Counting sort

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

Hvordan ser C ud efter vi har akkumuleret?

$A = [4, 3, 4, 5, 1, 1, 5, 1, 3]$, $k = 5$
Hvad er C i linje 9?

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

$[0, 3, 0, 2, 2, 2]$ A

$[0, 3, 3, 5, 7, 9]$ B

$[0, 0, 3, 3, 5, 7]$ C

socrative.com → Student login,
Room name: ABRAHAMSEN3464

$[0, 3, 3, 5, 9, 7]$ D

$[1, 3, 3, 5, 7, 9]$ E

Tidskompleksitet

$n = A.length$

```
COUNTING-SORT(A, B, k)
 1  let C[0..k] be a new array
 2  for i = 0 to k                                   ] Θ(k)
 3      C[i] = 0
 4  for j = 1 to A.length                            ] Θ(n)
 5      C[A[j]] = C[A[j]] + 1
 6  // C[i] now contains the number of elements equal to i.
 7  for i = 1 to k                                   ] Θ(k)
 8      C[i] = C[i] + C[i − 1]
 9  // C[i] now contains the number of elements less than or equal to i.
10  for j = A.length downto 1                        ] Θ(n)
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

I alt: $\Theta(n + k)$.

Tidskompleksitet

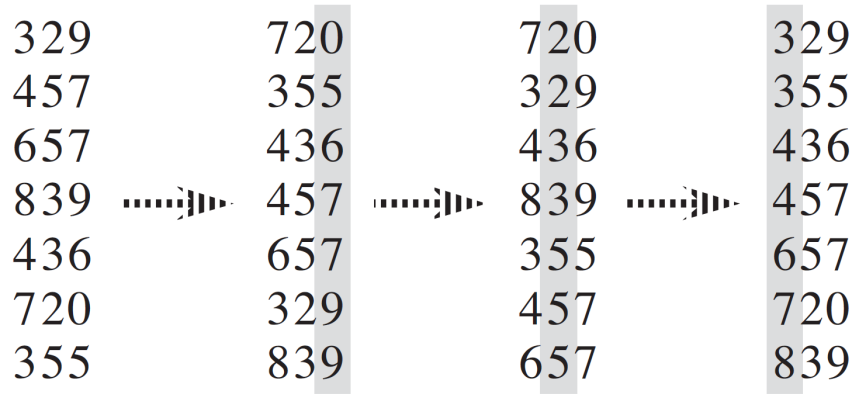
$n = A.length$

```
COUNTING-SORT(A, B, k)
 1  let C[0..k] be a new array
 2  for i = 0 to k                                   ] Θ(k)
 3      C[i] = 0
 4  for j = 1 to A.length                            ] Θ(n)
 5      C[A[j]] = C[A[j]] + 1
 6  // C[i] now contains the number of elements equal to i.
 7  for i = 1 to k                                   ] Θ(k)
 8      C[i] = C[i] + C[i − 1]
 9  // C[i] now contains the number of elements less than or equal to i.
10  for j = A.length downto 1                        ] Θ(n)
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

I alt: $\Theta(n + k)$.

Hvis $k = O(n)$: Sortering i $\Theta(n)$ tid!

Radix sort



RADIX-SORT(A, d)

1 **for** $i = 1$ **to** d

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

Radix sort

1001
1101
0100
1110
0101
0011

Hvad er rækkefølgende efter sortering af de to sidste cifre?

RADIX-SORT(A, d)

1 **for** $i = 1$ **to** d

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

A
0100
1110
1001
1101
0101
0011

B
1001
1101
0101
1110
0011
0100

C
1001
0011
0100
1101
0101
1110

D
0100
1001
1101
0101
1110
0011

E
1001
1101
0100
1110
0101
0011

socrative.com → Student login,
Room name: ABRAHAMSEN3464

Eksempel

Eksempel: Antag at vi vil sortere n binære tal fra mængden $\{0, 1, \ldots, n^4\}$.

Eksempel

Eksempel: Antag at vi vil sortere n binære tal fra mængden $\{0, 1, \ldots, n^4\}$.

