Intersection of 2 Arrays

Constraint
↓
Each array contains distinct elements

(N) Arr1: { 30, 10, 45, 20, 65, 15, 70 }

(M) Arr2: { 40, 45, 10, 17, 23, 33, 65, 20 }

{ 10, 45, 20, 65 }
↓
⇒ Answer order does not order

Brute force ⇒ 2 Nested loops

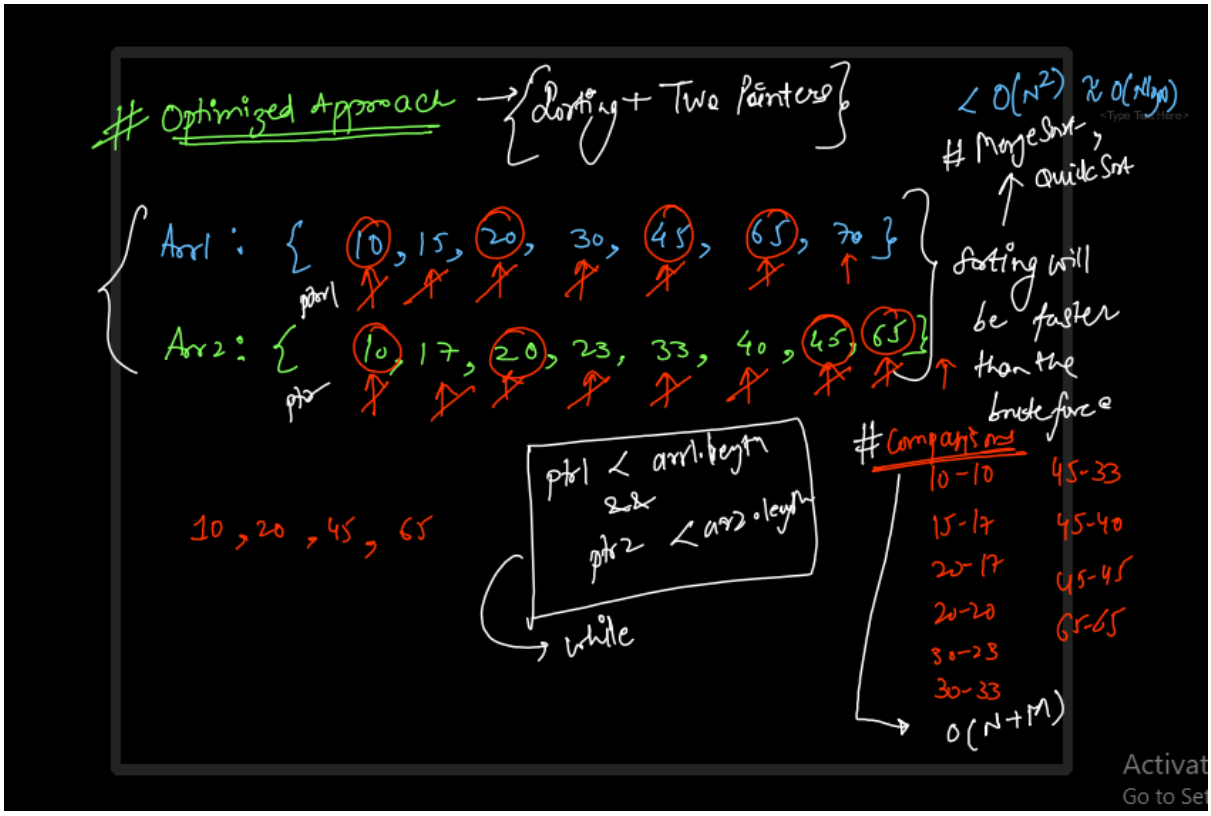
↳ outer loop :→ first array → (N)

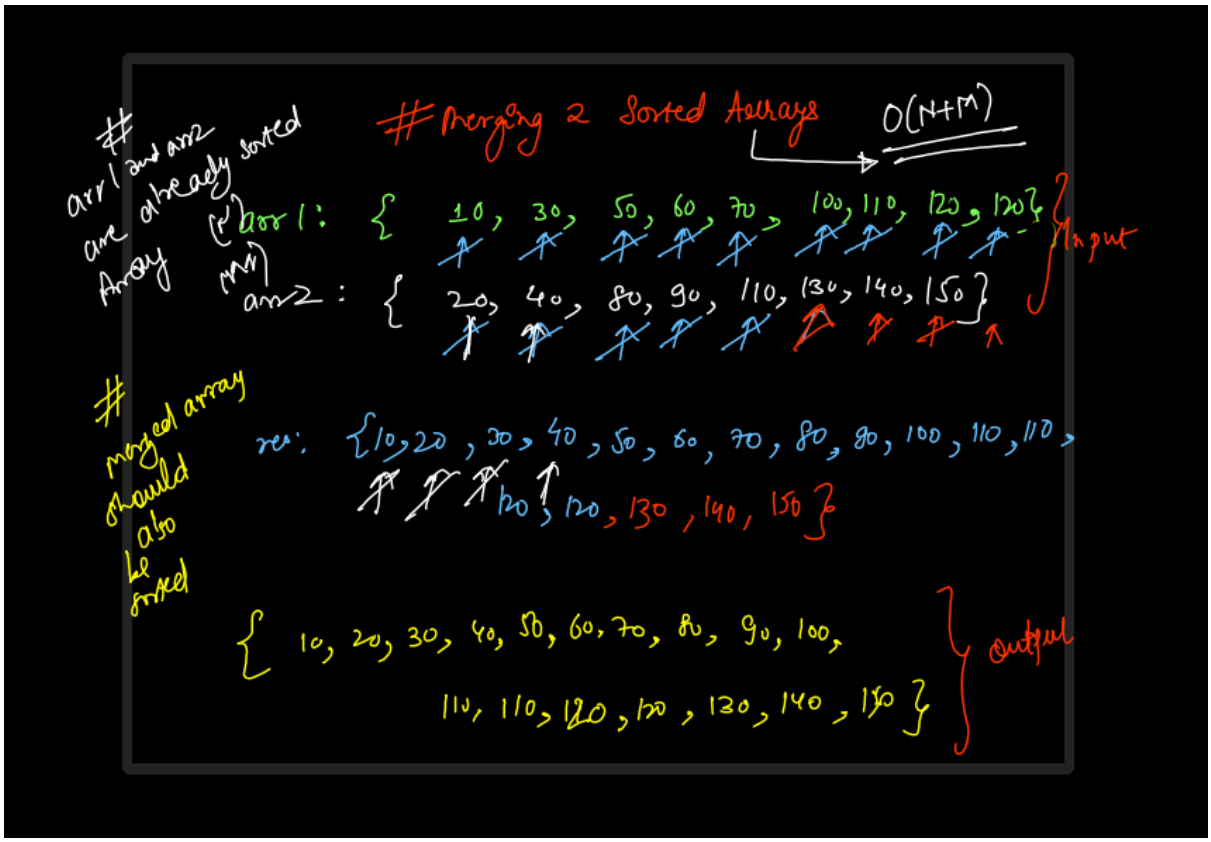
inner loop : → Second array → (M)

```
if (arr1[i] == arr2[j])
```

↑ ↑
Intersection found

Comparisons → O(N*M)


Optimized Approach → {Sorting + Two Pointers}
< O(N^2) ≥ O(NlogN)
MergeSort, QuickSort
Arr1 : { 10, 15, 20, 30, 45, 65, 70 }
Arr2 : { 10, 17, 20, 23, 33, 40, 45, 65 }
ptr1
ptr2
Sorting will be faster than the brute force
10, 20, 45, 65
ptr1 < arr1.length && ptr2 < arr2.length
while
Comparisons
10-10
15-17
20-17
20-20
30-23
30-33
45-33
45-40
45-45
65-65
O(N+M)


# Merging 2 Sorted Arrays
O(N+M)
# arr1 and arr2 are already sorted Array (ascending)
arr1 : { 10, 30, 50, 60, 70, 100, 110, 120, 130 }
arr2 : { 20, 40, 80, 90, 110, 130, 140, 150 }
Input
merged array should also be sorted
res : {10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 110, 120, 120, 130, 140, 150}
{ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 110, 120, 130, 140, 150 }
Output

Array Rotation { left }

R=0, 6, 12, 18, 24 ← 6*m+0
{10, 20, 30, 40, 50, 60}

R=1, 7, 13 ← 6*m+1
{ 20, 30, 40, 50, 60, 10 }

R=2, 8, 14 ← 6*m+2
{ 30, 40, 50, 60, 10, 20 }

R=3, 9, 15 ← 6*m+3
{ 40, 50, 60, 10, 20, 30 }

R=4, 10, 16 ← 6*m+4
{ 50, 60, 10, 20, 30, 40 }

R=5, 11, 17 ← 6m+5
{ 60, 10, 20, 30, 40, 50 }

Smaller Rotation for any R will be R%size

```
int ptr1 = 0, ptr2 = 0, ptr3 = 0;
int[] res = new int[arr1.length + arr2.length];

while(ptr1 < arr1.length && ptr2 < arr2.length){
    if(arr1[ptr1] <= arr2[ptr2]){
        res[ptr3] = arr1[ptr1];
        ptr1++; ptr3++;
    }
    else {
        res[ptr3] = arr2[ptr2];
        ptr2++; ptr3++;
    }
}
```

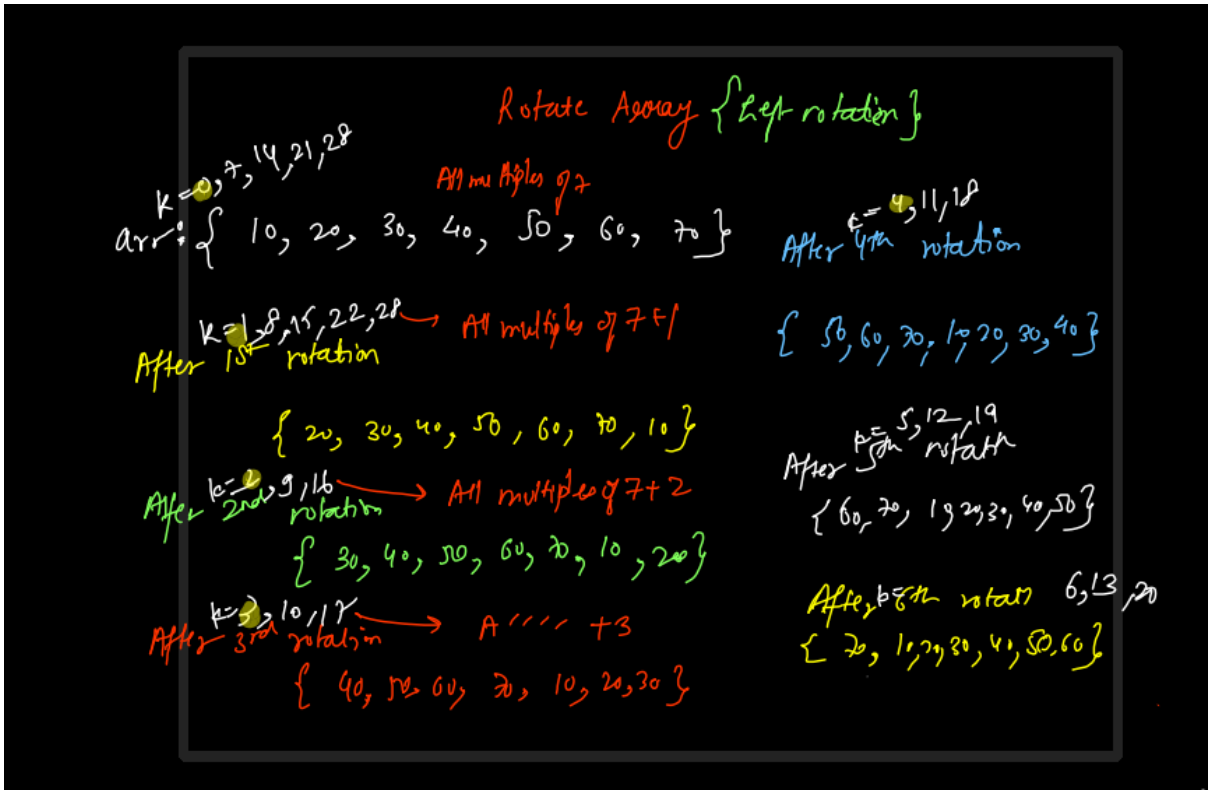
③ arr1: { 2, 3, 5 }

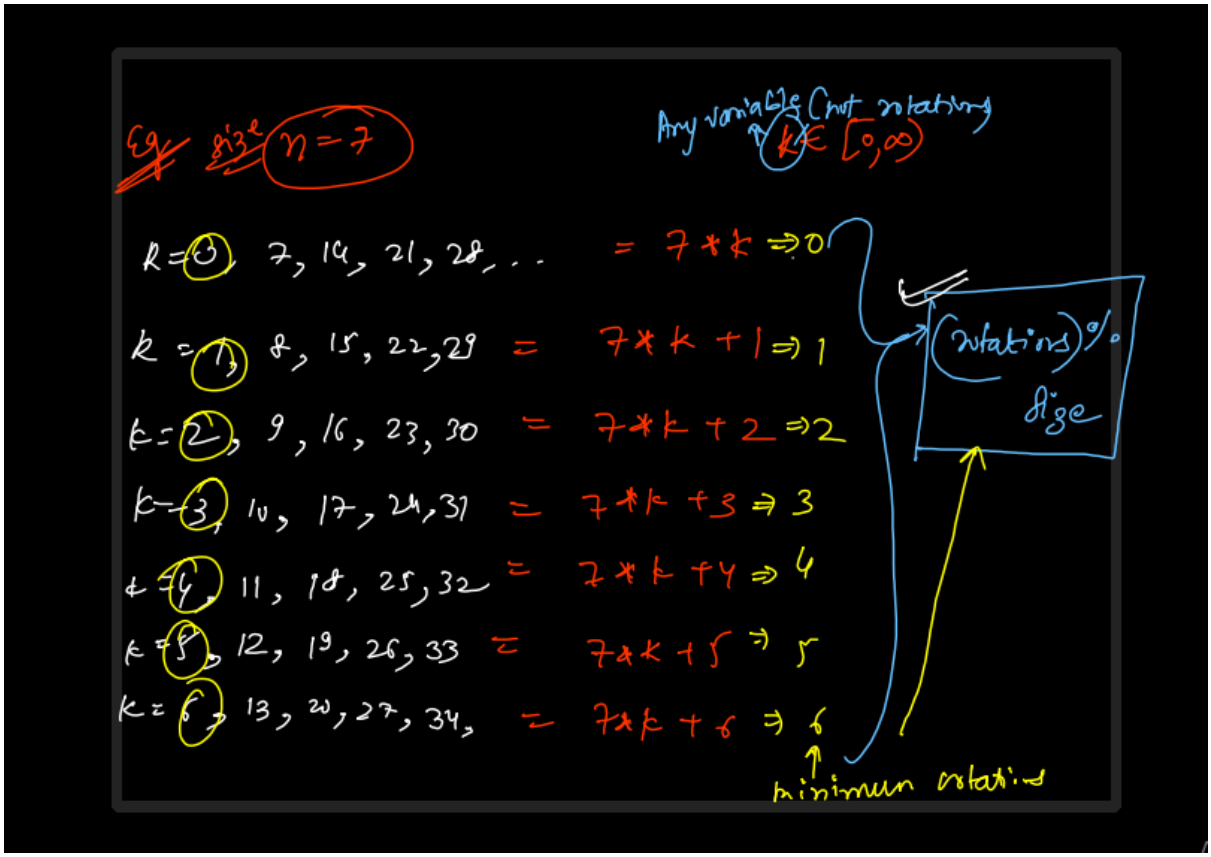
④ arr2: { 1, 4, 5, 8 }

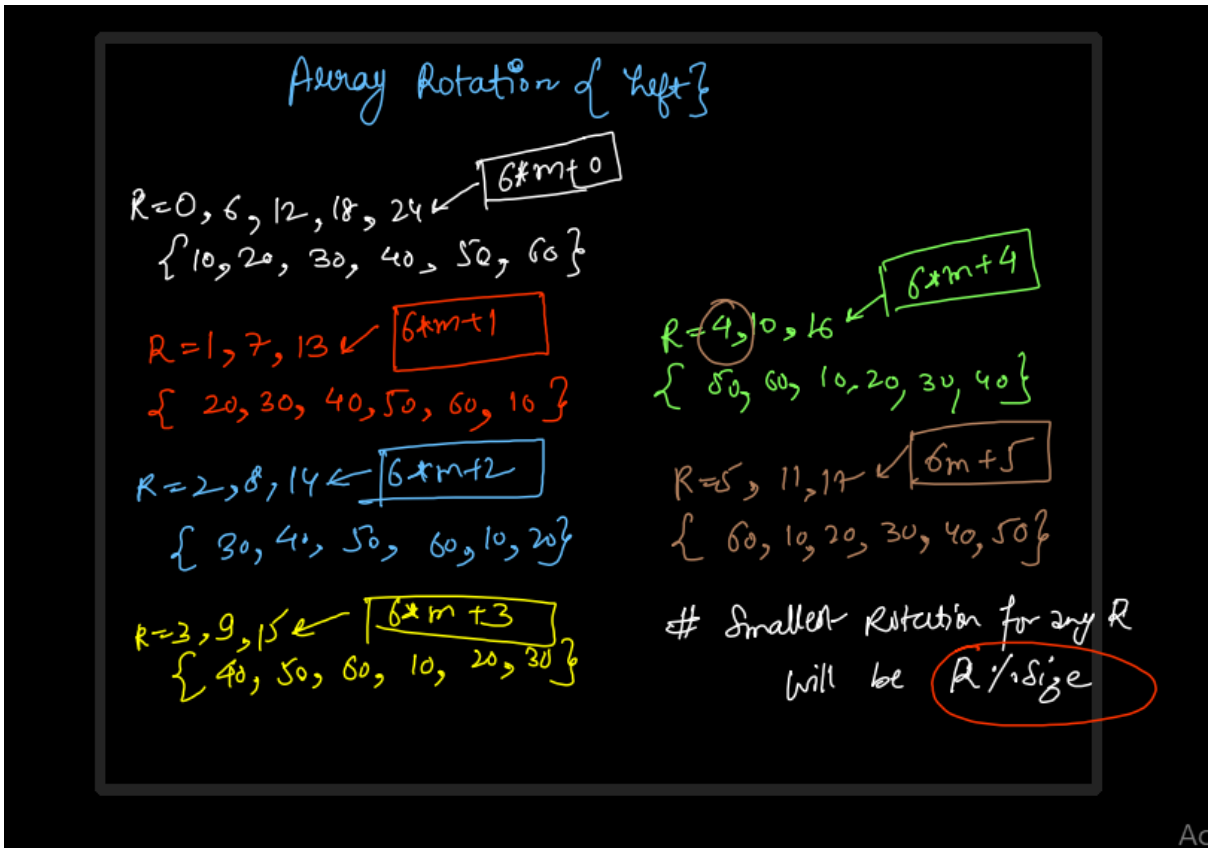
res: { 1 2 3 4 5 5 8 }

```
while(ptr1 < arr1.length){
      res[ptr3] = arr1[ptr1];
      ptr1++; ptr3++;
}

while(ptr2 < arr2.length){
     res[ptr3] = arr2[ptr2];
     ptr2++; ptr3++;
}
```

9:15


Rotate Array {Left rotation}
K = 0, 7, 14, 21, 28
All multiples of 7
arr: { 10, 20, 30, 40, 50, 60, 70 }
After 1st rotation
K = 1, 8, 15, 22, 28
All multiples of 7+1
{ 20, 30, 40, 50, 60, 70, 10 }
After 2nd rotation
K = 2, 9, 16
All multiples of 7+2
{ 30, 40, 50, 60, 70, 10, 20 }
After 3rd rotation
K = 3, 10, 17
A.P. +3
{ 40, 50, 60, 70, 10, 20, 30 }
After 4th rotation
K = 4, 11, 18
{ 50, 60, 70, 10, 20, 30, 40 }
After 5th rotation
K = 5, 12, 19
{ 60, 70, 10, 20, 30, 40, 50 }
After 6th rotation
6, 13, 20
{ 70, 10, 20, 30, 40, 50, 60 }


Eg size n = 7
Any variable (not rotating) k ∈ [0,∞)
R=0, 7, 14, 21, 28, ... = 7*k ⇒0
k=1, 8, 15, 22, 29 = 7*k+1 ⇒1
k=2, 9, 16, 23, 30 = 7*k+2 ⇒2
k=3, 10, 17, 24, 31 = 7*k+3 ⇒3
k=4, 11, 18, 25, 32 = 7*k+4 ⇒4
k=5, 12, 19, 26, 33 = 7*k+5 ⇒5
k=6, 13, 20, 27, 34, = 7*k+6 ⇒6
(rotations)% size
minimum rotation


Array Rotation { left }
R=0, 6, 12, 18, 24 ← 6*m+0
{10, 20, 30, 40, 50, 60}
R=1, 7, 13 ← 6*m+1
{ 20, 30, 40, 50, 60, 10 }
R=2, 8, 14 ← 6*m+2
{ 30, 40, 50, 60, 10, 20 }
R=3, 9, 15 ← 6*m+3
{ 40, 50, 60, 10, 20, 30 }
R=4, 10, 16 ← 6*m+4
{ 50, 60, 10, 20, 30, 40 }
R=5, 11, 17 ← 6m+5
{ 60, 10, 20, 30, 40, 50 }
Smaller Rotation for any R will be R%size

(1) Brute force ⟹ By taking another array of same size

Taking another array
- → Reduce the value of k by k%size
- → First skipping the starting k elements
- → Store the remaining elements
- → Store the starting k elements

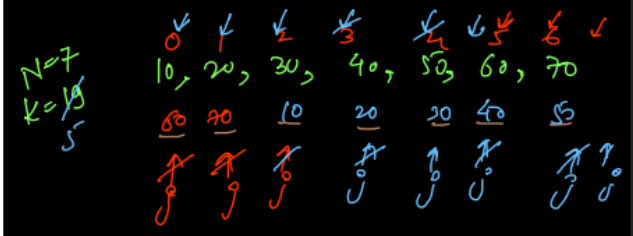
(2) Rotating 1 one by one

(3) Reversal algorithm

(1)
(2)
(3)
(4)

```
k = k % n; // Reduce the va

int[] output = new int[n];
int j = 0;

for(int i=k; i<n; i++){
    output[j] = input[i];
    j++;
}

for(int i=0; i<k; i++){
    output[j] = input[i];
    j++;
}
```

N=7
K=19
5



In-place

② Rotating the Array 1 place at a time

performing the algorithm on the input array

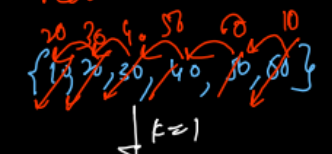
call this fn k time

Rotateby1()

k = 16 ⇒ 16 % 6 = 4

k%n

reduced value

eg 

k=1

{20, 30, 40, 50, 60, 10}

Rotateby1 fn

① Reduce the value of k

② Store 0th index value in temp

③ Shift all indexes to left

④ Place temp at the last index

```
public static void rotateby1(int[] arr){
    int temp = arr[0];
    int n = arr.length;

    for(int i=1; i<n; i++){
        arr[i - 1] = arr[i];
    }
    arr[n - 1] = temp;
}
```

```
k = k % n;
```

```
for(int i=0; i<k; i++){
    rotateby1(input);
}

for(int i=0; i<n; i++){
    System.out.print(input[i] + " ");
}
```

{10, 20, 30, 40, 50}

↓ rotate by 1()

{20, 30, 40, 50, 10}

↓ rotateby1()

{30, 40, 50, 10, 20}

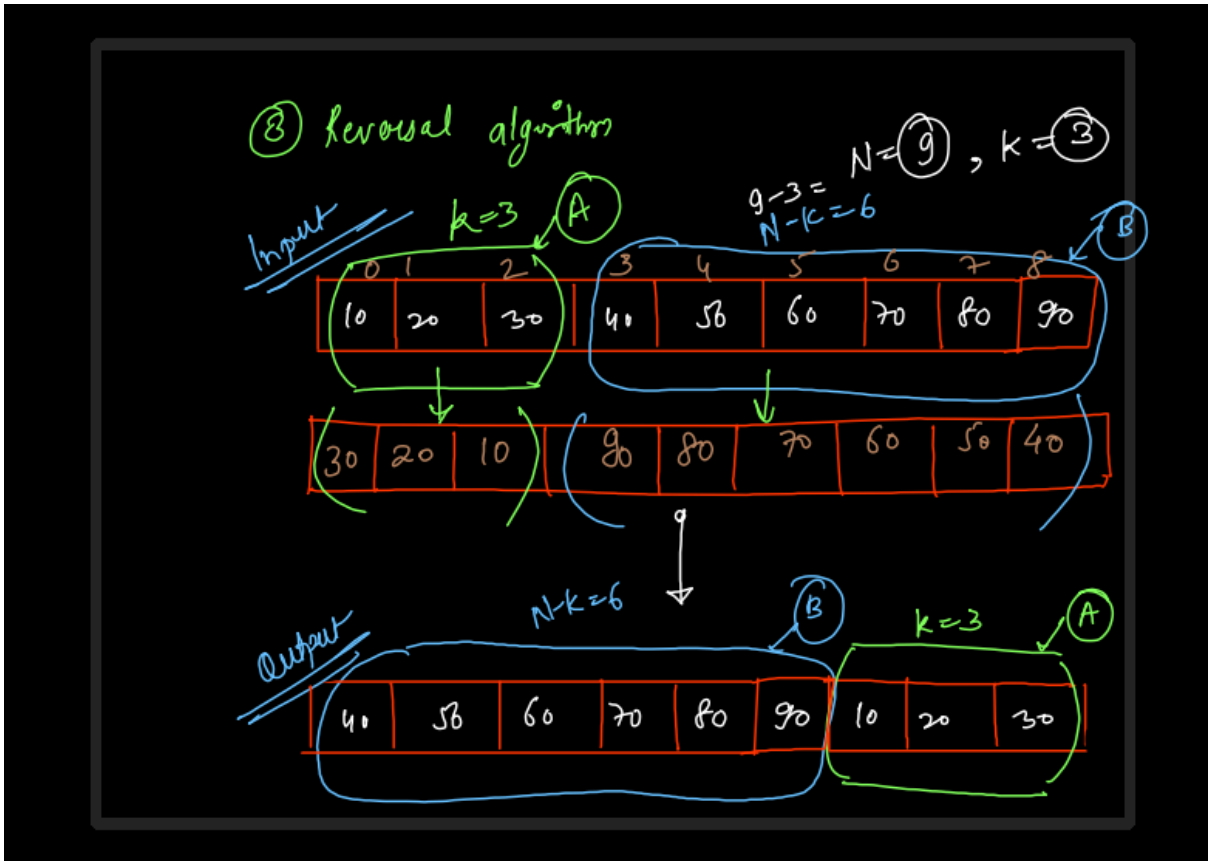
↓ rotateby1()

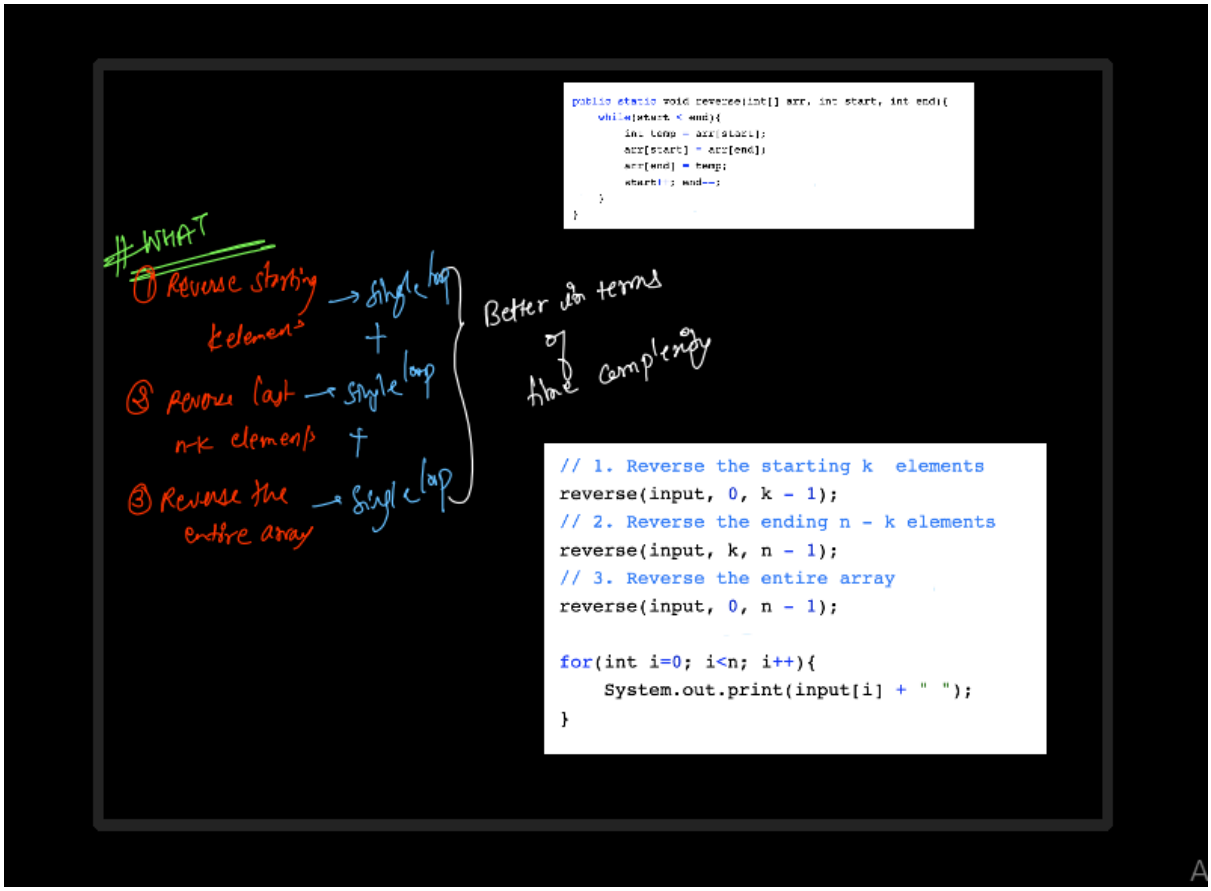
{40, 50, 10, 20, 30}

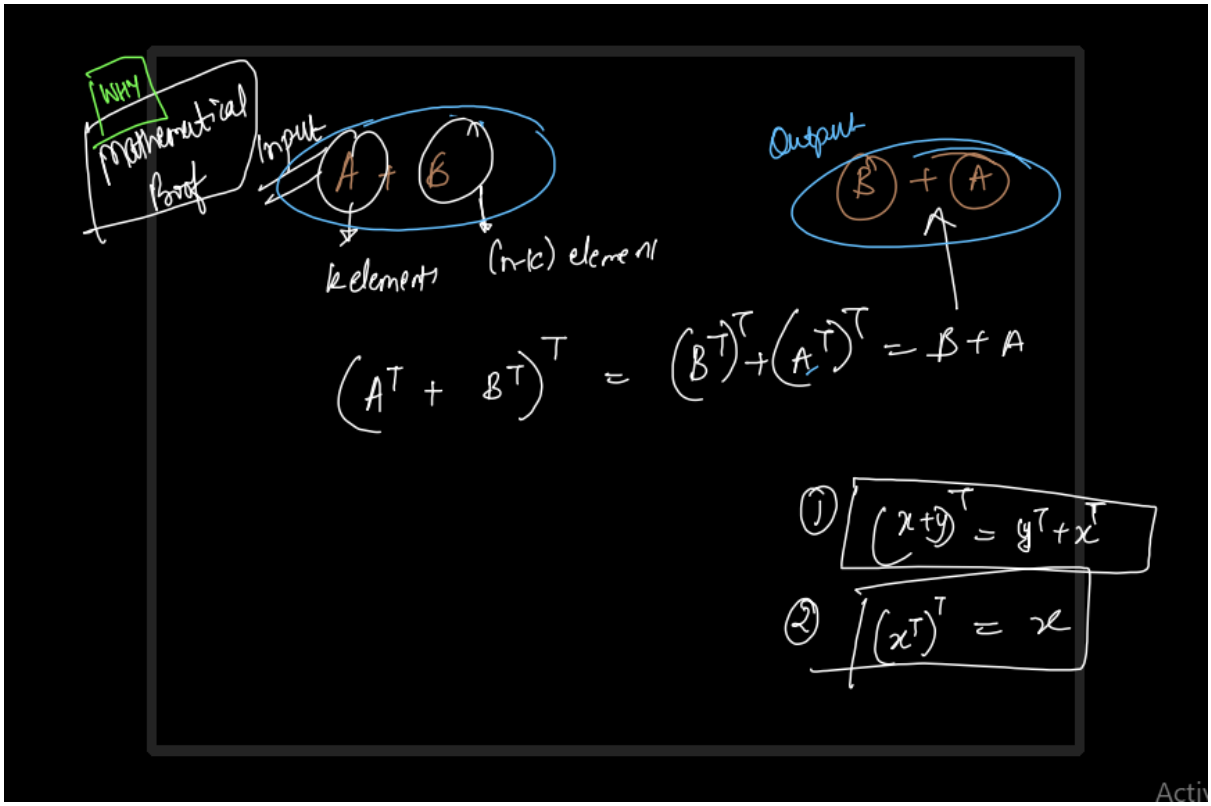
rotations

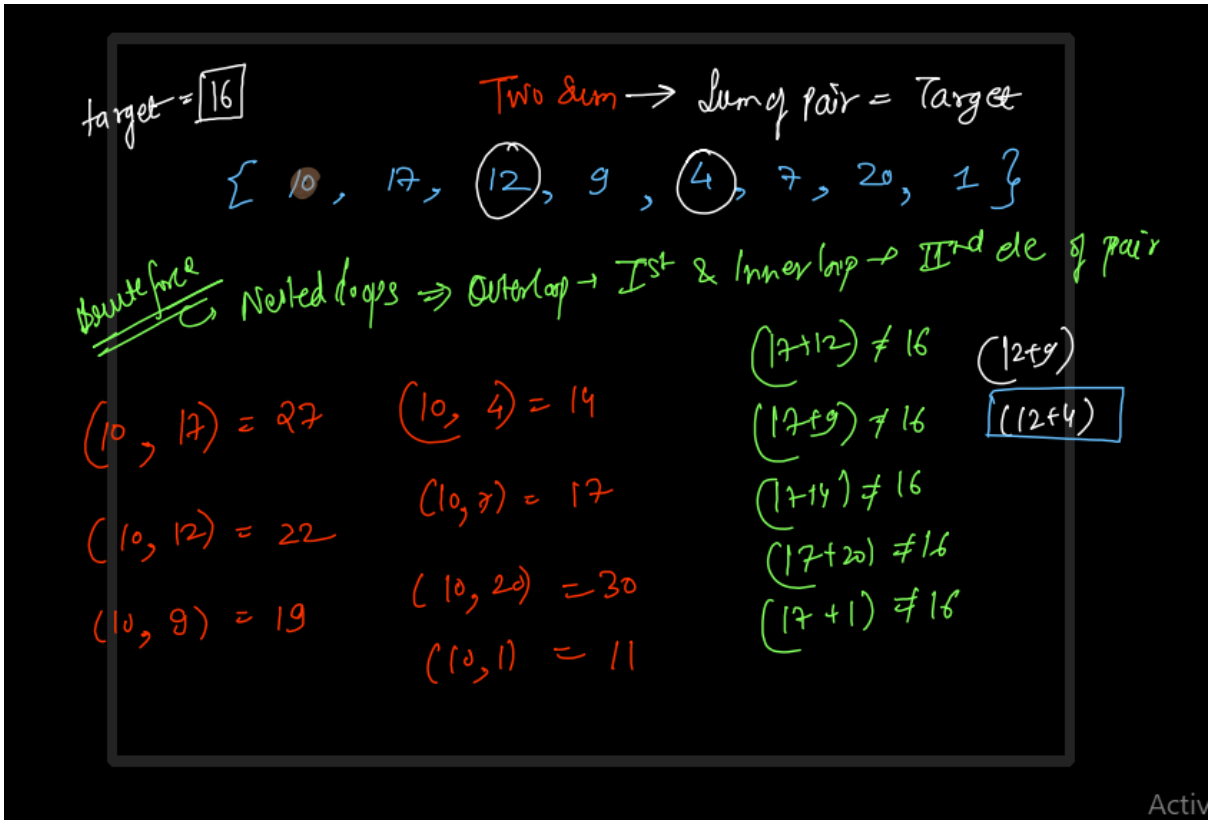
k = 3%5

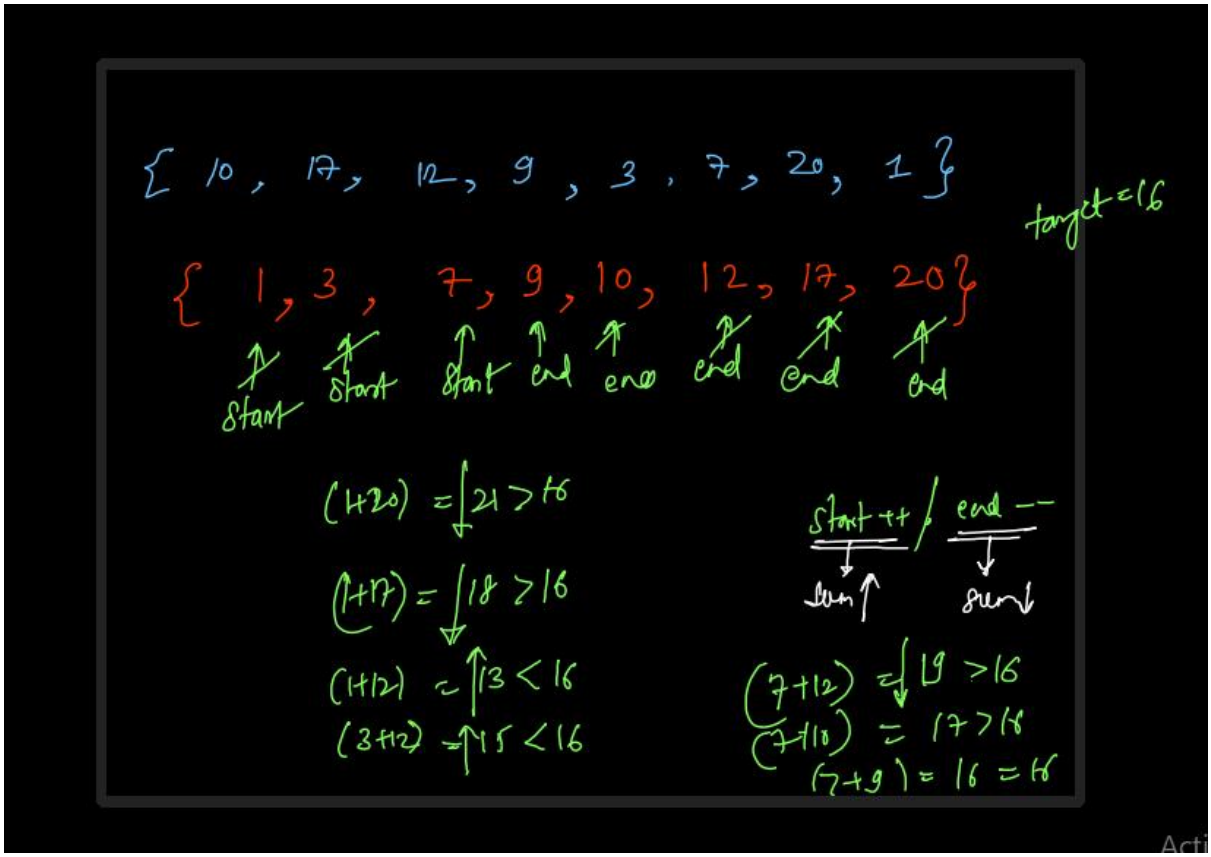
= 3

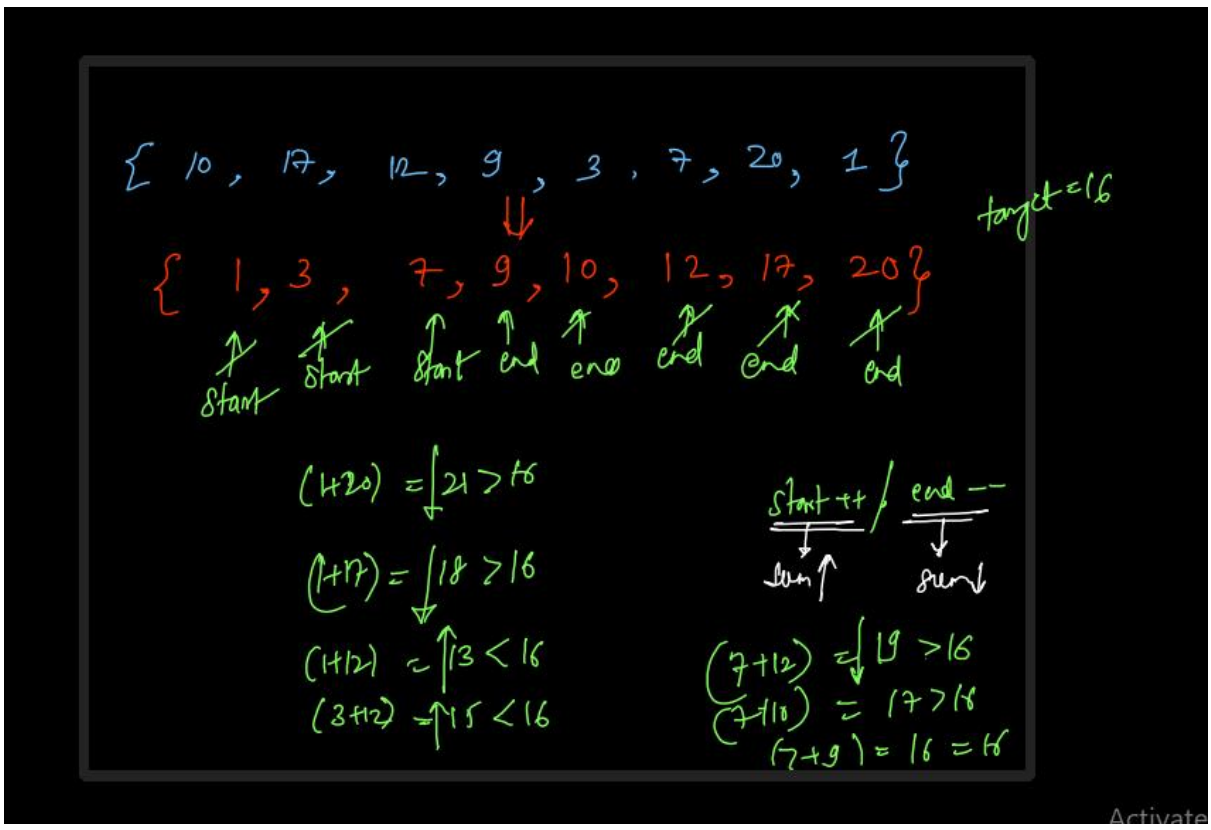

(3) Reversal algorithm
N=9, k=3
Input
k=3
A
B
9-3 = N-k=6
0 1 2 3 4 5 6 7 8
10 20 30 40 50 60 70 80 90
30 20 10 90 80 70 60 50 40
N-k=6
B
k=3
A
Output
40 50 60 70 80 90 10 20 30

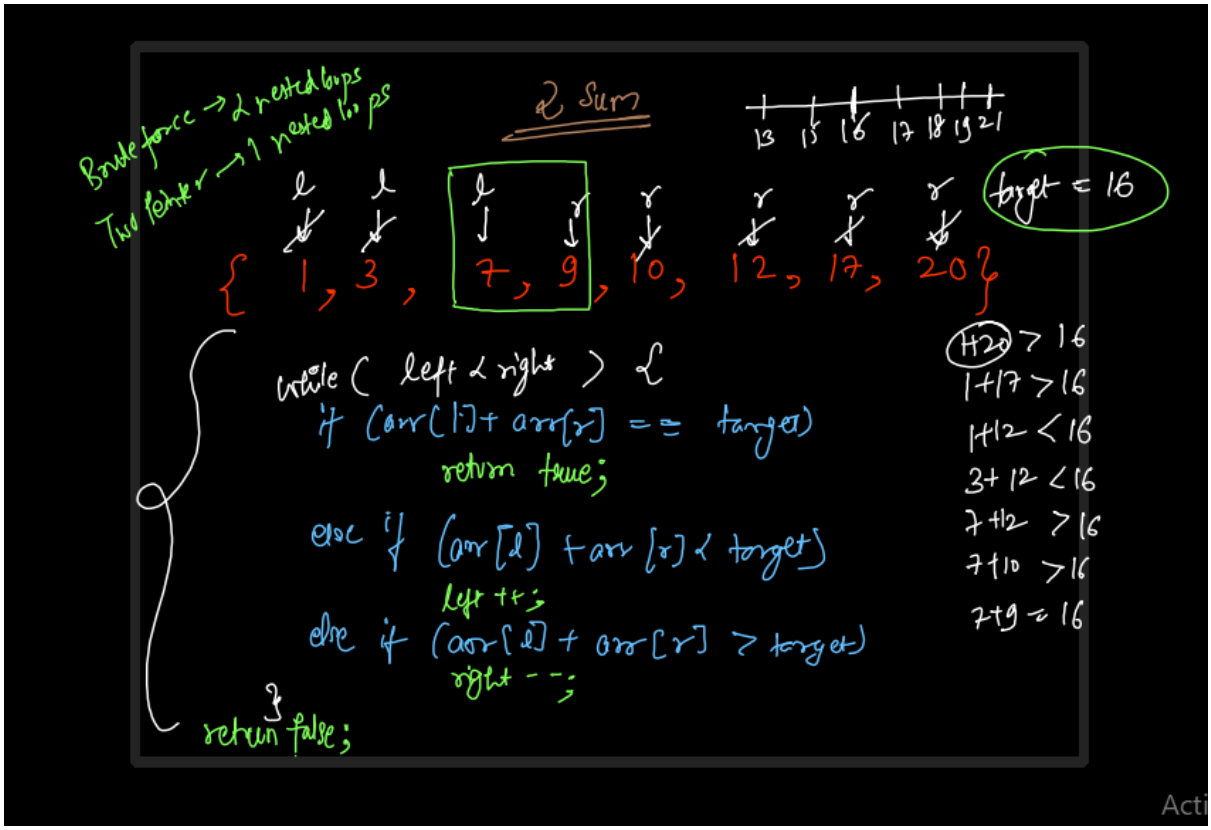

public static void reverse(int[] arr, int start, int end){
while(start < end){
int temp = arr[start];
arr[start] = arr[end];
arr[end] = temp;
start++; end--;
}
}
# WHAT
1 Reverse starting k elements → single loop
+
2 Reverse last n-k elements → single loop
+
3 Reverse the entire array → single loop
Better in terms of time complexity
// 1. Reverse the starting k elements
reverse(input, 0, k - 1);
// 2. Reverse the ending n - k elements
reverse(input, k, n - 1);
// 3. Reverse the entire array
reverse(input, 0, n - 1);
for(int i=0; i<n; i++){
System.out.print(input[i] + " ");
}

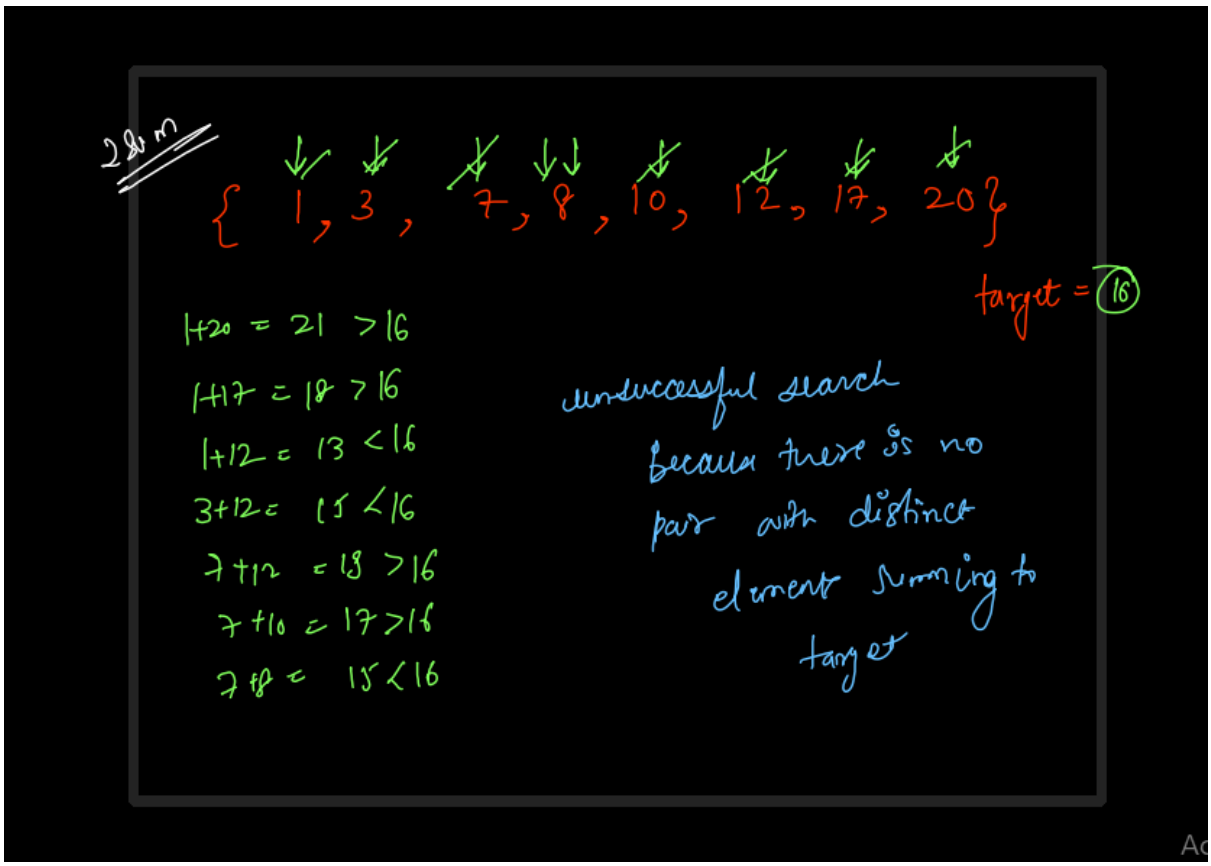

Why
Mathematical Proof
Input
A + B
k elements
(n-k) elements
Output
B^T + A
$(A^T + B^T)^T = (B^T)^T + (A^T)^T = B + A$
① $(x+y)^T = y^T + x^T$
② $(x^T)^T = x$

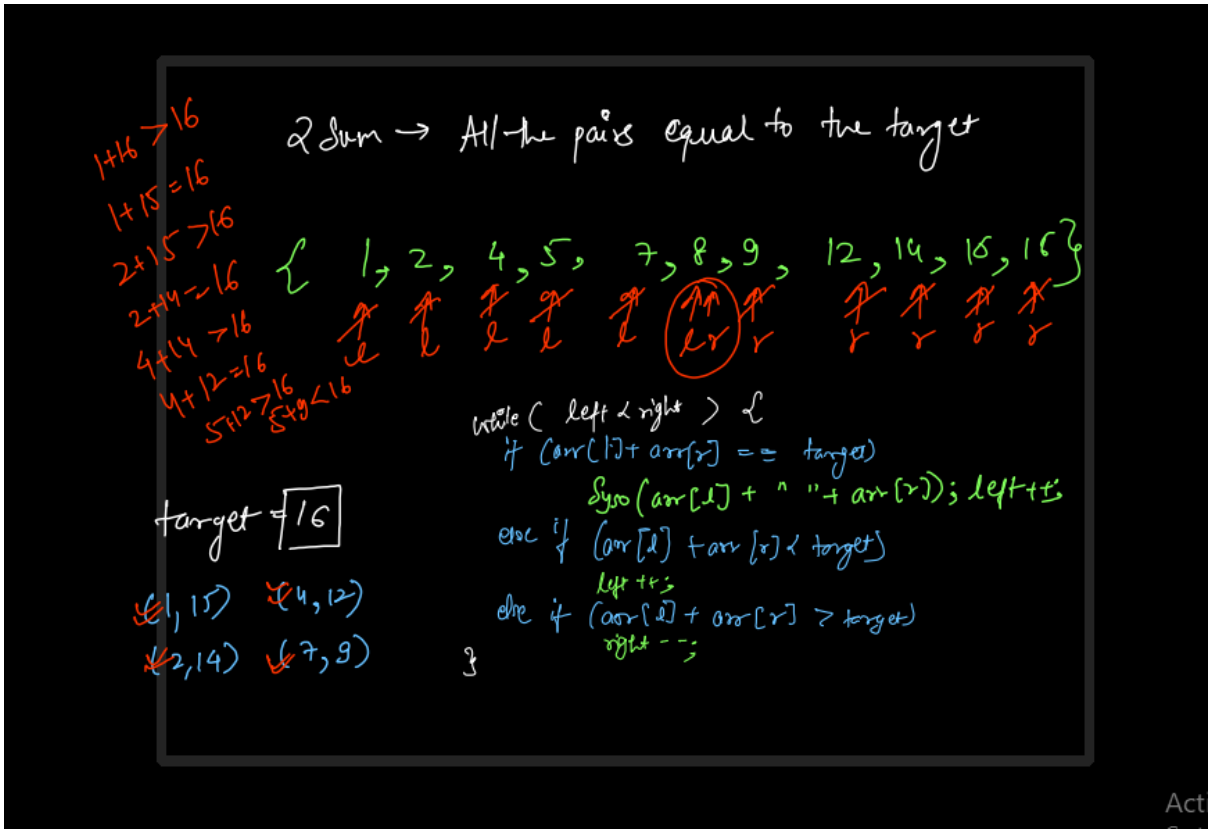

target = 16
Two sum → sum of pair = Target
{ 10, 17, 12, 9, 4, 7, 20, 1 }
Brute force → Nested loops ⇒ Outer loop → I^st & Inner loop → II^nd ele of pair
(10, 17) = 27
(10, 12) = 22
(10, 9) = 19
(10, 4) = 14
(10, 7) = 17
(10, 20) = 30
(10, 1) = 11
(17+12) ≠ 16
(17+9) ≠ 16
(17+4) ≠ 16
(17+20) ≠ 16
(17+1) ≠ 16
(12+9)
(12+4)

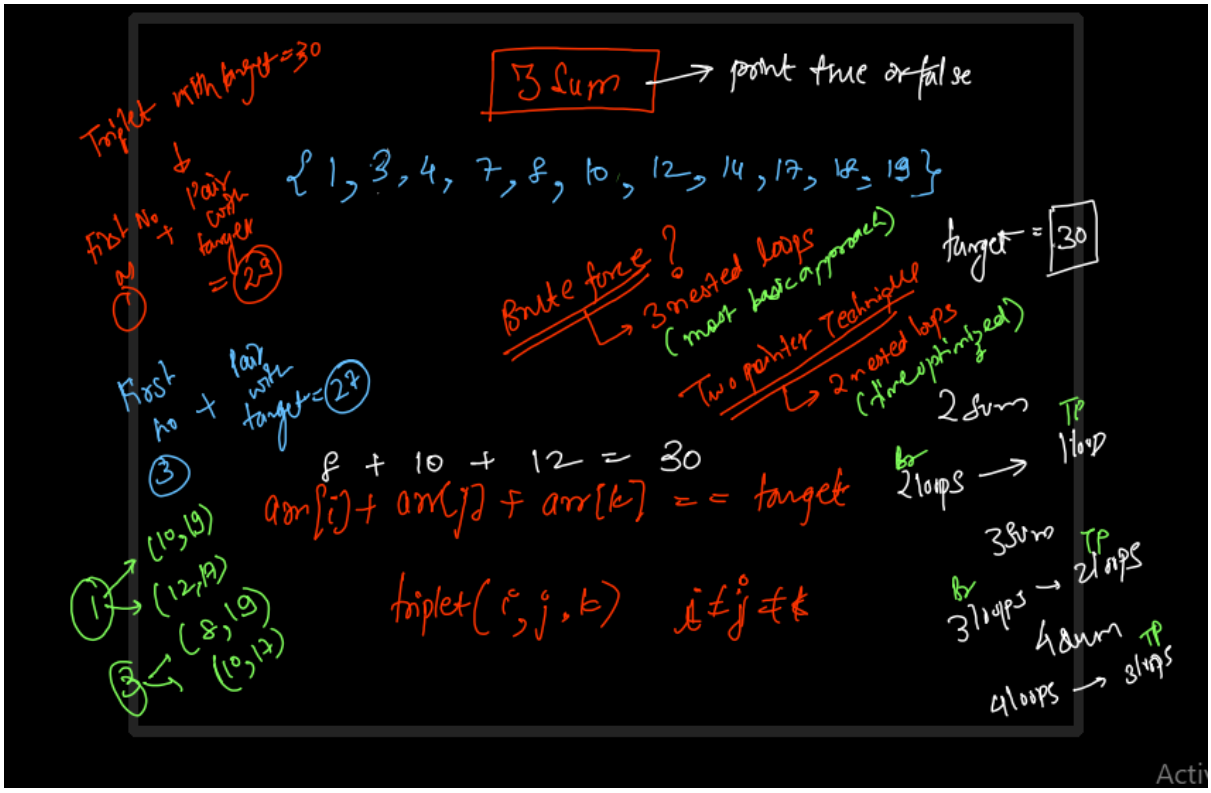

{ 10, 17, 12, 9, 3, 7, 20, 1 }
target = 16
{ 1, 3, 7, 9, 10, 12, 17, 20 }
start start start end end end end
(1+20) = 21 > 16
(1+17) = 18 > 16
(1+12) = 13 < 16
(3+12) = 15 < 16
start ++ / end --
sum ↑ sum ↓
(7+12) = 19 > 16
(7+10) = 17 > 16
(7+9) = 16 = 16

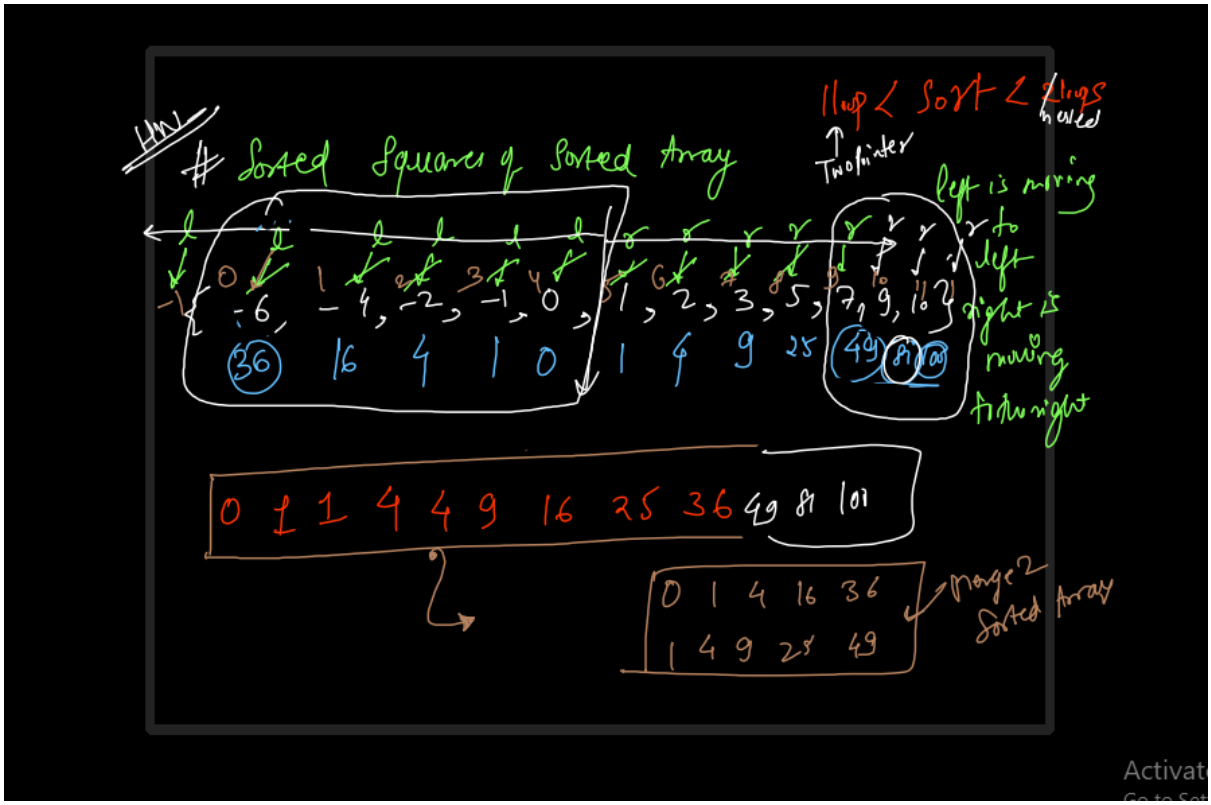

{ 10, 17, 12, 9, 3, 7, 20, 1 }
target = 16
{ 1, 3, 7, 9, 10, 12, 17, 20 }
start start start end end end end
(1+20) = 21 > 16
(1+17) = 18 > 16
(1+12) = 13 < 16
(3+12) = 15 < 16
start ++ / end --
sum ↑ sum ↓
(7+12) = 19 > 16
(7+10) = 17 > 16
(7+9) = 16 = 16

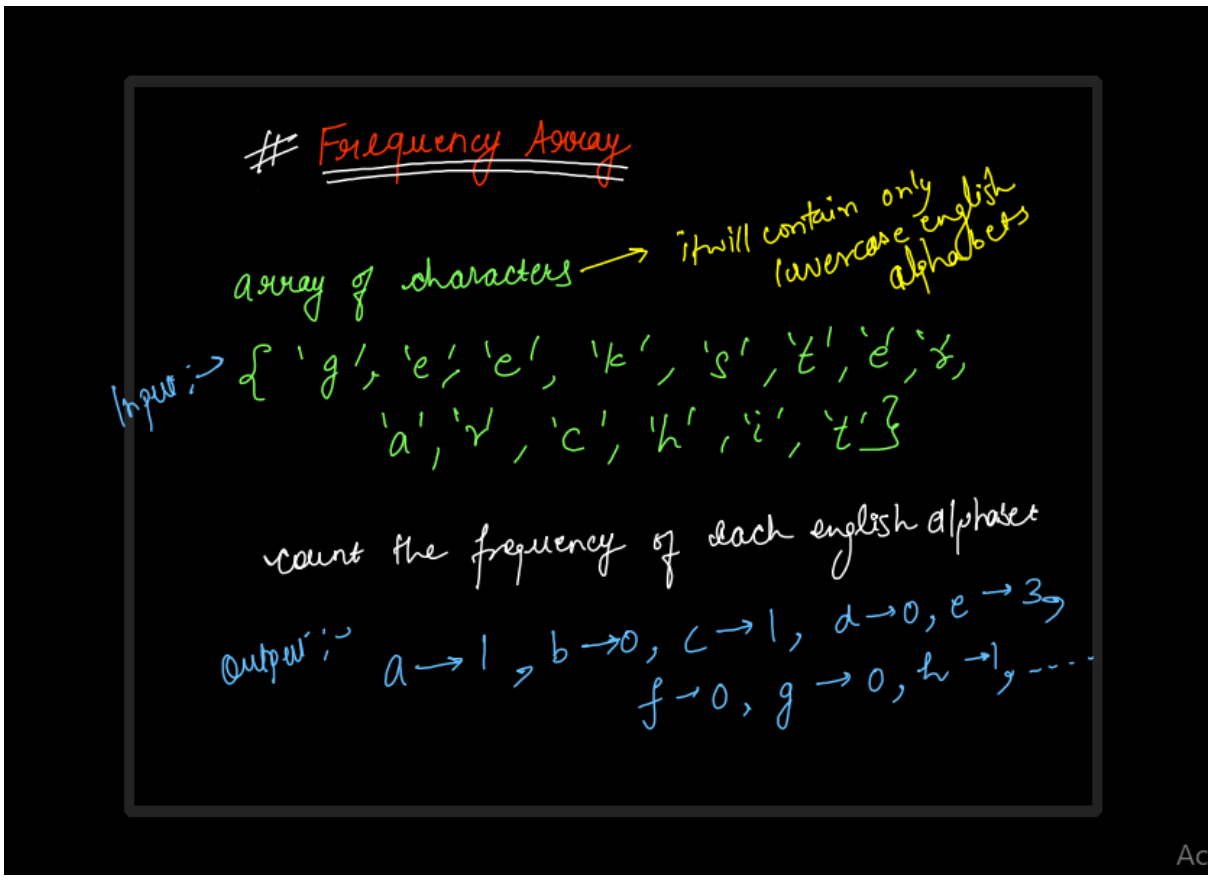

Brute force → 2 nested loops
Two pointer → 1 nested loops
2 Sum
13 15 16 17 18 19 21
target = 16
{ 1, 3, 7, 9, 10, 12, 17, 20 }
while (left < right) {
if (arr[l] + arr[r] == target)
return true;
else if (arr[l] + arr[r] < target)
left ++;
else if (arr[l] + arr[r] > target)
right --;
}
return false;
1+20 > 16
1+17 > 16
1+12 < 16
3+12 < 16
7+12 > 16
7+10 > 16
7+9 = 16


2 Sum
{ 1, 3, 7, 8, 10, 12, 17, 20 }
target = 16
1+20 = 21 > 16
1+17 = 18 > 16
1+12 = 13 < 16
3+12 = 15 < 16
7+12 = 19 > 16
7+10 = 17 > 16
7+8 = 15 < 16
unsuccessful search
because there is no
pair with distinct
element summing to
target


2 Sum → All the pairs equal to the target
{ 1, 2, 4, 5, 7, 8, 9, 12, 14, 15, 16 }
1+16 > 16
1+15 = 16
2+15 > 16
2+14 = 16
4+14 > 16
4+12 = 16
5+12 > 16
5+9 < 16
target = 16
(1, 15) (4, 12)
(2, 14) (7, 9)
while (left < right) {
if (arr[l] + arr[r] == target)
Sysо (arr[l] + " " + arr[r]); left++;
else if (arr[l] + arr[r] < target)
left++;
else if (arr[l] + arr[r] > target)
right--;
}


Triplet with target = 30
3 Sum → print true or false
{ 1, 3, 4, 7, 8, 10, 12, 14, 17, 18, 19 }
First No + pair with target = 29
First No + pair with target = 27
target = 30
Brute force ?
3 nested loops (most basic approach)
Two pointer Technique
2 nested loops (time optimized)
8 + 10 + 12 = 30
arr[i] + arr[j] + arr[k] == target
triplet (i, j, k) i ≠ j ≠ k
2 Sum
for 2 loops → TP 1 loop
3 Sum
for 3 loops → TP 2 loops
4 Sum
4 loops → TP 3 loops


Hint
Sorted Squares of Sorted Array
Two pointer
left is moving to left
right is moving to right
-6 -4 -2 -1 0 1 2 3 5 7 9 10
36 16 4 1 0 1 4 9 25 49 81 100
0 1 1 4 4 9 16 25 36 49 81 101
0 1 4 16 36
1 4 9 25 49
merge 2 sorted array


Frequency Array
array of characters → it will contain only lowercase english alphabets
Input :- { 'g', 'e', 'e', 'k', 's', 'f', 'o', 'r', 'a', 'n', 'd', 'h', 'i', 't' }
count the frequency of each english alphabet
Output :- a → 1, b → 0, c → 1, d → 0, e → 3, f → 0, g → 0, h → 1, ...

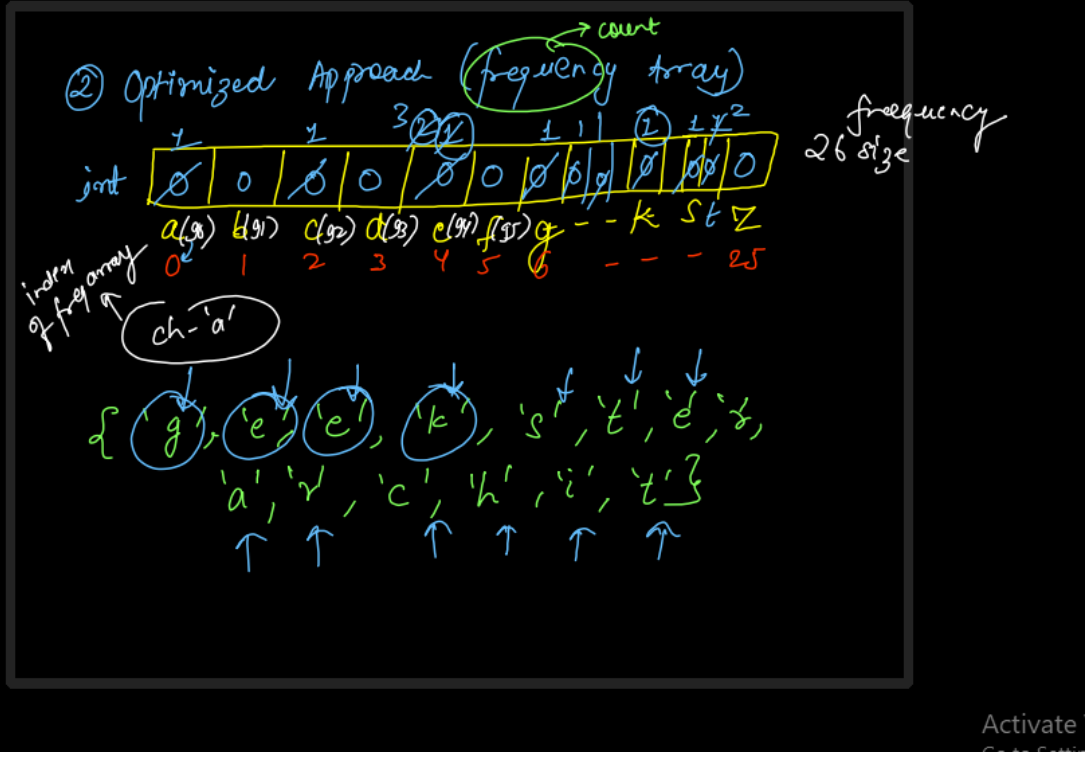
```java
// Brute Force
// Outer Loop: Character
for(char i='a'; i<='z'; i++){

    // Inner Loop: Array
    int count = 0;
    for(int j=0; j<n; j++){
        if(arr[j] == i){
            count++;
        }
    }

    System.out.println(i + " -> " + count);
}
```

26 * n

↳ Comparisons or operations

or

TC



```java
// Optimized Approach
int[] freq = new int[26];
// Char 'a' frequency will be stored in 0th index
// Char 'b' frequency will be stored in 1st index
// Char 'c' frequency will be stored in 2nd index
// Char 'z' frequency will be stored in 25th index

// Loop will run for N iterations
for(int i=0; i<n; i++){
    int index = arr[i] - 'a';
    freq[index]++;
}

// Loop will run for 26 iterations
for(int i=0; i<26; i++){
    System.out.print(freq[i] + " ");
}
```

O(N)

+

O(26)

O(N + 26)

These are not nested loops

Q2 There are N students. Each students have got marks from 0 to 100.

You have to count how many students got O, A+, A, B+, B, C+, C, D+, D, F

O: >= 90
A+: >= 80, < 90
A: >= 70, < 80
B+: >= 60, < 70
B: >= 50, < 60
C+: >= 40, < 50
C: >= 30, < 40
D+: >= 20, < 30
D: >= 10, < 20
F: < 10

{ 10, 20, 40, 5, 92, 96, 33, 44, 55, 60, 88 }

count of O = 2, count of A+ = 1, count of A = 0, ...

