Øving 3 Algoritme og datastrukturer IDATT2101

# ***Sorteringsoppgave 1, quicksort med hjelpealgoritme***

# Hjelpealgoritmen

I denne oppgaven har jeg brukt innsettingsalgoritmen som hjelpealgoritme til quicksort-algoritmen. Innsettingsalgoritmen er enkel og effektiv på små datasett, og spesielt god hvis dataen er allerede sortert fra før av. Av de sorteringsalgoritmene som vi ble introdusert for i forelesningen 05.09.2023, syntes jeg at innsettingsalgoritmen var en av de mest utfordrede å ta fatt på. Derfor endte jeg opp med denne hjelpealgoritmen.

# Implementasjon

Figurene under viser hvordan implementasjonen av algoritmene ble utført.

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelseEt bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelseEt bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelseEt bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

Figur 1: median3sort algoritmen for å finne et pivotelement.

Figur 2: innsettingssorteringsalgoritmen som hjelpealgoritme.

Figur 3: Hovedmetoden for quicksort algoritmen med innsettingsmetoden som hjelpe algoritmen.

Figur 4: splitt-metode for å dele delområdet og returnere en ny indeks for pivotelementet.

I quicksort-metoden er den største endringen implementeringen av innsettingssort-metoden som hjelpemetode. Her sjekker quicksort først om størrelsen mellom og indeksen er større enn to, og deretter sjekker den om størrelsen mellom og er mindre enn en «passende» verdi. Hvis en deltabell er mindre enn denne verdien, kommer quicksort-algoritmen til å bryte rekursjonen, og hoppe på innsettingsalgoritmen i stedet. Ellers utføres quicksort med vanlig rekursjon. Det som er interessant er å finne ut av hvilke verdier for denne størrelsen som optimaliserer tidsforbruket.

# **Tidsmålinger**

Figur 5: Metode for tidsmålinger av for å finne den beste passende størrelsen. Merk at metoden clone() blir brukt for at tidtakingsalgoritmen ikke skal la quicksort sortere sorterte tabeller.

# Et bilde som inneholder tekst, skjermbilde, Font, nummer Automatisk generert beskrivelse

Ved å bruke metoden illustrert over kan det lett foretas tidsmålinger over quicksort-algoritmen med ulike verdier for den passende størrelsen når rekursjonen skal brytes. I målingene kaller jeg denne verdien for «terskel». Tabellen som brukes i parameteren for alle målingene har en størrelse på 1000000. I tillegg finnes det en metode som genererer annethvert tall i tabellen tilfeldig i et område fra 0 til 100000. I likhet med de tidligere øvingene har jeg både gjort tidsmålinger på min bærbare og stasjonære datamaskin.

**Ut ifra målingene ser vi at det er en stor forskjell på tidsforbruket til den stasjonære og bærbare datamaskinen. Det som i midlertidig er interessant er verdiene til terskelen hvor tidsbruket er lavest. Figur 8 illustrerer dette.**

Figur 6: Tabell over tidsmålingene av quicksort med hjelpealgoritmen på den bærbare pc-en.

Figur 7: Tabell over tidsmålingene av quicksort med hjelpealgoritmen på den stasjonære pc-en.

Et bilde som inneholder line, Plottdiagram, diagram, Parallell

Automatisk generert beskrivelse

Figur 8: Måleverdier fra tidsmålingen på den stasjonære PC-en. Geogebra sitt verktøy for regresjonsanalyser fant ingen passende funksjon for måledataene. Linjestykker er derfor plassert mellom punktene.

**Slik som figuren over illustrerer for tidsmålingen på den stasjonære PC-en ser vi at det er i området mellom 10 og 80 hvor terskelen gir optimalisert tidsforbruk. Vi kan deretter innsnevre tidsmålingene i dette området for å finne en mer nøyaktig verdi for terskelen.**



Figur 9: Tabell over tidsmålingene av quicksort med hjelpealgoritmen på den bærbare pc-en for gunstige verdier.

Figur 10: Tabell over tidsmålingene av quicksort med hjelpealgoritmen på den stasjonære pc-en for gunstige verdier.

Av figur 10 og 9 som viser resultatene fra tidsmålingene, ser vi at den korteste tiden som quicksort algoritmen bruker ligger rundt 40 og 45. Dette varierer fra gang til gang man utfører tidsmålingene. Tidsmålingene kan også vise at korteste tid er når terskelen er 50, 30, 35 i noen tilfeller. Den mest hyppigste verdien jeg har fått for best tid er 40.

Et bilde som inneholder line, Plottdiagram, diagram, Parallell

Automatisk generert beskrivelse

Figur 11: Måleverdier fra tidsmålingen til den stasjonære pc-en. Her ser vi at de beste verdiene for terskelen 45, noe som forekommer ofte under flere tidsmålinger. Den nest beste verdien er 40.

# Oppsummering

Til slutt kom jeg fram til at den passende størrelsen for når quicksort-algoritmen skal bryte rekursjonen, er 40. Dette var resultatet jeg fikk flest ganger som ga best tid for den stasjonære pc-en. For den bærbare fikk jeg ofte det samme resultatet, men det er mye mer tilfeldig hva som faktisk ender opp med å bli best. F.eks. i enkelte tidsmålinger opplevde jeg at den bærbare fant at verdiene 60, 70, og av og til 20 ga best tid. Derfor lente jeg meg mest på tidsmålingene til den stasjonære pc-en. Ut ifra at innsettingsmetoden er kjent for å være enkel og effektiv på små datasett, gir det mening at den passende størrelsen burde ligge lavt på rundt 50.

# Felles krav

For at felleskravende skal godkjennes har jeg laget en egen testkode for at programmet sorterer korrekt, og at summen før og etter er like.

I main-metoden i kildekoden har jeg implementert tidsmålingene slik at quicksort først kjøres uten hjelpealgoritme, og deretter med hjelpealgoritme for flere ulike verdier av terskelen. Deretter kjøres tidsmålingene på nytt, denne gangen med allerede sorterte tabeller for å vise at det ikke oppstår noen -problemer osv. Til slutt kommer det en siste tidsmåling for gunstige verdier av terskelen. (Alt dette kan ta noen sekunder å gjennomføre, og jeg er klar over at det er litt uoversiktlig.)