ALGORITMY A DATOVÉ STRUKTURY I

Luděk Kučera
kam.mff.cuni.cz/~ludek
algoviz.org

Slovníkové datové struktury
- množina, kt. se dá prohledávat
množina M, Č (prvky objekty - identifikační čísla)
- search - zda je č v množině
- insert - přidat č do množiny
- delete - vynechání čísla (zadané hodnoty, nejdříve search)
objektu, čísla, na kt. ukazuje pointer
(známe pouze čísla)

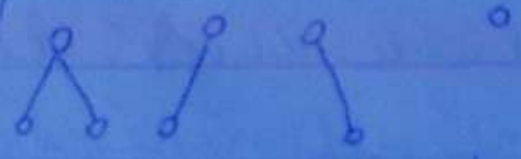
~

dictionary (slovník) operace search, insert, delete
- min, max
- predecessor, successor
- implementace pomocí spojových seznamů
- pomalé vyhledávání
- prvky uspořádané (pole delete) - rychlejší vyhl. pomocí půlení (implementace pomocí pole) $\log_2$
insert a delete pomalejší
↳ prvky za přidávaným číslem musíme posunout

Binární vyhledávací strom

kořen
listy

každý vrchol 2 nást.



levý syn
pravý syn

levý a pravý syn se rozlišuje
vrchol - objekt, pamatuje si levého a pravého syna
podstrom - všichni následníci

° pokud existuje cesta mezi vrcholy

podstromy nalevo nižší čísla než ve vrcholu
napravo vyšší
(n. stejné)

vytvořím si strom se stejným počtem vrcholů
K+1 ve vrcholu
K v levém podstromu
neklesající posloupnost (sloupce pod vrcholy podle velikosti)
- search - od kořene, čísla menší → do levého syna

. . . .
k

$1+2+4+\ldots+2^k = 2^{k+1}-1$ vrcholů

$\log_2 n$

$k = \lceil \log_2 n + 1 \rceil - 1$

- min Kořen → kdaná levého následníka (hledání - ∞ ...)
- max
- před největší číslo v podstromu
- přidávání - search, připojím jako syna
- delete - podle počtu následníků

žádný, 1 (podstrom napojím), 2 (v podstromu najdeme max) před následníkem

vyhledá prázdný vrchol s 1
odebere se jeho číslo, ne vrchol

Hloubka stromu (nejdelší cesta z kořene do listu)
na hloubce závisí složitost operací
náhodně vytvořené stromy (nejvyšší a průměrná hl.)

algoritmy a datové struktury I, II, III

přidávání náhodně

prům hloubka ~ $\log_2 n$

(použijí se distr.)

náhodně přidávaná čísla z intervalu, kt. se zvyšují

nevyvážené stromy

vrcho

vyvážené BVS

AVL strom — červeno-černé stromy

rozdíl výšek podstromů max 1

Adelson-Velskij, Landis

rotace – přepojení pointerů
- zachovává podmínky BVS
- pohyb jen nahoru a dolů
- nemění se klíč vrcholu
- výměna role otec syn

operace, kt. nemění tvar stromu stejné jako u BVS

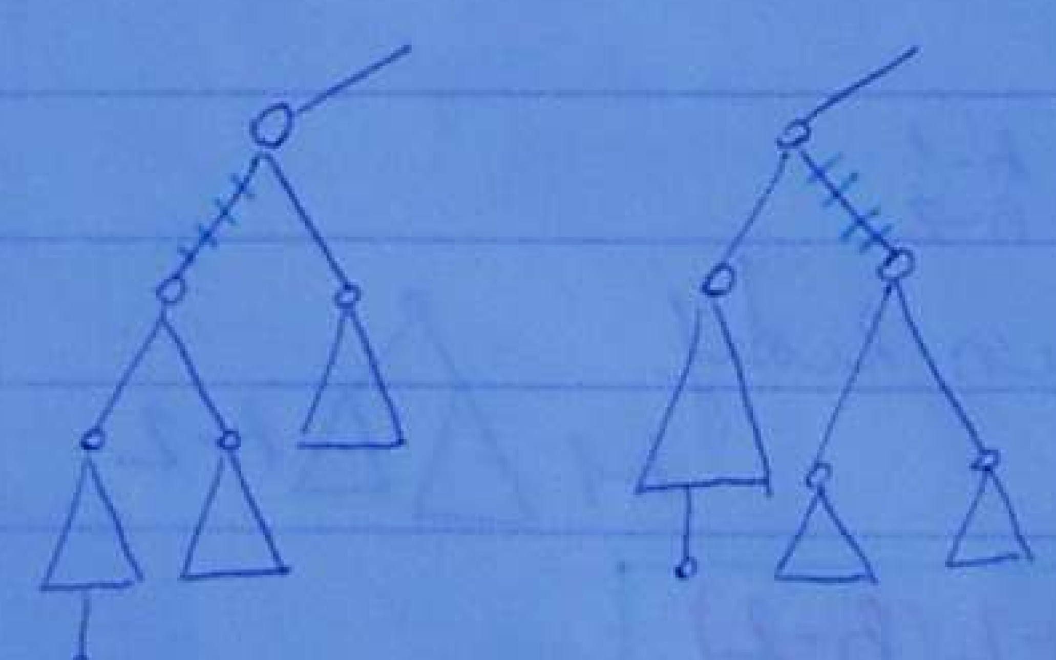
insert, delete – provede se BVS operace
- kontrola podmínek
- modifikace pomocí rotací

insert – v každém vrcholu info o vyváženosti −1,0,1

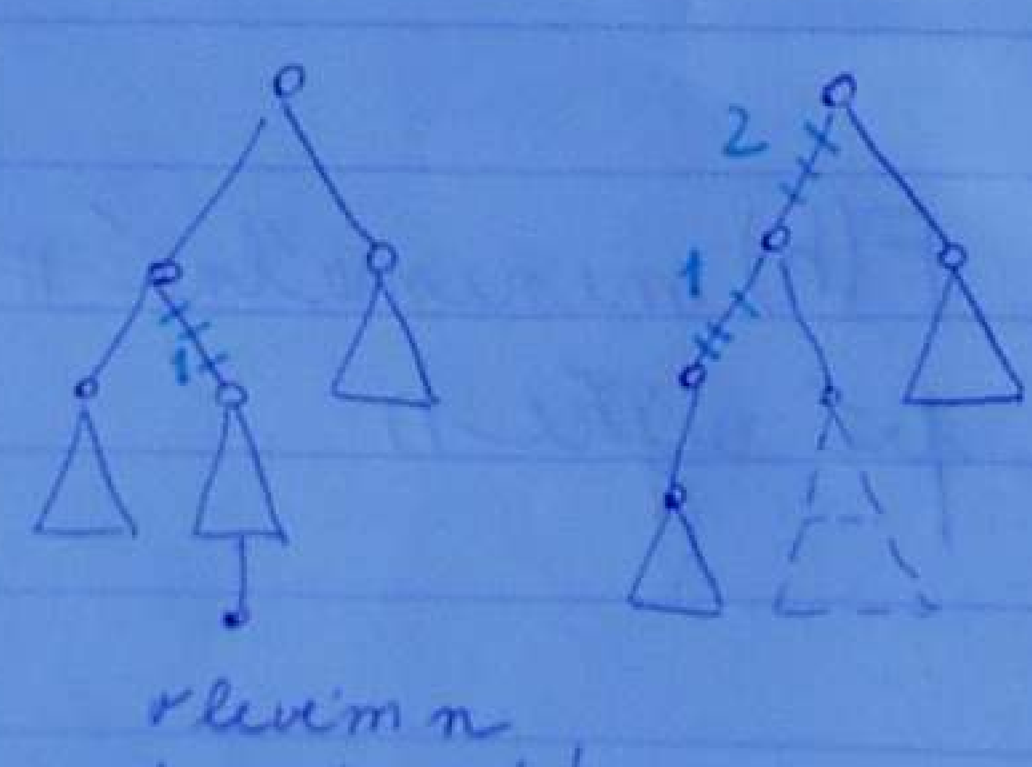
(rozdíl hloubek)

po přidání se tato hodnota nemění

průchod zpět z přid. vrcholu do kořene – hledá nevyv. vrcholy



vrcholy nad po rotaci do původ. stavu



v levém n pravém podstr.

delete - list - pokud 1 silně nevyvážený vrchol
- rodič se pokusí vyvážit vrcholy
- s 1 synem
- s 2 syny

Červeno-černý strom

- kořen černý
- žádný červený vrchol nemá červeného syna
- každá cesta z kořene do vrcholu, kde končí v listě nebo ve vrcholu s 1 synem, musí mít stejný počet černých vrcholů

spn

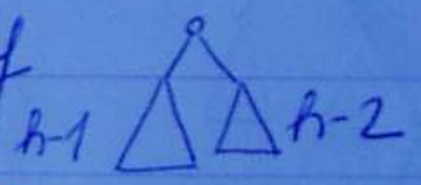
(kde schází doplníme černým pomocným vrcholem
cesta z kořene do listu stejný počet černých vrcholů)

BVS / $n-1$ hladina
průměrně $\frac{1}{2}$

AVL hloubka
$\geq F(h)$ listů
$F(0)=1$

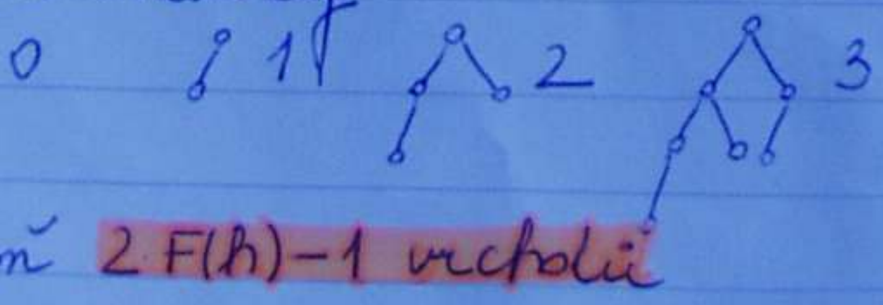
$h \geq 2$

strom s co nejmenším listy



min # listů = $L(h) = L(h-1) + L(h-2)$
$L(0)=1$
$L(1)=1$

Min strom hladiny



alespoň $2 \cdot F(h)-1$ vrcholů

$F(h)$ minimální # listů. pro výšce h vrcholů

Algoritmy III

N, H

$$N \geq F(H) \geq c^{H-1}$$

$c = \frac{1+\sqrt{5}}{2}$ zlatý řez

$$c^2 = c + 1$$

Důkaz MI

1) $H = 0$ platí

$H = 1$ =

2) $F(H) = F(H-1) + F(H-2) \geq c^{H-2} + c^{H-3} =$

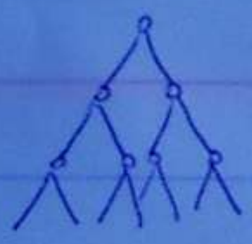
$= c^{H-3}(c+1) = c^{H-3} \cdot c^2 = c^{H-1}$

$$\log_2 N \geq H - 1 \cdot \log_2 c$$

$$1 + \frac{1}{\log_2 c} \log_2 N \geq H$$

$\| 1,44..$

$$H \leq 1 + 1,44 \log_2 N$$



0 1
1 2
2 4
⋮ ⋮
H 2^H

$$N \leq 2^{H+1} - 1$$

$$H \geq \log_2(N+1) - 1$$

max. hl. o 44% než prům.

