

Velkommen til IN2010 repitisjonstime for sortering

Heapsort, Bucket sort, Radixsort



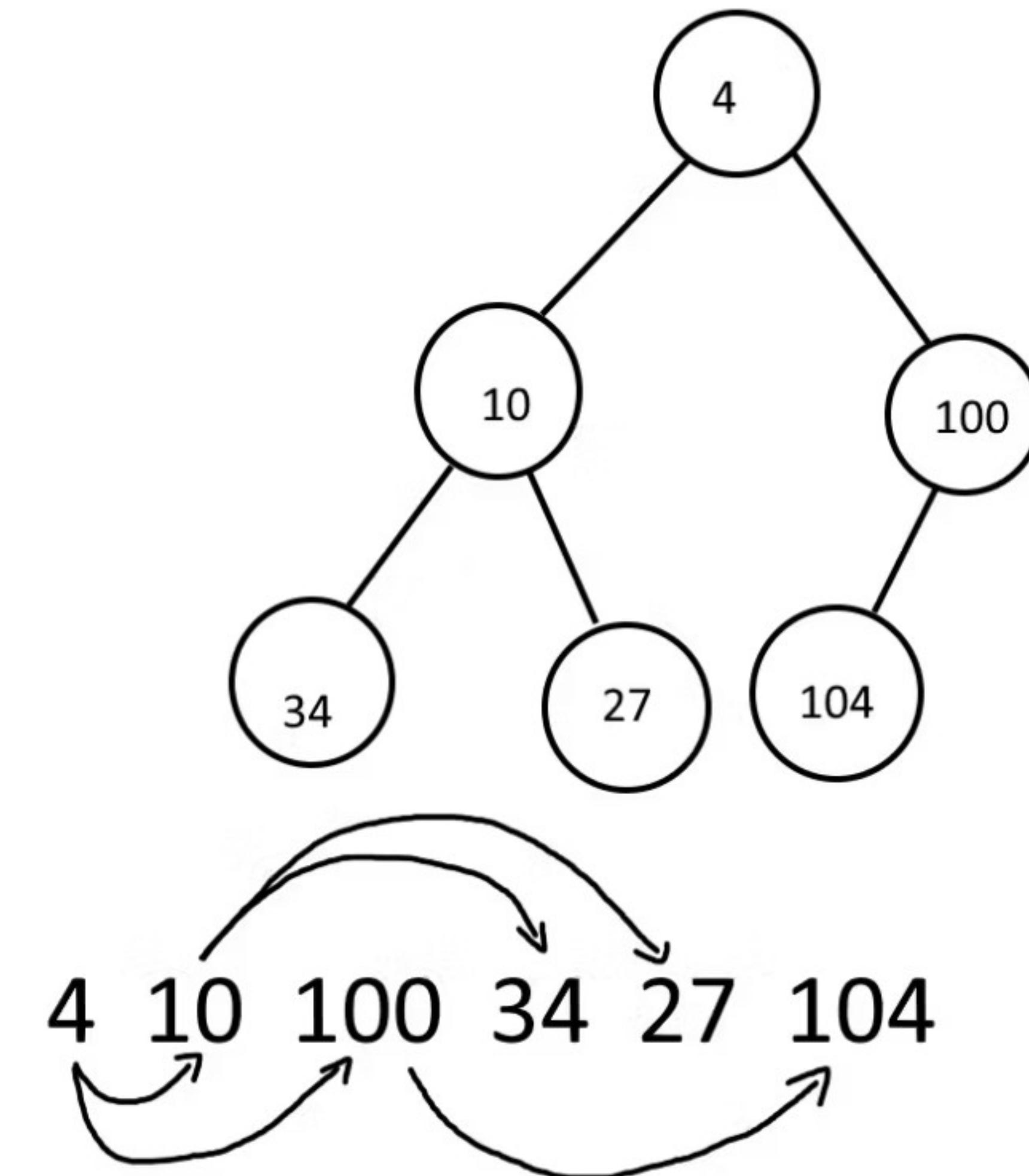
Planen for i dag

- Heapsort, Bucketsort, Radixsort
- Oppgaver relatert til sorteringsalgoritmene
- Spørsmål generelt eller oppgaver



Heapsort (ikke in-place)

- Konverterer listen vi sorterer om til en **Min-Heap**
 - BuildMinHeap(A)
- Popper det minste elementet i **heapen** og legger til på slutten av en **NY** liste
 - Repeterer dette helt til heapen er tom



Heapsort (in-place)

