Counting sort

```
COUNTING-SORT(A, B, k)
let C[0..k] be a new array
for i = 0 to k
        C[i] = 0
for j = 1 to A.length
        C[A[j]] = C[A[j]] + 1
// C[i] now contains the number of elements equal to i.
for i = 1 to k
        C[i] = C[i] + C[i − 1]
// C[i] now contains the number of elements less than or equal to i.
for j = A.length downto 1
        B[C[A[j]]] = A[j]
        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 $\rightarrow$ 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)
let C[0..k] be a new array
for i = 0 to k                                  Θ(k)
    C[i] = 0
for j = 1 to A.length                           Θ(n)
    C[A[j]] = C[A[j]] + 1
// C[i] now contains the number of elements equal to i.
for i = 1 to k                                  Θ(k)
    C[i] = C[i] + C[i − 1]
// C[i] now contains the number of elements less than or equal to i.
for j = A.length downto 1                       Θ(n)
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] − 1
```

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

Tidskompleksitet

$n = A.length$

```
COUNTING-SORT(A, B, k)
let C[0..k] be a new array
for i = 0 to k                                          ⎤ Θ(k)
    C[i] = 0                                            ⎦
for j = 1 to A.length                                   ⎤ Θ(n)
    C[A[j]] = C[A[j]] + 1                               ⎦
// C[i] now contains the number of elements equal to i.
for i = 1 to k                                          ⎤ Θ(k)
    C[i] = C[i] + C[i − 1]                              ⎦
// C[i] now contains the number of elements less than or equal to i.
for j = A.length downto 1                               ⎤
    B[C[A[j]]] = A[j]                                   ⎥ Θ(n)
    C[A[j]] = C[A[j]] − 1                               ⎦
```

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

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

Radix sort

329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

RADIX-SORT(A, d)

```
for i = 1 to d
    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

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal kan repræsenteres med $b = 4r$ bits.

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal kan repræsenteres med $b = 4r$ bits.
Bevis for påstand: Det største tal man kan skrive med b bits er
$N = (\underbrace{111\ldots1}_{b \text{ bits}})_2 = 1 + 2 + 4 + \ldots + 2^{b-1}$.

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal kan repræsenteres med $b = 4r$ bits.
Bevis for påstand: Det største tal man kan skrive med b bits er

$$N = (\underbrace{111\ldots1}_{b \text{ bits}})_2 = 1 + 2 + 4 + \ldots + 2^{b-1}.$$

Vi har

$$1 + N = \underbrace{\underbrace{\underbrace{\underbrace{1 + 1}_{2} + 2}_{4} + 4}_{8} + \ldots + 2^{b-1}}_{2^b} = 2^b.$$

Derfor er $N = 2^b - 1 = 2^{4r} - 1 = n^4 - 1$.

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal er repræsenteret med $b = 4r$ bits.

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

$\boxed{\textbf{Påstand: } \text{Hvert tal er repræsenteret med } b = 4r \text{ bits.}}$

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.

$$\begin{array}{cc}
\ldots & 1\,0 \\
\ldots & 0\,1 \\
 & 1\,1 \\
\vdots & 0\,0 \\
 & 0\,0 \\
 & 1\,0 \\
 & 0\,1 \\
 & 0\,1 \\
 & 1\,1 \\
 & 0\,0 \\
\ldots & 1\,0
\end{array}$$

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal er repræsenteret med $b = 4r$ bits.

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.

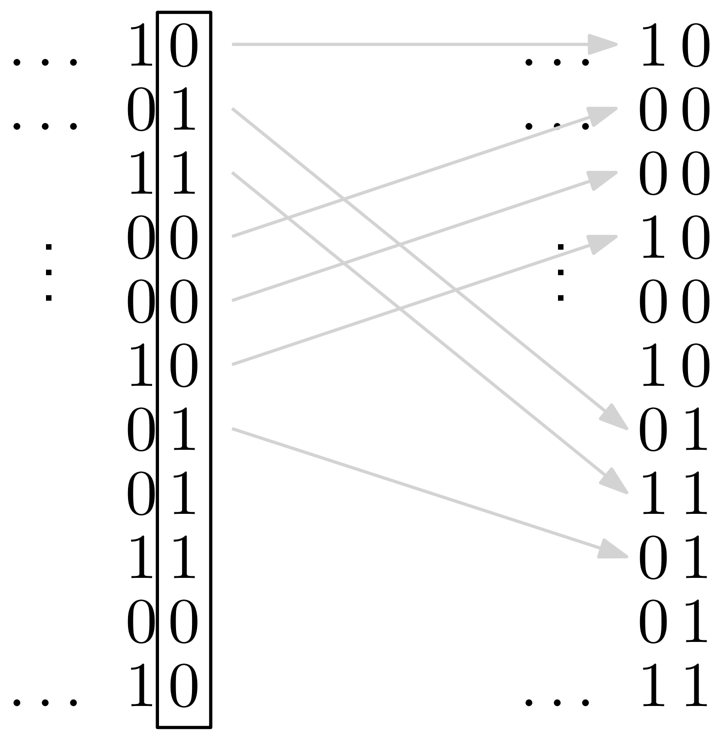
$$
\begin{array}{ccc}
\ldots & 1 & \boxed{0} \\
\ldots & 0 & 1 \\
 & 1 & 1 \\
\vdots & 0 & 0 \\
 & 0 & 0 \\
 & 1 & 0 \\
 & 0 & 1 \\
 & 0 & 1 \\
 & 1 & 1 \\
 & 0 & 0 \\
\ldots & 1 & 0
\end{array}
$$

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \dots, n^4 - 1\}$.

