Föll og aðferðir

Teljum 1 (eða O(1)) aðgerð fyrir frumstæðar aðgerðir.

Þegar við köllum á föll, aðferðir eða smið teljum við fjölda aðgerða í því falli

Dæmi:

```
int[] a = new int[N]; // kostar O(N) aðgerðir
int max = find_max(a); // skoðar allt fylkið O(N)
Arrays.sort(a); // O(N log(N)) skv. lýsingu
```

Æfingar

1.

```
while (N > 1) {
   N = N / 2;
   ...
}
```

svar: $O(\log_2(N))$

2.

```
for (int i = 0; i < N; i++)
   ...
```

svar: $O(N)$

3.

```
for (int i = 0; i < N; i++)
   for (int j = 0; j < N; j++)
      ...
```

svar: $O(N^2)$

Æfingar

4.

```
public static void g(int N) {
   if (N == 0) return;
   g(N/2);
   g(N/2);
   for (int i = 0; i < N; i++)
      ...
}
```

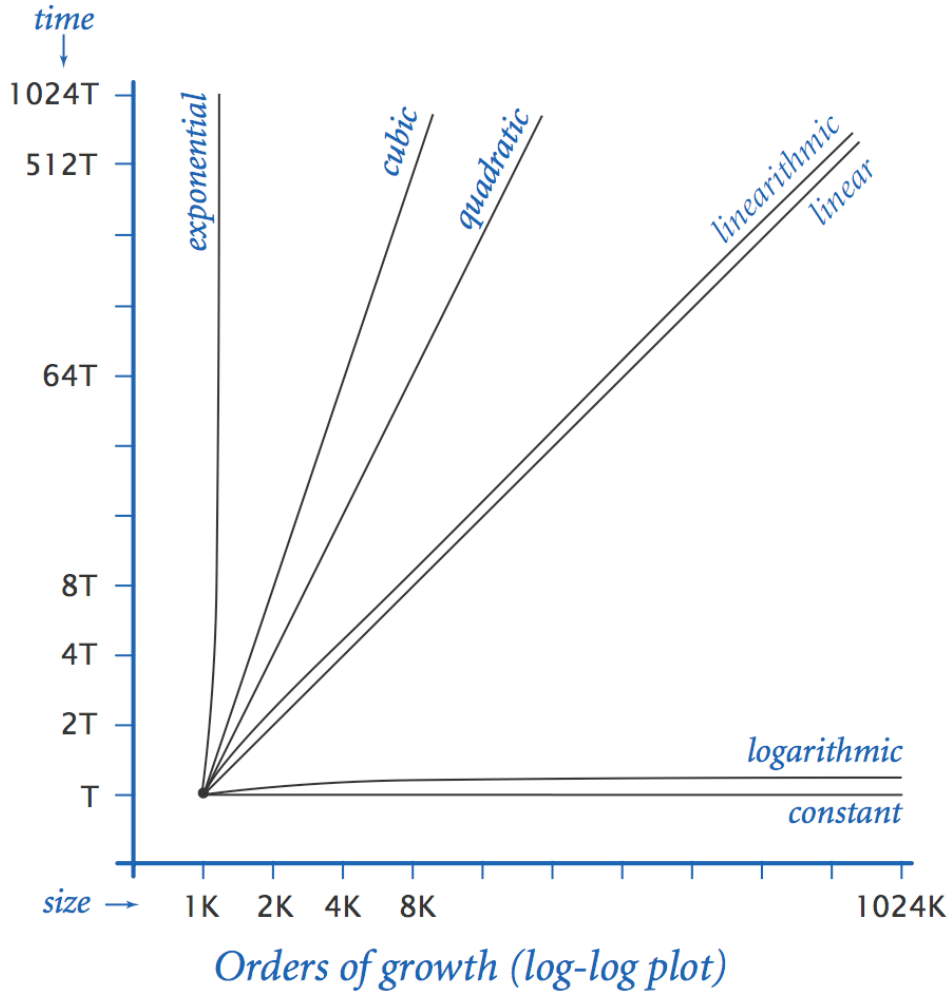
svar: O(Nlog(N))

5.

```
public static void f(int N) {
   if (N == 0) return;
   f(N-1);
   f(N-1);
   ...
}
```

svar: $O(2^N)$

Túlkun á O(…)



Orders of growth (log-log plot)

order of growth *description*	*function*	*factor for doubling hypothesis*
constant	1	1
logarithmic	$\log N$	1
linear	N	2
linearithmic	$N \log N$	2
quadratic	N^2	4
cubic	N^3	8
exponential	2^N	2^N

