INTRODUCTION TO ALGORITHMS

EC351-V SEM

ASSIGNMENT-1

By – Jennifer yennam (18BEC018)

Problem statement 1:

Consider the following Fibonacci series and solve the following conditions:

fib(n)=fib(0),fib(1),fib(2)……fin(n)

where, fib(n)=fib(n-1)+fib(n-2)

1. Draw the flow chart, algorithms in pseudo code for solving.

2. Write two types of algorithm (recursive and non recursive) for fib(5) and fib(500) series.

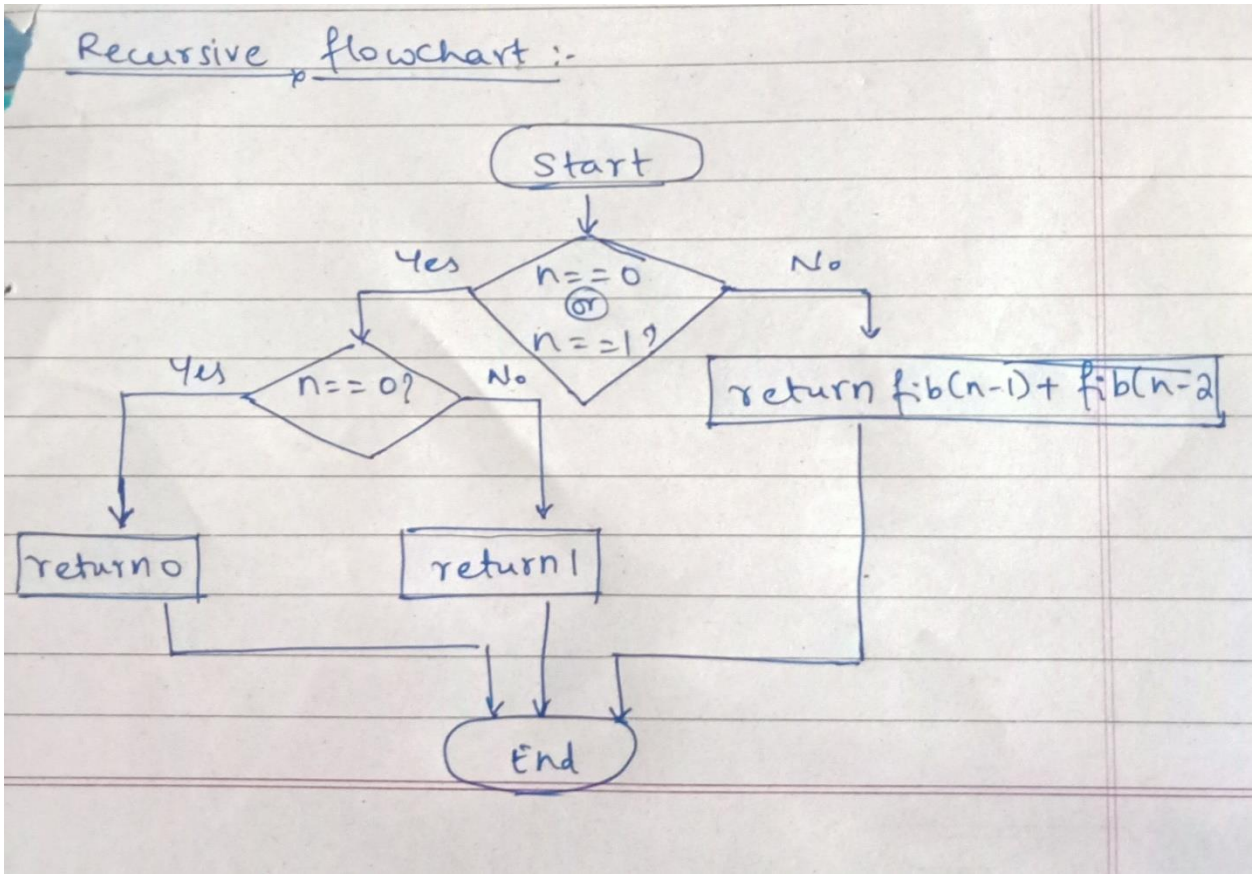
3. Find out the total memory or space required to perform these Fibonacci series computational operations.

