

# Algoritmer og Datastrukturer 1

Gerth Stølting Brodal

**Quicksort**  
**[CLRS, kapitel 7]**



# Quicksort:

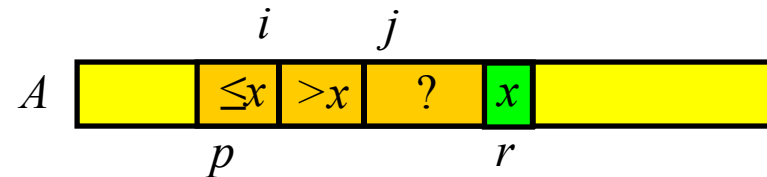
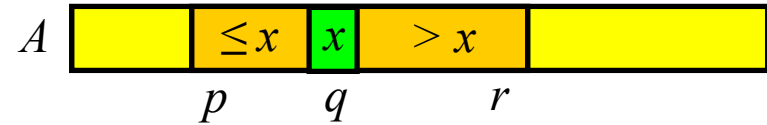
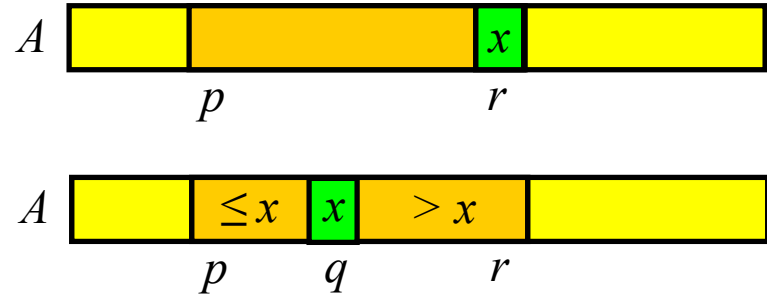
Sorter  $A[p..r]$

QUICKSORT( $A, p, r$ )

```
1  if  $p < r$ 
2    then  $q \leftarrow$  PARTITION( $A, p, r$ )
3         QUICKSORT( $A, p, q - 1$ )
4         QUICKSORT( $A, q + 1, r$ )
```

PARTITION( $A, p, r$ )

```
1   $x \leftarrow A[r]$ 
2   $i \leftarrow p - 1$ 
3  for  $j \leftarrow p$  to  $r - 1$ 
4    do if  $A[j] \leq x$ 
5       then  $i \leftarrow i + 1$ 
6           exchange  $A[i] \leftrightarrow A[j]$ 
7  exchange  $A[i + 1] \leftrightarrow A[r]$ 
8  return  $i + 1$ 
```

