CS 101 - Algorithms & Programming I

Fall 2021 - HOMEWORK 2

Due: December 8, 2021

With this assignment, we aim to stress the fact that the same problem can be solved in multiple different ways, some of which are more efficient than others in terms of computational resources such as time.

The **Fibonacci numbers** are defined by the following recurrence:

$F_0 = 0, F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}, n > 1,$

where each Fibonacci number is the sum of the two previous ones: 0, 1, 1, 2, 3, 5 …

We would like to implement a static method that, given an integer n, calculates and returns F_n with the following interface:

```
public static int calcFib(int n) // returns nth Fibonacci number
```

Consider the two following algorithms for doing this:

Algorithm A

This algorithm is a recursive one (calls itself as needed to compute what's needed).

```
calcFibA(n)
    if n < 2 return n
    return calcFib(n-1) + calcFib(n-2)
```

Algorithm B

This algorithm, on the other hand, is non-recursive and relies on calculating Fibonacci numbers starting from the smallest ones and working up.

```
calcFibB(n)
    if n < 2 return n
    prev = 0
    this = 1
    i = 2
    while (i < n)
        next = prev + this
        prev = this
        this = next
        i++
    return next
```

Implement Algorithms A & B

First, implement/define the static method mentioned earlier using both algorithms calling them `calcFibA` and `calcFibB`, respectively.

Measure Running Times

Now we would like you to measure the running times of these methods as n increases (use the `nanoTime` method mentioned here). So, write a main method that is structured as follows:

- Makes calls to `calcFibA` for values of n from 0 to 50 with increments of 5 (0, 5, 10, …, 50), and measure and record the running times (in nanoseconds) in the following format:

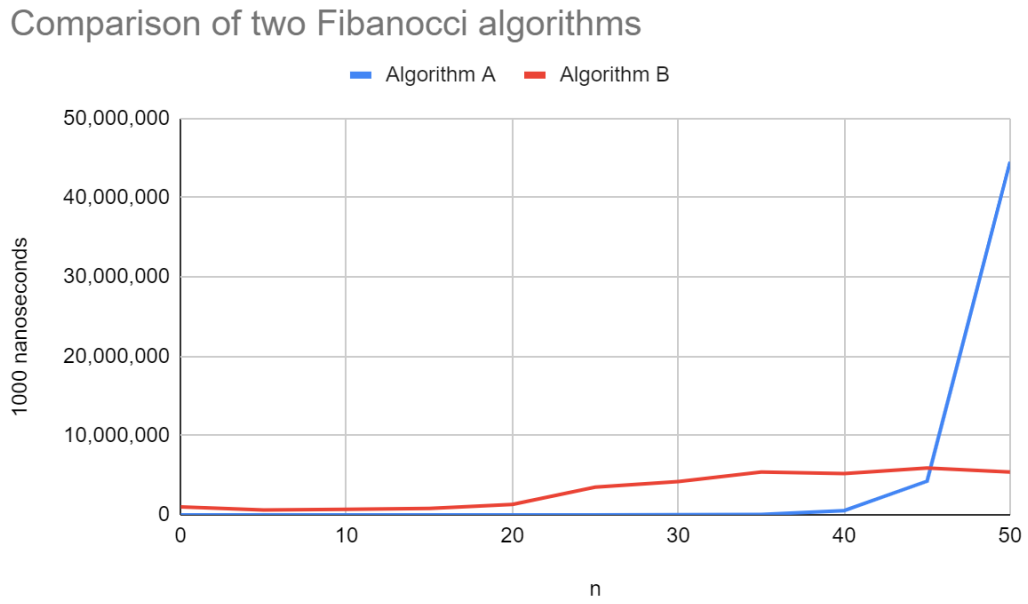
```
1     a1    // for n=0, F_0 is calculated in a1 nanoseconds
2     a2    // for n=5, F_5 is calculated in a2 nanoseconds
3     a3    …
```

- Makes calls to `calcFibA` for values of n from 0 to 50 with increments of 5 (0, 5, 10, …, 50), and measure and record the running times (in nanoseconds) in the following format:

```
1     b1    // for n=0, F_0 is calculated in b1 nanoseconds
2     b2    // for n=5, F_5 is calculated in b2 nanoseconds
3     b3    …
```

Discuss Results

Now that we have the outputs, plot the performance of both algorithms graphically (n vs time) in a single plot. A good way to format your plot is exemplified below:



Then, contrast the performance of the two algorithms in at most a few sentences and briefly comment on why this might be the case. Feel free to research this on the Internet but make sure to properly cite any references you make use of.

Also state what happens if you increase n from 50 to a larger number, say 100. Provide a brief explanation of why this might be the case.

Submission

You are to submit a single .java file (containing both methods along with your main method that performs these tests) and a PDF document (no hand written documents) that contains Results (text output of your test code, a plot for the output) & Discussion.