Rapport til arbeidskrav 2 IDATT2101

I denne oppgaven har jeg sammenlignet tidsbruk av sortering av arrays ved bruk av «quicksort» og «dual pivot quicksort». Kodene finner du i filene Sortering.java og DobbelPivotSortering.java.

**Quicksort:**

I oppgaveteksten stod det at man burde bruke quicksort metoden som man finner i læreboka. Grunnet at akademika har vært utsolgt for læreboka i faget har jeg den ikke og måtte derfor bruke en metode jeg fant på geeksforgeeks (<https://www.geeksforgeeks.org/quick-sort/>). I oppgaveteksten stod det at vi skulle teste med arrayer med 10-100 millioner tall, men når jeg forsøkte dette med quicksort metoden fikk jeg java.lang.StackOverflowError exceptions. For å få gjennomført sorteringen måtte jeg derfor ned i 10 000 tall (den sorterte arryen kunne ikke sorteres med større tall, tilfeldig array klarte 1 million). Ar jeg ikke fikk brukt ønsket størrelse var litt beklagelig men jeg antar at det viktigste med øvingen er å forstå sorteringsalgoritmer samt bruke dem slik at de funker i praktis.

Før jeg testet hastigheten gjorde jeg sumsjekktest og rekkefølgetest på alle 3 arrayene for å forsikre meg om at de stemte. Etter alle testene var vellykket startet jeg å måle tidsbruk ved å sortere listene fem ganger hver og ta gjennomsnittet av tidsbruket til å sette en «fasit».

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tilfeldig | Annenhver lik | Sortert |
| 1000 tall | 1ms | 3,4ms | 7ms |
| 10 000 tall | 7,6ms | 24,8ms | 36,8ms |

Disse testene er kanskje ikke de beste (grunnet små arrayer), men de gir et godt innblikk i hvor lang tid de ulike arrayene bruker. Her ser man kanskje overraskende nok at en helt tilfeldig array bruker kortest tid, og en ferdig sortert en bruker lengst tid med en forskjell på 480% mer tid for enn sortert enn en tilfeldig med 10 000 tall.

**Dual pivot quicksort:**

Når jeg skulle bruke dual pivot quicksort brukte jeg den som var linket til i oppgaveteksten med den endringen som var beskrevet. På samme måte som istad sjekket jeg alle arrayene gjennom sjekksumtesten og rekkefølgetesten for å forsikre meg om at metoden fungerte. Når alle testene var vellykket tok jeg fem målinger av hver array og brukte gjennomsnittet som «fasit». Dobbel pivot quicksorten kunne håndtere mye større arrayer, men ikke 100 millioner. Da fikk jeg en Java heap space exception. Derfor testet jeg på 1 million og 10 millioner.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tilfeldig | Annenhver lik | Sortert |
| 1 million tall | 137,5ms | 116,8ms | 87,4ms |
| 10 millioner tall | 941ms | 732,6ms | 603,4ms |

Her ser vi en tydelig endring fra single pivot. Både med 1 million tall og 10 millioner tall så går det raskest å sortere den sorterte listen, så den med annenhver lik og til slutt den tilfeldige listen. Dette stemmer mer overens med det jeg forventet fra før av. Det er litt vanskelig å sammenligne dobbel pivot quicksort med vanlig quicksort siden størrelsene på arrayene er så forskjellige, men spesielt hvis vi ser på den sorterte listen så tar det bare litt mer enn dobbelt så lang tid å sortere 1 millioner tall som 10 000 tall selv om det er 100x større array. Dette tyder på at dobbel pivot quicksort er raskere enn vanlig quicksort (spesielt med større array). Samt at vanlig quicksort ikke engang kunne håndtere arrayer større enn 10 000 tall.

Sortering.java inneholder quicksort metoden, sumsjekktest for quicksort og rekkefølgetest. Main metoden som kjører programmet er også der. Testene er kommentert ut men de er testet og de fungerer. Ellers er det bare å erstatte tall og variabler/metode kall for å teste at alt fungerer.