LCA a RMQ

©kuko 6.12.2018

1 Úvod

- RMQ (range minimum query – hľadanie minima v intervale):
  - dané je pole $A[0, \ldots, n-1]$
  - chceme si ho predspracovať tak, aby sme pre ľubovoľnú dvojicu i, j vedeli rýchlo povedať $\mathrm{rmq}(i,j) = \mathrm{argmin}_{i \le k \le j} A[k]$
- jednoduché riešenia:
  - žiadne predspracovanie, query v $O(n)$
  - predpočítame každú dvojicu – $O(n^2)$ pamäť aj predspracovanie, $O(1)$ query
  - intervalový strom – $O(n)$ pamäť a predspracovanie, $O(\log n)$ query
- dá sa to lepšie?
- áno: netreba predpočítať každú dvojicu; stačí intervaly dĺžky 2^k;
  - ľubovoľný interval totiž vieme zložiť z dvoch (prekrývajúcich sa) intervalov dĺžky 2^k
  - $O(n \log n)$ pamäť a predspracovanie, $O(1)$ query (tento trik funguje vďaka tomu, že min je idempotentná operácia)

```
        |-------------------|             - interval dĺžky 11
x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x
        |-------------|                   - pokryjeme dvoma intervalmi
              |-------------|               dĺžky 8
```

- dá sa to ešte lepšie?

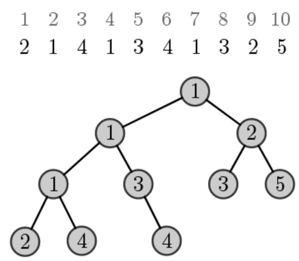
- LCA (lowest common ancestor – najnižší spoločný predok):
  - daný je strom
  - chceme si ho predspracovať tak, aby sme pre ľubovoľnú dvojicu vrcholov x, y vedeli rýchlo nájsť ich lca (najnižší vrchol na spoločnej ceste od koreňa ku x aj y)
- jednoduché riešenia:
  - žiadne predspracovanie, query v $O(n)$
  - predpočítame každú dvojicu – $O(n^2)$ pamäť aj predspracovanie, $O(1)$ query
- dá sa to lepšie?
- áno: pre každý vrchol si predpočítajme jeho hĺbku a $\log n$ pointrov – skok o 1, 2, 4, 8, 16, … hore
  - query: najskôr vyskáčeme z hlbšieho vrcholu na úroveň toho vyššie, následne binárnym vyhľadávaním nájdeme najhlbší spoločný vrchol
  - presnejšie: skúsime skočiť z x aj y do polovice cesty ku koreňu; ak sa dostaneme na spoločný vrchol, lca je toto, alebo niečo nižšie – preto si vrchol zapamätáme, vrátime sa a skúsime skočiť menej; ak sa dostaneme na dva rôzne vrcholy, premiestnime sa tam a zmenšíme veľkosť skoku o polovicu
  - $O(n \log n)$ pamäť a predspracovanie, $O(\log n)$ čas
- dá sa to ešte lepšie??

- spoiler: obidve úlohy sa dajú vyriešiť v $O(1)$ čase, $O(n)$ pamäť a predspracovanie
- a v skutočnosti majú obe úlohy viac spoločného, ako sa na prvý pohľad zdá

2 Vzťah RMQ a LCA

2.1 RMQ → LCA

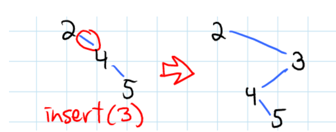
- pre dané pole A vyrobme Kartézsky strom, t.j.
  - koreň bude $\min_{0 \le k < n} A[k]$ (resp. niektoré z miním)
  - ľavý podstrom budú vrcholy v poli naľavo,
  - pravý podstrom budú vrcholy v poli napravo



- tento strom je halda a inorder prechod dá pôvodné pole
- navyše, ak očíslujeme vrcholy stromu podľa indexov v poli, tak:

$$\mathrm{rmq}(i, j) = \mathrm{lca}(i, j)$$

- táto redukcia sa dá spraviť v lineárnom čase:
  - prechádzame pole A zľava doprava
  - ak prichádzajú stále väčšie prvky, pripájame ich k ceste
  - keď príde menší prvok, potrebujeme vyjsť vyššie, kým nenarazíme na niečo menšie a upraviť hrany takto:



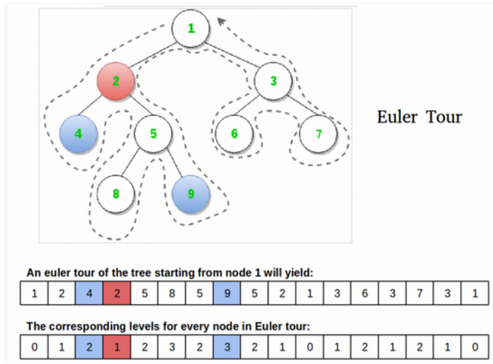
  - môžeme si predstaviť, že vrcholy na ceste z koreňa stále doprava máme na stacku, ktorý push-ujeme a pop-ujeme
  - každý prvok najviac raz push-neme a raz pop-neme, preto je algoritmus lineárny

2.2 LCA → RMQ

- pre daný strom T spravme inorder prechod a poznačme si hĺbky vrcholov, potom

$$\mathrm{lca}(i, j) = \mathrm{rmq}(i, j)$$

- všimnime si však ešte čosi zaujímavé: výsledná úloha RMQ je pole len s číslami od 0 po n
- ba čo viac, namiesto inorder prechodu môžeme spraviť Eulerovskú cestu (predstavte si, že idete okolo stromu)
- každý prvok tak zapíšeme aspoň dvakrát, na druhej strane dostaneme pole, kde sa susedné prvky sa líšia iba o ± 1 (tzv. RMQ$\pm$1 úloha)
- tzn. ľubovoľné pole A vieme pomocou redukcii RMQ → LCA → RMQ$\pm$1 zmeniť na pole, kde sa prvky líšia len o ± 1, ale pozícia minima (argmin) je pre každý interval rovnaká



- strom T a pole A navyše navzájom prelinkujeme, t.j.:
  - pre každý vrchol si zapamätáme index do poľa, keď sme vrchol prvýkrát navštívili a
  - pre každý prvok v poli si zapamätáme pointer na príslušný vrchol v strome

$$\mathrm{lca}(i, j) = \mathrm{rmq}(\mathrm{first}[i], \mathrm{first}[j])$$

3 Optimálny algoritmus

- úlohu najskôr zredukujeme na RMQ± 1
- odrazíme sa od $O(1)$ / $O(n \log n)$ riešenia z úvodu, t.j. interval dĺžky k pokryjeme dvoma intervalmi dĺžky $2^{\lfloor \lg k \rfloor}$
- riešenie je dobré, má však trochu veľkú pamäť; ako to zlepšíme?
- rozdelíme celé pole na bloky dĺžky $n' = \frac{1}{2} \lg n$
- pre každý blok si spočítame minimum – tieto minimá si uložme do poľa B
- keďže B má dĺžku iba $2n/\log n$, môžeme naň použiť náš $O(n \log n)$ algoritmus – prespracovanie a pamäť bude $O(n)$ pre túto časť
- odpoveď na ľubovoľnú query do poľa A vieme poskladať z query na pole B (minimum z blokov) + treba nám môže z každej strany ešte kúsok vytŕčať
- potrebujeme teda úlohu ešte vyriešiť pre jednotlivé bloky
- a tu príde ten ofajč:
  - ak všetky prvky v jednom bloku zvýšime alebo znížime o rovnako veľa, pozícia minima sa nezmení
  - môžeme si teda predstaviť, že každý blok začína od nuly
  - každý blok potom vieme zapísať jednoducho ako bitstring: 0, ak nasleduje menší prvok, 1, ak nasleduje väčší prvok
  - všetkých možných bitstringov je málo – iba $2^{n'} = 2^{\frac{1}{2} \lg n} = O(\sqrt{n})$
  - aj všetkých možných otázok je málo – iba $O(\log^2 n)$
  - každá odpoveď zaberá iba $O(\log n)$, dokonca $O(\log \log n)$ bitov
- riešenie: spravíme si jednu „veľkú“ tabuľku, kde si pre každý možný blok (t.j. pre každý možný bitstring) a pre každú možnú otázku predpočítame odpoveď
- (N.B.: my nemáme tabuľku pre každý blok, ale jednu „globálnu“ pre každý blok)
- odpovedať teda budeme vedieť v $O(1)$
- veľkosť tabuľky je $O(\sqrt{n} \cdot \log^3 n)$, čo je $o(n)$ – menej, ako lineárna