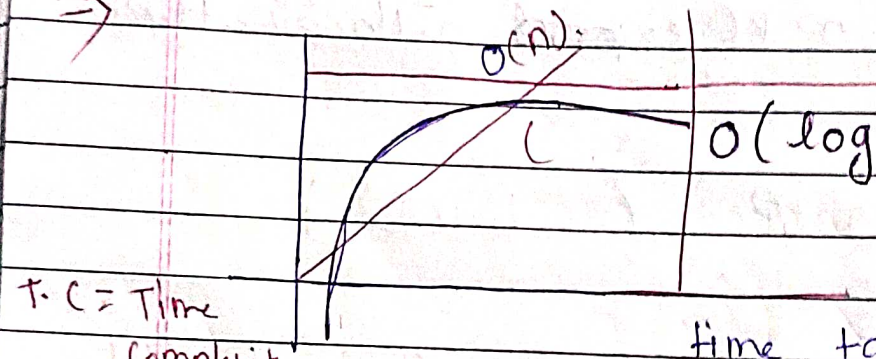
day 1st Time or Space Complexity (2 days)

1. Time Complexity $\underset{\text{not}}{!=}$ Time taken

⇒ • It is function that tell us how the time will grow as input is grow.

⇒



T.C = Time Complexity

time taken by linear Search is more than binary Search.

• $O(1) < O(\log N) < O(n)$

less T.C — best T.C

more than O(1) — average T.C

less more than O(logn) — worst T.C

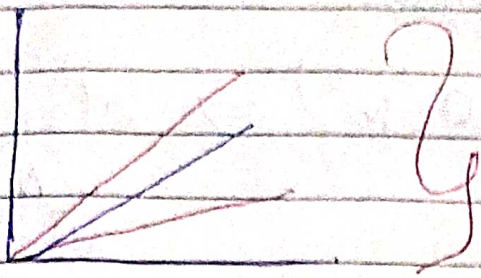
1° Alway look for worst Case Complexity

• (we have not worry about small data we have to worry about large data

Teacher's Signature.....

$O(1) < O(\log N)$
smaller larger

2. Always look at Complexity for larger data.

3. 

- Even those value of actual time is define they are all growing linearly.
- we dont worry about actual time
- we alway ignore all Constant.

$n = 2y$
$n = y$
$n = 3y$

we will ignor constant.

4. $O(3N^3 + 4N^2 + 5N + 6)$.

Alway ignor less dominating terms.

$O(\cancel{3}N^3 + \cancel{4}N^2 + \cancel{5}N + \cancel{6})$

alway ignor constant & less time

$O(N^3)$ equal to

$O(1) < O(\log N) < O(N) < O(2^n)$.

Teacher's Signature.....

⇒ Big - oh - notation (O)

upper bound. (worst case)

$O(N^3)$ → algo may be solve in N^2, N, log n but N^3 is jada me mati

⇒  omega - notation (Ω)

opposite of O (Big-oh-notation).

lower bound.

(best case) $\Omega(N^3)$ → atleast N^3 but isse kam nahi it may be N^4, N^5 more but not less

⇒ Theta Notation (Θ)

Average Case

both upper bound and lower bound

Teacher's Signature.....

Recurrence

⇒ Little O notation :

This is also giving upper bound but not strictly.

Big oh	Little O (strongest)
$f = O(g)$	$f = o(g)$
growth of O is $f \leq g$	$f < g$ strictly slower

⇒ Little Ω notation

lower bound but losely not strictly.

Big Ω	Little ω
$f = \Omega(g)$	$f = \omega(g)$
$f \geq g$	$f > g$.

2 type of recusions :

(1) Linear
$F(N) = F(N-1) + F(N-2)$

(2) Divide & Conquer.
$F(N) = F(\frac{N}{2}) + O(1)$

Recurrence relation.

Ex $f(n) = 3n^2 + 5n \longrightarrow O(n^2)$

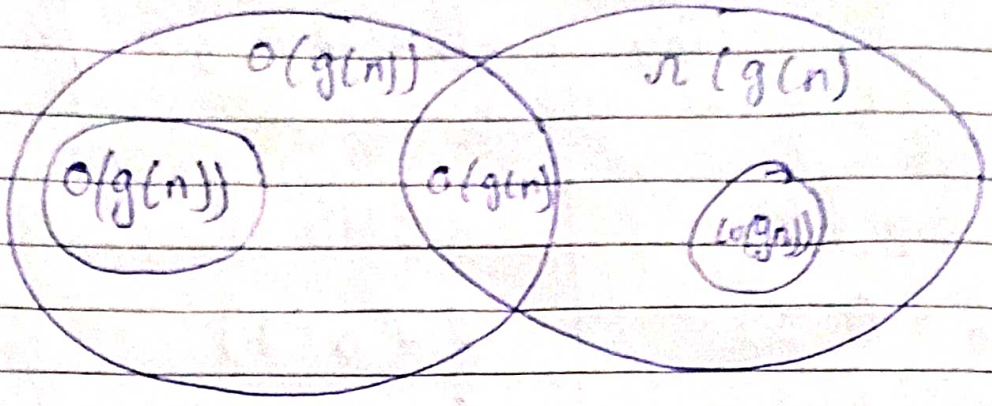
$f(n) = n + \log n \rightarrow O(n)$

$f(n) = 3n^3 + 45^5 \rightarrow O(n^5)$

$f(n) = 1000 \rightarrow O(1)$

Teacher's Signature.....

