Leon Egeberg Hesthaug,
Eirik Elvestad, og
Trym Hamer Gudvangen

Øving 2 Eksponent Rekursjon

Analyse

Som følge av at algoritmen i oppgave 2.1-1 kun går 1 ned per rekursive kall, kan man trivielt se at denne algoritmen har en lineær tidskompleksitet. $\Theta(n)$
Masters Teorem beskriver forskjellige tidskompleksiteter basert på variablene som kommer fra algoritmen, og med bruk av denne finner vi tidskompleksiteten til algoritmen i oppgave 2.2-3. Algoritmen har dermed en logaritmisk tidskompleksitet $\Theta(\log n)$, se utregning under.

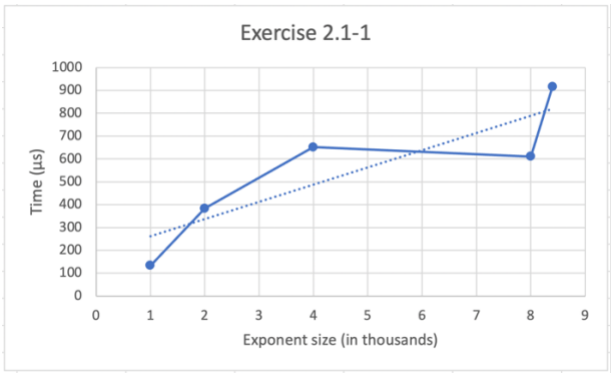
I oppgave 2.2-3 bruker vi en rekursiv algoritme som deler seg, derfor kan vi bruke Master Teorems formel:

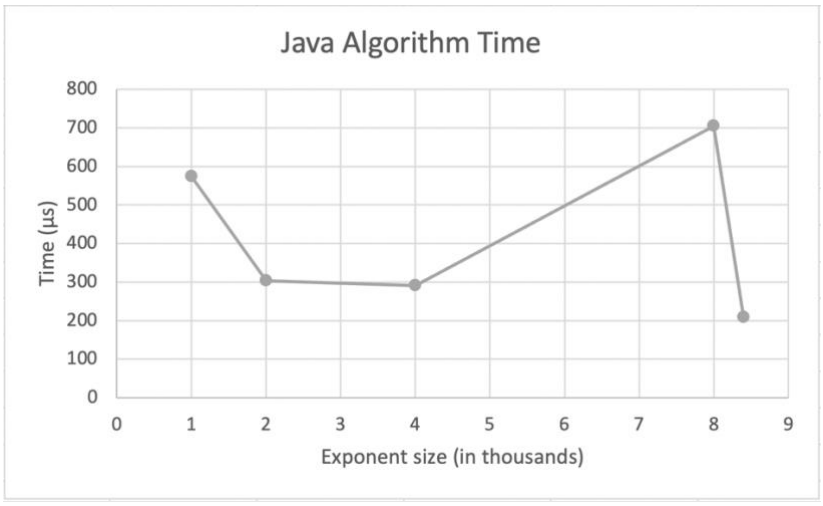
$$T(n) = aT\left(\frac{n}{b}\right) + cn^k$$
$$T(n) = 1T\left(\frac{n}{2}\right) + cn^0$$
$$T(n) = T\left(\frac{n}{2}\right) + 1$$

Siden størrelsen på datasettet (n) er delt på to for hver rekursiv kall, blir b = 2. Videre, siden algoritmen ikke har noe for-løkker så blir k = 0. Variabelen c er en konstant som er avhengig av datamaskinen som blir brukt og kan derfor bli sett bort i fra i en asymptotisk analyse.

$$b^k = a \rightarrow T(n) \in \theta(n^k \cdot \log n)$$
$$2^0 = 1 \rightarrow T(n) \in \theta(n^0 \cdot \log n)$$
$$T(n) \in \theta(1 \cdot \log n)$$
$$T(n) \in \theta(\log n)$$

Etter å ha analysert de to algoritmene, forventer vi å se en lineær sammenheng mellom mengden med datapunkter og tiden brukt i 2.1-1 algoritmen og en logaritmisk sammenheng i 2.2-3 algoritmen.





Det er viktig å merke seg at mengde med data var bare på tusen skalaen, som er relativt lite. Dette kan være en av grunnene for hvorfor det ikke finnes en tydelig trend.

Som forventet var 2.2-3 algoritmen, med sin $T(n) \in \theta(\log n)$ tidskompleksitet, betydelig raskere i kalkulasjonene sine enn både 2.1-1 og Java sin algoritme.