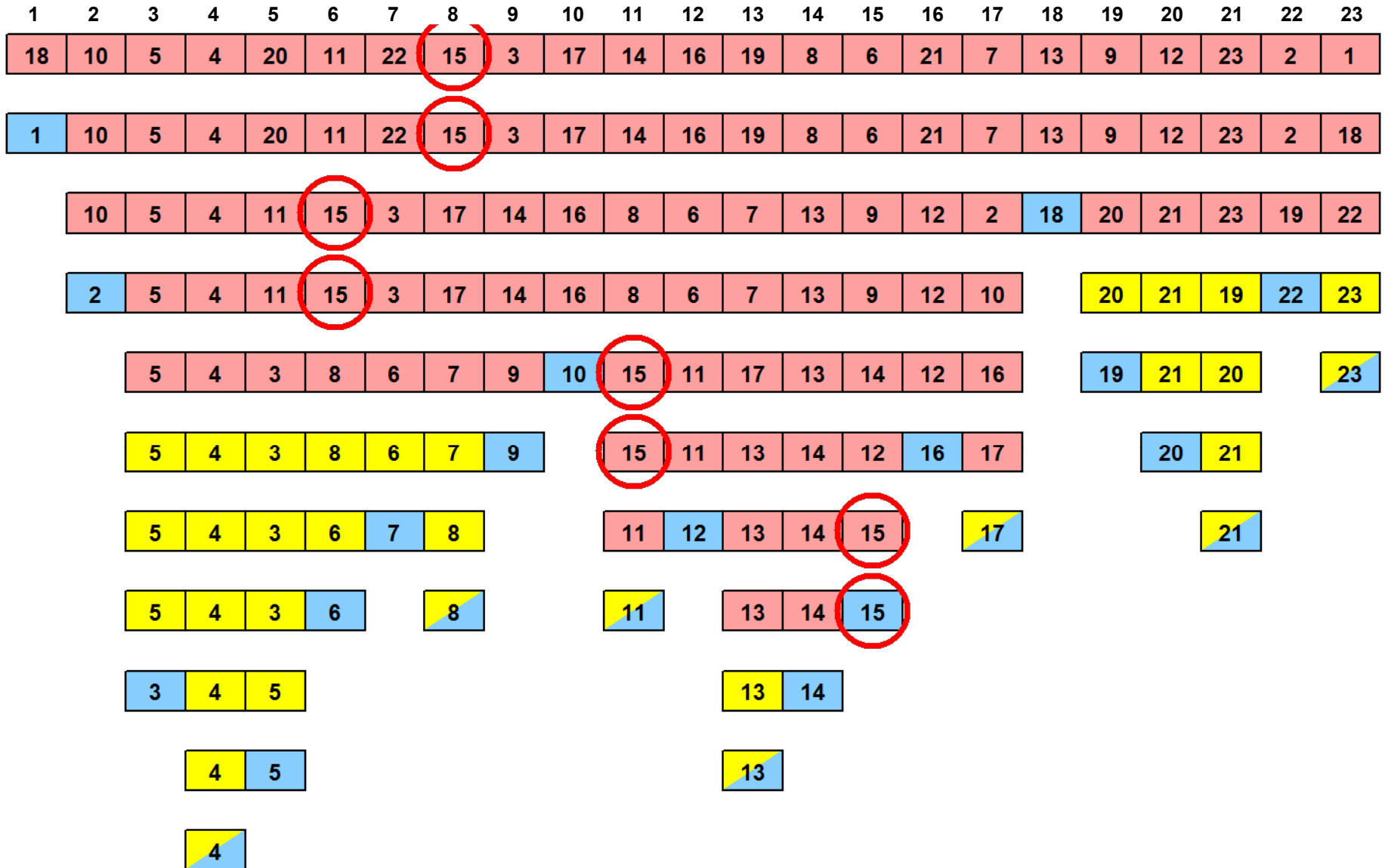


Worst-case time  $O(n^2)$

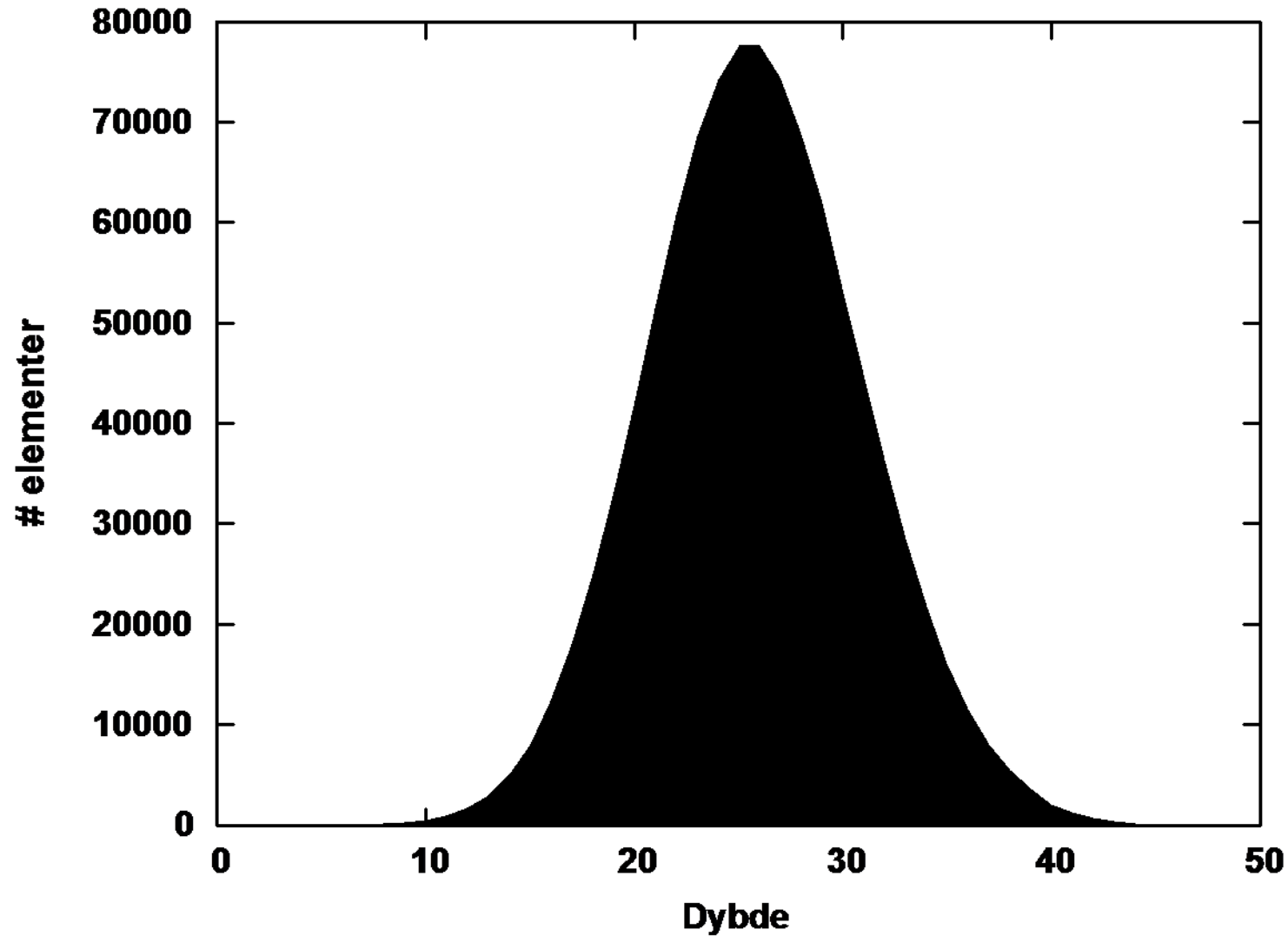
Hoare, 1961



# Quicksort : Rekursionen for 15



# Quicksort : Dybde ved $n \approx 2^{20}$



# Randomized Quicksort

RANDOMIZED-QUICKSORT( $A, p, r$ )

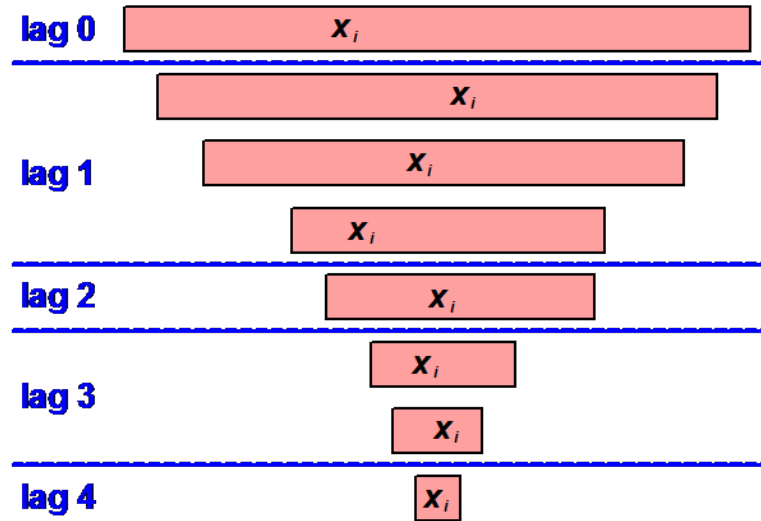
```
1  if  $p < r$ 
2      then  $q \leftarrow$  RANDOMIZED-PARTITION( $A, p, r$ )
3          RANDOMIZED-QUICKSORT( $A, p, q - 1$ )
4          RANDOMIZED-QUICKSORT( $A, q + 1, r$ )
```

RANDOMIZED-PARTITION( $A, p, r$ )

```
1   $i \leftarrow$  RANDOM( $p, r$ )
2  exchange  $A[r] \leftrightarrow A[i]$ 
3  return PARTITION( $A, p, r$ )
```

