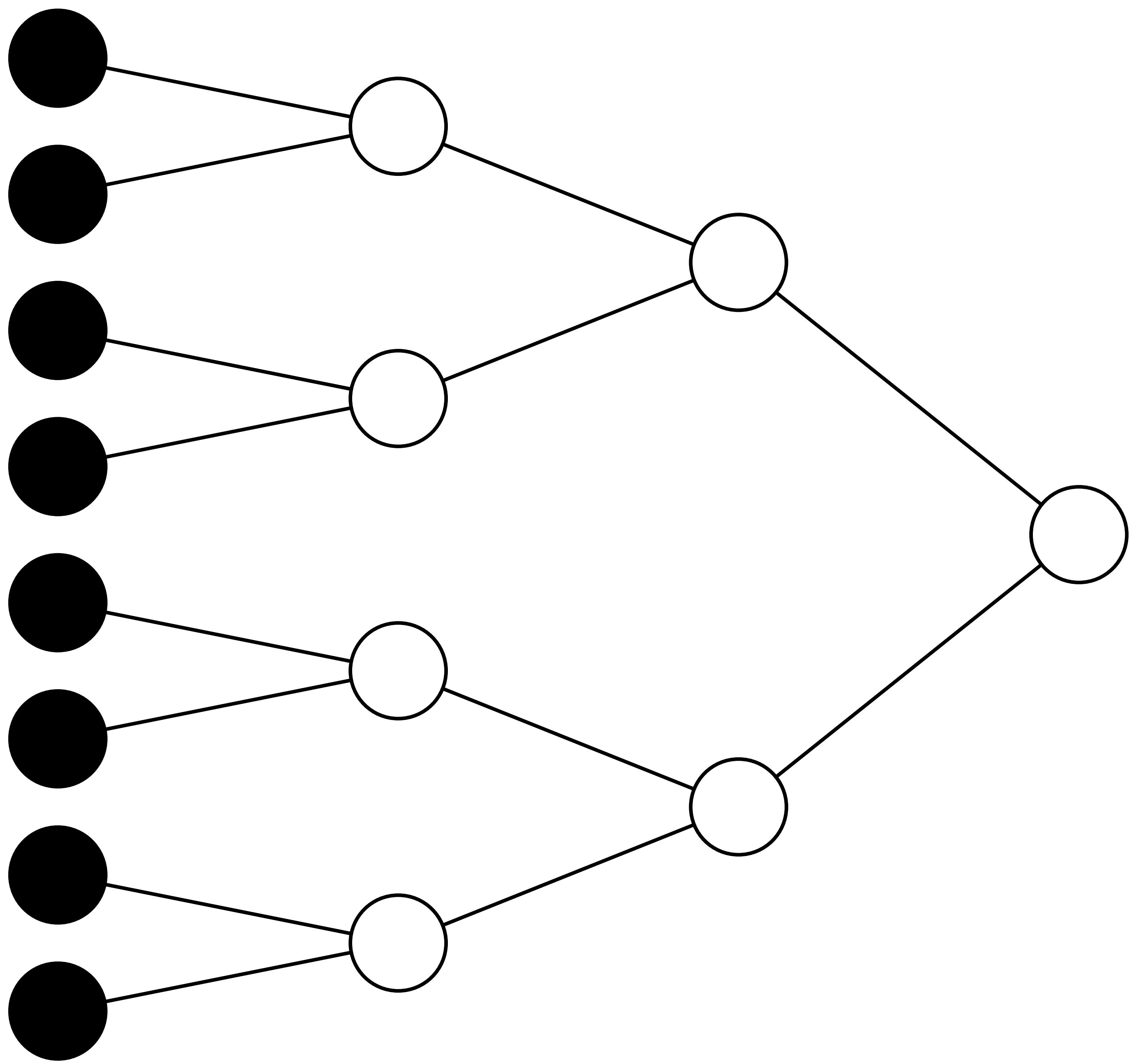
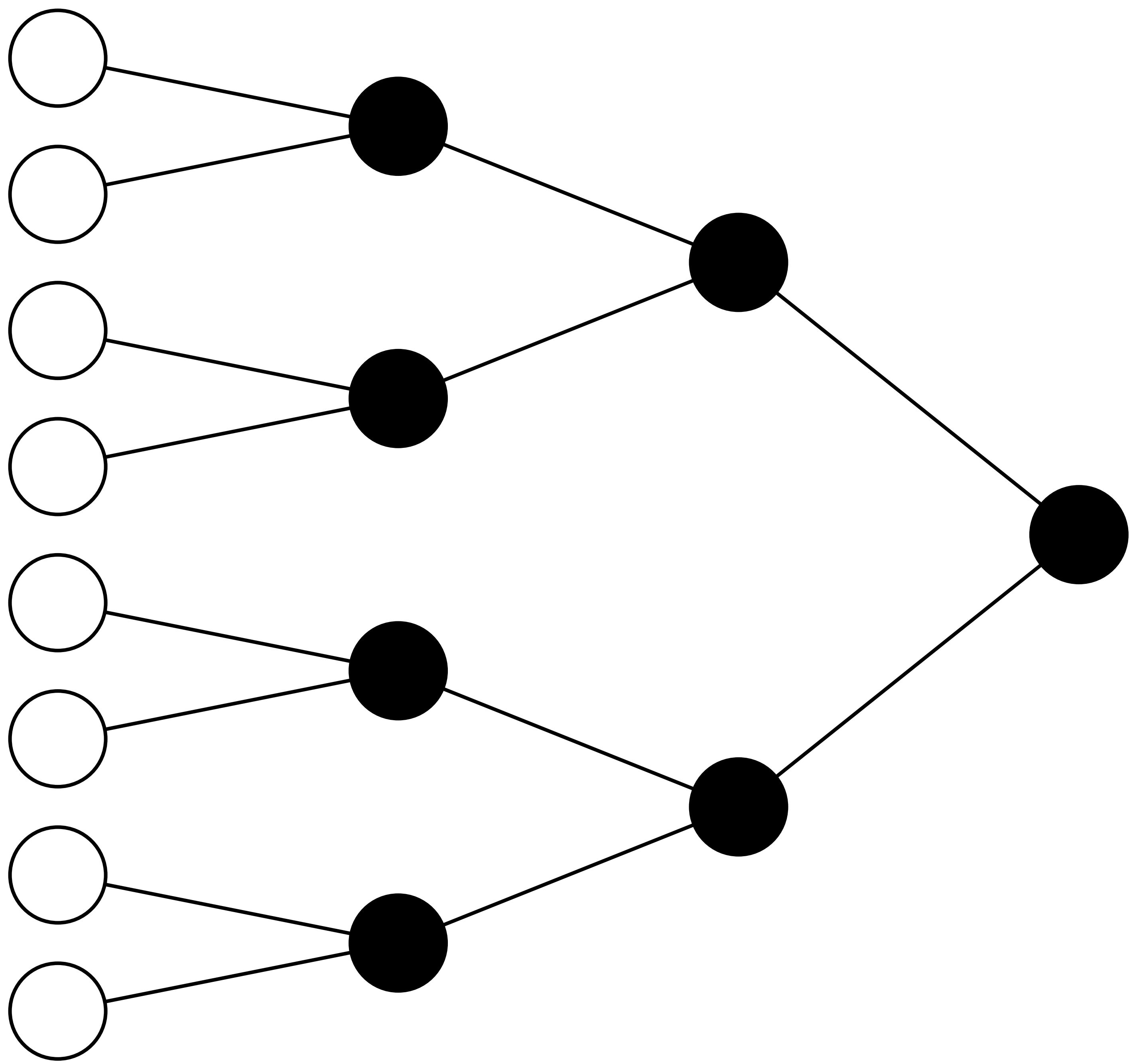
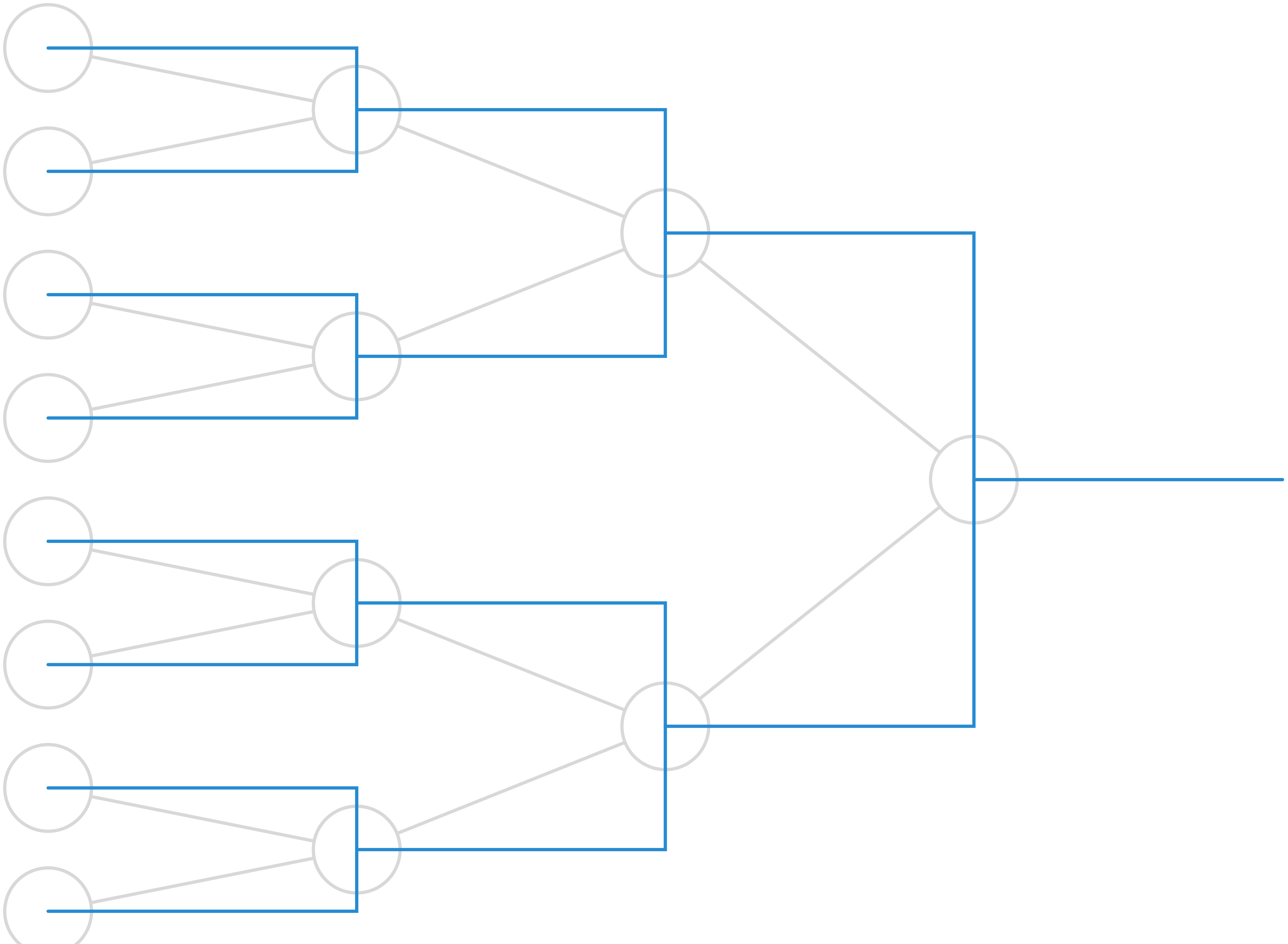


