Chauhan

Date
Page

DAA Assignment.

20

Q1→ Asymptotic notations are mathematical tools to represent the time complexity of algorithms for asymptotic analysis. Asymptotic notations describes the running time of an algorithm for given input.

There are mainly three notations are:

① Big-O Notations (O)

~~② Omega~~ Represents the upper bound of the running time of an algorithm. It represents the max. time an algorithm will take to complete for any given input size.

② Omega Notations (Ω)

It represents the lower bound of a function. It is used to describe the best case scenario for any algorithm.

③ Theta Notation (Θ)

It represents both the upper & lower bounds of a function. It is used to describe the tight bound of an algorithm. It provides an exact bound on the growth rate of the function. It tells the exact time.

(2)→

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

1, 2, 4, 8, 16, 32 ..... K terms

K_{th} term $= a r^{K-1}$ $\{r = 4/2 = 2\}$ $a = 1$

Teacher's Signature.....

Date
Page

k^{th} term $= 1 \cdot 2^{k-1}$

$n = 2^{k-1}$

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

$2n = 2^k$

Taking log both sides

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

$\log 2^n = k$

$k = \log_2 2 + \log_2 n$

$k = 1 + \log_2 n$

Complexity $\Rightarrow O(1 + \log_2 n)$ ignoring const.

$\Rightarrow O(\log_2 n)$

Ans 3 $\rightarrow$ $T(n) = 3T(n-1)$ ——— ①

$T(0) = 1$ ——— ②

Putting $n = n-1$ in eq ①

$T(n-1) = 3T(n-2)$ ——— ③

Putting $n = n-2$ in eq ①

$T(n-2) = 3T(n-3)$ ——— ④

Putting $n = n-3$ in eq ①

$T(n-3) = 3T(n-4)$ ——— ⑤

Putting ⑤ in ④

$T(n-2) = 9T(n-4)$ ——— ⑥

Putting 6 in ③

$T(n-1) = 3^3 T(n-4)$ ——— 7

for k terms

$T(n) = 3^k T(n-k)$ ——— ⑧

$n - k = 0$

$n = k$

Teacher's Signature

Date
Page

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

$T(n) = 3^n$

Complexity $\Rightarrow O(3^n)$

Ans 4 → $T(n) = 2T(n-1) - 1$ ——— ①

$T(0) = 1$ ——— ②

Put $n = n-1$ in ①

$T(n-1) = 2T(n-2)$ ——— ③

Put $n-2$ in ①

$T(n-2) = 2T(n-3)$ ——— ④

Put $n = n-3$ in ①

$T(n-3) = 2T(n-4)$ ——— ⑤

Put ⑤ in ④

$T(n-2) = 2 \cdot 2T(n-4) - 2 - 1$ ——— ⑥

Put ⑥ in ③

$T(n-1) = 2 \cdot 2 \cdot 2T(n-4) - 2 \cdot 2 - 2 - 1$ ——— ⑧

for k terms

$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} - 2^{k-3} \ldots\ldots - 2^0$

Assume $n-k = 0$

$\Rightarrow 2^k T(0) - 2^{k-1} - 2^{k-2} - 2^{k-3} \ldots\ldots 2^0$

$\Rightarrow 2^k - 2^{k-1} - 2^{k-2} \ldots\ldots 2^0$

$\Rightarrow 2^k - (2^0 + 2^1 + \ldots\ldots + 2^{k-3} + 2^{k-2} + 2^{k-1})$

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

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

$\Rightarrow 1$

Ans → $O(1)$

Teacher's Signature.....

Ans 5

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

i = 1, 2, 3, 4, 5, 6 . . . .

S = 1, 3, 6, 10, 15 . . . . .

loop will run till $S <= n$

Sum for k^{th} terms $= \frac{k * (k+1)}{2}$

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

$2n = k^2 + k$

$k(1+k) = 2n$

$k + k^2 = 2n$

$k^2 = n$

$k = \sqrt{n}$

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

Ans 6

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

$\Rightarrow 1, 2^2, 3^2, 4^2 \ldots\ldots k^{th}$

k^{th} term $= k * k$

$k^{th} <= n$

$k * k <= n$

Teacher's Signature.....

Date
Page

$k^2 = n$

$k = \sqrt{n}$

Complexity $\Rightarrow O(\sqrt{n})$

7 →
```
void function (int n) {
    int i, j, k;
    for (i = n/2; i <= n; i++) {
        for (j = 1; j <= n; j = j * 2) {
            for (k = 1; k <= n; k = k + 2) {
                cout++;
            }
        }
    }
}
```

Time complexity of inner most loop.

$k = 1$ to n, $k = k * 2$

1, 2, 4, 8, 16 ..... k^{th} term.

k^{th} term $= 2^{k-1}$

$n = \frac{2^k}{2} \Rightarrow 2n = 2^k = 1$

Taking $\log_2$ both sides

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

$\log_2 2n = k$

$k = \log_2 2 + \log_2 n$

$\boxed{k = 1 + \log_2 n}$

* Complexity of middle loop

$j = 1$; 1 to n; $j = j \times 2$

1, 2, 4, 8, 16

$= (1 + \log_2 n)$

Teacher's Signature.....

Date
Page

It means for each value i, this loop runs $(1+\log_2 n)$ times
Complexity of outermost loop.

$i = n/2$ to n, $i++$
$n/2, n/2+1, n/2+2, n/2+3 \ldots\ldots$ kth term
k^{th} term $= n/2 + k$
$n = n/2 + k$, $\Rightarrow k = n - n/2$

Till loop will run $n/2$ times.

Total complexity $\Rightarrow n/2 + (1+\log_2 n) * (1+\log_2 n)$
$\Rightarrow n/2 + n/2 \log_2 n + n/2 (\log_2 n) + n/2 \log_2 n$

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

8 →
```
function(int n)
{
  if(n==1)
    return;
  for(i to n){
    for(j to n){
      printf("*");
    }}}
```

i	j
1	1 → n
2	1 → n
3	1 → n

. n*n times

Teacher's Signature.....

Date
Page

for function(n-3)

$n, n-3, n-6, n-9 \ldots\ldots$ k^{th} term

$n, n-1.3, n-2.3, n-3.3 \ldots\ldots$ k^{th} term.

k^{th} term $= n-(k-1).3$

$1 = n-3k-3$

$\Rightarrow n-3k-3-1=0$

$n-3k-4=0 \quad \Rightarrow k = \frac{n-4}{3}$

Inner most loop will work $= n*n*\frac{n-4}{3}$

$= \frac{n^3-4n^2}{3}$

Complexity $= O(n^3)$ ignore constants

Ans 9 →

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

outer loop will n times

for i=1, j will run n times

for i=2, j will run n/2 times

for i=3, j will run n/3 times

Inner loop will run $\Rightarrow (n + \frac{n}{2} + \frac{n}{3} \ldots\ldots - \frac{n}{n})$...

$\Rightarrow n.\log n$

sum of $(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \ldots\ldots + \frac{1}{n})$ is $\log n$

complexity $\Rightarrow O(n.\log n)$

Teacher's Signature.....

Date
Page

Ans 10 $n^k = O(c^n)$ as n approaches infinity n^k is bounded above.