Algoritmer og Datastrukturer 1

Gerth Stølting Brodal

...mere Sortering
[CLRS, kapitel 8]

AARHUS UNIVERSITET

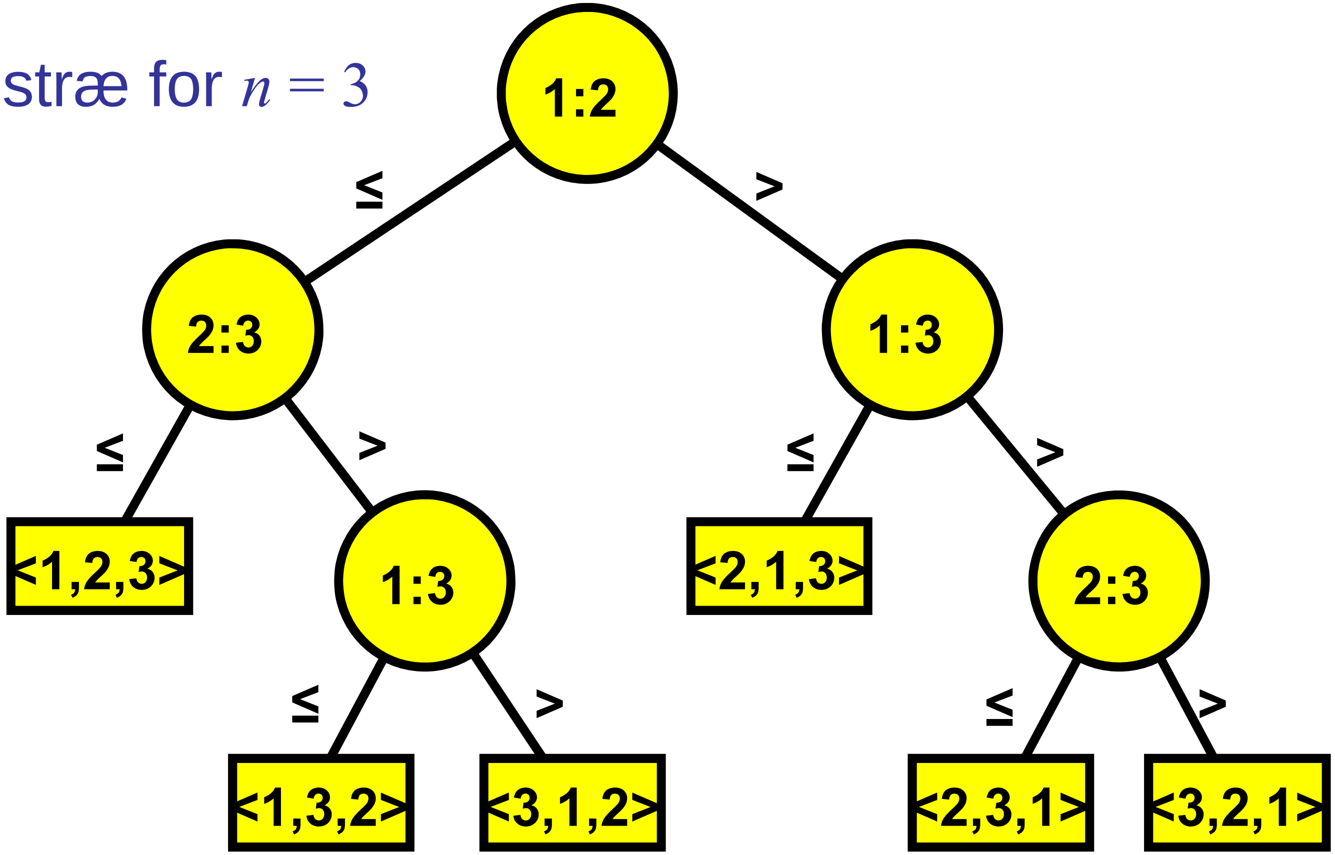
Sorterings-algoritmer
(sammenligningsbaserede)

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

Beslutningstræ for $n = 3$



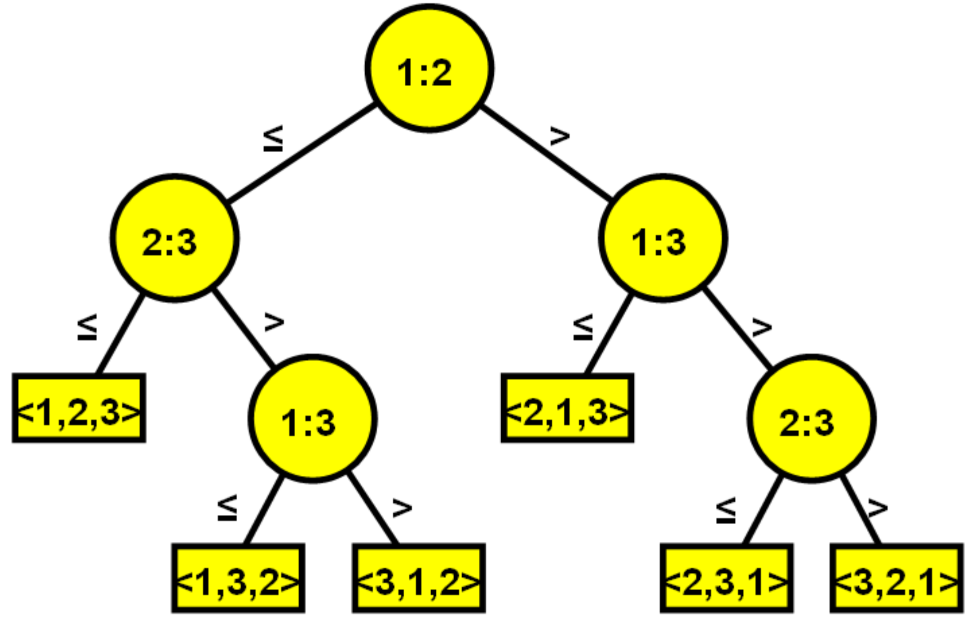
Interne knude $i{:}j$ sammenligner x_i og x_j

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)
for i ← 0 to k
    do C[i] ← 0
for j ← 1 to length[A]
    do C[A[j]] ← C[A[j]] + 1
▷ C[i] now contains the number of elements equal to i.
for i ← 1 to k
