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 kombiner løsningene.

Pensum

- ☐ Innledningen til del II
- ☐ 2. Getting started: 2.3
- ☐ 4. D&C: Innl. og 4.3–4.5
- ☐ 7. Quicksort
- ☐ Oppg. 2.1-4, 2.3-6 og 4.5-3 med løsning (binærsøk)
- ☐ App. B og C i pensumheftet

Læringsmål

Fra pensumheftet

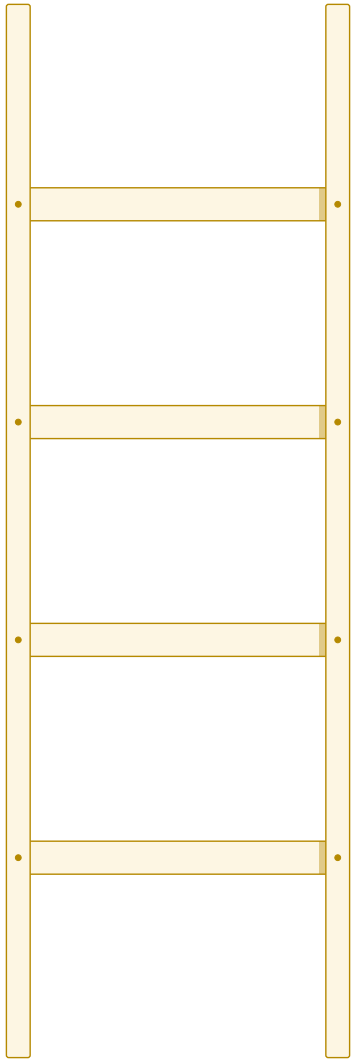
- [C_1] Forstå *divide-and-conquer* (*splitt og hersk*)
- [C_2] Forstå BISECT og BISECT′ (se appendiks C i pensumheftet)
- [C_3] Forstå MERGE-SORT
- [C_4] Forstå QUICKSORT og RANDOMIZED-QUICKSORT
- [C_5] Kunne løse rekurrenser med *substitusjon*, *rekursjonstrær* og *masterteoremet*
- [C_6] Kunne løse rekurrenser med *iterasjonsmetoden* (se appendiks B i pensumheftet)

mlh@ntnu.no

Forelesningen filmes

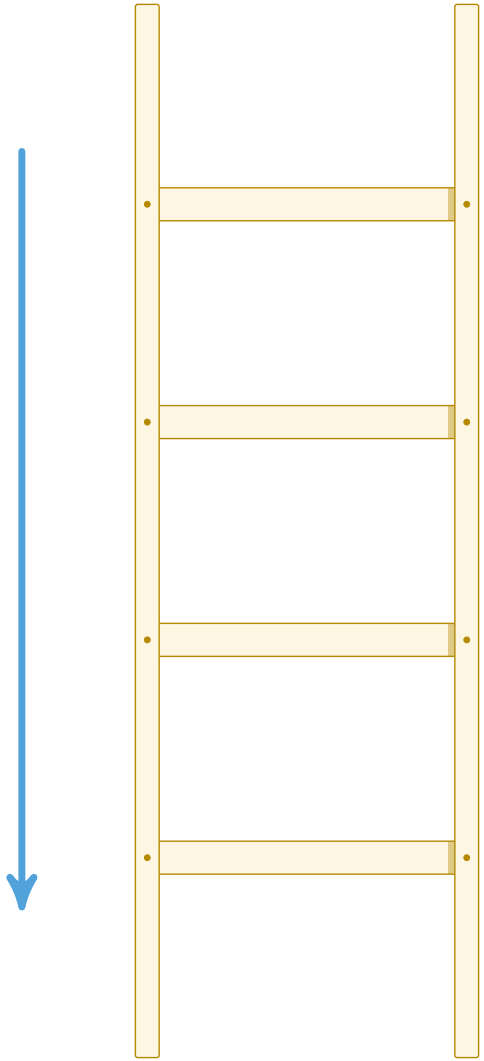
Fra forrige gang:

Ved reallokering av hashtabell, sett inn alle elementer på nytt, med ny hashfunksjon.



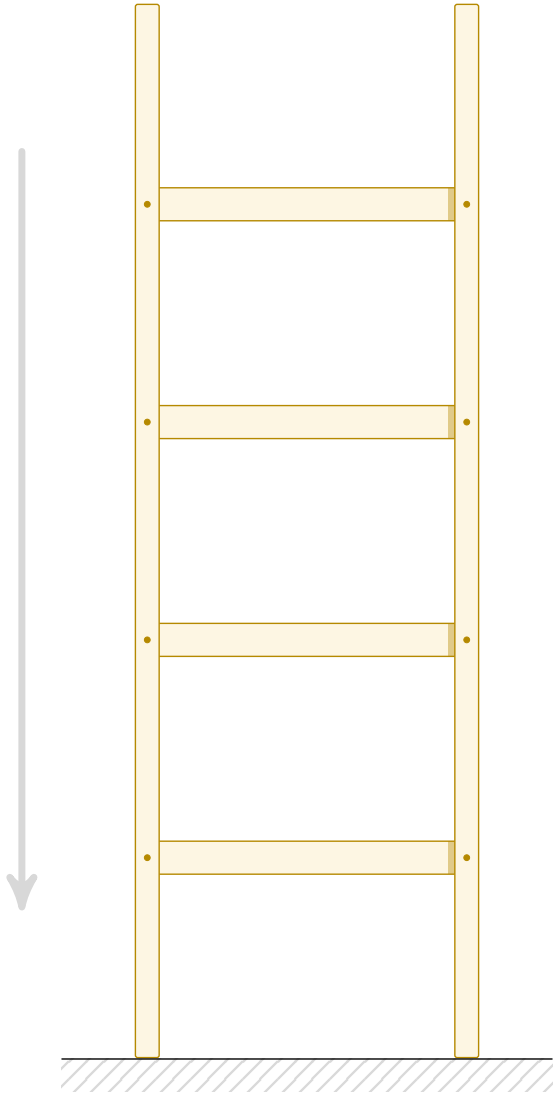
Dette er den velkjente induksjonsstigen vår

mlh@ntnu.no



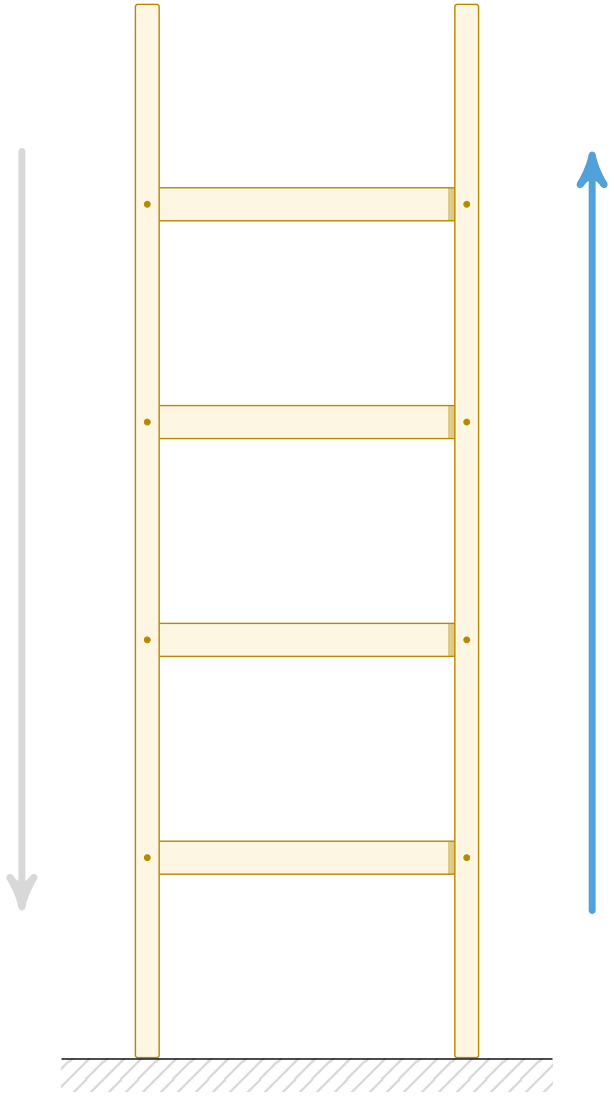
Vi reduserer en instans til mindre instanser . . .

mlh@ntnu.no



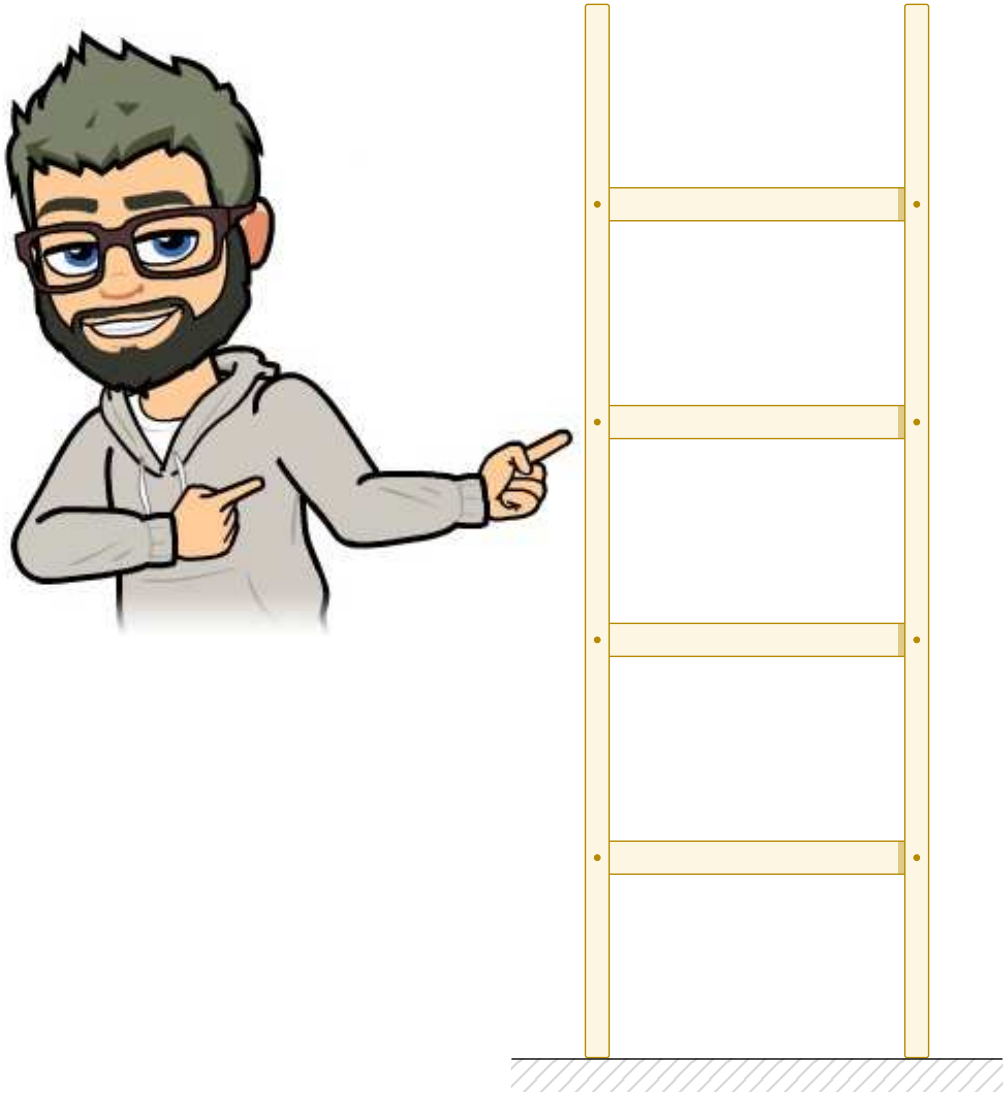
. . . til vi treffer grunntilfellet

mlh@ntnu.no



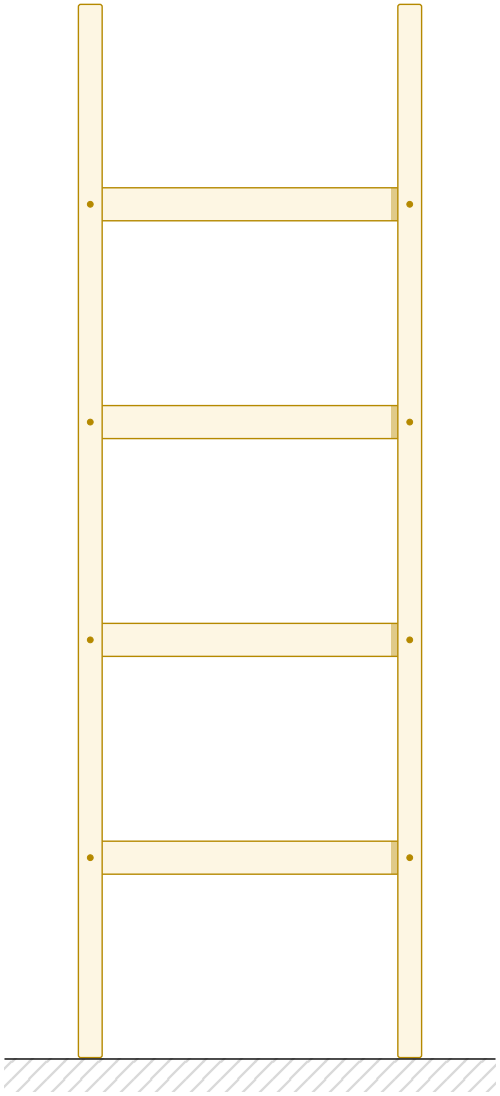
Vi løser grunntilfellet og bygger vi oss opp igjen

mlh@ntnu.no



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

mlh@ntnu.no

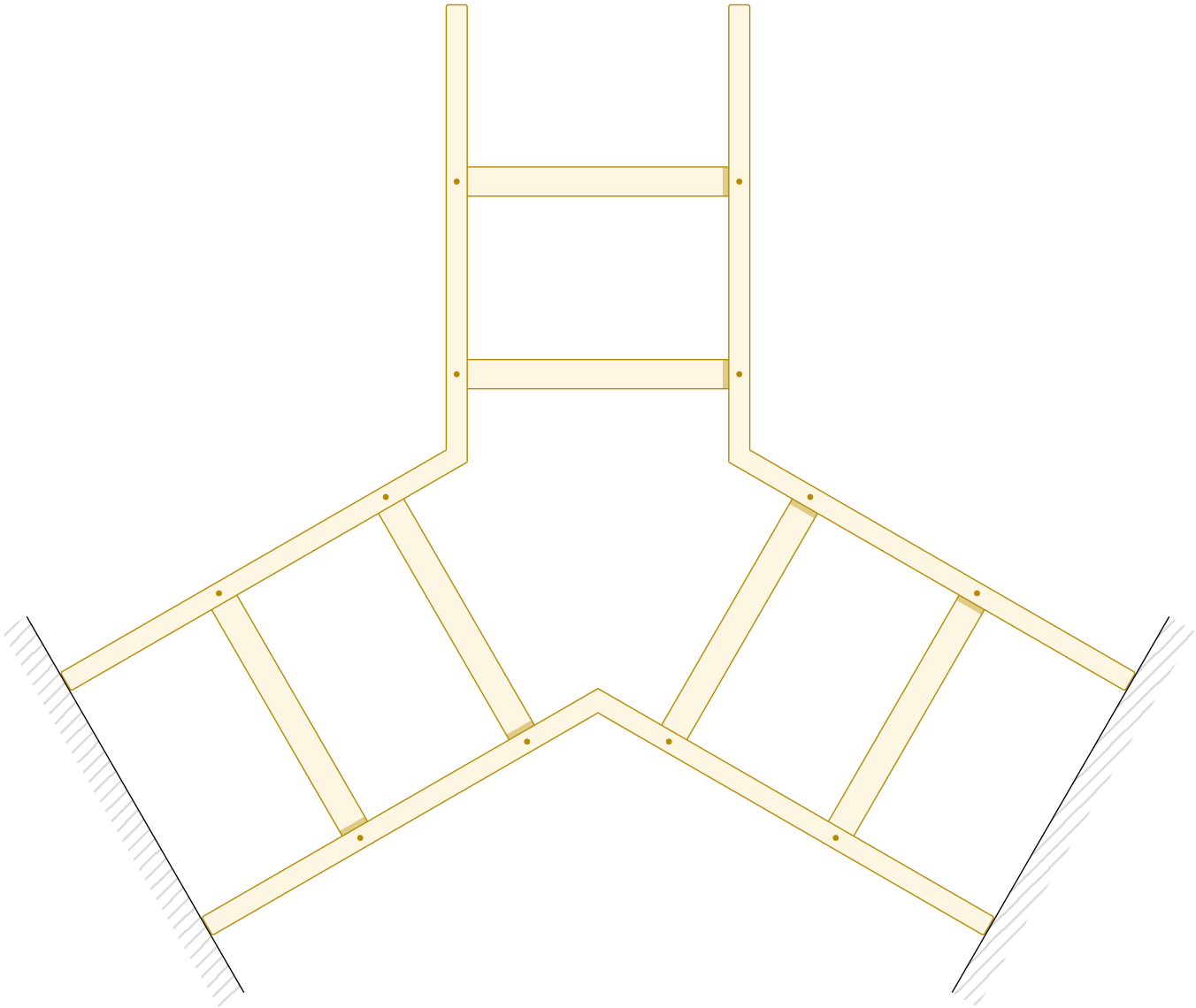


Men vi kan også spalte i flere delinstanser

mlh@ntnu.no

Men vi kan også spalte i flere delinstanser

mlh@ntnu.no



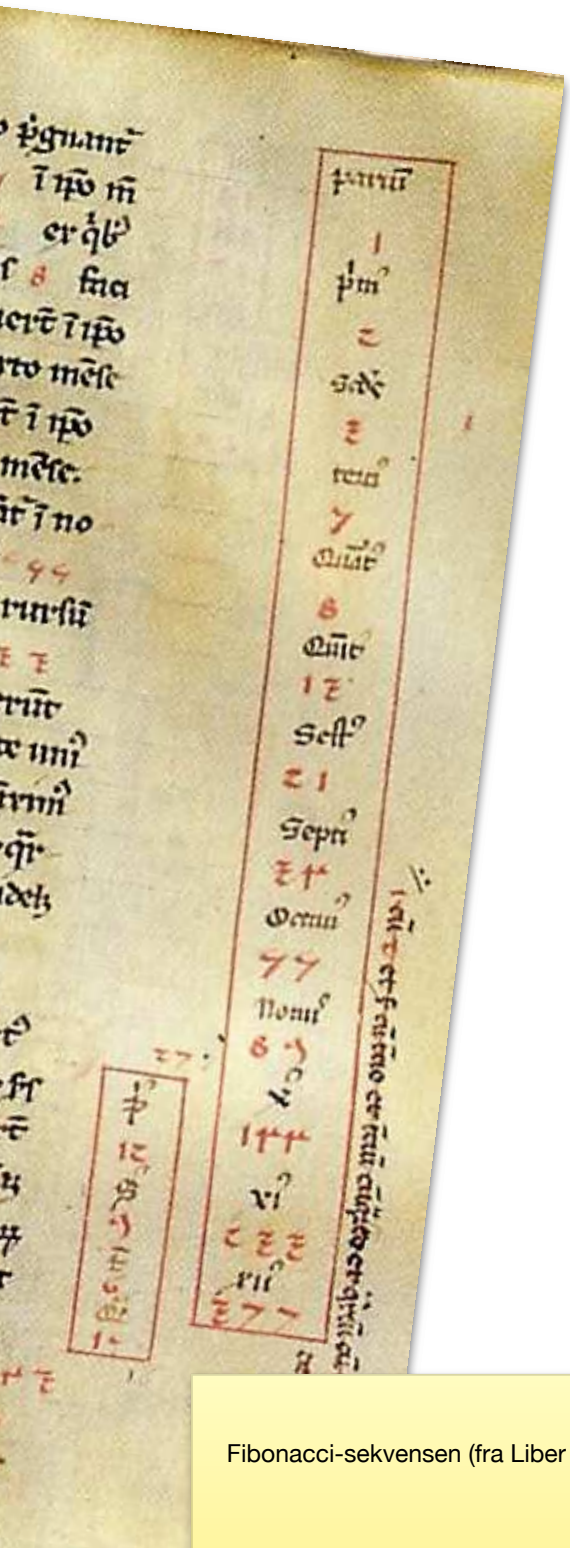
Denne designmetoden kalles «splitt og hersk» (*divide & conquer*)

mlh@ntnu.no

Forelesning 3

Splitt & hersk

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



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

1:5

Rekurrenser

Fibonacci-sekvensen (fra Liber Abaci, 1228)

Ser bort fra implisitt avrunding, her. Mer om det i boka.

(Innbakt i masterteoremet.)

$$f(\boxed{n}) = 2\boxed{n} + 1$$

Vi definerer en funksjon slik

mlh@ntnu.no

$$f(\boxed{n}) = 2\boxed{n} + 1$$

$$f(\boxed{x}) = 2\boxed{x} + 1$$

$$f(\boxed{4}) = 2\boxed{4} + 1$$

$$f(\boxed{}) = 2\boxed{} + 1$$

Da er disse ligningene gyldige

mlh@ntnu.no

$$f(\boxed{n}) = f(\boxed{n}/2) + 1$$

$$f(\boxed{x}) = f(\boxed{x}/2) + 1$$

$$f(\boxed{4}) = f(\boxed{4}/2) + 1$$

$$f(\boxed{}) = f(\boxed{}/2) + 1$$

Fungerer selv om f finnes på begge sider

mlh@ntnu.no

$$f(\boxed{n}) = f(\boxed{n}/2) + 1$$

Hva om vi vil bli kvitt f til høyre?

mlh@ntnu.no

$$f(\boxed{n}) = f(\boxed{n}/2) + 1$$

Da må vi finne ut hva $f(\boxed{n}/2)$ er

mlh@ntnu.no

$$f(\boxed{n}) = f(\boxed{n}/2) + 1$$

$$f(\boxed{n}/2) =$$

Da må vi finne ut hva $f(\boxed{n}/2)$ er

mlh@ntnu.no

$$f(\boxed{n}) = f(\boxed{n}/2) + 1$$

$$f(\boxed{n}/2) =$$

Men $\boxed{n}/2$ er jo et helt vanlig argument!

mlh@ntnu.no

$$f(\boxed{n}) = f(\boxed{n}/2) + 1$$

$$f(\boxed{\boxed{n}/2}) = f(\boxed{\boxed{n}/2}/2) + 1$$

Men $\boxed{\boxed{n}/2}$ er jo et helt vanlig argument!

mlh@ntnu.no

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

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

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

Og vi kan fortsette på samme måte!

mlh@ntnu.no

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

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

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

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

Og vi kan fortsette på samme måte!

mlh@ntnu.no

$$f(\boxed{n}) = f(\boxed{n}/2) + 1$$

$$f(\boxed{\boxed{n}/2}) = f(\boxed{\boxed{n}/2}/2) + 1$$

$$f(\boxed{\boxed{\boxed{n}/2}/2}) = f(\boxed{\boxed{\boxed{n}/2}/2}/2) + 1$$

$$f(\boxed{\boxed{\boxed{\boxed{n}/2}/2}/2}) = f(\boxed{\boxed{\boxed{\boxed{n}/2}/2}/2}/2) + 1$$

Utfordringen blir å komme oss til bunns og nøste oss tilbake

mlh@ntnu.no

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

- **Finn løsning:**
  - **Iterasjonsmetoden**
  - **Rekursjonstrær**
  - **Masterteoremet**
- **Verifiser løsning:**
  - **Substitusjon (dvs. induksjon)**

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

$$T(1) = 1$$

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

mlh@ntnu.no

Eksempel: $1 + 1 + \ldots$

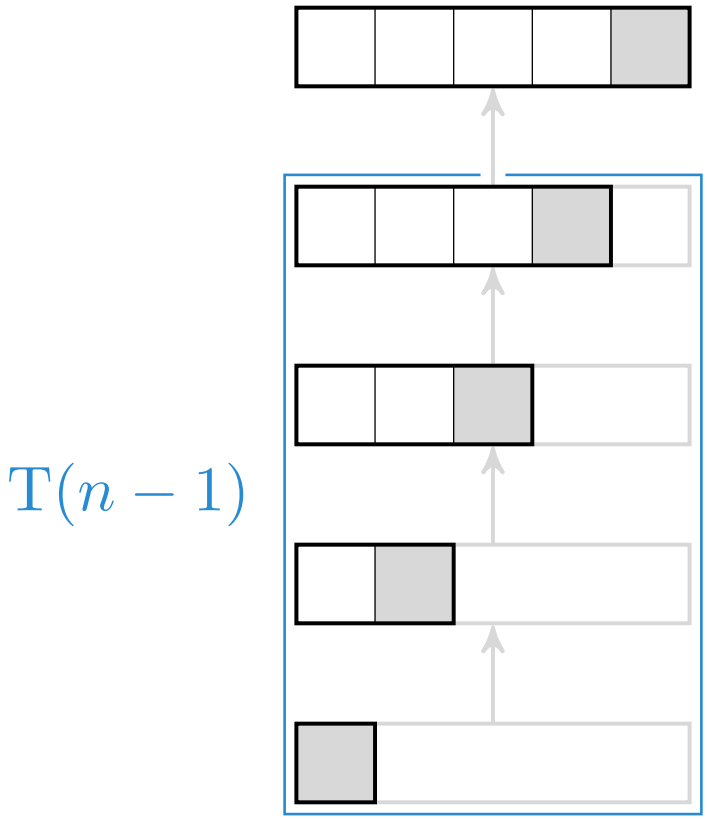
$$\mathrm{T}(n) = \mathrm{T}(n-1) + 1$$

mlh@ntnu.no

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

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

mlh@ntnu.no



Rekursjonstre

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

$$+ 1 \qquad (1)$$

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

mlh@ntnu.no

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

$$+ 1 \tag{1}$$

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

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

mlh@ntnu.no

Verifikasjon

Med substitusjon/induksjon

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

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

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

mlh@ntnu.no

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

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

$$= n - 1 + 1$$

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

mlh@ntnu.no

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

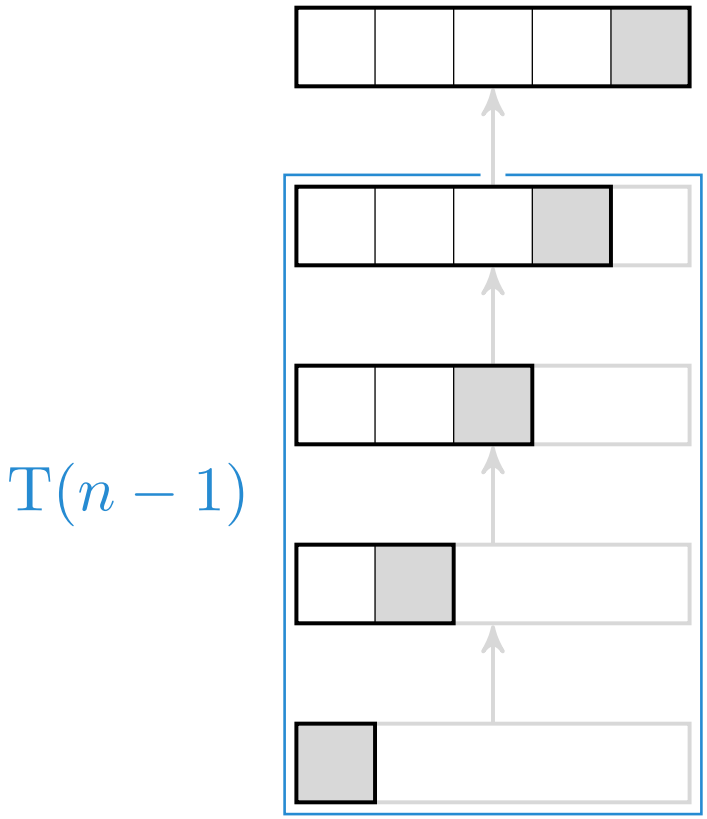
Gitt antagelsen $\mathrm{T}(n-1) = n - 1$, vis at $\mathrm{T}(n) = n$

mlh@ntnu.no

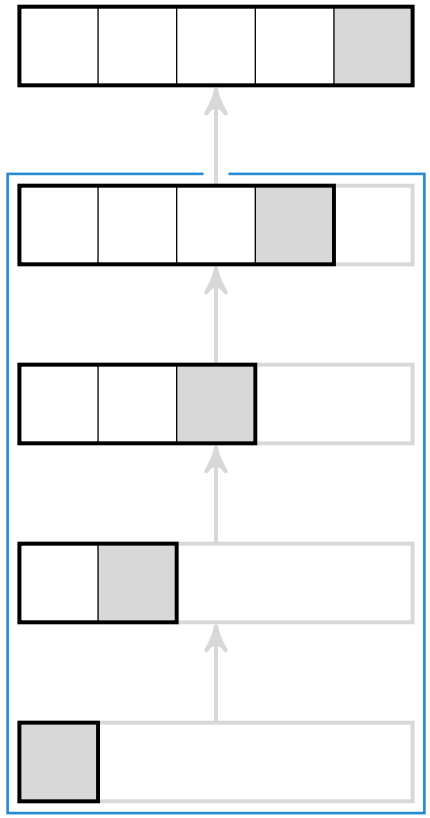
$$\begin{aligned}\mathrm{T}(n) &= \mathrm{T}(n-1) + 1 \\ &= (n-1) + 1 \\ &= n - 1 + 1 \\ &= n\end{aligned}$$

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

mlh@ntnu.no



mlh@ntnu.no



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

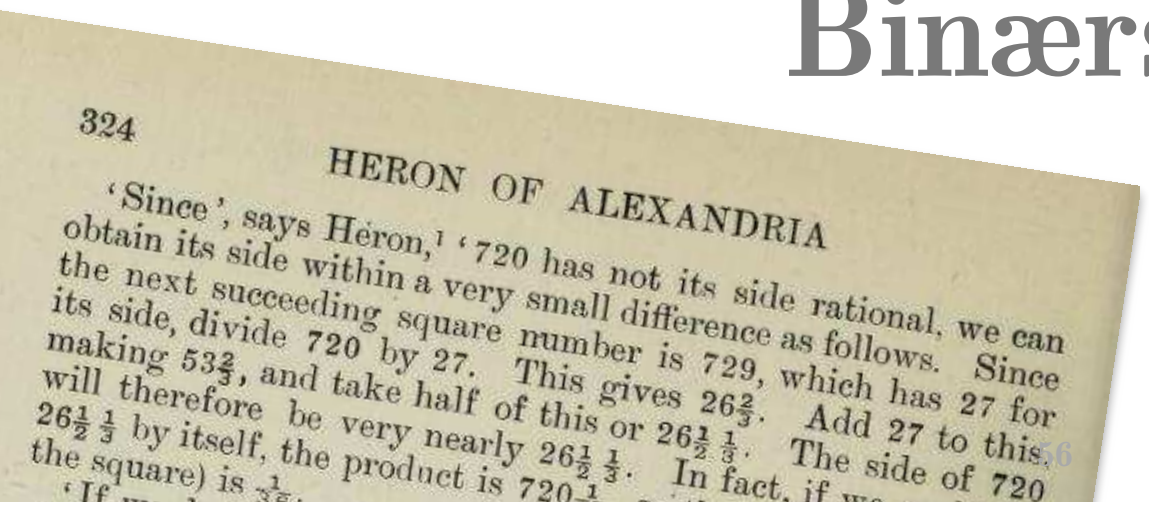
$$\mathrm{T}(n) = n$$

mlh@ntnu.no

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).

2:5

Binærsøk



324 HERON OF ALEXANDRIA

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

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

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

Instans	Løsning
Dekomponering	Kombinasjon IS
Delinstans	Delløsninger IH
Grunntilfelle ...	Grunnløsning ...

mlh@ntnu.no

Instans	Løsning
Dekomponering	Kombinasjon IS
Delinstans	Delløsninger IH
Grunntilfelle ...	Grunnløsning ...

Hva er input?

mlh@ntnu.no

Instans En ikke tom, sortert sekvens A og en verdi v	Løsning
Dekomponering	Kombinasjon IS
Delinstans	Delløsninger IH
Grunntilfelle ...	Grunnløsning ...

Hva er input?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	
Dekomponering	Kombinasjon IS
Delinstans	Delløsninger IH
...	...
Grunntilfelle	Grunnløsning

Hva ønsker vi at output skal være?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering	Kombinasjon IS
Delinstans	Delløsninger IH
...	...
Grunntilfelle	Grunnløsning

Hva ønsker vi at output skal være?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering	Kombinasjon IS
Delinstans	Delløsninger IH
...	...
Grunntilfelle	Grunnløsning

Hvordan kan vi krympe eller dele opp instansen?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering	Kombinasjon IS
Er v i venstre eller høyre halvdel av A?	
Delinstans	Delløsninger IH
...	...
Grunntilfelle	Grunnløsning

Hvordan kan vi krympe eller dele opp instansen?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering Er v i venstre eller høyre halvdel av A?	Kombinasjon IS
Delinstans	Delløsninger IH
Grunntilfelle	Grunnløsning

Hva har vi brutt problemet ned til?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering	Kombinasjon IS
Er v i venstre eller høyre halvdel av A?	
Delinstans	Delløsninger IH
Venstre eller høyre halvdel, $A[p:r]$	
...	...
Grunntilfelle	Grunnløsning

Hva har vi brutt problemet ned til?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering	Kombinasjon IS
Er v i venstre eller høyre halvdel av A?	
Delinstans	Delløsninger IH
Venstre eller høyre halvdel, $A[p:r]$	
...	...
Grunntilfelle	Grunnløsning

Hvor stopper dekomponeringen?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering	Kombinasjon IS
Er v i venstre eller høyre halvdel av A?	
Delinstans	Delløsninger IH
Venstre eller høyre halvdel, $A[p:r]$	
...	...
Grunntilfelle	Grunnløsning
Intervall av lengde 1, altså der $p == r$	

Hvor stopper dekomponeringen?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering	Kombinasjon IS
Er v i venstre eller høyre halvdel av A?	
Delinstans	Delløsninger IH
Venstre eller høyre halvdel, $A[p:r]$	
Grunntilfelle	Grunnløsning
Intervall av lengde 1, altså der $p == r$	

Hva er løsningen da?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering	Kombinasjon IS
Er v i venstre eller høyre halvdel av A?	
Delinstans	Delløsninger IH
Venstre eller høyre halvdel, $A[p:r]$	
Grunntilfelle	Grunnløsning
Intervall av lengde 1, altså der $p == r$	Hvis $A[p] = v$ er løsningen p; ellers NIL

Hva er løsningen da?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering	Kombinasjon IS
Er v i venstre eller høyre halvdel av A?	
Delinstans	Delløsninger IH
Venstre eller høyre halvdel, $A[p:r]$	
...	...
Grunntilfelle	Grunnløsning
Intervall av lengde 1, altså der $p == r$	Hvis $A[p] = v$ er løsningen p; ellers NIL

Hva er løsningene for delinstansene?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering	Kombinasjon IS
Er v i venstre eller høyre halvdel av A?	
Delinstans	Delløsninger IH
Venstre eller høyre halvdel, $A[p:r]$	Indeks fra riktig halvdel, eller NIL
Grunntilfelle	Grunnløsning
Intervall av lengde 1, altså der $p == r$	Hvis $A[p] = v$ er løsningen p; ellers NIL

Hva er løsningene for delinstansene?

mlh@ntnu.no

Instans	Løsning
En ikke tom, sortert sekvens A og en verdi v	Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomponering	**Kombinasjon** IS
Er v i venstre eller høyre halvdel av A?	
Delinstans	**Delløsninger** IH
Venstre eller høyre halvdel, $A[p:r]$	Indeks fra riktig halvdel, eller NIL
Grunntilfelle	**Grunnløsning**
Intervall av lengde 1, altså der $p == r$	Hvis $A[p] = v$ er løsningen p; ellers NIL

Hvordan bygger vi en full løsning av delløsninger?

mlh@ntnu.no

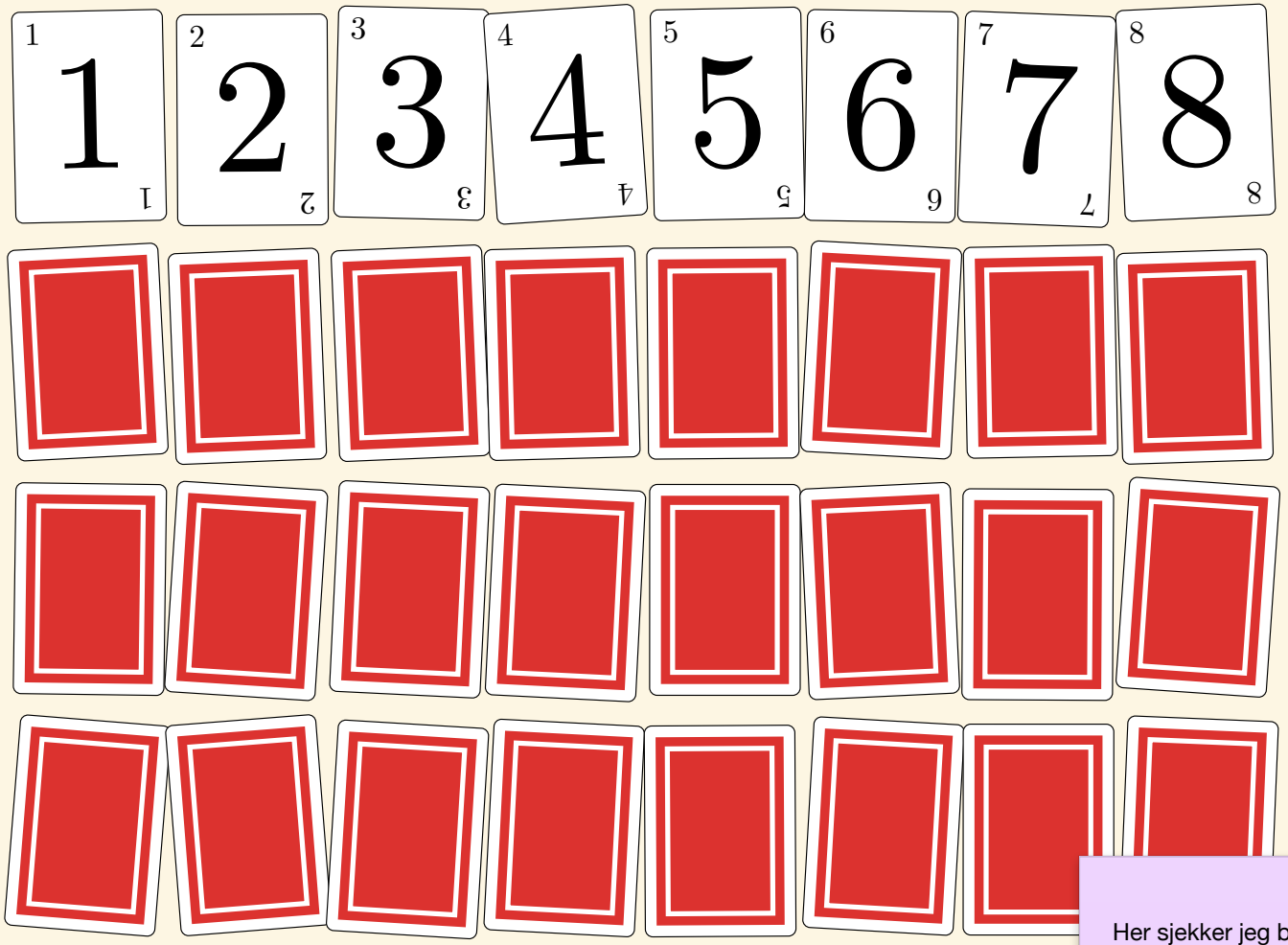
Instans En ikke tom, sortert sekvens A og en verdi v	Løsning Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomposering Er v i venstre eller høyre halvdel av A?	Kombinasjon ↑ IS
Delinstans Venstre eller høyre halvdel, $A[p:r]$	Delløsninger IH Indeks fra riktig halvdel, eller NIL
Grunntilfelle Intervall av lengde 1, altså der $p == r$	Grunnløsning Hvis $A[p] = v$ er løsningen p; ellers NIL

Hvordan bygger vi en full løsning av delløsninger?

mlh@ntnu.no

Instans En ikke tom, sortert sekvens A og en verdi v	Løsning Om mulig, en indeks q, så $A[q] = v$; ellers NIL
Dekomposering Er v i venstre eller høyre halvdel av A?	Kombinasjon ↑ IS
Delinstans Venstre eller høyre halvdel, $A[p:r]$	Delløsninger IH Indeks fra riktig halvdel, eller NIL
Grunntilfelle Intervall av lengde 1, altså der $p == r$	Grunnløsning Hvis $A[p] = v$ er løsningen p; ellers NIL

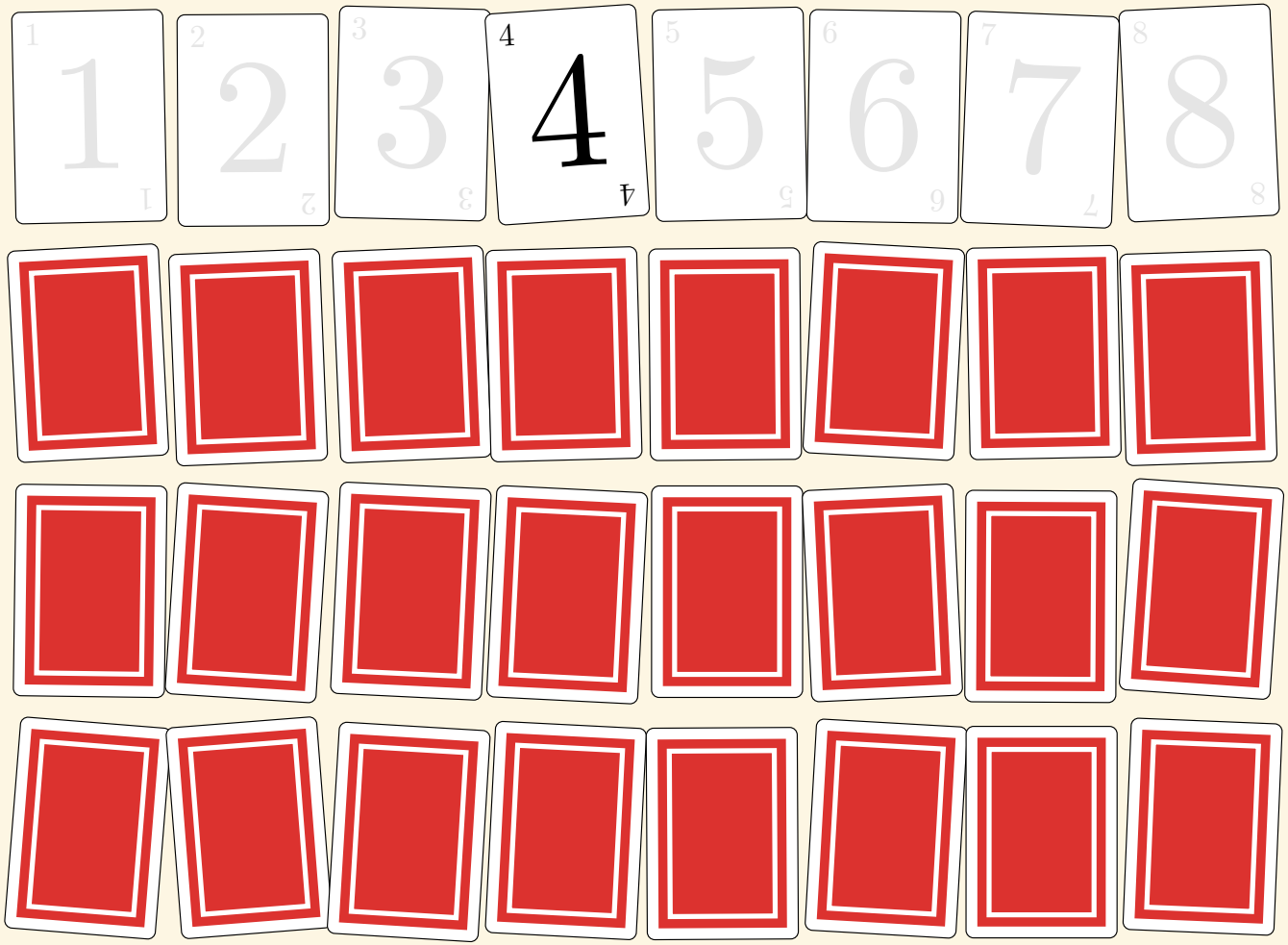
mlh@ntnu.no



(Anta at kortet ditt er 6)

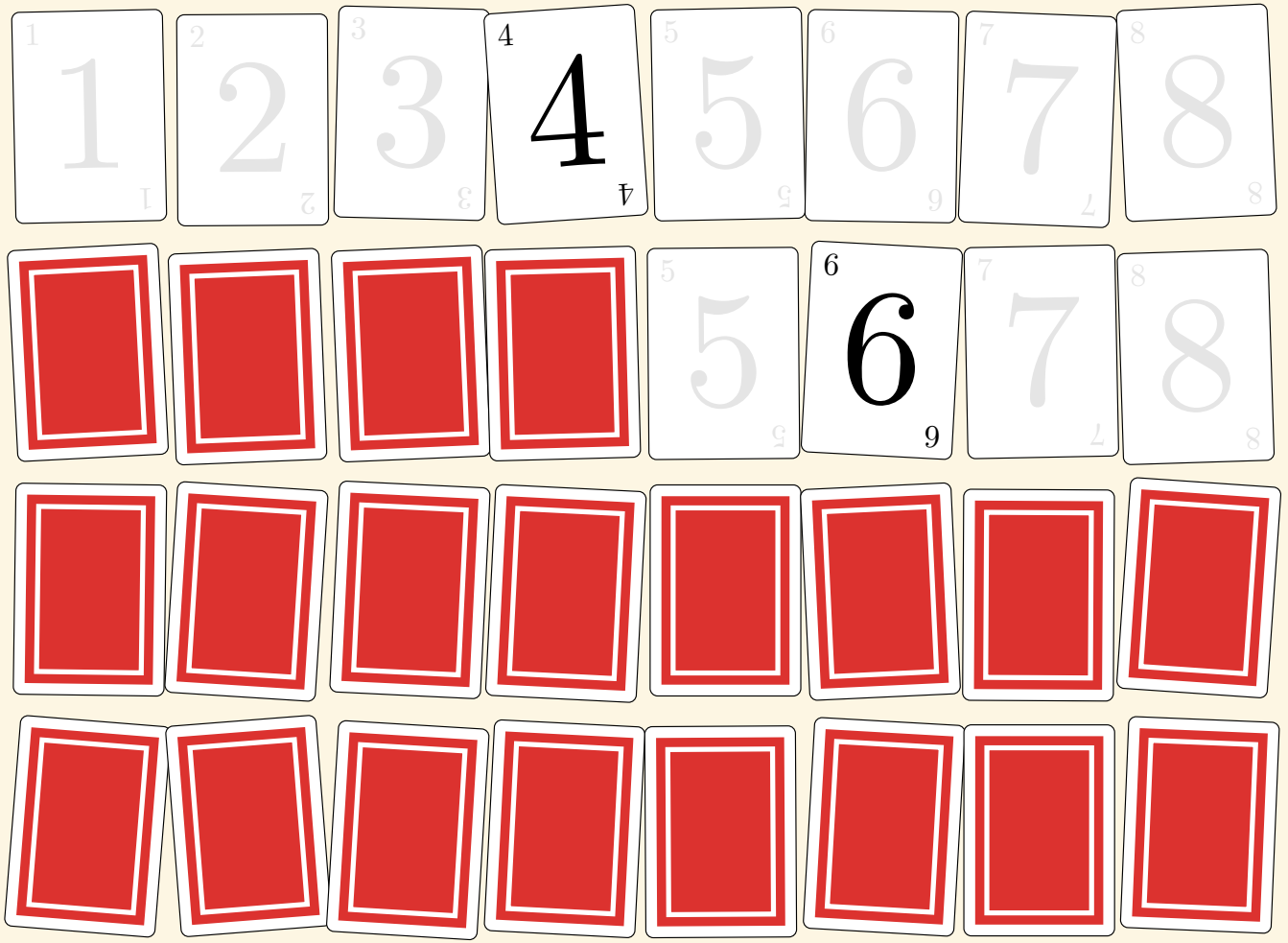
Her sjekker jeg bare hvilken halvdel jeg skal lete i – og sjekker ikke om kortet vi ser på er det vi er ute etter før vi kommer til bunnen. I pseudokoden senere har vi den sjekken i tillegg.

mlh@ntnu.no



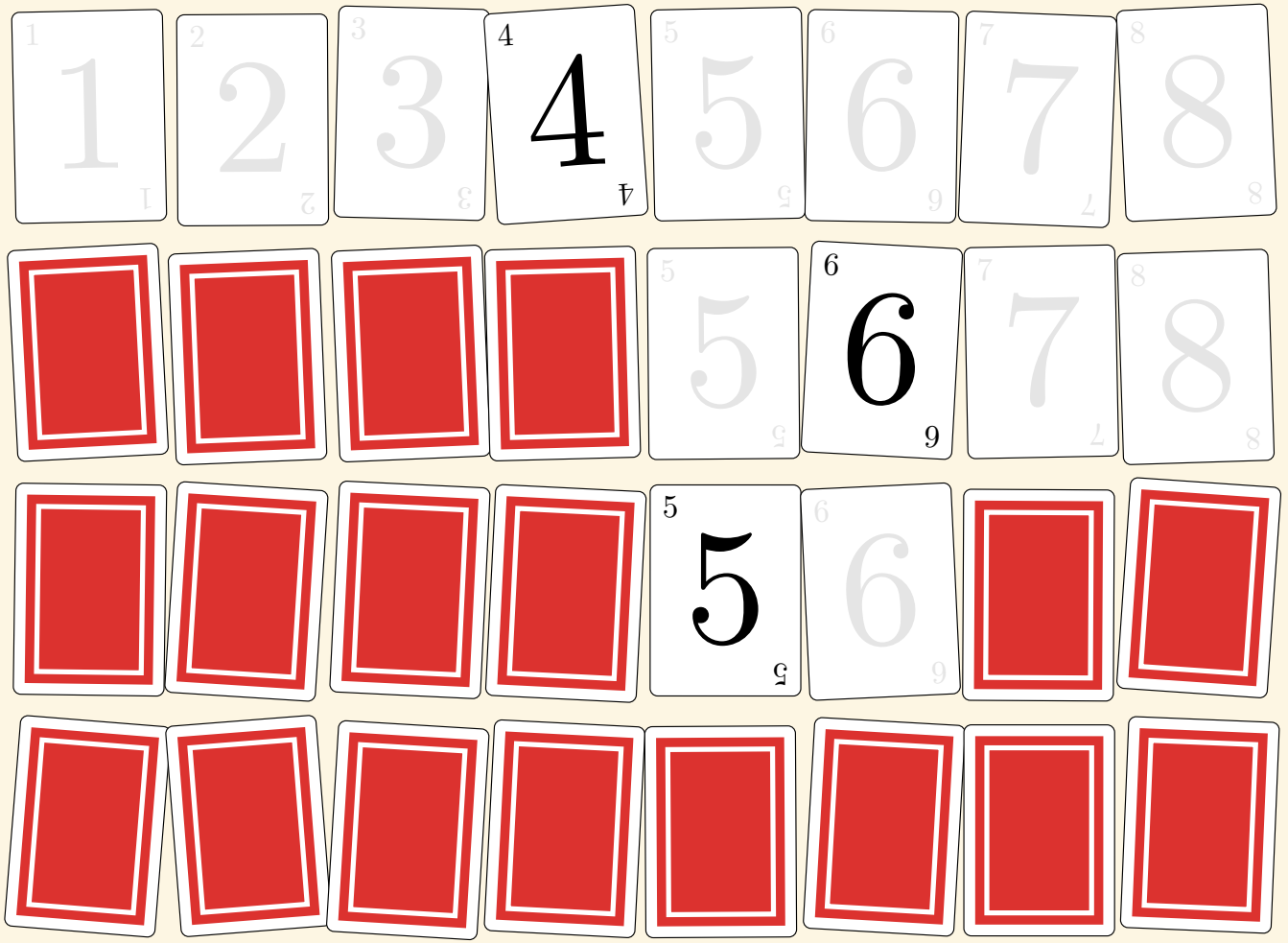
Er kortet større enn A[4]?

mlh@ntnu.no



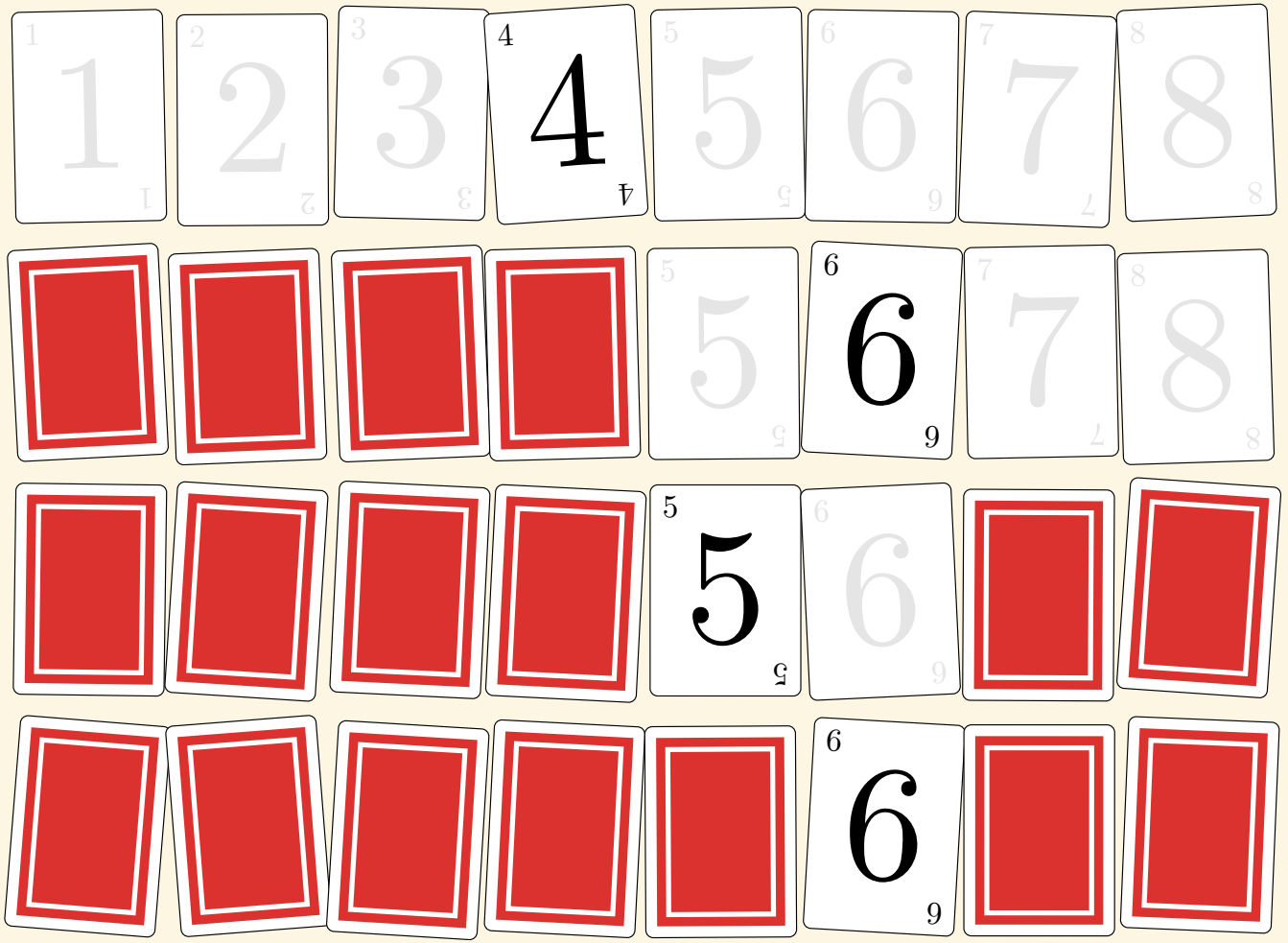
Er kortet større enn A[6]?

mlh@ntnu.no



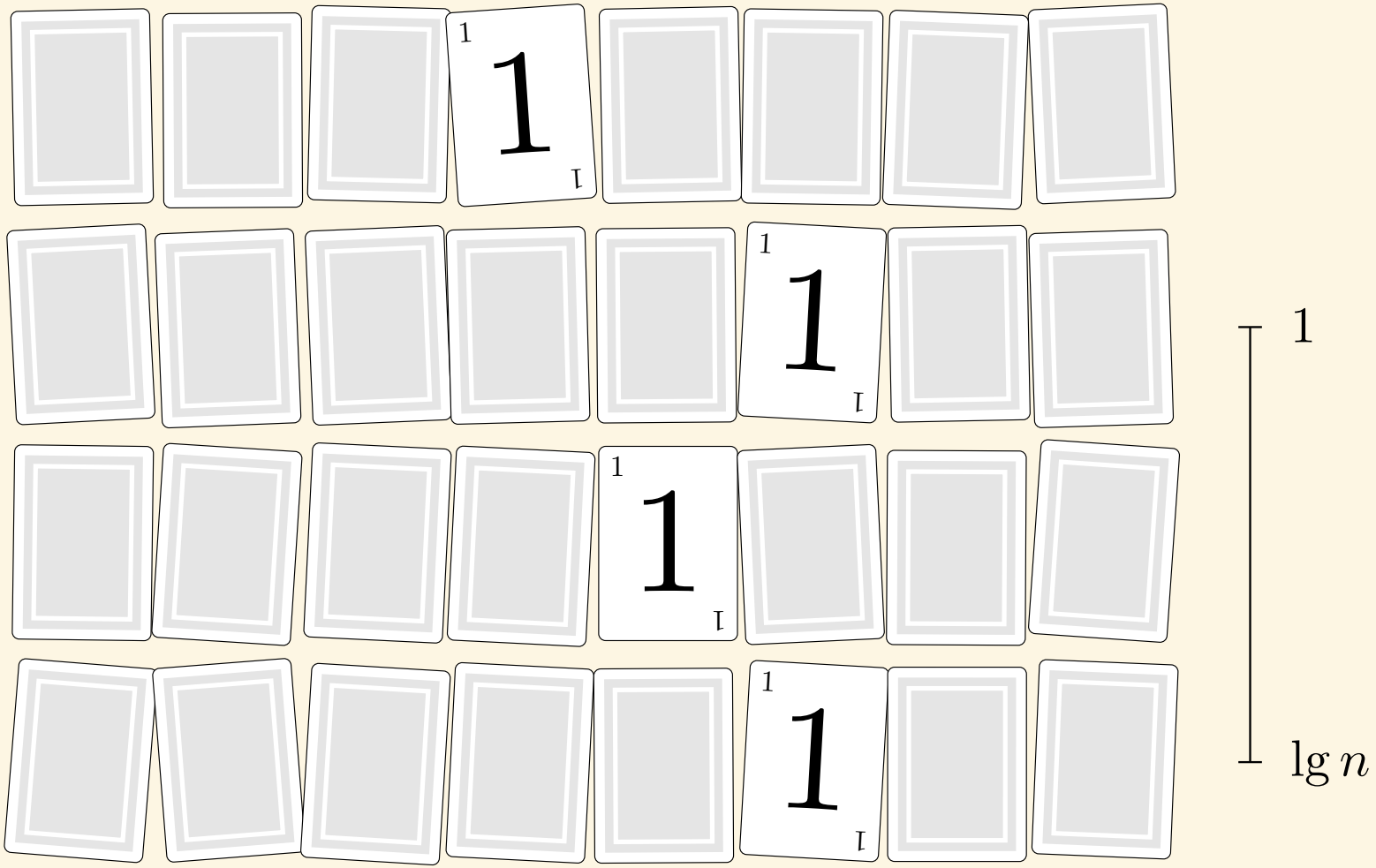
Er kortet større enn A[5]?

mlh@ntnu.no



Er A[6] kortet ditt?

mlh@ntnu.no



Antall spørsmål: $\lg n + 1$

mlh@ntnu.no

Implementasjon

- Vi kan også sjekke for likhet med midt-element, og ev. avslutte tidlig
- Må da ta høyde for tomme intervaller/tabeller, som blir nytt grunntilfelle

BISECT(A, p, r, v)

A	tabell
p	venstre
r	høyre
v	søkeverdi

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

mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
```

A tabell
p venstre
r høyre
v søkeverdi

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

mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
```

A tabell
p venstre
r høyre
v søkeverdi
q midten

Del på midten

mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
3        if v == A[q]
```

A tabell
p venstre
r høyre
v søkeverdi
q midten

Er dette verdien vi leter etter?

mlh@ntnu.no

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

A tabell
p venstre
r høyre
v søkeverdi
q midten

I så fall er vi ferdige

mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
3        if v == A[q]
4              return q
5        elseif v < A[q]
```

A tabell
p venstre
r høyre
v søkeverdi
q midten

Ellers: Må v ligge til venstre?

mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
3        if v == A[q]
4              return q
5        elseif v < A[q]
6              return BISECT(A, p, q − 1, v)
```

A tabell
p venstre
r høyre
v søkeverdi
q midten

Let videre til venstre

mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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)
```

A tabell
p venstre
r høyre
v søkeverdi
q midten

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

mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

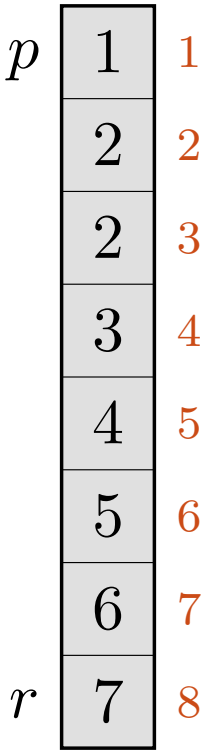
A tabell
p venstre
r høyre
v søkeverdi
q midten

Vi fant ikke v, så vi returnerer NIL

mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

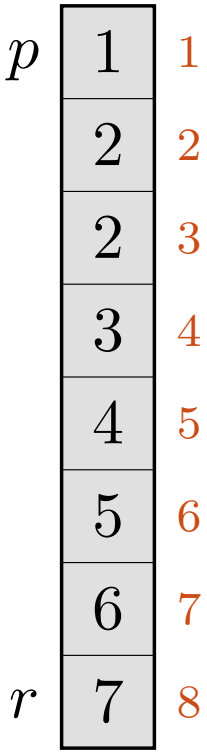
$p, r, v \;=\; 1, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

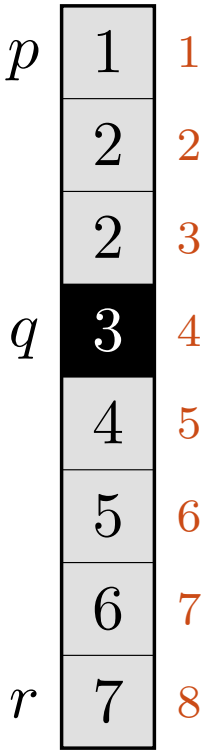
$p, r, v \;=\; 1, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

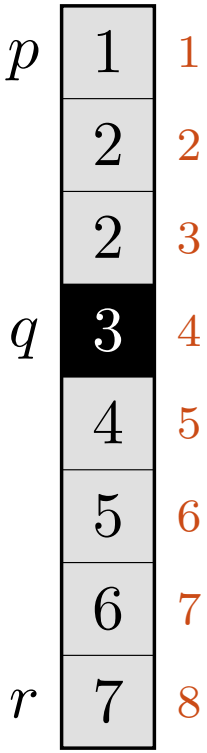
$p, r, v \;=\; 1, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

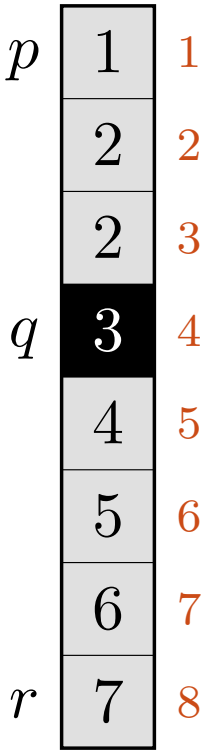
$p, r, v \;=\; 1, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

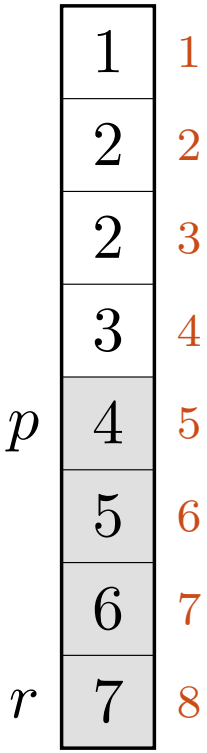
$p, r, v \;=\; 1, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

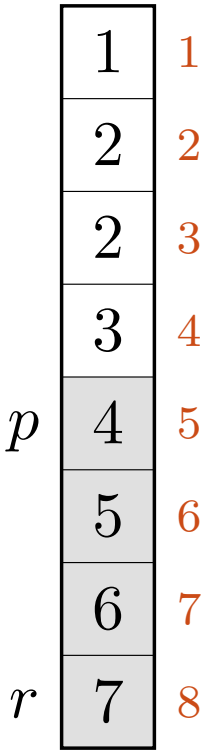
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

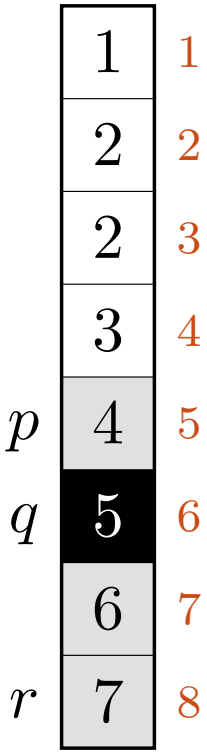
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

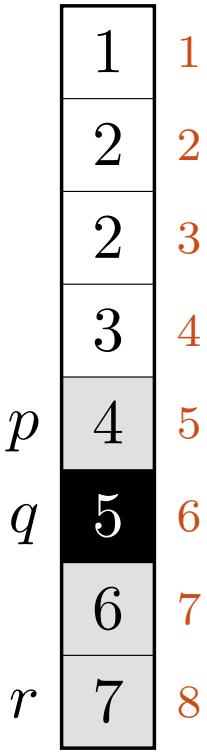
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

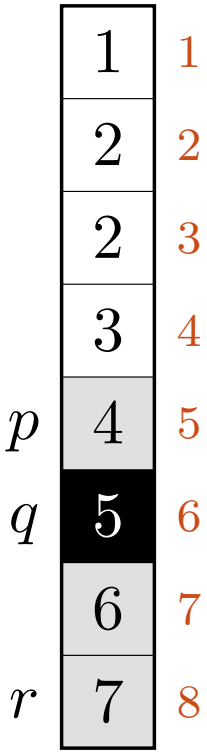
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

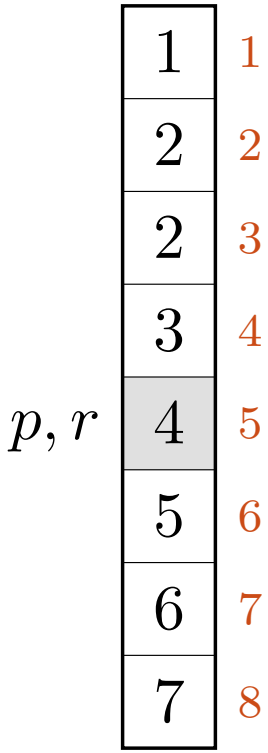
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

