Algoritmer og Datastrukturer 1

Gerth Stølting Brodal

Quicksort
[CLRS, kapitel 7]

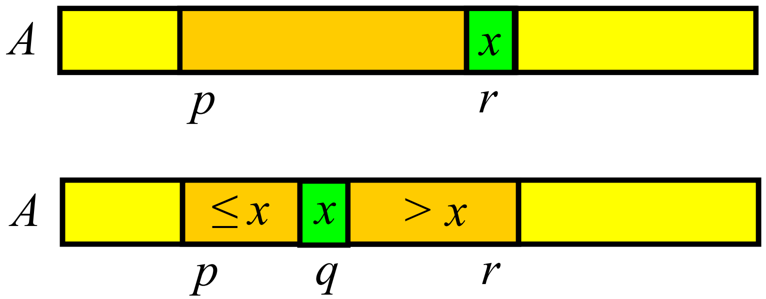
AARHUS UNIVERSITET

Sandsynligheden for at slå krone
1/2

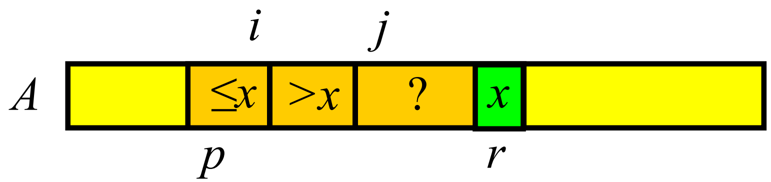
Quicksort:

Sorter $A[p..r]$

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q - 1)
    QUICKSORT(A, q + 1, r)
```



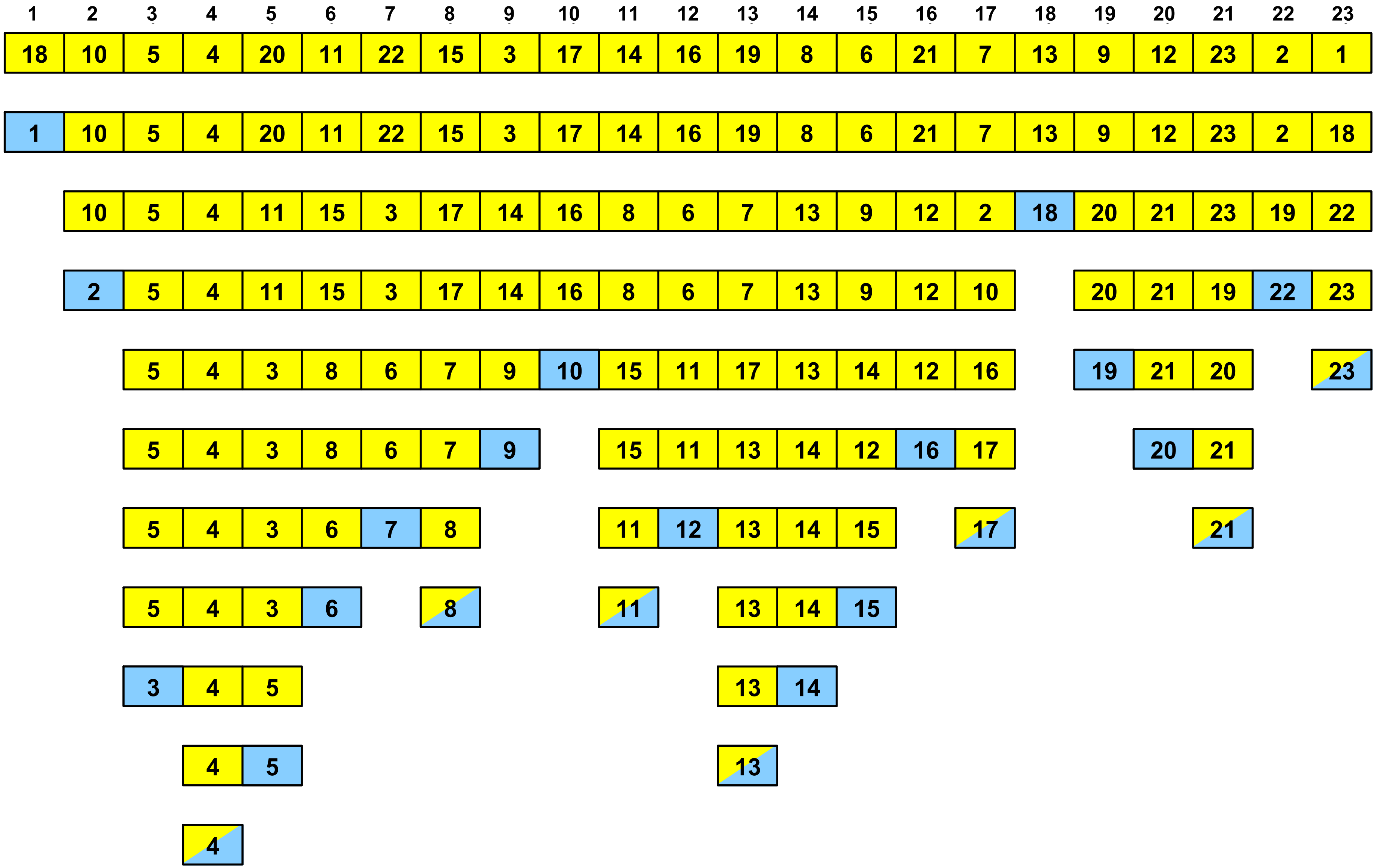