Red-black tree

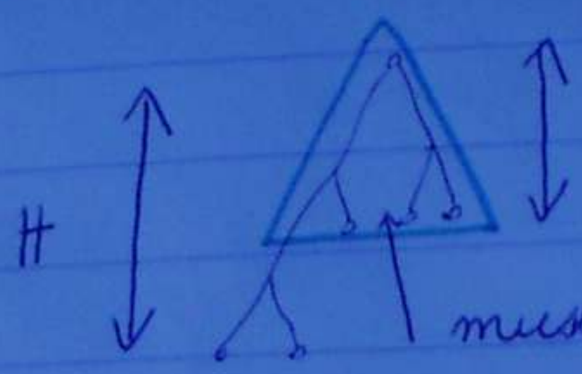
$H \leq 2 \cdot \log_2 N$

$$\frac{\text{nejd. cesta z kořene do listu}}{\text{nejkr. cesta z kořene do vrcholu, kt. schází syn}} \leq 2$$

H hran na nejd. cestě → H+1 vrcholů

alespoň $\frac{H+1}{2}$ černých vrcholů

cesta do vrcholu $s < 2$ synů má alespoň $\frac{H-1}{2}$ hran



musí být úplný binární strom
jinak by byla kratší cesta

určuje $\frac{H-1}{2}$ a alespoň $2^{\frac{H-1}{2}+1}-1$ vrcholů

$N \geq 2^{\frac{H-1}{2}+1}$ $\left(-1\right)$

$\frac{H-1}{2} \leq \log_2 N - 1$

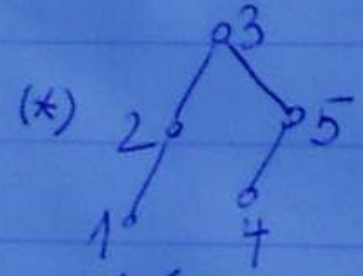
$H \leq 2(\log_2 N - 1) + 1$

$\boxed{H \leq 2 \log_2 N}$

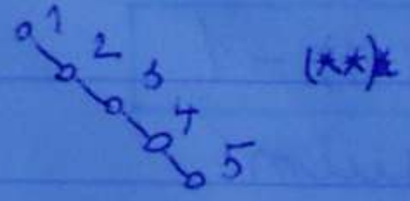
BVS

1, ... n → změna pořadí (permutace)

3, 2, 5, 4, 1



1, ... 5



$n!$ možných způsobů

π - permutace

T_π - strom * přidáváním prvků v pořadí podle perm.

$H(T)$ - hloubka

$\frac{1}{n!} \sum_\pi H(T_\pi)$ průměrná hl.

algoritmy III, IV

$h(T)$ – průměrná hloubka vrcholu

(*) $\frac{0+1+1+2+2}{5} = \frac{6}{5}$

$H(T) = 2$

$h(T) = \frac{6}{5}$

(**)

$H(T) = n-1$

$h(T) = \frac{0+1+\ldots+(n-1)}{n} =$

$= \frac{n-1}{2}$

$h(T,v)$ – délka cesty z kořene do v

$h(T) = \frac{1}{n} \sum_v h(T,v)$

prům. max hl. = $\frac{1}{n} \sum_v$ max v $h(T,v)$

$\sum$ max

$\sum \sum$

$\Rightarrow$ střed prům. max. hl. $\geq$ prům. prům. hl.

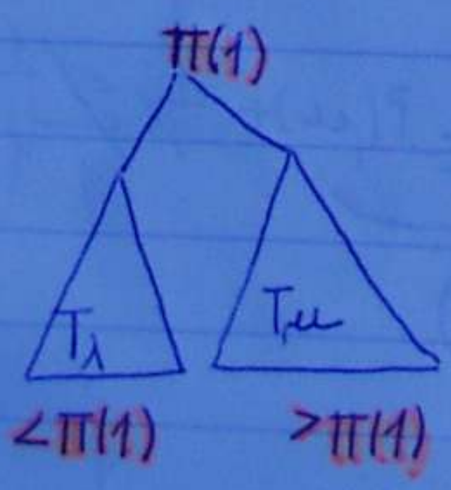
$$\frac{\frac{1}{n!} \sum_\pi \frac{1}{n} \sum_{v \in T_\pi} h(T_\pi, v)}{\frac{1}{n} \frac{1}{n!} \sum_\pi \sum_{v \in T_{\#\pi}} h(T_\pi, v)}$$

$$\frac{1}{n} \sum_\pi \sum_{v \in T_\pi} h(T_\pi, v) = Q(n)$$

$Q(n) \leq 4n \log_2 n$

$\frac{Q(n)}{n} \leq 4 \log_2 n$

Důkaz



výběr permutace

1) zvolím π(1) n způsobů

2) zvolím permutaci čísel 1 ... π(1) − 1

(π(1) − 1)! způsobů

3) zvolím π(1)+1 ... n

(n − π(1) − 1)!

4. kvalitní množ. $A \subset \{2, \dots n\}$ *

$|A| = \pi(1) - 1$

1..7

$\pi(1) = 3$

1 2 3 4 5 6 7

3 5 2 7 1 4 6

1,2 2,1

4,5,6,7 5,7,4,6

$\pi(1) = j$

$n(j-1)!\,(n-j)!\binom{n-1}{j-1} = n(j-1)!\,(n-j)!\,\frac{(n-1)!}{(n-j)!\,(j-1)!} = n \cdot (n-1)! = n!$

$$P(\pi) = \sum_{v \in T_\pi} h(T_\pi, v)$$

$$Q(n) = \frac{1}{n!} \sum_{\pi} P(\pi) = \frac{1}{n!} \sum_{j=1}^{n} \sum_{\lambda} \sum_{\mu} \sum_{A} (j-1) + P_\lambda + (n-j) + P_\mu$$

$$P(\pi) = \frac{1}{n!} \sum_{j=1} \binom{n-1}{j-1} \sum_{\lambda \in S_{j-1}} \sum_{\mu \in S_{n-j}} \big(P(\lambda) + (j-1) + P(\mu) + (n-j) \big)$$

součet

$P(\pi)$

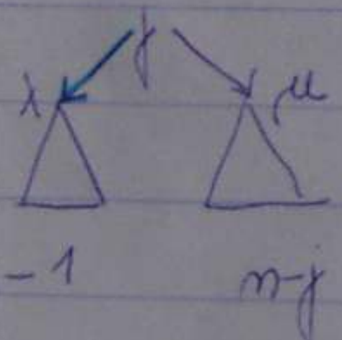
$\pi(1) = j$ volba j

kladek vpředu

v podstromu λ + $j-1$ jedniček

$j \dots\dots$

$\underbrace{\qquad}_{n-1}$



$j-1$ čísel menších než j

na sebe navazují pozice čísla větší

$$\frac{1}{n} \sum_{j=1} \frac{1}{(j-1)!} \frac{1}{(n-j)!} \sum_{\lambda} \sum_{\mu} \big(P(\lambda) + P(\mu) + n - 1 \big)$$

$$\frac{1}{n} \sum_{j} \frac{1}{(j-1)!} \frac{1}{(n-j)!} \sum_{\lambda} \sum_{\mu} P(\lambda) + \frac{1}{n} \sum_{j} \frac{1}{(j-1)!} \frac{1}{(n-j)!} \sum_{\lambda} \sum_{\mu} P(\mu) + \frac{1}{n} \sum_{j} \frac{1}{(j-1)!} \frac{1}{(n-j)!} \sum_{\lambda}$$

$Q(j-1)$ $Q(n-j)$

algoritmy II

$$Q(n) = \frac{1}{n}\sum_{j=1}^{n} Q(j-1) + \frac{1}{n}\sum_{j=1}^{n} Q(n-j) + (n-1)$$

$$Q(0) + Q(1) + \dots + Q(n-1)$$

$$Q(n) = \frac{2}{n}\sum_{j=1}^{n} Q(j-1) + (n-1)$$

$$Q(n) = \frac{2}{n}\sum_{k=0}^{n-1} Q(k) + (n-1)$$

$$\sum_{k=1}^{n-1} k \cdot \log_2 k = \sum_{k=1}^{\lceil n/2 \rceil - 1} k \log k + \sum_{k=\lceil n/2 \rceil}^{n-1} k \log k \leq \sum_{k=1}^{\lceil n/2 \rceil - 1} k(\log n - 1) + \sum k \log n =$$

$$= \left(\sum_{k=1}^{n-1} k\right) \cdot \log n - \sum_{k=1}^{\lceil n/2 \rceil - 1} k = \frac{n(n-1)}{2} \log n - \frac{\lceil n/2 \rceil (\lceil n/2 \rceil - 1)}{2} \leq$$

$$\leq \frac{1}{2} n^2 \log n - \frac{n^2}{8} + \left(\frac{n+1}{4} - \frac{n}{2} \log n\right) \leq \frac{1}{2} n^2 \log n - \frac{n^2}{8}$$

$\log \frac{n}{2}$

lichokávorka ≤ 0

MI:

$$Q(n) \leq \frac{2}{n}\sum_{k=1}^{n-1} 4k \log k + n - 1 = \frac{8}{n}\sum_{k=1}^{n-1} k \cdot \log k + n - 1 \leq \frac{8}{n}\frac{n^2}{2}\log n - \frac{n^2}{2 \cdot 8} \cdot \frac{8}{n} +$$

$Q(0) = 0$

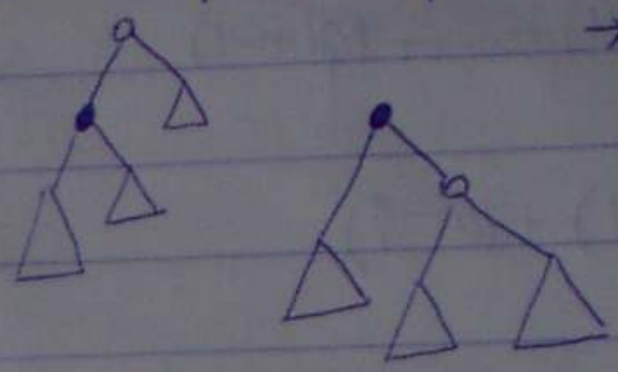
$$+ n - 1 = 4n \log n - n + n - 1 = 4n \log n - 1$$

