

# DESIGN AND ANALYSIS OF ALGORITHMS.

## TUTORIAL-1

(11/03/2022)

Answer 1: Asymptotic notations are used to find the complexity of an algorithm when input is very large.

- Big Oh ( $O$ ):  $f(n) = O(g(n))$   
iff  $f(n) \leq c_1 g(n)$   
for all  $n > k$  and  
for some constant  $c > 0$   
 $g(n)$  is "tight upper bound" of  $f(n)$
- Big Omega ( $\Omega$ ):  $f(n) = \Omega(g(n))$   
iff  $f(n) \geq c_2 g(n)$   
for  $n \geq k$   
for some constant  $c > 0$ .  
 $g(n)$  is "tight lower bound" of  $f(n)$ .
- Big Theta ( $\Theta$ ):  $f(n) = \Theta(g(n))$   
iff  $c_1 g(n) \leq f(n) \leq c_2 g(n)$   
for  $n \geq \max(n_1, n_2)$   
for some constant  $c_1 > 0$  and  $c_2 > 0$   
 $g(n)$  is both "tight upper bound" and "tight lower bound" of  $f(n)$ .

Answer 2 : for  $(i=1 \text{ to } n)$  &  $i = i * 2;$  }

1, 2, 4, 8, ... n

let  $k^{\text{th}}$  term = n

$$n = 1 \cdot (2^{k-1})$$

taking log on both sides.

$$\log n = (k-1) \log 2$$

$$k = \log n + 1$$

$$O(1 + \log n)$$

$$\underline{O(\log n)} = \text{Ans.}$$

Answer 3:  $T(n) = 3T(n-1) \rightarrow (1)$

putting  $n = n-1$  in eq<sup>n</sup> (1)

$$T(n-1) = 3T(n-2) \rightarrow (2)$$

put (2) in (1)

$$T(n) = 9T(n-2)$$

putting  $n = n-2$  in eq<sup>n</sup> (1)

$$T(n-2) = 3T(n-3) \rightarrow (3)$$

$$T(n) = 27T(n-3)$$

$$T(n) = 3^k T(n-k)$$

$$n-k=0$$

$$n=k$$

$$T(n) = 3^n T(n-n)$$

$$= 3^n T(0)$$

$$= 3^n$$

$$\underline{O(3^n)} = \text{Ans.}$$



Answer 4:

$$T(n) = 2T(n-1) \rightarrow (i)$$

$$n = n-1 \text{ in eqn (i)}$$

$$T(n-1) = 2T(n-2) \rightarrow (ii)$$

$$T(n) = 4T(n-2) \rightarrow (iii)$$

putting  $n = n-2$  in eqn (i)

$$T(n-2) = 2T(n-3) \rightarrow (iv)$$

$$T(n) = 8T(n-3)$$

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

$$n-k = 0$$

$$k = n$$

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

$$= 2^n T(0)$$

$$= 2^n$$

$$O(2^n) = \text{Ans.}$$

Answer 6:

void function (int n)

{

int i; count = 0;

for (i = 1; i \* i ≤ n; i++)

count++;

}

complexity:  $O(1 + \sqrt{n} + \sqrt{n} + \sqrt{n})$

$$O(1 + 3\sqrt{n})$$

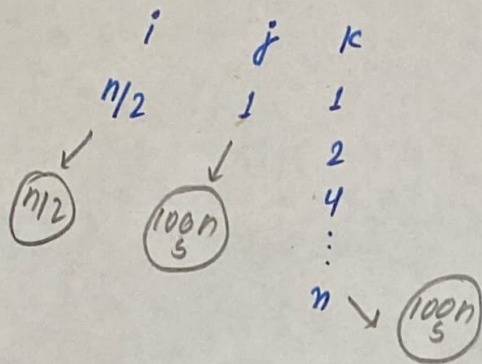
$$O(3\sqrt{n})$$

$$O(\sqrt{n})$$

$$O(n^{1/2}) = \text{Ans.}$$

Answer 7:

```
void function (int n)
{
    int i, j, k, count = 0;
    for (i = n/2; i <= n; i++)
        for (j = 1; j <= n; j = j * 2)
            for (k = 1; k <= n; k = k * 2)
                count++;
}
```

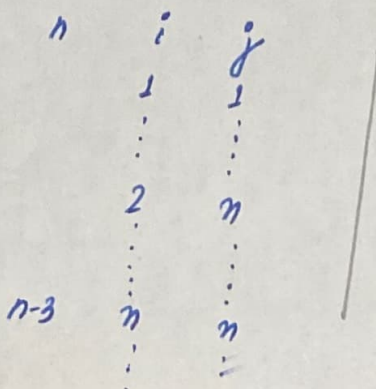


$$O\left(\frac{n}{2} \times \frac{100n}{5} \times \frac{100n}{5}\right)$$

$$O\left(n(100n)^2\right) = \text{Ans.}$$

Answer 8:

```
function (int n)
{
    if (n == 1)
        return;
    for (i = 1 to n)
    {
        for (j = 1 to n)
        {
            printf ("x");
        }
        function (n-3);
    }
}
```





$n-6$

$\vdots$

complexity:

$$1 + 4 + 7 + \dots + n$$

$$n = 1 + 3(k+1)$$

$$= 3k + 2$$

no. of terms

$$k = \frac{n+2}{3}$$

$$= \frac{n+2}{6} \left[ 2 + \left(\frac{n-1}{3}\right) \times 3 \right]$$

$$= \left[ \frac{n+2}{6} (n+1) \right] \times n^2$$

$$= O \left[ \frac{(n^2 + 3n + 2)}{6} \times n^2 \right]$$

$$= O(n^4) \text{ ans}$$

Answer 9:

void function (int n)

{

for (i=1 to n)

{

for (j=1; j<=n; j+=j+i)

    printf ("x");

}

$$O(n + n^2 + n^2 + n^2)$$

$$O(3n^2 + n)$$

$$O(n^2) \text{ ans}$$

answers:

```
int i=1, s=1;  
while (s<=n);  
{  
    i++; s=s+i;  
    printf("%d\n", i);  
}
```

$$i = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad \dots$$

$$s = 1 + 3 + 6 + 10 + 15 + \dots + n.$$

$$\text{sum of } s = 1 + 3 + 6 + 10 + \dots + n \rightarrow (1)$$

$$\text{also } s = 1 + 3 + 6 + 10 + \dots + n-1 + n \rightarrow (2)$$

from (1) - (2)

$$0 = 1 + 2 + 3 + 4 + \dots + n - n$$

$$T_k = 1 + 2 + 3 + 4 + \dots + k$$

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

for  $k$ , inter

$$1 + 2 + 3 + \dots + k \leq n$$

$$k(k+1)/2 \leq n$$

$$(k^2 + k)/2 \leq n$$

$$O(k^2) \leq n$$

$$k = O(\sqrt{n})$$

$$T(n) = O(\sqrt{n})$$

$$\underline{O(n^{1/2})} \text{ Ans.}$$