```
PARTITION(A, p, r)
x = A[r]
i = p - 1
for j = p to r - 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



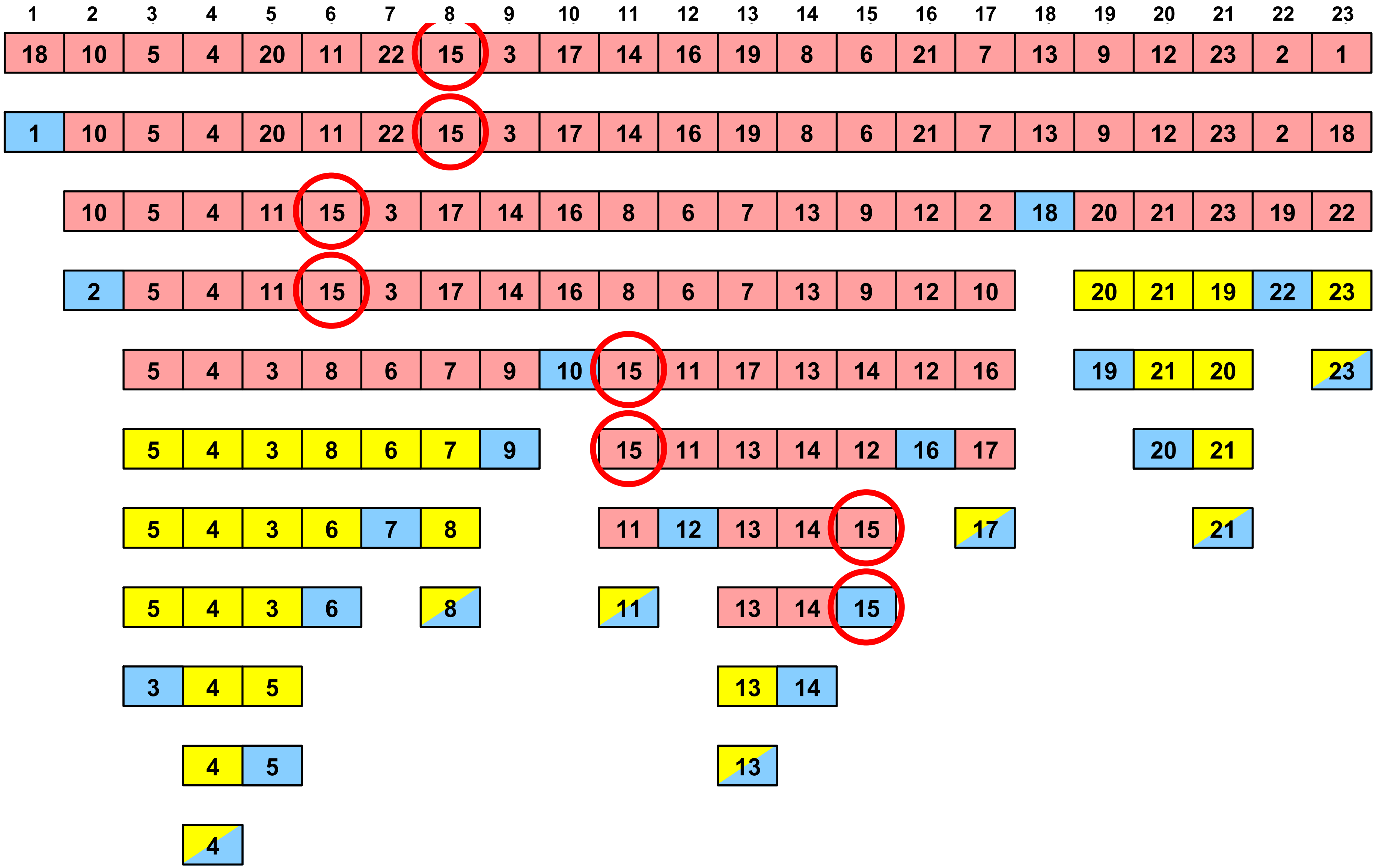
Worst-case tid $O(n^2)$

Hoare, 1961

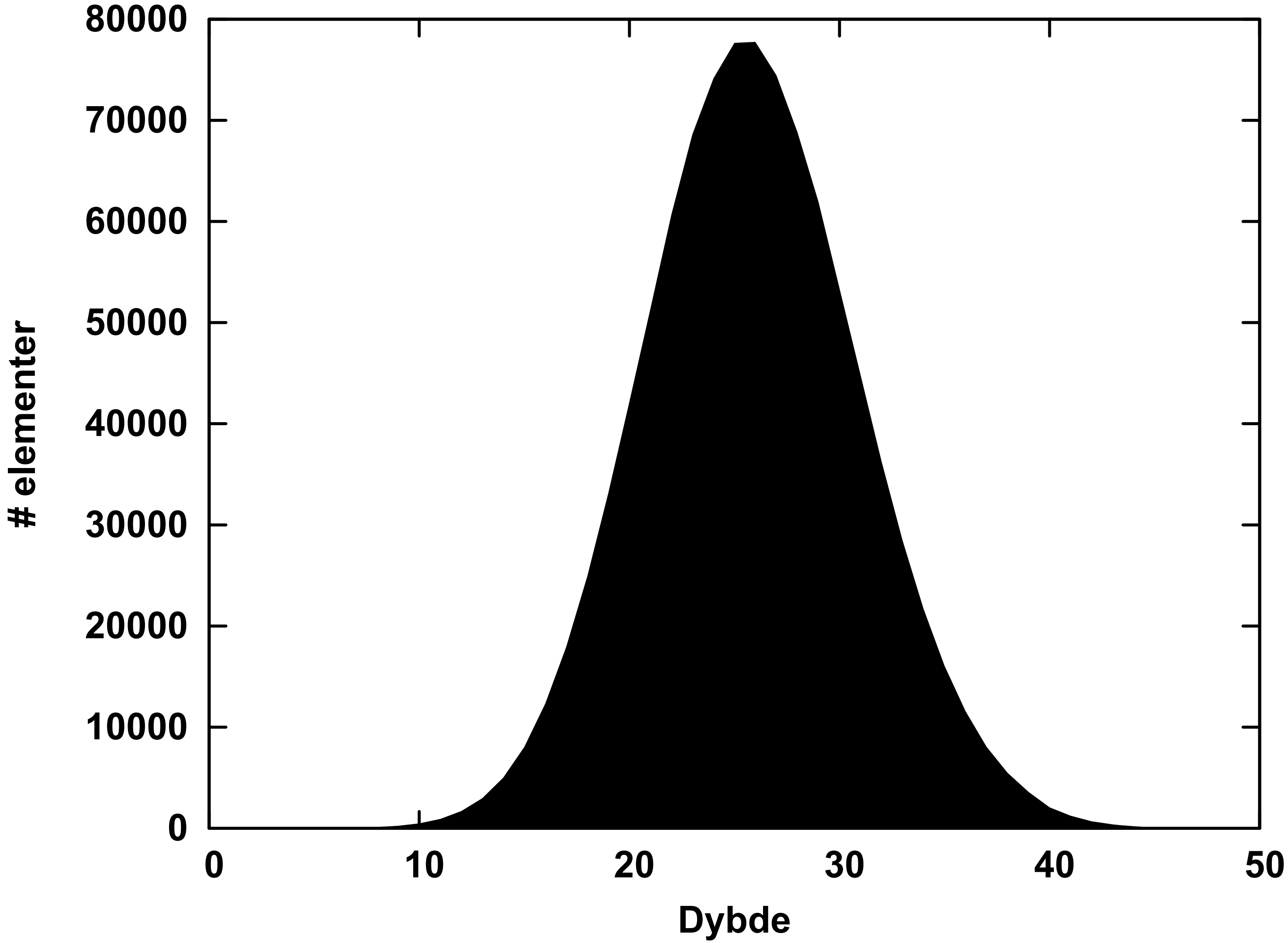
Quicksort på 23 elementer



Quicksort : Rekursionen for 15



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