Commonly encountered growth functions

Gróf röðun: $O(1) << O(\log(N)) << O(N^a) << O(a^N)$

Túlkun á O(…)

order of growth	*predicted running time if problem size is increased by a factor of 100*
linear	a few minutes
linearithmic	a few minutes
quadratic	several hours
cubic	a few weeks
exponential	forever

Effect of increasing problem size for a program that runs for a few seconds

order of growth	*predicted factor of problem size increase if computer speed is increased by a factor of 10*
linear	10
linearithmic	10
quadratic	3-4
cubic	2-3
exponential	1

Effect of increasing computer speed on problem size that can be solved in a fixed amount of time

Helmingunarleit

Köllum keyrslutímann T(N)

```
int helmingunarleit(int[] a, int x, int l, int h) {
    if (l > h) return -1;
    int mid = (l+h)/2;
    if (a[mid] < x) {
        return helmingunarleit(a,x,mid+1,h);
    } else if (a[mid] > x) {
        return helmingunarleit(a,x,l,mid-1);
    } else {
        return mid;
    }
}
```

$T(N) = O(1) + T(N/2), T(0) = O(1)$

svar: $T(N) = O(\log_2(N))$

Helmingunarleit

Hve oft getum við deilt N með 2 þangað til við fáum 1?

$$1$$
$$2 \rightarrow 1$$
$$4 \rightarrow 2 \rightarrow 1$$
$$8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$
$$1024 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

svar: $\log_2(N)$ – fjöldi bita sem þarf til að tákna N

Innsetningar röðun:

- Einföld röðunaraðferð
- Förum frá vinstri til hægri í gegnum fylkið
- Skiptum á staki til vinstri ef það er stærra

i	j	a							
		0	1	2	3	4	5	6	7
6	6	and	had	him	his	was	you	the	but
6	5	and	had	him	his	was	the	you	but
6	4	and	had	him	his	the	was	you	but
		and	had	him	his	the	was	you	but

Inserting `a[6]` *into position by exchanging with larger entries to its left*

Fastayrðing lykkju?

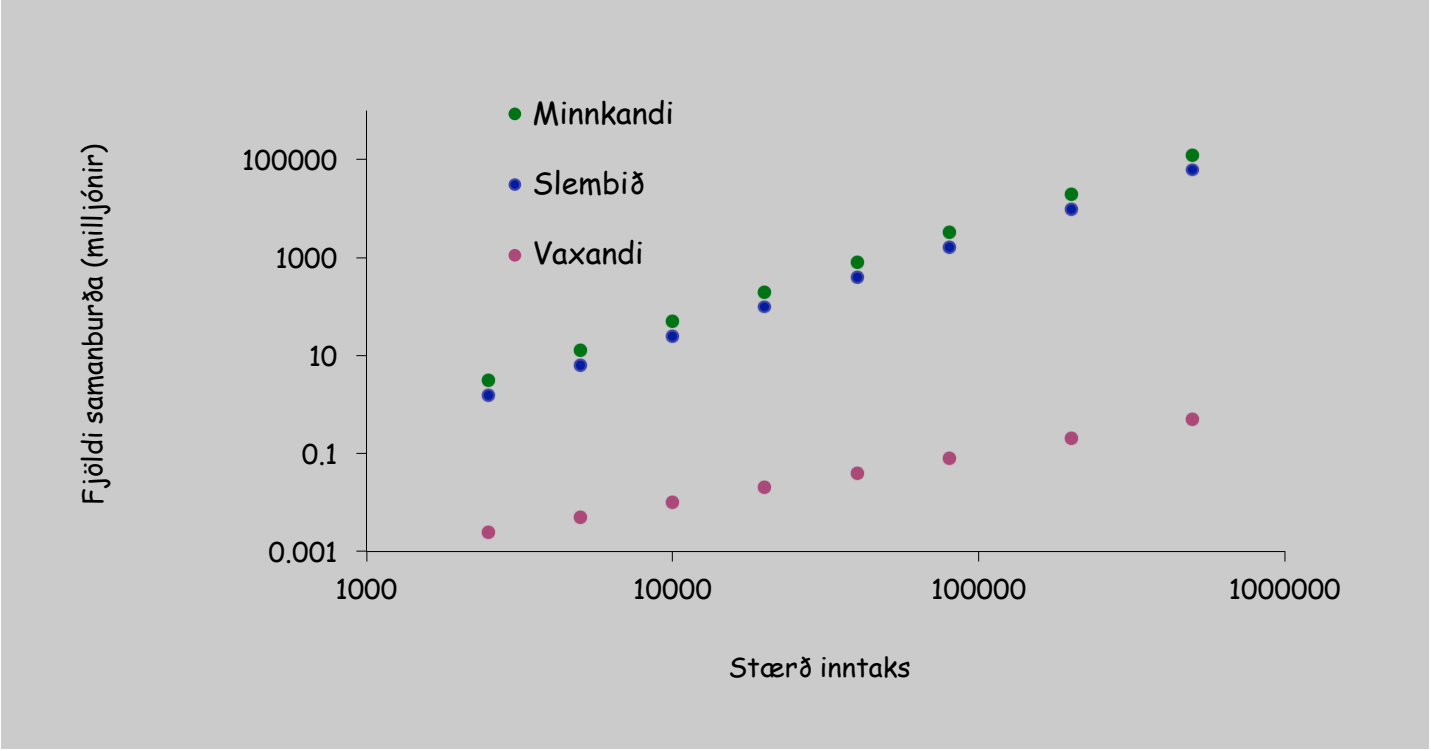
Java útfærsla

```
public class Insertion {

   public static void sort(String[] a) {
      int N = a.length;
      for (int i = 1; i < N; i++)
         for (int j = i; j > 0; j--)
            if (a[j-1].compareTo(a[j]) > 0)
               swap(a, j-1, j);
            else break;
   }

   private static void swap(String[] a, int i, int j) {
      String t= a[i];
      a[i] = a[j];
      a[j] = t;
   }
}
```

