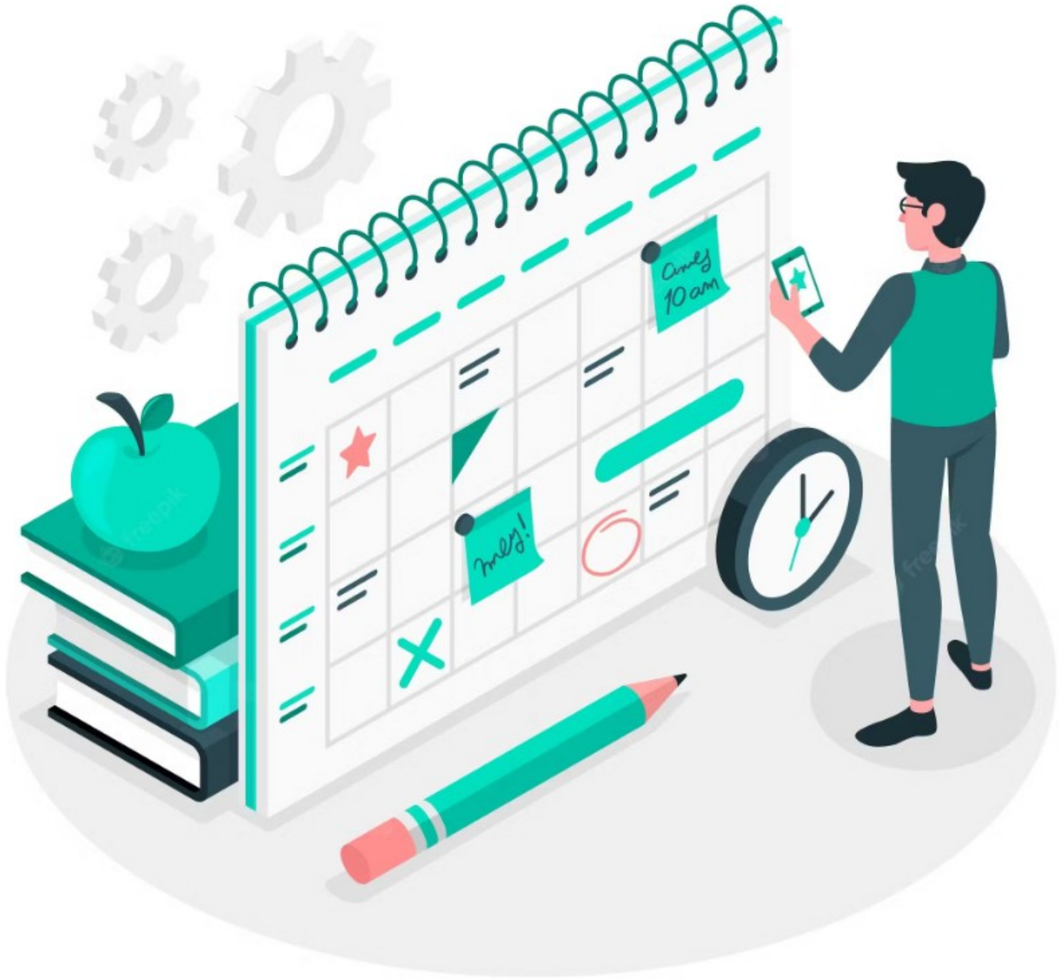
IN2010 Gruppe 4

Uke 11 - Sortering: Heap - Quick - Bucket - Radix

Mentimeter

Bli med :)

