Heapsort

Tölvunarfræði 2, vor 2012

Hallgrímur H. Gunnarsson

Háskóli Íslands

2012-03-09

Röðun með forgangsbiðröð

```
public static void pqsort(String[] f)
{
    PriQueue pq = new PriQueue(f.length);

    for (int i = 0; i < f.length; i++)
        pq.Put(f[i]);

    for (int i = 0; i < f.length; i++)
        f[i] = pq.deleteMin();
}
```

Tímaflækja

Forsendur:

- Put er O(log n) aðgerð
- deleteMin er O(log n) aðgerð

Tímaflækja fyrir röðun n gilda með forgangsbiðröð:

- n $\times$ Put aðgerðir, sem gerir O(n log n)
- n $\times$ deleteMin aðgerðir, sem gerir O(n log n)

Heildartímaflækja er O(n log n), en röðunin tekur O(n) minni.

Spurning:

Getum við verið sniðug og raðað innan fylkisins án auka minnis?

Heapsort

Tvö skref:

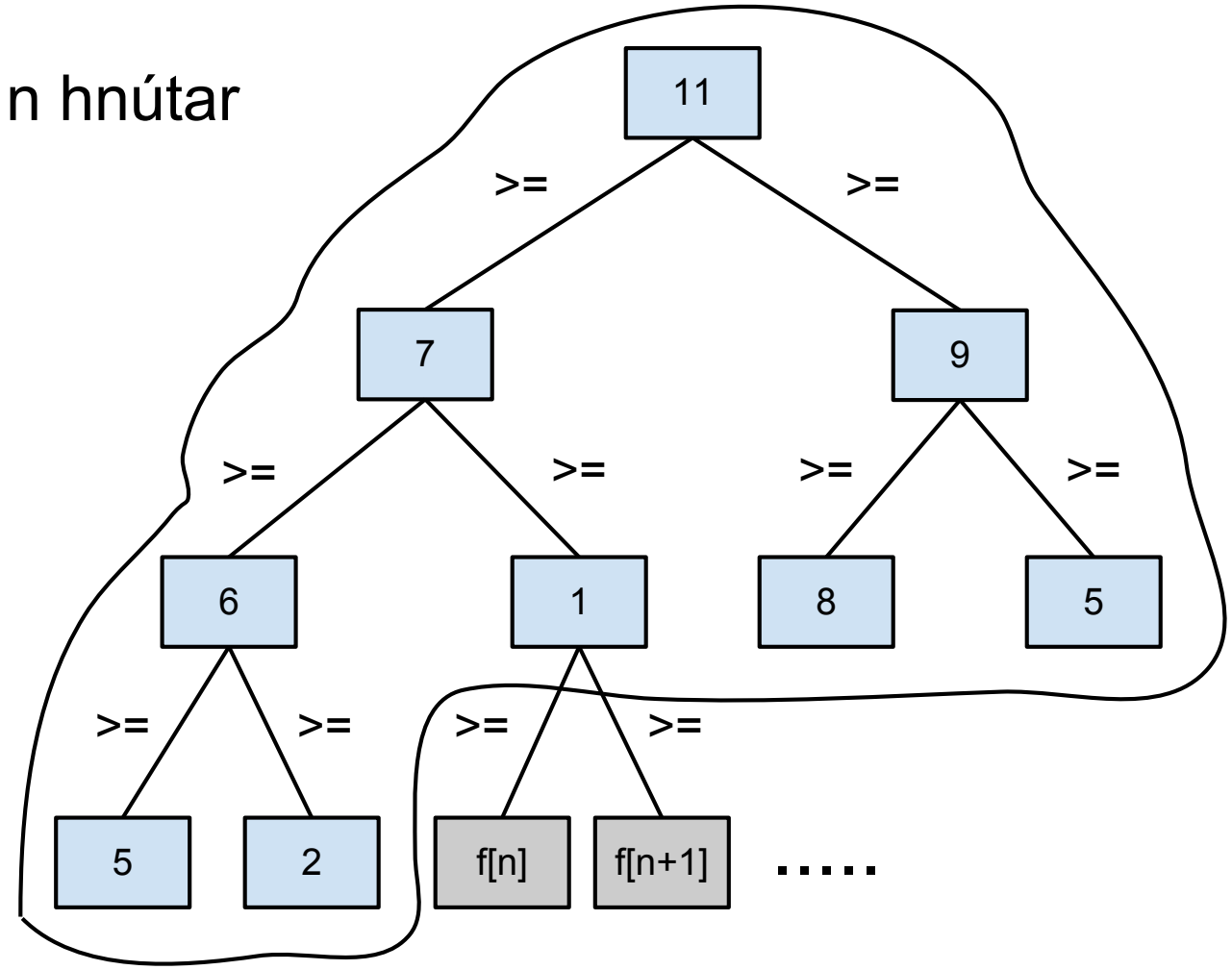
1. Gerum hrúgu
2. Röðum upp úr hrúgunni

Við skulum byrja á því að skoða skref númer 2 – hvernig við getum raðað í vaxandi röð upp úr hrúgu með stærsta stak efst án þess að taka neitt auka minni.

