Forelesning 3

Rekursiv dekomponering er kanskje den viktigste ideen i hele faget, og designmetoden *splitt og hersk* er en grunnleggende utgave av det: Del instansen i mindre biter, løs problemet rekursivt for disse, og kombinér løsningene.

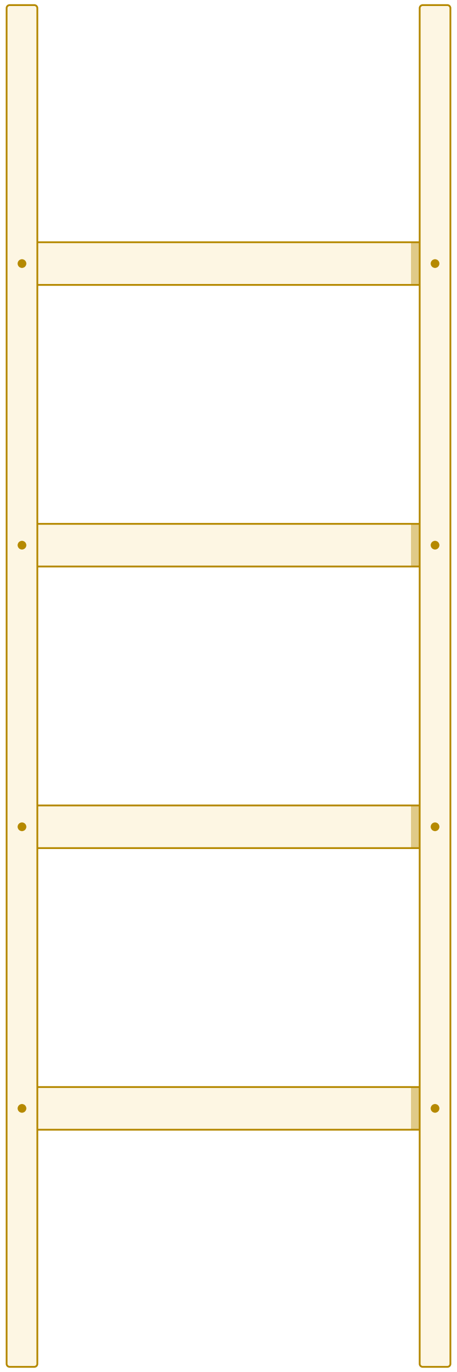
Pensum

- ☐ Kap. 2. Getting started: 2.3
- ☐ Kap. 4. Divide-and-conquer: Innledning, 4.1 og 4.3–4.5
- ☐ Kap. 7. Quicksort
- ☐ Oppgaver 2.3-5 og 4.5-3 med løsning (binærsøk)
- ☐ Appendiks B og C i pensumheftet

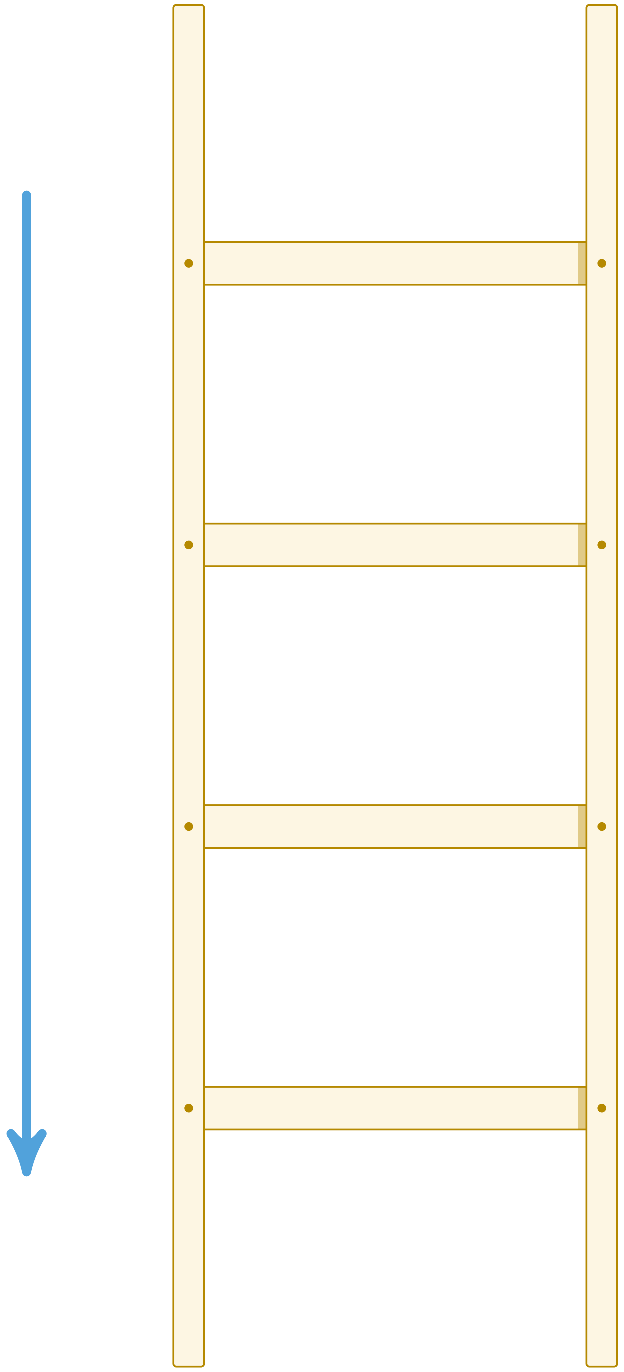
Læringsmål

- [C_1] Forstå *divide-and-conquer* (*splitt og hersk*)
- [C_2] Forstå *maximum-subarray*-problemet med løsninger
- [C_3] Forstå Bisect og Bisect′ (se appendiks C i pensumheftet)
- [C_4] Forstå Merge-Sort
- [C_5] Forstå Quicksort og Randomized-Quicksort
- [C_6] Kunne løse rekurrenser med *substitusjon*, *rekursjonstrær* og *masterteoremet*
- [C_7] Kunne løse rekurrenser med *iterasjonsmetoden* (se appendiks B i pensumheftet)
- [C_8] Forstå hvordan *variabelskifte* fungerer

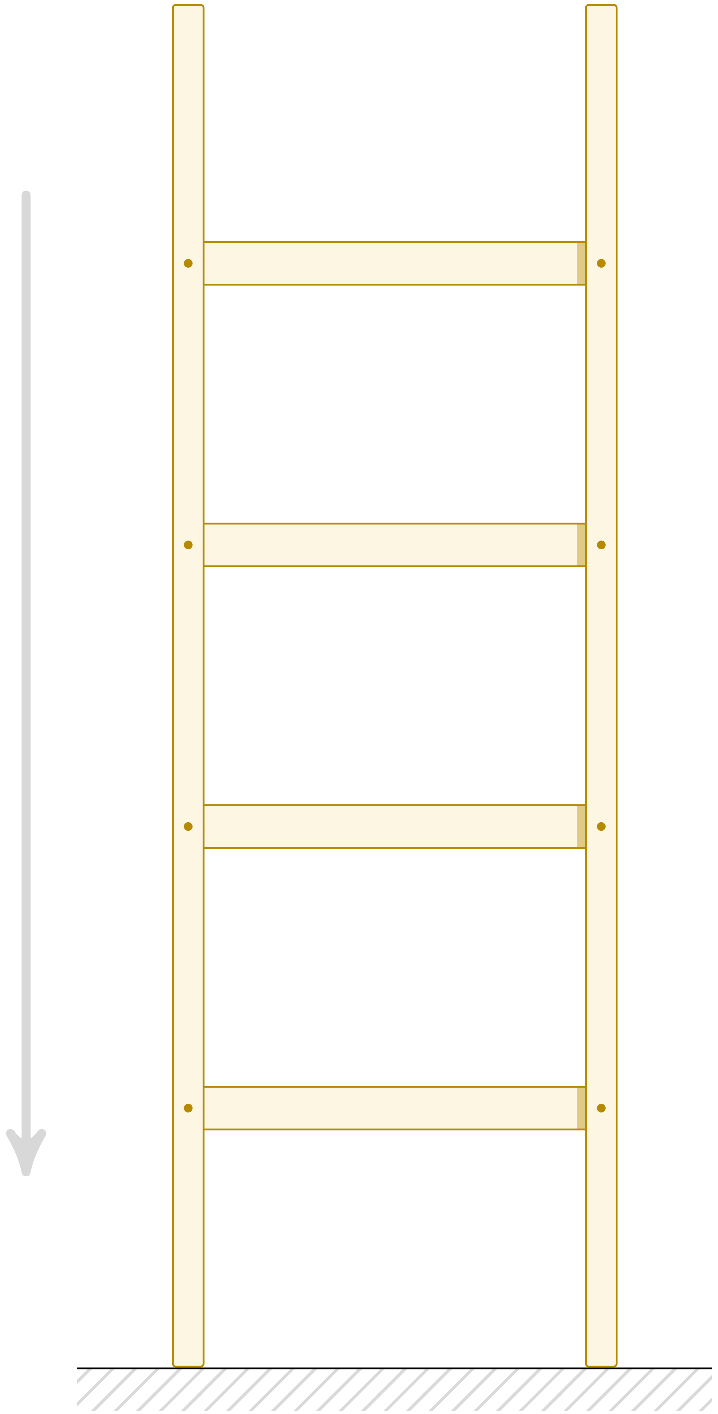
Forelesningen filmes



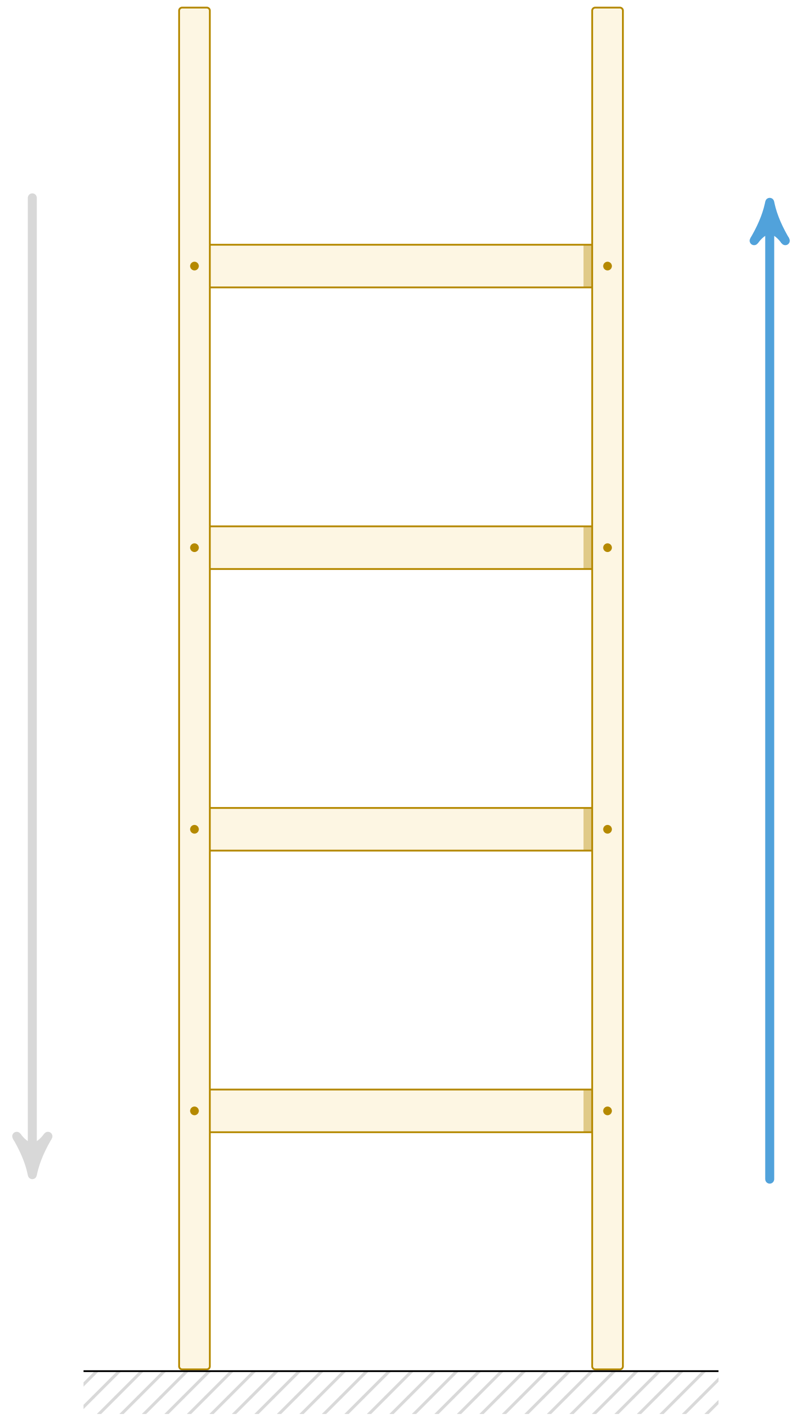
Dette er den velkjente induksjonsstigen vår



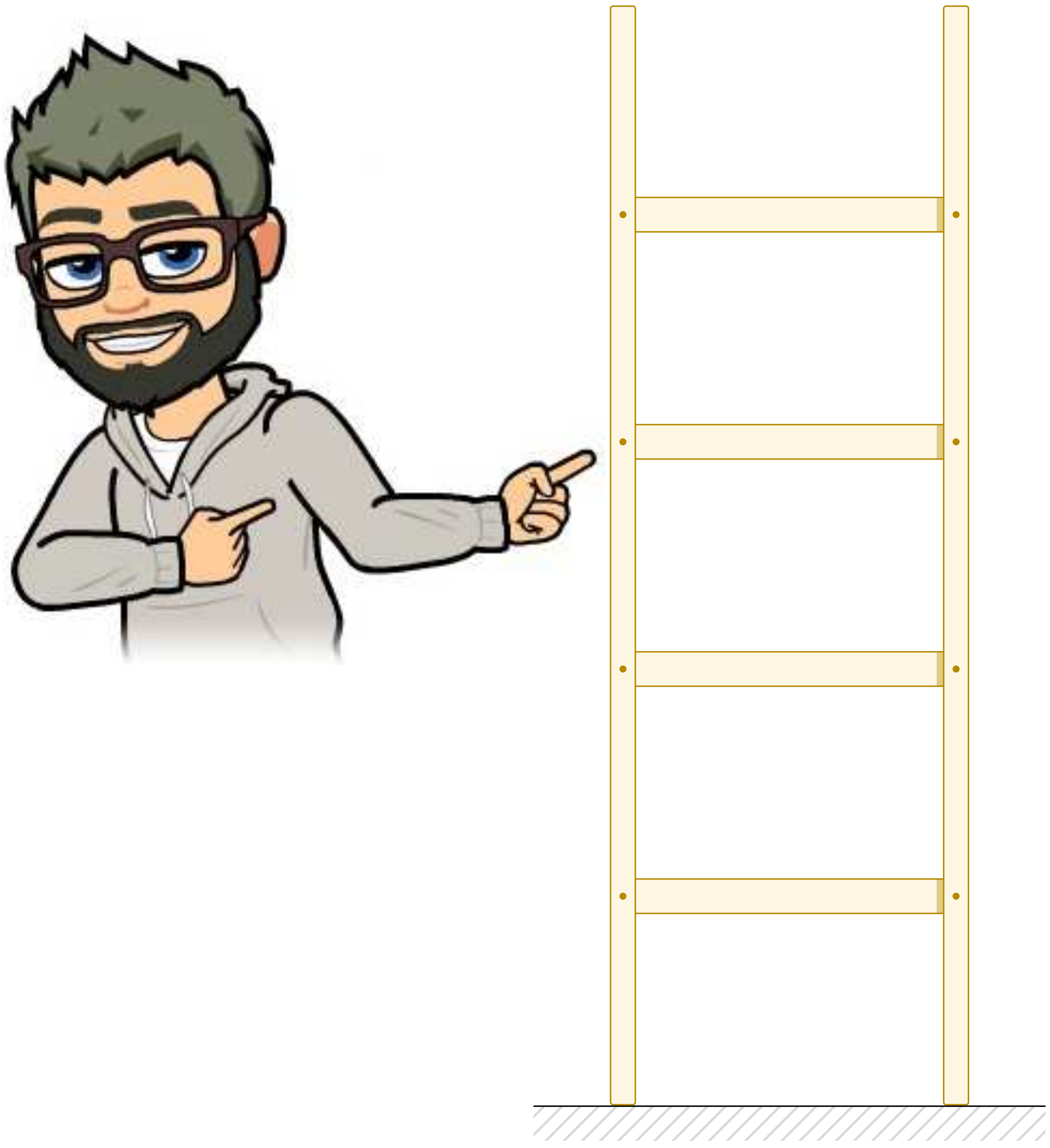
Vi reduserer en instans til mindre instanser . . .



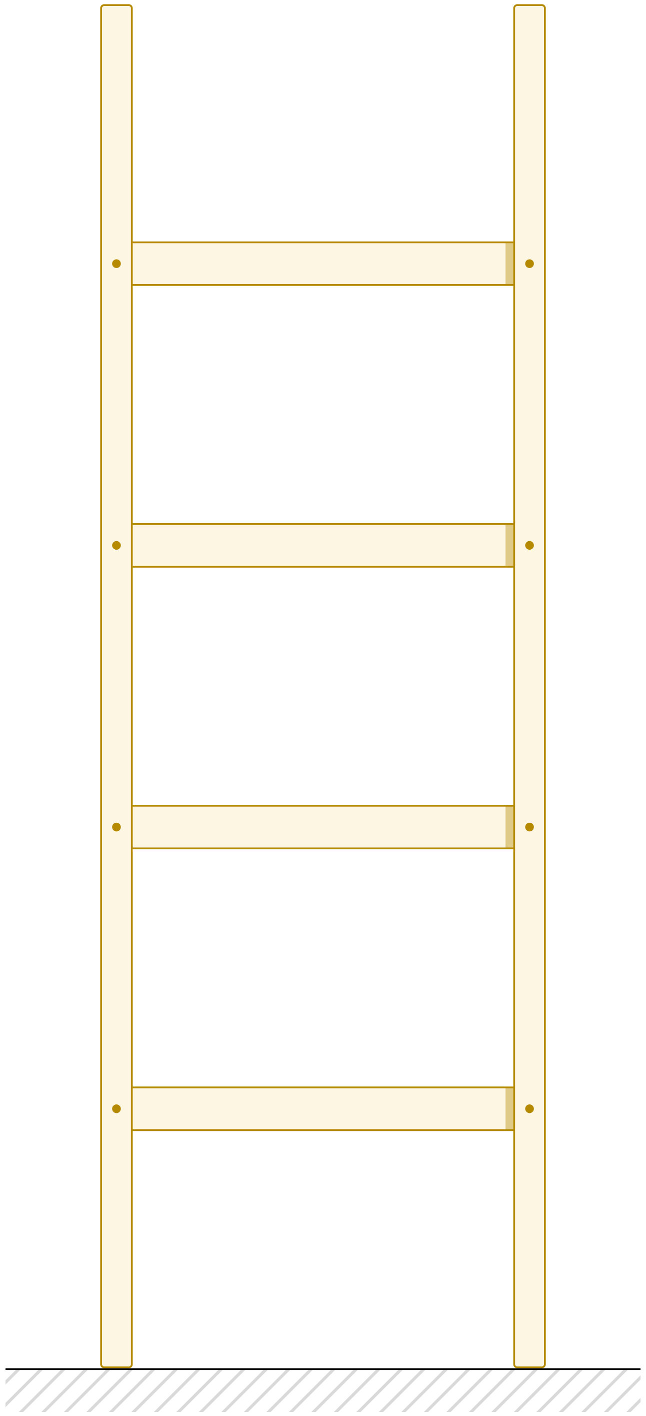
. . . til vi treffer grunntilfellet



Vi løser grunntilfellet og bygger vi oss opp igjen

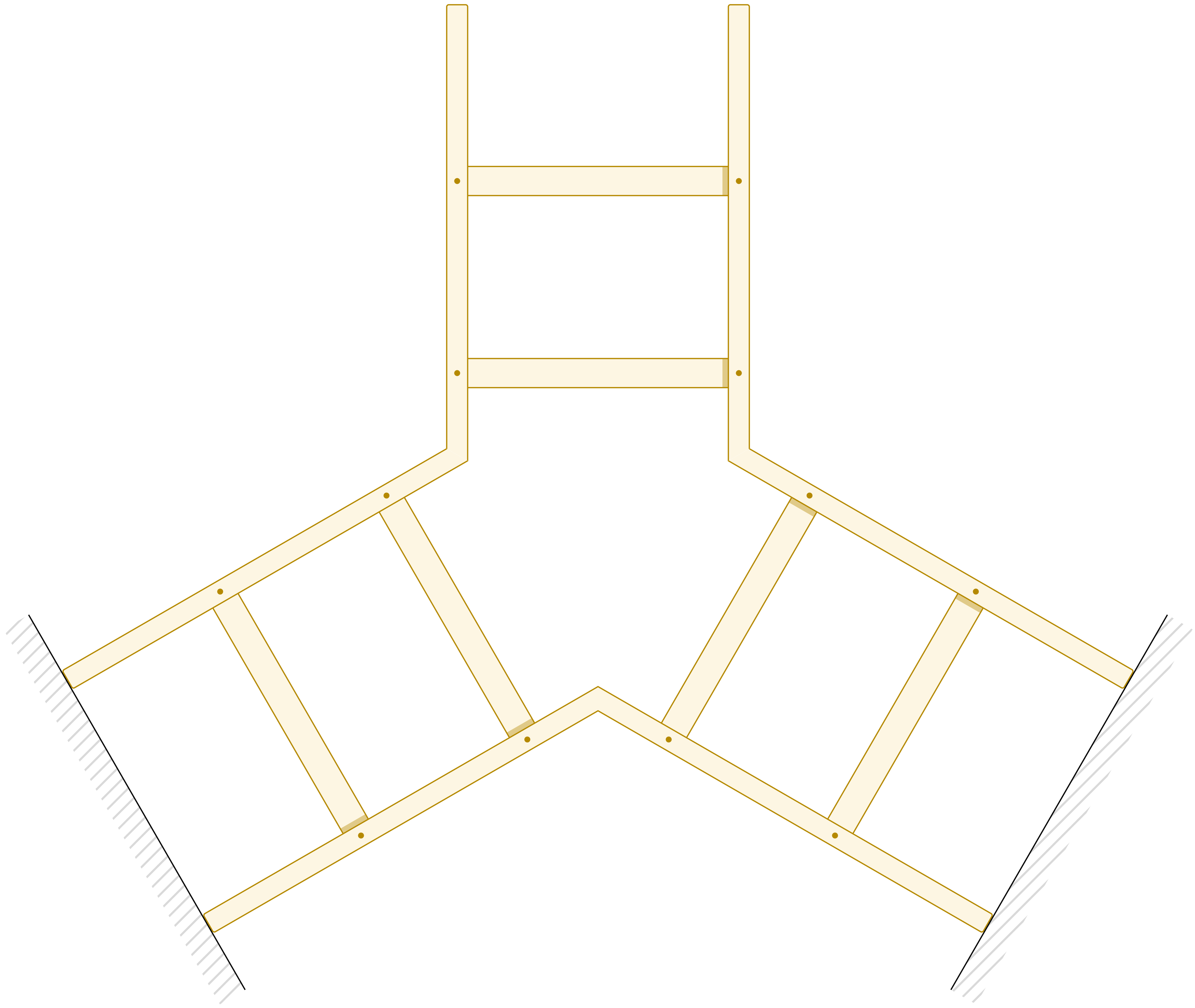


Alle induktive trinn er like, så ser bare på ett av dem



Men vi kan også spalte i flere delinstanser

Men vi kan også spalte i flere delinstanser



Denne designmetoden kalles «splitt og hersk» (*divide & conquer*)

Forelesning

3

Splitt & hersk

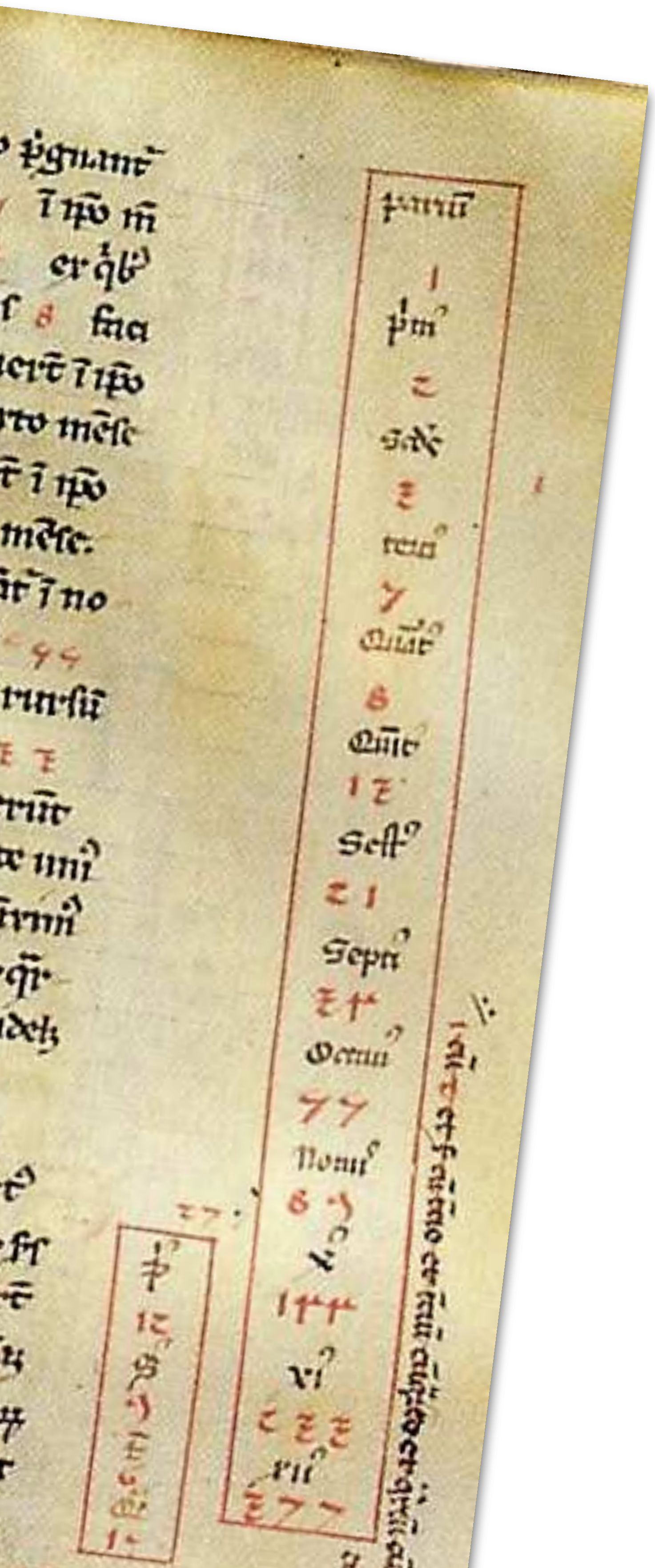
1. **Rekurrenser**
2. **Binærsøk**
3. **Merge sort**
4. **Quicksort**
5. **Masterteoremet**
6. **Variabelskifte**

4.1: Selvstudium

Én ting er at vi her regner på kjøretider, men merk at strukturen til utregningene og bevisene er den samme som for algoritmedesign og korrekthetsbevis – det er bare «innholdet» som er forskjellig. Det å forstå rekurrensregning kan være nyttig for å bedre forstå rekursiv dekomponering og algoritmedesign generelt – ikke bare for kjøretidsberegninger.

1:6

Rekurrenser



Fibonacci-sekvensen (fra Liber Abaci, 1228)

- **Rekurrens: Rekursiv ligning**
- **Beskriver f.eks. kjøretiden til rekursive algoritmer**
- **Må kvitte oss med den rekursive biten**

- **Finn løsning:**
  - **Iterasjonsmetoden**
  - **Rekurrenstrær**
  - **Masterteoremet (senere i dag)**
- **Verifisér løsning:**
  - **Substitusjon (dvs. induksjon)**

Iterasjonsmetoden står ikke oppført som læringsmål (en forglemmelse!) – men den er pensum (og står også oppført som pensum).

«Iterasjonsmetoden»: Gjentatt ekspandering av den rekursive forekomsten av funksjonen – det gir oss en sum som vi kan regne ut.

$$T(1) = 1$$

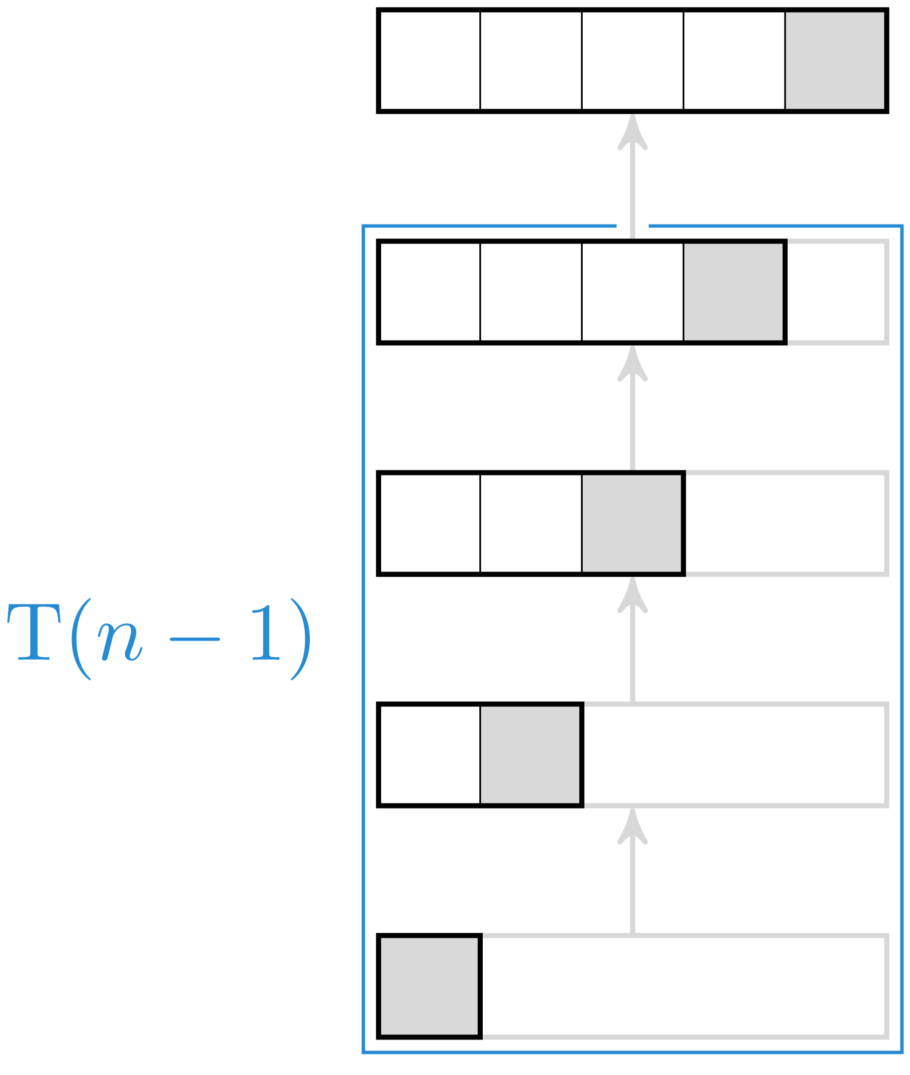
Antar stort sett dette (evt. $T(1) = O(1)$, e.l.)

Eksempel: $1 + 1 + \ldots$

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

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

Ett rekursivt kall (størrelse $n - 1$) pluss én operasjon



Rekurrens-tre

$$\begin{aligned} T(n) &= 1 \\ &\quad + T(n-1) \end{aligned} \tag{1}$$

$$
\begin{aligned}
T(n) &= 1 \\
&\quad + T(n-1) \qquad (1)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= 1 \\
&\quad + 1 && (1) \\
&\quad + \mathrm{T}(n - 1 - 1) && (2)
\end{aligned}
$$

$$\begin{aligned} \mathrm{T}(n) &= 1 \\ &+ 1 \qquad (1) \\ &+ \mathrm{T}(n-2) \qquad (2) \end{aligned}$$

$$
\begin{aligned}
T(n) &= 1 \\
&\quad + 1 && (1) \\
&\quad + 1 && (2) \\
&\quad + T(n-3) && (3)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= 1 \\
&\quad + 1 && (1) \\
&\quad + 1 && (2) \\
&\quad + 1 && (3) \\
&\quad \vdots && \vdots \\
&\quad + \mathrm{T}(n - ?) && (?)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= 1 \\
&\quad + 1 && \color{blue}{(1)} \\
&\quad + 1 && \color{blue}{(2)} \\
&\quad + 1 && \color{blue}{(3)} \\
&\quad \vdots && \color{blue}{\vdots} \\
&\quad + \mathrm{T}(n - ?) && \color{blue}{(n-1)}
\end{aligned}
$$

$$
\begin{aligned}
T(n) = 1 & \\
+\,1 & \qquad (1) \\
+\,1 & \qquad (2) \\
+\,1 & \qquad (3) \\
\vdots\;\, & \qquad \vdots \\
+\,T(n-(n-1)) & \qquad (n-1)
\end{aligned}
$$

$$
\begin{aligned}
T(n) = 1 & \\
+\,1 & \qquad (1) \\
+\,1 & \qquad (2) \\
+\,1 & \qquad (3) \\
\vdots\; & \qquad \vdots \\
+\,T(1) & \qquad (n-1)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= 1 \\
&\quad + 1 && \color{blue}{(1)} \\
&\quad + 1 && \color{blue}{(2)} \\
&\quad + 1 && \color{blue}{(3)} \\
&\quad\ \vdots && \color{blue}{\vdots} \\
&\quad + 1 && \color{blue}{(n-1)}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= 1 \\
&\quad + 1 && (1) \\
&\quad + 1 && (2) \\
&\quad + 1 && (3) \\
&\quad \vdots && \vdots \\
&\quad + 1 && (n-1)
\end{aligned}
$$

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

Verifikasjon

Med substitusjon/induksjon

Her må vi også huske å verifisere at grunntilfellet (f.eks. T(1)=1) stemmer med løsningen vår. Det har jeg ikke eksplisitt diskutert her, men det stemmer for de eksemplene vi ser på.

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

Denne løsningen har vi kommet frem til litt uformelt – vi kan verifisere den med induksjon. Det kalles «substitusjonsmetoden», og den kan også brukes om vi bare *gjetter* svaret.

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

Gitt antagelsen $\mathrm{T}(n-1) = n-1$, vis at $\mathrm{T}(n) = n$

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

Gitt antagelsen $\mathrm{T}(n-1) = n-1$, vis at $\mathrm{T}(n) = n$

$$
\begin{aligned}
\mathrm{T}(n) &= \mathrm{T}(n-1) + 1 \\
&= (n-1) + 1
\end{aligned}
$$

Gitt antagelsen $\mathrm{T}(n-1) = n-1$, vis at $\mathrm{T}(n) = n$

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

$$= (n-1) + 1$$

$$= n - 1 + 1$$

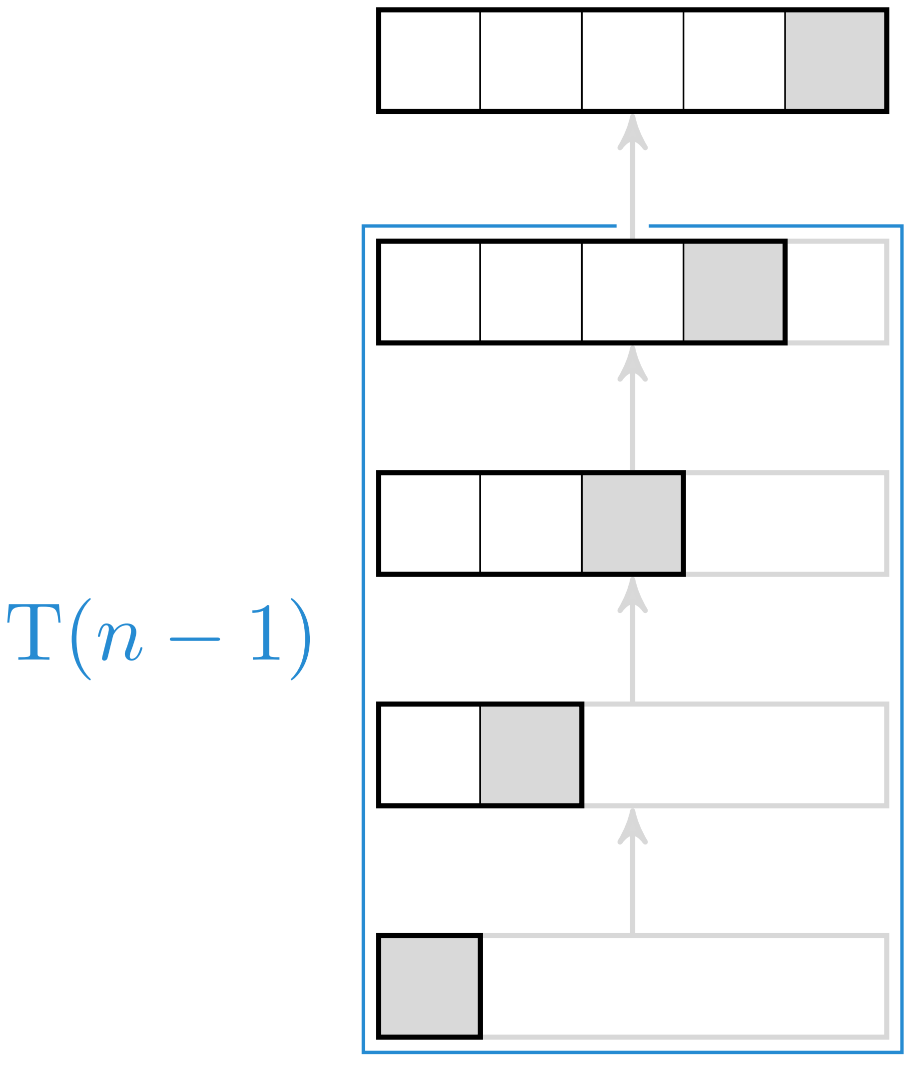
Gitt antagelsen $\mathrm{T}(n-1) = n-1$, vis at $\mathrm{T}(n) = n$

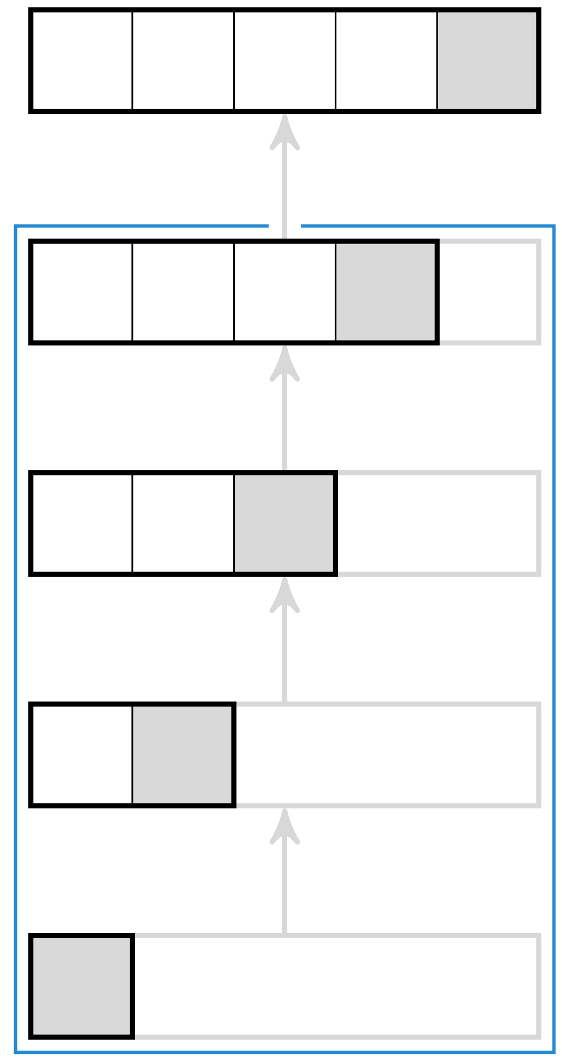
$$
\begin{aligned}
T(n) &= T(n-1) + 1 \\
&= (n-1) + 1 \\
&= n - 1 + 1 \\
&= n
\end{aligned}
$$

Gitt antagelsen $T(n-1) = n-1$, vis at $T(n) = n$

$$
\begin{aligned}
\mathrm{T}(n) &= \mathrm{T}(n-1) + 1 \\
&= (n-1) + 1 \\
&= n - 1 + 1 \\
&= n
\end{aligned}
$$

Gitt antagelsen $\mathrm{T}(n-1) = n - 1$, vis at $\mathrm{T}(n) = n$


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



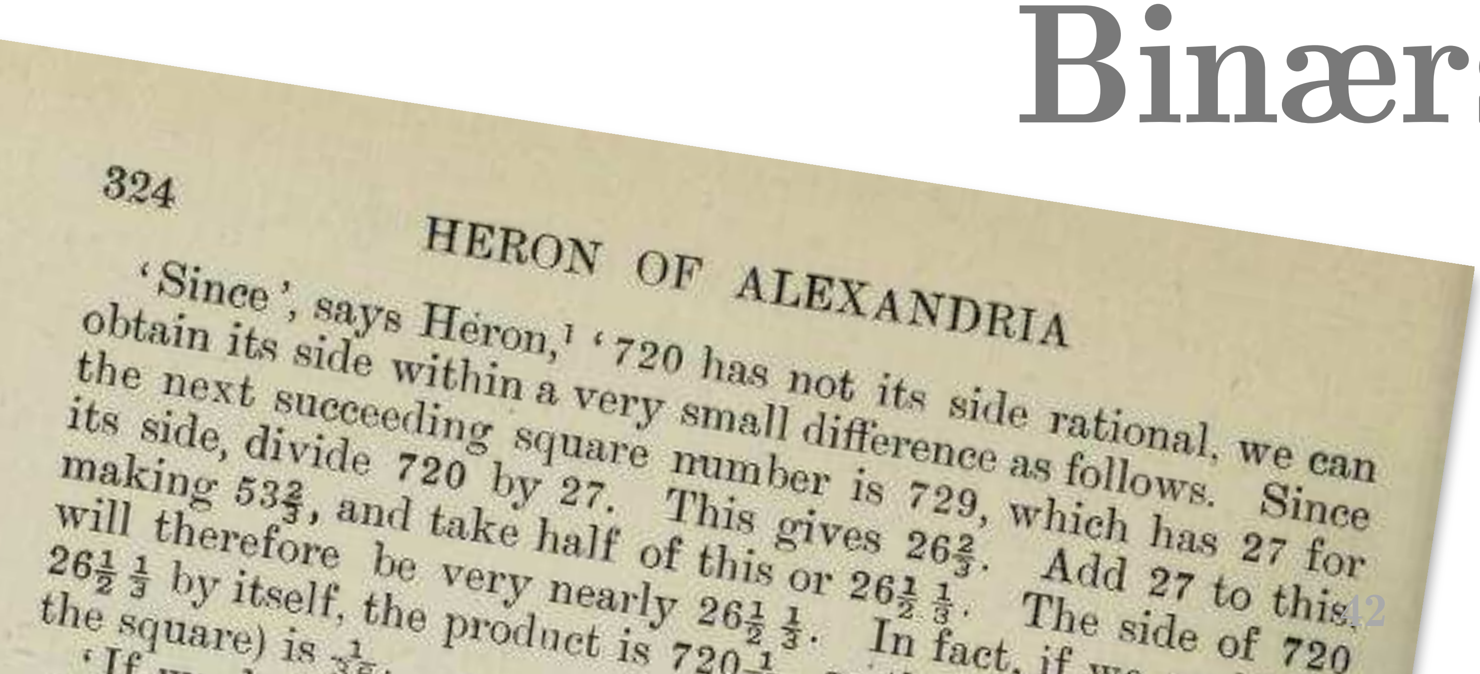
$T(n-1)$

$$T(n) = n$$

Her bruker vi strengt tatt fortsatt bare én delinstans – men strategien med å halvere instansen er en vanlig ingrediens i mer typiske D&C-algoritmer (som vi straks skal se på).

2:6

Binærsøk


324

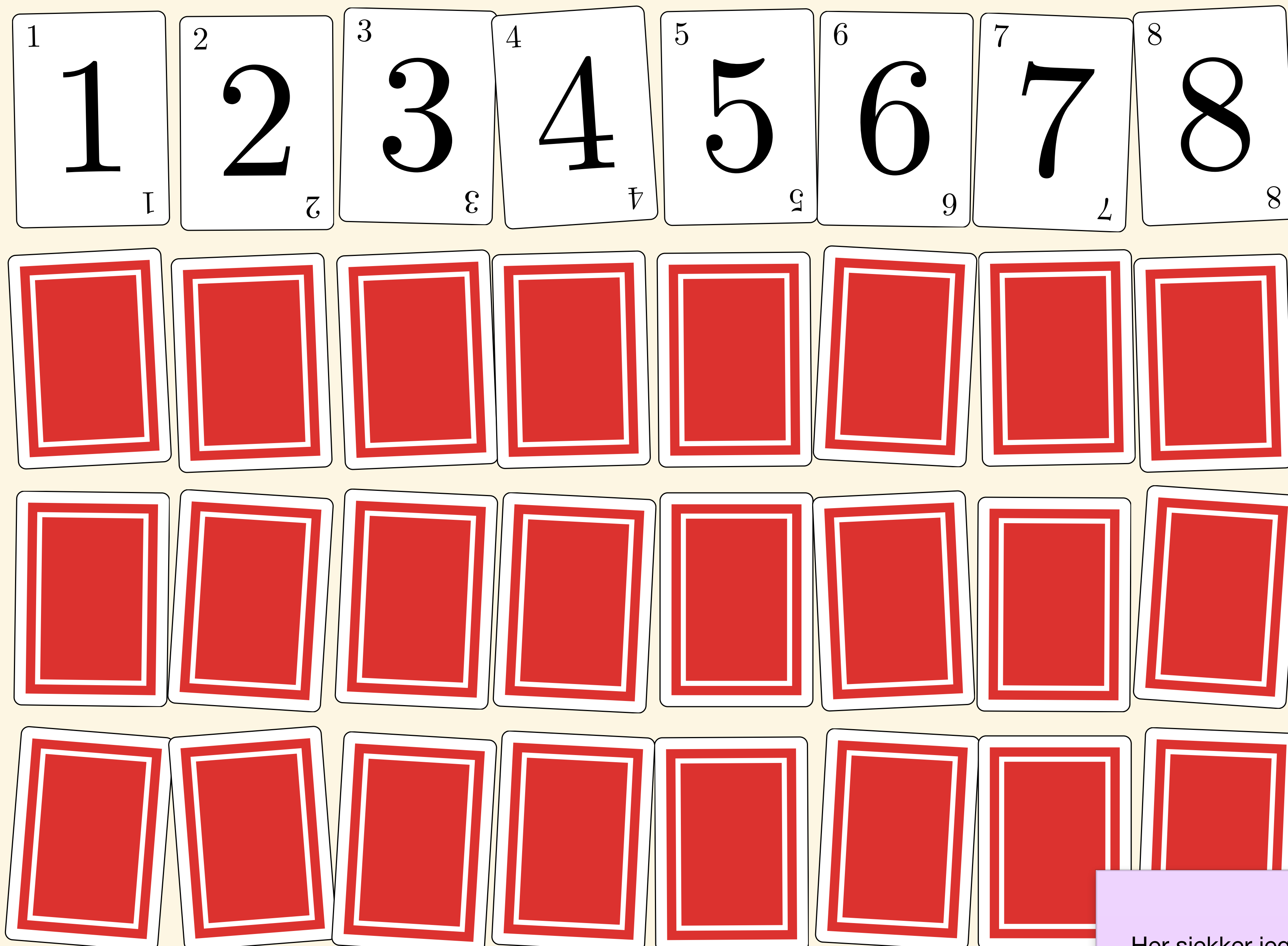
HERON OF ALEXANDRIA

'Since', says Heron,[1] '720 has not its side rational, we can obtain its side within a very small difference as follows. Since the next succeeding square number is 729, which has 27 for its side, divide 720 by 27. This gives $26\frac{2}{3}$. Add 27 to this, making $53\frac{2}{3}$, and take half of this or $26\frac{1}{2}\,\frac{1}{3}$. The side of 720 will therefore be very nearly $26\frac{1}{2}\,\frac{1}{3}$. In fact, if we multiply $26\frac{1}{2}\,\frac{1}{3}$ by itself, the product is $720\frac{1}{36}$ ... the square) is $\frac{1}{36}$ ...

Sir Thomas Little Heath sin beskrivelse av Herons metode for å approksimere kvadratrøtter ved hjelp av en nær slektning av binærsøk – en metode som antagelig ble brukt allerede av babylonerne.

