Introduction to Algorithms

EC301

Assignment-1

By- Harshil Goolla (18BEC014)

Yashwanth K (18BEC020)

Bharath (18BEC023)

Srinivas Nayaka (18BEC046)

Pramod Betha (18BEC007)

Aditi M (18BEC002)

1. Fibonacci Series (Recursive) :

Algorithm/Pseudocode :

- procedure fibonacci(n):
- if n = 0 then return 0
- if n = 1 then return 1
- else
- return fibonacci(n-1)+fibonacci(n-2)
- end procedure

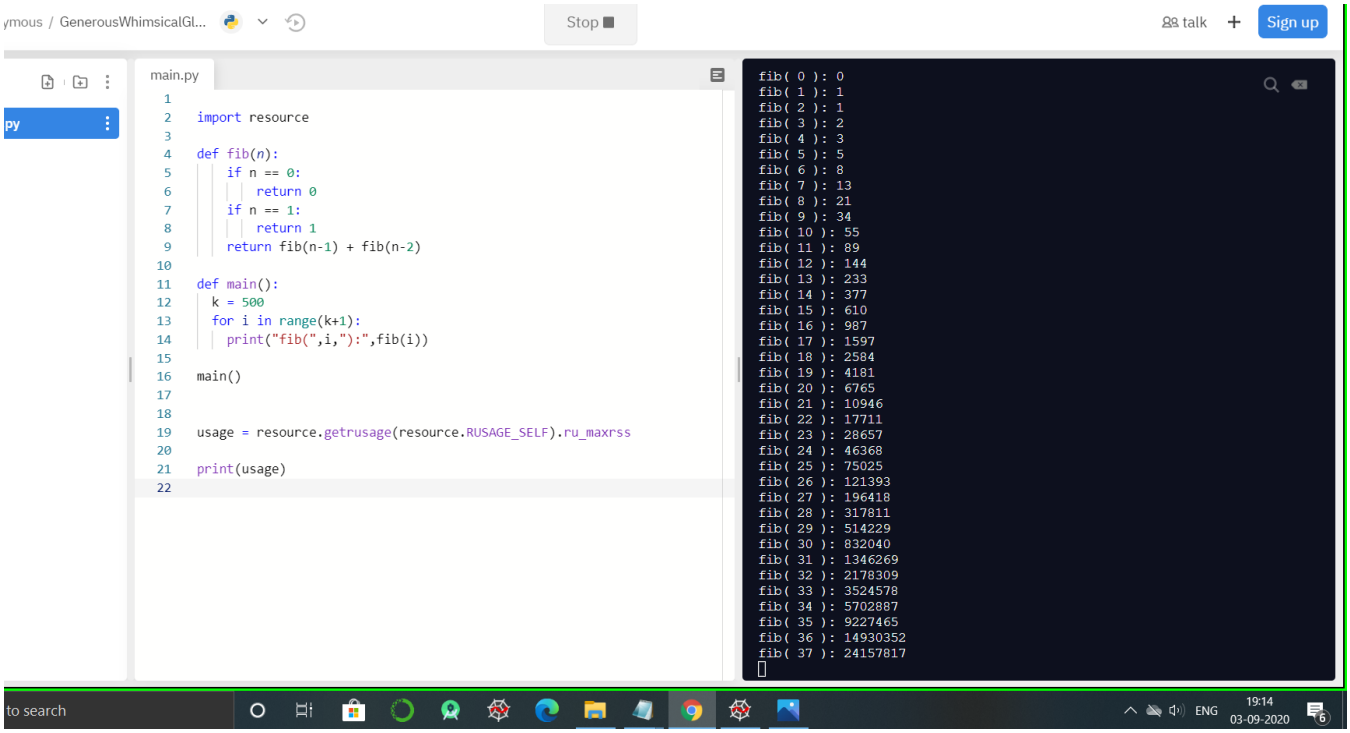
(Output is nth fibonacci number)

Time Complexity: Exponential

Execution Time :

Fibonacci(5) : 0.00066304 seconds

Fibonacci(500) :



Observation:

The recursion algorithm gives satisfactory results only for generating Fibonacci series till 38-41. After that it takes a lot of time, nearly some hours as the algo has exponential time complexity.