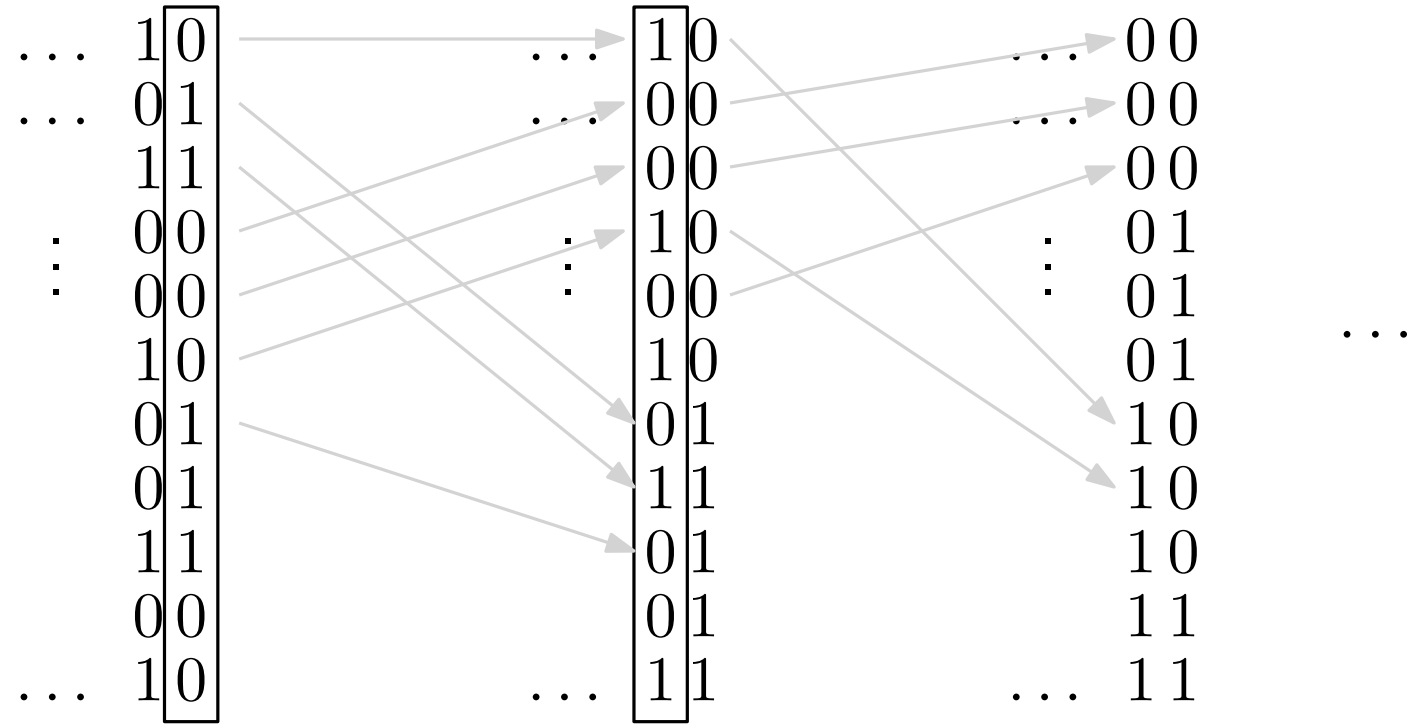
Forsøg 1: Vi bruger counting sort direkte, med $k = n^4$.
Tid: $\Theta(n + k) = \Theta(n^4)$.

Eksempel

Eksempel: Antag at vi vil sortere n binære tal fra mængden $\{0, 1, \ldots, n^4\}$.

Forsøg 1: Vi bruger counting sort direkte, med $k = n^4$.
Tid: $\Theta(n + k) = \Theta(n^4)$.

Forsøg 2: Vi bruger radix sort på hver bit.

