Stromy a jejich reprezentace

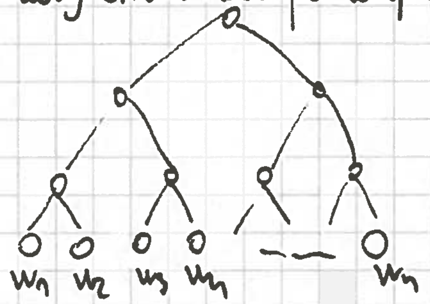
Cíl: Navrhnout DS pro stromy, který bude umět dotazy na cesty...
(vzdálený)

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

Statické cesty

$$\underset{1}{o} \to \underset{2}{o} \to \underset{3}{o} \to \dots \to \underset{n}{o}$$ + váhy ve vrcholech

Vytvoříme intervalový strom nad posloupností vah:

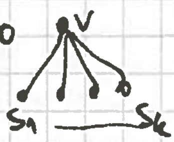


- úplný bin. strom hloubky $O(\log n)$ uložený "jako halda"
- podcesta $i \to j$ → interval listů lze pokrýt $O(\log n)$ podstromy, kořen $\forall$ podstromu si pamatuje min } cestový dotaz v $O(\log n)$
- bodový update → přepočítám cestu do kořene → $O(\log n)$
- cestový update → rozložíme na $O(\log n)$ podstromy
  update podstromu líným vyhodnocováním:
  Do kořene umístíme poznámku "vše níže zvyš o δ", kdykoliv na ni při průchodu shora narazíme, prostě ji přesuneme do obou synů → ostatní operace poznámky nepotkají, vždy jim tu část stromu, do níž přijdou, vyčistíme. } $O(\log n)$

Heavy-light dekompozice (HLD)

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

Df: Pro



hrana vs_i je *těžká* $\equiv s(s_i) \geq s(v)/2$,
jinak je *lehká*

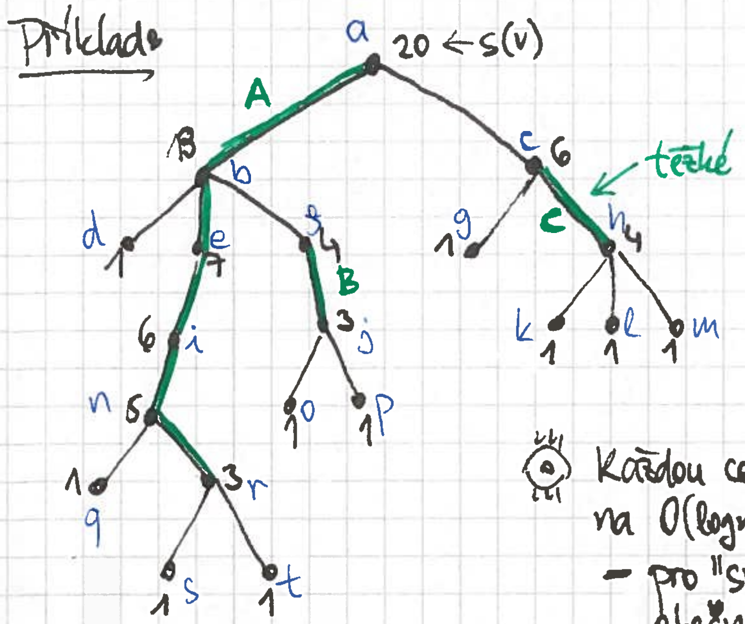
① z $\forall v$ vede dolů nejvýše 1 těžká hrana → $\forall v$ leží na právě 1 *těžké cestě* (možná 1vrcholové)

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

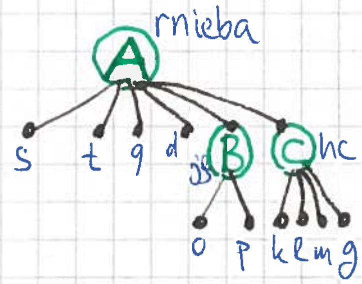
⇒ strom rozložíme na $O(n)$ těžkých cest, jsou propojené lehkými hranami

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

Příklad:



komprese těžkých cest →

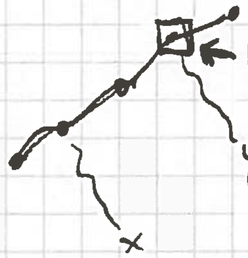


Každou cestu ve stromu můžeme rozložit na $O(\log n)$ lehkých hran a $O(\log n)$ částí těžkých cest

– 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
  - pro $\forall$ těžkou cestu: kam z jejího nejvyššího bodu vede lehká hrana (ehm, opačně...) } $O(n)$
  - pak skáčeme z x,y po těžkých cestách, až objevíme nějakou společnou: } $O(\log n)$