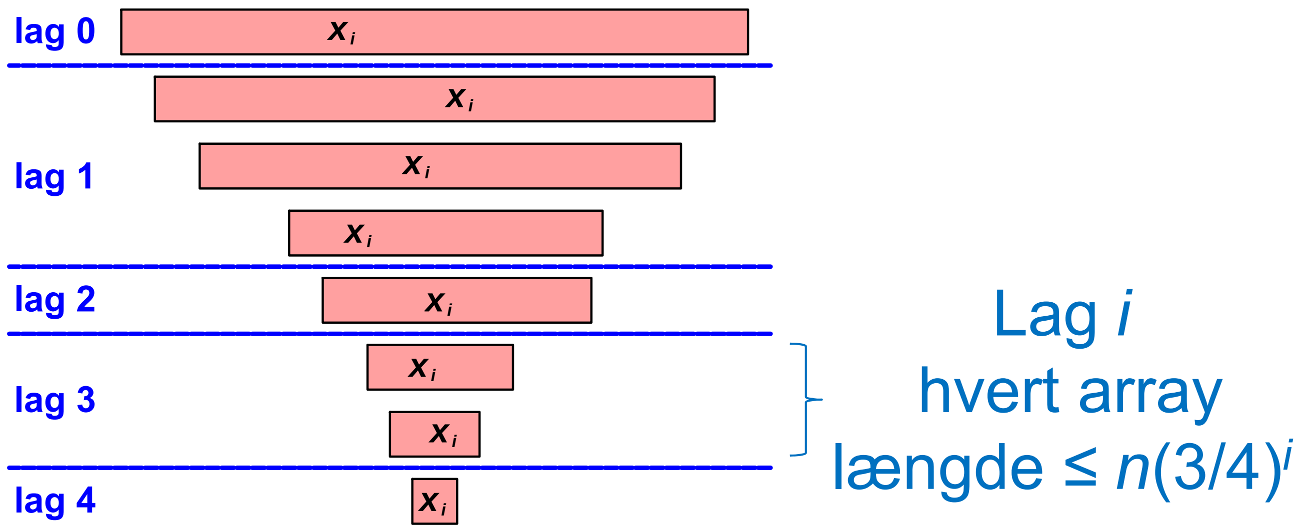
Randomized Quicksort

```
RANDOMIZED-QUICKSORT(A, p, r)
if p < r
    q = RANDOMIZED-PARTITION(A, p, r)
    RANDOMIZED-QUICKSORT(A, p, q - 1)
    RANDOMIZED-QUICKSORT(A, q + 1, r)
```

```
RANDOMIZED-PARTITION(A, p, r)
i = RANDOM(p, r)
exchange A[r] with A[i]
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 $\leq$ ¾ elementer (mindst +1 lag) – sker med sandsynlighed ≥ 0.5
