Korrekthet

Oppgaven: Du skal redgjøre for hvilke sorteringsalgoritmer som er implementert, og hva som er testet for å sjekke at alle algoritmene gir rimelige svar.

Svar: Jeg har implementert algoritmene bucketsort, heapsort, insertionsort og mergesort. Jeg har testet programmet algoritmene mot eksempel output fra oppgaveteksten og alle algoritmene får sorterer riktig. Jeg har også testet at alle algoritmene fungerer mot alle inputer fra ressurssiden for Sortering: https://github.uio.no/IN2010/sortering-ressursside/tree/master/inputs

Sammenligninger, bytter og tid

Se diagrammene nedenfor for mer informasjon.

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

Svar: Bucketsort: Teoretisk kjøretid O(n). Passer best for data som er jevnt fordelt over et jevnt intervall. I praksis er den effektiv særlig på store mengden data med en jevn fordeling.

Hepasort: Teoretisk kjøretid O(n*log(n)). Stemmer bra med resultatene fra diagrammene som viser jevnt god ytelse uansett dataens orden.

Insertionsort: Teoretisk kjøretid $O(n^2)$. Har bedre ytelse på mindre lister og nesten sorterte lister, sammenlignet med mer usorterte og større lister. Se diagrammene.

Mergesort: Teoretisk kjøretid: O(n log(n)). Den gir stabile resultater nesten alltid, noe som betyr at den bevarer rekkefølgen for like elementer.

• ***Hvordan er antall sammenligninger og antall bytter korrelert med kjøretiden?***

Svar: Bucketsort: Har relativt få sammenligninger og bytter når dataen er jevnt fordelt. Siden kjøretiden er O(n) vil den være effektiv med store mengder data.

Heapsort: Har en moderat mengde sammenligninger og bytter, men holder seg generelt nær O(n*log(n)) både i sammenligner og bytter.

Insertionsort: Har mange sammenligninger og bytter for store og ustrukturerte data $O(n^2)$. For små lister eller nesten sorterte lister, reduserer antall operasjoner som gir raskere kjøretid.

Mergesort: Har mange sammenligninger (men ikke bytter), som gir en O(n*log(n)) ytelse i kjøretid.

• ***Hvilke sorteringsalgoritmer utmerker seg positivt når n er veldig liten? Og når n er veldig stor?***

Svar: Liten n: Insertionsort: Svært effektiv for små lister pga sin enkle struktur.

Stor n: Mergesort og heapsort er stabile pga de har O(n*log(n)) som god ytelse uavhengig av dataens spesifikke orden.

Bucketsort kan være effektiv hvis dataen en jevnt fordelt.

• ***Hvilke sorteringsalgoritmer utmerker seg positivt for de ulike inputfilene?***

Svar: Nesten sorterte file: Insertionsort utmerket seg pga færre sammelinger og bytter.

