**Assignment 2:** Recursive Function and Efficiency Analysis - Write a recursive function pseudocode and calculate the nth Fibonacci number and use Big O notation to analyze its efficiency. Compare this with an iterative approach and discuss the pros and cons in terms of space and time complexity.

**ALGORITHM:**

1.start and define a function “fib” which takes integer n as in input.

2.check if n is less than or equal to 1.If the condition true return n.

3.if n is greater than 1,recursively calls the fib function with (n-1) and(n-2)

4.sum the results of two recursive calls to get the nth Fibonacci number.

**PSEUDOCODE:**

function fib(n):

if n<=1

return n

else

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

This function calculated the nth Fibonacci number by recursively summing (n-1)th term and (n-2)th terms.

Fibonacci function without using recursive function:

int fib\_itr(int n) {

int a = 0, b = 1, next;

if (n <= 1)

return n;

for (int i = 2; i <= n; i++) {

next = a + b;

a = b;

b = next;

}

return b;

}

The time complexity of the recursive Fibonacci function is O(2n).For each call to function it makes two recursive call resulting in exponential growth in number of function calls with respect to n.

Since the time complexity increases exponentially as n increases it takes more time to calculate the nth fibnoacci numbers grows rapidly makes it inefficentfor large values of n.

The time complexity of the iterative approach is O(n) and the space complexity is also O(n).

Recursive Approach:

Pros:

\*Easy to understand and implement.

Cons:

\*Exponential time complexity makes it inefficient for large values of n.

\*Recursion depth can lead to stack overflow errors for large values of n.

Iterative Approach:

Pros:

\*Linear time complexity makes it more efficient than the recursive approach for large values of n.

\*No risk of stack overflow error due to excessive recursion.

Cons:

\*More complex than the recursive approach.-