**Forventet tid  $O(n \cdot \log n)$**

# Randomized Quicksort : Analyse



- Et array er  $i$  lag  $j$  hvis længde  $n(3/4)^{j+1}.. n(3/4)^j$
- En opdeling er **god** hvis hver del  $\leq 3/4$  elementer (mindst +1 lag) – sker med **sandsynlighed  $\geq 0.5$**
- $x_i$  forventes  $\leq 2$  gange i hvert lag
- **Forventede dybde af  $x_i \leq 2 \cdot \log_{4/3} n$**

# Randomized Quicksort : Analyse

**Forventede** tid for randomized quicksort

$$= O(\sum_{i=1..n} \text{forventede dybde af input } x_i)$$

$$= O(\sum_{i=1..n} \log n)$$

$$= O(n \cdot \log n)$$



# Sorterings-algoritmer

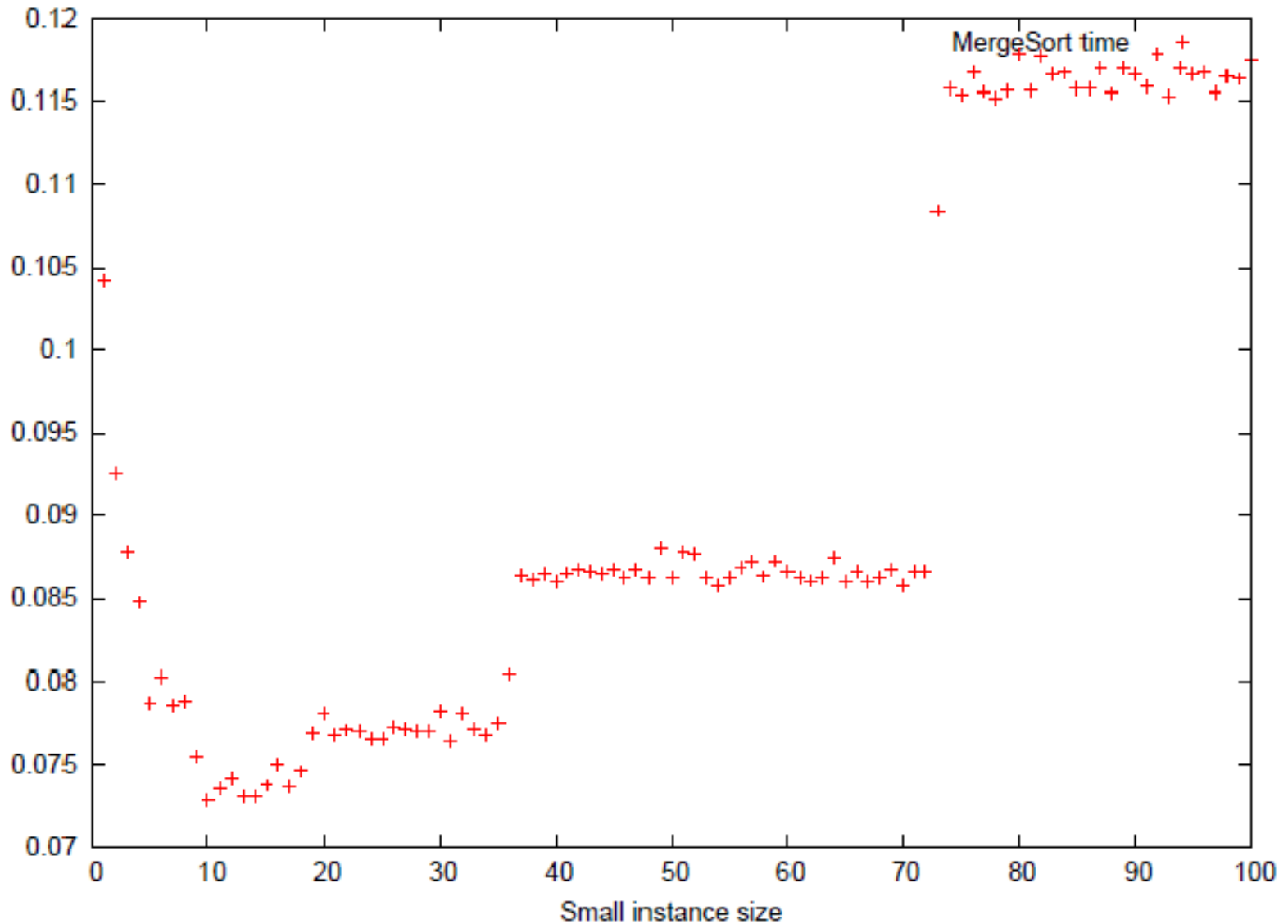
Algoritme	Worst-Case Tid
Heap-Sort	$O(n \cdot \log n)$
Merge-Sort	
Insertion-Sort	$O(n^2)$
QuickSort (Deterministic og randomiseret)	$O(n^2)$

