Grundlæggende Algoritmer og Datastrukturer

Quicksort
[CLRS, kapitel 7]

Sandsynlighedsteori Baggrund?

a) Ingen
b) Gymnasiet
c) Gymnasiet + Universitetet
d) Universitetet
e) Andet steds
f) Ved ikke

Sandsynligheden for at slå krone

1/2

Eksperiment
Kast en Mønt

a) Plat
b) Krone
c) På højkant
d) Ingen mønt

Eksperiment
Antal kast før man får krone ?

a) 1
b) 2
c) 3
d) 4
e) 5
f) 6
g) 7
h) 8
i) 9 eller flere

Sætning

Forventede antal kast for at få krone = 2

Bevis

$$\sum_{i=1}^{\infty} i \cdot \left(\frac{1}{2}\right)^{i-1} \cdot \frac{1}{2} = \sum_{i=1}^{\infty} i \cdot \left(\frac{1}{2}\right)^{i} = 2$$

kast

s.s. for først i-1 kast er plat

s.s. for i'te kast er krone

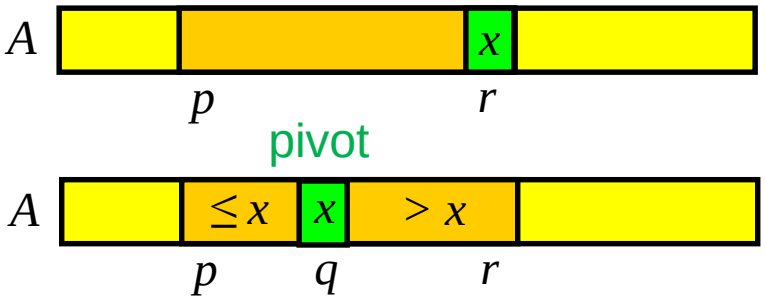
□

cs.au.dk/~gerth/slides/math.pdf

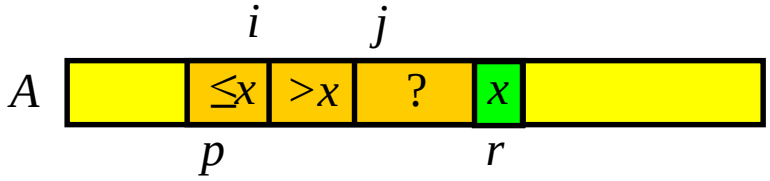
Quicksort:

Sorter $A[p..r]$

```
QUICKSORT(A, p, r)
1  if p < r
2      q = PARTITION(A, p, r)
3      QUICKSORT(A, p, q − 1)
4      QUICKSORT(A, q + 1, r)
```



