23rd Feb 2024

Name : AKSHITA DHOUNDIYAL SEM : 04
Roll-no : 11 SEC : ML

ASSIGNMENT - 1

Ques 1.
Ans Asymptotic notations are used in computer science and mathematics to describe the limiting behaviour of a function as its input approcaches infinity. There are following asymptotic notations :

(1) Big O notation (O)
→ Denotes the [g(n)] tight upper bound of f(n)

$f(n) = O\,g(n)$
iff $f(n) \leq (g(n))$
$\forall\ n \geq n_0$

for some constant $c > 0$.

Example :
$f(n) = 2n^2 + 3n + 1$
is $O(n^2)$

(2) Big Omega notation (Ω)
→ Denotes the [g(n)] tight lower bound of f(n)

$f(n) = \Omega\, g(n)$
$f(n) \geq g(n)$
$\forall\ n \geq n_0$

for some constant $c > 0$

Example :
$f(n) = n^2$ is $\Omega(n)$

(3) Theta notation (θ)
→ Theta denotes both tight upper bound and tight lower bound.

$f(n) = \theta\, g(n)$
$f(n) = O\, g(n)$ and $\Omega\, g(n)$

$f(n) = \theta\, g(n)$
iff $c_1 (gn) \leq f(n) \leq c_2\, g(n)$
$\forall\ n \geq \max(n_1, n_2)$ and
for some constant $c_1 > 0$ and $c_2 > 0$

Example :
$f(n) = n^2$ is $\theta(n^2)$

(4) Small o notation (o)
→ Denotes the [g(n)] upper bound of f(n)

$f(n) = o(g(n))$
iff $f(n) < g(n)$
$\forall\ n > n_0$
(for all) $\forall\ c > 0$

Example :
$f(n) = n$ is $o(n^2)$

(5) Small omega notation (ω)
→ denotes the lower bound of f(n)

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

$f(n) \succ (g(n)$

iff $f(n) \succ (g(n)$

$\forall$ $n \succ n_0$

(for all) $\forall$ $C \succ 0$

Example:

$f(n) = n^2$ is $\omega(n)$

Ques2.

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

complexity –

```
for (i=1 ; i<n; i++)
{
    i = i*2;
}
```

GP

2 4 6 ... n/2 n

$a_n = ar^{n-1}$

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

$a_n = 2^k$

$n = 2^k$

[take log on both sides]

$\log_2 n = \log_2 2^k$

$\log_2 n = k \log_2 2$

$k = \log_2 n$

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

Ques3.

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

$T(0) = 1$

$T(n) = 3T(n-1)$ – (1)

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

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

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

put value of $T(n-1)$ in eq (1)

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

$Tn = 9T(n-2)$ – (2)

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

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

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

put value of $T(n-2)$ in eq (2)

$T(n) = 9[\ 3T(n-3)\ ]$

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

$n-k=0$

$\Rightarrow n=k$ [put value of k in n]

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

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

$= 3^n$

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

Ques 4.

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

$T(n) = 2T(n-1)-1$ -①

$T(0) = 1$

put $n = n-1$ in eq ①

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

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

put value of $T(n-1)$ in eq ①

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

$T(n) = 4T(n-2)-2$ -②

put value of $n = n-2$ in eq ①

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

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

put value of $T(n-2)$ in eq ②

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

$T(n) = 8T(n-3)-3$

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

$n-k=0$

$n=k$

$T(0) = 1$

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

$= 2T(0)-1$

$= 2-1 = 1$

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

$= 2T(1)-1$

$= 2(2-1)-1$

$= 4-2-1 = 1$

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

$2T(2)-1$

$= 2(4-2-1)-1$

$= 8-4-2-1 = 1$

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

$T(4) = 2T(3) - 1$
$= 2(8-2-2-1) - 1$
$= 16 - 8 - 4 - 2 - 1 = 1$

$T(n) = O(1)$

Ques 5: what should be time complexity of -

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

i	s
1	1
2	1+1 = 2
3	1+1+2 = 4
⋮	1+1+2+3 = 7
K	⋮

Assume $s > n$ (stopping condition)

$1 + 1 + 2 + 3 + 4 + \cdots + K$

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

$1 + \frac{K(K+1)}{2} > n \Rightarrow \frac{K(K+1)}{2} > n-1$

$K^2 > n-1 \Rightarrow K > \sqrt{n-1}$

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

Ques 6: Time complexity of -

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

3

The loop will run $\sqrt{n}$ times

$i * i < n$

$i^2 < n$

$i < \sqrt{n}$

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

Ques 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++;
}
```

i will run $n/2 + 1$ times

as for j and K -

j	
1	1 2 4 8 ... n
2	$a_n = ar^{n-1}$
⋮	$n = 1 \cdot 2^{k-1}$
K	$n = 2^k/2 \Rightarrow 2n = 2^k$

applying log both sides

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

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

$K = \log_2 n \Rightarrow$ T.C = $O(\log_2 n)$

T.C = $O(n/2) \cdot O(\log_2^2 n) \Rightarrow$ T.C = $O(n/2 \log_2^2 n)$

Ques 8. Time complexity of-

```
function (int n){
  if (n == 1) return;
  for (i = 1 to n){          (n times)
    for (j = 1 to n){        (n times)
      printf("*");
    }
  }
  function (n-3);  T(n-3) times
}
```

] $O(n^2)$

The time complexity of both the inner loops is $O(n^2)$

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

as $T(1) = O(1)$

Thus, $T.C = O(n^2)$

Ques 9. Time complexity of-

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

for j :- $n/1 + n/2 + n/3 + \cdots + n/n$

$n = 1\,2^{k-1} \Rightarrow n = 2^k/2 \Rightarrow 2n = 2^k$

taking log both side

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

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

Thus, the T.C is

$O(n \log_2 n)$

Ques 10. For the functions, n^k and c^n, what is the asymptotic relationship between these functions? Assume that $k \geq 1$ and $c > 1$ are constant. Find the value of c and n_0 for which relation holds.

n^k grows polynomially with n

c^n grows exponentially with n

thus $c^n \geq n^k$

so, n^k is $O(c^n)$

find the value of c and n_0

$\log n^k = \log c(c^n)$

$\Rightarrow c \geq e$ and $n_0 = k$