Assignment 1

Rajdhani
DATE / /

Title:- Recursive and Iterative algorithm.

Aim:-

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

Requirements:-

- 64 bit OS
- i5, 8 GB RAM
- VS Code

Objectives:-

- Learn to implement procedure and to pass parameters.
- Understand how to use recursive to define concepts.
- Learn to implement recursion function and handle a stack using low level instruction.
- Analyze algorithm in terms of time and space complexity.

Theory:-

Recursive Function:

A recursive function is one which calls itself. The typical example presented when recursion is first encounted is the factorial

Rajdhani
DATE / /

function.
The factorial 'n' is defined as-

$$n! = 1 \times 2 \times 3 \times \cdots \times (n-2) \times (n-1) \times n$$

A Recursive factorial function :

Any iterative function can be implemented recursively and vice-versa. The recursive function will always be slower due to the added by function. But it is often times easier to state the solution of a problem recursively.

$$n! = \begin{cases} 1 & \text{if } n=0 \\ n \times (n-1)! & \text{if } n>0 \end{cases}$$

Recursive -

```
int recursivefactorial (int n)
{
  int result;
  if (n==0)
      result = 1;
  else
      result = n * recursivefactorial (n-1);
  return result;
}
```

Rajdhani
DATE / /

An iterative function -

```
int iterativefactorial (int n)
{
  int product = 1;
  for (int i = 2; i <= n ; i ++)
     product = product * i;
  return product;
}
```

Fibonacci No. :-
The fibonacci series are the number in the following integer sequence 0, 1, 2, 3, 5, 8, 13, -----

Algorithm :-
1. START
2. Read 'n' value for computing n^{th} term in fibonacci series.
3. Call fibonacci (n).
4. Point the n^{th} term.
5. End fibonacci (n)

$$Fib_n = \begin{cases} 1 & \text{if } (n=0) \\ 1 & \text{if } (n=1) \\ Fib_{n-1} + Fib_{n-2} & \text{if } (n>1) \end{cases}$$

Step 1 : if n=0 then go to step 2 else go to step 3.
Step 2 : return 0.

Rajdhani
DATE / /

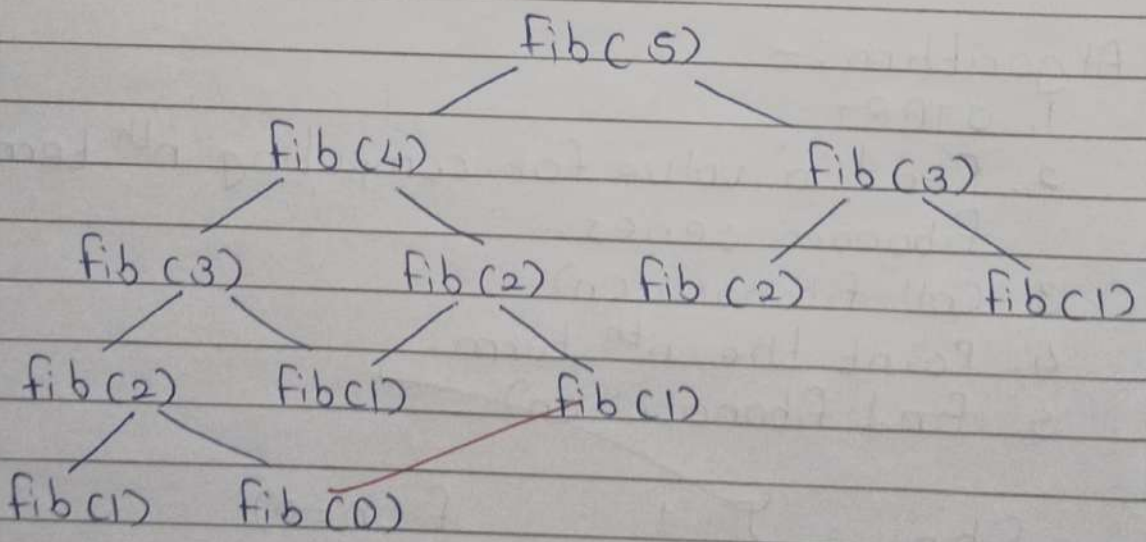
Step 3: If n=1 then go to step 4, else go to step 5.

Step 4: return 1.

Step 5: return (fib (n-1) + fib (n-2))

Time Complexity:-
Exponential as every function calls two other function.

Original tree for recursion:-



Space complexity - O(n), if we consider the function call stack
O(1), otherwise

Rajdhani
DATE / /

Iterative function:

Step 1: If n=0, then go to step 2 else go to step 3.
Step 2: return 0.
Step 3: if n=1, then go to step 4 else go to step 5.
Step 4: return 1.
Step 5: for (i=3; i<=n; i++)
{
c=a+b, a=b, b=c
return c;
}

Time complexity - O(n)
Space complexity - O(1)

Input ⇒ n=8
Output ⇒ fibonacci no. of 8 is 21.

Conclusion:-
Recursive and iterative program to calculate fibonacci number is implemented and analyzed successfully.