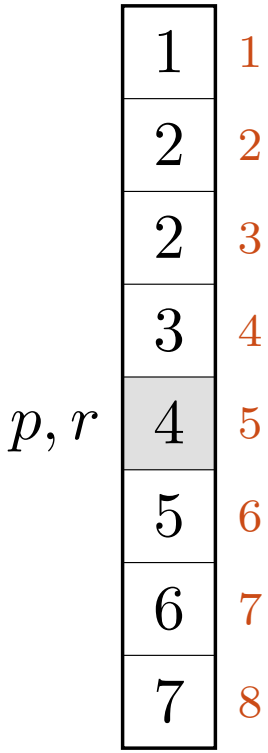
$p, r, v \;=\; 1, 8, 4 \rightarrow 5, 8, 4 \rightarrow 5, 5, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

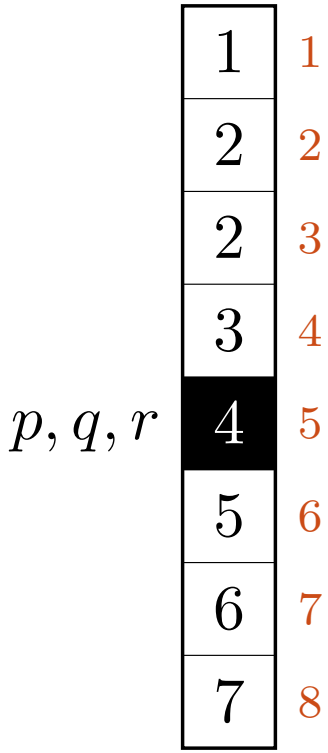
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4 \;\rightarrow\; 5, 5, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

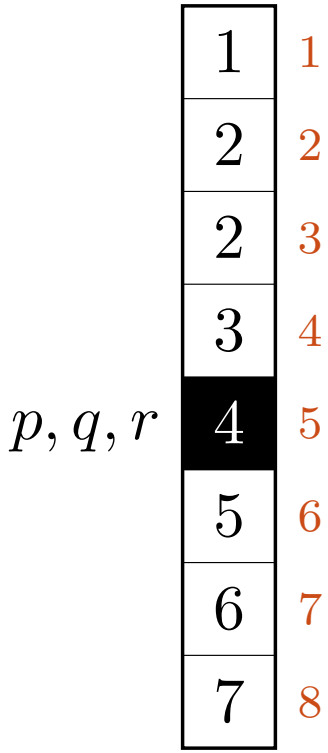
$p, r, v \;=\; 1, 8, 4 \rightarrow 5, 8, 4 \rightarrow 5, 5, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

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

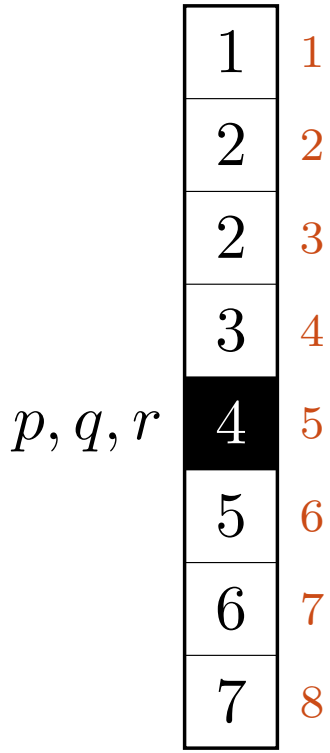


mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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

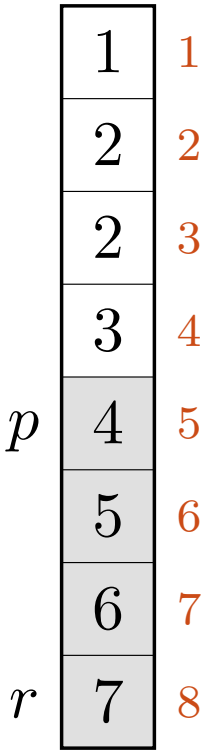
p, r, v = 1, 8, 4 › 5, 8, 4 › 5, 5, 4



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

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

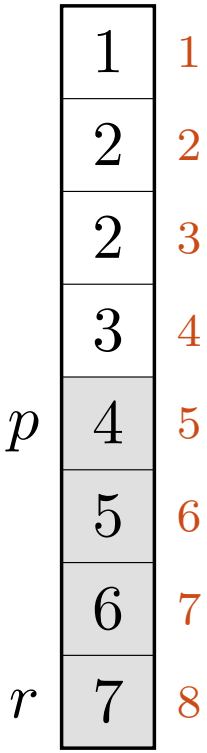


mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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

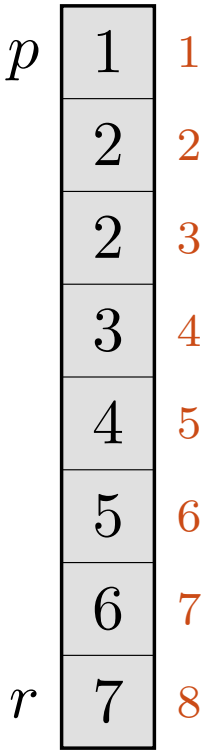
$p, r, v = 1, 8, 4 \rightarrow 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

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

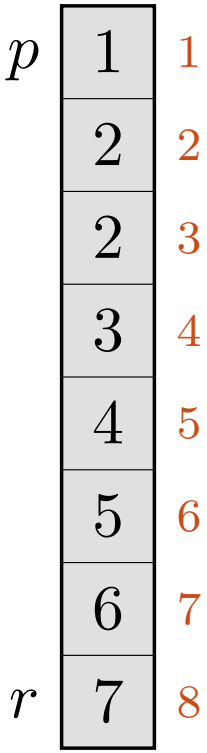


mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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

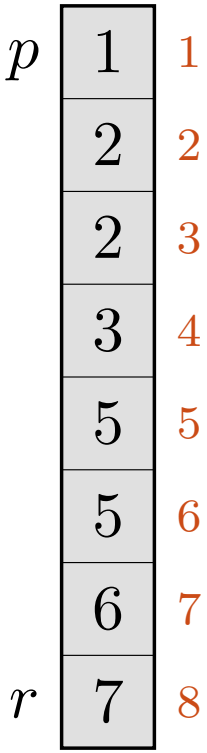
$p, r, v = 1, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

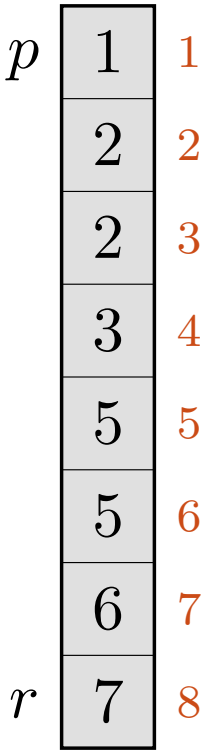
$p, r, v \;=\; 1, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

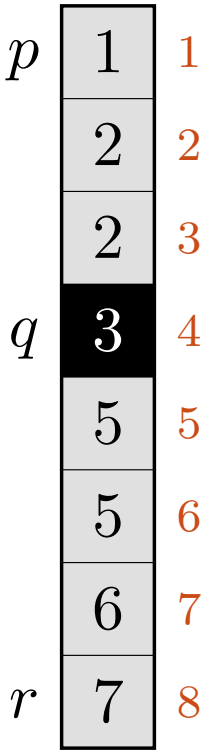
$p, r, v \;=\; 1, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

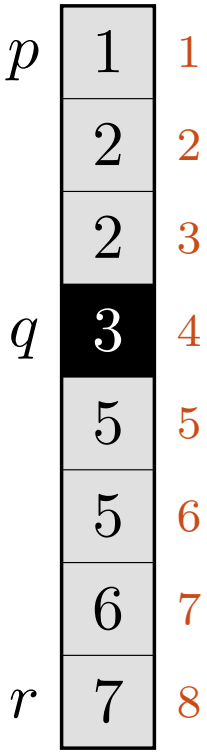
$p, r, v \;=\; 1, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

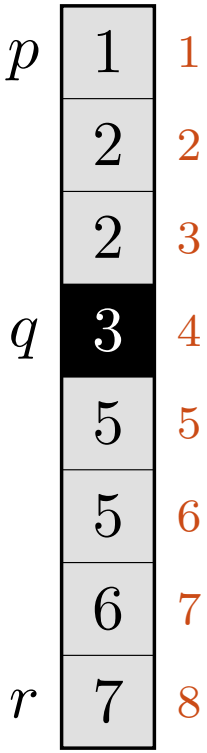
$p, r, v \;=\; 1, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

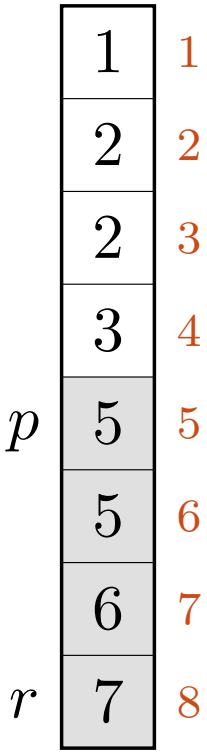
$p, r, v \;=\; 1, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

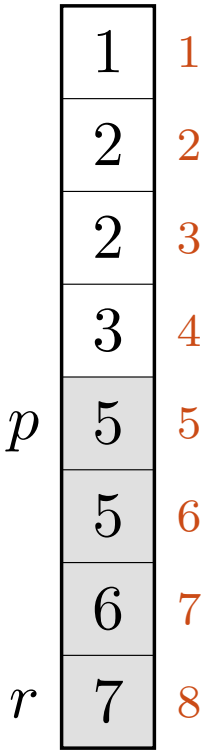
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

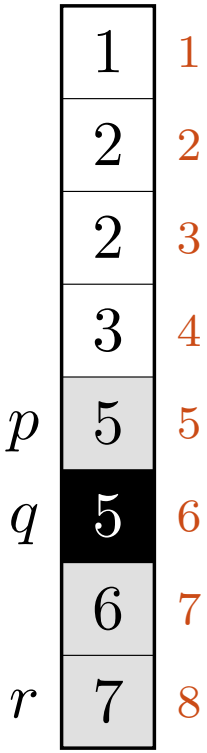
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

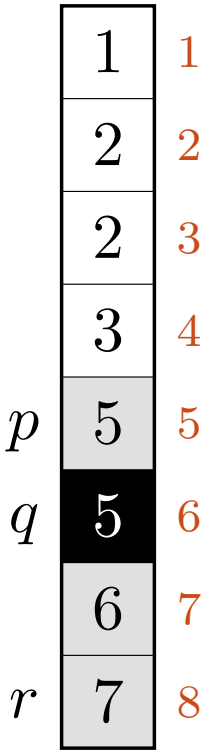
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

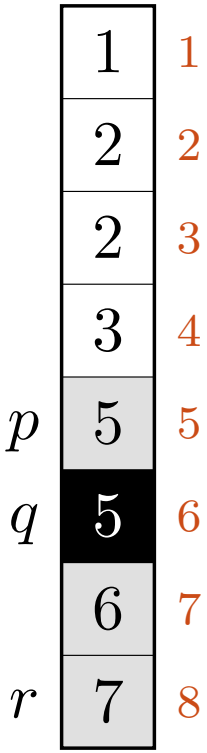
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

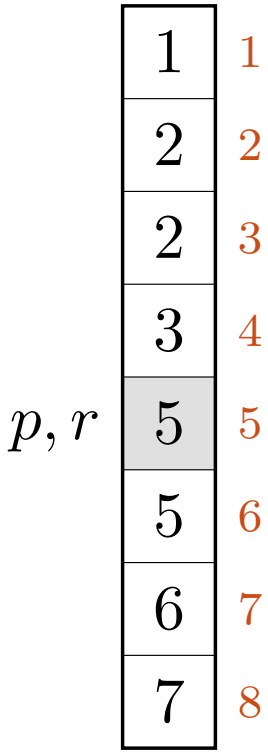
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

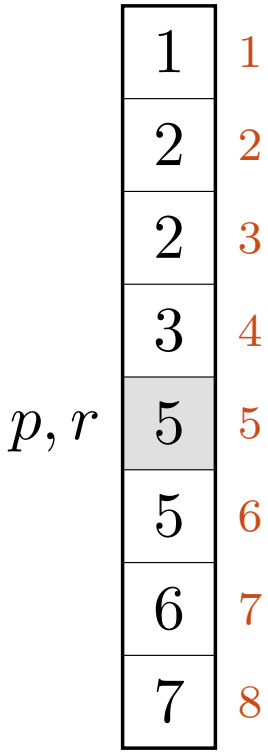
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4 \;\rightarrow\; 5, 5, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

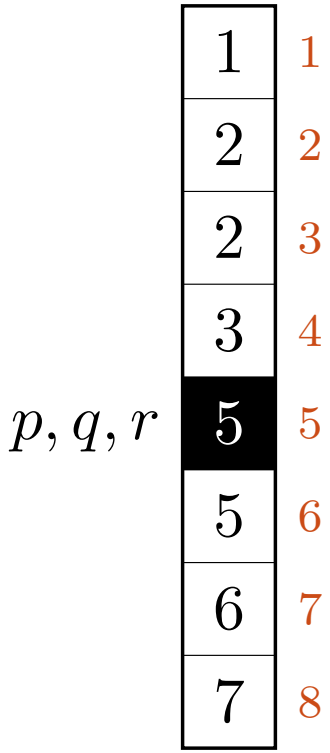
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4 \;\rightarrow\; 5, 5, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

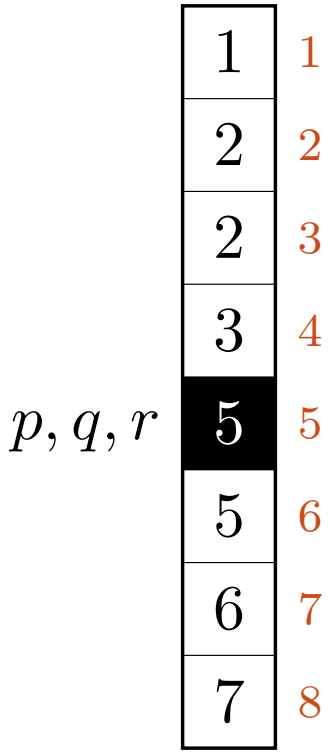
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4 \;\rightarrow\; 5, 5, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

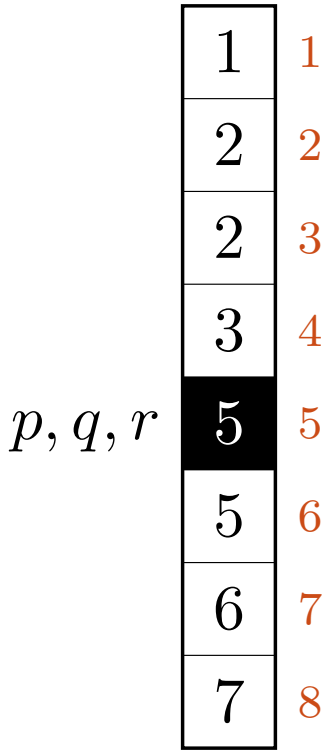
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4 \;\rightarrow\; 5, 5, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

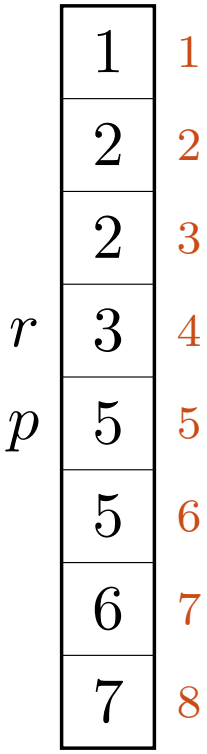
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4 \;\rightarrow\; 5, 5, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

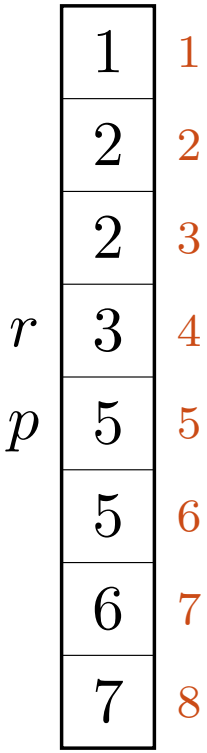
$p, r, v \;=\; 1, 8, 4 \rightarrow 5, 8, 4 \rightarrow 5, 5, 4 \rightarrow \mathbf{5, 4, 4}$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

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

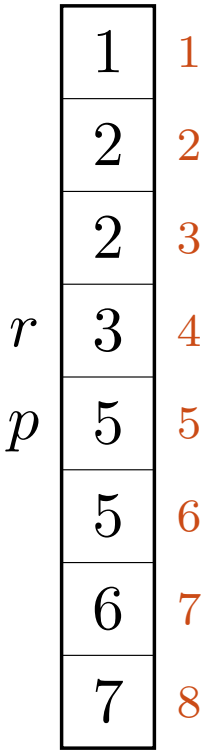


mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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

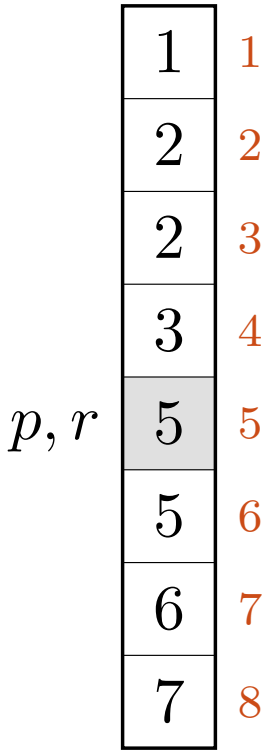
p, r, v = 1, 8, 4 › 5, 8, 4 › 5, 5, 4 › **5, 4, 4**



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

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

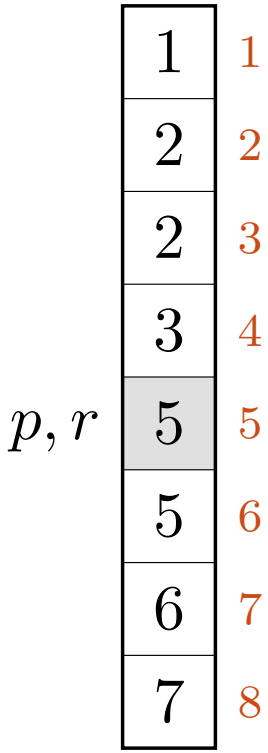


mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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

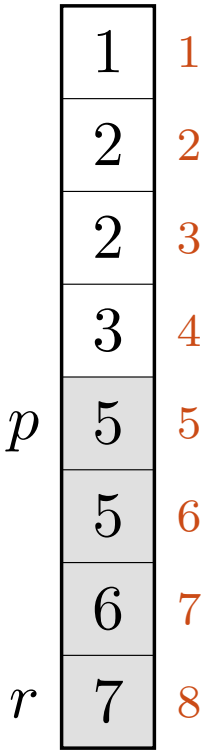
$p, r, v \;=\; 1, 8, 4 \;\rightarrow\; 5, 8, 4 \;\rightarrow\; 5, 5, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

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

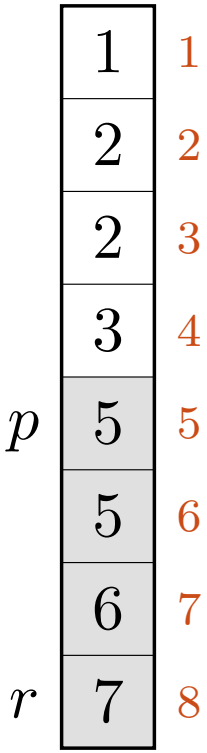


mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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

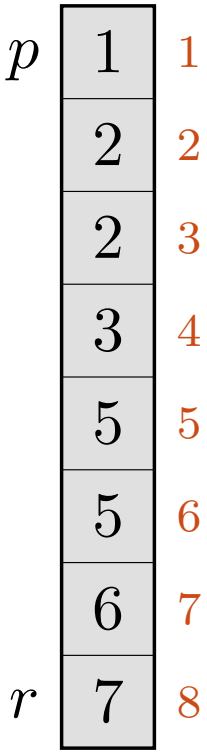
$p, r, v = 1, 8, 4 \rightarrow 5, 8, 4$



mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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
```

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

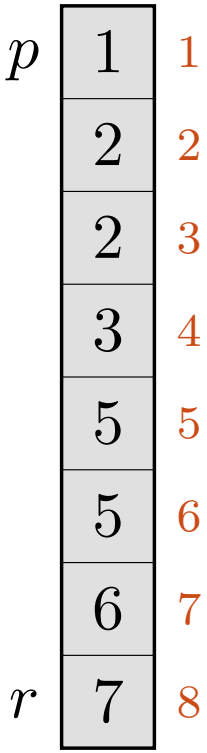


mlh@ntnu.no

```
BISECT(A, p, r, v)
1  if p ⩽ r
2        q = ⌊(p + r)/2⌋
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

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



mlh@ntnu.no

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

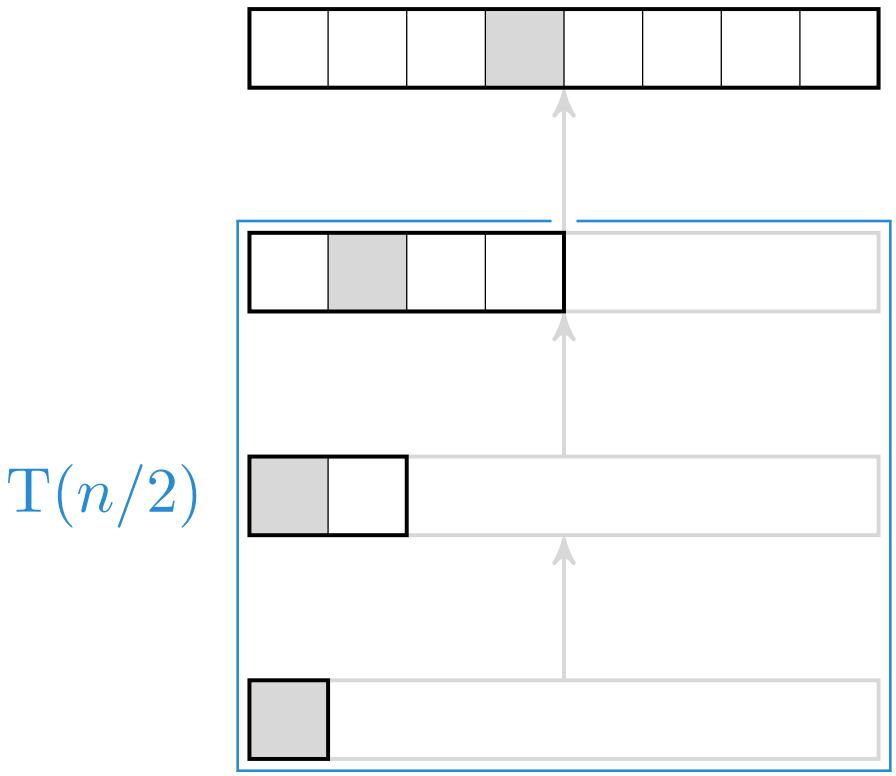
Her gjør jeg endel antagelser/forenklinger, f.eks. at delproblemet har størrelse n/2 og ikke n/2 – 1 – og at vi ikke finner v før vi har kommet til bunnen. Disse antagelsene har ikke noe å si for den asymptotiske worst-case-kjøretiden.

mlh@ntnu.no

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

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

mlh@ntnu.no



mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

$$+ 1 \qquad (1)$$

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

mlh@ntnu.no

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

$$+ 1 \tag{1}$$

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

mlh@ntnu.no

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

$$+ 1 \tag{1}$$

$$+ 1 \tag{2}$$

$$+ \mathrm{T}(n/8) \tag{3}$$

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

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

mlh@ntnu.no

Verifikasjon

Med substitusjon/induksjon

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

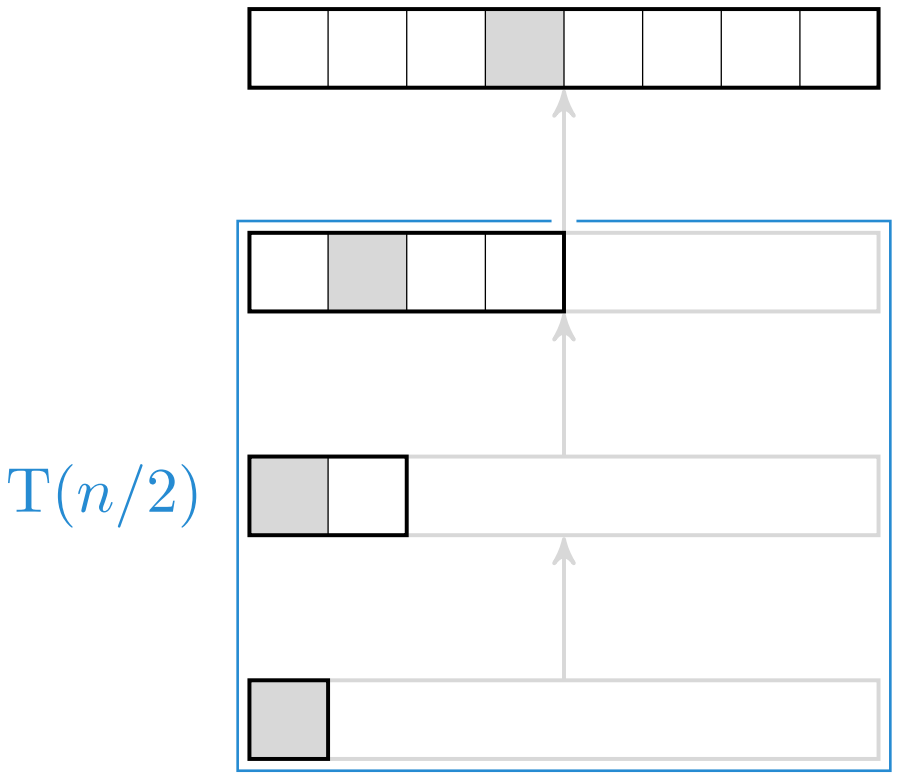
Gitt $\mathrm{T}(k) = \lg k + 1$ for $k < n$, vis $\mathrm{T}(n) = \lg n + 1$

mlh@ntnu.no

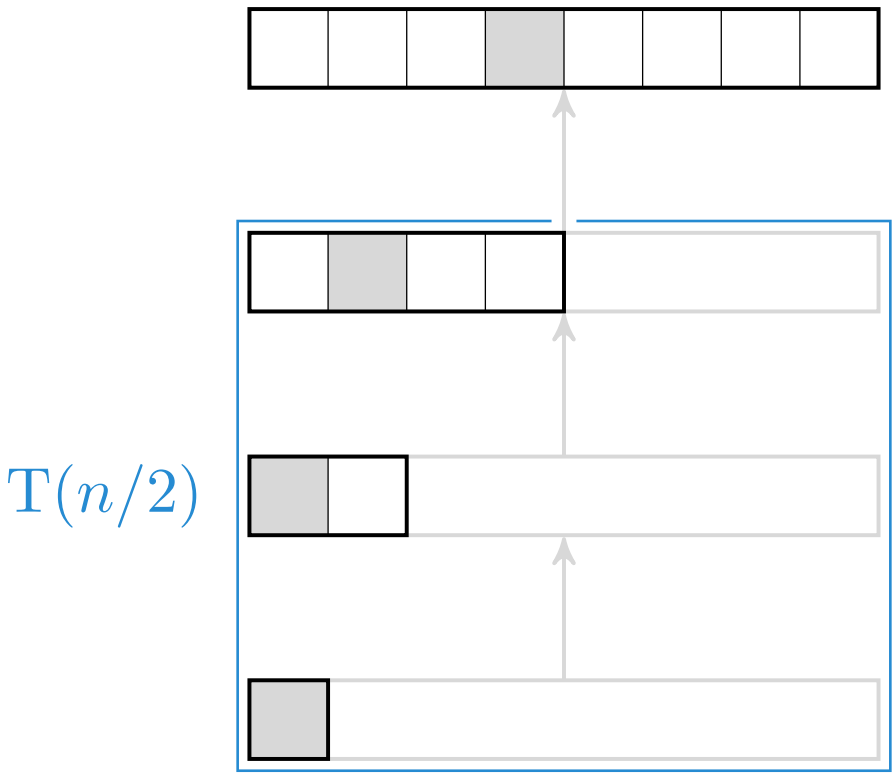
$$
\begin{aligned}
\mathrm{T}(n) &= \mathrm{T}(n/2) + 1 \\
&= (\lg \frac{n}{2} + 1) + 1 \\
&= \lg n - \lg 2 + 2 \\
&= \lg n + 1
\end{aligned}
$$

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

mlh@ntnu.no



mlh@ntnu.no



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

mlh@ntnu.no

«[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

3:5

Merge sort

Kalles også «flettesortering» på norsk.

Instans	Løsning
Dekomponering	Kombinasjon IS
Delinstanser	Delløsninger IH
Grunntilfelle ...	Grunnløsning ...

mlh@ntnu.no

Instans	Løsning	
Dekomponering	Kombinasjon	IS
Delinstanser	Delløsninger	IH
...	...	
Grunntilfelle	Grunnløsning	

Hva er input?

mlh@ntnu.no

Instans En sekvens A	Løsning
Dekomponering	Kombinasjon IS
Delinstanser	Delløsninger IH
Grunntilfelle	Grunnløsning

Hva er input?

mlh@ntnu.no

Instans En sekvens A	Løsning
Dekomponering	Kombinasjon IS
Delinstanser	Delløsninger IH
Grunntilfelle ...	Grunnløsning ...

Hva ønsker vi at output skal være?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Delinstanser	Delløsninger IH
Grunntilfelle	Grunnløsning

Hva ønsker vi at output skal være?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Delinstanser	Delløsninger IH
...	...
Grunntilfelle	Grunnløsning

Hvordan kan vi krympe eller dele opp instansen?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering Del på midten	Kombinasjon IS
Delinstanser …	Delløsninger … IH
Grunntilfelle	Grunnløsning

Hvordan kan vi krympe eller dele opp instansen?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Del på midten	
Delinstanser	Delløsninger IH
...	...
Grunntilfelle	Grunnløsning

Hva har vi brutt problemet ned til?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Del på midten	
Delinstanser	Delløsninger IH
Venstre og høyre halvdel	
...	...
Grunntilfelle	Grunnløsning

Hva har vi brutt problemet ned til?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Del på midten	
Delinstanser	Delløsninger IH
Venstre og høyre halvdel	
Grunntilfelle	Grunnløsning

Hvor stopper dekomponeringen?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	**Kombinasjon** IS
Del på midten	
Delinstanser	**Delløsninger** IH
Venstre og høyre halvdel	
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	

Hvor stopper dekomponeringen?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	**Kombinasjon** IS
Del på midten	
Delinstanser	**Delløsninger** IH
Venstre og høyre halvdel	
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	

Hva er løsningen da?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	**Kombinasjon** IS
Del på midten	
Delinstanser	**Delløsninger** IH
Venstre og høyre halvdel	
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	Samme sekvens, allerede sortert

Hva er løsningen da?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Del på midten	
Delinstanser	Delløsninger IH
Venstre og høyre halvdel	
Grunntilfelle	Grunnløsning
Sekvens med maks ett element	Samme sekvens, allerede sortert

Hva er løsningene for delinstansene?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Del på midten	
Delinstanser	Delløsninger IH
Venstre og høyre halvdel	Venstre og høyre halvdel, sorterte
Grunntilfelle	Grunnløsning
Sekvens med maks ett element	Samme sekvens, allerede sortert

Hva er løsningene for delinstansene?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	**Kombinasjon** IS
Del på midten	
Delinstanser	**Delløsninger** IH
Venstre og høyre halvdel	Venstre og høyre halvdel, sorterte
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	Samme sekvens, allerede sortert

Hvordan bygger vi en full løsning av delløsninger?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	**Kombinasjon** IS
Del på midten	Flett venstre og høyre halvdel
Delinstanser	**Delløsninger** IH
Venstre og høyre halvdel	Venstre og høyre halvdel, sorterte
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	Samme sekvens, allerede sortert

Hvordan bygger vi en full løsning av delløsninger?

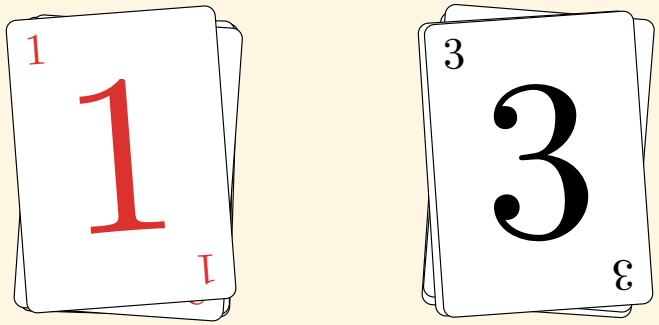
mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	**Kombinasjon** IS
Del på midten	Flett venstre og høyre halvdel
Delinstanser	**Delløsninger** IH
Venstre og høyre halvdel	Venstre og høyre halvdel, sorterte
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	Samme sekvens, allerede sortert

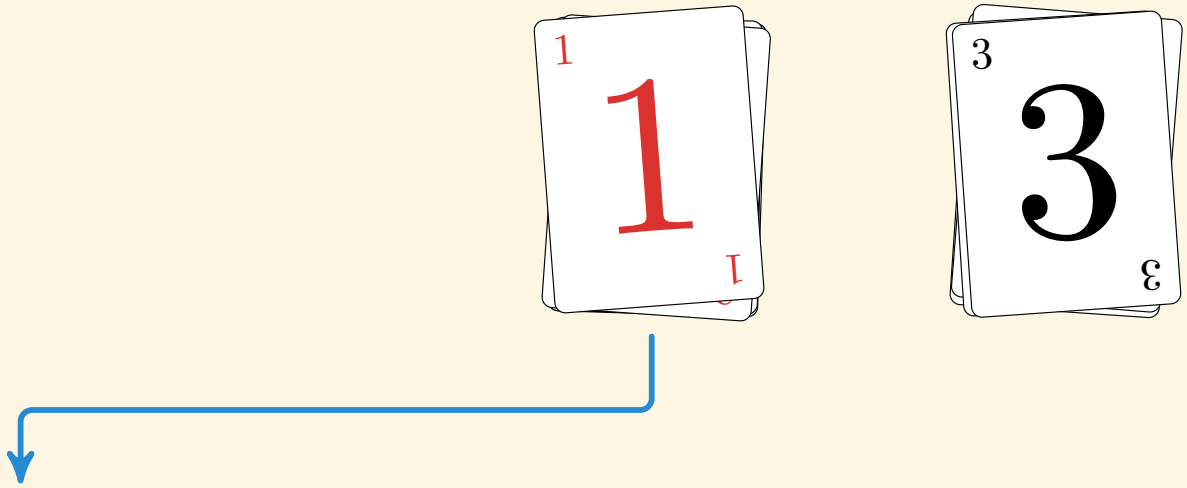
mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Del på midten	Flett venstre og høyre halvdel
Delinstanser	Delløsninger IH
Venstre og høyre halvdel	Venstre og høyre halvdel, sorterte
Grunntilfelle	Grunnløsning
Sekvens med maks ett element	Samme sekvens, allerede sortert

