

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

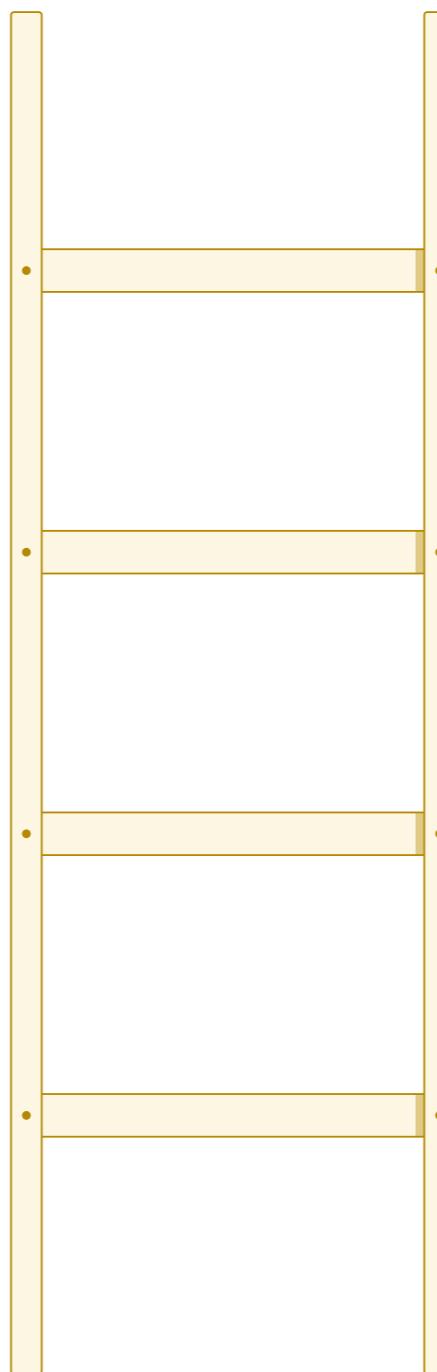
- Kap. 2. Getting started: 2.3
- Kap. 4. Divide-and-conquer:
Innledning, 4.1 og 4.3–4.5
- Kap. 7. Quicksort
- Oppgaver 2.3-5 og 4.5-3 med
løsning (binærersøk)
- Appendiks B og C i pensumheftet

Læringsmål

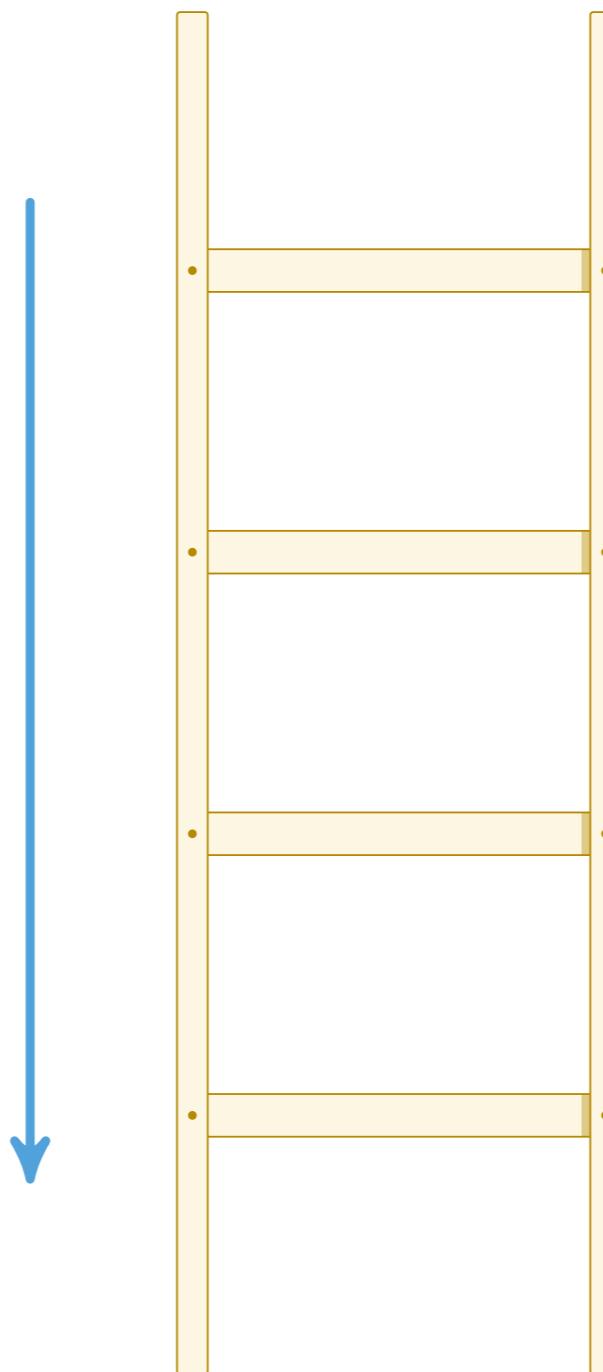
- [C₁] Forstå *divide-and-conquer* (*splitt og hersk*)
- [C₂] Forstå *maximum-subarray*-problemet med løsninger
- [C₃] Forstå BISECT og BISECT' (se appendiks C i pensumheftet)
- [C₄] Forstå MERGE-SORT
- [C₅] Forstå QUICKSORT og RANDOMIZED-QUICKSORT
- [C₆] Kunne løse rekurrens med *substitusjon*, *rekursjonstrær* og *masterteoremet*
- [C₇] Kunne løse rekurrens med *iterasjonsmetoden* (se appendiks B i pensumheftet)
- [C₈] Forstå hvordan *variabelskifte* fungerer

Forelesningen filmes

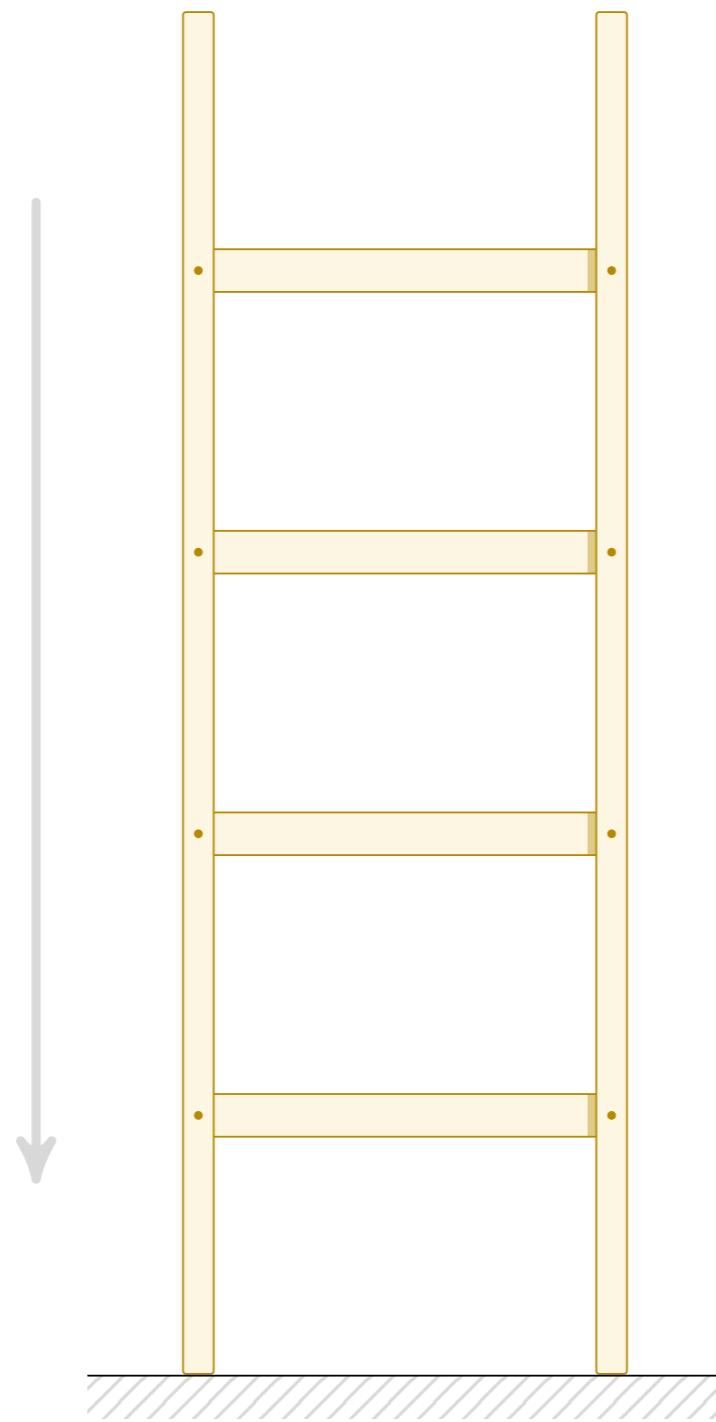




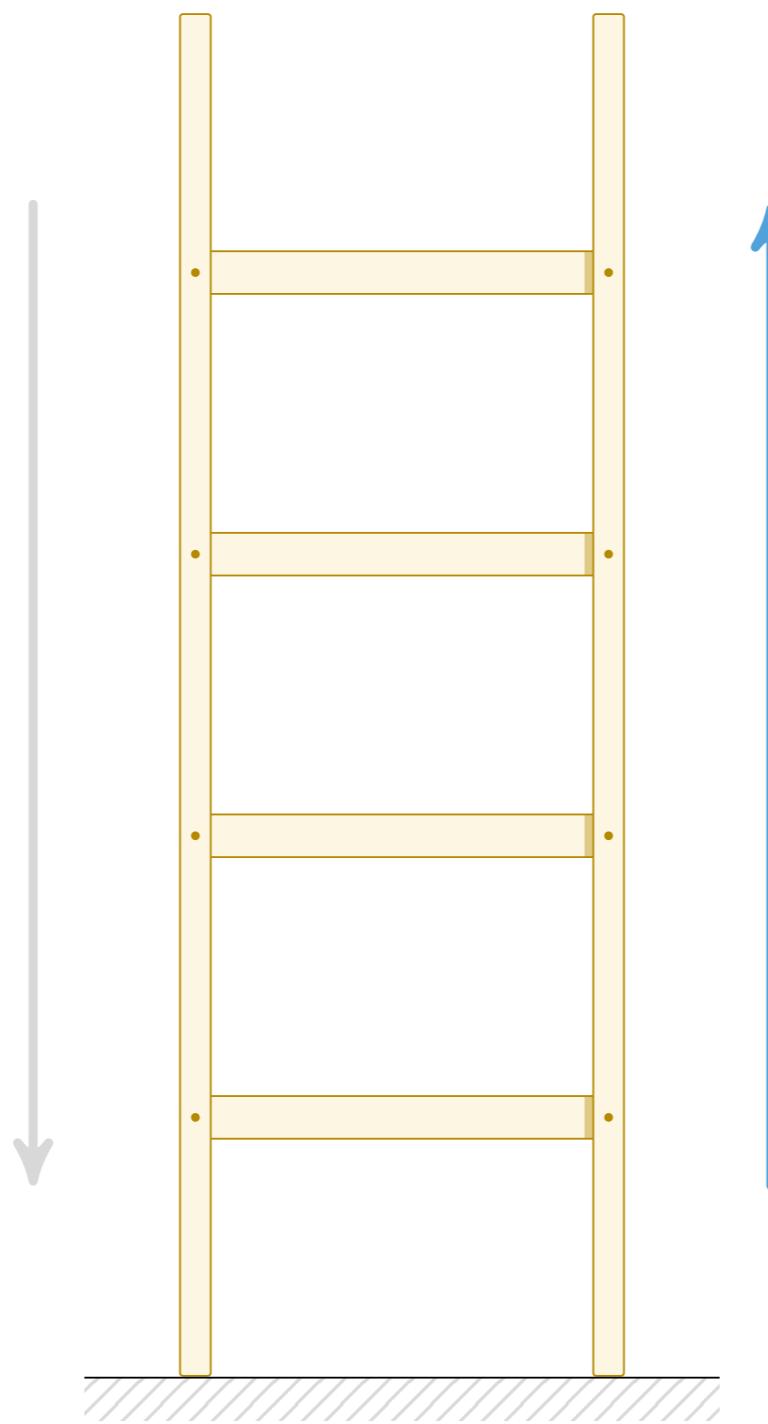
Dette er den velkjente induksjonsstigen vår



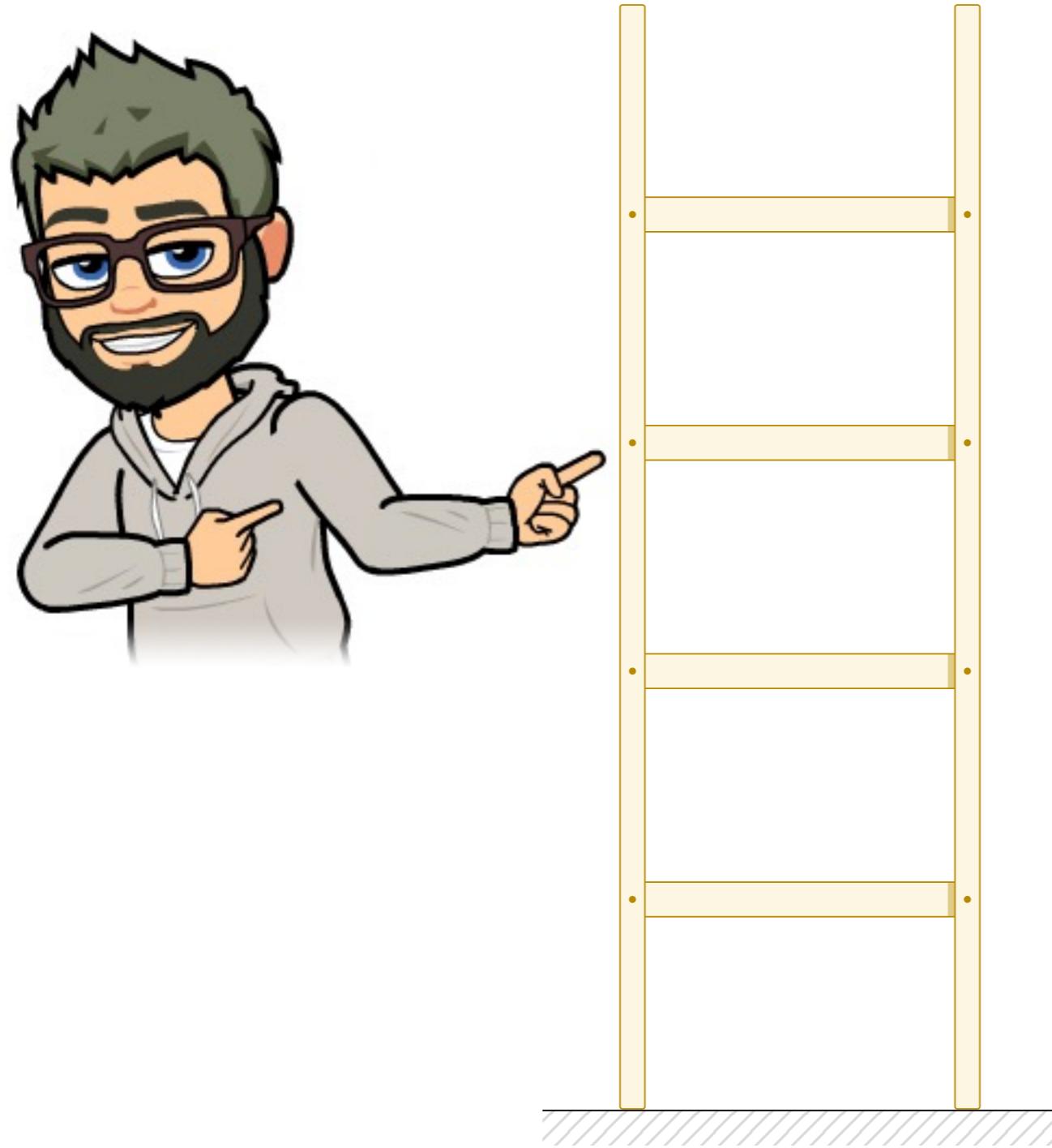
Vi reduserer en instans til mindre instanser . . .



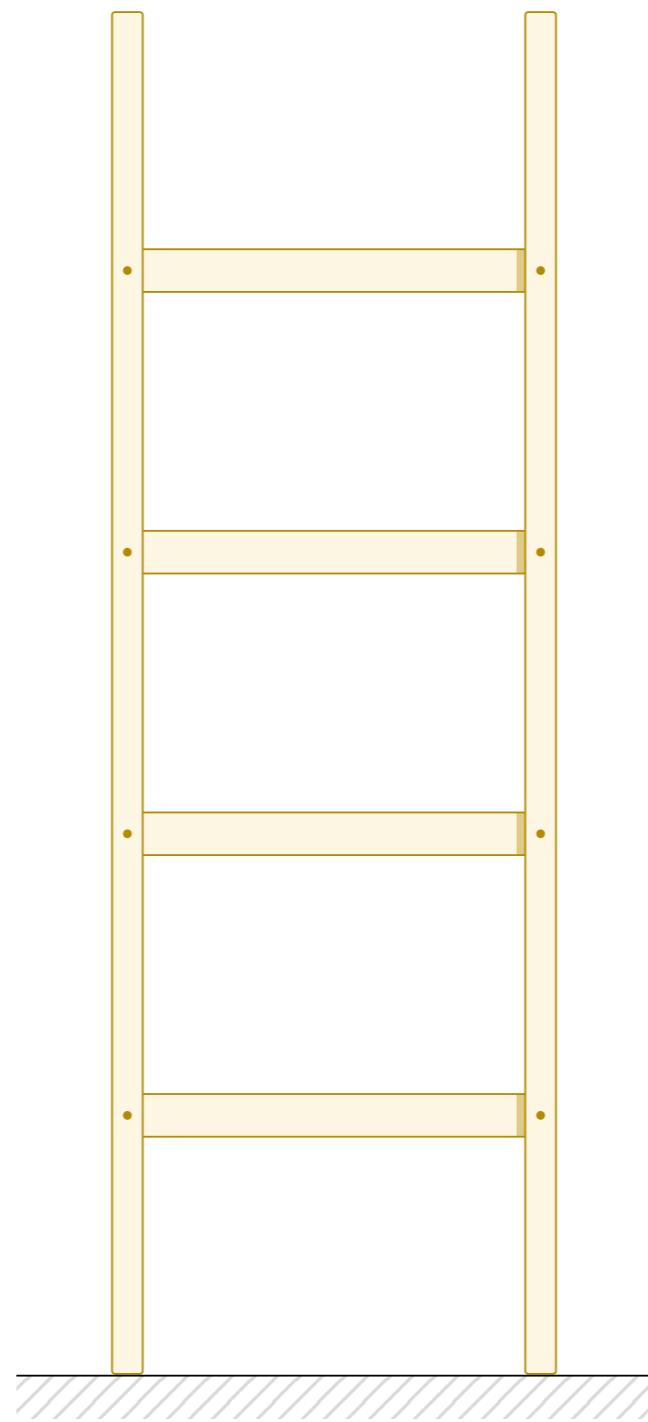
... til vi treffer grunntilfellet



Vi løser grunntilfellet og bygger vi oss opp igjen



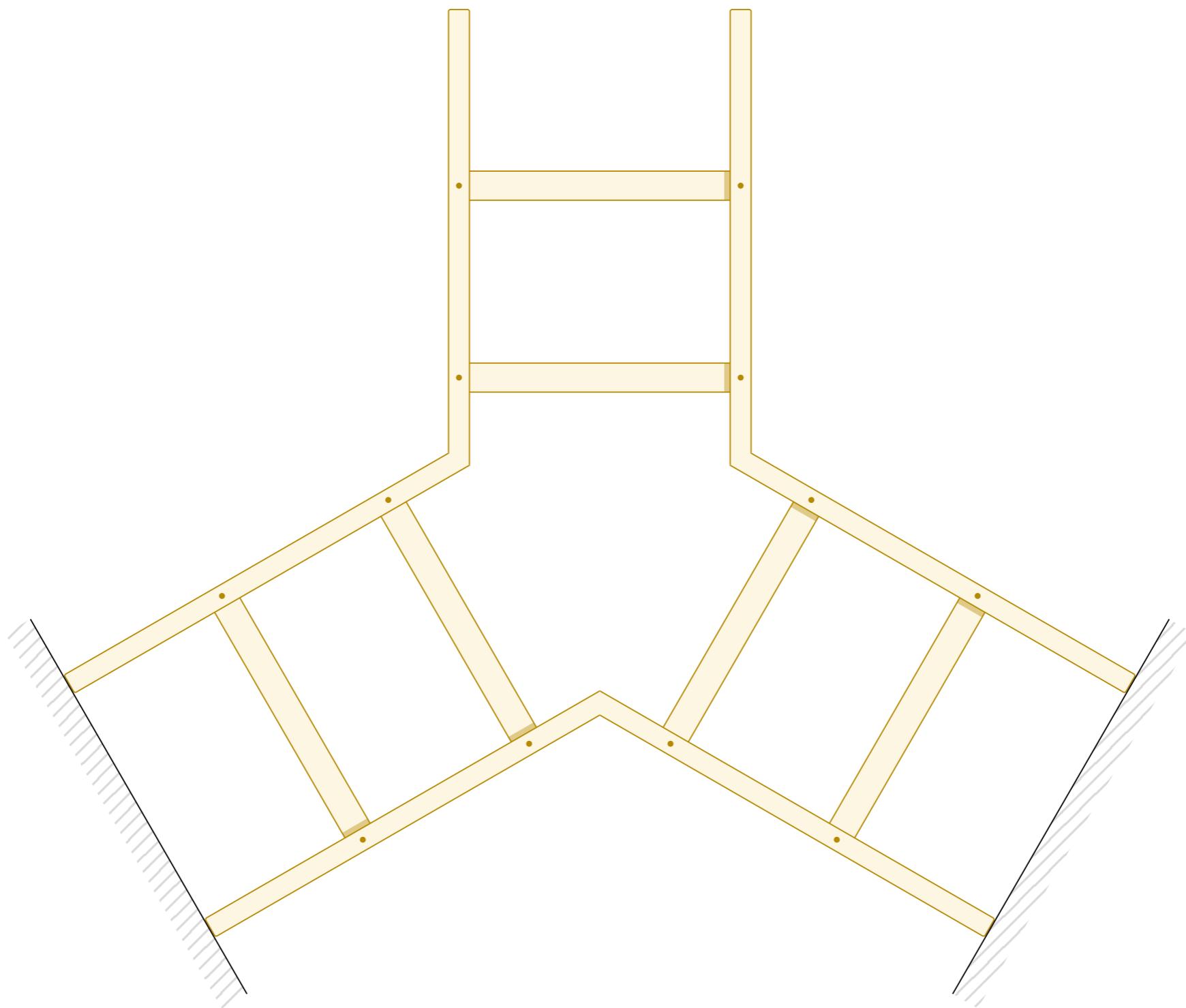
Alle induktive trinn er like, så ser bare på ett av dem



Men vi kan også spalte i flere delinstanser



Men vi kan også spalte i flere delinstanser



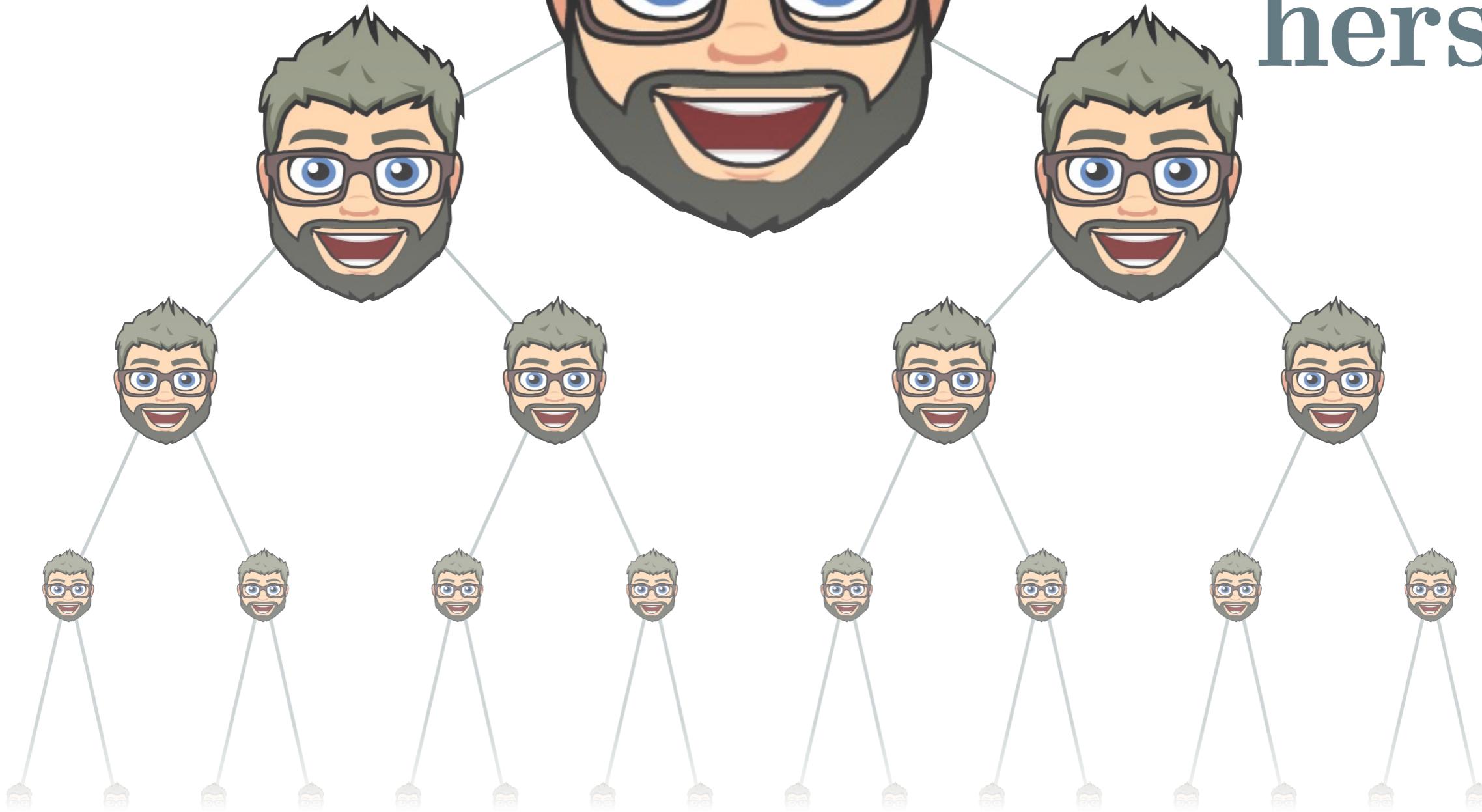
Denne designmetoden kalles «splitt og hersk» (*divide & conquer*)

Forelesning

3



Splitt &
hersk



1. Rekurrenser

2. Binærsøk

3. Merge sort

4. Quicksort

5. Masterteoremet

6. Variabelskifte

4.1: Selvstudium

	Januāri
i	Januāri
ii	Febrūri
iii	Martī
iv	Aprilē
v	Mayō
vi	June
vii	July
viii	Augustō
ix	Septēmber
x	Octōber
xi	Novēmber
xii	Dēcēmber

1:6

Rekurrenser

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.

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

- Finn løsning:

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

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

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

$$T(1) = 1$$

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

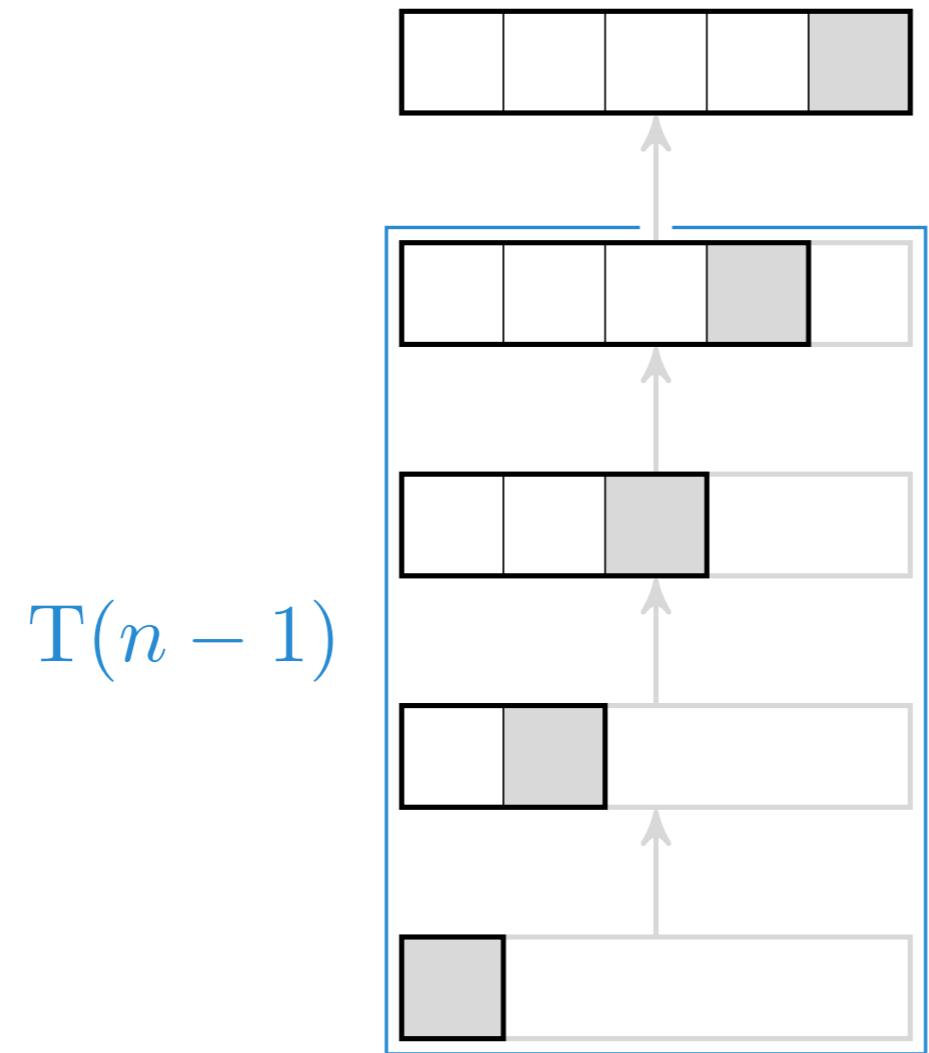
Eksempel: $1 + 1 + \dots$

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

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

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

D&C → rekurrensen → $T(n) = T(n - 1) + 1$



Rekurrens-tre

D&C → rekureren → $T(n) = 1 + T(n - 1)$

$$T(n) = 1$$

$$+ T(n - 1) \quad (1)$$

D&C → rekurrensen → $\text{T}(\textcolor{blue}{n}) = 1 + \text{T}(n - 1)$

$$\text{T}(n) = 1$$

$$+ \text{T}(\textcolor{blue}{n} - 1) \quad (1)$$

D&C → rekurrensen → $T(n) = 1 + T(\textcolor{blue}{n} - 1)$

$$T(n) = 1$$

$$+ \textcolor{red}{1} \tag{1}$$

$$+ T(\textcolor{blue}{n} - 1 - 1) \tag{2}$$

D&C → rekureren → $T(n) = 1 + T(n - 1)$

$$T(n) = 1$$

$$+ 1 \tag{1}$$

$$+ T(n - 2) \tag{2}$$

D&C → rekureren → $T(n) = 1 + T(n - 1)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ T(n - 3) \quad (3)$$

D&C → rekureren → $T(n) = 1 + T(n - 1)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

⋮

$$+ T(n - ?) \quad (?)$$

D&C → rekureren → $T(n) = 1 + T(n - 1)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

⋮

⋮

$$+ T(n - ?) \quad (n - 1)$$

D&C → rekurrensen → $T(n) = 1 + T(n - 1)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

$$\vdots \qquad \vdots$$

$$+ T(n - (n - 1)) \quad (n - 1)$$

D&C → rekureren → $T(n) = 1 + T(n - 1)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

$$\vdots \qquad \vdots$$

$$+ T(1) \quad (n - 1)$$

D&C → rekureren → $T(n) = 1 + T(n - 1)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

⋮

⋮

$$+ 1 \quad (n - 1)$$

D&C → rekureren → $T(n) = 1 + T(n - 1)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

$$\vdots \qquad \vdots$$

$$+ 1 \quad (n - 1)$$

$$T(n) = n$$

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.

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

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.

D&C → rekureren → $T(n) = T(n - 1) + 1$

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

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

D&C → rekureren → $T(n) = T(n - 1) + 1$

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

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

D&C → rekurrensen → $T(n) = T(n - 1) + 1$

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

$$= (\textcolor{blue}{n} - 1) + 1$$

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

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

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

$$= n - 1 + 1$$

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

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

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

$$= n - 1 + 1$$

$$= n$$

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

D&C → rekurrensen → $T(n) = T(n - 1) + 1$

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

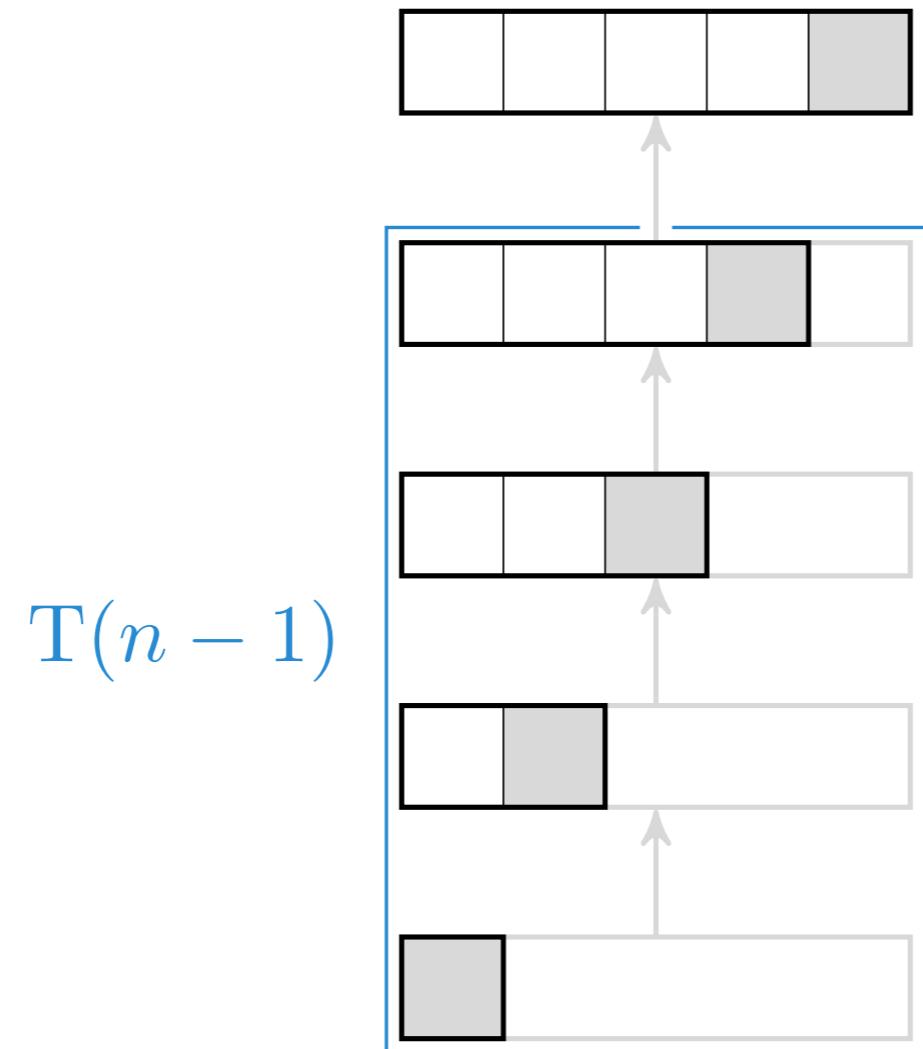
$$= (n - 1) + 1$$

$$= n - 1 + 1$$

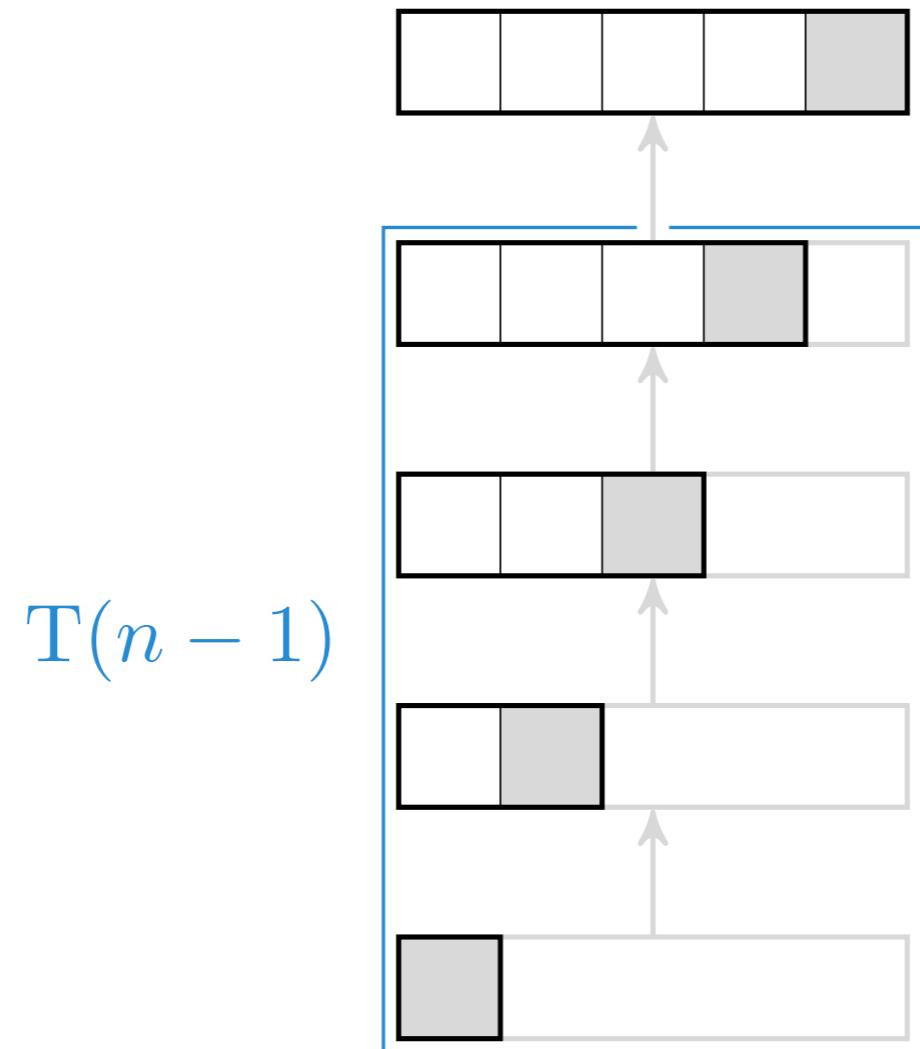
$$= n$$

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

D&C → rekurrensen → $T(n) = T(n - 1) + 1$



D&C → rekurrensen → $T(n) = T(n - 1) + 1$



$T(n) = n$

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

Binærso

324

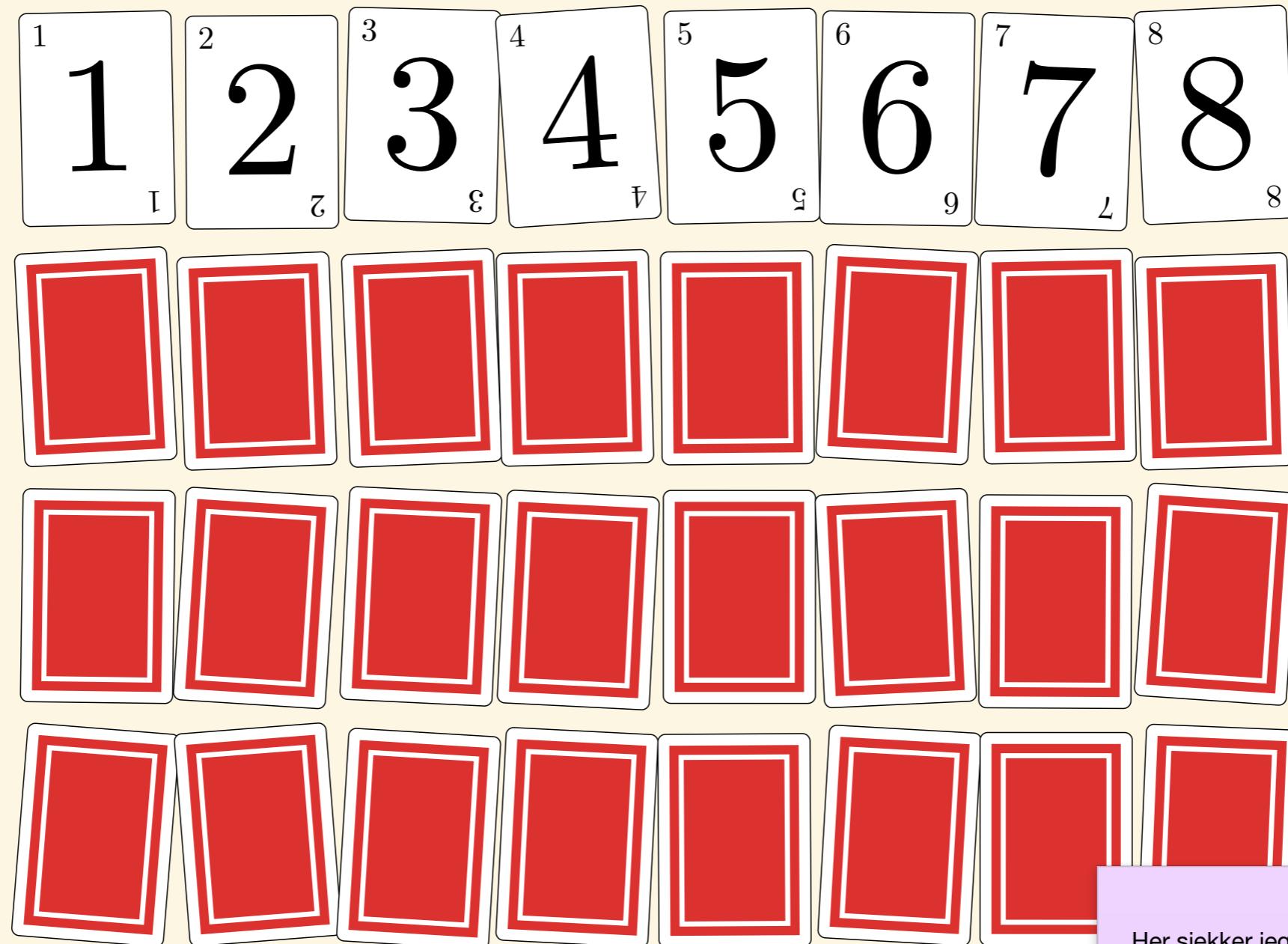
HERON OF ALEXANDRIA

'Since', says Héron,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 $26\frac{1}{3}$. The side of 720 will therefore be very nearly $26\frac{1}{3}$. In fact, if we make $26\frac{2}{3}$ by itself, the product is $720\frac{1}{3}$. If we make the square) is $\frac{1}{3}$.

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

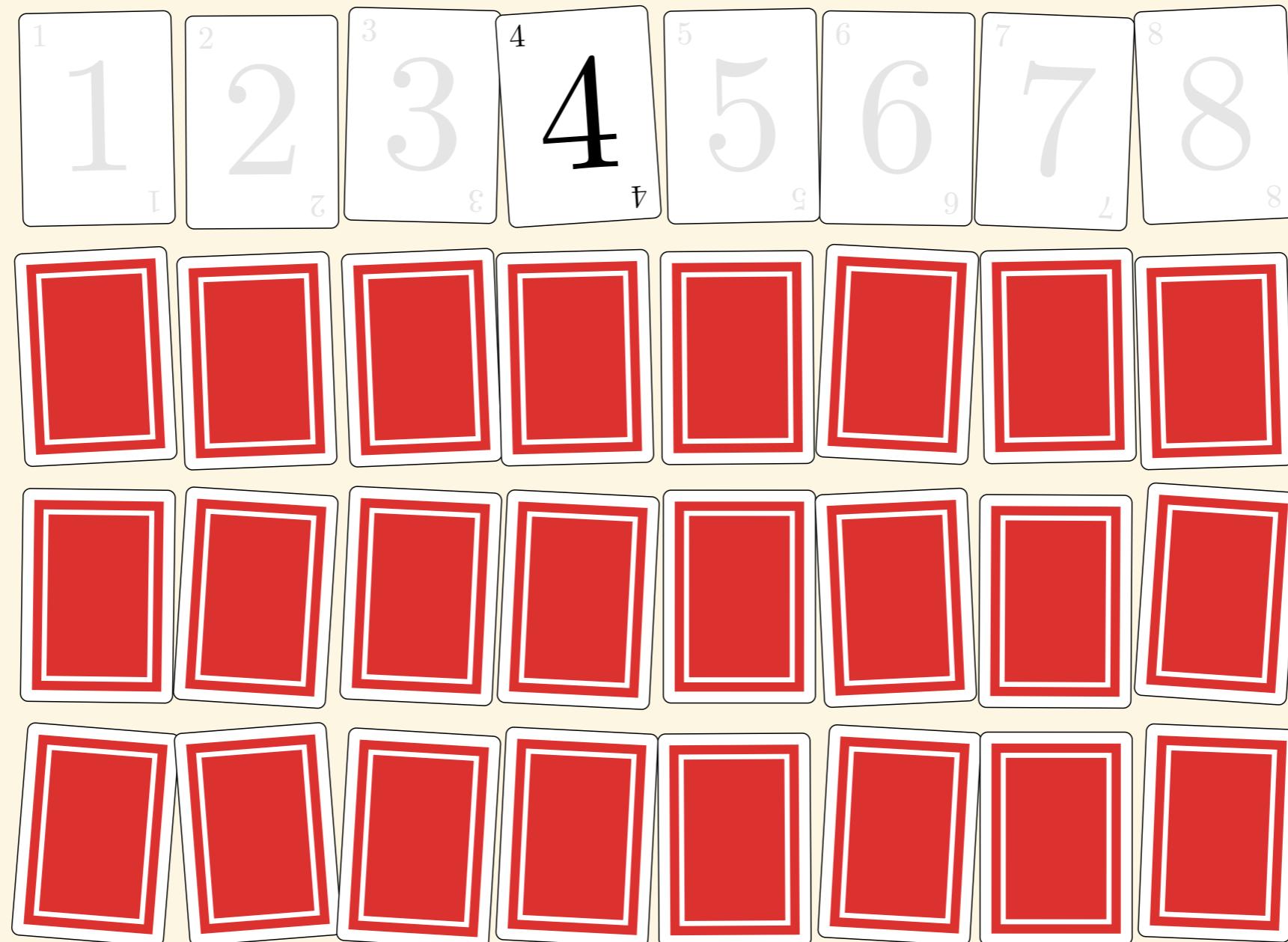
(Heath, A History of Greek Mathematics, vol. 2, 1921; Fowler & Robson, Square Root Approximations in Old Babylonian Mathematics: YBC 7289 in Context, 1998)

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

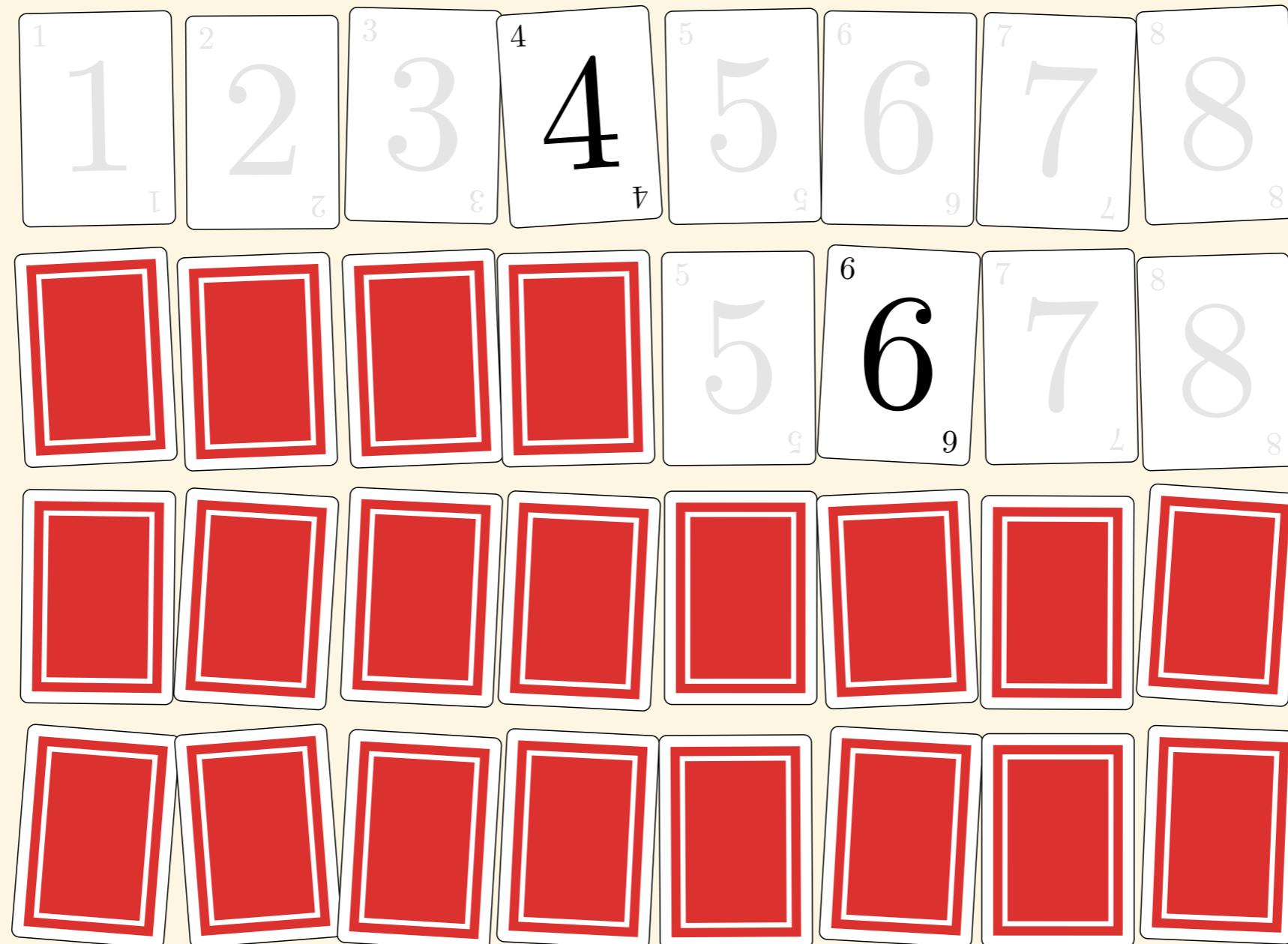


Her sjekker jeg bare hvilken halvdel jeg skal lete i – og sjekker ikke om kortet vi ser på er det vi er ute etter før vi kommer til bunnen. I pseudokoden senere har vi den sjekken i tillegg.

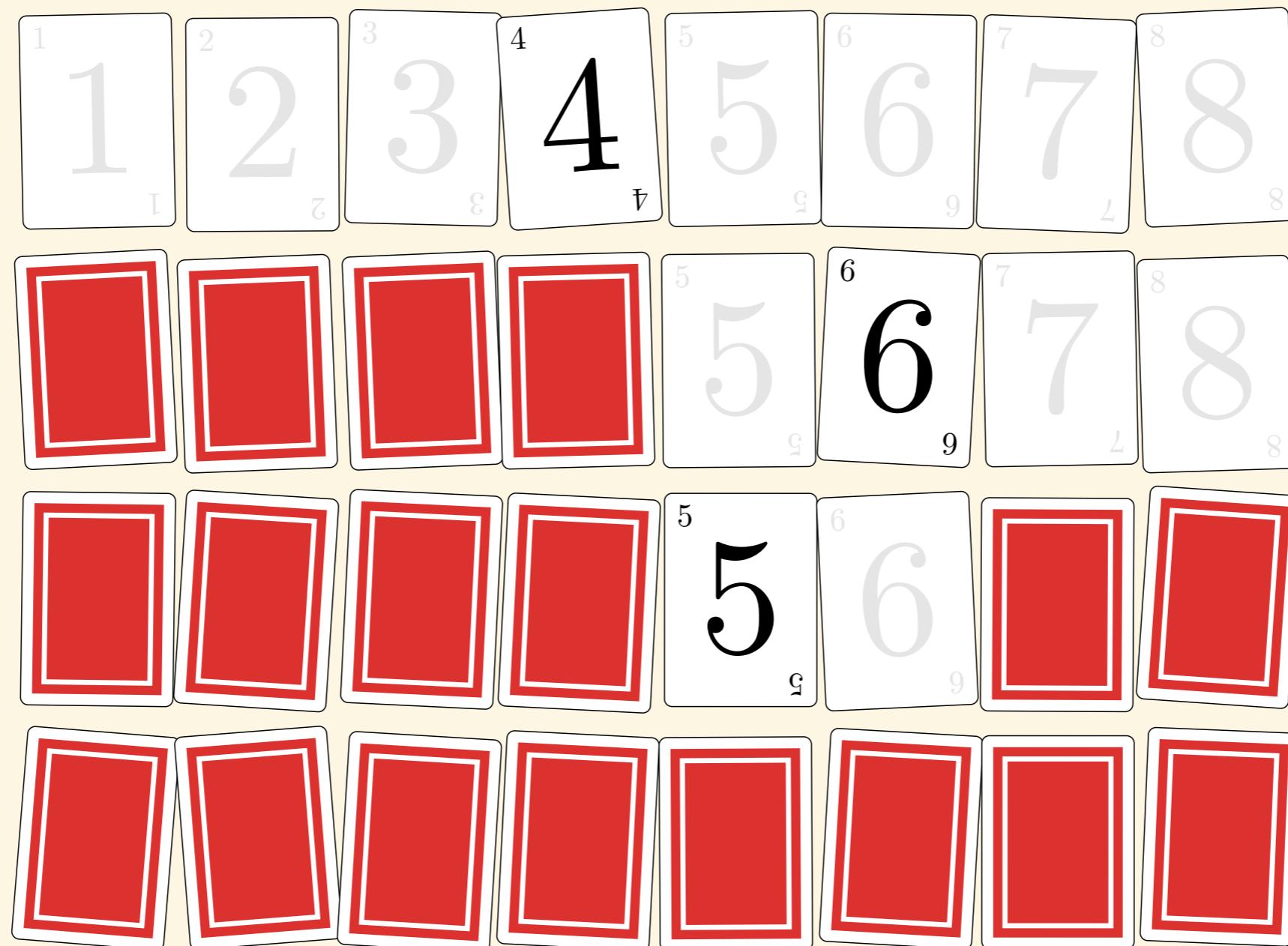
(Anta at kortet ditt er 6)



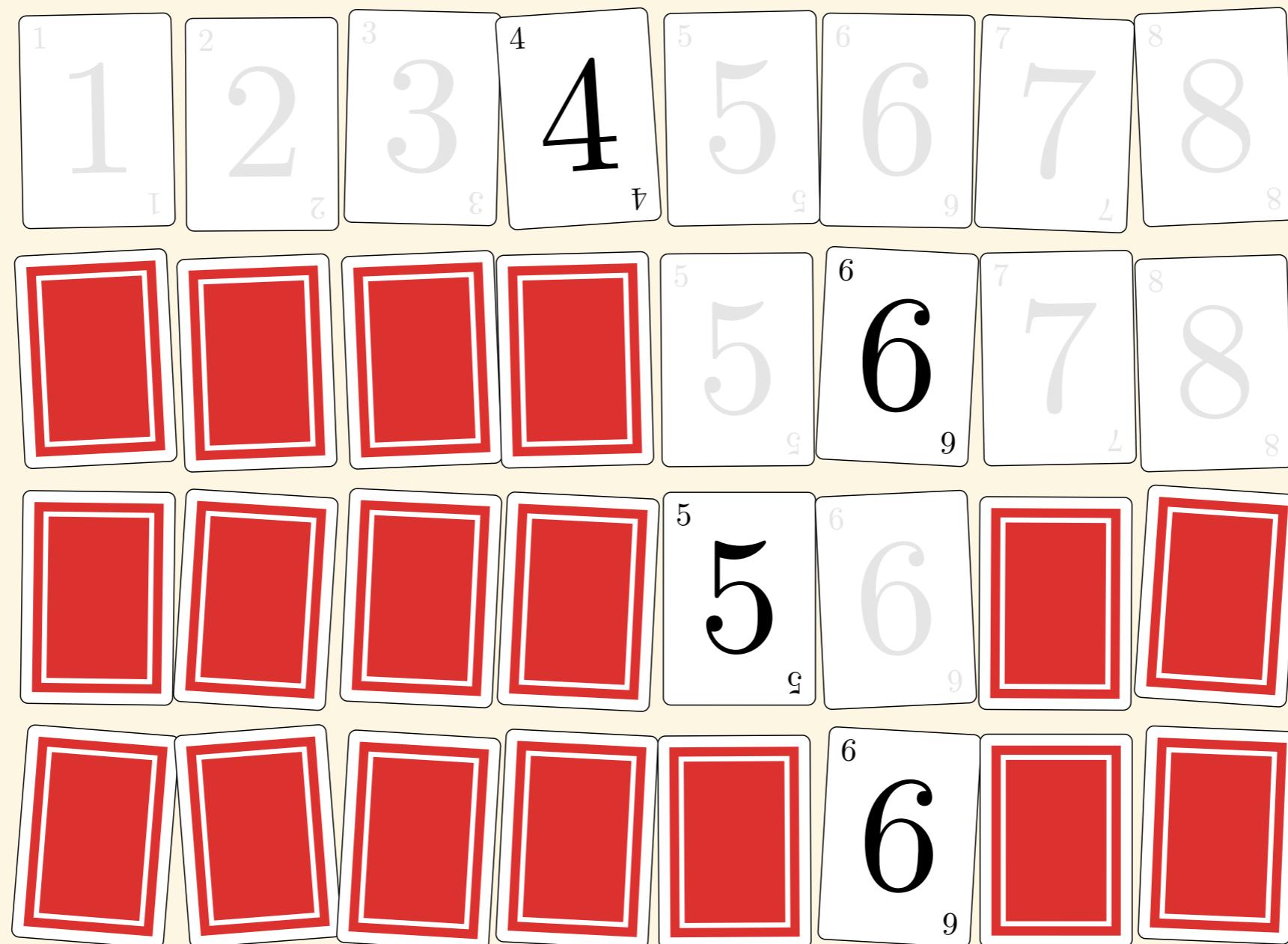
Er kortet større enn A[4]?



