Algoritmer og Datastrukturer 1

...mere Sortering [CLRS, kapitel 8]

Gerth Stølting Brodal

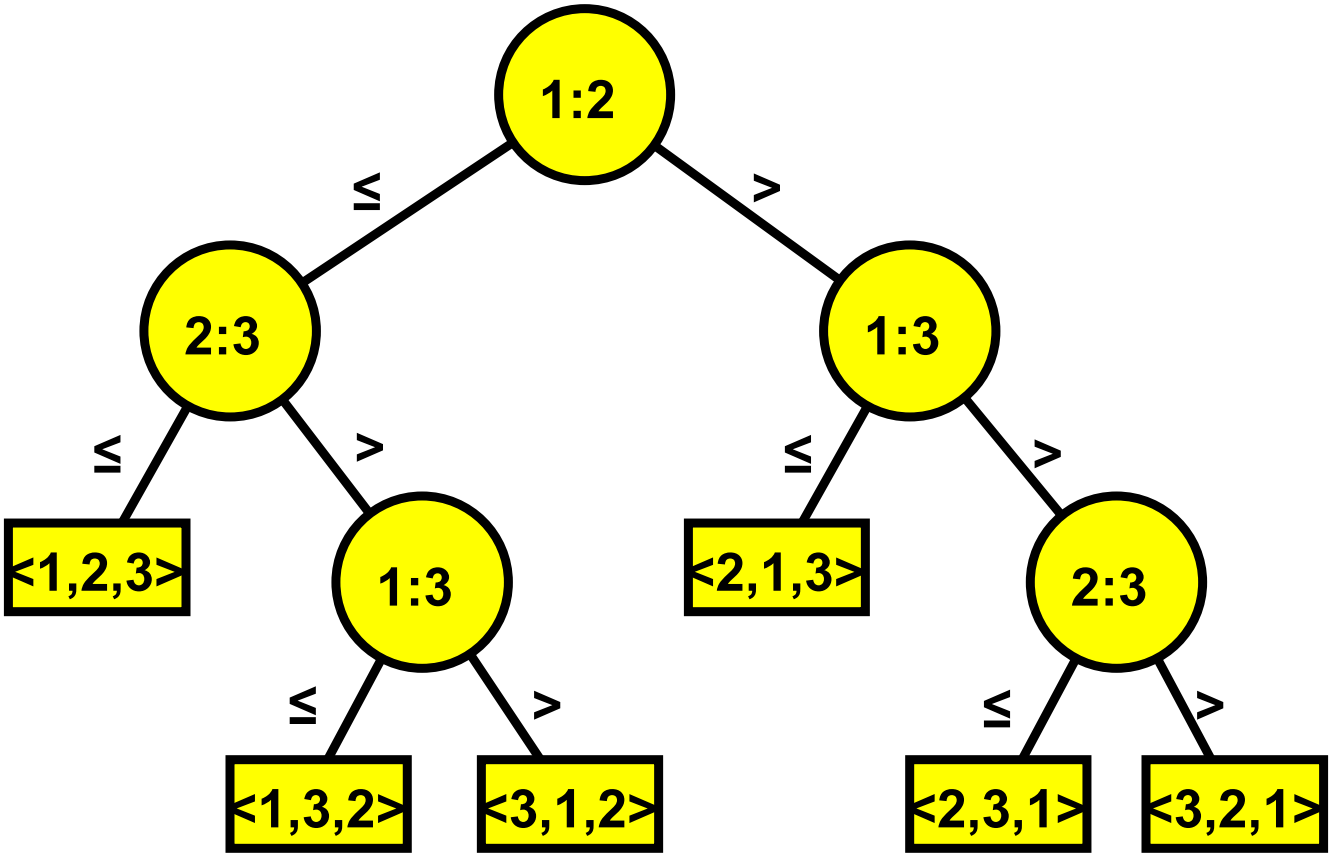
Aarhus Universitet

Sorterings-algoritmer
(sammenligningsbaserede)