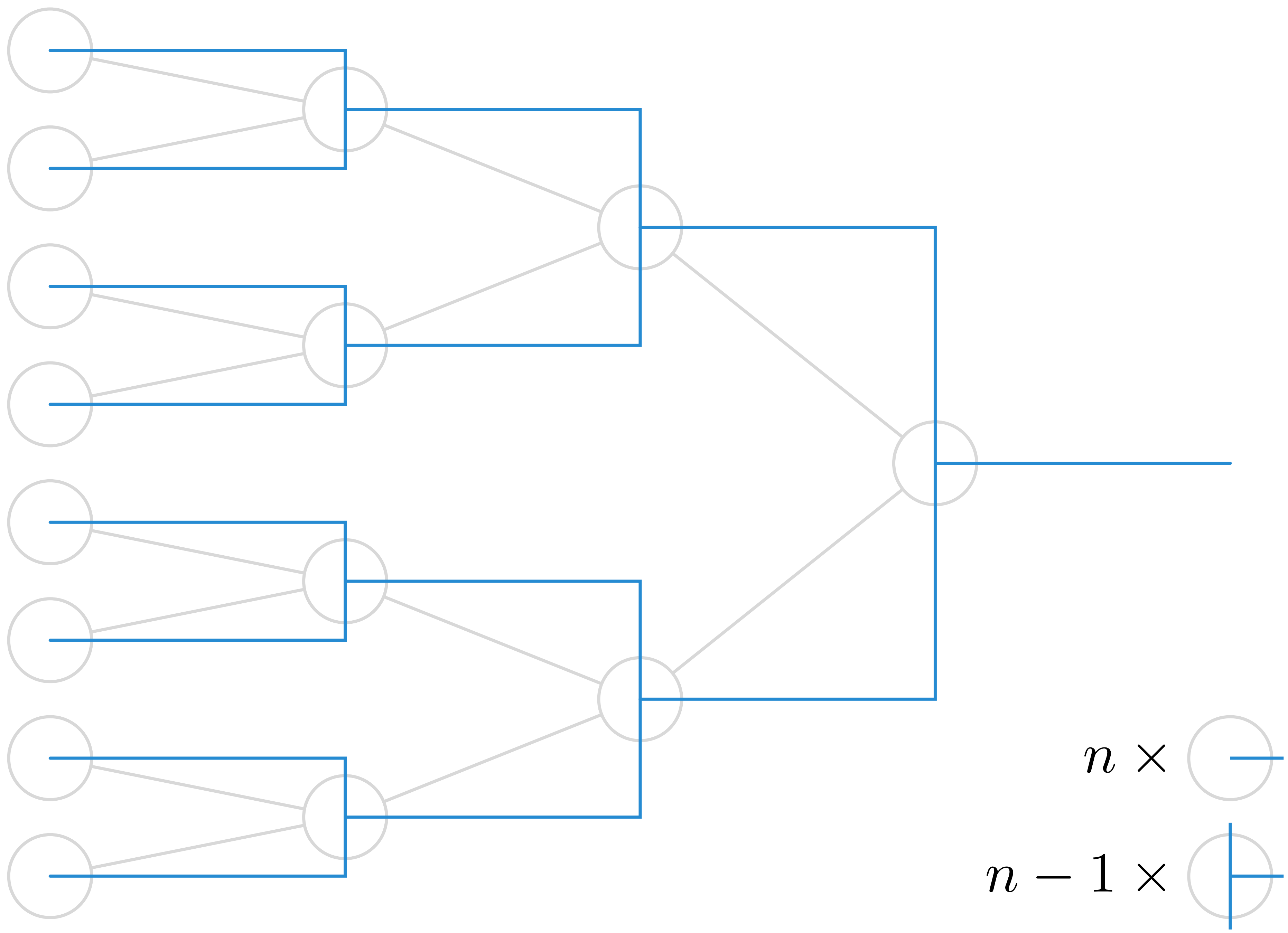
Binært tre med n løvnoder



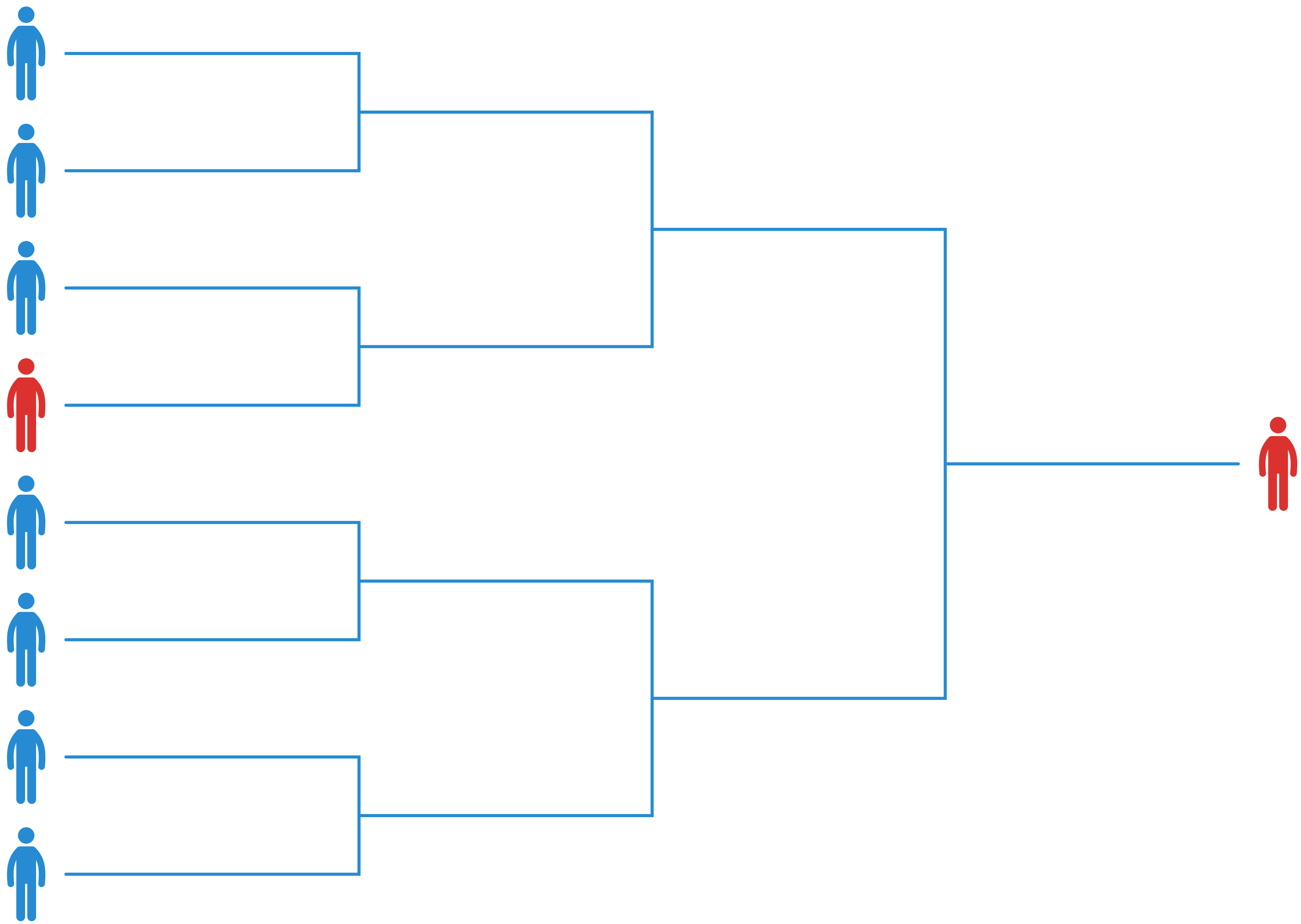
Hvorfor $n - 1$ interne noder?



Tenk på det som en knockout-turnering



$n - 1$ matcher slår ut alle unntatt vinneren



$n - 1$ matcher slår ut alle unntatt vinneren

Forelesning 4

Rangering i lineær tid

1. **Sorteringsgrensen**

2. **Tellesortering**

3. **Radikssortering**

4. **Bøttesortering**

5. **Randomized Select**

6. **Select**

0:6

Variabelskifte

Å bytte ut variable er en teknikk som også brukes i f.eks. kalkulus.

