Stromy a jejich reprezentace

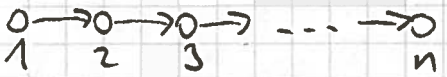
Motivace: verifikace / sensitivita min. kostry → problém cestách maxim

Cíl: Navrhnout DS pro stromy, které bude umět dotazy na cesty...

(vzdálený)

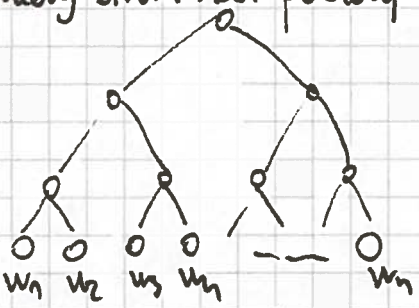
- ohodnocení (váhy) třeba ve vrcholech
- dotazy: "vyjmenuj vrcholy na cestě x→y" (to nejspíš nepůjde rychle)
  "najdi nejlehčí vrchol na cestě x→y" (to už půjde) – cestové minimum (nebo nějaká jiná asociativní operace)
- změny: váha vrcholu – bodový update
  "zvýš o δ všechny váhy na cestě x→y" – cestový update
  rozděl strom / spoj 2 stromy hranou – změny struktury

Statické cesty



+ váhy ve vrcholech

Vytvoříme intervalový strom nad posloupností vah: (inorderem, listové vrcholy odpovídají hranám původního stromu)



- úplný bin. strom hloubky $O(\log n)$ uložený "jako halda"
- podcesta $i \to j$ → interval listů
  Lze pokrýt $O(\log n)$ podstromy, kořen v podstromu si pamatuje min

} cestový dotaz v $O(\log n)$

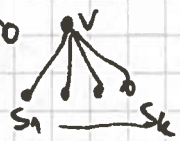
- bodový update → přepočítám cestu do kořene → $O(\log n)$
- cestový update → rozložíme na $O(\log n)$ podstromů
  update podstromu líným vyhodnocováním:
  Do kořene umístíme poznámku "vše níže zvýš o δ",
  kdykoli na ni při průchodu shora narazíme, prostě ji přesuneme do obou synů → ostatní operace poznámky nepotřebují, vždy jsme tu část stromu, do níž přijdou, vyčistili.

] $O(\log n)$

mezi kořenem a těmito podstromy leží $O(\log n)$ vrcholů, jejich minima musíme přepočítat

Heavy-light dekompozice (HLD)

- mějme zakořeněný strom, $s(v)$:= velikost podstromu zakořeněného v v

Def: Pro  hrana vs_i je **těžká** ≡ $s(s_i) \geq s(v)/2$, jinak je **lehká**

Vše vyjde lépe, pokud hrany orientujeme směrem ke kořeni

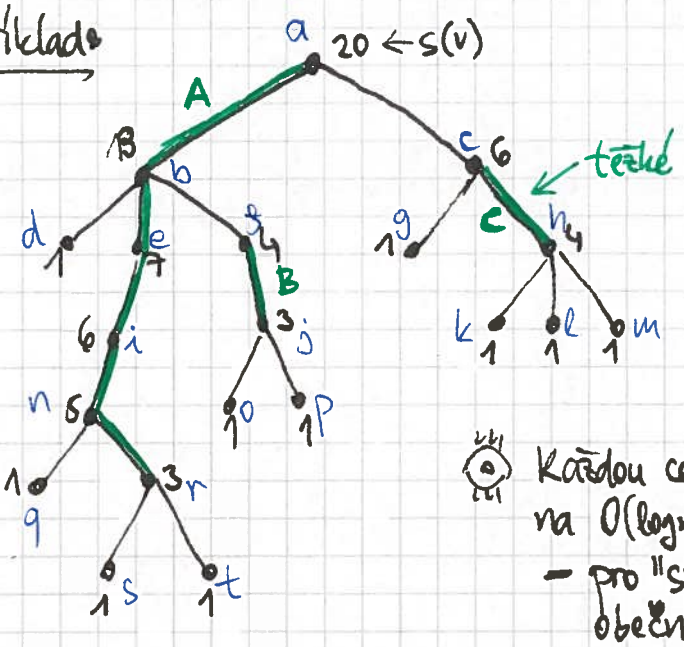
① z $\forall v$ vede dolů nejvýše 1 těžká hrana → $\forall v$ leží na právě 1 **těžké cestě** (možná 1vrcholové)

② na $\forall$ cestě kořen → list leží max. $\log n$ lehkých hran.

