# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Fall 2021** |
|  | | | | | |
| Points |  | Document author: | **Marlon Branham** | | |
|  | Author’s email: | **zekebranham@mail.fresnostate.edu** | | |
| Laboratory number: | **Laboratory 2** | | |
|  | | | | | |

1. **Statement of Objectives**

In this lab, we are to create a program that produces the Fibonacci Sequence at a value “n.” Firstly by using a natural solution that we feel is correct for the problem, then the more efficient algorithm to find the solution. After both functions are created, we will be recoding the time it takes to run the program to the 120th Fibonacci number. The major significance of this lab is to compare a somewhat time-consuming procedure using recursion methods versus iterative methods and see which is more efficient. These goals were accomplished though this lab successfully other than some minor issues that will be discussed in the “Encountered Problems” section of this report. In this report, I will be going over my experimental procedure, analysis, encountered problems, and of course my conclusion.

**2. Experimental Procedure**

This section will be broken down into two parts since we were to use two different methods to execute the Fibonacci Sequence. Both parts included methods to calculate elapsed time of the execution but will be discussed further in the “Analysis” section.

The first part will be discussing the recursive “natural solution” to the problem. By natural solution, I refer to the brainstorming I came up with personally to attack this problem, which in turn would be recursive functions. Usually when I am dealing with a problem that requires repeated arithmetic or repeated patterns of any kind, recursion is the first thing that comes to mind. I created a function called “int FibonacciRec” that takes in the integer “n” that will go to the nth position in the Fibonacci Sequence. My function returned the arithmetic sum of the original function call with n-1, and the original function call with n-2 to create the sequence. In main, I simply created a variable within a “while” loop that kept updating until the user inputted value “n” was reached in the sequence then displayed the variable.

A screen shot of a computer screen

Description automatically generatedA computer screen with colorful text

Description automatically generated

In the second part of the experiment, we are to accomplish the same task but instead of recursion, use a more efficient method, that being an iterative method. Initially, I picture anything iterative to be slower and less efficient than a method of recursion, but when dealing with large amounts of data with exponentially increasing arithmetic, iterative methods are much faster than recursive. In doing so, I created a function called “void FibFor” that, as the title hints at, uses a “for” loop to create the Fibonacci Sequence. The function simply takes in the user inputted “n” value for what index of the Fibonacci Sequence to display, and uses variables created within the function but outside of the loop, to execute the arithmetic of the sequence. In main, the function is called with value “n” and displays the nth index of the Fibonacci Sequence.

A screen shot of a computer program

Description automatically generated

A computer code on a black background

Description automatically generated

**3. Analysis**

My results follow in accordance with the instructions for the lab assignment. Although the assignment was to find the runtime of both functions with the 120th index of the Fibonacci Sequence, the runtime required for this using recursion is enormous and theoretically impossible, so I did the 40th index of the Fibonacci Sequence instead to complete the assignment in a timely manner. Otherwise, the program would still be running for a very long time in the future. My results showed that the iterative function was much quicker and more efficient than the recursive. Of course, the result of the nth value is the same for both functions but the runtime for the recursive function was around 2.511 seconds with only 40 values, yet the runtime with the iterative function was less than 1 second with 40 values. This function being consistently under 1 second through testing up to 120 values.

A screen shot of a computer

Description automatically generated

**4. Encountered Problems**

The main issue I faced, as discussed above in previous sections, using the recursive method, the runtime required for the program using 120 values is immense and completely unpractical for this lab, not to mention my computer not being able to process it. I could not “solve” this solution any other way than to decrease the input size to 40 values, as was an alternative mentioned during the in-person lab. Other than the number of values, there were no issues with finding my ways though these requirements.

**5. Conclusions**

In this lab, we used two methods to generate the Fibonacci Sequence to a user inputted value. The two being recursively and iteratively. What was found was that the recursive method is far more inefficient than an iterative method. The iterative method was so much more efficient that it was able to handle all 120 values easily, while my computer could not handle the strain it would take to process those steps recursively. Overall, I now understand the iterative methods are better for handling large amounts of data that all need actions taken on them. A successful lab with a successful outcome.

**6. References**

none