

Name: Md. Si-fat Kamal

ID: 20101231

Section: 04

Ans. to the question no: 2

Implementation - 1

```
def fibonacci_1(n):
```

```
    if n <= 0:
        print("Invalid input!") } o(1)
```

```
    elif n <= 2:
        return n-1 } o(1)
```

```
    else:
        return fibonacci_1(n-1) + fibonacci_1(n-2)
```

```
n = int(input("Enter the number:"))
```

```
nth_fib = fibonacci_1(n)
```

```
print("The %d-th fibonacci number is %d" % (n, nth_fib))
```

Here in this code the time complexity of if and elif statement will

$O(1)$  or Big of 1. Here ~~we can~~ provide another statement is else statement.

In the else statement there is a

recursion that means the function will call several times by himself

in the else statement. Here for the

time complexity we have to

consider the worst case in the

worst situation.

Let's consider the time complexity of the whole code will be  $T(n)$ .

So the equation will be :

$$T(n) = O(1) + O(1) + T(n-1) + T(n-2)$$

$$= 1 + 1 + T(n-1) + T(n-2)$$

$$= 2 + T(n-1) + T(n-2)$$

$$= 2 + T(n-1) + T(n-1-1)$$

$$= 2 + T(n-1) + T(n-1) - T(1)$$

~~$$= 2 + 1 + T(n-1) + T(n-1)$$~~

$$= 2 + T(n-1) + T(n-1) - 1$$

$$\Rightarrow T(n) = 1 + 2T(n-1)$$

$$\Rightarrow 2T(n) = 2 + 2^2 T(n-1)$$

So, the equation becomes.

$$T(n) = 1 + 2 + 2^2 + 3^2 + \dots + 2^{n-1} + 2^{n-1}$$

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

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

$$\frac{(1/2) 2^{n+1} - (1/2)}{1 - (1/2)}$$

$$1 - (1/2) T + (1/2) = 1/2$$

$$= 2^{n+1} - 1$$

$$= 2^n \cdot 2 = (1) \text{ (constant)} \cdot 2^n$$

$$\Rightarrow 2^n \text{ (constant)}$$

∴ The time complexity of the code

will be  $2^n$  : (c = 2) bits

[1-a] power increment - answer

[6-i] power increment +

[1-i] power increment - answer



## Implementation-2

```
def fibonacci_2(n):
```

```
    fibonacci_array = [0, 1]
```

```
    if n < 0:
```

```
        print("Invalid input!")
```

```
    elif (n <= 2):
```

```
        return fibonacci_array[n-1]
```

```
    else:
```

```
        for i in range(2, n):
```

```
            fibonacci_array.append(fibonacci_array[i]
```

```
            + fibonacci_array[i-2])
```

```
        return fibonacci_array[-1]
```

```
n = int(input("Enter a number"))
```

```
nth_fib = fibonacci_2(n)
```

```
print("The %d-th fibonacci number is %d" % (n, nth_fib))
```

In this code there are three statements.  
Among them for the code we have  
to consider worst case i.e. worst  
case scen. So for this if condition  
will won't be true. So for this  
the time complexity will be  $O(1)$  on  
Big O of  $\frac{1}{2}$ . Similarly it will be  
also applicable for the elif statement.  
So the time complexity will be  
 $O(1)$ . Now, since both became  
false so the code will enter to



The else statement. Here in the else statement there is a for loop as well. and the if loop will be run from 2 to  $(n-1)$  times. So, the time complexity will be  $O(n-1)$  times. Total time complexity for the above statement will be  $O(n-1) * O(1)$ .

Now, the total equation will be:

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

~~$O(1) + O(1)$~~  Here we won't count the  $O(1)$  since  $O(1) < O(n)$   
 $\therefore O(n)$