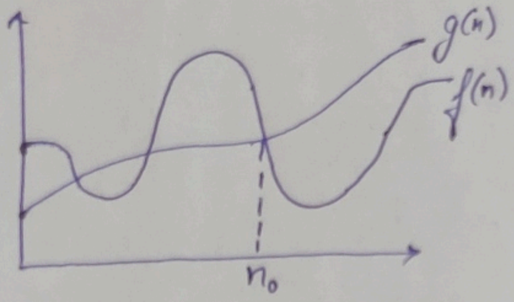
Rahan Bhandari
CST-44

Tutorial-01

Que 1. **Asymptotic Notation** :- They are the Mathematical notation used to describe the running time of algorithm when the input tends towards a particular value or a limiting value.

There are mainly three asymptotic notations :-

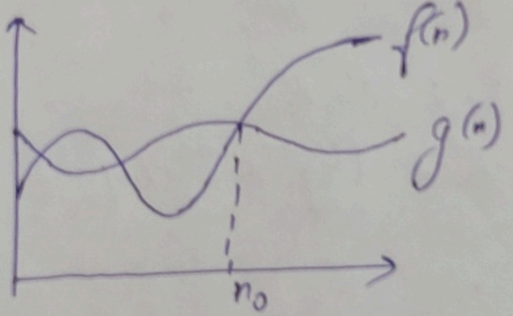
① **Big O Notation (O)** :- (i) Provide worst complexity
(ii) Provide upper bound of an running time algo.



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

$O(g(n)) = \{ f(n)$: there exist +ve constant c & n_0 such that $0 \leq f(n) \leq g(n)$ for all $n \geq n_0 \}$

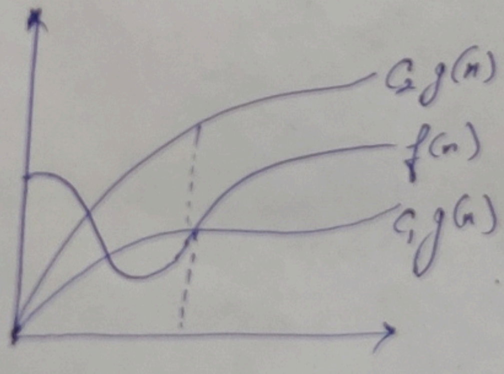
② **Omega Notation (Ω)** :- (i) Provides best case complexity
(ii) Provide lower bound of running time algo.



$f(n) = \Omega(g(n))$

$\Omega(g(n)) = \{ f(n)$: there exist +ve constant c & n_0 ~~≠0~~ such that $0 \leq (g(n)) \leq f(n)$ for all $n \geq n_0 \}$

③ **Theta Notation (θ)** :- • Used for analysing avg. time complexity



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

$\theta(g(n)) = \{ f(n)$: there exist +ve constant c_1, c_2 n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0 \}$

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

Sol^n:- Let us solve this f^{n} with substitution.

$$T(n) = \cancel{2}(3T(n-1)$$
$$= 3(3T(n-2))$$
$$= 3^2T(n-2)$$
$$= 3^3T(n-3) \text{ ------} = 3^nT(n-n)$$
$$= 3^nT(0)$$
$$\Rightarrow 3^n$$

The complexity of this f^n is : $\underline{O(3^n)}$.

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

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

$$= 2(2T(n-2)-1)-1$$
$$= 2^2(T(n-2))-2-1$$
$$= 2^2(2T(n-3)-1)-2-1$$
$$= 2^3T(n-3)-2^2-2^1-2^0$$
$$\vdots$$
$$= 2^nT(n-n)-2^{n-1}-2^{n-2}-2^{n-3} \text{ ----} 2^2-2^1-2^0$$
$$= 2^n-2^{n-1}-2^{n-2}-2^{n-3} \text{ ----} 2^2-2^1-2^0 \quad \{\because 2^{n-1}+2^{n-2}+\dots+2^0=2^n-1\}$$
$$= 2^n-(2^n-1)$$

$T(n) = \underline{1}$

Que. 5. Time Complexity of -

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

Sol^n:- Let the loop execute 'k' times. It will execute as long as s is less than n.

