Forgangsbiðraðir

Tölvunarfræði 2, vor 2012

Hallgrímur H. Gunnarsson

Háskóli Íslands

2012-03-07

Biðröð vs. forgangsbiðröð

Biðröð (FIFO)

Fyrstur inn er fyrstur út

Forgangsbiðröð

Hæsti forgangur er fyrstur út

Forgangsbiðröð

Forgangsbiðröð er gagnamót sem styður eftirfarandi aðgerðir:

- Put: setur nýtt gildi í forgangsbiðröðina
- deleteMax: tekur út stærsta gildið og skilar því

Einnig til afbrigði þar sem við höfum aðgang að minnsta gildinu og erum með deleteMin() í staðinn fyrir deleteMax()

Hugtök:

- Minimum priority queue: minnsta gildið hefur hæstan forgang
- Maximum priority queue: hæsta gildið hefur hæstan forgang

(Gerum ráð fyrir að gildin séu samanburðarhæf – Comparable)

Hagnýting

Notkun á forgangsbiðröðum:

- Hermun (e. discrete event simulation)
- Verkröðun (e. scheduling), t.d.
- - I/O verkröðun fyrir diska
- - CPU verkröðun fyrir forrit
- - Röðun pakka fyrir netkort

Líka grunnur fyrir mörg önnur reiknirit, t.d.

- Reiknirit Dijkstra til að finna stystu leið í neti
- Þjöppun með Huffman kóðum
- Lágmarks þekjutré (e. minimal spanning tree), reiknirit Prim
- Gervigreind, A* leit

Þrjár mismunandi útfærslur

Skoðum þrjár útfærslur sem byggja á ólíkri gagnaskipan:

1. Útfærsla með fylki í engri röð
2. Útfærsla með fylki í vaxandi röð
3. Útfærsla með fylki í hrúgu röð

Útfærsla 1

Útfærsla með fylki í engri röð

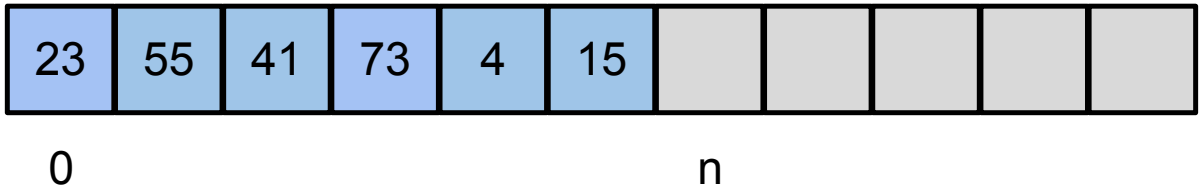
Útfærsla 1 – Gildi geymd í fylki í engri röð

```
public class PriQueue
{
    // Gildin í forgangsbiðröðinni eru í f[0..n-1]
    // í engri sérstakri röð.
    private String[] f;
    private int n;
```

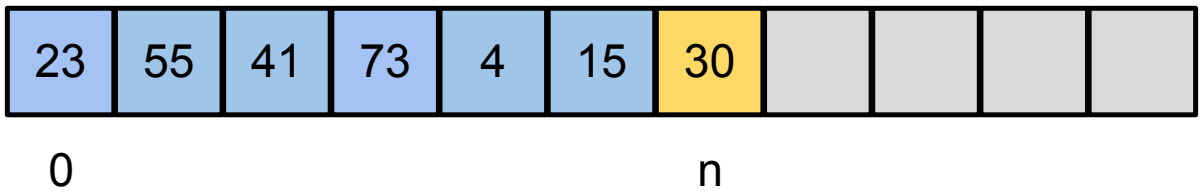
Aðgerðir:

- Put: O(1)
- deleteMax: O(n)

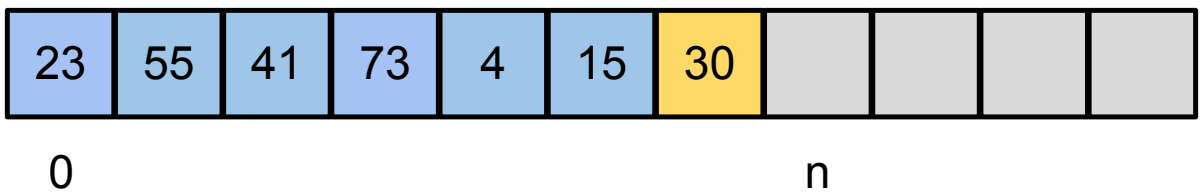
Innsetning í fylki í engri röð




```
f[n] = 30;
```



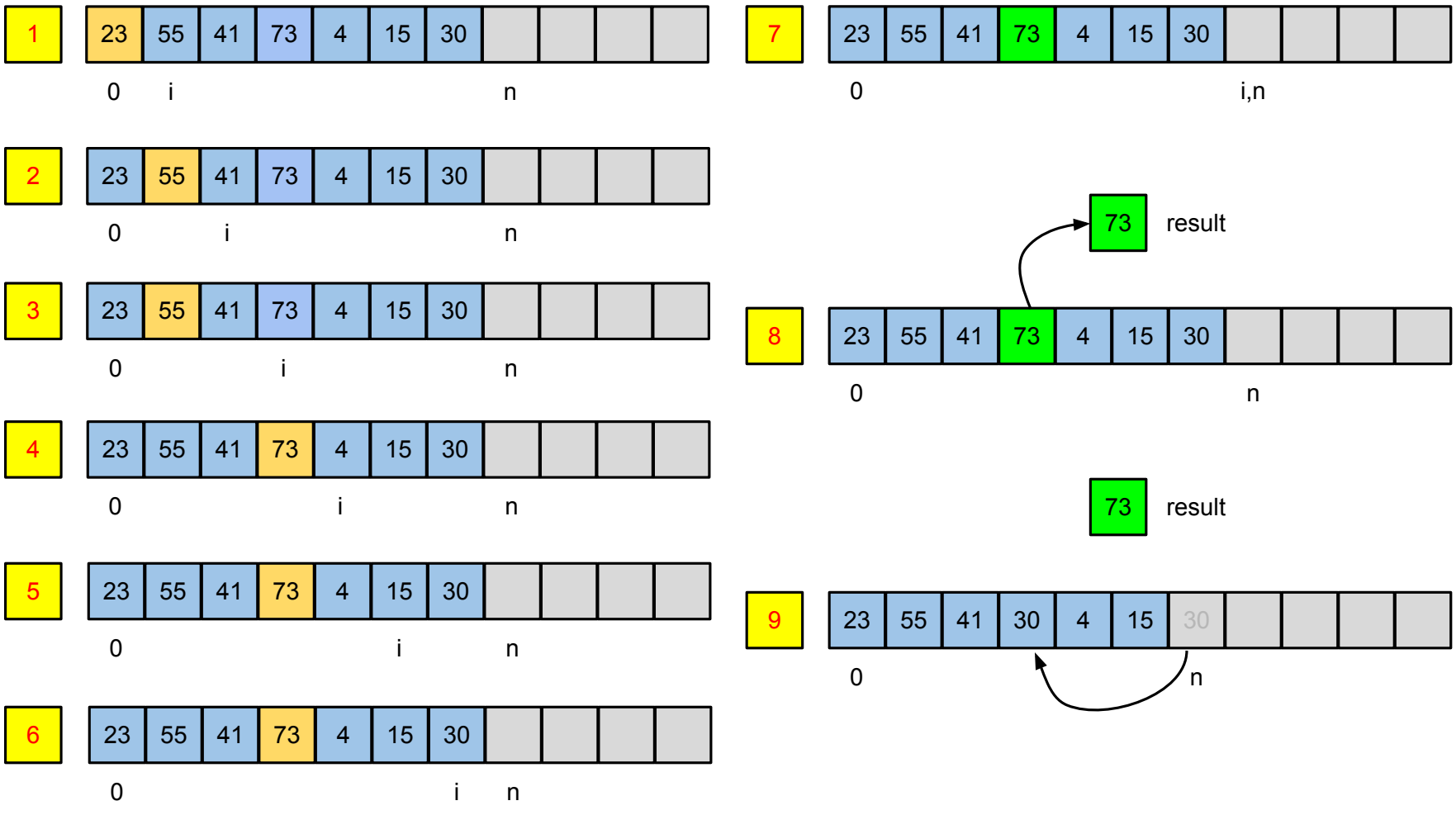

```
n++;
```



Kóði fyrir innsetningu

```
// Notkun: pq.Put(s);
// Fyrir:  pq er ekki full.
// Eftir:  s hefur verið bætt við pq.
public void Put(String s)
{
    f[n++] = s;
}
```

deleteMax fyrir fylki í engri röð



Kóði fyrir deleteMax

```
// Notkun: s = pq.deleteMax();
// Fyrir:  pq er ekki tóm.
// Eftir:  s er eitt stærsta gildið sem var
//         í pq. s hefur verið fjarlægt úr pq.
public String deleteMax()
{
    int max = 0;

    for (int i = 1; i < n; i++)
        if (f[i].compareTo(f[max]) > 0)
            max = i;

    String result = f[max];
    f[max] = f[--n];

    return result;
}
```

