Obecné

Časová složitost $T(n) := \max \{ t(x) \mid x \text{ je vstup velikosti } n \}$

Prostorová složitost $S(n) := \dots s(x) \dots$

DFS + BFS

DFS stavy:

- zavřený (už bylo vše dokončeno)
- otevřený (koná se)
- nenalezen (nenašel se)

DFS: → inicializace jde na začátku se všemi vrcholy jako „nenalezený"

1. stav(v) ← otevřený
2. Pro hrany vw:
3. Pokud stav(w) = nenalezeno
4. DFS(w)
5. stav(v) ← zavřený

Budíky: In(v) a Out(v) – reprezentují „čas" od spuštění

Lemma: DFS se zastaví v čase $O(n+m)$

Stavy: $N \to O \to Z$

⇒ Vrchol otevřen $!= 1$ ⇒ DFS na vrchol volá max 1.

1) $O\Big(n + \underbrace{\sum_{v \in V} \text{deg-out}(v)}_{m}\Big) = O(n+m)$

Lemma: Po dokončení DFS je tv stav(v) = nenalezený / zavřený ⟺ pak je dosažitelný z v_0

⇒ Pokud jsme zavřeli, museli jsme ho otevřít.
Takže jsme se zavolali na ten vrchol, tzn. ∃ hrana uv z otevřeného u. IP ⇒ u je dosažitelný z v_0

⇐ Existují dosažitelné, nenalezené. Zvolme v špatný a nejbližší k v_0.

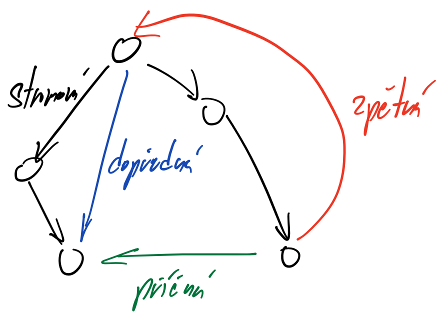
$p :=$ předposlední vrchol nejkratší cesty z v_0 do v.

↳ pak hrana pv byla objevena, tudíž bylo zavoláno na v. ↯

Definice hran

- zpětná
- dopředná
- příčná
- stromová

tyto existují i v grafu neorientovaném



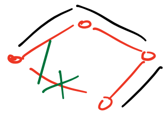
Typ hrany zjistíme v $O(1)$ – při prohledávání grafu

Věta: DFS v čase $O(n+m)$ a prostoru $O(n+m)$ najde dosažitelné vrcholy a klasifikuje dosažitelné hrany.

Most je hrana e v grafu ⟺ $G-e$ má více komponent než G.

Lemma: Hrana e není most <=> e leží na kružnici.

- Zpětná hrana nikdy nebude most

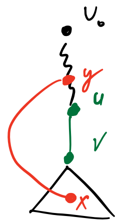


Stále si udržuju jednu komponentu cestou „dolů"

uv leží na kružnici

⇕

$\exists xy \in E$ zpětná t.ž. x je potomkem v, y je předkem u } $in(y) < in(v)$



$$low(v) := \min\left\{ in(y) \;\middle|\; \begin{array}{l} xy \text{ je zpětná} \\ \& \ x \text{ pod } v \end{array} \right\}$$

<=> $low(v) < in(v)$

to se vypočítá při odchodu (uzavírání) v, tedy jdu odzadu ⇝ $O(deg(v))$

Věta: Alg. najde všechny mosty v čase a prostoru $\Theta(m+n)$

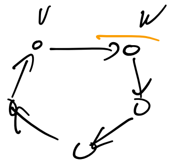
Acyklické orientované grafy DAGy

Existuje dosažitelný zpětný cyklus <=> DFS najde zpětnou hranu.

<= triviální

=> zvolme $v \in C$ s min. $out(v)$

-jediné, kde roste out, je na zpětné



Topologické uspořádání je lin. uspořádání $\preceq$ na $V(G)$ t.ž. $\forall xy \in E(G): x \preceq y$.

- alt. je to očíslování vrcholů

Zdroj $v \in V(G)$ je $v \equiv deg^{in}(v) = 0$

Stok $v \in V(G)$ je $v \equiv deg^{out}(v) = 0$

Graf má TU <=> je to DAG

=> triviální, protože pak není cyklus na grafu

<= Postupně budu hledat a odtrhávat zdroje, získám DAG

Lemma: Každý DAG má zdroj

- půjdu odzadu (jako bych šel od listů do kořene)

Pořadí, v němž DFS opouští vrcholy, je opačné topologické.

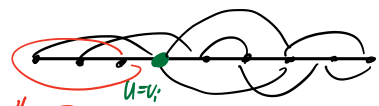
Triviální z myšlenky DFS

Topologická indukce.

Příklad: Zvolme $u \in V$, chceme $\forall v \in V: c(v) := \#$ cest z u do v

└ pro DAG

Nechť $v_1 \dots v_n$ je TU.



celkem $\Theta(n+m)$

$v_i = 1$

Pro $j > i$ indukčně díky tomu, že to jsou předchůdci

$c(v_j) = \sum_{p:\, pv_j \in E} c(p)$ --> součet předchůdců, tedy už je vypočítaný

Silná souvislost

Relace $\rightsquigarrow$ na V: $u \rightsquigarrow v \equiv \exists$ sled z u do v

$\leftrightsquigarrow$ na V: $u \leftrightsquigarrow v \equiv u \rightsquigarrow v \ \& \ v \rightsquigarrow u$

$\leftrightsquigarrow$ je ekvivalence — třídy jsou komponenty silné souvislosti

G je silně souvislý <=> # komp. sil. souv. = 1

Graf komponent $C(G)$: vrcholy: komponenty G

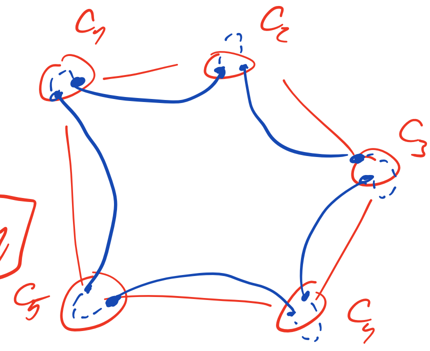
hrany: (C_i, C_j) : $\exists x \in C_i, \exists y \in C_j, xy \in E(G)$

Lemma: Graf komponent je vždy DAG

- orientovaný je z definice
- kdyby měl cyklus:

  různé: $C_1, C_2 - C_k, C_1$

- tak pak jsou všechny $C_1 - C_k$ ve stejné komponentě ↯



Hledání komponent silné souvislosti

Ze stokové komponenty se nedostanu do jiné komponenty

Pokud spustíme opakovaní DFS, pak vrchol s max. out leží ve zdrojové komp.

Protože se k němu dostanu (uzavřu ho) nejpozději, takže až ve chvíli, kdy jsem všechno ostatní uzavřel.

Transpozice grafu $G^T := (V(G), \{vu \mid uv \in E(G)\})$

G je DAG $\iff$ G^T je DAG

G a G^T mají stejné ekv. třídy ~>

zdroj $\xleftrightarrow{G^T}$ stok (prohození)

Algoritmus

-> je to ale pomalé

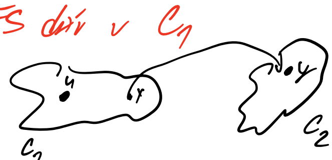
Najdeme zdrojový vrchol z G^T, takže máme stokový vrchol z G. Na něj pustím DFS, které najde stokovou komponentu, takže takhle najdu všechny komponenty.

Zrychlení

-> Procházíme vrcholy v pořadí klesajících outů v G^T, pokud ještě nemají přiřazenou komponentu, spustím z nich DFS.

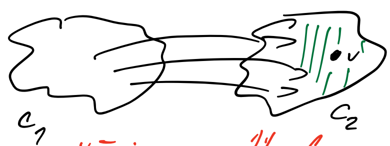
-> Pokud C_1, C_2 jsou komponenty pak platí, $\max\limits_{u \in C_1} out(u) > \max\limits_{v \in C_2} out(v)$

Případy: -> DFS dřív v C_1



- nejdřív se vrátím z C_2, pak až z C_1

DFS nejdřív v C_2



Než jsem se vrátil z C_2, tak jsem se nemohl dostat do C_1. ⊠

Finální algoritmus

1) Sestrojím G^T
2) $Z \leftarrow$ prázdný zásobník
3) Opakovaní DFS v G^T, při opouštění vrcholu přidáváme do Z.
4) $\forall v$ $komp(v) = \emptyset$
5) Odebíráme v ze Z:
6) Pokud $komp(v) = \emptyset$
7) DFS(v), chodíme jen do vrcholů s $komp = \emptyset$ a nastavujeme $komp$ všude na v.