2



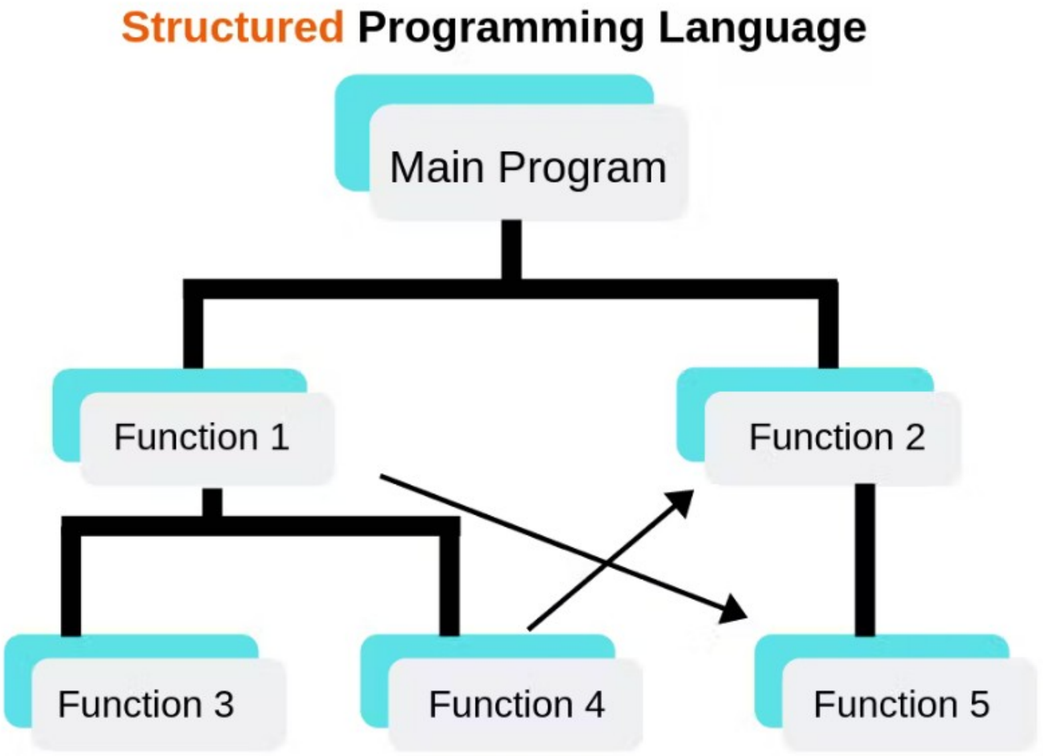
Dagens Plan

- → Info
- → Background/Recap
- → Pensumgjennomgang
- → Gruppeoppgaver

Siste uke med pensum

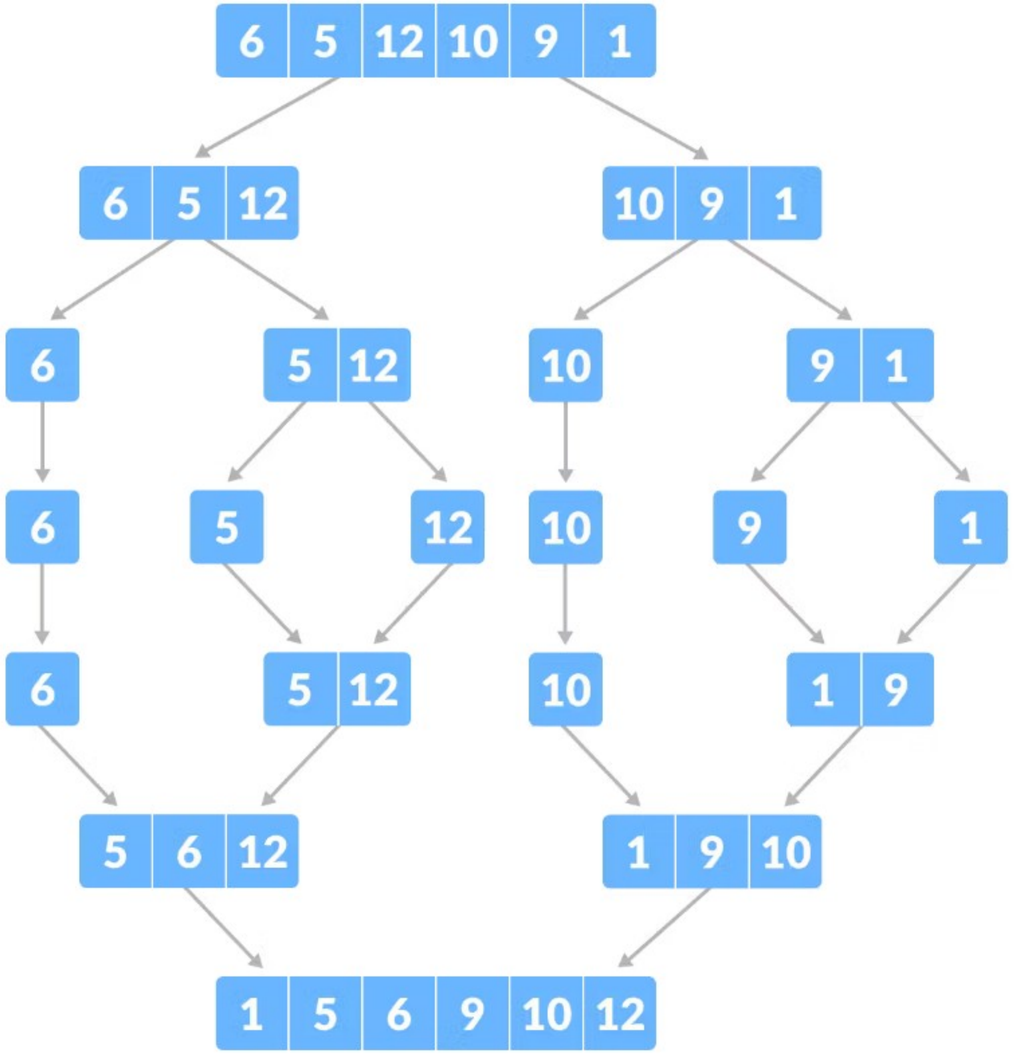
Bonuspensum er fortsatt pensum!

