Assignment-1

Name - Vibha Rani
Rollno - 45
Section - CE [Computer Engineering]
University Rollno - 2017395

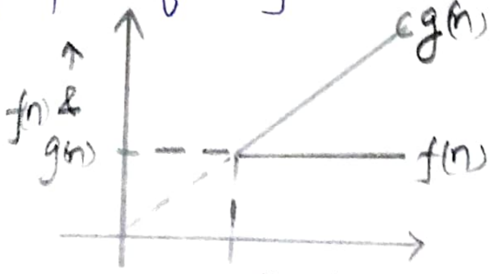
Tutorial-1

Ques 1:- Asymptotic Notations :- It gives us an idea about how good a given algorithm is, as compared to some others algorithm.
There are 3 types widely used Asymptotic Notations
(i) Big O (O)
(ii) Big Omega (Ω)
(iii) Big Theta (Θ)

a) Big O Notation :- This notation defines an upper bound of an algorithm. it bounds a function only from above

$$f(n) \leq c\, g(n)$$

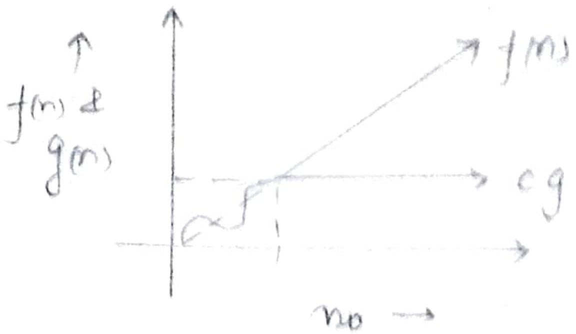
Graph of Big O Notation.



b) Big Omega (Ω) Notation :- This notation provides an asymptotic upper bound on a function. Ω notation provides an asymptotic lower bound.

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

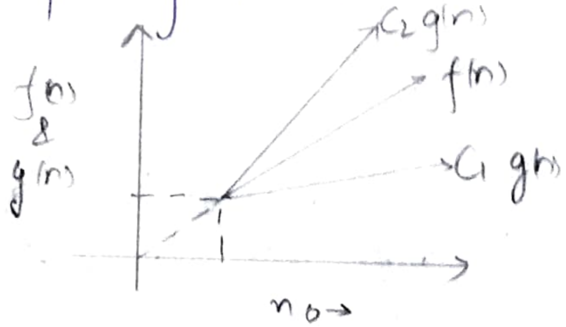
Graph of Big Omega Notation



c) Theta (Θ) Notation:- This notation bounds a function from above and below. so it defines exact asymptotic behaviours.

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

Graph of Theta (Θ) Notation:



Ques 2:- Time complexity of loop:

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

i = 1, 2, 4, 8, ... n

It forms a gp so

$n = 2^{K-1}$

$\log_2 n = K-1$

$K = \log_2 n + 1$

$O(K) = O(\log_2 n + 1)$

$T(n) = O(\log_2 n)$

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

$T(n) = 3T(n-1)$ — (1)

put $n = n-1$ in eq^n 1

$T(n-1) = 3T(n-2)$ — (2)

put eq^n (2) in eq^n (1)

$T(n) = 3[3T(n-2)]$ — (3)

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

$T(n) = 3T(n-3)$ — (4)

put eq^n (4) in eq^n (3)

$T(n) = 3^2[3T(n-3)]$ — (5)

$T(n) = 3^3[T(n-3)]$ — 6

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

put $n-k = 0$

$\boxed{n = k}$

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

$\boxed{T(n) = O(3^n)}$

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

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

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

$$T(n) = 2[2T(n-2) - 1] - 1$$
$$T(n) = 2^2 T(n-2) - 2 - 1$$
$$T(n) = 2^2 [2T(n-3) - 1] - 2 - 1$$
$$T(n) = 2^2 [2T(n-3)] - 2^2 - 2^1 - 1$$
$$T(n) = 2^3 T(n-3) - 2^2 - 2^1 - 1$$
$$\vdots$$
$$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} - \cdots - 2^1 - 2^0$$

put $n-k=0$

$$\boxed{n=k}$$

$$T(n) = 2^n T(0) - 2^{n-1} - 2^{n-2} - \cdots - 2^1 - 2^0$$
$$a = 2^{k-1} / 1$$
$$r = \frac{2^{k-2}}{2^{k-1}} = 2^{k-2-k+1} = 2^{-1} \text{ or } \frac{1}{2}$$