Útfærsla 2

Útfærsla með fylki í vaxandi röð

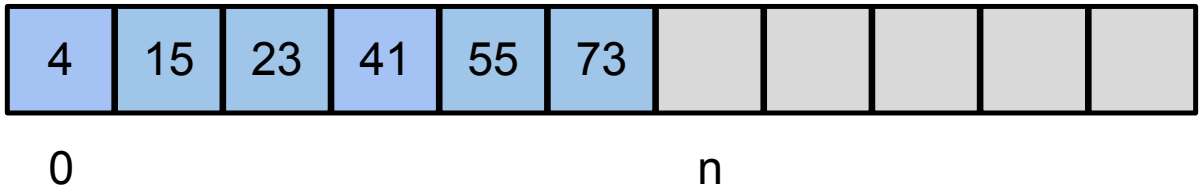
Útfærsla 2 – Gildi geymd í fylki í vaxandi röð

```
public class PriQueue
{
    // Gildin í forgangsbiðröðinni eru í f[0..n-1]
    // í vaxandi röð frá f[0] til f[n-1], þ.e.
    // f[0] er minnsta gildið og f[n-1] er stærsta.
    private String[] f;
    private int n;
```

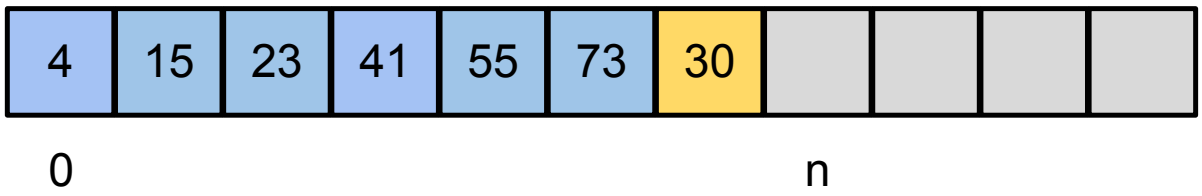
Aðgerðir:

- Put: O(n)
- deleteMax: O(1)

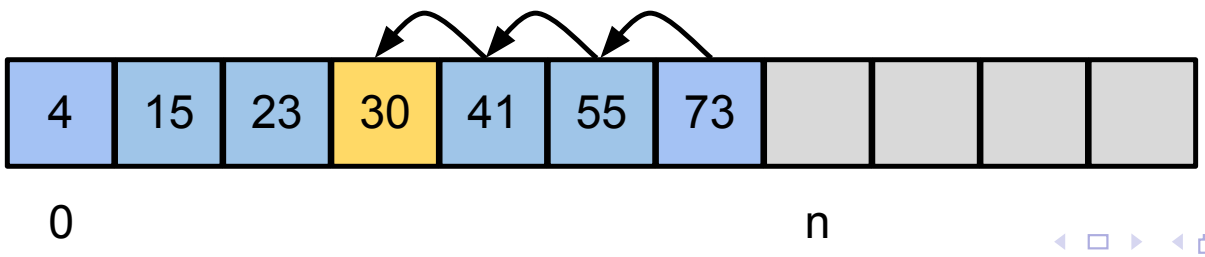
Innsetning í fylki í vaxandi röð




```
f[n++] = s;
```

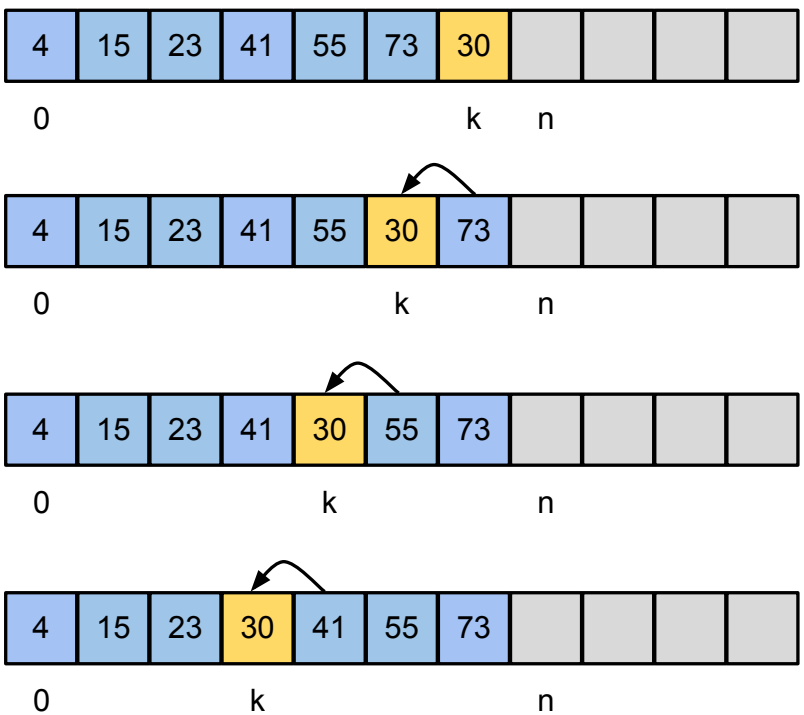



```
int k = n-1;
while (k > 0 && f[k].compareTo(f[k-1]) < 0) {
    swap(f, k-1, k);
    k--;
}
```



Innsetning í fylki í vaxandi röð

```
int k = n-1;
while (k > 0 && f[k].compareTo(f[k-1]) < 0) {
    swap(f, k-1, k);
    k--;
}
```



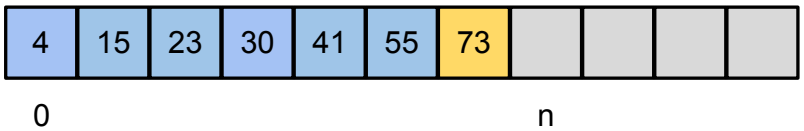
Kóði fyrir innsetningu

```
// Notkun: pq.Put(s);
// Fyrir:  pq er ekki full.
// Eftir:  s hefur verið bætt við pq.
public void Put(String s)
{
    f[n++] = s;

    int k = n-1;
    while (k > 0 && f[k].compareTo(f[k-1]) < 0)
    {
        swap(f, k-1, k);
        k--;
    }
}
```

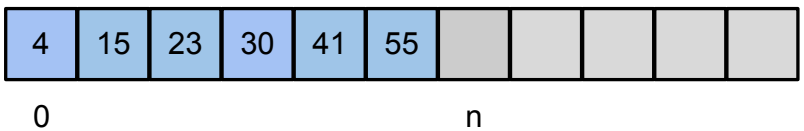
Kóði fyrir deleteMax

```
// Notkun: s = pq.deleteMax();
// Fyrir:  pq er ekki tóm.
// Eftir:  s er eitt stærsta gildið sem var
//         í pq. s hefur verið fjarlægt úr pq.
public String deleteMax()
{
    return f[--n];
}
```

Fyrir:



Eftir:



Útfærsla með hrúgu

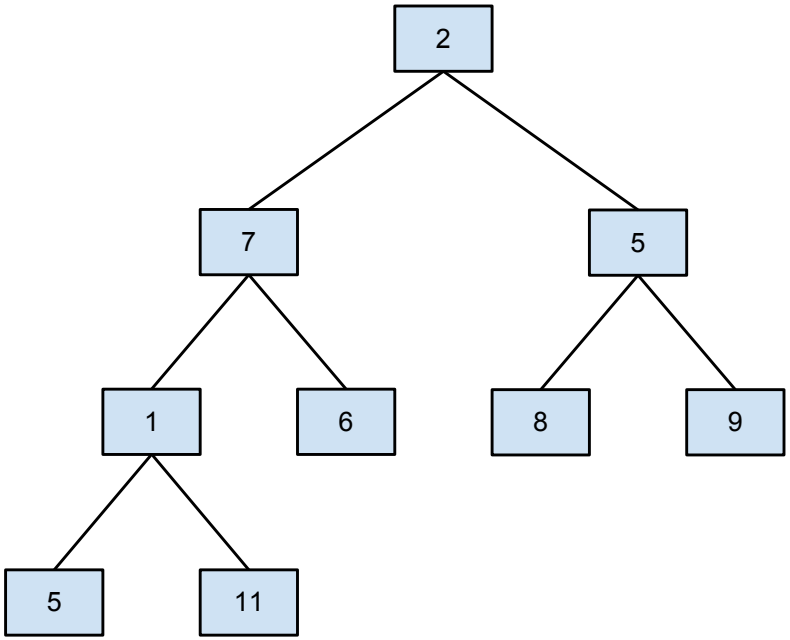
Útfærsla 3 – Gildi geymd í fylki í hrúgu

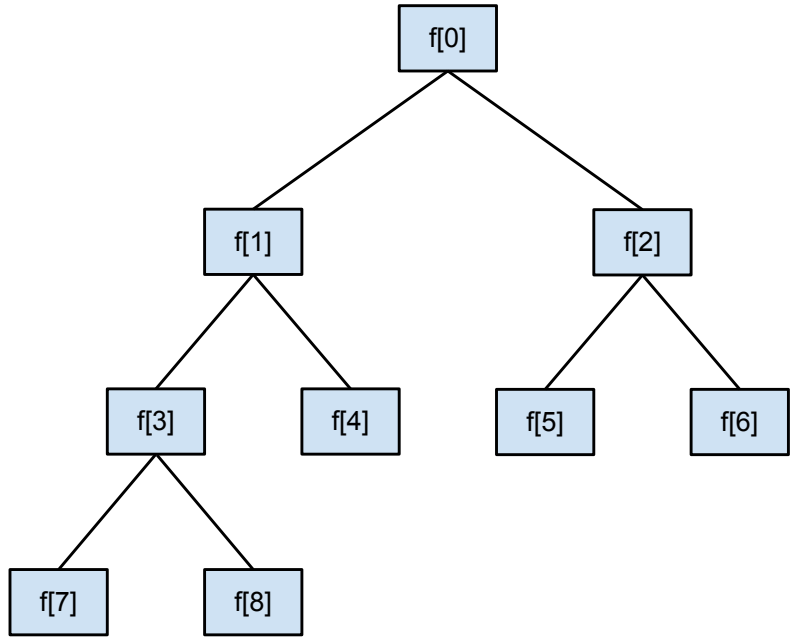
```
public class PriQueue
{
    // Gildin í forgangsbiðröðinni eru í f[0..n-1]
    // í hrúgu með stærsta stak efst
    private String[] f;
    private int n;
```

