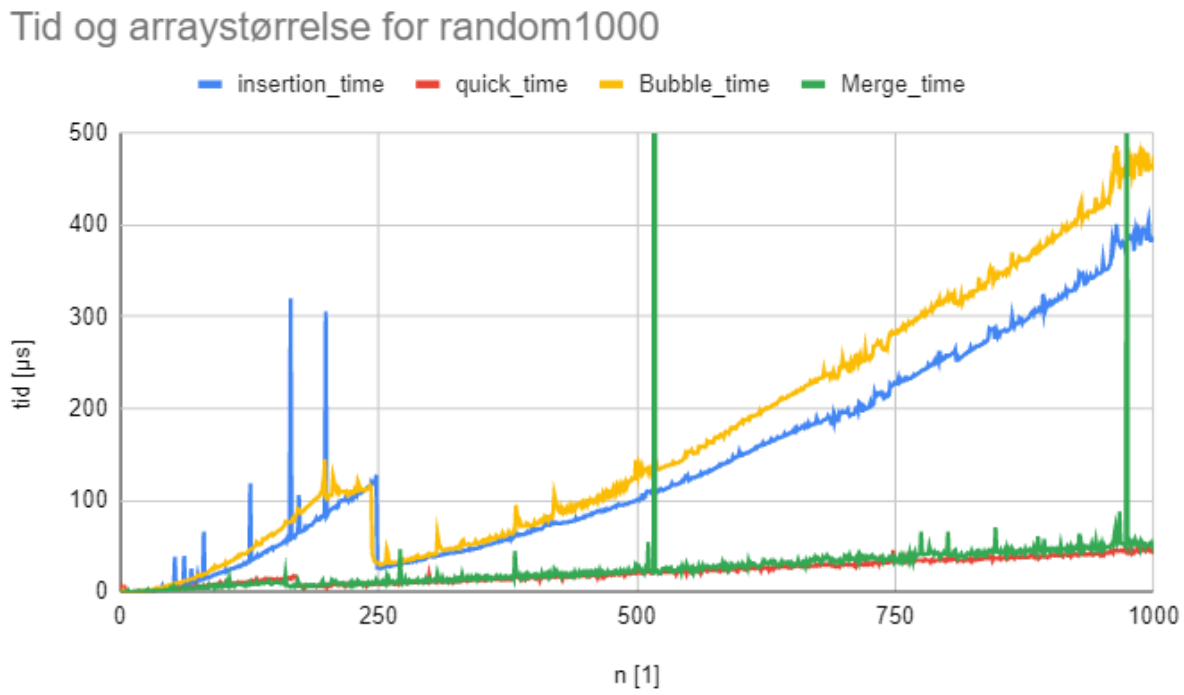
Oppgave 3:

Vi ser her en graf som sammenligner kjøretiden til de forskjellige sorteringsalgoritmene med størrelsen på arrayet n:

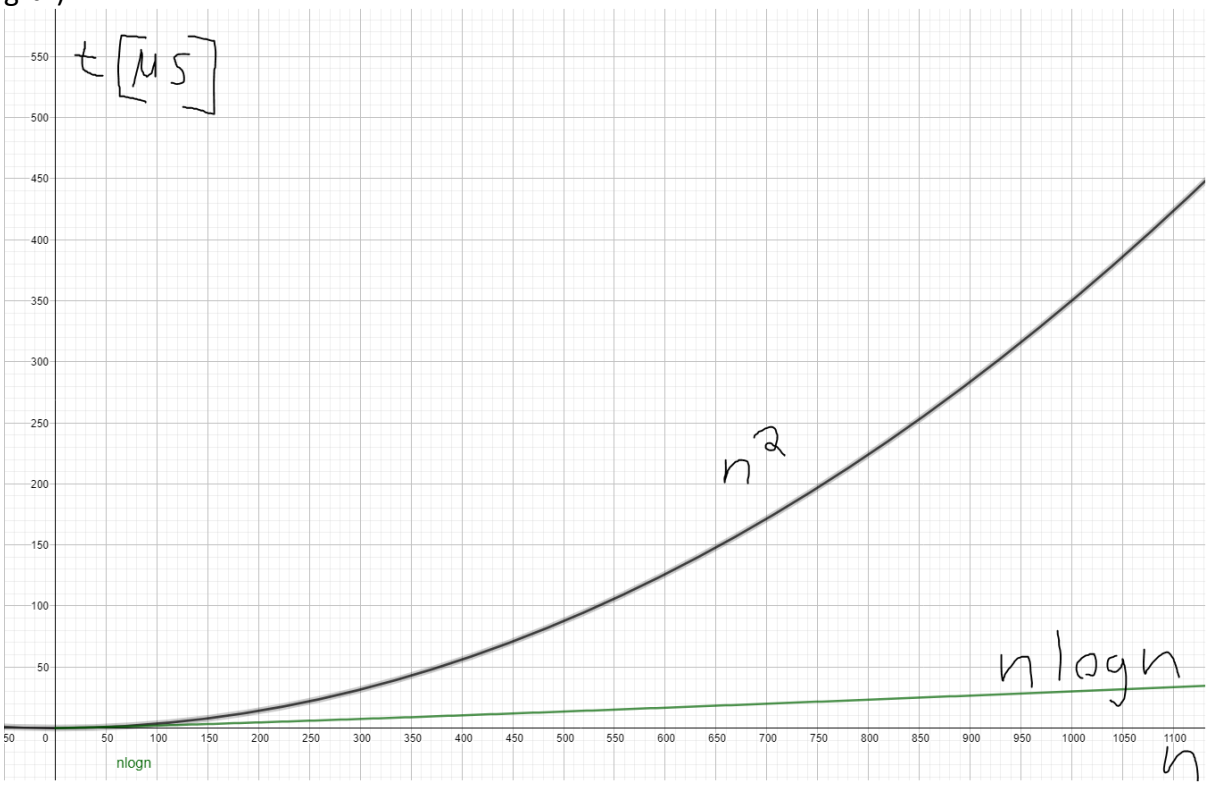


Et par ting å merke er at hvis vi sammenligner med kjøretidsanalyse, så ser vi i grafen vi lager i GeoGebra med funksjonsuttrykk y(x) = x*log(x), og g(x) = x^2 at det ikke virker som om kjøretidenskompleksiteten stemmer.

Geogebra representasjon av funksjonene n^2 (svart graf) og n log n (grønn graf)

Vi ser spesielt at vi ikke får opp noe som ligner på n^2. Derimot så stemmer ikke kjøretiden til sorteringsalgoritmene perfekt med funksjonssuttrykkene fordi vi kan gjøre flere operasjoner, innsettinger og andre steg som kan endre på selve stigningstallet. Ganger vi uttrykket til x*log(x) med en konstant 0.01, og uttrykket til x^2 med en annen konstant 0.0035 så får vi noe som ligner mer på våre resultater.

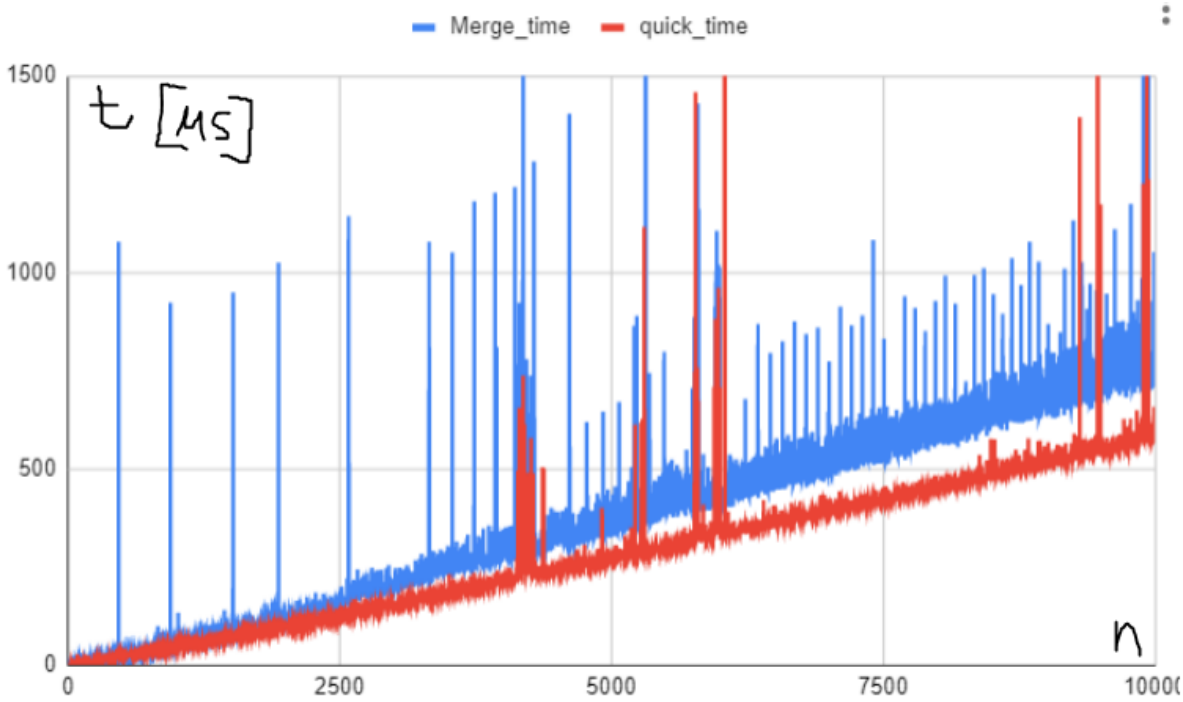
Geogebra representasjon av funksjonene 0.00035*n^2 (svart graf) og 0.01*n log n (grønn graf)