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

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

Sortering ved sammmenligninger: Nedre grænse



Interne knude ***i*:*j*** sammenligner x_i og x_j

Blade beskriver output **permutation**

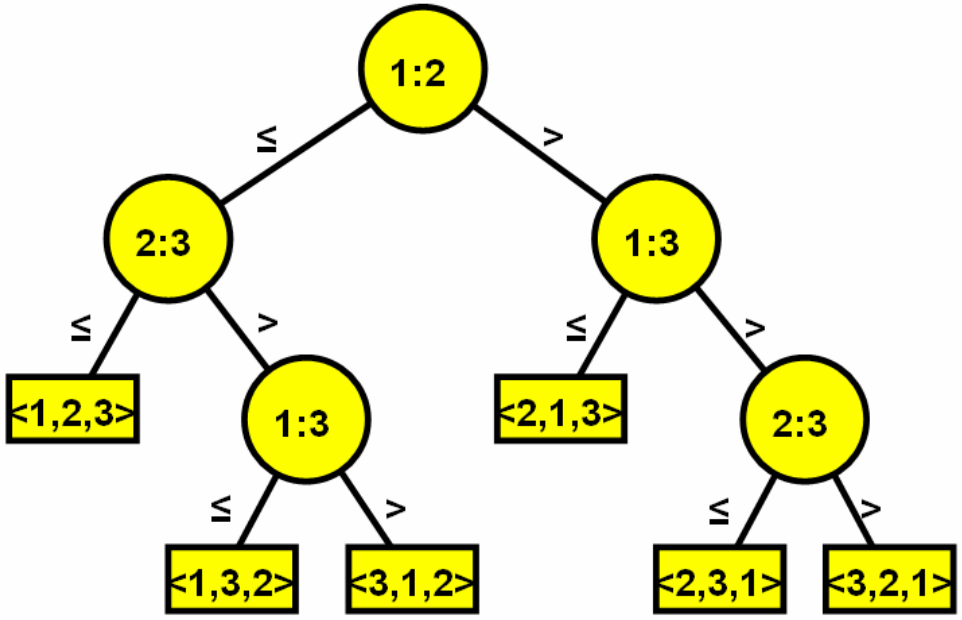
Sortering ved sammmenligninger: Nedre grænse

$n!$ forskellige output

$\geq n!$ forskellige blade

træ af højde h har

$\leq 2^h$ blade



$$n! \leq 2^h$$

$$h \geq \log(n!) \geq \log((n/2)^{n/2}) = \Omega(n \cdot \log n)$$

Worst-case $\Omega(n \cdot \log n)$ sammenligninger

Sortering af heltal

...udnyt at elementer er bitstrenge

Counting-Sort:

Input: A, ouput: B, tal fra $\{0,1, \ldots,k\}$

```
COUNTING-SORT(A, B, k)
 1  for i ← 0 to k
 2       do C[i] ← 0
 3  for j ← 1 to length[A]
 4       do C[A[j]] ← C[A[j]] + 1
 5  ▷ C[i] now contains the number of elements equal to i.
 6  for i ← 1 to k
 7       do C[i] ← C[i] + C[i − 1]
 8  ▷ C[i] now contains the number of elements less than or equal to i.
 9  for j ← length[A] downto 1
10       do B[C[A[j]]] ← A[j]
11          C[A[j]] ← C[A[j]] − 1
```

Worst-case tid $O(n+k)$

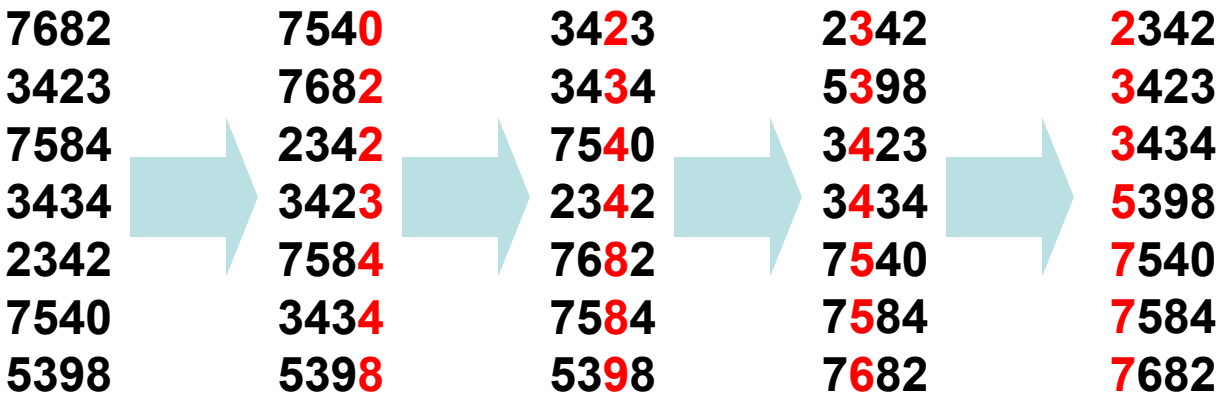
Radix-Sort:

