TUTORIAL - 2

1.

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

i	j
0	1
1	1
1+2 = 3	2
1+2+3 = 6	3
⋮	⋮
1+2+3+ ---k	k

∴ for i

0, 1, 3, 6, ------ (1+2+3+ ---+k)

k terms

$\because T_k = Ak^2 + Bk + C$

for k=1

$T_1 = A + B + C \Rightarrow A+B+C = 0$ —(1) $[\because T_1 = 0]$

for k=2

$T_2 = 4A + 2B + C \Rightarrow 4A + 2B + C = 1$ —(2)

for k=3

$T_3 = 9A + 3B + C \Rightarrow 9A + 3B + C = 3$ —(3)

solving eq. ① , ② & ③ , we get

$A = \frac{1}{2}$, $B = \frac{1}{2}$ & $C = 0$

$\therefore \quad T_k = \frac{k^2}{2} + \frac{k}{2}$

$\because \quad k^2 < n$

$\Rightarrow \quad k < \sqrt{n}$

$T_n = \sqrt{n}$

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

2.

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

Recurrence relation

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

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

$\Rightarrow \quad T(n) = 2T(n-1) + 1 \quad \text{—— ①}$

put $n = n-1$ in eq. ①

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

put the value of $T(n-1)$ in eq. ①

$$T(n) = 4T(n-2) + 2 + 1 \quad \text{—— ②}$$

Put $n = n-2$ in eq. (1)

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

put the value of $T(n-2)$ in eq. (2)

$$T(n) = 8T(n-3) + 4 + 2 + 1 \quad \text{---} \quad (3)$$

$\therefore$ from eq. (1), (2) & (3)

$$T(n) = 2^k T(n-k) + \underbrace{1+2+4+ \cdots +}_{k} 2^{k-1}$$

put $n-k = 0$

$\Rightarrow n = k$

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

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

$$\Rightarrow \boxed{T(n) = O(2^n)}$$

Space complexity = $O(n)$

As for this program the time complexity will depend on the depth of recursive tree, which is n.

3. (i) Program with Time Complexity $n(\log n)$

```
int main ()
{
    int n , count=0;
    cin >>n;
    for (int i= 0;  i <n ; i++)
    {
        for (int j =0 ; j<n ; j* =2)
        {
            count++;
        }
    }
    cout << count <<endl;
}
```

(ii) Program with Time Complexity n^3

```
int main ()
{
    int n , count = 0;
    cin >> n;
    for (int i=0 ; i<n ; i++)
    {
        for (int j=0 ; j<n ; j+=2)
        {
            for (int k=0 ; k<n ; k++)
            {
                count++;
            }
        }
    }
    cout << count << endl;
}
```

(iii) Program with Time Complexity $\log(\log n)$

```
int main ()
{
    int n , p=0;
    cin >> n;
    for (int i=0 ; i<n ; i*=2)
        p++;

    for (int j=1 ; j<p ; j*=2)
        cout << j ;
}
```

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

$T(n/4)$ will be ignored as it is of lower order

$\Rightarrow T(n) = T(n/2) + cn^2$ ——— (1)

put $n = n/2$ in eq. (1)

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

Put the value of $T(\frac{n}{2})$ in eq. (1)

$$T(n) = T\left(\frac{n}{4}\right) + c\frac{n^2}{4} + cn^2 \quad \text{— (2)}$$

Put $n = \frac{n}{4}$ in eq. (1)

$$T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) + \frac{cn^2}{16}$$

Put the value of $T(\frac{n}{4})$ in eq. (2)

$$T(n) = T\left(\frac{n}{8}\right) + c\frac{n^2}{16} + c\frac{n^2}{4} + cn^2 \quad \text{— (3)}$$

from eq. (1), eq. (2) & eq (3)

$$T(n) = T\left(\frac{n}{2^k}\right) + cn^2\underbrace{\left[1 + \frac{1}{4} + \frac{1}{16} + \cdots + \frac{1}{4^{k-1}}\right]}_{k}$$

Put,

$$\frac{n}{2^k} = 1$$

$$\Rightarrow 2^k = n$$

Substituting,

$$\Rightarrow T(1) + cn^2 \frac{\left[1 \times \left(1 - \left(\frac{1}{4}\right)^k\right)\right]}{1 - \frac{1}{4}}$$

$$1 + cn^2\left[\frac{4}{3} - \frac{4}{3} \times \left(\frac{1}{4}\right)^k\right]$$

$$1 + cn^2 \times \frac{4}{3} - cn^2 \times \frac{4}{3} \times \frac{1}{2^{2k}}$$

$$1 + cn^2 \times \frac{4}{3} - cn^2 \times \frac{4}{3} \times \frac{1}{n^2}$$

$$\Rightarrow 1 + cn^2 \times \frac{4}{3} - c \times \frac{4}{3}$$

$$\therefore \boxed{T(n) = O(n^2)}$$

5.

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

i	j
1	1, 2, 3, - - - n times
2	1, 3, 5, - - - n → n/2 times
3	1, 4, 7, - - - - → n/3 times
⋮	⋮
n	1, - - - - - n/n times

$1, 3, 5, \cdots, n$ (k terms)

$\Rightarrow 1 + (k-1)2 = n$

$k = \frac{n+1}{2}$

Total time complexity $\Rightarrow n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n}$

$\Rightarrow n\left[1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right]$

$\Rightarrow n \cdot \sum_{k=1}^{n} \frac{1}{k}$

$\Rightarrow n \cdot \log(n)$

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

6.

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

$i \rightarrow 2^k, 2^{2k}, \ldots\ldots, 2^{k^i}$

for the termination of loop

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

Taking log,

$$k^i \log_2 2 = \log_2 n$$

$$k^i = \log_2 n$$

again taking log

$$i \log_k k = \log_k \log_2 n$$

$$\Rightarrow i = \log_k \log_2 n$$

Time complexity $\Rightarrow \log_k \log_2 n$

8.

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

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

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