Page No.
Date : / /20

Name → Akshit Mahar
Un. Roll No. → 1961008
Sem → 5
Sec → A
Course → BTech (CS)

Q-1)

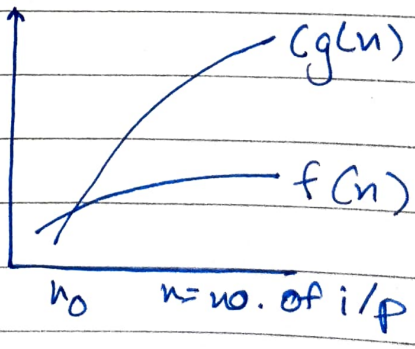
Ans- Asymptotic Notations are the mathematical notations used to describe the running time of an algo when the I/P tends towards a particular value or a limiting value.

→ There are mainly 3 Asymptotic notations:-

(i) Big O Notation

- It represents the upper Bound of running time of an algo.
- This notation is called ac upper bound of the algo, or a work case of an algo.
- $O(g(n)) = \{f(n)$: there exist positive constants c & no such that $0 \leq f(n) \leq g(n)$ for all $n \geq no$, where $c > 0$ & $n \geq no$.
- eg.

$f(n) = 3 \log n + 100$
$g(n) = \log n$
$3 \log n + 100 \leq (c * \log(n))$
$c = 1$ & $n > 2$
(undefined at $n = 1$)



(ii) Big Omega (Ω) Notation

- It represents the lower bound of the running time of an algo.

Teacher's Signature .....

- This notation is known as lower bound of an algo, or best case of an algo.
- $\Omega(g(n)) = \{f(n)$: there exist positive constant C & n_0 such that $0 \leq C g(n) \leq f(n)\ \forall\ n, n \geq n_0$
- e.g.

$f(n) = 3n+2$

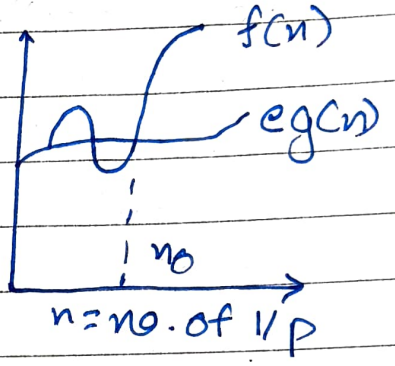
$Cg(n) \leq f(n)$

[c = constant, $g(n) = n$]

$Cn \leq 3n+2$

$Cn - 3n \leq 2$

$n(C-3) \leq 2 \Rightarrow n \leq \frac{2}{C-3}$



if we assume $C=4$, then $n_0 = 2$.

$\boxed{C = 4,\ n_0 = 2}$

(iii) Theta (Θ) notation

- It enclose the fun" from above & below. Since, it represents the upper & lower bound of running time of an algo
- $\Theta(g(n)) = \{f(n)$: there exist positive constant C_1 & C_2 & n_4 such that $0 \leq C_1 * g(n) \leq f(n) \leq C_2 * g(n)\ \forall\ n > n_0$
- e.g.

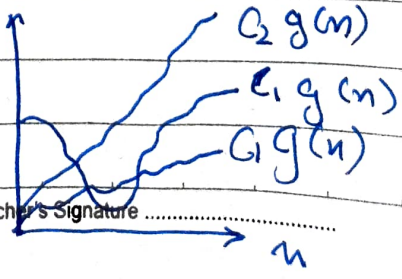
$f(n) = 5n^3 + 16n^2 + 3n + 8$

$5n^3 \leq n^3 + 16n^2 + 3n + 8 \leq (5+16+3+8)n^3$

$5n^3 \leq f(n) \leq 32n^3$

$\boxed{C_1 = 5,\ C_2 = 32,\ n_0 = 1}$

$f(n) \leftrightarrow \Theta(n^3)$



Teacher's Signature .....