Se f.eks. https://en.wikipedia.org/wiki/Change_of_variables

$$\mathrm{T}(\sqrt{n}) = \lg n$$

$$\mathrm{T}(n^{\frac{1}{2}}) = \lg n$$

$$m \overset{\text{def}}{=} \lg n$$

For å bli kvitt logaritmen: Gi $\lg n$ et nytt navn

$$m \stackrel{\text{def}}{=} \lg n$$

$$T(2^{\frac{m}{2}}) = m$$

Potensen er fortsatt problematisk

$$m \overset{\text{def}}{=} \lg n$$

$$\mathrm{T}(2^{\frac{m}{2}}) = m$$

$$\mathrm{S}(m) \overset{\text{def}}{=} \mathrm{T}(n) = \mathrm{T}(2^m)$$

Gi $\mathrm{T}(2^m)$ et nytt navn!

$$m \stackrel{\text{def}}{=} \lg n$$

$$\mathrm{T}(2^{\frac{m}{2}}) = m$$

$$\mathrm{S}(m) \stackrel{\text{def}}{=} \mathrm{T}(n) = \mathrm{T}(2^m)$$

$$\mathrm{T}(2^{\frac{m}{2}}) = \mathrm{S}(m/2)$$

Hvis $\mathrm{S}(m) = \mathrm{T}(2^m)$ så må $\mathrm{S}(m/2)$ være $\mathrm{T}(2^{\frac{m}{2}})$

$$m \overset{\text{def}}{=} \lg n$$

$$\mathrm{T}(2^{\frac{m}{2}}) = m$$

$$\mathrm{S}(m) \overset{\text{def}}{=} \mathrm{T}(n) = \mathrm{T}(2^{m})$$

$$\mathrm{T}(2^{\frac{m}{2}}) = \mathrm{S}(m/2)$$

$$\mathrm{S}(m/2) = m$$

Slik ser ligningen vår ut med nye navn

$$m \stackrel{\text{def}}{=} \lg n$$

$$\mathrm{T}(2^{\frac{m}{2}}) = m$$

$$\mathrm{S}(m) \stackrel{\text{def}}{=} \mathrm{T}(n) = \mathrm{T}(2^m)$$

$$\mathrm{T}(2^{\frac{m}{2}}) = \mathrm{S}(m/2)$$

$$\mathrm{S}(m/2) = m$$

$$\mathrm{S}(m) = 2m$$

(Evt. enda et skifte: $x = m/2 \implies \mathrm{S}(x) = 2x \implies \mathrm{S}(m) = 2m$)

$$m \overset{\text{def}}{=} \lg n$$

$$\mathrm{T}(2^{\frac{m}{2}}) = m$$

$$\mathrm{S}(m) \overset{\text{def}}{=} \mathrm{T}(n) = \mathrm{T}(2^m)$$

$$\mathrm{T}(2^{\frac{m}{2}}) = \mathrm{S}(m/2)$$

$$\mathrm{S}(m/2) = m$$

$$\mathrm{S}(m) = 2m$$

$$\mathrm{T}(n) = 2\lg n$$

Her bruker vi bare definisjonene våre: $\mathrm{S}(m) = \mathrm{T}(n)$ og $m = \lg n$

$$T(n) = 2T(\sqrt{n}) + \lg n$$

$$T(n) = 2T(n^{\frac{1}{2}}) + \lg n$$

$$m \overset{\text{def}}{=} \lg n$$

For å bli kvitt logaritmen: Gi $\lg n$ et nytt navn

$$m \overset{\text{def}}{=} \lg n$$

$$\mathrm{T}(2^m) = \mathrm{T}(2^{\frac{m}{2}}) + m$$

Potensen er fortsatt problematisk

$$m \overset{\text{def}}{=} \lg n$$

$$\mathrm{T}(2^m) = \mathrm{T}(2^{\frac{m}{2}}) + m$$

$$\mathrm{S}(m) \overset{\text{def}}{=} \mathrm{T}(n) = \mathrm{T}(2^m)$$

Gi $\mathrm{T}(2^m)$ et nytt navn!

$$m \stackrel{\text{def}}{=} \lg n$$

$$\mathrm{T}(2^m) = \mathrm{T}(2^{\frac{m}{2}}) + m$$

$$\mathrm{S}(m) \stackrel{\text{def}}{=} \mathrm{T}(n) = \mathrm{T}(2^m)$$

$$\mathrm{T}(2^{\frac{m}{2}}) = \mathrm{S}(m/2)$$

Hvis $\mathrm{S}(m) = \mathrm{T}(2^m)$ så må $\mathrm{S}(m/2)$ være $\mathrm{T}(2^{\frac{m}{2}})$

$$m \overset{\text{def}}{=} \lg n$$

$$\mathrm{T}(2^m) = \mathrm{T}(2^{\frac{m}{2}}) + m$$

$$\mathrm{S}(m) \overset{\text{def}}{=} \mathrm{T}(n) = \mathrm{T}(2^m)$$

$$\mathrm{T}(2^{\frac{m}{2}}) = \mathrm{S}(m/2)$$

$$\mathrm{S}(m) = 2\mathrm{S}(m/2) + m$$

Slik ser ligningen vår ut med nye navn

$$m \overset{\text{def}}{=} \lg n$$

$$\mathrm{T}(2^m) = \mathrm{T}(2^{\frac{m}{2}}) + m$$

$$\mathrm{S}(m) \overset{\text{def}}{=} \mathrm{T}(n) = \mathrm{T}(2^m)$$

$$\mathrm{T}(2^{\frac{m}{2}}) = \mathrm{S}(m/2)$$

$$\mathrm{S}(m) = 2\mathrm{S}(m/2) + m$$

$$\mathrm{S}(m) = m \lg m + m$$

Standard rekurrensløsning, som tidligere i dag

$$
\begin{aligned}
m &\overset{\text{def}}{=} \lg n \\
\mathrm{T}(2^m) &= \mathrm{T}(2^{\frac{m}{2}}) + m \\
\mathrm{S}(m) &\overset{\text{def}}{=} \mathrm{T}(n) = \mathrm{T}(2^m) \\
\mathrm{T}(2^{\frac{m}{2}}) &= \mathrm{S}(m/2) \\
\mathrm{S}(m) &= 2\mathrm{S}(m/2) + m \\
\mathrm{S}(m) &= m \lg m + m \\
\mathrm{T}(n) &= \lg n \lg \lg n + \lg n
\end{aligned}
$$

Her bruker vi bare definisjonene våre: $\mathrm{S}(m) = \mathrm{T}(n)$ og $m = \lg n$

Hvor fort kan vi sortere n elementer om den eneste informasjonen vi får kommer fra parvise sammenligninger (såkalt sammenligningsbasert sortering)?

1:6

Sorteringsgrensen

A TOURNAMENT PROBLEM

LESTER R. FORD, JR.* AND SELMER M. JOHNSON, The RAND Corporation

Introduction. In his book,† Steinhaus discusses the problem of ranking n objects according to some transitive characteristic, by means of successive pair-wise comparisons. In this paper we shall adopt the terminology of a tennis tournament by n players. The problem may be briefly stated: "What is the smallest number of matches which will always suffice to rank all n players?"

Steinhaus proposes an inductive method whereby, the first k players having

Denne artikkelen, fra 1959, beskriver beslutningstre-modellen for analyse av sortering og rangering.

«Tenk på en permutasjon»

… av n! mulige

Trenger maks lg n! ja-nei-spørsmål

Worst-case: Tenk deg at en kjiping kjenner algoritmen din og velger input for deg.

«Tenk på en permutasjon»

… av n! mulige

Trenger lg n! ja-nei-spørsmål

$$\geqslant n!$$

Vi må skille mellom n! sekvenser med ja/nei-spørsmål

$$2^{\mathrm{T}(n)} \geqslant n!$$

Hvert spørsmål dobler hva vi kan skille mellom

$$\mathrm{T}(n) \geqslant \lg n!$$

$\lg n!$ er antall halveringer fra $n!$ til 1

$$T(n) \geqslant \lg n!$$

Om vi *ikke har flaks* kan vi ikke gjøre det bedre

$$\mathrm{T}(n) \geqslant \lg n!$$

Dette er en *nedre grense* for *verste tilfelle*

$$\mathrm{T_W}(n) \geqslant \lg n!$$

Dette er en *nedre grense* for *verste tilfelle*

$$\mathrm{T}_{\mathrm{W}}(n) = \Omega(\lg n!)$$

Dette er en *nedre grense* for *verste tilfelle*

$$
\begin{aligned}
\mathrm{T}_{\mathrm{W}}(n) &= \Omega(\lg n!) \\
&= \Omega(n \lg n)
\end{aligned}
$$

Fra Stirlings approksimasjon: $n! \geqslant (n/e)^n \implies \lg n! \geqslant n \lg n - n \lg e$

$$
\begin{aligned}
\mathrm{T}_{\mathrm{A}}(n) &= \Omega(\lg n!) \\
&= \Omega(n \lg n)
\end{aligned}
$$

$\lg n!$ er også informasjonsinnholdet i en tilfeldig permutasjon

$$T_W(n) =$$

$$T_W(n) =$$

$$T_W(n) = \Omega(n \lg n)$$

Dette har vi nettopp etablert

$$T_W(n) = O(\infty)$$

$$T_W(n) =$$

$$T_W(n) = \Omega(n \lg n)$$

Algoritmer kan bli vilkårlig dårlige

$$T_W(n) = O(\infty)$$

$$T_W(n) = \Theta(?)$$

$$T_W(n) = \Omega(n \lg n)$$

Vi har ingen generell felles øvre og nedre grense

$$\mathrm{T}_{\mathrm{A}}(n) = \mathrm{O}(\infty)$$

$$\mathrm{T}_{\mathrm{A}}(n) = \Theta(\,?\,)$$

$$\mathrm{T}_{\mathrm{A}}(n) = \Omega(n \lg n)$$

Det samme gjelder forventet kjøretid

$$\mathrm{T}_{\mathrm{B}}(n) = \mathrm{O}(\infty)$$

$$\mathrm{T}_{\mathrm{B}}(n) = \Theta(?)$$

$$\mathrm{T}_{\mathrm{B}}(n) = \Omega(n)$$

For å garantere rett svar må hele sekvensen behandles

Bryt grensen

... vha. ekstra antagelser

Vi antar at elementene består av ett siffer!
(Eller at de er verdier i et begrenset verdiområde 0…k)

2:6

Tellesortering

Fra 1954

$\textsc{Counting-Sort}(\mathrm{A}, \mathrm{B}, k)$

A input
B output
k maks

$\mathrm{A}, \mathrm{B}, k$: input, output, maksverdi $(\mathrm{A}[i] \in \{0, \ldots, k\})$

COUNTING-SORT(A, B, k)
1 let C[0 . . k] be a new array

A input
B output
k maks
C antall

Antall forekomster av hver mulig verdi

COUNTING-SORT(A, B, k)
let C[$0 \mathinner{..} k$] be a new array
for $i = 0$ **to** k

A input
B output
k maks
C antall

For hver mulig verdi i . . .

```
COUNTING-SORT(A, B, k)
let C[0..k] be a new array
for i = 0 to k
        C[i] = 0
```

A input
B output
k maks
C antall

Ingen forekomster funnet av i ennå

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
```

A input
B output
k maks
C antall
j pos. i A

For hvert posisjon i input-tabellen …

```
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
```

A input
B output
k maks
C antall
j pos. i A

Vi har nå sett enda en forekomst av A[j]

```
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
for i = 1 to k
```

A	input
B	output
k	maks
C	antall
j	pos. i A

For hver av tellingene unntatt 0 . . .

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** $A.\mathit{length}$
    $C[A[j]] = C[A[j]] + 1$
for $i = 1$ **to** k
    $C[i] = C[i] + C[i-1]$

A	input
B	output
k	maks
C	antall
j	pos. i A
i	pos. i C

Gjør C *kumulativ*: $C[i] =$ Hvor skal den siste i-en være?