Background/Recap



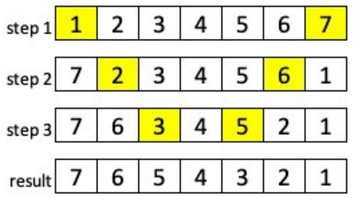
Måter å implementere algoritmer(Vanlig/intuisjon)

- En mer gripelig måte å løse oppgaver
- Gir bedre oversikt
- Kopiering av data og datastrukturer
- Bruk af flere datastrukturer
- Flere eksempel



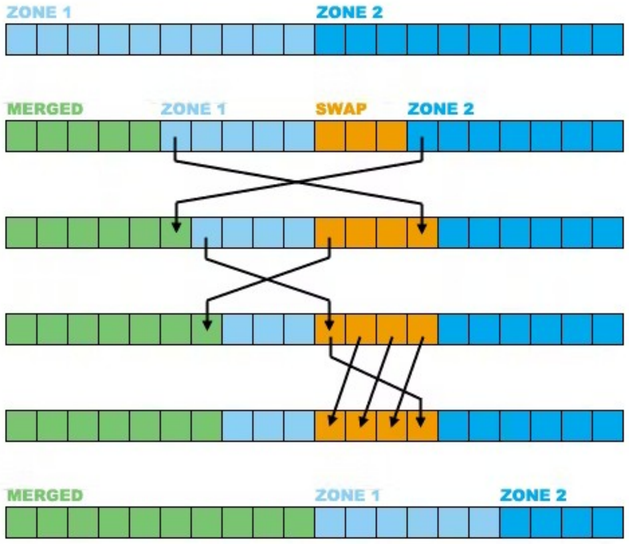
Merge-sort eksempel

- → Delte array i 2
- → Ved å lage nye arrays



In place

- Dette betyr at vi ikke lager nye datastrukturer(arrays)
- Men heller flytter på elementene innad i arrayet
- Selv på rekursive kall



Merge-sort eksempel

- → Pekere for sub-array 1 og 2
- → Ingeen nye arrays blir laget

```
def mergeSort(arr, l, r):
    if (l < r):

        # Same as (l + r) / 2, but avoids overflow
        # for large l and r
        m = l + (r - l) // 2

        # Sort first and second halves
        mergeSort(arr, l, m)
        mergeSort(arr, m + 1, r)

        merge(arr, l, m, r)
```

```
def merge(arr, start, mid, end):
    start2 = mid + 1

    # If the direct merge is already sorted
    if (arr[mid] <= arr[start2]):
        return

    # Two pointers to maintain start
    # of both arrays to merge
    while (start <= mid and start2 <= end):

        # If element 1 is in right place
        if (arr[start] <= arr[start2]):
            start += 1
        else:
            value = arr[start2]
            index = start2

            # Shift all the elements between element 1
            # element 2, right by 1.
            while (index != start):
                arr[index] = arr[index - 1]
                index -= 1

            arr[start] = value

            # Update all the pointers
            start += 1
            mid += 1
            start2 +
```

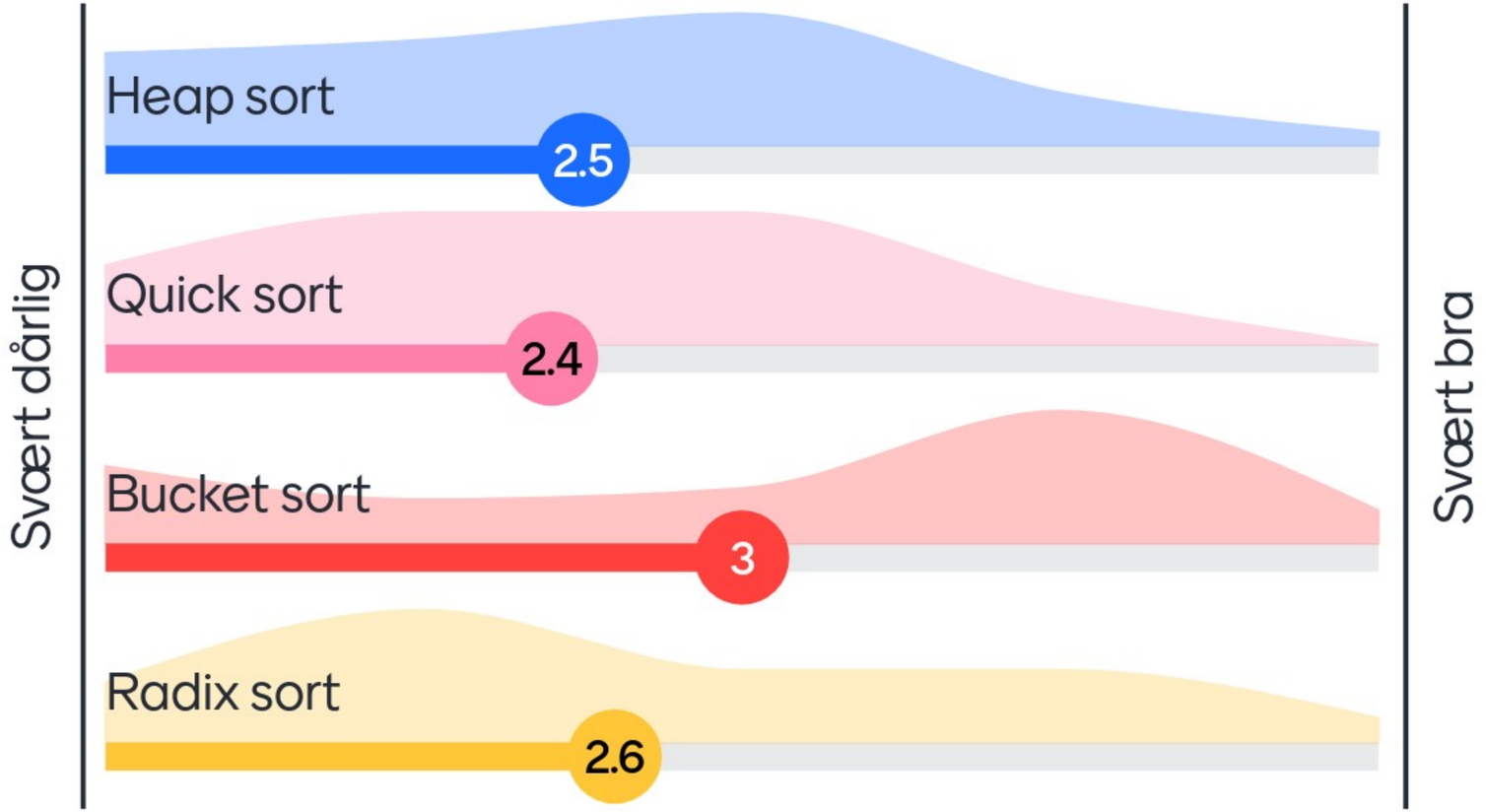
Hvorfor er dette viktig?