Input: array A, tal med d cifre fra $\{0,1,...,k\}$

RADIX-SORT(A, d)

1 **for** $i \leftarrow 1$ **to** d

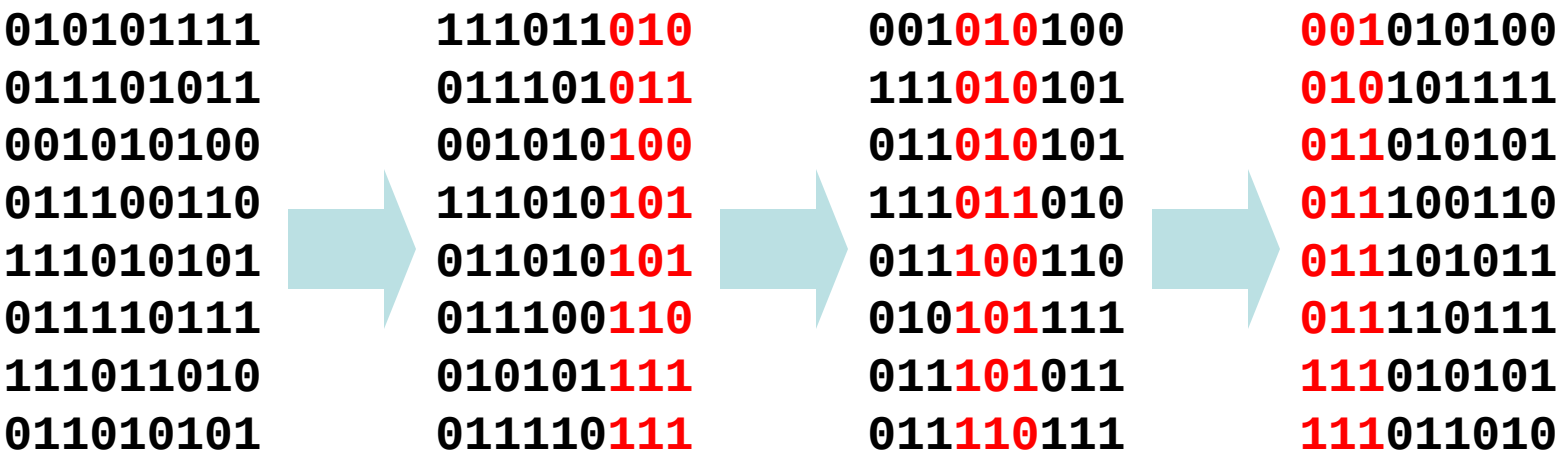
2 **do** use a stable sort to sort array A on digit i



Worst-case tid $O(d \cdot (n+k))$

Radix-Sort:

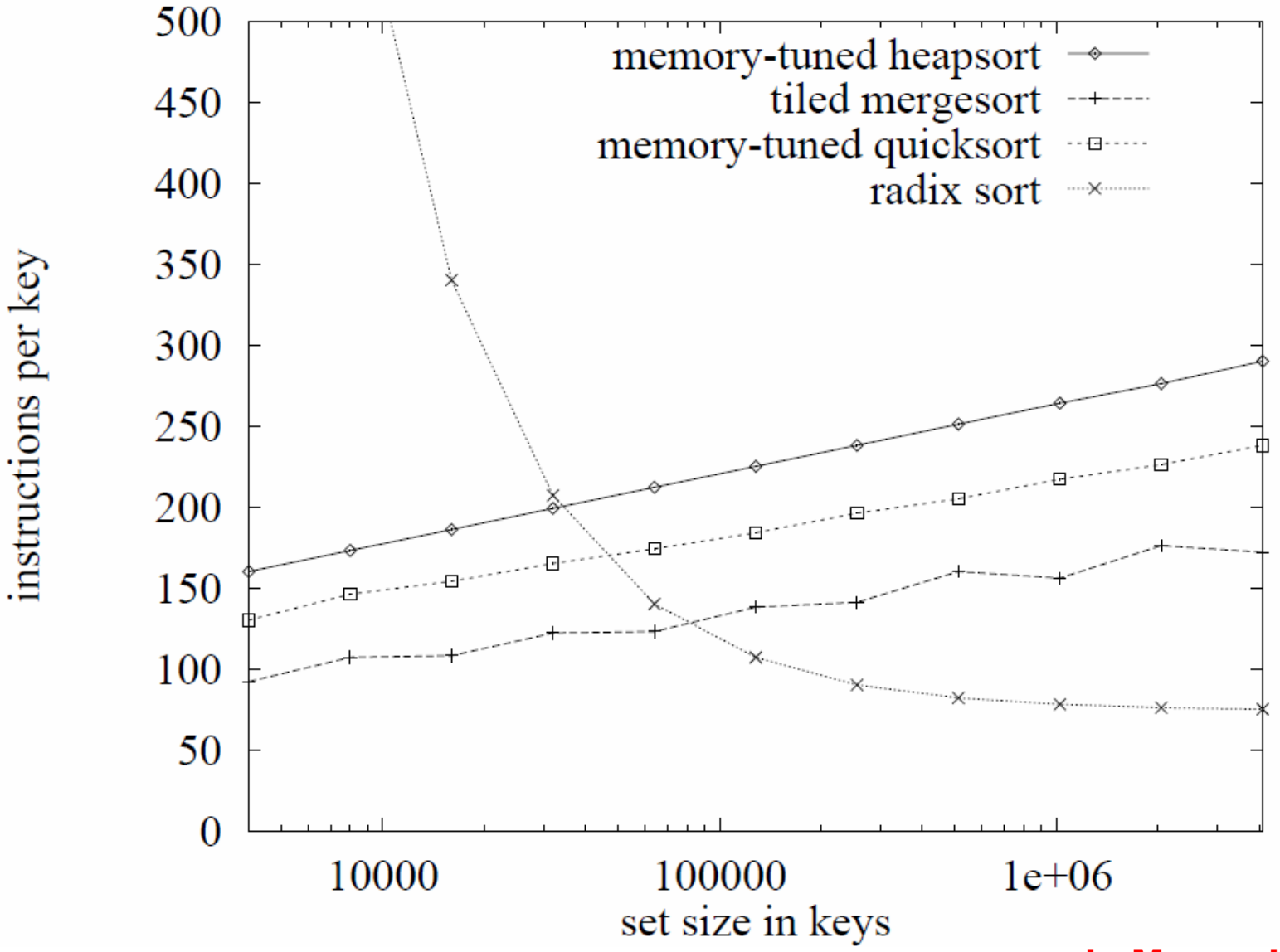
Input: array A, n tal med b bits



n = 8, k = 7 (3 bits = **log n** bits), d = 3 (3 x 3 bits)

