Answer to the Question no: 2

Implemention 1:

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

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

$$= 2^1 T(n-2\times1) + c$$

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

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

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

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

$$= 8T(n-6) + 7c$$

$$= 2^3 T(n-2\times3) + (2^3-1)c$$

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

$$= 16T(n-8) + 15c$$

$$= 2^4 T(n-2\times4) + (2^4-1)c$$

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

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

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

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

$$\simeq 2^{n/2}$$

Therefore, the upper bound is $O(2^n)$

Implementation 02 :

fibonacci_array = [0, 1] $\rightarrow$ $O(1)$

If Block $\longrightarrow$ $O(1)$

ElseIF Block $\longrightarrow$ $O(1)$

Else Block $\longrightarrow$ For loop

For loop $\longrightarrow$ $O(n)$

therefore time complexity for Implementation 2 is $O(n)$ which is better than Implementation 1 as it takes less time.

Answer to the Question no: 4

Pseudocode:-

```
for i = 0 to n-1                 .... (i)
                                  .... (ii)
    for j = 0 to n-1
                                  ..... (iii)
        for k = 0 to n-1
            C[i,j] += A[i,k] * B[k,j]
```

section (i) runs $(n-1)$ or n times with a complexity of $O(n)$.

section (ii) runs under the (i) section and also runs for $n-1$ time or n times resulting a time complexity of $O(n^2)$.

Section (iii) runs under section (ii) and also runs $n-1$ time resulting the time complexity of $O(n^3)$

There for the time complexity for these three nested loop will be $O(n^3)$

Ar.

Answer to the Question no: 05

(i) Given, $T(n) = T(n/2) + n - 1$

Or, $T(n) = T(n/2) + n$

According to master theorem,

$T(n) = aT(n/b) + O(n^d)$

In case, $a = b^d$ then $O(n^d \log n)$

Or, $a < b^d$ then $O(n^d)$

Or, $a > b^d$ then $O(n^{\log_b(a)})$

In this problem,

$a < b^d$

So, the time complexity will be, $O(n^d)$

$= O(n^1)$

$= O(n)$

(ii) Given, $T(n) = T(n-1) + n-1$ where $T(1) = 0$

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

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

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

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

Here the highest order of n will be n^2.
as a result, the time complexity will be
$O(n^2)$

An.

(iii) Given, $T(n) = T(n/3) + 2T(n/3) + n$

or, $T(n) = 3T(n/3) + n$

Applying master theorem,

In this case $a = b^d$ as $a = 3$, $b = 3$, $d = 1$

Therefore the time complexity will
be $O(n^d \log n)$
$= O(n \log n)$

An

(iv) Given,

$$T(n) = 2T(n/2) + n^2$$

Applying master theorem.

In this case $a < b^d$ where $a = 2$, $b = 2$, $d = 2$

Therefore the time complexity will be, $O(n^d)$

$= O(n^2)$