-> Celé je $\Theta(n+m)$, prostor i čas !

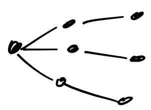
Algoritmus najde komp. souvislosti v čase $O(n+m)$.

Nejkratší cesta – Dijkstra

Délka cesty $u \to v := \ell(P) = \sum_{e \in P} \ell(e)$

Vzdálenost $d(u,v) := \min\{\ell(P) \mid P$ je uv-cesta$\}$

Strom nejkratších cest :



kompaktní reprezentace vzdálenosti

- strom na V
- podgraf G
- orientovaný od kořene u
- $\forall v \in V$: cesta z u do v je jedna z nejkratších cest mezi u-v

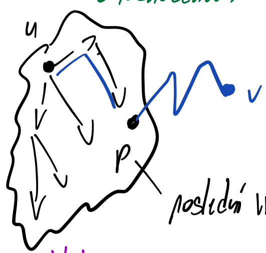
– myšlenka BFS a budíky, kdy se prochází po „vlnách"

Lemma: Pokud S je uv-sled, pak P je uv-cesta t.ž. $\ell(S) \geq \ell(P)$

– Opět pomocí vystřižení cyklů, pokud nějaké existují

– rozbije se se zápornou hranou

Lemma: Strom nejkratších cest existuje i v ohodnoceném grafu.



prefix nejkratší cesty je zase nejkratší cesta.

Taková cesta je tedy zase nejkratší.

Dijkstra alg.:

1) $\forall v$ $h(v) \leftarrow +\infty$, $s(v) \leftarrow$ nenalezený
   $h(u) \leftarrow 0$, $s(u) \leftarrow$ otevřený
2) Dokud existují otevřené vrcholy
3) $v \leftarrow$ otevřený s $\min(v)$ — $O(n)$
4) Pro všechny hrany vw:
5) Pokud $h(v) + \ell(v,w) < h(w)$:
6) $h(w) \leftarrow h(v) + \ell(v,w)$
7) $s(w) \leftarrow$ otevřený
8) $s(v) \leftarrow$ zavřený

$O(1)$ $O(n)$

každou hranu jen jednou $O(m)$

vrchol zavřu jen jednou

Časová složitost:

Celkem tedy $O(n^2)$

– minimum se dá počítat lépe

implementace pomocí haldy

Prostorová složitost:

$O(n+m)$, jelikož si pamatuju jen hrany a vrcholy

$O(n \cdot T_{insert} + n \cdot T_{pop} + m \cdot T_{decrease})$

Které operace potřebuju:

- Pop (min) $\leq n$
- Insert $\leq n$
- Decrease $\leq m$ – sníží ohodnocení vrcholu

Pole: $O(n \cdot 1 + n \cdot n + m \cdot 1) = O(n^2)$

Halda: $O(n \cdot \log n + n \cdot \log n + m \cdot \log n) = O((n+m) \log n)$

pomalé pro husté grafy

Relaxační algoritmy

1) $\forall v$ $h(v) = +\infty$, $s(v) =$ nenalezený
   $h(u) = 0$, $s(u) =$ otevřený
2) Dokud $\exists v$ otevřený:
   relaxuj v.

Dijkstra vybere nejmenší otevřený

① $\forall v$ má ohodnocení $h(v)$ z počátku $+\infty$ ⇣ konverguje k $d(u,v)$

② Relaxace – vylepšení ohodnocení vrcholů pomocí ohodnocení jiných vrcholů

③ Stavy, abych se neztratil

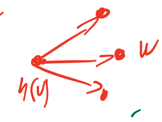
otevřený $\Leftrightarrow$ od poslední relaxace se změnilo hodnocení daného vrcholu.

Lemma D: Pokud se alg. zastaví, pak $\forall v$:

v je dosažitelný z u
$\Updownarrow$ --> korektnost DFS/BFS
v je uzavřený
$\Updownarrow$ triv.
h(v) je konečný

Invariant O: $\forall v$ $h(v)$ nikdy neroste (1)
Pokud je $h(v)$ konečné, je rovno délce nějakého uv sledu (2)

1) Triviálně z definice relaxace
2) inic: OK
relaxace:



$h(v) = \ell(s)$
$h(v) + \ell(v,w) = \ell(s')$

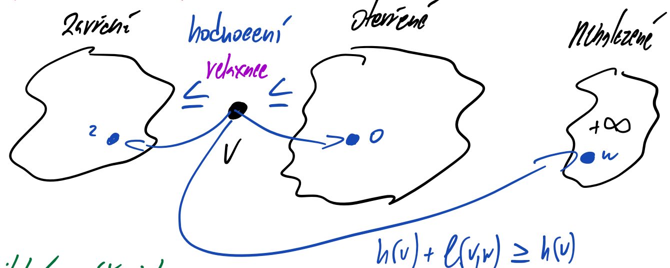
Lemma V: Pokud se alg. zastaví, pak $\forall v \in V$: $h(v) = d(u,v)$

1) Pokud není dosažitelný, je $h(v) = +\infty = d(u,v)$, jinak vše konečné
2) Díky invariant O: $h(v) \geq d(u,v)$ --> Kdyby $h(v) > d(u,v)$, pak máme díky relaxaci p (vyvolává 2 poslední relaxace) $\underbrace{h(v) \leq h(p) + \ell(p,v)}_{d(u,v)}$ ↯

Pro Dijkstra na grafech s $\ell \geq 0$:

Invariant M: Kdykoliv je o otevřený a z zavřený:
1) $h(z) \leq h(o)$
2) $h(z)$ se nezmění

a) zpočátku ✓
b) při relaxaci: $h(z) \leq h(v) \leq h(o)$
- nově nalezení $h(w) \geq h(v)$
- do zavřených nelze měnit hodnotu



Věta: Dijkstra zavírá vrcholy v pořadí podle vzdálenosti, každý dosažitelný právě jednou.
$h(v)$ v okamžiku zavření je rovno $d(u,v)$

- Důkaz: plyne z výše zmíněné invarianty.

Bellman-Fordův alg.

- relaxační algoritmus s otevřenými vrcholy ve frontě
- relaxace pracuje s frontou
  ↳ zavírá nejstarší z otevřených vrcholů

fází $\leq n$

Df: Fáze výpočtu:

F_0 := otevření u

F_i := zavírání vrcholů otevřených v F_{i-1}, otevírání následovníků.

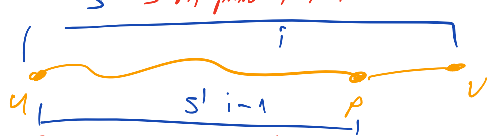
Věta: Bellman-Ford najde nejkratší vzdálenost v čase $O(n \cdot m)$ pro graf bez záporných cyklů.

$\forall$ fázi $\forall$ vrchol zavřen jen jednou, takže nejvýše zavřeno m hran v jedné fázi

Lemma: Na konci fáze F_i:
$\forall v \in V$: $h(v) \leq$ délka nejkratšího uv sledu o max. i hranách.

<-> Pak po nejvýše n-1 fázích $h(v) \leq d(u,v)$
=> n-tá fáze nic nezmění

a) pro $i = 0$ ✓
b) Na konci i fáze.
vrchol v, nejkratší uv sled S:
- pokud S má < i hran, tvrzení platilo i v předchozí fázi
- S má právě i hran



Podle IP na konci F_i máme $h(p) \leq \ell(S')$, nastaveno nejvýše v F_{i-1}, tedy p otevřeno, nejpozději v F_i zavřeno, tedy relaxováno
$h(v) \leq h(p) + \ell(p,v)$

$h(p) \leq \ell(S')$
$h(p) + \ell(p,v) \leq \ell(S)$

Floyd-Warshallův alg.

Časová složitost $\Theta(V^3)$

- jde prakticky o tři vnořené cykly přes všechny hrany

Paměťová složitost $\Theta(V^3)$

Musím si pamatovat $V \cdot D^{|V| \times |V|}$ matic pro interní cyklus.

