VANSH SHARMA SEC-H CLASS RNO→01
UNIVERSITY RNO→2017119

TUTORIAL-2

Ans-1)

```
void fun(int n)
{
    int j=1, i=0;
    while(i<n){
        i= i+j;
        i++;
    }
}
```

$j=1,\ i=0+1$

$j=2,\ i=0+1+2$

$j=3,\ i=0+1+2+3$

loop ends when $i>=n$

$0+1+2+3+\dots+n > n$

$\frac{K(K+1)}{2} > n$

$K^2 > n$

$K > \sqrt{n}$

$O(\sqrt{n})$

Ans-2) Recurrence Relation for Fibonacci Series

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

$$T(0) = T(1) = 1$$

- if $T(n-1) \simeq T(n-2)$

(lower bound)

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

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

$$= 4\cdot(2T(n-6)) = 8T(n-6)$$

$$= 8(2T(n-8)) = 16T(n-8)$$

$$\vdots$$

$$T(n) = 2^k T(n-2k)$$

$n - 2^K = 0$

$n = 2^K$

$T(n) = 2^{n/2} T(0) = 2^{n/2}$

$T(n) = \Omega(2^{n/2})$

- if $T(n-2) \neq T(n-1)$

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

$= 2[2T(n-2)] = 4T(n-2)$

$= 4(2T(n-3)) = 8T(n-3)$

$= 2^k T(n-k)$

$n - k = 0$

$k = n$

$T(n) = 2^k \times T(0) = 2^n$

$T(n) = O(2^n)$ (upper bound)

Ans → 3) • $O(n \log n)$ ⇒

```
for (int i=0; i<n; i++) {
    for(int j=1; j<n; j=j*2) {
        // some O(1)
    }
}
```

- $O(n^3)$ ⇒

```
for (int i=0; i<n; i++) {
    for(int j=0; j<n; j++) {
```

```
for(int k=0; k<n; k++) {
        // some O(1)
    }
}
```

- $O(\log(\log n))$ ⇒

```
for(int i=1; i<=n; i=i*2) {
    for(int j=1; j<=n; j=j*2) {
        // some O(1)
    }
}
```

Ans-4)

$$T(n) = T(n/4) + T(n/2) + cn^2$$

lets assume $T(n/2) >= T(n/4)$

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

applying Master's theorem

$$a = 2,\ b = 2,\ f(n) = n^2$$

$$c = \log_b a = 1$$

$$n^c = n$$

Compare n^c & $f(n)$

$$f(n) > n^c \quad \text{so}$$

$$T(n) = \Theta(n^2)$$ Ans

Ans-5)

```
int fun (int n) {
    for (int i=1; i<=n; i++) {
        for(int j=1; j<n ; j+=i )
        {
            //some O(1)
        }
    }
}
```

$i = 1 \longrightarrow$ $j=1, j=2, j=3, \ldots, j=n$ } n times

$i = 2 \longrightarrow$ $j=1, j=3, j=5, j=7$ } Loop ends when $j > n$

$1+3+5+7 > n$

$k > n/2$

$\rightarrow$ n times

$i = 3 \rightarrow$ $j=1, j=4, j=7$ $\rightarrow$ } $1+4+7 > n$

$k > n/3$

$i = 4 \rightarrow$ $k > n/4$

$\vdots$

$i = n$

$\therefore$ total complexity $= O(n^2) + O(n^2) + \ldots$

$= O(n^2)$

Ans→6)

```
for (int i=2; i<=n; i= Pow(i, K)) {
    // some O(1)
}
```

complexity Pow (i, K) → O(log N)

= log (K)

$i = 2$

$i = 2^K$

$i = 2^{K^2}$

$i = 2^{K^3}$

$i = 2^{K^4}$

⋮

$i = 2^{K^M}$

loop ends when $i > n$

$$2^{K^M} > n$$

$$\log 2^{K^M} > \log n$$

$$K^M > \log n$$

$$\log K^M > \log \log n$$

$$M > \frac{\log \log n}{\log K}$$

$$T.C. = O(\log(\log n))$$

Ans-8) a) $100 < \log n < \sqrt{n} < n < \log(\log n) < n\log n < \log n! < n! < n^2 < \log^{2n} < 2^n < 2^{2n} < 4^n$

b) $1 < \sqrt{\log n} < \log n < 2\log n < \log 2N < N < 2N < 4N < \log(\log N) < N\log N < \log N! < N! < N^2 < 2\times 2^N$

c) $96 < \log_8 N < \log_2 N < n\log_6 N < n\log_2 N < \log n! < N! < 5N < 8N^2 < 7N^3 < 8^{2N}$