SAGAR NEGI SEC-H ROLL NO-10

Tutorial - II

Que 1. What is the time complexity of below code and how?

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

Sol:- $j=1$, $i=0+1$

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

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

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

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

$$K^2 > n$$

$$K > \sqrt{n}$$

$$\therefore T\cdot C = O(\sqrt{n})$$

Que 2. Write recurrence relation for the recursive function that ~~prices~~ prints fibonacci series. Solve the recurrence relation to get time complexity of the program. What will be the space complexity of this program and why?

Sol:- Recurrence relation for fibonacci series:

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

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

if $T(n-1) \approx T(n-2)$ ~~(lower bound)~~

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

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

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

$$= 4[2T(n-6)]$$

$$= 8T(n-6)$$

$\Rightarrow 8[2T(n-8)]$

$\Rightarrow 16T(n-8)$

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

$n-2k=0$

$k=\frac{n}{2}$ $\quad T(n) = 2^{n/2} T(0)$

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

if $T(n-2) \approx T(n-1)$ {upper bound}

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

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

$= 4T(n-2)$

$= 4[2T(n-3)]$

$= 8T(n-3)$

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

$n-k=0$

$\boxed{k=n}$

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

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

Que 3. Write program which have complexity:
n(log n), n^3, log(log n)

Sol: i) n(log n)

```
=> for(int i=0; i<n; i++)
   {
   for (int j=1; j<n; j=j*2)
   {
    s= s+i;
   }
   }
```

$O(n^3)$

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

$O(\log(\log n))$

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

Que 4. Solve the following recurrence relation $T(n) = T(n/4) + T(n/2) + cn^2$

Sol:- $T(n) = T(\frac{n}{4}) + T(\frac{n}{2}) + cn^2$

Let us assume $T(n/2) >= T(n/4)$

So $T(n) = 2T(\frac{n}{2}) + cn^2$

Applying master's theorem,

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

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

$$c = \log_b a = \log_2 2 = 1$$

$$n^c < f(n)$$

$$n < n^2$$

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

Que 5: what is the time complexity of following function fun()?

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

Sol:- $i=1$ — $j=1$, $j=2$, $j=3$, ..., $j=n$ → n times

$i=2$ — $j=1$, $j=3$, $j=5$, $j=7$ → $k > \frac{n}{2}$

$i=3$ — $j=1$, $j=4$, $j=7$ → $k > \frac{n}{3}$

$\therefore T(c) = O(n^2+n^2+n^2)$

$= O(n^2)$

Que 6: what should be the time complexity of

```
for (int i=2; i<=n; i= pow(i,k))
{
//some O(1) task
}
```

where k is a constant

Sol:- complexity of pow (i,k) - $O(\log N)$

- $O(\log(k))$

$i=2$

$i=2^k$

$i=2k^2$

$\vdots$

~~$i=2^{kM}$~~ $i=2^{kM}$

loop codes when i > n

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

$$\log(2^{k^M}) > \log n$$

$$k^M \log 2 > \log n$$

$$k^M > \log n$$

$$\log(k^M) > \log(\log n)$$

$$M \log k > \log(\log n)$$

$$M > \frac{\log(\log n)}{\log(k)}$$

$$TC = O(\log(\log n))$$

Que 8. Arrange the following in increasing order of rate of growth

(a) n, n!, log n, log log n, root(n), log(n!), nlogn, log^2(n), 2^n, 2^(2^n), 4^n, n^2, 100

Sol:- $100 < \log n < \sqrt{n} < n < \log(\log n) < n\log n < \log n! < n! < n^2 < \log 2^n < 2^n < 2^{2n} < 4^n$

(b) 2(2^n), 4n, 2n, 1, log(n), log(log(n)), $\sqrt{\log(n)}$, log 2n, 2 log(n), n, log(n!), n!, n2, n log(n)

Sol:- $1 < \sqrt{\log n} < \log n < 2\log n < \log 2n < n < 2n < 4n < \log(\log n) < n\log n < \log n! < n! < n^2 < 2 \times 2^n$

(c) 8^(2n), $\log_2(n)$, $n\log_6(n)$, $n\log_2(n)$, log(n!), n!, $\log_8(n)$, 96, $8n^2$, $7n^3$, 5n

Sol:- $96 < \log_8 n < \log_2 n < n\log_6 n < n\log_2 n < \log n! < n! < 5n < 8n^2 < 7n^3 < 8^{2n}$