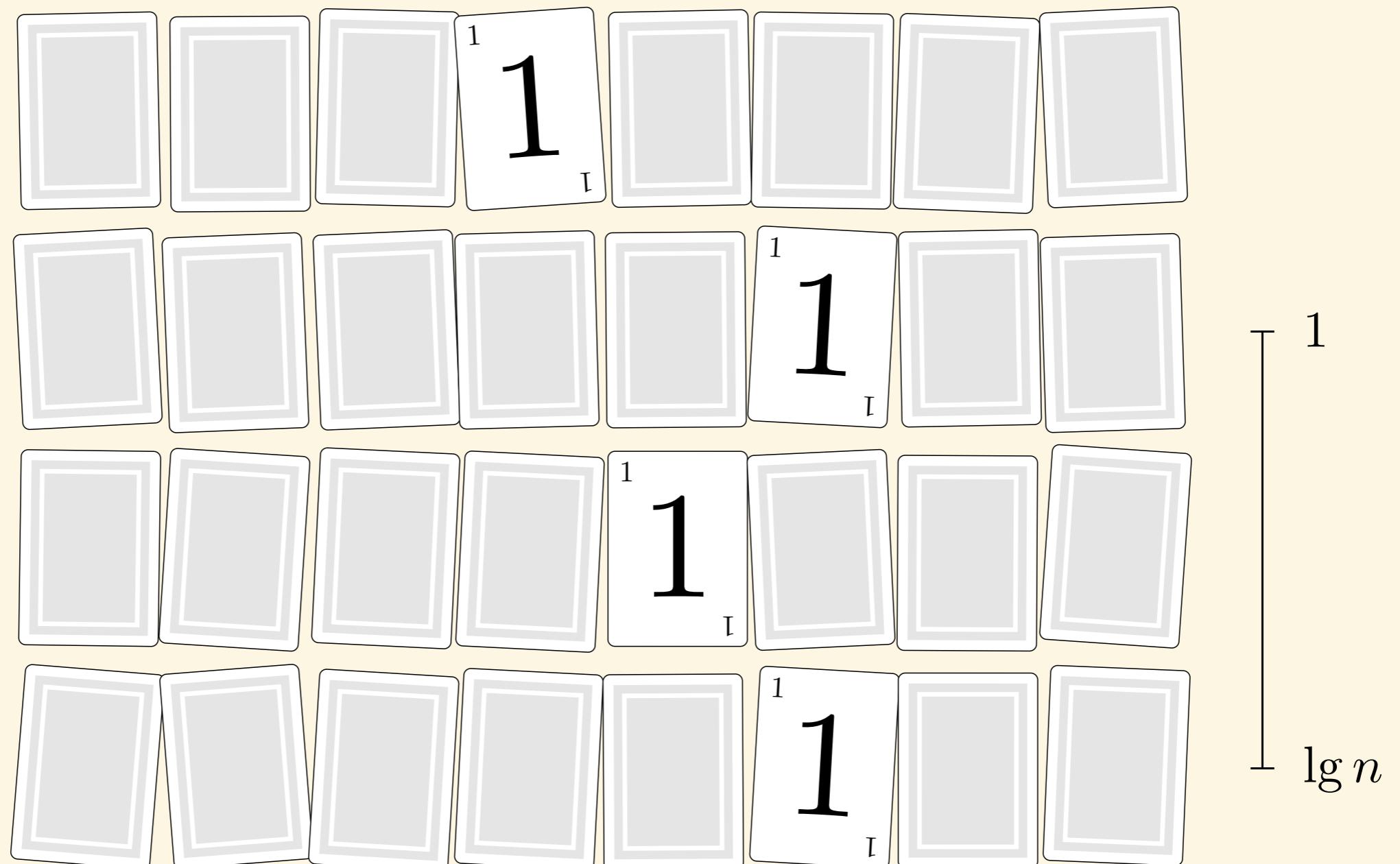
Er kortet større enn A[6]?



Er kortet større enn A[5]?



Er A[6] kortet ditt?



Antall spørsmål: $\lg n + 1$

BISECT(A, p, r, v)

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

BISECT(A, p, r, v)

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

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

```
BISECT(A, p, r, v)
1  if  $p \leq r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
```

Del på midten

```
BISECT(A, p, r, v)
1  if  $p \leq r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      if  $v == A[q]$ 
```

Er dette verdien vi leter etter?

```
BISECT(A, p, r, v)
1  if  $p \leq r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      if  $v == A[q]$ 
4          return  $q$ 
```

I så fall er vi ferdige

```
BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
```

Ellers: Må v ligge til venstre?

```
BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
```

Let videre til venstre

```
BISECT(A, p, r, v)
1  if  $p \leq r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      if  $v == A[q]$ 
4          return  $q$ 
5      elseif  $v < A[q]$ 
6          return BISECT(A, p, q - 1, v)
7      else return BISECT(A, q + 1, r, v)
```

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

```
BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL
```

Vi fant ikke v , så vi returnerer NIL

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  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)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  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)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  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)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  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)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  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)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A, p, q - 1, v)
7    else return BISECT(A, q + 1, r, v)
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A, p, q - 1, v)
7    else return BISECT(A, q + 1, r, v)
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

p, r

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

p, r

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

p, q, r

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A, p, q - 1, v)
7    else return BISECT(A, q + 1, r, v)
8  return NIL

```

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

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

p, q, r

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL
→ 5

```

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

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

p, q, r

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A, p, q - 1, v)
7    else return BISECT(A, q + 1, r, v)
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

→ 5

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

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

p

r

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  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)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL
→ 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)
1  if  $p \leq r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      if  $v == A[q]$ 
4          return  $q$ 
5      elseif  $v < A[q]$ 
6          return BISECT(A,  $p, q - 1, v$ )
7      else return BISECT(A,  $q + 1, r, v$ )
8  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)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  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)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A, p, q - 1, v)
7    else return BISECT(A, q + 1, r, v)
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A, p, q - 1, v)
7    else return BISECT(A, q + 1, r, v)
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      if  $v == A[q]$ 
4          return  $q$ 
5      elseif  $v < A[q]$ 
6          return BISECT(A,  $p, q - 1, v$ )
7      else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

p, r

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

p, r

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

p, q, r

```
BISECT(A, p, r, v)
```

```
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL
```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

p, q, r	5	5
	5	6
	6	7
	7	8

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2       $q = \lfloor (p + r)/2 \rfloor$ 
3      if  $v == A[q]$ 
4          return  $q$ 
5      elseif  $v < A[q]$ 
6          return BISECT(A,  $p, q - 1, v$ )
7      else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A, p, q - 1, v)
7    else return BISECT(A, q + 1, r, v)
8  return NIL

```

→ NIL

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

→ NIL

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

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

p, r

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A, p, q - 1, v)
7    else return BISECT(A, q + 1, r, v)
8  return NIL

```

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

→ NIL

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

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

```

BISECT(A, p, r, v)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  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)
1  if  $p \leq r$ 
2     $q = \lfloor (p + r)/2 \rfloor$ 
3    if  $v == A[q]$ 
4      return  $q$ 
5    elseif  $v < A[q]$ 
6      return BISECT(A,  $p, q - 1, v$ )
7    else return BISECT(A,  $q + 1, r, v$ )
8  return NIL

```

→ NIL

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

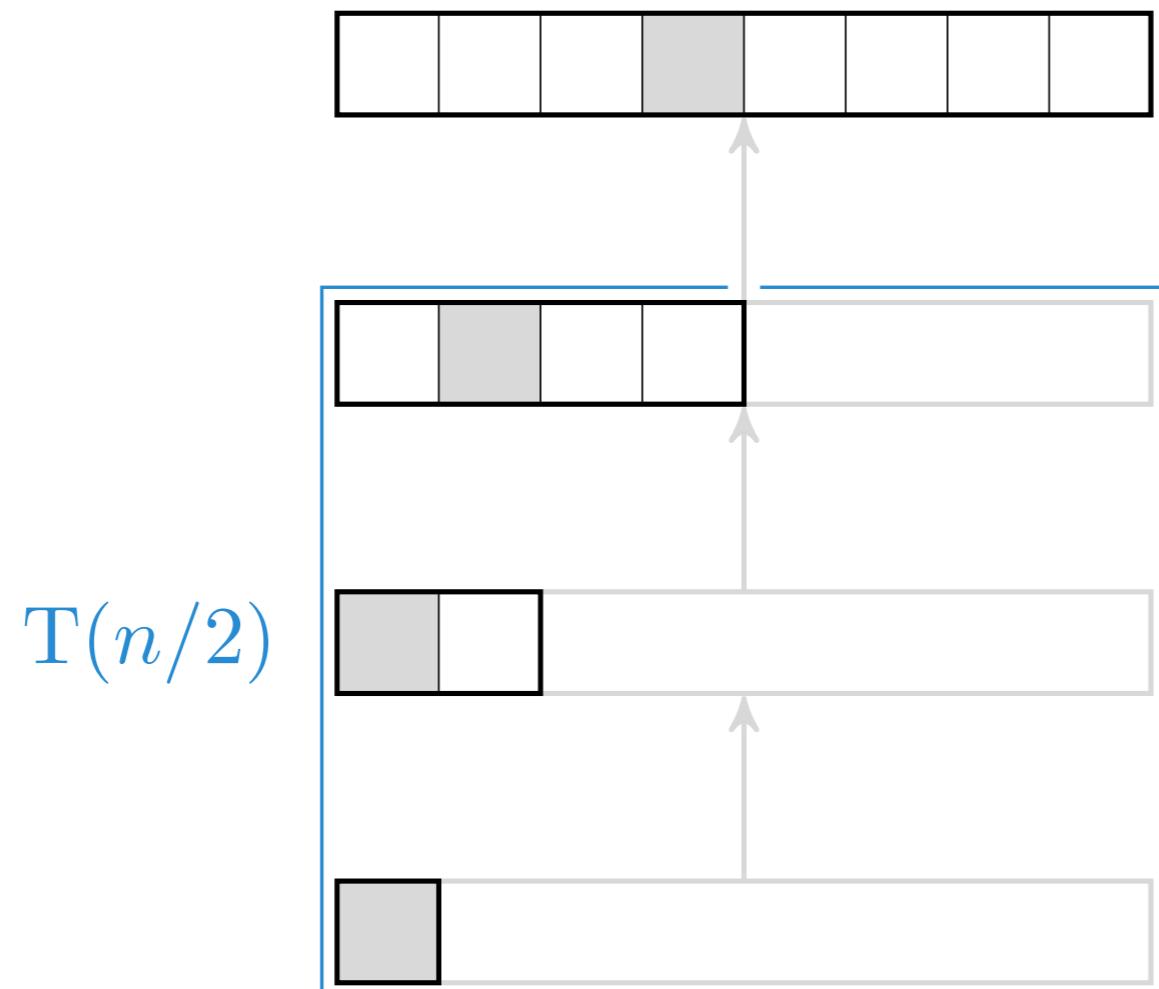
p	1	1
	2	2
	2	3
	3	4
	5	5
	5	6
	6	7
r	7	8

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

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.

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

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



D&C → binærsøk → $T(n) = 1 + T(n/2)$

$$T(n) = 1$$

$$+ T(n/2) \quad (1)$$

D&C → binærsøk → $\text{T}(\textcolor{blue}{n}) = 1 + \text{T}(n/2)$

$$\text{T}(n) = 1$$

$$+ \text{T}(\textcolor{blue}{n}/2) \quad (1)$$

D&C → binærsøk → $T(n) = 1 + T(\textcolor{blue}{n}/2)$

$$T(n) = 1$$

$$+ \textcolor{red}{1} \tag{1}$$

$$+ T(\textcolor{blue}{n}/2/2) \tag{2}$$

D&C → binærsøk → $T(n) = 1 + T(n/2)$

$$T(n) = 1$$

$$+ 1 \tag{1}$$

$$+ T(n/4) \tag{2}$$

D&C → binærsøk → $T(n) = 1 + T(n/2)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ T(n/8) \quad (3)$$

D&C → binærsøk → $T(n) = 1 + T(n/2)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

⋮

⋮

$$+ T(n/2^?) \quad (?)$$

D&C → binærsøk → $T(n) = 1 + T(n/2)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

⋮

⋮

$$+ T(n/2^?) \quad (\lg n)$$

D&C → binærsøk → $T(n) = 1 + T(n/2)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

⋮

⋮

$$+ T(n/2^{\lg n}) \quad (\lg n)$$

D&C → binærsøk → $T(n) = 1 + T(n/2)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

⋮

⋮

$$+ T(n/n) \quad (\lg n)$$

D&C → binærsøk → $T(n) = 1 + T(n/2)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

$$\vdots \quad \vdots$$

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

D&C → binærsøk → $T(n) = 1 + T(n/2)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

$$\vdots \quad \vdots$$

$$+ 1 \quad (\lg n)$$

D&C → binærsøk → $T(n) = 1 + T(n/2)$

$$T(n) = 1$$

$$+ 1 \quad (1)$$

$$+ 1 \quad (2)$$

$$+ 1 \quad (3)$$

⋮

⋮

$$+ 1 \quad (\lg n)$$

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

Verifikasjon

Med substitusjon/induksjon

D&C → binærsøk → $T(n) = T(n/2) + 1$

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

D&C → binærsøk → $T(n) = T(n/2) + 1$

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

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

D&C ↗ binærsøk ↗ $T(n) = T(n/2) + 1$

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

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

D&C → binærsøk → $T(n) = T(n/2) + 1$

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

$$= \left(\lg \frac{n}{2} + 1 \right) + 1$$

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

D&C ↗ binærsøk ↗ $T(n) = T(n/2) + 1$

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

$$= \left(\lg \frac{n}{2} + 1 \right) + 1$$

$$= \lg n - \lg 2 + 2$$

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

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

$$= \left(\lg \frac{n}{2} + 1 \right) + 1$$

$$= \lg n - \lg 2 + 2$$

$$= \lg n + 1$$

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

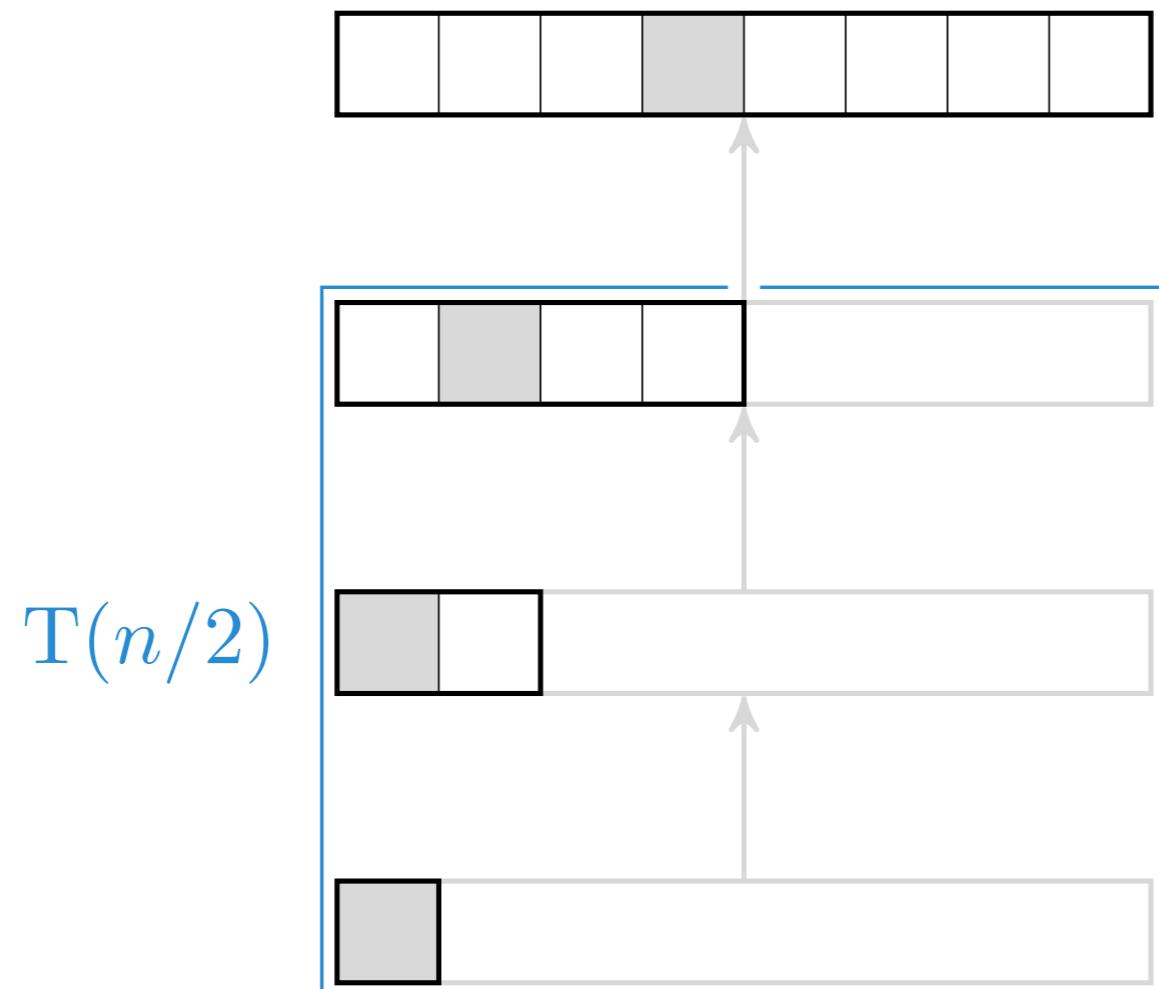
$$T(n) = T(n/2) + 1$$

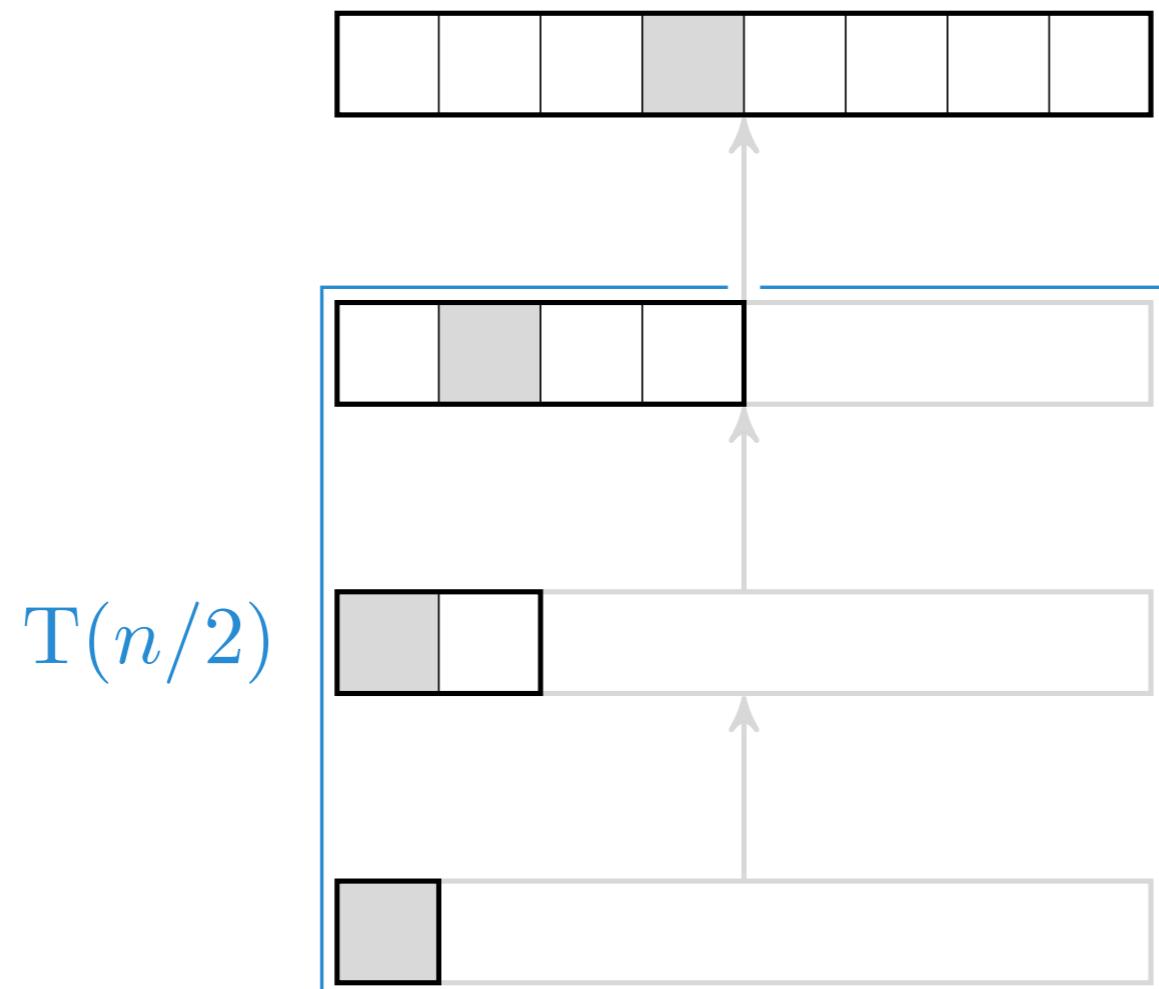
$$= \left(\lg \frac{n}{2} + 1 \right) + 1$$

$$= \lg n - \lg 2 + 2$$

$$= \lg n + 1$$

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





$T(n/2)$

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

«[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

⑤

(g) We now formulate a set of instructions to effect this 4-way decision between $(\alpha)-(s)$. We state again the contents of the short tanks already assigned:

1.) $N^{m(20)}$ 2.) $N^{m(20)}$ 3.) N^x 4.) N^y
5.) $N^{m(20)}$ 6.) $N^{m(20)}$ 7.) $N^{(2(20))}$ 8.) $N^{(2(20))}$
9.) $N^{(2(20))}$ 10.) $N^{(2(20))}$ 11.) $\dots \rightarrow C$

Now let the instructions occupy the (long tank) words 1., 2., ..

1.) $T_1 - S_1$ 2.) $\bar{S}_1 - T_1$
3.) $N^{m-m(20)}$ 4.) $N^{(2(20))}$ for $m = m$
5.) $N^{(2(20))}$ 6.) $N^{(2(20))}$ for $m \geq m$
7.) $N^{(2(20))}$ 8.) $N^{(2(20))}$ for $m \leq m$
9.) $N^{(2(20))}$ 10.) $N^{(2(20))}$ for $m \geq m$
11.) $N^{(2(20))}$ 12.) $N^{(2(20))}$ for $m \leq m$

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**

MERGE(A, p, q, r)

D&C ↗ merge sort

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

MERGE(A, p, q, r)

$$1 \quad n_1 = q - p + 1$$

D&C ↗ merge sort

Lengden til første segment, $A[p..q]$

MERGE(A, p, q, r)

$$1 \quad n_1 = q - p + 1$$

$$2 \quad n_2 = r - q$$

D&C ↗ merge sort

Lengden til andre segment, $A[p + 1..r]$

MERGE(A, p, q, r)

D&C ↗ merge sort

1 $n_1 = q - p + 1$

2 $n_2 = r - q$

3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays

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

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1

For hver indeks i i L...

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
- 5 $L[i] = A[p + i - 1]$

Kopier tilsvarende element fra $A[p..q]$

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2

For hver indeks j i R...

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2
7 $R[j] = A[q + j]$

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

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
 - 5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2
 - 7 $R[j] = A[q + j]$
- 8 $L[n_1 + 1] = \infty$

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

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
 - 5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2
 - 7 $R[j] = A[q + j]$
- 8 $L[n_1 + 1] = \infty$
- 9 $R[n_2 + 1] = \infty$

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

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
 5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2
 7 $R[j] = A[q + j]$
- 8 $L[n_1 + 1] = \infty$
- 9 $R[n_2 + 1] = \infty$
- 10 $i = 1$

Hvor er vi i L?

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
 5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2
 7 $R[j] = A[q + j]$
- 8 $L[n_1 + 1] = \infty$
- 9 $R[n_2 + 1] = \infty$
- 10 $i = 1$
- 11 $j = 1$

Hvor er vi i R?

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
 - 5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2
 - 7 $R[j] = A[q + j]$
- 8 $L[n_1 + 1] = \infty$
- 9 $R[n_2 + 1] = \infty$
- 10 $i = 1$
- 11 $j = 1$
- 12 **for** $k = p$ **to** r

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

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
 5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2
 7 $R[j] = A[q + j]$
- 8 $L[n_1 + 1] = \infty$
- 9 $R[n_2 + 1] = \infty$
- 10 $i = 1$
- 11 $j = 1$
- 12 **for** $k = p$ **to** r
 13 **if** $L[i] \leq R[j]$

Hvis $L[i]$ er minst ...

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
 5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2
 7 $R[j] = A[q + j]$
- 8 $L[n_1 + 1] = \infty$
- 9 $R[n_2 + 1] = \infty$
- 10 $i = 1$
- 11 $j = 1$
- 12 **for** $k = p$ **to** r
 13 **if** $L[i] \leq R[j]$
 14 $A[k] = L[i]$

Bruk $L[i]$

MERGE(A, p, q, r)

D&C ↳ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
 - 5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2
 - 7 $R[j] = A[q + j]$
- 8 $L[n_1 + 1] = \infty$
- 9 $R[n_2 + 1] = \infty$
- 10 $i = 1$
- 11 $j = 1$
- 12 **for** $k = p$ **to** r
 - 13 **if** $L[i] \leq R[j]$
 - 14 $A[k] = L[i]$
 - 15 $i = i + 1$

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

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
 - 5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2
 - 7 $R[j] = A[q + j]$
- 8 $L[n_1 + 1] = \infty$
- 9 $R[n_2 + 1] = \infty$
- 10 $i = 1$
- 11 $j = 1$
- 12 **for** $k = p$ **to** r
 - 13 **if** $L[i] \leq R[j]$
 - 14 $A[k] = L[i]$
 - 15 $i = i + 1$
 - 16 **else** $A[k] = R[j]$

MERGE(A, p, q, r)

D&C ↗ merge sort

- 1 $n_1 = q - p + 1$
- 2 $n_2 = r - q$
- 3 let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
- 4 **for** $i = 1$ **to** n_1
 - 5 $L[i] = A[p + i - 1]$
- 6 **for** $j = 1$ **to** n_2
 - 7 $R[j] = A[q + j]$
- 8 $L[n_1 + 1] = \infty$
- 9 $R[n_2 + 1] = \infty$
- 10 $i = 1$
- 11 $j = 1$
- 12 **for** $k = p$ **to** r
 - 13 **if** $L[i] \leq R[j]$
 - 14 $A[k] = L[i]$
 - 15 $i = i + 1$
 - 16 **else** $A[k] = R[j]$
 - 17 $j = j + 1$

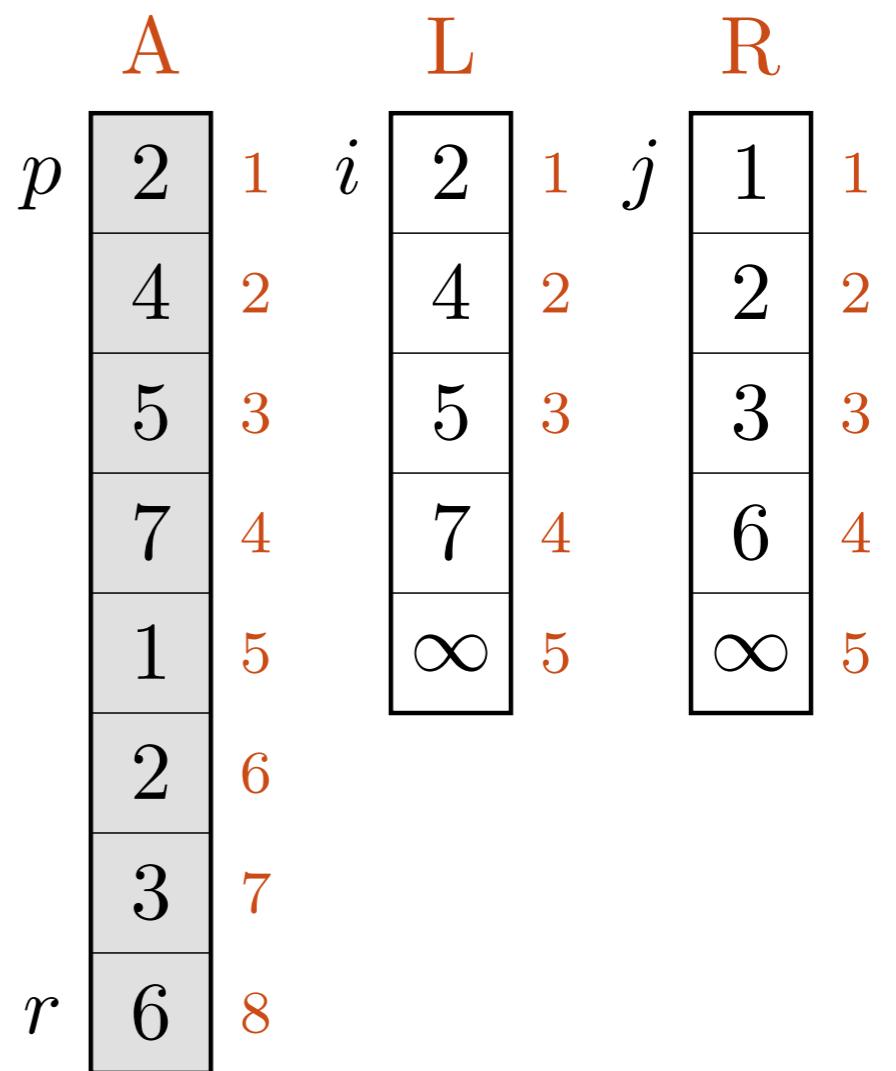
```
MERGE(A, p, q, r)
1  copy into L and R
2  for k = p to r
3      if L[i] ≤ R[j]
4          A[k] = L[i]
5          i = i + 1
6      else A[k] = R[j]
7          j = j + 1
```

Forenklet fremstilling av samme prosedyre

```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

```

	A	L	R
k, p	2 4 5 7 1 2 3 6 ∞	1 2 3 4 5 7 ∞	i 2 4 5 3 4 6 ∞
			j 1 2 3 4 5 4 ∞
r			1 2 3 4 5 6 7 ∞

```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

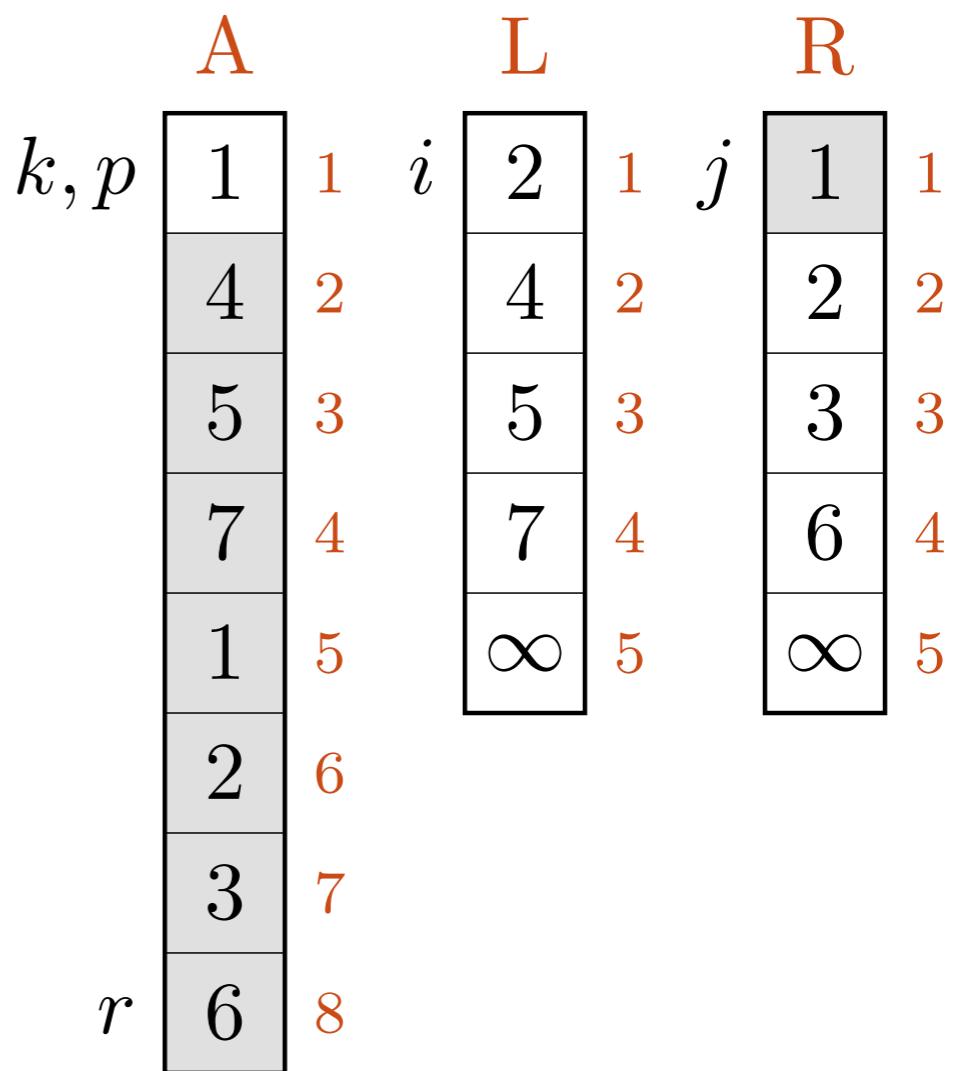
```

