3. cvičení

Datové struktury I, 14. 10. 2024 https://iuuk.mff.cuni.cz/~chmel/2425/ds1/

Úloha 1 (Staticky optimální BVS)
Máme množinu klíčů $k_1 < \ldots < k_n$, ze kterých chceme postavit statický BVS. Zároveň známe četnosti přístupů $w_1, \ldots, w_n$, kde $w_i > 0$. (Také se na tyto četnosti můžeme dívat jako na pravděpodobnosti.)
Vytvořte algoritmus pro sestrojení staticky optimálního BVS, který minimalizuje součet $\sum_{i=1}^{n} h(i) \cdot w_i$, kde $h(i)$ je hloubka klíče k_i, a kořen má hloubku 1. (Tedy hledáme takový strom, že provedení w_i přístupů ke klíči k_i pro každé i bude trvat co nejmenší dobu.)

Úloha 2 (Líně vyvažované stromy)
Připomeňte si, jak fungují líně vyvažované BVS a jakým potenciálem se analyzují.

a) Jak dlouho může trvat provedení k operací provedených na libovolném BB[α] stromu s n vrcholy?

b) Proč je v definici potenciálu výjimka pro rozdíl 1, tedy co by se pokazilo, kdybychom ji neudělali?

Úloha 3 (Hloubka splay stromů nemusí být vždy logaritmická)
Navrhněte posloupnost operací, která vytvoří splay strom s lineární hloubkou a to jak pro naivní splay tak pro správný splay.

Úloha 4 (Naivita se ne vždy vyplácí)
Co se pokazí, když operaci Splay implementujeme naivně, tedy jen pomocí jednoduchých rotací jedné hrany?

Úloha 5 (Splay stromy mají potenciál)
Ujistěte se, že chápete, jak je definovaný potenciál ve splay stromech a že rozumíte hlavním myšlenkám analýzy amortizované složitosti splaye.

a) Jak je definován potenciál splay stromu?

b) Jaká je amortizovaná cena rotace (dvojité a jednoduché)?

c) Jaká je amortizovaná cena celého splaye a jak plyne z amortizovaných cen rotace?

d) Jaké je *reálná* cena celého splaye (a v jakých jednotkách ji vlastně počítáme)?

e) Jaký je potenciál perfektně vyváženého stromu? (Stačí nám rozumný horní a dolní odhad, pro jednoduchost předpokládejte $n = 2^k - 1$.)

f) Jaký je potenciál cesty? (Opět stačí rozumné odhady.)

Úloha 6 (Potenciál pro následníka)
Na prvním cvičení jsme si ukázali, že použití n operací následníka na libovolném BVS, když začneme ve vrcholu s nejmenším klíčem, má složitost $\mathcal{O}(n)$. Jak to můžeme dokázat pomocí potenciálu?

Bonusové úlohy

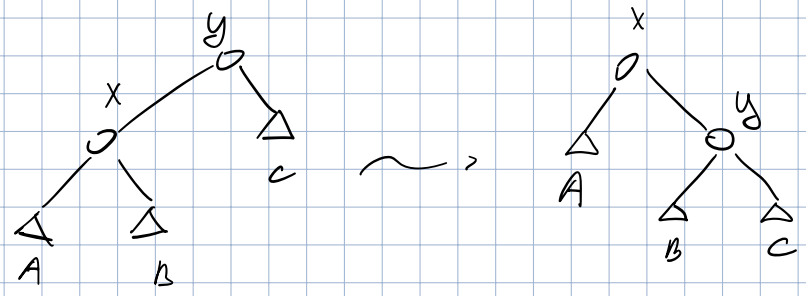
Úloha 7 (Pro fajnšmekry)
Proveďte přestavění BVS na perfektně vyvážený BVS v lineárním čase *a s konstantní pamětí.*

Úloha 8 (Splay spojový seznam)
Uvažme následující problém: máme spojový seznam, a v něm máme prvky $x_1, \ldots, x_n$. Dále pro každý prvek x_i víme, že pravděpodobnost dotazu na tento prvek je p_i, kde $\sum_{i=1}^{n} p_i = 1$. Cena dotazu na prvek x_i je jeho pořadí ve spojovém seznamu. Mějme prvky x_i označené tak, že $p_1 \geq p_2 \geq \ldots \geq p_n \geq 0$.