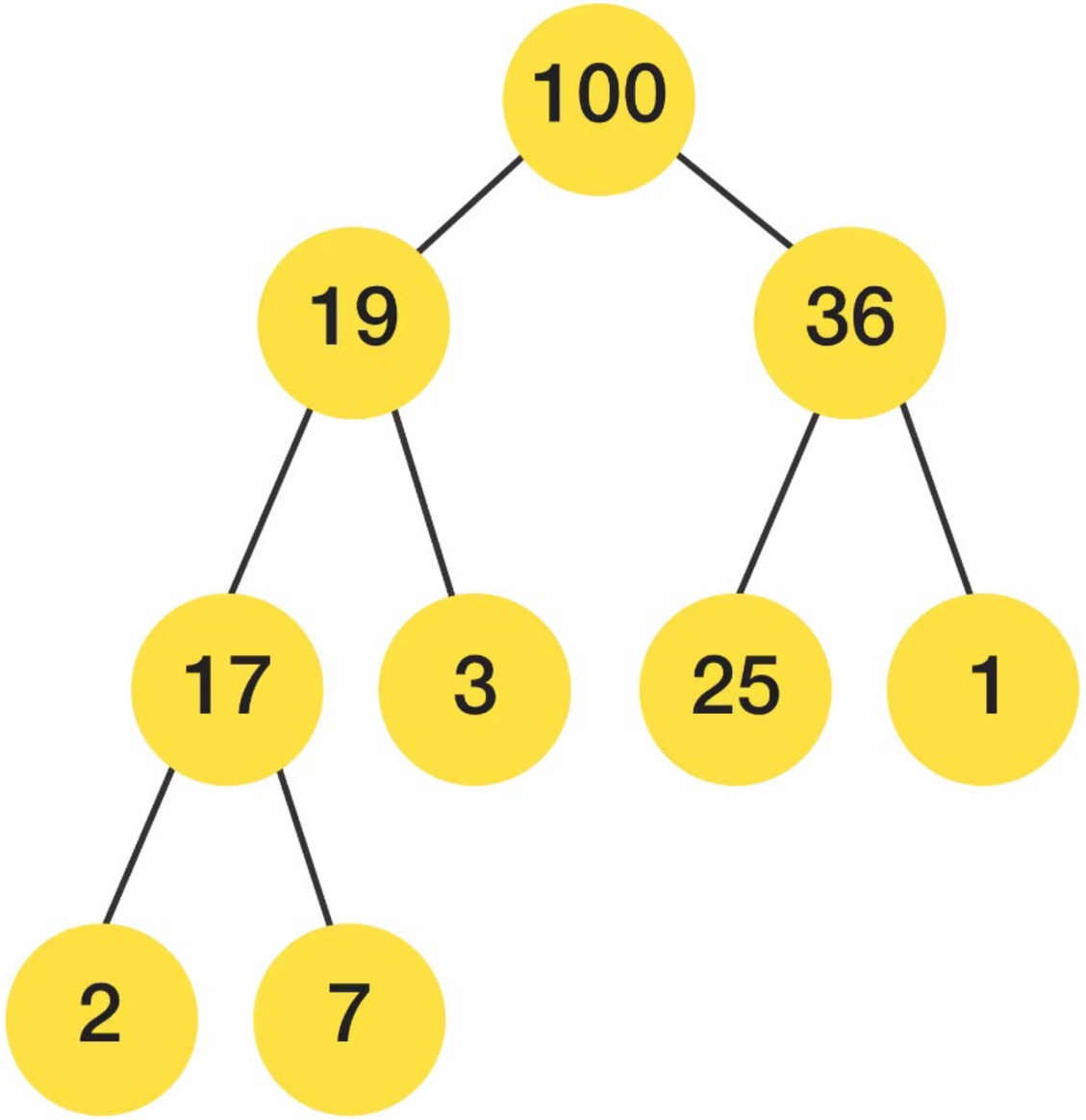
- Ukens pensum bruker in-place algoritmer(heap og quick sort)
- Viktig å kunne skille mellom in place og ikke in-place
- In-place er ofte litt vanskeligere å skjønne
- Begge kan brukes på eksamen(men må noteres dersom implementasjonen din ikke er fra forelesning)

