University of Calgary
Faculty of Science
Department of Computer Science

CPSC 331- Data Structures, Algorithms, and Their Analysis
Assignment 1
Efficiency and Time Analysis of Algorithms
Spring 2022

1 Objectives

This assignment has two main objectives: efficiency and time Analysis of Algorithms. It involves converting a recursive solution with multiple calls with the same input value(s) to a more efficient one using Memoization. Furthermore, it provides hands on experience in the analysis of algorithms by measuring the running times of three different solutions applied to the same computational problem on the same given data. The performance of these solutions will be illustrated using a simple graph.

2 Background

The Fibonacci sequence is used to model or describe an amazing variety of phenomena, in mathematics and science, art and nature. Originated in India, it was introduced to Europe by Fibonacci, also known as Leonardo Bonacci, Leonardo of Pisa, or Leonardo Bigollo Pisano, an Italian mathematician from the Republic of Pisa, who was considered to be "the most talented" Western mathematician of the Middle Ages. The famous Golden Ratio, spirals, and self-similar curves are sample of the mathematical ideas that stemmed from this sequence which have found its charm and beauty in a variety of applications in nature, art, and science. The figure below illustrates few areas influenced by Fibonacci sequence:

The Fibonacci sequence F_n could simply be generated by the following formula:

$$F_n = \begin{cases} 0 & \texttt{if} \quad \texttt{n} = 0 \\ 1 & \texttt{if} \quad \texttt{n} = 1 \\ F_{n-1} + F_{n-2} & \texttt{if} \quad \texttt{n} \geq 2 \end{cases} \quad (1)$$

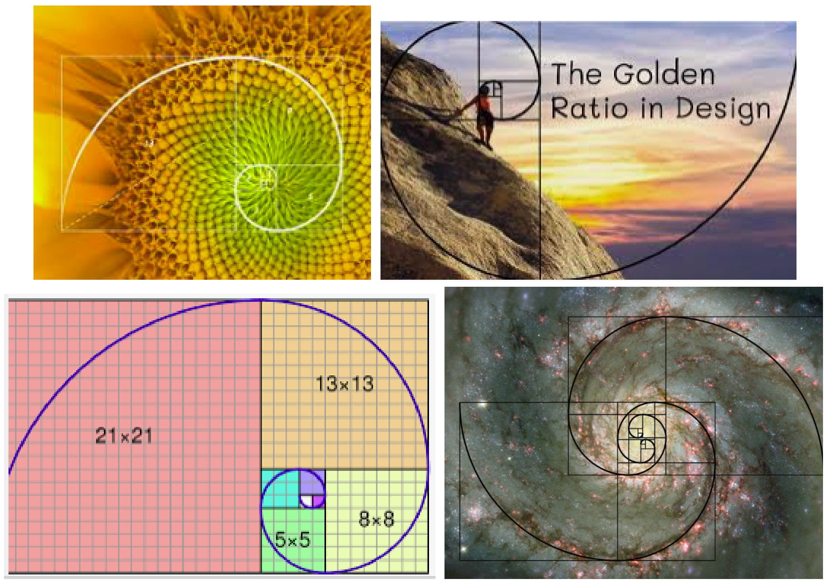

Figure 1: Fibonacci in Various Aspects of Life.

3 The Problem

3.1 Part 1

The simple solution to the Fibonacci number problem is the recursive one.

```
 integer fib (integer n) {
 if (n == 0) {
         return 0 }
else  if (n == 1) {
        return 1 }
        else  return  fib(n-2) + fib(n-1) }
```

However, this solution may duplicate the work by computing recursive calls multiple times with the same input value as shown in the Tracing tree below.

One solution to improve the efficiency of this algorithm is to use Memoization. This is a technique to improve the performance of a recursive solution by avoiding the repeated calls, mentioned above. The recursive algorithm can be rewritten so that previously computed values are stored in suitable Data Structure, and can be looked up rather than having to recalculate them.

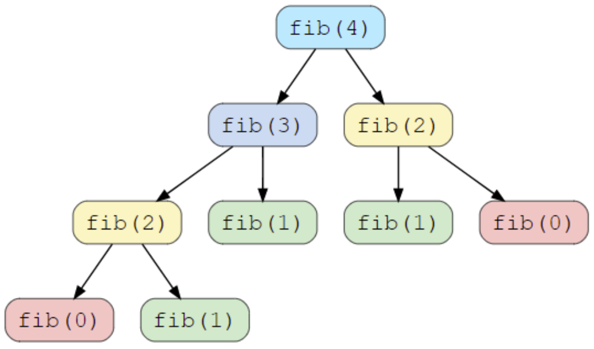

Figure 2: Tracing a recursive call

In this part, you need to provide a well-documented algorithm for computing the Fibonacci sequence using Memoization, and implement your solution in Java as per your TA instructor.