- Istedetfor å bruke en Min-Heap så bruker vi en **Max-Heap**
 - Det største elementet ligger på starten av heapen
- Først gjør vi om listen til en **Max-heap**
 - BuildMaxHeap(A)
- Bytter plass mellom første og siste element i heapen
- Deretter minker vi heapen til å bare gjelde for 0 til $n-2$
- Deretter igjen bobbler vi ned det elementet som ligger på starten
 - Med andre ord fikser opp i max-heapen som gjelder for 0 til $n-2$
- Repeterer dette til heapen blir minsket til 0



```
def heap_sort(A):
    build_max_heap(A, len(A))
    for i in range(len(A)-1, -1, -1):
        A[0], A[i] = A[i], A[0]
        bubble_down(A, 0, i)
```



```
def build_max_heap(A, n):
    for i in range(n//2, -1, -1):
        bubble_down(A, i, n)
```



```
def bubble_down(A, i, n):
    largest = i
    left = 2*i + 1
    right = 2*i + 2

    if left < n and A[largest] < A[left]:
        largest, left = left, largest

    if right < n and A[largest] < A[right]:
        largest, right = right, largest

    if i != largest:
        A[i], A[largest] = A[largest], A[i]
    bubble_down(A, largest, n)
```

```
def heap_sort(A):
    build_max_heap(A, len(A))
    for i in range(len(A)-1, -1, -1):
        A[0], A[i] = A[i], A[0]
        bubble_down(A, 0, i)
```



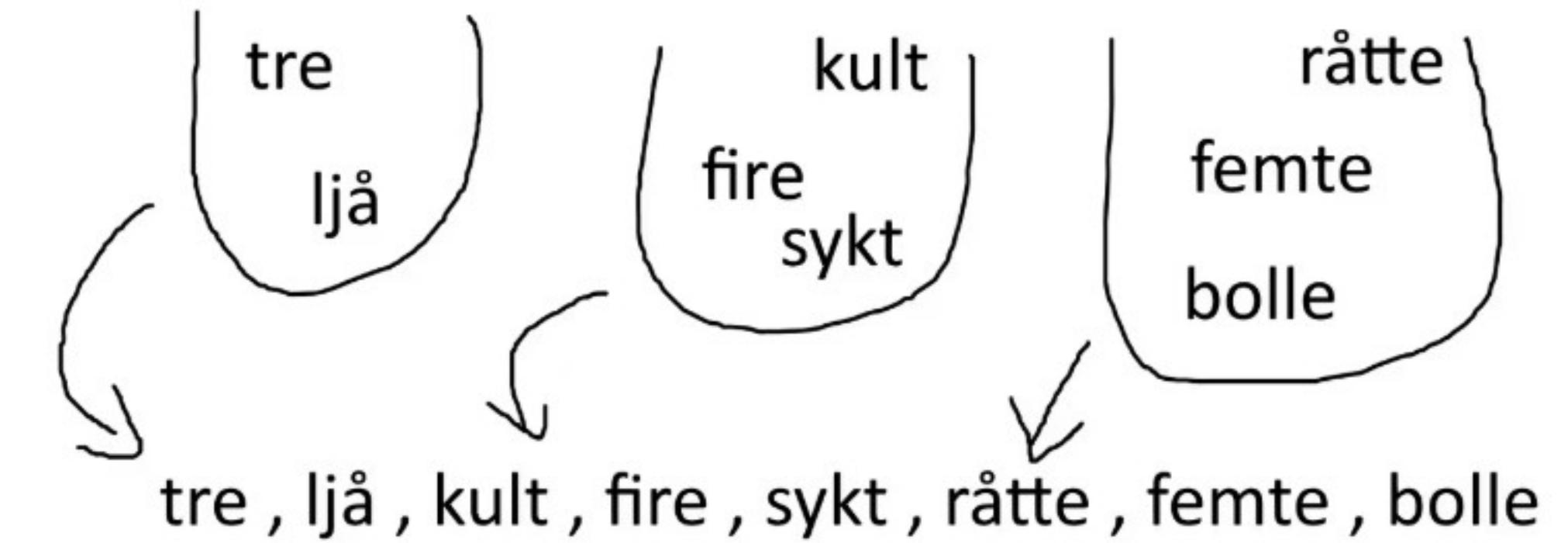
Heapsort kjøretid

- Begge implementasjonene baserer seg på en heap
- Å lage en heap er en $n \log n$ prosess
- Heaps har kjøretid $O(\log n)$ for innsetting, removeMin/Max
- Antall ganger vi bruker heapen under sorteringen er n ganger
- Dette gir total kjøretiden $O(n \log n)$



Bucket sort

- Lager N bøtter der hvor N er hvor mange kategorier det er
- Deretter går over alle elementene i listen og setter dem inn på slutten i deres tilhørende bøtte
- Til slutt går over alle bøttene og legger dem tilbake i listen



Bucket sort kort-eksempel

- La oss ta for oss spillkort
- Vi vil sortere på nummer så mønster
 - $1 < 2 < 3 < 4 < \dots < 13$
 - spar < hjerte < kløver < ruter
- Lager 13 bøtter for antall nummere
 - Setter dem inn i bøttene og tar dem ut
- Lager 4 nye bøtter for antall mønster
 - Setter dem inn i bøttene og tar dem ut