Påstand: Hvert tal er repræsenteret med $b = 4r$ bits.

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.

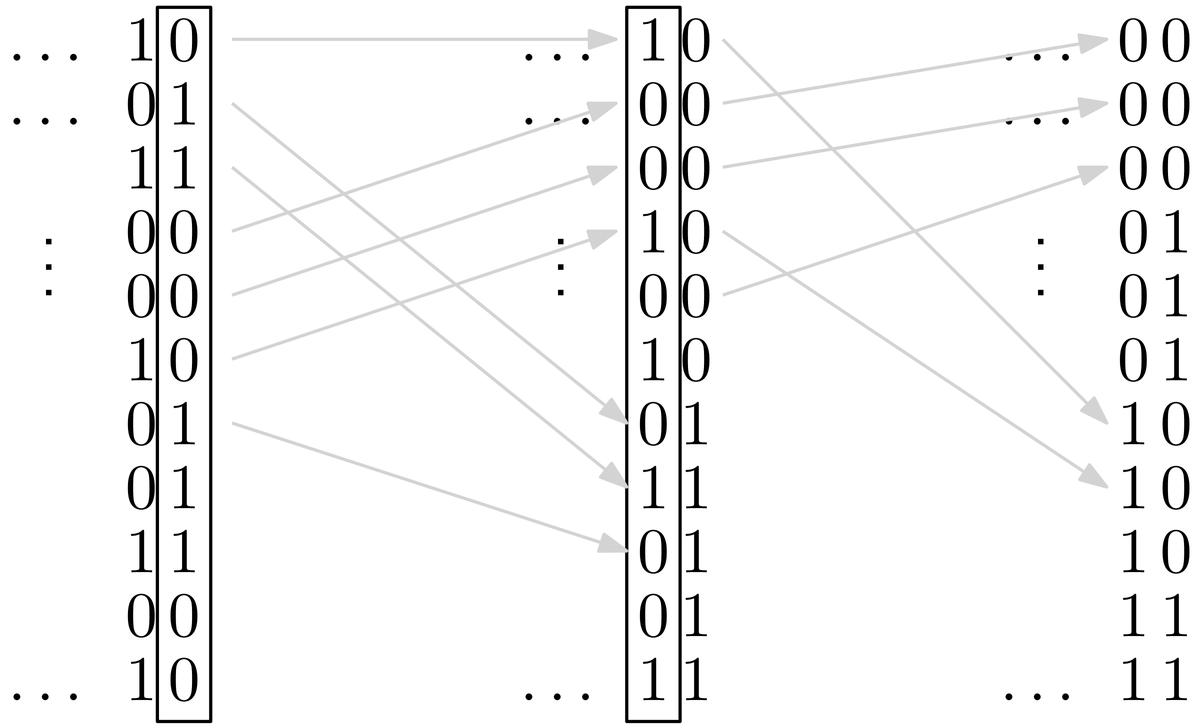


Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal er repræsenteret med $b = 4r$ bits.

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.

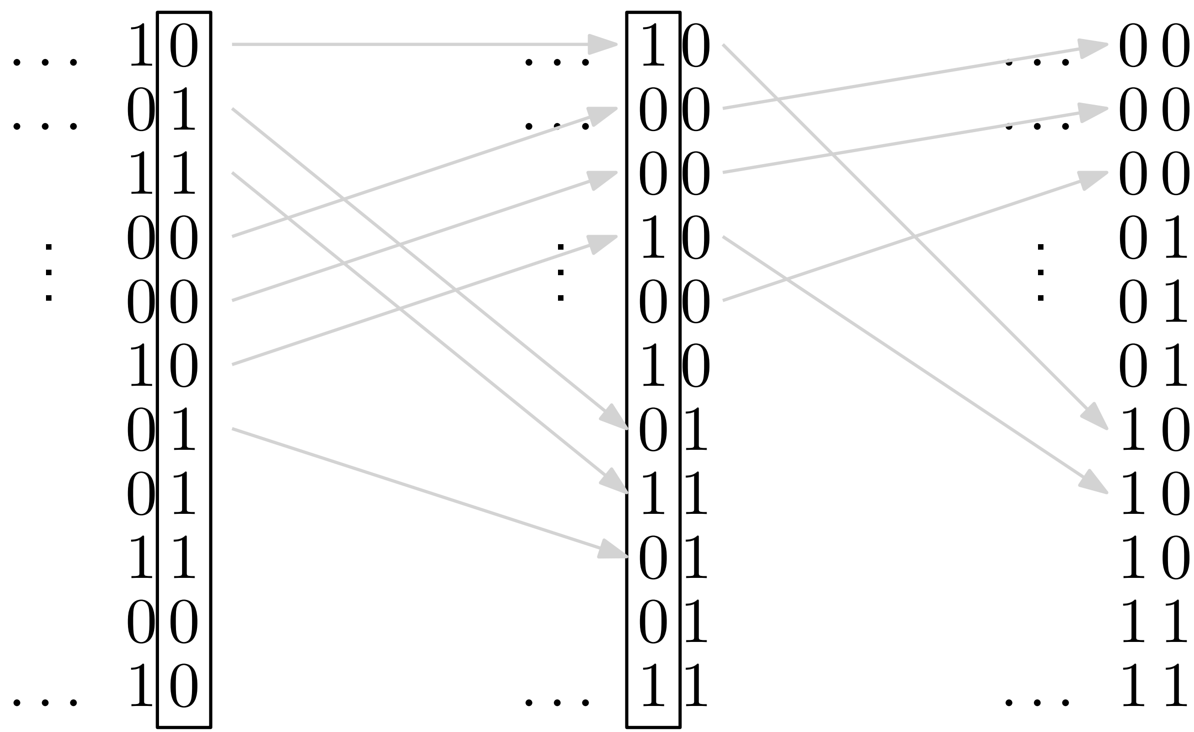


Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal er repræsenteret med $b = 4r$ bits.