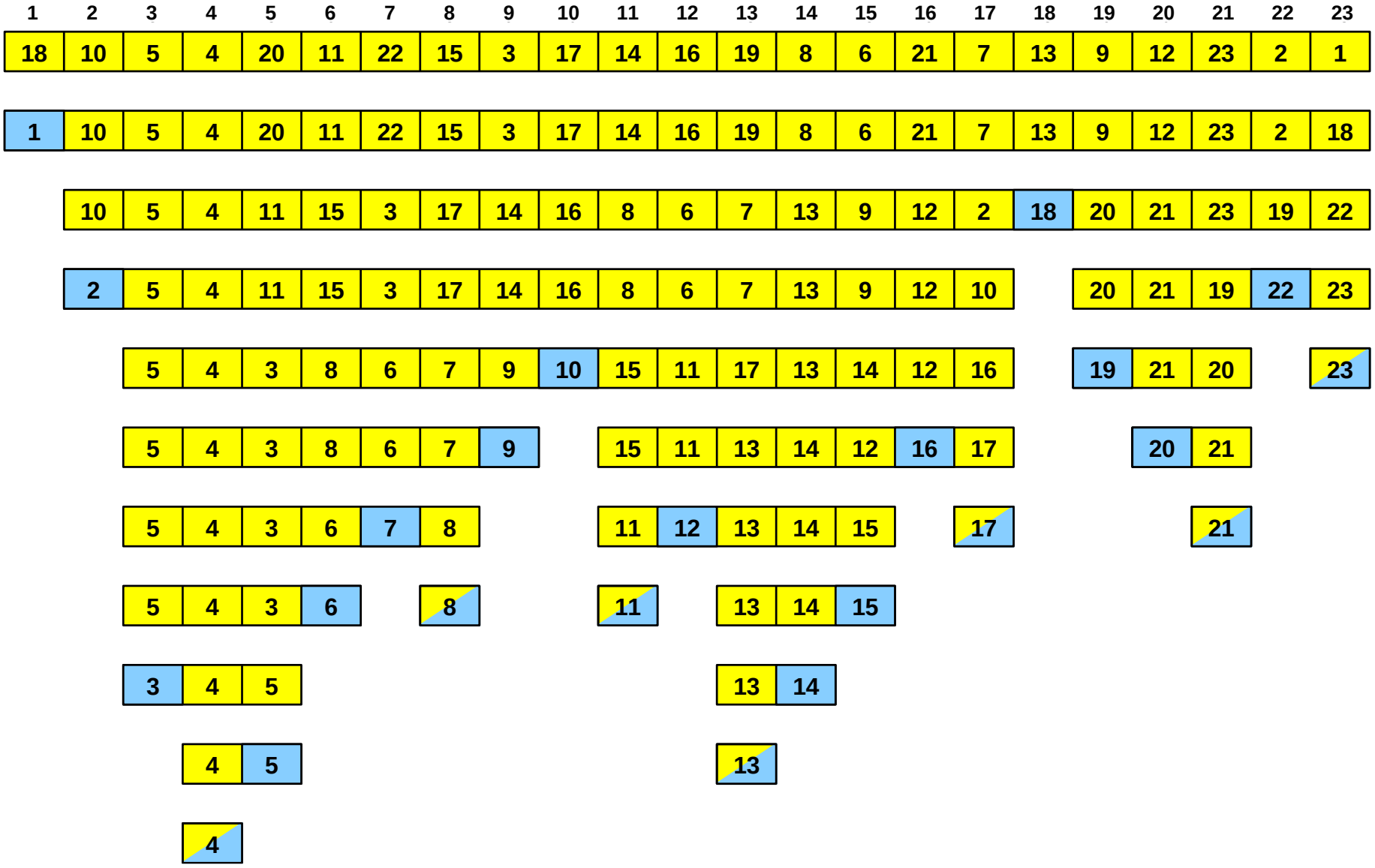

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```



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

Hoare, 1961

Quicksort på 23 elementer



Hvor Mange Kald til Partition ?

```
QUICKSORT(A, p, r)
1   if p < r
2       q = PARTITION(A, p, r)
3       QUICKSORT(A, p, q − 1)
4       QUICKSORT(A, q + 1, r)
```

a) O(log n)

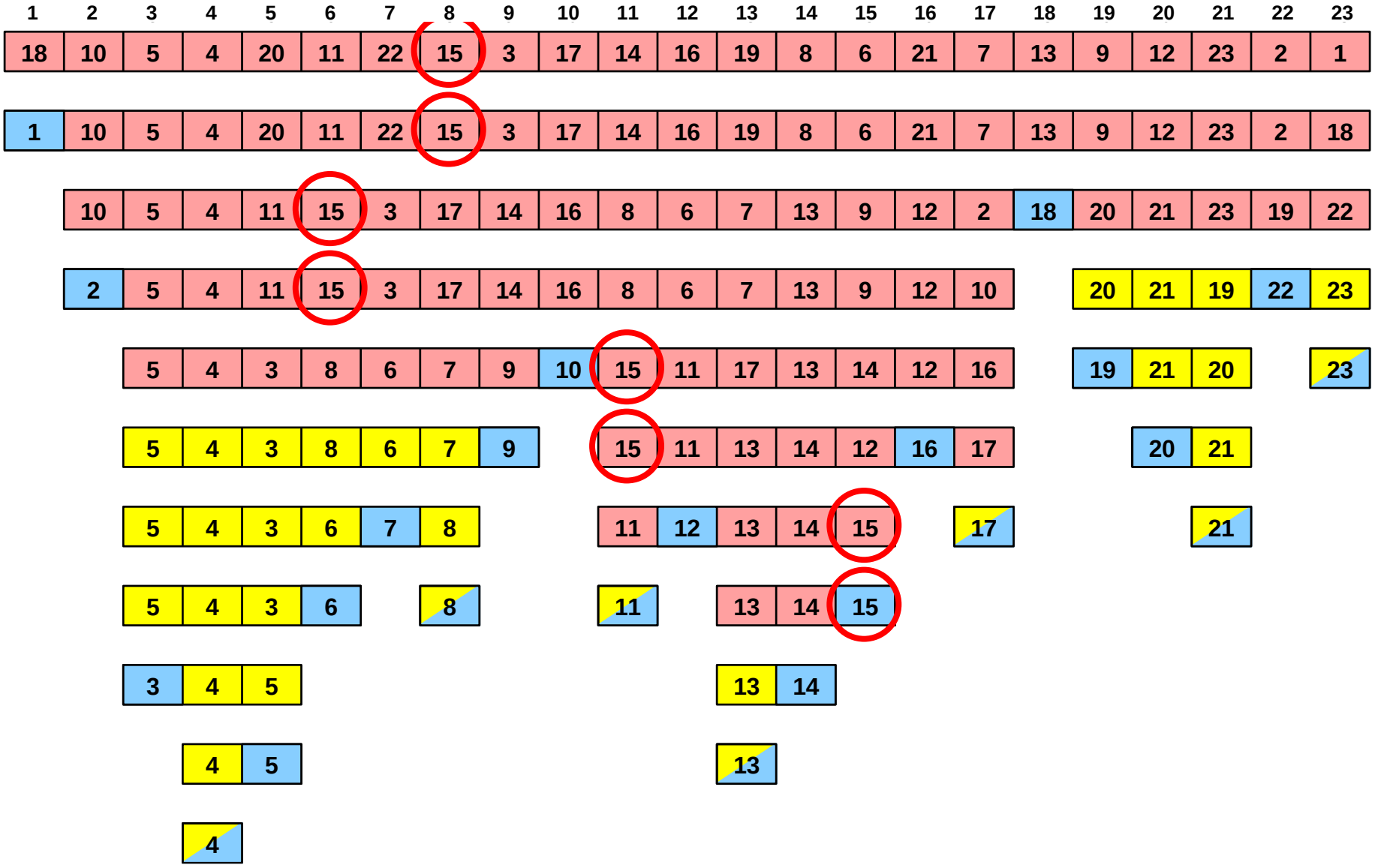
b) O(n)

c) O(n log n)

