Date / /
Page No.

Name :- PULKIT BANSAL
R. No. :- 30
Univ. Roll No. :- 2014789

DAA ASSIGNMENT - 01 (TCS 505)

Ans ① Asymptotic notation are the mathematical notation used to describe the running time of an algorithm when the input ~~tre~~ tends towards a particular value or a limiting value, Asymptotic Notation is a way to compare function that ignores constant factors and small input sizes.

Types of Asymptotic Notations :-

① Big Theta (Θ) → Tight bound, complexity respresented is like average value or range within which the actual time of execution will be.

② Big Oh (O) → This is used for upper bound of algorithm or worst case of an algorithm. It tells that a function will never exceed specified time for any value of input n.

③ Big Omega (Ω) → Used to define lower bound of any algorithm or the best case of a algorithm.

Ans ② $\sum_{i=1}^{n}$ step n

1, 2, 4, 8, ... n (k terms)

→ $2^0, 2^1, 2^2, 2^3$ .... n

taking log.

$\log(L^{(K-1)}) = \log n$

$(K-1) = \log n$

$K = \log n + 1$

$= O(\log n)$

Ans ③ $T(n) = \begin{cases} 3T(n-1) & \text{if } n>0 \\ 1 & \text{otherwise} \end{cases}$

$T(n) = 3T(n-1) \quad \forall\ n>0$ — ①

put $n = (n-1)$

$T(n-1) = 3T(n-2)$ — ②

putting ② in ①

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

$= 3^2\,T(n-2)$ — ③

putting $n = n-2$ in eq ①

$T(n-2) = 3T(n-3)$ — ④

$T(n) = 3^3\,T(n-3)$ — ⑤

$T(n) = 3^K\,T(n-K)$ — ⑥

Base Case ⇒ $T(0) = 1$

$n - K = 0$

$n = K$

$T(n) = 3^n\,T(n-n)$

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

$= 3^n$

Date / /

Page No.

Ans (4) $T(n) = \begin{cases} 2T(n-1) - 1 & \text{if } n>0 \\ 1 & \text{otherwise} \end{cases}$

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

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

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

$\Rightarrow 2^2 (2T(n-3) - 1) - 2 - 1$

$\Rightarrow 2^3 T(n-3) - 2^2 - 2^1 - 2^0$

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

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

$\Rightarrow 2^n - 2^n + 1 = 1$

$T(n) = 1$

Ans (5)

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

1, 3, 6, 10 .... n terms

$S = 1 + 3 + 6 + 10 + \ldots K$

$0 = 1 + 2 + 3 + 4 \ldots K$

$K = \frac{(K-1)}{2}$

$n \simeq K^2$

$K = \sqrt{n}$

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

Ans (6) $1, (2)^2, (3)^2, (4)^2 \ldots n$

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

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

Ans (7) $\sum_{i=n/2}^{n} \sum_{J=1}^{n (J*2)} \sum_{K=1}^{n}$

$\sum_{i=n/2}^{n} \sum_{J=1}^{n} \log(n)$
step $J*2$

$\sum_{i=n/2}^{n} (\log(n))^2$

$\Rightarrow \left(\frac{n}{2}+1\right)(\log(n))^2 \Rightarrow T(n) = O(n(\log n)^2)$

Ans (8) $\sum_{i=1}^{n} \sum_{J=1}^{n} \Rightarrow \sum_{i=1}^{n} n = n^2$

$T(n) = T(n-3) + n^2$ — (1)

Putting $n = n-3$ in eq (1)

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

$T(n_2) = T(n-a) + (n-b)^2$

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

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

$T(1) = 0$

$n - 3K = 1$

$K = \frac{n-1}{3}$

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

$\Rightarrow T(n) = n^3$ (4)

Ans (9) $\displaystyle\sum_{i=1}^{n} \sum_{0=1,\ \text{step}=i}^{n} 1$

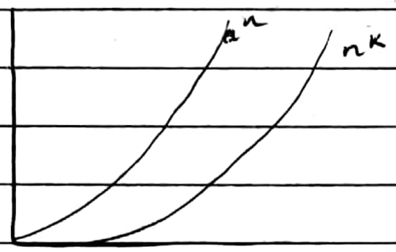
$$\sum_{i=1}^{n}\left(\frac{n-1}{i}+1\right) \Rightarrow (n-1)\sum_{i=1}^{n}\frac{1}{i}+\sum_{i=1}^{n}1$$

$$\Rightarrow (n-1)\sum_{i=1}^{n}\frac{1}{i}+n$$

$$(n-1)\log n+n$$

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

Ans (10)