Mentimeter

Pensumgjennomgang

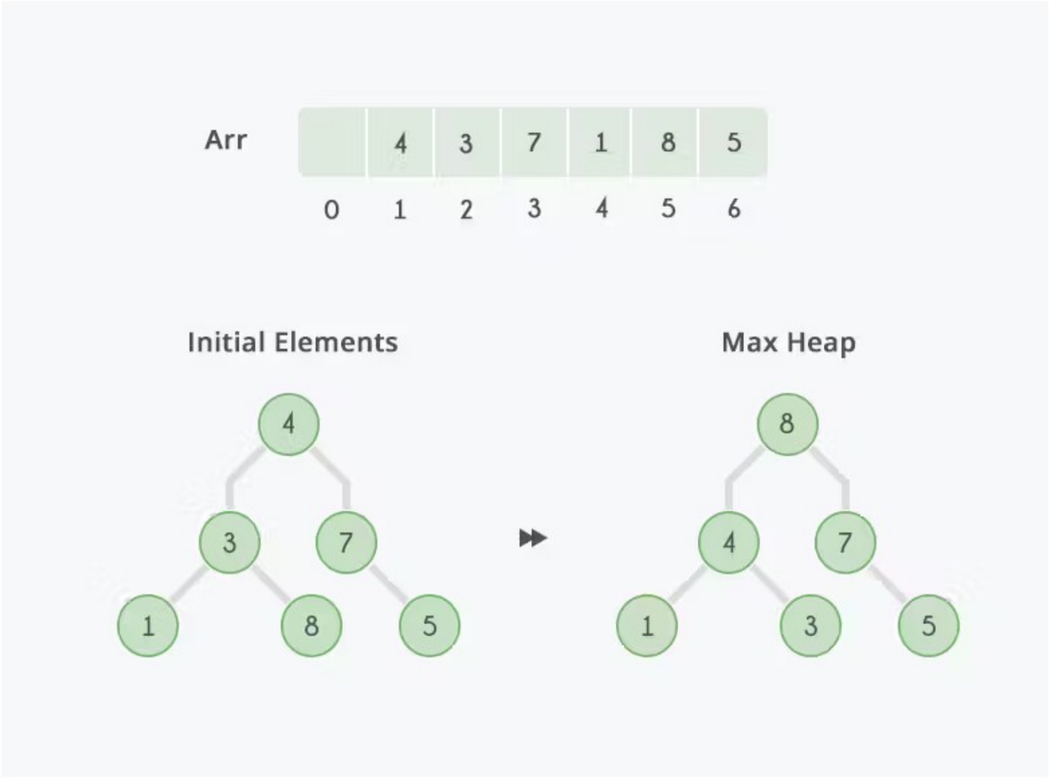
Hvor godt forsto du ukens pensum?





Heaps

- Egenskapene til en heap: Største element ligger i roten
- Bubble up og bubble down vil sørge for at største element alltid er på roten
- Ved Fjerning og innsetting



Heap sort

- Ide: Bygge en max-heap av et array
- Poppe elementene element for element
- In-place vs Ikke in-place

Heap-sort Ikke in-place

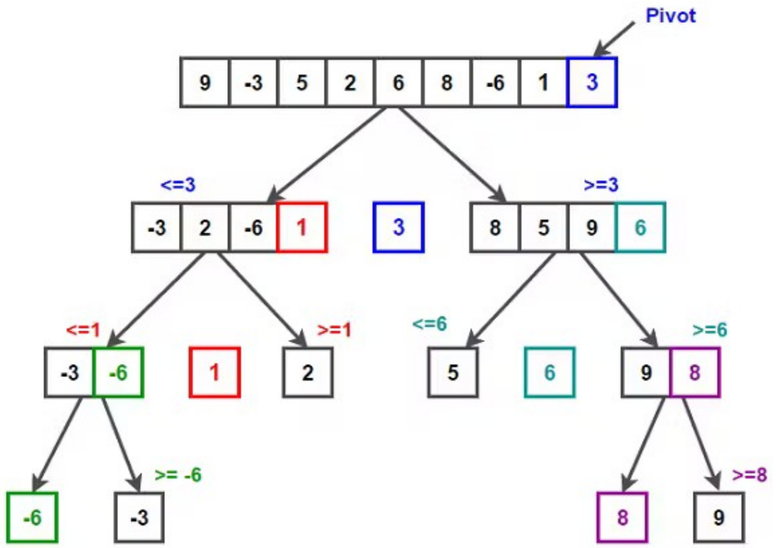
- Bygger en max heap
- Inisialiserer en tom array
- Popper største elementet fra heap og setter det inn i arrayet
- Arrayet fylles bak til frem