4. Find out the WORST CASE and BEST CASE scenario from the above identified approaches.

5. Write a program and compare the actual memory consumed by all the approaches.

(I) Fibonacci Series (recursive):

Flowchart :



Algorithm(Pseudo code):

Procedure fib(n):

If n=0 then return 0

If n=1 then return 1

Else

Return fib(n-1)+fib(n-2)

End procedure

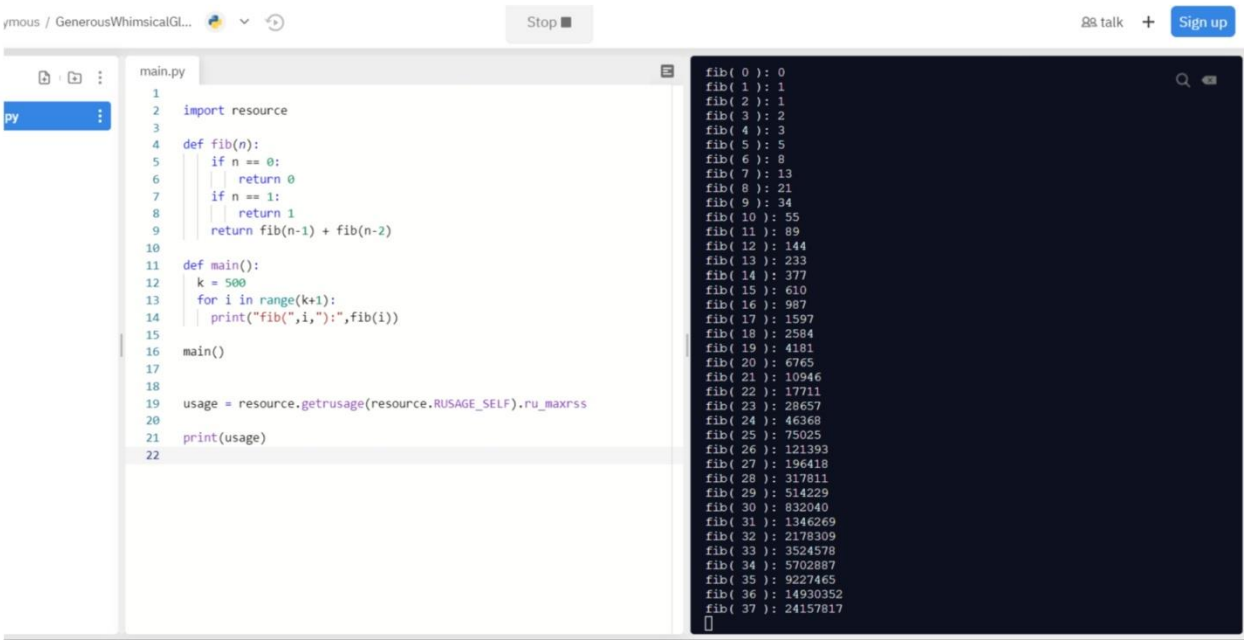
(Output will be in its nth Fibonacci number)

Here, the time complexity is exponential.

The execution time for,

fib(5)=0.00066304 seconds

fib(500)=



Observations:

The recursion algorithm gives adequate results, but only for generating Fibonacci series till 38-41. After 41 it takes a lot of time (nearly some hours) for the algorithm to execute and give out output, as the algorithm has exponential time complexity.