	A	L	R
k, p	2 4 5 7 1 2 3 ∞	1 2 3 4 5 7 4 ∞	i j 1 2 3 6 ∞
i	2 4 5 7 1 2 3 ∞	1 2 3 4 5 7 4 ∞	j 1 2 3 6 ∞
j	1 2 3 6 ∞	2 4 5 2 3 3 4 5	1 2 3 4 5 4 6 5

```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

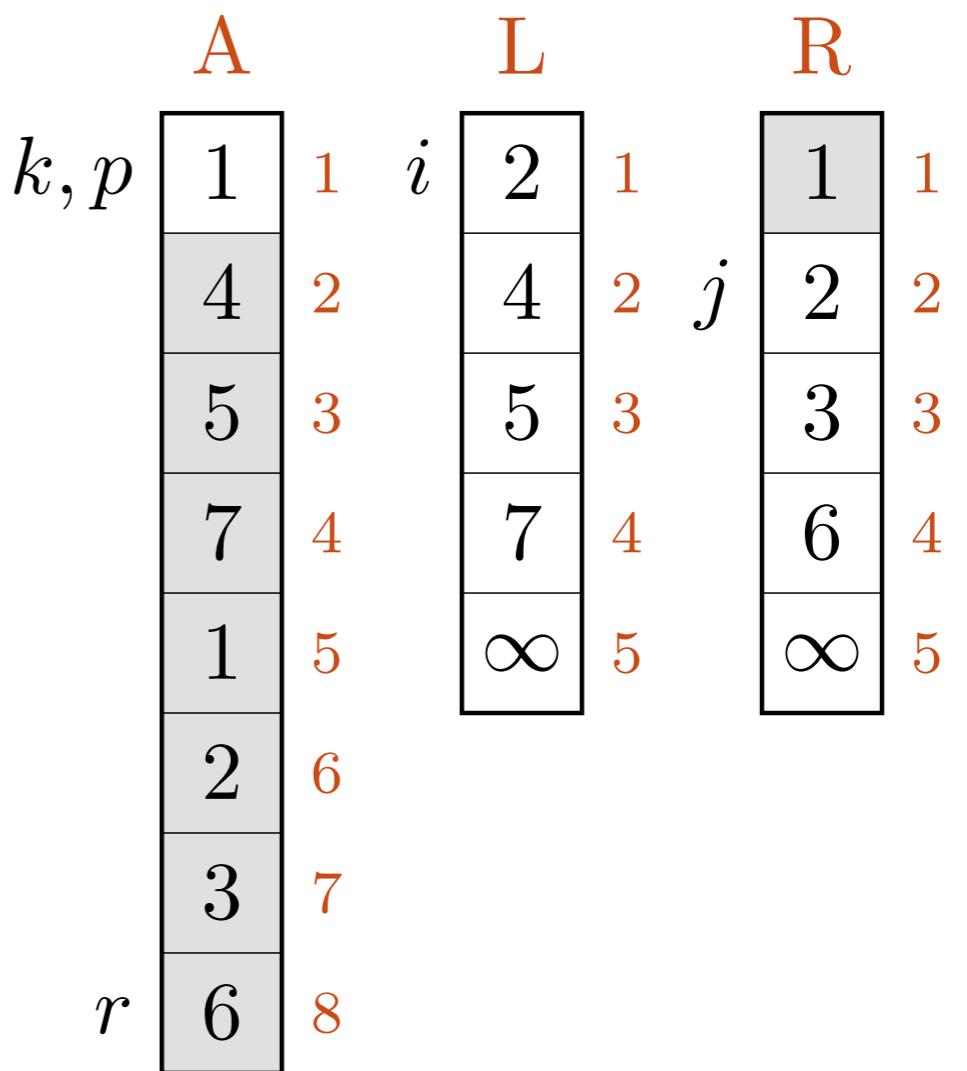
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

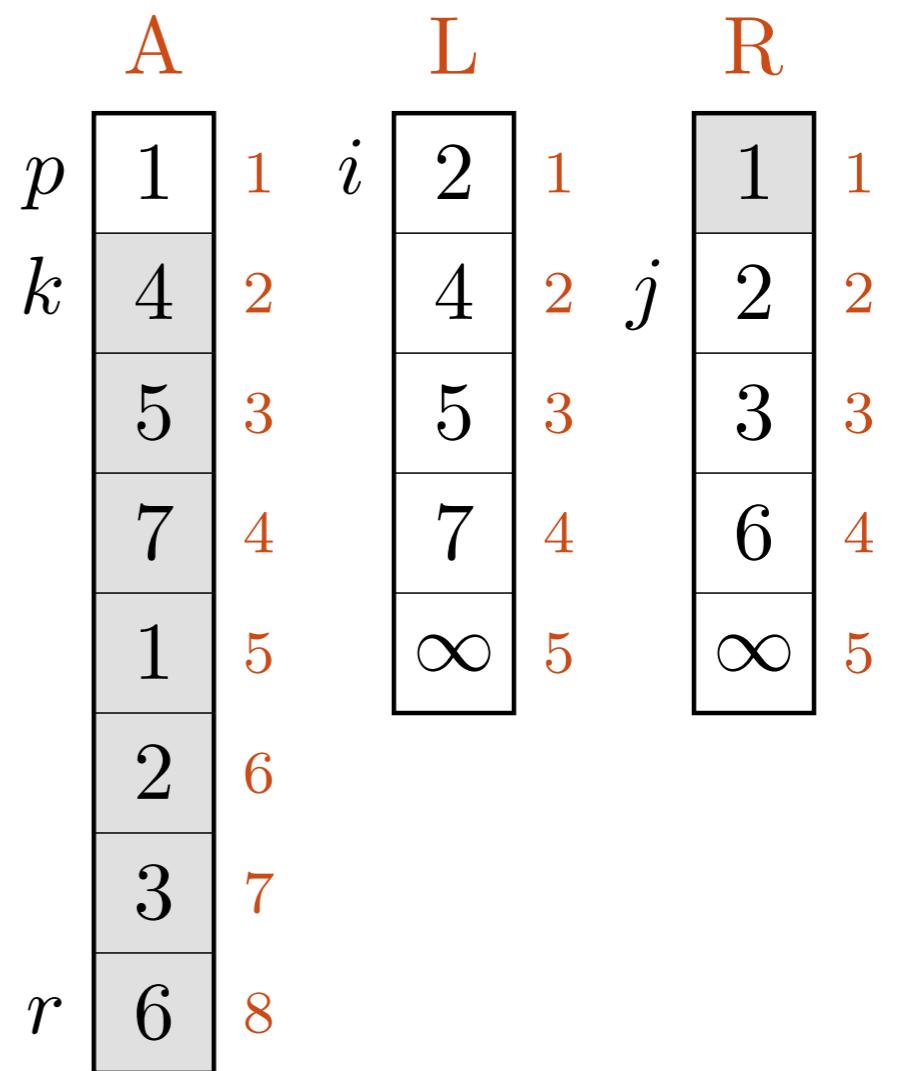
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

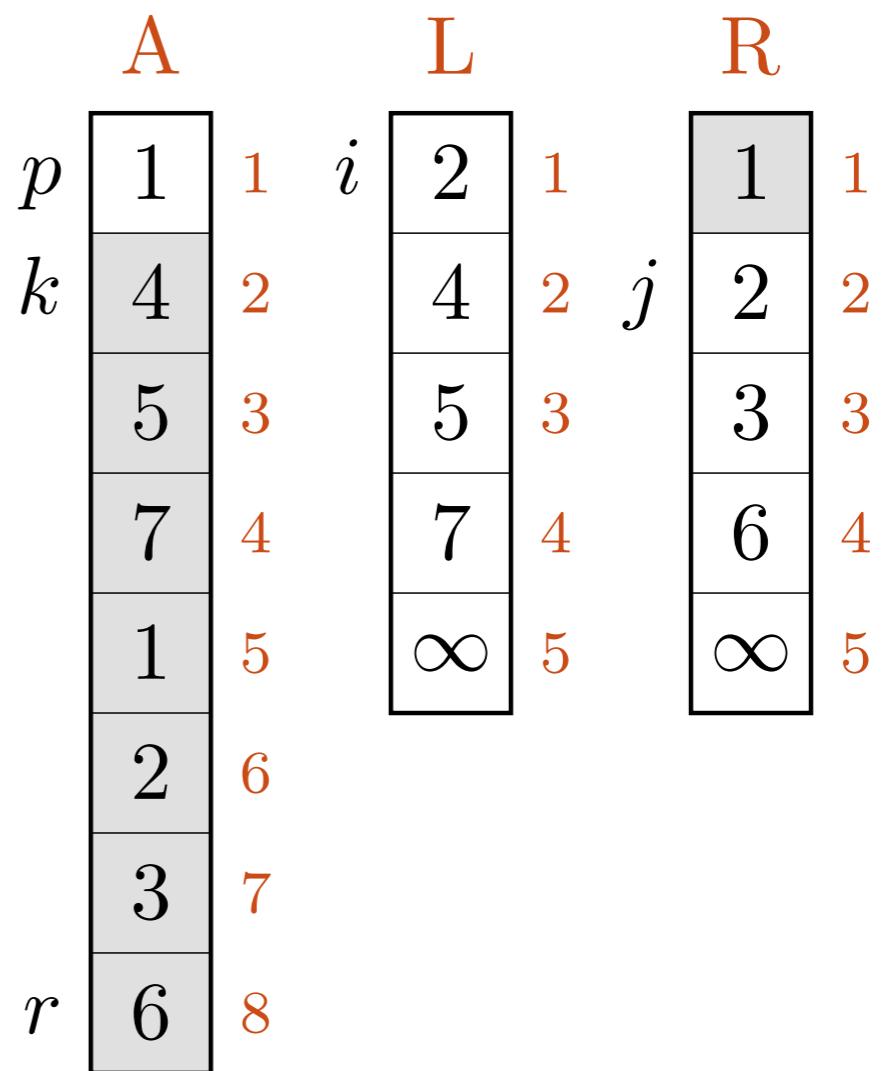
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

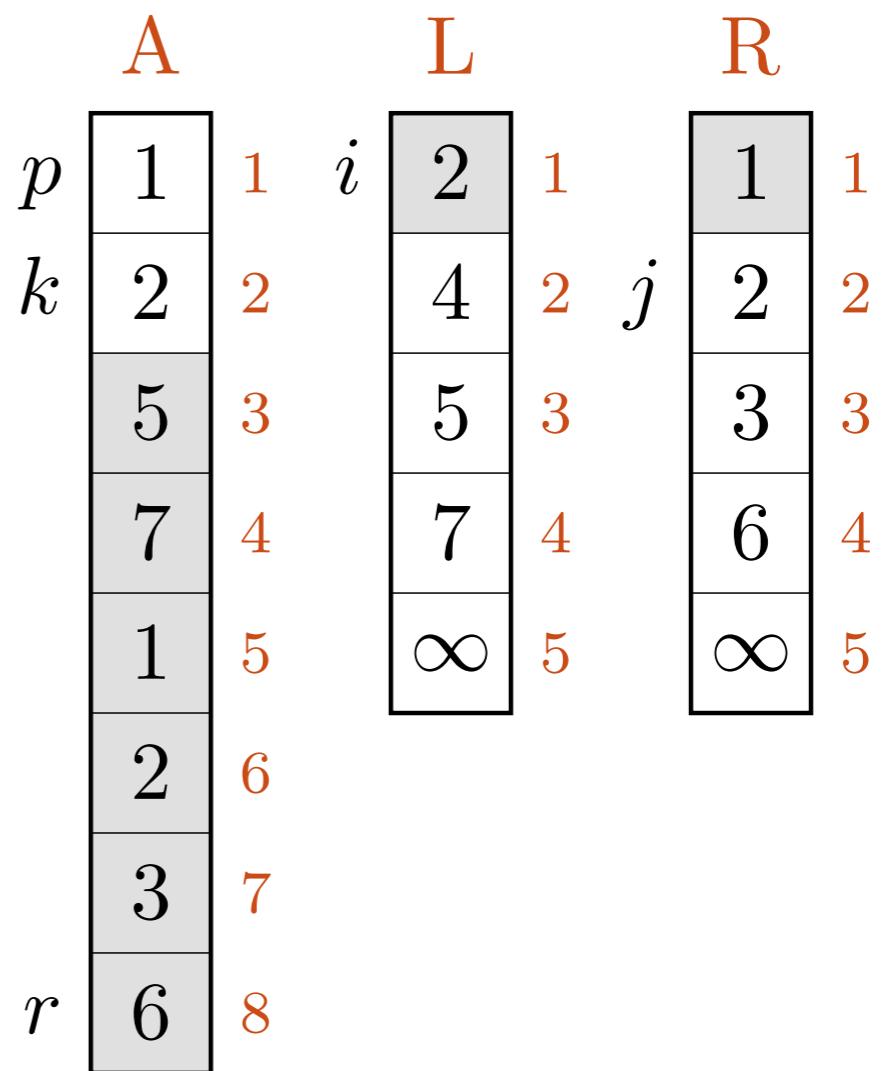
```



```

MERGE(A, p, q, r)
1  copy into L and R
2  for  $k = p$  to  $r$ 
3      if  $L[i] \leq R[j]$ 
4           $A[k] = L[i]$ 
5           $i = i + 1$ 
6      else  $A[k] = R[j]$ 
7           $j = j + 1$ 

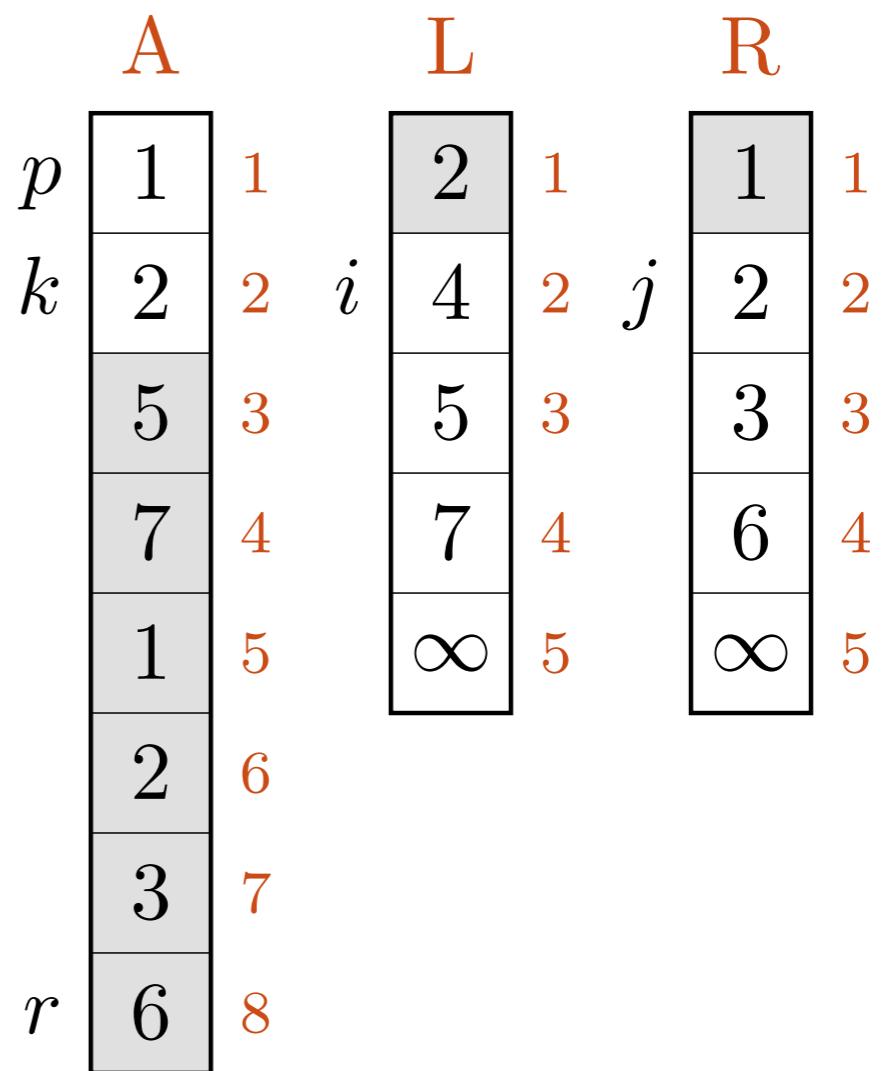
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

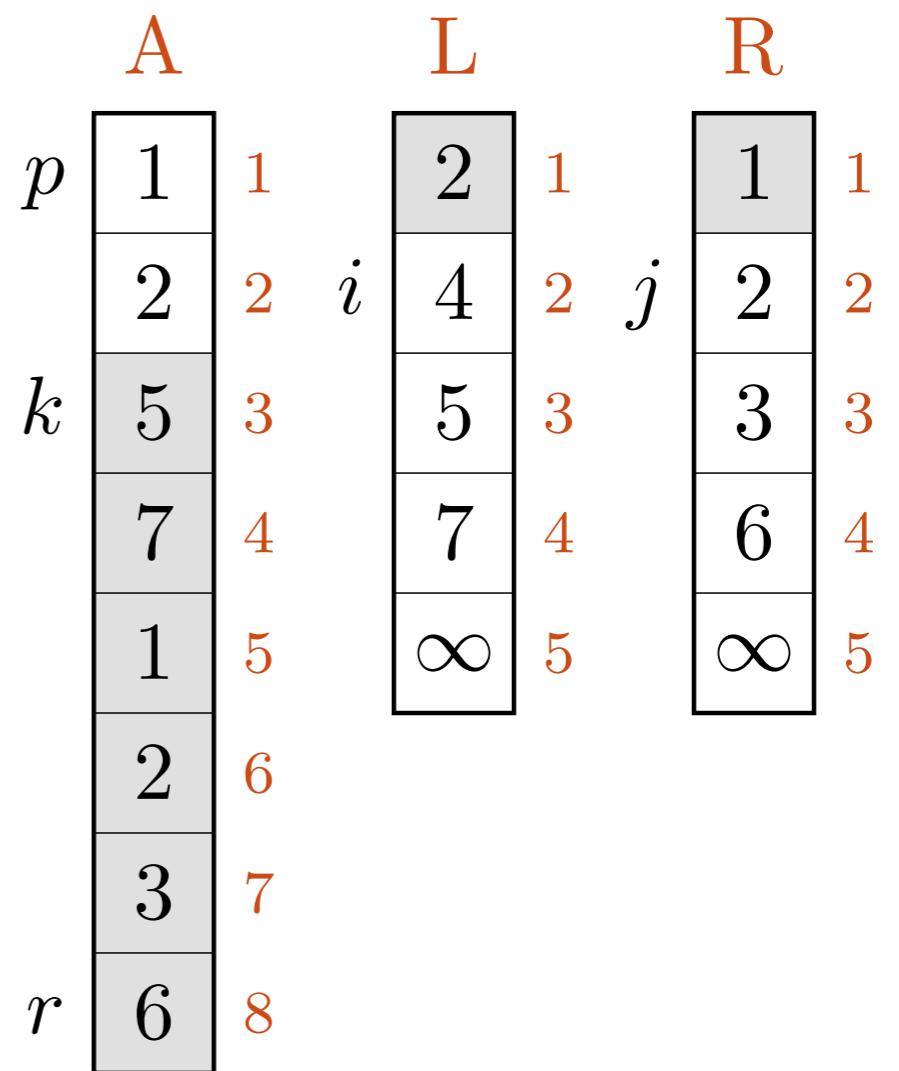
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

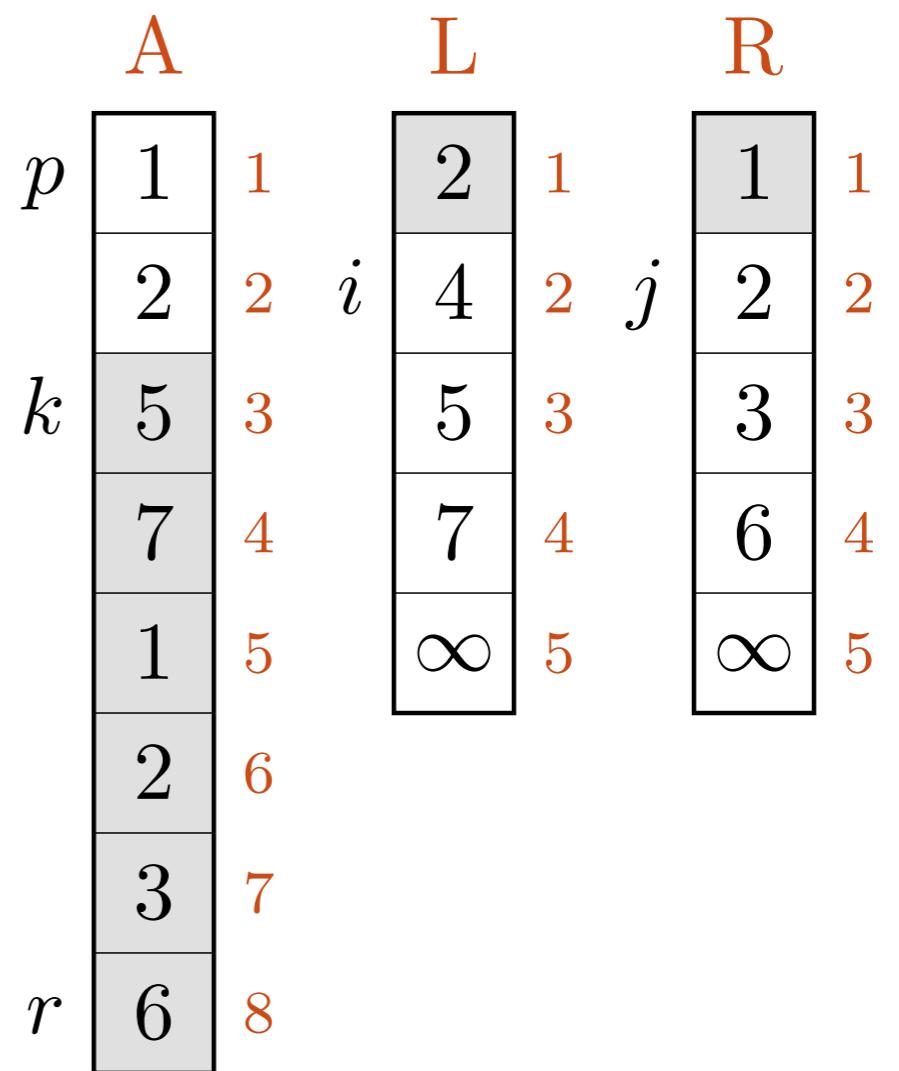
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

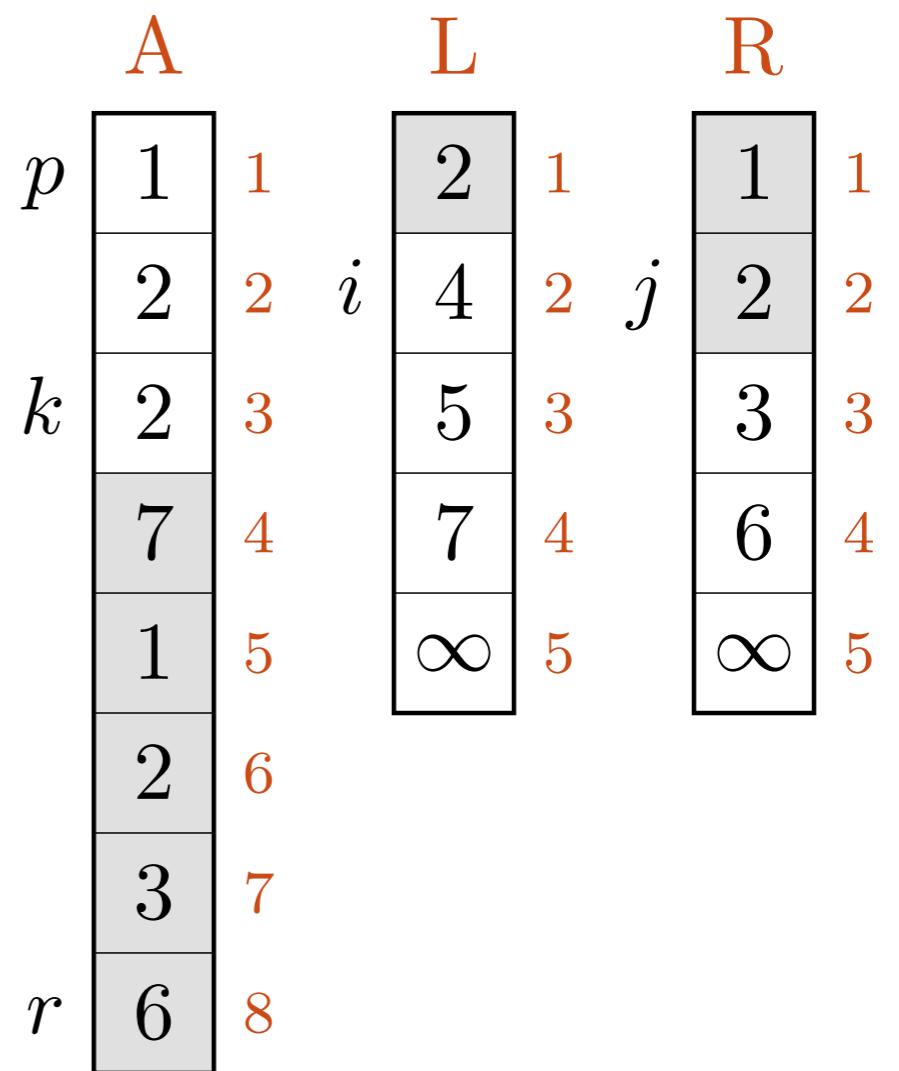
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

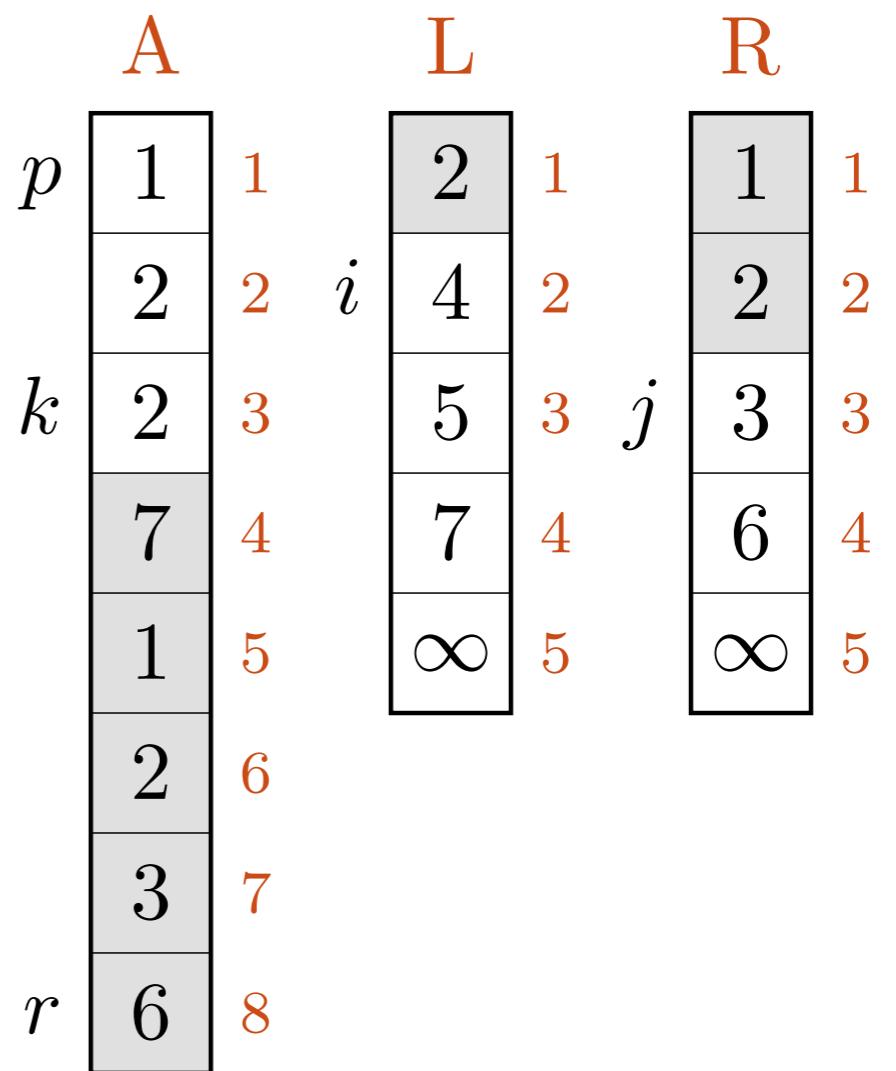
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

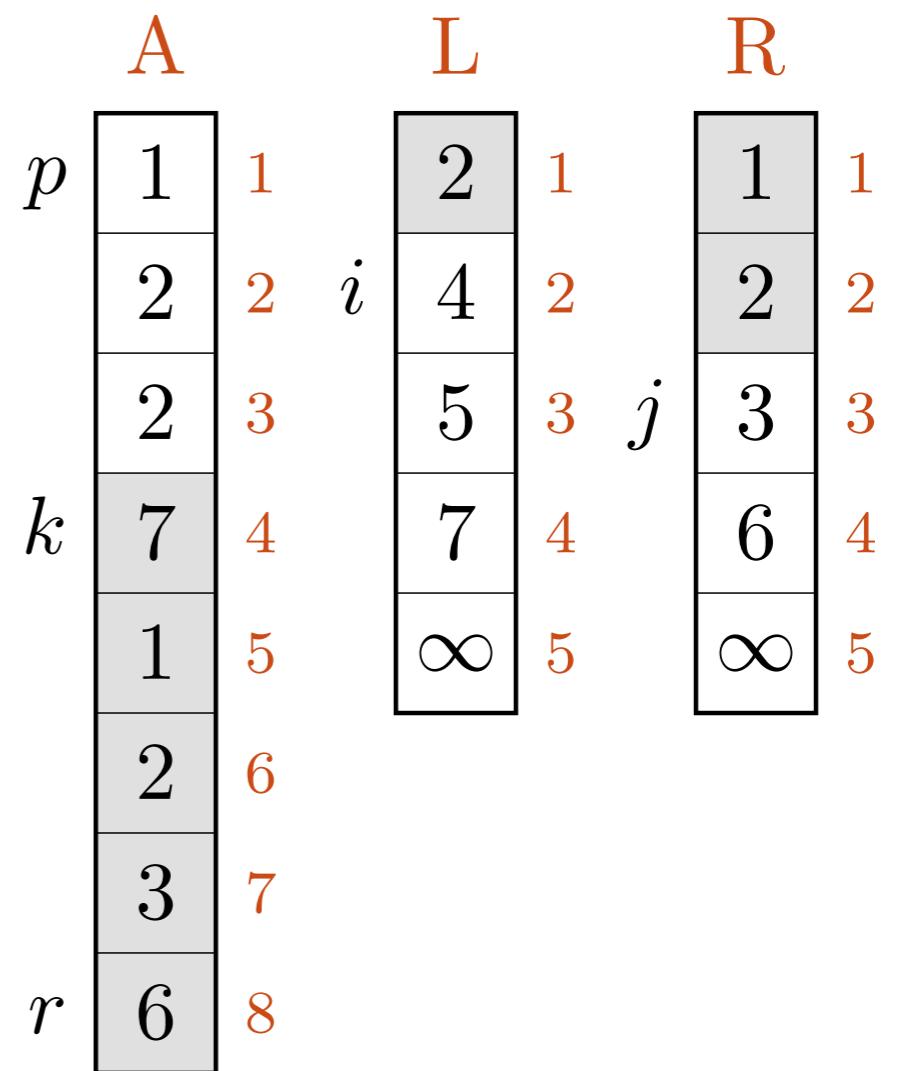
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

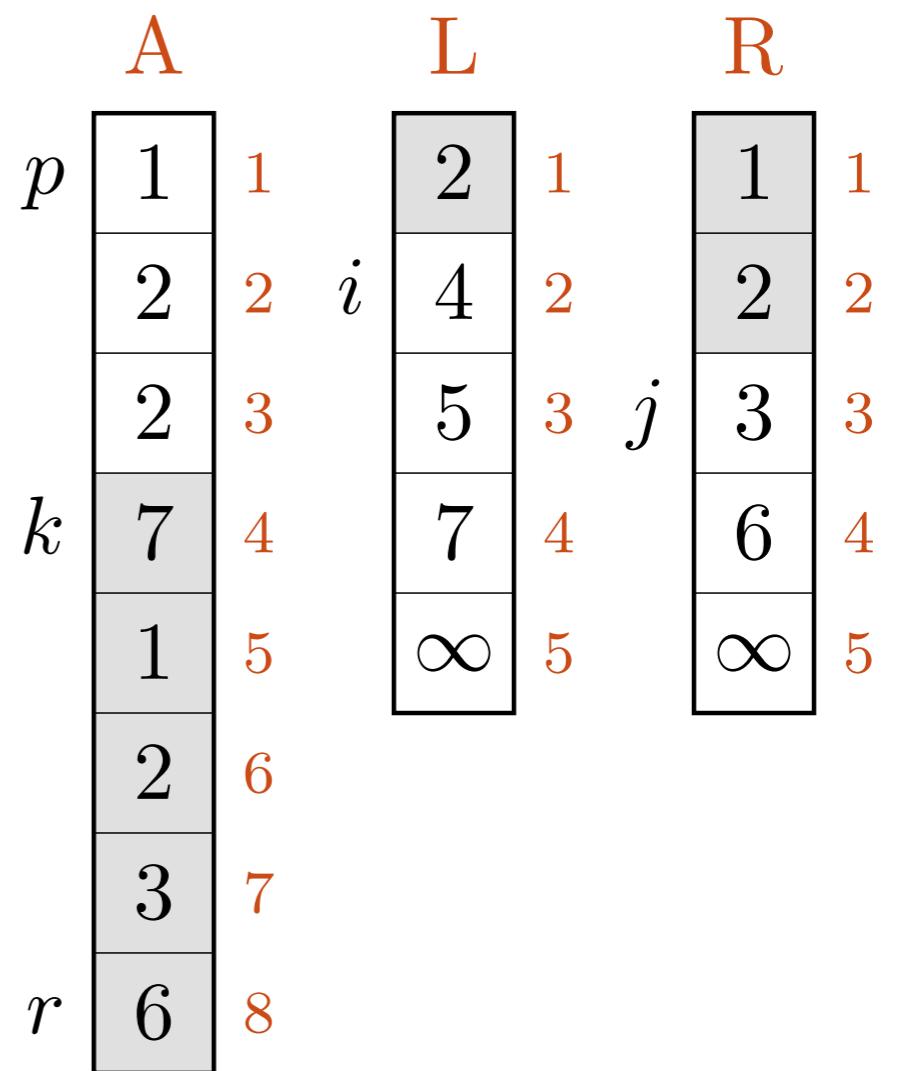
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

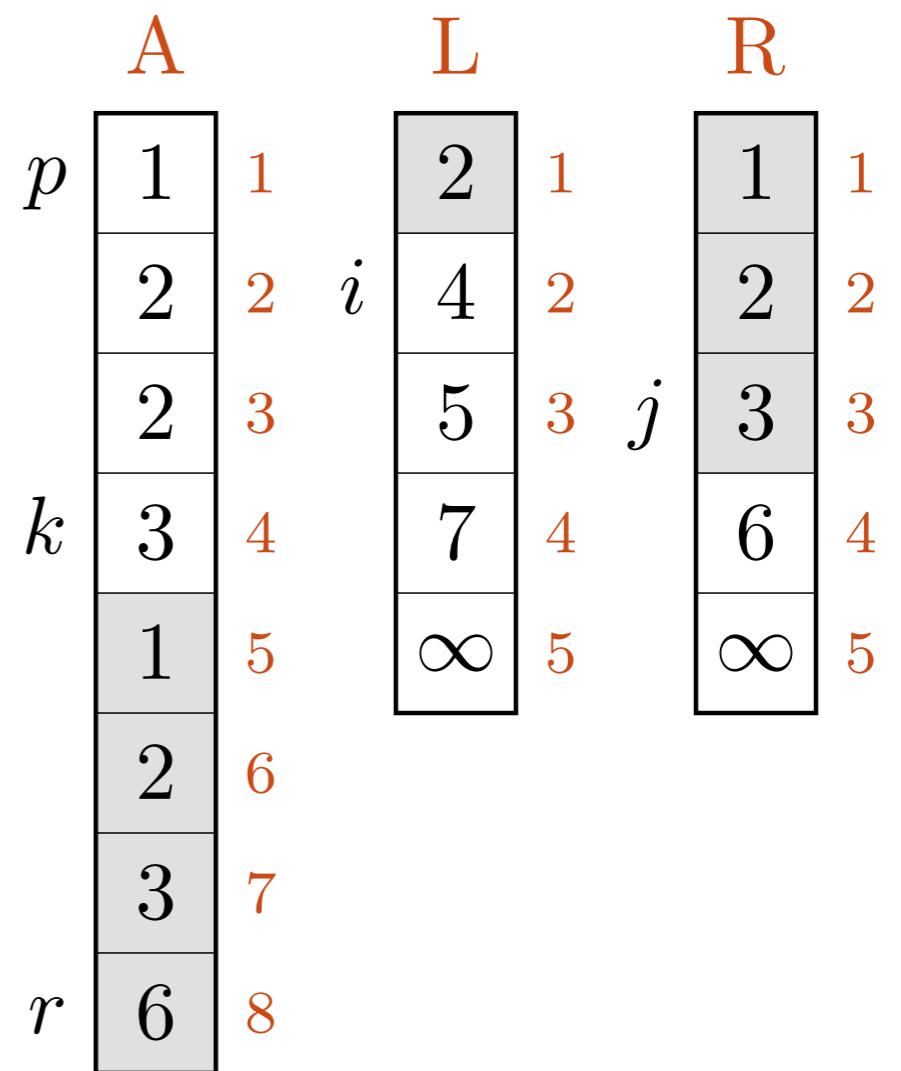
```



```

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

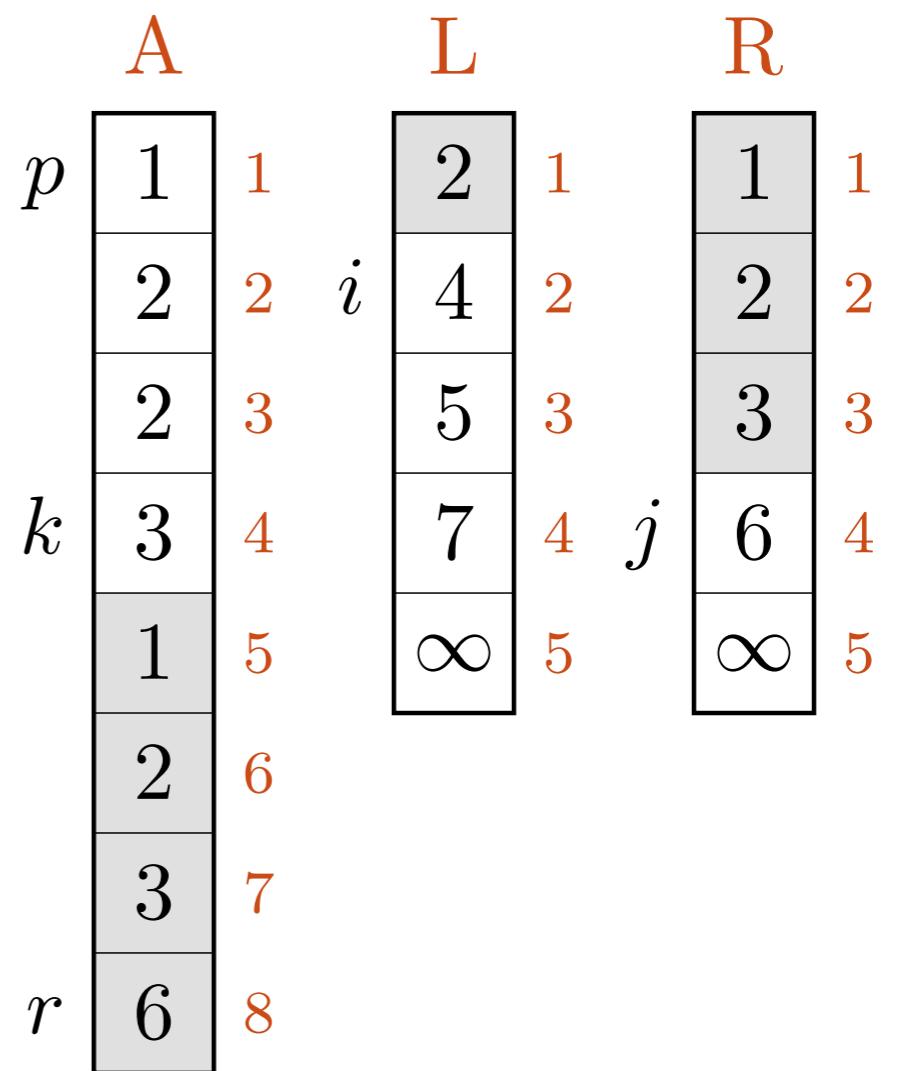
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

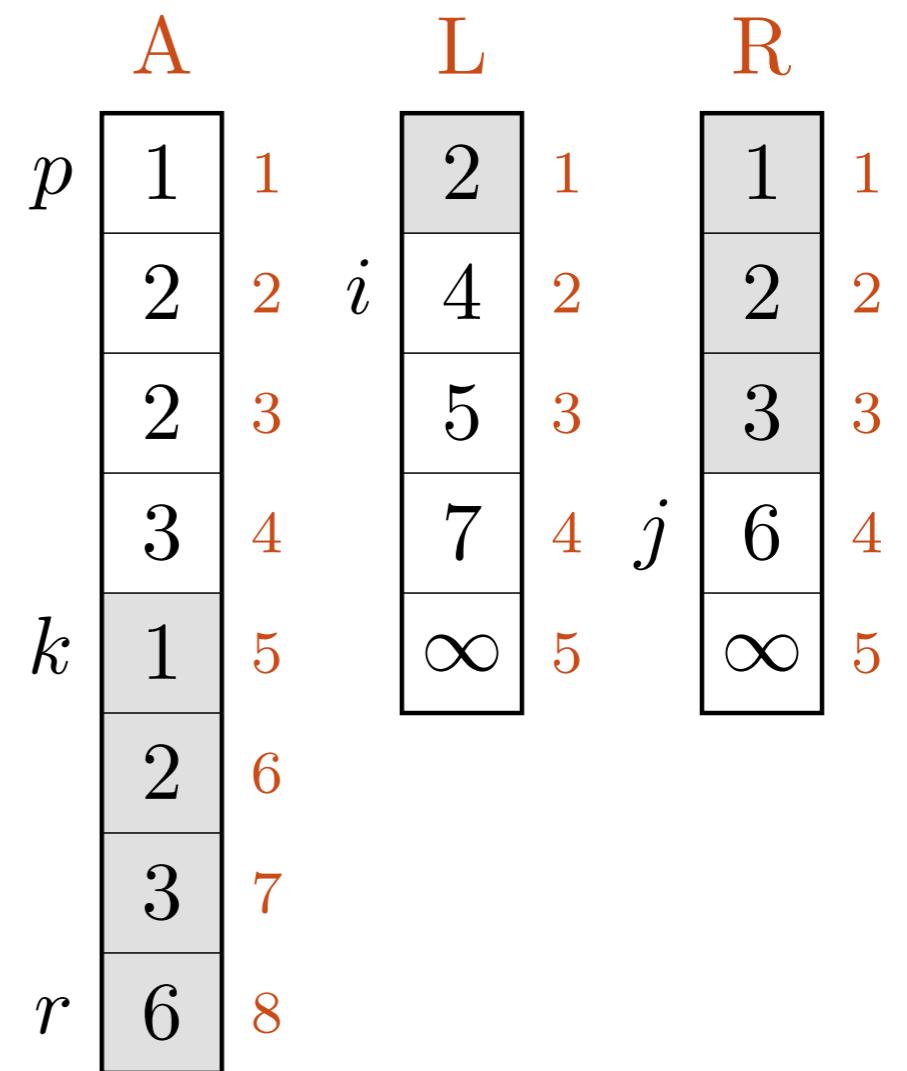
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

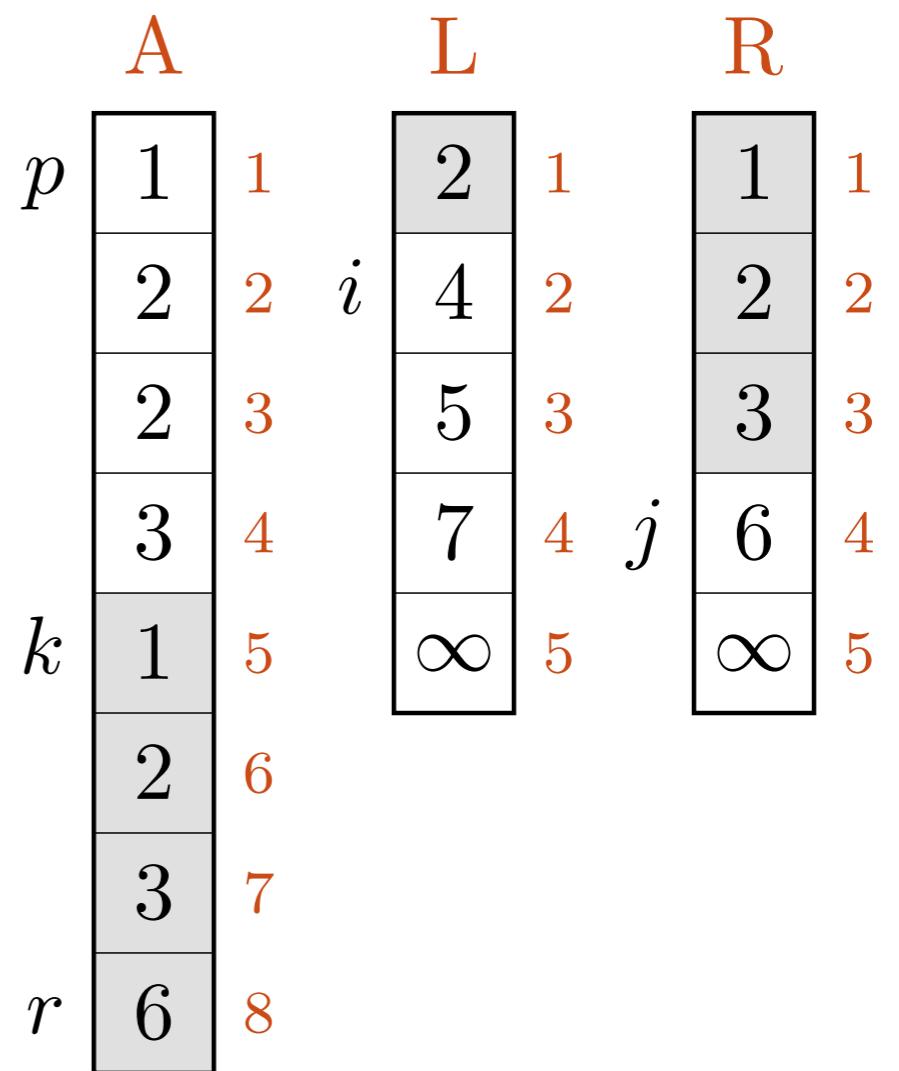
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

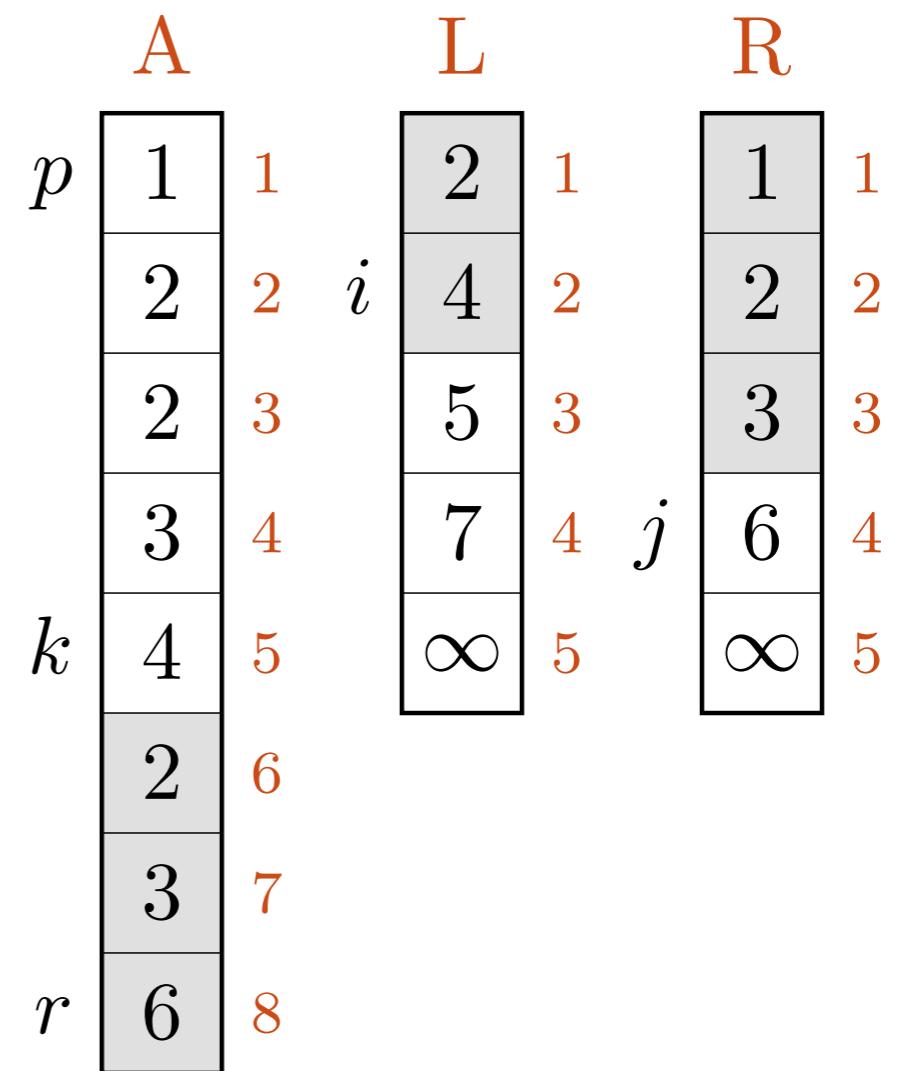
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

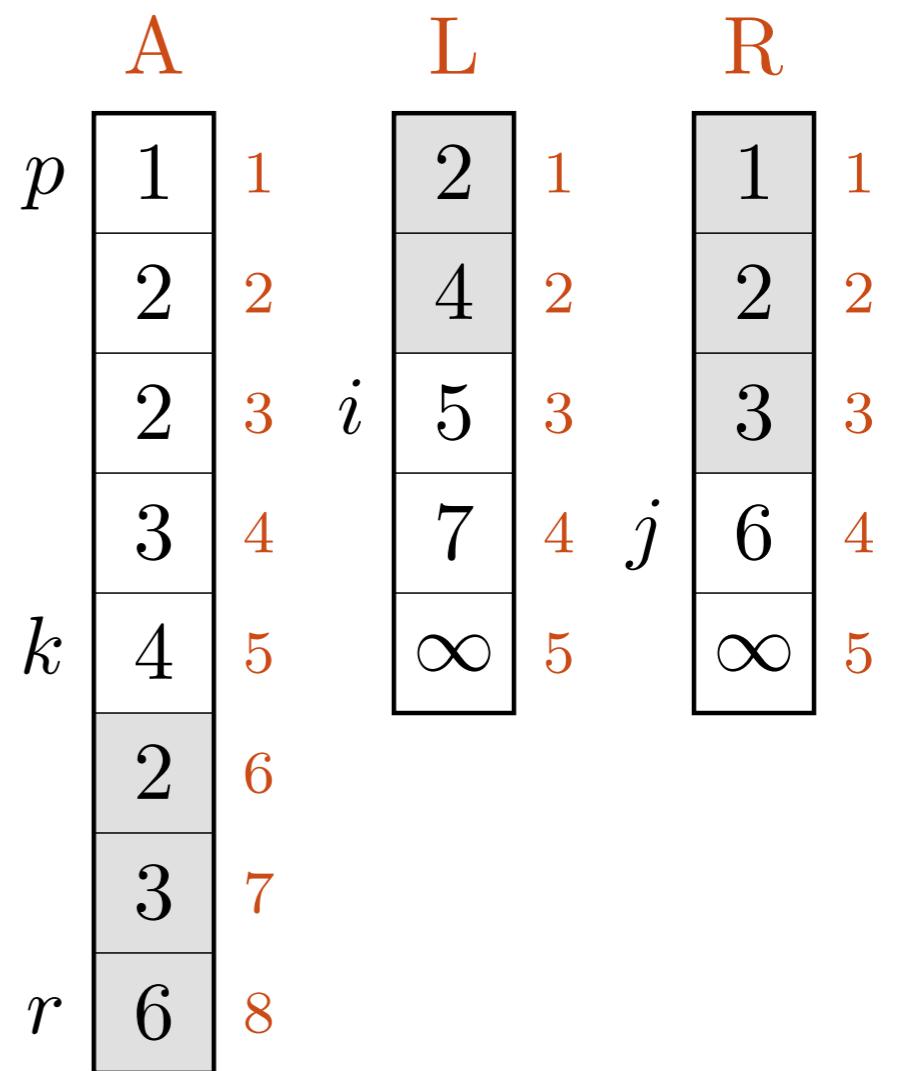
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

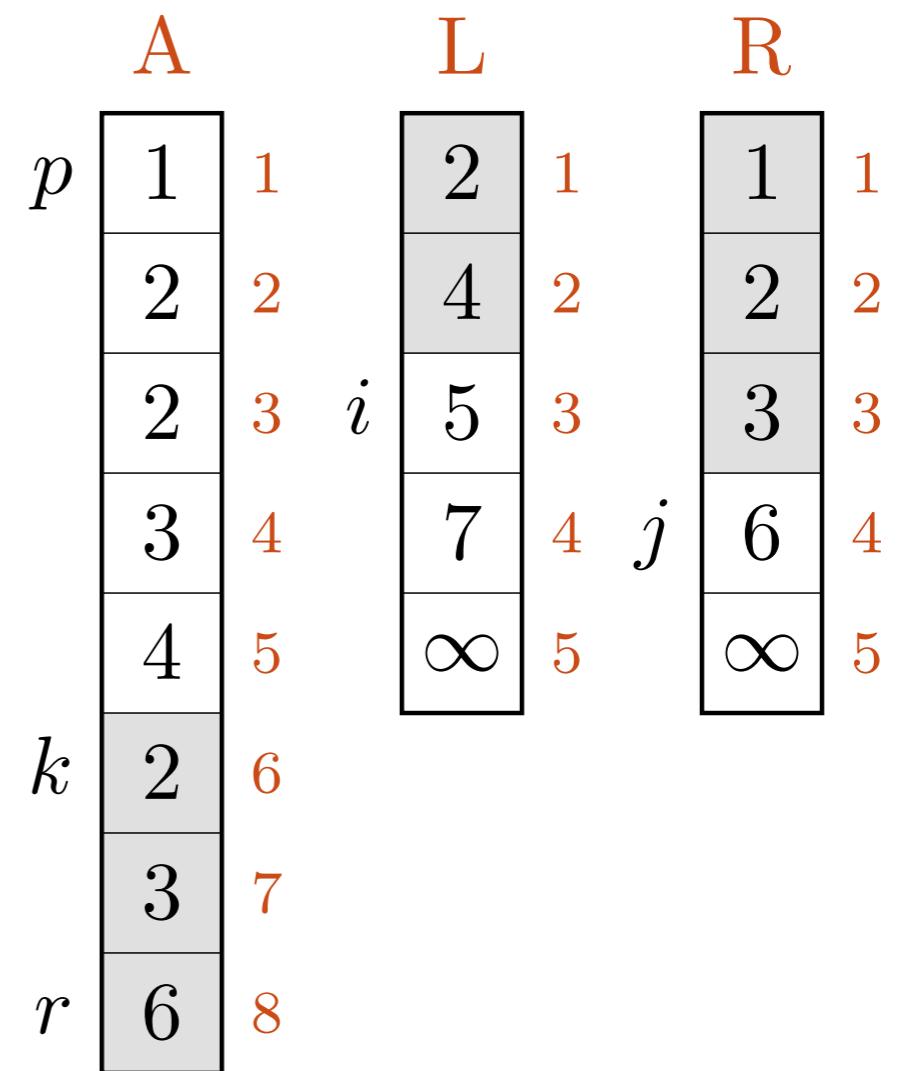
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

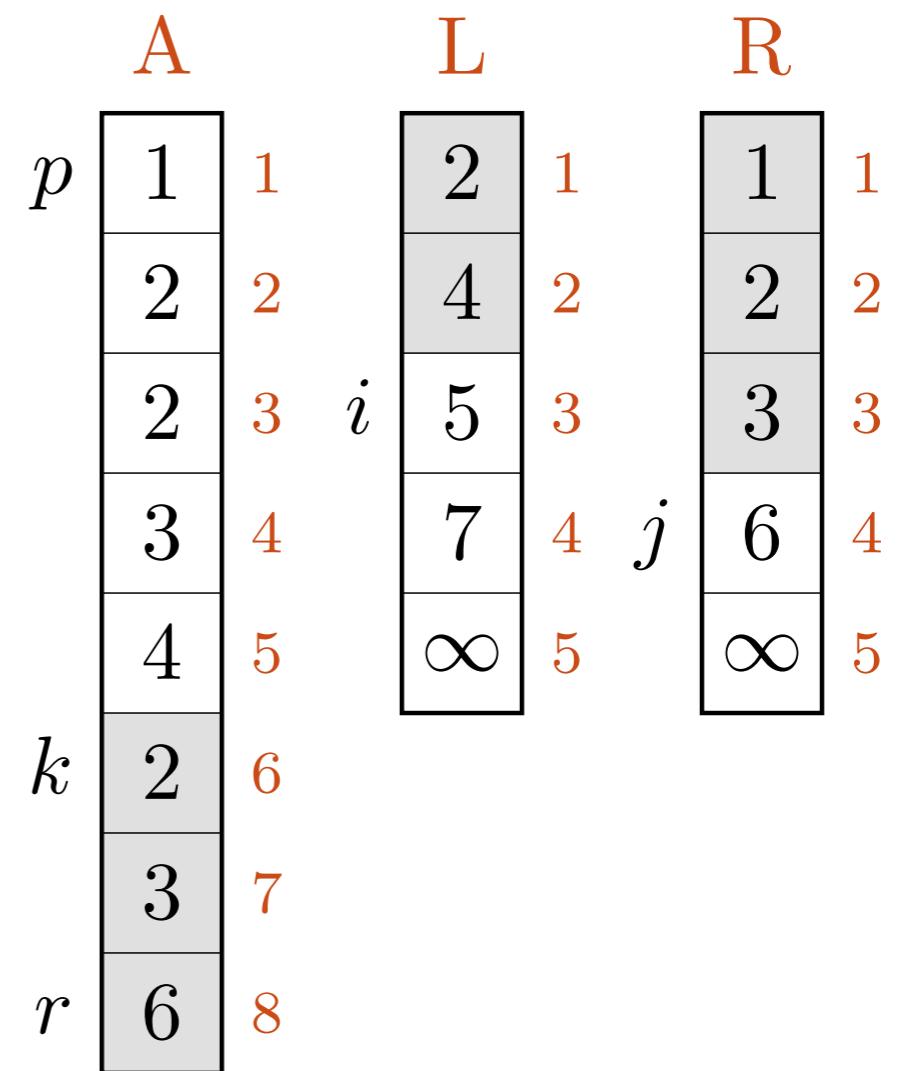
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

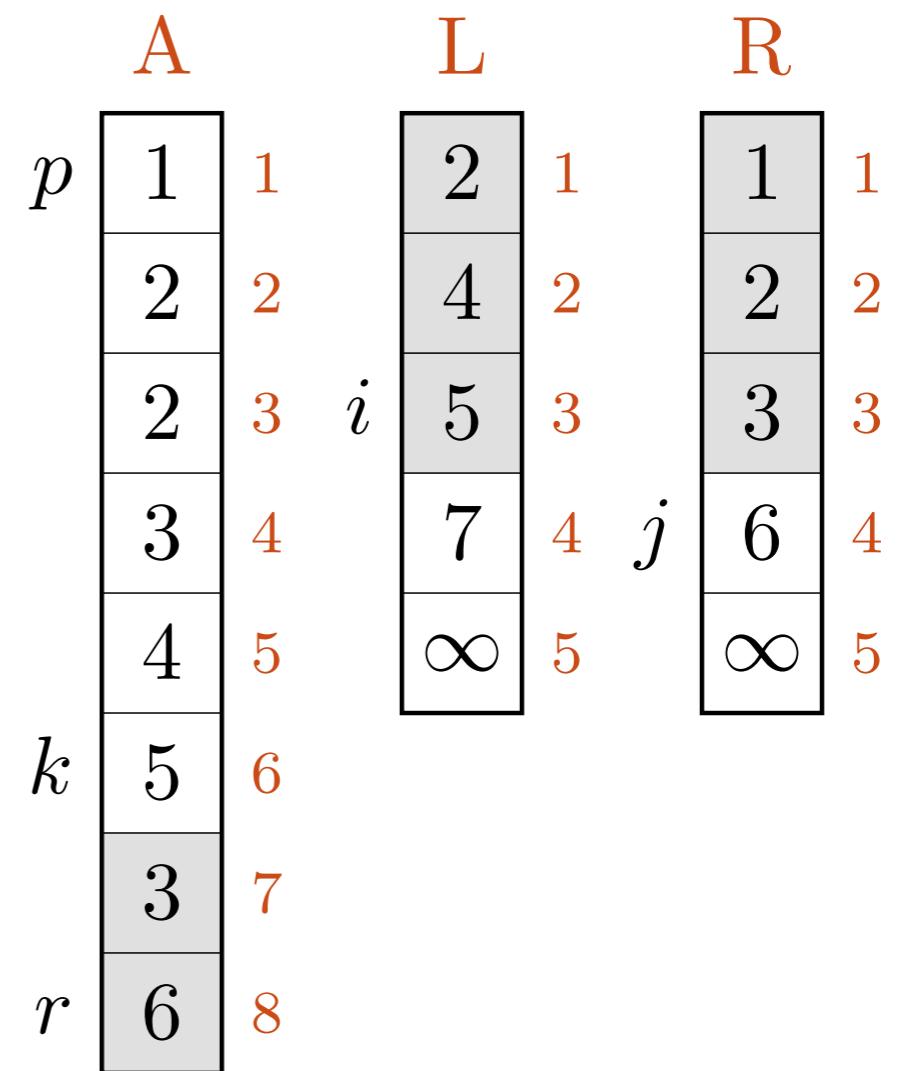
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

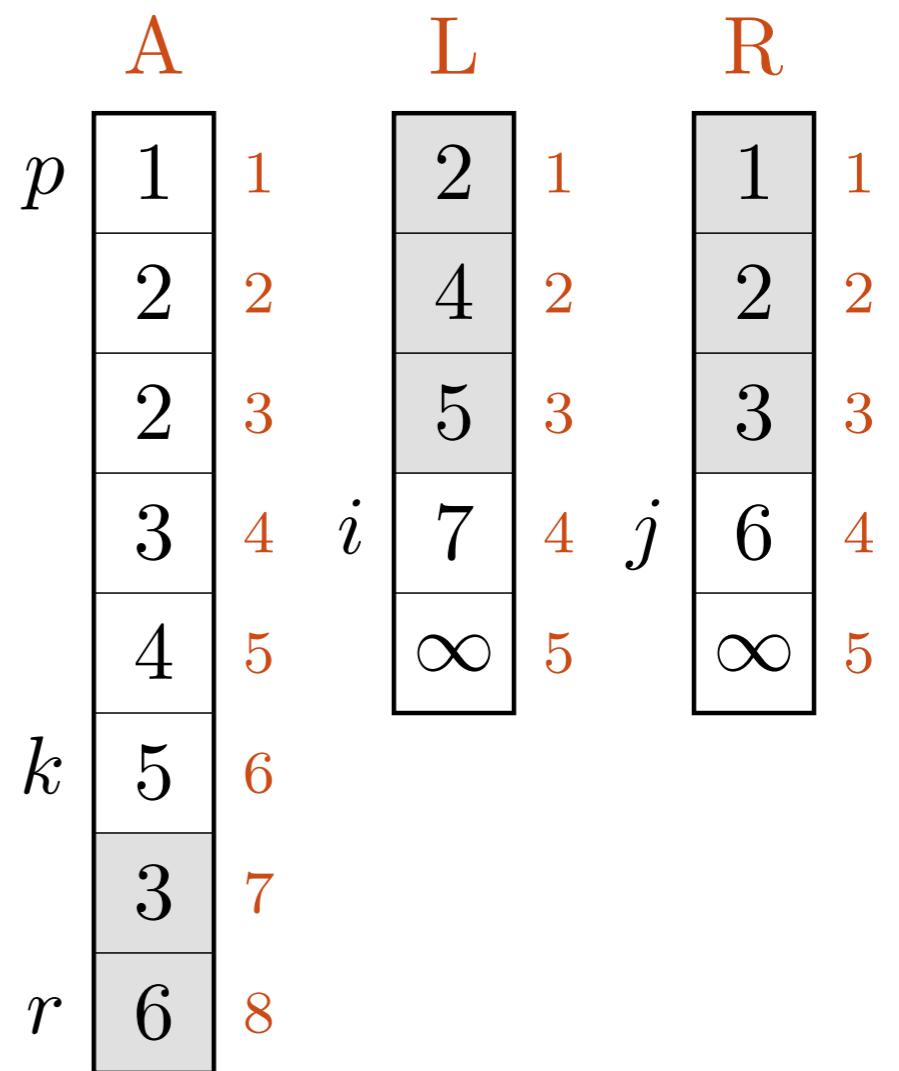
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

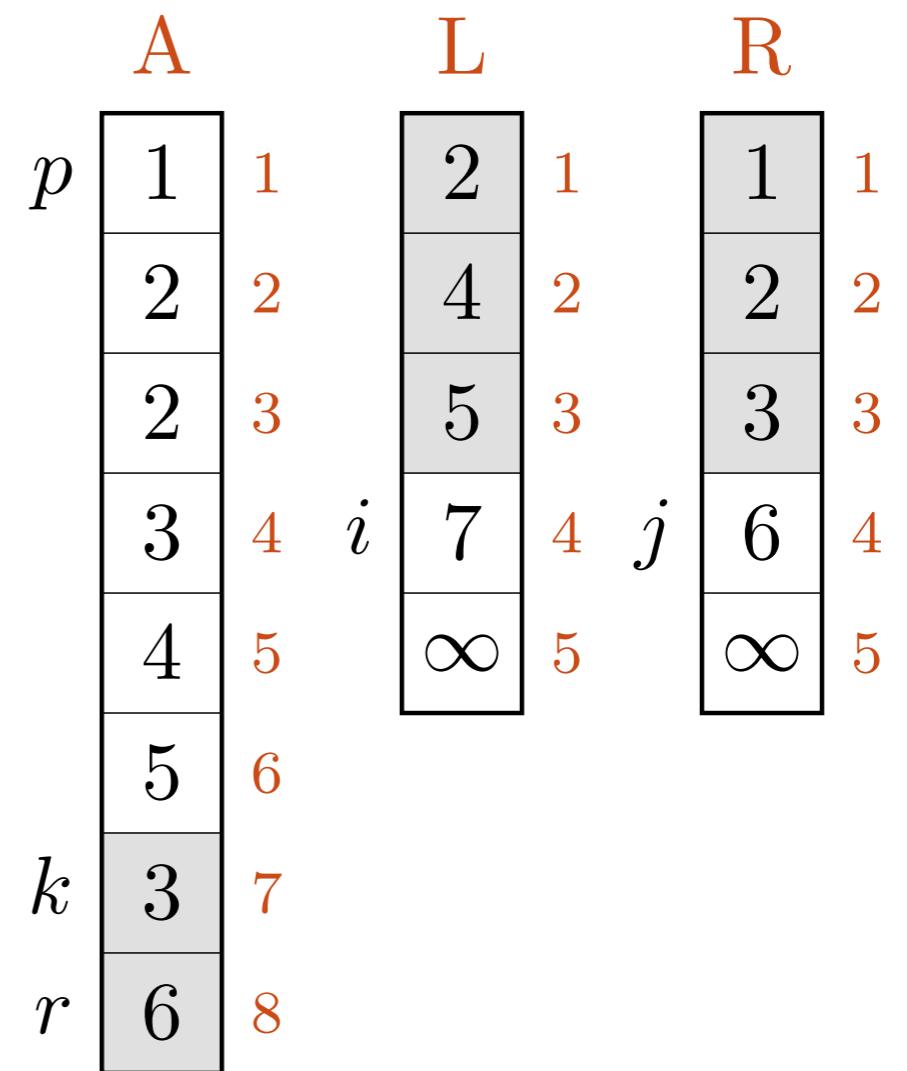
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
3   if L[i] ≤ R[j]
4     A[k] = L[i]
5     i = i + 1
6   else A[k] = R[j]
7     j = j + 1

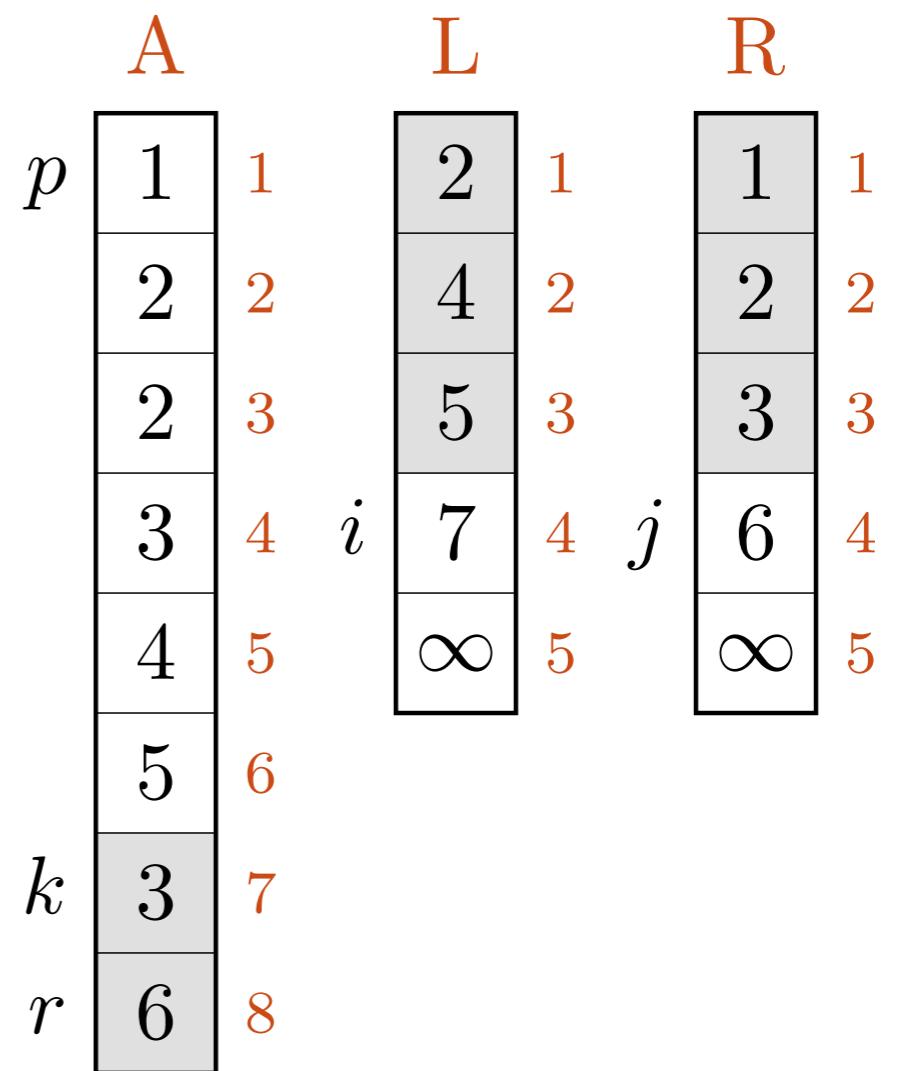
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

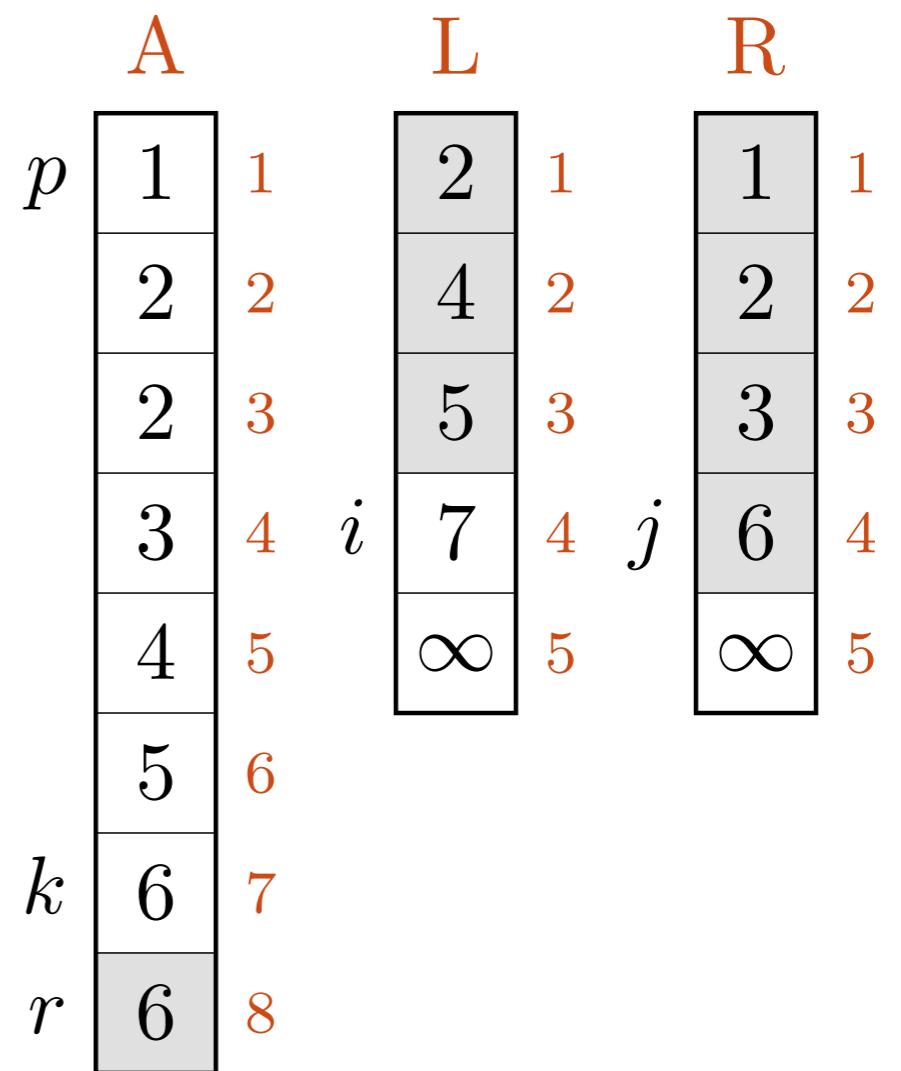
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

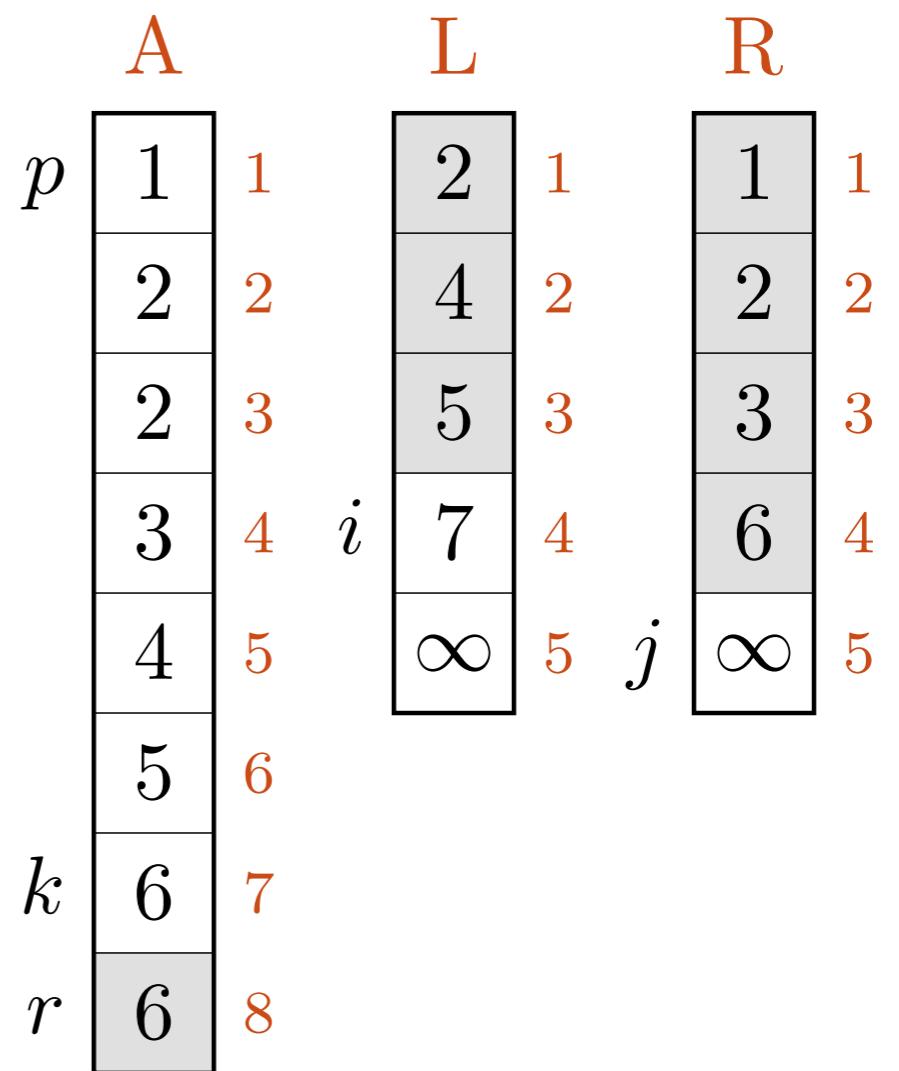
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

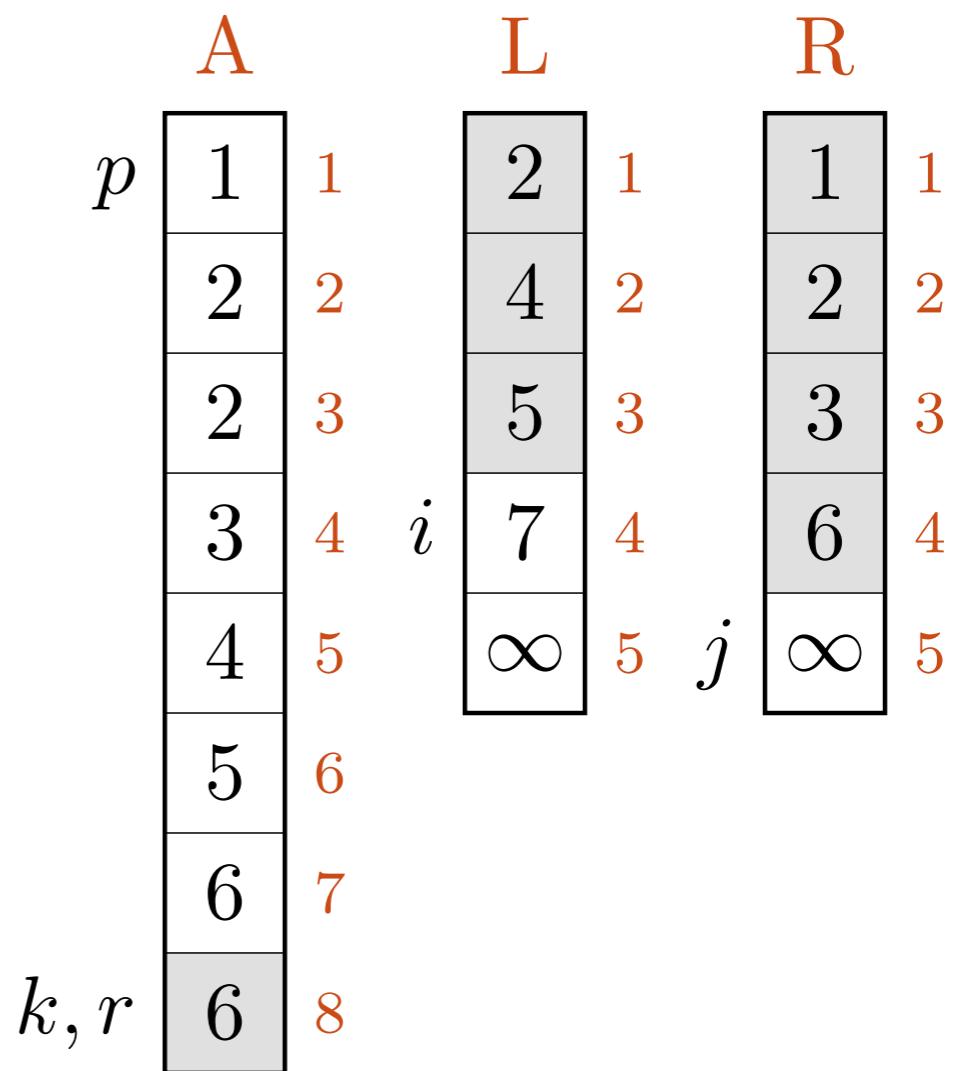
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

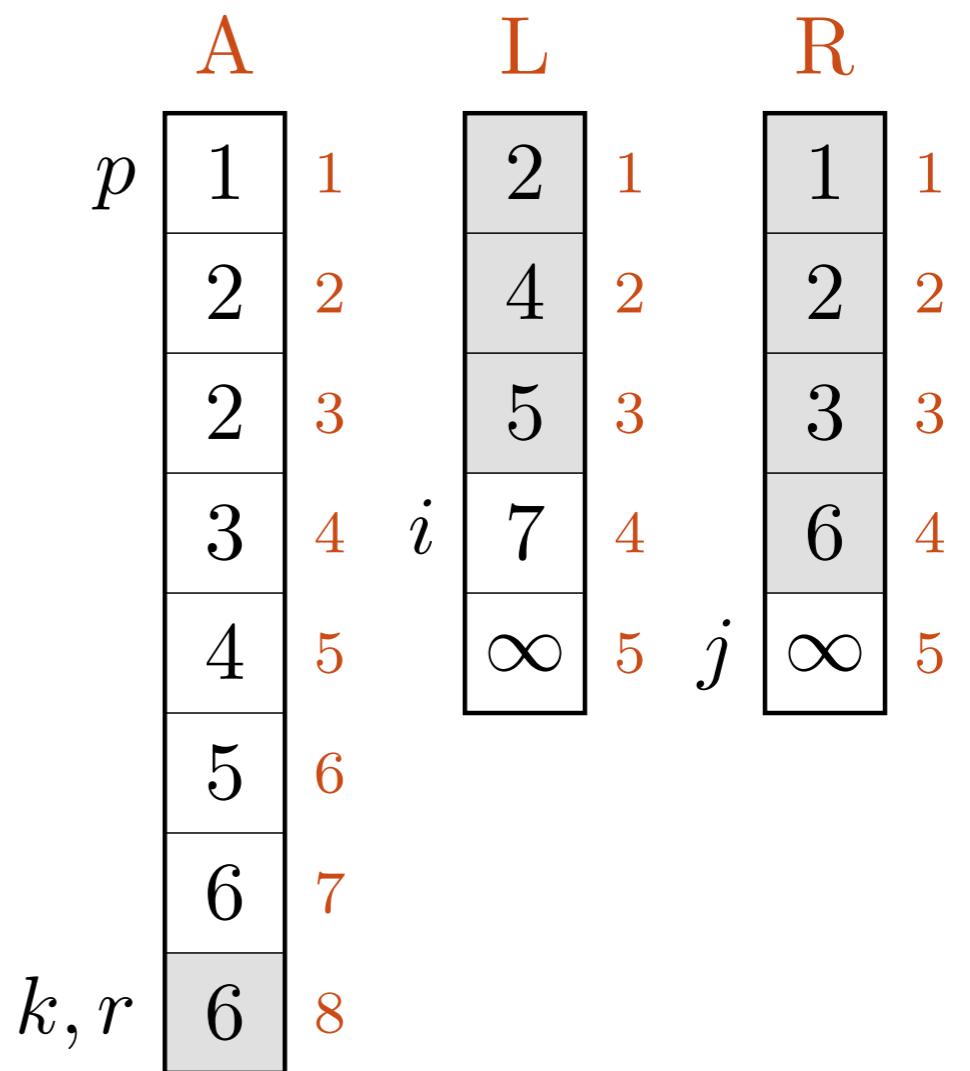
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

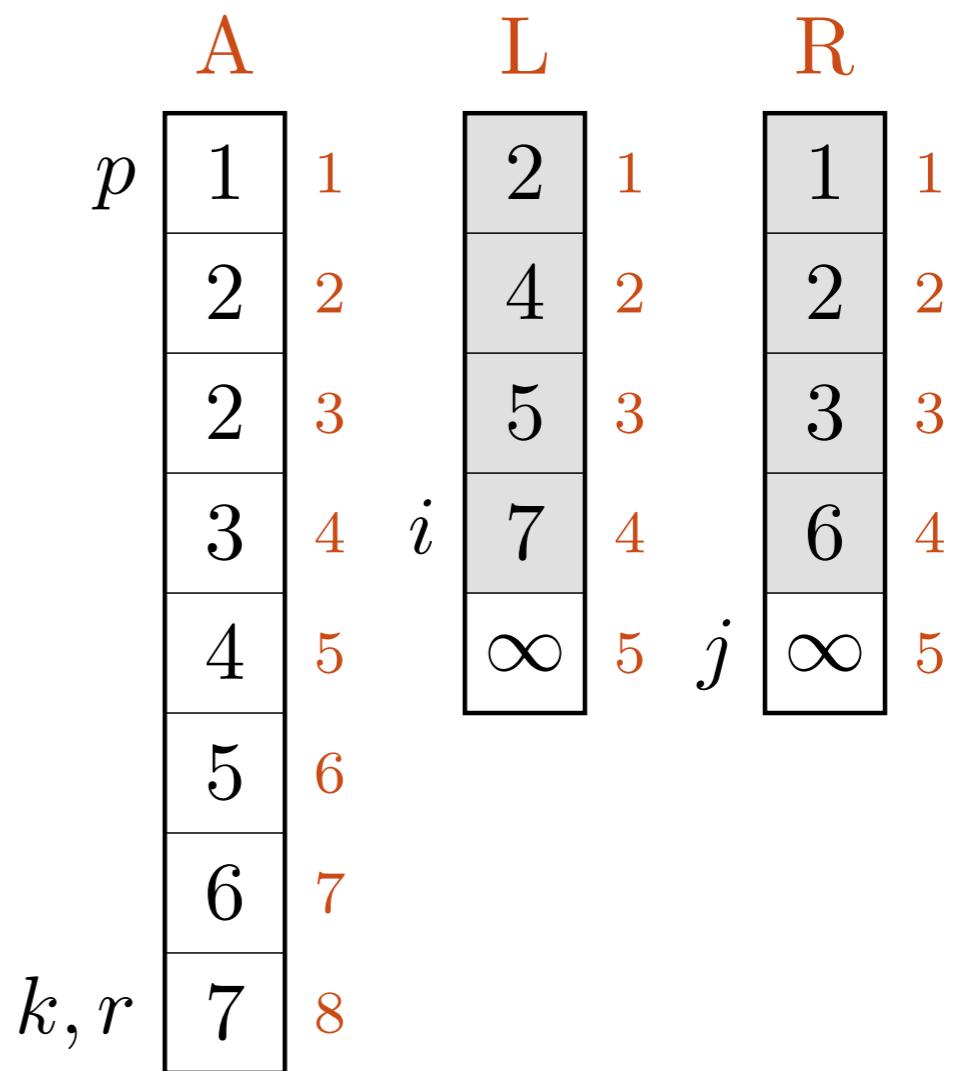
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

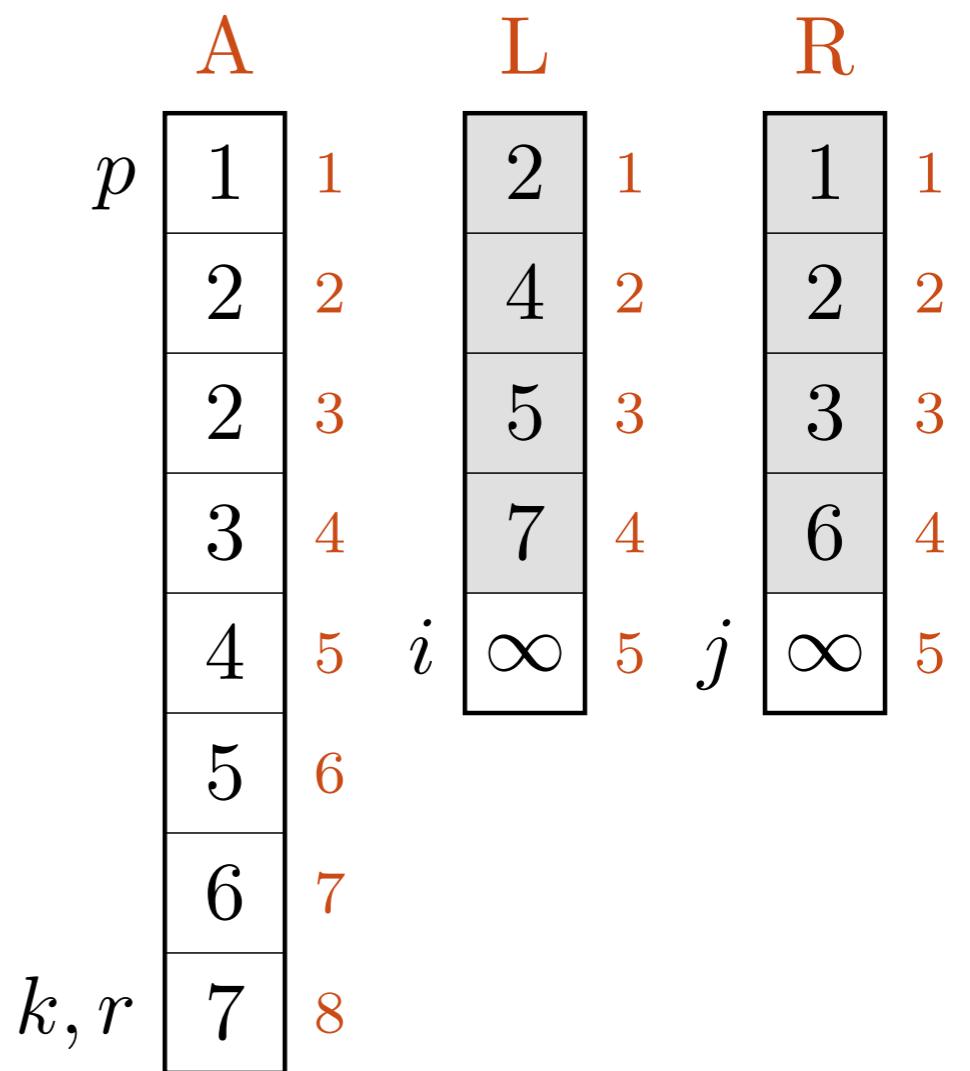
```



```

MERGE(A, p, q, r)
1 copy into L and R
2 for  $k = p$  to  $r$ 
3   if  $L[i] \leq R[j]$ 
4      $A[k] = L[i]$ 
5      $i = i + 1$ 
6   else  $A[k] = R[j]$ 
7      $j = j + 1$ 

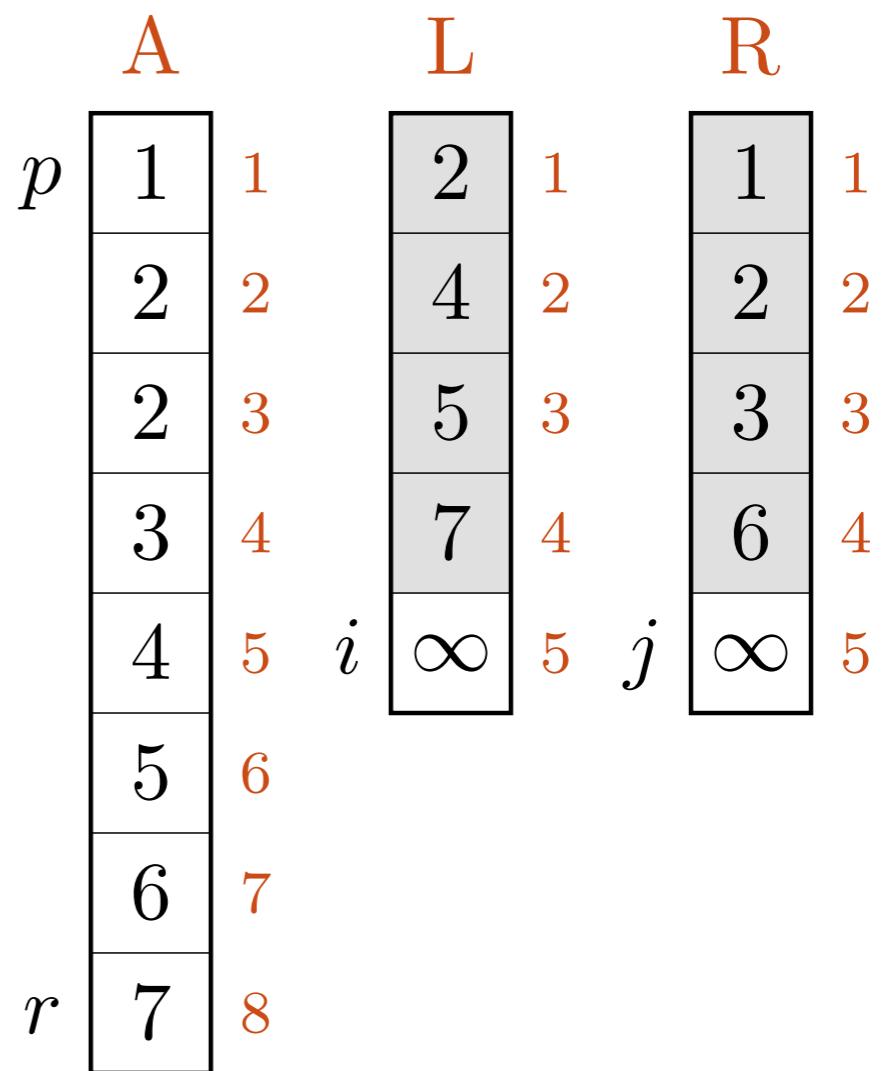
```



```

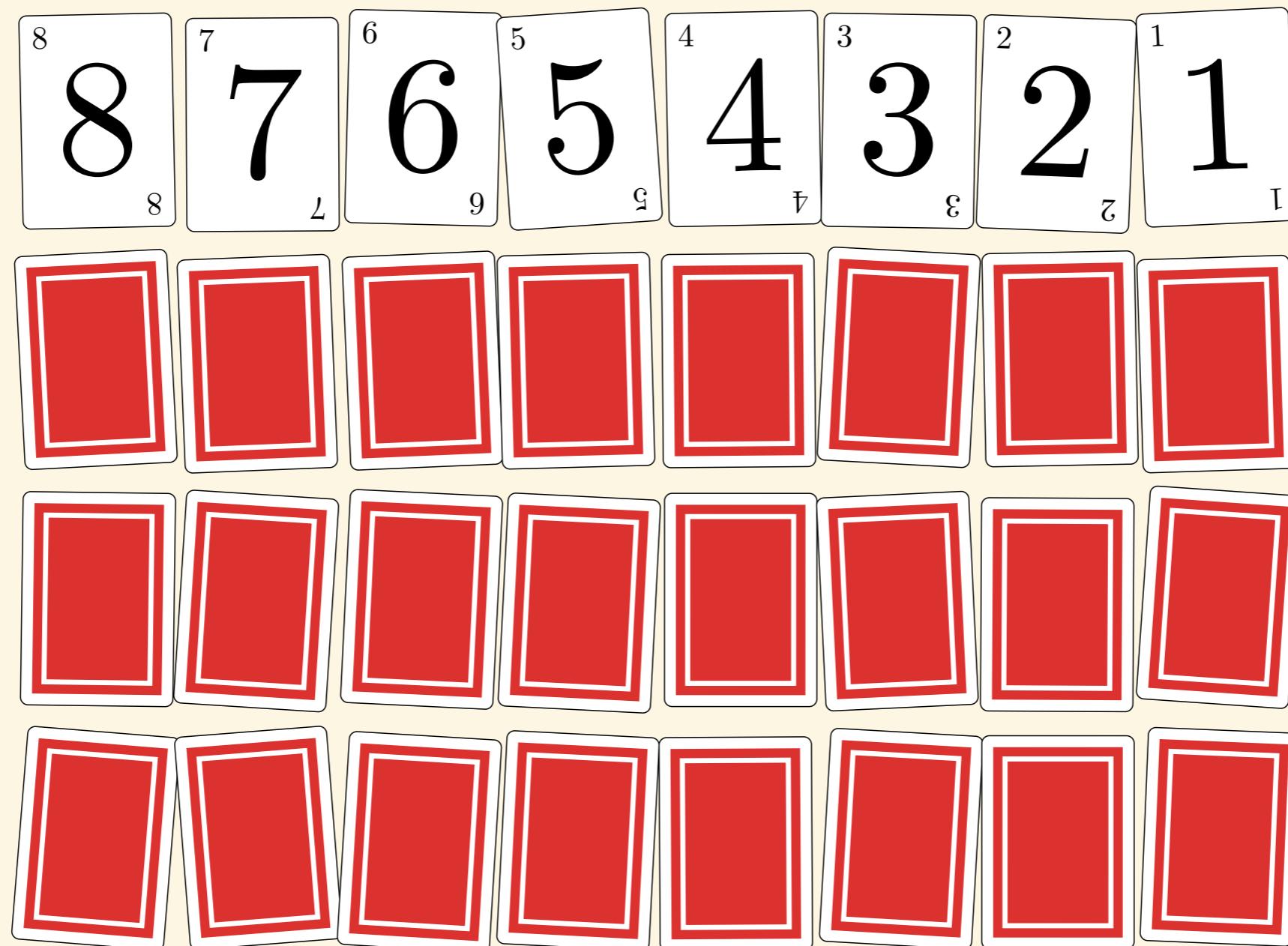
MERGE(A, p, q, r)
1 copy into L and R
2 for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
    j = j + 1