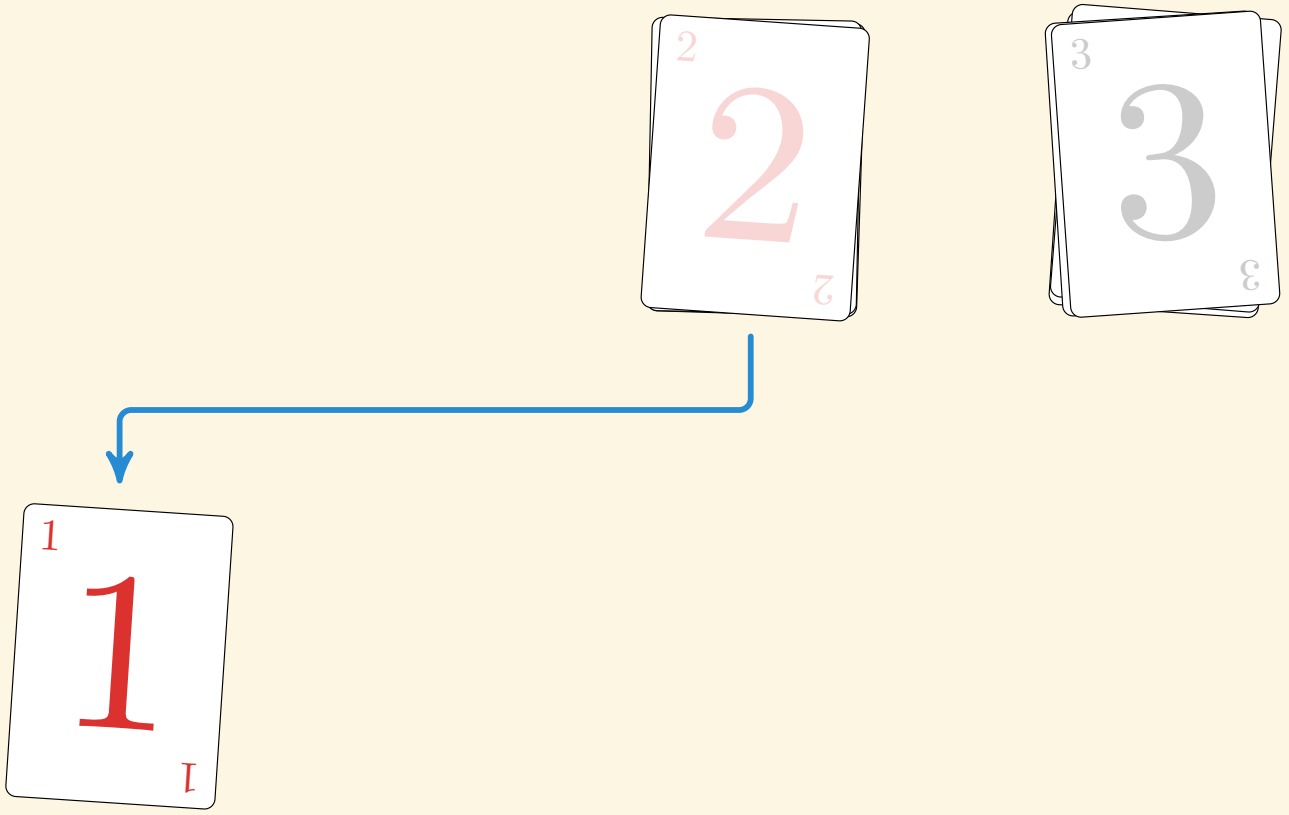
mlh@ntnu.no



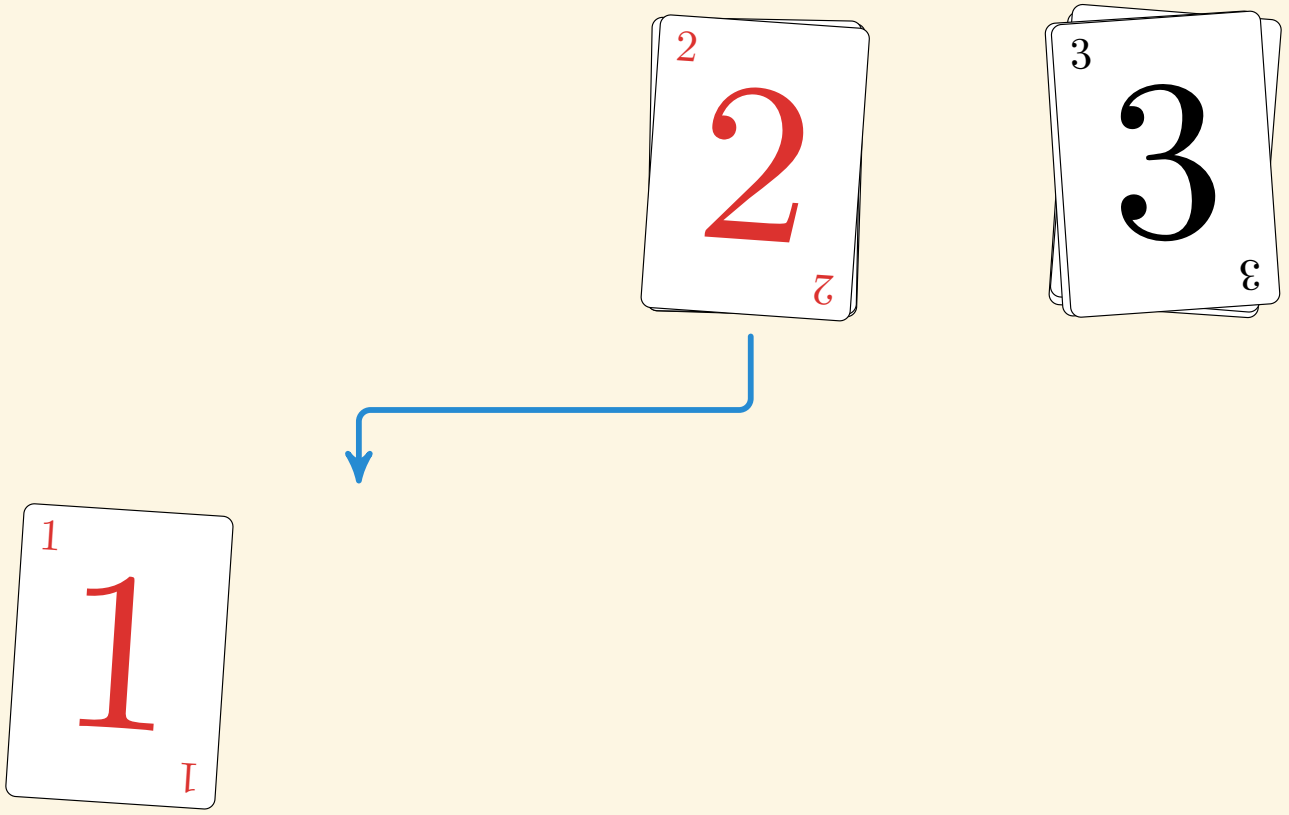
mlh@ntnu.no



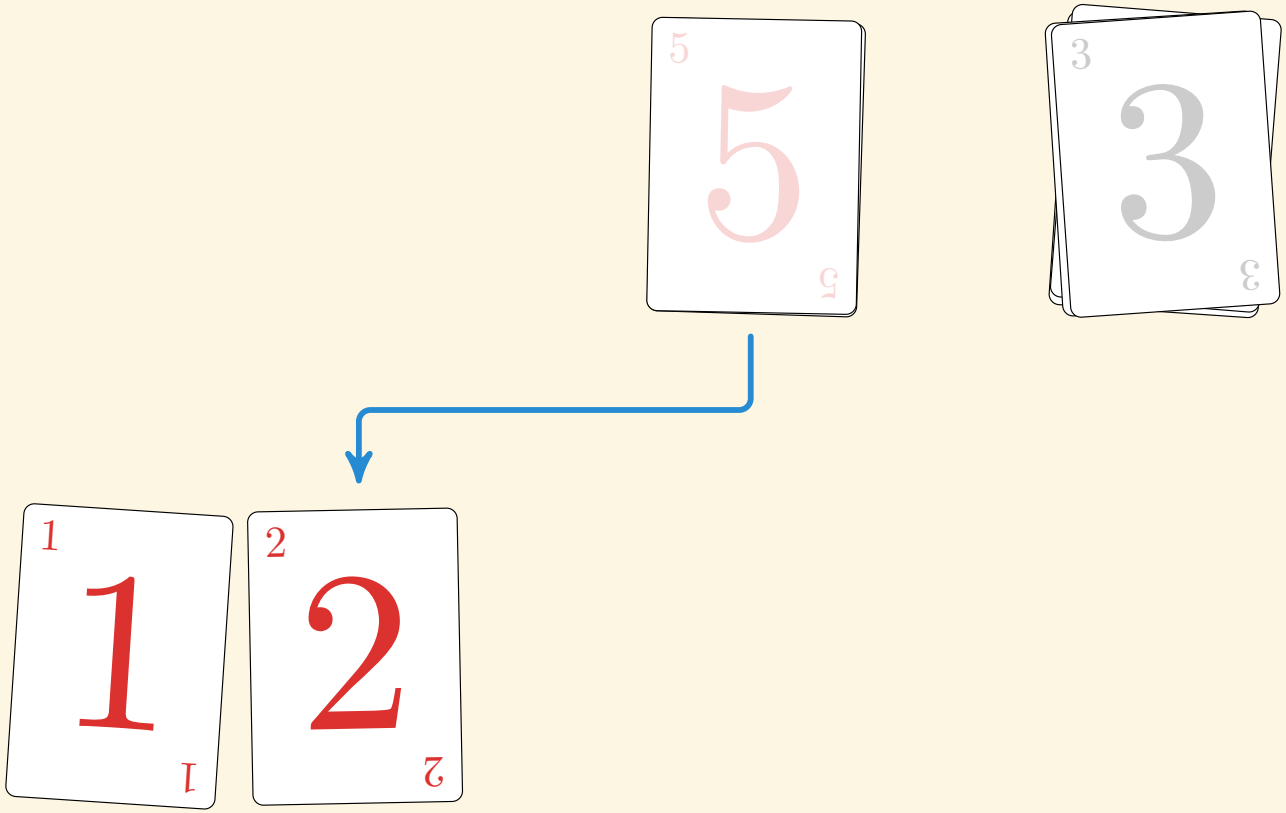
mlh@ntnu.no



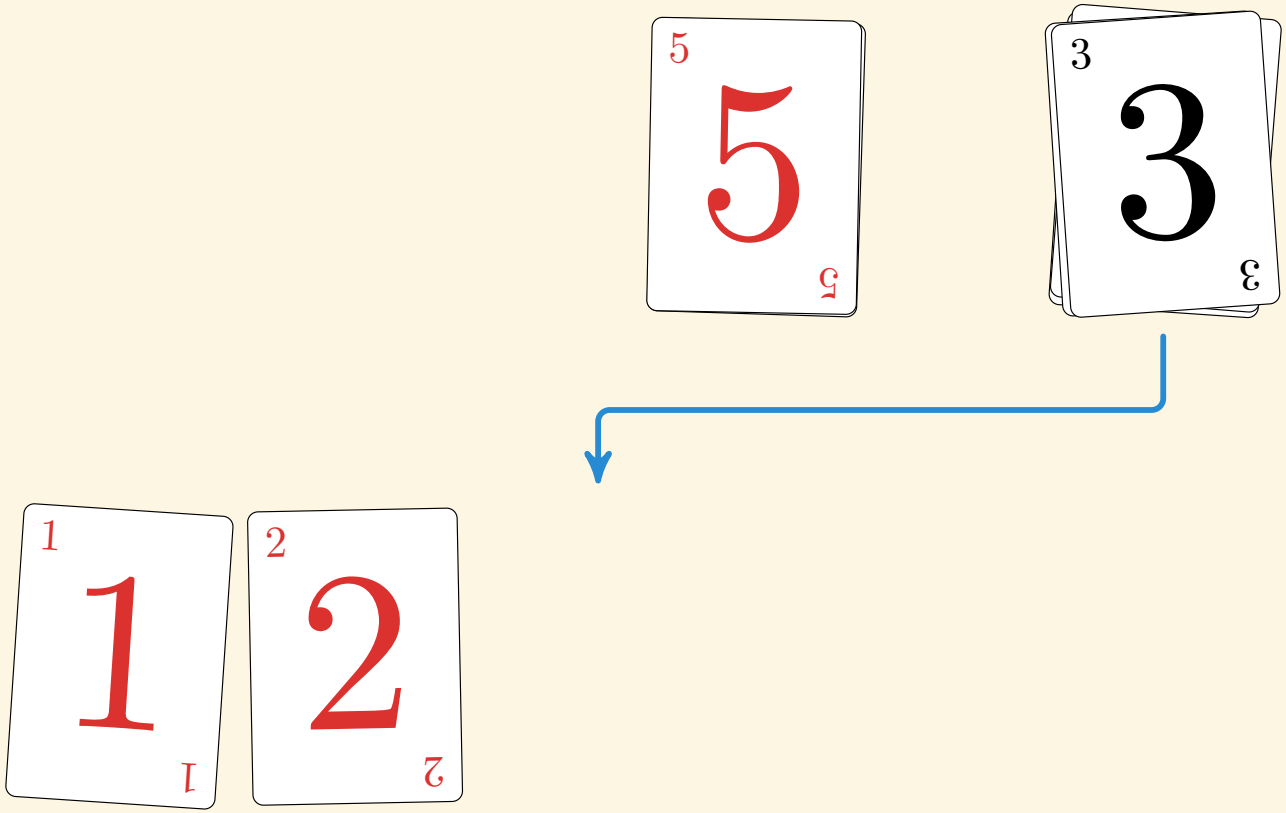
mlh@ntnu.no



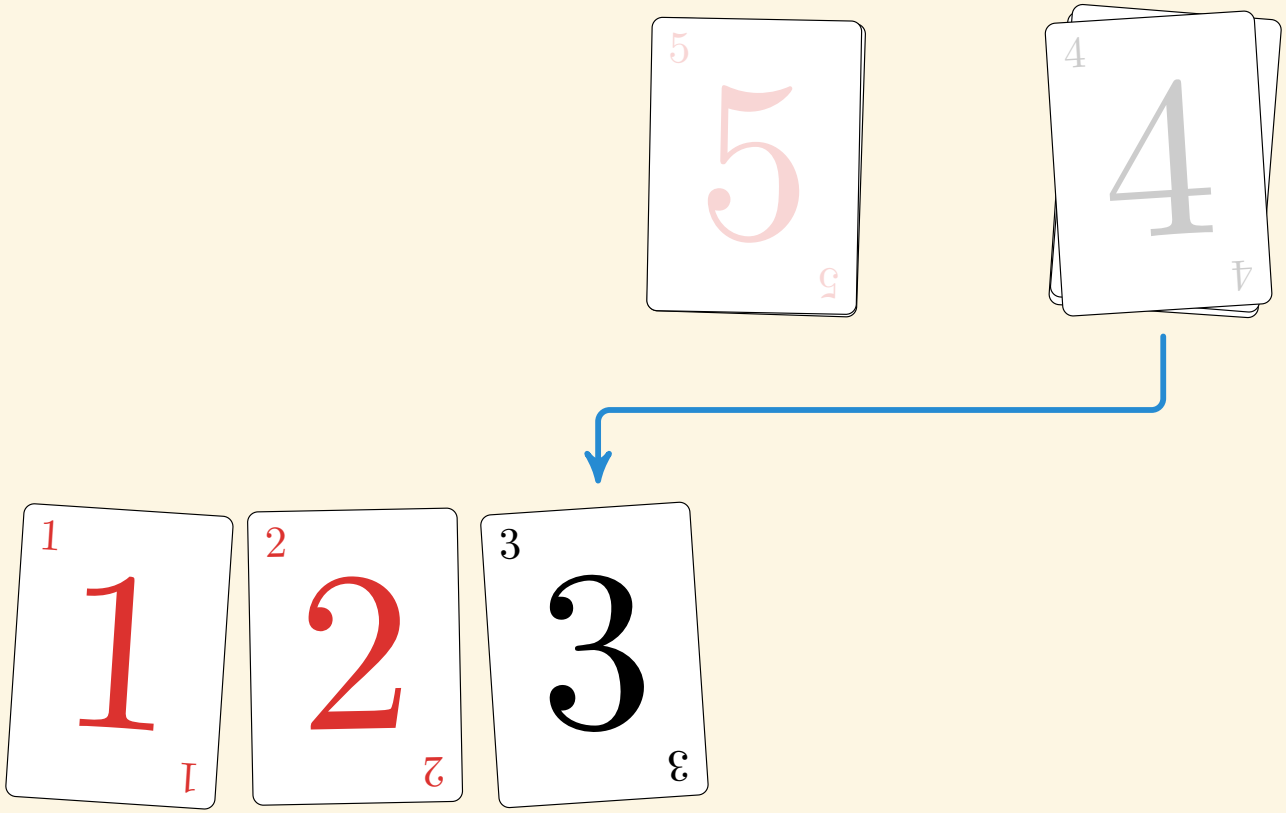
mlh@ntnu.no



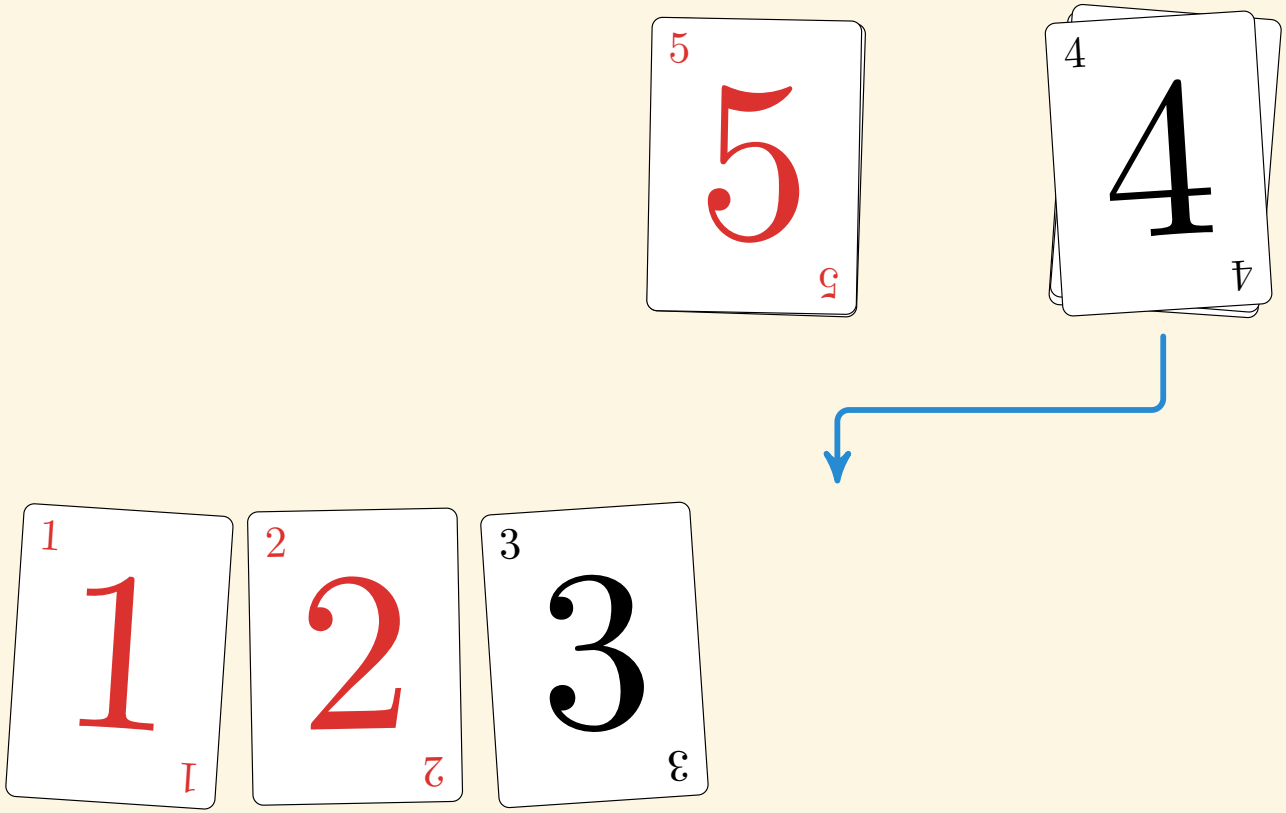
mlh@ntnu.no



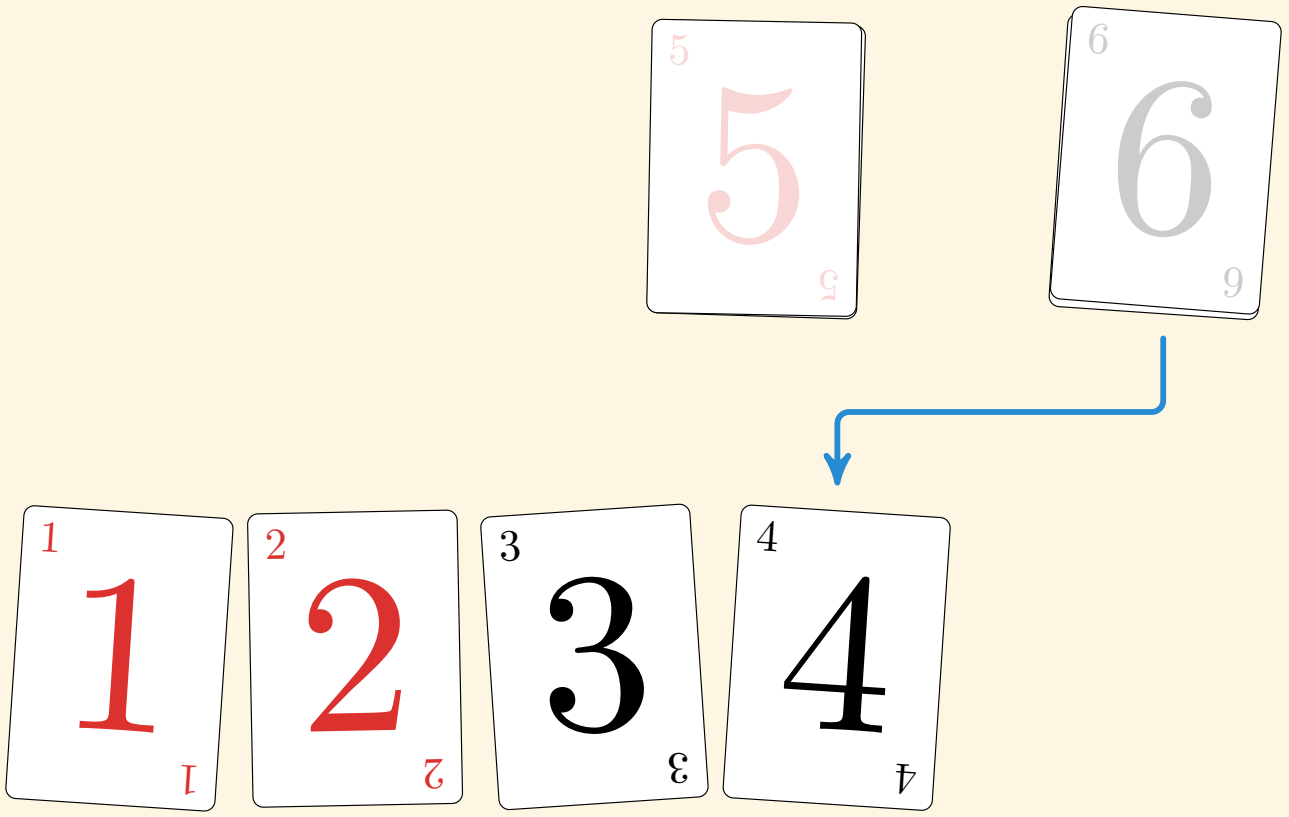
mlh@ntnu.no



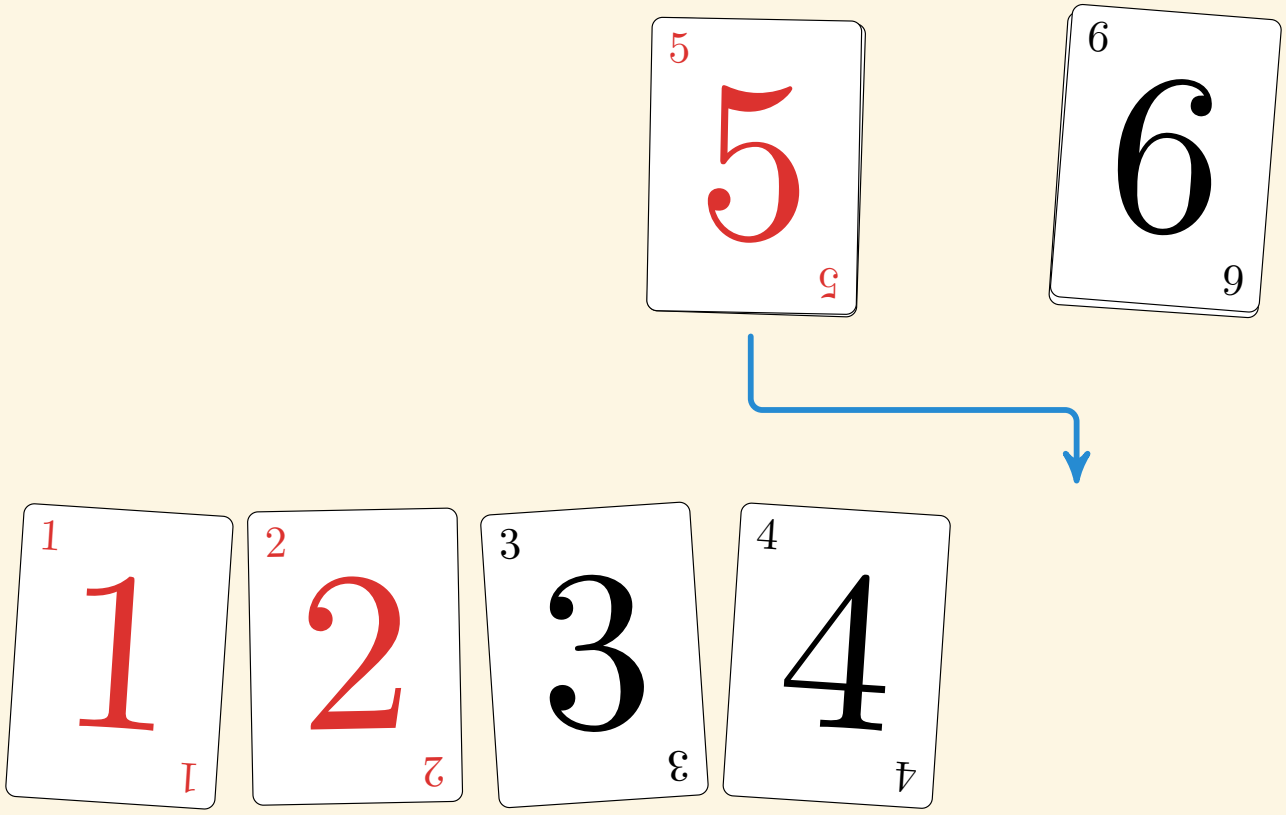
mlh@ntnu.no



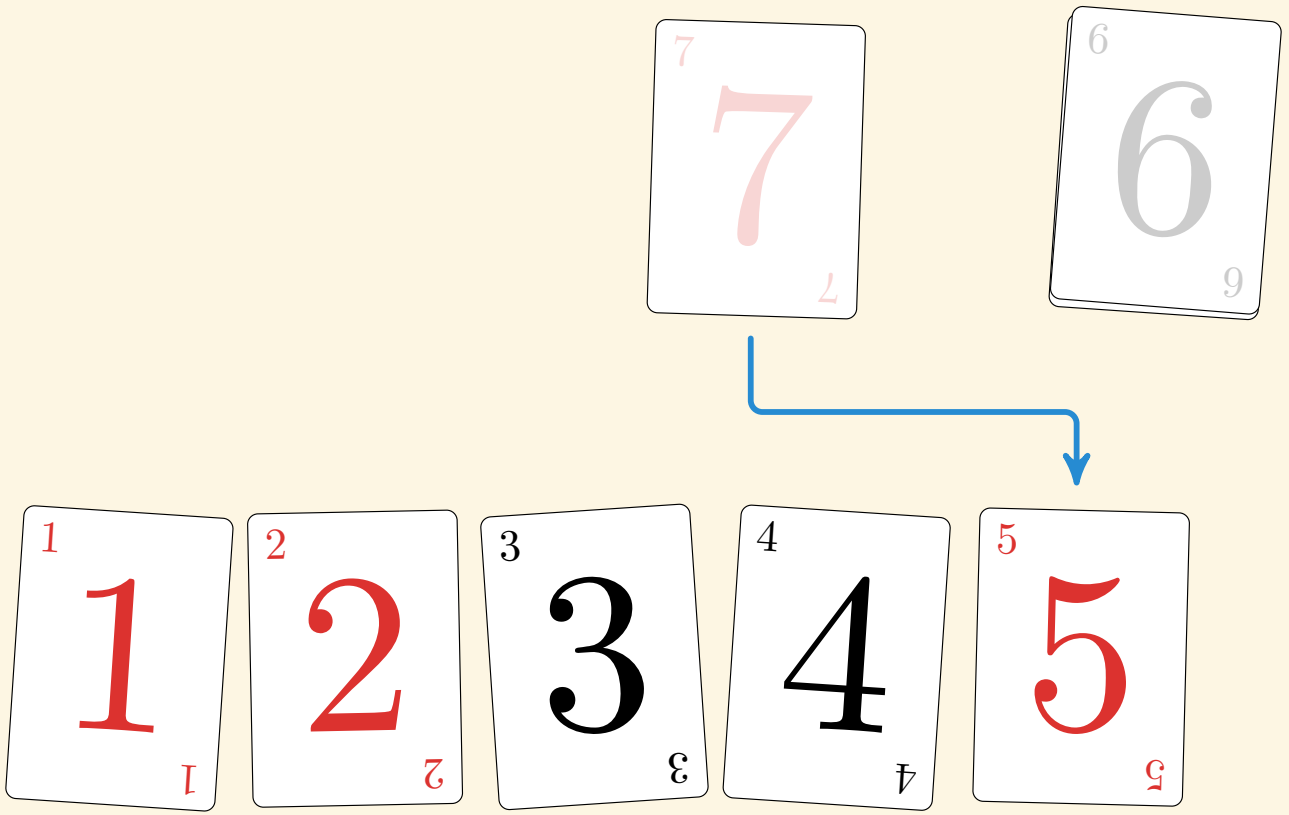
mlh@ntnu.no



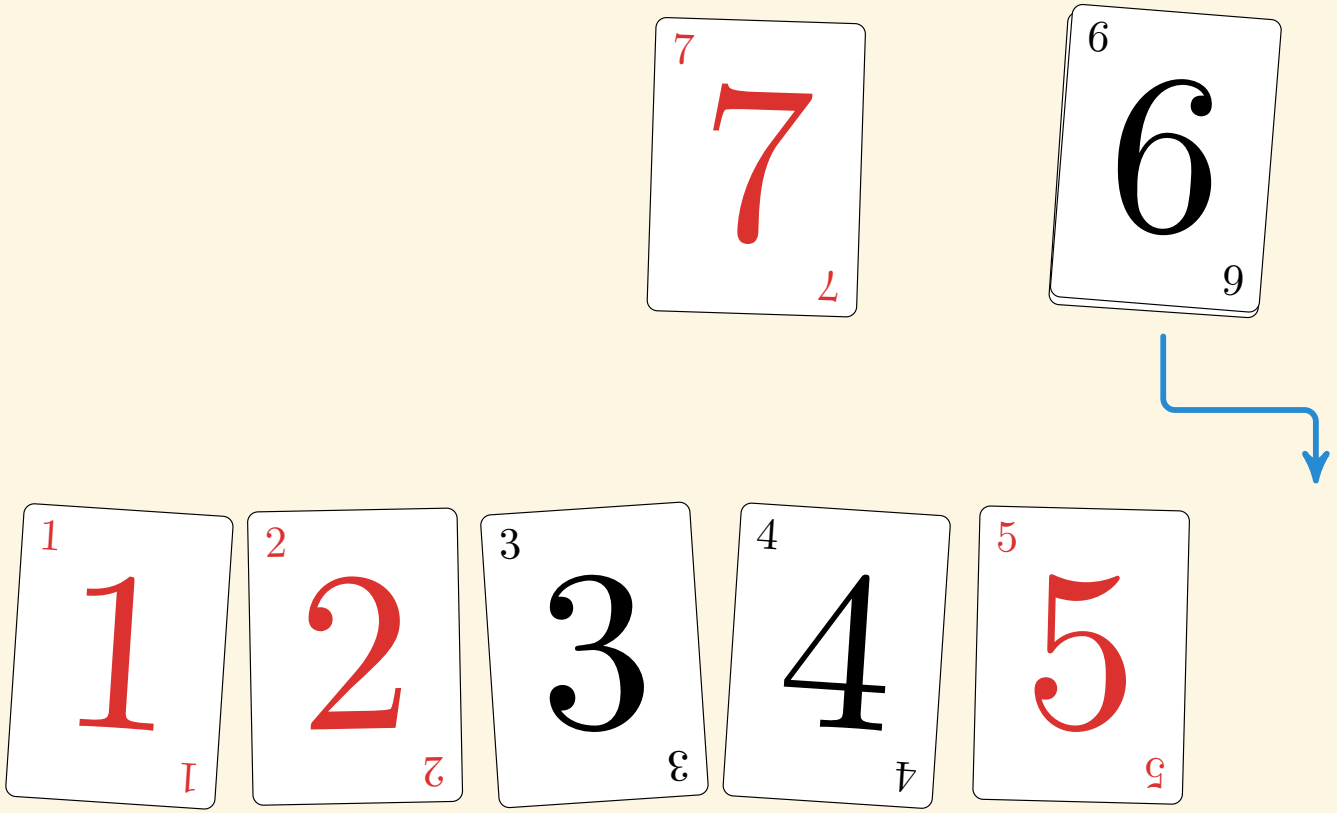
mlh@ntnu.no



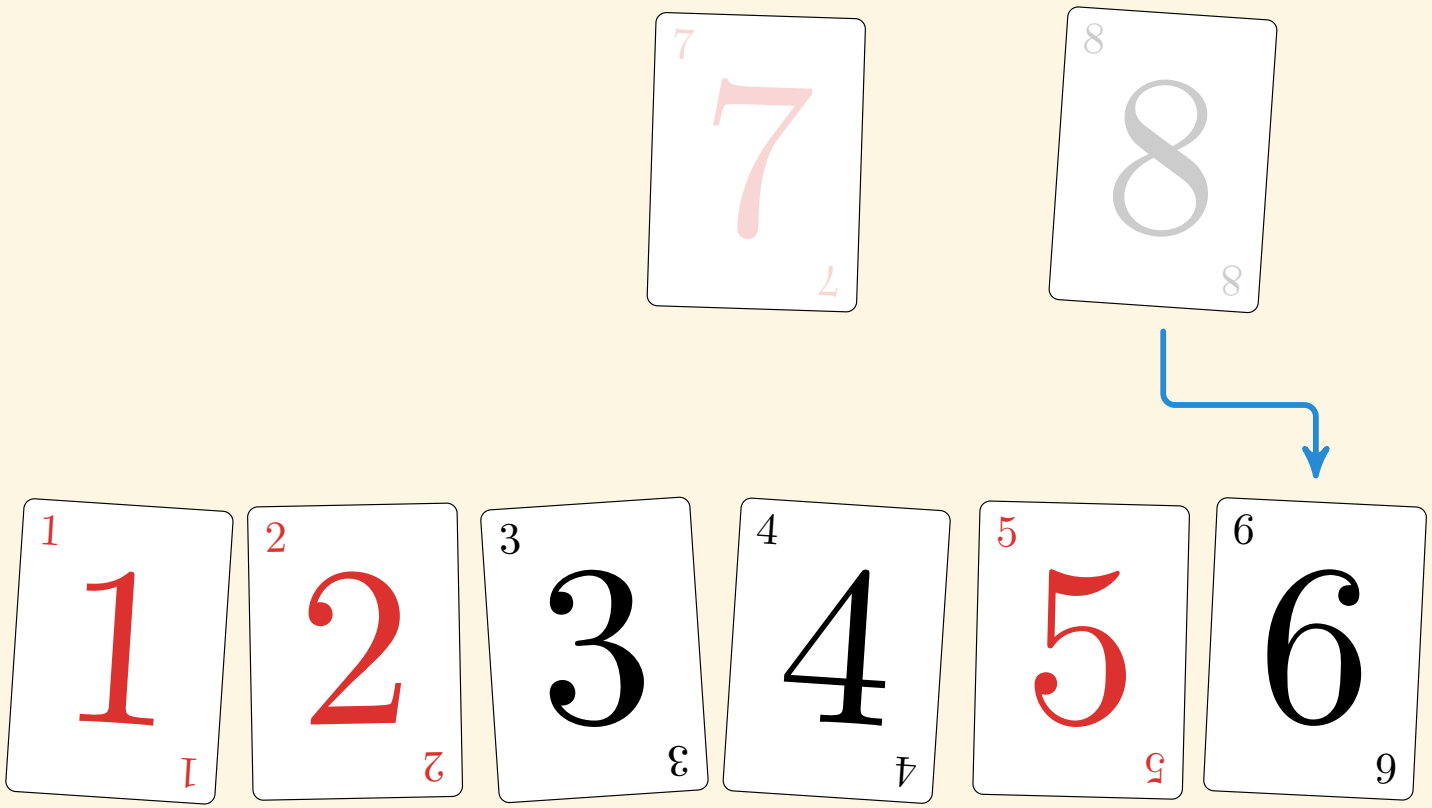
mlh@ntnu.no



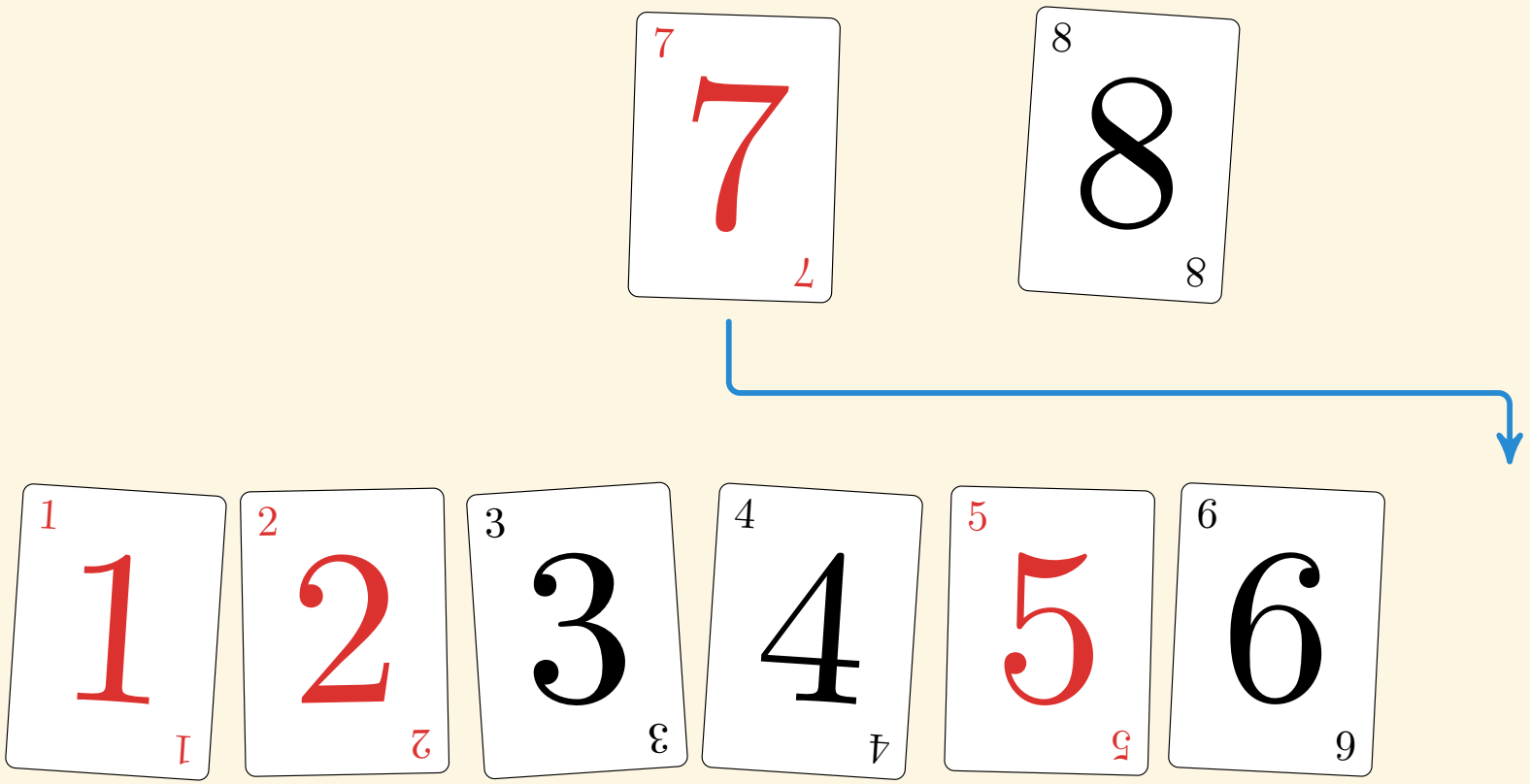
mlh@ntnu.no



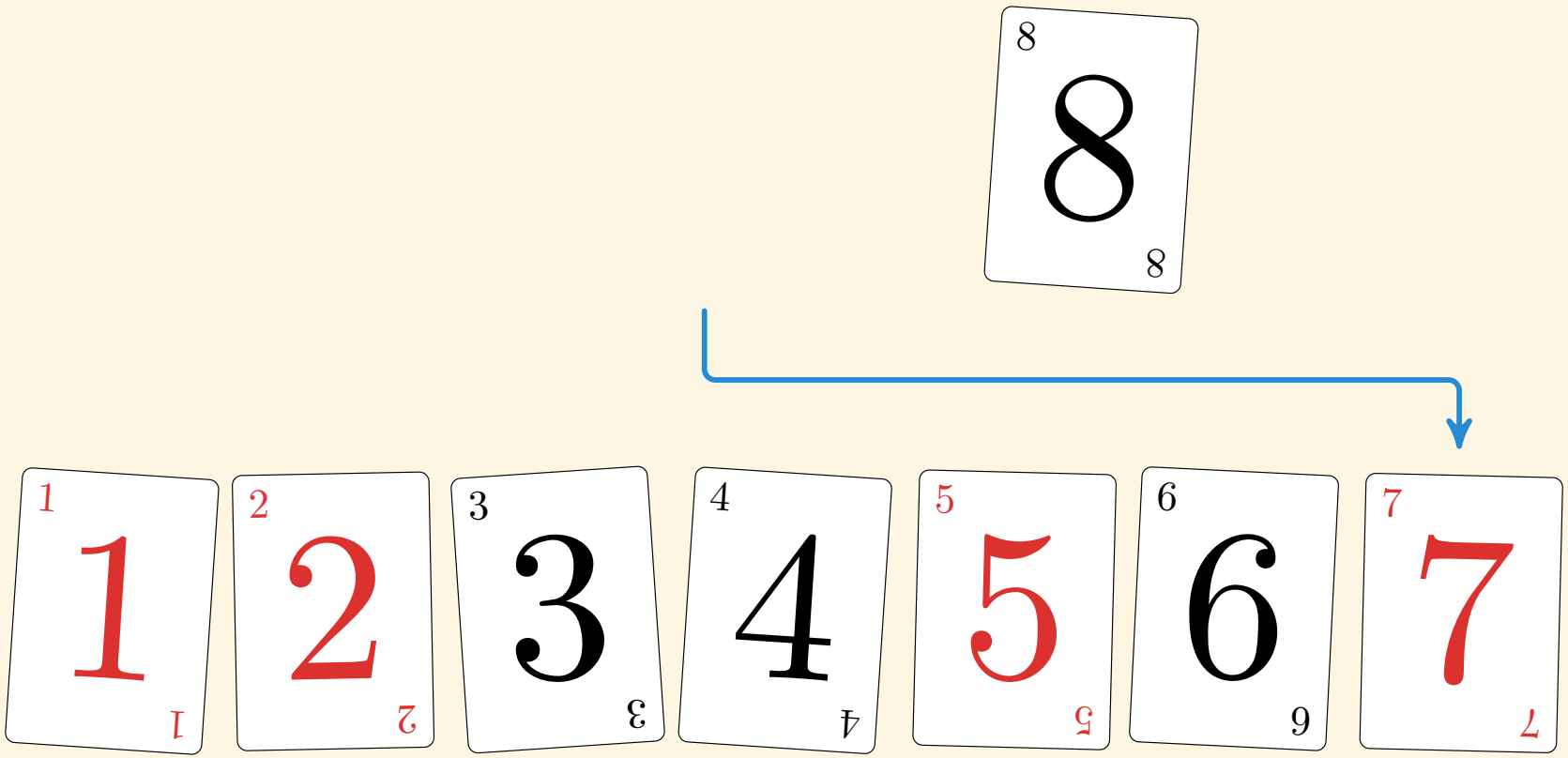
mlh@ntnu.no



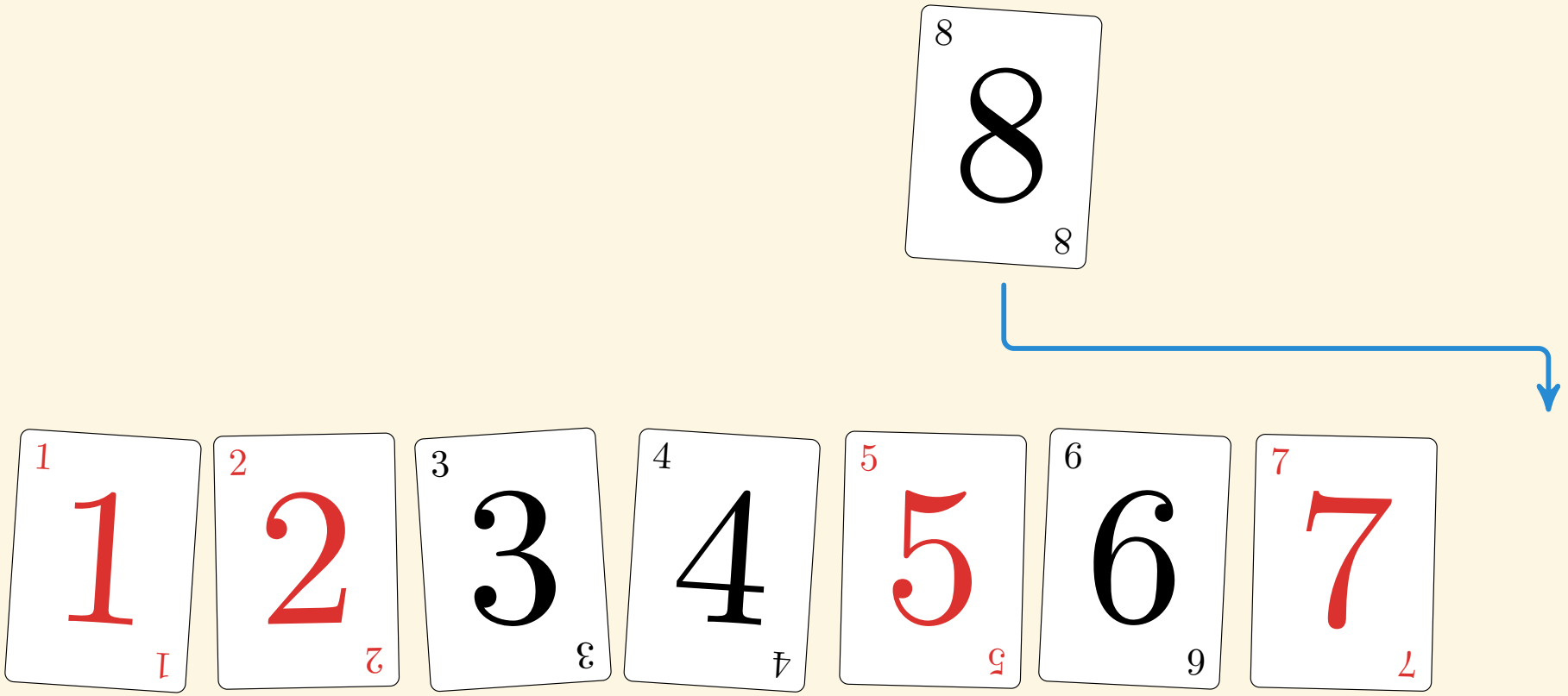
mlh@ntnu.no



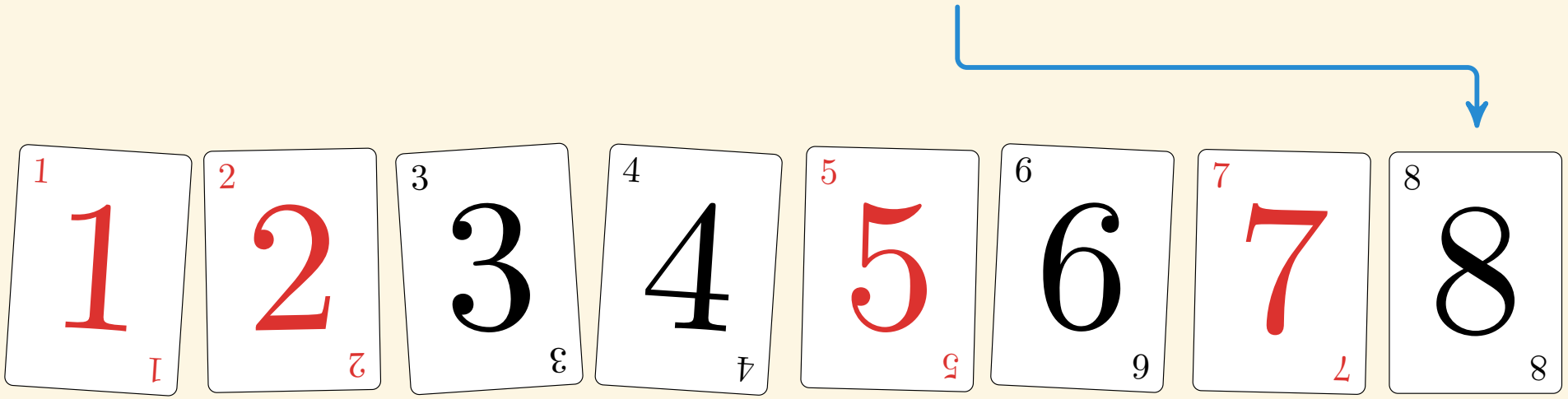
mlh@ntnu.no



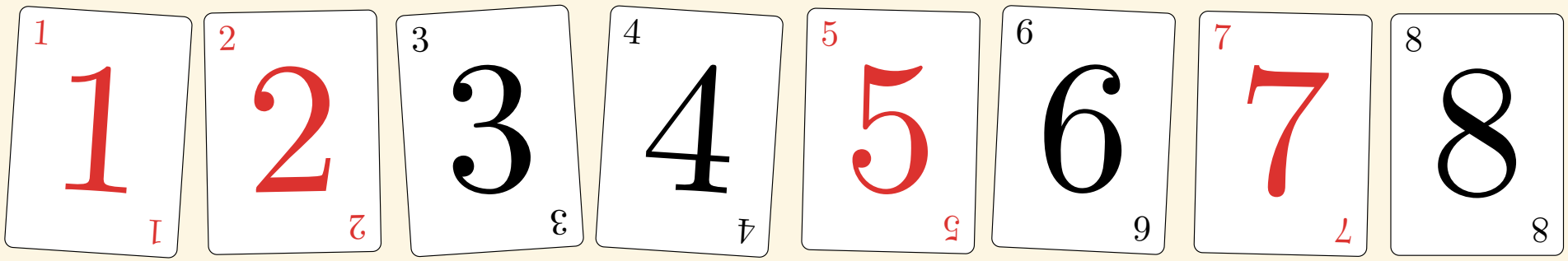
mlh@ntnu.no



mlh@ntnu.no



mlh@ntnu.no



mlh@ntnu.no

Bokas versjon er litt omstendelig ...

MERGE(A, p, q, r)

A tabell
p start
q splitt
r slutt

Dette er (en forenkling av) versjonen brukt i tidligere utgaver av boka.

Enklere versjon, med vaktposter (*sentinels*)

mlh@ntnu.no

MERGE(A, p, q, r)
1 copy into L and R

A tabell
p start
q splitt
r slutt

Venstre og høyre halvdel, begge med ∞ som «vaktpost» på slutten

mlh@ntnu.no

MERGE(A, p, q, r)

1 copy into L and R
2 **for** $k = p$ **to** r

A tabell
p start
q splitt
r slutt

For hver verdi som skal settes inn . . .

mlh@ntnu.no

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

A tabell
p start
q splitt
r slutt
i L-indeks
j R-indeks

Kan vi bruke første verdi til venstre? ($i, j = 1$, linje 1)

mlh@ntnu.no

```
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]
```

A	tabell
p	start
q	splitt
r	slutt
i	L-indeks
j	R-indeks

Sett den inn

mlh@ntnu.no

```
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
```

A	tabell
p	start
q	splitt
r	slutt
i	L-indeks
j	R-indeks

Neste i venstre «kø» rykker frem

mlh@ntnu.no

```
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]
```

A tabell
p start
q splitt
r slutt
i L-indeks
j R-indeks

Ellers bruk første til høyre

mlh@ntnu.no

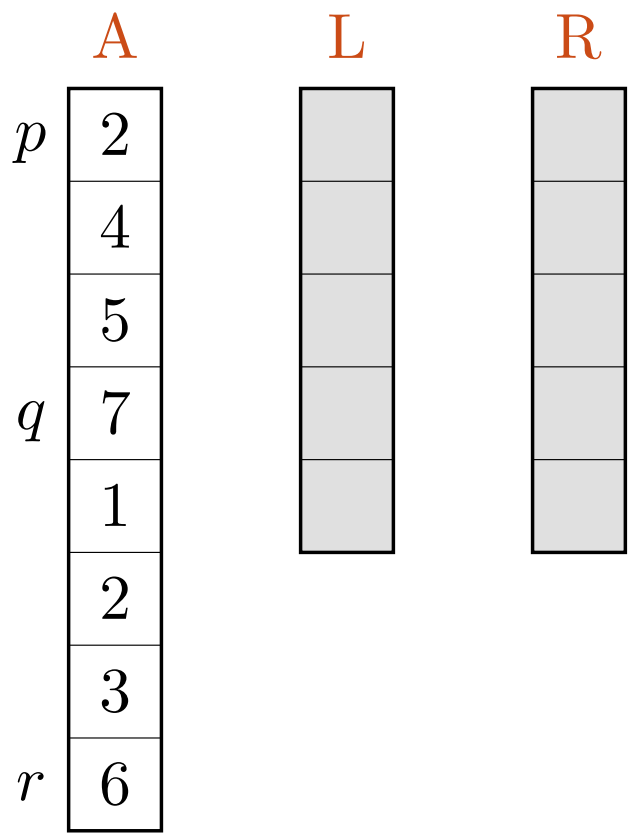
```
MERGE(A, p, q, r)
1  copy into L and R
2  for k = p to r
3        if L[i] ⩽ R[j]
4              A[k] = L[i]
5              i = i + 1
6        else A[k] = R[j]
7              j = j + 1
```

A	tabell
p	start
q	splitt
r	slutt
i	L-indeks
j	R-indeks

Neste i høyre «kø» rykker frem

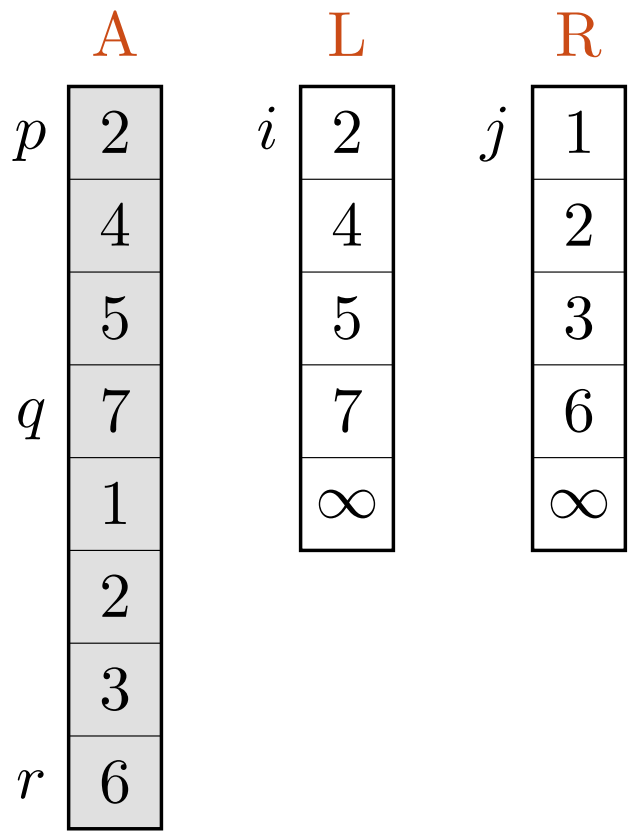
mlh@ntnu.no

```
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
```



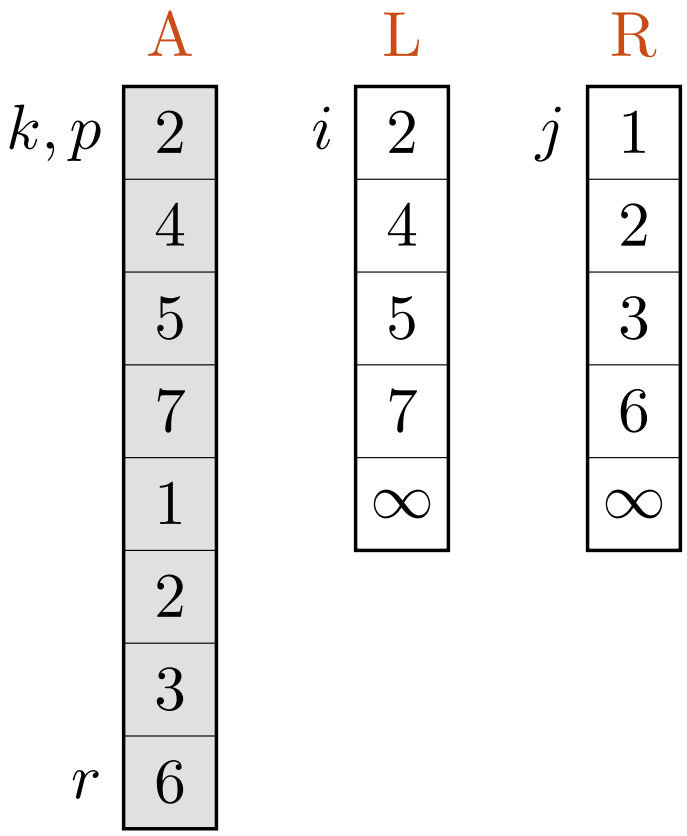
mlh@ntnu.no

```
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
```



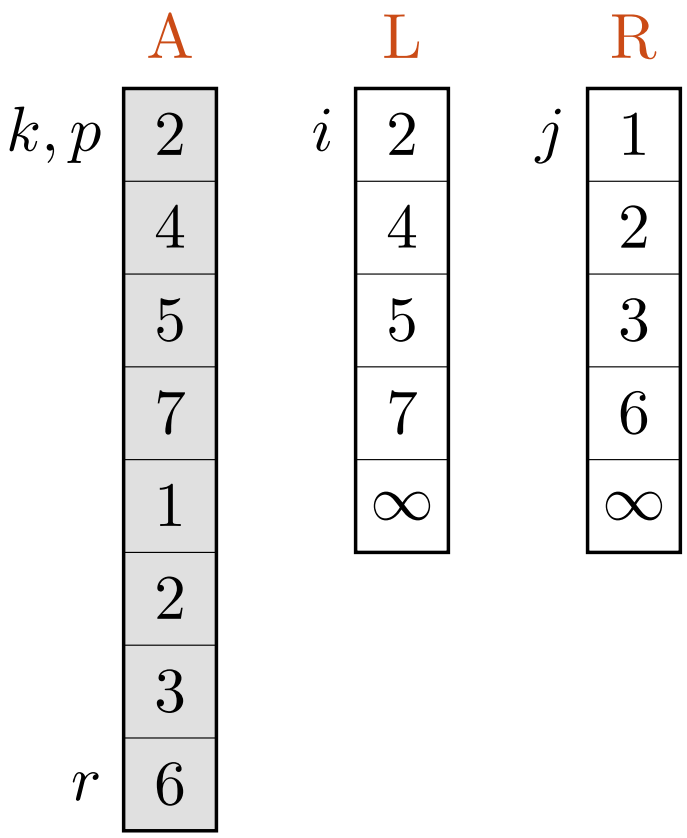
mlh@ntnu.no

```
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
```



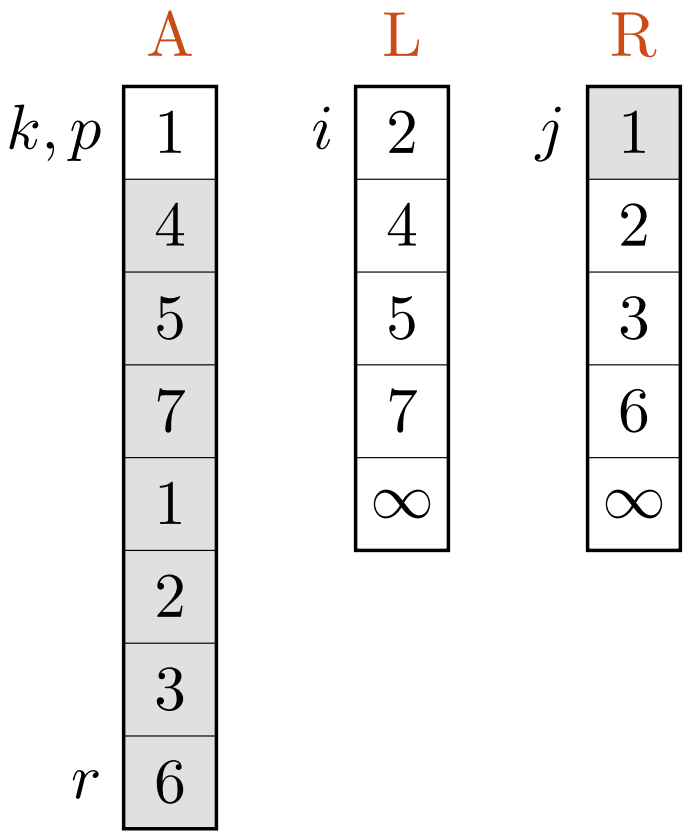
mlh@ntnu.no

```
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
```



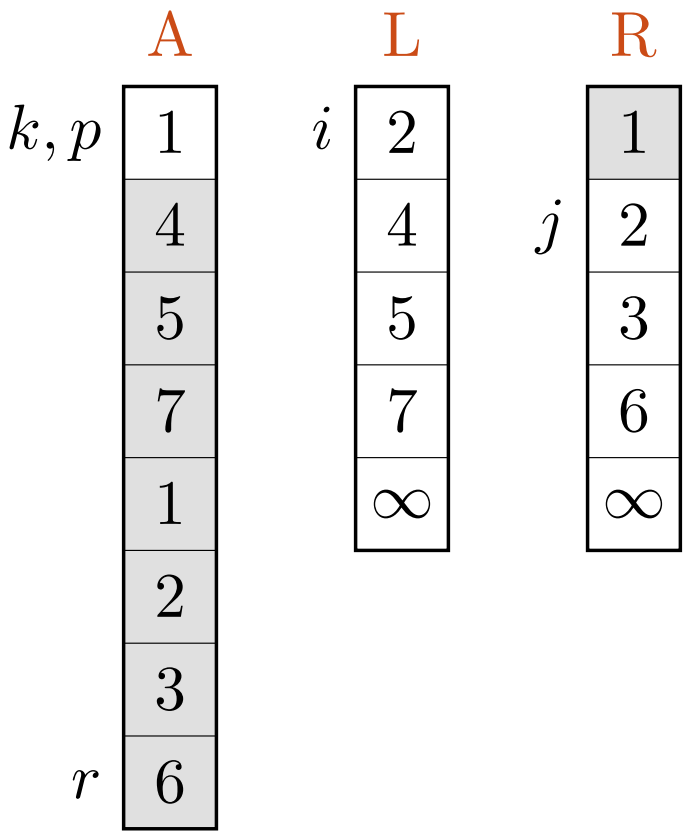
mlh@ntnu.no

```
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
```



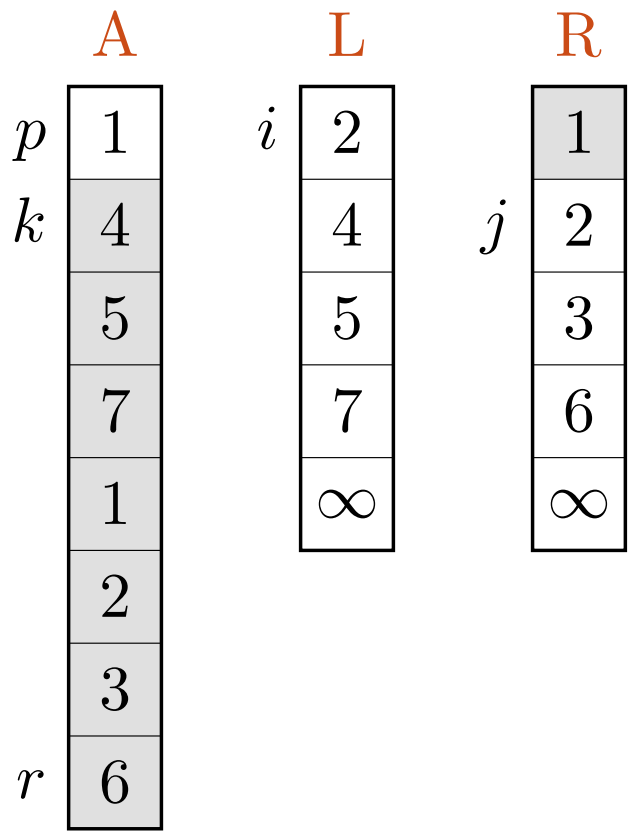
mlh@ntnu.no

```
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
```



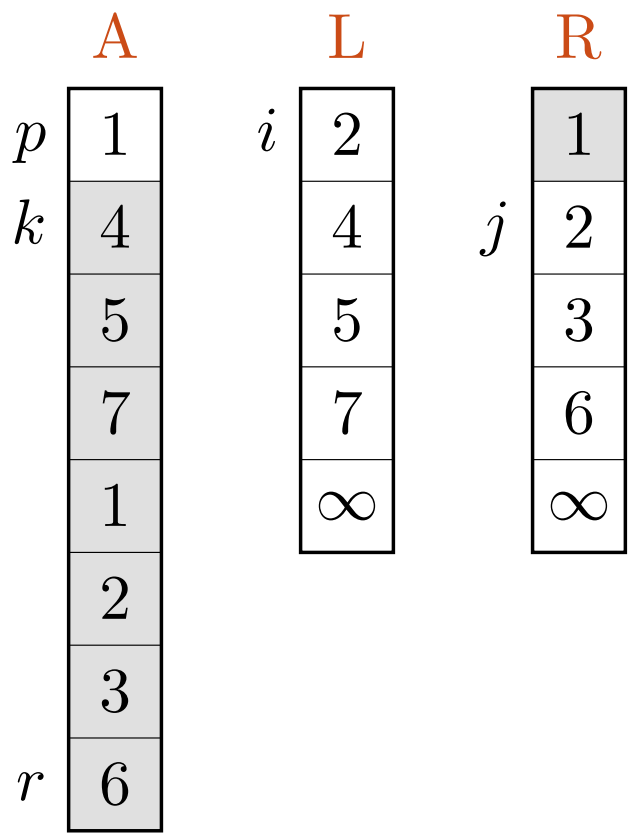
mlh@ntnu.no

```
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
```



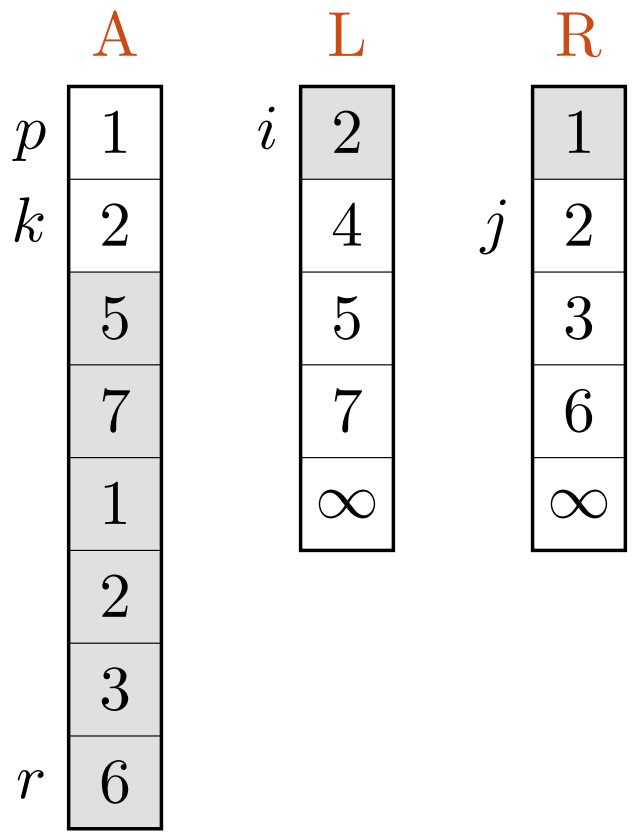
mlh@ntnu.no

```
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
```



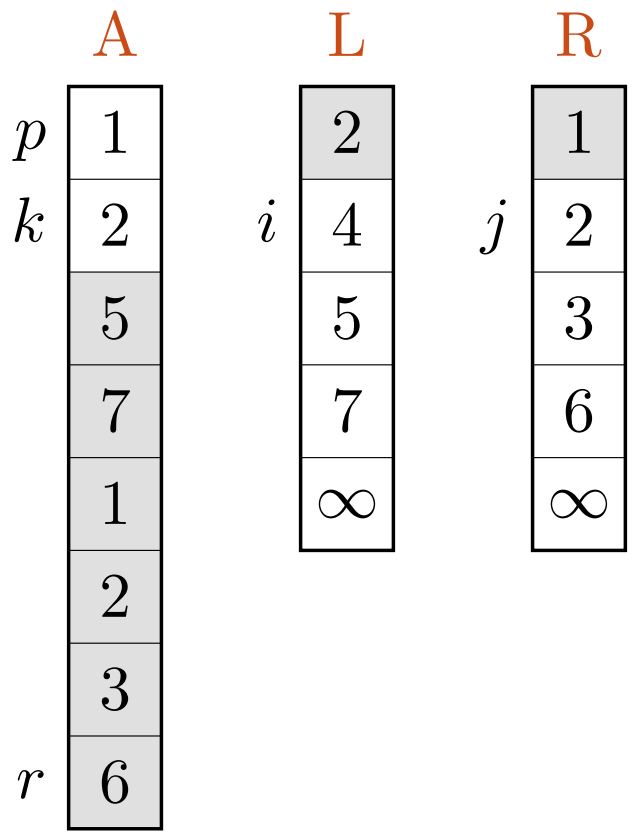
mlh@ntnu.no

```
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
```



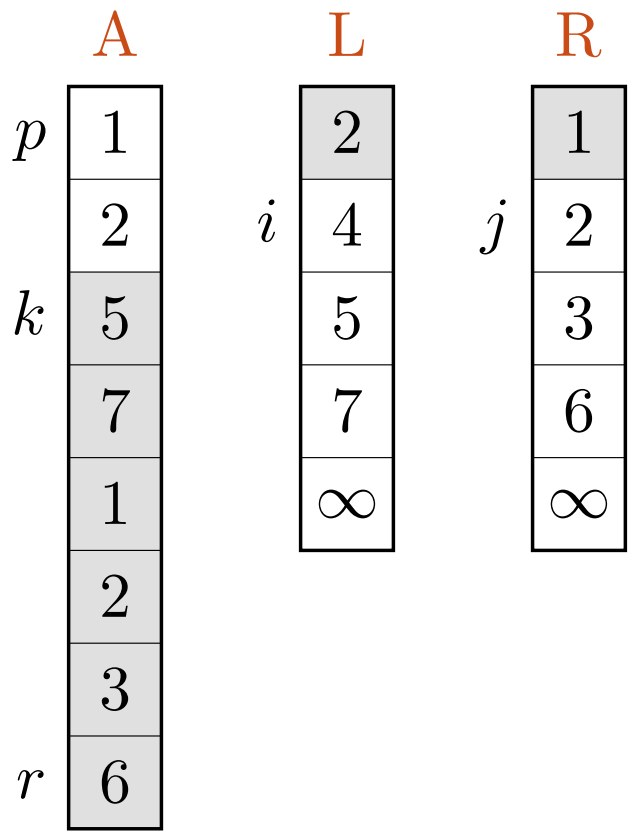
mlh@ntnu.no

```
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
```



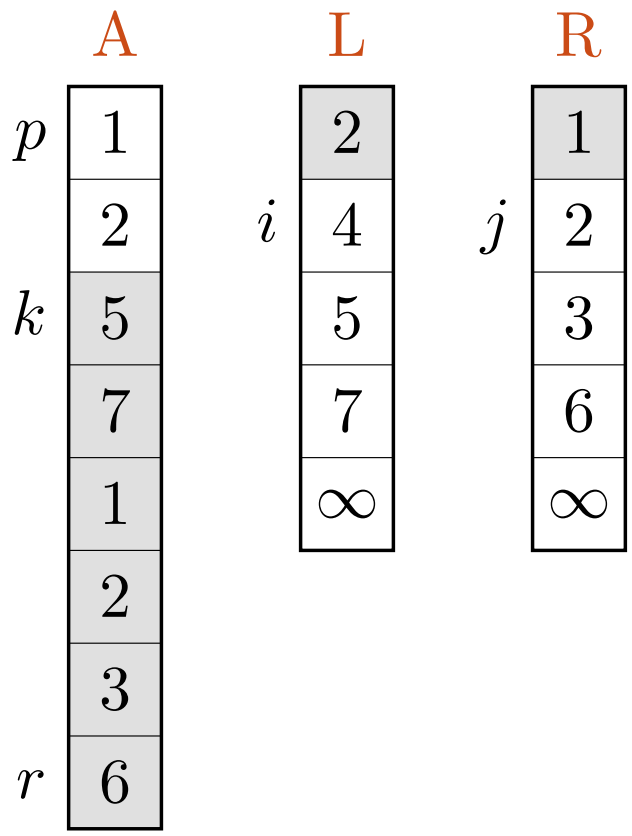
mlh@ntnu.no

```
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
```



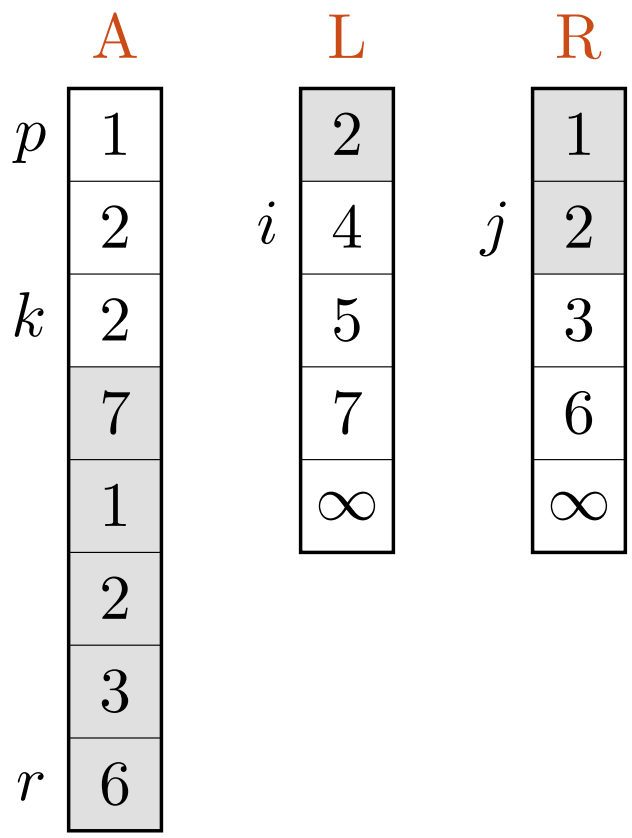
mlh@ntnu.no

```
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
```



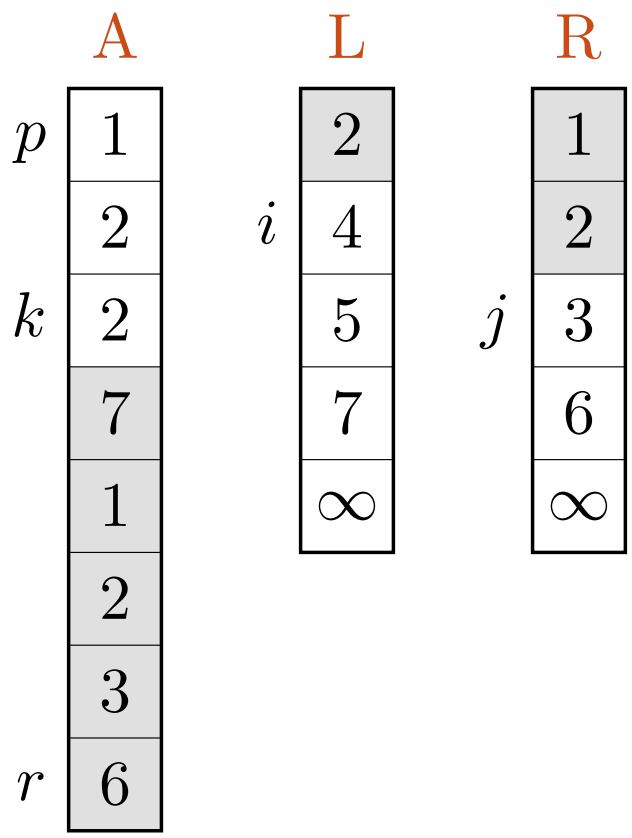
mlh@ntnu.no

```
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
```



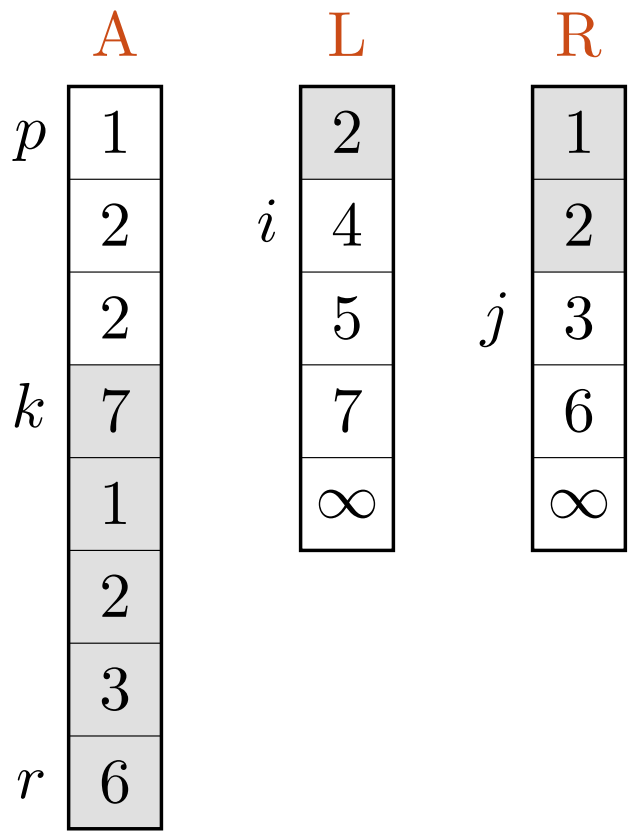
mlh@ntnu.no

```
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
```



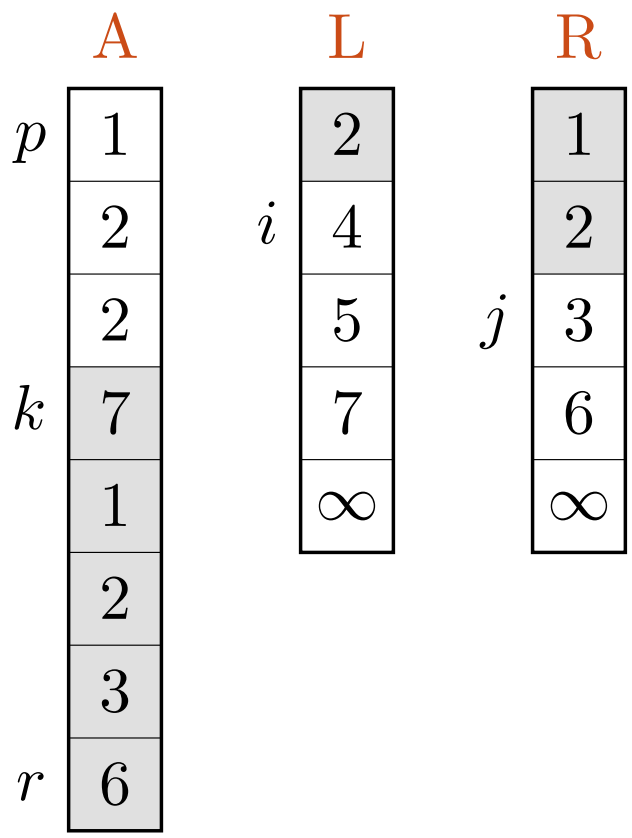
mlh@ntnu.no

```
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
```



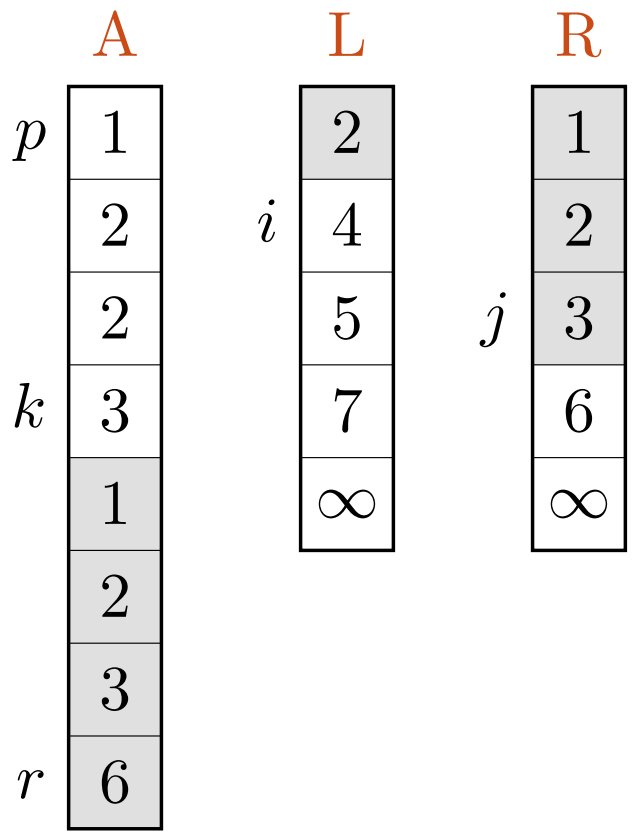
mlh@ntnu.no

```
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
```



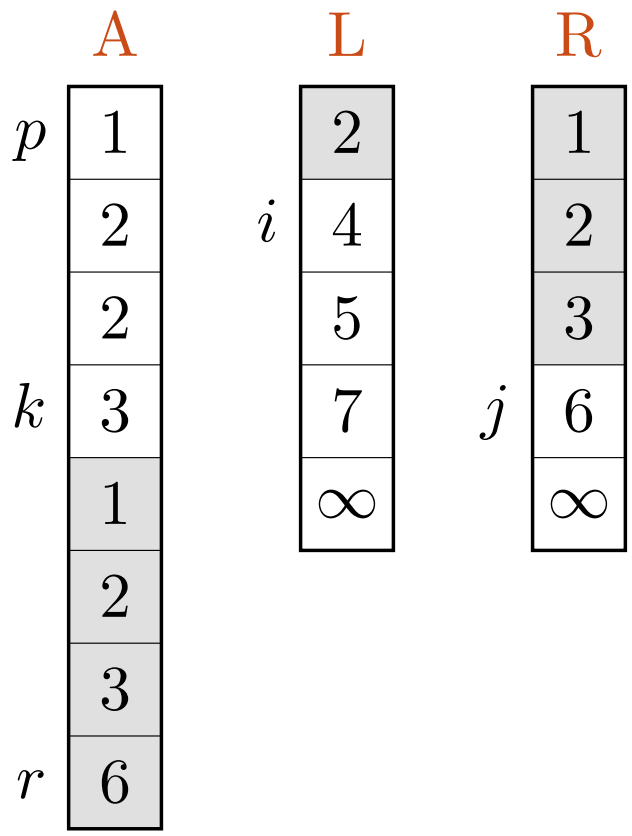
mlh@ntnu.no

```
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
```



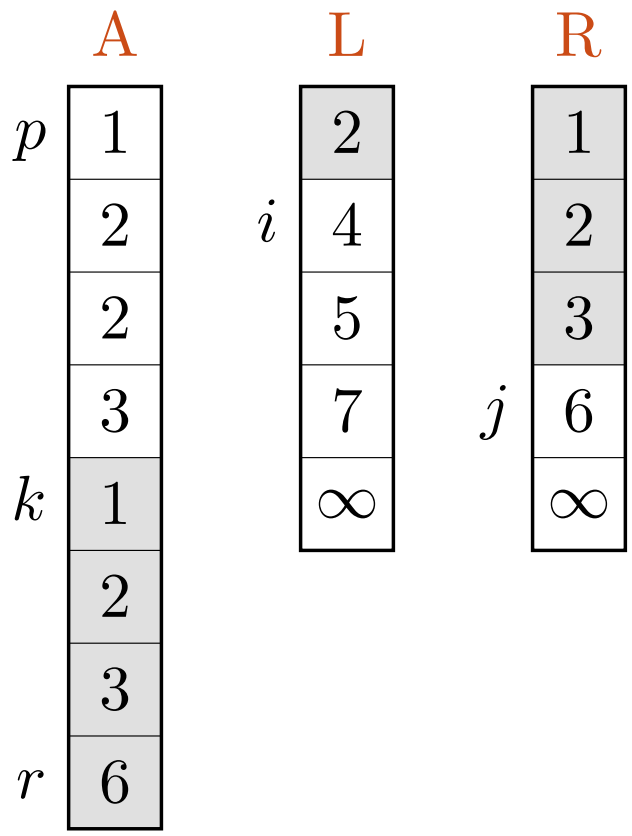
mlh@ntnu.no

```
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
```



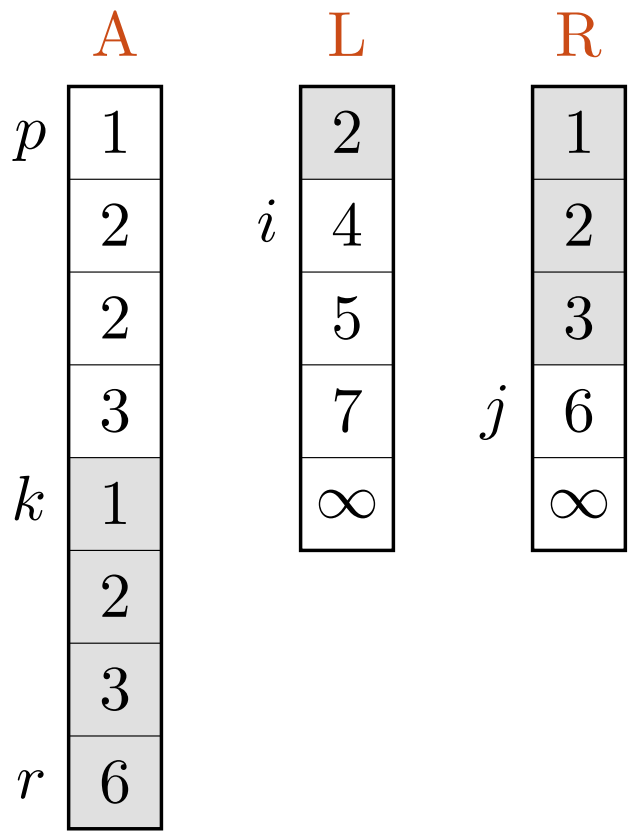
mlh@ntnu.no

```
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
```



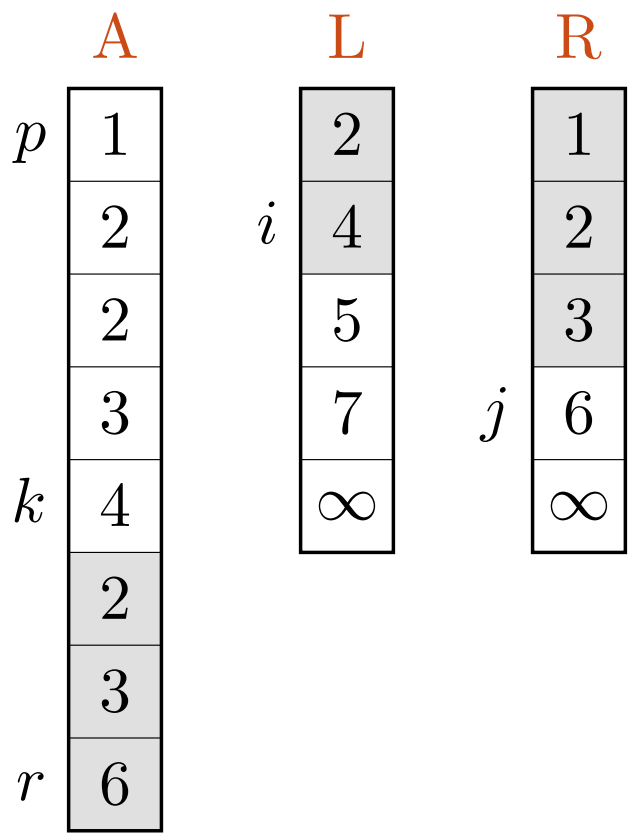
mlh@ntnu.no

```
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
```



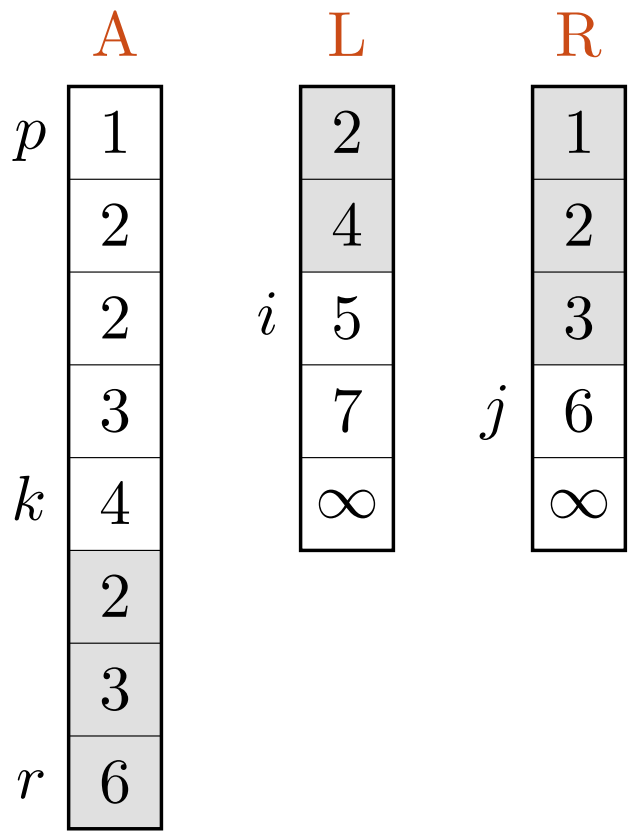
mlh@ntnu.no

```
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
```



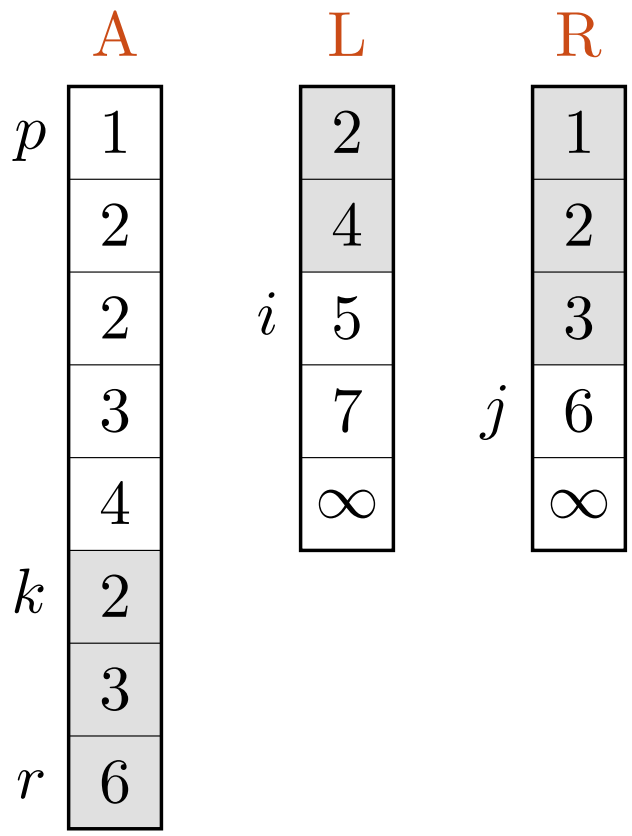
mlh@ntnu.no

```
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
```



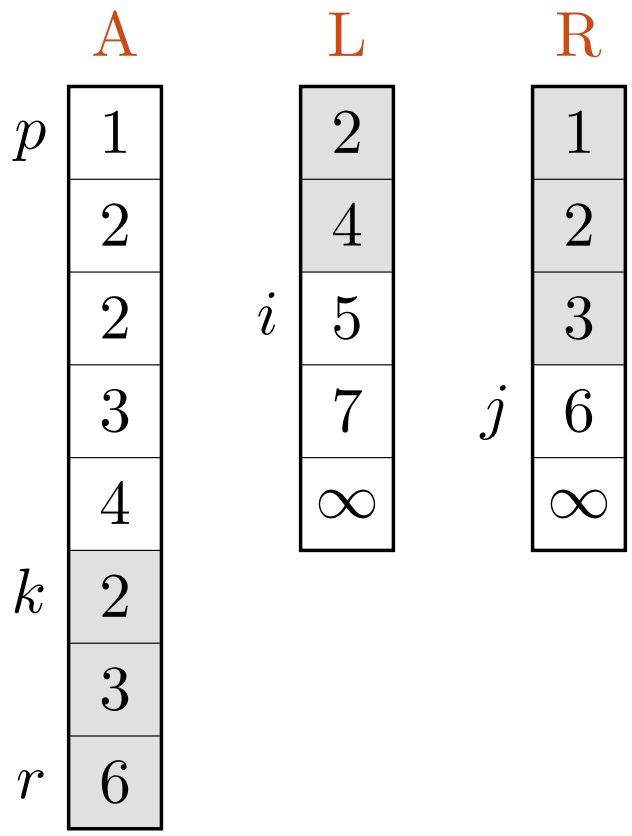
mlh@ntnu.no

```
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
```



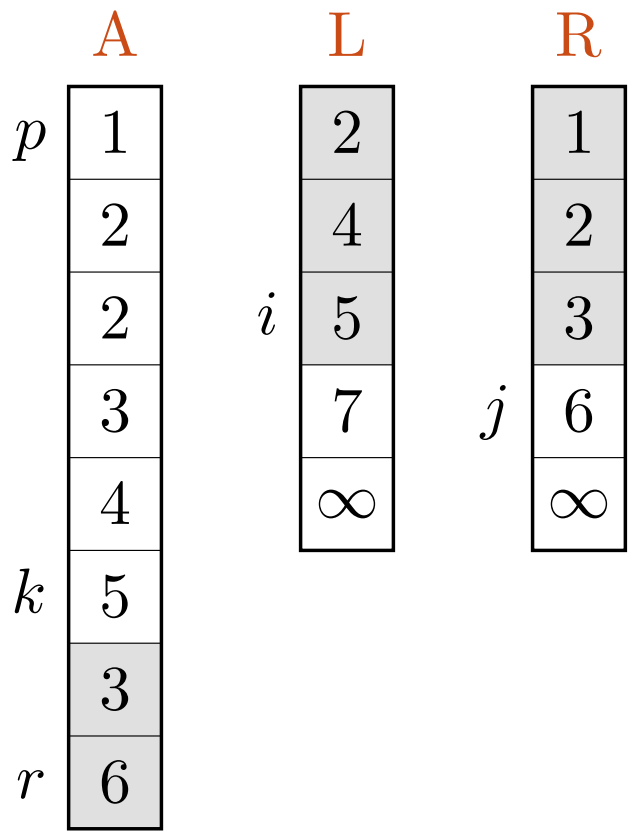
mlh@ntnu.no

```
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
```



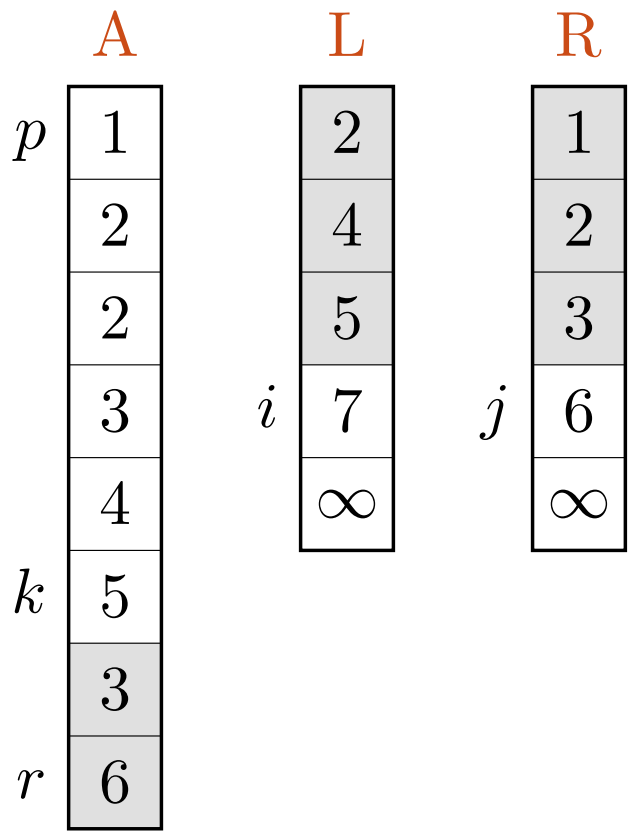
mlh@ntnu.no

```
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
```



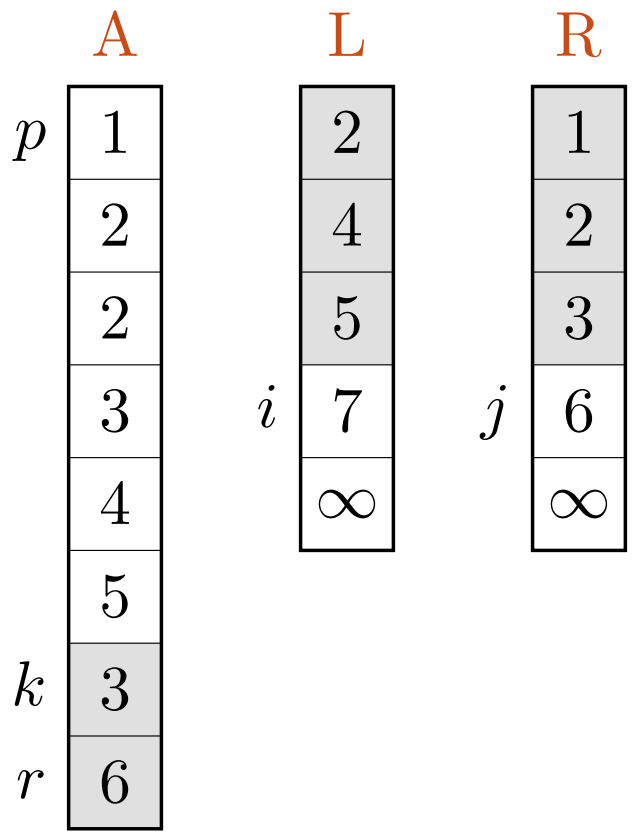
mlh@ntnu.no

```
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
```