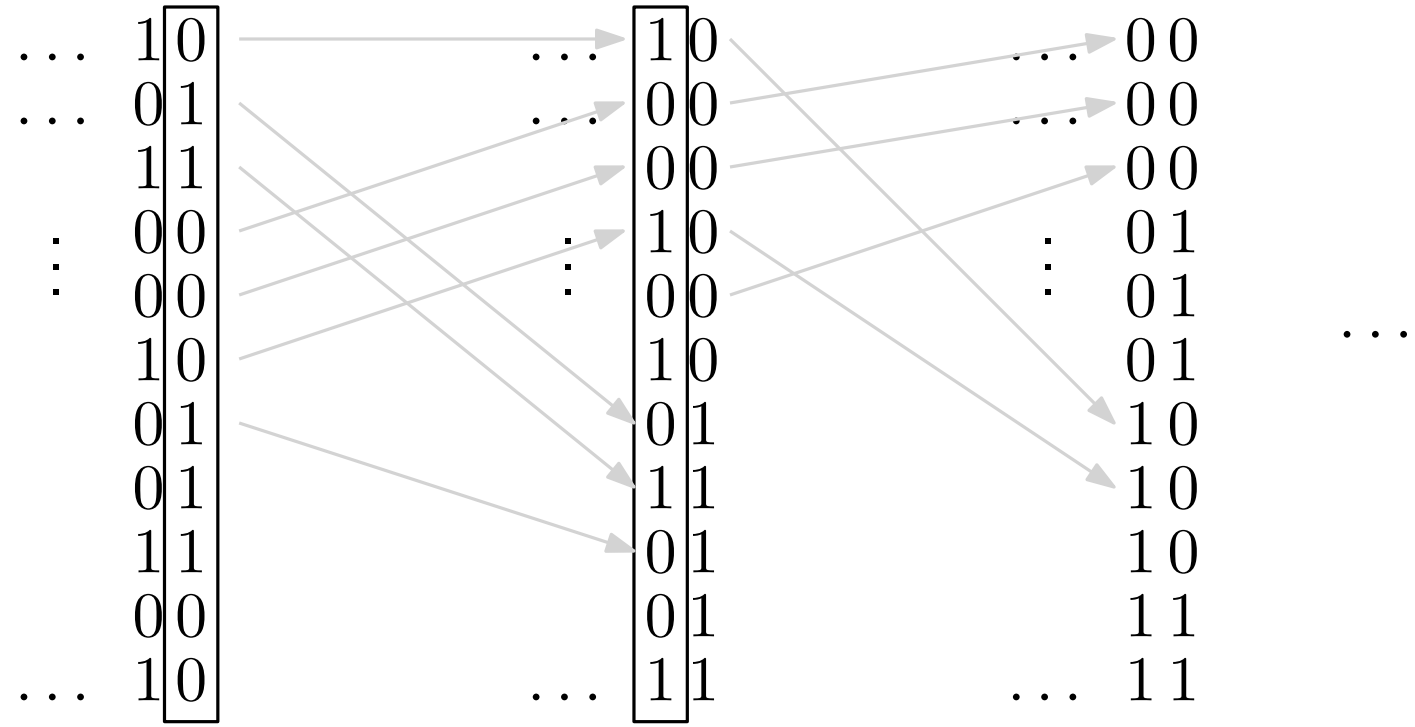


Eksempel

Eksempel: Antag at vi vil sortere n binære tal fra mængden $\{0, 1, \ldots, n^4\}$.

Forsøg 1: Vi bruger counting sort direkte, med $k = n^4$.
Tid: $\Theta(n + k) = \Theta(n^4)$.

Forsøg 2: Vi bruger radix sort på hver bit.



#bits $\approx \lg(n^4) = 4 \lg n = \Theta(\log n)$. Tid: $\Theta((n + 2) \log n) = \Theta(n \log n)$.

Korrekt formel: #bits i $x = \lceil \lg(x + 1) \rceil$.

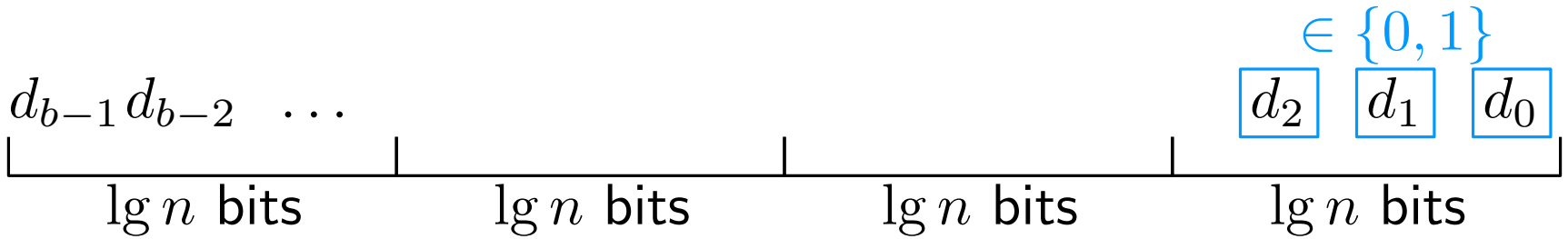
Eksempel, fortsat

Eksempel: Antag at vi vil sortere n binære tal fra mængden $\{0, 1, \ldots, n^4\}$.

Forsøg 1: Vi bruger counting sort direkte, med $k = n^4$.
Tid: $\Theta(n + k) = \Theta(n^4)$.

Forsøg 2: Vi bruger radix sort på hver bit, dvs. $k = 2$. #bits $\approx 4 \lg n$.
Tid: $\Theta(n \log n)$.

Forsøg 3: Vi inddeler bits i blokke af $\lg n$ bits og bruger radix sort på hver blok.



Antal muligheder for hver blok: $2^{\lg n} = n$. Dvs. $k = n$.

Tid: $O((n + k) \cdot 4) = O(n)$.

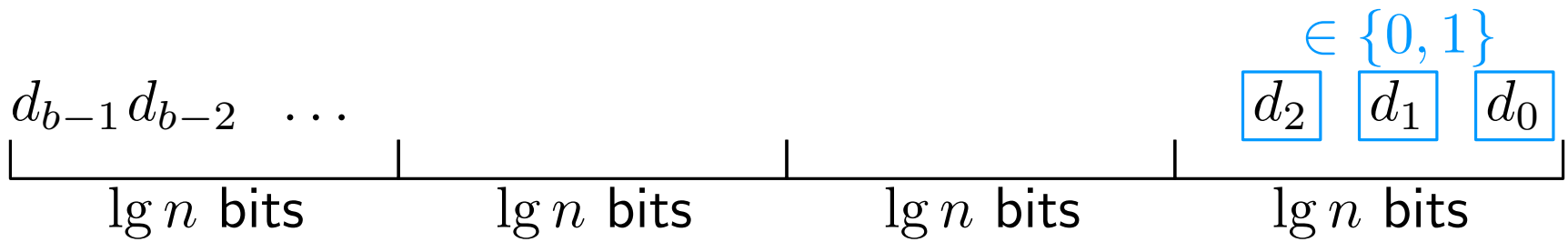
Eksempel, fortsat

Eksempel: Antag at vi vil sortere n binære tal fra mængden $\{0, 1, \ldots, n^4\}$.