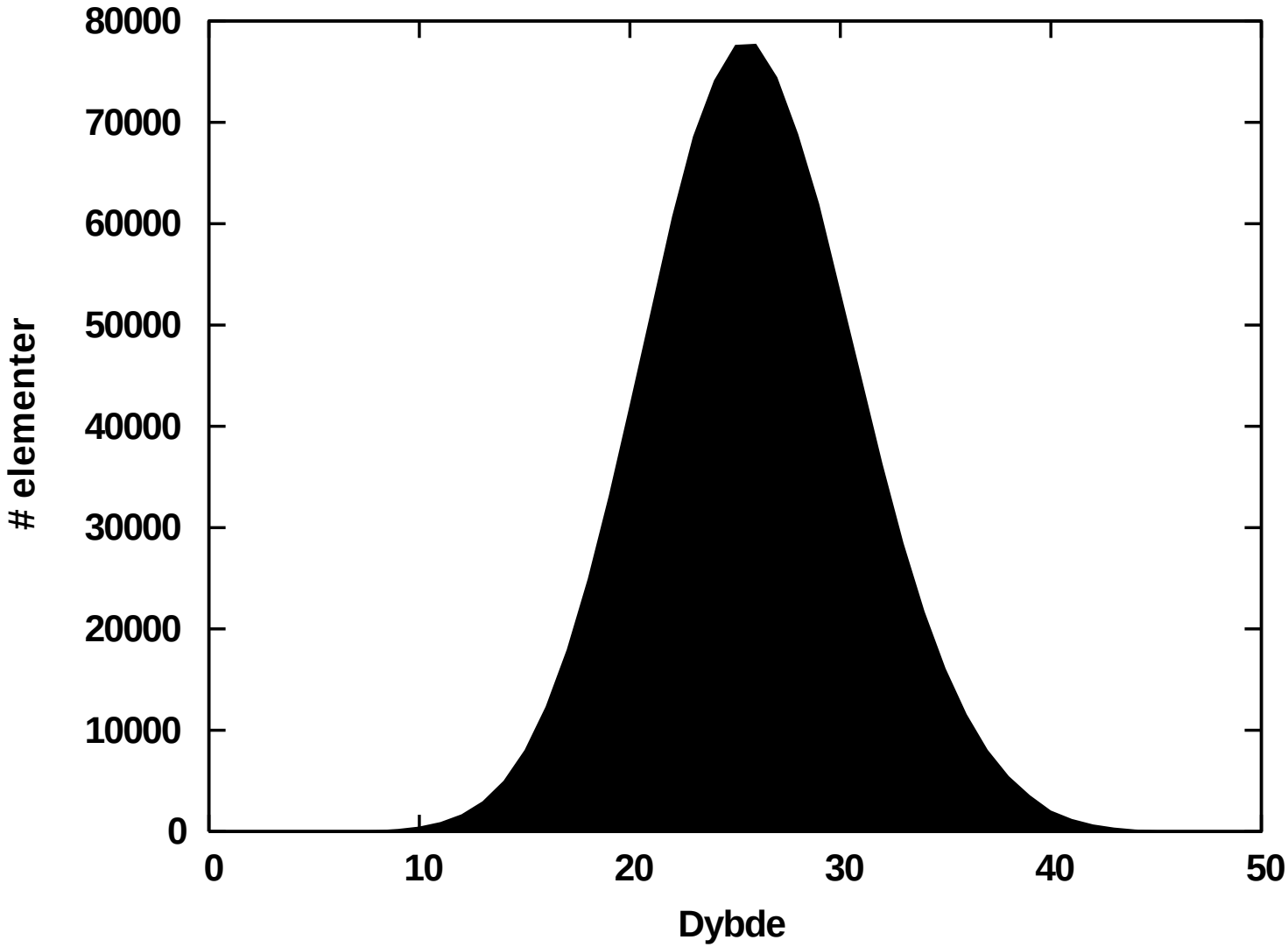
d) O(n^2)

e) Ved ikke

Quicksort : Rekursionen for 15



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



Randomized Quicksort

```
RANDOMIZED-QUICKSORT(A, p, r)
1  if p < r
2      q = 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 = RANDOM(p, r)
2  exchange A[r] with A[i]
3  return PARTITION(A, p, r)
```

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

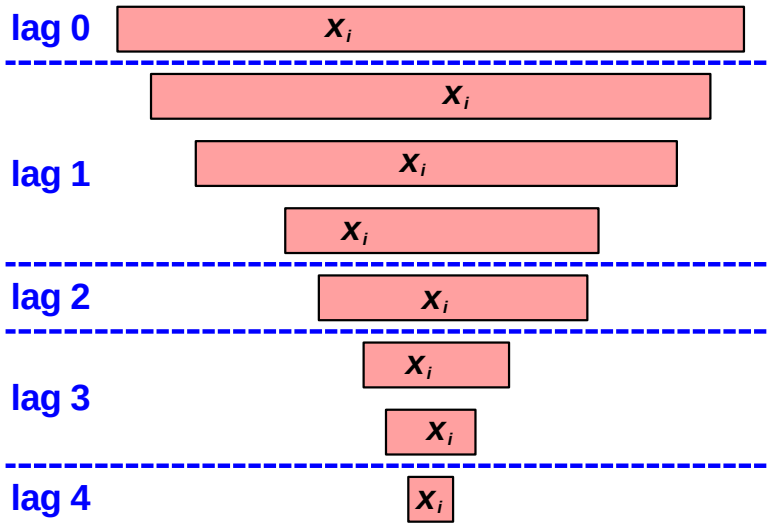
Forventede antal mønt-kast før man får krone 3 gange?