↳ ve skutečnosti $\Theta(V^2)$ stačí, protože můžu přepisovat na místě a musím znát jen D^{k-1} a D^k

Alg

- hrany mají ohodnocení

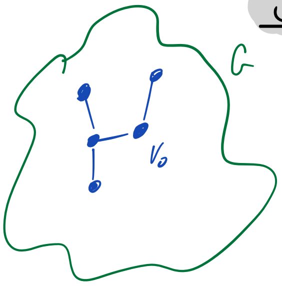
Matice $D^{|V| \times |V|}$, kde jsou doplněné vzdálenosti jednotlivých hran, na diagonále je nula, jinde $+\infty$

Pro $i = 1$ — $|V|$:
  Pro $j = 1$ — $|V|$:
    Pro $k = 1$ — $|V|$:
      Pokud $D(i,j) > D(i,k) + D(k,j)$
        $D(i,j) = D(i,k) + D(k,j)$

Problém minimální kostry: Jarník a Borůvka

- máme souvislý neorientovaný graf + váhy hran $w \to E \to \mathbb{R}$
- Chceme kostru T: $w(T)$ je minimální

Jarníkův Algoritmus



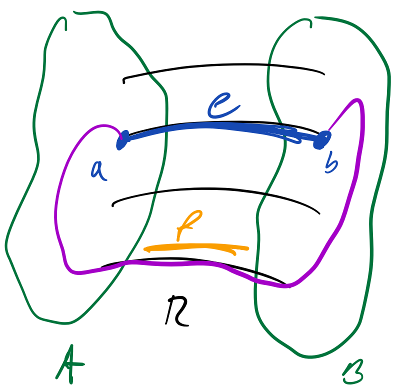
- pěstujeme strom T

1) Vybereme nejlehčí z hran mezi T a $G \setminus T$, přidáme do T.

Lemma: Jarníkův alg. se zastaví a T je kostra.
- vyplývá z principu alg. a postupy přidání listu

Řezové lemma:

Elementární řez je $R \subseteq E \equiv \exists A \subseteq V, B = V \setminus A, A, B \neq \emptyset$
t.ž. $R = E(A,B)$ → hrany mezi A, B



Nechť G je graf s unikátními hranami, R el. řez v G, e nejlehčí hrana v R, T nejlehčí kostra v G:

Pak $e \in T$

Sporem: Nejlehčí hrana řezu není v kostře:

Jelikož T je kostra, musí obsahovat cestu mezi a, b

Pak existuje hrana f, která je v řezu.

Když smažu f, rozbiju souvislost kostry, ale přidáním e ji opět opravím.

$\check{T} = T - f + e \qquad w(\check{T}) = w(T) \underbrace{- w(f) + w(e)}_{< 0}$

$\Rightarrow w(\check{T}) < w(T)$ ↯

Jarníkův alg. najde min. kostru.
- v každém kroku jsou hrany mezi T a $V \setminus T$ el. řez, tedy Jarník vždy vybere tu nejlehčí, tedy nalezne minimální kostru.

Všechny minimální kostry jsou si rovny

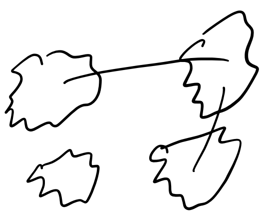
Nalezená kostra je podgrafem každé minimální kostry. Všechny kostry mají stejný počet hran, tudíž jsou si rovny.

Min. kostra je jednoznačně určena pořadím hran podle vah.

Jarník jen vždy porovnává a vybírá nejlehčí hranu z dané množiny.

Složitost: $O(m \cdot n)$

Borůvkův algoritmus



Postupně budují malé stromečky, $\forall$ stromeček vybere minimální hranu a tyto hrany přidáme. Zastavíme, pokud už je jen jeden stromeček.

Lemma: $\#$ fází $\leq \log n$

Na konci k-té fáze mají všechny stromečky alespoň 2^k vrcholů

$k=0 \to 2^0 = 1$ ✓

$k+1$: $\forall$ stromeček na konci fáze vznikne spojením alespoň 2 stromečků: $\#$ vrcholů $\geq 2^k + 2^k = 2^{k+1}$

Borůvkův alg. nalezne min. kostru v čase $O(m \cdot \log n)$

m je čas jedné fáze, $\log n$ je počet fází

Lemma: Výstup je minimální kostra
Hrany mezi stromečky jsou el. řez, tedy přidaná hrana je nejlehčí z el. řezu => je součástí min. kostry

Kruskalův alg + Union-Find problém

Kruskalův alg.

- setřídíme hrany od nejlehčí po nejtěžší
- postupně je od nejlehčí přidávám, pokud vytvoří cyklus, vynechám ji

$(m \cdot \text{Find} + n \cdot \text{Union})$

Lemma: Najde minimální kostru
Rozhodně vytvoří kostru
Minimální díky řezovému lemma.
- díky pořadí přidávání přidám jen tu nejlehčí z el. řezu.

Union-Find

udržujeme komp. souvislosti

Find(u,v) ... jsou u,v v téže komponentě?

Union(u,v) ... přidá hranu uv

Implementace:

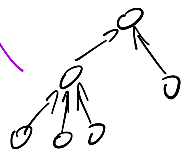
Pole: vrchol --> číslo komponenty

Find: $O(1)$
Union: $O(n)$
} Kruskal v čase $O(m+n^2) = O(n^2)$

Komponenty reprezentujeme jako keříky.

vrcholy si pamatují otce → 1 pole

Na začátku mám samostatné vrcholy, které postupně spojuju dohromady



Strom orientovaný ke kořeni, jeho vrcholy tvoří vrcholy komponenty

Find(u,v):
do u,v do kořenů keříků, kořeny porovnáme
Složitost: $O(\text{hloubka keříku})$

Union(u,v):
Najdeme kořeny u', v'
Pokud $u' = v'$: hotovo
Jinak přidáme hranu mezi kořeny
Složitost: $O(\text{hloubka keříku})$

Vylepšení:
udržuji si hloubky keříků, vždy připojím v Unionu mělčí pod hlubší

Lemma: keřík hloubky h má alespoň 2^h vrcholů

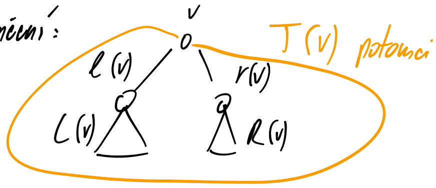
↳ hloubky jsou $\leq \log n$

Důsl. Union a Find trvají $O(\log n)$

Kruskalův alg.: $O(\underbrace{m\log n}_{\text{sort}} + \underbrace{m\log n}_{\text{findy}} + \underbrace{n\log n}_{\text{uniony}}) = O(m\log n)$

BST

Značení:



$h(v)$ hloubka := max. počet hran cesty mezi v a listem

- pokud chybí syn, pak je to None.

BST

- Vrcholy obsahují klíče $k(v) \in \mathcal{U}$
- a platí:

$$\forall \ell \in L(v): k(\ell) < k(v)$$

$$\forall r \in R(v): k(r) > k(v)$$

↳ všechny klíče jsou různé

Show: (Enumerate) $\Theta(n)$

Find: $\Theta(\text{hloubka})$

Insert: $\Theta(\text{hloubka})$ - připojuji na místo None

Delete:

1) Pokud nemá syny, rovnou mažu
2) Má jednoho syna, dosadím tam syna
3) Má oba syny, nahradím vrchol jiným listem (např: vpravo dole)

$\Theta(\text{hloubka})$

Dokonale vyvážený BST

Strom je dokonale vyvážený $\equiv$

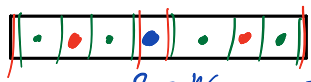
$$\forall v: \big| |L(v)| - |R(v)| \big| \le 1$$

Hloubka d.v. BST $\le \log_2 n$

- každým krokem snížím počet prvků pod sebou o polovinu

Tvorba ze setříděné posloupnosti:

$x_1 < x_2 \ldots < x_n$



$S = \lfloor n/2 \rfloor$ → musí to být ten prostřední prvek

- rekurzivně beru prostředky posloupností

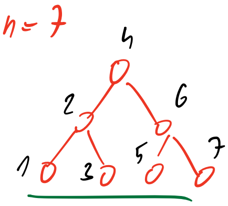
$\Theta(n)$ → každý vrchol navštívím jen jednou.

Věta: V každé implementaci Insert/Delete v BST má alespoň jedna operace složitost $\Omega(n)$ pro nekonečně mnoho hodnot n

Zvolíme $n = 2^h - 1$, klíče očíslujeme $1 \ldots n$

↓ jednoznačně určen strom

$n = 7$



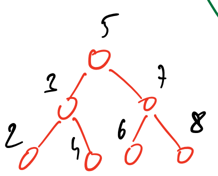
lichá v listech

Provedu Insert $(n+1)$

Delete (1)

Poté $2 \ldots n+1$ zase jednoznačně

Tzn. že Insert a Delete celkově trvají $\Omega(n)$



sudá v listech

$\Omega(n)$ vrcholů změnilo, zda-li jsou listy ⇒ v každém změna min. 1 ukazatele.

