Task-2(a)

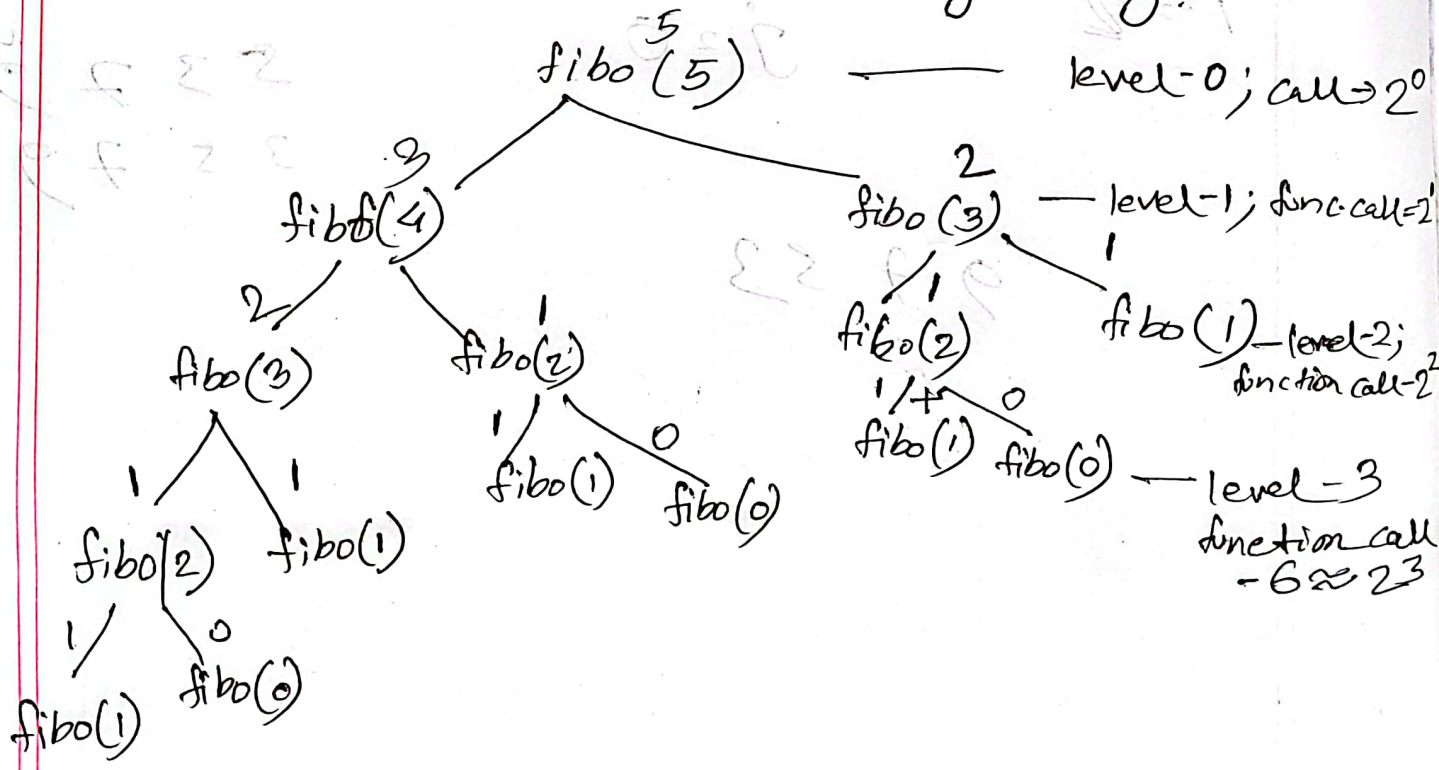
Id-2130499

Implementation-01

the time complexity for the given code is $O(2^n)$.

Explanation:

Let's take : $n = 5$. Now, the recursive function will run in the following way:-



In this case the time complexity would depend on number of function calls.
While calculating the time complexity we formulate the idea roughly from tracing the code.

Total no. of function call $= 2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^{n-1}$ — (i)

[We Put n-1 because the series starts from 0]

We can see (i) follows the is a geometrical Progression.
a = first term and r = common ratio and
Sum of Geometrical Progression, $S_n = \frac{a(r^n - 1)}{r-1}$ [r > 1]

$= \frac{2^0(2^n - 1)}{2-1}$

$= 2^n - 1$

$\simeq 2^n$

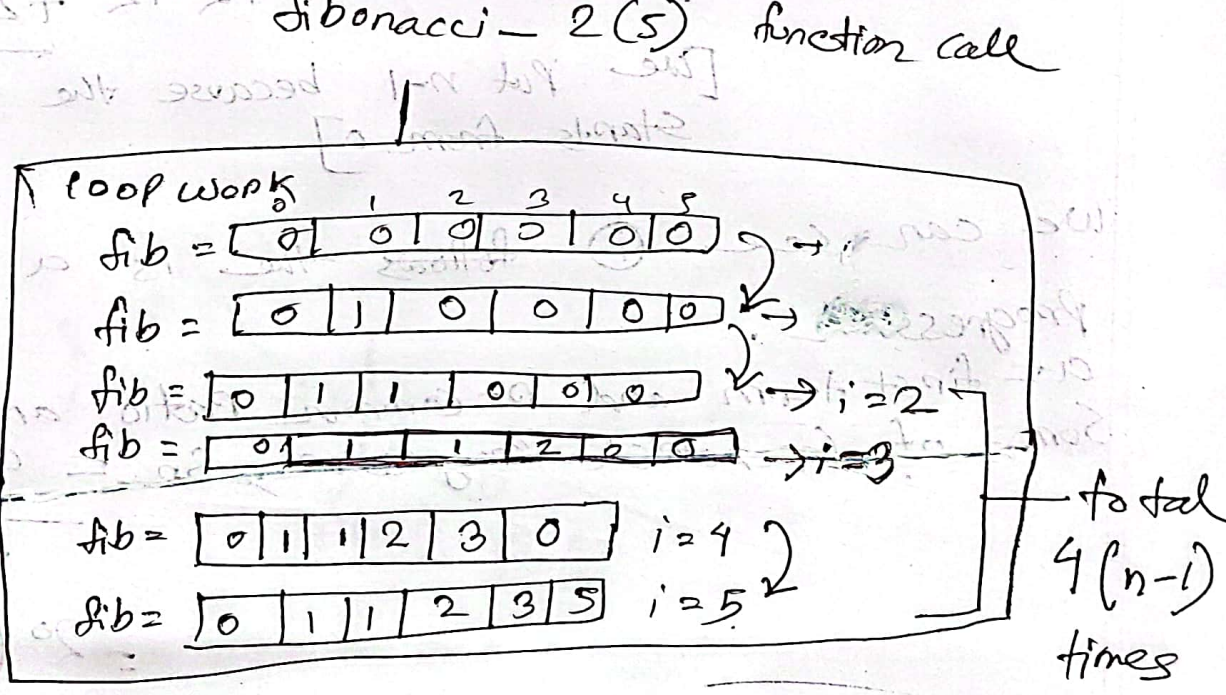
∴ time complexity is $O(2^n)$.

Implementation - 02

The time complexity for the given code is $O(n)$.

Explanation

Let's assume $n = 5$



In this case the function is called only once and the 'for loop' runs more or less $(n-1)$ times.

∴ time complexity will be $O(n)$.

Comparison

Implementation-02 is faster than implementation-01. In implementation-02, memoization is used, so number of function calls is reduced drastically.