$Q(n)$ průměr přes permutace prvním * hloubek

průměrná hloubka vrcholu $4 \log n$

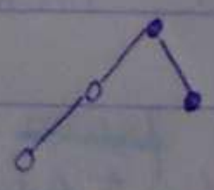
Červeno-černé stromy

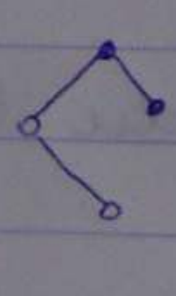
- do $\frac{h}{2}$ plně zaplněné
- zakázané vložení hrany, kde spodní vrchol je černý
  → porušení 3. podmínky



- pokles vložení / přebarvování vrcholů přebarvíme

- insert - search
  - barva → červený
    (kromě přidávání do prázdného stromu)
    přidání za vrchol - černý ✓
    (otec)
    - červený - strýc červený
      dědeček černý
      barva z dědečka na otce a strýce,
      opravujeme dále
      (přebarvení kořene na černo)
    - strýc černý
      vložení hrany od dědečka k dítěti
      (pouze 1x)

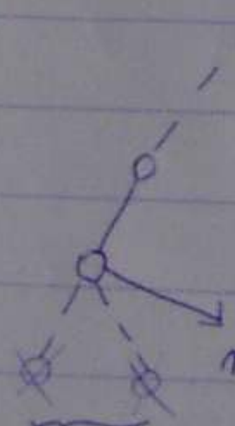




vložení červené dvojice
převod na předchozí případ

- delete

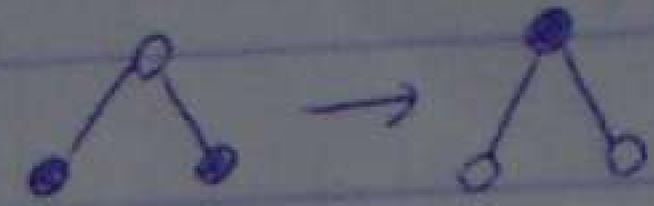
vrchol s 1 následníkem → musí to být červený
ustupujeme



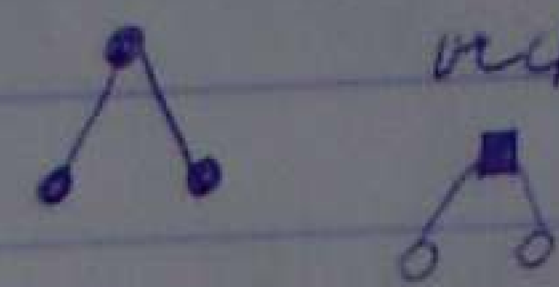
nemůže být černý
nemůže být černé ani červené

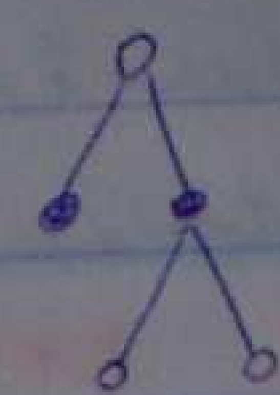
Algoritmy II, I

– černý list



vrchol dvojnásobně černý – 2x na cestě





přesun klíče

B-stromy

– databáze

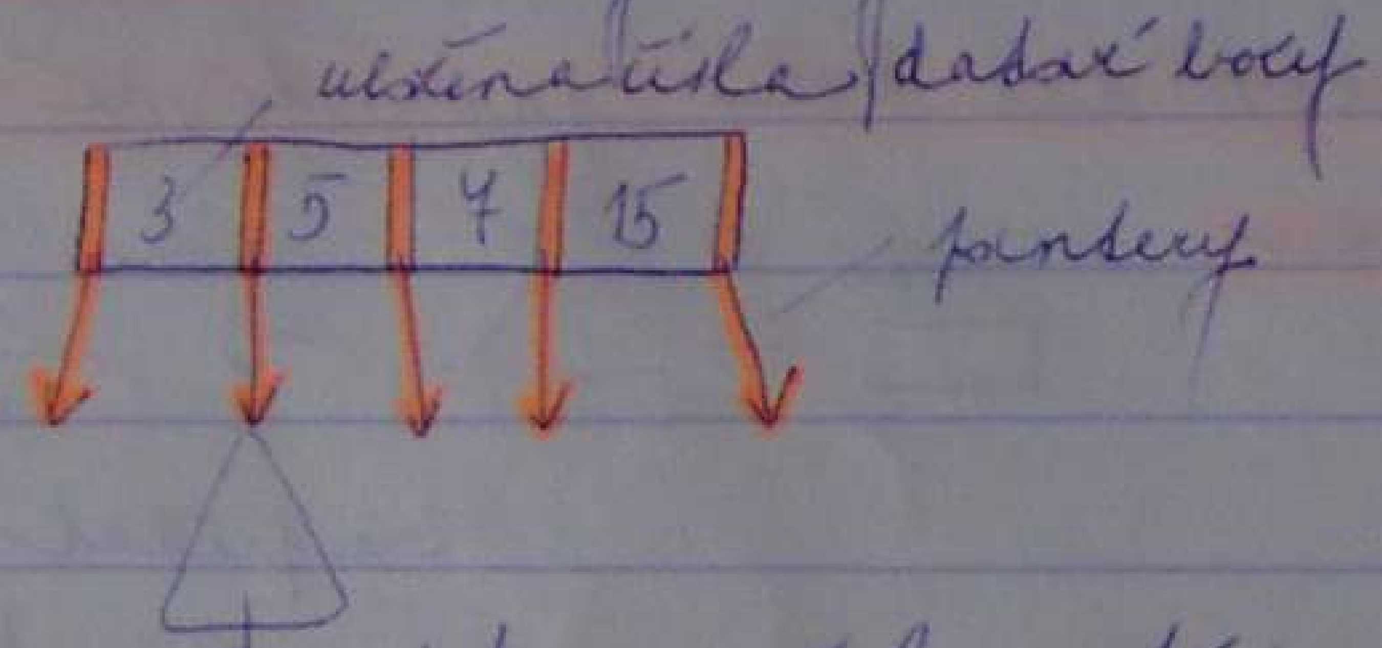
– minimalizovat počet přístupů do vněj. paměti

(→ optimální počet větvení e (ternární lepší než binární)

– pokud se stromy nevejdou do vnitřní paměti → široké uzly

větší počet větvení

menší hloubka



v podstromu čísla větší než 3 a menší než 5

– search, min, max

– tvar – ne všechny vrcholy mají stejný počet synů

datových bloků v uzlu **alespoň k**

nejvýše 2k+1 pro nějaké k > 1

kořen datových bloků **alespoň 1** ⇒ kořen stupně alespoň 2

nejvýše 2k+1

– všechny listy ve stejné hladině

1

2 $\quad 2k^{+1}+1$

2k $\quad (2k^{+1}+1)^2$

l vrstva $2k_{+1}^{l-1}$ $(2k_{+1}+1)^{l}$

hladka h alespoň $2(k+1)^{h-1}$ listů,

takže má i alespoň tolik vrcholů

$$2(k+1)^{h-1} \le n$$

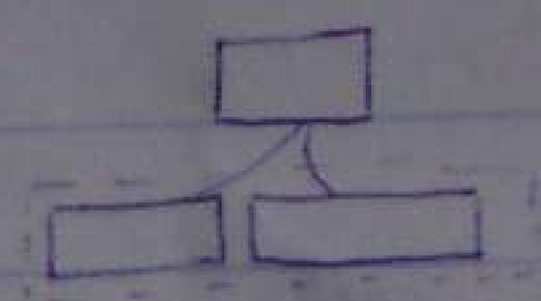
$$h \le 1 + \log_{k+1} \frac{n}{2}$$

$$h \le \log_{k+1} n$$

$\frac{\log_2 n}{\log_2 k}$ změna hloubky oproti $\log_2$

- insert - search
  - najdu datový blok by měl být v uzlu, kt. už má max. # dat. bloků → uzel se rozdělí na 2

K K+1



v uzlu o vrstvu výše musíme přidat datový blok, aby z další pointer → řetězová reakce, až rozštěpení vrcholu

řešení ⇒ po cestě dolů, každý najdeme vrchol s max. počtem dat. bloků, preventivně ho rozštěpíme

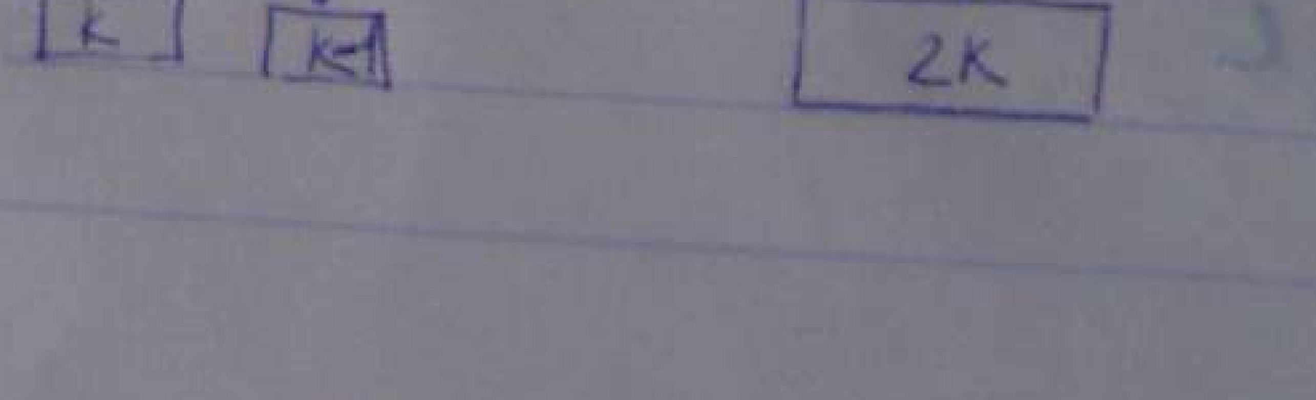
lze přidat do vyšší vrstvy

$2k+1$ → k, k, do vyšší vrstvy

- delete - listu (pointery nikam neukazují)
  - jiný (přesunout hodnotu nejbližšího vyššího ... listu

může z uzel s k-1 dat. bloky
jeho nejbližší bratr s dalšími aspoň k+1 dat. bloky
(adopce od staršího bratra)
přes otce
od mladšího

pokud nelze provést adopci - opačná operace k štěpení



alg. VI

Hašování

- insert
- delete
- search
- operace se provádějí rychle (v prům. případě)
- univerzální ⊥ pravděpodobnostní alg.
  - generátor náhodných čísel