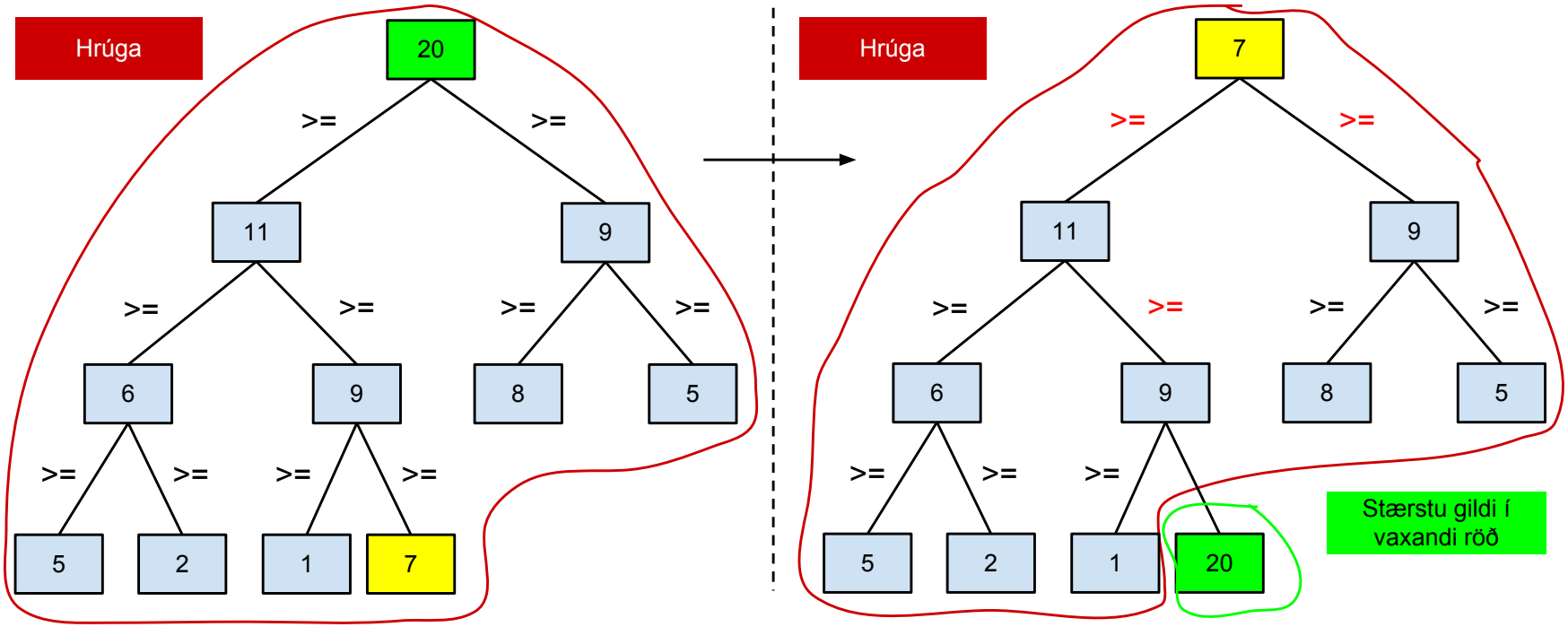
Síðan skoðum við tvær mismunandi aðferðir til að byggja hrúgu án þess að taka neitt auka minni.

