Design and Analysis of Algorithm

Tutorial-2

1.

Code is.

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

values of i

0
1
3
6
10
15
21
.
.
.
upto n.

$\Rightarrow i = 0, \underbrace{1, 3, 6, 10, 15, 21 \ldots\ldots n}_{K \text{ terms}}$

Let the sum of these K terms be S_K

$\Rightarrow S_K = 1+3+6+10+15+21 \ldots\ldots\ldots + T_K$ —— (1)

$S_{K-1} = 1+3+6+10+15+21 \ldots\ldots T_{K-1}$ —— (2)

Subtracting (2) from (1),

$\Rightarrow T_K = S_K - S_{K-1} = 1+2+3+4+5+6+\ldots\ldots+K$

we have $T_K = n$.

$\Rightarrow 1+2+3+4+5+6+\ldots\ldots+K = n$

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

$\Rightarrow K^2+K-2n=0$

$\Rightarrow K = \frac{-1 \pm \sqrt{8n+1}}{2}$, taking only positive value

get total no. times the loop runs for $i = K+1 = \frac{-1+\sqrt{8n+1}}{2} + 1$

$$= 1 - \frac{1}{2} + \frac{\sqrt{8n+1}}{2}$$

$$= \frac{1+\sqrt{8n+1}}{2}$$

⇒ Time complexity:-

$$T(n) = O\left(\frac{1+\sqrt{8n+1}}{2}\right) = O\sqrt{n}.$$

2.

pseudo Code:-

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

⇒ • Time Complexity:-

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

when $n=0$ & $n=1$

i.e, $T(0) = T(1) = 1$.

Here, $T(n-2) \approx T(n-1)$

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

put $n = n-1$ in eqn ①,

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

put in ①,

$T(n) = 2[2T(n-2)+1] + 1 = 4T(n-2) + 2 + 1$ ——②

put $n = n-2$ in eq^n ①,

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

put in ②,

$T(n) = 4 * [2T(n-3)+1] + 2 + 1$ ——③

$T(n) = 8T(n-3) + 4 + 2 + 1$ ——

⇒ Generalised form:-

$$T(n) = \underbrace{2^k T(n-k) + 2^{k-1} + 2^{k-2} + \dots + 1}_{k+1 \text{ terms}}$$

put $n-K=0$

$$\Rightarrow T(n) = 2^n T(0) + 2^{n-1} + 2^{n-2} + \dots + 1$$

$$= 2^n + 2^{n-1} + 2^{n-2} + \dots + 1$$

$$= \underbrace{1 + 2 + 4 + 8 + 16 + \dots + 2^n}_{k+1 \text{ terms}}$$

$a=1$, $a_{k+1} = 2^n$, $r = 2$

$\Rightarrow$ $(k+1)^{th}$ term:-

$$a_{k+1} = ar^{(k+1-1)}$$

$$2^n = 2^k$$

On comparing,

$$n = K$$

$\Rightarrow$ T.C. $= O(k+1) = O(n+1) = O(n)$.

• Space Complexity:- Here n is the no. of entries in a stack and for each function call one.

So space complexity for each case (call) is 1, i.e., O(1)

so for n no. of cases, $\leq n$

$\Rightarrow$ i.e. $O(n)$.

3.

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

$\Rightarrow$ $O(n(\log n))$.

```
for (int i=0; i<n; i++) {
  for (int j=0; j<n; j++) {
    for (int k=0; k<n; k++){
      // O(1) - Statements
      } } }
```

$\therefore$ $O(n^3)$.

~~for (int k int~~

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

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

4.

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

On removing $T(n/2)$ as smaller term,

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

Applying Master's Theorem,

$a = 0$, $b = 2$, $K = 2$, $P = 0$

$\log_b a = \log_2 0 = 0$.

$0 < 2$, i.e., $\log_b a < K$, & $P \geq 0$.

$\Rightarrow T.C = \theta(n^K \log^P n)$

$\therefore TC = \theta(n^2 \log^0 n)$

$= \theta(n^2)$.

Ans 5

time complexity of the function fun() is $O(n \log n)$.

» for $i = 1$, inner loop executed n times.

for $i = 2$, inner loop executed $n/2$ times

for $i = 3$, inner loop executed $n/3$ times

⋮

for $i = n$, inner loop executed $n/n = 1$ time.

$$\Rightarrow \quad n + \frac{n}{2} + \frac{n}{3} + \cdots\cdots + \frac{n}{n}$$

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

⇒ particular form for time complexity is $(\log n)$.

So for, total times, the loop executed,

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

Ans 6

```
for (int i = 2; i < n; i = pow(i,k)) {
    // O(1) - Expressions.
}
```

i takes the values like, $2, 2^k, k^2, 2^{k^3} \cdots 2^{k \log_k (\log n)}$;

last term must be less than or equal to n.

$$\Rightarrow \quad T(n) = O(\log_k (\log (n))).$$

Ans 8

a)

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

b)

$1 < \sqrt{\log n} < \log n < \log(n!) < \log(\log n) < \log(2n) < 2\log(n) < \log(n!) < n\log(n) < n < 2n < 4n < n! < 2^{6n}$

c)

$9b < \log_9(n) < \log_2(n) < \log(n!) < n! < n\log_6(n) < n\log_2(n) < 5n < \Theta n^2 < \Theta^{2n} < 7n^3.$