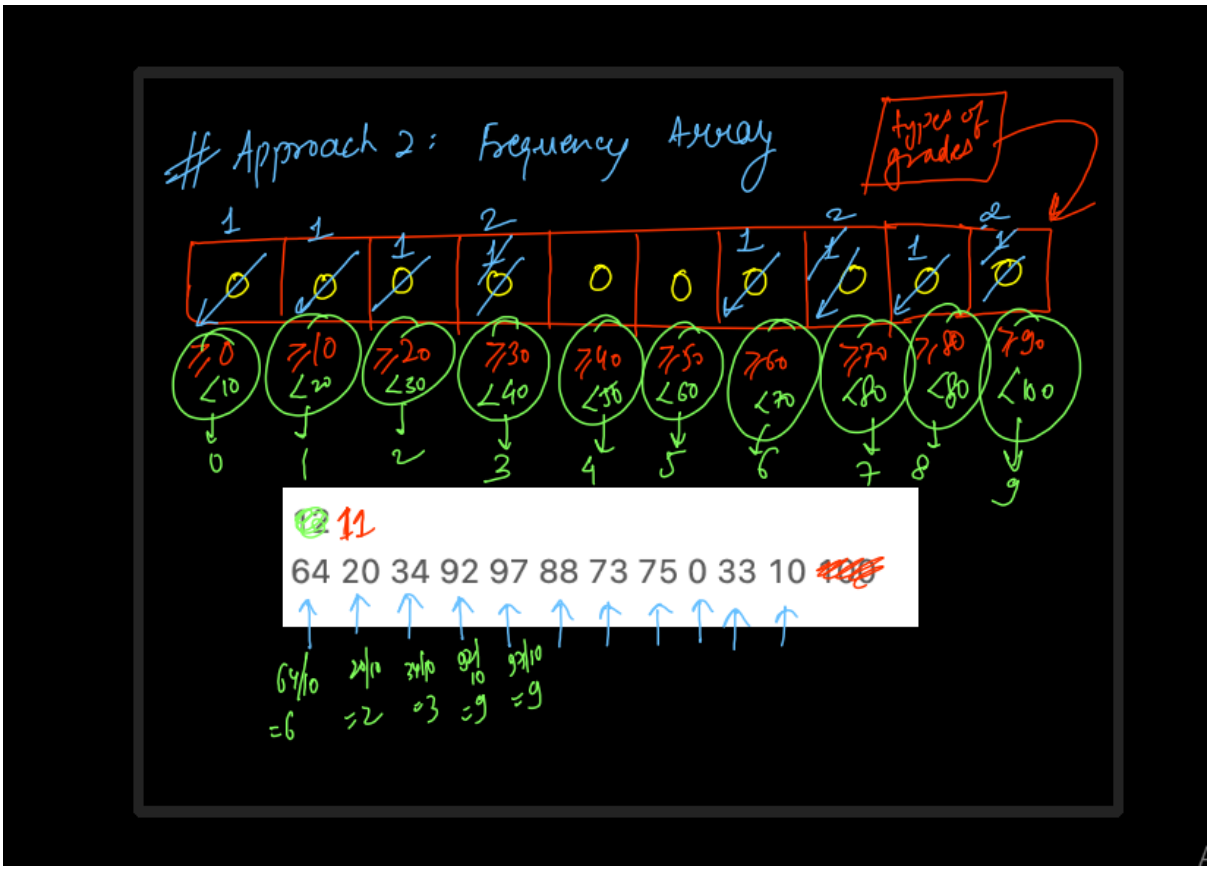
```java
// Brute Force
// Outer Loop: Pick a Grade
for(int grade=90; grade>=0; grade = grade-10){
    // Count Students in that Grade
    int count = 0;
    for(int i=0; i<n; i++){
        if(marks[i] >= grade && marks[i] < grade + 10){
            count++;
        }
    }
    System.out.println(" >= " + grade + " -> " + count);
}
```

O(10) * O(n)

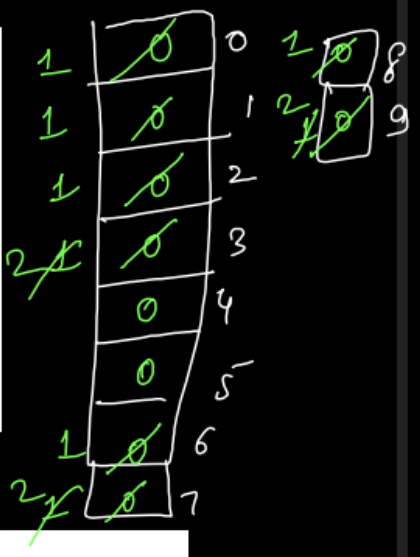
O(10 * n)

nested (generally)

```
Finished in 125 ms
 >= 90 -> 2
 >= 80 -> 1
 >= 70 -> 2
 >= 60 -> 1
 >= 50 -> 0
 >= 40 -> 0
 >= 30 -> 2
 >= 20 -> 1
 >= 10 -> 1
 >= 0 -> 1
stdin
12
64 20 34 92 97 88 73 75 0 33 10 100
```

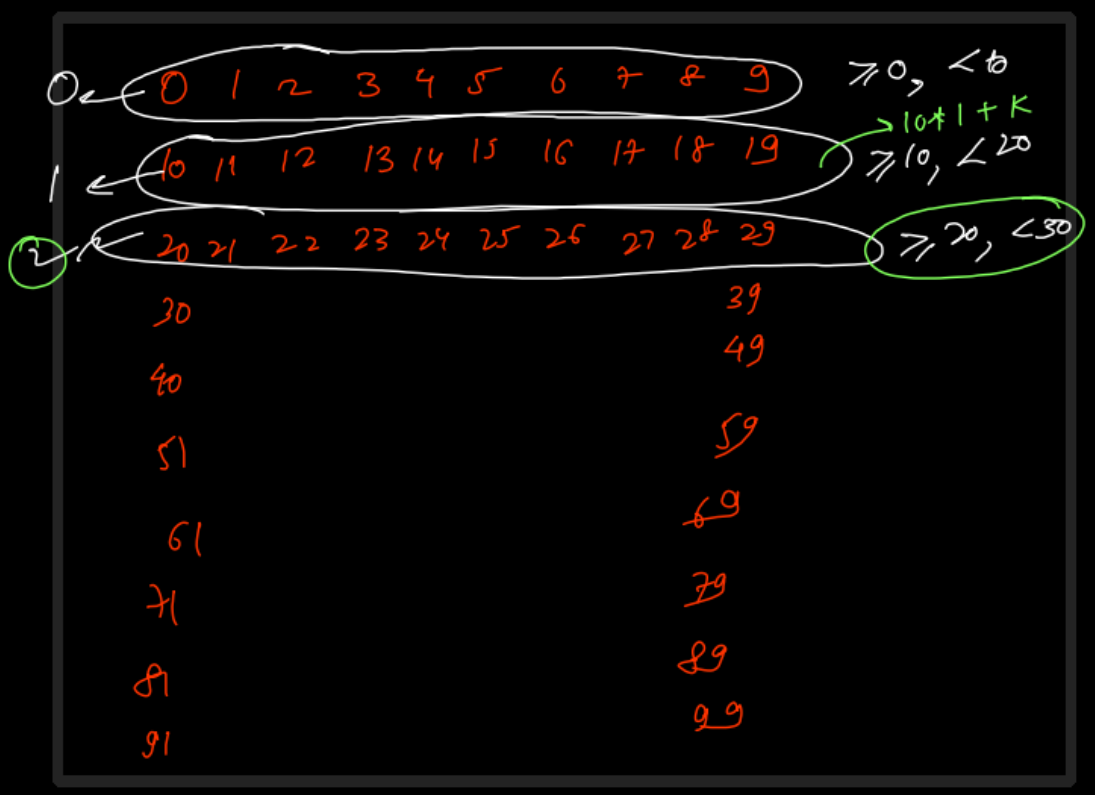


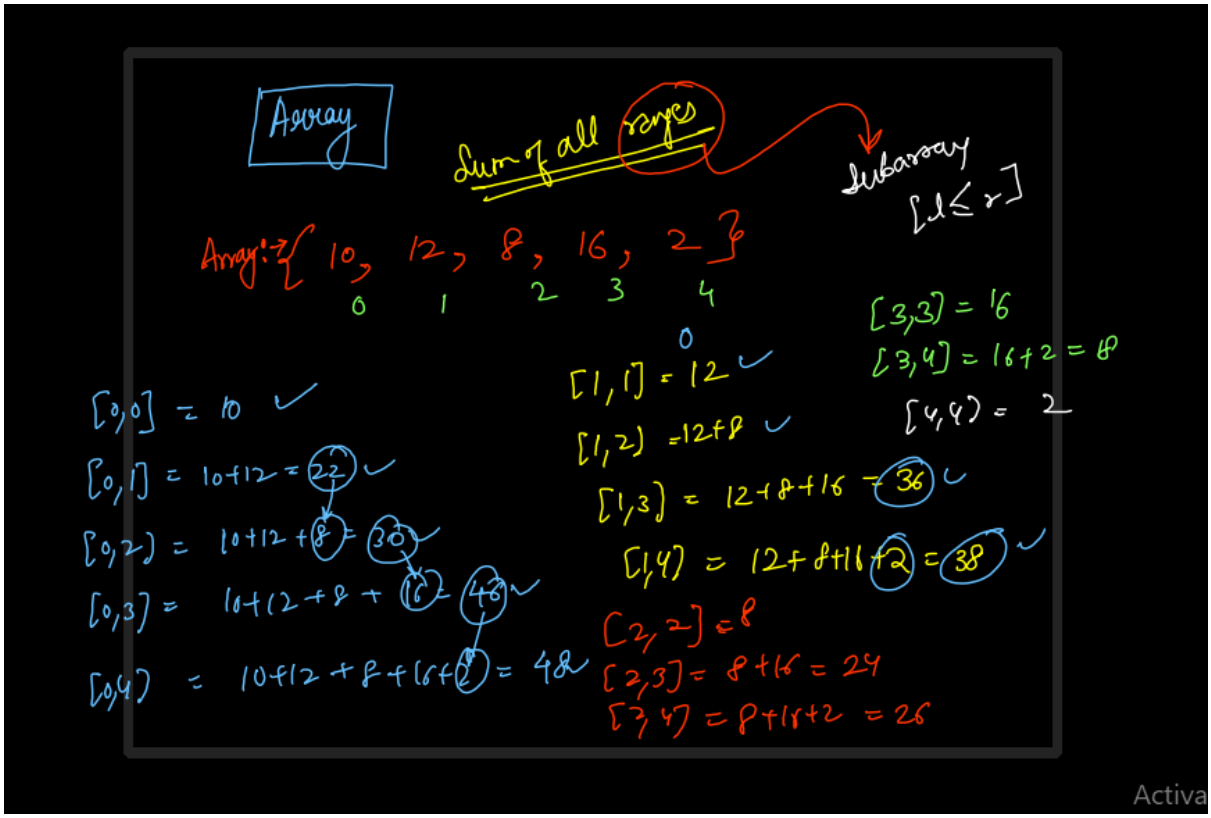
```java
int[] freq = new int[10];

// Optimized Approach
for(int i=0; i<n; i++){
    int index = marks[i] / 10;
    freq[index]++;
}

for(int i=0; i<10; i++){
    System.out.print(freq[i] + " ");
}
```

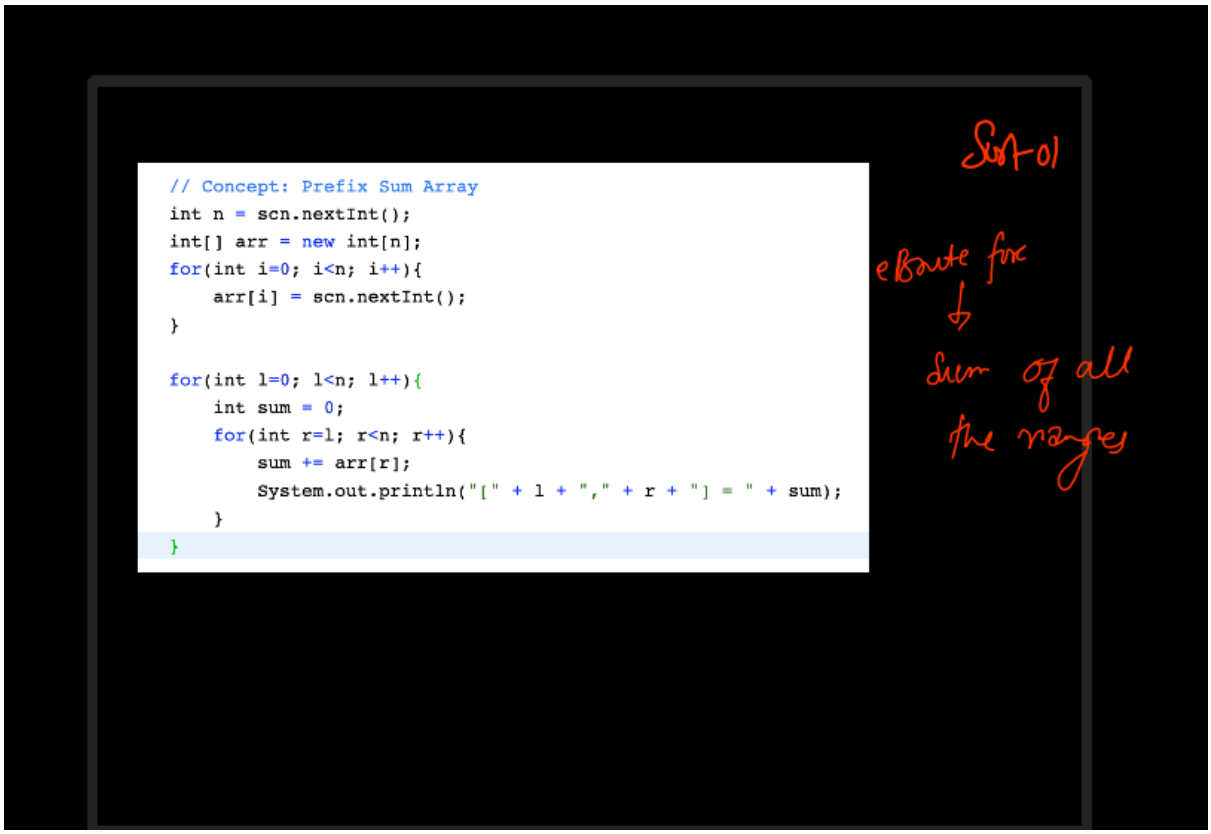


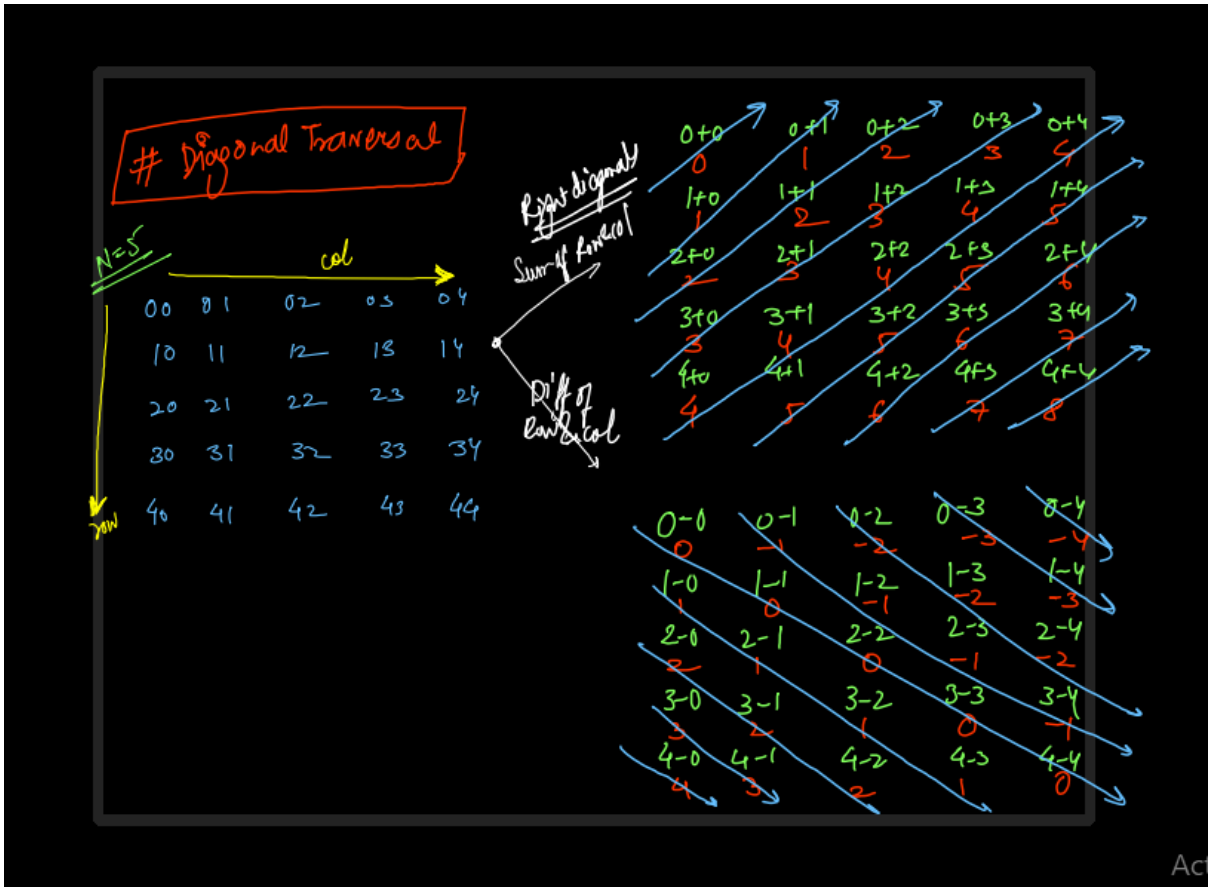
12

64 20 34 92 97 88 73 75 0 33 10 100

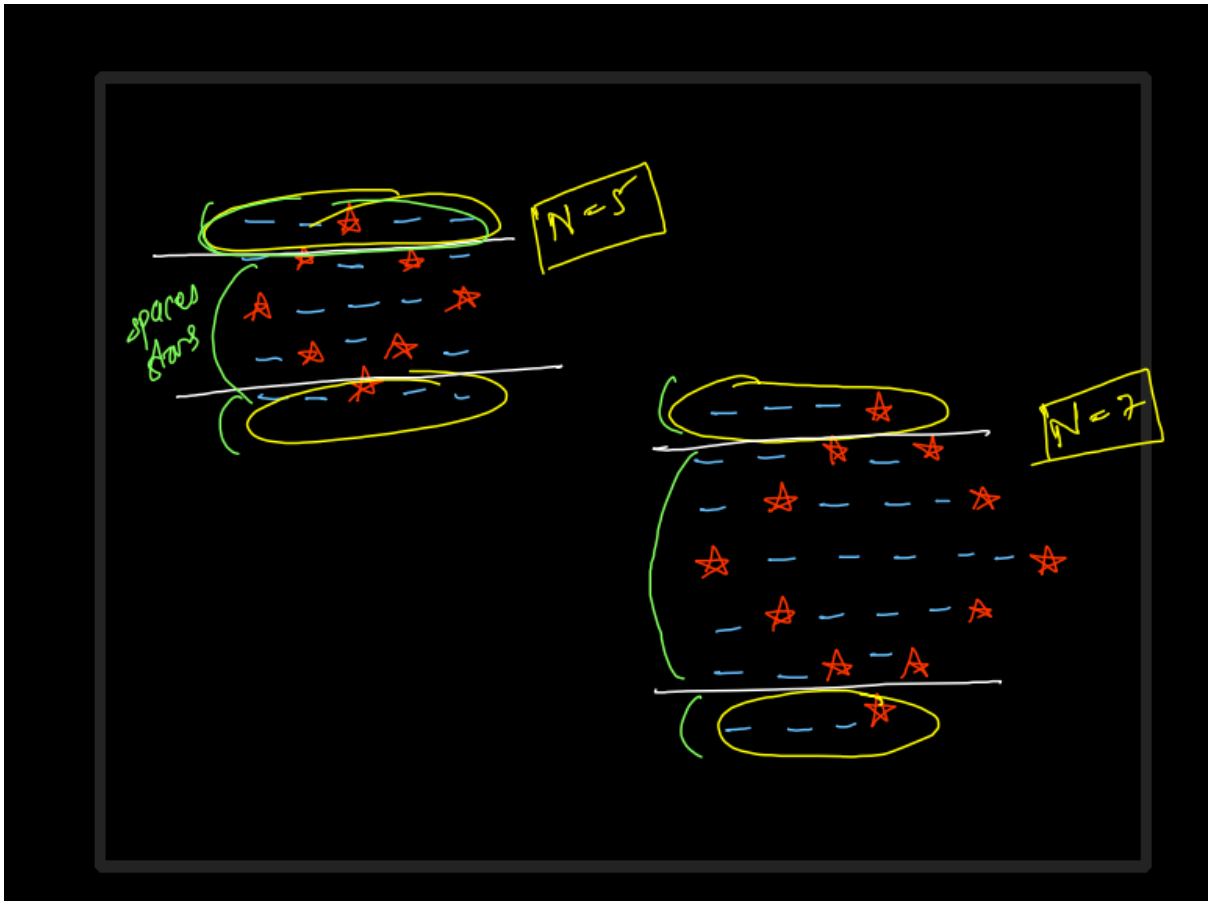


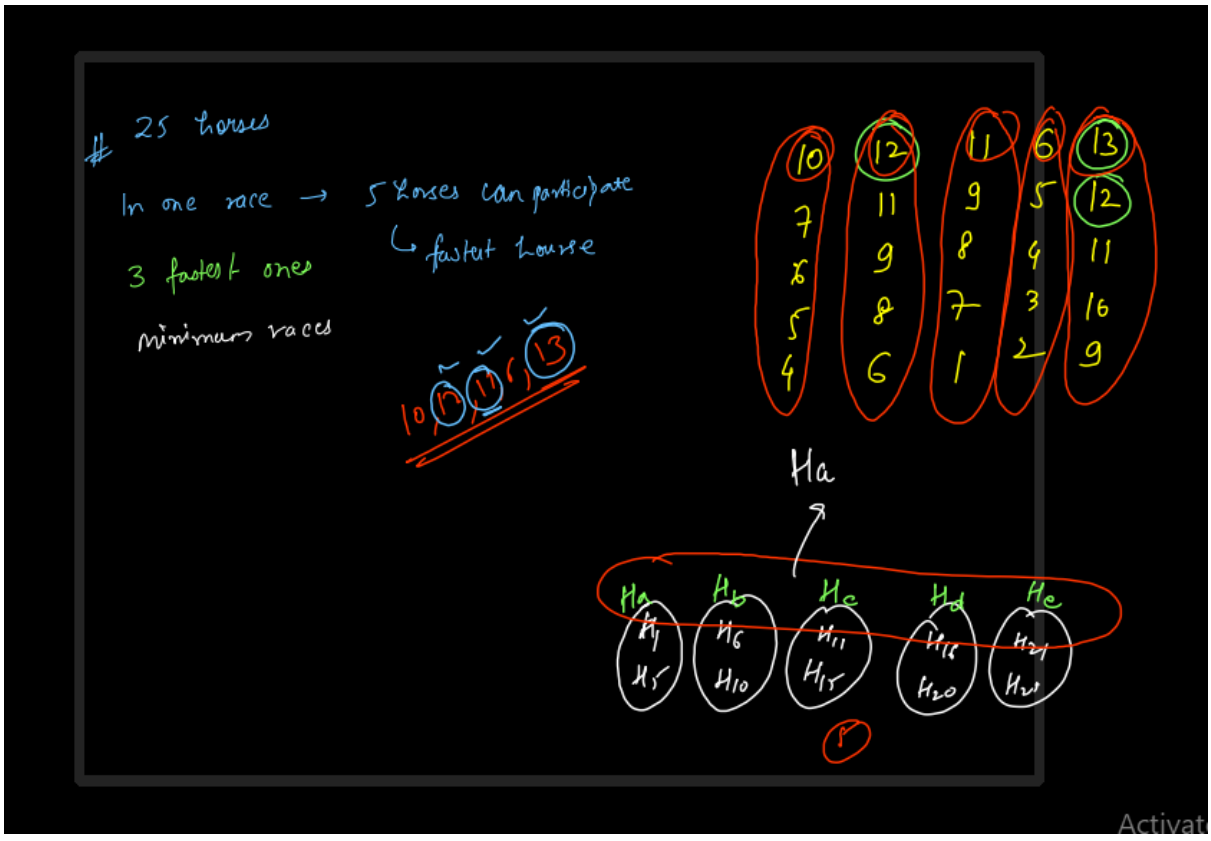

Array
Sum of all ranges
subarray [l ≤ r]
Array: { 10, 12, 8, 16, 2 }
0 1 2 3 4
[0,0] = 10
[0,1] = 10+12 = 22
[0,2] = 10+12+8 = 30
[0,3] = 10+12+8+16 = 46
[0,4] = 10+12+8+16+2 = 48
[1,1] = 12
[1,2] = 12+8
[1,3] = 12+8+16 = 36
[1,4] = 12+8+16+2 = 38
[2,2] = 8
[2,3] = 8+16 = 24
[2,4] = 8+16+2 = 26
[3,3] = 16
[3,4] = 16+2 = 18
[4,4] = 2

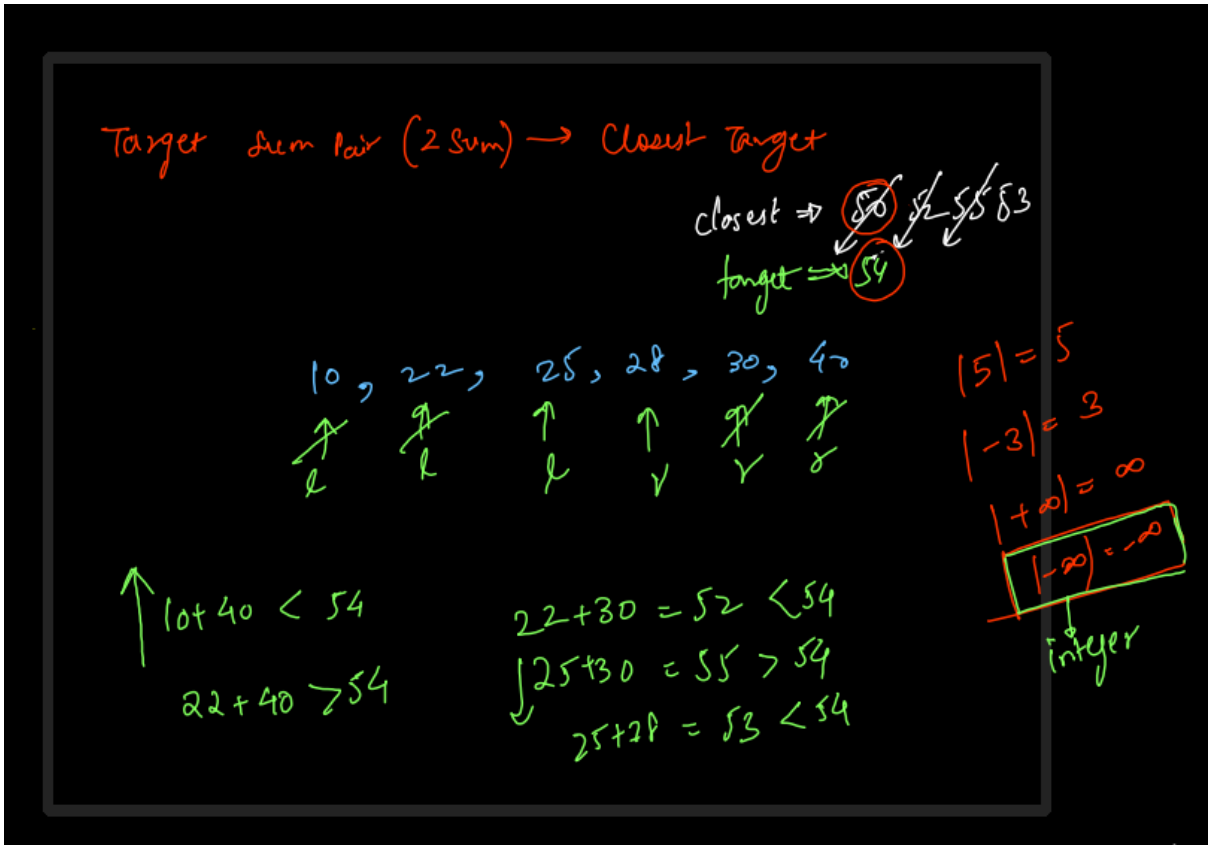

// Concept: Prefix Sum Array
int n = scn.nextInt();
int[] arr = new int[n];
for(int i=0; i<n; i++){
arr[i] = scn.nextInt();
}
for(int l=0; l<n; l++){
int sum = 0;
for(int r=l; r<n; r++){
sum += arr[r];
System.out.println("[" + l + "," + r + "] = " + sum);
}
}
Soln 01
eBaute fnc
↓
sum of all the ranges

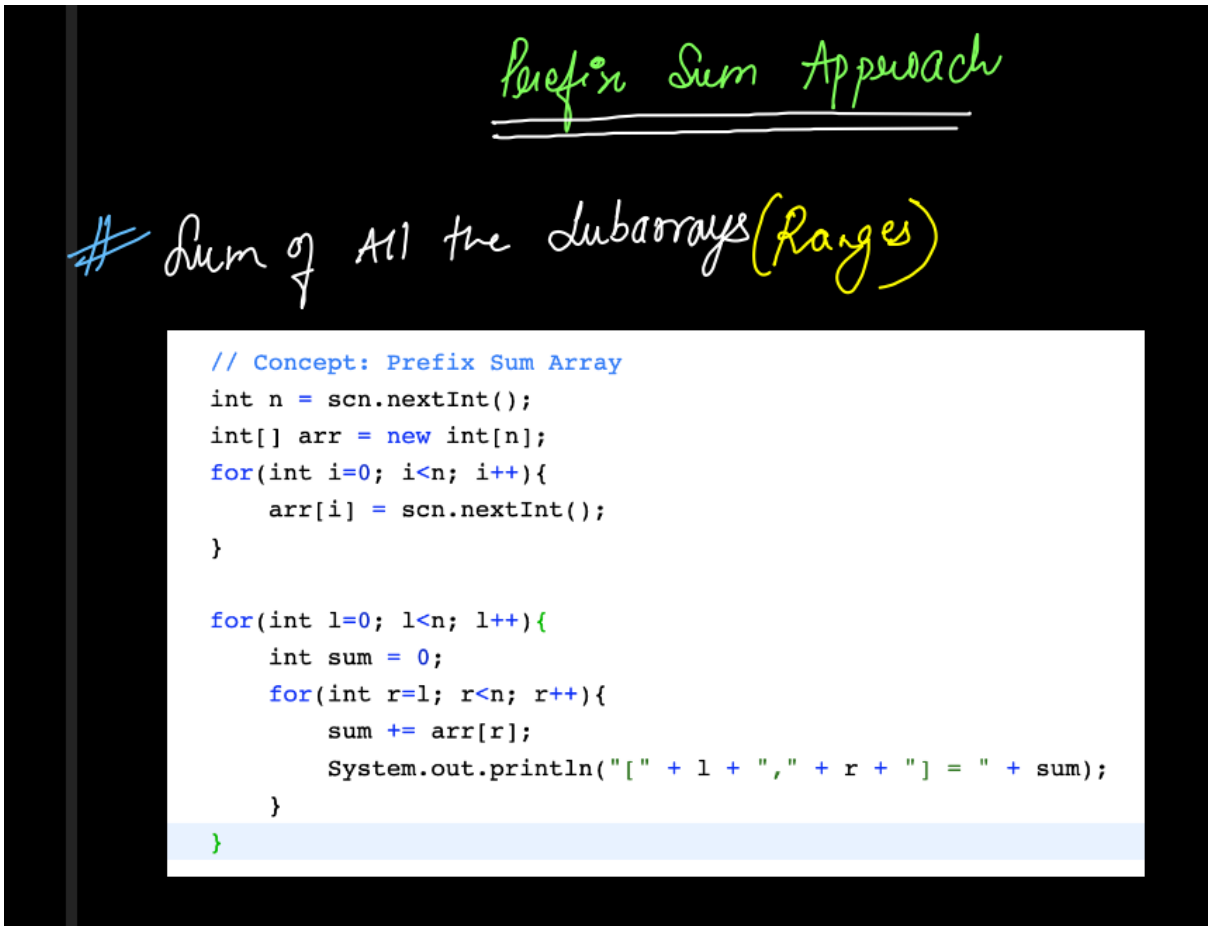



```java
// Sum of Row & Col Will Give Equal Elements on Right Diagonals
for(int row=0; row<n; row++){
    for(int col=0; col<n; col++){
        System.out.print(row + col + " ");
    }
    System.out.println();
}

System.out.println();

// DIfference of Row & Col Will Give Equal Elements on Left Diagonals
for(int row=0; row<n; row++){
    for(int col=0; col<n; col++){
        System.out.print((row - col) + " ");
    }
    System.out.println();
}
```

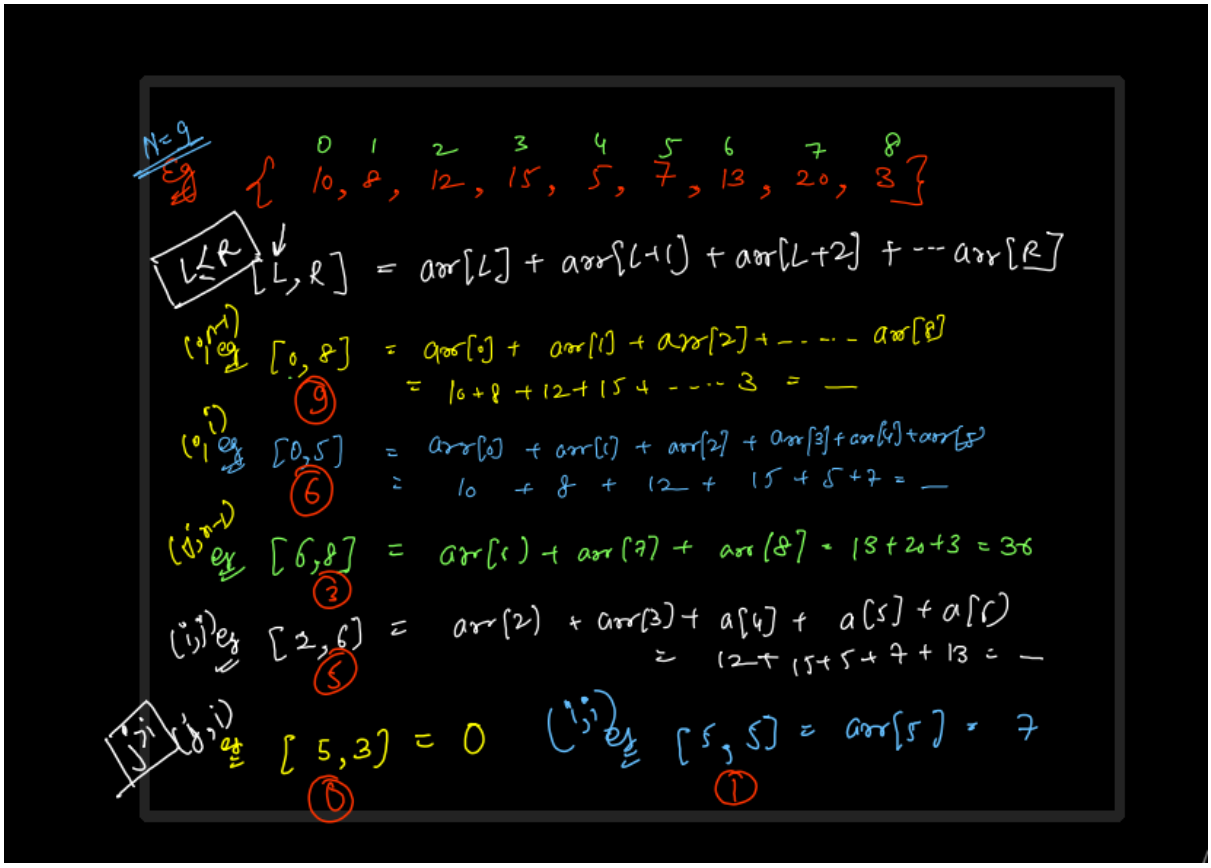

N=5
spaces
stars
N=7


25 horses
In one race → 5 horses can participate
↳ fastest horse
3 fastest ones
minimum races
10 7 8 5 4
12 11 9 8 6
11 9 8 7 1
6 5 4 3 2
13 12 11 10 9
Ha
Hb
Hc
Hd
He
H1 H5
H6 H10
H11 H15
H16 H20
H21
5






```java
// Concept: Prefix Sum Array
int n = scn.nextInt();
int[] arr = new int[n];
for(int i=0; i<n; i++){
    arr[i] = scn.nextInt();
}

for(int l=0; l<n; l++){
    int sum = 0;
    for(int r=l; r<n; r++){
        sum += arr[r];
        System.out.println("[" + l + "," + r + "] = " + sum);
    }
}
```




```java
public static void main(String[] args) {
    // Prefix Sum

    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for(int i=0; i<n; i++){
        arr[i] = scn.nextInt();
    }

    int l = scn.nextInt();
    int r = scn.nextInt();

    // Printing the Sum of Range [L, R] where both L, R are included, L <= R
    int sum = 0;
    for(int i=l; i<=r; i++){
        sum = sum + arr[i];
    }

    System.out.println(sum);
}
```

loop is working for $O(R-L+1)$

$\approx O(N)$

Single Query

Multiple RANGE SUM QUERY

```
int queries = scn.nextInt();
// Multiple Range Sum Queries

for(int q=1; q<=queries; q++){
    int l = scn.nextInt();
    int r = scn.nextInt();

    // Printing the Sum of Range [L, R] where both L, R are included, L <= R
    int sum = 0;
    for(int i=l; i<=r; i++){
        sum = sum + arr[i];
    }

    System.out.println(sum);
}
```

← O(queries)

← O(R-L+1)

Total time

O(Queries * (R-L+1))

Prefix Sum Approach

Input

0	1	2	3	4	5	6	7	8
10	8	12	15	5	7	13	20	3

Prefix

0	1	2	3	4	5	6	7	8
10	18	30	45	50	57	70	90	93

[0,0] [0,1] [0,2] [0,3] [0,4] [0,5] [0,6] [0,8] [0,8]

[0,8] ⇒ prefix[8] = 93

[0,5] ⇒ prefix[5] = 57

[5,3] ⇒ 0

[8,6] ⇒ 0

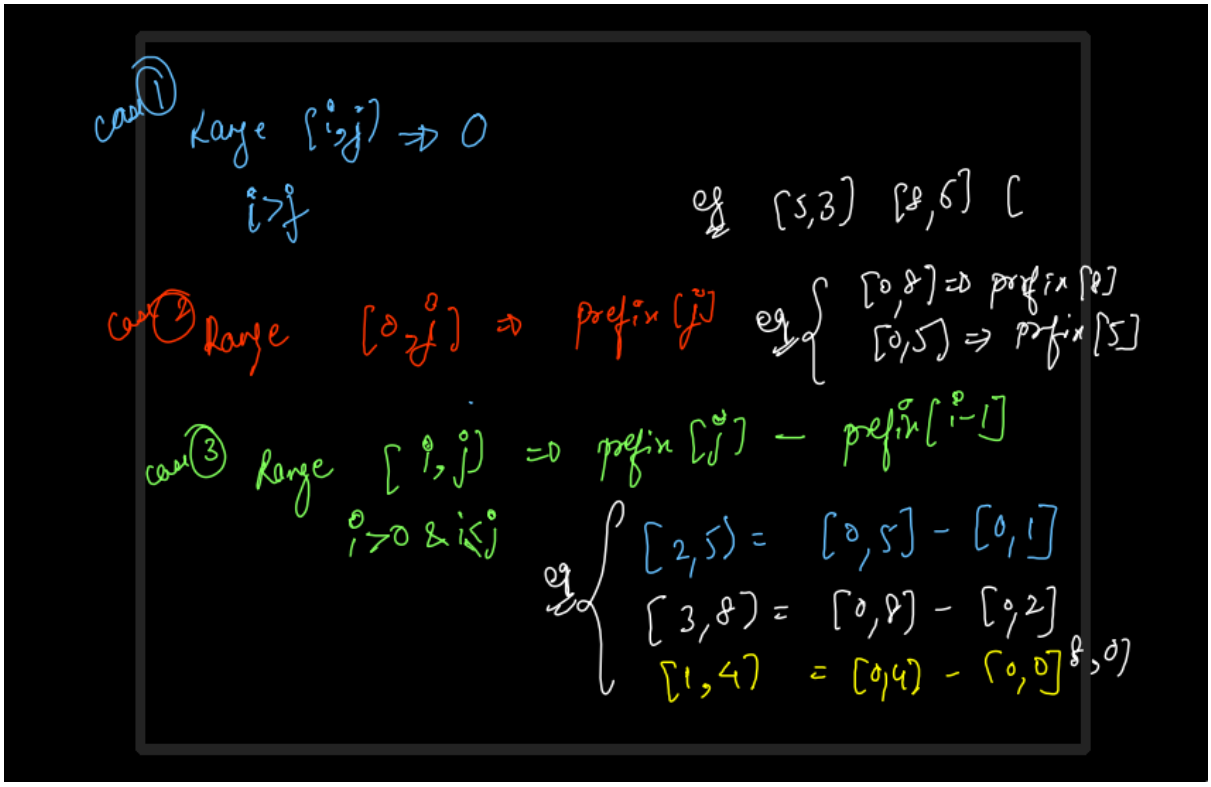
[2,6] = prefix[6] - prefix[1] = 70 - 18 = 52

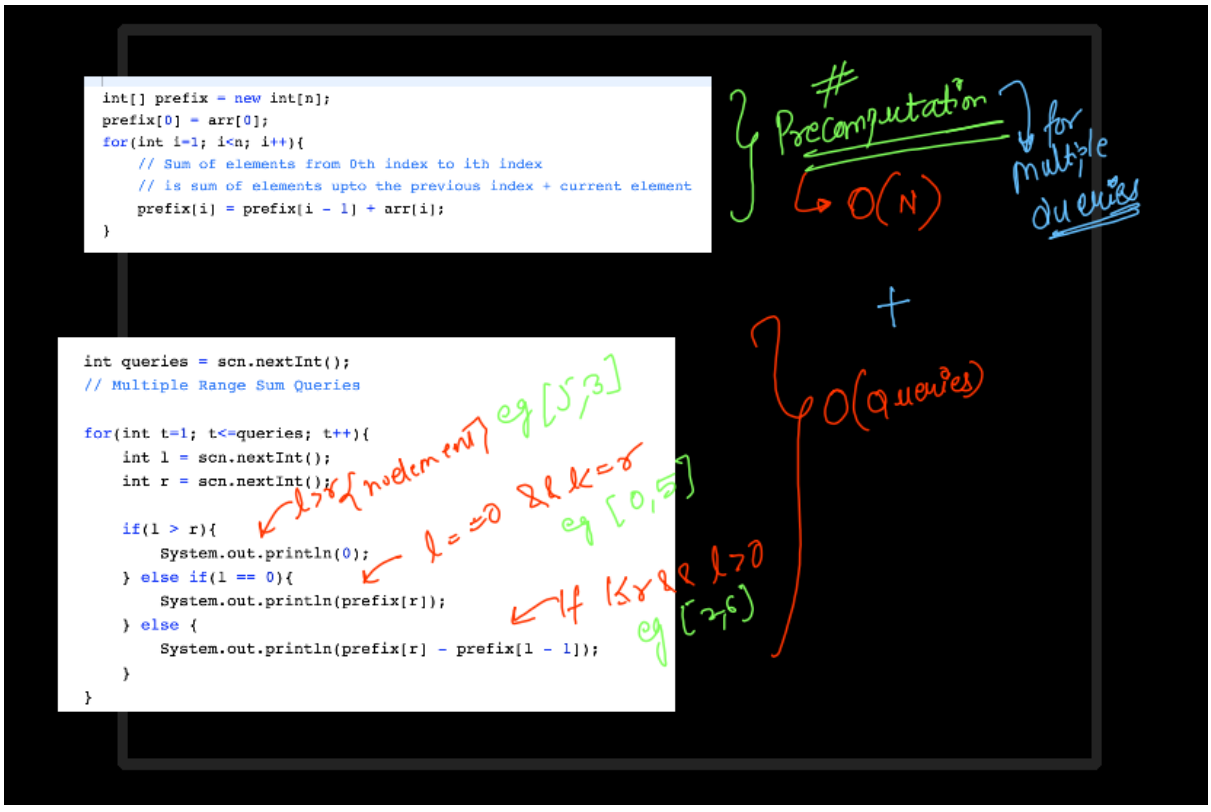
[3,7] = prefix[7] - prefix[2] = 90 - 30 = 60

[1,1] = prefix[1] - prefix[0] = 18 - 10 = 8

pref[i] = pref[i-1] + arr[i]

```
int[] prefix = new int[n];
prefix[0] = arr[0];
for(int i=1; i<n; i++){
    // Sum of elements from 0th index to ith index
    // is sum of elements upto the previous index + current element
    prefix[i] = prefix[i - 1] + arr[i];
}
```


case 1
Range [i,j] ⇒ 0
i > j
eg [5,3] [8,6] [
case 2
Range [0,j] ⇒ prefix[j]
eg [0,8] ⇒ prefix[8]
[0,5] ⇒ prefix[5]
case 3
Range [i,j] ⇒ prefix[j] − prefix[i−1]
i > 0 & i ≤ j
eg [2,5) = [0,5] − [0,1]
[3,8) = [0,8] − [0,2]
[1,4] = [0,4) − [0,0]
8,0)


int[] prefix = new int[n];
prefix[0] = arr[0];
for(int i=1; i<n; i++){
// Sum of elements from 0th index to ith index
// is sum of elements upto the previous index + current element
prefix[i] = prefix[i - 1] + arr[i];
}
# Precomputation
O(N)
for multiple queries
+
int queries = scn.nextInt();
// Multiple Range Sum Queries
for(int t=1; t<=queries; t++){
int l = scn.nextInt();
int r = scn.nextInt();
if(l > r){
System.out.println(0);
} else if(l == 0){
System.out.println(prefix[r]);
} else {
System.out.println(prefix[r] - prefix[l - 1]);
}
}
O(queries)
l > r {no element} eg [5,3]
l == 0 && l <= r
eg [0,5]
If l > r && l > 0
eg [2,6]

Leetcode 1732

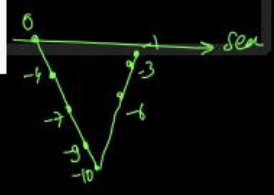
Find the highest altitude

There is a biker going on a road trip. The road trip consists of `n + 1` points at different altitudes. The biker starts his trip on point `0` with altitude equal `0`.

You are given an integer array `gain` of length `n` where `gain[i]` is the **net gain in altitude** between points `i` and `i + 1` for all (`0 <= i < n`).

Return *the* ***highest altitude*** *of a point.*

```
[-4,-3,-2,-1,4,3,2]
```




```java
class Solution {
    public int largestAltitude(int[] gain) {
        int height = 0;
        int maxHeight = 0; // It is Starting at altitude 0

        for(int i=0; i<gain.length; i++){
            height = height + gain[i];

            if(height > maxHeight) {
                maxHeight = height;
            }
        }

        return maxHeight;
    }
}
```

724. Find Pivot Index

Easy 3154 371 Add to List Share

Given an array of integers `nums`, calculate the **pivot index** of this array.

The **pivot index** is the index where the sum of all the numbers **strictly** to the left of the index is equal to the sum of all the numbers **strictly** to the index's right.

If the index is on the left edge of the array, then the left sum is `0` because there are no elements to the left. This also applies to the right edge of the array.

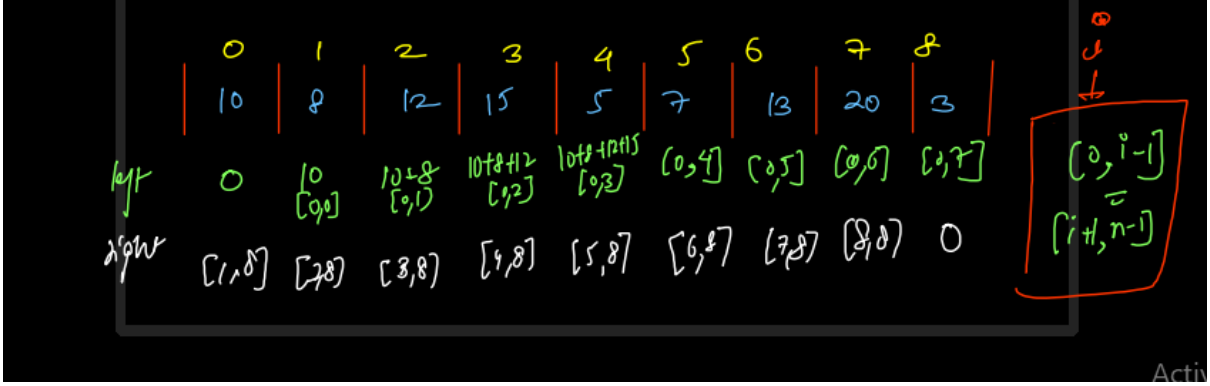
Return *the **leftmost pivot index***. If no such index exists, return -1.

Example 1:

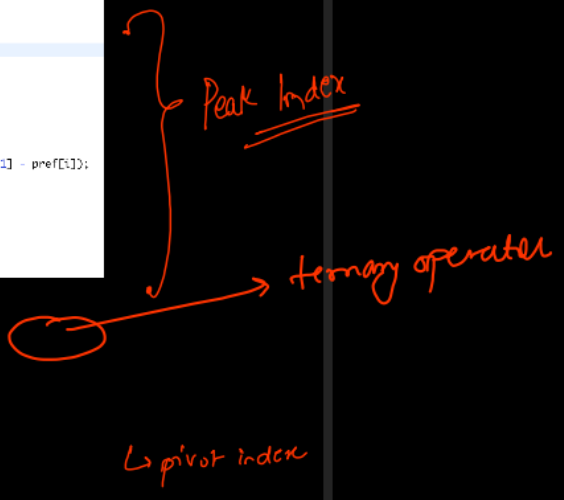
```
Input: nums = [1,7,3,6,5,6]
Output: 3
Explanation:
The pivot index is 3.
Left sum = nums[0] + nums[1] + nums[2] = 1 + 7 + 3 = 11
Right sum = nums[4] + nums[5] = 5 + 6 = 11
```

Example 2:

```
Input: nums = [1,2,3]
Output: -1
Explanation:
There is no index that satisfies the conditions in the
problem statement.
```




```
class Solution {
    public int pivotIndex(int[] nums) {
        int[] pref = new int[nums.length];
        pref[0] = nums[0];
        for(int i=1; i<nums.length; i++){
            pref[i] = pref[i - 1] + nums[i];
        }

        for(int i=0; i<nums.length; i++){
            int leftSum = (i == 0) ? 0 : pref[i - 1];
            int rightSum = (i == nums.length - 1) ? 0 : (pref[nums.length - 1] - pref[i]);

            if(leftSum == rightSum) return i;
        }

        return -1;
    }
}
```



```java
Scanner scn = new Scanner(System.in);
int row = scn.nextInt();
int col = scn.nextInt();
int[][] mat = new int[row][col];
for(int i=0; i<row; i++){
    for(int j=0; j<col; j++){
        mat[i][j] = scn.nextInt();
    }
}
```

outer → row
inner → Col

You have to input the elements of a matrix row by row

rows = 3, cols = 5

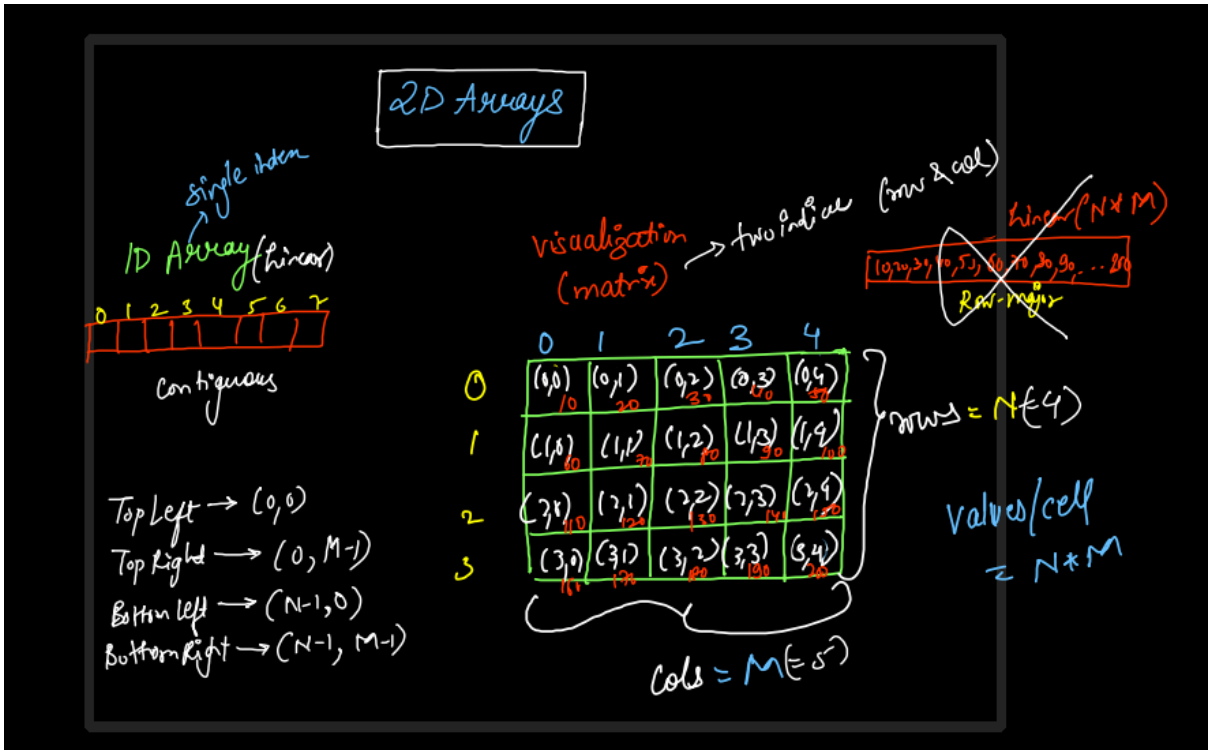
cols (j)

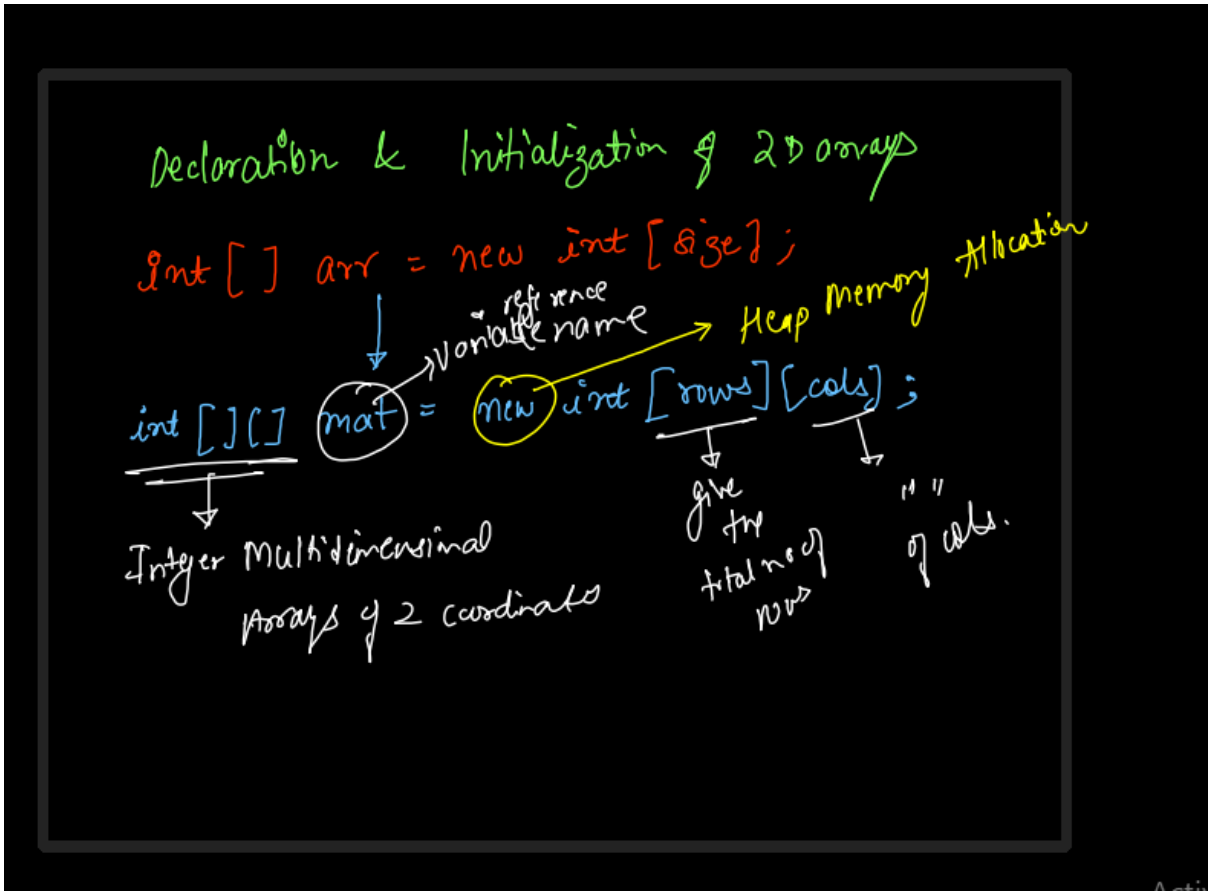
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150

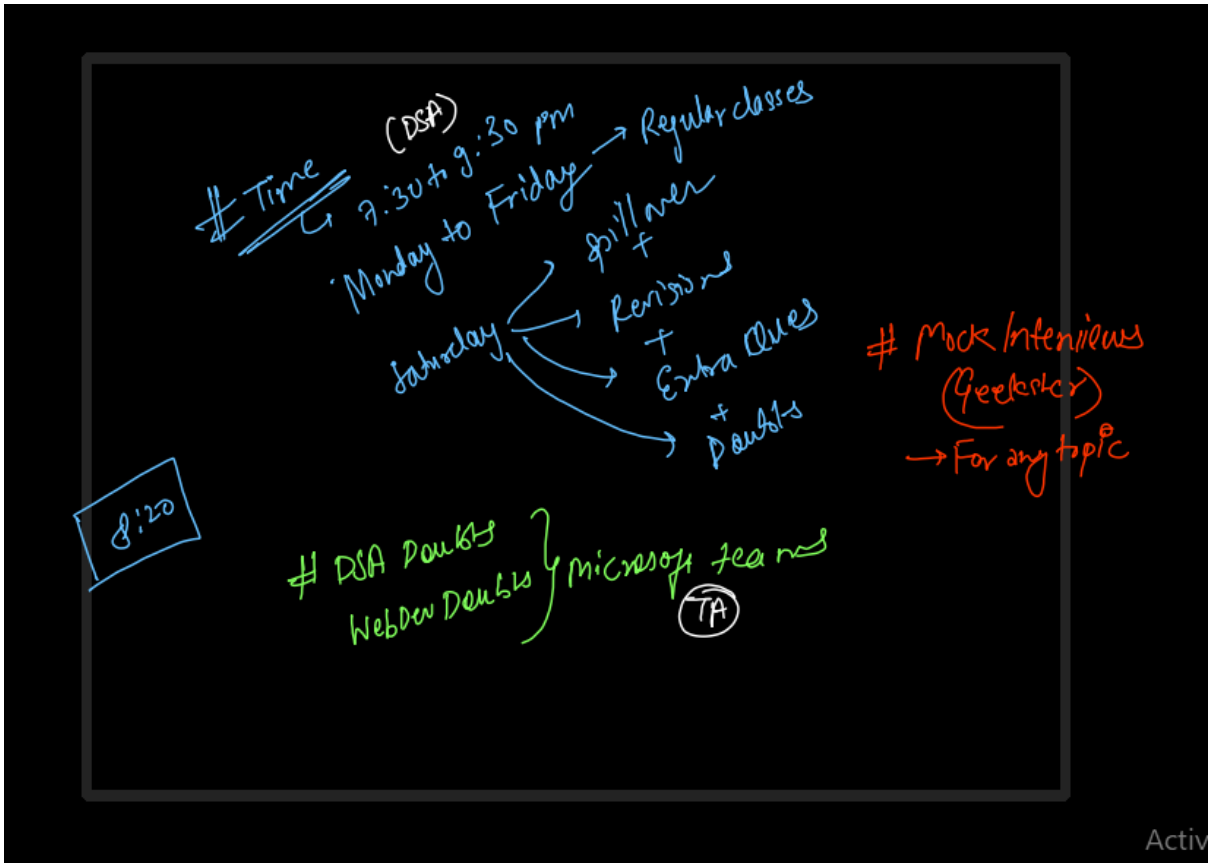
rows = i

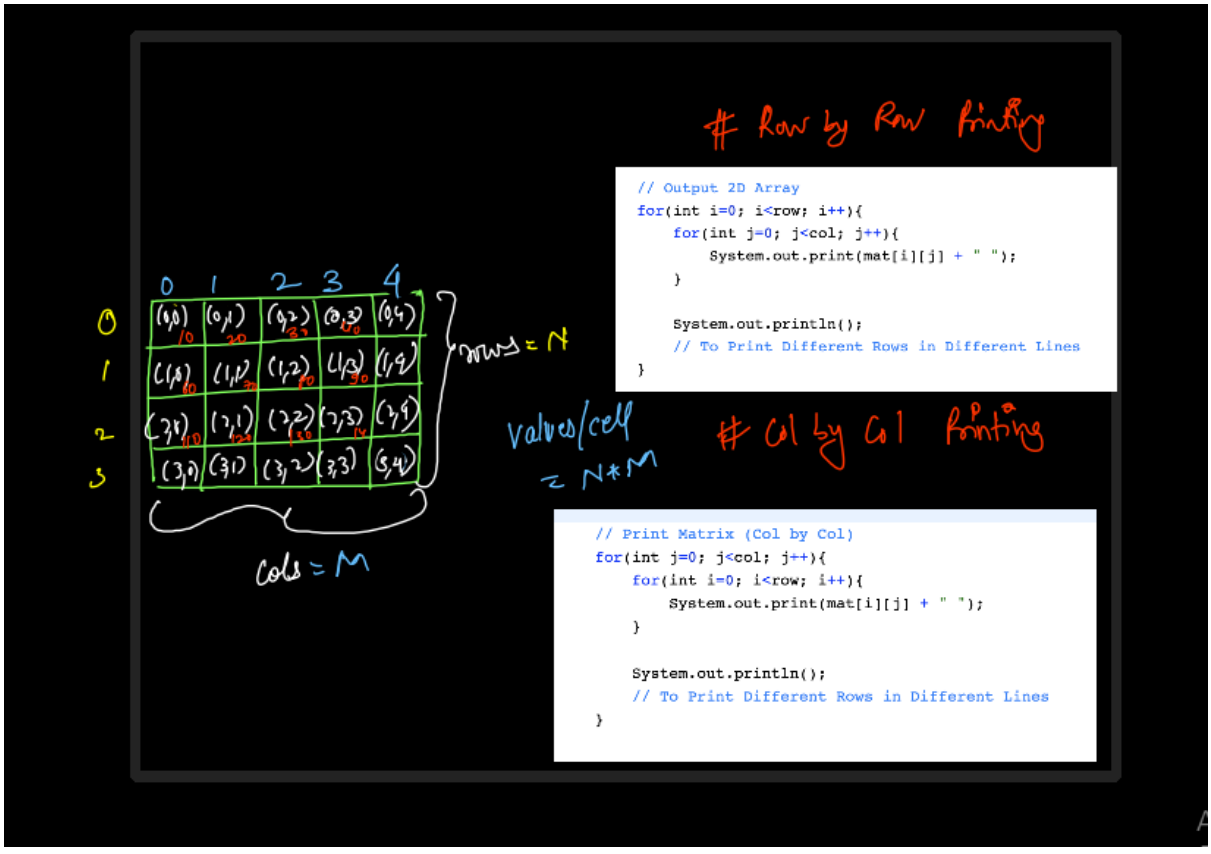
Applications (real life)

1. String maps
2. Image Processing (R, G, B)
   ↳ pixels of images forms matrix
3. Video Processing
4. Board Games → Sudoku, Tic Tac Toe, Snakes & ladder
5. Vectors


2D Arrays
single item
1D Array (linear)
0 1 2 3 4 5 6 7
Contiguous
Visualization (matrix) → two indices (row & col)
Linear (N*M)
Row-major
0 1 2 3 4
0
1
2
3
rows = N (=4)
cols = M (=5)
Top Left → (0,0)
Top Right → (0, M-1)
Bottom left → (N-1, 0)
Bottom Right → (N-1, M-1)
Values/cell = N*M


Declaration & Initialization of 2D arrays
int [] arr = new int [size];
int [][] mat = new int [rows] [cols];
reference variable name
Heap Memory Allocation
Integer Multidimensional Arrays of 2 coordinates
give the total no of rows
" " of cols.


Time (DSA)
9:30 to 9:30 pm
Monday to Friday → Regular classes
Saturday → spill over + Revision + Extra Class + Doubts
Mock Interviews (Geeksforgeeks)
→ For any topic
8:20
DSA Doubts
Web Dev Doubts
Microsoft Teams
TA


Row by Row Printing
// Output 2D Array
for(int i=0; i<row; i++){
for(int j=0; j<col; j++){
System.out.print(mat[i][j] + " ");
}
System.out.println();
// To Print Different Rows in Different Lines
}
Rows = N
Cols = M
Values/cell = N*M
Col by Col Printing
// Print Matrix (Col by Col)
for(int j=0; j<col; j++){
for(int i=0; i<row; i++){
System.out.print(mat[i][j] + " ");
}
System.out.println();
// To Print Different Rows in Different Lines
}

Task 1: Print all elements sum

```
//          // Task 1: Sum of All Elements of Matrix
        int sum = 0;
        for(int i=0; i<row; i++){
            for(int j=0; j<col; j++){
                sum = sum + mat[i][j];
            }
        }
```

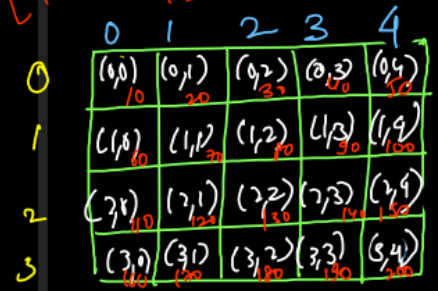
```
System.out.println("Sum of All Elements : " + sum);
```

Task 2: Print All Rows Sum

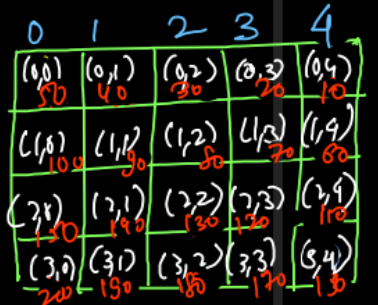
```
//          // Task 2: Sum of All Rows of Matrix
        for(int i=0; i<row; i++){
            int rowSum = 0;
            for(int j=0; j<col; j++){
                rowSum = rowSum + mat[i][j];
            }

            // Inside the (Row) outer loop, outside the (Col) inner loop
            System.out.println("Sum of " + i + " th Row : " + rowSum);
        }
```

Task 3: Print all Columns sum

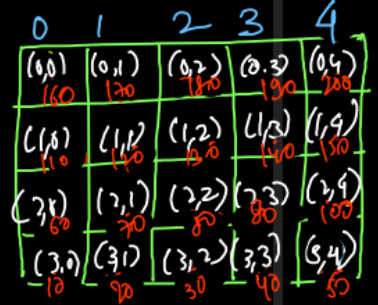
[4 rows, 5 columns] 4x5



① Reverse by matrix Row by Row → 4x5

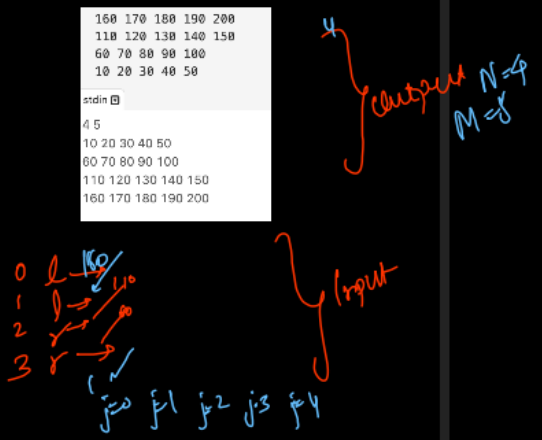


② Reverse the col by col → 4x5



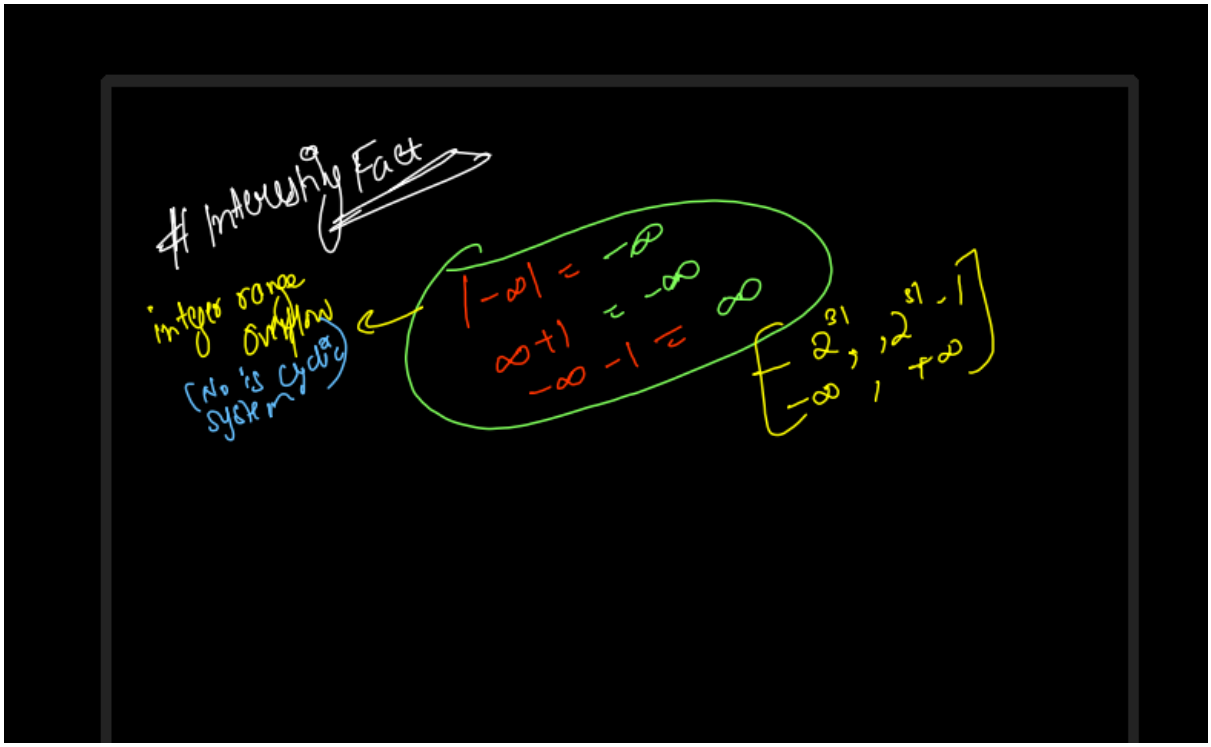
```
// Col by Col Reversal
for(int j=0; j<col; j++){

    // Reversing the ith Row
    int left = 0, right = row - 1;
    while(left < right){
        int temp = mat[left][j];
        mat[left][j] = mat[right][j];
        mat[right][j] = temp;
        left++; right--;
    }

}
```

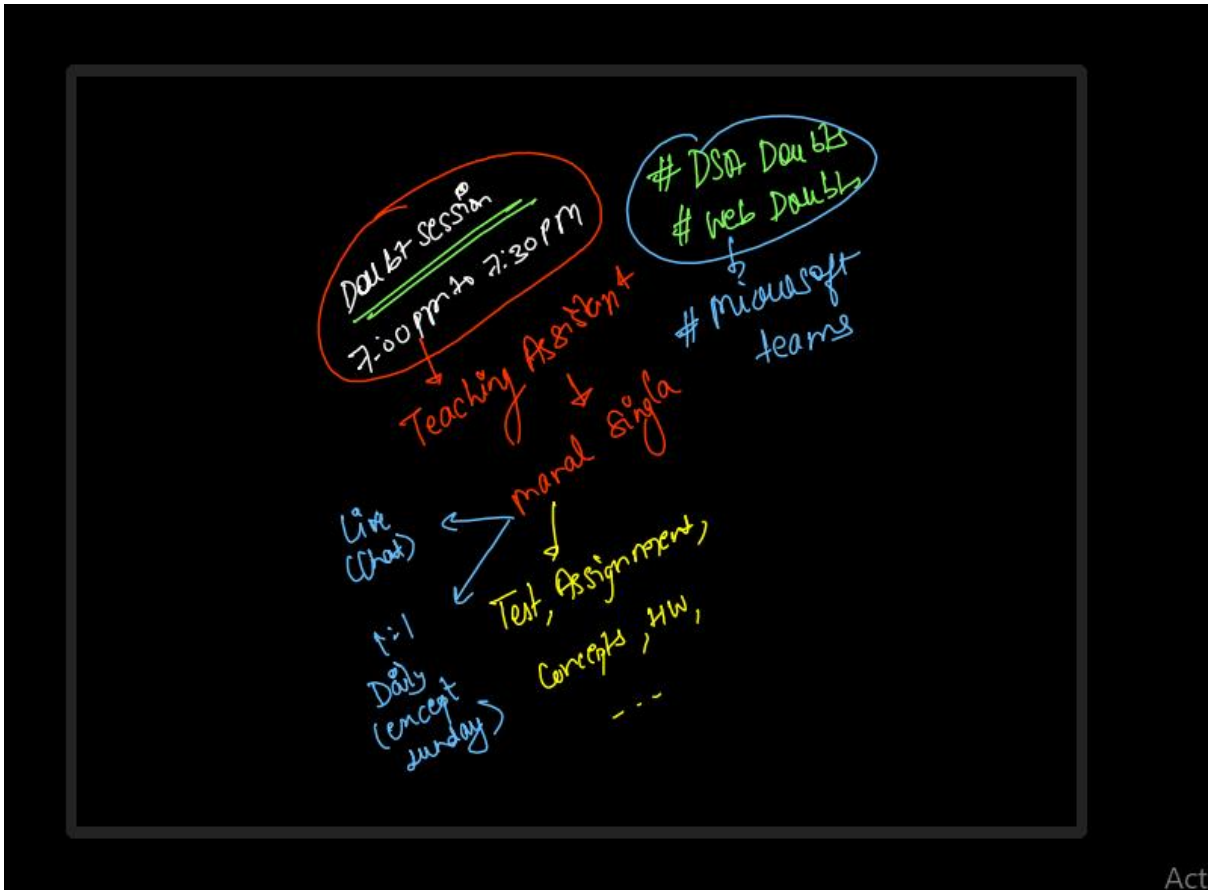
```
160 170 180 190 200
110 120 130 140 150
60 70 80 90 100
10 20 30 40 50

stdin
4 5
10 20 30 40 50
60 70 80 90 100
110 120 130 140 150
160 170 180 190 200
```

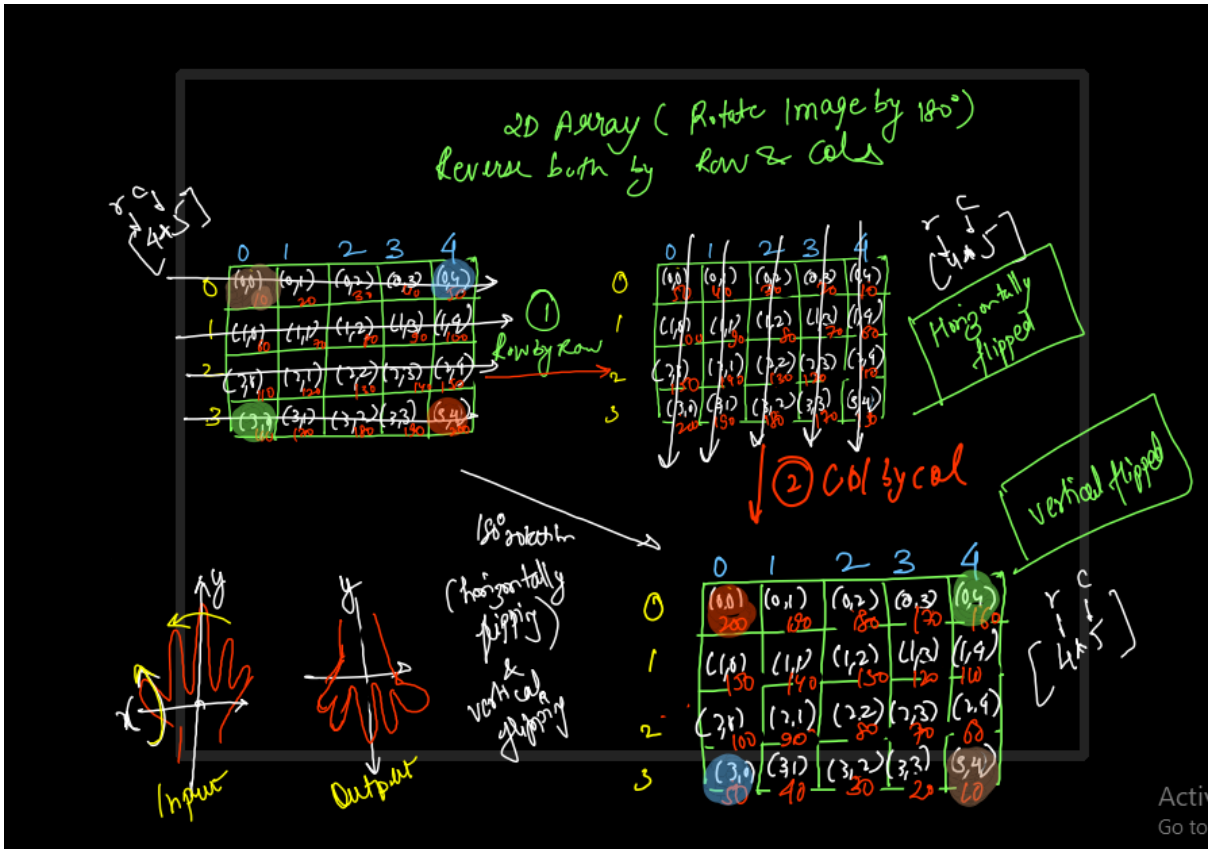


```
// Row by Row Reversal
for(int i=0; i<row; i++){

    // Reversing the ith Row
    int left = 0, right = col - 1;
    while(left < right){
        int temp = mat[i][left];
        mat[i][left] = mat[i][right];
        mat[i][right] = temp;
        left++; right--;
    }

}
```

```
Finished in 107 ms
10 20 30 40 50
60 70 80 90 100
110 120 130 140 150
160 170 180 190 200

50 40 30 20 10
100 90 80 70 60
150 140 130 120 110
200 190 180 170 160
```


Interesting Fact
integer range
Overflow
(No 'is cycle' system
|-∞| = -∞
∞+1 = -∞
-∞ -1 = ∞
[-2^31 , 2^31 - 1]
-∞ , +∞


Doubt session
7:00 pm to 7:30 pm
Teaching Assistant
Manal Singla
DSA Doubts
Web Doubts
Microsoft teams
Live (Chat)
1:1
Daily (except sunday)
Test, Assignment, Concepts, HW, ...