- množina U = universum

předpoklad $U = N$

$M \subset U$

↑ tuto množinu chceme uchovávat, rozumně velká

I mož. rozumně velká (řádově jako M) $I = \{0, \dots k-1\}$

$h: U \to I$ např. $h(x) = x \bmod k$

B_0, B_1, B_2, … B_{k-2}, B_{k-1} } indexované množ. I

$x \in M$

uložím do $B_{h(x)}$

bucket

- search $x \in M$?

  podíváme se do $B_{h(x)}$

  jestli $x \in M$, musí x de být

- insert do $B_{h(x)}$
- delete z $B_{h(x)}$

- s alg. byly prvky v bucketech rozděleny rovnoměrně

$M = I$ v ⌀ bude 1 prvek ⇒ ⌀ operace $O(1)$ případ hašovací fce)

- ~~prvky~~ α

$\frac{M}{I} = c$ $c \in (\frac{1}{2}, 2)$ čísla generovaný náhodně

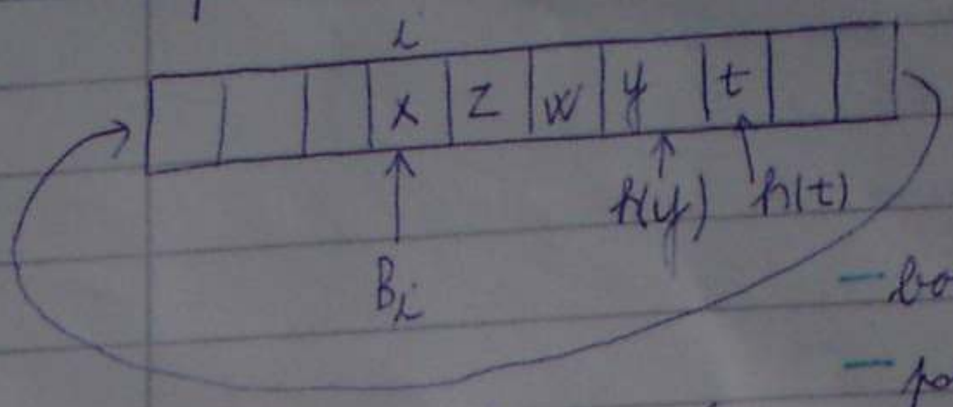
průměrný počet prvků v bucketu

bude snaha # prvků $\alpha \cdot \log|M|$

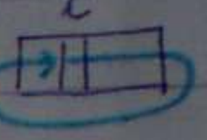
search v ⌀ $O(c)$

nejh. případ $O(\log M)$

- pokud $\frac{M}{l}$ velké ⇒ hodně prvků v bocích, implementují se špatně



- boky se překrývají
- pole musí být velké |M|
- B_i bok pokračuje přes další



odvětví adresování

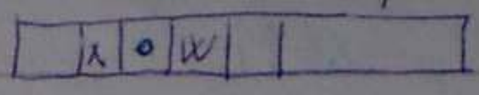
hašovací adresa

insert
- $h(x)=i=h(z)=h(w)$
- prvek se přidá do nejbližšího pokračování pole

search
- v $h(w)=i$, pokračuje se dokud na něj nenarazíme
- nebo dokud pole není prázdné

~~mohli bychom zkusit~~ (prvek se maže)

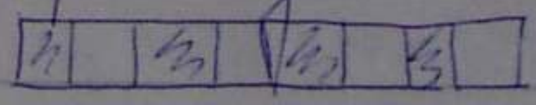
problémy s delete
- vynechání z – pokud bychom jen vynechali, tak operace by fungovaly špatně, vložíme značku, že máme pokračovat



- někdy lze značku odstranit – insert
- pokud jen hodně delete – samé značky

pole by mělo být naplněno zpola nový

v opt. případě



průměrná délka naplněných částí konstantní

nej. případ log

dvojité hašování

$h(x)$
$k(x)$

prvek zkusíme dát na $h(x)$, pokud naplněné $h(x)+k(x)$, ..
v původním případě $k(x)=1$
$h(x)$ a $k(x)$ soudělné → málo možností → volba délky pole prvočíselná

volba h podobně jako pseudonáhodná čísla
$h(seed)=s_1$ $h(s_1)=s_2$

H – množina možných hašovacích funkcí = třída hašovacích funkcí
náhodně si vybere 1x hašovací funkci a tu bude používat
v prvním případě bude náhodně vybraná hašovací funkce kolidovat
každou M dobrá (vysoká pravděpodobnost)
→ univerzální hašování

$H, M, h \in H$ (rand)

$(K = m)$

m – velikost hašovací tabulky

$U = \{0, \dots K-1\}$

$K \leq p$ p prvočíslo

$\lambda_{a,b} : \{0, \dots p-1\} \to \{0, \dots p-1\}$ $\quad 0 < a < p,\ 0 \leq b < p$

$\lambda_{a,b}(x) = (ax + b) \bmod p$

$U \subset_{\text{vše}} \{0, \dots p-1\} \xrightarrow{\lambda_{a,b}} \{0, \dots p-1\} \xrightarrow{x \to x \bmod m} \{0, \dots m-1\}$

$x \longrightarrow x \longrightarrow \lambda_{a,b}(x) \longrightarrow \lambda_{a,b}(x) \bmod m : h_{a,b}(x)$

$h_{a,b}(x) = ((ax + b) \bmod p) \bmod m$

Pokud a, b zvolíme náhodně s velkou pravděpodobností
bude při jakékoliv M hašovací tabulka dobrá.

$x, y \in \{0, \dots p-1\}$

$x \neq y$

$(\lambda_{a,b}(x), \lambda_{a,b}(y))$

uspořádaných dvojic $p(p-1)$
nezáporných
z množiny $\{0, \dots p-1\}$
$0 < a \leq p$ vybere a $p-1$ } # fcí $p(p-1)$
$0 \leq b \leq p$ vybere b p }

$u, v, u \neq v$ $\quad \exists!\ a, b$ $\quad \lambda_{a,b}(x) = u,\ \lambda_{a,b}(y) = v$

$(a', b') \neq (a'', b'')$ $\quad (\lambda_{a',b'}(x), \lambda_{a',b'}(y)) \neq (\lambda_{a'',b''}(x), \lambda_{a'',b''}(y))$

$x, y,\ x \neq y$

v alespoň jedné chybí
$a' \neq a'',\ b' \neq b''$

dikaz sporem

$\lambda_{a',b'}(x) = \lambda_{a'',b''}(x)$ $\quad \lambda_{a',b'}(y) = \lambda_{a'',b''}(y)$

$a'x + b' \equiv a''x + b'' \bmod p$

*) $(a'-a'')x \equiv b''-b' \bmod p$ $\quad (a'-a'')y \equiv b''-b' \bmod p$

$(a'-a'')x \equiv (a'-a'')y \bmod p$

$(a'-a'')(x-y) \equiv 0 \bmod p$

$x \neq y$

$x, y \in \{0, \dots p-1\}$

$x - y \neq 0 \bmod p$

p je prvočíslo $\Rightarrow$ jedno z čísel musí být nulové

$a' = a'' \bmod p$

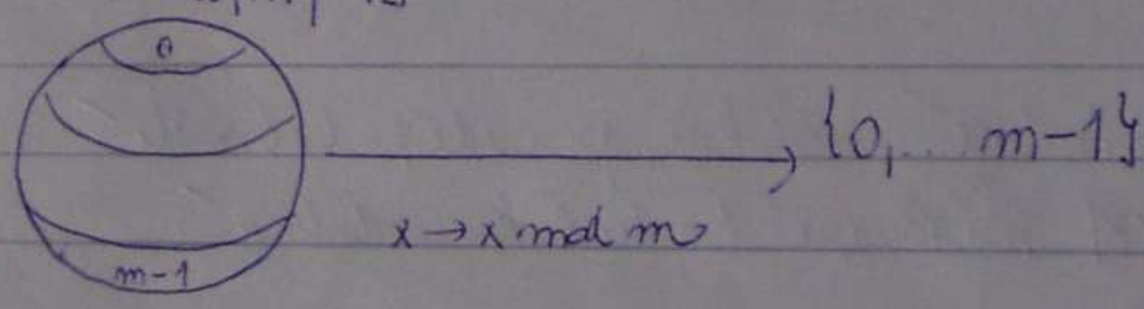
$a' = a''$

*) $0 \equiv b''-b'$

$b' = b''$

funkce vždy přiřadí na něco jiného $\Rightarrow$ všechny dvojice spolu

$P = \{0, \dots p-1\}$



konflikt - 2 různé prvky se dostanou hodnoty které fce $\rightarrow$ do stejného boxu
$\Rightarrow$ zajistit málo konfliktů

chceme vybrat dvojici, aby udělala konflikt, náhodné a, b $\rightarrow$ stejné pravděpodobné

M < P

a, b

$c(M, a, b)$ - # konfliktů kolizních množiny M při $h_{a,b}$

= # dvojic (x, y) $x, y \in M$, $x \neq y$, a zároveň $h_{a,b}(x) = h_{a,b}(y)$

alg. VII, VIII

$|M| = m$

$c(M,a,b) = 0$ Každý prvek ve svém boxu

$c(M,a,b) = m(m-1)$ #dvojic v boxu

$$\frac{1}{p(p-1)} \sum_{a=1}^{p-1} \sum_{b=1}^{p-1} c(M,a,b)$$

průměr přes všechny hašovací fce

#konfliktů pro pevné vložené hašovací fce

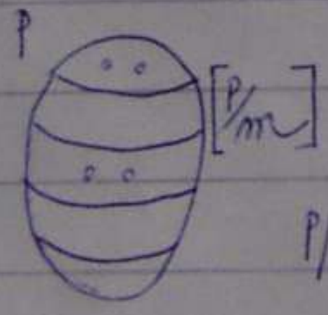
$$\frac{1}{p(p-1)} \sum_{a} \sum_{b} \sum_{\substack{x \in M \\ x \neq y}} \sum_{y \in M} \gamma(x,y,a,b)$$

$$\gamma(x,y,a,b) = \begin{cases} 1 & h_{ab}(x) = h_{ab}(y) \\ 0 & \text{jinak} \end{cases} \qquad |M| = n$$

$$\frac{1}{p(p-1)} \sum_{x} \sum_{y} \left(\sum_{a} \sum_{b} \gamma(x,y,a,b) \right) = \frac{n(n-1)}{p(p-1)} \sum_{a} \sum_{b} \gamma(x,y,a,b) =$$

$\lambda_{ab}(x) \equiv \lambda_{ab}(y) \bmod m$ ... vždy stejné číslo

$$= \frac{n^2}{m}$$



(průměrný #konfl $\doteq m \cdot p/m \cdot p/m$) $\frac{n^2}{p^2} = \frac{n^2}{m}$