```
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
for i = 1 to k
        C[i] = C[i] + C[i − 1]
for j = A.length downto 1
```

A input
B output
k maks
C antall
j pos. i A
i pos. i C

Vil ikke bytte like verdier; C angir *siste* forekomster!

```
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
  for i = 1 to k
      C[i] = C[i] + C[i − 1]
  for j = A.length downto 1
      B[C[A[j]]] = A[j]
```

A input
B output
k maks
C antall
j pos. i A
i pos. i C

$C[i]$ = Hvor skal den siste i-en være? $(i = A[j])$

```
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
for i = 1 to k
        C[i] = C[i] + C[i − 1]
for j = A.length downto 1
        B[C[A[j]]] = A[j]
        C[A[j]] = C[A[j]] − 1
```

A input
B output
k maks
C antall
j pos. i A
i pos. i C

Den neste i-en skal være ett hakk til venstre

```
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
for i = 1 to k
    C[i] = C[i] + C[i − 1]
for j = A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] − 1
```

	A	B
1	2	
2	5	
3	3	
4	0	
5	2'	
6	3'	
7	0'	
8	3''	

```
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
for i = 1 to k
        C[i] = C[i] + C[i − 1]
for j = A.length downto 1
        B[C[A[j]]] = A[j]
        C[A[j]] = C[A[j]] − 1
```

A

A	
2	1
5	2
3	3
0	4
2’	5
3’	6
0’	7
3”	8

C

C	
0	0
2	1
2	2
4	3
7	4
7	5

B

B	
0	1
0’	2
2	3
2’	4
3	5
3’	6
3”	7
5	8

$$\mathrm{T}(n) = \Theta(n + k)$$

$\Theta(1)$ iterasjoner over $\mathrm{A}[1\,..\,n]$, $\mathrm{B}[1\,..\,n]$ og $\mathrm{C}[0\,..\,k]$

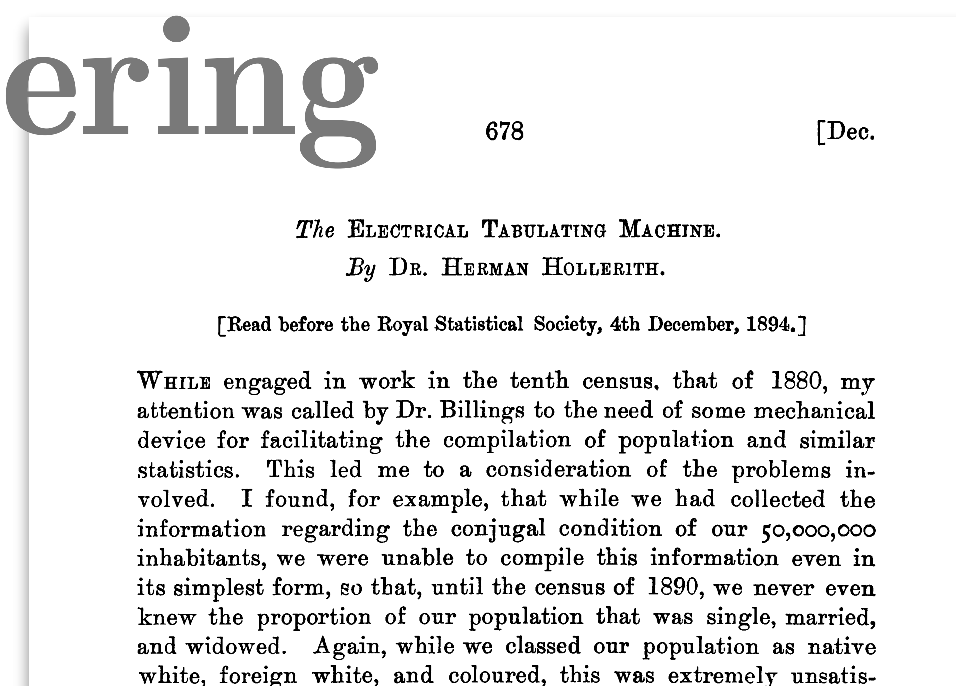
mlh@ntnu.no

Utvid verdiområdet

… uten lineær tidsøkning

3:6

Radikssortering



678 [Dec.

The Electrical Tabulating Machine.
By Dr. Herman Hollerith.

[Read before the Royal Statistical Society, 4th December, 1894.]

While engaged in work in the tenth census, that of 1880, my attention was called by Dr. Billings to the need of some mechanical device for facilitating the compilation of population and similar statistics. This led me to a consideration of the problems involved. I found, for example, that while we had collected the information regarding the conjugal condition of our 50,000,000 inhabitants, we were unable to compile this information even in its simplest form, so that, until the census of 1890, we never even knew the proportion of our population that was single, married, and widowed. Again, while we classed our population as native white, foreign white, and coloured, this was extremely unsatis-

```
RADIX-SORT(A, d)
for i = 1 to d
    sort* A by digit d
```

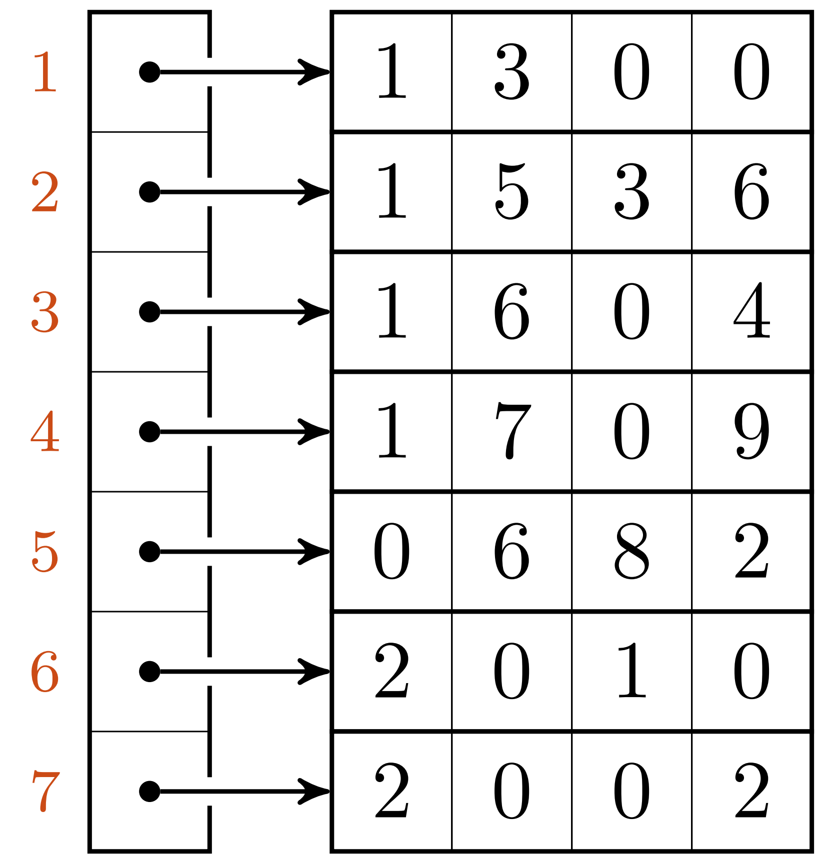
*Må være stabil[†]

[†]Må ikke bytte om like verdier

(Bruk f.eks. COUNTING-SORT)

RADIX-SORT(A, d)