$A\heartsuit 2\heartsuit A\clubsuit J\clubsuit 9\spadesuit 6\diamondsuit 7\diamondsuit 7\clubsuit K\heartsuit 8\heartsuit$

$[A\heartsuit A\clubsuit] [2\heartsuit] [] [] [] [6\diamondsuit] [7\diamondsuit 7\clubsuit] [8\heartsuit] [9\spadesuit] [] [J\clubsuit] [] [K\heartsuit]$

$A\heartsuit A\clubsuit 2\heartsuit 6\diamondsuit 7\diamondsuit 7\clubsuit 8\heartsuit 9\spadesuit J\clubsuit K\heartsuit$

$[A\clubsuit 7\clubsuit J\clubsuit] [6\diamondsuit 7\diamondsuit] [9\spadesuit] [A\heartsuit 2\heartsuit 8\heartsuit K\heartsuit]$

$A\clubsuit 7\clubsuit J\clubsuit 6\diamondsuit 7\diamondsuit 9\spadesuit A\heartsuit 2\heartsuit 8\heartsuit K\heartsuit$



Raddix sort

- Veldig kort er det en bucket sort inne i en for løkke
- Løkken iterer over en liste med greier man skal sortere
- Vi kaller antall greier man sorterer etter **d**
- Eksempel fra forelesningen
 - Sorterer etter sifferne i et tall, starter bakerst i tallet
 - **d** blir antall siffer i det største tallet

```
ALGORITHM: RADIX SORT FOR POSITIVE HEFTALL
Input: Et array A med n positive heltall
Output: Et sortert array med de samme n positive heltallene
1 Procedure RadixSort(A)
2   d ← antall siffer i det største tallet
3   for i ← d – 1 down to 0 do
4     | A ← BucketSort(A) etter det ite sifferet
5   return A
```



Kjøretid for Bucketsort og Raddixsort

- Bucketsort
 - Går over alle n elementer og setter dem inn de N bøttene
 - Tar ut n elementer fra alle N bøttene
 - Dette gir en $O(N + n)$ kjøretid (er n elementer til sammen i de N bøttene)
- Radixsort
 - Bucketsort har $O(N + n)$
 - d er antall greier man sorterer på
 - Med andre ord gjør Bucketsort d ganger
 - Det gir $O(d * (N + n))$
 - Hvis man vet N og d så kan man runde det ned til $O(n)$



