ESO207A – PROGRAMMING ASSIGNMENT 1

ASSIGNMENT DONE BY:

Hamza Masood
210403
Dept. EE

Amay Raj
210116
Dept. EE

THE OBJECTIVE OF THE ASSIGNMENT

1. To investigate whether the efficiency of an algorithm really matters in real world?

2. To verify how well the RAM model of computation captures the time complexity of an algorithm.

BACKGROUND OF THE ASSIGNMENT

Fibonacci numbers have many applications in theoretical as well as applied computer science. They are used in pseudo-random number generators, Fibonacci heap data structure, analyzing efficiency of Euclid's algorithm etc. Fibonacci numbers are also found in natural patterns like flower petals.

Let F(n) denote nth Fibonacci number. We had written a code for Fibonacci numbers during the course ESC101A. Consider a related computational problem whose input is an integer n, and output is (F(n) mod 2021). In the lecture, three algorithms for solving this problem were discussed. The first algorithm was recursive and denoted by RFib. The second algorithm was iterative and denoted by IFib. And the third algorithm, CleverFib was quite complex – it involved repeated squaring to compute some power of a cleverly defined 2 × 2 matrix.

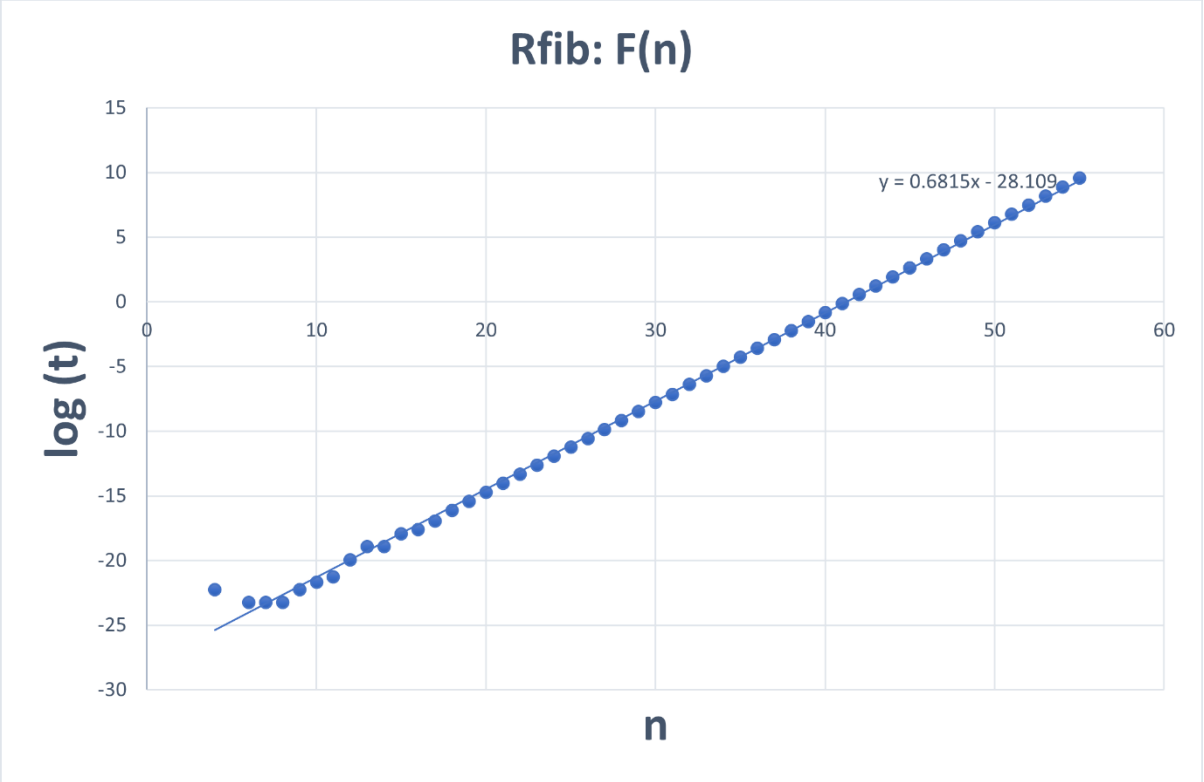
The implementation of each algorithm is supposed to work for each possible value of n up to 10^{18}. However, it will turn out that some of these algorithms will start taking too much time as n increases.