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.



Tid: Laver b gange counting sort, hvor $k = 2$. I alt $\Theta(b(n+k)) = \Theta(4r(n+2)) = \Theta(n \log n)$.

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

$\boxed{\textbf{Påstand: } \text{Hvert tal er repræsenteret med } b = 4r \text{ bits.}}$

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.
Tid: $O(n \log n)$.

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal er repræsenteret med $b = 4r$ bits.

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.
Tid: $O(n \log n)$.
2. forsøg: Inddel bits i 4 grupper af r bits. Betragt hver gruppe som ét ciffer.

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal er repræsenteret med $b = 4r$ bits.

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.
Tid: $O(n \log n)$.
2. forsøg: Inddel bits i 4 grupper af r bits. Betragt hver gruppe som ét ciffer.

$$d_{b-1} d_{b-2} \quad \ldots \qquad \boxed{d_2} \; \boxed{d_1} \; \boxed{d_0}$$

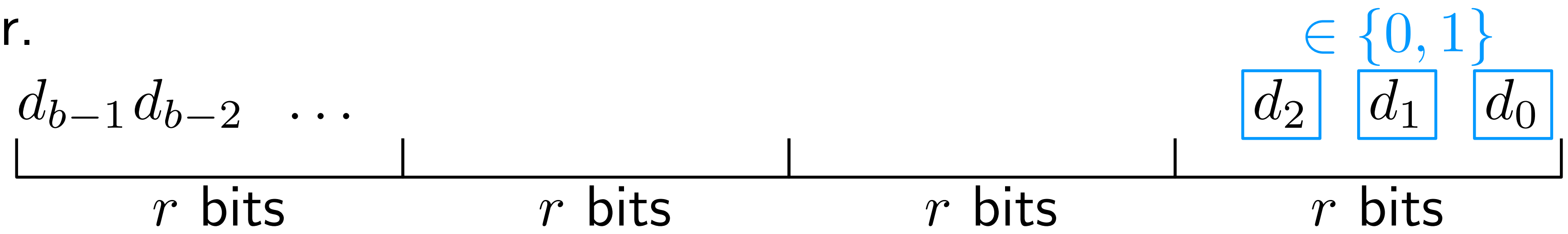
$\in \{0, 1\}$

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal er repræsenteret med $b = 4r$ bits.

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.
Tid: $O(n \log n)$.
2. forsøg: Inddel bits i 4 grupper af r bits. Betragt hver gruppe som ét ciffer.

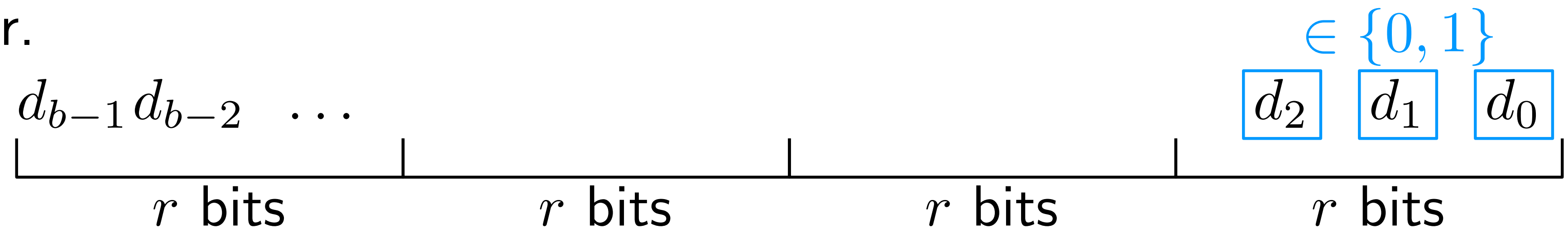


Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal er repræsenteret med $b = 4r$ bits.

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.
Tid: $O(n \log n)$.
2. forsøg: Inddel bits i 4 grupper af r bits. Betragt hver gruppe som ét ciffer.