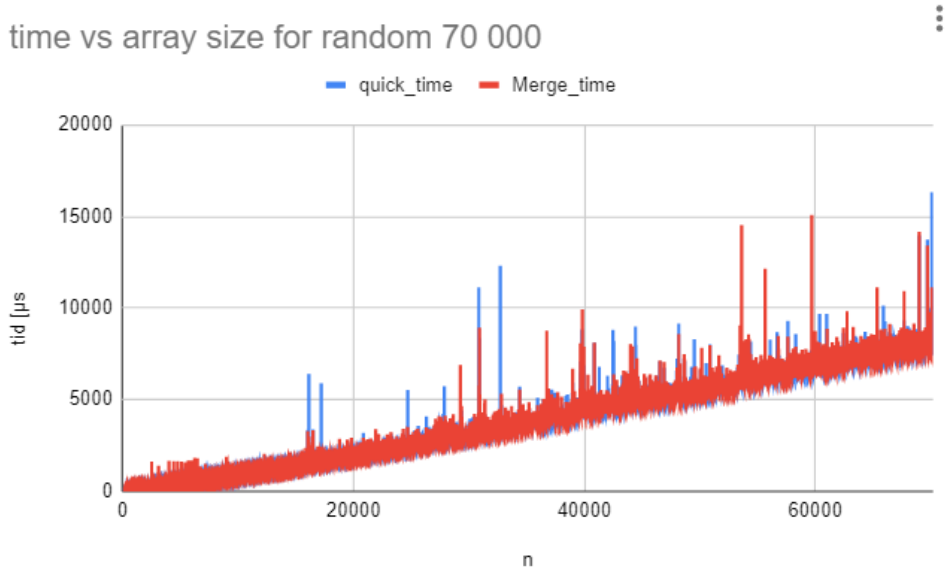
Likevel ser vi et par forskjeller i grafene som blant annet "pigger" i grafen, og et stort dropp rundt n = 250. Piggene kan hende kan forklares med at vi har tilfeldig data og derfor kan få tilfeller der dataen er svært usortert. Derimot ved tilfeldig data ville vi forventet flere tilfeller med mindre "pigger". Vi ser noen tilfeller i insertion sort, men kun et par tilfeller. Hvis dataen vår ikke er svært stor så kan dette likevel være en forklaring. En kunne tenke seg en forklaring der disse piggene ville bli forklart av at pcen har bakgrunnsprosesser, eller at det er tilfeldig opphop i prosessoren. Dette ville derimot ikke forklare hvorfor vi får pigger i de samme stedene hver gang. Det er derfor mer sannsynlig at det kommer fra noe relatert til datasettet eller algoritmen.

Noe annet vi ser er at mellom n = 0 og n = 250 så har vi stigning før et skarpt dropp. Det er usannsynlig at dette kun er tilfeldige data siden vi ser et klart mønster. Akkurat hva som skjer rundt n = 250 er vanskelig å si, men vår gjetning er at dette har noe med hvordan datamaskinen utfører operasjonene. Det kan hende at etter et visst antall kjøringer at det tar kortere tid basert på minneaksesering.

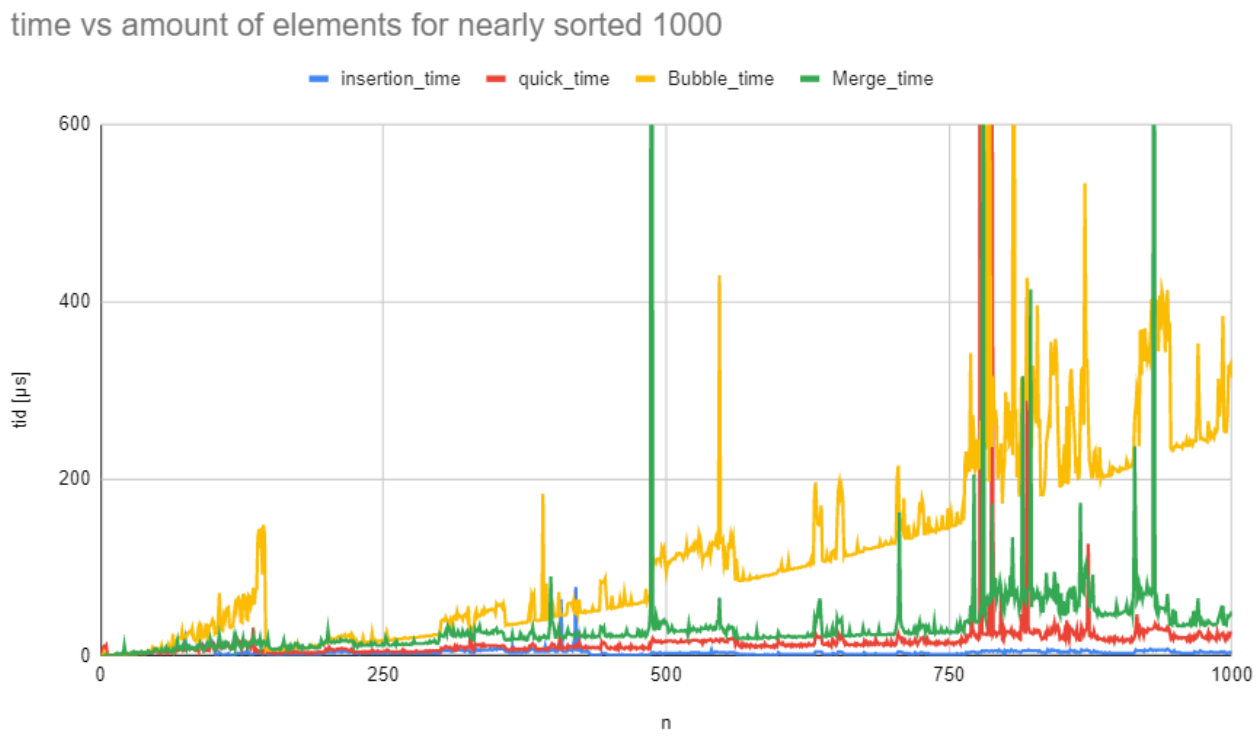
En ting som er vanskelig å se fra den første grafen er om vi har lineær vekst for quick og merge time, eller om vi har n log n. Vi ser derfor på et større datasett med n = 10 000.



Til og med for n = 10 000 så kunne en godt ha konkludert at dette var lineært. Som vi vet er n log n ikke så mye verre enn n. Bruker 10 000 så får vi at n = 10 000,
og at n log n = 10 000 * log(10 000) = 50 000.
Prøver vi også med et enda større array så får vi noe som ligner lineært:



Ser vi også på nearly sorted 1000 ser vi noe interessant.



Ser vi at insertion sort egner seg svært godt på nesten sortert datasett. En forklaring på dette kan være at vi gjør svært få bytter og sammenligninger hvis posisjonen på tallet er på nesten riktig sted. For andre sorteringsalgoritmer som for eksempel bubble sort så løper vi nesten gjennom hele arrayet og sammenligner verdier selv om det nesten er sortert.

Videre ser vi på antall bytter og sammenligninger i forhold til kjøretid.