Heap-sort In-place

- Bruker samme array og gjør det om til en max heap
- Største element vil ligge på index 0
- Vi starter på siste node i heapen
- For hver node(bakover) så bytter vi plass med roten
- Og bubler down

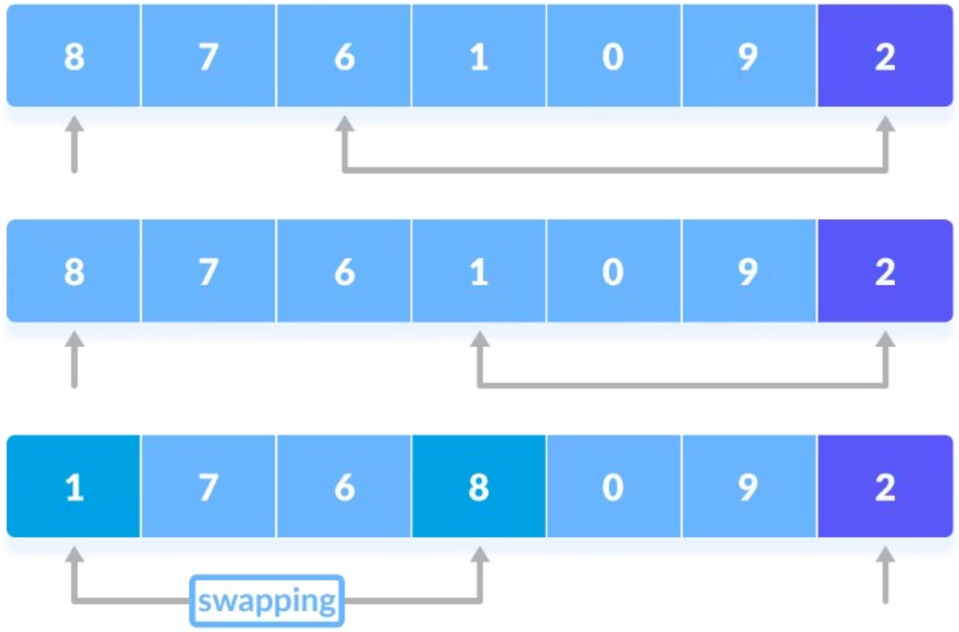
Quicksort

- Ide: Velge et element
- Sette alt som er mindre til venstre
- Alt som er større til høyre
- Elementet vi velger kalles en pivot
- Kan være elementet på index 0, n-1 eller n//2
- Elementene til venstre og høyre er nye "partisjoner"
- Så vi velger en pivot på de nye partisjonene rekursivt



Quick sort: Ikke in-place

- Når vi velger pivot så lager vi 2 nye arrays
- Velger pivots og partisjonerer elementene helt til vi har arrays på størrelse 1
- Resultatet vil være N arrays, der lementene er ordnet