To by platilo opakovaně při dalším použití dvojice Insert + Delete

Hloubkově vyvážený BST (AVL - stromy)

Strom je hloubkově vyvážený ≡

↗ neexistující podstrom má hloubku -1

$$\forall v: \left| h(L(v)) - h(R(v)) \right| \le 1$$

Dokonale vyvážený $\Downarrow$ $\not\Uparrow$ Hloubkově vyvážený

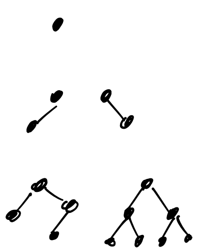
Věta: Hloubka AVL stromu s n vrcholy je $\Theta(\log n)$

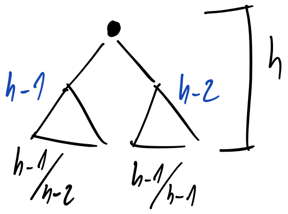
A) (Určit nejmenší vrcholů může mít AVL strom dané hloubky)

$A_0 = 1$

$A_1 = 2$

$A_2 = 4$





$$A_h = 1 + A_{h-1} + A_{h-2}$$

Indukcí podle h: $\underline{A_h \ge 2^{h/2}}$ ↗ tohle chceme

① $h = 0$, $A_0 = 1$

$h = 1$, $A_1 = 2 \ge 2^{\frac{1}{2}} = \sqrt{2} = 1{,}414$

② pro $h \ge 2$

$$A_h \ge A_{h-1} + A_{h-2} \ge 2^{h/2} \cdot \underbrace{\left(\frac{\sqrt{2}}{2} + \frac{1}{2}\right)}_{\ge 1} \ge 2^{h/2}$$

$A_{h-1} \ge 2^{\frac{h-1}{2}} = 2^{\frac{h}{2}} \cdot 2^{-\frac{1}{2}}$, $A_{h-2} \ge 2^{\frac{h-2}{2}} = 2^{\frac{h}{2}} \cdot 2^{-1}$ Podle IP

A_h roste exponenciálně

$\Rightarrow \exists c > 1 : A_h \ge c^h$

$\Rightarrow$ strom na n vrcholech má hloubku $\le \log_c n$ $\in O(\log n)$

kdyby $h > \log_c n$, pak $A_h \ge c^{>\log_c n} > n$ ✗

B) Maximální počet vrcholů podstromu hloubky h:

To je úplný strom. $B_h = 2^{h+1} - 1$ $h \ge \log_2 n+1$ $\Rightarrow$ <u>Hloubka $\in \Omega(\log n)$</u>

AVL - stromy

- Znaménko vrcholu je $\delta(v) := h(r(v)) - h(l(v)) \in \{-1, 0, 1\}$
  - pokud se vychýlí víc, začneme převažovat

Insert(x):

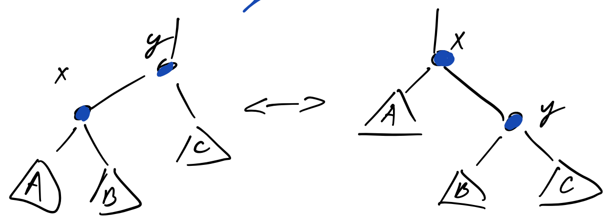
Vyvažování: Celkově:

- buď jen změním znaménko, občas pokračuji
- nebo rotace / dvojrotace

Delete(x):

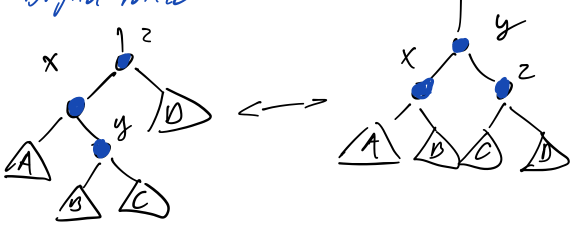
- převod na smazání (listu / vrcholu s 1 synem)

Rotace hrany:



Jednoznačně určen postup rotace

Dvojitá rotace:



- do vrcholu přijde signál „hloubka se zvýšila o 1".
- buď jen upravím informaci nebo rotuji

Vyvažování: Celkově

- buď změna znamének nebo (dvoj)rotace
- rušení pokračujeme

Věta: Find, Insert a Delete v AVL stromu mají časovou složitost $\Theta(\log n)$

Find vychází z vyváženosti binárního stromu.

Insert i Delete vyvažování se děje v konstantním čase na hladinu, tedy taky $O(\log n)$

(a, b) – stromy

- vícecestný vyhledávací strom
- a, b jsou parametry: $a \geq 2$, $b \geq 2a-1$
- všechny ext. vrcholy jsou stejně hluboko
- int. vrcholy mají a až b synů (a-1 až b-1 klíčů)
  - kořen má 2 až b synů (1 až b-1 klíčů)

Lemma: (a,b) strom s n klíči má hloubku $\Theta(\log n)$

$\Omega(\log n)$ vychází z podstaty BST

? Kolik nejméně může mít (a,b) strom klíčů ve stromu hloubky h?

∀ vrchol má pak min. synů/klíčů

#vrcholů na hladině

1

2

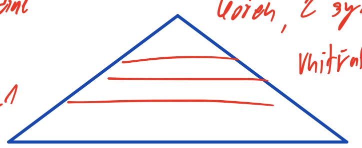
2a

$2a^2$

⋮

$2a^{i-1}$

Kořen, 2 syny



h-1

h

vnitřní vrcholy: a synů, a-1 klíčů

ext. vrcholy

$\frac{a^{h-1}-1}{a-1}$

$$= 1 + \sum_{i=1}^{h-1} 2a^{i-1}(a-1) = 1 + 2(a-1)\cdot\sum_{j=0}^{h-2} a^j$$

roste exponenciálně

$$= 1 + 2\cdot(a^{h-1}-1) = 2a^{h-1}-1$$

hloubka tedy roste logaritmicky

Kolik maximálně klíčů v (a,b) stromě hloubky h?

$\sim b^h$ => hloubka roste logaritmicky

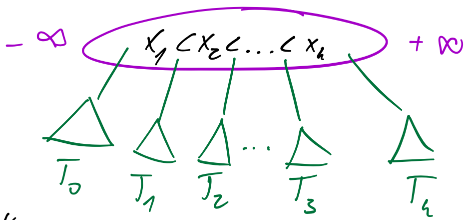
Ideální jsou (2,3) nebo (2,4) stromy, teoreticky.
V reálné ale díky blokům disku je nejlepší (256, 512)-strom

Externí vrchol – null pointer „prázdných listů"
- trochu to sjednotí myšlení nad vrcholy/stromy
- vyplňují rozsahům mezi klíči

Vícecestný vyhledávací strom

- zakořeněný strom, int + ext. vrcholy
- synové vrcholů mají pořadí
- ve vrcholech jsou klíče
  - uložený vzestupně $x_1 < x_2 < x \ldots < x_k$

#synů = #klíčů + 1



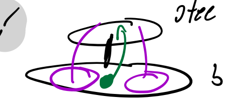
$\forall i\ \forall y$ klíč v T_i: $x_i < y < x_{i+1}$

Operace:

Find: $O(1)$ na hladinu, celkem $\Theta(\log n)$

Insert: Dělám Find, než dojdu do cíle. Pak přidám klíč do poslední vnitřní hladiny + ošetříme přetečení klíčů

Přetečení

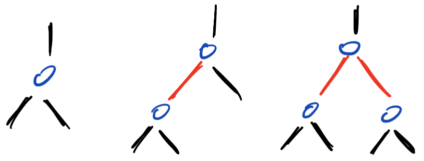


- do otce přidám klíč, případně opakuji pokud otec přetekl

Potenciální poloviny moc malé, proto $b \geq 2a-1$

$\Theta(\log n)$

LLRB strom a (2,4) strom



LLRB strom je BST s ext. vrcholy a hranami obarvenými černě a červeně t.ž.

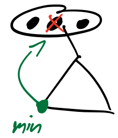
1. nejsou 2R těsně nad sebou
2. Pokud z vrcholu dolů vede 1R, pak doleva
3. hrany do listů jsou B
4. na ∀ cestě kořen–list je stejný #B hran

∃ bijekce mezi (2,4) stromy a LLRB stromy

Vychází z podskupiny těch 4 předpokladů pro LLRB

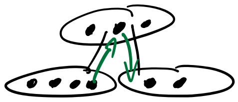
Delete: Nejprve převedeme na Delete z nejnižší vnitřní hladiny

Stačí umět řešit podtečení, tedy že má vrchol a-2 klíčů.