2D Array (Rotate Image by 180°)
Reverse both by row & cols
r c
[4x5]
0 1 2 3 4
0
1
2
3
1
Row by Row
Horizontally flipped
2 Col by col
Vertical flipped
180° rotation
(horizontally flipping & vertical flipping)
Input
Output
Activ
Go to

```
int row = mat.length, col = mat[0].length;
```

This syntax can be used if matrix is not empty

(HR)

```
// Step 1: Row by Row Reversal
for(int i=0; i<row; i++){

    int left = 0, right = col - 1;
    while(left < right){
        int temp = mat[i][left];
        mat[i][left] = mat[i][right];
        mat[i][right] = temp;

        left++; right--;
    }

}
```

+

(VR)

```
// Step 2: Column by Column Reversal
for(int j=0; j<col; j++){

    int left = 0, right = row - 1;
    while(left < right){
        int temp = mat[left][j];
        mat[left][j] = mat[right][j];
        mat[right][j] = temp;

        left++; right--;
    }

}
```

= 180° Rotation

Leetcode 48 Rotate by 90°C (clockwise)

1	2	3
4	5	6
7	8	9

7	4	1
8	5	2
9	6	3

N == M
rows = cols
Square matrix

8:20

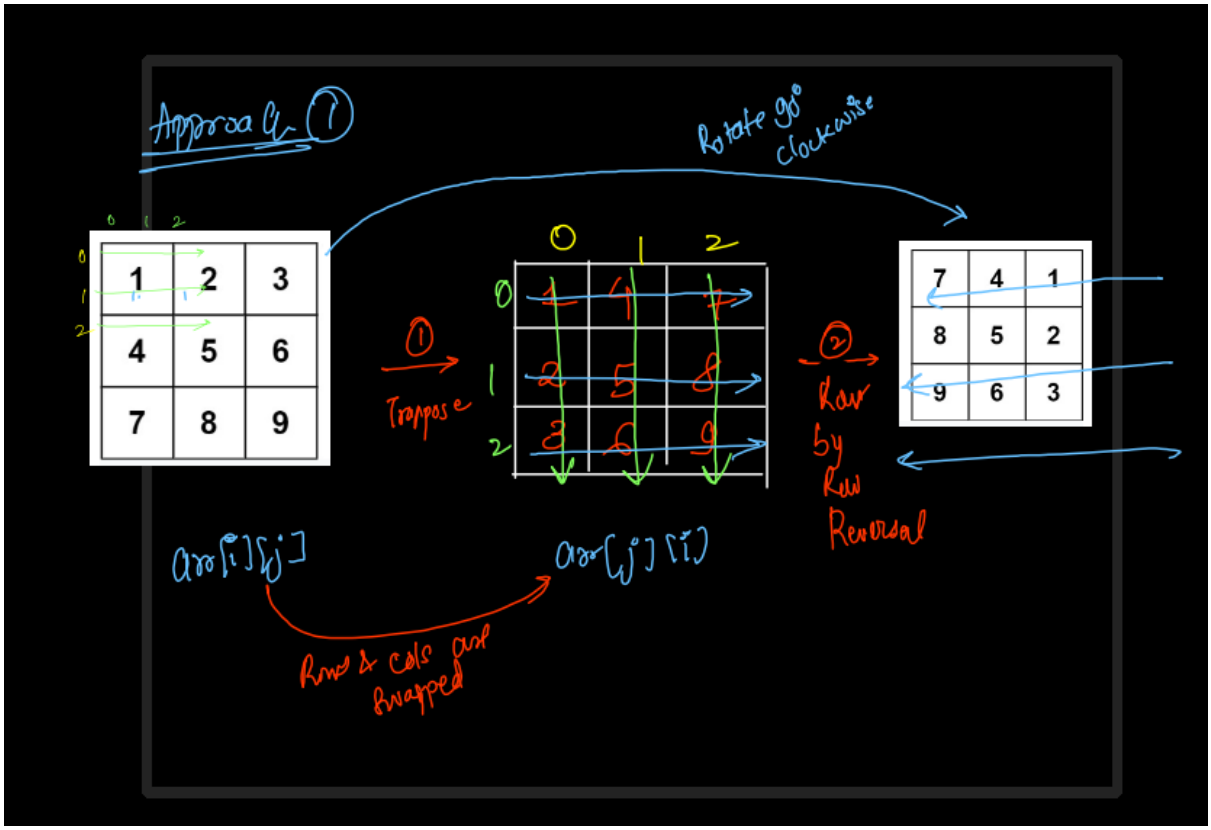
Input

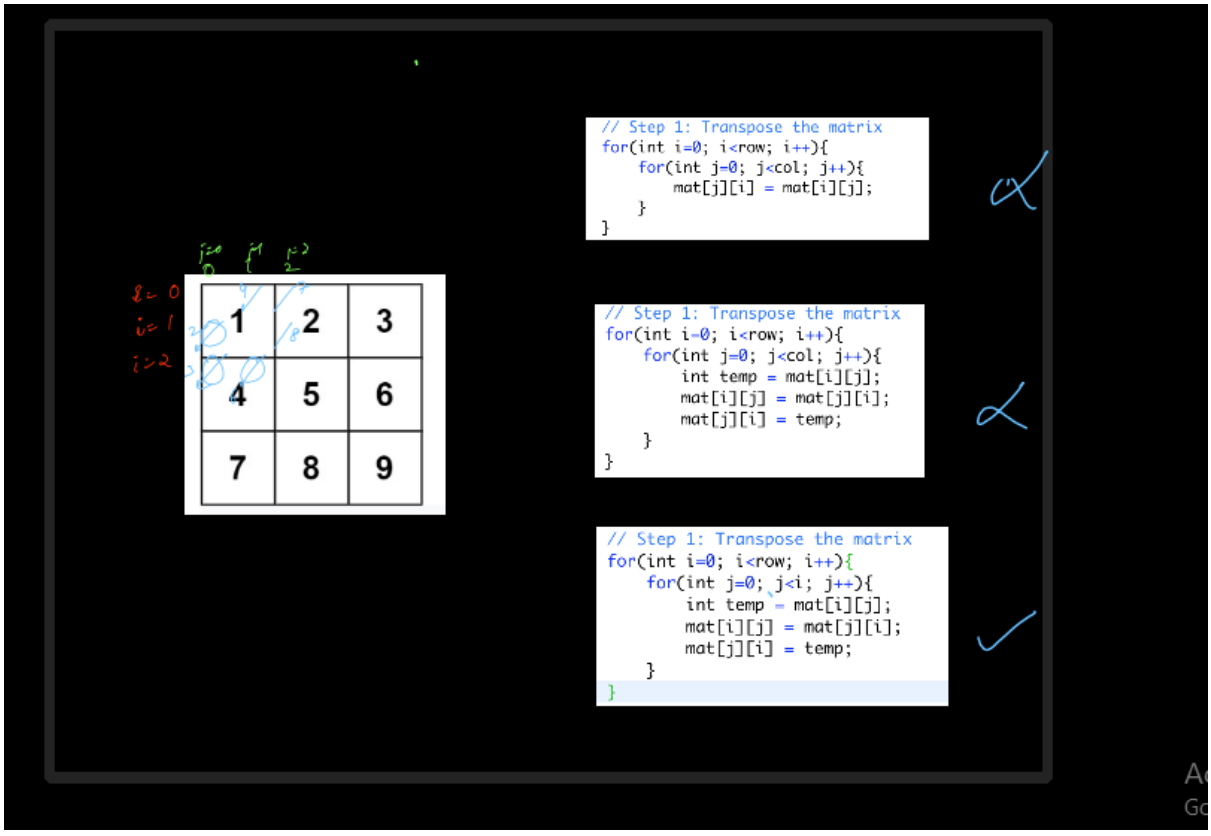
Output

R=4
C=5
R≠C

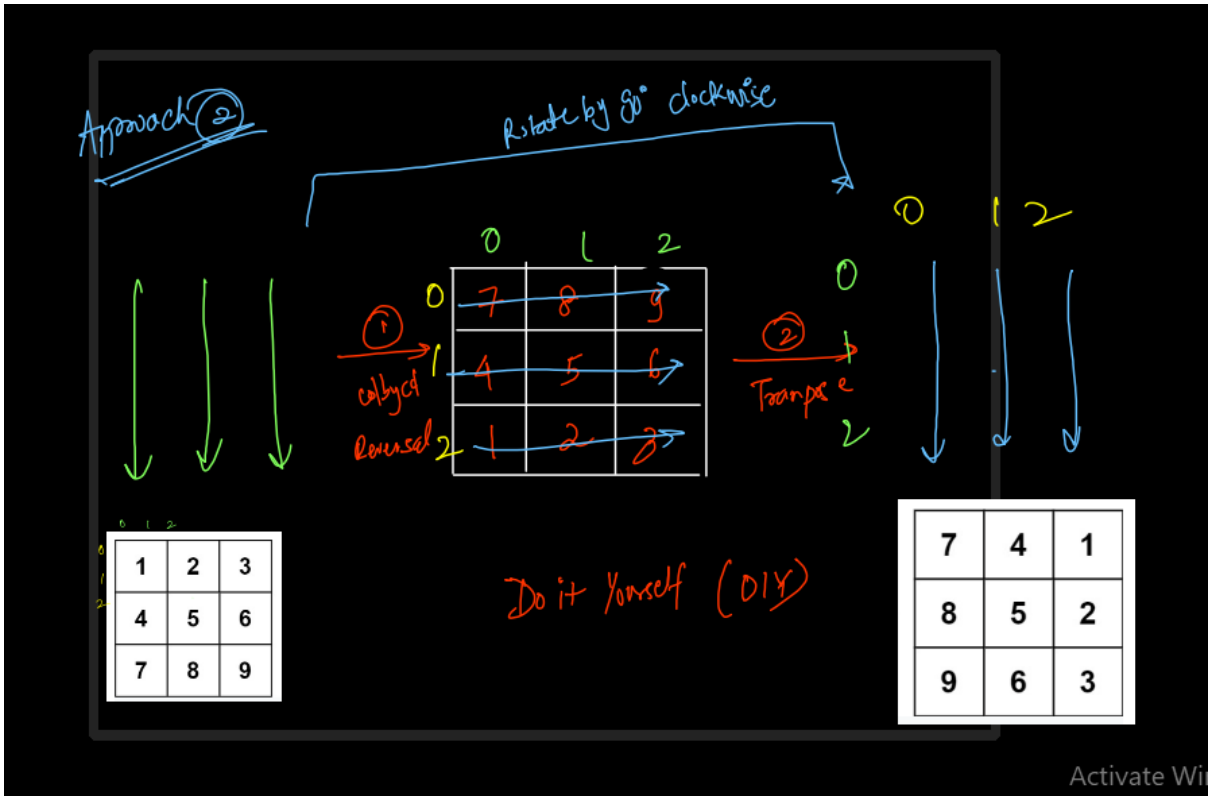
clockwise 90°

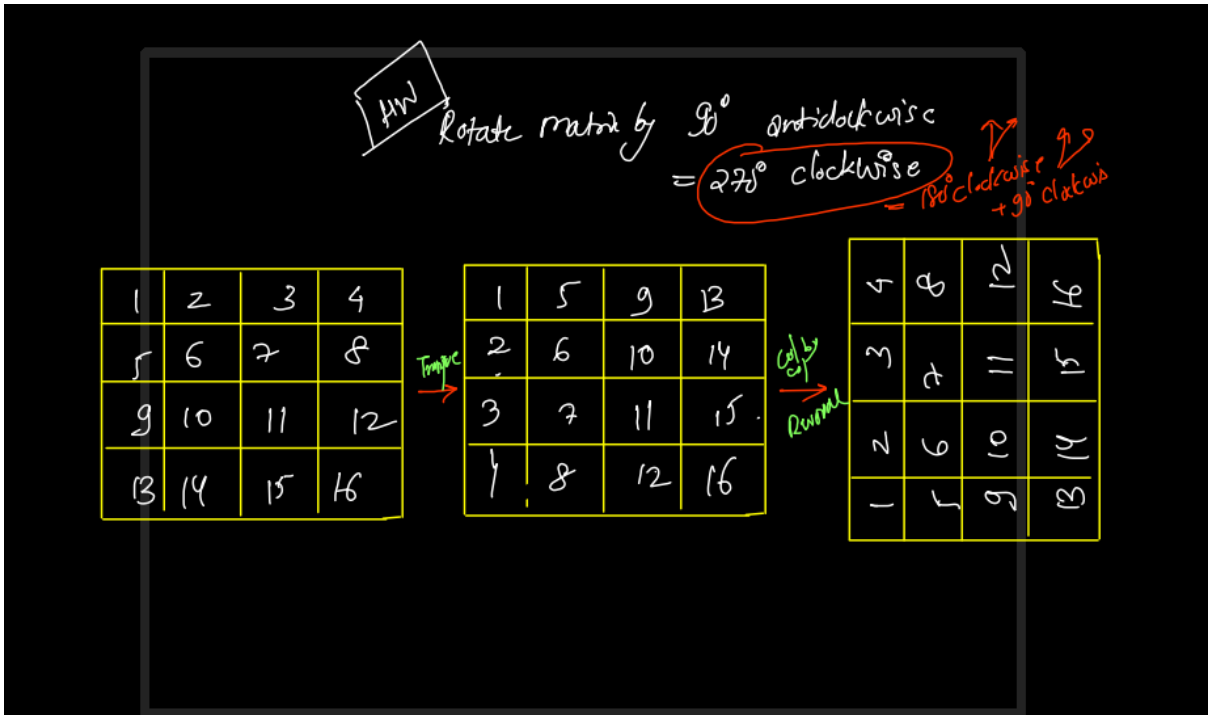
R=5
C=4


Approach 1
Rotate 90° clockwise
1 2 3
4 5 6
7 8 9
① Transpose
② Row by Row Reversal
7 4 1
8 5 2
9 6 3
arr[i][j]
arr[j][i]
Rows & cols are swapped


// Step 1: Transpose the matrix
for(int i=0; i<row; i++){
for(int j=0; j<col; j++){
mat[j][i] = mat[i][j];
}
}
// Step 1: Transpose the matrix
for(int i=0; i<row; i++){
for(int j=0; j<col; j++){
int temp = mat[i][j];
mat[i][j] = mat[j][i];
mat[j][i] = temp;
}
}
// Step 1: Transpose the matrix
for(int i=0; i<row; i++){
for(int j=0; j<i; j++){
int temp = mat[i][j];
mat[i][j] = mat[j][i];
mat[j][i] = temp;
}
}
1 2 3
4 5 6
7 8 9

```java
class Solution {
    public void rotate(int[][] mat) {
        int row = mat.length, col = mat[0].length;
        // Step 1: Transpose the matrix
        for(int i=0; i<row; i++){
            for(int j=0; j<i; j++){
                int temp = mat[i][j];
                mat[i][j] = mat[j][i];
                mat[j][i] = temp;
            }
        }

        // Step 2: Row by Row Reversal
        for(int i=0; i<row; i++){
            int left = 0, right = col - 1;
            while(left < right){
                int temp = mat[i][left];
                mat[i][left] = mat[i][right];
                mat[i][right] = temp;

                left++; right--;
            }
        }
    }
}
```


Approach 2
Rotate by 90° clockwise
0 1 2
0 7 8 9
1 4 5 6
2 1 2 3
1 colbycol Reversal
2 Transpose
1 2 3
4 5 6
7 8 9
Do it Yourself (DIY)
7 4 1
8 5 2
9 6 3
Activate Wi


HW
Rotate matrix by 90° anticlockwise
= 270° clockwise
= 180° clockwise + 90° clockwise
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Transpose
1 5 9 13
2 6 10 14
3 7 11 15
4 8 12 16
col by col Reverse
4 8 12 16
3 7 11 15
2 6 10 14
1 5 9 13

```java
// Check if sum of each row is same
int firstrowSum = 0;

for(int i=0; i<row; i++){

    int rowSum = 0;
    for(int j=0; j<col; j++){
        rowSum = rowSum + mat[i][j];
    }

    if(i == 0){
        firstrowSum = rowSum;
    } else {
        if(rowSum != firstrowSum)
            return false;
    }
}
```

return true;

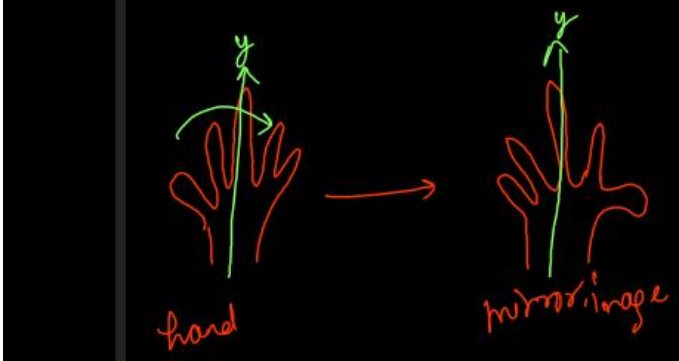
first row Sum = 10

i=0 | 2 | 3 | 4 ⇒ 10 (rowSum)

i=1 | 4 | 3 | 2 | 1 ⇒ 10 (rowSum)

i=2 | 1 | 1 | 2 | 3 ⇒ 7 (rowSum)

i=3 | 3 | 4 | 1 | 2 ⇒ 10



Reversing the row by row

Mirror the matrix

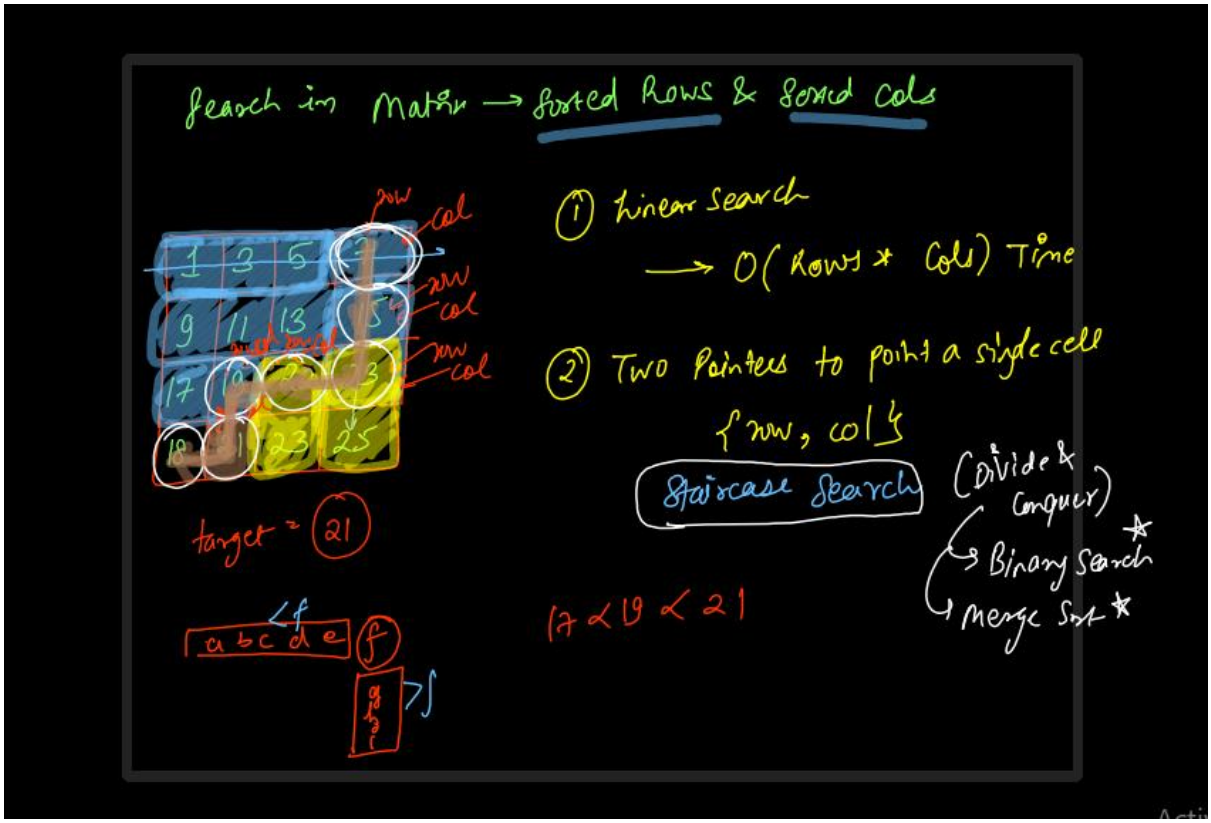
↓

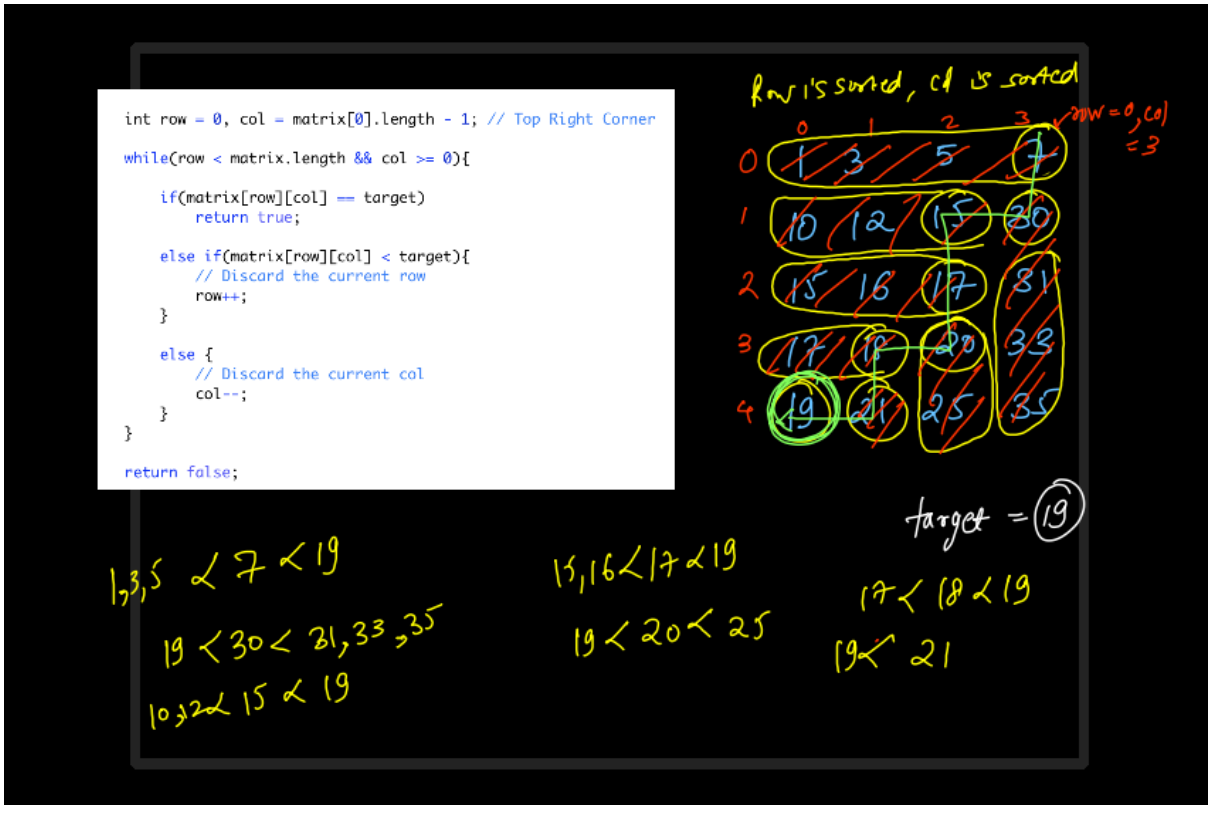
Row by Row Reversal

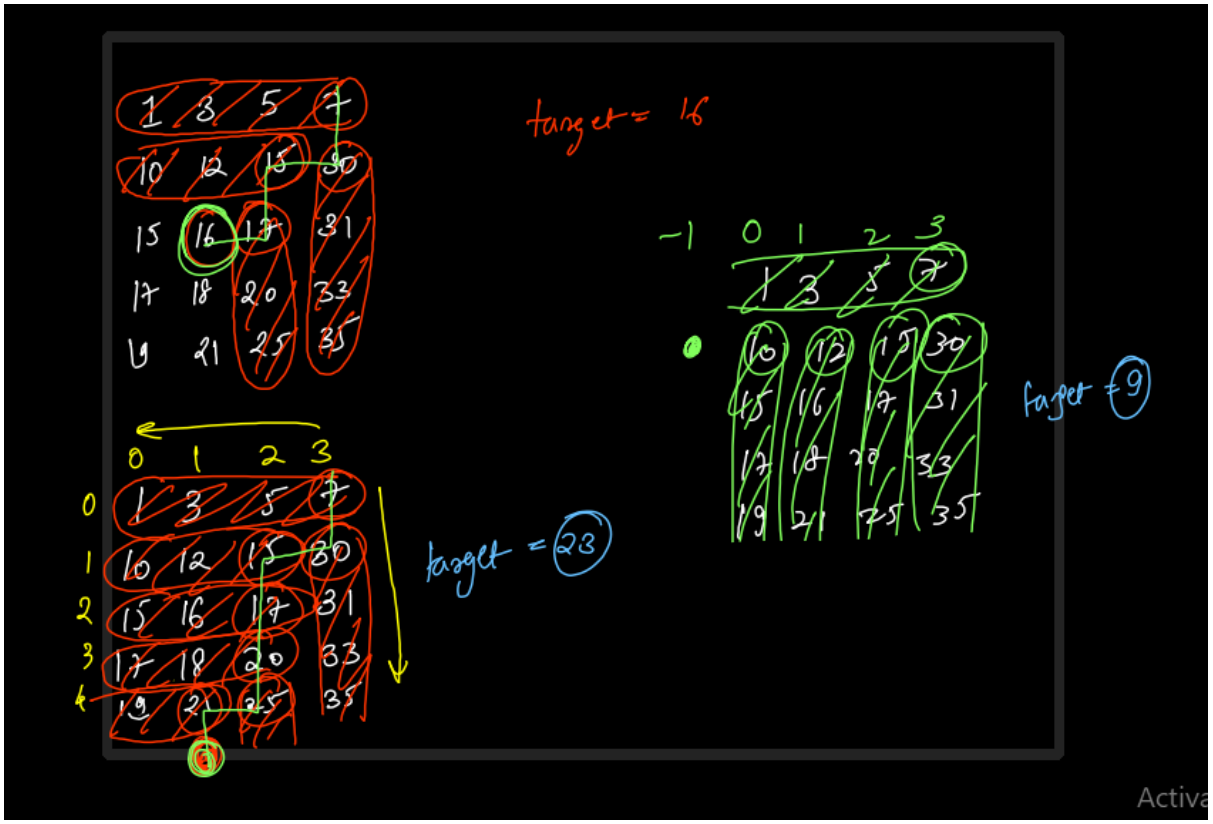
1 2 3 4 → 4 3 2 1

5 6 7 8 → 8 7 6 5

9 10 11 12 → 12 11 10 9


Search in Matrix → Sorted Rows & Sorted Cols
row
col
target = 21
① linear search
→ O(Rows * Cols) Time
② Two Pointers to point a single cell
{row, col}
Staircase Search
(Divide & Conquer)
Binary Search
Merge Sort
17 < 19 < 21


int row = 0, col = matrix[0].length - 1; // Top Right Corner
while(row < matrix.length && col >= 0){
if(matrix[row][col] == target)
return true;
else if(matrix[row][col] < target){
// Discard the current row
row++;
}
else {
// Discard the current col
col--;
}
}
return false;
Row is sorted, Col is sorted
row=0, col=3
target = 19
1,3,5 < 7 < 19
19 < 30 < 31, 33, 35
10, 12 < 15 < 19
15, 16 < 17 < 19
19 < 20 < 25
17 < 18 < 19
19 < 21



Iteration, Traversal, looping

visiting each element of array

1D → 1 loop

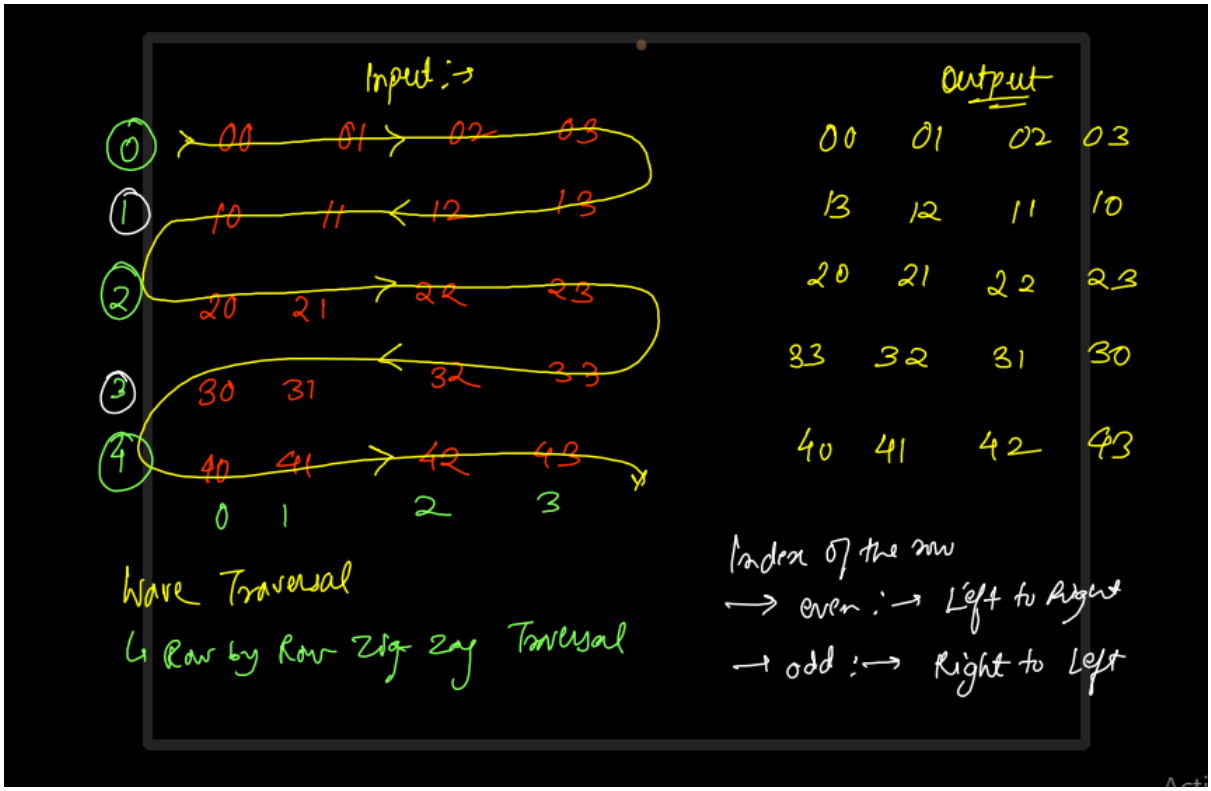
2D → 2 loops

3D → 3 loops

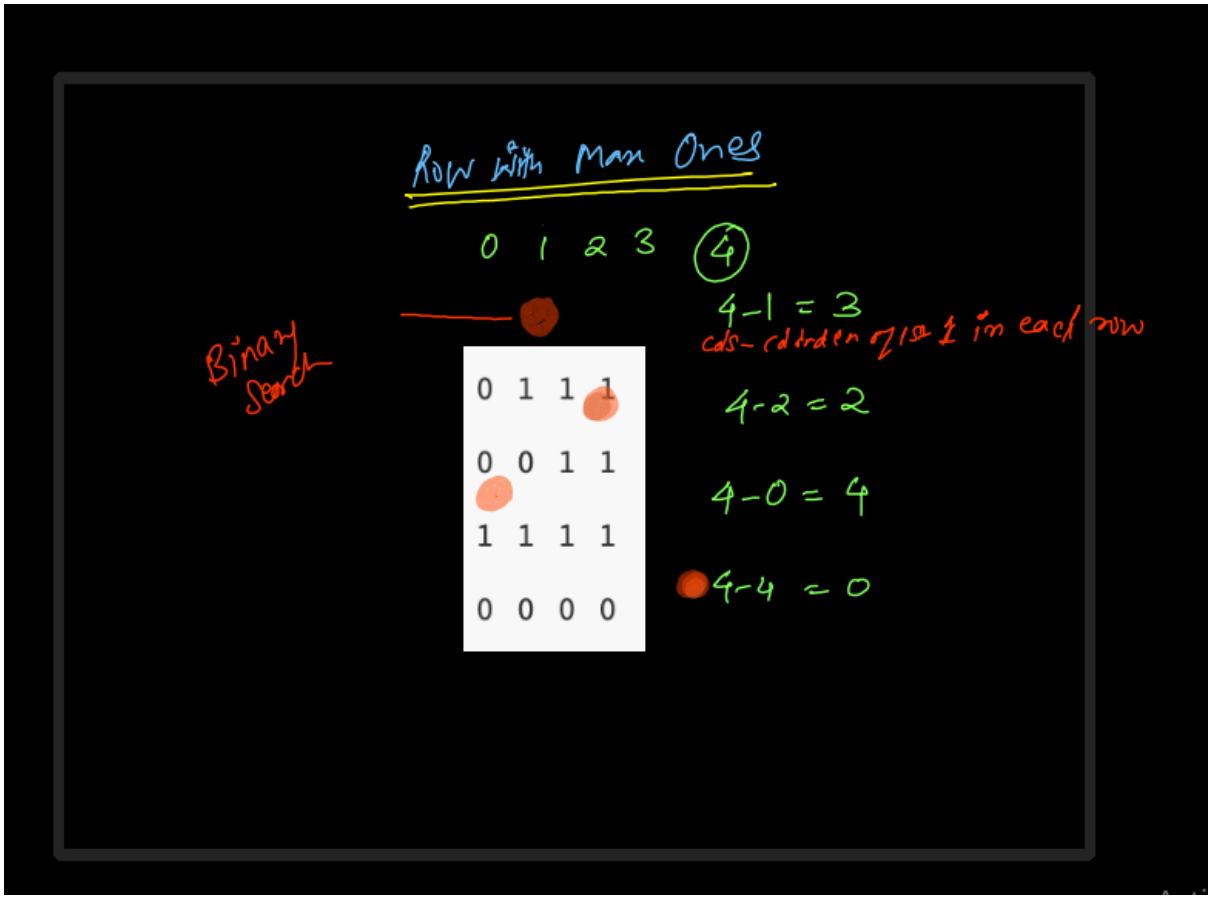
Inputting, outputting, Performing an operations

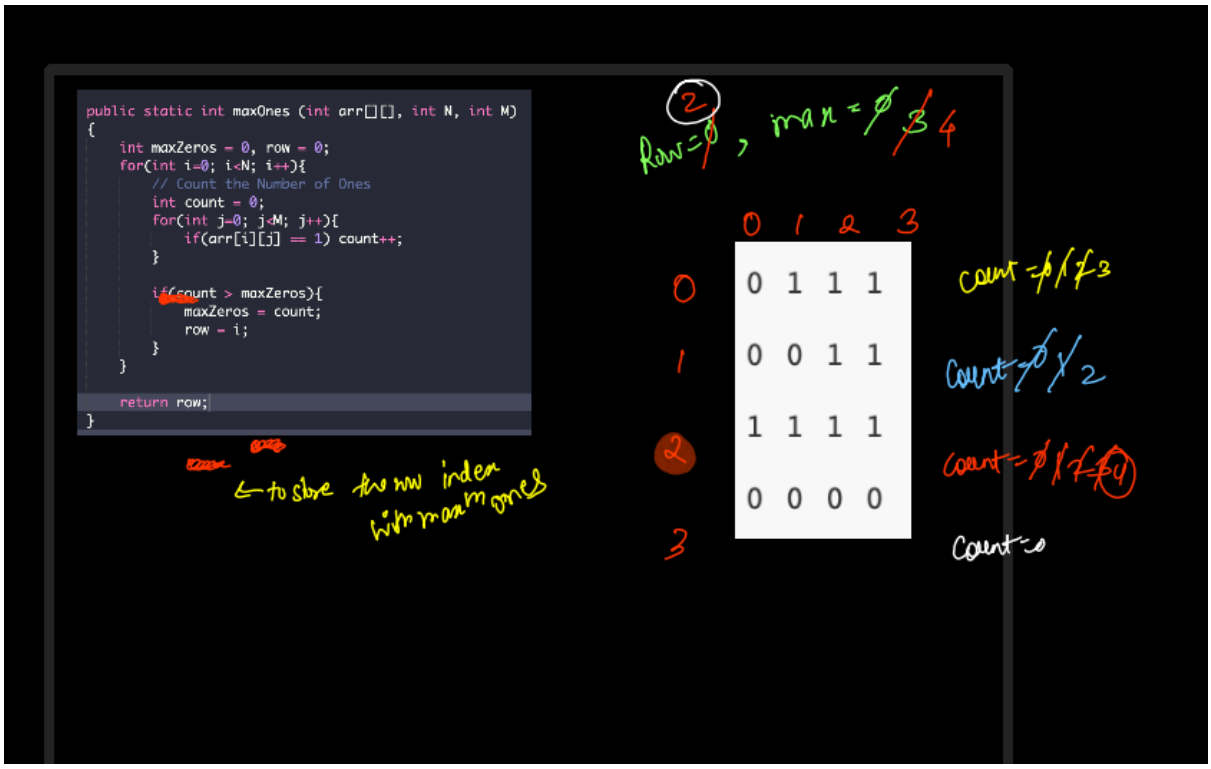
for(int i=0; i<n; i=i+2) ✗

for(int i=0; i<n; i++) ✓




```java
// Wave Traversal (Row by Row Zig-Zag Traversal)

for(int i=0; i<row; i++){

    if(i % 2 == 0){
        // Print Left to Right
        for(int j=0; j<col; j++){
            System.out.print(arr[i][j] + " ");
        }
    } else {
        // Print Right to Left
        for(int j=col-1; j>=0; j--){
            System.out.print(arr[i][j] + " ");
        }
    }

    System.out.println();
}
```


Row with Max Ones
0 1 2 3 4
Binary Search
0 1 1 1
0 0 1 1
1 1 1 1
0 0 0 0
4-1 = 3
cols - (index of 1st 1 in each row)
4-2 = 2
4-0 = 4
4-4 = 0


public static int maxOnes (int arr[][], int N, int M)
{
int maxZeros = 0, row = 0;
for(int i=0; i<N; i++){
// Count the Number of Ones
int count = 0;
for(int j=0; j<M; j++){
if(arr[i][j] == 1) count++;
}
if(count > maxZeros){
maxZeros = count;
row = i;
}
}
return row;
}
to store the row index with max ones
Row = 2, max = 4
0 1 2 3
0 1 1 1
0 0 1 1
1 1 1 1
0 0 0 0
count = 3
count = 2
count = 4
Count = 0

Contest

Rahul - 4
Nitin - 4
Vishal - 4
Sushant - 3
Jayraj - 3
Anil
Shivam
Drishti

Elite Batch → Serious students

Better Placement
1. Live attendance
2. Doubts
3. Tests
4. Assignments
5. Mock Interviews
6. Soft skills

Multiplication of 3 Nos as maximum

eg1: 5, 0, 2, 6, 0, 7, 9, 1, 3 → If all are positive take 3 max nos

Max Product = 9 * 7 * 6

eg2: -11, -10, 5, 6, 7, 0

Max product = -11 * -10 * 7

eg3: -11, -10, -9, -8, -7, -6, -5

Max → ① 3 max
② 1 max, 2 min

```
Arrays.sort(arr); // Inbuilt sorting

int max1 = arr[n - 1], max2 = arr[n - 2], max3 = arr[n - 3];
int prod1 = max1 * max2 * max3;

int min1 = arr[0], min2 = arr[1];
int prod2 = min1 * min2 * max1;

System.out.println(Math.max(prod1, prod2));
```

Maxm of 3 Nos

Sorting → O(NlogN)

→ O(N)

eg1 0, 0, 1, 1, 2, 2, 2, 3, 4, 5, 6, 7, 8, 9

7*8*9

eg2 −7, −6, −5, −5, −4, −3, −2, −1

−1 * −2 * −3 = −6

eg3 −8, −7, −5, 0, 4, 6, 10

−8 * −7 * 10 < 560

* 40

eg4 −5 −4 0 3 6 7

2 * 7 = 140

3 * 6 * 7 = 126


```
int max1 = Integer.MIN_VALUE, max2 = Integer.MIN_VALUE, max3 = Integer.MIN_VALUE;
int min1 = Integer.MAX_VALUE, min2 = Integer.MAX_VALUE;

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

    if(arr[i] >= max1){
        max3 = max2;
        max2 = max1;
        max1 = arr[i];
    }
    else if(arr[i] >= max2){
        max3 = max2;
        max2 = arr[i];

    } else if(arr[i] >= max3){
        max3 = arr[i];
    }


    if(arr[i] <= min1){
        min2 = min1;
        min1 = arr[i];
    } else if(arr[i] <= min2){
        min2 = arr[i];
    }
}

System.out.println(Math.max(max1 * max2 * max3, min1 * min2 * max1));
```

4 7 −2 −9 8 5 1 −10

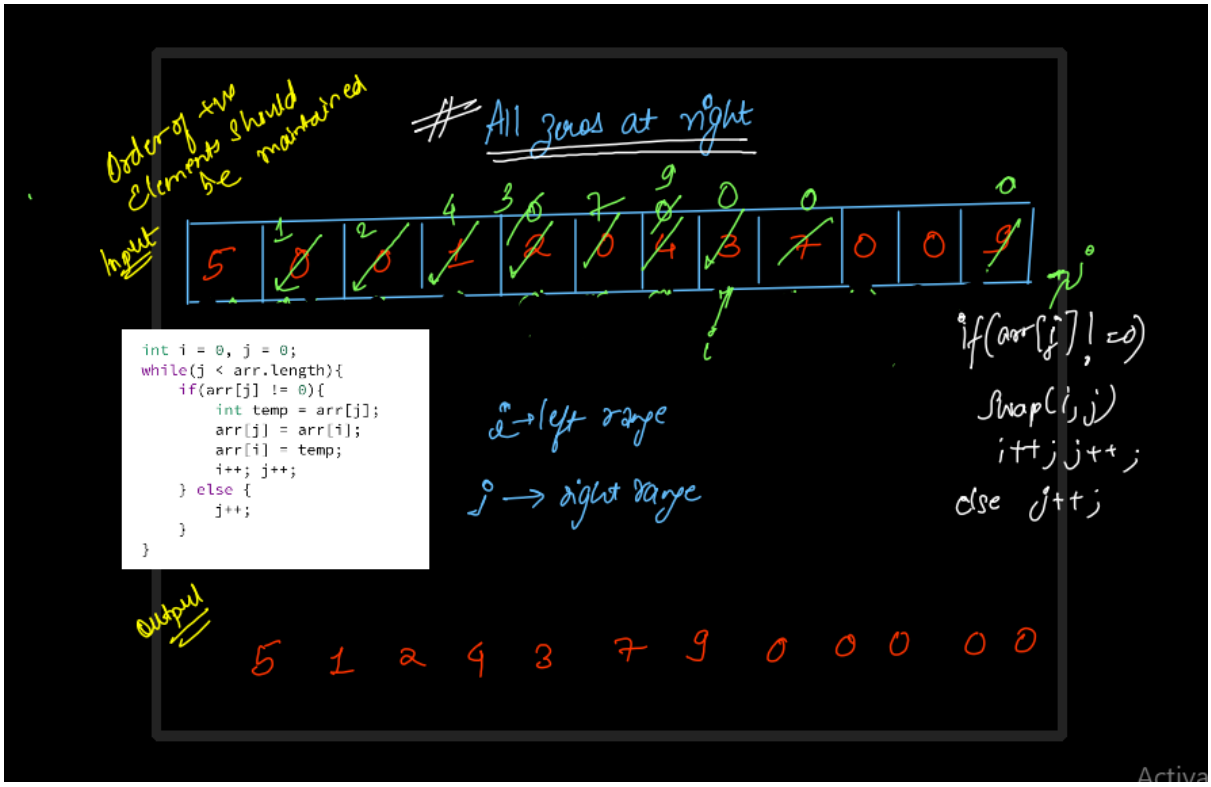
max1 = −∞ → 8

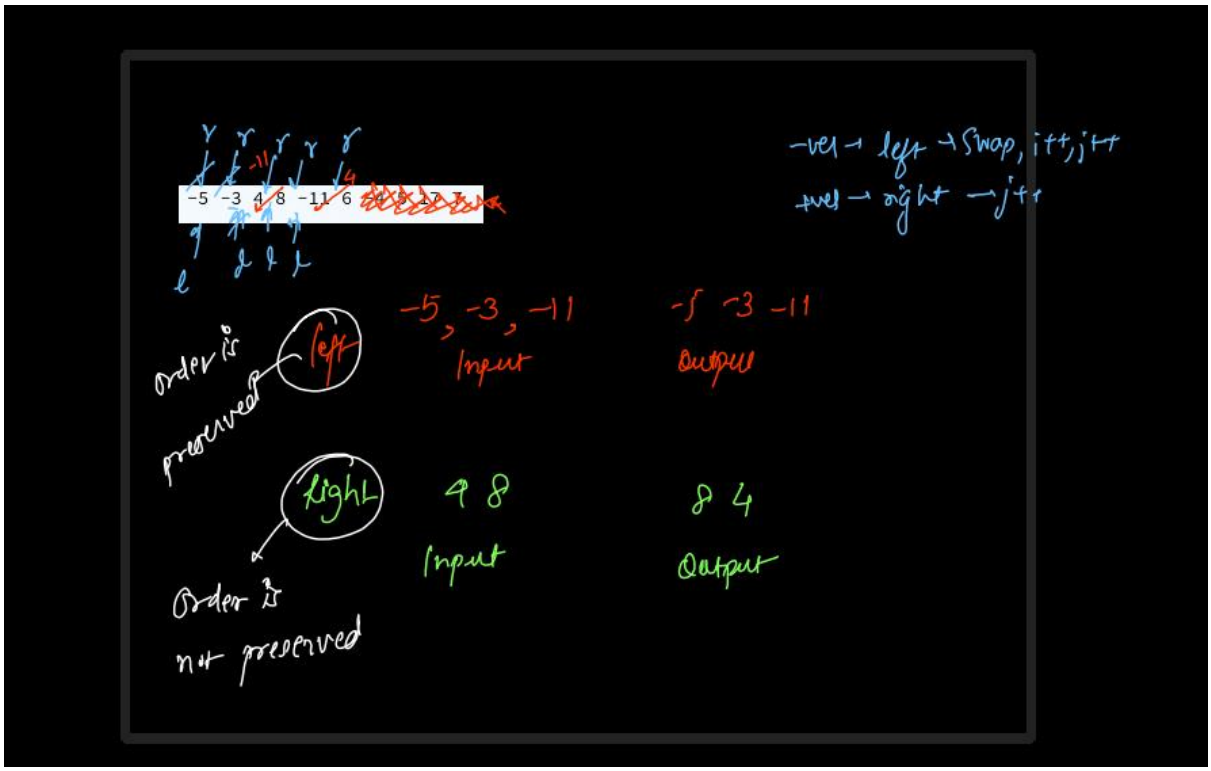
max2 = −∞ → 7

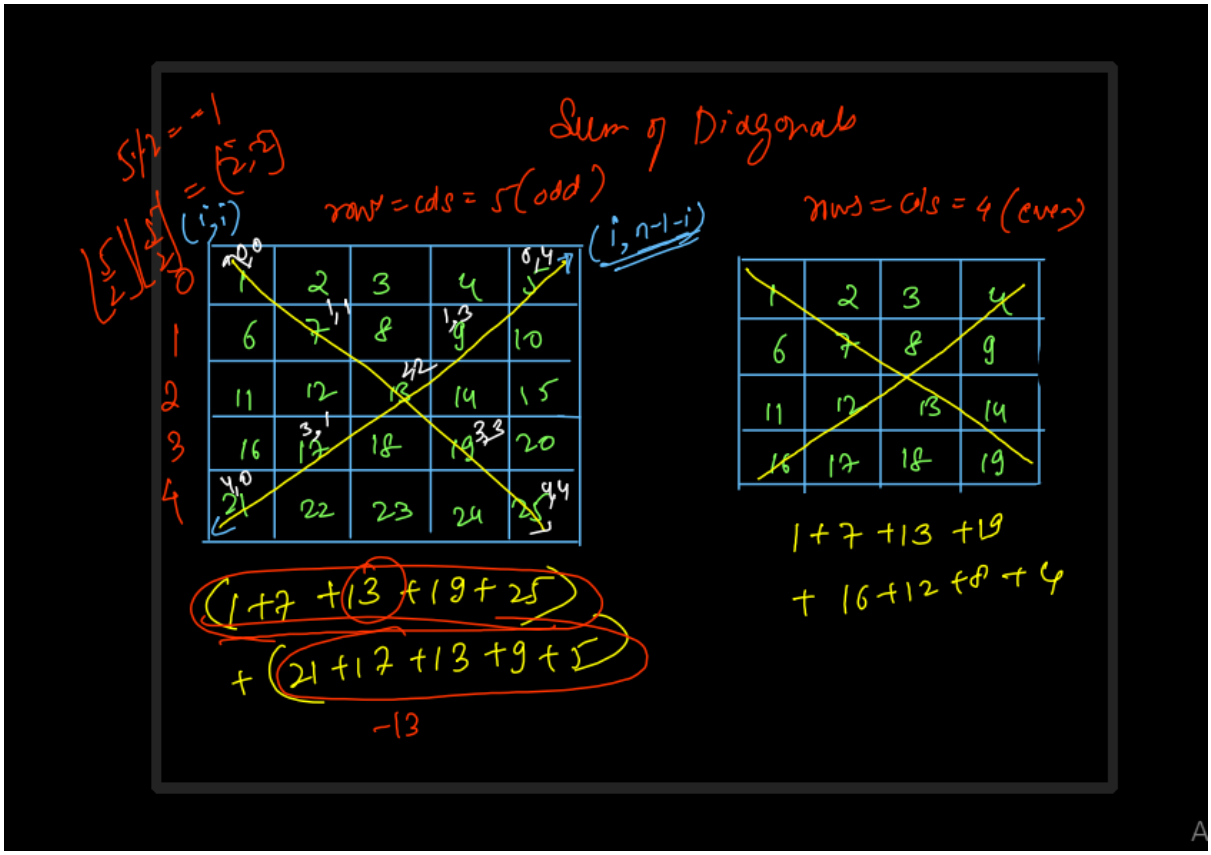
max3 = −∞ → 5

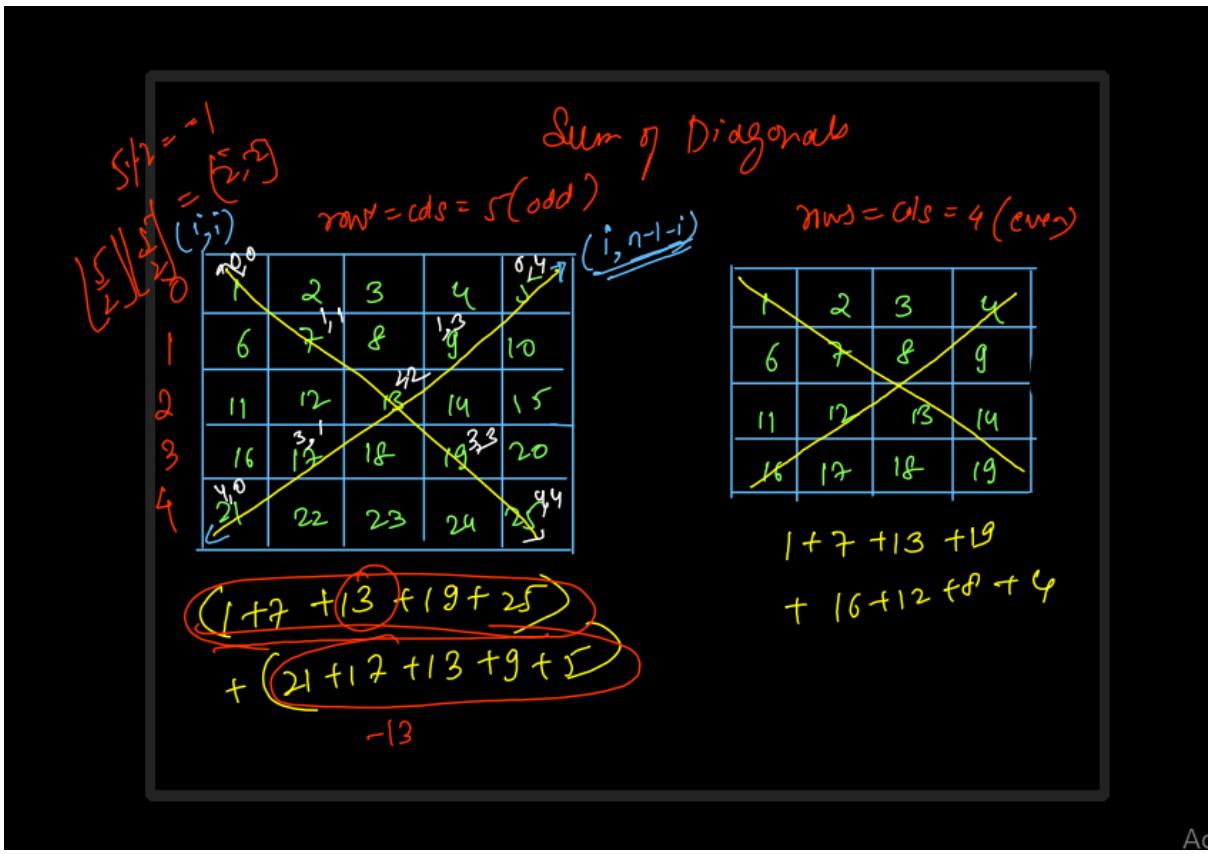
min1 = +∞ → −2 → −10

min2 = +∞ → 7 → −2 → −9


Order of +ve Elements should be maintained
All zeros at right
Input
5 1 2 4 3 7 9 0 0 0 0 0
i
j
int i = 0, j = 0;
while(j < arr.length){
if(arr[j] != 0){
int temp = arr[j];
arr[j] = arr[i];
arr[i] = temp;
i++; j++;
} else {
j++;
}
}
i → left range
j → right range
if(arr[j] != 0)
Swap(i,j)
i++; j++;
else j++;
Output
5 1 2 4 3 7 9 0 0 0 0 0


-5 -3 4 8 -11 6
-ve → left → Swap, i++, j++
+ve → right → j++
Order is preserved
left
-5, -3, -11
Input
-5 -3 -11
Output
right
4 8
Input
8 4
Output
Order is not preserved


Sum of Diagonals
rows = cols = 5 (odd)
rows = cols = 4 (even)
(i, n-1-i)
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
1 7 13 19
+ 25 17 13 9 1
- 13
1 2 3 4
6 7 8 9
11 12 13 14
16 17 18 19
1 + 7 + 13 + 19
+ 16 + 12 + 8 + 4


Sum of Diagonals
rows = cols = 5 (odd)
rows = cols = 4 (even)
(i, n-1-i)
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
1 7 13 19
+ 25 17 13 9 1
- 13
1 2 3 4
6 7 8 9
11 12 13 14
16 17 18 19
1 + 7 + 13 + 19
+ 16 + 12 + 8 + 4

```java
int sum = 0;

// TOP LEFT TO BOTTOM RIGHT
for(int i=0; i<n; i++){
    sum = sum + mat[i][i];
}

// TOP RIGHT TO BOTTOM LEFT
for(int i=0; i<n; i++){
    sum = sum + mat[i][n - 1 - i];
}

if(n % 2 == 1) sum = sum - mat[n/2][n/2];

System.out.println(sum);
```

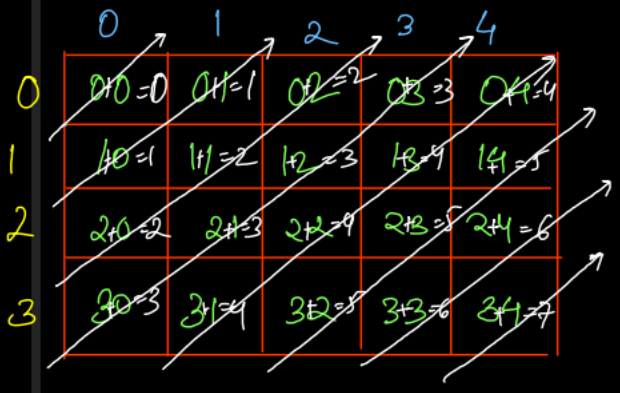
Doubt sessions

7:00 pm to 7:30 pm

(Monday to Friday)

Teams group? → Web Development

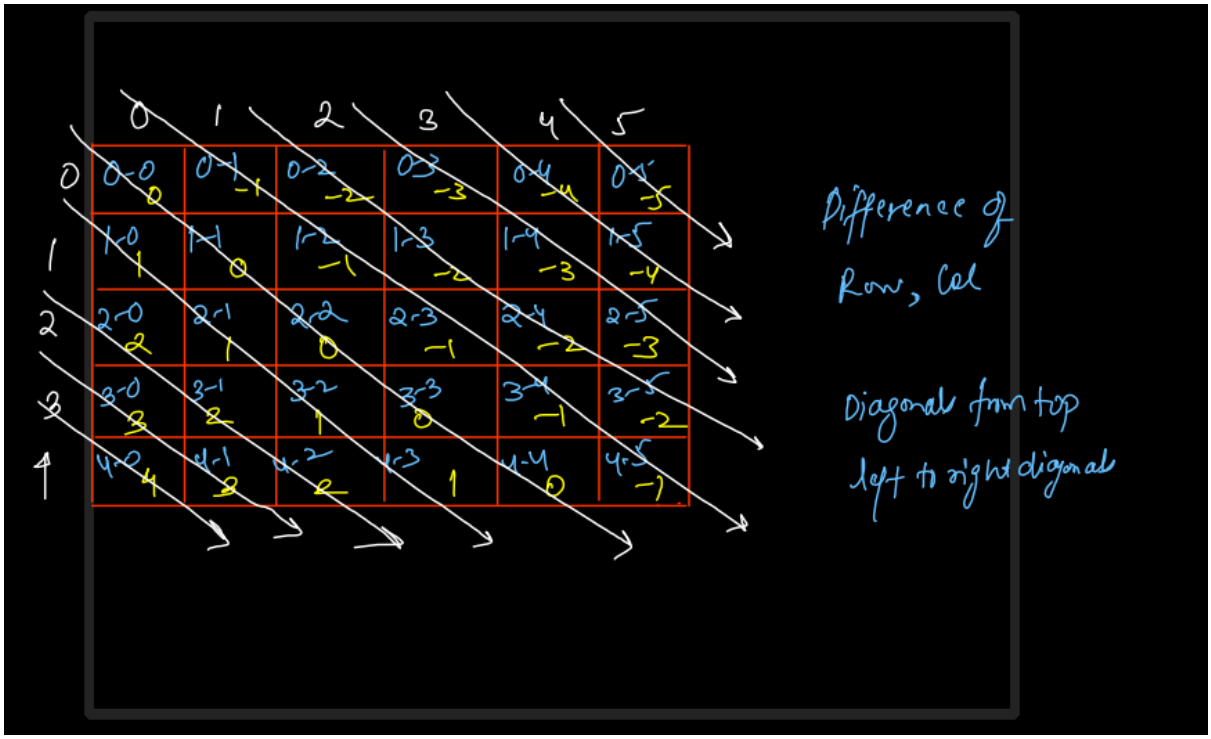
DSA - Java Doubts



Indexes

Sum of (Row, Col)

Diagonals from bottom - left to top - right

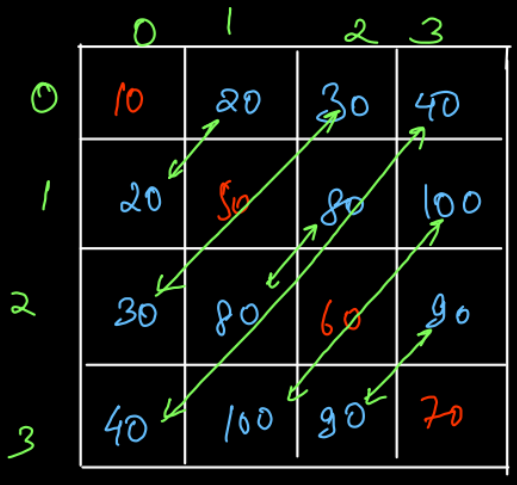



```java
int row = scn.nextInt(), col = scn.nextInt();
// Sum of Row, Col Index
for(int i=0; i<row; i++){
    for(int j=0; j<col; j++){
        System.out.print(i + j + " ");
    }
    System.out.println();
}

System.out.println();

// Difference of Row, Col Index
for(int i=0; i<row; i++){
    for(int j=0; j<col; j++){
        System.out.print(i - j + " ");
    }
    System.out.println();
}
```

Type ① BL to TR

Type ② TL to BR

Symmetric Matrix → true / false



Lower Left Triangle == Upper Right Triangle

⇒ Matrix = Transpose of Matrix

$\forall i, j \quad a[i][j] = a[j][i]$

Square matrix → rows == cols

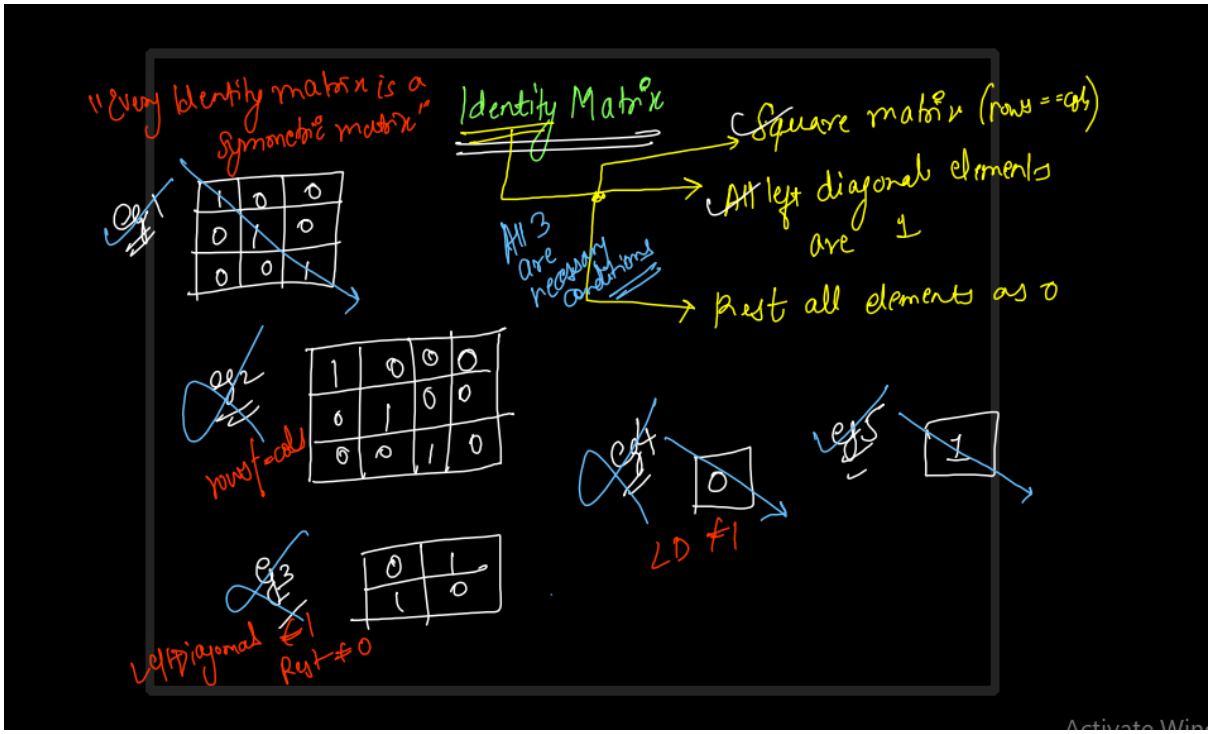
---

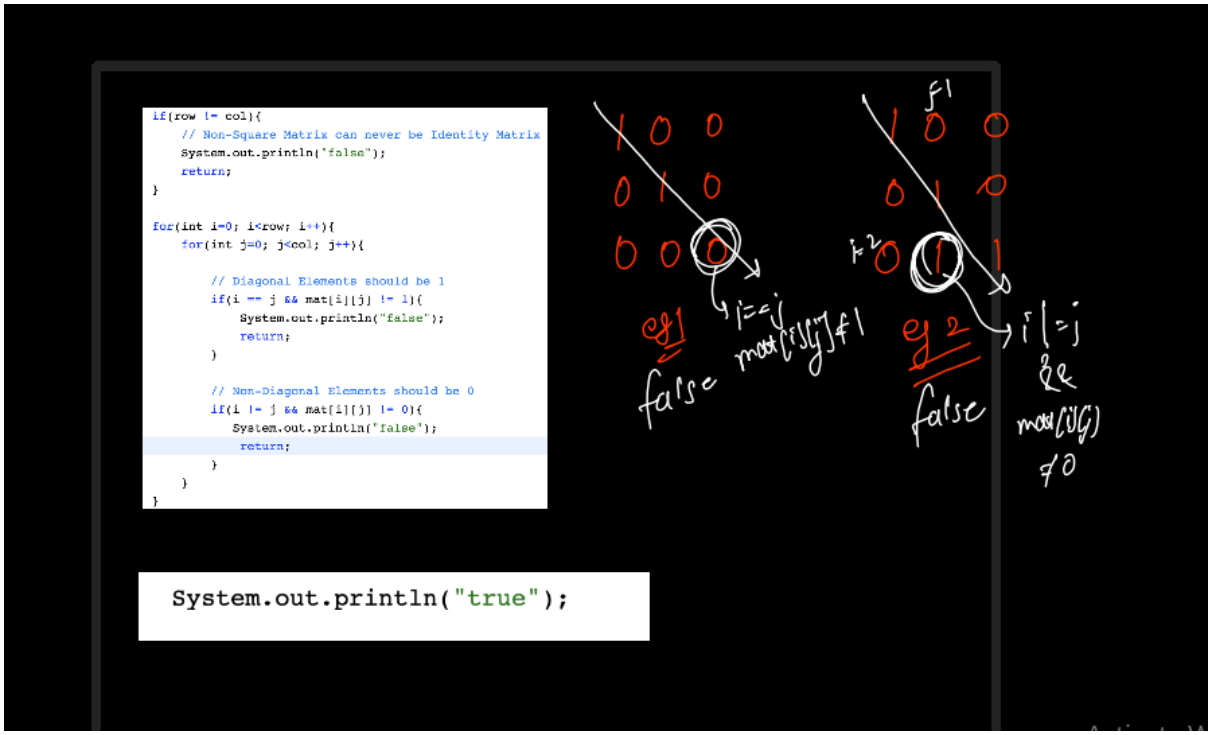
① ✗ Square, ✗ Symmetric

② ✓ Square, ✗ Symmetric

③ ✓ Square, ✓ Symmetric

④ ✗ Square, ✓ Symmetric } This is not possible.

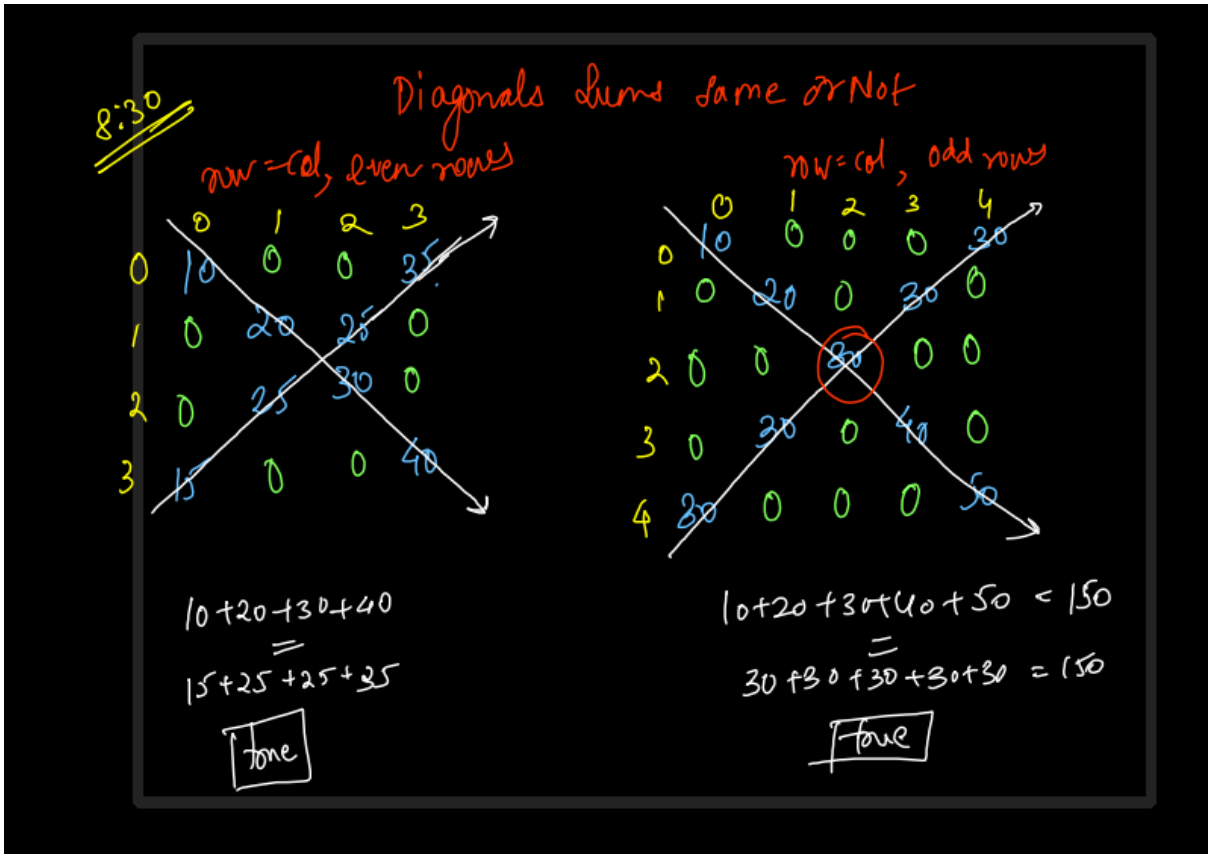




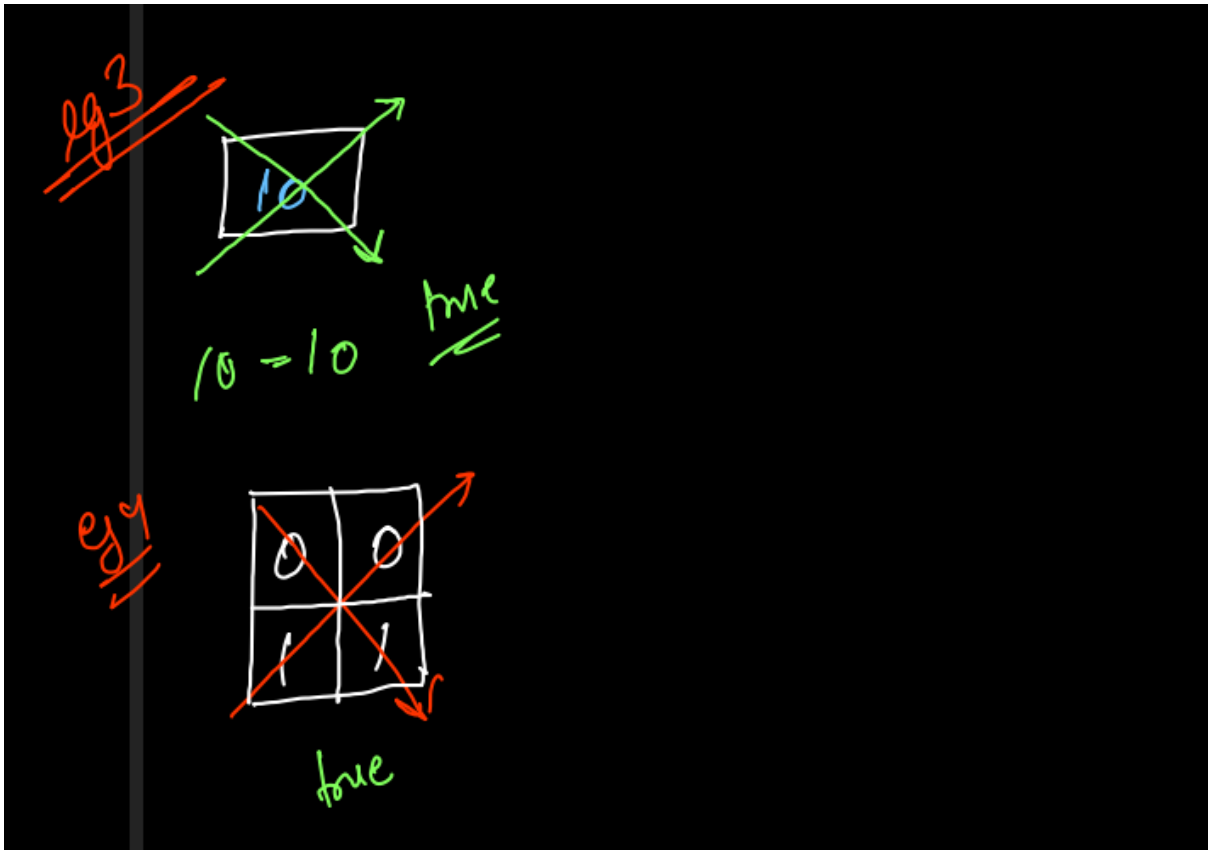
```java
if(row != col){
    // Non-Square Matrix can never be Identity Matrix
    System.out.println("false");
    return;
}

for(int i=0; i<row; i++){
    for(int j=0; j<col; j++){

        // Diagonal Elements should be 1
        if(i == j && mat[i][j] != 1){
            System.out.println("false");
            return;
        }

        // Non-Diagonal Elements should be 0
        if(i != j && mat[i][j] != 0){
            System.out.println("false");
            return;
        }
    }
}
```

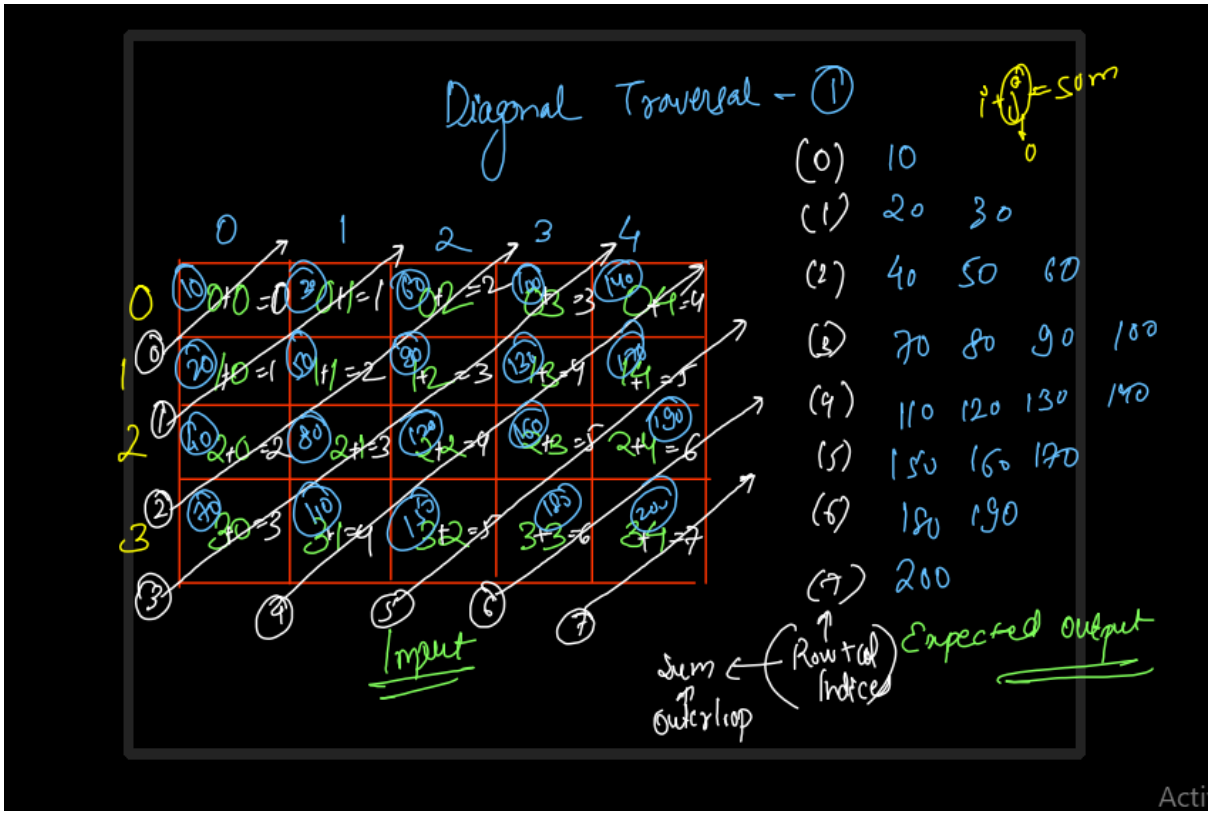


```java
System.out.println("true");
```



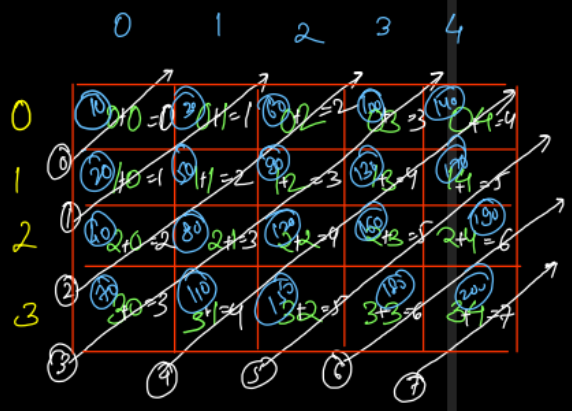

```java
// Check Diagonals Sum are equal or not (for Square Matrix)
int leftSum = 0;
for(int i=0; i<row; i++){
    leftSum = leftSum + mat[i][i];
}

int rightSum = 0;
for(int i=0; i<row; i++){
    rightSum = rightSum + mat[i][row - 1 - i];
}

if(leftSum == rightSum) System.out.println("true");
else System.out.println("false");
```


eg 3
10
10 = 10
true
eg 4
0 0
1 1
true


Diagonal Traversal - 1
i+j=sum
(0) 10
(1) 20 30
(2) 40 50 60
(3) 70 80 90 100
(4) 110 120 130 140
(5) 150 160 170
(6) 180 190
(7) 200
Input
sum = Row+Col Indices
outerloop
Expected output

Medium – Hard

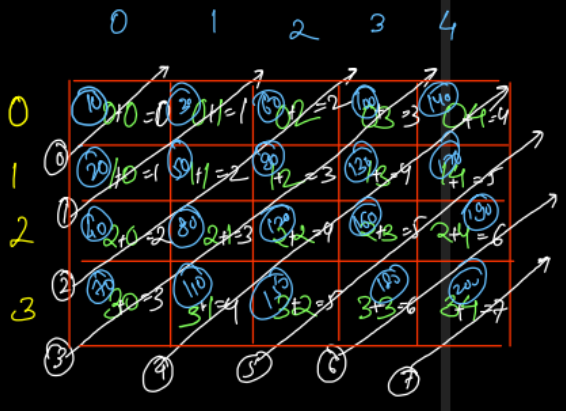
```java
// Outer Loop -> Sum of Row + Col -> Picking Diagonals
for(int sum = 0; sum <= (row - 1 + col - 1); sum++){

    // If the Diagonal Starts from the left Wall
    int i = sum, j = 0;

    // If the Diagonal Starts from Bottom Wall
    if(sum >= row){
        i = row - 1;
        j = sum - i;
    }

    while(i >= 0 && j <= col-1){
        System.out.print(mat[i][j] + " ");
        i--; j++;
    }

    System.out.println();
}
```