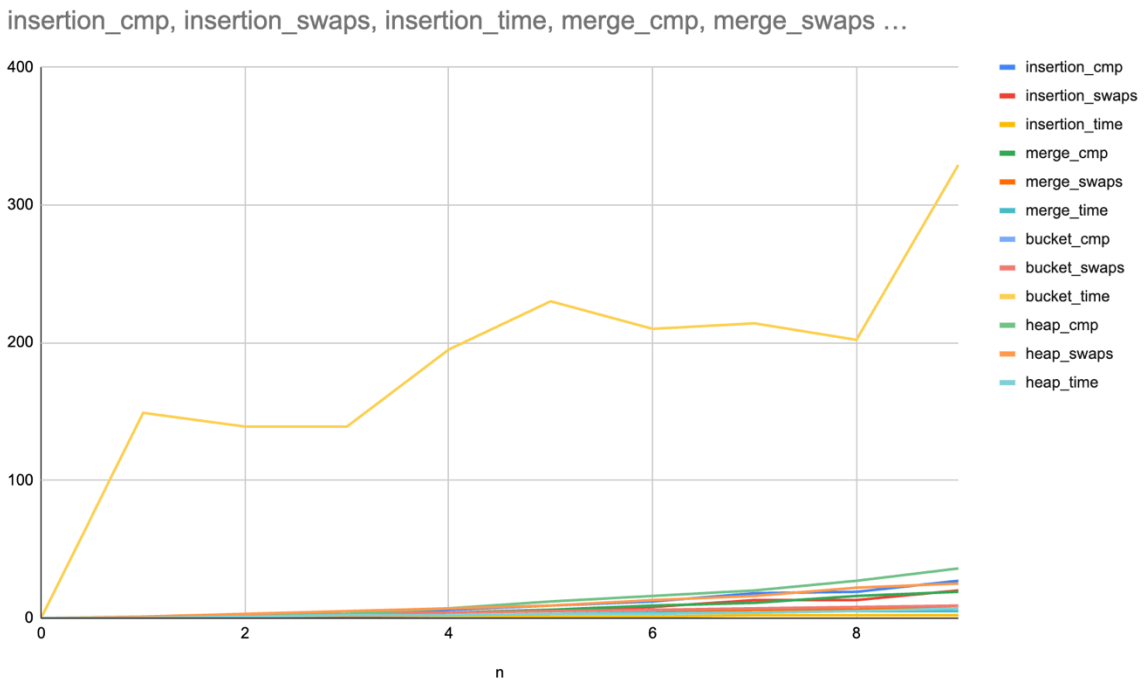
Usorterte file: Mergesort og Heapsort gir stabil ytelse uavhengig av hvilken ordning innputten har.

Lite antall elementer i filer: Insertionsort prestrer bra med lite antall elementer.

Større antall elementer: Mergesort og Heapsort gir høy ytelse pga deres effektivitet for store datasett. Bukcetsort er effektiv hvis dataen er jevnt fordelt.

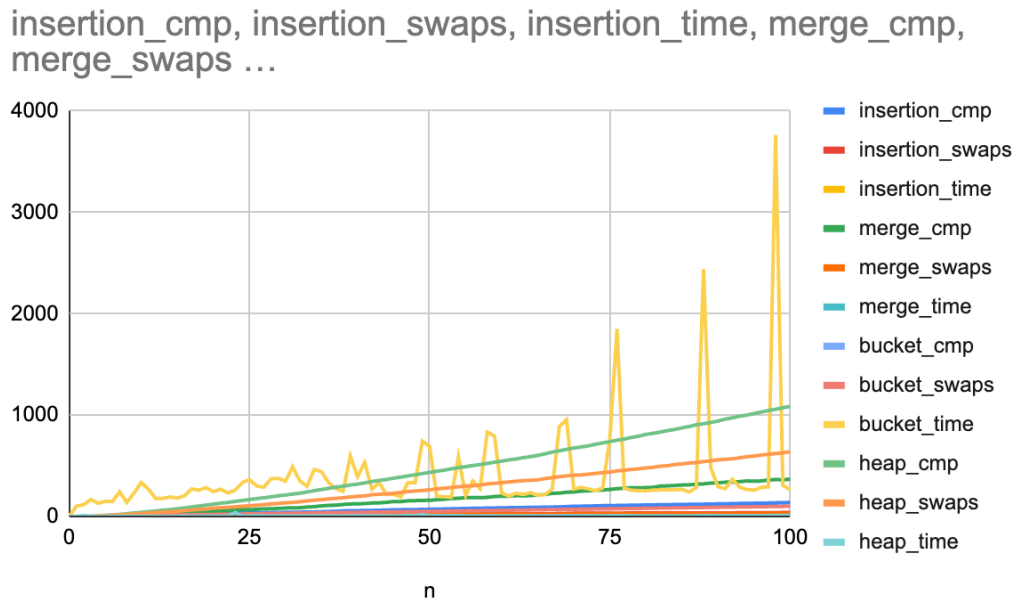
• ***Har du noen overraskende funn å rapportere?***

Jeg hadde ikke forventet at insertionsort skulle være like effektiv som den var på nesten sorterte dater.

