INDIAN INSTITUTE OF TECHNOLOGY GOA

Name: Aditya Rajesh Bawangade

Roll Number: 2103111

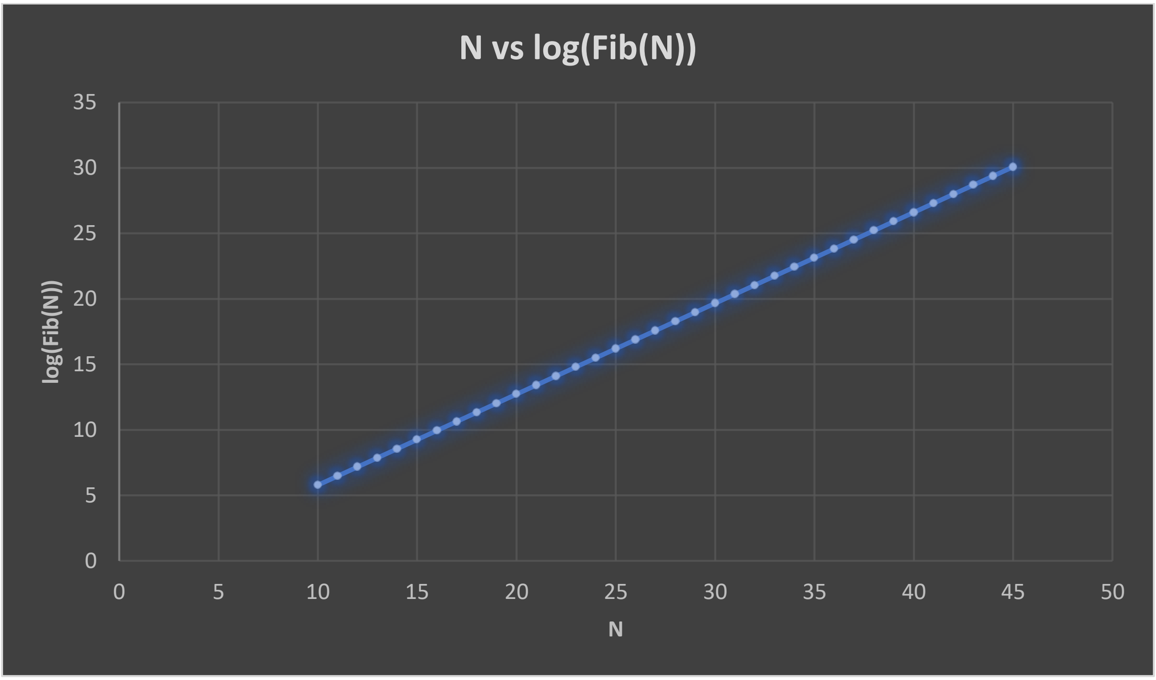
<u>The output of the program written for Q.1]</u>