- x_i forventes ≤ 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

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

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

$= O(n \cdot \log n)$ □

Sorterings-algoritmer

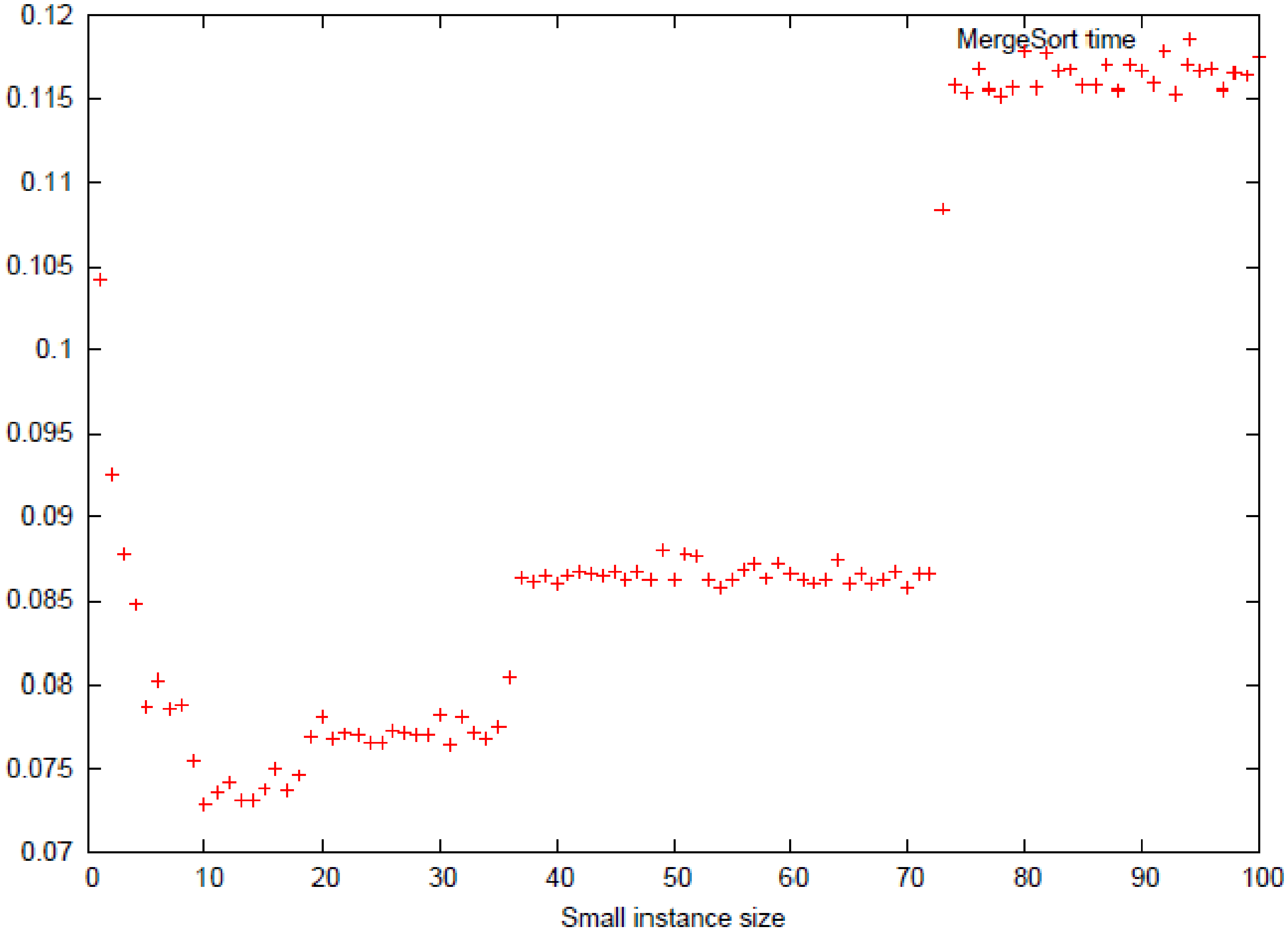
Algoritme	Worst-Case Tid
Heap-Sort	$O(n \cdot \log n)$
