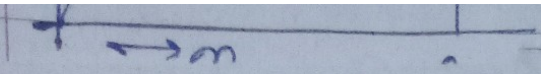


$\forall\ n > n_0$ and $c > 0$

ASSIGNMENT-1

NAME : SURBHI GUSAIN. SEMESTER : V^th

SEC - ML U. ROLL NO : 2014903

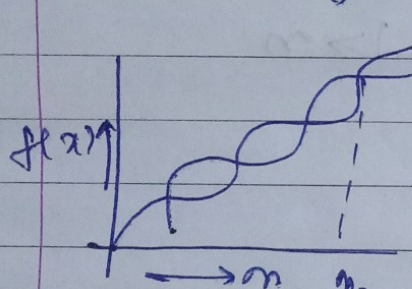
ROLL NO : 34

(Q1) The notations that are used to tell the complexity of an algo when input is very large is known as asymptotic notation.

The various types of asymptotic notation are :-

Big - O(n) :→ $f(n) = O(g(n))$

g(n) is tight upper bound.



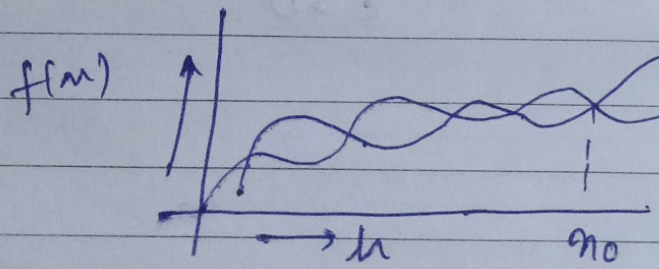
$f(n) = O(g(n))$

iff $f(n) <= c \cdot g(n)$

$\forall\ n >= n_0$ and $c > 0$

(ii) Big Omega (Ω) :- $f(n) = \Omega(g(n))$?

g(n) is tight lower bound of f(n)



$f(x) = \Omega(g(n))$

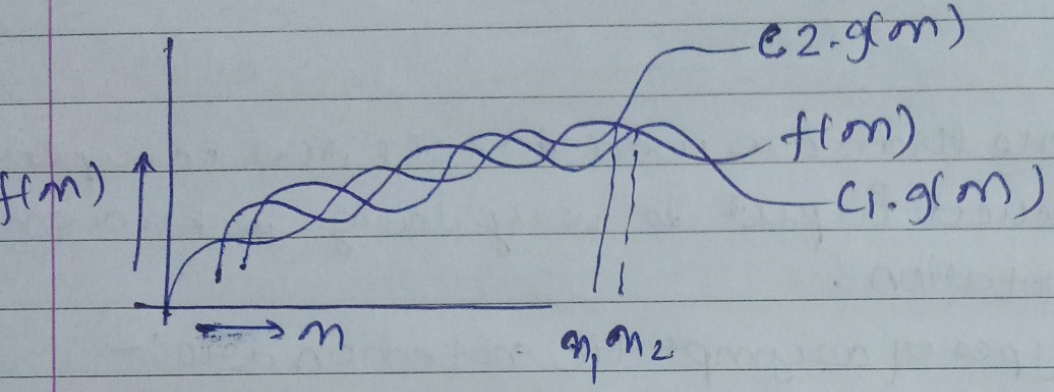
iff

$f(x) > c \cdot g(x)$

$\forall\ x >= n_0$ and $c > 0$

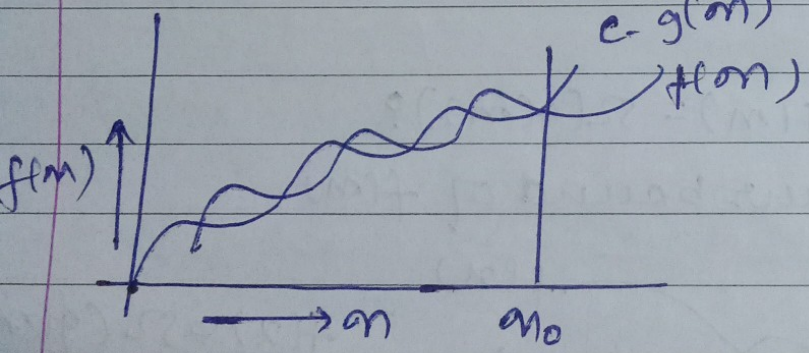
$\rightarrow n$ n_0

Theta (θ) :- it gives both lower & upper bound of $f(n) = \theta(g(n))$



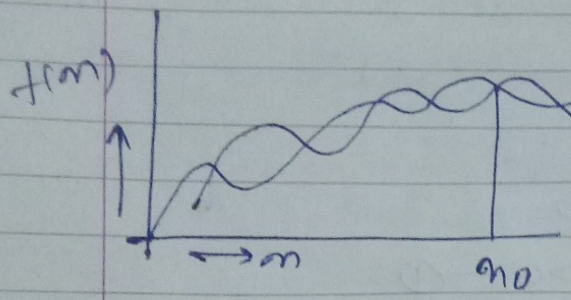
$\theta(g(n)) = f(n)$ iff $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$
$\forall c_1, c_2 > 0$.
$\max(n_1, n_2) \geq n$

