

Rótarþáttun

Bergur Snorrason

16. febrúar 2021

Dæmi

- ▶ Gefinn er listi með n tölum.
- ▶ Næst koma q fyrirspurnir, þar sem hver er af einni af tveimur gerðum:
 - ▶ Bættu k við i -tu töluna.
 - ▶ Reiknaðu summu allra talna á bilinu $[i, j]$.

Almenn k -þáttun

- ▶ Hvað ef við skiptum fylkinu upp í k (næstum) jafnstór hólf.
- ▶ Við getum þá haldið utan um, og uppfært, summu hvers hólfis auðveldlega.
- ▶ Til að finna summu á einhverju bili í fylkinu nægir að reikna summu hólfana á milli endapunktana og leggja svo afganginn við (afgangurinn er í mesta lagi lengd tveggja hólfra).
- ▶ Tökum eftirfarandi sýnidæmi sem skipt hefur verið í þrjú hólf,

$$p = [0 \ 1 \ 4 \mid 3 \ 4 \ 5 \mid 0 \ 1 \ 8 \ 9].$$

- ▶ Köllum fylkið sem geymir summu hvers hólfis s , sem verður þá

$$s = [5 \ 12 \ 18].$$

- ▶ Ef við viljum uppfæra, til dæmis bæta 5 við stak 2, þá þurfum við að sjálfsögðu að uppfæra p , en það þarf líka breyta s .
- ▶ Til að breyta p gerum við einfaldlega $p[2] += 5$.
- ▶ Til að uppfæra s þurfum við að finna hólfið sem stak 2 tilheyrir. Þar sem það er í hólfi 0 notum við $s[0] += 5$.
- ▶ Svona líta svo fylkin út, fyrir og eftir uppfærslu.

Fyrir breytingu	Eftir breytingu
$p = [0 \ 1 \ 4 \mid 3 \ 4 \ 5 \mid 0 \ 1 \ 8 \ 9]$	$p = [0 \ 1 \ 9 \mid 3 \ 4 \ 5 \mid 0 \ 1 \ 8 \ 9]$.
$s = [5 \ 12 \ 18]$	$s = [10 \ 12 \ 18]$

- ▶ Ég fór mjög losaralega í hvernig ætti að framkvæma seinni aðgerðina.
- ▶ Skoðum, sem dæmi, hverju eigi að skila fyrir 2 1 8.
- ▶ Það er aðeins eitt hólf á milli staks 1 og staks 8, hólf 1.
- ▶ „Afgangurinn”, eins og ég kallaði hann áðan, eru þau stök sem ekki eru í hólfi 1 en eru þó á bilinu frá 1 til 8.
- ▶ Þetta eru stök 1, 2, 6, 7 og 8 (samtals summan er þá 31).
- ▶ Við erum því að leggja saman rauðu stökin á myndinni fyrir neðan,

$$\begin{array}{l} p = [0 \text{ } 1 \text{ } 9 \mid 3 \text{ } 4 \text{ } 5 \mid 0 \text{ } 1 \text{ } 8 \text{ } 9] \\ s = [10 \text{ } 12 \text{ } 18] \end{array} \quad .$$

- ▶ En er þetta hraðar en frumstæða aðferðin sem við skoðuðum í upphafi?
- ▶ Það fer að sjálfsögðu allt eftir því hversu stór hólf við veljum.
- ▶ Ef fylkinu er skipt upp í n hólf er nokkuð ljóst að þessi aðferð er jafngild frumstæðu aðferðinni.
- ▶ Ef fylkinu er skipt upp í 1 hólf gildir það sama.
- ▶ Munið að við létum k tákna fjölda hólfa.
- ▶ Fyrri aðgerðin er ennþá $\mathcal{O}(1)$, en seinni aðgerðin verður $\mathcal{O}(n/k + k)$, svo tímaflækjan er $\mathcal{O}(qn/k + qk)$.

Skynsamlegt val á k

- ▶ Þar sem að fyrrir aðgerðin er ekki háð skiptingunni þá nægir að lágmarka $\frac{n}{k} + k$.
- ▶ Látum $f(k) = \frac{n}{k} + k$.
- ▶ Við höfum $f'(k) = -\frac{n}{k^2} + 1$.
- ▶ Útgildispunktur fást í

$$\begin{aligned}f'(k) &= 0 \\ \Rightarrow 1 - \frac{n}{k^2} &= 0 \\ \Rightarrow 1 &= \frac{n}{k^2} \\ \Rightarrow k^2 &= n \\ \Rightarrow k &= \sqrt{n}.\end{aligned}$$

- ▶ Nú þarf bara að ganga úr skugga um að þessi skipting sé betri en línuleg.

- ▶ Ef við veljum $k = \sqrt{n}$ þá er tímaflækja seinni aðgerðarinnar

$$\mathcal{O}\left(\frac{n}{\sqrt{n}} + \sqrt{n}\right) = \mathcal{O}(\sqrt{n} + \sqrt{n}) = \mathcal{O}(\sqrt{n}).$$

- ▶ Því er tímaflækjan á lausninni $\mathcal{O}(q\sqrt{n})$.
- ▶ Svo þessi aðferð er betri en sú frumstæða, ef við skiptum í \sqrt{n} hólf.
- ▶ Við köllum það *rótarþáttun* (e. *squareroot decomposition*) þegar við skiptum upp í \sqrt{n} hólf.


```

6 #define MAXN 2000000
7 int a[MAXN], s[MAXN], n, e; // n = e*e
8
9 void update(int x, int y)
10 { // Bætir y við x-ta stakið.
11     s[x/e] += y;
12     a[x] += y;
13 }
14
15 int query(int x, int y)
16 { // Finnum summuna yfir [x, y - 1].
17     int r = 0;
18     while (x%e != 0 && x < y) r += a[x++];
19     if (x == y) return r;
20     while (y%e != 0) r += a[--y];
21     while (x < y) r += s[x/e], x += e;
22     return r;
23 }

```

Lygn uppfærsla

- ▶ Við getum einnig framkvæmt lygnar uppfærslur þegar við notum rótarþáttun (líkt og með biltré).
- ▶ Við uppfærum þá beint þau gildi sem eru í sömu hólfum og endarpunktur bilsins sem við uppfærum yfir.
- ▶ Við framkvæmum svo lygna uppfærslu á þau hólf sem liggja þar á milli.

```

6 #define MAXN 2000000
7 int a[MAXN], s[MAXN], o[MAXN], n, e; // n = e*e
8
9 void prop(int x) // Hjálparfall
10 { // Framkvæmir þær uppfærslur sem á eftir að gera á fötu x.
11     int i;
12     s[x] += o[x]*e;
13     for (i = 0; i < e; i++) a[x*e + i] += o[x];
14     o[x] = 0;
15 }
16
17 int query(int x, int y)
18 { // Finnum summuna yfir [x, y - 1].
19     prop(x/e), prop((y - 1)/e);
20     int r = 0;
21     while (x%e != 0 && x < y) r += a[x++];
22     if (x == y) return r;
23     while (y%e != 0) r += a[--y];
24     while (x < y) r += s[x/e] + o[x/e]*e, x += e;
25     return r;
26 }
27
28 void update(int x, int y, int z)
29 { // Bætum z við stökin [x, y - 1].
30     prop(x/e), prop((y - 1)/e);
31     while (x%e != 0 && x < y) a[x] += z, s[x++/e] += z;
32     if (x == y) return;
33     while (y%e != 0) a[--y] += z, s[y/e] += z;
34     while (x < y) o[x/e] += z, x += e;
35 }

```