a) Ingen af nedenstående
b) Ved ikke
c) 3
d) 4
e) 5
f) 6
g) 7
h) 8
i) 9

Sætning

Forventede antal kast for at få krone **k** gange er **2k**

Randomized Quicksort : Analyse



Lag i
hvert array
længde $\leq n(3/4)^i$

- 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

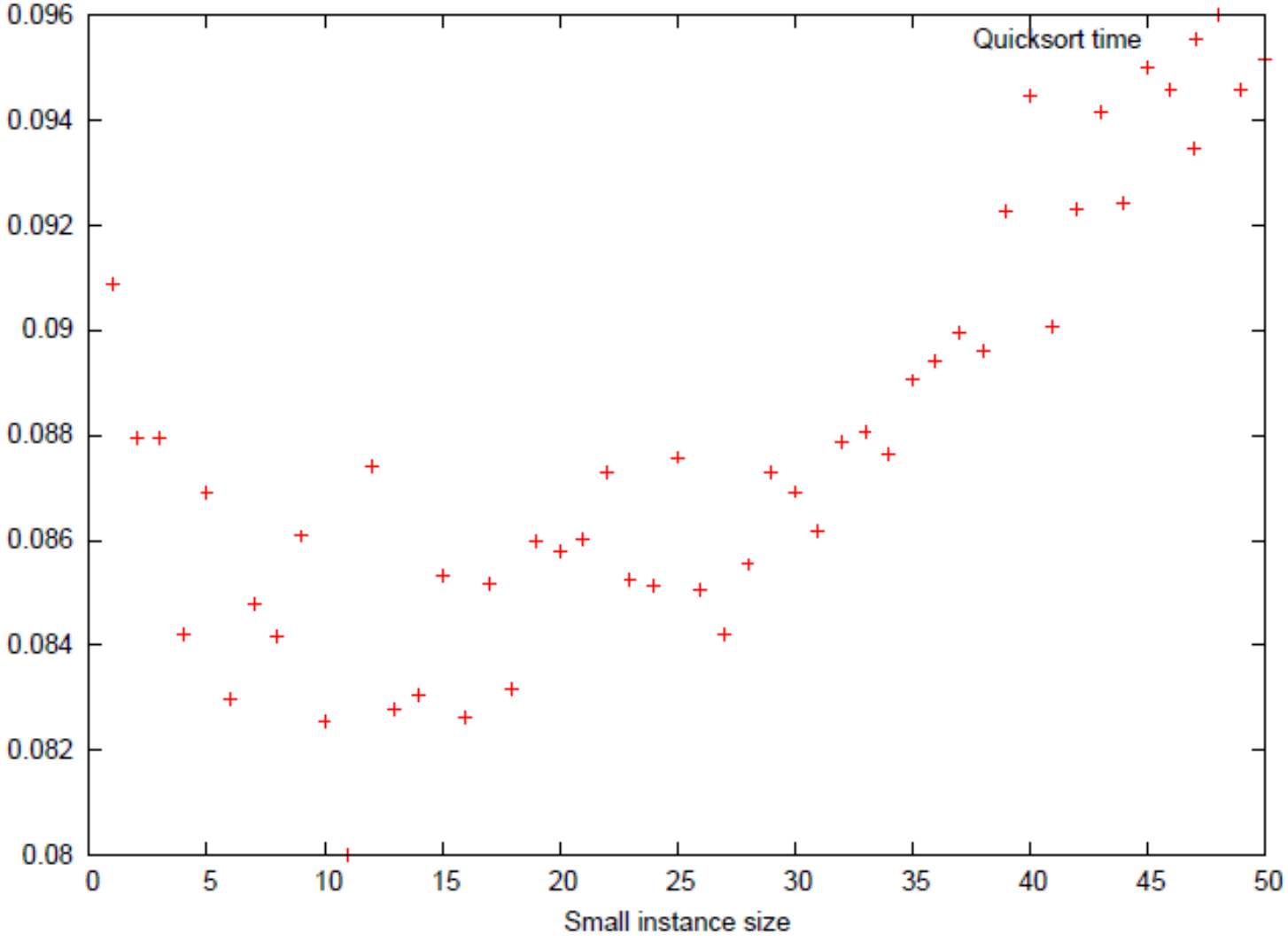
Algoritme	Worst-Case Tid
Heap-Sort	$O(n \cdot \log n)$
Merge-Sort	
Insertion-Sort	$O(n^2)$
QuickSort (Deterministic og randomiseret)	$O(n^2)$

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

Sortering:

