**INTORDUCTION TO ALGORITHMS – EC351**

**ASSIGNMENT – 1**

**Consider the following Fibonacci series and solve the**

**following conditions**

**fib (n) = fib(0), fib (1), fib (2),...........fib(n)**

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

1. **Draw the Flow chart , Algorithms in pseudo code for solving**

**Sol :** Step 1: Start

Step 2: Declare variables a,b,c,n,i

Step 3: Initialize variable a=0, b=1, i=2

Step 4: Read n from user Step

5: Print a and b Step

6: Repeat until i<=n c=a+b print c a=b, b=c i=i+1

Step 7: Stop .

For Recursive :

Step 1: Start

Step 2: Define fib(n)

Step 3: **Conditioning If(n==0)**

**return 1**

**elif(n==1)**

**else**

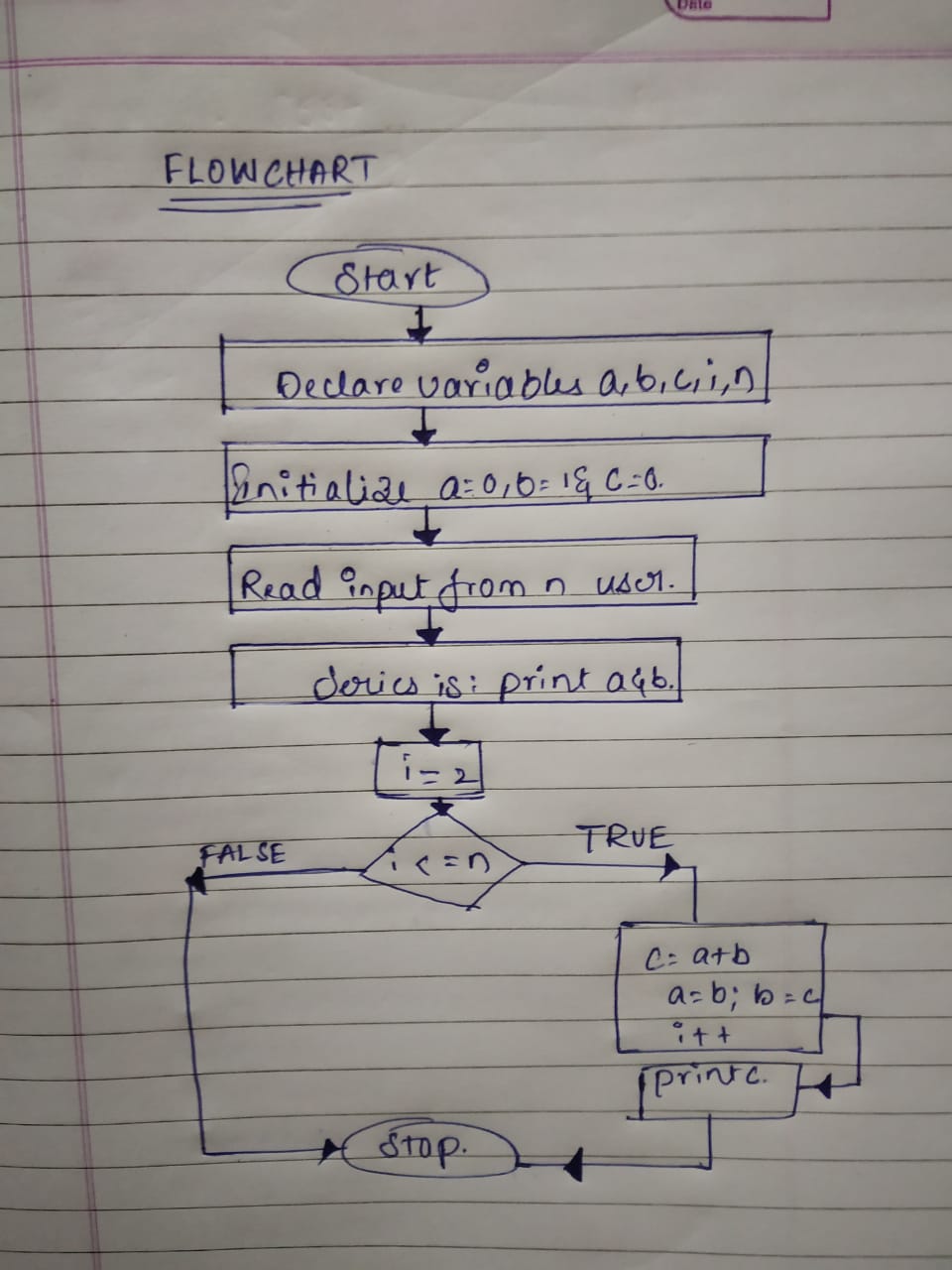
**Step 4: Return fib(n-1)+fib(n-2)**

Step 4: Read n from user

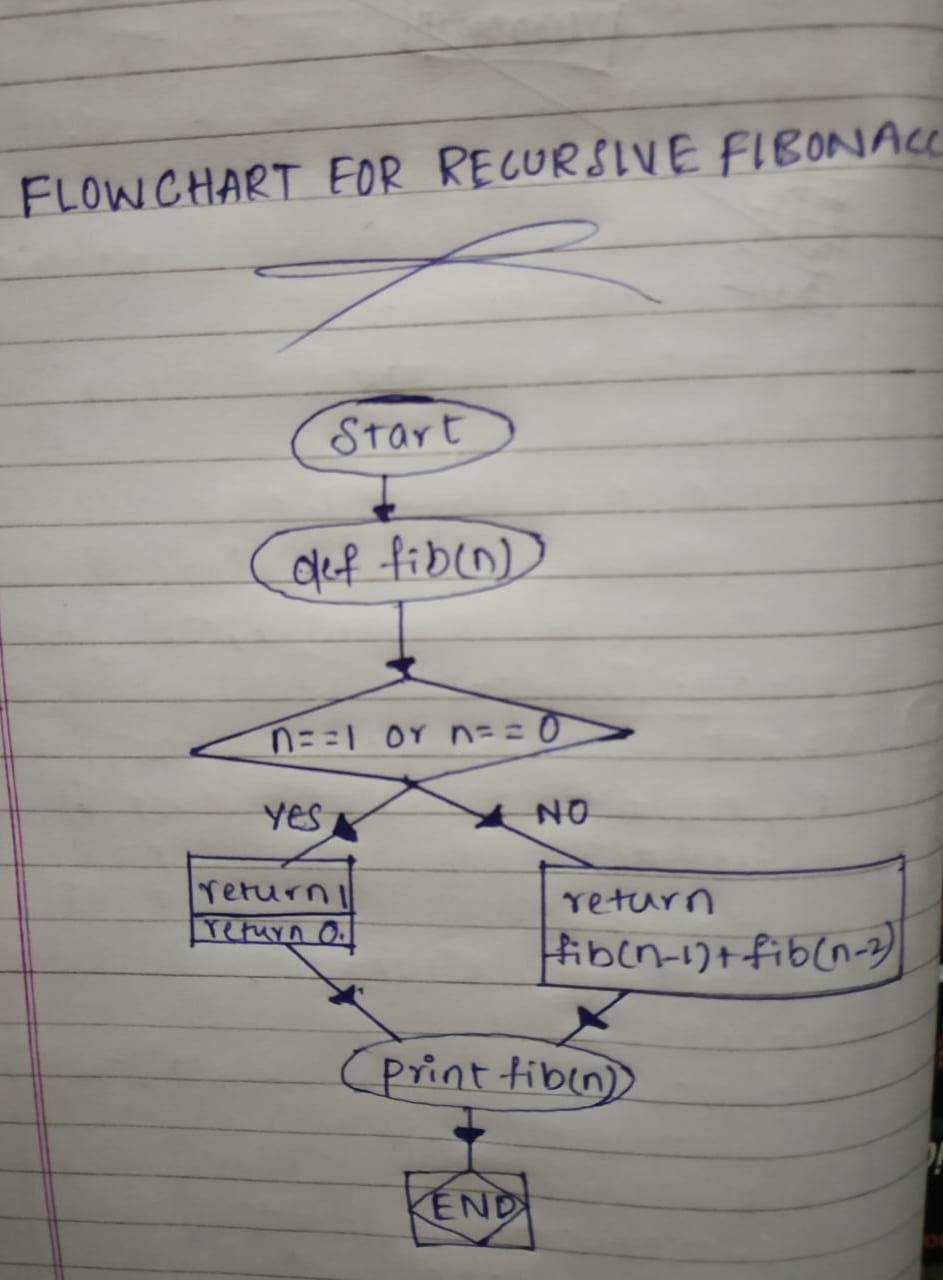
Step 5: Print fib(n)