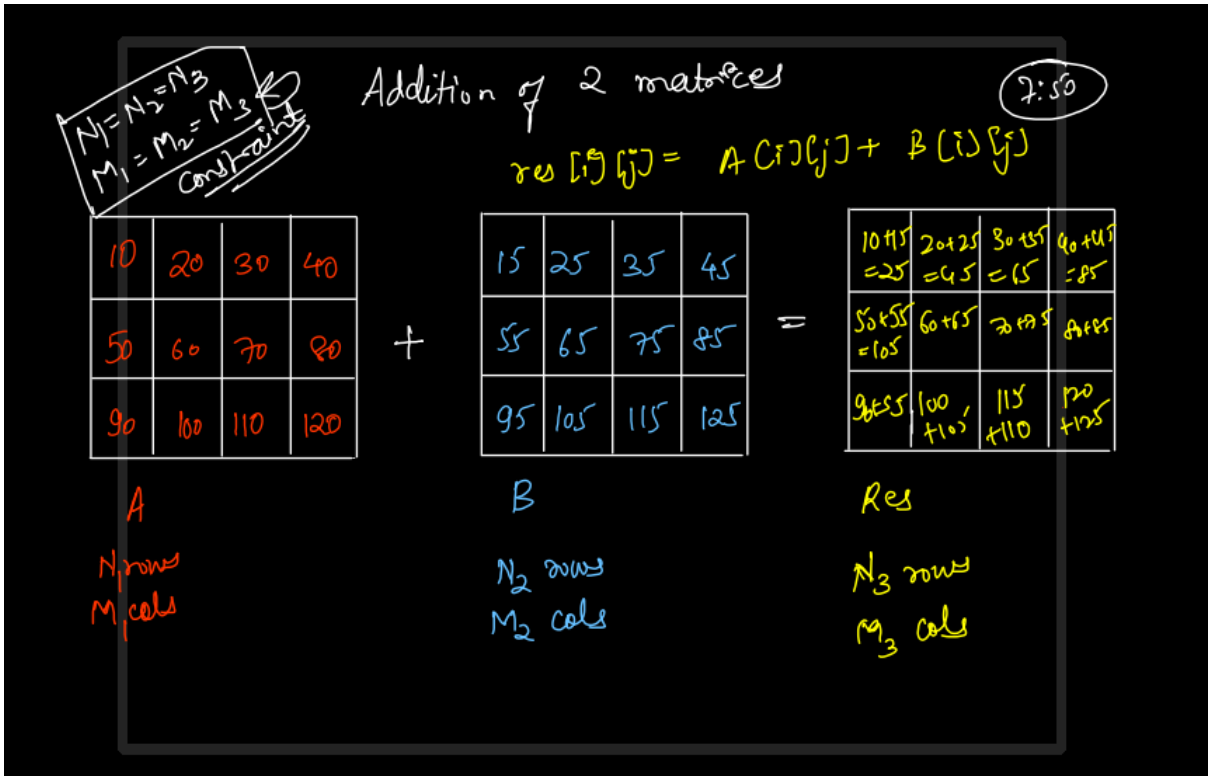
Diagonal Traversal

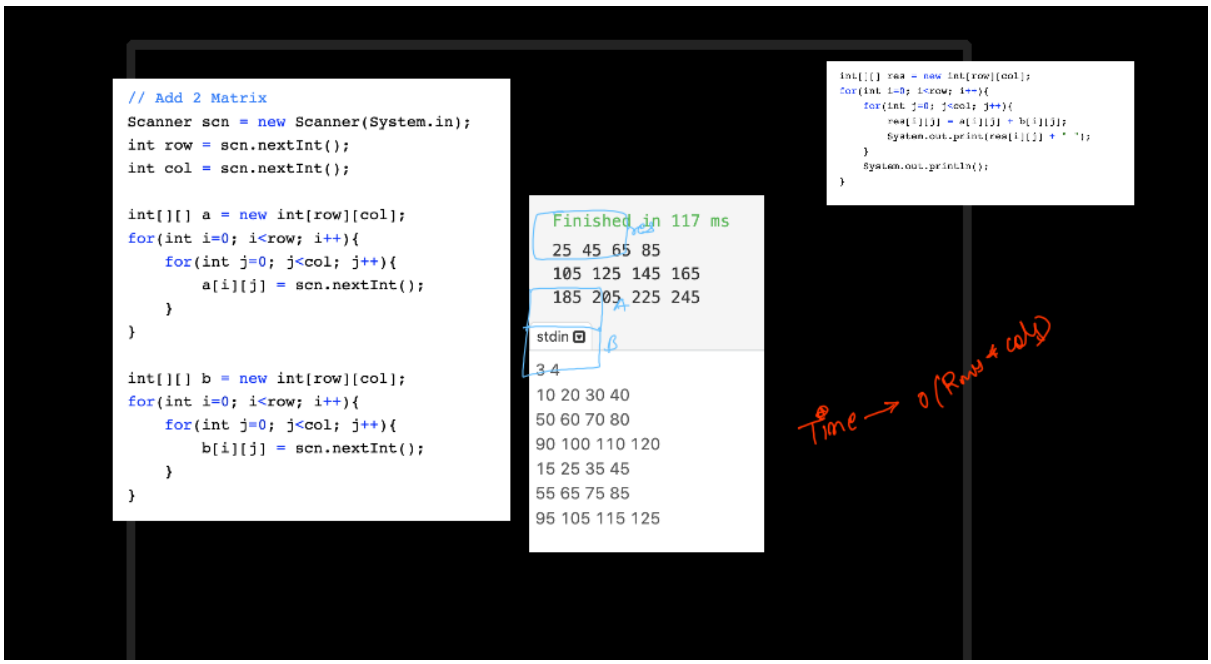
Medium – Hard

```java
// Outer Loop -> Sum of Row + Col -> Picking Diagonals
for(int sum = 0; sum <= (row - 1 + col - 1); sum++){

    // If the Diagonal Starts from the left Wall
    int i = sum, j = 0;

    // If the Diagonal Starts from Bottom Wall
    if(sum >= row){
        i = row - 1;
        j = sum - i;
    }

    while(i >= 0 && j <= col-1){
        System.out.print(mat[i][j] + " ");
        i--; j++;
    }

    System.out.println();
}
```



Diagonal Traversal



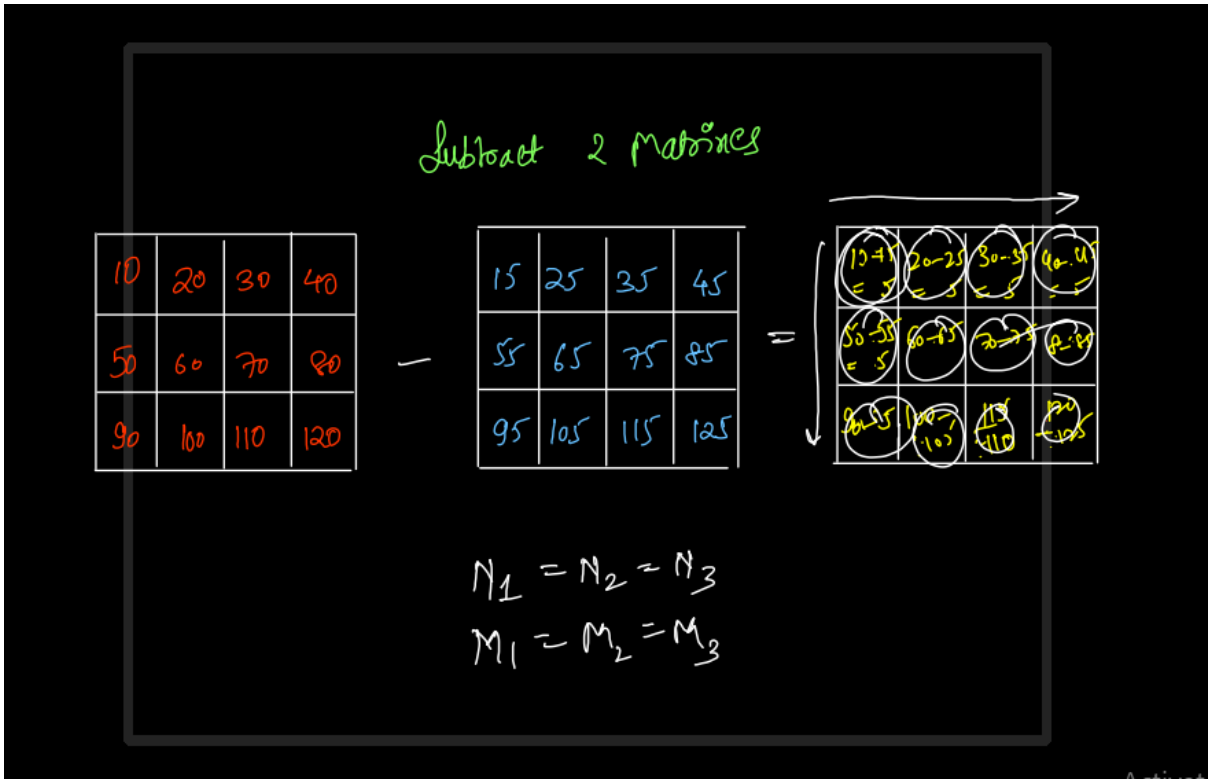


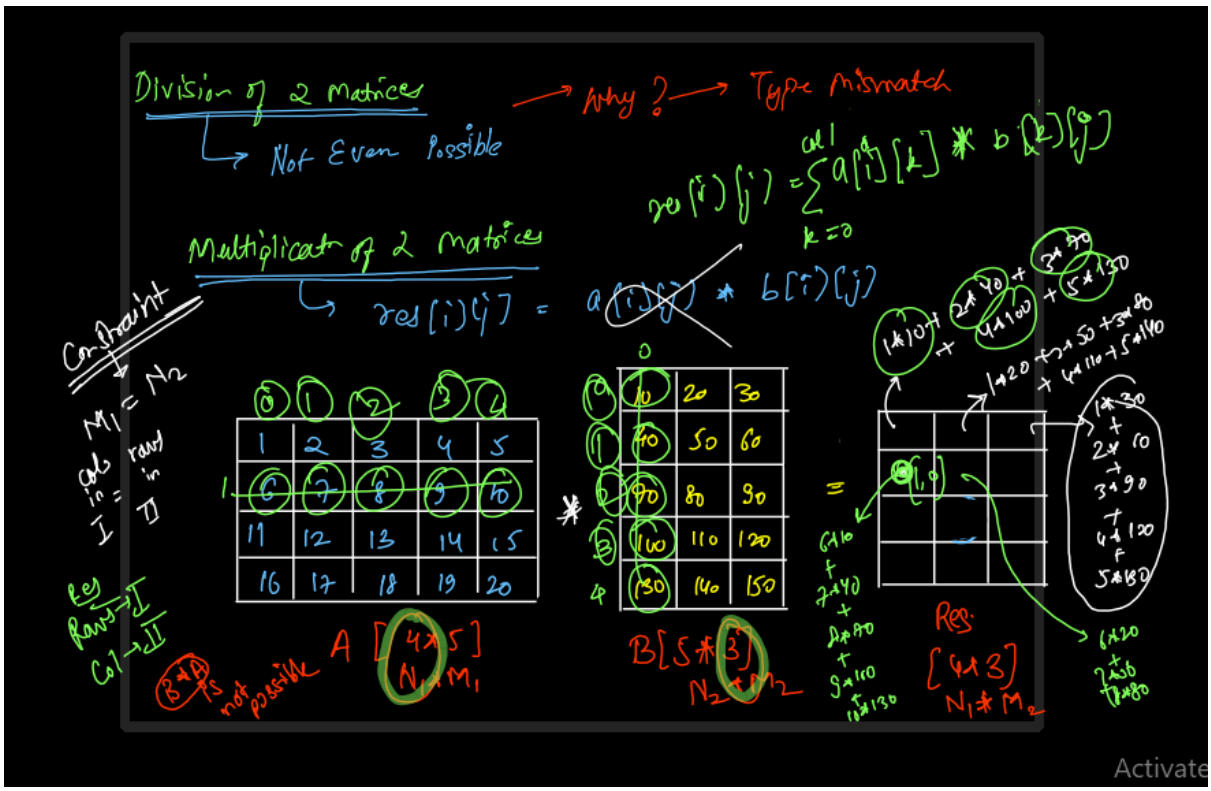
```java
// Add 2 Matrix
Scanner scn = new Scanner(System.in);
int row = scn.nextInt();
int col = scn.nextInt();

int[][] a = new int[row][col];
for(int i=0; i<row; i++){
    for(int j=0; j<col; j++){
        a[i][j] = scn.nextInt();
    }
}

int[][] b = new int[row][col];
for(int i=0; i<row; i++){
    for(int j=0; j<col; j++){
        b[i][j] = scn.nextInt();
    }
}
```

```java
int[][] res = new int[row][col];
for(int i=0; i<row; i++){
    for(int j=0; j<col; j++){
        res[i][j] = a[i][j] + b[i][j];
        System.out.print(res[i][j] + " ");
    }
    System.out.println();
}
```

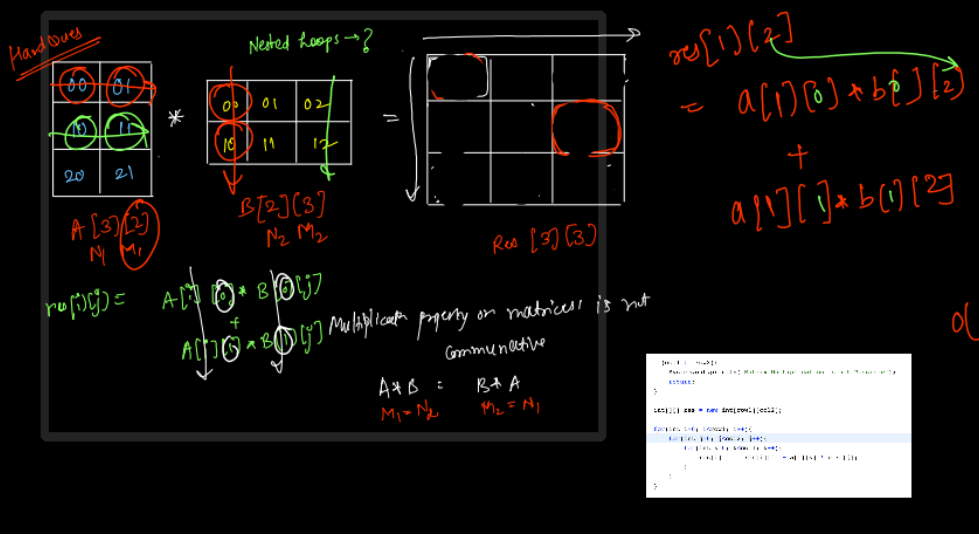
```
Finished in 117 ms
25 45 65 85
105 125 145 165
185 205 225 245
stdin
3 4
10 20 30 40
50 60 70 80
90 100 110 120
15 25 35 45
55 65 75 85
95 105 115 125
```

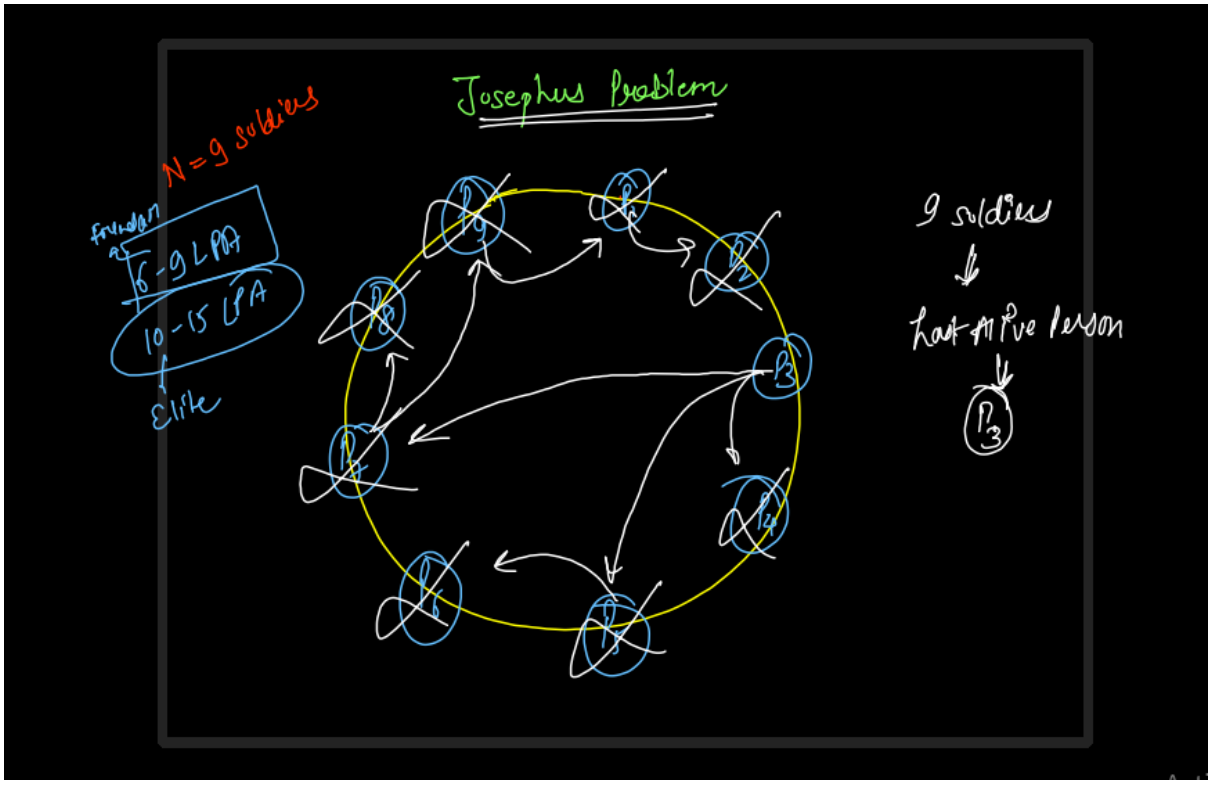
Time → O(Row * Col)

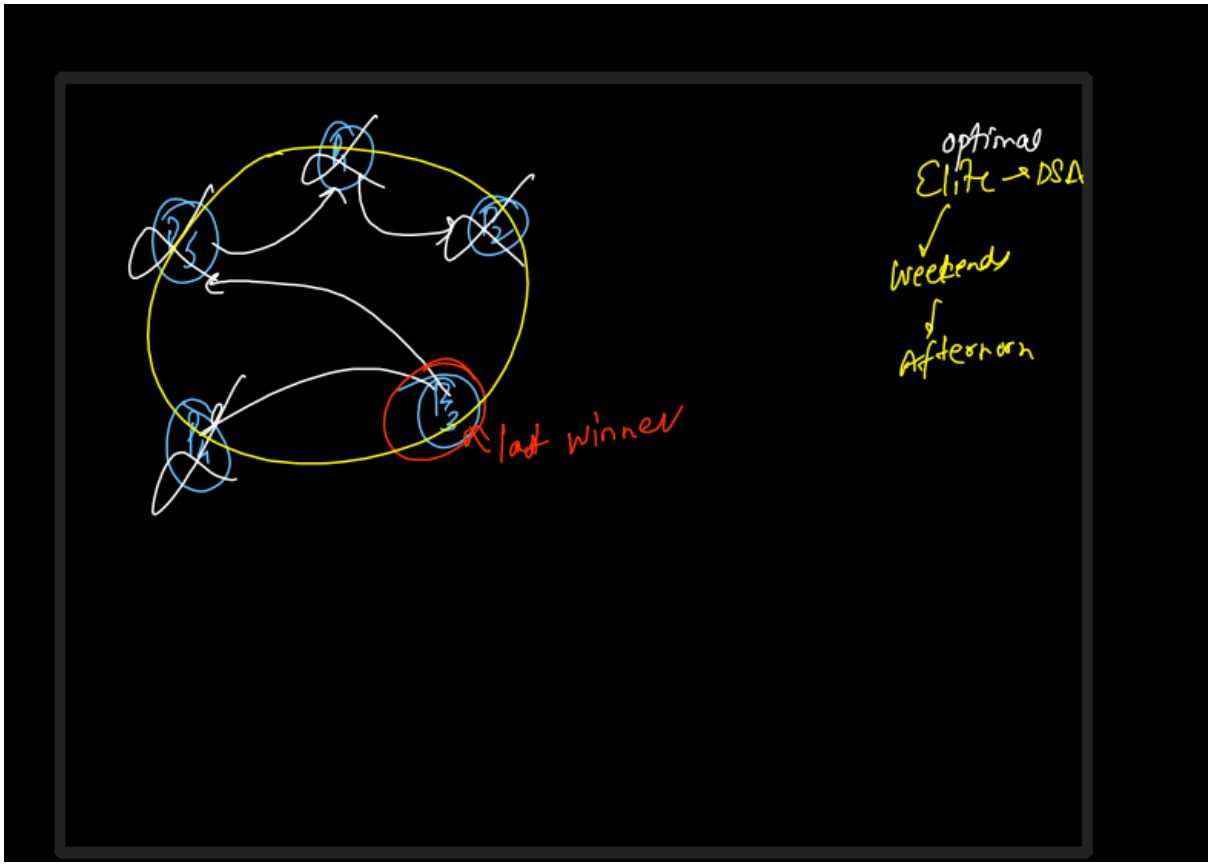

Subtract 2 Matrices
10 20 30 40
50 60 70 80
90 100 110 120
15 25 35 45
55 65 75 85
95 105 115 125
$N_1 = N_2 = N_3$
$M_1 = M_2 = M_3$


Division of 2 Matrices
Not Even Possible
Why ? → Type Mismatch
Multiplication of 2 matrices
res[i][j] = a[i][k] * b[k][j]
Constraint
$M_1 = N_2$
cols in I = rows in II
key
Row → I
Col → II
A [4*5]
$N_1 * M_1$
B[5*3]
$N_2 * M_2$
Res.
[4*3]
$N_1 * M_2$
Activate

$$res[i][j] = a[i][0] * b[0][j] + a[i][1] * b[1][j] + a[i][2] * b[2][j] + \vdots + a[i][4] * b[4][j]$$




Josephus Problem
N=9 soldiers
freshers
6-9 LPA
10-15 LPA
Elite
9 soldiers
last alive person


last winner
optimal
Elite → DSA
weekends
afternoon

Strings

int [] marks = {10, 20, 30, 40, 50}

(int - 4 bytes) * 5 = 20 bytes

char [] initials = {'A', 'B', 'C', 'D', 'E'}

single quoted

(char - 2 bytes) * 5 = 10 bytes

① Syntax

char [][]

{'A','R','C','H','I','T'}

{ Archit, Bhawani, Chirag, Deepak, Esha }

Group of characters

{'B','H','A','W','A','N','I'}

② Name → String → Archit / Chirag / Deepak / Sushant / Anilkumar

College →

Email ID

Reg ID

Already → length, Traverse

	String	Char Array
①	"Archit"	{'A', 'R', 'C', 'H', 'I', 'T'}
②	Size is not required during initialization	Size is required during initialization
③	Memory Efficient	Lot of memory wastage
④	Many functionalities: length, Reverse, substring, Concatenate	
⑤	Double quotes	Single quotes for characters

Memory is getting wasted

char [] = new char [30];

String → Array of characters

String name = "Archit";

String → class

String name2 = "Chirag";

Size of string indexing for declaration → not required

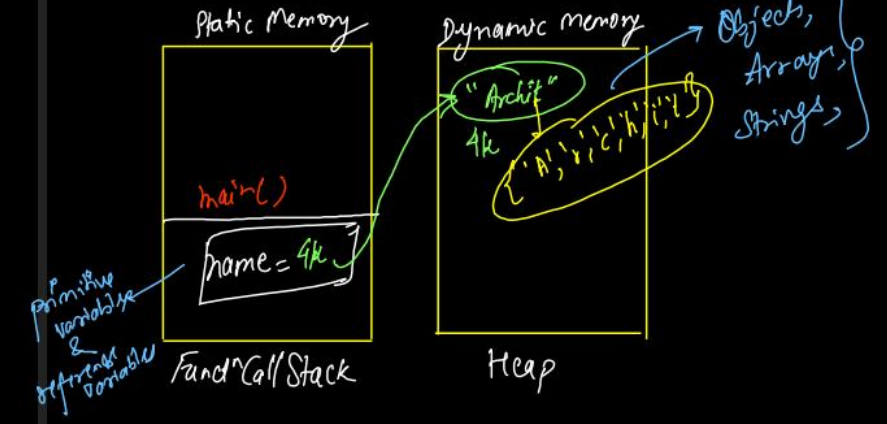
String[] names = {"Archit", "Bhawani", "Chirag", "Deepak", "Esha"};

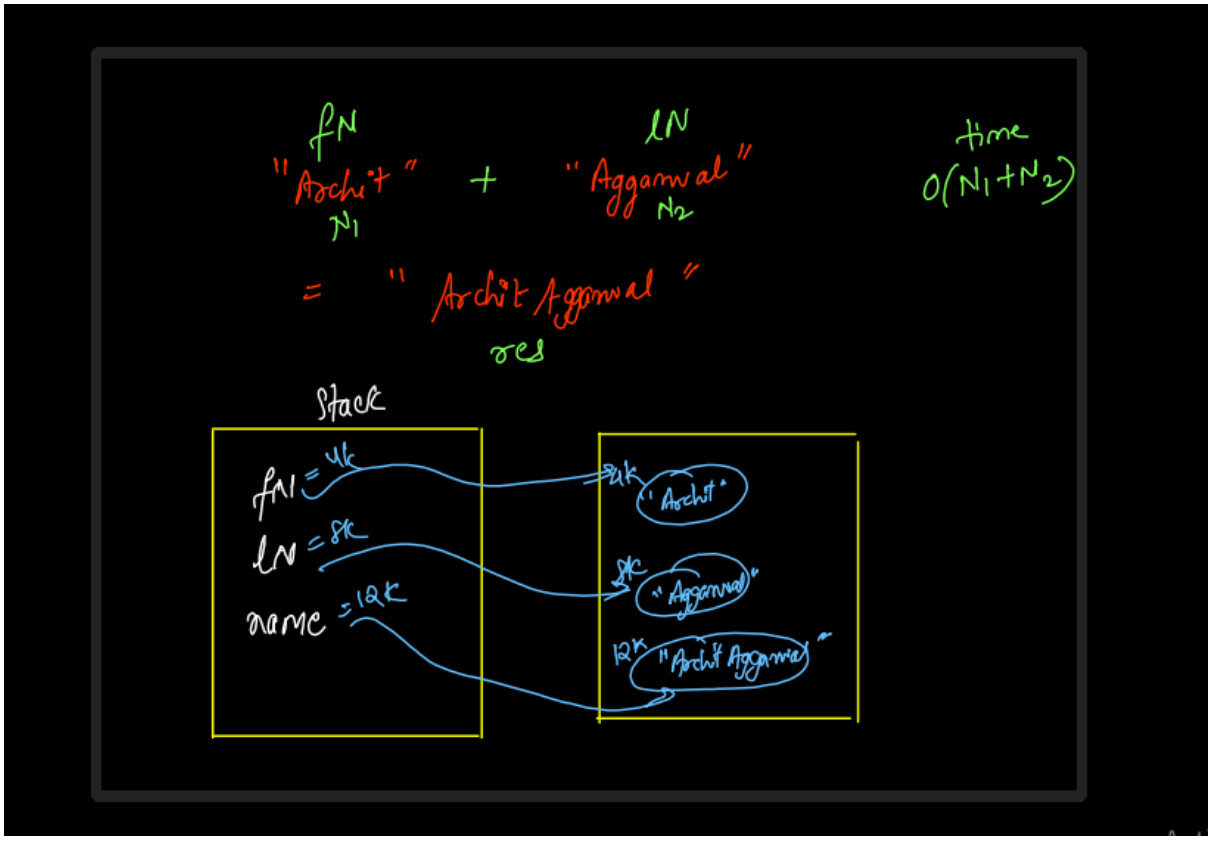
Memory Mapping

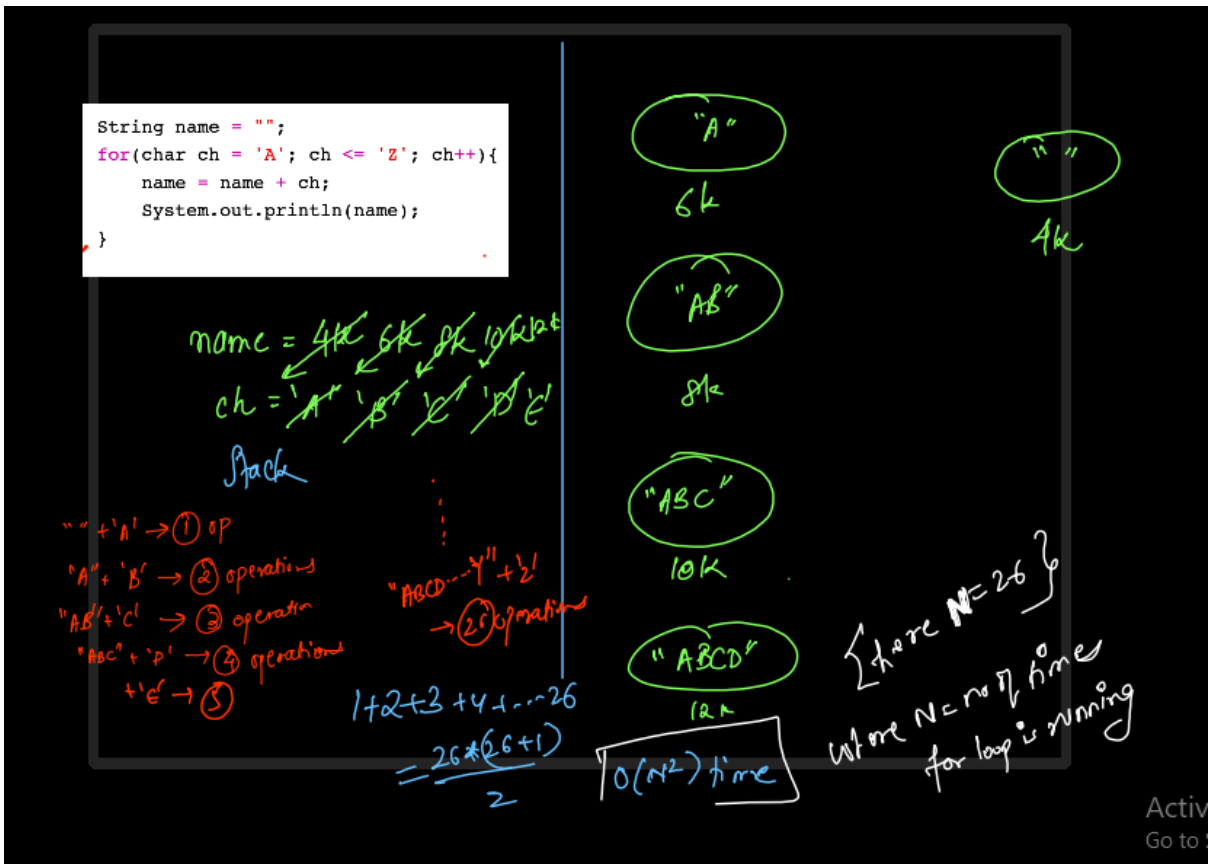
String name = "Archit";

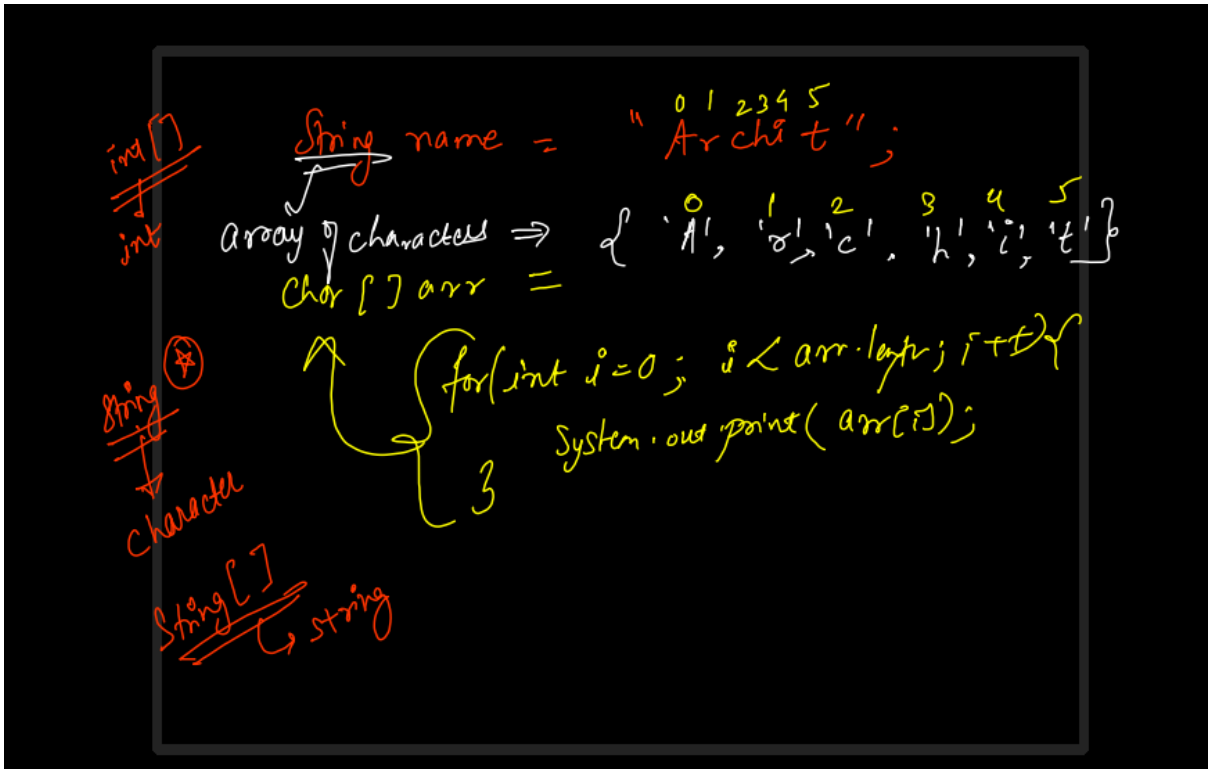
String str; ← Declaration

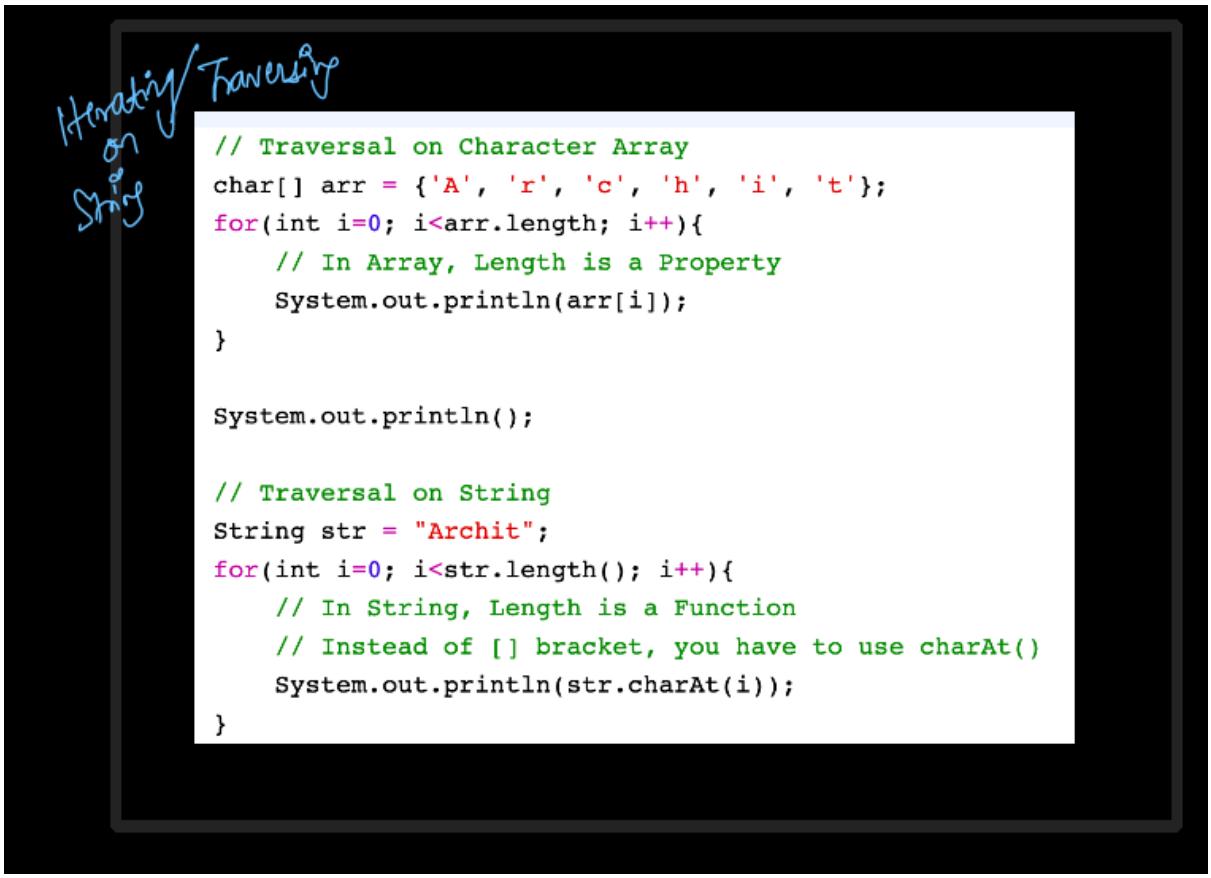
String str = "ABC"; ← Declaration & Initialization




fN
"Archit"
N1
+
lN
"Aggarwal"
N2
time
O(N1+N2)
= "Archit Aggarwal"
res
Stack
fN = 4k
lN = 8k
name = 12k
4k
"Archit"
8k
"Aggarwal"
12k
"Archit Aggarwal"


String name = "";
for(char ch = 'A'; ch <= 'Z'; ch++){
name = name + ch;
System.out.println(name);
}
name = 4k 6k 8k 10k 12k
ch = 'A' 'B' 'C' 'D' 'E'
Stack
"" + 'A' → 1 op
"A" + 'B' → 2 operations
"AB" + 'C' → 3 operations
"ABC" + 'D' → 4 operations
+'E' → 5
"ABCD...Y" + 'Z'
→ 26 operations
1+2+3+4+...26
= 26*(26+1)/2
O(N^2) time
"A"
6k
"AB"
8k
"ABC"
10k
"ABCD"
12k
""
4k
{here N=26
where N = no of times
for loop is running}






```java
// Traversal on Character Array
char[] arr = {'A', 'r', 'c', 'h', 'i', 't'};
for(int i=0; i<arr.length; i++){
    // In Array, Length is a Property
    System.out.println(arr[i]);
}

System.out.println();

// Traversal on String
String str = "Archit";
for(int i=0; i<str.length(); i++){
    // In String, Length is a Function
    // Instead of [] bracket, you have to use charAt()
    System.out.println(str.charAt(i));
}
```

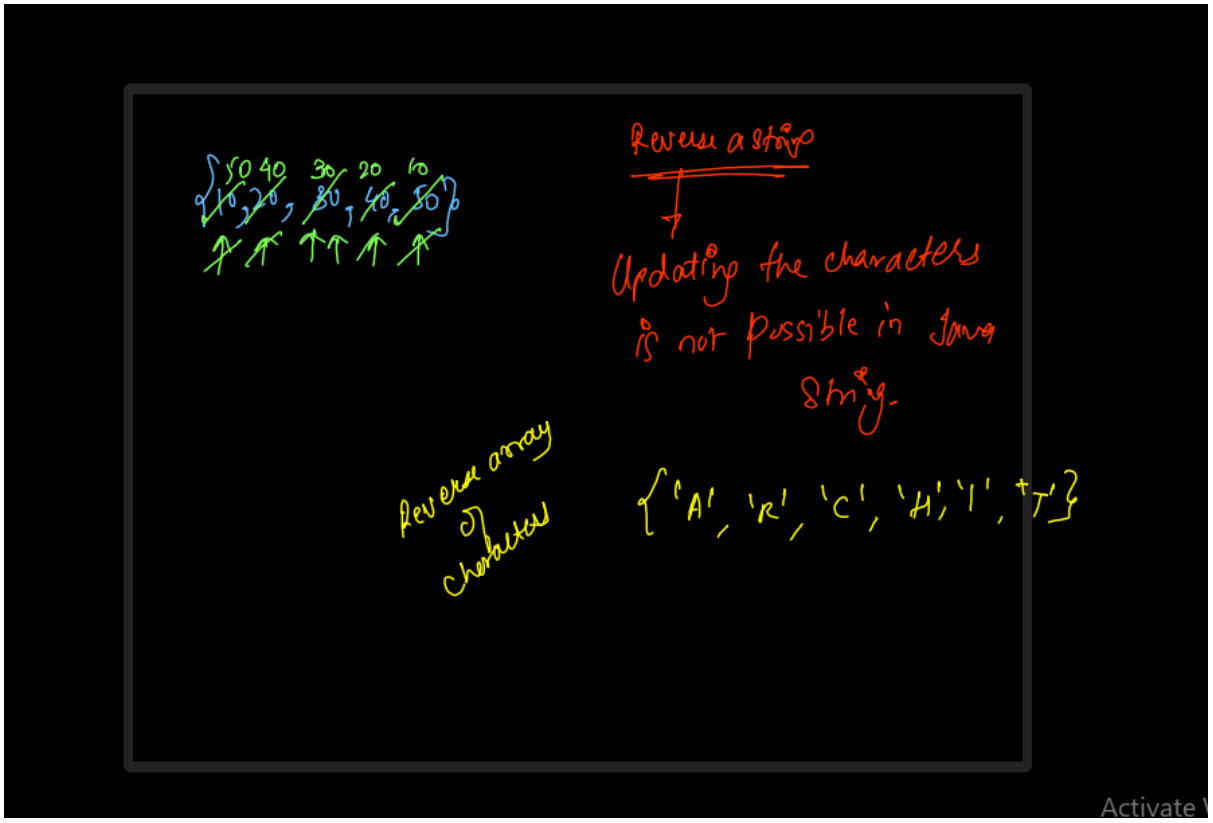
Count of vowels

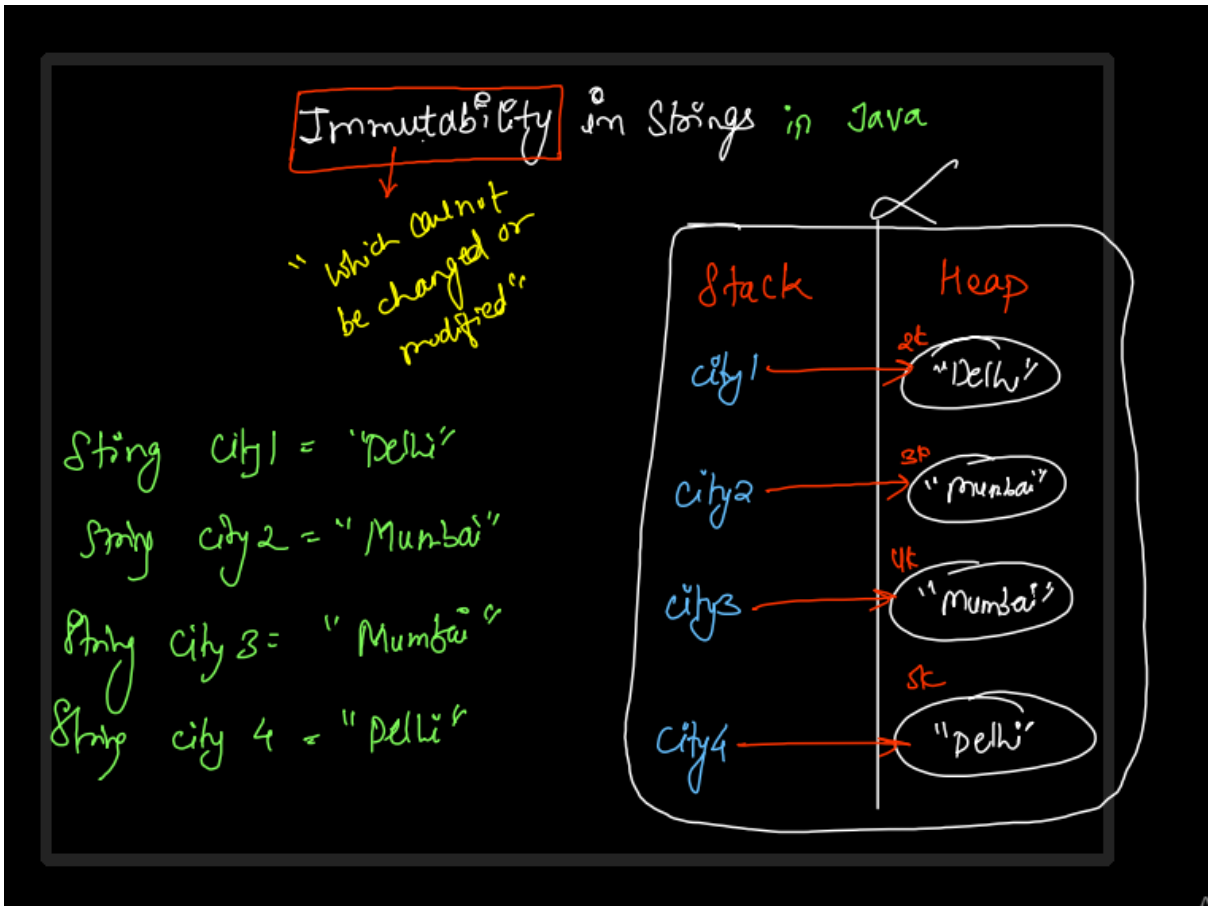
"Archit_Aggarwal" Count of vowels = 5

if (str.charAt(i) == 'A' || == 'E' || 'I' || 'O' || 'U' || 'a' || 'e' || 'i' || 'o' || 'u')

Count ++

```
// - WAP to count vowels in a string.
Scanner scn = new Scanner(System.in);
String str = scn.nextLine();
int vowel = 0;
for(int i=0; i<str.length(); i++){
    char ch = str.charAt(i);
    if(ch == 'A' || ch == 'E' || ch == 'I'
       || ch == 'O' || ch == 'U'
       || ch == 'a' || ch == 'e' || ch == 'i'
       || ch == 'o' || ch == 'u'){
        vowel++;
    }
}
System.out.println(vowel);
```


Reverse a string
Updating the characters is not possible in Java String.
Reverse array of characters
{'A', 'R', 'C', 'H', 'I', 'T'}


Immutability in Strings in Java
" which cannot be changed or modified "
String city1 = "Delhi"
String city2 = "Mumbai"
String city3 = "Mumbai"
String city4 = "Delhi"
Stack
Heap
city1
city2
city3
city4
"Delhi"
"Mumbai"
"Mumbai"
"Delhi"
2k
3k
4k
5k

Indian Railways

↳ millions of users → city { large section of cities will be exactly same }

U1 → Delhi
U2 → mumbai
U3 → Mumbai
U4 → Delhi
U5 → Kolkatta
U6 → Delhi
U7 → mumbai
U8 → Delhi

1000 users → "Delhi"

1000 * 5*2bytes

= 10000 bytes

500 users → "mumbai"

500 * 6*2bytes

= 6000 bytes

lot of space is going to be wasted when we are creating a diff string for each user

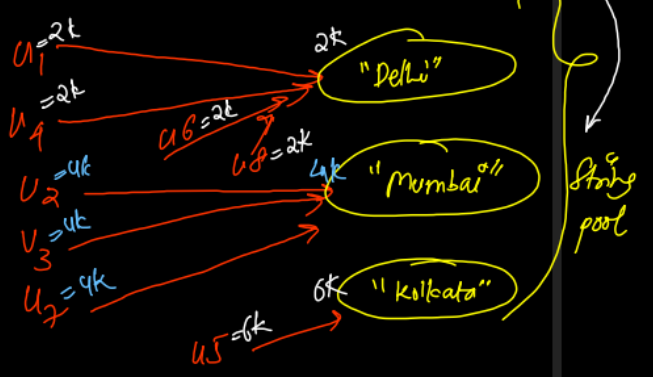
Interning in Java

WHAT

⇒ when we use the same string object for multiple reference variables in String, this concept is STRING INTERNING.

WHY

space optimization

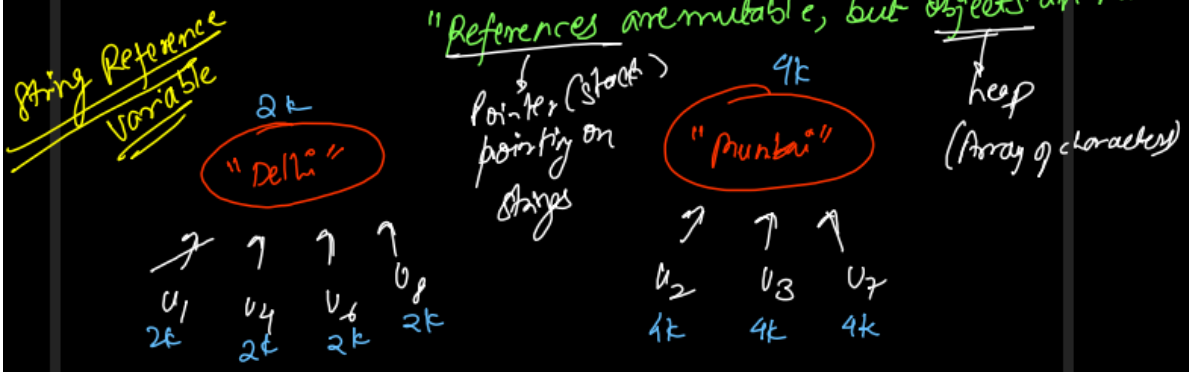


Immutability in Java Strings

{ This is a direct consequence of Interning }

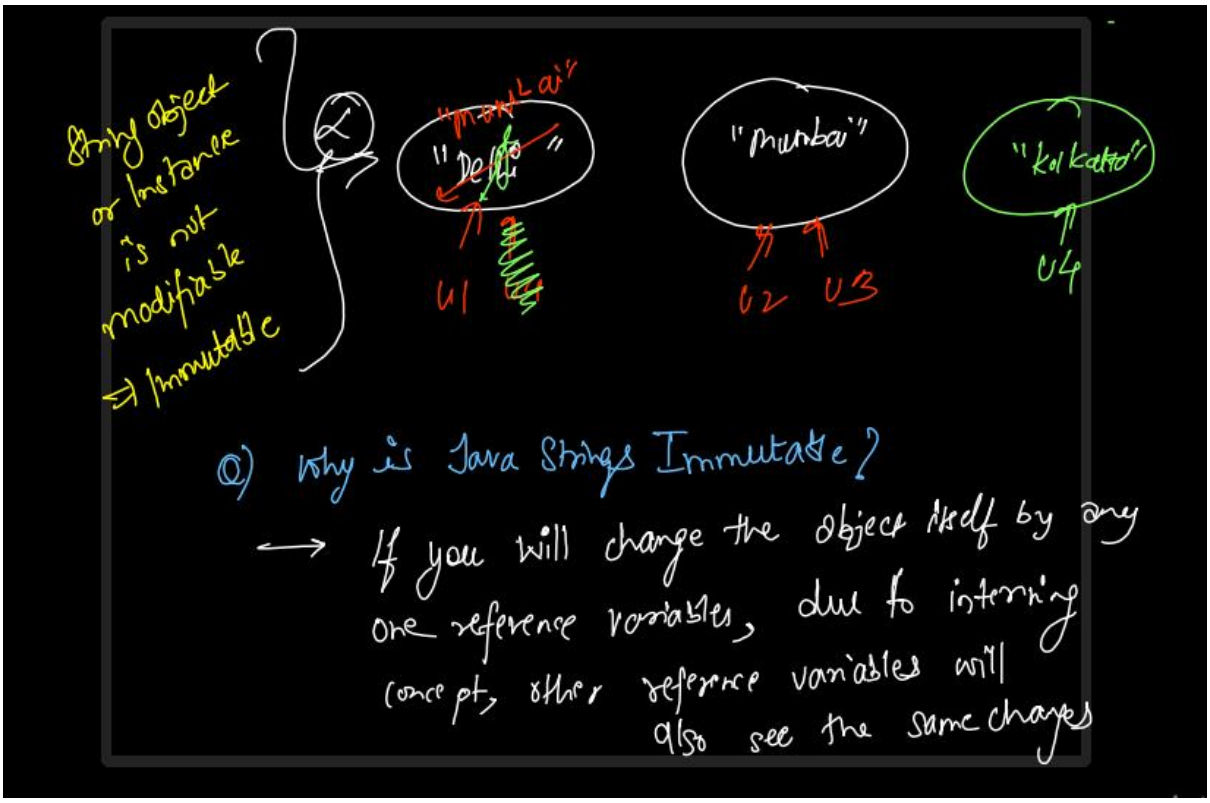
"References are mutable, but Objects are not"

String Reference Variable



If u1 wants to shift from Delhi to Mumbai,

⇒ just update reference variable



Q) Why is Java Strings Immutable?

→ If you will change the object itself by any one reference variables, due to interning concept, other reference variables will also see the same changes

"Archit" "Aggarwal"

Aci - gawl

#Word → Ideal word ← First letter as capital (Uppercase)
Rest all the letters as small (lowercase)

eg "Archit" → true

eg "archit" → false

eg "ARCHIT" → false

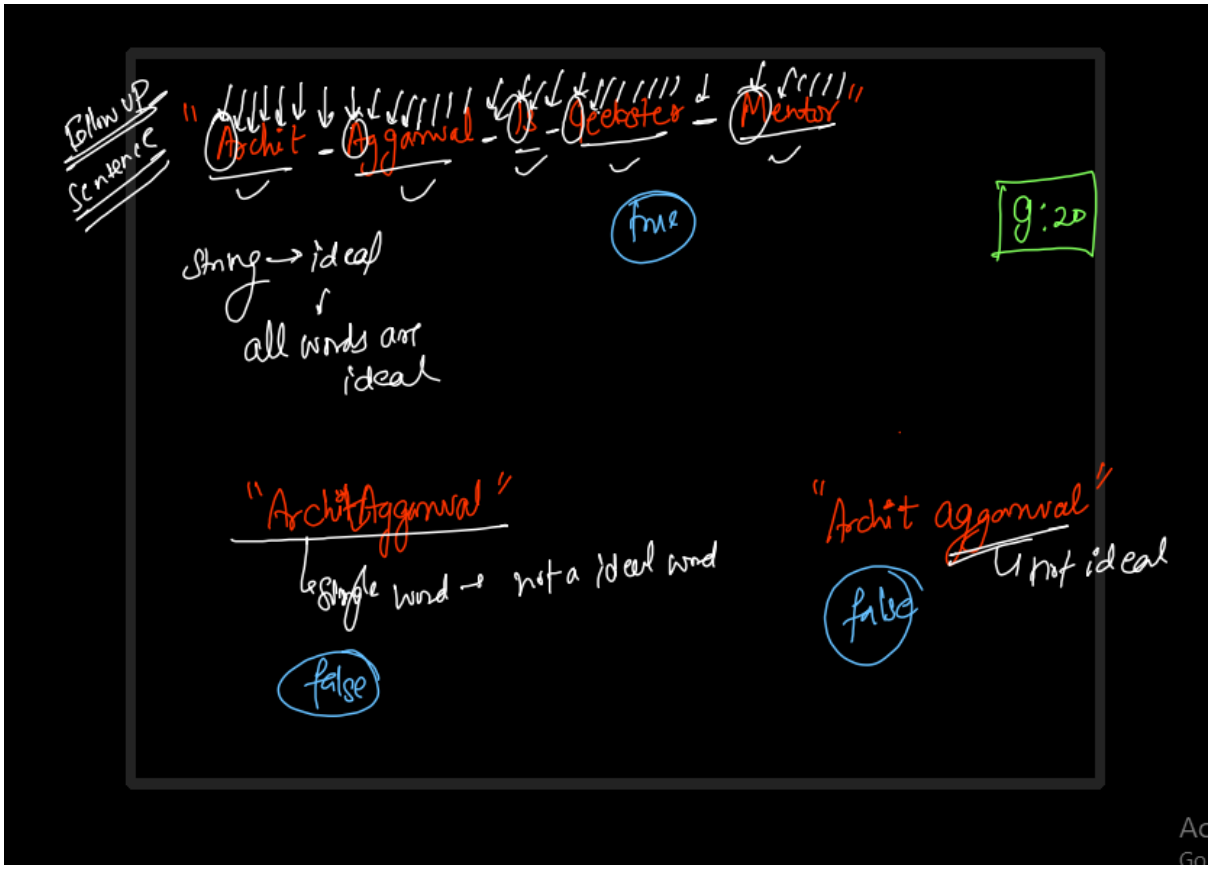
eg "A" → true

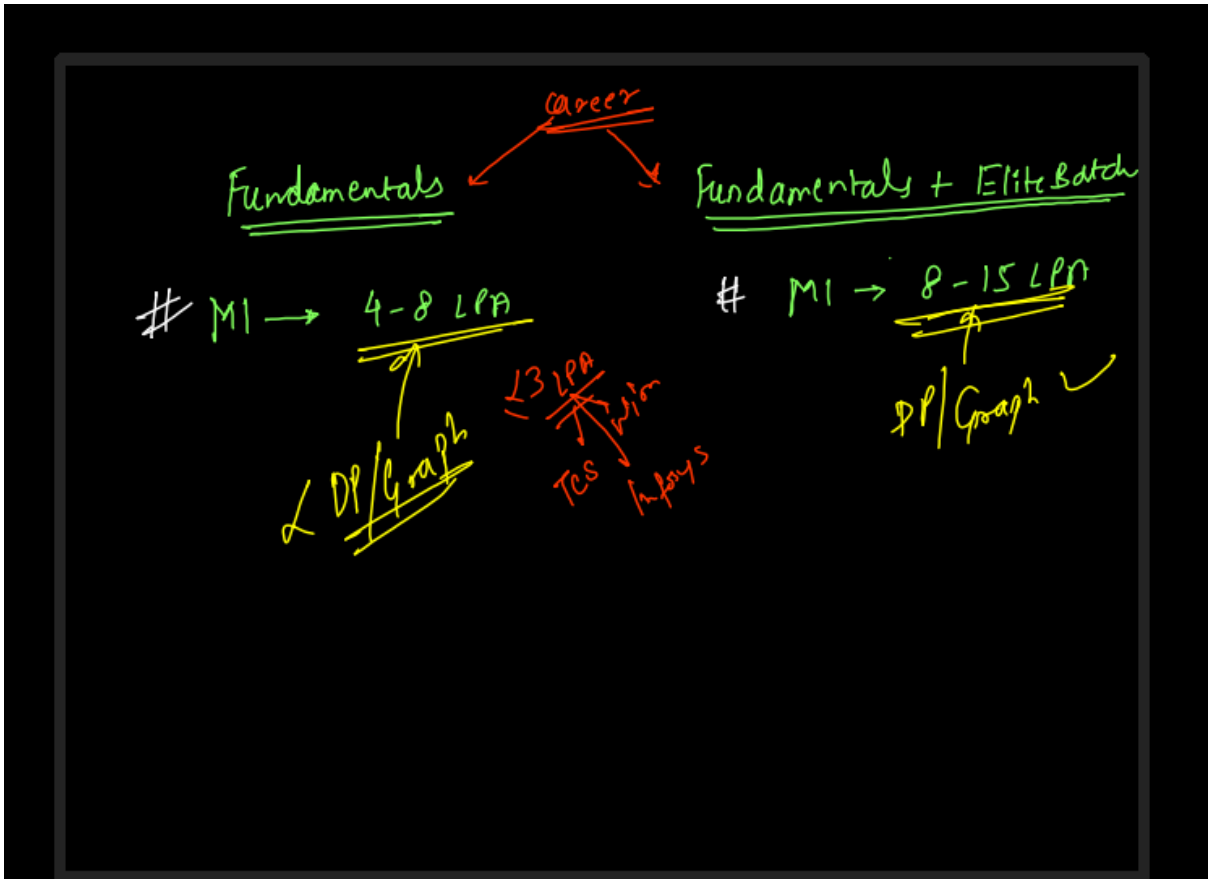

```java
String str = scn.next();
boolean isIdeal = true;
for(int i=0; i<str.length(); i++){
    char ch = str.charAt(i);

    if(i == 0){
        // First Letter
        // Should be Uppercase
        if(ch < 'A' || ch > 'Z'){
            isIdeal = false;
            break;
        }
    } else {
        // Remaining Letter
        // Should be Lowercase
        if(ch < 'a' || ch > 'z'){
            isIdeal = false;
            break;
        }
    }
}
```

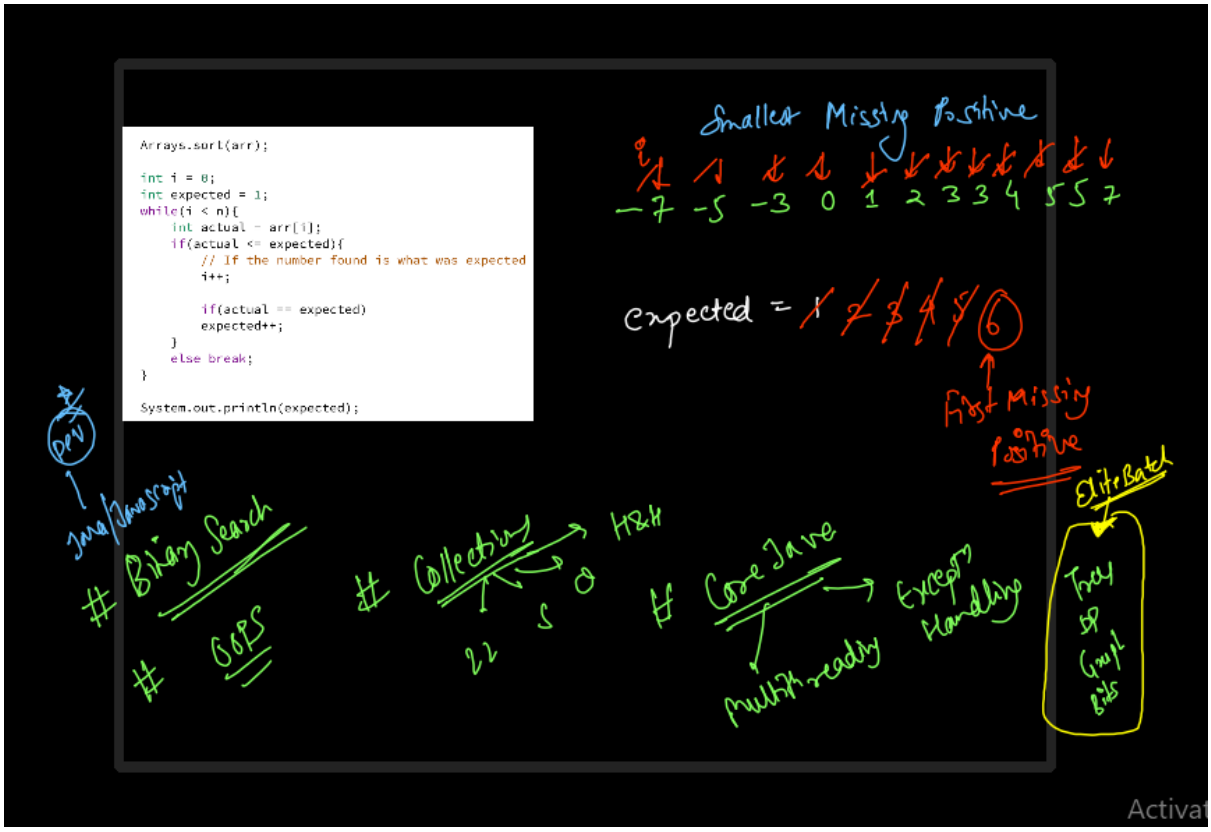
'A' → 65
'B' → 66
'C' → 67
'Z' → 90

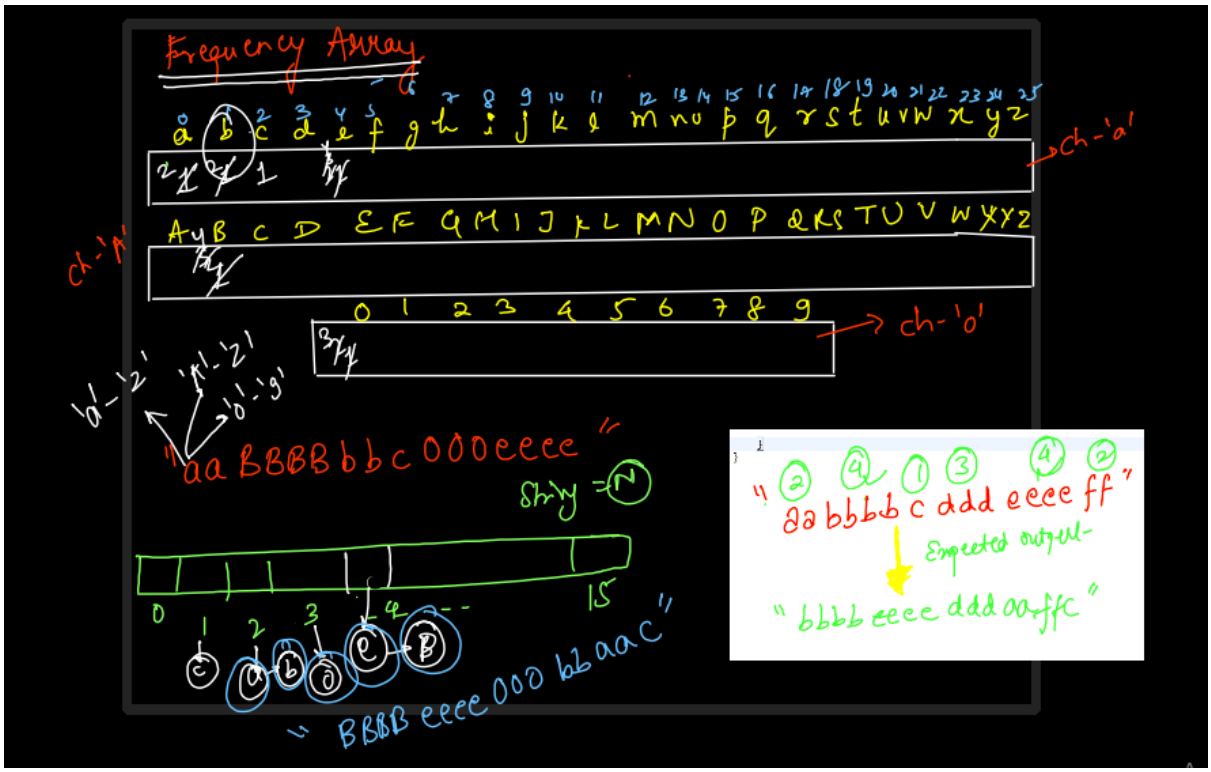
'a' → 97
'b' → 98
'c' → 99
'z' → 122

91 92 93 94 95 96

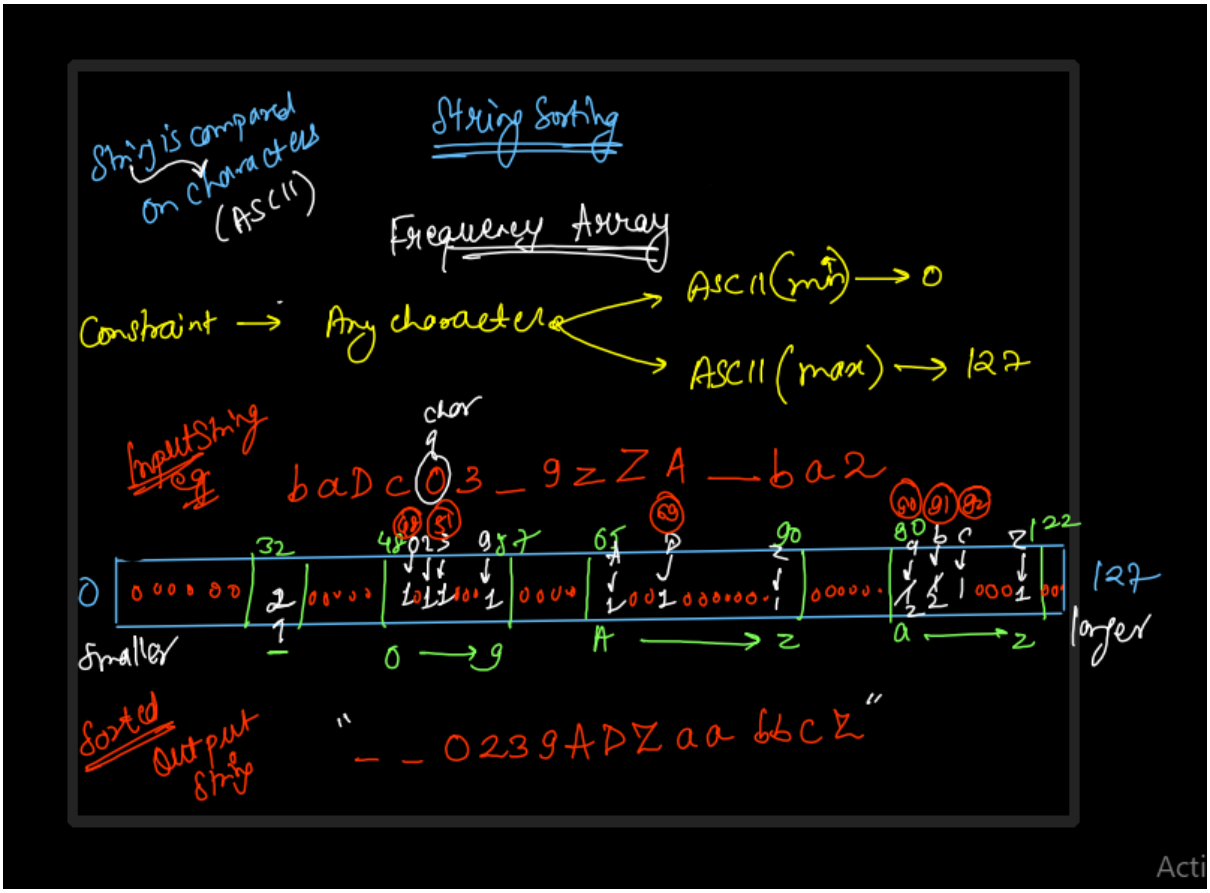

Follow UP Sentence
"Archit - Aggarwal - is - Geekster - Mentor"
String → ideal
all words are ideal
True
9:20
"ArchitAggarwal"
Single word → not a ideal word
false
"Archit aggarwal"
not ideal
false


Career
Fundamentals
Fundamentals + Elite Batch
M1 → 4-8 LPA
M1 → 8-15 LPA
DP/Graph
13 LPA
TCS
Infosys
DP/Graph

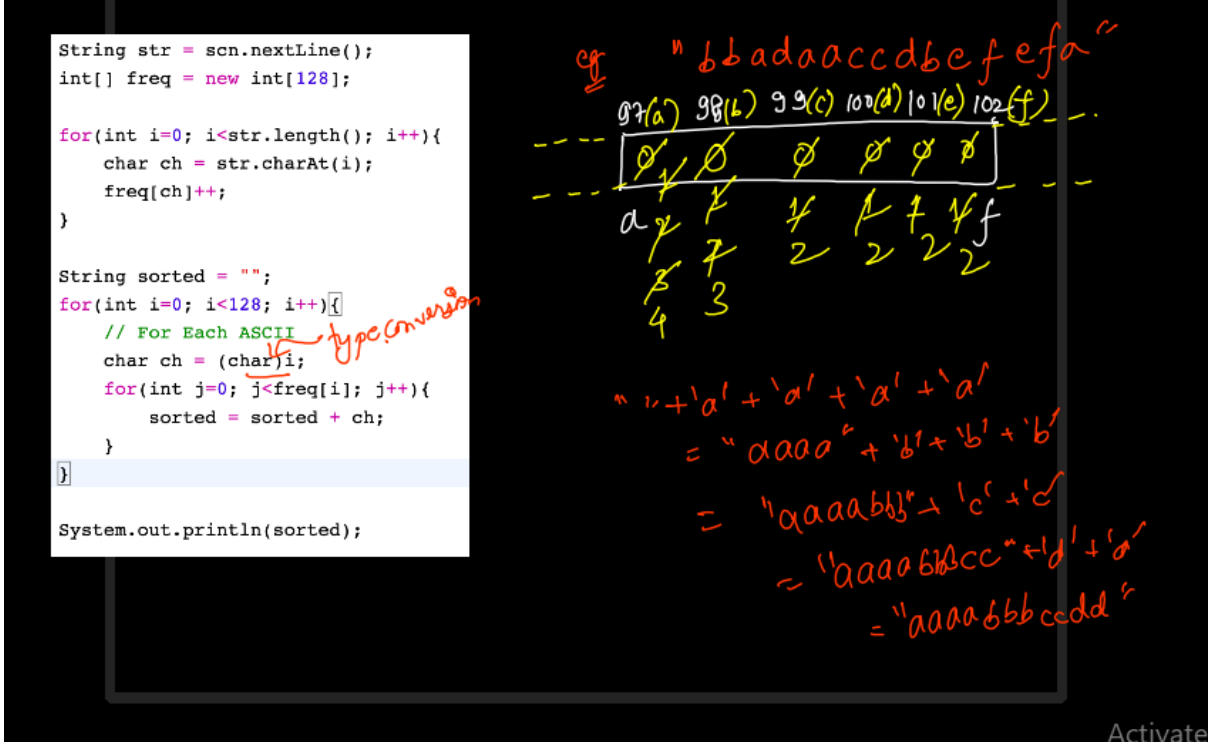

Arrays.sort(arr);
int i = 0;
int expected = 1;
while(i < n){
int actual = arr[i];
if(actual <= expected){
// If the number found is what was expected
i++;
if(actual == expected)
expected++;
}
else break;
}
System.out.println(expected);
Smallest Missing Positive
-7 -5 -3 0 1 2 3 3 4 5 5 7
expected = 1 2 3 4 5 6
First Missing Positive
Dev
Java/Javascript
Binary Search
OOPS
Collections
HashMap
Set
Core Java
Exception Handling
Multithreading
EliteBatch
Trees
DP
Graphs
Bits

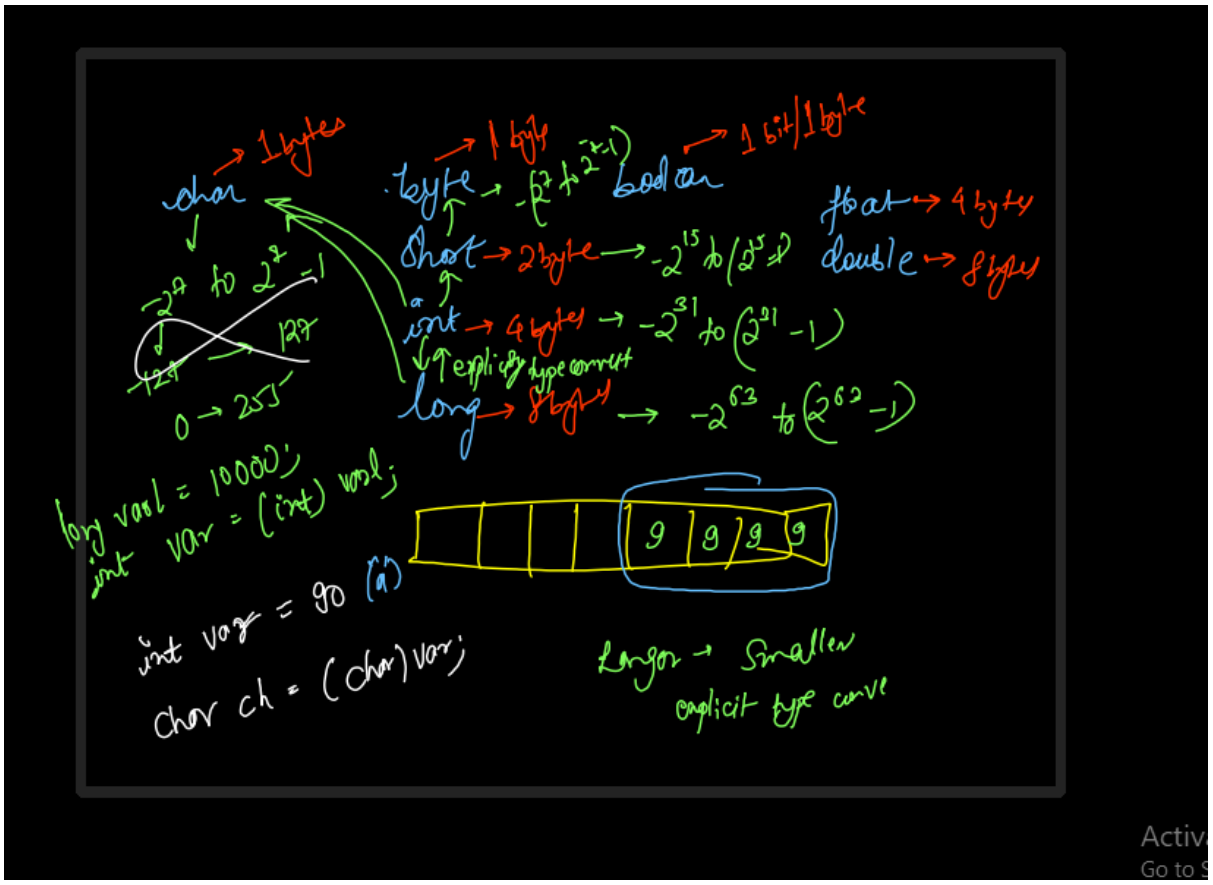

Frequency Array
a b c d e f g h i j k l m n o p q r s t u v w x y z
ch-'a'
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
ch-'A'
0 1 2 3 4 5 6 7 8 9
ch-'0'
'A'-'Z'
'a'-'z'
'0'-'9'
"aa BBBB bb c 000 eeee"
String
0 1 2 3 4 15
"BBBB eeee 000 bb aa c"
"2 4 1 3 4 2 aa bbbb c ddd eeee ff"
Expected output-
"bbbb eeee ddd 00 ff c"