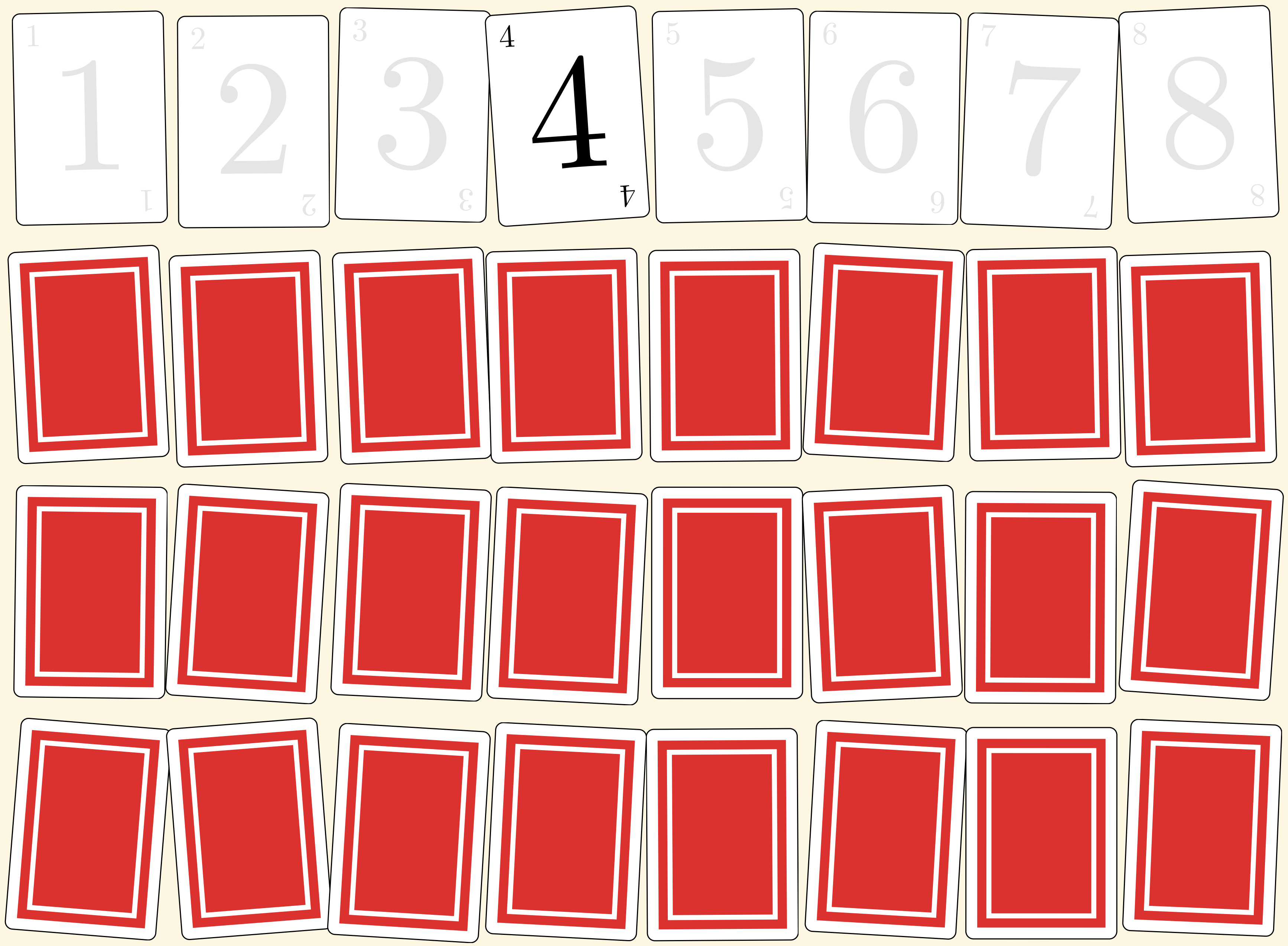
(Heath, A History of Greek Mathematics, vol. 2, 1921; Fowler & Robson, Square Root Approximations in Old Babylonian Mathematics: YBC 7289 in Context, 1998)

- **Sortert sekvens**
- **Er det du leter etter i første halvdel?**
  - **Hvis ja: Let videre der**
  - **Hvis nei: Let i den andre halvdelen**

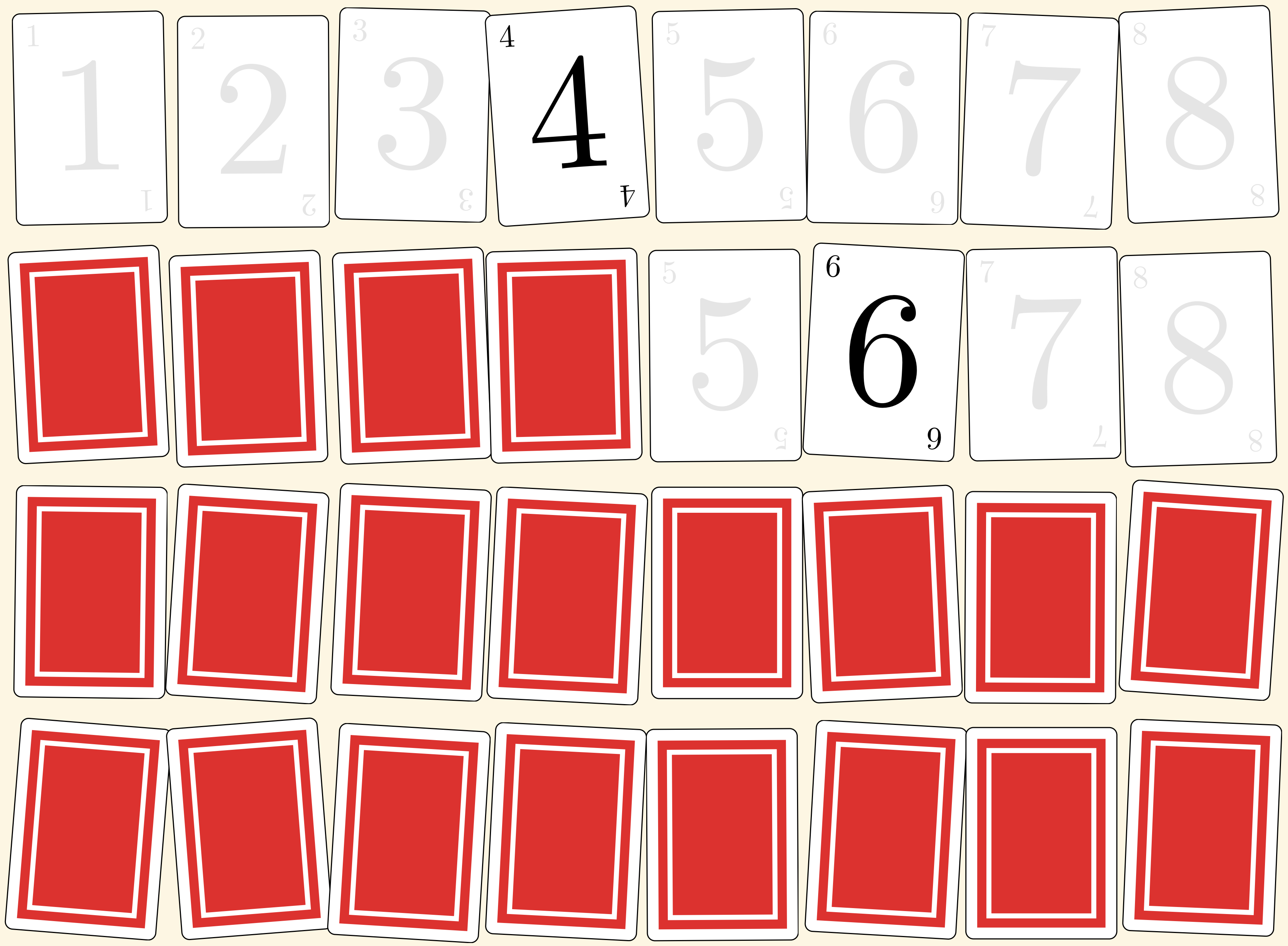


Her sjekker jeg bare hvilken halvdel jeg skal lete i – og sjekker ikke om kortet vi ser på er det vi er ute etter før vi kommer til bunnen. I pseudokoden senere har vi den sjekken i tillegg.

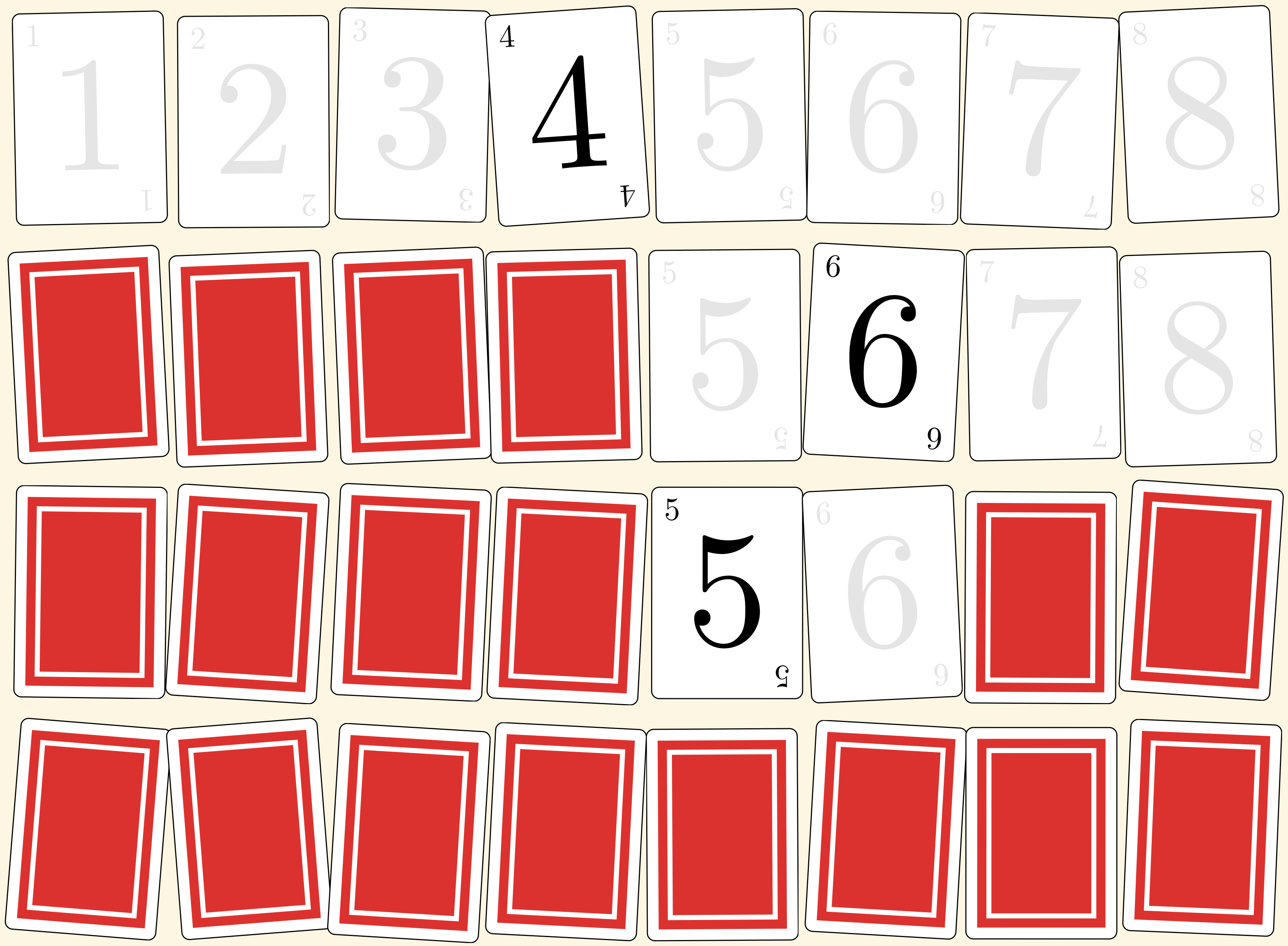
(Anta at kortet ditt er 6)



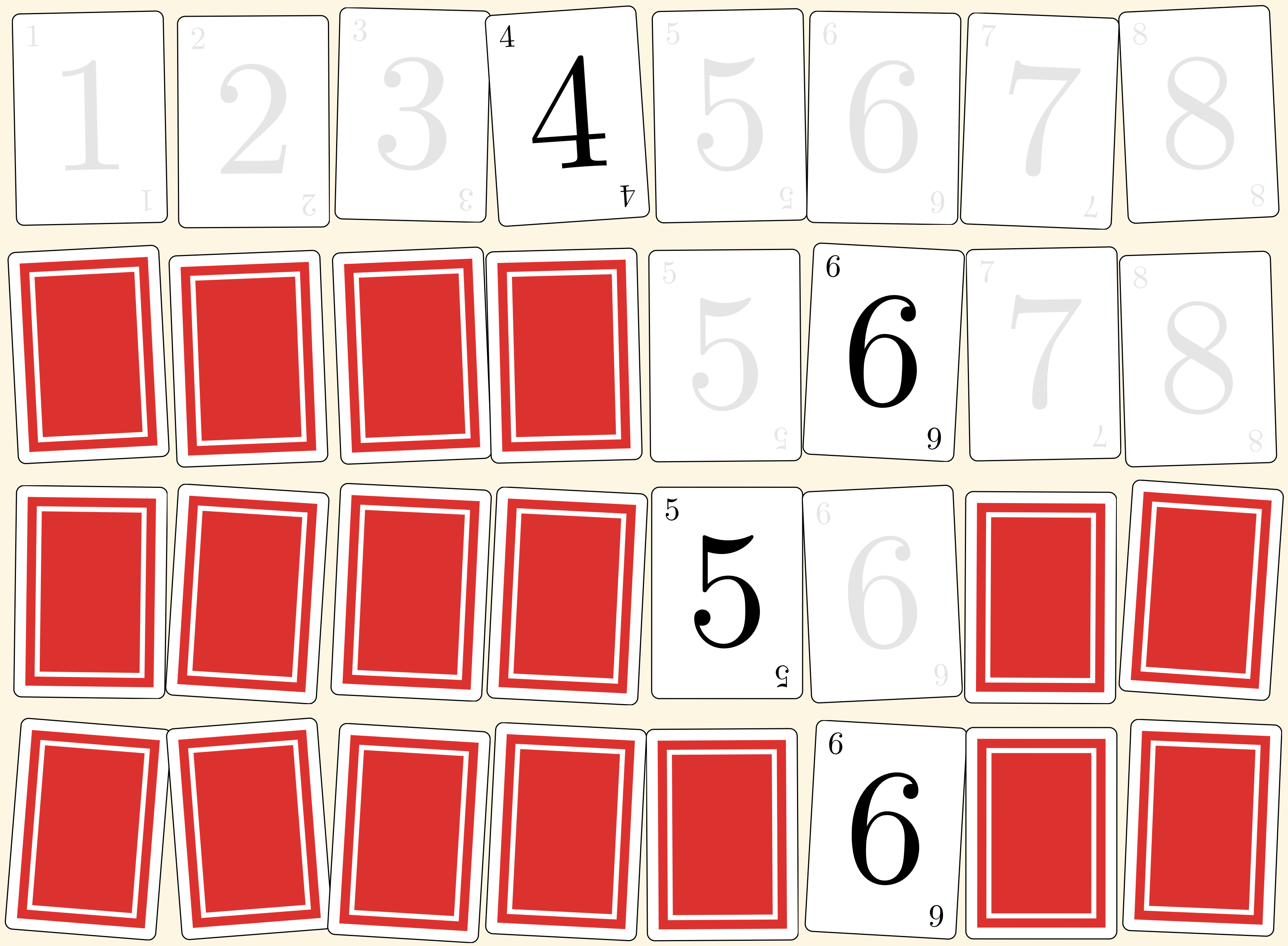
Er kortet større enn A[4]?



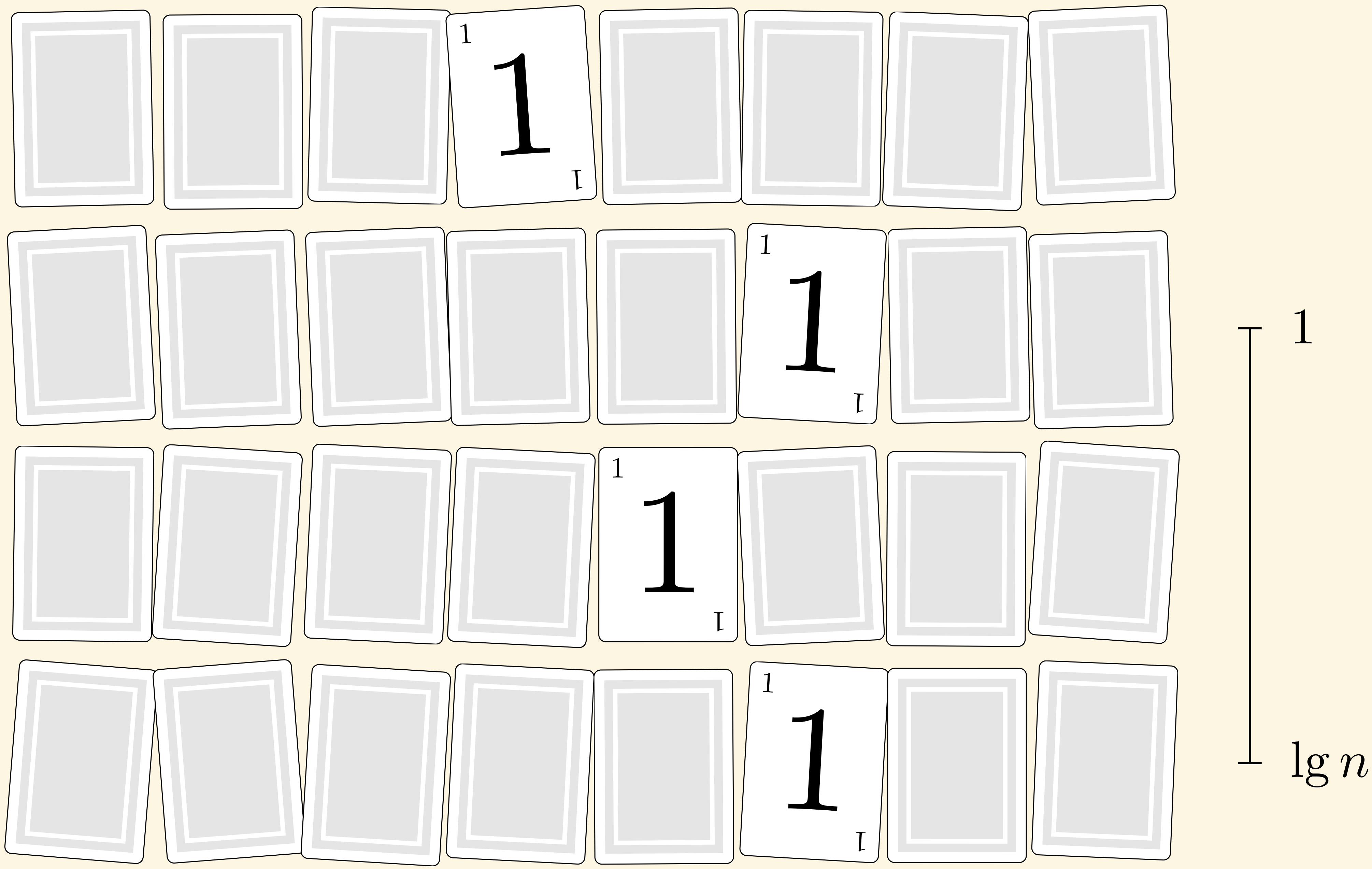
Er kortet større enn A[6]?



Er kortet større enn A[5]?



Er A[6] kortet ditt?



Antall spørsmål: $\lg n + 1$

$\textsc{Bisect}(\mathrm{A}, p, r, v)$

A	tabell
p	venstre
r	høyre
v	søkeverdi

A er sortert. Om mulig, finn en indeks i så $\mathrm{A}[i] == v$

$\textsc{Bisect}(A, p, r, v)$

1 **if** $p \leq r$

A	tabell
p	venstre
r	høyre
v	søkeverdi

Har vi i det hele tatt et segment å lete i?

$\textsc{Bisect}(\mathrm{A}, p, r, v)$
1 **if** $p \leq r$
2 $\quad q = \lfloor (p + r)/2 \rfloor$

A tabell
p venstre
r høyre
v søkeverdi
q midten

Del på midten

$\textsc{Bisect}(A, p, r, v)$

```
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
```

A	tabell
p	venstre
r	høyre
v	søkeverdi
q	midten

Er dette verdien vi leter etter?

$\textsc{Bisect}(\mathrm{A}, p, r, v)$

```
1  if p ≤ r
2      q = ⌊(p + r)/2⌋
3      if v == A[q]
4          return q
```

A	tabell
p	venstre
r	høyre
v	søkeverdi
q	midten

I så fall er vi ferdige

Bisect(A, p, r, v)

if $p \leq r$
  $q = \lfloor (p + r)/2 \rfloor$
  if $v == A[q]$
    return q
  elseif $v < A[q]$

A	tabell
p	venstre
r	høyre
v	søkeverdi
q	midten

Ellers: Må v ligge til venstre?

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
```

A	tabell
p	venstre
r	høyre
v	søkeverdi
q	midten

Let videre til venstre

BISECT(A, p, r, v)

if $p \leq r$

  $q = \lfloor (p + r)/2 \rfloor$

  if $v == A[q]$

    return q

  elseif $v < A[q]$

    return BISECT$(A, p, q - 1, v)$

  else return BISECT$(A, q + 1, r, v)$

A	tabell
p	venstre
r	høyre
v	søkeverdi
q	midten

Ellers må v ligge til høyre, så vi leter videre der

$\textsc{Bisect}(A, p, r, v)$

if $p \leq r$

  $q = \lfloor (p + r)/2 \rfloor$

  if $v == A[q]$

    return q

  elseif $v < A[q]$

    return $\textsc{Bisect}(A, p, q - 1, v)$

  else return $\textsc{Bisect}(A, q + 1, r, v)$

return NIL

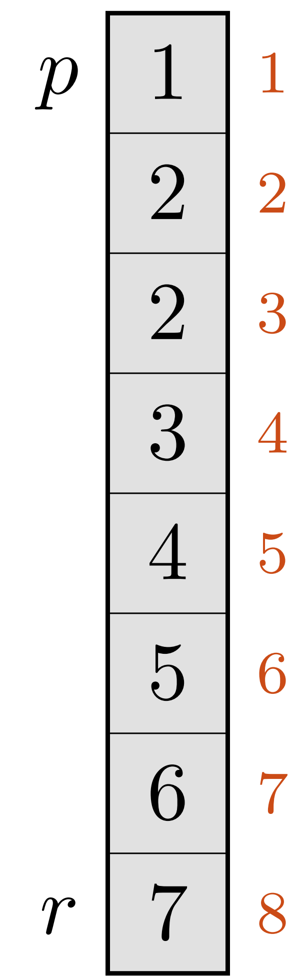
A	tabell
p	venstre
r	høyre
v	søkeverdi
q	midten

Vi fant ikke v, så vi returnerer NIL

mlh@ntnu.no

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```

$p, r, v \ = \ 1, 8, 4$



```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```

$p, r, v \;=\; 1, 8, 4$

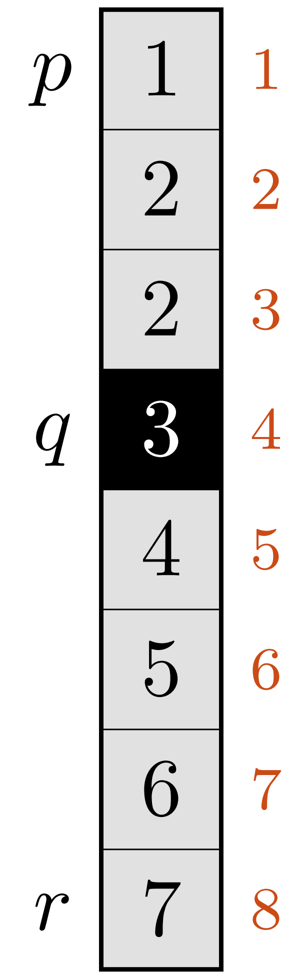
p	1	1
	2	2
	2	3
	3	4
	4	5
	5	6
	6	7
r	7	8

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```

$p, r, v \;=\; 1, 8, 4$

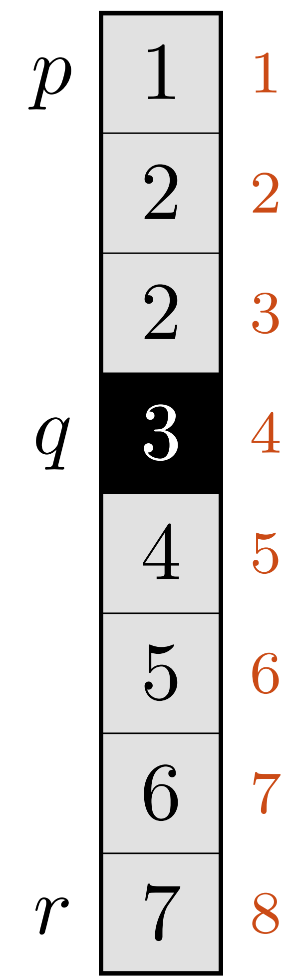
p	1	1
	2	2
	2	3
q	3	4
	4	5
	5	6
	6	7
r	7	8

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



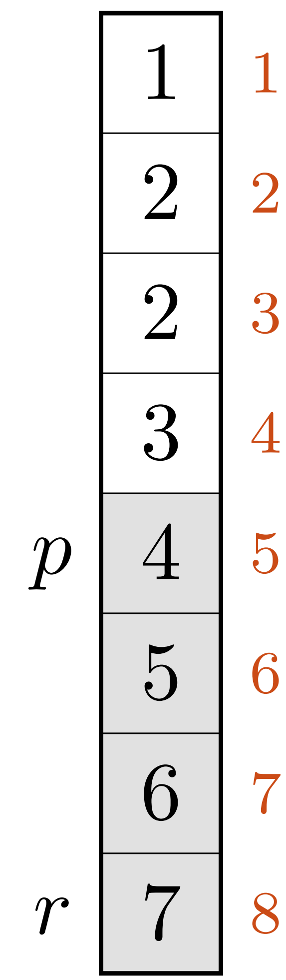
$p, r, v \;=\; 1, 8, 4$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



$p, r, v \;=\; 1, 8, 4$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



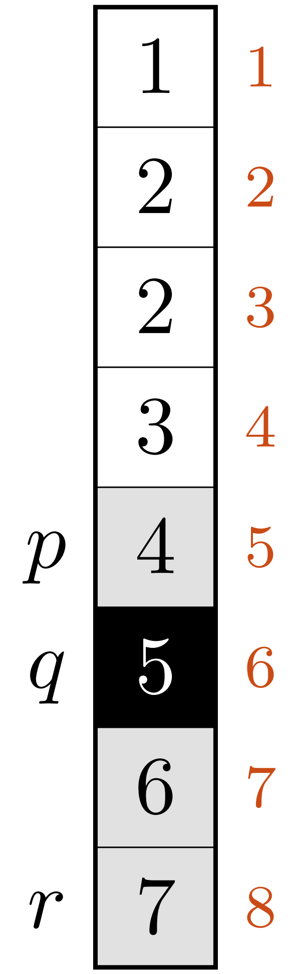
$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```

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

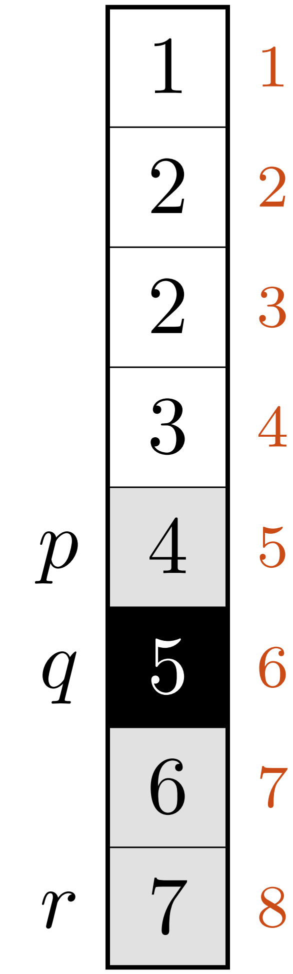
$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4$

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```



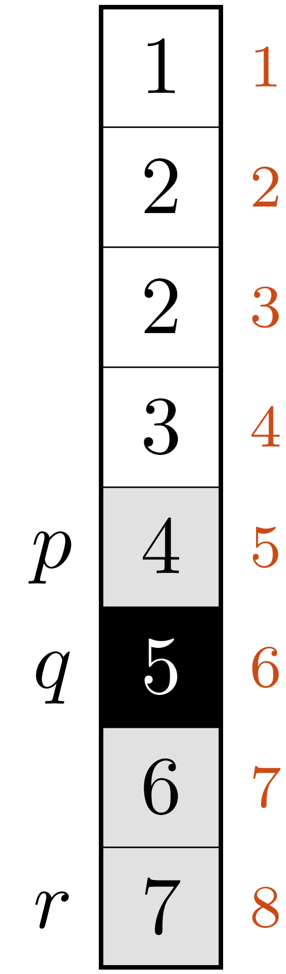
$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



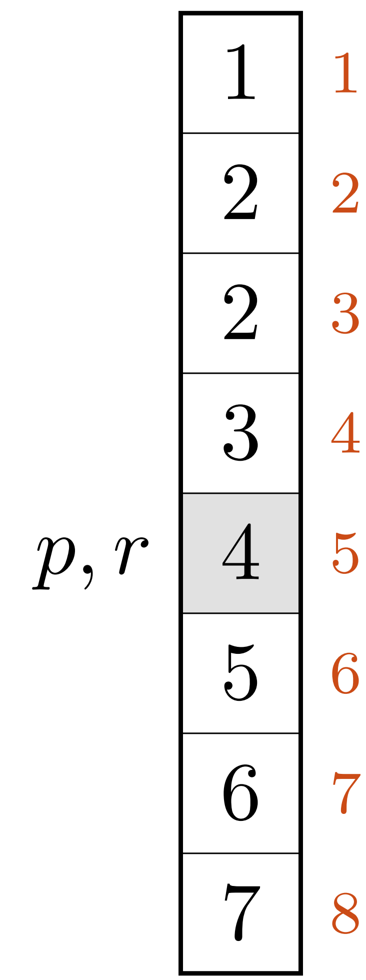
$p, r, v \; = \; 1, 8, 4 \; › \; 5, 8, 4$

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```



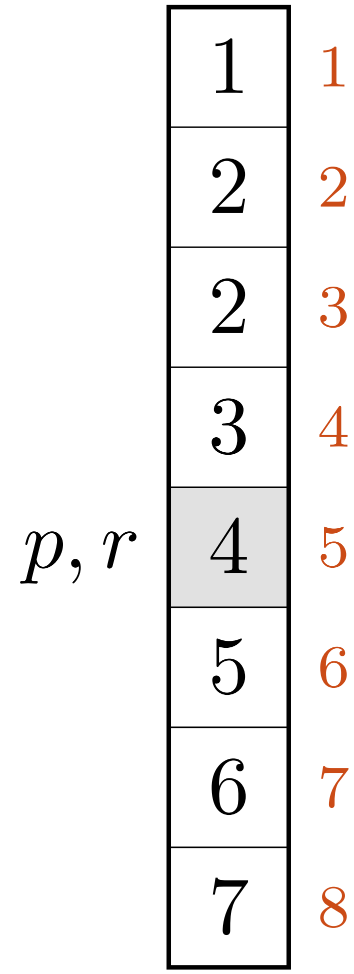
$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```

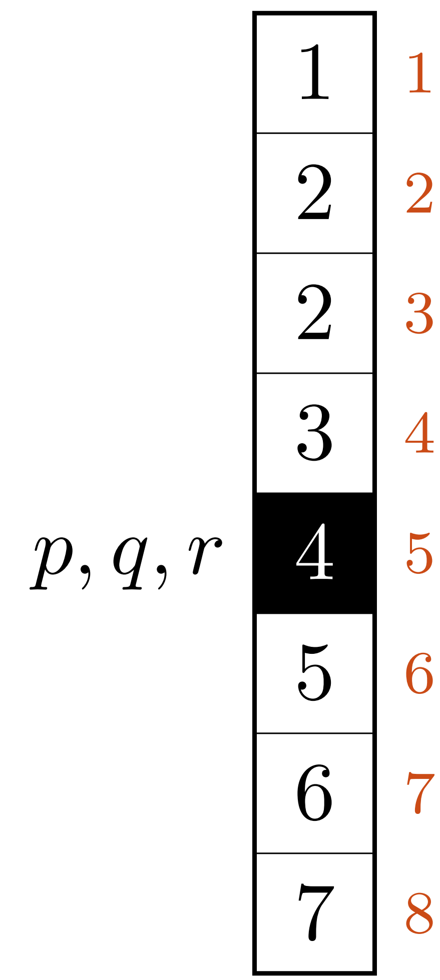


$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4 \;›\; \mathbf{5, 5, 4}$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



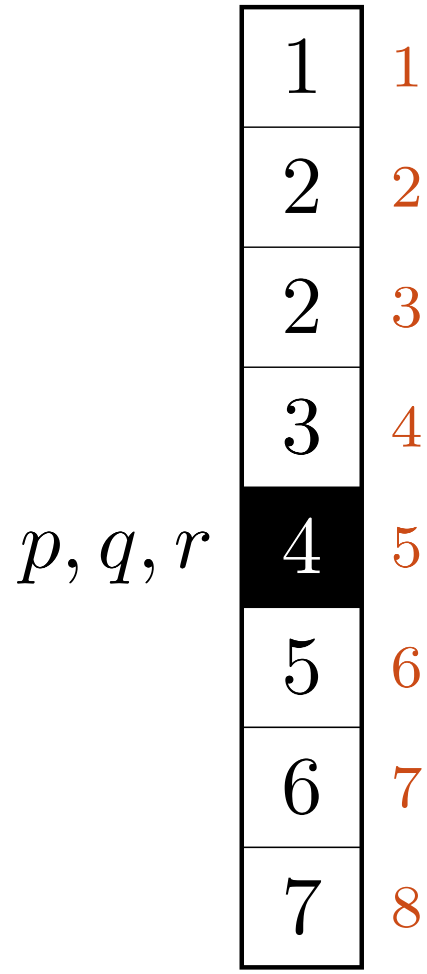
$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4 \;›\; \mathbf{5, 5, 4}$




```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```

$p, r, v \ = \ 1, 8, 4 \ \rangle \ 5, 8, 4 \ \rangle \ 5, 5, 4$

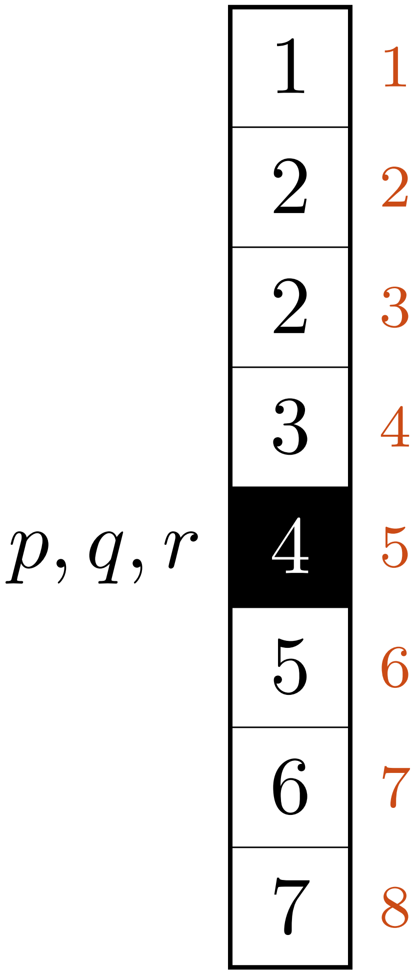
```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```



$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4 \;›\; \mathbf{5, 5, 4}$

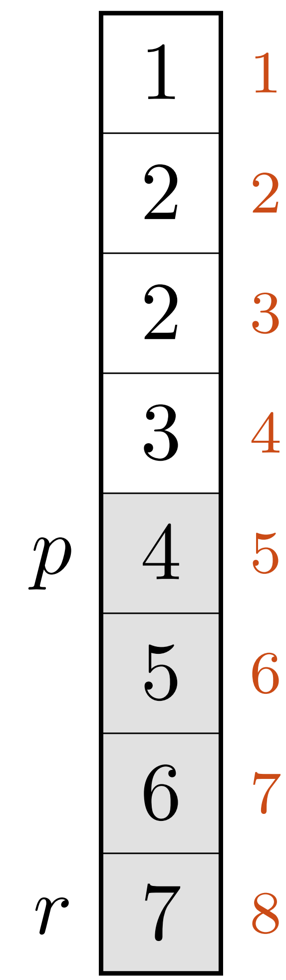
```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```

→ 5



$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4 \;›\; 5, 5, 4$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4$

Bisect(A, p, r, v)

if $p \le r$
  $q = \lfloor (p + r)/2 \rfloor$
  if $v == A[q]$
    return q
  elseif $v < A[q]$
    return Bisect$(A, p, q - 1, v)$
  else return Bisect$(A, q + 1, r, v)$
return NIL

$\rightarrow 5$

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

$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4$

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```

$p, r, v \;=\; 1, 8, 4$

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

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return NIL
```

$\rightarrow$ 5

$p, r, v \;=\; 1, 8, 4$

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

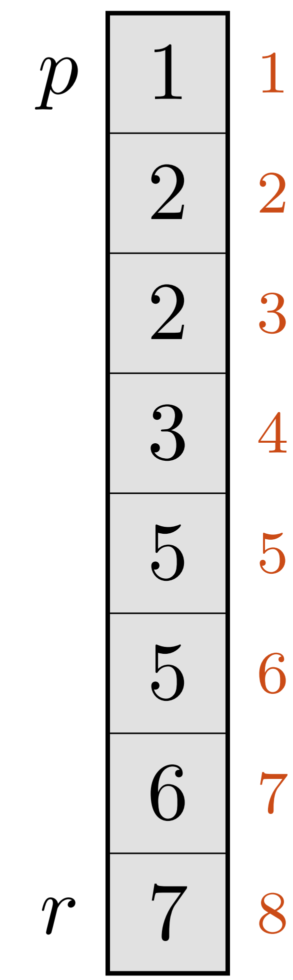
```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```

$p, r, v \;=\; 1, 8, 4$

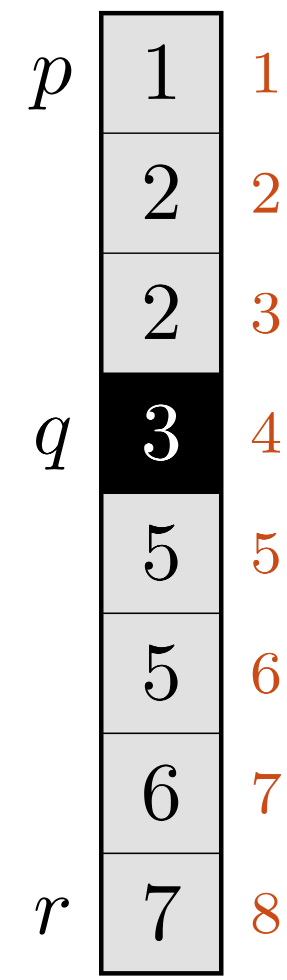
p	1	1
	2	2
	2	3
	3	4
	5	5
	5	6
	6	7
r	7	8

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```

$p, r, v \;=\; 1, 8, 4$



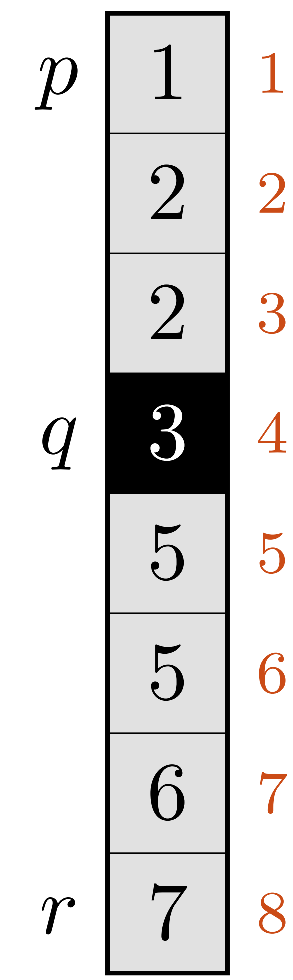
```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



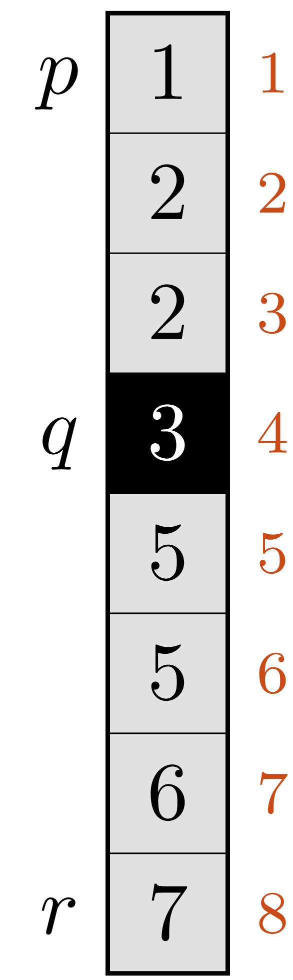
$p, r, v \;=\; 1, 8, 4$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```

$p, r, v \;=\; 1, 8, 4$

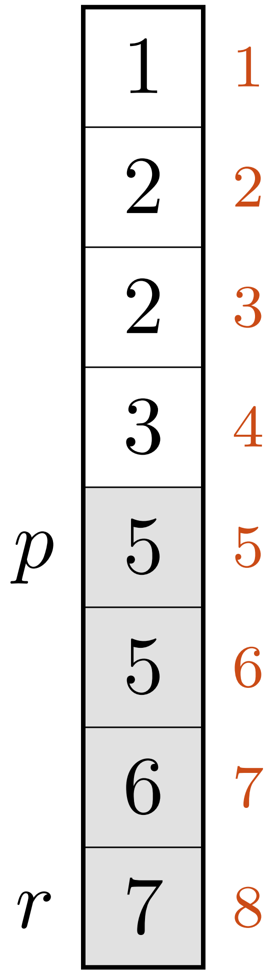


```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



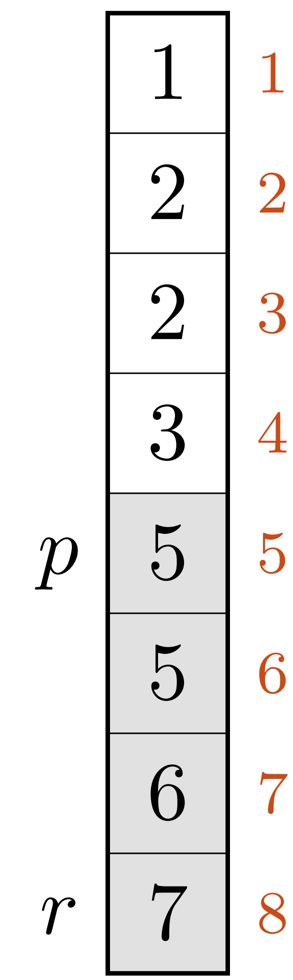
$p, r, v \; = \; 1, 8, 4$

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```



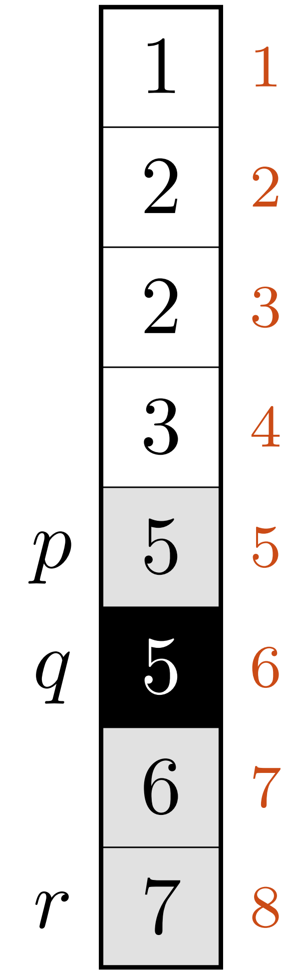
$p, r, v \;=\; 1, 8, 4 \;\rangle\; 5, 8, 4$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



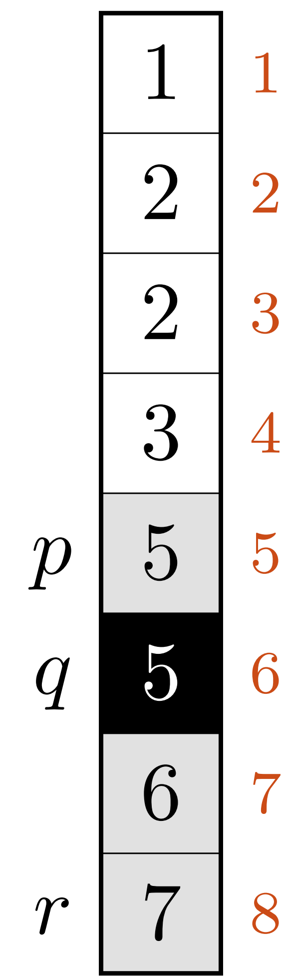
$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4$

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```



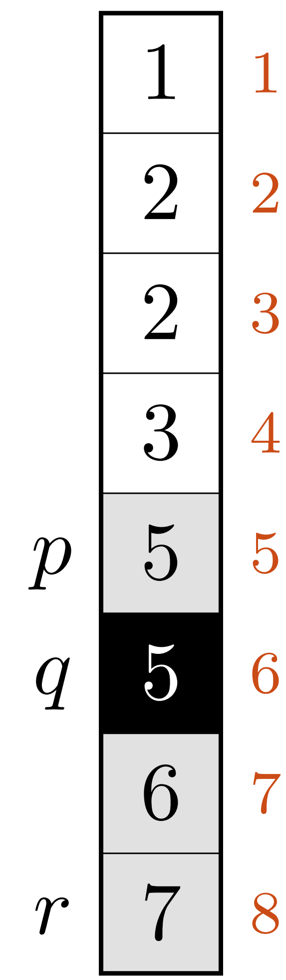
$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4$

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```



$p, r, v \;=\; 1, 8, 4 \;\rangle\; 5, 8, 4$

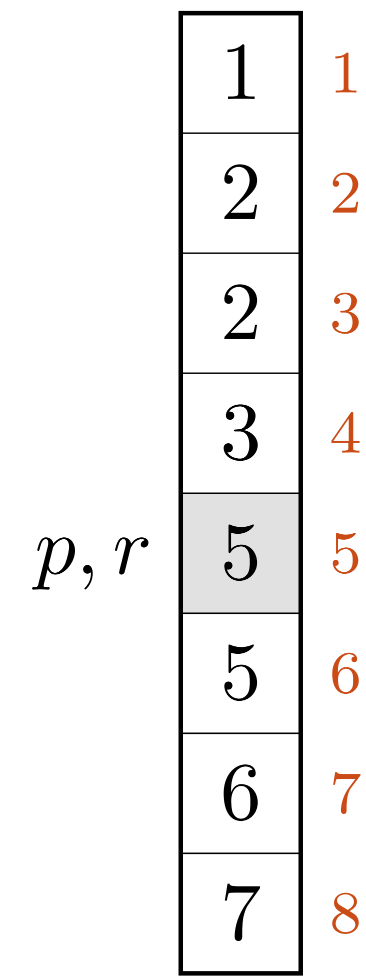
Bisect(A, p, r, v)
if $p \leq r$
  $q = \lfloor (p + r)/2 \rfloor$
  if $v == A[q]$
    return q
  elseif $v < A[q]$
    return Bisect(A, p, $q - 1$, v)
  else return Bisect(A, $q + 1$, r, v)
return nil



$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4$

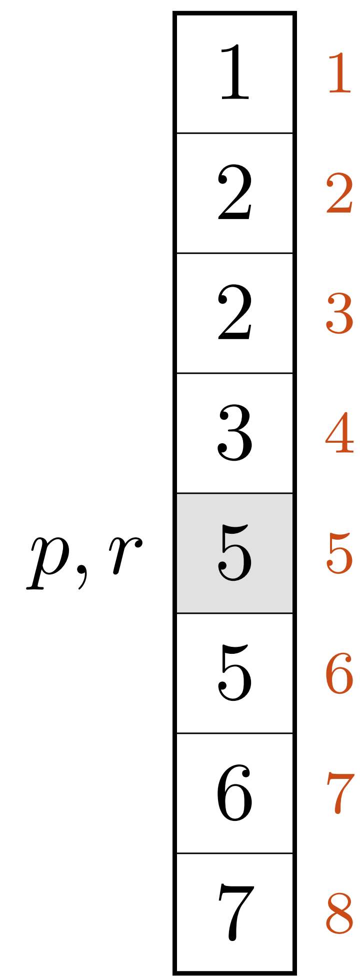
BISECT(A, p, r, v)

```
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



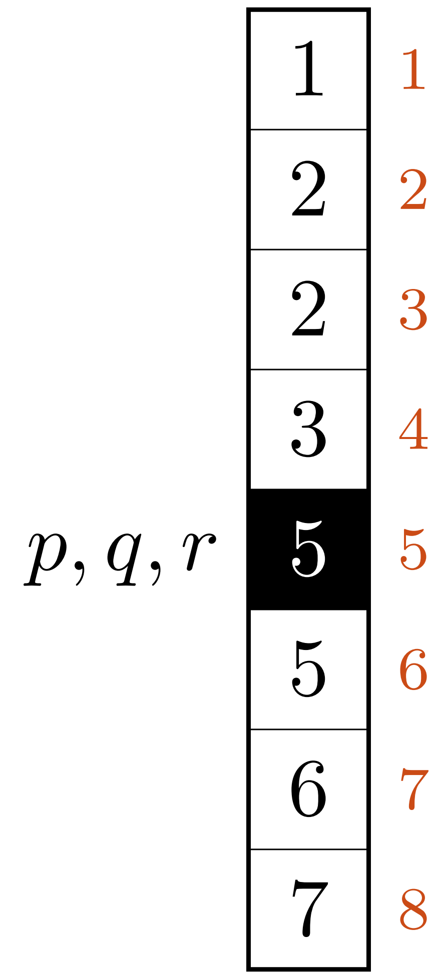
$p, r, v \;=\; 1, 8, 4$ › $5, 8, 4$ › $\mathbf{5, 5, 4}$

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```



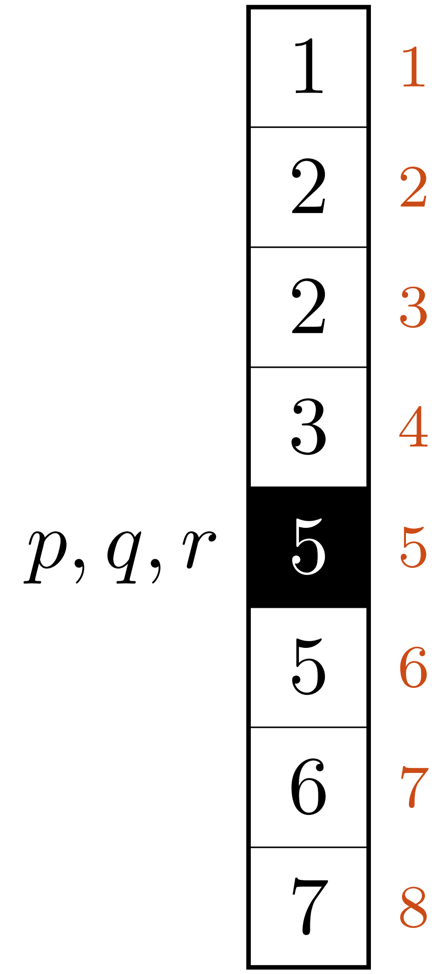
$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4 \;›\; \mathbf{5, 5, 4}$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



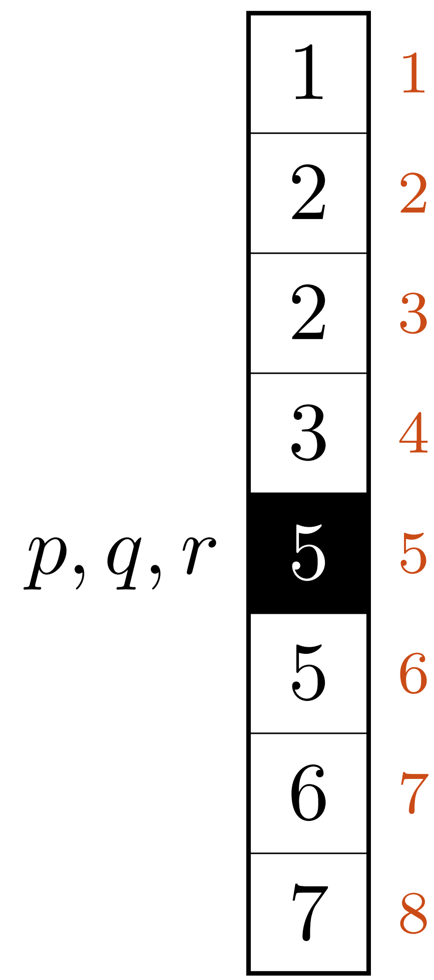
$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4 \;›\; 5, 5, 4$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```



$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4 \;›\; \mathbf{5, 5, 4}$

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```



$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4 \;›\; \mathbf{5, 5, 4}$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```

