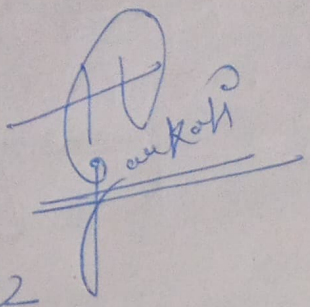
Tutorial - 1

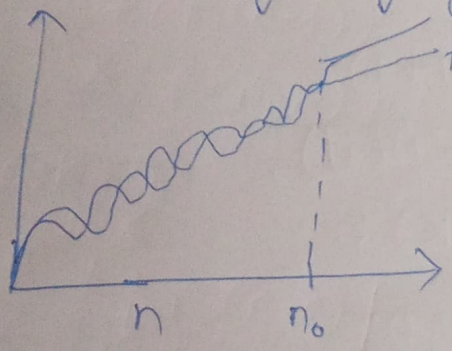
Name → Harsh Garkoti
Section → O
Semester → IV
Roll no → 04
Uni. Ro → 2016758
Date → 10-3-2022



Q-1 Asymptotic Notations

They help you to find complexity of an algorithm when input is very large

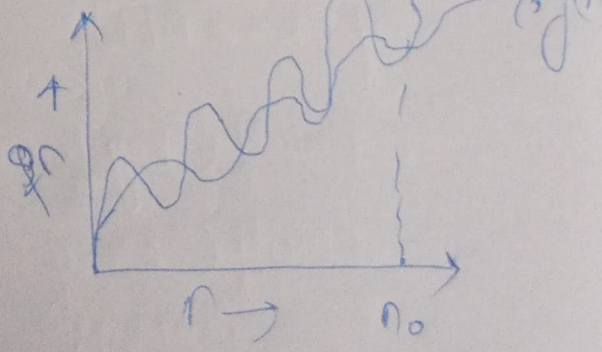
1) Big(O) →



$f(n) = O(g(n))$
iff $f(n) \leq c \cdot g(n)$
$\forall\ n \geqslant n_0$
for some constant $c > 0$

→ g(n) is tight upper bound of f(n)

ii) Big Omega (Ω)
$f(n) = \Omega(g(n))$
g(n) is tight lower bound of f(n)
$f(n) = \Omega(g(n))$
iff $f(n) \geqslant c \cdot g(n)$
$\forall\ n \geqslant n_0$ for some content $c > 0$



iii) Theta(Θ)