Najdeme sourozence (bližo levého)

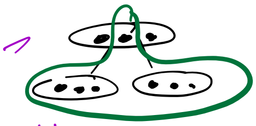
bratr má $> a-1$ klíčů



přesun klíčů

bratr má $a-1$ klíčů

slijeme dohromady (u otce 1 klíč smažeme)



$(a-2)+(a-1)+1 = 2a-2 \leq b-1$

otec mohl podtéct, pak opakujeme, když podtekl, pokud je prázdný, takže ho jen smažu

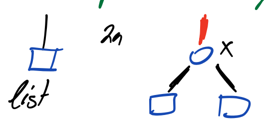
$\Theta(\log n)$

- $O(1)$ na hladinu, $\log n$ hloubka

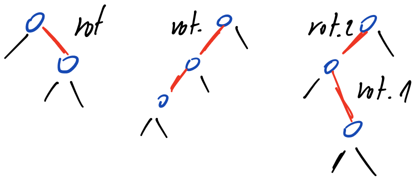
Insert: Směrem dolů přebarvujeme 4-vrcholy

↘ můžeme rozbít R axiomy

Nahradíme list za



Směrem nahoru opravujeme R axiomy rotacemi



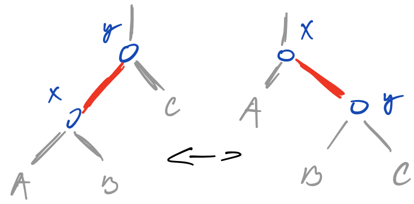
Jakmile se vrátím do kořene, mám korektní LLRB strom

↳ opět trvá $\Theta(\log n)$

- barvičku si může držet vrchol kvůli implementaci

Operace:

Rotace R hrany:



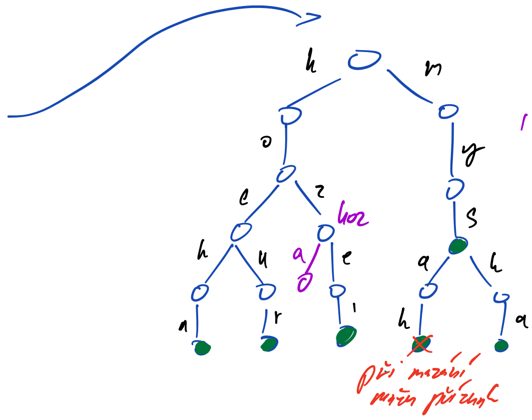
- zachovává B axiomy
- může rozbít R

Přebarvení 4-vrcholu:

- zachovává B axiomy
- může rozbít R

Trie

{kocka, kocur, kozel, mys, myska, mysak}



Vrcholy odpovídají prefixům, případně hodnota slova

Ve vrcholu pole ukazatelů indexovaných abecedou

Operace:

Member: $\Theta(|y|)$

Insert: $\Theta(|y|)$

} Jdu jen ve vzdálenosti slova

Delete:

- Smažu příznak ve vrcholu a cestou čistím mrtvé větve. $\Theta(|y|)$

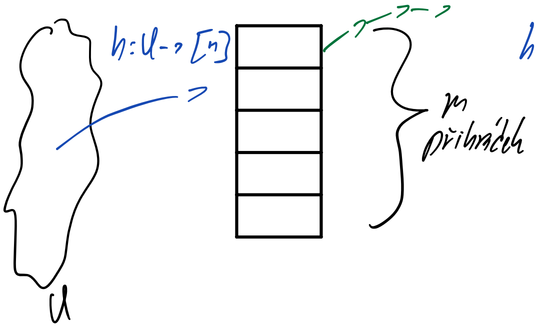
Paměť: $\Theta(\sum |x_i|)$ – musím sečíst všechny slova, když nemají stejný prefix

U velké abecedy bude každý vrchol držet BST místo pole:

Pamět. složitost: $\Theta\left(\sum_i |x_i|\right)$

Čas. složitost: $\Theta(|y| \cdot \log|\Sigma|)$ — velikost abecedy, což odpovídá BST size

Hashování



h: hashovací funkce

Kolize = více prvků v přihrádce
- v přihrádce může být seznam

Příklad:

$\{1212, 935, 1918, 1948, 1968, 1989, 2021\}$

$h(x) := x \bmod 10$

0	1	2	3	4	5	6	7	8	9
	2021	1212			935			1918, 1948, 1968	1989

Časová složitost závisí na obsazenosti jednotlivých přihrádek.

Představa: Rovnoměrné rozložení prvků v přihrádkách.
- pak Find, Insert a Delete pracují v $O(1)$

Jak volíme hashovací funkce?

- $x \to (ax) \bmod m$ — většinou prvočíslo, $a \sim 0.618m$ — lineární kongruence
- pro $U = [2^w]$, $m = 2^k$
  $x \to (ax \bmod 2^w) >> w-k$ — multiply-shift
- $x_0 \ldots x_{d-1} \to \left(\sum_i a_i x_i\right) \bmod m$ — sk. součin
- $x_0 \ldots x_{d-1} \to \left(\sum_i a^i x_i\right) \bmod m$ — polynom

Přehashování:

- sledujeme $\alpha := \frac{n}{m}$ — faktor zatížení
  - chceme $\alpha \le$ konst
- vzroste-li α příliš, přehashujeme

- zvolíme $m \to 2m$

  Přehashovat z 2^i do 2^{i+1} trvá $\Theta(n + 2^{i+1}) = \Theta(n)$

  $\frac{n}{2^i} >$ konst $\quad 2^i < \frac{n}{\text{konst}}$

  $\frac{n-1}{2^i} <$ konst $\Rightarrow 2^i > \frac{n-1}{\text{konst}} \Rightarrow 2^i \in \Theta(n)$

  Mezi přehashováním přibude mnoho operací

Nechť: $\alpha \le 1$, počty přihrádek jsou mocniny dvojky

1, 2, 4, 8, 16, 32 ...

$2^{i-1} \to 2^i \to 2^{i+1}$

$\underbrace{2^{i-1}+1 \quad 2^i+1}_{\text{přibylo } 2^{i-1} \text{ prvků}}$ trvá $\Theta(2^i) \to O(1)$ na přidaný prvek

[amortizace]

Univerzální hashování

Systém funkcí $\mathcal{H}$ z U do $[m]$ je c-univerzální pro $c > 0 \equiv$

$$\forall x,y \in U,\ x \neq y: \underset{h \in \mathcal{H}}{P}[h(x) = h(y)] \leq \frac{c}{m}$$

ideální funkce má $\frac{1}{m}$

Lemma: Nechť $\mathcal{H}$ je c-univerzální systém funkcí z U do $[m]$, $x_1 \dots x_n \in U$ navzájem různé, $y \in U$

Potom $\underset{h \in \mathcal{H}}{\mathbb{E}}\big[\overbrace{\#i : h(x_i) = h(y)}^{Y}\big] \leq c \cdot \frac{n}{m} + 1$

$+1$: y = jedno mu z x_i

Nechť $\forall i: x_i \neq y$

Zavedeme indikátory $I_1 \dots I_n$:

$$I_i = \begin{cases} 0 \\ 1 & \text{pro } h(x_i) = h(y) \end{cases}$$

- $Y = \sum_i I_i$
- $\mathbb{E}[Y] = \sum_i \mathbb{E}[I_i]$
- $\mathbb{E}[I_i] = P[h(x_i) = h(y)] \leq \frac{c}{m}$ (díky c-univerzalitě)

$\hookrightarrow \mathbb{E}[Y] \leq \frac{c}{m} \cdot n = c \cdot \frac{n}{m}$ $\boxed{\times}$

Důsledek: $\mathbb{E}$ časové složitosti Findu, Insertu, Deletu je $O(\frac{n}{m})$ → To umíme udržet $O(1)$ přehashováním

Příklad: Skalární součin nad tělesem $\mathbb{Z}_p$:

$U = \mathbb{Z}_p^d$, $a \in \mathbb{Z}_p^d$, přihrádky: $\mathbb{Z}_p$

$\mathcal{H} = \{h_a \mid a \in \mathbb{Z}_p^d\}$

$h_a(x) = a \cdot x$ (v tělese)

$(\sum_i a_i x_i) \bmod p$

Věta: Tento systém je 1-univerzální.