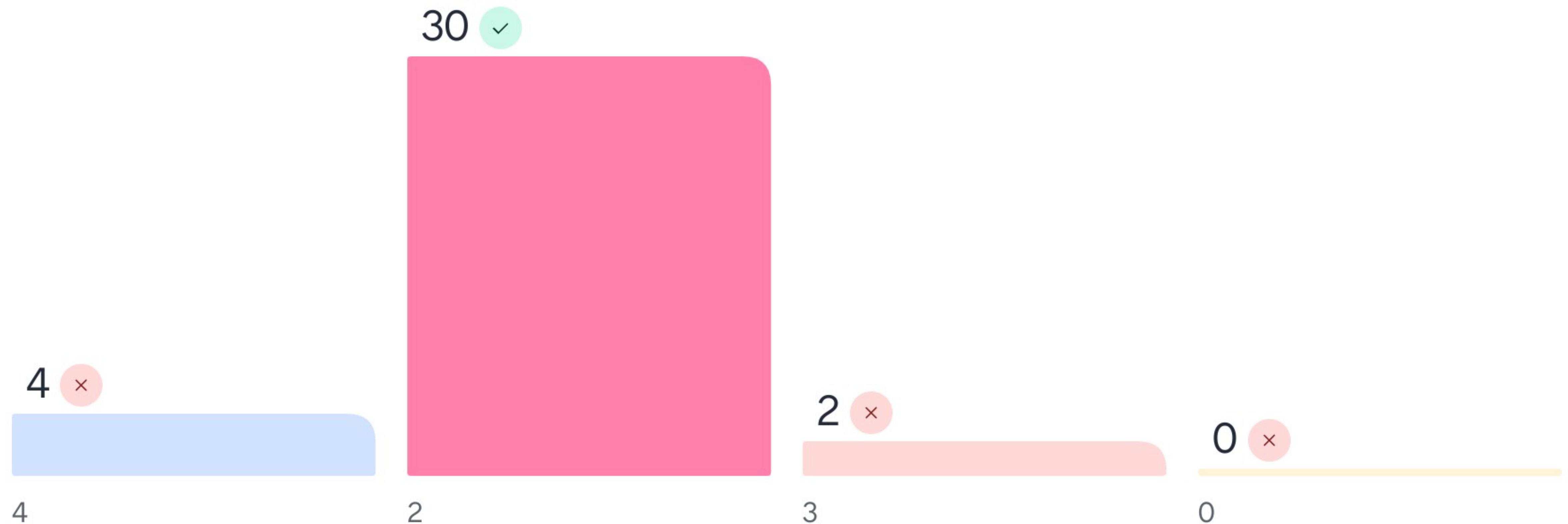
Kode showcase



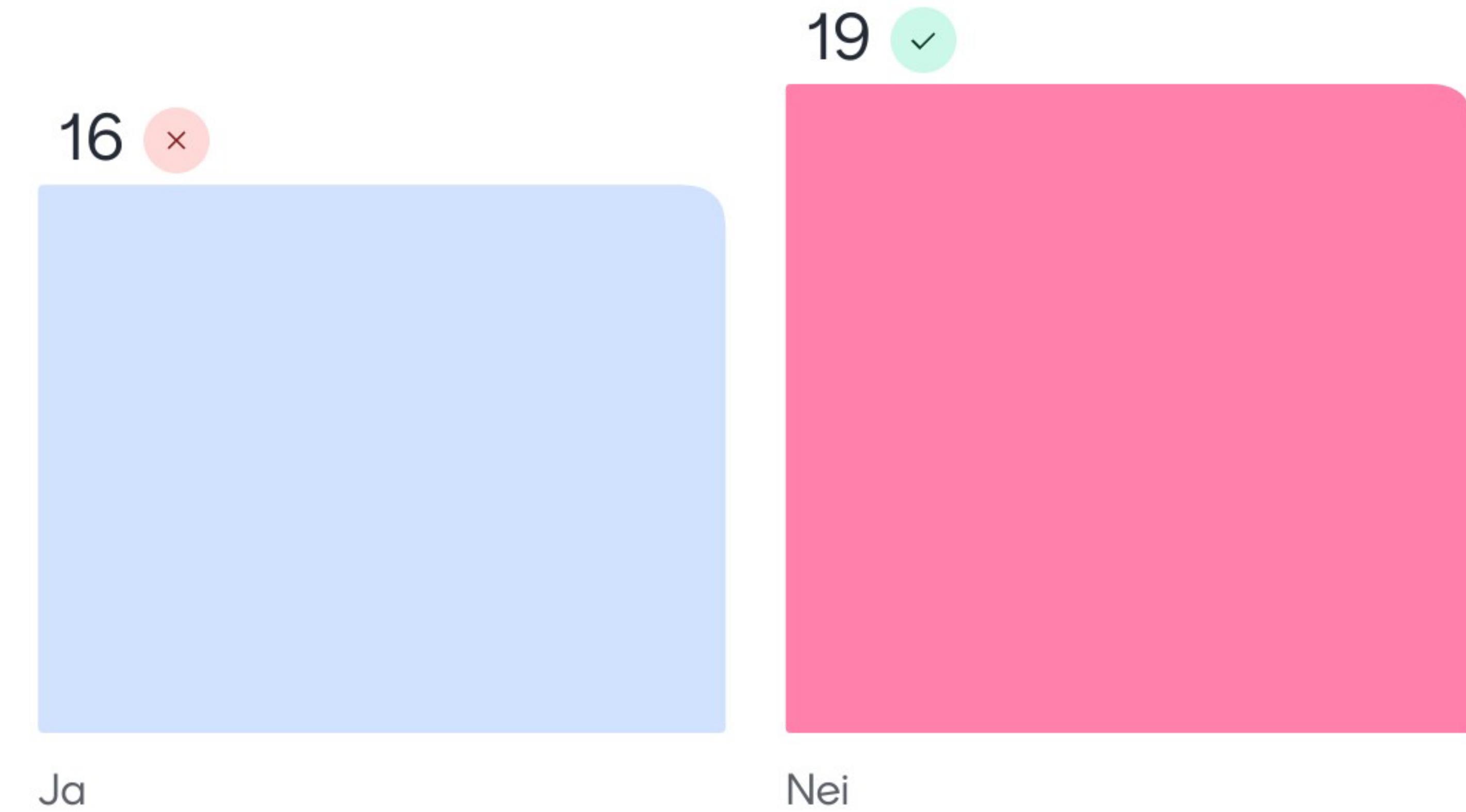
Quiz



Hvor mange iterasjoner inn i heapsort er denne listen på? [78, 50, 49, 38, 27, 12, 100, 102]