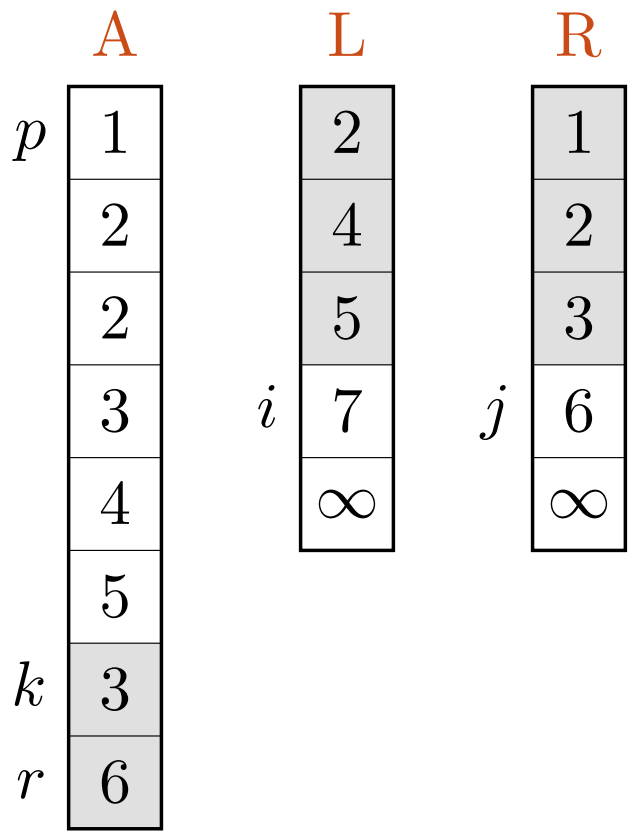
mlh@ntnu.no

```
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
```



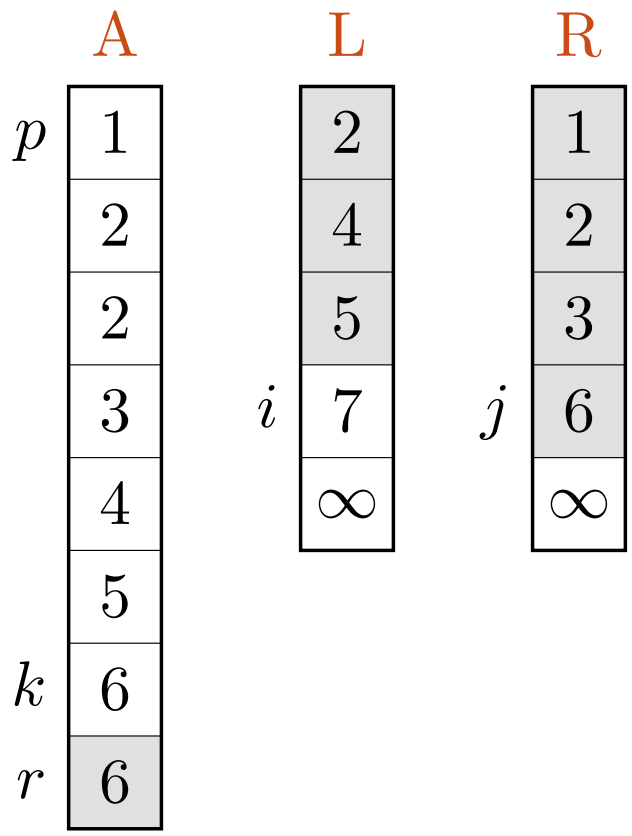
mlh@ntnu.no

```
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
```



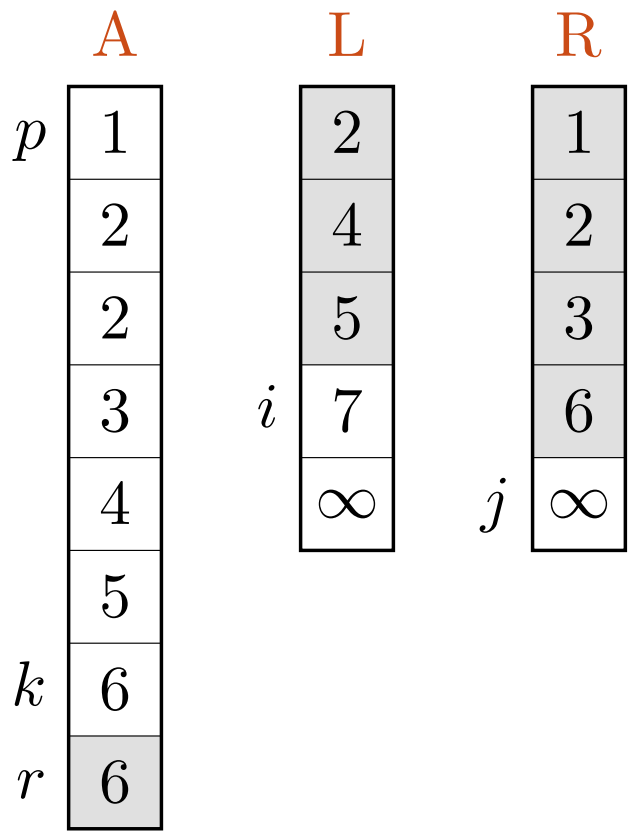
mlh@ntnu.no

```
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
```



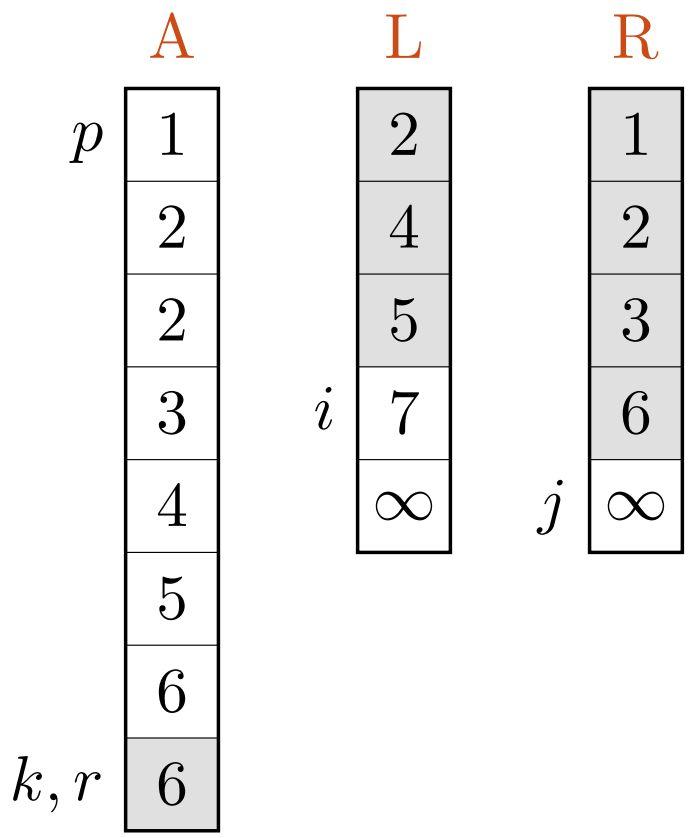
mlh@ntnu.no

```
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
```



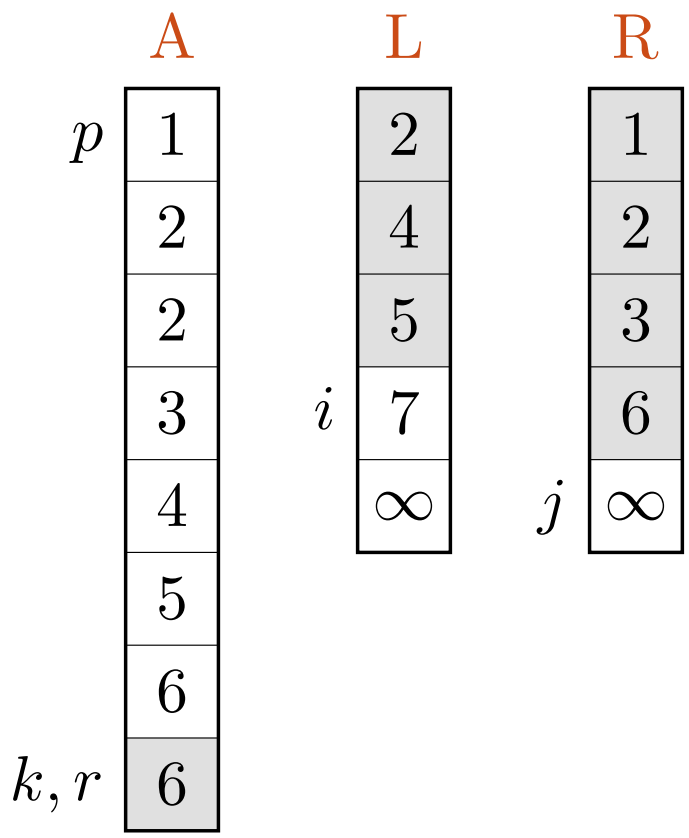
mlh@ntnu.no

```
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
```



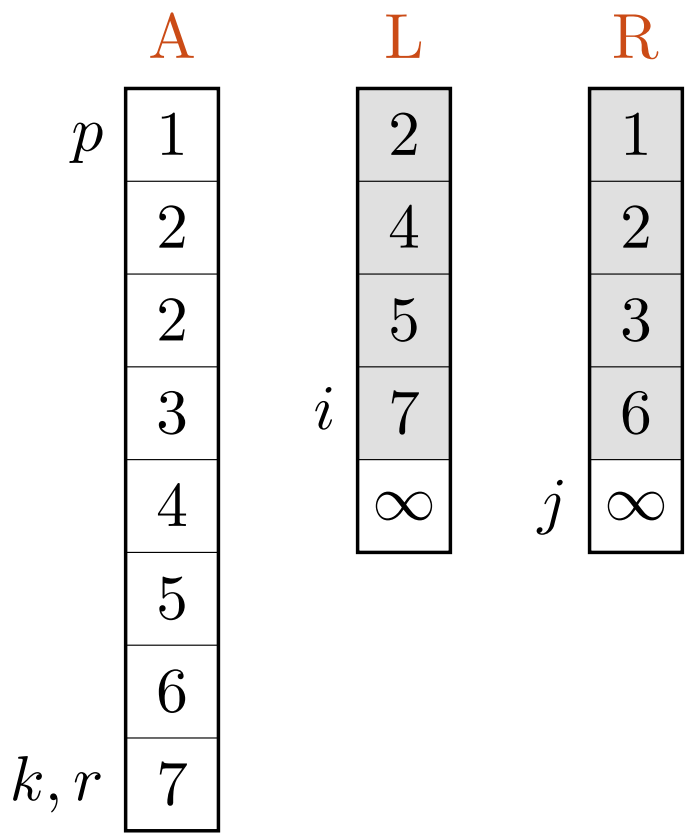
mlh@ntnu.no

```
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
```



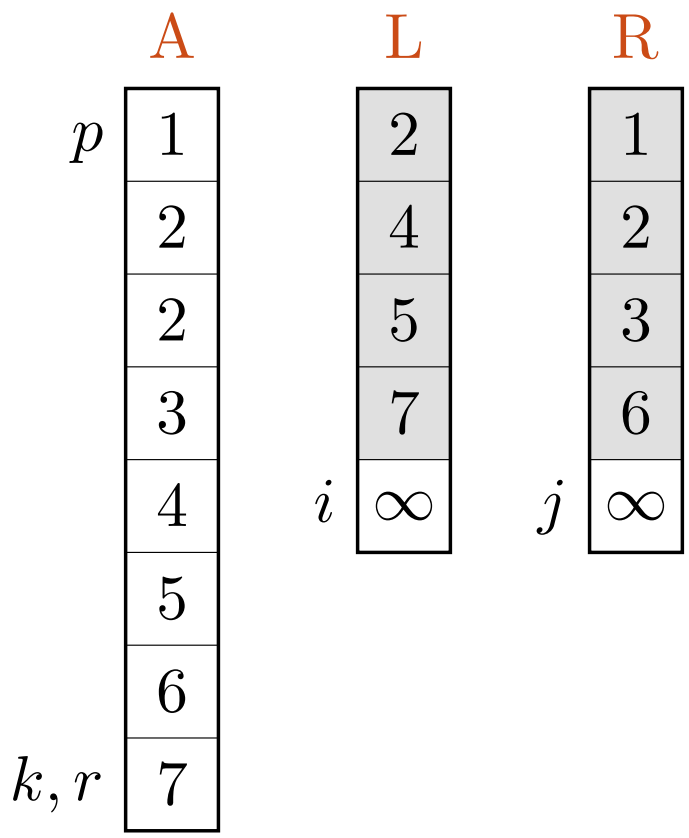
mlh@ntnu.no

```
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
```



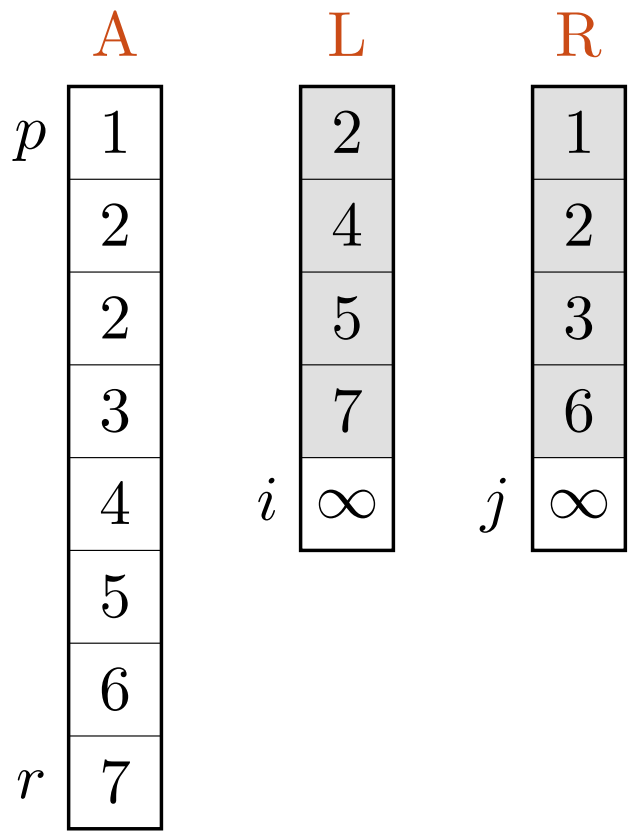
mlh@ntnu.no

```
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
```



mlh@ntnu.no

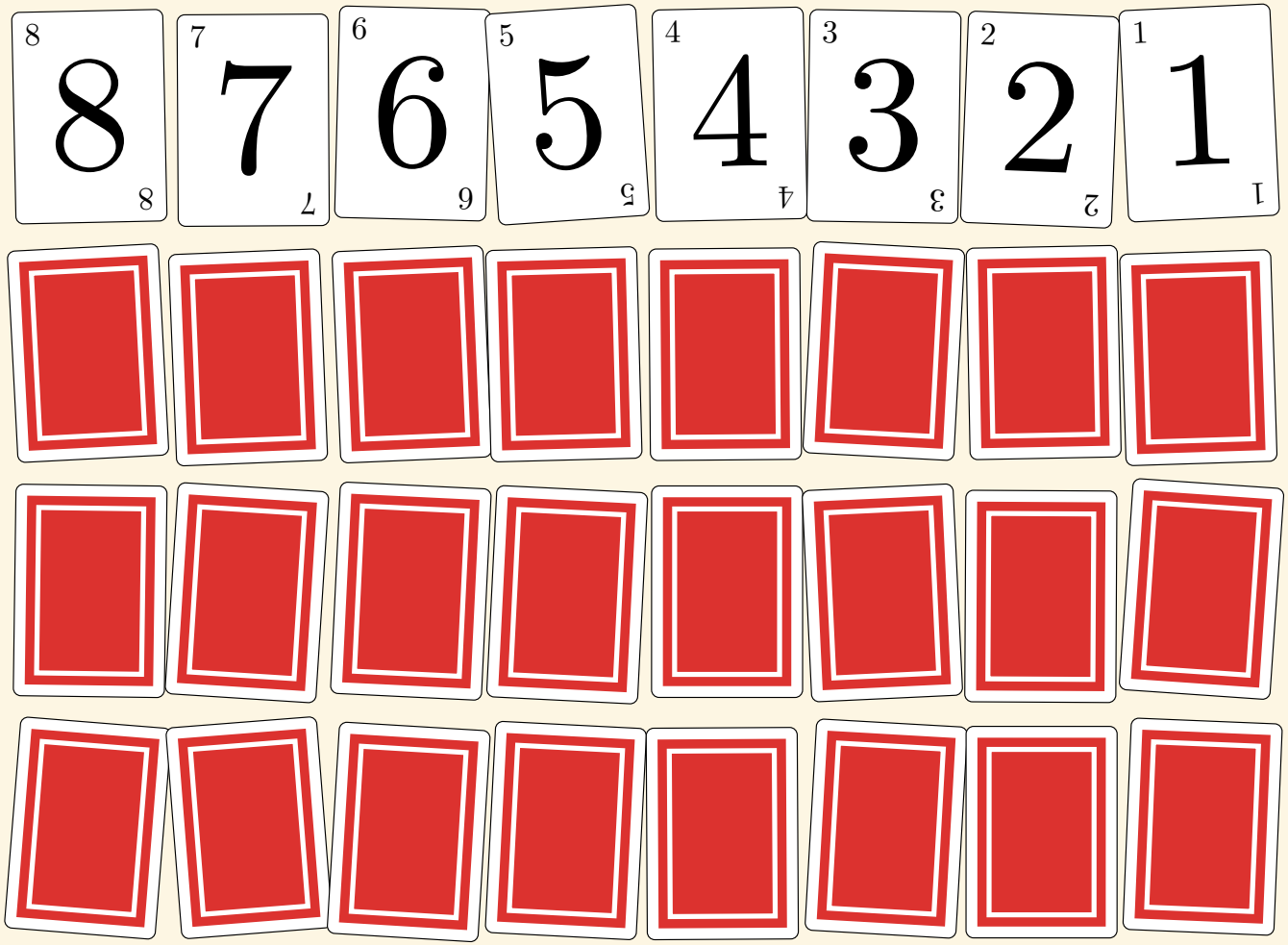
```
MERGE(A, p, q, r)
1  copy into L and R
2  for k = p to r
3        if L[i] ⩽ R[j]
4              A[k] = L[i]
5              i = i + 1
6        else A[k] = R[j]
7              j = j + 1
```



mlh@ntnu.no

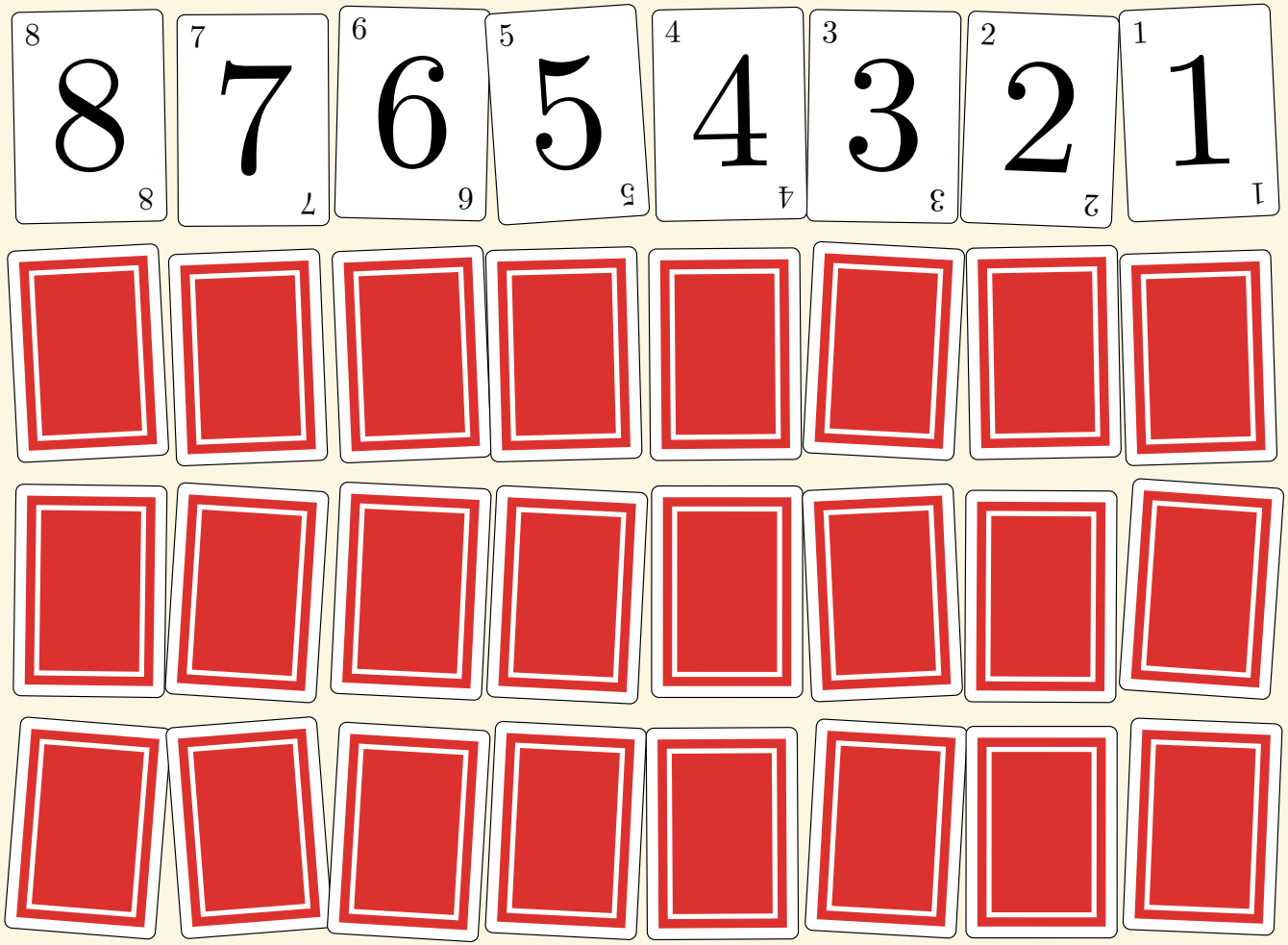
Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Del på midten	Flett venstre og høyre halvdel
Delinstanser	Delløsninger IH
Venstre og høyre halvdel	Venstre og høyre halvdel, sorterte
Grunntilfelle	Grunnløsning
Sekvens med maks ett element	Samme sekvens, allerede sortert

mlh@ntnu.no



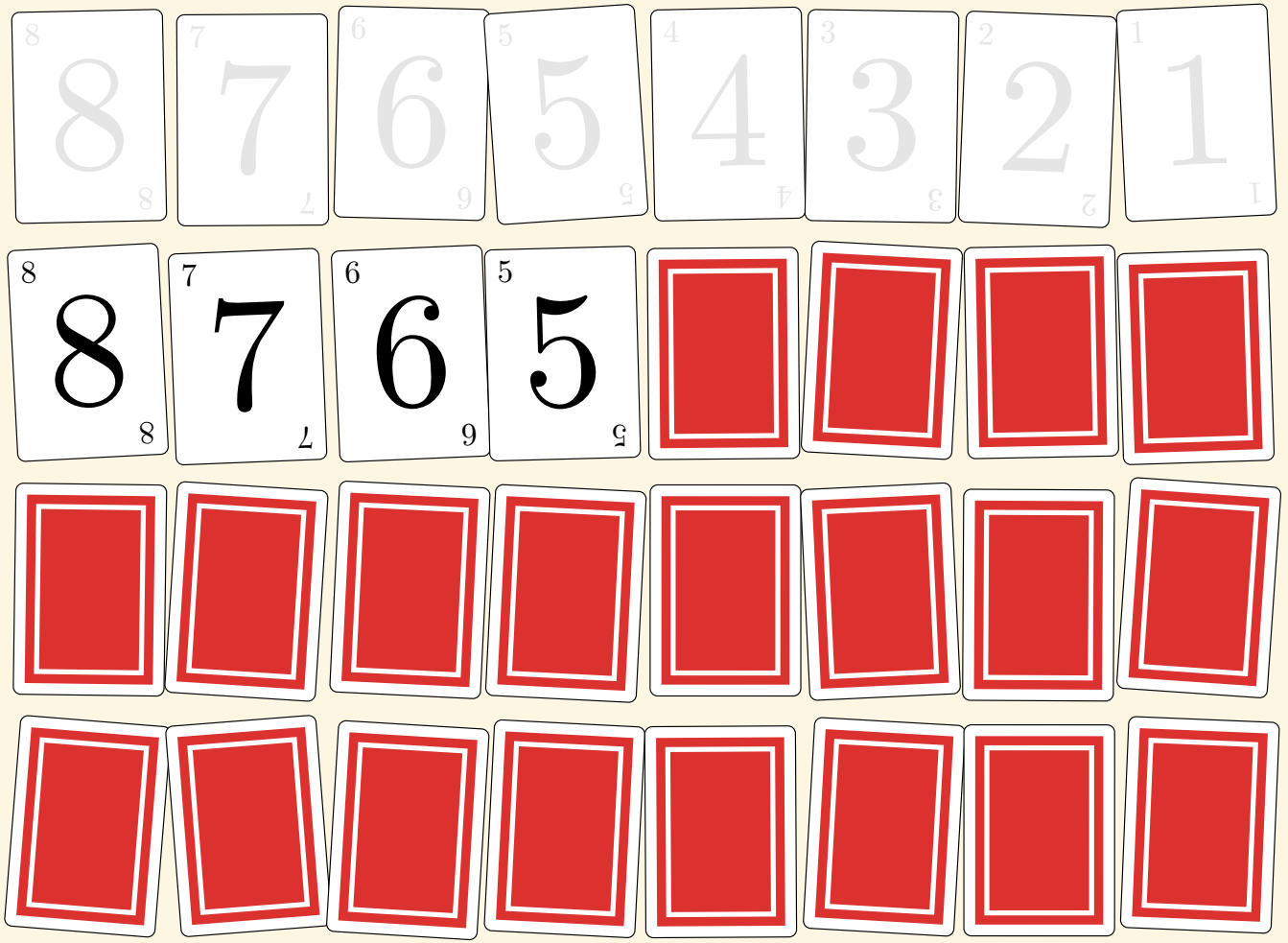
Skal sortere A[1 . . 8]

mlh@ntnu.no



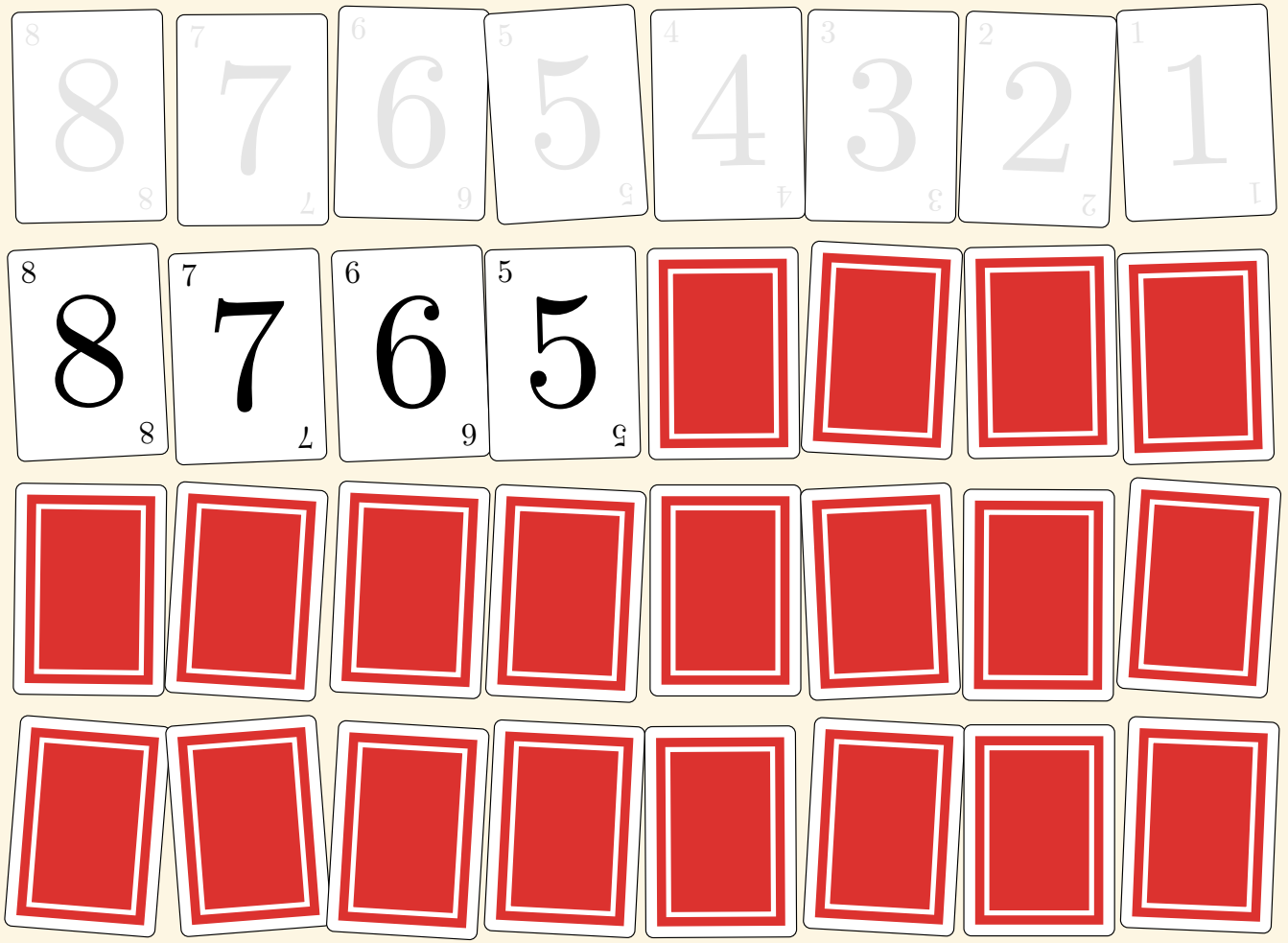
Sortér først A[1..4] og så A[5..8]

mlh@ntnu.no



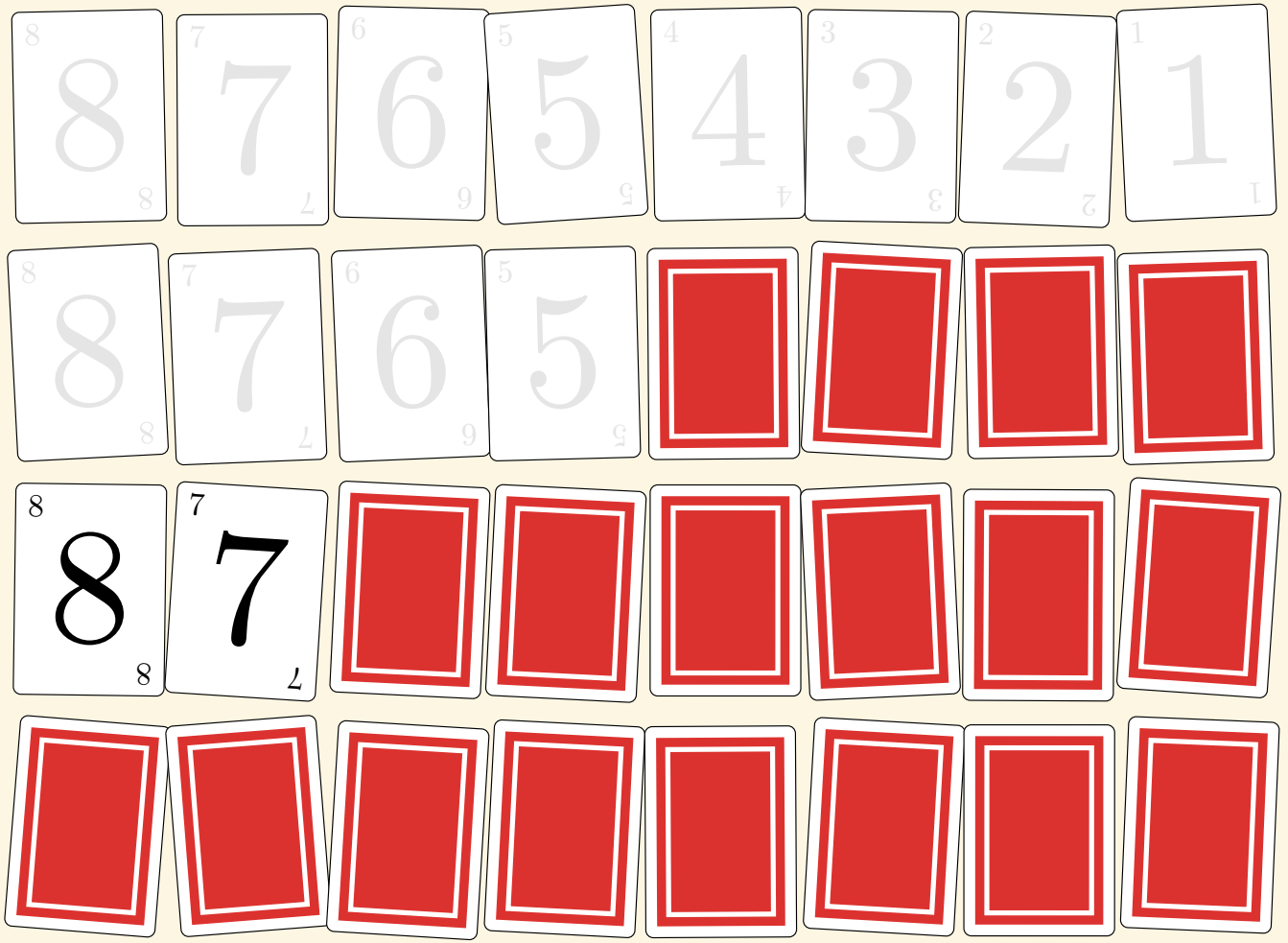
Skal sortere A[1 . . 4]

mlh@ntnu.no



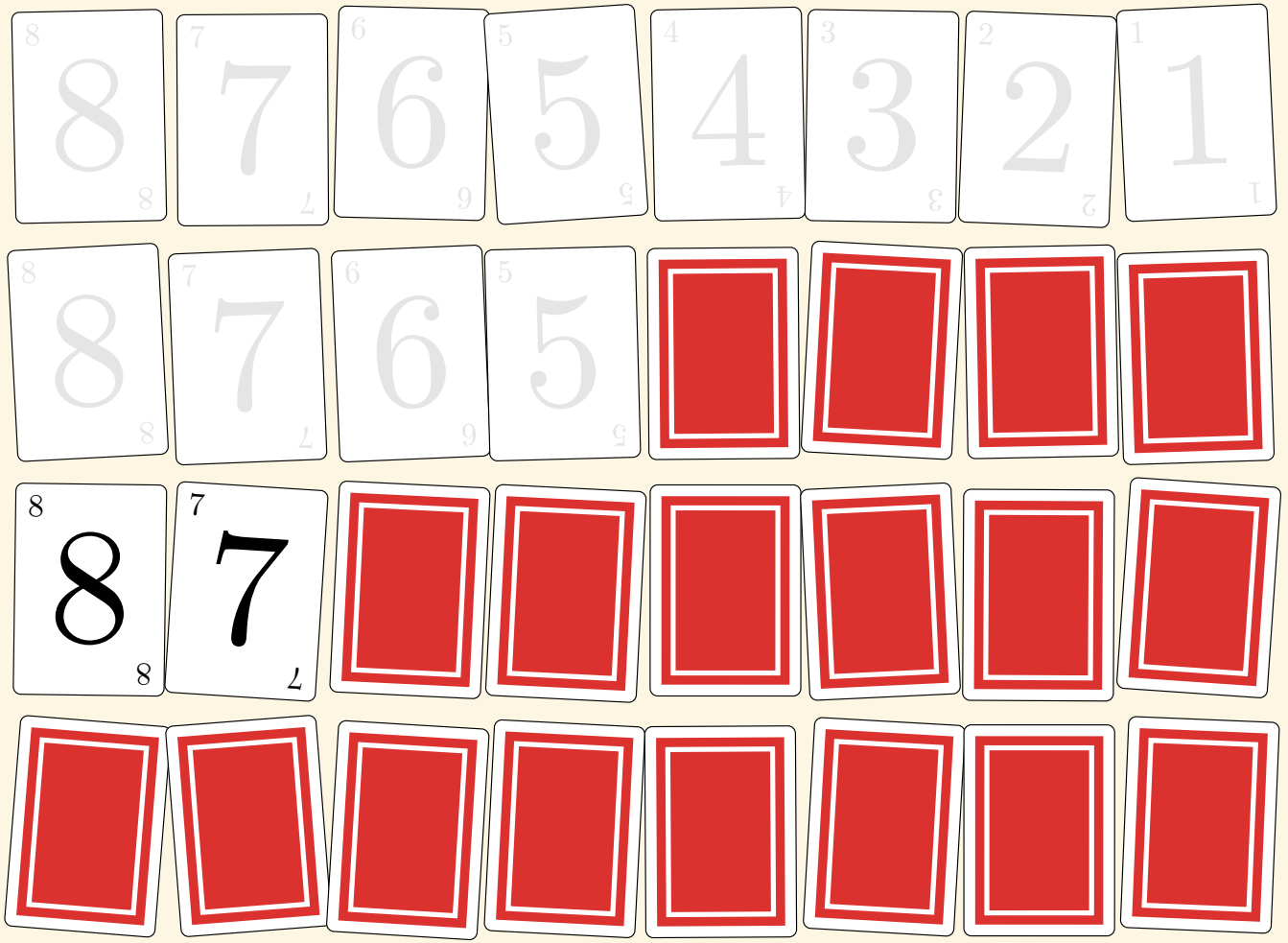
Sortér først A[1 . . 2] og så A[3 . . 4]

mlh@ntnu.no



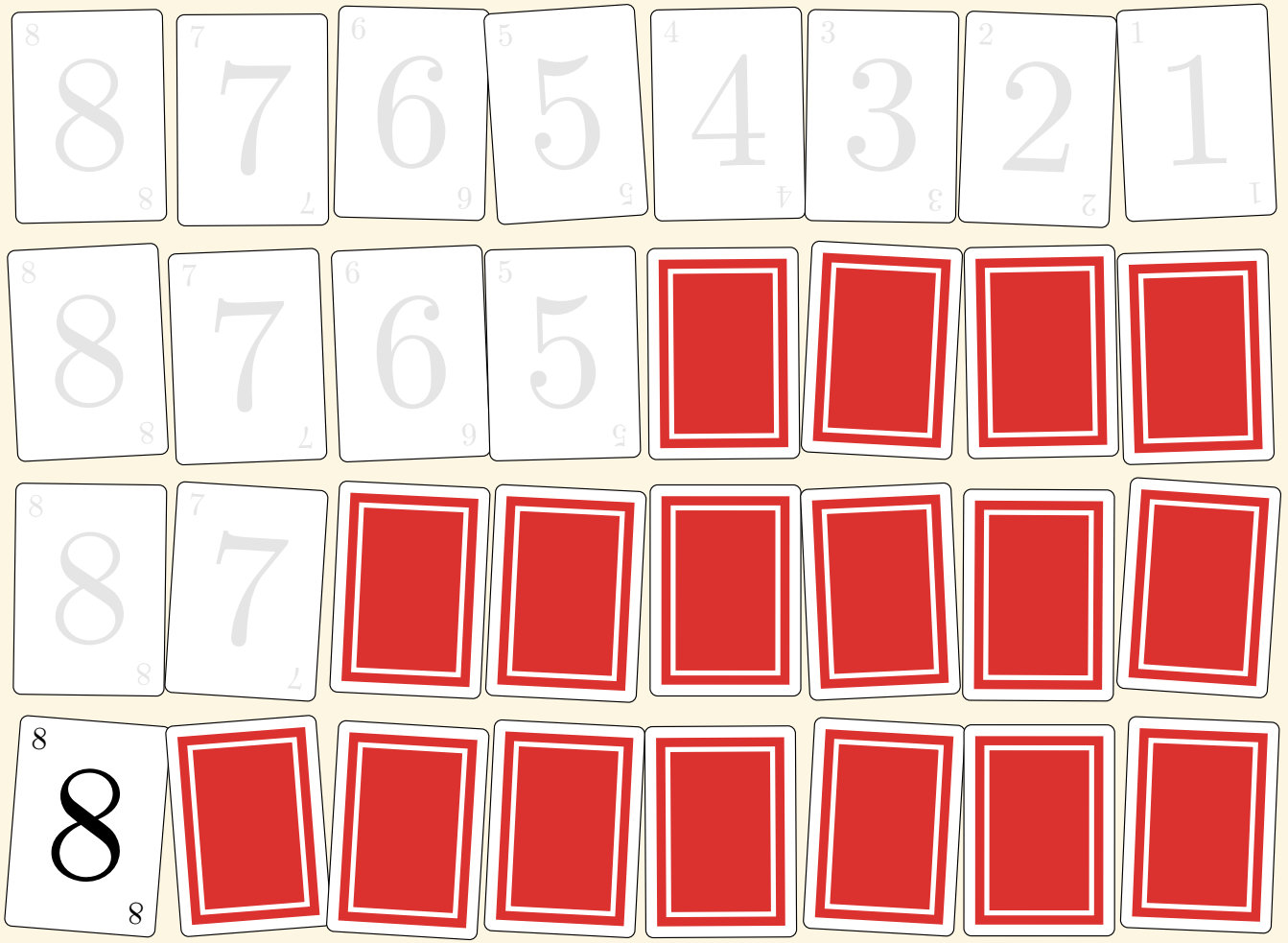
Skal sortere A[1 . . 2]

mlh@ntnu.no



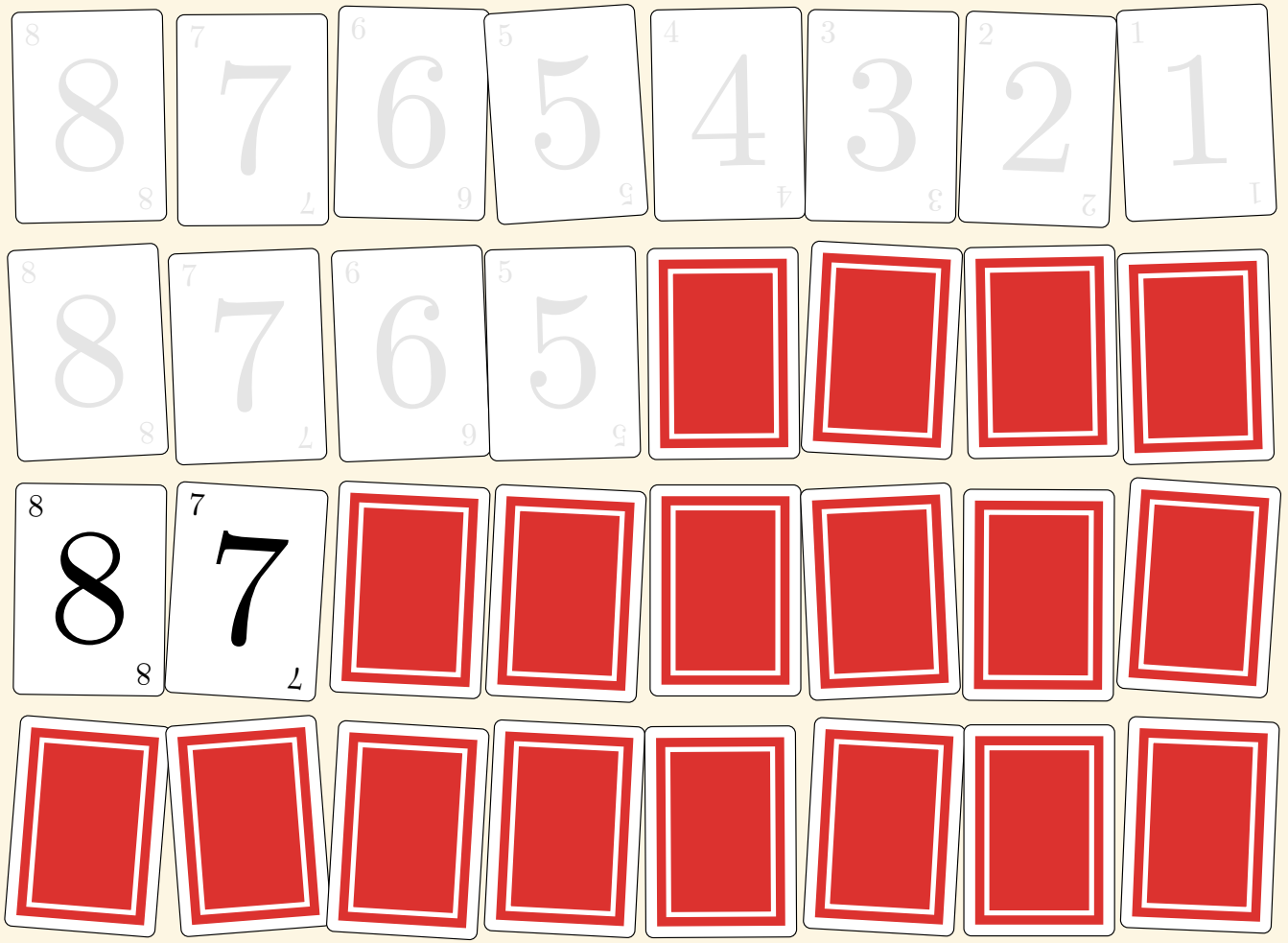
Sortér først A[1] og så A[2]

mlh@ntnu.no



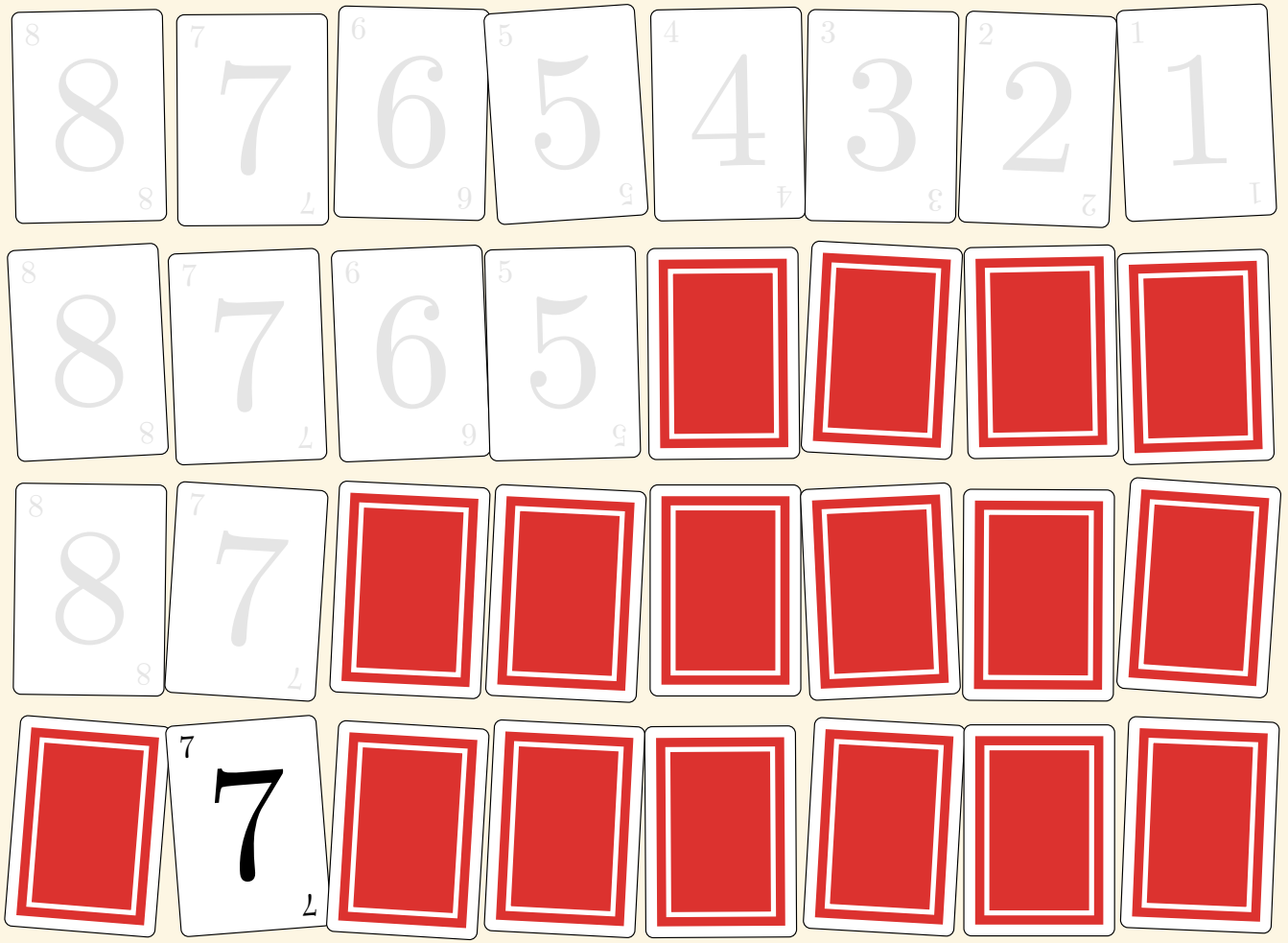
A[1] er sortert

mlh@ntnu.no



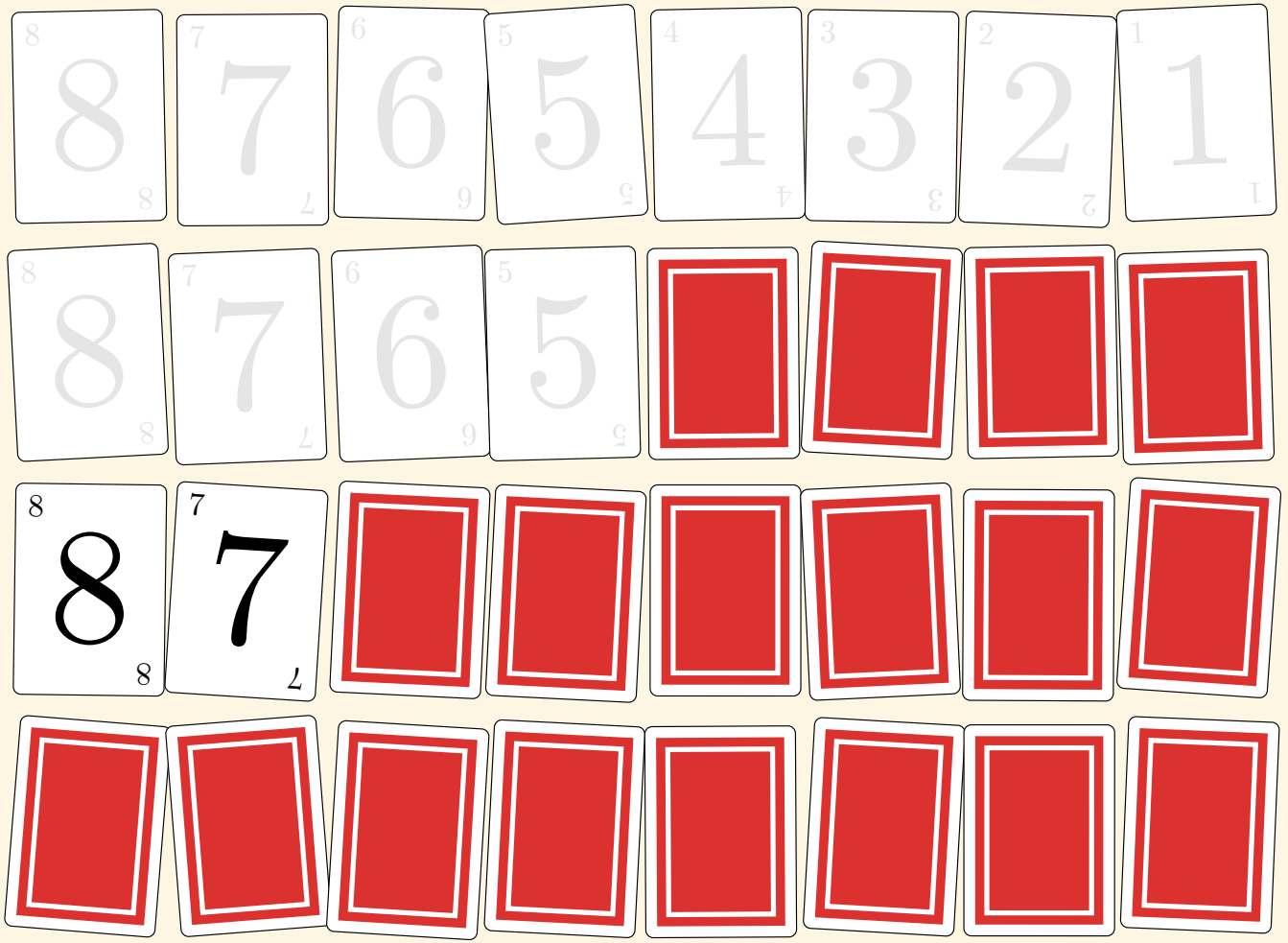
Sortér først A[1] og så A[2]

mlh@ntnu.no



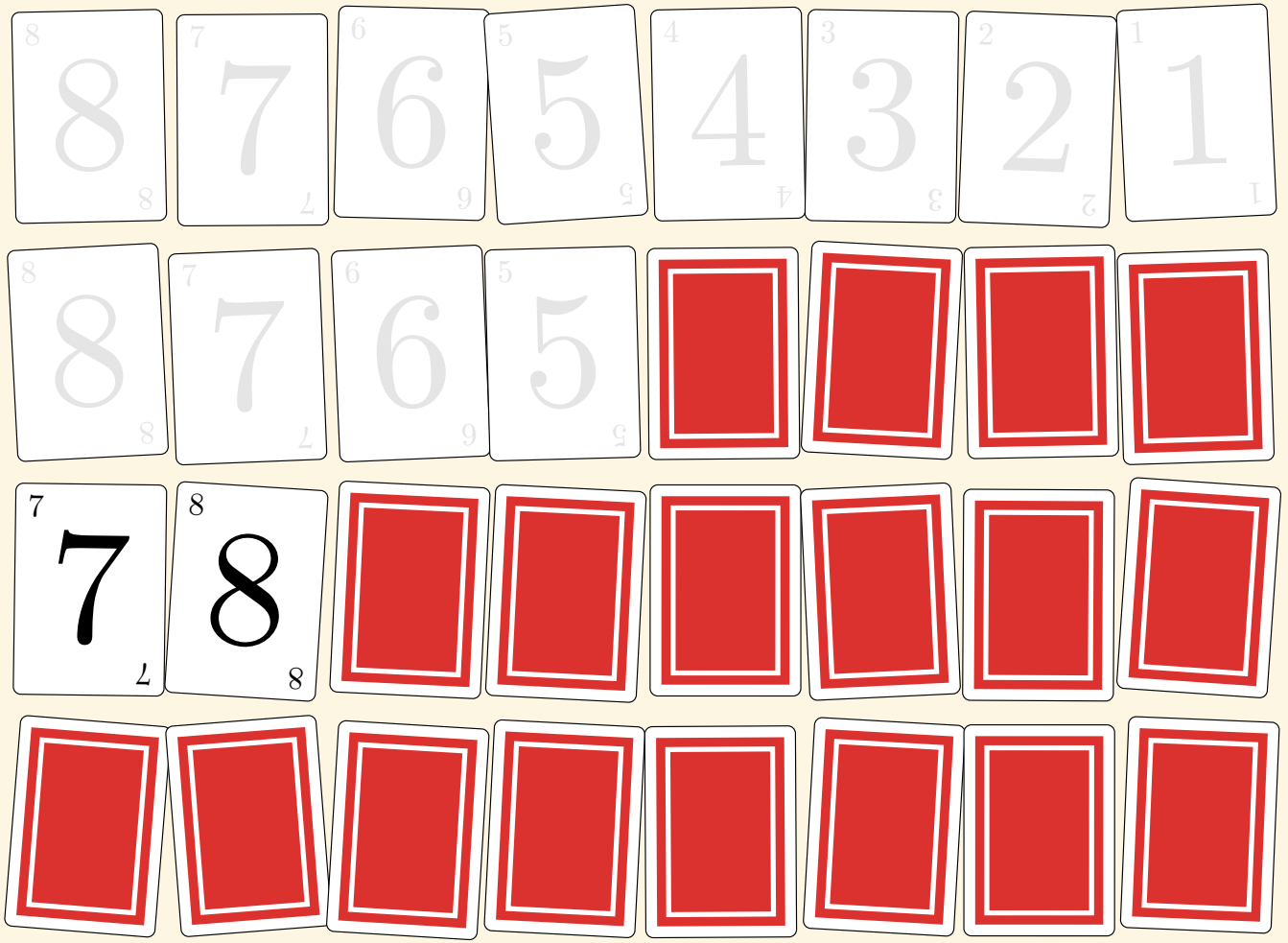
A[2] er sortert

mlh@ntnu.no



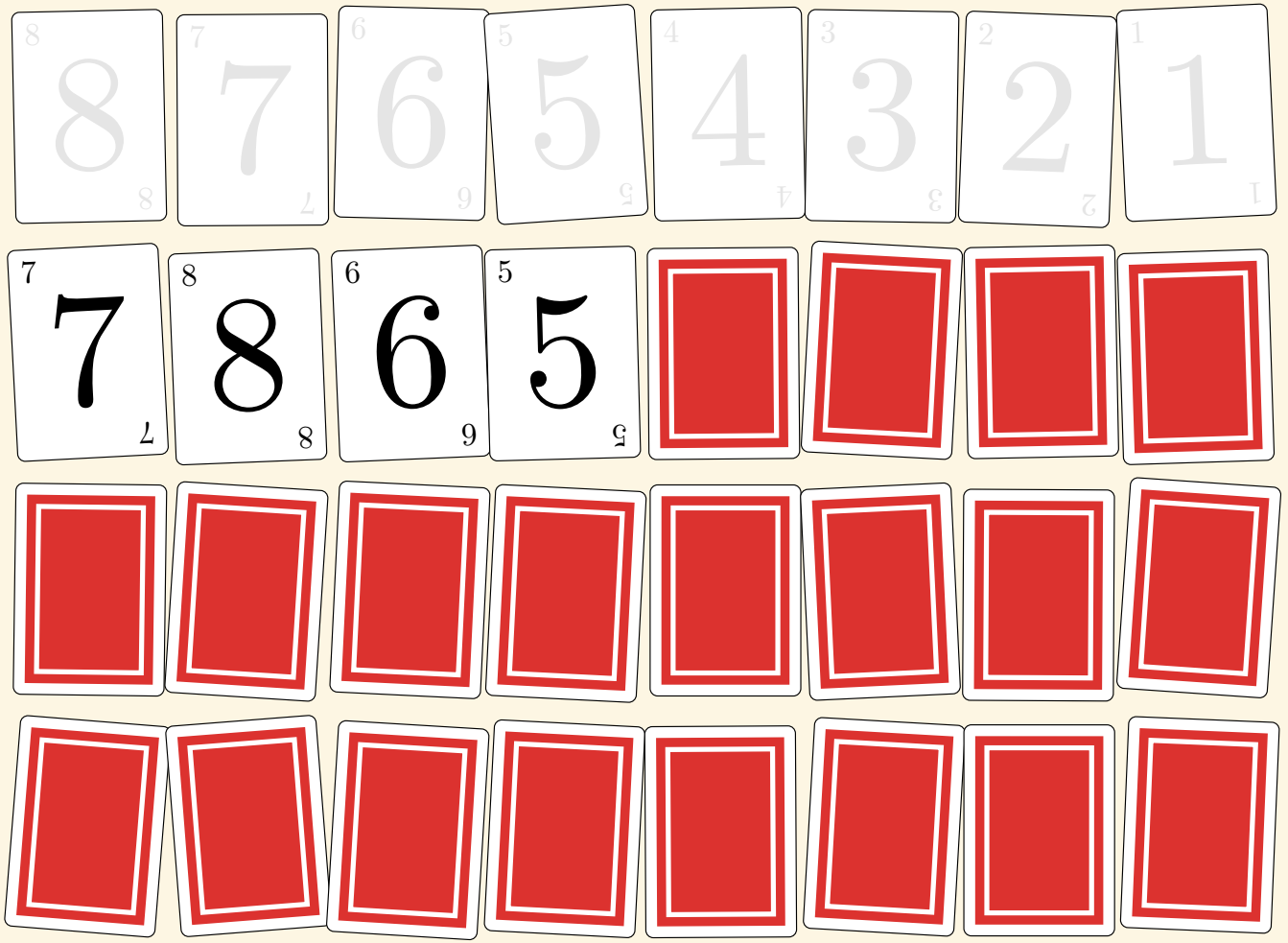
Flett sammen A[1] og A[2]

mlh@ntnu.no



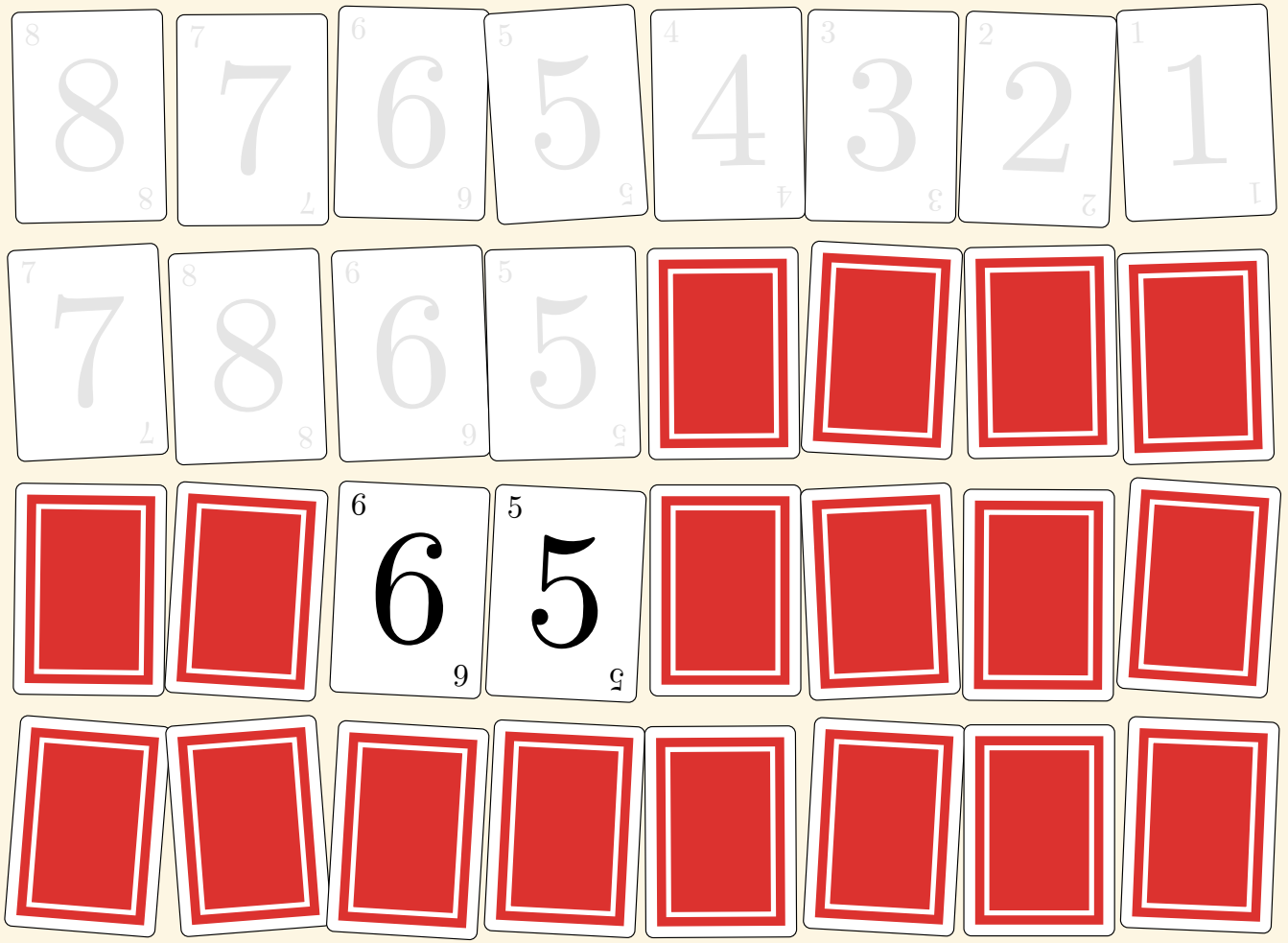
A[1 . . 2] er sortert

mlh@ntnu.no



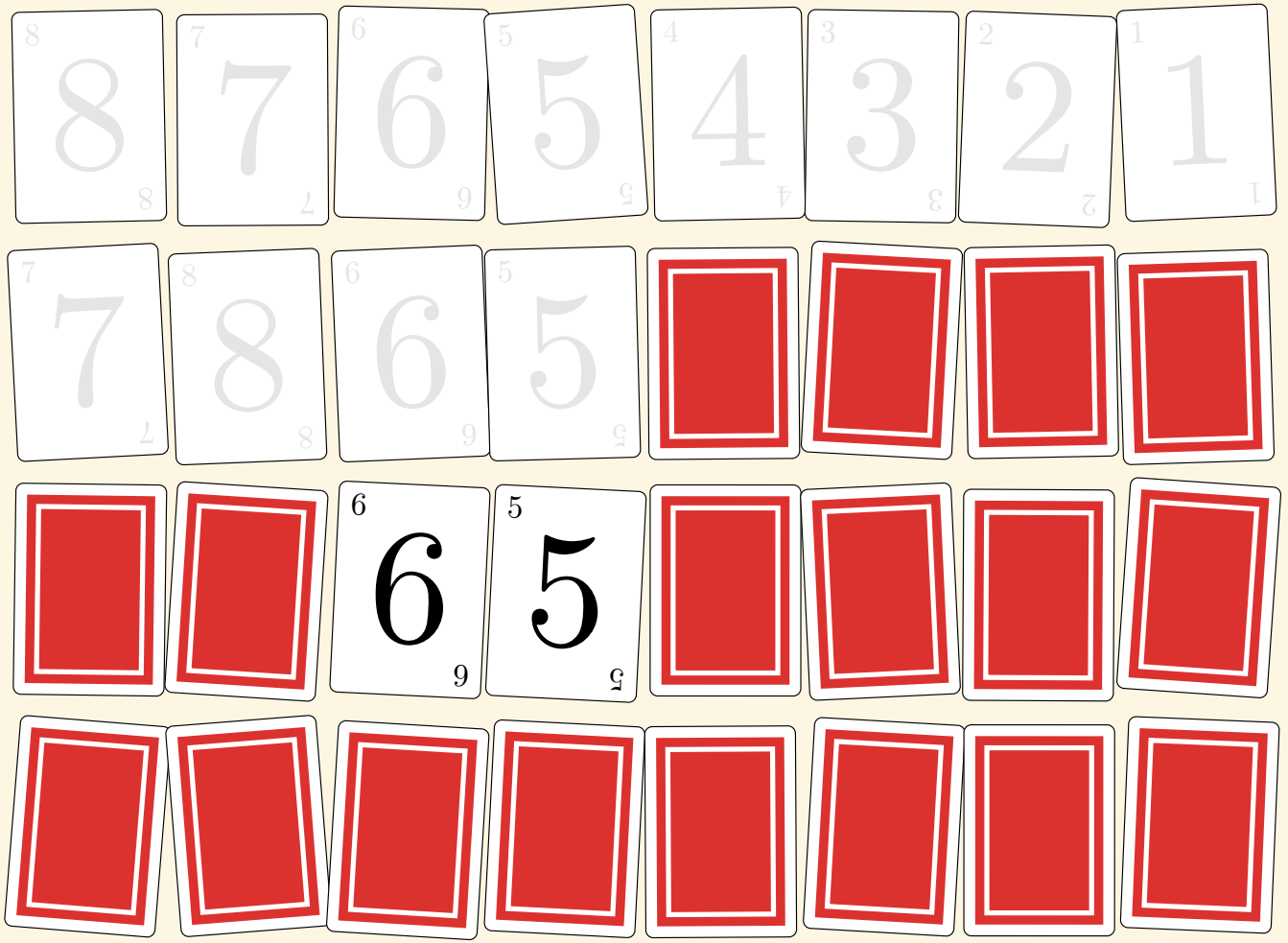
Sortér først A[1..2] og så A[3..4]