Pro $x \neq y$: $\underset{h \in \mathcal{H}}{P}[h(x) = h(y)] =$

$= \underset{a}{P}[ax = ay] \Rightarrow a \cdot (x - y) = 0$

BÚNO $x_1 \neq y_1$

$$\sum_{i=1}^{d} a_i \cdot (x_i - y_i) = 0$$

$$\underbrace{a_1}_{\text{neznámá}}\underbrace{(x_1 - y_1)}_{\text{nenulová konst}} + \underbrace{\sum_{i=2}^{d} a_i (x_i - y_i)}_{\text{konst}} = 0 \quad \Big\}\ \text{lin rovnice}$$

$\Downarrow$

$\exists!$ řešení pro a_1

$P[a_1 \text{ je řešení}] = \frac{1}{p}$ $\boxed{\times}$

Otevřená adresace

- při kolizi nezapojuji seznamy, ale hledám první volnou jinou přihrádku.

Každému $x \in U$ přiřadíme vyhledávací posloupnost $h(x,0), h(x,1), \dots, h(x, m-1)$

- což je permutace na přihrádkách → permutace

Příklady: **Lineární přidávání:**

- první prvek vyhl. posloupnosti vybrán pomocí hash. funkce, zbytek lineární procházka přihrádkami

Problém: čím větší souvislý úsek, tím delší ho udělám, takže se mi to hrupí do jedné nudle.

Operace:

Insert: Postupně procházím přihrádky, než najdu volné místo podle vyhledávací posl.

Find: Procházím v pořadí posloupnosti, končím nalezením či prázdnou přihrádkou

Delete: Místo prvku umístíme příznak – Insert příznak přepíše, Find nechá, takže časem musíme přehashovat dvojité hashování:

prvočíslo

$$h(x,i) = \big(f(x) + i\,g(x)\big) \bmod m$$

dvě hash. funkce

To dokáže rozhodit prvky do tabulky a každý prvek má vlastní skoky, takže se nebude tolik tvořit jeden velký modlk.

Věta: Pokud jsou vyhl. posloupnosti nezávislé plně náhodné permutace, pak

$$E[\#\text{přihrádek neúspěšného findu}] \le \frac{1}{1-\alpha}, \quad \alpha = \frac{n}{m}$$

$P_1 - \cancel{P_2} + \cancel{2P_2} - \cancel{2P_3} + \cancel{3P_3} - \cancel{3P_4} + \cancel{4P_4} - 4P_5 \dots$

$P_1 + P_2 + P_3 + P_4 \dots$

Nechť $y \in U$, $h_1 - h_m$ vyhl. posloupnost.

$P_i := P[\text{projdeme alespoň } i \text{ přihrádek}]$ $\le \frac{n}{m}$ $\le \frac{n}{m}$ $\le \frac{n}{m}$

$P_1 = 1 \quad P_2 = \frac{n}{m} = \alpha \quad P_i = 1 \cdot \frac{n}{m} \cdot \frac{n-1}{m-1} \cdot \frac{n-2}{m-2} \cdots \frac{n-(i-1)}{m-(i-1)}$

pravděpodobnost, že první je obsazená

$P_i \le \alpha^{i-1}$

$$E[\#\text{navštívených přihr.}] = \sum_{i \ge 1} i \cdot \underbrace{P[\text{navštíveno právě } i \text{ přihr.}]}_{P_i - P_{i+1}} =$$

$$= \sum_{j \ge 1} P_j \underbrace{(j - (j-1))}_{1} \le \sum_{j \ge 1} \alpha^{j-1} = \sum_{j \ge 0} \alpha^j = \frac{1}{1-\alpha}$$

Rozděl a panuj

MergeSort → $\mathrm{Merge}(x_1 - x_a,\ y_1 - y_b)$ v čase $\Theta(a+b)$

– slévání seřazených

Pro vstup $x_1 - x_n$:

Pokud $n \le 1$: Return vstup

$a_1 - a_{\lfloor n/2 \rfloor} \leftarrow \mathrm{MergeSort}(x_1 - x_{\lfloor n/2 \rfloor})$

$b_1 - b_{\lceil n/2 \rceil} \leftarrow \mathrm{MergeSort}(x_{\lfloor n/2 \rfloor + 1} - x_n)$

Vrátíme $\mathrm{Merge}(a_1 \dots, b_1 \dots)$

– rekurze je konečná

Pomocí rozepsání

Analýza složitosti

$T(n) = 2T(n/2) + n$

$T(1) = 1$

$T(n) = 2T(n/2) + n$

$2T(n/4) + n/2$

$T(n) = 4T(n/4) + 2n$

$2T(n/8) + n/4$

$T(n) = 8T(n/8) + 3n$

$\vdots$

$n/2^i = 1$

$\log n = i$

$T(n) = 2^i T(n/2^i) + in$

$T(n) = n \cdot T(1) + \log n \cdot n \in \Theta(n \log n)$

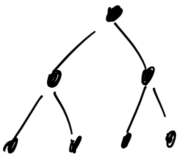
Paměť?

$M(n) = M(n/2) + n$

$M(n) = n + \frac{n}{2} + \frac{n}{4} + \dots$

$M(n) = 2n \in \Theta(n)$

Analýza pomocí stromu rekurze



$\log n$

# problémů	velikost problému	čas na hladinu
1	n	n
2	$n/2$	n
4	$n/4$	n
$\vdots$		$\vdots$
2^i	$n/2^i$	n
n	1	n

Celkem $n \cdot \log n$

Násobení dlouhých čísel

	$n/2$	$n/2$
X	A	B
Y	C	D

$X = A \cdot 10^{n/2} + B$

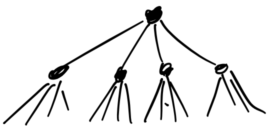
$Y = C \cdot 10^{n/2} + D$

$$XY = \underbrace{AC}_{O(n)} \cdot 10^n + \underbrace{(AD + BC)}_{O(n)\ \ O(n)} \cdot 10^{n/2} + \underbrace{BD}_{O(n)}$$

– 4 součiny $n/2$-ciferných čísel

$$T(n) = 4T(n/2) + n$$

strom rekurze:



$\log n$

#pp	velikost
1	n
4	$n/2$
$\vdots$	$\vdots$
4^i	$n/2^i$
$4^{\log_2 n}$	1

$\hookrightarrow (2^{\log_2 n})^2 = n^2$ 4

není to rychlejší, listů je furt moc

AB

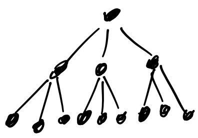
BD

Takže mám pouze tři rekurze!

$(A+B)\cdot(C+D) = AC + AD + BC + BD$

$(A+B)\cdot(C+D) - AB - BD = AD + BC$

$$T(n) = 3T(n/2) + n$$



$\log_2 n$

#pp	velikost	čas na pp	čas na hladinu
1	n	n	$1 \cdot n$
3	$n/2^i$	$n/2$	$3 \cdot n/2$
$\vdots$	$\vdots$		$\vdots$
3^i	$n/2^i$	$n/2^i$	$3^i \cdot n/2^i$
			Σ

$$T(n) = \sum_{i=0}^{\log_2 n} 3^i \cdot \frac{n}{2^i} = n \cdot \sum_{i=0}^{\log_2 n} \left(\frac{3}{2}\right)^i = n \cdot \frac{q^{\cancel{h+1}} \cancel{-1}}{\cancel{q-1}} = \Theta(n \cdot q^h) =$$

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

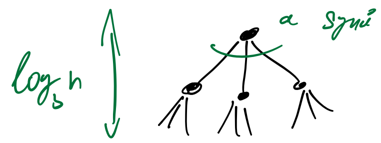
$$= \Theta\left(n^{\log_2 3}\right) \quad \log_2 3 \doteq 1{,}59$$

$$\Rightarrow T(n) = \Theta\left(n^{\log_2 3}\right)$$

Master Theorem

Věta: Rekurence $T(n) = \overbrace{a}^{\#pp} \cdot T(n/\overbrace{b}^{\text{faktor zmenšení}}) + \overbrace{\Theta(n^c)}^{\text{lokální práce}}$, $T(1)=1$ pro $a \geq 1$, $b > 1$, $c \geq 0$ má řešení:

$$T(n) = \begin{cases} \Theta(n^c \log n) & \left(\frac{a}{b^c}\right) = 1 \\ \Theta(n^c) & < 1 \\ \Theta(n^{\log_b a}) & > 1 \end{cases}$$



$n = b^k$

Na i-té hladině:

a^i podproblémů

$(n/b^i)^c$ čas na podproblém

$a^i \cdot \left(\frac{n}{b^i}\right)^c$ čas na hladinu

$$T(n) = \sum_{i=0}^{\log_b n} a^i \left(\frac{n}{b^i}\right)^c = n^c \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i$$

Pro $\left(\frac{a}{b^c}\right) = 1 \Rightarrow T(n) = n^c \cdot (\log_b n + 1) = \Theta(n^c \cdot \log n)$

