# Oppgave1:

Følgende 4 algoritme blir implementert:

1. Insertion sort: $O(n^2)$

2. Quick sort: O(nlog(n)) vanligvis, og $O(n^2)$ i verste tilfellet

3. Selection sort: $O(n^2)$

4. Merge sort: O(nlog(n))

Gjennom output filene (som skal være an sortert array) kan jeg kjekke om algoritmene gir riktig svar eller ikke. Og sammtidig skal antallet swap og antallet compares for hver algoritme være rimelig.

# Oppgave2:

Bytter og sammenligninger er målt ved bruk av funksjonene CountSwaps og CountCompares (som er fra prekoden).

**CountSwaps :**

I sort funksjonene blir listen gjort som en CountSwaps instance. For hver algoritme, ved å kalle metoden swap på en CountSwaps instance, øker variablen swap med en.

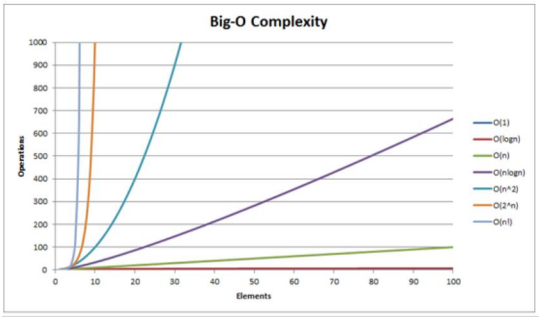
**CountCompares :**

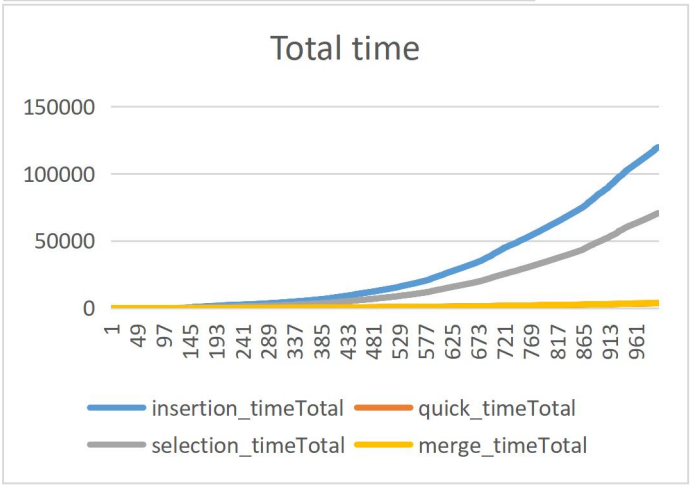
For hver element i listen, blir det opprettet en CountCompares instance. Hver gang når elementene/instance blir sammenlignet, øker variablen compares med en.

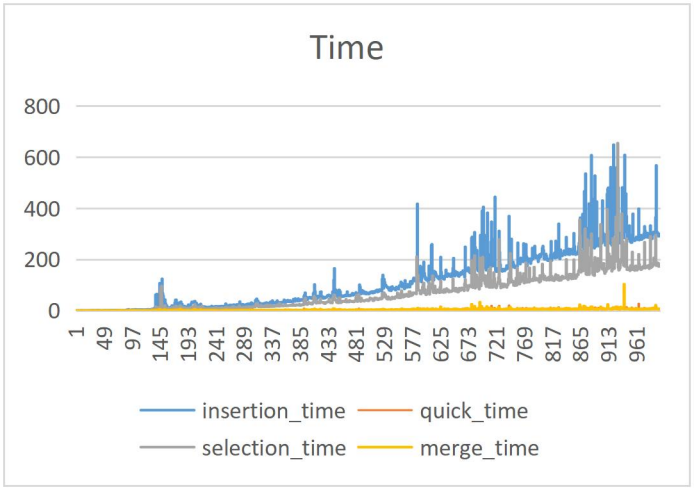
# Oppgave 3:

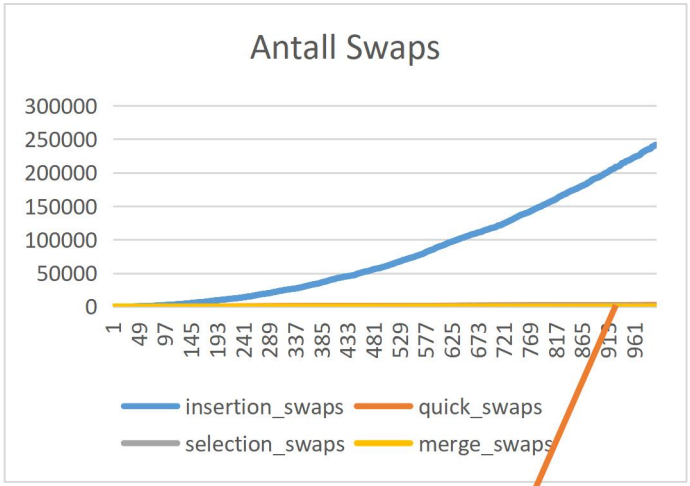
Følgende er tablene for total kjøretid, kjøretid, antall swaps of antall compares når n=1000.