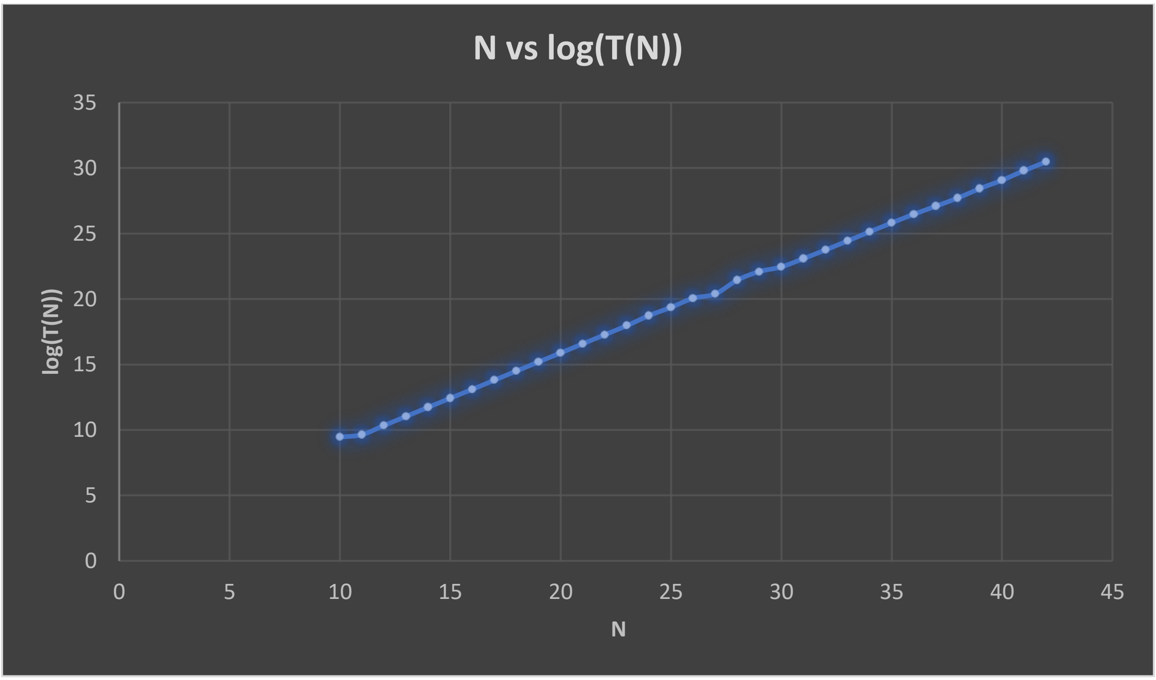
```
[Running] cd "d:\LECS AND MATERIAL\SEMESTER 4\Algorithm Design\Assignment2\" && g++ fibonacci_recursive_21031_11.cpp -o fibonacci_recursive_21031_11 && "d:\LECS AND MATERIAL\SEMESTER 4\Algorithm Design\Assignment2\"fibonacci_recursive_21031_11
Time taken by the recursive implementation of the 10th fibonacci number is: 500 nanoseconds
Time taken by the recursive implementation of the 11th fibonacci number is: 700 nanoseconds
Time taken by the recursive implementation of the 12th fibonacci number is: 1100 nanoseconds
Time taken by the recursive implementation of the 13th fibonacci number is: 1800 nanoseconds
Time taken by the recursive implementation of the 14th fibonacci number is: 2300 nanoseconds
Time taken by the recursive implementation of the 15th fibonacci number is: 3700 nanoseconds
Time taken by the recursive implementation of the 16th fibonacci number is: 5600 nanoseconds
Time taken by the recursive implementation of the 17th fibonacci number is: 9000 nanoseconds
Time taken by the recursive implementation of the 18th fibonacci number is: 14400 nanoseconds
Time taken by the recursive implementation of the 19th fibonacci number is: 23100 nanoseconds
Time taken by the recursive implementation of the 20th fibonacci number is: 37200 nanoseconds
Time taken by the recursive implementation of the 21th fibonacci number is: 59700 nanoseconds
Time taken by the recursive implementation of the 22th fibonacci number is: 108600 nanoseconds
Time taken by the recursive implementation of the 23th fibonacci number is: 156100 nanoseconds
Time taken by the recursive implementation of the 24th fibonacci number is: 255900 nanoseconds
Time taken by the recursive implementation of the 25th fibonacci number is: 408600 nanoseconds
Time taken by the recursive implementation of the 26th fibonacci number is: 666800 nanoseconds
Time taken by the recursive implementation of the 27th fibonacci number is: 1075800 nanoseconds
Time taken by the recursive implementation of the 28th fibonacci number is: 1737400 nanoseconds
Time taken by the recursive implementation of the 29th fibonacci number is: 2807800 nanoseconds
Time taken by the recursive implementation of the 30th fibonacci number is: 4545800 nanoseconds
Time taken by the recursive implementation of the 31th fibonacci number is: 7356500 nanoseconds
Time taken by the recursive implementation of the 32th fibonacci number is: 11912000 nanoseconds
Time taken by the recursive implementation of the 33th fibonacci number is: 23101800 nanoseconds
Time taken by the recursive implementation of the 34th fibonacci number is: 32760100 nanoseconds
Time taken by the recursive implementation of the 35th fibonacci number is: 51388500 nanoseconds
Time taken by the recursive implementation of the 36th fibonacci number is: 84813800 nanoseconds
Time taken by the recursive implementation of the 37th fibonacci number is: 132866900 nanoseconds
Time taken by the recursive implementation of the 38th fibonacci number is: 229052300 nanoseconds
Time taken by the recursive implementation of the 39th fibonacci number is: 351917400 nanoseconds
Time taken by the recursive implementation of the 40th fibonacci number is: 591228900 nanoseconds
Time taken by the recursive implementation of the 41th fibonacci number is: 1025066300 nanoseconds
Time taken by the recursive implementation of the 42th fibonacci number is: 1701851200 nanoseconds

[Done] exited with code=0 in 15.532 seconds
```

