Ziaul Choudhury.

Project: Integrative learning, Fibonacci, Big-O, and Fibonacci Prime.

Fibonacci Sequence was first introduced by an Italian mathematician named Leonardo Pisa. In his book, Liber Abaci (1202) introduced the name Fibonacci (Hom). Fibonacci sequence is a series of number(s) where after two starting values (commonly 0 and 1) each number came after starting values that are the sum of two proceeding numbers. For example: assume that we need to find x(n) and in this case n=5. The first two integers of this Fibonacci series are 0 and 1, or x(1) = 0 and x(2) = 1. To find x(n) = x(3), we add two sequences that came right before x(3) which are x(1) and x(2). So, in this case x(1) + x(2) = 0+1 = 1 = x(3). Similarly, we use the same technique to find x(5) = 1+2 = 3 (as x(3) = 1 and x(4) = 2). As we can see that to find any Fibonacci sequence, we actually use fowling expressions which can be mathematically expressed as: x(n) = {[x(n-1)] + [x(n-2)]}.

It is possible to find Fibonacci sequence in programming by using recursion. The definition of recursion is the act or process of returning. In the recursive part of my C++ program (Stated at the end), I have created a recursive function called “int Fibonacci (int n)”. In that function, the base case is given n = 0 and n = 1 which are the first and second elements of Fibonacci sequence. The reason for using these two statements is to tell when recursion to stop, so the recursion doesn’t run forever in an infinite loop and return 0 or 0 1 and to print only first or second element(s) of the Fibonacci sequence based on condition given. The next statement is in the recursive function is to calculate x(n) as described earlier and print the Fibonacci Sequence. If the user wants more than two elements of the Fibonacci Sequence.

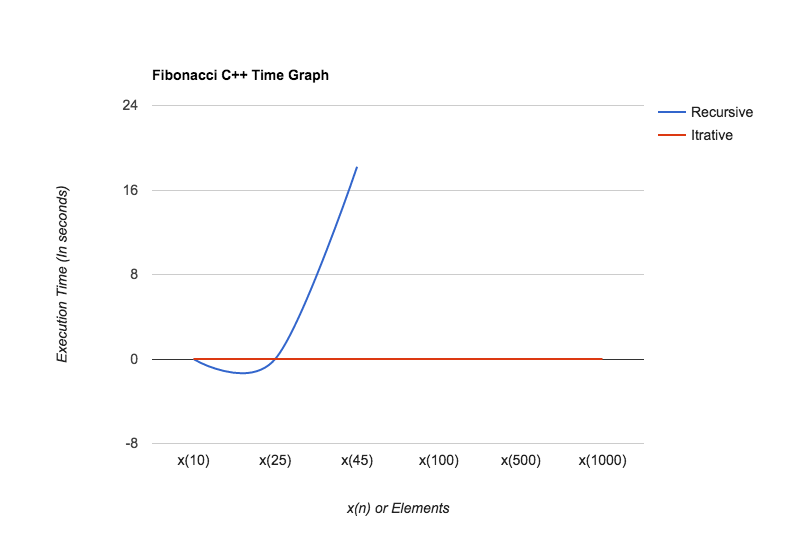
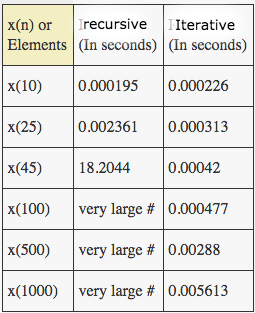
It basically tells the program that when the positive nth element entered by the user, run the for loop and print as long as “i <= elements” condition remains true and print the result using Fibonacci function. For example: let our nth element entered by the user is 4. In the ‘do-while’ loop and ‘if statement’ will verify if the value entered is a positive integer or not if the value is not positive the ‘do while’ loop will keep run until a positive nth integer element for Fibonacci series has has been entered. In our case the value is a positive integer, so the condition has been met. As a result, the program will ignore the ‘if statement’ in main function move on to the ‘for loop’ which is next instructions in the main function. In the ‘for loop’ the initial value of ‘i = 1’; on the next part, it will compare condition “i <= elements” (1<=4). In that case, the condition is true so, the ‘i’ will get an increment, so now is i = 2, and prints first element of the Fibonacci sequence by using recursive Fibonacci functions. In the next step the loop will follow the same procedures where the condition is (2<=4) and ‘i’ incremented to i=3. Now it will print second element of Fibonacci sequence. The loop will run and print third and fourth elements of the Fibonacci sequence by following the same procedure until ‘i’ gets incremented to (i = 5), in this case the condition given on the for loop “i <= elements” (5<=4) is false so, the loop will stop.

