Name : Gaurav Chauhan
Section - D
ROLL NO - 01

Design and Analysis Of Algorithm

Tutorial - 1

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

Ans. Asymptotic notations are those notations that describing the limiting behaviours of a function.

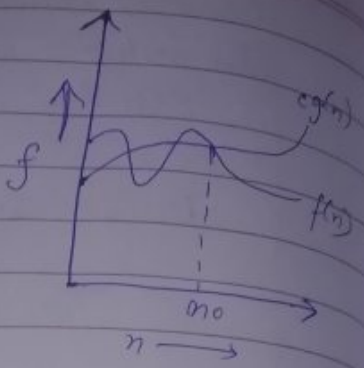
There are 3 different types of notations -

→ Big Oh (O)
→ Big (Ω)
→ Bing (Θ)

* Big Oh (O) notation gives an uppes bond for a function $f(n)$ to within a constant factor.

$$f(n) = O(g(n))$$

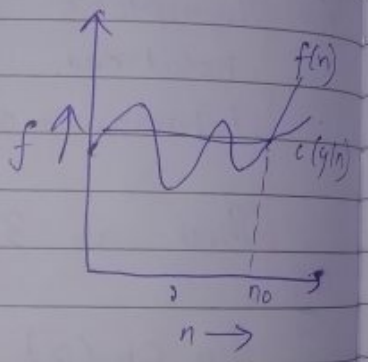
iff
$f(n) \leq c \cdot g(n)$
$\forall\ n \geq 0$.



ex for(i=1; i<=n; i++)
{
sum += i
}

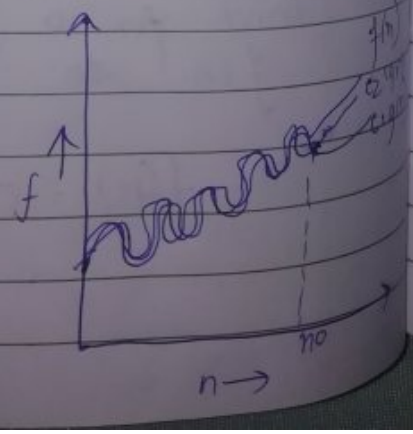
$\Rightarrow O(1+n+n+n)$
$= O(n)$

* Big Omega Notation
$f(n) = \Omega(g(n))$
g(n) is "tight" lowerbound
$f(n) = \Omega(g(n))$
iff
$f(n) \geq c \cdot g(n)$
$\forall\ n \geq n_0$.



* Θ (Theta
$f(n) = \Theta(g(n))$
g(n) is both "tight" upper and lower bond of fn.
$f(n) \approx \Theta(g(n))$
iff $c_1 g(n) \leq f(n) \leq c_2 g(n)$

$\forall\ n \geq \max(n_1, n_2)$



2. What should be time complexity of

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

Ans $i = 1, 2, 4, 8, 16, 32 \dots n$

$= 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, \dots n$

$= 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, \dots 2^k$

Time = no of iteration = no of terms = $0 \to k = k+1$

where

$n = 2^k$

$k = \log_2 n$ $\quad k+1 = \log_2 n + 1$

Time complexity

$\Rightarrow O(\log_2 n)$

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

Ans $T(n) = 3T(n-1)$ — (1)

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

$T(n-1) = 3T(n-2)$ — (2)

put $T(n-1)$ in (1)

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

$T(n) = 9T(n-2)$ — (3)

put $n = n-2$ in (1)

$T(n-2) = 3T(n-3)$ — (4)

put in (3)

$T(n) = 27T(n-3)$ — (5)

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

5.