$p, r, v \ = \ 1, 8, 4 \ › \ 5, 8, 4 \ › \ 5, 5, 4 \ › \ \mathbf{5, 4, 4}$

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

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return NIL
```

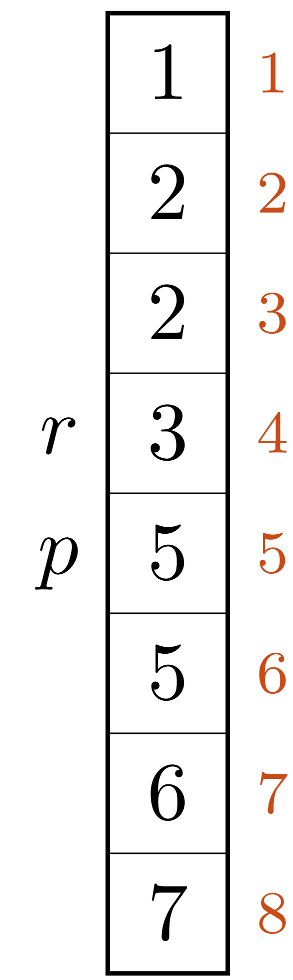
	1	1
	2	2
	2	3
r	3	4
p	5	5
	5	6
	6	7
	7	8

$p, r, v \; = \; 1, 8, 4$ › $5, 8, 4$ › $5, 5, 4$ › $5, 4, 4$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```

$\longrightarrow$ NIL

$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4 \;›\; 5, 5, 4 \;›\; \mathbf{5, 4, 4}$



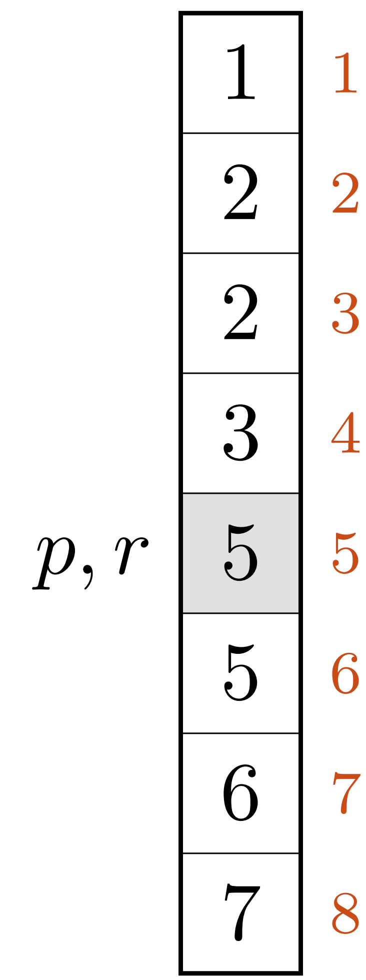
```
BISECT(A, p, r, v)
if p ≤ r
      q = ⌊(p + r)/2⌋
      if v == A[q]
            return q
      elseif v < A[q]
            return BISECT(A, p, q − 1, v)
      else return BISECT(A, q + 1, r, v)
return NIL
```

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

$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4 \;›\; \mathbf{5, 5, 4}$

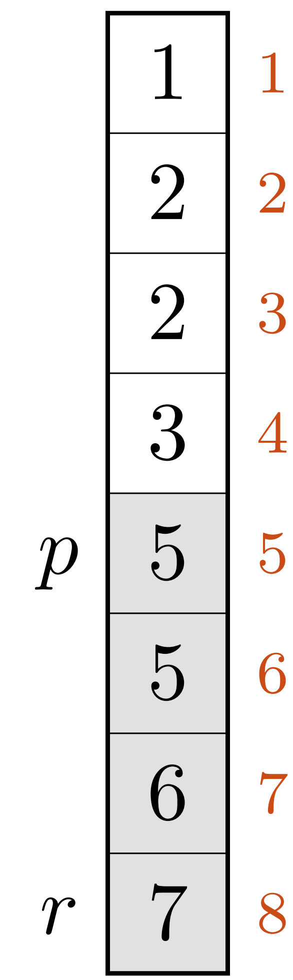
```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return nil
```

→ NIL



$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4 \;›\; \mathbf{5, 5, 4}$

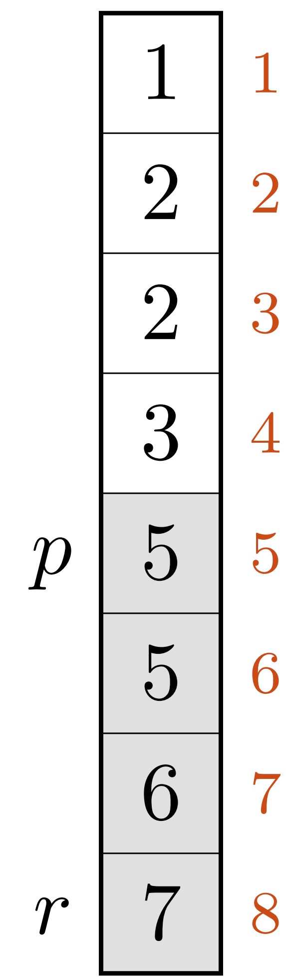
```
BISECT(A, p, r, v)
  if p ≤ r
      q = ⌊(p + r)/2⌋
      if v == A[q]
          return q
      elseif v < A[q]
          return BISECT(A, p, q − 1, v)
      else return BISECT(A, q + 1, r, v)
  return NIL
```



$p, r, v \;=\; 1, 8, 4 \;›\; 5, 8, 4$

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```

$\longrightarrow$ NIL



$p, r, v \;=\;$ 1, 8, 4 › 5, 8, 4

```
BISECT(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return BISECT(A, p, q − 1, v)
    else return BISECT(A, q + 1, r, v)
return NIL
```

$p, r, v \;=\; 1, 8, 4$

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

```
Bisect(A, p, r, v)
if p ≤ r
    q = ⌊(p + r)/2⌋
    if v == A[q]
        return q
    elseif v < A[q]
        return Bisect(A, p, q − 1, v)
    else return Bisect(A, q + 1, r, v)
return NIL
```

$\longrightarrow$ NIL

$p, r, v \;=\; 1, 8, 4$

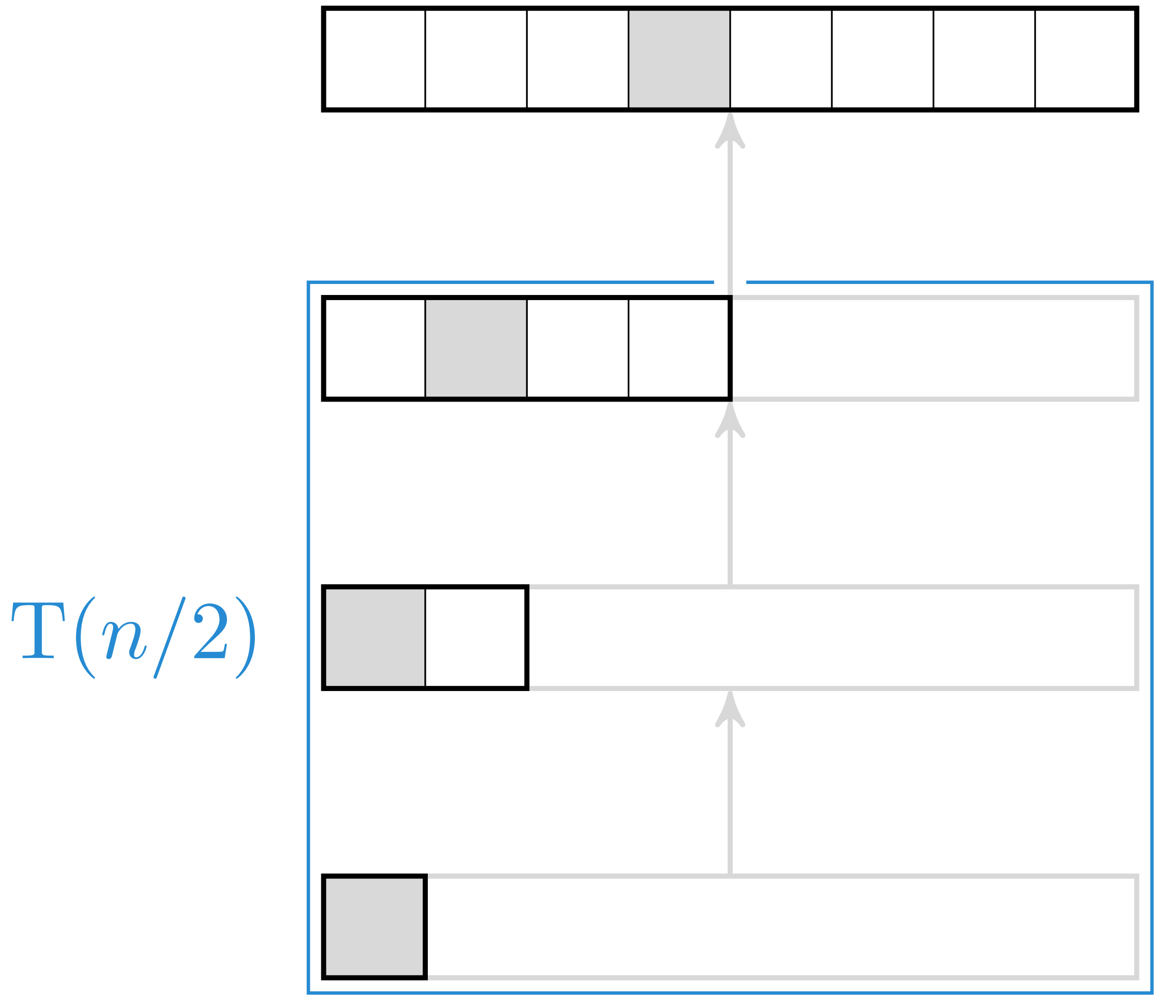
p	1	1
	2	2
	2	3
	3	4
	5	5
	5	6
	6	7
r	7	8

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

Her gjør jeg endel antagelser/forenklinger, f.eks. at delproblemet har størrelse n/2 og ikke n/2 – 1 – og at vi ikke finner v før vi har kommet til bunnen. Disse antagelsene har ikke noe å si for den asymptotiske worst-case-kjøretiden.

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

Ett rekursivt kall (størrelse $n/2$) pluss én operasjon


$T(n/2)$

$$\begin{aligned} \mathrm{T}(n) &= 1 \\ &\quad + \mathrm{T}(n/2) \end{aligned} \tag{1}$$

$$
\begin{aligned}
T(n) = 1& \\
+ T(n/2)& \qquad (1)
\end{aligned}
$$

$$
\begin{aligned}
T(n) = 1 & \\
+\, 1 & \qquad (1) \\
+\, T(n/2/2) & \qquad (2)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= 1 \\
&\quad + 1 && (1) \\
&\quad + \mathrm{T}(n/4) && (2)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= 1 \\
&\quad + 1 && (1) \\
&\quad + 1 && (2) \\
&\quad + \mathrm{T}(n/8) && (3)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= 1 \\
&\quad + 1 && (1) \\
&\quad + 1 && (2) \\
&\quad + 1 && (3) \\
&\quad \vdots && \vdots \\
&\quad + T(n/2^{?}) && (?)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= 1 \\
&\quad + 1 && (1) \\
&\quad + 1 && (2) \\
&\quad + 1 && (3) \\
&\quad \vdots && \vdots \\
&\quad + T(n/2^{?}) && (\lg n)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= 1 \\
&\quad + 1 && (1) \\
&\quad + 1 && (2) \\
&\quad + 1 && (3) \\
&\quad \vdots && \vdots \\
&\quad + T(n/2^{\lg n}) && (\lg n)
\end{aligned}
$$

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

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

$$
\begin{aligned}
\mathrm{T}(n) &= 1 \\
&\quad + 1 && (1) \\
&\quad + 1 && (2) \\
&\quad + 1 && (3) \\
&\quad \vdots && \vdots \\
&\quad + 1 && (\lg n)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= 1 \\
&\quad + 1 && (1) \\
&\quad + 1 && (2) \\
&\quad + 1 && (3) \\
&\quad \vdots && \vdots \\
&\quad + 1 && (\lg n)
\end{aligned}
$$

$$T(n) = \lg n + 1$$

Verifikasjon

Med substitusjon/induksjon

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

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

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

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

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

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

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

$$
\begin{aligned}
T(n) &= T(n/2) + 1 \\
&= (\lg \frac{n}{2} + 1) + 1 \\
&= \lg n - \lg 2 + 2
\end{aligned}
$$

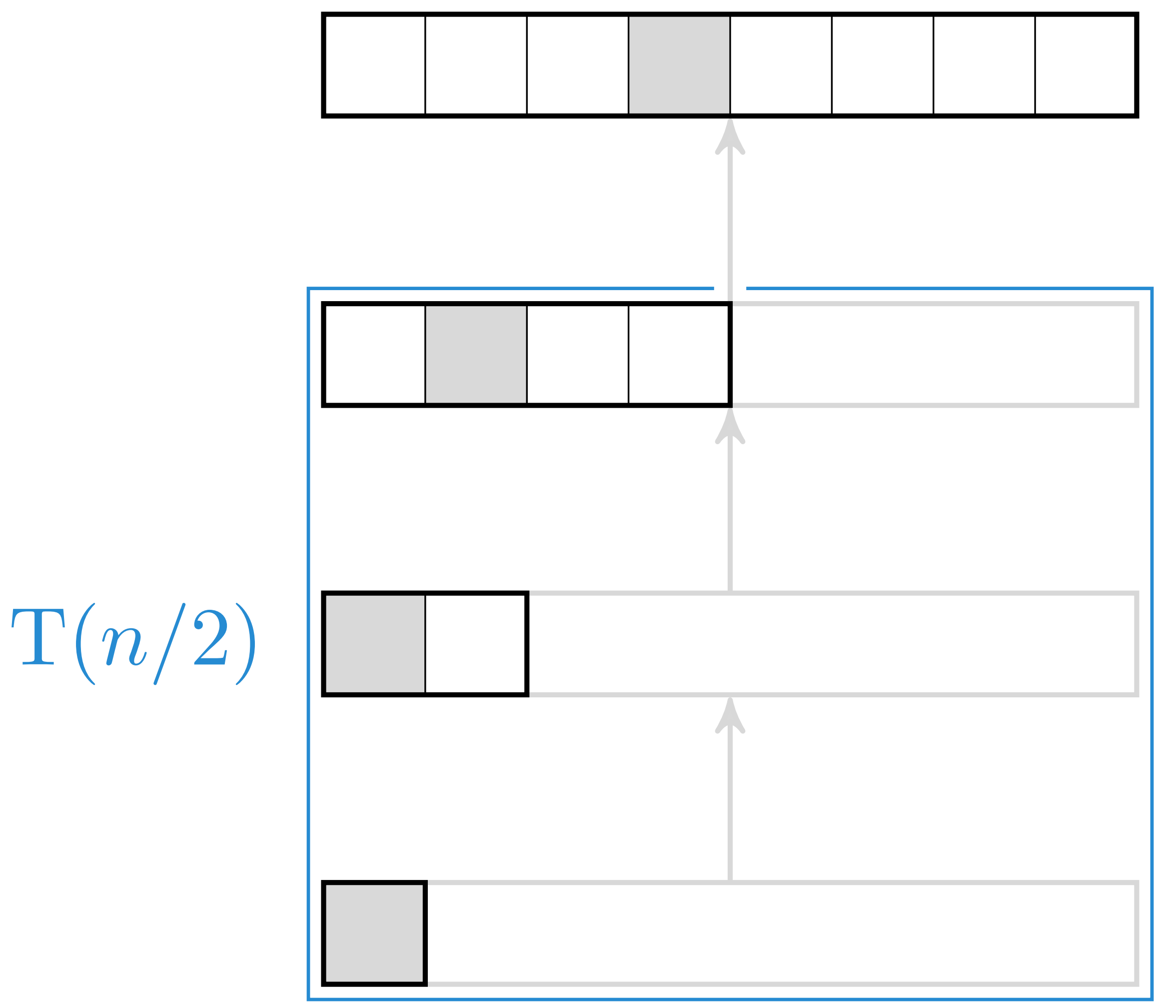
Gitt $T(k) = \lg k + 1$ for $k < n$, vis $T(n) = \lg n + 1$

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

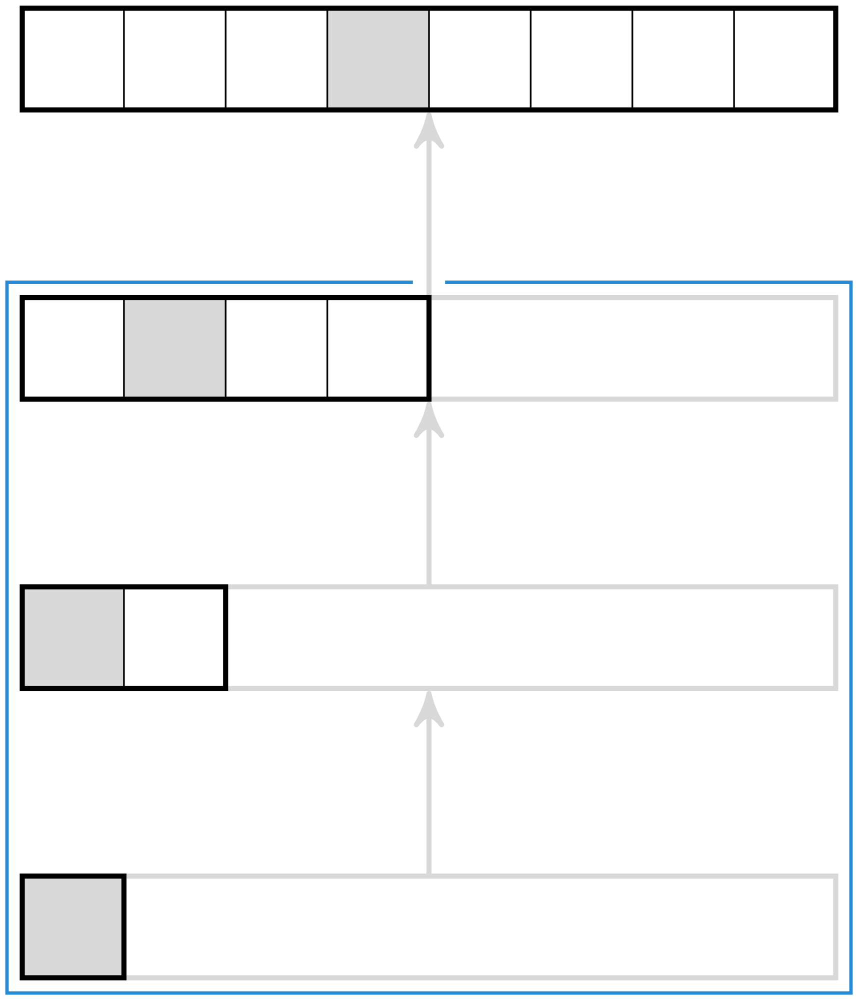
Gitt $\mathrm{T}(k) = \lg k + 1$ for $k < n$, vis $\mathrm{T}(n) = \lg n + 1$

$$
\begin{aligned}
T(n) &= T(n/2) + 1 \\
&= (\lg \frac{n}{2} + 1) + 1 \\
&= \lg n - \lg 2 + 2 \\
&= \lg n + 1
\end{aligned}
$$

Gitt $T(k) = \lg k + 1$ for $k < n$, vis $T(n) = \lg n + 1$


$T(n/2)$

$T(n/2)$



$$T(n) = \lg n + 1$$

«[Von Neumann's] manuscript, written in ink, is 23 pages long; the first page still shows traces of the penciled phrase "TOP SECRET," which was subsequently erased. (In 1945, work on computers was classified, due to its connections with military problems.)»

Donald Knuth, «Von Neumann's First Computer Program»
http://dl.acm.org/citation.cfm?doid=356580.356581

3:6

Merge sort

Kalles også «flettesortering» på norsk.

- **Sortér venstre halvdel rekursivt**
- **Sortér høyre halvdel rekursivt**
- **Flett sammen halvdelene**

- **Sortér venstre halvdel rekursivt**
- **Sortér høyre halvdel rekursivt**
- **Flett sammen halvdelene**

$\textsc{Merge}(\mathrm{A}, p, q, r)$

A	tabell
p	venstre
q	midten
r	høyre

Flett sorterte segmenter $\mathrm{A}[p \,..\, q]$ og $\mathrm{A}[q+1 \,..\, r]$

$\textsc{Merge}(\mathrm{A}, p, q, r)$

1 $n_1 = q - p + 1$

A	tabell
p	venstre
q	midten
r	høyre
n_1	v. lengde

Lengden til første segment, $\mathrm{A}[p \mathinner{.\,.} q]$

$\textsc{Merge}(\mathrm{A}, p, q, r)$

1 $n_1 = q - p + 1$

2 $n_2 = r - q$

A	tabell
p	venstre
q	midten
r	høyre
n_1	v. lengde
n_2	h. lengde

Lengden til andre segment, $\mathrm{A}[p+1\,..\,r]$

$\textsc{Merge}(A, p, q, r)$
1 $n_1 = q - p + 1$
2 $n_2 = r - q$
3 new: $L[1 \mathinner{..} n_1 + 1]$, $R[1 \mathinner{..} n_2 + 1]$

A	tabell
p	venstre
q	midten
r	høyre
n_1	v. lengde
n_2	h. lengde
L	kopi, v.
R	kopi, h.

Vi kopierer midlertidig innholdet i $A[p \mathinner{..} q]$ og $A[q+1 \mathinner{..} r]$ til L og R

$\textsc{Merge}(A, p, q, r)$

$n_1 = q - p + 1$
$n_2 = r - q$
new: $L[1 \,..\, n_1 + 1]$, $R[1 \,..\, n_2 + 1]$
for $i = 1$ **to** n_1

A	tabell
p	venstre
q	midten
r	høyre
n_1	v. lengde
n_2	h. lengde
L	kopi, v.
R	kopi, h.
i	pos. i L

For hver indeks i i L . . .

```
MERGE(A, p, q, r)
  n₁ = q − p + 1
  n₂ = r − q
  new: L[1 . . n₁ + 1], R[1 . . n₂ + 1]
  for i = 1 to n₁
      L[i] = A[p + i − 1]
```

A	tabell
p	venstre
q	midten
r	høyre
n_1	v. lengde
n_2	h. lengde
L	kopi, v.
R	kopi, h.
i	pos. i L

Kopier tilsvarende element fra A$[p \mathinner{..} q]$

```
Merge(A, p, q, r)
n1 = q − p + 1
n2 = r − q
new: L[1 .. n1 + 1], R[1 .. n2 + 1]
for i = 1 to n1
    L[i] = A[p + i − 1]
for j = 1 to n2
```

A	tabell
p	venstre
q	midten
r	høyre
n_1	v. lengde
n_2	h. lengde
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R

For hver indeks j i R . . .

MERGE(A, p, q, r)

$n_1 = q - p + 1$
$n_2 = r - q$
new: $L[1\,..\,n_1 + 1]$, $R[1\,..\,n_2 + 1]$
for $i = 1$ **to** n_1
    $L[i] = A[p + i - 1]$
for $j = 1$ **to** n_2
    $R[j] = A[q + j]$

A	tabell
p	venstre
q	midten
r	høyre
n_1	v. lengde
n_2	h. lengde
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R

Kopier tilsvarende element fra $A[q + 1\,..\,r]$

Merge(A, p, q, r)

$n_1 = q - p + 1$
$n_2 = r - q$
new: $L[1 \,..\, n_1 + 1]$, $R[1 \,..\, n_2 + 1]$
for $i = 1$ **to** n_1
    $L[i] = A[p + i - 1]$
for $j = 1$ **to** n_2
    $R[j] = A[q + j]$
$L[n_1 + 1] = \infty$

A	tabell
p	venstre
q	midten
r	høyre
n_1	v. lengde
n_2	h. lengde
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R

Vaktpost (*sentinel*) så vi slipper å sjekke om vi er ved slutten

$\textsc{Merge}(A, p, q, r)$

$n_1 = q - p + 1$
$n_2 = r - q$
new: $L[1\,..\,n_1 + 1]$, $R[1\,..\,n_2 + 1]$
for $i = 1$ **to** n_1
    $L[i] = A[p + i - 1]$
for $j = 1$ **to** n_2
    $R[j] = A[q + j]$
$L[n_1 + 1] = \infty$
$R[n_2 + 1] = \infty$

A	tabell
p	venstre
q	midten
r	høyre
n_1	v. lengde
n_2	h. lengde
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R

Vaktpost (*sentinel*) så vi slipper å sjekke om vi er ved slutten

$\textsc{Merge}(\mathrm{A}, p, q, r)$

```
n₁ = q − p + 1
n₂ = r − q
new: L[1 .. n₁ + 1], R[1 .. n₂ + 1]
for i = 1 to n₁
        L[i] = A[p + i − 1]
for j = 1 to n₂
        R[j] = A[q + j]
L[n₁ + 1] = ∞
R[n₂ + 1] = ∞
...
```

A	tabell
p	venstre
q	midten
r	høyre
n_1	v. lengde
n_2	h. lengde
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R

$\textsc{Merge}(A, p, q, r)$
10 $\dots$

A	tabell
p	venstre
q	midten
r	høyre
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R

Merge(A, p, q, r)

$\ldots$

$i = 1$

A	tabell
p	venstre
q	midten
r	høyre
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R

Hvor er vi i L?

$\textsc{Merge}(A, p, q, r)$

```
…
i = 1
j = 1
```

A	tabell
p	venstre
q	midten
r	høyre
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R

Hvor er vi i R?

```
MERGE(A, p, q, r)
  ...
  i = 1
  j = 1
  for k = p to r
```

A	tabell
p	venstre
q	midten
r	høyre
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R
k	pos. i A

Vi setter inn i $A[p \mathinner{..} r]$. For hver indeks k, bruk $\min\{L[i], R[j]\}$

```
Merge(A, p, q, r)
  . . .
  i = 1
  j = 1
  for k = p to r
        if L[i] ≤ R[j]
```

A	tabell
p	venstre
q	midten
r	høyre
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R
k	pos. i A

Hvis L[i] er minst . . .

MERGE(A, p, q, r)

```
...
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
```

A	tabell
p	venstre
q	midten
r	høyre
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R
k	pos. i A

Bruk L[i]

mlh@ntnu.no

```
MERGE(A, p, q, r)
  ...
  i = 1
  j = 1
  for k = p to r
      if L[i] ≤ R[j]
          A[k] = L[i]
          i = i + 1
```

A	tabell
p	venstre
q	midten
r	høyre
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R
k	pos. i A

Vi har brukt L[i], så vi flytter oss videre

```
Merge(A, p, q, r)
    …
    i = 1
    j = 1
    for k = p to r
        if L[i] ≤ R[j]
            A[k] = L[i]
            i = i + 1
        else A[k] = R[j]
```

A tabell
p venstre
q midten
r høyre
L kopi, v.
R kopi, h.
i pos. i L
j pos. i R
k pos. i A

Ellers så er R[j] minst, så vi bruker den

```
MERGE(A, p, q, r)
  ...
  i = 1
  j = 1
  for k = p to r
      if L[i] ≤ R[j]
          A[k] = L[i]
          i = i + 1
      else A[k] = R[j]
          j = j + 1
```

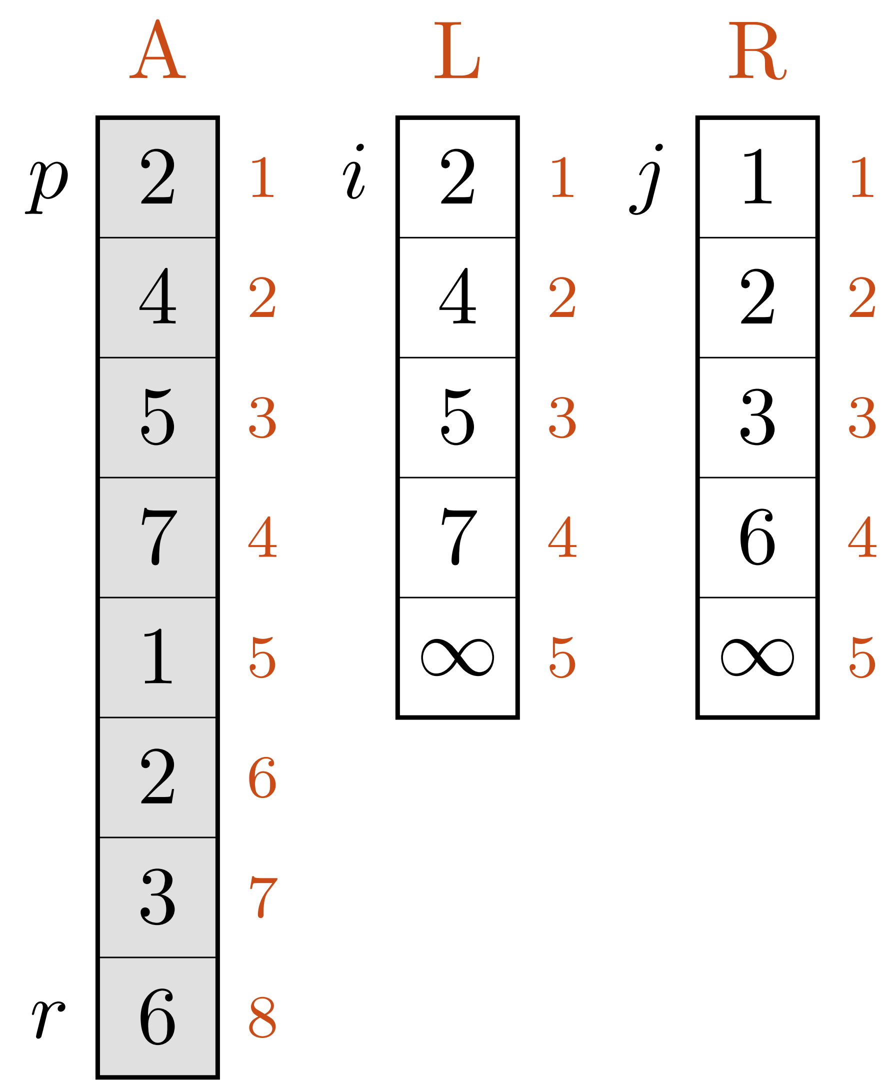
A	tabell
p	venstre
q	midten
r	høyre
L	kopi, v.
R	kopi, h.
i	pos. i L
j	pos. i R
k	pos. i A

Vi har brukt R[j], så vi flytter oss videre

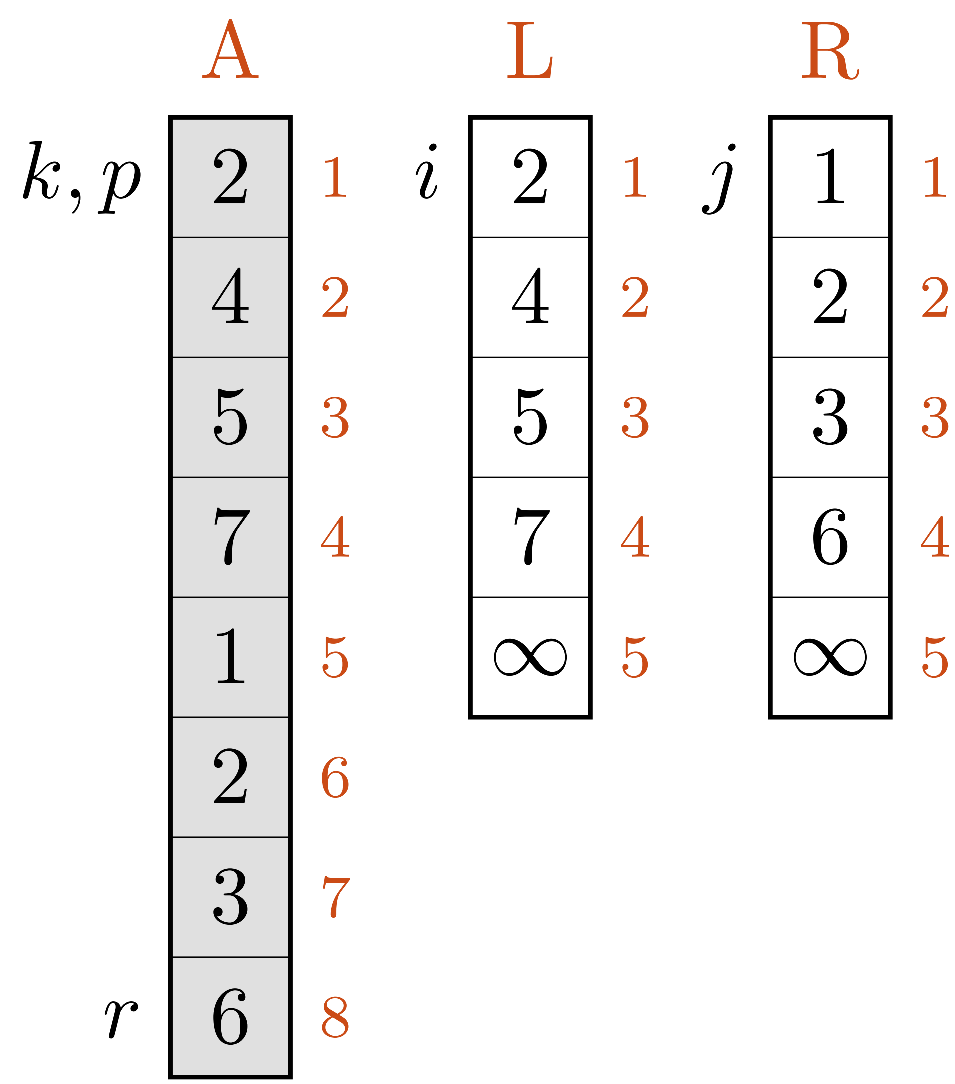
```
MERGE(A, p, q, r)
copy into L and R
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

Forenklet fremstilling av samme prosedyre

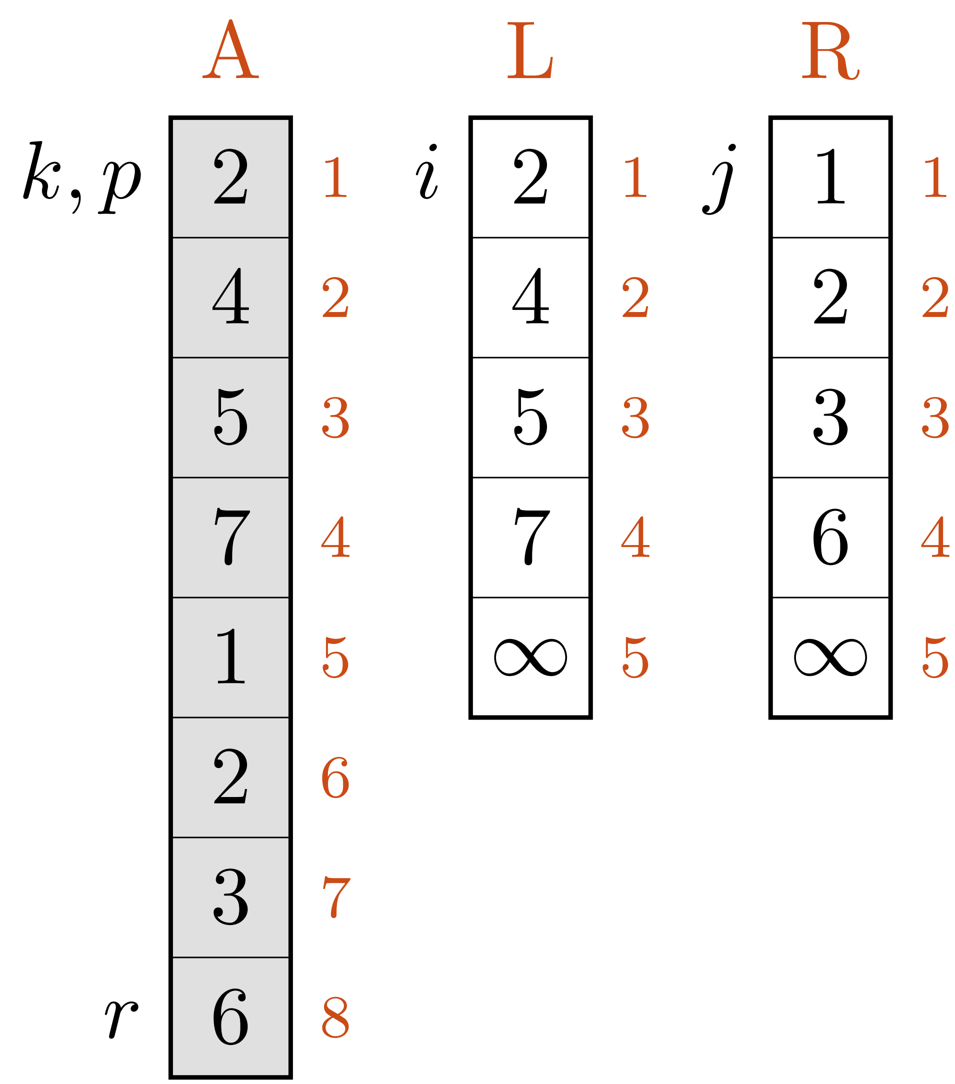
```
Merge(A, p, q, r)
copy into L and R
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```



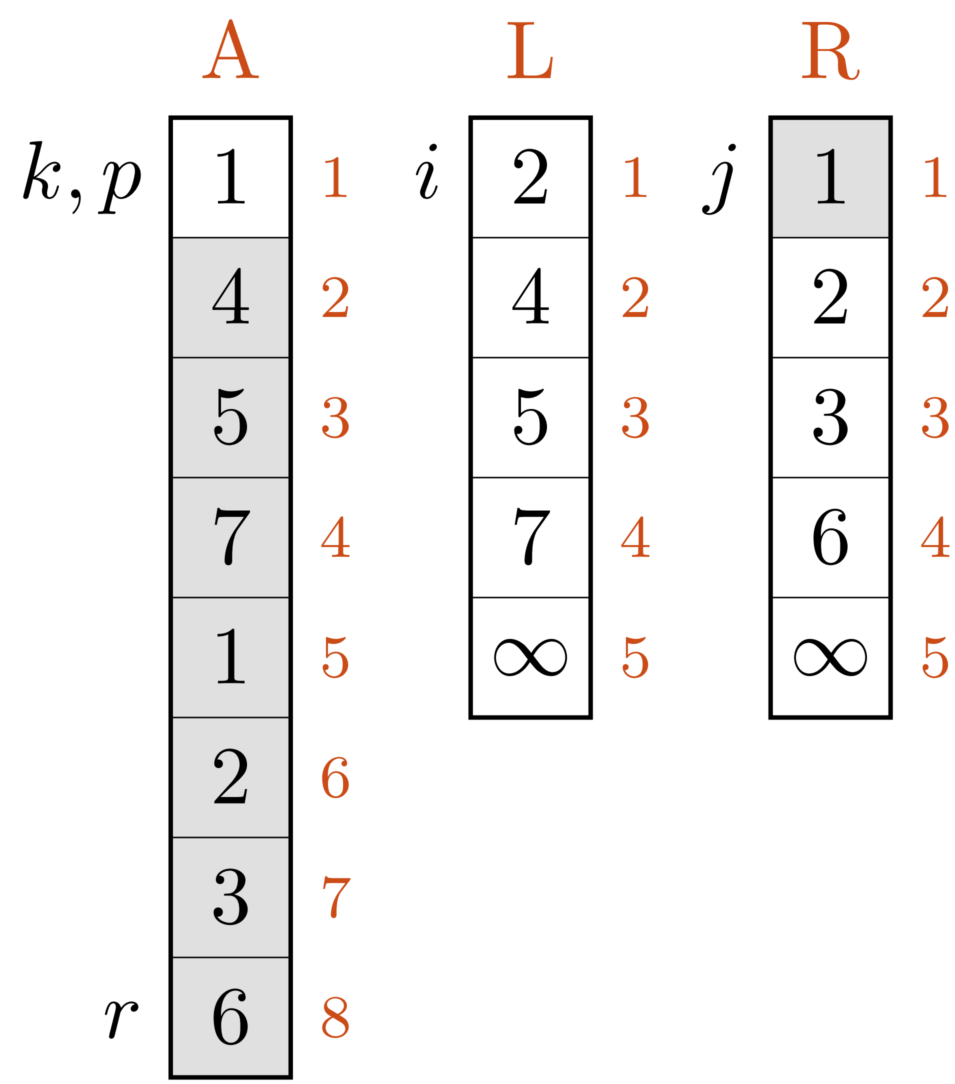
```
Merge(A, p, q, r)
copy into L and R
for k = p to r
        if L[i] ≤ R[j]
                A[k] = L[i]
                i = i + 1
        else A[k] = R[j]
                j = j + 1
```



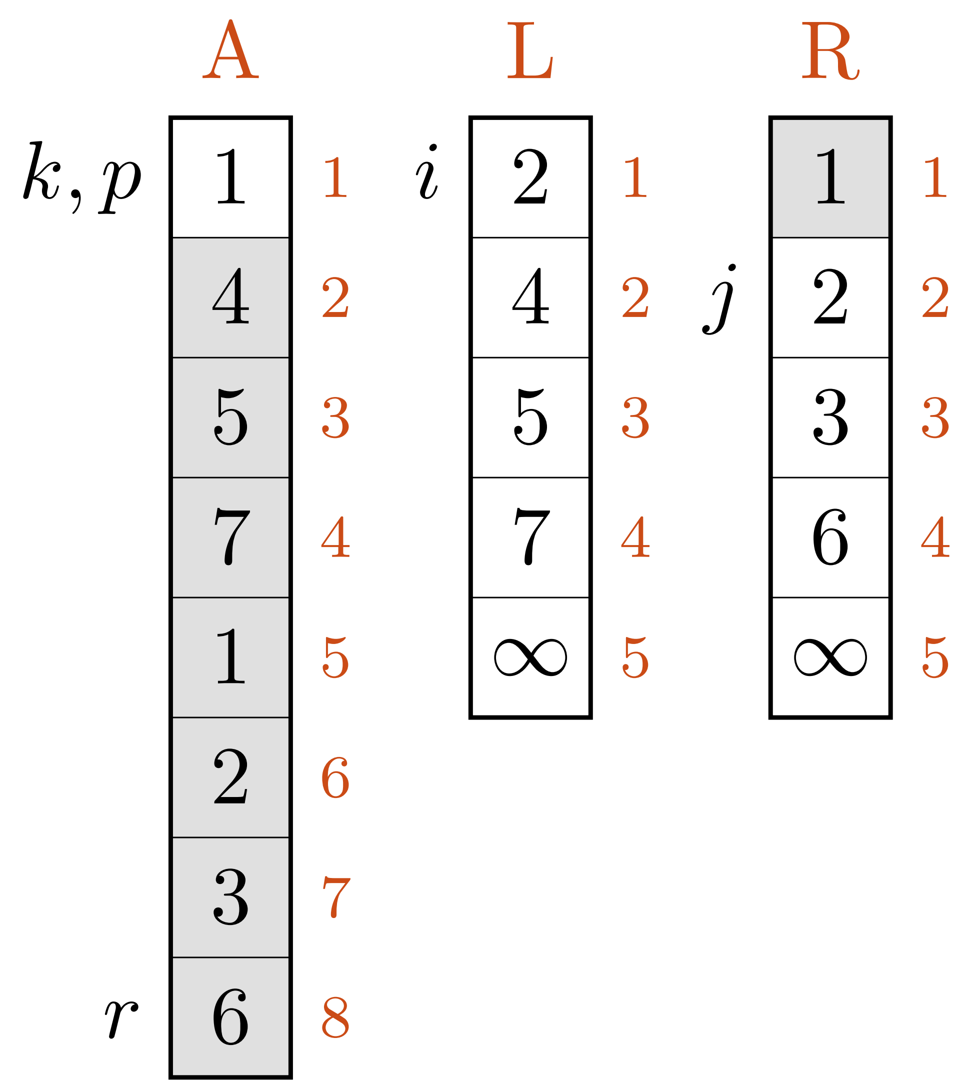
```
MERGE(A, p, q, r)
  copy into L and R
  for k = p to r
          if L[i] ≤ R[j]
                  A[k] = L[i]
                  i = i + 1
          else A[k] = R[j]
                  j = j + 1
```



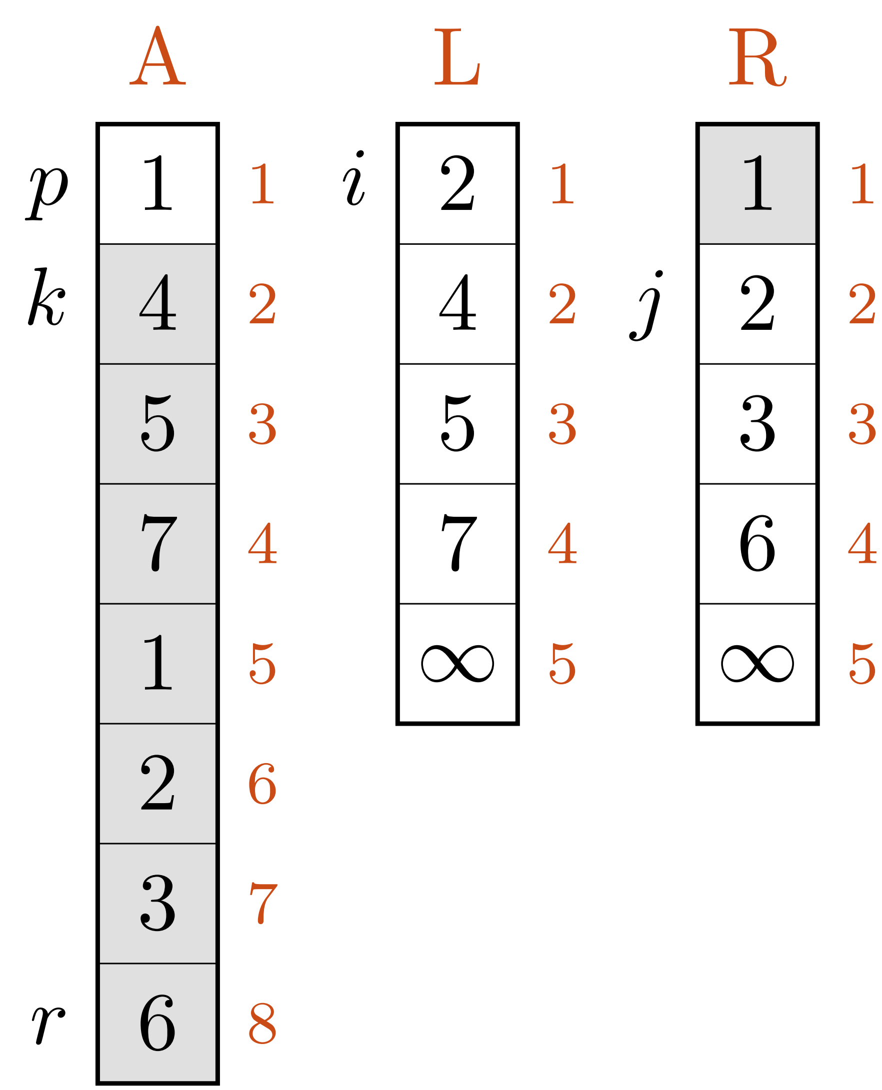
```
Merge(A, p, q, r)
  copy into L and R
  for k = p to r
        if L[i] ≤ R[j]
              A[k] = L[i]
              i = i + 1
        else A[k] = R[j]
              j = j + 1
```



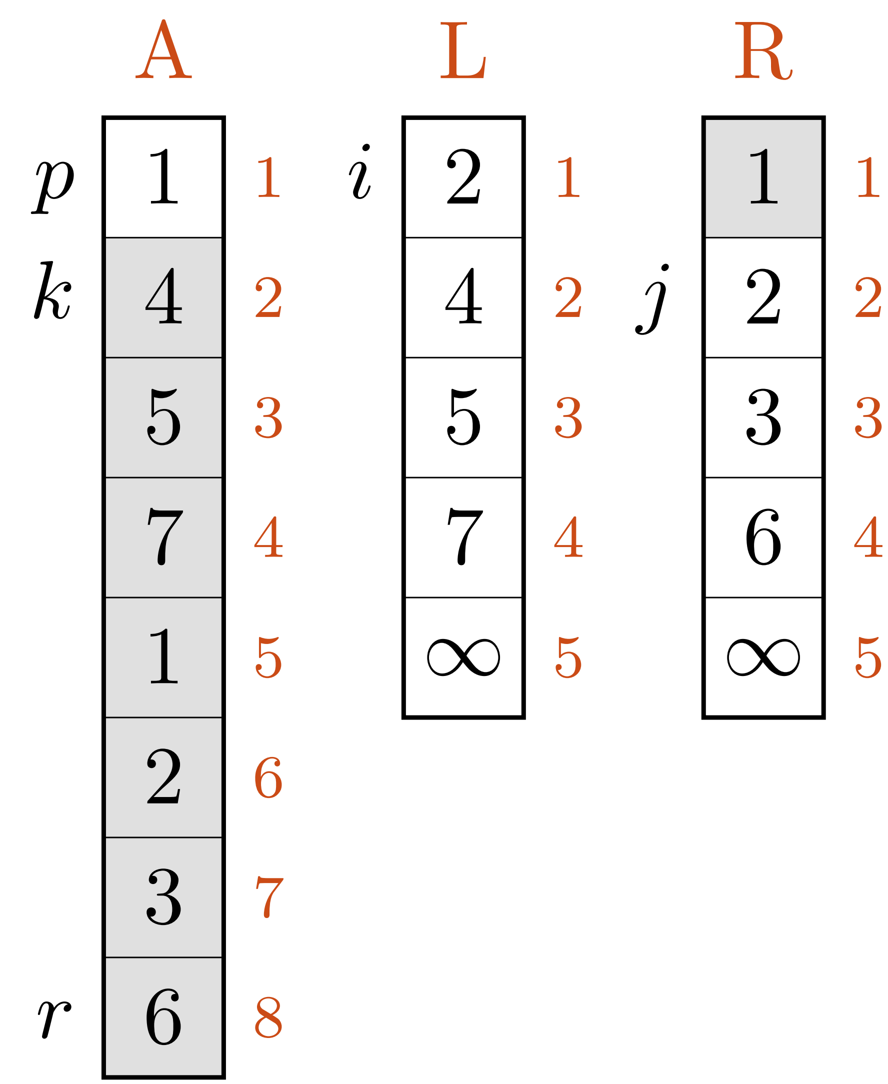
```
Merge(A, p, q, r)
  copy into L and R
  for k = p to r
      if L[i] ≤ R[j]
          A[k] = L[i]
          i = i + 1
      else A[k] = R[j]
          j = j + 1
```



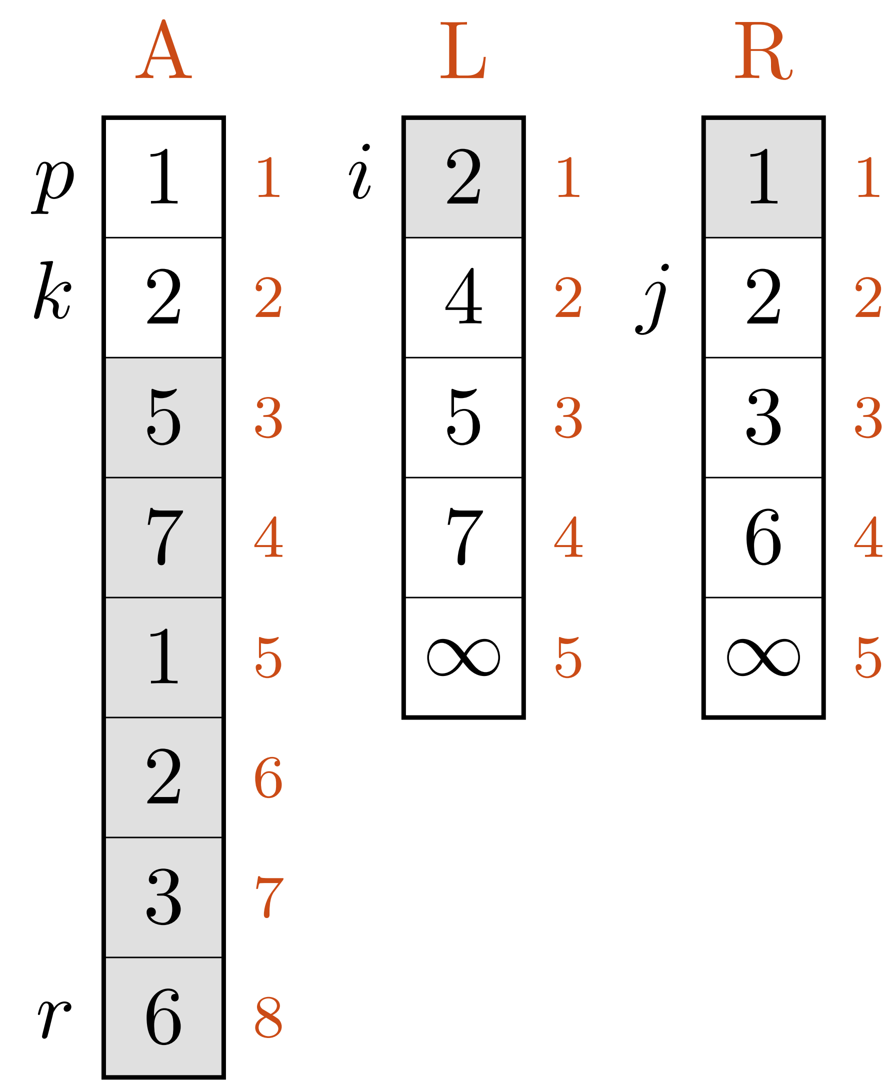
```
Merge(A, p, q, r)
copy into L and R
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```