Using iterative method in programming is efficient process for printing Fibonacci number(s). The definition of iteration is repeating the process. In the iterative part of my C++ program (Stated at the end), I have created a function called “void fibonacci()”. In that functions, the ‘do-while’ loop allows the user to enter a positive integer to print nth element(s) of Fibonacci sequence. It will also verify whether the integer is a positive integer, if not it will continue to ask user the enter a positive integer. Once a positive integer has been entered, it will exit the ‘do-while’ loop and move on to the for loop to produce desired element(s) of the Fibonacci sequence. In the for loop the initialization of the loop set ‘i=1’ and condition is given “i <= elements” and ‘i++’ will increment ‘i’ each time by 1 as the loop runs prints x(n) where n is a single element of Fibonacci Sequence. In the for loop we store ‘first’ and the ‘second’ element in the ‘temp variable’ which is current x(n). Then second becomes first and temp becomes the second variable and prints next sequence of the Fibonacci series which is stored in the current first variable. I have used “unsigned long long” for declaring data type for the variables to handle very large values.

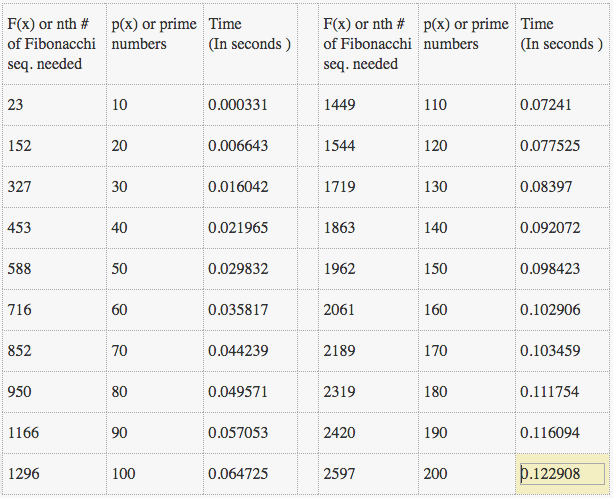
For example: lets enter 4 as user desired input. As the user input is a positive integer, it will check conditions and conditions are met in “do-while” loop. Then it will move to ‘for loop’ to print first four elements of Fibonacci sequence. In the for loop (1<=4) later (2<=4) as it gets incremented so, the condition “i <= elements” is true and i = 2, i = 3 are incremented. This will print the first element and second element of Fibonacci sequence ‘0’ and ‘1’ as given as the initial conditions and move on to the next part of the loop which is to switch values and calculation. In this case, temp is (0+1) as the second = 1, so the temp = 1 now and second value has moved into first. In next part temp becomes second and prints 1 as the third elements of Fibonacci sequence. In the for loop (3<=4) and i=4. Since, the first value remains unchanged (first=1) and second is (second=temp=1). So, on the iteration of the loop it follows the same process as (temp = first + second = 1 + 1 = 2) and prints the fourth elements on the screen. As i finally gets incremented to 5 and “i<=elements” (5<=4) is false, so the for loop finishes. In the main functions we just basically call the “void fibonacci()” functions to operate since a void functions can’t do anything unless it is being called in the main function.