Insertion sort: tímamælingar

Tíminn er háður inntakinu

- Minnkandi fylki: ~$N^2/2$
- Slembið fylki: ~$N^2/4$
- Vaxandi fylki: ~N



Mergesort byggir á endurkvæmri röðun

- Skiptum fylki í tvo helminga
- Röðum sitt hvorum helmingnum endurkvæmt
- Skeytum saman röðuðum helmingunum í rétta röð

input
was had him and you his the but

sort left
and had him was you his the but

sort right
and had him was but his the you

merge
and but had him his the was you

Mergesort

Rétt eins og í quicksort er erfiði hlutinn í hjálparfalli

```
public class Merge {

   public static void sort(String[] a) {
      sort(a, 0, a.length);
   }

   // Sort a[lo, hi).
   public static void sort(String[] a, int lo, int hi) {
      int N = hi - lo;
      if (N <= 1) return;

      // recursively sort left and right halves
      int mid = lo + N/2;
      sort(a, lo, mid);
      sort(a, mid, hi);

      // merge sorted halves
   }

}
```

Mergesort

Raða saman a[0..3] og a[4..7], útkoman er geymd í hjálparfylkinu aux

i	j	k	aux[k]	a							
				0	1	2	3	4	5	6	7
				and	had	him	was	but	his	the	you
0	4	0	and	and	had	him	was	but	his	the	you
1	4	1	but	and	had	him	was	but	his	the	you
1	5	2	had	and	had	him	was	but	his	the	you
2	5	3	him	and	had	him	was	but	his	the	you
3	5	4	his	and	had	him	was	but	his	the	you
3	6	5	the	and	had	him	was	but	his	the	you
3	6	6	was	and	had	him	was	but	his	the	you
4	7	7	you	and	had	him	was	but	his	the	you