Test som demonstrerer korrelasjon mellom antall sammenligninger og bytter med kjøretid, input-fil som ble brukt er nearly_sorted_1000, nearly_sorted 10 000

Insertion

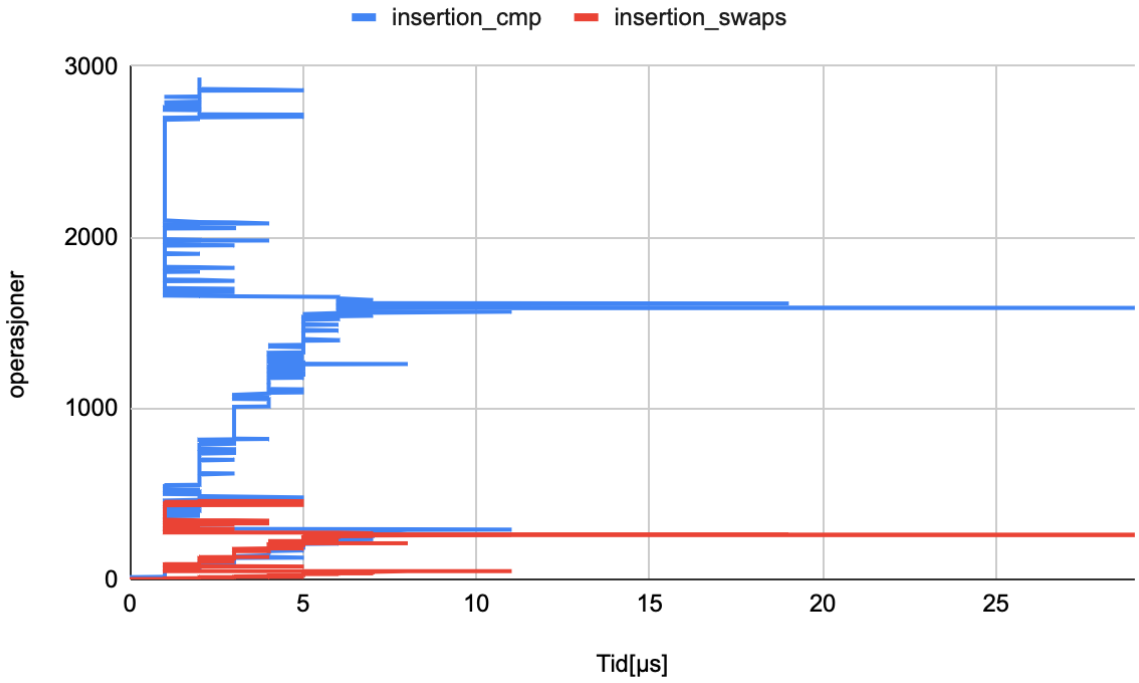


Fig (1) Resultat av nearly_sorted_1000

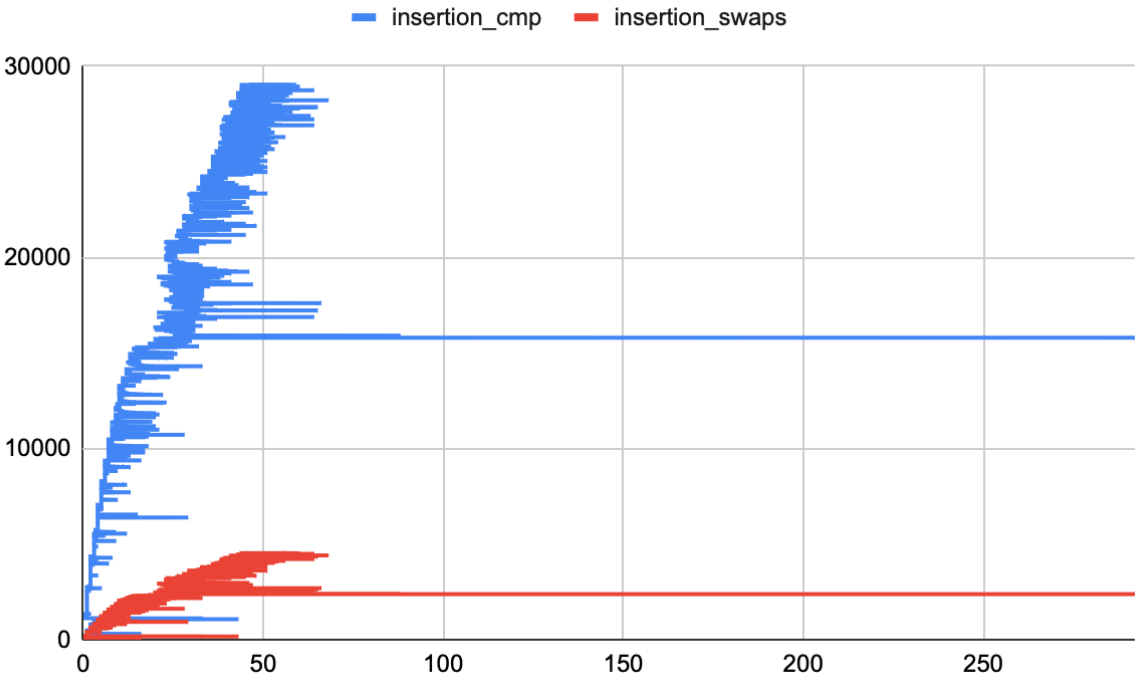


Fig (2) Resultat av nearly_sorted_10 000

-Periodisk svak linear korrelasjon mellom (cmp,swaps) og (tid), sammenhengende linear korrelasjon mellom (cmp) of (swaps), dvs at cmp og swaps er alltid proposjonelt relatert til hverandre, swaps øker med økende sammenligninger, cmp graf ligner på swaps sin, men økningen i antall sammenligninger er mye større en økningen av antall swaps, med det er likevel proposjonelt relatert til hverandre.

Dette mønsteret gjentar seg når vi bruker andre nearly_sorted_ X filer, den langsomme lineare korrelasjonen mellom cmp, swaps med tid er mer tydelig i Fig 2.