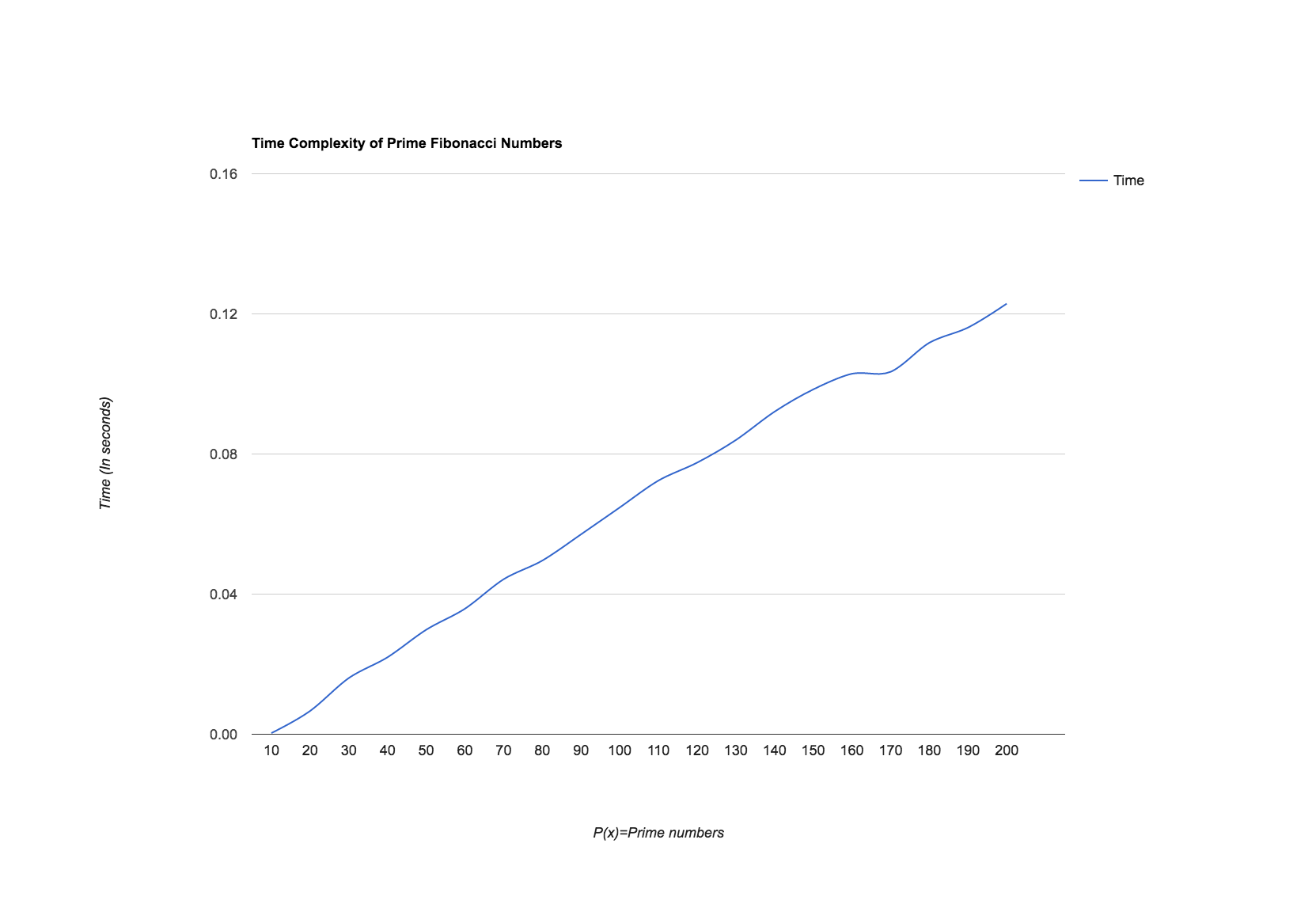
To run both of my codes, I have used many tests such as entering both positive and negative nth elements, entering small values, entering very large values, and etc. To begin writing my code, at first I had to learn how Fibonacci numbers are produced and then I created the algorithms for both programs. After, I added the conditions to both of program to produce Fibonacci Sequence. I have also added conditions to accept only positive nth elements for Fibonacci Sequence in both program. While running the program, I have found that iterative functions for Fibonacci much more efficient then recursive. To produce 100 sequences of Fibonacci iterative part of code takes less than a second, but the recursive part will take lot longer to produce those 100th Fibonacci Sequences. The reason recursion takes longer is because it creates copy of its all previous elements which requires more memory and hardware.

On the other hand, iteration is faster is because it takes less memory and hardware. Each time the loop iterates it doesn’t need to copy all the vales in the memory after the values being printed. Also, the values of Fibonacci numbers are changing depending how many elements or numbers of Fibonacci sequence needed to produce and the values are stored for short period of time.



Form the table and graph above showing the time complexity of both programs while running them. For the first 25 elements, both of code/program runs efficiently. After that iterative method works better as form the graph and table, and reference graph form the book named “Discrete mathematics and its applications” ( page 221 figure 3). We can conclude that the iterative function is Big-O(n) since the growth of the iterative function is linier. Also, for the recursive method we can conclude that the function is Big-O(2n) since the graph shows exponential growth.