Röðun upp úr hrúgu

Hrúga

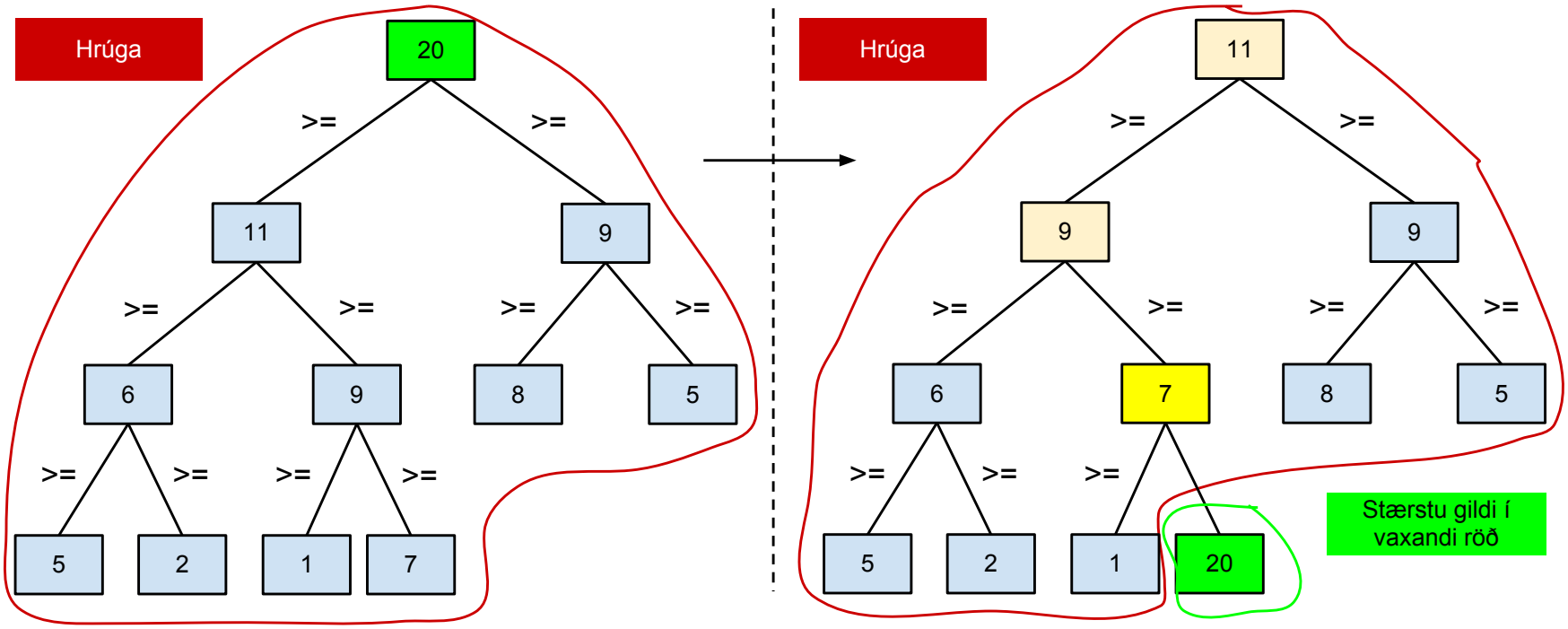


Röðun með hrúgu



Röðun með hrúgu

Tökum stærsta gildið og víxlum því neðst í hrúguna. Minnkum hrúgusvæðið um einn og lagfærum hrúguna.