mlh@ntnu.no



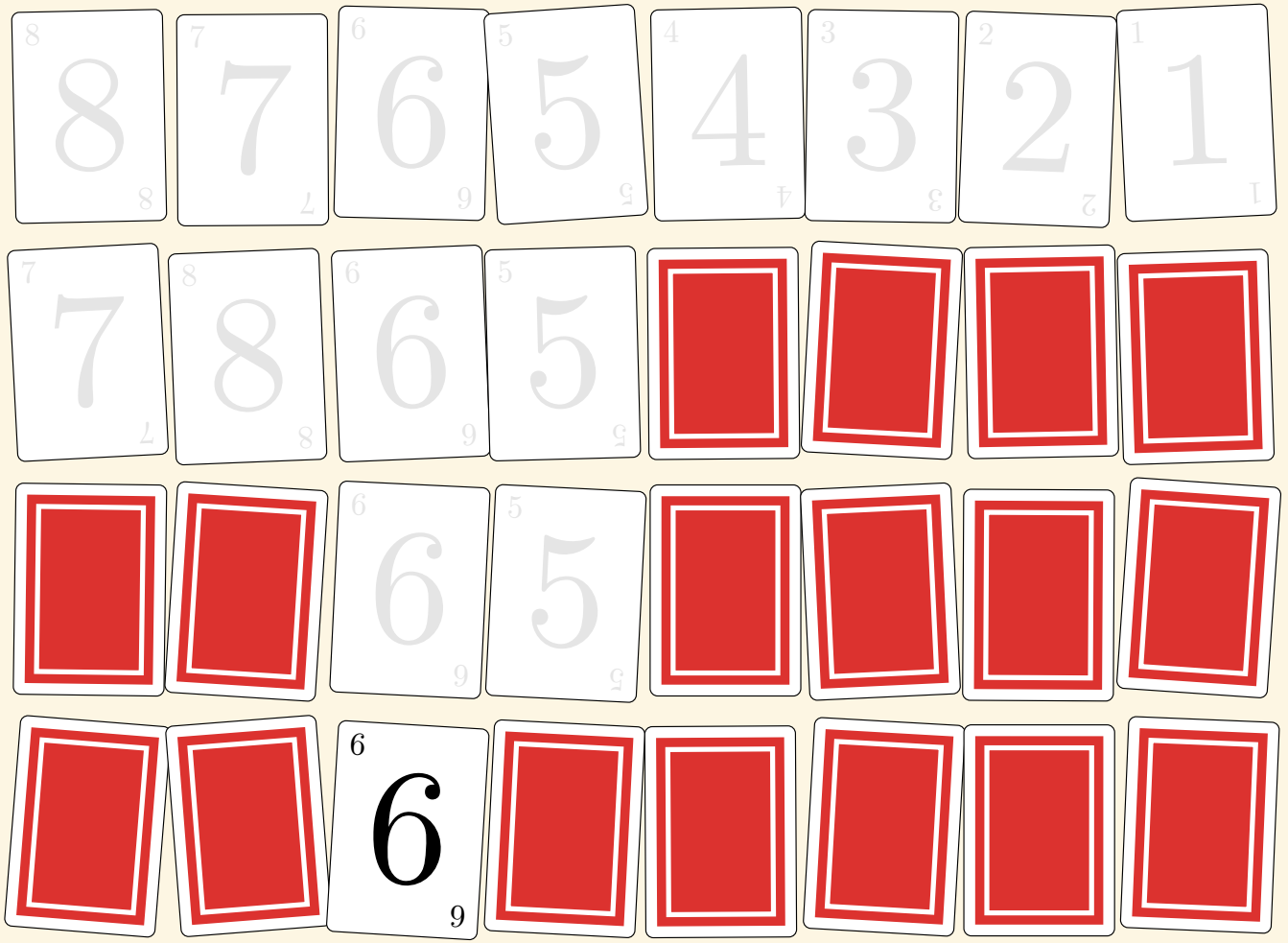
Skal sortere A[3 . . 4]

mlh@ntnu.no



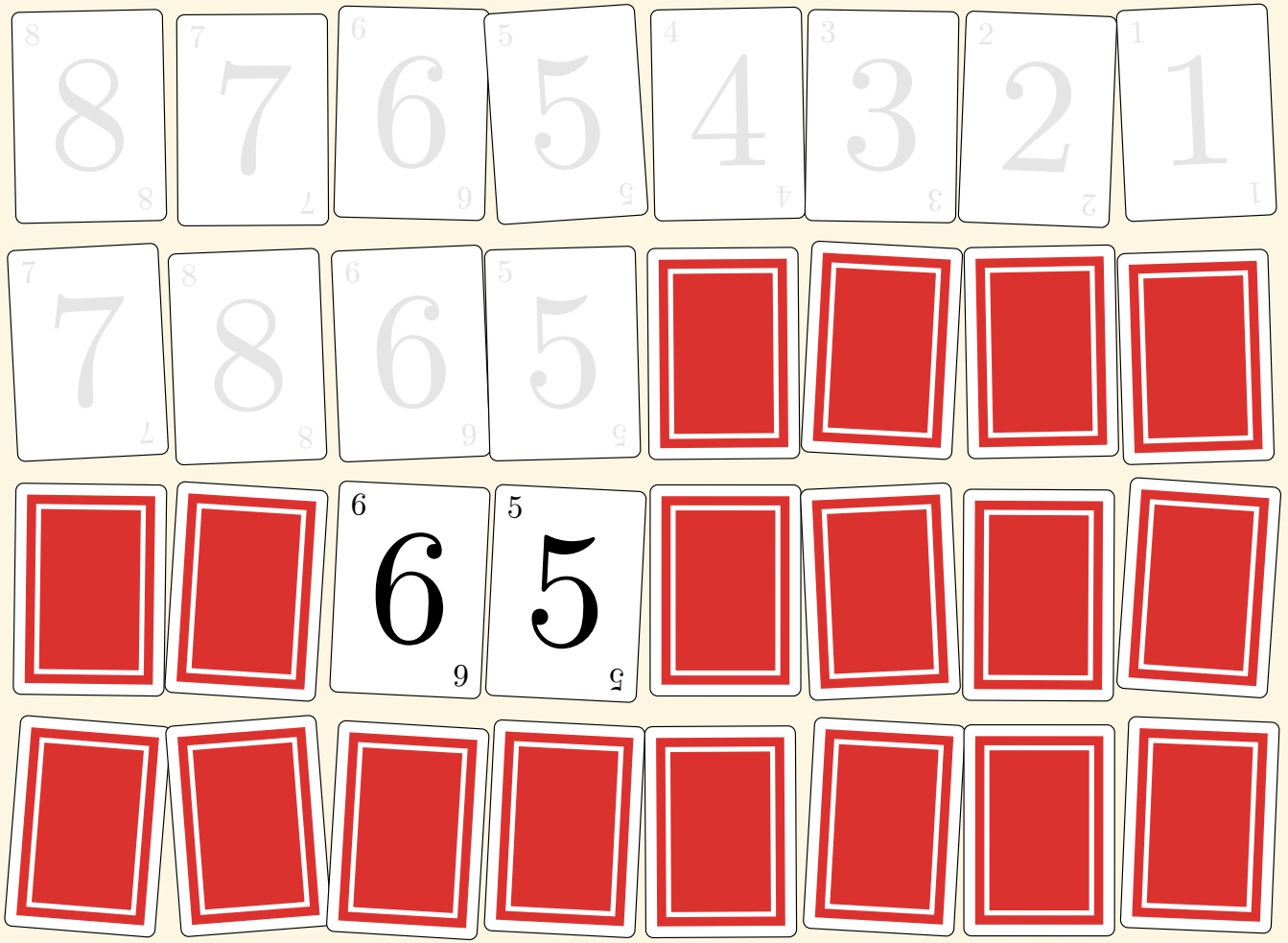
Sortér først A[3] og så A[4]

mlh@ntnu.no



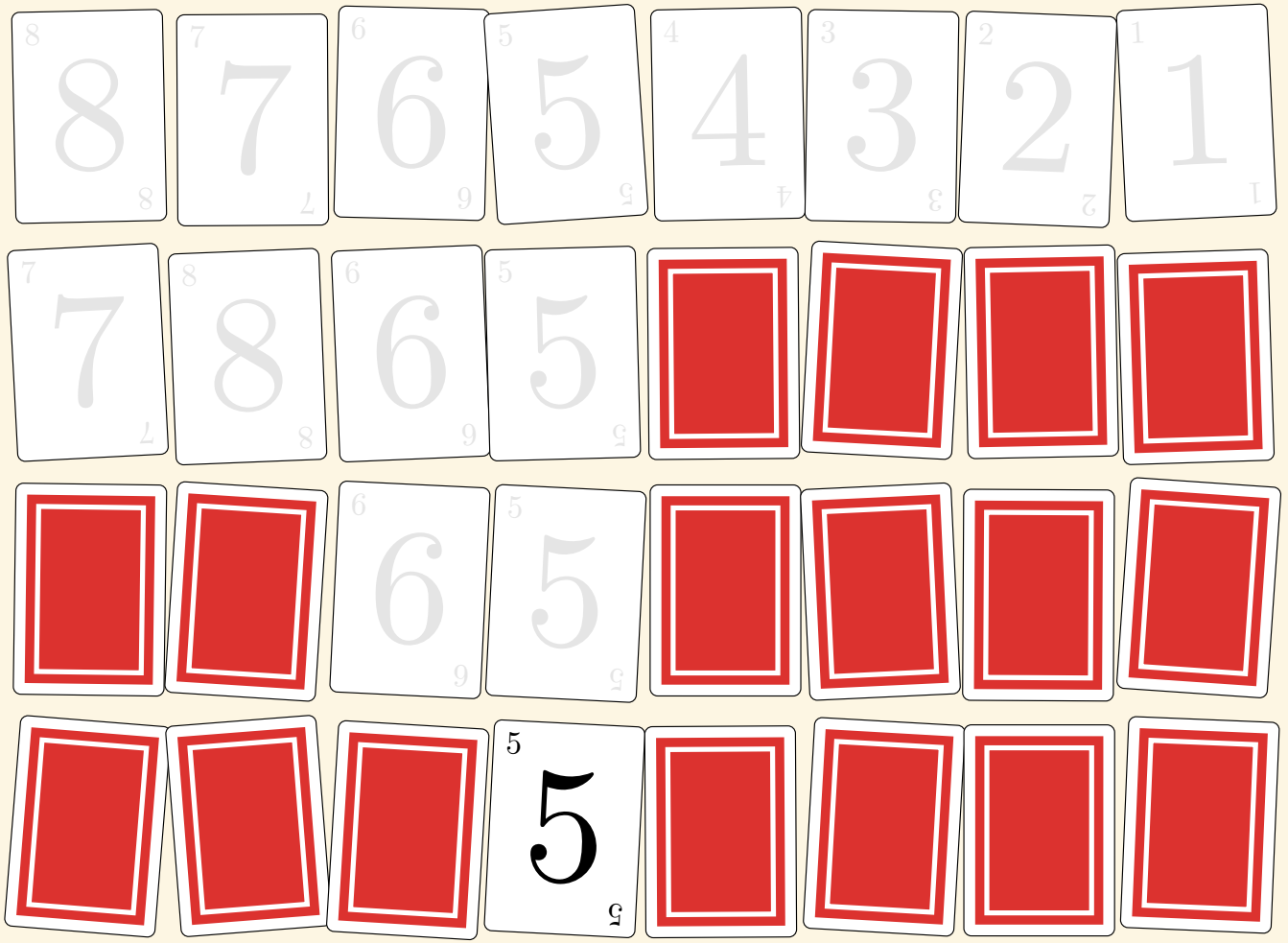
A[3] er sortert

mlh@ntnu.no



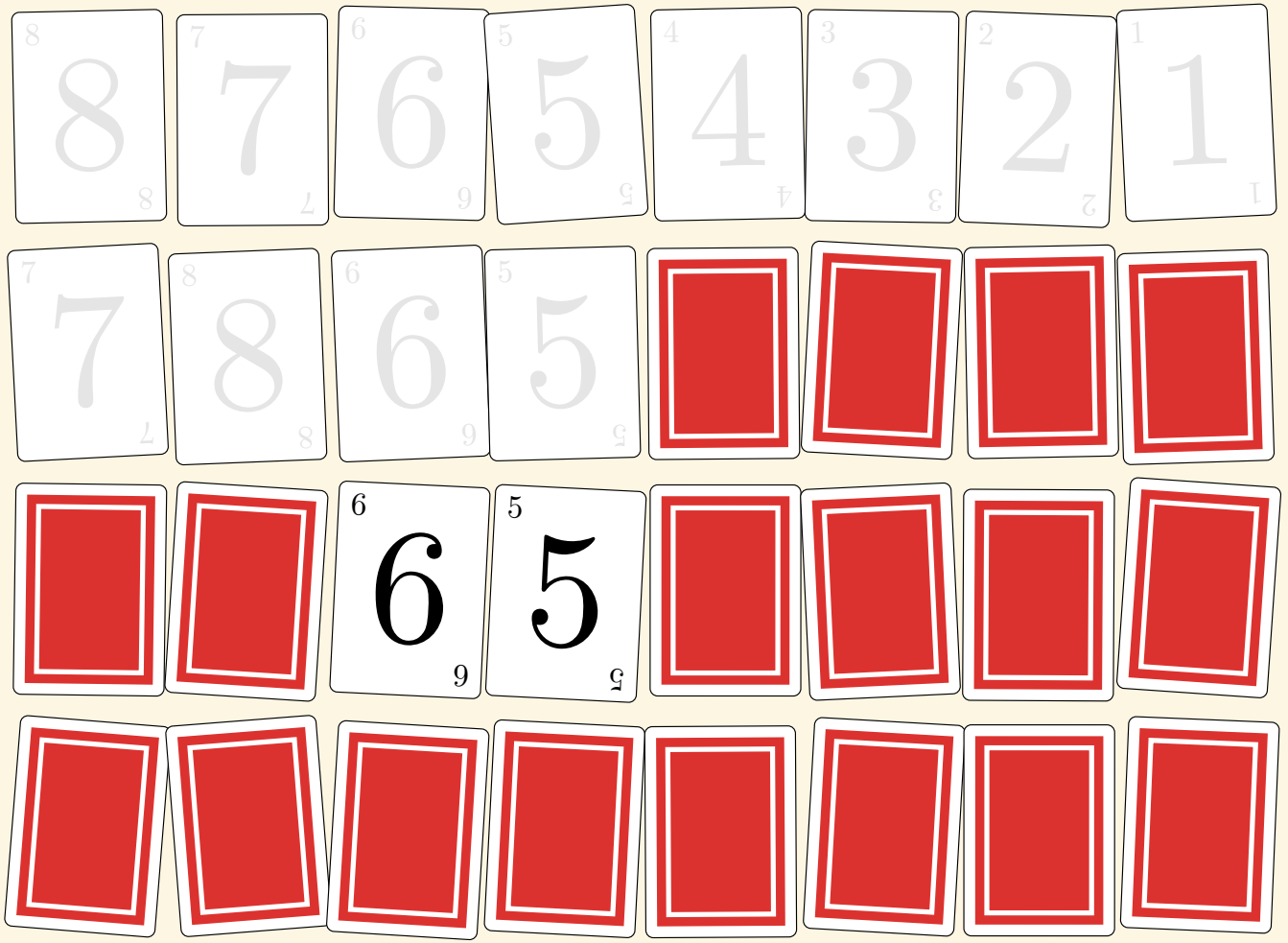
Sortér først A[3] og så A[4]

mlh@ntnu.no



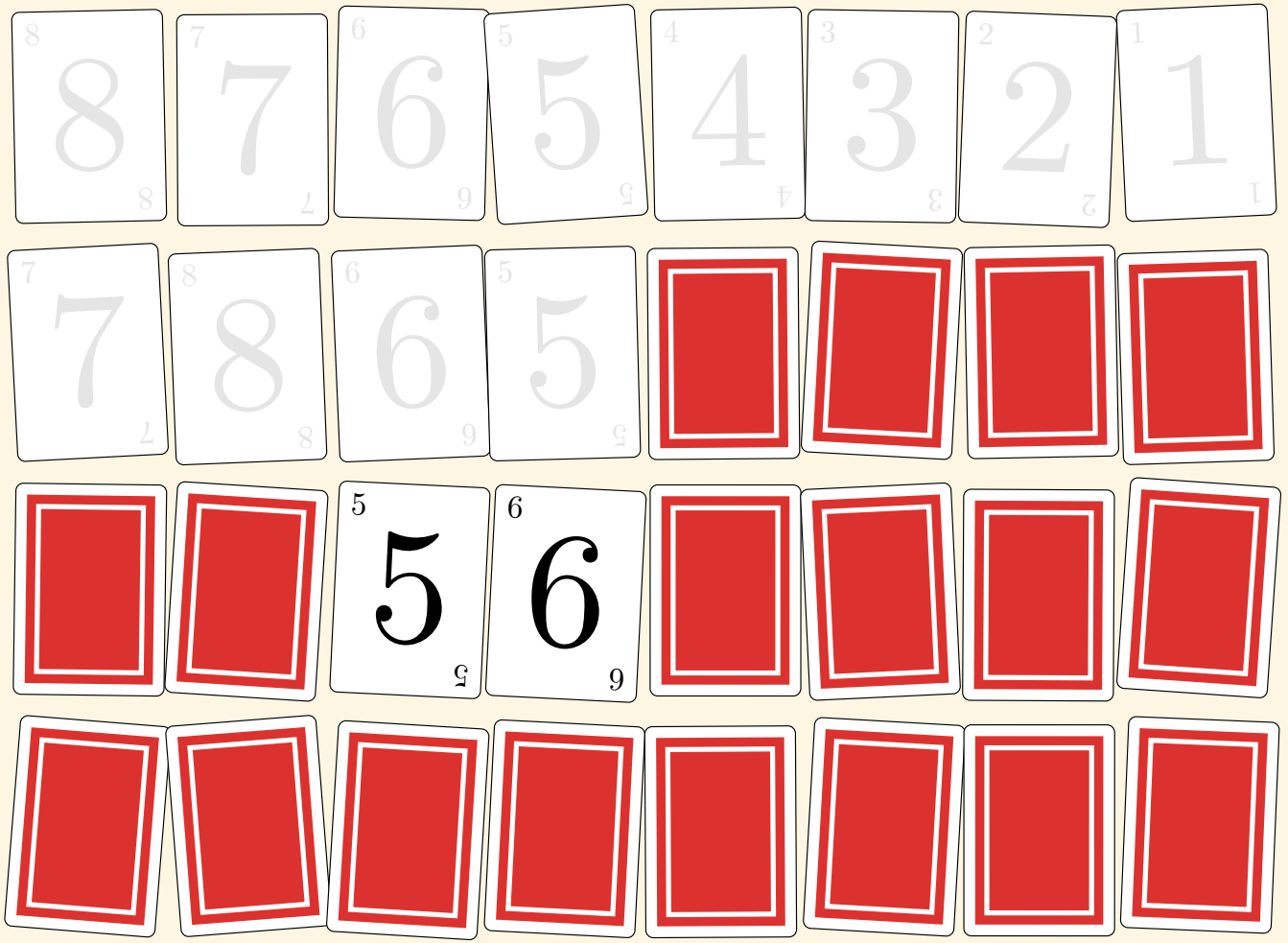
A[4] er sortert

mlh@ntnu.no



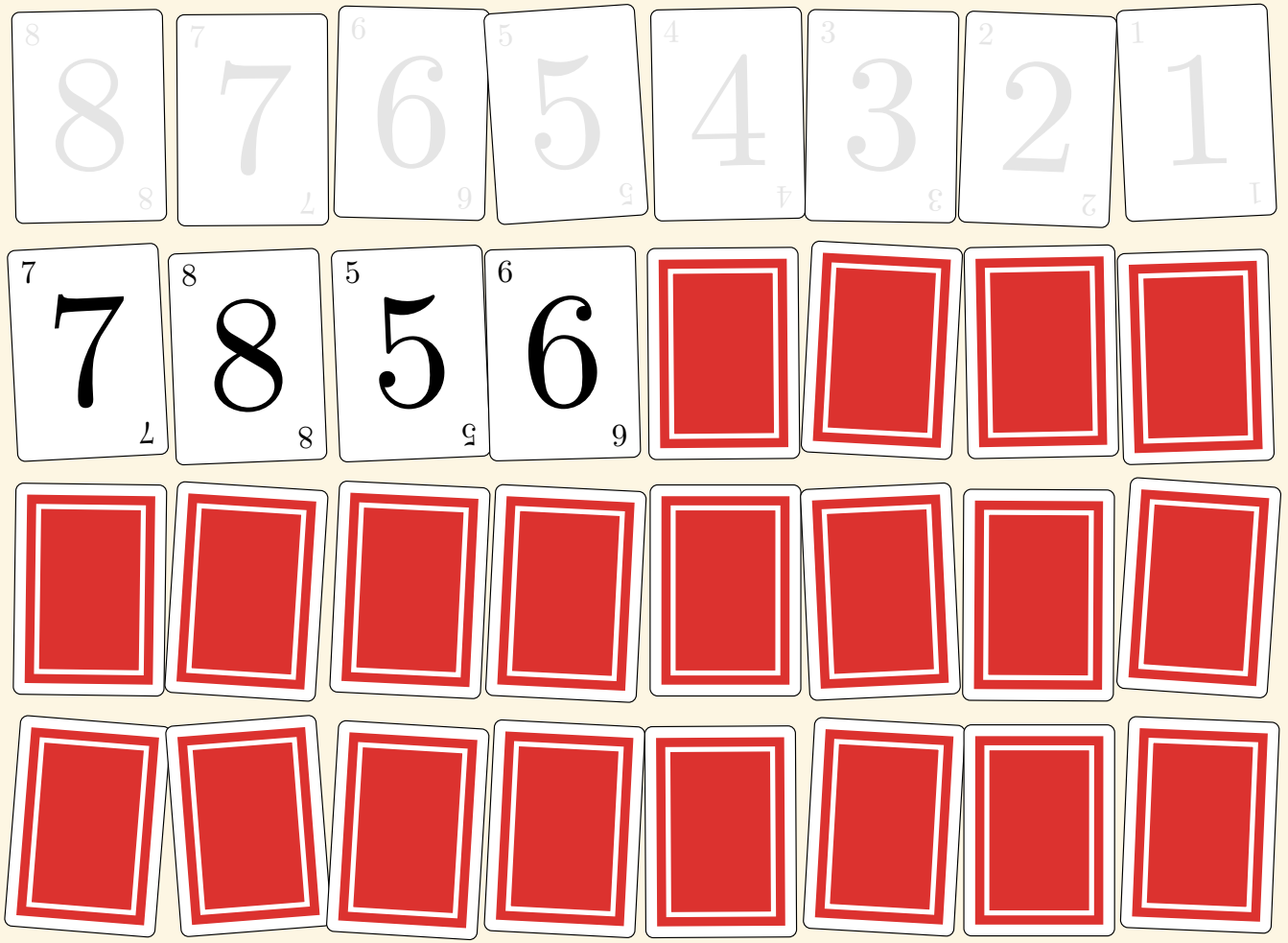
Flett sammen A[3] og A[4]

mlh@ntnu.no



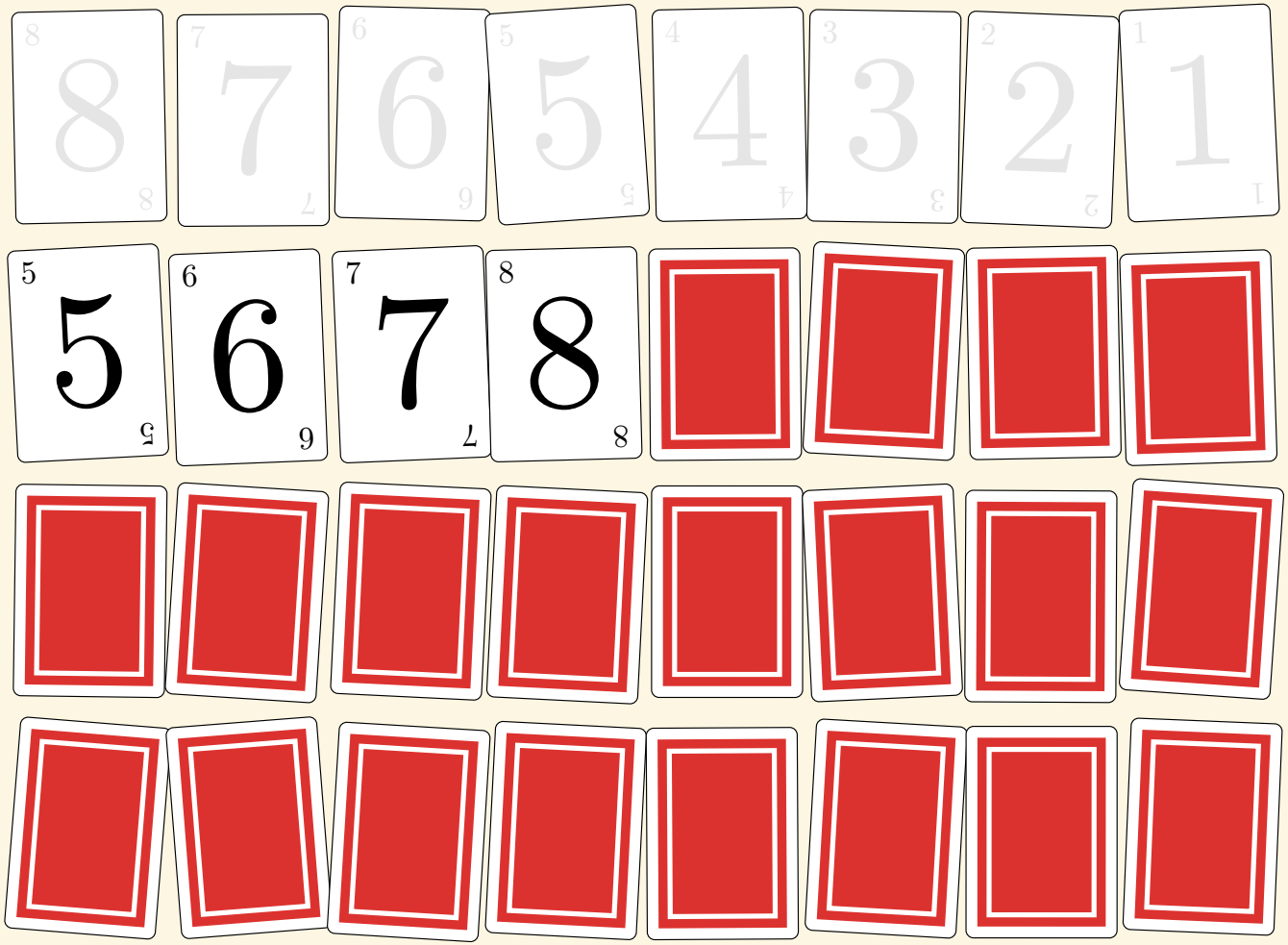
A[3 . . 4] er sortert

mlh@ntnu.no



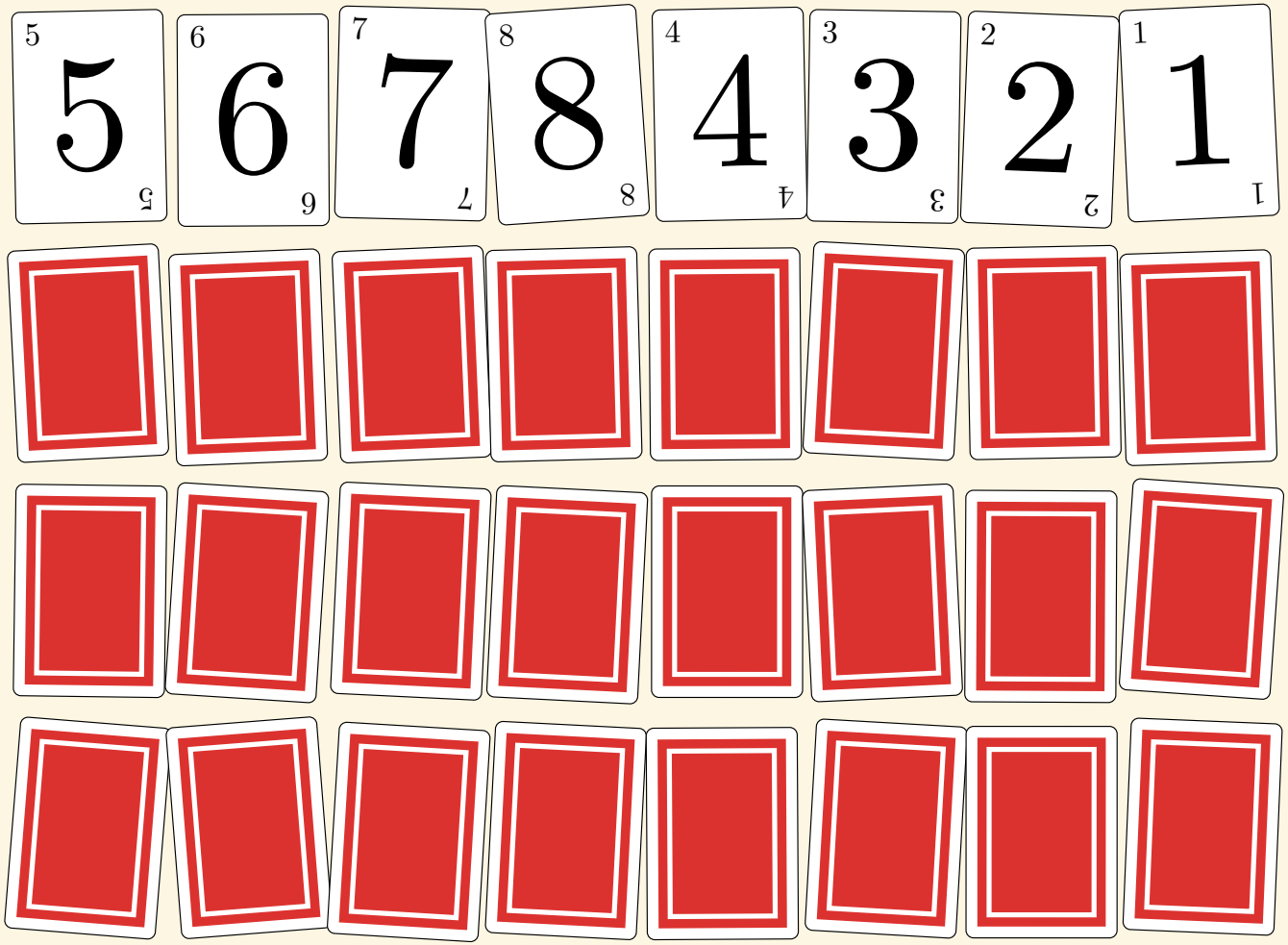
Flett sammen A[1 . . 2] og A[3 . . 4]

mlh@ntnu.no



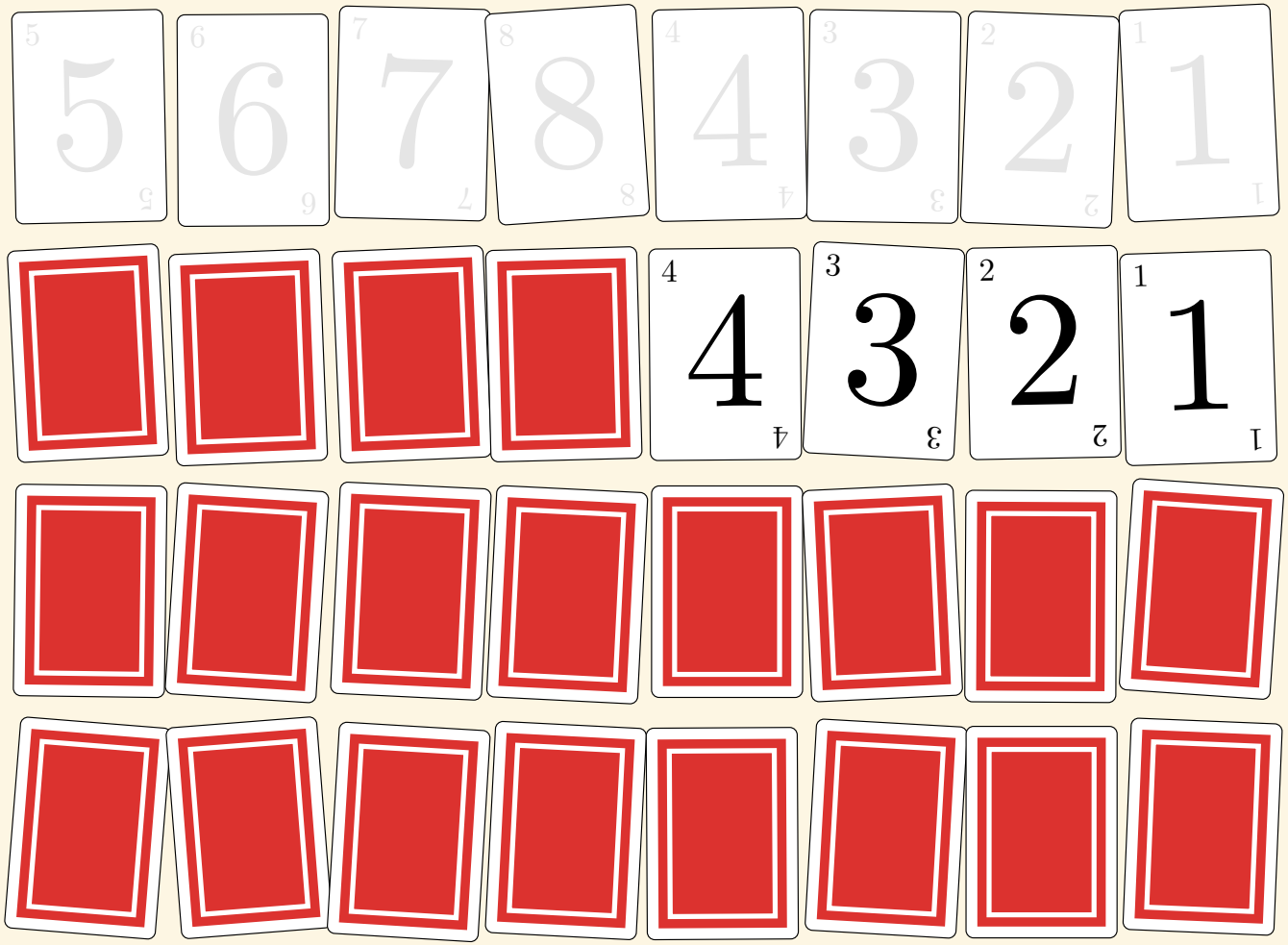
A[1 . . 4] er sortert

mlh@ntnu.no



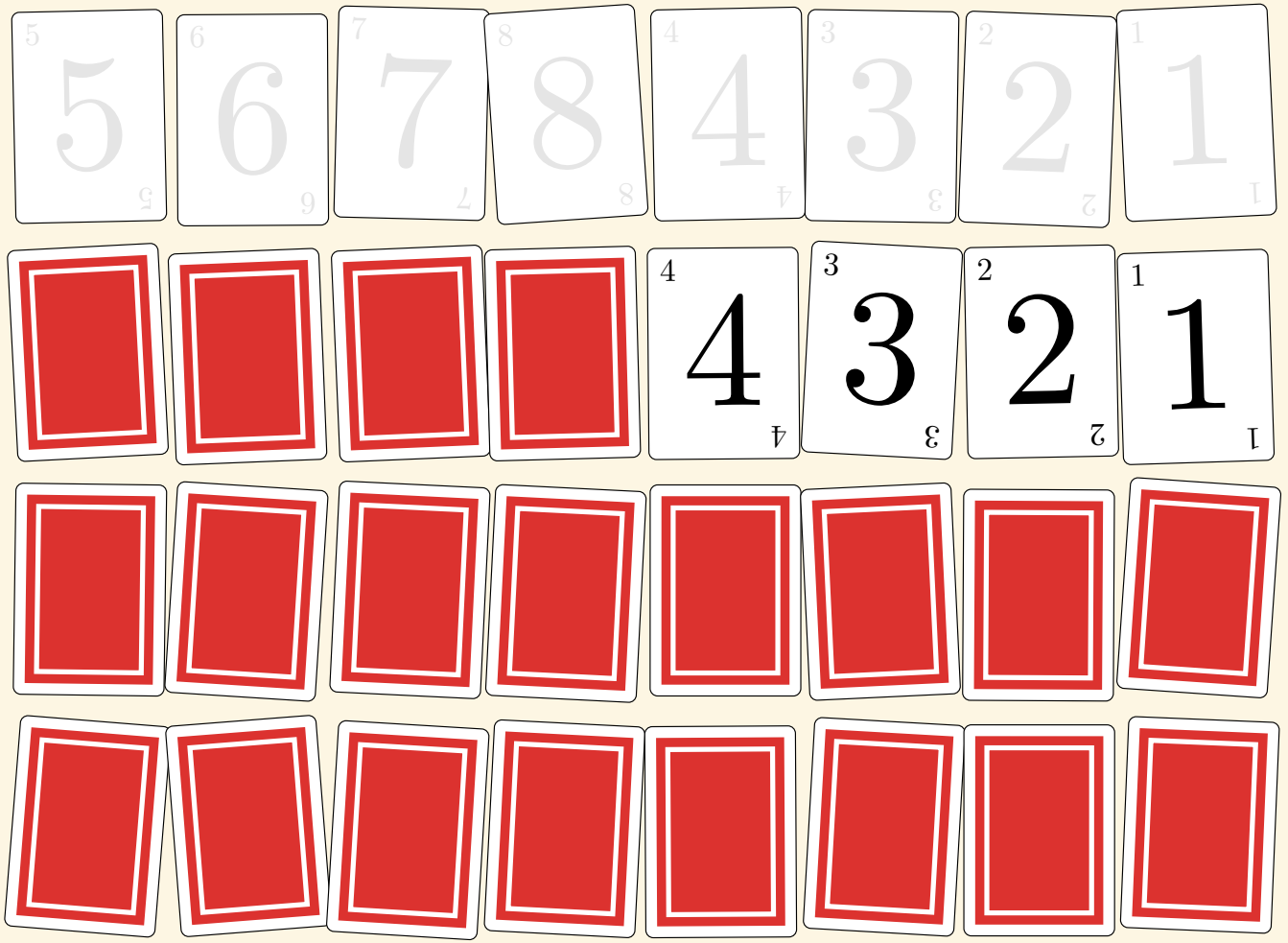
Sortér først A[1..4] og så A[5..8]

mlh@ntnu.no



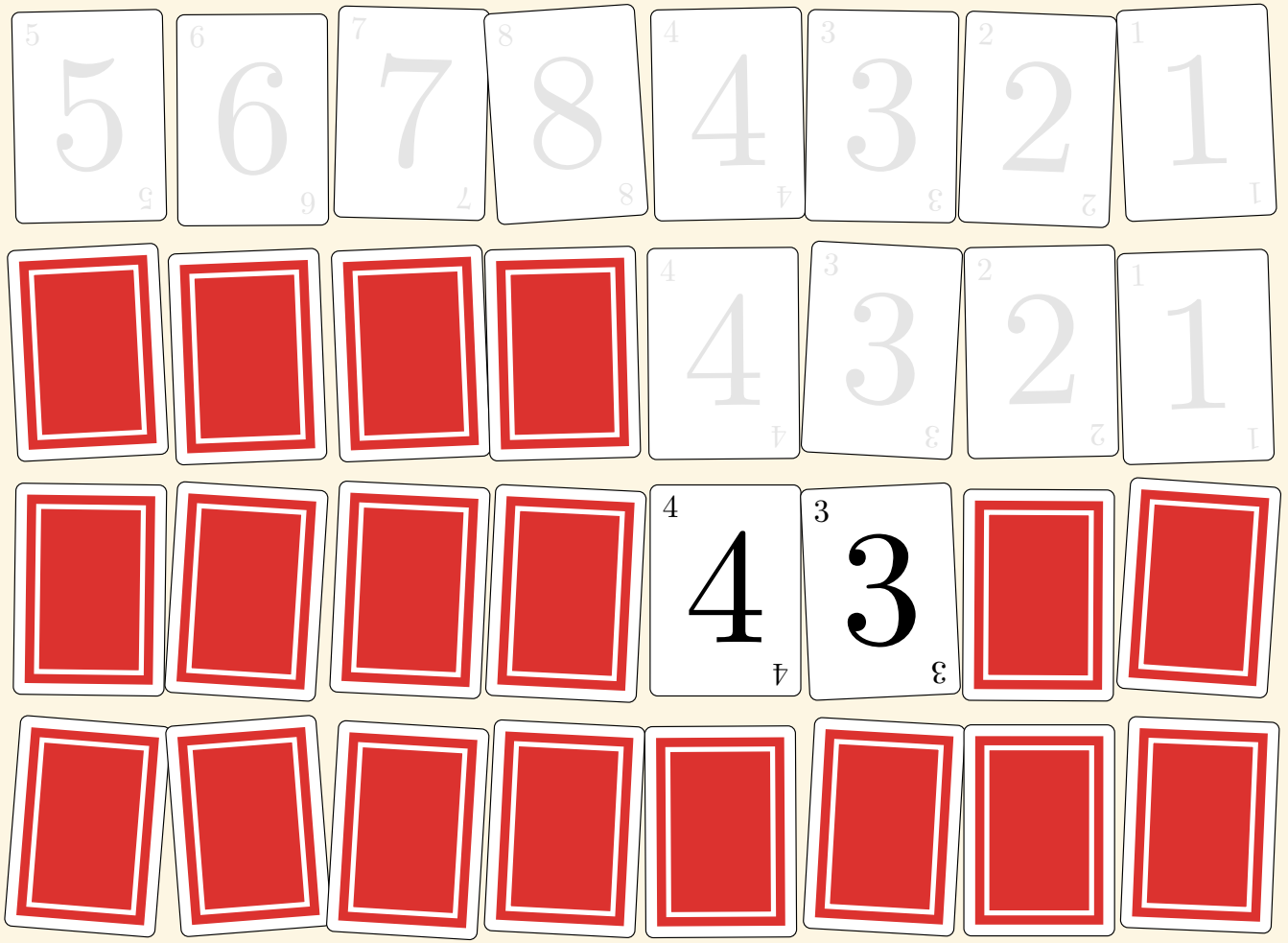
Skal sortere A[5 . . 8]

mlh@ntnu.no



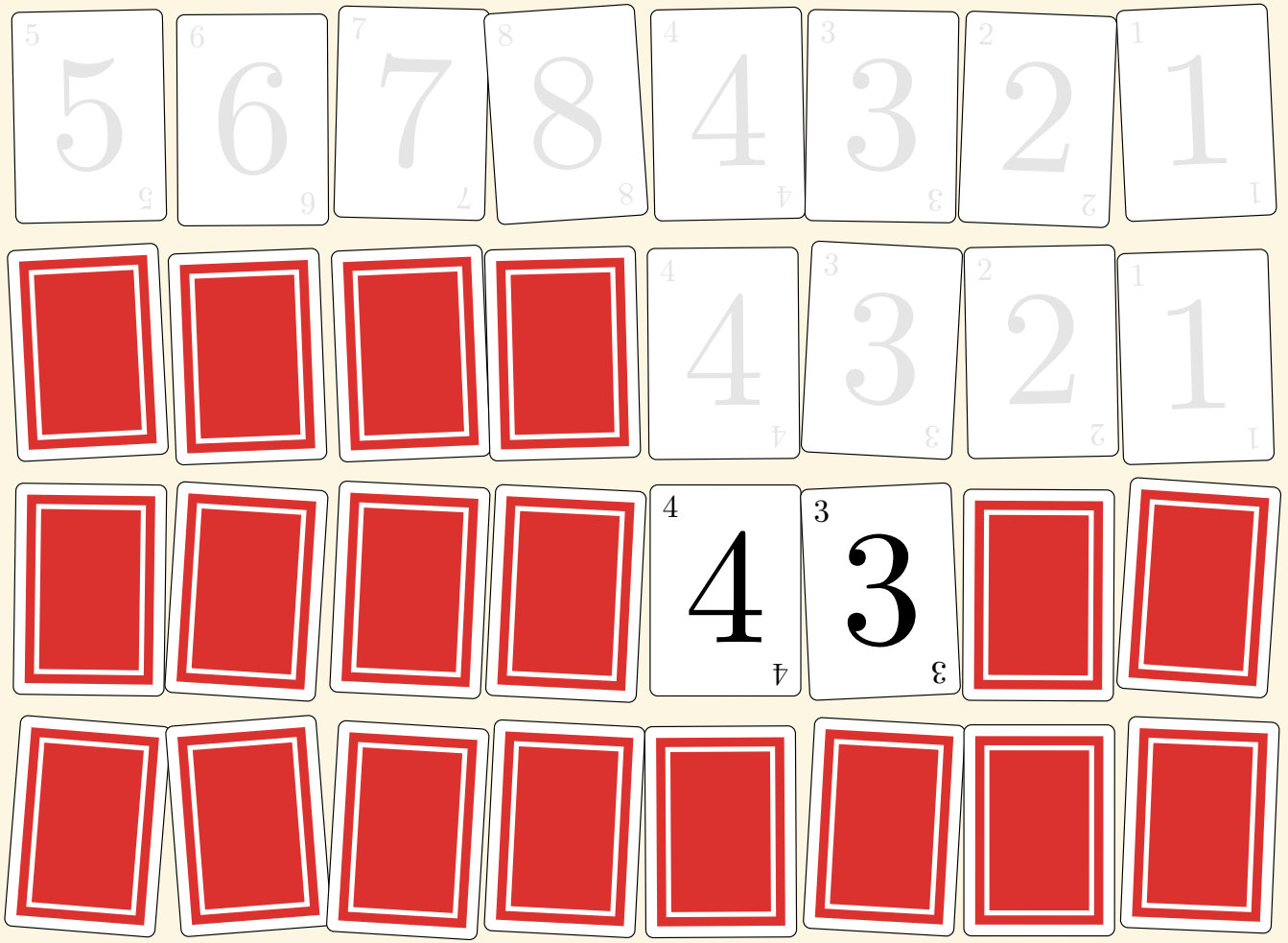
Sortér først A[5 . . 6] og så A[7 . . 8]

mlh@ntnu.no



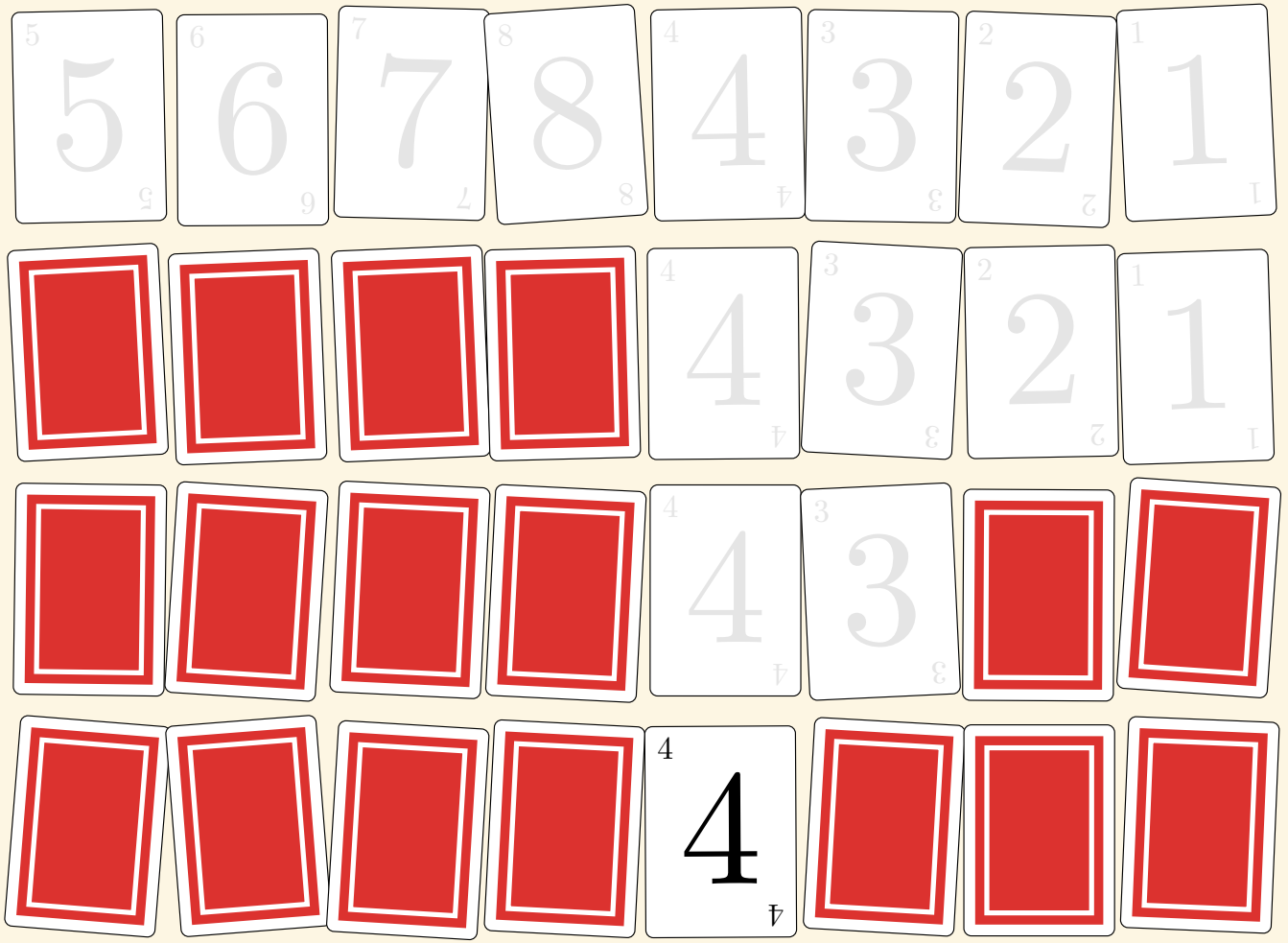
Skal sortere A[5 . . 6]

mlh@ntnu.no



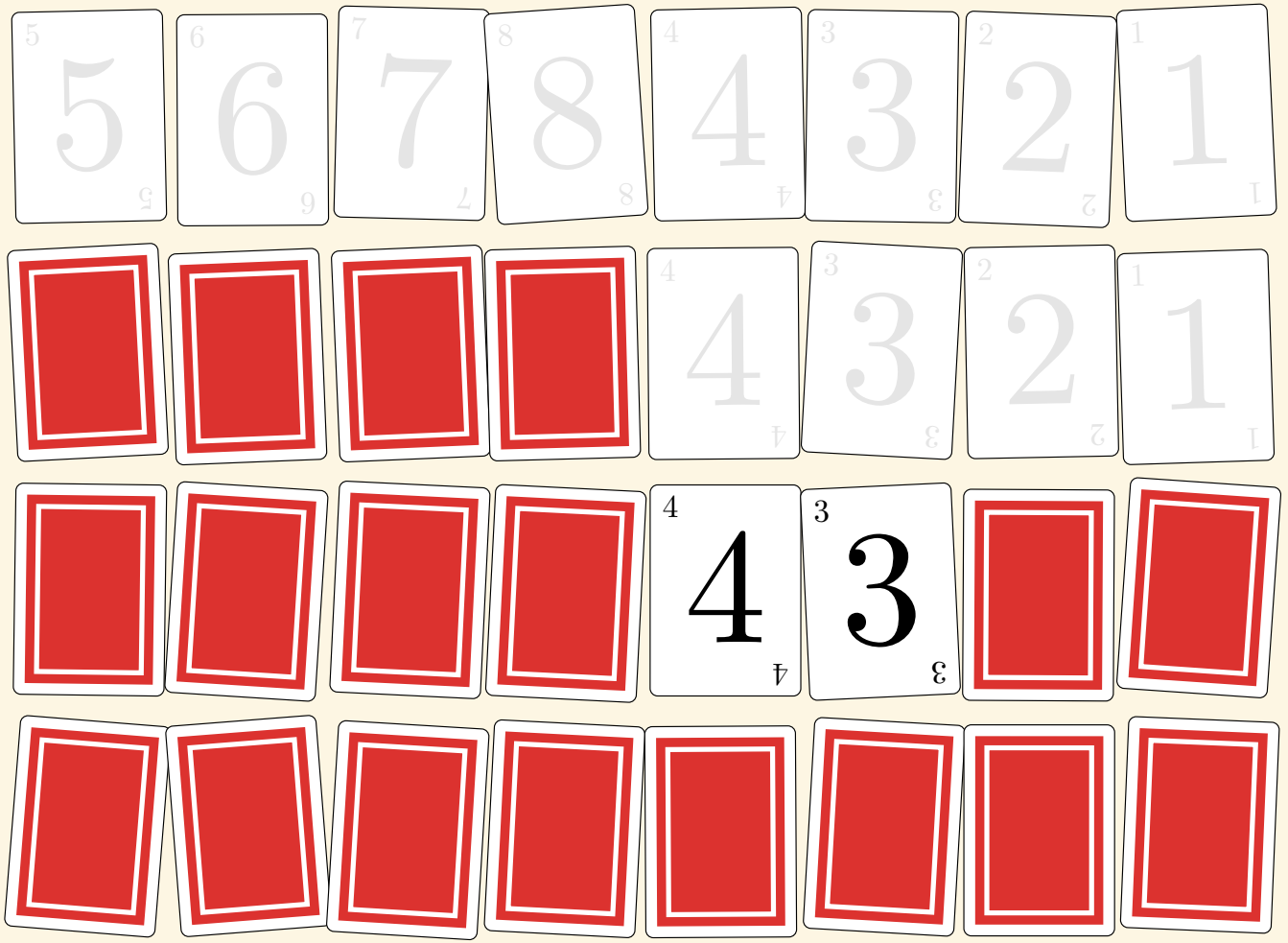
Sortér først A[5] og så A[6]

mlh@ntnu.no



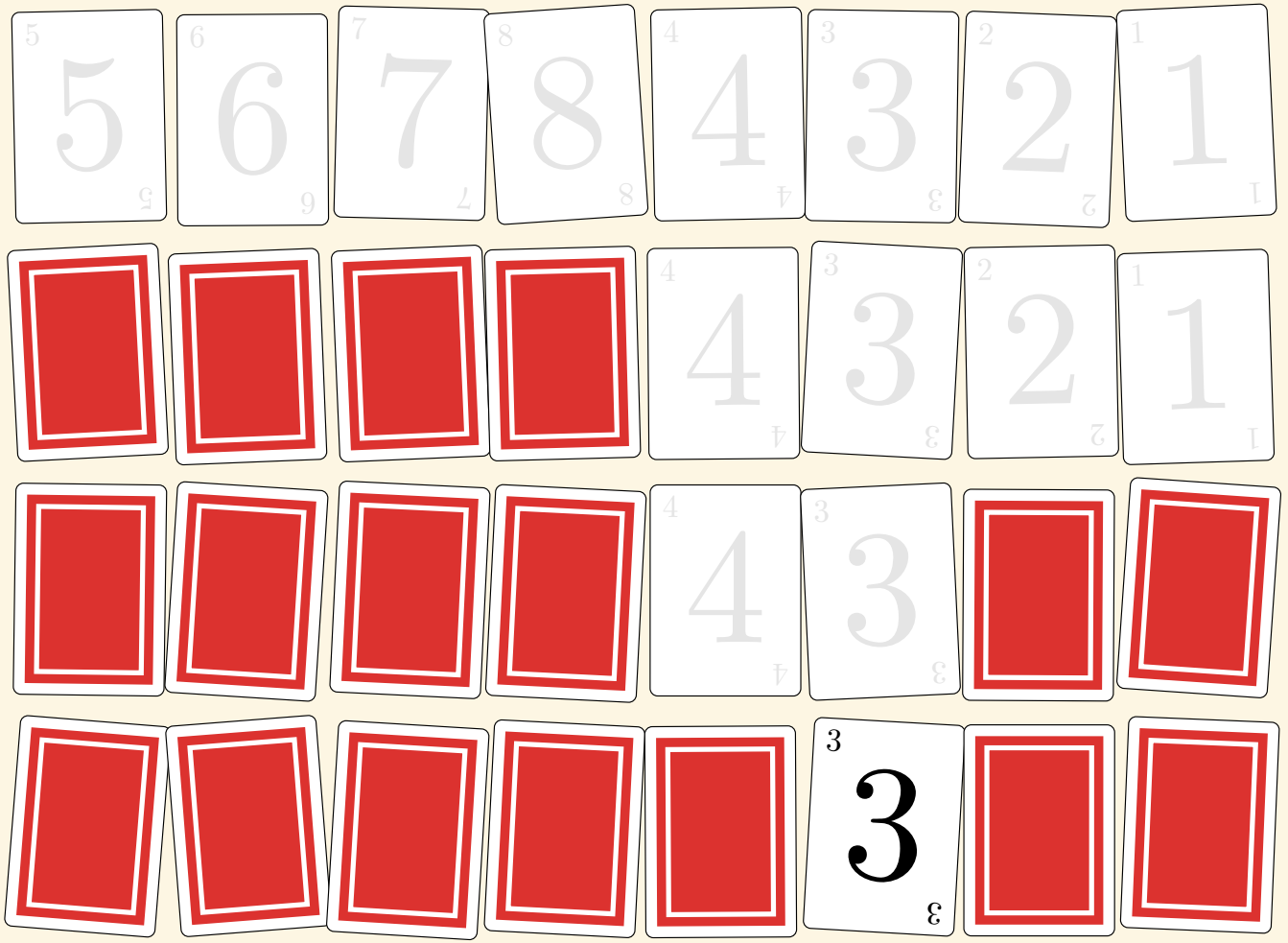
A[5] er sortert

mlh@ntnu.no



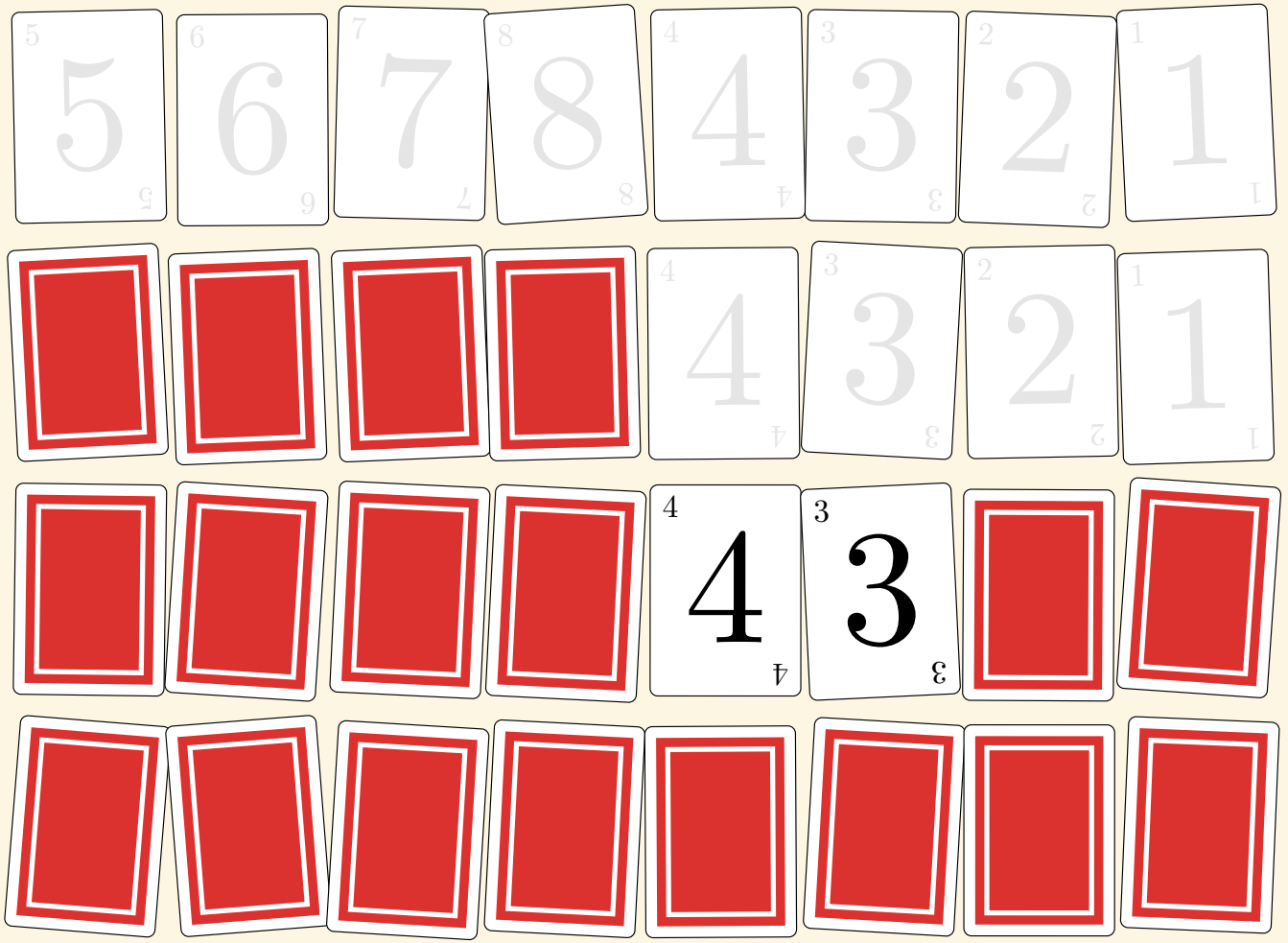
Sortér først A[5] og så A[6]

mlh@ntnu.no



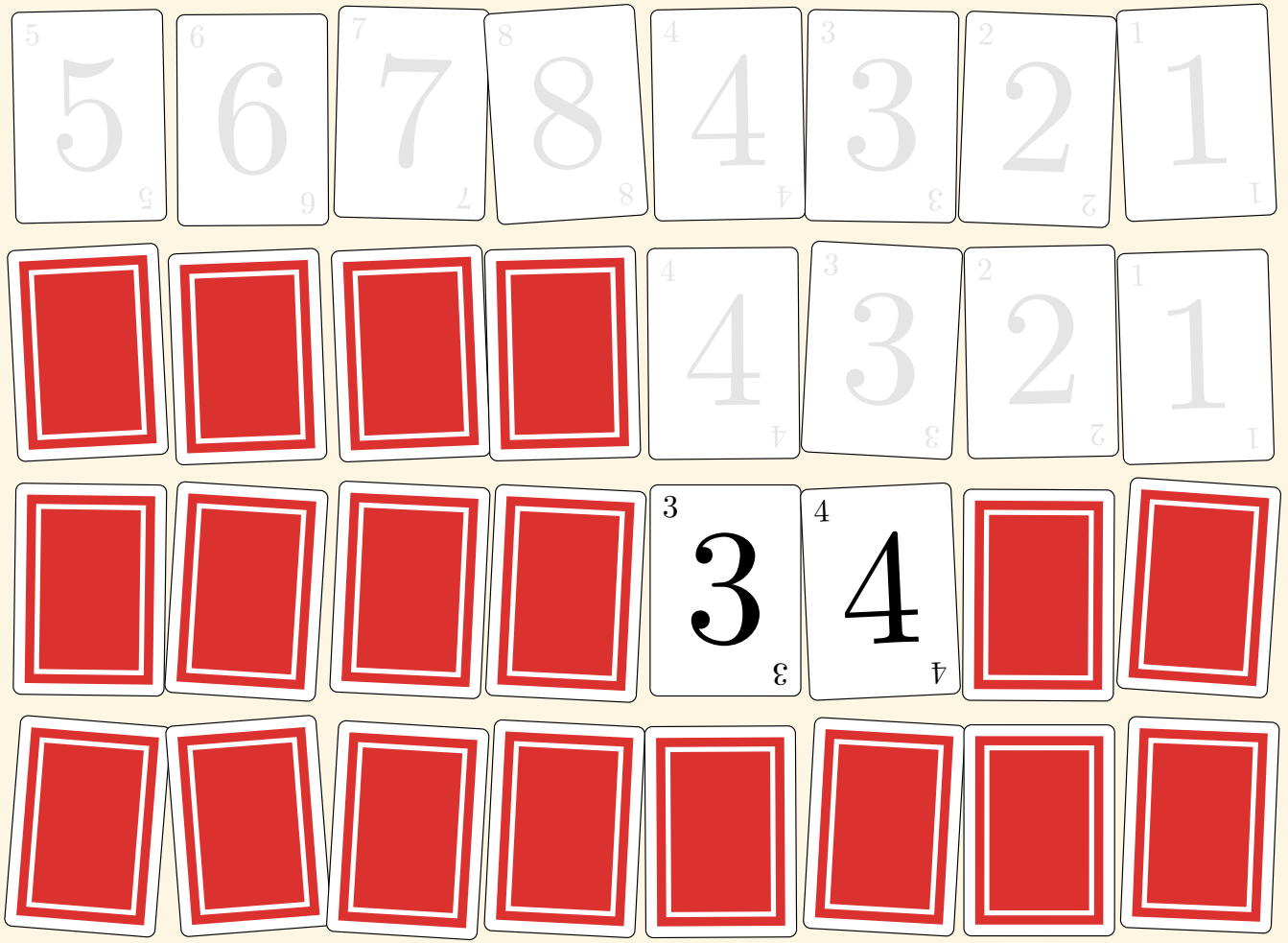
A[6] er sortert

mlh@ntnu.no



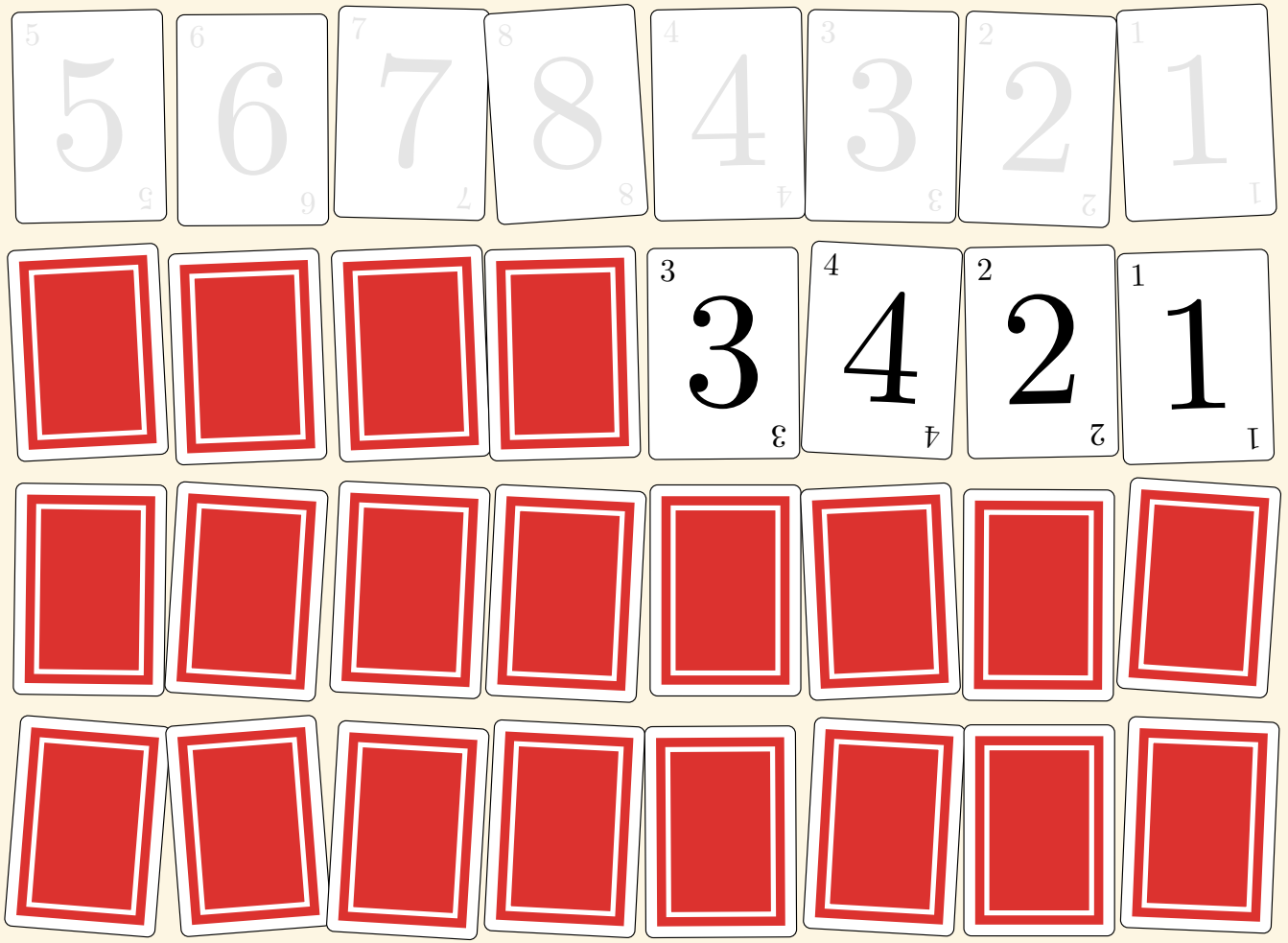
Flett sammen A[5] og A[6]