Quick

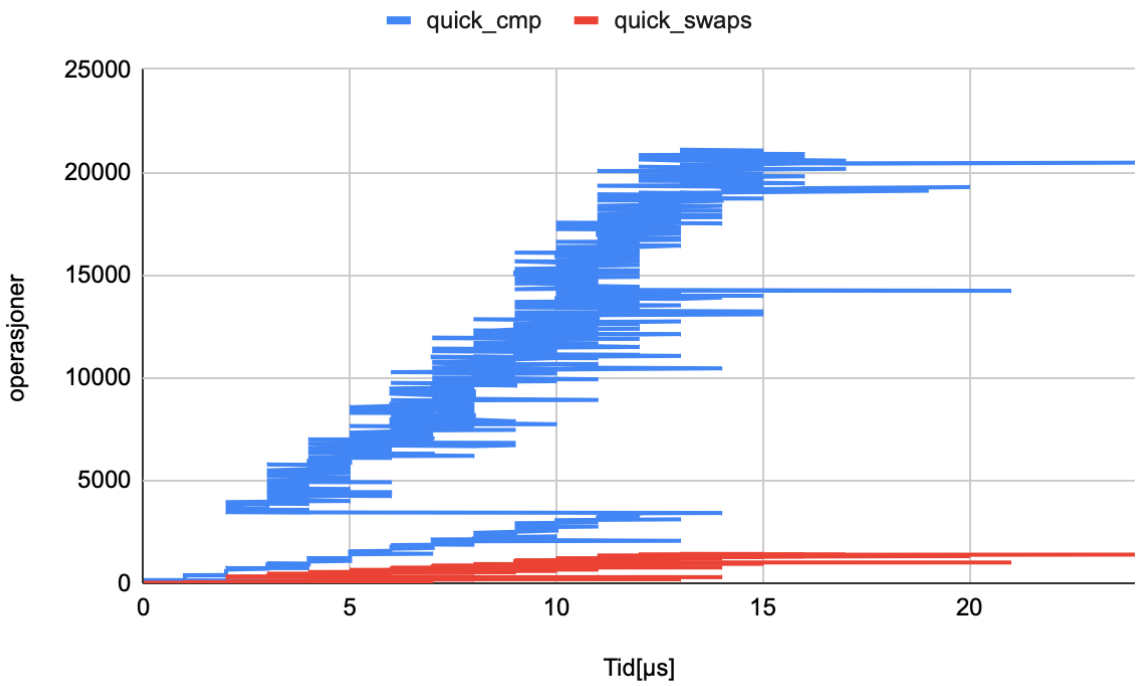


Fig (3) Resiultat av nearly_sorted_1000

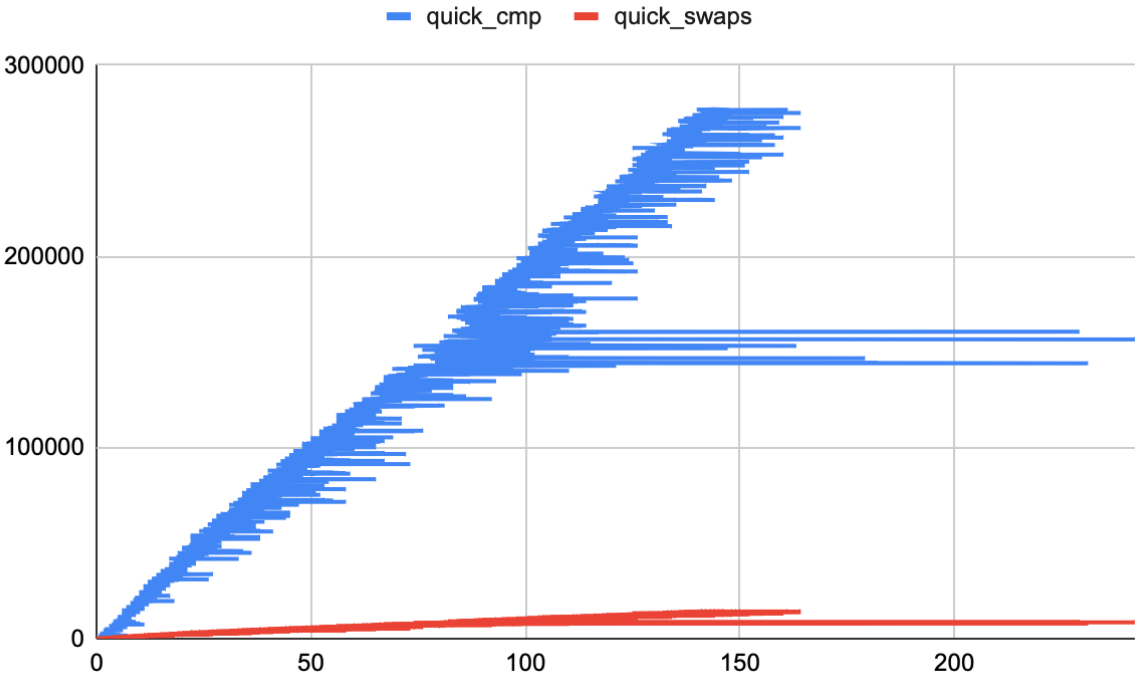


Fig (4) Resultat av nearly_sorted_10000

Begge grafene viser en mer tydelig generel linear korrelasjon mellom cpm,swaps og tid, i tilleg til en sammenhengende linear korrelasjon mellom cmp og swaps gjennom hele utførelsev av algoritmen.
Mønsteret i Insertion (støy i grafen) gjentar seg når det gjelder periodisk uavhengighet mellom (antall swaps, antall cmp) og tid, dvs at mer swaps bruker ikke nødvendigvis mer tid, og mer cmp bruker ikke mer tid, PERIODISKMESSIG, men generelt, og over hele kjøretidsperiode så merker man en tydeligigere linear korrelasjon mellom cmp, swaps og tid (også den er enda mer tydeligere i både Fig 4 enn i Fig 3 hvor begge tilhører Quick algoritme).

Bubble

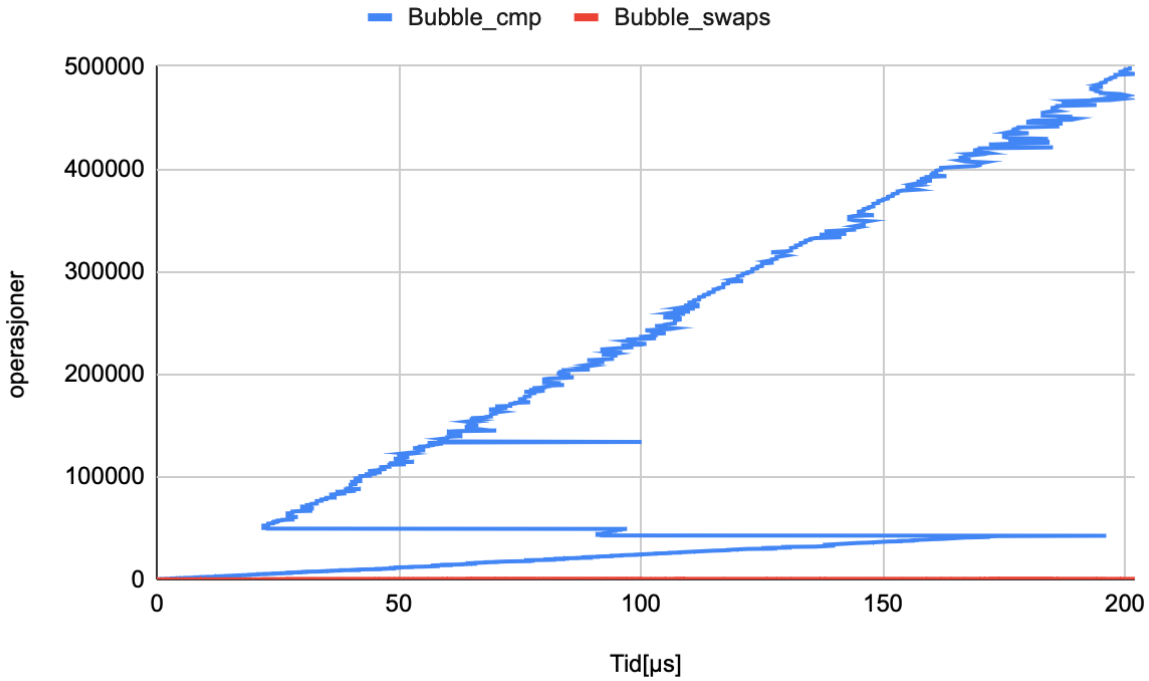


Fig (5) Resultat av nearly_sorted_1000

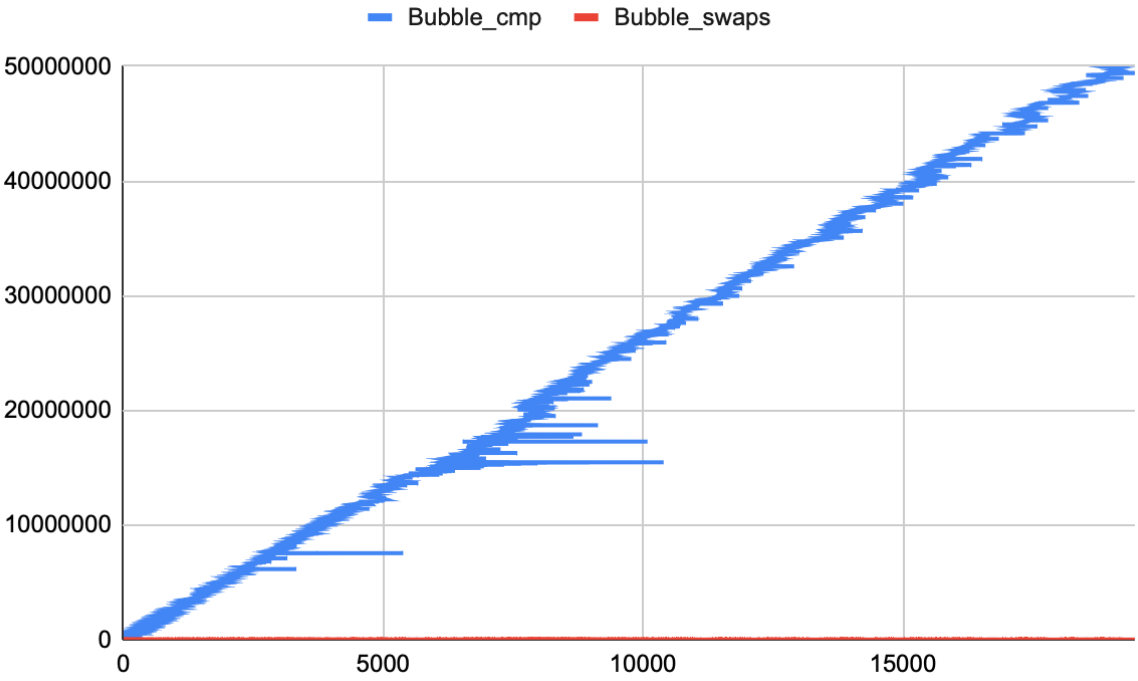


Fig (6) Resultat av nearly_sorted_ 10000

Begge grafene viser en linear korrelasjon mellom antall cmp, antall swaps og en god men ikke-fast /støyete, dvs ikke alltid korrelasjon mellom antall cmp og tid, antall cmp og tid, støyet her er mindre enn det vi så med QUICK og INSERTION grafene.