$\frac{n(n-1)}{m}$ průměrný #konfliktů

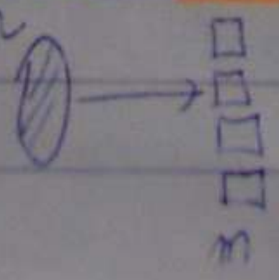
(#konfliktů, kdyžjsem #prvků zobrazili do 1 boxu / #boxů)

$\Rightarrow$ platí pro jakoukoliv množinu konkrétní množina, všechny haš. fce

Perfektní hašování

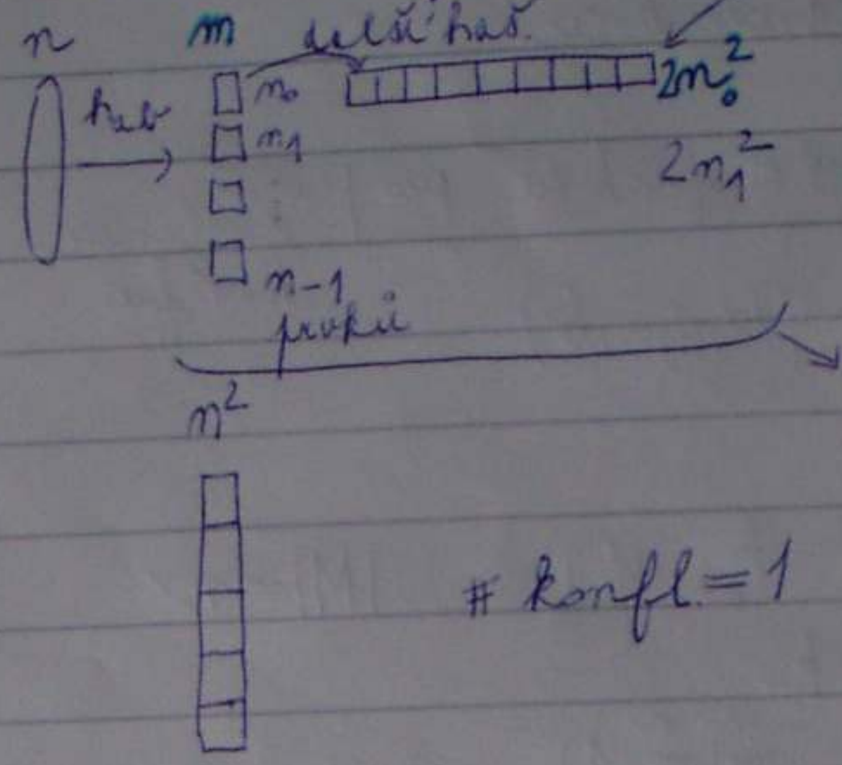
pevná M

vyrobí hašovací schéma, aby nebyly žádné konflikty



$\frac{n^2}{m}$ konfliktů

$n \sim m \Rightarrow n$ konfliktů



$2n^2$ — $1/2$ → # 2. fce bez konfliktů

náhodně vybíráme HT → pravděpodobné, že vyberem bezkonfl. fci

Třídění

(Mergesort, Heapsort, Quicksort, Bubblesort)

- zaměřeno na posunání 2 čísel

posunání v nejhorším případě $P(\pi)$

nejlepším

průměrném $\frac{1}{n!} \sum_{\pi} P(\pi)$

1-n čísel

n! permutací

Mergesort $\underbrace{a_1 \dots}_{\text{setřídíme}} | \underbrace{\dots a_n}_{\text{setřídíme}}$ (n mocnina 2)

nejh. $P_m(n) = \underbrace{2 \cdot P_m\left(\frac{n}{2}\right)}_{\text{setřídění}} + \underbrace{n-1}_{\text{posunání}}$

$P_m(n) \leq 2 \cdot P_m\left(\frac{n}{2}\right) + n = n + 2 \cdot \frac{n}{2} + 2 \cdot 2 \cdot P\left(\frac{n}{4}\right)$

$= \underbrace{n + n + \dots + n}_{k\text{-krát}} + 2^k P\left(\frac{n}{2^k}\right)$ dokud $\frac{n}{2^k} = 1$

$= n \cdot \log_2 n$ prok = log n

Alg VIII

$n \le 2^l \le 2n$ (pokud n není mocnina 2)

$P(n) \le P(2^l) = 2^l \cdot l \le 2n \cdot \log_2 2n$

Heapsort – find min
– vkládání
– vyjmutí min $2^n \log n$ porovnání n vyjmutí ?

$3n \log_2 n$ nejh. $n \log n$

Quicksort $a_1 \dots a_n$ p – pivot

záleží na volbě pivota

setřídíme $\begin{cases} a_i < p \\ a_i = p \\ a_i > p \end{cases}$

nejl. množiny stejně velké

$2n + 2 \cdot P\left(\frac{n}{2}\right)$

$p = a_{\lceil \frac{n}{2} \rceil}$

nejh. případ jedna z množin prázdná (p = min v max)

$P_q(n) = 2n + P_q(n-1) = 2n + 2(n-1) + 2(n-2) + \dots + 2 \cdot 1 + 2 \cdot 0 = \frac{2n(n+1)}{2}$

$= n^2 + n$

(#porovnání s pivotem)

Bubblesort

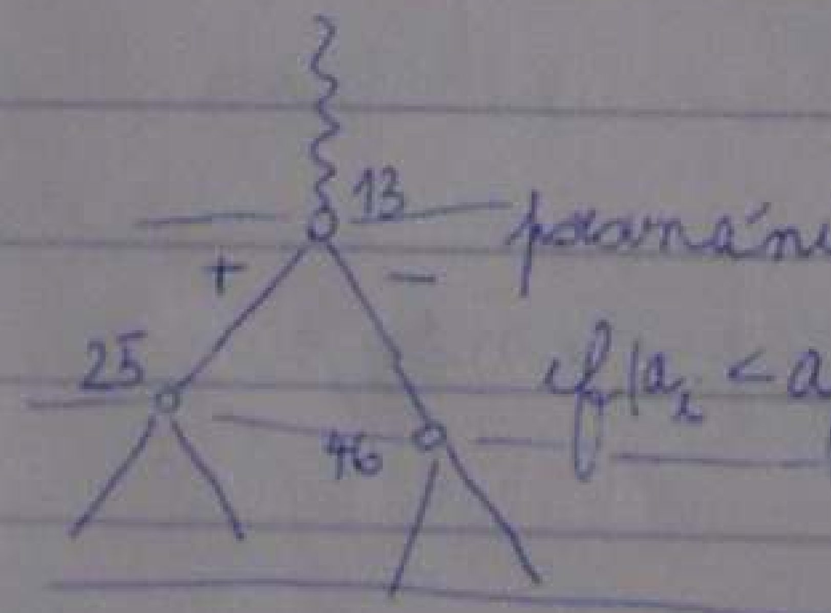
nejh. setříděné opačně

$\frac{n(n-1)}{2}$

nejl. případ – (skoro) setříděná posloupnost

pouze porovnání $O(n)$

prům. případ polovina nejhoršího



porovnání

if($a_i < a_j$)

možné průběhy výpočtu → strom

#listů = n! (permutace čísel

1 list odpovídá 1 vstupu)

rozdělíme do hladin, ve kt. se provedlo porovnání

k porovnání

⇓

2^k listů

na nejdelší vstup k porovnání $\Rightarrow$ $\leq 2^k$ listů

$n! \leq 2^k$ alg. založený na porovnání

$\exists$ nějaký vstup s k porovnání

$\log_2 n! \leq k$

Stirling $n! \doteq \frac{n^n}{e^n}\sqrt{2\pi n}$

$n\log n - n\log_2 e +$

$n! \geq \frac{n}{2} \cdots \frac{n}{2} = \left(\frac{n}{2}\right)^{\frac{n}{2}}$

$\frac{n}{2}$ krát

nemůže být alg., který by třídil v nejhorším případě lépe než $n\log n$

$1, \dots, n$

π

+ − (zda porovnání úspěšné) if $(a_i < a_j)$

posloupnost + − jak dopadlo porovnání

$A(\pi)$ ++−−+−++−

délka posloupnosti = # porovnání

$|A(\pi)|$ $\quad \frac{1}{n!}\sum_{\pi}|A(\pi)| \geq \alpha \cdot n \cdot \log_2 n$

průměrná výpočetní složitost

$\{A(\pi)\}_\pi$ tento systém je rozlišující, pokud $\pi_1 \neq \pi_2$ $\quad A(\pi_1)$ ++− …

$A(\pi_2)$ +−+ …

se od některého místa liší

nemůže být některá posloupnost částí druhé

když $\{A(\pi)\}_\pi$ je rozlišující, potom platí $\frac{1}{n!}\sum_{\pi}|A(\pi)| \geq \alpha n \log_2 n$

Důkaz: MI dle n

1) $n=1$ platí

2) platí pro $1 \dots n-1$

a) $\forall \pi$ A_π začíná +

b) $\forall \pi$ A_π začíná −

c) zbývající

alg. VIII

Systém posl + a - $\{A_i\}_{i\in I}$ je rozlišující $i \neq j$ $A_i = a_{i1} \dots a_{ik_i}$
$A_j = a_{j1} \dots a_{jk_j}$

$\exists l \leq \min k_i, k_j \quad a_{il} \neq a_{jl}$

*) $\frac{1}{|I|} \sum_{i\in I} |A(i)| \geq$ * $\log_2 |I|$

pokud a) odebereme +, dostaneme kratší posl a převedeme na c)
↓ menší suma

/ posl a zavírající + I_1
\
−||− − I_2

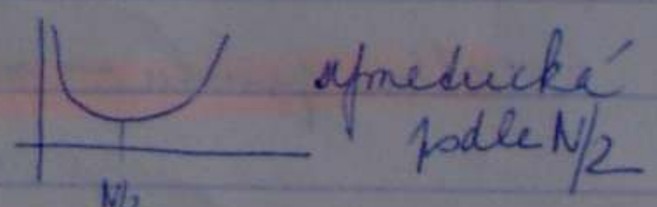
I_1 posl z I_1 po odebrání úvodního + $\quad i \in I$ $B(i)$ je $A(i)$ s odbržením 1. prvkem
I_2 z I_2 −

~ prodyto posl lze použít *)

$$\sum_I |A(i)| = \sum_{i\in I_1} |A(i)| + \sum_{i\in I_2} |A(i)| = |I_1| + \sum_{i\in I_1} |B(i)| + |I_2| + \sum_{i\in I_2} |B(i)| =$$

$$= |I| + |I_1| \cdot \log_2 |I_1| + |I_2| \log_2 |I_2| = N + k \cdot \log_2 k + (N-k) \log_2 (N-k) =$$