$\frac{\log n}{\log b} \sim \log n$

$q < 1$: $\Sigma = \frac{1}{1-q} = \Theta(1)$

Pro $\left(\frac{a}{b^c}\right) < 1 \Rightarrow T(n) = n^c \cdot \sum_{i=0}^{\log_b n} q^i = \Theta(n^c)$

Pro $\left(\frac{a}{b^c}\right) > 1 \Rightarrow T(n) = n^c \cdot \sum_{i=0}^{\log_b n} q^i$ → $q > 1$: $\Sigma = \frac{q^{k+1}-1}{q-1} = \Theta\left(q^{\log_b n}\right) = \left(\frac{a}{b^c}\right)^{\log_b n} = \frac{a^{\log_b n}}{\left(b^{\log_b n}\right)^c} = \frac{a^{\log_b n}}{n^c}$

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

Zbývá pojednat, kdy n není mocnina b.

n^+ … nejbližší vyšší mocnina

n^- … nejbližší nižší mocnina

$n^- \leq n \leq n^+$ $\quad n^- \sim n^+$

$\left(\frac{n}{b}\right)^- \leq \left\lceil \frac{n}{b} \right\rceil \leq \left(\frac{n}{b}\right)^+$

$T(n^-) \leq T(n) \leq T(n^+)$

řešení se asymptoticky neliší, protože n^- je asymptoticky rovno n^+.

Násobení matic - Strassenův alg.

Buno $n = 2^k$

$$\underbrace{\begin{bmatrix} A & B \\ C & D \end{bmatrix}}_{X} \cdot \underbrace{\begin{bmatrix} P & Q \\ R & S \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} I & J \\ K & L \end{bmatrix}}_{Z}$$

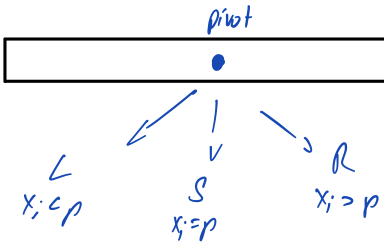
$I = AP + BR$, tedy 8 součinů matic

$T(n) = 8T(n/2) + \Theta(n^2)$ => kuchařka: $a=8, b=2, c=2$ => $\Theta(n^{\log_2 8}) = n^3$

Strassenův trik: Stačí pouze 7 násobení:

$T(n) = 7T(n/2) + \Theta(n^2)$ => kuchařka: $a=7, b=2, c=2$ => $\Theta(n^{\log_2 7}) \doteq n^{2,807}$

Quick Select



Kdybych setřídil: | L | S | R |

rekurz. alg

- Pokud $k \le |L|$
  - k-tý nejmenší v L
- Pokud $|L| < k \le |L| + |S|$
  - k je pivot
- Jinak
  - $(k - |L| - |S|)$-tý nejmenší v R

Nejlepší případ: p = medián

$|L|, |R| \le n/2$

$$T(n) = T(n/2) + \Theta(n) \qquad a = 1,\ b = 2,\ c = 1 \;\Big\}\; \gamma < 1 \quad T(n) = \Theta(n)$$

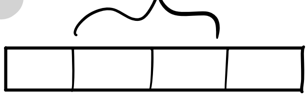
Nejhorší případ: $p = \min$, $k = n$

$|L| = 0$, $|S| = 1$, $|R| = n - 1$

$$T(n) = n + (n-1) + (n-2) + \dots + 1 = \Theta(n^2)$$

Skoromedián:

tady je



$|L|, |R| \le \frac{3}{4} n$

$$T(n) = T(\tfrac{3}{4} n) + \Theta(n)$$

$$= n + \tfrac{3}{4} n + (\tfrac{3}{4})^2 n + \dots = \Theta(n)$$

Lemma: Pokud $P[\text{pokus}] = p$, pak $\mathbb{E}[\#\text{pokusů do úspěchu}] = \frac{1}{p}$

$$\mathbb{E} = \sum_n n \cdot P\underbrace{[\#\text{pokusů} = n]}_{(1-p)^{n-1} \cdot p} =$$

Randomizované hledání skoromediánu:

1) Vyberu náhodné p
2) Spočítám $x_i < p$, $x_i > p$ $\Big\}\ \Theta(n)$
3) Pokud p není skoromedián, zahodím a znovu

Věta. $\mathbb{E}[\text{čas. složitost hledání skoromediánu}] = \Theta(n)$

☼ $P[\text{najdu skoromedián}] \ge \frac{1}{2}$

Pak $\mathbb{E}[\#\text{pokusů}] \le 2$

Volím pivota náhodně

Rozdělíme běh na fáze.
Fáze končí výběrem skoromediánu

☼ $P[\text{trefil jsem se}] \ge \frac{1}{2}$

$\mathbb{E}[\#\text{pokusů}] \le 2$ $\Big\}\ \mathbb{E}[\text{čas na fázi}]\ \Theta(n)$

☼ Každá fáze zmenší n alespoň $\frac{3}{4}$-krát

$\sum$ přes fáze

$$\Theta(n + \tfrac{3}{4} n + (\tfrac{3}{4})^2 n + \dots) = \Theta(n)$$

k-tý nejmenší prvek lineárně

1) Vyberu pivota z $x_1 - x_n$
2) Rozdělíme vstup na L, S, R
3) Podle pivota se zarekurzíme do některé z částí

součástí výběru

Select $(x_1 - x_n, k)$:

1) Rozdělíme $x_1 - x_n$ na pětice $P_1 - P_t$
2) Najdeme mediány 5-tic
3) Najdeme medián mediánů pětic:

$\text{Select}(m_1 - m_t, \lceil t/2 \rceil)$

– toto bude pivot pro hledání k-tého nejmenšího

QuickSort

L	S	R

1) Zvolíme pivota p
2) Rozdělíme podle pivota na L, S, R
3) L = QuickSort(L)
   R = QuickSort(R)
4) Vrátíme $L \| S \| R$

Složitost:

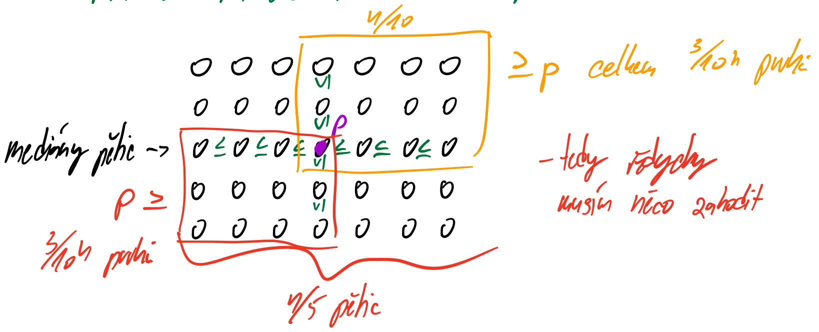
Pokud p medián:

$$\to T(n) = 2T(n/2) + \Theta(n)$$

$$T(n) = \Theta(n \cdot \log n)$$

Pokud p min/max: $T(n) = T(n-1) + T(0) + \Theta(n) = \Theta(n^2)$

Věta: Select má časovou složitost $\Theta(n)$



Vždy zahodím alespoň $3/10\,n$ prvků, tedy zbývá jen $7/10$.

$$T(n) = T(n/5) + T(7/10\,n) + n$$

$$T(1) = \Theta(1)$$

Uhodneme: $T(n) = cn$

$$cn = \underbrace{\frac{1}{5}cn + \frac{7}{10}cn}_{9/10\,n} + n$$

$$\frac{1}{10}cn = n \implies c = 10$$

$\boxtimes$

Tedy jsme dokázali, že výběr lineárně

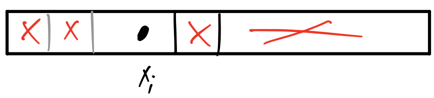
Věta: QuickSort s náhodnou volbou pivota má časovou složitost se střední hodnotou $\Theta(n \cdot \log n)$

Dk #1: Porovnání naúčtujeme prvku, který nebyl pivotem, napočítáme kolikrát jsme ho porovnali $= P_i$

$\#\text{porovnání} = \sum P_i$ čas $= \Theta(\#\text{porovnání})$

$E[\#\text{porovnání}] = \sum E[P_i]$

sledujeme velikost P_p s x_i:



Fáze končí volbou p = pseudomedián
– fáze zmenší n na $\frac{3}{4}n$
– počet fází $= O(\log n)$

$E[\text{krokŭ ve fázi}] \le 2$

Takže jeden prvek porovnán nejvíc $\log n$ krát, tedy celkem je porovnání $\Theta(n \log n)$

Dynamické programování – nejdelší rost. podposl.