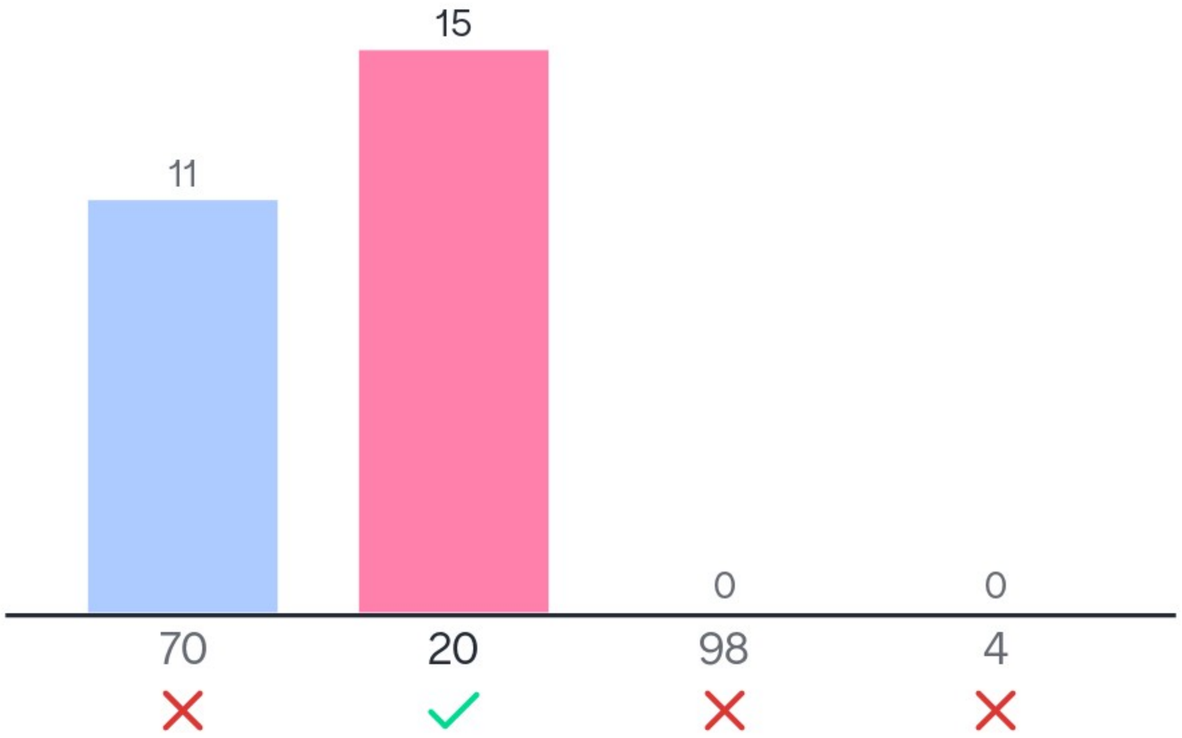
Quicksort: In-place

- Velger pivot
- Går fra venstre og høyre frem til de møtes
- mens elementene på venstre er mindre og elementet på høyre er større
- Når det stopper, bytt plass med left og right element
- Så fortsetter algoritmer