```
for i = 1 to d
    sort A on digit d
```

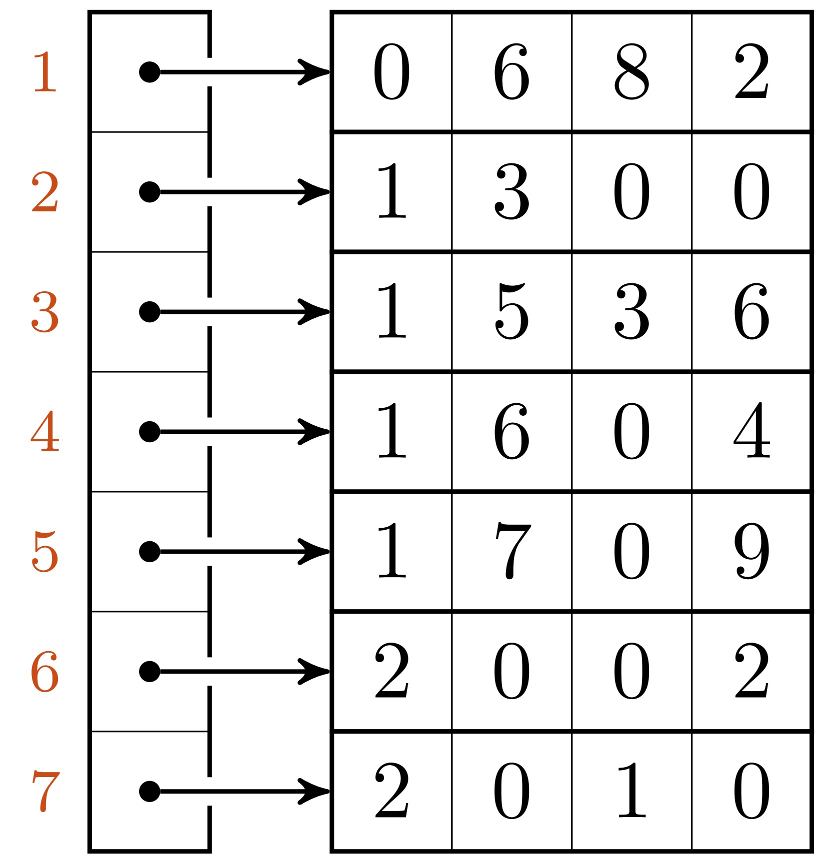


i = -

Radix-Sort(A, d)

for $i = 1$ **to** d
    sort A on digit d

i = -



Det *finnes* varianter som går denne veien også – men de gjør andre ting *i tillegg*. Så denne rett-frem-varianten er rett og slett feil.

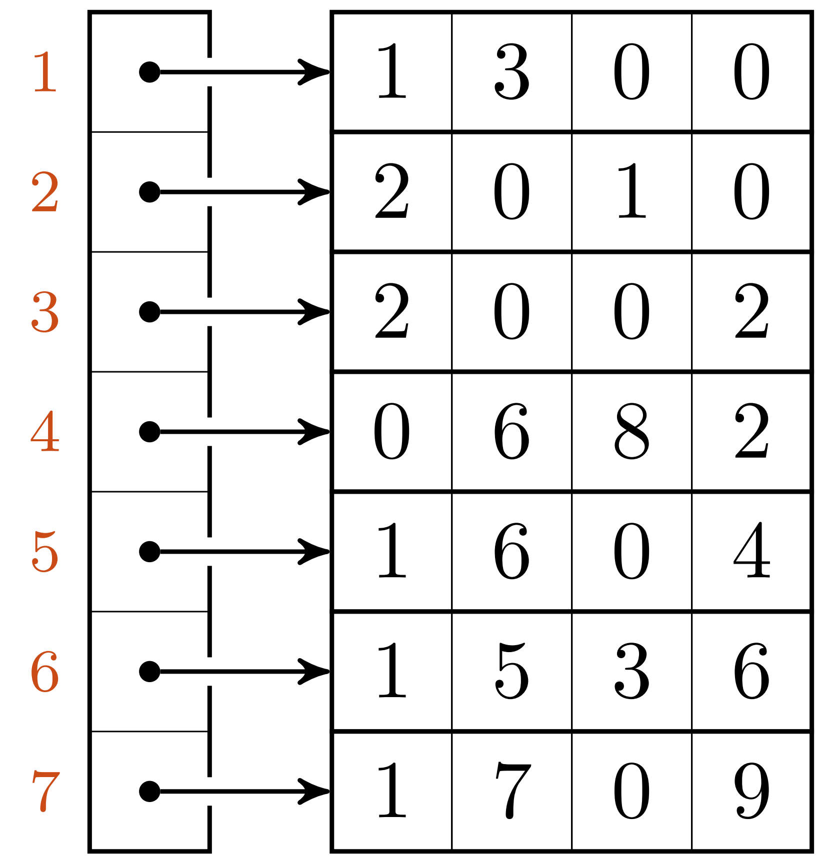
```
XIDAR-NON-SORT(A, d)
for i = d downto 1
    sort A on digit d
```

1	1	3	0	0
2	1	5	3	6
3	1	6	0	4
4	1	7	0	9
5	0	6	8	2
6	2	0	1	0
7	2	0	0	2

i = -

XIDAR-NON-SORT(A, d)

for $i = d$ **downto** 1
    sort A on digit d



i = -

$$T(n) = \Theta(d \cdot (n + k))$$

Kjøretiden er altså avhengig av antall verdier (n), antall mulige verdier for hvert siffer (k) og antall siffer (d).

COUNTING-SORT (kjøretid $\Theta(n + k)$) for hvert av d siffer

Bryt grensen

… denne gang for AC

Ingen garantier, men antar at inputs er uniformt fordelte tilfeldige tall i intervallet [0,1).

4:6

Bøttesortering

Sorting by Address Calculation*

E. J. Isaac and R. C. Singleton

Stanford Research Institute, Menlo Park, California

General Description of Method

Sorting in a random access memory is essentially a process of associating the address of the location in which each item is to be placed with the identifying key of the item. The fewer times the items have to be moved from one location

BUCKET-SORT(A)

A input

Verdier i A: Uniformt fordelte, $[0, 1)$

Bucket-Sort(A)
1 $n = A.length$

A input
n lengde

Antall bøtter; ett element per bøtte

```
BUCKET-SORT(A)
n = A.length
create B[0 . . n − 1]
```

A input
n lengde
B bøtter

Hodepekere til bøttene (lister)

```
BUCKET-SORT(A)
  n = A.length
  create B[0 . . n − 1]
  for i = 1 to n
```

A input
n lengde
B bøtter
i bøttenr

mlh@ntnu.no