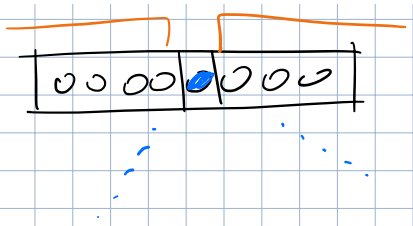
a) Ukažte, že mezi všemi *fixními* spojovými seznamy je námi vytvořený spojový seznam v pořadí $x_1, x_2, \ldots, x_n$ dle pravděpodobností ve střední hodnotě optimální. Tuto hodnotu dále značíme jako $\mathbb{E}[T_{\text{opt}}]$.

b) Uvažme dále tzv. *move-to-front* spojový seznam, kde v každém kroku prvek ve spojovém seznamu vyhledáme, a potom ho umístíme na začátek. (Na začátku je pořadí prvků libovolné.) Ukážeme, že v tomto případě bude střední hodnota ceny maximálně dvojnásobná, a provedeme to následovně. Prvně budeme předpokládat, že na každý prvek jsme se už alespoň jednou podívali, takže pořadí prvků ve spojáku nezávisí na původním pořadí, a budeme zkoumat pořadí prvků v nějaký fixní moment v čase. Z prvního kroku máme, že $\mathbb{E}[T_{\text{opt}}] = \sum_{i=1}^{n} i \cdot p_i$, a dále chceme spočítat $\mathbb{E}[T_{\text{mtf}}] = \sum_{i=1}^{n} \mathbb{E}[R_i] \cdot p_i$. kde R_i je náhodná veličina popisující pořadí prvku x_i v náš konkrétní moment. Konkrétně $\mathbb{E}[R_i] = 1 + \sum_{j=1, j\neq i}^{n} 1 \cdot \Pr[x_j \text{ je před } x_i]$ z linearity střední hodnoty. Vaším úkolem je nyní jen určit pravděpodobnost $\Pr[x_j \text{ je před } x_i]$, a dále odhadnout $\mathbb{E}[T_{\text{mtf}}] \leq 2 \cdot \mathbb{E}[T_{\text{opt}}]$.

Naivní rotace splay stromů: – musí se zachovat pořadí podstromů



Úloha 1 (Staticky optimální BVS)
Máme množinu klíčů $k_1 < \ldots < k_n$, ze kterých chceme postavit statický BVS. Zároveň známe četnosti přístupů $w_1, \ldots, w_n$, kde $w_i > 0$. (Také se na tyto četnosti můžeme dívat jako na pravděpodobnosti.)
Vytvořte algoritmus pro sestrojení staticky optimálního BVS, který minimalizuje součet $\sum_{i=1}^{n} h(i) \cdot w_i$, kde $h(i)$ je hloubka klíče k_i, a kořen má hloubku 1. (Tedy hledáme takový strom, že provedení w_i přístupů ke klíči k_i pro každé i bude trvat co nejmenší dobu.)



– vždycky zkusím každý vrchol jako kořen, podstromy vlevo a vpravo už mám pomocí dyn. prog. vyřešení, tak je jen připojím.

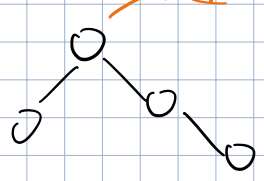
Bude to $O(n^3)$, protože procházím pro n vrcholů, postupně až n prvků, úrovní mám až n.

Úloha 2 (Líně vyvažované stromy)
Připomeňte si, jak fungují líně vyvažované BVS a jakým potenciálem se analyzují.

a) Jak dlouho může trvat provedení k operací provedených na libovolném BB[α] stromu s n vrcholy?

b) Proč je v definici potenciálu výjimka pro rozdíl 1, tedy co by se pokazilo, kdybychom ji neudělali?

b) kdybych neměl tu podmínku:



→ Φ = délka pravé větve

takže při rebuildu by se mi mohl potenciál zvýšit až o délku téhle dlouhé větve

$$\Phi(v) = \begin{cases} 0 & \text{pokud } |s(\ell(v)) - s(r(v))| \leq 1 \\ |s(\ell(v)) - s(r(v))| & \end{cases}$$