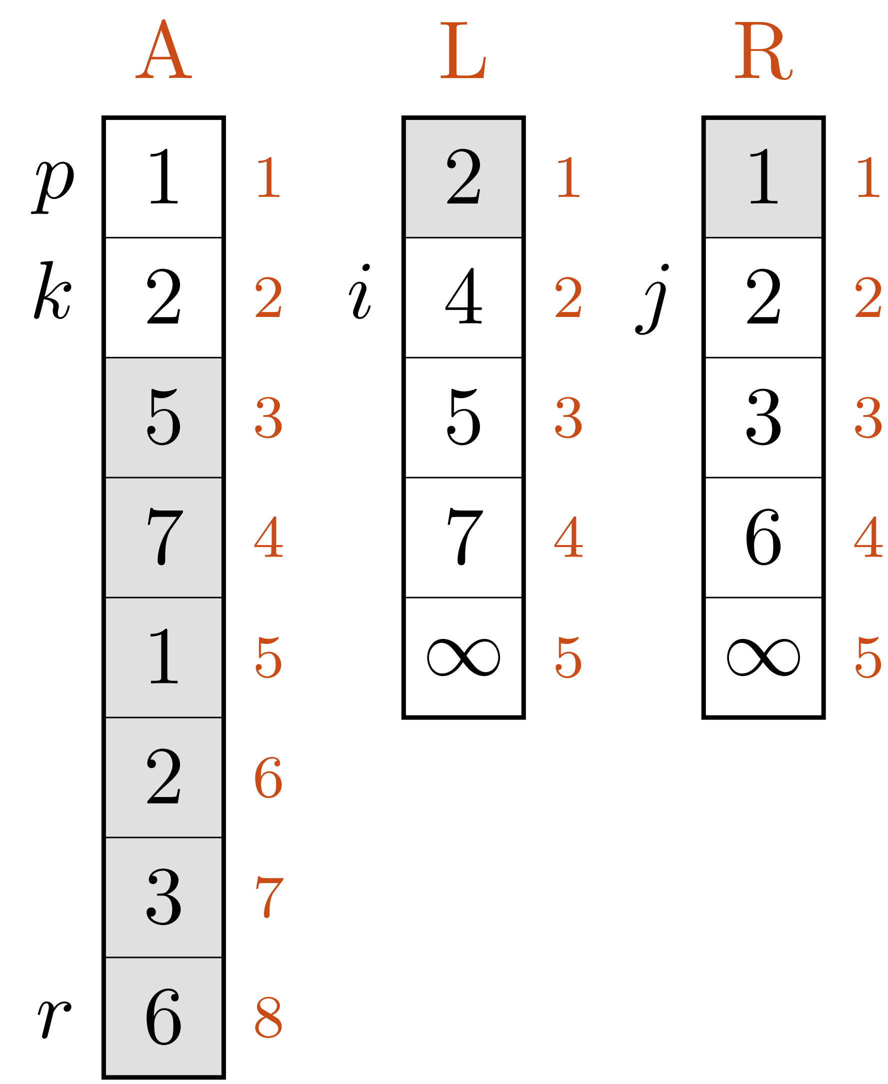
```
Merge(A, p, q, r)
  copy into L and R
  for k = p to r
        if L[i] ≤ R[j]
              A[k] = L[i]
              i = i + 1
        else A[k] = R[j]
              j = j + 1
```



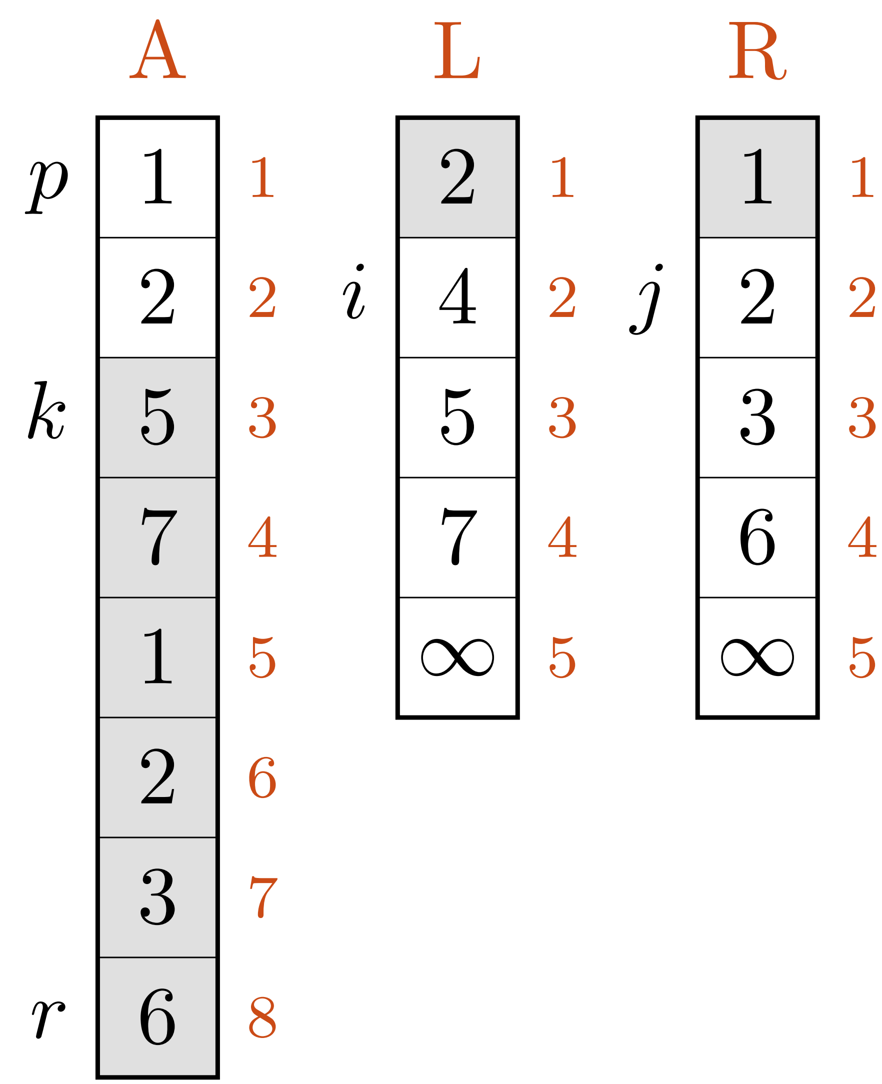
```
MERGE(A, p, q, r)
copy into L and R
for k = p to r
        if L[i] ≤ R[j]
                A[k] = L[i]
                i = i + 1
        else A[k] = R[j]
                j = j + 1
```



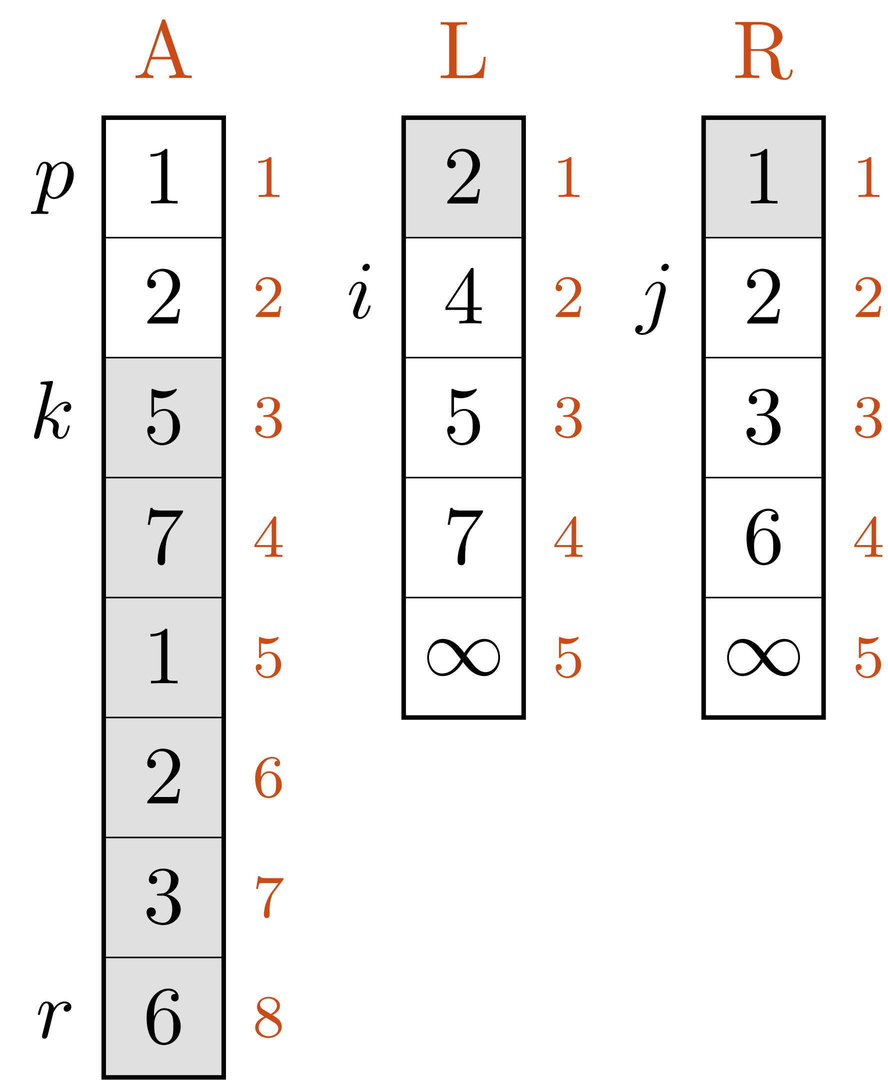
```
MERGE(A, p, q, r)
copy into L and R
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```



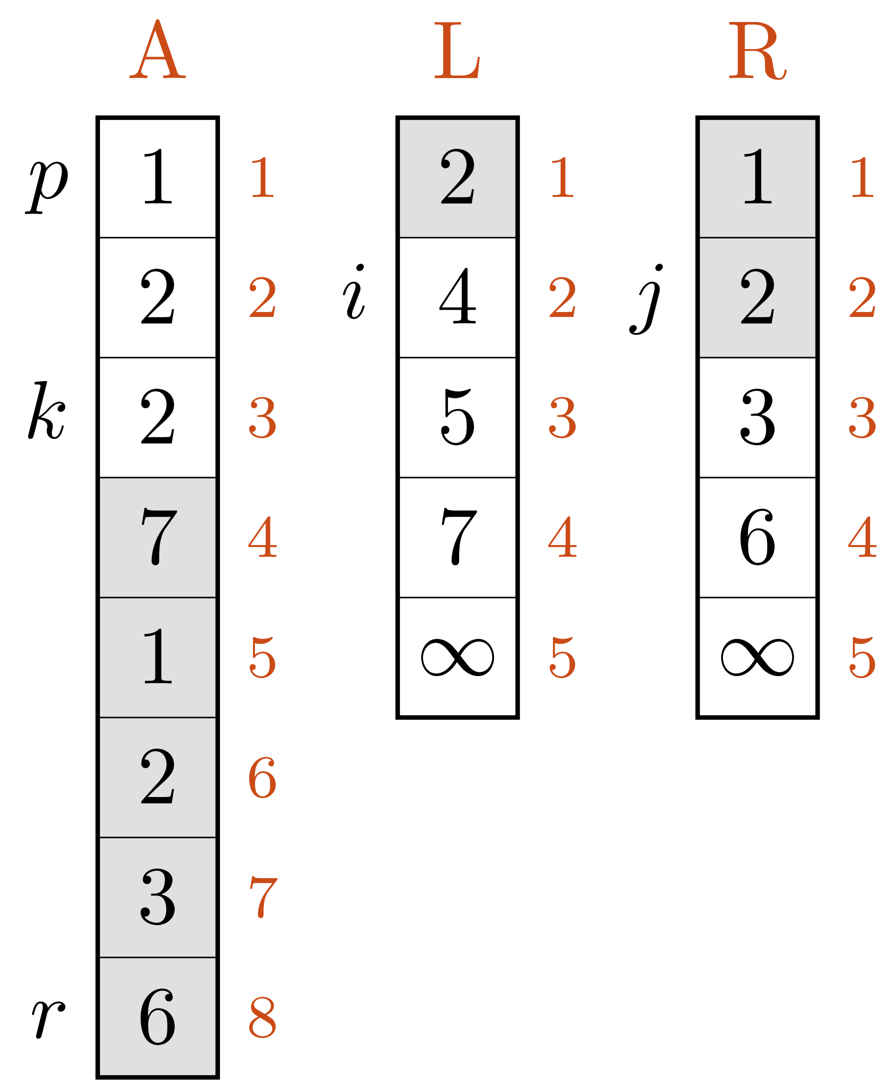
```
Merge(A, p, q, r)
copy into L and R
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```



```
MERGE(A, p, q, r)
copy into L and R
for k = p to r
        if L[i] ≤ R[j]
                A[k] = L[i]
                i = i + 1
        else A[k] = R[j]
                j = j + 1
```

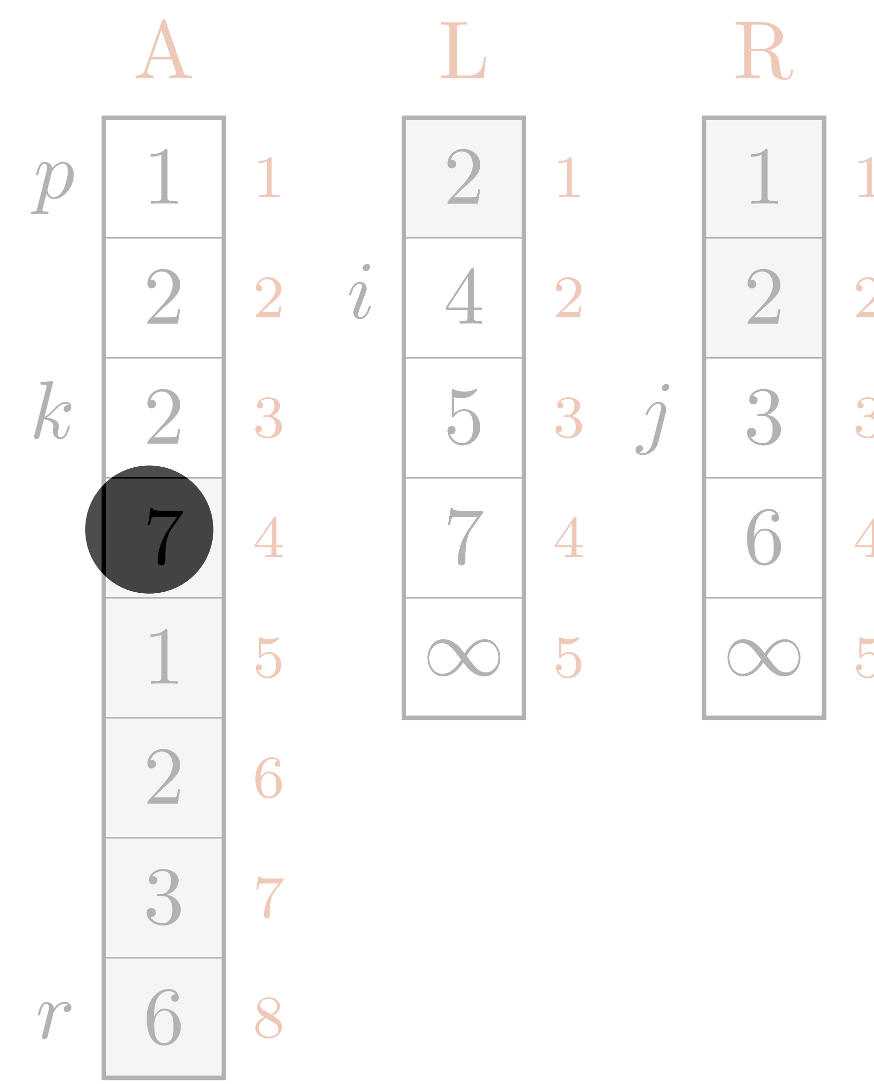


```
Merge(A, p, q, r)
copy into L and R
for k = p to r
        if L[i] ≤ R[j]
              A[k] = L[i]
              i = i + 1
        else A[k] = R[j]
              j = j + 1
```

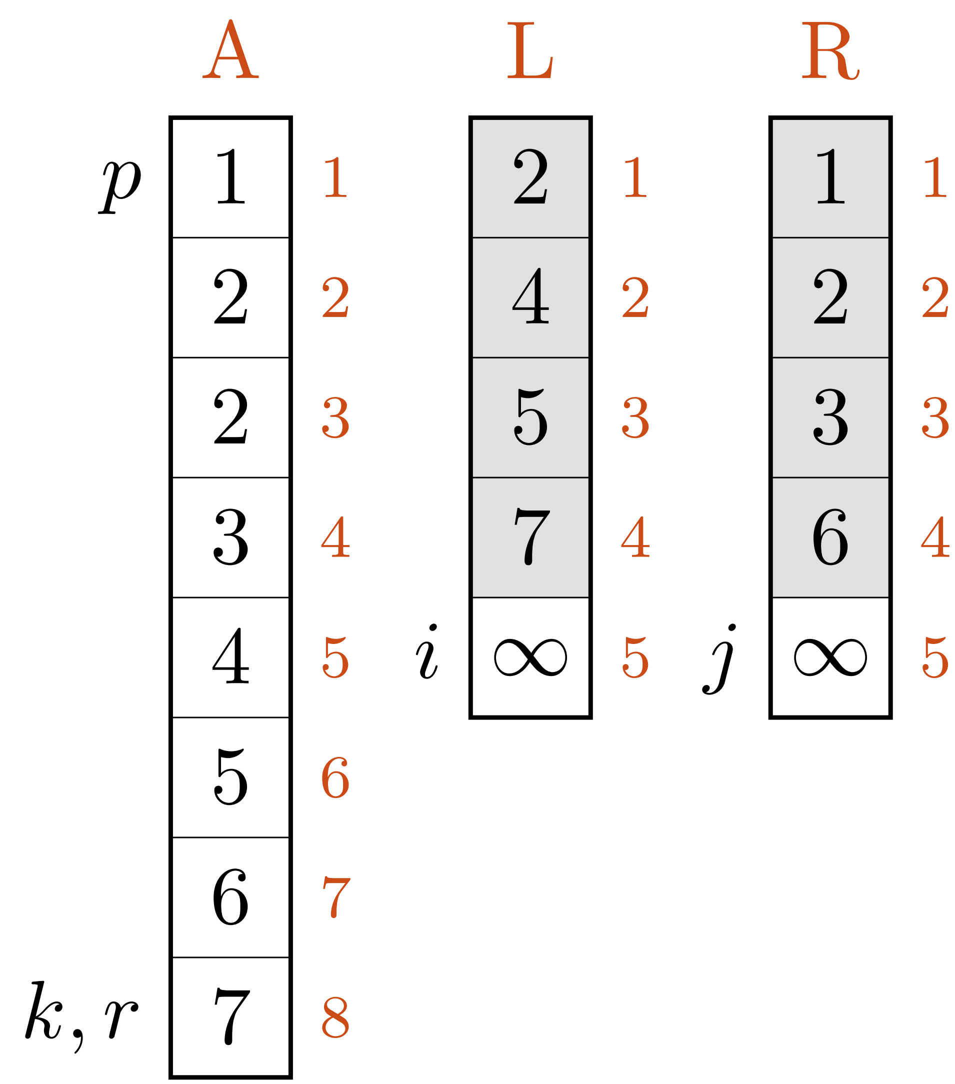


Merge(A, p, q, r)

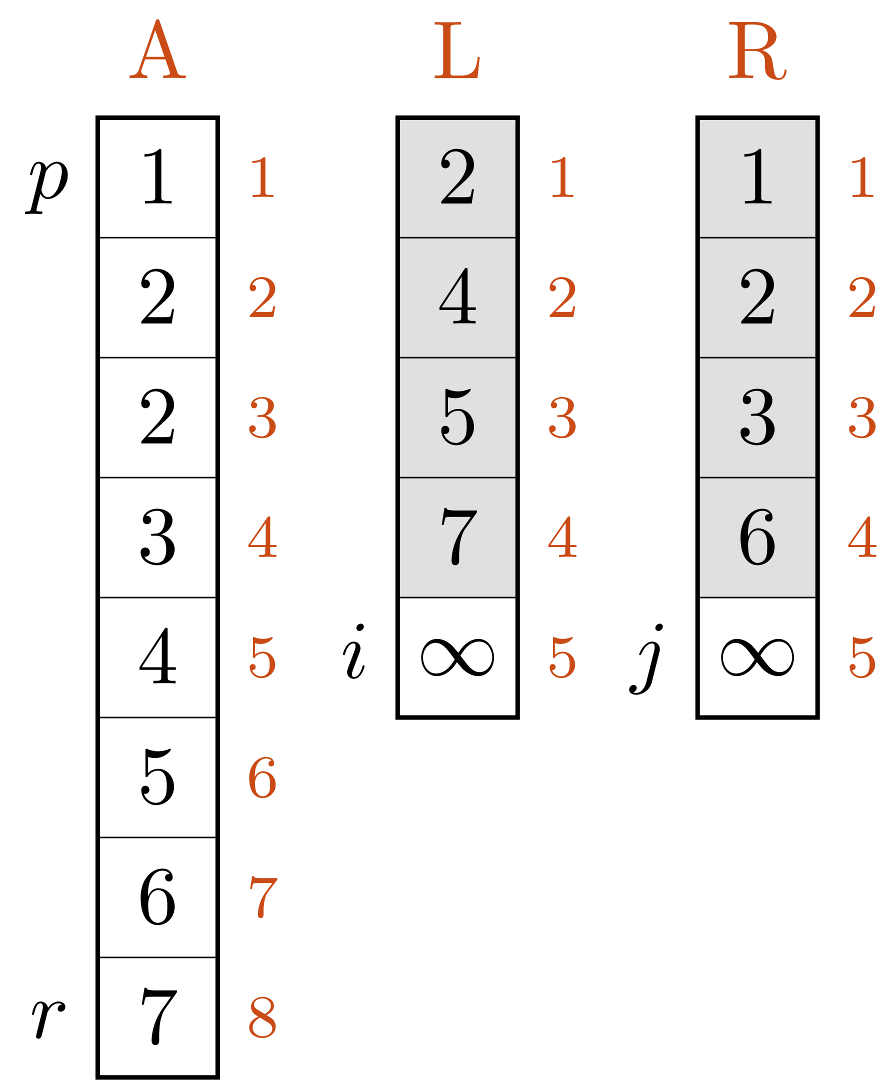
```
copy into L and R
for k = p to r
        if L[i] ≤ R[j]
                A[k] = L[i]
                i = i + 1
        else A[k] = R[j]
                j = j + 1
```



```
Merge(A, p, q, r)
copy into L and R
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

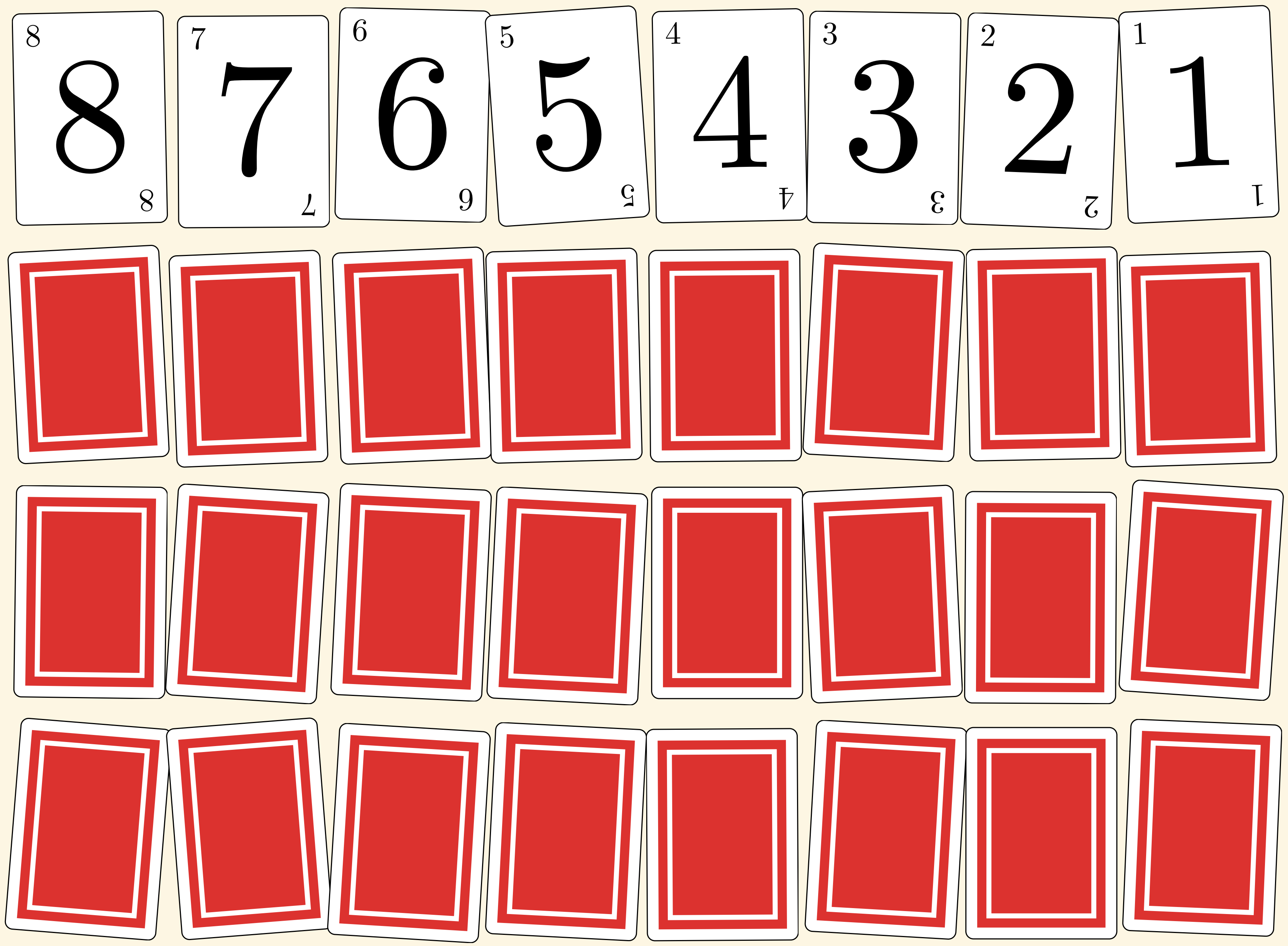