$|I| = N$
$|I_1| = k$
$|I_2| = N-k$



symetrická podle N/2

$$\geq N + 2 \cdot \frac{N}{2} \log_2 \frac{N}{2} = N + N \log_2 \frac{N}{2} = N + N \log_2 N - N = N \log_2 N$$

$\log_2 2 = 1$

$N = n!$

$\sum_\pi |A_\pi| \geq n! \log_2 n!$

~ primitivní # porovnání

	nejh.	ϕ	nejl.
Merge	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$
Heap	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$
Quick	$\Theta(n^2)$	$\Theta(n \log n)$	$\Theta(n \log n)$
Bubble	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(1)$

Q: předpokl. polaprod. ϕ prodá $n!$ permutací

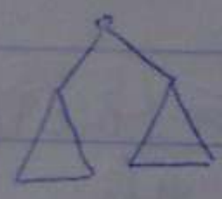
v 1/n případů jako pivot 1 $(n-1)!$
2 $(n-1)!$
⋮

$$\frac{1}{n!}\sum_{\pi} P(\pi) = \frac{1}{n!}\sum_{i=1}^{n}\sum_{\substack{\pi \\ \pi(p)=i}} P(\pi) = \frac{1}{n!}\sum_{i=1}^{n}\sum_{\substack{\pi \\ \pi(p)=i}} [n + P(\mu) + P(\lambda)]$$

↑ na místě pivota má hodnotu i

↓ posunání s pivotem

$i-1$ μ
1
$n-i$ λ



ϕ hloubka ϕ výhl. stromu

Hloubka rekurze = součet počtu posunání než to vyřešíme

$\geq \alpha \cdot n \log_2 n$

- volba pivotu → medián

nejh. $\Theta(n \log n)$

- randomizovaný quicksort – náhodný výběr pivota pokaždé

- pro 1 algoritmus různé průběhy výpočtu

⇒ dojdeme ke definici vzorečku

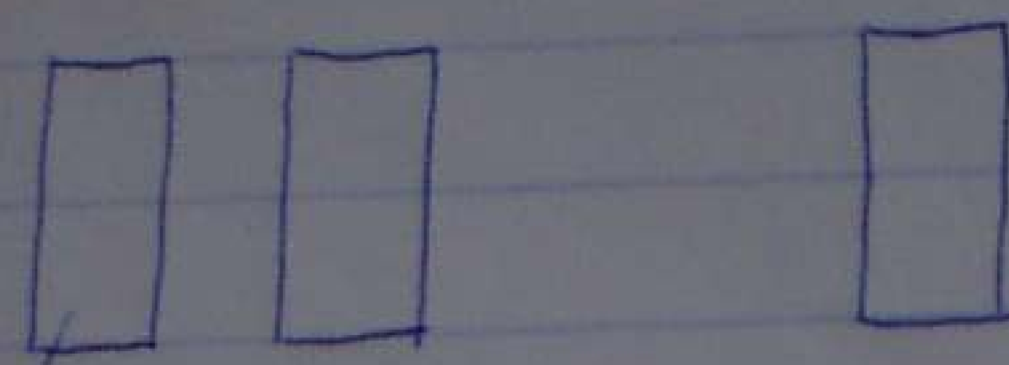
průměr možných průběhů výpočtu

malá pravděpodobnost, že se odchýlí od ϕ

alg.

$a_{min} \le a_i \le a_{max}$ $a_1, \dots a_n$

$a_{min}, a_{min}+1, \dots, a_{max}$



boxy

do boxu uložíme číslo
vypíšeme obsah boxů za sebou
některé mohou být prázdné

$n + (a_{max} - a_{min} + 1)$ radix sort

- netřídí na základě porovnání → adresování pomocí klíčů
- číslo se používá jako index → tříděná čísla by měla být přirozená čísla
  rozmezí min–max musí být rozumné

[prostá posl. – v 1 boxu 1 číslo → pole
nejdříve spočítat, jak budou boxy velké

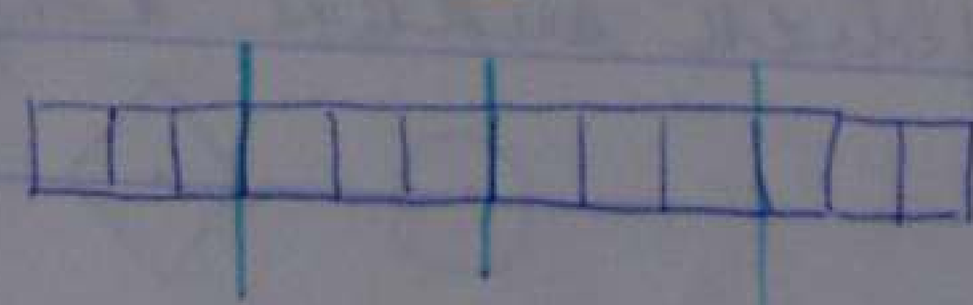
pole $b_{a_{min}}, \dots b_{a_{max}}$

$A_1, \dots A_n$ → posl. ve setř. podobě

for $j = a_{min}$ to a_{max} $b_j = 0$;
for $i = 1$ to n do b_{a_i}++;
for $j = a_{min}+1$ to a_{max} $b_j = b_j + b_{j-1}$;
↖ kolik prvků je ≤ tomuto prvku
* for $i = n$ downto 1 do $A[b_{a_i}] = a_i$; b_{a_i}--;

*u víceprůchodové ne čísla, ale části čísla

velké rozmezí
8bytové čísla



– setřídit podle 1. bytu do stejných částí, třídit podle 2.
– 0–255 (a_{min} – a_{max})
setřídit podle nejméně významného bytu, podle 2.
pokud jsou stejné → zachovat původní pořadí *

lexikografické uspořádání

slova stejně dlouhá → projdeme tolikrát, jaká je délka sl.

nestejně dlouhá → doplnit speciálním symbolem menším než jakékoliv písmeno

ahoj
she␣
a␣␣␣

→ nejdříve setřídit podle délky

- nejob...
- ahoj
- she
- a

při prvním průchodu setřídíme nejd. jen slova, pak přidáme kratší

O(součet délek slov) — vstup

j=1 do velikost abecedy

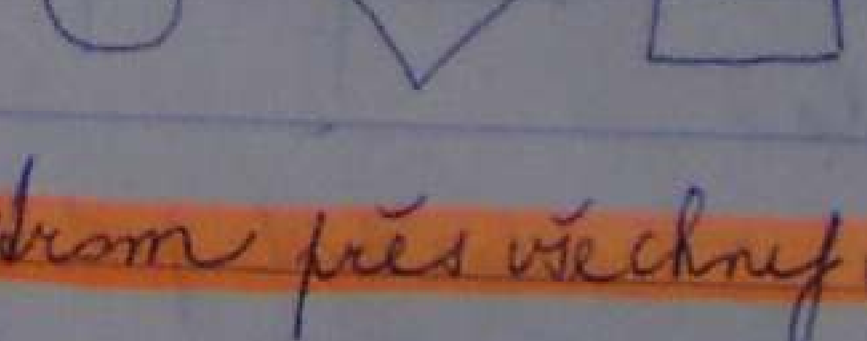
Kostra grafu

- všechny hrany povoleny
- některé hrany zakázány – podle euklidovské vzdálenosti (L2) — metriky
- povolené hrany ohodnoceny

L1 manhatanské

L∞ maximovská

obecné schéma
- vybereme komponentu
- najdeme nejbližší sousední komponentu,
- propojíme



Úkol – souvislý graf bez cyklu → strom přes všechny vrcholy, povolené hrany
– hrany s kladným ohodnocením → co nejmenší součet ohodnocení

kostra = spanning tree

alg. XI

X n vrcholů

R množina povolených hran

hledáme $T \subseteq R$, kde T je minimální kostra

$S \subseteq R$ aktuální množina hran, v 1. kroku přidáme 1 hranu

V každém okamžiku výpočtu S je částí min. kostry

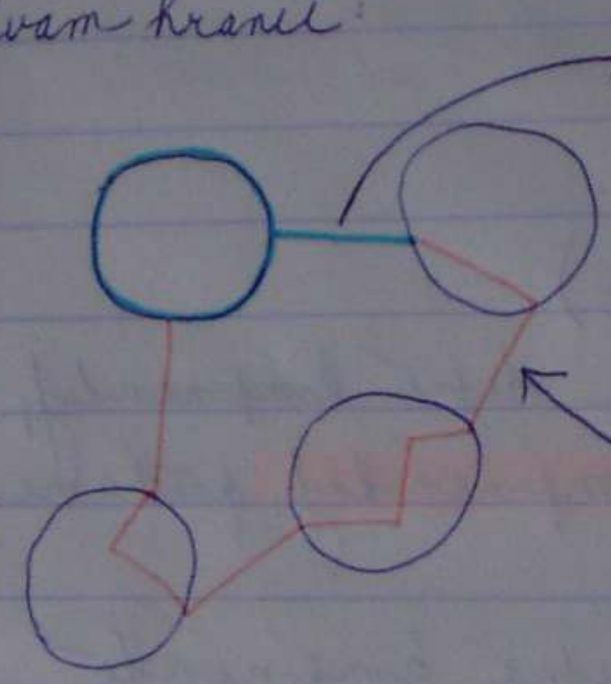
S je kostra $S \subseteq T$ n-1 hran $\Rightarrow S = T$

na konci výpočtu

- na začátku platí, žádný krok nepokazí

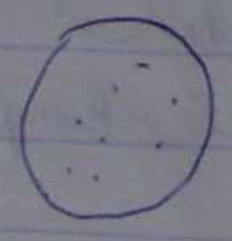
$S = \phi$, má řešení T $S \subseteq T$

Když přidávám hranu:

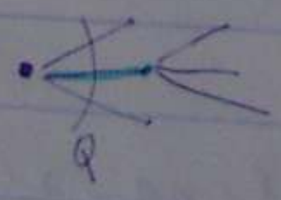


$E(m^2)$ $V(n \cdot m)$

Prim: - vybereme vrchol, propojíme s nejbližším
- vždy vybereme komponentu s původním vrcholem
- velká komponenta a vzdálené body



Q - v prioritní frontě (Halda, jinou se hledá min)
hrany z komponenty $v(\log n)$



min.
přidáme hranu do komponenty
pokud hrany z vrcholu už nejsou v Q

přidáme ji do Q_1 pokud byla, vyhodíme

$n-1$ min

$2m$ insert nebo delete

$3m$ $O(\log m)$

↑ velikost fronty

$O(m \log m)$

$m \leq n^2$

$\log m \leq 2 \log n$ $O(m \log n)$

Kruskal (Jarník):

