Hrúgur

Bergur Snorrason

14. febrúar 2022

Hrúgur

- Rótartvíundatré sem uppfyllir að sérhver nóða er stærri en börnin sín er sagt uppfylla *hrúguskilyrðið*.
- Við köllum slík tré *hrúgur* (e. *heap*).
- Hrúgur eru heppilega auðveldar í útfærslu.
- Við geymum tréð sem fylki og eina erfiðið er að viðhalda hrúguskilyrðinu.

- Þegar við geymum tréð sem fylki notum við eina af tveimur aðferðum.
- Sú fyrri:
  - Rótin er í staki 1 í fylkinu.
  - Vinstra barn staks i er stak $2 \cdot i$.
  - Hægra barn staks i er stak $2 \cdot i + 1$.
  - Foreldri staks i er stakið $\lfloor i/2 \rfloor$.
- Sú seinni:
  - Rótin er í staki 0 í fylkinu.
  - Vinstra barn staks i er stak $2 \cdot i + 1$.
  - Hægra barn staks i er stak $2 \cdot i + 2$.
  - Foreldri staks i er stakið $\lfloor (i - 1)/2 \rfloor$.

- Bein afleiðing af hrúguskilyrðinu er að rótin er stærsta stakið í trénu.
- Við getum því alltaf fengið skjótan aðgang að stærsta stakinu í trénu.
- Algengt er að *forgangsbiðraðir* (e. *priority queues*) séu útfærðar með hrúgum.

```
#define PARENT(i) ((i - 1)/2)
#define LEFT(i)   ((i)*2 + 1)
#define RIGHT(i)  ((i)*2 + 2)
int h[MAXN], hn = 0;
void swap(int* x, int* y) { int t = *x; *x = *y; *y = t; }
void fix_down(int i) // Hjálparfall.
{ // Ferðast niður tréð og lagar hrúguskilyrðið á leiðinni.
    int mx = i;
    if (RIGHT(i) < hn && h[mx] < h[RIGHT(i)]) mx = RIGHT(i);
    if (LEFT(i) < hn && h[mx] < h[LEFT(i)]) mx = LEFT(i);
    if (mx != i) swap(&h[i], &h[mx]), fix_down(mx);
}

void fix_up(int i) // Hjálparfall.
{ // Ferðast upp tréð og lagar hrúguskilyrðið á leiðinni.
    if (i == 0 || h[i] <= h[PARENT(i)]) return;
    swap(&h[i], &h[PARENT(i)]), fix_up(PARENT(i));
}

void pop()
{ // Fjarlægir stærsta stakið.
    h[0] = h[--hn];
    fix_down(0);
}

int peek() { return h[0]; } // Skilar stærsta stakinu.
int size() { return hn; } // Skilar stærð hrúgurnar.
void push(int x)
{ // Bætir x við hrúguna.
    h[hn++] = x;
    fix_up(hn - 1);
}
```

- Gerum ráð fyrir að við séum með n stök í hrúgunni.
- Þá er hæð trésins $\mathcal{O}(\log n)$.
- Þar sem `pop()` þarf aðeins að ferðast einu sinni niður að laufi er tímaflækjan $\mathcal{O}(\log n)$.
- Þar sem `push(...)` þarf aðeins að ferðast einu sinni upp að rót er tímaflækjan $\mathcal{O}(\log n)$.
- Nú þarf `peek()` ekki að gera annað en að lesa fremsta stakið í fylki svo tímaflækjan er $\mathcal{O}(1)$.