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

$$\boxed{T(n) = O\, 2^n}$$

Ques 5:- int i = 1 , s = 1;

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

$i = 1, 2, 3, 4, 5 \ldots\ldots + T(n)$ — (1)

$S = 1, 3, 6, 10, 15 \ldots\ldots T(n-1) + T(n)$ — (2)

$S = 1 + 3 + 6 + 10 + 15 + \ldots\ldots + T(n)$

$- \; S = 1 + 3 + 6 + 10 + \ldots\ldots + T(n-1) + T(n)$

$0 = 1 + 2 + 3 + 4 + 5 + \ldots + T(n) + \ldots - T(n-1) - T(n)$

$0 = 1 + 2 + 3 + 4 + \ldots + n - Tn$

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

for K iterations,

$1 + 2 + 3 + \ldots K \leq n$

$\frac{K(K+1)}{2} \leq n$

$\frac{K^2 + K}{2} \leq n$

$O(K^2) \leq n$

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

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

Ques 6 - void function (int n)

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

$i = i * i = i^2$ | $i = 1, 2, 2^2,$

$i^2 = i^2 * i^2 = i^4$

$i, i^2, i^4 \ldots \; i^2$

$i^{2k} = n$

$i^2 \leq n$

$i \leq \sqrt{n}$

$i = 1, 2, 3, 4, \ldots \ldots \sqrt{n}$

$i = 1 + 2 + 3 + 4 + \ldots \; \sqrt{n}$

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

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

Ques 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 ++ ;
     }
  }
}
```

for k · k^2

$k = 1, 2, 4, 8 \dots n$

for $i = n/2$. $i = n/2+1$

$k = 1, 2, 4, 8, \dots n$. $k = 1, 2, 4, 8 \dots n$

$j = \log n$. $j = \log n$

$T(n) = O[\frac{n}{2} \times \log n . \log n]$

$T(n) = O[n \log_2^n]$

Ques 8 - function(int n) — T(n)

```
{ if (n==1)          — O(1)
    return;
  for (i=1 to n)     — T(n)
  {  for (j=1 to n)  → T(n)
     {  printf("*");
     }
  }
```

$T(n) = T(n)$

$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} (1)$

$T(n) = \sum_{i=1}^{n} 1+1+1 \dots n$

$T(n) = \sum_{i=1}^{n} n = n+n+n+\dots n$ times

$n(n) = n^2$

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

$$T(n) = \sum_{i=1}^{n} n$$

$= n + n + \dots n$ times

$= n[1 + 1 + \dots n \text{ times}]$

$= n(n)$

$$\boxed{T(n) = O(n^2)}$$

Ques 9: void function (int n)

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

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} (1)$$

$$T(n) = \sum_{i=1}^{n} (1 + 1 + 1 + \dots n \text{ times})$$

$$T(n) = \sum_{i=1}^{n} n$$

$T(n) = \sum n (n\ 1 + 1 + \dots n \text{ times})$

$= n(n)$

$$\boxed{T n = O(n^2)}$$

Ques 10: For the function, n^k and c^n. what is asymptotic relation between these functions? Assume that $k >= 1$ and $c > 1$ are constants. find the value of c and n_0 for which relation holds

Relation b/w n^k and c^n is $n^k = O(k^n)$

as $n^k < a c^n$

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

for $n_0 = 1$

$c = 2$

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

$\boxed{n_0 = 1 \text{ and } c = 2}$