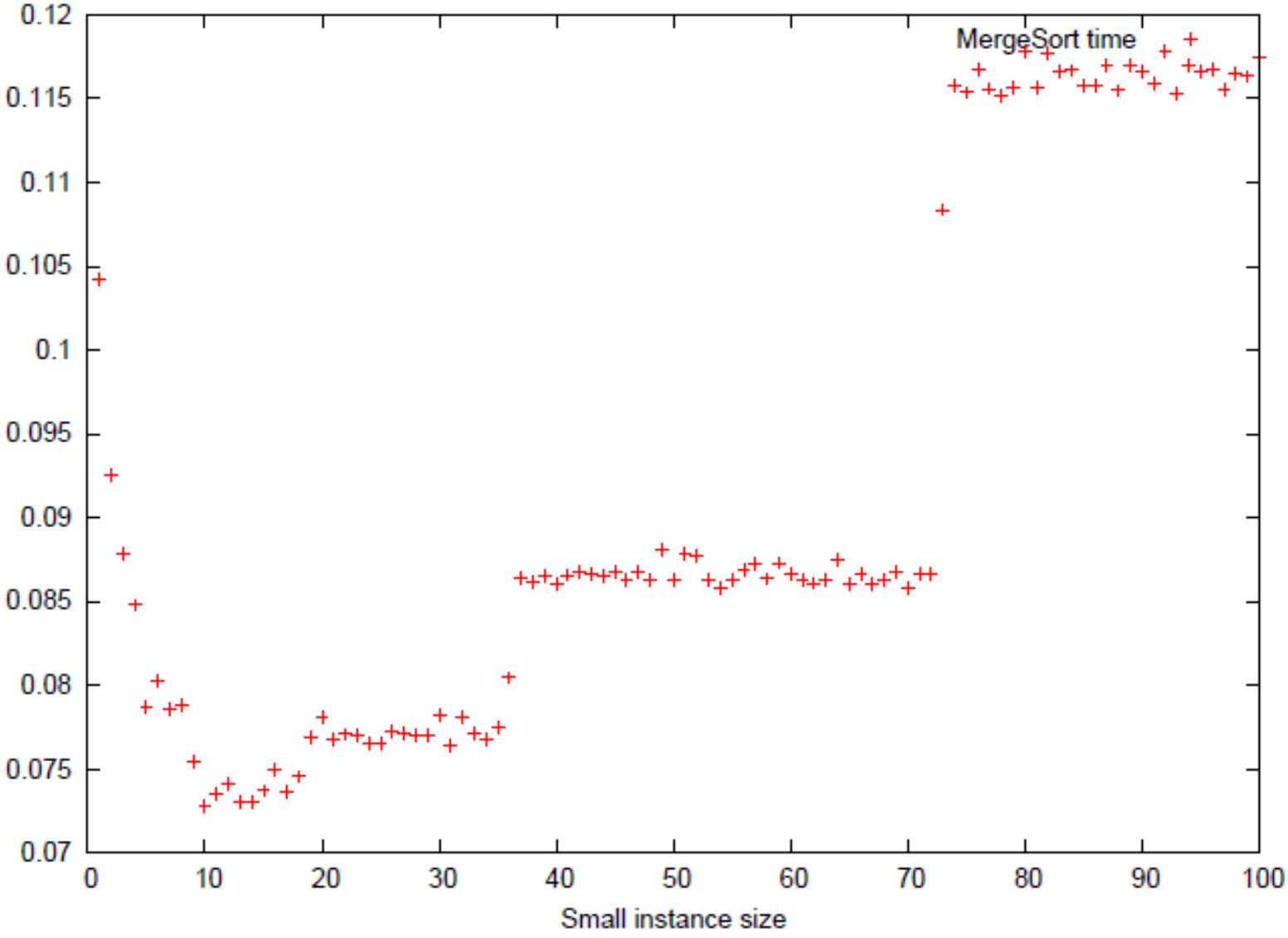
Eksperimentelle resultater

Quicksort med skift til Insertion-sort



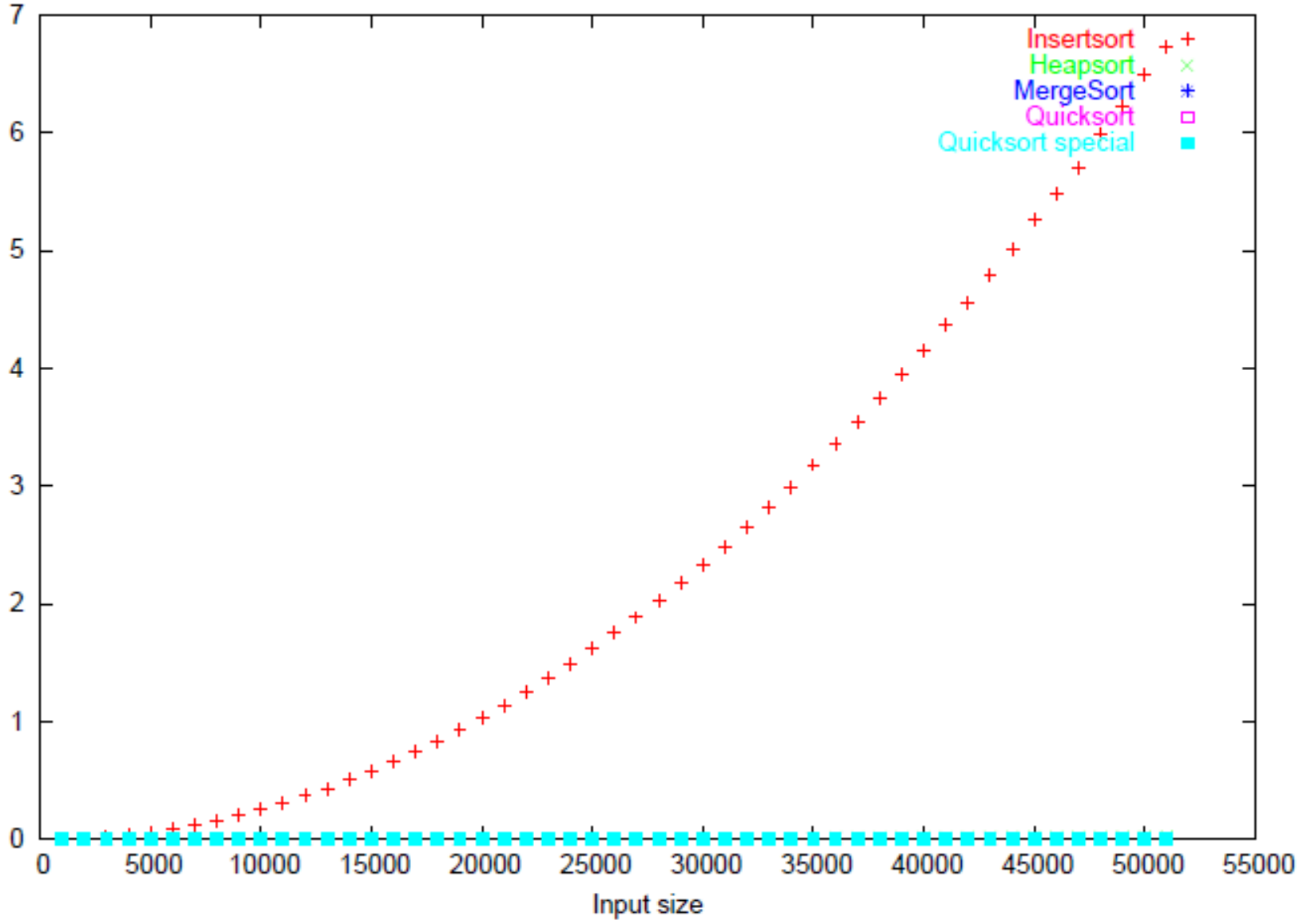
Skift til insertion-sort ved små problemstørrelser n = 300.000 elementer