Algoritme	Forventet tid
Randomiseret QuickSort	$O(n \cdot \log n)$

**Sortering:**

**Eksperimentelle resultater**

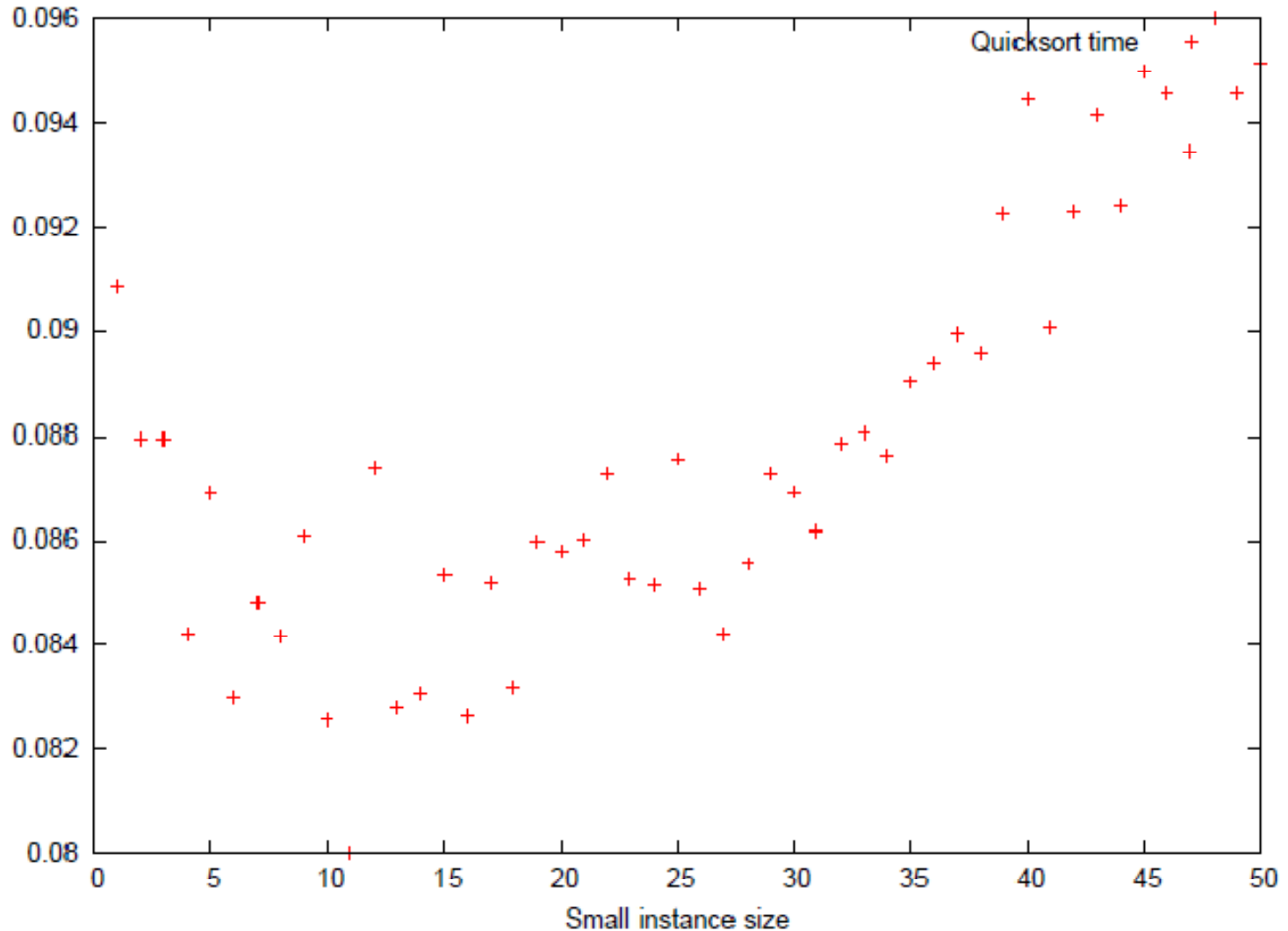
# Mergesort med skift til Insertion-sort