Aðgerðir:

- Put: O(log n)
- deleteMax: O(log n)

Tré geymt í fylki

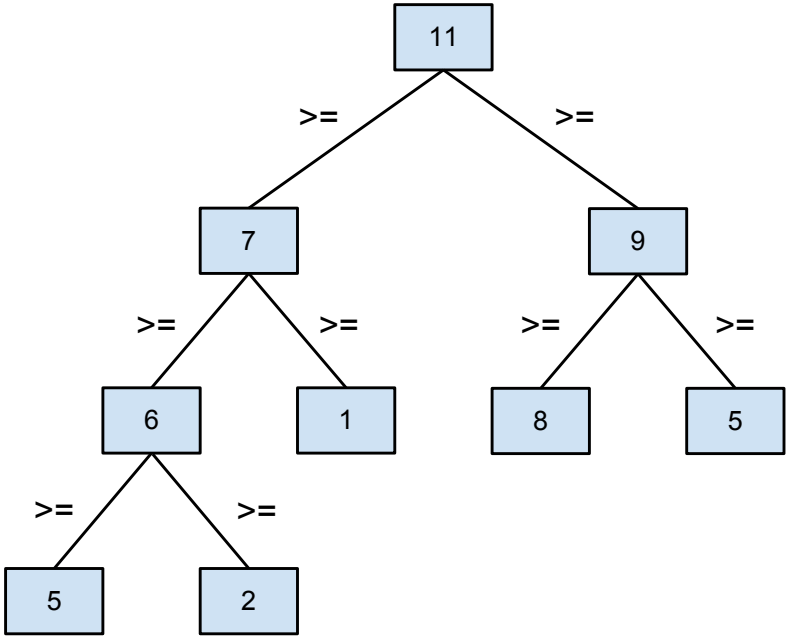


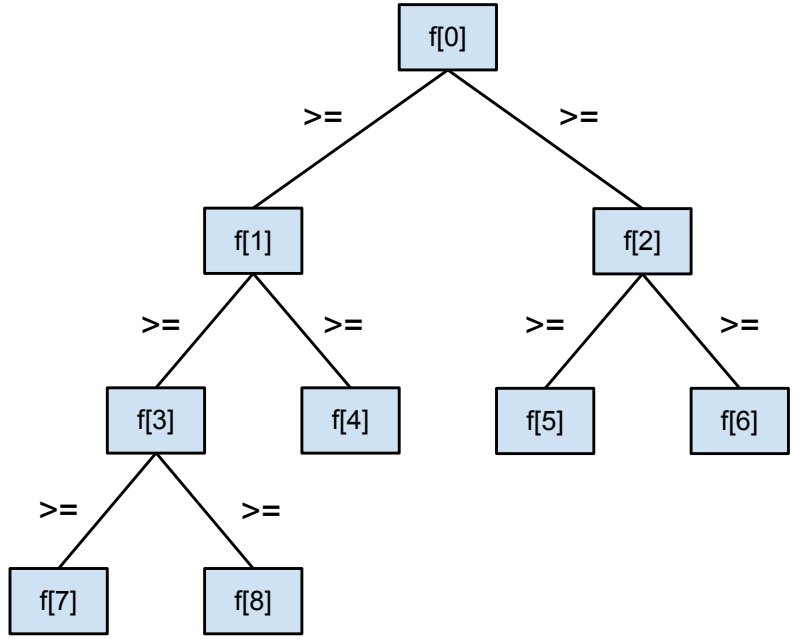


Reglur um sætaskipan:

- f[i] hefur vinstra barn f[2*i + 1] (ef það er innan fylkisins)
- f[i] hefur hægra barn f[2*i + 2] (ef það er innan fylkisins)
- f[i] hefur foreldri f[(i-1)/2] (ef það er innan fylkisins)

Hrúga í fylki með stærsta gildi efst (max heap)



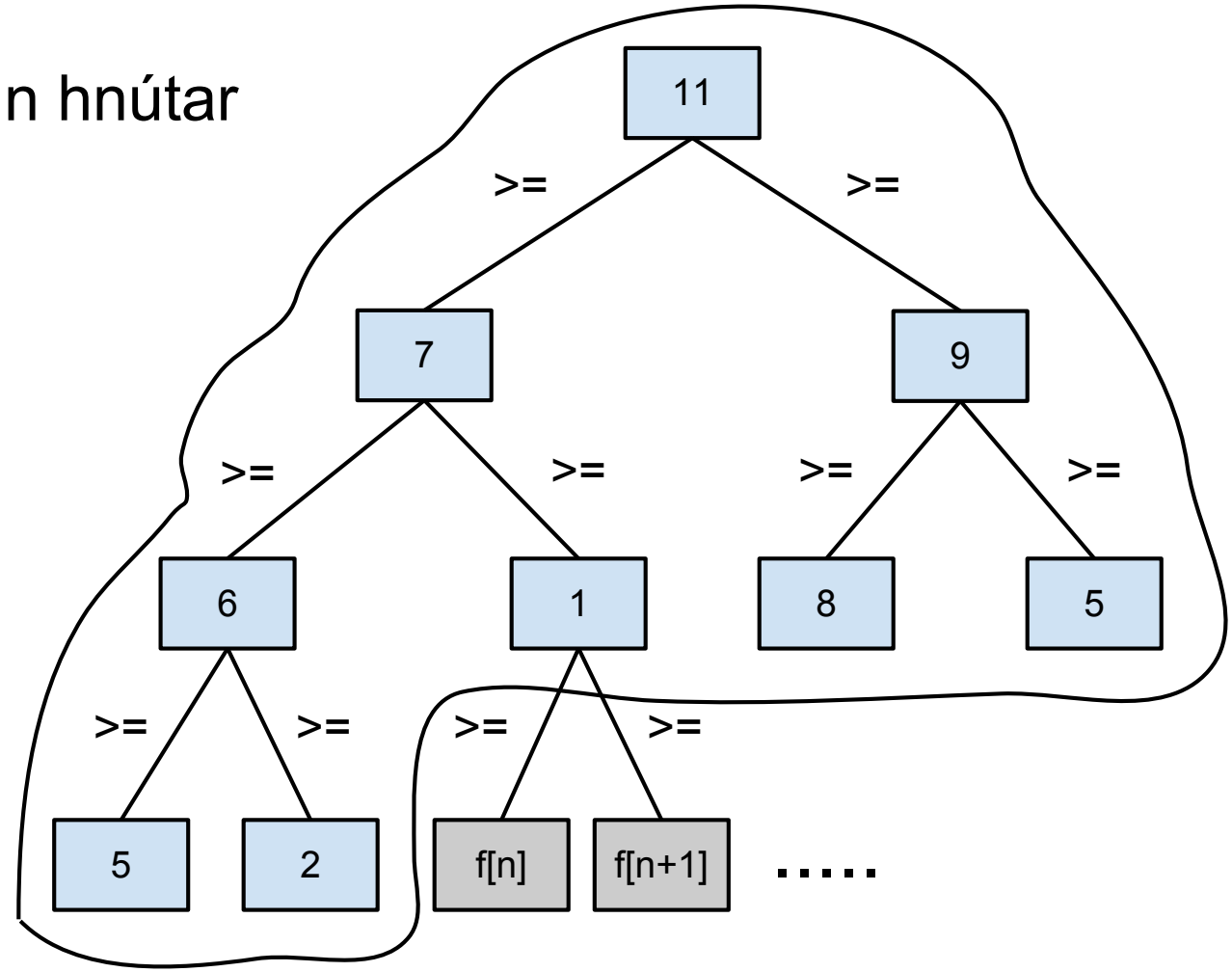


Reglur um röð (heap order):

- f[i] >= f[2*i + 1] (foreldri stærra en vinstra barn)
- f[i] >= f[2*i + 2] (foreldri stærra en hægra barn)
- f[i] <= f[(i-1)/2] (barn minna en foreldri)