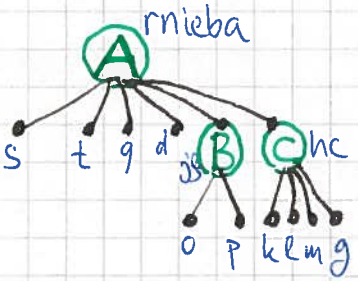
⇒ strom rozložíme na $O(n)$ těžkých cest, jsou propojené lehkými hranami

∞ HLD najdeme v čase $O(n)$ pomocí DFS.

Příklad:



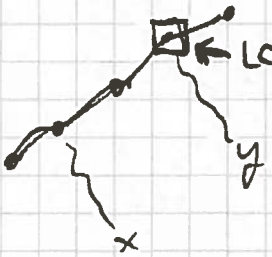
Komprese těžkých cest →



Každou cestu ve stromu umíme rozložit na $O(\log n)$ lehkých hran a $O(\log n)$ částí těžkých cest
- pro "svislou" cestu snadné,
obecnou cestu rozdělíme na 2 svislé v LCA(x,y)

Aplikace:

- LCA(x,y) - nejbližší společný předchůdce
  - pro $\forall v$ předpočítáme: id těžké cesty, pozici na ní } $O(n)$
  - pro $\forall$ těžkou cestu: kam z jejího nejvyššího bodu vede lehká hrana (chm, opět...)
  - pak skáčeme z x,y po těžkých cestách, než objevíme vrcholy společnou:



} $O(\log n)$

- cestové dotazy
  - pro $\forall$ těžkou cestu pořídíme reprezentaci intervalovým stromem
    → cestový dotaz: $O(\log n)$ lehkých hran
    + $O(\log n)$ intervalových dotazů po $O(\log n)$ } $O(\log^2 n)$

... a umíme bodový i cestový update

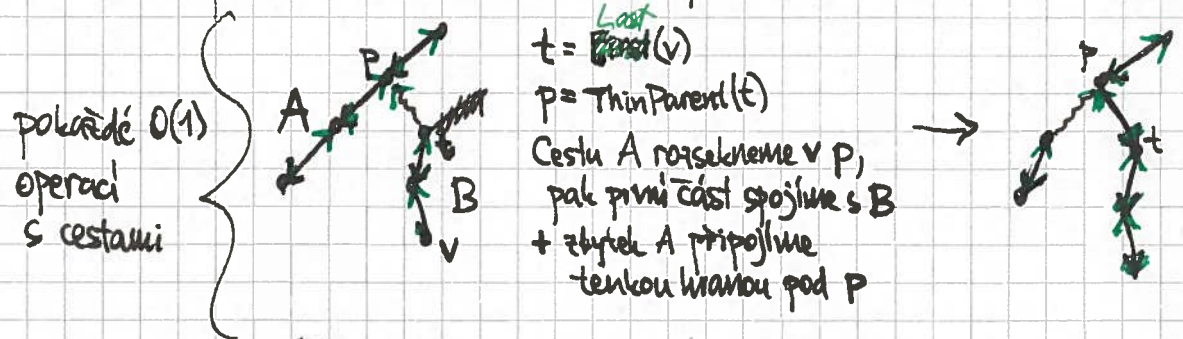
- zrychlení pro statické váhy

z $O(\log n)$ intervalů jsou až na 1 výjimku vše prefixy/suffixy
→ 1 interval po $O(\log n)$, ostatní v $O(1)$ po předvýpočtu px/sx minim
→ celý cestový dotaz v $O(\log n)$

Dynamická dekompozice – Link-Cut stromy [Sleator & Tarjan 1982]

(pozdější verze se Splay-stromy... 1985)