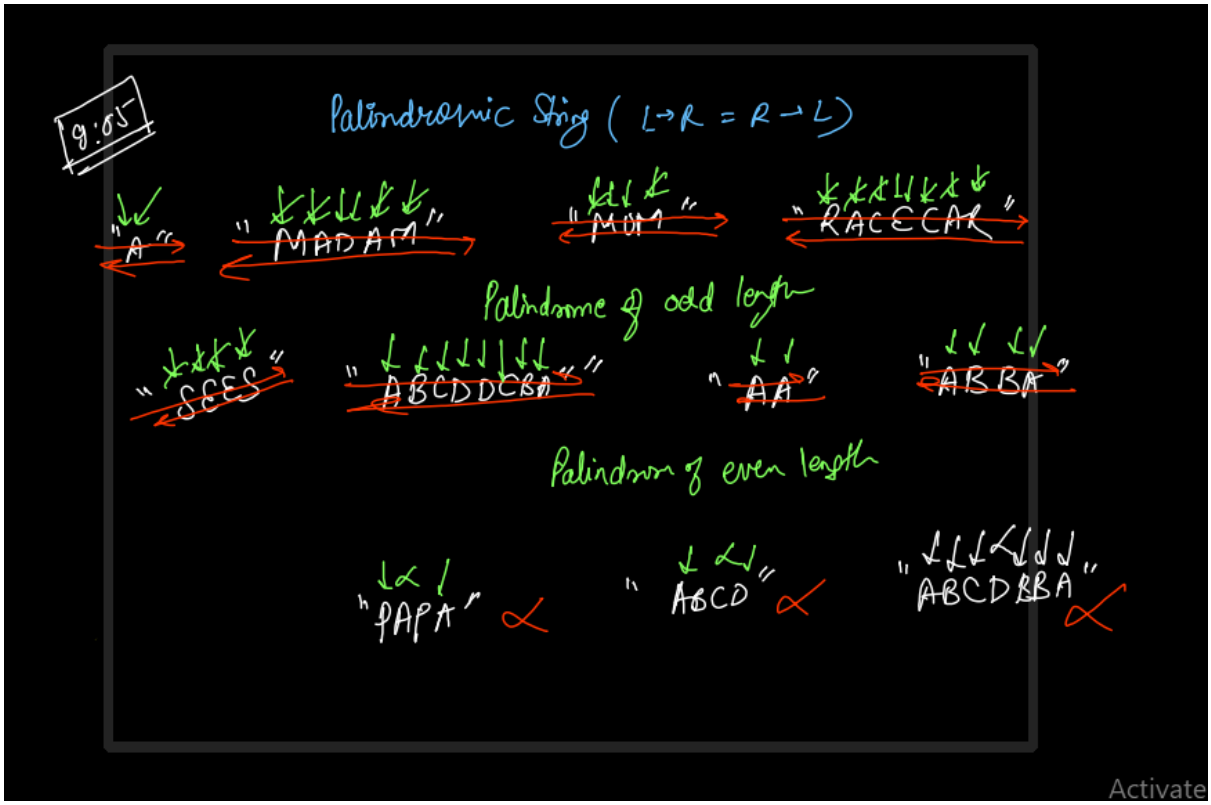
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);

    int min = Integer.MAX_VALUE;
    int max = Integer.MIN_VALUE;
    while(scn.hasNextInt()){
        int val = scn.nextInt();
        min = Math.min(min, val);
        max = Math.max(max, val);
    }

    System.out.print(min + " " + max);
}
```



```java
String str = scn.nextLine();
int[] freq = new int[128];

for(int i=0; i<str.length(); i++){
    char ch = str.charAt(i);
    freq[ch]++;
}

String sorted = "";
for(int i=0; i<128; i++){
    // For Each ASCII
    char ch = (char)i;
    for(int j=0; j<freq[i]; j++){
        sorted = sorted + ch;
    }
}

System.out.println(sorted);
```



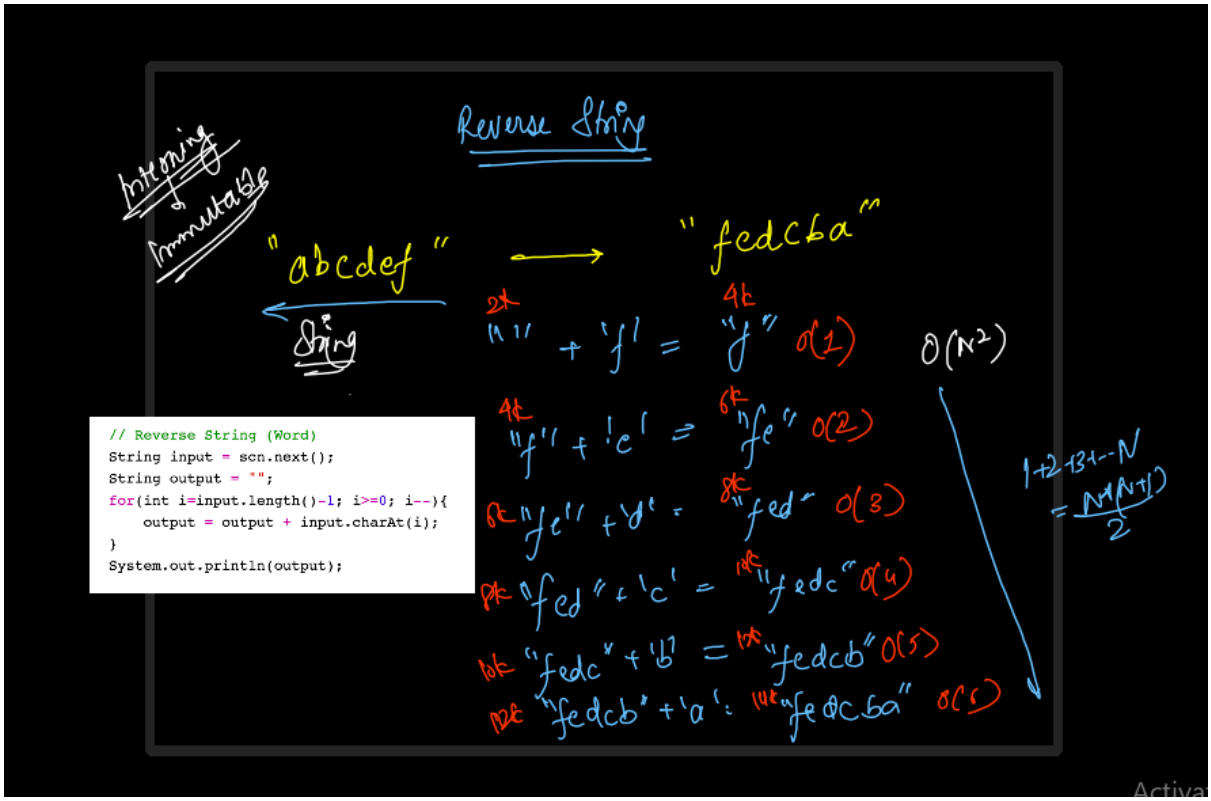


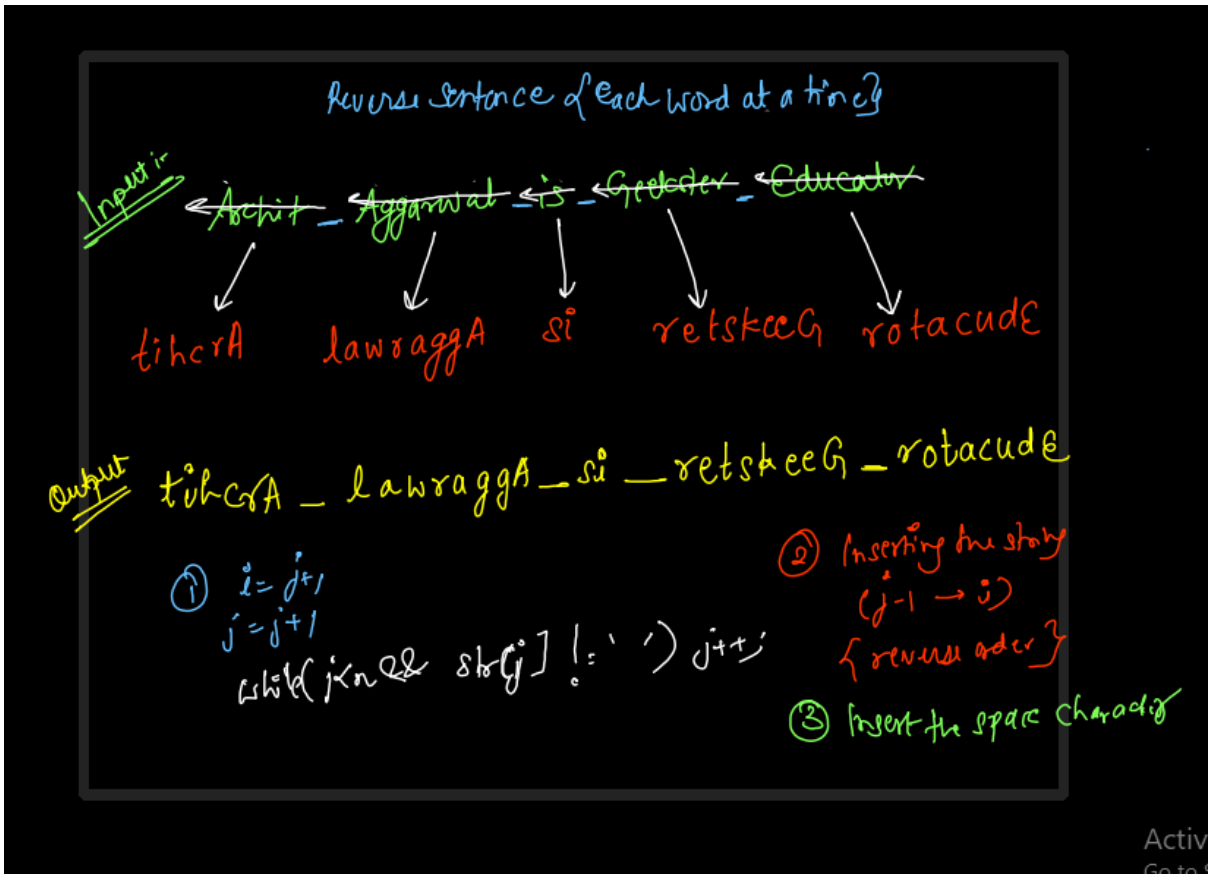



```java
// String is having Alphanumeric Characters
public boolean isPalindrome(String s) {
    int left = 0, right = s.length() - 1;
    while(left <= right){
        char chl = s.charAt(left);
        char chr = s.charAt(right);

        if(chl != chr){
            return false;
        }

        left++; right--;
    }

    return true;
}
```

Time ↓

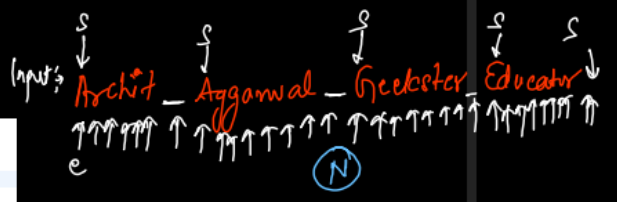
O(N) Linear


Reverse String
Interning
Immutable
"abcdef"
String
"fedcba"
// Reverse String (Word)
String input = scn.next();
String output = "";
for(int i=input.length()-1; i>=0; i--){
output = output + input.charAt(i);
}
System.out.println(output);
"" + 'f' = "f" O(1)
"f" + 'e' = "fe" O(2)
"fe" + 'd' = "fed" O(3)
"fed" + 'c' = "fedc" O(4)
"fedc" + 'b' = "fedcb" O(5)
"fedcb" + 'a' = "fedcba" O(6)
O(N^2)
1+2+3+...N = N(N+1)/2


Reverse Sentence of each word at a time
Input is
Archit Aggarwal is Geekster Educator
tihcrA lawraggA si retskeeG rotacudE
Output tihcrA_lawraggA_si_retskeeG_rotacudE
① i = j+1
j = j+1
while(j<n && str[j] != ' ') j++;
② Inserting the string (j-1 → i) {reverse order}
③ Insert the space character

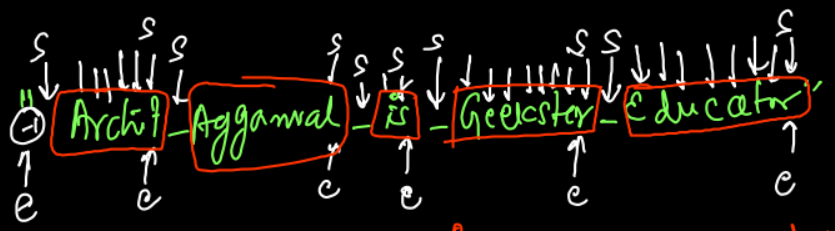
557 Leetcode

```
String output = "";
int start = 0, end = 0;

// Exploring Each Word
while(start < input.length()){

    // Taking the End Pointer to the next Space or End of String
    while(end < input.length() && input.charAt(end) != ' '){
        end++;
    }

    // Inserting the word in reverse order [end - 1, start]
    for(int i=end-1; i>=start; i--){
        output = output + input.charAt(i);
    }

    // Adding Space after every word but not for the last word
    if(end < input.length()){
        output = output + ' ';
    }

    // Going to the next word
    end++;
    start = end;
}

return output;
```



output :→ tihcrA_lawraggA_retskeeG_rotacudE

leetcode 151

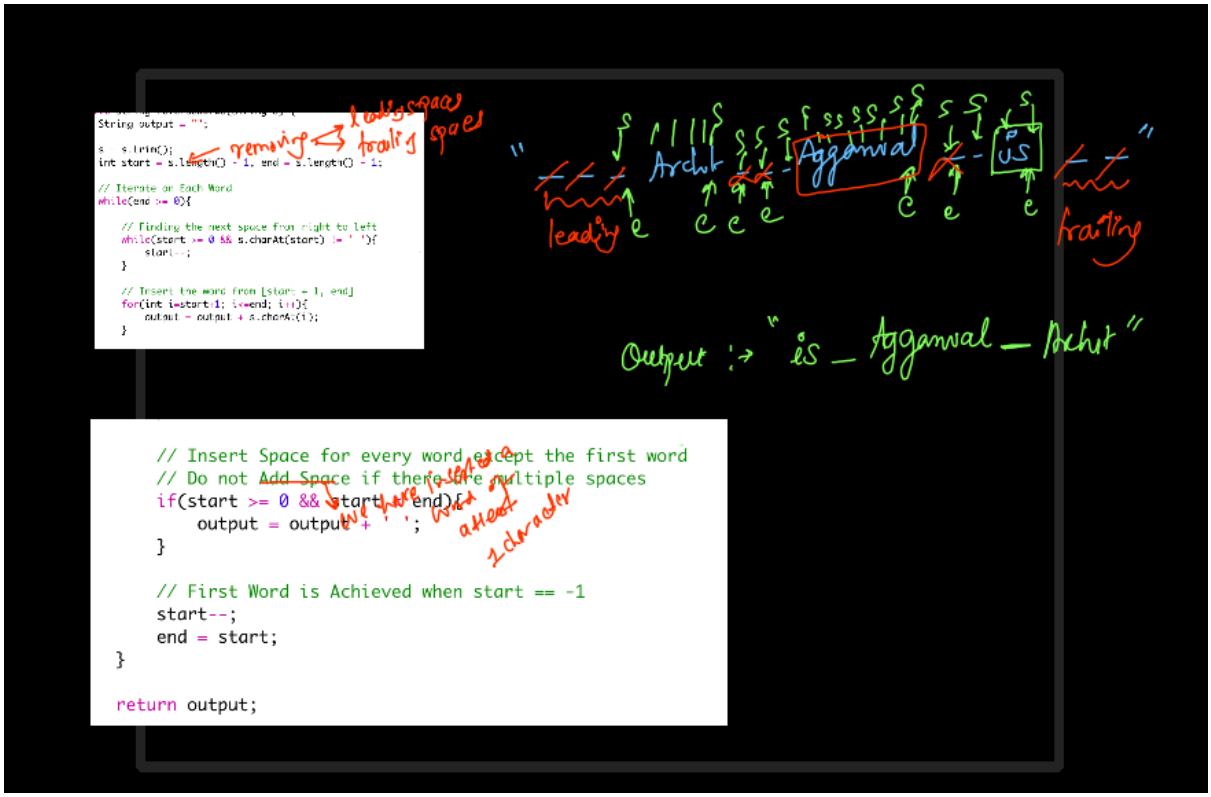


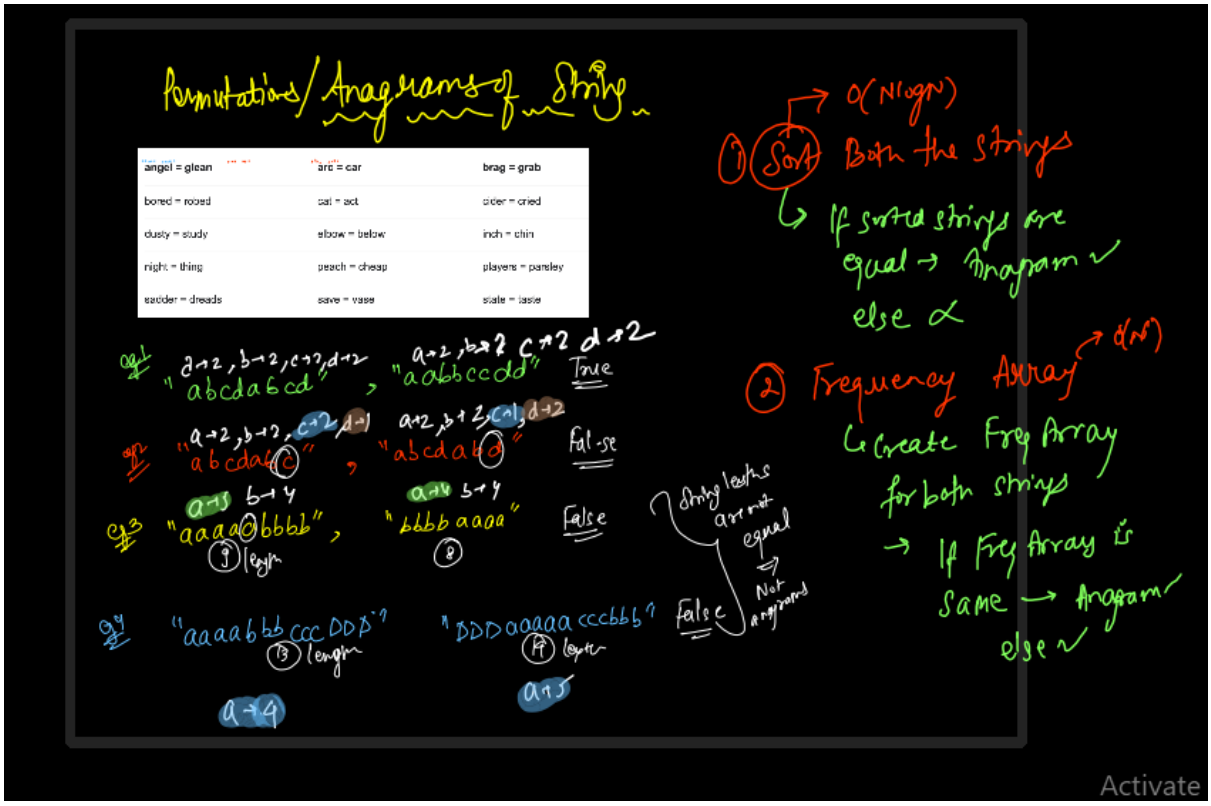
"Educator_Geeksfor_is_Aggarwal_Archit"

"Educator Geeksfor is Aggarwal Archit"

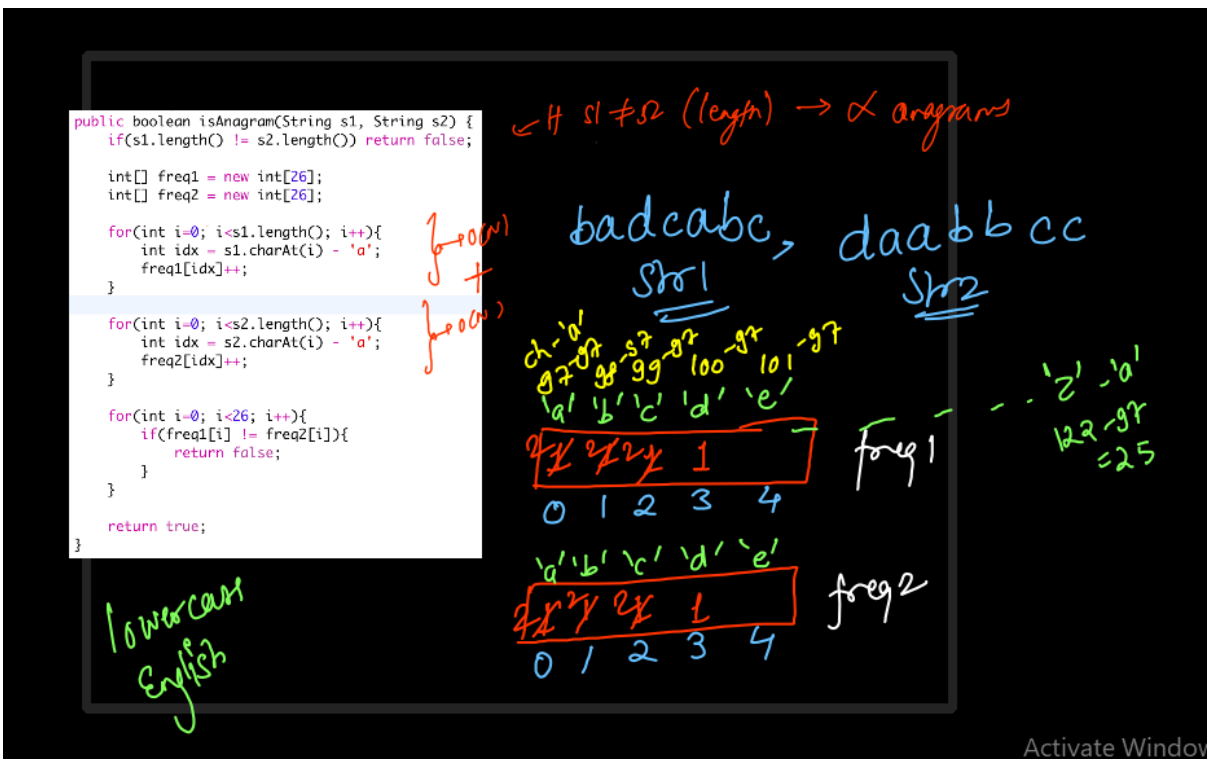
✓ Logic 8:33

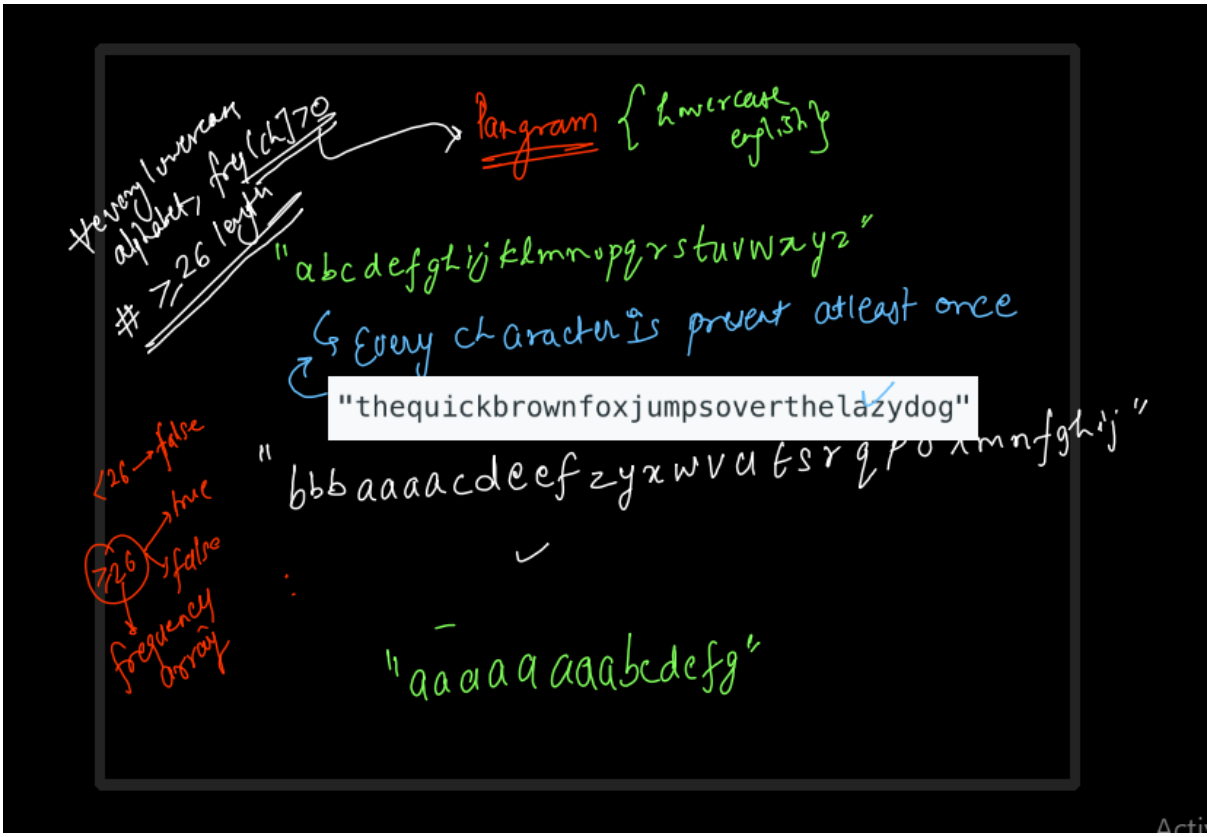
✓ Code 8:47


String output = "";
s = s.trim();
int start = s.length() - 1, end = s.length() - 1;
// Iterate on Each Word
while(end >= 0){
// Finding the next space from right to left
while(start >= 0 && s.charAt(start) != ' '){
start--;
}
// Insert the word from [start + 1, end]
for(int i=start+1; i<=end; i++){
output = output + s.charAt(i);
}
removing
leading spaces
trailing spaces
Archit
Aggarwal
US
leading
trailing
Output :→ "is _ Aggarwal _ Archit"
// Insert Space for every word except the first word
// Do not Add Space if there is the multiple spaces
if(start >= 0 && start != end){
output = output + ' ';
}
// First Word is Achieved when start == -1
start--;
end = start;
}
return output;

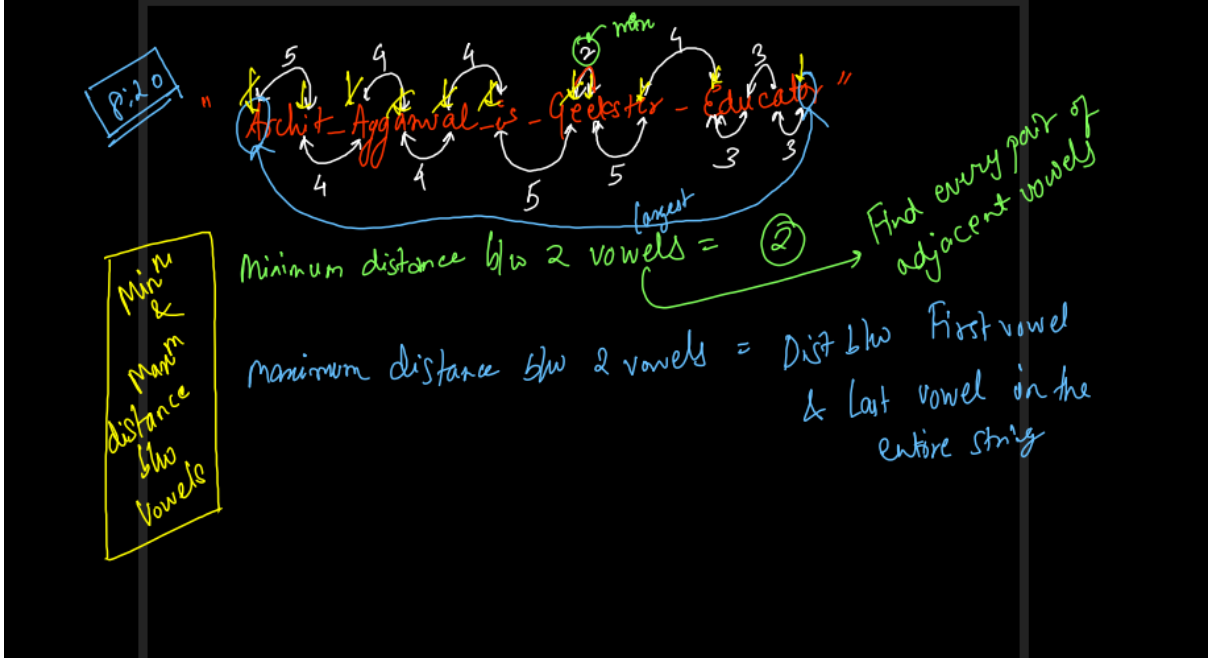

Permutations / Anagrams of String
angel = glean
arc = car
brag = grab
bored = robed
sat = act
cider = cried
dusty = study
elbow = below
inch = chin
night = thing
peach = cheap
players = parsley
sadder = dreads
save = vase
state = taste
O(NlogN)
Sort
Both the strings
If sorted strings are equal → Anagram
else ✗
Frequency Array
O(N)
Create Freq Array for both strings
→ If Freq Array is same → Anagram
else
True
False
False
String lengths are not equal ⇒ Not anagrams
False
Activate

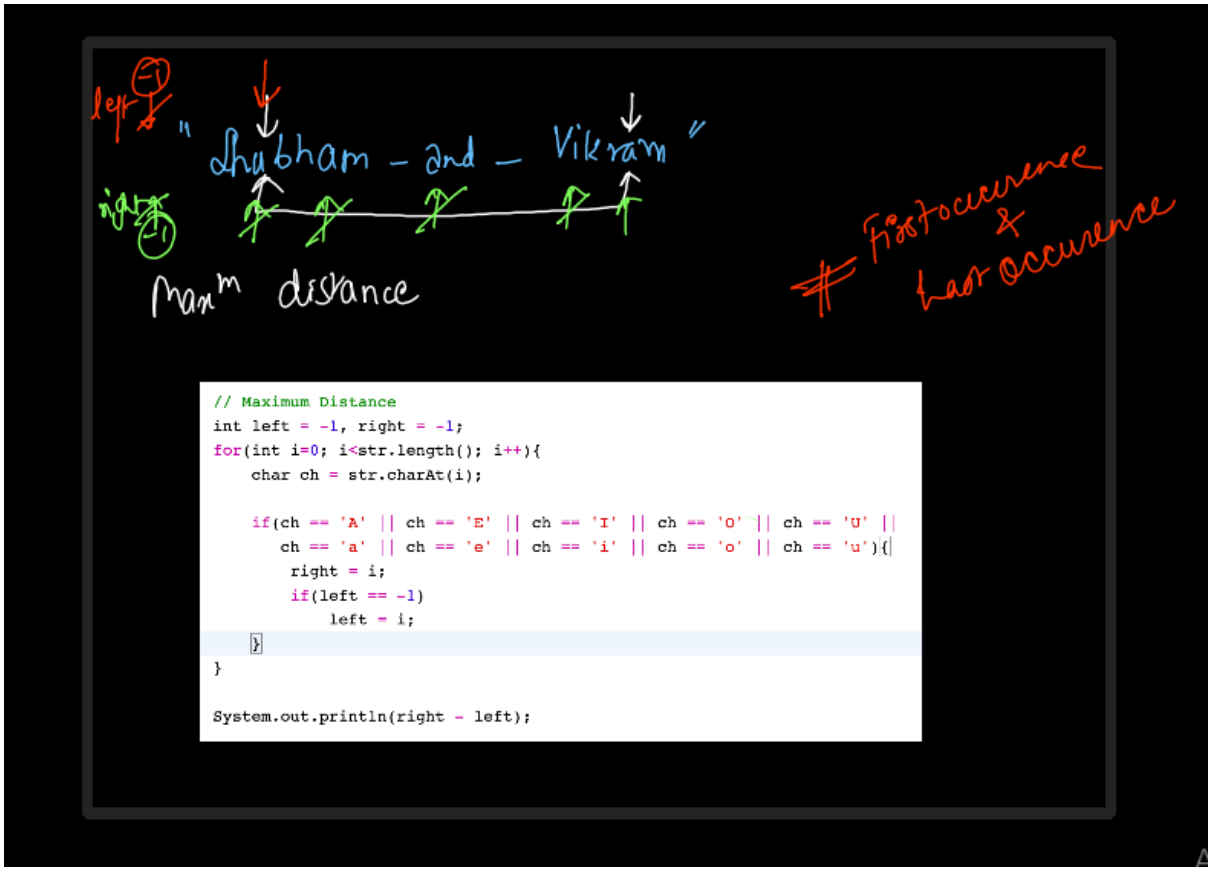
```java
public boolean isAnagram(String s1, String s2) {
    if(s1.length() != s2.length()) return false;

    int[] freq1 = new int[26];
    int[] freq2 = new int[26];

    for(int i=0; i<s1.length(); i++){
        int idx = s1.charAt(i) - 'a';
        freq1[idx]++;
    }

    for(int i=0; i<s2.length(); i++){
        int idx = s2.charAt(i) - 'a';
        freq2[idx]++;
    }

    for(int i=0; i<26; i++){
        if(freq1[i] != freq2[i]){
            return false;
        }
    }

    return true;
}
```



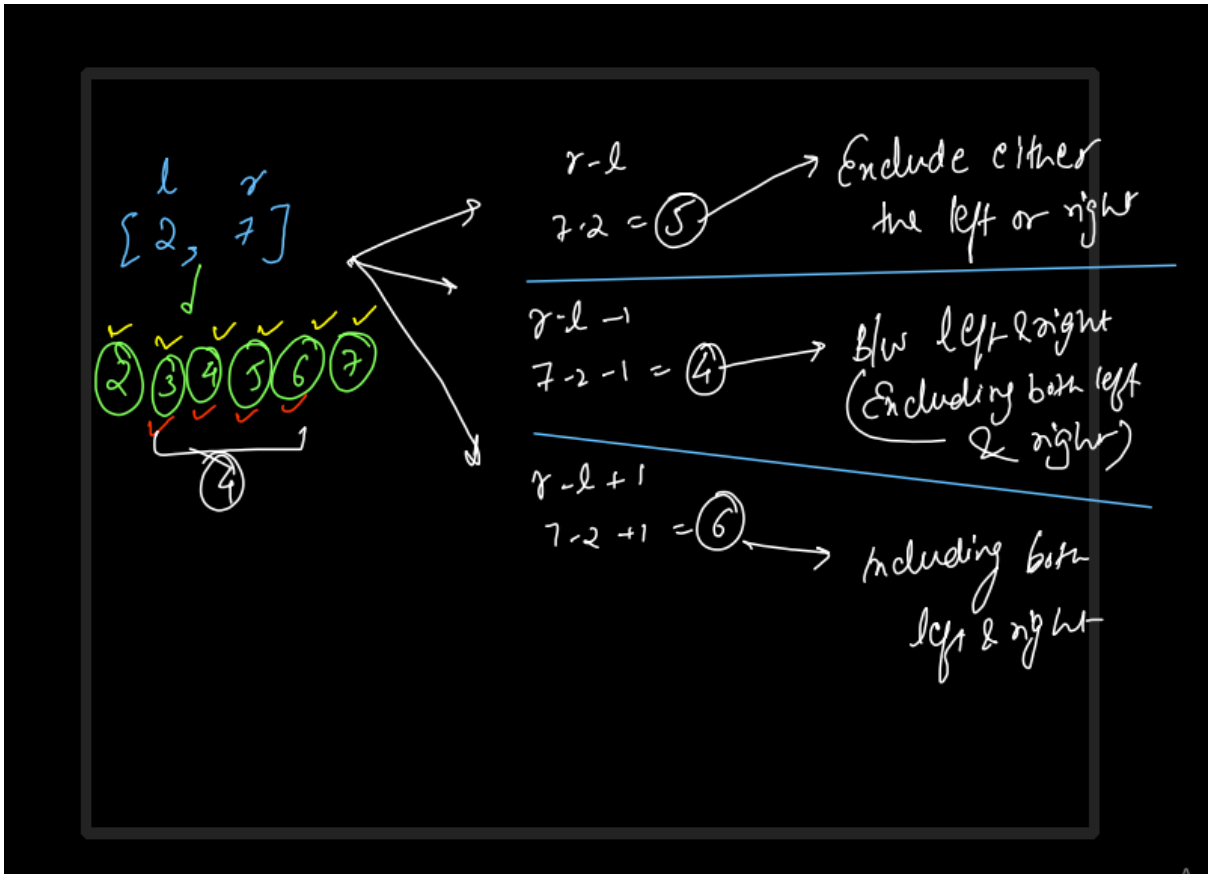


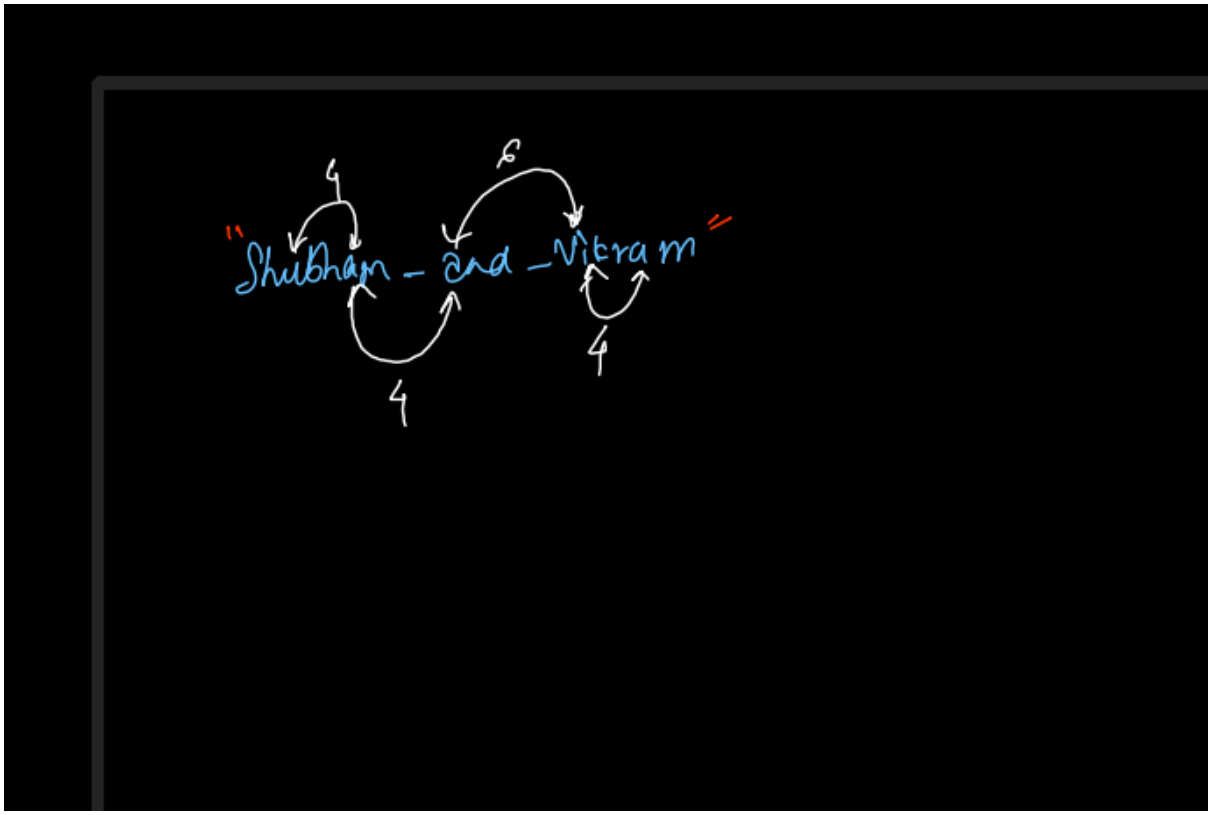
```java
public boolean checkIfPangram(String str) {
    if(str.length() < 26) return false;

    int[] freq = new int[26];
    for(int i=0; i<str.length(); i++){
        int idx = str.charAt(i) - 'a';
        freq[idx]++;
    }

    for(int i=0; i<26; i++){
        if(freq[i] == 0){
            return false;
        }
    }

    return true;
}
```



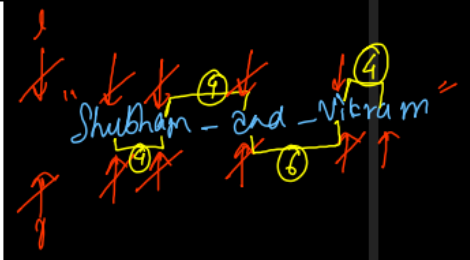



```java
// Maximum Distance
int left = -1, right = -1;
for(int i=0; i<str.length(); i++){
    char ch = str.charAt(i);

    if(ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U' ||
       ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u'){
        right = i;
        if(left == -1)
            left = i;
    }
}

System.out.println(right - left);
```






```java
// Minimum Distance
int left = -1, right = -1;
int min = Integer.MAX_VALUE;
for(int i=0; i<str.length(); i++){
    char ch = str.charAt(i);

    if(ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U' ||
       ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u'){
        left = right;
        right = i;

        if(left != -1){
            min = Math.min(min, right - left + 1);
        }
    }
}

System.out.println(min);
```



min = ~~+∞~~
4

Convert String to Integer

String ⟶ Integer

"25052022" ⟶ 25052022

"9999999" ⟶ 9999999

"10000" ⟶ 10000

"0001" ⟶ 1

"00000" ⟶ 0

---

① inbuilt method

int var = Integer.parseInt(str);

② Parsing each digit (character) & convert it into a integer

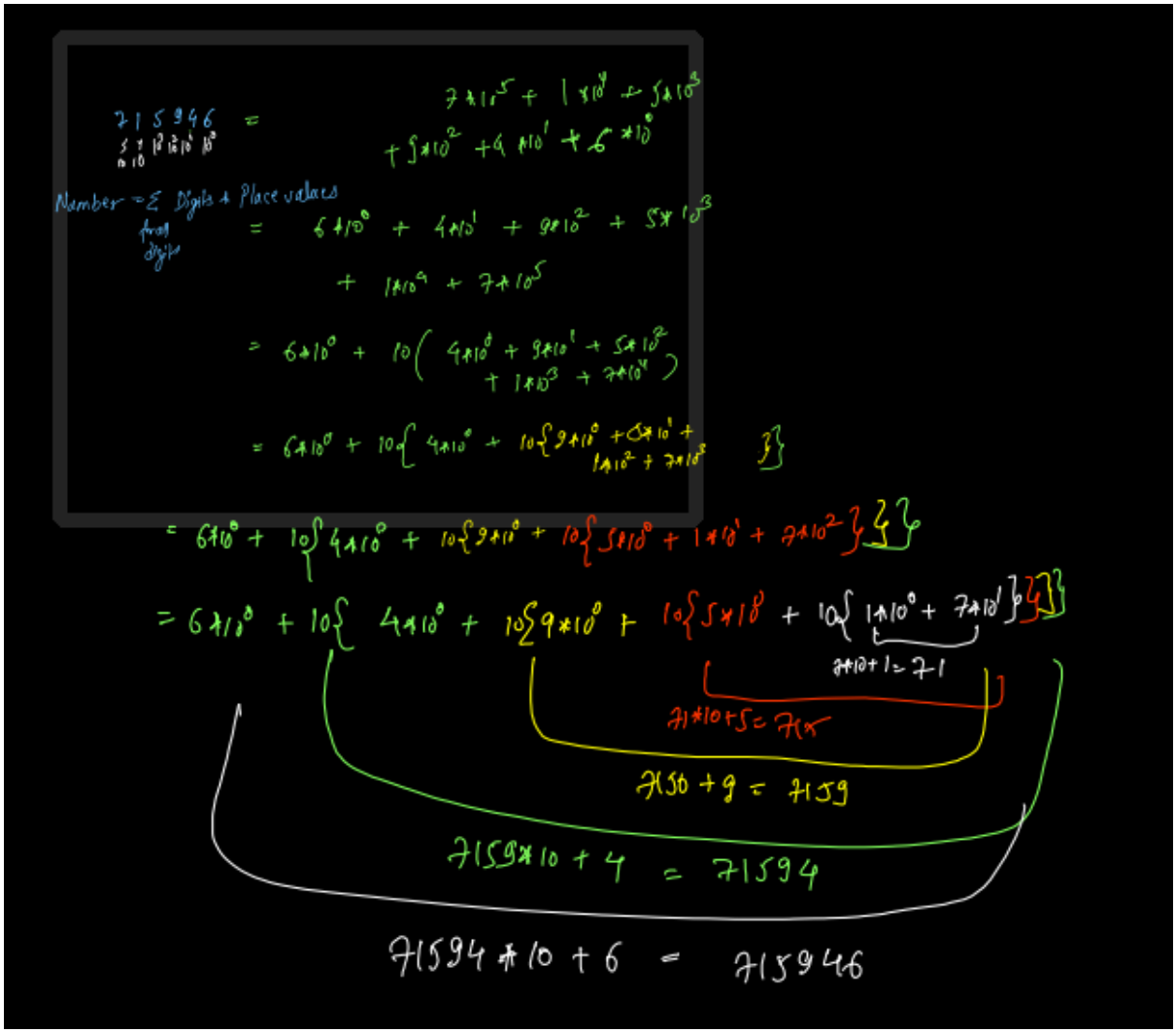
"2505 2022"

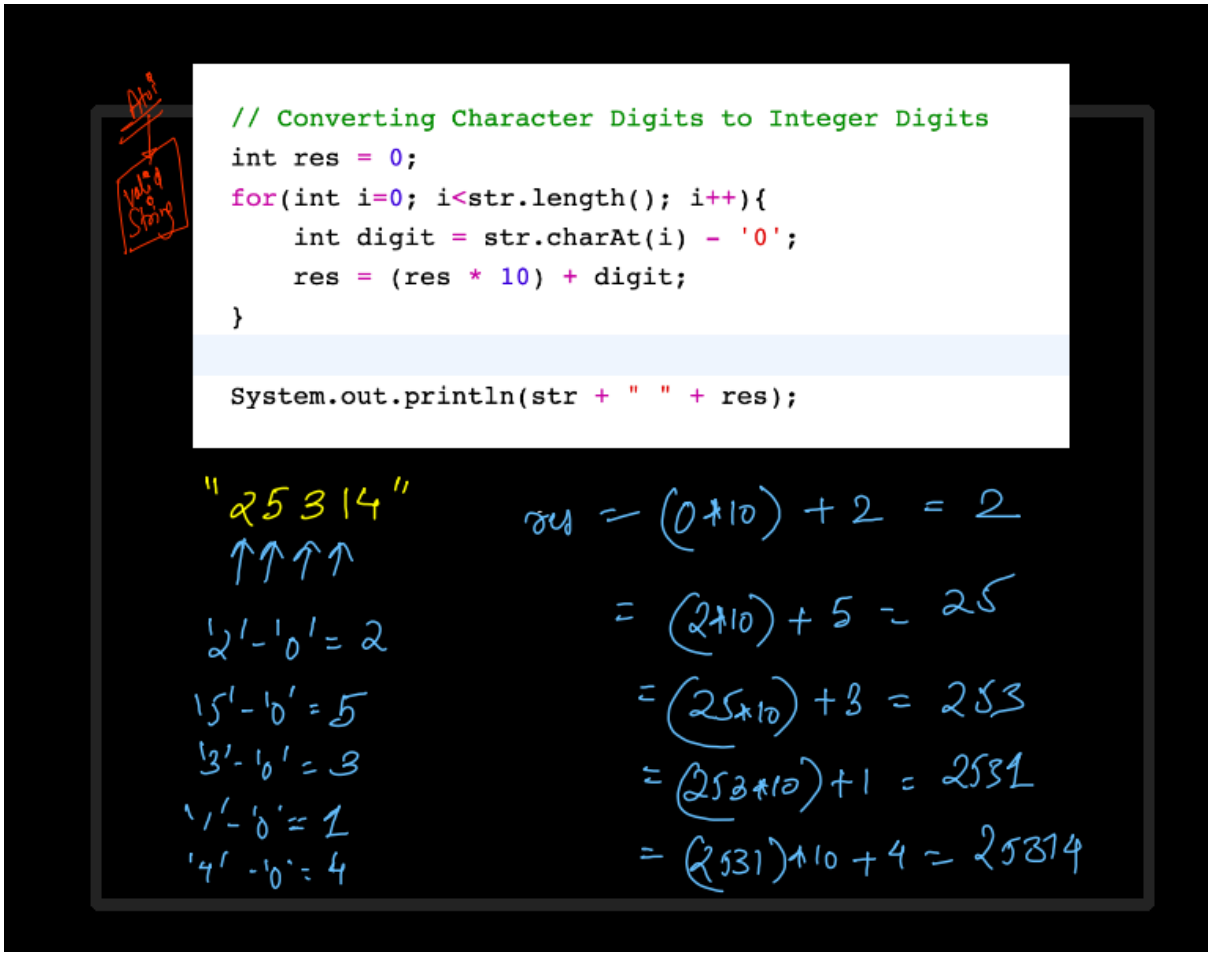
'3' - '0' = 51 - 48 = 3
'2' - '0' = 50 - 48 = 2
'1' - '0' = 49 - 48 = 1
'0' - '0' = 48 - 48 = 0

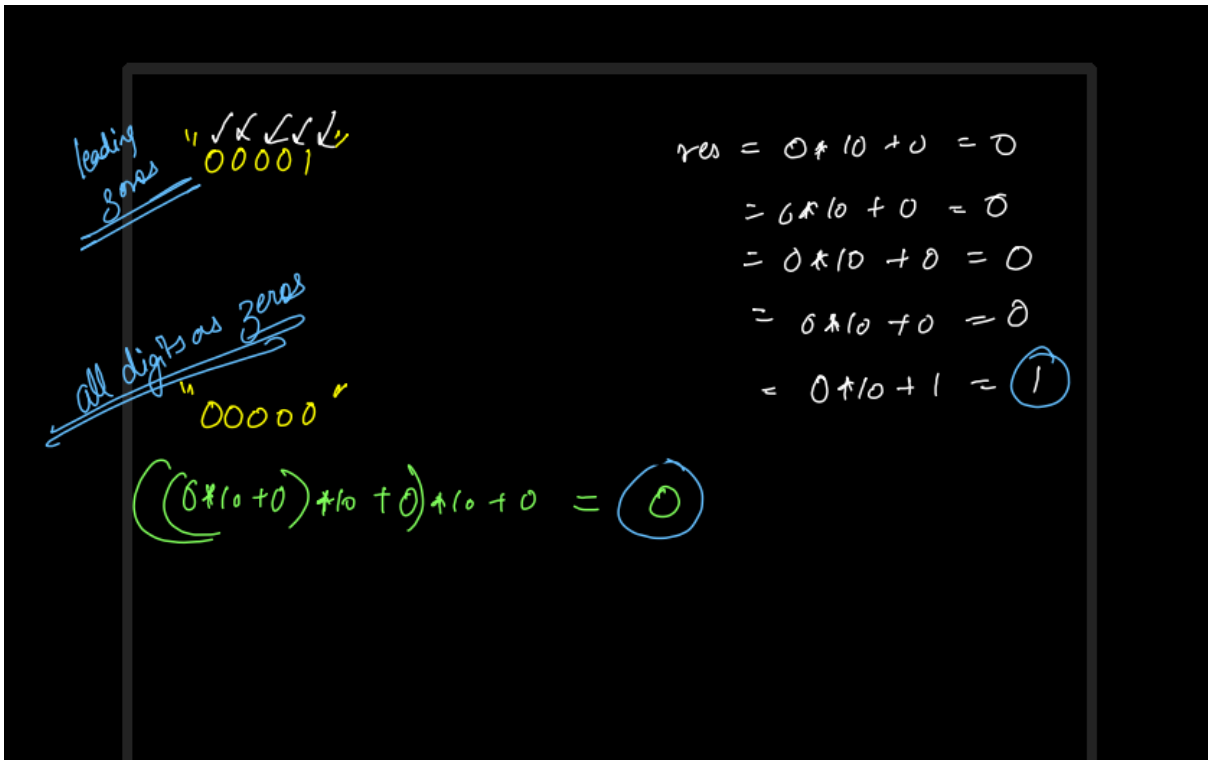
res = 0 +2 = 2

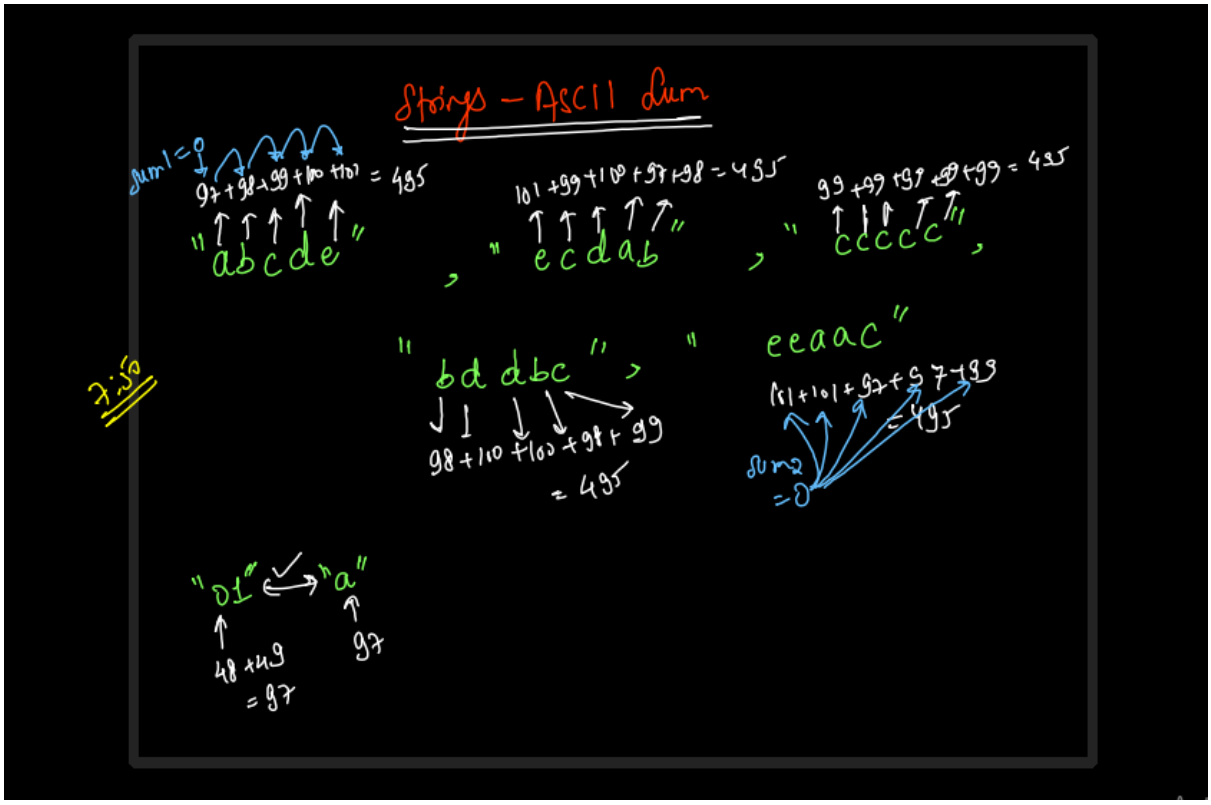
'digit' - '0' = digit

char → integer

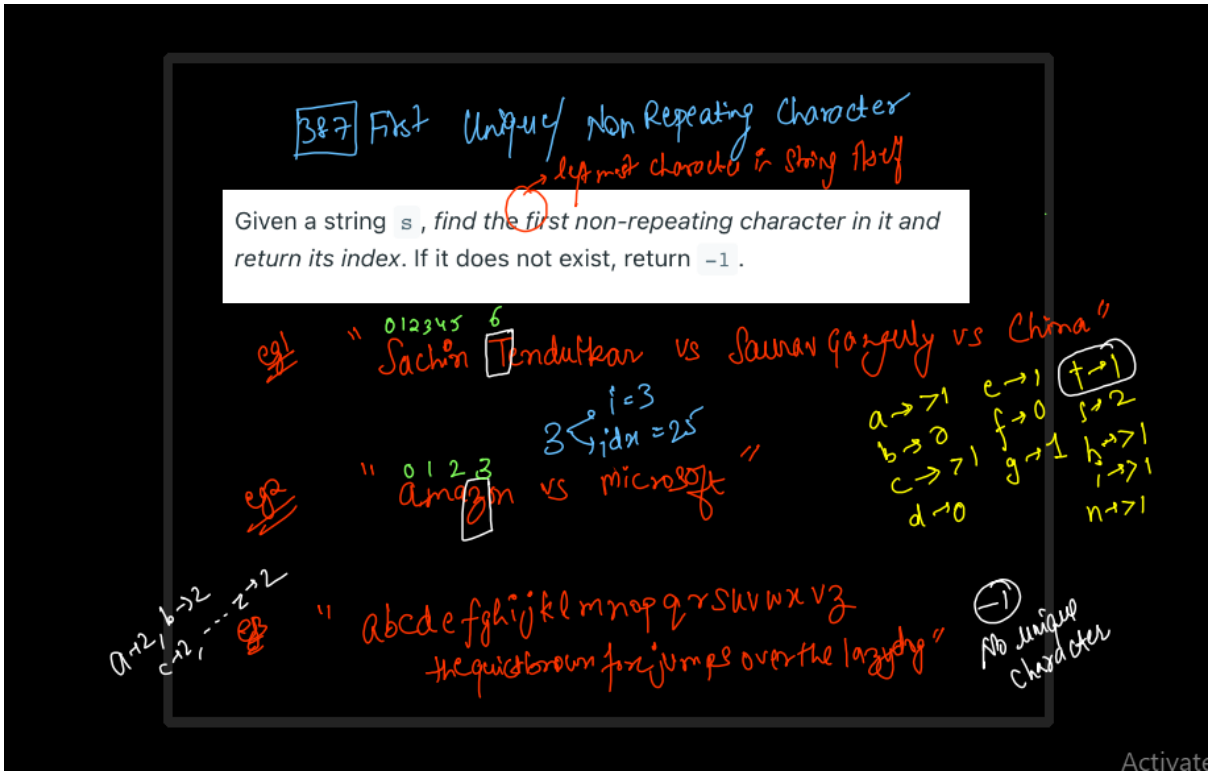

715946
Number = Σ Digits × Place values
= 6*10^0 + 4*10^1 + 9*10^2 + 5*10^3 + 1*10^4 + 7*10^5
= 6*10^0 + 10(4*10^0 + 9*10^1 + 5*10^2 + 1*10^3 + 7*10^4)
= 6*10^0 + 10{4*10^0 + 10{9*10^0 + 10{5*10^0 + 1*10^1 + 7*10^2}}}
= 6*10^0 + 10{4*10^0 + 10{9*10^0 + 10{5*10^0 + 10{1*10^0 + 7*10^1}}}}
7*10+1 = 71
71*10+5 = 715
7150 + 9 = 7159
7159*10 + 4 = 71594
71594*10 + 6 = 715946

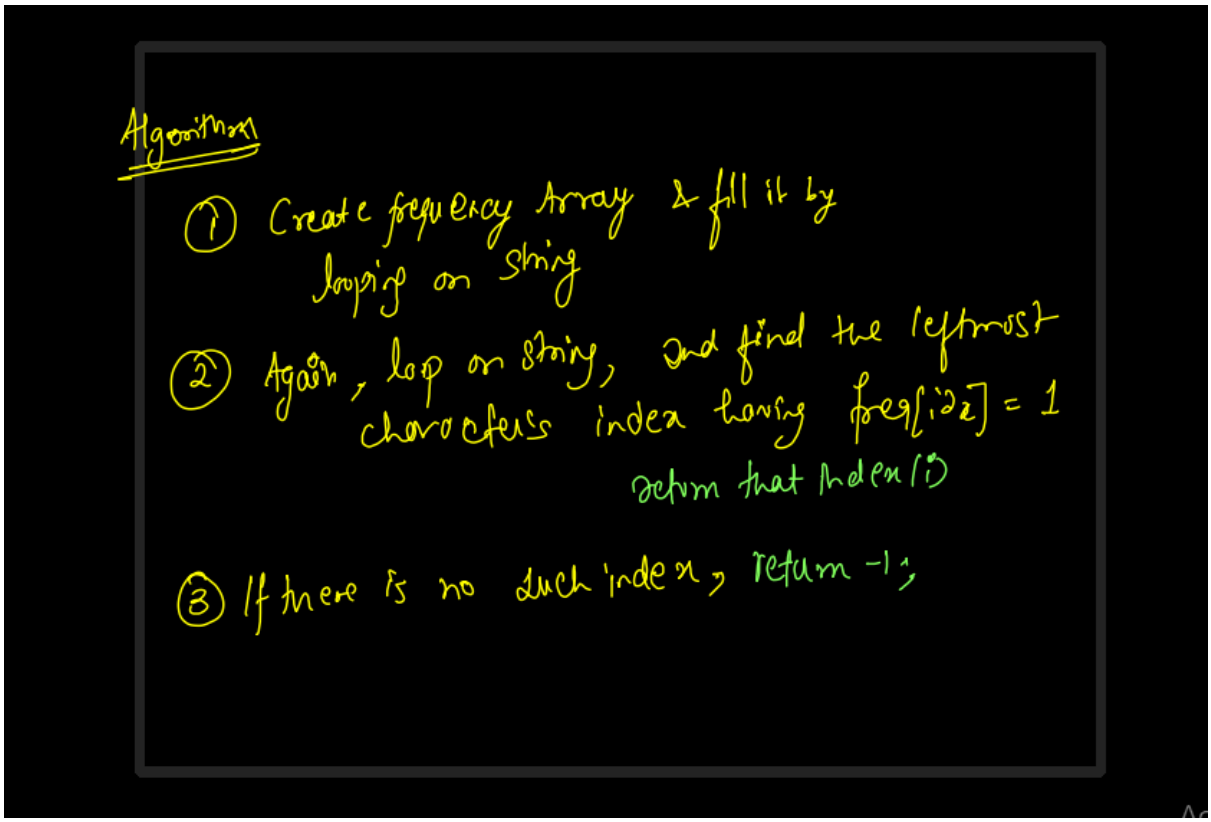

// Converting Character Digits to Integer Digits
int res = 0;
for(int i=0; i<str.length(); i++){
int digit = str.charAt(i) - '0';
res = (res * 10) + digit;
}
System.out.println(str + " " + res);
"25314"
'2'-'0' = 2
'5'-'0' = 5
'3'-'0' = 3
'1'-'0' = 1
'4'-'0' = 4
res = (0*10)+2 = 2
= (2*10)+5 = 25
= (25*10)+3 = 253
= (253*10)+1 = 2531
= (2531)*10+4 = 25314


leading zeros
"00001"
all digits as zeros
"00000"
((0*10+0)*10+0)*10+0 = 0
res = 0*10+0 = 0
= 0*10+0 = 0
= 0*10+0 = 0
= 0*10+0 = 0
= 0*10+1 = 1




```java
// Ascii sum is the sum of ascii codes of all characters in the string.
Scanner scn = new Scanner(System.in);
String s1 = scn.next();
String s2 = scn.next();

int sum1 = 0, sum2 = 0;
for(int i=0; i<s1.length(); i++){
    sum1 = sum1 + s1.charAt(i);
}

for(int i=0; i<s2.length(); i++){
    sum2 = sum2 + s2.charAt(i);
}

if(sum1 == sum2) System.out.println(true);
else System.out.println(false);
```


387 First Unique/ Non Repeating Character
leftmost character in string itself
Given a string s , find the first non-repeating character in it and return its index. If it does not exist, return -1 .
eg1 " Sachin Tendulkar vs Saurav Ganguly vs China "
012345 6
i=3
3 idx=25
eg2 " aimazon vs microsoft "
0 1 2 3
a→1 e→1 t→1
b→0 f→0 s→2
c→1 g→1 h→1
d→0 i→1
n→1
a→2, b→2, ... z→2
eg3 " abcdefghijklmnopqrstuvwxyz
thequickbrownfoxjumpsoverthelazydog "
-1
No unique character

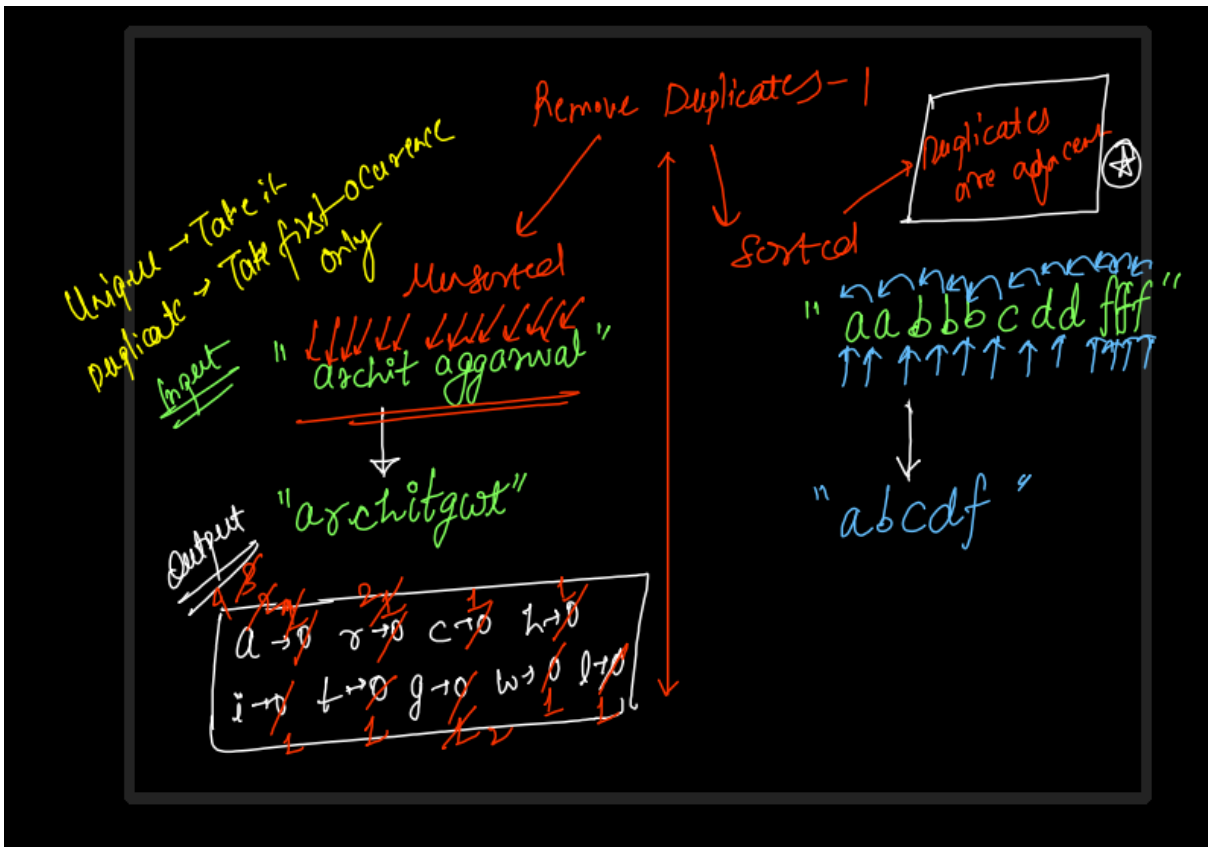

Algorithm
1. Create frequency Array & fill it by looping on string
2. Again, loop on string, and find the leftmost character's index having freq[idx] = 1
return that index(i)
3. If there is no such index, return -1;

```java
int[] freq = new int[26];

for(int i=0; i<s.length(); i++){
    int idx = s.charAt(i) - 'a';
    freq[idx]++;
}

for(int i=0; i<s.length(); i++){
    int idx = s.charAt(i) - 'a';
    if(freq[idx] == 1) return i;
}

return -1;
```

Filling the freq Array

Skip freq[ch] == 1

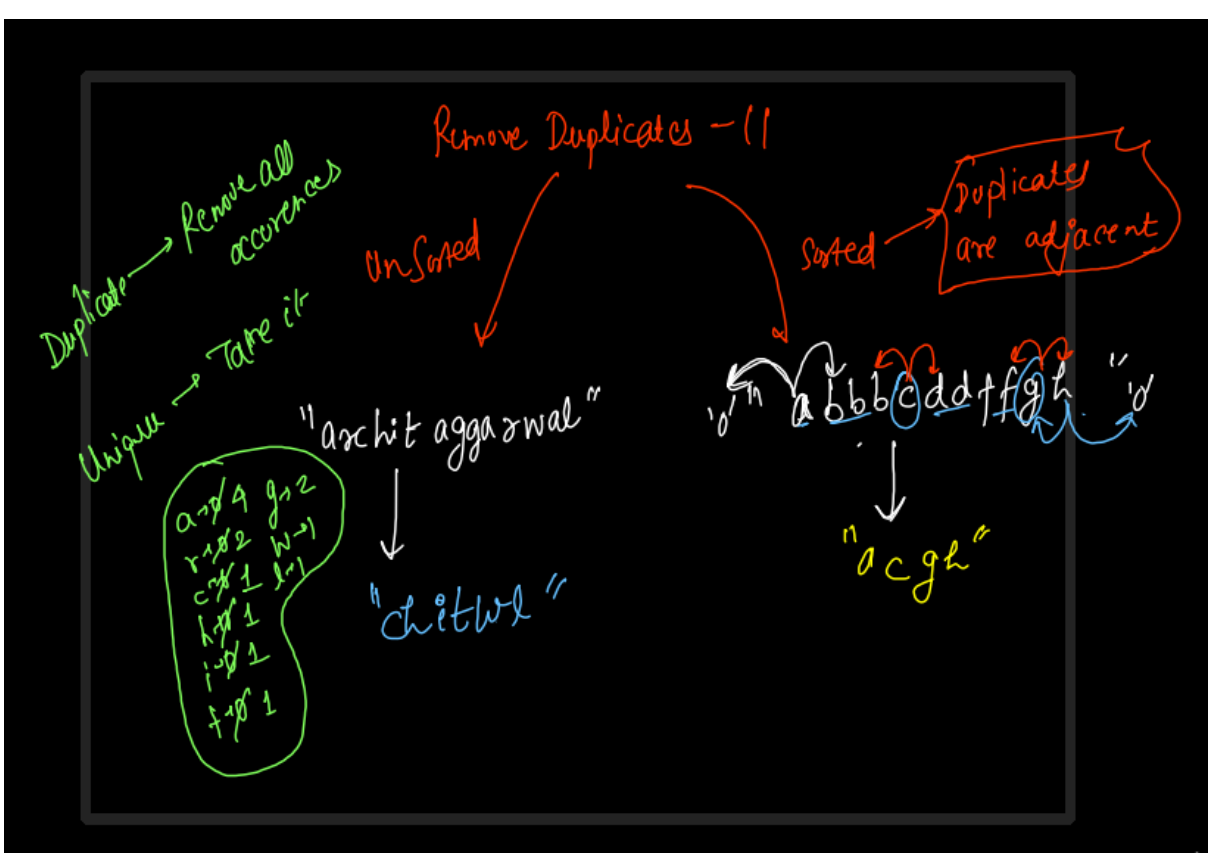
return index in String



```java
// Version - 1 : Unsorted
int[] freq = new int[26];
String output = "";
for(int i=0; i<str.length(); i++){
    int idx = str.charAt(i) - 'a';
    if(freq[idx] == 0){
        output = output + str.charAt(i);
    }

    freq[idx]++;
}
System.out.println(output);
```

N

```java
// Version - 1: Sorted
String output = "";
for(int i=0; i<str.length(); i++){
    // Adding the First Occurence of each character
    if(i == 0 || str.charAt(i - 1) != str.charAt(i)){
        output = output + str.charAt(i);
    }
}
System.out.println(output);
```



```
// Version - 2: Unsorted
int[] freq = new int[26];
for(int i=0; i<str.length(); i++){
    int idx = str.charAt(i) - 'a';
    freq[idx]++;
}

String output = "";
for(int i=0; i<str.length(); i++){
    int idx = str.charAt(i) - 'a';
    if(freq[idx] == 1){
        output = output + str.charAt(i);
    }
}
System.out.println(output);
```

N
N + f
= 2xN

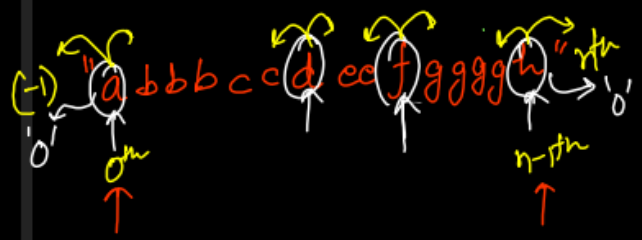
Archi aggarwal

a→4 r→2 c→1
h→1 i→1 t→1
g→2 w→1 l→1

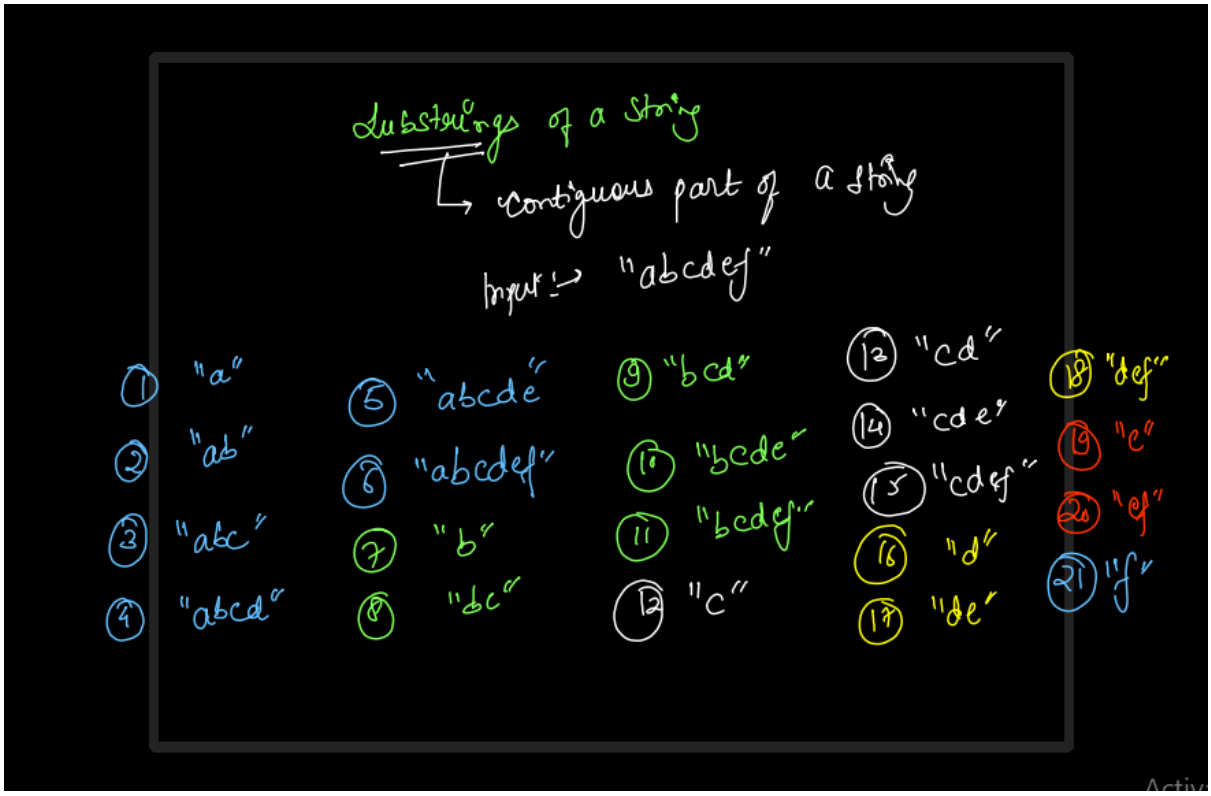
"chiftwl"

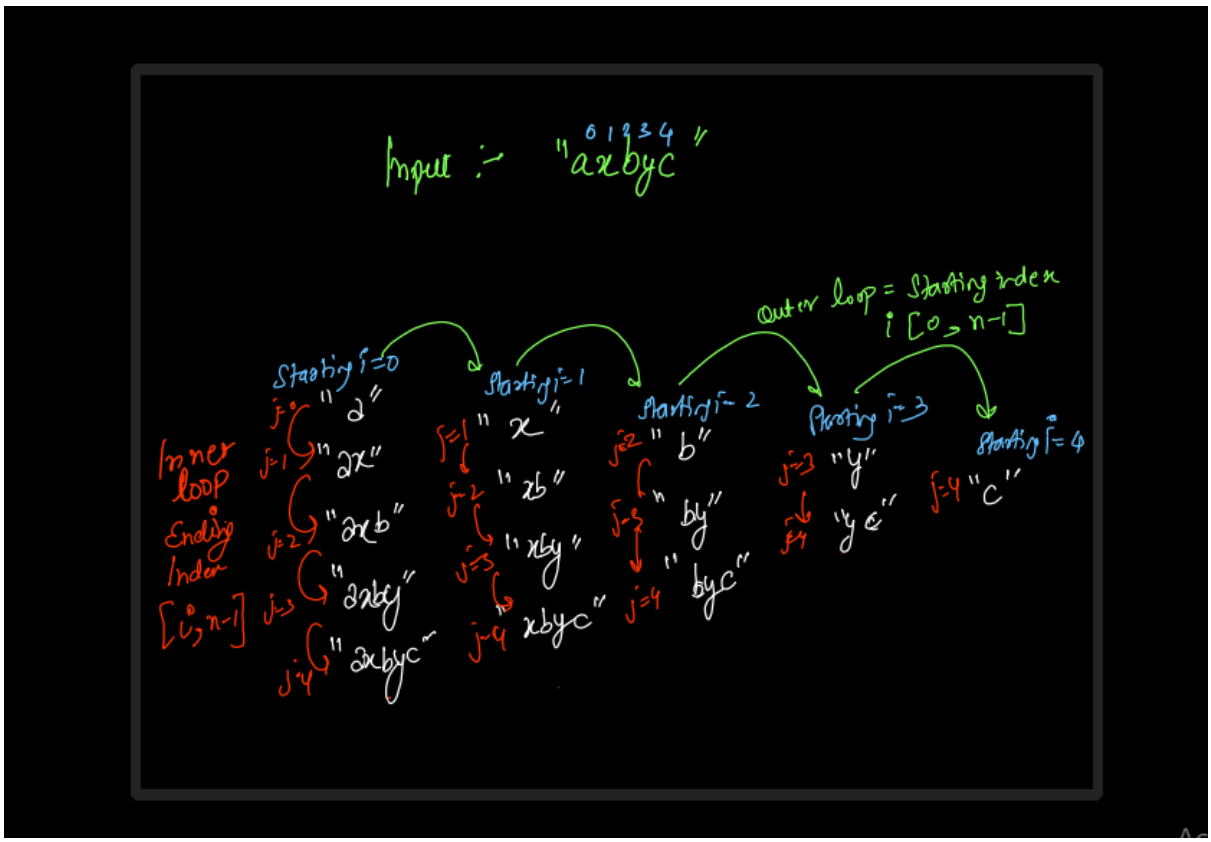
Ternary Operator

(cond) ? truestat : falsestat ;




