Tutorial - 2

Unzila
Sec - I

Ans 1)

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

$i = 0, 1, 3, 6, 10, 15$ ---- k terms

so general form would be $\frac{k(k+1)}{2}$

k^{th} term $= n$

$\Rightarrow \frac{k(k+1)}{2} = n$

$\Rightarrow k^2 + k = 2n$

$\Rightarrow k^2 = n$

$\Rightarrow k = \sqrt{n}$

Time Complexity $= O(\sqrt{n})$

Ay.

Ans 2) Recursive Function :-

```
int fib (int n)
{   if (n <= 1)
        return n;
    return fib(n-1) + fib(n-2);
}
```

Recurrence Relation :-

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

$T(n) = 2T(n-1) + C$ $\quad \because T(n-1) \approx T(n-2)$

By Backward Substitution :-

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

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

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

$$T(n) = 4T(n-2) + 3C$$

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

$$T(n) = 4(2T(n-3) + C) + 3C$$

$$T(n) = 8T(n-3) + 7C$$

Generalizing,

$$2^k T(n-k) + (2^k - 1)C$$

assume $n - k = 0$

$\Rightarrow n = k$

$$= 2^n T(0) + (2^n - 1)C$$

$$= 2^n + (2^n - 1)C$$

$$= 2^n (1 + C) - C$$

$$= 2^n$$

Time Complexity $= O(2^n)$

Ay.

Space Complexity :-

For fibonacci recursive implementation, the space required is directly proportional to the maximum depth of Recursion tree, Since Maximum depth is directly proportional to the number of elements.

So, Space Complexity $= O(n)$

Ay.

Ans 3 → Programs which have complexity $n \log n$.

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

(ii). $O(n^3)$

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

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

(iii). log(log n)

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

Ans 4)

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

$$\because T(n/4) \approx T(n/2)$$

$$\Rightarrow T(n) = 2T(n/2) + Cn^2$$

Using Master's Theorem :-

$$T(n) = aT(n/b) + f(n)$$

$$c = \log_b a = 1$$

$$f(n) > n^c \Rightarrow Cn^2 > n^c$$

$$T(n) = O(f(n))$$

$$= O(n^2)$$

Ay.

Ans 5)

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

for $i=1$, $1+2+3+ \dots (n+1) = n$

for $i=2$, $1+3+5+ \dots n = n/2$

for $i=3$, $1+4+7+ \dots n = n/3$

$$n + \frac{n}{2} + \frac{n}{3} + \dots + 1$$

$$\Rightarrow n\left(1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}\right)$$

Now, we know that

$$n\left(1+\frac{1}{2}+\frac{1}{3}+\dots+\frac{1}{n}\right) \le n\left(1+\frac{1}{2}+\frac{1}{2}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4}\dots\right)$$

Time Complexity $= O(n\log n)$

Ay.

Ans 6 → `for (int i=2; i<=n; i= pow(i,k))`

```
{
    // O(1)
}
```

$i = 2,\ 2^k,\ 2^{k^2} \dots 2^{k^i}$

$$2^{k^i} = n$$

$$\log n = \log 2^{k^i}$$

$$i = \log(\log n)$$

Time Complexity $= O(\log(\log n))$

Ay.

Ans 8 →

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

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