Assignment 1

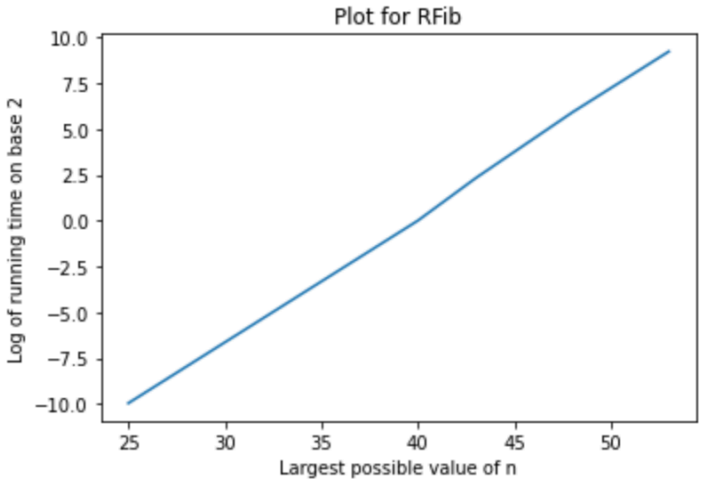
Task 1

Time	0.001 sec	0.1 sec	1 sec	5 sec	60 sec	600 sec
RFib	25	35	40	43	48	53
IFib	1.5E+05	1.62E+07	1.62E+08	8.13E+08	1E+10	1E+11
CleverFib	> 1e18	> 1e18	> 1e18	> 1e18	> 1e18	> 1e18

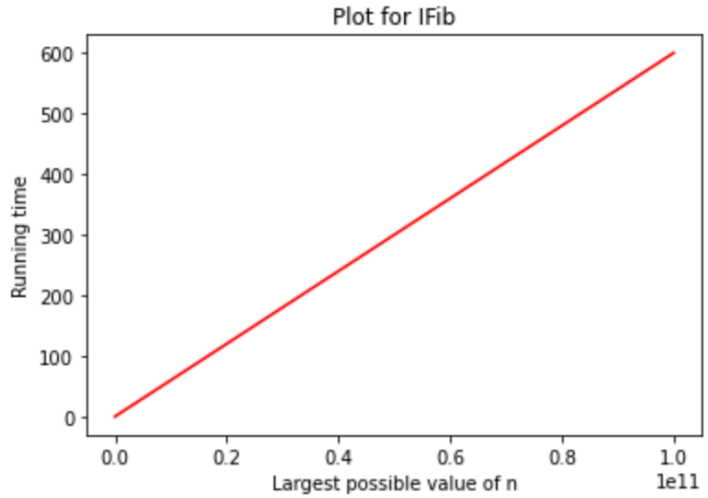
Task 2

The graphs were plotted using Matplotlib in Python.

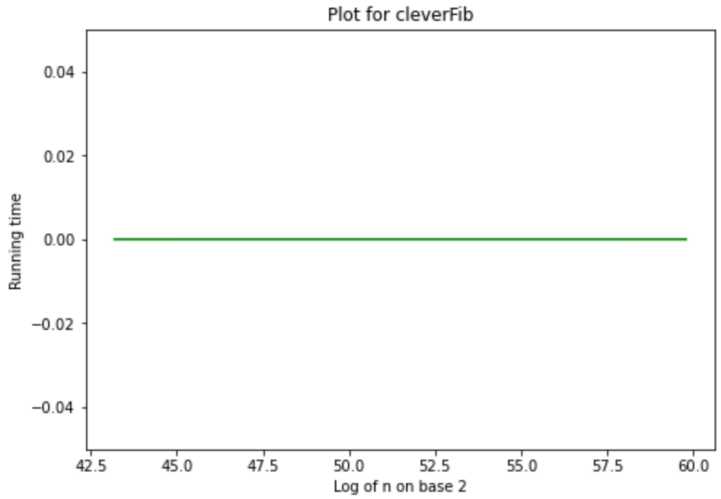
(i)



(ii)



(iii)



(a)

(i) In RFib for n>1,

T(n)=T(n-1)+T(n-2)+c

To get an estimate for an upper bound, we can assume T(n-2)~T(n-1) as T(n-2)<=T(n-1):

T(n)=2T(n-1)+c

Using this recurrence relation, we get

T(n)=2^k*T(n-k) + (2^k-1)*c

So we see that the time taken is approximately proportional to 2^n and hence we get a straight line.

(ii) In IFib for n>1,

Total number of instructions = 6+3(n-1)

T(n)~kn where k is some positive constant

So we see that time taken is approximately proportional to n and hence we get a straight line.

(iii) In cleverFib (for n>1,

Number of instructions = $k*\log_2(n)+c$

where k is the number of instructions in the recursive call to compute the exponent matrix and c encompasses the instructions which involve the assignment of the initial 2x2 matrix and the final return statements.

$T(n)\sim k\log_2(n)$

Thus CleverFib is highly efficient in terms of time complexity and can compute across the entire range of long long int in less than 0.001s and hence we approximately get a horizontal line.

(b)

```
slopeR, interceptR = np.polyfit(RFib,log2t, 1)
print("The value of slope for the graph of RFib is",slopeR)
```

```
The value of slope for the graph of RFib is 0.6897384334524178
```

```
slopeI, interceptI = np.polyfit(IFib,time, 1)
print("The value of slope for the graph of IFib is",slopeI)
```

```
The value of slope for the graph of IFib is 5.9996614278160355e-09
```

```
slopeC, interceptC = np.polyfit(log2n,times, 1)
print("The value of slope for the graph of cleverFib is",slopeC)
```

```
The value of slope for the graph of cleverFib is 2.494248535501558e-07
```

The graphs are all straight lines however the constants being multiplied (as elucidated in the previous answer) are different for each plot and depend upon the number of statements in each recursive or iterative cycle. Also each statement takes different time to get executed which leads to the difference in the slopes.

(c)

(i) The time taken to execute *each call* of cleverFib is more than RFib or IFib. However, this difference is very small since the time taken to perform a single addition operation varies from the time taken to perform a single multiplication operation only slightly.
As the value of n increases, the *number of calls* increases at a faster rate for Fib and much more so for RFib hence they taken more time to compute the result.

(ii) It doesn't affect the relative speed of the algorithm as the speed of the algorithm depends both upon the number of calls as well as the time it takes to execute each call (the contribution of the latter becoming increasingly irrelevant as n increases). The very small number of calls in the cleverFib algorithm make it extremely fast.

Task 3

The RAM model helped us evaluate the approximate running time based upon the number of instructions that the computer must execute to arrive at the result. It was **quite accurate** in computing the running time of an algorithm which we can infer from the fact that the practical data that was obtained and then plotted on the graph followed the shape that was expected from the word RAM model. Also it was **very accurate** in comparing the execution time of two algorithms as the data obtained in the table was consistent with the behaviour expected based upon the RAM model.

Prepared By-
Ravija Chandel
Siddhant Jakhotiya