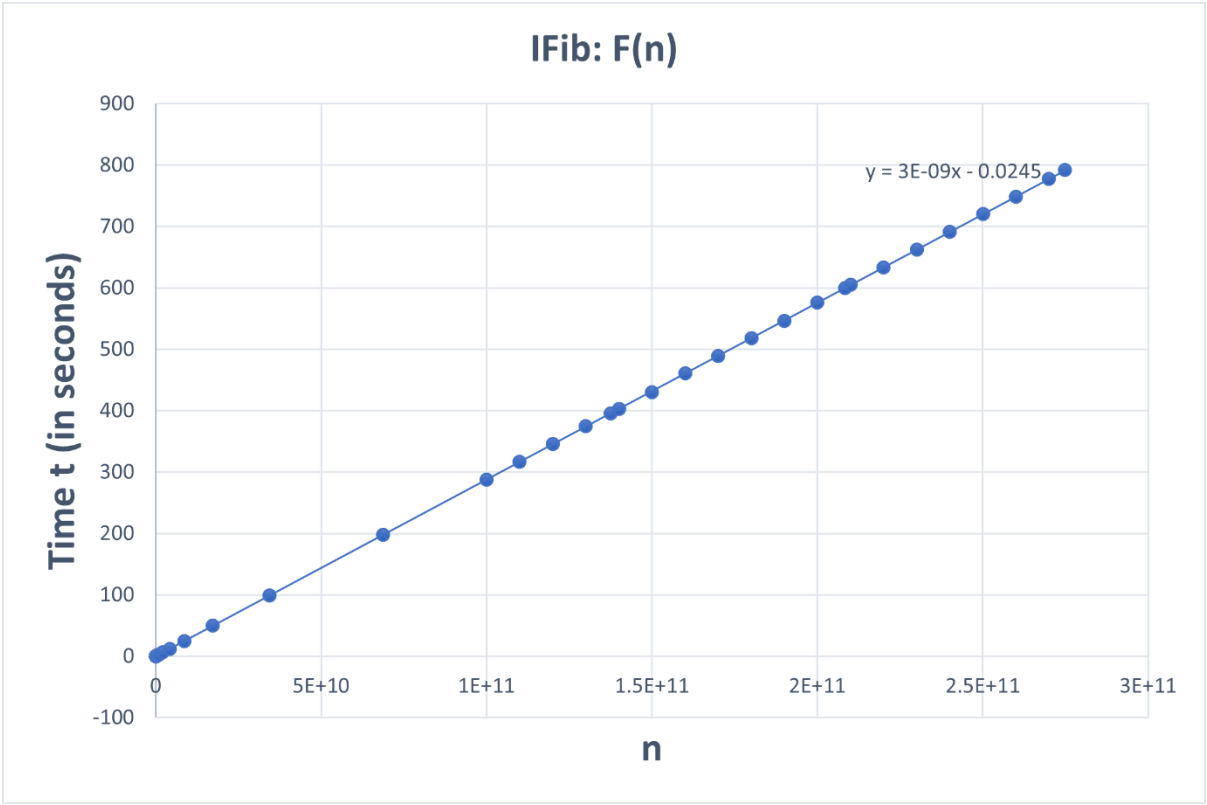
TASK 1:

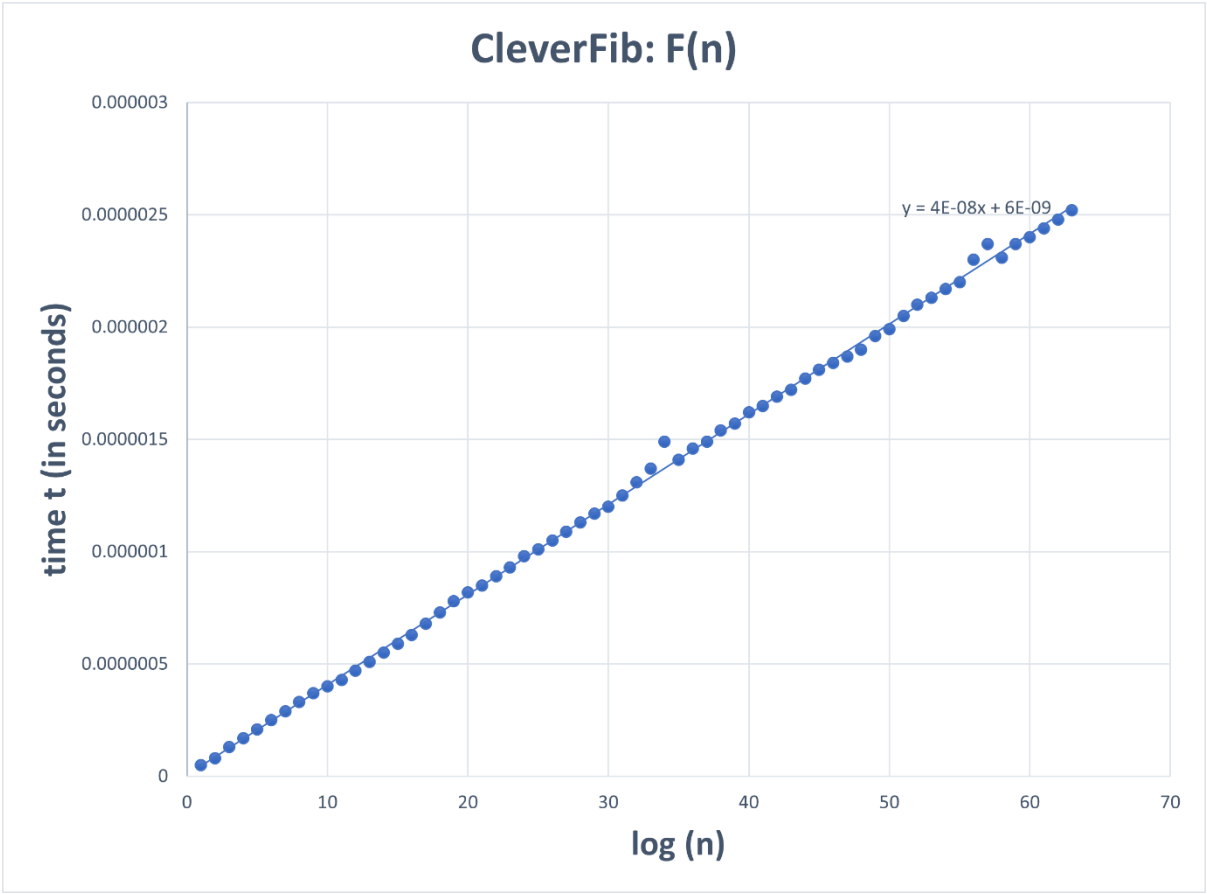
Time →	0.001 sec	0.1 sec	1 sec	5 sec	60 sec	600 sec
RFib	26	36	41	44	49	54
IFib	319999	30005000	$3.21*10^{8}$	$1.6*10^{9}$	$2*10^{10}$	$2.08*10^{11}$
CleverFib	$>10^{18}$	$>10^{18}$	$>10^{18}$	$>10^{18}$	$>10^{18}$	$>10^{18}$

TASK 2:

GRAPHS:




IFib: F(n)
y = 3E-09x - 0.0245
Time t (in seconds)
900
800
700
600
500
400
300
200
100
0
-100
0
5E+10
1E+11
1.5E+11
2E+11
2.5E+11
3E+11
n


CleverFib: F(n)
y = 4E-08x + 6E-09
time t (in seconds)
0.000003
0.0000025
0.000002
0.0000015
0.000001
0.0000005
0
0
10
20
30
40
50
60
70
log (n)

SOME PERTINENT QUESTIONS

Ques 2. (a): Provide precise and concise justification for the shapes of the graphs you obtain.

