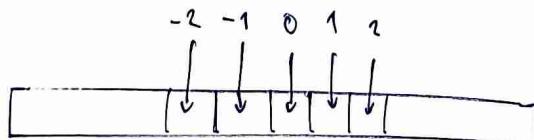


• Random Access Machine - RAM

- počítá s celými čísly neomezené množstvem čísel
- paměť: pole čísel indexované celými čísly



- instrukce

① $Ram \leftarrow co$ literál ~ imm
obsah buněky [adresa] literál
nepřímá adresace [[adresa]]

-1	0	1	2	3
4				-1

$$[[3]] = [-1] = 4$$

- ② $Ram \leftarrow a OP b$ +, -, *, /, mod, dletož 0 je základní

\xrightarrow{XOR} elociálně

&, |, \oplus , \ll , \gg

$$x \ll k = x \cdot 2^k$$

$$x \gg k = x / 2^k$$

následující operace

- ③ halt - zastavení programu

+ za programem je vždy implicitní halt

goto KAM - nepodmíněný skok

if a RELACE b goto KAM
 $= \neq < > \leq \geq$

- vstup: je ve směrových buněkách, jinde čísla

- výstup: rade v nejlevnějších směrových buněkách \leftarrow po haldu

Def: Algoritmus je program RAMu.

Příklad: Bubble sort.

Opakovat

$p \leftarrow \text{NE} \leftarrow 0$

Pro $i = 1, \dots, m-1$:

Pokud $x_i > x_{i+1}$:

$x_i \leftrightarrow x_{i+1}$

$p \leftarrow \text{ANO} \leftarrow 1$

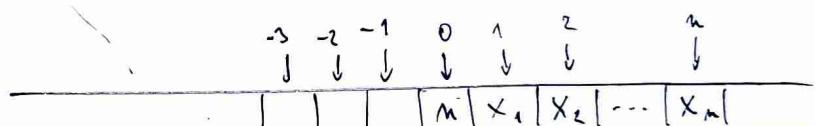
Dokud $p = \text{ANO}$

• v nejlevnějším případě

$$4m + 1$$

• v nejhorším případě

$$8m^2 - 9m + 6 \in \mathbb{H}(m^2)$$



\hookrightarrow směry $A := [-1], B := [-2], \dots, Z := [-26]$

if $[0] \leq 1$ goto END | instrukce main

ZNOVU: $P \leftarrow 0$

$I \leftarrow 1$

DALŠÍ: $J \leftarrow I+1$

if $[I] \leq [J]$ goto OK

$P \leftarrow 1$

$T \leftarrow [I]$

$[I] \leftarrow [J]$

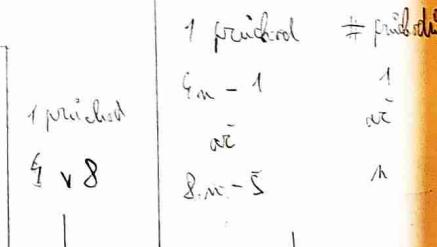
$[J] \leftarrow T$

OK: $I \leftarrow I+1$

if $I < [0]$ goto DALŠÍ

if $P = 1$ goto ZNOVU

END: halt



1 průchod $\#$ průchody
 $4m - 1$
 \vdots
 $8m - 5$

\downarrow

celkově:

$4m + 1$

\vdots

$8m^2 - 5m + 2$

ne zkontrolované kód
poslední případ je vždy
jen $4m - 1$
 $\Rightarrow 8m^2 - 5m + 6$

inicializace

if $[0] \leq 1$ goto END $N \leftarrow [0]$ $B \leftarrow 1$

goto BUILDHEAP

index maxima dvoj. pořadí

MAXCHILD $J \leftarrow 2 * I$ $K \leftarrow J + 1$ if $K \leq N$ goto 2SYN

goto 1SYN

2SYN if $[K] \leq [J]$ goto 1SYN $J \leftarrow K$

1SYN goto MAXFOUND

pokládání hodnot na horní

BUBLANI $R \leftarrow I$ ZNOVU $J \leftarrow 2 * I$ if $J > N$ goto KONECgoto MAXCHILD # $J = \max\{pořadí I\}$ MAXFOUND if $[I] \geq [J]$ goto KONEC $T \leftarrow [I]$ $[I] \leftarrow [J]$ $[J] \leftarrow T$ $I \leftarrow J$

goto ZNOVU

KONEC $I \leftarrow R$ if $B = 1$ goto BUILDING

goto SORTING

build maxheap

BUILDHEAP $I \leftarrow [0]$

DALSI goto BUBLANI

BUILDING $I \leftarrow I - 1$ if $I \geq 1$ goto DALSI $B \leftarrow 0$

zámožné řazení

 $H \leftarrow [0]$ $I \leftarrow 1$ $T \leftarrow [i]$ $[i] \leftarrow [H]$ $[H] \leftarrow T$ $N \leftarrow H - 1$

goto BUBLANI

 $H \leftarrow H - 1$ if $H \geq 1$ goto DALSI #2

konec

END hall



Cena 1 instrukce - něčí normativi

1, jednotková $\Rightarrow \text{čas} = \sum \text{instrukcí}$ - všechny bude zahrnovat do 1 funkce čísla
 \rightarrow ročně jede se konstantním časem!

2, omezená síra slova na W b. \Rightarrow omezená max. abs. hodnota čísla na W

\rightarrow omezení jenom paměti na $2^{W+1} \leftarrow \log_2^n$ $\rightarrow \log(n) := \log_2(n)$

\Rightarrow pro vstup délky n musí být $W \geq \log(n)$ \rightarrow abstraktní kalkulace

\rightarrow použijeme $W = c \cdot \log(n)$ $\Rightarrow |\text{čísla}| \leq n^c$ $\rightarrow \max(n, c')$ pro $n = 0, 1$

\rightarrow tyto dodatečné algoritmy jsou snadno rabiři

3, logaritmická

- instrukce nemají konstantní cenu - rážeři na 10m, s jiné velikosti č. funkce

cena = # bitů operandy včetně adres - i na adresaci něco slouží

4, relativní logaritmická

$\text{cena} = \lceil \frac{\# \text{bitů}}{\log(n)} \rceil$ \rightarrow pro polynomické velikosti čísla \Rightarrow jednotková / konstantní

\rightarrow pro obecná čísla \Rightarrow logaritmická

\Rightarrow budeme používat 4, ale pro normální programy to je 1,

Def: Dobře běhu algoritmu pro vstup X

$t(x) :=$ součet cen provedených instrukcí \leftarrow může byt i nesčítaná

Def: Časová složitost \leftarrow nejhorší případ

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

Def: Prostor běhu algoritmu pro vstup X

$s(x) :=$ max. adresa - min. adresa + 1

\leftarrow rozšiřená během běhu programu

Def: Prostorová složitost

$S(n) := \max \{ s(x) \mid x \text{ je vstup velikosti } n \}$

\rightarrow prostorový prostor \rightarrow je částečná paměť, kde byl vstup se jenom čte

\hookrightarrow výstup smíšený se vlastním rapsorací, ne čte

Def: Asymptotická notace $f: \mathbb{N} \rightarrow \mathbb{R}$ $\mathcal{H}^* := \exists m_0 \in \mathbb{N}: \forall n \geq m_0 - f \text{ roste nejrychleji než } g$

$f \in O(g) \equiv \exists c \mathcal{H}^m: f(n) \leq c \cdot g(n) - f \text{ roste nejrychleji než } g$

$f \in \Omega(g) \equiv \exists c \mathcal{H}^m: f(n) \geq c \cdot g(n) - f \text{ roste alespoň jich } g$

$f \in \Theta(g) \equiv \exists c, c' \mathcal{H}^m: c'g(n) \leq f(n) \leq c'g(n) - f \text{ roste stejně jich } g$

$$\Theta(g) = O(g) \wedge \Omega(g)$$

GRAFOVÉ ALGORITMY

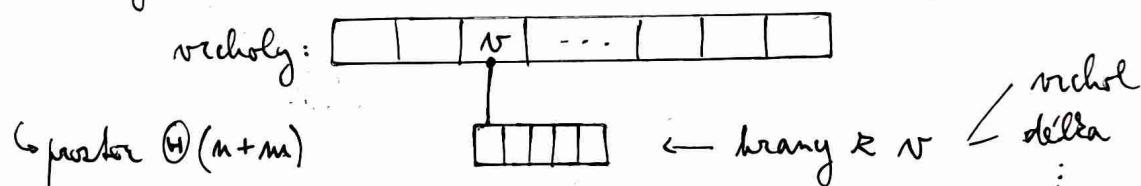
- BFS ~ malejeme do grafu veden
- DFS ~ Ariadnina nit

Def: Graf G BÚNO orientovaný, $n := |V(G)|$, $m := |E(G)|$.

• Reprezentace grafu

① matice sousednosti $A_{ij} := \begin{cases} 1, & v_i, v_j \in E(G) \\ 0, & \end{cases} \rightarrow$ veliká $\Theta(n^2)$

② soumany sousedů



• Prohledávání do hloubky - DFS

$\rightarrow \text{star}(v) = \begin{cases} \text{nevřídel} \\ \text{olvídel} \\ \text{kavřídel} \end{cases}$

$\forall v \in V: \text{star}(v) \leftarrow \text{nevřídel}$

DFS(v): $\begin{array}{l} 1. \text{ star}(v) \leftarrow \text{olvídel} \\ 2. \text{ Pro } t \text{ tak } vt \in E: \\ \quad 3. \text{ Pokud } \text{star}(u) = \text{nevřídel}: \\ \quad \quad 4. \text{ DFS}(u) \\ 5. \text{ star}(v) \leftarrow \text{kavřídel} \end{array}$ $\Theta(\deg^{\text{out}}(v))$

smysl starin: $N \rightarrow O \rightarrow Z$

\Rightarrow pro dležitost není rádny
verchol olvídel

DFS nevšerš se kavří verchol některá

\Rightarrow do vercholu násouhne max 1

Lemma: Po dokončení DFS(v) je $\forall u \in V(G)$ $\text{star}(u) = \begin{cases} \text{zavřený} & \Leftrightarrow \exists \text{ cesta z } v \text{ do } u \\ \text{neviděný}, \text{ jinak.} \end{cases}$

Důkaz: 1, v zavřený $\Rightarrow v$ dosažitelný $\Leftrightarrow v$ otevřený v zavřeném \Rightarrow dosažitelný

\rightarrow Důkaz indukce podle doby běhu programu - invariant

$\rightarrow v$ otevřelo $x \Rightarrow$ podle i.p. je x dosažitelné



$\Rightarrow x$ nede do v hranu

$\Rightarrow x$ v nede do v sled/cesta $\Rightarrow v$ dosažitelné

2) v dosažitelný $\Rightarrow v$ na konci zavřený

\rightarrow Důkaz na lezení minimálního protipříkladu



pro spor: $\exists x \in V(G)$ l.e. x je dosažitelný, ale neviděný

\Rightarrow existuje x l.e. cesta z v do x má nejméně hranu

\Rightarrow předchozí P je dobrý \Rightarrow viděný \Rightarrow otevřený \Rightarrow objevili jsme x \square

Lemma: DFS běží v čase $\Theta(n+m)$

Důkaz:

$$1) \sum_v \Theta(1 + \deg^{\text{out}}(v)) \in \Theta(n+m)$$

→ vrcholy: $1, 5 \rightarrow \Theta(n) \quad \left\{ \begin{array}{l} \text{vrcholy: } 1, 5 \rightarrow \Theta(n) \\ \text{hrany: } 2, 3, 4 \rightarrow \Theta(m) \end{array} \right\} \Theta(n+m)$

2) včítajeme vrcholy a hrany

Opozvané DFS

1. $\forall v \in V(G)$: $\text{star}(v) \leftarrow$ neviděn

← navštívím celý graf + stále $\Theta(n+m)$

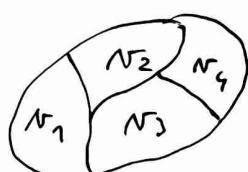
2. Pro $\forall v \in V(G)$:

→ pro \forall podgraf H je $\Theta(n'+m')$

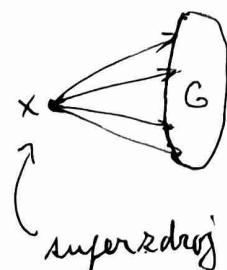
3. Pokud $\text{star}(v) =$ neviděn:

\Rightarrow posčítá se k v na $\Theta(n+m)$

4. $\text{DFS}(v)$



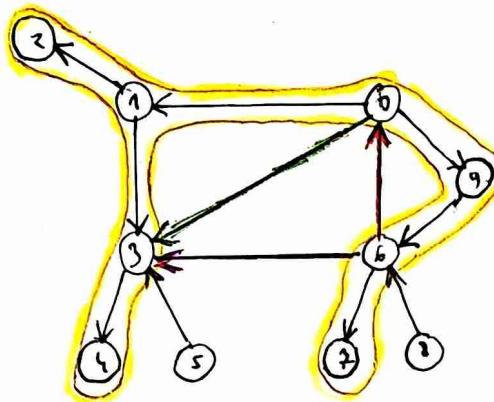
↳ nebo alternativně:



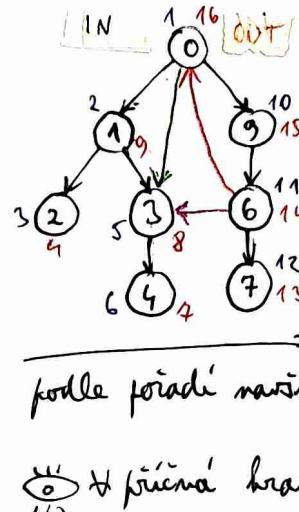
$\text{DFS}(x) \Rightarrow \Theta(n+m)$ snadno vidět

• DFS strom grafu

- získáme kostru dosažitelné části grafu



DFS(0) hranou pomocí jednoho vrcholu mají prioritu podle čísla cílového vrcholu



- stromové hranu \rightsquigarrow dosažitelné ježíšky, objevené pořadí
- spěšné - vede do předku
- dopředné - vede do zavřeného počtu
- příčné - vede do navštíveného vrcholu což nemá předek ani potomce

podle pořadí navštívení

\Downarrow příčná hraná vede \leftarrow do minimality

inicializace

$\forall v \in V(G)$: star(v) \leftarrow neviděný, in(v), out(v) $\leftarrow \emptyset$

$T \leftarrow 0$ \rightarrow hodiny když někdo při zkoumání skočí

DFS(v): 1. star(v) \leftarrow zavřený

2. $T \leftarrow T+1$, in(v) $\leftarrow T$

3. Pro $w \in V(G)$:

4. Pokud star(w) = neviděný:

5. $DFS(w)$

6. star(v) \leftarrow zavřený

7. $T \leftarrow T+1$, out(v) $\leftarrow T$

\rightarrow zavřené a zrovnaží \sim záviseční

$(_0(1(2)_2(3(4)_4)_3)_1(9(6(7)_7)_6)_9)_0$

\rightarrow in(v) = pozice (

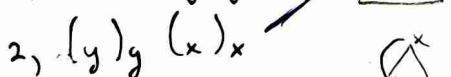
out(v) = pozice)_v

• Klasifikace bran pomocí rámců

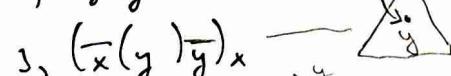
\rightarrow pro branu $x, y \in E(G)$:



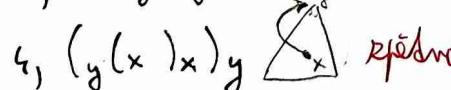
příčná star(y) = zavřený



stromová star(y) = neviděn



dopředná star(y) = zavřený



spěšná star(y) = zavřený při objevení

\rightarrow pokud brány umíme v $O(1)$ ejakulid, kterého druhé je

\rightarrow v neorientovaném grafu

brana x,y ještěna

forwarde
stromová

spěšná

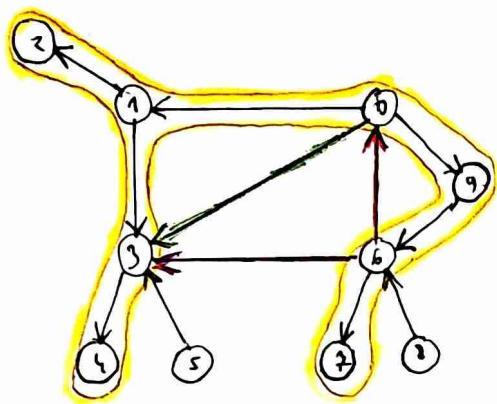
backward
spěšná

dopředná

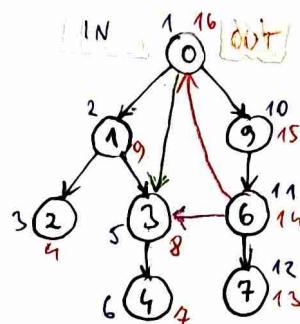
\rightarrow nejsou tedy příčné brany

• DFS strom grafu

- získáme kostru dosažitelné části grafu



DFS(0) hrany jdoucí z jednoho vrcholu mají prioritu podle čísla cílového vrcholu



- **stromové hrany** as domácí hrany, nejsou významné pro řešení
- **spěšné** - vede do předku
- **dopředné** - vede do zavřeného vrcholu
- **příčné** - vede do navštívěného vrcholu což nemá předek ani potomci

podle pořadí navštívění

→ příčná hrana vede ← do minulosti

inicializace

$\forall v \in V(G)$: star(v) \leftarrow neviděný, in(v), out(v) $\leftarrow \emptyset$

T $\leftarrow 0$ čas - hodiny kroků pri znávání stavu

DFS(v): 1. star(v) \leftarrow zavřený

2. T $\leftarrow T+1$, in(v) $\leftarrow T$

3. Pro $vw \in E(G)$:

4. Pokud star(w) = neviděný:

5. DFS(w)

6. star(v) \leftarrow zavřený

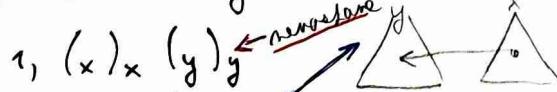
7. T $\leftarrow T+1$, out(v) $\leftarrow T$

→ otevírání a zavírání v závislosti
 $(_0(1(2), (3(4, 5)), 6, 7, 8, 9), 10, 11, 12, 13)$

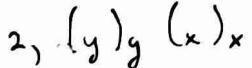
→ in(v) = pozice (v)
out(v) = pozice (v)

• Klasifikace bran jenom v rámcích

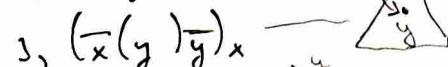
→ pro brannu $xy \in E(G)$:



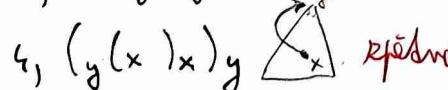
→ **opravná** star(y) = zavřený



→ **stromová** star(y) = neviděn



→ **dopředná** star(y) = zavřený

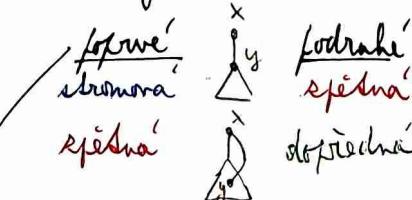


→ **spěšná** star(y) = zavřený při objevení

→ pokud hrana umíme v $O(1)$ sjistit, kterého druhu je

→ v neorientovaném grafu

hrana x,y ještě ne



→ region den příčné hrany

Věta: DFS(v) běží v čase $\Theta(n+m)$ a prostoru $\Theta(n+m)$ a majde dosažitelné vrcholy a klasifikaci bran mezi nimi.

Alebojní mosty v neorientovaných grafech

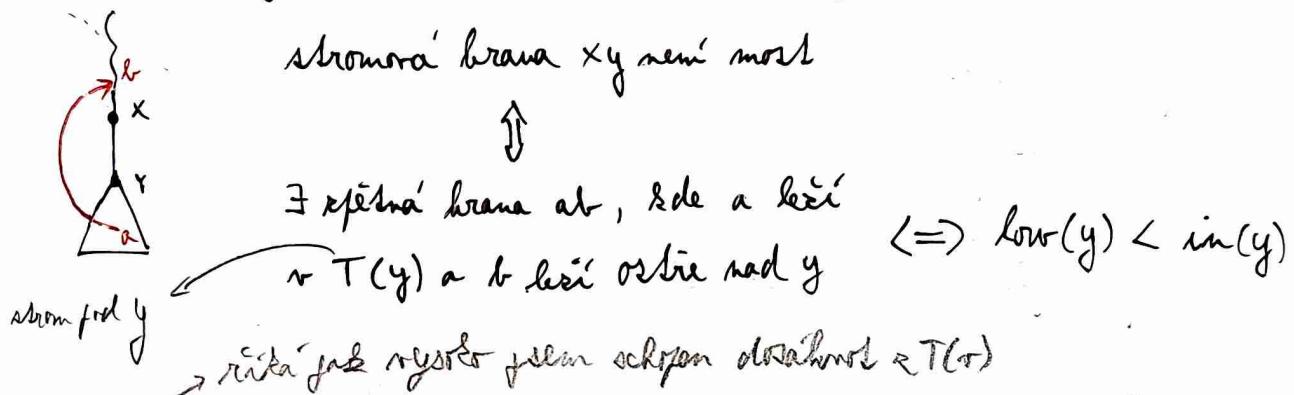
Def: Brana e je most v $G \equiv G - e$ má více komp. souč. než G .

\Leftrightarrow e nemá most $\Leftrightarrow e$ leží na kružnici



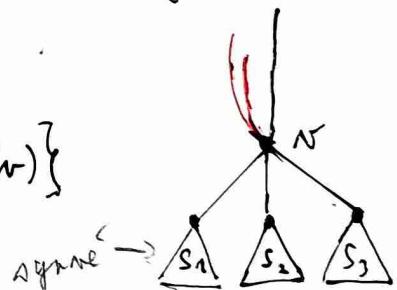
\circlearrowleft Zpětná brana nemůže být most.

\Rightarrow Které mosty jsou stromové hrany \rightarrow kdy leží s. brana na kružnici?



Def: $\text{low}(v) := \min \{ \text{in}(b) \mid ab \text{ je zpětná brana s } a \in T(v) \}$

\Rightarrow chci spočítat $\text{low}(v)$ pro $\forall v \in V$ v lineárním čase

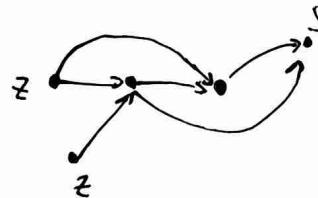


$\Theta(n+m)$ $\text{low}(v) = \min \{ \text{low}(s_1), \dots, \text{low}(s_k), \text{in}(b) \mid vb \text{ zpětné hrany} \}$

• Acyklicky orientované grafy - DAGy

Def: Zdroj je vrchol s $\deg^{\text{in}}(-) = 0$.

Sleč je vrchol s $\deg^{\text{out}}(-) = 0$.



Lemma: Kardý konečný DAG má alespoň 1 zdroj a sleč.

Dě: Pro sleč: vyber vrchol $v \rightarrow$ je sleč ✓

↳ nemá sleč \Rightarrow vede k ní brana do v ,

1, mazacíme na sleč ✓

2, počítáme do $\infty \rightarrow$ & graf je konečný

3, sleč nenajdeme \Rightarrow kružnice &

Def: Transformovaný graf $G^T :=$ graf vzniklý z G otočením říp.

⊗ Transpozice cyklu je cyklus $\Rightarrow G$ je DAG $\Leftrightarrow G^T$ je DAG. } nejde sleč z G^T

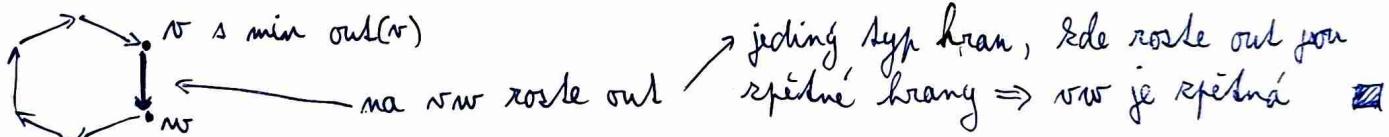
⊗ v je zdroj z $G \Leftrightarrow v$ je sleč z G^T } \Rightarrow zdroj z G

Alg: Je G DAG? \Leftrightarrow Má G cyklus?

⊗ Vzpětná hrana leží na cyklu

Lemma: Na každém cyklu leží alespoň 1 vzpětná hrana.

Dě: Na cyklu si vyberu vrchol s největším $\text{out}(v)$



Věta: Graf je DAG \Leftrightarrow opakování DFS nena jede vzpětnou hranu.

• Topologické uspořádání

→ Kolba průby pro poslouchání

Def: Topologické uspořádání grafu $G = (V, E)$ je lineární uspořádání \preceq na V t.ž.

$$\forall u, v \in E: u \preceq v.$$

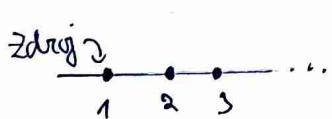
Topologické řazení = řazení vrcholů podle 1. nsp.

Věta: Graf G má T.U. $\Leftrightarrow G$ je DAG.

Dě: \Rightarrow : má T.U. \Rightarrow nemá cyklus \Rightarrow je DAG

\Leftarrow : postupným odváháváním zdrojů

libovolný zdroj schází může být pouze
→ vzhledem k \preceq další zdroj



Toto leží $O(n \cdot m)$, ale jde k lineárně
→ pamatuj si někde zdroje →

⊗ G má jednoznačné T.U. \Leftrightarrow tento alg. má v \forall vrcholu jen 1 zdroj na výběr.

Věta: Opakování DFS opouští vrcholy v opacím topologickém uspořádání.

Důkaz: $\text{out}(v)$ říší vrcholy \Rightarrow lineární uspořádání

\Rightarrow sestrojíme opacné a mláceme, že je topologické

- Pro $u, v \in E$ chceme: $u \leq v \Leftrightarrow \text{out}(u) > \text{out}(v) \Leftrightarrow$ na uv elevá out
 $\Leftrightarrow uv$ není řešená

$\Rightarrow v$ DAGu řešené hrany nejsou \Rightarrow Abylo můžou udelat pro každou hranu

- opakování DFS majde T.V. v čase $\Theta(m+n)$

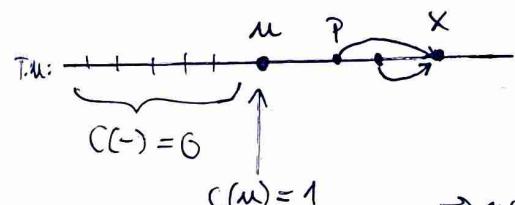
Využití T.V.: Linearizace částečného uspořádání.

\hookrightarrow Č.M. \sim DAG kde vrcholy jsou řešeny a hrany nerovnosti \Rightarrow doplníme ho na lineární

Alg: Počet cest z u do v (v DAGu).

\rightarrow počítáme $C(x) := \# \text{cest z } u \text{ do } x$ $C(N)$

\rightarrow indukce podle T.V. \rightarrow předchůdci x leží před $x \Rightarrow$ z i. předchozích mě
enáme jejich $C(P)$

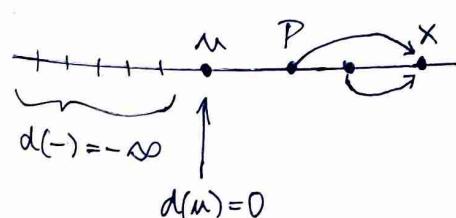


$$C(x) = \sum_{P \in E} C(P)$$

\rightarrow majdu T.M. + konstantní čas na vrchol i na hranu \Rightarrow $\Theta(n+m)$

Alg: Délka nejdelsí (nejkratší) cesty z u do v (v DAGu).

\rightarrow počítáme $d(x) :=$ délka nejdelsí cesty z u do x



$$d(x) = 1 + \max \{ d(p) \mid p \in E \} \rightarrow \Theta(n+m)$$

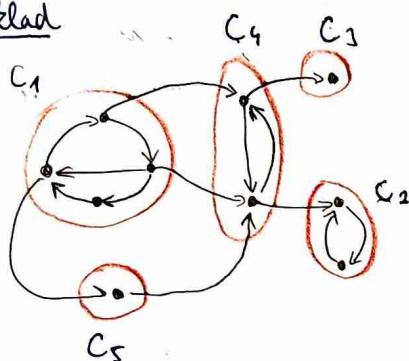
Komponenty silné souvislosti

Def: Nechť $G = (V, E)$ je orientovaný graf. Definují \rightarrow, \leftarrow relace na V :

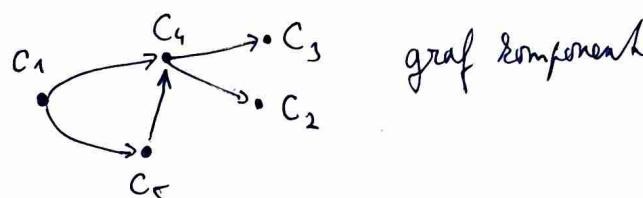
$$\begin{aligned} u \rightarrow v &\equiv \exists \text{ cesta z } u \text{ do } v \\ u \leftarrow v &\equiv u \rightarrow v \wedge v \rightarrow u \end{aligned} \quad \left\{ \begin{array}{l} \leftarrow \text{ je ekvivalence} \end{array} \right.$$

Def: Komponenty silné souvislosti jsou podgrafy indukované ekvivalencečními řídáním \leftrightarrow .