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

$g(n)$ is ~~both~~ ~~tight~~ upper bound and lower bound

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

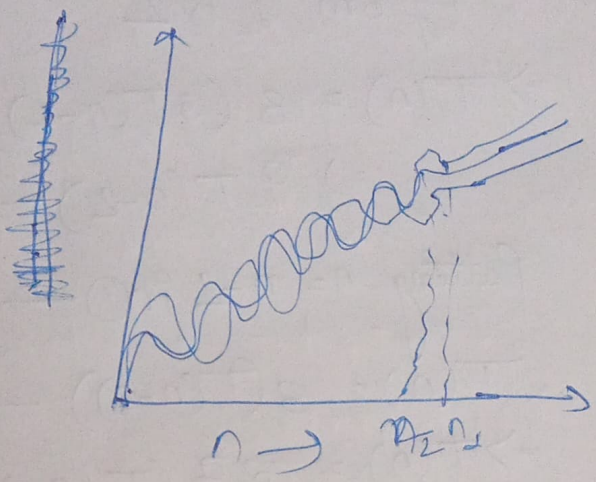
iff

$C_1 g(n) \leq f(n) \leq C_2 g(n)$

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

for some constant

$C_1 > 0$ & $C_2 > 0$



Q-2 T.C

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

$\Rightarrow \sum_{i=1}^{n} 1 + 2 + 4 + 8 + - - - + n$

G.P K^{th} value $\Rightarrow T_K = G_r^{K-1}$

$\Rightarrow 1 * 2^{K-1}$

$\Rightarrow n = n^K$

$\Rightarrow 2n = 2^K$

$\Rightarrow \log 2n = K \log 2$

$\Rightarrow \log_2 + \log n = K \log 2$

$\Rightarrow \log n + 1 = K$

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

$\Rightarrow O(\log n)$

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

$T(n) = 3T(n-1) \longrightarrow (1)$

Put $n = n-1$

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

from 1 & 2

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

$\Rightarrow 9T(n-2) \longrightarrow (3)$

Putting $n = n-2$ in (1) —

$T(n) = 3(T(n-3)) \longrightarrow (4)$

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

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

Putting $n-k = 0$

$\Rightarrow n = k$

$\Rightarrow T(n) = 3^n [T(n-n)]$ $\quad [T(0)=1]$

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

$\Rightarrow T(n) = 3^n \times 1$

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

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

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

Let $n = n-1$

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

from (2) & (1)

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

$\Rightarrow T(n) = 4T(n-2) - 2 - 1$

let $n = n-2$

$\Rightarrow T(n-2) = 2T(n-3)+1 \longrightarrow (4)$

from (3) and (4)

$\Rightarrow T(n) = 4[2T(n-3)-1]-2-1$

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

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

$\Rightarrow GP = 2^{k-1} + 2^{k-2} + 2^{k-3} - \cdots$

$a = 2^{k-1}$

$r = 1/2$

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

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

$= 2^k(1-(1/2)^k)$

$= 2^k - 1$

Let $n-k=0$

$\Rightarrow n=k$

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

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

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

$T(n) = O(1)$

Q-5 T.C of

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

$i = 1, 2, 3, 4 - - -$

$S = 1+3+6+10+15 - - - n$

Sum of $S = 1+3+6+10 \quad \cdots \quad n$

also $S = 1+3+6+10+ \cdots T_{n-1}+T_n$

from ① - ②

① $= 1+2+3+4+ \cdots n = T_n$

$\Rightarrow T_k = 1+2+3+4+ \cdots k$

$T_k = \frac{1}{2} K(K+1)$

$\Rightarrow$ for k iterations

$1+2+3+ \cdots +k <= n$

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

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

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

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

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

Q.6 T.C of

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

$\Rightarrow$ as $i^2 <= n$

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

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

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

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

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

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

Q.7 for k= k*2

K=1, 2, 4, 8 - - - n

=> GP → a=1, r=2

$GP = \frac{a(r^n-1)}{r-1}$

$\Rightarrow \frac{1(2^k-1)}{1}$

$n = 2^k$

$\Rightarrow \log n = k$

=>

i	j	k
1	logn	logn * logn
2	logn	⋮
⋮	⋮	⋮
n	logn	logn * logn

$\Rightarrow O(n * \log n * \log n)$

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

Q-8 T.C of

```
function (int n)
{ int (n==1)
  return;
for (i=1 to n)                    //O(1)
{ for (j=1 to n)                  // i= 1,2,3,4 - - - n
    { Print ("*");                // j= 1,2,3,4 - - - n
    }
}
function (n-3)                    T(n/3)
}
```

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

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

$c = \log_3 1 = 0$

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

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

Q-9 T.C of →

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

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

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

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

for $i=n \Rightarrow j=1$

$\Rightarrow \sum_{j=n}^{1} n + n/2 + n/3 + n/4 + - - - +1$

$\Rightarrow \sum_{j=n}^{1} n\,[1 + 1/2 + 1/3 + 1/4 + - - - + 1/n]$

$\sum_{j=n} n\,[\log n]$

$\Rightarrow T(n) = [n \log n]$

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

Q-10 For functions, n^k & , what is the asymptotic relation between these functions? assume that $k>=1$, & c are constant. Find out the value of c & n_0 for which relation holds

⇒ as given $n^k \in O(c^n)$

relation btw $n^k \in c^n$ is

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

as $n^k \leq a c^n$

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

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq a_1$

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