Merge-Sort	
Insertion-Sort	$O(n^2)$
QuickSort (Deterministic og randomiseret)	$O(n^2)$

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

Sortering:

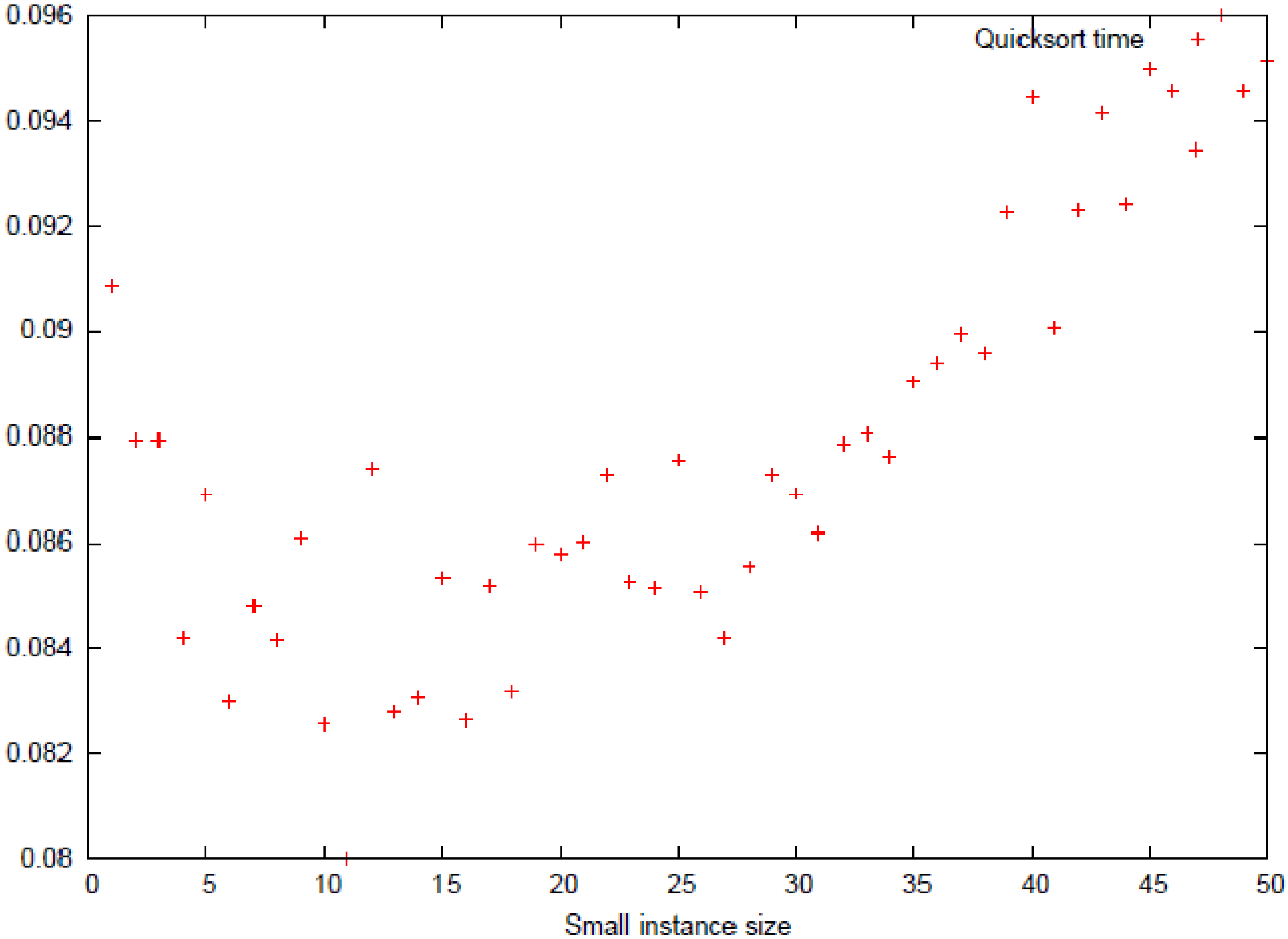
Eksperimentelle resultater

Mergesort med skift til Insertion-sort



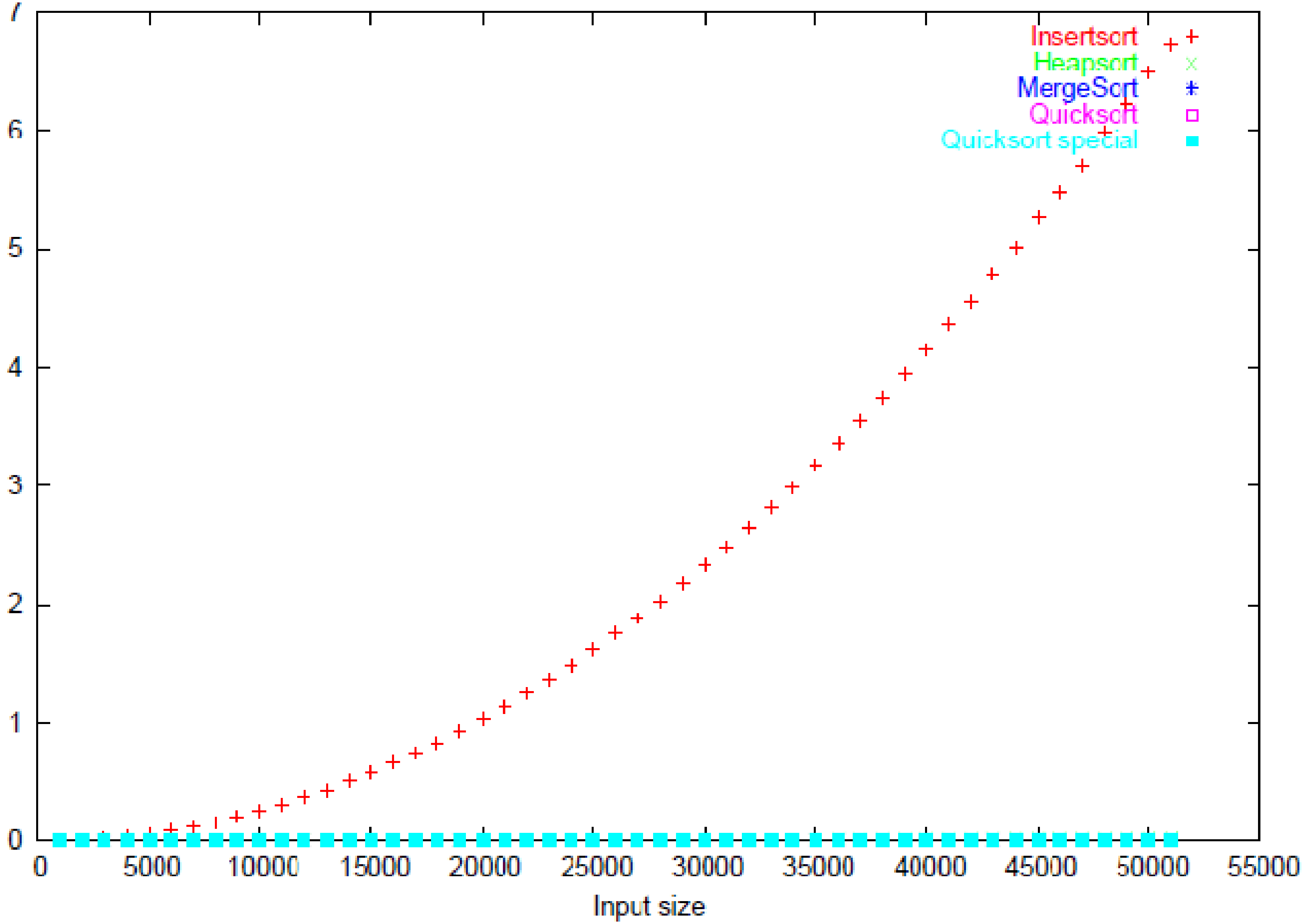
Skift til insertion-sort ved små problemstørrelser n = 300.000 elementer

