MATADOR PAGE NO.: DATE: / /

2) For finding the n-th fibonacci number in the implementation I we can see for two sub problem there we will have:

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

As $T(n-2) \simeq T(n-1)$ so we can write as if, in,

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

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

$$= 2T(n-2\times1) + c$$

By breaking

$$= 2\{2T(n-4) + c\} + c$$

$$= 4T(n-4) + 2c + c$$

$$= 2^2 T(n-(2\times2)) + 3c$$

$$= 2^2 T(n-(2\times2)) + (2^2-1)c$$

$$= 4\{2T(n-6) + c\} + 3c$$

$$= 8T(n-6) + 4c + 3c$$

$$= 2^3 T(n-6) + 7c$$

$= 2^3 T(n - \overset{(2\times 3)}{6}) + (2^3 - 1)c$

$= \cancel{4\{2T(n-6) + c\}}$

$= 8\{2T(n-8) + c\} + 7c$

$= 16T(n-8) + 8c + 7c$

$= 2^4 T(n-8) + (2^4 - 1)c$

$= 2^4 T(n - (2\times 4)) + (2^4 - 1)c$

So from ~~now~~ this we can come to a general form which is,

$= \cancel{2^{\frac{n-2}{2}} T(n}$

$= 2^{\frac{n-2}{2}} T\{n - (2 \times \frac{n-2}{2})\} + (2^{\frac{n-2}{2}} + 1)c$

Here, $T\{n - (2 \times \frac{n-2}{2})\} \simeq T(2)$

So, $\simeq 2^{\frac{n-2}{2}} T(2) + c$

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

$\simeq 2^{\frac{n}{2} - \frac{2}{2}} \simeq 2^{\frac{n}{2} - 1} \simeq 2^{\frac{n}{2}}$

~~So here~~

∴ upper bound = $O(2^n)$

Now for implementation 2

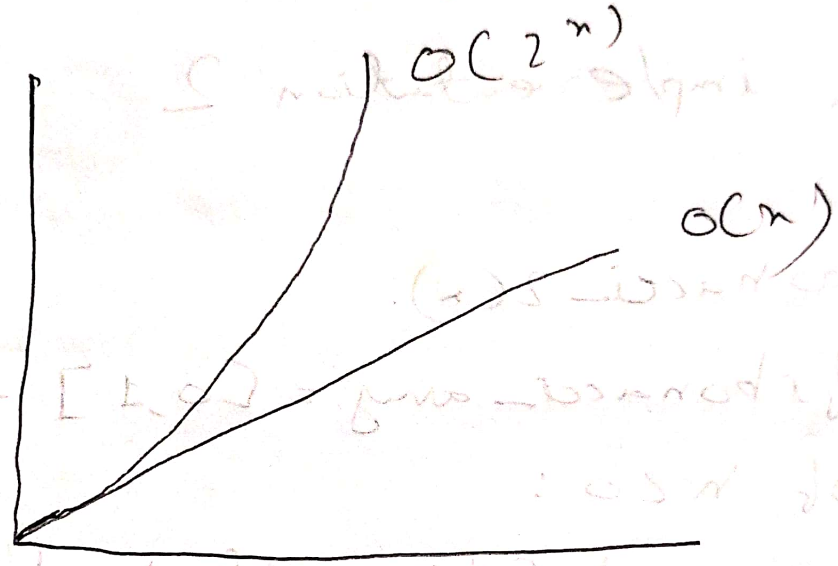
```
def fibonacci_2(n):
    fibonacci_array = [0,1]          → O(1)
    if n<0:                          → O(1)
        print ("Invalid Input")
    elif n<=2:                       → O(1)
        return fibonacci_array [n-1]

    else:                            → O(n)
        for i in range(2,n):         → O(n)
            fibonacci_array.append(fibonacci_
            array[i-1]+fibonacci_array[i-2])
        return fibonacci_array [-1]
```

So for implemention -2 it is $O(n)$

By ~~plo~~ plotting the $O(2^n)$ and

$O(n)$ we can see,



So, by comparing from the graph we

can say $O(n)$ is better.

4) So,

for i = 0 to n-1 → for this loop $\boxed{O(n)}$

for j = 0 to n-1 → for this loop $\boxed{O(n)}$

for k = 0 to n-1 → for this loop $\boxed{O(n)}$

C[i, j] + = A[i, k] * B[k, j]

end for

end for

end for

As they are nested loop,
So the time complexity will be $O(n*n*n)$

$\Rightarrow O(n^3)$

5) (c.1) Given,

$$T(n) = T\left(\frac{n}{2}\right) + n - 1, \quad T(1) = 0$$

$$\simeq T\left(\frac{n}{2}\right) + n$$

From the master's theoram ~~we know~~ we know,

$$T(n) = aT\left(\frac{n}{b}\right) + cn^k$$

By comparing it with $T\left(\frac{n}{2}\right) + n$ we get,

$$a = 1, \; b = 2, \; k = 1$$

As, $T(n) = O(n^k)$ if $b^k > a$

here, $b^k > a$

$\cancel{b}\ 2^1 > 1$

$\therefore$ Time complexity $= O(n^k)$

$$= O(n^1)$$

$$= O(n)$$

2) Given,

$$\cancel{3}T(n) = T(n-1) + n - 1, \quad T(1) = 0$$

$$= \cancel{\{T(n-2)} + n$$

$$= \{T(n-1-1) + n-1-1\} + n - 1$$

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

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

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

$$= T(n-n) + n + n + n \ldots\ldots - (1+2+3+\ldots n)$$

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

$$\simeq n^2$$

$\therefore$ Time complexity $\Rightarrow O(n^2)$

3) Given,

$$T(n) = T\left(\frac{n}{3}\right) + 2T\left(\frac{n}{3}\right) + n$$

$$= 3T\left(\frac{n}{3}\right) + n$$

~~As~~

From the Master's theorem,

here, $a = 3, b = 3, k = 1$

As, $T(n) = \Theta(n^k \log_b n)$ if $b^k = a$

Here, $b^k = a$

$\Rightarrow 3^1 = 3$

$\therefore$ Time complexity $= \Theta(n^k \log_b n)$

$$= O(n^1 \log_3 n)$$

$$= O(n \log_3 n)$$

4) Given,

$$T(n) = 2T\left(\frac{n}{2}\right) + \cancel{b3}\, n^2$$

In the Master's theorem,

$$T(n) = aT\left(\frac{n}{b}\right) + cn^k$$

By comparing we get,

$$a = 2,\ b = 2,\ k = 2$$

As,

$$T\left(\frac{n}{b}\right) = O(n^k) \text{ if } b^k > a$$

And here,

$$b^k > 1$$

$$\cancel{b}\, 2^2 > 1$$

$\therefore$ Time complexity $= O(n^k)$

$= O(n^2)$

So, worst case time complexity n^2

[Proved].