```

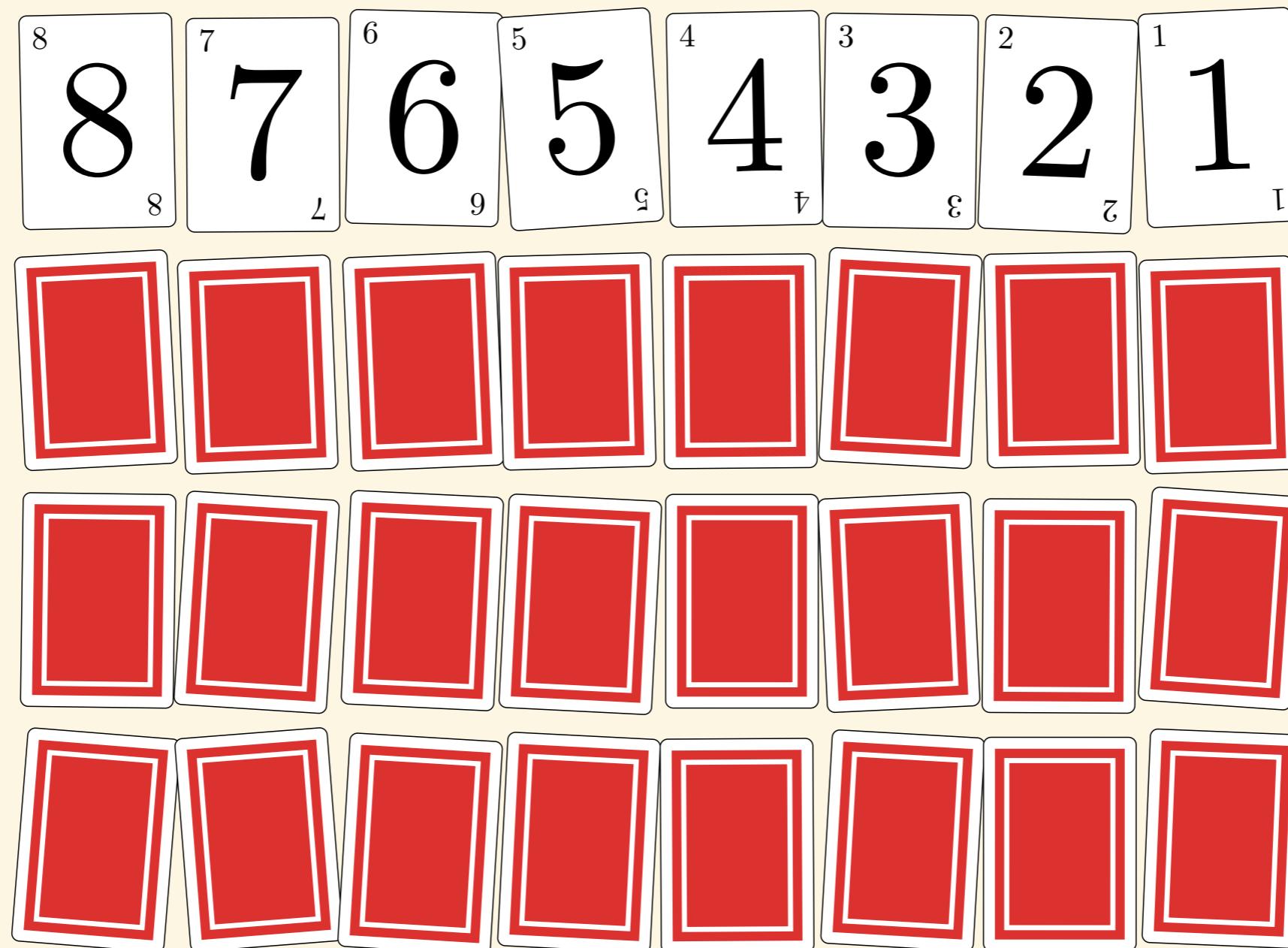


- Sortér venstre halvdel rekursivt
- Sortér høyre halvdel rekursivt
- **Flett sammen halvdelene**

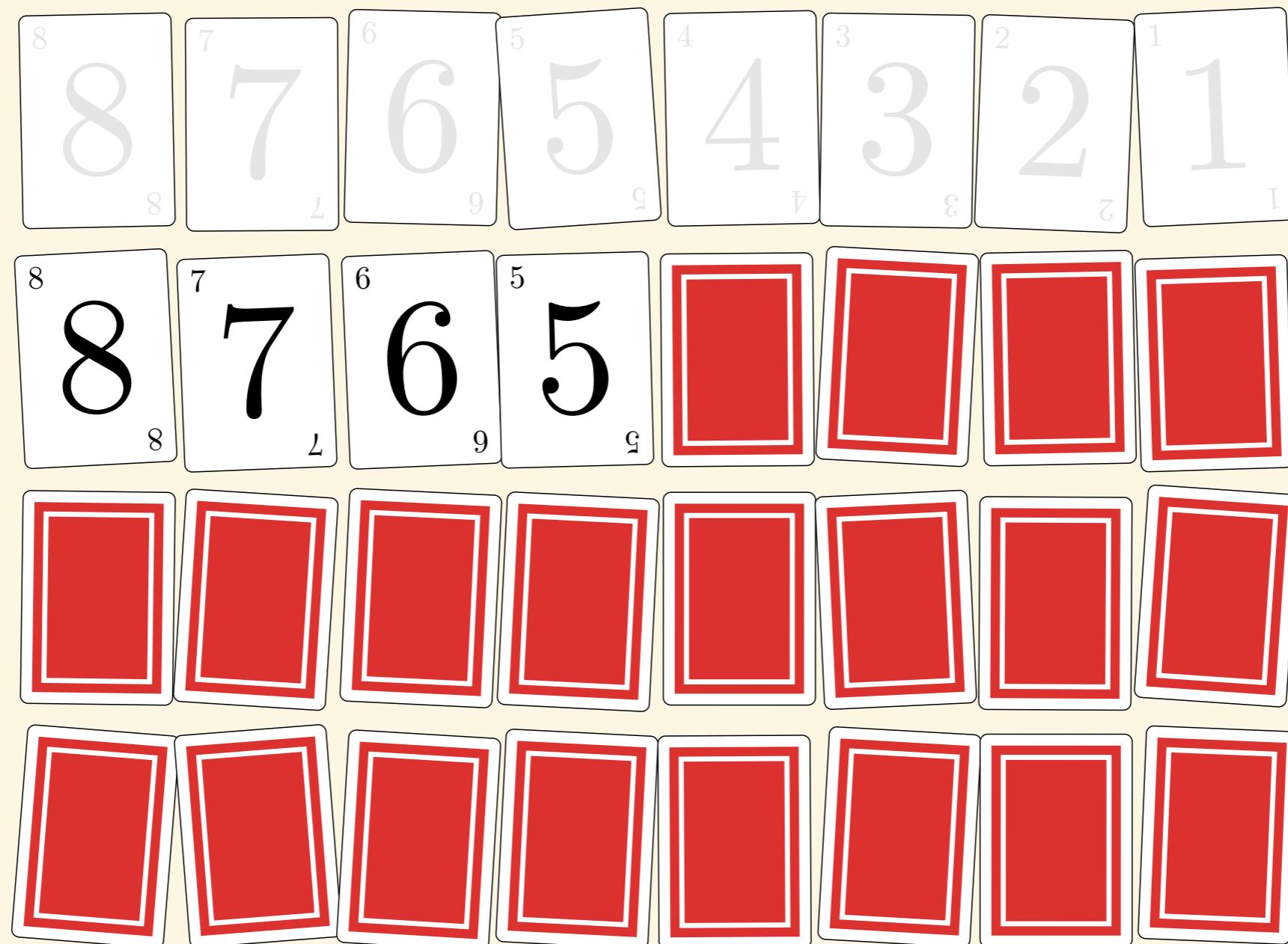
- Sortér venstre halvdel rekursivt
- Sortér høyre halvdel rekursivt
- Flett sammen halvdelene