Quicksort med skift til Insertion-sort

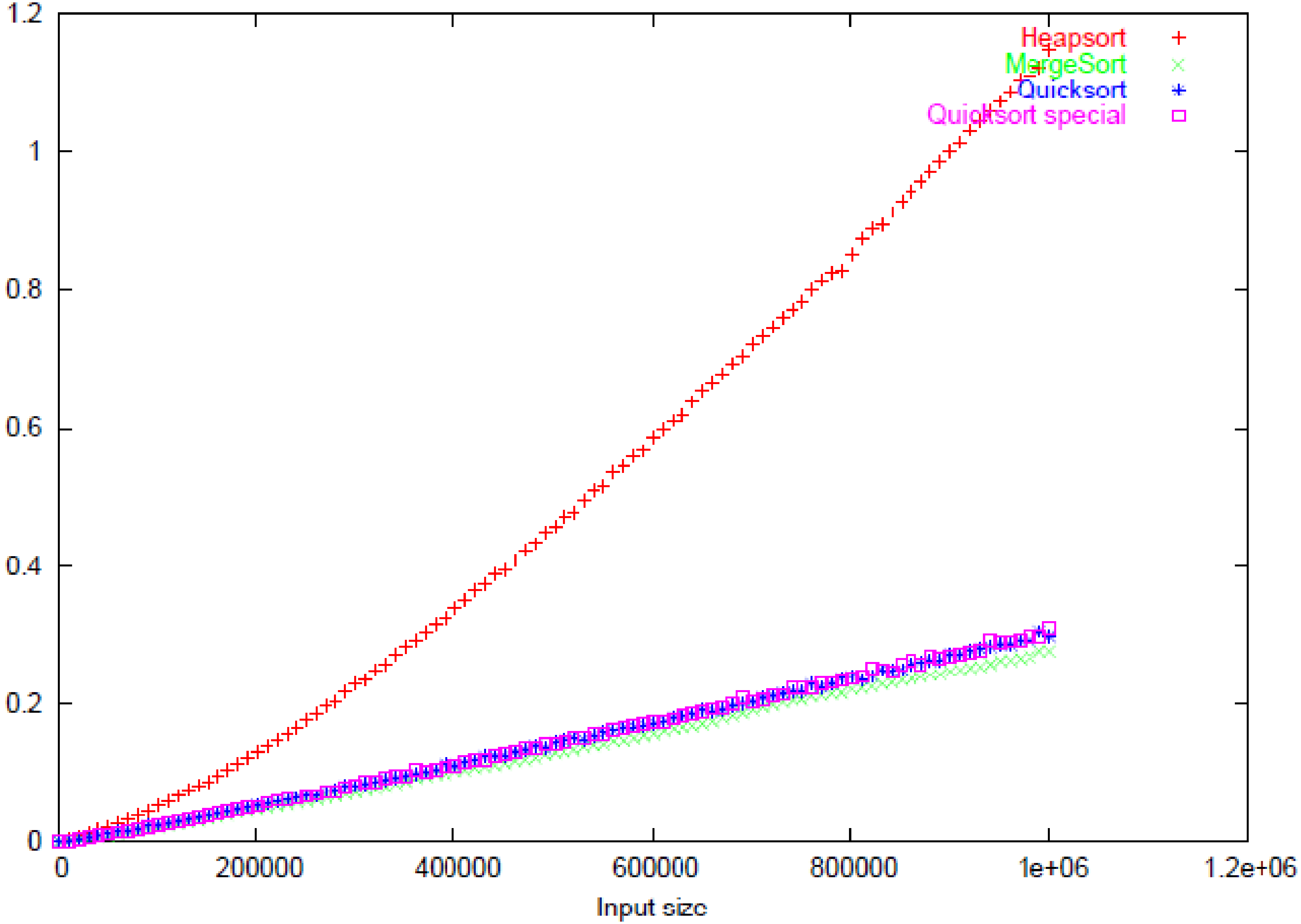


Skift til insertion-sort ved små problemstørrelser n = 300.000 elementer

Tiden for Sorterings Algoritmer



Tiden for Sorterings Algoritmer



Algoritmer og Datastrukturer 1

Gerth Stølting Brodal

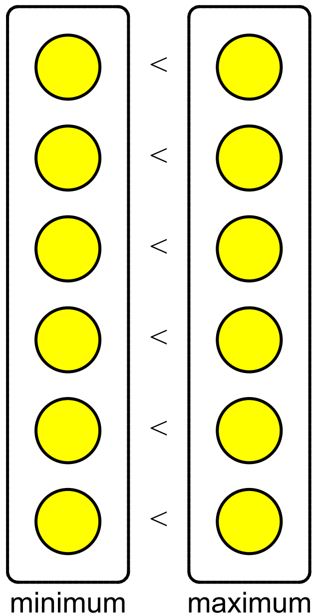
Randomized-select