Příklad



→ když cyclus komunikuje se svým vlastním jen jednosměrně, tak tvorí komponentu



Def: Pro o. graf G definujeme graf komponent $K(G)$:

$V(K(G)) :=$ komponenty silné souvislosti G

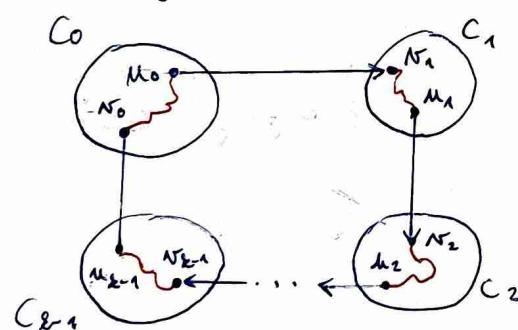
$$C_1 C_2 \in E(K(G)) \equiv \exists v_i \in C_1, v_j \in C_2 : v_i v_j \in E(G)$$

$\left\{ \begin{array}{l} K(G) lze z G vyhodit \\ \text{postupnou konstrukcí bran} \\ \text{mezi vrcholy ve stejné} \\ \text{komponentě} \end{array} \right.$

Věta: Pro každý G je $K(G)$ DAG.

index mod 2

Def: Kdyby v $K(G)$ byl cyclus $C_0 C_1 \dots C_{k-1} C_0$



→ z definice $K(G)$

$$\forall i \exists v_i \in C_i, v_{i+1} \in C_{i+1} : v_i v_{i+1} \in E(G)$$

→ \exists cesta z v_i do v_i vnitř C_i

⇒ všechna $v_i v_i$ leží na cyclus v G

⇒ C_0, \dots, C_{k-1} nejsou silné komponenty \square

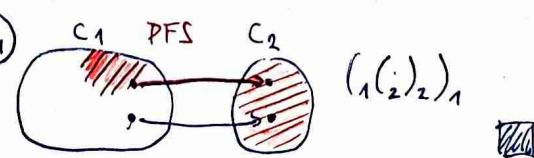
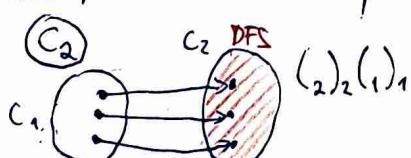
Důsledek: Můžeme rozlišit zdrojové a stohové komponenty.

Podporání:

- ① Je-li C stohová komponenta a $v \in C$, pak DFS(v) projde prvně C . → můžem najít stohovou komponentu?
- ② Vrchol s min. outem leží ve zdrojové komponentě $\leftarrow v \text{ DAGu!}$ } Pro DAGy z T.U.
- ③ Vrchol s max. outem leží ve zdrojové komponentě $\leftarrow v$ obecném grafu
- ④ Transpozice stohí relaci \rightarrow , ale \leftrightarrow neplatí \Rightarrow rachovává komp. s. s. $\Rightarrow v K(G^T)$ se očekává sítě
- ⑤ $\Rightarrow K(G^T) \cong K(G)^T \Rightarrow$ majdu $v \in$ zdroj. k. $v G^T \Rightarrow v \in$ stoh. k. $v G$

Lemma: Pokud $C_1 C_2 \in E(K(G))$, pak $\max_{u \in C_1} (\text{out}(u)) > \max_{v \in C_2} (\text{out}(v))$.

Def: Operace DFS vzdoučí dvířko do



Při procházení vrcholu podle klesajících outů, tak C_1 podráží před C_2 ⇒ komponenty objevujeme od zdrojových ke stohovým

Algoritmus na komponenty silné souvislosti

1. sestroj G^T] $\Theta(n+m)$
2. $Z \leftarrow$ prázdný zášobník
3. opakované DFS na G^T ,
při zavírání v jej přidáme do Z] $\Theta(n+m)$
4. $\forall v : \text{komp}(v) \leftarrow \emptyset, k=0$ $\nwarrow Z$. budu mít klesající řadu na $G^T \Rightarrow$ když je k . G^T
 \Rightarrow stohové ř. G
 \hookrightarrow použití na něj DFS
5. Postupně odberáme vrcholy v ze Z :
6. Pokud $\text{komp}(v) = \emptyset$:] $\Theta(n+m)$
7. $\text{komp}(v) \leftarrow k++$
8. $\text{DFS}(v) \sim G$, nechceme do vrcholů s $\text{komp} \neq \emptyset$
navštívěným vrcholem nastavujeme $\text{komp}(-) \leftarrow \text{komp}(v)$

Věta: Komponenty silné souvislosti lze majít v čase i prostoru $\Theta(n+m)$.

Nejkratší cesty

→ orientovaný graf $G = (V, E)$ a délkami hran $l: E \rightarrow \mathbb{R}_0^+$

Def: Délka sledu S : $l(S) := \sum_{e \in E(S)} l(e)$

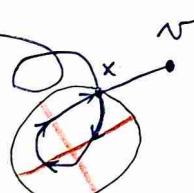
$\rightarrow d$ není symetrická
 \Rightarrow není kvadratice

Def: Vzdálenost z u do v : $d(u, v) := \min \{l(\pi) \mid \pi \text{ je cesta z } u \text{ do } v\}$, $d: V^2 \rightarrow \mathbb{R}_0^+ \cup \{\infty\}$

Lemma: Pro $\forall u, v$ sled S existuje i už cesta P t. j. $l(P) \leq l(S)$

\rightarrow min přes sledy je
 \nearrow stejně jako přes cesty

Def: M

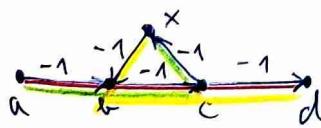


\rightarrow z S odstraníme cykly } \rightarrow sled má konec dostanu cestu
} \Rightarrow # hran kleene
} $\Rightarrow l(S)$ nevzroste

Lemma (Δ -meromnost): $\forall u, v, w: d(u, v) \leq d(u, w) + d(w, v)$

Def: neznačíme nejkratší už a nové cesty \Rightarrow slevením už sled délky $\Rightarrow \exists$ už cesta, co nemá délku

→ Co když připustíme $l < 0$?



\Rightarrow rozbije se do Lemma \Rightarrow rozbije se Δ -meromnost

$$d(a, d) \leq d(a, x) + d(x, d)$$

$$-3 \leq -3 + -3$$

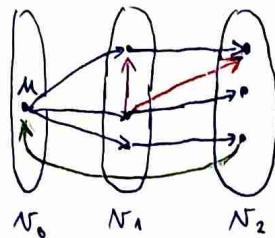
{ neexistuje nejkratší ad-sled
nejkratší cesty jsou dvě
 \hookrightarrow nemůže je mít polynomiálně

→ problém jsou zájorné cykly \Rightarrow pak neplatí do Lemma

• Prohledávání do šířky - BFS

• BFS(u)

• vzdely:



hrany → střemové → vyhledávání stromu nejkratších cest
příčné → obsahuje nejkratší cesty z u do všech ostatních vrcholů
zpečné
dopředně příčné

☒ Neexistuje hrana, která by vedla o více než 1 vrcholu dopředu.

$\Theta(m+m)$

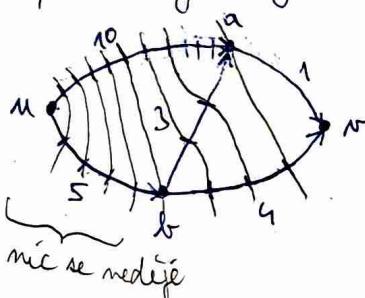
• fronta: vzobírám v_i | vyhledávím v_{i+1}

• $d(v) :=$ číslo vzdely nebo $\emptyset \Rightarrow \text{☒ } d(v) = d(u, v) \rightarrow$ pro hrany délky 1

• $p(v) :=$ předchůdce $v \rightarrow$ z předchůdců lze sestrojit strom nejkratších cest

• BFS s podrozdělováním bran

→ pro hrany délky $l \in \mathbb{N} \rightarrow$ hrany podrozdělíme na jednotkové hrany & BFS



$\Theta(m + m L)$

↑ max $l(e)$

⇒ pomalej je to čítání na dlouhých hranaх

⇒ přesložitě to ⇒ Dijkstra pro $N_0 \rightarrow Q^+$

• Dijkstrův algoritmus

slav(v) ... 0, 2, ∞

$h(v)$... hodnota

Inicializace:

$\forall v: \text{slav}(v) \leftarrow \text{neviděn}, h(v) \leftarrow +\infty, p(v) \leftarrow \emptyset$] $\Theta(n)$

právě nejvíce n-tak

$\text{slav}(u) \leftarrow \text{obslíbený}, h(u) \leftarrow 0, \text{Insert}(u)$

1. Dokud $\exists v: \text{slav}(v) = \text{obslíbený} \wedge h(v)$ je minimální: ExtractMin] $m \cdot T_X$

2. Pro $\forall vw \in E:$

3. Pokud $h(w) > h(v) + l(vw):$

4. $h(w) \leftarrow h(v) + l(vw)$ → neviděný $\text{Insert}(w)$

5. $\text{slav}(w) \leftarrow \text{obslíbený}$ → obslíbený $\text{Decrease}(w)$

6. $p(w) \leftarrow v$

7. $\text{slav}(v) \leftarrow \text{zavřený}$

$m \cdot T_I$
+
 $m \cdot T_D$

→ z BFS plyne, že pro gladné délky hrani nikdy neobslován zavřený vrchol

→ každý vrchol zavřen právě jednou ← pro $l \in \mathbb{R}_0^+$!

Věta: Dijkstrův algoritmus v poli spočte $d(u, *)$ v čase $\Theta(m^3)$.

Složitost Dijstry: $\mathcal{O}(m \cdot T_I + m \cdot T_X + m \cdot T_D)$

	hole	halda	fibonacci halda	d-reg. halda
Extract Min	T_X	m	$\log n$	$\log m$
Insert	T_I	1	$\log n$	1
Decrease	T_D	1	$\log n$	1
	O	m^2	$(m+m) \log m$	$\frac{m \cdot \log(m)}{\log(m/m)} \in O(m \log m)$
jídké: $m \in O(n)$		m^2	$n \log n$	$n \cdot \log n$
huské: $m \in O(n^2)$		m^2	$n^2 \log n$	$n^2 = m$

• Binární halda = úplný bin. strom

↳ rychlejší než jídké i huské

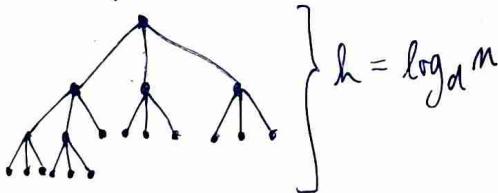
→ haldové uspořádání: $\forall u, v \in E : h(u) \leq h(v) \Rightarrow \text{kören} = \min$

- Extract Min → procházet kořen a posledníノvek \Rightarrow bubláni a řešení dolů
- Insert → přidat na konec + bubláni nahoru } Insert = Decrease
- Decrease → snížit hodnotu + bubláni nahoru } Extract Min = Increase

↳ musíme si měkkle nějak pro každý vrchol pamatovat jeho index v halde

↳ decreasní uspořádání nazíváme ⇒ chceme je rychlejší

• d-reg. halda



- bubláni nahoru $\rightarrow \log_d^m = \frac{\log n}{\log d} \rightarrow$ rychlejší
- bubláni dolů $\rightarrow d \cdot \log_d^m = d \cdot \frac{\log n}{\log d} \rightarrow$ pomalejší

$$DA: O\left(m \cdot \frac{\log n}{\log d}\right) + m \cdot \frac{d \cdot \log m}{\log d} + m \cdot \frac{\log m}{\log d} = O\left(\log m \left(\frac{m \cdot d}{\log d} + \frac{m}{\log d}\right)\right)$$

$$\Rightarrow \text{chceme minimalizovat } O\left(\frac{m \cdot d}{\log d} + \frac{m}{\log d}\right) \Rightarrow m \cdot d = m \Rightarrow d = \frac{m}{m} = 1 \Rightarrow O\left(\frac{m \log m}{\log(m/m)}\right) \in O(m \log m)$$

$$\Rightarrow \text{zvolíme } d = \max\left(\lceil \frac{m}{m} \rceil, 2\right)$$

↳ rychlejší

$$\Rightarrow O\left(\frac{m \log m}{\log(m/m)}\right) \in O(m \log m)$$

$$\Rightarrow \text{něco menší: } m \in O(m^{1+\varepsilon}) \Rightarrow O\left(\frac{m \cdot \log m}{\log(m^\varepsilon)}\right) = O\left(\frac{m \log m}{\varepsilon \cdot \log(m)}\right) = \underline{\underline{O(m)}}$$

• Správnost Dijstry

→ mákali jsme pro $l \in \mathbb{Q}^+$

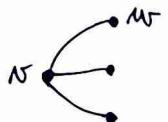
→ odvoďme správnost pro obecnější algoritmus s $l \in \mathbb{R}^+$

Relaxační algoritmy

① udržujeme ohodnocení vrcholů $h: V \rightarrow \mathbb{R}^*$

- inicializace: $h(*) \leftarrow \infty$, $h(u) \leftarrow 0$
- cíl: $\forall v: h(v) = d(u, v)$

② relaxace v



$\forall w \in E: \text{počasime se snížit } h(w) \text{ na } h(v) + l(vw)$

③ slavy vrcholů

- neviděn $\rightarrow h(v) = +\infty$
- otevřený \rightarrow od poslední změny $h(v)$ nebyl relaxován
- zavřený \rightarrow od poslední relaxace nebyla $h(v)$ změněna

Inicializace:

$$h(*) \leftarrow \infty, \text{star}(*) \leftarrow \text{neviděný}$$

$$h(u) \leftarrow 0, \text{star}(u) \leftarrow \text{otevřený}$$

1. Dokud $\exists v$ otevřený:

1. relaxujeme v , při změně $h(v)$:
 $\text{star}(v) \leftarrow \text{otevřený}$
2. $\text{star}(v) \leftarrow \text{zavřený}$

Invariant O (ohodnocení):

- 1, $h(v)$ vždy naroste ✓
- 2) Počud $h(v) < \infty$, tak
 $\exists vw$ -sled délky $h(v)$

Dle: indukci podle délky běhu
 \Rightarrow vždy mám vw -sled, chci uv -sled

$$u \xrightarrow{\quad} v \xrightarrow{\quad} w \quad h(w) = h(v) + l(vw)$$

Lemma D (dosážitelnost): Když se alg. rozstaví, pak $h(v) < \infty \Leftrightarrow v$ je dosážitelný z u .

Dle: $\Rightarrow: \exists O. \Leftarrow: \text{minimální protipříklad}$

\hookrightarrow nejmenší nejbližší co do počtu bran špatný vrchol x

$$u \xrightarrow{\quad} P \xrightarrow{\quad} x \quad P \text{ dobrý} \Rightarrow \text{otevřen} \Rightarrow \text{relaxován} \Rightarrow x \text{ objeveno} \quad \square$$

Lemma V (vzdálenost): Po rozstavení: $\forall v: h(v) = d(u, v)$.

Dle: Uživo v dosážitelný (viz D). v je špatný $\equiv v$ je dosážitelný & $h(v) > d(u, v)$.

\Rightarrow ze všech nejbližších špatných vrcholů si vyberu ten, který je nejbližší

$$\circlearrowleft x \neq u \Rightarrow \exists P: u \xrightarrow{\quad} P \xrightarrow{\quad} x$$

\hookrightarrow co do # bran $\rightarrow x$

$$P \text{ dobrý} \Rightarrow \text{otevřen} \Rightarrow \text{relaxován} \Rightarrow \text{pak } h(x) \leq \underbrace{h(P) + l(Px)}_{d(u, P)} = d(u, x)$$

Věta: Počud relaxační algoritmus doběhne, tak spočítá správné vzdálenosti.
Brihemž doběhne v grafech bez károvaných cyklů.

Dijstra je relaxační

→ v kroku 1 vybíráme v s.t. $h(v)$ je minimální

Invarian M (monotonie): Pro graf s nezápornými délkami hran platí:

① $\forall v$ otevřený, $\forall z$ zavřený: $h(z) \leq h(v) \Rightarrow zavřené \leq h(v) \leq otevřené < neviděné$

② $h(z)$ zavřeného z se nemění \Rightarrow řádný vektor neotevřené druhou

Důkaz: indukce podle # relaxací \sim podle doby běhu.