Step 6: Stop .

Flow chart :



Recursive :



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

**and fib(500) series**

**Iterative Algorithm for fib(5)**

Step 1: Start

Step 2: Declare variables a,b,c,n,i

Step 3: Initialize variable a=0, b=1, i=2 and n=5

Step 4: Print a and b

Step 5: Repeat until i<=n c=a+b print c a=b, b=c i=i+1

Step 6: Stop .

**Iterative Algorithm for fib(500)**

Step 1: Start

Step 2: Declare variables a,b,c,n,i

Step 3: Initialize variable a=0, b=1, i=2 and n=500

Step 4: Print a and b

Step 5: Repeat until i<=n c=a+b print c a=b, b=c i=i+1

Step 6: Stop .

**Recurssion Algorithm :**

Step 1: Start

Step 2: Read n from user

Step 3: Conditioning

If(n==0)

return 1

elif(n==1)

else

Step 4: Return fib(n-1)+fib(n-2)

Step 5: end

**3. Find out the Total memory or space required to perform these**

**Fibonacci series computational operations**

**Sol.** **For Iterative method :**

Space Required / Total Memory = 4 Bytes \* 5 variables

= 20 Bytes

Therefore , Space complexity is O(1) / O(Constant Space).

**For Recursion method :**

Space Required / Total Memory = 4 Bytes \* 5 variables + O(n)

= 20 Bytes + O(n)

= O(n) .

[ There will be n recursive calls , so there will be n stacks used. Hence O(n) Space ]

Therefore , Space complexity is O(n) .

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

**identified approaches**

**Sol :**

Recursive Fibonacci Algorithm holds the worst case scenario , has it occupies O(n) space , the total memory consumption depends on the n .

Iterative Fibonacci Algorithm holds the best case scenario , has it occupies O(1) space / O(Constant space) , the total memory consumption doesn’t depend on the n .

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

**Sol :**

**Iterative code :**

import os

import psutil

def fib1(n):

a = 0

b = 1

if n < 0:

print("Incorrect input")

elif n == 0:

return a

elif n == 1:

return b

else:

for i in range(2,n):

c = a + b

a = b

b = c

i = i + 1

return b

k = int(input("enter Fibonacci sequence index number: "))

print(fib1(k))

process = psutil.Process(os.getpid())

print(process.memory\_info().rss)

**Recurssion code :**

import os

import psutil

def fib(n):

If n <0:

print("incorrect input")

elif n == 0:

return 0

elif n == 1:

return 1

else:

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

k = int(input("enter Fibonacci sequence index number: "))

print(fib(k))

process = psutil.Process(os.getpid())

print(process.memory\_info().rss)

**Observation and Conclusion :**

On running the above codes(both iterative and recursive) we observed that

i.e;

For small ‘n’ values memory consumption in both cases are nearly equal.

Only for large values of ‘n’ we can observe the exact space complexity difference between them.

So consider n=5

**Non recursive case :**

The memory consumption is 121438208 bytes.

**Recursive case :**

The memory consumption is 12152422 bytes.

by

N.Nikhil Kumar

18BEC031.