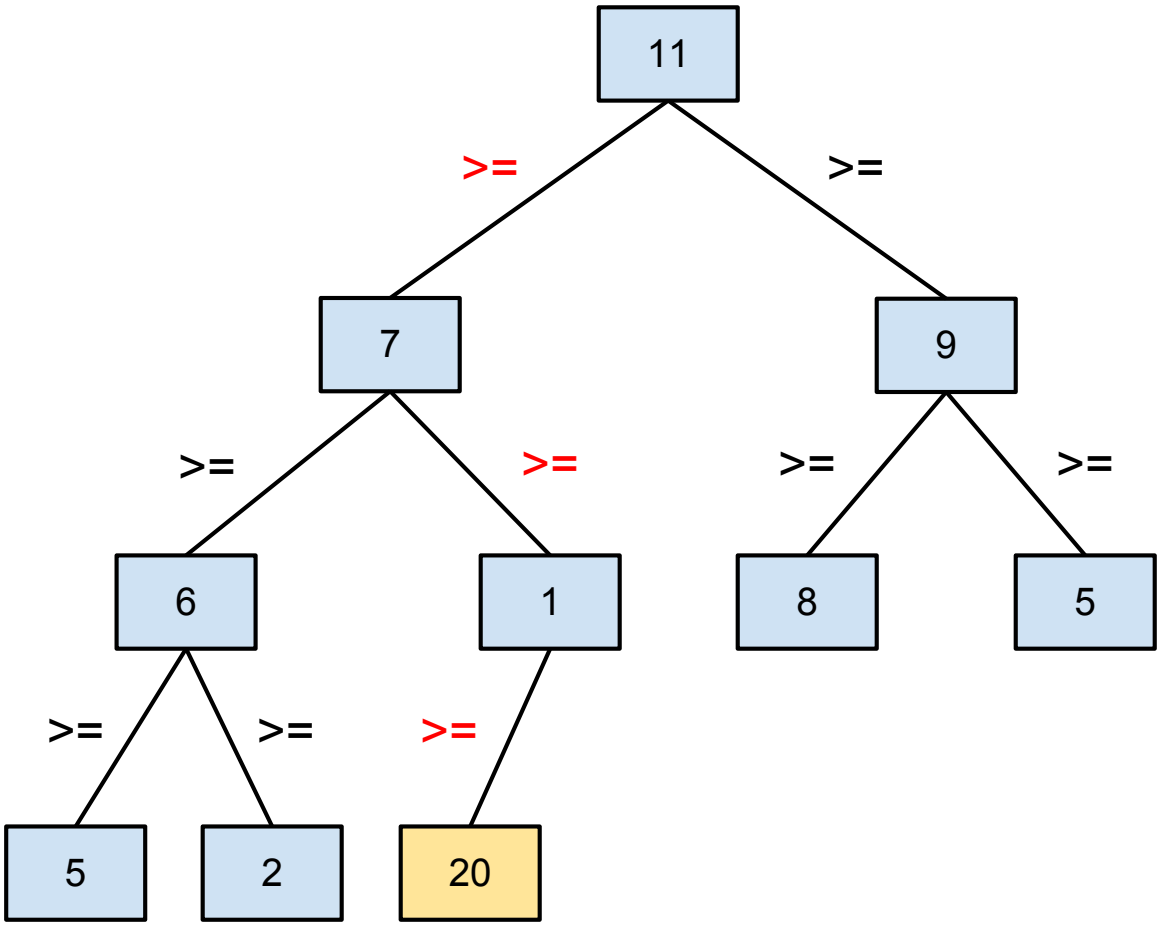
Neðst til hægri í hrúgunni / aftast í fylkinu

f[0..n-1] er hrúga:



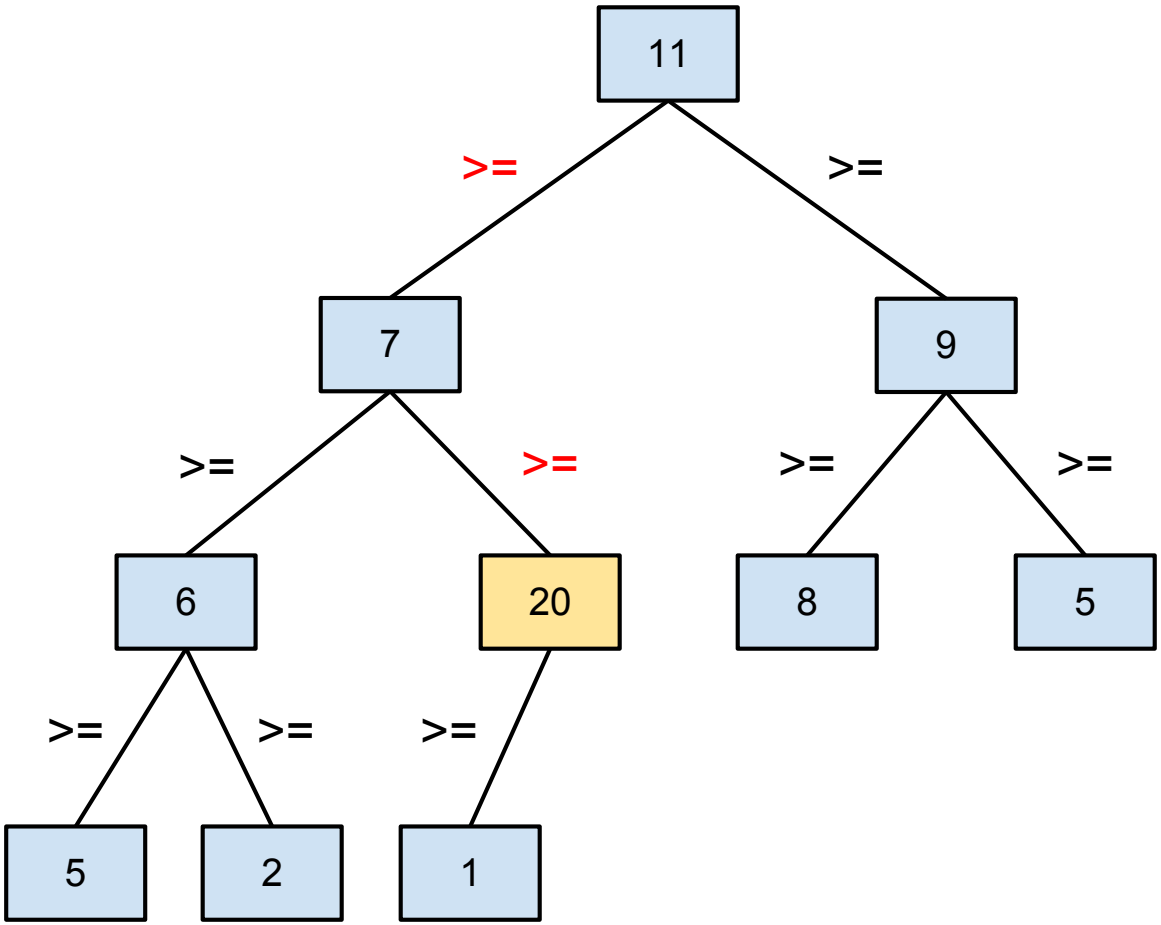
Innsetning í hrúgu

Bætum gildinu neðst til hægri (aftast í fylkið) og lagfærum síðan hrúguna.



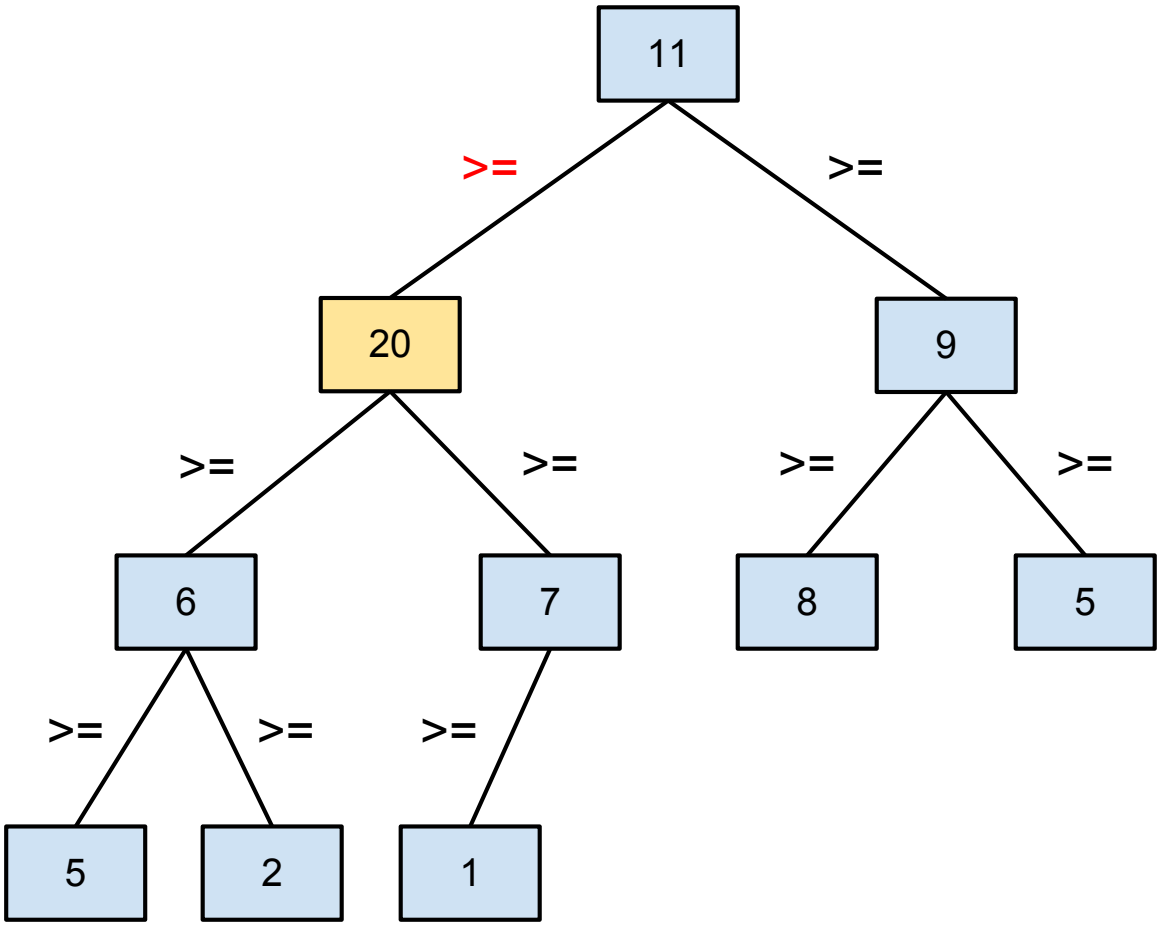
Innsetning í hrúgu

Bætum gildinu neðst til hægri (aftast í fylkið) og lagfærum síðan hrúguna.



