Tobias Skipevåg Oftedal 05.09.2023

Øving 3, avansert sortering

Oppgave 2, Quicksort med tellesortering

Implementasjon:

- Single pivot, baseres på det midterste tallet i arrayen
- Implementerte tellesortering fra boken, men endret denne slik at den bare sorterer en array fra og mellom 2 gitte indekser. Metoden endrer også selve input arrayen i stedet for å ha en egen array som er output
- Tellesorteringen er ikke endret for å fungere for negative tall, da en slik algoritme medfører flere ulemper

Resultater

Kjøretid ved ved lite verdi-område for mulige tall:

```
Test where the possible integers are small (0-100)
| Quicksort  | Quicksort improved   | n                  |
| 30         | 40                   | 100000             |
| 176        | 19                   | 1000000            |
| 1821       | 166                  | 10000000           |
| 18264      | 1834                 | 100000000          |
```

De første tallene har litt unøyaktighet fordi n er for liten til å måle nøyaktig. Vi ser tendenser til at quicksort følger den hypotetiske kompleksisteten på O(n log(n)) som gjennomsnittlig kjøretid. Vi ser her at den nye algoritmen med tellesortering går mye kjappere i disse tilfellene, og legger seg en plass mellom o(n) og o(n*logn). Den nye implementasjonen øker ikke gjennomsnittlig tidskompleksitet, da den utfører et lineært arbeid, som ikke er nøstet. Worst case scenario for tidskompleksitet vil fremdeles være det samme, selv om den nye algoritmen i mange tilfeller vil bruke mer tid, pga flere operasjoner.

Kjøretid ved ved større verdi-område for mulige tall:

```
Test where the possible integers are larger (0-Integer.Max)
| Quicksort   | Quicksort improved   | n                    |
| 26          | 18                   | 100000               |
| 193         | 207                  | 1000000              |
| 2152        | 2175                 | 10000000             |
```

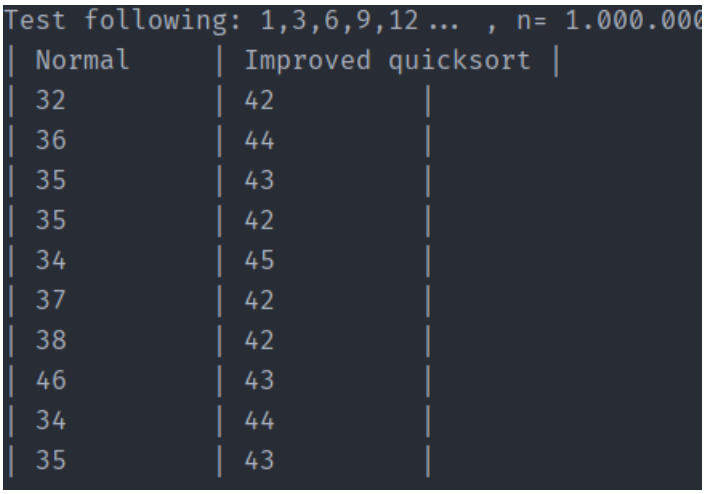
Når vi har tall som er innenfor et større område, ser vi at den nye algoritmen har færre muligheter for å kjøre tellesortering. Dette resulterer i at den nye algoritmen gjennomfører en god del unødvendige operasjoner, sammenlignet med vanlig quicksort, noe som gjenspeiles i testkjøringen.
På denne testen ser man og ulempen ved den nye metoden, ettersom den ikke klarer å sortere en array som har en lengde på mye mer enn 10 millioner før man får en exception på grunn av mangel av plass i heap.

