Obecně

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

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

DFS + BFS

DFS stavy:

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

DFS:

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

→ inicializace je na začátku se všemi vrcholy jako „nenalezen"

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

Lemma: DFS se zastaví v čase $O(n+m)$

Stavy: N -> O -> Z
=> Vrchol otevřen != 1 => DFS na vrchol vůbec max 1.

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

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, tedy. $\exists$ hrana uv z otevřeného u. IP => u je dosažitelné z v_0

<= Existují dosažitelní, nenalezení. Zvolme v špatný a nejbližší k v_0.

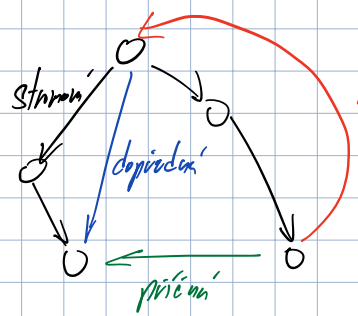
p := předposlední vrchol nejkratší cesty z v_0 do v.

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

Definice hran

- zpětná
- dopředná
- příčná
- stromová

tyto existují i v grafu neorientovaném



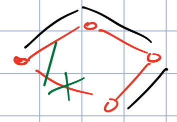
Typ hrany zjistíme v O(1) – při procházení grafu

Věta: DFS v čase $\Theta(n+m)$ a prostoru $\Theta(n+m)$ 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 $\Leftrightarrow$ e leží na kružnici.

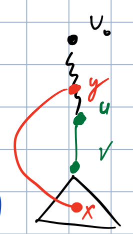
- Zpětná hrana nikdy nebude most



Pozorování: uv leží na kružnici

$\Updownarrow$

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



$\Leftrightarrow$ $low(v) < in(v)$

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

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

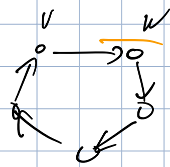
Věta: Alg. najde všechny mosty v čase a prostoru $\Theta(m+n)$

Acyklické orientované grafy DAGy

Existuje dosažitelný zpětný cyklus $\Leftrightarrow$ DFS najde zpětnou hranu.

$\Leftarrow$ triviální

$\Rightarrow$ zvolme $v \in C$ s min. $out(v)$
- jediný, kde roste out, je na zpětné



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

- 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 $\Leftrightarrow$ je to DAG

$\Rightarrow$ triviální, protože pak není cyklus na grafu

$\Leftarrow$ Postupně budu hledat a odtrhávat zdroje, získám DAG

Lemma: Každý DAG má zdroj
- půjdu odzadu (jako bych šel od listů do kořene)

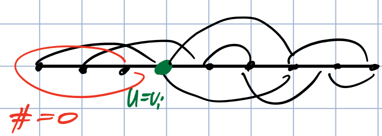
Pořadí, v němž DFS opouští vrcholy, je opačné topologické.

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 $j > i$ indukcí díky tomu, že to jsou předchůdci

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

Silná souvislost

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

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

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

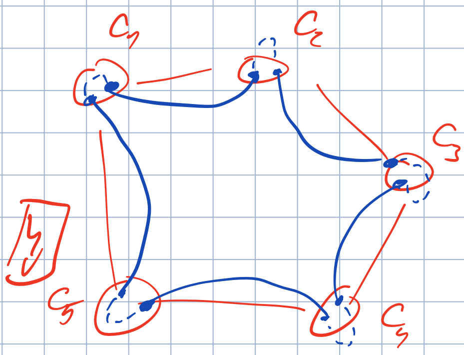
G je silně souvislý $\Leftrightarrow$ # komp. sil. souv. $= 1$

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 opakovaný 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 $\rightsquigarrow$

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

Algoritmus

→ ? je to ale pomalé.

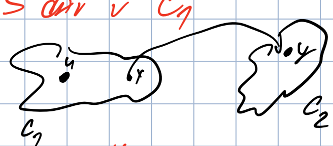
Najdeme zdrojový vrchol z G^T, takže mám stokový vrchol z G. Na něj pustím DFS, který najde stokovou komponentu, takže takhle najdu všechny komponenty.

Zrychlení:

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

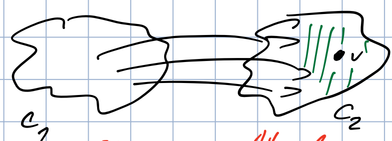
→ Pokud C_1, C_2 jsou komponenty, pak platí, $\max_{u \in C_1} out(u) > \max_{v \in C_2} out(v)$

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) Opakovaný 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!

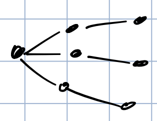
Algoritmus najde komp. souvislosti v čase $\Theta(n+m)$.