Trace of the merge of the sorted left half with the sorted right half

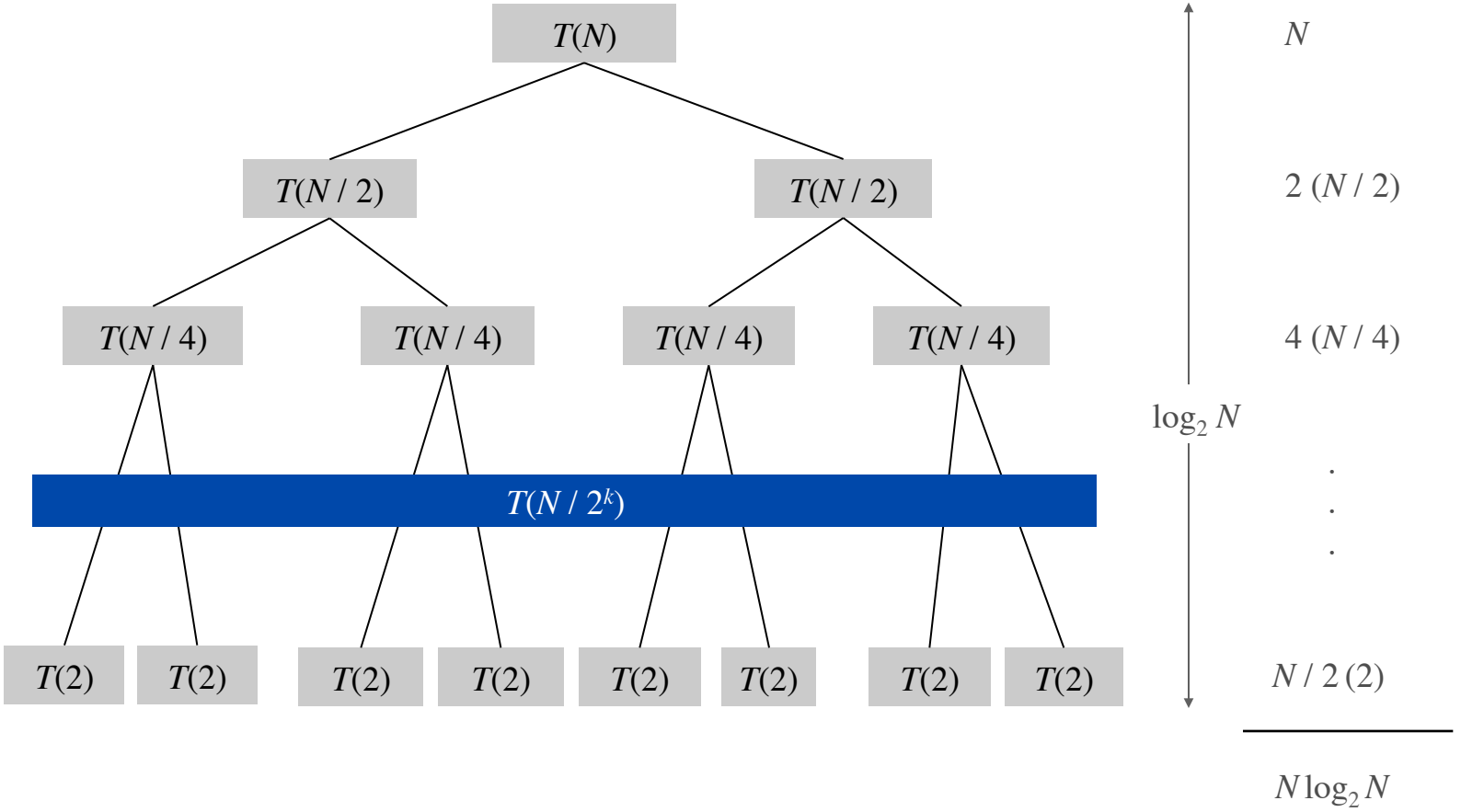
Mergesort

Mergesort þarf auka hjálparfylki aux, ólíkt quicksort.

```
String[] aux = new String[N];
// merge into auxiliary array
int i = lo, j = mid;
for (int k = 0; k < N; k++) {
   if       (i == mid) aux[k] = a[j++];
   else if  (j == hi)  aux[k] = a[i++];
   else if (a[j].compareTo(a[i]) < 0) aux[k] = a[j++];
   else                               aux[k] = a[i++];
}

// copy back
for (int k = 0; k < N; k++) {
   a[lo + k] = aux[k];
}
```

Tímagreining á Mergesort

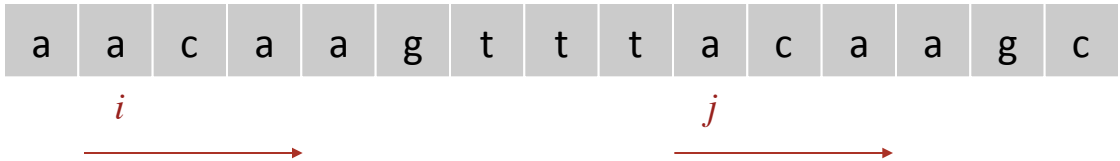
Keyrslutíminn T(N) fyrir Mergesort uppfyllir, T(2) = O(1) og T(N) = T(N/2) + T(N/2) + O(N)



Röðun og leit

Finnum lengsta endurtekna hlutstreng, þ.e. kemur fyrir a.m.k. tvisvar

Einföld lausn, prófum alla mögulega staði sem strengirnir byrja, finnum lengsta sameiginlega forskeyti, $O(N^2)$ keyrslutími



Hagnýtingar: gagnaþjöppun, lífupplýsingafræði