Forsøg 1: Vi bruger counting sort direkte, med $k = n^4$.
Tid: $\Theta(n + k) = \Theta(n^4)$.

Forsøg 2: Vi bruger radix sort på hver bit, dvs. $k = 2$. #bits $\approx 4 \lg n$.
Tid: $\Theta(n \log n)$.

Forsøg 3: Vi inddeler bits i blokke af $\lg n$ bits og bruger radix sort på hver blok.



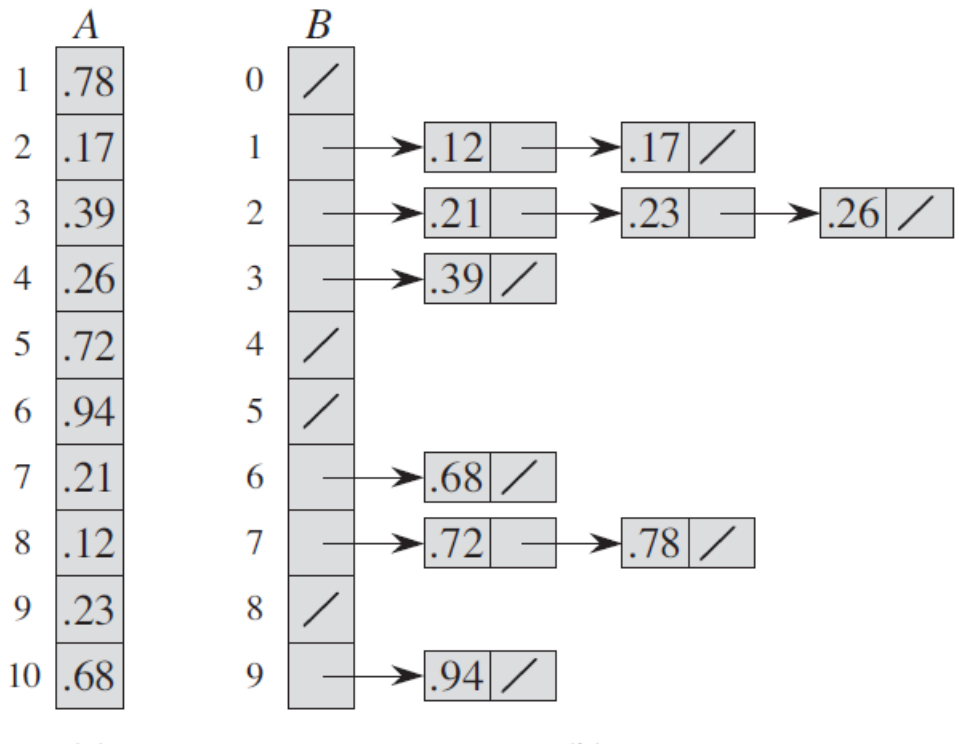
Antal muligheder for hver blok: $2^{\lg n} = n$. Dvs. $k = n$.

Tid: $O((n + k) \cdot 4) = O(n)$.

Lemma 8.4

Given n b-bit numbers and any positive integer $r \leq b$, RADIX-SORT correctly sorts these numbers in $\Theta((b/r)(n + 2^r))$ time if the stable sort it uses takes $\Theta(n + k)$ time for inputs in the range 0 to k.

Bucket sort



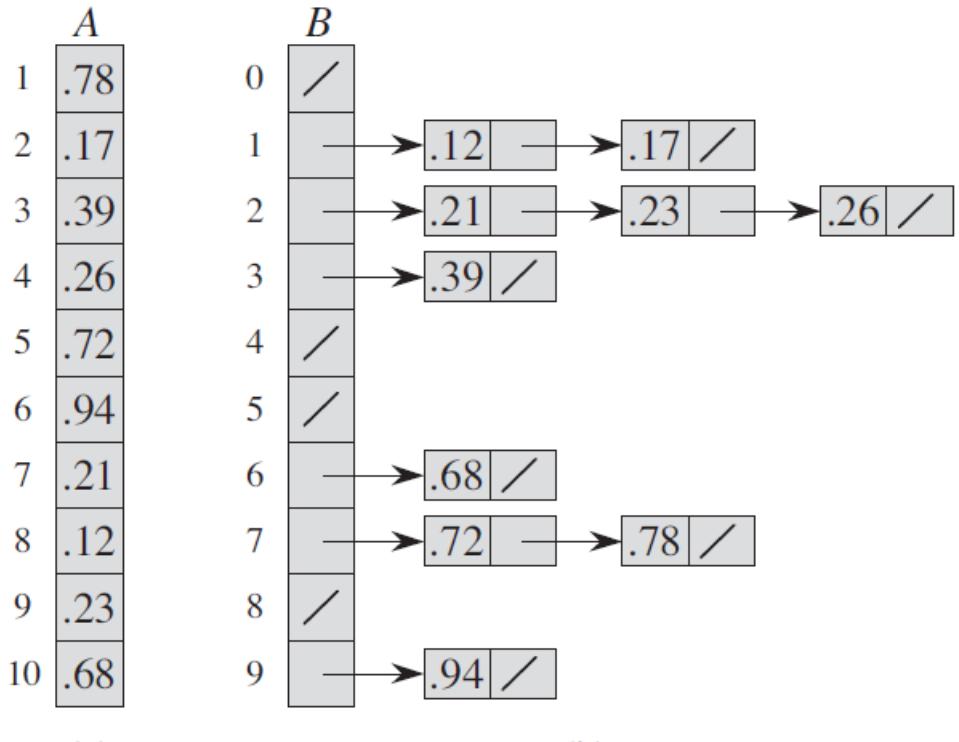
BUCKET-SORT(A)