Skift til insertion-sort ved små problemstørrelser

$n = 300.000$  elementer

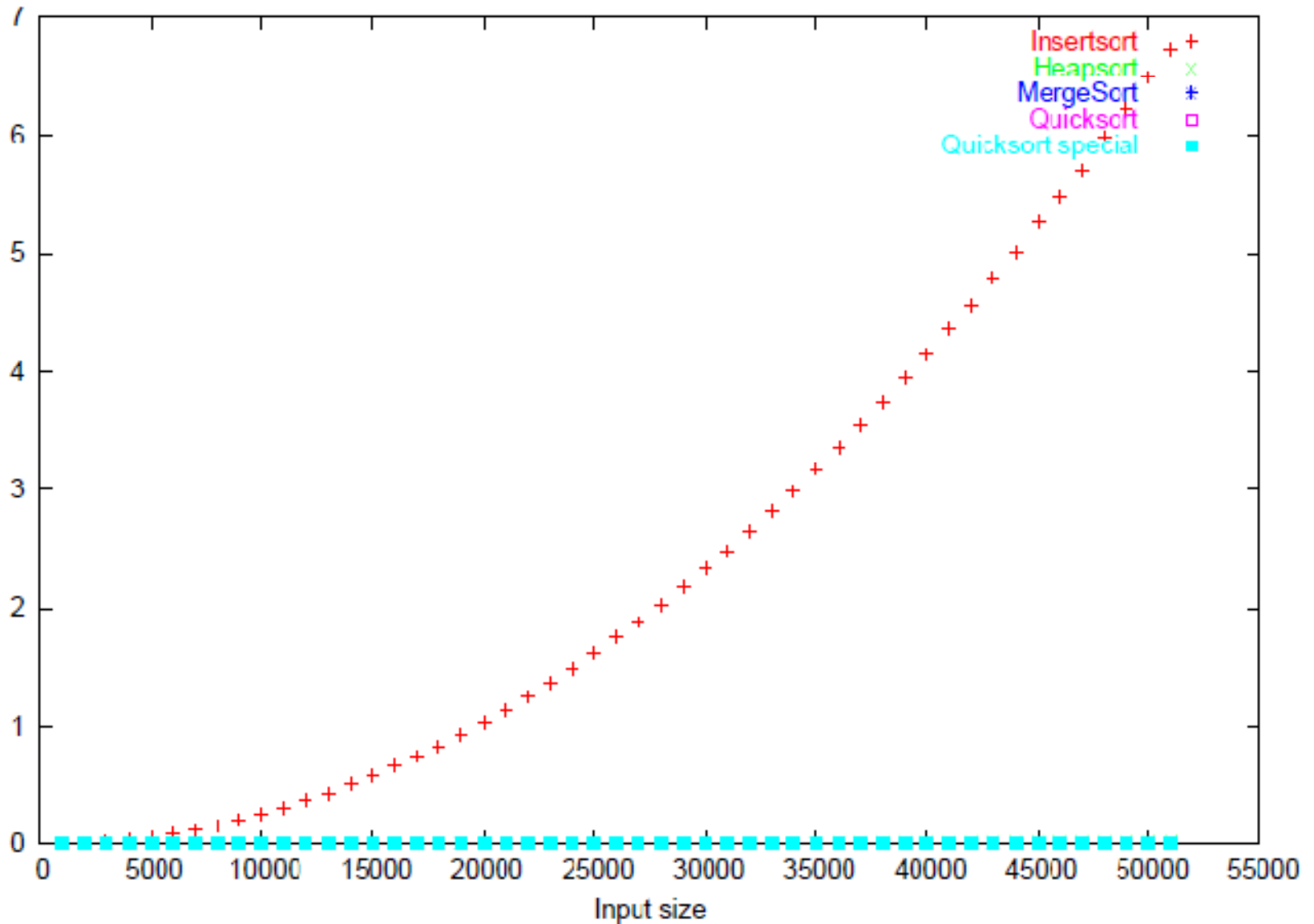
# Quicksort med skift til Insertion-sort