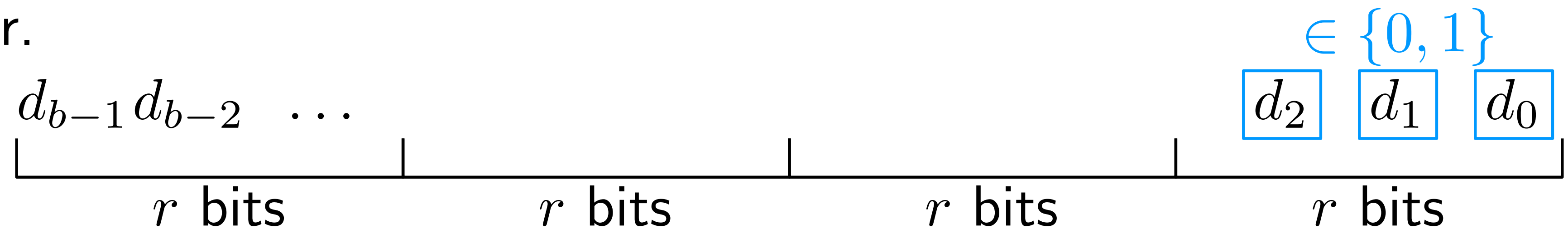
Antal muligheder for hvert af de 4 cifre: $2^r = n$.
Dvs. når vi skal bruge counting sort på ét ciffer er $k = n$.

Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal er repræsenteret med $b = 4r$ bits.

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.
Tid: $O(n \log n)$.
2. forsøg: Inddel bits i 4 grupper af r bits. Betragt hver gruppe som ét ciffer.



Antal muligheder for hvert af de 4 cifre: $2^r = n$.
Dvs. når vi skal bruge counting sort på ét ciffer er $k = n$.

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

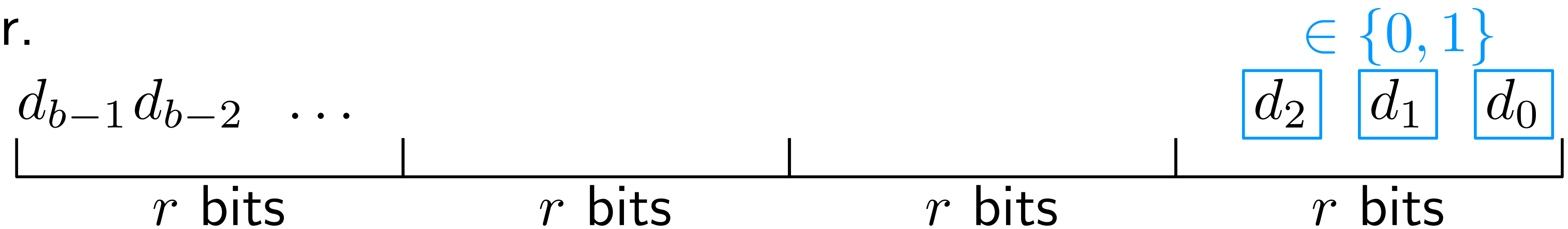
Radix sort

Eksempel: Antag at vi vil sortere $n = 2^r$ binære tal, $r \in \mathbb{N}$, fra mængden $\{0, 1, \ldots, n^4 - 1\}$.

Påstand: Hvert tal er repræsenteret med $b = 4r$ bits.

1. forsøg: Brug radix sort ved at sortere hver bit med counting sort.
Tid: $O(n \log n)$.

2. forsøg: Inddel bits i 4 grupper af r bits. Betragt hver gruppe som ét ciffer.



Antal muligheder for hvert af de 4 cifre: $2^r = n$.
Dvs. når vi skal bruge counting sort på ét ciffer er $k = n$.

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

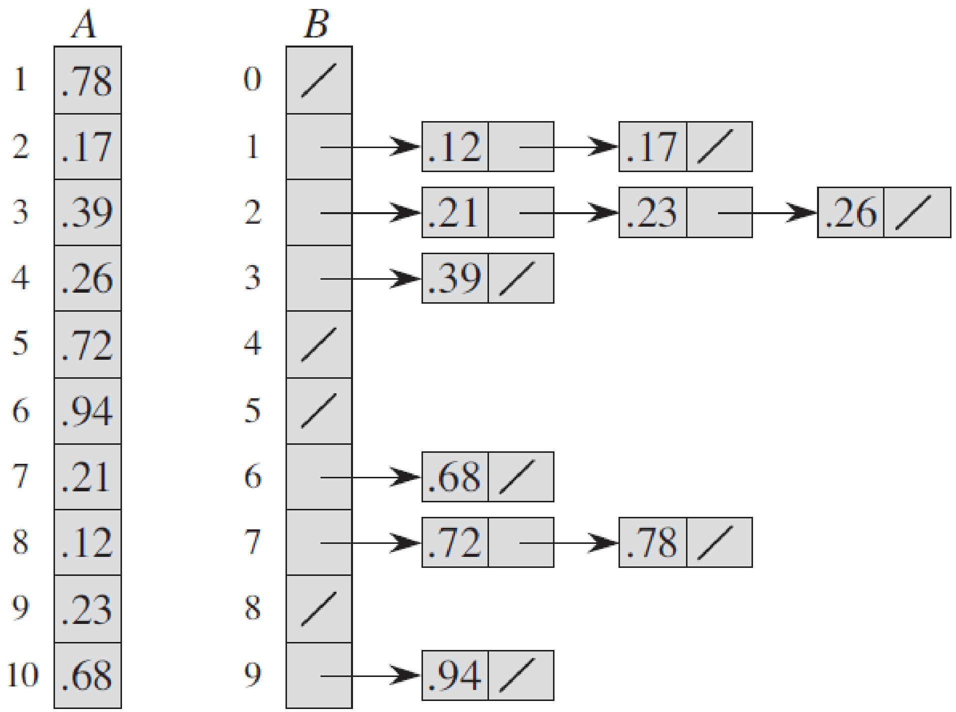
Lemma 8.4
Given n b-bit numbers and any positive integer $r \le b$, RADIX-SORT correctly sorts these numbers in $\Theta((b/r)(n + 2^r))$ time if the stable sort it uses takes $\Theta(n + k)$ time for inputs in the range 0 to k.

