Quick Sort

I. SORTERING

II. QUICK SORT

Kap. 8 (kun 8.1.1, 8.1.2, 8.3; kursorisk 8.4, 8.5; unntatt 8.1.3, 8.2, 8.6)

Sortering

- Bubble $O(n^2)$

prioritetskø basert

- Insertion $O(n^2)$
- Selection $O(n^2)$
- Heap $O(n \log n)$

"splitt og hersk"

- Merge $O(n \log n)$
- Quick $O(n \log n)$

- Bucket
- Radix

‘‘Splitt og Hersk’’

Prob(*n*) :

Splitt : hvis *n* er basistilfelle – gjør det du skal
ellers del *n* i *mindre subproblemer* *n1* ... *nk*

Rekursjon : løs rekursivt **Prob**(*n1*) ... **Prob**(*nk*)

Hersk : **kombiner** resultater til en løsning for hele *n*

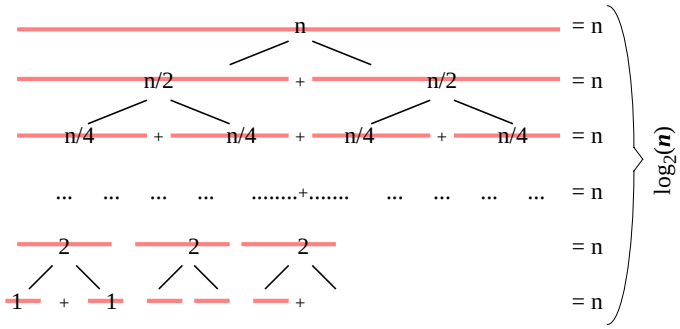
Merge-Sort

MS(*S*) :

Splitt : hvis *S* har minst 2 elementer
del *S* i midten i to *like lange*
subsekvenser *S1* og *S2* *O*(**1** ev. ***n***)

Rekursjon : sorter rekursivt **MS**(*S1*) og **MS**(*S2*)

Hersk : **flett** de sorterte resultatene til en sortert hele *O*(***n***)



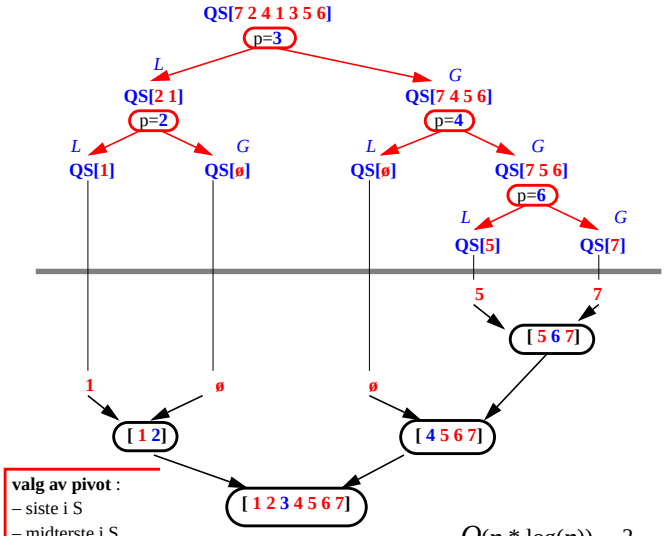
MS(***n***) = *O* (***n*** * log(***n***))

Quick Sort (S)

Splitt : hvis S har minst 2 elementer
velg et element x fra S (pivot)
del S i tre mengder L, E og G slik at : $O(n)$
L inneholder alle elementene (fra S) mindre enn x
E inneholder alle elementene (fra S) lik x
G inneholder alle elementene (fra S) større enn x

Rekursjon : sorter rekursivt **QS**(L) og **QS**(G)

Hersk : returner sekvensen L–E–G $O(n)$

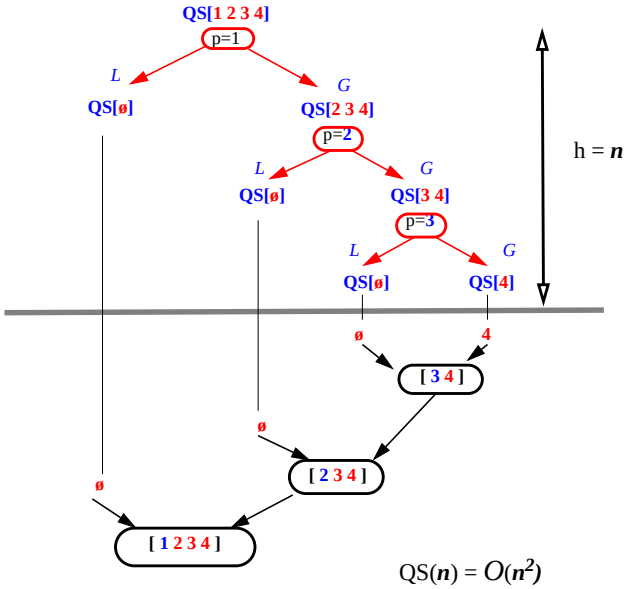


valg av pivot :
– siste i S
– midterste i S
– midterste bland {første, midterste, siste} i S
– random

$O(n * \log(n))$ ... ?

I Verste Fall ...

pivot(S) = første elementet i S (el. siste)
... og S er initielt sortert !!!



QS(n) = $O(n^2)$

Problem : *velg pivot slik at den vil dele sekvensen i 2 omtrent like lange deler !*

Randomisering

```
import java.util.Random;
Random rg = new Random();
```

Splitt : hvis S har minst 2 elementer
velg et element x fra S (pivot)
x = **S[rg.newInt() % S.length()]**
del S i tre mengder L, E og G slik at :
L inneholder alle elementene (fra S) mindre enn x
E inneholder alle elementene (fra S) lik x
G inneholder alle elementene (fra S) større enn x

Rekursjon : sorter rekursivt **QS**(L) og **QS**(G)

Hersk : returner sekvensen L–E–G

8.3. ***Forventet*** kjøretid av randomisert QS på en sekvens av lengde ***n*** er ***O***(***n*** * log ***n***)

QuickSort (med passende valg av pivot) er, *i praksis*, den beste kjente sorteringsalgoritme

In-Place

En (sortering) algoritme er "**in-place**" dersom
den kun trenger **konstat lagringsplass** i tillegg til selve argumentet.

```
Sequence MS(Sequence S)
   int l = S.length();
   if (l==1) return S ;
   else int m= l/2;
     S1= new Sequence(S[0...m]) ;
     S2= new Sequence(S[m=1...l]) ;
     R1= MS(S1); R2= MS(S2);
     return flett(R1,R2) ;
```

```
Sequence QS(Sequence S)
   int l = S.length();
   if (l==1) return S ;
   else  p= pivot(S);
     L= new Sequence(... < p) ;
     G= new Sequence(... > p);
     E= new Sequence(... = p);
     L1= QS(L); G1= QS(G);
     return L1–E–G1
```