Ans: The Word RAM model predicts the following number of instructions required for each of the algorithms:

Algorithms for Fib(n) mod 2021	No. of instructions in Word RAM model
RFib(n)	$>2^{(n-2)/2}$
IFib(n)	3n
CleverFib(n)	$37 \log_2(n-1) + 11$

RFib(n): Since $\log_2$(no. of instructions executed by RFib) is directly proportional to n and each instruction takes almost same time for executing, the graph comes out to be approximately a straight line.

IFib(n): Since no. of instructions executed by IFib is directly proportional to n and each instruction takes almost same time for executing, the graph comes out to be approximately a straight line.

CleverFib(n): Since no. instructions executed by CleverFib is directly proportional to $\log_2(n)$ and each instruction takes almost same time for executing, the graph comes out to be approximately a straight line.

Ques 2. (b): If a graph is a line, what is the value of its slope? If each graph is a line, can you provide an explanation for the difference in their slopes?

Ans:

Graph	Slope
$\log_2$(time taken by RFib) vs n	0.6815
time taken by IFib vs n	$3*10^{-9}$
time taken by CleverFib vs $\log_2(n)$	$4*10^{-8}$

The slope is varying because of the dependent and independent variables being used as x and y to plot the curve.

$$\textbf{slope} = \Delta y / \Delta x$$

For RFib: Δy is $\log_2(t)$ and $\log_2(t)$ varies from -23.2535 to 9.578782. Δx is n and n varies from 1 to 55

For IFib: Δy is t and t varies from 0 to 791. Δx is n and n varies from 1 to $2.78 * 10^{11}$

For CleverFib: Δy is t and t varies from 0 to $2.52 * 10^{-6}$. Δx is $\log_2(n)$ and $\log_2(n)$ varies from 1 to 63.

As it is evident from the analysis, despite all the plots coming almost a straight line, the slope of all the plot will vary accordingly.

Ques 2. (c): In each call of CleverFib, the number of instructions executed (excluding the recursive call invoked) are significantly more than RFib. Moreover, CleverFib involved multiplication operation whereas the other two algorithms involved only addition operation. We know that multiplying a pair of numbers takes more time than adding a pair of numbers on a computer.

(i) Did these facts influence the running time of CleverFib?
Ans: The number of instructions executed by CleverFib per recursive call are definitely more than RFib, but the number of recursive calls are significantly lesser in CleverFib, which makes it much more efficient than RFib as the graph of time taken by RFib vs n would be exponential whereas the graph of time taken by CleverFib vs n would be logarithmic. Moreover, multiplying very few number of times takes much less time than adding numbers very large number of times.

(ii) Did these facts affect the relative speed of CleverFib compared to the other 2 algorithms? If not, then state the reason.
Ans: No, these facts did not affect the relative speed of CleverFib compared to IFib and RFib. This is because, CleverFib takes much less recursive calls to return the final answer compared to both IFib and RFib. For any given n, the number of recursive calls needed by CleverFib are $\log_2(n-1)$ whereas that for IFib are n and for RFib are even more than $2^{(n-2)/2}$.

TASK 3: ACCURACY OF THE RAM MODEL OF COMPUTATION

The RAM model of computation was very accurate in comparing the efficiency of the three algorithms.

With the number of instructions executed decreasing, the time required for calculating the nth Fibonacci sequence also decreased almost proportionally. This was evident from the data received and graphs plotted for the three algorithms.

FEEDBACK FOR THE ASSIGNMENT

This assignment was an eye-opener for me. I found it immensely useful.

It picked up a very simple problem we had already done in ESC101A using the iterative and recursive methods which were not very efficient and demonstrated the power of an efficient algorithm. I was amazed by the time it took for the CleverFib algorithm to calculate the Fibonacci numbers for such large values of n.