**Oppgave 1:**

|  |  |
| --- | --- |
| **Fib. Number** | **Real time (ms)** |
| f(41) = 165580141 | 1630 |
| f(42) = 297914296 | 2122 |
| f(43) = 433494437 | 3410 |
| f(44) = 701408733 | 5362 |
| f(45) = 1134903170 | 8320 |
| f(46) = 1836311903 | 13478 |
| f(47) = 2971215073 | 21632 |
| f(48) = 4807526976 | 34923 |
| f(49) = 7778742049 | 55898 |
| f(50) = 12586269025 | 90510 |

Rekusiv

**Hvorfor tidsbruken blir som den blir?**

Som vi kan se fra tabell, tid for beregning av ny fibanacci tall øker med «Fibonacci økning». Det vil si at de to forrige tidene adderes til hverandre og vi får tid til den tredje fibonacci tall. Som vi kan se i tabell f(41) har tid av 1630 ms, f(42) har tid av 2122.

1630 + 2122 = 3752, som er nesten som f(43) = 3410. Så derfor kan vi si at summen av to forrige tider skal vise tid til den neste med liten unøyaktighet.

Tid endrer seg eksponentielt fordi man bruker rekursiv beregning av fibonacci tall. Det vil si, men økning av fibonacci tall trenges det mye mer beregninger.

**Hvor mange primitive operasjoner utfører denne algoritmen for n like 1? n lik 2? n lik 3?**

-----------

**Plott:**