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:



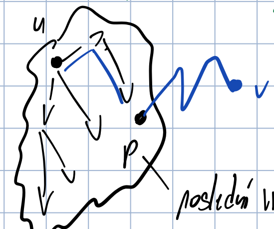
kompaktní reprezentace vzdáleností:
- strom na V
- podgraf G
- orientovaný od kořene u
- $\forall v \in V$: cesta z u do v je jedna z nejkratších cest mezi u–v

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

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

– Opět pomocí vystřižení cyklů, pokud nejsou záporné
– 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 $\min h(v)$ — $O(n)$
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

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

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

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

→ pomalí pro husté grafy

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

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

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

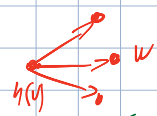
otevřený ⟺ od poslední relaxace se změnilo hodnocení daného vrcholu.

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

v je dosažitelný z u
⇕ → korektnost DFS/BFS
v je uzavřený
⇕ 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) Triviální z definice relaxace
2) inic.: OK
relaxace:



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

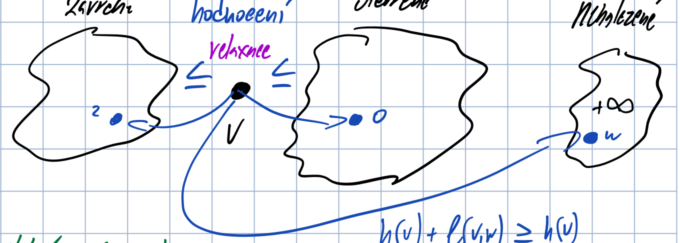
Lemma V: Pokud se alg. zastaví, pak $\forall v \in V$: $h(v) = d(u,v)$

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 mám $h(v) \leq h(p) + \ell(p,v)$ (díky relaxaci p – vyplývá z předpokladu relaxace), kde $h(p) + \ell(p,v) = d(u,v)$ ↯

Pro Dijkstru 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 nastavení $h(w) \geq h(v)$
– do zavřených nelze nic měnit hodnotu



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

– Důkazem nechť výše zmíněné invarianty.

Bellman-Fordův alg.

- relaxační algoritmus s otevřenými vrcholy ve frontě
- relaxace popuje z fronty
  ↳ zavírá nejstarší z otevřených vrcholů

Df: Fáze výpočtu:

F_0 := otevření u

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

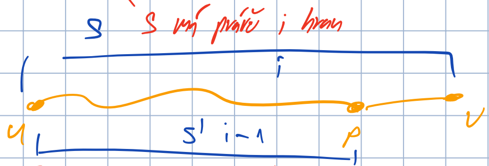
Věta: Bellman-Ford najde nejkratší vzdálenost v čase $O(n \cdot m)$ pro graf bez záporných cyklů.

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

fází $\leq n$

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. – 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}, tehdy p otevřeno, nejpozději v F_i zavřeno, tehdy 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

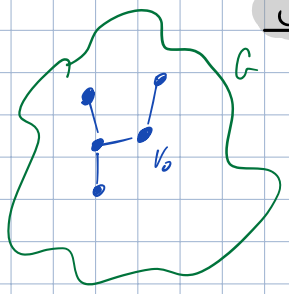
Matice $D^{|V| \times |V|}$, kde jsou doplňkové vzdálenosti jednotlivých hran, na diagonále je nula, jinde $+\infty$

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 $w: E \to \mathbb{R}$
- 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 podstaty přidání listu

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

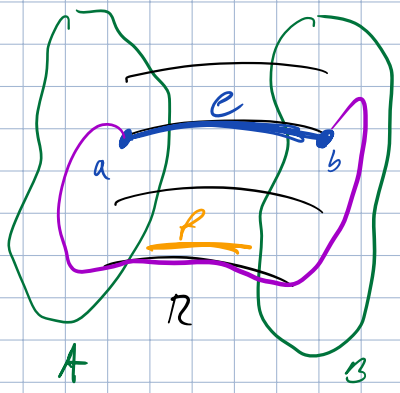
Min. kostra je jednoznačně určena počtem hran podle vah.

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

Složitost: $O(m \cdot n)$

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.

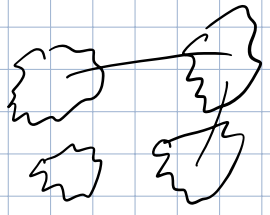
Když smažu f, rozbiju souvislost kostry, ale přidáním e ji opět opravím.

$\check{T} = T - f + e \quad w(\check{T}) = w(T) - w(f) + w(e)$

< 0

$\to w(\check{T}) < w(T)$ ↯

Borůvkův algoritmus



Postupně hledají malé stromečky, v stromečcích vybere minimální hranu a tyto hrany přidáme. Dostaňme, pokud už je jen jeden stromeček

Lemma: # fází $\leq \log n$

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

$k=0 \to 2^0 = 1$ ✓

$k+1$: v stromečcích na konci fáze vzniklé sjednocením 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. řezy,
tedy přidaná hrana je nejlehčí z el. řezu => je součástí min. kostry

Kruskalův alg + Union-Find problem

Kruskalův alg.