    do C[i] ← C[i] + C[i − 1]
▷ C[i] now contains the number of elements less than or equal to i.
for j ← length[A] downto 1
    do B[C[A[j]]] ← A[j]
       C[A[j]] ← C[A[j]] − 1
```

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

Radix-Sort:

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

RADIX-SORT(A, d)

for $i \leftarrow 1$ **to** d
  do use a $\boxed{\text{stable}}$ sort to sort array A on digit i

7682	7540	3423	2342	2342
3423	7682	3434	5398	3423
7584	2342	7540	3423	3434
3434	3423	2342	3434	5398
2342	7584	7682	7540	7540
7540	3434	7584	7584	7584
5398	5398	5398	7682	7682

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

Radix-Sort:

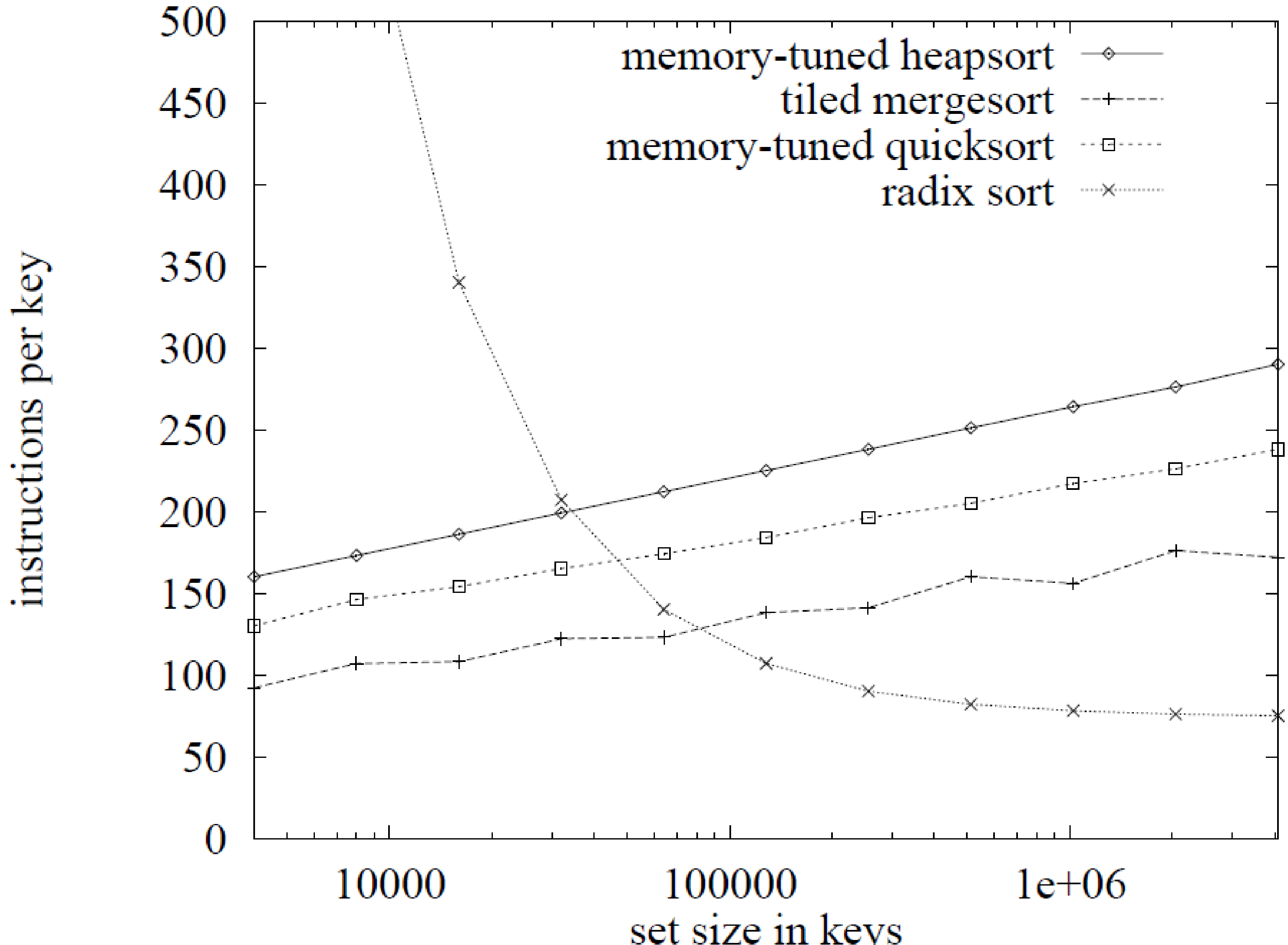
Input: array A, n tal med b bits

010101111	111011010	001010100	001010100
011101011	011101011	111010101	010101111
001010100	001010100	011010101	011010101
011100110	111010101	111011010	011100110
111010101	011010101	011100110	011101011
011110111	011100110	010101111	011110111
111011010	010101111	011101011	111010101
011010101	011110111	011110111	111011010

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

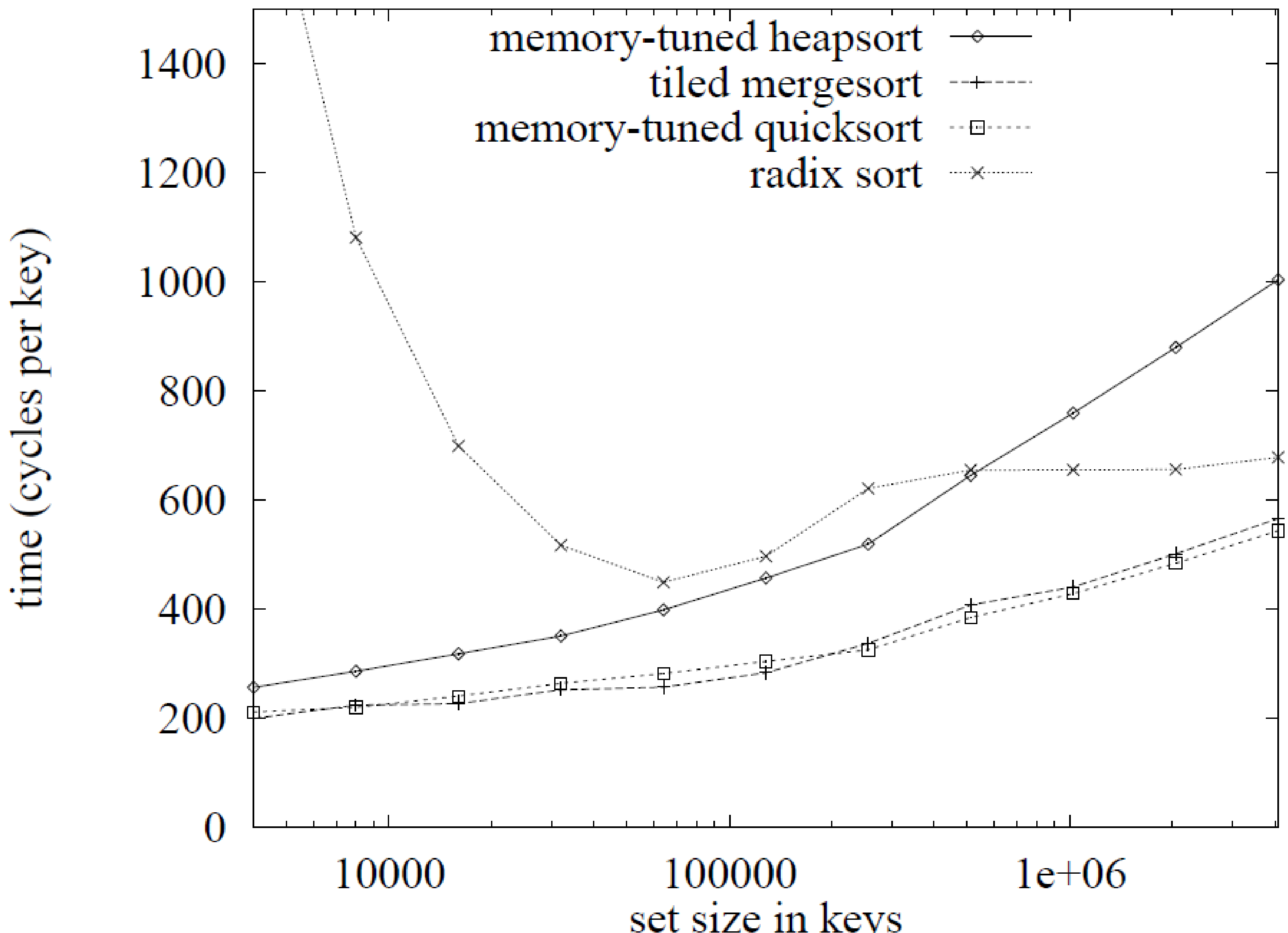
