Group A: Design and Analysis of Algorithms

1. Write a program non-recursive and recursive program to calculate Fibonacci numbers and analyze their time and space complexity.

Iterative Program

```
# Program to display the Fibonacci sequence up to n-th term


nterms = int(input("Enter number of terms "))
# first two terms
n1, n2 = 0, 1

count = 0

# check if the number of terms is valid
if nterms <= 0:
   print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
   print("Fibonacci sequence upto", nterms,":")
   print(n1)
# generate fibonacci sequence
else:
   print("Fibonacci sequence:")
   while count < nterms:
       print(n1)

       nth = n1 + n2

       # update values
       n1 = n2
       n2 = nth
       count += 1
```

Output

```
Enter number of terms 4 Fibonacci sequence:
```

0

1

1

2

Recursive Algorithm

```
Algorithm  rFibonacci(n)
{
        if (n <= 1)
                return n;
        else
                return rFibonacci(n - 1) + rFibonacci(n - 2); }
```

Analysis

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

$= 2T(n-1) + c$ //from the approximation $T(n-1) \sim T(n-2)$

$= 2*(2T(n-2) + c) + c$

$= 4T(n-2) + 3c$

$= 8T(n-3) + 7c$

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

Let's find the value of k for which: $n - k = 0$
$k = n$
$T(n) = 2^n * T(0) + (2^n - 1)*c$

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

Recursive Program

```
def fibonacci(n):
    if(n <= 1):
        return n
    else:
        return(fibonacci(n-1) + fibonacci(n-2))
n = int(input("Enter number of terms:"))
print("Fibonacci sequence:")
for i in range(n):
    print(fibonacci(i))
```

Output

```
Enter number of terms:4 Fibonacci sequence:

0

1

1

2
```