1 let $B[0..n-1]$ be a new array
2 $n = A.length$
3 **for** $i = 0$ **to** $n-1$
4 make $B[i]$ an empty list
5 **for** $i = 1$ **to** n
6 insert $A[i]$ into list $B[\lfloor n\,A[i] \rfloor]$
7 **for** $i = 0$ **to** $n-1$
8 sort list $B[i]$ with insertion sort
9 concatenate the lists $B[0], B[1], \ldots, B[n-1]$ together in order

Bucket sort

Køretid:

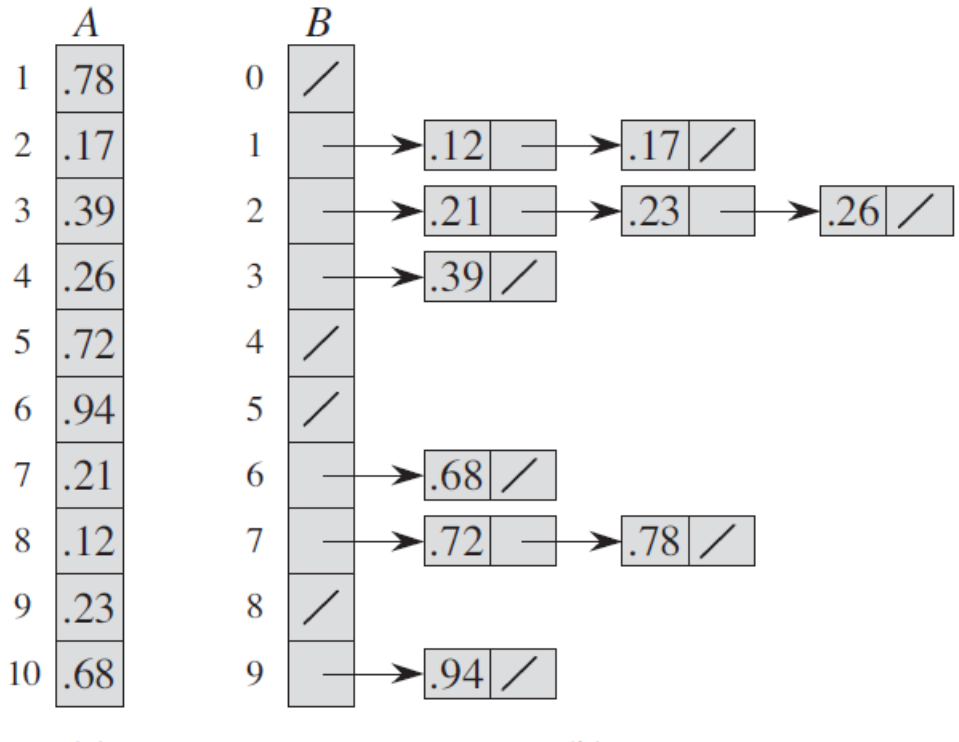
$\Theta(n) + \Theta(n_0^2 + n_1^2 + \ldots + n_{n-1}^2)$



BUCKET-SORT(A)

1 let $B[0..n-1]$ be a new array
2 $n = A.length$
3 **for** $i = 0$ **to** $n-1$
4 make $B[i]$ an empty list
5 **for** $i = 1$ **to** n
6 insert $A[i]$ into list $B[\lfloor n\,A[i] \rfloor]$
7 **for** $i = 0$ **to** $n-1$
8 sort list $B[i]$ with insertion sort
9 concatenate the lists $B[0], B[1], \ldots, B[n-1]$ together in order

Bucket sort



Køretid:

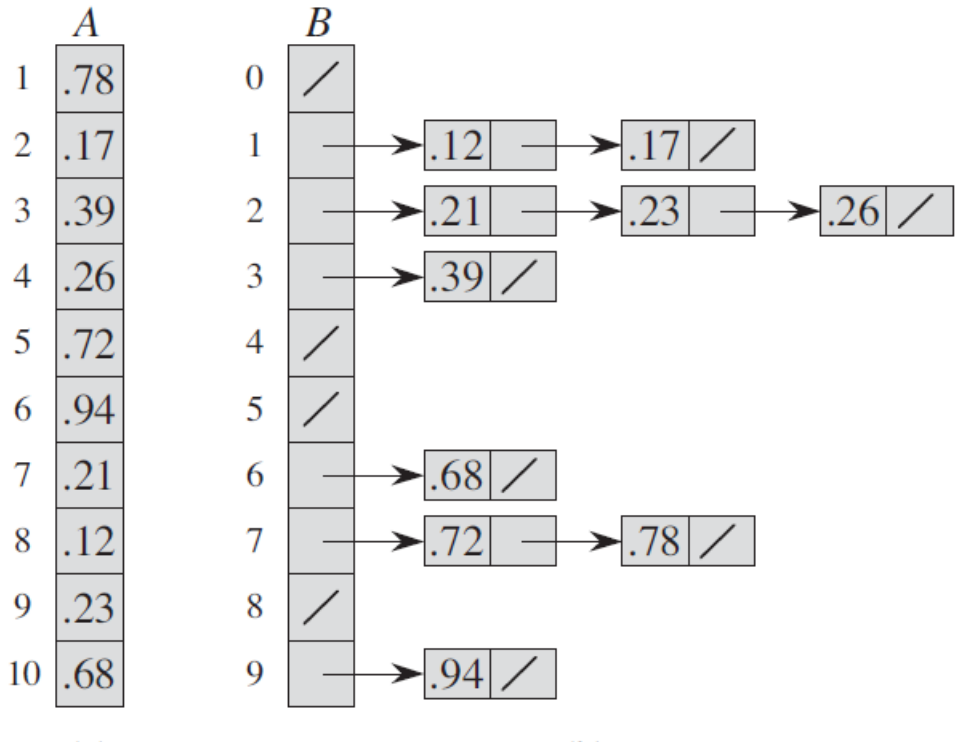
$\Theta(n) + \Theta(n_0^2 + n_1^2 + \ldots + n_{n-1}^2)$

Værste fald: $\Theta(n^2)$

BUCKET-SORT(A)

```
1  let B[0..n − 1] be a new array
2  n = A.length
3  for i = 0 to n − 1
4      make B[i] an empty list
5  for i = 1 to n
6      insert A[i] into list B[⌊n A[i]⌋]
7  for i = 0 to n − 1
8      sort list B[i] with insertion sort
9  concatenate the lists B[0], B[1], . . . , B[n − 1] together in order
```

Bucket sort



Køretid:
$\Theta(n) + \Theta(n_0^2 + n_1^2 + \ldots + n_{n-1}^2)$

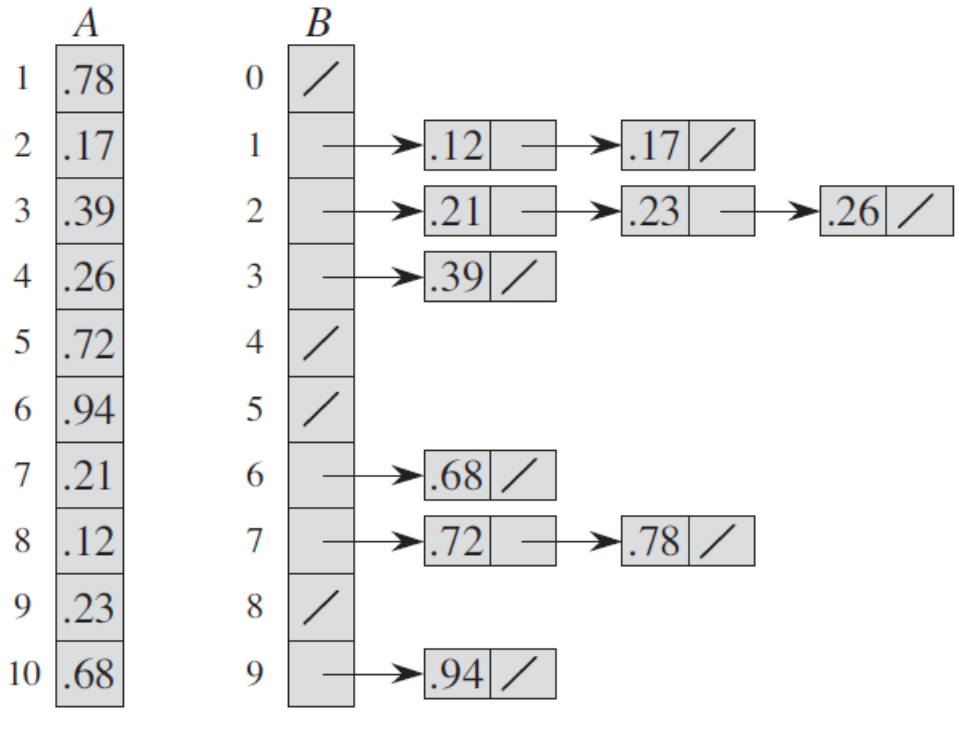
Værste fald: $\Theta(n^2)$

Tal i A uafhængige og uniformt tilfældige i $[0, 1)$:
$\mathrm{E}[n_i^2] = 2 - 1/n$

BUCKET-SORT(A)

```
1  let B[0..n − 1] be a new array
2  n = A.length
3  for i = 0 to n − 1
4      make B[i] an empty list
5  for i = 1 to n
6      insert A[i] into list B[⌊n A[i]⌋]
7  for i = 0 to n − 1
8      sort list B[i] with insertion sort
9  concatenate the lists B[0], B[1], . . . , B[n − 1] together in order
```