Ln 26, Col 19   Spaces: 4   UTF-8   CRLF   C++   Win32

Q.1]

The graphs obtained by storing the data from the recursive implementation of the Fibonacci numbers in a CSV file and plotting them using excel are as follows:





1.1

Each number in the Fibonacci series, which often begins with 0 and 1, is the sum of the two numbers before it. The recurrence relation

F(n) = F(n-1) + F(n-2)

defines the Fibonacci sequence, where

F(0) = 0 and **F(1) = 1.**

Clearly, with the increasing values of n, the growth of the Fibonacci numbers is exponential. It is also evident from the graph of **N vs log(F(N))** which is a straight line that, the Fibonacci numbers grow exponentially with N.

The time required to calculate the n-th Fibonacci number using this recurrence relation increases exponentially with n, as is well known. This is due to the function's ability to call itself twice in a recursive manner, each time with a problem of size n-1 and n-2, which causes exponential growth in the number of recursive calls. When utilizing this recurrence relation to calculate the n-th Fibonacci number, the time required to do so rises exponentially with n since the number of recursive calls grows exponentially with the input size.

1.2

The slope of the first graph can be calculated as follows:

$$\frac{\log(\mathrm{Fib}(31)) - \log(\mathrm{Fib}(28))}{31 - 28} = M1$$

M1 = 0.694233

The slope of the first graph can be calculated as follow:

$$\frac{\log(\mathrm{T}(31)) - \log(\mathrm{T}(28))}{31 - 28} = M2$$

M2 = 0.5483

1.3

It is pretty clear from the previous question that the function F(N) can be written as follows, Since, the fraction log(F(N))/N = 0.694233, thus:

