Algoritmer og datastrukturer øving 2

Tidsmålinger

For alle målingene brukte jeg x = 100. Sum1 brukte metode 1, og sum2 brukte metode 2.

Måling 1, n = 10

```
Sum1:   Millisekund pr. runde:3.689751523277019E-5
Sum2:   Millisekund pr. runde:2.5330606206495002E-5
```

Måling 2, n=100

```
Sum1:   Millisekund pr. runde:2.2360264557706142E-4
Sum2:   Millisekund pr. runde:3.1144561061792836E-5
```

Måling 3, n=1000

```
Sum1:   Millisekund pr. runde:0.004184380544304221
Sum2:   Millisekund pr. runde:3.613661040834839E-5
```

Måling 4, n=10000

```
Sum1:   Millisekund pr. runde:0.05081042629947665
Sum2:   Millisekund pr. runde:4.072260636083039E-5
```

Gjør jeg målingene med n større enn 10000 får jeg en stack overflow error for metode 1. Metode 2 takler alle verdier for n.

Kjøretid for sum1:

For hver verdi av `n`, kaller den seg selv med `n-1`. Dette gjentas til `n` blir 1. Derfor er kjøretiden for denne metoden Θ(n).

Kjøretid for sum2:

For å forstå kjøretiden for `sum2`, kan vi se på hvordan `n` endres for hvert rekursivt kall:

1. Hvis `n` er et partall, blir `n` delt på 2: `n = n/2`.
2. Hvis `n` er et oddetall, blir `n` redusert med litt over halvparten: `n = (n-1)/2`.

Så I det verste tilfellet, som er når `n` er et oddetall hver gang. Blir `n` redusert med litt over halvparten for hvert kall.

Hvis `n` er 8 (som er et partall), vil det ta 3 kall for å redusere `n` til 1 (8 -> 4 -> 2 -> 1). Hvis `n` er 7 (som er et oddetall), vil det også ta 3 kall for å redusere `n` til 1 (7 -> 3 -> 1).

Dette mønsteret av å redusere `n` med omtrent halvparten for hvert kall er karakteristisk for logaritmisk vekst. Derfor kan vi si at kjøretiden for `sum2` er Θ(log n).