Assignment - 09

Recursive Function and Efficiency Analysis

Write a recursive function pseudocode 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.

Recursive Function Pseudocode for Fibonacci Numbers:

```
Function Fibonacci(n):

 If n <= 1:
 Return n
 Else:
 Return Fibonacci(n-1) + Fibonacci(n-2)
```

Efficiency Analysis of Recursive Approach:

The recursive approach to calculating the nth Fibonacci number is simple and elegant but not very efficient. The main issue with this approach is that it performs a lot of redundant calculations. For example, to calculate Fibonacci(5), the function will calculate Fibonacci(3) and Fibonacci(4), but to calculate Fibonacci(4), it needs Fibonacci(3) again. This redundancy grows exponentially with n. Time Complexity: The time complexity of the recursive Fibonacci function is O(2^n). This is because each call to Fibonacci(n) results in two more calls, leading to an exponential growth in the number of calls.

Space Complexity: The space complexity is O(n) due to the maximum depth of the recursion tree, which stores the execution context for each recursive call on the call stack.

Iterative Function Pseudocode for Fibonacci Numbers:

```
Function FibonacciIterative(n):
 If n <= 1:
 Return n
 fib0 = 0
 fib1 = 1
 For i From 2 To n:
 fibN = fib0 + fib1
 fib0 = fib1
 fib1 = fibN
 Return fib1
```

Efficiency Analysis of Iterative Approach: -
The iterative approach to calculating the nth Fibonacci number is more efficient in terms of both time and space complexity compared to the recursive approach.
Time Complexity: The time complexity of the iterative approach is O(n) because it calculates each Fibonacci number from 0 to n exactly once.

Space Complexity: The space complexity is O(1) as it only requires a constant amount of space to store the previous two Fibonacci numbers and the loop index.
Comparison of Recursive Approach and Iterative Approach:

-Recursive Approach Pros: The recursive approach is more straightforward to understand. It closely follows the mathematical definition of Fibonacci numbers.
Recursive Approach Cons: Its exponential time complexity makes it impractical for large values of n.

The space complexity due to recursive calls can also lead to stack overflow errors for large n. Iterative Approach Pros: The iterative approach is much more efficient, with linear time complexity and constant space complexity. It's more suitable for practical use, especially for large values of n.

Iterative Approach Cons: While still simple, the iterative approach is slightly more complex to understand than the recursive version. It doesn't demonstrate the recursive nature of the Fibonacci sequence as clearly.