Merge

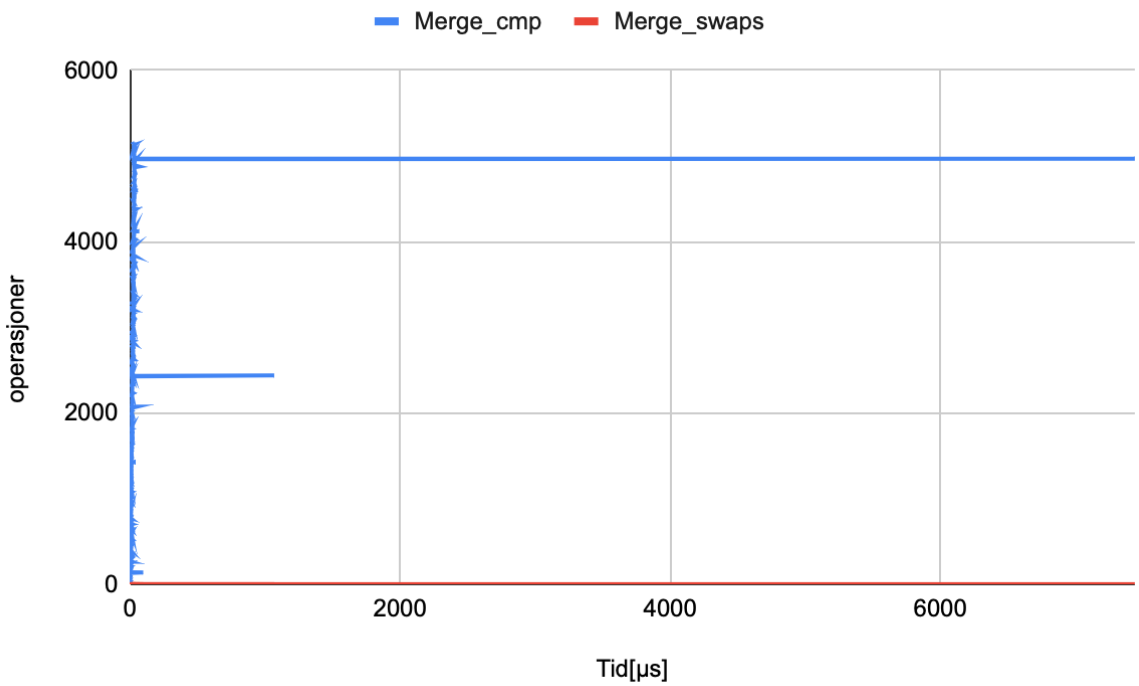


Fig (7) Resultat av nearly_sorted_1000

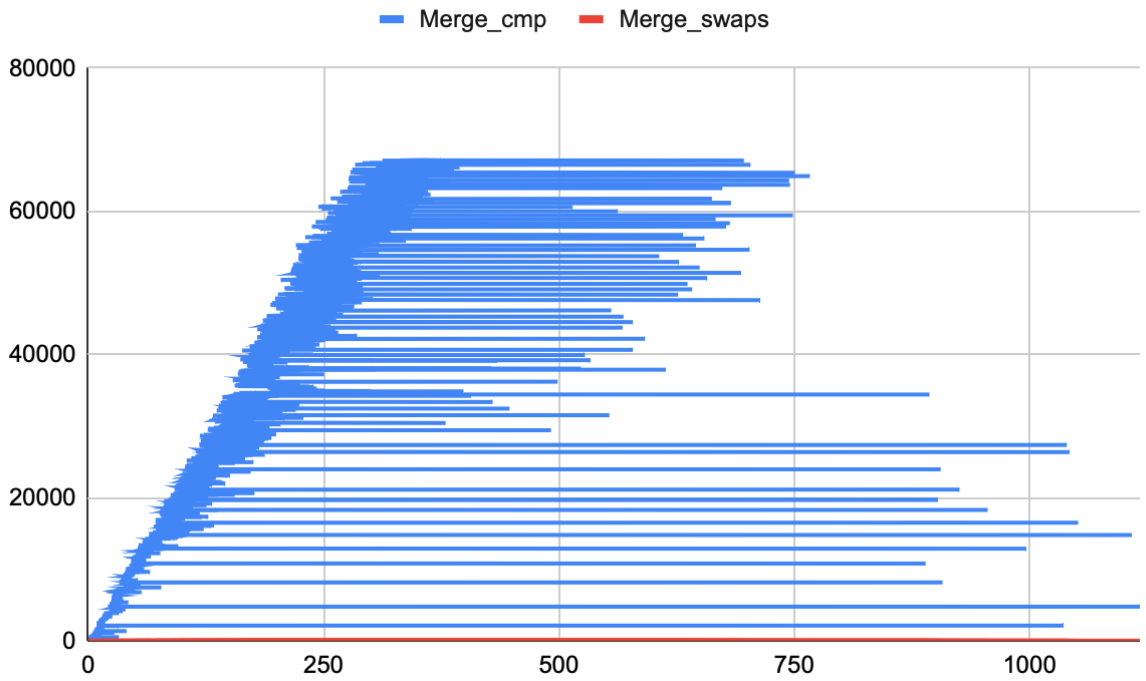


Fig (8) Resultat av nearly_sorted_ 10 000

I Merge er det ingen swaps, linear generell korrelasjon mellom antall cmp og tid som er mer tydelig i Fig (8) hvor man ser stigende linear generell korrelasjon gjennom hele kjøretiden, men støyet er mer frekvent her, dvs alternasjon av periodisk uavhengighet mellom antall cmp og kjøretid er mer hyppig enn i de andre algoritmene, men opprettholder grafen egenskapen av linear korrelasjon mellom antall cmp og tid når man vurderer hele kjøreprosessen av algoritmen.

Test som demonstrerer korrelasjon mellom antall sammenligninger og bytter med kjøretid, input fil som ble brukt er Random_1000

Insertion

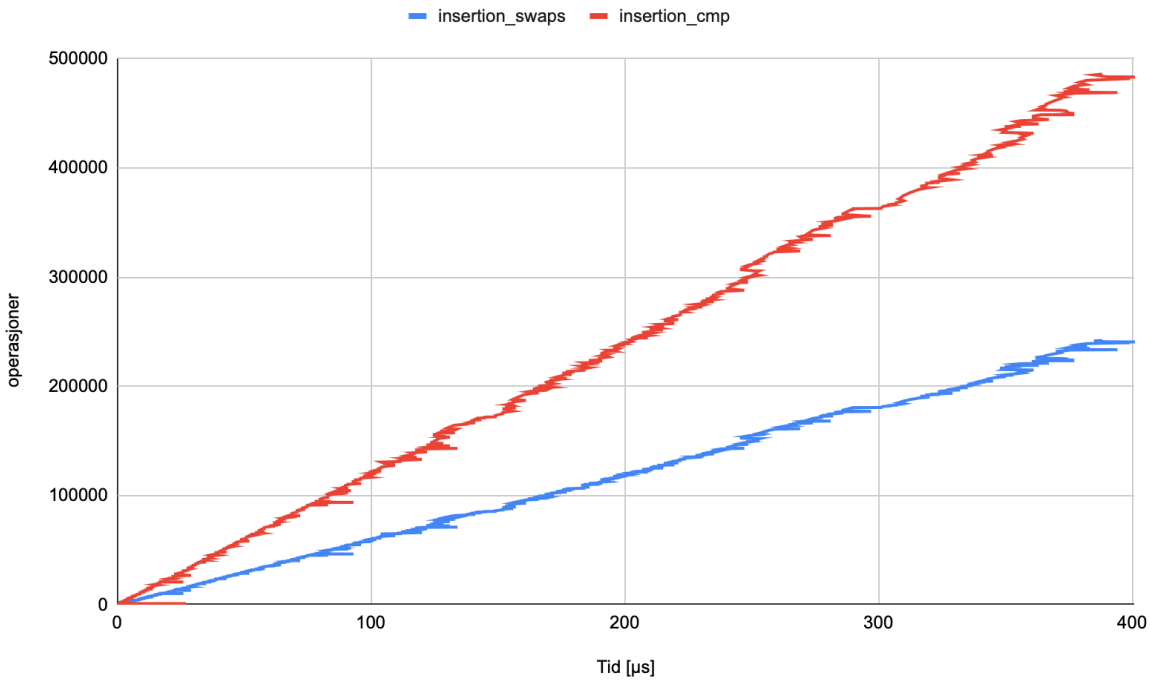


Fig 9 Resulatat av Random_sorted_1000

Grafen viser god linear korrelasjon mellom både antall sammenligninger og swaps med kjøretid, og en sammenhengende linear korrelasjon mellom antall cmp og antall swaps, til men ikke alltid i en stigende stil, dvs at tett sammenheng mellom antall cmp og antall swaps kan skje slik at begge variablene synker samtidig.