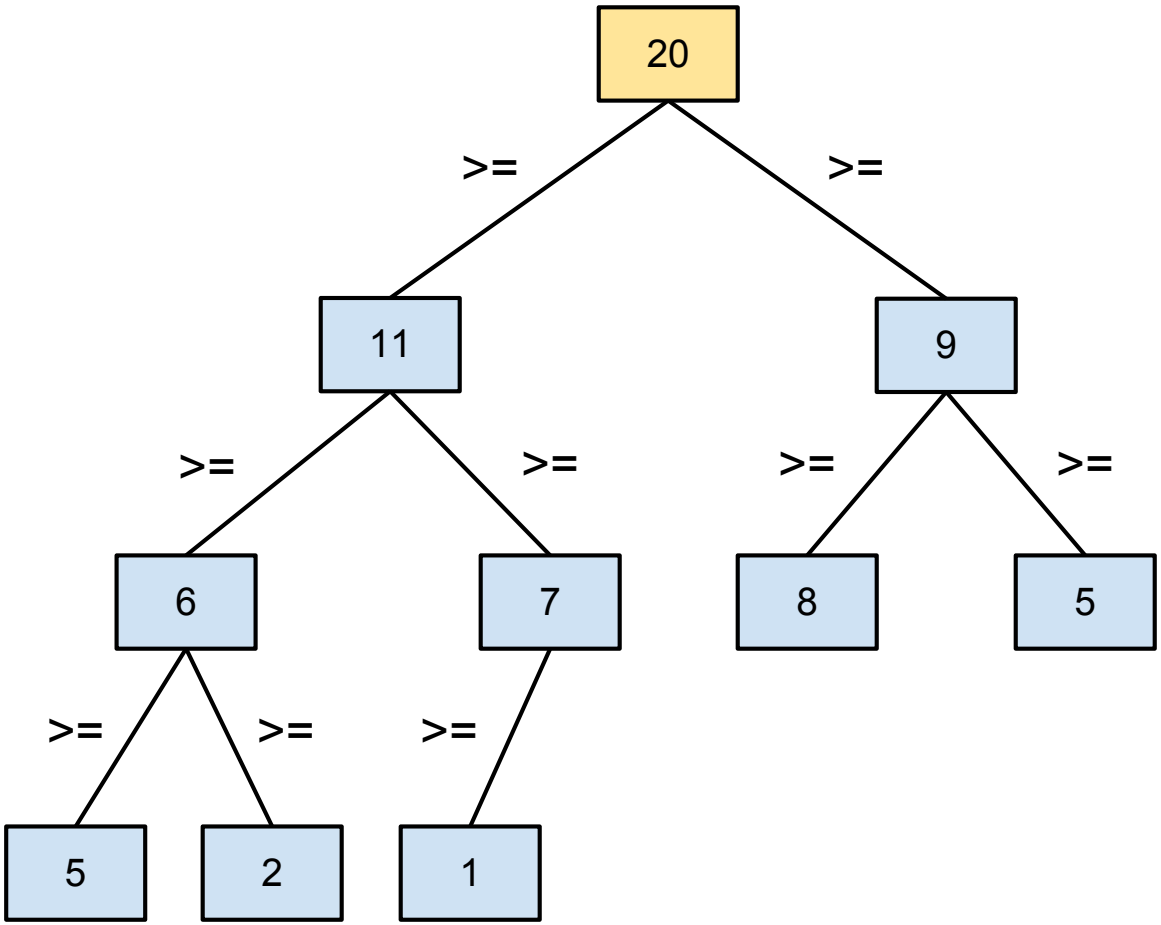
Innsetning í hrúgu

Bætum gildinu neðst til hægri (aftast í fylkið) og lagfærum síðan hrúguna.



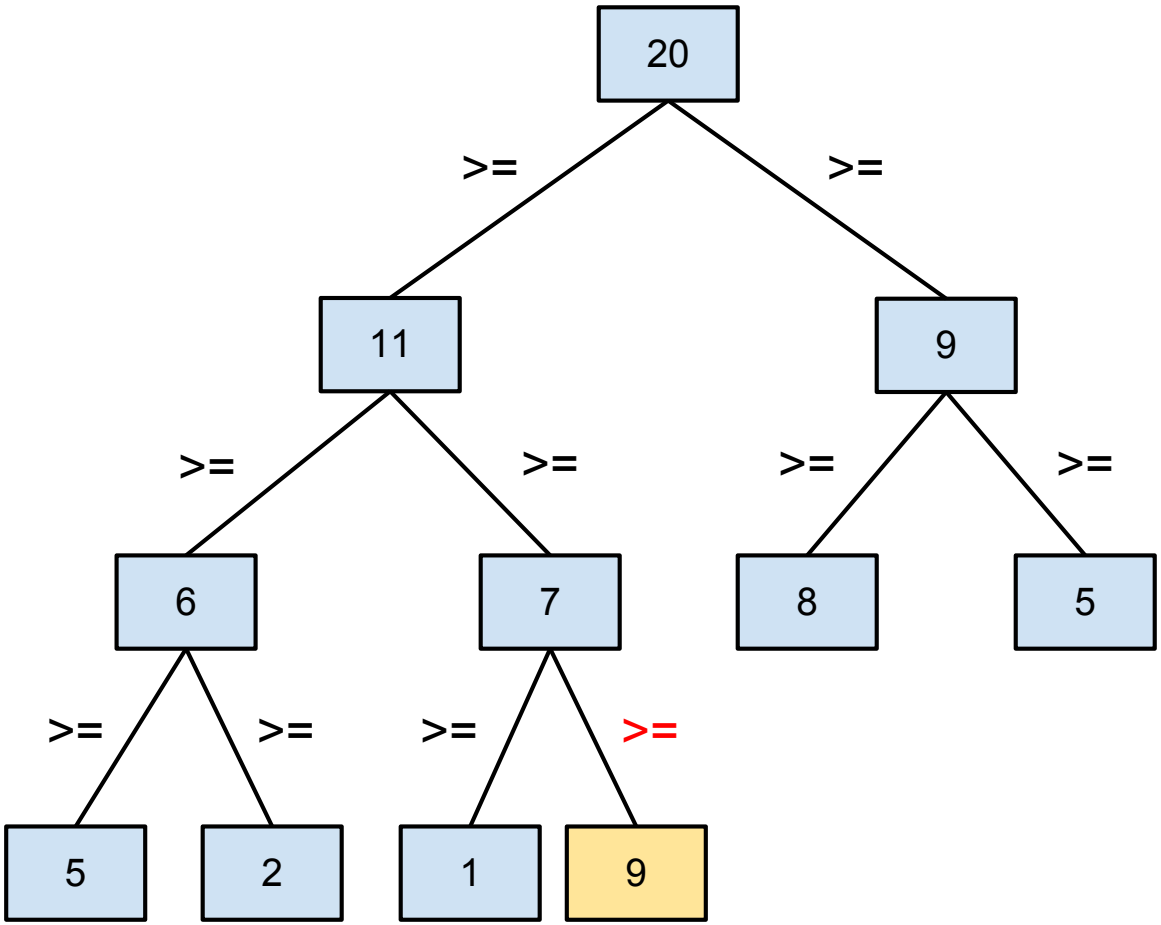
Innsetning í hrúgu

Bætum gildinu neðst til hægri (aftast í fylkið) og lagfærum síðan hrúguna.



