PROBLEM 0

2. Prove the time complexity of the algorithms

A. The time complexity of the given recursive algorithm for implementing the Fibonacci sequence is exponential, which is O(2^n).

- The function takes two recursive calls to itself with n-1 and n-2 until n reaches 0 or 1. This will be done when n>1 and makes two recursive calls to fib(n-1) and fib(n-2).
- Each of these recursive call's releases two further recursive calls, which leads to binary tree structure.
- Two child nodes will be generated by each non-leaf node in the tree and the total number of nodes in the tree grows exponentially with the depth.
- Therefore, time complexity of the algorithm grows exponentially with the input size of 'n' that makes inefficient for large values of 'n'.

Therefore, the time complexity of the algorithm is exponential, specifically O(2^n), where n is the input value for which we're computing the Fibonacci number.

3. Comment on way's you could improve your implementation

A. The ways that can improve the implementation are

i. Memoization: By utilizing memoization, the algorithm stores previously computed results to avoid redundant calculations, reducing the number of recursive calls and improving overall time complexity to O(n).
ii. Dynamic Programming: Employing dynamic programming involves solving subproblems iteratively, starting from base cases and progressing to the desired value of n. This approach is efficient and eliminates the recursion overhead, leading to linear time complexity O(n).
iii. Iterative Approach: An alternative to recursion is an iterative approach, which computes Fibonacci numbers using loops instead. This method is often faster and eliminates the overhead associated with recursive function calls, resulting in linear time complexity O(n).