Støyet her (periodisk uavhengighet av antall swaps og antall cmp med kjøretid) er til stedet men er mindre synlig, dette kan begrunnes av homogen distrubusjon av elementære verdier som gir en mer ryddig sammenheng i grafen, dvs at distrubusjon følger ikke noe orden som i nearly_sorted filer, dette fører til at lineare korrelasjon mellom de tre variablene som maåles ser mer forutsigbar her.

Quick

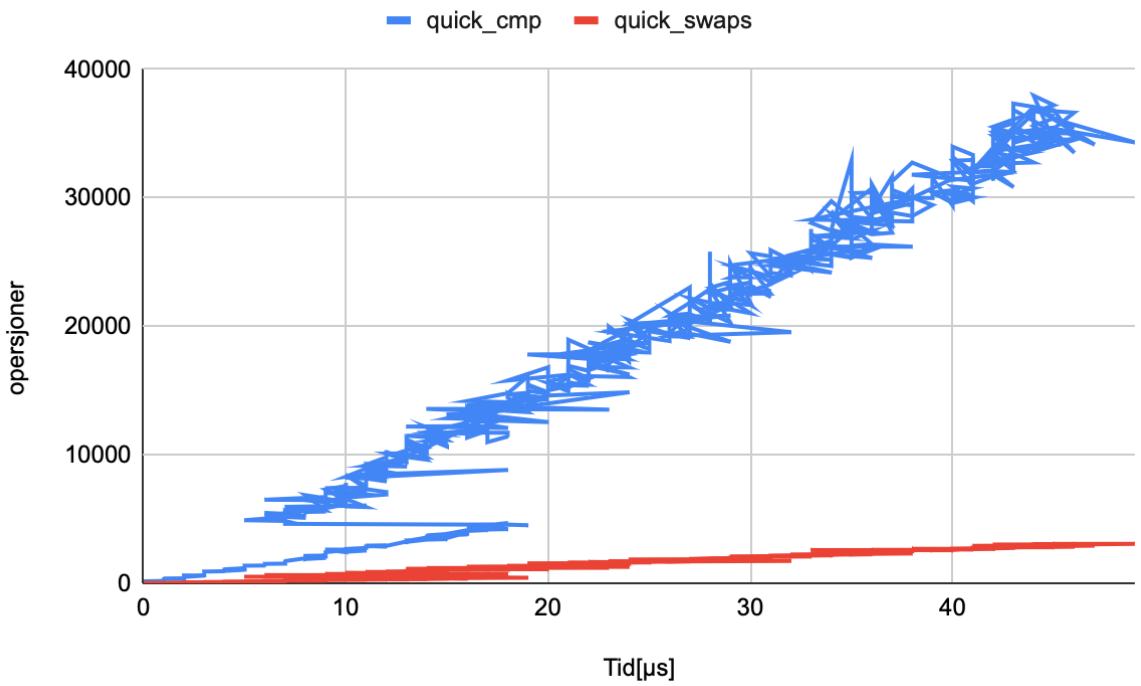


Fig (10) Resultat av random_sorted_1000

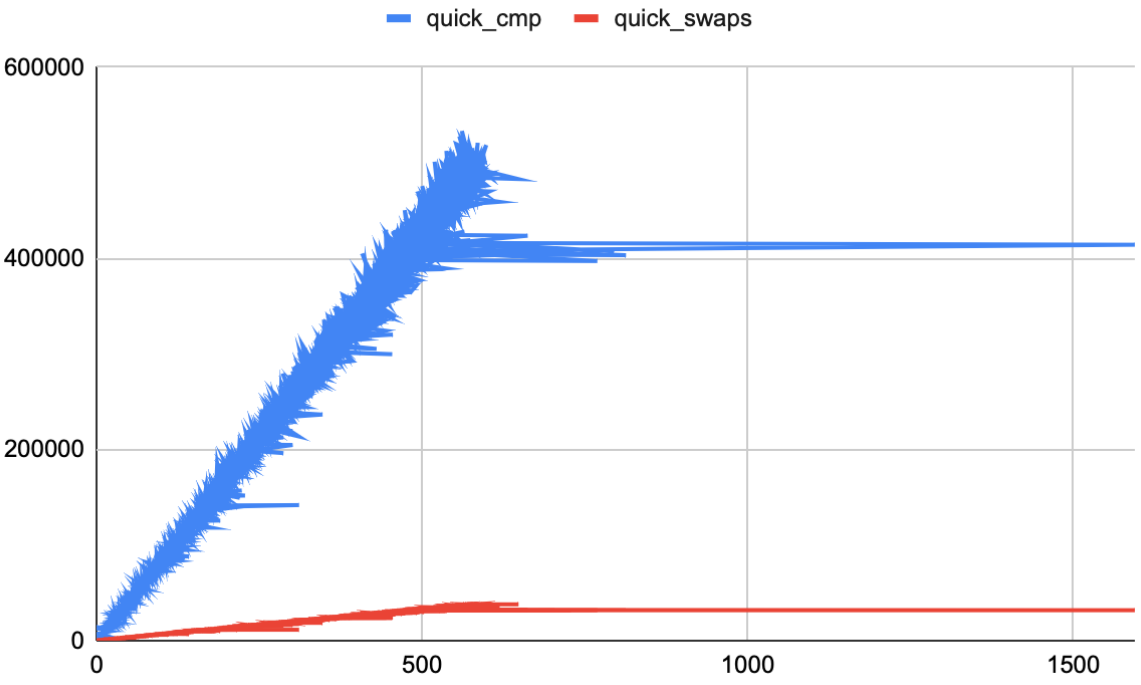


Fig (11) Resultat av random_sorted_10 000

Grafenene viser generel linear korrelasjon mellom antall sammenligninger, swaps med kjøretid, og sammenhengende linear korrelasjon mellom antall cmp og swaps, støyet her omhandler to fenomener, i tilleg til at mer cmp betyr ikke lengre kjøretid, og mer swaps betyr ikke lengre kjøretid, marker man at korrelasjonen mellom antall cmp og antall swaps skjer ikke alltid i en stigende modell, men i en synkende modell også, alternerende men stigende liear korrelasjon, dette gir en annen dimensjon av støyet som er mer synlig i Fig (11).