← LCA (poznáme podle pozice na cestě)

- 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

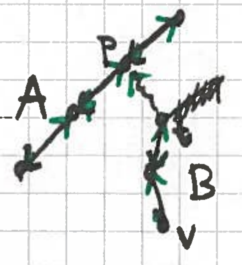
  → celý cestový dotaz v $O(\log n)$

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

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

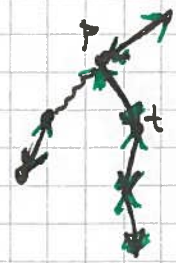
- Místo těžkých/lehkých hran tlusté/tenké
  - není dáno vlastnostmi stromu, ale historií struktury
  - $\forall$ vrchol má stále max. 1 tlustou hranu do syna
    - $\Rightarrow$ tlusté cesty spojené tenkými hranami, DS pro cesty dořešíme později
  - tentokrát o hloubce nic nevíme, ale amortizovaně vše dopadne nečekaně dobře
- Reprezentujeme les zakořeněných stromů s ohodnocenými vrcholy
  hrany zorientujeme 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 vahám na cestě Root(v) → v
- Interně: – problém vyřešíme nejprve pro cesty (viz níže), pak pro stromy pomocí rozkladu na tlusté/tenké
  - operace pro cesty: – 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)

pokaždé O(1) operací s cestami



t = First(v)
p = ThinParent(t)
Cestu A rozsekneme v p, pak první část spojíme s B + zbytek A připojíme tenkou hranou pod p

→



tlustá → tenká podobně (to děláme jen 1x pod v)

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

Věta: [S&T 1982] Expose provede amort. O(log n) kroků

$\Rightarrow$ při repr. cest vyváženými stromy se dostaneme na O(log n) na cestovou op.
$\to$ celkem $O(\log^2 n)$

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

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

Reprezentace cest

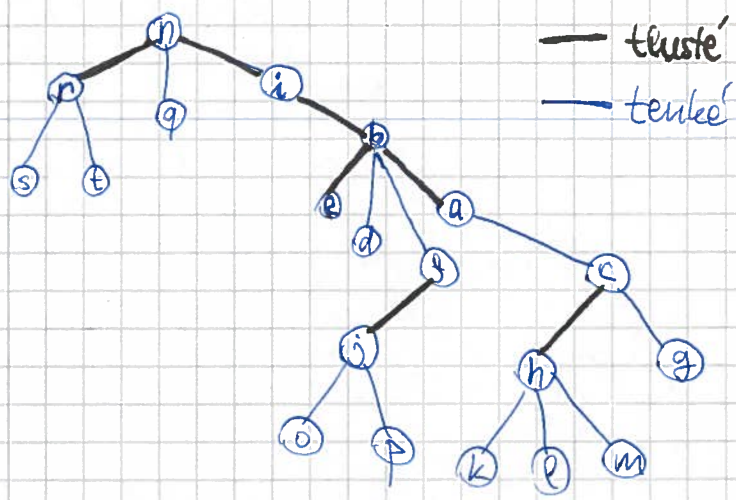
- Každou ~~tlust~~ tlustou cestu popíšeme Splay stromem
- Vrcholy nemají klíče, ale jejich symetrické pořadí odpovídá pořadí na cestě
- Kdykoli sáhneme na vrchol, vysplayujeme ho do kořene
- ve vrcholech minima podstromů, při rotacích snadno přepočteme
- PathMin(v): Splay(v), pak se podíváme do ~~levého~~ 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 neznáme abs. směr (vadí to? :))

Reprezentace stromů

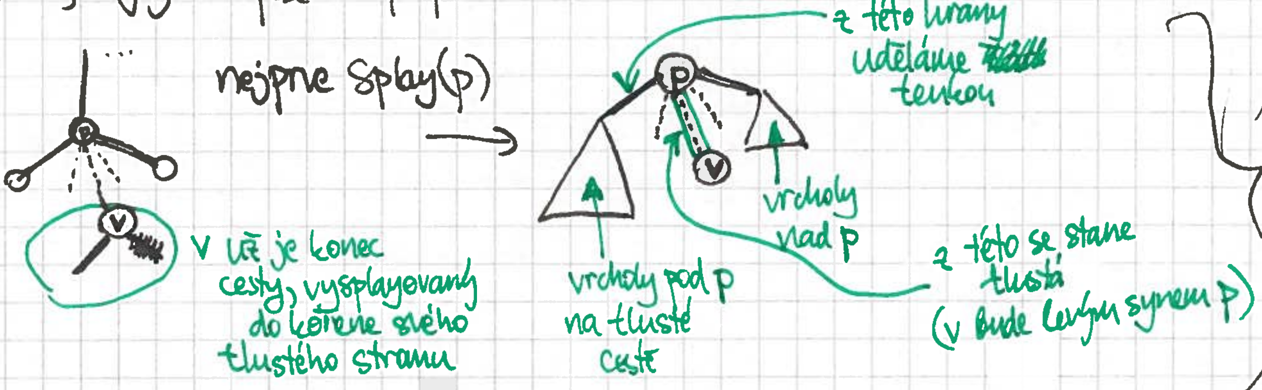
- potřebujeme ~~propojit~~ Splay stromy cest → vrcholy kromě L a P syna dostanou ještě tenké syny – odpovídají tenkým hranám, může jich být libovolně mnoho
  – ale pozor, poznáme je jen zdola (pamatuje si kořen podřízeného stromu) ↳ převěšujeme při rotacích
- tím vznikne jeden společný strom se dvěma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