Bucket sort



Køretid:
$\Theta(n) + \Theta(n_0^2 + n_1^2 + \ldots + n_{n-1}^2)$

Værste fald: $\Theta(n^2)$

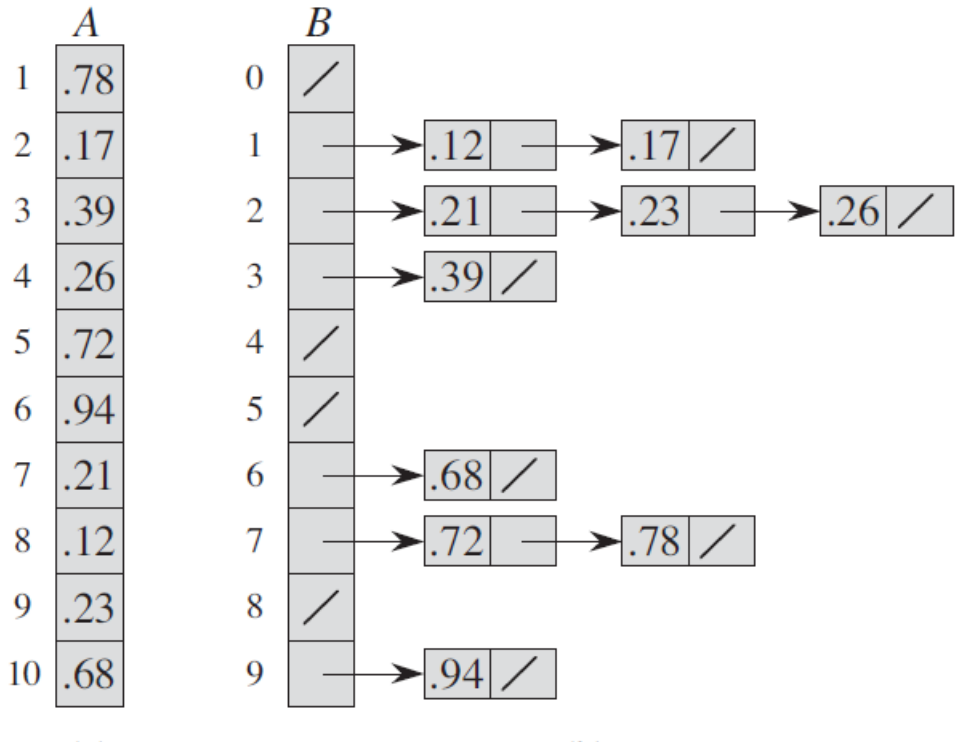
Tal i A uafhængige og uniformt tilfældige i $[0, 1)$:
$\mathrm{E}[n_i^2] = 2 - 1/n$

Forventet køretid:
$\Theta(n) + \Theta(n \cdot (2 - 1/n)) = \Theta(n)$.

BUCKET-SORT(A)

```
1  let B[0..n − 1] be a new array
2  n = A.length
3  for i = 0 to n − 1
4       make B[i] an empty list
5  for i = 1 to n
6       insert A[i] into list B[⌊n A[i]⌋]
7  for i = 0 to n − 1
8       sort list B[i] with insertion sort
9  concatenate the lists B[0], B[1], . . . , B[n − 1] together in order
```

Bucket sort



BUCKET-SORT(A)

1 let $B[0..n-1]$ be a new array
2 $n = A.length$
3 **for** $i = 0$ **to** $n-1$
4 make $B[i]$ an empty list
5 **for** $i = 1$ **to** n
6 insert $A[i]$ into list $B[\lfloor n\,A[i] \rfloor]$
7 **for** $i = 0$ **to** $n-1$
8 sort list $B[i]$ with insertion sort
9 concatenate the lists $B[0], B[1], \ldots, B[n-1]$ together in order

Køretid:
$\Theta(n) + \Theta(n_0^2 + n_1^2 + \ldots + n_{n-1}^2)$

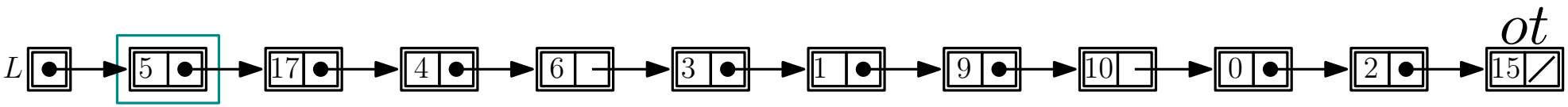
Værste fald: $\Theta(n^2)$

Tal i A uafhængige og uniformt tilfældige i $[0, 1)$:
$\mathrm{E}[n_i^2] = 2 - 1/n$