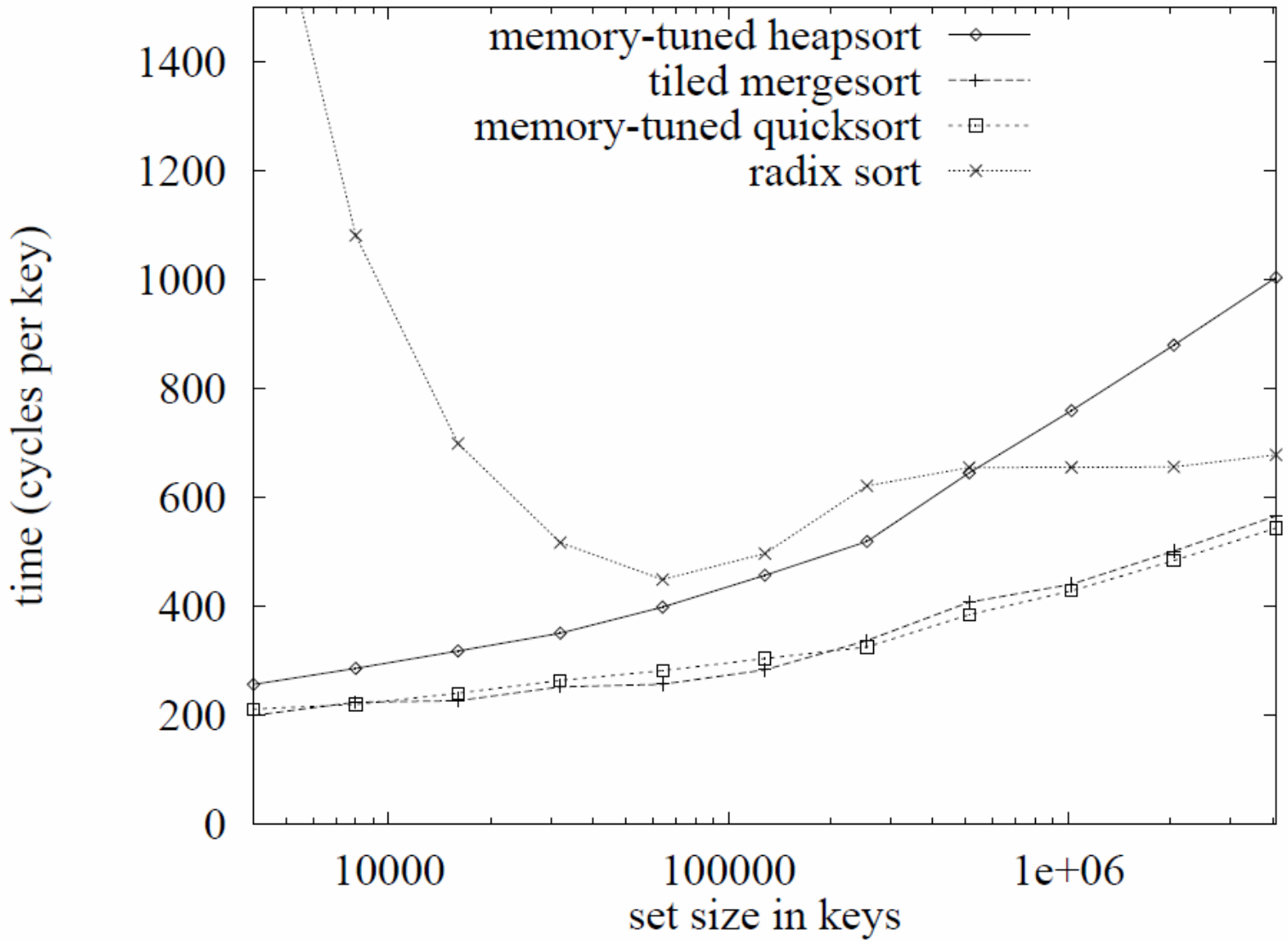
Input n tal med b bits: **Worst-case tid $O(n \cdot b/\log n)$**