①, ② platí a relaxujeme $v \rightarrow$ hrana do w

② w zavřené: $h(w) \leq h(v) \Rightarrow h(w)$ nemění

① finál: pokud minimální $h(w)$, tak $\underline{h(w)} = h(v) + l(vw) \geq h(v) \geq zavřené$ \blacksquare
 \Rightarrow pokud v zavřen a přidám k zavřeným

Důkaz: $\forall v$ je zavřen nejdříve jednorázově \Rightarrow # relaxací $\leq n$

\Rightarrow algoritmus se rozšiřuje → podle lemma V je správně

Věta: Dijstrin algoritmus využívá vektor v pojednávající $d(u, -)$,
 v okamžiku zavření je $h(v) = d(u, v)$ a v se nemění.

Když jsou zájmové hrany, tak
 Dijstra je správný, ale může být
 efektivitně špatný
 \square

zájmové cykly být mohou

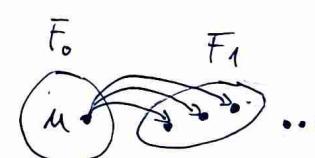
Bellman-Fordův algoritmus

→ relaxační algoritmus, kde zavíráme nejdříve otevřený vektor \Rightarrow otevřené vektoru jsou ve frontě

Věta: B.F. algoritmus spočítá $d(u, *)$ v čase $O(n \cdot (n+m))$ pro kádry graf bee ráporných cyklů.

Def: Fáze běhu F_0, F_1, \dots

$O(n \cdot m)$



→ až na jiné \exists , když máme na konci

• $F_0 \rightarrow$ otevřené u

• $F_{i+1} \rightarrow$ zavřené vektoru otevřených v a F_i

→ 1 vektor může být ve více fázích

⇒ fronta se zavádí

F_i \square F_{i+1}

Lemma: 1 fáze trvá $O(n+m)$.

$$\sum_{v \in F_i} deg^{out}(v) = n$$

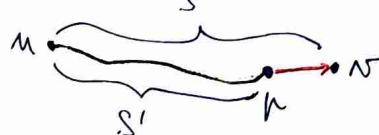
Důkaz: ve fázi je nejdříve n vektorů \Rightarrow všechny zavře a relaxuje \Rightarrow řádku max na m hran \blacksquare

Invarian: Na konci F_i je $\forall v$ $h(v) \leq$ délky všech uv sledů o nejdříve i hranách.

Důkaz: indukce podle i. $i-1 \rightarrow i$:

→ pro min. vek. i hran \in I.P.

$S =$ nejkratší uv sled s právě i hranami



• I.P. \Rightarrow na konci F_{i-1} : $h(p) \leq l(S)$

• nejpozději v F_{i-1} dostalo p into $h(p) \Rightarrow$ otevřeno p

• nejpozději v F_i bylo p zavřeno a relaxováno $\leftarrow h(p)$ se měnilo mohlo

\Rightarrow Takže: $h(v) \leq h(p) + l(pv) \leq l(S') + l(pv) = l(S)$ měnit, ale nevyroste

Důkaz: Na konci F_m je $\forall v$ $h(v) = d(u, v) \Rightarrow$ alg. se rozšiřuje po m fázích.

⇒ něta platí: $\therefore O(n) \cdot O(n+m) \in O(n \cdot (n+m)) \blacksquare$

• Minimální kostra - vždy také nejlehčí

→ Máme neorientovaný souvislý graf G , váhy $w: E \rightarrow \mathbb{R}$ BÚNO prostě

→ chceme kostru T grafu G t.č.

$$w(T) := \sum_{e \in E(T)} w(e) \quad \text{je minimální.}$$

• Jarníkův algoritmus

→ hledací algoritmus

1. $T \leftarrow \{\}$ pro $v \in V$ libovolný

2. Dokud $V(T) \neq V(G)$: \leftarrow n-bráň

3. $e \leftarrow$ nejlehčí hrana mezi T a zbytkem grafu $]_m$ $\Theta(m \cdot m)$

4. $T \leftarrow T + e$

Korektnost:

Lemma: J.a. se rozšiří a na konci je T kostra.

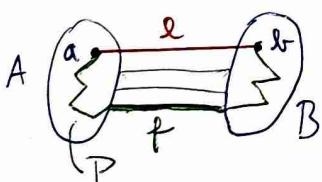
Důkaz: $\& T$ lepíme listy $\Rightarrow T$ je vždy strom + J.a. se rozšiří. Edgě má všechny V .

Def: Elementární řez $\sim G$ je $R \subseteq E(G) \equiv$

\exists rozklad $E(G)$ na $\{A, B\}$ t.č. $R = \{ab \in E(G) \mid a \in A, b \in B\} = E(A, B)$

Převrácené lemma: Bud G graf s unikátními vrátkami, R elementární řez $\sim G$, e nejlehčí hrana R a T jakožkoliv min. kostra G . Potom $e \in T$.

Důkaz: Spolem... T min. kostra, $e \notin T$



\exists cesta $P \sim T$ mezi a, b

$\Rightarrow \exists f \in P \setminus R, w(f) > w(e)$

$\Rightarrow T' := T - f + e$ je také kostra, ale $w(T') < w(T)$ \square

Důsledky:

① J.a. je korektní, protože hrany v každém kroku jsou el. řez \Rightarrow min. $\in T$

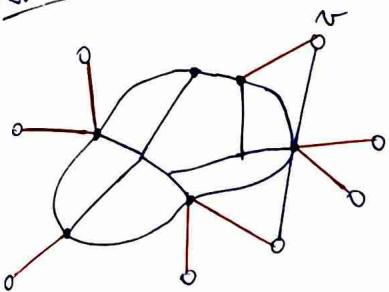
② Pro unikátní váhy je minimální kostra jediná \Rightarrow platí $T \subseteq$ každě min. kostry

③ Minimální kostra je určena porovnáváním vrak, hodnoty nepotřebujeme
 \Rightarrow stačí nám na hranach nadefinovat nějaké lineární uspořádání

\Rightarrow Edgě nemáme unikátní váhy hrani, tak si hrany očíslyujeme

(ale J.a. funguje i pro nemunikátní váhy, jen nefunguje sen důkaz)

Jarník ala Dijstra



- Pro v sousedící s T si pamatují aktívnu hrancu
- $a(v) :=$ nejlehčí hrana mezi v a T
- $h(v) := l(a(v))$... délka aktívnej hrany
- stav(v)
 - raviený \rightarrow vlastní T
 - ostrielený \rightarrow soused T
 - nervidin \rightarrow ostatní

Inicializace...

1. Počud $\exists v$ ostrielený s min. $h(v)$:

2. Pro $\forall vw \in E$, w ostrielené nebo nervidiné:

3. Počud $h(w) > l(vw)$:

4. $a(w) \leftarrow vw$

5. stav(w) \leftarrow ostrielený

6. stav(v) \leftarrow raviéný

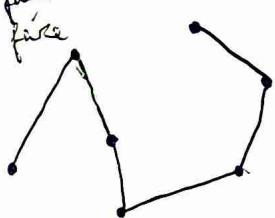
Složitosl:

• pole $O(n^2)$

• halda $O(m \log n)$

Borůvek algoritmus

1. fáze
2. fáze



Incializace: \forall vrchol je samostatný stromeciel

1 fáze: \forall strom si vybere nejlehčí hrancu ven } paralelní J.a.
a tyto hrany všechny přidáme

⊗ B.a. najde min. kostru (\approx riccerovo lemma)

⊗ 1 fáze trvá $O(n+m) \in O(n)$ (G je souvisly) $\rightarrow O(n)$

Implementace: řešíme si stromecily a pro všechny hrany se hodívaíme, moci
střejími dřežma vede \Rightarrow aktualizuj jejich průběžné minimum
 \rightarrow na konci fáze musíme najít mergnout ty stromecily $\rightarrow O(n)$

Lemma: Po k -lé fázi má \forall strom alespoň 2^k vrcholi.

Dle: indukci podle k . $k=0: 1 \geq 2^0$ ✓

$k-1 \rightarrow k:$

$$\geq 2^{k-1} + 2^{k-1} = 2^k \text{ vrcholi}$$

Důsledek: # fází $\leq \log_2(n)$ \because po $\log_2(n)$ -lé fázi má n vrcholi

Výta: Borůvek algoritmus najde minimální kostru v čase $O(m \cdot \log n)$.

Kruskalov algoritmus

$$m \in O(m^2)$$

- 1. seřídime hrany podle vah $\] O(m \log m)$
- 2. $T \leftarrow \{v \mid v \in V\}$... triv. les $\in O(m \log m)$

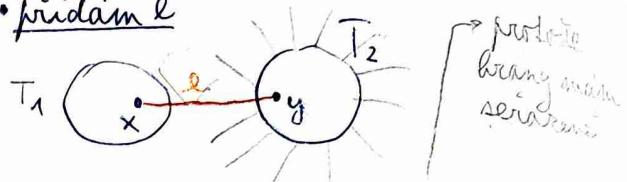
- 3. Ber $e \in E$ od nejlehčí po nejděšší:

- 4. Pokud $T + e$ je acylický: $\text{Find} \] m\text{-hráč}$
- 5. $T \leftarrow T + e$ $\text{Union} \] n\text{-hráč}$

$e = xy \rightarrow$ pokud x, y ve stejné kompl.,
 $T + e$ má cyklus

- Korektnost: z recenzeho lemmatu

- přidám e



$\rightarrow e$ je nejlehčí v řízení mezi T_2 a T_1
 $\Rightarrow e \in \text{min. kostra}$

- nepřidám e \rightarrow došlo k cyklu
 $\hookrightarrow T \subseteq \text{min. k.} \Rightarrow e$ nepotřebuju

Union-Find problem

- $\text{Find}(x, y) =$ jsou x, y v téže komponentě?
- $\text{Union}(x, y) =$ přidej hranu xy

- časová složitost:

$$O(m \cdot \log m + m \cdot T_F + m \cdot T_U)$$

① primitivně

- \rightarrow všechny vrcholy si pamatují ID komponenty
- $\text{Find } O(1), \text{ Union } O(n)$

$$\} O(m \log m + m^2)$$

② rychleji

Komponenta \sim kořík orientovaný do kořene
 \rightarrow všechny vrcholy si pamatují svého rodiče



- Find : vystoupá do kořene a počítá je

$$\} \text{oba } \sim \text{ čase } O(\text{hloubka})$$

- Union : naváže kořen 1 na kořen 2

\rightarrow kořeny si pamatují hloubku

\Rightarrow připojíme měří pod hloubkou

\Rightarrow hloubka vrostlé nejvýše τ^1

 kořík hloubky h má alespoň 2^h vrcholů (indukci)

\Rightarrow hloubka $\leq \log_2 n \Rightarrow U, F \sim \text{ čase } O(\log n)$

Věta: Kruskalov algoritmus najde minimální kostru v čase $O(m \cdot \log n)$.

Další struktury

- abstrakce uložení dat
 - statické - rychlejší, drahý
 - dynamické - náročně i úpravy
- rozhraní - poskytuje operace nad daty
- implementace - jak jsou data uložena, jak prováděné operace

Fronta

	Enqueue	Dequeue	
spojový seznam	$O(1)$	$O(1)$	
pole - lineární	$O(1)$	$O(n)$	\leftarrow read index } pro n prvků
pole - cyklický	$O(1)$	$O(1)$	\leftarrow read, write index } pole délky $\geq n+1$

Množina

→ univerzum prvků \mathcal{U} → DÚNO předpokládejme, že s prvek lze pracovat v $O(1)$

→ chceme reprezentovat $X \subseteq \mathcal{U}$, $n := |X|$

→ operace:

- Find (Member) $\sim a \in X?$
- Insert \sim posad $a \in X$, tak ne
- Delete \sim posad $a \notin X$, tak ne
- Build

Sestava	Pole	Seřiditelné pole	Vyhledávací strom
$\Theta(n)$	$\Theta(n)$	$\Theta(\log n)$	
$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	
$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	
		$\Theta(n \log n)$	$\Theta(\log n)$

↳ fajn statika struktura

• Definovaná množina - píše vyhl. strom / seřiditelné pole

→ operace náročné: Min, Max, Pred, Succ } $\Theta(\log n)$ → hash. tabulka $\Theta(n)$

Sloumík

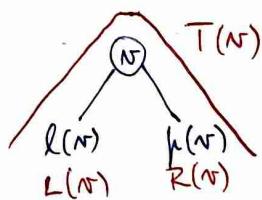
→ univerzum kliců } sloumík := $\{(k, v)\}$

→ univerzum hodnot }

→ Find vrací hodnotu, Insert přidá klic s hodnotou

→ množina se dá snadno přeškolat na sloumík

- Binární strom := rozložený strom, rozlišuje levého a pravého syna



$T(v)$:= podstrom v a všech jeho stranitlivých potomků

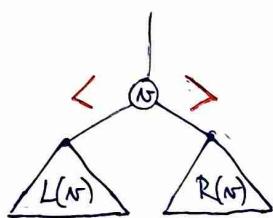
$$L(v) := T(l(v)) \quad \& \quad R(v) := T(r(v))$$

$h(v)$:= hloubka $T(v)$ v kratech

$$\Rightarrow \text{když syn chybí: } \begin{cases} l(v) = \emptyset \Rightarrow L(v) = \emptyset \\ r(v) = \emptyset \Rightarrow R(v) = \emptyset \end{cases} \quad \} \quad h(\emptyset) := -1$$

- Binární vyhledávací strom - BVS

→ klice $\ell(v) \in U$ Podmínka stromu:

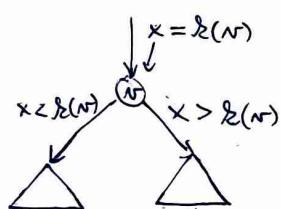


$$\begin{array}{l} \forall v: \begin{array}{l} \forall a \in L(v): \ell(a) < \ell(v) \\ \forall b \in R(v): \ell(b) > \ell(v) \end{array} \end{array} \quad \} \Rightarrow \text{klice jsou unikátní}$$

→ inorder $(L(v), v, R(v))$ průchod vypíše klice v rostoucím pořadí

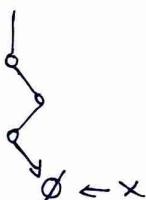
Operace:

- Find(x)



- 1) majdu $x \Rightarrow \checkmark$
- 2) majdu $\emptyset \Rightarrow x \notin T(v)$

- Insert(x)

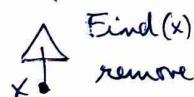


Find(x)

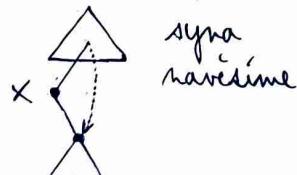
- 1) majdu $x \Rightarrow$ ne nedělám
- 2) majdu $\emptyset \Rightarrow$ rozložím nový list

- Delete(x)

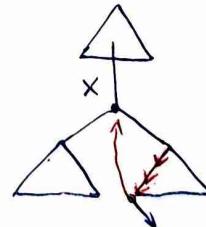
① x je list



② x má 1 syna



③ x má 2 syny



nebo max($P(x)$)

majdu min($P(x)$) → jdu doleva
 $\Rightarrow x$ nahradím sítim $m := \min(P(x))$
 $\Rightarrow \text{Delete}(m) \Rightarrow r(m) = \emptyset \Rightarrow \text{①}$
 $\hookrightarrow r(m) \neq \emptyset \Rightarrow \text{②}$

• složitost

→ složitost všech operací je $\Theta(h)$ (hloubka stromu)

⇒ chceme vyvážené stromy $h = \log(n)$

⇒ potom všechno $\Theta(\log n)$

degenerovaný
strom

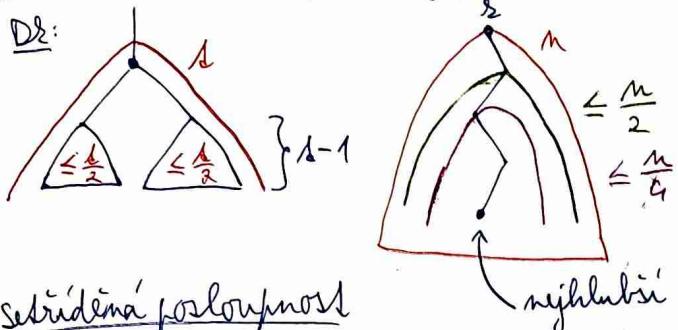


Dobrý vyvážený strom

Def: pro $\forall v$ $|L(v) - R(v)| \leq 1$.