3.2 Part 2

In this part you need to implement three other different algorithms to solve the Fibonacci problem and incorporate a mechanism to measure the running time of each these algorithms. You need to generate an output file with the specified format given below, and plot the results on a graph for better analysis of the performance of these algorithms.

Algorithm 1 This is a recursive one:

```
Fib1(n : integer) {
  if n < 2 then return n
  else return Fib1(n-1) + Fib1(n-2)
  }
```

Algorithm 2 This is an iterative one attempting to avoid multiple calls.

```
Fib2(n : integer){
      i = 1
      j = 0
      for k = 1 to n do {
            j = i + j
            i = j { i
        }
        return j
    }
```

Algorithm 3 The third algorithm is a bit mysterious!

```
    Fib3(n : integer){
           i = 1
           j = 0
           k = 0
           h = 1
           while n > 0 do {
                  if n is odd then {
                       t  =  j*h
                       j  =  i*h + j*k + t
                       i = i*k + t
                       }
                   t  =  h*h
                   h  =  2*k*h + t
                   k  =  k*k + t
                   n  =  n div 2
            }
          return j
          }
```

4 Implementation Details

The following steps are helpful in your implementation[1]

1. For the Memoization solution, you need to store the values of the calculated Fibonacci values in a suitable data structure to avoid recalculate them again. Whenever a value needs to be computed, your algorithm should check this data structure for previously calculated values and use them instead. The data structure you will use in this case is a Map for rapid access of values.

   Your method signature may look as follows:

[1]Please follow the directions of your TA in the tutorials sessions, and as posted on D2L shell of the course

```
Public int fibMemoization(int n, Map<int,int> calculatedVals)
```

You should use effective Javadoc, describe loop invariants where necessary, and document your code.

2. For the second part, you need to do similar implementation for each of the three algorithms above but after incorporating in your programs a mechanism to measure the run time of each of these algorithms, i.e. the actual time it takes to calculate the same number across each different algorithms.

3. You need to generate one output file that has the following format:

```
Fib1 computes F0 in <time>
Fib1 computes F1 in <time>
...
<blank line>
Fib2 computes F0 in <time>
Fib2 generates F1 in <time>
...
<blank line>
Fib3 computes F0 in <time>
Fib3 computes F1 in <time>
...
```

4. The Fibonacci sequence grows very rapidly. So, use multiple-precision arithmetic. For instance, the 100th Fibonacci number has 21 decimal digits. Multiple-precision should help you avoid problems caused by arithmetic overflow. If multiple-precision (64-bit integers) is not possible, you can report only the first 7 digits of the Fibonacci number. This can be accomplished by performing all the computations modulo 107.

5. Plot the Fibonacci numbers on the x-axis and the time measured on the y-axis. Choose the values for n (x-axis) as directed by your TA.

6. Use a plotting tool to produce analyzable plots.

5 Grading

Grading will be assigned for all aspects, including efficient programming, documentation style, and the identification of runtime/storage complexity. The total grade of this assignment will initially be set to 51 points but will be factored in the final grade as per the outline of the course. The following guidlines will be used:

Part 1 This part will carry 19 points. They will be assigned for properly programming the Memoization function, as well as effective use of Javadoc, documentation, pseudo code of the algorithm, etc.

Part 2 This part will carry 32 points as follows:

1. 21 points for developing the algorithms correctly and having them output the correct values.
2. 11 points for the generation of a proper automated procedure for outputting files, and the plotting of Fibonacci values using a plotting module in Java. These must be well documented using Javadoc with clear task of each function.

6 Grading policy

Students could be randomly selected to demonstrate their programs and to show that they understand their codes. Work should be done individually. Any source of help MUST be indicated and cited.

Grading will be done as follows:

1. You will receive less than 20% of the total grade if your program compiles, but it doesn't work at all.
2. You will receive 0 points if your program will not compile.
3. For identical code, in part or full, the UofC rules and regulations for plagiarism will apply.
4. After the due time, a %20 deduction will apply on every late day or part of the day.

It's better to have something working, even with little functionality, than a big program that crashes (or doesn't compile). We expect to see your own program, otherwise the above will apply.

7 To Submit

You need to submit source files, executables, along with a *Makefile* that will be used to compile and link your shell. These files will be combined into a tar file that will unpack the files into a directory whose name is your user-id. The tar file will be submitted on the D2L page of the course.

Your submission should include (please follow the instructions of your TA):

- A file contains the fibMemoization.Java

- A file for each of the three algorithms, Fib1, FIb2, and Fib3 (code and executable).
- Samples of output files
- Graph Plots (could be on a pdf file), as per TA instructor.

You are URGED to check your submission after loading on D2L. Empty submission after the deadline will be subject to the indicated penalties.

8 Deadline

Sunday May 22, 2022 before 11:55 P.M. For each late day (in full or in part), a deduction of %20 will be applied.

9 Reference

`https://commons.wikimedia.org/wiki/`
`https://blogs.unimelb.edu.au/sciencecommunication/`
`CPSC 331, Jalal Kawash.`