```
BUCKET-SORT(A)
n = A.length
create B[0 . . n − 1]
for i = 1 to n
        make B[i] an empty list
```

A	input
n	lengde
B	bøtter
i	bøttenr

(Evt. bruk dynamiske tabeller)

BUCKET-SORT(A)

$n = \text{A}.\mathit{length}$

create $\text{B}[0\,..\,n-1]$

for $i = 1$ **to** n

  make $\text{B}[i]$ an empty list

for $i = 1$ **to** n

A	input
n	lengde
B	bøtter
i	bøttenr

Bucket-Sort(A)

$n = A.\mathit{length}$
create $B[0\,..\,n-1]$
for $i = 1$ **to** n
  make $B[i]$ an empty list
for $i = 1$ **to** n
  add $A[i]$ to $B[\lfloor nA[i] \rfloor]$

A input
n lengde
B bøtter
i bøttenr

«Ryddig hashing»; kan regne ut rett bøtte direkte

Bucket-Sort(A)

$n = \mathrm{A}.\mathit{length}$
create $\mathrm{B}[0 \,..\, n-1]$
for $i = 1$ **to** n
    make $\mathrm{B}[i]$ an empty list
for $i = 1$ **to** n
    add $\mathrm{A}[i]$ to $\mathrm{B}[\lfloor n\mathrm{A}[i] \rfloor]$
for $i = 0$ **to** $n - 1$

A	input
n	lengde
B	bøtter
i	bøttenr

Bucket-Sort(A)

$n = \text{A}.\mathit{length}$
create $\text{B}[0\,..\,n-1]$
for $i = 1$ **to** n
    make $\text{B}[i]$ an empty list
for $i = 1$ **to** n
    add $\text{A}[i]$ to $\text{B}[\lfloor n\text{A}[i]\rfloor]$
for $i = 0$ **to** $n - 1$
    sort list $\text{B}[i]$

A input
n lengde
B bøtter
i bøttenr

Bruk f.eks. Insertion-Sort. Forventet: $\text{B}[i].\mathit{length} = \text{O}(1)$

BUCKET-SORT(A)

$n = \text{A}.\mathit{length}$
create B$[0\,.\,.\,n-1]$
for $i = 1$ **to** n
    make B$[i]$ an empty list
for $i = 1$ **to** n
    add A$[i]$ to B$[\lfloor n\text{A}[i] \rfloor]$
for $i = 0$ **to** $n-1$
    sort list B$[i]$
concatenate B$[0] \ldots$ B$[n-1]$

A	input
n	lengde
B	bøtter
i	bøttenr

Ferdig sorterte, i rett rekkefølge

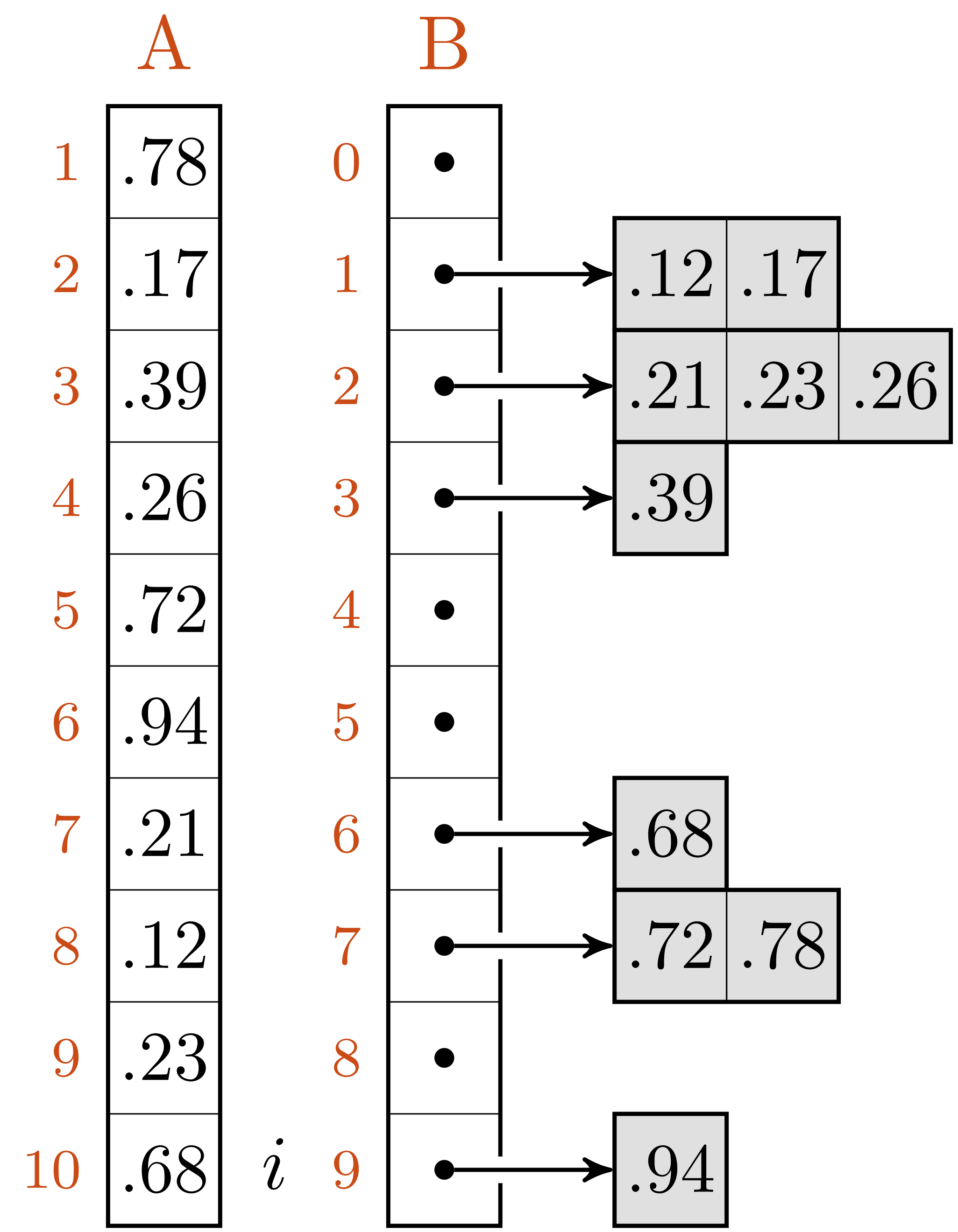
```
Bucket-Sort(A)
n = A.length
create B[0 . . n − 1]
for i = 1 to n
        make B[i] an empty list
for i = 1 to n
        add A[i] to B[⌊nA[i]⌋]
for i = 0 to n − 1
        sort list B[i]
concatenate B[0] . . . B[n − 1]
```

	A
1	.78
2	.17
3	.39
4	.26
5	.72
6	.94
7	.21
8	.12
9	.23
10	.68

	B
0	•
1	•
2	•
3	•
4	•
5	•
6	•
7	•
8	•
9	•

Bucket-Sort(A)

$n = \mathrm{A}.\mathit{length}$
create $\mathrm{B}[0\,..\,n-1]$
for $i = 1$ **to** n
    make $\mathrm{B}[i]$ an empty list
for $i = 1$ **to** n
    add $\mathrm{A}[i]$ to $\mathrm{B}[\lfloor n\mathrm{A}[i] \rfloor]$
for $i = 0$ **to** $n-1$
    sort list $\mathrm{B}[i]$
concatenate $\mathrm{B}[0] \ldots \mathrm{B}[n-1]$



$$\mathrm{T}_{\mathrm{W}}(n) = \Theta(n^2)$$

$$\mathrm{T}_{\mathrm{A}}(n) = \Theta(n)$$

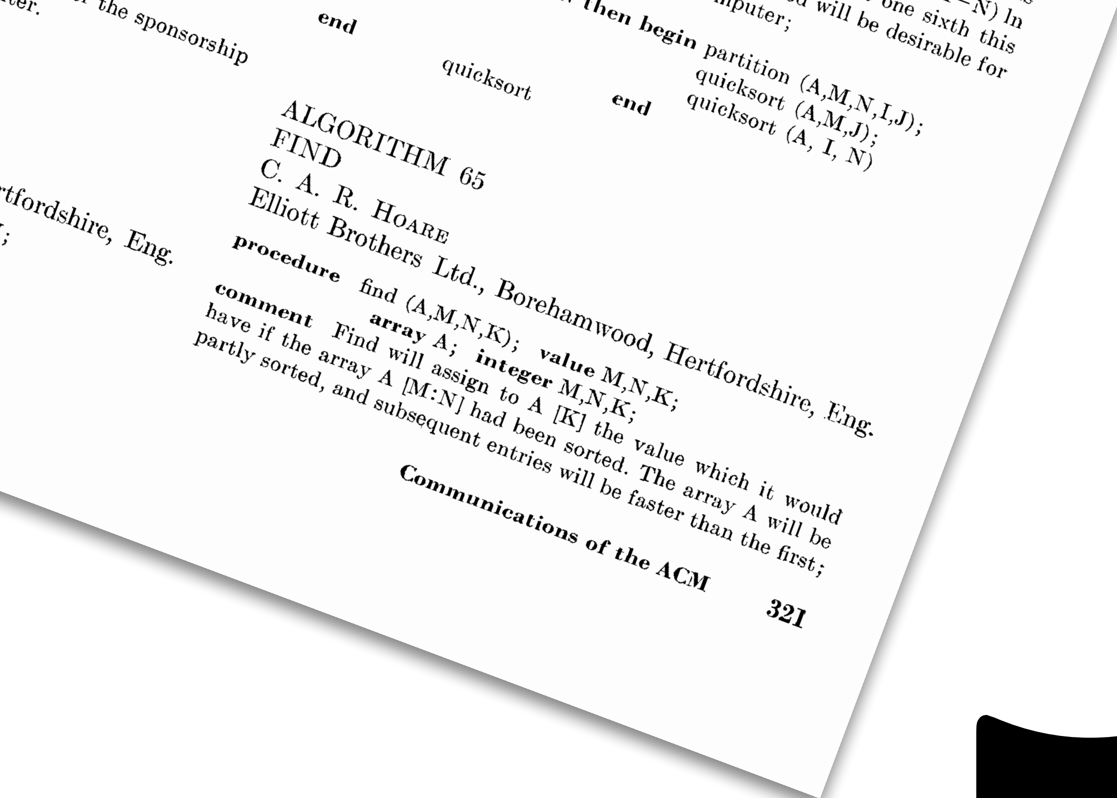
$$\mathrm{T}_{\mathrm{B}}(n) = \Theta(n)$$

INSERTION-SORT på n bøtter med forv. lengde $\Theta(1)$

Bryt grensen for AC

… ved å begrense problemet

Vi vil bare rangere *noen* elementer, heller enn *alle. Men vil antar her som for sammenligningsbasert sortering at vi bare kan rangere objekter parvis.


ALGORITHM 65
FIND
C. A. R. HOARE
Elliott Brothers Ltd., Borehamwood, Hertfordshire, Eng.

procedure find (A,M,N,K); **value** M,N,K;
array A; **integer** M,N,K;
comment Find will assign to A [K] the value which it would have if the array A [M:N] had been sorted. The array A will be partly sorted, and subsequent entries will be faster than the first;

Communications of the ACM **321**

5:6

Randomized Select

Hvem er på F.eks. 10. plass?

Antar distinkte verdier!

Induksjon/rekursjon

Anta mindre instanser kan løses

Vi kan da løse disse rekursivt!

«Quicksort som binærsøk»

$\textsc{Rand-Sel}(\mathrm{A}, p, r, i)$

A	tabell
p	venstre
r	høyre
i	rang

Finn det i-ende minste elementet i $\mathrm{A}[p \mathinner{.\,.} r]$

RAND-SEL(A, p, r, i)
1 **if** $p == r$

A	tabell
p	venstre
r	høyre
i	rang

Bare ett element: Det må være det vi leter etter!

Rand-Sel(A, p, r, i)
1 **if** $p == r$
2 **return** $A[p]$

A	tabell
p	venstre
r	høyre
i	rang

Bare ett element: Det må være det vi leter etter!

RAND-SEL(A, p, r, i)
1 **if** $p == r$
2 **return** $A[p]$
3 $q =$ RAND-PARTITION(A, p, r)

A	tabell
p	venstre
r	høyre
i	rang
q	splitt

Nå er $A[q]$ på rett plass, akkurat som i QUICKSORT!

```
RAND-SEL(A, p, r, i)
if p == r
    return A[p]
q = RAND-PARTITION(A, p, r)
k = q − p + 1
```

A tabell
p venstre
r høyre
i rang
q splitt
k rang, q

A[q] er det k-ende minste elementet i A[$p\,..\,r$]

Rand-Sel(A, p, r, i)

if $p == r$

  return $A[p]$

$q = $ Rand-Partition(A, p, r)

$k = q - p + 1$

if $i == k$

A	tabell
p	venstre
r	høyre
i	rang
q	splitt
k	rang, q

Er $A[q]$ det i-ende minste elementet i $A[p \,..\, r]$?

Rand-Sel(A, p, r, i)

if $p == r$
    return $A[p]$
$q = $ Rand-Partition(A, p, r)
$k = q - p + 1$
if $i == k$
    return $A[q]$

A	tabell
p	venstre
r	høyre
i	rang
q	splitt
k	rang, q

Da er det det vi leter etter!

```
Rand-Sel(A, p, r, i)
if p == r
        return A[p]
q = Rand-Partition(A, p, r)
k = q − p + 1
if i == k
        return A[q]
elseif i < k
```

A tabell
p venstre
r høyre
i rang
q splitt
k rang, q

Leter vi etter et mindre element?

