DAA Assignment-1

Name: Harshit Kushwaha
Roll no: 96
Sec: f

Ans1- Asymptotic Notation:

The notation are used to tell the complexity of an algorithm with respect to the input size.

Different Asymptotic notation:

1→ Big oh (O):

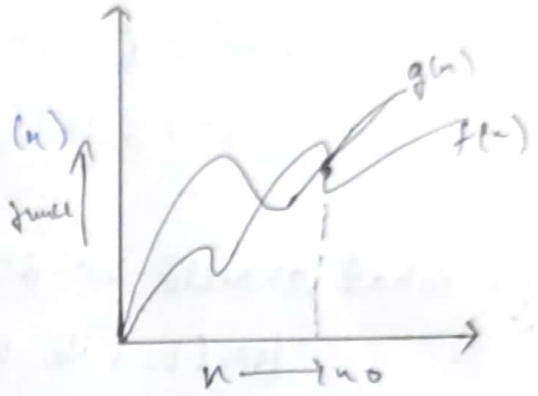
It is a upper tight bound of f(n).

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

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

e.g. $g(n) = n^2 + n + 5$

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



2→ Big Omenga (Ω)

It is a lower tight bound of f(n).

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

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

e.g. $g(n) = n^2 + \log n$

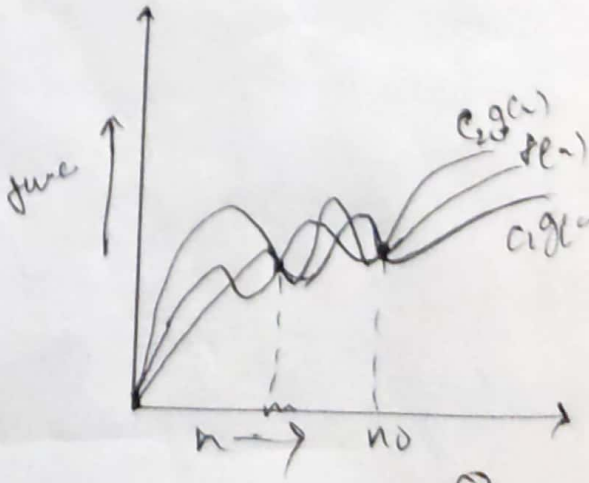
$f(n) = \Omega(\log n)$

3→ Theta (Θ)

It gives us a range of f(n)

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

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

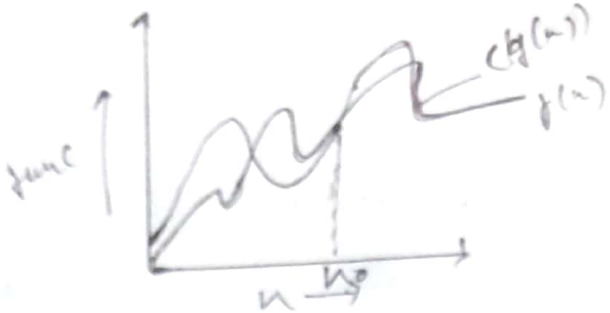


4. Small oh (o):

It is a upper bound of $f(n)$

$f(n) = o(g(n))$

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

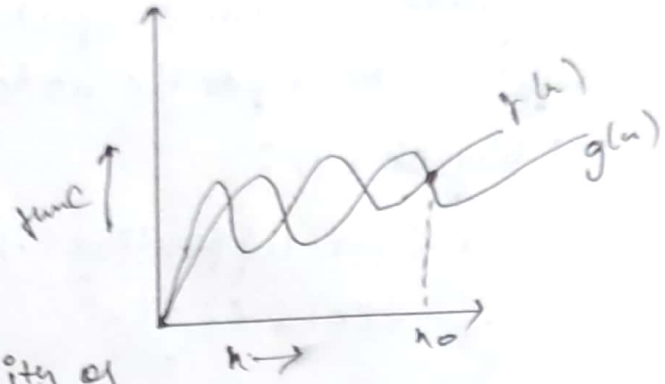


5. Small omega (ω)

It is a lower bound of $f(n)$

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

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



Q2 what should be time complexity of

```
for(i=1 to n)        // 1, 2, 4, 8, 16 ---- n
    {
    p=i*2;
    }
```