Let $k = n$

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

$T(n) = 3^n$

Complexity = $O(3^n)$

Ans

4. $T(n) = \begin{cases} 2T(n-1) - 1 & \text{if } n > 0 \\ \text{otherwise } 1 \end{cases}$

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

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

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

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

$= 2^3 T(n-3) - 2^2 - 2^1 - 2^0$

$= 2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3}$

$\cdots\cdots 2^2 - 2^1 - 2^0$

$= 2^n - 2^{n-1} - 2^{n-2} - \cdots 2^2 - 2^1 - 2^0$

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

$T(n) = 1$

$\Rightarrow O(1)$

5. What should be time complexity

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

Ans We can define term s according to relation $s_i = s_{i-1} + i$

value contained in 's' at i^{th} iteration = sum of first 'i' positive integers.

loop terminates if

$$1 + 2 + 3 \cdots k > n$$

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

$$k^2 = n$$

$$k = \sqrt{n}$$

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

6. Time complexity of

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

Ans as $i^2 <= n$

$i <= \sqrt{n}$

$i = 1, 2, 3, 4, \ldots k$ for some k^{th} iteration

and each is less than $\sqrt{n}$

$$T = \sum_{i=1}^{k} 1 <= \sum_{i=1}^{k} \sqrt{n}$$

$1 + 2 + 3 + 4 + \ldots + k <= k\sqrt{n}$

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

$k <= 2\sqrt{n}$

$1 + 1 + 1 + \ldots + k <= k\sqrt{n}$

$T = 1+1+1 \ldots$ k times $= k <= \sqrt{n}$

$O(\sqrt{n})$

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 ++                    // O(1)
}
```

Ans Calculate T for

```
for (k = 1 ; k<= n; k = k * 2)
```

$k = 1, 2, 4, 8, 16 \cdots n$

$= 2^0, 2^1, 2^2, \cdots 2^m$

where $2^m = n$

$m = \log_2 n$

T_1 = no of iterations

= no of terms = $0 \rightarrow m$ ($m+1$ terms)

$= m+1$

$$\boxed{T_1 = \log_2 n + 1}$$

Calculate T_2 for

for (j=1 ; j<=n ; j=j+2)

$j = 1, 2, 4, \ldots, n$

$= 2^0, 2^1, 2^2, \ldots, 2^m$

where $2^m = n$

$m = \log_2 n$

T_2 = no of iterations

= no of terms = $0 \to m$ ($m+1$ terms)

$= m+1$

$$\boxed{T_2 = \log_2 n + 1}$$

Calculate T_3 for

for (i = n/2 ; i<=n ; i++)

T_3 = total iterations $= n - \frac{n}{2} + 1 = \boxed{\frac{n}{2} + 1}$

We have nested loop

$T_3 = \left(\frac{n}{2}+1\right)\left(\log_2 n + 1\right)\left(\log_2 n + 1\right)$

$T_3 = \log_2^2 n \left(\frac{n}{2}+1\right) + \log_2 n \left(n+2\right) + \frac{n}{2}$

discard less significant terms

$\Rightarrow \boxed{O\left(n \log_2^2 n\right)}$

8.

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

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

$a = 1 \quad b = 3 \quad f(n) = n^2$

$c = \log_3 1 = 0$

$n^0 = 1 > (f(n) = n^2)$

$T(n) = \Theta(n^2)$

9. Time Complexity of

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

Ans $T = \frac{n-1}{1} + \frac{n-1}{2} + \frac{n-1}{3} \cdots + \frac{n-1}{n-1}$

$= n\left\{\frac{1}{1} + \frac{1}{2} + \frac{1}{3} \cdots + \frac{1}{n-1}\right\} - \left\{\frac{1}{1} + \frac{1}{2} + \frac{1}{3} \cdots + \frac{1}{n-1}\right\}$

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

$T = (n-1) \log (n-1)$

$\Rightarrow O(n \log n)$

10 as given n^k & c^n
relation b/w n^k & c^n is

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

as $n^k \leq c^n$

$\forall\ n \geq n_0$ and constant $a > 0$

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq a 2^1$

$\Rightarrow n_0 = 1$ & $c = 2$