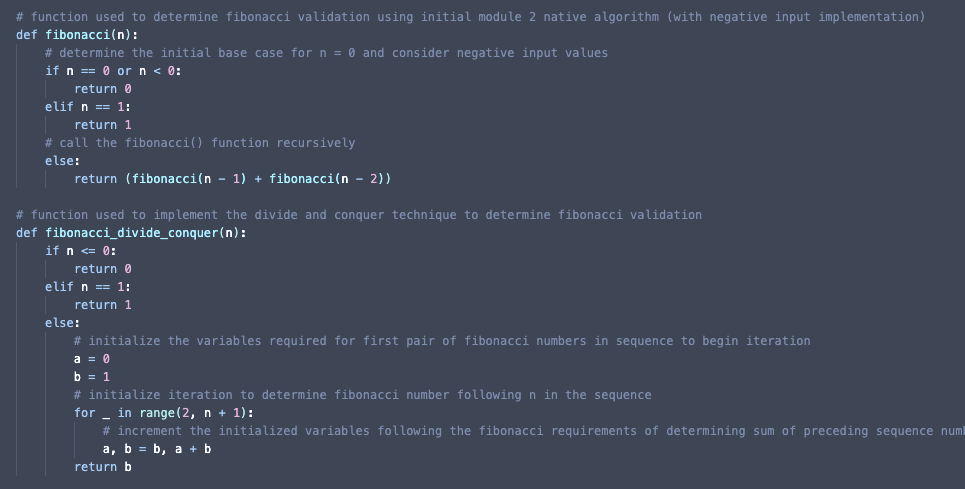
Eric Ramsey

CSCI 3202

Module 2 Assignment

Spring 2024

For a number to be considered a Fibonacci Number and part of the Fibonacci Sequence, an integer or number must follow the rule that each number is equal is equal to the sum of the preceding two numbers. This sequence starts with zero and can continue infinitely. For this assignment I chose to use Python programming and implement Unit Testing methods to determine the accuracy of my program logic, implement the Fibonacci native recursive algorithm provide in this module and also implement the divide and conquer technique in determining if an input ***n*** is in fact a member of the Fibonacci Sequence. I chose to implement both to examine the results of the native recursive algorithm and compare them to the results of my divide and conquer function.



The image above contains the two functions used in this assignment (and contained in my assignment submission) to determine if an integer ***n*** is considered a member of the Fibonacci Sequence. The first function **fibonacci(n)** accepts an integer ***n*** and is derived from the native recursive algorithm regarding Fibonacci numbers in this module. However, during testing I realized that I was not considering the possibilities of negative integer inputs, which would be against the rule of a Fibonacci Sequence values and would result in the final output of zero. I decided to alter the original algorithm and include negative input determination in the initial base-case conditional statement to compare test results in the divide and conquer function as well as be more proactive in my testing. The second function I used in the **fibonacci.py** file that was implemented to test the divide and conquer technique is the **fibonacci\_divide\_conquer(n)** function. This function accepts an integer ***n*** and if it does not meet the base requirements of the conditional statements, implements the divide and conquer technique to determine if the integer ***n*** is a member of the Fibonacci Sequence. Since the requirement of an integer to be considered a member of the Fibonacci Sequence involves preceding numbers, the divide and conquer technique located in the ***else*** section of the function’s conditional statemen, divides the two starting values of the Fibonacci Sequence into separate variables. These variables, ***a*** and ***b***, are updated accordingly based on the contained ***for loop***, which begins with the value ***2*** to account for the starting values of the Fibonacci Sequence ***0*** and ***1***, while include the range of ***n + 1*** to further test the required rule. As the process of the loop proceeds the variables are incremented by setting the variable ***a*** to the current value of ***b*** and setting the new value of ***b*** to the sum of ***a + b*** which implements the requirement of the Fibonacci Sequence numbers. While developing the required functions for this assignment, I implemented the file ***test\_fibonacci.py*** to determine the accuracy of the function’s logic and also identify any changes that could be made to increase their efficiency.

A screen shot of a computer program

Description automatically generated

The image above contains the test functions implemented in the ***test\_fibonacci.***py file, used to evaluate the accuracy and efficiency of previously mentioned functions. Using the information given in the ***module 2 resources*** regarding Fibonacci Sequence and Numbers, and also information contained in the resources listed below, I was able to determine what test cases to implement to effectively determine the accuracy of my functions. I began each test function with the base-cases of the Fibonacci Sequence and proceed with the following numerical values. Because the **test\_fibonacci\_divide\_conquer()** function was testing a function based of my own knowledge and not using the provided native recursive algorithm, I chose to implement more test cases. I chose to use values contained in the Fibonacci Numbers section located in the ***module 2 resources***, which are include the following: ***[(F6, 8), (F7, 13), (F8, 21), (F9, 34)].***

A diagram of a company

Description automatically generated

In the module resources, the Fibonacci native recursive algorithm is tested by what appears to be a recursive tree method by testing ***fib(6)***. Since the integer value ***5*** is tested in my unit test file, I chose to visualize the analysis of the divide and conquer function’s recurrences using the recursive tree method, located in the image above (and also in the LucidCharts link below). Based on the evidence given using the recursive tree method, the number of calls using this function doubles unless the value of input ***n*** contained in ***fib(n)*** is greater than 1. This gives the conclusion that depending on the input of the function, the growth of that function will be to the ***nth*** of that functions input. Therefore, since we established that the growth of the function doubles when ***n*** is greater than ***1***, it can be stated that this function has the time-complexity of ***O(2n)*** for determining if a value is a member of the Fibonacci Sequence.

**Resources:**

* <https://www.techtarget.com/whatis/definition/Fibonacci-sequence>
* <https://www.youtube.com/watch?v=zg-ddPbzcKM>
* <https://www.youtube.com/watch?v=6tNS--WetLI>
* <https://cse.buffalo.edu/~shil/courses/CSE531/Slides/DivideAndConquer-NA.pdf>
* <https://lucid.app/lucidchart/f8f339bc-81ac-4fa4-81be-631aad16d93c/edit?viewport_loc=-323%2C-127%2C2251%2C1309%2C0_0&invitationId=inv_c012a053-5446-478a-8321-9c26b89f387b>