This is a timing test of 2 methods of computing large Fibonacci numbers.

The first definition is the expected recursive definition.

```
fibStep[{fn1_, fn2_}] := {fn2, fn1 + fn2}

fib1[n_] := Nest[fibStep, {0, 1}, n - 1][[2]]
```

First let's check it for the 10th entry.

```
fib1[10]
```

```
55
```

Now let's do this for the 50,000th entry and then the 300,000th.

```
Timing[N[fib1[50 000]]]
```

```
{0.109 Second, 1.077773489307297×10^10449}
```

```
Timing[N[fib1[300 000]]]
```

```
{1.938 Second, 8.761732532916346×10^62695}
```

Here is an alternate definition using the method of raising a suitably defined matrix to the power desired.

```
fib2[n_] := ({0, 1} . MatrixPower[{{0, 1}, {1, 1}}, n - 1])[[2]]
```

```
? MatrixPower
```

```
MatrixPower[mat, n] gives the nth matrix power of mat. More...
```

```
fib2[11] (* check for sanity *)
```

```
89
```

Now let's immediately try the 300,000th..

```
Timing[N[fib2[300 000]]]
```

```
{0.047 Second, 8.761732532916346×10^62695}
```

The resulting time is shorter than the time taken to compute the 50,000th using the other method. Finally let's look at the 20th (again for sanity).

```
fib1[20]
```

```
6765
```

```
fib2[20]
```

```
6765
```

```
MatrixPower[{{0, 1}, {1, 1}}, 20] // MatrixForm
```

$$\begin{pmatrix} 4181 & 6765 \\ 6765 & 10\,946 \end{pmatrix}$$

Note that these Fibonacci computation methods are well documented in, for instance more than one book by Roman Maeder as well as a book by Kevin Coombes.