Radix sort

Lemma 8.4
Given n b-bit numbers and any positive integer $r \le b$, 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.

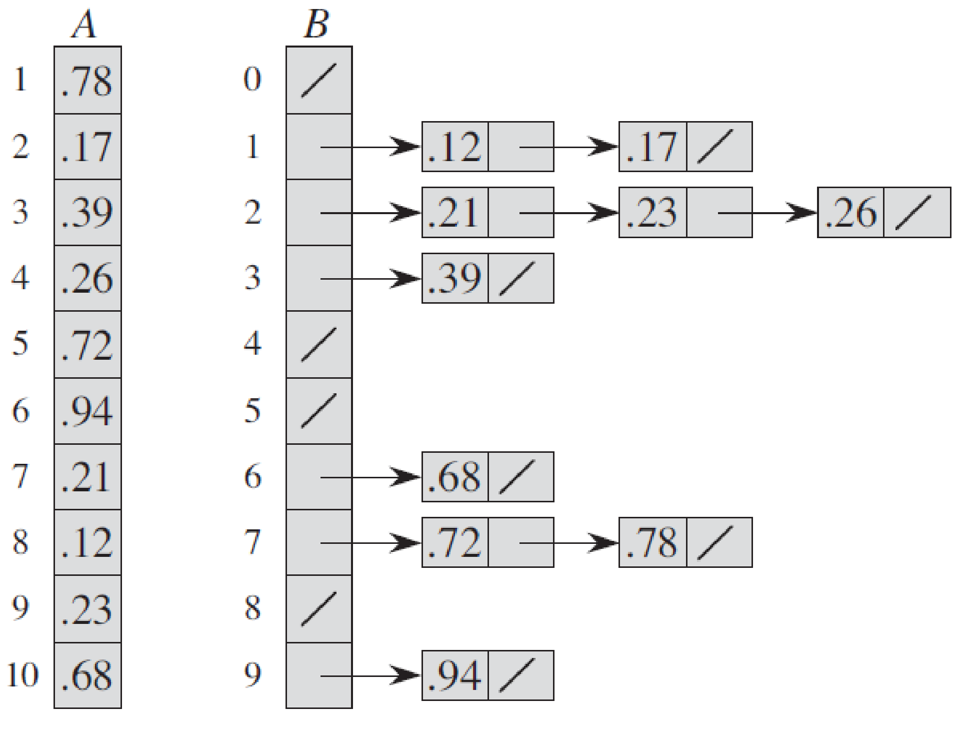
Sortere n tal fra $\{0, \ldots, n^p\}$, p konstant: Da er $b = p \lg n$.
Vælg $r = \lg n$, så $k = 2^r = 2^{\lg n} = n$. Køretid:
$\Theta((b/r)(n + 2^r)) = \Theta(p(n + 2^r)) = \Theta(pn) = \Theta(n)$.