Input n tal med b bits: **Worst-case tid** $O(n \cdot b / \log n)$

Radix-Sort: Eksperimenter



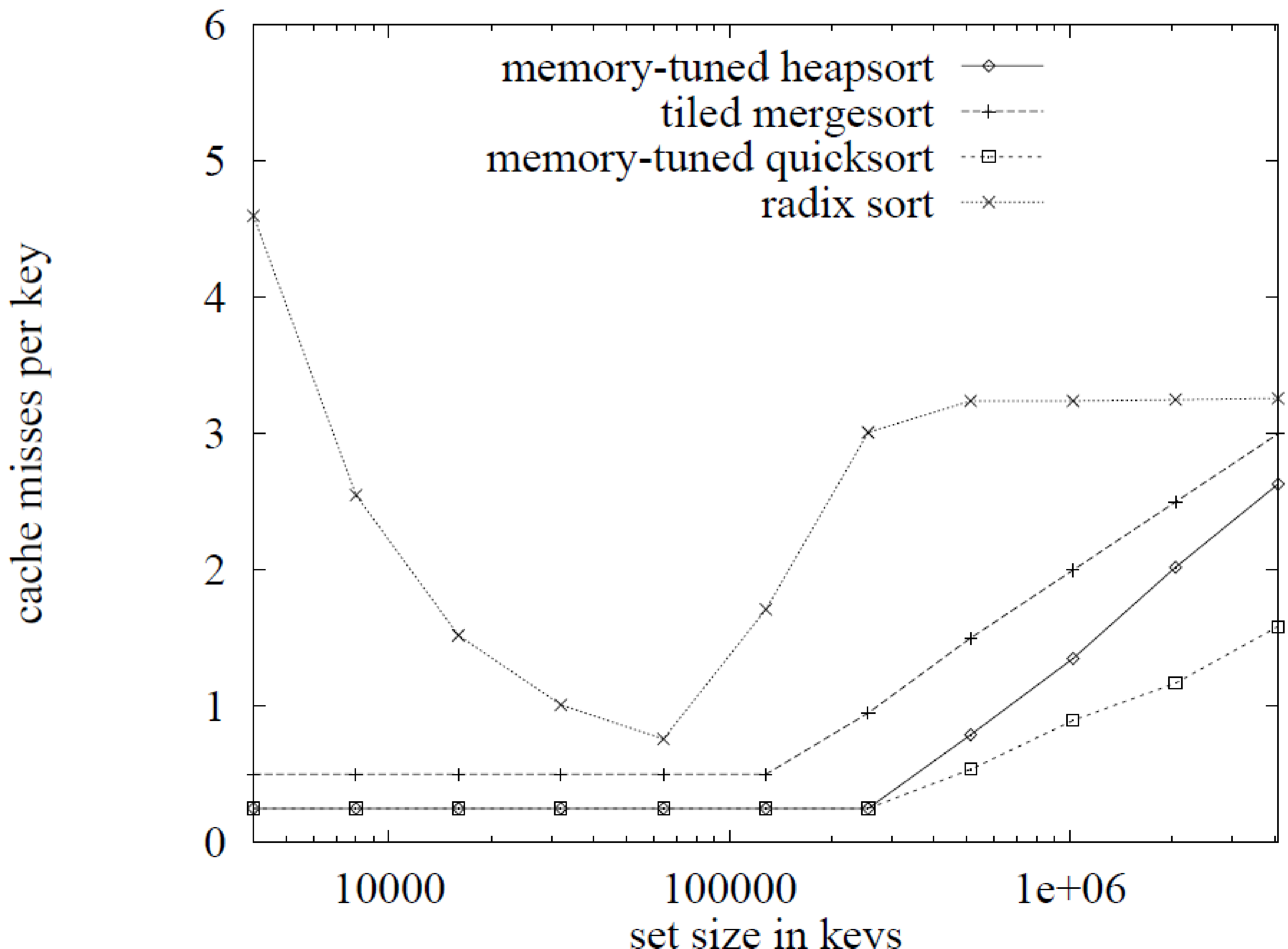
LaMarca, Ladner ’97

Radix-Sort: Eksperimenter



LaMarca, Ladner '97

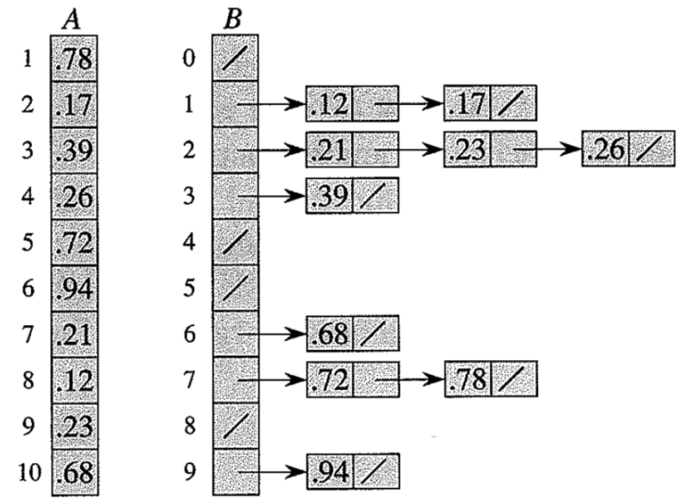
Radix-Sort: Eksperimenter



LaMarca, Ladner '97

Bucket-Sort:

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



BUCKET-SORT(A)

$n \leftarrow length[A]$
for $i \leftarrow 1$ **to** n
        do insert $A[i]$ into list $B[\lfloor nA[i] \rfloor]$
for $i \leftarrow 0$ **to** $n - 1$
        do sort list $B[i]$ with insertion sort
concatenate the lists $B[0], B[1], \ldots, B[n-1]$ together in order

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

Sortering – State-of-the-Art

Bedst kendte sorterings grænse for RAM modellen (med ubegrænset ordstørrelse) er en randomiseret algoritme der bruger $O(n)$ plads og har en forventet køretid på:

$$O(n\sqrt{\log\log n})$$

Om dette kan opnås uden randomisering er ukendt. $O(n \log\log n)$ er kendt. Kan man opnå $O(n)$ tid?

Yijie Han, Mikkel Thorup: Integer Sorting in $O(n\sqrt{\log\log n})$ Expected Time and Linear Space. Proc. 43rd Symposium on Foundations of Computer Science, 135-144, 2002.