Tvrdíme: Dobrý vyvážený DVS na n vrcholech má hloubku $\leq \log_2(n)$.

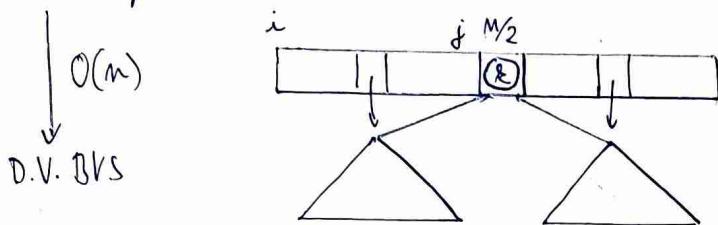
Dk:



délka cesty z nejhlubšího listu je logaritmická

$$\Rightarrow h \leq \log_2(n)$$

seřízená posloupnost



\Rightarrow jedo kořen se stojí prostřední prvek
 \Rightarrow rekurevní fce Build(i, j):

1. majdi střed $\rightarrow k$
2. $L(k) \leftarrow \text{Build}(i, k-1)$
3. $R(k) \leftarrow \text{Build}(k+1, j)$
4. return k

\Rightarrow D.V. DVS lze vyhodit \Rightarrow vždy existuje

\rightarrow Insert a Delete nejdou lepí než lineárně

\rightarrow je to dobrá statická, ale řešení dynamická datová struktura

AVL strom

Def: pro $\forall v$: $|h(l(v)) - h(r(v))| \leq 1$.

Příklady:

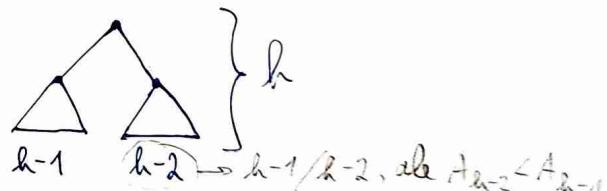


Věta: AVL strom na n vrcholech má hloubku $\mathcal{O}(\log n)$.

Dk: $A_h := \min. \# \text{vrcholů pro hloubku } h$.

$$A_0 = 1 \\ A_1 = 2 \\ \vdots$$

pro $h > 1$ uvažíme minimální strom ($\approx \lfloor V(T) \rfloor$) s hloubkou h



$$\Rightarrow A_h = 1 + A_{h-1} + A_{h-2} \quad \sim \text{Fibonacci}$$

$$\text{Tvrzení: } A_h \geq 2^{h/2} = \sqrt{2}^h$$

$$\text{Dk: indukce podle } h: h-1 \rightarrow h: A_h = 1 + A_{h-1} + A_{h-2} \geq \sqrt{2}^{h-1} + \sqrt{2}^{h-2} = \sqrt{2}^h \left(\frac{1}{\sqrt{2}} + \frac{1}{2} \right) > \sqrt{2}^h$$

$$\text{Dobídek: } A_h \geq C^h, C = \sqrt{2}$$

$$\Rightarrow \text{hloubka } h \leq \log_C(A_h) \leq \lg_C(n) \Rightarrow h \in O(\log n)$$

$$B_h := \max. \# \text{vrcholů pro hloubku } h$$

$$B_0 = 1 \\ B_1 = 3$$

$B_h = 2 \cdot B_{h-1} + 1 = 2^{h+1} - 1$

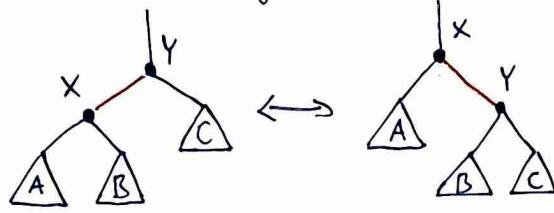
$$\Rightarrow \log(B_h + 1) - 1 = h \Rightarrow h > \log(n) - 1 \Rightarrow h \in \Omega(\log n)$$

$$h \in (\Omega)(\log n)$$



Vyvážování AVL stromu

① rotace hrany



1. Přehodí X a Y

2. Přiřadí jim A, B, C tak, aby bylo zachováno uspoř.

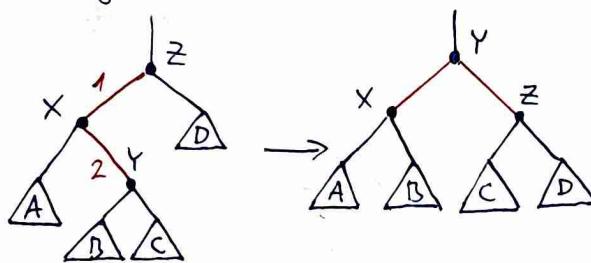
\Rightarrow jednoznačné přiřazení podstromů

$h(A) --$

$h(B)$

$h(C)++$

② dvojitá rotace



1. Dej Y do složené

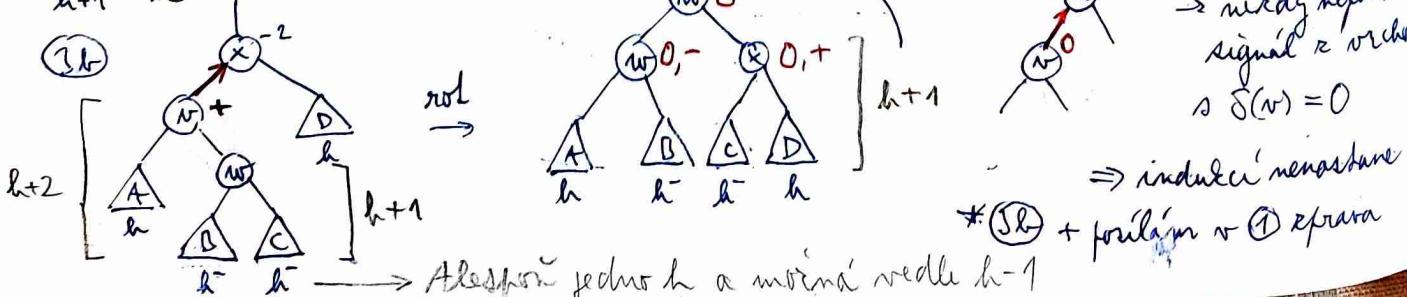
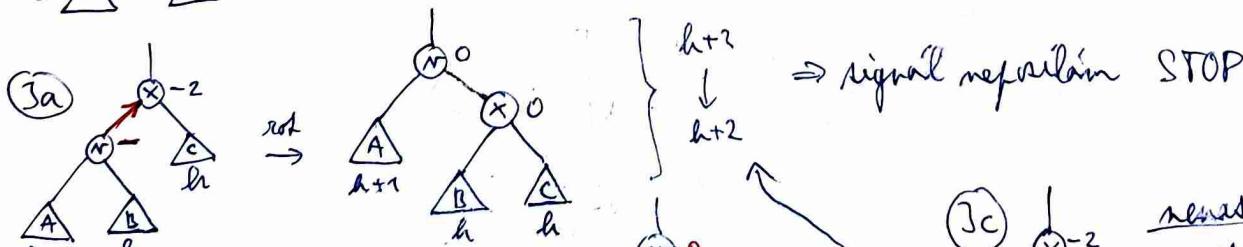
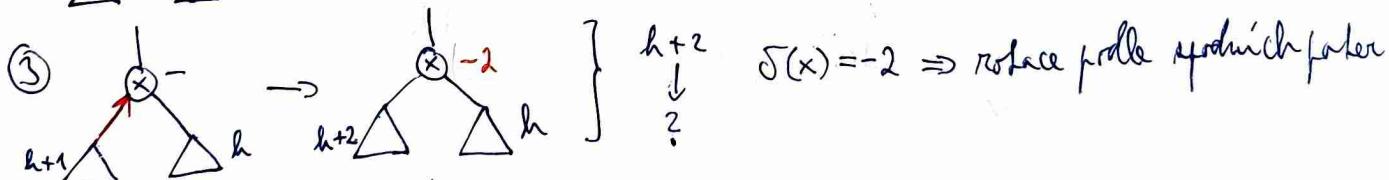
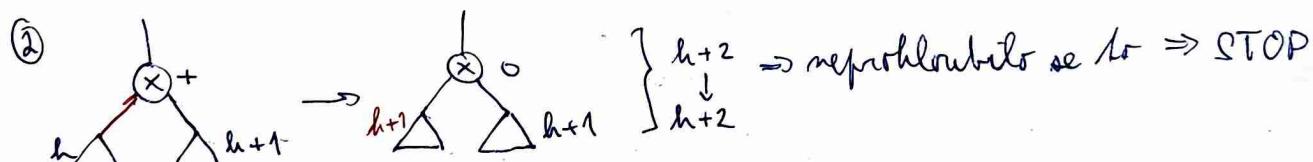
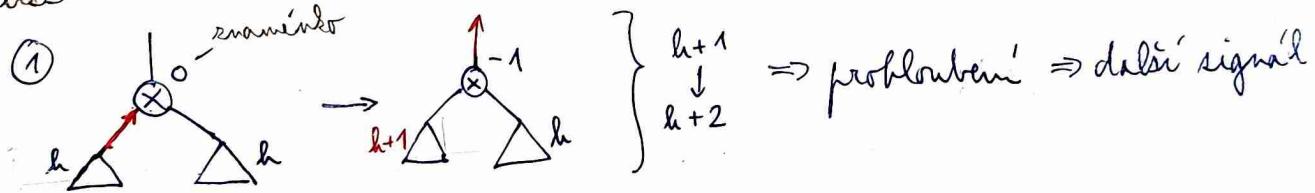
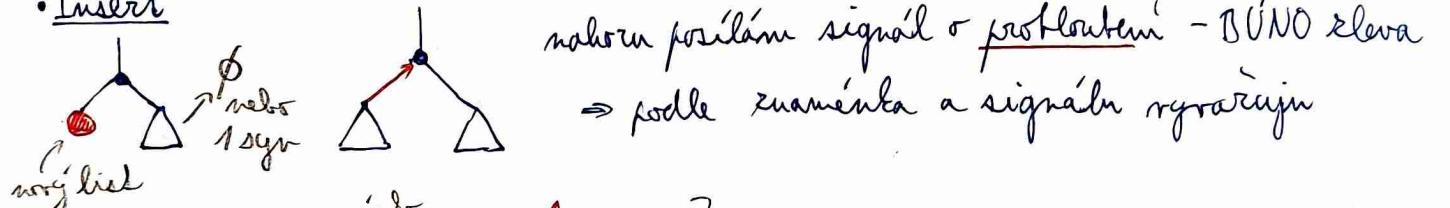
2. Jednoznačně přiřadí A, B, C, D vrcholům X, Y, Z

\Rightarrow dát se složit k rotaci hrany 1 a pak 2
 \Rightarrow tak se to programuje

Vyvážování při operacích

Def: Znaménko vrcholu v je $\delta(v) := h(r(v)) - h(l(v))$. $\Rightarrow \delta(v) \in \{+1, 0, -1\}$