So for this algorithm, The Best Case is : when n is low in fib(n) {n<42}

The Worst Case is : when n is relatively large. { n > 100 }

Memory Usage : 17668 Bytes (Calculated using 'resource' package of python)

2. Fibonacci Series (non-recursive):

Algorithm/Pseudocode :

- procedure fibonacci(n)
- if n = 0 then return [0]
- if n = 1 then return [1]
- initialize list fibs with [0,1]
- for i = 1 to n

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

    append k to fibs

- return fibs
- end procedure

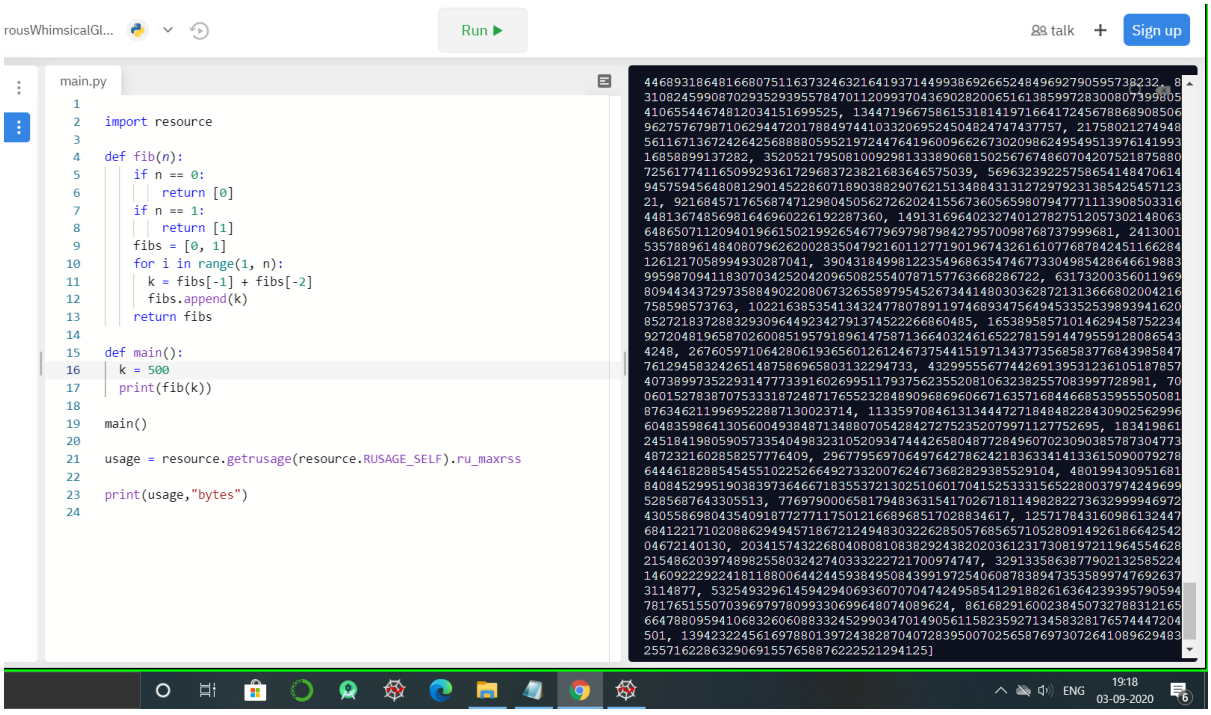
(Output is fibonacci series upto n)

Execution Time :

Fibonacci(5) : 5.3644e^-05 secs

Fibonacci(500) : 0.0023839 secs

Best Case : Good choice for computing Fibonacci series for large n.



Memory Usage :

20068 Bytes (Calculated using 'resource' package of python)

Observation :

- For generating Fibonacci Series upto a small value of n , any of the above algorithms can be used.
- But, for generating Fibonacci Series for a larger value of n .i.e, 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) :

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