- setřídíme hrany od nejlehčí po nejtěžší
- postupně je od nejlehčí přidáváním, pokud vytvoří cyklus, vynecháním ji

$(m \cdot Find + n \cdot Union)$

Lemma: Nejde minimální kostru
Rozhodně vytvoří kostru
Minimální díky úvodnímu lemmatu.
- díky pořadí přidávání přidám jen ty nejlehčí z el. řezu.

Union-Find

udržujeme komp. souvislosti

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

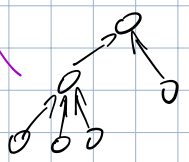
Union(u,v) ... přidá hranu uv

Implementace:

Pole: vrchol → číslo komponenty

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

Komponenty reprezentujeme jako lesíky.



Strom orientovaný ke kořeni
jeho vrcholy tvoří vrcholy komponenty

vrcholy si pamatují otce
↓
1 pole

Na začátku mám samostatné vrcholy, které postupně spojuju dohromady

Find(u,v):

do u,v do kořenů lesíků,
kořeny porovnáváme

Složitost: O(hloubka lesíků)

Union(u,v):

Najdeme kořeny u', v'
Pokud u' = v' : hotovo
Jinak přidáme hranu mezi kořeny

Složitost: O(hloubka lesíků)

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

Lemma: Lesík hloubky h má alespoň 2^h vrcholů

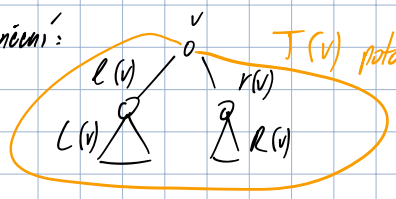
<-> hloubky jsou $\leq \log n$

Důsl. Union a Find trvají $O(\log n)$

Kruskalův alg: $O(m \log n + m \log n + n \log n) = O(m \log n)$
sort, finds, unions

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:

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

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

⇒ všechny klíče jsou různé

Show: (Enumerate) Find: Insert:

$\Theta(n)$ $\Theta(\text{hloubka})$ $\Theta(\text{hloubka})$

- přípojují na místo None

Delete:

1) Pokud nemá syny, rovnou maž
2) Má jednoho syna, dosaď tam syna
3) Má oba syny, nahraď ho vrchol jiným listem (např: vpravo dole)

$\Theta(\text{hloubka})$

Dokonale vyvážený BST

Strom je dokonale vyvážený ≡

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

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

- každým vrcholem snížím počet prvků pod sebou o polovinu

Tvorba ze setříděné posloupnosti:

$x_1 < x_2 \dots < x_n$

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

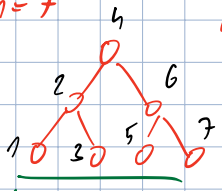
- rekurzivně brm prostředky posloupností

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

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

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

n = 7

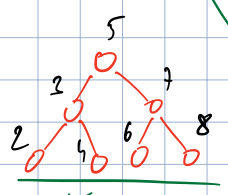


liché v listech

Provedu Insert (n+1)
Delete (1)

Poté 2.. n+1 zase jednoznačně

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



sudé v listech

$\Omega(n)$ vrcholů změnilo, zdali jsou listy

v každém změnu 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

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

Dokonale vyvážený ⇒ Hloubkově vyvážený

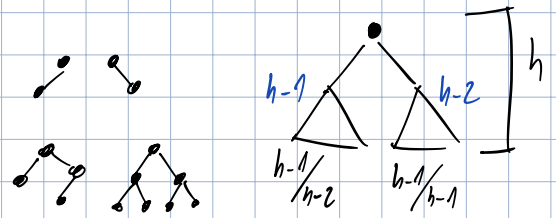
Věta: Hloubka AVL stromu s n vrcholy je $\Theta(\log n)$

A) (Alespoň nějakých 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: $A_h \geq 2^{h/2}$ ← takhle chceme

① $h=0$, $A_0 = 1$

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

② pro $h \geq 2$

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

(≥ 1)

$A_h \geq 2^{\frac{h-1}{2}}$, $2^{\frac{h-2}{2}} = 2^{\frac{h}{2}} \cdot 2^{-1}$

$2^{\frac{h}{2}} \cdot 2^{-\frac{1}{2}}$

Podle IP

A_h roste exponenciálně.

