Ayush Dhiman
(CSE) Section H
Roll No.-28

PAGE No .....

DAA Tutorial -2

Asymptotic notations are mathematical tools to represent the time complexity of algorithms for asymptotic analysis.

There are mainly three asymptotic notations-

1) Big O notation - The Big O notation defines an upper bound of an algorithm, it bounds a function only from above eg insertion sort take linear time in best case and quadratic time in worst case. We can safely say that the time complexity of insertion sort is $O(n^2)$

2) θ notation - The theta notation bounds a function from above and below, so it defines exact asymptotic behaviour. A simple way to get Theta notation of an expression is to drop low-order terms & ignore leading constants

TEACHER'S SIGNATURE .....

eg $3n^2 + 6n^2 + 6000 = \Theta(n^3)$

3) Ω Notation - Ω notation provides a asymptotic lower bound.
eg insertion sort time complexity in Ω notation will be $\Omega(n)$.

Ans2

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

$i = 1, 2, 4, 8 \ldots n$

$n = a r^{k-1}$

$n = 2^{k-1}$

$\log n = k - 1$

$k = \log_2 n + 1$

$T(n) = \Theta(\log n)$

Ans3 $T(n) = 3T(n-1)$ if $n > 0$ otherwise 1

$T(n) = 3T(n-1)$ $\quad T(0) = 1$

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

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

$T(n) = 3.3T(n-2)$

$T(n) = 3.3.3T(n-3)$

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

DATE .....

PAGE No .....

Let $n-k=0$

$n=k$

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

as $T(0)=1$

$T(n) = 3^n$

Time complexity $= O(3^n)$

Ans 4 $T(n) = 2T(n-1)-1$ if $n>0$ otherwise 1

$T(n) = 2T(n-1)$ $T(0)=1$

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

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

$T(n) = 2.2T(n-2) - 2 - 1$

$T(n) = 2.2(2T(n-3)-1) - 2 - 1$

$T(n) = 2.2.2T(n-3) - (1+2+2.2)$

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

Let $n-k=0$

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

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

$= 2^n - 2^n + 1$

$T(n) = O(1)$

TEACHER'S SIGNATURE .....

DATE ..... PAGE No .....

Ans: Time complexity of

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

for i=1, s=1

i=2, s=1+2

i=3, s=1+2+3

sum of n natural numbers so,

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

$$s <= n$$

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

$$\frac{k^2+k}{2} <= n$$

$$k^2 <= n$$

$$k <= \sqrt{n}$$

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

TEACHER'S SIGNATURE .....

DATE .....

PAGE No .....

Ans 6 Time complexity.

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

$i = 1, 2, 3, 4 \quad n$

$i^2 = 1, 4, 9, 16 \quad n$

So $i^2 <= n \Rightarrow i <= \sqrt{n}$

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

$a \quad i <= \sqrt{n}$

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

$k <= \sqrt{n}$

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

Ans 7

```
void fun (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++;
}
```

TEACHER'S SIGNATURE .....

DATE .....

PAGE No .....

i	j	k
$n/2$	$\log_2 n$	$\log_2 n$
⋮	⋮	⋮
n	$\log_2 n$	$\log_2 n$

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

$$O(i*j*k) = O\left(\left(\frac{n}{2}+1\right) \times \log n \times \log n\right)$$

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

Ans 8 Time complexity -

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

TEACHER'S SIGNATURE .....

DATE .....

PAGE No .....

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

$T(1) = 1$

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

$+(n-3)$

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

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

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

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

$+ \_\_\_ + n^2$

let $n - 3k = 1$

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

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

$\_\_\_ + n^2$

$T(n) = T(1) + [n - (n-1-3)]^2 + \_\_\_ + n^2$

$T(n) = 1 + (3+1)^2 + (4+1)^2 + \_\_\_ + n^2$

$T(n) = 1 + 4^2 + 5^2 + \_\_\_ + n^2$

$T(n) = \frac{n(n+1)(2n+1)}{6}$

$T(n) = \Theta(n^3)$

TEACHER'S SIGNATURE .....