Page No.
Date : / /20

Q 2)

Ans) $i = 2, 4, 8, 16, \cdots\cdots K^{th}$ term $\cdots\cdots n$

$G_n = a_1 r^{n-1}$

$G_n = 1(2)^{K-1}$

$n = 2^{K-1}$

$\log_2 n = (K-1)\log_2 2$

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

$O(n) = \log n$

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

Ans)

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

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

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

$T(n) = 3 \times 3 \times 3T(n-3)$

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

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

$T(n) = 3^3 \times 3T(n-4)$

$T(n) = 3^4 \times T(n-4)$

⋮

General form :-

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

$T(n-i) = T(0)$

$n - i = 0$

$\boxed{n = i}$

Putting $n = i$ in eq^n (i);

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

$T(n) = 3^n T(0)$ $\quad [T(0) = 1 \text{ given}]$

$T(n) = 3^n$

$\boxed{T(n) = O(3^n)}$

Teacher's Signature .....

Page No.
Date : / /20

Q4)

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

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

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

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

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

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

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

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

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

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

⋮

General form:-

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

$T(n-i) = T(0)$

$n - i = 0$

$\boxed{n = i}$

$T(n) = 2^n T(0) - (1 + 2 + 2^2 + 2^3 + \cdots 2^{n-1})$

$[T(0) = 1]$

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

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

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

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

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

$\boxed{T(n) = O(2^n)}$

Teacher's Signature .....

Page No.
Date : / /20

Qs→

No. of steps (K)	S	i
0	0	1
1	1	2
2	3	3
3	6	4
4	10	5
5	15	6
6	21	7
⋮	⋮	⋮
K step	n	

$T(n) = O(K)$

$S = 0, 1, 3, 6, 10, \cdots\cdots n$

$S_n = 1 + 3 + 6 + 10 + 15 + \cdots\cdots + n$

$S_n = 1 + 3 + 6 + 10 + \cdots\cdots + (n-1) + n$

$0 = 1 + 2 + 3 + 4 + 5 + \cdots\cdots - n$

$n = 1 + 2 + 3 + 4 + \cdots\cdots$ K step

$n = \frac{K}{2}[2(1) + (K-1)1]$

$2n = K[2 + K - 1]$

$2n = K^2 + K \Rightarrow 2n = \left(K + \frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2$

$2n + \left(\frac{1}{2}\right)^2 = \left(K + \frac{1}{2}\right)^2$

Teacher's Signature .....

$K + 1/2 = \sqrt{2n + (1/2)^2}$

$K = \sqrt{2n + (1/2)^2} - 1/2$

$T(n) = T(K)$

$T(n) = T\left(\sqrt{2n + (1/2)^2} - 1/2\right)$

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

Q6) Since, i is moving from 1 to $\sqrt{n}$ with
Ans) linear growth

So

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

Q7)

Ans) $O(n \log n \log n)$

$$\boxed{O(n(\log n)^2)}$$

Q 8)

Ans) $T(n) = T(n-1) + n^2$ $[T(n-1) = T(n-2) + (n-1)^2]$

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

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

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

⋮

General Term:-

$T(n) = T(n-i) + n^2 + (n-1)^2 + (n-2)^2 + \cdots (n-i)^2$

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

$n = i + 1 \Rightarrow \boxed{n-1 = i}$

Teacher's Signature .....

Page No.
Date : / /20

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

$T(n) = T(1) = n^2 + (n-1)^2 + (n-2)^2 + \cdots 1^2$

$T(n) = 1 + 1^2 + 2^2 + 3^2 + \cdots + n^2$

$T(n) = \frac{n(n+1)(2n+1)}{6}$

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

Q9→

Ans→ $O(n\sqrt{n})$

Q10→

Ans→ If $c > 1$ then the exponential c^n for outgrows any term, so that answer is: n^k is $O(c^n)$

Q11→

Ans→ $i = 0, 1, 3, 6, 10, 15, \cdots$

$j = 1, 2, 3, 4, 5, 6 \cdots$

So, i will go on till n & general formula for K^{th} term is $n = \frac{K(K+1)}{2}$

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

Q12→

Ans→ $T(n) = T(n-1) + T(n-2) + c$

$T(n-2) \approx T(n-1)$

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

↳ $T(n-1) = 2T(n-2) + c$

Teacher's Signature .....