• Insert

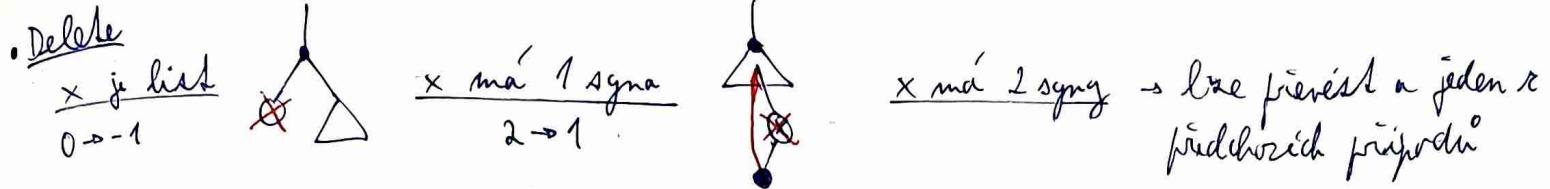


nenastane
 \Rightarrow nikdy nepřijím signál k vrcholu s $\delta(v) = 0$

$* (3b) + posláním v ① efektu$

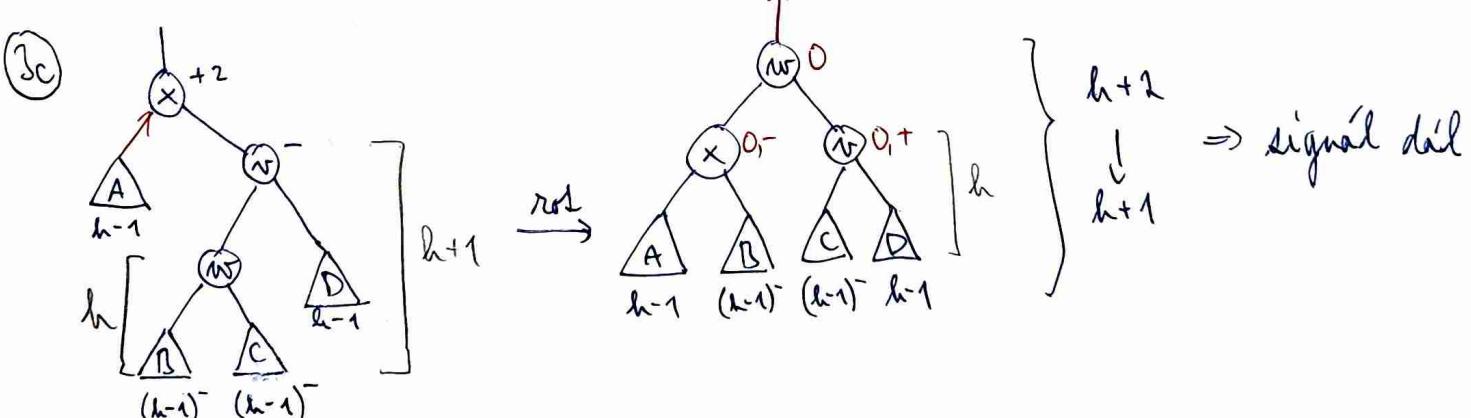
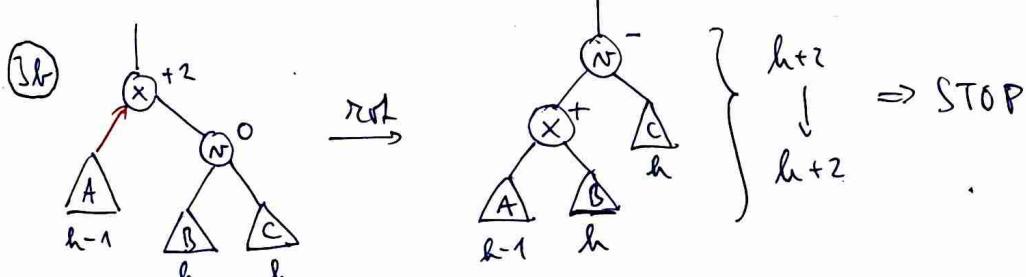
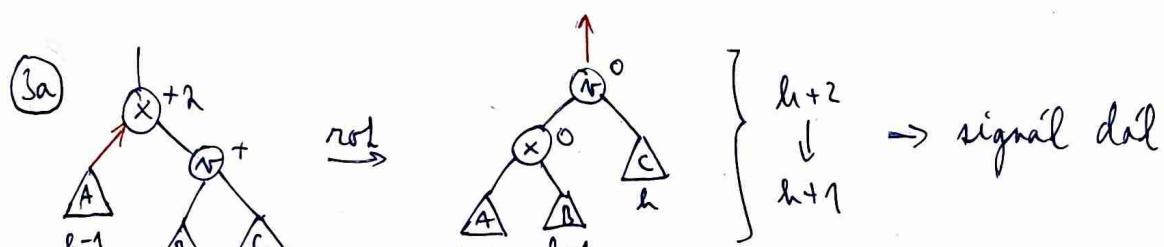
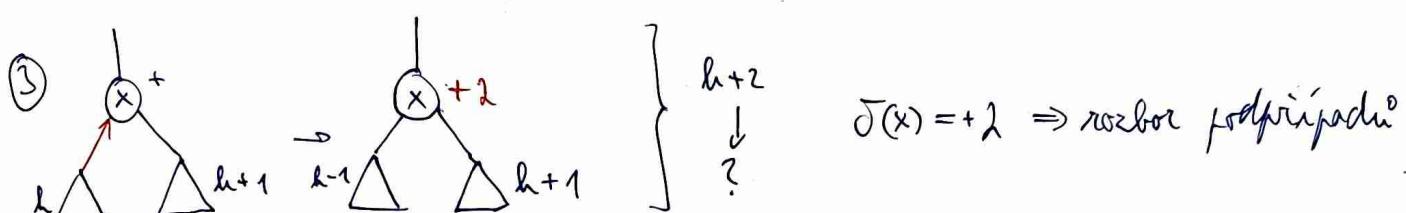
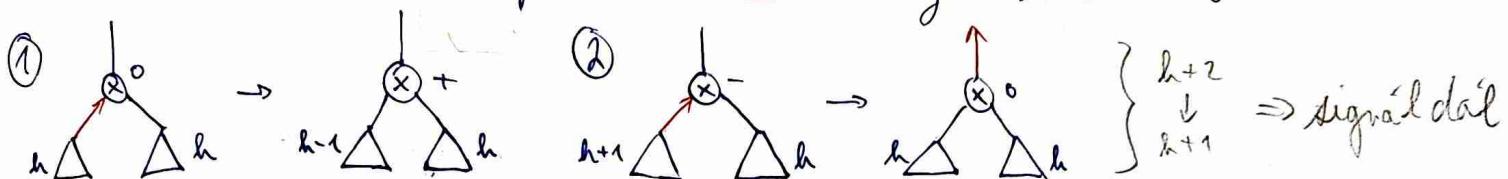
\Rightarrow indukci' nenastane

Alespoň jedno h a minima nedeje h-1



\Rightarrow vždy se sníží hloubka nejdelšího podstromu

\Rightarrow nahoru posílám signál o snížení hloubky - BÍNO zleva

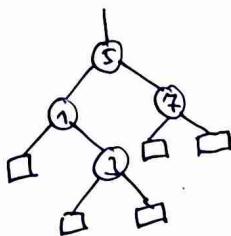


\Rightarrow na Insert délka nejvýše 1 rotace, na Delete může i více, ale v obou případech srovnána všech hodinách konstantní dobou

Věta: Find, Insert, Delete \sim AVL stromu mají časovou složitost $\underline{\Theta}(\log n)$.

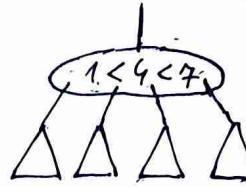
• (a, b) -stromy

• Externí vrcholy



- každý vrchol má buď syny, nebo ho mají bratří
- vlastní null-pointery false v programu

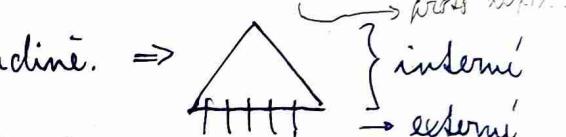
• Vícecestné vyhledávací stromy



- & kliců $\Rightarrow k+1$ podstromů
- dleto sloví jde o oddilování pro dleto podstromů
- operace podobně jako na BST, jen máme konstantně kliců → možná najít podstrom

Def: (a, b) -strom pro $a \geq 2$, $b \geq 2a - 1$ je vícecestný VS s externími vrcholy t.j.

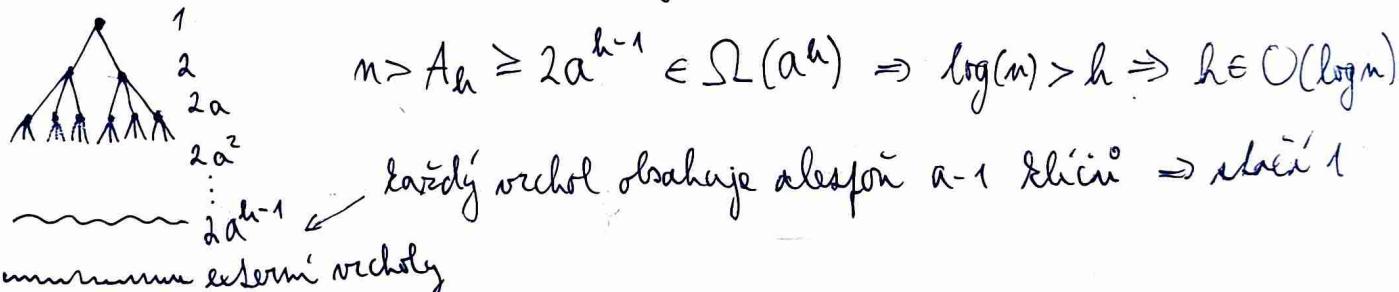
- ① # synů každého interního vrcholu $\in [a, b]$ \Rightarrow # kliců $\in [a-1, b-1]$
synů korene $\in [2, b]$.
- ② všechny externí vrcholy leží na stejné hladině. \Rightarrow



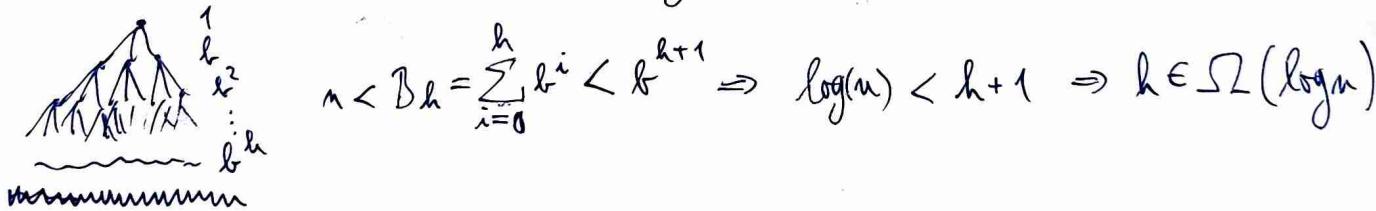
Lemma: Hloubka (a, b) -stromu je $\Theta(\log n)$. \Rightarrow rávni na a, b

Def: Stejný princip jako u AVL stromů \Rightarrow užívá, že # kliců roste exp. s hloubkou

$$A_h := \min \# \text{ kliců ve stromu hloubky } h$$



$$B_h := \max \# \text{ kliců ve stromu hloubky } h$$



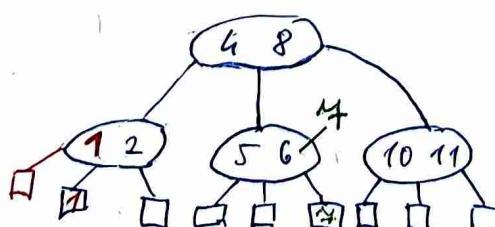
• Find

$O(1)$ čas na hladinu - konstantní kliců \Rightarrow celkově $\Theta(\log n)$

• Bridžad. Insertu

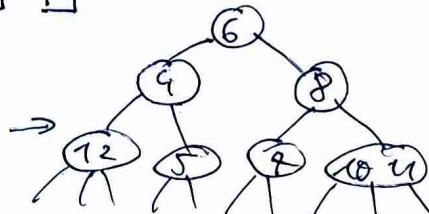
$(2, 3)$ -strom

kliců 1 nebo 2



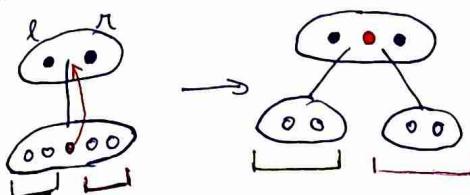
Insert(1)

Insert(7)



Edgě klicé přesízen, dal prostřední vrchol vytlačen nahoru

• Insert



→ v řadě to ráze mohlo překonat
⇒ počítacímu náhorně ⇒ může rozměr i koreni

↳ když elice překonat ⇒ b elici ⇒ 1 nahoru

⇒ může vzniknout vrchol s
méně než $a-1$ dělici?

⇒ kdyby se počalo, tak $\lfloor \frac{b-1}{2} \rfloor < a-1 \Rightarrow \frac{b-1}{2} < a-1 \Rightarrow b < 2a-1$

⇒ proto $b \geq 2a-1$

→ na každou hladinu trávíme konstantní čas → $\Theta(\log n)$

• Delete

① ~~x nemá poslední int. hladinu~~

→ najdu minimum z $R(x)$

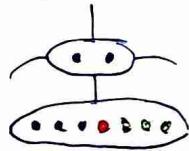
⇒ nahradím ⇒ převedu na ②



②a soused má $a-1$ dělici

⇒ můžeme merge + nejméně střední vrchol

→ proto bych měl 2 pořadiny vedle sebe



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

→ když pořadí otec, tak ráze využij jeho souseda ⇒ znova ②a nebo ②b

→ tehdy se můžem dostat až do kořene

→ pořadí kořena seberu poslední elici, tak se mi Strom smířil

→ proto # synů kořene může být až 2

②b soused má více než $a-1$ dělici

$$a \geq 2$$

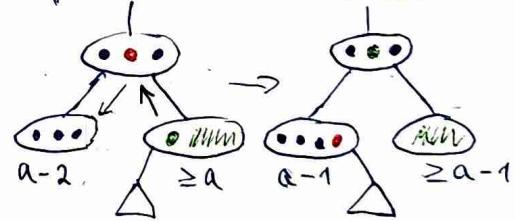
→ pořadí vrchol nepodstaví ✓

→ pořadí podstaví, tak využijeme souseda



②b soused má více než $a-1$ dělici

→ jeden dělici mu seberu



• Červeno-černé stromy

- další varianta vyhledávacího stromu

→ vlastní jde o kódování (2,4)-stromu do BVS

$$\text{Hodí se } b=2a$$

Větší a, b:

→ 1 vrchol se může nejít do
1 bloku cache \rightarrow selije formaly

→ 1 vrchol do 1 seběhru na disku
 \rightarrow souběžný cache + velmi malé

• Řetězce

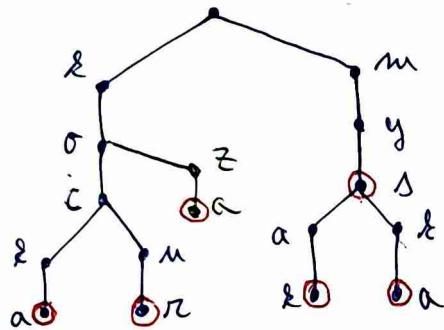
Řetězce nad abecedou Σ $\left\{ \begin{array}{l} \{0,1\} \\ \{A, \dots, Z\} \end{array} \right.$
Unicode ... ~1M znaků

x_1, \dots, x_n řetězce délky L

\Rightarrow BVS: operace \sim čase $\Theta(L \cdot \log n)$ \rightarrow používání dvoř řetězci je $\Theta(L)$

• Písmenkové stromy = Prefixní stromy = Trie

$\{kota, kočka, myš, myšák, myška\}$ Insert($kota$)



\rightarrow když dojdou k kořenu do vrcholu, tak jsem přišel ke nejdelšímu prefixu

\Rightarrow vrcholy Trie \approx prefixy slov ve slovnici

$\Rightarrow \# \text{vrcholů} \in O(\text{součet délek slov})$

\Rightarrow vrcholy obsahují směry "dovec slova" \rightarrow rozhodování při Findu

+ pole uvedeného na sygy indexované abecedou \Rightarrow každý vrchol má $|\Sigma|$ synů

• Find $\Theta(L)$ \rightarrow maximální délka L hledání

• Insert \rightarrow užití Find

1, cílové slovo je v Trie \Rightarrow přidám znak

2, cílové slovo je jen prefix \Rightarrow prostřednictvím vrcholu a následně znak $\} \Theta(L)$

• Delete \rightarrow jde o Find najdu směr

\Rightarrow rekurezivně: pokud jsem ve vrcholu co nemá ani znaků ani synů, tak ho smařu \Rightarrow jdu do otce $\} \Theta(L)$

• Závislost na velikosti abecedy

Find $\rightarrow \Theta(L)$

Insert $\rightarrow \Theta(|\Sigma| \cdot L)$ \rightarrow zahádá pole $|\Sigma|$ null-pointrov

Delete $\rightarrow \Theta(|\Sigma| \cdot L)$ \rightarrow mazí pole pointrov + kontroluje všechny syny

• Trie \rightarrow BVS ve vrcholech

\rightarrow ve vrcholech bude mít sloužit ve svazku $\{\Sigma\text{malá abeceda: pointer na vrchol}\}$

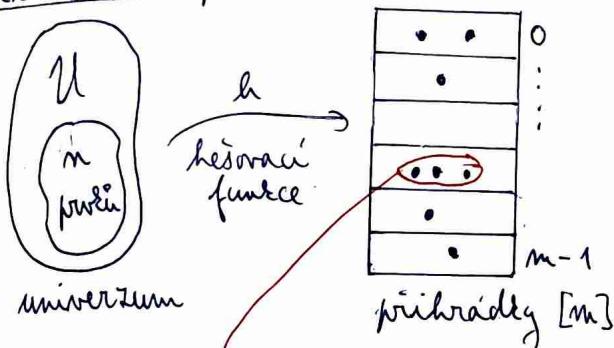
\Rightarrow řeší problém velké abecedy

\Rightarrow všechny operace jsou $\Theta(L \cdot \log |\Sigma|)$

\hookrightarrow až $|\Sigma|$ prvek

- číslicový strom = Radix tree
 - čísla uložená jako řetězce do stromu
 - základ soustavy: Z
 - číslo $\in [0, M]$ má nejvíce $\log_2 M + 1$ číslíků
 - \Rightarrow Find, Insert, Delete v čase $\Theta(\log_2 M)$
- asymptoticky si nevinníme
ale herce konstanty