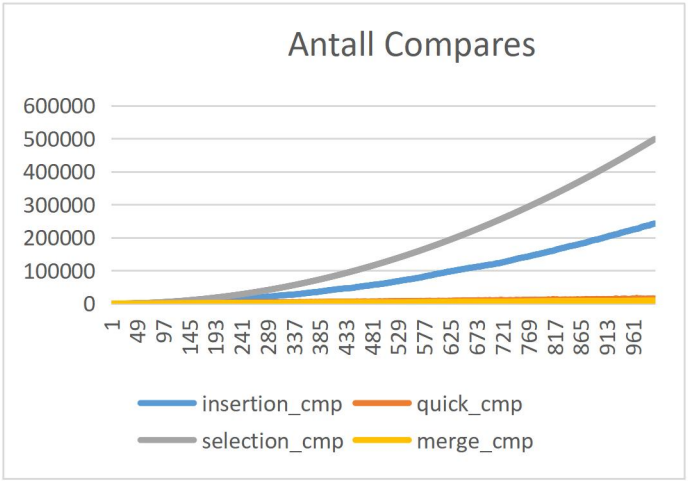
Samt er det en Big-O Complexity tabel som hentet fra nettet.

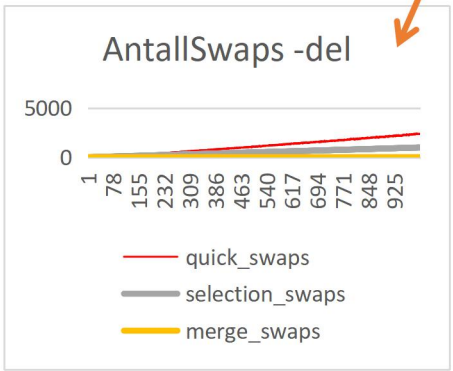


### (1) I hvilken grad stemmer kjøretiden overens med kjøretidsanalysene (store O) for de ulike algoritmene?

a) Insertionsort og selectionsort:

I følge kjøretidsanalysene skal big O for insertionsort og selectionsort være **$\underline{O(n^2)}$**

Men fra tabel <Total time>, <Time> og < Big-O Complexity> kan vi se at **insertionsort og selectionsort er nærme kurvet for O(nlog(n)).**

b) Quicksort og mergesort:

I følge kjøretidsanalysene skal big O for quicksort og mergesort være **$\underline{O(nlog(n))}$**

Men fra tabel <Total time> og < Big-O Complexity> kan vi se at **quicksort og mergesort er mellom kurvet for O(log(n)) og kurvet for O(n).**

Konklusion:

**Det betyr at kjøretiden stemmer ikke veldig godt med kjøretidsanalysene for alle 4 algoritmene.**

### (2) Hvordan er antall sammenligninger og antall bytter korrelert med kjøre-tiden?

Fra tabel <Antall Swaps>, <Antall Compares> og <Time> kan vi se at:

**Antall swap:** Insertionsort har mye høyere antall swaps enn de andre tre.

**Antall compares:** Selectionsort har største antall compares. Insertionsort har lavere antall compares. Og antall compares for quicksort og mergesort er veldig lav.

**Kjøretiden for insertionsort** er største, og den har største antall swaps (mye større enn de andre tre) og andre største antall compares.

**Kjøretiden for selection** sort er den andre største, og den har største antall compares.

Det vil si at insertionsort og selectionsort har større antall swaps / antall compares, og disse to algoritme har større kjøretid.

**Konklusion:** Jo større antall swaps/antall compares er, jo større er kjøretid.

### (3) Hvilke sorteringsalgoritmer utmerker seg positivt når n er veldig liten? Og når n er veldig stor?

Alle fire algoritme er veldig raske når n er veldig liten. I følge resultatet jeg har fått, når n<=20, kjøretid på alle fire algoritme er 0.

Fra tabel <Total time> og <Time> kan vi se at quicksort og mergesort er mye raskere når n er veldig stor.

### (4) Hvilke sorteringsalgoritmer utmerker seg positivt for de ulike inputfilene?

I følge resultatene fra ulike inputfilene, ligger kjøretid for mergesort og quicksort alltid lavt, dvs at disse to algoritmer utmerker seg positivt for de ulike inputfilene.

### (5) Har du noen overraskende funn å rapportere?

Fra tabel <Time> kan vi se at kjøretid i noen steder blir pluselig veldig høyt for insertionsort og selectionsort.

Jeg har forventet at kurvene kan være litt 'ustabil'/'ujevn' siden det har noe å si med beste tilfellet og verste tilfellet, met et annet ord tilfeldighet. Men det fortsatt overrasker meg når det er så 'vodsomt ujevn'.