

# Hrúgur

Bergur Snorrason

February 8, 2024

- ▶ 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  $\lfloor (i - 1)/2 \rfloor$ .



- ▶ 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  $\lfloor (i - 1)/2 \rfloor$ .
- ▶ Takið eftir í fyrri aðferðinni notum við ekki stak 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  $2 \cdot i + 1$ .
  - ▶ Hægra barn staks  $i$  er stak  $2 \cdot i + 2$ .
  - ▶ Foreldri staks  $i$  er stakið  $\lfloor (i - 1)/2 \rfloor$ .
- ▶ Takið eftir í fyrri aðferðinni notum við ekki stak 0 í fylkinu.
- ▶ Þetta má sjá sem bæði kost og galla.

- ▶ 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  $\lfloor (i - 1)/2 \rfloor$ .
- ▶ Takið eftir í fyrri aðferðinni notum við ekki stak 0 í fylkinu.
- ▶ Þetta má sjá sem bæði kost og galla.
- ▶ Það stak má nota til að geyma, til dæmis, stærðina á trénu.

- ▶ 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.

```

7
8 #define PARENT(i) ((i)/2)
9 #define LEFT(i)    ((i)*2)
10 #define RIGHT(i)   ((i)*2 + 1)
11 void swap(int* x, int* y) { int t = *x; *x = *y; *y = t; }
12 void fix_down(int *h, int i)
13 {
14     int mx = i;
15     if (RIGHT(i) <= h[0] && h[mx] < h[RIGHT(i)]) mx = RIGHT(i);
16     if (LEFT(i) <= h[0] && h[mx] < h[LEFT(i)]) mx = LEFT(i);
17     if (mx != i) swap(&h[i], &h[mx]), fix_down(h, mx);
18 }
19
20 void fix_up(int *h, int i)
21 {
22     if (i == 1 || h[i] <= h[PARENT(i)]) return;
23     swap(&h[i], &h[PARENT(i)]), fix_up(h, PARENT(i));
24 }
25
26 void pop(int *h)
27 {
28     h[1] = h[h[0]--];
29     fix_down(h, 1);
30 }
31
32 void push(int *h, int x)
33 {
34     h[++h[0]] = x;
35     fix_up(h, h[0]);
36 }
37
38 int peek(int *h) { return h[1]; }
39 int size(int *h) { return h[0]; }
40 void init(int *h) { h[0] = 0; }

```

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

- ▶ 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 lafi 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  $\mathcal{O}(1)$ .

- ▶ 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  $\mathcal{O}(1)$ .



- ▶ 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  $\mathcal{O}(1)$ .
- ▶ Við getum útfært þetta með ítrun, í stað þess að nota endurkvæmni.

```

12 void pop(int *h)
13 {
14     h[1] = h[h[0]--];
15     for (int i = 1, mx = i; ; swap(&h[i], &h[mx]), i = mx)
16     {
17         if (RIGHT(i) <= h[0] && h[mx] < h[RIGHT(i)]) mx = RIGHT(i);
18         if (LEFT(i) <= h[0] && h[mx] < h[LEFT(i)]) mx = LEFT(i);
19         if (i == mx) return;
20     }
21 }
22
23 void push(int *h, int x)
24 {
25     h[++h[0]] = x;
26     for (int i = h[0]; i > 1 && h[i] > h[PARENT(i)]; i = PARENT(i))
27         swap(&h[i], &h[PARENT(i)]);
28 }
29
30 int peek(int *h) { return h[1]; }
31 int size(int *h) { return h[0]; }
32 void init(int *h) { h[0] = 0; }

```