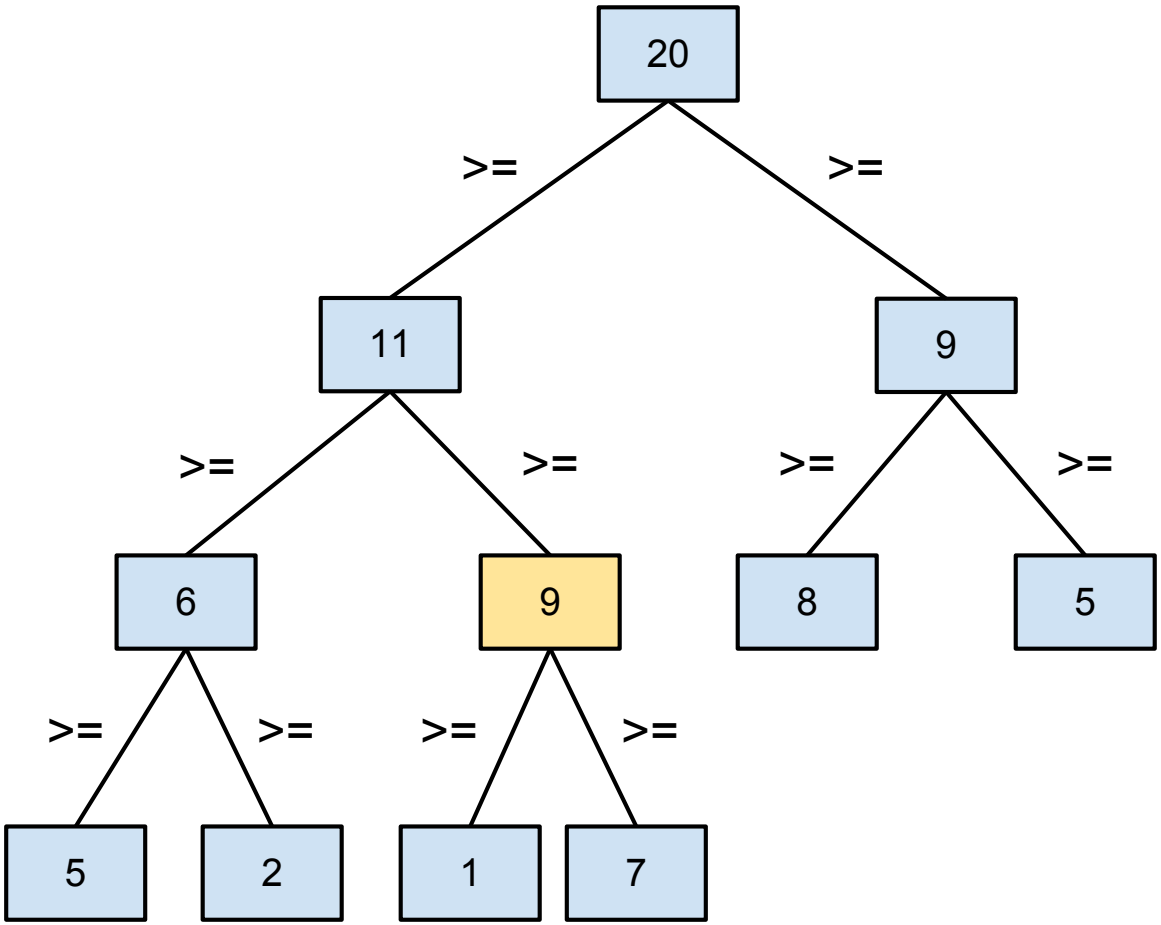
Önnur innsetning í hrúgu

Bætum gildinu neðst til hægri (aftast í fylkið) og lagfærum síðan hrúguna.



Önnur innsetning í hrúgu

Bætum gildinu neðst til hægri (aftast í fylkið) og lagfærum síðan hrúguna.



Tímaflækjan fyrir innsetningu

Hrúga er alltaf fullskipað tré.

Fullskipað tré með n gildum hefur hæð $\lfloor \log_2 n \rfloor$

Innsetning víxlar einu gildi upp tréð, hver víxlun færir gildið upp um eina hæð

$\Longrightarrow$ Tímaflækjan fyrir innsetningu er O(log n)

Binary heap demo

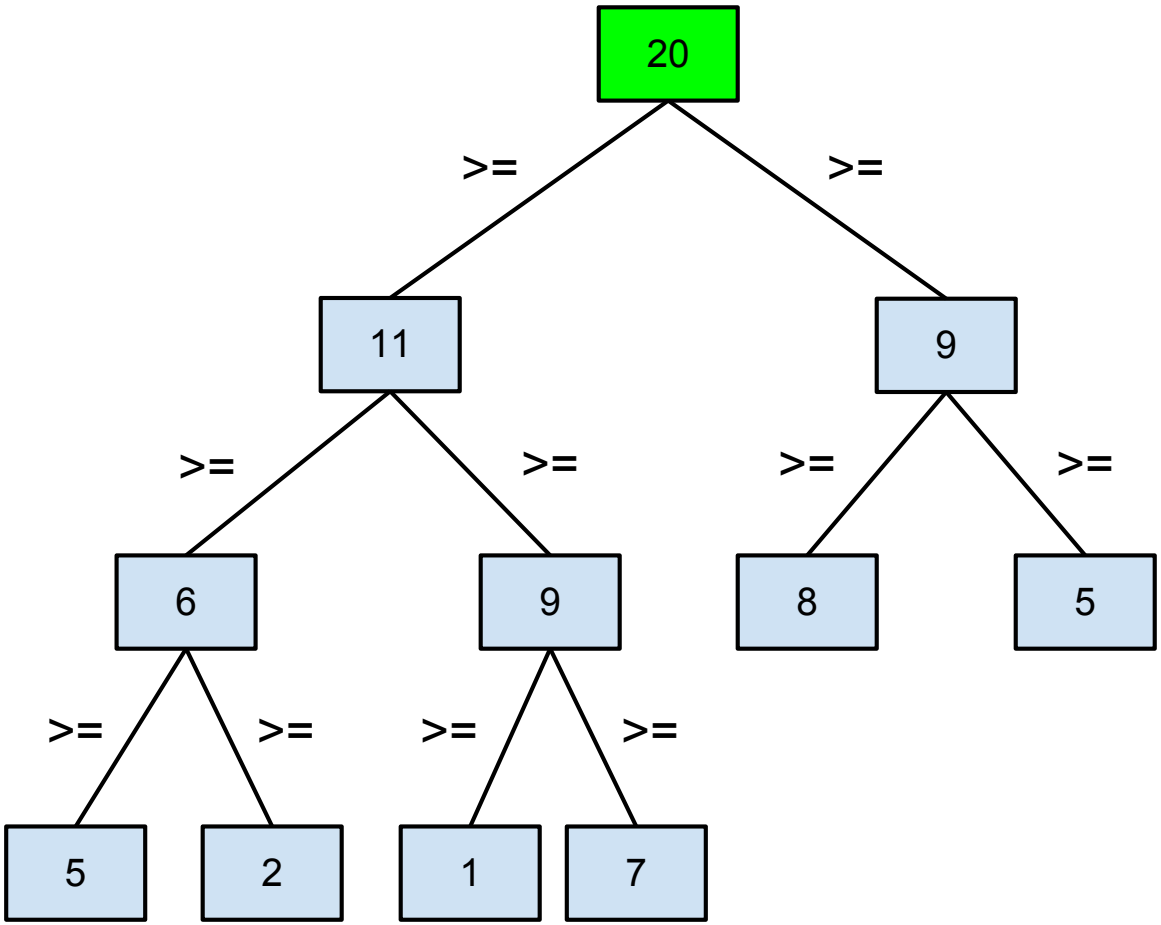
http://algs4.cs.princeton.edu/lectures/24DemoBinaryHeap.mov

Kóði fyrir innsetningu

```
public void Put(String s)
{
    f[n++] = s;

    int i = n-1;

    while (i > 0) {
        // f[0..n-1] er í hrúgu nema e.t.v.
        // er f[i] of neðarlega.

        int parent = (i-1)/2;

        if (f[parent].compareTo(f[i]) >= 0)
            break;

        swap(f, i, parent);
        i = parent;
    }
}
```

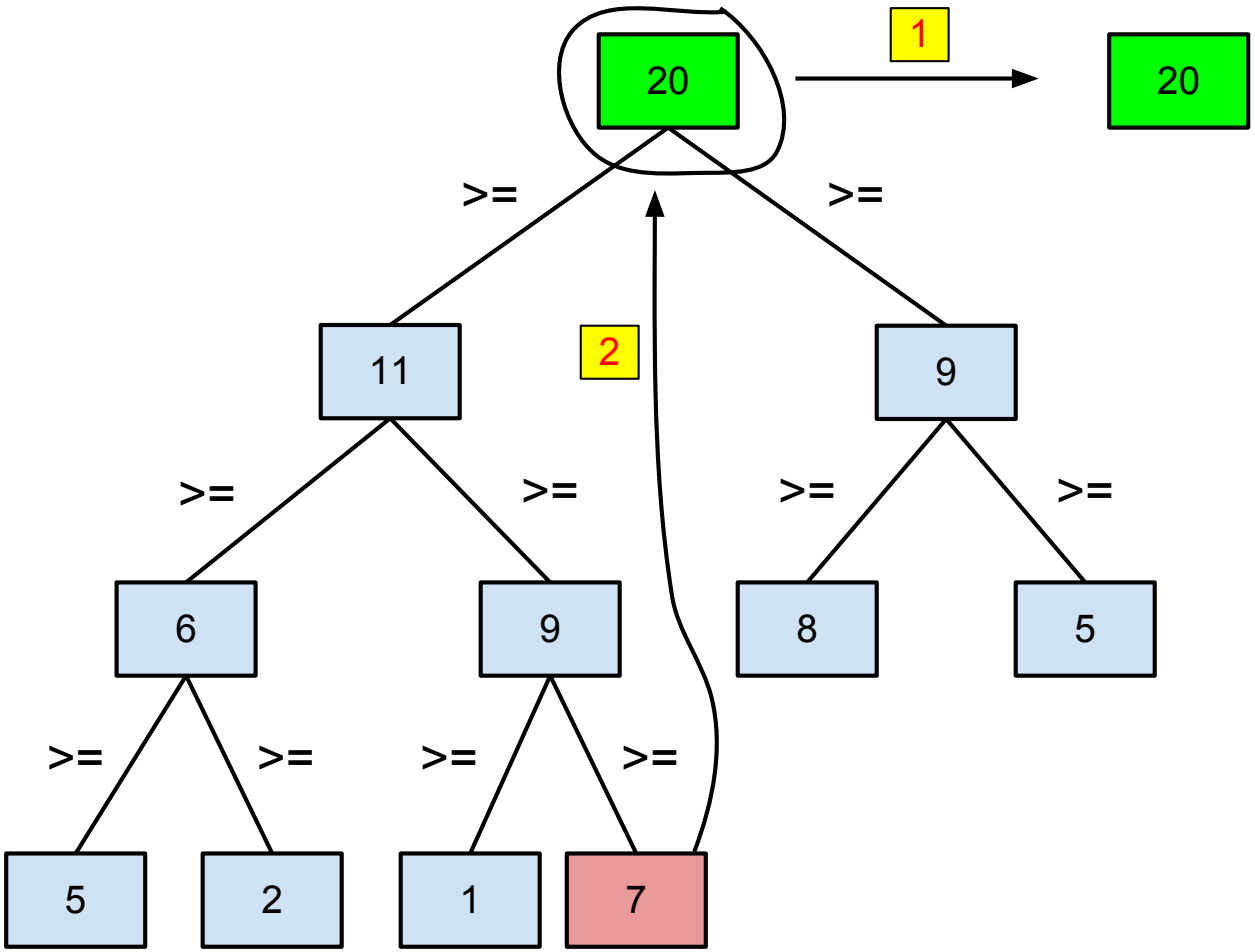
Taka út stærsta gildið

Stærsta gildið er alltaf efst í hrúgunni, en hvernig fjarlægjum við það úr hrúgunni?