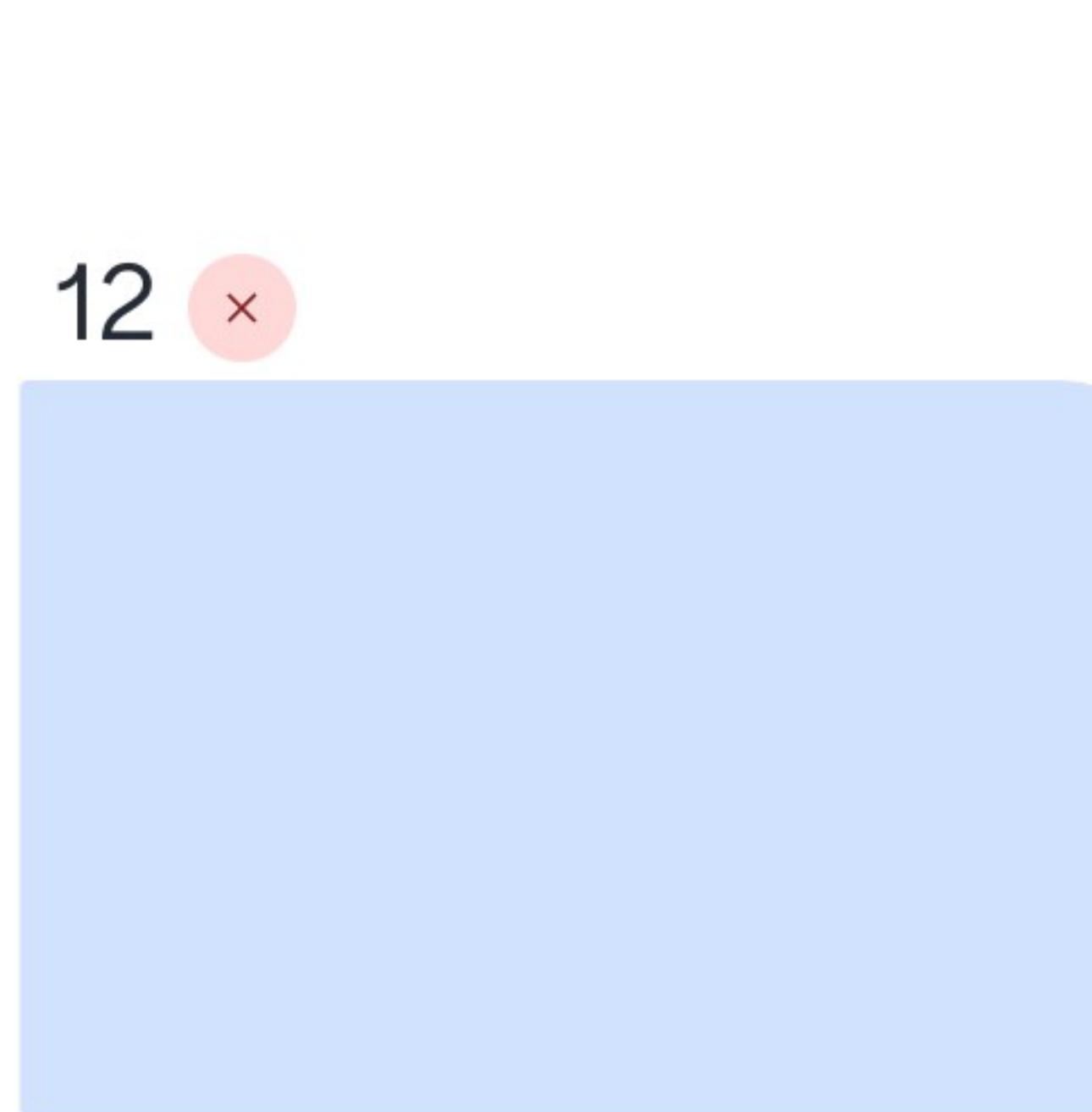


Er Heapsort stabil?