Mergesort med skift til Insertion-sort

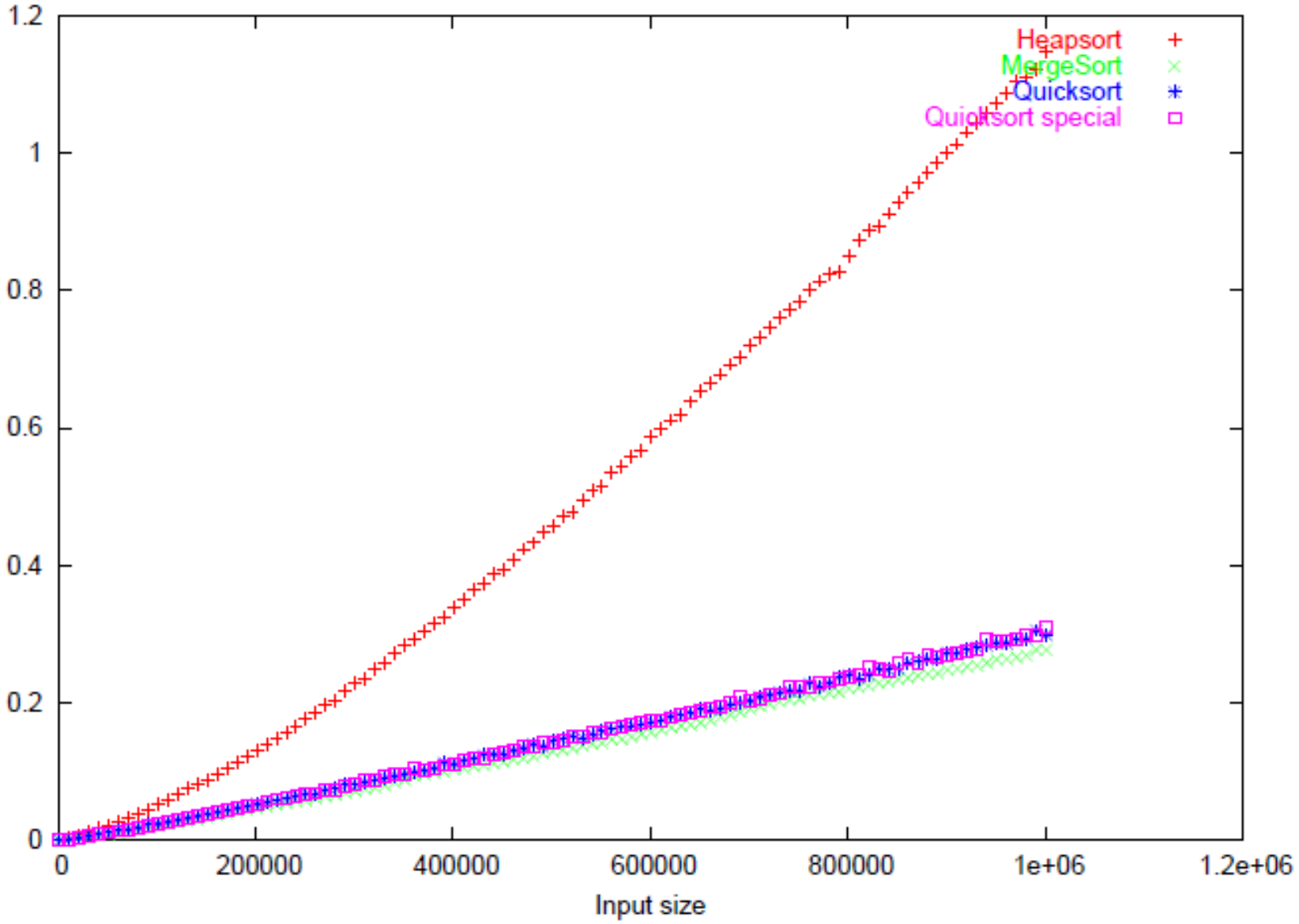


Skift til insertion-sort ved små problemstørrelser n = 300.000 elementer

Tiden for Sorterings Algoritmer



Tiden for Sorterings Algoritmer



Sortering i Praksis

Typisk anvendes **QuickSort** (C, C++, Java)