After analyzing the efficiently of programs, I used both part of my programs to determine and print prime Fibonacci numbers. Prime numbers are numbers that are only divisible by itself and one. I have used Boolean to determine prime numbers in both programs and in the for loop after Boolean checks whether the Fibonacci number(s) is/are prime or not. I’ve also included two variables ‘sum’ and ‘k’ to count how many prime numbers are being produced in the Fibonacci sequence. Form the graph and table below implies that the growth of execution time in iterative program and number of prime numbers in Fibonacci series is very low. The graph is also a Big-O(log n) where n grows very slowly for large values of p(x) = prime numbers. I didn’t use recursive function because it’ll take very long to find those large numbers shown on the table and associated graph below. 



In both program, I have also used “ctime” library to calculate the execution time. Additionally, the smallest Fibonacci number greater than 1,000,000 is **1346269** which is the 32nd nth element of Fibonacci series and the smallest Fibonacci number greater than 1,000,000.000 is **1134903170** which is 46th nth element of Fibonacci sequence. I have found those numbers by using ‘if statements’ according to their conditions.

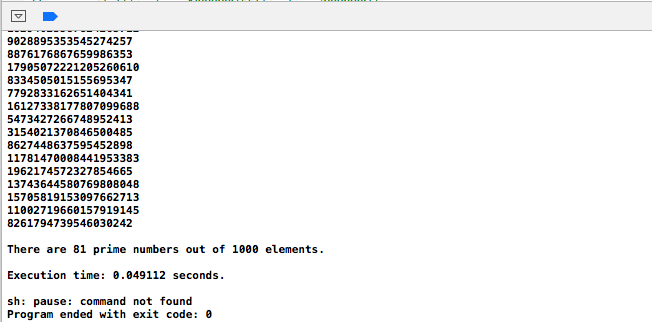
The applications Fibonacci sequence is widely present in our science and nature. Some of applications of Fibonacci sequence are counting flower pallets, the golden ratio, growth of rabbit population, branching trees, planting, and stock market behavior. According to Investopedia, Fibonacci tool can be used to predict market by 70% success rate when applied correctly. Traders uses Fibonacci ratios, most commonly 24%, 38%, 50%, 62%, and 100% (rounded) to track growth or decay of price in stock market and to invest in the stock market. The major usages of Fibonacci in stock market is to find support or resistance level and identify price targets. For example: suppose you are applying Fibonacci ratio in down tread to find an investing time. To examine its ratio, you need to pick a starting point and an ending point. Suppose the starting point of the price 10 days ago was 100 units and now it is 23 units. When the price was 23 units you have invested and after 10 days of investment the price went up to units 62. As you can see the price has revised after 23 unites. At 23 unites it hits the investors the uptrend that in one of common ratio stated earlier. However, solely depending on Fibonacci ratio doesn’t always guarantee profit growth, we need to consider other factors in the market too.

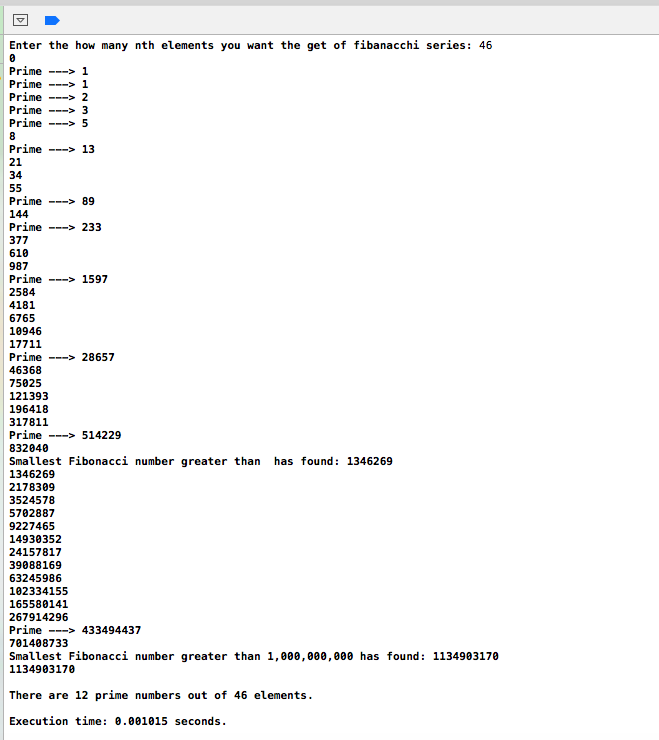
Fibonacci sequence is very poplar topic in world of mathematics and computer science. While doing the assignment most challenging part of this assignment were the creating the algorithm and finding prime Fibonacci numbers. To resolving my issues, I had to analyze how Fibonacci numbers are being produced and apply this analogy create algorithms. The problem I was having was printing prime numbers. At the beginning, I provided a meted to calculate prime numbers by using enhanced for loop but the compiler was printing same prime multiple times. Then I realized that I have already used for loop earlier to produce Fibonacci sequence, so don’t need to use another for loop to produce numbers again. So, removed that for loop and

