Hrúgur

Bergur Snorrason

16. febrúar 2021

Hrúgur

- Rótartvíundatré sem uppfyllir að sérhver nóða er stærri en börnin sín er sagt uppfylla *hrúguskilyrðið*.

Hrúgur

- Rótartvíundatré sem uppfyllir að sérhver nóða er stærri en börnin sín er sagt uppfylla *hrúguskilyrðið*.
- Við köllum slík tré *hrúgur* (e. *heap*).

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.

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.

- Þegar við geymum tréð sem fylki notum við eina af tveimur aðferðum.
- Sú fyrri:

- Þegar við geymum tréð sem fylki notum við eina af tveimur aðferðum.
- Sú fyrri:
  - Rótin er í staki 1 í fylkinu.

- 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 $2 \cdot i$.

- 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 $2 \cdot i + 1$.

- 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 $\lfloor i/2 \rfloor$.

- 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:

- 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.

- 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 $2 \cdot i + 1$.

- 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 $2 \cdot i + 2$.

- 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.

- Bein afleiðing af hrúguskilyrðinu er að rótin er stærsta stakið í trénu.
- Við getum því alltaf fengið skjótan aðgang að stærsta stakinu í trénu.

- 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.

```
1  #define PARENT(i) ((i - 1)/2)
2  #define LEFT(i)   ((i)*2 + 1)
3  #define RIGHT(i)  ((i)*2 + 2)
4  #define MAXN 1000000
5  int h[MAXN], hn = 0;
6
7  void swap(int* x, int* y) { int t = *x; *x = *y; *y = t; }
8  void fix_down(int i) // Hjálparfall.
9  { // Ferðast niður tréð og lagar hrúguskilyrðið á leiðinni.
10     int mx = i;
11     if (RIGHT(i) < hn && h[mx] < h[RIGHT(i)]) mx = RIGHT(i);
12     if (LEFT(i) < hn && h[mx] < h[LEFT(i)]) mx = LEFT(i);
13     if (mx != i) swap(&h[i], &h[mx]), fix_down(mx);
14 }
15
16 void fix_up(int i) // Hjálparfall.
17 { // Ferðast upp tréð og lagar hrúguskilyrðið á leiðinni.
18     if (i == 0) return;
19     else if (h[i] > h[PARENT(i)])
20         swap(&h[i], &h[PARENT(i)]), fix_up(PARENT(i));
21 }
22
23 void pop()
24 { // Fjarlægir stærsta stakið.
25     h[0] = h[--hn];
26     fix_down(0);
27 }
28
29 int peek() { return h[0]; } // Skilar stærsta stakinu.
30
31 void push(int x)
32 { // Bætir x við hrúguna.
33     h[hn++] = x;
34     fix_up(hn - 1);
35 }
```

- Gerum ráð fyrir að við séum með n stök í hrúgunni.

- Gerum ráð fyrir að við séum með n stök í hrúgunni.
- Þá er hæð trésins $\mathcal{O}($ $)$.

- Gerum ráð fyrir að við séum með n stök í hrúgunni.
- Þá er hæð trésins $\mathcal{O}(\log n)$.

- 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 $\mathcal{O}(\quad)$.

- 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 $\mathcal{O}(\log n)$.

- 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 $\mathcal{O}(\quad)$.

- 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 $\mathcal{O}(\log n)$.

- 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 $\mathcal{O}(\quad)$.

- 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 $\mathcal{O}(1)$.