Bucket sort




```
BUCKET-SORT(A)
let B[0 . . n − 1] be a new array
n = A.length
for i = 0 to n − 1
        make B[i] an empty list
for i = 1 to n
        insert A[i] into list B[⌊n A[i]⌋]
for i = 0 to n − 1
        sort list B[i] with insertion sort
concatenate the lists B[0], B[1], . . . , B[n − 1] together in order
```

Bucket sort



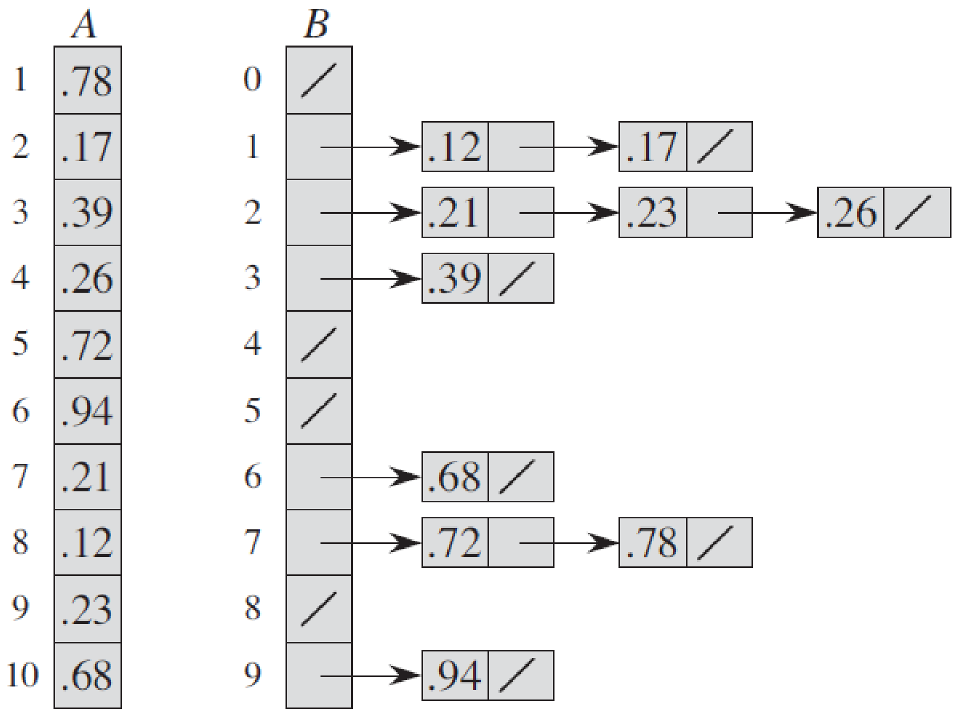
Køretid:

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

BUCKET-SORT(A)

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

Bucket sort



Køretid:

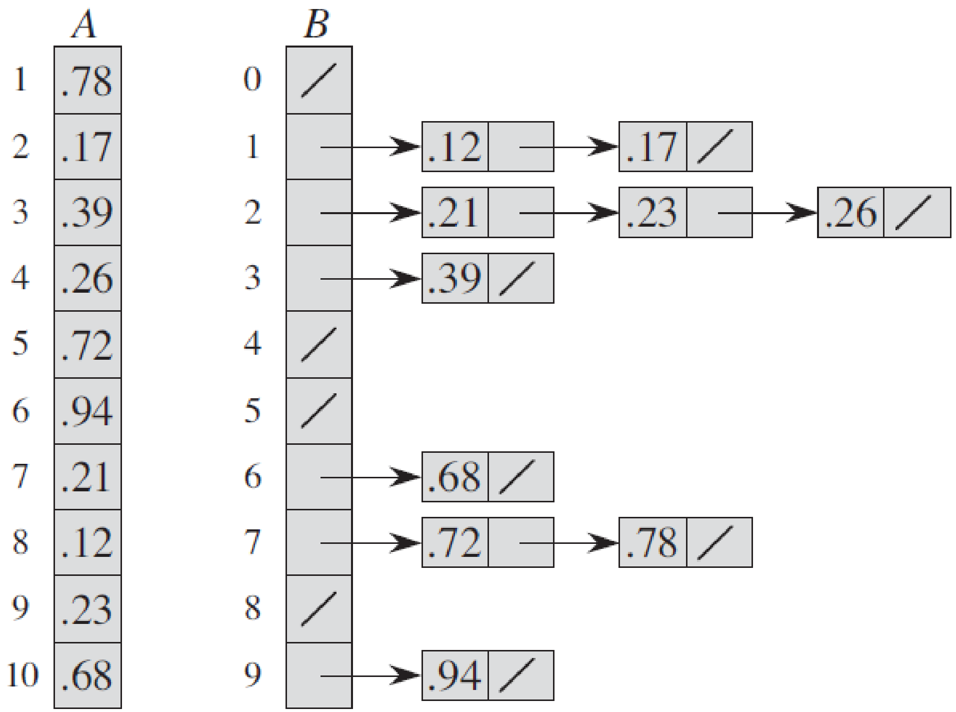
$$\Theta(n) + \Theta(n_0^2 + n_1^2 + \ldots + n_{n-1}^2)$$

Værste fald: $\Theta(n^2)$

BUCKET-SORT(A)

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

Bucket sort



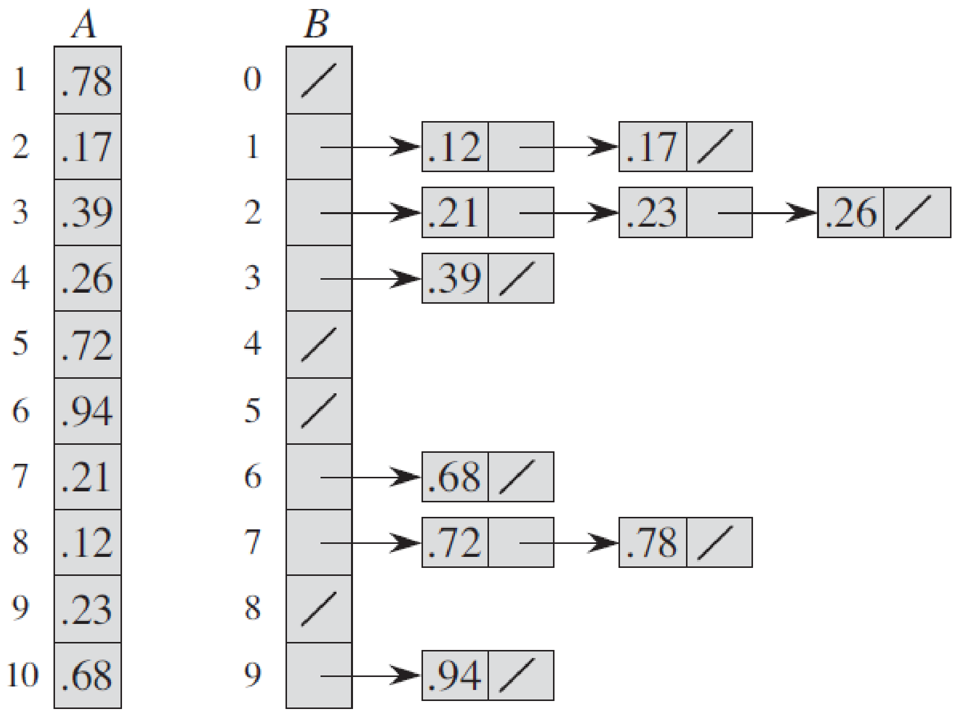
Køretid:
$\Theta(n) + \Theta(n_0^2 + n_1^2 + \ldots + n_{n-1}^2)$