used produced Fibonacci numbers inside other for loop and used Boolean for complier to determine Fibonacci prime sequence.

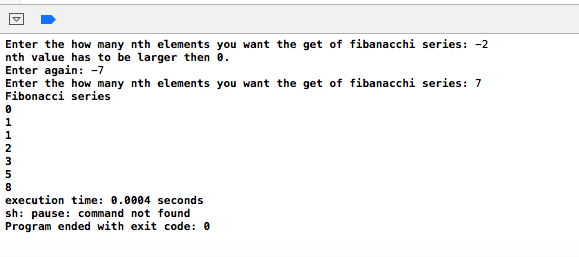
**Outputs:**

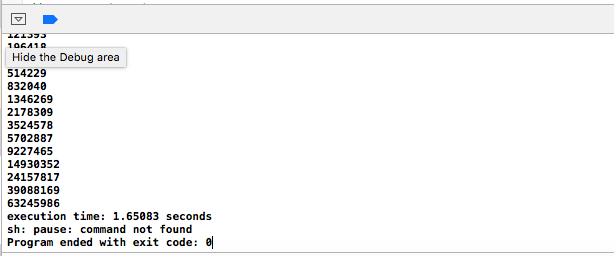
**Program 1: An iterative C++ function that inputs a nonnegative integer *n* and returns the *n*th Fibonacci number.**

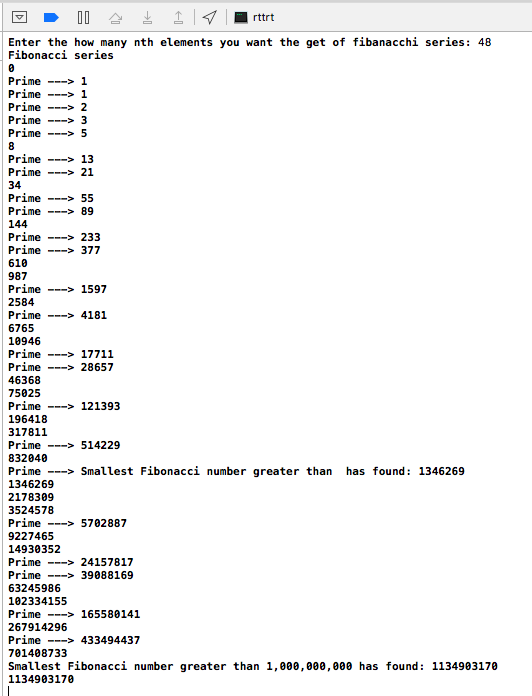


****

**Program 2: A recursive C++ function that inputs a nonnegative integer *n* and returns the *n*th Fibonacci number.**

****

****

****

**Works Cited**

Aspray, Thomas. "Fibonacci Analysis: Master the Basics." MoneyShow.com. N.p.,

17 Sept. 2013. Web. 07 May 2016.

Rosen, Kenneth H. "Ch 3, Algrithms." *Discrete Mathematics and Its Applications*.

7th ed. Boston: McGraw-Hill Higher Education, 2007. N. pag. Print.

Hom, Elaine J. "What Is the Fibonacci Sequence?" LiveScience.

TechMedia Network, 14 June 2013. Web. 06 May 2016.

Vonko, Dima. "Taking The Magic Out Of Fibonacci Numbers | Investopedia."

Investopedia. Investopedia, LLC., 07 Aug. 2006. Web. 07 May 2016.

"ICS 161: Design and Analysis of Algorithms Lecture Notes for January 9, 1996.

Fibonacci Numbers. UCI, 9 Jan. 1996. Web. 05 May 2016.

<https://www.ics.uci.edu/~eppstein/161/960109.html>.

“Cplusplus Forum” Web. 05 May 2016 <http://www.cplusplus.com/>