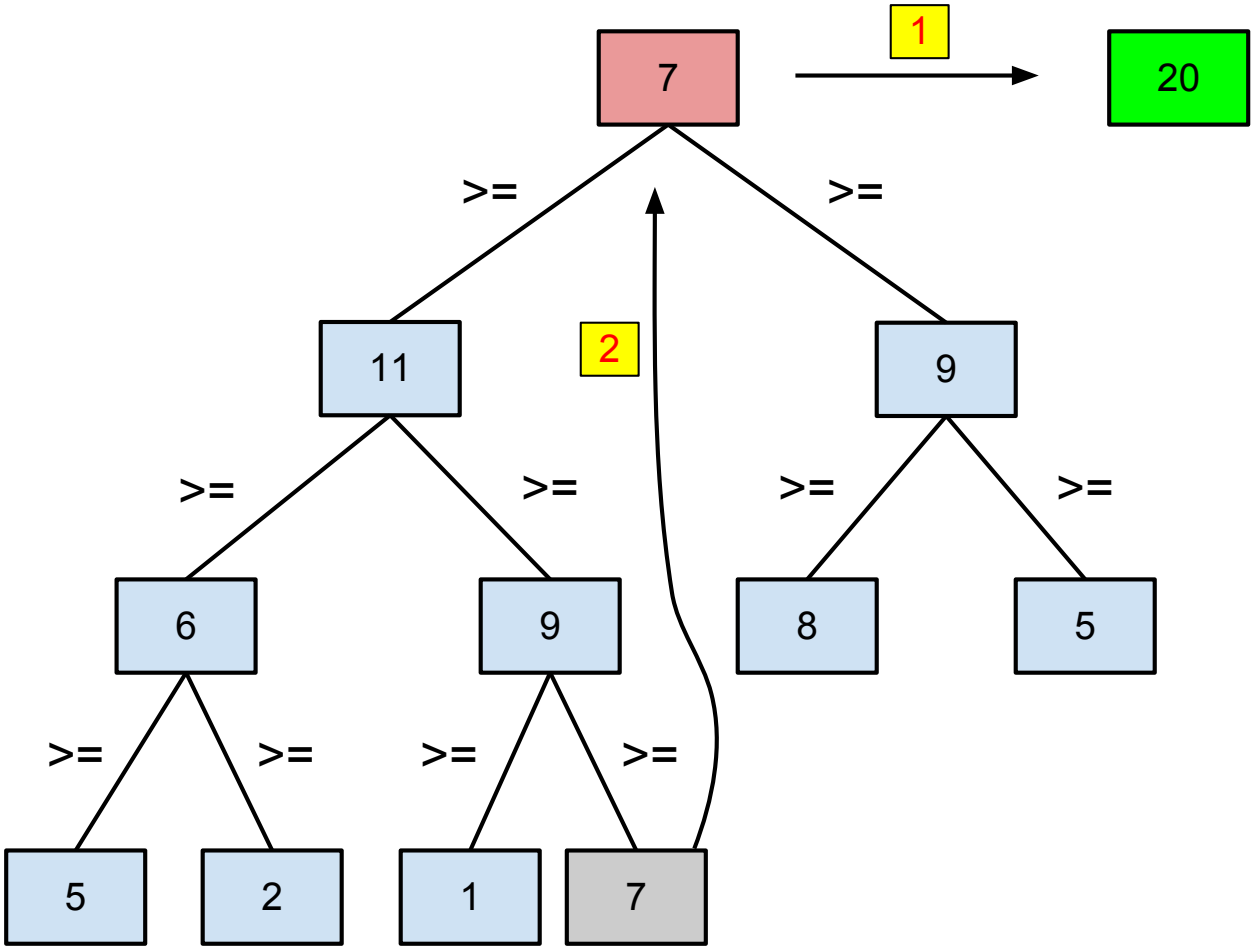
Taka út stærsta gildið

Tökum út efsta gildið, færum neðsta gildið efst og lagfærum síðan hrúguna



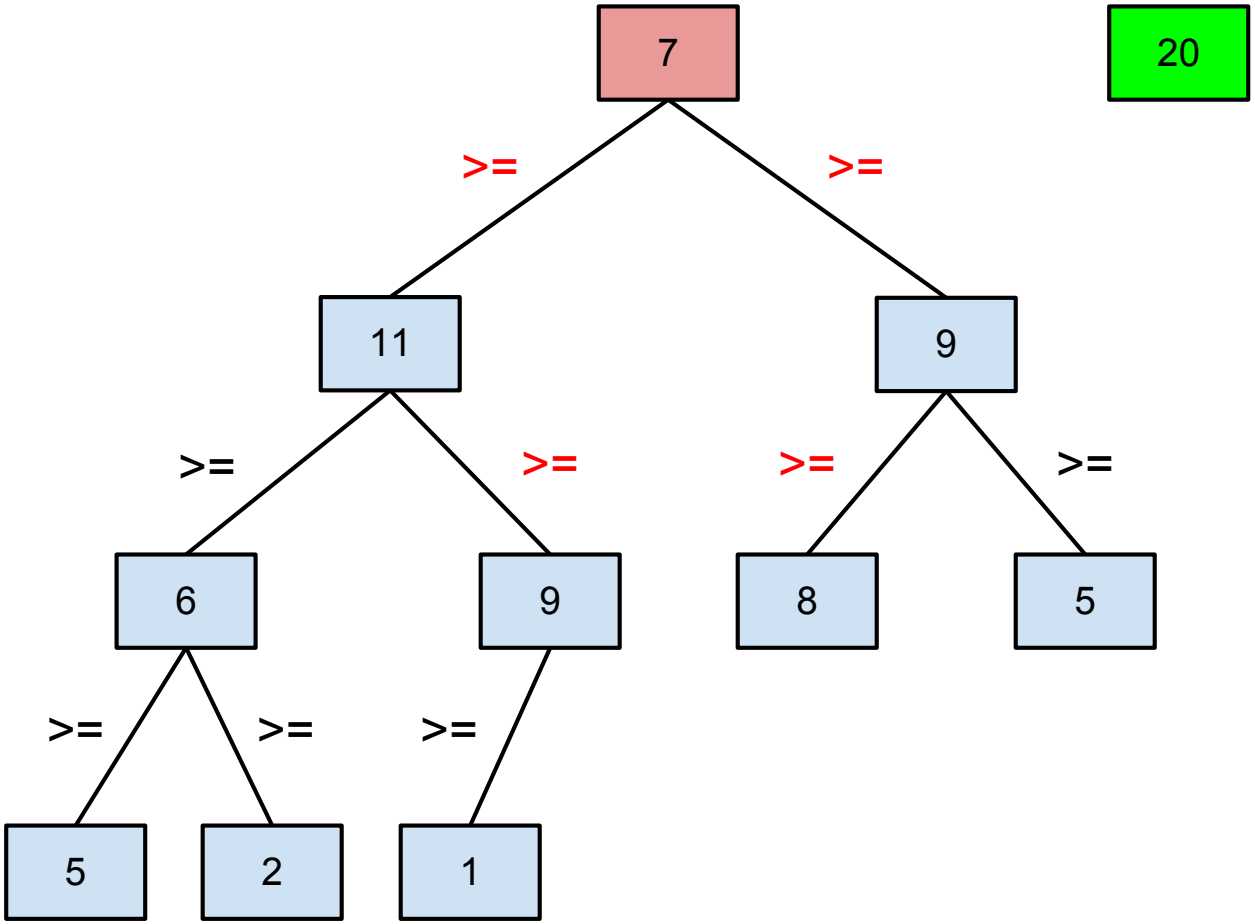
Taka út stærsta gildið

Tökum út efsta gildið, færum neðsta gildið efst og lagfærum síðan hrúguna



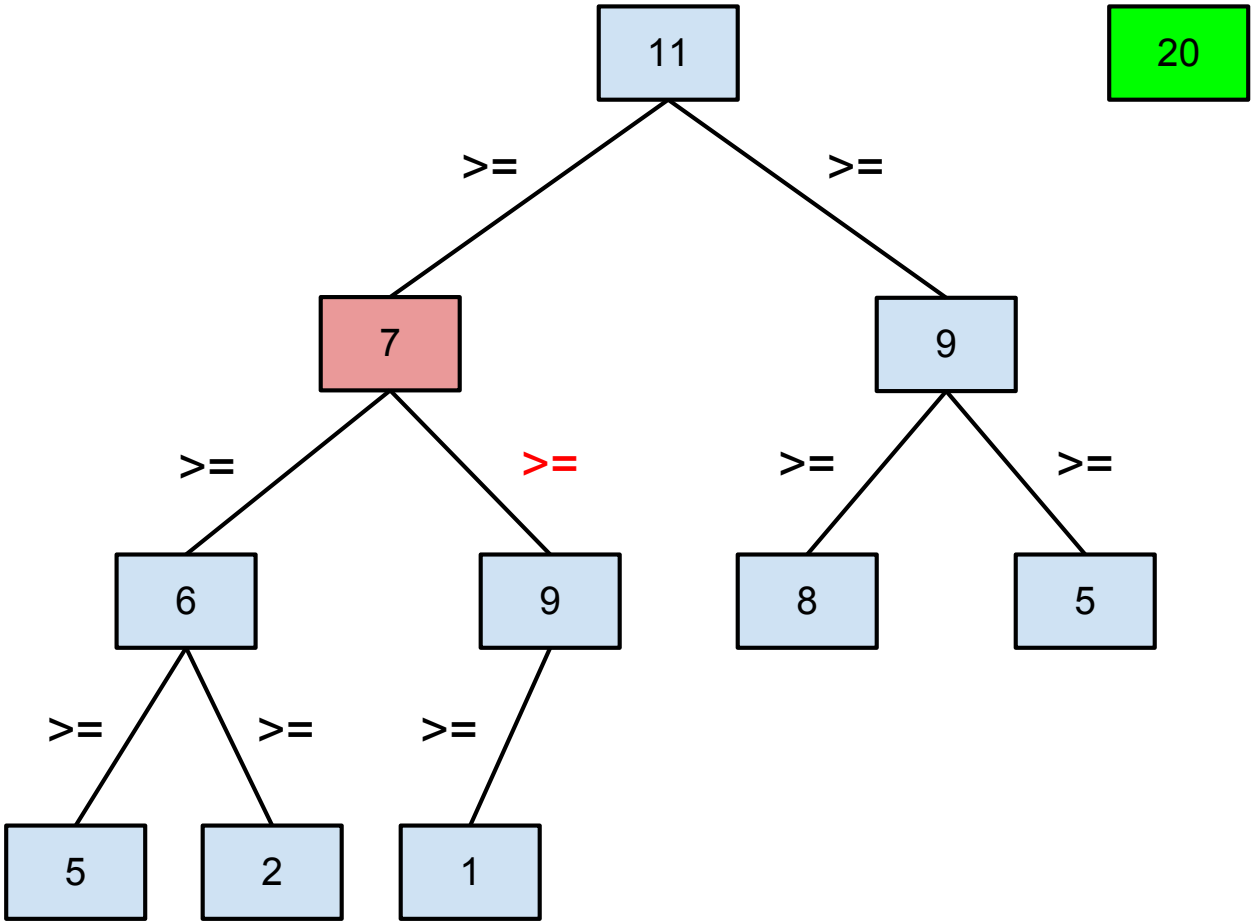
Taka út stærsta gildið

Víxlum alltaf við stærra barnið



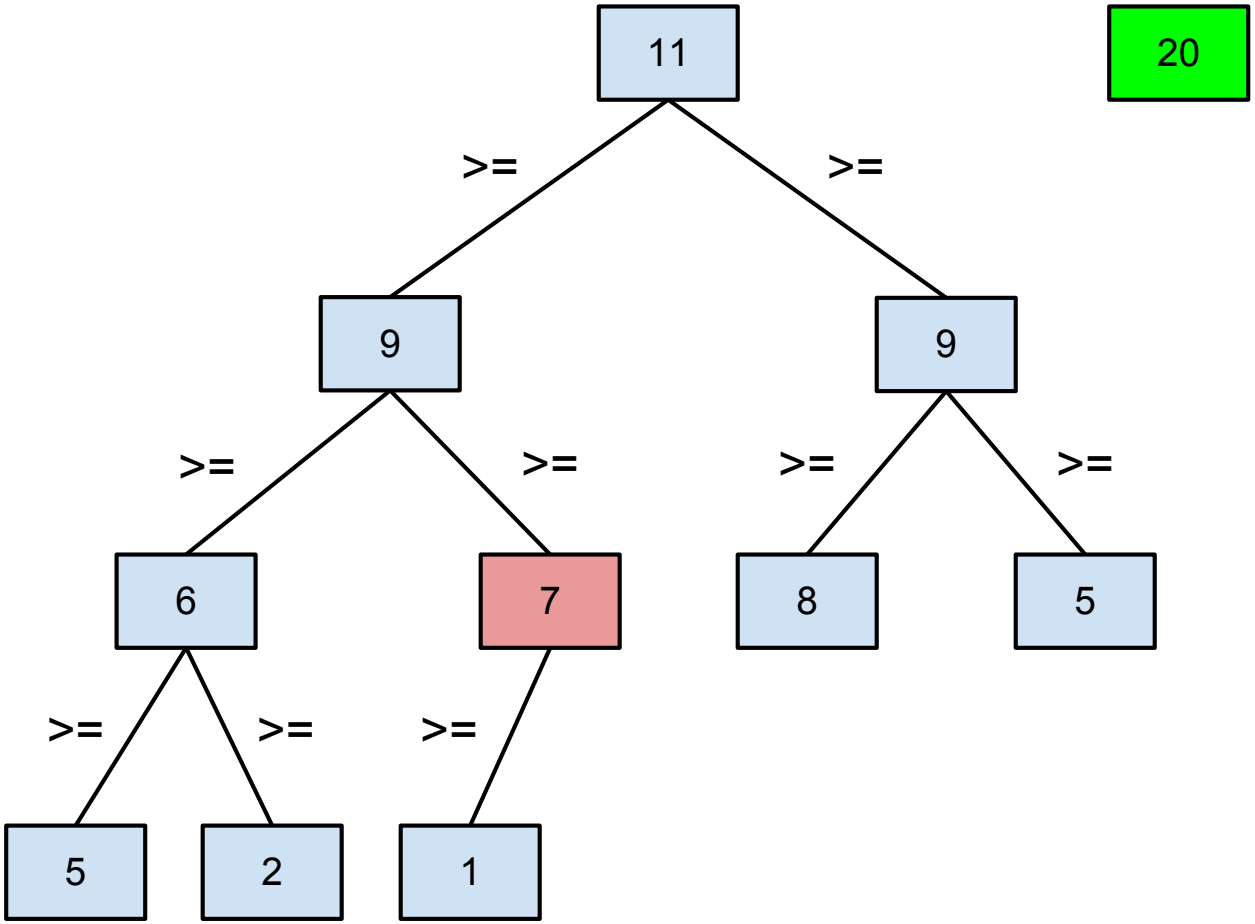
Taka út stærsta gildið

Víxlum niður



Taka út stærsta gildið

Hrúguskilyrðið uppfyllt á nýjan leik, fastayrðing gagna sönn



Kóði fyrir deleteMax

```
public String deleteMax()
{
    // Geymum gildið í rótinni (stærsta gildið)
    String result = f[0];

    // Færum neðsta gildið efst í rótina
    f[0] = f[--n];

    // Víxlum gildinu niður til að lagfæra hrúguna
    // ... framhald á næstu glæru
```

Kóði fyrir deleteMax framhald

```
int i = 0;
while (true) {
    // f[0..n-1] er hrúga nema e.t.v.
    // er f[i] of ofarlega.
    // 0 <= i <= n

    int b = 2*i+1;

    if (b >= n)
        // f[i] er barnlaust (lauf)
        return result;

    if (b+1 < n && f[b+1].compareTo(f[b]) > 0)
        // Veljum hægra barnið ef það er stærra