⇒ $\exists c > 1 : A_h \geq c^h$

⇒ Strom na h vrcholech má hloubku $\leq \log_c n$ ($\in O(\log n)$)

Kdyby $h > \log_c n$, pak $A_h \geq c^{\log_c n} > n$ ✗

B) Maximální počet vrcholů podstromu hloubky h:

To je úplný strom. $B_h = 2^{h+1} - 1$ $\quad h \geq \log_2 n + 1$ ⇒ Hloubka $\in \Omega(\log n)$

AVL - stromy

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

Insert(x):

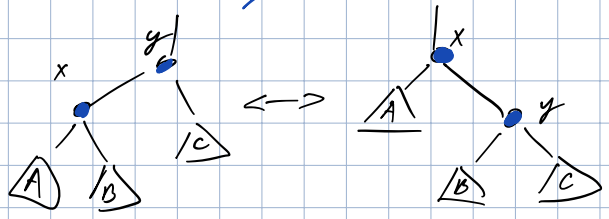
Vyvažování: Celkově:

- buď jen změním znaménka, občas pohraničuji
- nebo rotace / dvojrotace

Delete(x):

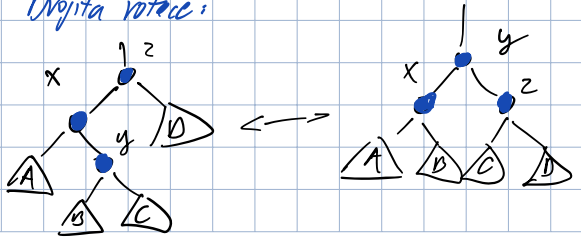
- převod na stromání (listu / vrcholu s 1 synem)

Rotace hrany:



Jednoznačně určen postup rotace

Dvojitá rotace:



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

Vyvažování: Celkově

- buď změním znaménka nebo (dvoj)rotace
- vrcholy pohraničníme

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

Find vychází z vyvážitelnosti binárního stromu.

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

(a,b) – stromy

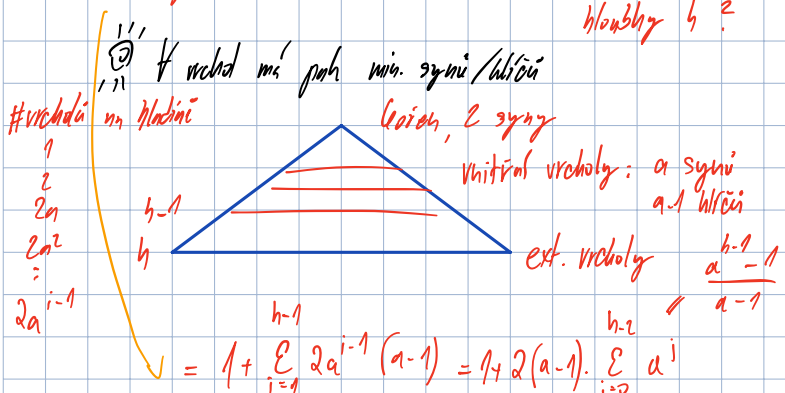
- víceestný vyhledávací strom
- a, b jsou parametry: $a \geq 2$, $b \geq 2a-1$
- všechny ext. vrcholy jsou stejně hluboko
- int. vrcholy mají a až b synů ($a-1$ až $b-1$ klíčů)
  - kořen má 2 až b synů (1 až $b-1$ klíčů)

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 ?

V vrchol má pak min. synů/klíčů



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

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

roste exponenciálně

hloubka tedy roste logaritmicky

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

$\sim b^h \Rightarrow$ hloubka roste logaritmicky

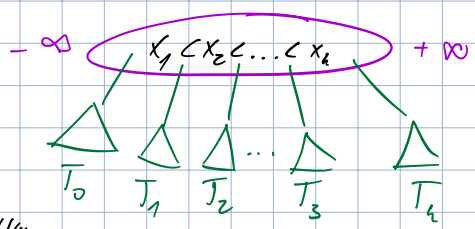
Ideální jsou $(2,3)$ nebo $(2,4)$ stromy, teoreticky.
V reálné ale dlouhém čísle je nejlepší $(256, 512)$-strom

Externí vrchol – null pointer „prázdných listů"
- trochu to zjednodušuje vyvažování mezi vrcholy/stromy
- vyplňují rozsahům mezi klíči

Víceestný vyhledávací strom

- uspořádaný 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:

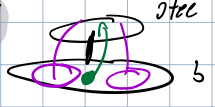
$x_i < y < x_{i+1}$

Operace:

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

Insert: Dělám Find, než dojdu do cíle. Pak přidám klíč do posledního vnitřního vrcholu + občas přetečení klíčů

Přetečení

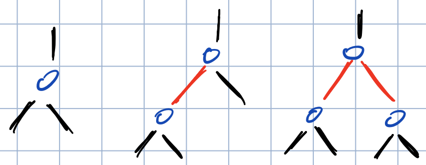


- do otce přidám klíč, případně vyřeším, pokud otec přetekl

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 2 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