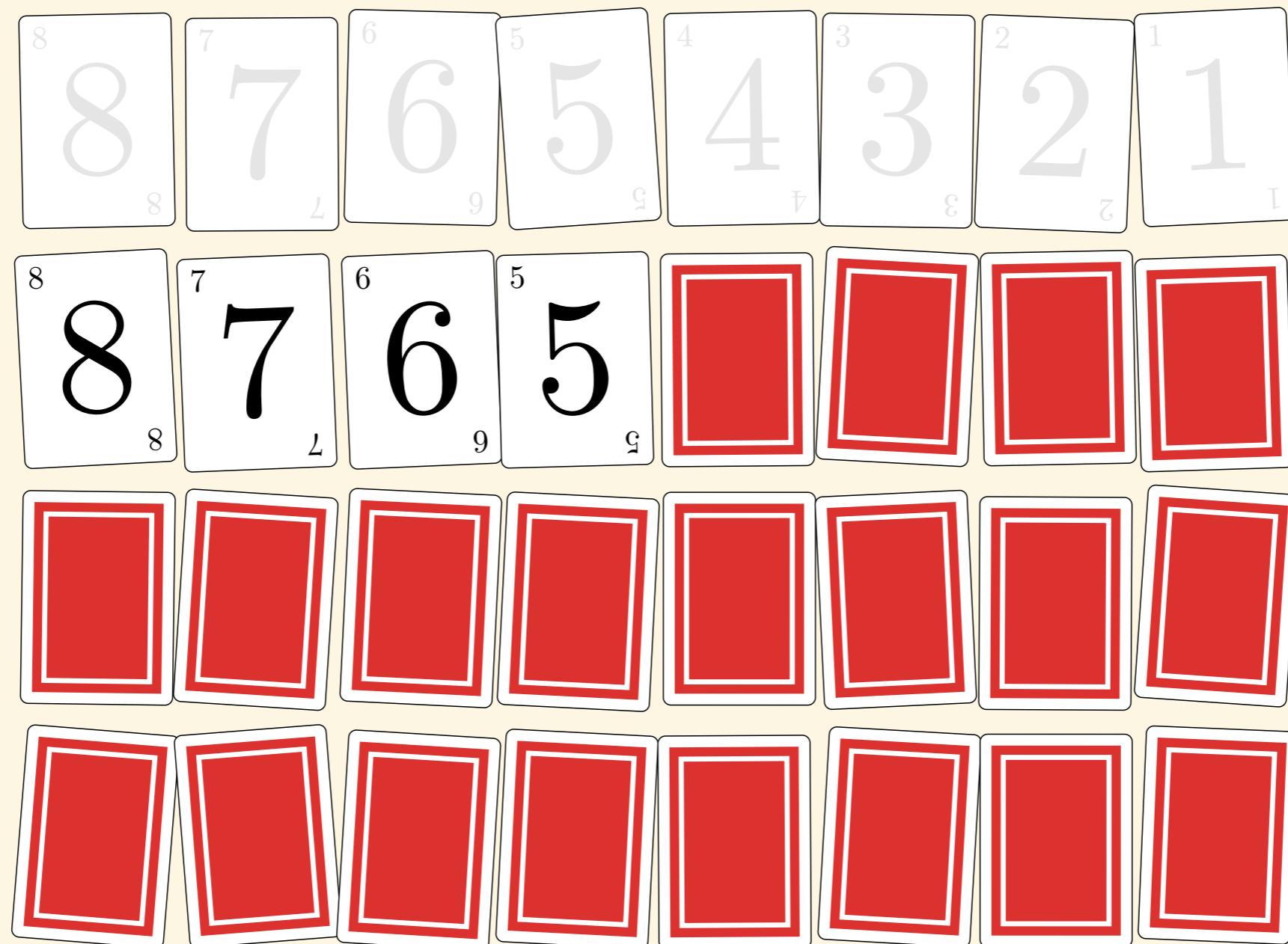
Skal sortere $A[1..8]$



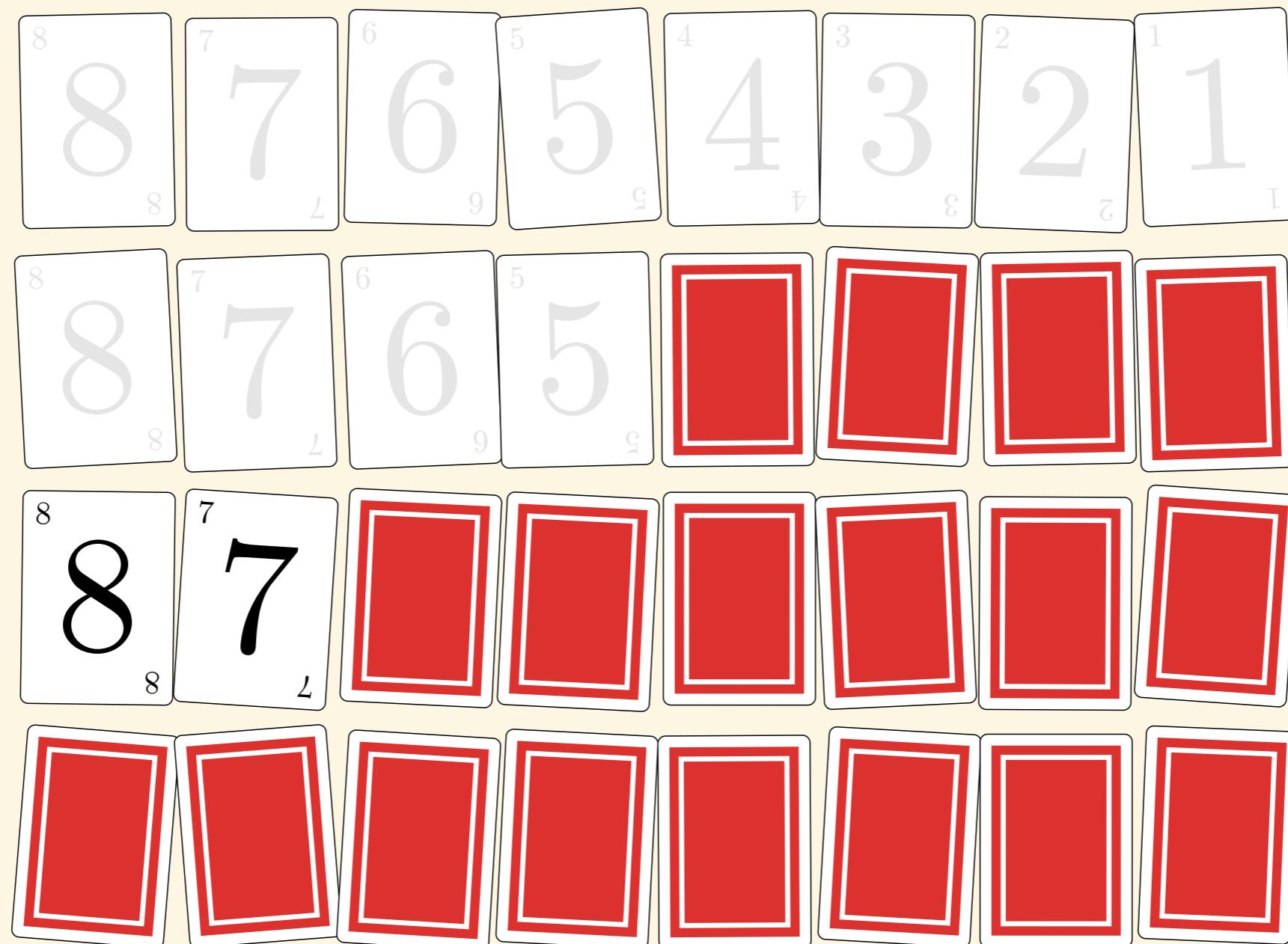
Sortér først $A[1..4]$ og så $A[5..8]$



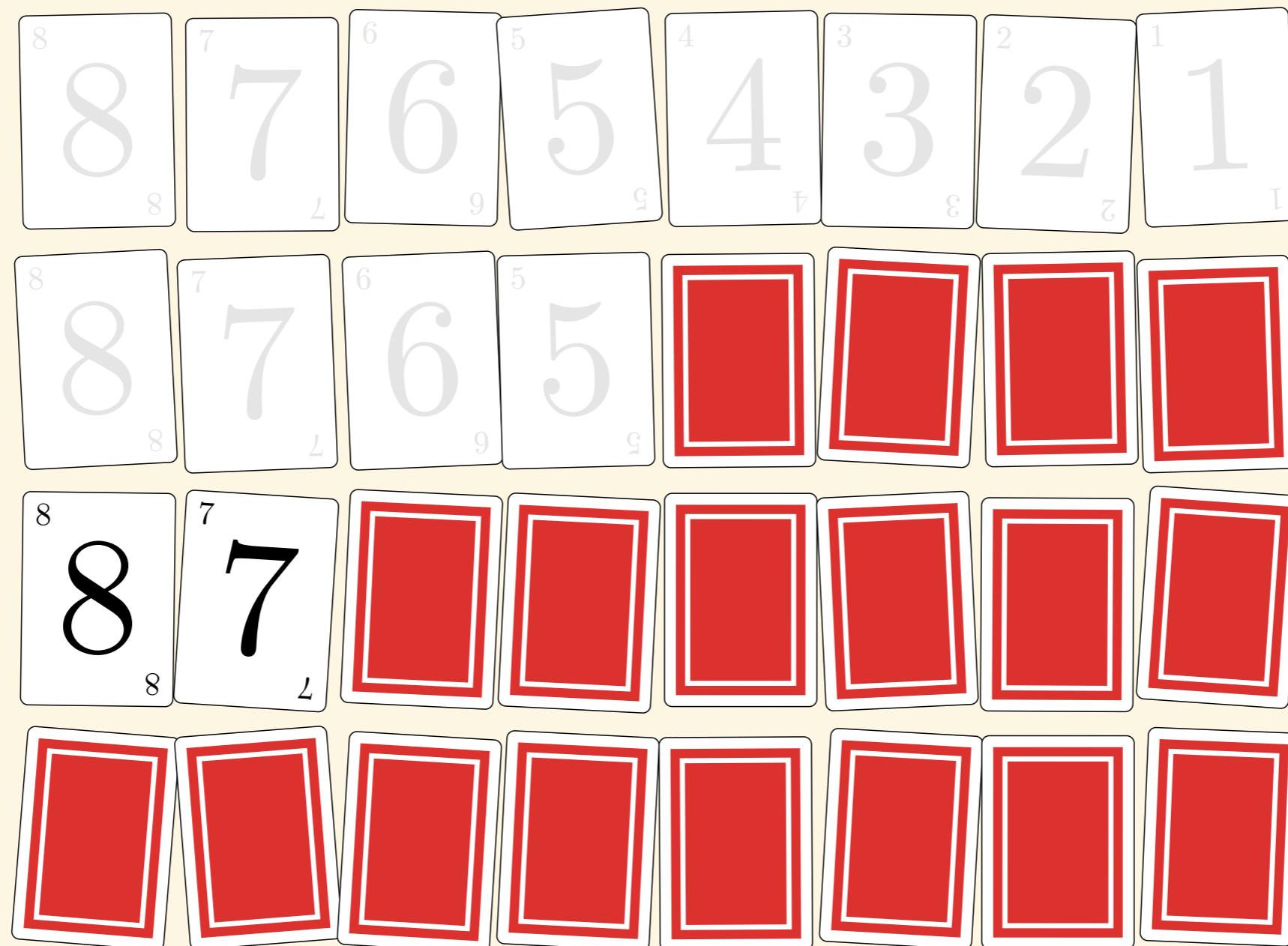
Skal sortere $A[1..4]$



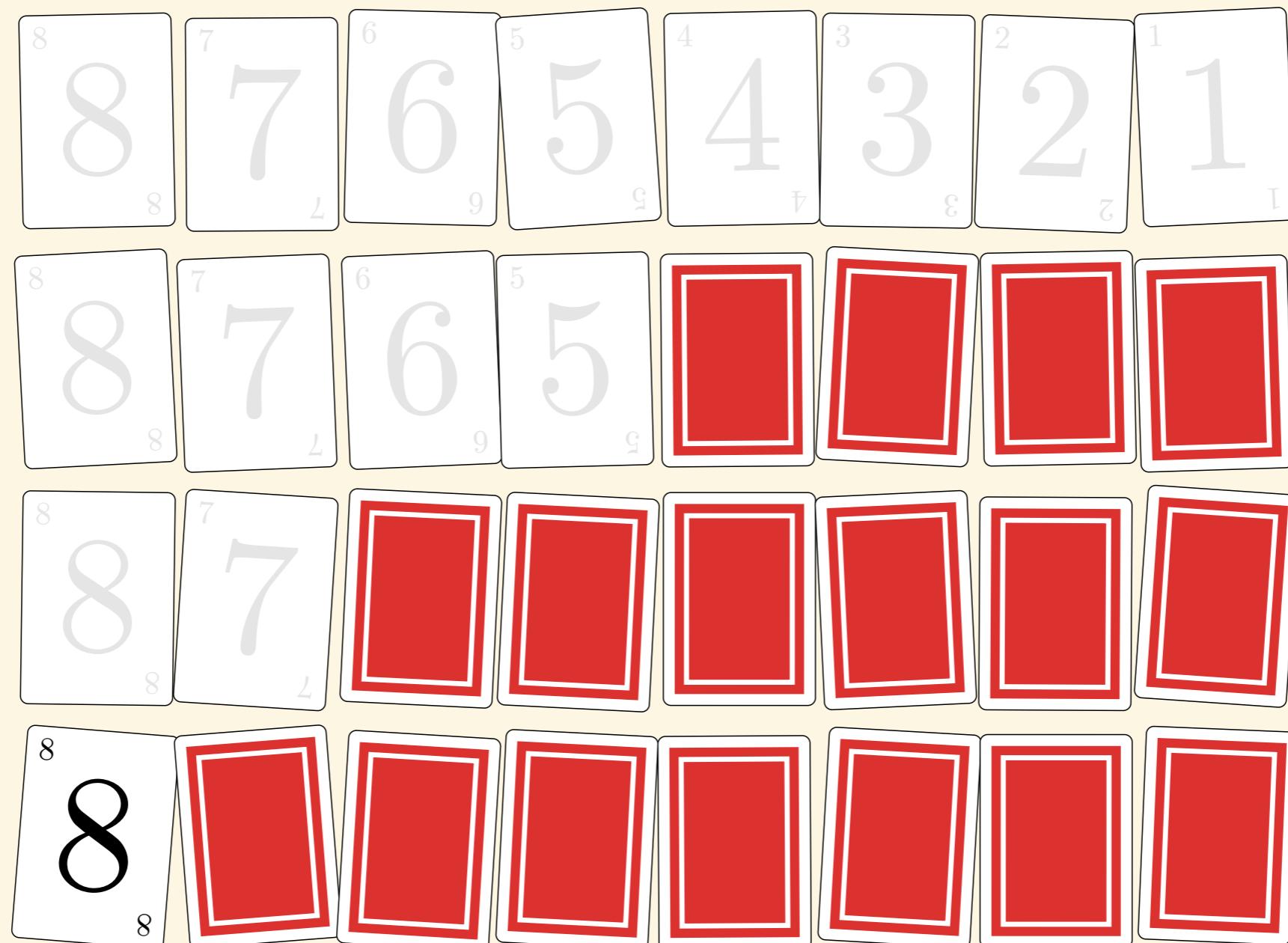
Sortér først $A[1..2]$ og så $A[3..4]$



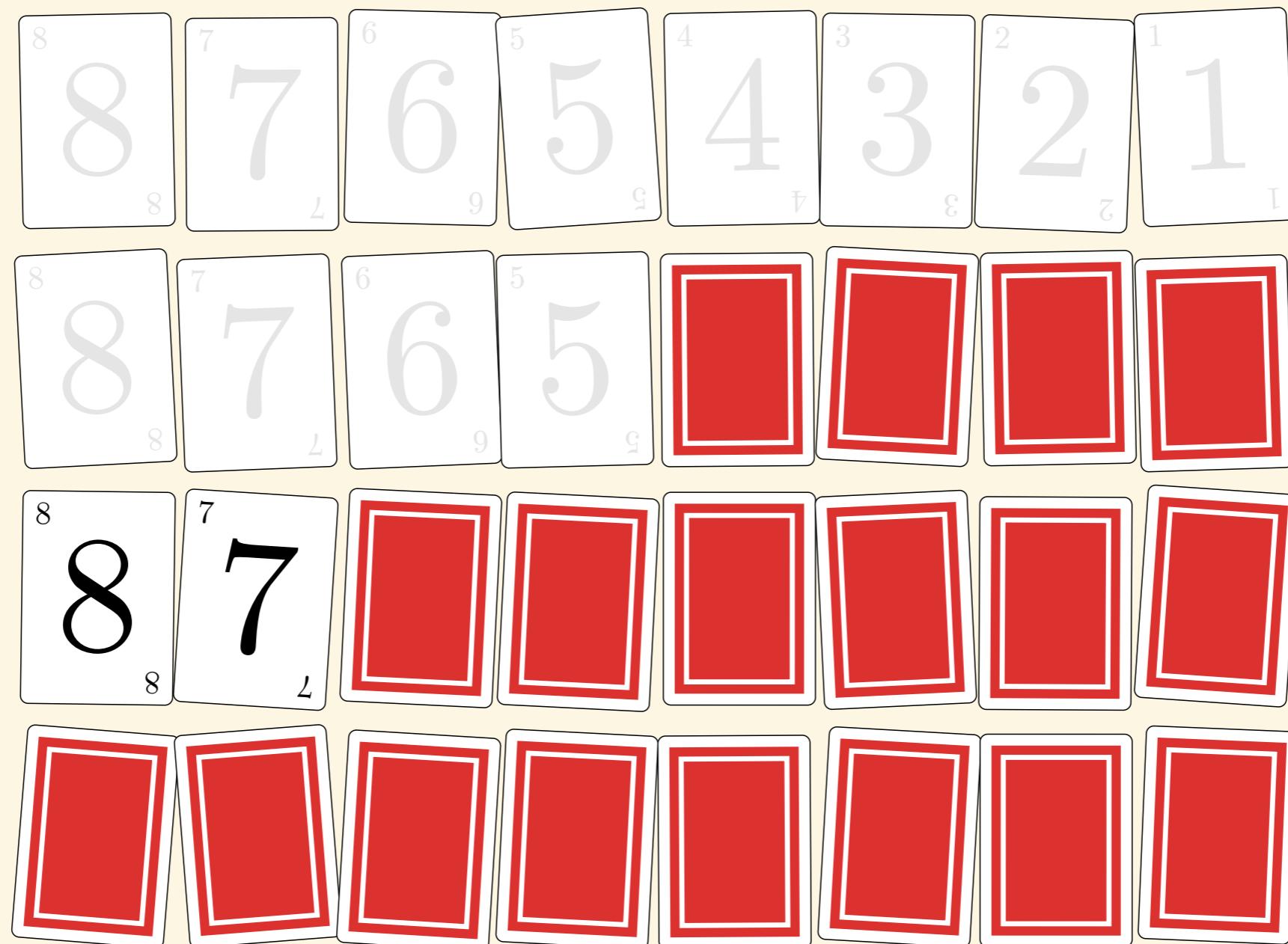
Skal sortere $A[1..2]$



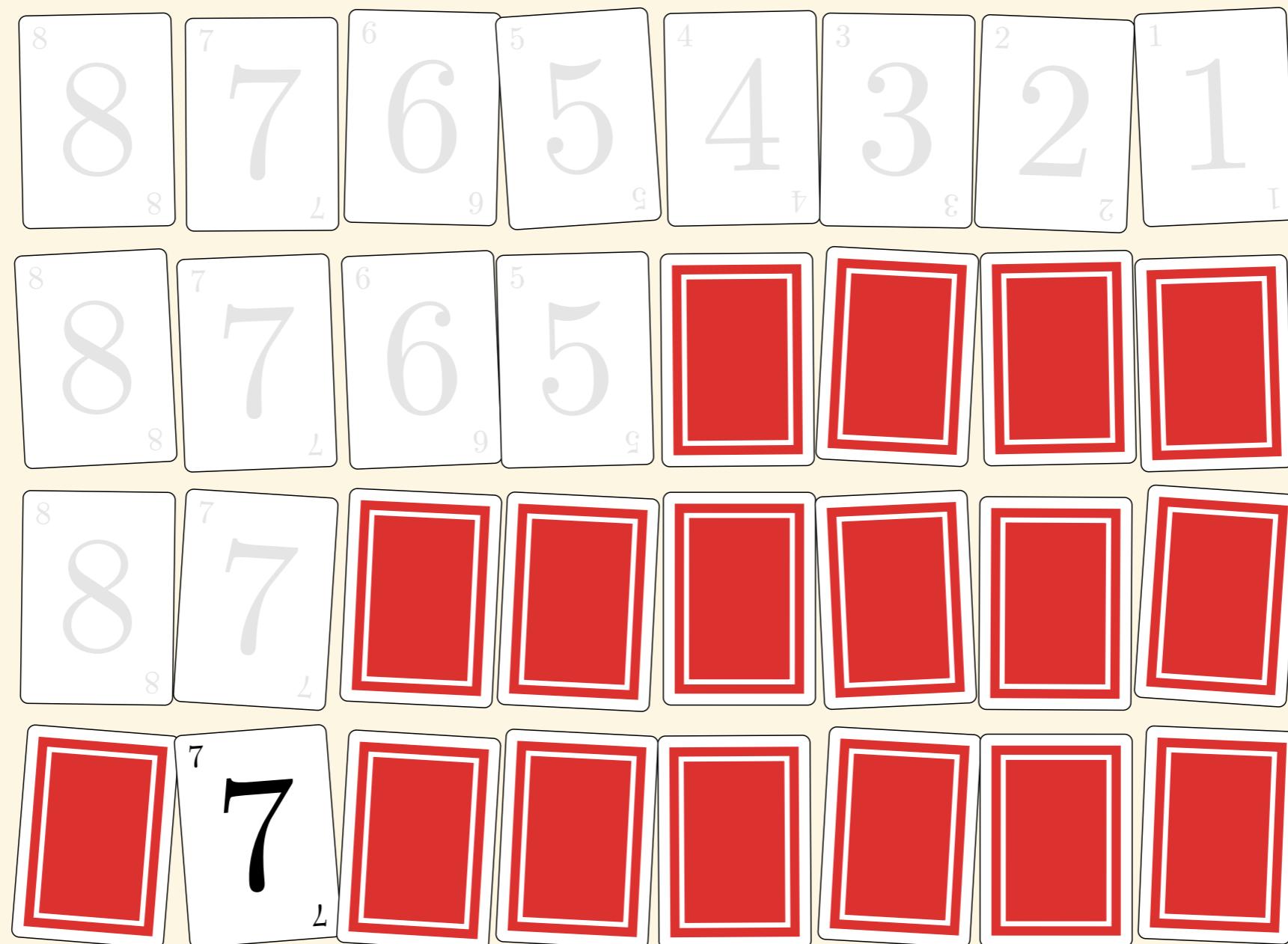
Sortér først $A[1]$ og så $A[2]$



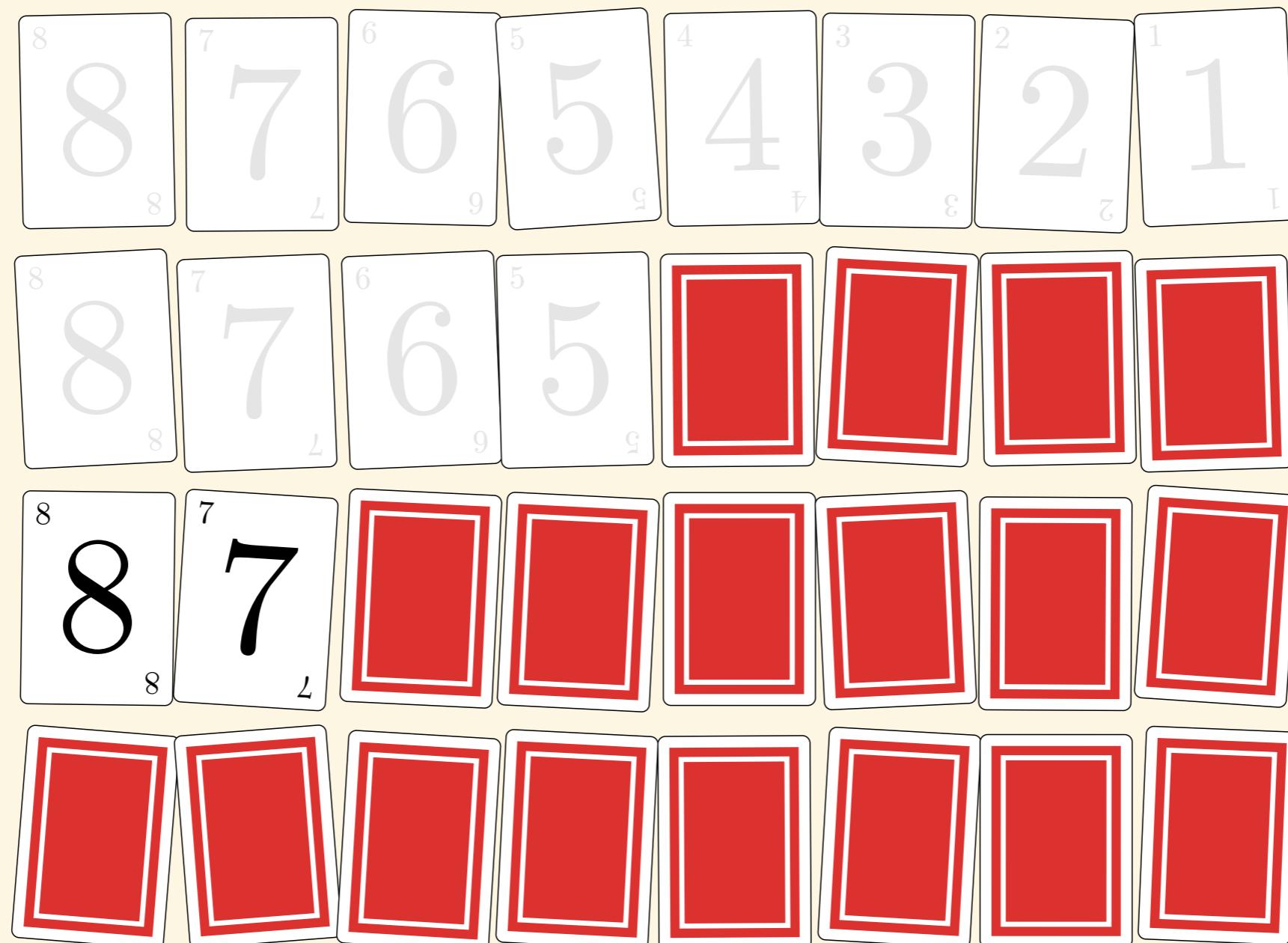
$A[1]$ er sortert



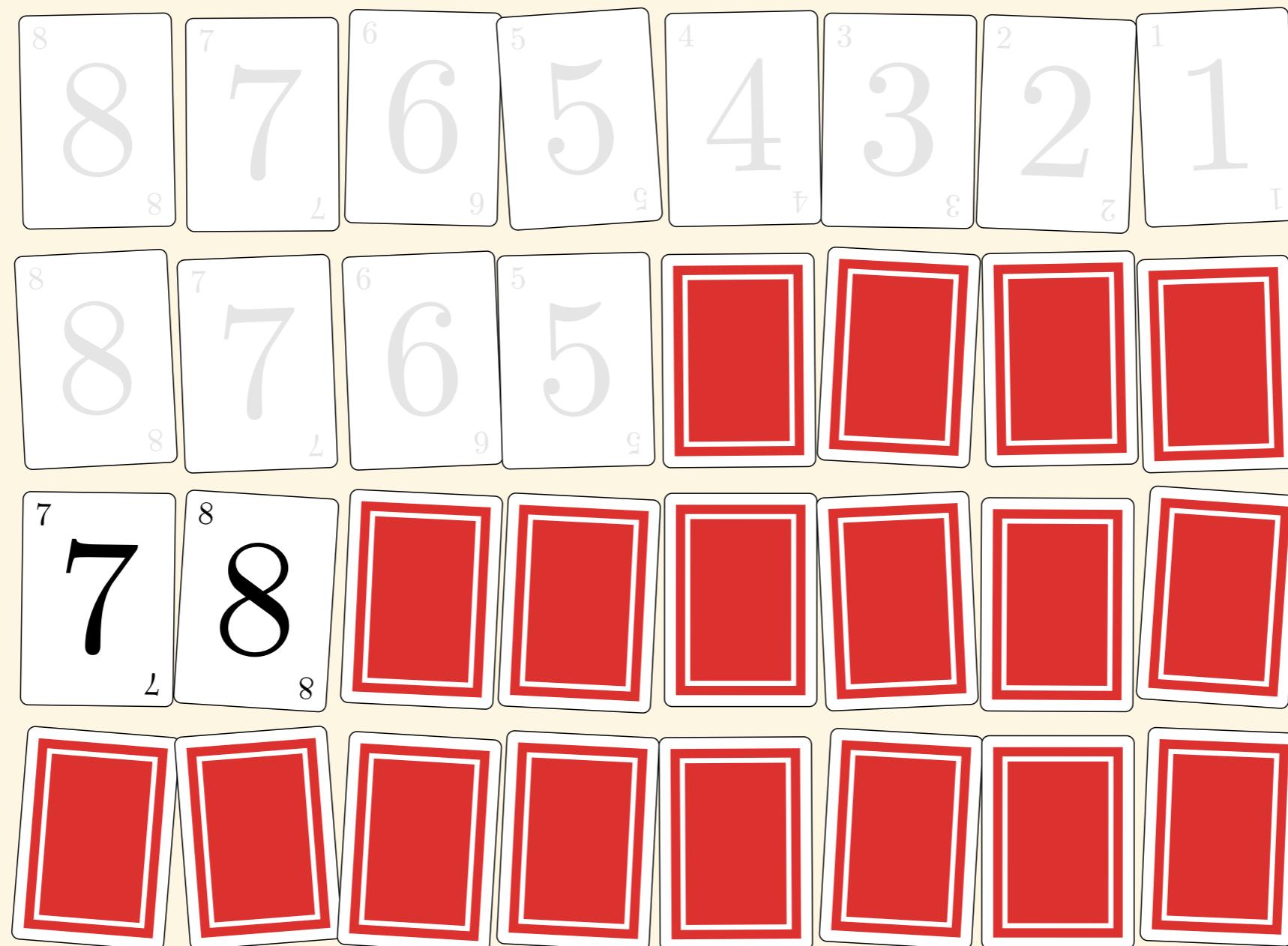
Sortér først $A[1]$ og så $A[2]$



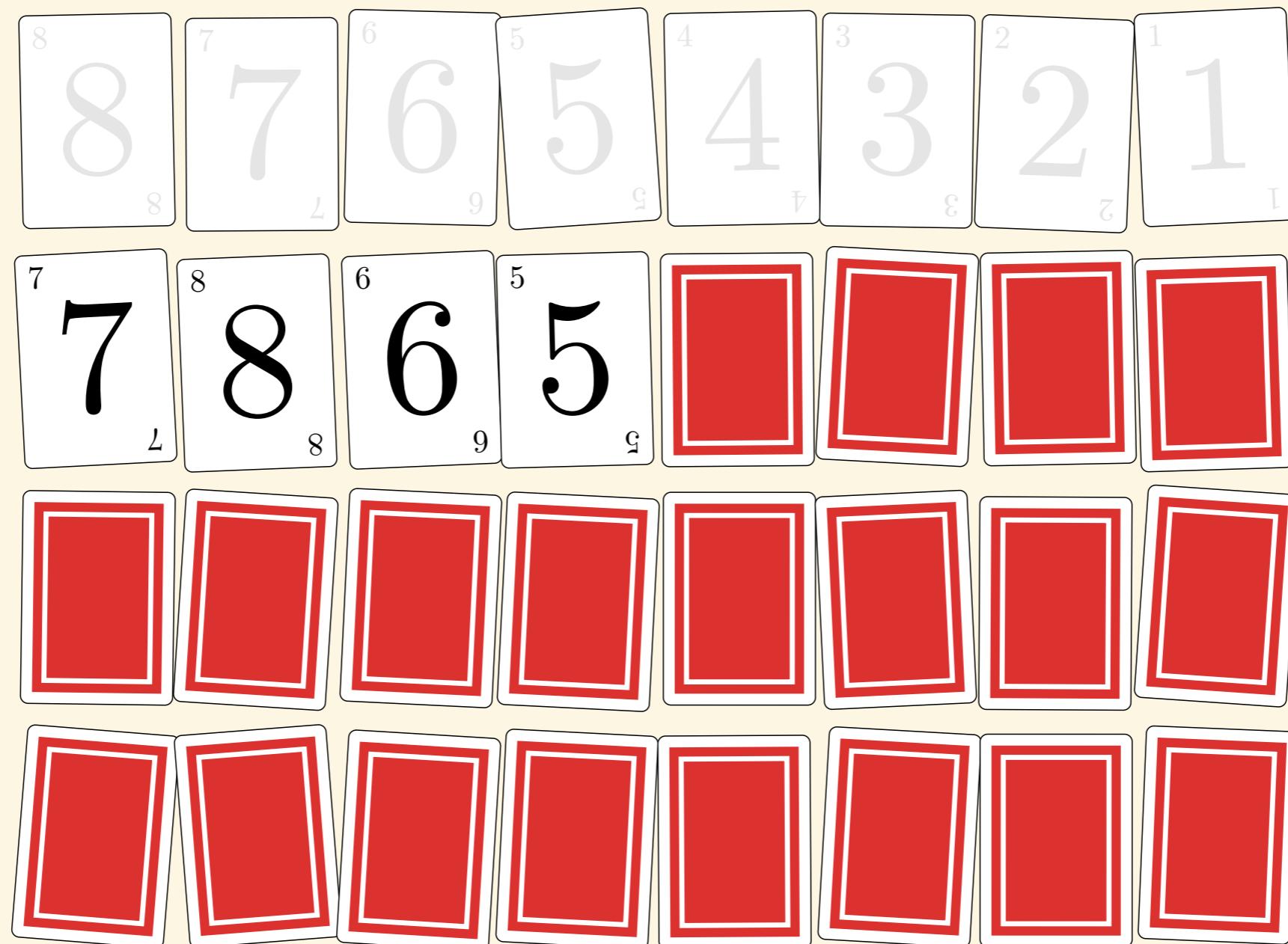
$A[2]$ er sortert



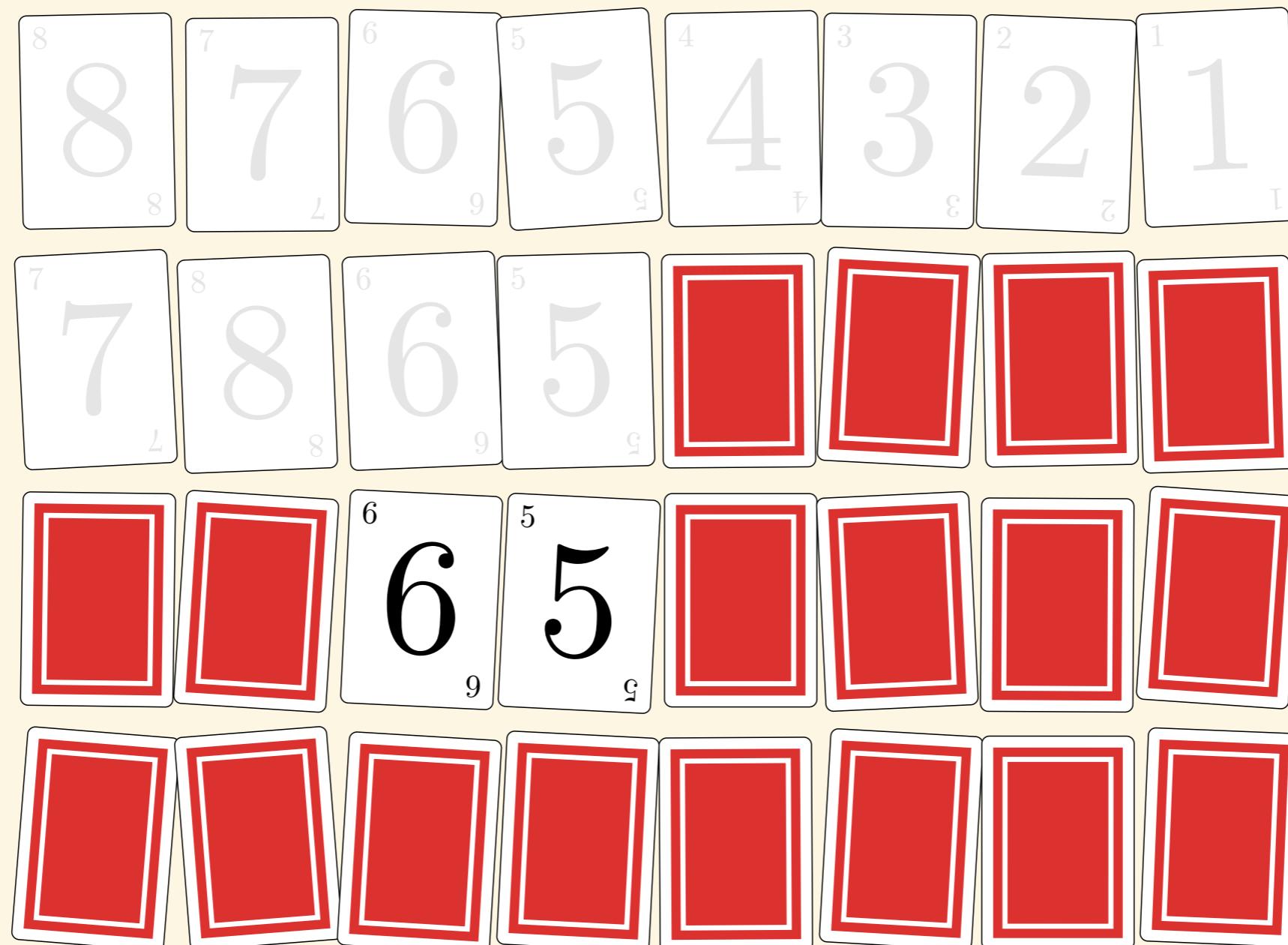
Flett sammen $A[1]$ og $A[2]$



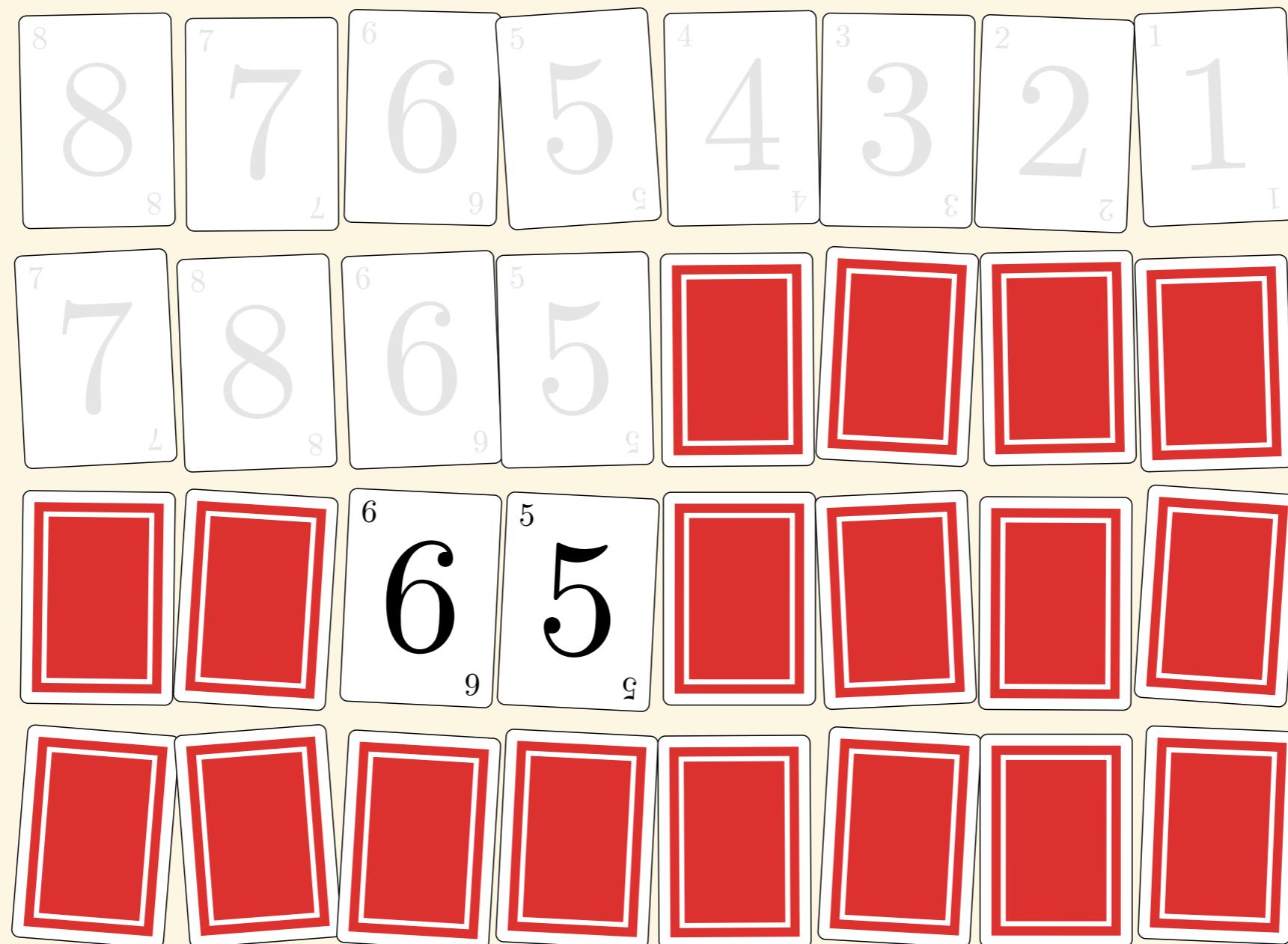
$A[1..2]$ er sortert



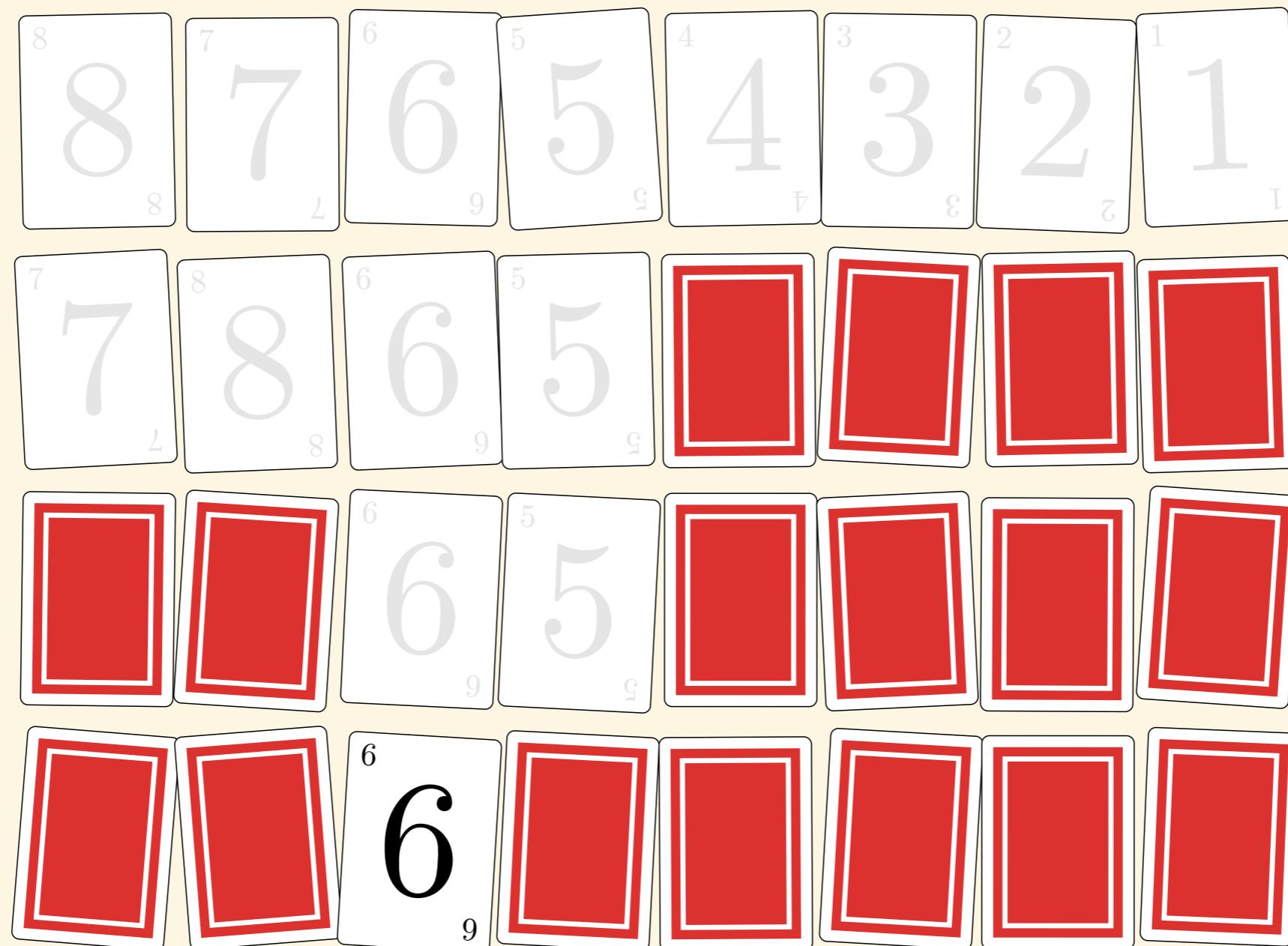
Sortér først $A[1..2]$ og så $A[3..4]$



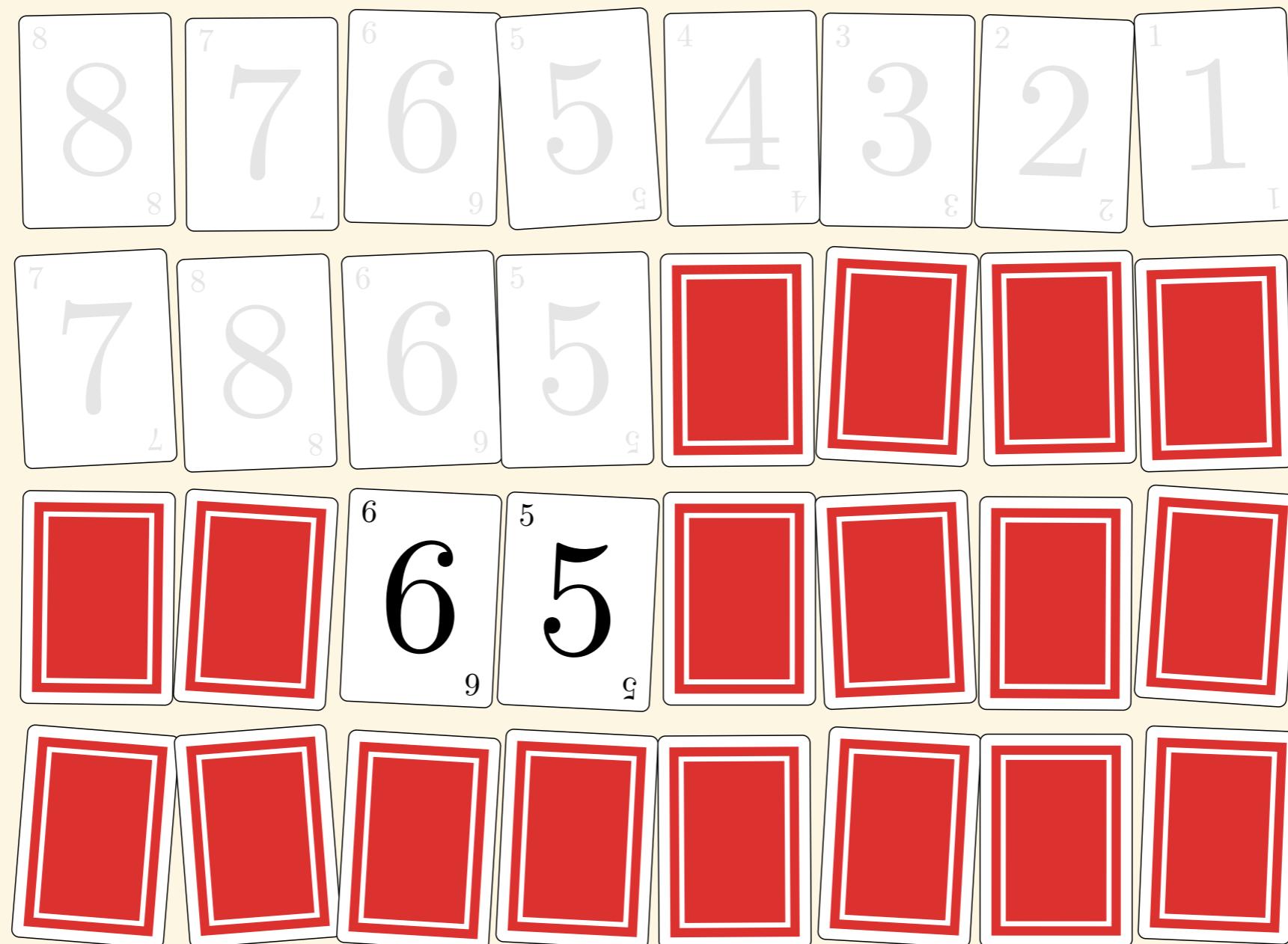
Skal sortere $A[3..4]$



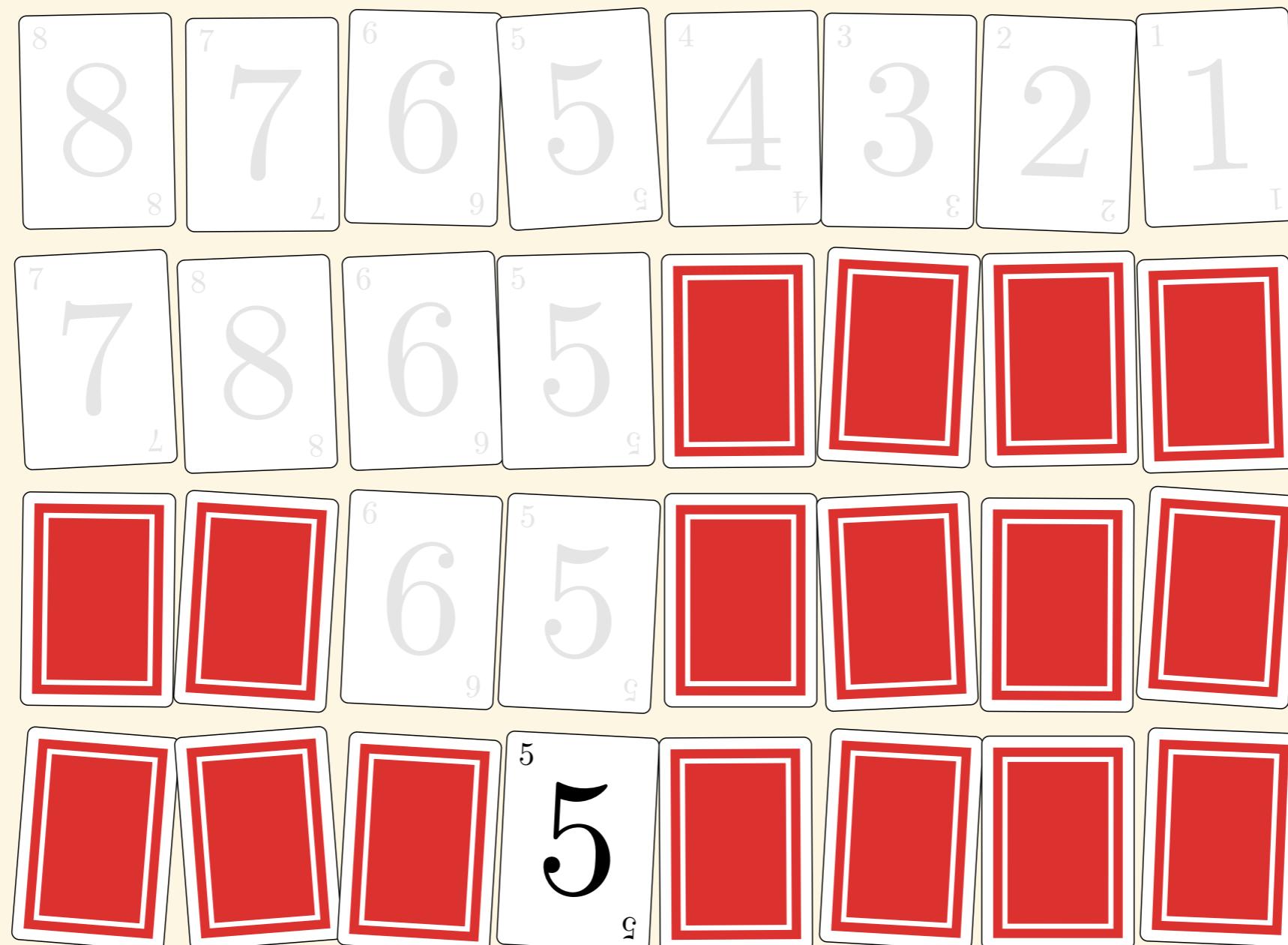
Sortér først $A[3]$ og så $A[4]$



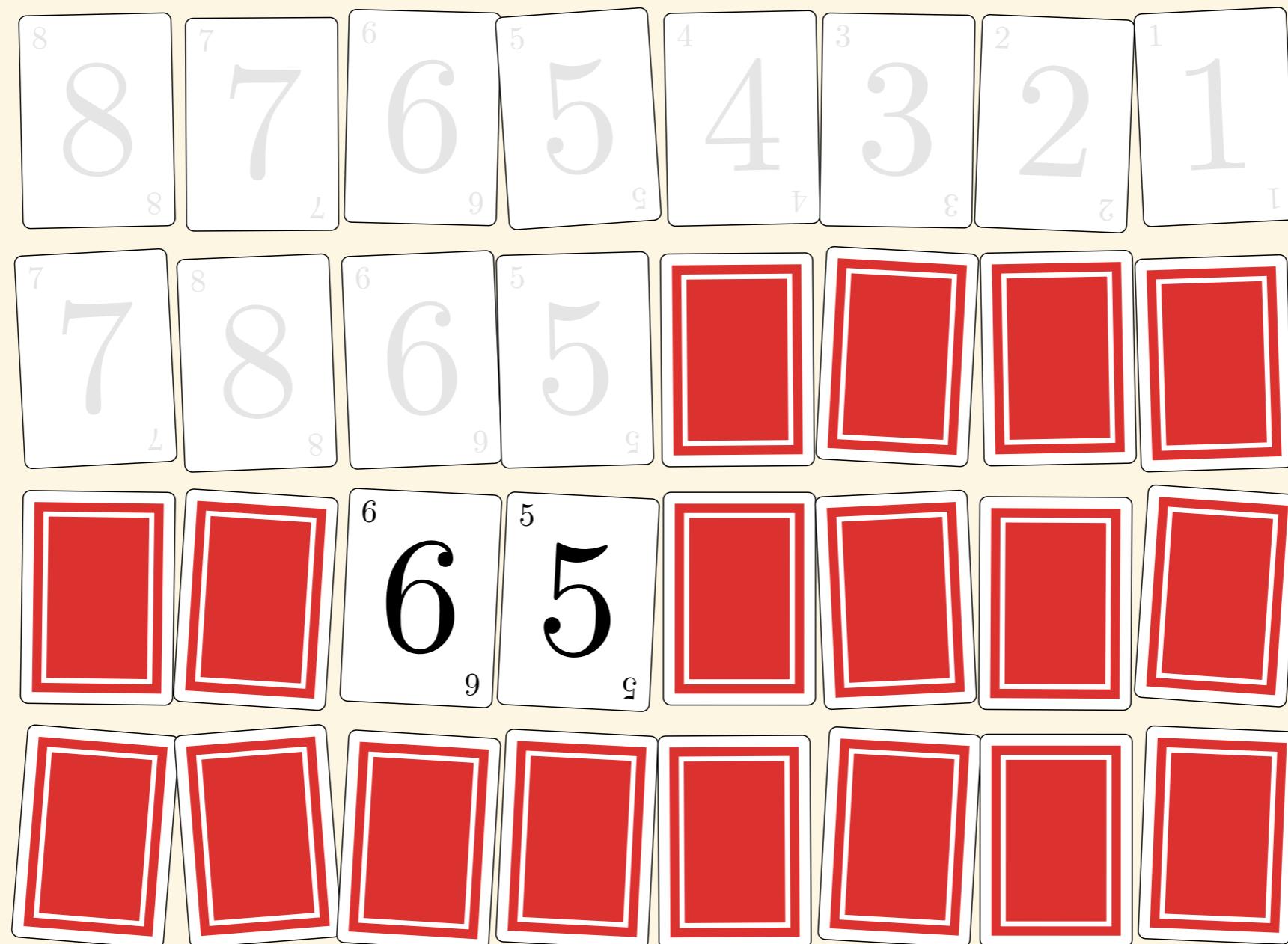
$A[3]$ er sortert



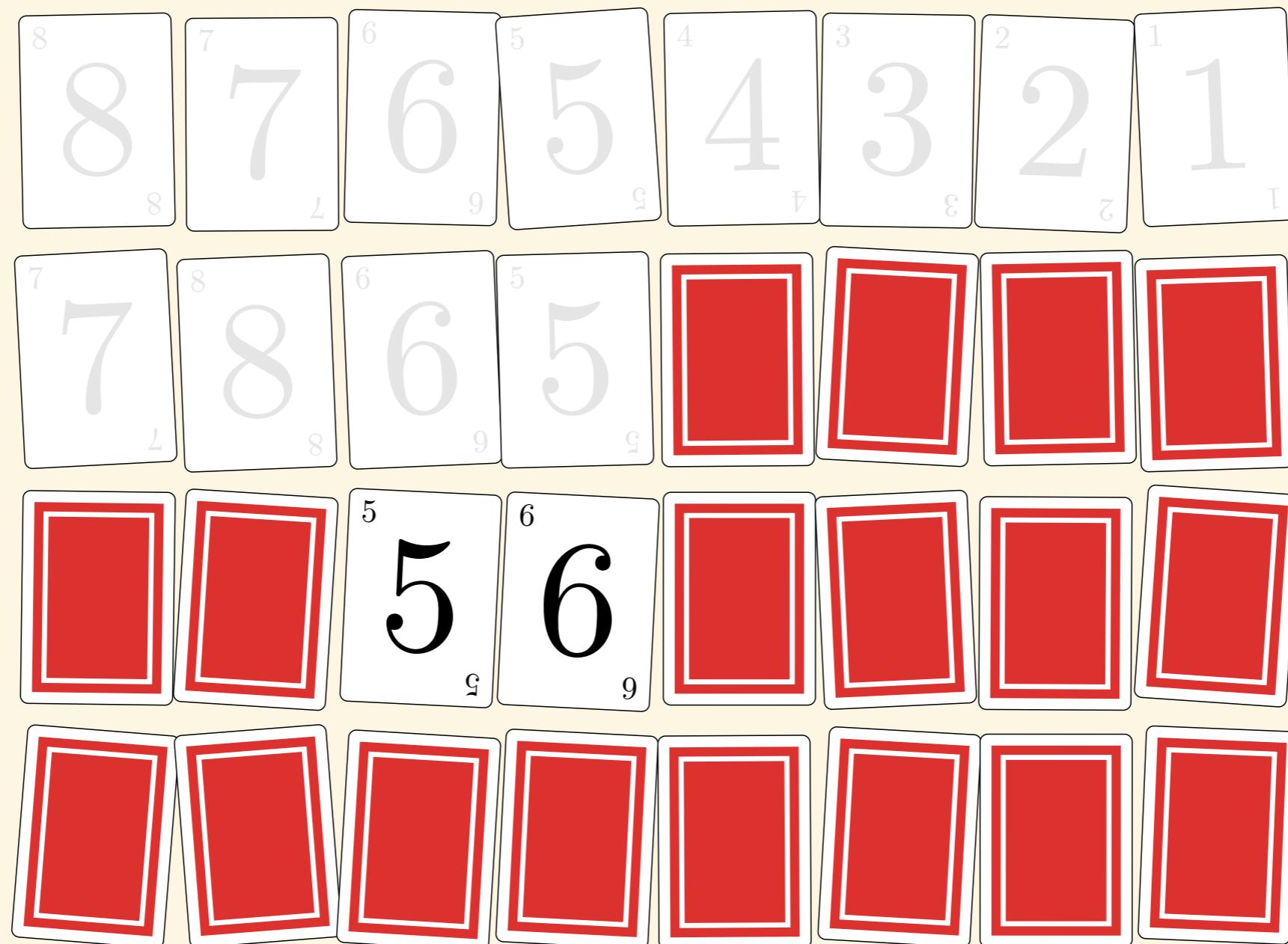
Sortér først $A[3]$ og så $A[4]$



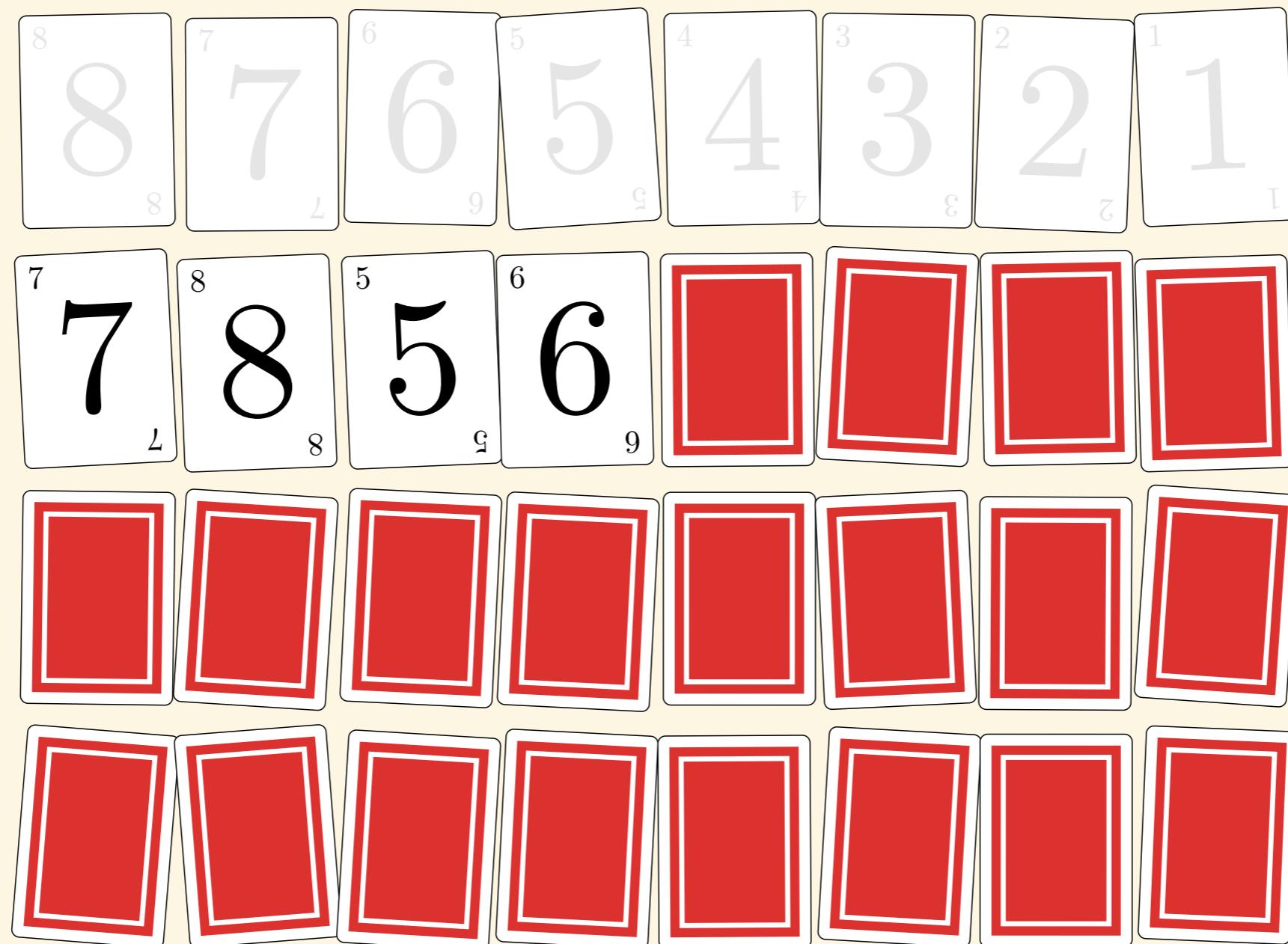
$A[4]$ er sortert



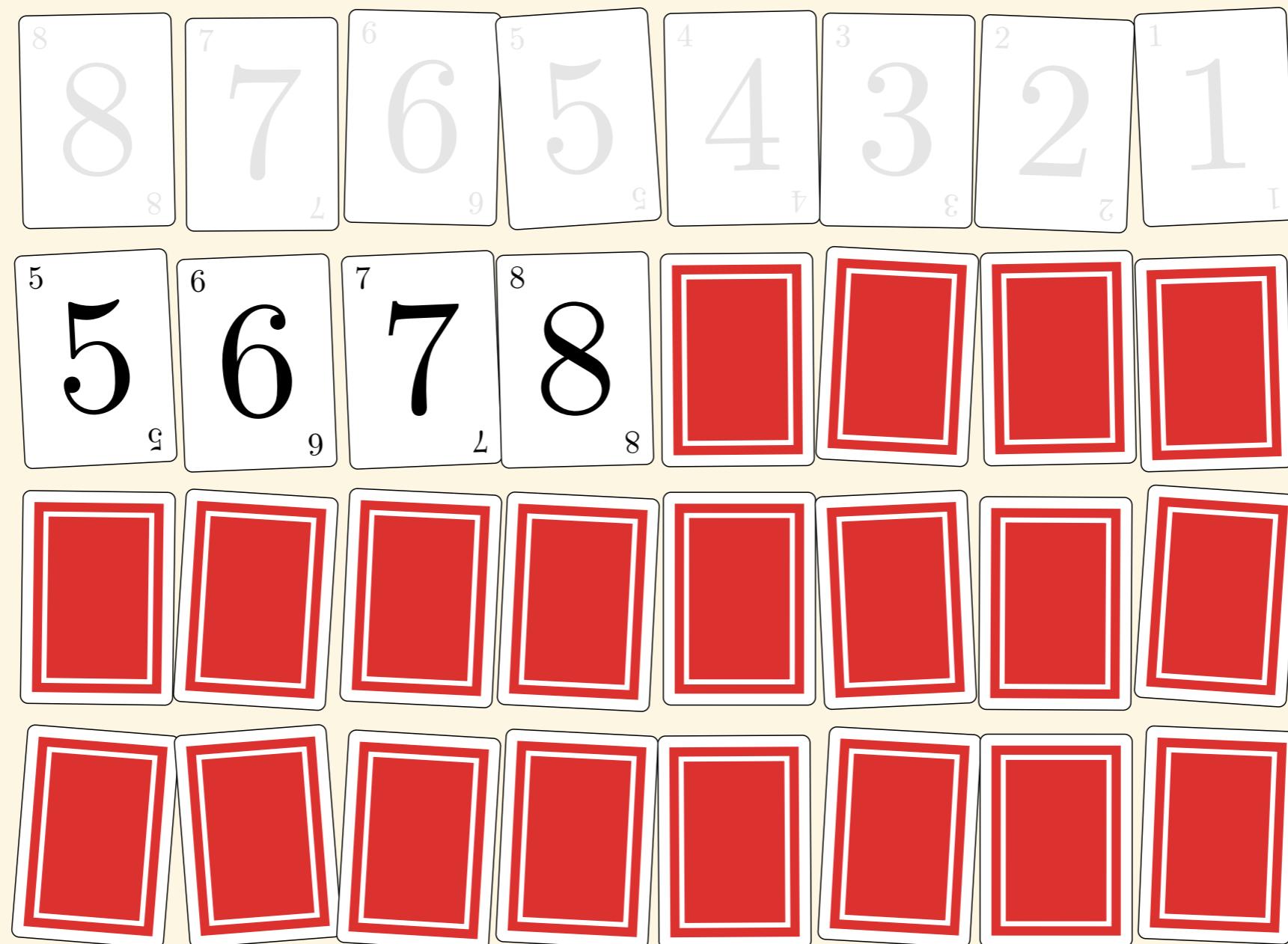
Flett sammen $A[3]$ og $A[4]$



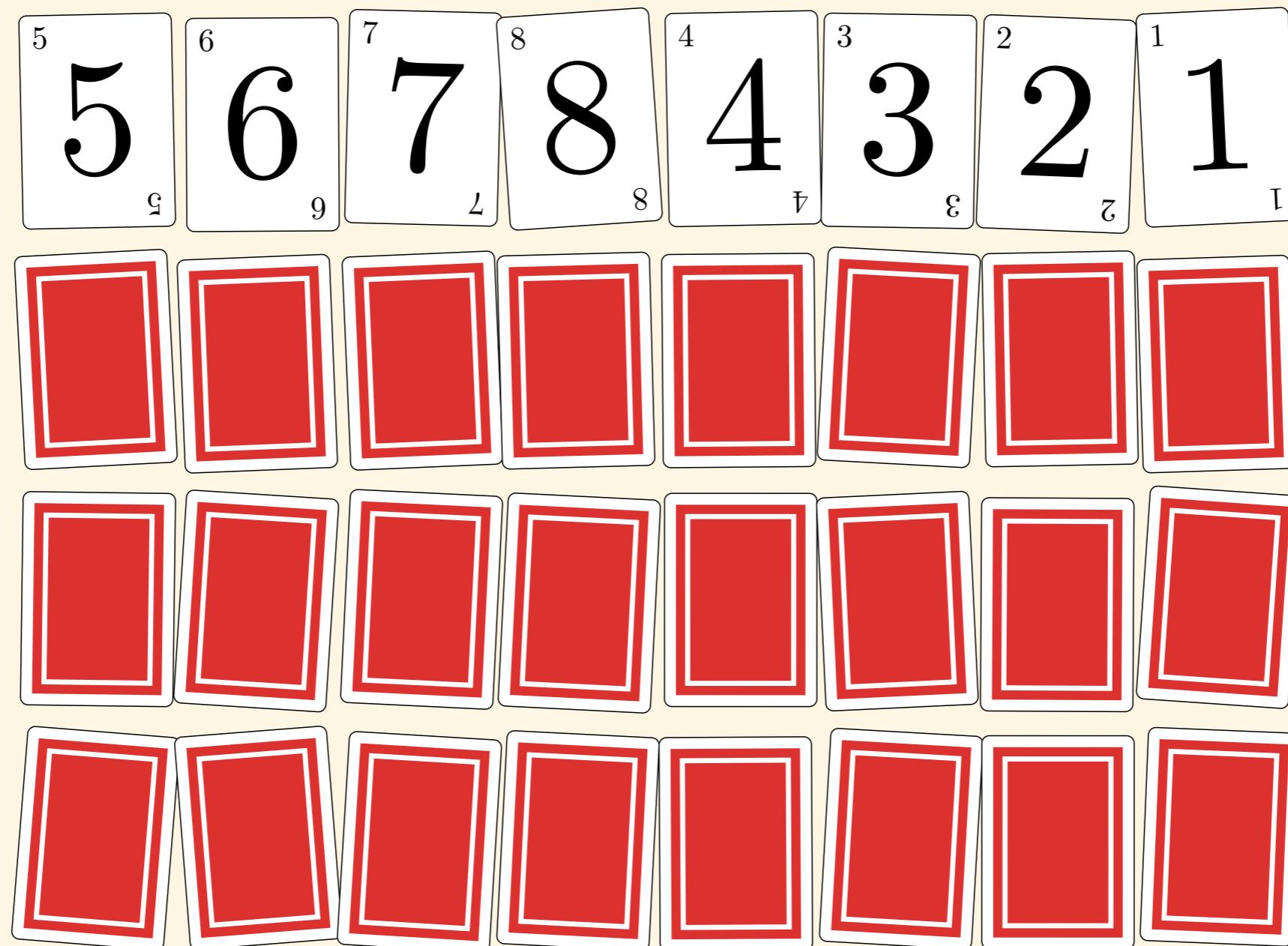
$A[3..4]$ er sortert



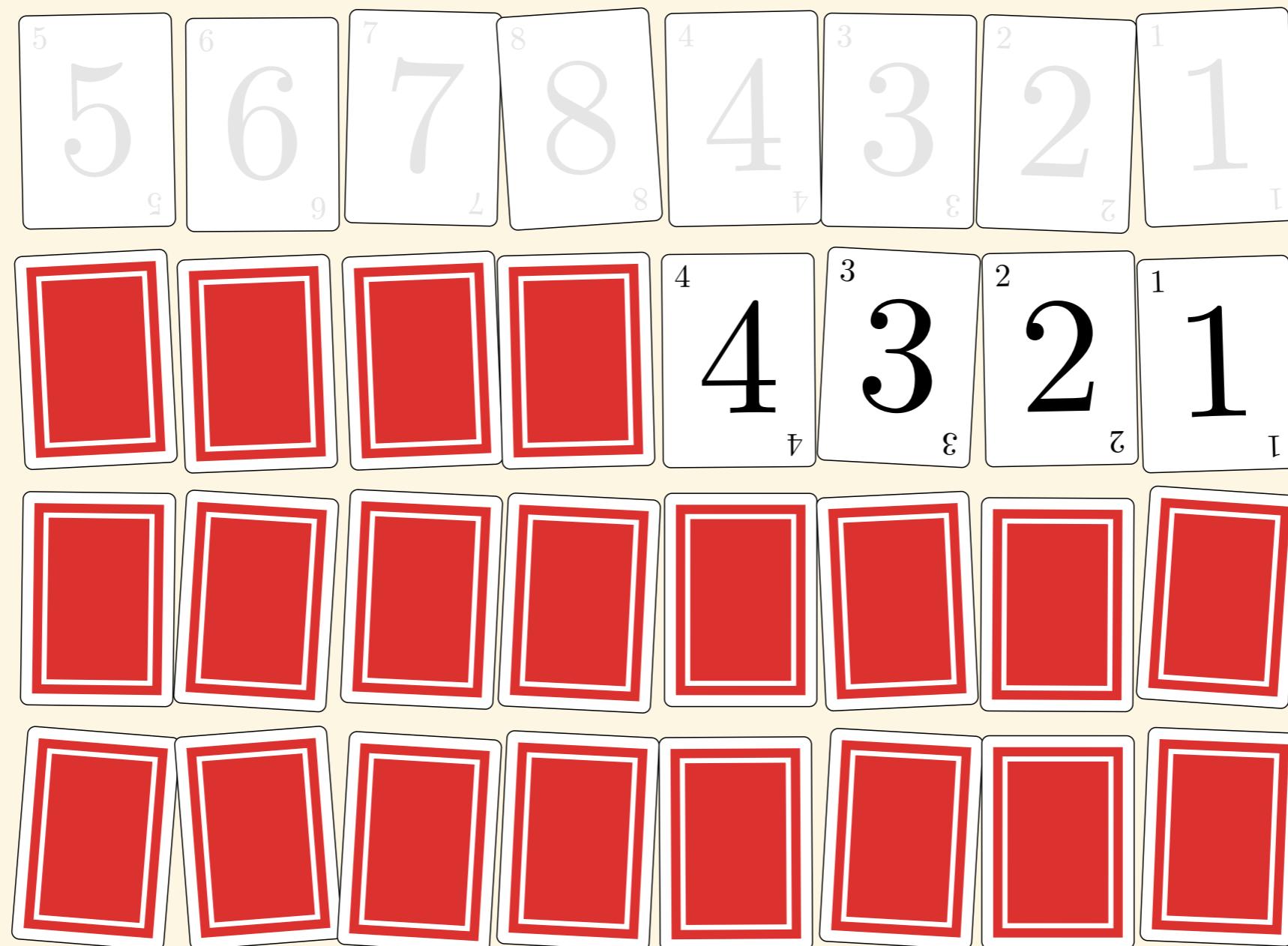
Flett sammen $A[1..2]$ og $A[3..4]$



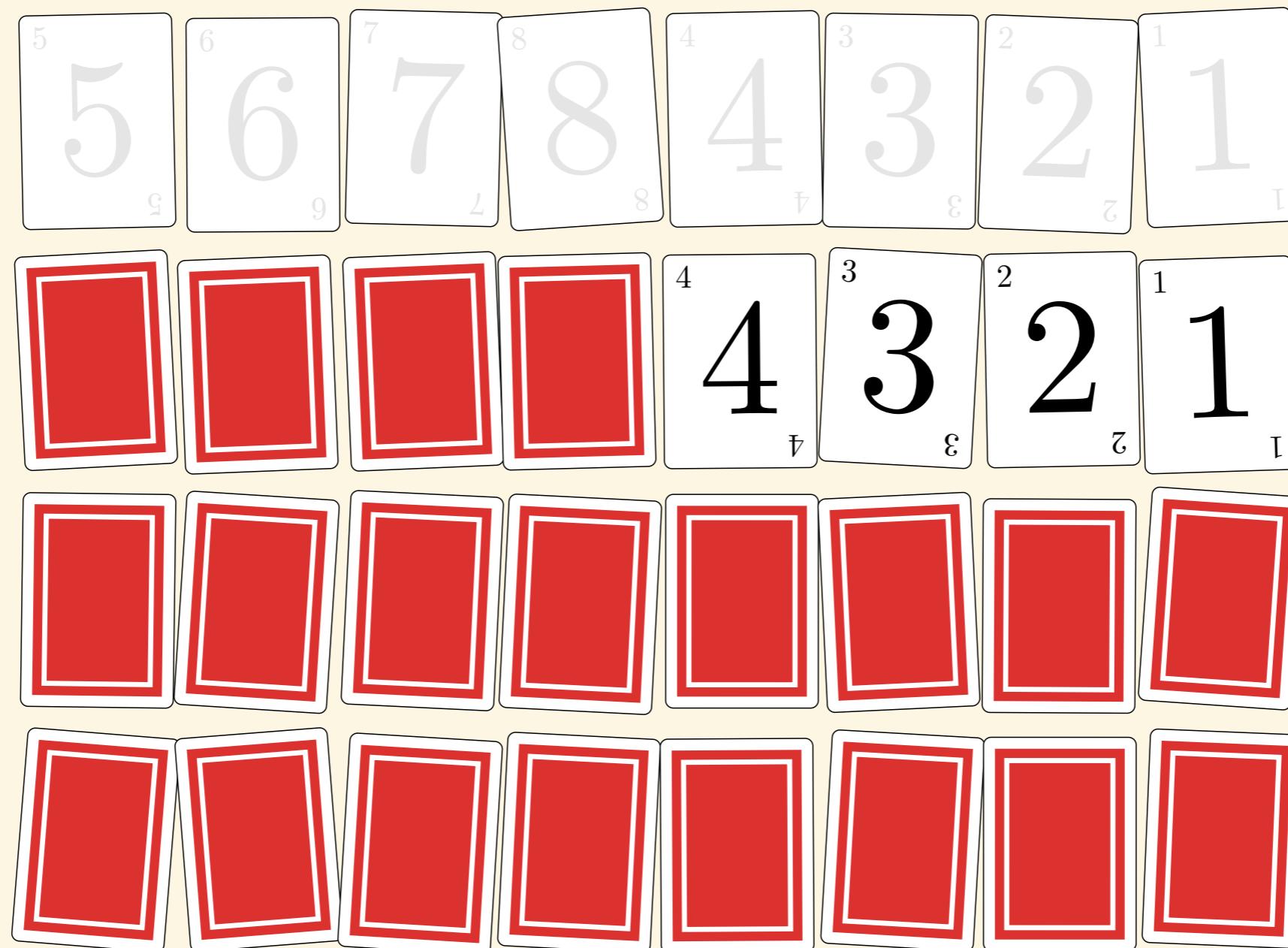
$A[1..4]$ er sortert



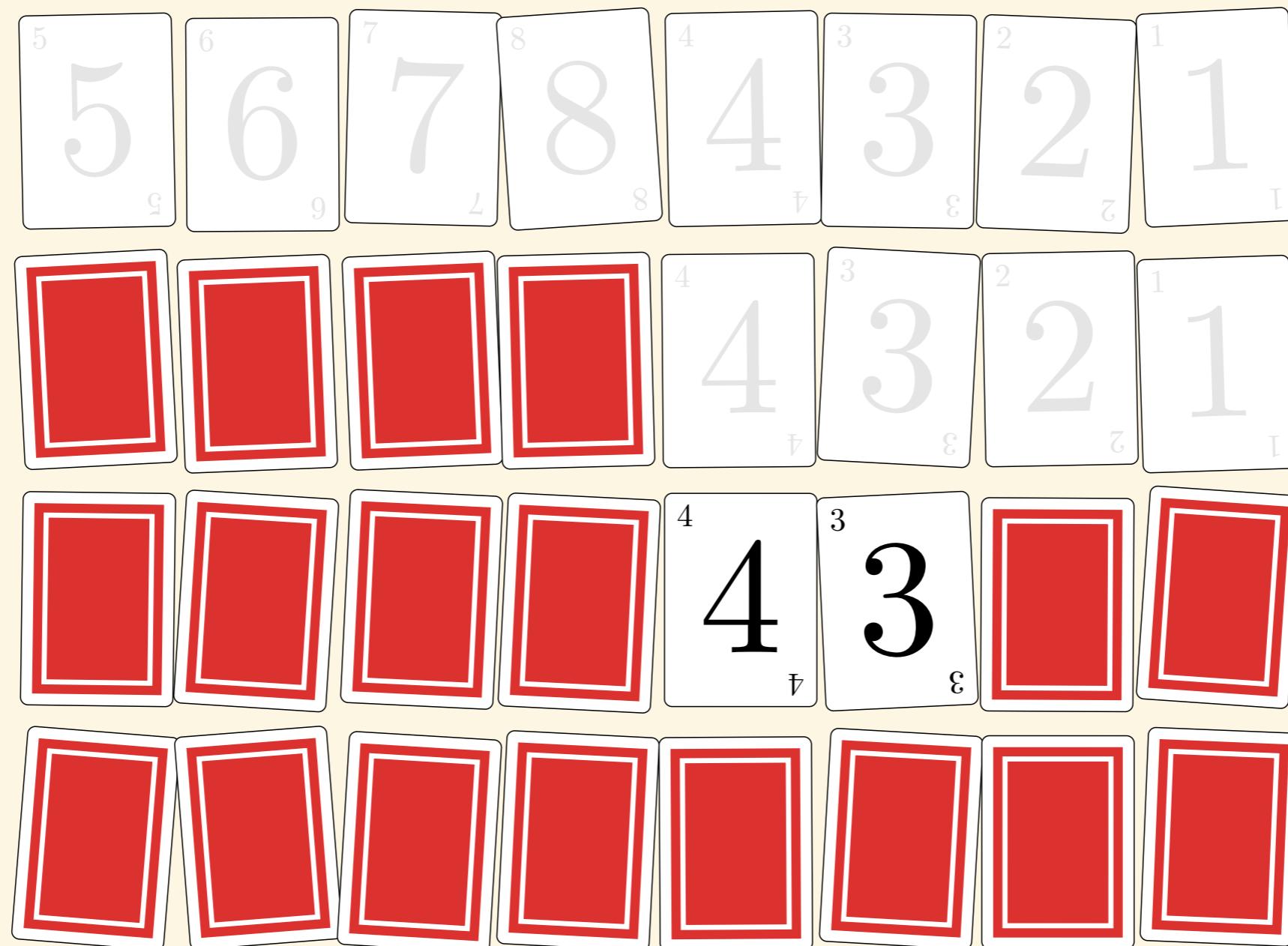
Sortér først $A[1..4]$ og så $A[5..8]$



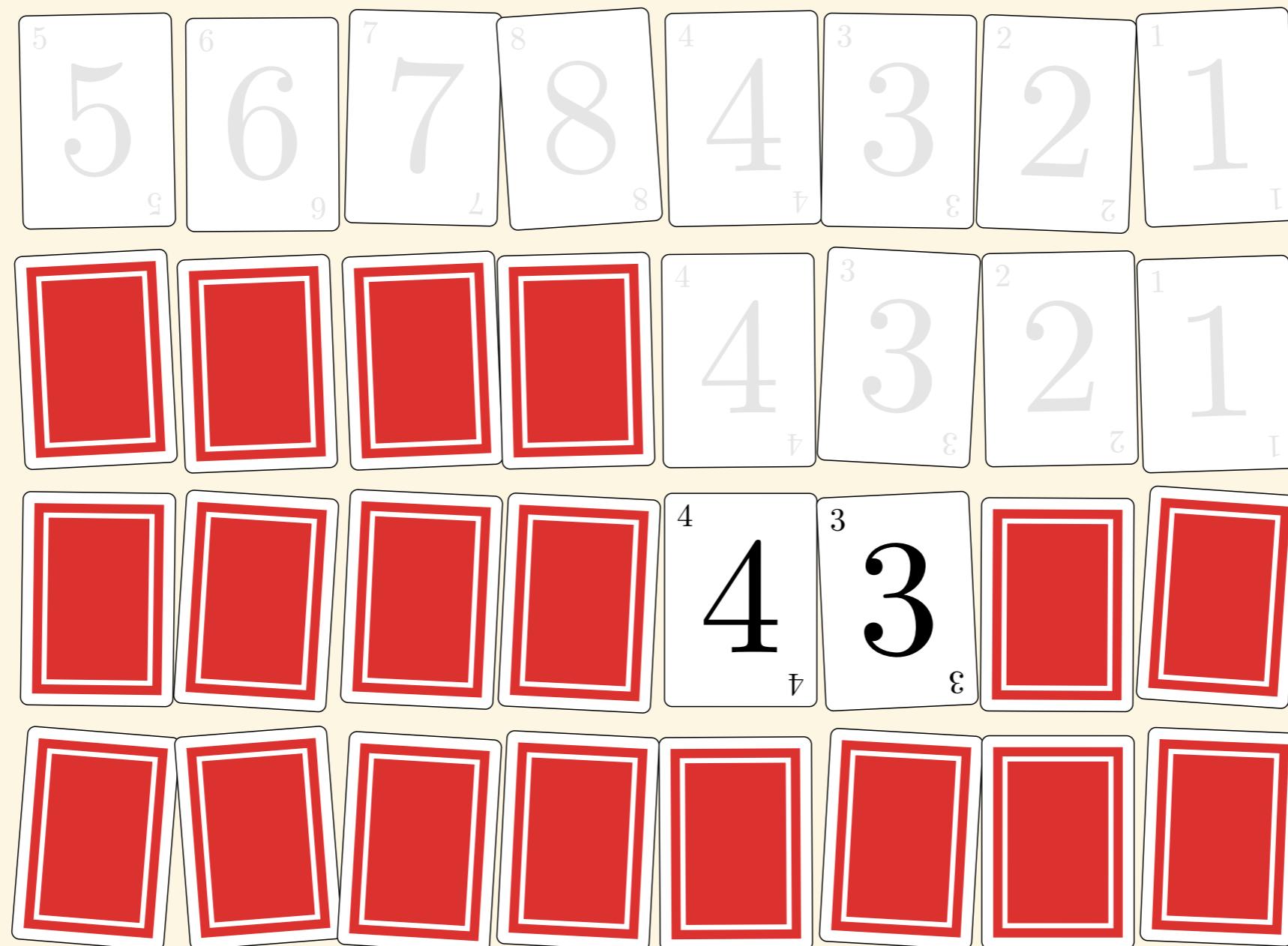
Skal sortere $A[5..8]$



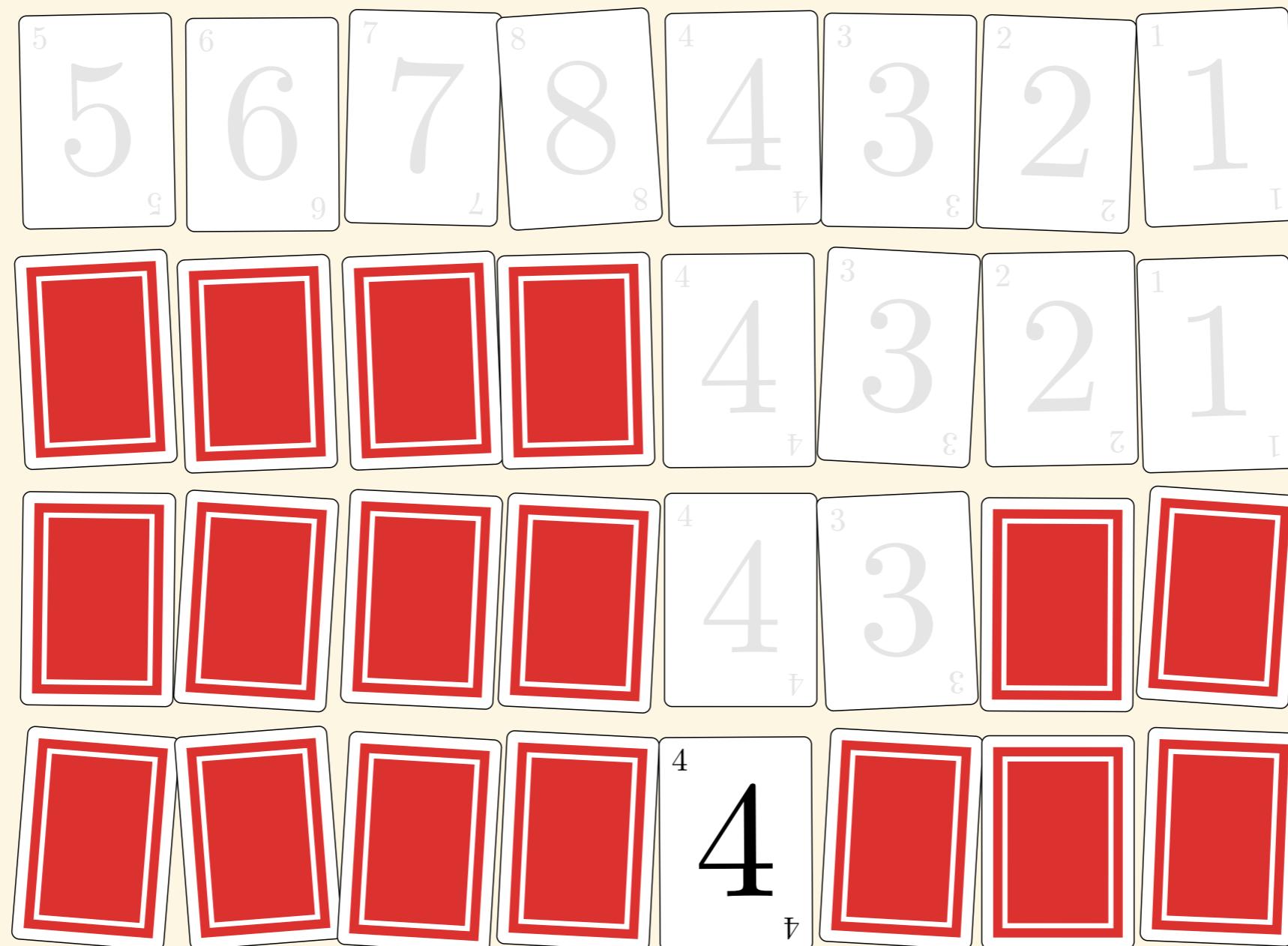
Sortér først $A[5..6]$ og så $A[7..8]$



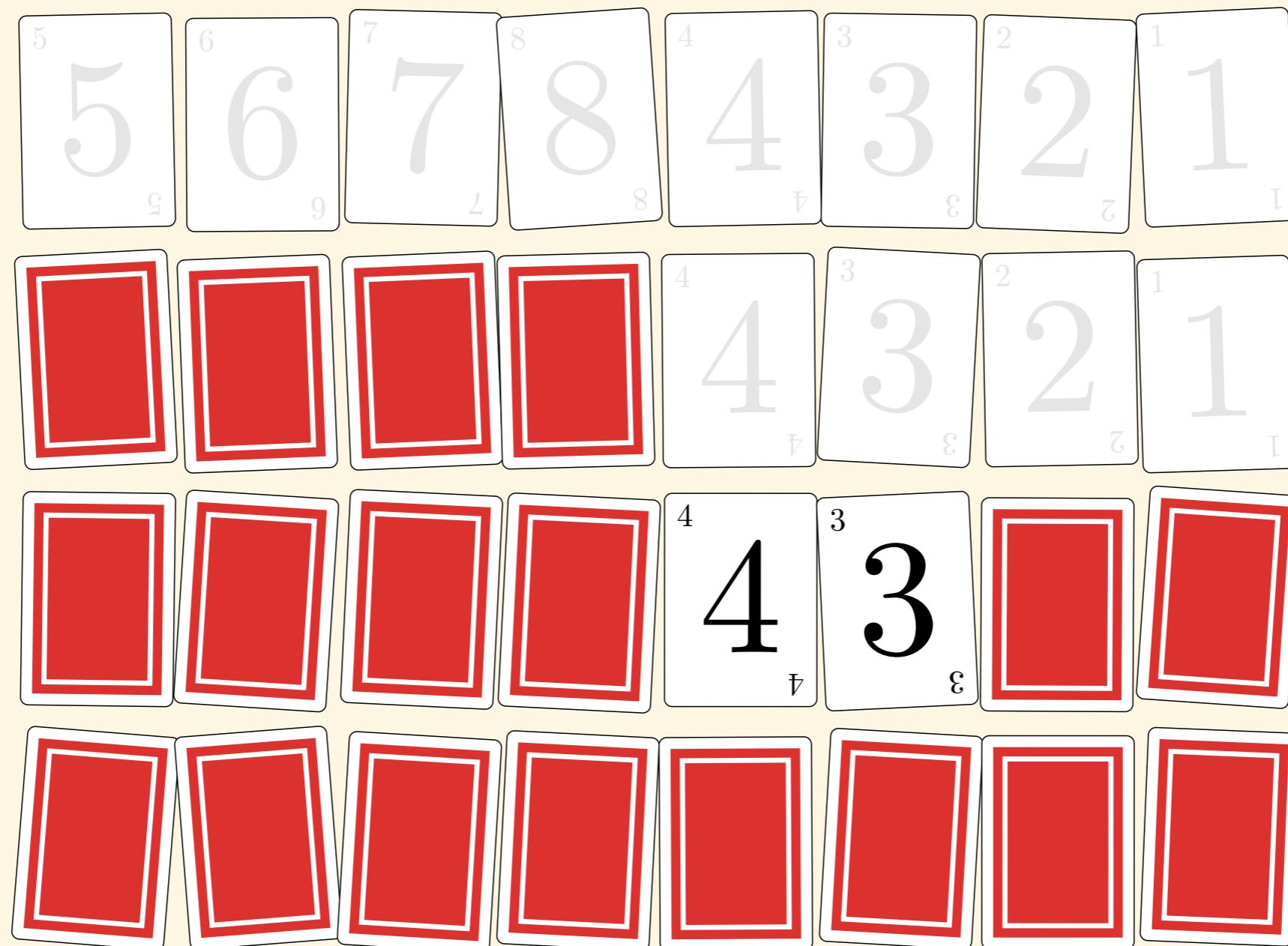
Skal sortere $A[5..6]$



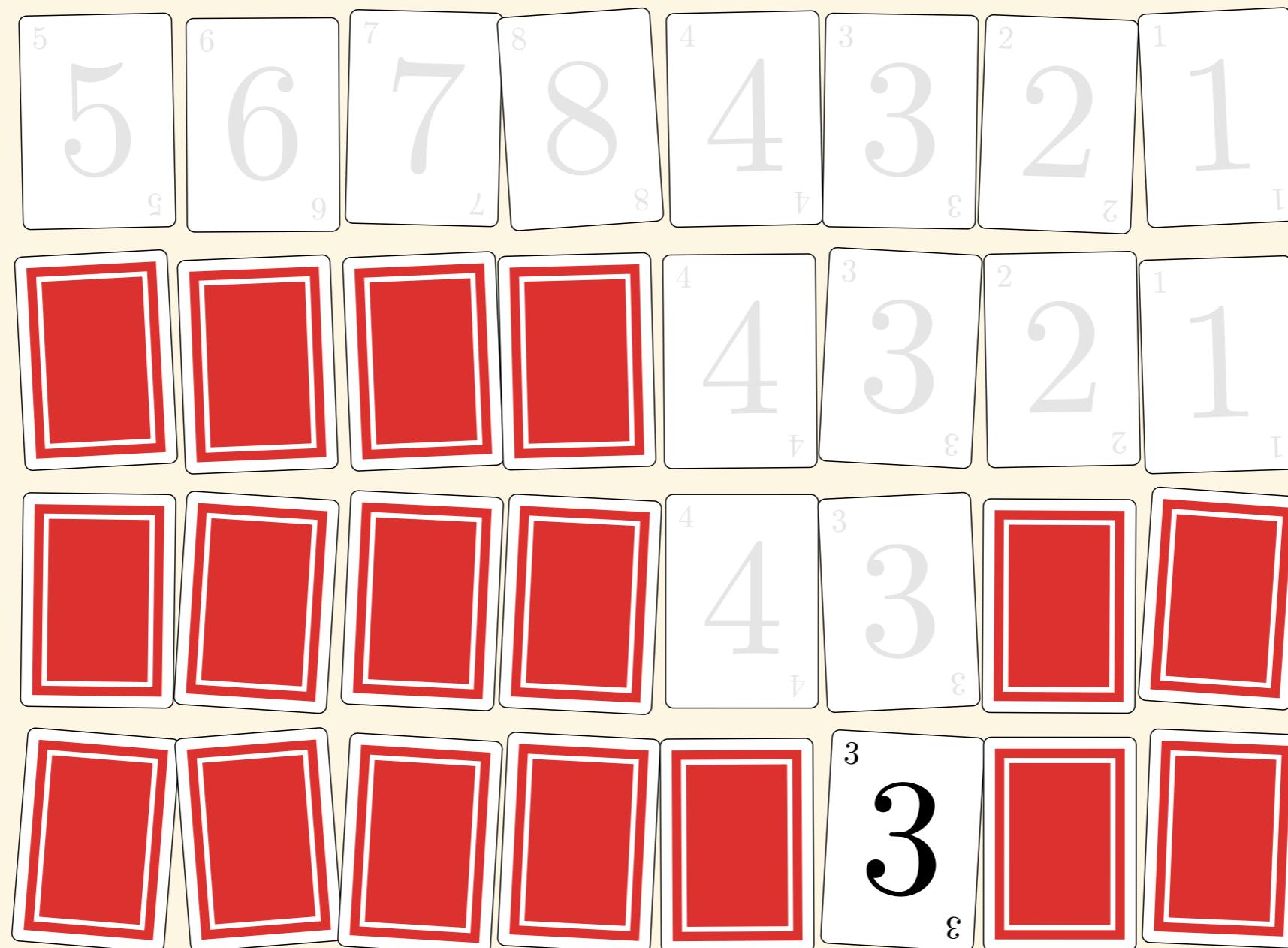
Sortér først $A[5]$ og så $A[6]$



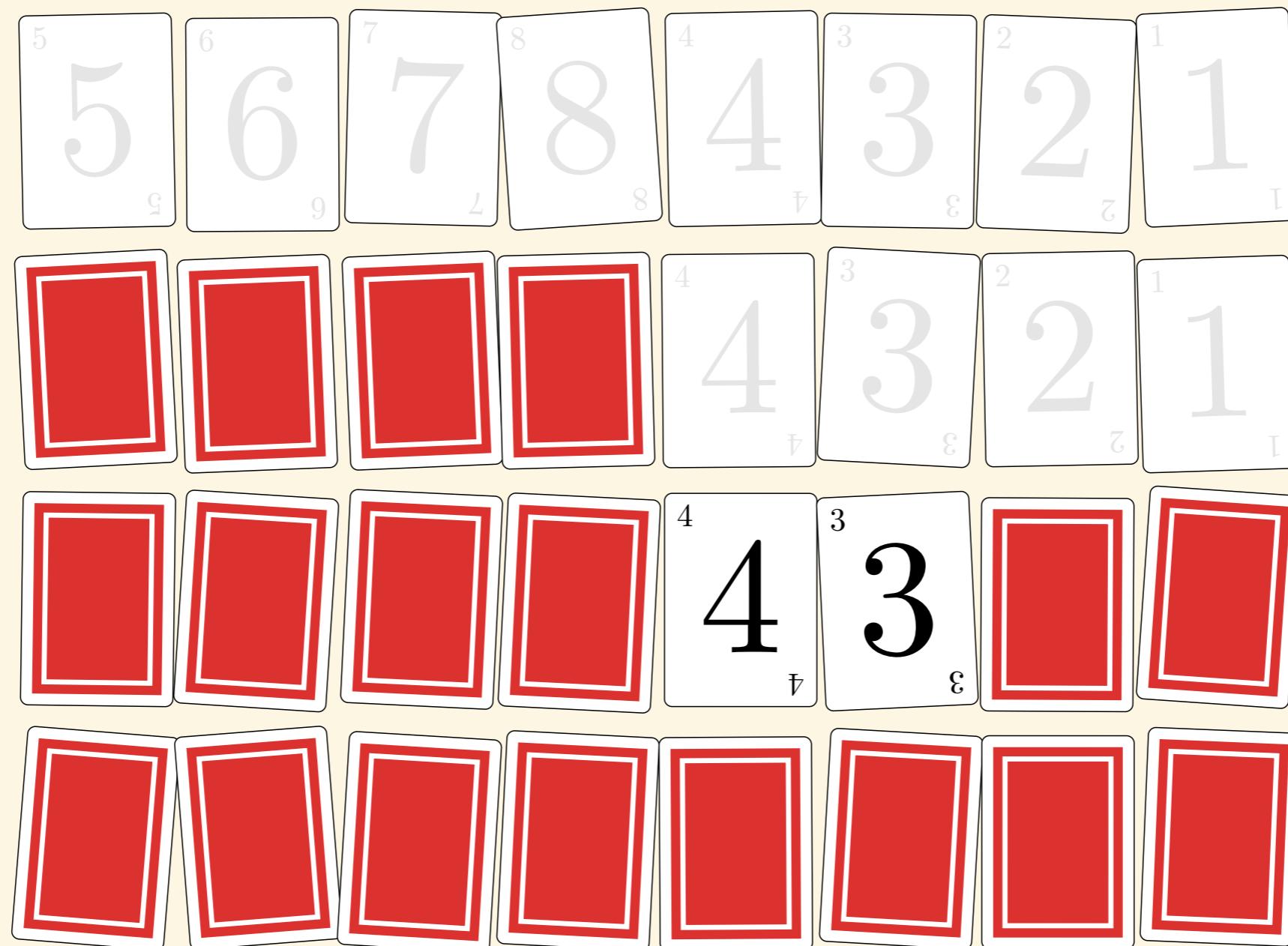
$A[5]$ er sortert



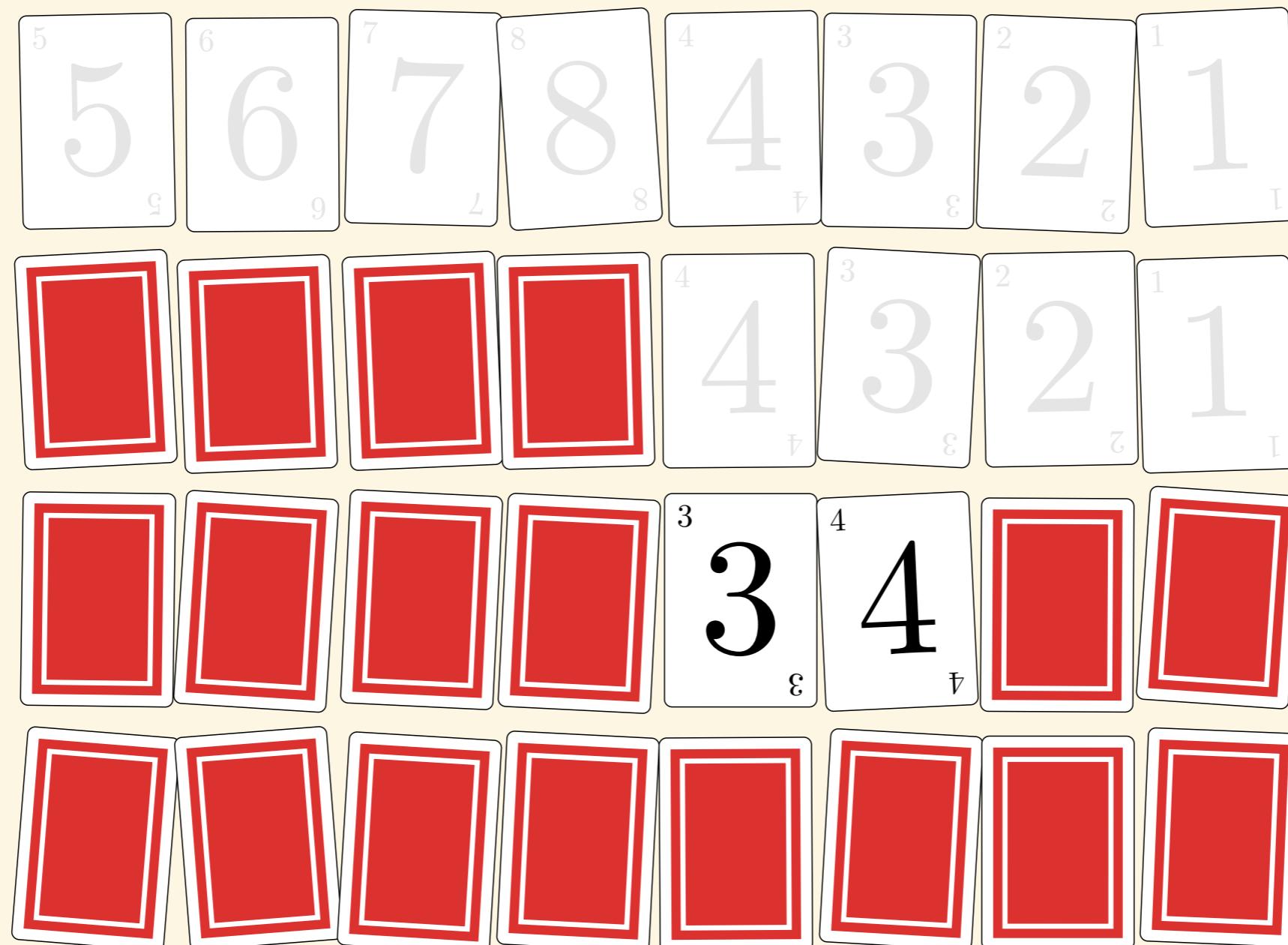
Sortér først $A[5]$ og så $A[6]$



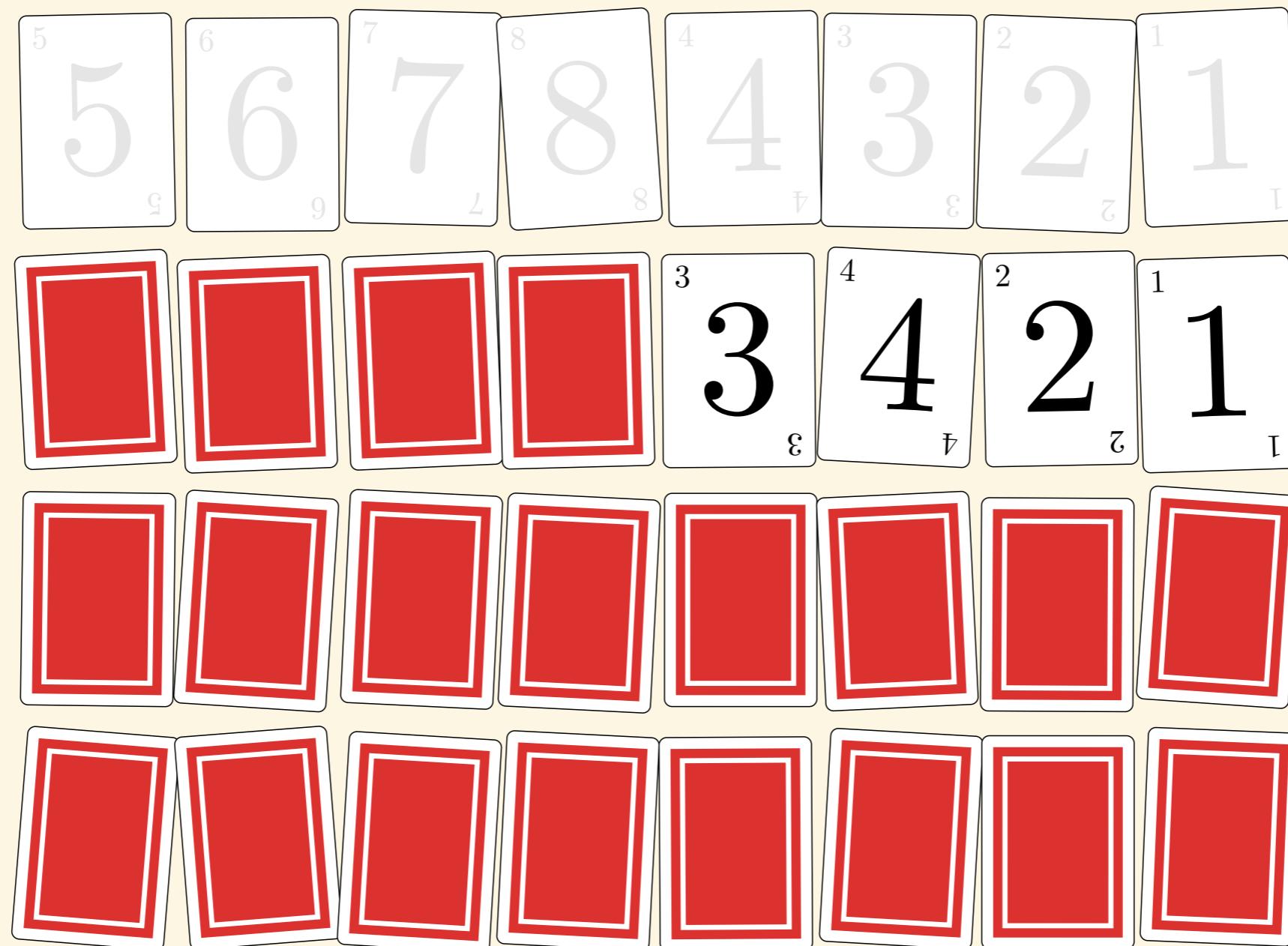
$A[6]$ er sortert



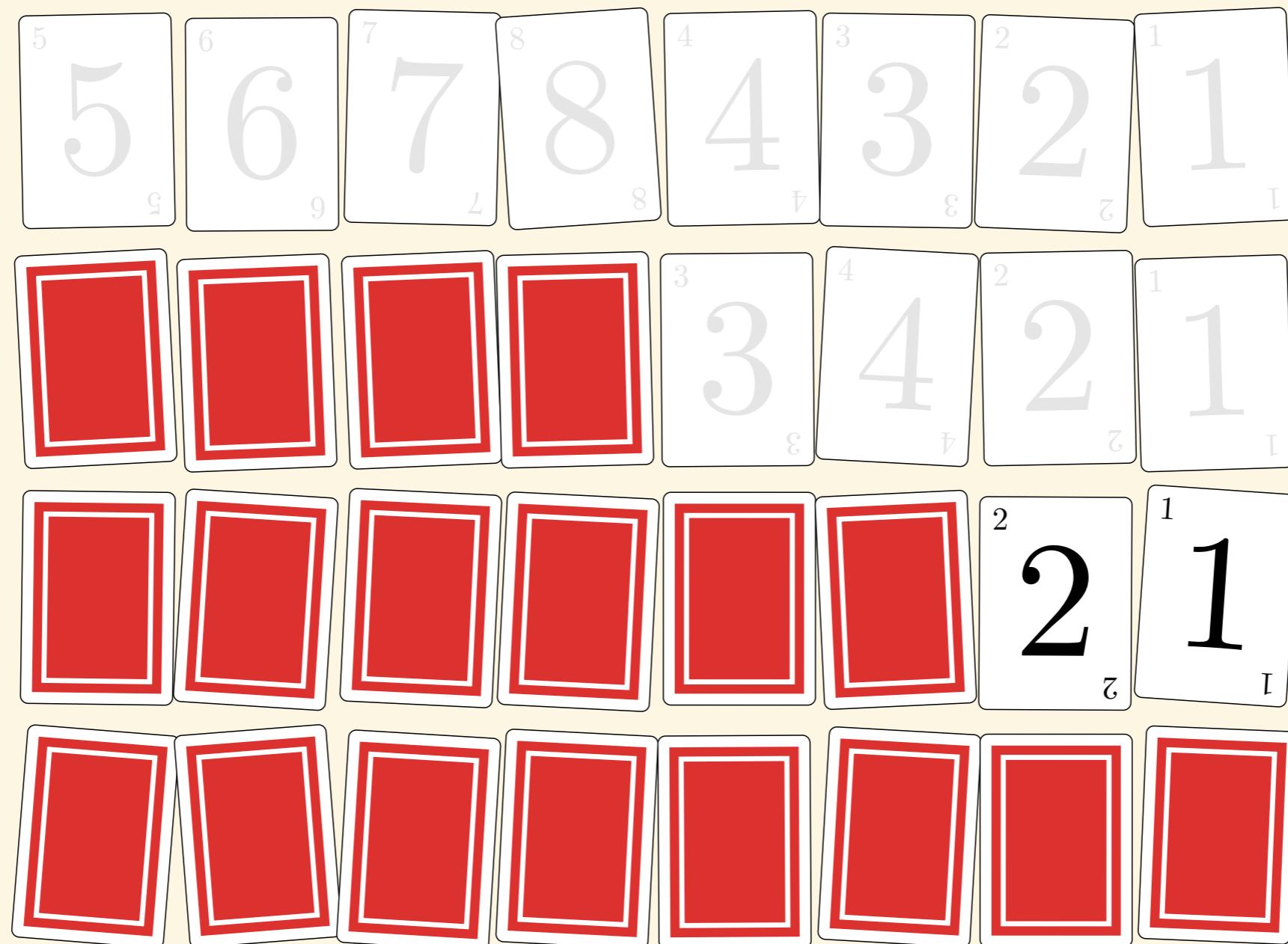
Flett sammen $A[5]$ og $A[6]$



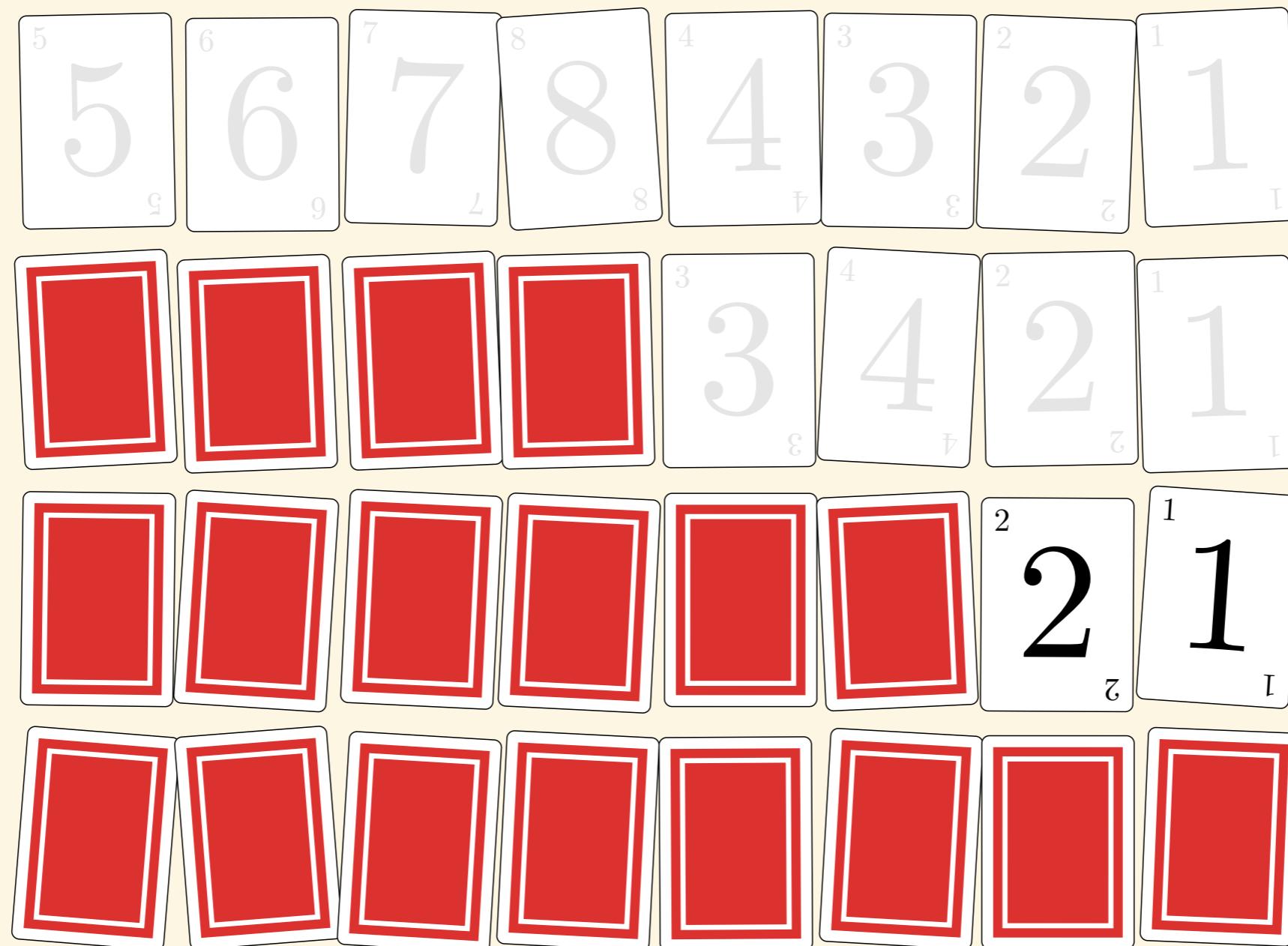
$A[5..6]$ er sortert



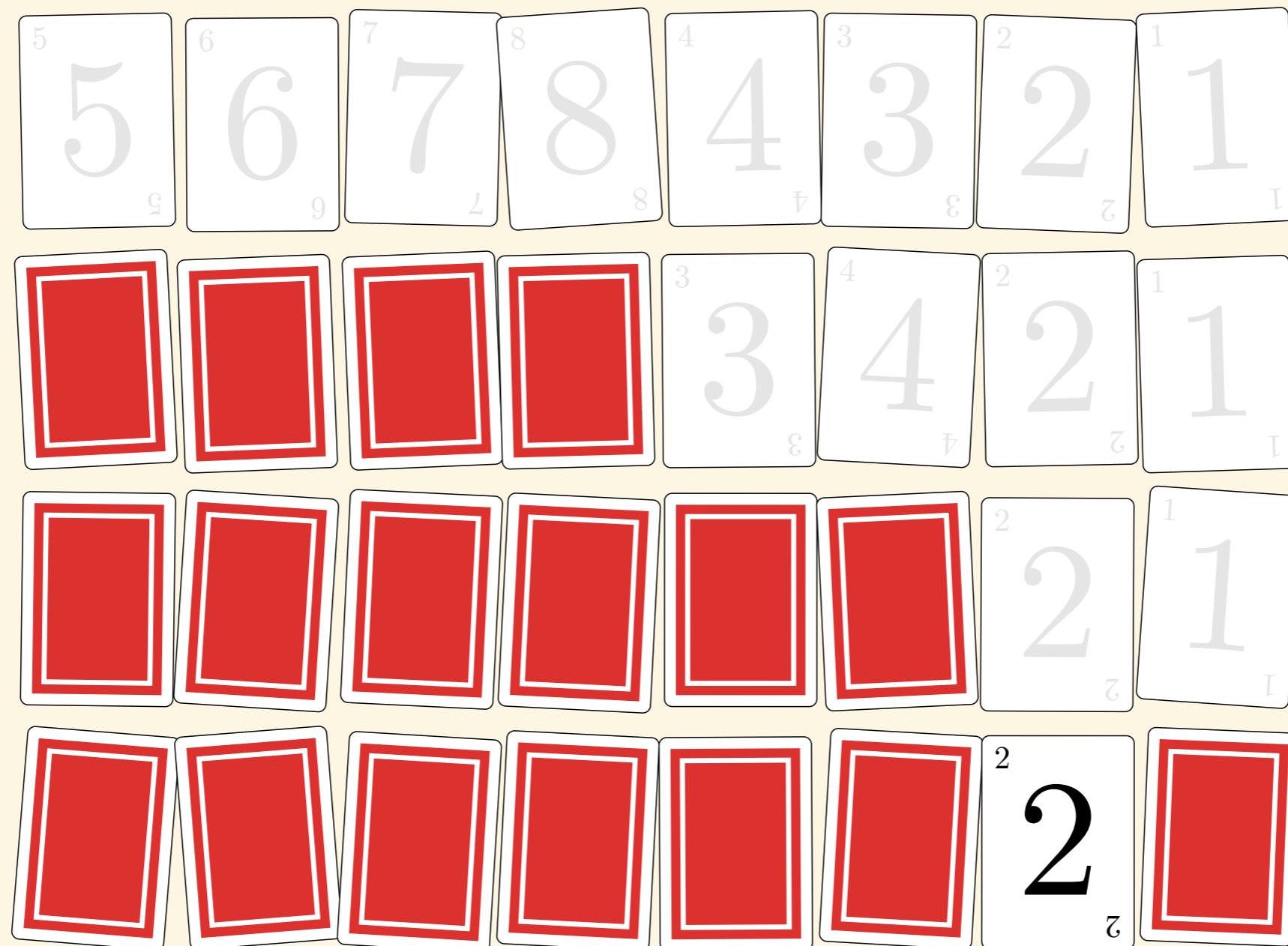
Sortér først $A[5..6]$ og så $A[7..8]$



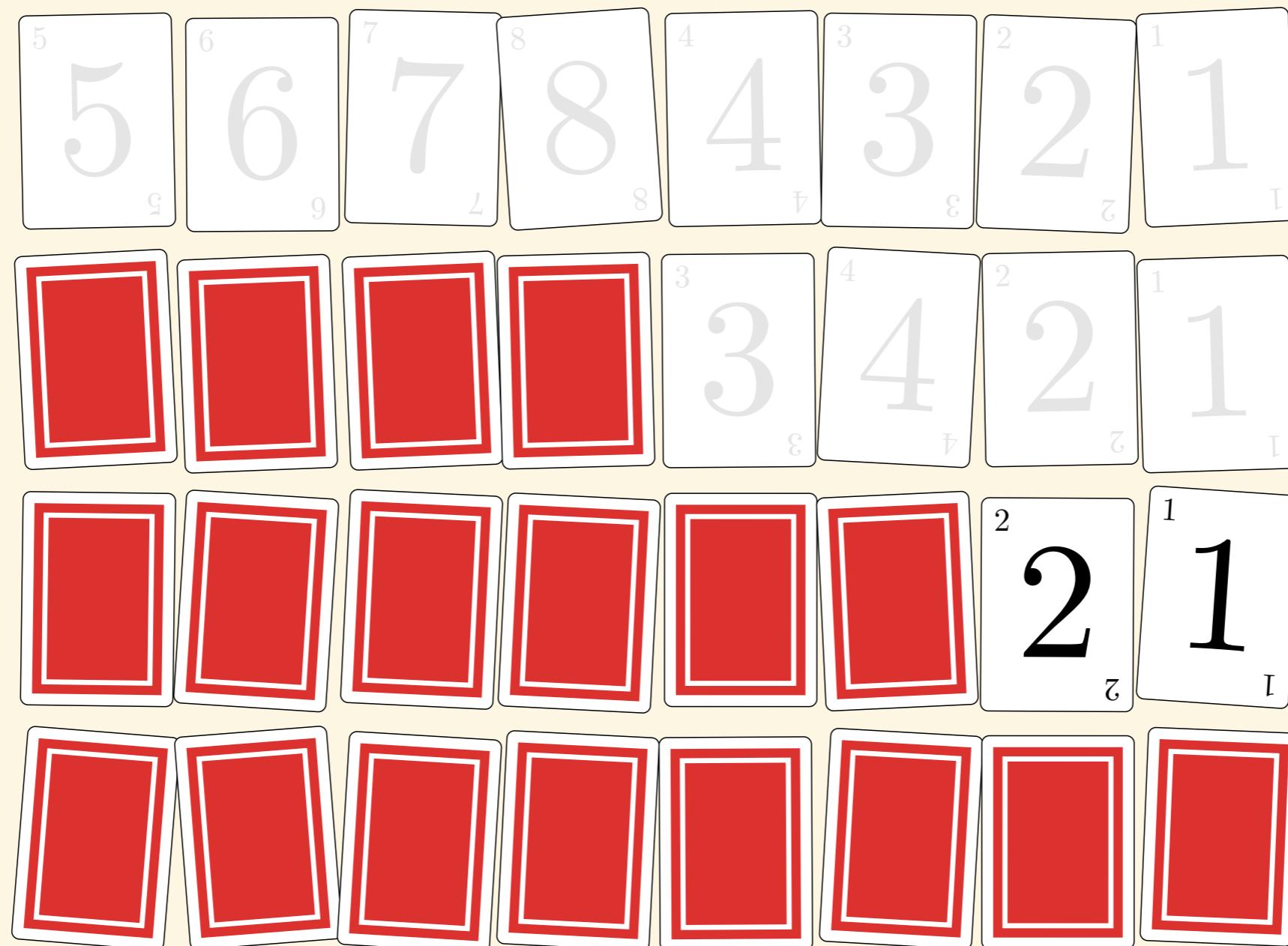
Skal sortere $A[7..8]$



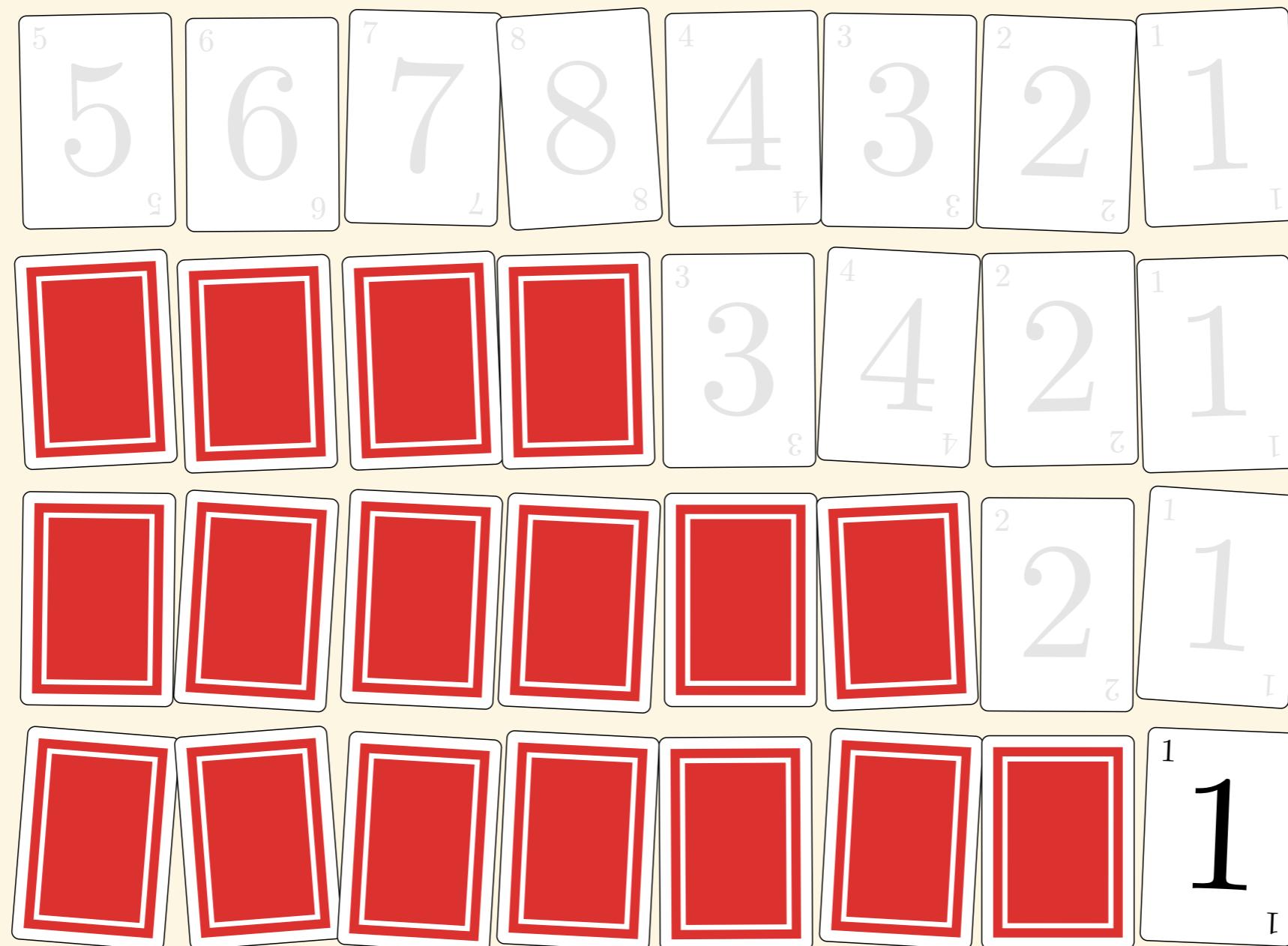
Sortér først $A[7]$ og så $A[8]$



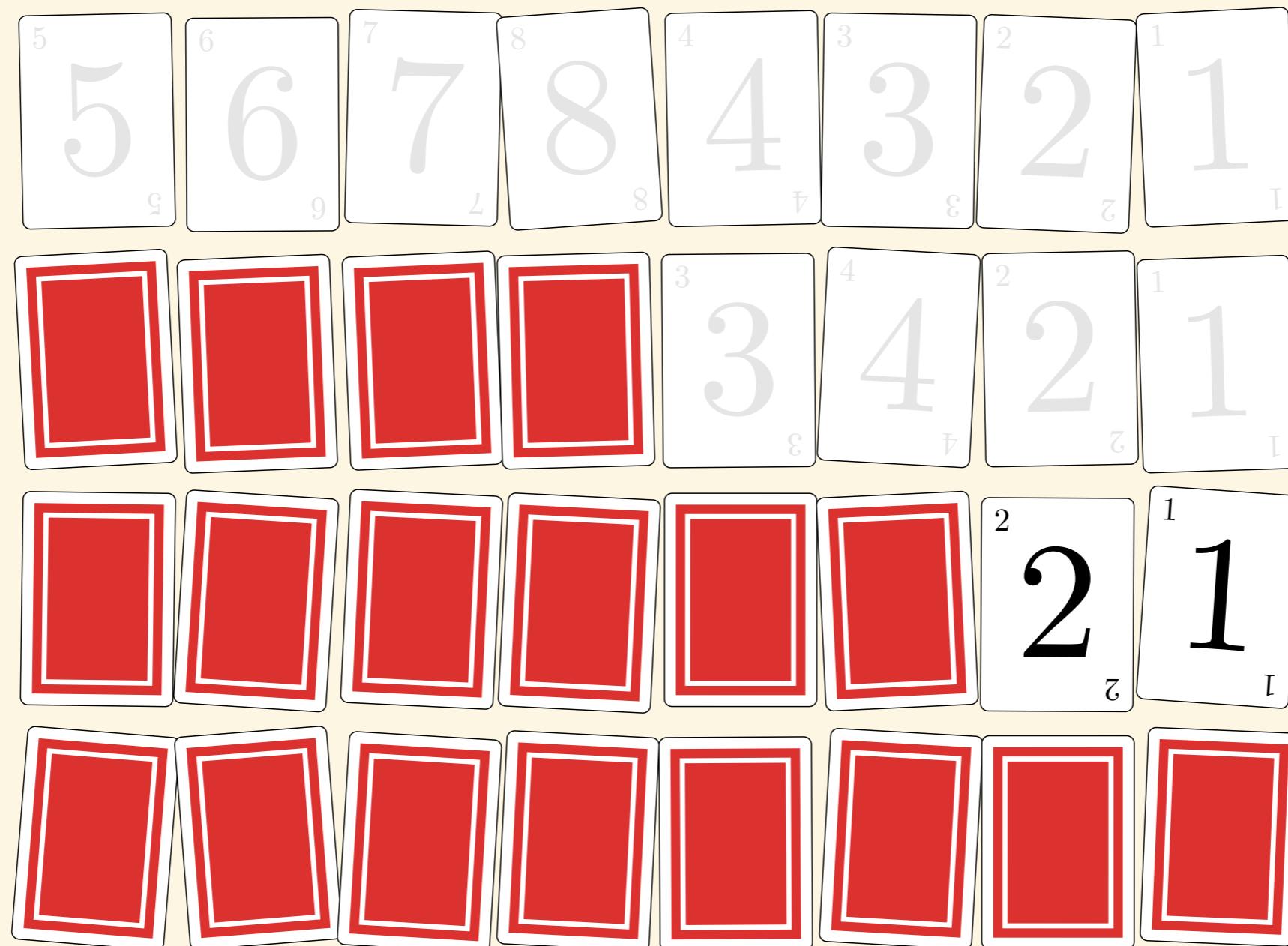
$A[7]$ er sortert



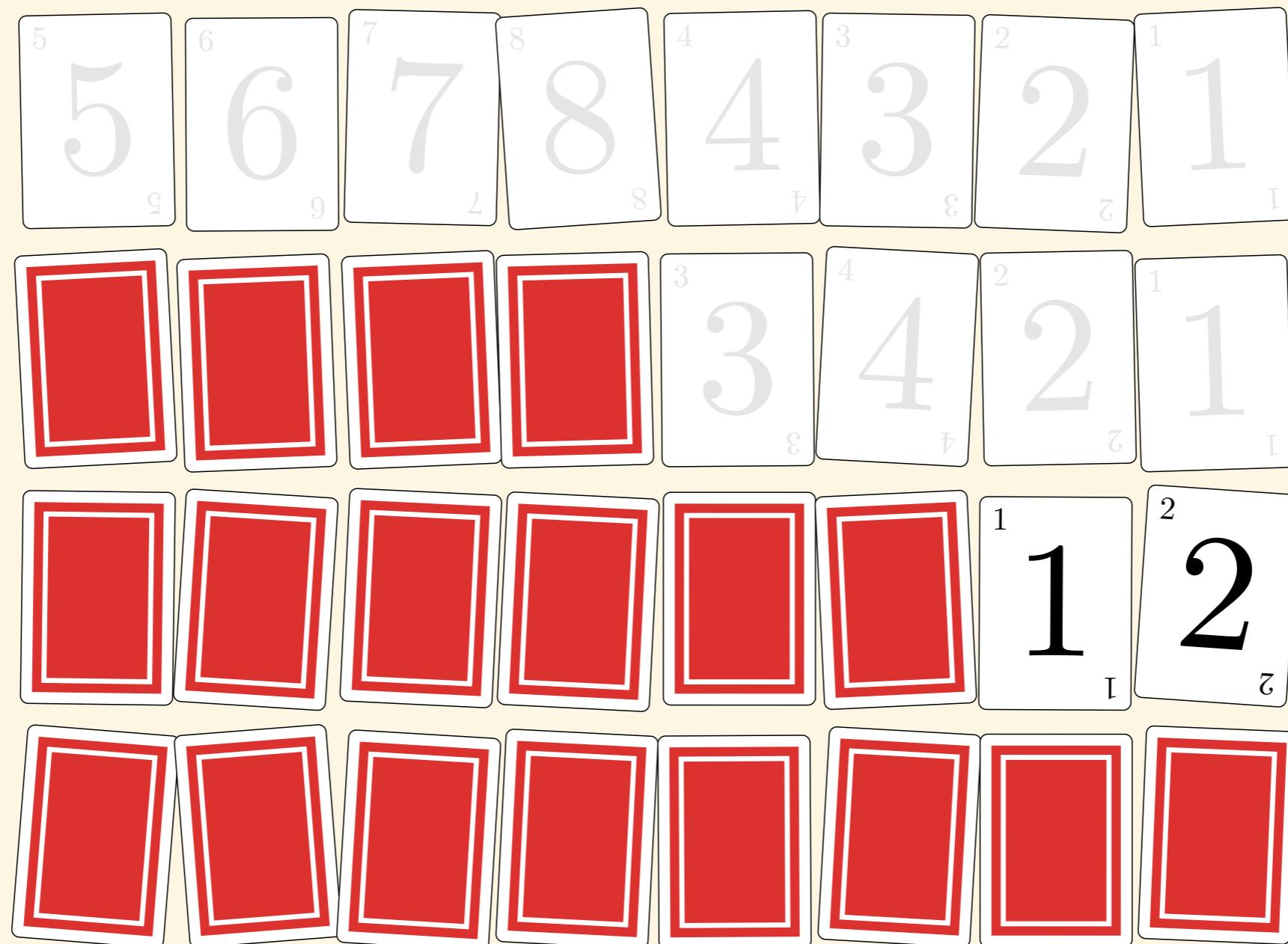
Sortér først $A[7]$ og så $A[8]$



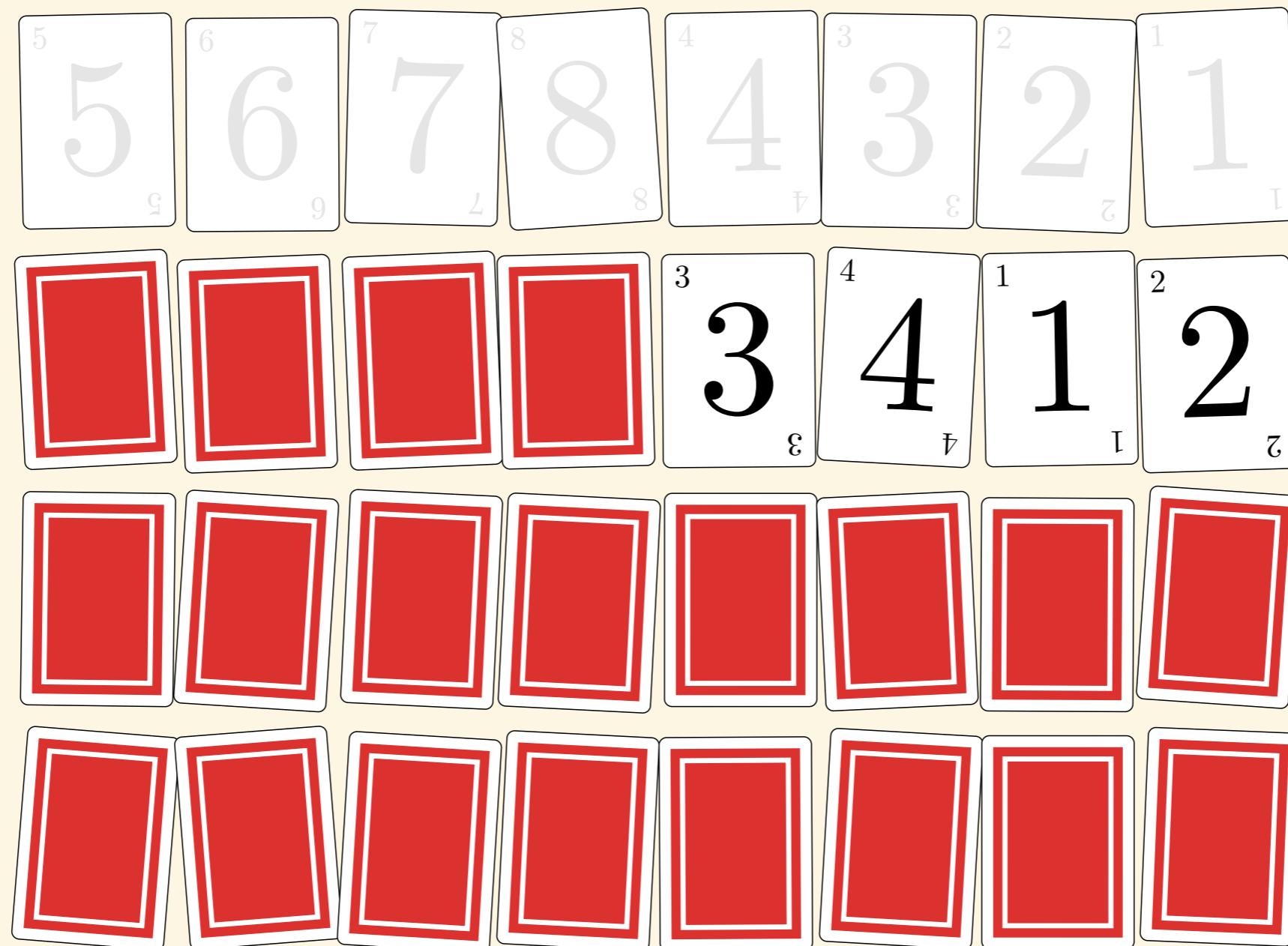
$A[8]$ er sortert



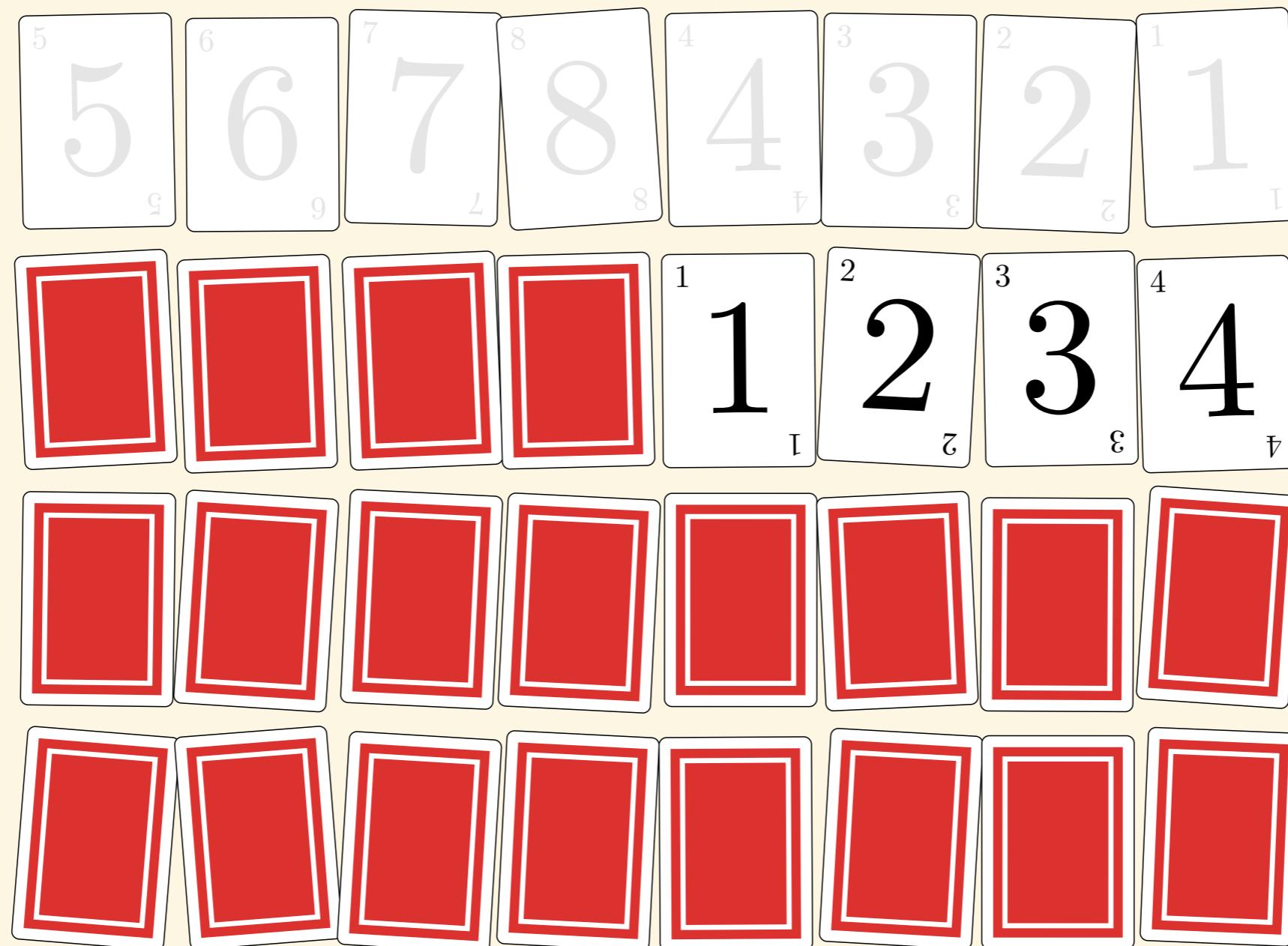
Flett sammen $A[7]$ og $A[8]$



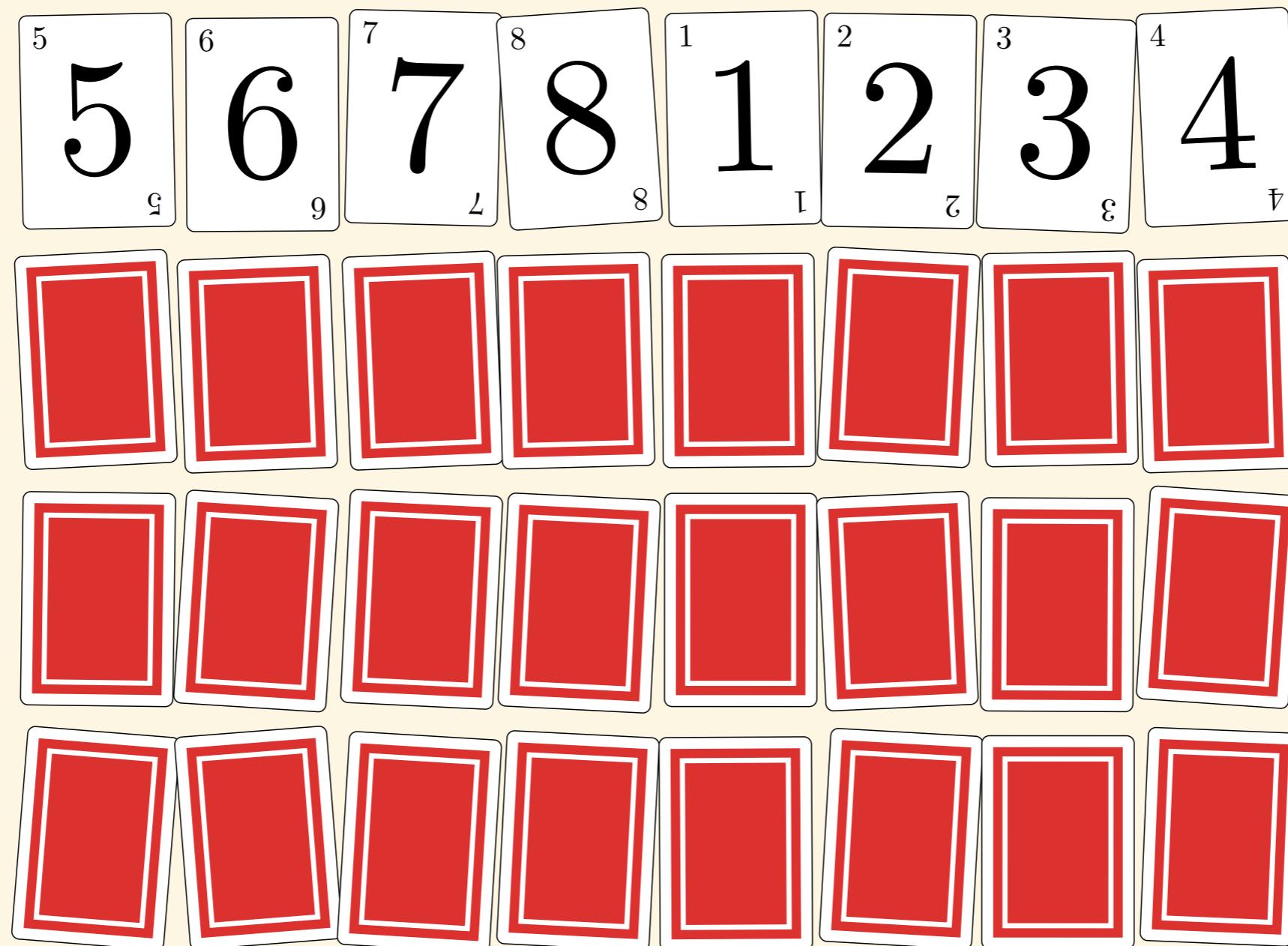
$A[7..8]$ er sortert



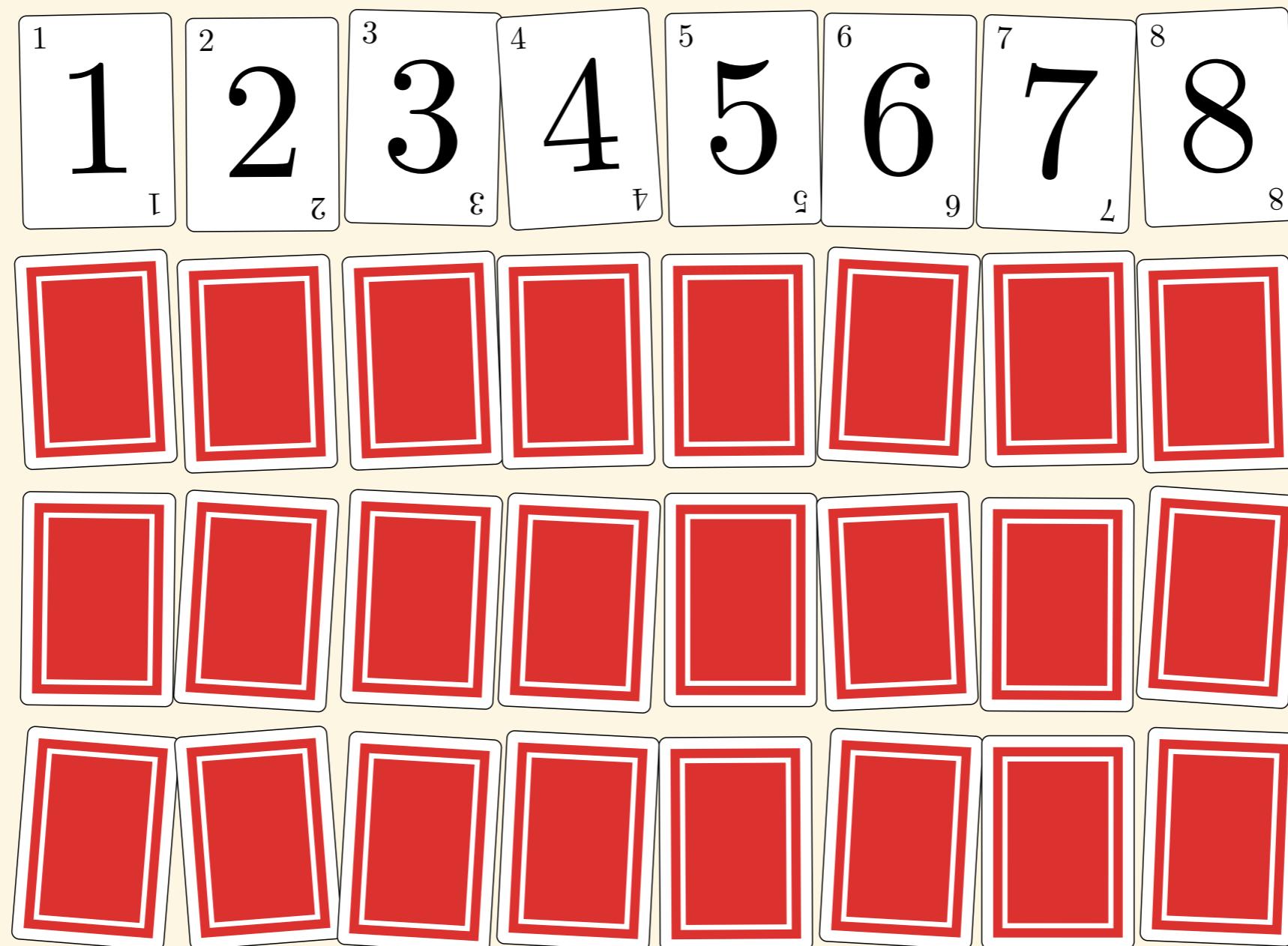
Flett sammen $A[5..6]$ og $A[7..8]$



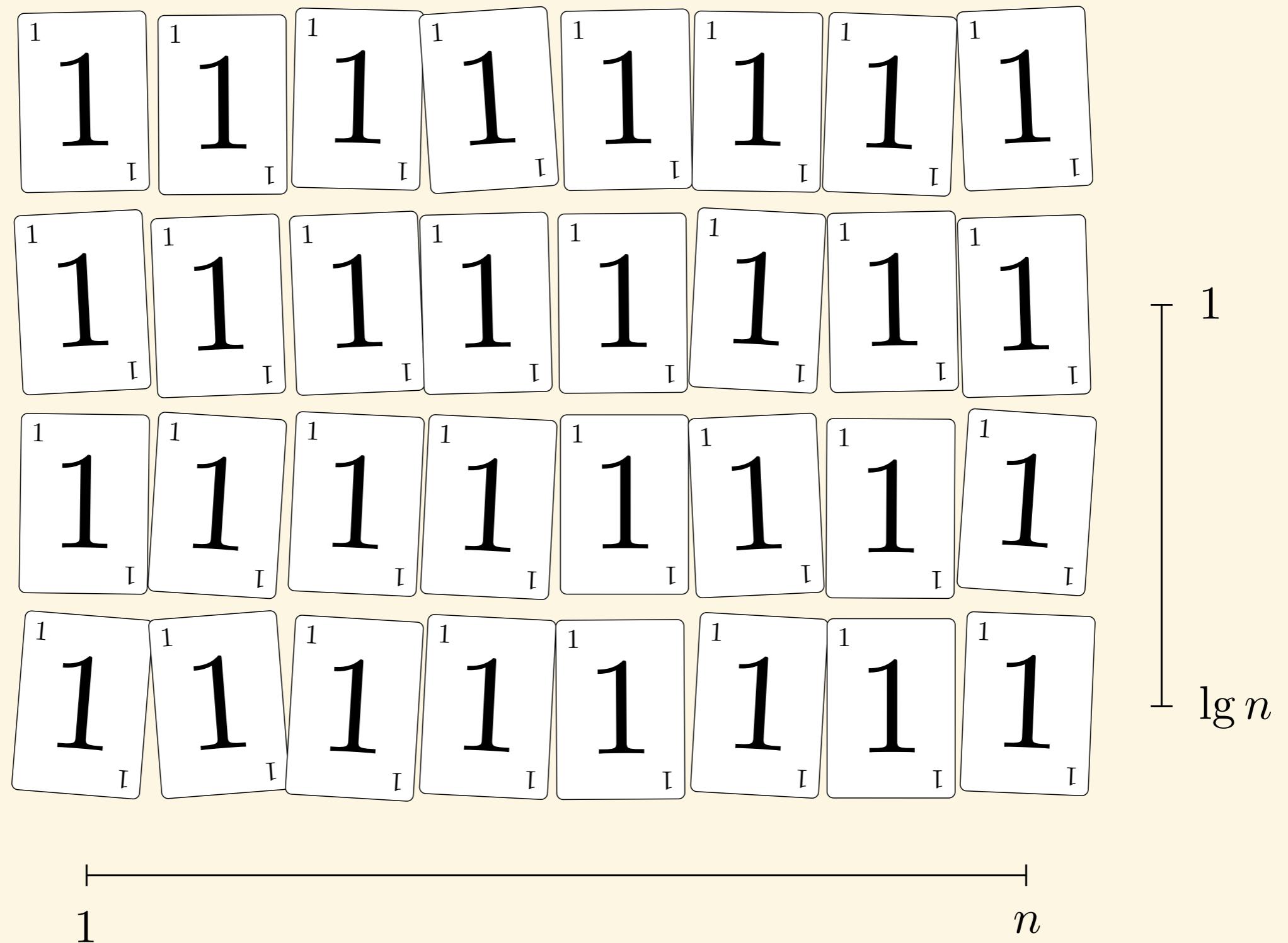
$A[5..8]$ er sortert



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



$A[1..8]$ er sortert



MERGE-SORT(A, p, r)

Sortér $A[p..r]$ 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       $q = \lfloor(p + r)/2\rfloor$ 
```

Del på midten

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

Sortér $A[p..q]$ rekursivt

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

Sortér $A[q + 1..r]$ rekursivt

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

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8$

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

MERGE-SORT(A, p, r)

```

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

```

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

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

MERGE-SORT(A, p, r)

```

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

```

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

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

MERGE-SORT(A, p, r)

```

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

```

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

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2 \rightarrow 1, 1$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2 \rightarrow 2, 2$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$

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

MERGE-SORT(A, p, r)

```

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

```

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

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4$

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

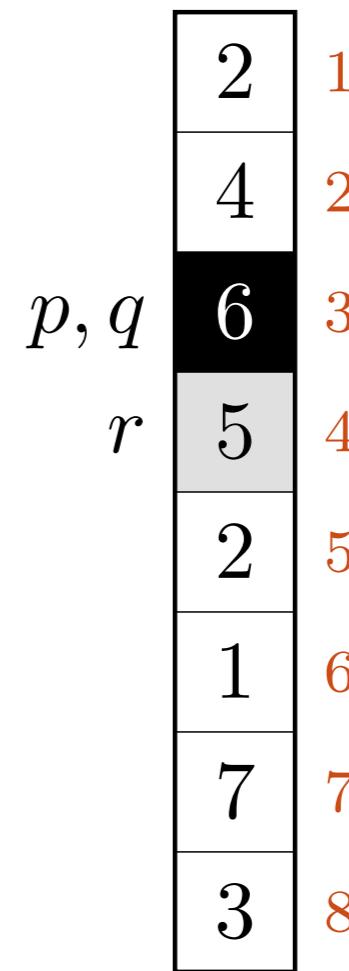
MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4$



MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4 \rightarrow 3, 3$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4$

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

MERGE-SORT(A, p, r)

```

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

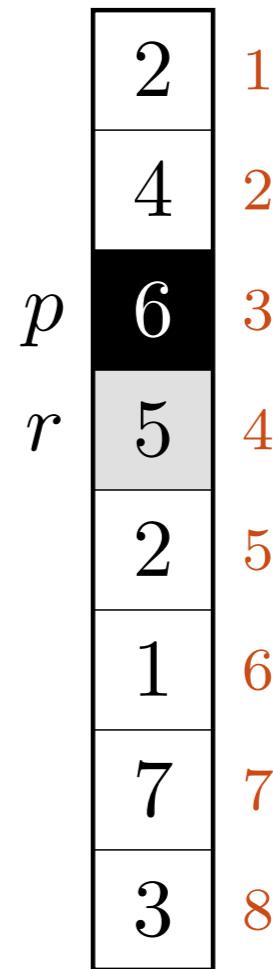
```

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4 \rightarrow 4, 4$

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

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

$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4$



MERGE-SORT(A, p, r)

```

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

```

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

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8$

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

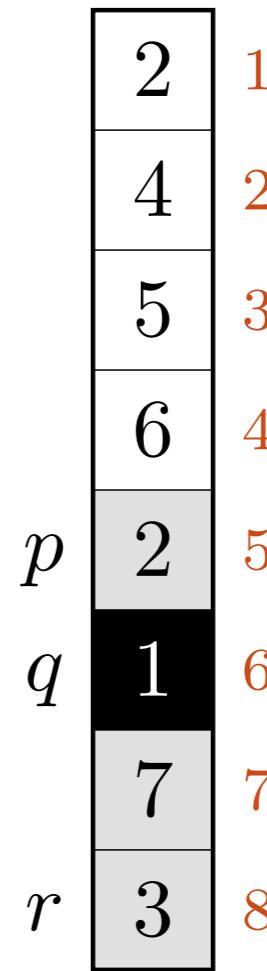
MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8$



MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6 \rightarrow 5, 5$

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

p, r

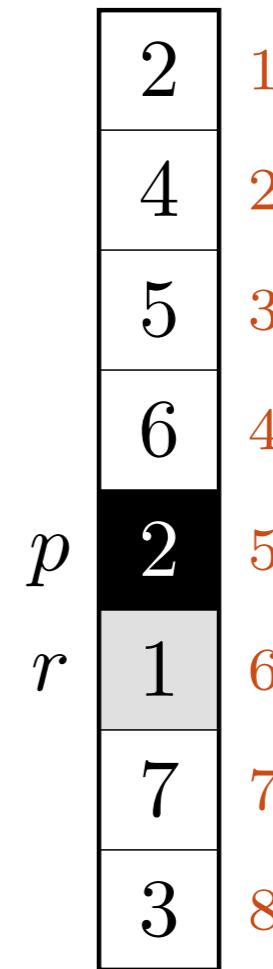
MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6$



MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6 \rightarrow 6, 6$

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

p, r

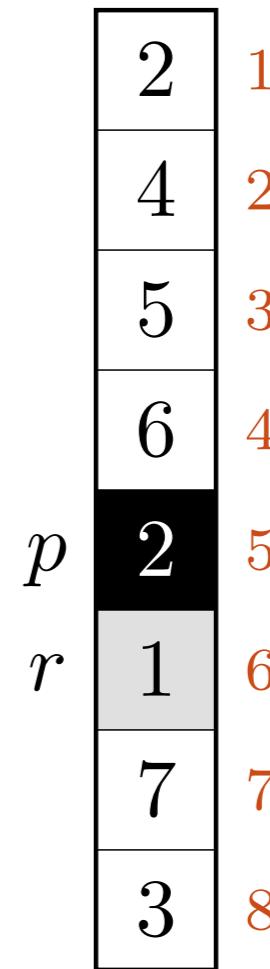
MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6$

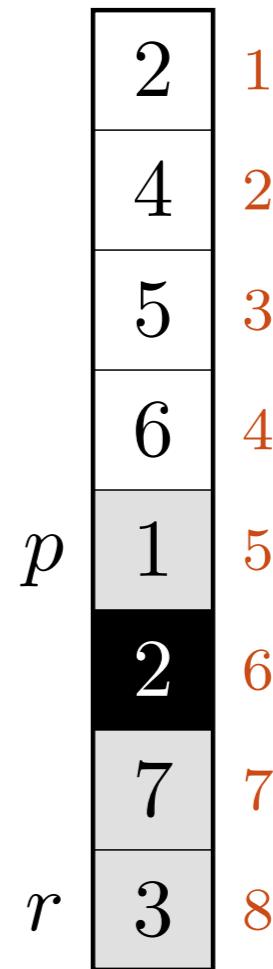


```

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

```

$p, r = 1, 8 \rightarrow 5, 8$



MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8$

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

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8$

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

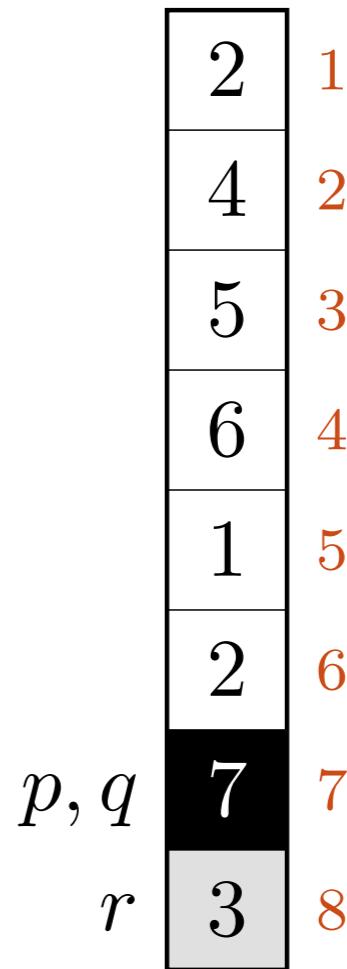
MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8$



MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8 \rightarrow 7, 7$

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

p, r

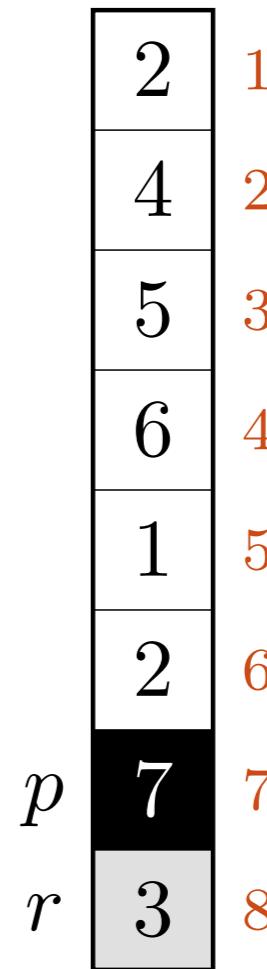
MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8$



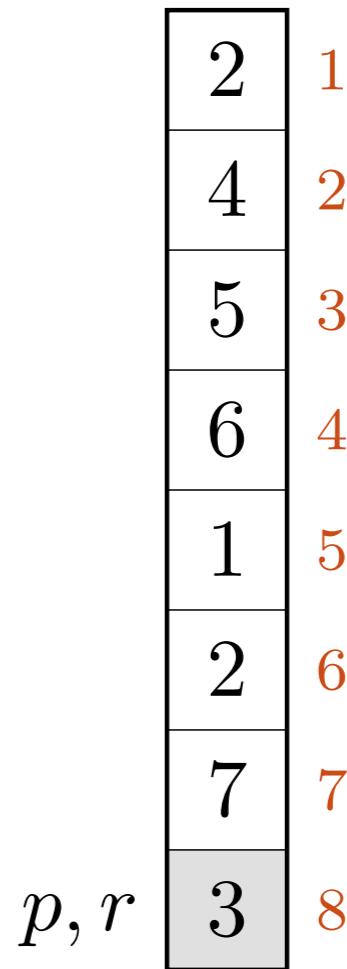
MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8 \rightarrow 8, 8$



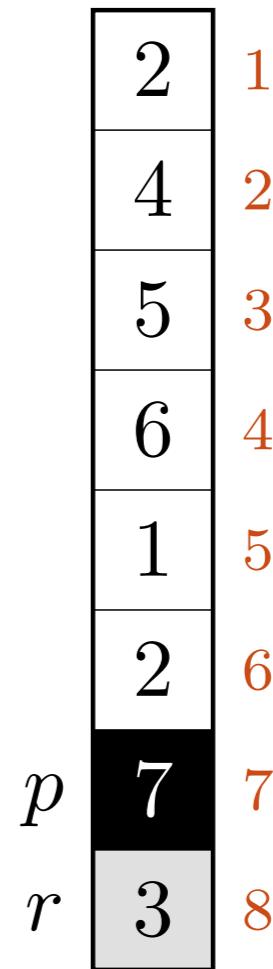
MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8$

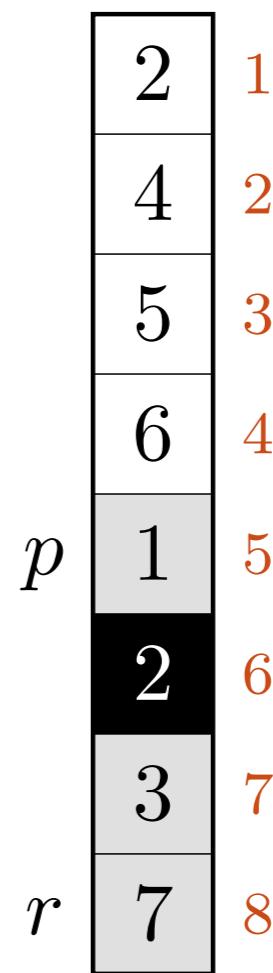


MERGE-SORT(A, p , r)

```

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

```



$$p, r = 1, 8 \rightarrow 5, 8$$

MERGE-SORT(A, p, r)

```

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

```

$p, r = 1, 8$

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

MERGE-SORT(A, p, r)

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

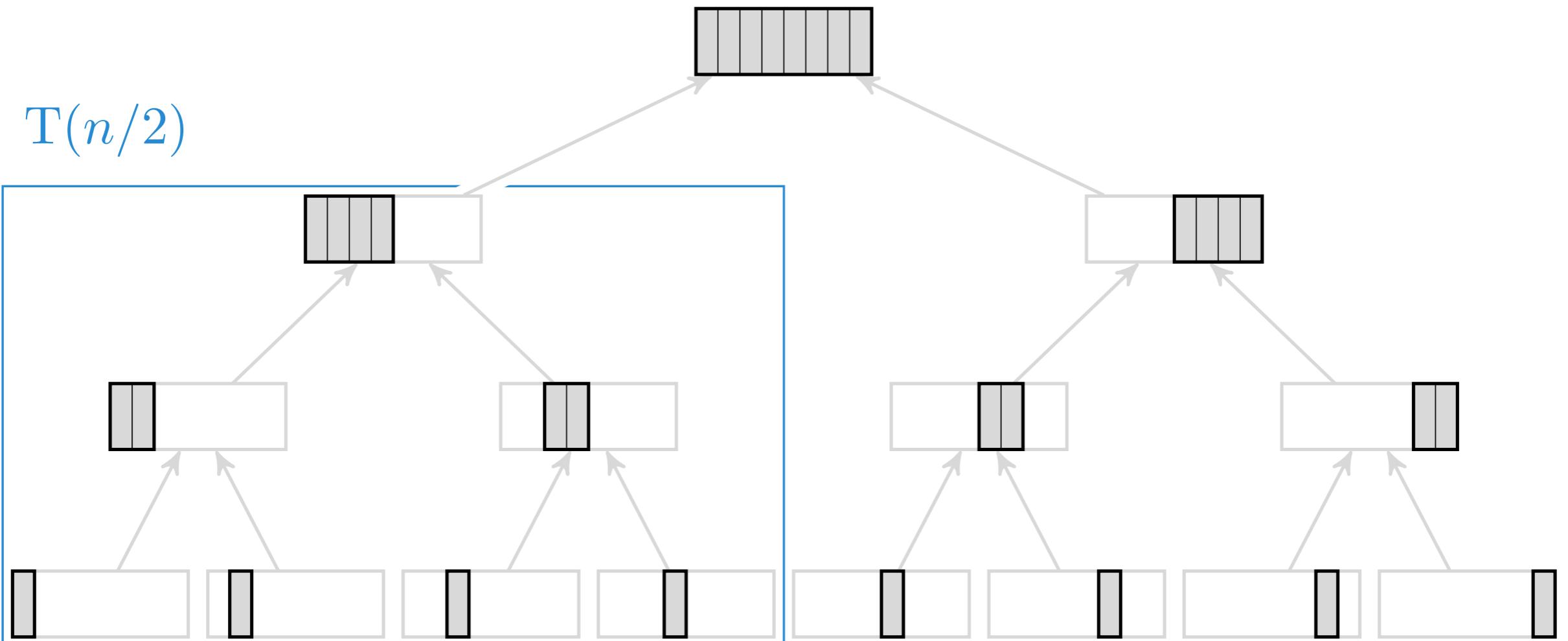
1	1
2	2
2	3
3	4
4	5
5	6
6	7
7	8

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

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

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

D&C \rightarrow merge sort $\rightarrow T(n) = 2T(n/2) + n$



D&C → merge sort → $T(n) = n + 2T(n/2)$

$$T(n) = n$$

$$+ 2T(n/2) \quad (1)$$

D&C → merge sort → $T(\textcolor{blue}{n}) = n + 2T(n/2)$

$$T(n) = n$$

$$+ 2\textcolor{red}{T}(\textcolor{blue}{n}/2) \quad (1)$$

D&C → merge sort → $T(n) = \textcolor{blue}{n} + 2T(\textcolor{blue}{n}/2)$

$$T(n) = n$$

$$+ 2 \cdot \textcolor{blue}{n}/2 \tag{1}$$

$$+ 2 \cdot 2T(\textcolor{blue}{n}/2/2) \tag{2}$$

D&C → merge sort → $T(n) = n + 2T(n/2)$

$$T(n) = n$$

$$+ n \quad (1)$$

$$+ 4T(n/4) \quad (2)$$

D&C → merge sort → $T(n) = n + 2T(n/2)$

$$T(n) = n$$

$$+ n \quad (1)$$

$$+ n \quad (2)$$

$$+ 8T(n/8) \quad (3)$$

D&C → merge sort → $T(n) = n + 2T(n/2)$

$$T(n) = n$$

$$+ n \quad (1)$$

$$+ n \quad (2)$$

$$+ n \quad (3)$$

⋮

⋮

$$+ 2^? \cdot T(n/2^?) \quad (?)$$

D&C → merge sort → $T(n) = n + 2T(n/2)$

$$T(n) = n$$

$$+ n \quad (1)$$

$$+ n \quad (2)$$

$$+ n \quad (3)$$

⋮

⋮

$$+ 2^? \cdot T(n/2^?) \quad (\lg n)$$

D&C → merge sort → $T(n) = n + 2T(n/2)$

$$T(n) = n$$

$$+ n \quad (1)$$

$$+ n \quad (2)$$

$$+ n \quad (3)$$

⋮ ⋮

$$+ 2^{\lg n} \cdot T(n/2^{\lg n}) \quad (\lg n)$$

D&C → merge sort → $T(n) = n + 2T(n/2)$

$$T(n) = n$$

$$+ n \quad (1)$$

$$+ n \quad (2)$$

$$+ n \quad (3)$$

$$\vdots \qquad \vdots$$

$$+ n \cdot T(n/n) \quad (\lg n)$$

D&C → merge sort → $T(n) = n + 2T(n/2)$

$$T(n) = n$$

$$+ n \quad (1)$$

$$+ n \quad (2)$$

$$+ n \quad (3)$$

⋮

⋮

$$+ n \cdot T(1) \quad (\lg n)$$

D&C → merge sort → $T(n) = n + 2T(n/2)$

$$T(n) = n$$

$$+ n \quad (1)$$

$$+ n \quad (2)$$

$$+ n \quad (3)$$

⋮

⋮

$$+ n \quad (\lg n)$$

D&C → merge sort → $T(n) = n + 2T(n/2)$

$$T(n) = n$$

$$+ n \quad (1)$$

$$+ n \quad (2)$$

$$+ n \quad (3)$$

$$\vdots \qquad \vdots$$

$$+ n \quad (\lg n)$$

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

Verifikasjon

Med substitusjon/induksjon

D&C → merge sort → $T(n) = 2T(n/2) + n$

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

D&C → merge sort → $T(n) = 2T(n/2) + n$

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

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

D&C → merge sort → $T(n) = 2T(n/2) + n$

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

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

D&C → merge sort → $T(n) = 2T(n/2) + n$

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

$$= 2\left(\frac{n}{2} \lg \frac{n}{2} + \frac{n}{2}\right) + n$$

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

D&C → merge sort → $T(n) = 2T(n/2) + n$

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

$$= 2\left(\frac{n}{2} \lg \frac{n}{2} + \frac{n}{2}\right) + n$$

$$= n \lg \frac{n}{2} + n + n$$

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

D&C → merge sort → $T(n) = 2T(n/2) + n$

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

$$= 2\left(\frac{n}{2} \lg \frac{n}{2} + \frac{n}{2}\right) + n$$

$$= n \lg \frac{n}{2} + n + n$$

$$= n(\lg n - \lg 2) + 2n$$

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

D&C → merge sort → $T(n) = 2T(n/2) + n$

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

$$= 2\left(\frac{n}{2} \lg \frac{n}{2} + \frac{n}{2}\right) + n$$

$$= n \lg \frac{n}{2} + n + n$$

$$= n(\lg n - \lg 2) + 2n$$

$$= n \lg n + n$$

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

D&C → merge sort → $T(n) = 2T(n/2) + n$

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

$$= 2\left(\frac{n}{2} \lg \frac{n}{2} + \frac{n}{2}\right) + n$$

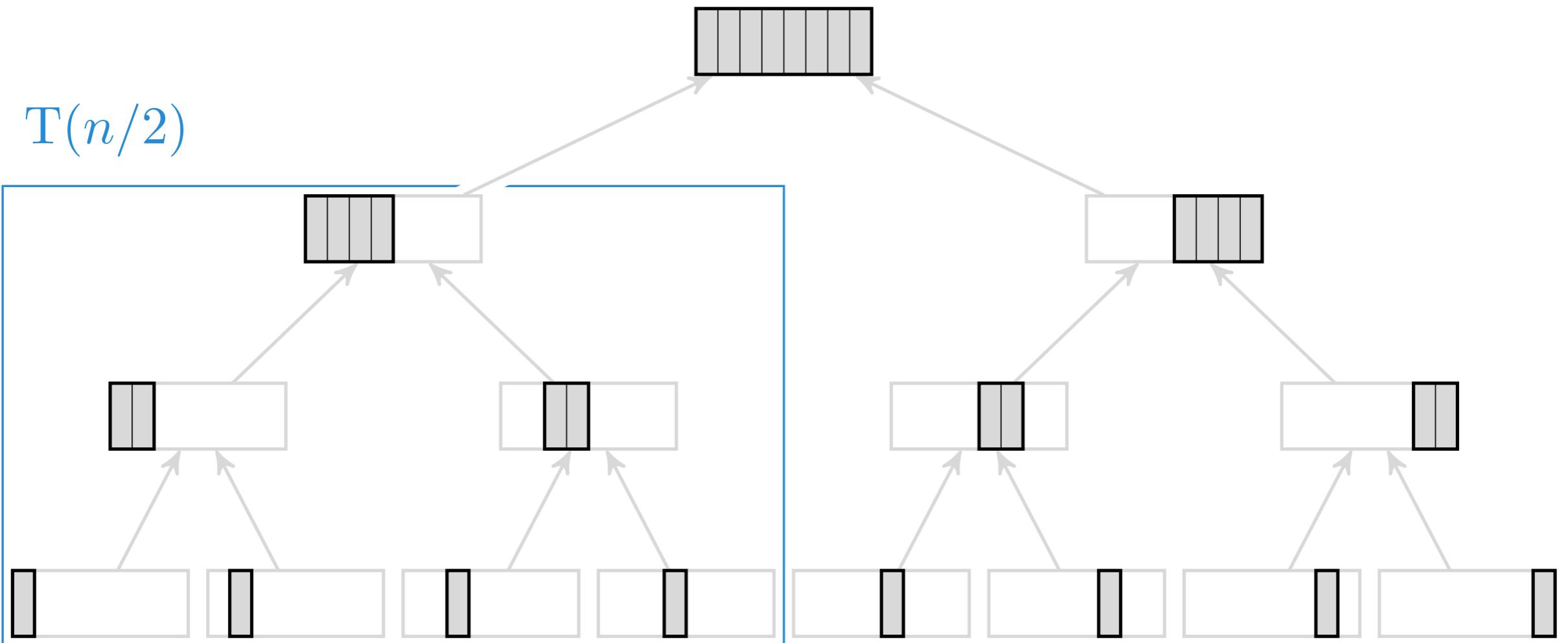
$$= n \lg \frac{n}{2} + n + n$$

$$= n(\lg n - \lg 2) + 2n$$

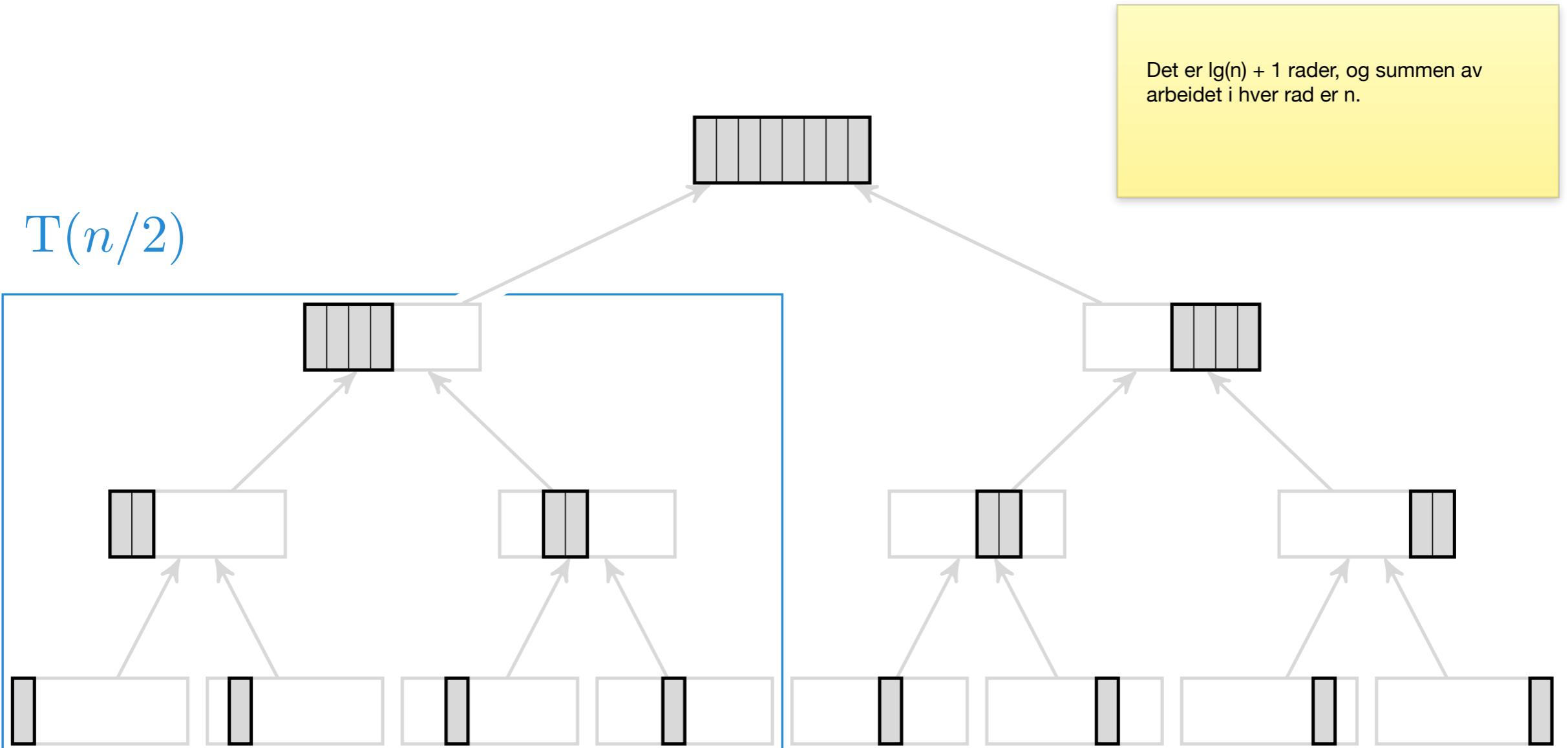
$$= n \lg n + n$$

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

D&C \rightarrow merge sort $\rightarrow T(n) = 2T(n/2) + n$



D&C → merge sort → $T(n) = 2T(n/2) + n$



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

4:6

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.

Part One: Theory

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

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 finds an item with key val-

- 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

PARTITION(A, p, r)

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

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

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

PARTITION(A, p, r)

1 $x = A[r]$

2 $i = p - 1$

Siste indeks i segmentet med små verdier. Foreløpig tomt

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p - 1
3  for j = p to r - 1
```

Siste indeks i segmentet med store verdier

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p - 1
3  for j = p to r - 1
4      if A[j] ≤ x
```

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

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p - 1
3  for j = p to r - 1
4      if A[j] ≤ x
5          i = i + 1
```

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

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p - 1
3  for j = p to r - 1
4      if A[j] ≤ x
5          i = i + 1
6      exchange A[i] with A[j]
```

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

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p - 1
3  for j = p to r - 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
```

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

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p - 1
3  for j = p to r - 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

Returnér hvor splitten havnet

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = -, -$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, -$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  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)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  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)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4      if  $A[j] \leq x$ 
5           $i = i + 1$ 