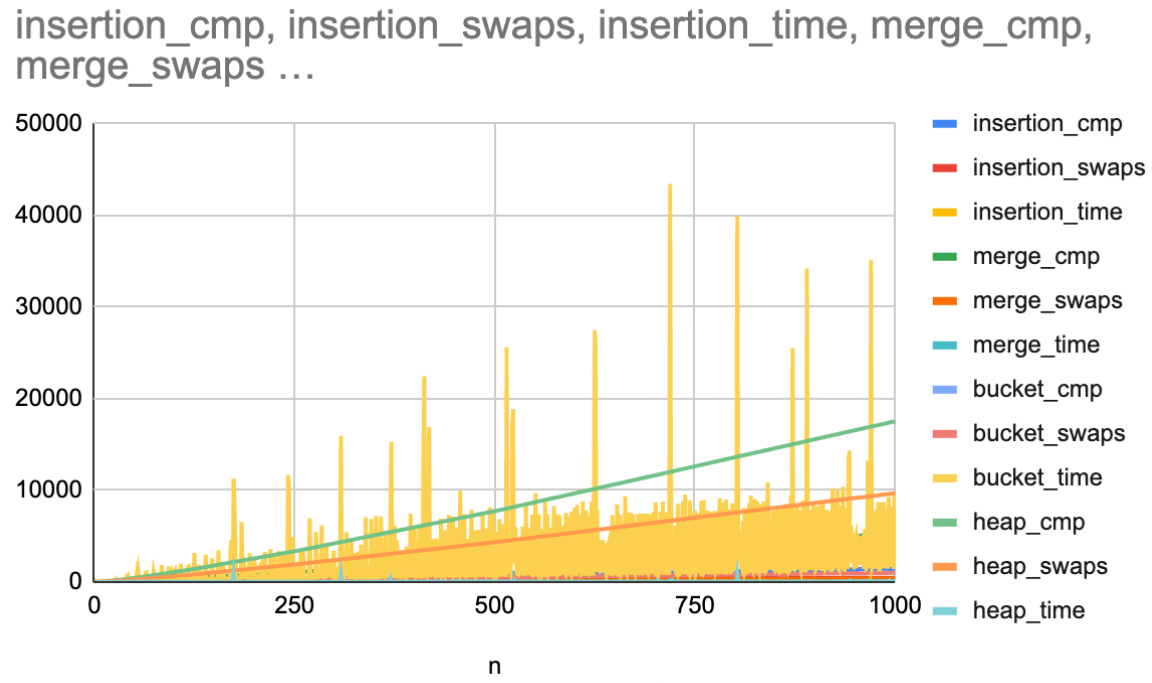


Eksempelfil fra oppgaveteksten med 9 elementer.

