Rótarþáttun

Bergur Snorrason

14. febrúar 2022

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 hólfs 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 hólfa).
- 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 hólfs 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.

$$\begin{array}{cc}
\text{Fyrir breytingu} & \text{Eftir breytingu} \\
p = [0\ 1\ 4 \mid 3\ 4\ 5 \mid 0\ 1\ 8\ 9] & p = [0\ 1\ 9 \mid 3\ 4\ 5 \mid 0\ 1\ 8\ 9] \\
s = [5\ 12\ 18] & s = [10\ 12\ 18]
\end{array}.$$

- 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,

$$
\begin{aligned}
p &= [0\ 1\ 9 \mid 3\ 4\ 5 \mid 0\ 1\ 8\ 9] \\
s &= [10\ 12\ 18]
\end{aligned}
.
$$

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

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

- 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.

```
#define MAXN 2000000
int a[MAXN], s[MAXN], n, e; // n = e*e

void update(int x, int y)
{ // Bætir y við x-ta stakið.
    s[x/e] += y;
    a[x] += y;
}

int query(int x, int y)
{ // Finnum summuna yfir [x, y - 1].
    int r = 0;
    while (x%e != 0 && x < y) r += a[x++];
    if (x == y) return r;
    while (y%e != 0) r += a[--y];
    while (x < y) r += s[x/e], x += e;
    return r;
}
```

Lygn uppfærlsa

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

```c
#define MAXN 2000000
int a[MAXN], s[MAXN], o[MAXN], n, e; // n = e*e

void prop(int x) // Hjálparfall
{ // Framkvæmir þær uppfærslur sem á eftir að gera á fötu x.
    int i;
    s[x] += o[x]*e;
    for (i = 0; i < e; i++) a[x*e + i] += o[x];
    o[x] = 0;
}

int query(int x, int y)
{ // Finnum summuna yfir [x, y - 1].
    prop(x/e), prop((y - 1)/e);
    int r = 0;
    while (x%e != 0 && x < y) r += a[x++];
    if (x == y) return r;
    while (y%e != 0) r += a[--y];
    while (x < y) r += s[x/e] + o[x/e]*e, x += e;
    return r;
}

void update(int x, int y, int z)
{ // Bætum z við stökin [x, y - 1].
    prop(x/e), prop((y - 1)/e);
    while (x%e != 0 && x < y) a[x] += z, s[x++/e] += z;
    if (x == y) return;
    while (y%e != 0) a[--y] += z, s[y/e] += z;
    while (x < y) o[x/e] += z, x += e;
}
```