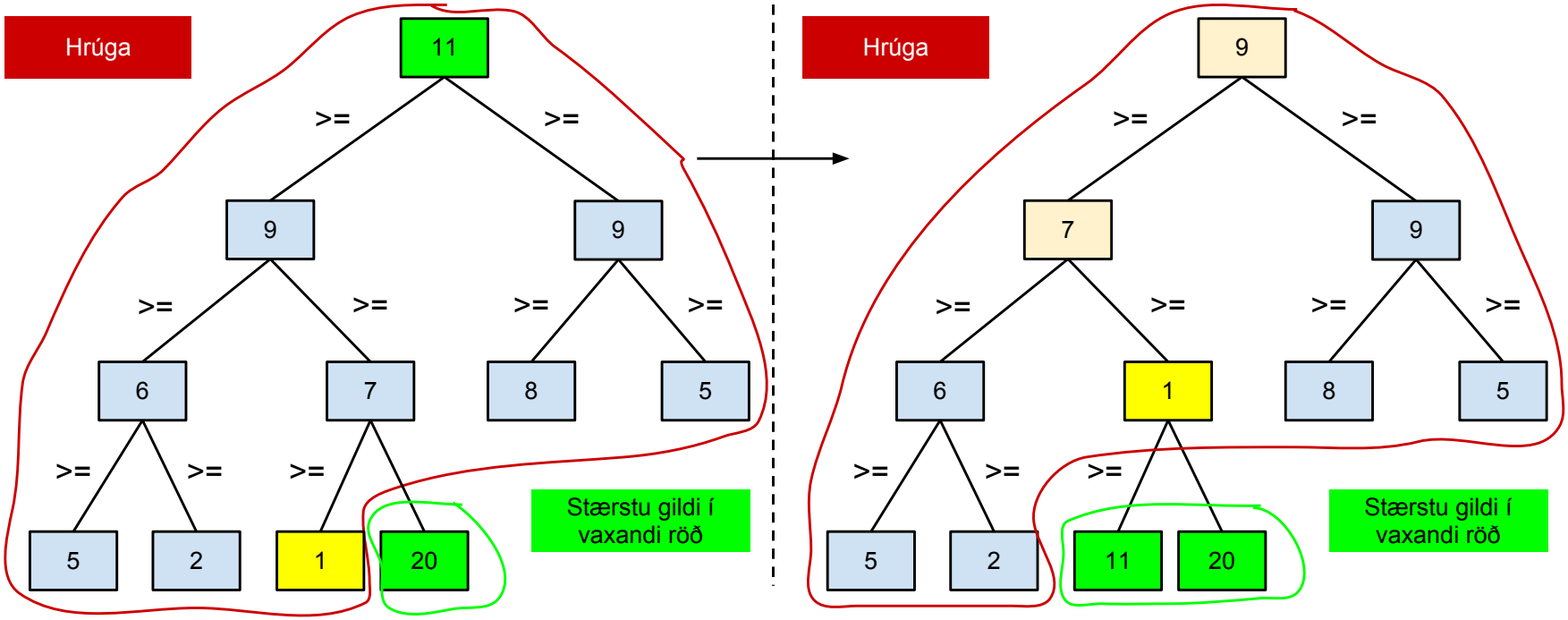


Við höldum utan um sérstakt svæði aftast sem inniheldur stærstu gildi í vaxandi röð.

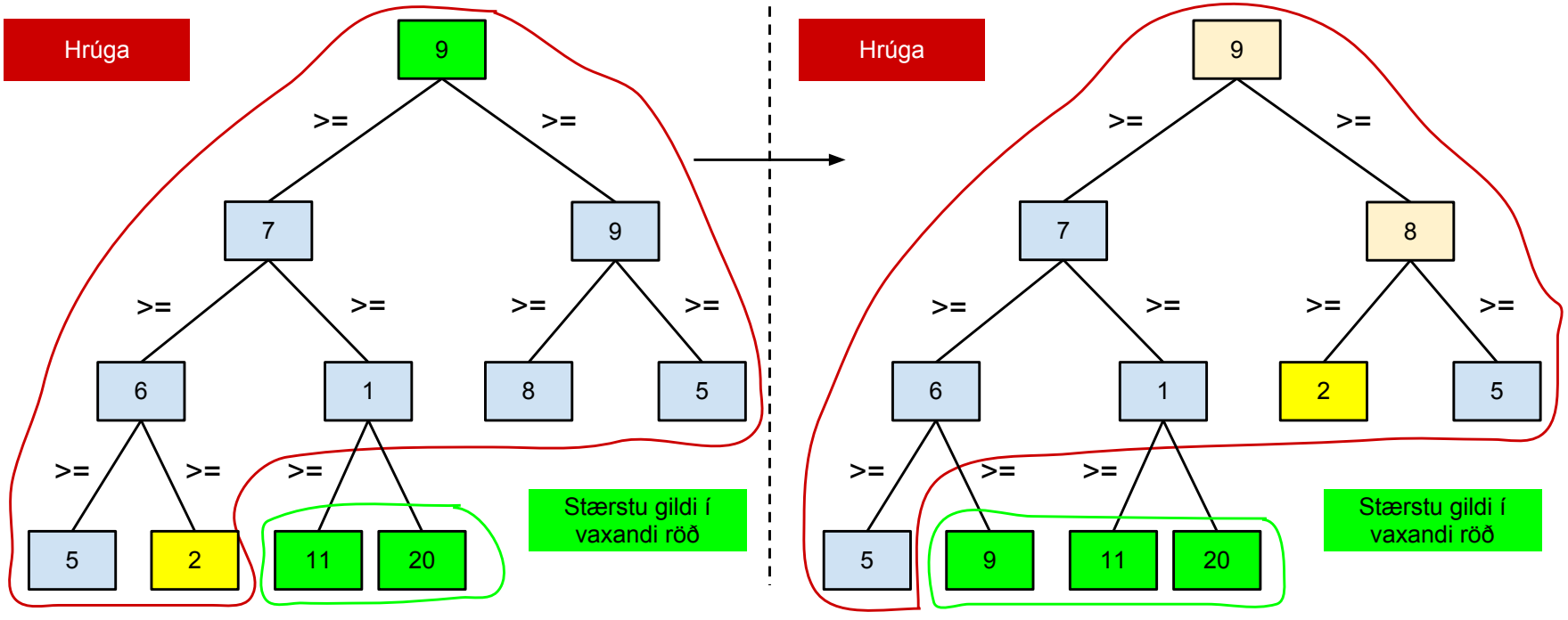
Röðun með hrúgu

Tvö svæði:

