Malvika Kunwarsen

20022594 B, 58

Date: ______
Page No: ______

Tutorial 1

Ans 1) Assymptotic Notation - It is used to describe running of an algorithm - how much time an algorithm takes with a given input 'n'. There are 3 different notations -

(i) Big O: - It gives the upper bound of an algorithm or how many iterations an algorithm takes in worst case scenario.

(ii) Big Ω -
Big omega describes the lower bound of an algorithm. Eg - bubble sort.

(iii) Θ Notation -
It encloses f from above and below. It represents upper and lower bound of an algorithm. Used for calculating average time complexity.

Ans 2)

```
for (int i=1 to n)
{

    i = i * 2;

}
```

loop runs upto n times
i.e 1 --- n

inside loop:

$i = i * 2$ i.e $i = 1 * 2$ (for first value)

$i = 4$ (2nd value)

$i = 8$ (3rd value)

$2^k = n$

$k = \log_2 n$

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

Q5) $T(n) = 3T(n-1)$ if $n > 0$

$h = n - 1$

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

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

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

$T(n) = 9T((n-1)-2)$

$T(n-1) = 9T(n-3)$

$T(n) = 3(9T(n-3))$

$= 27T(n-3)$

$3T(n-1) \to 9T(n-2) \to 27T(n-3) \cdots 3^n T(n-n)$

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

Q(No-2) $T(n) = 2T(n-1) - 1$ if $n > 0$

$n = n-1$

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

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

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

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

for $n = n-1$

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

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

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

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

$2T(n-1) - 1 \to 4T(n-2) - 1 - 2, \cdots$

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

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

$T(n) = 1$

$T(C) = O(1)$

Ques) Let $i=1, s=1$;

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

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

$k^2 = n$

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

Ans) C) void function (int n)

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

Date:
Page No:

i=1 ⇒ 1+1 <= n ⇒ k = n

for (i=2 ; i+2 <= n ⇒ 2 <= n

, *n <= n ⇒ n <= n

1, 2, ... n ⇒ AP

sum = $\frac{n(n+1)}{2}$ = $O(n^2)$

Que 7) void function (int n)

{

for i, j, k, count = 0;

for (i = n/2 ; i <= n ; i++) ⇒ n/2

for (j = 1 ; j <= n ; j = j*2) $\log_2 n$

for (k = 1 ; k <= n ; k = k*2) $\log_2 n$

count++.

}

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

Ans 8) function (int n)

{

int if (n=1)

return;

for (i=1 to n) → O(n)

{

for (j=1 to n) → O(n)

{

print ("*");

}

function (n-3);

}

T(C) = O(n²)

Ans 9) void function (int n)

{

for (i=1 to n)

{

for (j=1; j<=n; j=j+i)

printf (" *");

}

}

1, 2, 3 ... n

1, 2, 3 ... n-1

1, 2, 3 ... n

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

(iii) (c) function - n^k , c^n

Relation between the two:

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