Therefore,

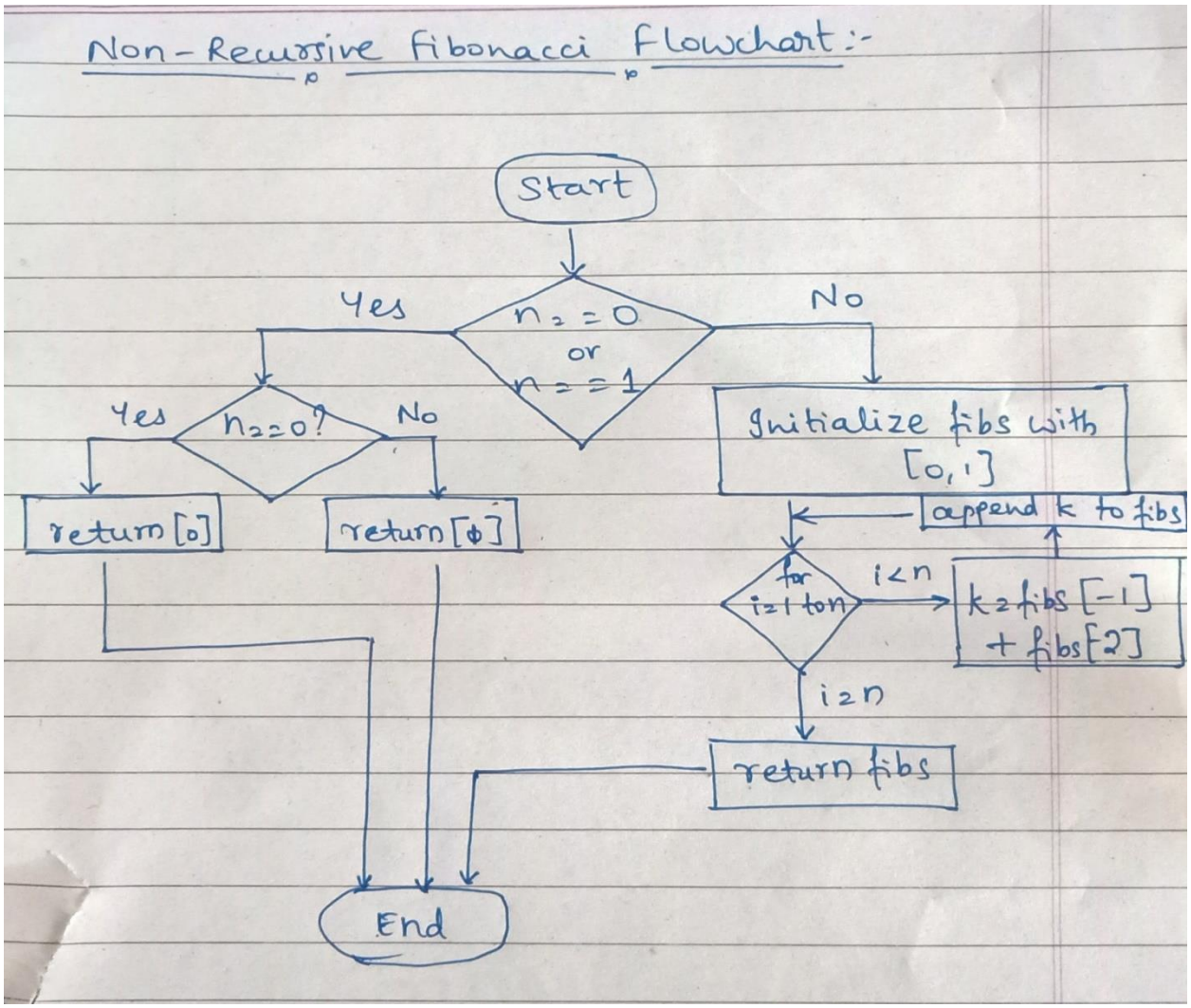
Best case – when n is less in fin(n) {n<42}

Worst case – when n is relatively large. {n>100}

Memory usage – 17668 bytes (calculated using ‘resource’ package of python)

(II) Fibonacci Series (non recursive):