Radix-Sort: Eksperimenter



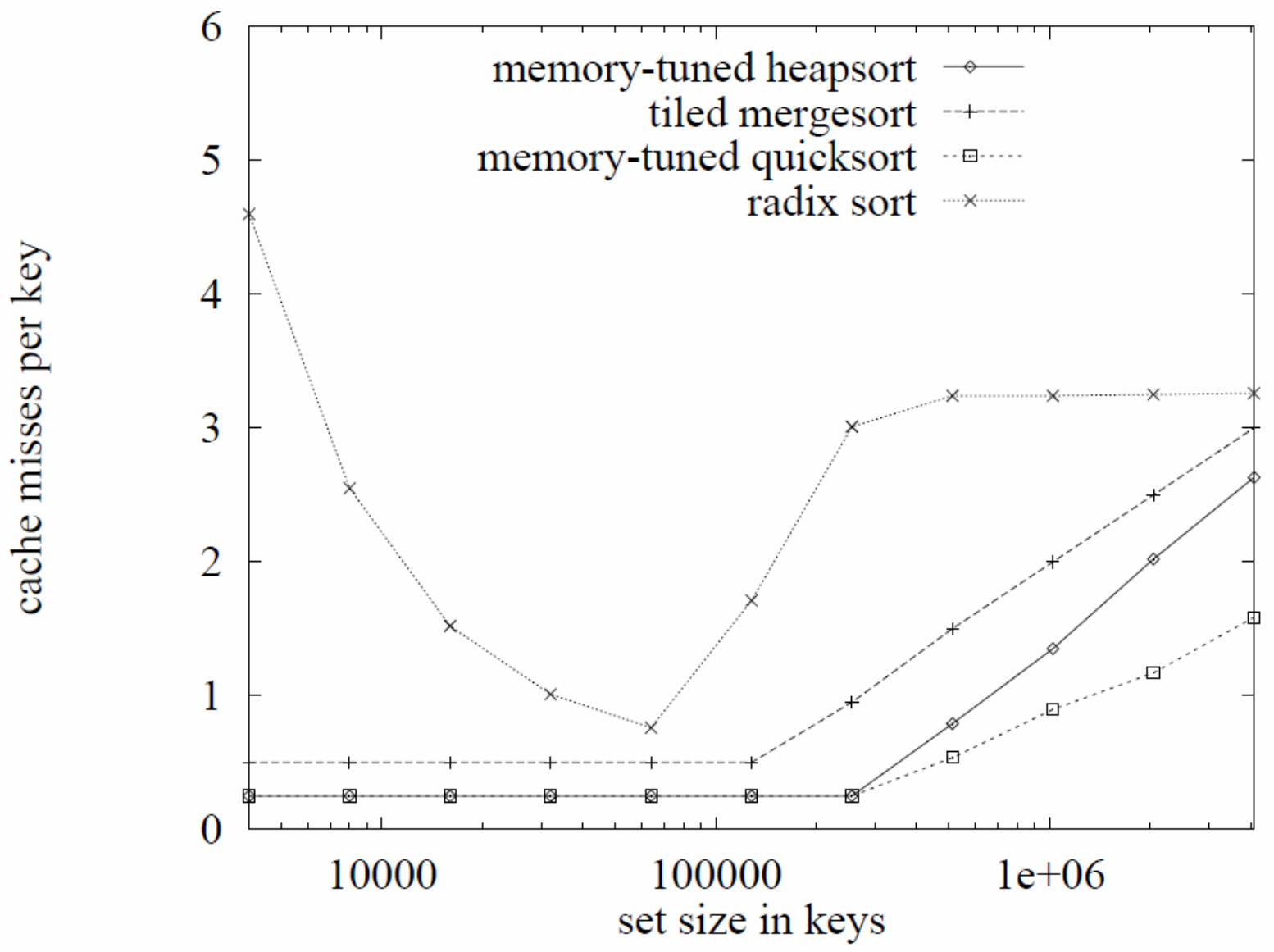
LaMarca, Ladner ’97

Radix-Sort: Eksperimenter



LaMarca, Ladner ’97

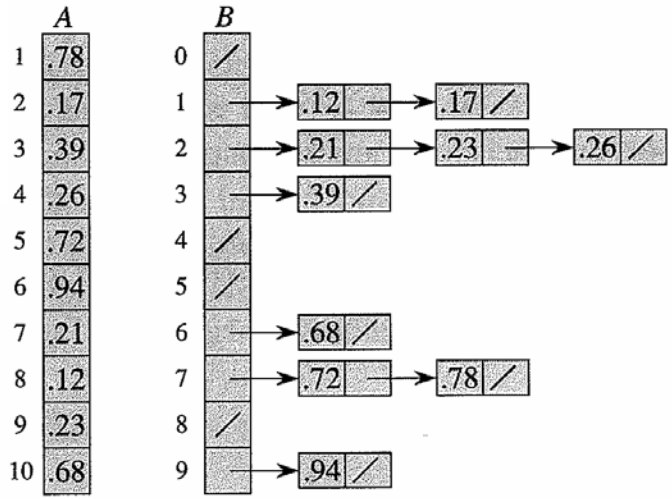
Radix-Sort: Eksperimenter



LaMarca, Ladner ’97

Bucket-Sort:

Input: A, reelle tal fra [0..1]




```
BUCKET-SORT(A)
1  n ← length[A]
2  for i ← 1 to n
3        do insert A[i] into list B[⌊nA[i]⌋]
4  for i ← 0 to n − 1
5        do sort list B[i] with insertion sort
6  concatenate the lists B[0], B[1], . . . , B[n − 1] together in order
```

Forventet tid $O(n)$ – for tilfældigt input