```
if (i-1>=0)
  left = s[i-1];
else
  left = '0';
```


Substrings of a string
↳ contiguous part of a string
Input :- "abcdef"
1 "a"
2 "ab"
3 "abc"
4 "abcd"
5 "abcde"
6 "abcdef"
7 "b"
8 "bc"
9 "bcd"
10 "bcde"
11 "bcdef"
12 "c"
13 "cd"
14 "cde"
15 "cdef"
16 "d"
17 "de"
18 "def"
19 "e"
20 "ef"
21 "f"


Input :- "axbyc"
0 1 2 3 4
Outer loop = Starting index
i [0, n-1]
Starting i=0
Starting i=1
Starting i=2
Starting i=3
Starting i=4
Inner loop Ending Index [i, n-1]
"a"
"ax"
"axb"
"axby"
"axbyc"
"x"
"xb"
"xby"
"xbyc"
"b"
"by"
"byc"
"y"
"yc"
"c"

substrings of a string

```
for(int st=0; st<n; st++){
    String substr = "";
    for(int end=st; end<n; end++){
        substr = substr + str.charAt(end);
        System.out.println(substr);
    }
}
```

O(N*N*N)
st, end, +concatenate

"abcde"
st='a' (0)
end=0 "a"
end=1 "ab"
end=2 "abc"
end=3 "abcd"
end=4 "abcde"

st='b' (1)
end="1" "b"
end=2 "bc"
end=3 "bcd"
end=4 "bcde"

st='c' (2)
end=2 "c"
end=3 "cde"
end=4 "cde"

st='d' (3)
end=3 "d"
end=4 "de"

st='e' (4)
end=4 "e"

Inbuilt substring method in Java

Str.substring(startingIndex, endingIndex)
startingIndex: it is included
endingIndex: it is excluded

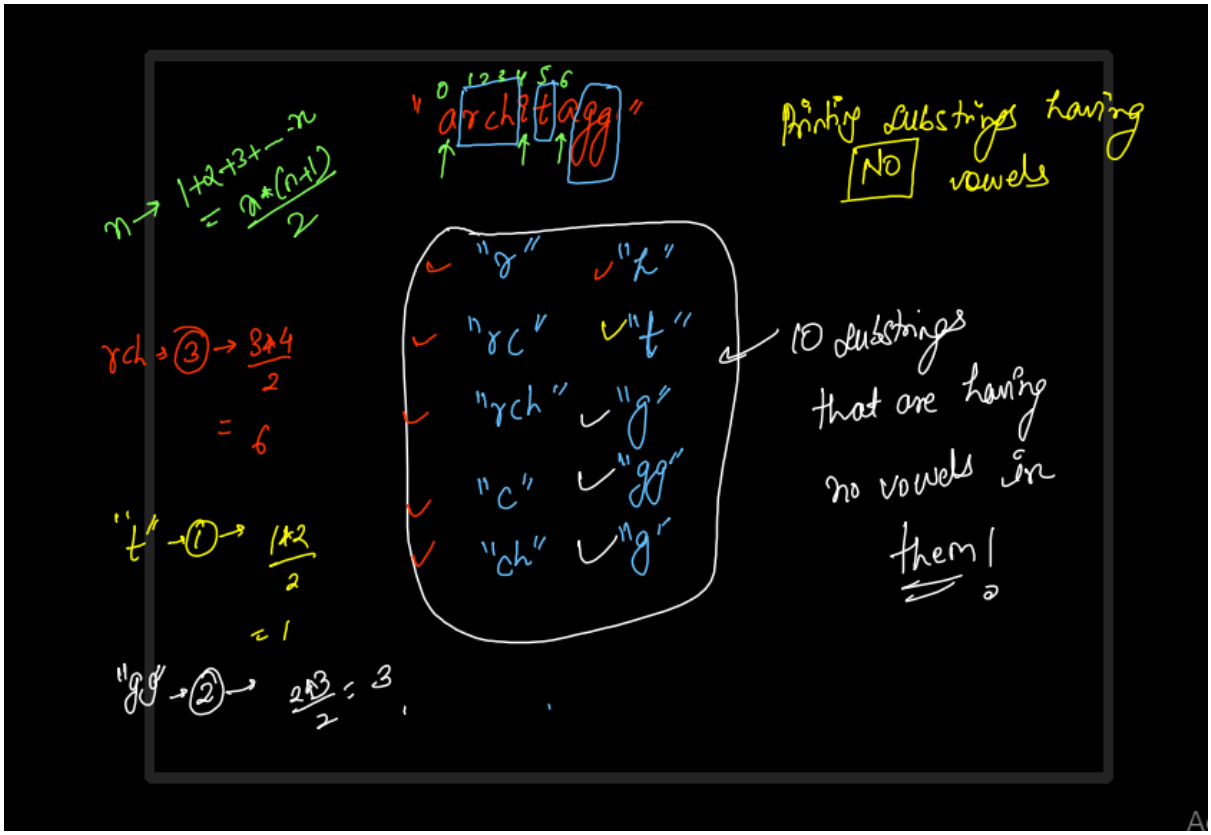
"abcde" (0 1 2 3 4)
str → Str.substring(1, 3) → "bc"

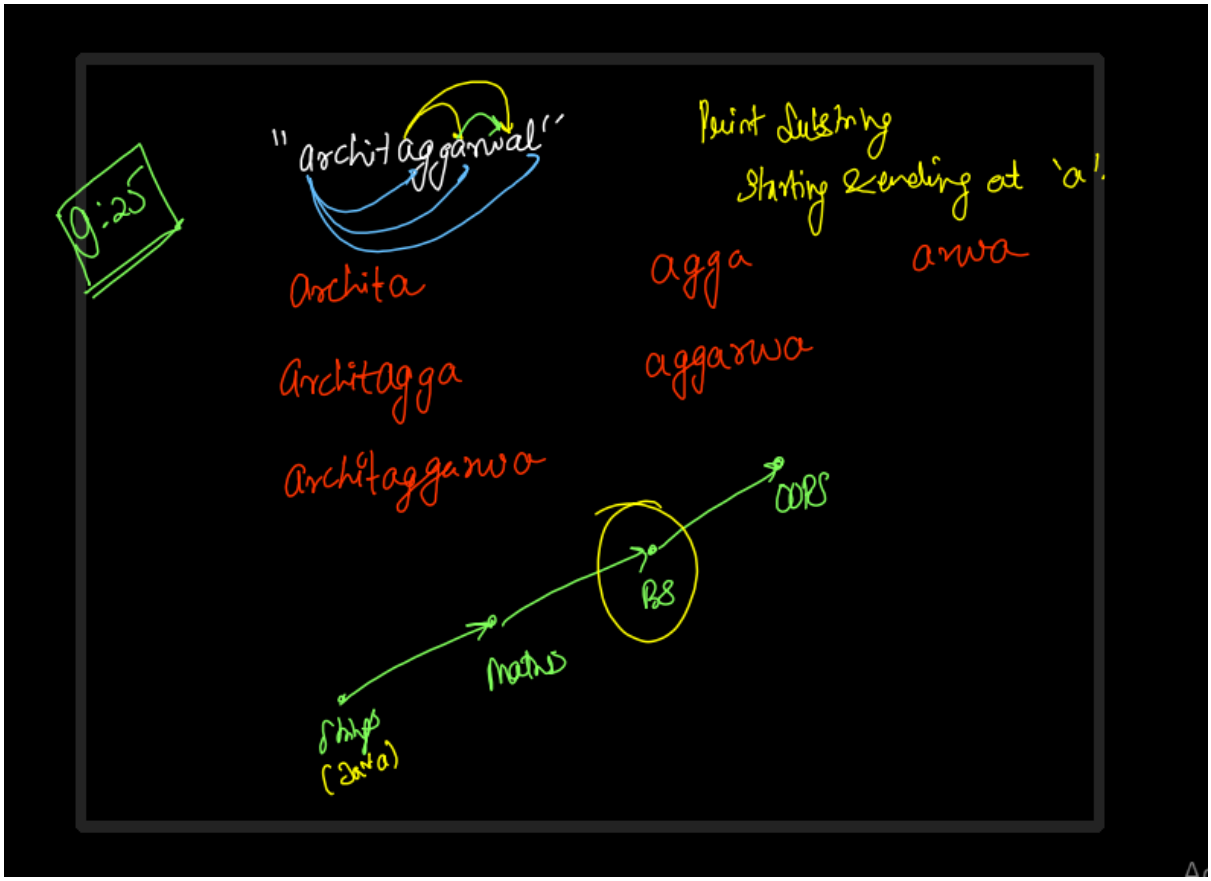
"abcdefg" (0 1 2 3 4 5 6)
Str → str.substring(2, 6) → "cdef"

"abcdef" (0 1 2 3 4 5)
str.substring(3, 4) → "d"

"abcdef" (0 1 2 3 4 5 6)
Str.substring(0,6) → "abcdef"
Str.substring(3,3) → ""

```
for(int st=0; st<n; st++){
    for(int end=st; end<n; end++){
        System.out.println(str.substring(st, end + 1));
    }
}
```


n → 1+2+3+...n = n*(n+1)/2
"architagg"
0 1 2 3 4 5 6
Printing substrings having NO vowels
"r" "h"
"rc" "t"
"rch" "g"
"c" "gg"
"ch" "g"
10 substrings that are having no vowels in them!
rch → 3 → 3*4/2 = 6
"t" → 1 → 1*2/2 = 1
"gg" → 2 → 2*3/2 = 3


9:25
"architaggarwal"
Print substring starting & ending at 'a'.
archita
architagga
architaggarwa
agga
aggarwa
arwa
CORS
RS
Maths
Philips (Sweta)

String Compression → Reducing length

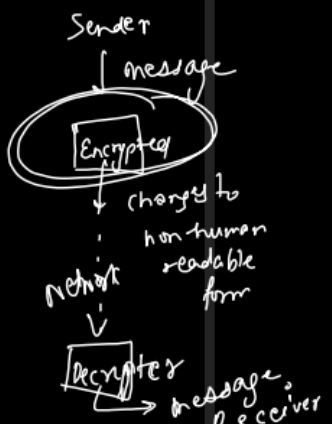
Input " a a a b b c c c c d a a a a b b b e "

Output " a3 b2 c4 d a4 b3 e "

aaaaaaa bbbbb ccccc de

a 7 b 5 c 4 de

can Not decrypted



" a a a b b c c c c d a a a a b b b e "

7:55

char = 'a' 'b' 'c' 'd' 'a' 'b' 'e'

count = 3 2 4 1 4 3 1

"a3b2c4d1a4b3e1"

```java
String encode(String str)
{
    if(str.length() == 0) return "";

    char ch = str.charAt(0);
    int freq = 1;
    String output = "";

    for(int i=1; i<str.length(); i++){
        if(str.charAt(i - 1) != str.charAt(i)){
            output = output + ch + freq;
            ch = str.charAt(i);
            freq = 1;
        } else {
            freq++;
        }
    }

    output = output + ch + freq;
    return output;
}
```

String Compression

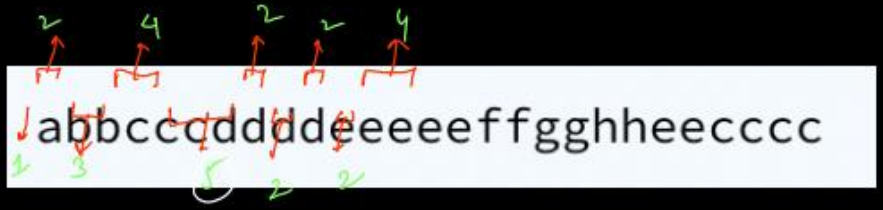
"aaabbbccddd"

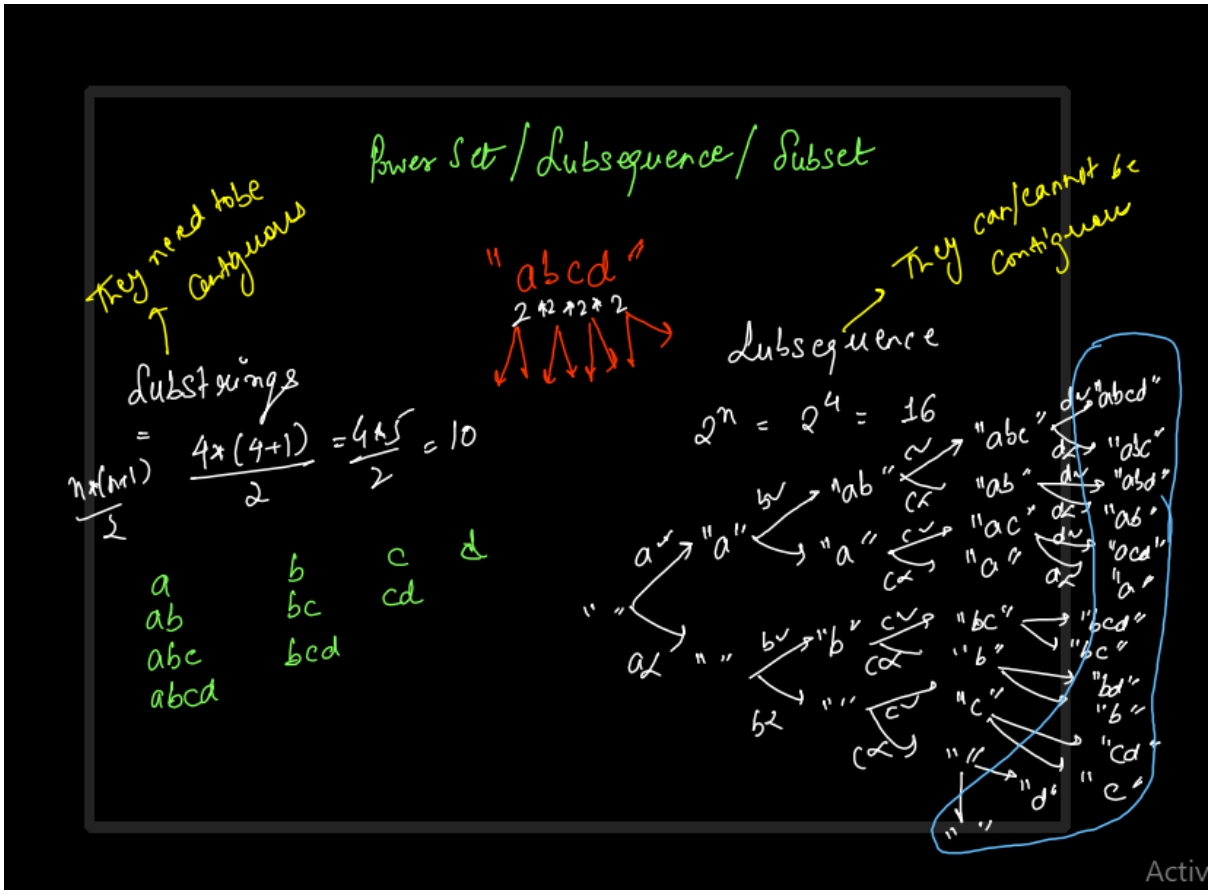
output = "a3b4c2d3"

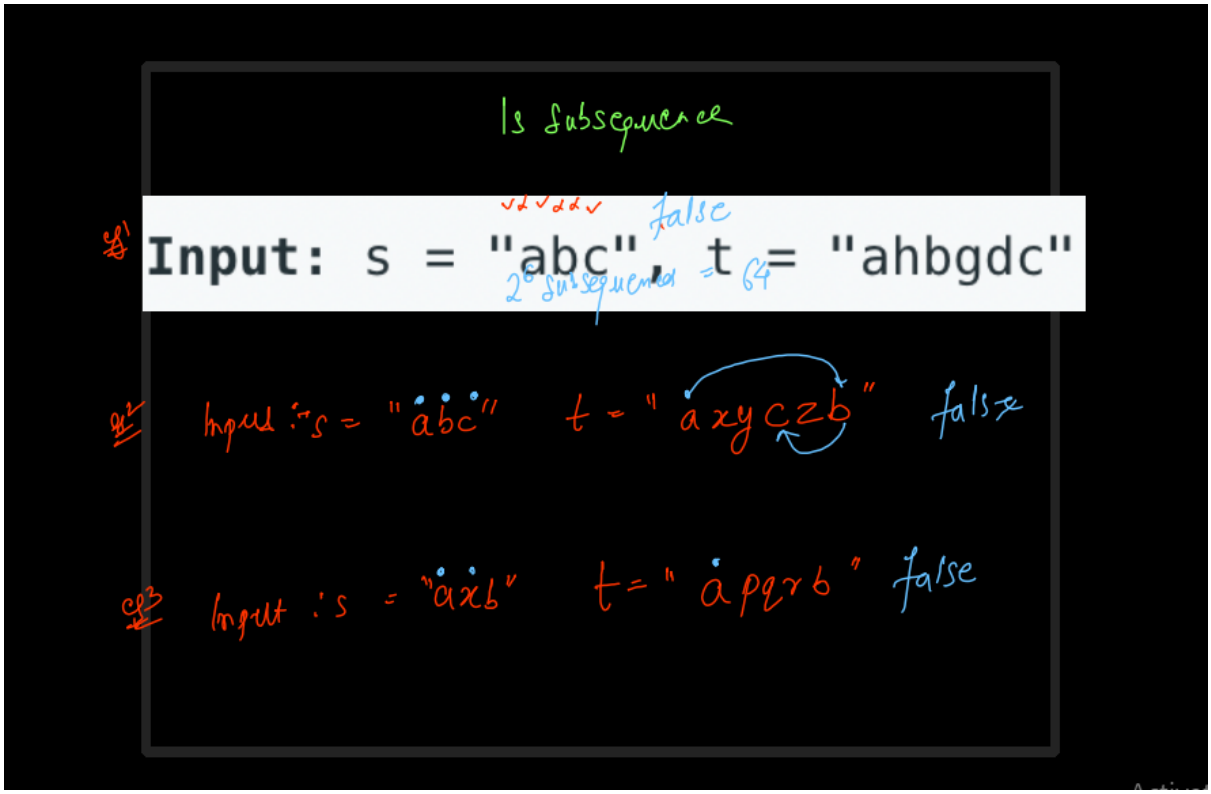
ch = 'a' 'b' 'c' 'd'

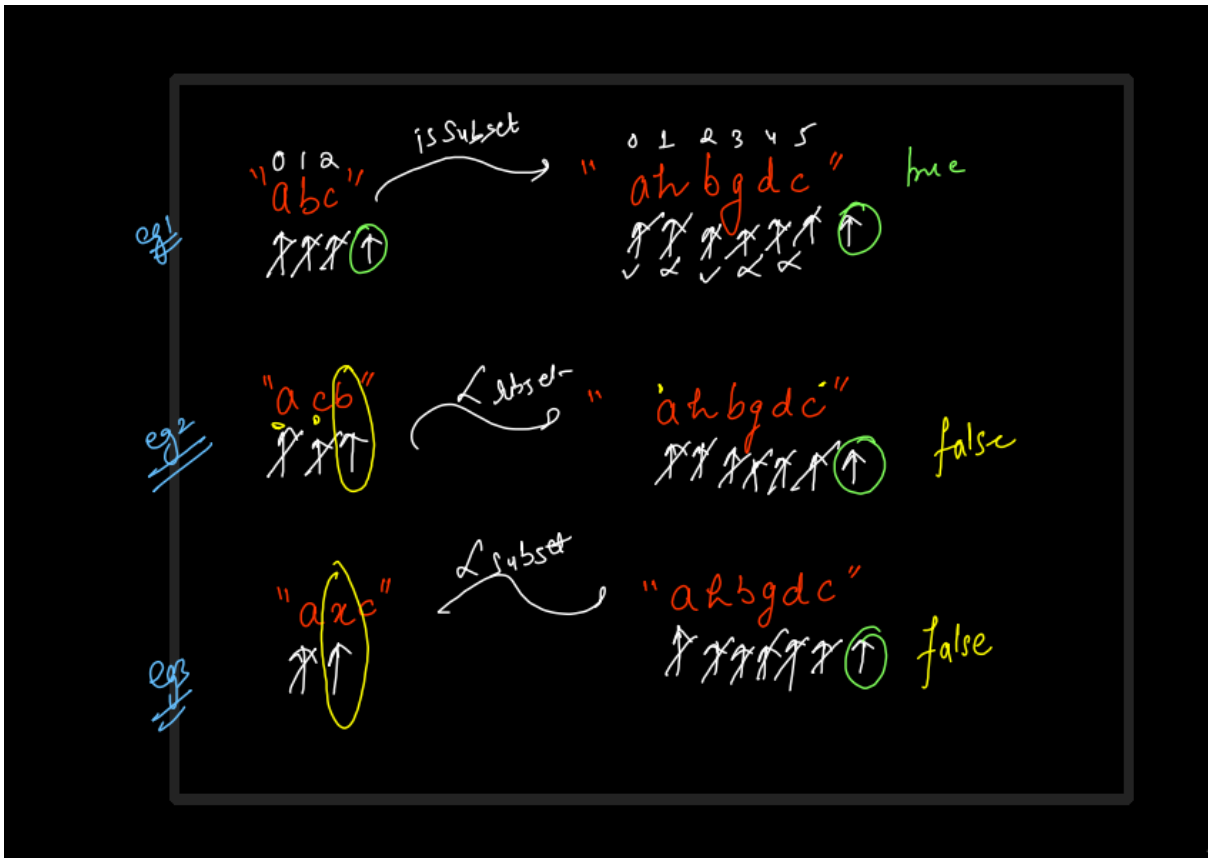
freq = 3 4 2 3

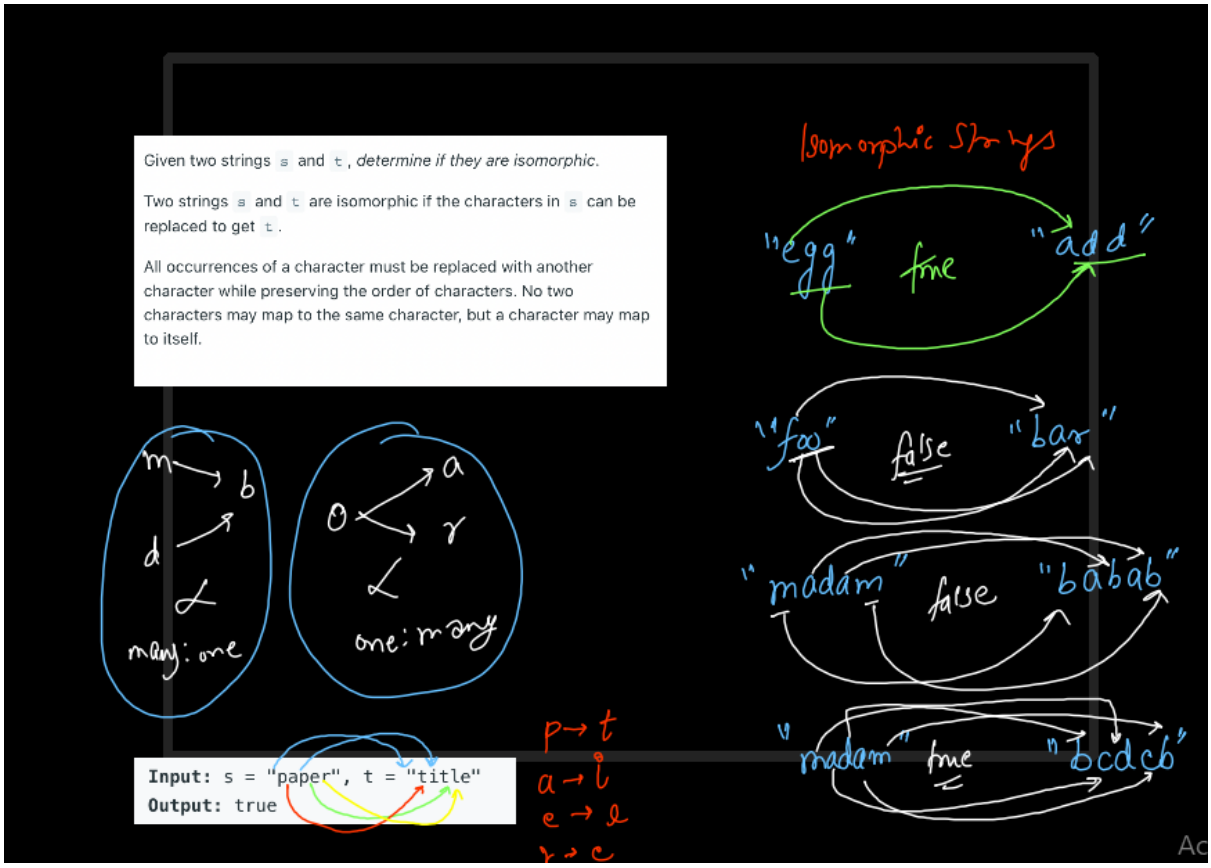
Power of a string

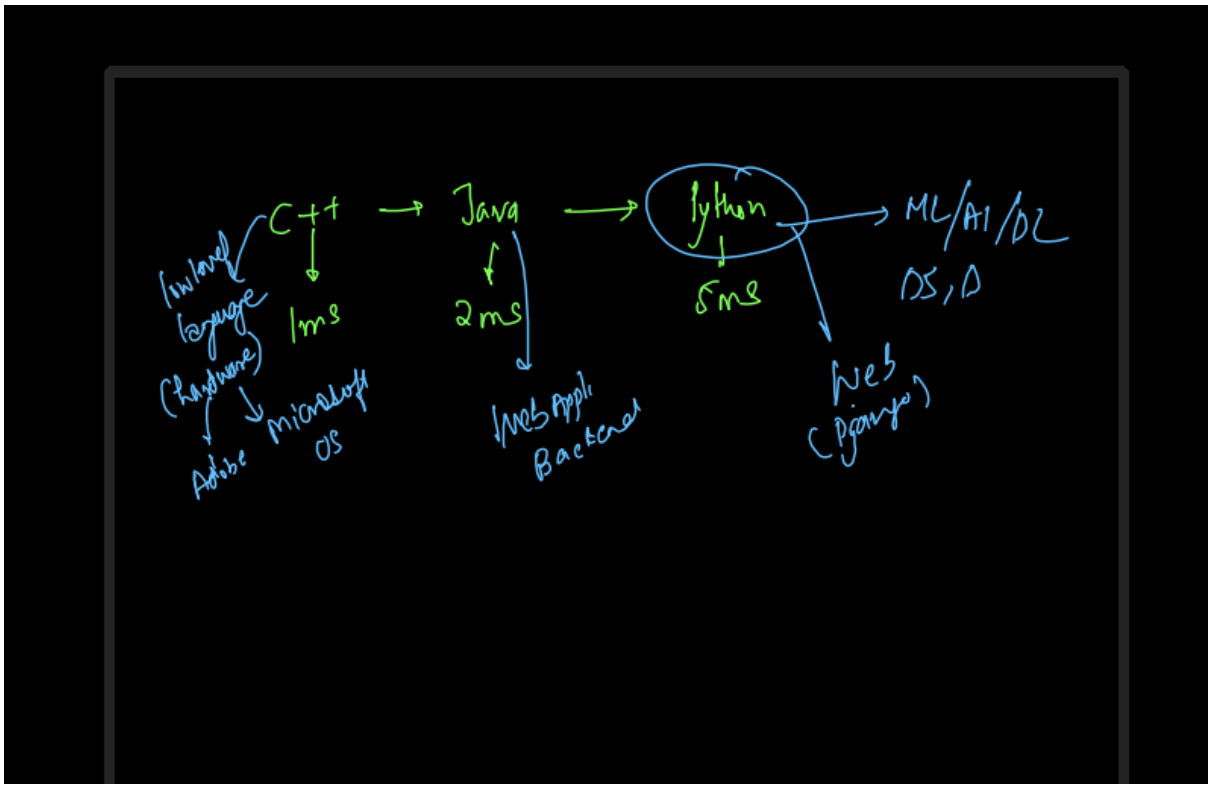


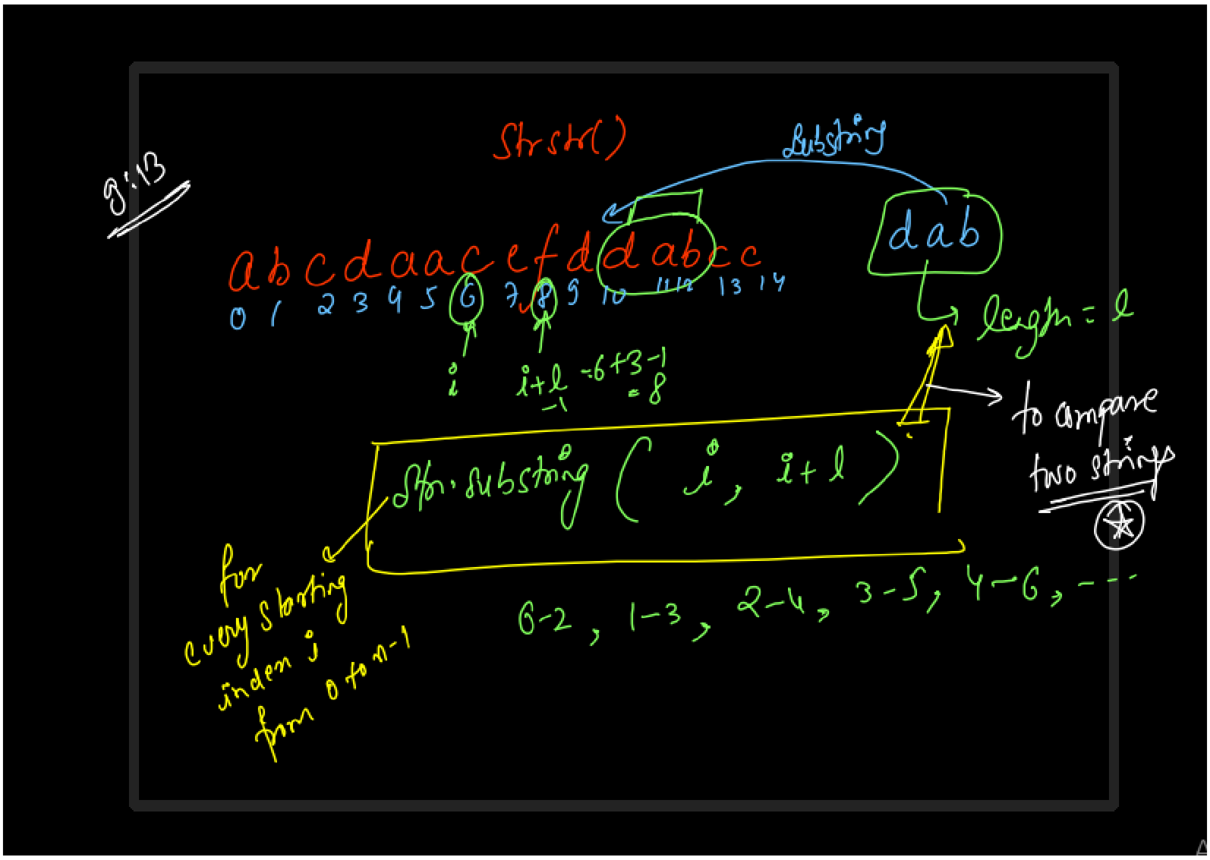

Power Set / Subsequence / Subset
They need to be Contiguous
They can/cannot be Contiguous
"abcd"
2 * 2 * 2 * 2
Substrings
= n*(n+1)/2 = 4*(4+1)/2 = 4*5/2 = 10
a b c d
ab bc cd
abc bcd
abcd
Subsequence
2^n = 2^4 = 16
"abc"
"ab"
"a"
"ac"
"bc"
"b"
"c"
"abcd"
"abd"
"acd"
"bcd"
"d"
"e"

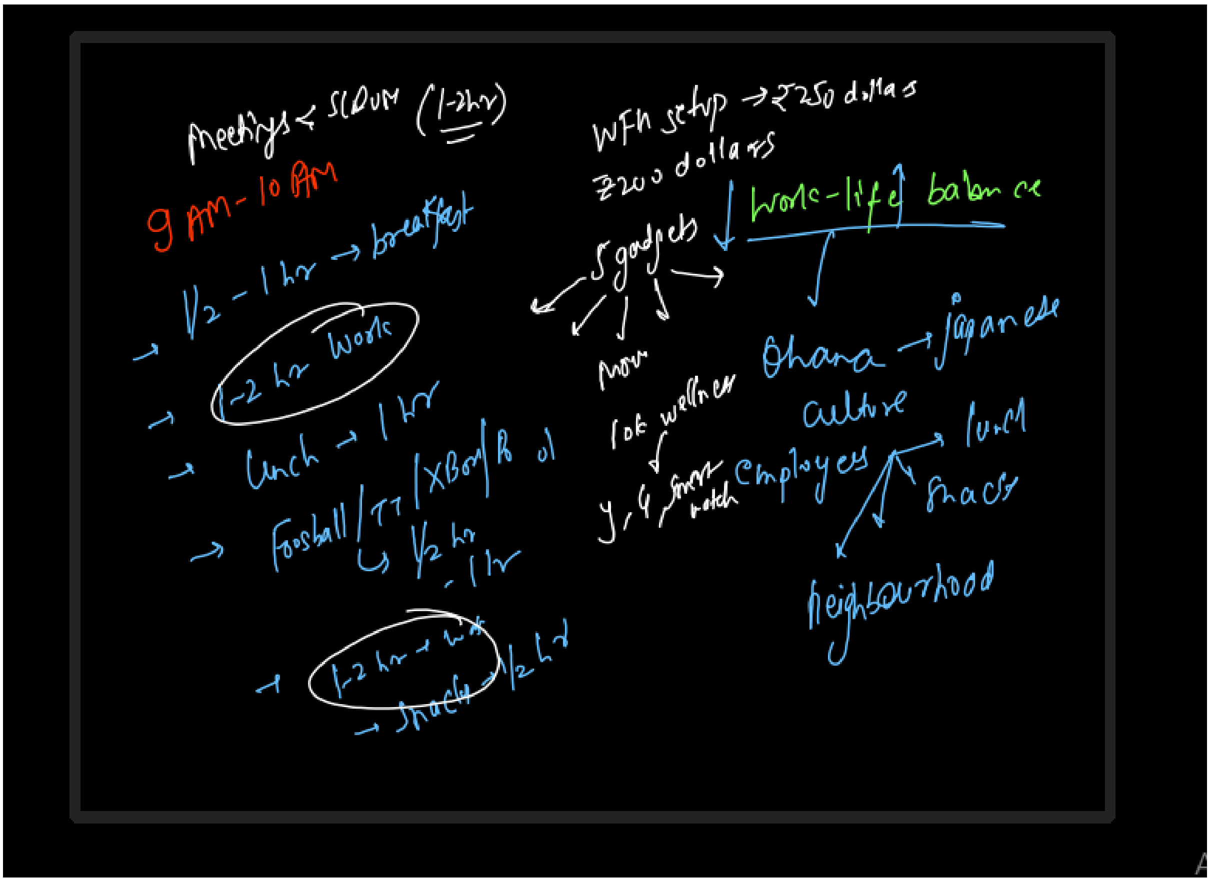

Is subsequence
eg1
Input: s = "abc", t = "ahbgdc"
false
2^6 subsequences = 64
eg2
Input: s = "abc" t = "axyczb" false
eg3
Input: s = "axb" t = "apqrb" false


eg1
"abc"
0 1 2
isSubset
"ahbgdc"
0 1 2 3 4 5
true
eg2
"acb"
isSubset
"ahbgdc"
false
eg3
"axc"
isSubset
"ahbgdc"
false


Isomorphic Strings
Given two strings s and t, determine if they are isomorphic.
Two strings s and t are isomorphic if the characters in s can be replaced to get t.
All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.
"egg" true "add"
"foo" false "bar"
"madam" false "babab"
"madam" true "bcdcb"
many: one
one: many
Input: s = "paper", t = "title"
Output: true
p → t
a → i
e → l
r → e

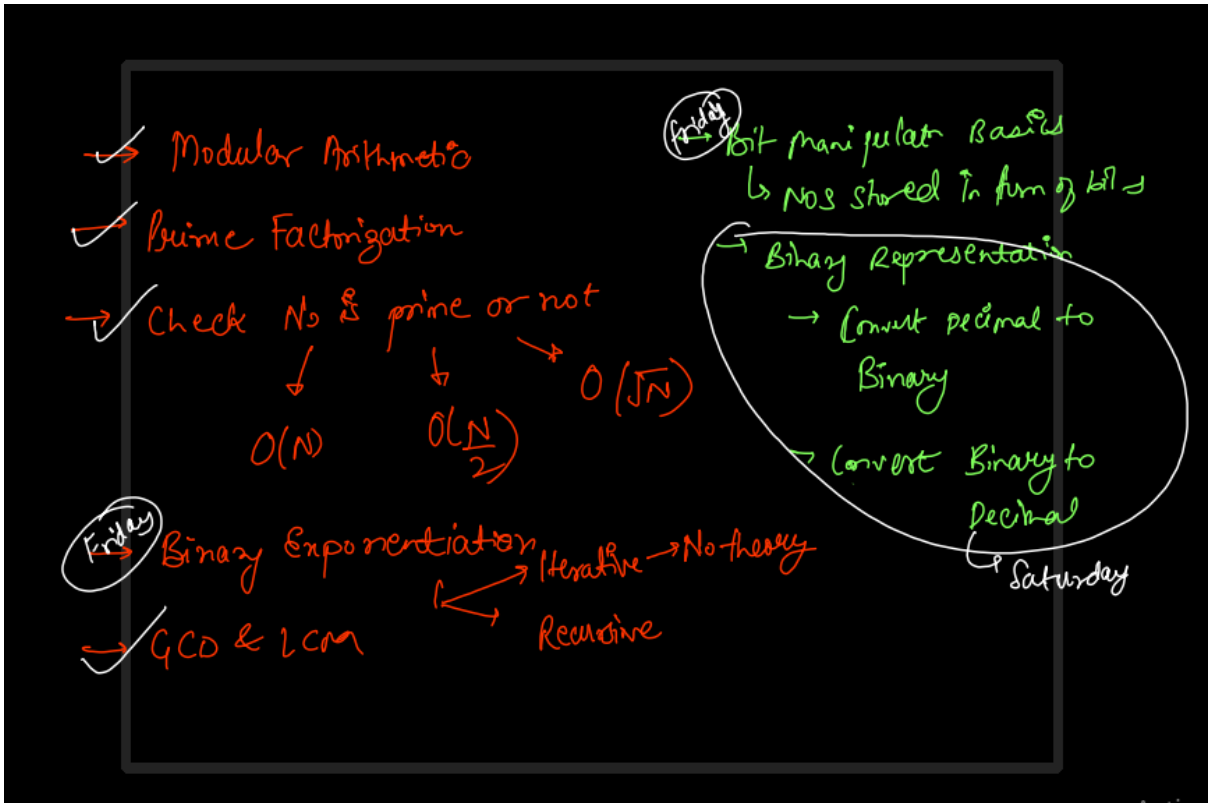

C++
Java
Python
ML/AI/DL
DS, A
1ms
2ms
5ms
Adobe
Microsoft
OS
Web Appli
Backend
Web
(Django)


strstr()
substring
9:13
abcdaacefddabcc
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
dab
length = l
i
i+l-1 = 6+3-1 = 8
str.substring (i, i+l)
to compare two strings
for every starting index i from 0 to n-1
0-2 , 1-3 , 2-4 , 3-5 , 4-6 , - - -


Meetings < SCRUM (1-2hr)
9 AM - 10 AM
1/2 - 1 hr → breakfast
1-2 hr work
lunch → 1 hr
Foosball / TT / XBox / Pool
1/2 hr - 1 hr
1-2 hr → work
→ snacks → 1/2 hr
WFH setup → 250 dollars
200 dollars
5 gadgets
work-life balance
Ohana → japanese culture
employees
lunch
snacks
neighbourhood

→ Bruteforce → O(N*M)

```
for(int i=0; i<= actual.length()-required.length(); i++){
    if(actual.substring(i, i + required.length()).equals(required) == true){
        return i;
    }
}

return -1;
```



Modulo operation

$a \% b$ → It gives remainder when a is divided by b.

If $a < b \Rightarrow a \% b = a$

$1 \% 100 = 1$

$2 \% 100 = 2$

$3 \% 100 = 3$

$4 \% 100 = 4$

$\vdots$

$99 \% 100 = 99$

$1 \% \text{Integer.MAX\_VALUE} = 1$

$2 \% \infty = 2$

$3 \% \infty = 3$

$4 \% \infty = 4$

$\vdots$

$(\infty - 1) = 2^{31} - 2 \% 2^{31} - 1 = 2^{31} - 2$

Remainders are of uniform

Division by composite no

$11 \% 10 = 1$

$12 \% 10 = 2$

$13 \% 10 = 3$

$14 \% 10 = 4$

$15 \% 10 = 5$

$16 \% 10 = 6$

$17 \% 10 = 7$

$18 \% 10 = 8$

$19 \% 10 = 9$

$20 \% 10 = 0$

$21 \% 10 = 1$

$22 \% 10 = 2$

$23 \% 10 = 3$

$24 \% 10 = 4$

$25 \% 10 = 5$

$26 \% 10 = 6$

$27 \% 10 = 7$

$28 \% 10 = 8$

$29 \% 10 = 9$

$30 \% 10 = 0$

Division by prime no

$14 \% 13 = 1$

$15 \% 13 = 2$

$16 \% 13 = 3$

$17 \% 13 = 4$

$\vdots$

$25 \% 13 = 12$

$26 \% 13 = 0$

$27 \% 13 = 1$

$28 \% 13 = 2$

$29 \% 13 = 3$

$\vdots$

$38 \% 13 = 12$

$39 \% 13 = 0$

→ Modular operation should be performed using a "large" and "prime" no to get non-uniform distribution.

88, 91, 93, ...

8 bytes * 135 core * 2
4 bytes * 135 core * 2

9319117887 → 1911 7887 int

you can't store it in a integer

Properties of modular Arithmetic

(1) $(a+b)\%m = (a\%m + b\%m)\%m$

eg (1) $(7+12)\%10$ → $19\%10 = 9$

$(7\%10 + 12\%10)\%10 = (7+2)\%10 = 9\%10 = 9$

(2) $(100+139)\%3$ → $239\%3 = 2$

$(100\%3 + 139\%3)\%3 = (1+1)\%3 = 2$

② $(a * b) \% m = ((a \% m) * (b \% m)) \% m$

eg① $(10 * 12) \% 7$ (a, b, m) $\longrightarrow 120 \% 7 = 1$

$((10 \% 7) * (12 \% 7)) \% 7$

$= (3 * 5) \% 7 = 15 \% 7$

$= 1$

eg② $(13 * 17) \% 5 \longrightarrow 221 \% 5 = 1$

$((13 \% 5) * (17 \% 5)) \% 5 = (3 * 2) \% 5$

$= 1$

③ $(a - b) \% m = (a \% m - b \% m + m) \% m$

to avoid negative we getting modulo

eg① $(39 - 11) \% 4 = 28 \% 4 = 0$

$(39 \% 4 - 11 \% 4 + 4) \% 4 = (3 - 3 + 4) \% 4$

$= 4 \% 4 = 0$

eg② $(12 - 39) \% 4 \longrightarrow -27 \% 4 = 1$

$(12 \% 4 - 39 \% 4 + 4) \% 4$

$= (0 - 3 + 4) \% 4 = 1 \% 4 = 1$

$13 = 4*3 + 1$

$d = m*x + r$ (r: $d \% m$)

$4 \overline{)13} \; 3$ (12, 1)

$15 = 4*3+3$

$4 \overline{)15} \; 3$ (12, 3)

$4 \overline{)2} \; 0$ (0, 2)

$2 \% 4 = 2$

$2 = 4*0 + 2$

dvd dvs quot rem

This is not applicable

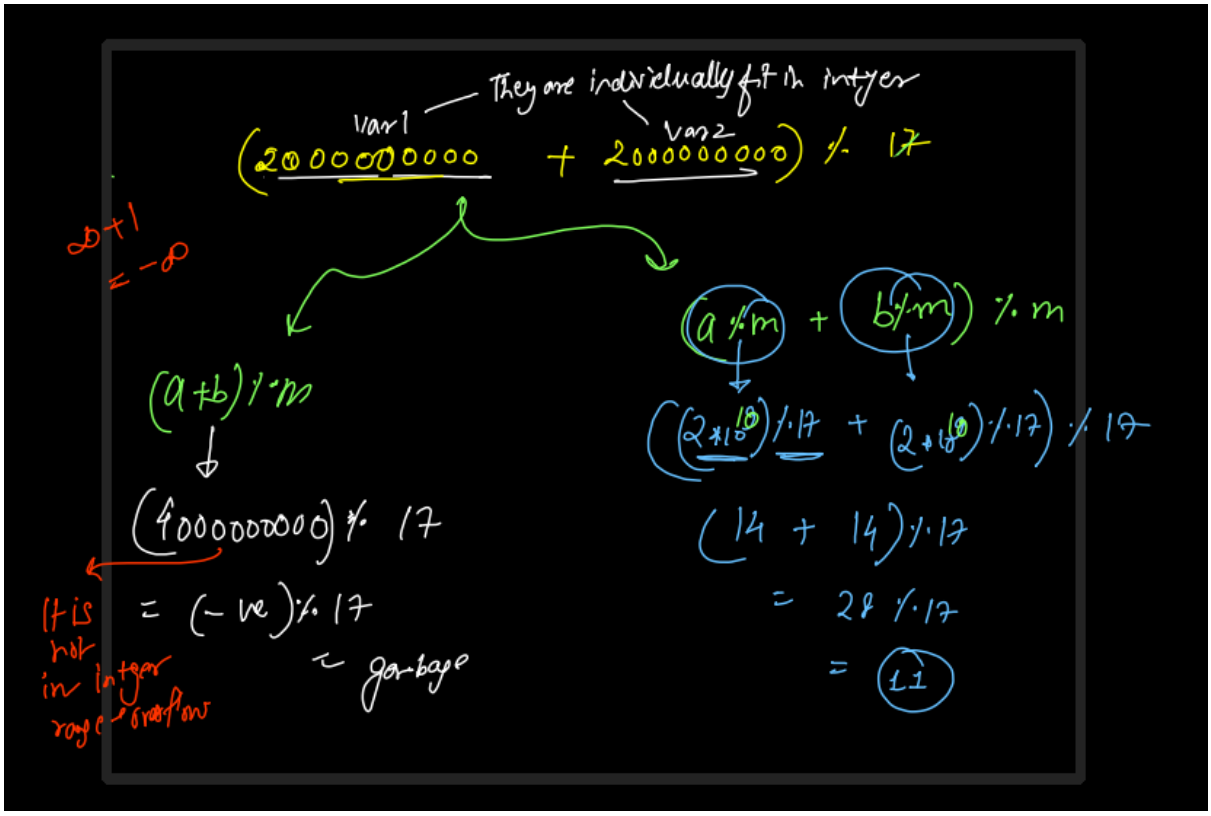
(4) $(a/b) \% m \neq ((a \% m) / (b \% m)) \% m$

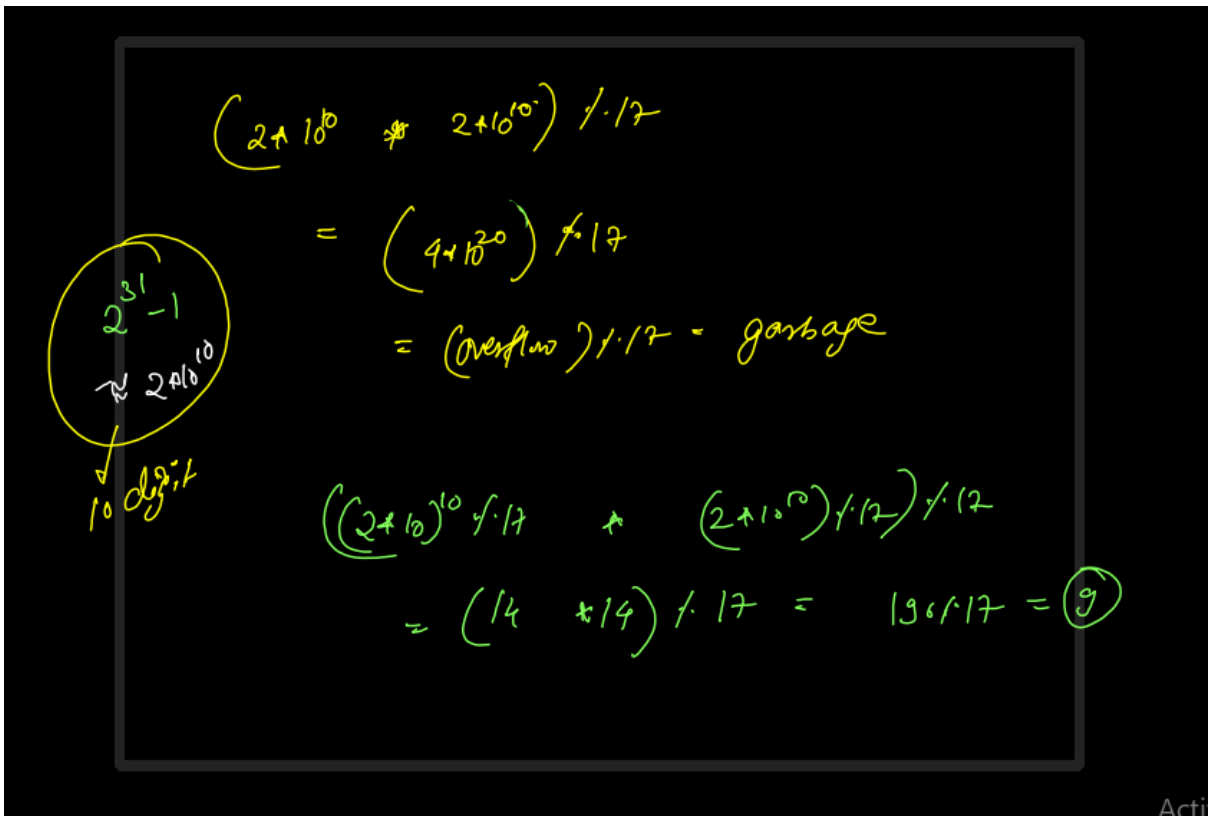
eg $(39/17) \% 4 = (2 \cdot 9) \% 4 = 2$

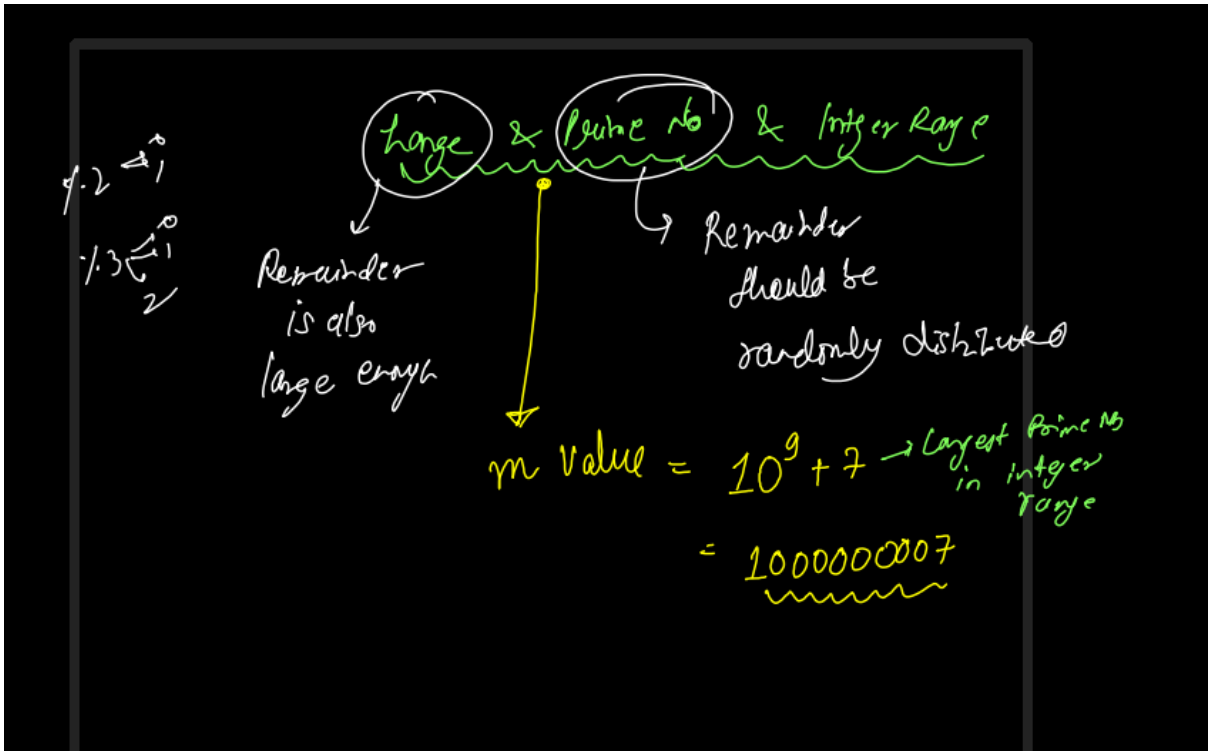
$((39 \% 4) / (17 \% 4)) \% 4 = (3/1) \% 4 = 3$

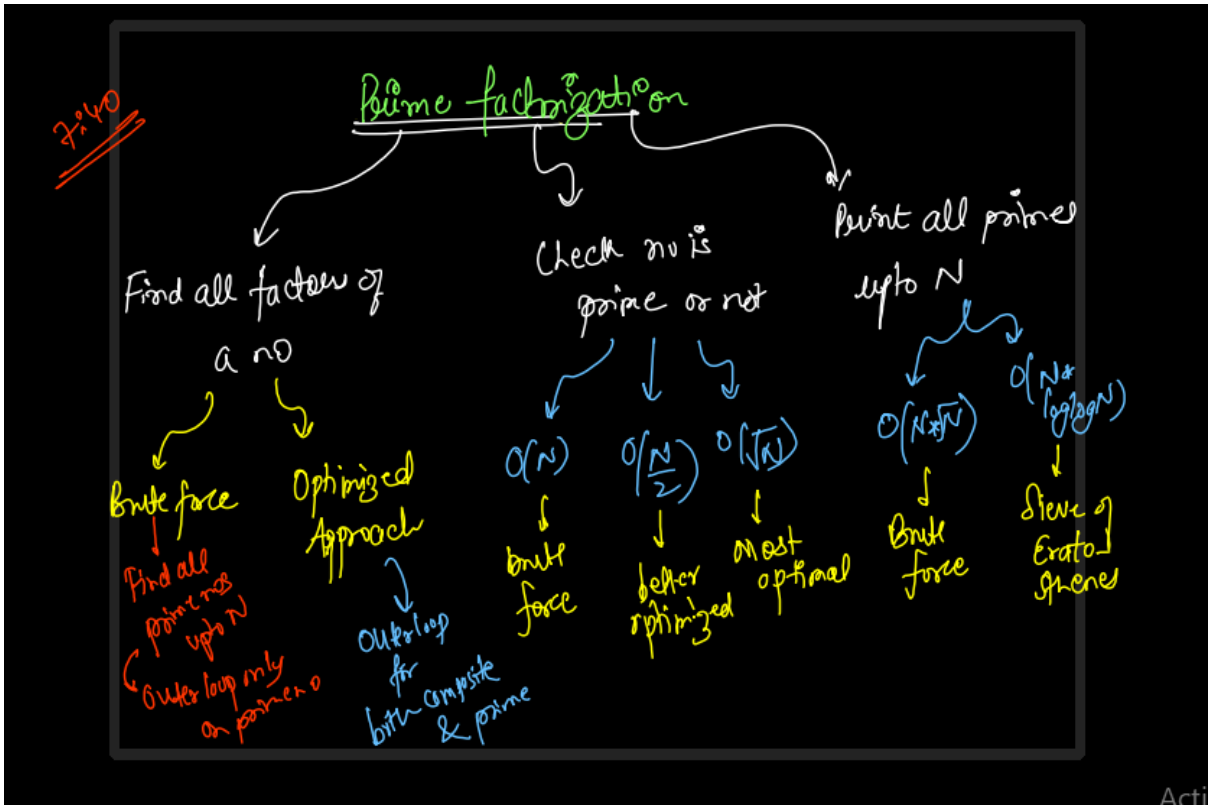
$(a/b) \% m = ((a \% m) * (b^{-1} \% m)) \% m$

b^{-1}: multiplicative modulo inverse of b


They are individually fit in intyes
var1
var2
(2000000000 + 2000000000) % 17
a+b = -ve
(a+b)%m
(a%m + b%m) % m
(4000000000) % 17
(2*10^10)%17 + (2*10^10)%17) % 17
(14 + 14)%17
= 28 %17
= 11
It is not in integer range
= (-ve)%17
= garbage
overflow


(2*10^10 * 2*10^10) %17
= (4*10^20) %17
= (overflow) %17 = garbage
2^31 - 1
≈ 2*10^10
10 digit
((2*10)^10 %17 * (2*10^10)%17)%17
= (14 * 14) %17 = 196%17 = 9


large & prime no & Integer Range
Remainder is also large enough
Remainder should be randomly distributed
m value = $10^9 + 7$ → Largest Prime No in integer range
= 1000000007


7:40
Prime factorization
Find all factors of a no
Check no is prime or not
Print all primes upto N
Brute force
Optimized Approach
Find all prime nos upto N
Outer loop only on primes
Outer loop for both composite & prime
O(N)
O(N/2)
O(√N)
O(N*√N)
O(N* log log N)
Brute force
better optimized
Most optimal
Brute force
Sieve of Eratosthenes

Prime factorization

2	120
2	60
2	30
3	15
5	5
	1

120 = 2 * 2 * 3 * 5

2	1386
3	693
3	231
7	77
11	11
	1

1386 = 2 * 3 * 3 * 7 * 11

~~2, 3, 5, 7, 9, 11, 13, 17~~
~~Store all prime no's in array~~

O(N log N) Avg case
O(N^2) in worst case

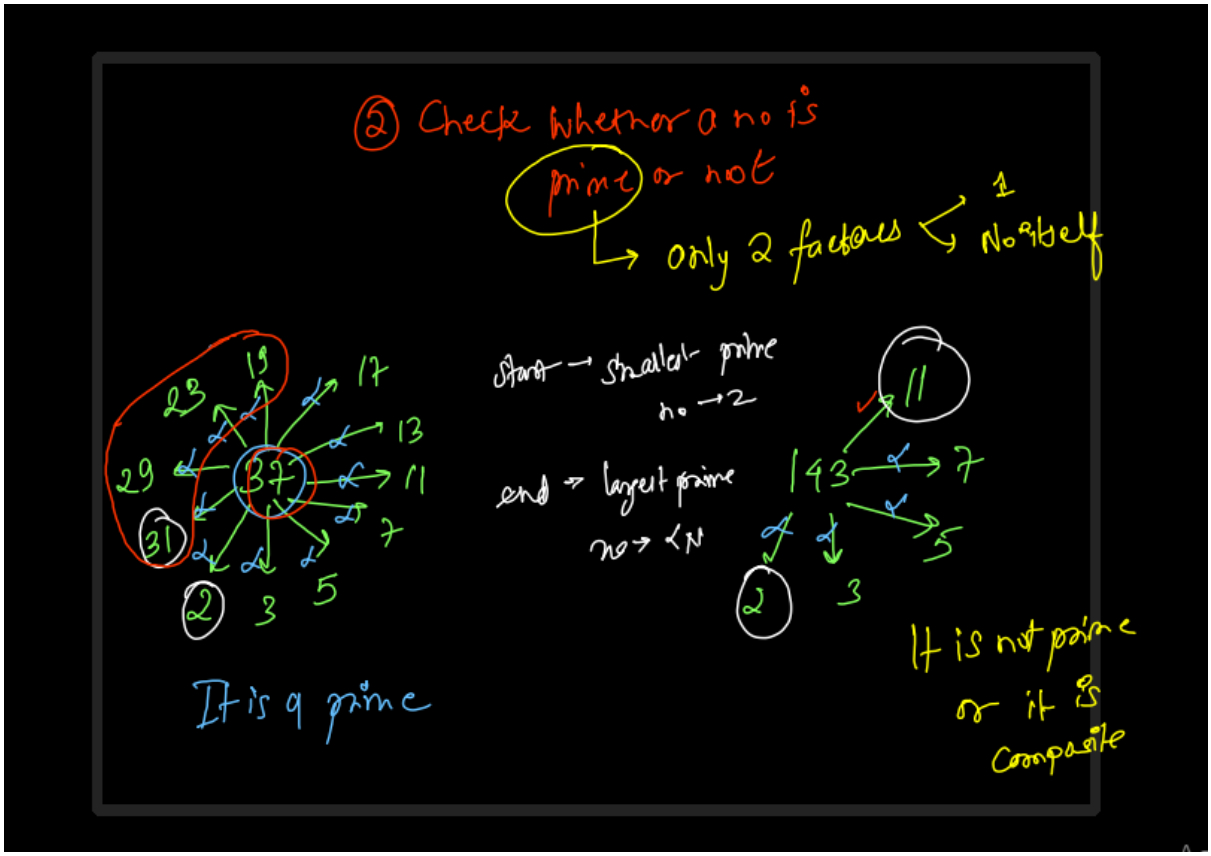
Try to divide N by each prime factor

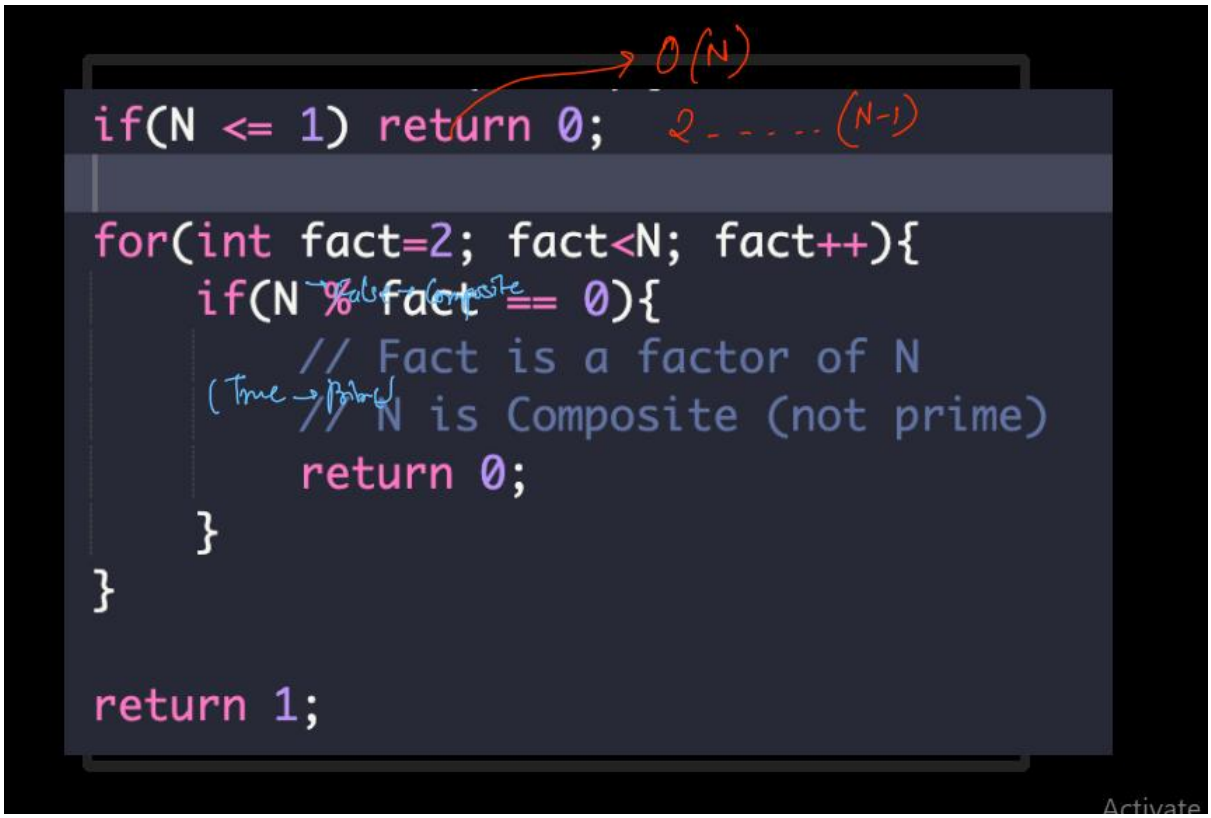
```
// Prime Factorization
int N = scn.nextInt();

for(int fact=2; fact<=N; fact++){
    // fact is a prime factor of N
    while(N % fact == 0){
        N = N / fact;
        System.out.print(fact + " ");
    }
}
```

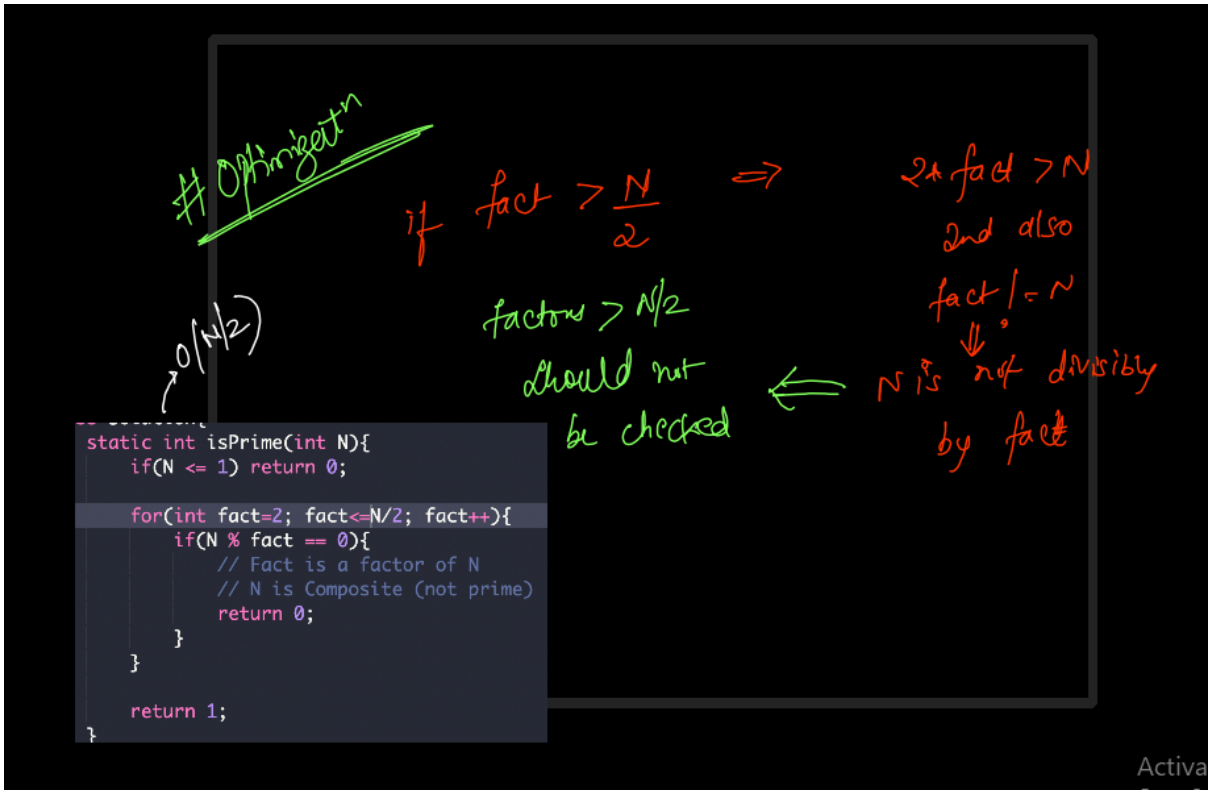
2, 3, 4, 5, 6, 7, 8, 9, 10, 11

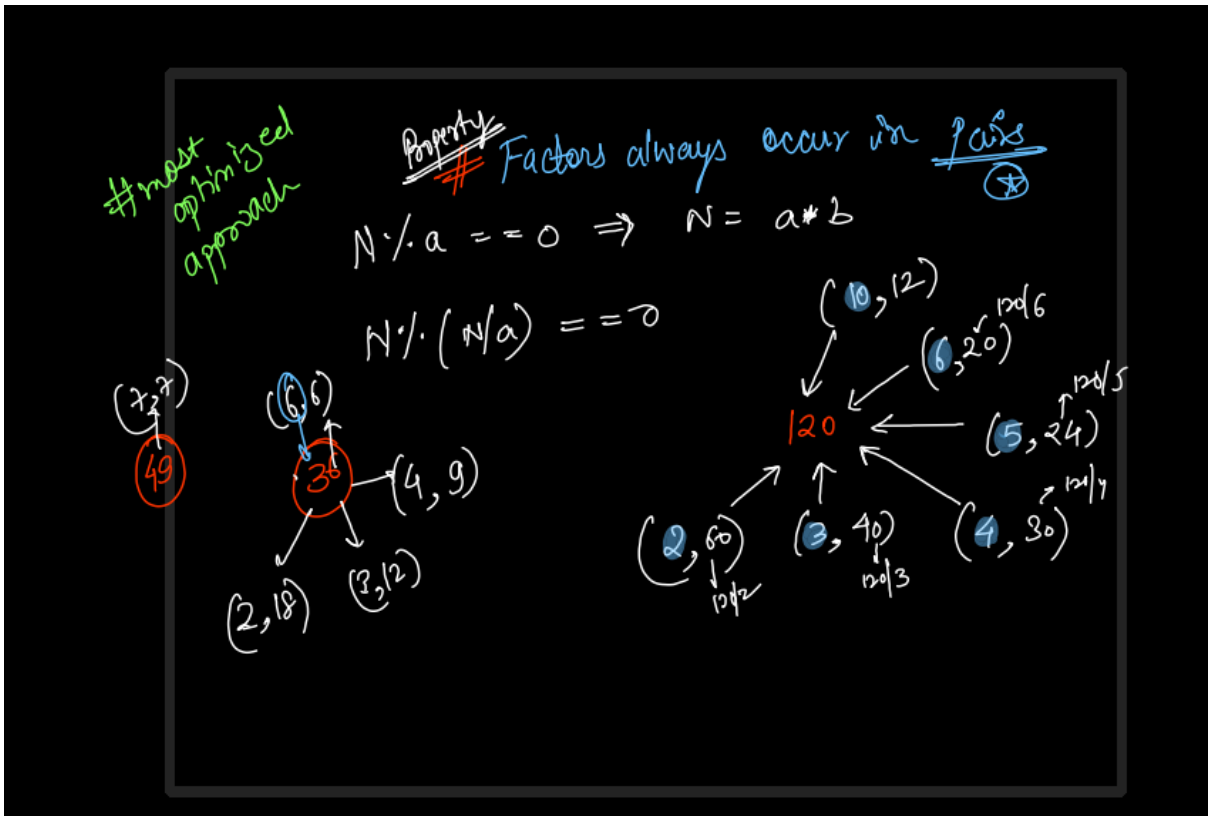
Divide N by fact as long as it is divisible

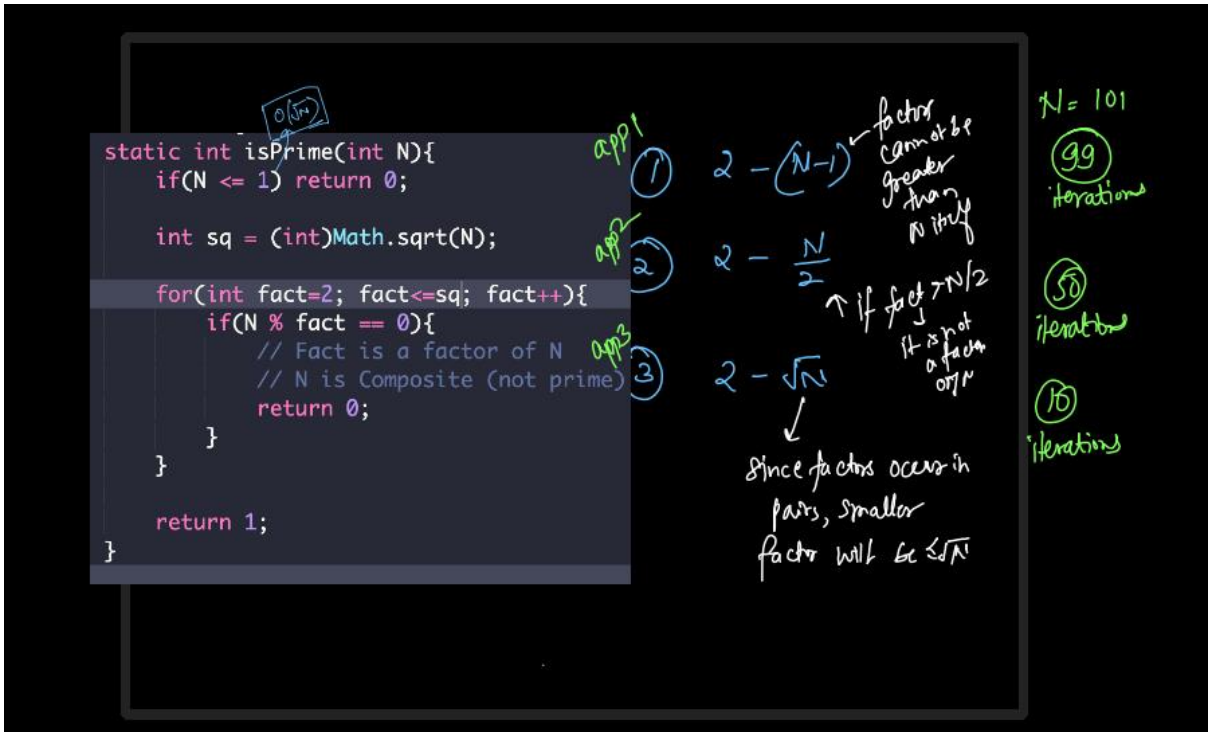
fact = 2	5940 (N)
2	2970
fact = 3	1485
3	495
3	165
fact = 5	55
11	11
	1

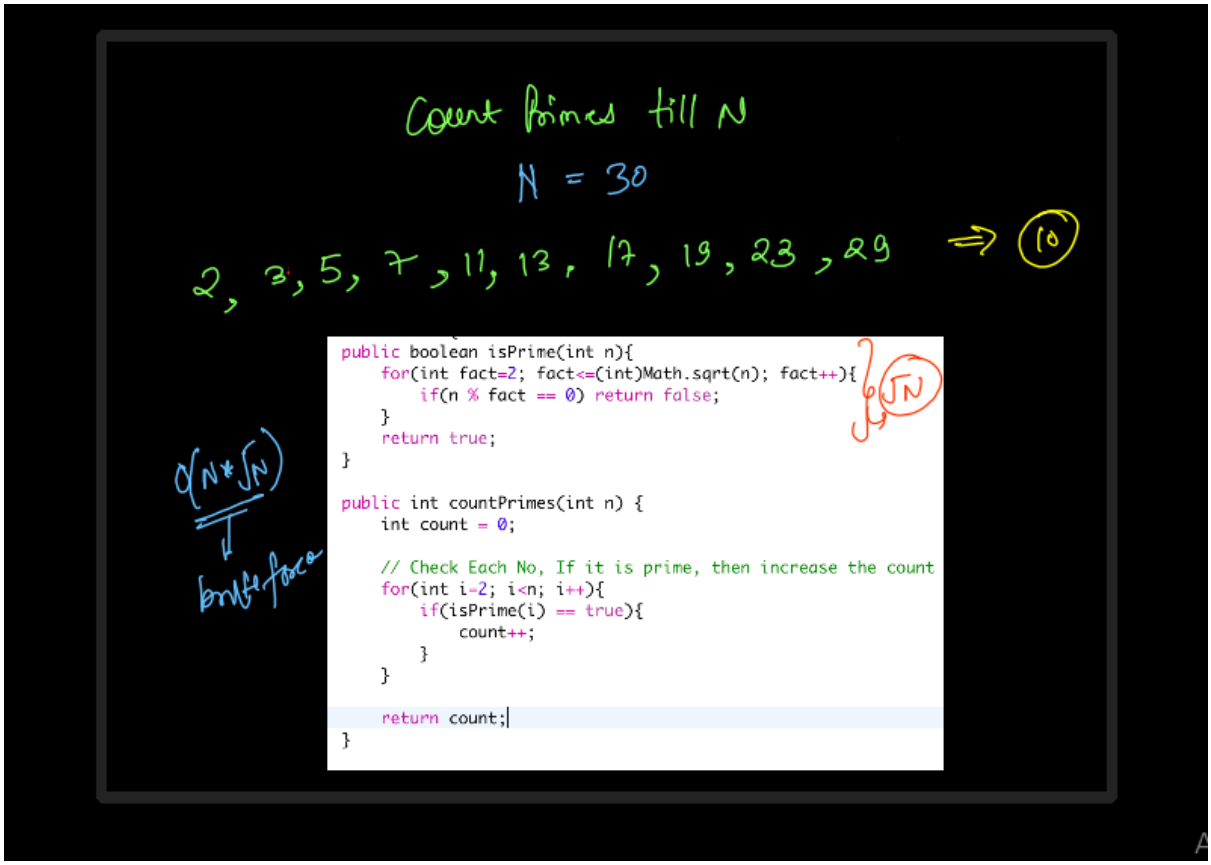

② Check whether a no is
prime or not
only 2 factors 1 & No itself
23
19
17
13
29
37
11
31
7
2
3
5
start → smallest prime no → 2
end → largest prime no → <N
143
11
7
5
2
3
It is a prime
It is not prime or it is Composite

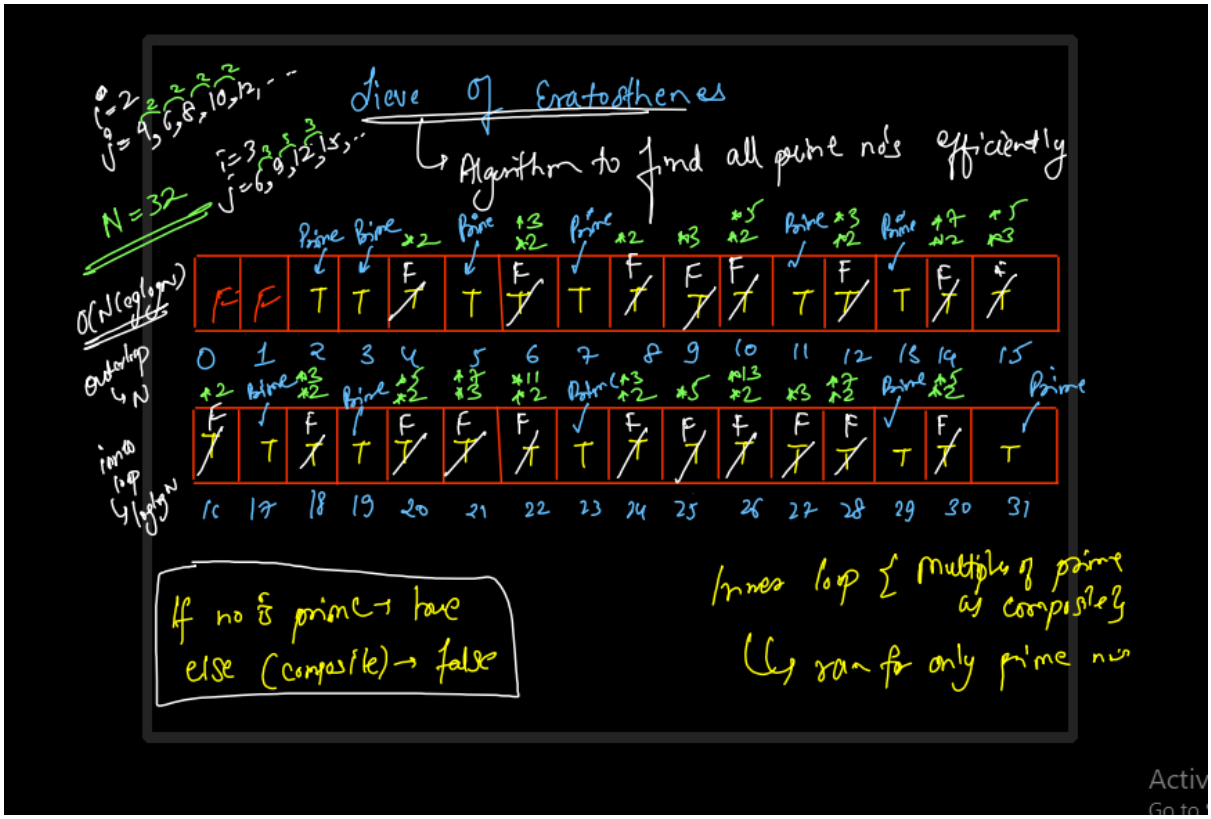

O(N)
2 ..... (N-1)
if(N <= 1) return 0;
for(int fact=2; fact<N; fact++){
if(N % fact == 0){
// Fact is a factor of N
// N is Composite (not prime)
return 0;
}
}
return 1;

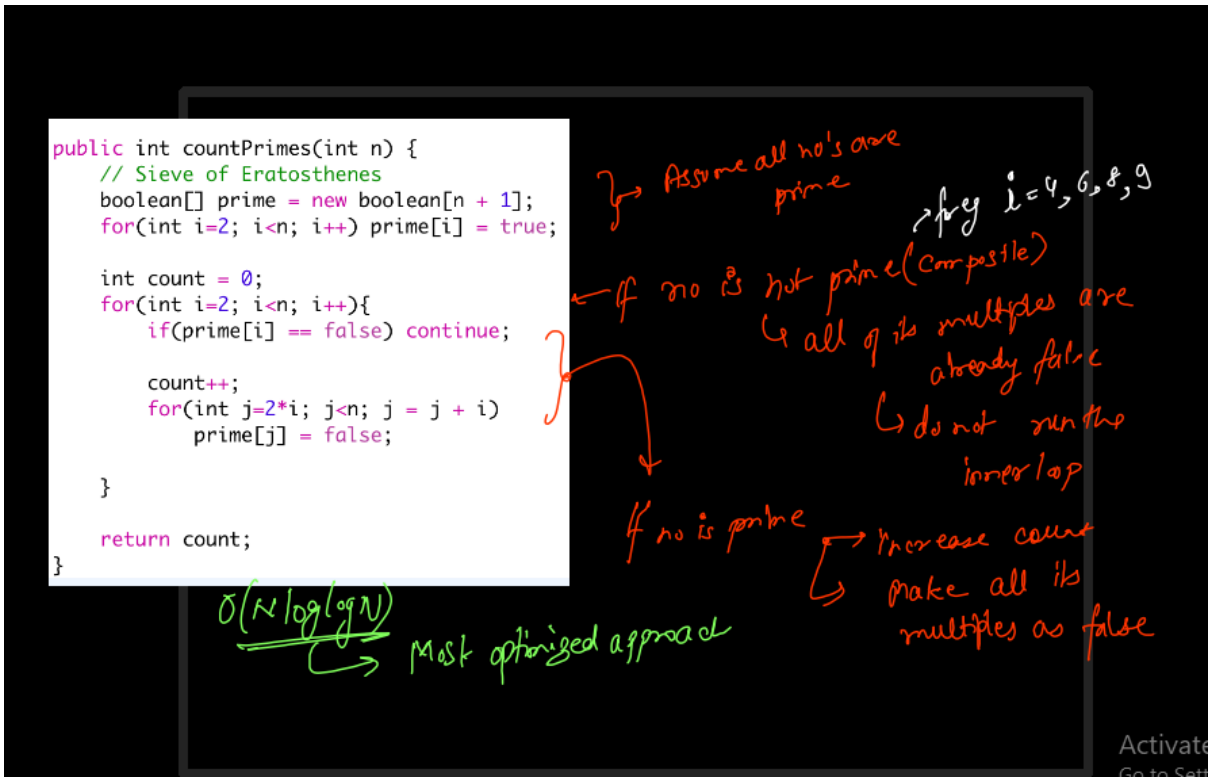

Optimization
if fact > N/2 ⇒ 2*fact > N and also fact != N
factors > N/2 should not be checked ⇐ N is not divisibly by fact
O(N/2)
static int isPrime(int N){
if(N <= 1) return 0;
for(int fact=2; fact<=N/2; fact++){
if(N % fact == 0){
// Fact is a factor of N
// N is Composite (not prime)
return 0;
}
}
return 1;
}


most optimized approach
Property
Factors always occur in pairs (*)
N % a == 0 ⇒ N = a * b
N % (N/a) == 0
(7,7) 49
(6,6) 36
(4,9)
(3,12)
(2,18)
(10,12)
(6,20)
120
(5,24)
(4,30)
(2,60)
(3,40)
120/6
120/5
120/4
120/2
120/3


O(√N)
static int isPrime(int N){
if(N <= 1) return 0;
int sq = (int)Math.sqrt(N);
for(int fact=2; fact<=sq; fact++){
if(N % fact == 0){
// Fact is a factor of N
// N is Composite (not prime)
return 0;
}
}
return 1;
}
app1
app2
app3
1
2
3
2 - (N-1)
factor cannot be greater than N itself
2 - N/2
↑ if fact > N/2 it is not a factor of N
2 - √N
Since factors occur in pairs, smaller factor will be ≤ √N
N = 101
99
iterations
50
iterations
10
iterations


Count Primes till N
N = 30
2, 3, 5, 7, 11, 13, 17, 19, 23, 29 ⇒ 10
public boolean isPrime(int n){
for(int fact=2; fact<=(int)Math.sqrt(n); fact++){
if(n % fact == 0) return false;
}
return true;
}
public int countPrimes(int n) {
int count = 0;
// Check Each No, If it is prime, then increase the count
for(int i=2; i<n; i++){
if(isPrime(i) == true){
count++;
}
}
return count;
}
√N
O(N*√N)
brute force


Sieve of Eratosthenes
Algorithm to find all prime nos efficiently
N = 32
O(N(log(log N)))
outer loop → N
inner loop → log(log N)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Prime
If no is prime → true
else (composite) → false
Inner loop → multiples of prime as composite
ran for only prime nos


public int countPrimes(int n) {
// Sieve of Eratosthenes
boolean[] prime = new boolean[n + 1];
for(int i=2; i<n; i++) prime[i] = true;
int count = 0;
for(int i=2; i<n; i++){
if(prime[i] == false) continue;
count++;
for(int j=2*i; j<n; j = j + i)
prime[j] = false;
}
return count;
}
Assume all no's are prime
for i = 4, 6, 8, 9
If no is not prime (composite)
all of its multiples are already false
do not run the inner loop
If no is prime
increase count
make all its multiples as false
O(N log(log N))
Most optimized approach

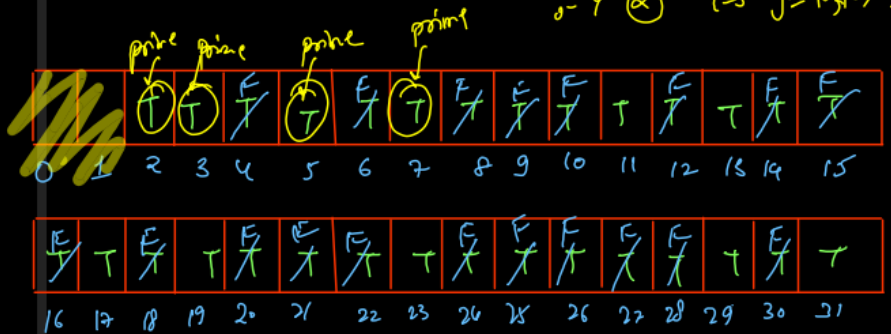
```java
public int countPrimes(int n) {
    // Sieve of Eratosthenes
    boolean[] prime = new boolean[n + 1];
    for(int i=2; i<n; i++) prime[i] = true;

    int count = 0;
    for(int i=2; i<n; i++){
        if(prime[i] == false) continue;

        count++;
        for(int j=2*i; j<n; j = j + i)
            prime[j] = false;