Venn diagram



Question.

Q1
```
#include <stdio.h>
void func1 (int array [], int length)
{
    int Sum = 0;
    int product = 1;
    for (int i = 0; i < length; i++)
    {
        Sum += array [i];
    }
    for (int i = 0; i < length; i++)
    {
        Product = array [i];
    }
}
int main
{
    int arr [] = {3, 5, 6};
    fun1 (arr, 3);
    return 0;
```

F_1 K_1

F_2 $K_2 n$

F_3 $K_3 n$

$O(length)$.

Teacher's Signature

$T_n = f + f_2 + f_3$

$= k_1 + k_2 n + k_3 n$

$= (k_2 + k_3) n$

$= k_4 n \rightarrow O(n)$

$O =$ length

$O(length)$.

Q2 void fun (int n)

{

int sum = 0;

int product = 1;

for (int i = 0; i < n; i++) { k_1 } n

{

for (int j = 0; j < i; j++) { k_2 } $0+1+2+\cdots+n$

{

print ("%d, %d\n", i, j); } k_2

}

}

$[n + n + n \cdots (n-1)n] k_2$

$n k_2 (1 + (1))$

$O(n^2)$.

Teacher's Signature

Q3
```
int function (int n)
{
    int i;
    if (n <= 0)
    {
        return 0;
    }
    else
    {
        i = radom(n-1);
        printf("the\n");
        return function(i) + function(n-1-i);
    }
}
```

O(random)
any value

Q4

(1) O(N+P), where P < N/9.

O(n).

(2) O(9N - N)

O(N)

Teacher's Signature

3 $O(N + 8 \log N)$

$O(\log N + N)$

$O(N)$

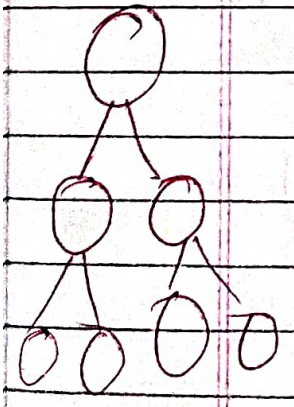
4 $O(N + M^2)$.

$O(N + M^2)$

Q5 Binary Search tree.

```
int Sum(Node node)                    k1 n
{
    if (node = null)                  k2 n
    {
        return 0;
    }
    return Sum(node.left) + node.value + Sum(node.right);   k3 n
}
```

B.T.T



$O(n)$

equal work for all

$k_1 n + k_2 n + k_3 n$

$O(n)$

Teacher's Signature.....

6 Prime or not?

```
int isPrime (int n) {
    if (n == 1) {
        return 0;
    }
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0)
            return 0;
    }
    return 1;
}
```

→ 1

→ K2

$O(\sqrt{n})$.

7

```
int isprime (int n) {
    for (int i = 2; i * i <= 10000; i++) {
        if (n % i == 0)
            return 0;
    }
    return 1;
}
```

constant

K,

$O(1)$

Teacher's Signature.....

nested loop

```
public static void main (String args[]){
    Scanner sc = new Scanner (System.
    int n = sc.nextInt();
    for (int i=0; i<n; i++){
        for (int j = 0; j<n; j++){
            System.out.print ("hello");
```

nested loop.

$i = 0 \rightarrow j = 0, 1, 2, \ldots n$.

$i = n \rightarrow j = 0 \quad n$.

$n \times n$. $O(n^2)$.

Q when input is m and n both

then $O(m \times n)$.

when

```
int n = sc.nextInt();
int m = sc.nextInt();
```

Q when for loop is not nested

then $O(n)$.

Teacher's Signature

```
public Static void main( String arg[]) {
    Scanner sc = new Scanner (System.in);

    int n = sc.nextInt();
    int m = sc.nextInt();

        for(int i=0;i<n ;i++) {
            System.out.print ("hello");
        }

    for ( int j=0; j<m; j++) {
        System.out.print( "hello");
    }
}
```

i=0 to n

j=0 to m

n * m O (n × m).

⇒ Space Complexity.

How much space it take in memory.

Arrays.

It takes more space offset memory size.

Teacher's Signature.....