Small - oh (o) :-
It gives upper bound of function



$f(n) = O(g(n))$
iff
$f(n) < c \cdot g(n) \; \forall \; n > n_0$ and
$c > 0$

(V) Small omega (ω) aka rho!-
It gives two lower bound of the function (algorithm)



$f(n) = \omega(g(n))$
iff
$f(n) > c \cdot g(n)$
$\forall\ n > n_0$ and $c > 0$

Ques2 for (i=1 to n)
$i = i * 2$ $\sum_{i=1}^{n}$ step 2 i

↳ 1, 2, 4 . . . , n (k terms)
↳ $2^{k-1} = n$
$k - 1 = \log(n)$
$k = \log(n) + 1$ complexity → $O(\log n)$

Ques3 $T(n) = \begin{cases} 3T(n-1) & n > 0 \\ \text{otherwise} & \end{cases}$

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

putting $n = n-1$ in eqn (1)
$T(n-1) = 3T(n-2)$ —(2)

putting eqn (2) in (1)
$T(n) = 3(3T(n-2)) \rightarrow 3^2 . T(n-2)$ —(3)

putting $n = n-2$ in eqn (1)
$T(n-2) = 3T(n-3)$ —(4)

Putting eq (4) in (5)
$T(n) = 3^3 T(n-3)$ —(5)
→ $T(n) = 3^k T(n-k)$ —(6)

Base case.
$T(0) = 1$
↳ $n - k = 0 \Rightarrow n = k$

putting $k=n$ in eqn (6)

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

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

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

(Ques4) $$T(n) = \begin{cases} 2T(n-1)-1 & n>0 \\ & n<=0 \end{cases}$$

$$T(n) = 2T(n-1) - 1 \quad \forall\ n>0 \quad \text{---(1)}$$

putting $n = n-1$ in eqn (1)

$$T(n-1) = 2T(n-2) - 1 \quad \text{---(2)}$$

putting eq (2) in (1)

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

$$2^2 T(n-2) + 1 + 2 \quad \text{---(3)}$$

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

$$T(n-2) = 2T(n-3) - 1 \quad \text{---(4)}$$

Putting eq (4) in eq (3)

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

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

$$\hookrightarrow T(n) = 2^k T(n-k) - (1 + 2 + 2^2 - \dots + 2^k) \quad \text{---(5)}$$

Base case $\rightarrow T(0) = 1$

$$n - k = 0$$

$$k = n$$

putting $k = n$ in eqn (5)

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

$$= 2^n - (1 + 2 + 2^2 + \dots + 2^n)$$

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

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

$$\hookrightarrow T(n) = O(1)$$

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

i	s
1	1
2	3
3	6
4	10

k = 4

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

↳ $S = 1+3+6+10+ \dots k$.

$S = +1 +3 +6 +10 \dots +k_{n-1} +k_n$

$0 = 1+2+3+4+ \dots k$

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

↳ $n \simeq k^2$ $\quad k = \sqrt{n}$

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

Ques6
```
void function(int n){
    int i, count=0;
    for(int i; i*i<=n; i++)
        count++;
}
```

~~i+2~~ 1, 2, 3 ... $\sqrt{n}$ $\quad T(n) = O(\sqrt{n})$

Ques7
```
void function(int n)
{   int i, j, k, count=0;
    for(int i=n/2; i<=n; i++)
        for(j=1; j<=n; j=j*2)
            for(k=1; k<=n; k=k*2)
                count++;
}
```

↳ $\sum_{i=n/2}^{n} \sum_{j=1}^{n(j*2)} \sum_{k=1}^{n} 1$

step = k*2

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

$(n/2+1)(\log(n))^2 \rightarrow T(n) = O(n \log n)$

(Ques 8)
```
function(int n){
    if(n==1) return;
    for(int i=1 to n)
        for(j=1 to n){ printf("*");
    }
    function(n-3);
}
```

$T(n) = T(n-3) + n^2$ — (1)

putting $n = n-3$ in eqn (1)