NRP(i): → délka NRP vybrané z $x_i … x_{n+1}$

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

① exp. složitost → existuje 2^n podposloupností

② existuje pouze $i \in \{0 … n+1\}$ argumentů
→ volání se opakují

③ Přidáme cache
→ fce je zavolána $O(n)$ krát
- pokaždé počítá v $O(n)$
} celkem $O(n^2)$

④ Vyplňujeme tabulku cyklem
↳ běží v $O(n^2)$, obsahuje jen 2 vnořené cykly

NRP(n):
1. $P[n+1] = 1$
2. Pro $i = n, …, 0$ pozpátku:
3. $d = 1$
4. Pro $j = i+1 … n+1$:
5. Pokud $x_i < x_j$:
6. $d = \max(P[j]+1, d)$
7. $P[i] = d$
8. Return $P[0]$

Editační vzdálenost

Editační operace: Změna, Vložení, Smazání

Editační vzdálenost řetězců α, β := min. posloupnost edit. operací, aby z α udělaly β.

Edit(i,j): → spočítat $L(\alpha_i … \alpha_n, \beta_j … \beta_m)$

1) Pokud $i > n$: Vrátíme $m-j+1$
   Pokud $j > m$: Vrátíme $n-i+1$
2) $l_v := 1 + Edit(i, j+1)$
3) $l_s := 1 + Edit(i+1, j)$
4) $l_z := Edit(i+1, j+1)$
5) Pokud $\alpha_i \neq \beta_j$: $l_z = l_z + 1$
6) Vrátíme $\min(l_s, l_v, l_z)$

Lze prováděť zleva doprava:

$\alpha = \alpha_1 … \alpha_n$, $\beta = \beta_1 … \beta_m$

$L(\alpha, \beta)$:
- smaže α_1 — $L(\alpha_2 … \alpha_n, \beta)$
- změní α_1 na β_1 — $L(\alpha_2 … \alpha_n, \beta_2 … \beta_m)$
- vloží β_1 před α_1 — $L(\alpha, \beta_2 … \beta_m)$
- ponechá α_1 a β_1 — $L(\alpha_2 … \alpha_n, \beta_2 … \beta_m)$

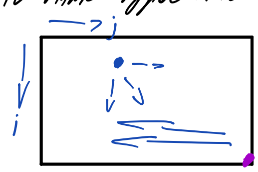
zkusíme všechny, vybereme min.

↳ Je to hrozně pomalé! $O(2^n)$

Opět má omezený počet argumentů: $i \in \{1 … n\}$, $j \in \{1 … m\}$

Takže celkem $O(n \cdot m)$ → takže zavedeme hashovací tabulku

Tabulku vyplňujeme zprava odspodu, protože už to máme vypočítané předem



Nerekurzivně:

1) Pro $j=1,\dots,m+1$: $T[n+1,j] = m-j+1$
Pro $i=1,\dots,n$: $T[i,m+1] = n-i+1$ } Operace, pokud jsem přetekl, ty jsou jasně dané

2) Pro $i=n,\dots,1$:
3) Pro $j=m,\dots,1$:
4) $\delta = 0$, pokud $\alpha_i = \beta_j$, jinak 1
5) $T[i,j] = \min(1+T[i+1,j],\ 1+T[i,j+1],\ \delta+T[i+1,j+1])$

Časová i prostorová složitost $\Theta(n \cdot m)$

Optimální BST

Dány klíče $x_1 < \dots < x_n$, váhy $w_1 \dots w_n \in \mathbb{N}$

Pro BST T na $x_1 \dots x_n$: hloubky $h_1 \dots h_n$

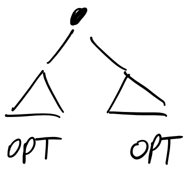
$h_i :=$ # vrcholů na cestě do x_i

$\rightarrow C(T) := \sum_i h_i \cdot w_i$

Cíl: Najít T s nejmenší cenou

→ když bychom věděli opt. kořen

$OPT(x_1 \dots x_n) = OPT(x_1 \dots x_{i-1}) + OPT(x_{i+1} \dots x_n) + \sum_{j=1}^{n} w_j$ → protože se zvětšila hloubka, musím přičíst ty váhy pro všechny vrcholy



– my ale kořen neznáme, tak zkusíme $x_1 \dots x_n$ a najdeme min. cenu

$OptBST(\ell, p)$ → opt. cena BST pro $x_\ell \dots x_p$

1) Pokud $l > p$: Return 0
2) $\min(C_\ell \dots C_p) + \sum_{i=\ell}^{p} w_i$
kde $C_i := OptBST(\ell, i-1) + OptBST(i+1, p)$

Je to pomalé

$\ell, p \in \{1 \dots n\} \rightarrow O(n^2)$ pp

Pomocí cache počítáme $O(n^2)$ pp, každý v čase $O(n)$ } celkem $O(n^3)$

Nahradíme rekurzi cyklem... od nejkratších po nejdelší

1) Pro $\ell = 1 \dots n$: $T[\ell, \ell-1] = 0$
2) Pro $d = 1 \dots n$: d = délka intervalu
3) Pro $\ell = 1 \dots n-d+1$: ℓ = levý okraj
4) $p = \ell + d - 1$ p = pravý okraj
5) $T[\ell, p] = \min(C_\ell \dots C_p) + \sum_{i=\ell}^{p} w_i$
6) Return $T[1,n]$

$C_i := T[\ell, i-1] + T[i+1, p]$

Čas $O(n^3)$, prostor $O(n^2)$

→ Máme cenu, pro strom si budeme u vrcholu pamatovat optimální kořen pro daný interval

Dolní odhady

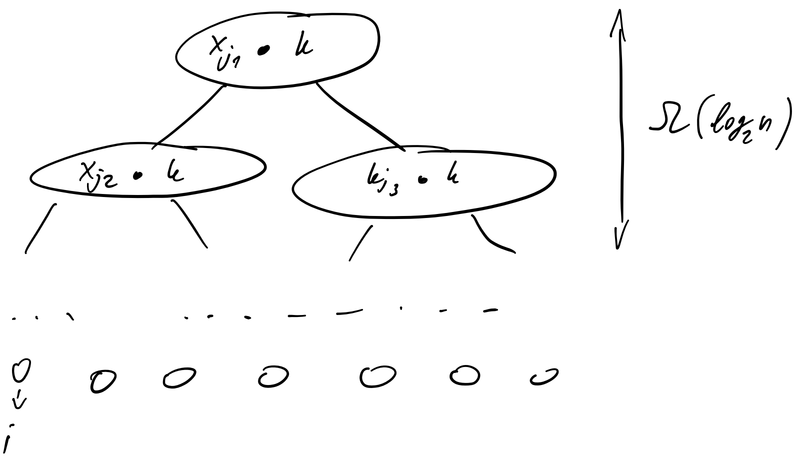
BST

Věta: Každý deterministický alg. pro vyhledávání v porovnávacím modelu použije v nejhorším případě $\Omega(\log n)$ porovnání

Spustíme alg. na vstupu $1..n$ pro $k=1..n$

V listu vrátíme výsledek

$\#$ listů $\geq n$ → pro různé k různý výsledek



Max. $\#$ listů v BST hloubky $h = 2^h$

hloubka listu $= \#$ porovnání $= O(\log n)$

Třídění:

vstup: permutace $\{1..n\}$, $\#$ vstupů $= n!$

Potřebujeme o vstupu určit $\geq \log_2 n!$ bitů informací

Lemma: $\log_2 n! \geq \Omega(n \cdot \log n)$

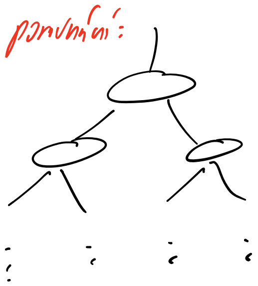
$n! \geq \left(\frac{n}{2}\right)^{\frac{n}{2}}$

$n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot 1$
– prvních $n/2$ prvků je alespoň $n/2$

$\log_2 n! \geq \frac{n}{2} \log_2 \frac{n}{2} = \frac{n}{2} \cdot \log n - 1.$

Věta: Každý deterministický algoritmus pro třídění v porovnávacím modelu použije v nejhorším případě $\Omega(n \log n)$ porovnání.

Rozhodovací strom porovnání:



$\#$ listů $\geq n!$

↳ pro dvě různé permutace skončí výpočet v různých listech

Protože alg. lze upravit, aby vracel pro každý prvek původní index ve vstupní posl.

Pak hloubka $= \Omega(\log n!) = \Omega(n \log n)$

$\Rightarrow$ existuje list, který odpovídá vstupní permutaci v hloubce $n \log n$