    }

    return count;
}
```

3 composite
1 prime

i=2
j = 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30

i=3
j = 6, 9, 12, 15, 18, 21, 24, 27, 30

i=4 (2) i=5 j = 10, 15, 20, 25, 30



N = 32

GCD or HCF

Greatest Common Divisor

Highest Common Factor

2	60, 48	= 12*5, = 12*4
2	30, 24	
3	15, 12	
	5, 4	

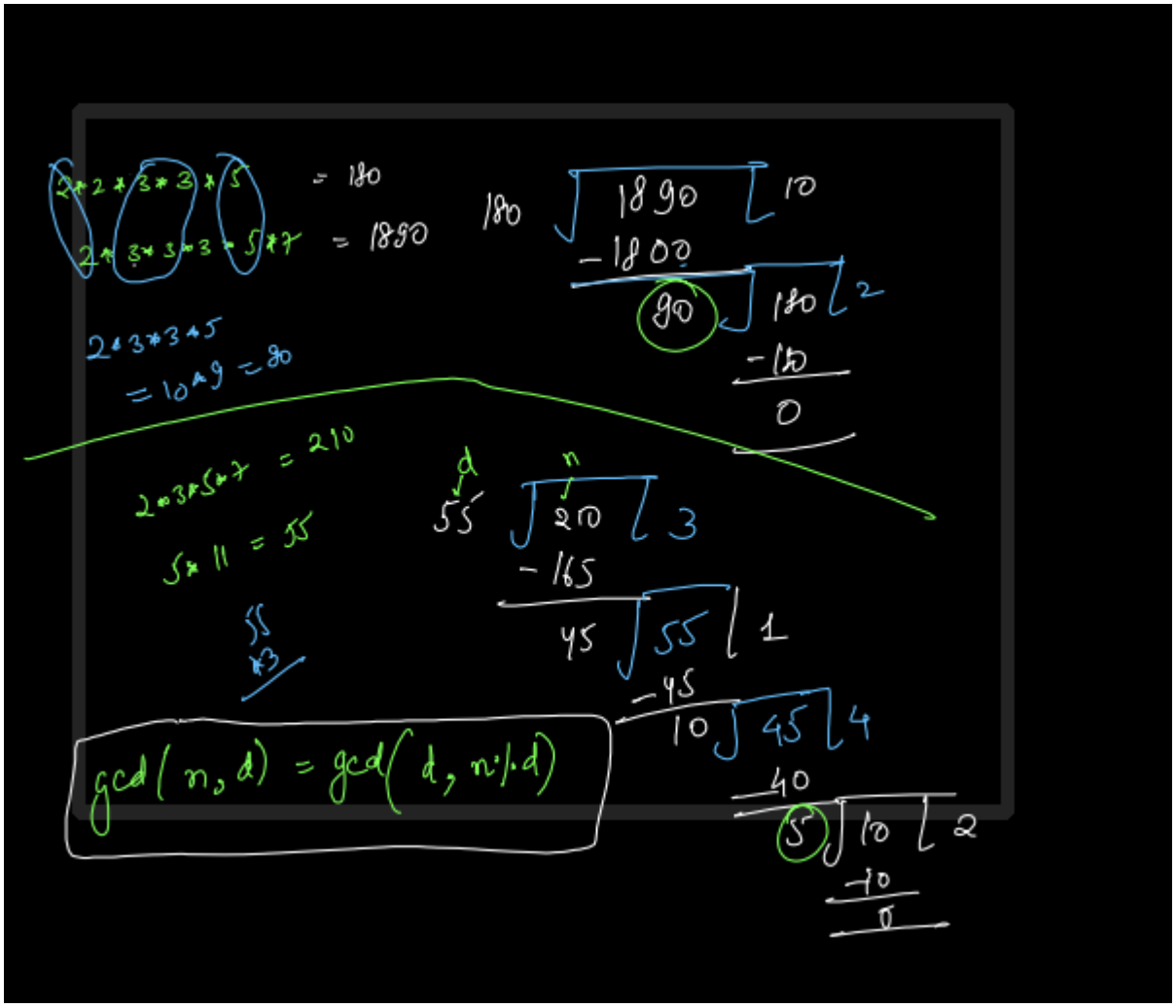
2*2*3 = 12

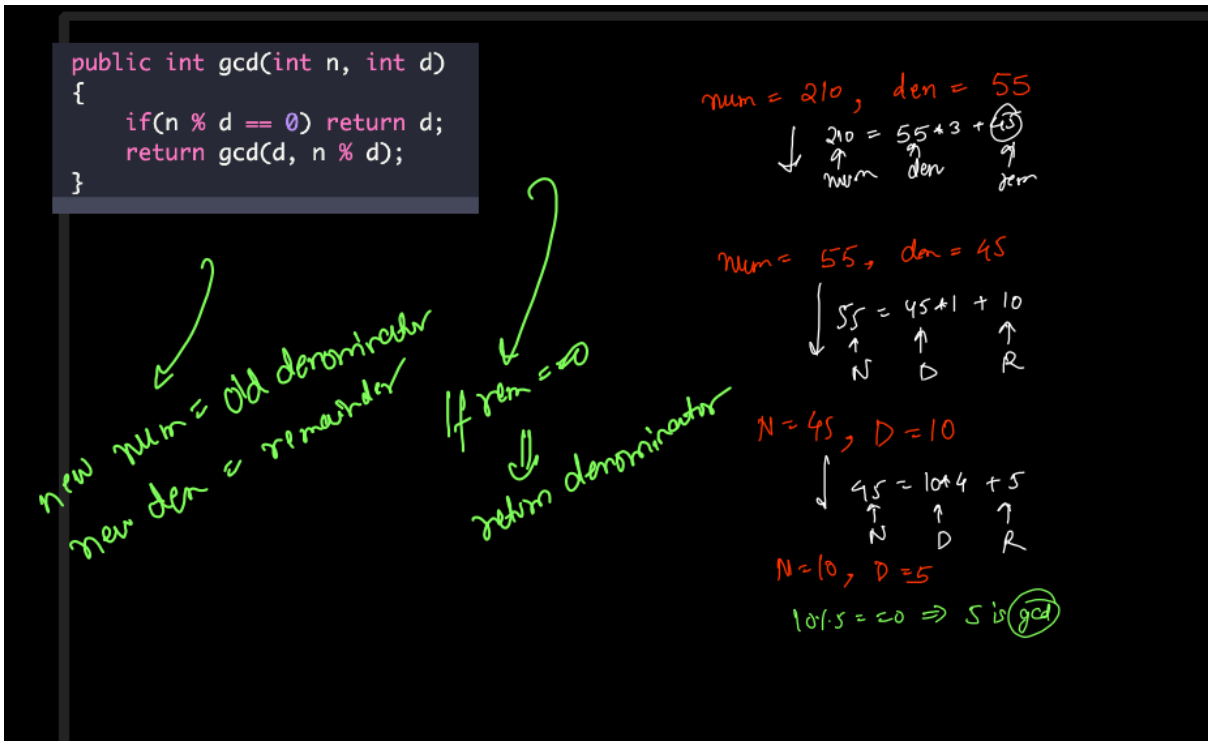
Brute force: Prime factorisation of both nos together

48) 60 (1
- 48
12) 48 (4
- 48
0

ans

Long Division Method


2*2*3*3*5 = 180
2*3*3*3*5*7 = 1890
2*3*3*5 = 10*9 = 90
2*3*5*7 = 210
5*11 = 55
55 *3
gcd(n, d) = gcd(d, n%d)
180 | 1890 | 10
-1800
90 | 180 | 2
-180
0
55 | 210 | 3
-165
45 | 55 | 1
-45
10 | 45 | 4
-40
5 | 10 | 2
-10
0


public int gcd(int n, int d)
{
if(n % d == 0) return d;
return gcd(d, n % d);
}
new num = old denominator
new den = remainder
If rem = 0
return denominator
num = 210, den = 55
210 = 55*3 + 45
num den rem
num = 55, den = 45
55 = 45*1 + 10
N D R
N = 45, D = 10
45 = 10*4 + 5
N D R
N = 10, D = 5
10%5 = 0 ⇒ 5 is gcd

num = 4, den = 16

$4 = 16 * 0 + 4$

(N, D, R)

num = 16, den = 4

$16 \overline{)4} \; 0$

-0

$4 \overline{)16} \; 4$

-16

0

num = 48, den = 60

$60 \overline{)48} \; 0$

-0

$48 \overline{)60} \; 1$

-48

$12 \overline{)48} \; 4$

-48

0

$LCM(48, 60) = \frac{48 * 60}{12} = 240$

(12 → HCF)

$48 = 12 * 4$

$60 = 12 * 5$

(HCF/GCD)

$LCM = 12 * 4 * 5 = 240$

$= \frac{(12*4)*(12*5)}{12} = \frac{48*60}{12}$

$LCM = \frac{a*b}{HCF}$

$a*b = LCM(a,b) * HCF(a,b)$

Binary Exponentiation

$2^p \Rightarrow 2*2*2*\cdots$ p times $\longrightarrow$ O(p) time

$2^0 \Rightarrow 1$

$2^1 \Rightarrow 1*2 \Rightarrow 2$

$2^2 \Rightarrow 1*2*2 = 4$

$2^3 \Rightarrow 1*2*2*2 = 8$

Binary Search Algorithm

Linear search

Unsorted array

{ 10, 7, 5, 9, 4, 8, 2, 6 }

target = 9 ⇒ true

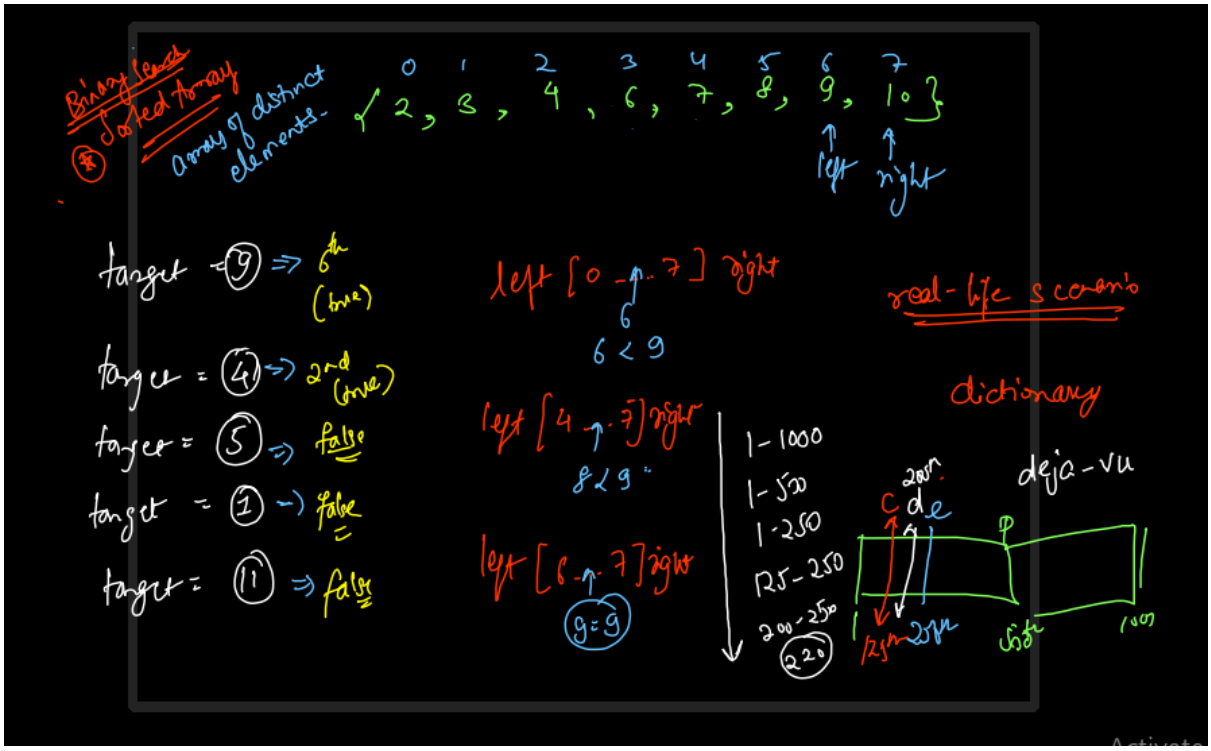
target = 3 ⇒ -1 false

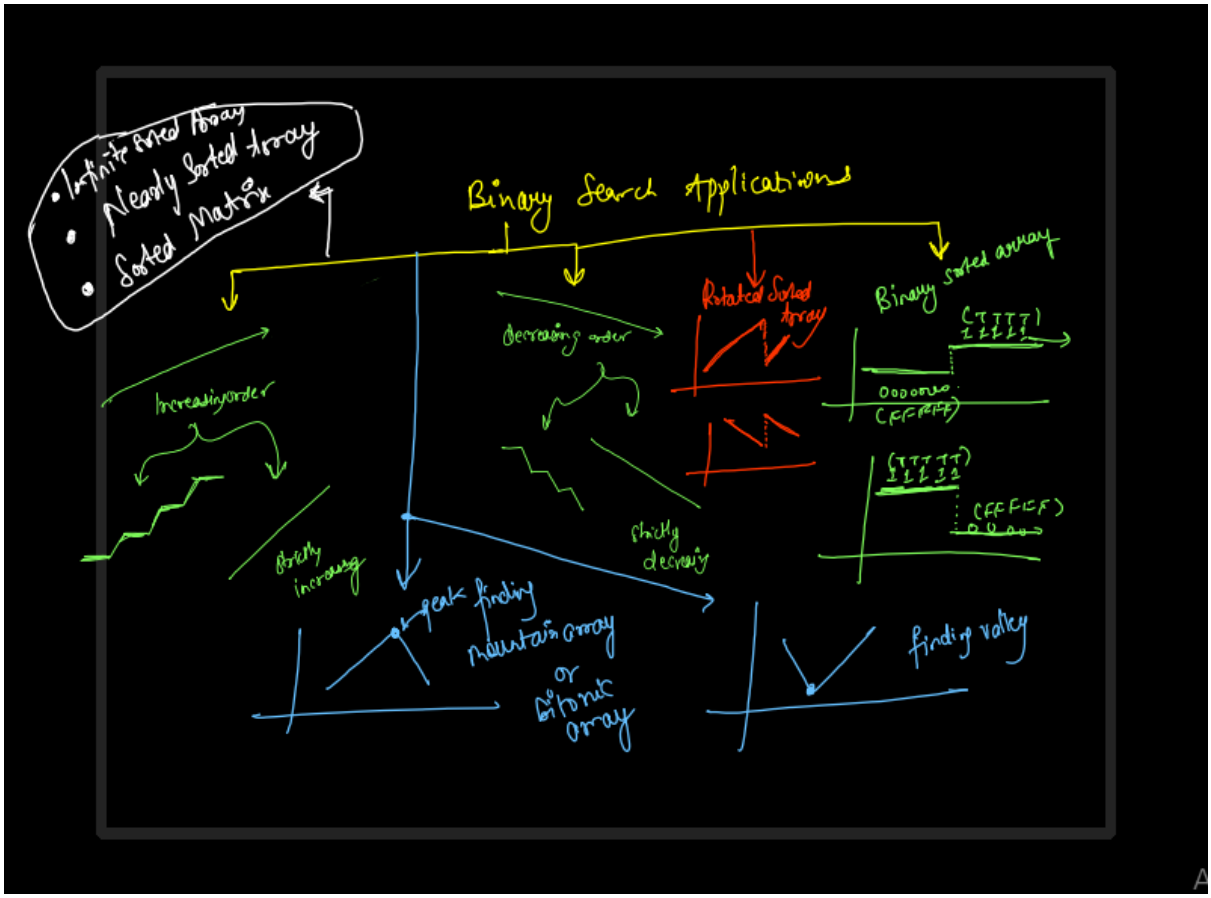
best case $\longrightarrow$ O(1) in the 0th index itself

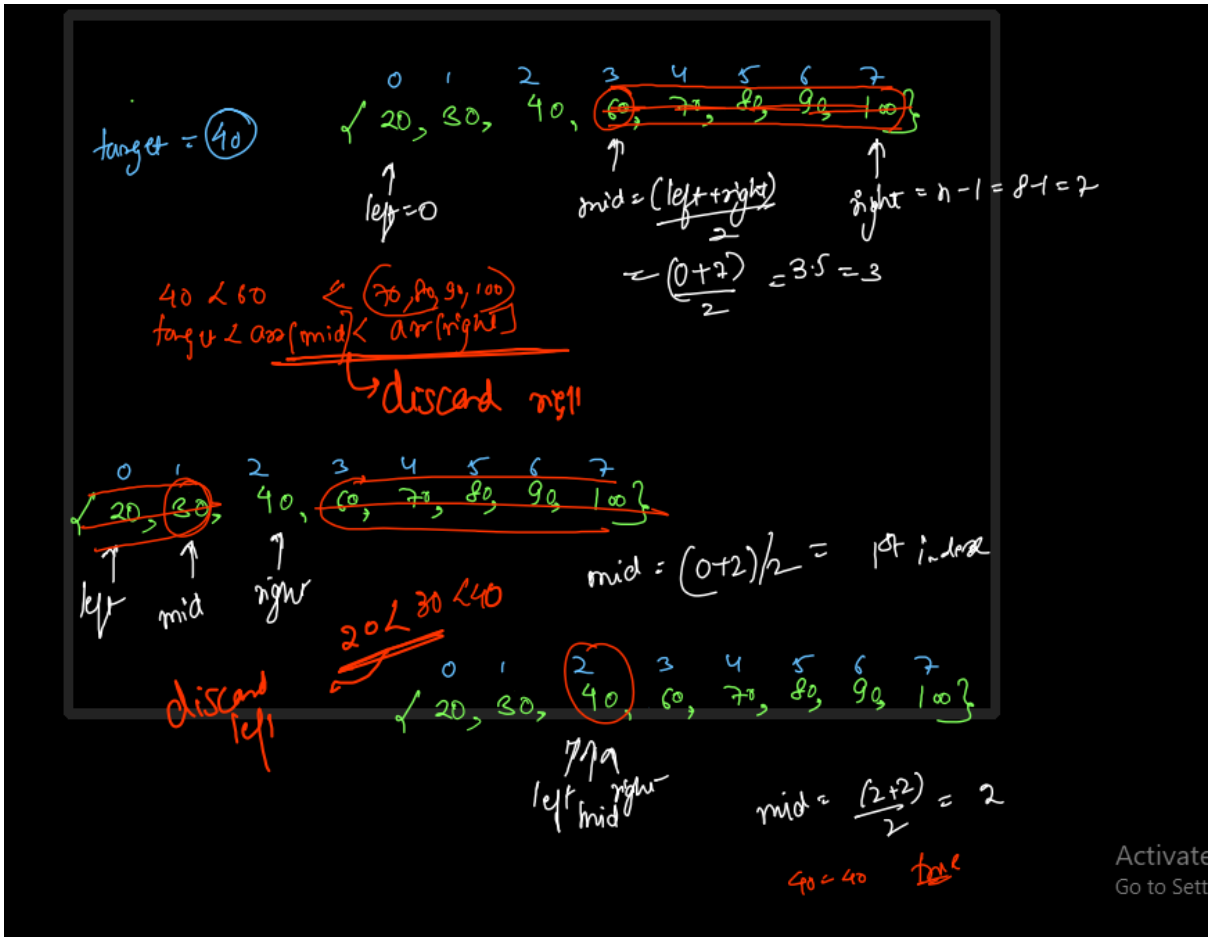
worst case $\longrightarrow$ O(n) unsuccessful search

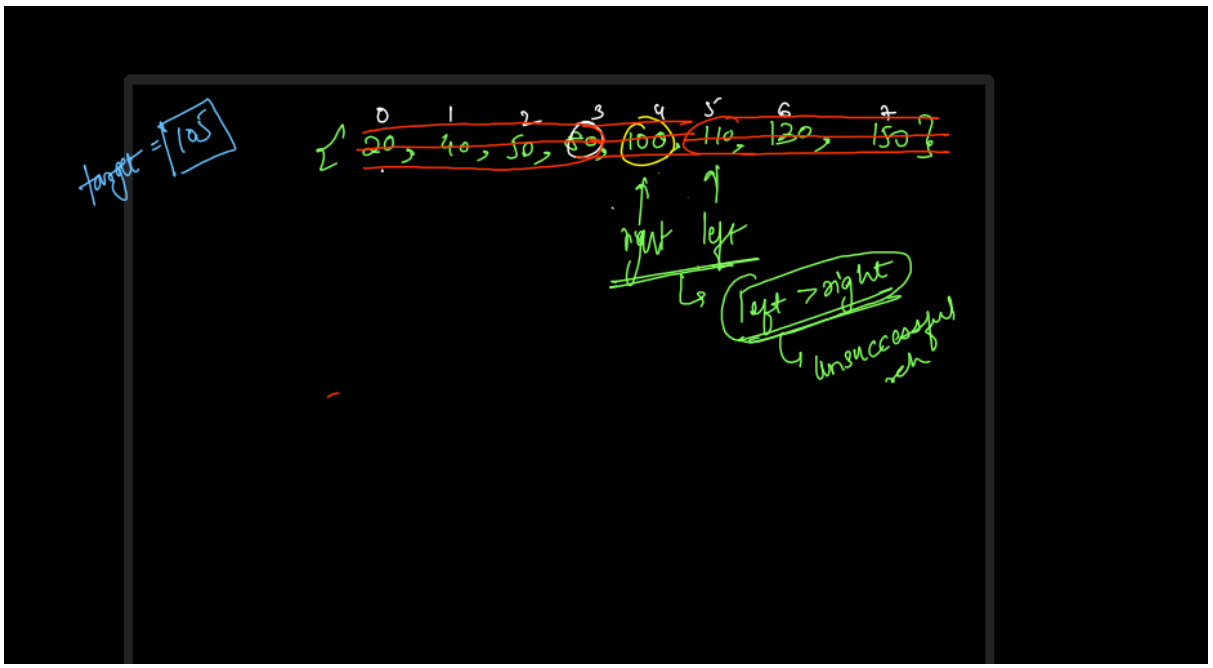
avg case $\longrightarrow$ $O(n) = \frac{n}{2}$

N → N-1 → N-2 → N-3 → N-4 → N-5 → ... → 0


Binary Search
Sorted Array
Array of distinct elements
0 1 2 3 4 5 6 7
2, 3, 4, 6, 7, 8, 9, 10
left
right
target = 9 ⇒ 6th (true)
target = 4 ⇒ 2nd (true)
target = 5 ⇒ false
target = 1 ⇒ false
target = 11 ⇒ false
left [0 → 7] right
6 < 9
left [4 → 7] right
8 < 9
left [6 → 7] right
9 = 9
real-life scenario
dictionary
1-1000
1-500
1-250
125-250
200-250
220
deja-vu
200
c
d
e
p
125
250
500
1000


Infinite sorted Array
Nearly Sorted Array
Sorted Matrix
Binary Search Applications
Increasing order
decreasing order
Rotated Sorted array
Binary sorted array
strictly increasing
strictly decreasing
peak finding
Mountain array
or
Bitonic array
finding valley


target = 40
0 1 2 3 4 5 6 7
20, 30, 40, 60, 70, 80, 90, 100
left=0
mid = (left+right)/2
right = n-1 = 8-1 = 2
= (0+7)/2 = 3.5 = 3
40 < 60
target < arr[mid] < arr[right]
discard right
mid = (0+2)/2 = 1st index
20 < 30 < 40
discard left
mid = (2+2)/2 = 2
40 = 40
Activate
Go to Sett


target = 105
0 1 2 3 4 5 6 7
20, 40, 50, 60, 100, 110, 120, 150
right left
left > right
unsuccessful

Big 2 $N \to N/2 \to N/4 \to N/8 \to N/16 \dots \to 0$

```java
public int search(int[] nums, int target) {
    int left = 0, right = nums.length - 1;

    while(left <= right){
        int mid = (left + right) / 2;

        if(nums[mid] == target) return mid;
        else if(nums[mid] < target)
            left = mid + 1; // Discard the Left Range
        else right = mid - 1; // Discard the Right Range
    }

    return -1;
}
```

Best case $\to O(1)$

Worst case $\to O(\log_2 n)$

Avg case $\to O(\log_2 n)$

Time Complexity $\to O(\log_2 n)$

```java
int mid = left + (right - left) / 2;
```

$l + \left(\frac{r-l}{2}\right) \quad l + \frac{r}{2} - \frac{l}{2} = \frac{l}{2} + \frac{r}{2}$

$= \frac{1}{2}(l+r)$

Integer Overflow

$l = 2*10^{10}, \; r = 2*10^{10}$

$\frac{(l+r)}{2} = \frac{(2*10^{10} + 2*10^{10})}{2}$

$= (4*10^{10})/2$ overflow

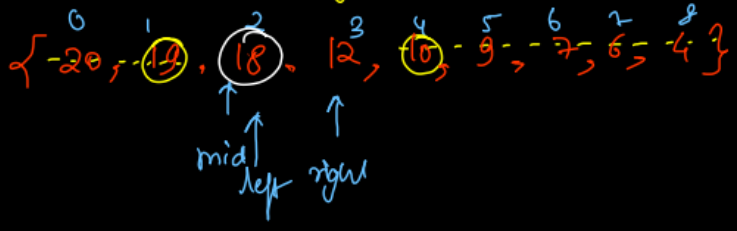
$l + (r-l)/2$

$= 2*10^{10} + \frac{(2*10^{10} - 2*10^{10})}{2} = 2*10^{10}$ ✓

Time Complexity Chart

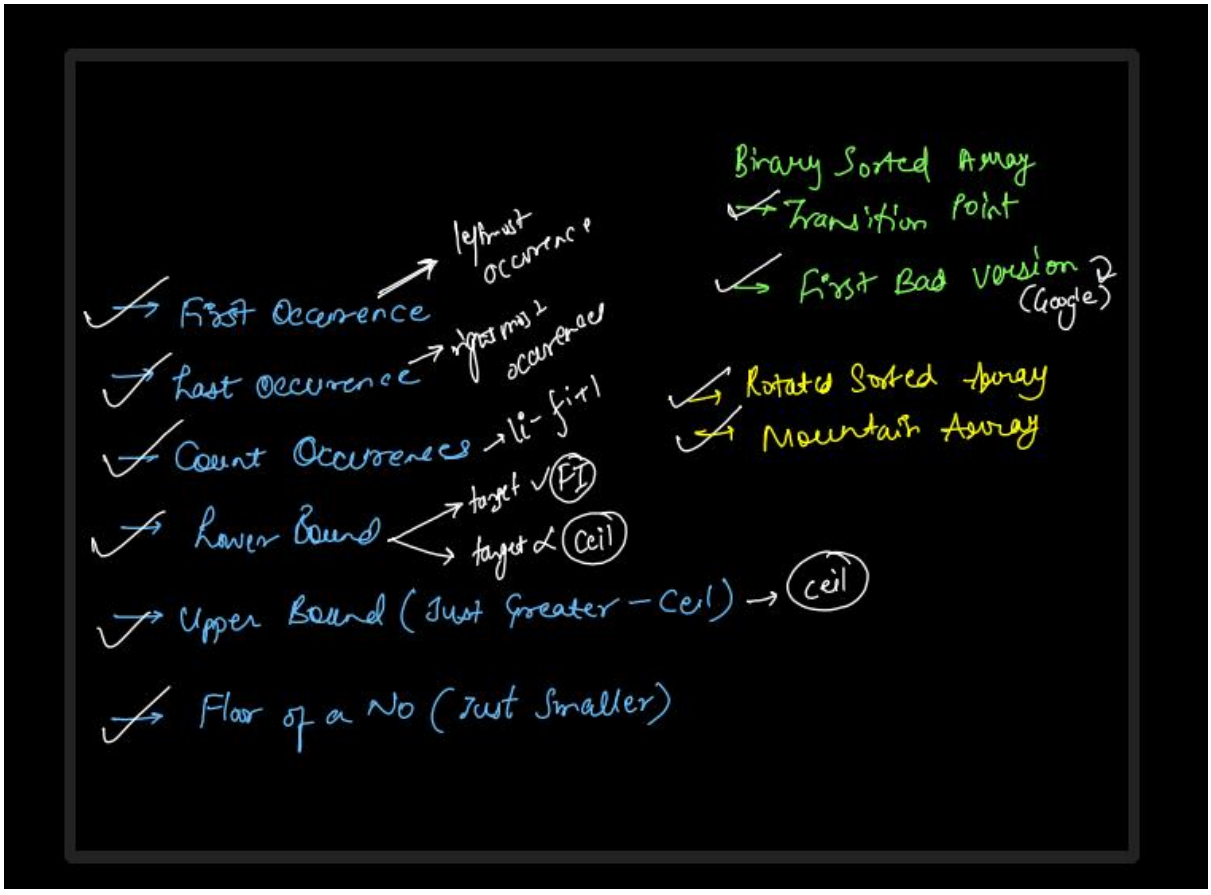
	Linear Search ⇒ O(n)	Binary Search ⇒ O(log₂n) Divide & Conquer
$N = 2^0$	1	1
$N = 2^1$	2	2
$N = 2^2$	4	2
$N = 2^3$	8	3
$N = 2^4$	16	4
$N = 2^5$	32	5
$N = 2^6$	64	6

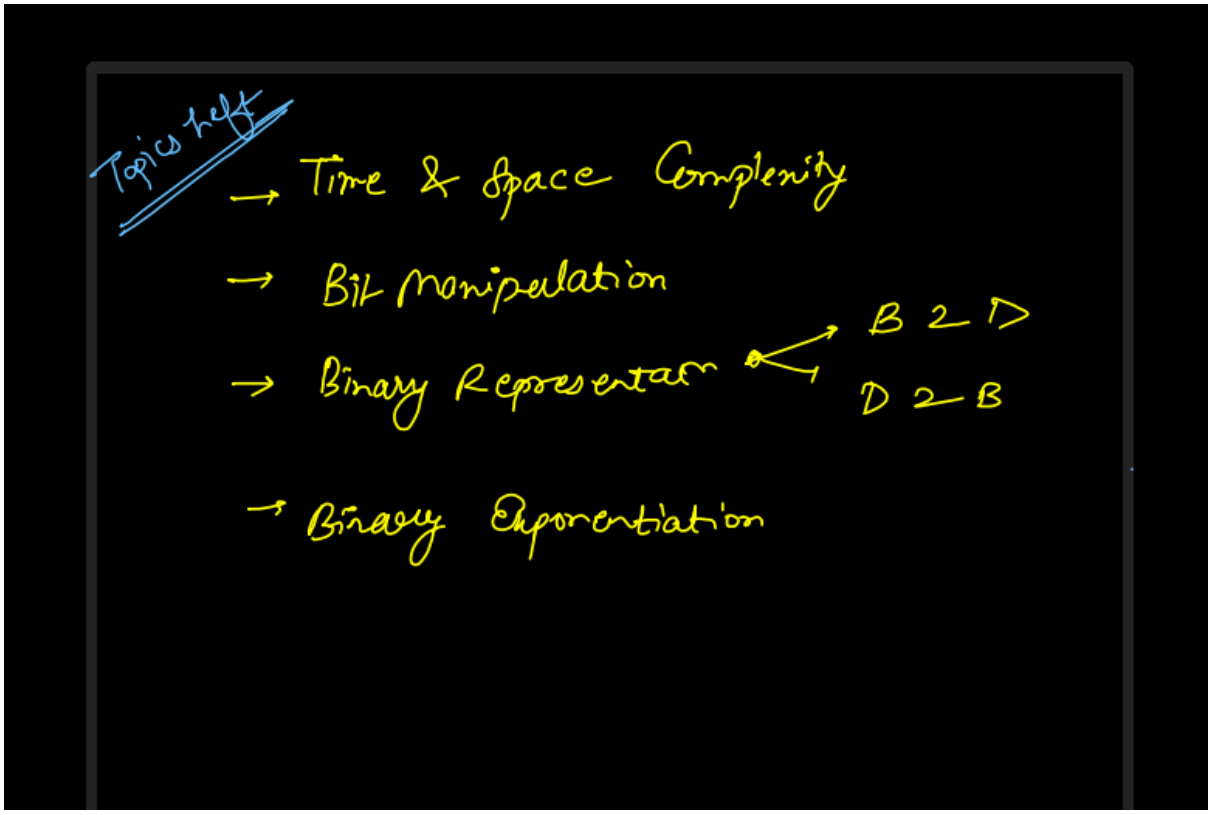
Binary Search on Reverse Sorted Array

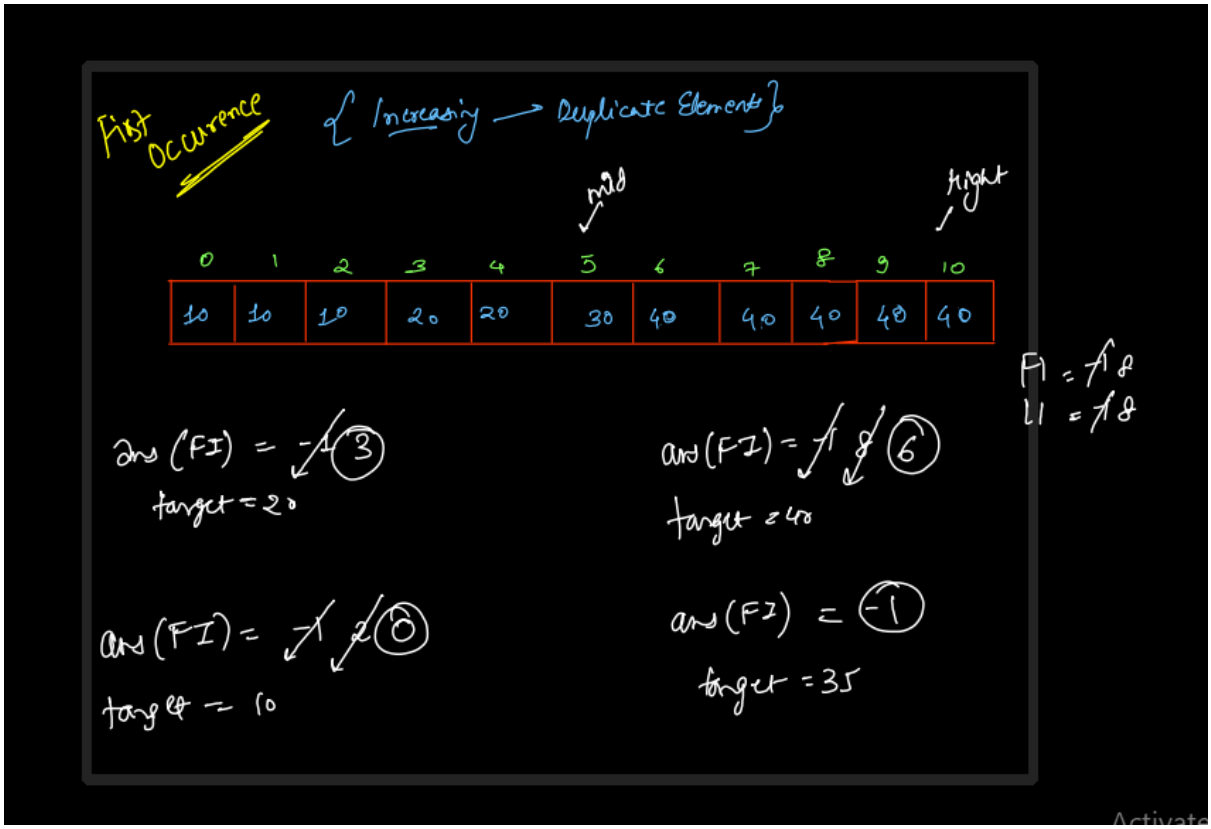


target = 18

```java
public int searchInDecreasing(int[] nums, int target) {
    int left = 0, right = nums.length - 1;

    while(left <= right){
        int mid = left + (right - left) / 2;

        if(nums[mid] == target) return mid;
        else if(nums[mid] > target)
            left = mid + 1; // Discard the Left Range
        else right = mid - 1; // Discard the Right Range
    }

    return -1;
}
```

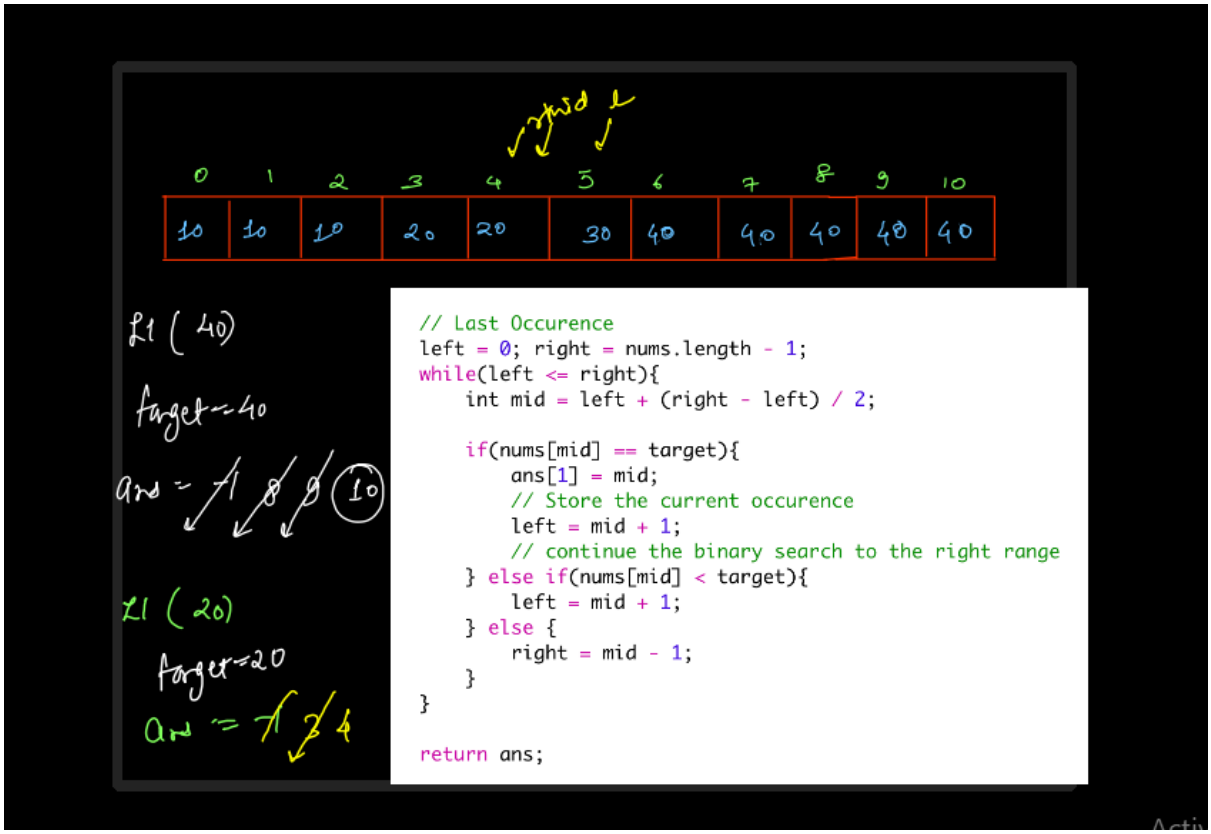

First Occurrence → leftmost occurrence
Last occurrence → rightmost occurrence
Count Occurrences → li - fi + 1
Lower Bound → target ✓ (FI)
target ∠ (ceil)
Upper Bound (Just Greater - Ceil) → (ceil)
Floor of a No (Just Smaller)
Binary Sorted Array
Transition Point
First Bad Version (Google)
Rotated Sorted Array
Mountain Array


Topics left
→ Time & Space Complexity
→ Bit Manipulation
→ Binary Representation
B 2 D
D 2 B
→ Binary Exponentiation

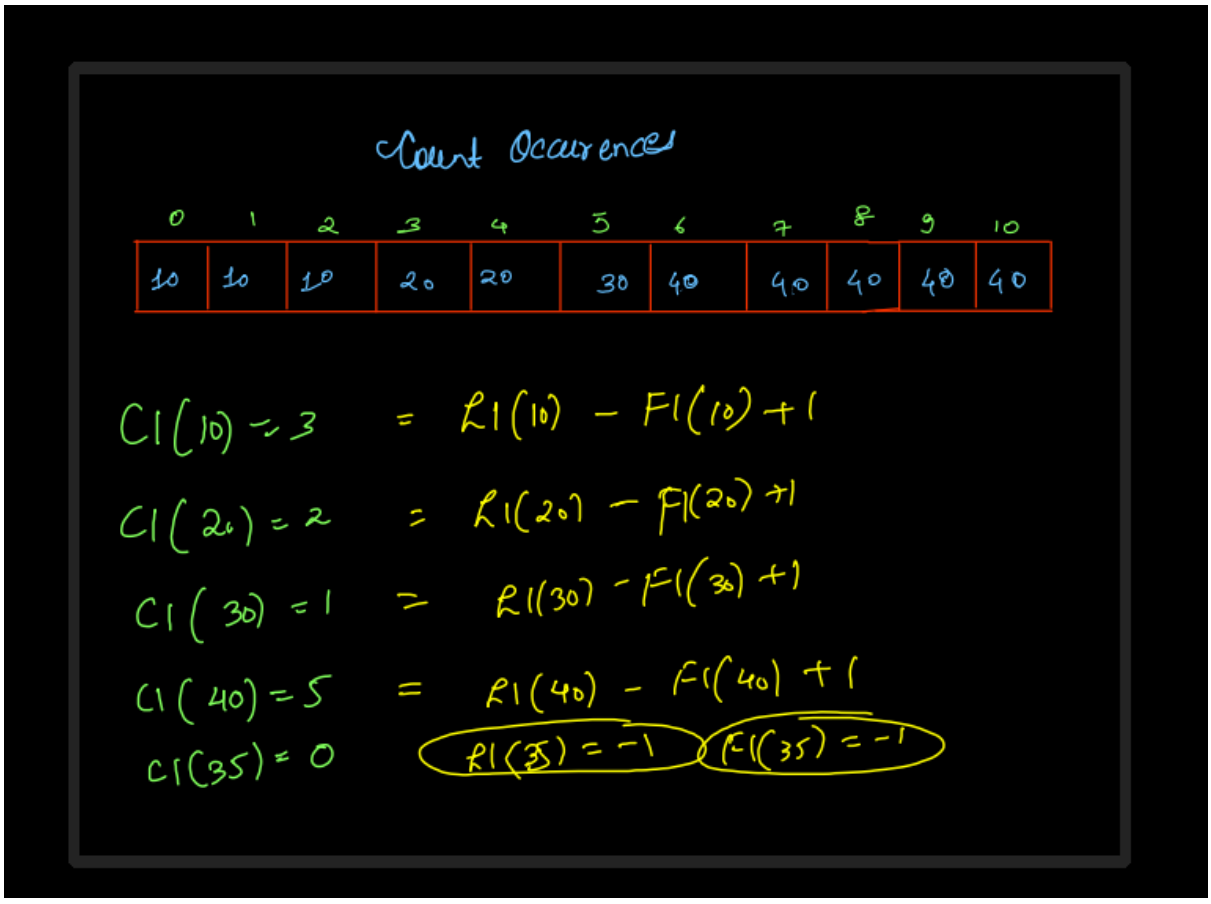



```java
int[] ans = new int[2];
ans[0] = ans[1] = -1;

// First Occurence
int left = 0, right = nums.length - 1;
while(left <= right){
    int mid = left + (right - left) / 2;

    if(nums[mid] == target){
        ans[0] = mid;
        // Store the current occurence
        right = mid - 1;
        // continue the binary search to the left range
    } else if(nums[mid] < target){
        left = mid + 1;
    } else {
        right = mid - 1;
    }
}
```

```java
// Last Occurence
left = 0; right = nums.length - 1;
while(left <= right){
    int mid = left + (right - left) / 2;

    if(nums[mid] == target){
        ans[1] = mid;
        // Store the current occurence
        left = mid + 1;
        // continue the binary search to the right range
    } else if(nums[mid] < target){
        left = mid + 1;
    } else {
        right = mid - 1;
    }
}

return ans;
```




```
// Last Occurence
left = 0; right = nums.length - 1;
while(left <= right){
    int mid = left + (right - left) / 2;

    if(nums[mid] == target){
        ans[1] = mid;
        // Store the current occurence
        left = mid + 1;
        // continue the binary search to the right range
    } else if(nums[mid] < target){
        left = mid + 1;
    } else {
        right = mid - 1;
    }
}

return ans;
```



Lower Bound & Upper Bound

0	1	2	3	4	5	6	7	8	9	10
10	10	10	20	20	30	40	40	40	40	40

Lower Bound → Search Insert Position

- If the target is found, then first occurrence is the lower bound
- If target is not found, just greater value or ceil value's first occurrence

LB(10) = 0 , LB(20) = 3 ; LB(30) = 5 ; LB(40) = 6

LB(5) = 0 , LB(18) = 3 , LB(35) = 6, LB(45) = 11 (arr.length)

0	1	2	3	4	5	6	7	8	9	10
10	10	10	20	20	30	40	40	40	40	40

```
class Solution {
public int searchInsert(int[] nums, int target) {
    int ans = nums.length;
    int left = 0, right = nums.length - 1;
    while(left <= right){
        int mid = left + (right - left) / 2;

        if(nums[mid] >= target){
            ans = mid;
            // Store the current occurence
            right = mid - 1;
            // continue the binary search to the left range
        } else if(nums[mid] < target){
            left = mid + 1;
        }
    }

    return ans;
}
```

LB(35)

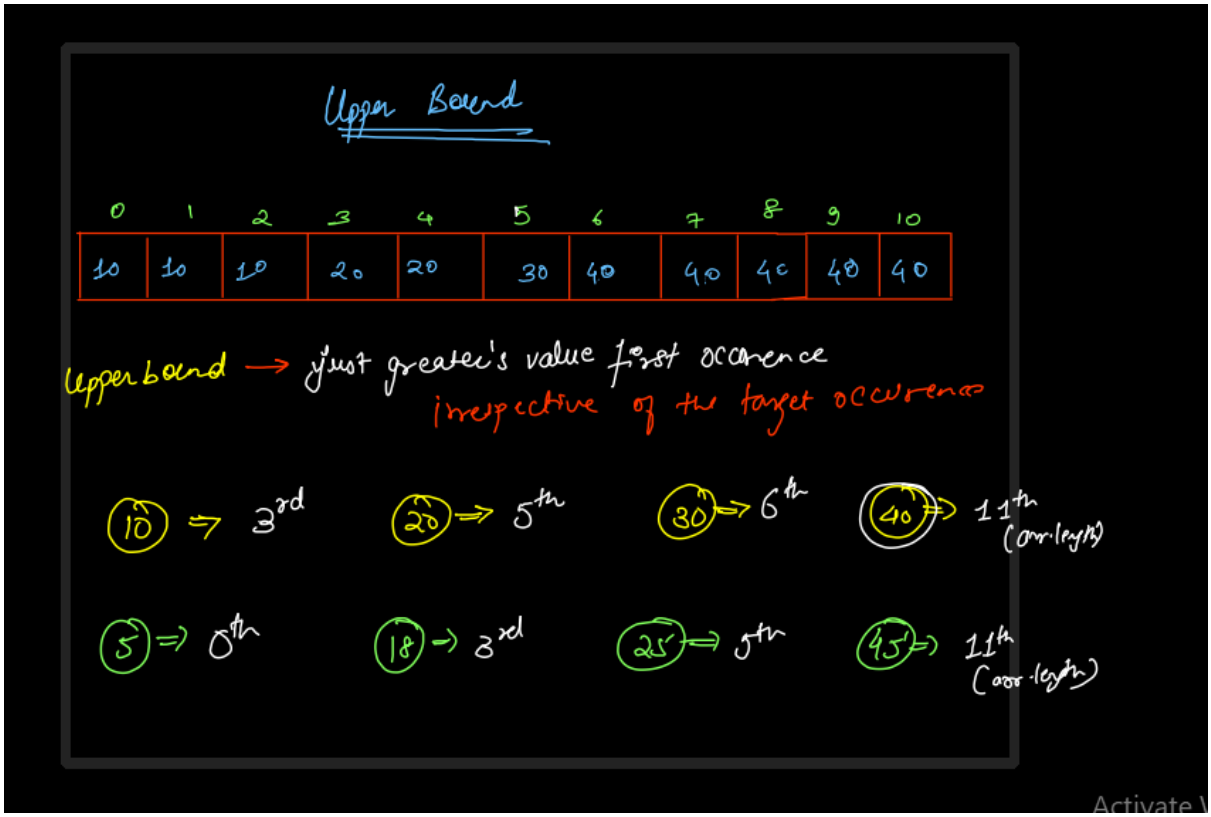
ans = ~~11~~ ~~8~~ 6

LB(5)

ans = ~~11~~ ~~5~~ ~~2~~ 0

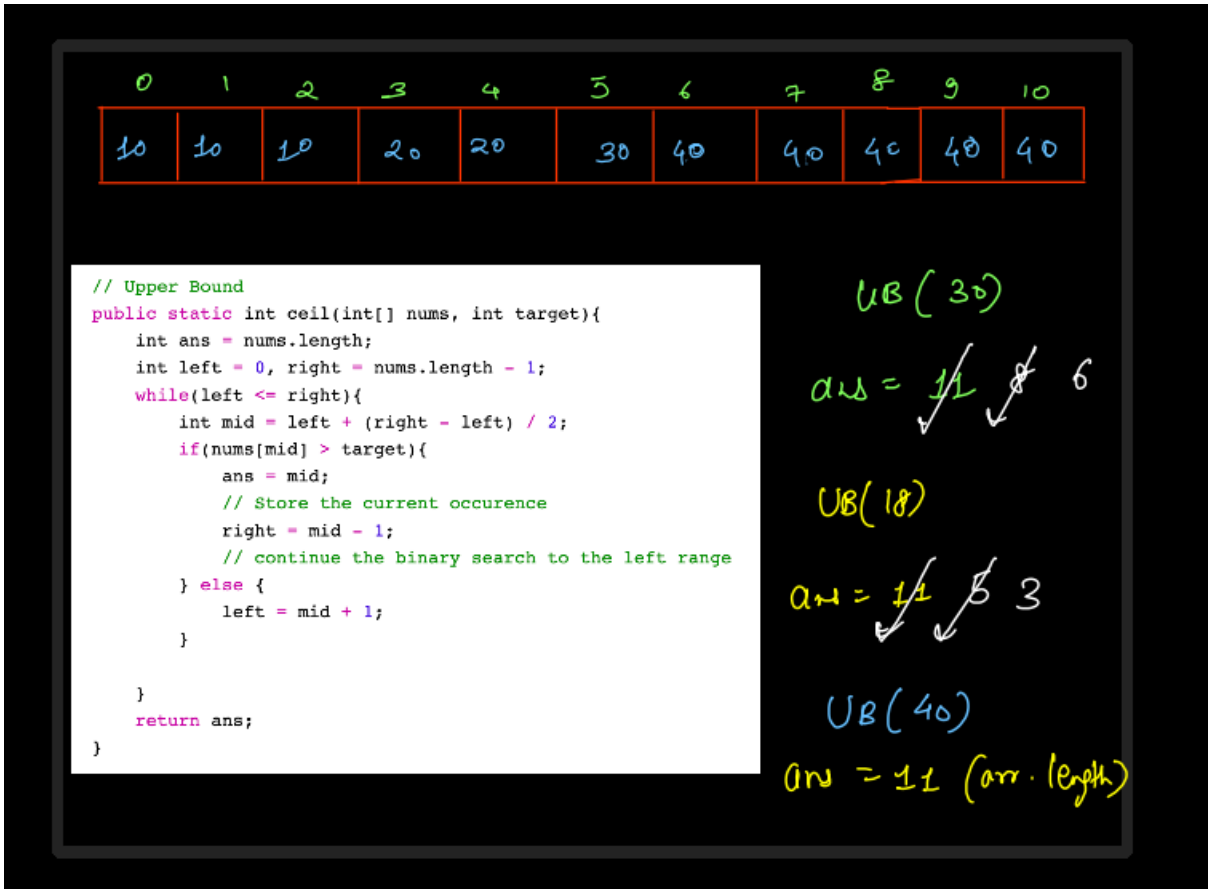
LB(45)

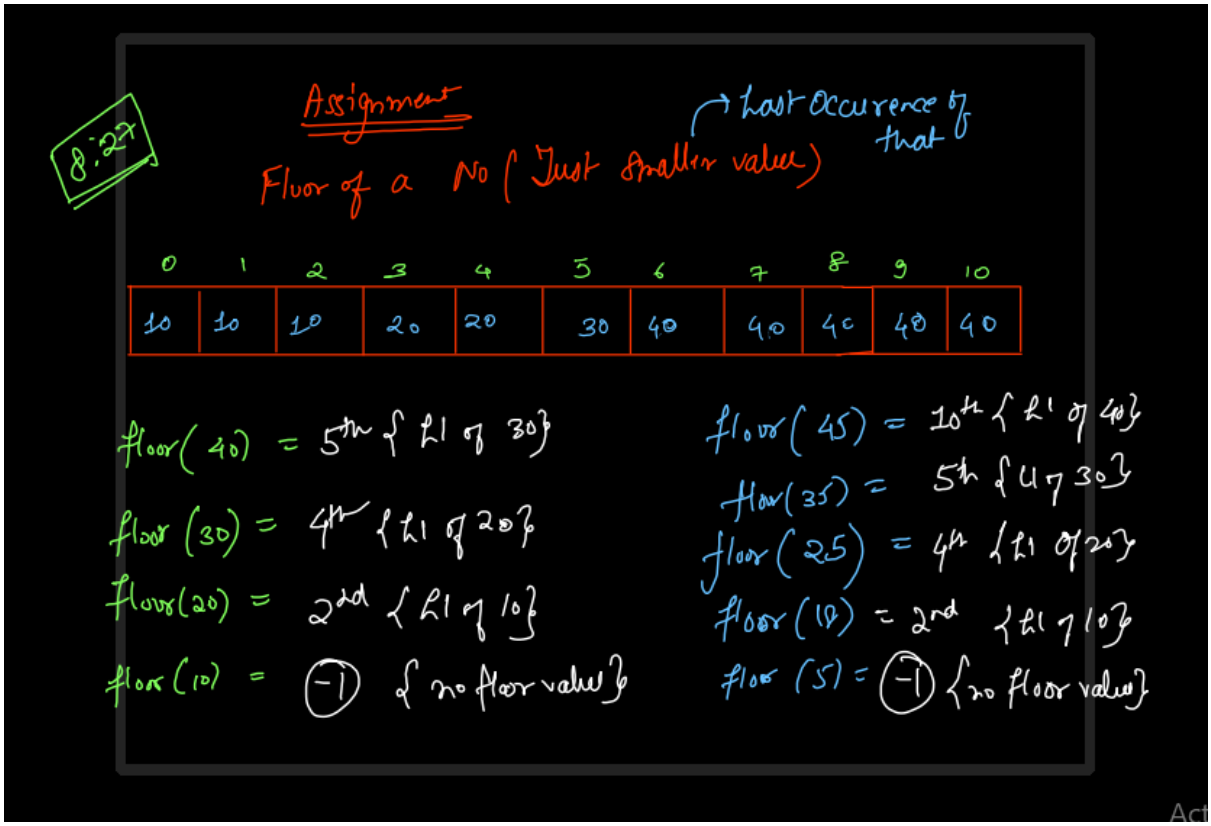
ans = 11




```java
// Lower Bound
public int searchInsert(int[] nums, int target) {
    int ans = nums.length;
    int left = 0, right = nums.length - 1;
    while(left <= right){
        int mid = left + (right - left) / 2;

        if(nums[mid] >= target){
            ans = mid;
            // Store the current occurence
            right = mid - 1;
            // continue the binary search to the left range
        } else {
            left = mid + 1;
        }
    }

    return ans;
}
```

```java
// Upper Bound
public static int ceil(int[] nums, int target){
    int ans = nums.length;
    int left = 0, right = nums.length - 1;
    while(left <= right){
        int mid = left + (right - left) / 2;
        if(nums[mid] > target){
            ans = mid;
            // Store the current occurence
            right = mid - 1;
            // continue the binary search to the left range
        } else {
            left = mid + 1;
        }

    }
    return ans;
}
```

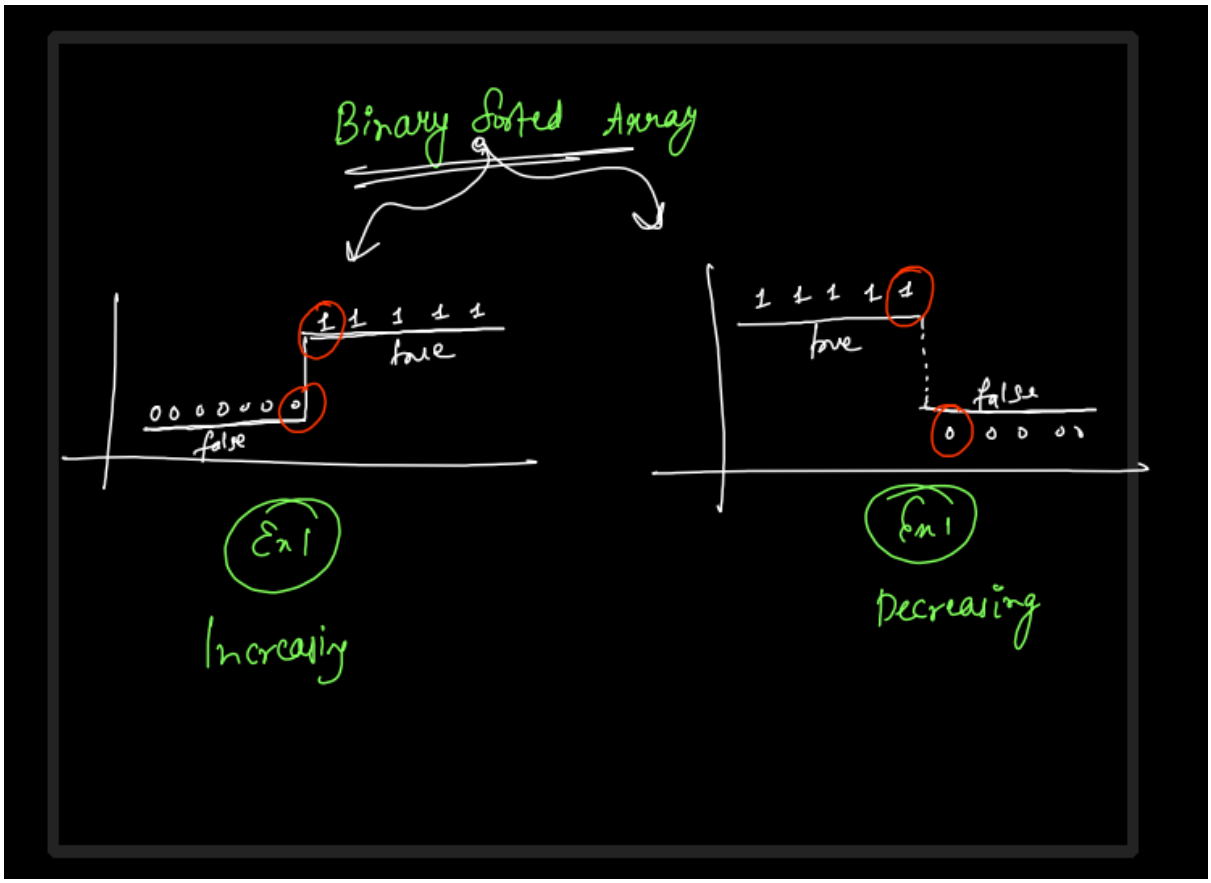

0 1 2 3 4 5 6 7 8 9 10
10 10 10 20 20 30 40 40 40 40 40
// Upper Bound
public static int ceil(int[] nums, int target){
int ans = nums.length;
int left = 0, right = nums.length - 1;
while(left <= right){
int mid = left + (right - left) / 2;
if(nums[mid] > target){
ans = mid;
// Store the current occurence
right = mid - 1;
// continue the binary search to the left range
} else {
left = mid + 1;
}
}
return ans;
}
UB (30)
ans = 11 6
UB(18)
ans = 11 6 3
UB(40)
ans = 11 (arr. length)


8:27
Assignment
Floor of a No (Just Smaller value)
last occurence of that
0 1 2 3 4 5 6 7 8 9 10
10 10 10 20 20 30 40 40 40 40 40
floor(40) = 5th {LI of 30}
floor(30) = 4th {LI of 20}
floor(20) = 2nd {LI of 10}
floor(10) = -1 {no floor value}
floor(45) = 10th {LI of 40}
floor(35) = 5th {LI of 30}
floor(25) = 4th {LI of 20}
floor(18) = 2nd {LI of 10}
floor(5) = -1 {no floor value}

Time → O(log n)

```
// n is the size of array
static int findFloor(long arr[], int n, long x)
{
    int left = 0, right = n - 1;
    int ans = -1;

    while(left <= right){
        int mid = left + (right - left) / 2;
        if(arr[mid] <= x){
            ans = mid;
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    return ans;
}
```

Note :- It is for either Last Occurence of target else it is just smaller value's last occurence



Transition point

left, right, mid

{ 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1 }
0 1 2 3 4 5 6 7 8 9 10

first occurrence of 1

left = 0 , right = 10 , ans = -1 → 8 → 7

if(arr[mid] == 0)
left = mid + 1

else arr[mid] == 1
right = mid - 1,
ans = mid

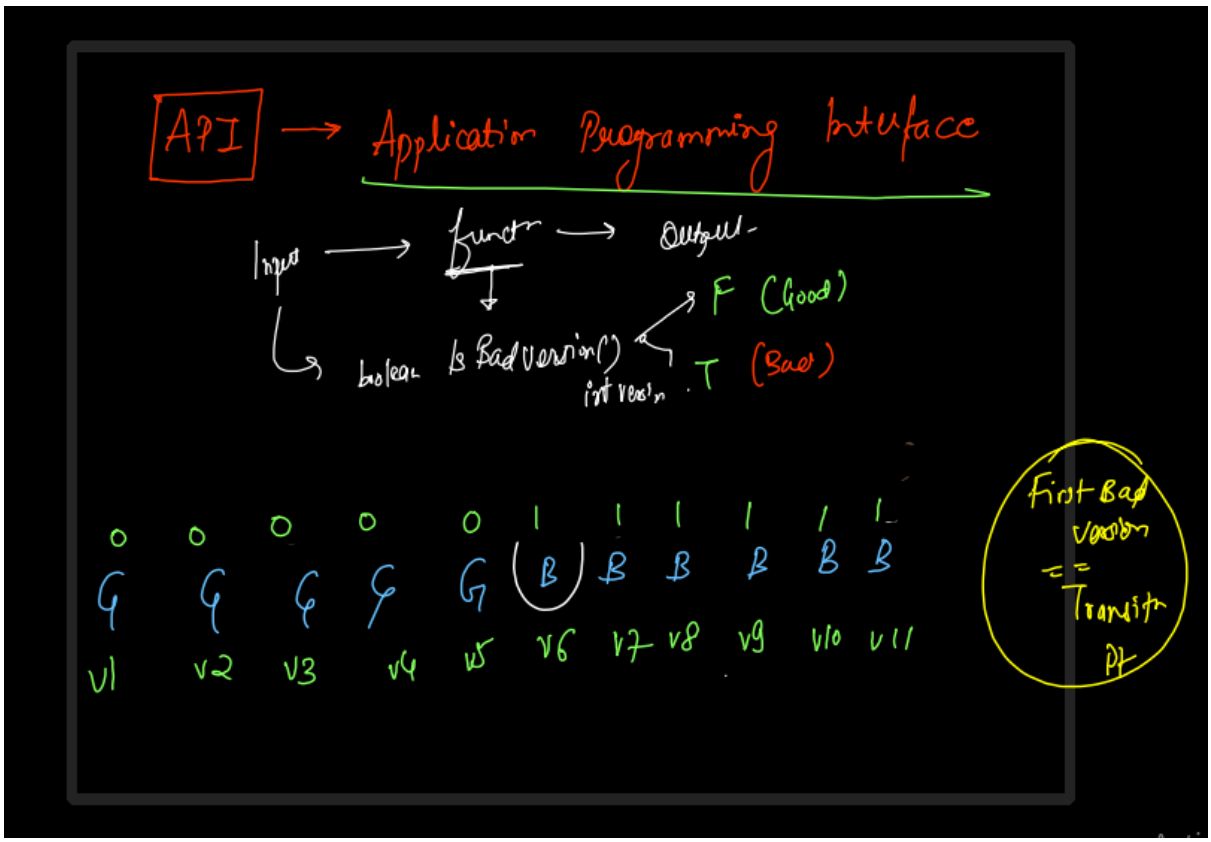
Corner case 1 :- All elements are 0
⇒ no transition point

{ 0, 0, 0, 0, 0 } ans = -1

Corner Case 2 :- All elements are 1
⇒ 0th index is transition pt of First occurrence of 1

{ 1, 1, 1, 1, 1 } ans = 0