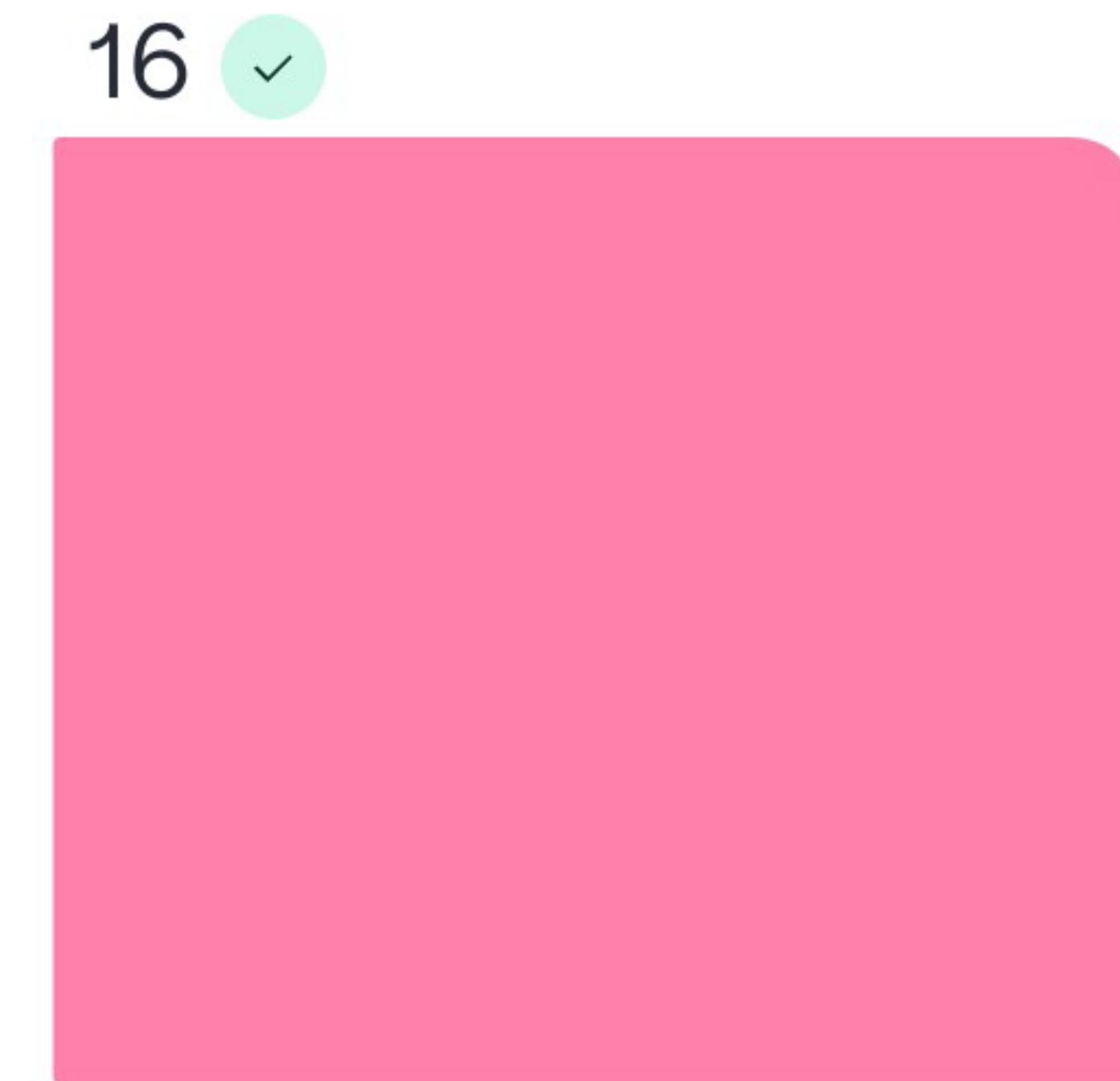


BRUKER IN-PLACE HEAPSORT

Hva må man gjøre for å reverse en heapsort? (reversere vil si at de største elementene ligger først)