Hesování s přibrádkami a řetězci



cháeme: h je rozměrná, ale deterministická

$$\Rightarrow \# prvních v přibrádce \sim \frac{m}{m}$$

$$\Rightarrow pro m \in \Theta(n) je h \sim O(1)$$

$$\Rightarrow \text{Find, Ins, Del v čase } O(1)$$

?

kolize \Rightarrow pro každou přibrádku si pamatují seznam jejich prvních elementů
 \Rightarrow Hesování s řetězci - k tomu seznamu se říká řetězec

\rightarrow když $|U| = \infty$, tak \exists přibrádka, do které se zobrazí ∞ prvních

\Rightarrow minimálně několik množin, které se celé zobrazí do jedné přibrádky

Def: Systém h funkcií z U do $[m]$ je c-univerzální pro $c > 0 \equiv$

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

\rightarrow kdybych měl uplně náhodnou fci, tak by to bylo $\frac{1}{m}$. (y má m možnosti)

Věta: Pro $x_1, \dots, x_m, y \in U$ mazajíme různé a h c-univerzální z U do $[m]$:

$$E_{h \in H} [\#\{i : h(x_i) = h(y)\}] \leq \frac{c \cdot m}{m}. \quad \leftarrow nevolím uplně náhodnou fci, ale náhodnou z toho mnoha univerzálních H$$

Dk: Přes linearitu E a indikátory.

$$I_j := \begin{cases} 1 & \text{pokud } h(x_j) = h(y) \\ 0 & \text{jinak} \end{cases} \Rightarrow E(I_j) = 0 + 1 \cdot \Pr[h(x_j) = h(y)] \leq \frac{c}{m}$$

$$\#\text{kolizí} = \sum_j I_j \Rightarrow E[\#\text{kolizí}] = E\left(\sum_j I_j\right) = \sum_j E(I_j) \leq m \cdot \frac{c}{m} \quad \blacksquare$$

Princip: Nezajíma mě řešitost vstupních dat, protože pro dobré zvolený systém hesovacích funkcí h a k něj náhodně vybranou funkcí h bude amortizovaná složitost $O(c \cdot \frac{m}{m}) = O(c) = O(1)$

Konstrukce systému hesoracích funkcí

- zvolíme nejalejší konečné těleso $\mathbb{Z}_p \Rightarrow m=p$ příhrádek $0, \dots, p-1$
 $U = \mathbb{Z}_p^d \rightarrow d$ -složkové vektory nad $\mathbb{Z}_p \quad (p \in \mathbb{P})$

\Rightarrow hesorací funkce bude $h_1(x) = 1 \cdot x$ - skalární součin s nějakým $1 \in \mathbb{Z}_p^d$

Věta: Systém funkcí $H = \{h_1 \mid 1 \in \mathbb{Z}_p^d\}$ je 1-univerzální.

Důkaz: Chceme: $\forall x, y \in \mathbb{Z}_p^d : \Pr_{1 \in \mathbb{Z}_p^d} [h_1(x) = h_1(y)] \leq \frac{1}{m} = \frac{1}{p}$.

\Rightarrow Máme dva různé vektory x, y , nechť \mathbf{x} je souřadnice v místě $x_i \neq y_i$

\Rightarrow skalární součin nezávislý na pořadí složek \Rightarrow BÚNO $k=d$.

$$\begin{aligned} \Pr_{1 \in \mathbb{Z}_p^d} [x \cdot 1 = y \cdot 1] &= \Pr [(x-y) \cdot 1 = 0] = \Pr \left[\sum_{i=1}^d (x_i - y_i) 1_i = 0 \right] \\ &= \Pr [(x_d - y_d) 1_d = \sum_{i=1}^{d-1} (x_i - y_i) 1_i] = \Pr [a \cdot 1_d = b] \end{aligned}$$

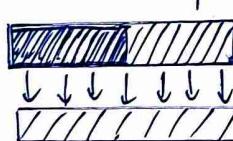
\rightarrow pokud někdo máme náhodně zvolili $1_1, \dots, 1_{d-1}$ a teď náhodně volíme 1_d , tak zde je nastane právě pro jednu volbu x $f=m$ možných

Příklad: Chci hesovat 32-bit čísla do cca 250 příhrádek.

$$p=257 \quad \text{int32: } [88.188.188.188] \quad 2^8=256 \Rightarrow \text{int32} \sim \mathbb{Z}_{257}^4$$

$$123\ 456\ 789 = 7 \cdot 2^8 + 91 \cdot 2^{16} + 205 \cdot 2^{24} + 21 \sim x=(7, 91, 205, 21)$$

Nafukovací pole



\mathfrak{l} ... kapacita pole

n ... # prvek

$$\mathfrak{l}' = 2\mathfrak{l}$$

Edyž $m=2$, tak pole nafukovu na \mathfrak{l}'

řídí $2^{\mathfrak{l}}$
po $2^{\mathfrak{l}+1}$

čas na vložení n prvků celkem?

pocáteční $\mathfrak{l}_0 = 1$

$\mathfrak{l}_1 = 2$

$\mathfrak{l}_2 = 4$

\vdots

$\mathfrak{l}_i = 2^i$

konečná $\mathfrak{l}_{n+1} = 2^{n+1}$

$$\text{realizace: } 1 + 2 + 2^2 + \dots + 2^{\mathfrak{l}} = 2^{\mathfrak{l}+1} - 1$$

$$\underbrace{2^{\mathfrak{l}} < m \leq 2^{\mathfrak{l}+1}}_{2^{\mathfrak{l}+1} \leq 2m} \Rightarrow \underline{2^{\mathfrak{l}+1}} \in \underline{\Theta(m)}$$

$\Rightarrow n$ prvek ... $\Theta(m) \Rightarrow$ amortizovaná pro 1 prvek je $O(1)$

Znělkovací tabulky - stejný princip, jen to musíme přehesovat

\Rightarrow cena na znělkování je průměrně $O(1)$ \Rightarrow vložení n prvků je průměrně $\Theta(n)$

\Rightarrow cena na vložení 1 prvek je průměrně $O(1)$

Rozděl a pánež

Mergesort (a_1, \dots, a_n)

1. Pokud $n \leq 1$ return vstup

2. Mergesort ($a_1, \dots, a_{\lfloor n/2 \rfloor}$) $\rightarrow X_1, \dots, X_{\lfloor n/2 \rfloor}$

3. Mergesort ($a_{\lceil n/2 \rceil}, \dots, a_n$) $\rightarrow Y_1, \dots, Y_{\lceil n/2 \rceil}$

4. return Merge (X, Y) $\leftarrow \Theta(n)$



Merge: $O(s+l)$

analýza substitucí $T(n) :=$ čas pro vstup délky n

$$T(1) = 1$$

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + C \cdot n$$

$$T(n) = 2 \cdot \left(2 \cdot T\left(\frac{n}{4}\right) + C \cdot \frac{n}{2}\right) + C \cdot n = 4 \cdot T\left(\frac{n}{4}\right) + 2 \cdot C \cdot n$$

$$T(n) = 2^i T\left(\frac{n}{2^i}\right) + i \cdot C \cdot n, \text{ kde } T\left(\frac{n}{2^i}\right) = 1 \Rightarrow i = \log_2 n$$

$$\Rightarrow T(n) = n \cdot T(1) + C \cdot n \log_2 n, \text{ pro } i = \log_2 n$$

$$\Rightarrow T(n) = n + C \cdot n \log_2 n \in \Theta(n \log n)$$

rekurzivní předpokládání
 $n = 2^k$

Pánež: sčítáme v $\Theta(n)$ + já'a 1 z podproblémů si pamatujeme vstup

$$S(n) = d \cdot n + S\left(\frac{n}{2}\right) \rightarrow \text{já'a} + \text{podproblem}$$

$$S(1) = 1$$

$$\Rightarrow S(n) = d \cdot n + d \cdot \frac{n}{2} + S\left(\frac{n}{4}\right) = dn + d \cdot \frac{n}{2} + d \cdot \frac{n}{4} + \dots \leq 2dn \in \Theta(n)$$

nejde se tam i sčítání

analýza stromem rekurze

vel. fp.	#fp.	merge		čas na 1 fp	čas na hladinu
		čas na 1 fp	čas na hladinu		
m	1	.	$m = m$		
$m/2$	2	.	$m/2 = m$		
$m/4$	4	.	$m/4 = m$		
\vdots	\vdots		\vdots		
$m/2^i$	2^i	.	$m/2^i = m$		
listy nmmm	1	n	$1 = m$		

čas na 1 hladinu: n
hladin: $\log_2 n$
 $\Rightarrow \Theta(n \log n)$

pánež: každý ještě v nejdelejším podproblému, tak si pamatují všechny jeho předky

$$\Rightarrow \text{závorbík: } n + \frac{n}{2} + \frac{n}{4} + \dots \leq 2n \in \Theta(n)$$

• Násobení velkých čísel - Karacubaov algoritmus $\ll_{10} n$

$$\begin{array}{c} m=2^e \\ \times \quad \boxed{A \quad B} \quad X = A \cdot 10^{m/2} + B \\ Y \quad \boxed{C \quad D} \quad Y = C \cdot 10^{m/2} + D \end{array} \quad \left. \begin{array}{l} X \cdot Y = AC \cdot 10^m + AD \cdot 10^{m/2} + BC \cdot 10^{m/2} + BD \\ \hookrightarrow \oplus a \cdot 10^m \in \Theta(n) \end{array} \right\}$$

$$\Rightarrow T(n) = 4 \cdot T\left(\frac{n}{2}\right) + C \cdot n$$

vel. fp.	# fp.	čas na hladinu
m	1	
$m/2$	4	
:	:	
$m/2^i$	4^i	$n \cdot 2^i$
1	$4^{\log_2 m} = m^2$	

normalní násobení je n^2
 \Rightarrow tříle je špatné

⇒ stáčí nám 3 násobení: $(A+B)(C+D) = AC + AD + BC + BD$

$$\begin{aligned} X \cdot Y &= AC \cdot 10^m + (AD + BC) \cdot 10^{m/2} + BD \\ &= AC \cdot 10^m + [(A+B)(C+D) - AC - BD] \cdot 10^{m/2} + BD \end{aligned}$$

$$\Rightarrow \text{čas na hladinu} \in \Theta\left(\frac{m}{2^i} 3^i\right) = \Theta\left(m \left(\frac{3}{2}\right)^i\right)$$

$$\sum_{i=0}^k q^i = \frac{q^{k+1}-1}{q-1} \in \Theta(q^k)$$

$$\Rightarrow T(n) \stackrel{\Theta}{=} \sum_{i=1}^{\log_2 m} m \left(\frac{3}{2}\right)^i \stackrel{\Theta}{=} m \cdot \left(\frac{3}{2}\right)^{\log_2 m} = 3^{\log_2 m} = 2^{\log_2 3} \cdot \log_2 m = n^{\log_2 3}$$

Věta: Karacubaov algoritmus běží v čase $\Theta(n^{\log_2 3}) = \Theta(n^{1.58})$

• Obecně

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

předpokládáme $n = b^e$

$$\# \text{ hladin} = \log_b n$$

$$\text{vel. fp.} = n/b^i$$

$$\# \text{ fp.} = a^i$$

$$\left. \begin{array}{l} \text{čas na fp.} = \left(\frac{n}{b^i}\right)^c \\ \text{čas na hladinu} = \left(\frac{n}{b^i}\right)^c a^i = n^c \left(\frac{a}{b^c}\right)^i \end{array} \right\}$$

$$\Rightarrow T(n) = \sum_{i=1}^{\log_b n} n^c \left(\frac{a}{b^c}\right)^i = n^c \sum_{i=1}^{\log_b n} q^i ; \quad q = \frac{a}{b^c}$$

prí posunu v hladinu
 dole se práce
 směří q -září

• $q < 1: \sum \in O(1) \Rightarrow \Theta(n^c) = \text{práce v rovině}$

• $q = 1: \sum = \log_b n \Rightarrow \Theta(n^c \cdot \log n) = \text{práce na bl.} \cdot \# \text{ hladin}$

• $q > 1: \sum \in \Theta\left(\left(\frac{a}{b^c}\right) \log_b n\right) \stackrel{\Theta}{=} \frac{a \log_b n}{n^c} = \frac{1}{n^c} b^{\log_b a} \log_b n = \frac{1}{n^c} n^{\log_b a} \Rightarrow \Theta(n^{\log_b a})$

\rightarrow co bylo $m \neq b^k$?

\rightarrow speciální m : $m^- \leq m \leq m^+$ ← nejbližší mocnina b

$$\textcircled{1} \quad T(m^-) \leq T(m) \leq T(m^+)$$

\Rightarrow indukce od boku do koreňe - pro listy $T(m^-) = T(m^+) = T(1) = 1 \checkmark$

$$T(m) = a \cdot T\left(\frac{m}{b}\right) + \textcircled{H}(m^c) \quad \leftarrow T(m) \text{ je monotonní}$$

\Rightarrow počet řešení pro podproblemy, tak i pro rodicovský problem \blacksquare

$$\rightarrow \frac{m^+}{m^-} = b \quad \Rightarrow T(m^-) \in \textcircled{H}(T(m^+)) \Rightarrow T(m) \in \textcircled{H}(T(m^+))$$

Věta (Eukaristová): Rekurentní rovnice $T(m) = a \cdot T\left(\frac{m}{b}\right) + \textcircled{H}(m^c)$, $T(1) = 1$

pro $a \in \mathbb{N}$, $a \geq 1$ = počet podproblemů

$b \in \mathbb{R}^+$, $b > 1$ = faktor rozdělení podproblemů mezi

možné řešení: $c \in \mathbb{R}_0^+$, $c \geq 0$ $\rightarrow m^c =$ práce v podproblemech relativně m

$$\begin{array}{lll} \text{takže } \textcircled{1} \text{ } \textcircled{H}(m^c) & \dots q < 1 & \\ \text{listy } \textcircled{2} \text{ } \textcircled{H}(m^{\log_b a}) & \dots q > 1 & \left\{ \begin{array}{l} \text{pro } q = \frac{a}{b^c} \\ \text{stejně } \textcircled{3} \text{ } \textcircled{H}(m^c \log m) \dots q = 1 \end{array} \right. \\ \text{stejně } \textcircled{3} \text{ } \textcircled{H}(m^c \log m) \dots q = 1 & & m^c \rightarrow a \cdot \left(\frac{m}{b}\right)^c = m^c \cdot \frac{a}{b^c} \end{array}$$

Příklady:

$$\text{mergesort: } a = 2, b = 2, c = 1 \dots q = \frac{2}{2} = 1 \Rightarrow m \cdot \log m$$

$$\text{násobení: } a = 4, b = 2, c = 1 \dots q = \frac{4}{2} = 2 \Rightarrow m^2$$

$$a = 3, b = 2, c = 1 \dots q = \frac{3}{2} \Rightarrow m^{\log_2 3}$$

$$\text{bin-search: } a = 1, b = 2, c = 0 \dots q = \frac{1}{1} = 1 \Rightarrow 1 \cdot \log m$$

Násobení matic - Strassenův alg.

