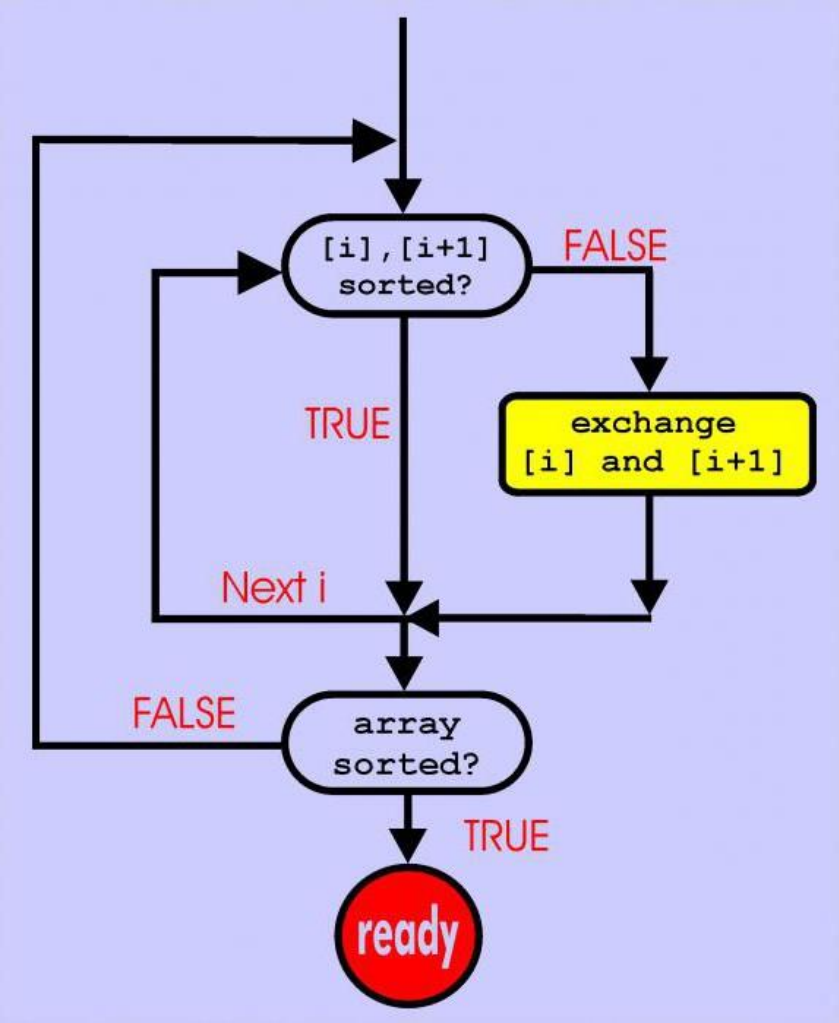
H.C. Ørsted Gymnasiet

2D Projekt

Sortering og Søgning

Programmering, 2022



Overordnet problemstilling

Sortering respektive Søgning af information er en fundamental og central opgave.
Mange opgaver er hurtigere i sorteret information (tænk på ordbøger, telefonbøger, adresselister i telefoner, . . .). Dette gælder både for mennesker og for computere. Sortering er ofte en byggesten i algoritmer for andre problemer og Søgning gør gentagne opslag meget hurtigere.

Rammer for projektet

I dette projekt skal I arbejde individuelt !!!
Projektet indgår i Mundtlig Årsprøve i Programmering 2022.
Vi bruger modulerne i ugerne 18 + 19 på at arbejde med projektet.

Projekt

Du skal redegøre for en række sorteringsalgoritmer samt søgealgoritmer. Herunder liste pseudokode eller kildekode. Du skal gøre rede for den enkelte algoritmes kompleksitet samt implementere algoritmen og fastslå køretiden for algoritmen (dvs. bestemme konstanten c).
Du skal overvej dit valg af Design af din Implementation !

Den færdige rapport (i pdf format) samt Java kildekode skal afleveres i zip-format i Lectio !!!
Deadline er: Søndag d. 15. Maj, kl. 21:00

Mål

- at I anvender Java som programmeringssprog
- at I anvender datalogiske metoder og teori
- at I benytter Lærebogen samt Internettet
- at I kan dokumentere herunder plotte sammenhænge på en overskuelig måde

Kernestof

Java, Algoritme, Klasser, Sortering, Søgning samt Design af Implementation og Dokumentation.

Projektopgave

1. Sortering.

Du skal arbejde med følgende algoritmer:

I. Insertionsort.

II. Mergesort.

III. Quicksort.

IV. Selectionsort.

2. Søgning.

Du skal arbejde med følgende algoritmer:

1) Sekventiel søgning.
2) Binær søgning.

3. Data.

Du skal tidsmåle dine algoritmer på forskellige samlinger af data der hver indeholder N elementer.

Tidsenheden bør vælges så også N=10 giver en værdi større end nul. Du kan fastlægge konstanten c(N) idet der gælder T(N) = c(N) * kompleksiteten. Til sidst kan du beregne et vægtet gennemsnit og fastlægge een enkelt c værdi for hver algoritme.

a) N = 10.
b) N = 50.
c) N = 200.
d) N = 500.
e) N = 1000.
f) N = 2000.

- Dine data skal være i tilfældig orden.
- Dine data skal allerede være sorteret i stigende orden.
- Dine data skal allerede være sorteret i aftagende orden.

For at kunne sammenligne resultater algoritmer imellem, SKAL du benytte Seed i random() funktionen. Herved vil dine tilfældige data være de samme hver gang du kører din kode.
Du vælger selv om data skal være heltal eller flydende tal.

4. Dokumentation.

Du skal overvej dit valg af Design af din Implementation !!!

Benyt plots samt tabeller til at sammenligne algoritmerne i de to typer, Sortering resp. Søgning.

5. Konklusioner

1. Hvilken sorteringsalgoritme er den bedste teoretisk og hvilken er den hurtigste på din maskine (dvs. hvilken har den bedste kompleksitet og den mindste konstant c) ?
2. Hvilke problemer kan der være med tidtagning på din maskine ?
3. Hvilke problemer / fordele er der ved brug af heltal versus flydende tal ?
4. Hvad ønsker du ellers at fortælle ?
   // end