Lage en maxheap med de $n/2$ bakerste elementene

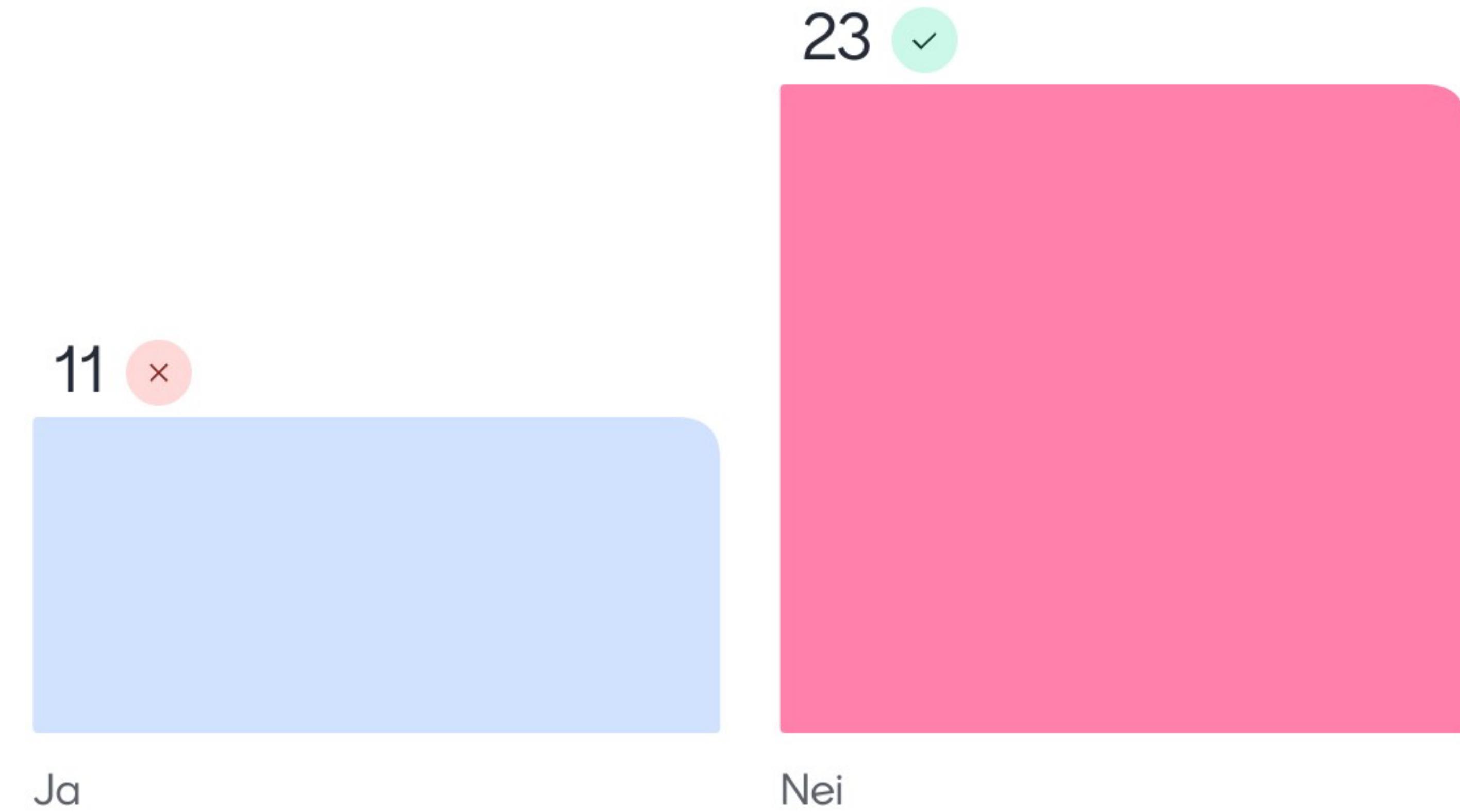


Bytte ut max-heapen med en min-heap

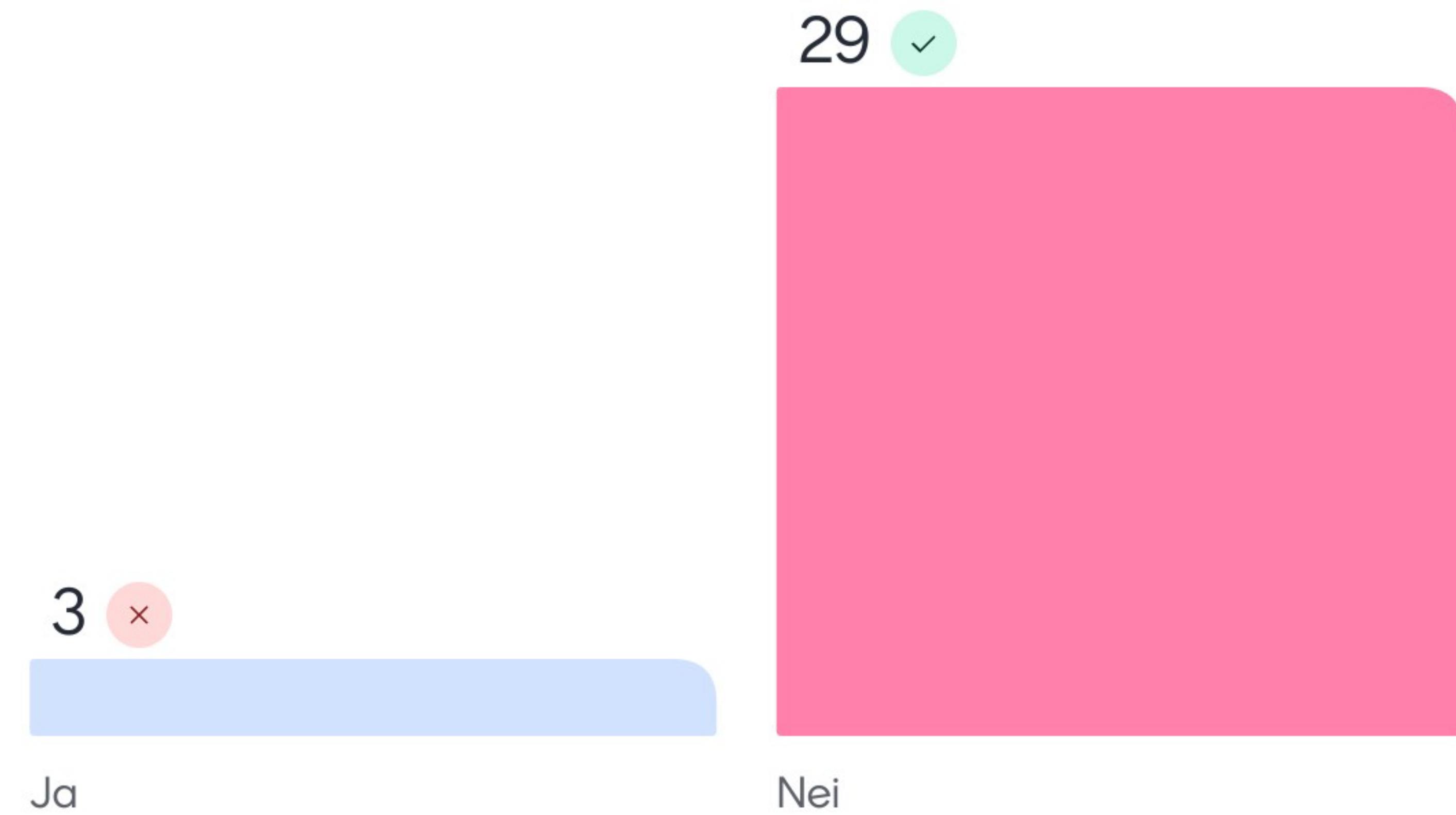


Flytte det bakerste elementet i heapen til første posisjon