- Místo těžkých/lehkých hran tlusté/tenké
  - není dáno vlastnostmi stromu, ale historií struktury
  - ∀ vrchol má stále max. 1 tlustou hranu do syna
    → tlusté cesty spojené tenkými hranami, DS pro cesty dořešíme později
  - tentokrát o hloubce nic neříkáme, ale amortizovaně vše dopadne očekávaně dobře
- Reprezentujeme les zakořeněných stromů s ohodnocenými vrcholy
  hrany orientujeme do kořene
- Operace:
  - strukturální dotazy: Parent(v), Root(v)
  - strukturální změny: Cut(v) – smaže hranu mezi v a Parent(v)
    Link(u,v) – natáhne hranu z u do v (v musí být kořen)
    Evert(v) – překoření strom za v
  - dotazy na váhy: Cost(v)
    PathMin(v) – min. na cestě mezi v a Root(v)
  - změny vah: SetCost(v,x)
    PathUpdate(v,δ) – přičte δ ke všem váhám na cestě Root(v)→v
- Interně:
  - problém vyřešíme nejprve pro cesty (viz níže), pak pro stromy pomocí rozkladu na tlusté/tenké
  - pro cesty: operace
    - Prev, Next, First, Last
    - Cut, Link, Reverse
    - Cost, PathMin
    - SetCost, PathUpdate
    } Path* z v do Last(v)
  - Expose(v): předělá reprezentaci tak, že cesta Root(v)→v je tlustá & pod v není žádná tlustá hrana
    - kroky: tenká → tlustá (můžeme provést mnohokrát)



tlustá → tenká podobně (to děláme jen k pod v)

- Všechny operace převádíme na Expose + op. na tlusté cestě
  (rozmyslet Evert, ten jako jediný potřebuje Reverse cesty)

Věta [SeT 1982] Expose provede amort. O(log n) kroků

⇒ při repr. cest vyváženými stromy se dostaneme na O(log n) na cestovou op.
→ celkem O(log² n)

Lze vylepšit na O(log n), dokonce w.c. [SeT 1982], ale je to dost pracné.
My ukážeme O(log n) amort. pomocí Splay stromů. [SeT 1985]

Opakování Splay stromů

- Splay(v) "vyrotuje" v do kořene (rotace + dvojrotace)
- Amortizace: - vrcholům přiřadíme libovolné váhy $w(v) > 0$ [struktura o nich neví!]
  - velikost podstromu $s(v) := \sum_{u \in T_v} w(u)$
  - rank vrcholu $r(v) := \log s(v)$
  - potenciál struktury $\Phi := \sum_v r(v)$

Lemma: (přístupové) Splay(v) ve stromu s kořenem k stojí $O(r(k) - r(v))$ rotací

⇒ pro $w = 1$ dostaneme $r = O(\log n)$ ⇒ Splay stojí $O(\log n)$
– nám se časem bude hodit nastavovat váhy jinak, dostaneme jiné odhady ...

Reprezentace cest

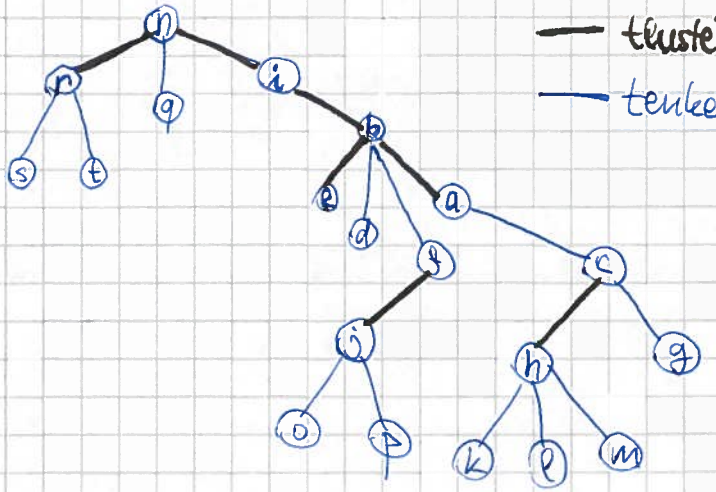
- Každou ~~tlust~~ tlustou cestu popíšeme Splay stromem
- Vrcholy nemají klíče, ale jejich symetrické pořadí odpovídá pořadí na cestě
- Kdykoli sáhneme na vrchol, vysplayujeme ho do kořene
- ve vrcholech minima podstromů, při rotacích snadno přepočteme
- PathMin(v) = Splay(v), pak se podíváme do ~~levého~~ pravého syna na předpočtené min.
- PathUpdate vyhodnocujeme líně, při rotacích čistíme
- Reverse: instrukce "v podstromu prohoď směry", opět vyhodnocujeme líně

! pozor na to, abychom chodili shora dolů, jinak nezvládne abs. směr (vadí to? ??)