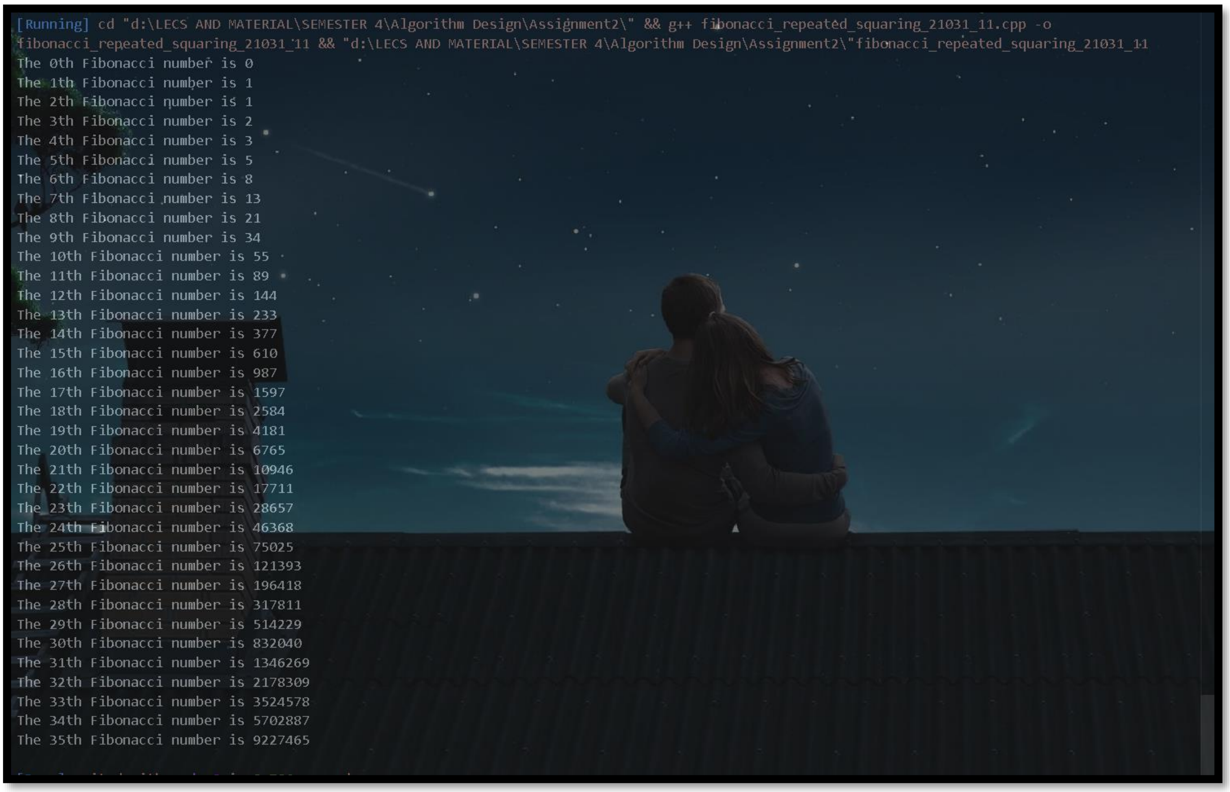
$$F(N) = 2^{0.694233N}$$

1.4

Similar to the previous question the function T(N) can be written as follows, Since, the fraction log(T(N))/N = 0.5483, thus :

$$T(N) = 2^{0.5483N}$$

Q.2]



```
[Running] cd "d:\LECS AND MATERIAL\SEMESTER 4\Algorithm Design\Assignment2\" && g++ fibonacci_repeated_squaring_21031_11.cpp -o fibonacci_repeated_squaring_21031_11 && "d:\LECS AND MATERIAL\SEMESTER 4\Algorithm Design\Assignment2\"fibonacci_repeated_squaring_21031_11
The 0th Fibonacci number is 0
The 1th Fibonacci number is 1
The 2th Fibonacci number is 1
The 3th Fibonacci number is 2
The 4th Fibonacci number is 3
The 5th Fibonacci number is 5
The 6th Fibonacci number is 8
The 7th Fibonacci number is 13
The 8th Fibonacci number is 21
The 9th Fibonacci number is 34
The 10th Fibonacci number is 55
The 11th Fibonacci number is 89
The 12th Fibonacci number is 144
The 13th Fibonacci number is 233
The 14th Fibonacci number is 377
The 15th Fibonacci number is 610
The 16th Fibonacci number is 987
The 17th Fibonacci number is 1597
The 18th Fibonacci number is 2584
The 19th Fibonacci number is 4181
The 20th Fibonacci number is 6765
The 21th Fibonacci number is 10946
The 22th Fibonacci number is 17711
The 23th Fibonacci number is 28657
The 24th Fibonacci number is 46368
The 25th Fibonacci number is 75025
The 26th Fibonacci number is 121393
The 27th Fibonacci number is 196418
The 28th Fibonacci number is 317811
The 29th Fibonacci number is 514229
The 30th Fibonacci number is 832040
The 31th Fibonacci number is 1346269
The 32th Fibonacci number is 2178309
The 33th Fibonacci number is 3524578
The 34th Fibonacci number is 5702887
The 35th Fibonacci number is 9227465
```

Because the power function squares the matrix recursively and cuts the size of the problem in half at each step, its time complexity is **O(log n).**

The time complexity of the multiply function, which multiplies two 2x2 matrices, is **O(1)** because it only requires four n-bit integer multiplications and additions.

The Fibonacci function's overall time complexity is then

O(log n) * O(1) = O. (log n).

The multiply function's time complexity in terms of **M(n)** is **O(M(n))**, making the Fibonacci function's overall time complexity in terms of **M(n)** equal to **O(log n * M(n))**.

The time complexity of **M(n)** is dependent on the algorithm used to multiply two n-bit integers, but it is typically **O(n^2)** or **O (n log n)**