Fibonacci numbers are defined recursively as follows:

- $F(0) = 0$
- $F(1) = 1$
- $F(n) = F(n - 1) + F(n - 2)$ for $n > 1$

Write the following methods to compute $F(n)$:

- A $O(2^n)$ method based on the recursive definition
- A $O(n)$ method that uses a loop
- A $O(1)$ method that uses the closed-form solution:

$$f(n) = \left\lfloor \frac{\alpha^n}{\sqrt{5}} + \frac{1}{2} \right\rfloor$$

where

$$\alpha = \frac{1 + \sqrt{5}}{2} = 1.6180339877...$$

Due to the finite precision of your computer, for large values of $n$ your third method might give you (slightly) different results from the others; this is normal.

Test all three methods using various values of $n$ and compare their running times (use the function `System.nanoTime()` to compute start and end time of each method call). If a particular algorithm and input combination does not return an answer in a reasonable amount of time, note that in your report (that is, don’t wait for hours (or worse) for your program to finish). To generate reliable statistics, run each method several times for each value of $n$ using a loop, discard the lowest and highest running times, and report the average of the remaining runs.

Write a report describing your work. We are particularly interested in your observations about the behavior of each algorithm as the size of the input data increases and also in the comparison of different algorithms for each input size. Use plots to illustrate this (see example below illustrating the running time of the recursive method). Also, feel free to run and report additional experiments for a more thorough analysis.