Pro naši ukázkovou dekompozici



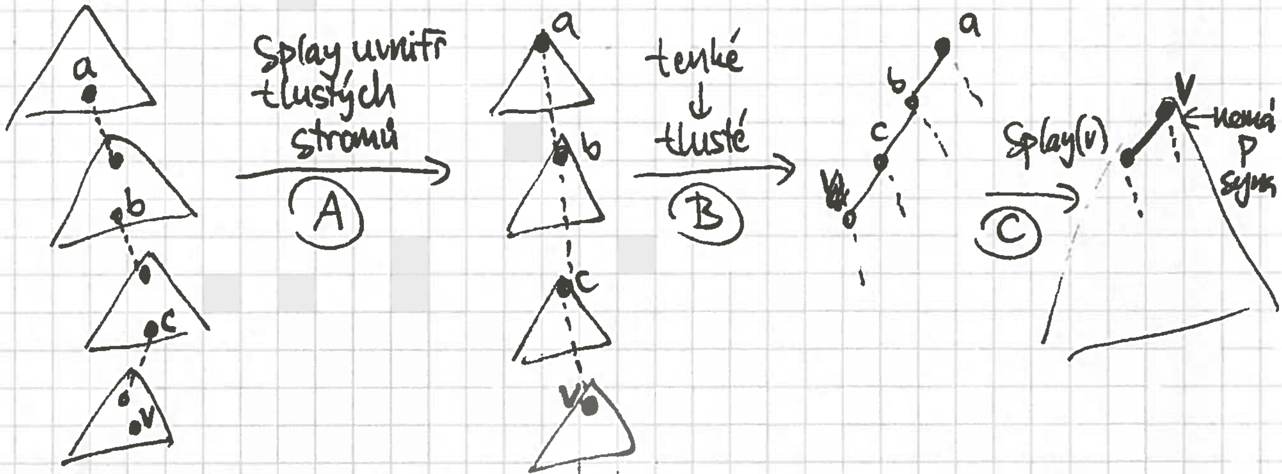
! líné updaty se nepropagují po tenkých hranách (ani by to nešlo :))

- jak funguje Expose: (případ tenká → tlustá)



1 krok Expose

Celkově:



Amortizace: Budeme chtít, aby platilo $s(v)$ = # potomků v včetně podřízených stromů pod ~~tlustými~~ tenkými hranami

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

☺ výměny tenká ⇔ tlustá neovlivňují Φ

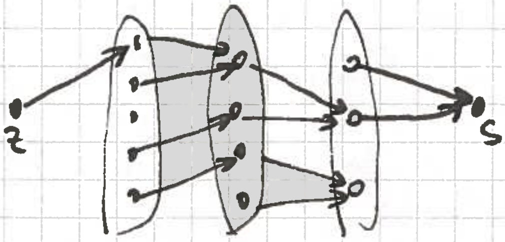
A zaplatíme z potenciálu
~~B~~ jakbysmet
B shora omezíme časem na C

vše trvá $O(r(\text{původní kořen}) - r(v))$
[sumy se steleskopují ...]
... ale tu $r(v) \le \log n$
⇒ Expose trvá $O(\log n)$
⇒ 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 $\mathrm{Root}(z)=s$:
- $v \leftarrow \mathrm{PathMin}(z)$
- ~~PathUpdate~~
- Pokud $\mathrm{Cost}(v)=0$: $\mathrm{Cut}(v)$ } čistíme hrany s nulovou rezervou
  Jinak: $\mathrm{PathUpdate}(z,-\mathrm{Cost}(v))$ } posíláme po stromu

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

↳ provedeme $O(m)$ operací, každá stojí $O(\log n)$
↳ blok. tok najdeme v $O(m \log n)$
↳ max. tok najdeme v $O(nm \log n)$.

[umí se $O(nm)$ – Orlin 2012]