

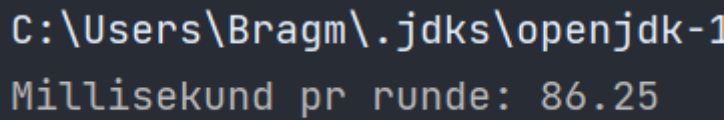
Brage Minge

Algoritmer og datastrukturer

Øving 3

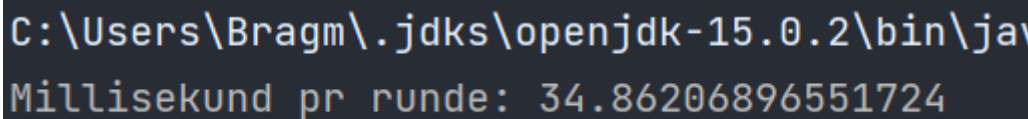
Jeg har valgt å gjøre oppgave 2

Jeg fikk quicksort til å bli enda raskere som vist av tallene nedenfor. Bilde 1 er vanlig quicksort og bilde 2 er quicksort med counting sort. Vanlig quicksort ligger et sted mellom 80 og 90 ms per runde på min maskin, og den forbedrede ligger på mellom 30 og 40 ms per runde.



```
C:\Users\Bragm\.jdk\openjdk-15.0.2\bin\java.exe "-javaagent:C:\Users\Bragm\workspace\src\main\resources\jvarkit.jar"
Millisekund pr runde: 86.25
```

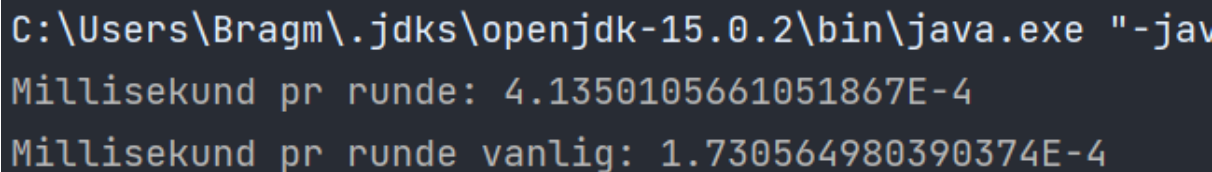
Bilde 1



```
C:\Users\Bragm\.jdk\openjdk-15.0.2\bin\java.exe "-javaagent:C:\Users\Bragm\workspace\src\main\resources\jvarkit.jar"
Millisekund pr runde: 34.86206896551724
```

Bilde 2

Gitt et datasett der ingen av testene vil slå inn, bare kjøres gjennom, som f.eks. en tabell med tregangen, kan man se av bilde 4 at vanlig er mye raskere.



```
C:\Users\Bragm\.jdk\openjdk-15.0.2\bin\java.exe "-javaagent:C:\Users\Bragm\workspace\src\main\resources\jvarkit.jar"
Millisekund pr runde: 4.1350105661051867E-4
Millisekund pr runde vanlig: 1.730564980390374E-4
```

Bilde 4

I klassen main.java, kan man se at den nye quicksort algoritmen klarer alle testene gitt i oppgaven, som å sortere en million tall og å sortere dem igjen på rimelig tid, uten feil.

Av bilde 5 kan man se hvilke tester som gjøres, og på bilde 6 kan man se at de bekrefter at algoritmen virker som den skal.

```

int sumBefore = 0;
for (int i = 0; i < arr.length; i++) {
    sumBefore += arr[i];
}

Algorithm.newQuicksort(arr, v: 0, h: arr.length-1);
int sumAfter = 0;
for (int i = 0; i < arr.length; i++) {
    sumAfter += arr[i];
}
for (int i = 1; i < arr.length; i++) {
    if (arr[i] < arr[i - 1] && arr[i] != arr[i - 1]) {
        System.out.println("Wrong sorting");
        break;
    }
}
System.out.println(sumBefore == sumAfter);

```

Bilde 5

```

C:\Users\Bragm\.jdk\openjdk-15.0.
true

Process finished with exit code 0

```

Bilde 6