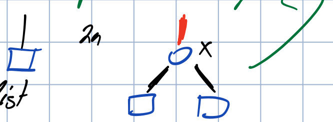
Vychází z podsktry těch 4 předpokladů pro LLRB

Insert:

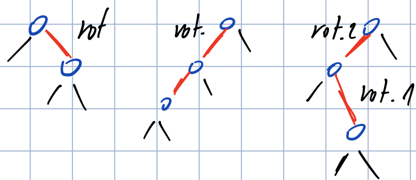
Směrem dolů přebarvujeme 4-vrcholy

↳ můžeme rozbít R axiomy

Nahradíme list za



Směrem nahoru opravujeme R axiomy rotacemi



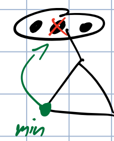
Jakmile se vrátím do kořene, mám korektní LLRB strom

↳ opět trvá $\Theta(\log n)$

– barvičku si může držet vrchol kvůli implementaci

Delete:

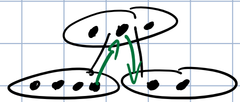
Nejprve převedeme na Delete z nejnižší vnitřní hladiny



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

Najdeme sourozence (třeba levého)

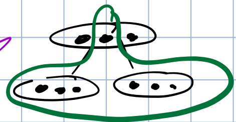
bratr má > a-1 klíčů



přesun klíčů

bratr má a-1 klíčů

slijeme dohromady (u otce 1 klíč smažeme)



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

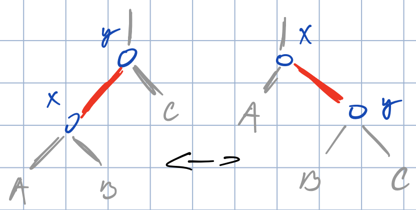
Otec mohl podtéct, pak opakujeme, když podteče, pokud je pravidelný, takže ho jen smažu

$\Theta(\log n)$

– $O(1)$ na hladinu, $\log n$ hloubka

Operace:

Rotace R hrany:



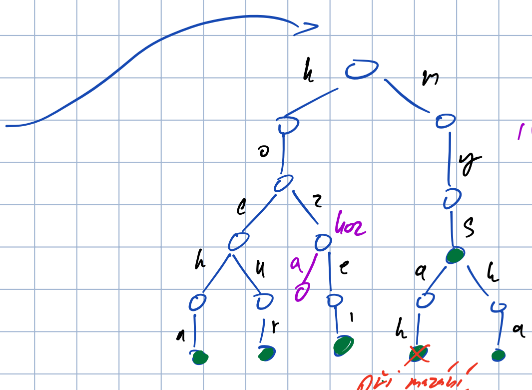
- zachovává B axiomy
- může rozbít R

Přebarvení 4-vrcholu:

- zachovává B axiomy
- může rozbít R

Trie

{kocka, kocour, koza, myš, myška, myšák}



Při mazání mažu příznak

Vrcholy odpovídají prefixům, případné hodnotě slova

Ve vrcholu pole ukazatelů indexované abecedou

Operace:

Member: $\Theta(|y|)$ } Ideální jen ve vzdálenosti slova
Insert: $\Theta(|y|)$

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

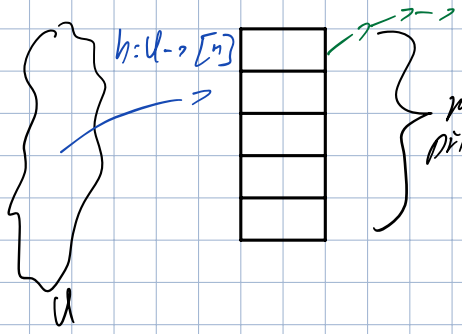
Paměť: $\Theta(\sum |x_i|)$ – musím sečíst všechny slova, když nemají stejný prefix

U velké abecedy bude lepší místo pole:

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

Čas. složitost: $\Theta(|y| \cdot \log |\mathcal{C}|)$ — velikost abecedy, což odpovídá BST stromu

Hashování



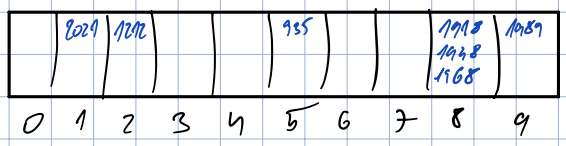
h: hashovací funkce

Kolize ≡ více prvků v přihrádce
– v přihrádce vlastně je seznam

Příklad:

$\{1212, 955, 1018, 1948, 1968, 1989, 2021\}$

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



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, $a \sim 0.618\,m$ — lineární kongruence
- pro $U = [2^w]$, $m = 2^k$

  $x \to (ax \bmod 2^k) >> w-k$ — multiply-shift
- $x_0 \dots x_{d-1} \to \left(\sum_i a_i x_i\right) \bmod m$ — skal. součin
- $x_0 \dots x_{d-1} \to \left(\sum_i a^i x_i\right) \bmod m$ — polynom

