**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.**

**Pseudocode of recursive function of Fibonacci number**

Function Fibonacci(n):

If n<=1

return n;

else :

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

**Efficiency using Big O notation:-**

The time complexity of this recursive approach is O(2^n). this is because for each recursive call, there are two more recursive call taken until it reached (n<=1). So, the number of function grow exponentially with n.

**Iterative approach:**

Int fibonacci (int n) {

Int a = 0, b = 1, temp;

If (n<=1)

return n;

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

temp = a +b;

a = b;

b= temp;

}

return b;

}

The time complexity of this iterative approach is O(n), it only needs to iterate through the sequence once to compute the nth Fibonacci number.

**Space Complexity:**

The recursive approach has higher overhead due t the recursive calls, which consume memory on the stack . it requires memory for each function until reach the base case.

Iterative approach has a constant space complexity of O(1) . it only uses a constant amount of memory for storing few variable of the input size.

-------------------------------------------------------------------------------------------------------------------------------

Iterative approach is more efficient in both term of time and space complexity while recursive approach is less efficient in time complexity due to exponential growth in function.