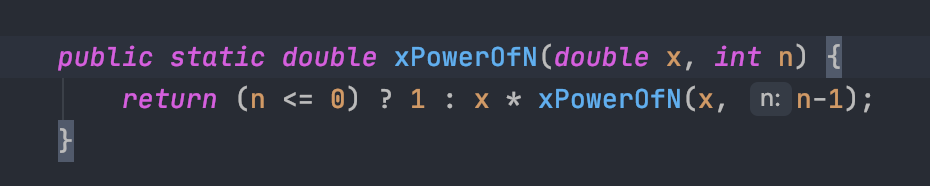
Øving 1 – Rekursiv Programmering

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X - Verdi | N - Verdi | Metode 1 | Metode 2 | Metode 3 | Resultat |
| 1.001 | 50 | 1.112 \* 10^-4 | 3.941 \* 10^-5 | 5.380 \* 10^-5 | 1.051245 |
| 1.001 | 500 | 8.851 \* 10^-4 | 4.914 \* 10^-5 | 6.299 \* 10^-5 | 1.648309 |
| 1.001 | 5000 | 0.0151 | 5.690 \* 10^-5 | 6.512 \* 10^-5 | 148.042 |
| 1.001 | 10000 | 0.0327 | 6.144 \* 10^-5 | 6.611 \* 10^-5 | 21916.681 |
| 1.001 | 10500 | 0.0337 | 5.938 \* 10^-5 | 6.688 \* 10^-5 | 36125.472 |

Tabellen over skal vise tidsforbruket fra de forskjellige algoritmene. Jeg har valgt å bruke samme x verdi og kun øke n verdien for å lettere kunne forstå hvordan tidsforbruket utvikler seg. Jeg har også valgt å inkludere avrundede resultater som algoritmene får, for å vise at de faktisk regner riktig. Når vi ser på tidsforbruket til de ulike algoritmene er det lett å se hvordan de ulike algoritmene vokser i tid.

**Metode 1**



I den første algoritmen bruker jeg en «conditional operator» hvor jeg først sjekker om N er mindre eller lik 0. Jeg inkluderte mindre enn i tilfellet n skulle være en negativ eksponent som hadde fått programmet til å gå i en evig loop. Hvis ikke n er mindre eller lik 0 så returner den x ganget med xPowerOfN igjen, hvor vi bruker samme x men n-1. For hver gjennomkjøring av algoritmen vil man trekke fra 1 fra n. Algoritmen vil derfor være lineært avhengig av n, **O(n),** ut ifra en asymptotisk analyse.

**Metode 2**

Et bilde som inneholder skjerm, overvåke, TV, holder

Automatisk generert beskrivelse

I den andre metoden har vi også en sjekk om a n er mindre eller lik 0. Etter det sjekker vi om tallet er et partall. Det gjøres da man benytter ulik n verdi i rekursjonen avhengig av om det partall eller ikke. Denne algoritmen baserer seg på rekursjon med halvert størrelse noe som har en tidskompleksitet av **O(log2(n))**.