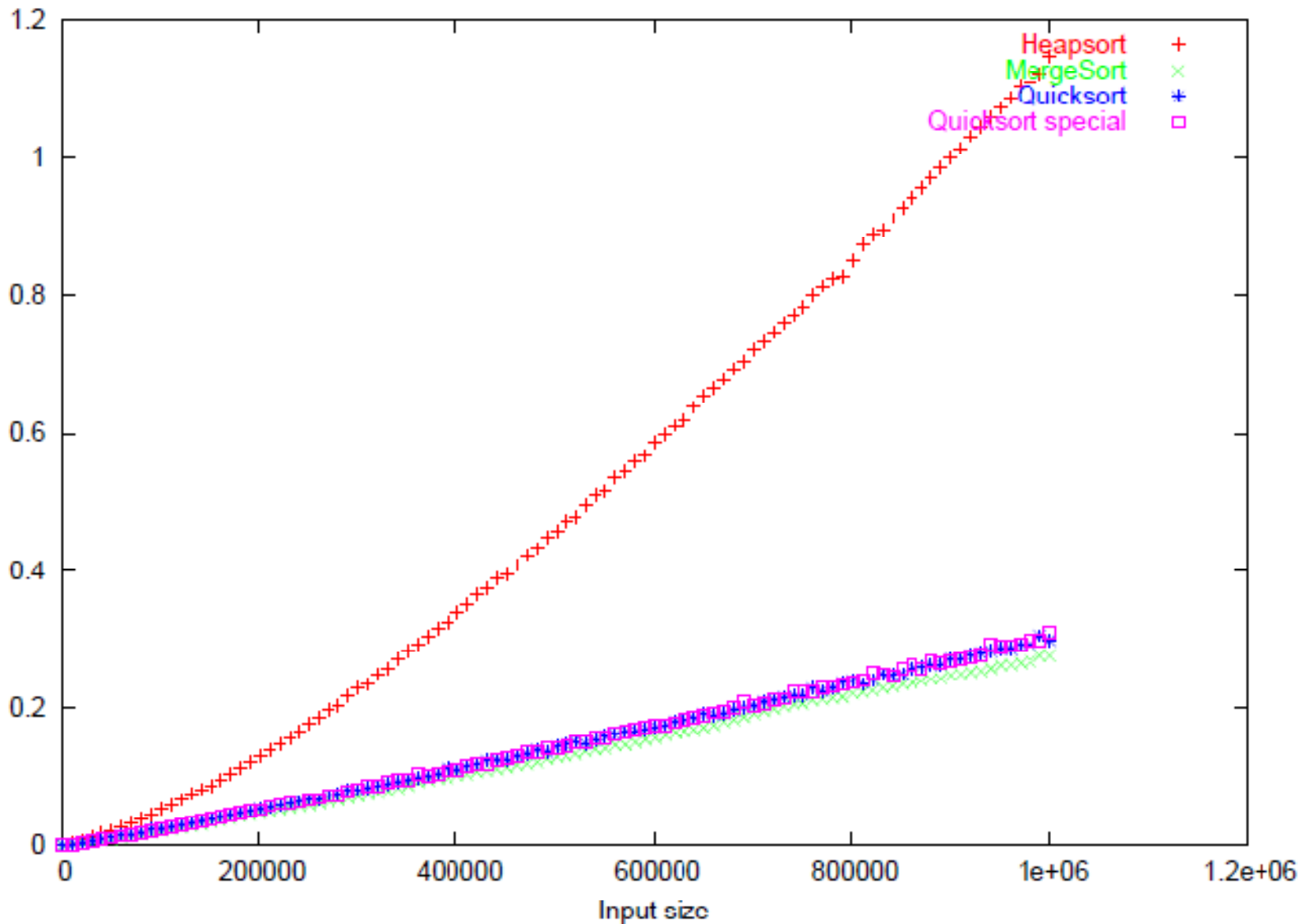
Skift til insertion-sort ved små problemstørrelser