- Inplace (kræver ingen yderligere arrays)
- Hurtig

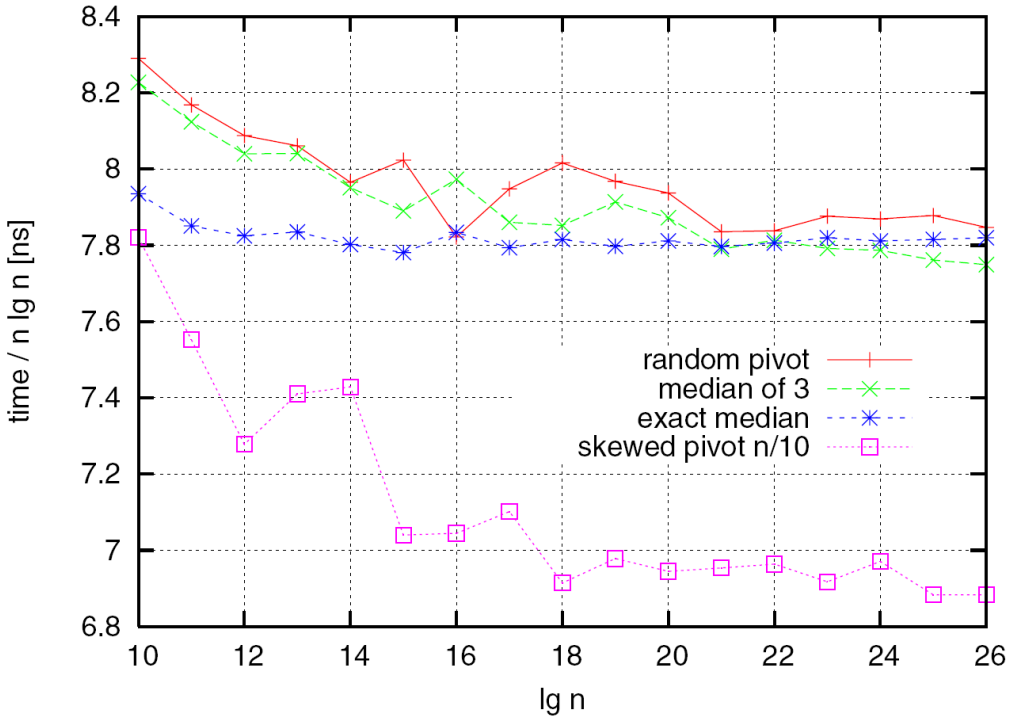
...dog med et par twists

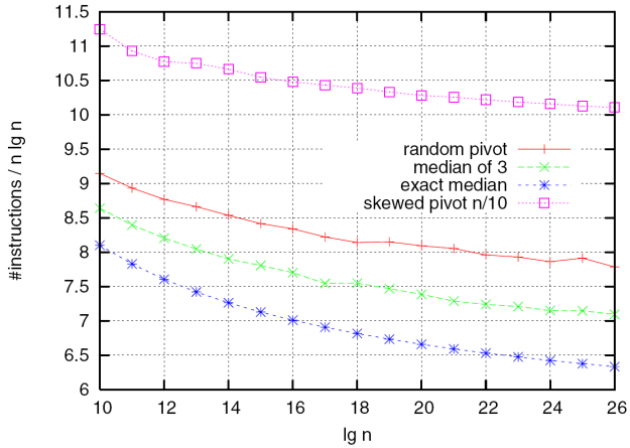
- skift til **InsertionSort** i bunden af rekursionen
- skift til **MergeSort** hvis QuickSort tager for lang tid
- Vælg Pivot'et som medianen ud af O(1) (tilfældige) elementer

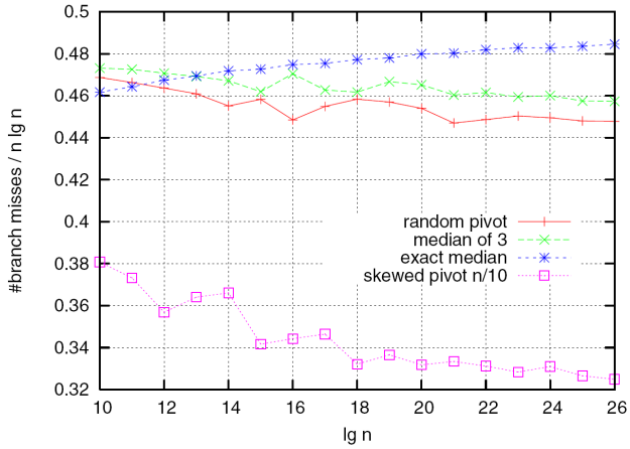
“How Branch Mispredictions Affect Quicksort”

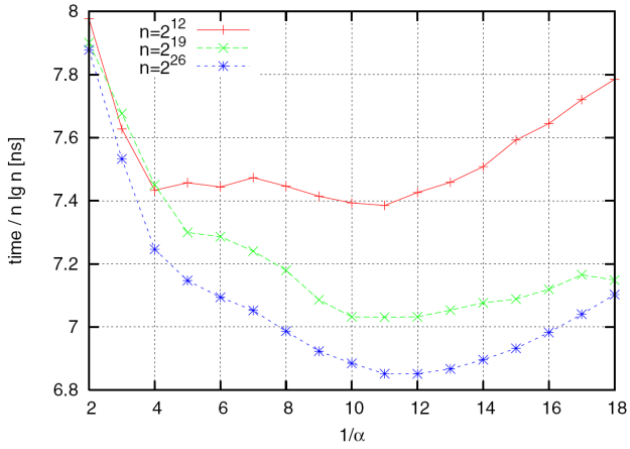
Kanela Kaligosi and Peter Sanders

14th Annual European Symposium on Algorithms (ESA 2006)









3GHz Pentium 4 Prescott, GCC 3.3, PAPI, gns. 100 kørsler med tilfældige permutationer af [1..n]

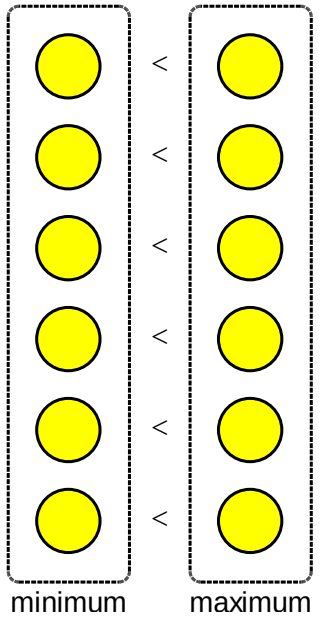
Grundlæggende Algoritmer og Datastrukturer

Randomized-select
[CLRS, kapitel 9.1-9.2]

Beregning af Minimum of Maximum

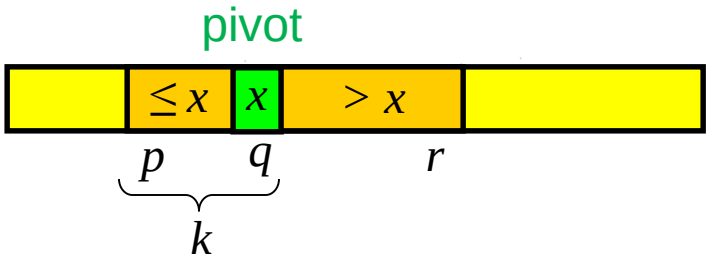
- At finde *minimum* af n elementer kræver n-1 sammenligninger