```
MERGE(A, p, q, r)
copy into L and R
for k = p to r
        if L[i] ≤ R[j]
                A[k] = L[i]
                i = i + 1
        else A[k] = R[j]
                j = j + 1
```

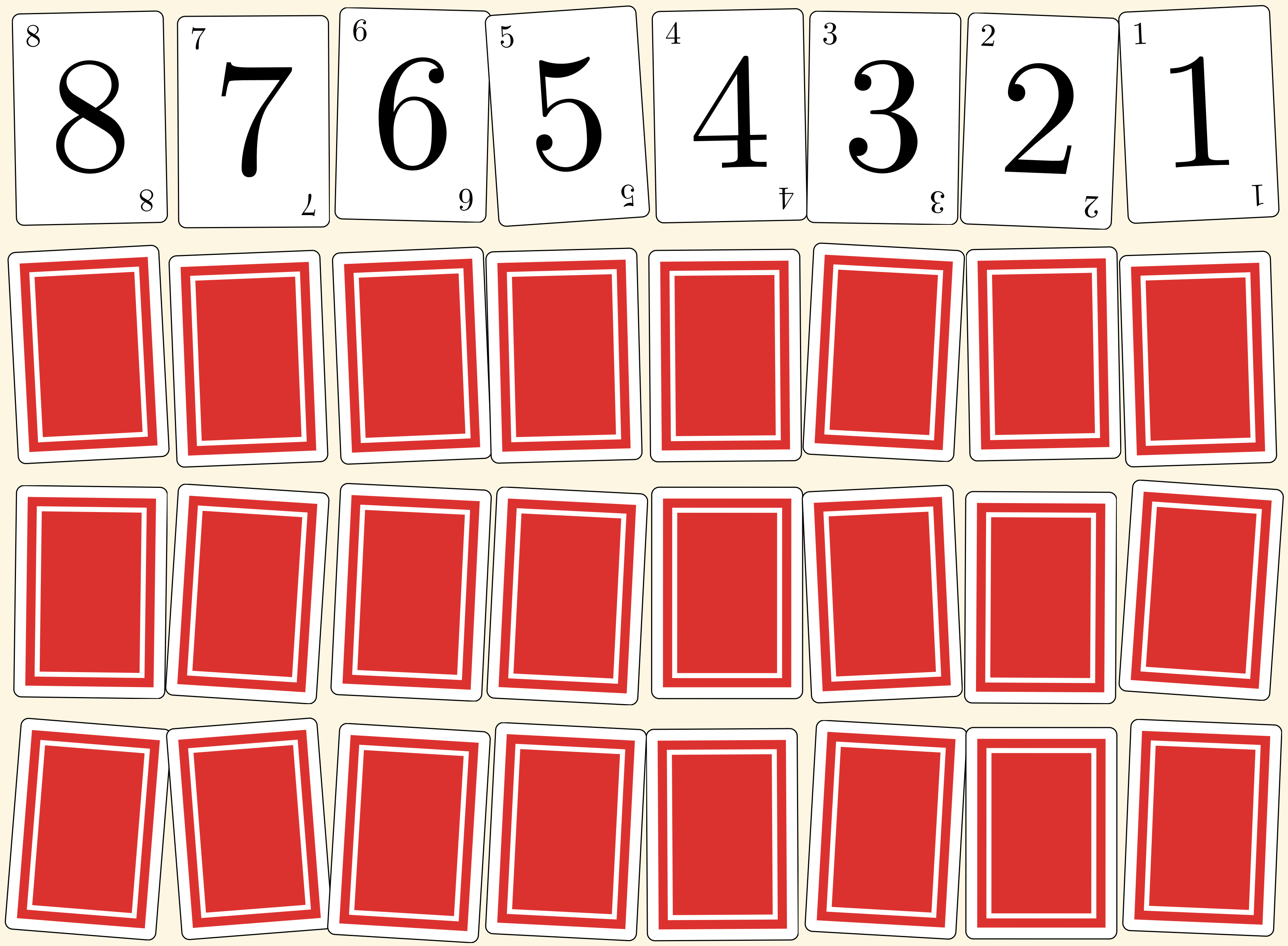


- Sortér venstre halvdel rekursivt
- Sortér høyre halvdel rekursivt
- **Flett sammen halvdelene**

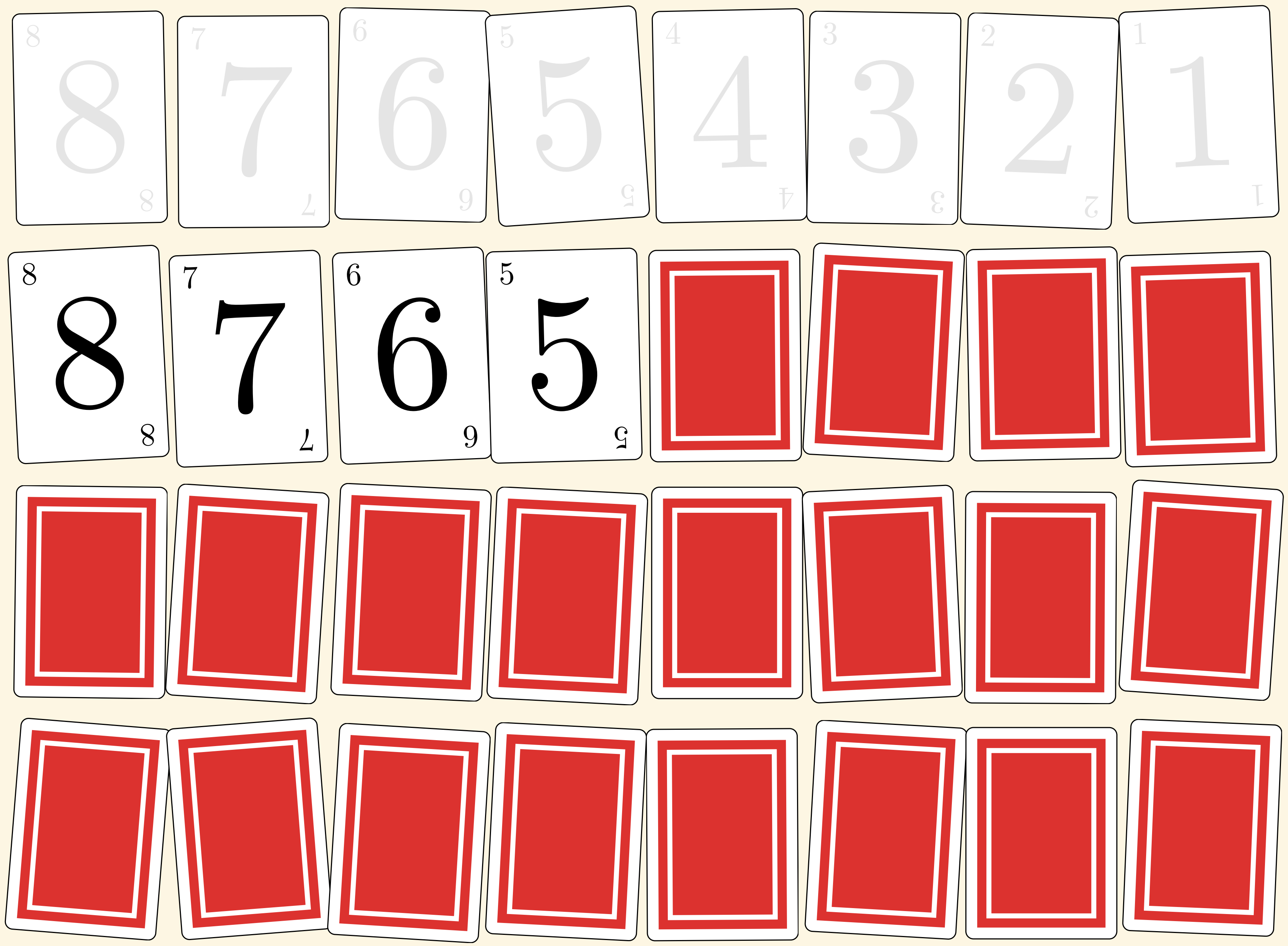
- **Sortér venstre halvdel rekursivt**
- **Sortér høyre halvdel rekursivt**
- **Flett sammen halvdelene**



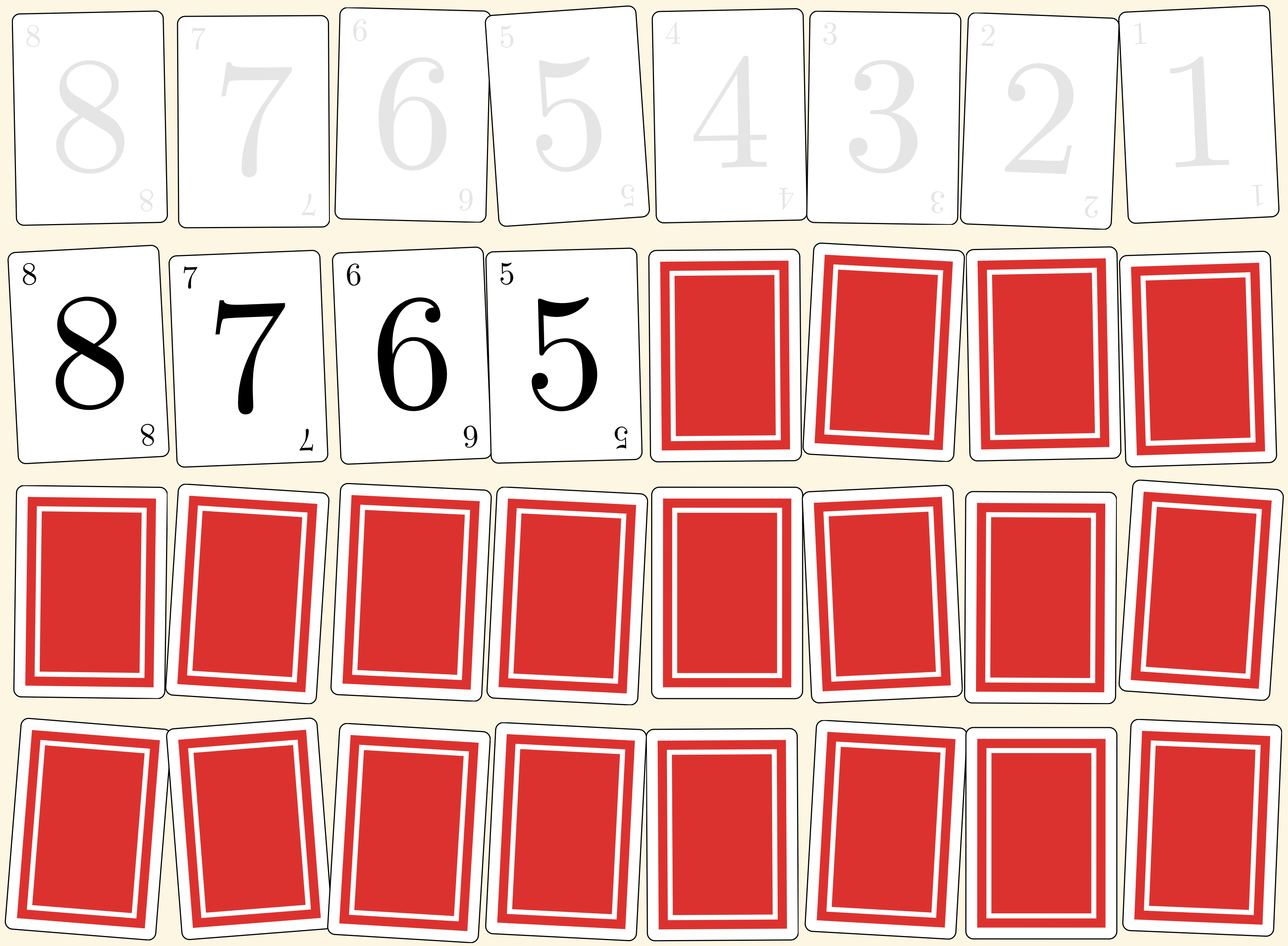
Skal sortere A[1 . . 8]



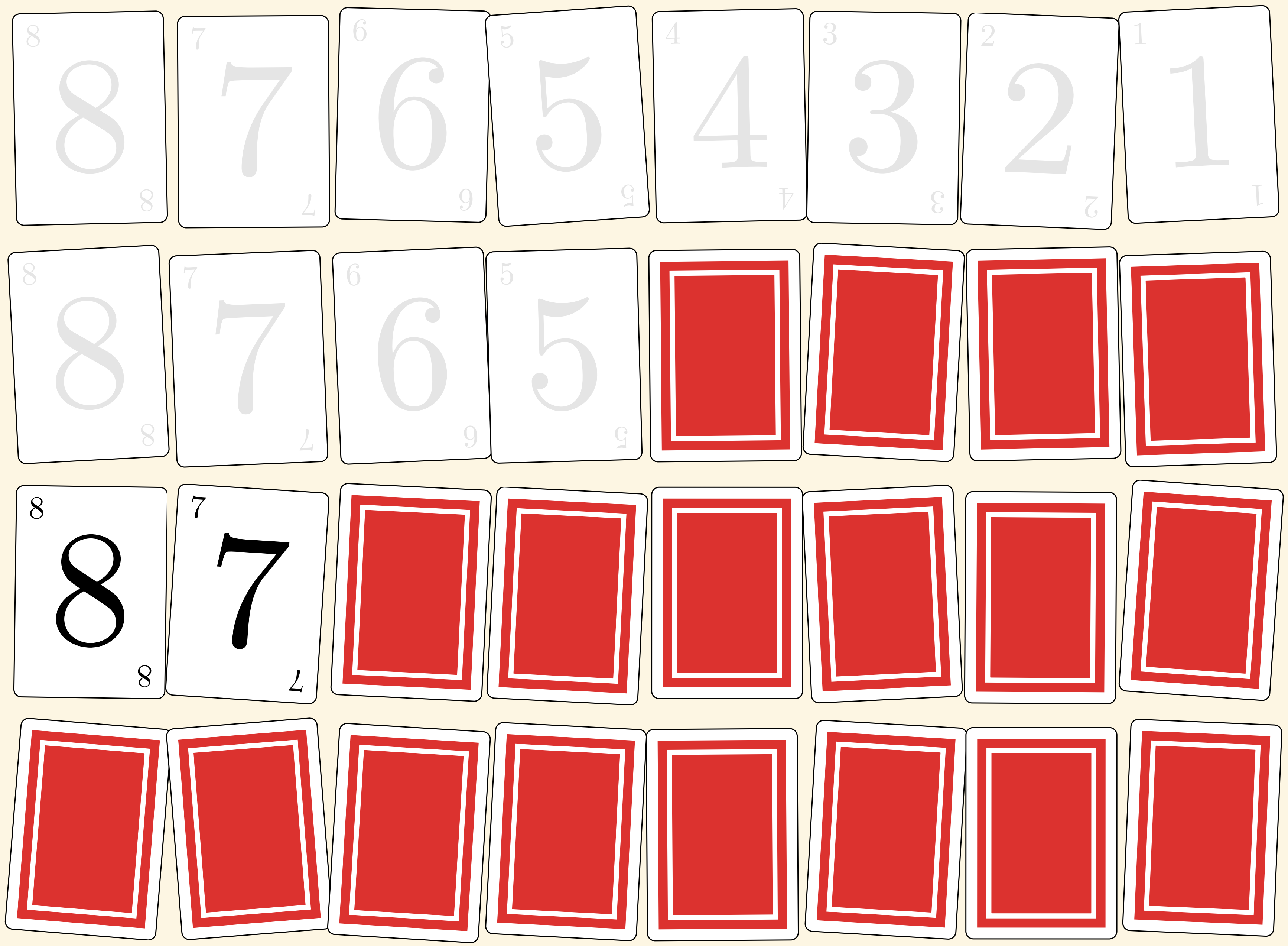
Sortér først A[1 . . 4] og så A[5 . . 8]



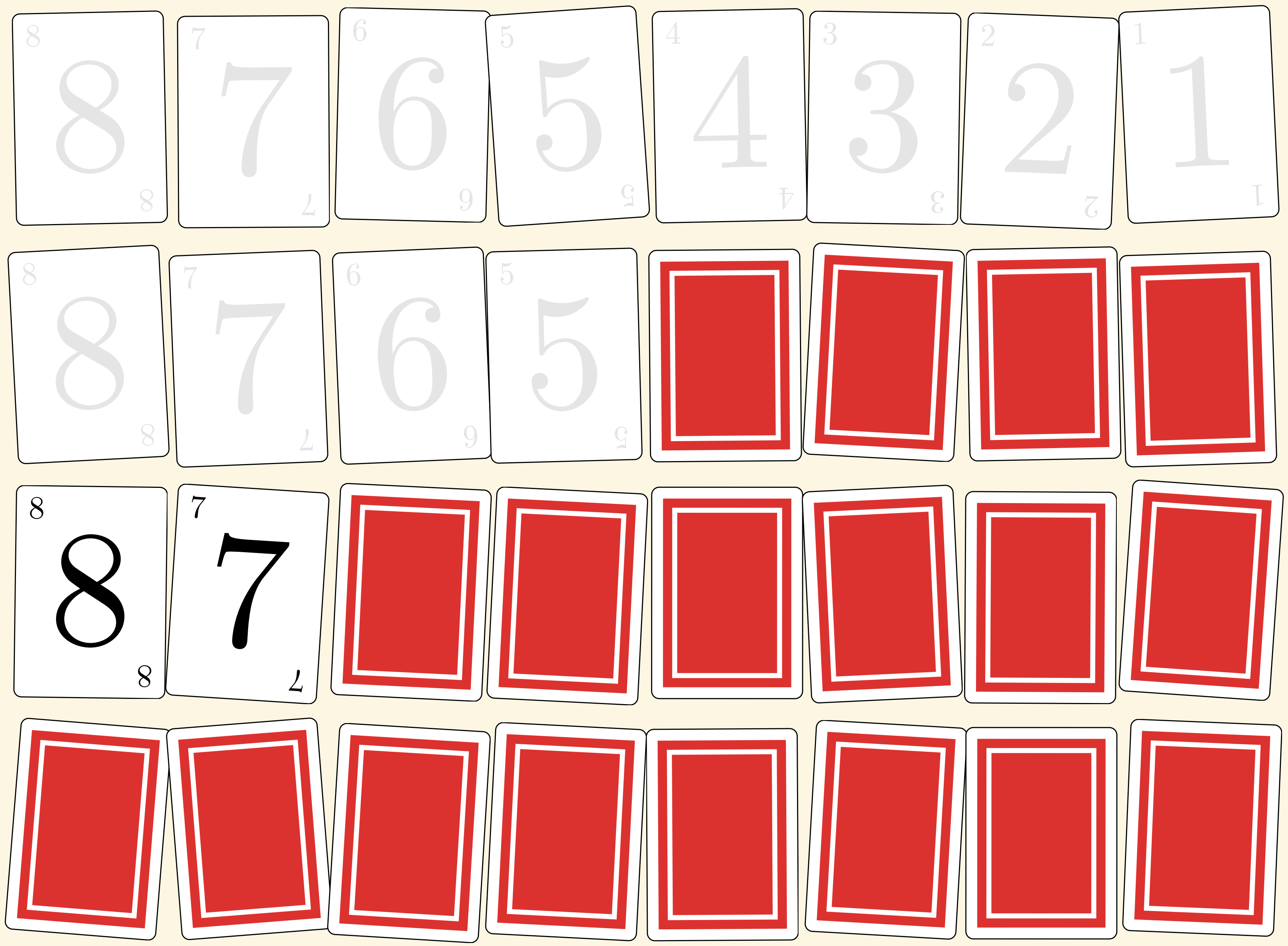
Skal sortere $A[1\,..\,4]$



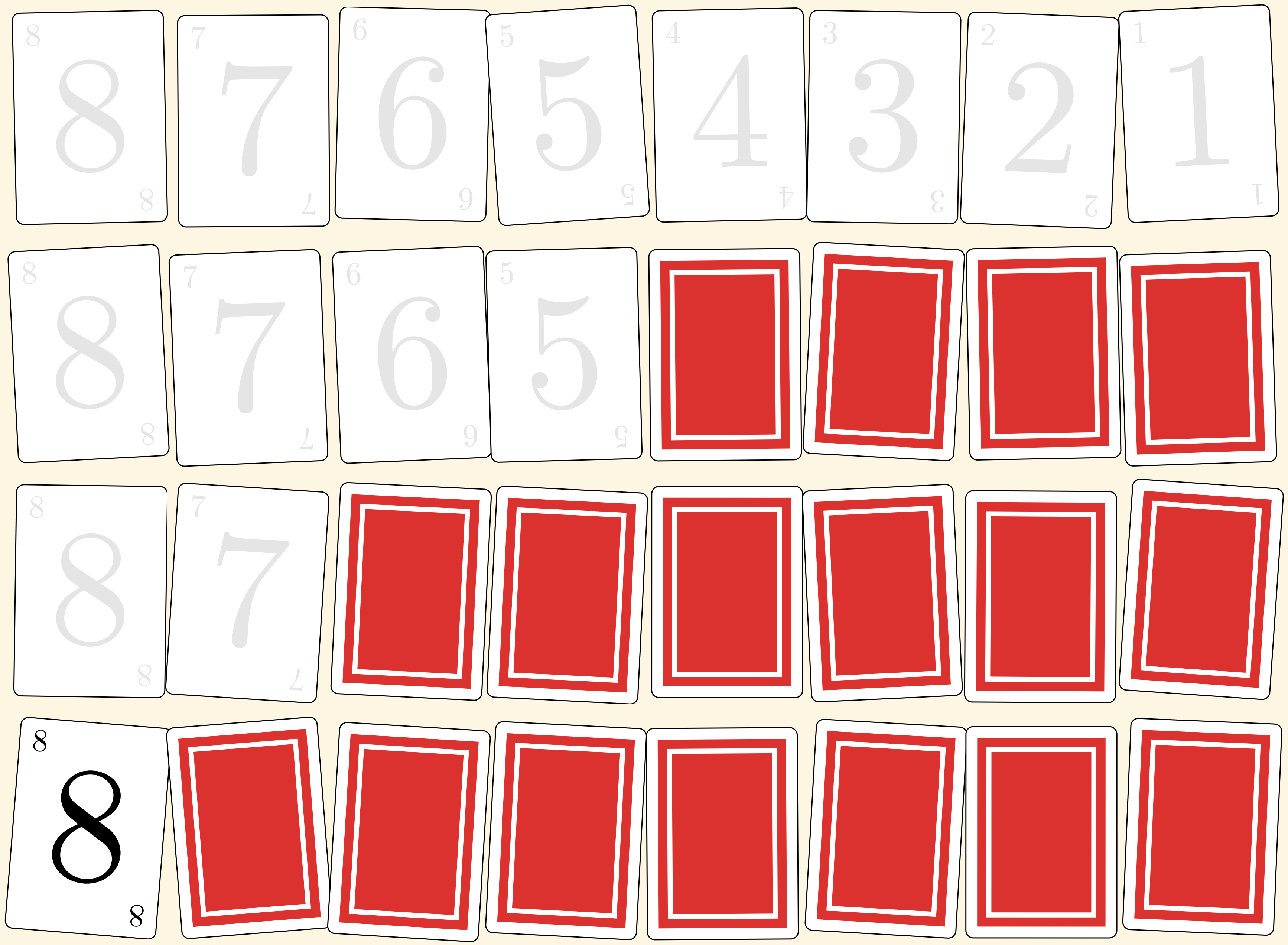
Sortér først A[1 . . 2] og så A[3 . . 4]



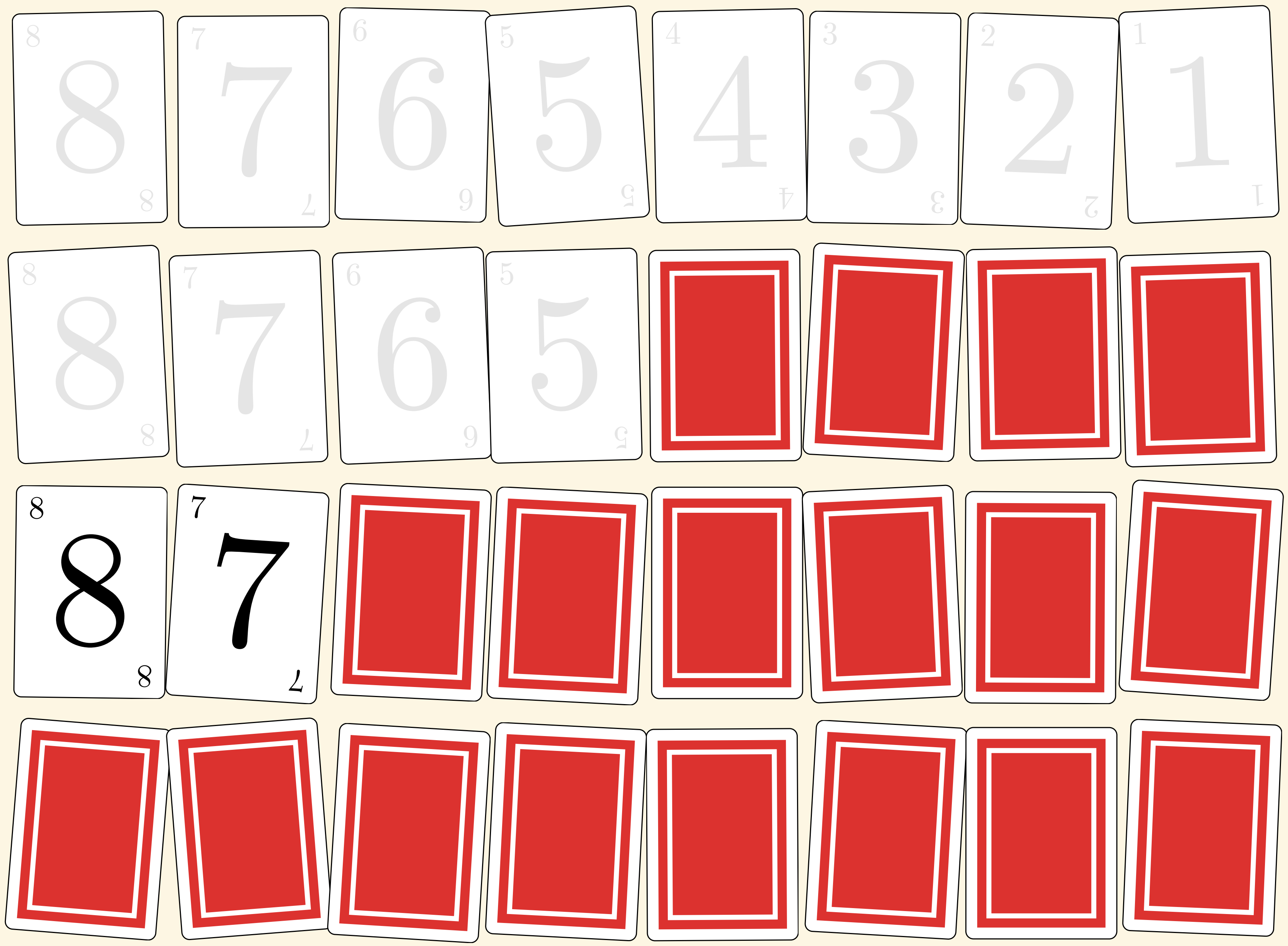
Skal sortere A[1 . . 2]



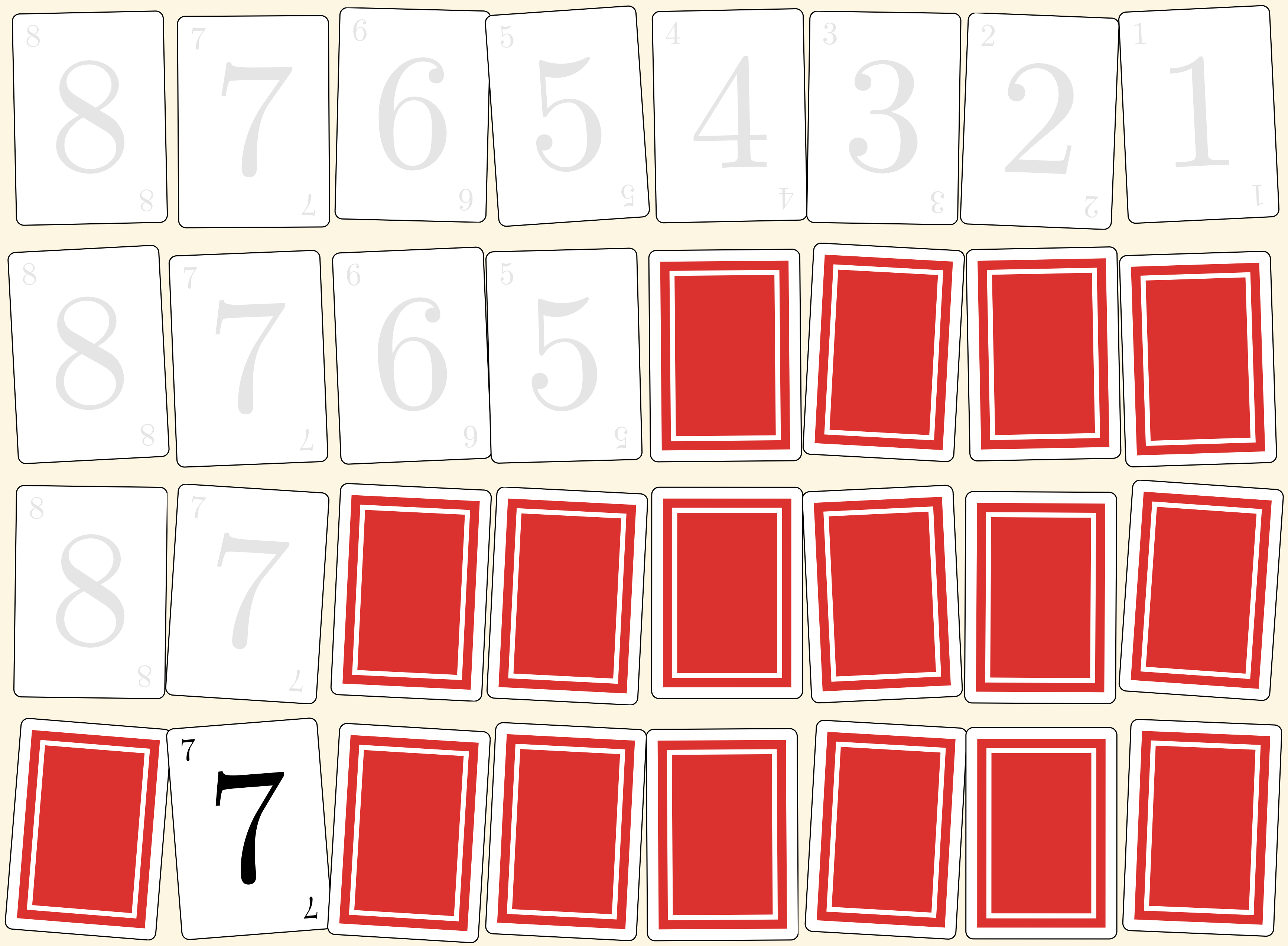
Sortér først A[1] og så A[2]



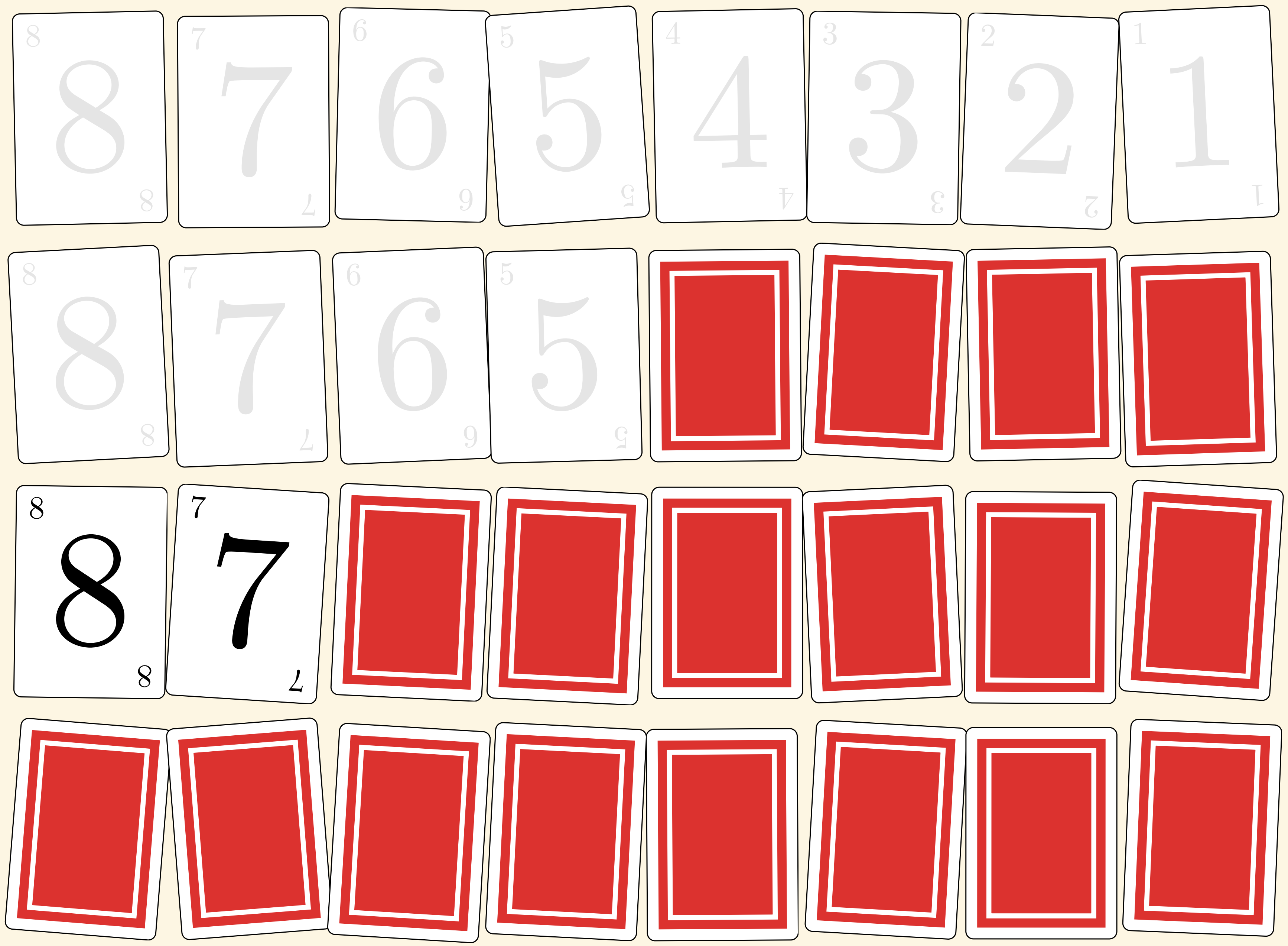
A[1] er sortert



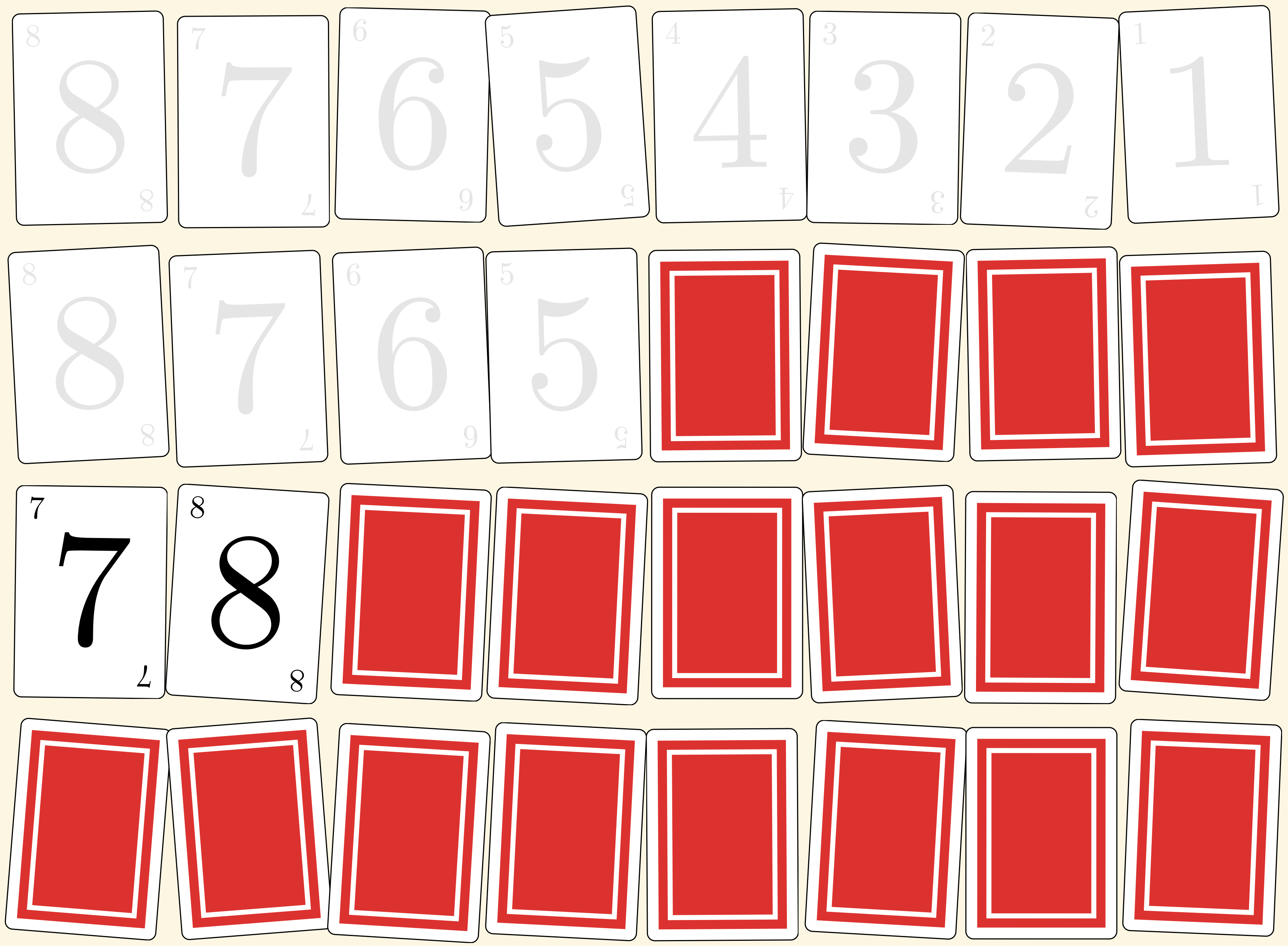
Sortér først A[1] og så A[2]



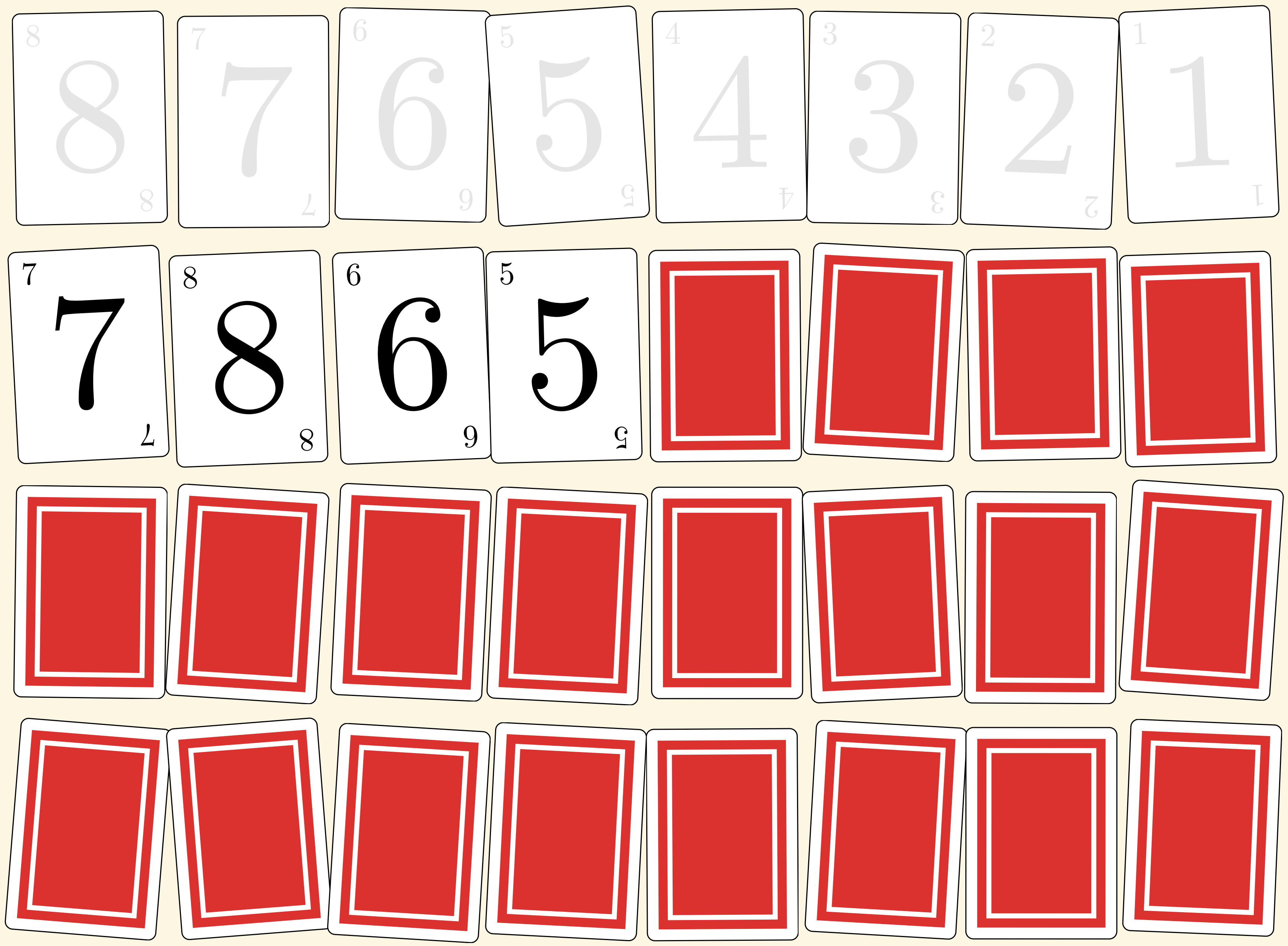
A[2] er sortert



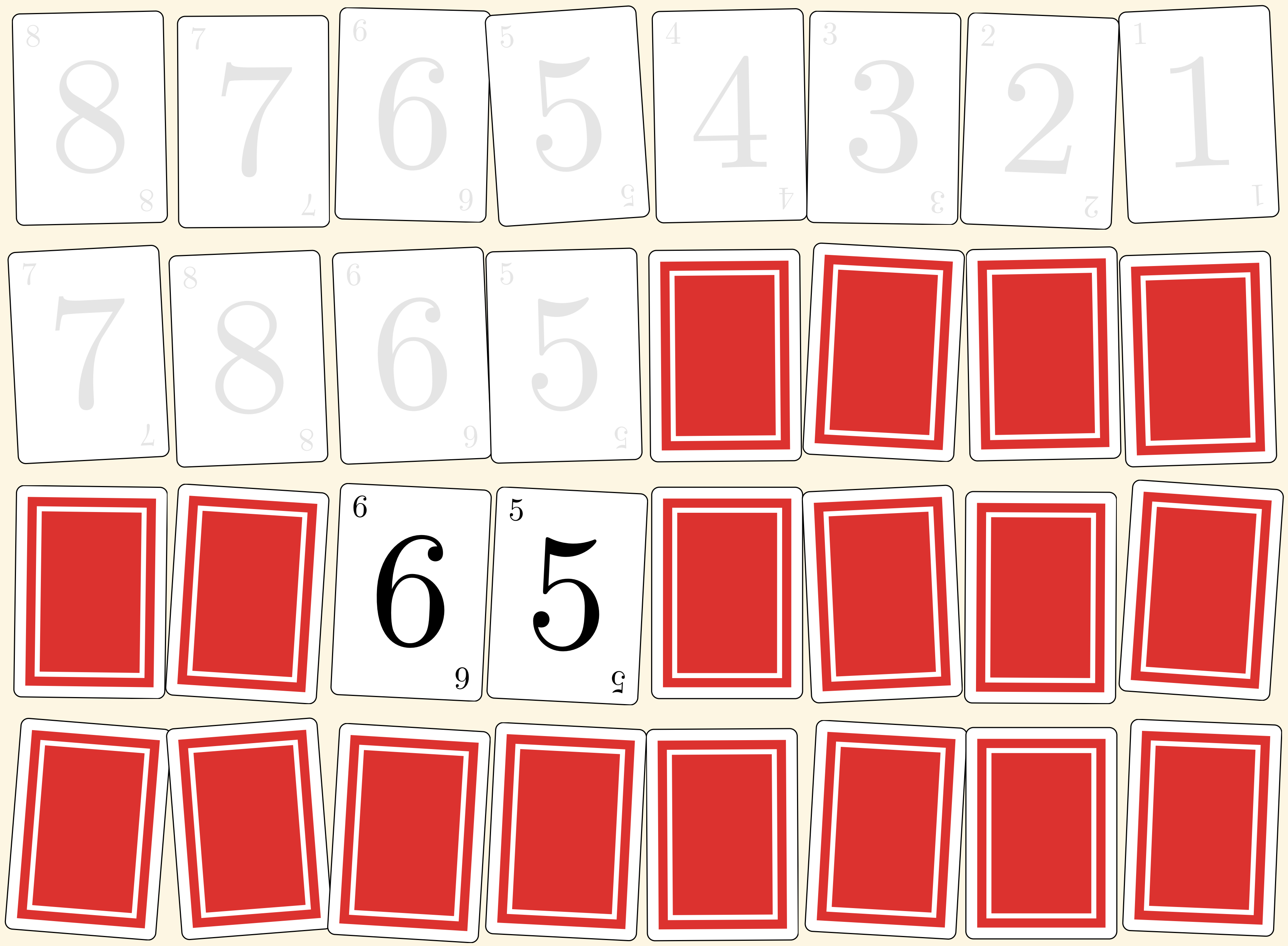
Flett sammen A[1] og A[2]



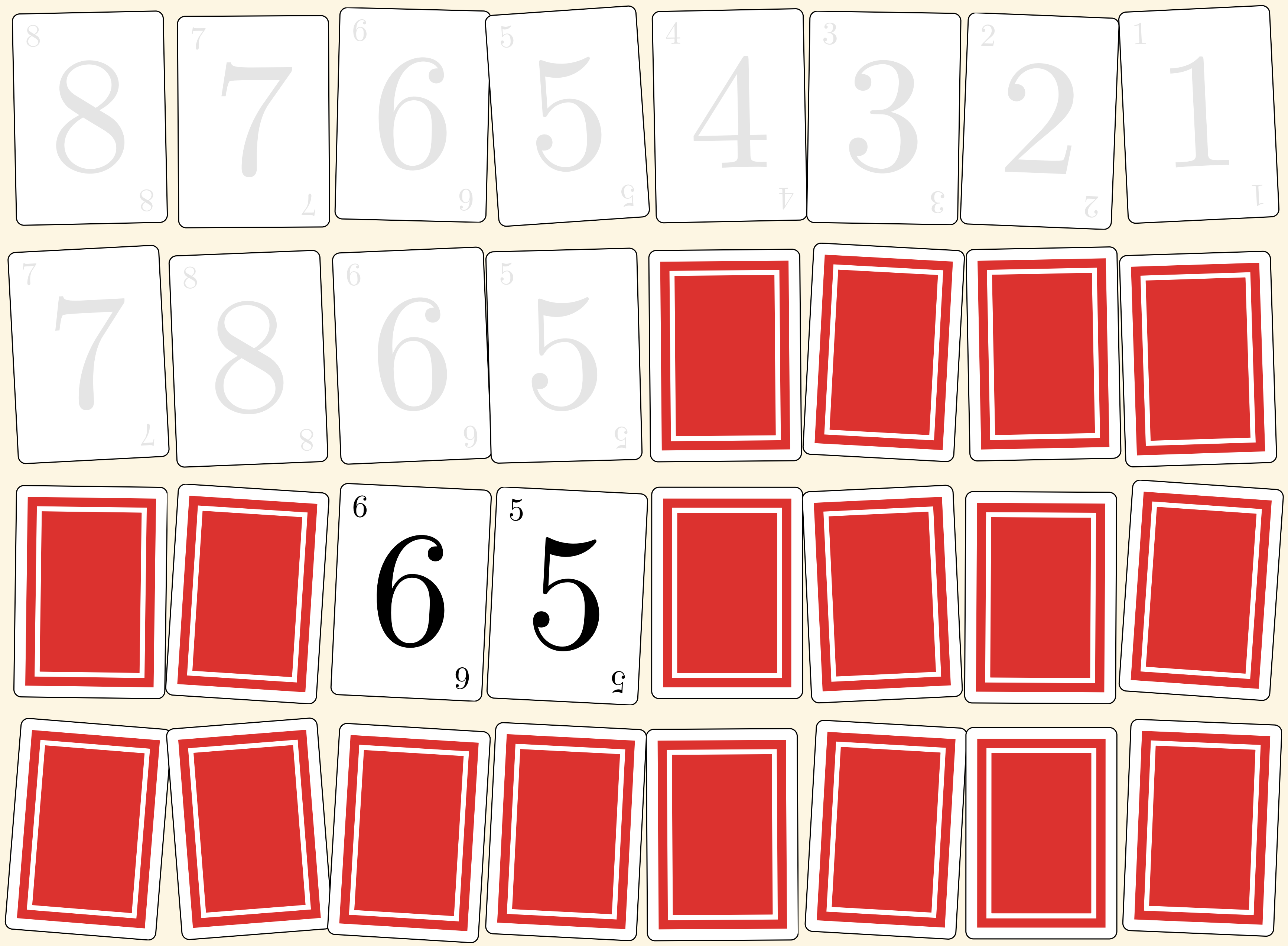
A[1 . . 2] er sortert



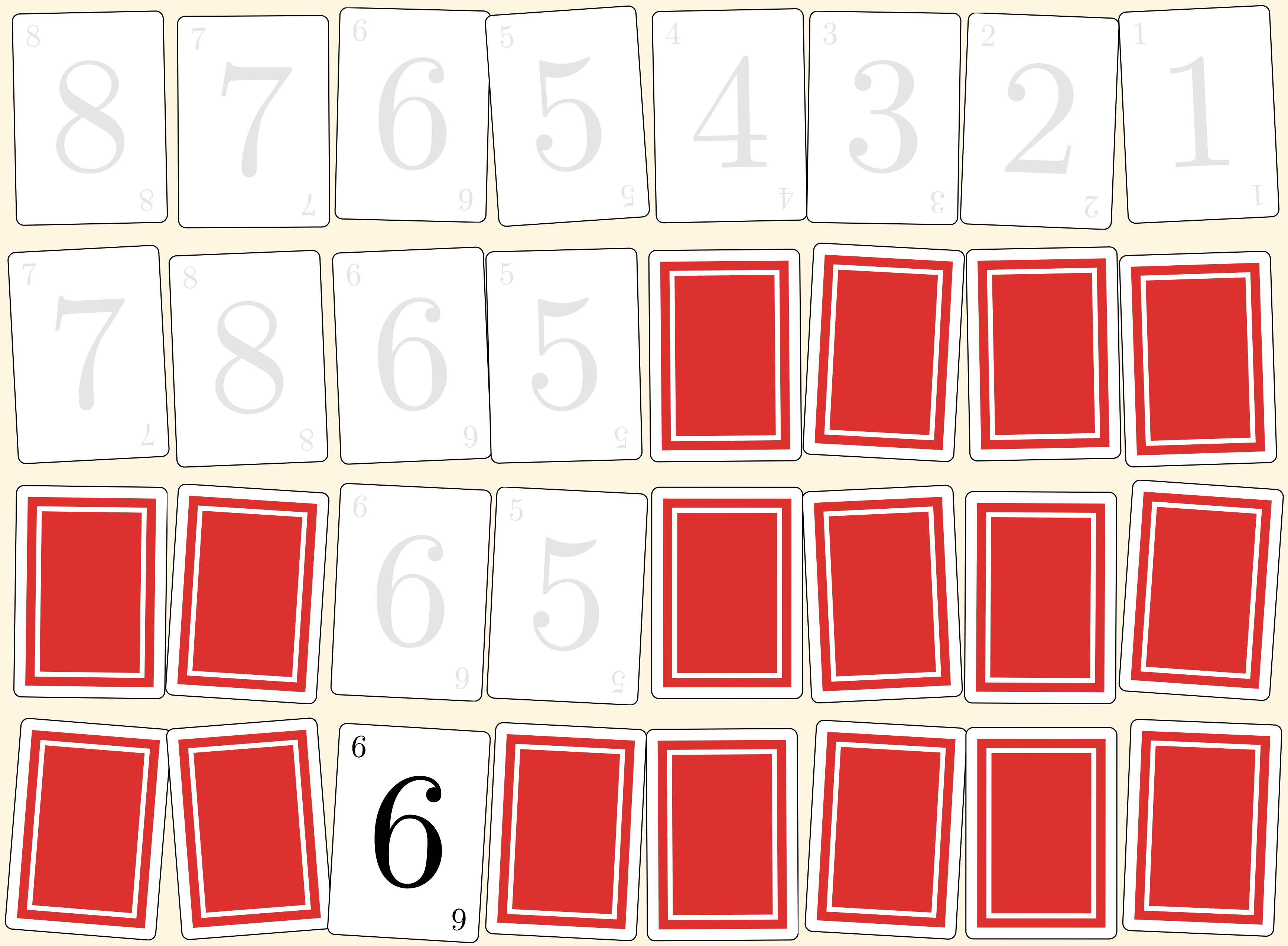
Sortér først A[1 . . 2] og så A[3 . . 4]



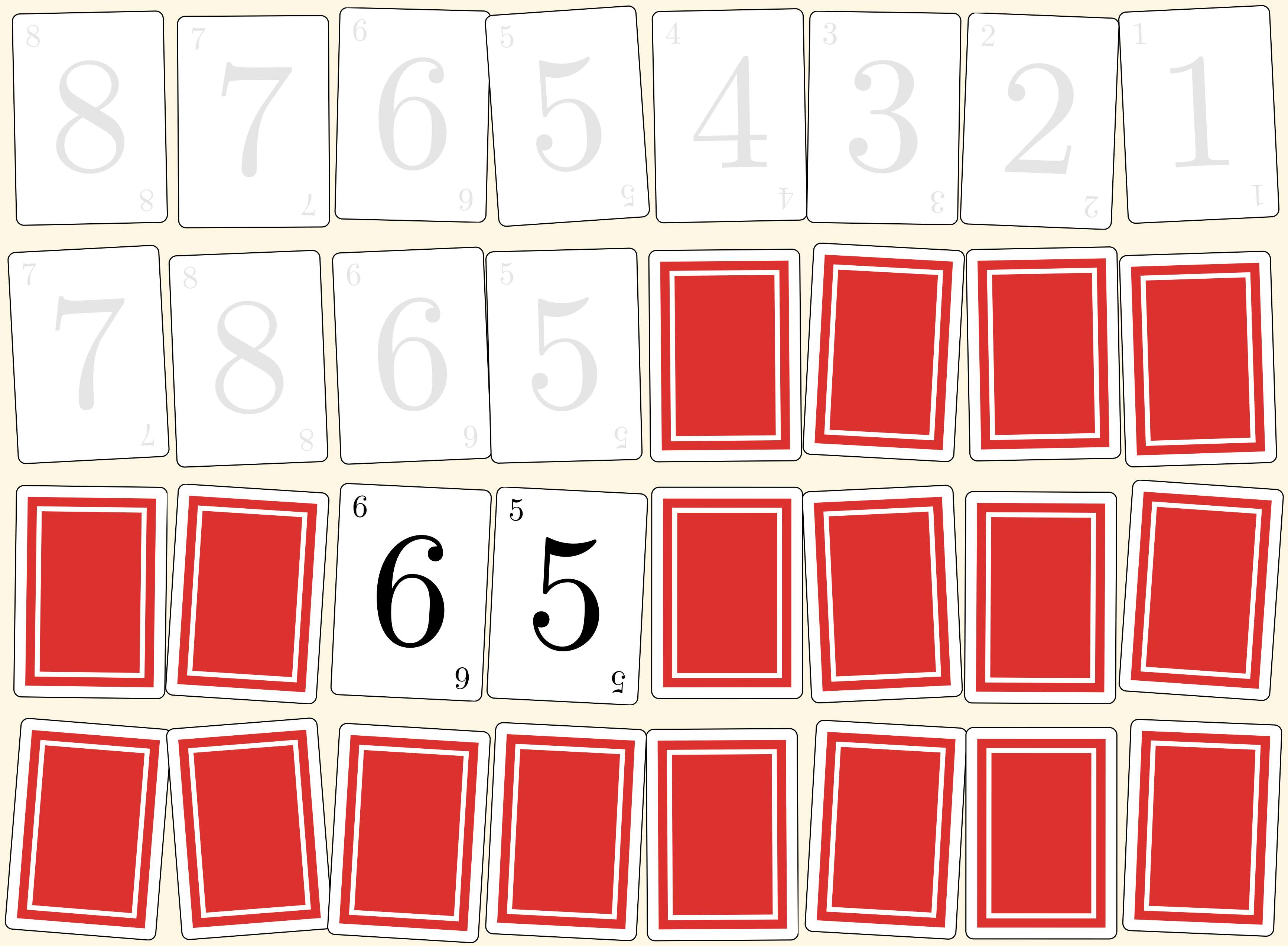
Skal sortere A[3 . . 4]



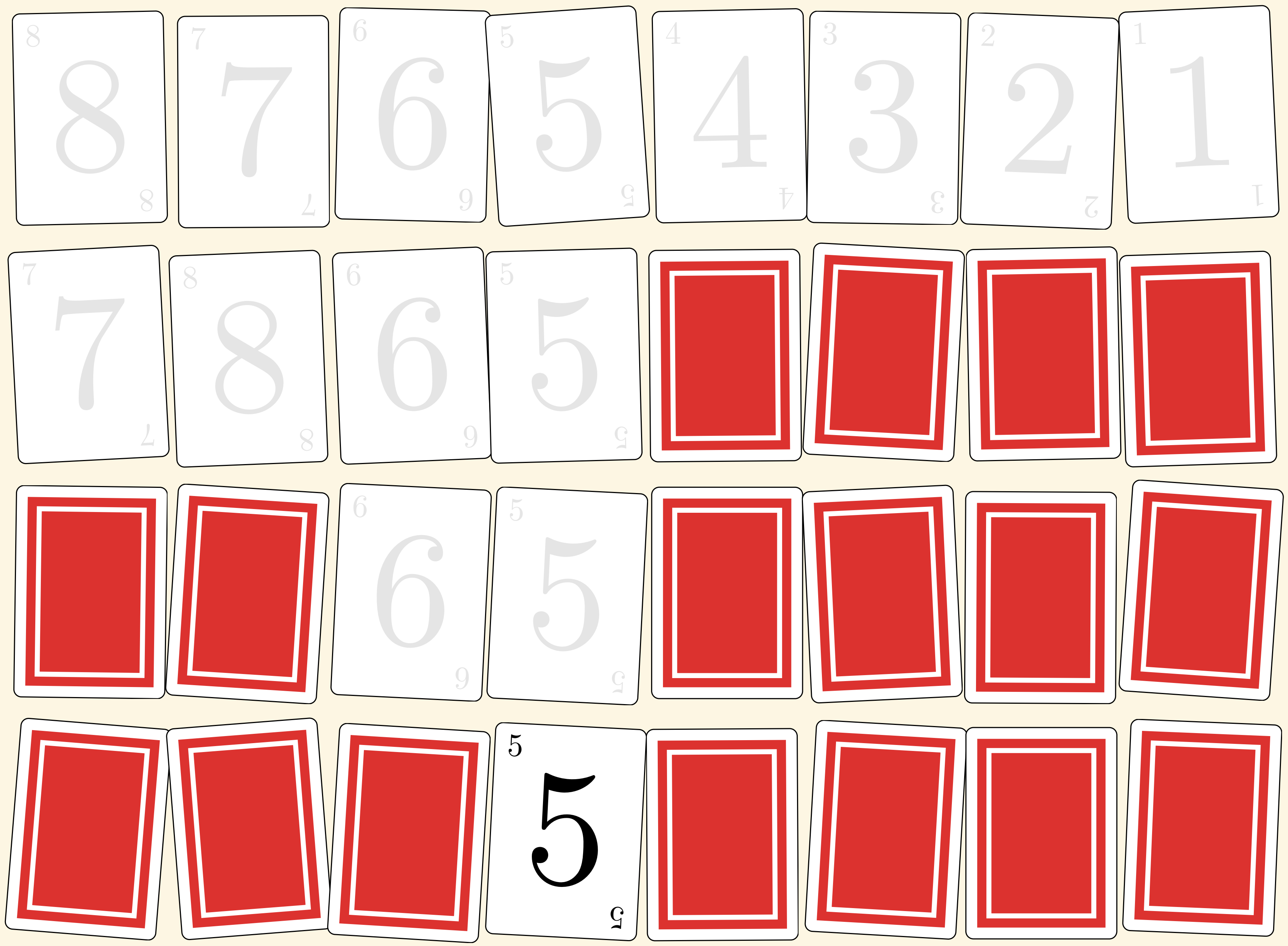
Sortér først A[3] og så A[4]



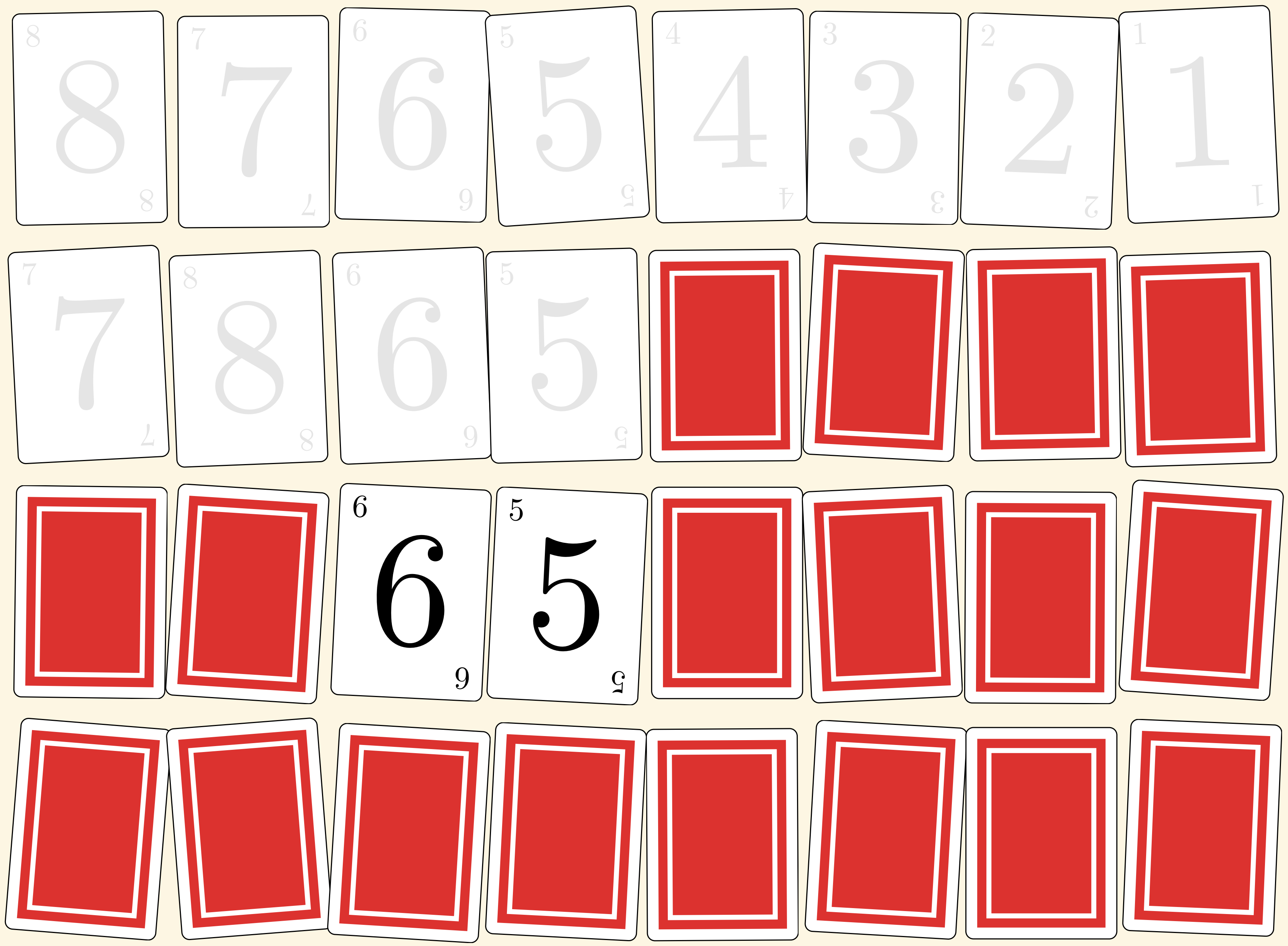
A[3] er sortert



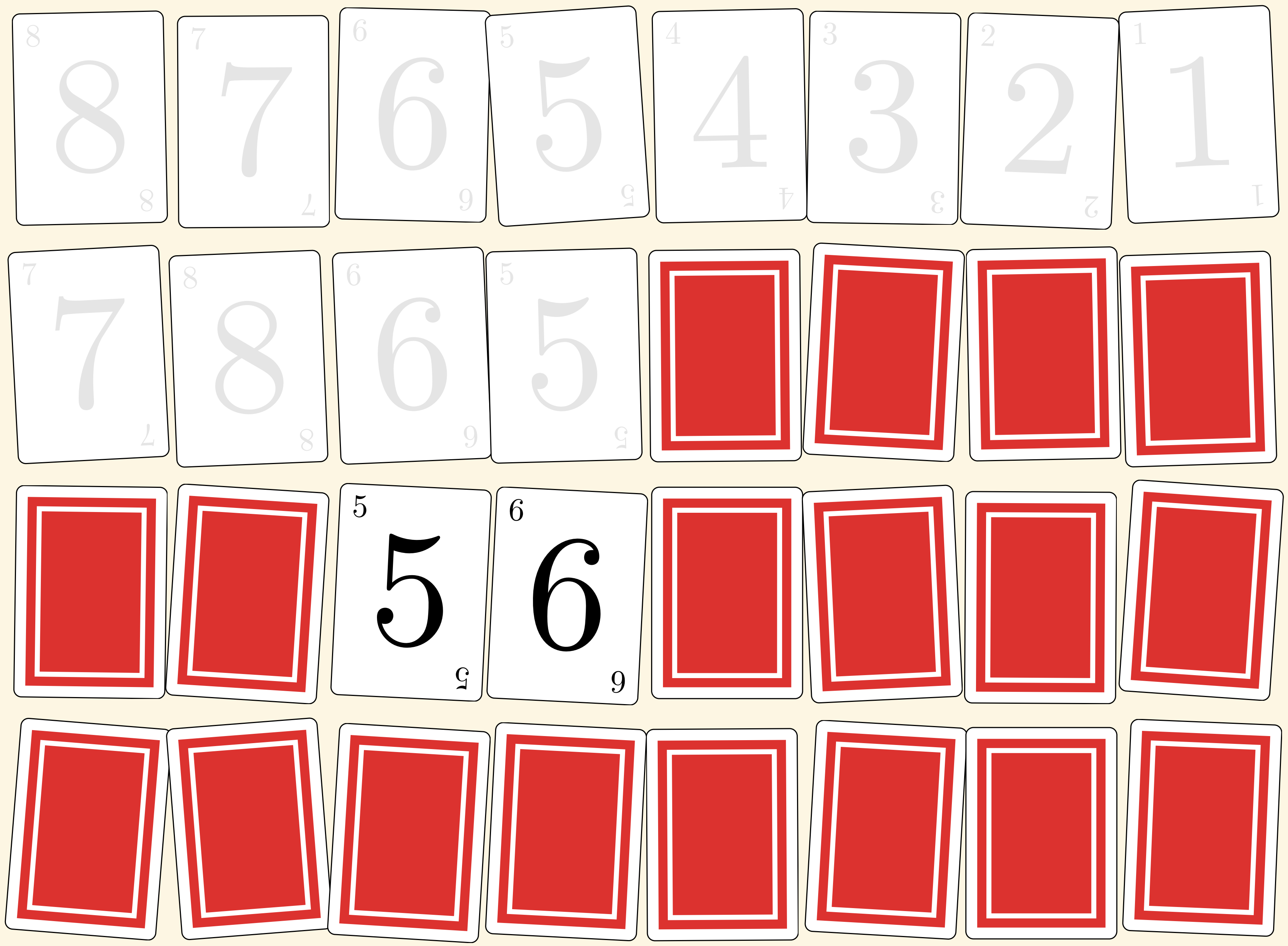
Sortér først A[3] og så A[4]



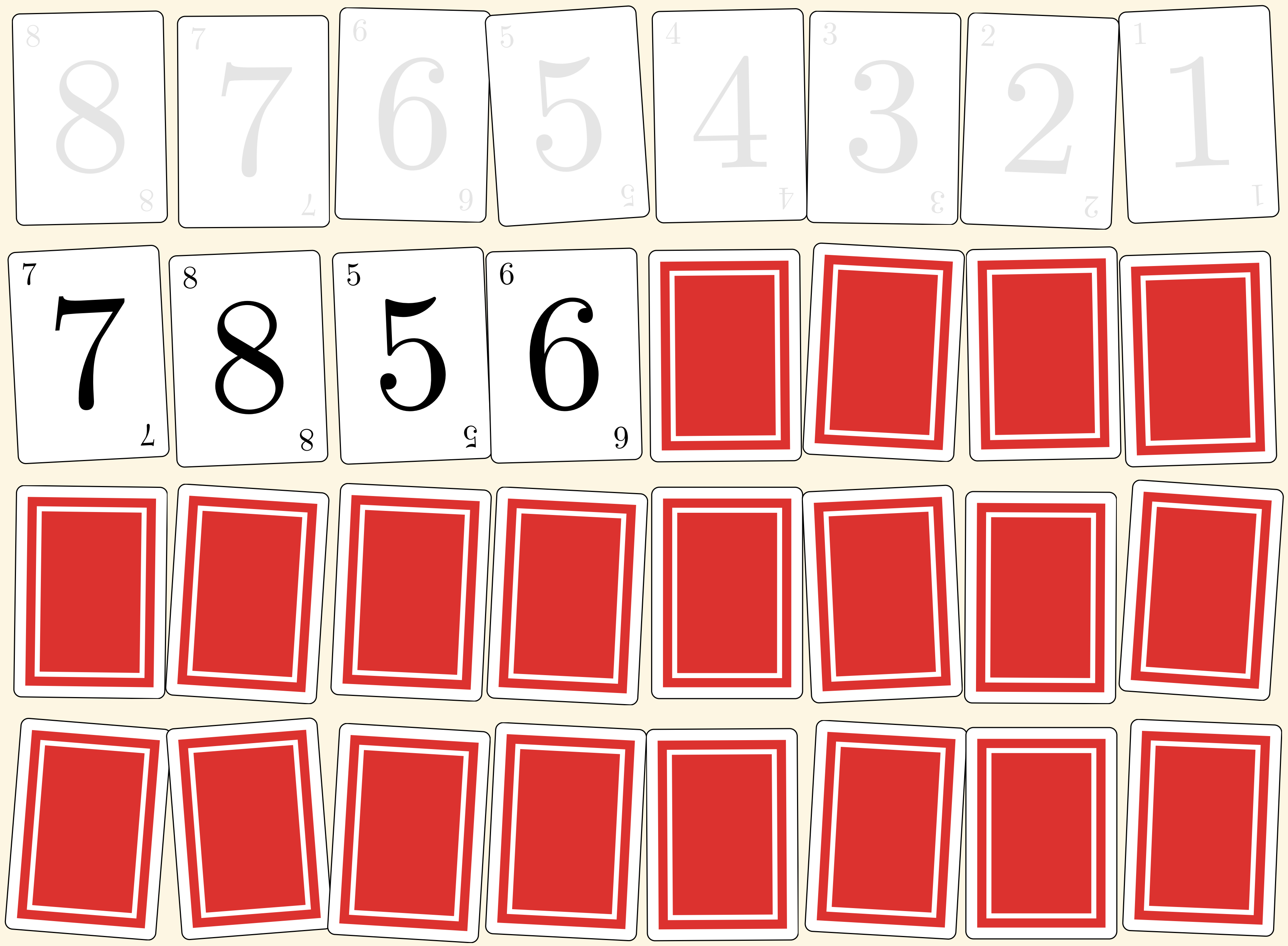
A[4] er sortert



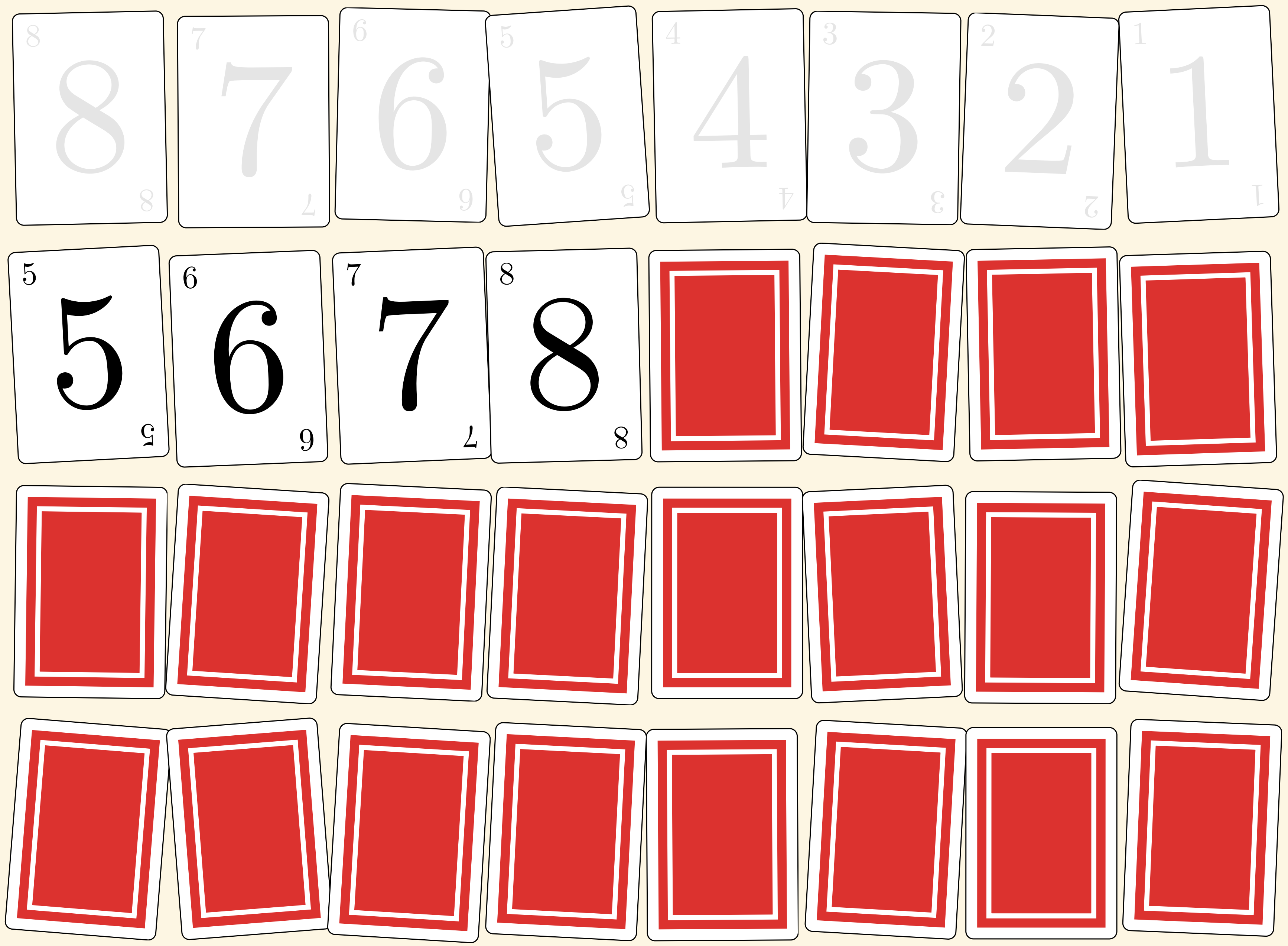
Flett sammen A[3] og A[4]



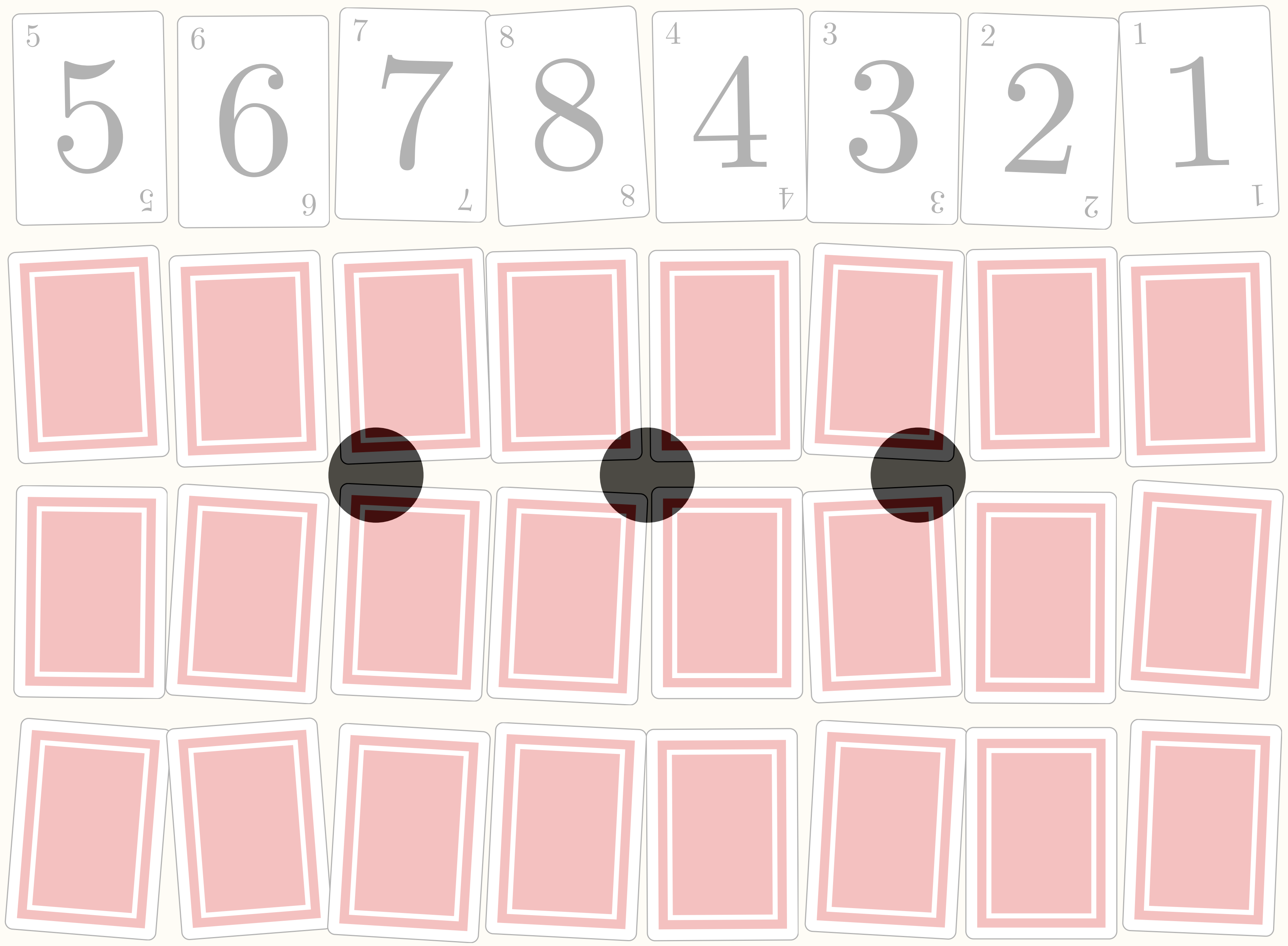
A[3 . . 4] er sortert



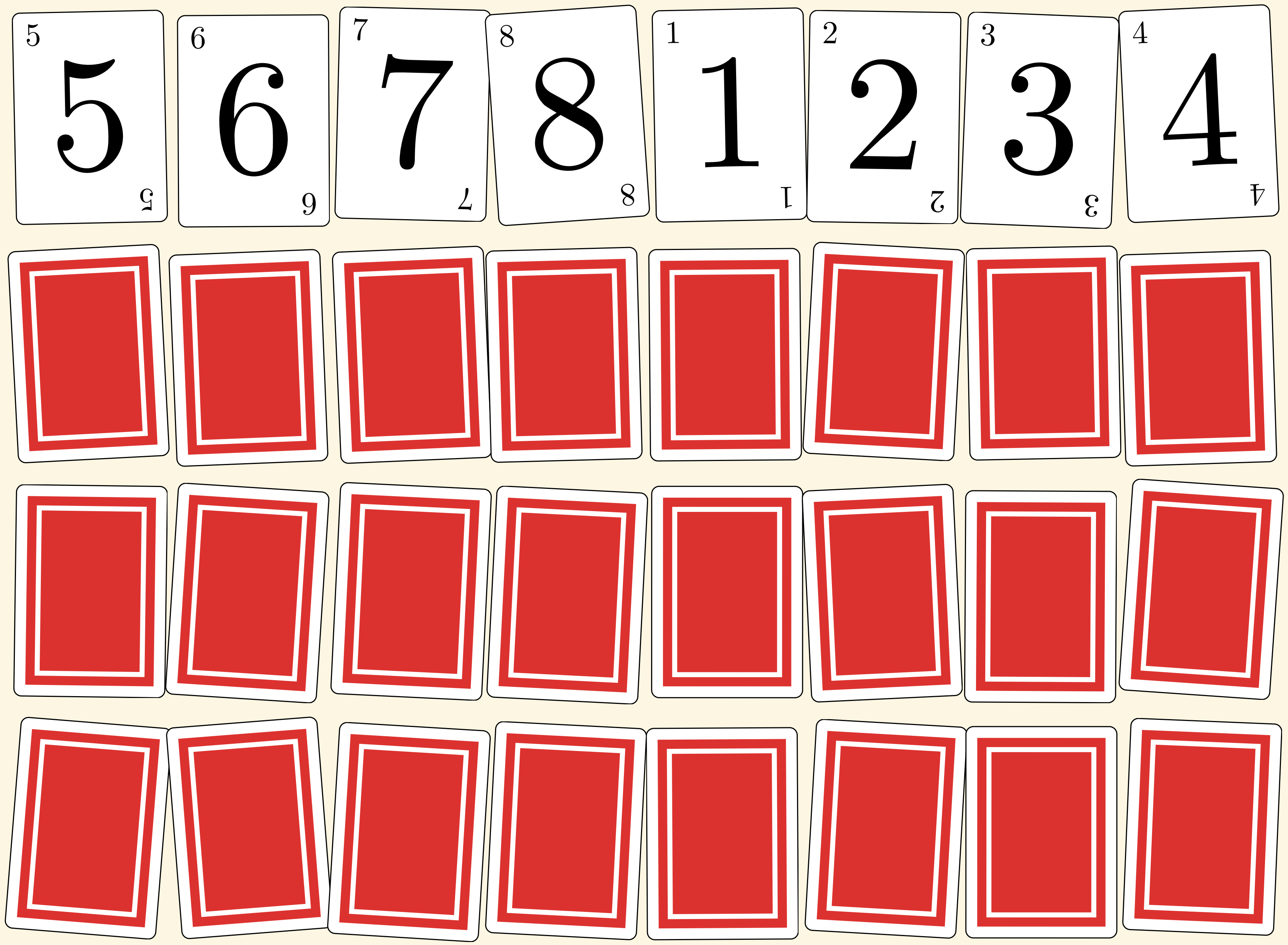
Flett sammen $A[1\,..\,2]$ og $A[3\,..\,4]$



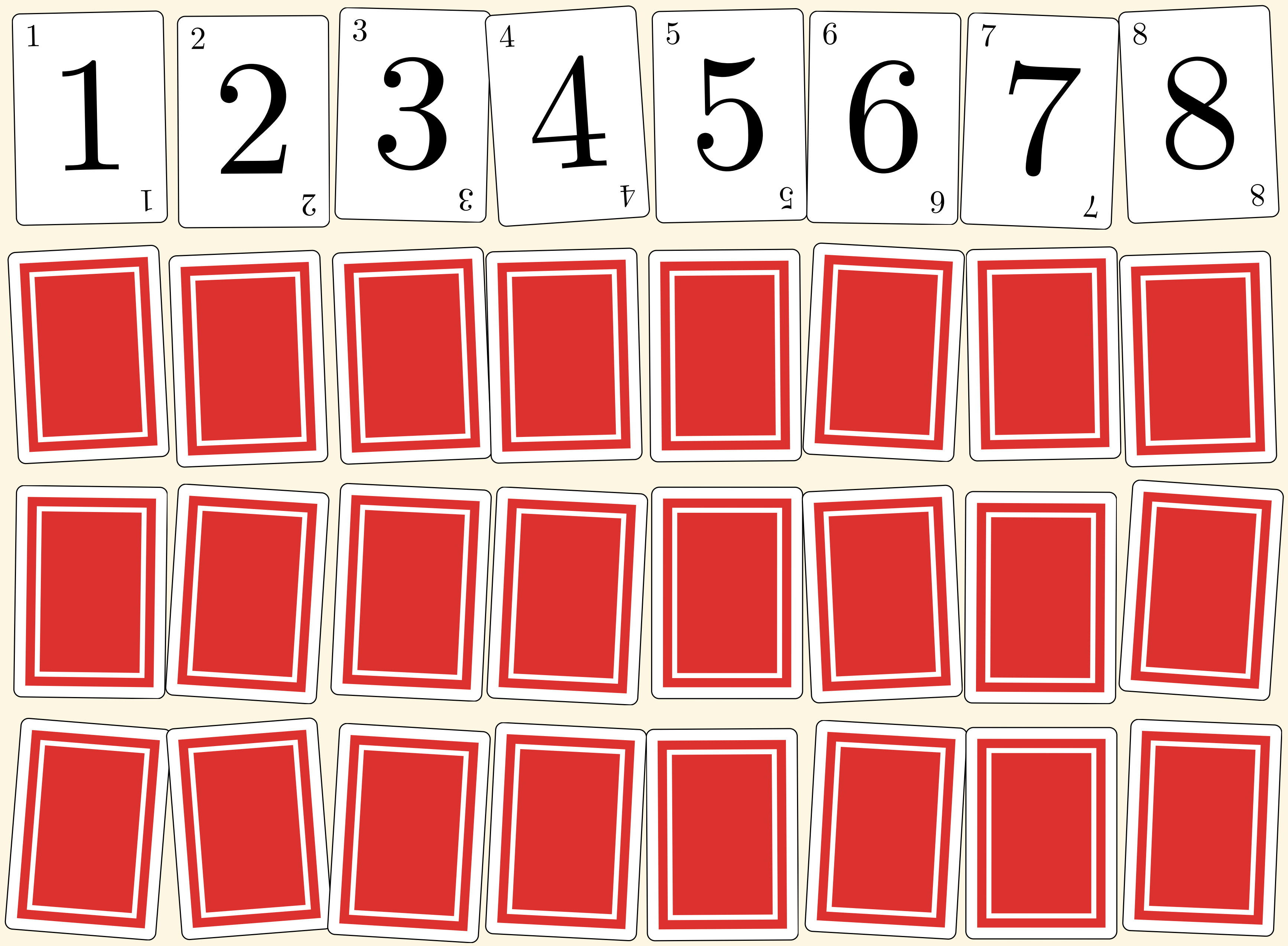
A[1 . . 4] er sortert



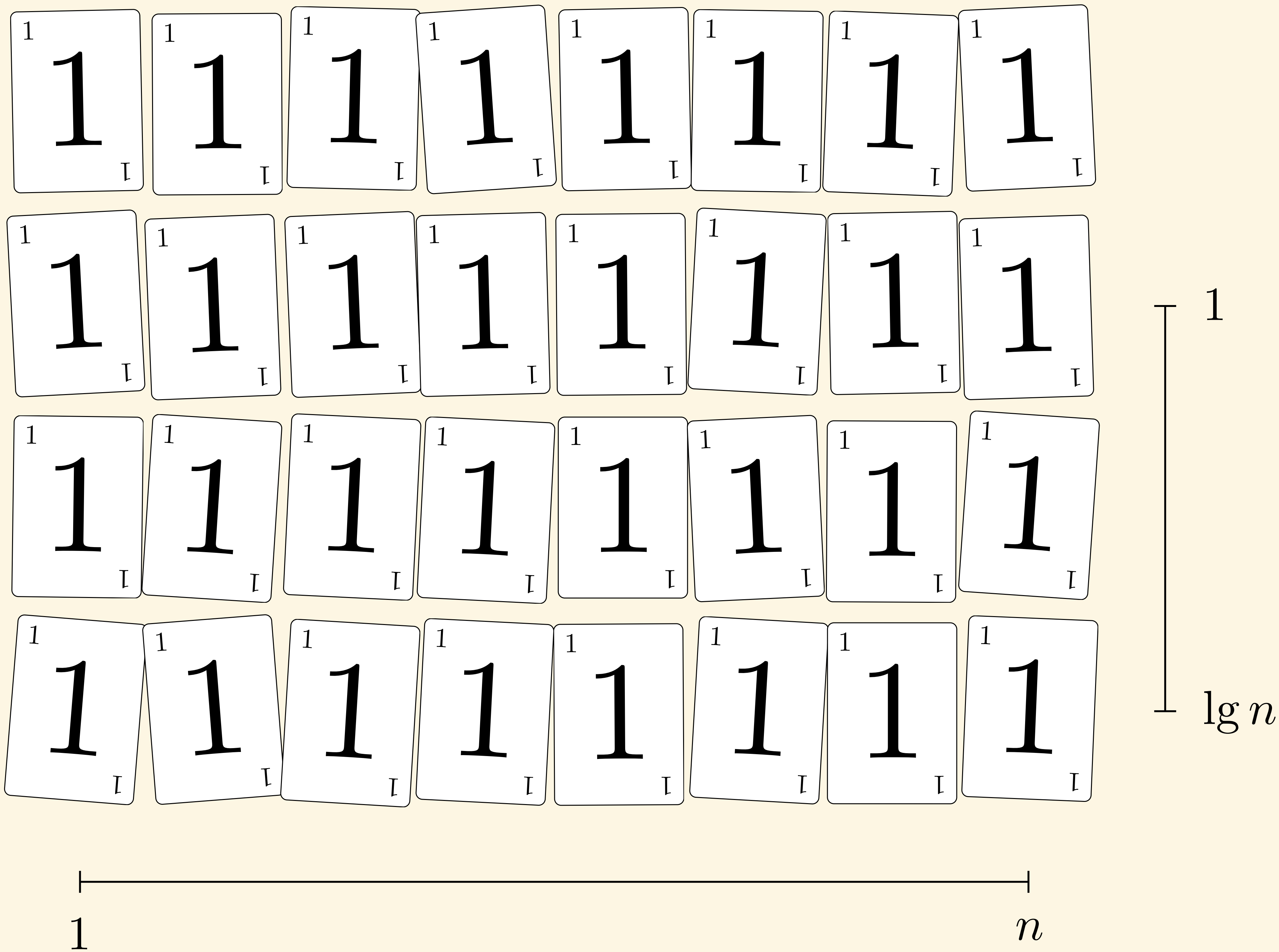
Sortér først A[1 . . 4] og så A[5 . . 8]



Flett sammen A[1 . . 4] og A[5 . . 8]



A[1 . . 8] er sortert


1
lg n
1
n

MERGE-SORT(A, p, r)

Sortér $A[p \mathinner{\ldotp\ldotp} r]$ ved å sortere halvdelene først

MERGE-SORT(A, p, r)
1 **if** $p < r$

Hvis A[$p\,..\,r$] har lengde minst 2 . . .

MERGE-SORT(A, p, r)
1 **if** $p < r$
2 $\quad q = \lfloor (p + r)/2 \rfloor$

Del på midten

Merge-Sort(A, p, r)
if $p < r$
    $q = \lfloor (p + r)/2 \rfloor$
    Merge-Sort(A, p, q)

Sortér $A[p \mathinner{..} q]$ rekursivt