$$s(\ell(v)) \leq \alpha \cdot s(v)$$

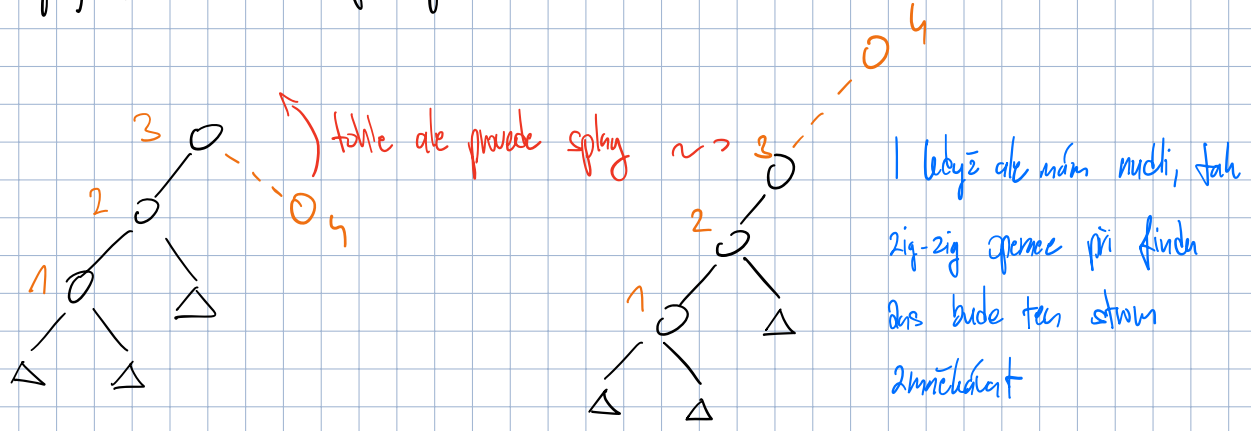
a) bude to trvat $O((k+n)\log n)$

$n \cdot \log n$ → cena potenciálu, kdyžě je strom nevyvážený a pak ho rozbiju

$k \cdot \log n$ → jedna operace stojí $\log n$, provedu jich $k \cdot \log n$

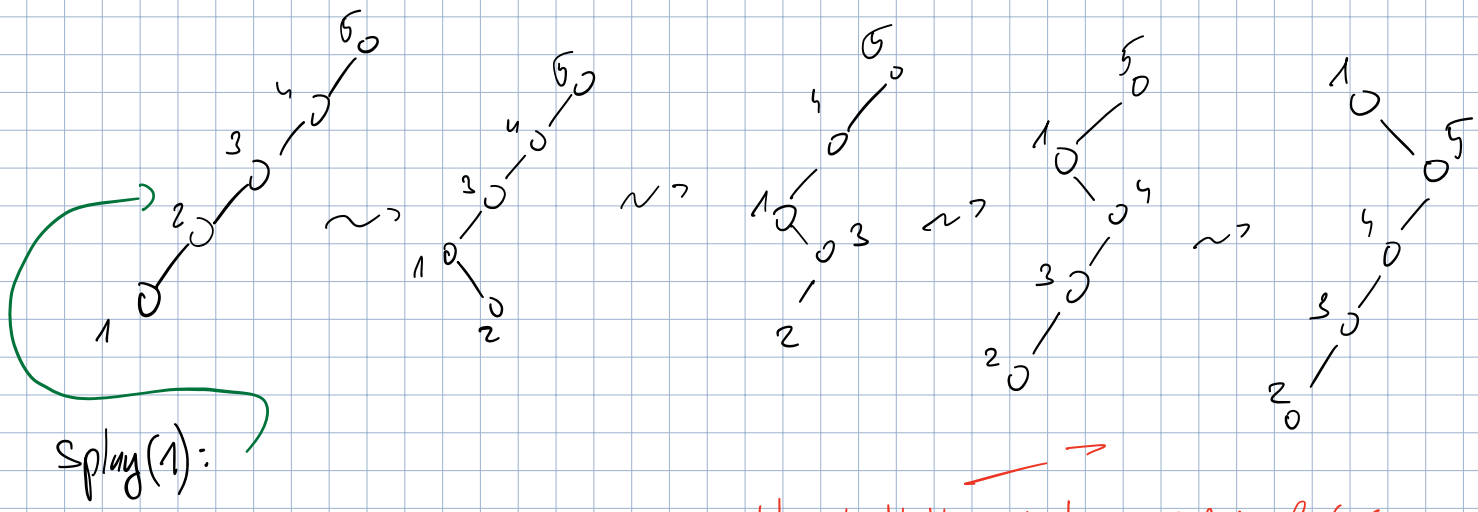
Úloha 3 (Hloubka splay stromů nemusí být vždy logaritmická)
Navrhněte posloupnost operací, která vytvoří splay strom s lineární hloubkou a to jak pro naivní splay tak pro správný splay.

Postupný insert rostoucí posloupnosti



I když ale mám mudli, tak 2zig-2zig operace při findu dos bude ten strom zmenšovat

Úloha 4 (Naivita se ne vždy vyplácí)
Co se pokazí, když operaci SPLAY implementujeme naivně, tedy jen pomocí jednoduchých rotací jedné hrany?



takže ta hloubka mi pak vychází furt lineární, celkem tedy na tu operaci bude potřebovat $O(n^2)$ času.