$T(n-3) = T(n-6) + (n-3)^2$ — (2)

Putting (2) in eqn (1)

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

putting $n = n-6$ in eqn (1)

$T(n-6) = T(n-9) + (n-6)^2$ — (4)

putting eqn (4) in (3)

$T(n) = T(n-9) + n^2 + (n-3)^2 + (n-6)^2$

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

$T(1) = 0$

$n - 3k = 1 \quad k = \frac{n-1}{3}$

$T(n) = n^2 + (n-3)^2 + \dots + (n-k)^2$

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

(Ques 9)
```
void function(int n){
    for(int i=1;i<=n;i++)
        for(j=1 to j<=n j=j+i)
            printf("*");
}
```

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \frac{1}{\text{step}=1}$$

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

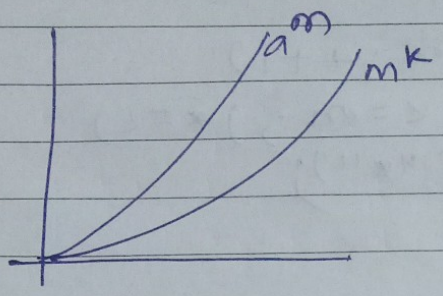
$$(n-1)\sum_{i=1}^{n} 1/i + \sum_{i=1}^{n} 1.$$

$(n-1)\sum_{i=1}^{n} 1/i + \sum_{i=1}^{n} 1.$

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

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

Ques 10



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

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

put $n = n_0$

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

$n_0^q \leq c \cdot 3^{n_0}$ $k = a = 3$ (say)

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

Ques 11

```
void fun(int n)
{
    int j=1, i=0;
    while (i<n) { j = j+j; j++; }
}
```

$s = 0, 1, 3, 6, 10, 15 \cdots n$ — (1)

$s = 0, 1, 3, 6, 10 \cdots + n$ — (2)

Sub (2) from (1)

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

$n = 0 + 1 + 2 \cdots k$

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

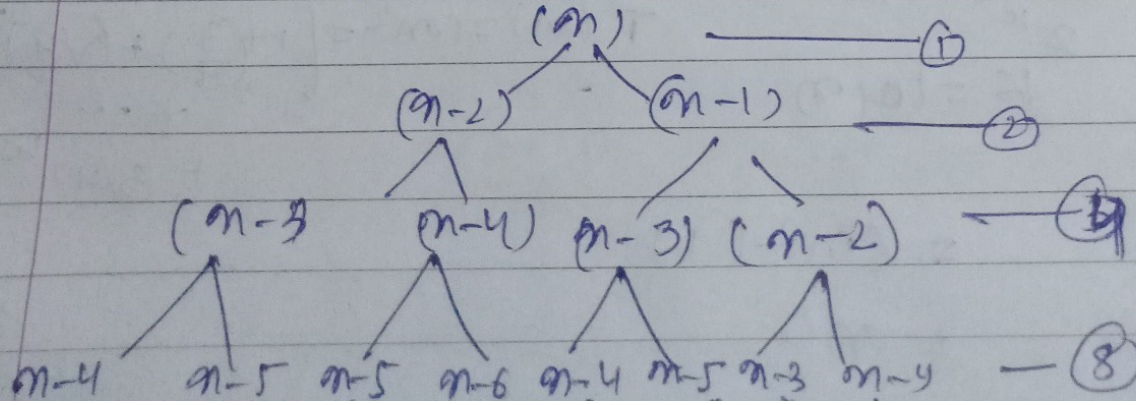
$n \simeq k^2$ $\quad k = \sqrt{n}$

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

Ques 12

$0, 1, 2, 3, \cdots T_n$

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



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

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

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

Ques (i)

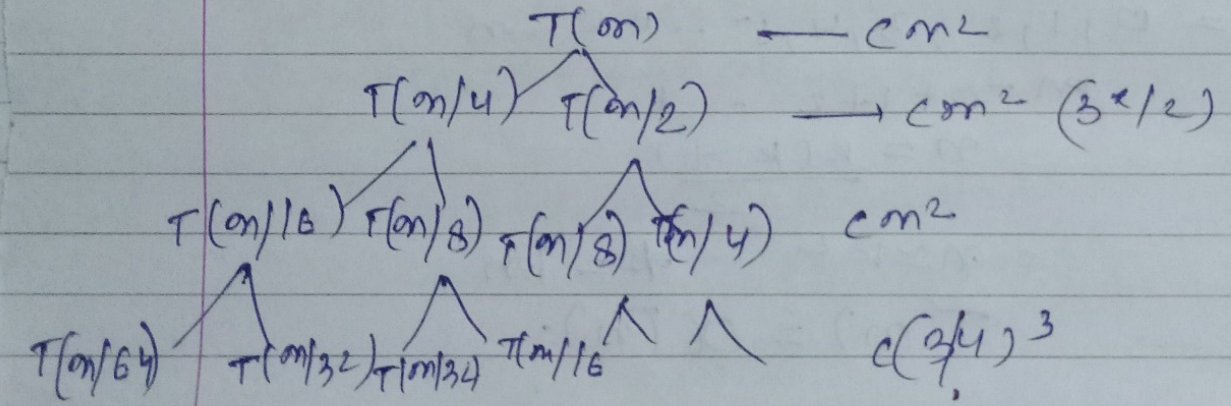
```
for(int i=0; i<=n ; ++i)
    for( j=1  to j<=n ; j*=2)
        printf("*");
```