Kjøretid på tabell som allerede er sortert:

```
Test to check for no n^2 problems for improved quicksort
| Improved quicksort | relative difference|
| 1084               | 1                  |
| 428                | 0.3948339483394834 |
| 335                | 0.30904059040590404 |
| 323                | 0.29797047970479706 |
| 325                | 0.29981549815498154 |
| 323                | 0.29797047970479706 |
| 319                | 0.29428044280442806 |
| 327                | 0.3016605166051661 |
| 320                | 0.2952029520295203 |
| 323                | 0.29797047970479706 |
| 321                | 0.29612546125461253 |
```

Vi ser at algoritmen ikke har problemer med å sortere en tabell som allerede er sortert. Tabellen i eksempelet er 1 million lang med siffer fra 0 til 1 million. Vi ser også at sorteringstiden reduseres betraktelig til ca 30% av den originale sorteringstiden.

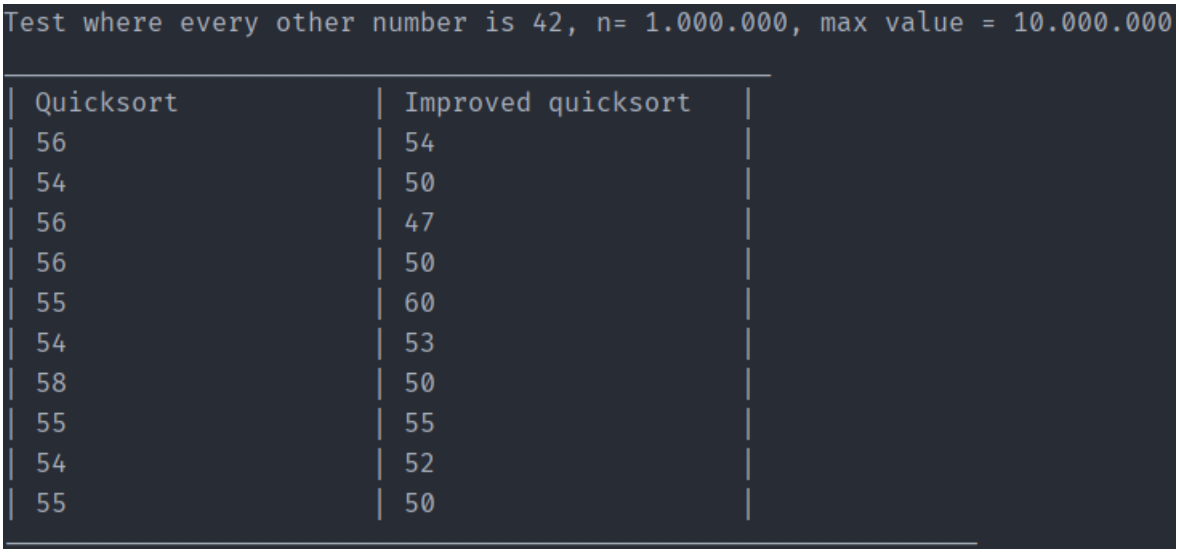
Kjøring av rekken 1,3,6,9 for n = 1 million:



```
Test following: 1,3,6,9,12 ...  , n= 1.000.000
| Normal     | Improved quicksort |
| 32         | 42                 |
| 36         | 44                 |
| 35         | 43                 |
| 35         | 42                 |
| 34         | 45                 |
| 37         | 42                 |
| 38         | 42                 |
| 46         | 43                 |
| 34         | 44                 |
| 35         | 43                 |
```

Vi ser at den normale quicksort varianten er marginalt bedre, fordi den forbedrede algoritmen aldri får utført en tellesortering slik arrayen er satt opp.

Test når annethvert tall er 42



```
Test where every other number is 42, n= 1.000.000, max value = 10.000.000
| Quicksort          | Improved quicksort |
| 56                 | 54                 |
| 54                 | 50                 |
| 56                 | 47                 |
| 56                 | 50                 |
| 55                 | 60                 |
| 54                 | 53                 |
| 58                 | 50                 |
| 55                 | 55                 |
| 54                 | 52                 |
| 55                 | 50                 |
```

Vi ser at når annenhvert tall er 42, bruker begge metodene ca like lang tid, og at hvilken algoritme som «vinner» varierer etter hva arrayen inneholder.