Page No.
Date : / /20

$T(n) = 2\,(2T(n-2) + c) + c$

$T(n) = 2^2 T(n-2) + 2c + c$

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

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

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

⋮

General Term :—

$T(n) = 2^i T(n-i) + (2^0 + 2^1 + 2^2 + \cdots + 2^{i-1})c$

$n - i = 0$

$\boxed{n = i}$

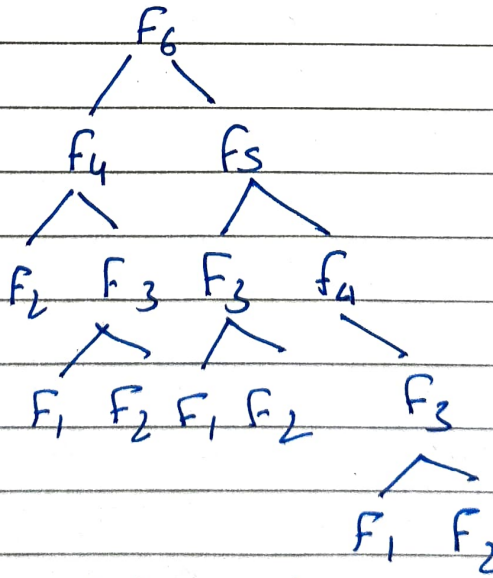
$T(n) = 2^n T(0) + (2^0 + 2^1 + 2^2 + \cdots + 2^{n-1})c$

$T(n) = 2^n (1) + 2^0 \frac{(2^{n-1})c}{2-1}$

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

$\boxed{T(n) = O(2^n)}$

fib (6)



The max depth is proportional to N, hence the space comp. of Fibonacci recursive is O(n)

Teacher's Signature .....

Page No.
Date : / /20

Q13→ (i) $n \log n$

```
void fun()
{
    int i, j;
    for (i=1; i<=n; i++)
    {
        for (j=0; j<=n; j=j*2){
            printf ("#");
        printf ("\n");}
    }
```

(ii) n^3

```
void fun (int n)
{
    int i, j, k;
    for (i=0; i<=n; i++)
    {
        for (j=0; j<=n; j++)
        {
            for (k=0; k<=n; k++)
                printf ("#");}
    }
}
```

(iii) $\log(\log(n))$

```
void fun (int n)
{
    bool primes [n+1];
```

Teacher's Signature .....

```
memset (prims, true, sizeof(prims));
for (int p=2; p*p<=n; p++)
{
    if (prime[p]==true)
    {
        for (int i = p*p; i<=n; i+=p)
            prime[i]= false;
    }
}
for (int p=2; p<=n; p++)
    if (prime[p])
        cout<<p<<endl;
}
```

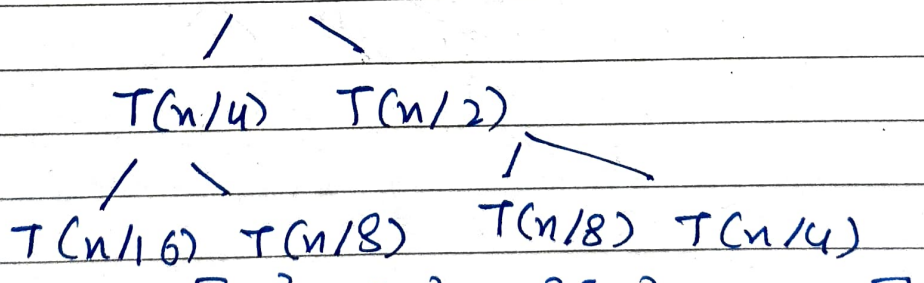
Q14)

Ans→ $T(1) = c$

$n = n/2$

$T(n/2) = T(n/8) + T(n/4) + c(n^2/4)$

$T(n) = T(n/4) + 2T(n/16) + c(n^2/16 + n^2/4 + n^2)$



$$T(n) = c\left[n^2 + \frac{5n^2}{16} + \frac{25n^2}{256} + \cdots\right]$$

$$T(n) = n^2 c\left[1 + \frac{5}{16} + \frac{5^2}{16^2} + \cdots\right]$$

$$\boxed{T(n) = O(n^2)}$$

Teacher's Signature .....