mlh@ntnu.no



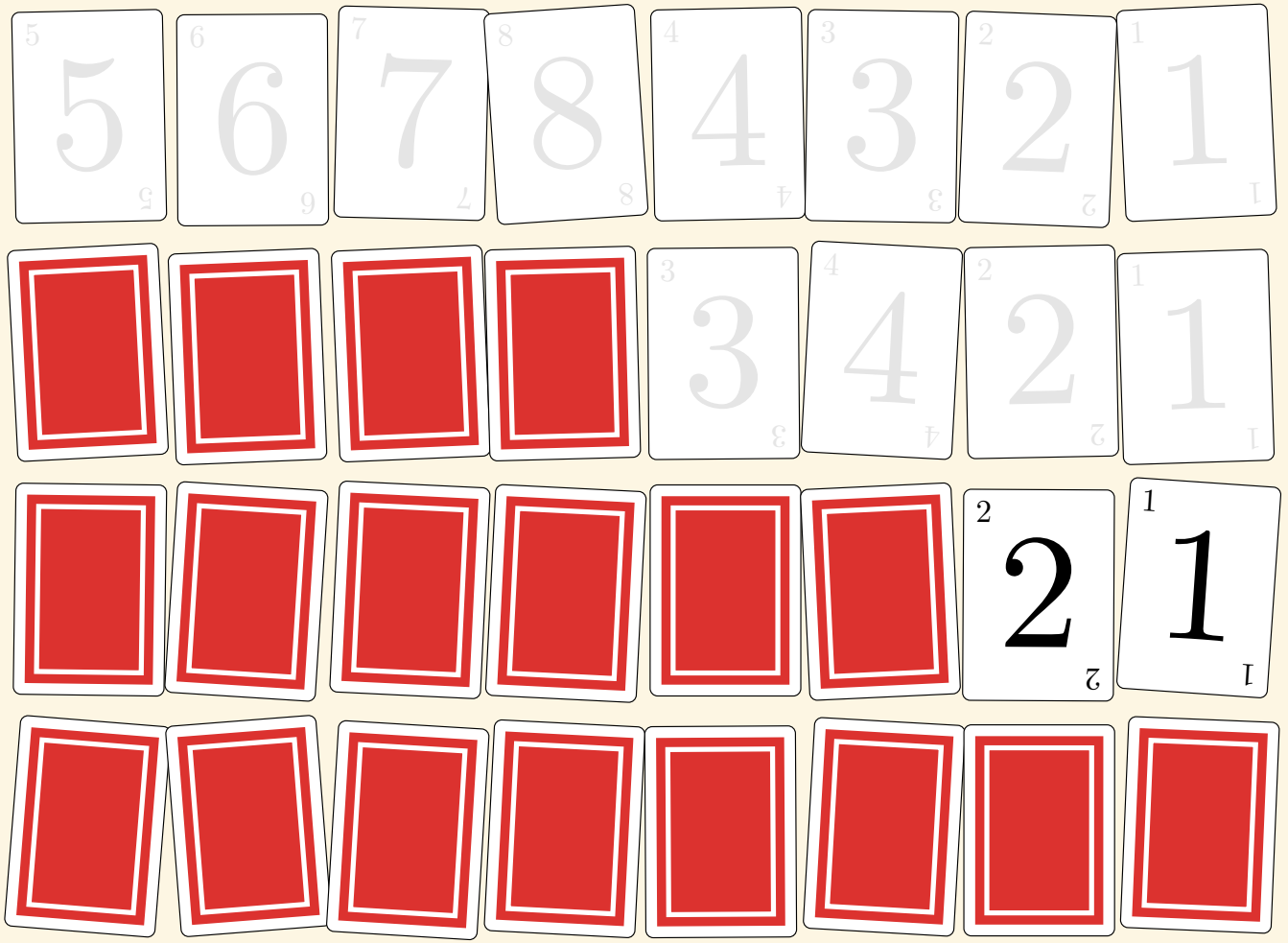
A[5 . . 6] er sortert

mlh@ntnu.no



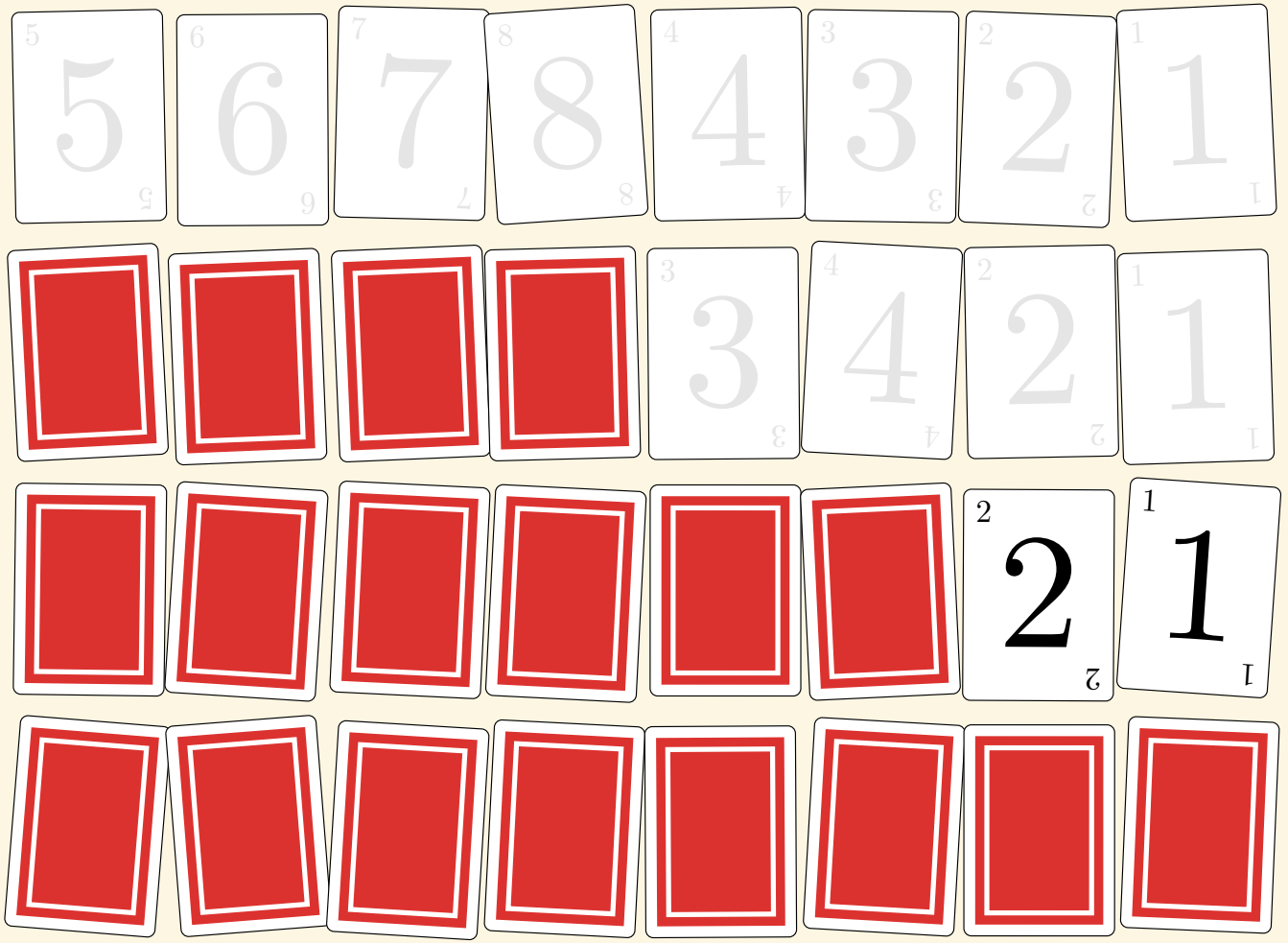
Sortér først A[5 . . 6] og så A[7 . . 8]

mlh@ntnu.no



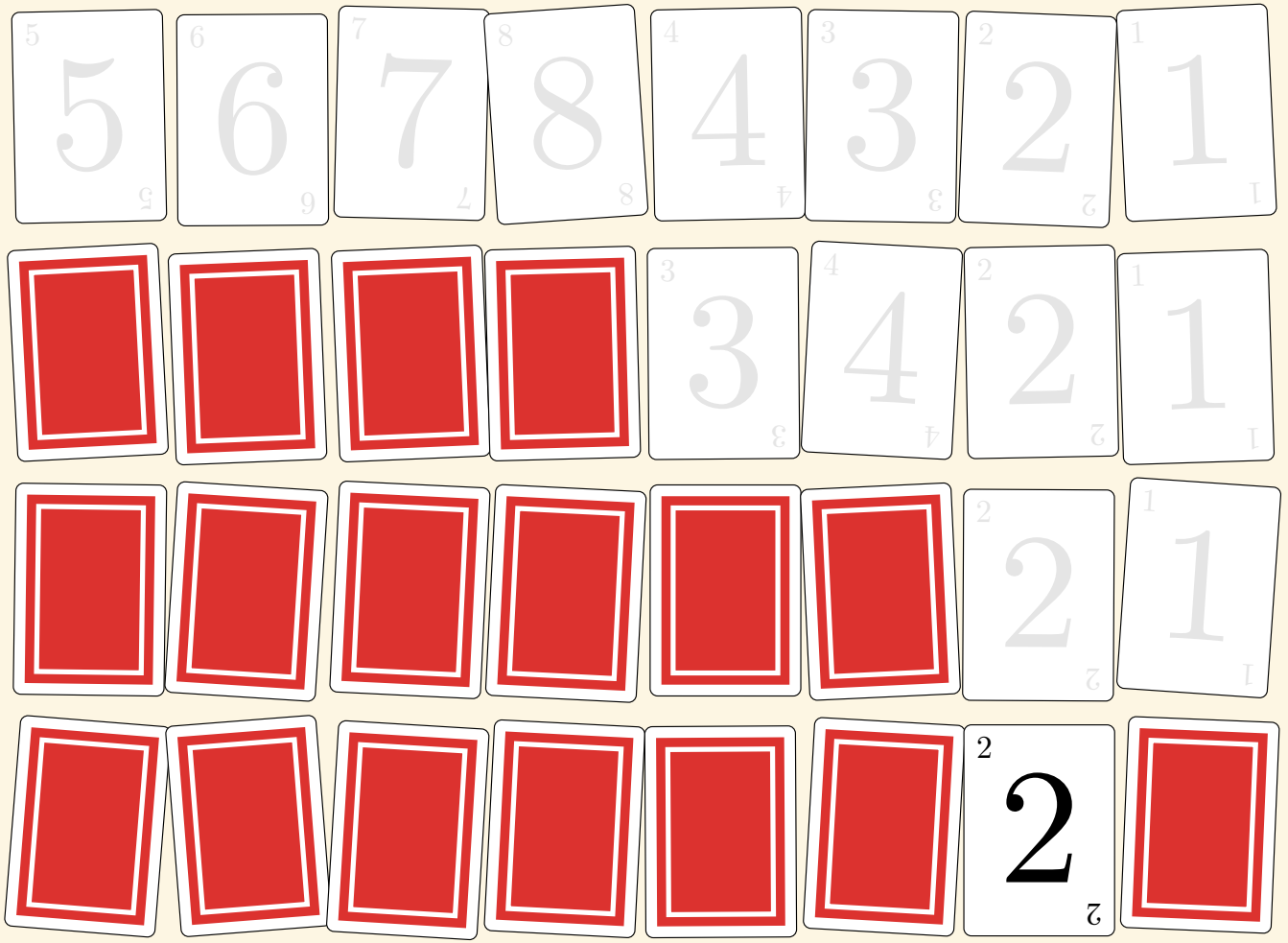
Skal sortere A[7 . . 8]

mlh@ntnu.no



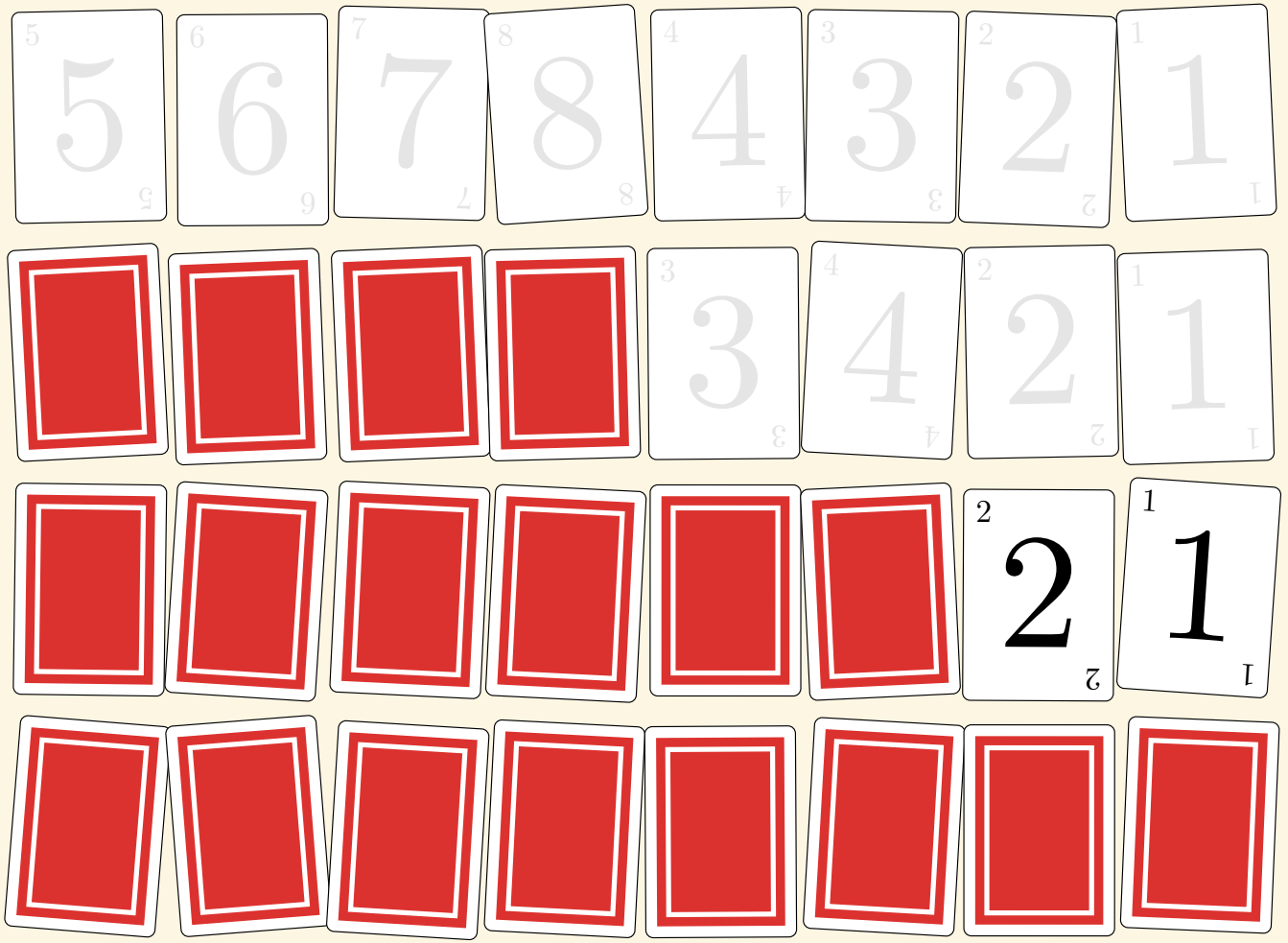
Sortér først A[7] og så A[8]

mlh@ntnu.no



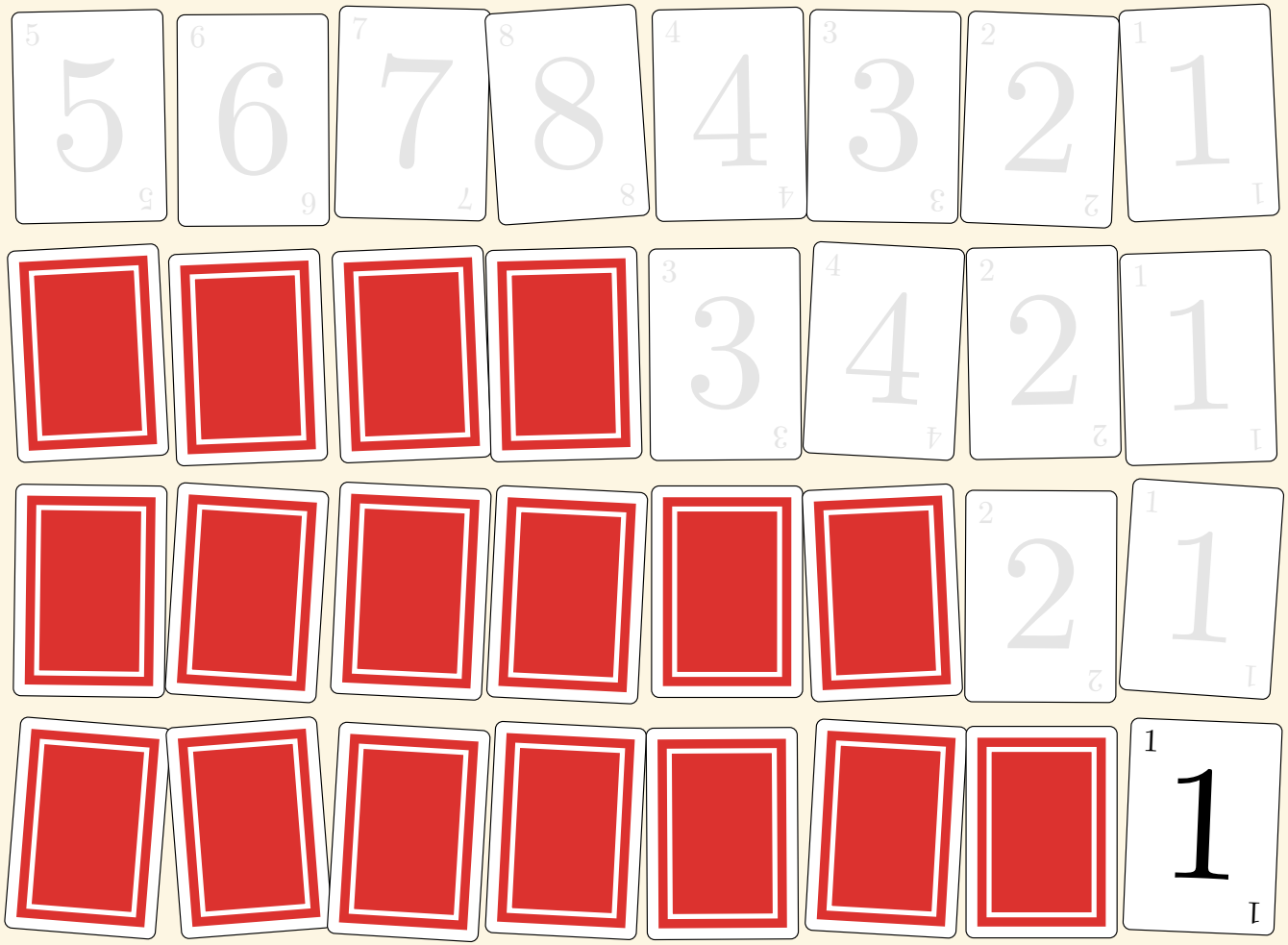
A[7] er sortert

mlh@ntnu.no



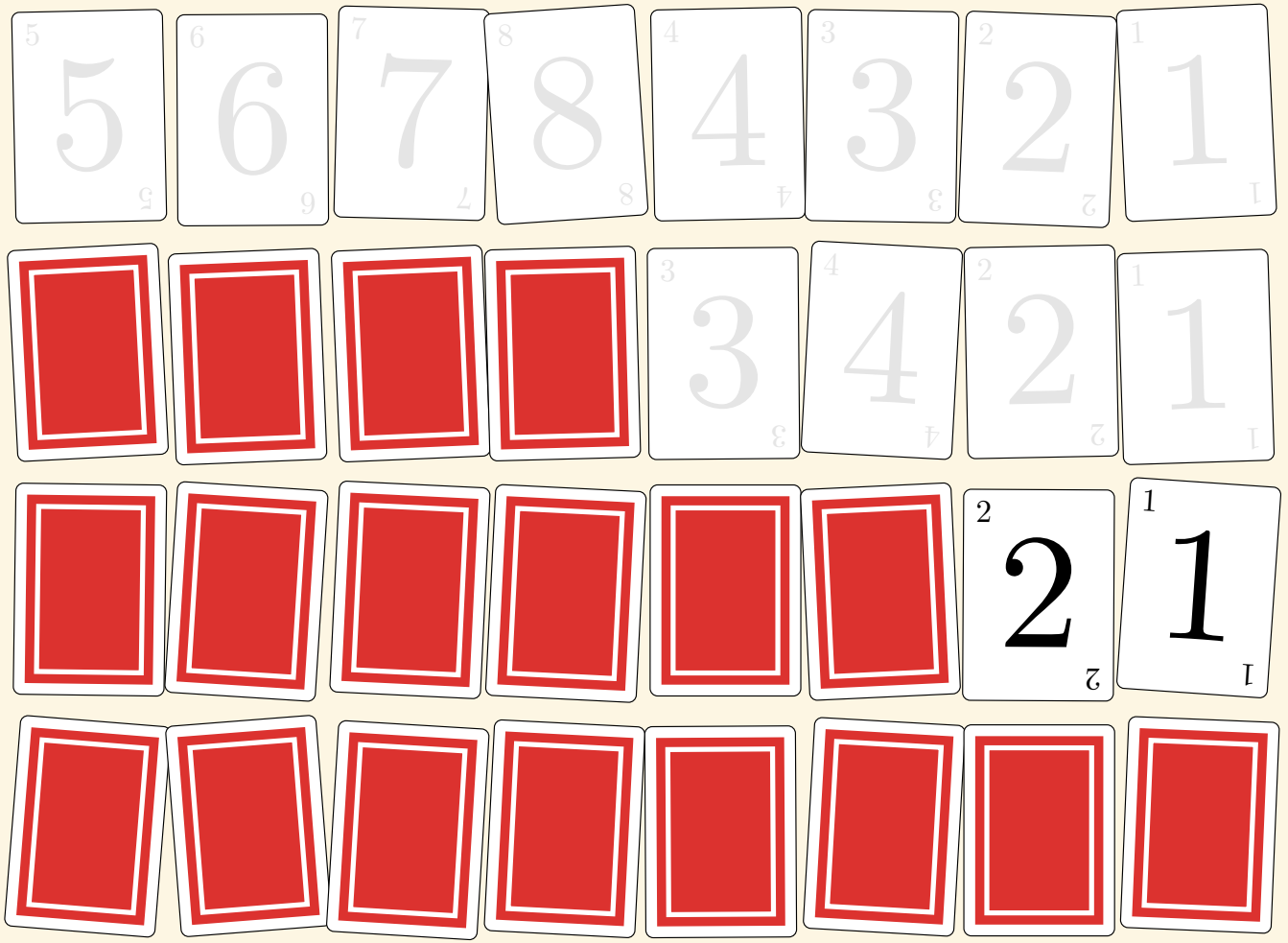
Sortér først A[7] og så A[8]

mlh@ntnu.no



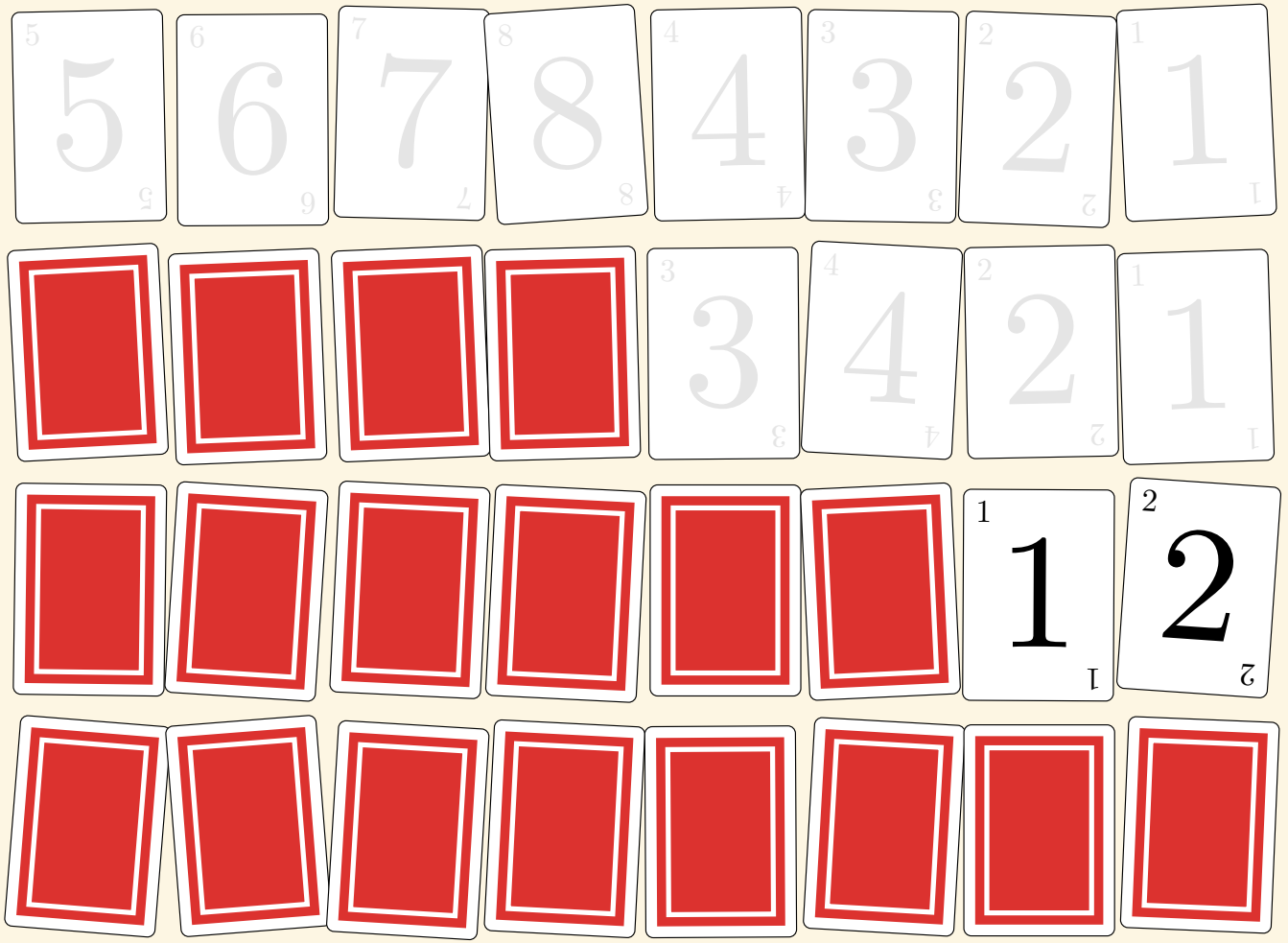
A[8] er sortert

mlh@ntnu.no



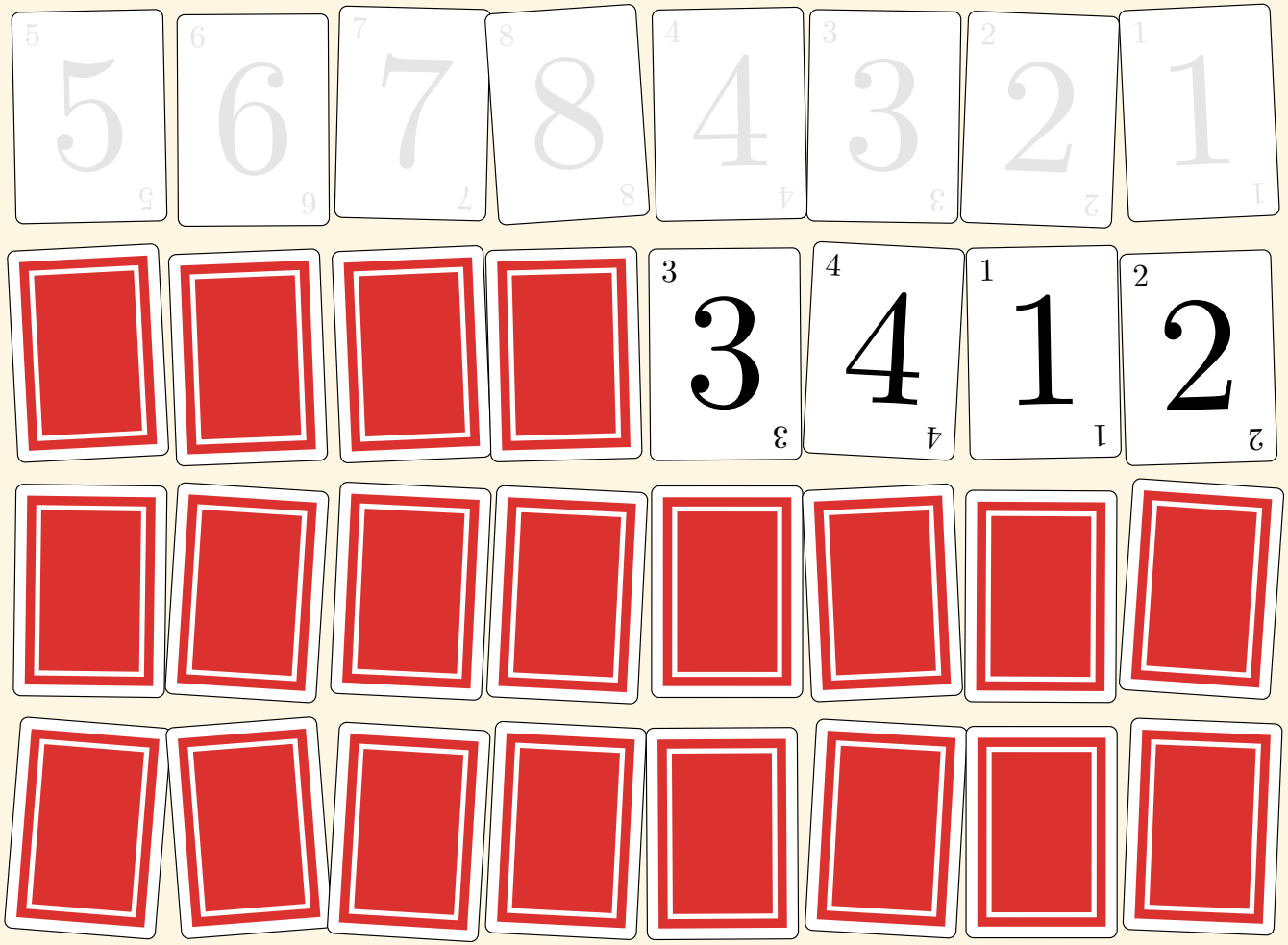
Flett sammen A[7] og A[8]

mlh@ntnu.no



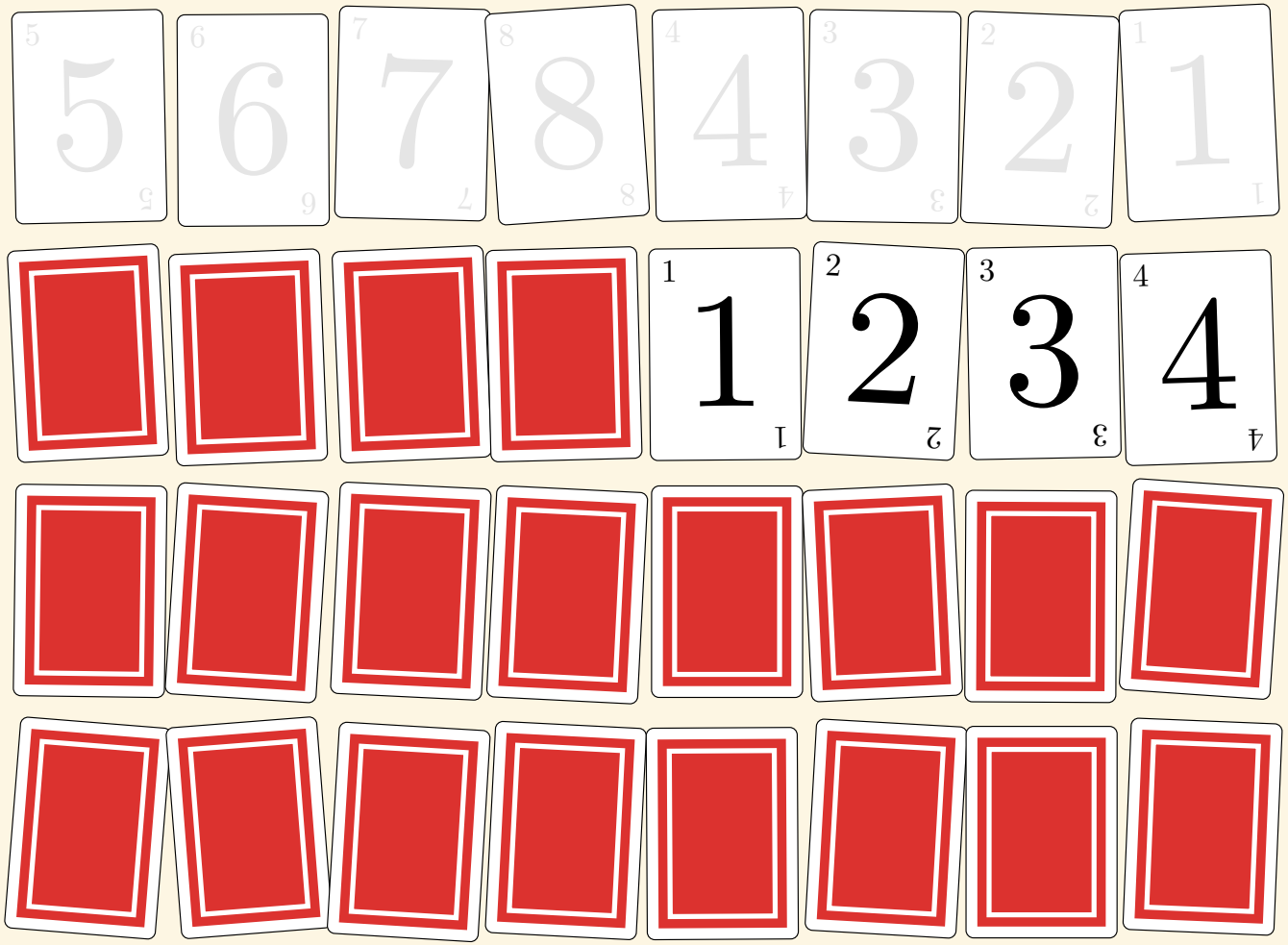
A[7 . . 8] er sortert

mlh@ntnu.no



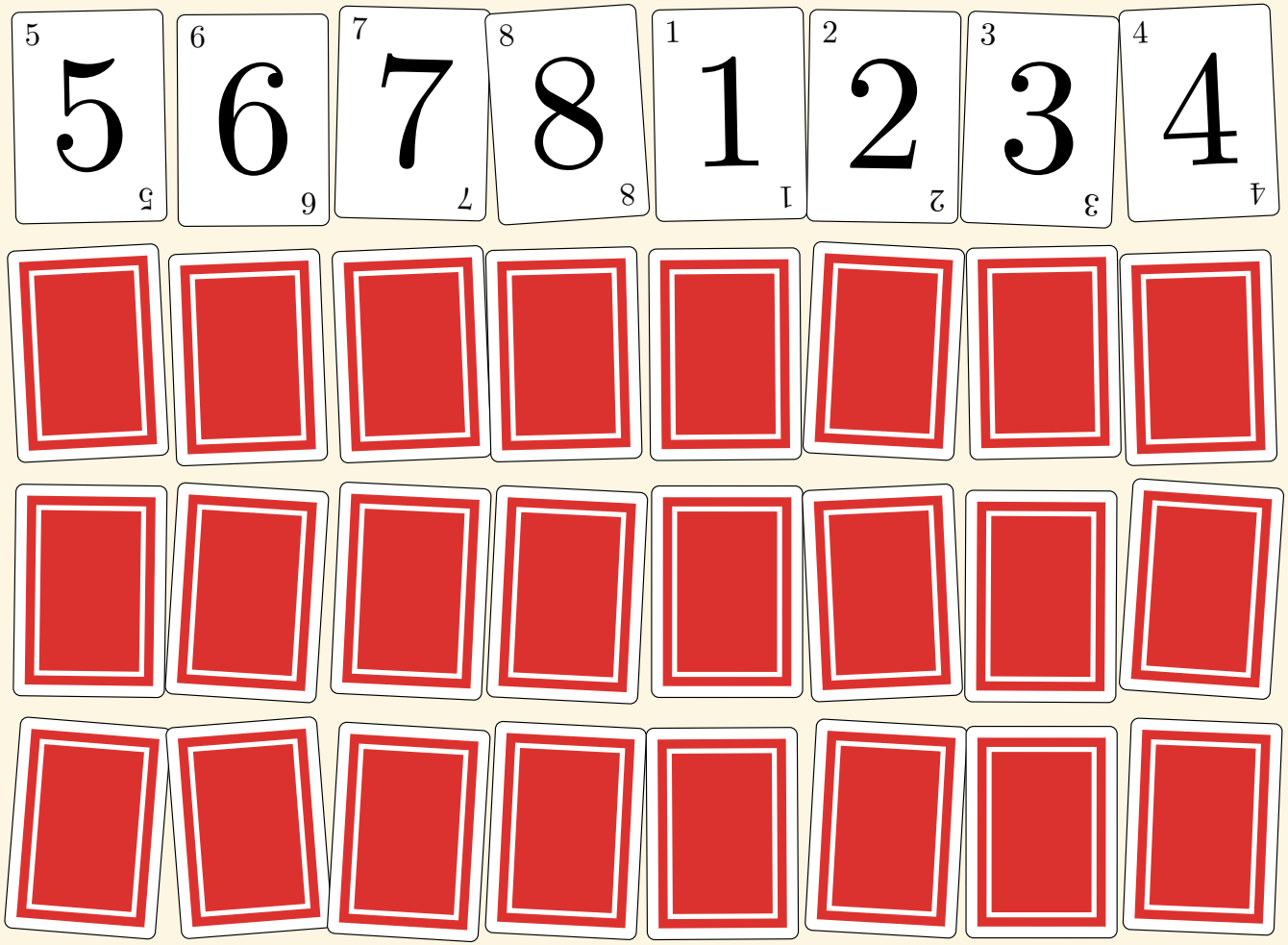
Flett sammen A[5 . . 6] og A[7 . . 8]

mlh@ntnu.no



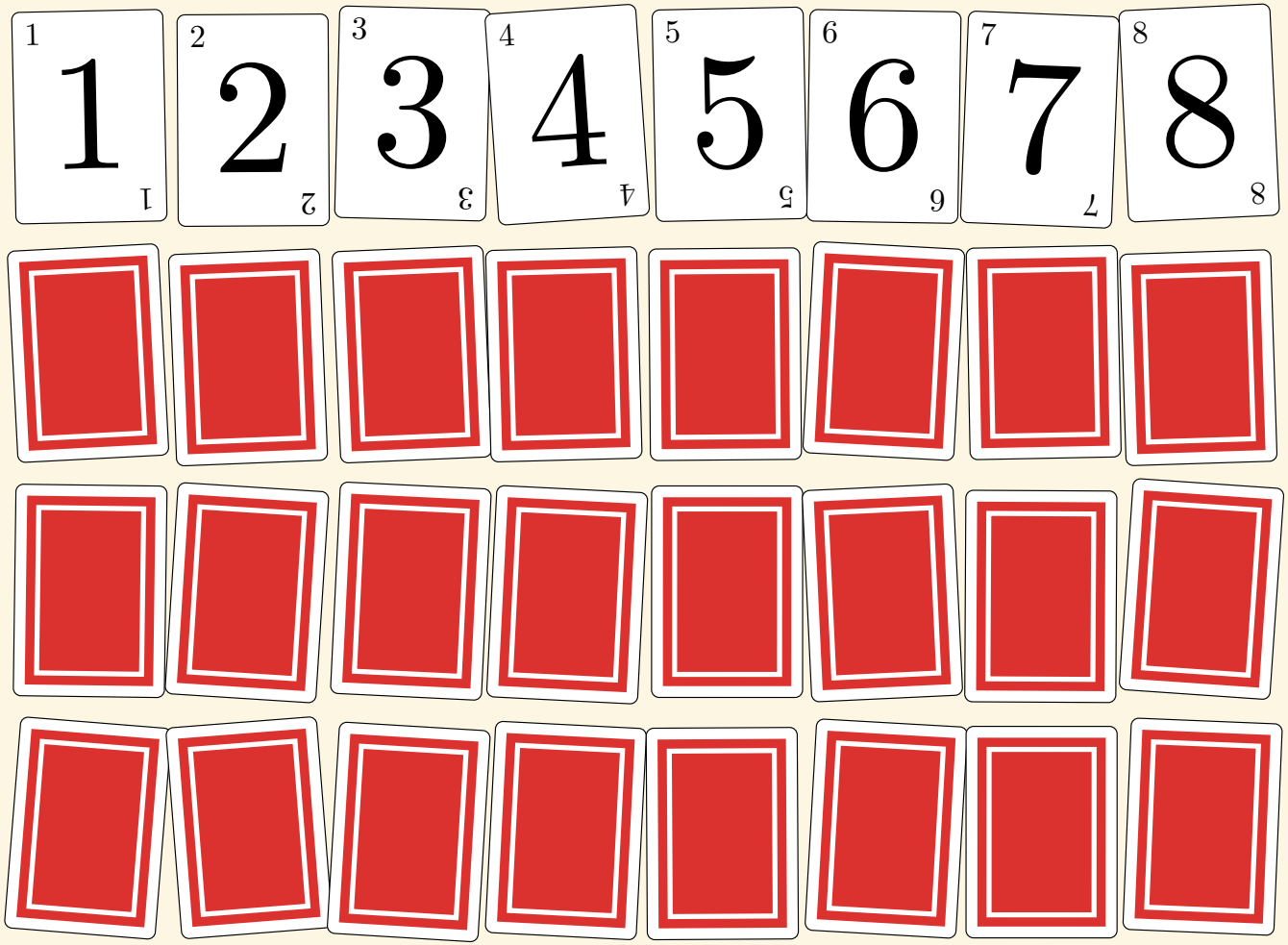
A[5 . . 8] er sortert

mlh@ntnu.no



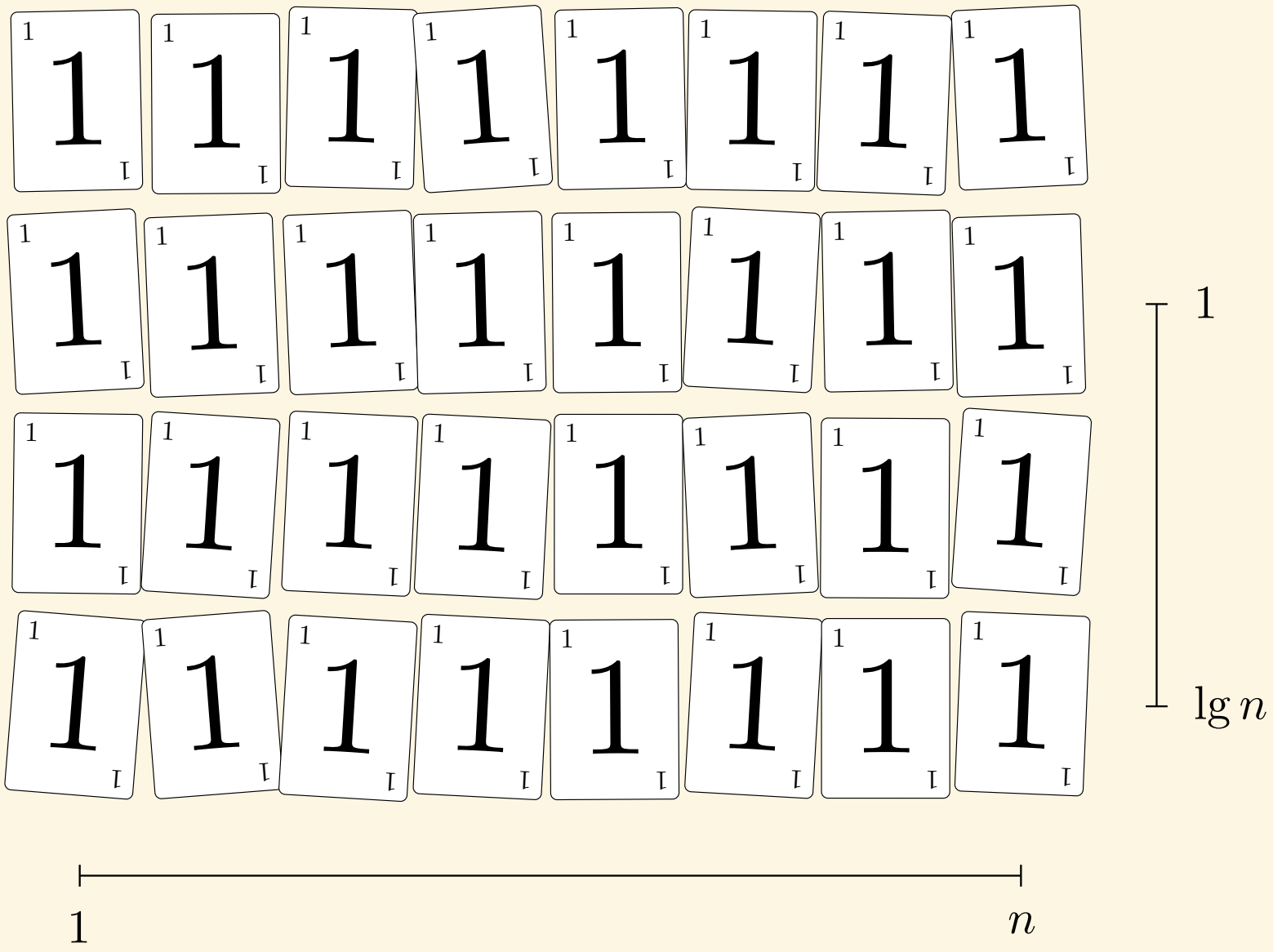
Flett sammen A[1..4] og A[5..8]

mlh@ntnu.no



A[1 . . 8] er sortert

mlh@ntnu.no



mlh@ntnu.no

MERGE-SORT(A, p, r)

A tabell
p start
r slutt
q midt

Sorter A$[p:r]$ ved å sortere halvdelene først

mlh@ntnu.no

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

A tabell
p start
r slutt
q midt

Hvis A[$p : r$] har lengde maks 1 . . .

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
```

A tabell
p start
r slutt
q midt

Grunntilfelle: Allerede sortert

mlh@ntnu.no

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

A tabell
p start
r slutt
q midt

Ellers: Del på midten

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
```

A tabell
p start
r slutt
q midt

Sorter $A[p:q]$ rekursivt

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
```

A tabell
p start
r slutt
q midt

Sorter $A[q+1:r]$ rekursivt

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

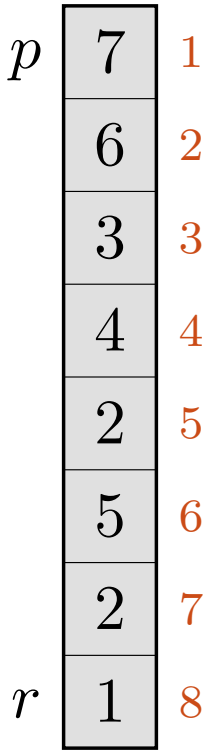
A tabell
p start
r slutt
q midt

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

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

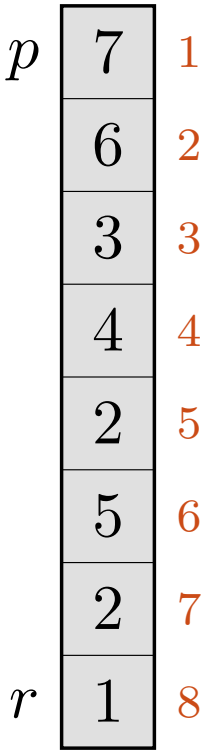
$p, r \;=\; 1, 8$



mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

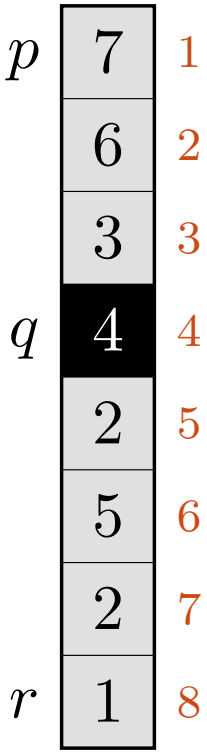
$p, r \;=\; 1, 8$



mlh@ntnu.no

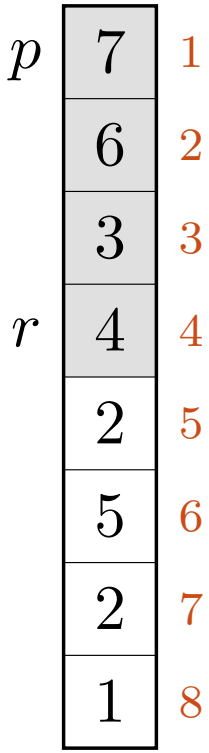
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

$p, r = 1, 8$



mlh@ntnu.no

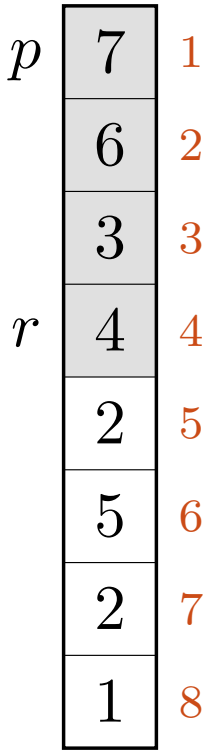
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

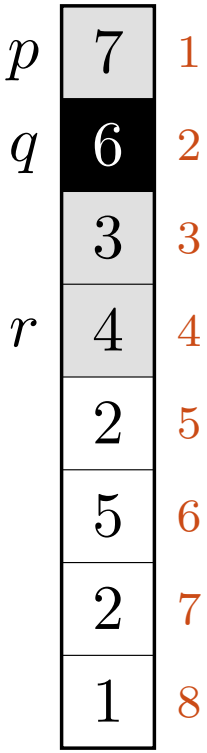
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

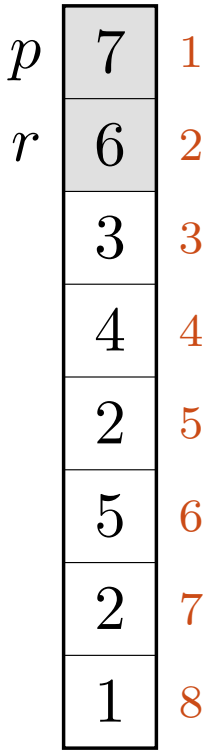
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

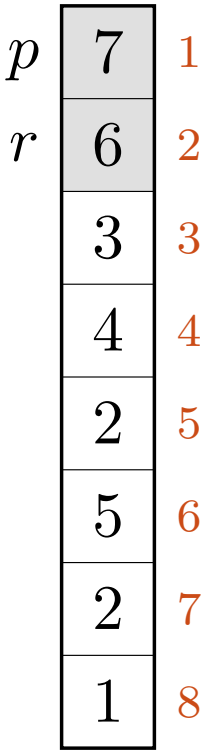


$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

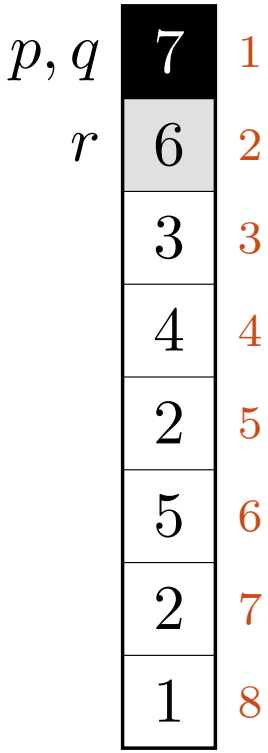
$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$



mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

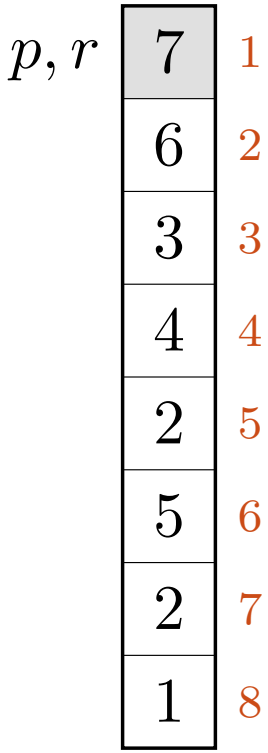
$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2$



mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

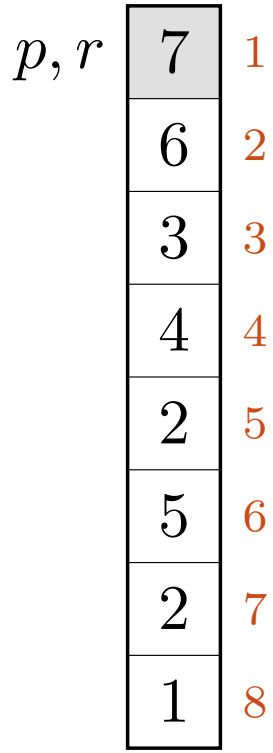
$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 1, 2 \rightarrow 1, 1$



mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

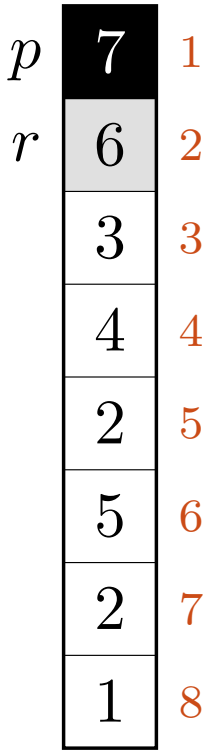
p, r = 1, 8 › 1, 4 › 1, 2 › 1, 1



mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

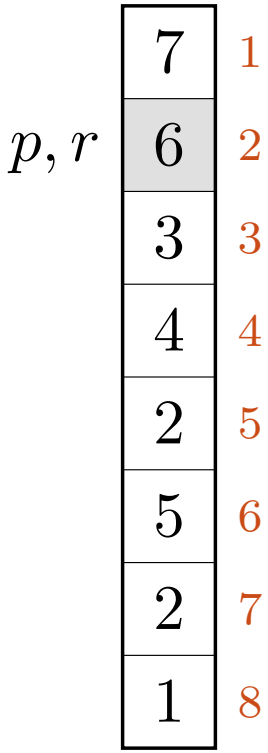
$p, r \;=\; 1, 8 \;\rightarrow\; 1, 4 \;\rightarrow\; 1, 2$



mlh@ntnu.no

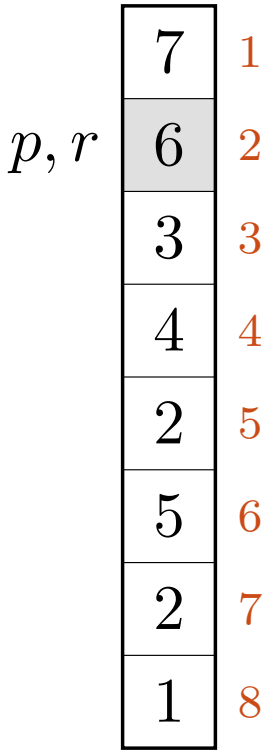
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

p, r = 1, 8 › 1, 4 › 1, 2 › 2, 2



mlh@ntnu.no

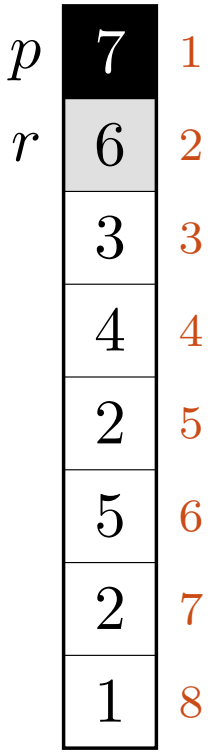
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



p, r = 1, 8 › 1, 4 › 1, 2 › 2, 2

mlh@ntnu.no

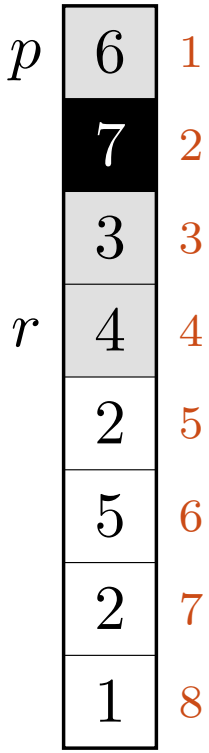
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

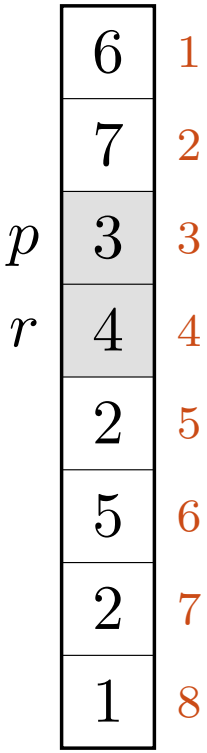
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

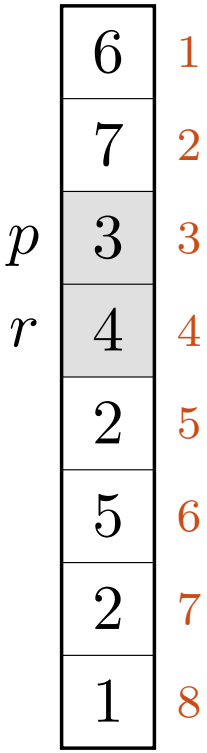
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

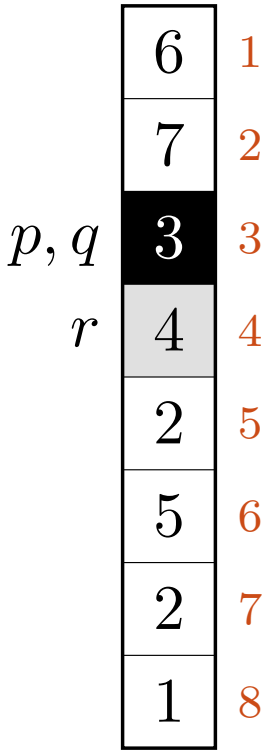


$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4$

mlh@ntnu.no

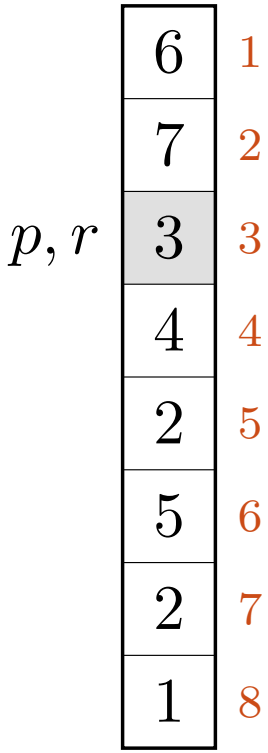
MERGE-SORT(A, p, r)
1 **if** $p \geqslant r$
2 **return**
3 $q = \lfloor (p + r)/2 \rfloor$
4 MERGE-SORT(A, p, q)
5 MERGE-SORT(A, $q + 1, r$)
6 MERGE(A, p, q, r)

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



mlh@ntnu.no

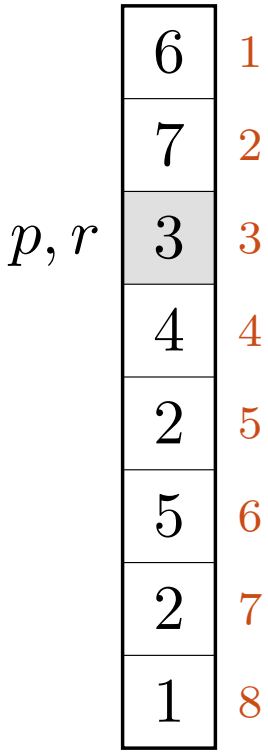
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

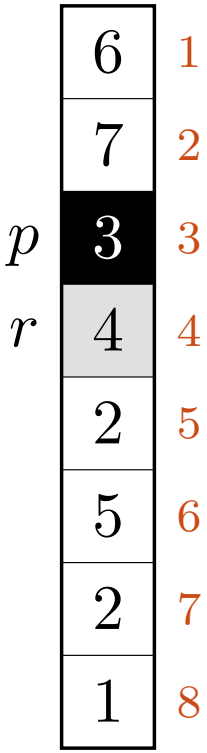
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

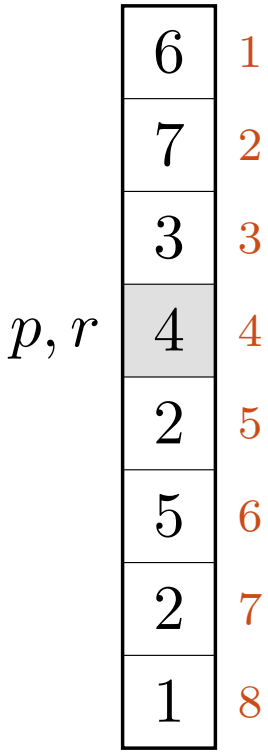
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

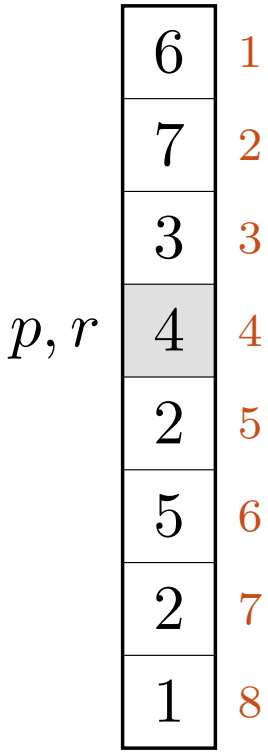


$p, r = 1, 8 \rightarrow 1, 4 \rightarrow 3, 4 \rightarrow 4, 4$

mlh@ntnu.no

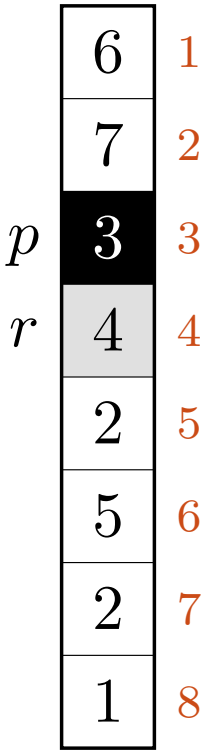
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

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



mlh@ntnu.no

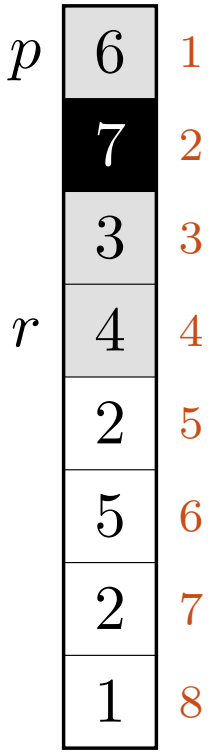
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

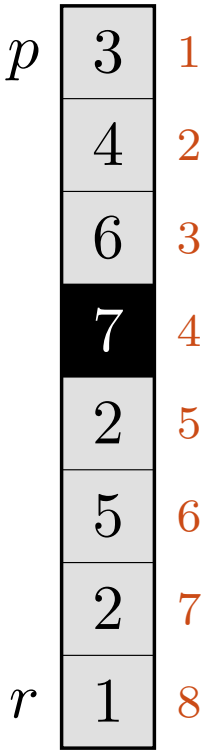


$p, r = 1, 8 \rightarrow 1, 4$

mlh@ntnu.no

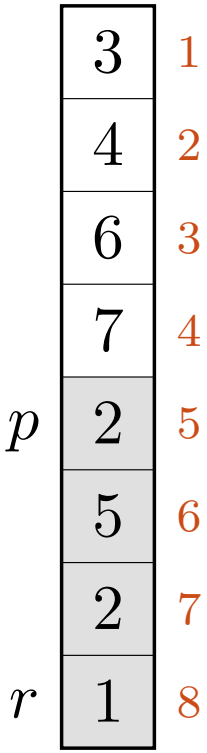
MERGE-SORT(A, p, r)
1 **if** $p \geqslant r$
2 **return**
3 $q = \lfloor (p + r)/2 \rfloor$
4 MERGE-SORT(A, p, q)
5 MERGE-SORT(A, $q + 1, r$)
6 MERGE(A, p, q, r)

$p, r = 1, 8$



mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

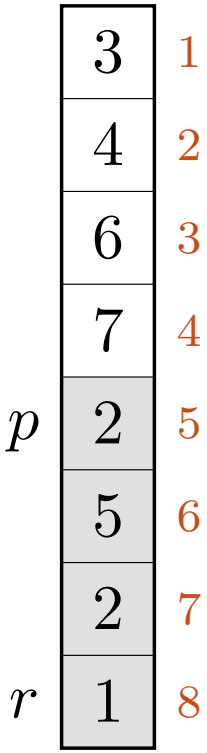


$p, r = 1, 8 \rightarrow 5, 8$

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

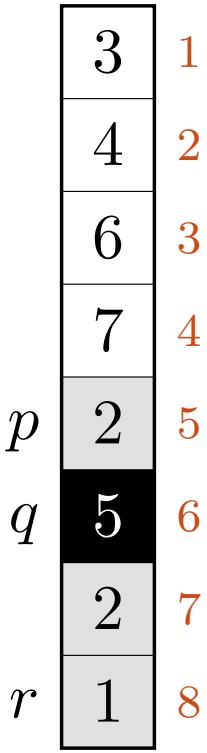
$p, r = 1, 8 \rightarrow 5, 8$



mlh@ntnu.no

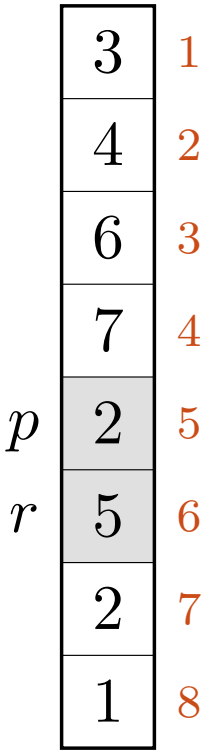
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

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



mlh@ntnu.no

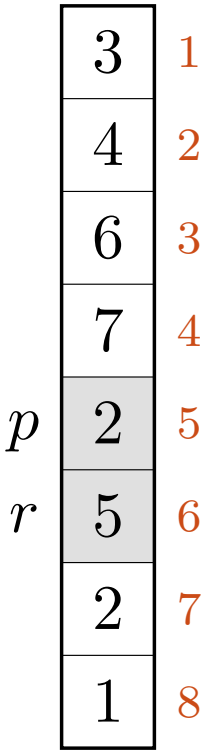
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

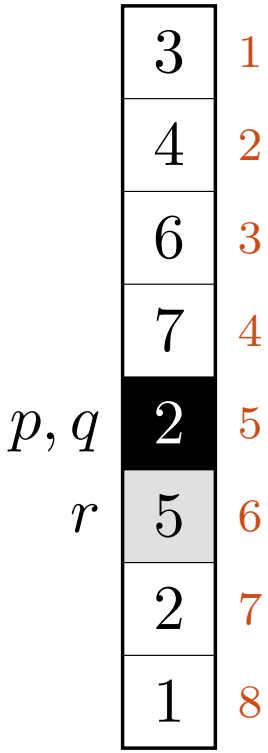
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

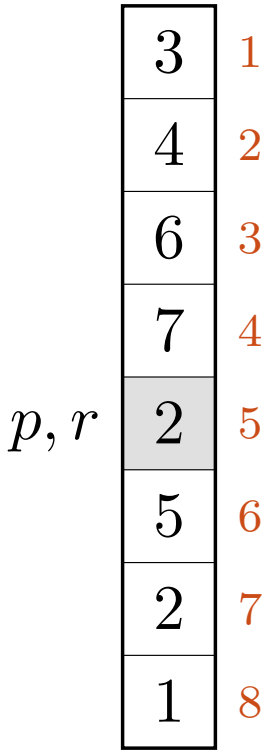
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

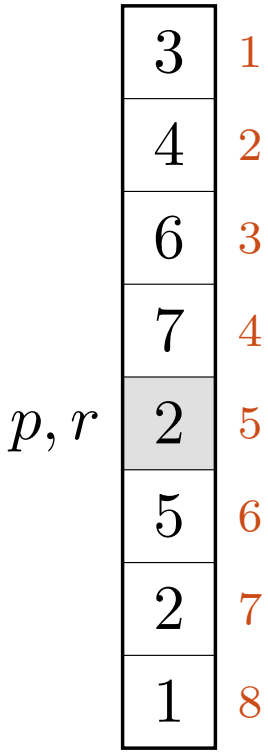
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

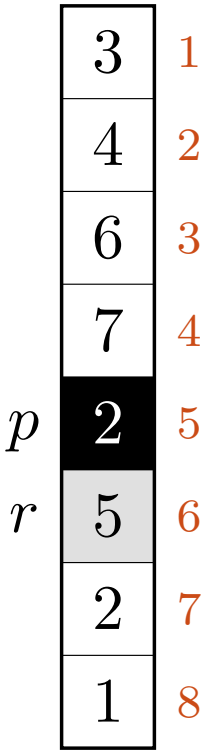
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

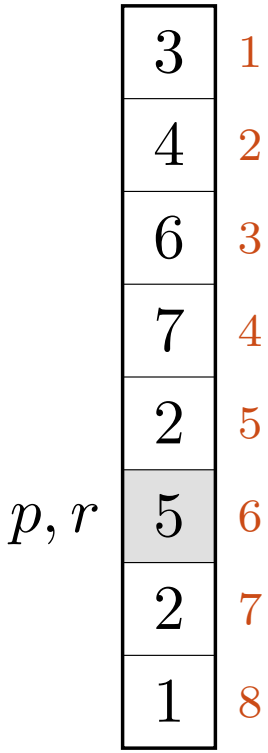
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

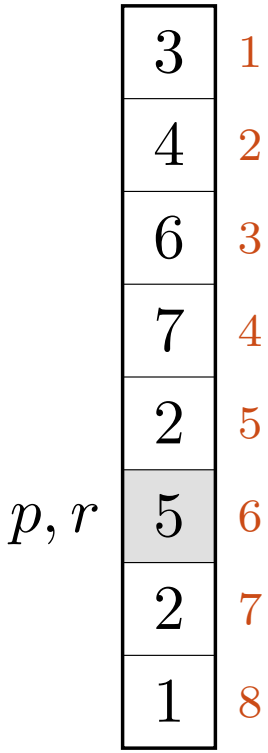
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

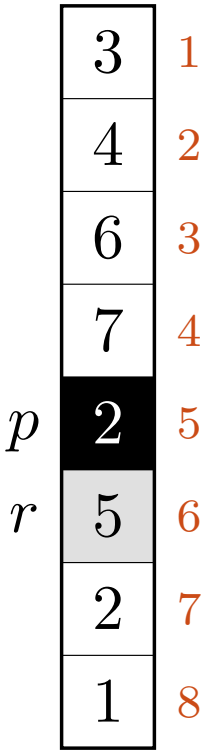
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

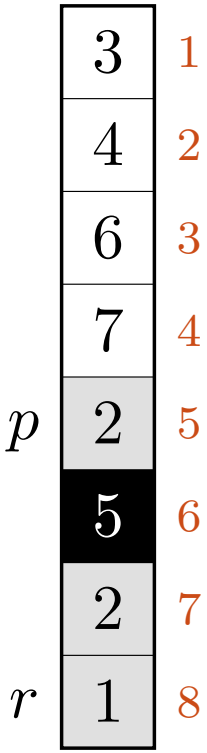


$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 5, 6$

mlh@ntnu.no

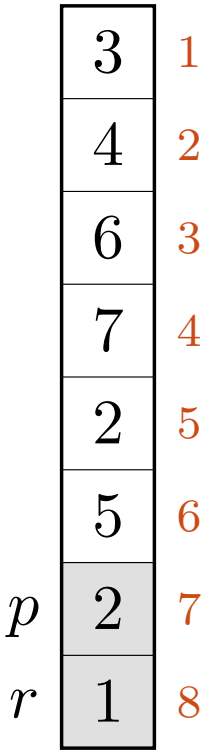
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```

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



mlh@ntnu.no

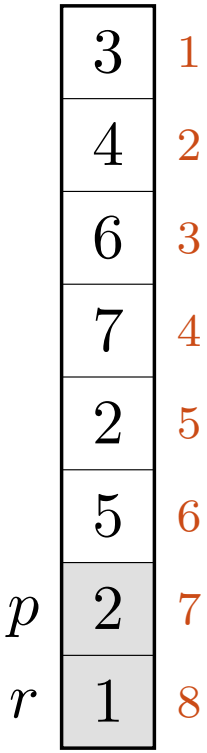
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

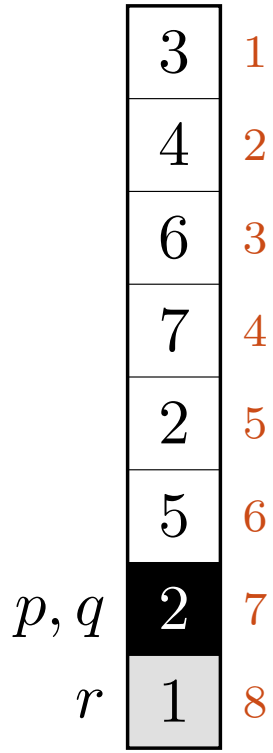
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



$p, r \;=\; 1, 8 \;\rangle\; 5, 8 \;\rangle\; 7, 8$

mlh@ntnu.no

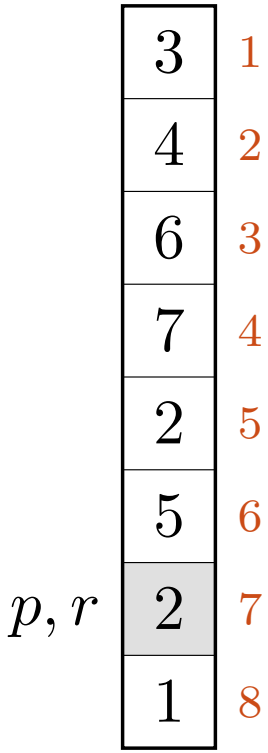
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

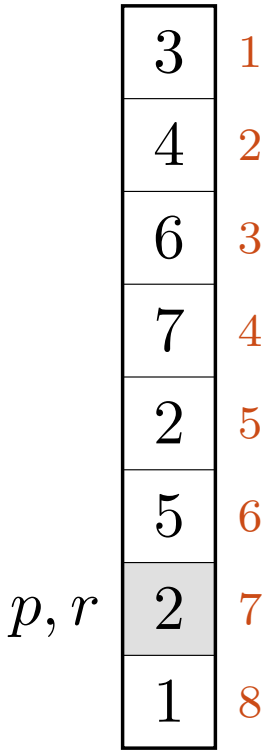
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



$p, r = 1, 8 \rightarrow 5, 8 \rightarrow 7, 8 \rightarrow \mathbf{7, 7}$

mlh@ntnu.no

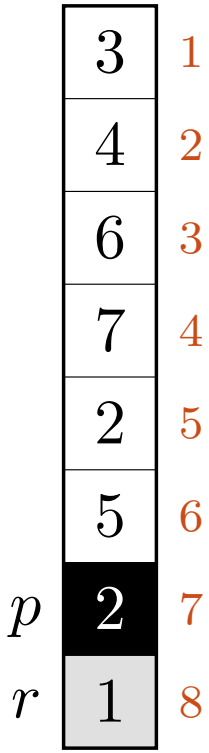
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

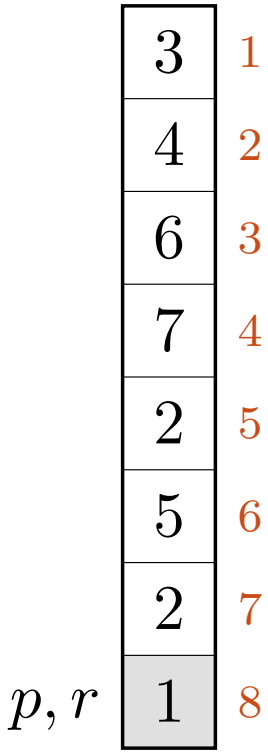
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

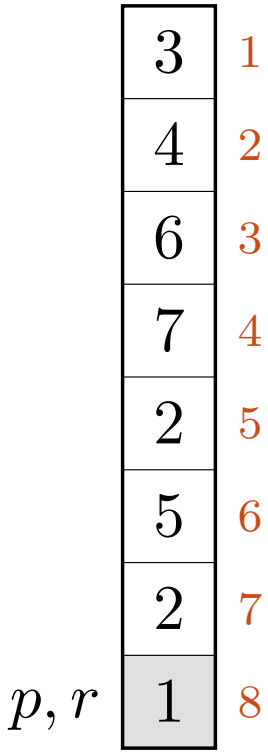
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

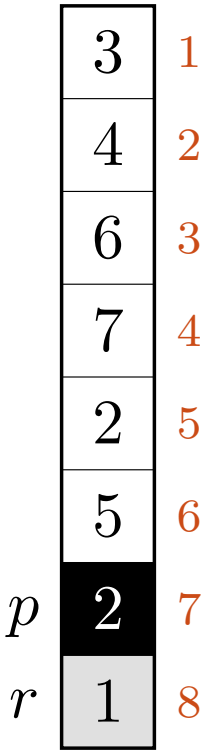
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

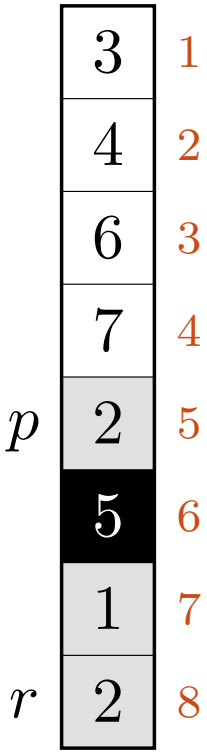
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

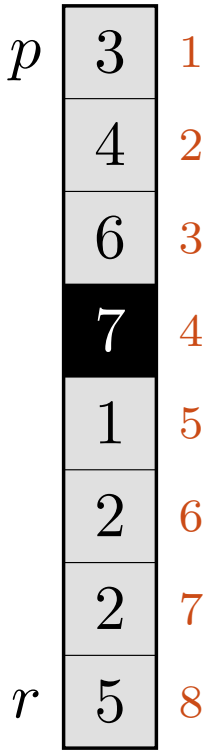
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



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

mlh@ntnu.no

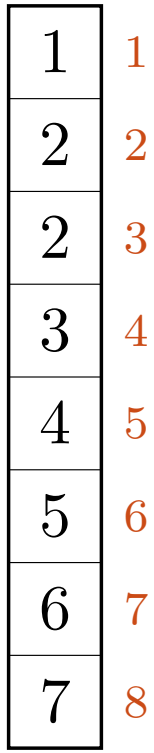
```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