6          exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  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)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  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	8

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  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	8

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 1$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 1$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 1$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 1$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 1$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 1$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 2$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 2$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 2$

p	2	1
i	1	2
j	3	3
r	4	4
p	2	1
i	1	2
j	3	5
r	4	8

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 2$

p	2	1
i	1	2
j	3	3
r	4	4
p	2	1
i	1	2
j	3	5
r	4	8

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 3$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 3$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 3$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 3$

p	2	1
i	3	3
j	5	5
r	4	8

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 3$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 3$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 3$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

$x, i = 4, 3$

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

PARTITION(A, p, r)

```

1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4    if  $A[j] \leq x$ 
5       $i = i + 1$ 
6      exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

→ 4

$x, i = 4, 3$

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

- 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

$$\text{QUICKSORT}(A, p, r)$$

Sortér $A[p..r]$ ved å sortere små og store verdier hver for seg

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

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

```
QUICKSORT(A, p, r)
1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
```

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

```
QUICKSORT(A, p, r)
1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3       $\text{QUICKSORT}(A, p, q - 1)$ 
```

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

```
QUICKSORT(A, p, r)
1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3       $\text{QUICKSORT}(A, p, q - 1)$ 
4       $\text{QUICKSORT}(A, q + 1, r)$ 
```

Sortér store verdier $A[q + 1..r]$ rekursivt

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, -, 8$

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, -, 8$

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

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

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, -, 6$

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, -, 6$

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

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

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

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

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3       $\text{QUICKSORT}(A, p, q - 1)$ 
4       $\text{QUICKSORT}(A, q + 1, r)$ 
```

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

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3       $\text{QUICKSORT}(A, p, q - 1)$ 
4       $\text{QUICKSORT}(A, q + 1, r)$ 
```

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

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 1, 1, 2 \rightarrow 1, -, 0$

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 1, 1, 2 \rightarrow 1, -, 0$

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3       $\text{QUICKSORT}(A, p, q - 1)$ 
4       $\text{QUICKSORT}(A, q + 1, r)$ 
```

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

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

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

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

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

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3       $\text{QUICKSORT}(A, p, q - 1)$ 
4       $\text{QUICKSORT}(A, q + 1, r)$ 
```

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

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

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

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 4, -, 6$

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 4, -, 6$

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

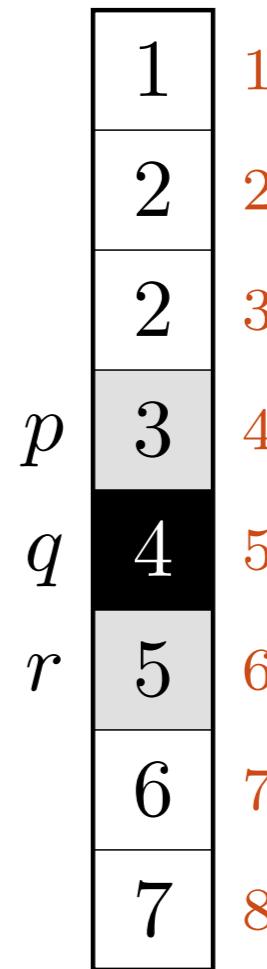
QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 4, 5, 6$



QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 4, 5, 6 \rightarrow 4, -, 4$

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

p, r

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 4, 5, 6 \rightarrow 4, -, 4$

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

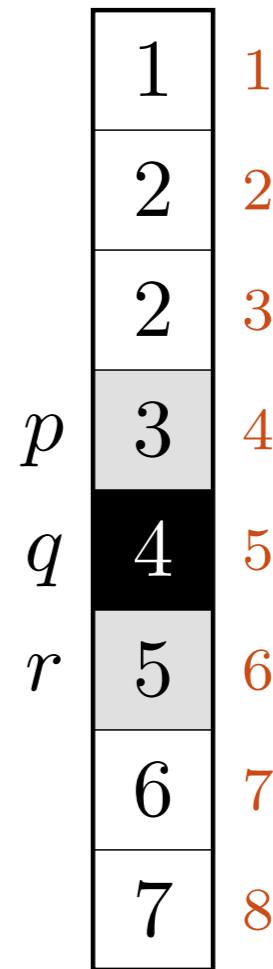
QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 4, 5, 6$



QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 4, 5, 6 \rightarrow 6, -, 6$

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

p, r

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 4, 5, 6 \rightarrow 6, -, 6$

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

p, r

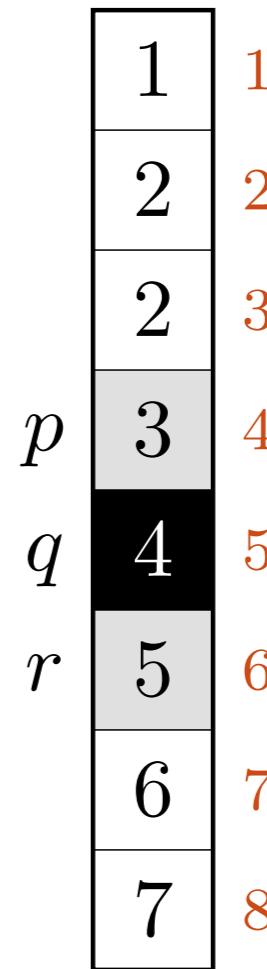
QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 4, 5, 6$



QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

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

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

QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

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

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

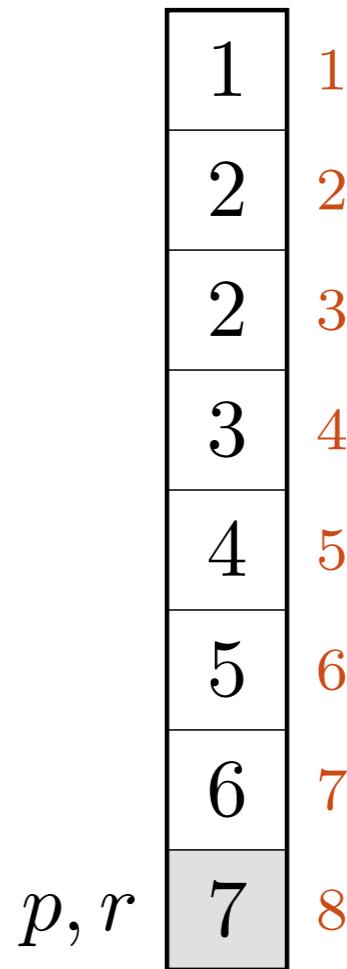
QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 8, -, 8$



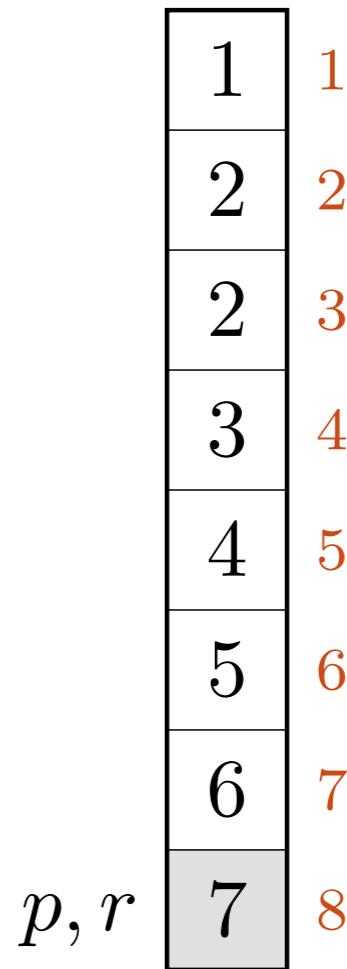
QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

$p, q, r = 1, 7, 8 \rightarrow 8, -, 8$



QUICKSORT(A, p, r)

```

1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )

```

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

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

QUICKSORT(A, p, r)

```
1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      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

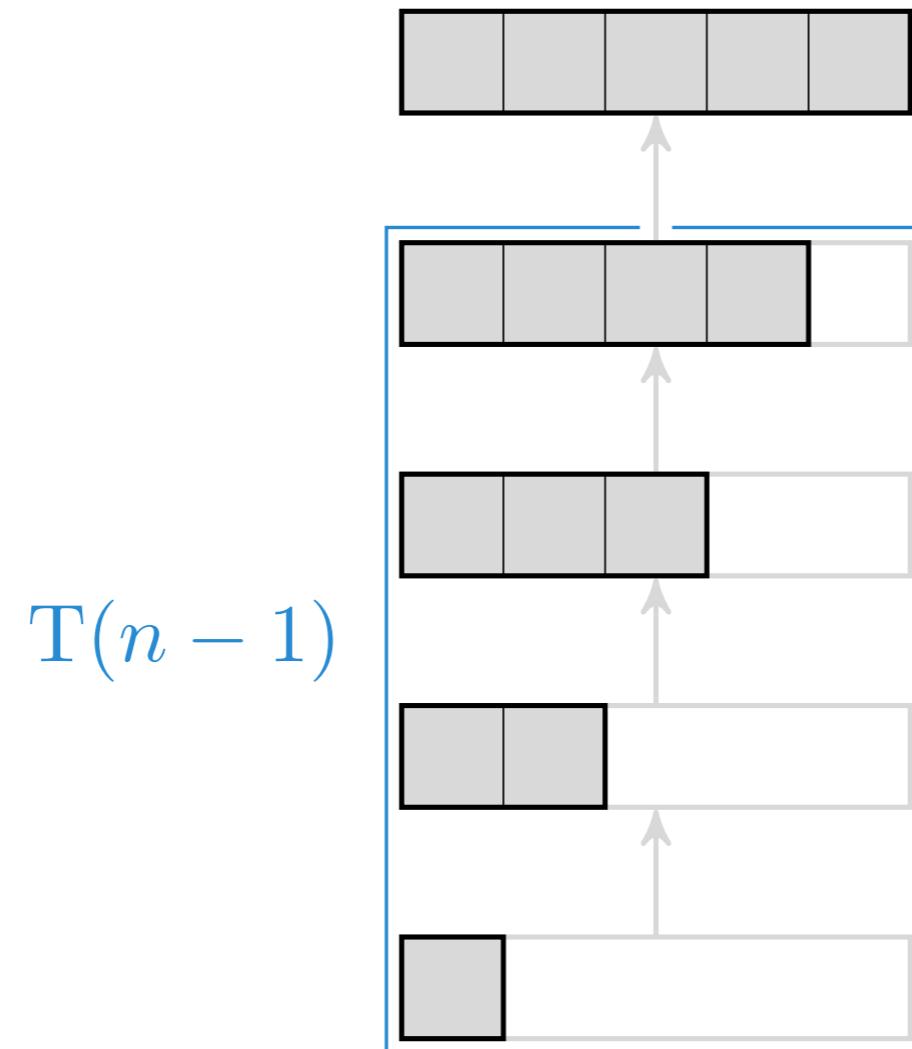
I verste tilfelle
Pivot blir helt alene

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

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

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

$$\text{D\&C} \rightarrow \text{quicksort} \rightarrow \text{WC} \rightarrow T(n) = T(n - 1) + n$$