Q 15→

Ans→ for $i=1$, inner loop is executed n times
for $i=2$, inner loop is executed $n/2$ times
for $i=3$, inner loop is executed $n/3$ times
⋮
for $i=n$, inner loop is executed n/n times

Total time $= n + n/2 + n/3 + \cdots + n/2$
$= n + (1 + 1/2 + 1/3 + \cdots + 1/n)$
$= n \log n$

$\boxed{T(n) = O(n \log n)}$

Q 16→

Ans→ $\boxed{O(\log(\log n))}$

Q 18→

Ans→ a→ 100, $\log\log n$, $\log n$, root n, $n \log n$, n^2, 2^n, 2^{2n}, 4^n, $n!$

b→ 1, $\log(\log(n))$, $\sqrt{\log n}$, $\log n$, $\log(2n)$, $\log(n!)$, $2\log(n)$, n, $2n$, $4n$, $n\log(n)$, n^2, $2(2^n)$, $n!$

c→ 96, $\log_8 n$, $\log_2 n$, $\log(n!)$, $5n$, $n \log_6 n$, $n \log_2 n$, $8n^2$, $6n^3$, 8^{2n}, $n!$

Q 19→

Ans→ Linear Search (A, Key)
comp ← 0, f ← 0
for $i = 1$ to A.length

Teacher's Signature .....

```
    comp ← comp+1
if A[i] == key
    print "Element found"
f = 1
if  f == 0
    print "Element not found"
print comp
```

Q 20→

Ans→ Iterative Method of Insertion Sort –

```
InsertSort(A)
for j = 2 to A length
key = A[j]
i = j-1
while i > 0 & A[i] > key
    A[i+1] = A[i]
  i = i-1
A[i+1] = key
```

Recursive Method →

```
Insertion Sort(A, n)
if n ≤ 1
    return
Insertion Sort(A, n-1)
key = [n-1];
j = n-2
while j ≥ 0 and A[j] > key
```

A[j+1] = A[j]

j = j - 1

A[j+1] = Key

→ Insertion Sort considers one i/p element per iteration & produces a partial solution without considering future elements that's why it is called online sorting.

Other sorting algos that have been discussed in lecture are :—

- Bubble Sort
- Selection Sort
- Quick sort
- Merge Sort
- Heap Sort
- Counting Sort

Q2)

Ans →

	Best Case	Average Case	Worst case
Bubble Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$
Selection Sort	$\Omega(N^2)$	$\Theta(N^2)$	$O(N^2)$
Insertion Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$
Merge Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$
Heap Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$
Quick Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N^2)$
Counting Sort	$\Omega(N+K)$	$\Theta(N+K)$	$O(N+K)$

Teacher's Signature .....

Page No.
Date : / /20

Q 22)

Ans)

	In Place	Stable	Online
Bubble Sort	yes	yes	yes
Insertion Sort	yes	yes	yes
Selection Sort	yes	no	yes
Merge Sort	NO	yes	yes
Quick sort	yes	no	yes
Heap SORT	yes	no	yes
Count Sort	no	yes	yes

Q 23)

Ans) - Linear Search ⇒

```
Linear Search (A, Key)
found ← 0
for i=1 to N {
    if A[i] == key
        found ← 1
    printf "Element found"
    break }
if found == 0
    print "Element Not found"
```

Time complexity — O(n)

Space complexity — O(1)

- Binary Search (Iterative) ⇒

```
Binary Search (A, beg, end, key)
    while beg ≤ end
        mid = beg + (end - beg) / 2
```

Teacher's Signature .....

```
if mid == key
    return mid
if A[mid] < key
    beg = mid + 1
if A[mid] > key
    end = mid - 1
return -1
```

Time Complexity – $O(\log_2 n)$
Space Complexity – $O(1)$

Binary search (Recursive) →

```
Binary search (A, beg, end, key)
    if end > beg
        mid = (beg + end) / 2
    if A[mid] == item
        return mid + 1
    else if A[mid] < item
        return Binary Search (A, mid+1, end, key)
    else
        return Binary Search (A, beg, mid-1, end)
    return -1
```

Time Complexity – $O(\log n)$
Space Complexity – $O(1)$

Q24→
Ans → $T(n) = T(n/2) + C$