$n^k = o(a^n)$

$n^k \leq a^n \cdot c \ \forall \ c>0$ and $n \geq n_0$

let $n = n_0$

$n_0^{\,k} \leq c \cdot a^{n_0}$

$n_0^{\,3} \leq c \cdot 3^{n_0}$ $\quad\quad k = c = 3$ (say)

$\Rightarrow c \geq 1$ & $n_0 \geq 1$

[redacted]

Ans (11)

J	i
1	0
2	1
3	3
4	6
5	10
6	15
7	21

$S = 0, 1, 3, 6, 10, 15 \ldots$ — (1)

$S = 0, 1, 3, 6, 10 \ldots$ — (2)

$0 = 0, 1, 2, 3, 4, 5 \ldots k - n$

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

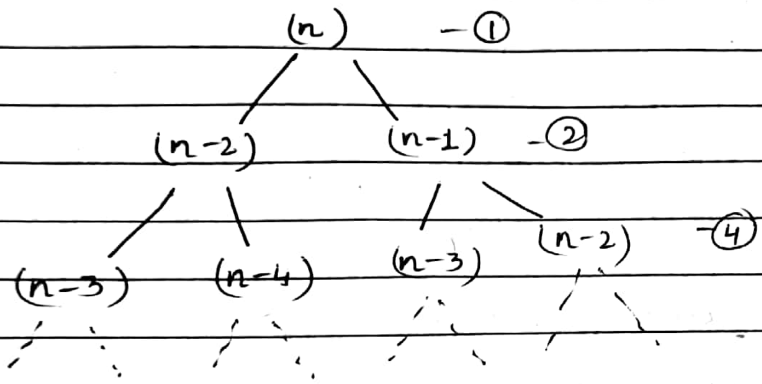
$n \simeq k^2$

$k = \sqrt{n}$

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

Ans (12) $0, 1, 1, 2, 3 \ldots n.$

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



$T = 1 + 2 + 4 + 8 \ldots 2^n$

$a = 1$

$r = 2$

$$T = \frac{1(2^{n+1} - 1)}{2 - 1}$$

$= 2^{n+1} - 1$

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

Space complexity $= O(n)$
bcz max. stack frame is
same as longest node.

Ans (13) i)

```
for(int i=0; i<=n; i++)
{
    for(int j=1; j<=n; j*=2)
        print(" ");
```

ii)

```
int func(int n)
{
    if(n<=2)
        return 1;
    else
        return(fun(floor(sqrt(n))+n);
}
```

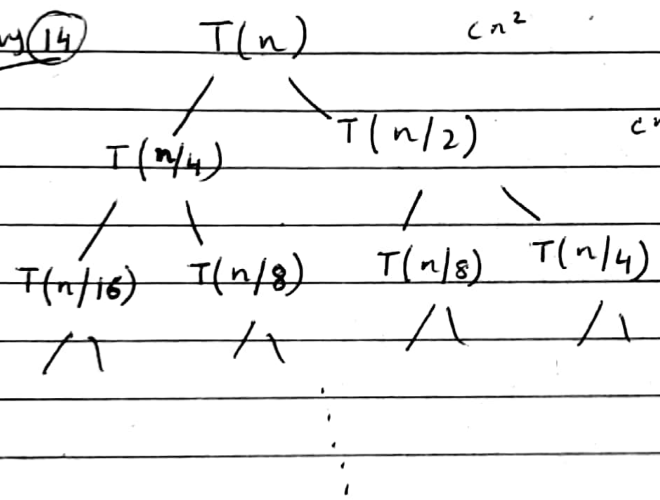
$\log(\log n)$

iii)

```
for(int i=0; i<n; i++)
    for(int j=0; j<n; j++)
        for(int k=0; k<n; k++)
            print("*");
```

n^3

Ans (14)



cn^2

$cn^2\left(\frac{3n}{2}\right)$

$c\left(\frac{3}{4}\right)^3 n^2$

$cn^2\left(\frac{3}{4}\right)^k n$

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

$n = 2^k$

$k = \log n$

$$T(n) = cn^2\left[1 + \frac{3}{4} + \left(\frac{3}{4}\right)^2 + \cdots \left(\frac{3}{4}\right)^{\log n}\right]$$