Přehashování:

- sledujeme $\alpha := \frac{n}{m}$ — faktor zaplnění
  – chceme $\alpha \le$ konst
  – vzroste-li α příliš, přehashujeme
- zvolíme $m \to 2m$ // $\Theta(n)$

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

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

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

Mezi přehashováním potřebu přidání n operací

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

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

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

$2^{i-1}+1 \dots 2^i+1$ přibylo 2^{i-1} 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 $c > 0$ $\equiv$

ideální funkce má $\frac{1}{m}$

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

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

Potom $\mathbb{E}_{h \in \mathcal{H}}[\underbrace{\#i : h(x_i) = h(y)}_{Y}] \leq c \cdot \frac{n}{m} + 1$ *($+1$: y = jednomu z x_i)*

Nechť $\forall i\ x_i \neq y$

Zavedeme indikátory $I_1 \ldots I_n$:

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

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

$\Rightarrow \mathbb{E}[Y] \leq \frac{c}{m} \cdot n = c \cdot \frac{n}{m}$ ☒

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

$= \Pr_a[ax = ay] \Rightarrow a \cdot (x - y) = 0$

BÚNO $x_1 \neq y_1$

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

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

$\Downarrow$

$\exists!$ řešení pro a_1

$\Pr[a_1 \text{ je řešení}] = \frac{1}{p}$ ☒

Důsledek: $\mathbb{E}$ časové složitosti Findu, Insertu, Deletu je $O(\frac{n}{m})$

To umíme udržet $O(1)$ přehashováním

Otevřená adresace

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

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

- což je permutace na přihrádkách → permutace

Operace:

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

Find: Procházím v pořadí posloupnosti, dokud nalezneme či předem přihrádku

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

Příklady: Lineární přidávání:

- první prvek vyhl. posloupnosti vybrán pomocí hash. funkce, zbytek lineárně procházím přihrádkami

Problém: Čím větší souvislý úsek, tím déle ho udržím, takže se mi to hrne do jedné rudle.

dvojité hashování: prvočíslo

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

dvě hash. funkce

To dobře rozhodit prvek do tabulky a každý prvek má vlastní skok, takže se nemusí tolik trvorit jedna velká rad/c.

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

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

$P_1 - P_2 + 2P_2 - 2P_3 + 3P_3 - 3P_4 + 4P_4 - 4P_5 \ldots$

$P_1 + P_2 + P_3 + P_4 \ldots$

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

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

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

pravděpodobnost, že první je obsazená

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

$\mathbb{E}[\#\text{navštívených přihr.}] = \sum_{i \ge 1} i \cdot P[\text{navštíveno právě } i \text{ přihr.}] =$

$P[\text{navštíveno právě } i \text{ přihr.}] = P_i - P_{i+1}$

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

Rozděluj a panuj

MergeSort $\to$ Merge $(x_1 \ldots x_a, y_1 \ldots y_b)$ ✓ čas $\Theta(a+b)$

– slévání setříd

Pro vstup $x_1 \ldots x_n$:

Pokud $n \le 1$: Return vstup

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

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

Vrátíme Merge $(a_1 \ldots, b_1 \ldots)$

– rekurze je konečná

→ Pomocí rozepsání

Analýza složitosti

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

$T(1) = 1$

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

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

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

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

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

$\vdots$

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

$n/2^i = 1$

$\log n = i$

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

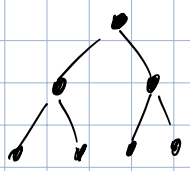
Pamět' v 2:

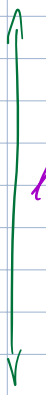
$M(n) = M(n/2) + n$

$M(n) = n + n/2 + n/4 + \ldots$

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

Analýza pomocí stromu rekurze

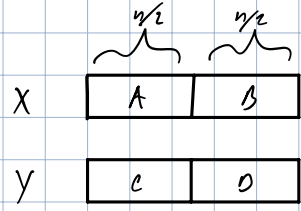




#problémů	velikost problému	čas na hladině
1	n	n
2	$n/2$	n
4	$n/4$	n
⋮	⋮	⋮
2^i	$n/2^i$	n
n	1	n

Celkem $n \cdot \log n$

Násobení dlouhých čísel



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

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

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

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

– 4 součiny $n/2$-ciferných čísel

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

AB
BD

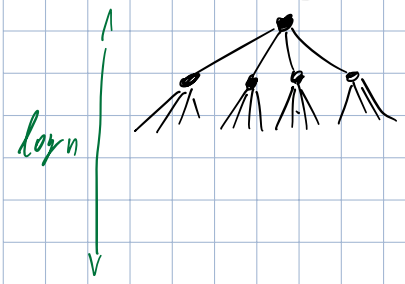
Takže mám pouze tři rekurze!