[CLRS, kapitel 9.1-9.2]

AARHUS UNIVERSITET

Beregning af Minimum of Maximum

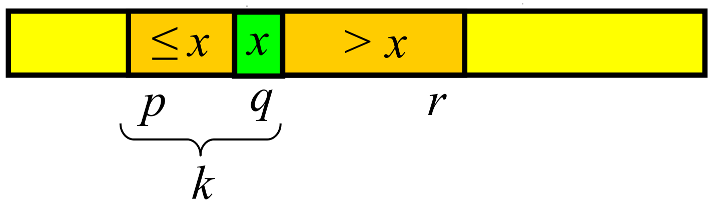
- At finde *minimum* af n elementer kræver n-1 sammenligninger

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



Randomized Select:

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

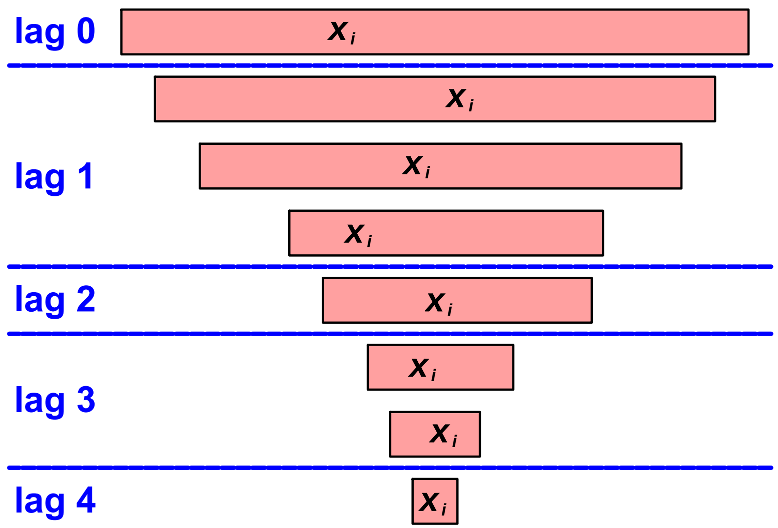