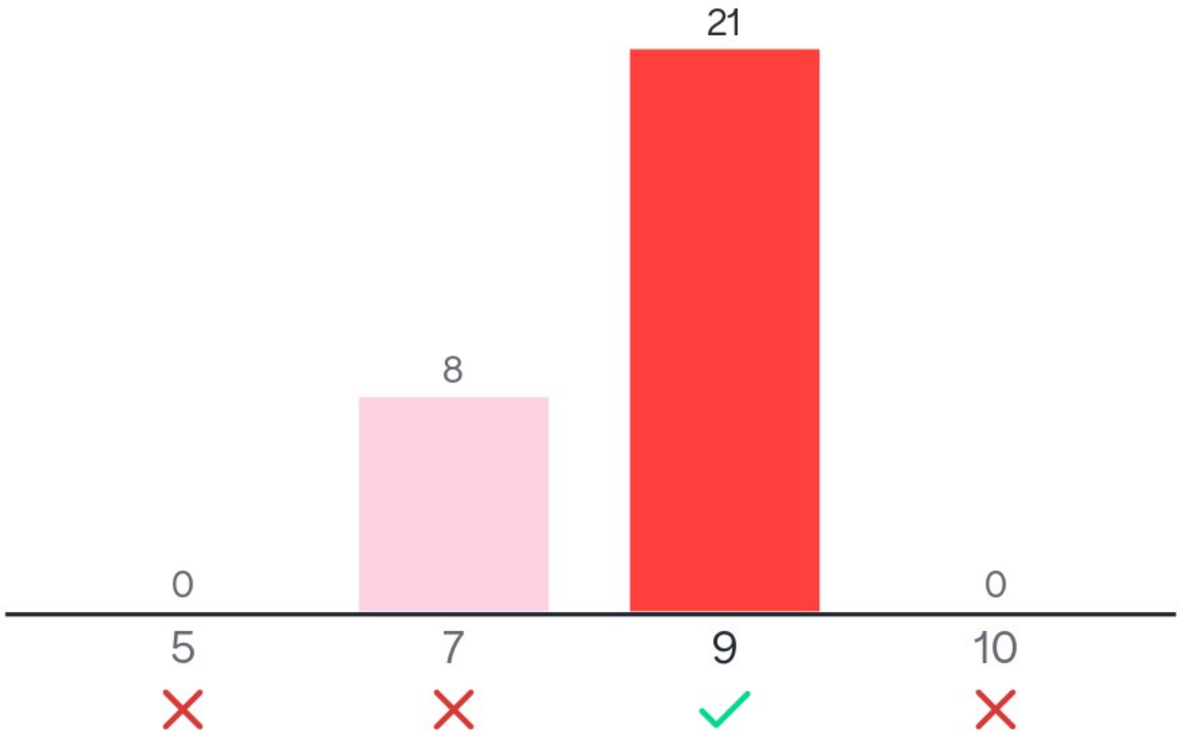
Mentimeter

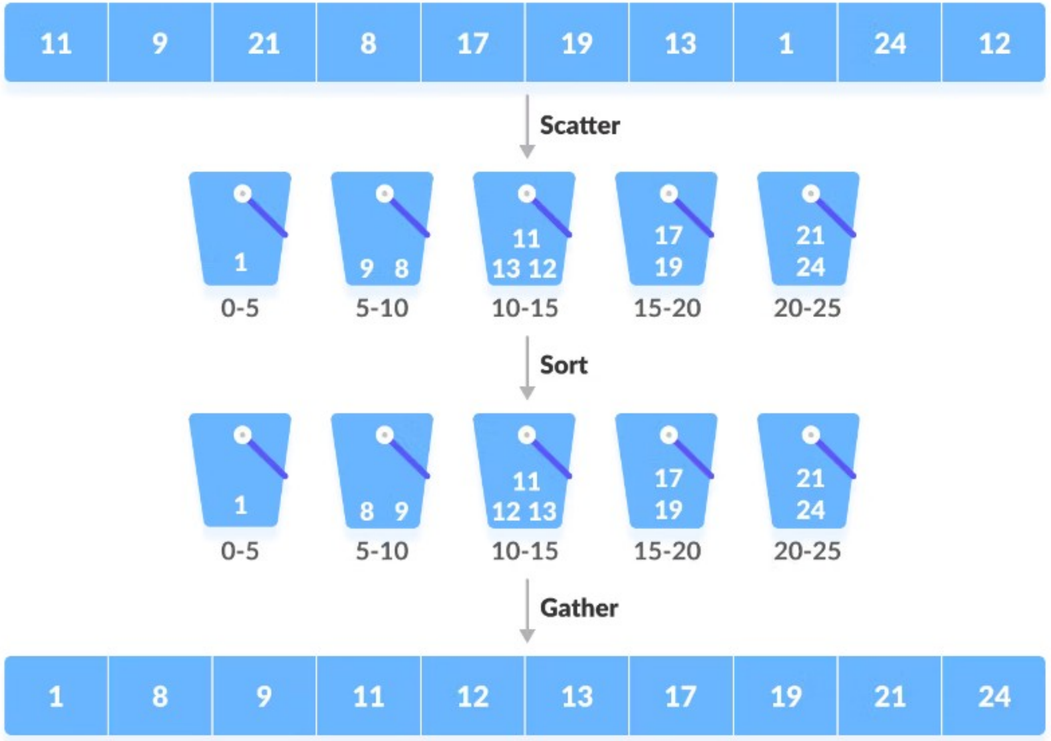
Quiz

Vi har følgende sekvens [11, 4, 20, 45, 32, 60, 98, 70] - Hvilket element ser ut til å være pivot?



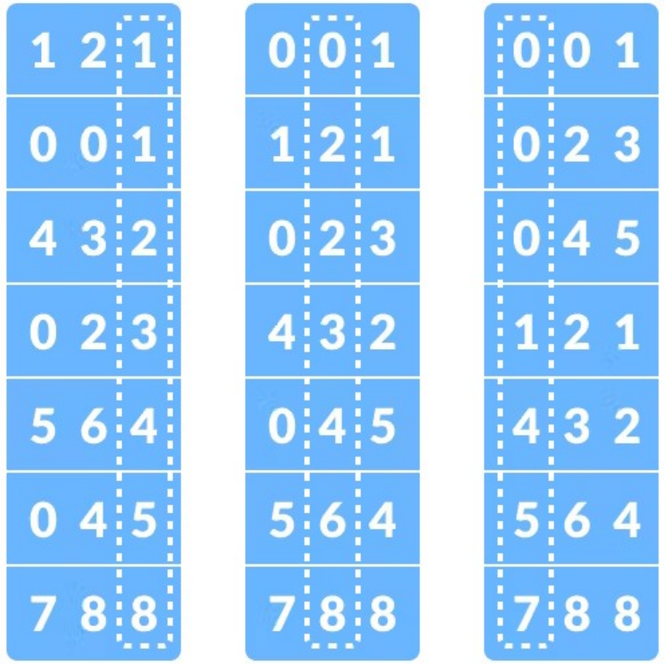
Vi har følgende sekvens [2, 5, 1, 7, 9, 12, 11, 10] - Hvilket element ser ut til å være pivot?





Bucket sort

- Ide: Lage N bøtter med kategorier
- Hvert element i listen går i en bøtte basert på sin kategori
- NB: Elementene må oftes sorteres innad i bøttene



sorting the integers according to units, tens and hundreds place digits

Radix sort

- Ide: Bucketsort der bøttene er "siffer"
- Lager bøttter med verdi 0-9
- Starter på siste siffer i hvert element og legger verdien i rett bøtte
- Går bakover til neste siffer, og gjør det for alle siffer

Mentimeter

Quiz

Om man skal sortere personer på deres buersdag(DD/MM/YYYY), Hvilken algoritme ville vært best å bruke? Hvorfor?

18 responses

Bogo	bucket sort	radix
radix	Radix	radix
radix	Radix	Radix sort :3

Om man skal sortere personer på deres buersdag(DD/MM/YYYY), Hvilken algoritme ville vært best å bruke? Hvorfor?

18 responses

Radix

Radix

RAAAADIX

hvorfor radix > sortere år, så måned, så dato

radix

Radixxxxxx

radix

radix

bucket

Mentimeter

Gruppeoppgaver

Stabilitet

Anta at arrayet A er usortert og inneholder personobjekter som alle har et felt for alder. Anta videre at personobjektene som ligger på A[3] og A[42] begge er 22 år gamle.

Arrayet A blir sortert etter alder. Etter sorteringen får du vite at:

- Personobjektet som lå på A[3] før sorterting, ligger nå på A[9]
- Personobjektet som lå på A[42] før sorterting, ligger nå på A[7]

(a) Var sorteringen stabil?

(b) Hva er alderen til personobjektet som ligger på A[8] etter sortering?

(c) Dersom du får vite at ingen personer i A er eldre enn 100 år gammel. Hvilken sorteringsalgoritme bør da benyttes, med hensyn til kjøretidseffektivitet?

Eksamen 2021