for $(i=1)$
$s = 1+2;$
for $(i=2)$
$s = 1+2+3$
for $(i=k)$

$= 1+2+\dots+k <= n$

$= k\left(\frac{k+1}{2}\right) <= n \quad = \left(\frac{k^2+k}{2}\right) \le n$

$\Rightarrow O(k^2) <= n \Rightarrow k = O(\sqrt{n})$

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

Que. 6. Void function (int n) {
int i, Count = 0;
for (int i=1, i*i <= n, i++)
Count ++. $\rightarrow O(1)$
}

Soln:- Let 'k' be max. +ve value such that

$k^2 \le n$

$\therefore \quad k = \sqrt{n}$

$i^2 \le n$

$\therefore \quad \sum_{i=1}^{n} \Rightarrow 1+1+\dots k$ times

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

Que 7. Void function (int n) {
int i, j, k, Count = 0;
for (i = N/2 , i <= n, i++)
{
for (j=1, j<=n, j=i*2)
{
for (k=1; k<=n; k = k*2)
count ++;
}
}

Soln:- Let 'm' be highest value of 2^m such that $2^m <= n$

$\therefore \quad m = \log_2 n$

$\Rightarrow$ for $i = n/2$, $j = \log n$, $k = \log n$
$i = n/2 + 1$, j ,, , k ,,
& $\vdots$
$i = n$, $j = \log n$, $k = \log n$

$\therefore \sum_{i=n/2}^{n} j \times k$

$\Rightarrow n/2 (\log n)^2$

$\Rightarrow T(n) = O(n \log_2^2 n)$

Que 8. function (int n)

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

Sol^n:- for ($i = 1$ to n)
we get $j = n$ times every turn

$\therefore \; i \times j = n^2$

Now, $T(n) = n^2 + T(n-3)$;

$$\left.\begin{aligned} T(n-3) &= T(n-6) + (n-3)^2 \\ T(n-6) &= T(n-9) + (n-6)^2 \end{aligned}\right\} k \text{ terms}$$

$\vdots$

$T(1) = 1$;

Now substituting each value in $T(n)$.

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \dots + 1$

Let $(n-3k) = 1$

$\therefore k = (n-1)/3$

Total terms $= k+1$

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \dots + 1$

$T(n) \approx n^2 + n^2 + n^2 + \dots$ (k times + 1)

$T(n) \approx kn^2$

$T(n) \approx k(n-1)/3 \times n^2$

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

Que. 9. function (int n)

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

Solⁿ:- for $i=1$ $j = 1+2+\dots$ $(n \geq j+1)$

$i=2$ $j = 1+3+5+\dots$ $(n \geq j+1)$

$i=3$ $j = 1+4+7+\dots$ $(n \geq j+1)$

n^{th} term of AP is

$T(m) = a + d \times m$

$T(m) = 1 + d(m)$

$(n-1)/d = m$

$\therefore$ for $i=1$ $(n-1)/1$ times

$i=2$ $(n-1)/2$ times

$i=3$ $(n-1)/3$ times

$i=n-1$

We get, $T(n) = i_1 j_1 + i_2 j_2 + \dots i_{n-1} j_{n-1}$

$= \frac{(n-1)}{1} + \frac{(n-2)}{2} + \frac{(n-3)}{3} \dots +$

$$= n + \frac{n}{2} + \frac{n}{3} + \dots + \frac{n}{n-1} - n \times 1 + 1$$

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

$$= n \times \log n - n + 1$$

$\because \int \frac{1}{x} = \log x \quad \therefore T(n) = O(n \log n)$

Qu. 10. We have been given

n^k and c^n

as $k=1$ & $c>1$

$\Rightarrow$ for values $k=1$, $c>1$

We have $c^n > n^k$

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

$\forall n \geq n_0$, & some constant $k > 0$

$\Rightarrow k_0 c^n \geq n^k$

for $c>1$ & $n \geq 1$

we get

$\Rightarrow k_0 c \geq 1$

$\therefore \boxed{c > 1 \ \& \ n_0 = 1}$

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

Solⁿ:- $i = 1, 2, 2^2, \dots 2^k$

$2^k \leq n \Rightarrow k = \log_2 n$

$2^k \leq n \Rightarrow k = \log_2^n$

$\therefore \sum_{i=1}^{k} \Rightarrow 1+1+1 \dots k$ times

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

Rohan Bhandari

2017536