Værste fald: $\Theta(n^2)$

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

```
BUCKET-SORT(A)
let B[0..n − 1] be a new array
n = A.length
for i = 0 to n − 1
    make B[i] an empty list
for i = 1 to n
    insert A[i] into list B[⌊n A[i]⌋]
for i = 0 to n − 1
    sort list B[i] with insertion sort
concatenate the lists B[0], B[1], ..., B[n − 1] together in order
```

Bucket sort



BUCKET-SORT(A)

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

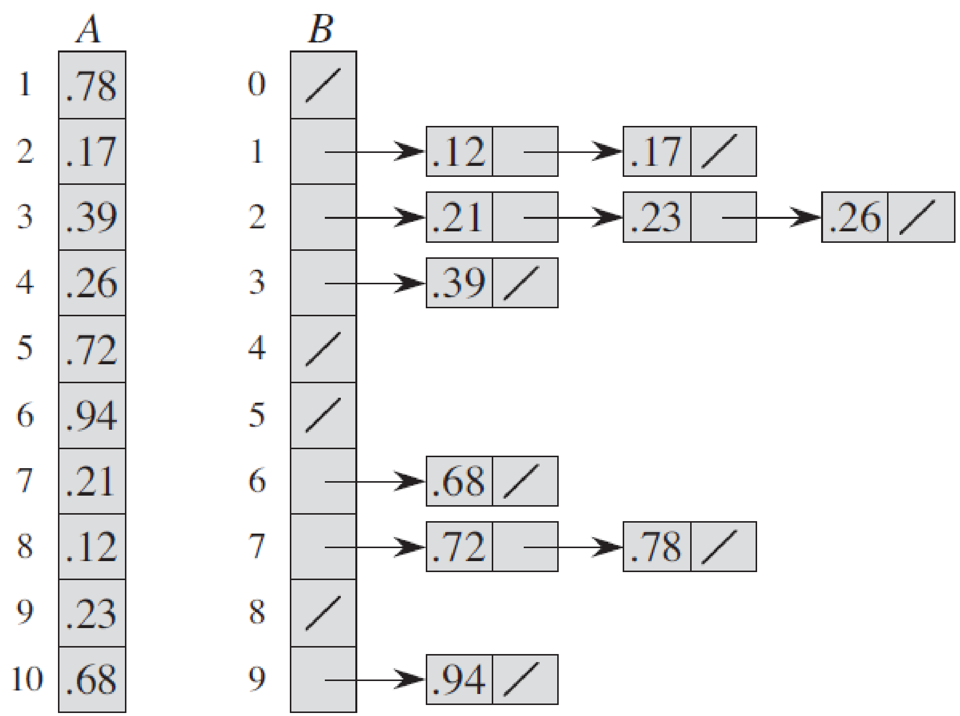
Køretid:
$\Theta(n) + \Theta(n_0^2 + n_1^2 + \ldots + n_{n-1}^2)$

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

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 $[0, 1)$:
$\mathrm{E}[n_i^2] = 2 - 1/n$

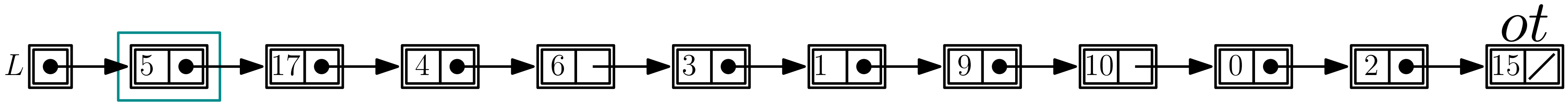
Forventet køretid:
$\Theta(n) + \Theta(n \cdot (2 - 1/n)) = \Theta(n)$.