$\sum_{i=1}^{n} = 1+2+4+8+16 \cdots n$

k^{th} value $= T_k = ar^{k-1}$

$a = 1, r = 2$

$T_k = 1 \times 2^{k-1} = 2^{k-1}$

Put $n = 2^{k-1}$

$2n = 2^k$

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

$\Rightarrow \log_2 2 + \log_2 n = k$

$\Rightarrow \log_2 n + 1 = k$

$\Rightarrow \log n + 1 = k$

$\Rightarrow O(\log n)$

Q3 $T(n) = \{3T(n-1)$ if $n>0$, otherwise $1\}$

$T(0) = 1$

$T(n) = 3T(n-1)$ — (1)

Put $n = n-1$

$T(n-1) = 3T(n-2)$ — (2)

Put value of $T(n-1)$ in (1)

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

$= 9T(n-2)$ — (3)

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

$T(n-2) = 3T(n-3)$ — (4)

Put value of $T(n-2)$ in (3)

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

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

Put $k = n$

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

$T(n) = 3^n$

Q4 $T(n) = \{2T(n-1) - 1$ if $n>0$, otherwise $1\}$

$T(n) = 2T(n-1) - 1$ — (1)

$T(0) = 1$

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

$T(n-1) = 2T(n-2) - 1$ — (2)

Put value of $T(n-1)$ in (1)

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

$T(n) = 4T(n-2) - 3$ — (3)

~~Put value~~ Put $n = n-2$ in (1)

$T(n-2) = 2T(n-3) - 1$ — (4)

Put value of $T(n-2)$ in (3)

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

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

$T(n) = 2^k T(n-k) - (2^{k-1} + 2^{k-2} + 2^{k-3} \ldots\ldots 1)$

Consider the G.P $\cdot 2^{k-1} + 2^{k-2} \ldots\ldots 1$

$a = 2^{k-1}$

$r = 1/2$

$\Rightarrow \frac{a(1-r^k)}{1-r}$

$=$

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

$= 2^k T(n-k) - (1 + 2 + 4 \ldots\ldots 2^{k-1})$

$\Rightarrow 2^k - \left[a \cdot \frac{(2^k - 1)}{2-1} \right]$

$\Rightarrow 2^k - 2^k + 1$

$T(n) = 1$

5. what should be time complexity of -

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

We can define the term 's' according to relation $s_i = s_{i-1} + i$. The value of 'i' increases by one for each iteration. The value contained in 's' at the i^{th} iteration is the sum of the first 'i' positive integers. If k is total number of iterations taken by the program, then while loop terminates if

$1 + 2 + 3 \ldots\ldots k = \frac{k(k+1)}{2} > n$

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

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

6. Time complexity of

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

$i^2 <= n$

$\Rightarrow i <= \sqrt{n}$

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

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

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

$\Rightarrow T(n) = \frac{n \times \sqrt{n}}{2}$

$T(n) = O(n)$

7. Time complexity of

```
void function(int n)
{ int i, j, k, count=0;
  for(i=n/2; i<=n; i++)  // First loop
    for(j=1; j<=n; j=j*2)  // Second loop
      for(k=1; k<=n; k=k*2)  // Third loop
        count++
}
```

First loop will executes $n/2$ times

Second loop will executes $\log n$ times

Third loop will executes $\log n$ times

$T(n) = \frac{n}{2} \times \log n \times \log n$

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

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);
}
```

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

$T(n-3) + n^2 \quad -(1)$

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

from eq (1):

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

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

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

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

$n - 3k = 0$

$k = n/3$

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

$= T(0) + n\frac{(n+1)(n+2)}{6}$

$T(n) = \frac{n*n*n}{6}$

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

9- Time complexity of-

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

for $i=1 \Rightarrow j = 1, 2, 3, 4, \ldots n = n$

for $i=2 \Rightarrow j = 1, 3, 5, \ldots n = n/2$

for $i=3 \Rightarrow j = 1, 4, 7, \ldots n = n/3$

for $i=n = j=1 \quad = 1$

$$\Rightarrow \sum_{j=n}^{i} = n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \ldots 1$$

$$\Rightarrow \sum_{j=n}^{i} n\left(1 + \frac{1}{2} + \frac{1}{3} \ldots \frac{1}{n}\right)$$

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

10- For the function, n^k & c^n, what is the asymptotic relationship between these function?
Assume that $k \geq 1$ & $c > 1$ are constants. Find out the value of c & n_0 for which relation holds.

The relation between n^k & c^n is

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

as $n^k < ac^n$

$\forall n > n_0$ & some constant $a > 0$

for $n_0 = 1$

$c = 2$

$\rightarrow 1^n \leq a 2^n$

$n_0 = 1$ & $c = 2$