}$
= ∞ pokud \nexists cesta z u do v

Dijkstrův algoritmus

- pro grafy s nezáporným ohodnocením hran
(kvůli trojúhelníkové nerovnosti)

vstup: G, v_0

1. Pro \forall vrcholy v :
 $h(v) \leftarrow \text{nenalezený}$
2. $h(v_0) \leftarrow \infty$ // ohodnocení vrcholu
3. $P(v) \leftarrow \text{undefined}$ // předchůdce v
4. $\text{stav}(v_0) \leftarrow \text{otevřený}$
5. $h(v_0) \leftarrow 0$
6. Dokud \exists otevřené vrcholy
7. vyberu otevřené v s nejménším $h(v)$
8. Pro všechny sousedy w vrcholu v :
Pokud $h(w) > h(v) + l(v, w)$
 $h(w) \leftarrow h(v) + l(v, w)$
9. $\text{stav}(w) \leftarrow \text{otevřený}$
10. $P(w) \leftarrow v$
11. $\text{stav}(v) \leftarrow \text{uzavřený}$
12. $\text{stav}(v) \leftarrow \text{uzavřený}$
13. $\text{stav}(v) \leftarrow \text{uzavřený}$
14. $\text{stav}(v) \leftarrow \text{uzavřený}$

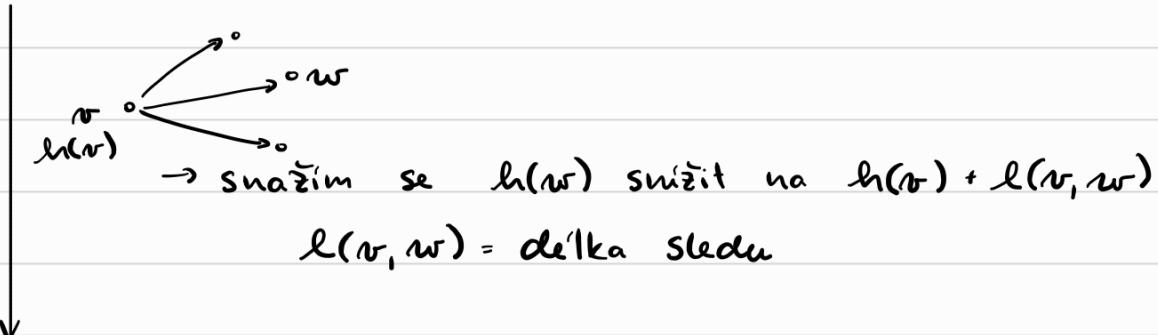
výstup: vzdálenosti h , předchůdci P

čas: $O(n^2)$, s binární haldu otevřených vrcholů $O((n+m) \log n)$

Bellman-Fordův algoritmus

- funguje i pro záporné hrany

relaxační algoritmus:



1. $\forall v: h(v) \leftarrow \infty$

$s(v) \leftarrow \text{neviděný}$

2. $h(u) \leftarrow 0$

$s(u) \leftarrow \text{otevřený}$

3. dokud \exists otevřený vrchol v :

relaxuj v

BF algoritmus je jiná příma aplikace relaxace

• otevřené vrcholy se drží ve frontě

\rightarrow zavírám nejstarší z otevřených vrcholů

čas: $O(n \cdot m)$ pro graf bez záporných cyklů

def. $G = (V, E)$ souvislý neorientovaný graf, $|V| = n$, $|E| = m$

váha grafu $w: E \rightarrow \mathbb{R}$

váha podgrafu = součet vah jeho hran

kostra grafu G je podgraf G obsahující všechny vrcholy, který je strom

kostra je minimální, pokud má mezi všemi kostrami nejmenší váhu

Borůvkov algoritmus

vstup: souvislý graf s unikátními vahami

1. $T \leftarrow (V, \emptyset)$ //trivialní les izolovaných vrcholů
2. Dokud T není souvislý:
3. $T \rightarrow$ komponenty souvislosti T_1, \dots, T_k
4. Pro $\forall T_i$:
5. $e_i \leftarrow$ nejlehčí hrana z T_i do zbytku G
6. Přidám do $\{e_1, \dots, e_k\}$ do T

výstup: minimální kostra T

čas $O(m \log n)$

↳ logaritmický # iterací

Jarníkův algoritmus

vstup: souvislý graf s vážovou funkcí w

1. Pro \forall vrchol v :
 2. stav(v) \leftarrow mimo
 3. $h(v) \leftarrow \infty$
 4. $p(v) \leftarrow$ nedefinováno $\quad /2.$ konec nejlehčí hrany
 5. $v_0 \leftarrow$ libovolný vrchol grafu
 6. $T \leftarrow$ strom pouze s vrcholem v_0 .
 7. stav(v_0) \leftarrow soused
 8. $h(v_0) \leftarrow 0$
 9. Dokud \exists sousední vrcholy:
 10. $u \leftarrow$ sousední vrchol s nejménším $h(u)$
 11. stav(u) \leftarrow uvnitř
 12. Pokud je $p(u)$ definováno:
 13. přidám hranu $\{u, p(u)\}$ do T
 14. Pro \forall hranu (u, v) :
 15. if $\text{stav}(v) \in \{\text{soused}, \text{mimo}\} \wedge h(v) > w(u, v)$:
 16. stav(v) \leftarrow soused
 17. $h(v) \leftarrow w(u, v)$
 18. $p(v) \leftarrow u$

výstup: minimální kostra T

ochodnocení v poli $\rightarrow \Theta(n^2)$

ochodnocení v haldě $\rightarrow \Theta(m \log n)$

def. **sít** = (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 hran**

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

Fordův - Fulkersonův algoritmus

vstup: **sít** (G, z, s, c)

1. $f \leftarrow 0$

2. Dokud $\exists P$ nenasycená cesta z, s ($\forall e: r(e) > 0$)

3. $\varepsilon \leftarrow \min_{e \in P} r(e)$

4. Pro $\forall u, v \in P$:

5. $\delta \leftarrow \min(\varepsilon, f(vu))$

6. $f(vu) \leftarrow f(vu) - \delta$

7. $f(uv) \leftarrow f(uv) + \varepsilon - \delta$

výstup: maximální tok f

V: Pro nejkratší nenasycenou cestu je # iterací $O(n \cdot m)$

\rightarrow F-F běží v čase $O(n \cdot m^2)$

- pro celočíselné c se algoritmus vždy zastaví
- pro racionalní c také \rightarrow lze převést na celočíselné
- Obecně se nezastaví vždy

def. hranový řez:

$$A, B \subseteq V : E(A, B) := \{ab \in E \mid a \in A \wedge b \in B\}$$
$$= E \cap (A \times B)$$

elementární řez

$$E(A, B) \text{ t.z. } A \cup B = V \wedge A \cap B = \emptyset \wedge \exists a \in A \wedge s \in B$$

\forall tok f , \forall řez $E(A, B) : |f| \leq c(A, B)$

\hookrightarrow kapacity řezu

$|f| = c(A, B) \rightarrow$ maximální tok, minimální řez