Reprezentace stromů

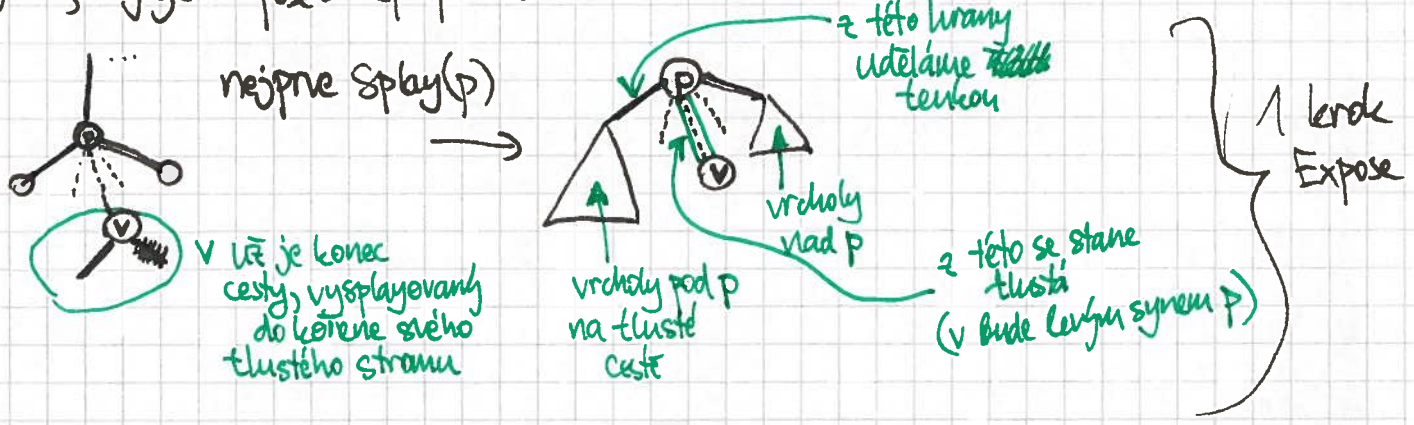
- potřebujeme propojit Splay stromy cest → vrcholy kromě L a P syna dostanou ještě tenké syny – odpovídají tenkým hranám, může jich být libovolně mnoho
  - ale pozor, pozvedne je jen zdola (pamatuje si kořen podřízeného stromu)
  - ↳ přepočítáváme při rotacích
- tím vznikne jeden společný strom se dvěma typy hran

