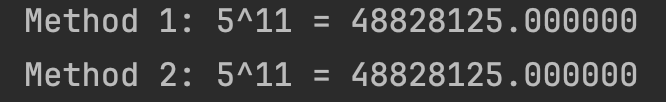
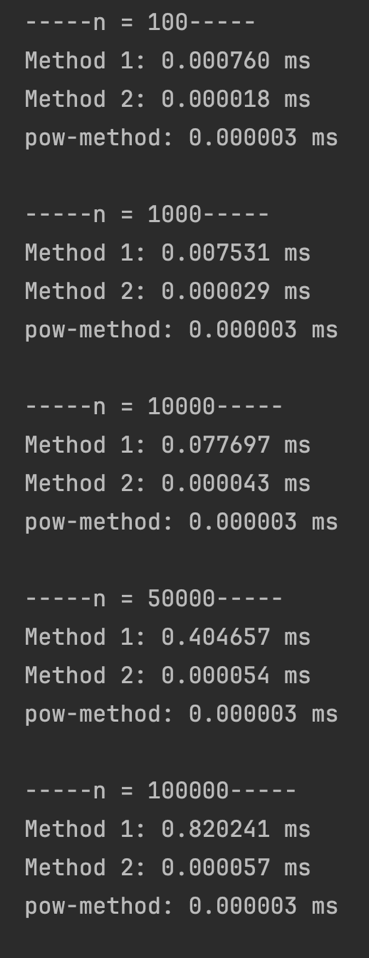
O2i

# Tidsmålinger og kalkulasjoner



Metode 1 og 2 får begge to det riktige svaret.



Jeg har gjort tidsmålinger med 4 ulike verdier for n som strekker seg fra 100 – 100 000.

Som man ser så varierer utføringstiden på metode 1 veldig på størrelsen på n. Det ser ut som en lineær tidskompleksitet.

Metode 2 på den andre siden endrer seg mindre, men det er fortsatt en økning på noen millisekunder når n tidobles.

# Asymptotisk analyse

Jeg skal nå foreta en asymptotisk analyse av alle metodene.

**Metode 1**

Metoden har en kontroll-setning (if-setning) som sjekker om n er lik en. Dette gjøres i konstant tid O(1). Når n > 1 gjøres det et rekursivt kall med n redusert med 1 helt til den treffer base casen. N minker altså lineært så det er sikkert å si at denne metoden har en tidskompleksitet på O(n). Dette stemmer overens med tidene.

**Metode 2**

Metode 2 har den samme kontroll-setningen som metode 1, som utføres i konstant-tid. Hvis n er et partall så gjør funksjonen et rekursivt kall med n / 2 og x\*x. Hvis n er et oddetall gjør funksjonen et rekursivt kall med (n-1) / 2 og x\*x, og multipliserer resultatet med x.

Hvert rekursive kall reduserer n med halvparten (n / 2 eller (n-1)/2). Antallet rekursive kall er proporsjonalt med til logaritmen til n med base 2. Derfor er tidskompleksiteten til metode 2 O(log n). Dette stemmer også overens med tidene. Tiden øker omtrent dobbelt fra n = 100 til n = 1000, imens det så vidt er noen endring fra n = 10000 til n = 100000.

**Metode 3**

Finner ikke dokumentasjonen på denne metoden, og hvordan den faktisk er skrevet. Men ut i fra tidsmålingene som alle er like ser det ut som at tidskompleksiteten er konstant, altså O(1). Men det er også mulig at det er en mer optimalisert versjon av metode 2, og at den dermed har O(log n).