# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Spring 2022** |
|  | | | | | |
| Points |  | Document author: | **MingkuanPang** | | |
|  | Author’s email: | **Yafking20 @mail.fresnostate.edu email** | | |
| Laboratory number: | **03** | | |
|  | | | | | |

**1. Statement of Objectives**

This lab asks for calculate the nth Fibonacci number by using two different algorithms, one is inefficient, and another one is efficient. In this report, I will describe the procedures that I wrote in the program and the way I implemented them.

**2. Experimental Procedure**

**Fibonacci (Implement in recursive way, inefficient):**

Text

Description automatically generated

According to the formula of Fibonacci:

F (0) = 0 and F (1) =1

F(n)=F(n-1) +F(n-2)          for n>1

We can transfer it into code straight forward.

**Fibonacci (Implement in iterative way, efficient):**

Text

Description automatically generated

According to the formula of Fibonacci, if N is less or equal than 1, we return it. Otherwise, for Nth Fibonacci number, we keep calculating F(k) = F(k-1) +F(k-2), until K = N.

**Procedure add (adding two numbers which in string format):**

A screenshot of a computer

Description automatically generated with medium confidence

In this procedure, it travels two strings from back. Then it uses string[i]- ‘0’ to make it to be integer.

Just like we do the adding by hand. we sum those two integers and update the mover. Then it put the digit result into a new string. If the string having more digits to go through after another string done its travelling, then for the longer string, we keep adding the mover to the rest of digit number until the mover is 0. After two string finish travelling, if the mover is not zero, then we put the mover into the first place of the result string.

**3. Analysis**

**Input: (30th Fibonacci number)**

A screenshot of a computer screen

Description automatically generated with medium confidence

**Output:**

Text

Description automatically generated

By the result, we can see the procedure implemented by iterative algorithm is much efficient than the recursion one.

**Input: (120th Fibonacci number)**

A screenshot of a computer screen

Description automatically generated

**Output:**

Text

Description automatically generated

For getting 120th Fibonacci number, the iterative algorithm only needs 0.022 seconds. Meanwhile the recursive algorithm seems like it just hovers right there for a very long time. It seems like the recursive algorithm is not able to get the 120th Fibonacci number with my computer.

**4. Encountered Problems**

The Fibonacci number getting larger and larger as we are going further, and even the INT\_MAX is not able to hold that large number. Therefore, I transferred those numbers to string format. That way we can handle those numbers with regardless of its size basically.

**6. References**

I did not use any references in this report.