- f[0..k-1] er hrúga með öll minnstu gildi f[0..n-1]
- f[k..n-1] er í vaxandi röð með stærstu gildi f[0..n-1]



Röðun með hrúgu



Röðun með hrúgu

```
// f[0..n-1] er hrúga með stærsta stak efst

k = n;

while (k > 0)
{
    // f[0..k-1] er hrúga með stærsta efst og
    // inniheldur minnstu stök f[0..n-1].
    // f[k..n-1] er í vaxandi röð og
    // inniheldur stærstu stök f[0..n-1].
    // 0 <= k <= n
    k--;
    swap(f, 0, k);
    rolldown(f, 0, k, 0);
}
```

rolldown notkunarlýsing

```
// Notkun: rolldown(f,i,j,k);
// Fyrir:  f[i..j-1] er hrúga nema e.t.v.
//         er f[k] of ofarlega
// Eftir:  f[i..j-1] er hrúga
static void rolldown(String[] f, int i, int j, int k)
{
    // ...
}
```

rolldown kóði

```
// Notkun: rolldown(f,i,j,k);
// Fyrir:  f[i..j-1] er hrúga nema e.t.v.
//         er f[k] of ofarlega
// Eftir:  f[i..j-1] er hrúga
static void rolldown(String[] f, int i, int j, int k)
{
    while (true)
    {
        // f[i..j-1] er hrúga nema e.t.v.
        // er f[k] of ofarlega
        // i <= k <= j
        int b = 2*k+1;
        if (b >= j) return;
        if (b+1 < j && f[b+1].compareTo(f[b]) > 0) b++;
        if (f[k].compareTo(f[b]) >= 0) return;
        swap(f, k, b);
        k = b;
    }
}
```

Hrúgugerð

Hvernig búum við til hrúguna?

Byggjum hrúgu (heapify)

```
// Notkun: heapify(f);
// Fyrir:  Ekkert
// Eftir:  f er hrúga með stærsta stak efst
public static void heapify(String[] f)
```

Tvær aðferðir til að útfæra heapify:

1. Byrja fremst, stækka og víxla aftasta gildinu upp (rollup)
2. Byrja aftast, stækka og víxla fremsta gildinu niður (rolldown)

Heapify með rollup

```
// Notkun: heapify(f);
// Fyrir:  Ekkert
// Eftir:  f er hrúga með stærsta stak efst
public static void heapify(String[] f)
{
    int n = f.length;
    int k = 0;

    while (k < n)
    {
        // f[0..k-1] uppfyllir hrúguskilyrði,
        // 0 <= k <= n
        rollup(f, 0, k, k);
        k++;
    }
}
```

rollup hjálparfall

```
// Notkun: rollup(f,i,j,k);
// Fyrir:  f[i..j-1] er hrúga nema e.t.v.
//         er f[k] of neðarlega
// Eftir:  f[i..j-1] er hrúga
static void rollup(String[] f, int i, int j, int k)
{
    while (k > i)
    {
        // f[i..j-1] er hrúga nema e.t.v.
        // er f[k] of neðarlega.
        int parent = (k-1)/2;

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

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

Heapify með rolldown

```
// Notkun: heapify(f);
// Fyrir:  Ekkert
// Eftir:  f er hrúga með stærsta stak efst
public static void heapify(String[] f)
{
    int n = f.length;
    int k = n/2 - 1;

    while (k >= 0)
    {
        // f[k+1..n-1] uppfyllir hrúguskilyrði,
        // 0 <= k <=n
        rolldown(f, k, n, k);
        k--;
    }
}
```

Að lokum: Heapsort

Heapsort

```
static void hsort(String[] f)
{
    heapify(f);

    int n = f.length;
    int k = n;
    while (k > 0)
    {
        // f[0..k-1] uppfyllir hrúguskilyrði og
        // inniheldur minnstu stök f[0..n-1].
        // f[k..n-1] er í vaxandi röð og
        // inniheldur stærstu stök f[0..n-1].
        // 0 <= k <= n
        k--;
        swap(f, 0, k);
        rolldown(f, 0, k, 0);
    }
}
```