$O(n \log n)$

(ii) $O(\log(\log(n)))$

(iii) $O(n^3)$

Ques $T(n) = T(n/4) + T(n/2) + cn^2$



$cn^2(3/4)^k n$

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

$K = \log n$

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

$= cn^2(1)$

$= n^2$

$= n^2$

$\hookrightarrow T(n) = O(n^2)$.

(Ques 5) $T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n-1} (1)$.

Step i

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

$= (n-1)(1 + 1/2 + 1/3 + 1/4 + \cdots n)$

$= (n-1) \log n$

$T(n) = n \log n$

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

Ques 6. $i = 2\ 2^k\ 2^{k^2}\ 2^{k^3} \cdots$

$2^{k^k} = n$

$\therefore k$ is constant.

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

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

$\log(n) = k^n$

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

~~$= n = \log \log n$~~

Ques 7 $T(n) = T\left(\frac{99}{100} n\right) + n/100$

$T(1) = 0$ — ①

putting $n = (99/100)n$ in eqn ①

$T(99/100\, n) = T\left((99/100)^2 n\right) + n/100$.

putting eq ② in ①

$T(n) = T\left(\left(\frac{99}{100}\right)^k n\right) + k^n/100$ — *

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

$$k = \log\left(\frac{100}{99}\right)^n \quad \text{(5)}$$

put $k = \log\frac{100}{99}\, n$ in eqn (4)

$$T(n) = n\underbrace{\left(\log\frac{100}{99}\right)^n}_{100}$$

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

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

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

(c) $96 < \log_2(n) = \log_8(n) < n\log_6(n) < n\log_2 = \log(n!) < 5n < 8n^2 < 7n < 8^2 n.$

Ques 19

```
INPUT array[N], key;
for(i=0 to n-1){
        if (arr(i) == key){
                return i;
        }
}
return -1.
```

Ques. Iterative Insertion Sort.

```
void insertion(int arr[], int n)
{
    int i, temp, j;
    for(i=1 to n-1){
        temp = arr[i]
        j = i-1;
        while(j>=0 and arr[j]>temp){
            arr(j+1) = arr(j);
            j = j-1;
        }
        arr(j+1) = temp;
    }
}
```

② Recursive Insertion sort

```
void insertionsort(int arr[], int n)
    if(n<2)
        return
    insertion(arr, n-1)
    last = arr(n-1), j = n-2;
    while(j>=0 arr[j]> temp{
            arr[j+1] = arr[j];
            j = j-1;
    }
    arr[j+1] = last;
}
```

Ans 21.

	Algorithm	Bestcase	Avgcase	Worstcase
(1)	BubbleSort	$O(n^2)$	$O(n^2)$	$O(n^4)$
(2)	Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
(3)	Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
(4)	Merge Sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$
(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.

	Algo	In-place	Stable	Online
(1)	Bubblesort	✓	✓	✗
(2)	Selection	✓	✗	✗
(3)	Insertion	✓	✓	✓
(4)	Merge	✗	✓	✗
(5)	Quick	✗	✗	✗
(6)	Heap	✓	✗	✗

(Ans 23) Iterative BS.

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

Recursive BS.

```
int BS(int arr[], int l, int r, int x).
{
    if (l<=l>r)
        return -1
    int m = (l+r)/2;
    if (arr[m]=x)
        return m,
    else if (arr[m]<x)
    return BS(arr, m+1, r, x);
    else
    return BS(arr, l, m-1, r).
```

Time complexity:

Space complexity

① Iterative BS.

Best case → O(1)
Avg. case → O(log m)
Worst case → O(log m).

O(1) (All cases)

② Recursive BS.

Best case → O(1)
Avg case → O(log m)
Worst case → O(log m)

Best case = O(1)
Worst case = O(log m)
Avg case = O(log m)