```
Merge-Sort(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    Merge-Sort(A, p, q)
    Merge-Sort(A, q + 1, r)
```

Sortér A[$q + 1\,..\,r$] rekursivt

MERGE-SORT(A, p, r)

if $p < r$

  $q = \lfloor (p + r)/2 \rfloor$

  MERGE-SORT(A, p, q)

  MERGE-SORT$(A, q + 1, r)$

  MERGE(A, p, q, r)

Flett sorterte segmenter $A[p \,..\, q]$ og $A[q + 1 \,..\, r]$

MERGE-SORT(A, p, r)

```
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

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

$p, r \; = \; 1, 8$

```
MERGE-SORT(A, p, r)
  if p < r
      q = ⌊(p + r)/2⌋
      MERGE-SORT(A, p, q)
      MERGE-SORT(A, q + 1, r)
      MERGE(A, p, q, r)
```

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

$p, r \; = \; 1, 8$

```
MERGE-SORT(A, p, r)
  if p < r
      q = ⌊(p + r)/2⌋
      MERGE-SORT(A, p, q)
      MERGE-SORT(A, q + 1, r)
      MERGE(A, p, q, r)
```

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

$p, r \;=\; 1, 8$

MERGE-SORT(A, p, r)

if $p < r$

  $q = \lfloor (p + r)/2 \rfloor$

  MERGE-SORT(A, p, q)

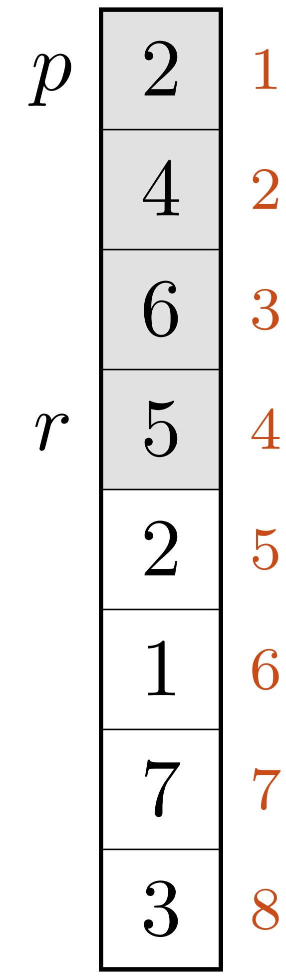
  MERGE-SORT$(\mathrm{A}, q + 1, r)$

  MERGE(A, p, q, r)

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

$p, r \; = \; 1, 8 \rangle 1, 4$

```
MERGE-SORT(A, p, r)
  if p < r
        q = ⌊(p + r)/2⌋
        MERGE-SORT(A, p, q)
        MERGE-SORT(A, q + 1, r)
        MERGE(A, p, q, r)
```



$p, r \;=\; 1, 8 \;›\; 1, 4$

Merge-Sort(A, p, r)
if $p < r$
  $q = \lfloor (p + r)/2 \rfloor$
  Merge-Sort(A, p, q)
  Merge-Sort$(A, q + 1, r)$
  Merge(A, p, q, r)

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

$p, r \;=\; 1, 8 \;›\; 1, 4$

MERGE-SORT(A, p, r)

```
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

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

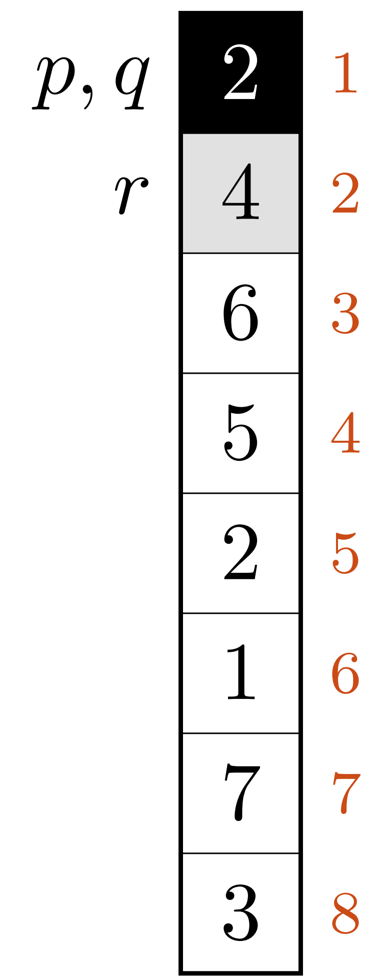
$p, r \;=\; 1, 8 \;›\; 1, 4 \;›\; 1, 2$

```
MERGE-SORT(A, p, r)
  if p < r
      q = ⌊(p + r)/2⌋
      MERGE-SORT(A, p, q)
      MERGE-SORT(A, q + 1, r)
      MERGE(A, p, q, r)
```

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

$p, r \;=\; 1, 8 \;›\; 1, 4 \;›\; 1, 2$

MERGE-SORT(A, p, r)
if $p < r$
  $q = \lfloor (p + r)/2 \rfloor$
  MERGE-SORT(A, p, q)
  MERGE-SORT(A, $q + 1$, r)
  MERGE(A, p, q, r)



$p, r \;=\; 1, 8 \;›\; 1, 4 \;›\; 1, 2$

Merge-Sort(A, p, r)

```
if p < r
    q = ⌊(p + r)/2⌋
    Merge-Sort(A, p, q)
    Merge-Sort(A, q + 1, r)
    Merge(A, p, q, r)
```

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

$p, r \;=\; 1, 8 \;›\; 1, 4 \;›\; 1, 2 \;›\; 1, 1$

Merge-Sort(A, p, r)
if $p < r$
 $q = \lfloor (p+r)/2 \rfloor$
 Merge-Sort(A, p, q)
 Merge-Sort(A, $q+1$, r)
 Merge(A, p, q, r)

$p, r \;=\; 1, 8 \,›\, 1, 4 \,›\, 1, 2$

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

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

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

$p, r \;=\; 1, 8$ › $1, 4$ › $1, 2$ › $2, 2$

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

```
Merge-Sort(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    Merge-Sort(A, p, q)
    Merge-Sort(A, q + 1, r)
    Merge(A, p, q, r)
```

$p, r \;=\; 1, 8 \;›\; 1, 4 \;›\; 1, 2$

Merge-Sort(A, p, r)

```
if p < r
    q = ⌊(p + r)/2⌋
    Merge-Sort(A, p, q)
    Merge-Sort(A, q + 1, r)
    Merge(A, p, q, r)
```

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

$p, r \;=\; 1, 8 \;›\; 1, 4$

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

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

$p, r \; = \; 1, 8 \;›\; 1, 4 \;›\; 3, 4$

Merge-Sort(A, p, r)

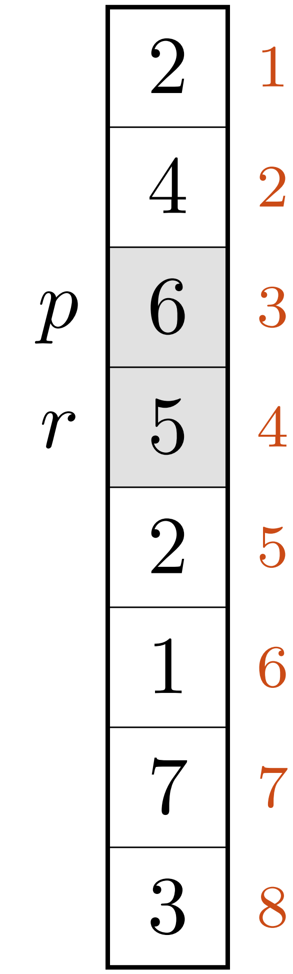
if $p < r$

$\quad q = \lfloor (p + r)/2 \rfloor$

$\quad$Merge-Sort(A, p, q)

$\quad$Merge-Sort$(A, q + 1, r)$

$\quad$Merge(A, p, q, r)



$p, r \;=\; 1, 8 \;›\; 1, 4 \;›\; 3, 4$

```
MERGE-SORT(A, p, r)
if p < r
        q = ⌊(p + r)/2⌋
        MERGE-SORT(A, p, q)
        MERGE-SORT(A, q + 1, r)
        MERGE(A, p, q, r)
```

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

$p, r \;=\; 1, 8 \;›\; 1, 4 \;›\; 3, 4$

Merge-Sort(A, p, r)

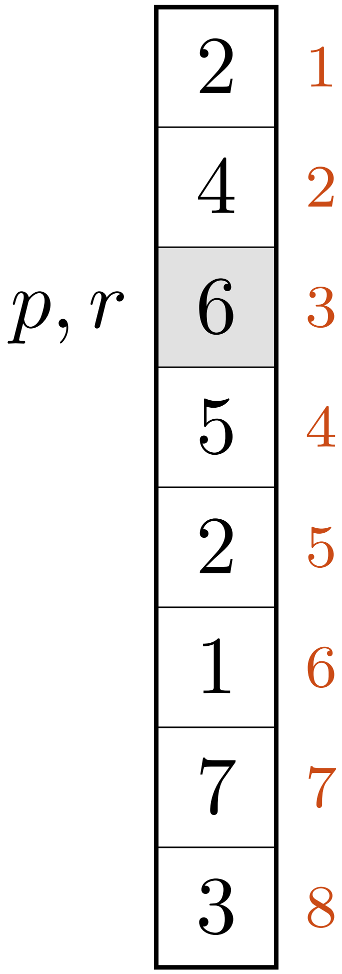
if $p < r$

  $q = \lfloor (p + r)/2 \rfloor$

  Merge-Sort(A, p, q)

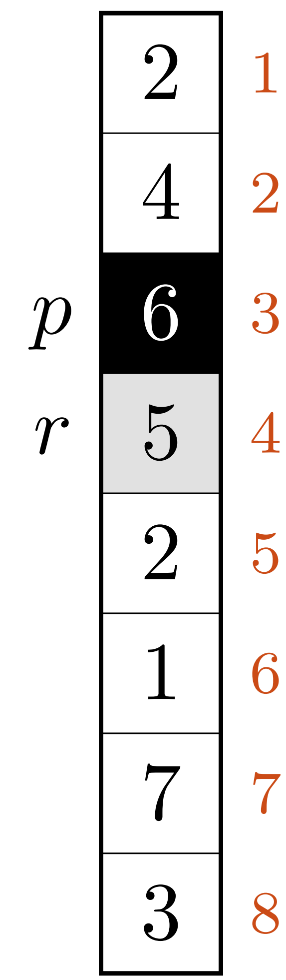
  Merge-Sort$(A, q + 1, r)$

  Merge(A, p, q, r)



$p, r \;=\; 1, 8 \;›\; 1, 4 \;›\; 3, 4 \;›\; \mathbf{3, 3}$

```
MERGE-SORT(A, p, r)
if p < r
        q = ⌊(p + r)/2⌋
        MERGE-SORT(A, p, q)
        MERGE-SORT(A, q + 1, r)
        MERGE(A, p, q, r)
```



$p, r \;=\; 1, 8 \;›\; 1, 4 \;›\; 3, 4$

Merge-Sort(A, p, r)

```
if p < r
        q = ⌊(p + r)/2⌋
        Merge-Sort(A, p, q)
        Merge-Sort(A, q + 1, r)
        Merge(A, p, q, r)
```

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

$p, r \;=\; 1, 8 \;›\; 1, 4 \;›\; 3, 4 \;›\; 4, 4$

Merge-Sort(A, p, r)
```
if p < r
    q = ⌊(p + r)/2⌋
    Merge-Sort(A, p, q)
    Merge-Sort(A, q + 1, r)
    Merge(A, p, q, r)
```

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

$p, r \;=\; 1,8 \;›\; 1,4 \;›\; 3,4$

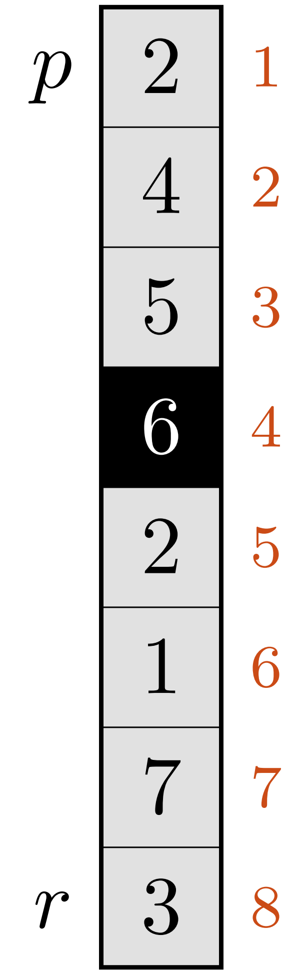
MERGE-SORT(A, p, r)

```
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

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

$p, r \;=\; 1, 8 \;›\; 1, 4$

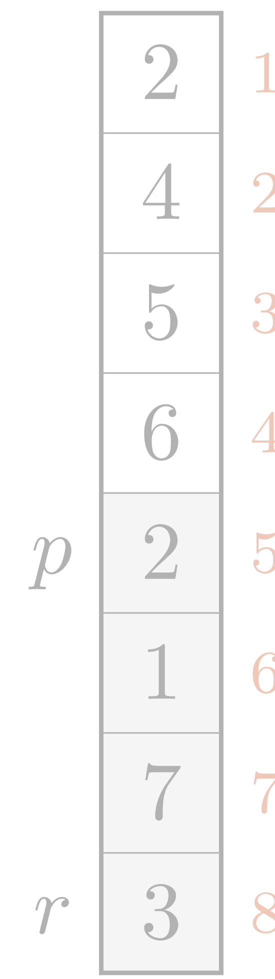
```
Merge-Sort(A, p, r)
  if p < r
        q = ⌊(p + r)/2⌋
        Merge-Sort(A, p, q)
        Merge-Sort(A, q + 1, r)
        Merge(A, p, q, r)
```



$p, r \;=\; 1, 8$

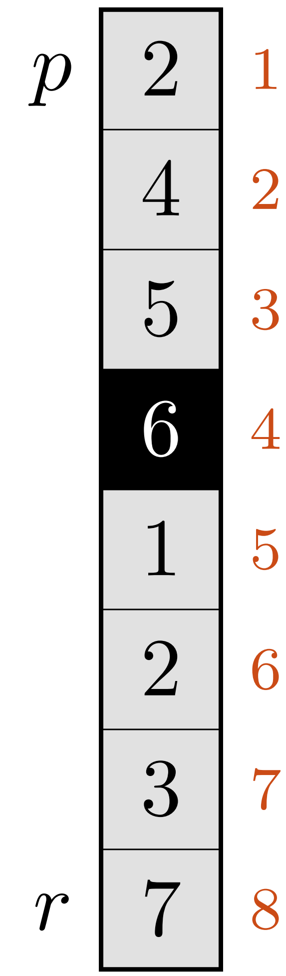
Merge-Sort(A, p, r)

```
if p < r
    q = ⌊(p + r)/2⌋
    Merge-Sort(A, p, q)
    Merge-Sort(A, q + 1, r)
    Merge(A, p, q, r)
```



$p, r \;=\; 1, 8 \;›\; 5, 8$

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```



$p, r = 1, 8$

Merge-Sort(A, p, r)

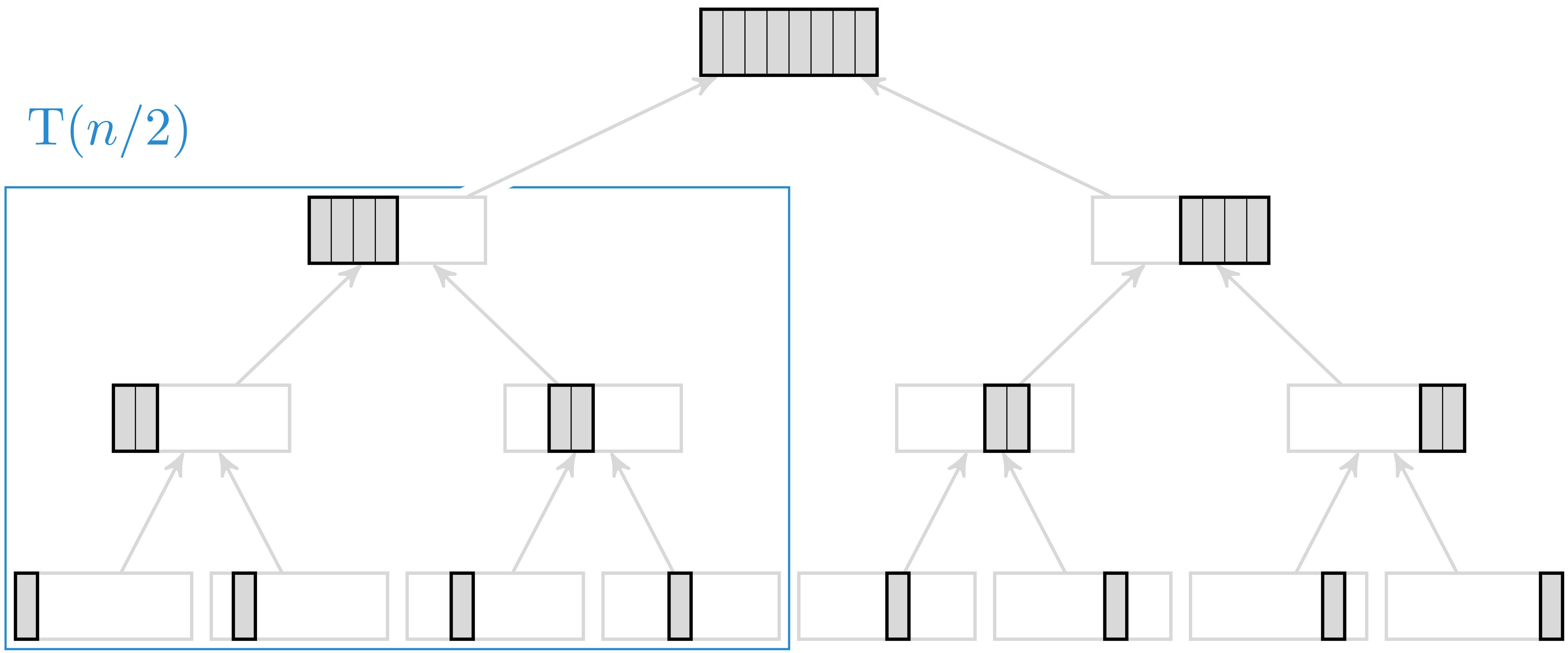
if $p < r$
    $q = \lfloor (p + r)/2 \rfloor$
    Merge-Sort(A, p, q)
    Merge-Sort(A, $q + 1$, r)
    Merge(A, p, q, r)

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

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

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

To rekursive kall (størrelse $n/2$) pluss n operasjoner


$T(n/2)$

$$\begin{aligned}\mathrm{T}(n) = n& \\ + 2\mathrm{T}(n/2)& \qquad (1)\end{aligned}$$

$$
\begin{aligned}
T(n) &= n \\
&\quad + 2T(n/2) \qquad (1)
\end{aligned}
$$

$$
\begin{aligned}
T(n) = n& \\
+ 2 \cdot n/2& \qquad (1) \\
+ 2 \cdot 2T(n/2/2)& \qquad (2)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= n \\
&\quad + n && (1) \\
&\quad + 4T(n/4) && (2)
\end{aligned}
$$

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

$$
\begin{aligned}
T(n) &= n \\
&\quad + n && (1) \\
&\quad + n && (2) \\
&\quad + n && (3) \\
&\quad \vdots && \vdots \\
&\quad + 2^{?} \cdot T(n/2^{?}) && (?)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= n \\
&+ n && (1) \\
&+ n && (2) \\
&+ n && (3) \\
&\vdots && \vdots \\
&+ 2^{?} \cdot T(n/2^{?}) && (\lg n)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= n \\
&\quad + n && (1) \\
&\quad + n && (2) \\
&\quad + n && (3) \\
&\quad \vdots && \vdots \\
&\quad + 2^{\lg n} \cdot \mathrm{T}(n/2^{\lg n}) && (\lg n)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= n \\
&\quad + n && (1) \\
&\quad + n && (2) \\
&\quad + n && (3) \\
&\quad \vdots && \vdots \\
&\quad + n \cdot \mathrm{T}(n/n) && (\lg n)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= n \\
&\quad + n && (1) \\
&\quad + n && (2) \\
&\quad + n && (3) \\
&\quad \vdots && \vdots \\
&\quad + n \cdot \mathrm{T}(1) && (\lg n)
\end{aligned}
$$

$$
\begin{aligned}
T(n) = {} & n \\
& + n && (1) \\
& + n && (2) \\
& + n && (3) \\
& \vdots && \vdots \\
& + n && (\lg n)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= n \\
&+ n & (1) \\
&+ n & (2) \\
&+ n & (3) \\
&\vdots & \vdots \\
&+ n & (\lg n)
\end{aligned}
$$

$$T(n) = n \lg n + n$$

Verifikasjon

Med substitusjon/induksjon

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

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

Gitt $T(k) = k \lg k + k$ for $k < n$, vis $T(n) = n \lg n + n$

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

Gitt $\mathrm{T}(k) = k \lg k + k$ for $k < n$, vis $\mathrm{T}(n) = n \lg n + n$

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

Gitt $\mathrm{T}(k) = k\lg k + k$ for $k < n$, vis $\mathrm{T}(n) = n\lg n + n$

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

Gitt $\mathrm{T}(k) = k\lg k + k$ for $k < n$, vis $\mathrm{T}(n) = n\lg n + n$

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

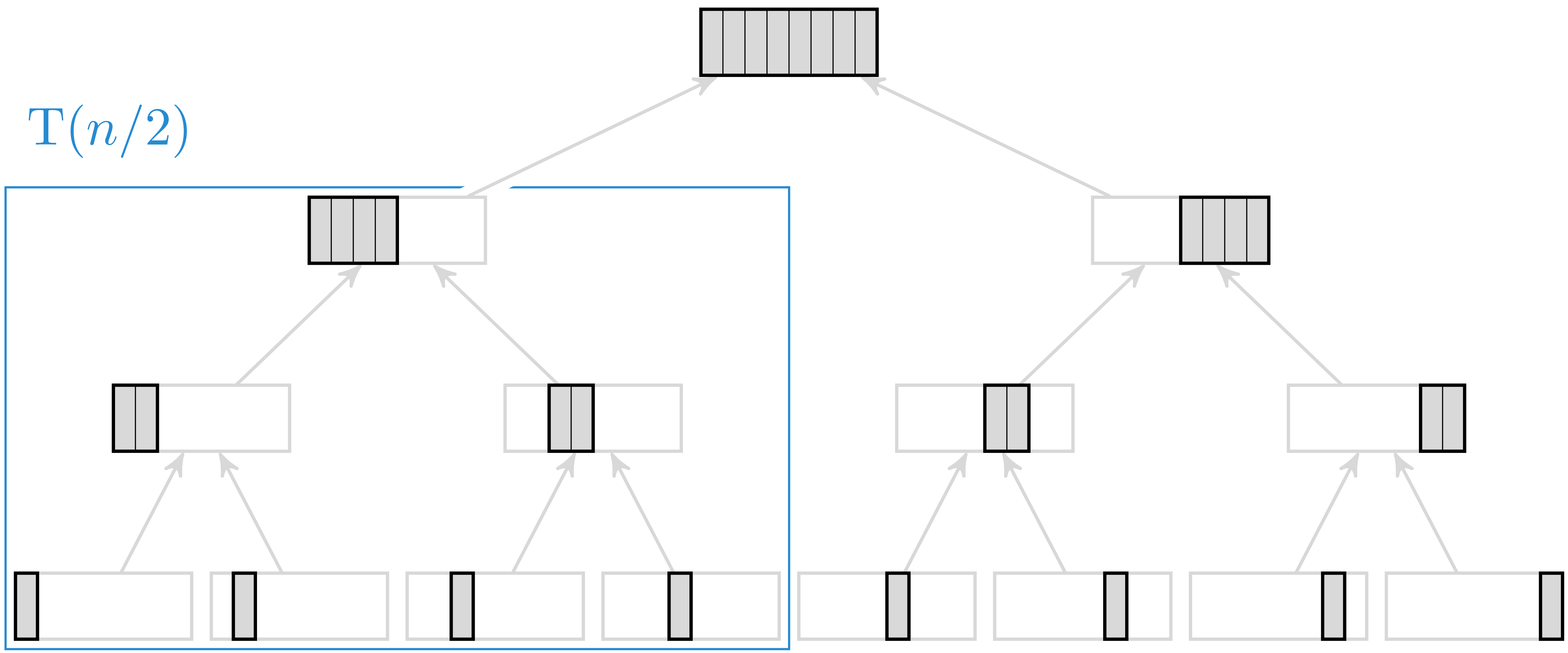
Gitt $\mathrm{T}(k) = k\lg k + k$ for $k < n$, vis $\mathrm{T}(n) = n\lg n + n$

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

Gitt $\mathrm{T}(k) = k\lg k + k$ for $k < n$, vis $\mathrm{T}(n) = n\lg n + n$

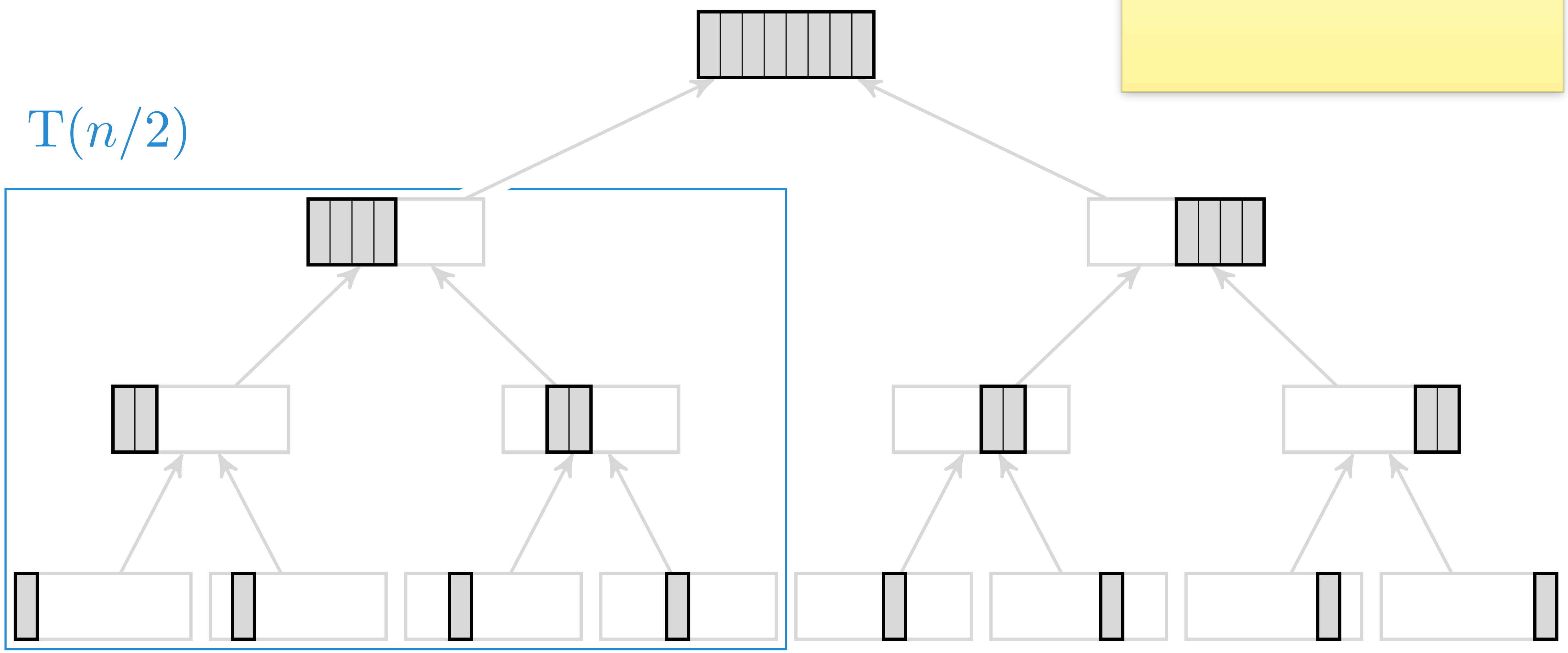
$$
\begin{aligned}
\mathrm{T}(n) &= 2\mathrm{T}(n/2) + n \\
&= 2\left(\frac{n}{2}\lg\frac{n}{2} + \frac{n}{2}\right) + n \\
&= n\lg\frac{n}{2} + n + n \\
&= n(\lg n - \lg 2) + 2n \\
&= n\lg n + n
\end{aligned}
$$

Gitt $\mathrm{T}(k) = k\lg k + k$ for $k < n$, vis $\mathrm{T}(n) = n\lg n + n$


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

Det er lg(n) + 1 rader, og summen av arbeidet i hver rad er n.

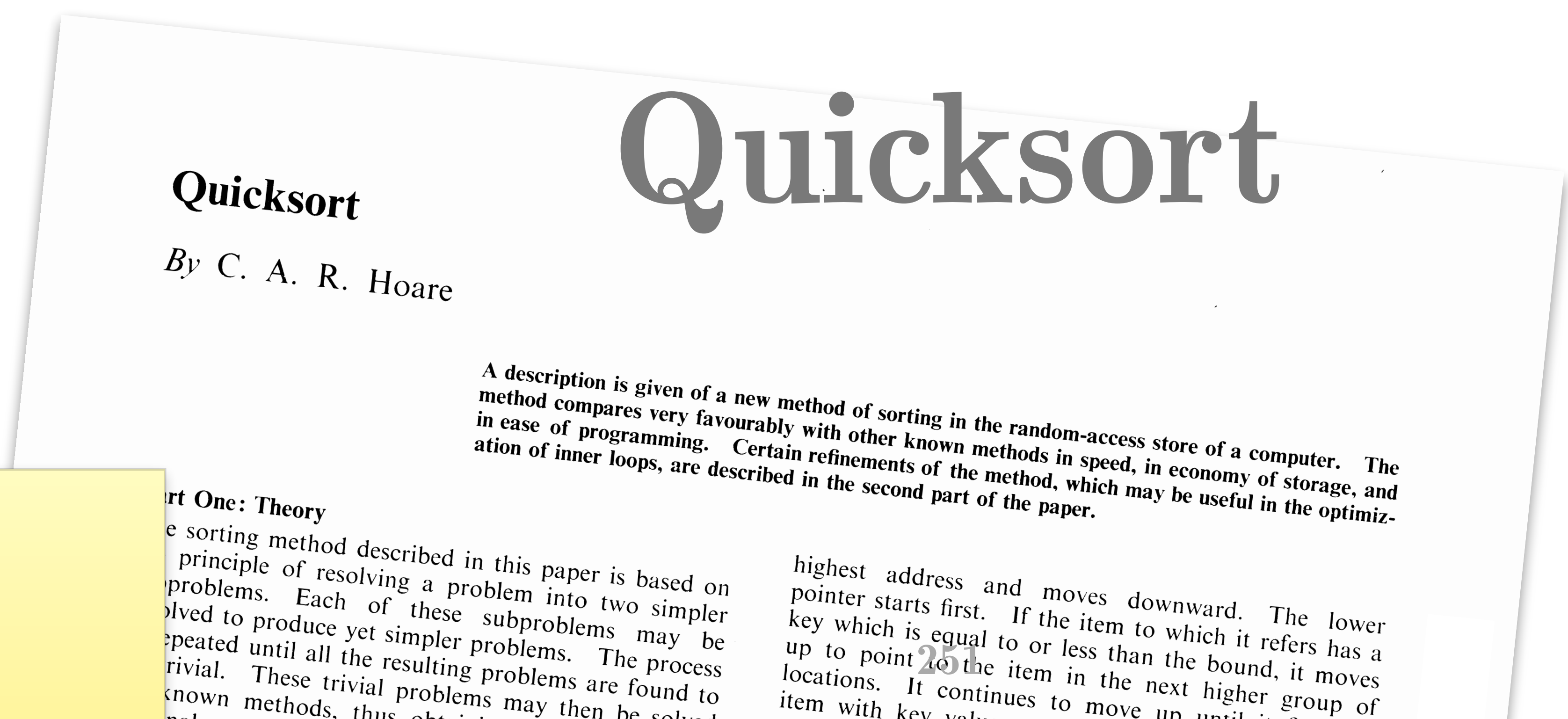
$\mathrm{T}(n/2)$



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

4:6

Quicksort



Quicksort

By C. A. R. Hoare

A description is given of a new method of sorting in the random-access store of a computer. The method compares very favourably with other known methods in speed, in economy of storage, and in ease of programming. Certain refinements of the method, which may be useful in the optimization of inner loops, are described in the second part of the paper.

rt One: Theory

e sorting method described in this paper is based on principle of resolving a problem into two simpler problems. Each of these subproblems may be olved to produce yet simpler problems. The process epeated until all the resulting problems are found to rivial. These trivial problems may then be solved known methods, thus obt...

highest address and moves downward. The lower pointer starts first. If the item to which it refers has a key which is equal to or less than the bound, it moves up to point to the item in the next higher group of locations. It continues to move up until it ... item with key val...