```
RANDOMIZED-SELECT(A, p, r, i)
if p == r
    return A[p]
q = RANDOMIZED-PARTITION(A, p, r)
k = q − p + 1
if i == k              // the pivot value is the answer
    return A[q]
elseif i < k
    return RANDOMIZED-SELECT(A, p, q − 1, i)
else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

Randomized-Select 15

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
18	10	5	4	20	11	22	15	3	17	14	16	19	8	6	21	7	13	9	12	23	2	1
1	10	5	4	20	11	22	15	3	17	14	16	19	8	6	21	7	13	9	12	23	2	18
	10	5	4	11	15	3	17	14	16	8	6	7	13	9	12	2	18	20	21	23	19	22
	2	5	4	11	15	3	17	14	16	8	6	7	13	9	12	10						
		5	4	3	8	6	7	9	10	15	11	17	13	14	12	16						
										15	11	13	14	12	16	17						
										11	12	13	14	15								
												13	14	15								

Randomized Select : Analyse



Forventede tid for randomized select

$= O(\Sigma_j$ **forventede tid** i lag $j)$

$\leq O(\Sigma_j\, n \cdot (3/4)^j \cdot$ **# forventede arrays** i lag $j)$

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

$=$ **$O(n)$** □

Selektion

Algoritme	Tid
Randomized-Select [CLRS, Kap. 9.2]	$O(n)$ forventet $O(n^2)$ worst-case
Deterministic-Select [CLRS, Kap. 9.3] *ikke pensum*	$O(n)$ worst-case