$n \times n \rightarrow n = 2^k$, jinak lze dátého oddělení z nul

$$\frac{n}{2} \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right) \cdot \left(\begin{array}{c|c} P & Q \\ \hline R & S \end{array} \right) = \left(\begin{array}{c|c} AP + BR & AQ + BS \\ \hline CP + DR & CQ + DS \end{array} \right) \rightarrow 8 \text{ násobení } \left(\frac{n}{2} \right) \times \left(\frac{n}{2} \right) + \text{ sčítání, sčítání, ...} \in \textcircled{H}(n^2)$$

$$\text{násobení matic: } a = 8, b = 2, c = 2 \dots q = \frac{8}{4} = 2 \Rightarrow n^3 \sim jako \text{ z definice}$$

$$\text{Strassen: } a = 7, b = 2, c = 2 \dots q = \frac{7}{4} \Rightarrow n^{\log_2 7} \approx n^{2.807}$$

\hookrightarrow Strassenovy maticy \rightarrow spočítat se 7 součinů $\Rightarrow \textcircled{H}(n^{\log_2 7})$

\rightarrow Ačkoliv je super, protože správnou problémů lze

převést na násobení matic

• Quickselct

→ alg. na k -tg nejménší prvek \Rightarrow median $\sim \lfloor \frac{m}{2} \rfloor$

Def: m je median \equiv nejvýš $\lfloor \frac{m}{2} \rfloor$ prvků jsou menších & nejvýš $\lfloor \frac{m}{2} \rfloor$ je větších

x_1	\dots	x_m
$L < p$	$S = p$	$P > p$
$p = x_i \dots$ pivot		

pokud $\ell \leq |L| \rightarrow$ hledán v L

$|L| < \ell \leq |L| + |S| \rightarrow$ pivot je následek

$|L| + |S| < \ell \rightarrow$ hledán v P

Quickselct (X, ℓ):

1. vybereme pivot $p \in X$

2. rozdilíme X na L, S, P podle p

3. Pokud $\ell \leq |L|: \text{return Quickselct}(L, \ell)$

$|L| < \ell \leq |L| + |S|: \text{return } p$

$|L| + |S| < \ell: \text{return Quickselct}(P, \ell - |L| - |S|)$

① Pivot = median: $|L|, |P| \leq \frac{m}{2} \Rightarrow T(m) \in \Theta\left(m + \frac{m}{2} + \frac{m}{4} + \dots\right) \in \Theta(m)$

② Pivot = maximum: $T(m) \in \Theta(m + m-1 + m-2 + \dots) \in \Theta(m^2)$

③ Pivot = skoromedian
 $T(m) \in \Theta\left(m + \frac{3}{4}m + \left(\frac{3}{4}\right)^2m + \dots\right) \in \Theta(m)$


 mimo P střed m. mimo L \Rightarrow vždy zahrádím alefou $\frac{m}{4}$ prvků

• Randomizovang RAM - má instrukci pro random čísla

$x \leftarrow \text{random}(y) \dots x \in [y]$ rovnomerne náhodné

\rightarrow různé výpočty \sim jevy

$\Rightarrow T(n)$ je náhodná veličina $\Rightarrow \mathbb{E}[T(n)] = ? \quad P[T(n) > 2^n] = ?$

• Hledání skoromediana

\rightarrow opakovane vyberame $p \in X$, dleud to nem' skoromedian \rightarrow kontrola $\Theta(n)$

Lemma (O džbánu): Nechť přes nepije s pravd. $p > 0$, pak

$$\mathbb{E}[\#\text{opakování do 1. úspěchu}] = \frac{1}{p}$$

$\rightarrow P[p \text{ je skoromedian}] \geq \frac{1}{2} \rightarrow \mathbb{E}[\#\text{pokusů}] \leq 2 \quad \left. \begin{array}{l} \mathbb{E}[T(n)] \in \Theta(n) \\ \Rightarrow \text{průměrně } 2 \cdot \Theta(n) \in \Theta(n) \end{array} \right\} \mathbb{E}[T(n)] \in \Theta(n)$

③ Pokud skoromedian hledám takhle $\Rightarrow \mathbb{E}[T(\text{Quickselct})] \in \Theta(n)$

④ Pivot = náhodný prvek \rightarrow prvky co nijsem skorom m. nezahrani na rozdíl od ③

\Rightarrow určitě nem' horší než ③ $\Rightarrow \mathbb{E}[T(n)] \in \Theta(n)$

④ analýza pomocí epoch - epocha čonci, když $p = \text{skoromedian}$

$$m = \frac{3}{4}m + \left(\frac{3}{4}\right)^i m$$

$$\mathbb{E}[\#\text{prichodů v epochě}] \leq 2$$

$$O(n) \text{ má } 1 \text{ prichod}$$

$$\left\{ \begin{array}{l} \mathbb{E}[T(\text{epoch})] \in \Theta(n) \\ \mathbb{E}[T(n)] \in \Theta(n) \end{array} \right.$$

$$\rightarrow T(n) = \sum T(\text{epoch}) = \sum \left(\Theta\left(\left(\frac{3}{4}\right)^i \cdot n\right) \right) \in \Theta(n) \leftarrow \text{v prvním}\right.$$

Quicksort (x)

0. Pokud $|X| \leq 1$: return X

1. vybereme pivot $p \in X$

2. rozdělíme X na L, S, P podle p

3. $L' \leftarrow \text{Quicksort}(L)$

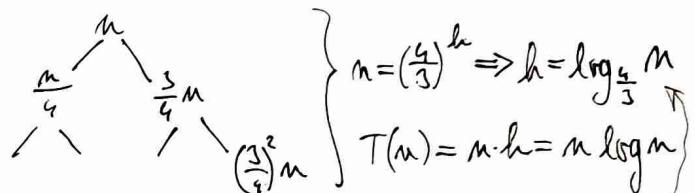
$P' \leftarrow \text{Quicksort}(P)$

4. return $\boxed{L' | S | P'}$

① p je medián $\Rightarrow \Theta(n \log n)$

② p je maximum $\Rightarrow \Theta(n^2)$

③ p je skoromedian $\Rightarrow \Theta(n \log n)$



p je náhodný pivot

$T_i := \# \text{porovnání, kterých se číslo} x_i \text{ (když nebyl pivotem)} \sim \text{čas}$

$$T(n) = \sum T_i \Rightarrow \mathbb{E}[T(n)] = \sum_i \mathbb{E}[T_i]$$

$\rightarrow T_i: \approx 1$ podproblemu 1 porovnání (společně)

\rightarrow epochy jde o Quicksort $\Rightarrow \mathbb{E}[\#\text{pf v epochě}] \leq 2$

(epocha čonci skoromedianem $\Rightarrow \#\text{epoch} \leq \log_{\frac{4}{3}} n \in \Theta(\log n)$)

$$\Rightarrow \mathbb{E}[T_i] \in 2 \cdot \Theta(\log n) \in \Theta(\log n) \Rightarrow \mathbb{E}[T(n)] \in \Theta(n \log n)$$

Linearselect

\rightarrow Quicksort pro náhodné pivoly byl \approx nejhorším případě $\Theta(n^2) \rightarrow$ jde o lineární

\rightarrow rozložím pivky na řetice \rightarrow najdu mediální řetice $\rightarrow p = \text{median těchto mediálních řetice}$

Linearselect (x, k)

0. Pokud $|x| \leq 5$: řešení triviálně

1. pivky rozdělíme na řetice $P_1, \dots, P_{\lfloor \frac{n}{5} \rfloor}$

2. $m_i \leftarrow \text{Linearselect}(P_i, 3)$

3. $p \leftarrow \text{Linearselect}(\{m_1, \dots, m_{\lfloor \frac{n}{5} \rfloor}\}, \lfloor \frac{n}{10} \rfloor)$

4. rozdělíme x na L, S, P podle p

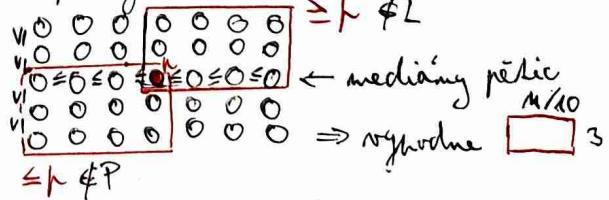
5. Pokud $k \leq |L|$: return $LS(L, k)$

$|L| < k \leq |L| + |S|$: return p

$|L| + |S| < k$: return $LS(P, k - |L| - |S|)$

⊗ vždy náhodně ale srovnatelně 3/10 pravdě

Děl: pivky si uspořádám do řetice:



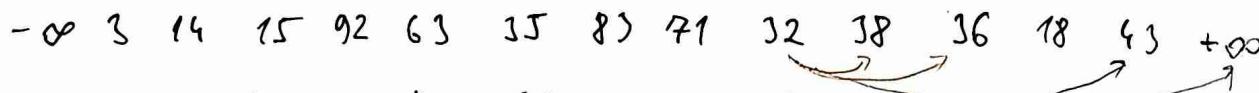
$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + \Theta(n)$$

$\frac{2}{10}n + \frac{4}{10}n \rightarrow \frac{6}{10}n \Rightarrow$ exp. řešení \Rightarrow řešení dominuje

$$\Rightarrow T(n) \in \Theta(n)$$

• Dynamické programování

• Nejdlevnější rozložení podpolohovostí - grafogrý pohled



- hrana vede z menšího čísla do většího
- to určuje topologické uspořádání toho DAGu
- chci nejdlevnější cestu z $-\infty$ do $+\infty \Rightarrow \Theta(n^2)$

• Editační vzdálenost = Lvenschteinova vž.

opec ↑ zmena 1 znaku
 opec → vložení 1 znaku } editační operace
 opec ↓ smazání 1 znaku

$$L(x_1-x_m, y_1-y_m) := \text{délka nejkratší polohovosti ed. op., } \leq \max(m, n) \\ \text{střed } x_1-x_m \text{ přeložit na } y_1-y_m \text{ přepisování.}$$

⊗ L je metrika na množině všech řetězců

→ Odhad fází 1. znak výsledného řetězce?

$$\left. \begin{array}{l} \textcircled{1} \quad x_1 = y_1 : L(x_2-x_m, y_2-y_m) \\ \textcircled{2} \quad \text{změna } x_1 \text{ na } y_1 : 1 + L(x_2-x_m, y_2-y_m) \\ \textcircled{3} \quad \text{vložení } y_1 : 1 + L(x_1-x_m, y_2-y_m) \\ \textcircled{4} \quad \text{smazání } x_1 : 1 + L(x_2-x_m, y_1-y_m) \end{array} \right\} \begin{array}{l} L = \min(\textcircled{1}, \textcircled{2}, \textcircled{3}, \textcircled{4}) \\ L(\emptyset, \emptyset) = 0 \end{array}$$

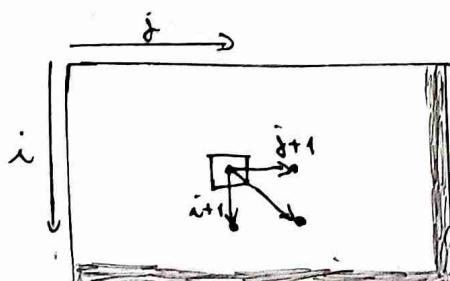
→ když řetězce se nemění ⇒ stačí předávat indexy

$$l(i, j) := L(x_i-x_m, y_j-y_m) \text{ kde } 1 \leq i \leq m+1, 1 \leq j \leq m+1$$

$$l(i, j) = \min \left\{ \begin{array}{l} l(i+1, j+1) \dots x_i = y_j \\ l(i+1, j+1) + 1 \\ l(i, j+1) + 1 \\ l(i+1, j) + 1 \end{array} \right\} \begin{array}{l} L = l(1, 1) \end{array}$$

→ zde je zjevně exponenciální rekurze

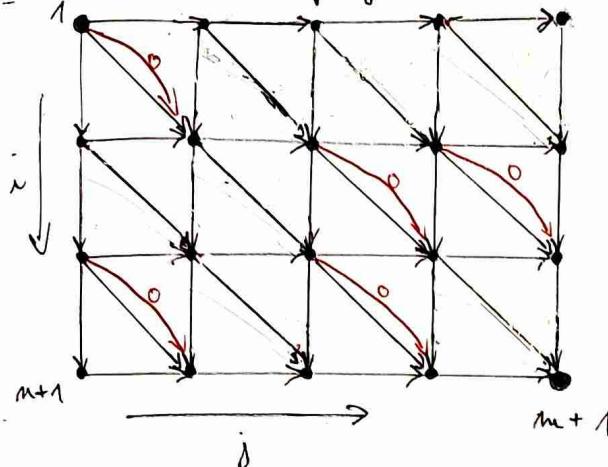
→ celkem $m \cdot m$ možných hodnot $l(i, j) \Rightarrow$ řešování $\Rightarrow \Theta(m \cdot m)$



→ Sobaku vyplňujeme zespoda po řádcích a v řádku výdaje z prava dolera

⇒ nepotřebujeme rekurzi

• Levenshteinův graf



Editační operace ~ cesta po hrani

- vložení x_j
 - ↓ smazání x_i
 - ↘ změna $x_i \neq y_j$
 - ↗ $x_i = x_j$
- } cena 1
} cena 0

$\Rightarrow L$ = nejkratší cesta z $(1, 1)$ do $(m+1, m+1)$

\Rightarrow je $\text{A} \in \text{DAG}$ s $\Theta(m \cdot m)$ vrcholy a hrany

\Rightarrow cestu najdeme top. indukcií v $\Theta(m \cdot m)$

• Floydův-Marshallův algoritmus

vstup: matice délek hrani $L = [m \times m]$ $L_{ij} = \begin{cases} l(N_i, N_j) & N_i, N_j \in E \\ +\infty & N_i, N_j \notin E \end{cases}$

výstup: matice vzdáleností $D = [m \times m]$ $D_{ij} = d(N_i, N_j)$ nebo $+\infty$

$\Rightarrow D_{ij}^k =$ délka nejkratší cesty z N_i do N_j přes N_1, \dots, N_k

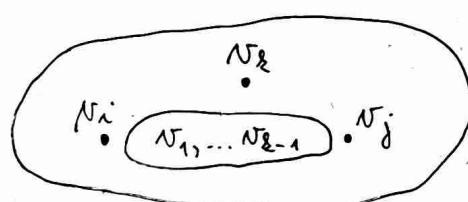
$$D^0 = L$$

$$D^m = D$$

$$D^{k-1} \rightarrow D^k ?$$

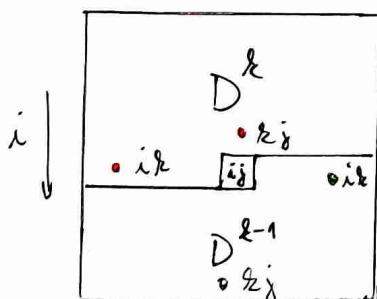
$$\Theta(n^2)$$

vrcholy cesty $\in \{N_1, \dots, N_k\}$



$$\Rightarrow \text{celkově } T(n) = \Theta(n^3)$$

$\Rightarrow S(n) = \Theta(n^2)$ - je možné k dílčím na místě různou z L na D



\rightarrow Zdey se kde může podílet?

1) D_{ij}^{k-1} ... když jsem ještě nepřejel ✓

2) D_{ik}, D_{kj}^{k-1} ... když může být v D^k , ale $D_{ik}^{k-1} = D_{ik}$, takže kde může

• Princip dynamického programování

\rightarrow máme nějaké podproblemy = stavy

\rightarrow a máme graf co říká, který podproblem potřebuje výsledky kterých podproblemů

\Rightarrow tento graf je DAG \Rightarrow ty podproblemy můžeme řešit podle 1. uspořádání