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



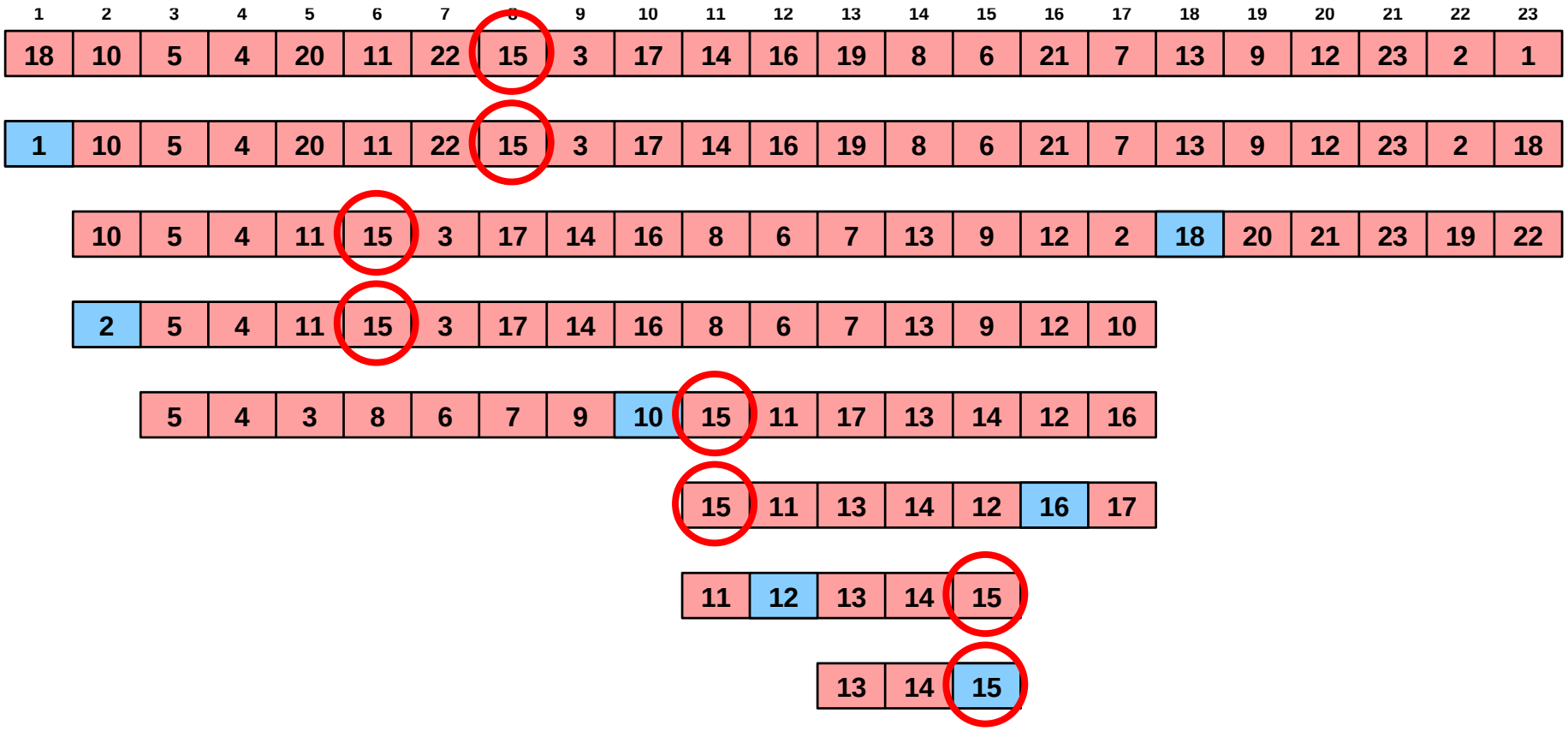
Randomized Select:

Find det *i*'te mindste element i $A[p..r]$ $(1 \le i \le r\text{-}p\text{+}1)$




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

Randomized-Select 15

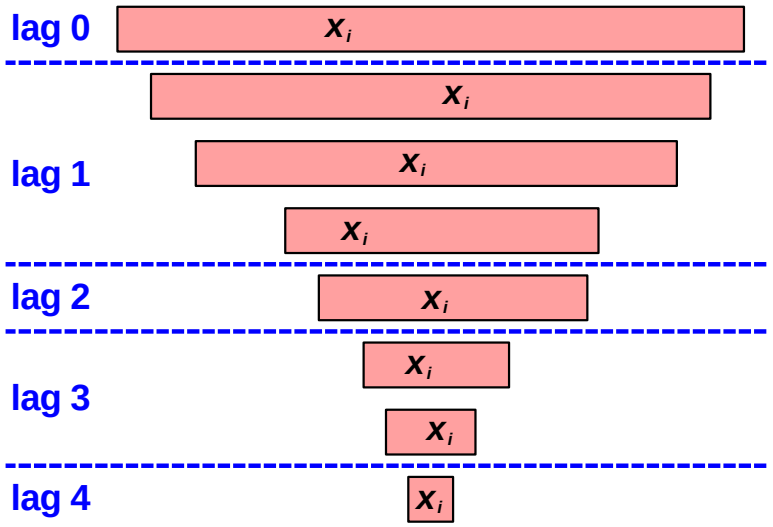


Worst-case tid for Randomized-Select ?

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

a) O(log n)
b) O(n)
c) O(n log n)
d) O(n^2)
e) Ved ikke

Randomized Select : Analyse



Forventede tid for randomized select

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

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

$= O(\Sigma_j\ n\cdot(3/4)^j \cdot 2)$ ← $\sum_{i=0}^{\infty} c^i = \frac{1}{1-c}$ for $0 < c < 1$

$= \mathbf{O(n)}$

□

Selektion

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