$p, r = 1, 8$

mlh@ntnu.no

```
MERGE-SORT(A, p, r)
1  if p ⩾ r
2        return
3  q = ⌊(p + r)/2⌋
4  MERGE-SORT(A, p, q)
5  MERGE-SORT(A, q + 1, r)
6  MERGE(A, p, q, r)
```



mlh@ntnu.no

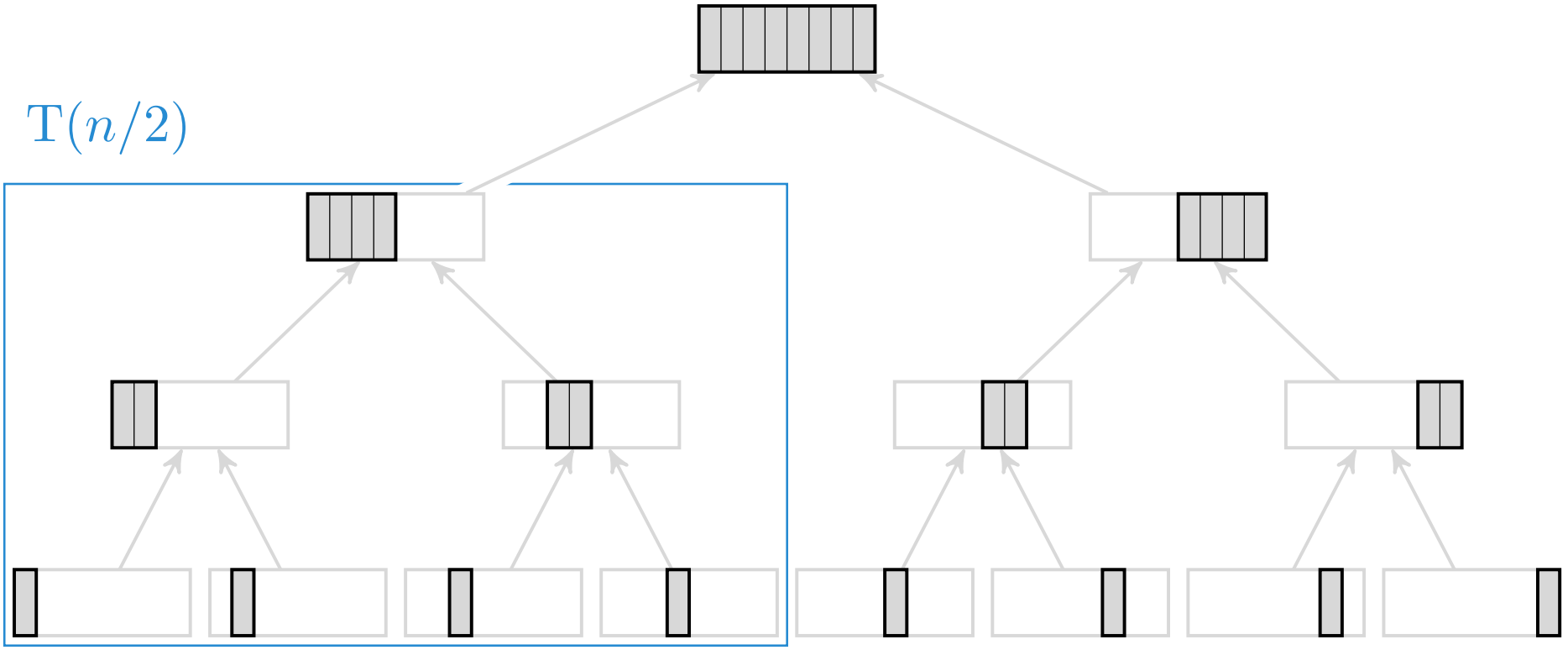
$$\mathrm{T}(n) = 2\mathrm{T}(n/2) + n$$

mlh@ntnu.no

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

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

mlh@ntnu.no



mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

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

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

mlh@ntnu.no

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

$$+ n \tag{1}$$

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

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

mlh@ntnu.no

Verifikasjon

Med substitusjon/induksjon

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

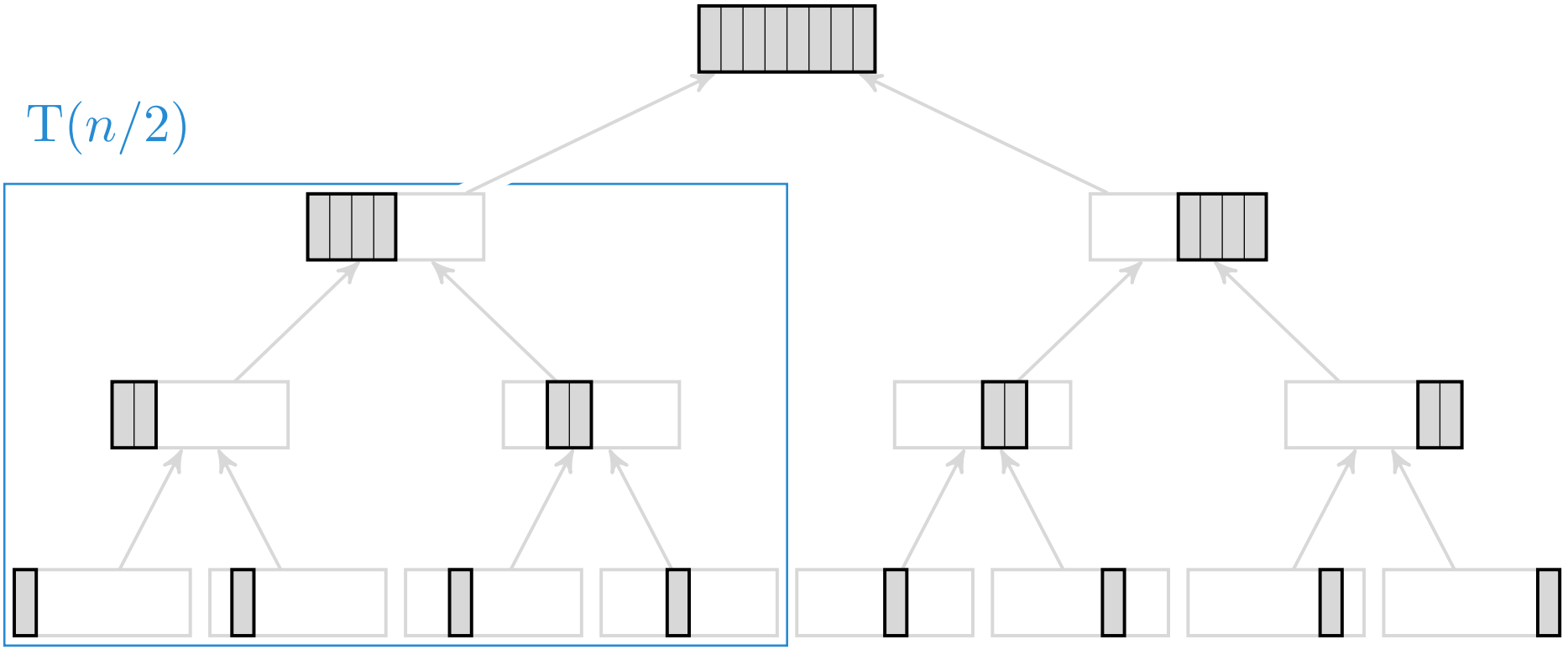
Gitt $\mathrm{T}(k) = k\lg k + k$ for $k < n$, vis $\mathrm{T}(n) = n\lg n + n$

mlh@ntnu.no

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

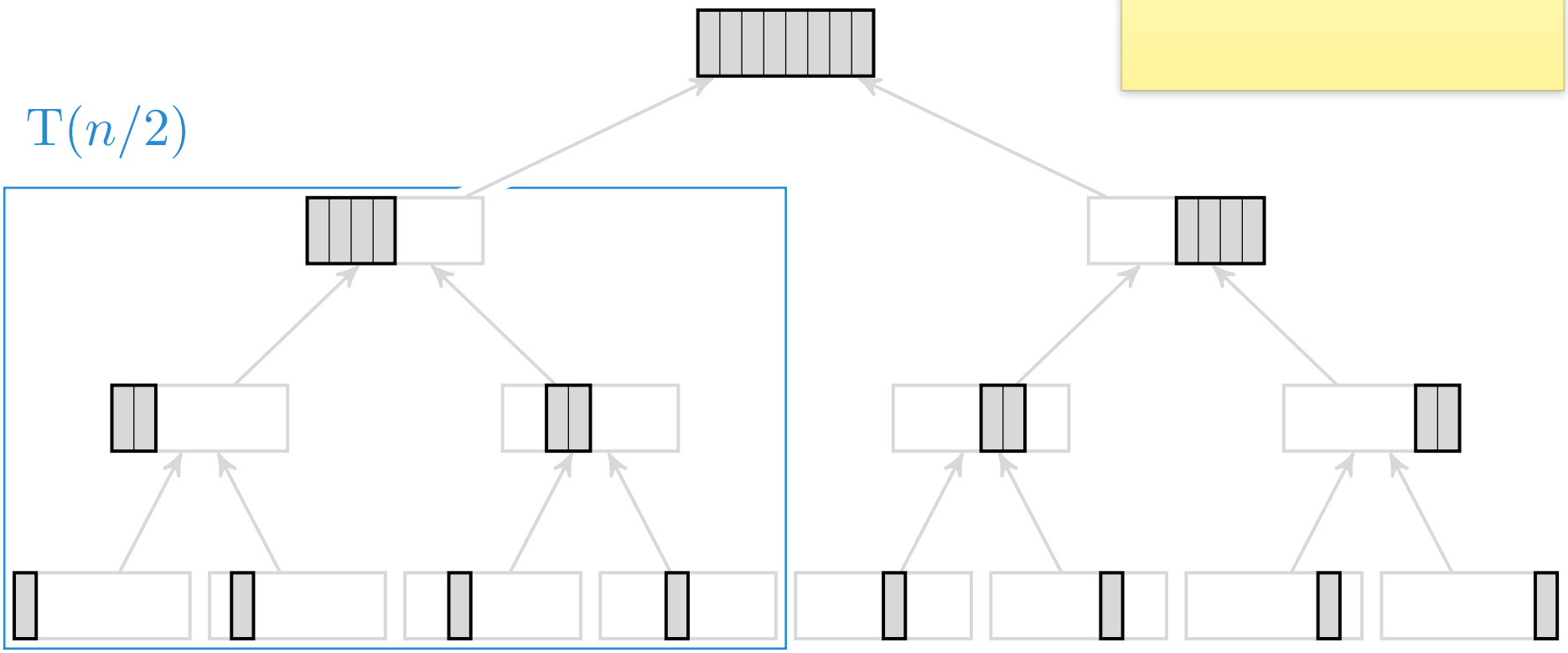
Gitt $\mathrm{T}(k) = k\lg k + k$ for $k < n$, vis $\mathrm{T}(n) = n\lg n + n$

mlh@ntnu.no



mlh@ntnu.no

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

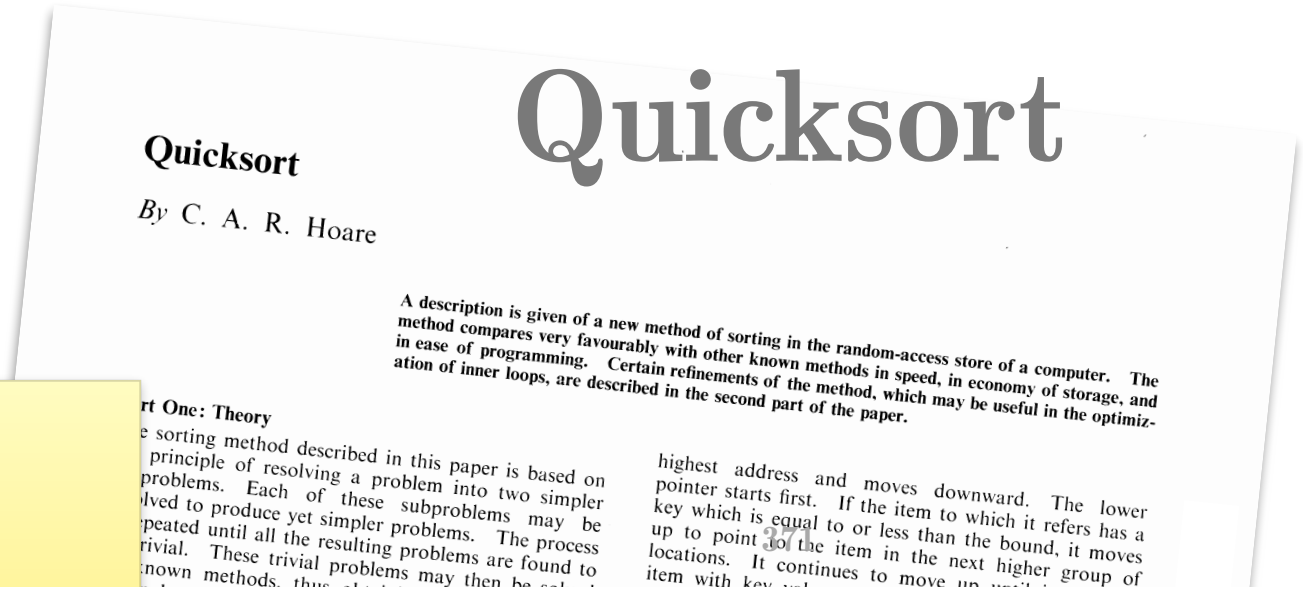


$$\mathrm{T}(n) = n \lg n + n$$

mlh@ntnu.no

4:5

Quicksort



Quicksort

By C. A. R. Hoare

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

rt One: Theory

e sorting method described in this paper is based on principle of resolving a problem into two simpler problems. Each of these subproblems may be lved to produce yet simpler problems. The process peated until all the resulting problems are found to rivial. These trivial problems may then be solved nown methods, thus obtaining a 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 reaches an item with key value

Fra 1962

Instans	Løsning
Dekomponering	Kombinasjon IS
Delinstanser	Delløsninger IH
Grunntilfelle	Grunnløsning

mlh@ntnu.no

Instans	Løsning
Dekomponering	Kombinasjon IS
Delinstanser	Delløsninger IH
Grunntilfelle ...	Grunnløsning ...

Hva er input?

mlh@ntnu.no

Instans En sekvens A	Løsning
Dekomponering	Kombinasjon IS
Delinstanser	Delløsninger IH
Grunntilfelle ...	Grunnløsning ...

Hva er input?

mlh@ntnu.no

Instans En sekvens A	Løsning
Dekomponering	Kombinasjon IS
Delinstanser	Delløsninger IH
Grunntilfelle ...	Grunnløsning ...

Hva ønsker vi at output skal være?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Delinstanser	Delløsninger IH
Grunntilfelle ...	Grunnløsning ...

Hva ønsker vi at output skal være?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Delinstanser ...	Delløsninger IH ...
Grunntilfelle	Grunnløsning

Hvordan kan vi krympe eller dele opp instansen?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	Delløsninger IH
...	...
Grunntilfelle	Grunnløsning

Hvordan kan vi krympe eller dele opp instansen?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	Delløsninger IH
...	...
Grunntilfelle	Grunnløsning

Hva har vi brutt problemet ned til?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	Delløsninger IH
Liten og stor del, til venstre og høyre	
Grunntilfelle	Grunnløsning

Hva har vi brutt problemet ned til?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	Delløsninger IH
Liten og stor del, til venstre og høyre	
Grunntilfelle	Grunnløsning

Hvor stopper dekomponeringen?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	**Kombinasjon** IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	**Delløsninger** IH
Liten og stor del, til venstre og høyre	
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	

Hvor stopper dekomponeringen?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	Delløsninger IH
Liten og stor del, til venstre og høyre	
Grunntilfelle	Grunnløsning
Sekvens med maks ett element	

Hva er løsningen da?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	**Kombinasjon** IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	**Delløsninger** IH
Liten og stor del, til venstre og høyre	
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	Samme sekvens, allerede sortert

Hva er løsningen da?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomposering	**Kombinasjon** IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	**Delløsninger** IH
Liten og stor del, til venstre og høyre	
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	Samme sekvens, allerede sortert

Hva er løsningene for delinstansene?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomposering	**Kombinasjon** IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	**Delløsninger** IH
Liten og stor del, til venstre og høyre	Liten og stor del, til venstre og høyre, sorterte
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	Samme sekvens, allerede sortert

Hva er løsningene for delinstansene?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	**Kombinasjon** IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	**Delløsninger** IH
Liten og stor del, til venstre og høyre	Liten og stor del, til venstre og høyre, sorterte
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	Samme sekvens, allerede sortert

Hvordan bygger vi en full løsning av delløsninger?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	**Kombinasjon** IS
Fordel i to deler, venstre verdier mindre enn høyre	↑
Delinstanser	**Delløsninger** IH
Liten og stor del, til venstre og høyre	Liten og stor del, til venstre og høyre, sorterte
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	Samme sekvens, allerede sortert

Hvordan bygger vi en full løsning av delløsninger?

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	**Kombinasjon** IS
Fordel i to deler, venstre verdier mindre enn høyre	↑
Delinstanser	**Delløsninger** IH
Liten og stor del, til venstre og høyre	Liten og stor del, til venstre og høyre, sorterte
Grunntilfelle	**Grunnløsning**
Sekvens med maks ett element	Samme sekvens, allerede sortert

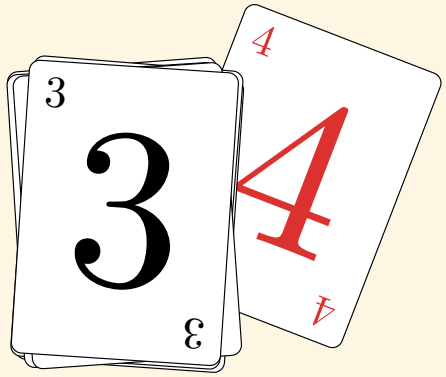
mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	Delløsninger IH
Liten og stor del, til venstre og høyre	Liten og stor del, til venstre og høyre, sorterte
Grunntilfelle	Grunnløsning
Sekvens med maks ett element	Samme sekvens, allerede sortert

mlh@ntnu.no

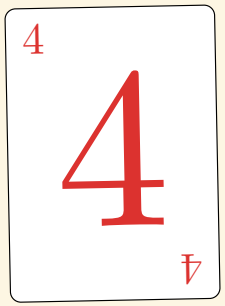
Vi skal dele i «små» og «store»

mlh@ntnu.no



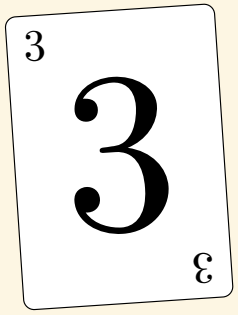
Vi plukker vilkårlig ut siste kort som splittelement eller «pivot»

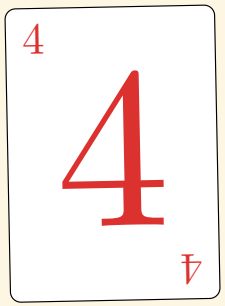
mlh@ntnu.no



«Små» defineres som mindre eller lik pivot; «store» er større

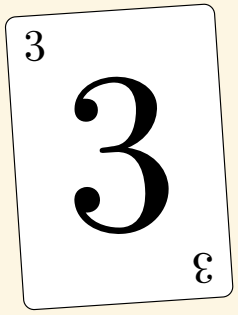
mlh@ntnu.no

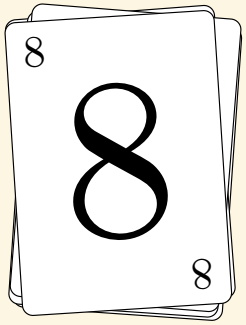


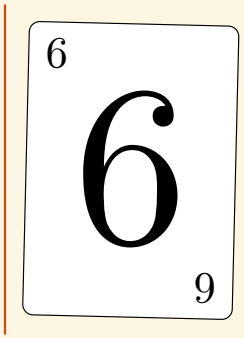


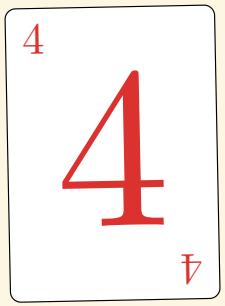
$3 \leqslant 4$, så 3 er liten og plasseres til venstre

mlh@ntnu.no



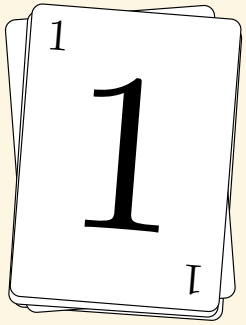


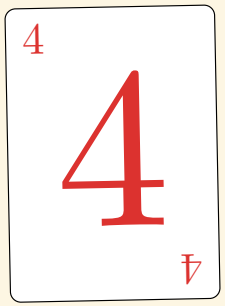


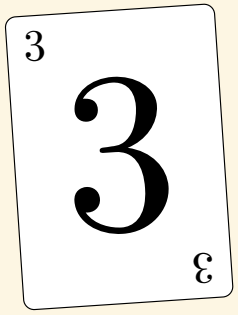


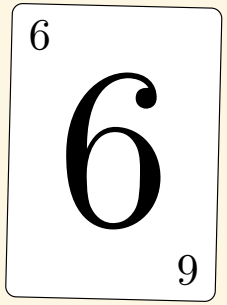
$6 > 4$, så 6 er stor og plasseres til høyre

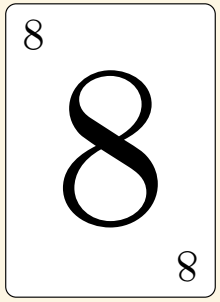
mlh@ntnu.no





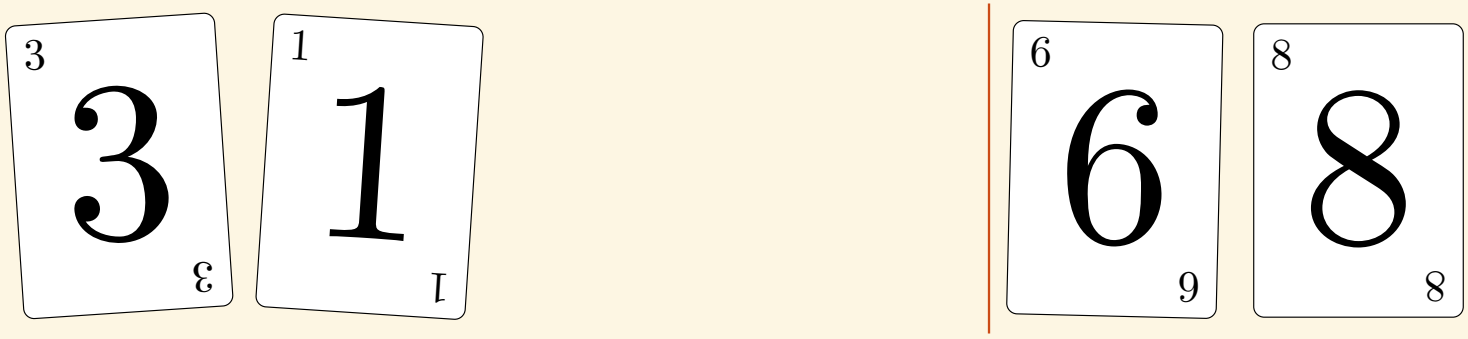


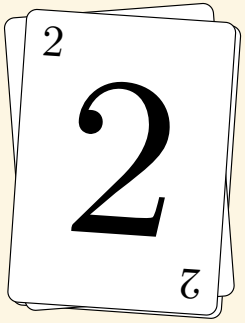


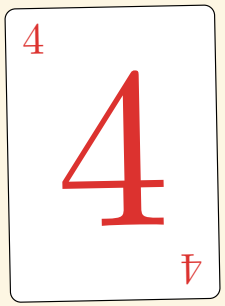


$8 > 4$, så 8 er stor og plasseres til høyre

mlh@ntnu.no

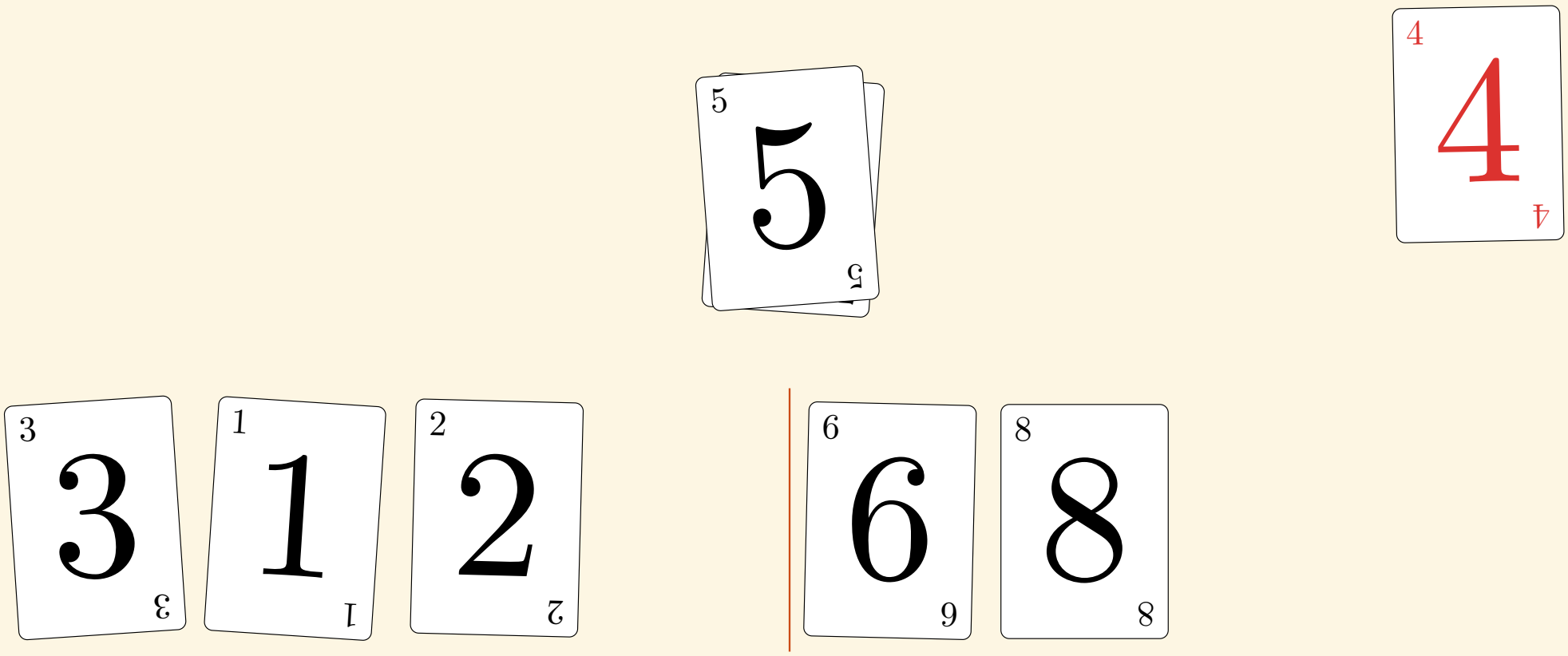






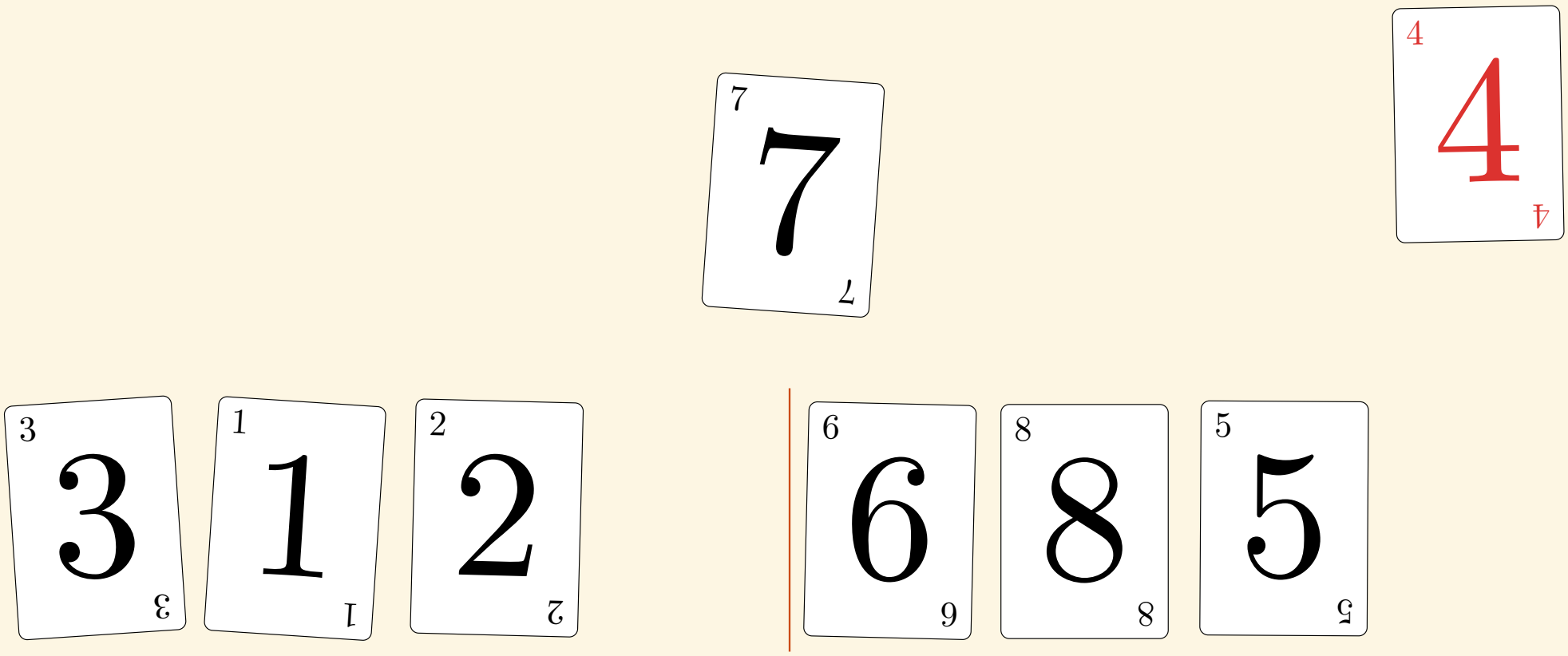
$1 \leqslant 4$, så 1 er liten og plasseres til venstre

mlh@ntnu.no



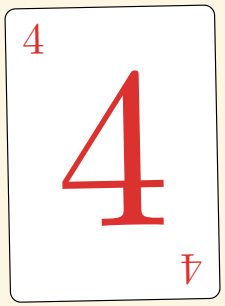
$2 \leqslant 4$, så 2 er liten og plasseres til venstre

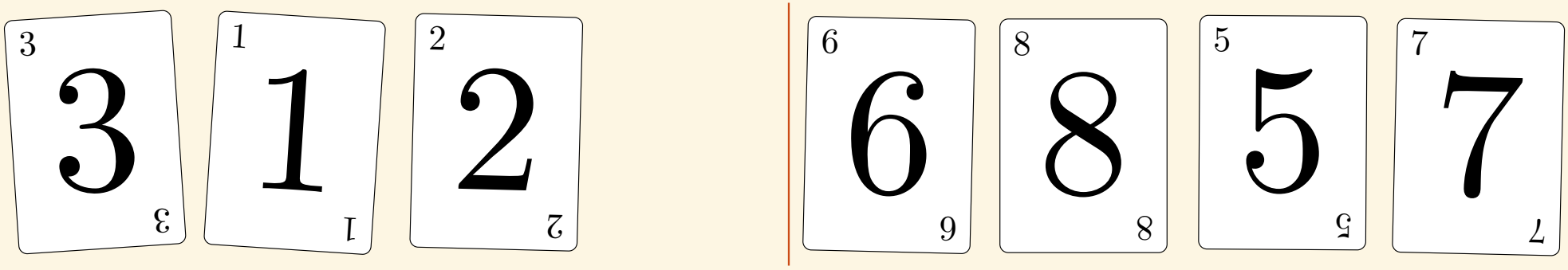
mlh@ntnu.no



5 > 4, så 5 er stor og plasseres til høyre

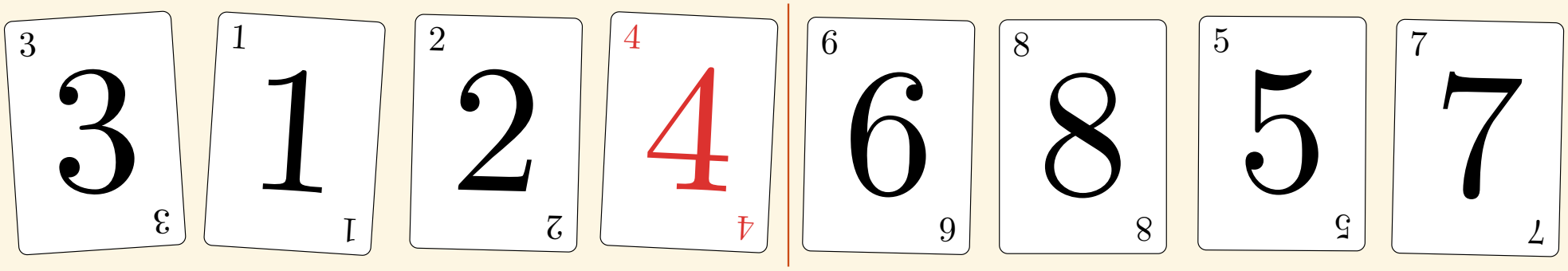
mlh@ntnu.no





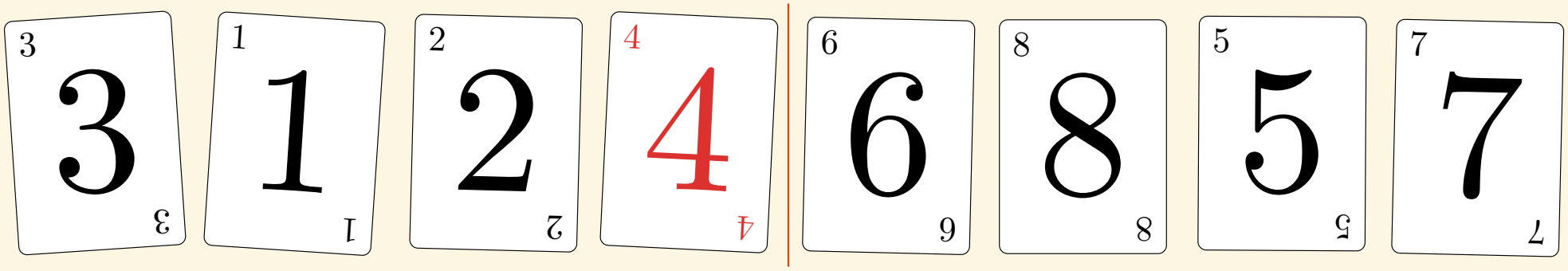
$7 > 4$, så 7 er stor og plasseres til høyre

mlh@ntnu.no



Pivoten plasseres mellom de to segmentene

mlh@ntnu.no



(Merk: Pivoten er nå på riktig plass, og flyttes ikke igjen!)

mlh@ntnu.no

PARTITION(A, p, r)

A tabell
p venstre
r høyre

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

mlh@ntnu.no

PARTITION(A, p, r)
1 $x = \mathrm{A}[r]$

A tabell
p venstre
r høyre
x splitt

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

mlh@ntnu.no

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

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

Siste indeks i segmentet med små verdier. Foreløpig tomt

mlh@ntnu.no

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

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

Siste indeks i segmentet med store verdier

mlh@ntnu.no

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

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

mlh@ntnu.no

```
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
```

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

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

mlh@ntnu.no

```
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]
```

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

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

mlh@ntnu.no

```
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]
```

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

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

mlh@ntnu.no

```
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
```

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

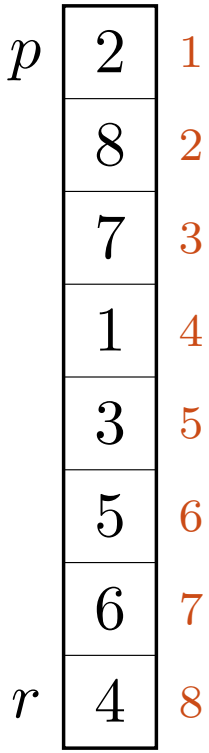
Returner hvor splitten havnet

mlh@ntnu.no

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
```

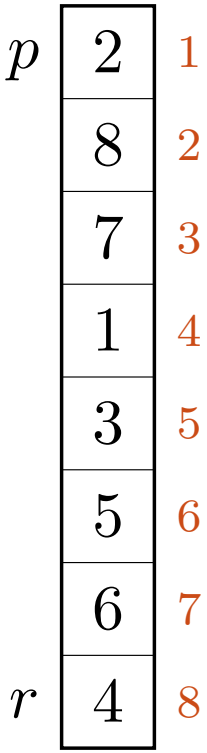
x, i = –, –



mlh@ntnu.no

```
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
```

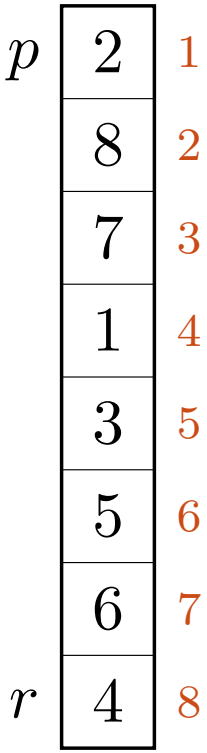
$x, i \;=\; 4, -$



mlh@ntnu.no

```
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
```

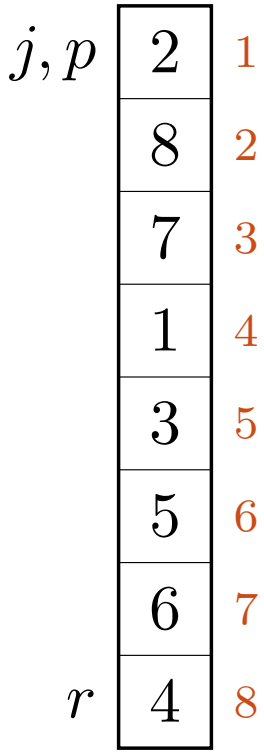
$x, i \;=\; 4, 0$



mlh@ntnu.no

```
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
```

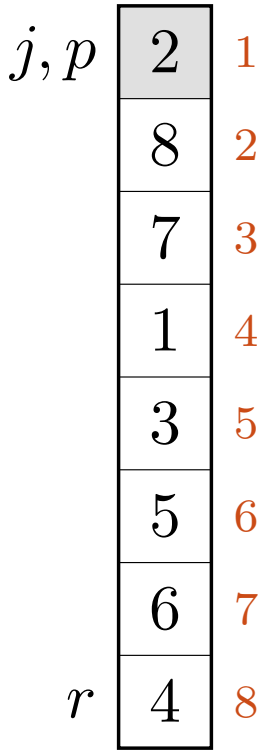
$x, i \;=\; 4, 0$



mlh@ntnu.no

```
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
```

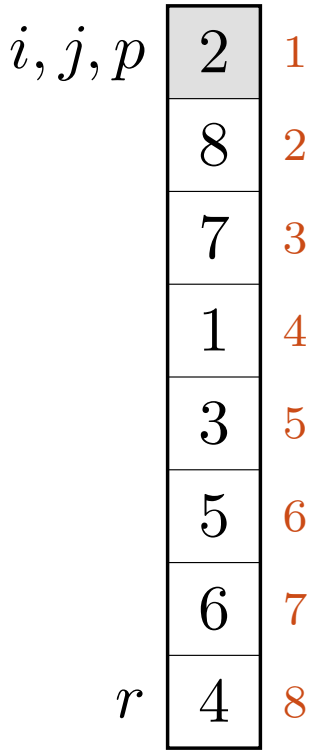
$x, i \;=\; 4, 0$



mlh@ntnu.no

```
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
```

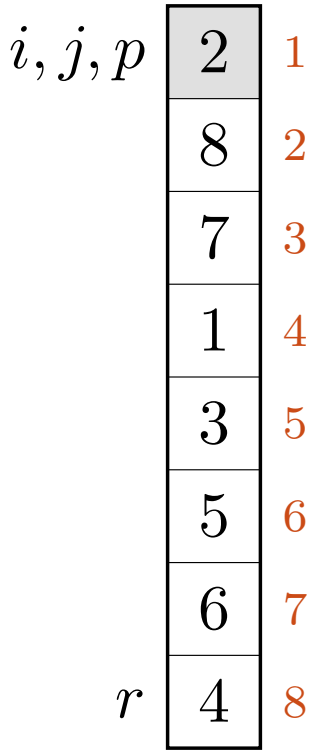
$x, i \;=\; 4, 1$



mlh@ntnu.no

```
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
```

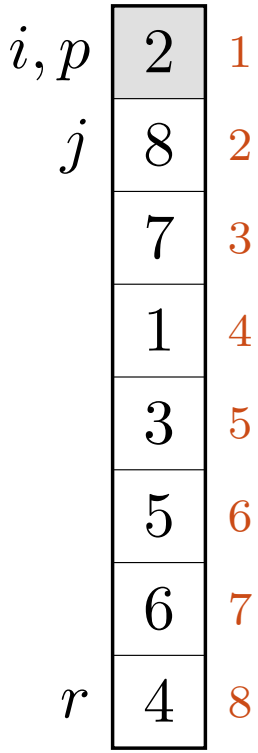
$x, i \;=\; 4, 1$



mlh@ntnu.no

```
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
```

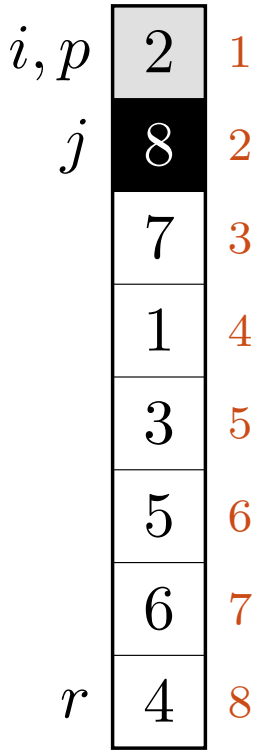
$x, i \;=\; 4, 1$



mlh@ntnu.no

```
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
```

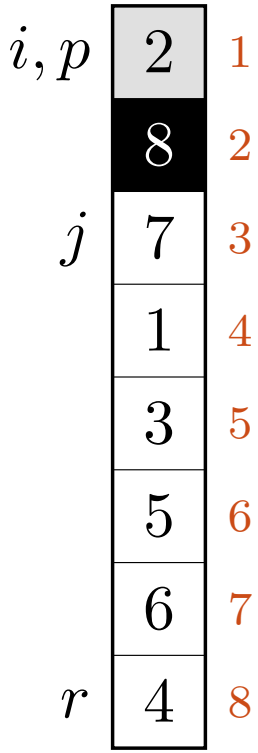
$x, i \;=\; 4, 1$



mlh@ntnu.no

```
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
```

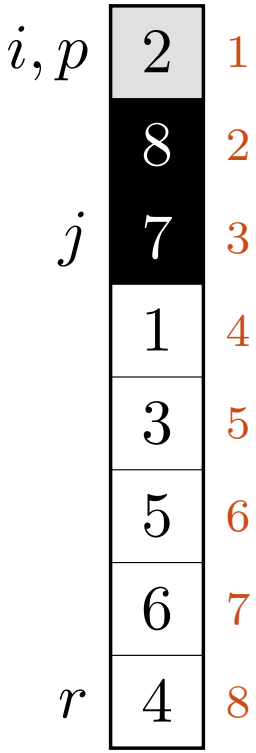
$x, i = 4, 1$



mlh@ntnu.no

```
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
```

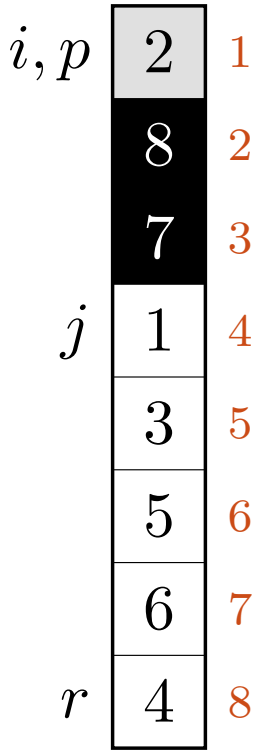
$x, i \;=\; 4, 1$



mlh@ntnu.no

```
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
```

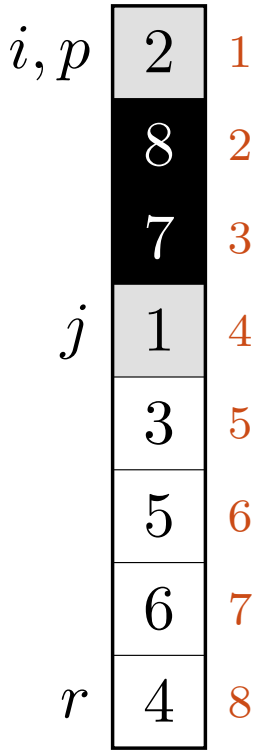
$x, i = 4, 1$



mlh@ntnu.no

```
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
```

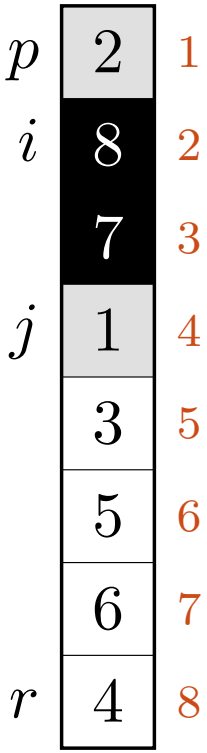
$x, i = 4, 1$



mlh@ntnu.no

```
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
```

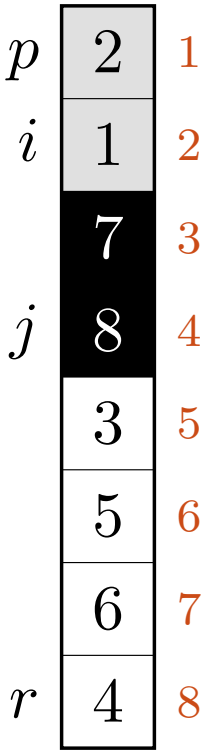
$x, i \;=\; 4, 2$



mlh@ntnu.no

```
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
```

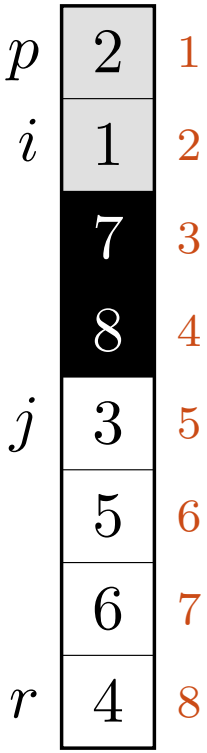
$x, i \;=\; 4, 2$



mlh@ntnu.no

```
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
```

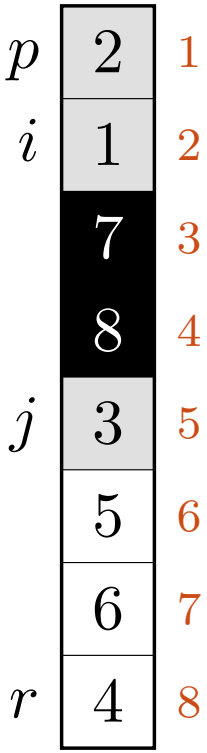
$x, i \;=\; 4, 2$



mlh@ntnu.no

```
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
```

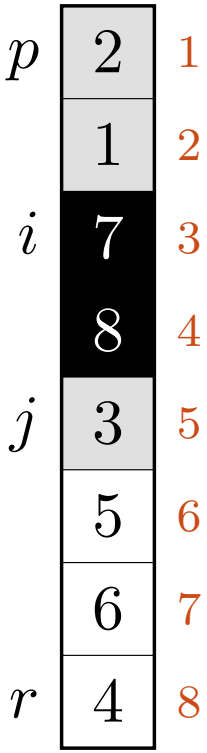
$x, i \;=\; 4, 2$



mlh@ntnu.no

```
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
```

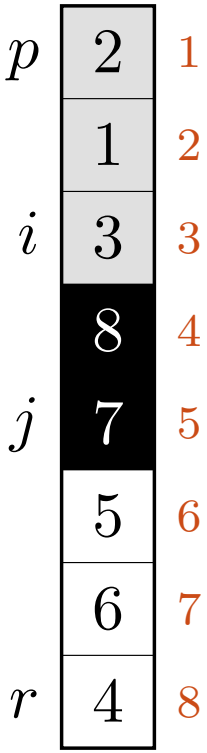
$x, i \;=\; 4, 3$



mlh@ntnu.no

```
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
```

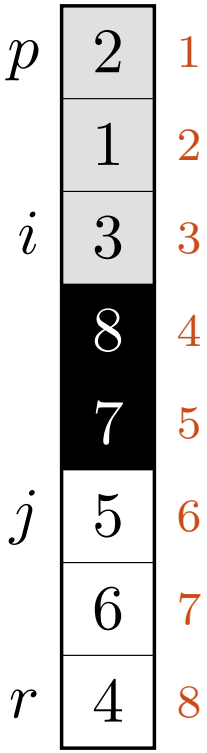
$x, i \;=\; 4, 3$



mlh@ntnu.no

```
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
```

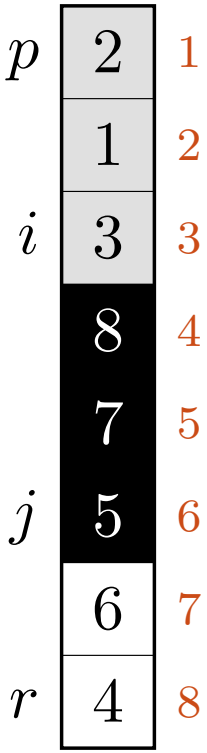
$x, i \;=\; 4, 3$



mlh@ntnu.no

```
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
```

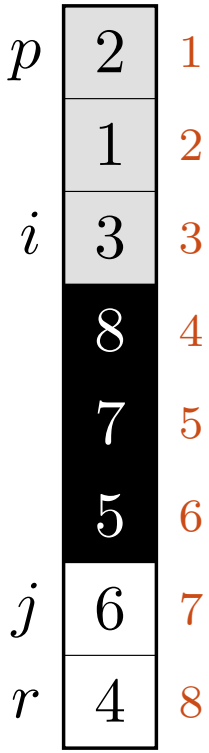
$x, i \;=\; 4, 3$



mlh@ntnu.no

```
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
```

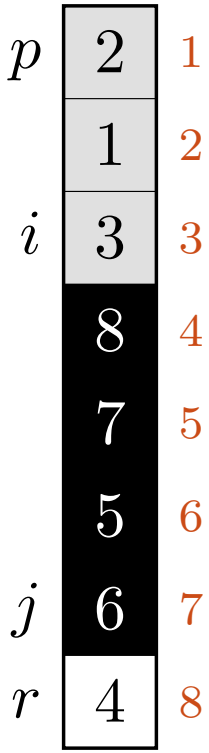
$x, i = 4, 3$



mlh@ntnu.no

```
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
```

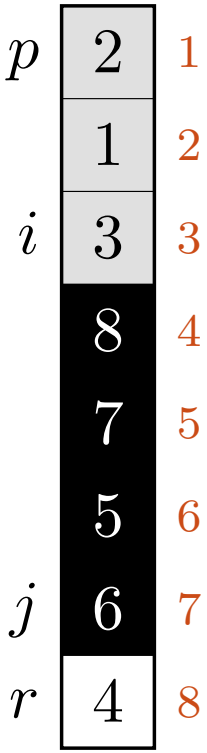
$x, i \;=\; 4, 3$



mlh@ntnu.no

```
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
```

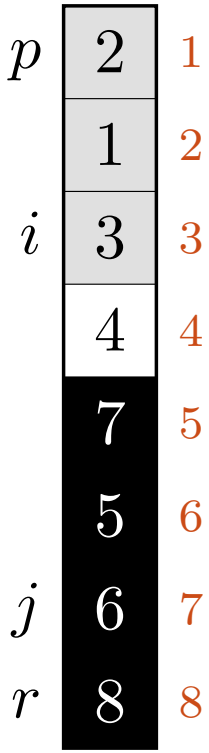
$x, i \;=\; 4, 3$



mlh@ntnu.no

```
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
```

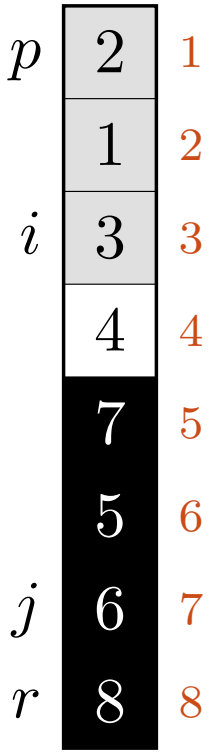
$x, i = 4, 3$



mlh@ntnu.no