Fra 1962

251

- **Skill «små» og «store» tall**
- **Sortér små tall rekursivt**
- **Sortér store tall rekursivt**

- **Skill «små» og «store» tall**
- **Sortér små tall rekursivt**
- **Sortér store tall rekursivt**

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

A	tabell
p	venstre
r	høyre

Del to segmenter: Ett med små og ett med store verdier

PARTITION(A, p, r)
1 $x = A[r]$

A	tabell
p	venstre
r	høyre
x	splitt

Splittelement (*pivot*): Vilkårlig grense for hva vi kaller smått

$\textsc{Partition}(\mathrm{A}, p, r)$
1 $x = \mathrm{A}[r]$
2 $i = p - 1$

A	tabell
p	venstre
r	høyre
x	splitt
i	siste, små

Siste indeks i segmentet med små verdier. Foreløpig tomt

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

1 $x = \mathrm{A}[r]$

2 $i = p - 1$

3 **for** $j = p$ **to** $r - 1$

A	tabell
p	venstre
r	høyre
x	splitt
i	siste, små
j	siste, store

Siste indeks i segmentet med store verdier

Partition(A, p, r)

$x = A[r]$

$i = p - 1$

for $j = p$ **to** $r - 1$

  if $A[j] \leq x$

A	tabell
p	venstre
r	høyre
x	splitt
i	siste, små
j	siste, store

$A[j]$ er liten: Må flyttes til segmentet med små verdier

Partition(A, p, r)

$x = A[r]$
$i = p - 1$
for $j = p$ **to** $r - 1$
    if $A[j] \le x$
        $i = i + 1$

A	tabell
p	venstre
r	høyre
x	splitt
i	siste, små
j	siste, store

Utvid segmentet med små verdier. $A[i]$ foreløpig stor

```
PARTITION(A, p, r)
  x = A[r]
  i = p − 1
  for j = p to r − 1
      if A[j] ≤ x
          i = i + 1
          exchange A[i] with A[j]
```

A	tabell
p	venstre
r	høyre
x	splitt
i	siste, små
j	siste, store

Både A[i] og A[j] er i feil segment, så de bytter plass

Partition(A, p, r)

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

A	tabell
p	venstre
r	høyre
x	splitt
i	siste, små
j	siste, store

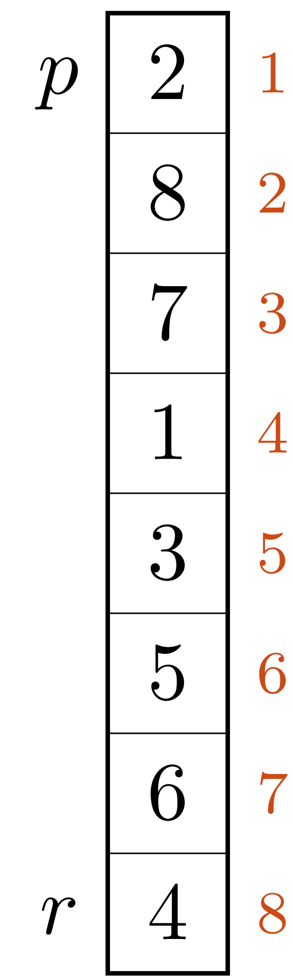
Splittelementet er nå på riktig (sortert) plass, mellom segmentene

```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

A	tabell
p	venstre
r	høyre
x	splitt
i	siste, små
j	siste, store

Returnér hvor splitten havnet

```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
        if A[j] ≤ x
                i = i + 1
                exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



x, i = -, -

```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
        if A[j] ≤ x
                i = i + 1
                exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

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

$x, i \;=\; 4, -$

```
PARTITION(A, p, r)
  x = A[r]
  i = p − 1
  for j = p to r − 1
        if A[j] ≤ x
              i = i + 1
              exchange A[i] with A[j]
  exchange A[i + 1] with A[r]
  return i + 1
```

$x, i \;=\; 4, 0$

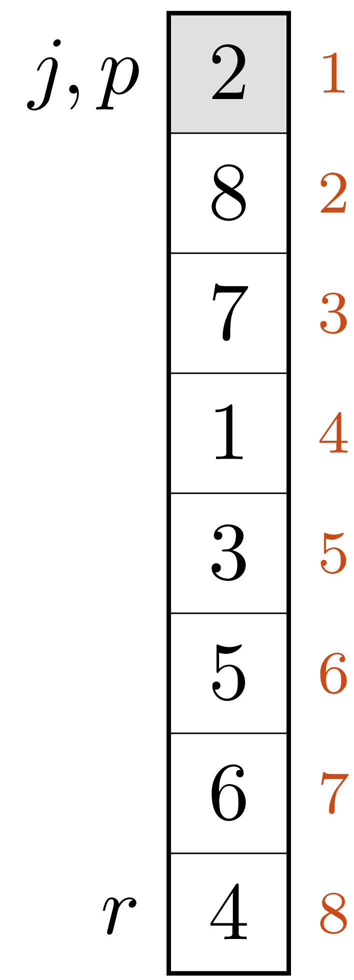
p	2	1
	8	2
	7	3
	1	4
	3	5
	5	6
	6	7
r	4	8

```
PARTITION(A, p, r)
  x = A[r]
  i = p − 1
  for j = p to r − 1
      if A[j] ≤ x
          i = i + 1
          exchange A[i] with A[j]
  exchange A[i + 1] with A[r]
  return i + 1
```

$x, i \;=\; 4, 0$

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

```
Partition(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
        if A[j] ≤ x
                i = i + 1
                exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



$x, i \;=\; 4, 0$

```
Partition(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
        if A[j] ≤ x
                i = i + 1
                exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

$x, i \;=\; 4, 1$

i, j, p	2	1
	8	2
	7	3
	1	4
	3	5
	5	6
	6	7
r	4	8

PARTITION(A, p, r)

```
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

i, j, p	2	1
	8	2
	7	3
	1	4
	3	5
	5	6
	6	7
r	4	8

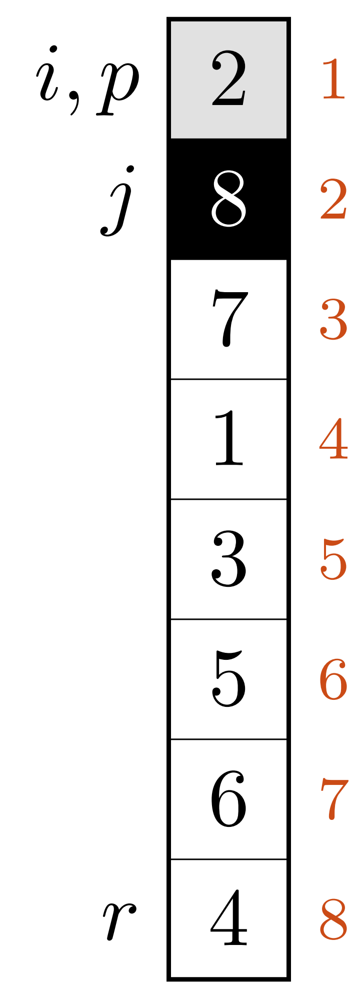
$x, i \ = \ 4, 1$

```
PARTITION(A, p, r)
  x = A[r]
  i = p − 1
  for j = p to r − 1
      if A[j] ≤ x
          i = i + 1
          exchange A[i] with A[j]
  exchange A[i + 1] with A[r]
  return i + 1
```

i, p	2	1
j	8	2
	7	3
	1	4
	3	5
	5	6
	6	7
r	4	8

$x, i \;=\; 4, 1$

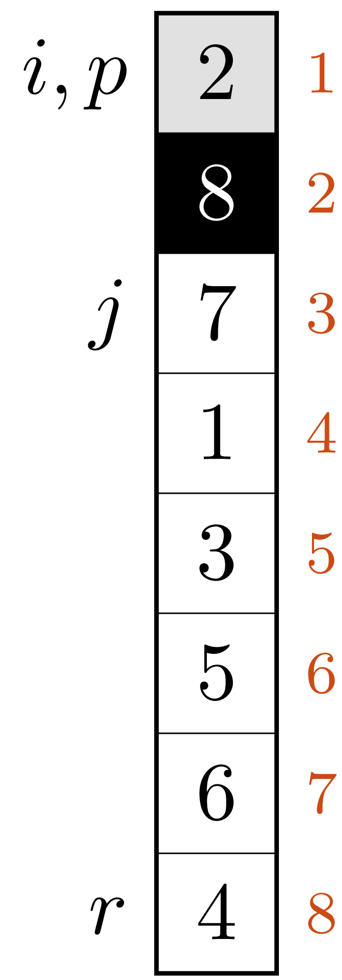
```
PARTITION(A, p, r)
  x = A[r]
  i = p − 1
  for j = p to r − 1
      if A[j] ≤ x
          i = i + 1
          exchange A[i] with A[j]
  exchange A[i + 1] with A[r]
  return i + 1
```



$x, i \;=\; 4, 1$

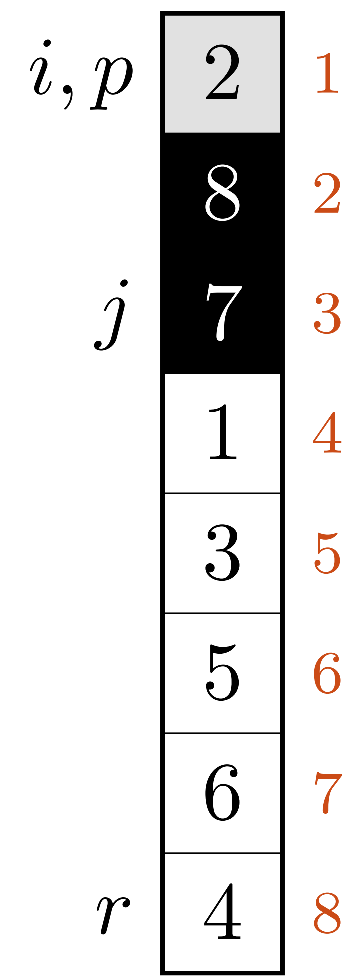
Partition(A, p, r)

```
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



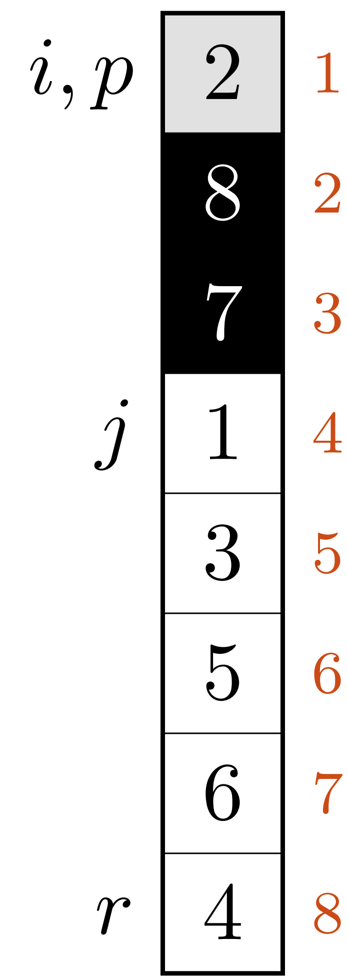
$x, i \;=\; 4, 1$

```
PARTITION(A, p, r)
  x = A[r]
  i = p − 1
  for j = p to r − 1
        if A[j] ≤ x
              i = i + 1
              exchange A[i] with A[j]
  exchange A[i + 1] with A[r]
  return i + 1
```



$x, i \;=\; 4, 1$

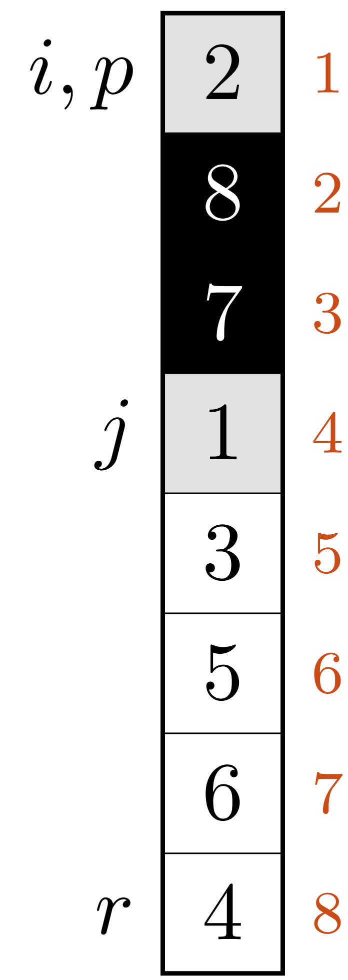
```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
      if A[j] ≤ x
            i = i + 1
            exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



$x, i \;=\; 4, 1$

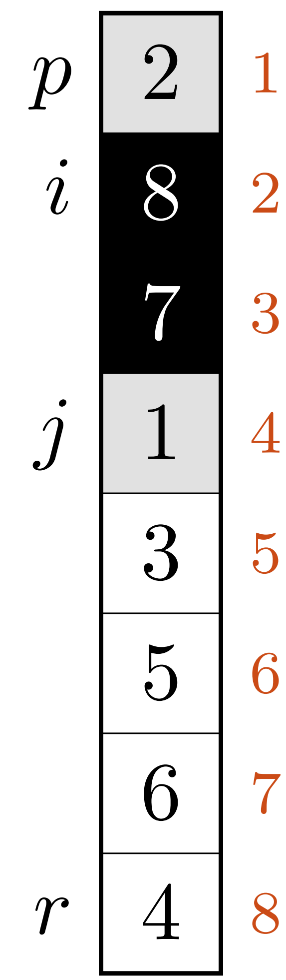
PARTITION(A, p, r)

```
x = A[r]
i = p − 1
for j = p to r − 1
        if A[j] ≤ x
                i = i + 1
                exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



$x, i \ = \ 4, 1$

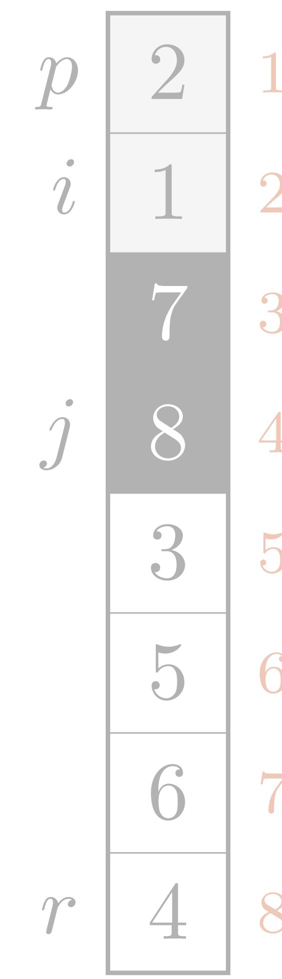
```
PARTITION(A, p, r)
  x = A[r]
  i = p − 1
  for j = p to r − 1
        if A[j] ≤ x
              i = i + 1
              exchange A[i] with A[j]
  exchange A[i + 1] with A[r]
  return i + 1
```



$x, i \;=\; 4, 2$

Partition(A, p, r)

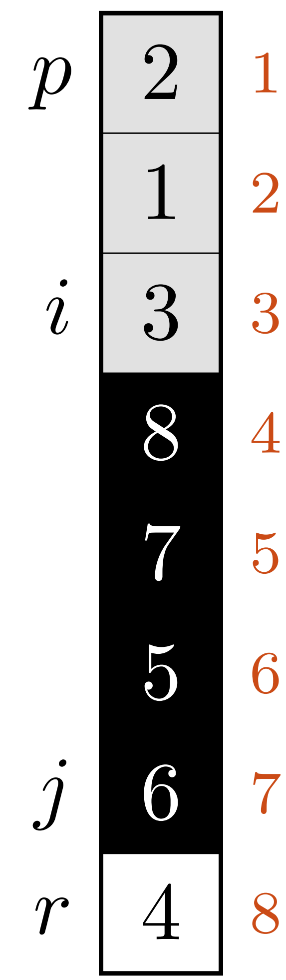
```
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



$x, i \;=\; 4, 2$

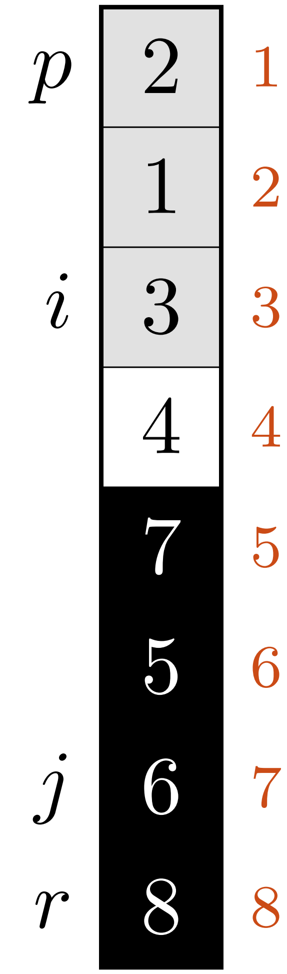
```
Partition(A, p, r)
  x = A[r]
  i = p − 1
  for j = p to r − 1
          if A[j] ≤ x
                  i = i + 1
                  exchange A[i] with A[j]
  exchange A[i + 1] with A[r]
  return i + 1
```

$x, i \;=\; 4, 3$



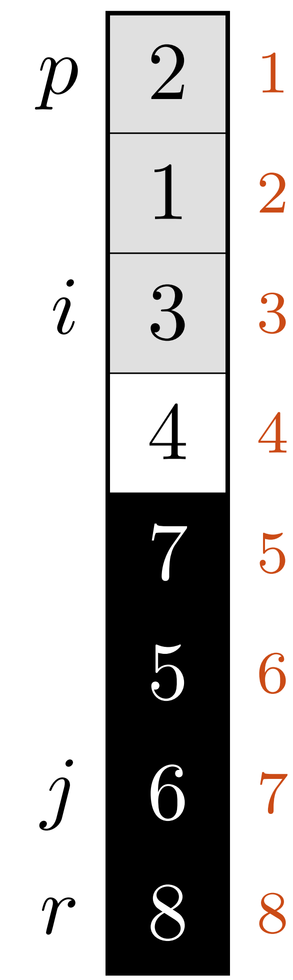
```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
        if A[j] ≤ x
                i = i + 1
                exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

$x, i \;=\; 4, 3$



```
PARTITION(A, p, r)
  x = A[r]
  i = p − 1
  for j = p to r − 1
      if A[j] ≤ x
          i = i + 1
          exchange A[i] with A[j]
  exchange A[i + 1] with A[r]
  return i + 1
```

$\rightarrow 4$



$x, i \ = \ 4, 3$

- **Skill «små» og «store» tall**
- **Sortér små tall rekursivt**
- **Sortér store tall rekursivt**

- **Skill «små» og «store» tall**
- **Sortér små tall rekursivt**
- **Sortér store tall rekursivt**

$\textsc{Quicksort}(A, p, r)$

Sortér $A[p \mathinner{.\,.} r]$ ved å sortere små og store verdier hver for seg

QUICKSORT(A, p, r)
1 **if** $p < r$

Hvis $A[p \mathinner{..} r]$ har lengde minst $2 \ldots$

Quicksort(A, p, r)

if $p < r$

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

$A[p \mathinner{.\,.} q-1] \leq A[q] \leq A[q+1 \mathinner{.\,.} r]$. Merk at $A[q]$ er på rett plass

```
QUICKSORT(A, p, r)
  if p < r
        q = PARTITION(A, p, r)
        QUICKSORT(A, p, q − 1)
```

Sortér små verdier $A[p \,.\,.\, q-1]$ rekursivt

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```

Sortér store verdier $A[q+1 \mathinner{..} r]$ rekursivt

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

Quicksort(A, p, r)

if $p < r$
    $q = \text{Partition}(A, p, r)$
    $\text{Quicksort}(A, p, q-1)$
    $\text{Quicksort}(A, q+1, r)$

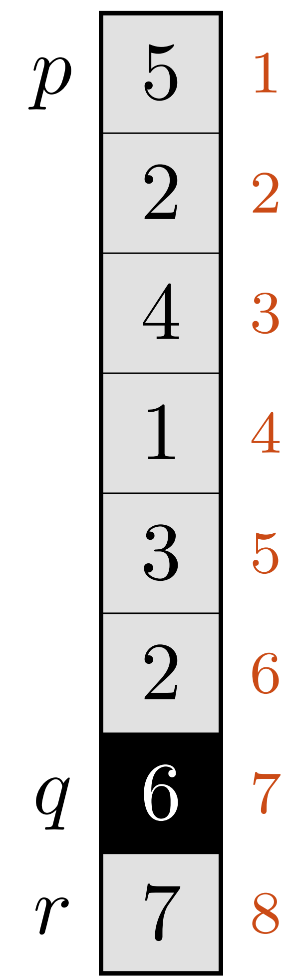
$p, q, r \; = \; 1, \text{-}, 8$

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

```
Quicksort(A, p, r)
if p < r
        q = Partition(A, p, r)
        Quicksort(A, p, q − 1)
        Quicksort(A, q + 1, r)
```

$p, q, r \;=\; 1, \text{-}, 8$

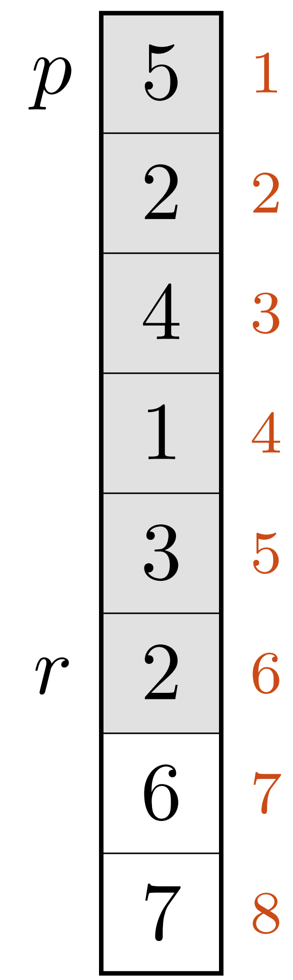
```
QUICKSORT(A, p, r)
if p < r
        q = PARTITION(A, p, r)
        QUICKSORT(A, p, q − 1)
        QUICKSORT(A, q + 1, r)
```



$p, q, r \; = \; 1, 7, 8$

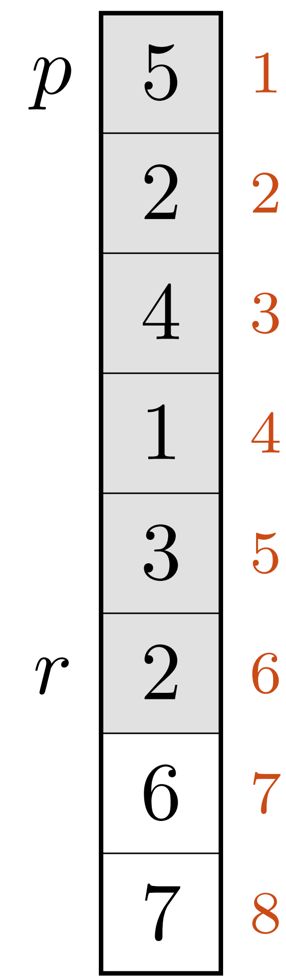
QUICKSORT(A, p, r)

```
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```



$p, q, r \;=\; 1, 7, 8 \;›\; 1, \text{-}, 6$

```
Quicksort(A, p, r)
  if p < r
      q = Partition(A, p, r)
      Quicksort(A, p, q − 1)
      Quicksort(A, q + 1, r)
```



$p, q, r \;=\; 1, 7, 8 \;›\; 1, \text{-}, 6$

```
Quicksort(A, p, r)
if p < r
    q = Partition(A, p, r)
    Quicksort(A, p, q − 1)
    Quicksort(A, q + 1, r)
```

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

$p, q, r \;=\; 1, 7, 8 \;›\; 1, 3, 6$

```
Quicksort(A, p, r)
if p < r
        q = Partition(A, p, r)
        Quicksort(A, p, q − 1)
        Quicksort(A, q + 1, r)
```

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

$p, q, r \;=\; 1, 7, 8 \,›\, 1, 3, 6 \,›\, 1, \text{-}, 2$

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

```
QUICKSORT(A, p, r)
if p < r
        q = PARTITION(A, p, r)
        QUICKSORT(A, p, q − 1)
        QUICKSORT(A, q + 1, r)
```

$p, q, r \;=\; 1, 7, 8 \;›\; 1, 3, 6 \;›\; 1, \text{–}, 2$

```
Quicksort(A, p, r)
  if p < r
      q = Partition(A, p, r)
      Quicksort(A, p, q − 1)
      Quicksort(A, q + 1, r)
```

$p, q, r \;=\; 1, 7, 8 \;›\; 1, 3, 6 \;›\; 1, 1, 2$

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

$\textsc{Quicksort}(A, p, r)$

if $p < r$
    $q = \textsc{Partition}(A, p, r)$
    $\textsc{Quicksort}(A, p, q - 1)$
    $\textsc{Quicksort}(A, q + 1, r)$

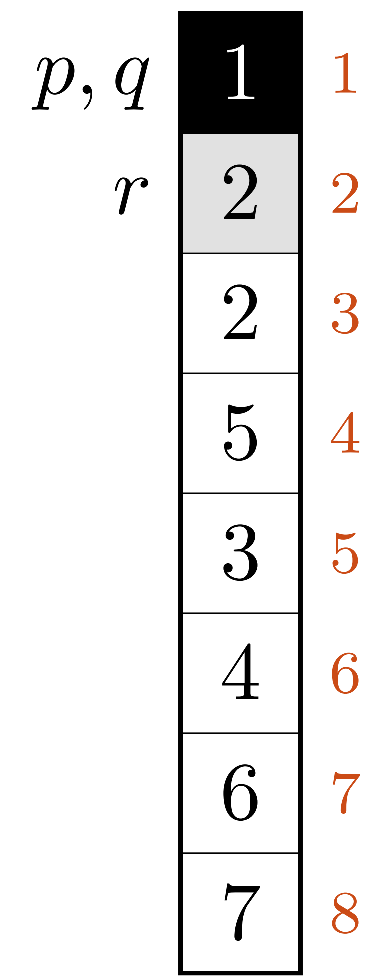
p	1	1
	2	2
	2	3
	5	4
	3	5
	4	6
	6	7
	7	8

$p, q, r \;=\; 1, 7, 8 \;›\; 1, 3, 6 \;›\; 1, 1, 2 \;›\; 1, \text{-}, 0$

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```

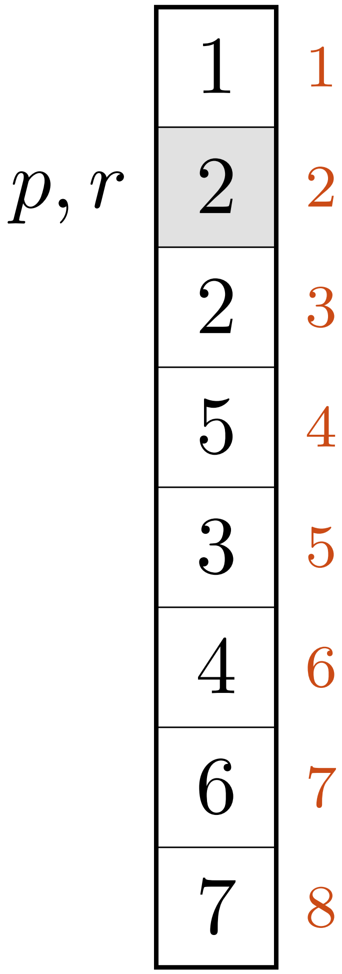
p	1	1
	2	2
	2	3
	5	4
	3	5
	4	6
	6	7
	7	8

$p, q, r \;=\; 1, 7, 8 \rangle 1, 3, 6 \rangle 1, 1, 2 \rangle 1, \text{-}, 0$




```
Quicksort(A, p, r)
  if p < r
        q = Partition(A, p, r)
        Quicksort(A, p, q − 1)
        Quicksort(A, q + 1, r)
```

$p, q, r \;=\; 1, 7, 8 \;›\; 1, 3, 6 \;›\; 1, 1, 2$



QUICKSORT(A, p, r)
if $p < r$
$\quad q = $ PARTITION(A, p, r)
$\quad$ QUICKSORT$(A, p, q-1)$
$\quad$ QUICKSORT$(A, q+1, r)$

$p, q, r \;=\; 1, 7, 8$ › $1, 3, 6$ › $1, 1, 2$ › $2, -, 2$

```
QUICKSORT(A, p, r)
if p < r
        q = PARTITION(A, p, r)
        QUICKSORT(A, p, q − 1)
        QUICKSORT(A, q + 1, r)
```

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

$p, q, r \;=\; 1, 7, 8 \;›\; 1, 3, 6 \;›\; 1, 1, 2 \;›\; 2, \text{-}, 2$

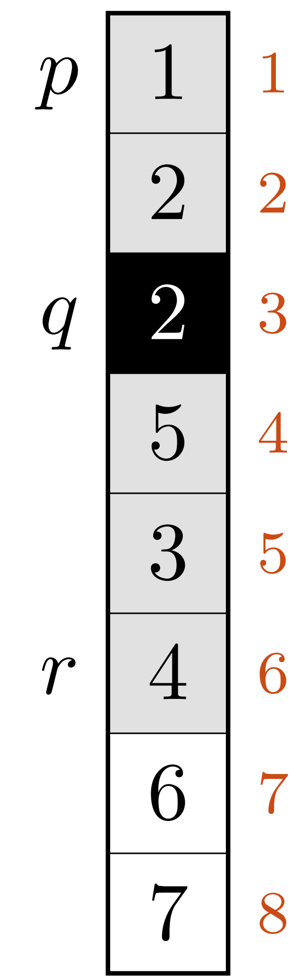
p, q	1	1
r	2	2
	2	3
	5	4
	3	5
	4	6
	6	7
	7	8

Quicksort(A, p, r)

```
if p < r
    q = Partition(A, p, r)
    Quicksort(A, p, q - 1)
    Quicksort(A, q + 1, r)
```

$p, q, r \;=\; 1, 7, 8 \;›\; 1, 3, 6 \;›\; 1, 1, 2$

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```



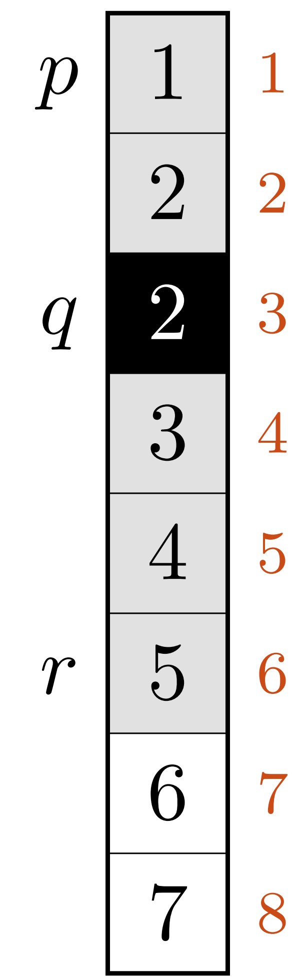
$p, q, r \;=\; 1, 7, 8 \;\rangle\; 1, 3, 6$

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```

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

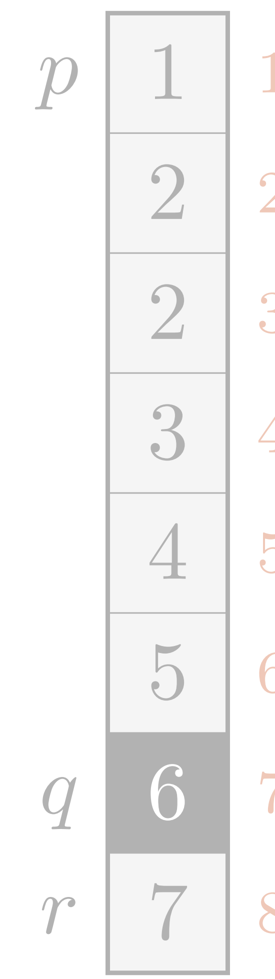
$p, q, r \;=\; 1, 7, 8 \;›\; 1, 3, 6 \;›\; 4, \text{–}, 6$

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```



$p, q, r \;=\; 1, 7, 8 \;\rangle\; 1, 3, 6$

```
QUICKSORT(A, p, r)
  if p < r
      q = PARTITION(A, p, r)
      QUICKSORT(A, p, q − 1)
      QUICKSORT(A, q + 1, r)
```



$p, q, r \;=\; 1, 7, 8$

```
QUICKSORT(A, p, r)
  if p < r
      q = PARTITION(A, p, r)
      QUICKSORT(A, p, q − 1)
      QUICKSORT(A, q + 1, r)
```

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

I beste/gjsn. tilfelle

Akkurat som merge sort

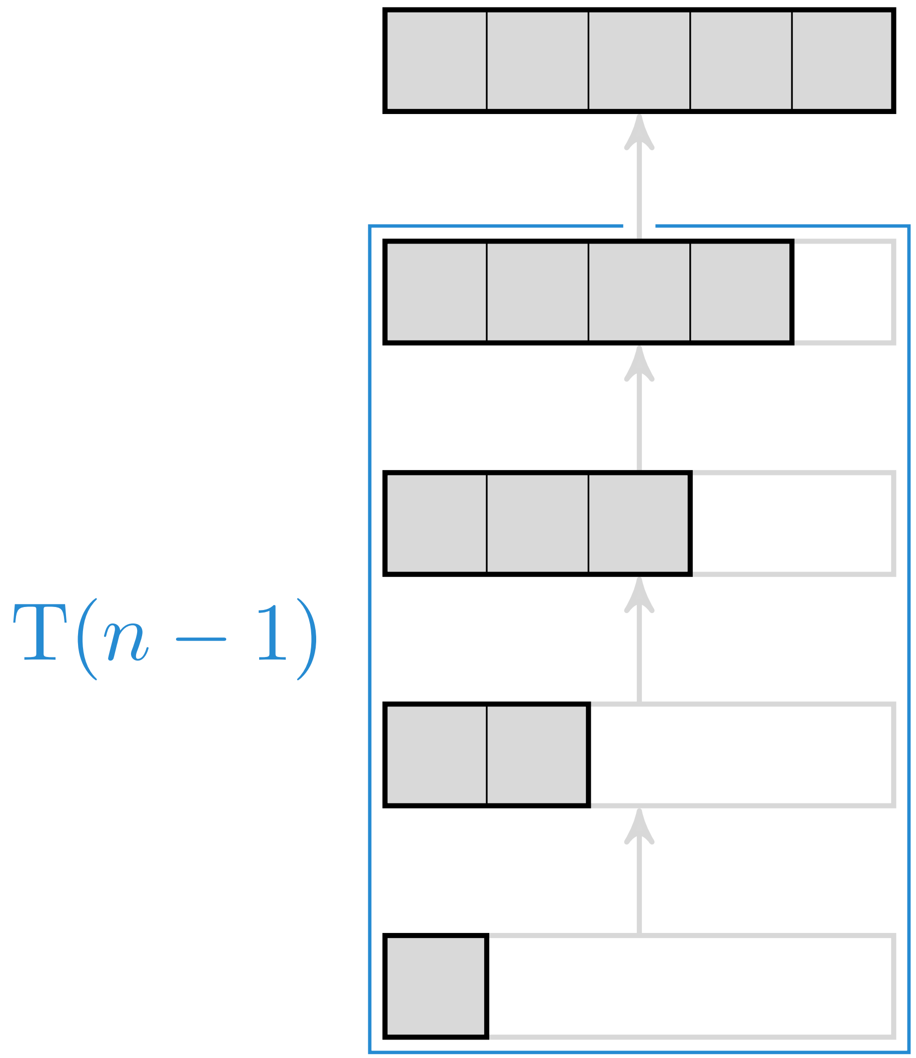
I verste tilfelle

Pivot blir helt alene

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

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

Ett rekursivt kall (størrelse $n - 1$) pluss n operasjoner


$T(n-1)$

$$\begin{aligned} \mathrm{T}(n) = n \\ + \mathrm{T}(n-1) \end{aligned} \tag{1}$$

$$
\begin{aligned}
T(n) = n& \\
+ T(n-1)& \qquad (1)
\end{aligned}
$$

$$
\begin{aligned}
T(n) = n & \\
+ \, n - 1 & \qquad (1) \\
+ \, T(n - 1 - 1) & \qquad (2)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= n \\
&\quad + n - 1 && (1) \\
&\quad + \mathrm{T}(n-2) && (2)
\end{aligned}
$$

$$
\begin{aligned}
T(n) = n & \\
+\, n - 1 & \qquad (1) \\
+\, n - 2 & \qquad (2) \\
+\, T(n - 3) & \qquad (3)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) = n & \\
+\, n - 1 & \qquad (1) \\
+\, n - 2 & \qquad (2) \\
+\, n - 3 & \qquad (3) \\
\vdots\quad & \qquad \vdots \\
+\, \mathrm{T}(n - ?) & \qquad (?)
\end{aligned}
$$

$$
\begin{aligned}
T(n) = n & \\
+\, n - 1 & \qquad (1) \\
+\, n - 2 & \qquad (2) \\
+\, n - 3 & \qquad (3) \\
\vdots\ & \qquad \vdots \\
+\, T(n - ?) & \qquad (n-1)
\end{aligned}
$$

$$
\begin{aligned}
T(n) = n & \\
+\, n - 1 & \qquad (1) \\
+\, n - 2 & \qquad (2) \\
+\, n - 3 & \qquad (3) \\
\vdots\ \ & \qquad \vdots \\
+\, T(n - (n - 1)) & \qquad (n - 1)
\end{aligned}
$$

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

$$
\begin{aligned}
T(n) = n & \\
+\, n - 1 & \qquad (1) \\
+\, n - 2 & \qquad (2) \\
+\, n - 3 & \qquad (3) \\
\vdots\ & \qquad \vdots \\
+\, 1 & \qquad (n-1)
\end{aligned}
$$

$$
\begin{aligned}
T(n) = n & \\
+\, n - 1 & \qquad (1) \\
+\, n - 2 & \qquad (2) \\
+\, n - 3 & \qquad (3) \\
\vdots\quad & \qquad \vdots \\
+\, 1 & \qquad (n-1)
\end{aligned}
$$

$$T(n) = n(n+1)/2$$

Verifikasjon

Med substitusjon/induksjon

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

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

Gitt $\mathrm{T}(n-1) = (n-1)n/2$, vis $\mathrm{T}(n) = n(n+1)/2$

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

Gitt $T(n-1) = (n-1)n/2$, vis $T(n) = n(n+1)/2$

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

$$= \frac{(n-1)n}{2} + n$$

Gitt $\mathrm{T}(n-1) = (n-1)n/2$, vis $\mathrm{T}(n) = n(n+1)/2$

$$\begin{aligned}
\mathrm{T}(n) &= \mathrm{T}(n-1) + n \\
&= \frac{(n-1)n}{2} + n \\
&= \frac{n^2 - n + 2n}{2}
\end{aligned}$$

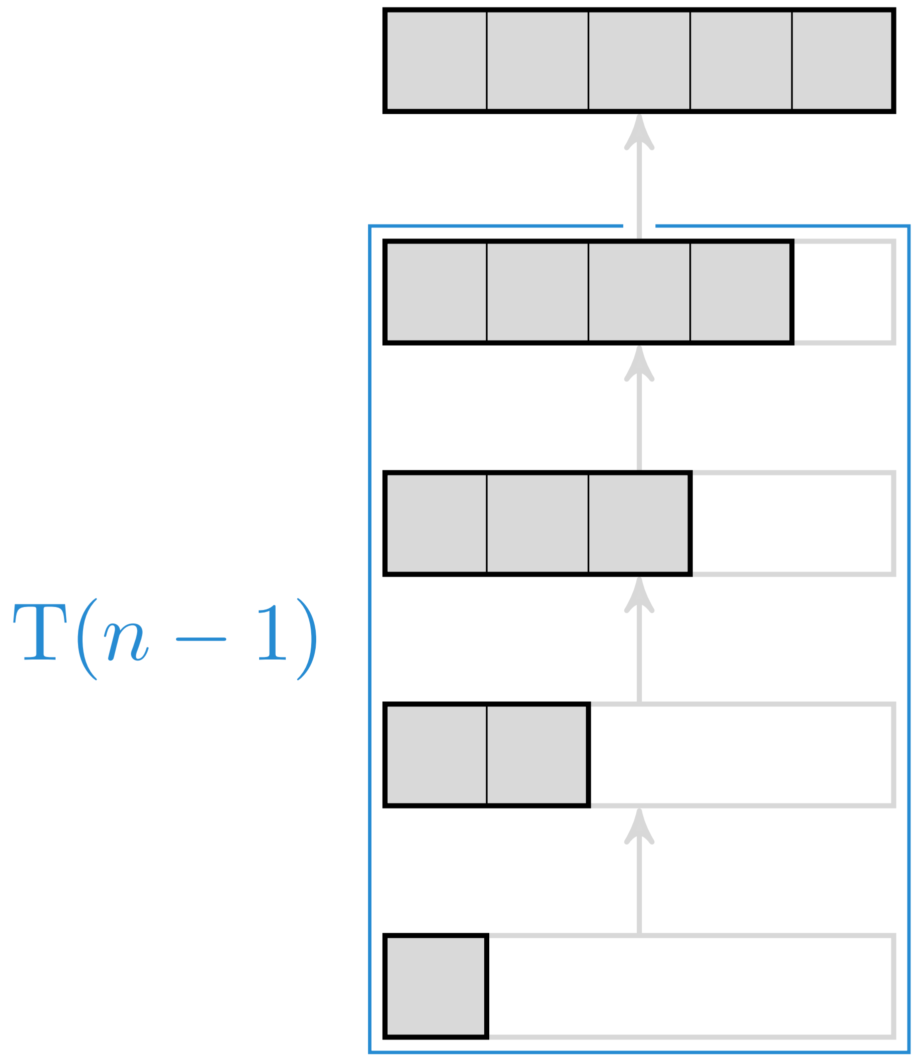
Gitt $\mathrm{T}(n-1) = (n-1)n/2$, vis $\mathrm{T}(n) = n(n+1)/2$

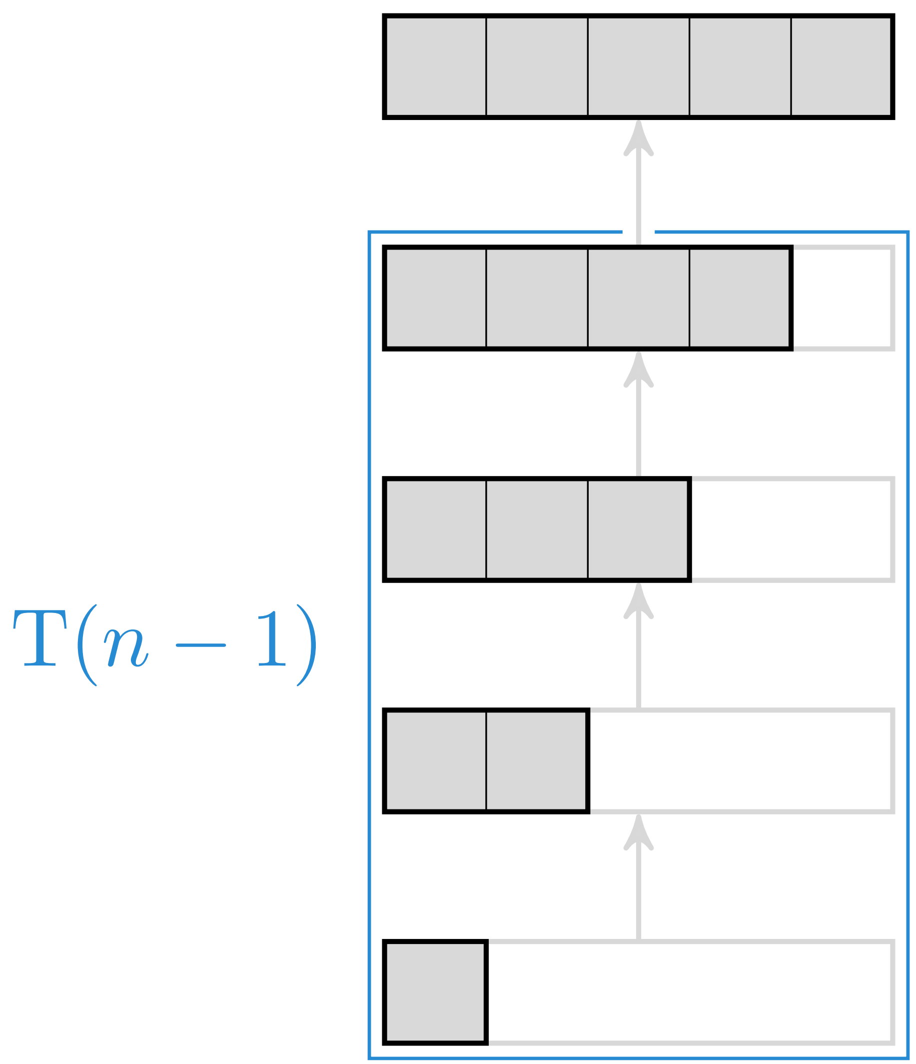
$$
\begin{aligned}
T(n) &= T(n-1) + n \\
&= \frac{(n-1)n}{2} + n \\
&= \frac{n^2 - n + 2n}{2} \\
&= \frac{n(n+1)}{2}
\end{aligned}
$$

Gitt $T(n-1) = (n-1)n/2$, vis $T(n) = n(n+1)/2$

$$
\begin{aligned}
\mathrm{T}(n) &= \mathrm{T}(n-1) + n \\
&= \frac{(n-1)n}{2} + n \\
&= \frac{n^2 - n + 2n}{2} \\
&= \frac{n(n+1)}{2}
\end{aligned}
$$

Gitt $\mathrm{T}(n-1) = (n-1)n/2$, vis $\mathrm{T}(n) = n(n+1)/2$


$T(n-1)$



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

Randomisering

Så ingen input alltid er kjip

Dette tabbet de seg ut på i implementasjonen av Swift sin sorteringsalgoritme, opprinnelig: https://lemire.me/blog/2017/02/06/sorting-sorted-arrays-in-swift/

(Arkivert: https://archive.is/ItULW)

```
RANDOMIZED-PARTITION(A, p, r)
  i = RANDOM(p, r)
  exchange A[r] and A[i]
  return PARTITION(A, p, r)
```

Tilfeldig splittelement, heller enn vilkårlig

mlh@ntnu.no

Randomized-Quicksort(A, p, r)

if $p < r$
  $q =$ Randomized-Partition(A, p, r)
  Randomized-Quicksort$(A, p, q - 1)$
  Randomized-Quicksort$(A, q + 1, r)$

Unngå at f.eks. sortert input alltid gir verste kjøretid

For spesielt interesserte: Akra–Bazzi-metoden er en generalisering av dette.

https://en.wikipedia.org/wiki/Akra–Bazzi_method

5:6

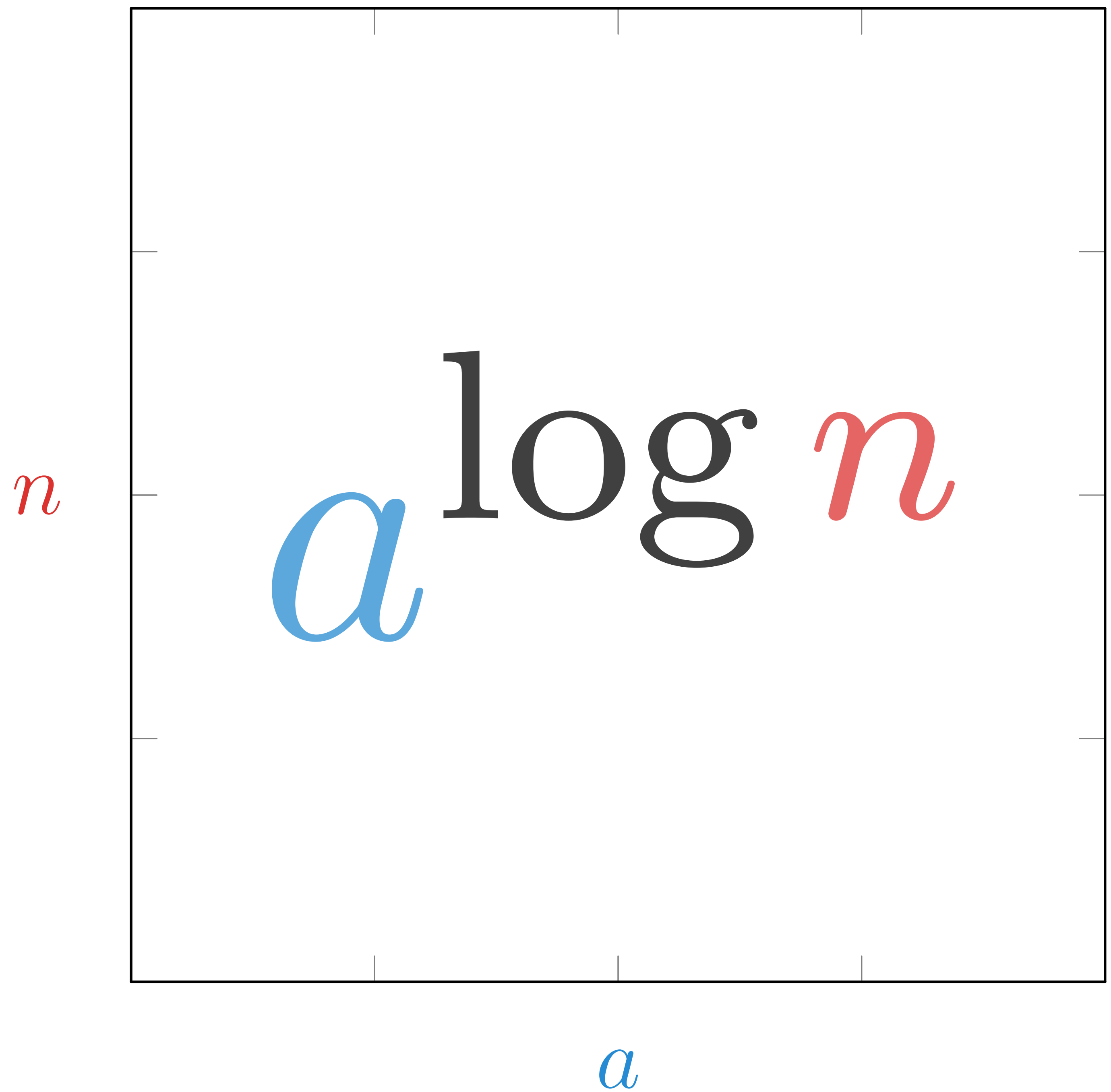
Masterteoremet

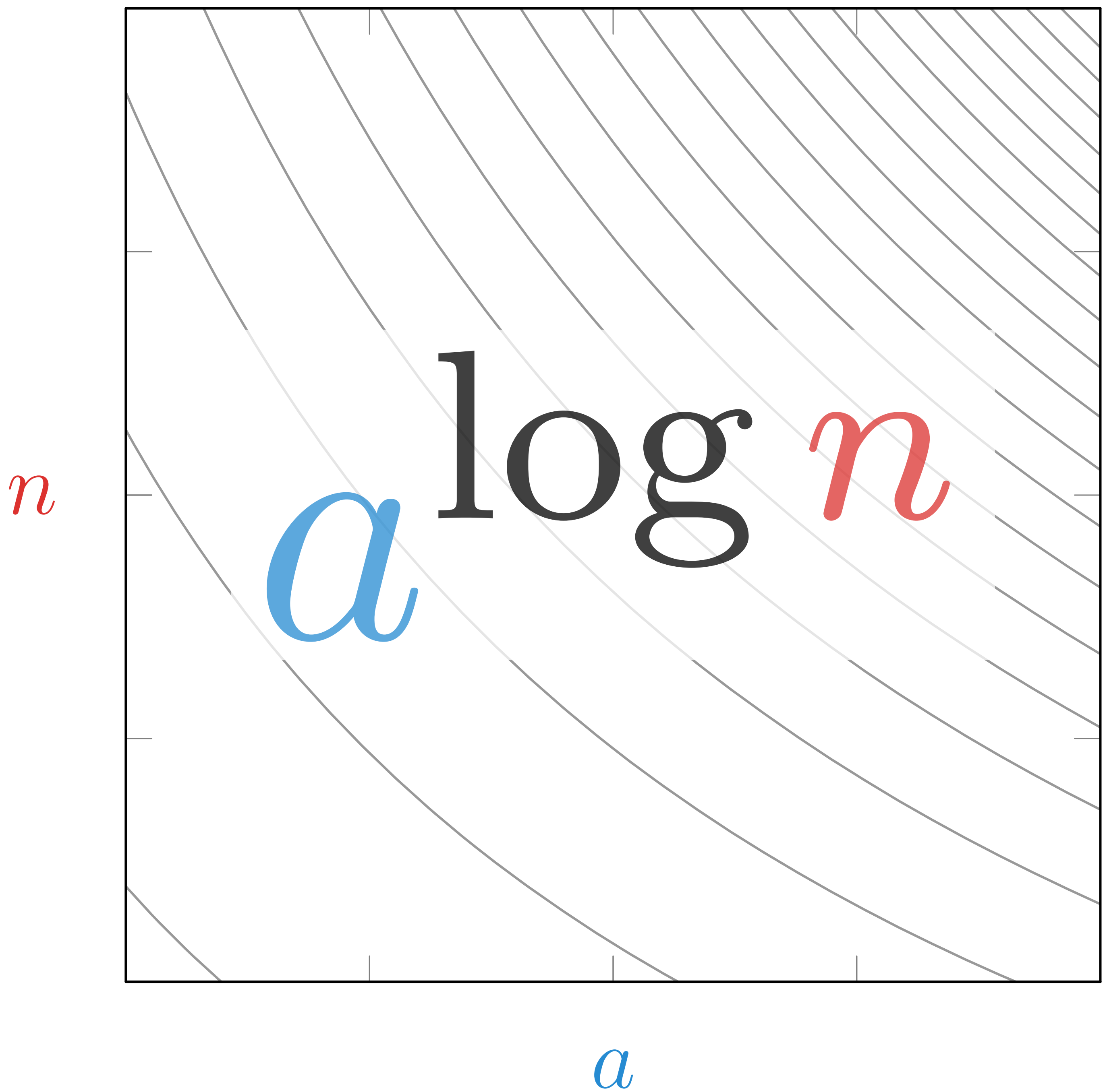
SIGACT News 36 Fall 1980

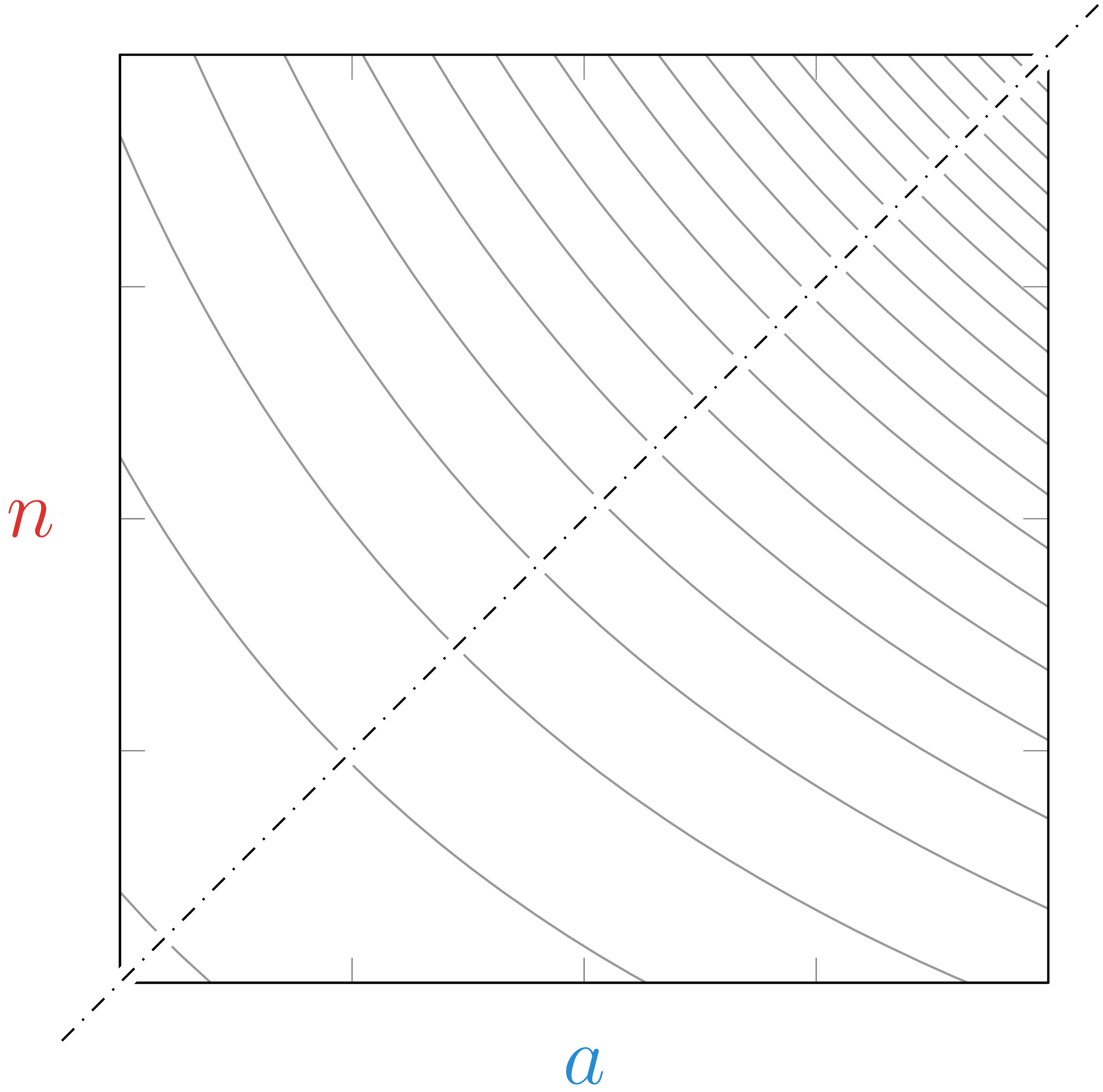
A General Method for Solving Divide-and-Conquer Recurrences[1]

