Lenivé stromy

©kuko 26.9.2019

1 Scapegoat tree

- Rivest, Galperin 1993
- myšlienka: lenivý prístup; predstavme si izbu, ktorú neupratujem (som lenivý); avšak nájsť hocičo viem pomerne rýchlo... po čase sa však bordel nakopí – vtedy sa naštvem a upracem si
- aplikované na stromy: strom necháme rásť (`insert`y), až kým nie je príliš nevyvážený – vtedy nájdeme „vinníka" a celý jeho podstrom prebudujeme na perfektne vyvážený
- prebudovať strom s k vrcholmi na perfektne vyvážený trvá až $O(k)$ času, avšak ukážeme si, že prebudovávať budeme len zriedka
- dôležité je tiež, že ak je nevyvážená len nejaká (malá) časť stromu, prebudujeme len tú časť, nie celý strom

- nech $\text{size}(v)$ je počet vrcholov v podstrome v (vrátane v)
- hovoríme, že strom je v *rovnováhe*, ak pre každý vrchol v a syna s platí:

$$\text{size}(s) \leq \frac{2}{3} \cdot \text{size}(v)$$

- ak je strom v rovnováhe, jeho výška je $\leq \log_{3/2} n \approx 1.7 \lg n$
- ak má jeden syn najviac 2/3 vrcholov v, ten druhý musí mať aspoň 1/3 (zhruba)
- intuícia: na začiatku/hneď po prebudovaní je vrchol v rovnováhe a obaja synovia majú zhruba 1/2 vrcholov; treba lineárne veľa `insert`ov, aby bolo v jednom podstrome viac ako 2/3, takže lineárny čas prebudovania sa rozpočíta na lineárne veľa operácií

- a ako implementujeme vymazávanie? úplne lenivo:
- namiesto odstránenia vrchol iba označíme za vymazaný (pozor, musíme opraviť `find`, aby ignoroval vymazané vrcholy)
- ak strom obsahuje aspoň polovicu vrcholov, prebudujeme celý strom a pritom označené vrcholy odstránime (to trvá $O(n)$, ale predtým muselo byť $n/2$ `delete`-ov)

2 Analýza

- invariant: každý vrchol má našetrené $|\text{size}(\ell(v)) - \text{size}(r(v))|$, kde $\ell(v), r(v)$ je ľavý a pravý syn v, okrem prípadu, že je tento rozdiel ≤ 1 a vtedy netreba mať našetrené nič
- tesne po prebudovaní podstromu v je tento podstrom perfektne vyvážený a rozdiel veľkostí je ≤ 1 (netreba mať našetrené nič)
- čím je vrchol menej vyvážený, tým má viac našetrené
- ukážeme, že ak za každú operáciu `insert` dostaneme $4 \log n\$$, vystačí nám to
  - z $2 \log n\$$ zaplatíme hľadanie správneho miesta a vloženie
  - $2 \log n\$$ si odložíme na neskôr: každý vrchol na ceste od koreňa do nového vrcholu dostane 2\$ – všimnime si, že to sú presne tie vrcholy, ktorým sa zväčší rozdiel veľkostí synov o 1 a keď každému prispejeme 2\$, invariant ostane zachovaný
  - ak je po vložení strom stále v rovnováhe, operácia týmto končí
  - ukážeme, že ak dôjde k prebudovaniu nevyváženého podstromu, dokážeme to zaplatiť z už našetrených mincí, tzn. z pohľadu amortizovanej analýzy bude prebudovanie „zadarmo"
- ak $\text{size}(a) > \frac{2}{3}\,\text{size}(v)$, potom $\text{size}(b) < \text{size}(v) - \text{size}(a) < \frac{1}{3}\,\text{size}(v)$ a teda

$$|\,\text{size}(a) - \text{size}(b)| > \frac{1}{3}\,\text{size}(v)$$

- takže v má $> \frac{1}{3}\,\mathrm{size}(v)\$$, z čoho zaplatíme $O(\mathrm{size}(v))$ práce pri prebudovaní

- na `delete` nám stačí $2\log n + 1\$$, pričom $2\log n\$$ zaplatíme za vyhľadanie a označenie vrcholu a posledný 1\$ si odložíme na prípadné neskoršie prebudovanie
- na začiatku, alebo po poslednom kompletnom prebudovaní je 0 označených vrcholov a za každý označený vrchol si odložíme 1\$, tzn. keď je polovica vrcholov označených, máme $n/2\$$, z ktorých hravo zaplatíme $O(n)$ práce pri kompletnom prebudovaní
- pozn.: n tu označovalo počet vrcholov stromu, čo je najviac dvakrát skutočný počet prvkov v strome

- záver: scapegoat tree podporuje operácie
  - `insert` a `delete` v čase $O(\log n)$ amortizovane (v najhoršom prípade môže jedna operácia trvať až $\Theta(n)$)
  - `find` má garantovaný čas $O(\log n)$ v najhoršom prípade – hĺbka stromu je $\leq 1.7\lg n$
  - (konštanta ≈ 1.7 zjavne súvisí s voľbou 2/3 v definícii vyváženosti; táto voľba je trade-off medzi hĺbkou stromu a teda rýchlym `find`om a častejšími prebudovaniami a pomalším `insert`om)