Øving 5

Oppgave 1

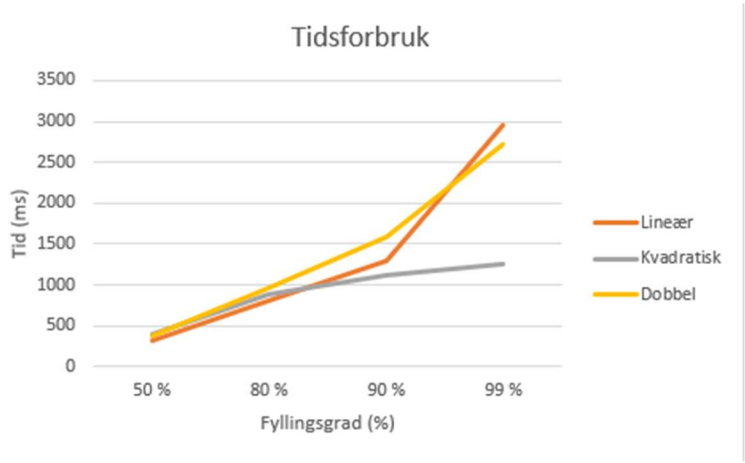
```
Load factor: 0.7814569536423841
Mean collisions per person: 0.24503311258278146
Number of collisions during insertion: 37
```

Over kan man se en utskrift fra programmet. Her er alpha = 0,78, gjennomsnittlig kollisjoner per person er 0,25 og antall kollisjoner totalt er 37.

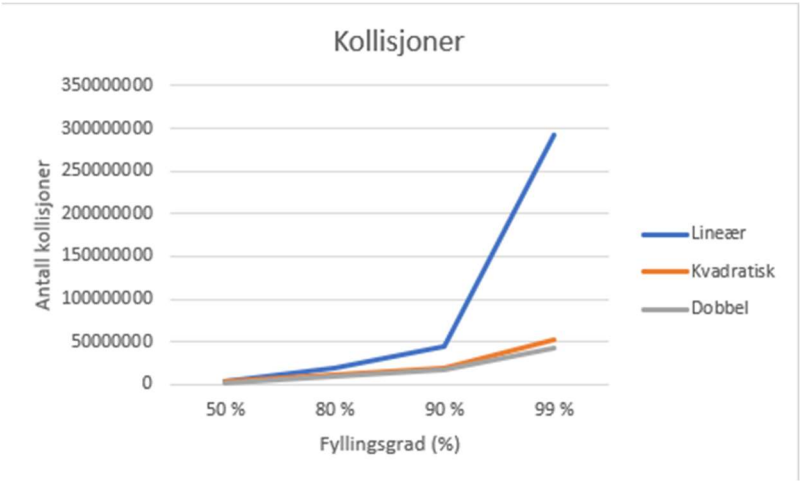
```
Anders Tellefsen
```

Når man søker etter en person i lista, dersom denne personen er i lista blir navnet til hen printet ut, hvis ikke blir det kastet en NullPointerException, fordi dersom ikke personen ikke er i lista er det heller ikke opprettet en Node for personen. Det vil da si at når programmet caller getValue() på objektet, så er objektet Null.

Oppgave 2



I diagrammet over kan man se tiden det tar for de ulike hashfunksjonene for ulik fyllingsgrad. Ut fra diagrammet kan man se at den kvadratiske funksjonen er raskest når fyllingsgraden er over 80%. Når fyllingsgraden er mellom 50% og 80% er lineær den raskeste. Dobbel er den tregeste, med unntak når fyllingsgraden blir 99%. Da er den raskere enn den lineære. Grunnen til at jeg ikke har tatt med for 100% fyllingsgrad er at jeg ikke fikk noe output da jeg kjørte de forskjellige hashfunksjonene.



I diagrammet ovenfor kan man se antall kollisjoner i forhold til fyllingsgrad for hver av de ulike hashene. Her kan man bl.a. se at Dobbel har lavest antall kollisjoner uansett fyllingsgrad, og lineær har betydelig større antall kollisjoner. Når halvparten av tallene settes inn, altså fyllingsgrad lik 50%, så ser man ikke like tydelig forskjellen på antall kollisjoner i mitt diagram pga. dimensjonene på diagrammet.

Ut fra begge diagrammene over kan man se at antall kollisjoner ikke har noe å si for kjøretiden egt. Det kommer an på hvilken åpen adresserings-metode man ser på. I den lineære kan man se at jo større fyllingsgrad, jo større antall kollisjoner og desto høyere tidsforbruk. Ser man på den doble, kan man se at den har lavest antall kollisjoner hele tiden, mens tidsforbruket er ca. som den lineære. Den kvadratiske har nesten likt antall kollisjoner som den doble, men klart best tidsforbruk jo større fyllingsgrad.

I en dårlig utnyttet hashtabell er lastfaktoren nesten lik 0, men i en full tabell er den lik 1.

```
Load factor: 0.8

Duration linear: 767ms
Number of collisions in the linear probe: 18701932

Duration square: 815ms
Number of collisions in the square probe: 11385386

Duration dubble: 880ms
Number of collisions in the double probe: 9726693
```

Som man kan se på bildet over har jeg en lastfaktor på 80%, som vil si at jeg fyller hashtabellene ca. 80%.

Den mest interessante observasjonen jeg har gjort er at den kvadratiske proben er den raskeste av alle når fyllingsgraden er 99%. Ut fra antall kollisjoner, så ville jeg trodd at dobbel skulle være raskest også, men den er nesten tregest hele tiden.