Pro naši ukázkovou dekompozici

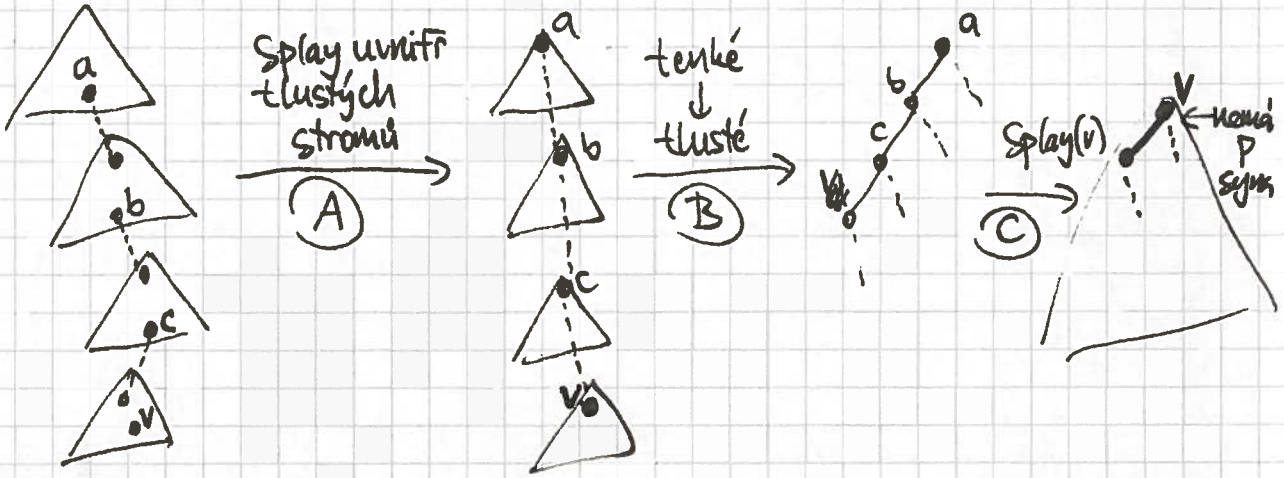


! Líné updaty se nepropagují po tenkých hranách (ani by to nešlo ??)

- jak funguje Expose (případ tenká → tlustá)



Celkově:



Amortizace: Budeme chtít, aby platilo $\hat{s}(v) = \#$ potomků v včetně podřízených stromů pod tenkými hranami

(v) V každém tlustém stromu můžeme nastavit váhy tak, aby $s(v)$ vyšly tak, jak potřebujeme; Φ počítáme dohromady přes celý společný strom

(v) výměny tenká $\leftrightarrow$ t tě neovlivňují Φ

(A) zaplatíme z potenciálu
(C) jakbysmet
(B) shora omezíme časem na (C)

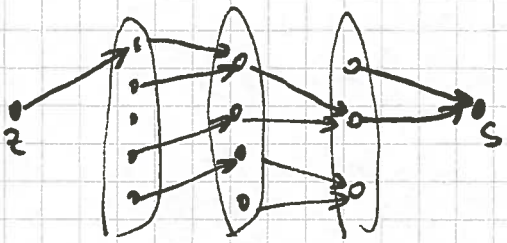
↳ pozor na +1 za každý splay, ale to se nejvíc dá

Vše trvá $O(r(\text{původní kořen}) - r(v))$ [sumy se teleskopují ...]
... ale tv $r(v) \leq \log n$
$\Rightarrow$ Expose trvá $O(\log n)$
$\Rightarrow$ všechny odvozené operace také.

Aplikace Link-Cut Stromů

- Dinicův alg. na hledání max. toku: $n \times$ blokující tok ve vrstevnaté síti
  - obvykle hladově v $O(nm)$ ... opakovaně posílám po cestách (pokaždé $O(n)$ díky vrstvám)
    ... vždy vypadne aspoň 1 hrana $\Rightarrow$ max. m-krát
    + čištění, celkem v $O(m)$

- Zrychlení pomocí Link-Cut:



každý vrchol si vybere 1 odchozí hranu
⇓
vzniknou stromy orientované doprava
↓
L-C strom s vahami na hranách, to jsou rezervy v síti

Opakujeme: 1) Pokud $Root(z) = s$:
- $v \leftarrow PathMin(z)$
- ~~PathUpdate~~
- Pokud $Cost(v) = 0$: $Cut(v)$
  Jinak: $PathUpdate(z, -Cost(v))$

} čistíme hrany s nulovou rezervou
} posíláme po stromu

2) $r \leftarrow Root(z)$
Pokud ∃ neoznačená hrana rt pro nějaké t:
  $Link(r,t)$
} (označím ji) rozšiřujeme strom doprava
Jinak ~~skončíme~~ smažeme všechny hrany do r, na vybraných udělám Cut
} už nešel rozšířit & smažeme jeho kořen (opět čištění)
~~Pokračuj~~
↳ & pokud $r = z$, skončíme.
} už není co smazat → máme prázdnou síť

↳ provedeme $O(m \cdot$ ~~max~~$)$ operací, každá stojí $O(\log n)$
↳ blok. tok najdeme v $O(m \log n)$ – Jak zjistíme, kolik nakonec kudy teče? Stačí porovnat rezervy s kapacitami...
↳ max. tok najdeme v $O(nm \log n)$. & u hran ve stromech řeknu strom, u už smazaných je $r=0$, jinde neteče nic.

[umí se $O(nm)$ – Orlin 2012]