X

R: $h_1 \dots h_m$

$c(h_1) \leq c(h_2) \leq \dots \leq c(h_m)$

pokud hrana spojuje vrcholy ze dvou různých komponent, vyhodíme ji; pokud různé komponenty, přidáme ji → sjednocení komponent

h_i spojuje vrcholy x, y

předchozí hrany už jsou ve dvou komponentách

setřídění hran podle velikosti $O(m \log n)$

ve které komponentě x a y

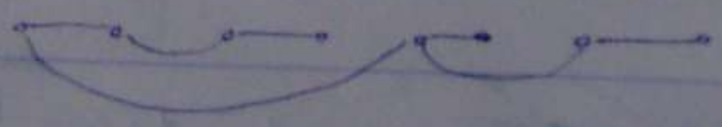
1) sjednocení komponent - přečíslováním vrcholů v komponentě → snadné zjištění, ve které komponentě je vrchol

zapamatoval si #vrcholů v komp. - přečíslovat menší komp.

nejhorší případ:

připojování jednotlivých k velké komponentě

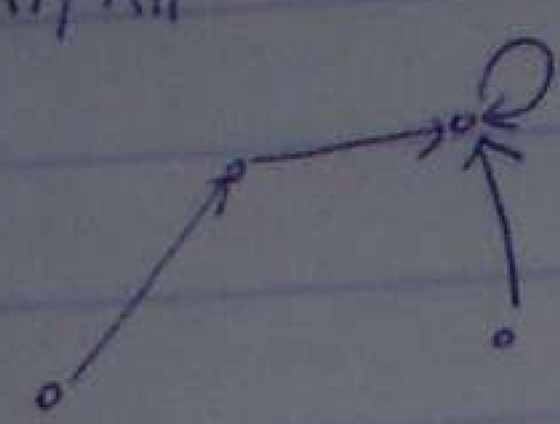
$1 + 2 + \dots + n-1$



2) snadné sjednocení

reprezentant komponenty → vrchol ukazatel na sebe při spojím na reprezentanta

alg XI, XII



obtížnější zjištění reprezentanta

nejh. případ

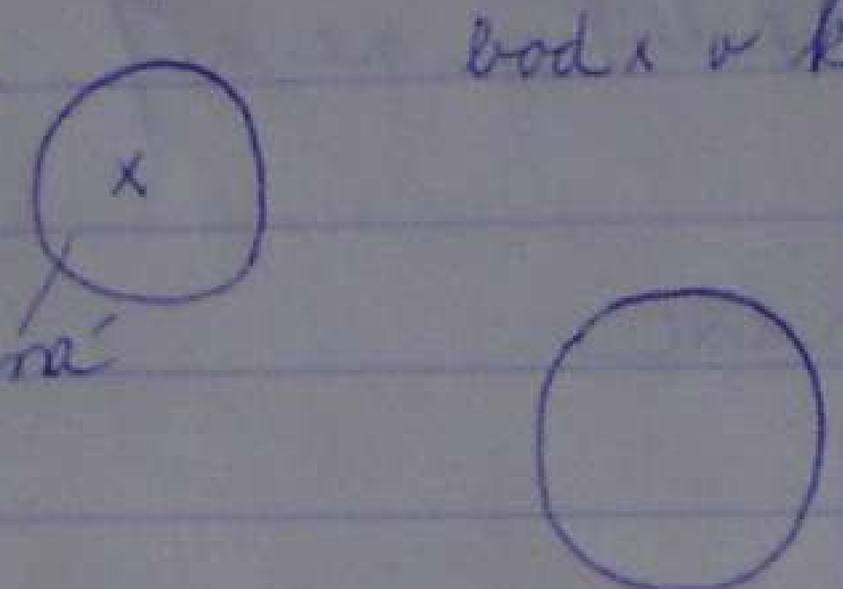
dotaz na reprezentanta třídy (cesta délky n-1)

$\frac{n(n-1)}{2}$

podřízená komponenta - v ní se přepisují čísla n. připojuje pointer
- bereme vždy menší komp.

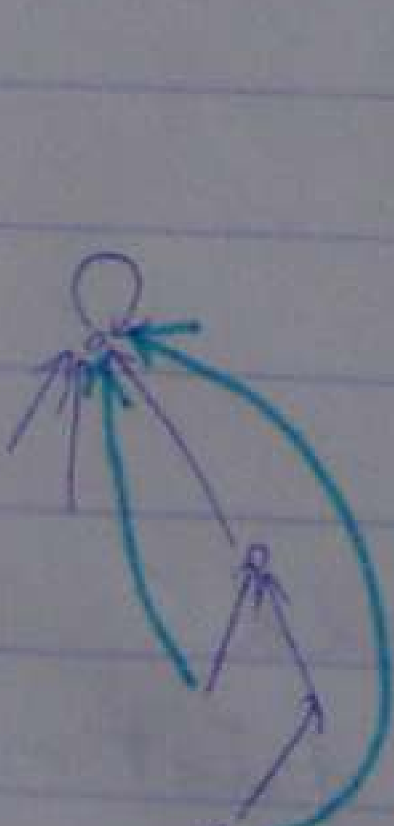
vrchol v podř. komp:

bod x v komponentě s k prvky



spojením ⇒ komp. s 2K

- žádný vrchol se nepřečísluje víc jak $\log_2 n$
  $n \log_2 n$
- přepojování pointerů – vzdálenost od reprezentanta se zvýší o 1
  \# přepojování $\log_2 n$
  ⇒ hloubka stromu logaritmická



(projdeme cestu $\log_2 n$)
zeptáme se z jakého reprezentanta má vrchol, připojíme přímo na něj, příští dotaz bude konstantní

záleží na pořadí, jak se ptáme na vrcholy

Čas strávený na 1 vrcholu menší než $\log^* n$

$$\varphi(1) = 1$$
$$\varphi(k) = 2^{\varphi(k-1)}$$

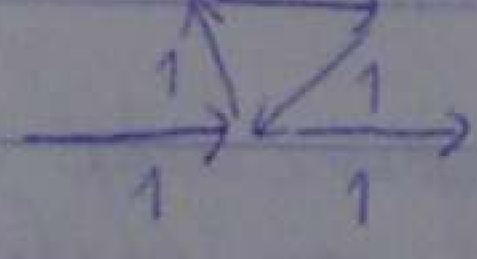
$\log^* n$ - inverzní fce k $\varphi(k)$

k	$\varphi(k)$
1	1
2	2
3	4
4	$2^4 = 16$
5	$2^{16} = 65\,000$
6	$2^{65\ldots}$
7	$2^{2^{65\ldots}}$ → tuto hodnotu již nezapíšeme

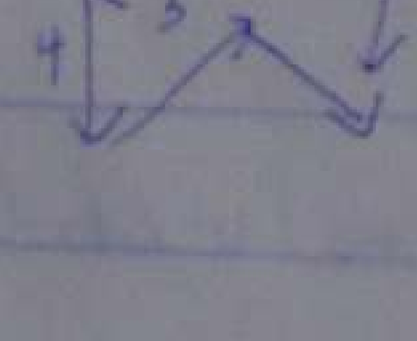
všechny operace v konst. čase
→ **faktorová množina**
pro reprezentanta si pamatuju kolik je na něj
zapojeno členů
slučujeme menší s větší komponentou
nejnáročnější setřídění hran podle ceny

Nejkratší cesta

- orientovaný graf
- nejkratší cesta z vrcholu A do všech vrcholů grafu (do vrcholu B)
- ze všech vrcholů do všech vrcholů
- ohodnocení kladnými čísly → délka hrany
  i zápornými → cena hrany



nejkratší cesta není definována
→ nejkratší prostá cesta



se zápornými čísly najdeme nejdelší
prostá cesta s n-1 hranami → přes vrcholy
NP-úplné

Alg XII
nejkr. cesta - prostá
- acyklická (pokud kladné ohodnocení)

Dijkstra ohodnocení nezáporná
Bellman-Ford nejsou záporné cykly, najde nejk. cesty pokud dám, jinak ho nejbližší, nejpomal.
CPM critical path method
nejsou orientované cykly
cyklický alg.
nejdelší cesta → optimalizace

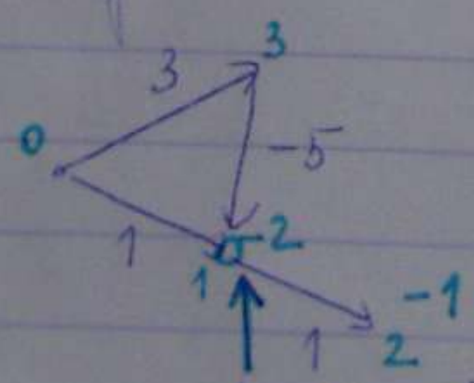
Dijkstra:
$V, H, V_0 \in V$
vrcholy - definitorové
- nedefinitorové
$L(V,W)$ - délka hrany z V do W
$D(V)$ - odhad délky z V do V_0
1. inicializace
$\forall$ nedefinitorové $D(V_0) = 0$
ostatní vrcholy nedef. $D(V) = +\infty$
while $\exists$ nedef. vrchol do 4
$V :=$ nedef. vrchol s minimální $D(V)$;
V označ jako def.;
$\forall$ hrany (V,W) if $D(V) + L(V,W) < D(W)$ then $D(W) := D(V) + L(V,W)$
;

implementace
+ hrany
$O(n^2 + m)$
nedef. vrcholů nalezení min

prioritní fronta
halda
insert
delete min
m snížení klíč
$O(\log n)$
$(n+m)$

Fibonacciho halda
$O(n \log n + m)$

- nezacyklí se - while cyklus proběhne nx

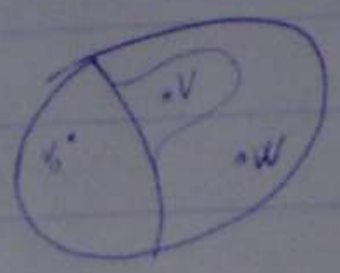


dendo vrchol označený jako nedef.
→ nepravé

Věta

invariant
1) v_i def & W je nedef $\Rightarrow D(v) \leq D(W)$
2) $D(W)$ je délka nejkratší cesty z v_0 do W, že kromě W všechny vrcholy jsou def.
na konci případu 1 vrcholy def $\Rightarrow$ nejkr. cesta

