Tutorial - 1

1. What do you understand by Asymptotic notations. Define different Asymptotic notation with examples.

→ Asymptotic notation is used to describe the running time of an algorithm and how much time an algorithm takes with a given input and when Input is very large.

Types of notations -

1. Big oh notation, O - The notation O(n) is the formal way to express the upper bound of an algorithm's running time. It measure the worst time complexity on the longest amount of ~~amount~~ time an algorithm can possibly take to complete.

   e.g. for funcⁿ f(n):

   $f(n) = O\{g(n)\}$, $\forall c > 0, n > n$.

   $f(n) \le c \cdot g(n)$

   g(n) is tight upper bound of f(n).

2. Big Omega notation, Ω - The notation Ω(n) is the formal way to express the lower bound of an algorithm's running time. It measures the best case time complexity or the best amount of time an algorithm can possibly take to complete.

   e.g. $f(n) = \Omega(g(n))$ $\forall$ $c > 0, n \geq n_0$

   $f(n) \geq c \cdot g(n)$

3. Theta notation, θ - This notation is formal way to express both lower bound and the upper bound of an algorithm's running time.

   e.g. $f(n) = \theta(g(n)$

   if $c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$ $\forall n \geq \max(n_1, n_2)$

   $c_1 > 0$, $c_2 > 0$

Q2. What should be the time complexity of -

for (i=1 to n) { i = i * 2; }

→ for (i=1 to n) // O(log n)

{

i = i * 2; // O(1)

}

for $i = 1, 2, 4, 8, \dots 2^k$, this means (k) times as per this code, it will run till $2^k = n$ which means $k = \log n$ then,

complexity = $O(\log n)$

$$\sum_{i=1}^{n} 1 + 2 + 4 + 8 + \dots + n$$

T_k (k^{th} term) = ar^{k-1}

$a = 1$, $r = \frac{2}{1} = 2$

$$\Rightarrow ar^{k-1} = n = 1 \cdot 2^{k-1} = 2^{k-1}$$

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

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

$$\Rightarrow k = \log(2n)$$

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

3. $T(n) = \{ 3T(n-1)$ if $n > 0$, otherwise $1 \}$

→ $T(n) = 3T(n-1)$ — ①

$\Rightarrow T(n-1) = 3T(n-2)$ — ②

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

$\Rightarrow T(n) = 9T(n-2)$ — ③

Now, $T(n-2) = 3T(n-3)$ — ④

$\therefore T(n) = 9(3T(n-3))$ (from eqⁿ ③ & ④)

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

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

Put $n - k = 0 \Rightarrow n = k$

$T(n) = 3^n T(n-n) = 3^n \cdot T(0)$

$\Rightarrow T(n) = 3^n \cdot 1$ $\{T(0) = 1\}$

$\Rightarrow T(n) = O(3^n)$

4. $T(n) = \{2T(n-1) - 1, \text{ if } n>0, \text{ otherwise } 1\}$

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

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

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

$\Rightarrow T(n) = 4T(n-2)-2-1$

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

$\Rightarrow T(n) = 4(2T(n-3)-1)-2-1$

$\Rightarrow T(n) = 8T(n-3)-4-2-1$

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

Put $n-k=0 \Rightarrow n=k$

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

$\Rightarrow T(n) = 2^n \cdot 1 - (2^n - 1)$

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

$\therefore T(n) = O(1)$

Q5 What should be time complexity of -

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

$\rightarrow$ $i = 1, 2, 3, 4, 5, 6 \dots$

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

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

$1+2+3+4+\dots k > n$

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

$\Rightarrow k^2 > n$

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

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

6. Time Complexity of -

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

→ $i = 1, 2, 3 \ldots n$

$i^2 = 1, 4, 9 \ldots n$

$i^2 <= n$ & $i <= \sqrt{n}$

k^{th} term, $T_k = a + (k-1)d$

$a = 1, d = 1$

$T_k <= \sqrt{n}$

$\sqrt{n} = 1 + (k-1) \cdot 1$

$\Rightarrow \sqrt{n} = k$

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

7) Time complexity of -

```
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 ++;
}
```

→

i	j	K	time
n/2	log n	log n	(n+1)/2
n/1	⋮	⋮	⋮
n	log n	log n	

$O(i * j * K) = O\left[\left(\frac{n+1}{2} * \log n * \log n\right)\right]$

$= O\left(\left(\frac{n+1}{2}\right) * (\log_2 n)^2\right)$

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

8. Time Complexity of -

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

→ $T(n) = T(n-3) + n^2$

$T(1) = 1$

Put $n = n-3$

$T(n-3) = T(n-6) + (n-3)^2$

$\Rightarrow T(n) = T(n-6) + (n-3)^2 + n^2$

$T(n-6) = T(n-9) + (n-6)^2$

$\Rightarrow T(n) = T(n-9) + (n-6)^2 + (n-3)^2 + n^2$

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

$n - 3k = 1 \Rightarrow k = \frac{n-1}{3}$

$\therefore T(n) = T(1) + \left[n - 3\left(\frac{n-1}{3} - 1\right)\right]^2 + \left[n-3\left(\frac{n-1}{3} - 2\right)\right]^2 + \cdots + n^2$

$T(n) = 1 + 4^2 + 7^2 + \cdots + n^2$

$\Rightarrow n^2 + \cdots + 1$

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

9. Time complexity of -

```
void function (int n) {
    for (i=1 to n) {
        for (j=1; j<=n; j=j+i)
            printf ("*");
    }
}
```

→

i	j
1	n times
2	$1+3+5+\cdots+n$ times

$a_n = a + (k-1)d$

$a = 1,\ d = 2$

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

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

$\Rightarrow k = \frac{n+1}{2}$ (no. of terms)

for $i = 2$, $j = \frac{n+1}{2}$ times

for $i = 3$, $j = 1 + 4 + 7 + \cdots + n$ times

$n = 1 + \cancel{(k} + (k-1)d \Rightarrow n = 1 + (k-1) \cdot 3$

$\frac{n-1}{3} + 1 = k \Rightarrow k = \frac{n+2}{3}$ (no. of terms)

Generalising,

for $(i = n)$, $j = \frac{n+k-1}{i}$ times

$T(n) = n + \frac{n+1}{2} + \frac{n+2}{3} + \cdots + \frac{n+k-1}{k}$ (n term)

$\therefore \sum_{i=1}^{n} \frac{n+k-1}{k} \Rightarrow \frac{\sum_{i=1}^{n} n + \sum_{i=1}^{n} k - \Sigma 1}{k}$

$\Rightarrow \frac{\frac{n(n+1)}{2} + nk - n}{k} \Rightarrow \frac{\frac{n^2+n}{2} + nk - n}{k}$

$\Rightarrow \frac{n^2 + n + 2nk - 2n}{2k}$

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

10) for the func., n^k & c^n, what is asymptotic relationship b/w these functions ?
Assume that $k >= 1$ & $c > 1$ are constants. Find out the value of c & n_0 for which relation holds.

$\rightarrow n^k = O(c^n)$

as $n^k \leq a \cdot c^n \ \forall\ n \geq n_0$, for some constant $a > 0$

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq O(2)$

$n_0 = 1$ & $c = 2$