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

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

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

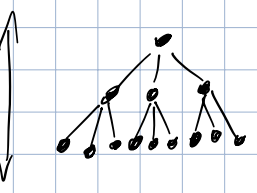
Strom rekurze:



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

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

Není to rychlejší, listů je furt moc



#pp	velikost	čas na pp	čas na hladině
1	n	n	$1 \cdot n$
3	$n/2^i$	$n/2$	$3 \cdot n/2$
⋮	⋮	⋮	⋮
3^i	$n/2^i$	$n/2^i$	$3^i \cdot n/2^i$

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

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

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

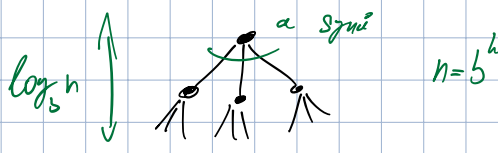
$$\Rightarrow T(n) = \Theta\left(n^{\log_2 3}\right)$$

Master Theorem

Věta: Rekurence $T(n) = a \cdot T(n/b) + \Theta(n^c)$, $T(1) = 1$ pro $a \geq 1, b > 1, c \geq 0$ má řešení:

(a = počet podproblémů, b = faktor zmenšení, $\Theta(n^c)$ = lokální práce)

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



Na i-té hladině:
- a^i podproblémů
- $\left(\frac{n}{b^i}\right)^c$ čas na podproblém
- $a^i \cdot \left(\frac{n}{b^i}\right)^c$ čas na hladině

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

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

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

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

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

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

Zbývá pojednat, kdy n není mocnina b.

n^+ ... nejbližší vyšší mocnina
n^- ... nejbližší nižší mocnina

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

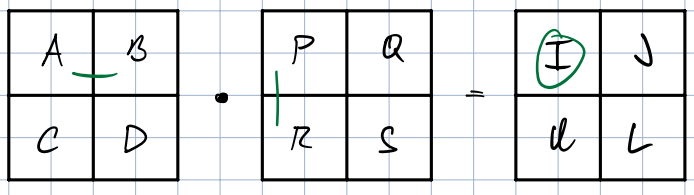
$\left(\frac{n}{b}\right)^- \leq \frac{T(n)}{T(b)} \leq \left(\frac{n}{b}\right)^+$

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

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

Násobení matic - Strassenův alg.

Buňo $n = 2^h$



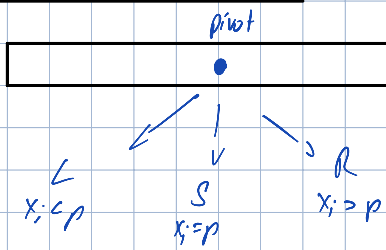
$I = AP + BR$, tedy 8 součinů matic

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

Strassenův trik: Stačí pouze 7 násobení:

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

Quick Select



Vždy bych 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

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

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

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:

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

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

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 $\mathbb{E}[\#\text{pokusů}] \le 2$

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

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

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

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

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)

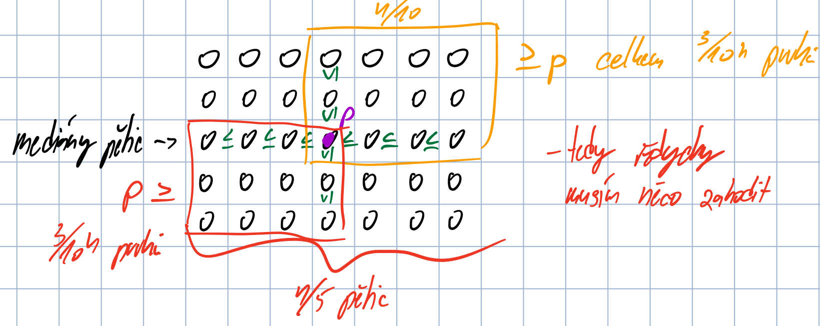
Select($x_1 - x_n$, k):

1) Rozdělíme $x_1 - x_n$ na pětice $p_1 - p_t$
2) Najdeme mediány 5-tic
3) Najdeme medián mediánů pětic:

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

- tohle bude pivot pro hledání k-tého nejmenšího

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



Vždy zahodím alespoň $\frac{3}{10}n$ prvků, tedy zbývá jen $\frac{7}{10}$.

$$T(n) = T(n/5) + T(\tfrac{7}{10}n) + n$$

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

Uhodneme: $T(n) = cn$

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

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

Tedy jsme dokázali, že výběr lineárně.

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: $T(n) = T(n-1) + T(0) + \Theta(n) = \Theta(n^2)$

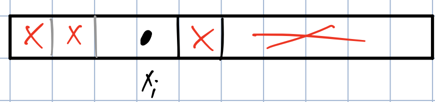
Věta: QuickSort s náhodnou volbou pivota má časovou složitost se střední hodnotou $\Theta(n \cdot \log n)$

Dk#1: Porovnání načítáme prvku, který vzal 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 s x_i:



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

$E[\text{kroků ve fázi}] \le 2$

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

Dynamické programování – nejdelší rost. podposl.

