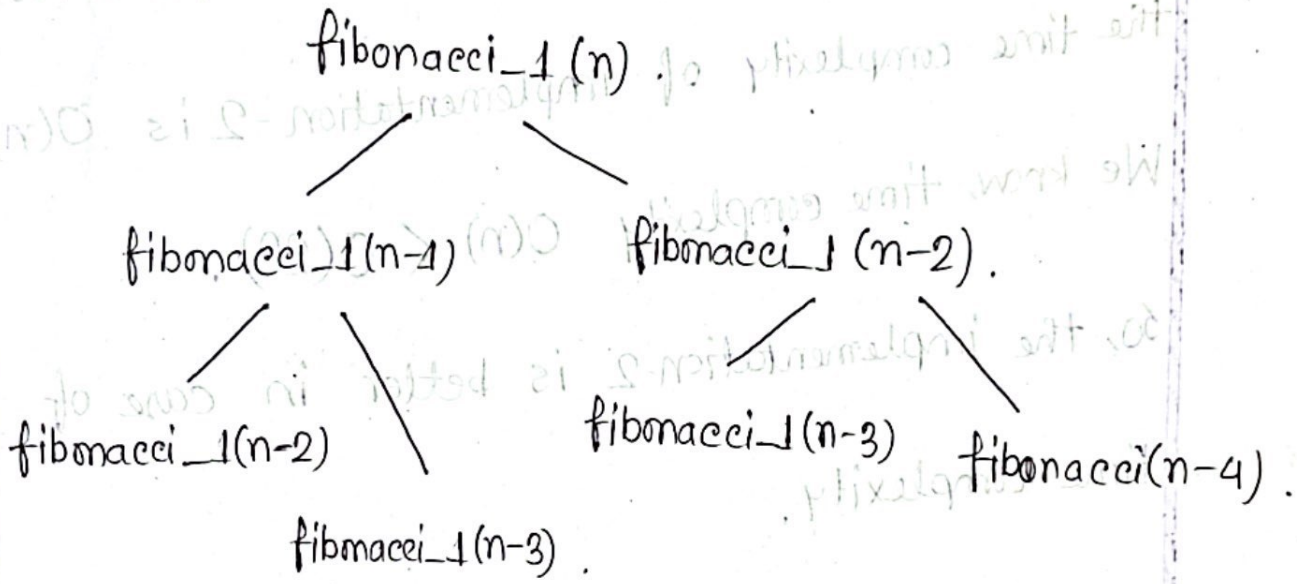
Task 2: Time Complexity Analysis

We have given two implementation code of for finding n-th Fibonacci number.

The condition for fibonacci is: fibonacci_1(n-1) + fibonacci_1(n-2)

The way the tree representation works is that:



It is visible that, we start from a given number and goes until we reach at the base case. In this way, many functions get recalled more than once in implementation 1, which results in taking a huge amount of space. As a result, the time complexity for implementation 1 is $O(2^n)$.

On the ~~order~~ other hand, in implementation 2, we store the outputs in fib list, and when we are about to store another ~~valu~~ result, we check whether we already have this value stored or not. As a result, the time complexity of implementation-2 is $O(n)$.

We know, time complexity $O(n) < O(2^n)$.

So, the implementation-2 is better in case of time complexity.