$n = 300.000$  elementer

# Tiden for Sorterings Algoritmer



# Tiden for Sorterings Algoritmer



# Algoritmer og Datastrukturer 1

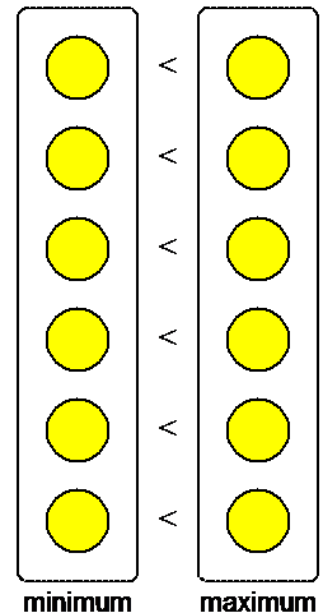
Gerth Stølting Brodal

**Randomized-select**  
**[CLRS, kapitel 9.1-9.2]**



# Beregning af Minimum of Maximum

- At finde *minimum* af  $n$  elementer kræver  $n-1$  sammenligninger
- At finde *minimum og maximum* af  $n$  elementer kræver  $3/2 \cdot n - 2$  sammenligninger



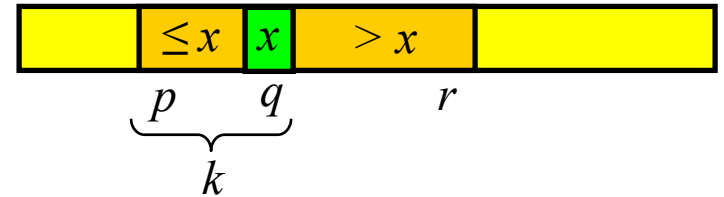


# Randomized Select:

Find det  $i$ 'te mindste element i  $A[p..r]$

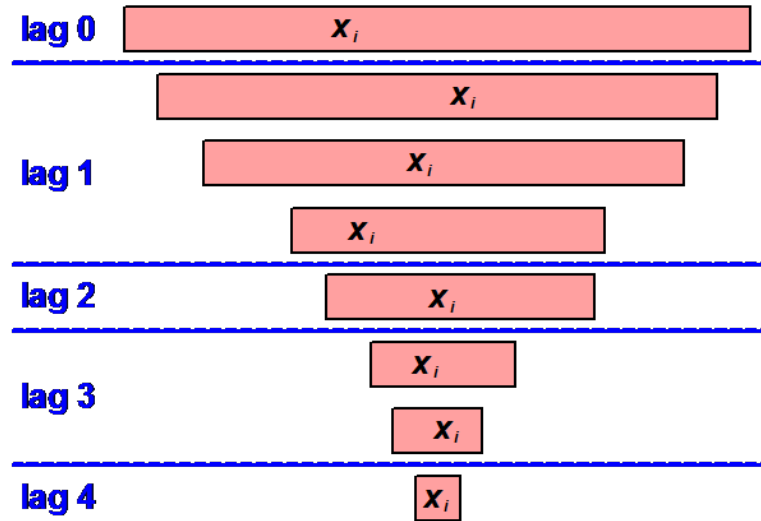
RANDOMIZED-SELECT( $A, p, r, i$ )

```
1  if  $p = r$ 
2    then return  $A[p]$ 
3   $q \leftarrow$  RANDOMIZED-PARTITION( $A, p, r$ )
4   $k \leftarrow q - p + 1$ 
5  if  $i = k$        $\triangleright$  the pivot value is the answer
6    then return  $A[q]$ 
7  elseif  $i < k$ 
8    then return RANDOMIZED-SELECT( $A, p, q - 1, i$ )
9  else return RANDOMIZED-SELECT( $A, q + 1, r, i - k$ )
```



**Forventet tid  $O(n)$**

# Randomized Select : Analyse



**Forventede tid** for randomized select

$$= O(\sum_j \text{forventede tid i lag } j)$$

$$\leq O(\sum_j n \cdot (3/4)^j \cdot \text{\# forventede arrays i lag } j)$$

$$\leq O(\sum_j n \cdot (3/4)^j \cdot 2)$$

$$= O(n)$$