1) na začátku 1 nedefinidované



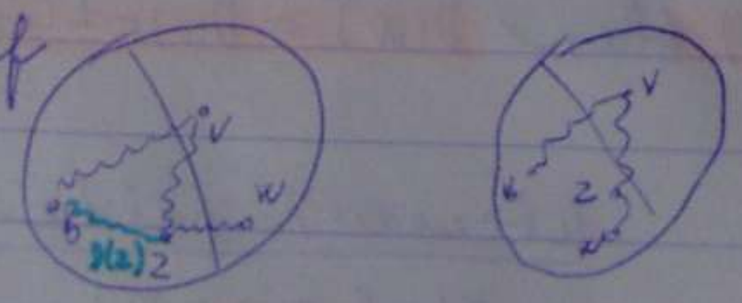
v označíme jako def.
vzali jsme si vrchol s min. $D(v) \Rightarrow D(v) \leq D(w)$
W musí být mezi nedef. vrcholy, které byly mezi def. $D(v) \geq D(w)$
$D(w) := D(v) + L(v, w)$
$\geq D(v)$
protože délka hrany je nezáporná

v_i ~~defin~~ nedefinidované vrchol

$v_0 v_1 \dots v_i \dots v_k = w$
$v_0 \dots v_{i-1} v_{i+1} \dots v_{k-1}$ def.

a) $i = k-1$
$v_0 \dots v\, w$
délka $= D(v)$
délka $< D(w)$ $\quad D(v) + L(v, w)$

b) $i < k-1$



z předposledního vrchol cesty def
$D(z) \leq D(v)$
$D(w) \leq D(z) + L(z, w)$

rekonstrukce cesty:
$P(v_0) = v_0$
$P(w) = v_i$
$\Rightarrow$ w nebyla

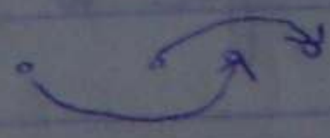
alg

CPM acyklický orientovaný graf

Topologické uspořádání grafu

vrcholy uspořádáme $v_0, \dots v_{n-1}$

(v_i, v_j) hrana $\Rightarrow i < j$



Lemma: V acykl. orient. grafu existuje vrchol, do kterého nevstupuje hrana

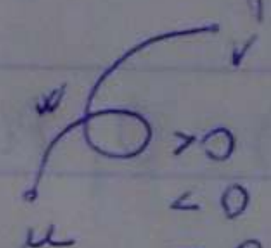
konečném

program

~ hledají nejkratší sledy
cest je konečně (n+1)!

Věta: Buď G graf bez záporných cyklů, S nejkratší sled mezi nějakými $u, v \in V(G)$. Pokud S je cesta.

Důkaz: Kdyby S nebyla cesta:



vezmeme sled bez kladného příspěvku

G graf, $v_0 \in V(G)$ počáteční vrchol, M, N

$L(u,v)$ délka $(u,v) \in E(G)$

$L(v_1, v_2, \dots v_k)$ délka sledu

$D(v)$ délka nejkr. sledu $v_0 \to v$ (nebo $+\infty$, pokud neexistuje)

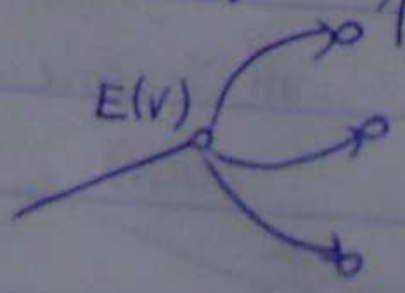
$E(v)$ odhad $D(v)$ (odhad = estimate)

$P(v)$ předchůdce

Kostrový alg. Vstup: G, L, v_0

Out: E, P

Scan(v)



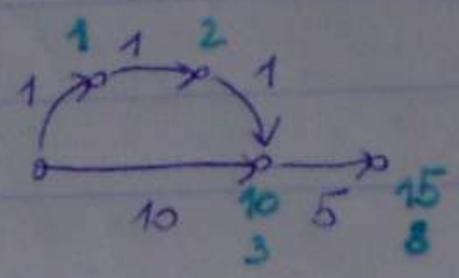
pokud se dostaneme přes E(v) lepe upravíme

$E(*) := +\infty$, $E(v_0) = 0$, $P(*) := ?$

repeat
  hodovo = true;
  for $v \in V(G)$:
    for w $(v,w) \in E(G)$:
      $l := E(v) + L(v,w)$
      if $l < E(w)$: $E(w) = l$, $P(w) := v$
        hodovo := false

] Lan(v)

until hodovo



stále připočítává, i hodnoty, kt. se nemění

Bellman-Ford:

Q fronta
Z(v) značka

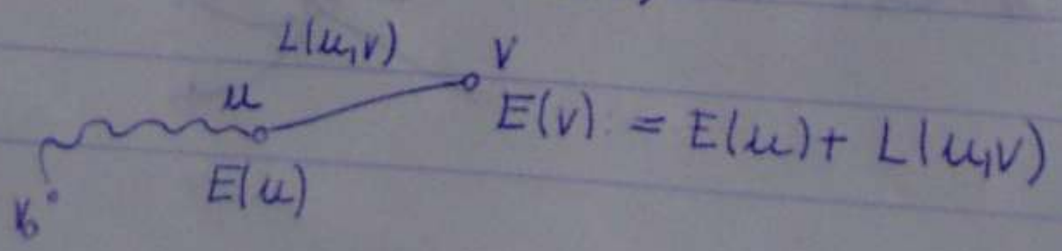
$E(*) := +\infty$, $E(v_0) = 0$, $P(*) := ?$
$Q := \{v_0\}$
while $Q \neq \phi$
  ~~$Q = Q_1$~~ $Q' := Q$, $Z(*) := false$ $O(N)$
  for $v \in Q$: $O(N)$
    for w: $(v,w) \in E(G)$: $l = E(v) + L(v,w)$ $O(M)$
      if $l < E(w) = E(w) := l$, $P(w) := v$
      if $\neg Z(w)$: $Z(w) := true$
      $Q' := Q' + w$
  $Q := Q'$.

false

Lemma 0: $\forall v$ $E(v)$ nikdy neroste

Lemma 1: $\forall v$ $E(v) < \infty \Rightarrow \exists$ sled $v_0, v_1, \dots v_K = v$: $E(v) = L(v_0, v_1, \dots v_K)$

Důkaz MI měníme $E(v)$



alg
- $E(u)$ už známe, pro něj platí
- v čase 0 platí $\forall$ vrcholy
- — 1 $\forall$ vrcholy

Lemma 2: $\forall$ sled $v_0 v_1 \dots v_k$ platí, že na konci k-té fáze je $E(v_k) \leq L(v_0 v_1 \dots v_k)$

Důkaz MI dle k

1) $k=0$ inicializace
2) $k>0$ na konci $(k-1)$ní fáze je $E(v_{k-1}) \leq L(v_0 \dots v_{k-1})$
buď i fáze, kdy se $E(v_{k-1})$ naposledy změnilo
$i \leq k-1 \Rightarrow v$ $(i+1)$ní fázi se v_{k-1} scanuje
$E(v_k) \leq E(v_{k-1}) + L(v_{k-1}, v_k) \leq L(v_0 \dots v_{k-1}) + L(v_{k-1}, v_k) =$
$= L(v_0 \dots v_k)$

κ 1) $E(v) \geq D(v)$
κ 2) $D(v) \geq E(v)$ $\Rightarrow E(v) = D(v)$

Věta BF algoritmus se zastaví $\Leftrightarrow$ v neobsahuje záporný cyklus dosažitelný z v_0. Pokud se zastaví, pak
1) proběhne $\leq N$ fází
2) spotřebuje čas $O(N(M+N))$
3) $\forall v$ $E(v) = D(v)$
4) $\forall v$ $E(v) < \infty$: $P(v)$ předchůdce na některé nejkratší cestě

Důkaz 1) $\Leftarrow$ κ L1, L2 + 1), 3)
$\Rightarrow$ kdyby se alg. zastavil, $\exists$ záporný cyklus dosažitelný z v_0 ($E(v) + K \cdot L(\text{záp. cyklus})$?!)
$E(v)$ by muselo být $-\infty$ $\Rightarrow$ spor

2) indukcí dle 1)
κ N fází
1 fáze: ve frontě se vrchol 1x (přidán, označen, ...)
každou hranu max 1x

v každém kroku algoritmu

Lemma P: je-li $u = P(v)$ definováno, pak $E(u) + L(u,v) \leq E(v)$

Na konci alg. platí =

Důkaz: poslední přiřazení do $E(v)$, které nastavuje $P(v) = u$,

tehdy platilo =

$\Rightarrow$ pak jen klesá $E(u)$

$LP^* \; \forall v_1 \ldots v_k \; \forall i \; v_{i-1} = P(v_i)$

$E(v_1) + L(v_1 \ldots v_k) \leq E(v_k)$

$E(v_0) + L(v_0 \ldots v_k) \leq E(v_k) = D(v_k)$

pokud jdeme po předchůdcích ... nejkr. sled

nebsahne cyklus (přidání

poslední hrany do cyklu)

Detekce záporných cyklů

Věta $\forall v \in V(G)$: v leží na dosažitelné záporné kružnici $\Leftrightarrow$ mezi N- tou a 2N- tou fází se $E(v)$ změní

Důkaz $\Leftarrow$

$\Rightarrow$ po N-té fázi je $E(v)$ konečné

sled $v_0 \ldots v_k = v$ přidáváme-k němu průchody zápornou kružnicí, mezi N a 2N espoň 1 padne

Floyd-Warshall:

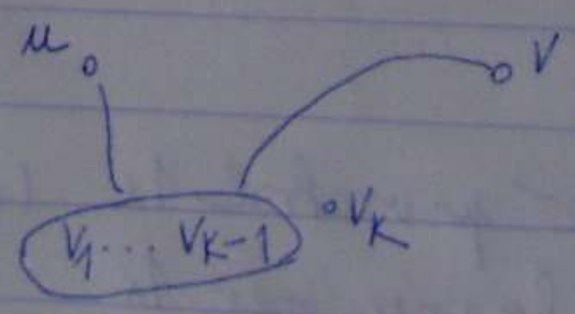
nejkr. cesta z každého vrcholu do každého

$D^k(u,v)$ = délka nejkr. cesty $u \to v$ v $G \cap \{u, v_1, v_{11} \ldots v_k\}$

$D^0(u,v) = L(u,v)$

$D^n(u,v) = D(u,v)$

$\forall u,v \; D^{k-1}(u,v) \rightsquigarrow \forall u,v \; D^k(u,v)$



alg

$$D^k(u,v) = \begin{cases} D^{k-1}(u,v) \\ \min \; D^{k-1}(u,k) + D^{k-1}(k,v) \end{cases}$$

N matic $N \times N$
2 matice
1 matice – přepisujeme na sdě'Z

In: $D(u,v)$ = delky hran
Out: $\neq D(u,v)$ = delky nejkr. cest

```
for k := 1 to N:
    for u := 1 to N:
        for v := 1 to N:
            D(u,v) := min( D(u,v), D(u,k) + D(k,v) )
```

$\} O(N^3)$

bez zápor. kružnic (sklepní 2 cest) → nejkr. kružnice