```
Rand-Sel(A, p, r, i)
if p == r
    return A[p]
q = Rand-Partition(A, p, r)
k = q − p + 1
if i == k
    return A[q]
elseif i < k
    return Rand-Sel(A, p, q − 1, i)
```

A	tabell
p	venstre
r	høyre
i	rang
q	splitt
k	rang, q

Let blant de små: Finn det k-ende minste i $A[p \mathinner{.\,.} q-1]$

$\textsc{Rand-Sel}(A, p, r, i)$

if $p == r$
    return $A[p]$
$q = \textsc{Rand-Partition}(A, p, r)$
$k = q - p + 1$
if $i == k$
    return $A[q]$
elseif $i < k$
    return $\textsc{Rand-Sel}(A, p, q-1, i)$
else return $\textsc{Rand-Sel}(A, q+1, r, i-k)$

A	tabell
p	venstre
r	høyre
i	rang
q	splitt
k	rang, q

Ellers: Finn det $(i - k)$-ende minste i $A[q+1 \,..\, r]$

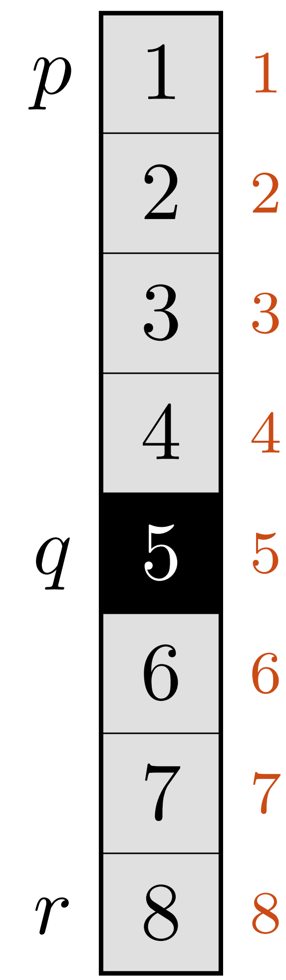
```
RS(A, p, r, i)
if p == r
        return A[p]
q = Rand-Partition(A, p, r)
k = q − p + 1
if i == k
        return A[q]
elseif i < k
        return RS(A, p, q − 1, i)
else return RS(A, q + 1, r, i − k)
```

$k, i \;=\; \text{-}, 4$

p	7	1
	8	2
	6	3
	1	4
	4	5
	2	6
	3	7
r	5	8

```
RS(A, p, r, i)
1  if p == r
2      return A[p]
3  q = Rand-Partition(A, p, r)
4  k = q − p + 1
5  if i == k
6      return A[q]
7  elseif i < k
8      return RS(A, p, q − 1, i)
9  else return RS(A, q + 1, r, i − k)
```

$\longrightarrow$ 4

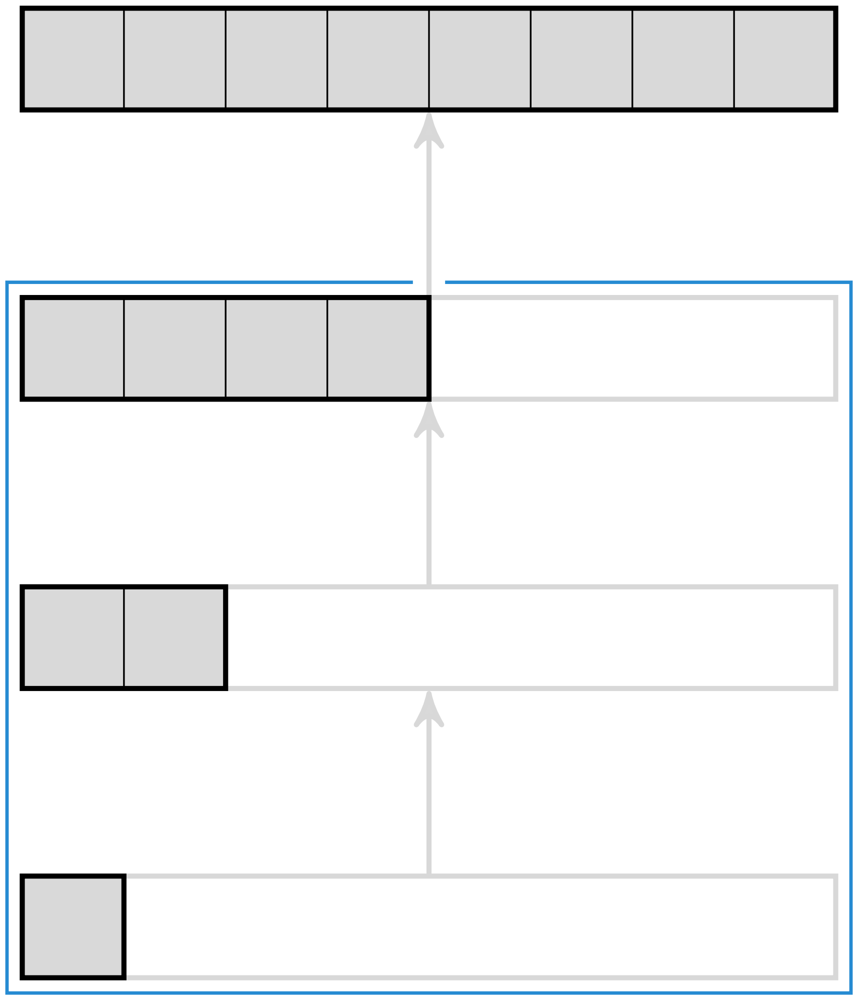


$k, i \;=\; 5, 4$

$$T(n) = T(n/2) + n$$

Ett rekursivt kall (størrelse $n/2$) pluss n operasjoner

$\mathrm{T}(n/2)$



$$
\begin{aligned}
\mathrm{T}(n) = n & \\
+\, n/2 & \qquad (1)\\
+\, n/4 & \qquad (2)\\
+\, n/8 & \qquad (3)\\
\vdots\ & \qquad \vdots\\
+\, 1 & \qquad (\lg n)
\end{aligned}
$$

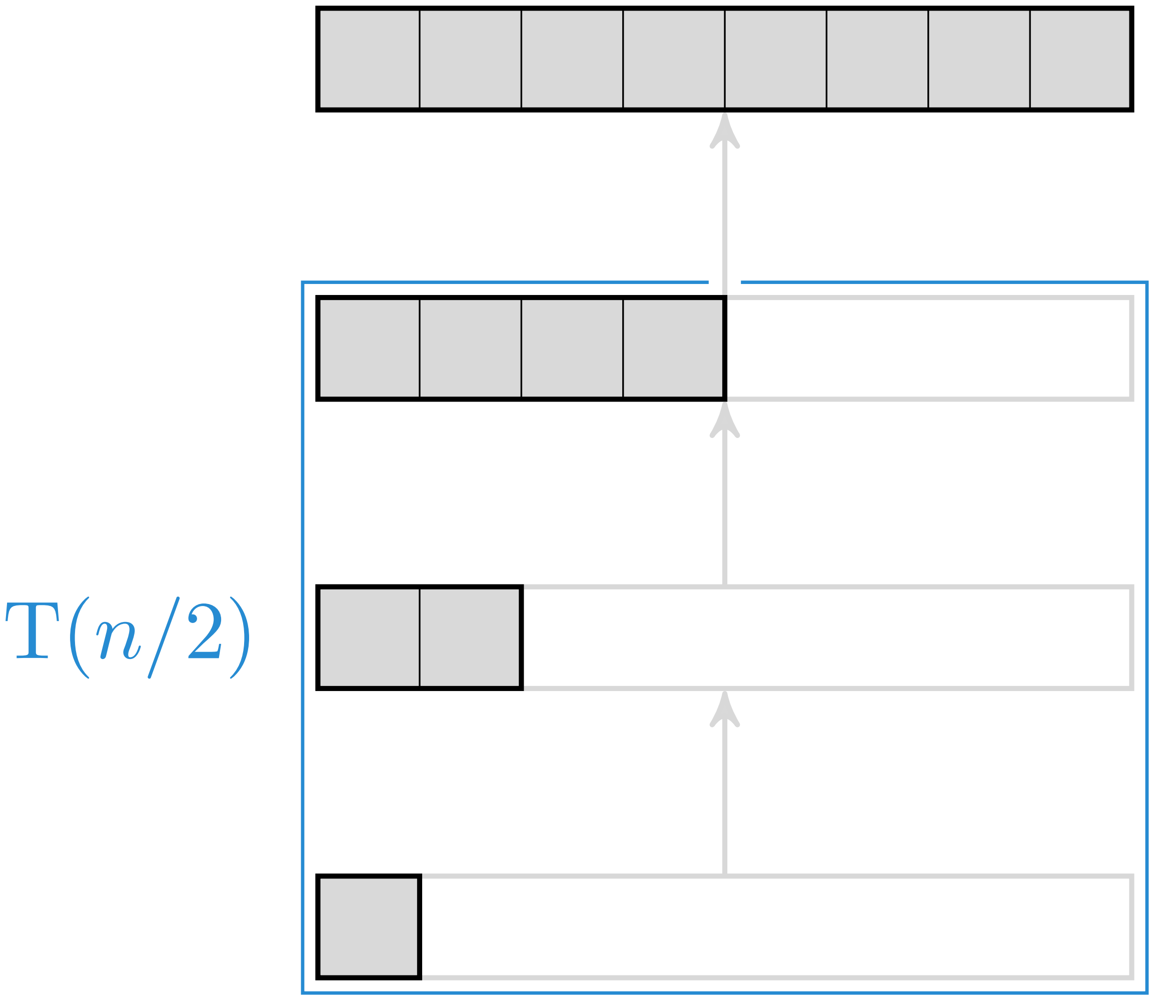
$$\mathrm{T}(n) = 2n - 1$$

Verifikasjon

Med substitusjon/induksjon

$$
\begin{aligned}
\mathrm{T}(n) &= \mathrm{T}(n/2) + n \\
&= \left(2 \cdot \frac{n}{2} - 1\right) + n \\
&= n - 1 + n \\
&= 2n - 1
\end{aligned}
$$

Gitt $\mathrm{T}(k) = 2k - 1$ for $k < n$, vis $\mathrm{T}(n) = 2n - 1$



$$T(n) = 2n - 1$$

$$\mathrm{T_W}(n) = \Theta(n^2)$$

$$\mathrm{T_A}(n) = \Theta(n)$$

$$\mathrm{T_B}(n) = \Theta(n)$$

I verste tilfelle: Pivot alene, akkurat som QUICKSORT

$$\mathrm{T}_{\mathrm{W}}(n) = \Theta(n^2)$$

$$\mathrm{T}_{\mathrm{A}}(n) = \Theta(n)$$

$$\mathrm{T}_{\mathrm{B}}(n) = \Theta(n)$$

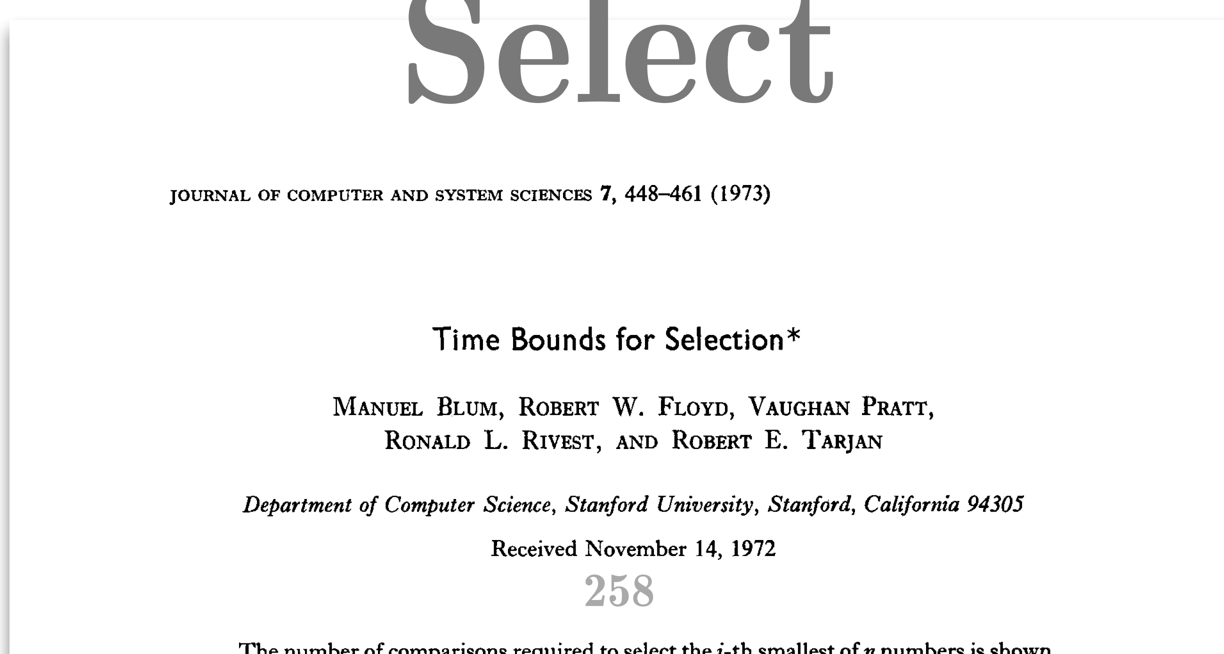
Forventet: $\Theta(n + n/2 + \cdots + 4 + 2 + 1)$ operasjoner

Gjenta suksessen!

... denne gang for WC

6:6

Select


JOURNAL OF COMPUTER AND SYSTEM SCIENCES **7**, 448–461 (1973)

Time Bounds for Selection*

MANUEL BLUM, ROBERT W. FLOYD, VAUGHAN PRATT,
RONALD L. RIVEST, AND ROBERT E. TARJAN

Department of Computer Science, Stanford University, Stanford, California 94305

Received November 14, 1972

258

The number of comparisons required to select the i-th smallest of n numbers is shown

Trenger god pivot

Bruk … Select?

«Median av medianer»

PARTITION(A, p, r)

$x = \mathrm{A}[r]$
$i = p - 1$
for $j = p$ **to** $r - 1$
    if $\mathrm{A}[j] \leq x$
        $i = i + 1$
        exchange $\mathrm{A}[i]$ with $\mathrm{A}[j]$
exchange $\mathrm{A}[i+1]$ with $\mathrm{A}[r]$
return $i + 1$

Bruker $\mathrm{A}[r]$ som pivot

