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

$\therefore$ for i

$0, 1, 3, 6, - - - (1+2+3+ - - - +K)$
K-terms

$\because T_K = AK^2 + BK + C$

for $K=1$

$T_1 = A+B+C \Rightarrow A+B+C=0$ —— (i)

for $K=2$

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

for $K=3$

$T_3 = 9A+3B+C \Rightarrow 9A+3B+C=3$ —— (iii) solving eq ①, ② & ③ we get

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

$\therefore T_K = \frac{K^2}{2} + \frac{K}{2}$

$\because K^2 < n$

$\Rightarrow K < \sqrt{n}$

$T_n = \sqrt{n}$

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

$\Rightarrow T(n) = 2T(n-1) + 1$ —— (i)

put $n = n-1$ in (i)

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

Put the value of $T(n-1)$ in eq. (i)

$T(n) = 4T(n-2) + 2 + 1$ —— (ii)

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

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

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

$T(n) = 8T(n-3) + 4 + 2 + 1$ —— (iii)

$\therefore$ from eq. i, ii & (iii)

$T(n) = 2^K T(n-K) + 1 + 2 + 4 - - - + 2^{K-1}$

put $n - K = 0$

$\Rightarrow n = K$

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

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

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

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++;
      }
```

Scanned by TapScanner

```
    }
  }
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 lower order

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

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

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

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

$T(n) = T(n/4) + \frac{cn^2}{4} + cn^2$ —(ii)

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

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

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

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

from eq 1, 2 & 3

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

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

Substituting,

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

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

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

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

5.
```
int fun(int n){
  for (int i=1; i<=n; i++) {
     for (int j=1; j<n; j+=i)
     {
        //some O(1) task
     }
  }}
```

i	j
1	1, 2, 3 - - - n times
2	1, 3, 5 - - - n $\rightarrow \frac{n}{2}$ times

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

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

3	1, 4, 7, - - - - $\rightarrow n/3$ times
⋮	
n	1, - - - - - $\frac{n}{n}$ times

Scanned by TapScanner

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

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

$$\Rightarrow n \cdot \sum_{K=1}^{n} \frac{1}{K} \Rightarrow n.\log(n)$$

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

6. for (int i=2; i<=n; i = pow(i,K))

{ // Some O(1) expressions

}

$i \rightarrow 2^K, 2^{2K}, \text{---}, 2^{K^i}$

for the termination of loop

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

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

$$K^i = \log_2 n$$

again taking log

$$i \log_K \cancel{K} = \log_K \log_2 n$$

$$i = \log_K \log_2 n$$

Time complexity $\Rightarrow \log_K \log_2 n$

Scanned by TapScanner

8.

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

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