D&C → quicksort → WC → $T(n) = n + T(n - 1)$

$$T(n) = n$$

$$+ T(n - 1) \quad (1)$$

D&C ↗ quicksort ↗ WC ↗ $T(\textcolor{blue}{n}) = n + T(n - 1)$

$$T(n) = n$$

$$+ T(\textcolor{blue}{n} - 1) \quad (1)$$

D&C → quicksort → WC → $T(n) = \textcolor{blue}{n} + T(\textcolor{red}{n} - 1)$

$$T(n) = n$$

$$+ \textcolor{blue}{n} - 1 \tag{1}$$

$$+ T(\textcolor{blue}{n} - 1 - 1) \tag{2}$$

D&C → quicksort → WC → $T(n) = n + T(n - 1)$

$$T(n) = n$$

$$+ n - 1 \quad (1)$$

$$+ T(n - 2) \quad (2)$$

D&C → quicksort → WC → $T(n) = n + T(n - 1)$

$$T(n) = n$$

$$+ n - 1 \quad (1)$$

$$+ n - 2 \quad (2)$$

$$+ T(n - 3) \quad (3)$$

$$T(n) = n$$

$$+ n - 1 \quad (1)$$

$$+ n - 2 \quad (2)$$

$$+ n - 3 \quad (3)$$

⋮

⋮

$$+ T(n - ?) \quad (?)$$

D&C → quicksort → WC → $T(n) = n + T(n - 1)$

$$T(n) = n$$

$$+ n - 1 \quad (1)$$

$$+ n - 2 \quad (2)$$

$$+ n - 3 \quad (3)$$

⋮ ⋮

$$+ T(n - ?) \quad (n - 1)$$

D&C → quicksort → WC → $T(n) = n + T(n - 1)$

$$T(n) = n$$

$$+ n - 1 \quad (1)$$

$$+ n - 2 \quad (2)$$

$$+ n - 3 \quad (3)$$

⋮ ⋮

$$+ T(n - (n - 1)) \quad (n - 1)$$

D&C → quicksort → WC → $T(n) = n + T(n - 1)$

$$T(n) = n$$

$$+ n - 1 \quad (1)$$

$$+ n - 2 \quad (2)$$

$$+ n - 3 \quad (3)$$

⋮

⋮

$$+ T(1) \quad (n - 1)$$

D&C → quicksort → WC → $T(n) = n + T(n - 1)$

$$T(n) = n$$

$$+ n - 1 \quad (1)$$

$$+ n - 2 \quad (2)$$

$$+ n - 3 \quad (3)$$

⋮ ⋮

$$+ 1 \quad (n - 1)$$

D&C → quicksort → WC → $T(n) = n + T(n - 1)$

$$T(n) = n$$

$$+ n - 1 \quad (1)$$

$$+ n - 2 \quad (2)$$

$$+ n - 3 \quad (3)$$

$$\vdots \qquad \qquad \qquad \vdots$$

$$+ 1 \qquad \qquad \qquad (n - 1)$$

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

Verifikasjon

Med substitusjon/induksjon

D&C → quicksort → WC → $T(n) = T(n - 1) + n$

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

D&C → quicksort → WC → $T(n) = T(n - 1) + n$

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

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

D&C → quicksort → WC → $T(n) = T(n - 1) + n$

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

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

D&C → quicksort → WC → $T(n) = T(n - 1) + n$

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

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

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

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

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

$$= \frac{n^2 - n + 2n}{2}$$

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

D&C → quicksort → WC → $T(n) = T(n - 1) + n$

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

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

$$= \frac{n^2 - n + 2n}{2}$$

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

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

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

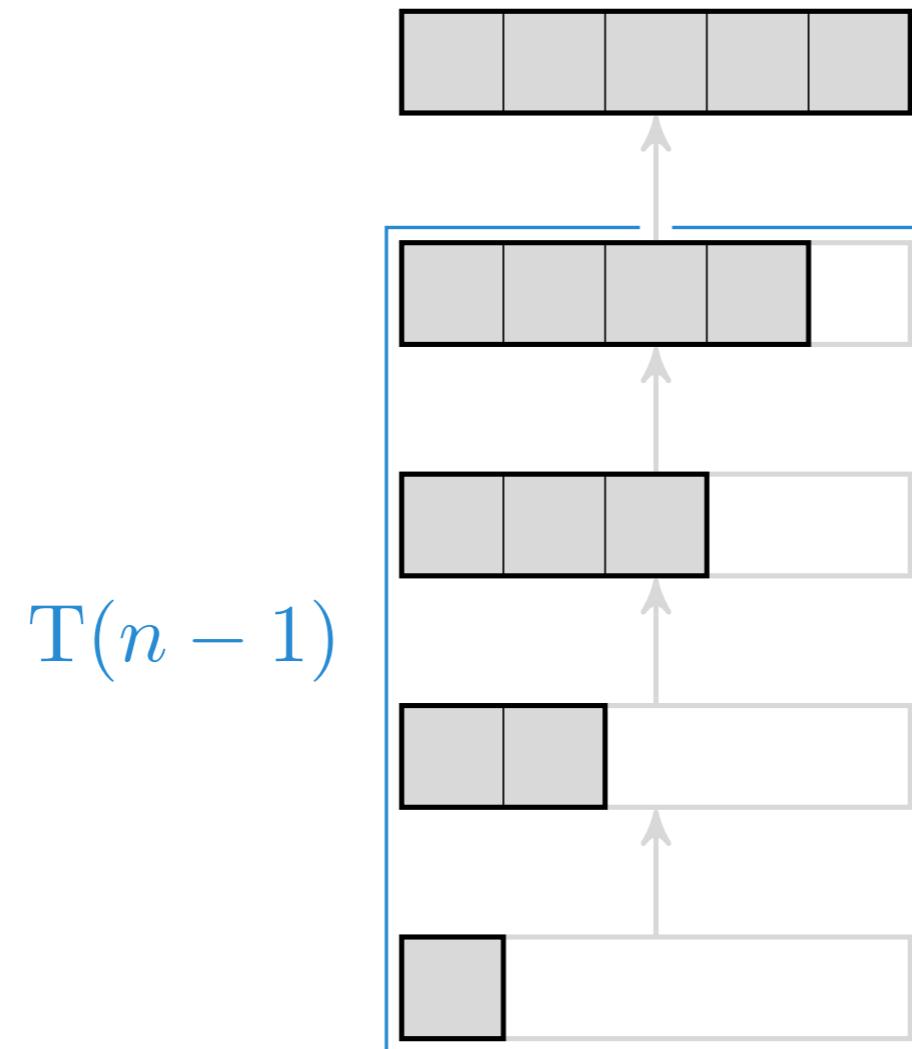
$$= \frac{(n - 1)n}{2} + n$$

$$= \frac{n^2 - n + 2n}{2}$$

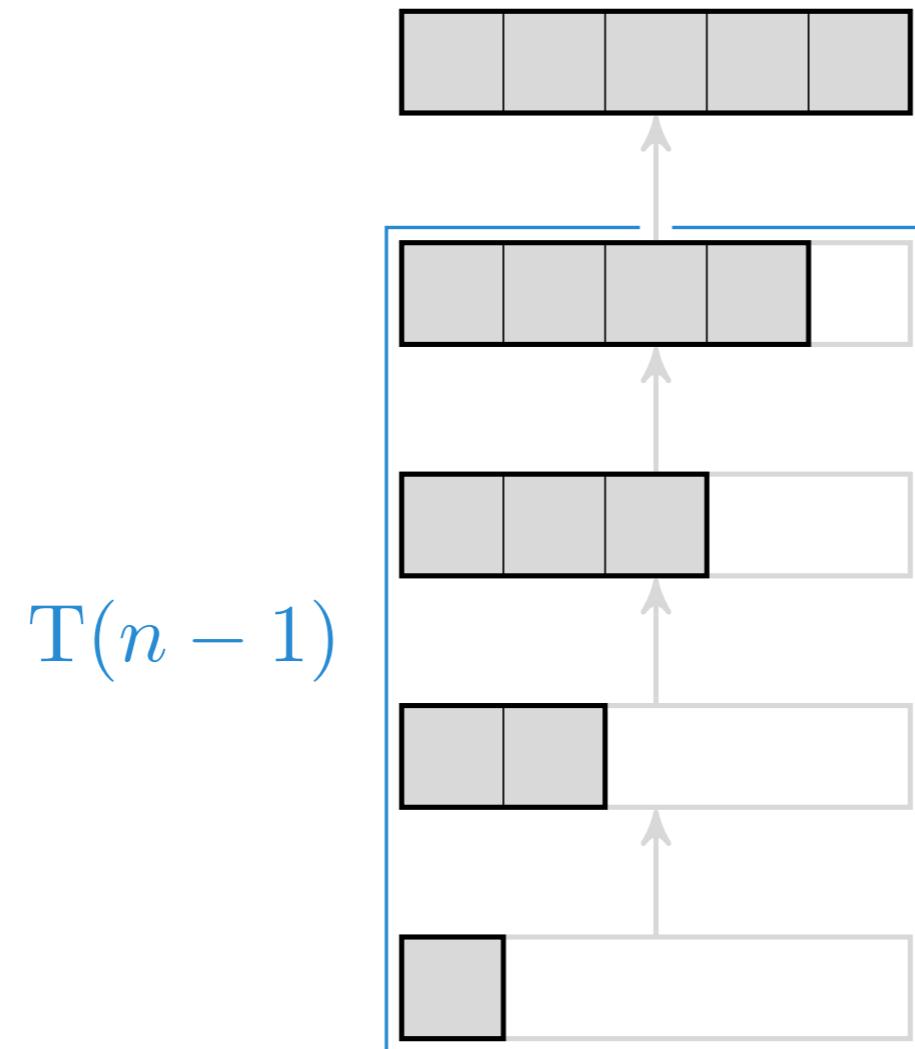
$$= \frac{n(n + 1)}{2}$$

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

$$\text{D\&C} \rightarrow \text{quicksort} \rightarrow \text{WC} \rightarrow T(n) = T(n-1) + n$$



D&C → quicksort → WC → $T(n) = T(n - 1) + n$



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

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/ltULW>)

RANDOMIZED-PARTITION(A, p, r)

- 1 $i = \text{RANDOM}(p, r)$
- 2 exchange $A[r]$ and $A[i]$
- 3 **return** PARTITION(A, p, r)

Tilfeldig splittelement, heller enn vilkårlig

```
RANDOMIZED-QUICKSORT(A, p, r)
1  if  $p < r$ 
2       $q = \text{RANDOMIZED-PARTITION}(A, p, r)$ 
3      RANDOMIZED-QUICKSORT(A, p, q - 1)
4      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 Recurrences¹

Jon Louis Bentley²

Dorothea Haken

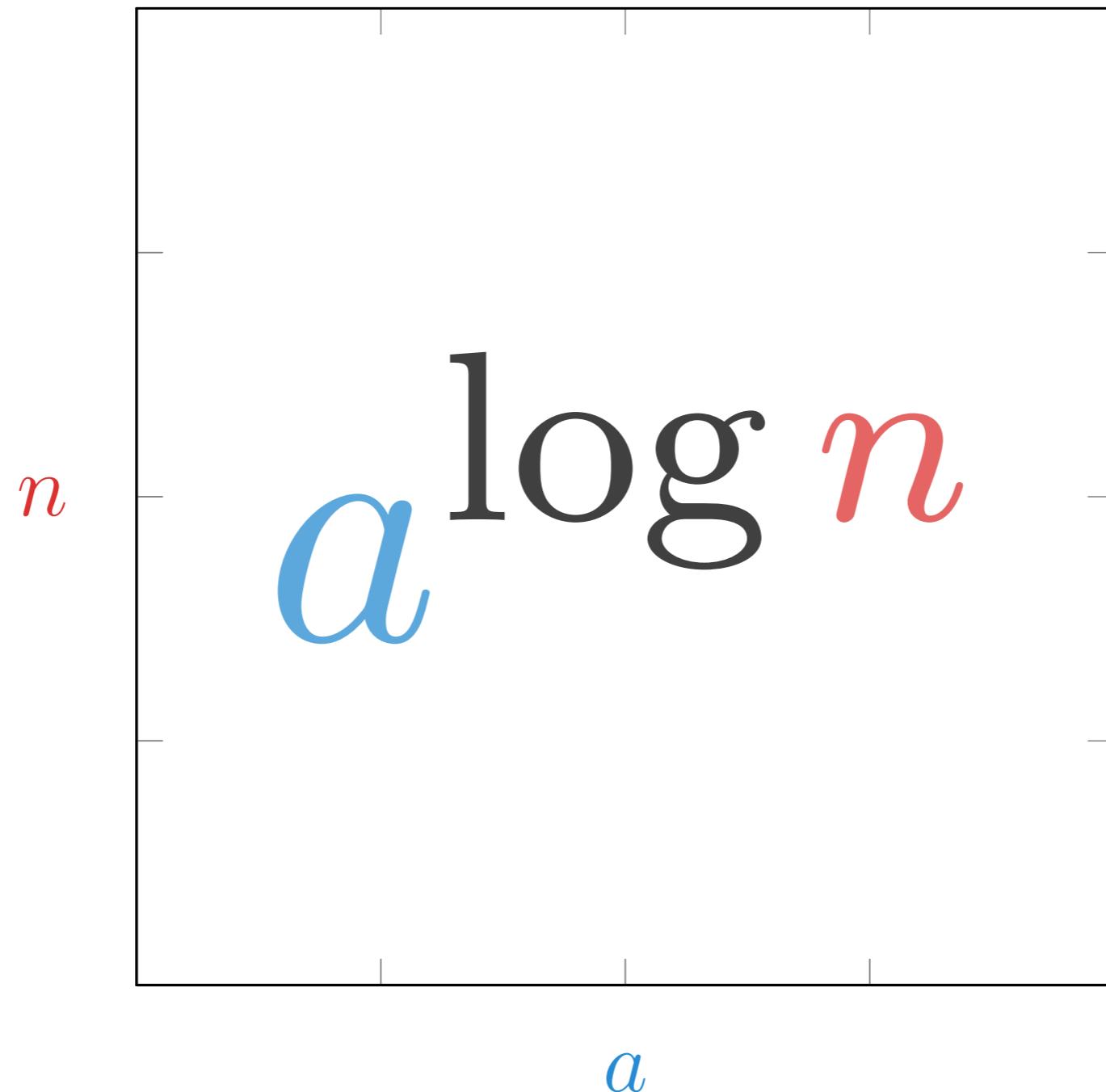
James B. Saxe

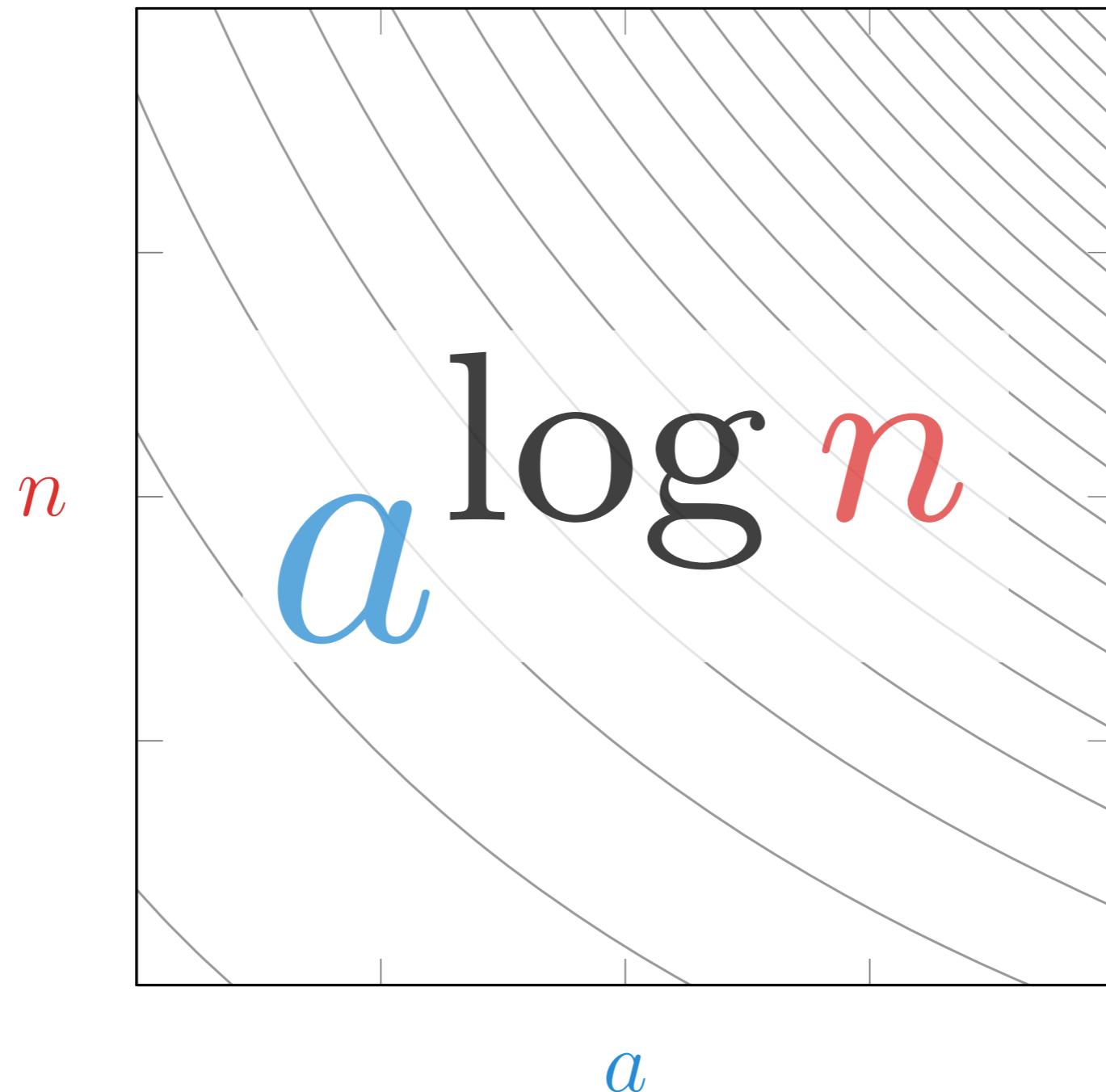
Department of Computer Science

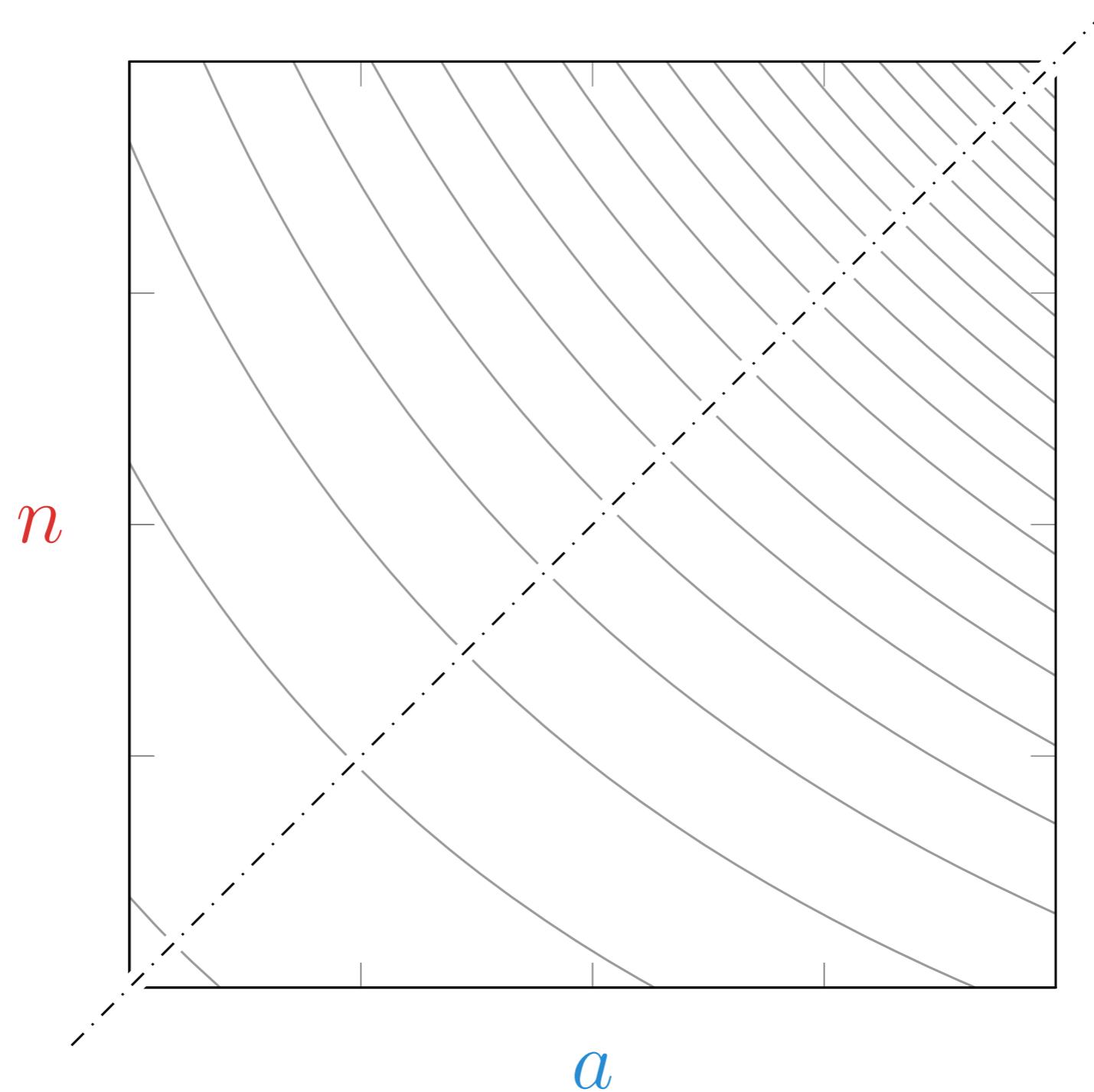
Carnegie-Mellon University

Pittsburgh, Pennsylvania 15213

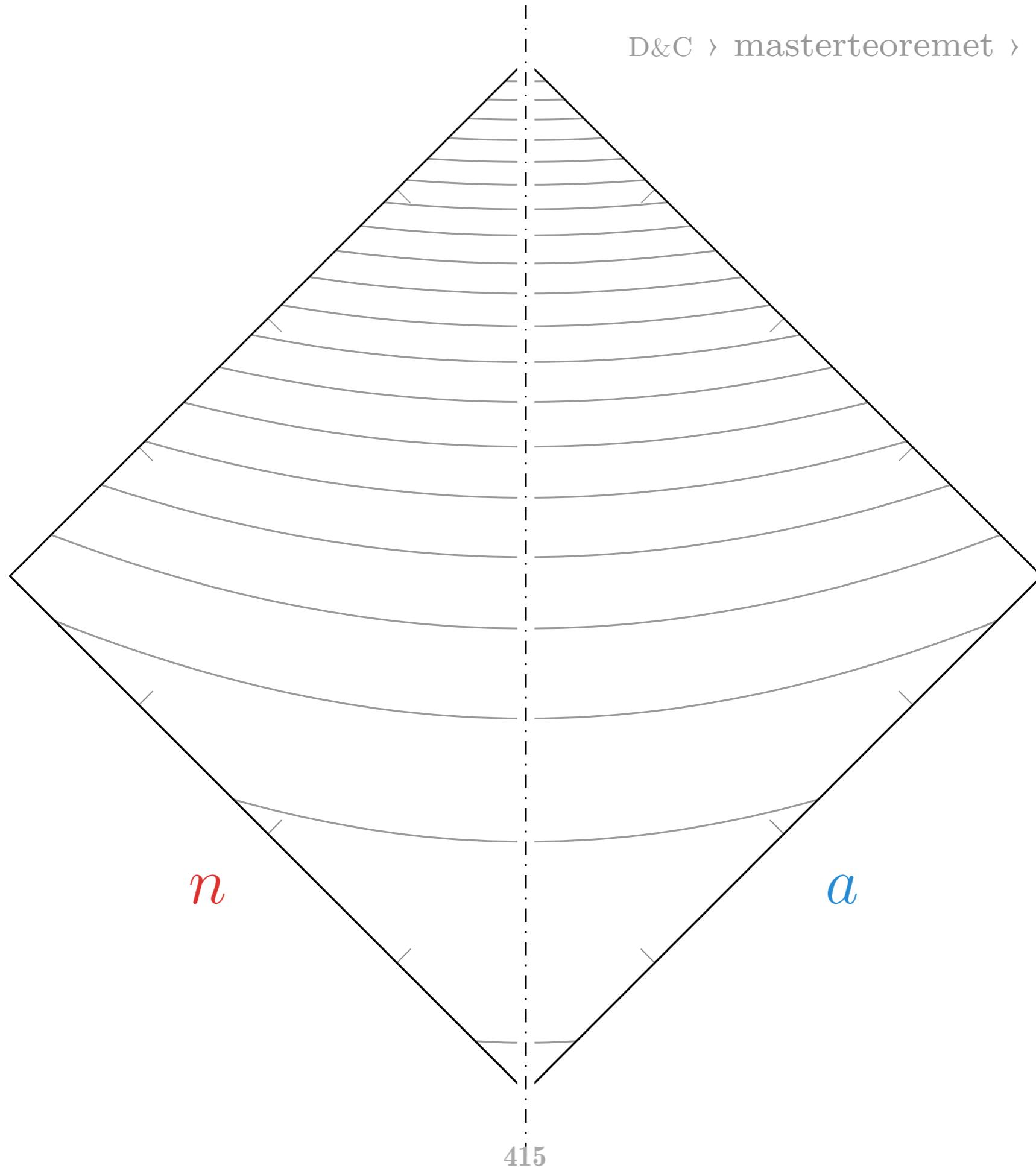
$a^{\log n}$

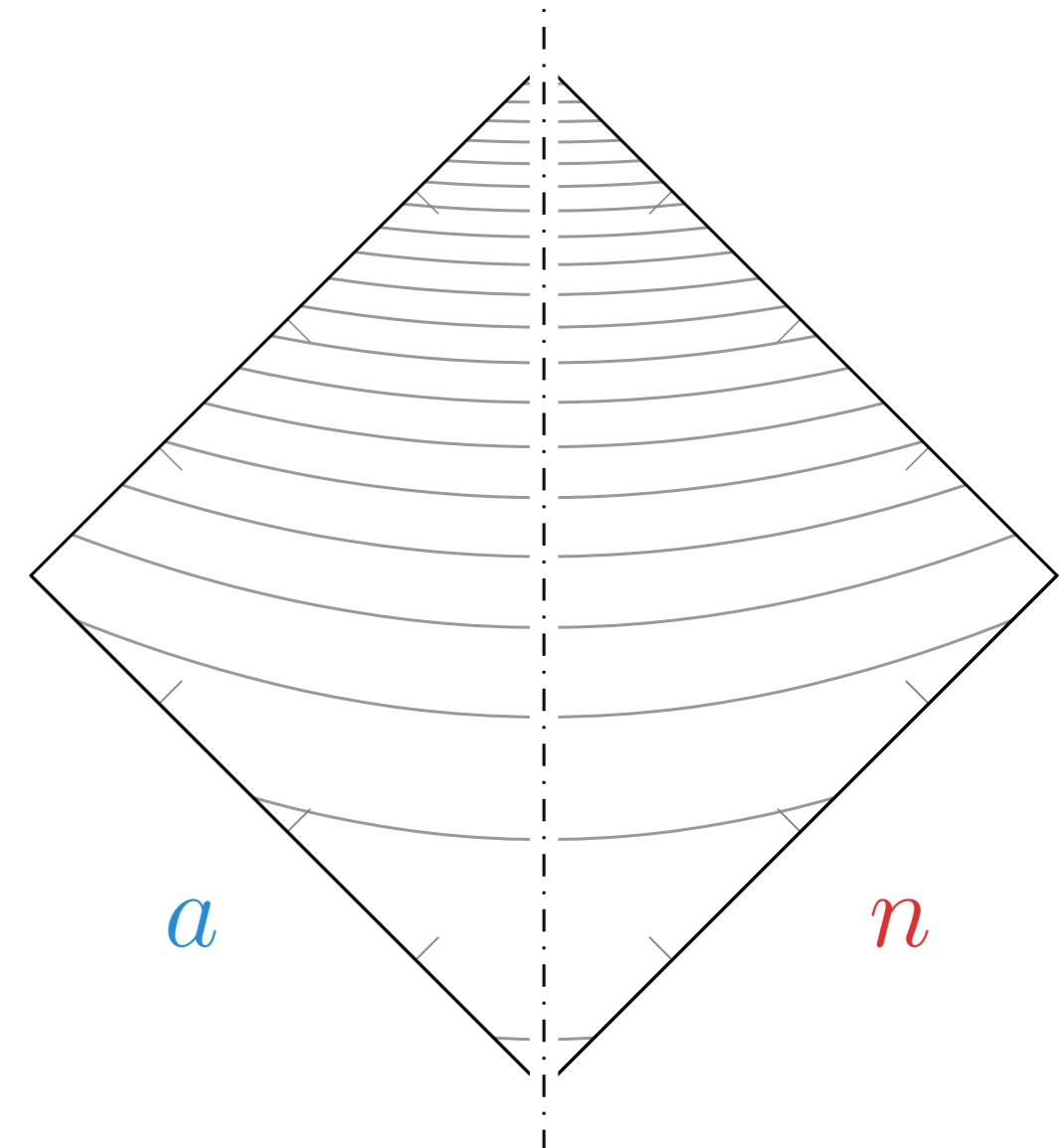
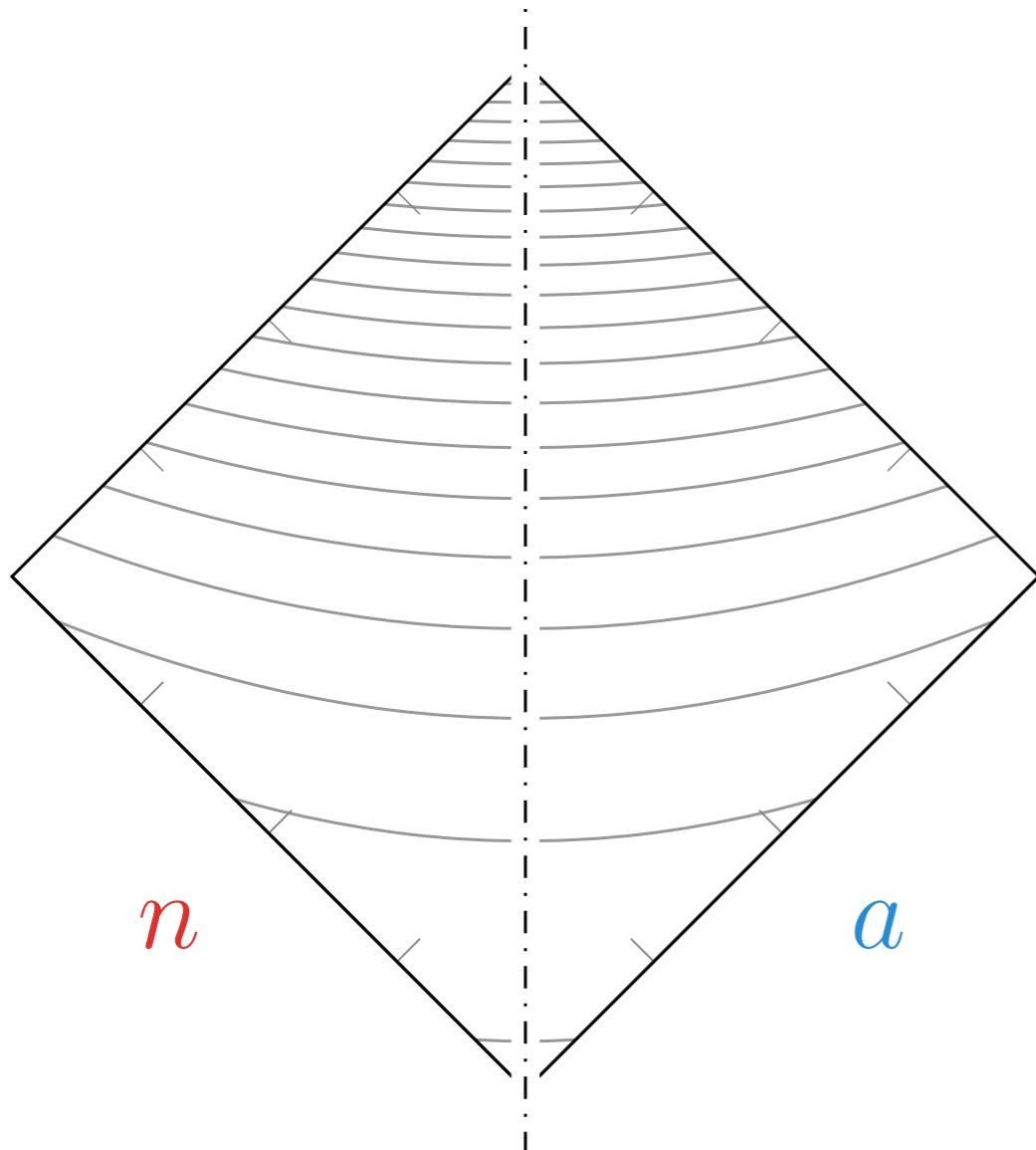


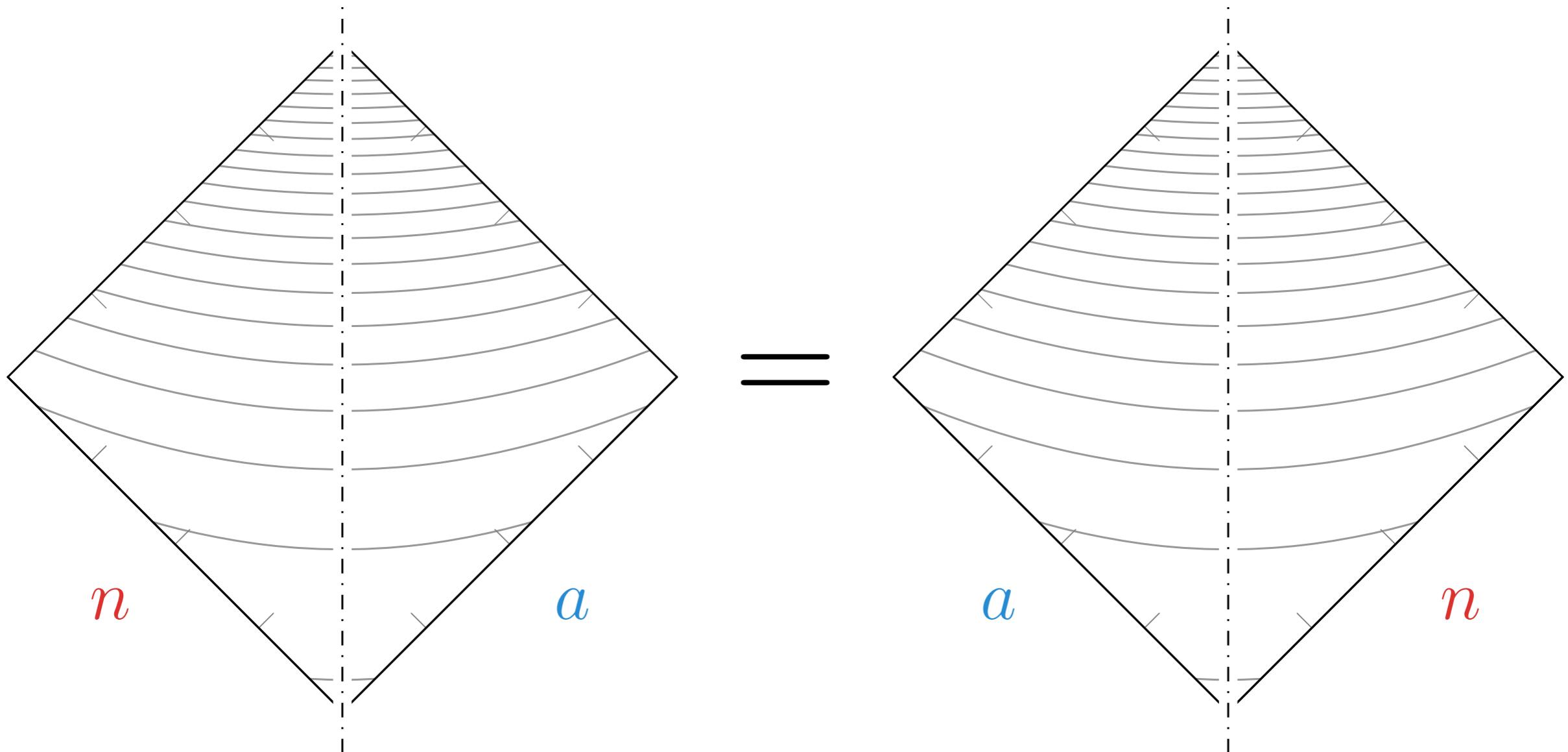


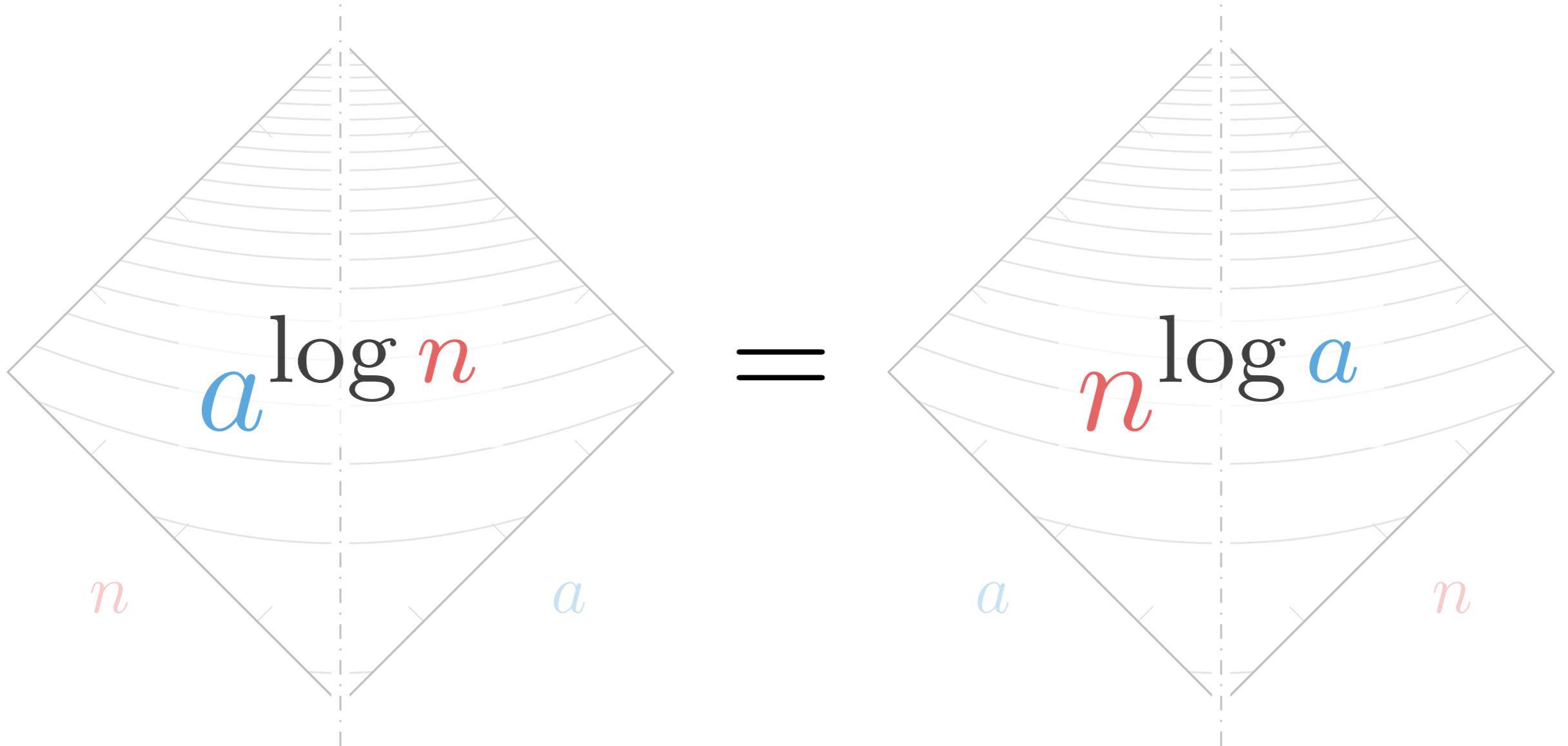


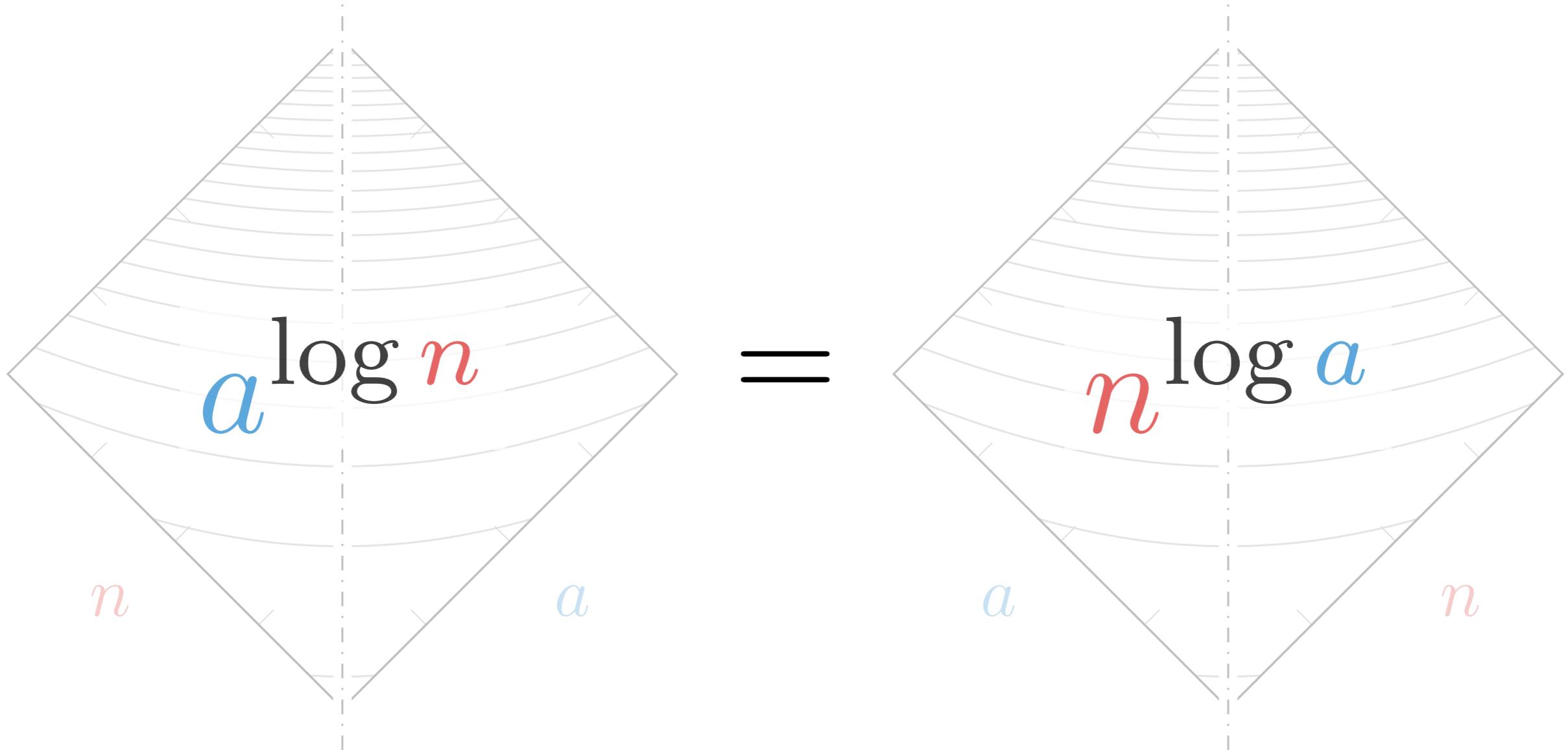
D&C → mastertheorem → $a^{\log n} = n^{\log a}$











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

$$T(n) = aT(n/b) + f(n)$$

$$T(n) = aT(n/b) + f(n)$$

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

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

D&C → masterteoremet → $T(\textcolor{blue}{n}) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

$$+ aT(\textcolor{blue}{n}/b) \quad (1)$$

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

$$+ a \cancel{f}(n/b) \quad (1)$$

$$+ a \cancel{a} T(n/b/b) \quad (2)$$

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

$$+ a^3 T(n/b^3) \quad (3)$$

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

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

⋮

⋮

$$+ a^? T(n/b^?) \quad (?)$$

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

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

⋮

⋮

$$+ a^? T(n/b^?) \quad (\log_b n)$$

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

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

⋮

⋮

$$+ a^{\log_b n} T(n/b^{\log_b n}) \quad (\log_b n)$$

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

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

⋮

⋮

$$+ a^{\log_b n} T(n/n) \quad (\log_b n)$$

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

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

⋮

⋮

$$+ a^{\log_b n} T(1) \quad (\log_b n)$$

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

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

⋮

⋮

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

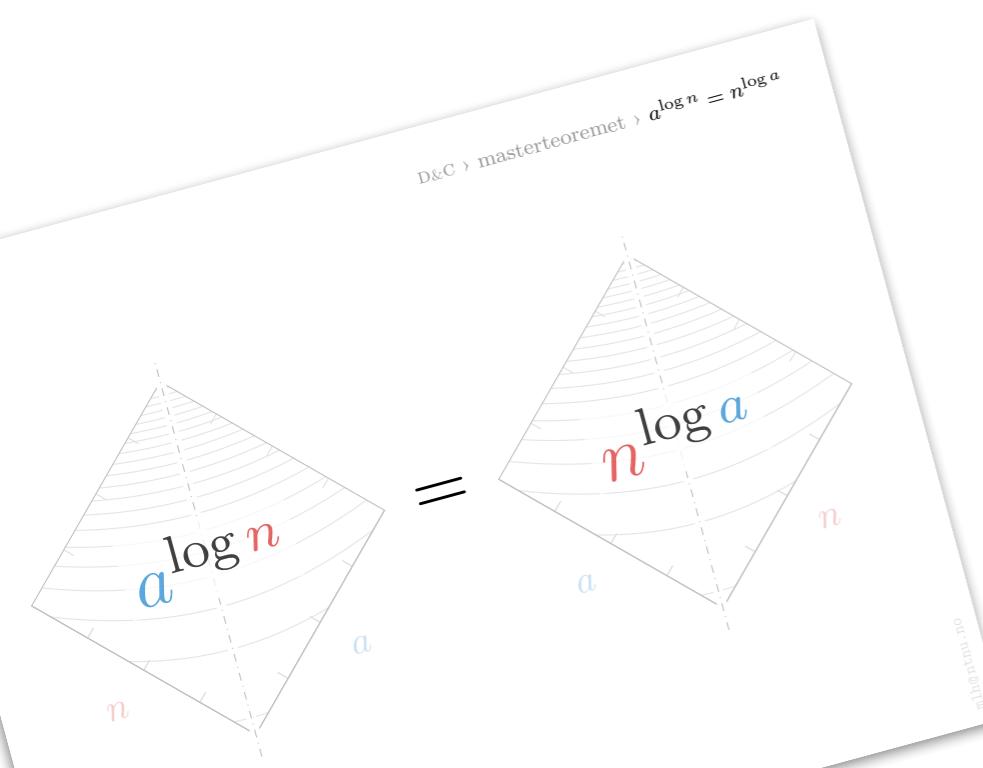
$$+ af(n/b) \quad (1)$$

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

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

⋮

$$+ a^{\log_b n} (\log_b n) \quad \vdots$$



D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

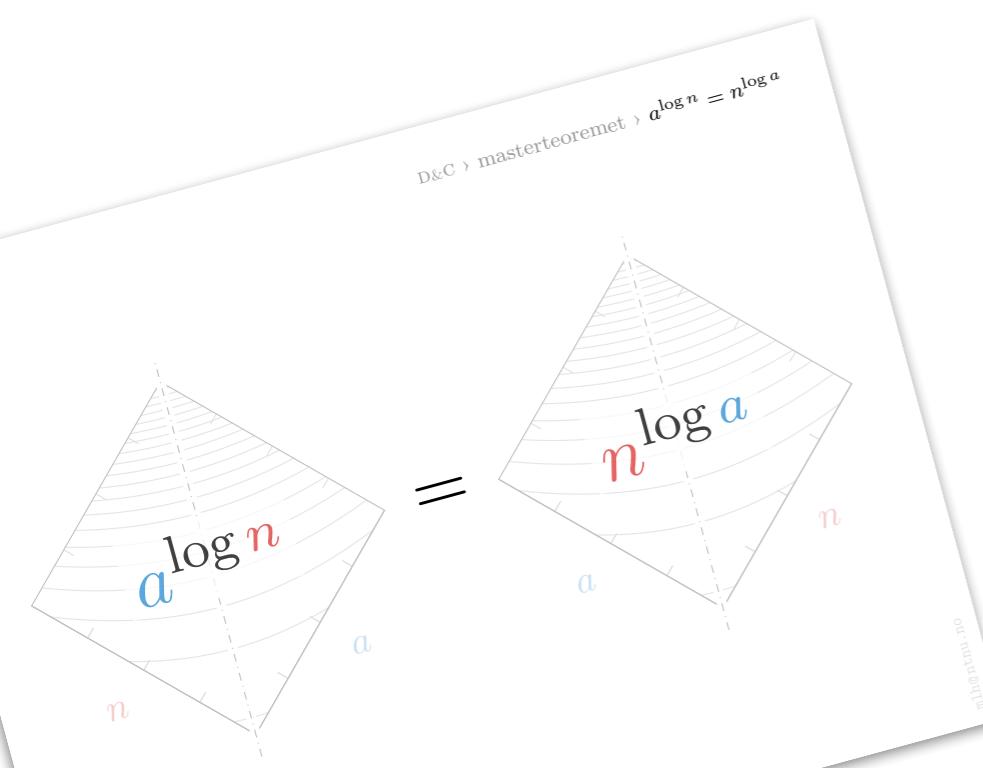
$$+ af(n/b) \quad (1)$$

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

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

⋮

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



D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

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

⋮

⋮

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

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

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

$$\vdots \qquad \qquad \qquad \vdots$$

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

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

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

$$\vdots \qquad \qquad \qquad \vdots$$

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

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

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

⋮

⋮

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

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

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

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

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

$$+ a^3f(n/b^3) \tag{3}$$

$$\vdots \qquad \qquad \qquad \vdots$$

$$+ n^{\log_b a} \tag{\log_b n}$$

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

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

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

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

$$+ a^3f(n/b^3) \tag{3}$$

$$\vdots \qquad \qquad \qquad \vdots$$

$$+ \Theta(n^{\log_b a}) \tag{\log_b n}$$

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

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

$$+ \Theta(n^{\log_b a}) \quad (1)$$

$$+ \Theta(n^{\log_b a}) \quad (2)$$

$$+ \Theta(n^{\log_b a}) \quad (3)$$

⋮

⋮

$$+ \Theta(n^{\log_b a}) \quad (\log_b n)$$

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

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

$$+ \Theta(n^{\log_b a}) \quad (1)$$

$$+ \Theta(n^{\log_b a}) \quad (2)$$

$$+ \Theta(n^{\log_b a}) \quad (3)$$

$$\vdots \qquad \qquad \qquad \vdots$$

$$+ \Theta(n^{\log_b a}) \quad (\log_b n)$$

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

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

$$+ \Theta(n^{\log_b a}) \quad (1)$$

$$+ \Theta(n^{\log_b a}) \quad (2)$$

$$+ \Theta(n^{\log_b a}) \quad (3)$$

⋮

⋮

$$+ \Theta(n^{\log_b a}) \quad (\log_b n)$$

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

$$T(n) = f(n)$$

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

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

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

⋮

⋮

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

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

$$T(n) = f(n)$$

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

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

$$+ a^3 f(n/b^3) \tag{3}$$

$$\vdots$$

$$\vdots$$

$$+ n^{\log_b a} \tag{\log_b n}$$

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

$$T(n) = f(n)$$

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

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

$$+ a^3 f(n/b^3) \tag{3}$$

⋮

⋮

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

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

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

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.

$$T(n) = f(n)$$

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

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

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

⋮

⋮

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

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

D&C → masterteoremet → $T(n) = f(n) + aT(n/b)$

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

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

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

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

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

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

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

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

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

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

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

$af(n/b) \leq cf(n)$
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

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

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

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

D&C → variabelskifte → $T(n^{\frac{1}{2}}) = \lg n$

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

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

D&C → variabelskifte → $T(n^{\frac{1}{2}}) = \lg n$

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

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

Potensen er fortsatt problematisk

D&C → variabelskifte → $T(n^{\frac{1}{2}}) = \lg n$

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

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

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

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

D&C → variabelskifte → $T(n^{\frac{1}{2}}) = \lg n$

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

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

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

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

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

D&C → variabelskifte → $T(n^{\frac{1}{2}}) = \lg n$

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

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

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

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

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

Slik ser ligningen vår ut med nye navn

D&C → variabelskifte → $T(n^{\frac{1}{2}}) = \lg n$

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

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

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

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

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

$$S(m) = 2m$$

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

D&C → variabelskifte → $T(n^{\frac{1}{2}}) = \lg n$

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

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

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

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

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

$$S(m) = 2m$$

$$T(n) = 2 \lg n$$

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

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

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

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

D&C → variabelskifte → $T(n) = 2T(n^{\frac{1}{2}}) + \lg n$

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

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

D&C ↗ variabelskifte ↗ $T(n) = 2T(n^{\frac{1}{2}}) + \lg n$

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

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

Potensen er fortsatt problematisk

D&C → variabelskifte → $T(n) = 2T(n^{\frac{1}{2}}) + \lg n$

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

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

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

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

D&C → variabelskifte → $T(n) = 2T(n^{\frac{1}{2}}) + \lg n$

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

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

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

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

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

D&C → variabelskifte → $T(n) = 2T(n^{\frac{1}{2}}) + \lg n$

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

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

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

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

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

Slik ser ligningen vår ut med nye navn

D&C → variabelskifte → $T(n) = 2T(n^{\frac{1}{2}}) + \lg n$

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

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

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

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

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

$$S(m) = m \lg m + m$$

Standard rekurrensløsning, som tidligere i dag

D&C → variabelskifte → $T(n) = 2T(n^{\frac{1}{2}}) + \lg n$

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

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

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

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

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

$$S(m) = m \lg m + m$$

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

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

Neste gang:

Vi har en universell, absolutt
fartsgrense for sortering.

Hvordan kan vi omgå den?

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

D&C

6. Variabelskifte



1. Rekurrenser

2. Binærsøk

3. Merge sort

5. Masterteoremet

6. Variabel
skifte

D&C



1. Rekurranseteknikk

2. Binærsøk

3. Merge sort

5. Masterteoremet

6. Variabelskifte

D&C