```
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
```

→ 4



$x, i = 4, 3$

mlh@ntnu.no

Instans	Løsning
En sekvens A	A, sortert
Dekomponering	Kombinasjon IS
Fordel i to deler, venstre verdier mindre enn høyre	
Delinstanser	Delløsninger IH
Liten og stor del, til venstre og høyre	Liten og stor del, til venstre og høyre, sorterte
Grunntilfelle	Grunnløsning
Sekvens med maks ett element	Samme sekvens, allerede sortert

mlh@ntnu.no

QUICKSORT(A, p, r)

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

mlh@ntnu.no

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

Hvis $\mathrm{A}[p : r]$ har lengde minst 2 . . .

mlh@ntnu.no

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

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

mlh@ntnu.no

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

Sorter små verdier $A[p : q-1]$ rekursivt

mlh@ntnu.no

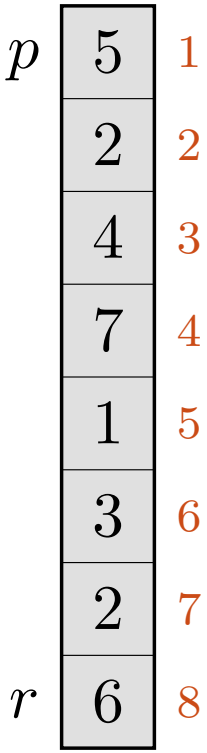
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```

Sorter store verdier $A[q+1:r]$ rekursivt

mlh@ntnu.no

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

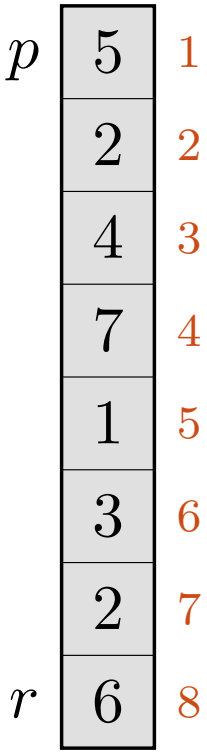
$p, q, r \;=\; 1, -, 8$



mlh@ntnu.no

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

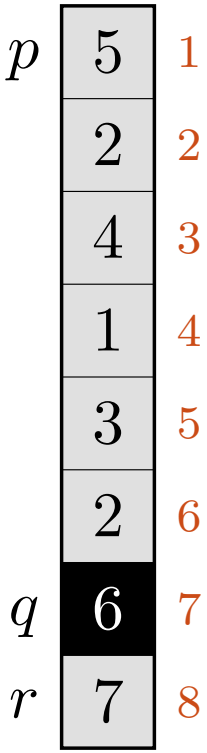
$p, q, r \;=\; 1, -, 8$



mlh@ntnu.no

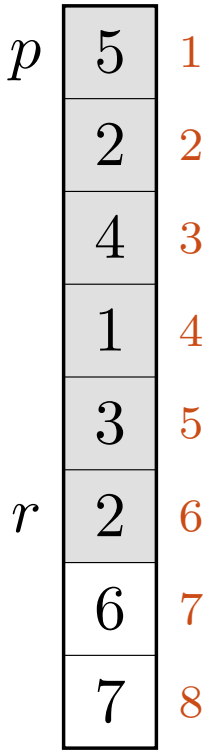
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```

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



mlh@ntnu.no

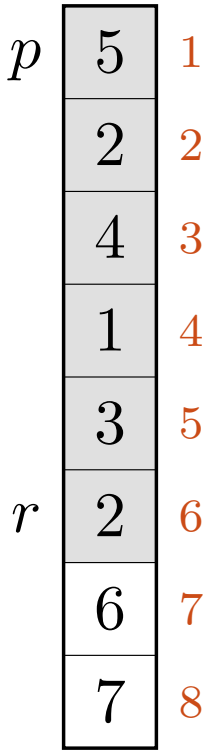
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```



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

mlh@ntnu.no

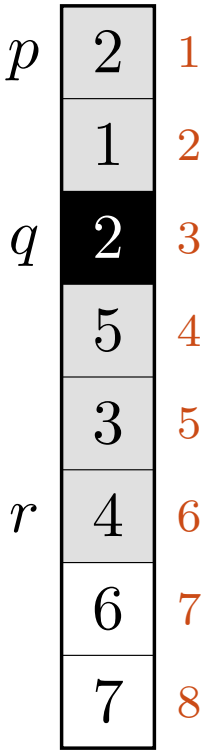
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```



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

mlh@ntnu.no

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

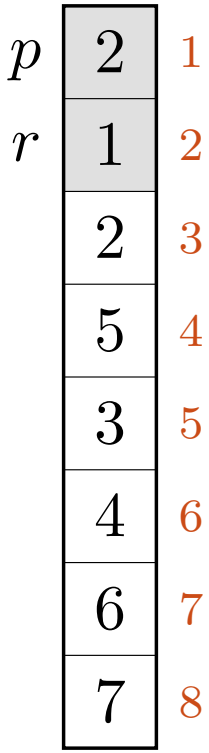


$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6$

mlh@ntnu.no

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

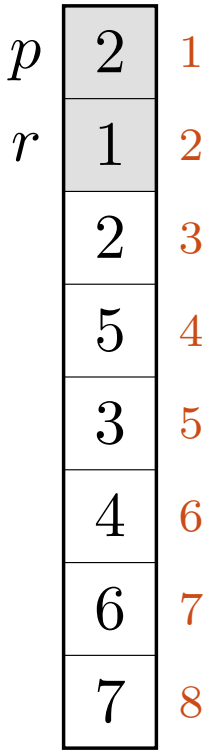
p, q, r = 1, 7, 8 › 1, 3, 6 › 1, –, 2



mlh@ntnu.no

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

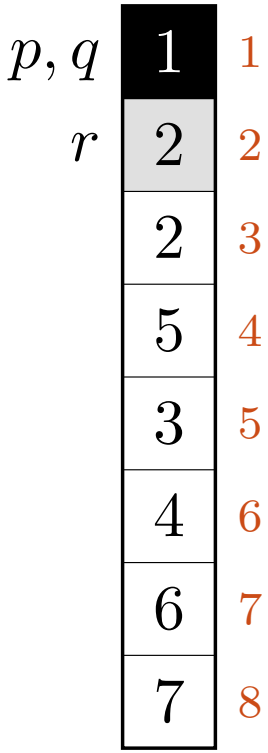
p, q, r = 1, 7, 8 › 1, 3, 6 › 1, –, 2



mlh@ntnu.no

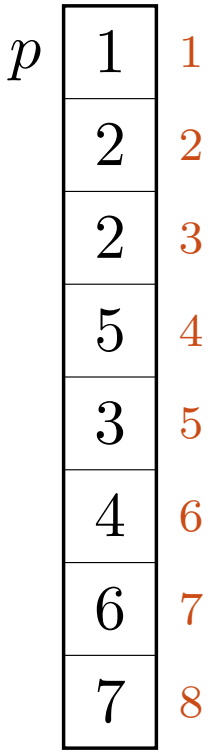
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```

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



mlh@ntnu.no

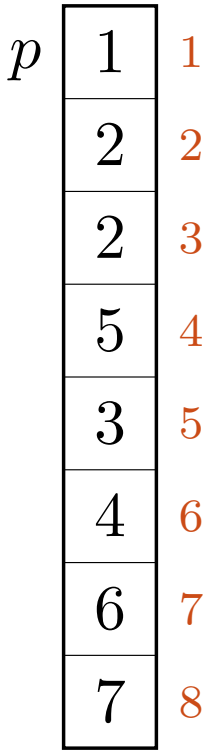
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```



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

mlh@ntnu.no

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

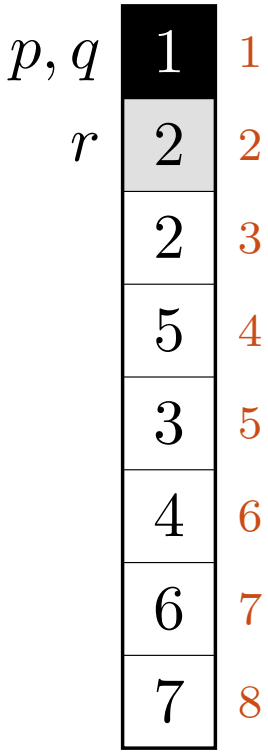


$p, q, r \;=\; 1, 7, 8 \;\rangle\; 1, 3, 6 \;\rangle\; 1, 1, 2 \;\rangle\; 1, -, 0$

mlh@ntnu.no

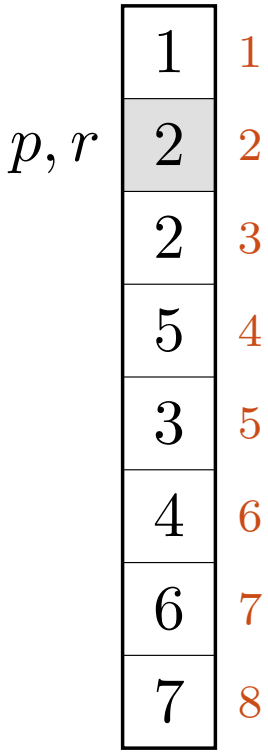
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```

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



mlh@ntnu.no

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

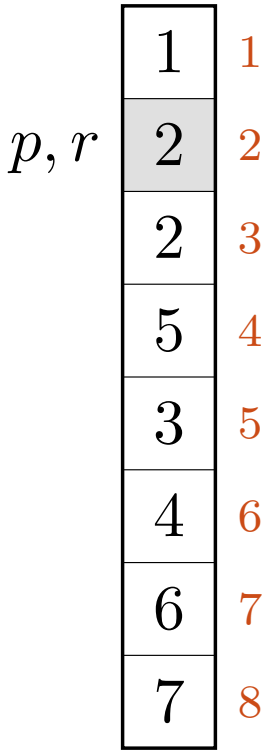


$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6 \rightarrow 1, 1, 2 \rightarrow 2, -, 2$

mlh@ntnu.no

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

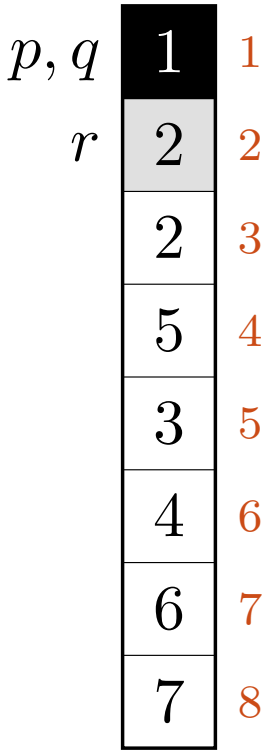
p, q, r = 1, 7, 8 › 1, 3, 6 › 1, 1, 2 › 2, –, 2



mlh@ntnu.no

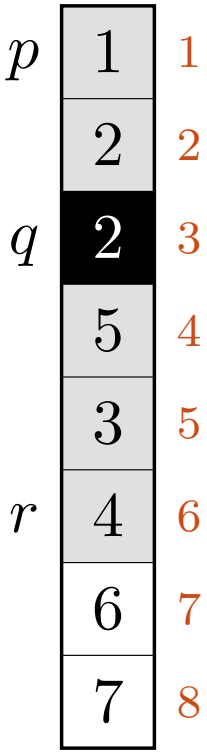
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```

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



mlh@ntnu.no

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

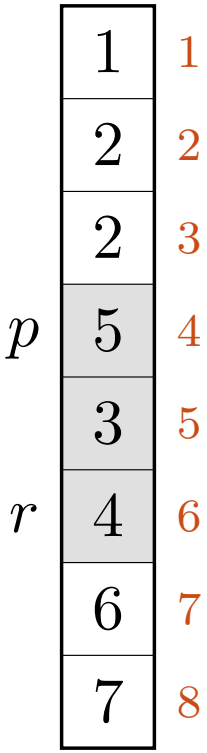


$p, q, r = 1, 7, 8 \rightarrow 1, 3, 6$

mlh@ntnu.no

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

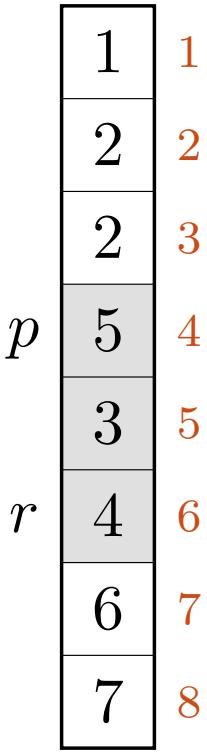
p, q, r = 1, 7, 8 › 1, 3, 6 › 4, –, 6



mlh@ntnu.no

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

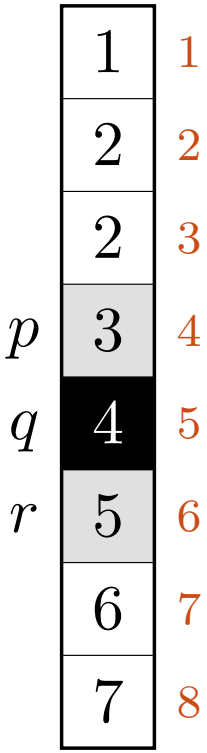
p, q, r = 1, 7, 8 › 1, 3, 6 › 4, –, 6



mlh@ntnu.no

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

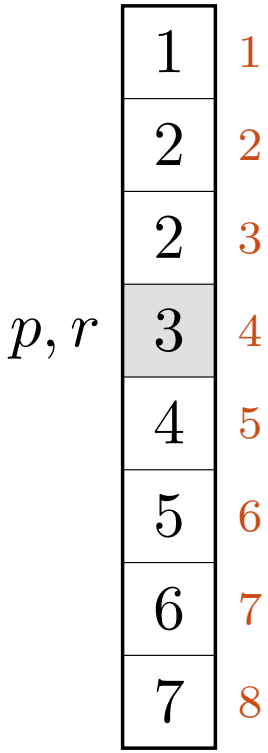
p, q, r = 1, 7, 8 › 1, 3, 6 › 4, 5, 6



mlh@ntnu.no

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

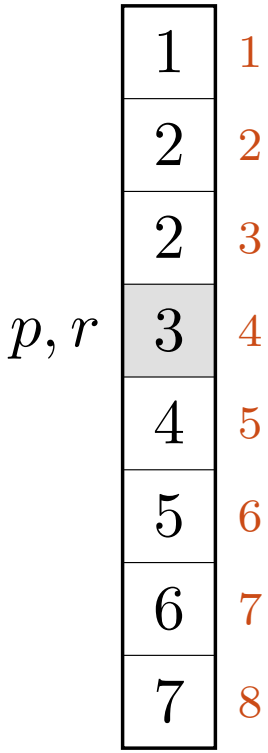
p, q, r = 1, 7, 8 › 1, 3, 6 › 4, 5, 6 › 4, –, 4



mlh@ntnu.no

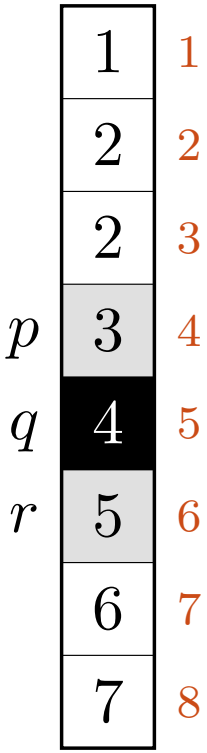
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```

p, q, r = 1, 7, 8 › 1, 3, 6 › 4, 5, 6 › 4, –, 4



mlh@ntnu.no

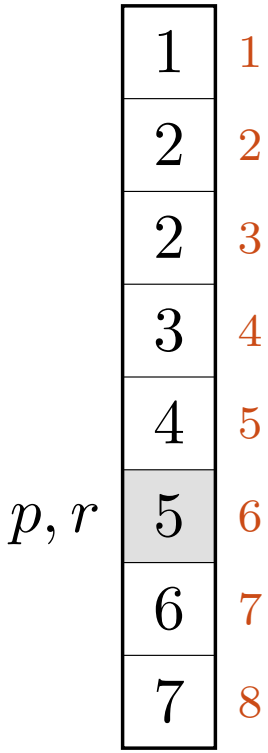
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```



p, q, r = 1, 7, 8 › 1, 3, 6 › 4, 5, 6

mlh@ntnu.no

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

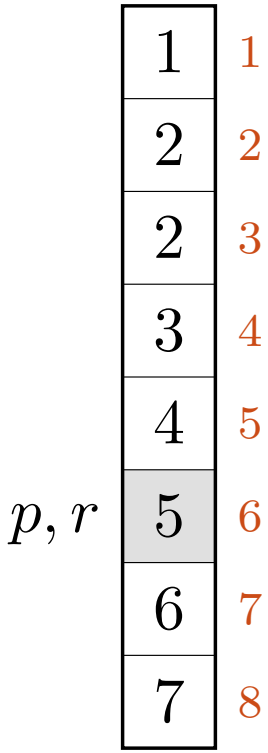


p, q, r = 1, 7, 8 › 1, 3, 6 › 4, 5, 6 › 6, –, 6

mlh@ntnu.no

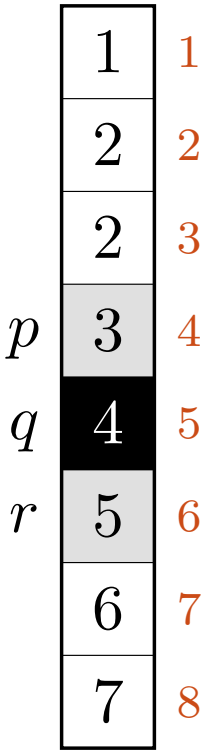
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```

p, q, r = 1, 7, 8 › 1, 3, 6 › 4, 5, 6 › 6, –, 6



mlh@ntnu.no

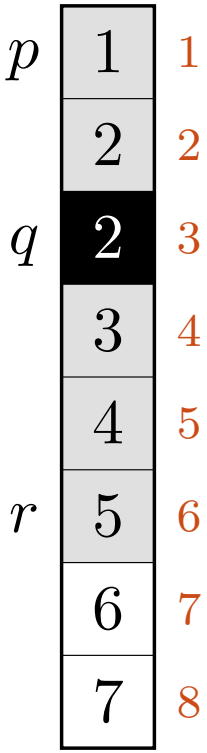
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```



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

mlh@ntnu.no

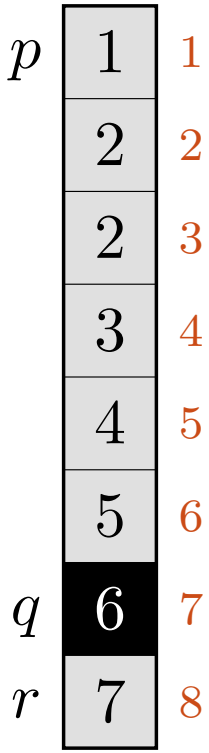
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```



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

mlh@ntnu.no

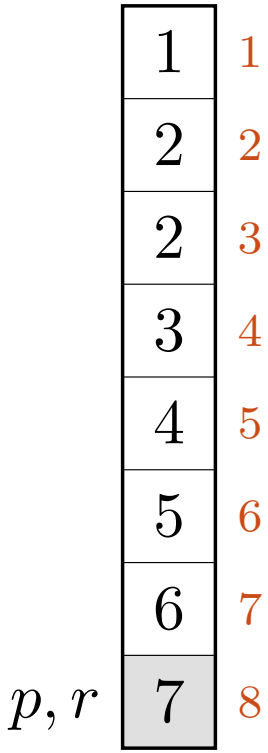
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```



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

mlh@ntnu.no

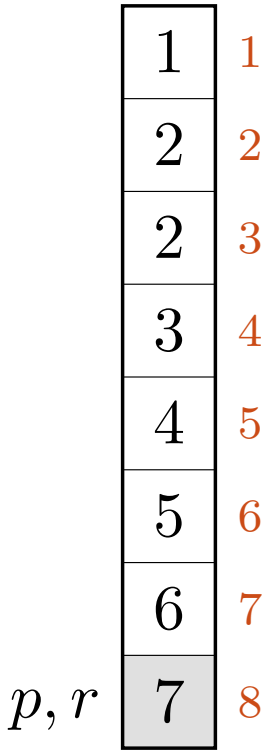
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```



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

mlh@ntnu.no

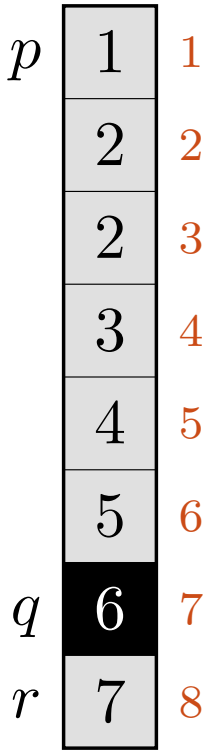
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```



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

mlh@ntnu.no

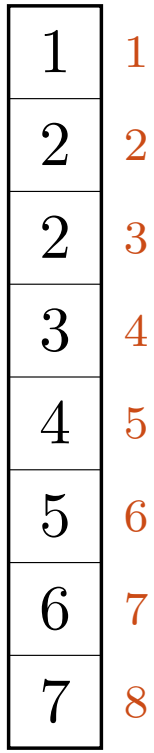
```
QUICKSORT(A, p, r)
1  if p < r
2        q = PARTITION(A, p, r)
3        QUICKSORT(A, p, q − 1)
4        QUICKSORT(A, q + 1, r)
```



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

mlh@ntnu.no

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



mlh@ntnu.no

I beste/gjsn. tilfelle

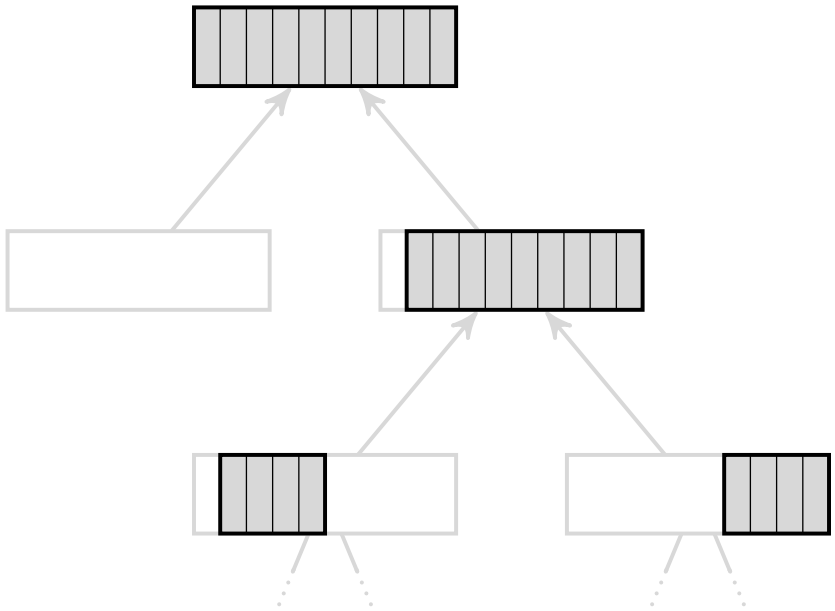
Akkurat som merge sort

Ang. Antagelsen om unike elementer for gjennomsnittlig kjøretid: Se også fotnote 1 (s. 182) og oppg. 7-2.

Gjsn: Antar unike elementer

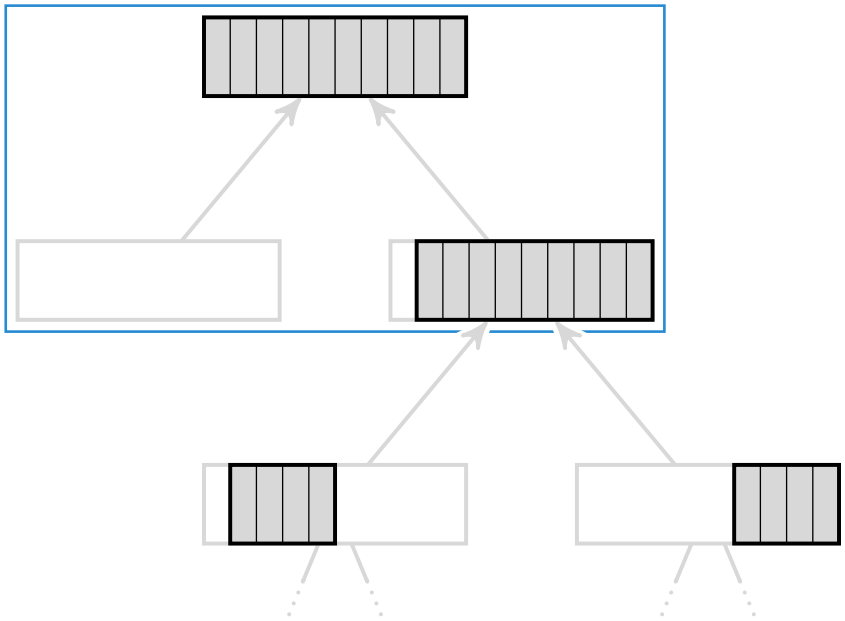
Intuisjon, gjsn.

- Hvis en splitt er veldig ujevn, koster den ene siden også mindre
- Hvis dette bare skjer en viss andel av tiden, er det ikke så farlig
- F.eks. om pivot havner helt ytterst i halvparten av tilfellene ...



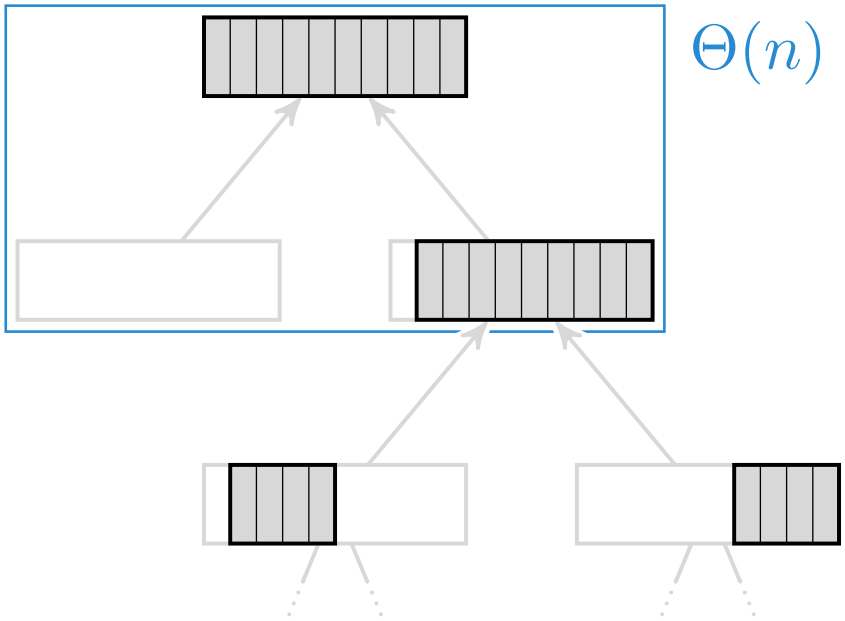
Anta helt jevn og helt ujevn splitt på annenhvert nivå

mlh@ntnu.no



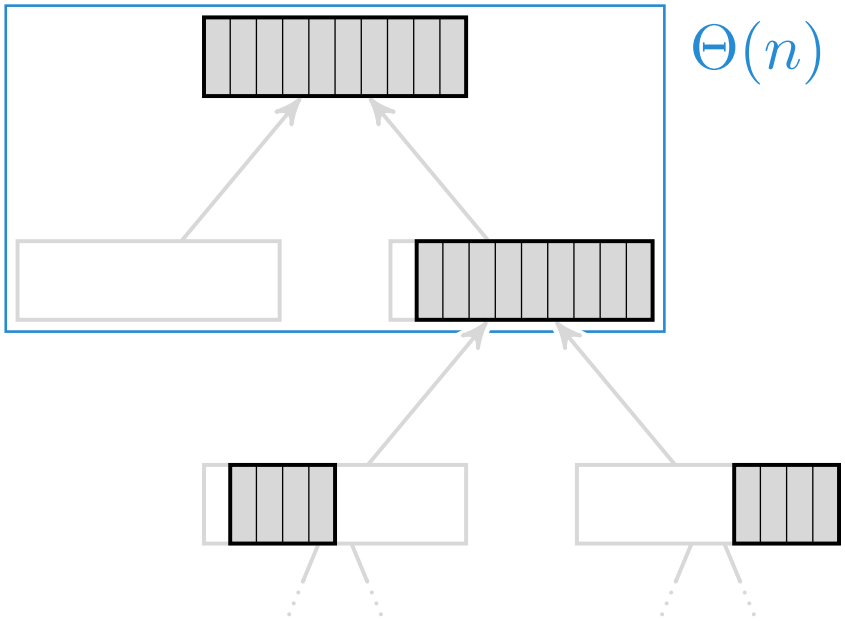
Vi kan regne på disse som en enhet

mlh@ntnu.no



Kostnaden er ikke mer en for én jevn splitt

mlh@ntnu.no



Totalt for hele treet: Fortsatt $\Theta(n \lg n)$

mlh@ntnu.no

I verste tilfelle

Pivot blir helt alene

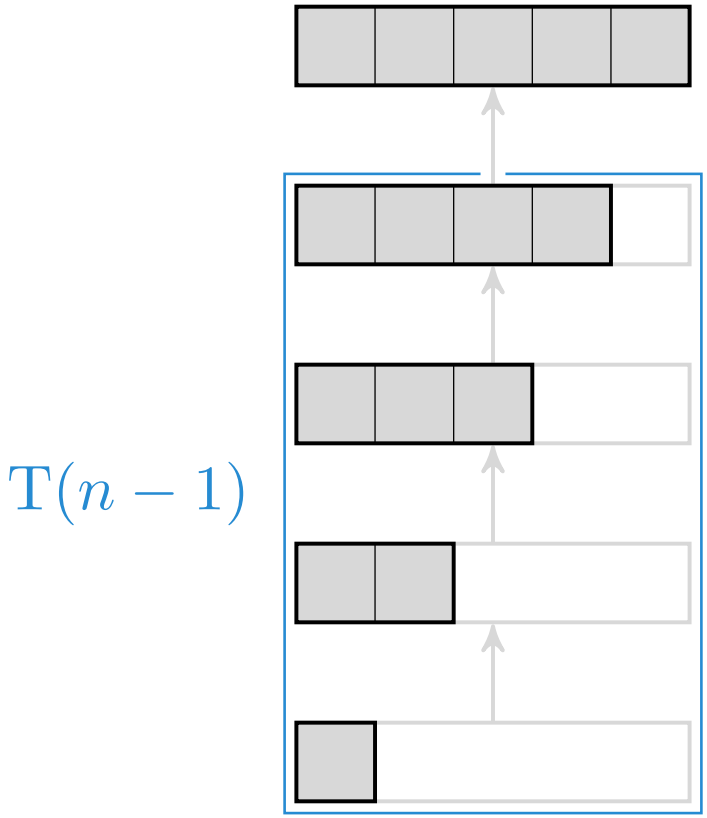
$$\mathrm{T}(n) = \mathrm{T}(n-1) + n$$

mlh@ntnu.no

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

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

mlh@ntnu.no



mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

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

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

mlh@ntnu.no

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

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

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

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

mlh@ntnu.no

Verifikasjon

Med substitusjon/induksjon

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

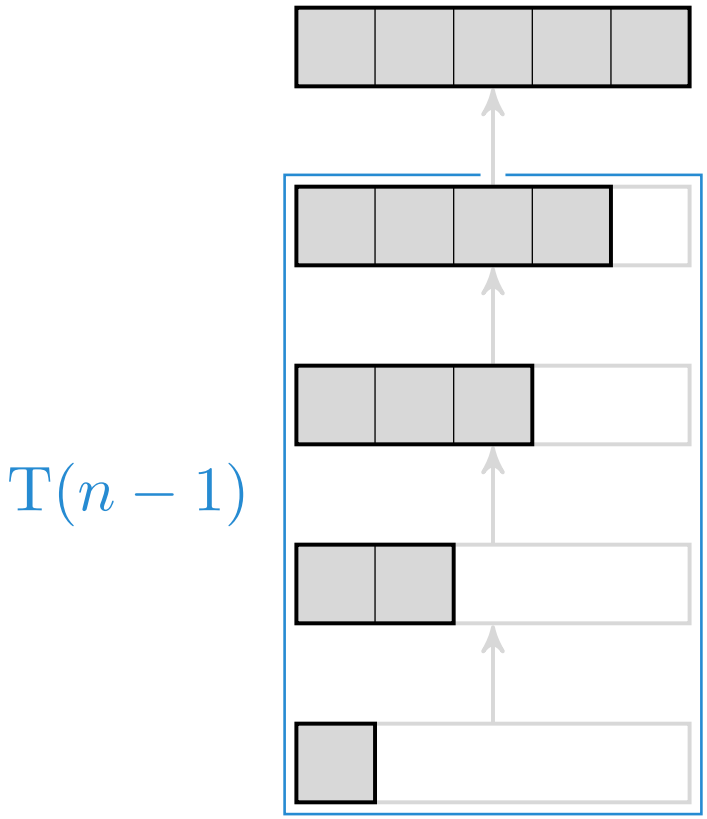
Gitt $\mathrm{T}(n-1) = (n-1)n/2$, vis $\mathrm{T}(n) = n(n+1)/2$

mlh@ntnu.no

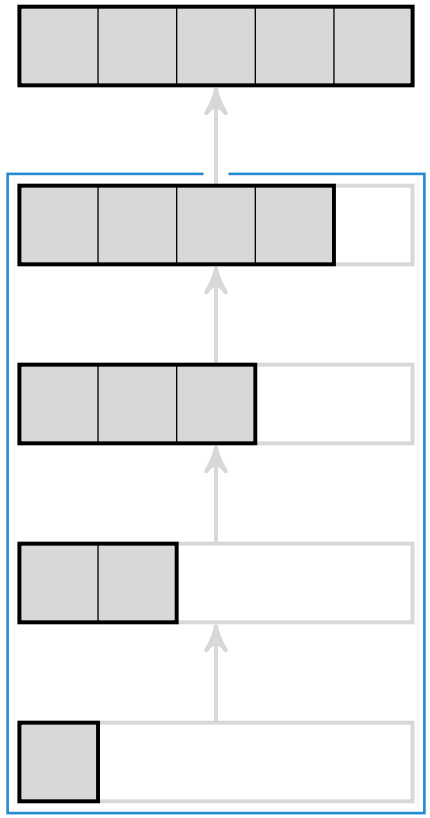
$$\begin{aligned}\mathrm{T}(n) &= \mathrm{T}(n-1) + n \\ &= \frac{(n-1)n}{2} + n \\ &= \frac{n^2 - n + 2n}{2} \\ &= \frac{n(n+1)}{2}\end{aligned}$$

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

mlh@ntnu.no



mlh@ntnu.no



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

mlh@ntnu.no

Randomisering

Så ingen input alltid er kjip

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

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

RANDOMIZED-PARTITION(A, p, r)
1 i = RANDOM(p, r)
2 exchange A[r] and A[i]
3 **return** PARTITION(A, p, r)

Tilfeldig splittelement, heller enn vilkårlig

mlh@ntnu.no

```
RANDOMIZED-QUICKSORT(A, p, r)
1  if p < r
2        q = 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

mlh@ntnu.no

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

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

5:5

Masterteoremet

SIGACT News 36 Fall 1980

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

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

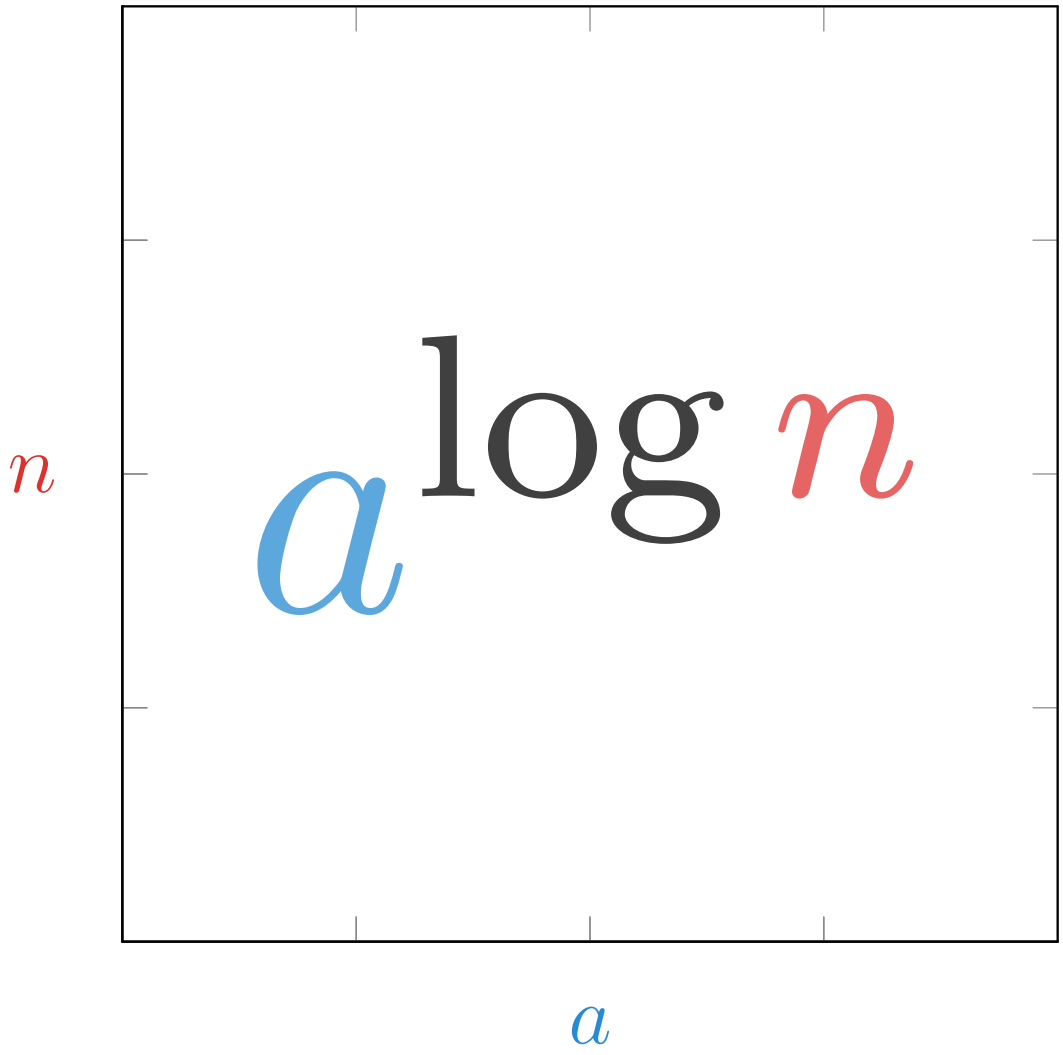
$$\mathrm{T}(n) = a\mathrm{T}(n/b) + f(n)$$

$f(n)$	drivende funksjon
$n^{\log_b a}$	vannskillefunksjon

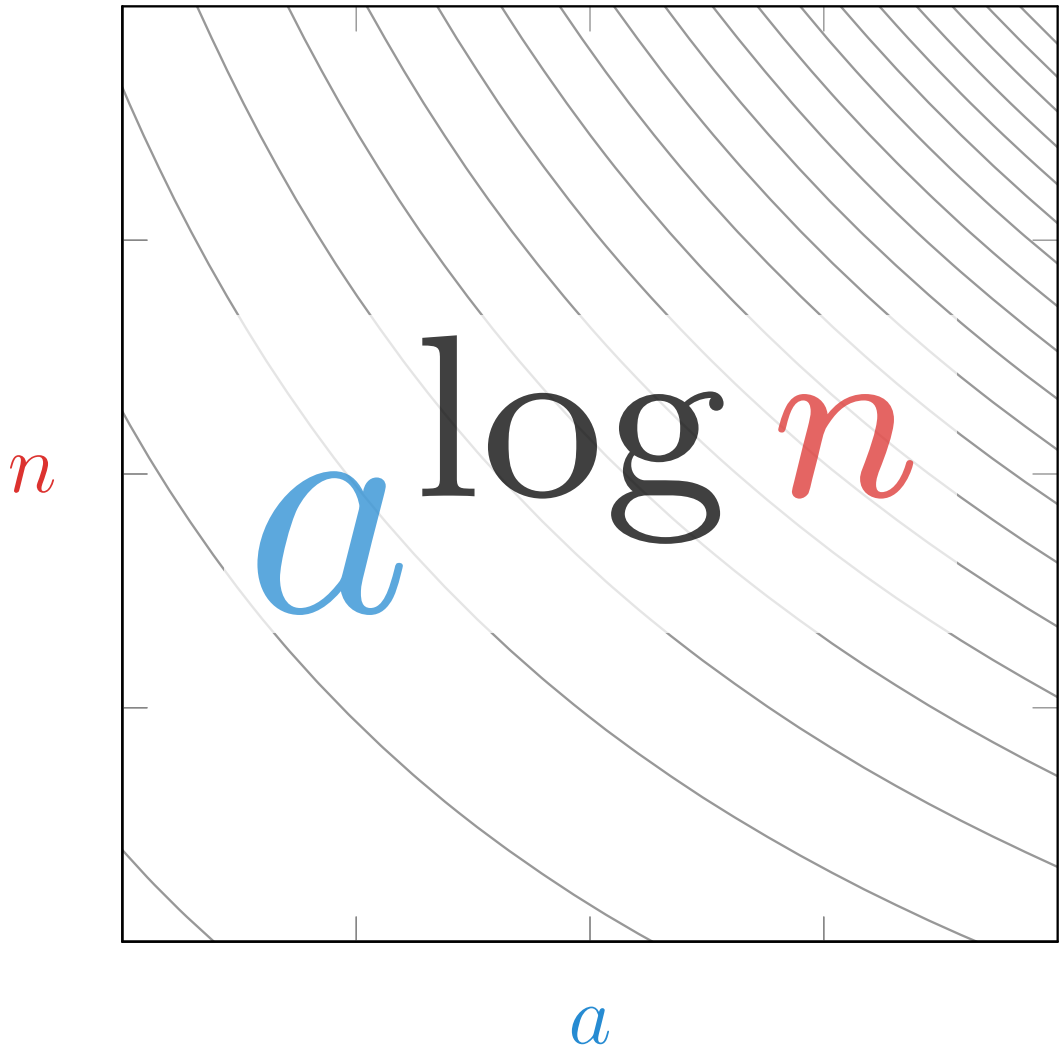
mlh@ntnu.no

$$a^{\log n}$$

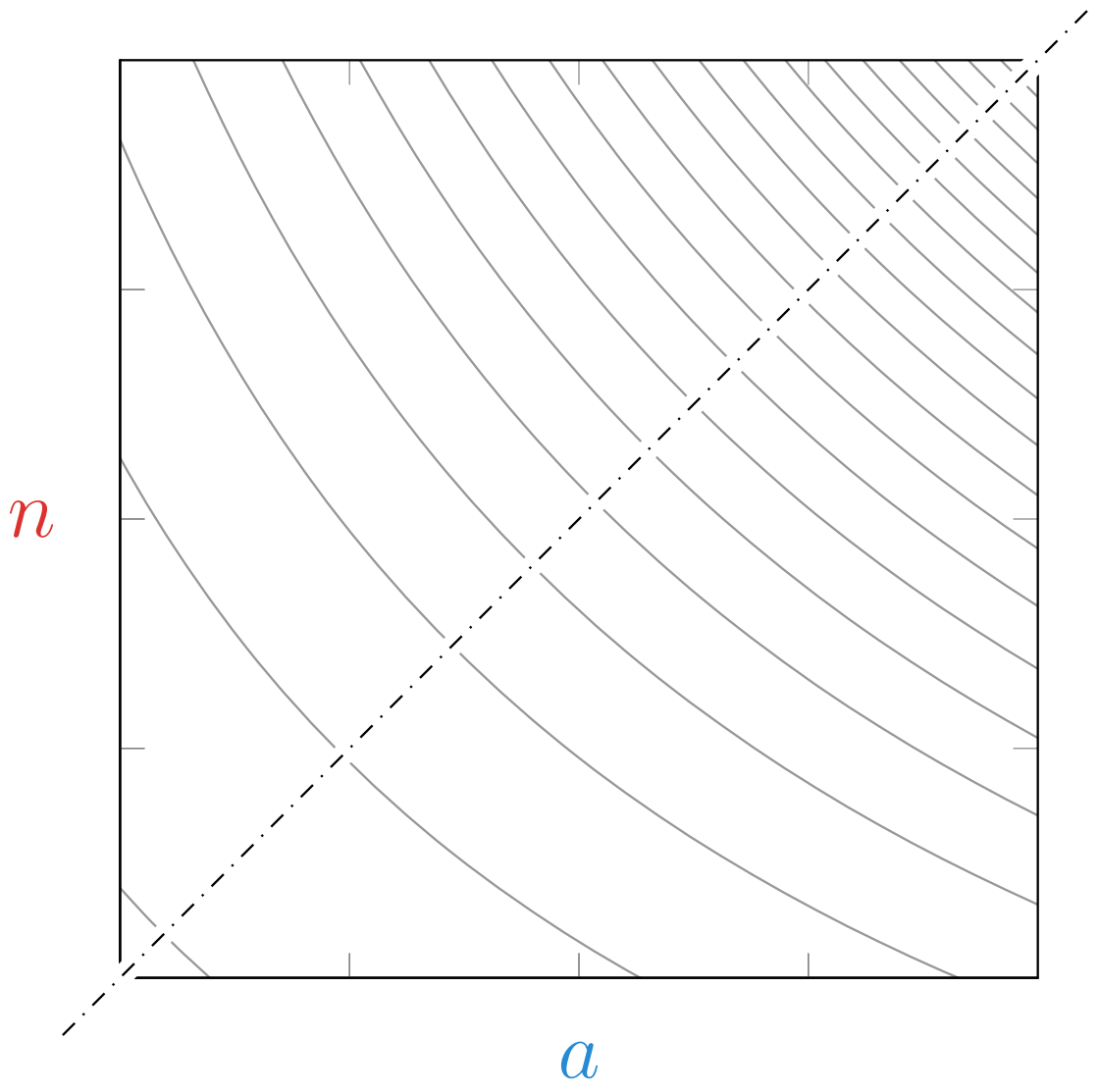
mlh@ntnu.no



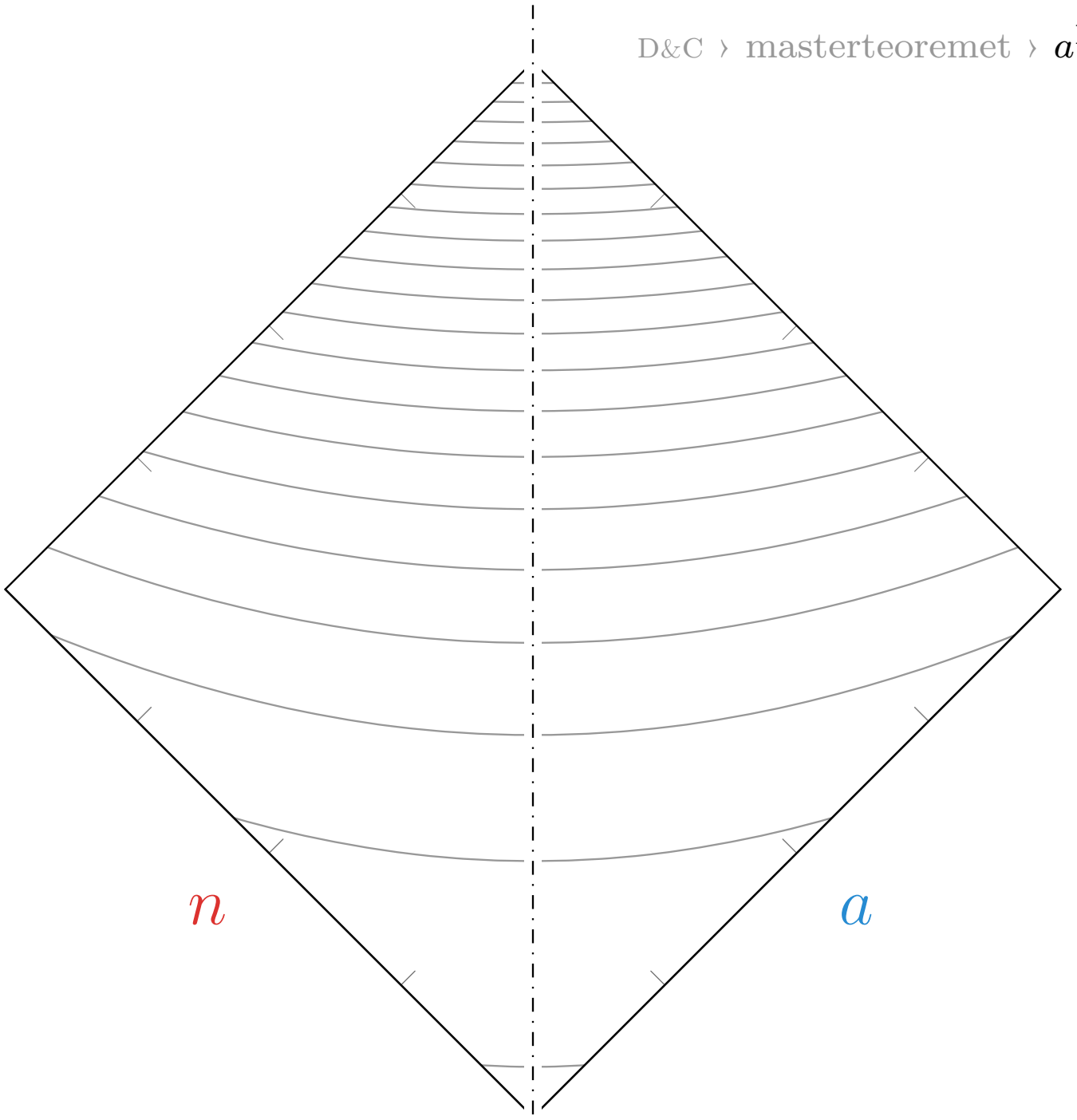
mlh@ntnu.no



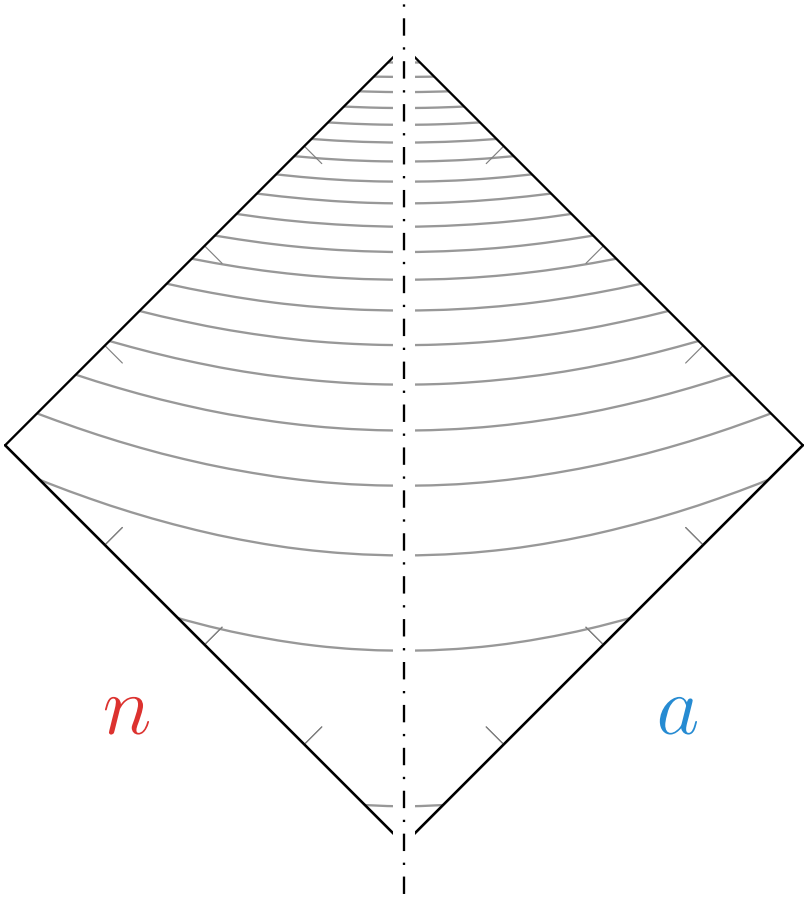
mlh@ntnu.no

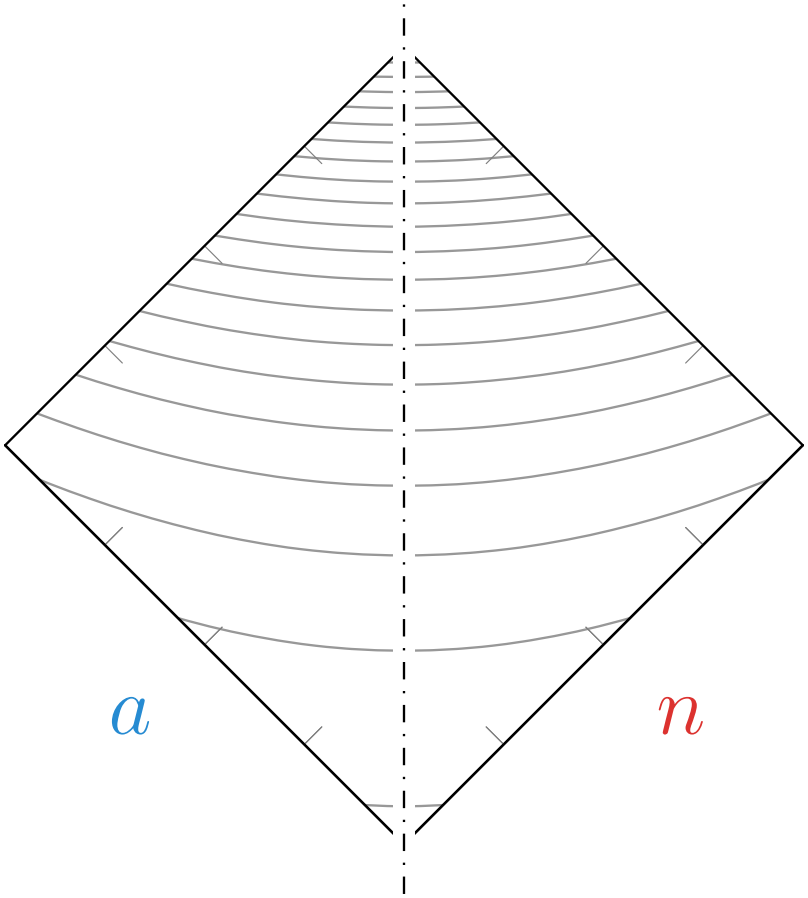


mlh@ntnu.no

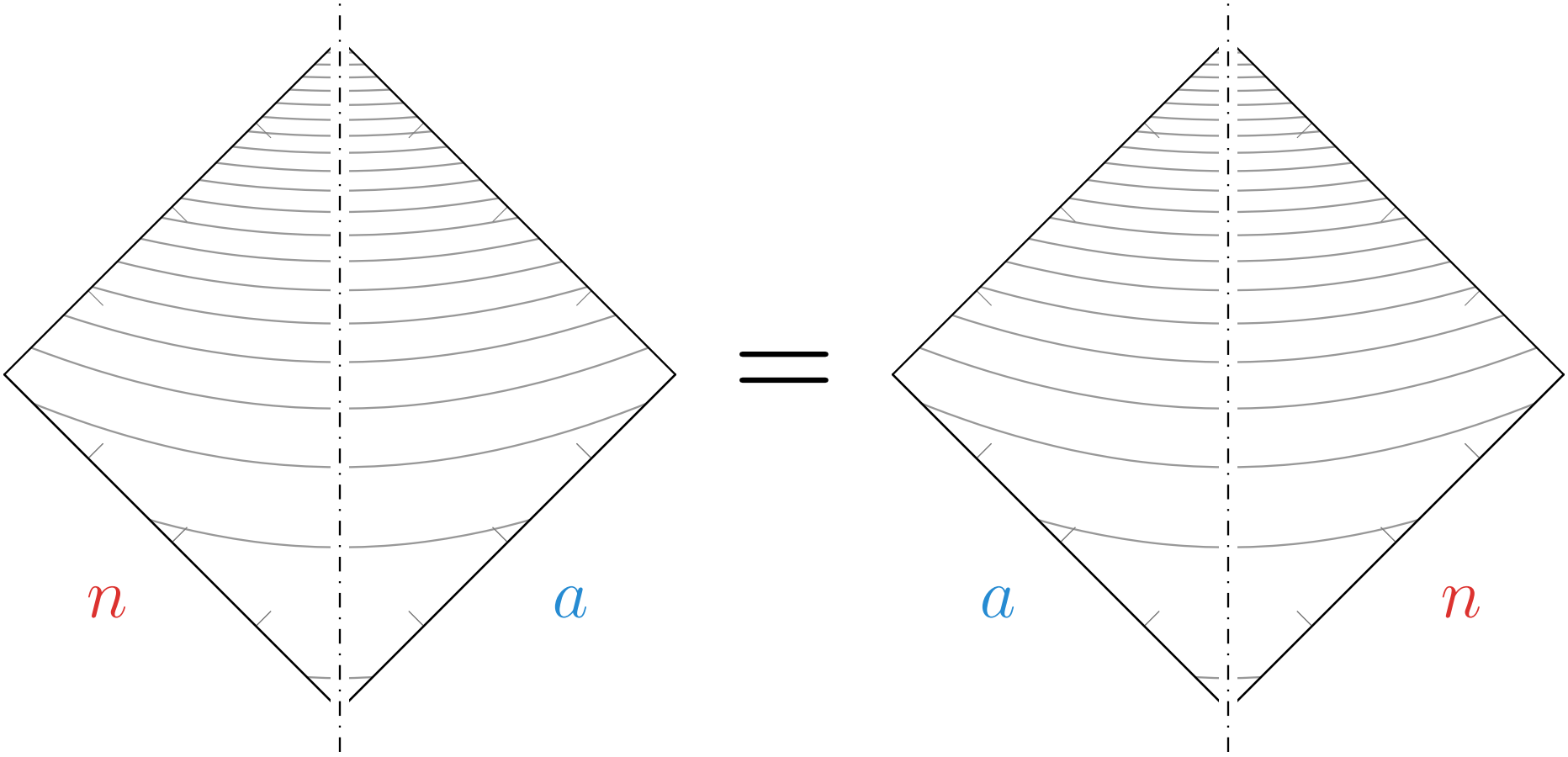


mlh@ntnu.no

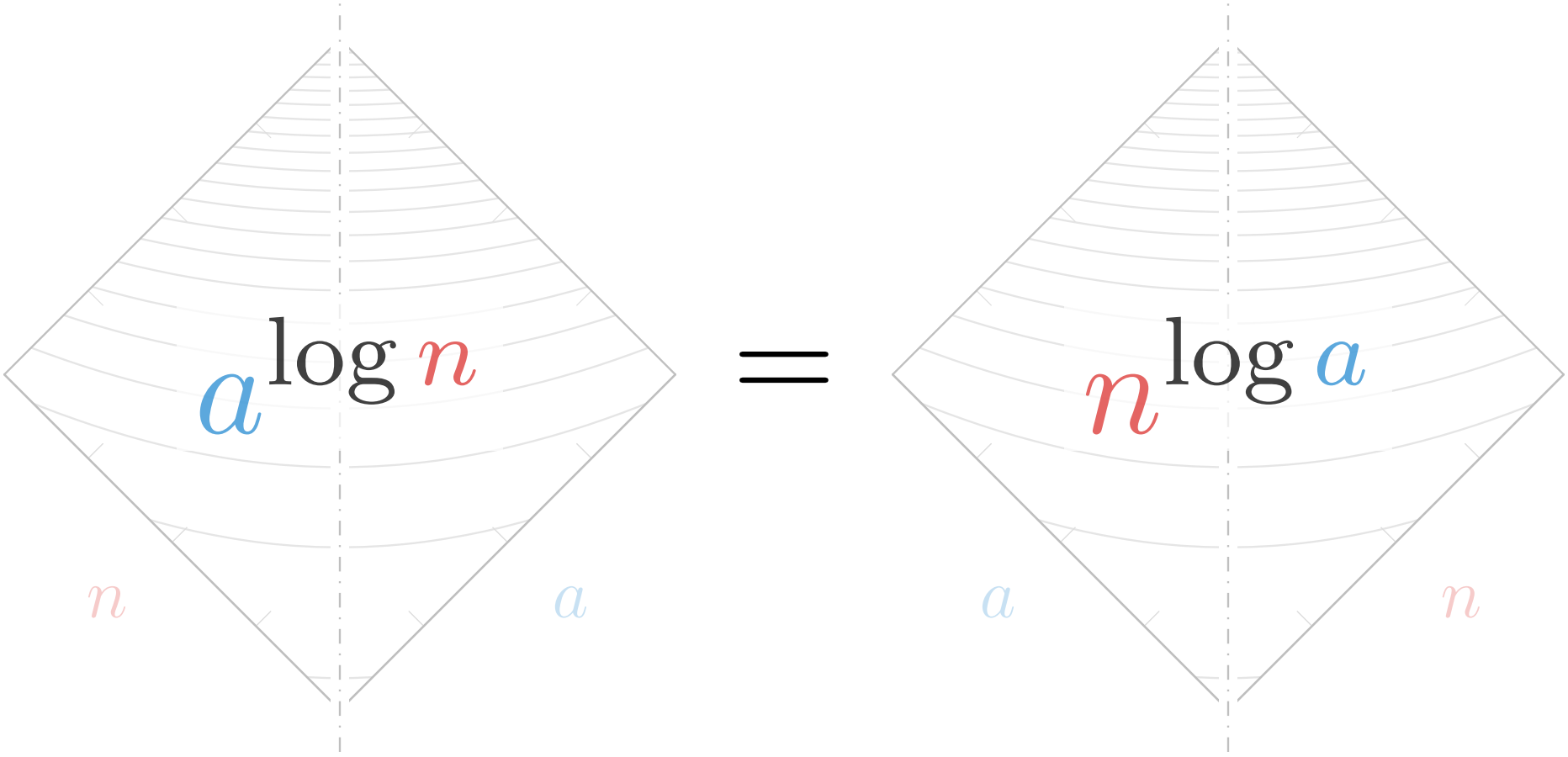




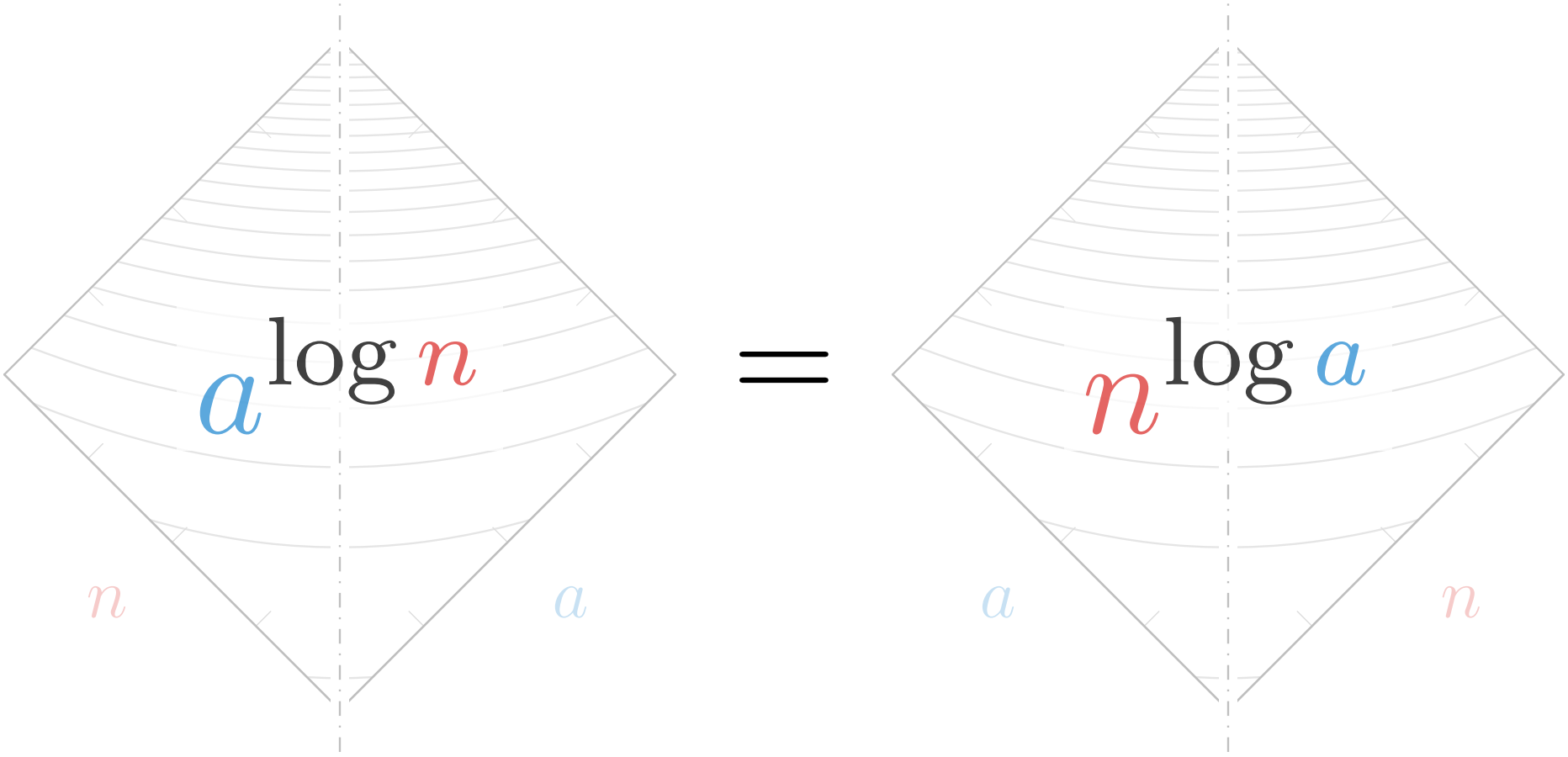
mlh@ntnu.no



mlh@ntnu.no



mlh@ntnu.no



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

mlh@ntnu.no

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

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

$$+ aa\mathrm{T}(n/b/b) \tag{2}$$

mlh@ntnu.no

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

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

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

mlh@ntnu.no

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

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

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

mlh@ntnu.no

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

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

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

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

$$\vdots \qquad \vdots$$

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

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

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

mlh@ntnu.no

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

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

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

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

$$\vdots \qquad \vdots$$

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

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

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

mlh@ntnu.no

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

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

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

mlh@ntnu.no

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

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

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

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

mlh@ntnu.no

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

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

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

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

Teorem 4.1: «The master theorem»

mlh@ntnu.no

$$f(n) = \Theta(n^{\log_b a} \qquad)$$

$$\Downarrow$$

$$\mathrm{T}(n) = \Theta(n^{\log_b a} \qquad \cdot \lg n)$$

Hvert av de $\Theta(\lg n)$ nivåene er $n^{\log_b a}$

mlh@ntnu.no

$$f(n) = \Theta(n^{\log_b a} \qquad)$$

$$\Downarrow$$

$$\mathrm{T}(n) = \Theta(n^{\log_b a} \qquad \cdot \lg n)$$

Men dette holder også om f er *litt* større

mlh@ntnu.no

$$f(n) = \Theta(n^{\log_b a} \lg^k n)$$

$$\Downarrow$$

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

En polylog-faktor ($k \geqslant 0$) over: Nivåene er fortsatt circa like

mlh@ntnu.no

$$f(n) = \Theta(n^{\log_b a} \lg^k n)$$

$$\Downarrow$$

$$\mathrm{T}(n) = \Theta(n^{\log_b a} \lg^k n \cdot \lg n)$$

Så vi har fortsatt $\mathrm{T}(n) = \Theta(f(n) \cdot \lg n)$

mlh@ntnu.no

$$f(n) = \Theta(n^{\log_b a} \lg^k n)$$

$$\Downarrow$$

$$\mathrm{T}(n) = \Theta(n^{\log_b a} \lg^k n \cdot \lg n)$$

(Nytt i 4. utgave)

mlh@ntnu.no

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

1. Rekurrenser

2. Binærsøk

3. Merge sort

4. Quicksort

5. Masterteoremet

1. Rekurrenser

2. Binærsøk

3. Merge sort

4. Quicksort

5. Mastersetningen

D&C

1. Rekurrenser

2. Binærsøk

3. Merge sort

4. Quicksort

5. Mastersetningen

D&C

D&C