Jon Louis Bentley[2]
Dorothea Haken
James B. Saxe
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

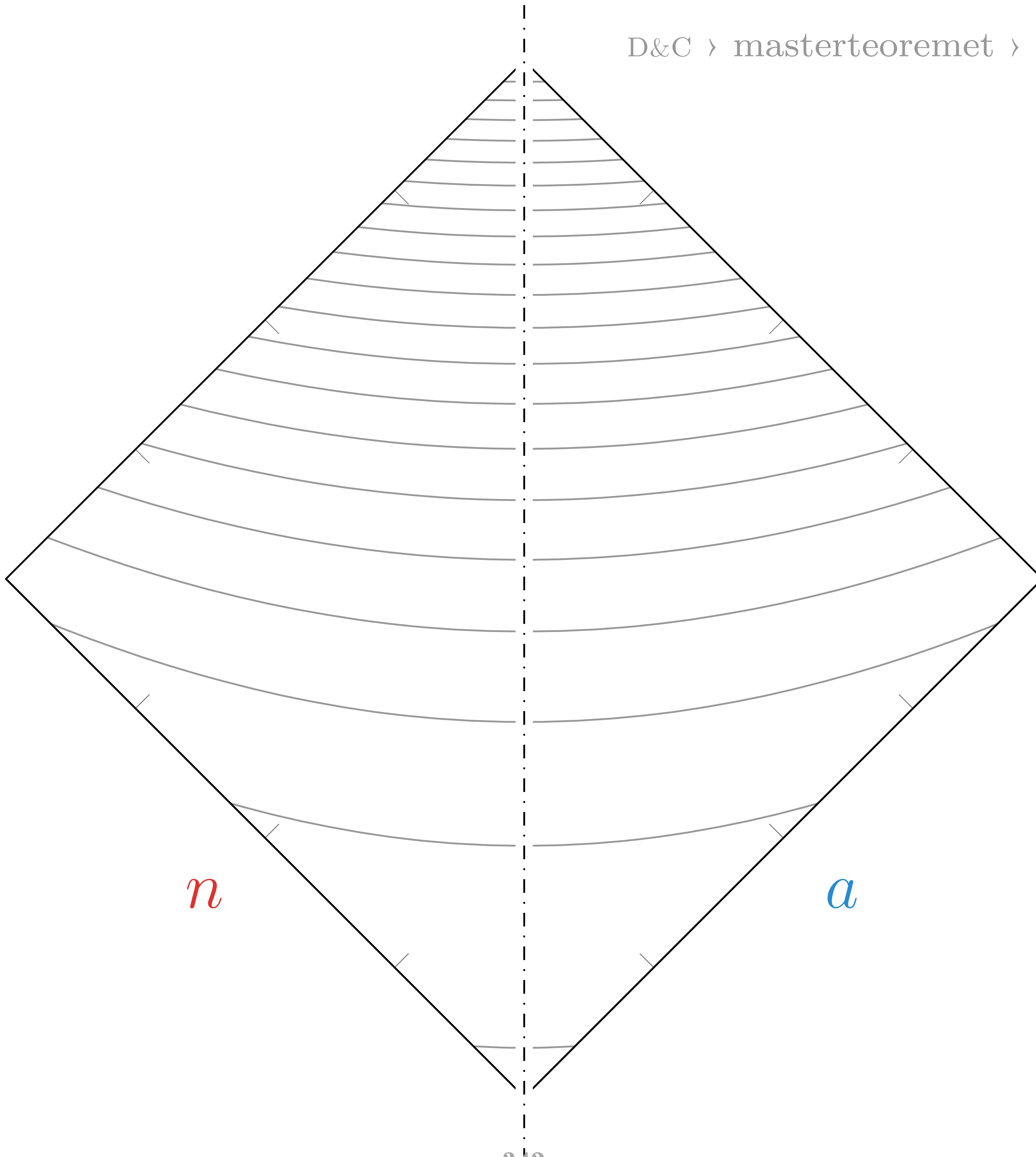
$$a^{\log n}$$


$a^{\log n}$
n
a

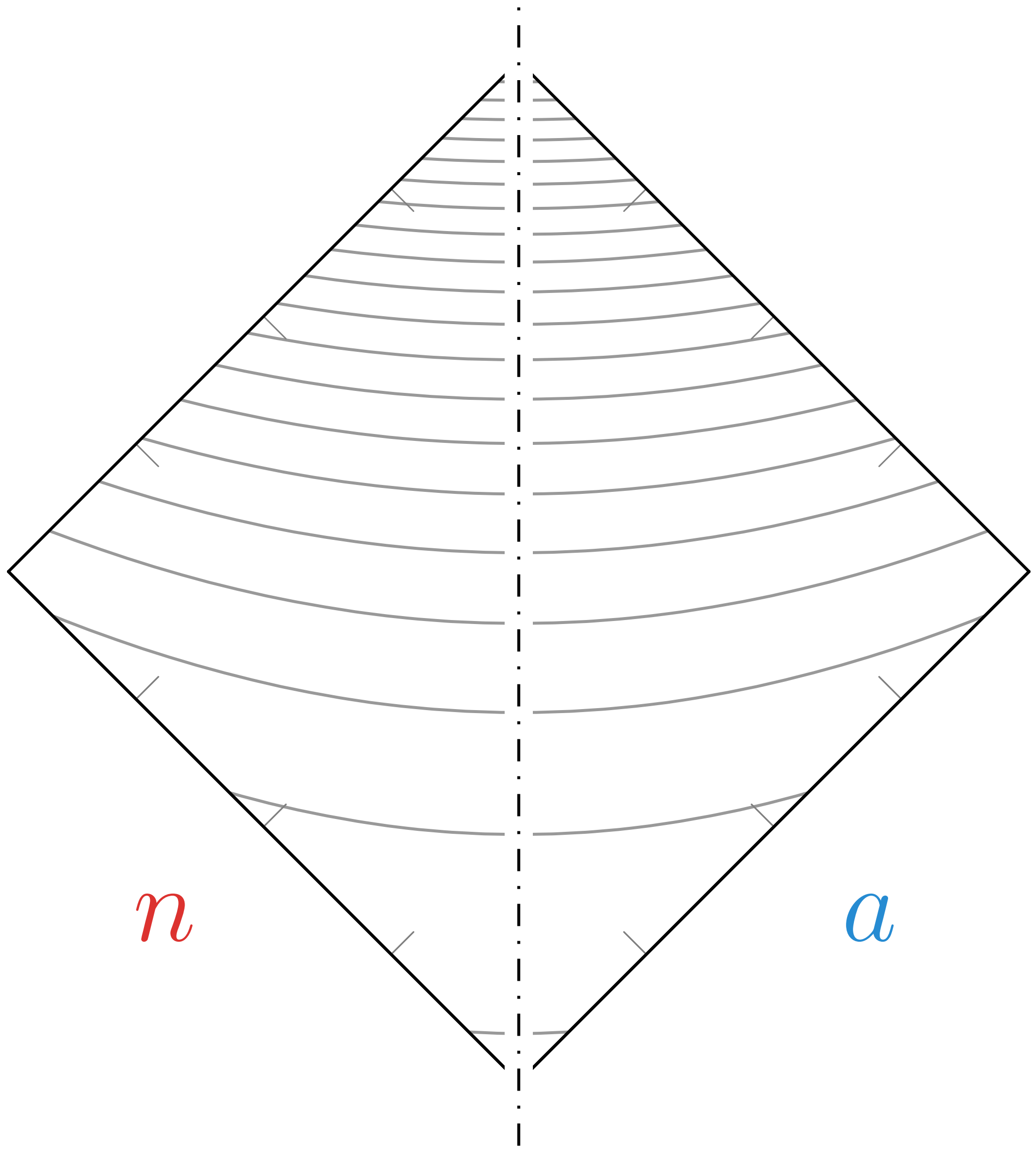

$a^{\log n}$
n
a

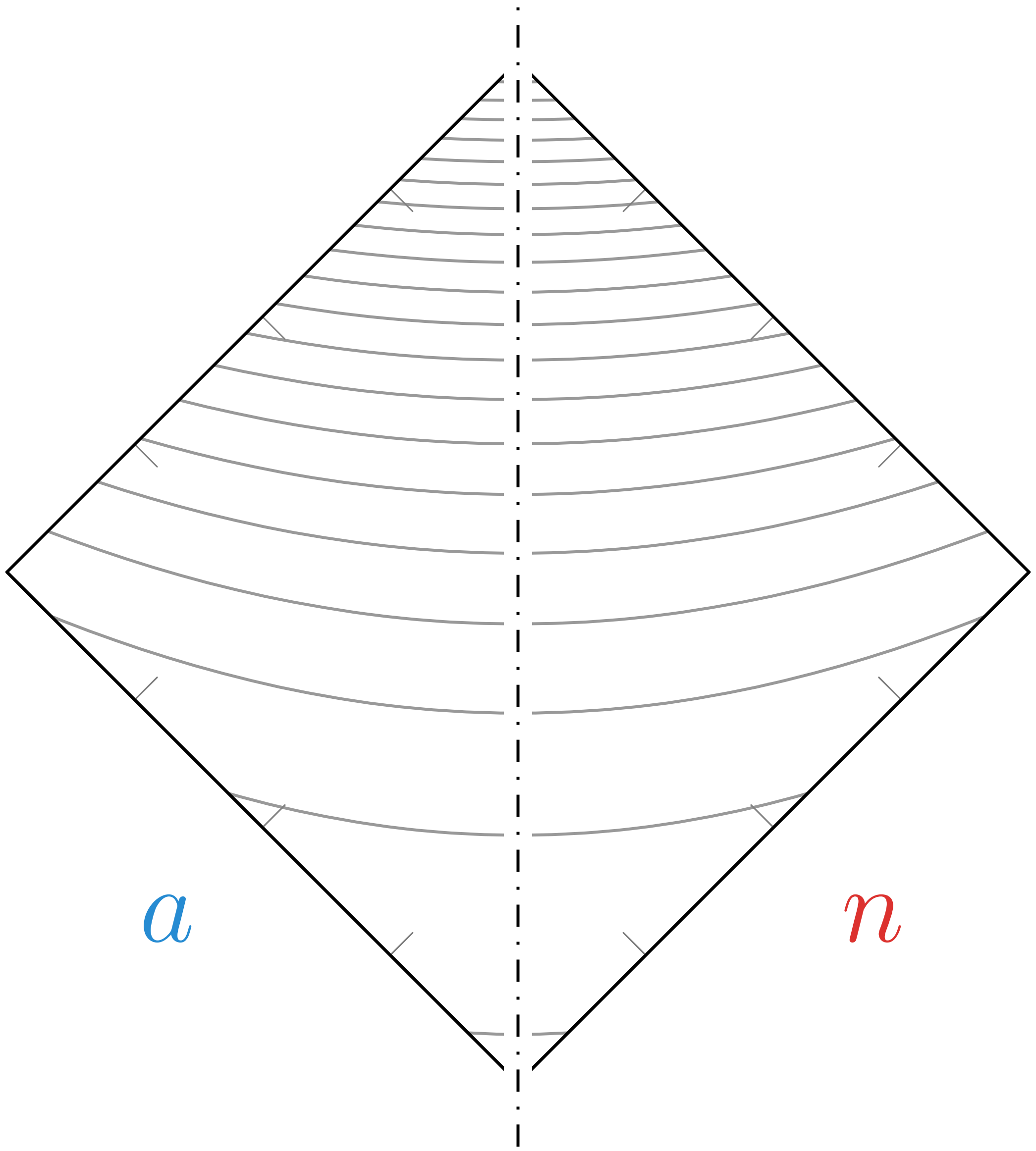


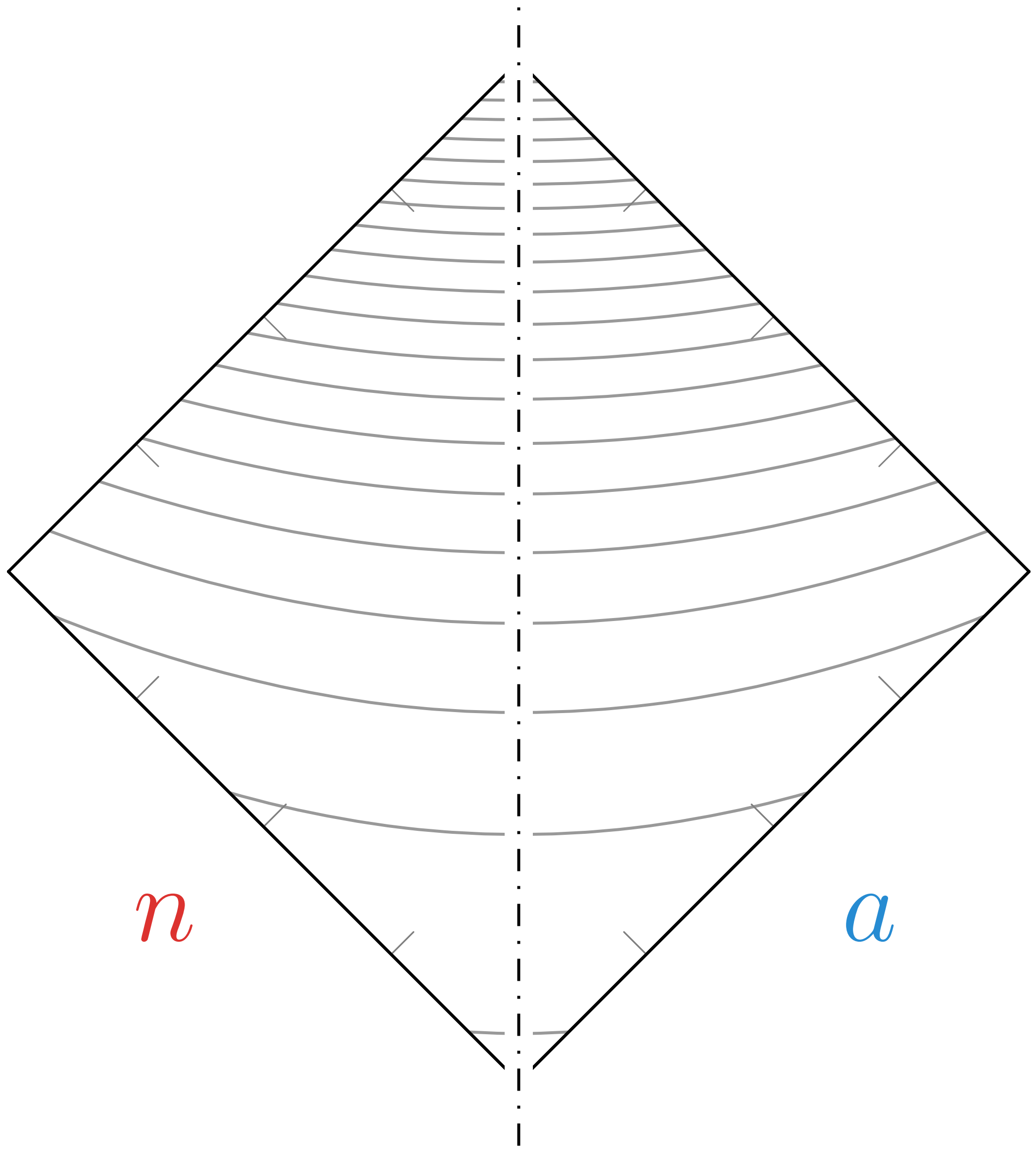
n
a



n
a

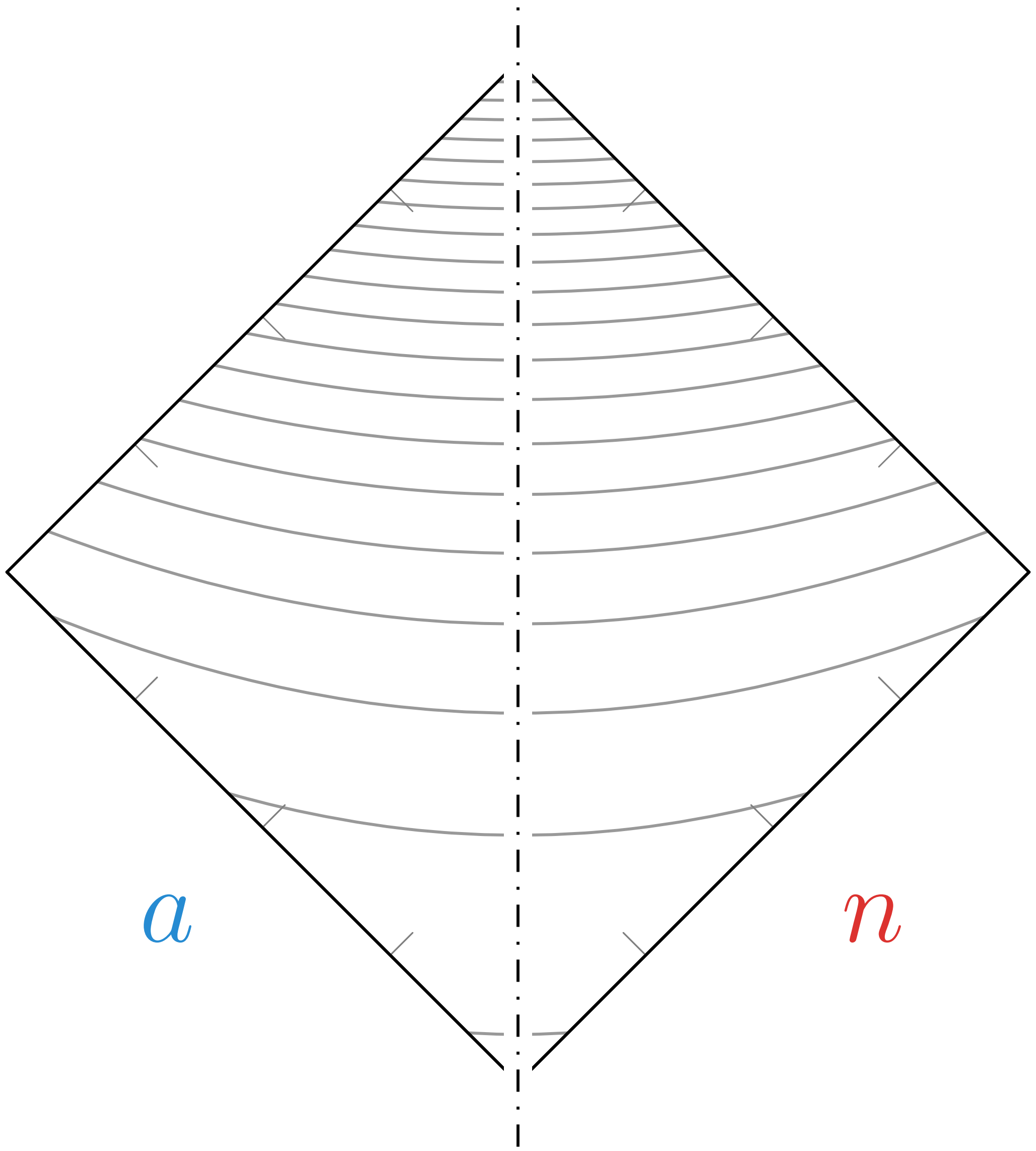

n
a


a
n

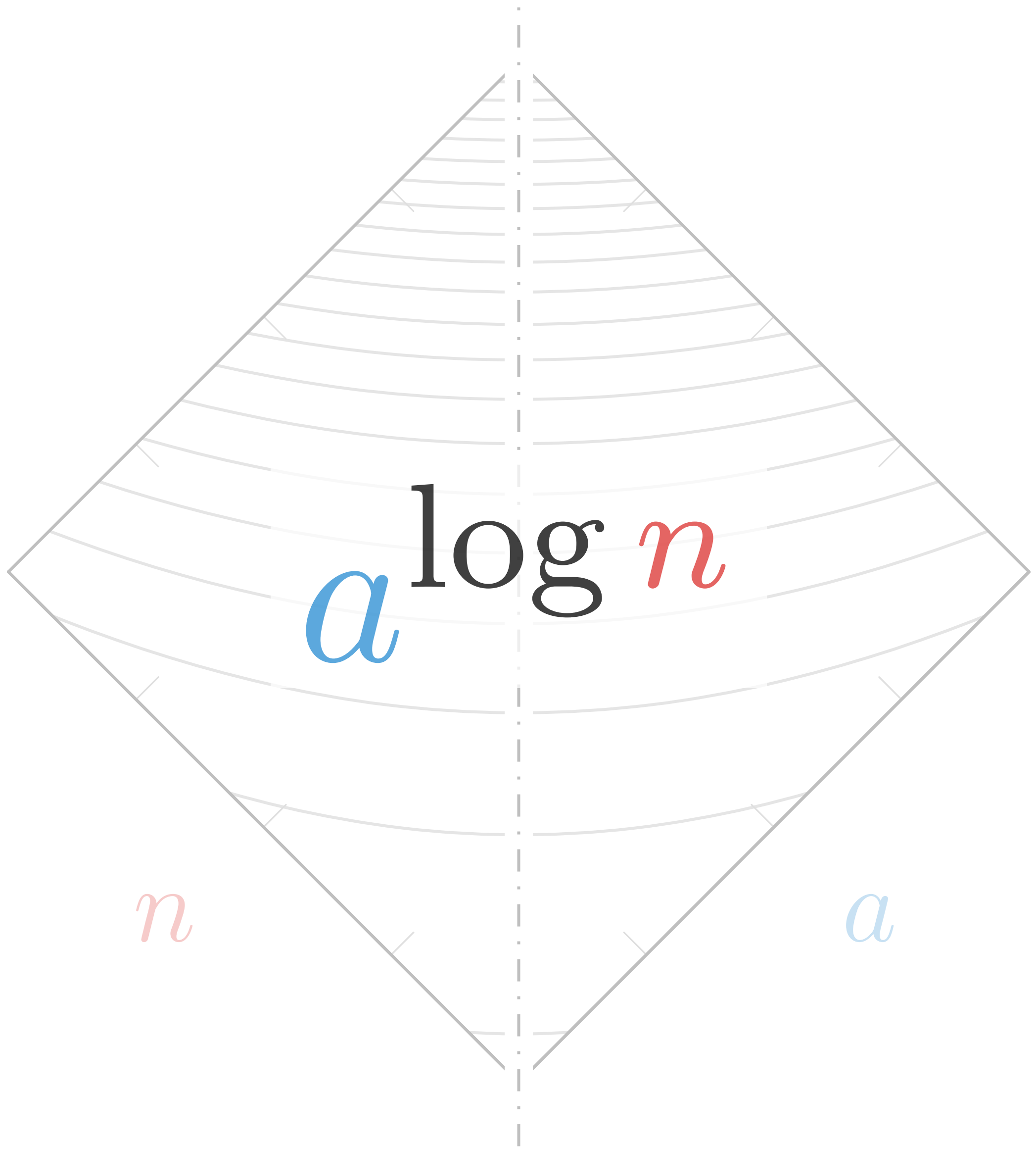


n
a

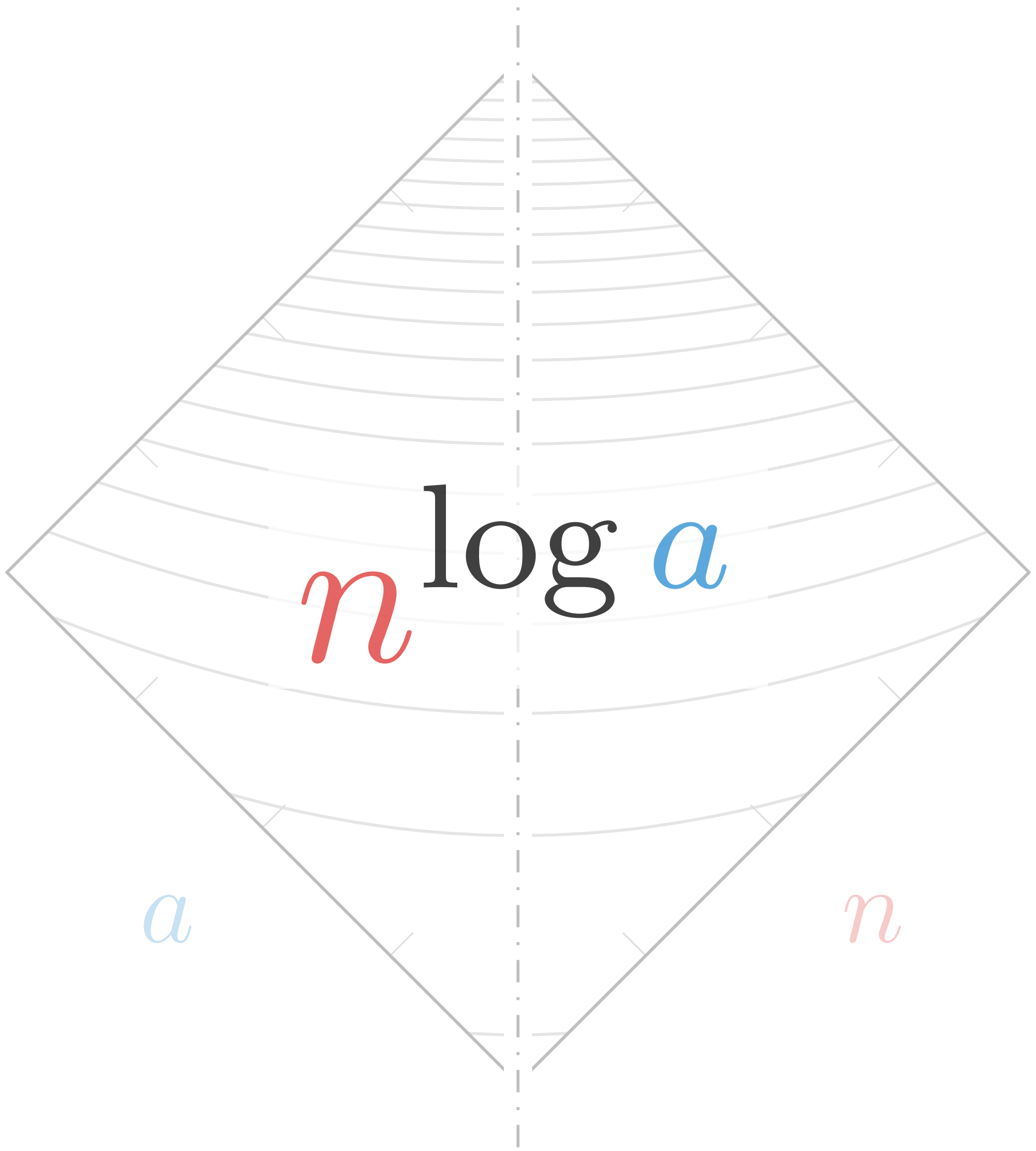
$=$

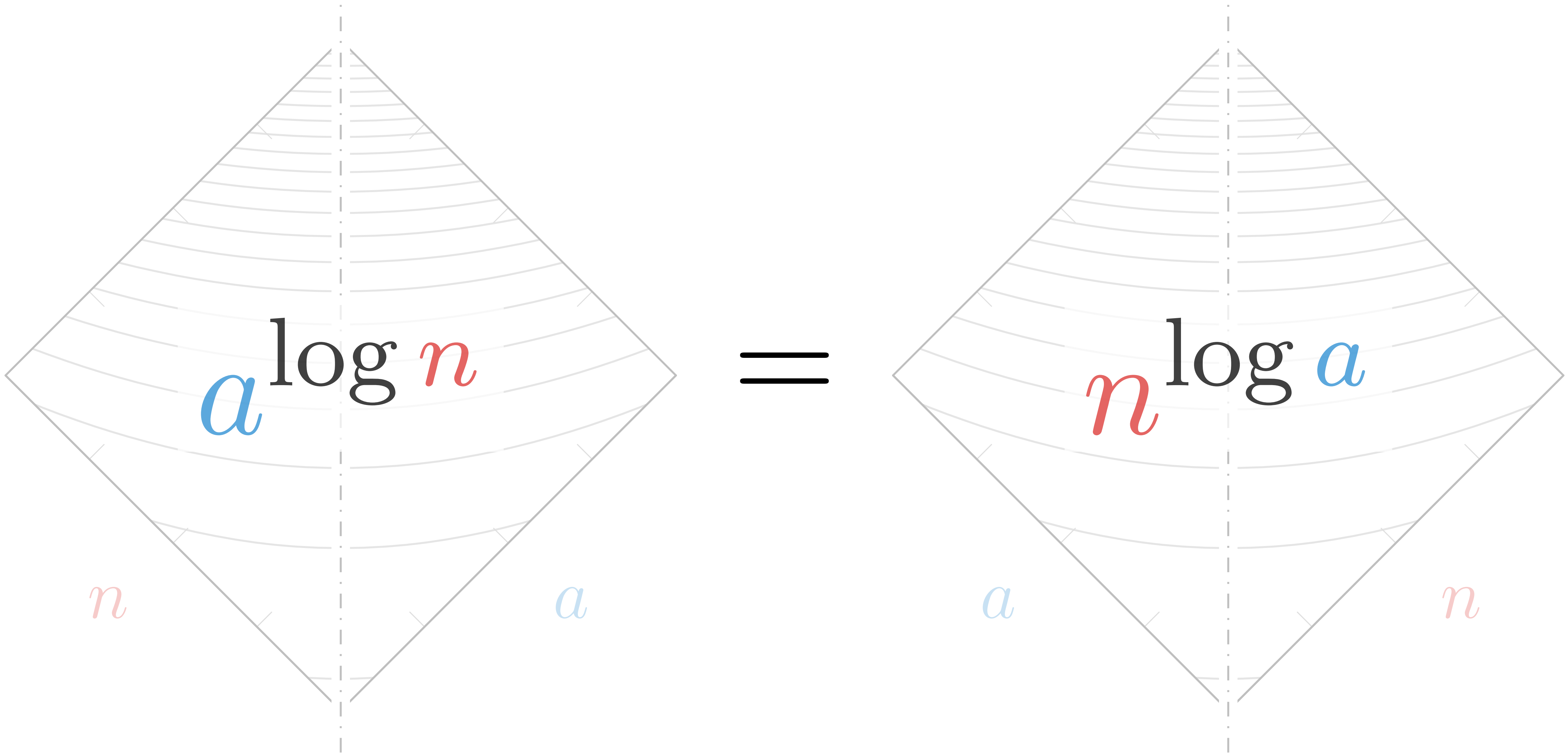


a
n


$a^{\log n}$
n
a

$=$


$n^{\log a}$
a
n



Ta logaritmen på begge sider: $\log a \cdot \log n = \log n \cdot \log a$

$$\mathrm{T}(n) = a\mathrm{T}(n/b) + f(n)$$

$$\mathrm{T}(n) = a\mathrm{T}(n/b) + f(n)$$

a rekursive kall (størrelse n/b) pluss $f(n)$ operasjoner

$$\begin{aligned}\mathrm{T}(n) = {} & f(n) \\ & + a\mathrm{T}(n/b) \end{aligned} \tag{1}$$

$$\begin{aligned} T(n) &= f(n) \\ &\quad + aT(n/b) \end{aligned} \tag{1}$$

$$
\begin{aligned}
T(n) &= f(n) \\
&+ af(n/b) && (1) \\
&+ aaT(n/b/b) && (2)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= f(n) \\
&\quad + af(n/b) && (1) \\
&\quad + a^2\,T(n/b^2) && (2)
\end{aligned}
$$

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

$$+ af(n/b) \tag{1}$$

$$+ a^2 f(n/b^2) \tag{2}$$

$$+ a^3\,\mathrm{T}(n/b^3) \tag{3}$$

$$
\begin{aligned}
\mathrm{T}(n) = {} & f(n) \\
& + af(n/b) && (1) \\
& + a^2 f(n/b^2) && (2) \\
& + a^3 f(n/b^3) && (3) \\
& \;\vdots && \vdots \\
& + a^{?}\,\mathrm{T}(n/b^{?}) && (?)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) = f(n) & \\
+\, a f(n/b) & \qquad (1) \\
+\, a^2 f(n/b^2) & \qquad (2) \\
+\, a^3 f(n/b^3) & \qquad (3) \\
\vdots\quad & \qquad \vdots \\
+\, a^{?}\,\mathrm{T}(n/b^{?}) & \qquad (\log_b n)
\end{aligned}
$$

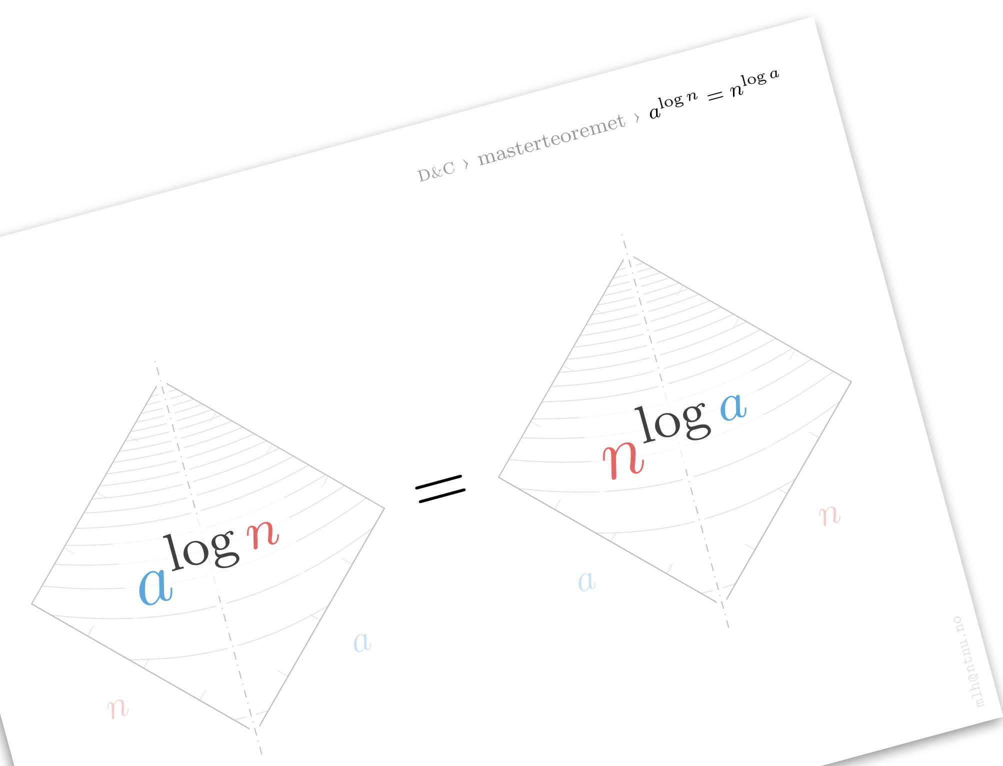
$$
\begin{aligned}
\mathrm{T}(n) &= f(n) \\
&\quad + af(n/b) && (1) \\
&\quad + a^2 f(n/b^2) && (2) \\
&\quad + a^3 f(n/b^3) && (3) \\
&\quad\ \vdots && \vdots \\
&\quad + a^{\log_b n}\,\mathrm{T}(n/b^{\log_b n}) && (\log_b n)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= f(n) \\
&\quad + af(n/b) && (1) \\
&\quad + a^2 f(n/b^2) && (2) \\
&\quad + a^3 f(n/b^3) && (3) \\
&\quad \vdots && \vdots \\
&\quad + a^{\log_b n}\,\mathrm{T}(n/n) && (\log_b n)
\end{aligned}
$$

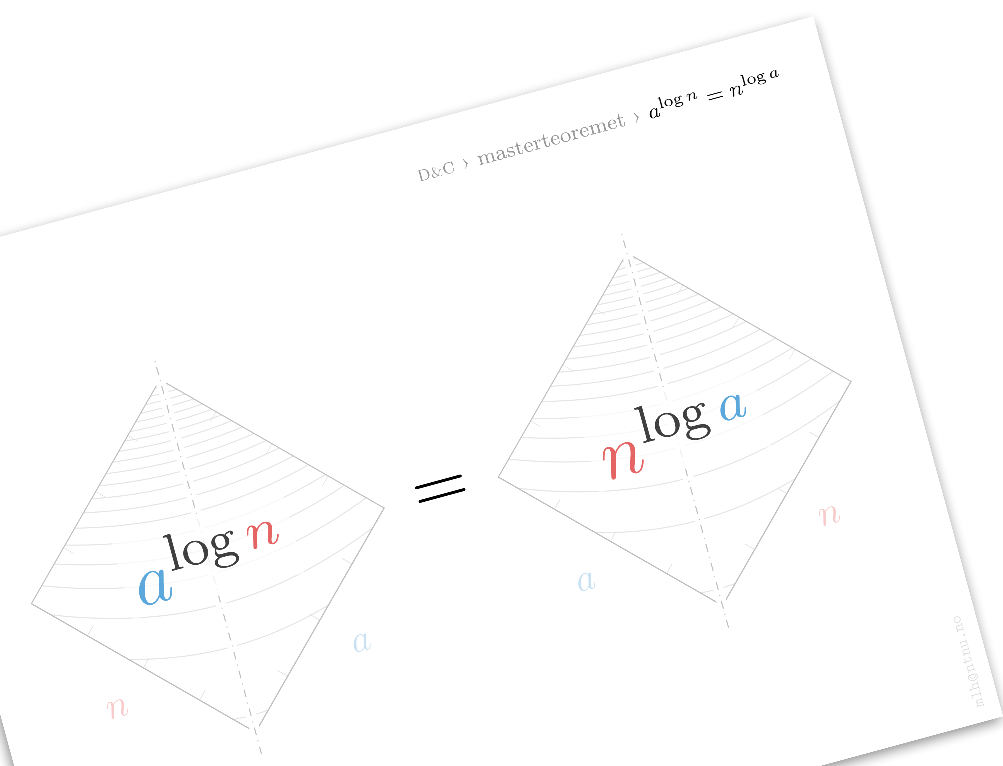
$$
\begin{aligned}
\mathrm{T}(n) = f(n) & \\
+\, af(n/b) & \qquad (1) \\
+\, a^2 f(n/b^2) & \qquad (2) \\
+\, a^3 f(n/b^3) & \qquad (3) \\
\vdots \qquad & \qquad \vdots \\
+\, a^{\log_b n}\,\mathrm{T}(1) & \qquad (\log_b n)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) &= f(n) \\
&\quad + af(n/b) && (1) \\
&\quad + a^2 f(n/b^2) && (2) \\
&\quad + a^3 f(n/b^3) && (3) \\
&\quad \vdots && \vdots \\
&\quad + a^{\log_b n} && (\log_b n)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{T}(n) = f(n) & \\
+\ a f(n/b) & \qquad (1) \\
+\ a^2 f(n/b^2) & \qquad (2) \\
+\ a^3 f(n/b^3) & \qquad (3) \\
\vdots\quad & \qquad \vdots \\
+\ a^{\log_b n} & \qquad (\log_b n)
\end{aligned}
$$



$$
\begin{aligned}
\mathrm{T}(n) &= f(n) \\
&\quad + af(n/b) && (1) \\
&\quad + a^2 f(n/b^2) && (2) \\
&\quad + a^3 f(n/b^3) && (3) \\
&\quad\ \vdots && \vdots \\
&\quad + n^{\log_b a} && (\log_b n)
\end{aligned}
$$



$$
\begin{aligned}
\mathrm{T}(n) = f(n) & \\
+\, af(n/b) & \qquad (1) \\
+\, a^2 f(n/b^2) & \qquad (2) \\
+\, a^3 f(n/b^3) & \qquad (3) \\
\vdots\quad & \qquad \vdots \\
+\, n^{\log_b a} & \qquad (\log_b n)
\end{aligned}
$$

$$f(n) = \mathrm{O}(n^{\log_b a - \epsilon}) \implies \mathrm{T}(n) =$$

$$
\begin{aligned}
\mathrm{T}(n) &= f(n) \\
&\quad + af(n/b) && (1) \\
&\quad + a^2 f(n/b^2) && (2) \\
&\quad + a^3 f(n/b^3) && (3) \\
&\quad \vdots && \vdots \\
&\quad + n^{\log_b a} && (\log_b n)
\end{aligned}
$$

$$f(n) = \mathrm{O}(n^{\log_b a - \epsilon}) \implies \mathrm{T}(n) =$$

$$
\begin{aligned}
\mathrm{T}(n) = {} & f(n) \\
& + af(n/b) && (1) \\
& + a^2 f(n/b^2) && (2) \\
& + a^3 f(n/b^3) && (3) \\
& \vdots && \vdots \\
& + n^{\log_b a} && (\log_b n)
\end{aligned}
$$

$$f(n) = \mathrm{O}(n^{\log_b a - \epsilon}) \implies \mathrm{T}(n) = \Theta(n^{\log_b a})$$

$$\begin{aligned}
\mathrm{T}(n) = f(n) & \\
+\, a f(n/b) & \qquad (1) \\
+\, a^2 f(n/b^2) & \qquad (2) \\
+\, a^3 f(n/b^3) & \qquad (3) \\
\vdots \qquad & \qquad \vdots \\
+\, n^{\log_b a} & \qquad (\log_b n)
\end{aligned}$$

$$f(n) = \Theta(n^{\log_b a}) \implies \mathrm{T}(n) =$$

$$
\begin{aligned}
\mathrm{T}(n) = {} & \Theta(n^{\log_b a}) \\
& + af(n/b) && (1) \\
& + a^2 f(n/b^2) && (2) \\
& + a^3 f(n/b^3) && (3) \\
& \vdots && \vdots \\
& + n^{\log_b a} && (\log_b n)
\end{aligned}
$$

$$f(n) = \Theta(n^{\log_b a}) \implies \mathrm{T}(n) =$$

$$
\begin{aligned}
\mathrm{T}(n) = {} & \Theta(n^{\log_b a}) & \\
& + af(n/b) & (1) \\
& + a^2 f(n/b^2) & (2) \\
& + a^3 f(n/b^3) & (3) \\
& \vdots & \vdots \\
& + \Theta(n^{\log_b a}) & (\log_b n)
\end{aligned}
$$

$$f(n) = \Theta(n^{\log_b a}) \implies \mathrm{T}(n) =$$

$$
\begin{aligned}
\mathrm{T}(n) &= \Theta(n^{\log_b a}) \\
&+ \Theta(n^{\log_b a}) && (1) \\
&+ \Theta(n^{\log_b a}) && (2) \\
&+ \Theta(n^{\log_b a}) && (3) \\
&\vdots && \vdots \\
&+ \Theta(n^{\log_b a}) && (\log_b n)
\end{aligned}
$$

$$f(n) = \Theta(n^{\log_b a}) \implies \mathrm{T}(n) =$$

$$
\begin{aligned}
\mathrm{T}(n) &= \Theta(n^{\log_b a}) && \\
&+ \Theta(n^{\log_b a}) && (1) \\
&+ \Theta(n^{\log_b a}) && (2) \\
&+ \Theta(n^{\log_b a}) && (3) \\
&\vdots && \vdots \\
&+ \Theta(n^{\log_b a}) && (\log_b n)
\end{aligned}
$$

$$f(n) = \Theta(n^{\log_b a}) \implies \mathrm{T}(n) = \Theta(n^{\log_b a} \cdot \log_b n)$$

$$
\begin{aligned}
T(n) &= \Theta(n^{\log_b a}) & \\
&+ \Theta(n^{\log_b a}) & (1)\\
&+ \Theta(n^{\log_b a}) & (2)\\
&+ \Theta(n^{\log_b a}) & (3)\\
&\;\vdots & \vdots\\
&+ \Theta(n^{\log_b a}) & (\log_b n)
\end{aligned}
$$

$$f(n) = \Theta(n^{\log_b a}) \implies T(n) = \Theta(n^{\log_b a} \cdot \lg n)$$

$$
\begin{aligned}
T(n) &= f(n) \\
&\quad + af(n/b) && (1) \\
&\quad + a^2 f(n/b^2) && (2) \\
&\quad + a^3 f(n/b^3) && (3) \\
&\quad \vdots && \vdots \\
&\quad + n^{\log_b a} && (\log_b n)
\end{aligned}
$$

$$f(n) = \Omega(n^{\log_b a + \epsilon}) \implies T(n) =$$

$$
\begin{aligned}
\mathrm{T}(n) = f(n) & \\
+\, af(n/b) & \qquad (1) \\
+\, a^2 f(n/b^2) & \qquad (2) \\
+\, a^3 f(n/b^3) & \qquad (3) \\
\vdots \quad & \qquad \vdots \\
+\, n^{\log_b a} & \qquad (\log_b n)
\end{aligned}
$$

Antar «regularitet»
$af(n/b) \leq cf(n)$
for store n, $c < 1$

$$f(n) = \Omega(n^{\log_b a+\epsilon}) \implies \mathrm{T}(n) =$$

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

$$+\ af(n/b) \qquad (1)$$

$$+\ a^2 f(n/b^2) \qquad (2)$$

$$+\ a^3 f(n/b^3) \qquad (3)$$

$$\vdots \qquad \vdots$$

$$+\ n^{\log_b a} \qquad (\log_b n)$$

Antar «regularitet»
$af(n/b) \leq cf(n)$
for store n, $c < 1$

$$f(n) = \Omega(n^{\log_b a + \epsilon}) \implies \mathrm{T}(n) =$$

Regularitetskriteriet er for at serien av a^i*f(n/b^i)-ledd ikke skal stige mer enn eksponentielt fort (altså ikke mer enn en konstantfaktor c per nivå); ellers vil ikke Omega-kriteriet vårt være nok til at f(n) dominerer kjøretiden asymptotisk.

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

$$+\, af(n/b) \qquad (1)$$

$$+\, a^2 f(n/b^2) \qquad (2)$$

$$+\, a^3 f(n/b^3) \qquad (3)$$

$$\vdots \qquad \vdots$$

$$+\, n^{\log_b a} \qquad (\log_b n)$$

Antar «regularitet»
$af(n/b) \leq cf(n)$
for store n, $c < 1$

$$f(n) = \Omega(n^{\log_b a + \epsilon}) \implies \mathrm{T}(n) = \Theta(f(n))$$

$$f(n) = \mathrm{O}(n^{\log_b a - \epsilon}) \implies \mathrm{T}(n) = \Theta(n^{\log_b a})$$

$$f(n) = \Theta(n^{\log_b a}) \implies \mathrm{T}(n) = \Theta(n^{\log_b a} \lg n)$$

$$f(n) = \Omega(n^{\log_b a + \epsilon}) \implies \mathrm{T}(n) = \Theta(f(n))$$

$$f(n) = \mathrm{O}(n^{\log_b a - \epsilon}) \implies \mathrm{T}(n) = \Theta(n^{\log_b a})$$

$$f(n) = \Theta(n^{\log_b a}) \implies \mathrm{T}(n) = \Theta(n^{\log_b a} \lg n)$$

$$f(n) = \Omega(n^{\log_b a + \epsilon}) \implies \mathrm{T}(n) = \Theta(f(n))$$

$af(n/b) \leq cf(n)$
for store n, $c < 1$

Her har jeg ikke kjørt verifikasjons-trinnet (med substitusjon/induksjon), men prøv gjerne selv!

$$f(n) = \mathrm{O}(n^{\log_b a-\epsilon}) \implies \mathrm{T}(n) = \Theta(n^{\log_b a})$$

$$f(n) = \Theta(n^{\log_b a}) \implies \mathrm{T}(n) = \Theta(n^{\log_b a} \lg n)$$

$$f(n) = \Omega(n^{\log_b a+\epsilon}) \implies \mathrm{T}(n) = \Theta(f(n))$$

$af(n/b) \leq cf(n)$
for store n, $c < 1$

Teorem 4.1: «The master theorem»

6:6

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

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

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

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

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

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

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

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

Potensen er fortsatt problematisk

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Slik ser ligningen vår ut med nye navn

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

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

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

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

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

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

(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 $m = \lg n$

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

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

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

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

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

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

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

Potensen er fortsatt problematisk

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

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

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

Gi $T(2^m)$ et nytt navn!

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

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

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

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

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

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

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

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

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

$$S(m) = 2S(m/2) + m$$

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

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

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

1. **Rekurrenser**
2. **Binærsøk**
3. **Merge sort**
4. **Quicksort**
5. **Masterteoremet**
6. **Variabelskifte**

1. Rekurrenser

2. Binærsøk

3. Merge sort

4. Quicksort

5. Masterteoremet

6. Variabelskifte

1. Rekurrenser

2. Binærsøk

3. Merge sort

4. Quicksort

5. Masterteoremet

6. Variabelskifte

D&C

1. Rekurrenser

2. Binærsøk

3. Merge sort

5. Masterteoremet

6. Variabelskifte

D&C

1. Rekur

2. Binærsøk

3. Merge sort

5. Masterteoremet

D&C

6. Variabel