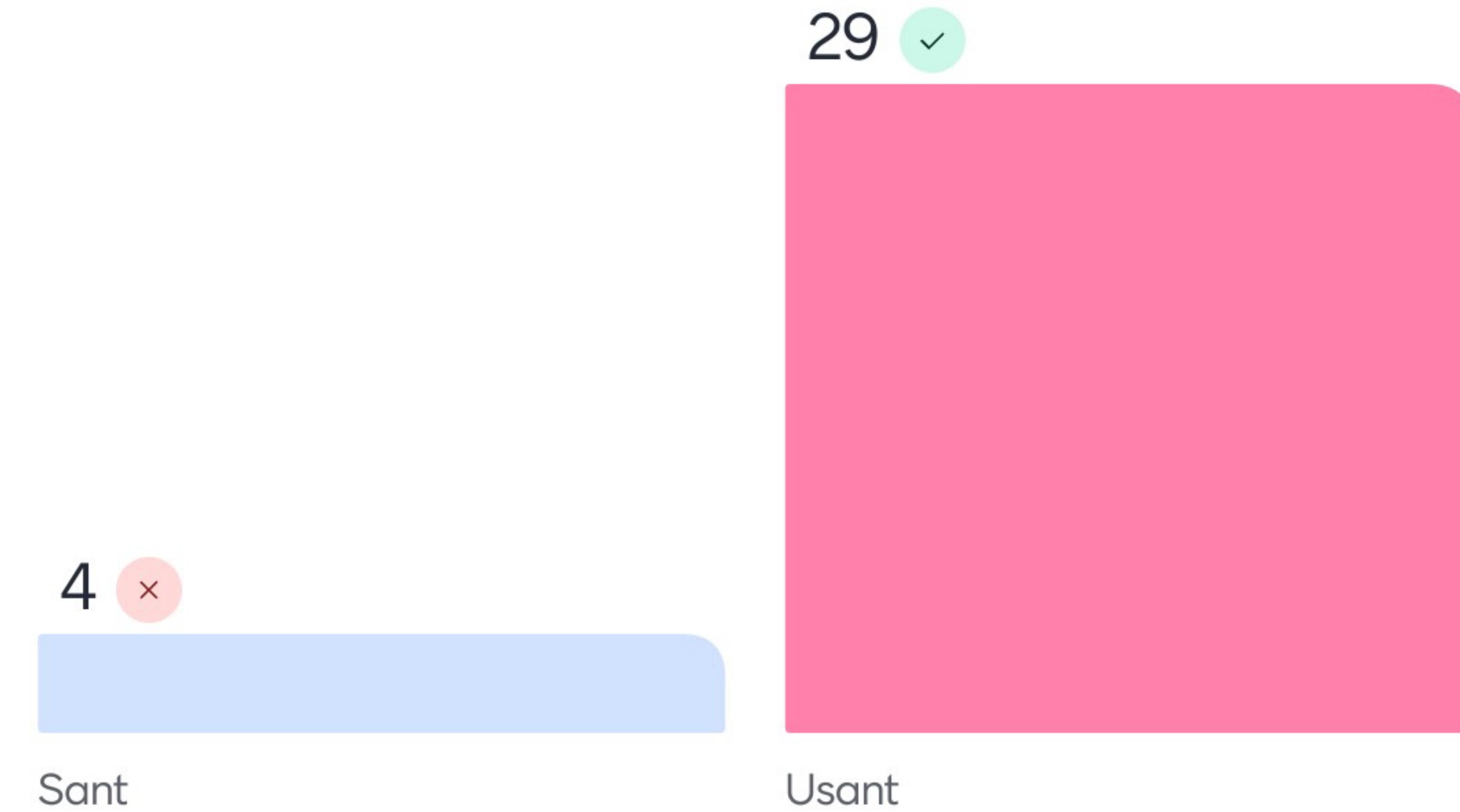
Er Radixsort in-place?



Er BucketSort in-place?



Bucketsort sammenligner elementer for å sortere



That's it folks

- Kattis oppgaver om sortering på emnesiden (grupperessurser)
- Gjerne jobb med andre greier og selvfølgelig