BUCKET-SORT(A)

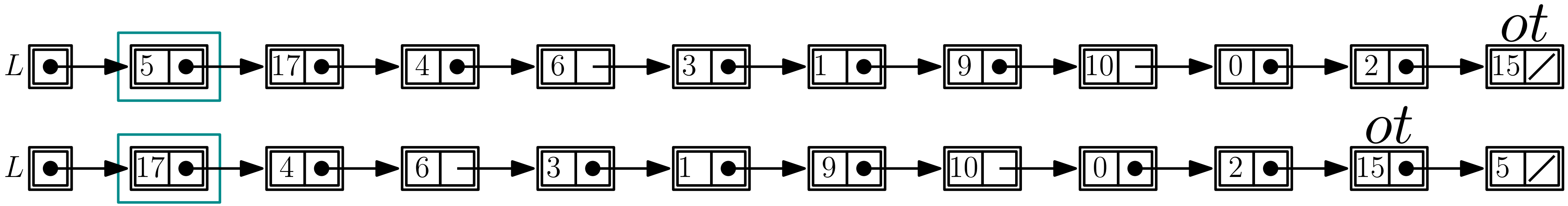
```
let B[0 .. n − 1] be a new array
n = A.length
for i = 0 to n − 1
    make B[i] an empty list
for i = 1 to n
    insert A[i] into list B[⌊n A[i]⌋]
for i = 0 to n − 1
    sort list B[i] with insertion sort
concatenate the lists B[0], B[1], ..., B[n − 1] together in order
```

Hvorfor insertion sort?
Hvorfor hægtede lister?

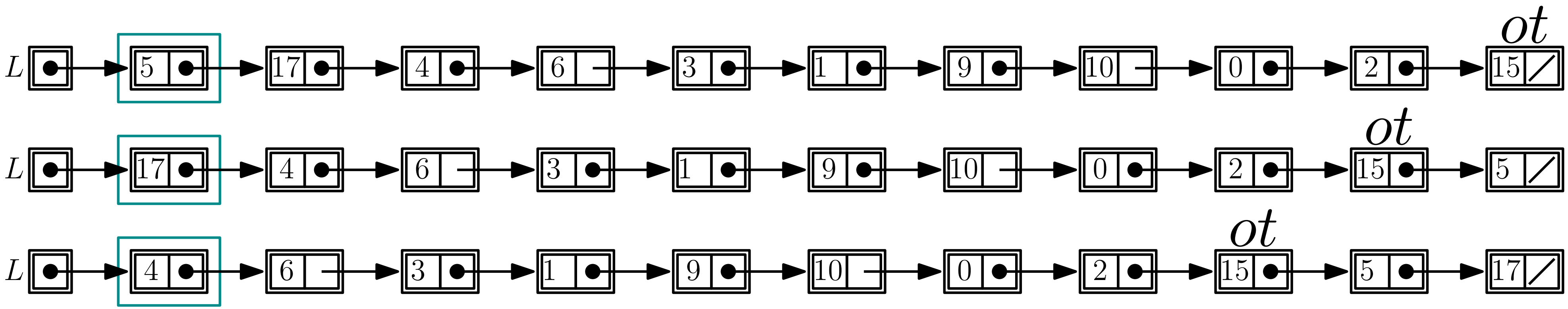
Insertion sort af enkelthægtet liste



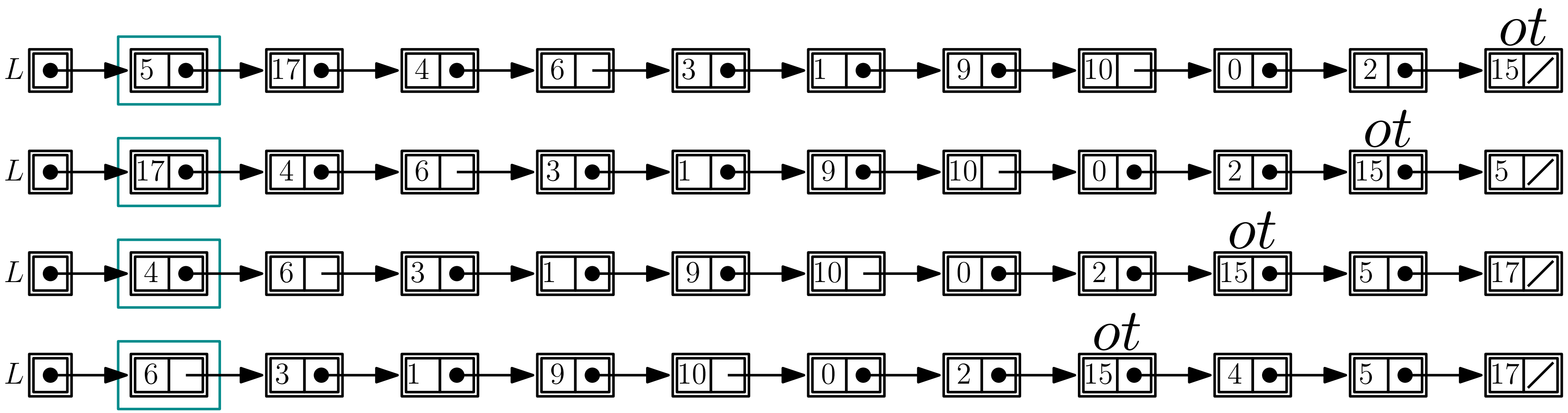
Insertion sort af enkelthægtet liste



Insertion sort af enkelthægtet liste



Insertion sort af enkelthægtet liste



Insertion sort af enkelthægtet liste