        b++;
```

Kóði fyrir deleteMax framhald

```
            if (f[i].compareTo(f[b]) >= 0)
                // f[i] er á réttum stað
                return result;

            // f[i] < f[b], þ.e. foreldrið er minna
            // en barnið og þurfum því að víxla þeim.

            swap(f, i, b);
            i = b;
        }
    }
```

deleteMax í heild sinni

```
public String deleteMax()
{
    String result = f[0];

    f[0] = f[--n];

    int i = 0;
    while (true) {
        // f[0..n-1] er hrúga nema e.t.v.
        // er f[i] of ofarlega.
        // 0 <= i <= n

        int b = 2*i+1;

        if (b >= n)
            return result;

        if (b+1 < n && f[b+1].compareTo(f[b]) > 0)
            b++;

        if (f[i].compareTo(f[b]) >= 0)
            return result;

        swap(f, i, b);
        i = b;
    }
}
```

Tímaflækjan fyrir deleteMax

Sambærileg röksemdafærsla og fyrir innsetningu

Hrúga er alltaf fullskipað tré.

Fullskipað tré með n gildum hefur hæð $\lfloor \log_2 n \rfloor$

deleteMax víslar einu gildi niður tréð, hver víxlun fyrir gildið niður um eina hæð

$\Longrightarrow$ Tímaflækjan fyrir deleteMax er O(log n)

Spurningar

Spurningar?