

# Project 2

Spring 2022 CPSC 335 - Algorithm Engineering  
Instructor: Dr. Sampson Akwafuo

## Abstract

In this project, you will develop exhaustive search and approximation algorithms for solving the **Fibonacci problem**. The time complexity of your algorithms will be either exponential, linear or polynomial. The second part of the project involves implementing an exhaustive search algorithm for the **largest sum subarray problem** with polynomial efficiency class.

### Problem 1: The Fibonacci Problem and the Golden Ratio

In Fibonacci sequence, each member (or the Fibonacci number,  $F_n$ ) is obtained by calculating the sum of two preceding members. Example of a sequence of Fibonacci numbers is shown below:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots \quad (1)$$

Mathematically, the Fibonacci sequence contains series of  $F_n$  such that

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

There are some efficient algorithms for solving this problem. Some of these approaches will reduce the running time complexity while increasing the space requirement. The dynamic programming approach is one of them and will be discussed in subsequent weeks.

For this project, you are expected to develop two algorithms to solve the Fibonacci problem.

### A. The Exhaustive Pattern

The first approach will involve recursive calculation of each member of the sequence.

- Using the formula above, develop a recursive algorithm to calculate the  $n$ th terms of the sequence. The first term of the sequence is 0, as shown above.
- Print out the 15<sup>th</sup> term of the sequence.

## B. The Golden Ratio Approach

The ratio of any two successive members of a Fibonacci sequence i.e.  $\left(\frac{f(n+1)}{f(n)}\right)$  is known as the golden ratio,  $\Phi$ . This ratio approximates 1.61803.

A Fibonacci number (a member of the sequence) can be calculated using the formula:

$$F_n = \frac{(1 + \sqrt{5})^n - (1 - \sqrt{5})^n}{2^n \sqrt{5}} \quad (3)$$

In relation to a previous member of the sequence, the  $n$ th term of the sequence can be calculated using:

$$F_n \approx F_p \left( \frac{1 + \sqrt{5}}{2} \right)^{n-p} \quad (4)$$

Where  $F_p$  represents any of the preceding terms of  $F_n$ . Recall that  $\frac{1+\sqrt{5}}{2}$  is equal to 1.61803 and thus may be substituted.

An approximation of the next term of the sequence can also be obtained by multiplying the preceding term by the Golden Ratio

$$F_{n+1} \approx F_n \left( \frac{1 + \sqrt{5}}{2} \right) \quad (5)$$

There is a maxim that the ratio of two consecutive Fibonacci numbers approaches the Golden Ratio, as  $n$  gets bigger.

- a. Design algorithms to obtain the Fibonacci numbers using equations (4) and (5) above

When implementing equation (4), ask the user to enter '**p**'. You should specify that '**p**' should be a positive integer and non-floating point. If a wrong number is entered, inform the user and request for a new number, until a correct input is received. Then ask the user to enter '**n**'. Use the calculated  $f_p$  to obtain  $f_n$ .

When implementing equation (5), equation (3) should be used to find  $f_n$

- b. Print the first 20 terms of the sequence

- c. Compare your outputs when equation (4) is used against results from equation (5)
- d. Check and confirm or disprove the maxim above using  $F_3$  and  $F_{30}$  (from equation 5implementation)

## Part 2: The Largest Sum Subarray Problem

The problem involves finding a contiguous part of an array of numbers that adds up to the maximum possible sum.

<i>largest sum subarray problem</i>
<b>input:</b> a non-empty vector $V$ of $n$ integers
<b>output:</b> indices $b, e$ such that $0 \leq b < e \leq n$ , such the slice $V[b:e]$ has maximum sum

Recall that the notation  $V[b:e]$  means the part of  $V$  starting at index  $V[b]$  and up to, but not including,  $V[e]$ . Note that the solution  $V[b:e]$  may not be empty due to the constraint  $b < e$ .

Sample input = (-3, -5, 5, -1, -3, 1, 5, -6)

Sample output = (5, -1, -3, 1, 5 )

## Exhaustive Search Algorithm

The following straightforward exhaustive search algorithm considers all pairs of indices  $b, e$ .

```
largest sum (V):
    b = 0, e = 1
    for i from 0 to n-1:
        for j from i+1 to n:
            if sum(V[i:j]) > sum(V[b:e]):
                b = i, e = j
    return (b, e)
```

## To Do

1. Produce a written project report **in PDF format**. Your report should include:
  - a. Your name(s), CSUF-supplied email address(es), and an indication that the submission is for project 2.

2. Develop the pseudocode for Part 1A and 1B, and implement the **three** algorithms (Pseudocode for Part 2 is already given) in Python or C++
3. Your codes should be saved and submitted in the corresponding executable file.
4. Mathematically analyze each algorithm and state the big  $O$  efficiency class
5. Using the given sample inputs, print 4 resulting largest sum sub-arrays. Include them in your PDF report

Sample inputs =      (10, 2, -5, 1, 9, 0, -4, 2, -2);  
                               (-7, 1, 8, 2, -3, 1);  
                               (9, 7, 2, 16, -22, 11);  
                               (6,1, 9, -33, 7, 2, 9, 1, -3, 8, -2, 9, 12, -4)

## Grading Rubric

The suggested grading rubric is given below.

### Part 1A (35)

- a. Clear and complete Pseudocode = 10 points
- b. Mathematical analysis and correct Big  $O$  efficiency class = 5 points
- c. Successful compilation of codes= 15 points
- d. Produces accurate result (15<sup>th</sup> term) = 5 points

### Part 1B (45)

- a. Clear and complete Pseudocode, using equation 4 and 5 = 10 points
- b. Mathematical analysis and correct Big  $O$  efficiency class = 5 points
- c. Successful compilation of codes= 15 points
- d. Checking if the user input, **p**, is a non-floating point positive integer = 5 points
- e. Printing the first 20 terms of the sequence = 5 points
- f. Comparison of outputs from both equations = 5 points
- g. Confirming or disproving the maxim = 5 points

### Part 2 (20)

- a. Mathematical analysis and correct Big  $O$  efficiency class = 5 points
- b. Successful compilation of codes= 10 points
- c. Printing the 4 resulting largest sum sub arrays = 5 points

Ensure your submissions are your own works. Be advised that your submissions may be checked for plagiarism using automated tools.

## **Submitting your code**

Submit your files to the Project 2 Assignment on Canvas. It allows for multiple submissions. You can submit your files separately or as a zip folder (do not use .rar)

## **Deadline**

The project deadline is **Sunday, March 27, 11:59 pm** on Canvas.

Penalty for late submission (within 48 hours) is as stated in the syllabus. Projects submitted more than 48 hours after the deadline will not be accepted.