```
PARTITION-AROUND(A, p, r, x)
i = 1
while A[i] ≠ x
        i = i + 1
exchange A[r] and A[i]
return PARTITION(A, p, r)
```

Får pivot oppgitt! Beskrevet uten kode i boka

```
Rand-Sel(A, p, r, i)
if p == r
    return A[p]
q = Rand-Partition(A, p, r)
k = q − p + 1
if i == k
    return A[q]
elseif i < k
    return Rand-Sel(A, p, q − 1, i)
else return Rand-Sel(A, q + 1, r, i − k)
```

La oss bytte ut alle Randomized-tingene

```
SELECT(A, p, r, i)
if p == r
    return A[p]
q = GOOD-PARTITION(A, p, r)
k = q − p + 1
if i == k
    return A[q]
elseif i < k
    return SELECT(A, p, q − 1, i)
else return SELECT(A, p + 1, r, i − k)
```

Partisjoneringen er kjernen: Finn en god pivot!

Good-Partition(A, p, r)

A	tabell
p	venstre
r	høyre

Velger pivot nøye. Beskrevet uten kode i boka

GOOD-PARTITION(A, p, r)

1 $n = r - p + 1$

A	tabell
p	venstre
r	høyre
n	antall

$$n = A[p\,..\,r].length$$

$\textsc{Good-Partition}(\mathrm{A}, p, r)$

$n = r - p + 1$

$m = \lceil n/5 \rceil$

A	tabell
p	venstre
r	høyre
n	antall
m	grupper

Vi vil dele $\mathrm{A}[p \mathinner{.\,.} r]$ i grupper på fem

$\textsc{Good-Partition}(\mathrm{A}, p, r)$

$n = r - p + 1$
$m = \lceil n/5 \rceil$
create $\mathrm{B}[1\,..\,m]$

A	tabell
p	venstre
r	høyre
n	antall
m	grupper
B	medianer

Vil inneholde medianen for hver av femmergruppene

Good-Partition(A, p, r)

1 $n = r - p + 1$
2 $m = \lceil n/5 \rceil$
3 create $B[1 \,..\, m]$
4 **for** $i = 0$ **to** $m - 1$

A	tabell
p	venstre
r	høyre
n	antall
m	grupper
B	medianer
i	gruppe $- 1$

For hver femmergruppe …

Good-Partition(A, p, r)

```
n = r − p + 1
m = ⌈n/5⌉
create B[1 . . m]
for i = 0 to m − 1
        q = p + 5i
```

A	tabell
p	venstre
r	høyre
n	antall
m	grupper
B	medianer
i	gruppe − 1
q	v., gruppe

Gruppen starter med A[q]

Good-Partition(A, p, r)

```
n = r − p + 1
m = ⌈n/5⌉
create B[1 . . m]
for i = 0 to m − 1
        q = p + 5i
        sort A[q . . q + 4]
```

A	tabell
p	venstre
r	høyre
n	antall
m	grupper
B	medianer
i	gruppe − 1
q	v., gruppe

For å finne medianen i gruppen. Bruk f.eks. Insertion-Sort

```
GOOD-PARTITION(A, p, r)
n = r − p + 1
m = ⌈n/5⌉
create B[1 . . m]
for i = 0 to m − 1
        q = p + 5i
        sort A[q . . q + 4]
        B[i] = A[q + 3]
```

A	tabell
p	venstre
r	høyre
n	antall
m	grupper
B	medianer
i	gruppe − 1
q	v., gruppe

Sett medianen inn i B

Good-Partition(A, p, r)

```
n = r − p + 1
m = ⌈n/5⌉
create B[1 . . m]
for i = 0 to m − 1
        q = p + 5i
        sort A[q . . q + 4]
        B[i] = A[q + 3]
x = Select(B, 1, m, ⌊m/2⌋)
```

A	tabell
p	venstre
r	høyre
n	antall
m	grupper
B	medianer
i	gruppe − 1
q	v., gruppe
x	splitt

Finn medianen av medianene . . . med Select!

GOOD-PARTITION(A, p, r)

$n = r - p + 1$

$m = \lceil n/5 \rceil$

create B$[1 \,..\, m]$

for $i = 0$ **to** $m - 1$

  $q = p + 5i$

  sort A$[q \,..\, q + 4]$

  B$[i]$ = A$[q + 3]$

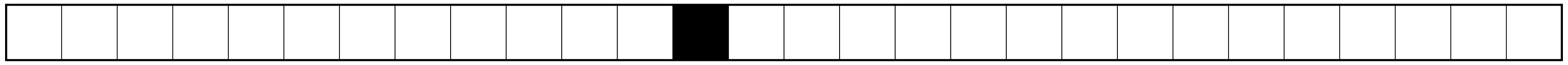
x = SELECT(B, 1, m, $\lfloor m/2 \rfloor$)

return PARTITION-AROUND(A, p, r, x)

A	tabell
p	venstre
r	høyre
n	antall
m	grupper
B	medianer
i	gruppe $-$ 1
q	v., gruppe
x	splitt

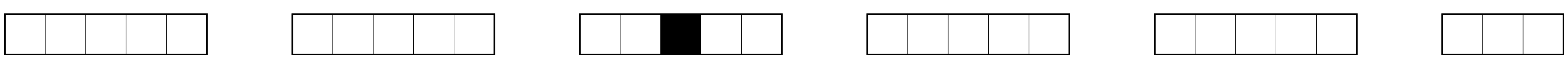
Bruk medianen av medianene som pivot

$$3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geqslant \frac{3n}{10} - 6$$



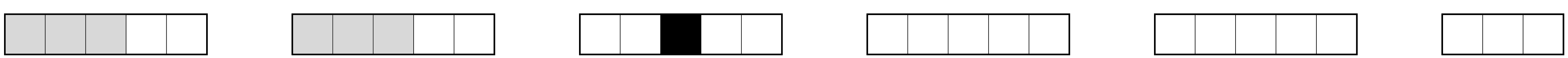
Hvor mange har vi på hver side av pivot?

$$3\left(\left\lceil\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil\right\rceil - 2\right) \geqslant \frac{3n}{10} - 6$$



Vi har delt inn i $\lceil n/5 \rceil$ grupper

$$3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geqslant \frac{3n}{10} - 6$$



Minst halvparten bidrar med 3 verdier mindre enn pivot

$$3\left(\left\lceil\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil\right\rceil - 2\right) \geqslant \frac{3n}{10} - 6$$

Unntatt én, om $\lceil n/5 \rceil > n/5$ . . .

$$3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil\right\rceil - 2\right) \geqslant \frac{3n}{10} - 6$$

… og unntatt gruppen med pivot

$$3\left(\left\lceil\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil\right\rceil-2\right)\geqslant\frac{3n}{10}-6$$

Vi har altså minst så mange elementer mindre enn pivot …

$$3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geqslant \frac{3n}{10} - 6$$

… og så mange som er større

$$\mathrm{T}(n) = \Theta(n)$$

Garantert en viss prosent på hver side av pivot

1. Sorteringsgrensen
2. Tellesortering
3. Radikssortering
4. Bøttesortering
5. Randomized Select
6. Select