Flowchart:



Algorithm(pseudo code):

procedure fib(n):

if n=0 then return(0)

if n=1 then return (1)

initialize list fibs with (0,1)

For i=1to n

k=fibs[-1] + fibs[-2]

append k to fibs

return fibs

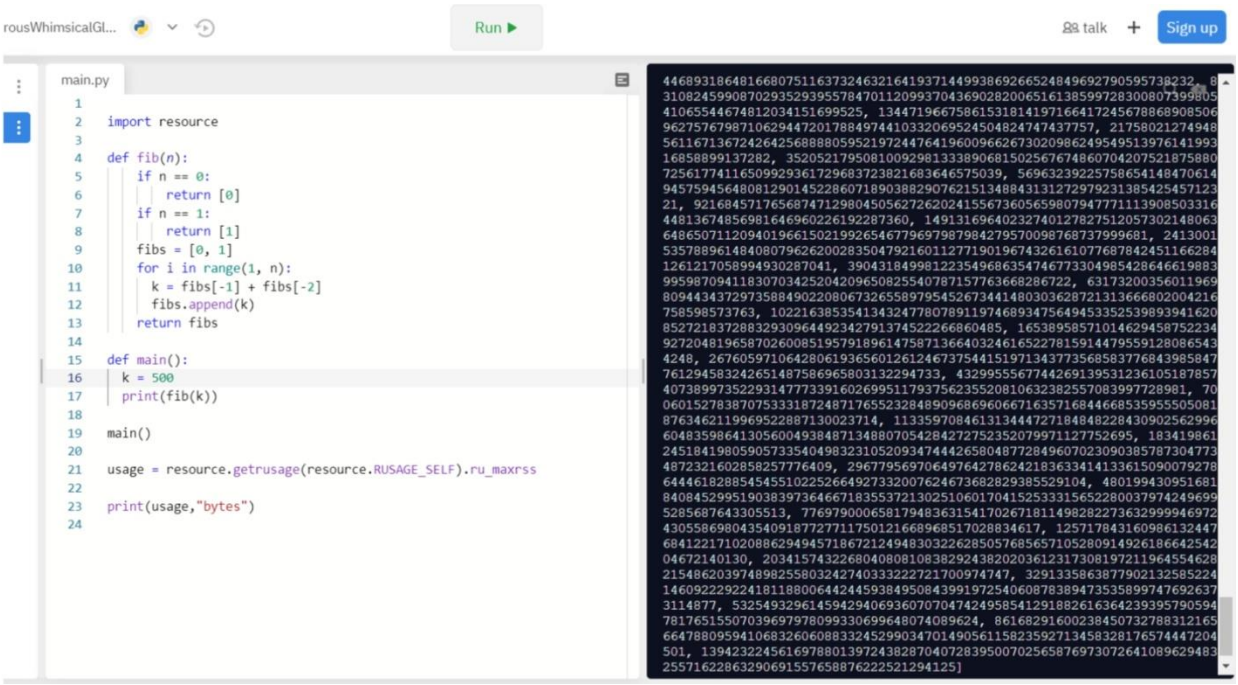
end procedure

(output is Fibonacci series upto n)

Execution time :

Fib(5) – 5.3644e^-05 secs

Fib(500) – 0.0023839 secs



Best case – it is a good option for computing Fibonacci series for large n.

Memory usage – 20068 bytes (calculated using ‘resource’ package of python)

Observation:

1. For generating Fibonacci series upto a small value of n, any of the other algorithms can be used.
2. But, for generating Fibonacci series for a larger value of n that is for n=100,300,500,1000 etc the recursive algorithm takes exponential time and the non-recursive algorithm takes minimal time. So, the second algorithm would be better for large n.

Example :

Fibonacci(10000):

1. Takes 127.09 seconds and consumes only 38152 bytes of memory using 2nd approach.
2. Can take hours (days) altogether using the recursive approach.