Deloppgave 1: Korrekthet

Vi implementerte sortingsalgoritmene: Insertion sort, Quick sort, Gnome sort og Heap sort.

Hva vi har testet for å sjekke at alle algoritmene gir rimelige svar er at vi har printet ut de sorterte arrayene for hver av sorteringsalgoritmene i terminalen, og vi sjekket om resultatet av sorteringen var likt for alle algoritmene og fikk true:

```
print(heapSorted[0] == gnomeSorted[0] == insertionSorted[0] == quickSorted[0] == sorted(arr[:x]))
```

Deloppgave 2: Sammenligninger, bytter og tid

Vi har målt sammenligninger og bytter ved å implementere variabler som vi inkrementerer når vi gjennomfører diverse sammenligninger og bytter i arrayet. Disse variablene skrives dermed ut til en CSV-fil for å analysere om det er riktig, ved hjelp av regnearksverktøy som Excel for å få bedre oversikt over verdiene til variablene over tid, inkludert å regne gjennomsnitt. Vi sammenlignet dermed disse verdiene for å sjekke om verdiene er rimelige.

Deloppgave 3: Eksperimentér

GJENNOMSNITTSVERDIER random_100

insert_cmp	insert_swaps	insert time	gnome_cmp	gnome_swaps	gnome_time
1761	881	157	5421	881	281

quick_cmp	quick_swaps	quick_time	heap_cmp	heap_swaps	heap_time
122	14	78	489	239	191

GJENNOMSNITTSVERDIER nearly_sorted_100

insert_cmp	insert_swaps	insert time	gnome_cmp	gnome_swaps	gnome_time
41	21	8	271	21	15

quick_cmp	quick_swaps	quick_time	heap_cmp	heap_swaps	heap_time
100	2	106	367	239	145

- Det er en viss sammenheng mellom O-notasjon og tiden brukt for hver sorteringsalgoritme med unntak av Quick sort som var vesentlig raskere til tross for å ha en kjøretidskompleksitet på O(n^2) i verste tilfelle.
- Som regel vil antall sammenligninger og bytter øke proporsjonalt tilsvarende størrelsen på kjøretiden for de ulike sorteringsalgoritmene. Hvis en algoritme har en dårlig kjøretid, vil antall sammenligninger og bytter ofte være relativt høyt, i motsetning til hvis algoritmen har bra kjøretid, hvor antallet vil være relativt lavt.
- Gnome sort blir veldig treig når n øker/er veldig stor, imens de andre sorteringsalgoritmene forblir relativt effektive når n blir stor. Insertion virker til å utmerke seg positivt når n er veldig liten, enda mer når den sorterer et nesten sortert array.

- Basert på de forskjellige eksperimentene/testene vi har utført for sorteringsalgoritmene som vi har implementert, utmerker insertion sort seg positivt for filene: nearly_sorted og når n er liten. Quick sort utmerker seg positivt for filene: random og når n er stor. Gnome sort utmerker seg positivt for filene: nearly_sorted uavhengig av hvor stor n er. Heap sort utmerker seg positivt for filene: random og når n er stor.

- Det er overraskende hvor stor forskjell det er angående sammenligninger og bytter imellom de ulike sorteringsalgoritmene, men dette kan skyldes en litt unøyaktig implementasjon av tellingen for sammenligninger og bytter.