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

0) Pokud $i = n+1$: return 1
1) d = 1
2) Pro $j = i+1 \ldots n+1$:
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 $i \in \{0 \ldots n+1\}$ argumentů → volání se opakují

③ Přidáme cache
→ fce je zavolaná $O(n)$ 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 $i = n, \ldots, 0$ pozpátku:
3. d = 1
4. Pro $j = i+1 \ldots n+1$:
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 $L(\alpha_i \ldots \alpha_n, \beta_j \ldots \beta_m)$

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 provádět dle dopravy:

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

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

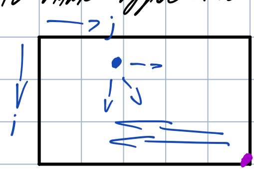
zkusíme všechny, vybereme min.

↳ Je to hodně pomalé! $O(2^n)$

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

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

Tabulku vyplňujeme zprava odspodu, pak už to máme vypočítané předem



Nerekurzivně:

1) Pro $j = 1, \_, m+1$: $T[n+1, j] = m - j + 1$
   Pro $i = 1, \_, n$: $T[i, m+1] = n - i + 1$
   } Operace, pokud jsem přičetl, ty jsou jasně dané
2) Pro $i = n, \_, 1$:
3) Pro $j = m, \_, 1$:
4) $\gamma = 0$, pokud $\alpha_i = \beta_j$, jinak 1
5) $T[i,j] = \min(1 + T[i+1, j], 1 + T[i, j+1], \gamma + T[i+1, j+1])$

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

Optimální BST

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

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

h_i = # vrcholů na cestě do x_i

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

Opt BST(ℓ, p) → opt. cena BST pro $x_\ell \_ x_p$

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

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

6) Return $T[1, n]$

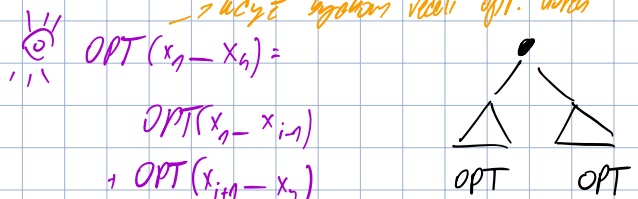
$C_i := T[\ell, i-1] + T[i+1, p]$

Čas $O(n^3)$, prostor $O(n^2)$

Máme ceny, pro strom si budeme u vrcholy pamatovat optimální kořen pro daný interval

C1: Najít T s nejmenší cenou

→ budeš hodně chtít opt. kořen



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

→ hrozně moc kořenů vyzkouším, tak zkusíme $x_1 \_ x_n$ a najdeme min. cenu

Je to pomalé

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

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

Nahradíme rekurzi cyklem... od nejkratších po nejdelší

Dolní odhady

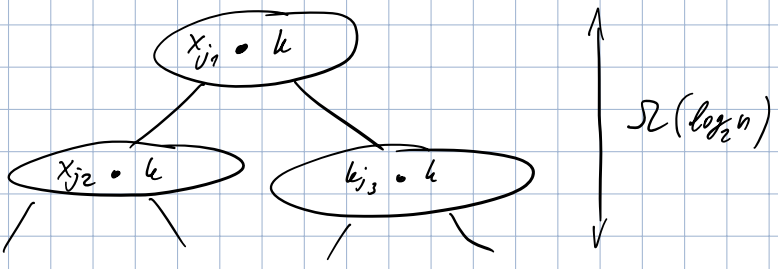
BST

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

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

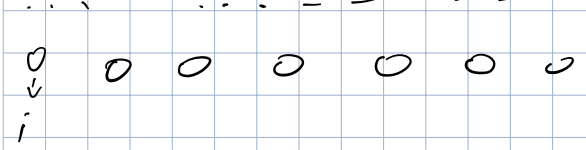
V listu vrátíme výsledek

#listů $\geq n$ ← pro různá k různý výsledek



Max. #listů v BST hloubky $h = 2^h$

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



Třídění:

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

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

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

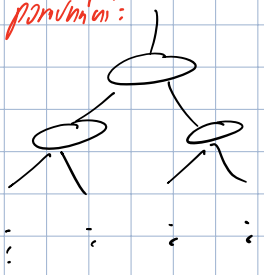
$n! \geq \left(\frac{n}{2}\right)^{\frac{n}{2}}$

$n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot 1$ – prvních $\frac{n}{2}$ prvků je alespoň $\frac{n}{2}$

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

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 vynutit, aby vracel pro každý prvek původní index ve vstupní posl.

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

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