Bubble

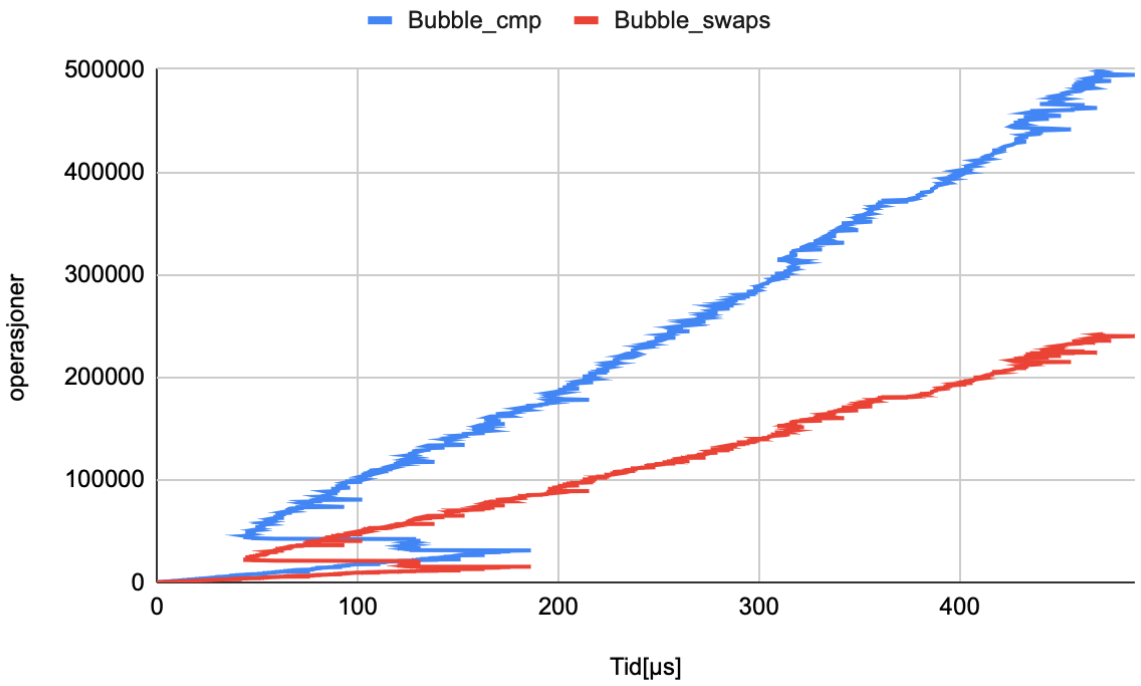


Fig (12) resutat av random_sorted_1000

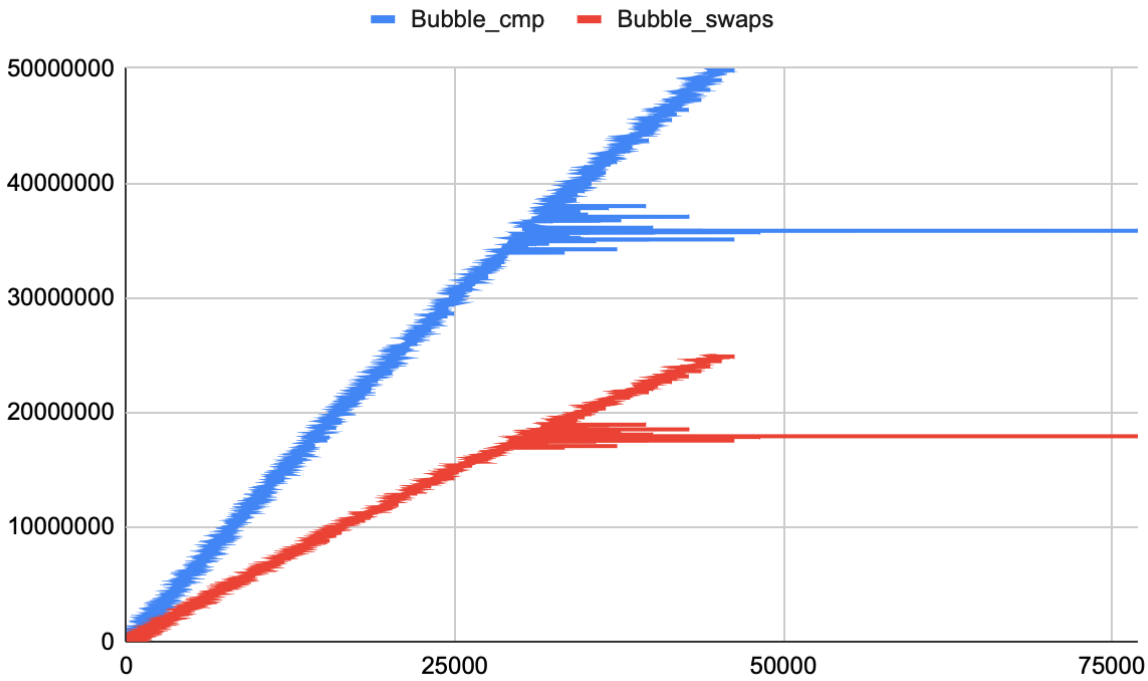


Fig (13) resutat av random_sorted_10 000

En generell linear og stigende linear korrelasjon mellom antall cmp, antall swaps med kjøretiden, til tross for en betydelige område i grafen hvor kjøretiden øker mye med økende antall swaps og cmp. Den kjøretids-økningen laster i en periode før kjøretiden blir justert igjen for resten av kjøreprossessen og følger den samme lineare relasjonen mellom swaps, cmp og tid, sammenheng mellom antall cmp og antall swaps er lineart men ikke optimalt stigende, faste samtidige nedganger som gjennom hele prossessen, men disse er ikke så synlige som i QUICK.

Merge

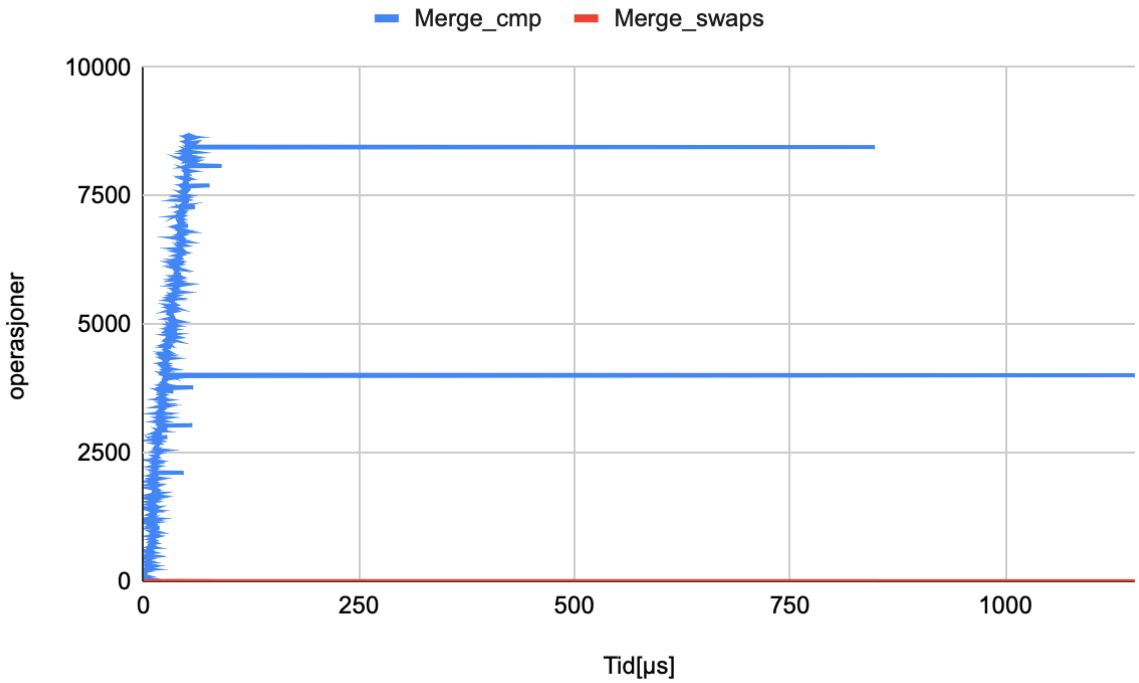


Fig (14) resultat av random_sorted_1000

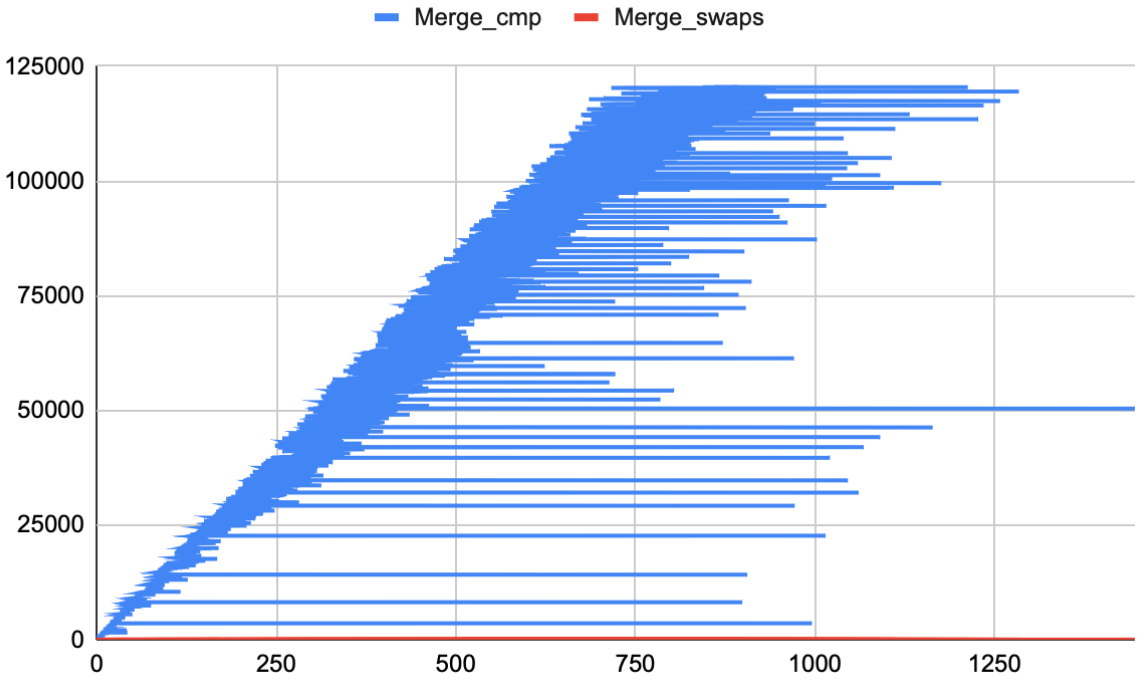


Fig (15) resultat av random_sorted_10 000
Ingen swaps
Hyppige spikes i kjøretiden

