Subject: Refatul Islam 20101482 Date:

Answer-02(i)

time com

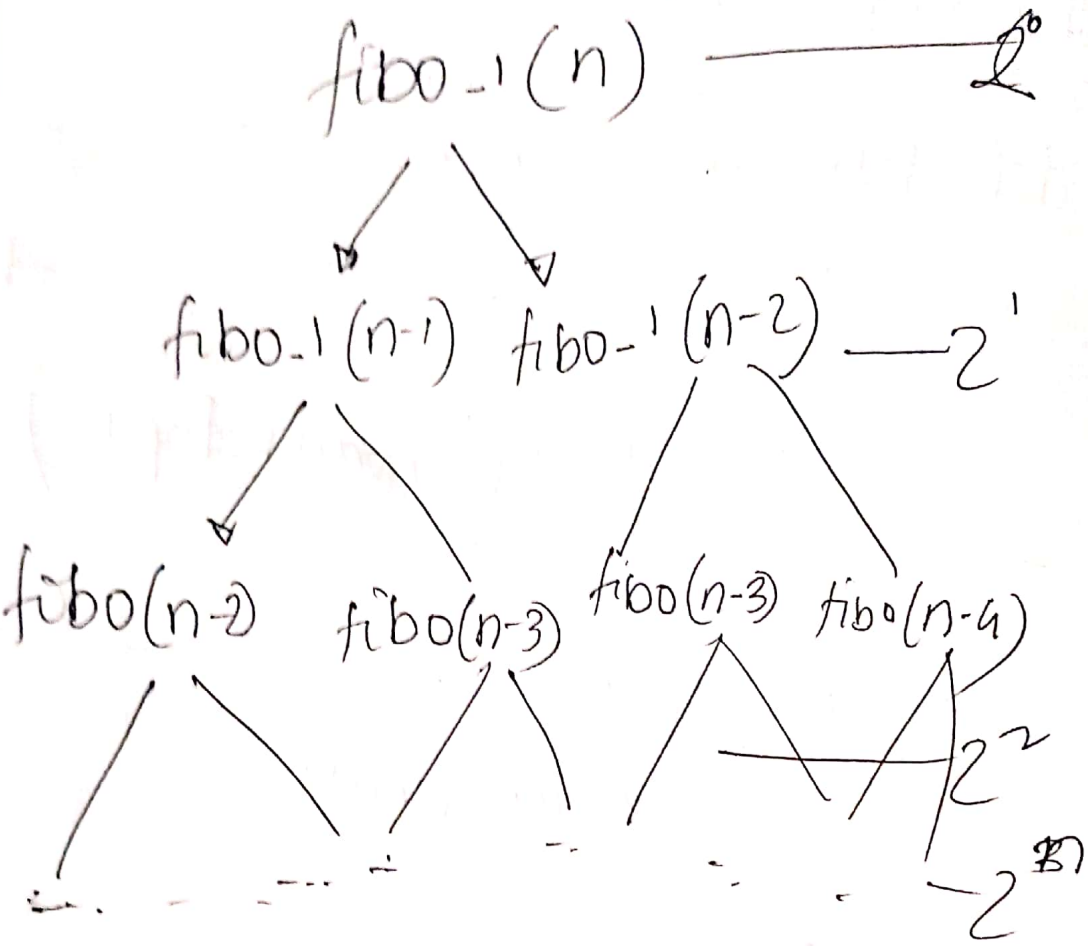
```
def fibonacci_1(n):
    if n=0:                          O(1)
        print("Invalid Input")

    else if n=2:                     O(1)
        return (n-1)

    else.
        return fibonacci_1(n-1) +
                    fibonacci(n-2).
```

tree diagram.

Subject : ..... Date : .....



Combining the all the steps from above, we get

$= O(1) + O(1) + 2^0 + 2^1 + 2^2 + \cdots 2^n$

$= O(1) + O(1) + O(2^{n+1} - 1)$

Subject : ..... Date : .....

$= O(2^{n+1} - 1)$ [ignoring the constants and lower values.]

$= O(2^n)$

therefore, the time complexity

$\longrightarrow O(2^n)$

Ans..

Subject : ..... Date : .....

2nd, Algorithm.

```
def fibonacci-2(n):
    array = [0, 1]                          ——> O(1)
    if (n<0) then,                          ——> O(1)
        print ("invalid input")
    else if (n<=2)then,                     ——> O(1)
        array[n-1]
    return fibonacci.

    else:
        for i in rang(2, n):                ——> O(n)
            array.appends(array[i-1] +
                          array[i-2])
```

Computing all the steps, we get.

$O(n) + O(1) + O(1) + O(1)$

$= O(n)$. [ignoring all constants and lower values]

$\therefore O(n)$

Time complexity = $O(n)$

Ans

Subject 20101422 G M. Refatufate Islam.

(4) Analyzing my solution:

[Ignoring all con O(1)]

During Initializing Matrix/Initial variable, we used a list-comprehent, so. Here the time complexity is O(n).

Then, in List3D function, the ~~tim~~ time complexity is O(n).

Then, in the multiplication for loops, there the complexity is ~~O(n) O(n) * O(n)~~ O(n*n*n) = O(n3)

then, lastly, For printing the matrix the time complexity is $O(n^2)$

total time complexity:

$$O(n) + O(n) + O(n^3) + O(n^2)$$

Since, $O(n^3)$ is the largest among other ~~time~~ $O(n)$, therfore, $O(n^3)$ is the time complexity.

Ans $O(n^3)$