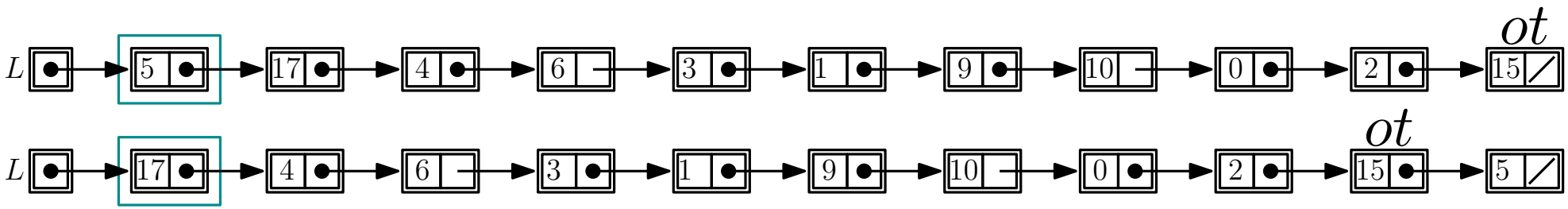
Forventet køretid:
$\Theta(n) + \Theta(n \cdot (2 - 1/n)) = \Theta(n)$.

Hvorfor insertion sort?
Hvorfor hægtede lister?

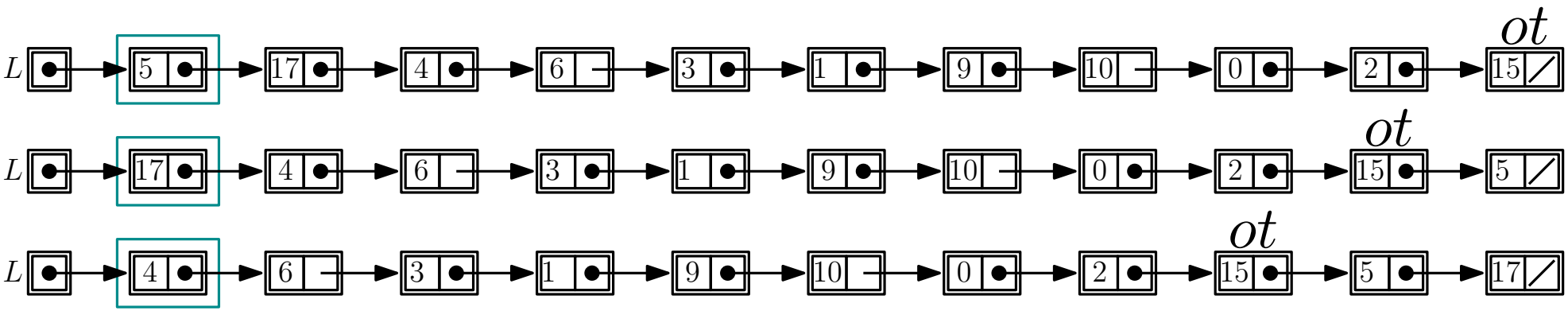
Insertion sort af enkelthægtet liste



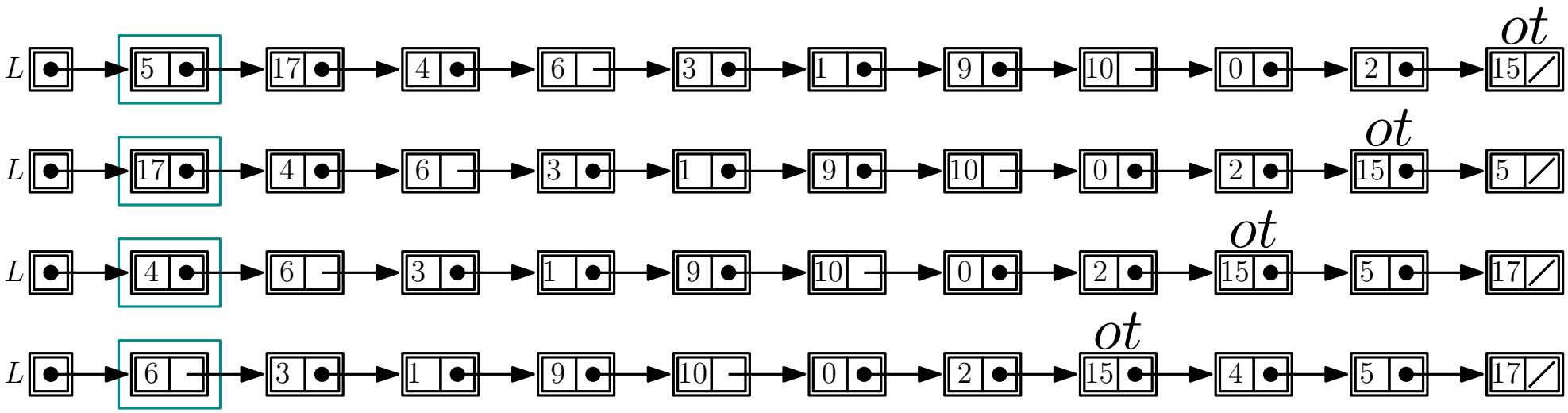
Insertion sort af enkelthægtet liste



Insertion sort af enkelthægtet liste



Insertion sort af enkelthægtet liste



Insertion sort af enkelthægtet liste