Nearly sorted

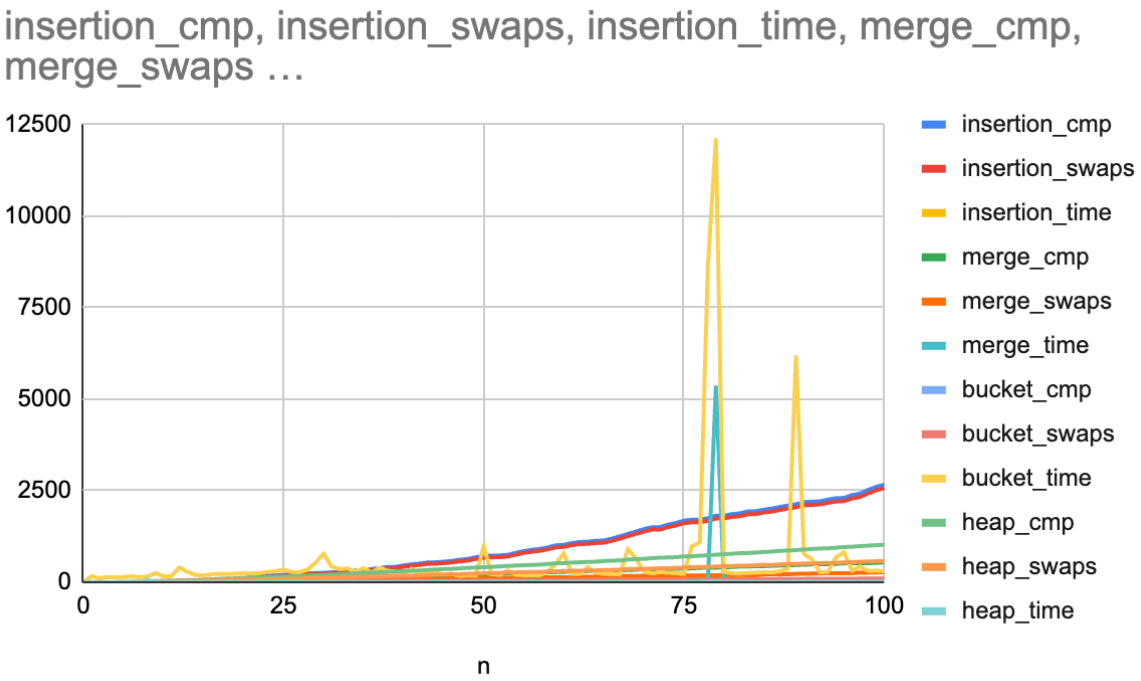


nearly_sorted_100.txt

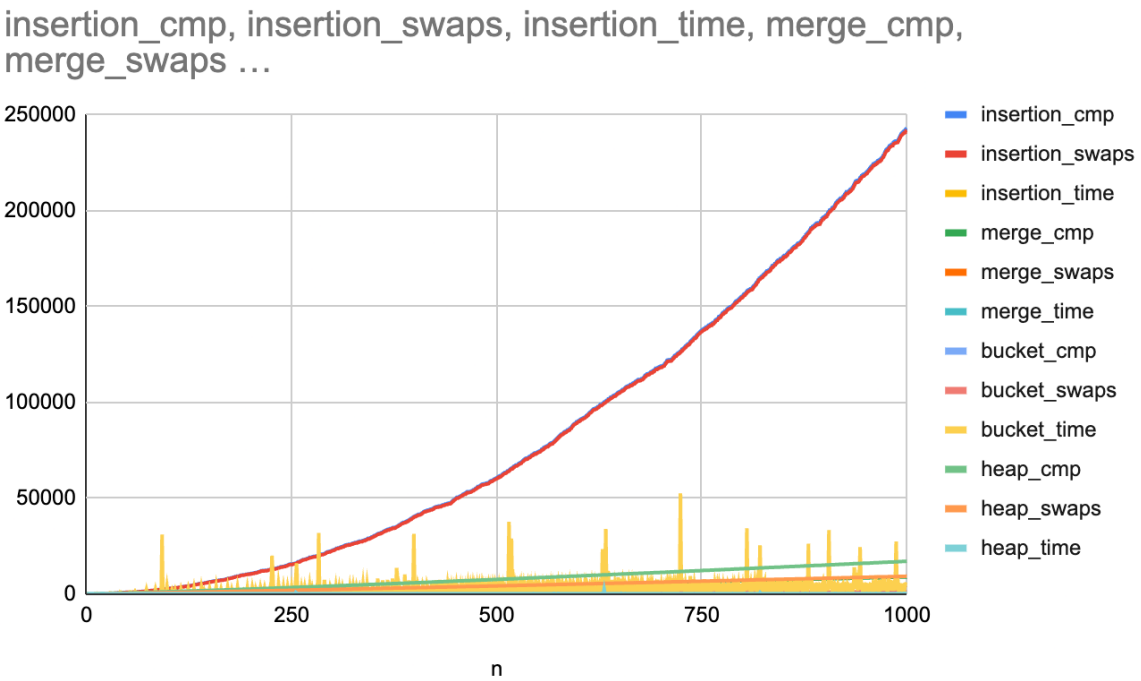


nearly_sorted_1000

Random



random_100.txt



Random_1000.txt