Antall cmp går opp og ned men ikke i en betydelig intervall, de fortsetter å gå opp generelt gjennom hele prosessen som gir en generel linear korrelasjon med kjøretiden til tross for hyppige spikes i kjøretide som tyder på stor grad av uavhengighet mellom cmp og tid, men likevel opprettholder grafen den generelle stigende oppgang av antall cmp hvor større og større lister introduserers til algoritmen.

Hvilke algoritmer utmerker seg når n er veldig stor, og når n er veldig liten:
Grafene nedest viser kjøretids-utviklingen til de 4 algoritmene, filen random_1000 er brukt for testing

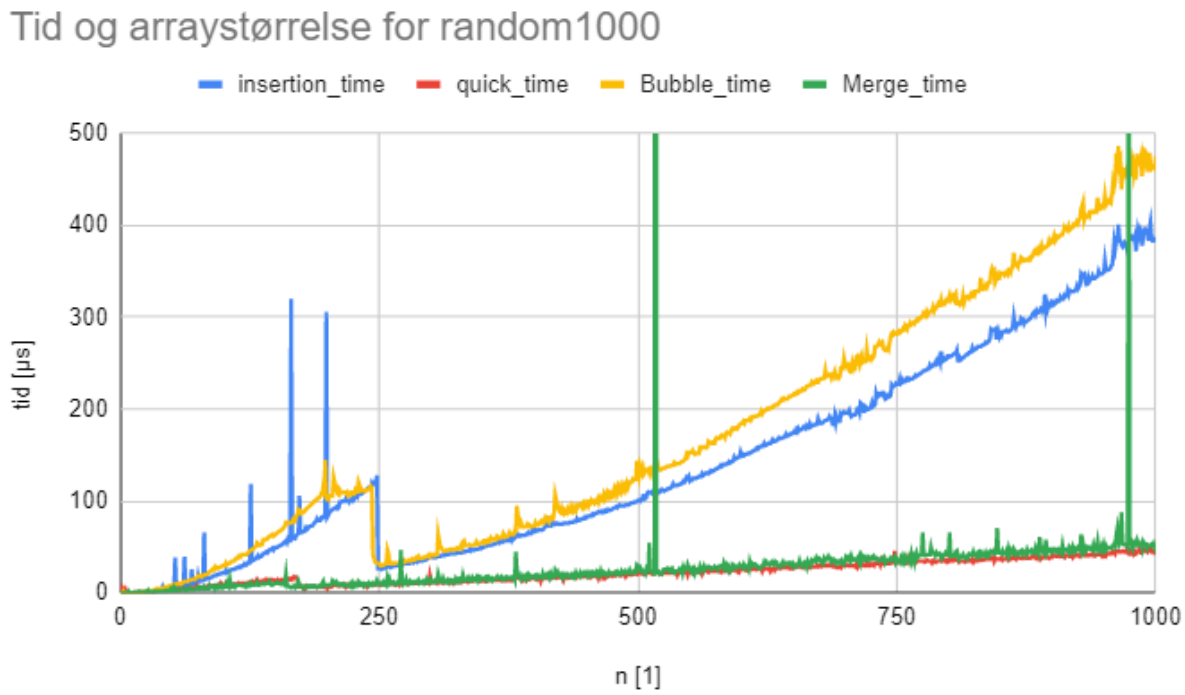


Fig (16) sammenligning av kjøretid i de 4 algoritmene.

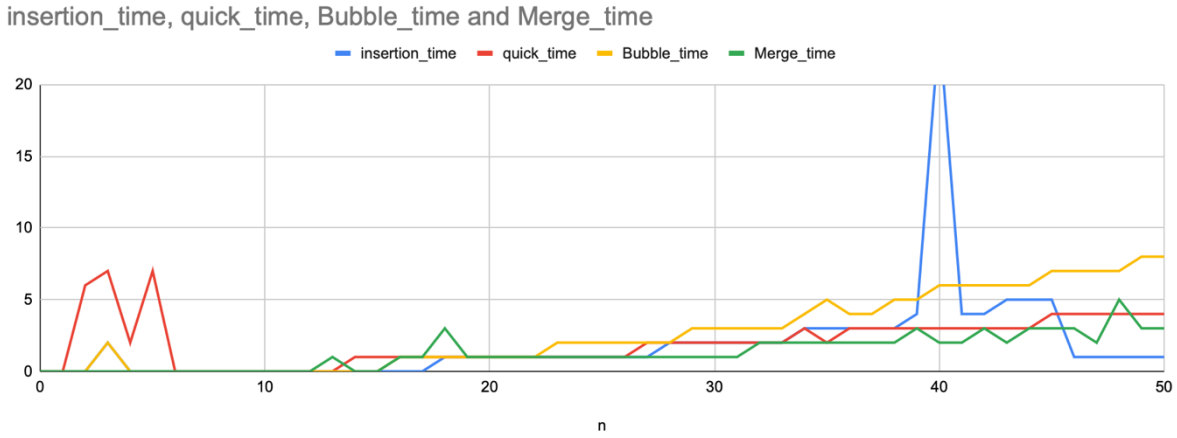


Fig 17 (antall elementer n ble redusert til intervall 0-50, og kjøretidsintervall ble redusert til 0-20 mikrosekunder)
Grafen viser at insertion og merge algoritmene er raskere når n er veldig liten, Quick og Merge er treigest når n er liten.

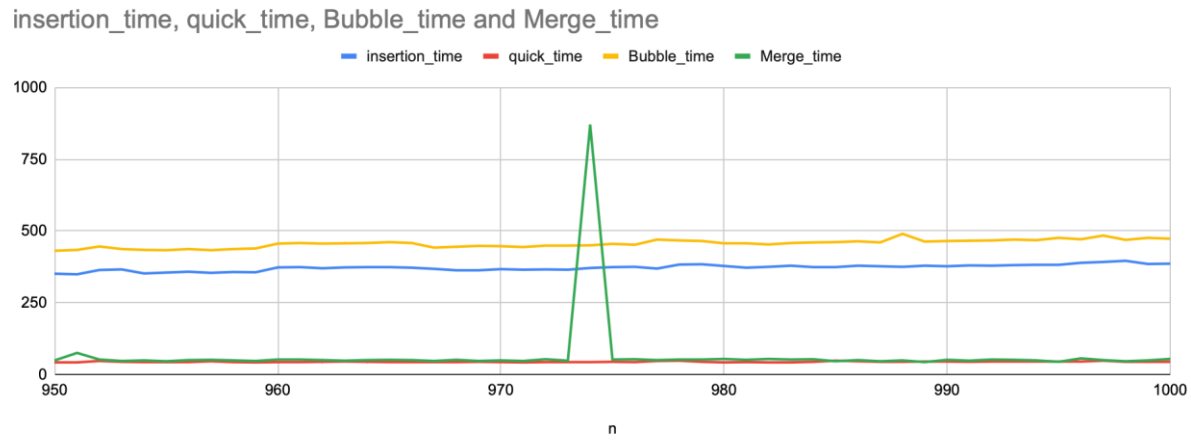


Fig 18 (n intervall er satt til intervall 950-1000)

Grafen viser at Quick og merge er raskest til tross for tilfeldige spikes i merge kjøretid, insertion og bubble er treigest når n er veldig stor.

For overraskende funn så har vi diskutert “pigger”, at insertion sort egner seg godt på nesten sorterte datasett, og at en god del av datasett har såkalte “slopes” i starten. Siden dette allerede er diskutert så tar vi ikke med det i et eget punkt.