$cn^2(1)$

$= n^2$

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

Ans (15) $T(n) = \sum_{i=1}^{n} \sum_{J=1}^{n-1} (1)$

$\Rightarrow \sum_{i=1}^{n} \frac{n-1}{i}$

$\Rightarrow (n-1)\sum_{i=1}^{n} \frac{1}{i} \Rightarrow (n-1)\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots n\right)$

$= (n-1)\log n$

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

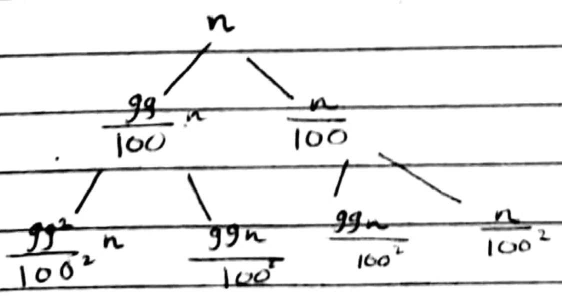
Ans (16) $i = 2, 2^k \cdot 2^{k^2}\ 2^{k^3} \cdots\ 2^{k^k}$

$\log(n) = k^2 \log 2$

$\log_k(\log(n)) = k \log k$

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

Ans (17) $T(n) = T\left(\frac{99}{100}n\right) + T\left(\frac{n}{100}\right)$



If we take longer branch, i.e, $\frac{99n}{100}$

$T_c = \log_{\frac{100}{99}} n \simeq \log n$

$\left(\frac{99}{100}\right)^k n = 1$

$n = \left(\frac{100}{99}\right)^k$ $\qquad K = \log_{\frac{100}{99}} n$

$K = \log_{\frac{100}{99}} n$

$$T(n) = \frac{n\left(\log_{\frac{100}{99}} n\right)}{100}$$

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

Ans (18) a) $100 < \log \log < \log n < \sqrt{n} < n < n \log n < n^2 < 2^n < 2^{2^n} < 4^n < n!$

b) $1 < \log\log(n) < \sqrt{\log(n)} < \log(n) < 2n < 4n < 2(2^n) < \log(2n) < 2\log(n) < n < n\log n = \log(n!) < N!$

c) $ab < \log 8^n < n \cdot \log_6 n = n\log_2 n < 5n < 8n^2 < 7n^3 < 9^{2n}$

Ans (19)

```
for( i=0 to k-1)
  {
    if (ar[i] = key)
      {
        return i;
      }
  }
return -1;
```

Ans (20) Iterative Insertion Sort

insertion sort (arr, n)

loop from i = 1 to i = n-1

pick close arr[i] and insert it into sorted seq.

arr [a . . i-1]

recursive insertion sort

insertion Sort (arr, n)

{

if n <= 1

return

recursively sort n-1 element

insertion Sort (arr, n-1)

Pick 1st element arr [i] and insert

it into sorted arr [0 . . i-1]

}

Insertion sort considers one input element per iteration and produces a partial solution without considering future elements. It is also called online sorting algorithm.

Ans (21)

1	Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
2	Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
3	Merge Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
4	Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$
5	Quick Sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
6	Heap Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

Ans (22)

Algorithm	Inplace	Stable	Online Sorting
Bubble Sort	✓	✓	x
Selection Sort	✓	x	x
Insertion Sort	✓	✓	✓
Merge Sort	x	✓	x
Quick Sort	x	x	x
Heap Sort	✓	x	x

Ans (23)

```
int binarysearch(int arr[], int l, int n)
{
    while (k=r)
    {
        int m=(l+r)/2;
        if (arr[m]= n)
            return m;
        else if (arr[m] <n)
            l=m+1;
        else
            r=m-1;
    }
        return -1;
}
```

Recursive Binary Search

```
int BinarySearch(int arr[], int c, int r, int x)
{
    if (L>r)
        return-1
    int m=(l+r)/2;
    if (arr[m] =n)
        return m;
```

else if (arr[m] < n)
return Binary Search(arr, m+1, r, n);
else
return Binary Search (arr, l, m-1, n)
}

Iterative Binary Search
Time complexity ⇒ Best = O(1) | Avg = O(log n), worst = O(log n)
Space ⇒ O(1)

Recursive binary ⇒
Time complexity ⇒ Best = O(1) Average = O(log n) worst = O(log n)

Space complexity ⇒ Best = O(1), Avg = O(log n) worst = O(log n)

Ans (4) $T(n) = T(n/2) + 1 = T(n) = O(\log n)$

* * * * *