```
int transitionPoint(int arr[], int n) {
    int left = 0, right = n - 1, ans = -1;

    while(left <= right){
        int mid = left + (right - left) / 2;

        if(arr[mid] == 0){
            left = mid + 1;
        } else {
            right = mid - 1;
            ans = mid;
        }
    }

    return ans;
}
```



```java
public int firstBadVersion(int n) {
    int left = 1, right = n, ans = n;

    while(left <= right){
        int mid = left + (right - left) / 2;

        if(isBadVersion(mid) == false) {
            left = mid + 1;
        } else {
            right = mid - 1;
            ans = mid;
        }
    }

    return ans;
}
```

→ arr[mid] <= 0

LC153

Rotated Sorted array

pivot / minimum value

idx	0	1	2	3	4	5	6	7	8	9
eg	50	60	70	80	90	100	10	20	30	40

Corner Case
⇒ Rotated Sorted Array

AC Rotations

10, 20, 30, 40, 50, 60, 70 k=0

20, 30, 40, 50, 60, 70, 10 k=1

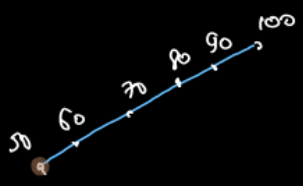
30, 40, 50, 60, 70, 10, 20 k=2

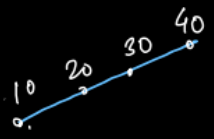
40, 50, 60, 70, 10, 20, 30 k=3

50, 60, 70, 10, 20, 30, 40 k=4

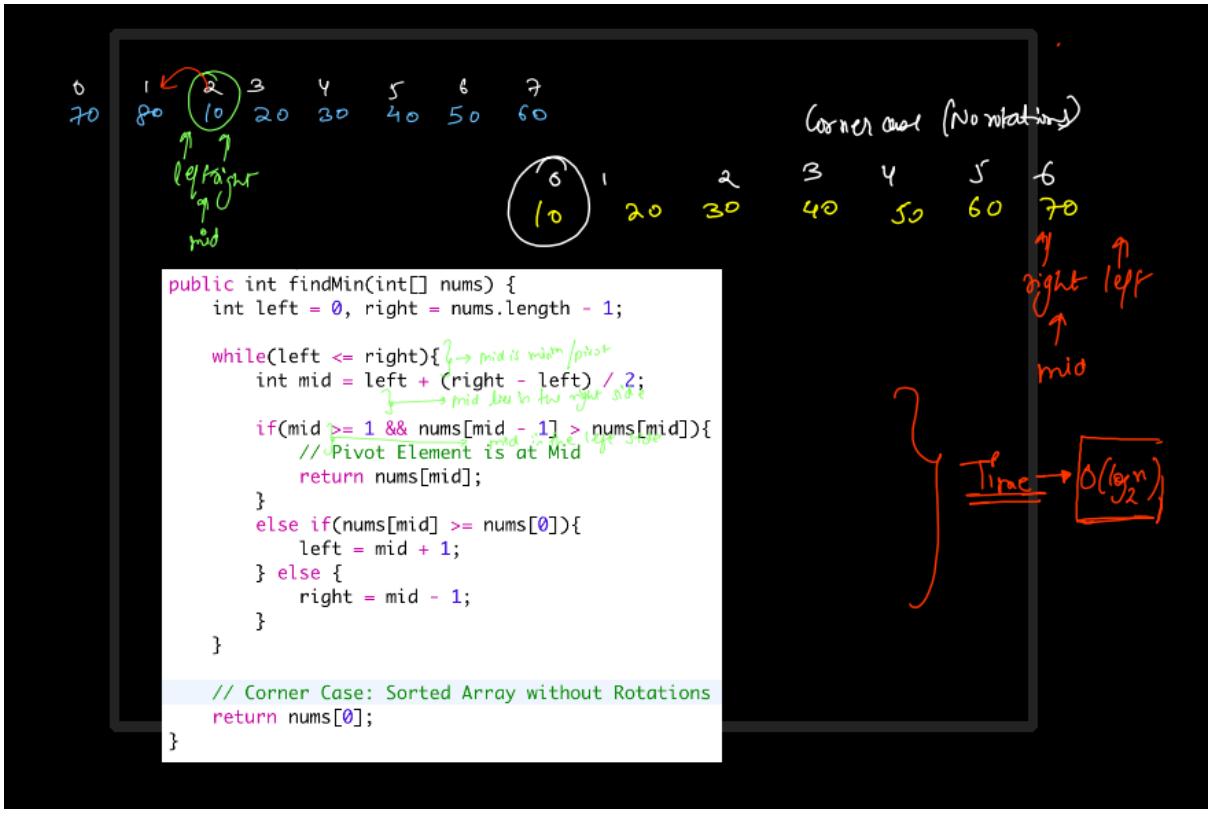
60, 70, 10, 20, 30, 40, 50 k=5

70, 10, 20, 30, 40, 50, 60 k=6

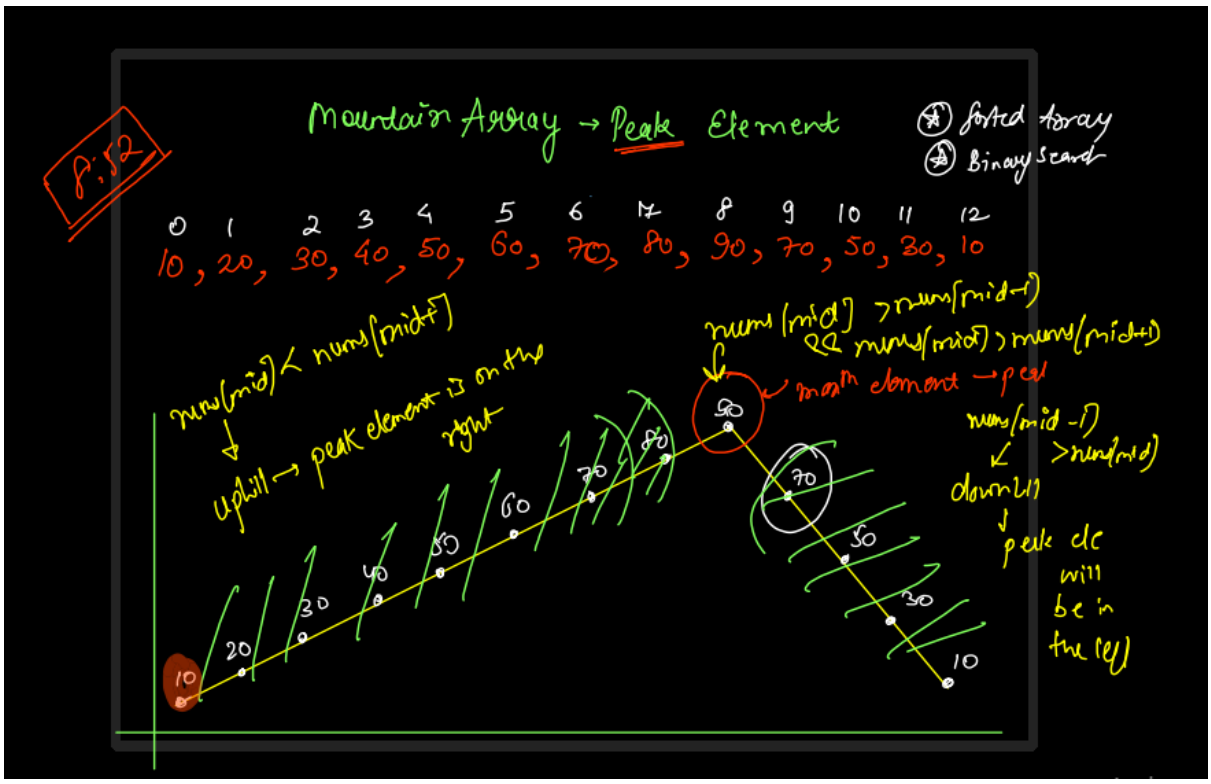




minimum value
↓
pivot index

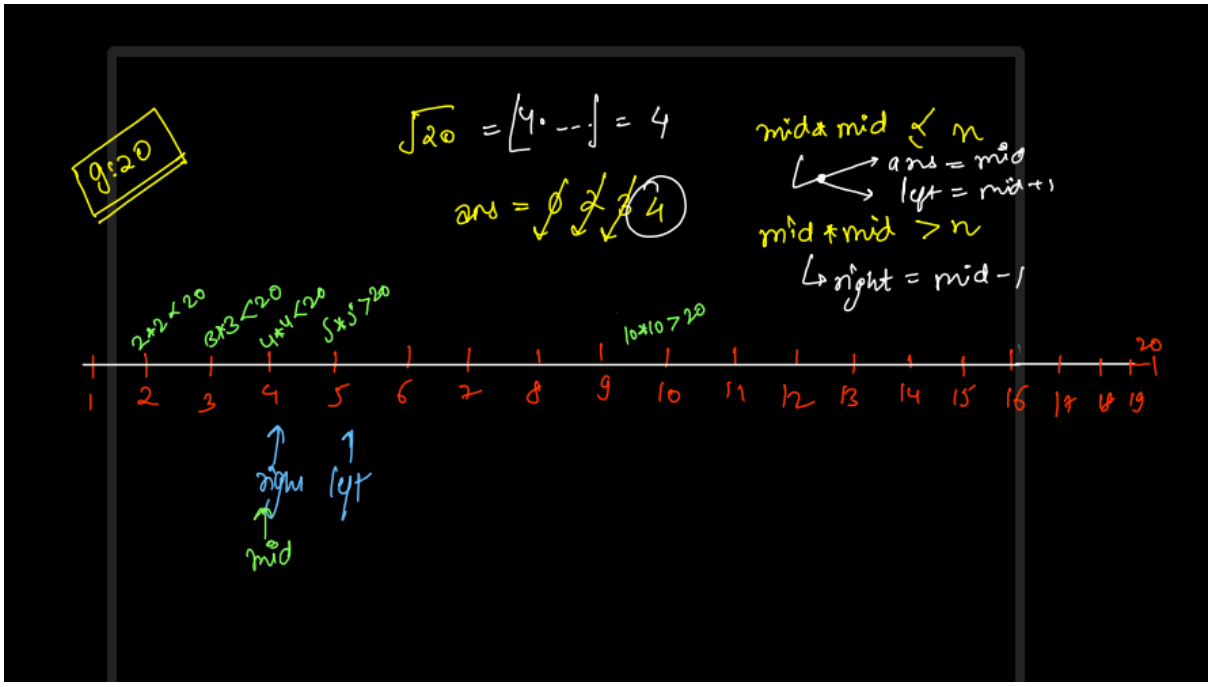



```java
public int findMin(int[] nums) {
    int left = 0, right = nums.length - 1;

    while(left <= right){
        int mid = left + (right - left) / 2;

        if(mid >= 1 && nums[mid - 1] > nums[mid]){
            // Pivot Element is at Mid
            return nums[mid];
        }
        else if(nums[mid] >= nums[0]){
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    // Corner Case: Sorted Array without Rotations
    return nums[0];
}
```



Time → $O(\log_2 N)$

```
int left = 1, right = nums.length - 2;
while(left <= right){
    int mid = left + (right - left) / 2;

    if(nums[mid] > nums[mid - 1] && nums[mid] > nums[mid + 1]){
        // Peak Element
        return mid;
    }

    else if(nums[mid] < nums[mid + 1]){
        // Uphill
        left = mid + 1;
    } else {
        // Downhill
        right = mid - 1;
    }
}
```



Binary No System

Decimal System No. → Base 10

0	10	20	80	----90	100
1	11	⋮		⋮	
2	12	⋮		⋮	
3	13	⋮		⋮	
4	14	⋮		⋮	
5	15	⋮			
6	16	⋮			
7	17				
8	18				
9	19	29	39	89	

7 digit

(OFF) 0 1 (ON)

Binary System → Base 2

000 → 0	1000 → 8
001 → 1	1001 → 9
010 → 2	1010 → 10
011 → 3	1011 → 11
100 → 4	1100 → 12
101 → 5	1101 → 13
110 → 6	1110 → 14
111 → 7	1111 → 15

Bit

$$\begin{array}{r} 1\ 1\ 0 \\ 9\ 7\ 3 \\ +\ 2\ 8\ 5 \\ \hline 1\ 2\ 5\ 8 \end{array}$$

$(10)_2 \neq 1+1+1+1+1+\cdots$ 10 times

$= 1+1$

$(10)_{10} = 1+1+1+1+\cdots$ 10 times

Adding 2 bits

$$\begin{array}{r} 0 \\ +\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +\ 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 1 \\ \hline 10 \end{array}$$

$(10)_2 = 1*2^1 + 0*2^0$

$= 2 + 0 = (2)_{10}$

$(10)_2 = 1*2^1 + 0*2^0$

$= 2 + 0 = (2)_{10}$

Decimal No

$(973)_{10} = 9*10^2 + 7*10^1 + 3*10^0$

$= 900 + 70 + 3 = 973$

$(285)_{10} = 2*10^2 + 8*10^1 + 5*10^0$

$= 200 + 80 + 5 = 285$

Binary No

$(10110011)_2 = 1*2^0 + 0*2^1 + 0*2^2 + 1*2^3 + 1*2^4 + 0*2^5 + 1*2^6 + 0*2^7 + 1*2^8$

$= (1 + 8 + 16 + 64 + 256)_{10} = (357)_{10}$

Binary Addition

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1\ 1 \\ 1\ 0\ 1\ 1\ 0\ 0\ 1 \\ +\ 1\ 1\ 0\ 0\ 0\ 1\ 1 \\ \hline 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0 \end{array}$$

$$\begin{array}{r} 1\ 1\ 1\ \ \ 1 \\ 1\ 0\ 1\ 1\ 0\ 0\ 1 \\ +\ 1\ 1\ 1\ 1\ 1\ 0\ 1 \\ \hline 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0 \end{array}$$

$(1+1+1) = (3)_{10}$

$(11)_2 = 1*2^1 + 1*2^0$

$= 2+1 = (3)_{10}$

Decimal Subtraction

Borrow → 10

```
  9 8 3 6
- 2 4 7 1
---------
  7 3 5 5
---------
```

$9836 = 9*10^3 + 8*10^2 + 3*10^1 + 6*10^0$

```
  6 A 2 4 9
- 6 5 9 2
---------
  0 6 5 7
---------
```

$8*10^2 + 3*10^1 \rightarrow 7*10^2 + 13*10^1$

830

$700 + 130 = 830$

Binary Subtraction

```
  1 0 1 0 0 1 0 1
- 1 0 0 1 1 0 1 1 0
-------------------
  0 0 0 1 0 1 1 1 1
-------------------
```

```
  1 0 0 0 0 0 0 1
- 1 0 0 1 1 1 1 1
-----------------
  0 0 0 1 0 1 0 1 0
-----------------
```

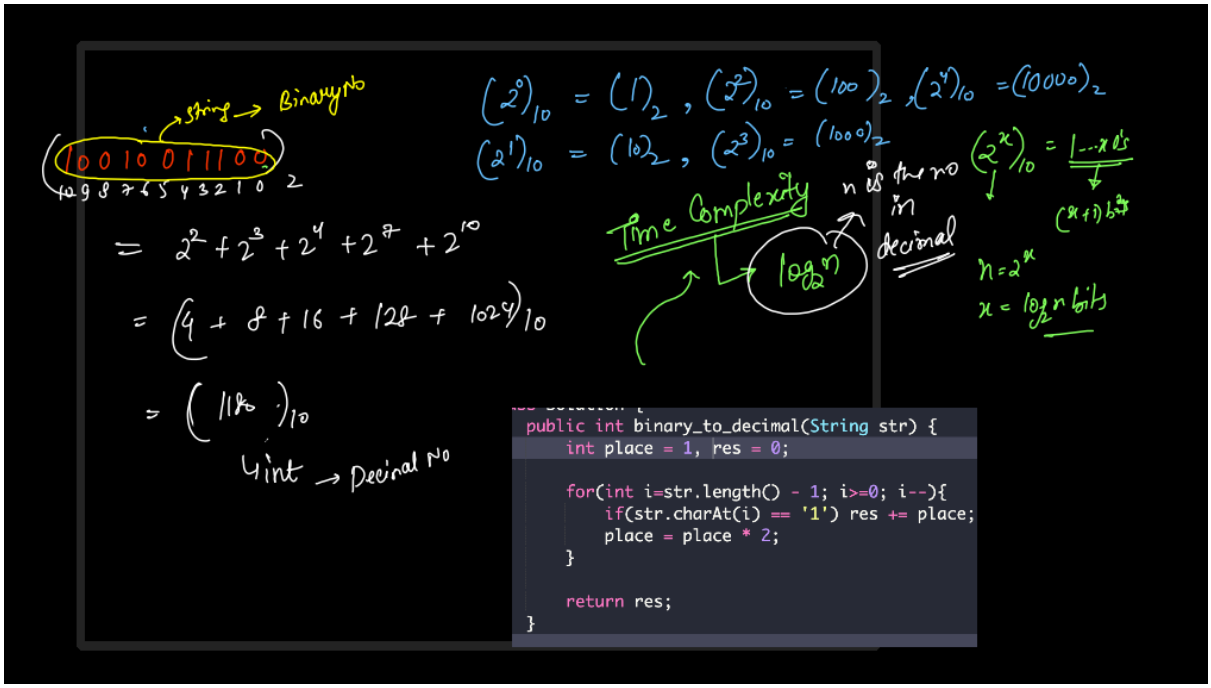
$1 - 0 = 1$

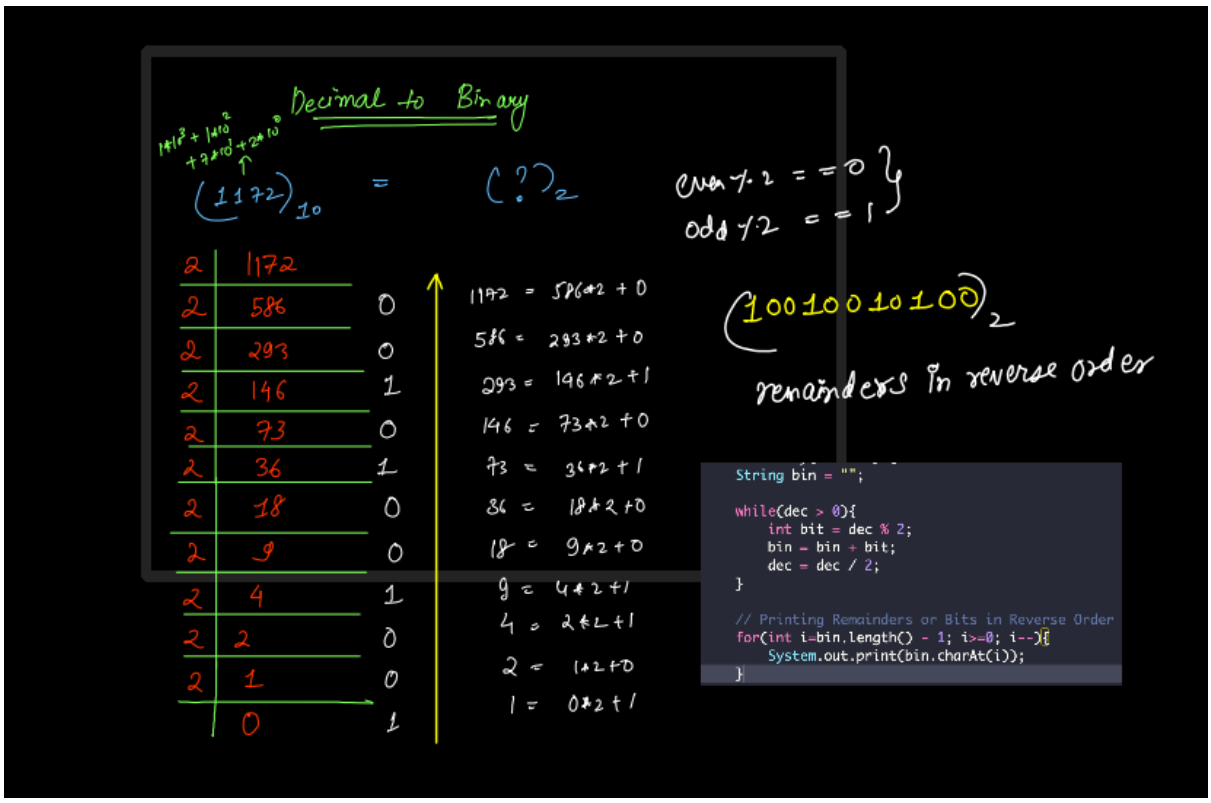
$0 - 0 = 0$

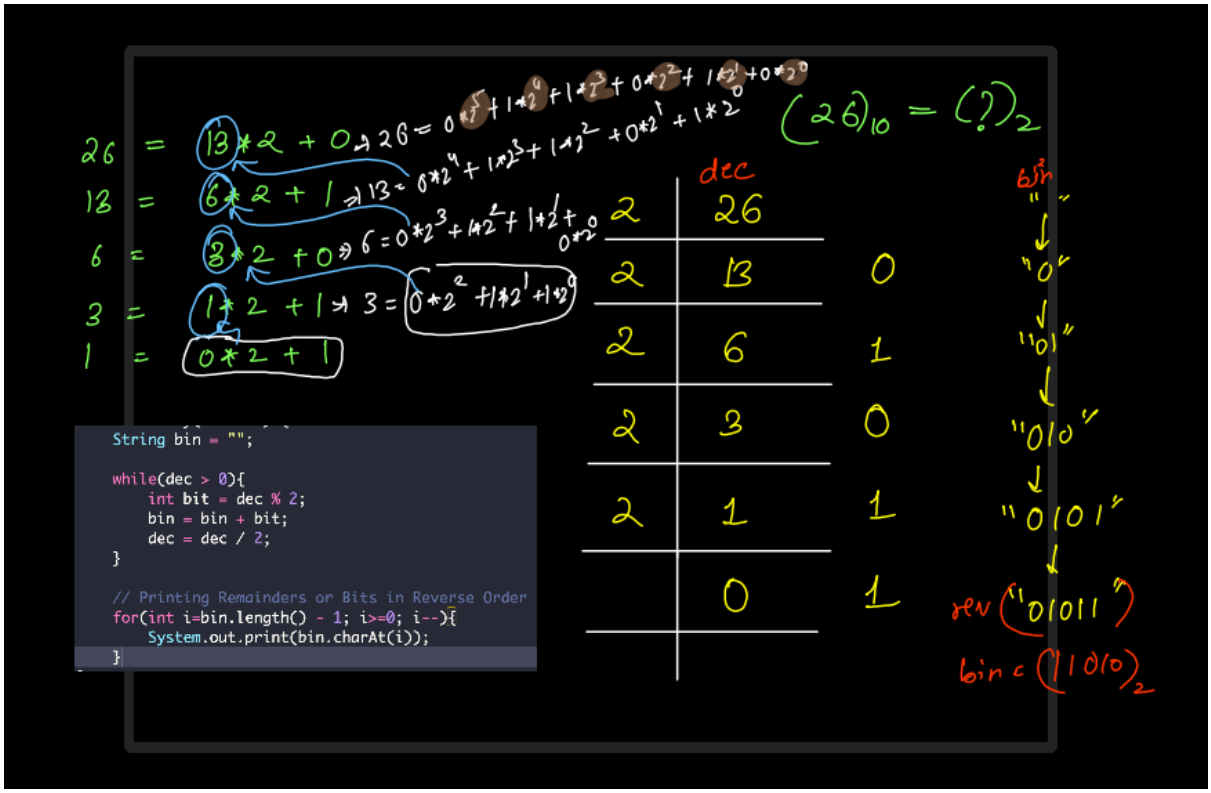
$1 - 1 = 0$

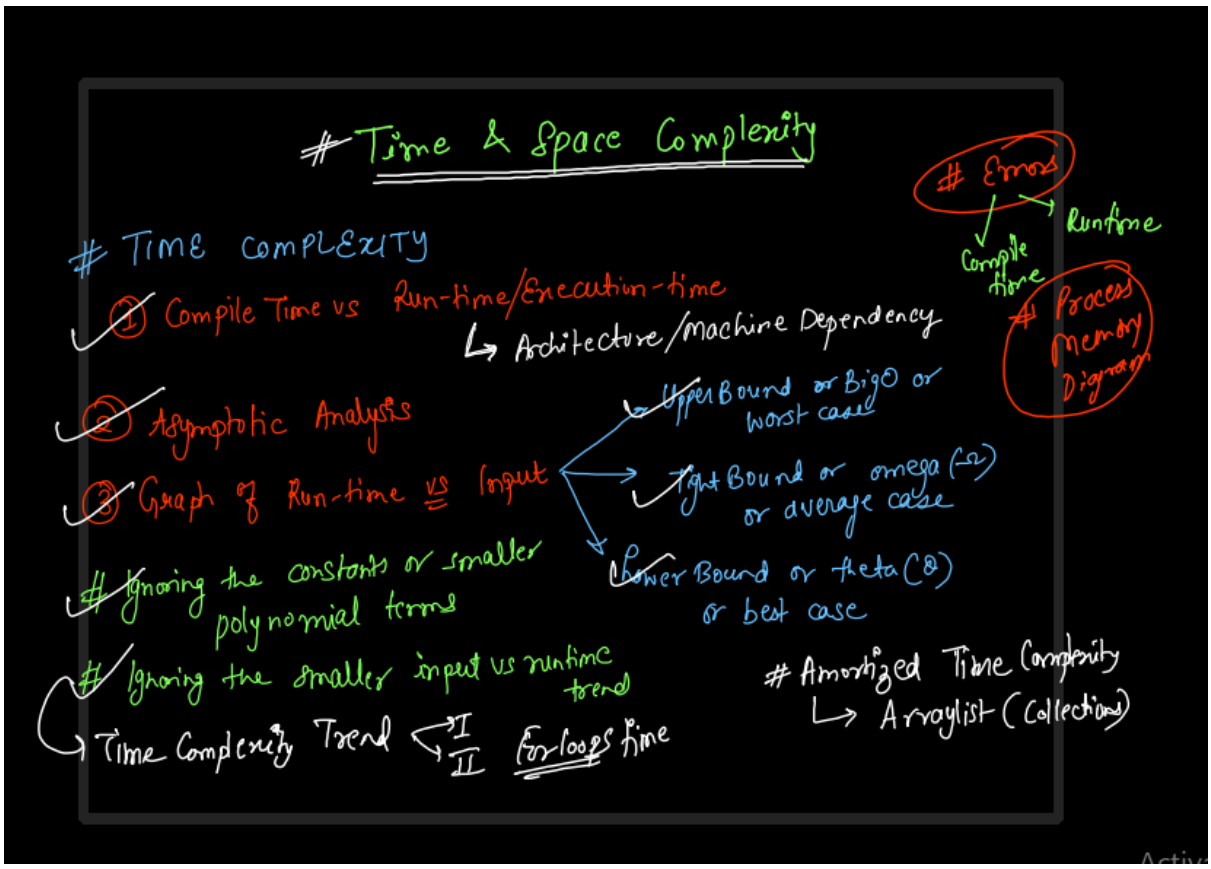
$-1 \rightarrow 0 \rightarrow +2$

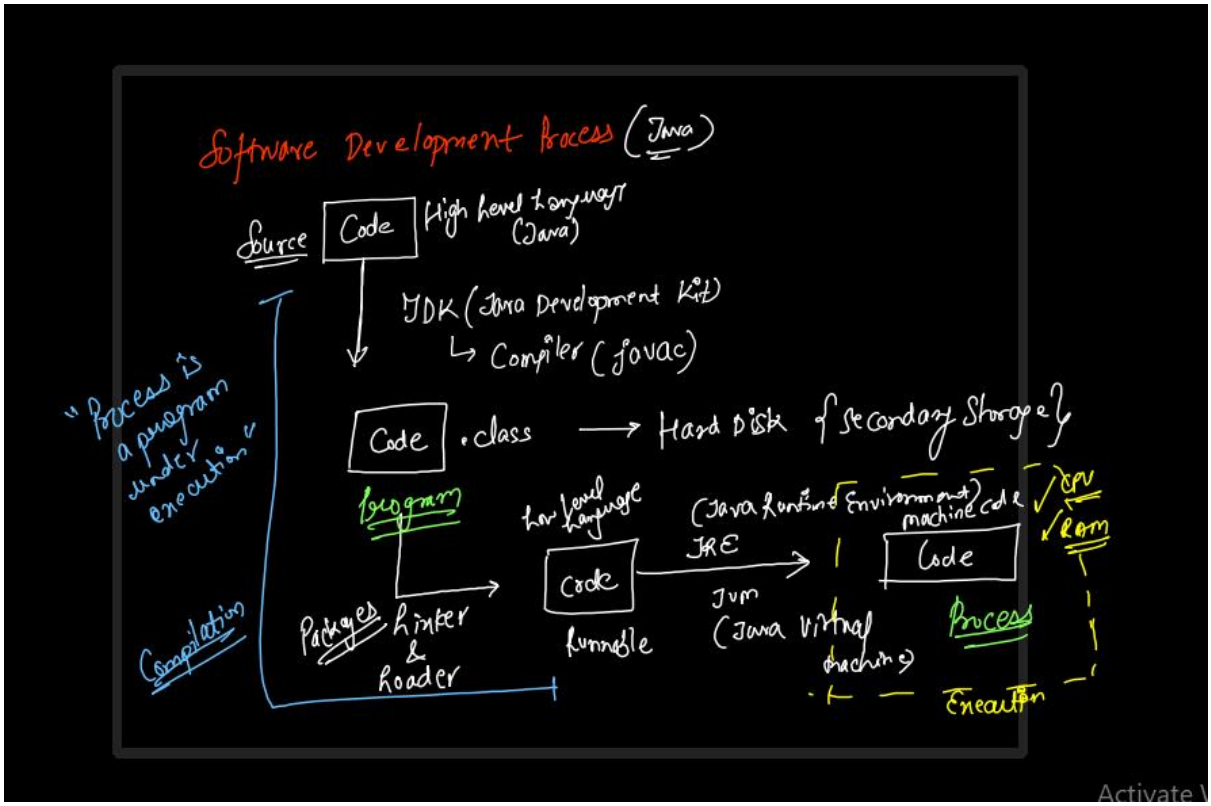
$0 - 1$

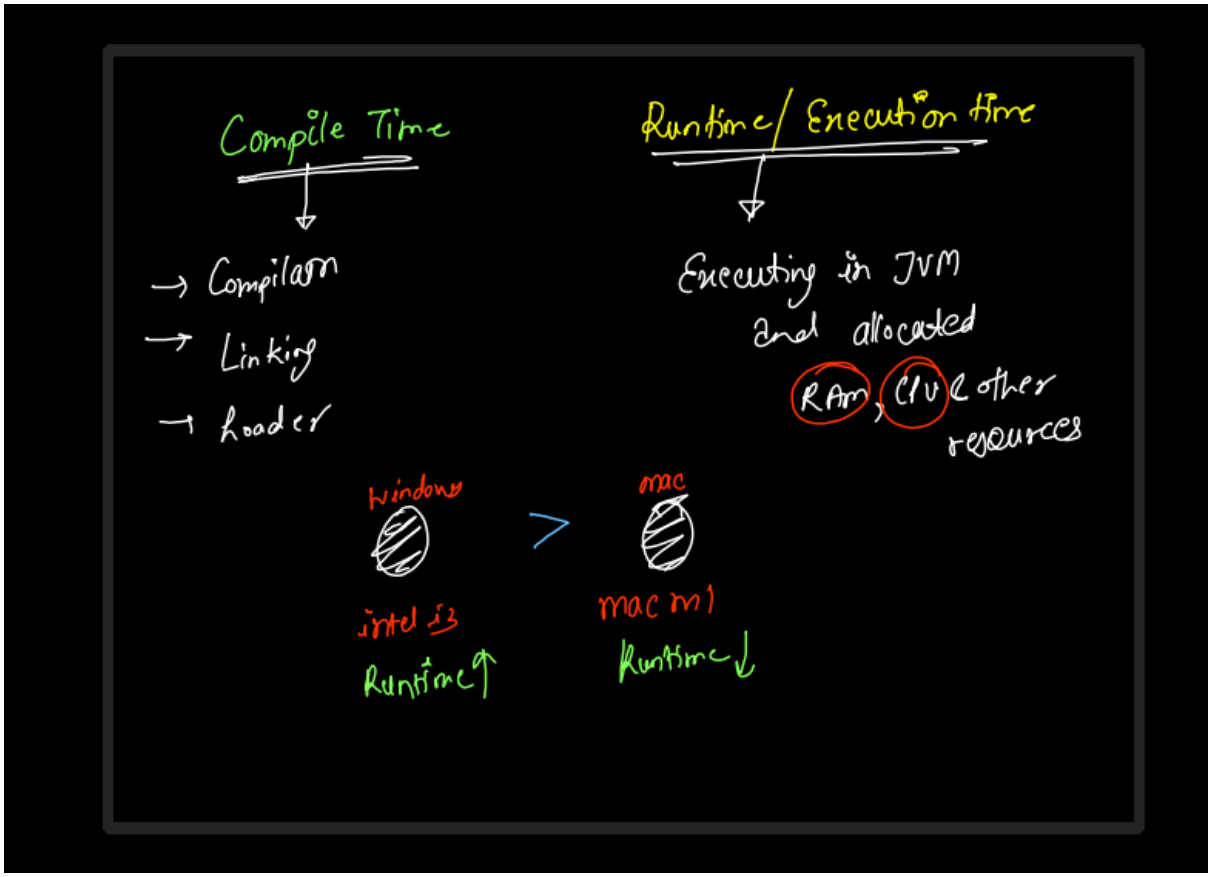

String → Binary no
1 0 0 1 0 0 1 1 1 0 0
10 9 8 7 6 5 4 3 2 1 0 2
= 2^2 + 2^3 + 2^4 + 2^7 + 2^10
= (4 + 8 + 16 + 128 + 1024)10
= (1180)10
4 int → Decimal no
(2^0)10 = (1)2 , (2^2)10 = (100)2 , (2^4)10 = (10000)2
(2^1)10 = (10)2 , (2^3)10 = (1000)2
Time Complexity
log2n
n is the no in decimal
(2^x)10 = 1---x 0's
(x+1) bits
n = 2^x
x = log2n bits
public int binary_to_decimal(String str) {
int place = 1, res = 0;
for(int i=str.length() - 1; i>=0; i--){
if(str.charAt(i) == '1') res += place;
place = place * 2;
}
return res;
}


Decimal to Binary
1*10^3 + 1*10^2 + 7*10^1 + 2*10^0
(1172)10 = (?)2
even %2 == 0
odd %2 == 1
2 | 1172
2 | 586 0
2 | 293 0
2 | 146 1
2 | 73 0
2 | 36 1
2 | 18 0
2 | 9 0
2 | 4 1
2 | 2 0
2 | 1 0
0 1
1172 = 586*2 + 0
586 = 293*2 + 0
293 = 146*2 + 1
146 = 73*2 + 0
73 = 36*2 + 1
36 = 18*2 + 0
18 = 9*2 + 0
9 = 4*2 + 1
4 = 2*2 + 1
2 = 1*2 + 0
1 = 0*2 + 1
(10010010100)2
remainders in reverse order
String bin = "";
while(dec > 0){
int bit = dec % 2;
bin = bin + bit;
dec = dec / 2;
}
// Printing Remainders or Bits in Reverse Order
for(int i=bin.length() - 1; i>=0; i--){
System.out.print(bin.charAt(i));
}


26 = 13*2 + 0 ⇒ 26 = 0*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0
13 = 6*2 + 1 ⇒ 13 = 0*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0
6 = 3*2 + 0 ⇒ 6 = 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0
3 = 1*2 + 1 ⇒ 3 = 0*2^2 + 1*2^1 + 1*2^0
1 = 0*2 + 1
(26)10 = (?)2
dec
2 | 26
2 | 13 | 0
2 | 6 | 1
2 | 3 | 0
2 | 1 | 1
0 | 1
bin
"1"
"01"
"010"
"0101"
"01011"
rev("01011")
bin = (11010)2
String bin = "";
while(dec > 0){
int bit = dec % 2;
bin = bin + bit;
dec = dec / 2;
}
// Printing Remainders or Bits in Reverse Order
for(int i=bin.length() - 1; i>=0; i--){
System.out.print(bin.charAt(i));
}


Time & Space Complexity
TIME COMPLEXITY
1. Compile Time vs Run-time/Execution-time
↳ Architecture/machine Dependency
2. Asymptotic Analysis
3. Graph of Run-time vs Input
Upper Bound or Big O or Worst case
Tight Bound or omega (Ω) or average case
Lower Bound or theta (Θ) or best case
Ignoring the constants or smaller polynomial terms
Ignoring the smaller input vs runtime trend
Time Complexity Trend ⇐ I / II For loops time
Amortized Time Complexity
↳ Arraylist (Collections)
Errors
Compile time
Runtime
Process memory Diagram


Software Development Process (Java)
Source
Code
High level language (Java)
JDK (Java Development Kit)
↳ Compiler (javac)
Code
.class
Hard disk of secondary Storage
"Process is a program under execution"
Program
low level language
Code
Runnable
JRE
(Java Runtime Environment)
machine code
Code
CPU
Ram
Process
Execution
Compilation
Packages
Linker & Loader
Jvm (Java virtual machine)


Compile Time
→ Compilation
→ Linking
→ Loader
Runtime / Execution time
Executing in JVM and allocated
RAM, CPU & other resources
window
>
mac
intel i3
Runtime ↑
mac m1
Runtime ↓

Algorithms

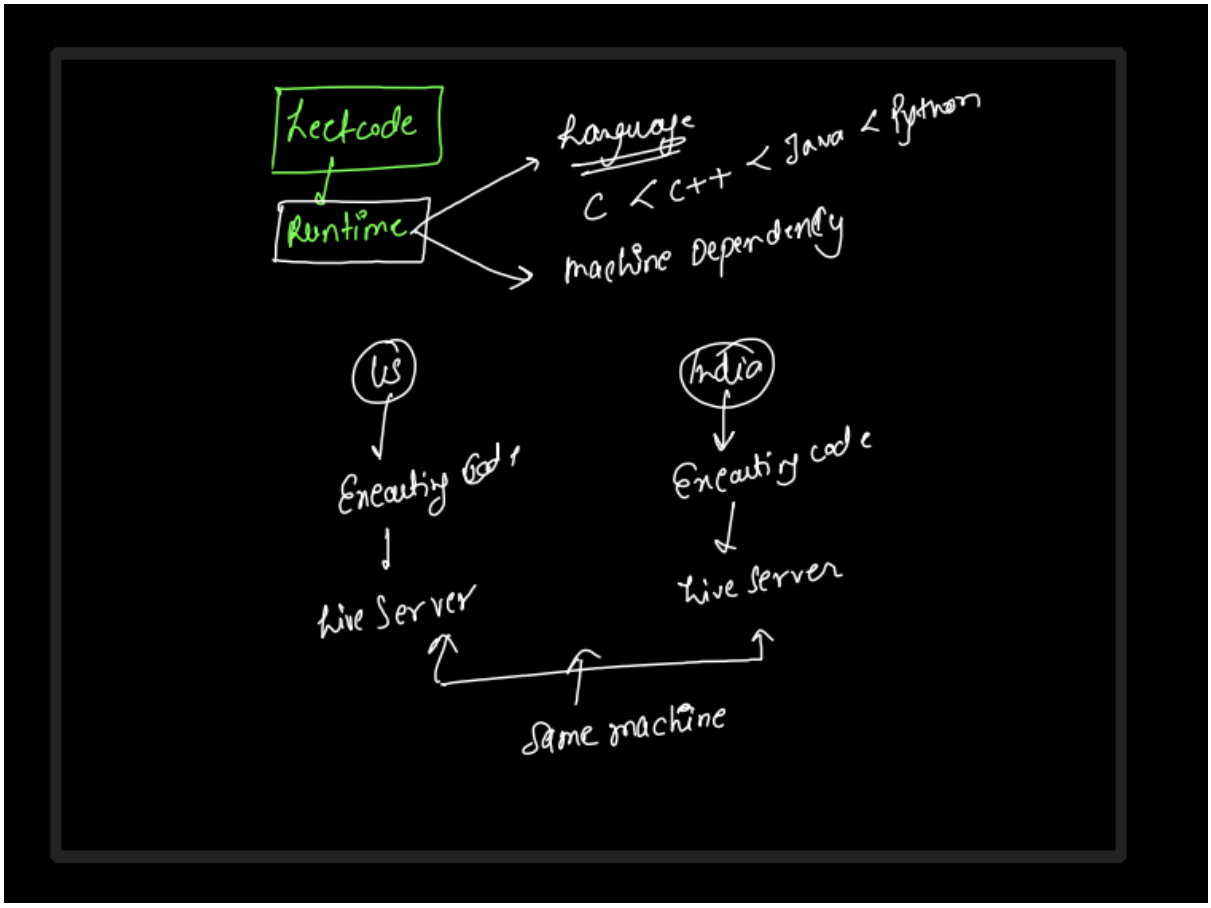
	Runtime i3, 4gb	Runtime m1, 16gb
Linear search		
n = 10	100 ms	10 ms
*10	*10	*10
n = 100	1000 ms	100 ms
Binary Serch		
n = 10	20 ms	1 ms
n = 100	40 ms	3 ms

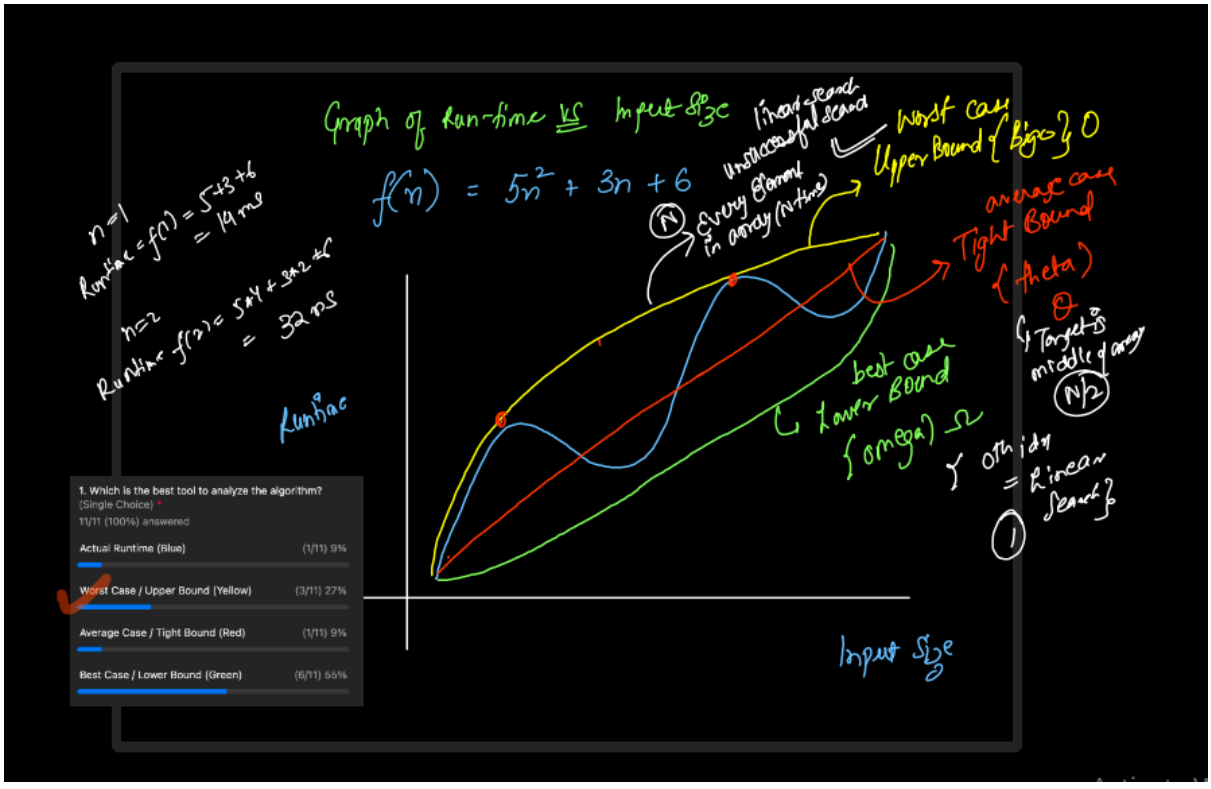
To compare algorithms based on runtime, you need to have the same machine.

Asymptotic Analysis

⇒ Analysis of rate of growth of runtime with respect to the input size

- Architecture neutrality (32bit/64bit) (8gbram/32gbram)
- machine independency (i3 = m1)


Leetcode
Runtime
Language
C < C++ < Java < Python
machine Dependency
US
Executing Code
Live Server
India
Executing code
Live Server
Same machine


Graph of Run-time vs Input Size
f(n) = 5n² + 3n + 6
n=1
Runtime = f(1) = 5+3+6
= 14 ms
n=2
Runtime f(2) = 5×4 + 3×2 + 6
= 32 ms
Runtime
Worst case
Upper Bound {Big-O}
Linear search
Unsuccessful search
Every element in array (N times)
average case
Tight Bound
{theta}
Θ
{ Target is middle of array
N/2
best case
{ Lower Bound
{omega} Ω
{ array
= Linear Search }
1
Input Size
1. Which is the best tool to analyze the algorithm?
(Single Choice) *
11/11 (100%) answered
Actual Runtime (Blue) (1/11) 9%
Worst Case / Upper Bound (Yellow) (3/11) 27%
Average Case / Tight Bound (Red) (1/11) 9%
Best Case / Lower Bound (Green) (6/11) 55%

① ignore the smaller terms

② ignore the constant term

③ ignoring the smaller trend

$f(n) = 5n^2 + 3n + 6$

$UB(f(n)) = 14n^4$

$14n^3$

$14n^2 \rightarrow O(n^2)$

$n \geq 1$

$14n^4 \geq 5n^2 + 3n + 6$

$14n^3 \geq 5n^2 + 3n + 6$

$14n^2 + 3n + 6 \geq 5n^2 + 3n + 6$

$14n^2 \geq 5n^2 + 3n + 6$

$LB(f(n)) = 0$

n

n^2

$4n^2 \rightarrow \Omega(n^2)$

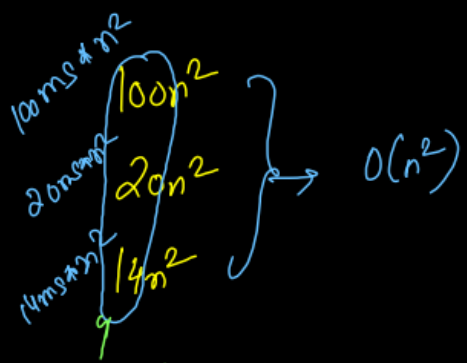
$0 \leq 5n^2 + 3n + 6$

$n \leq 5n^2 + 3n + 6$

$n^2 \leq 5n^2 + 3n + 6$

$4n^2 \leq 5n^2 + 3n + 6$

$TB(f(n)) = \frac{UB + LB}{2} = \frac{4n^2 + 14n^2}{2} = 9n^2 \rightarrow O(n^2)$



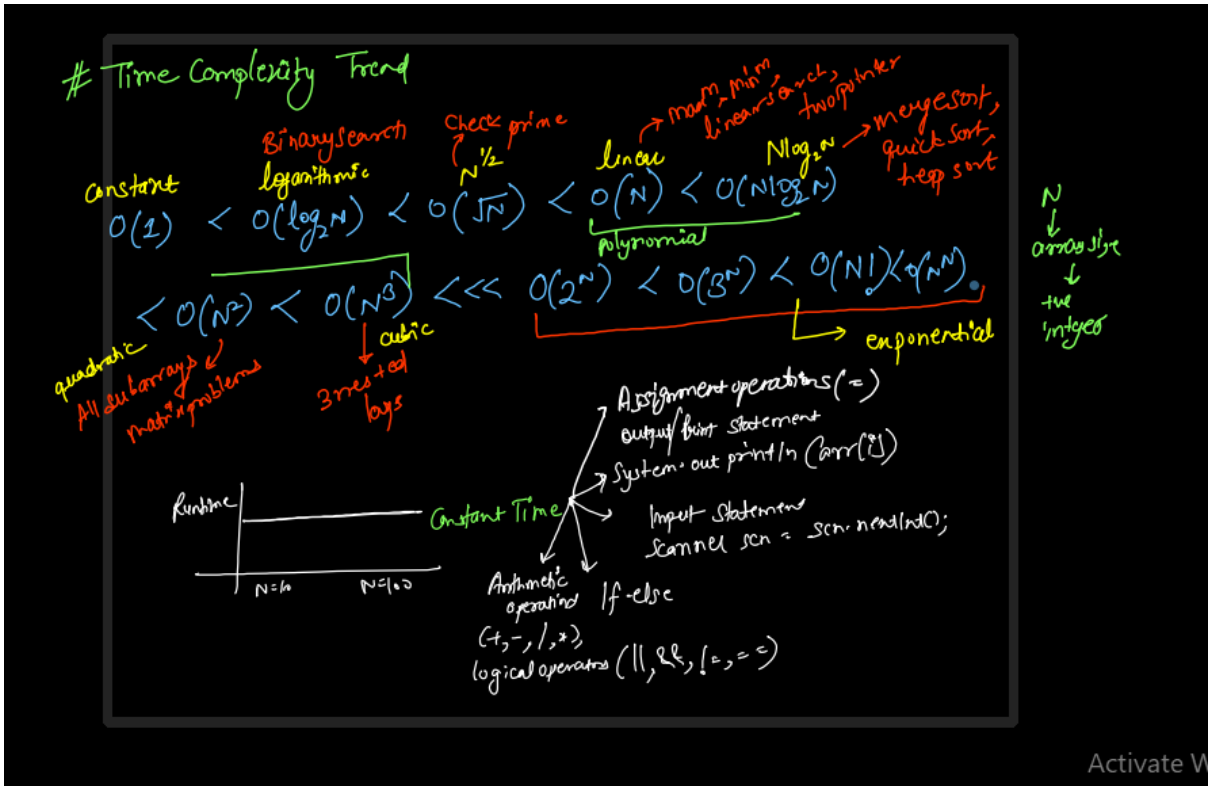
ignore the constant

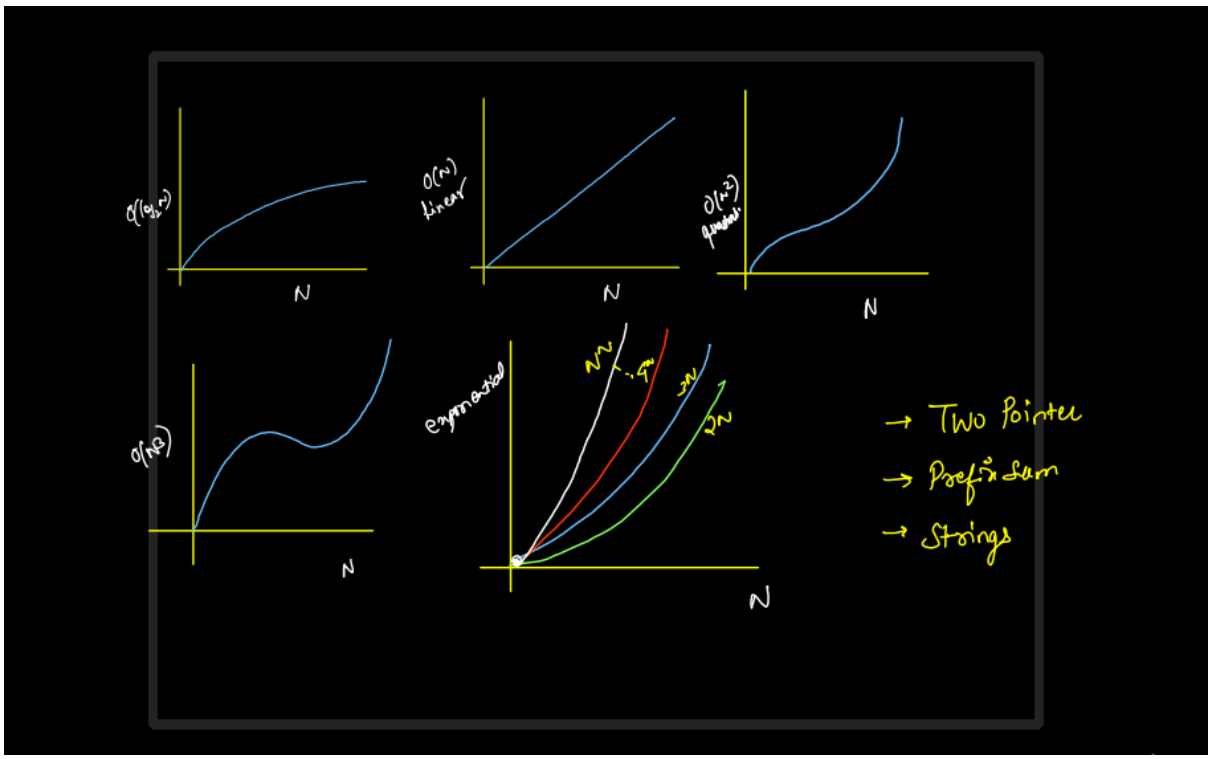
⇒ it does not affect the rate of growth of n^2

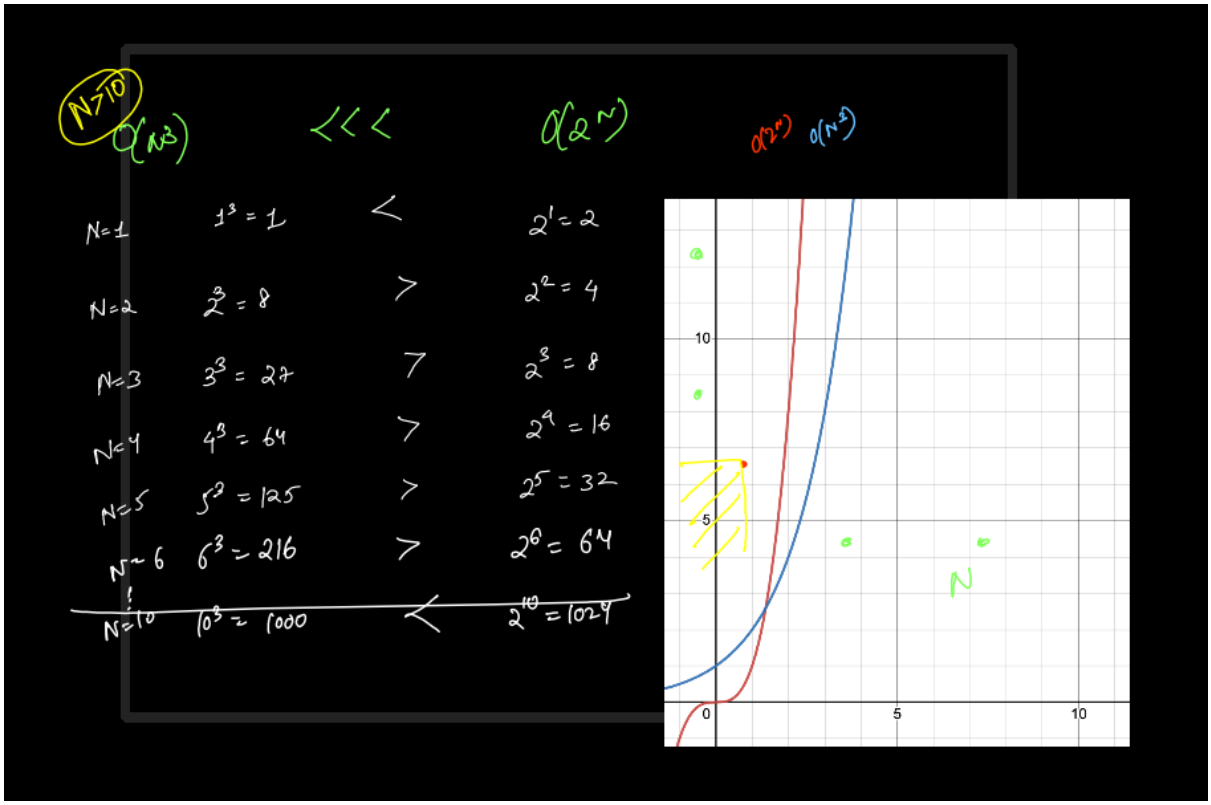
$n =$ 1 million

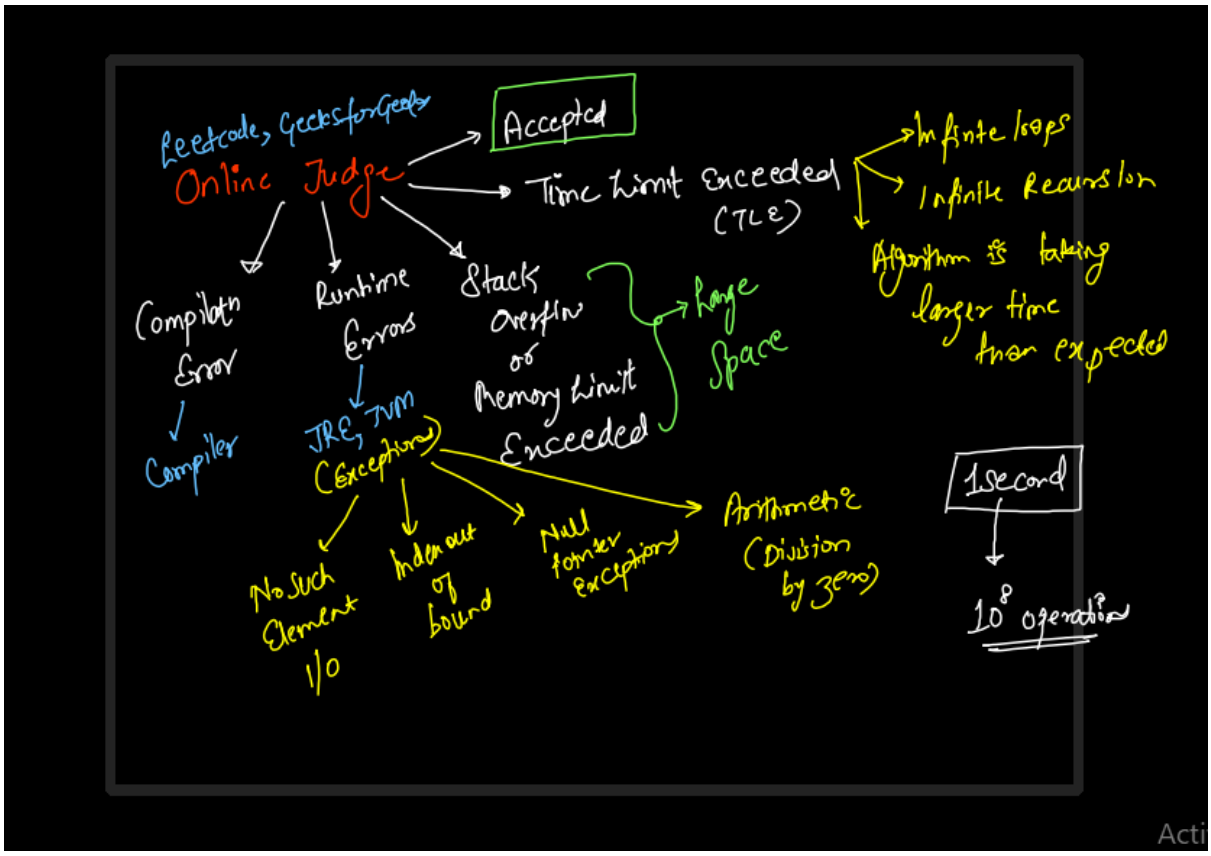
$100 * (1\text{million})^2$

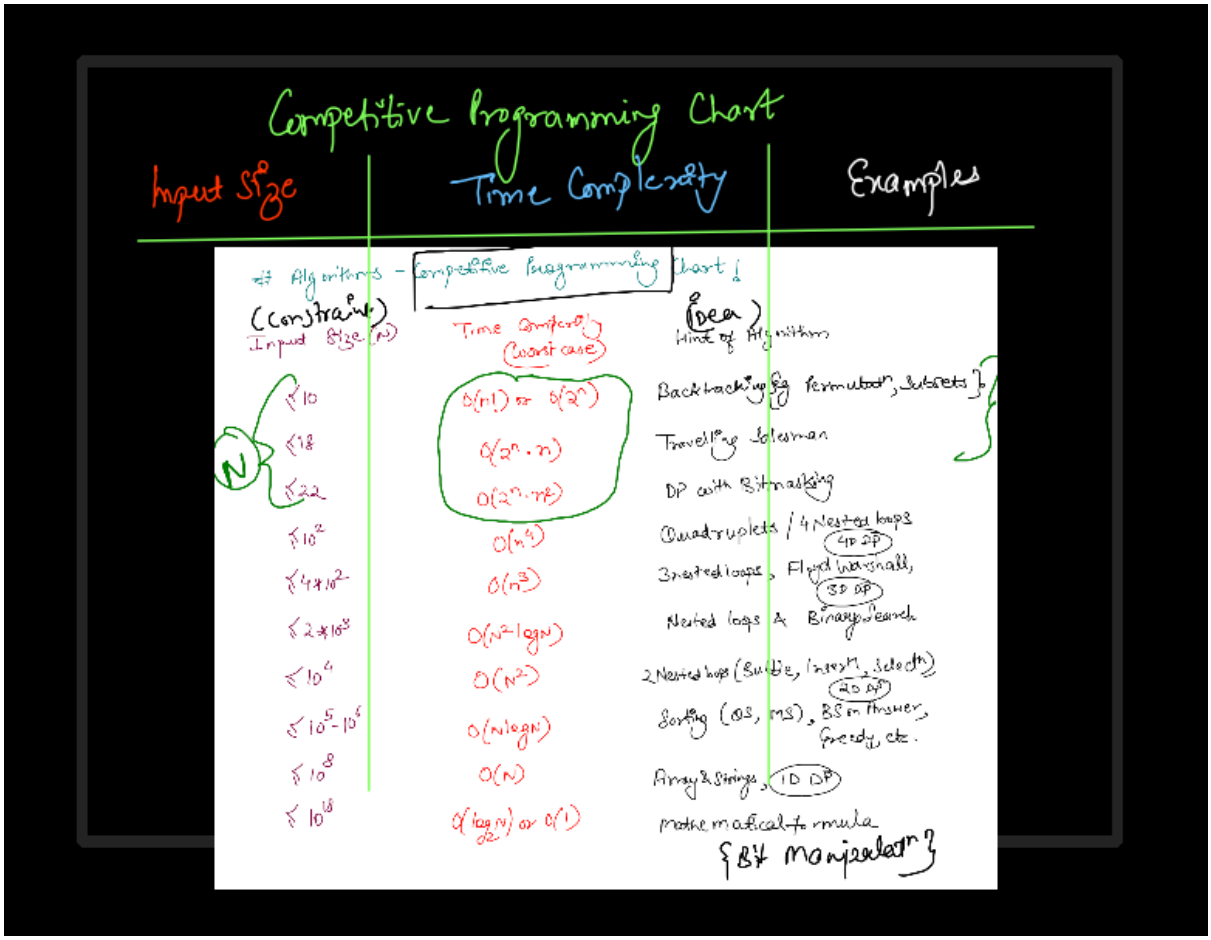
$20 * (1\text{million})^2$

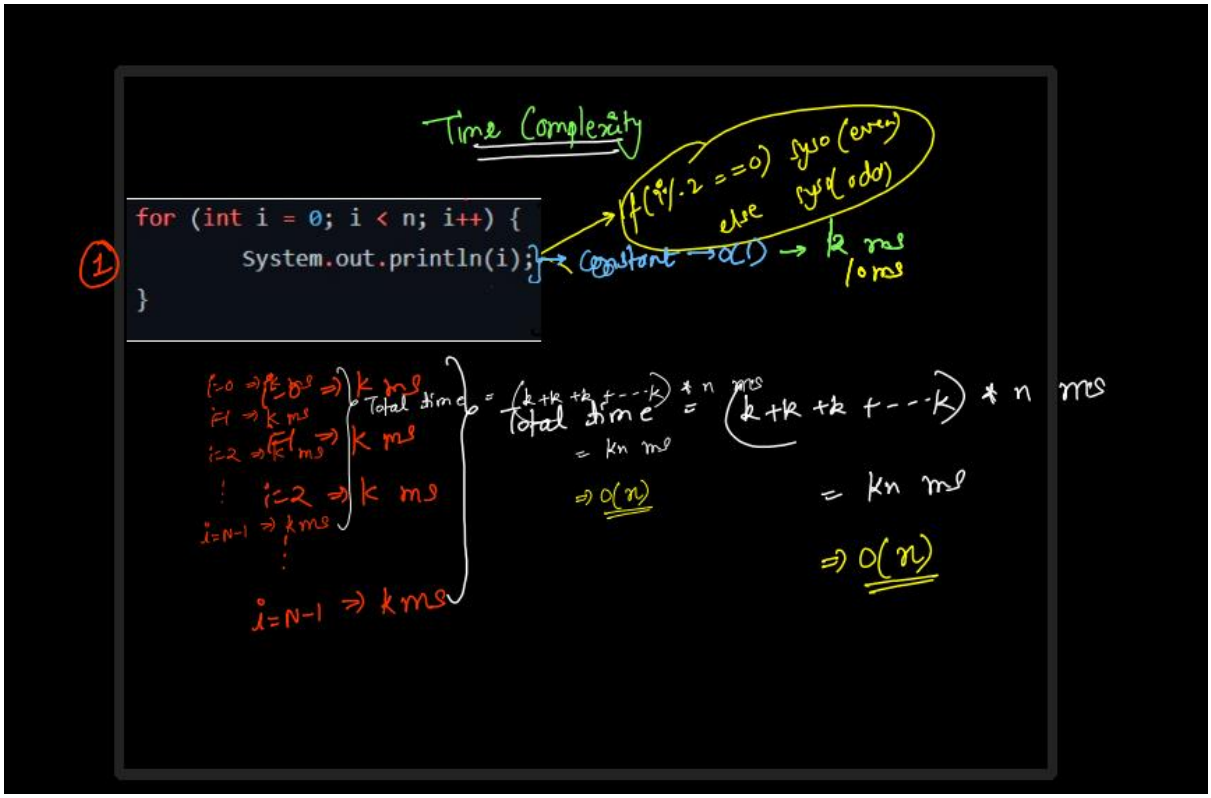

Time Complexity Trend
Constant
Binary search logarithmic
Check prime
linear
Nlog2N
polynomial
quadratic
All sub arrays
cubic
3 nested loops
exponential
N
array size
the integer
Runtime
Constant Time
N=10
N=100
Assignment operations (=)
output/print statement
System.out.println(arr[i])
Input Statements
Scanner scn = scn.nextInt();
Arithmetic operations
If-else
(+, -, /, *)
logical operators (||, &&, !=, ==)
Activate W


O(log2N)
O(N)
linear
O(N^2)
quadratic
O(N^3)
exponential
N
→ Two Pointer
→ Prefix Sum
→ Strings


N>10
O(N^3) <<< O(2^N)
O(2^N) O(N^3)
N=1 1^3 = 1 < 2^1 = 2
N=2 2^3 = 8 > 2^2 = 4
N=3 3^3 = 27 > 2^3 = 8
N=4 4^3 = 64 > 2^4 = 16
N=5 5^3 = 125 > 2^5 = 32
N=6 6^3 = 216 > 2^6 = 64
N=10 10^3 = 1000 < 2^10 = 1024
10
5
0
5
10
N


Leetcode, GeeksforGeeks
Online Judge
Accepted
Time limit Exceeded (TLE)
Infinite loops
Infinite Recursion
Algorithm is taking larger time than expected
Compilation Error
Compiler
Runtime Errors
JRE, JVM
(Exceptions)
Stack overflow or Memory limit Exceeded
Large Space
No Such Element
I/O
Index out of bound
Null pointer exceptions
Arithmetic (Division by Zero)
1 second
10^8 operations


Competitive Programming Chart
Input Size
Time Complexity
Examples
Algorithms - Competitive Programming Chart
(Constraint) Input Size (N)
Time Complexity (Worst case)
(Idea) Hint of Algorithm
N
≤10
≤18
≤22
≤10^2
≤4*10^2
≤2*10^3
≤10^4
≤10^5 - 10^6
≤10^8
≤10^18
O(n!) or O(2^n)
O(2^n * n)
O(2^n * n^2)
O(n^4)
O(n^3)
O(N^2 log N)
O(N^2)
O(N log N)
O(N)
O(log N) or O(1)
Backtracking (eg permutation, subsets)
Travelling Salesman
DP with Bitmasking
Quadruplets / 4 Nested loops
4D DP
3 nested loops, Floyd Warshall,
3D DP
Nested loops & Binary search
2 Nested loops (Bubble, Insertion, Selection)
2D DP
Sorting (QS, MS), BS in trees, greedy, etc.
Array & Strings, 1D DP
mathematical formula
{Bit manipulation}


Time Complexity
for (int i = 0; i < n; i++) {
System.out.println(i);
}
1
if(i%2==0) sys(even) else sys(odd)
constant → O(1) → k ms / 1 ms
i=0 ⇒ k ms
i=1 ⇒ k ms
i=2 ⇒ k ms
i=N-1 ⇒ k ms
Total time = (k+k+k+---k) * n ms = (k+k+k+---k) * n ms
= kn ms
⇒ O(n)
= kn ms
⇒ O(n)

(2)

```
for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
                System.out.println(i + " " + j);
        }
}
```

Nested loops

N=5

i=0 times: j=0, j=1, j=2, j=3, j=4

i=1 Local times: j=0, j=1, j=2, j=3, j=4

i=2 Local times: j=0, j=1, j=2, j=3, j=4

i=3 Local times: j=0, j=1, j=2, j=3, j=4

i=4 Local times: j=0, j=1, j=2, j=3, j=4

k*n + k*n + k*n + k*n + k*n ... = k*n*n

O(n²)

(3)

```
for (int i = 0; i < n; i++) {
        for (int j = n - 1; j >= 0; j--) {
                Sys o(i+j)
        }
}
```

N=5

i=0 times: j=4, j=3, j=2, j=1, j=0

i=1 Local times: j=0, j=1, j=2, j=3, j=4

i=2 Local times: j=0, j=1, j=2, j=3, j=4

i=3 Local times: j=0, j=1, j=2, j=3, j=4

i=4 Local times: j=0, j=1, j=2, j=3, j=4

kn + kn + kn + kn + kn

n times = O(kn*n) ⇒ O(n²)

④

```
for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
                System.out.println(i + " " + j);
        }
}
```

$k*(1+2+3+4+\dots n)$ ms

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

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

$O(n^2)$

N=5

i=0 j=0 → k*1 ms

i=1 j=0, j=1 → k*2 ms

i=2 j=0, j=1, j=2 → k*3 ms

i=3 j=0, j=1, j=2, j=3 → k*4 ms

i=4 j=0, j=1, j=2, j=3, j=4 → k*5 ms

⑤

```
for (int i = 1; i < n; i *= 2) {

}
```

N=16

i=0 $j=2^0$, $j=2^1$, $j=2^2$, $j=2^3$, $j=2^4$, $j=2^5$ ...

i=1 i=2 i=4 i=8 i=16 i=32

k + k + k + k + k = k*log n

$\Rightarrow O(\log_2 N)$

Multiplying or Dividing by 2

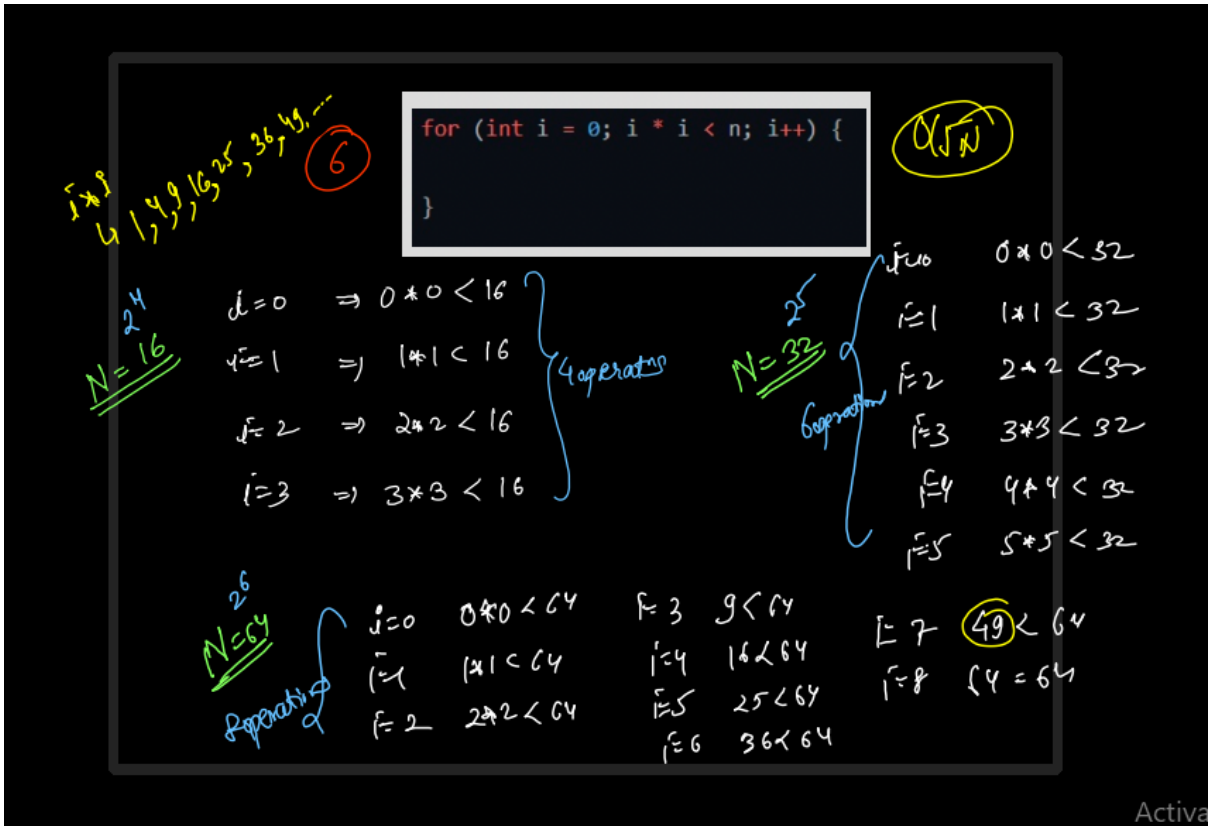
```
while (N >= 1)
{
    N = N/2;
}
```

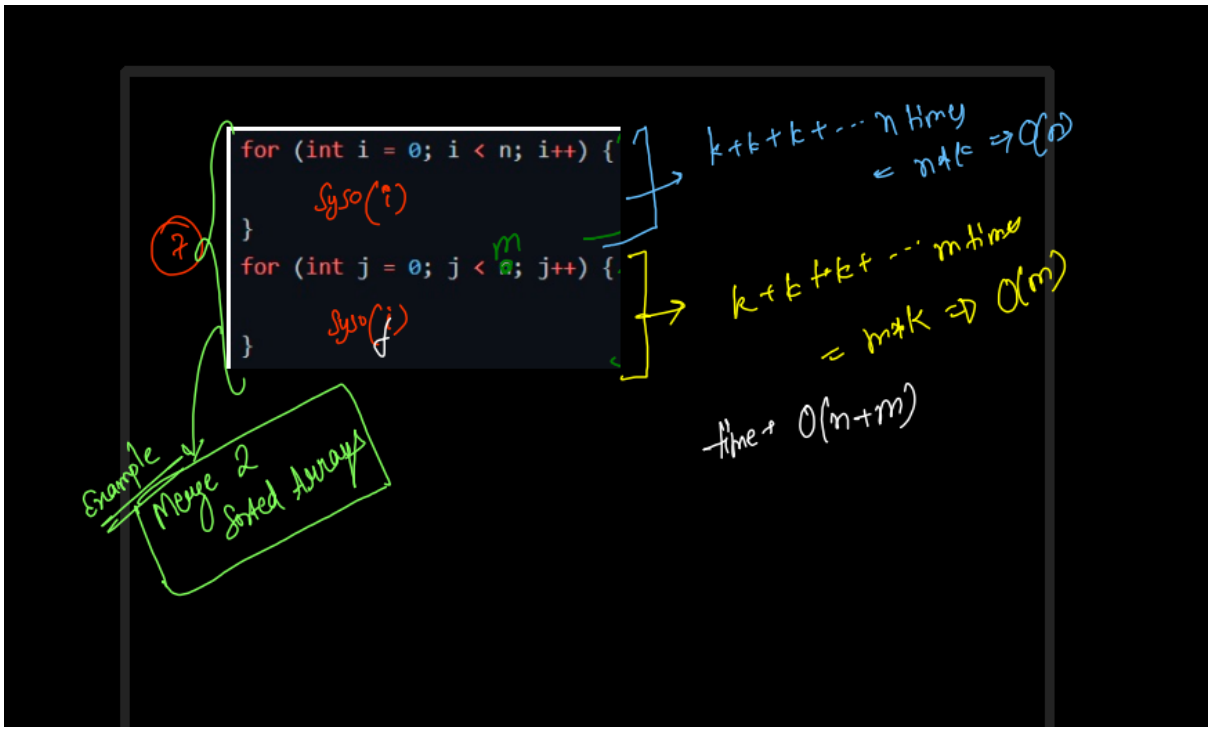
$2^5 \to 2^4 \to 2^3 \to 2^2 \to 2^1 \Rightarrow 5$ ops

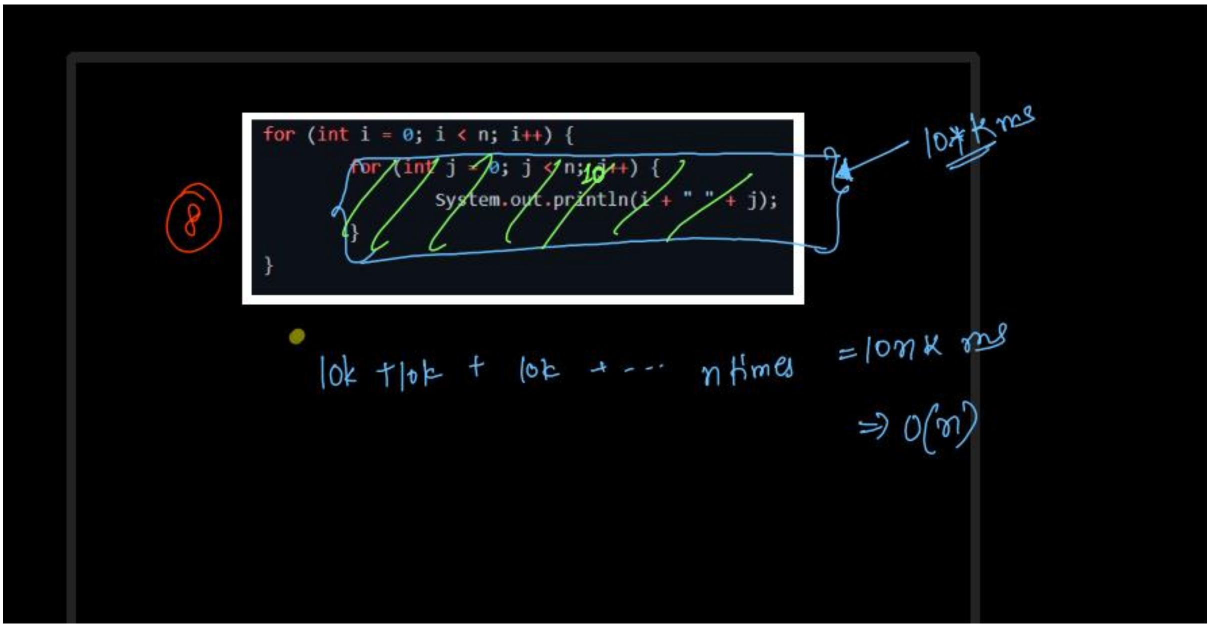
$2^{10} \to 2^9 \to 2^8 \dots \to 2^1 \Rightarrow 10$ ops

$N = 2^y \Rightarrow y = \log_2 N$

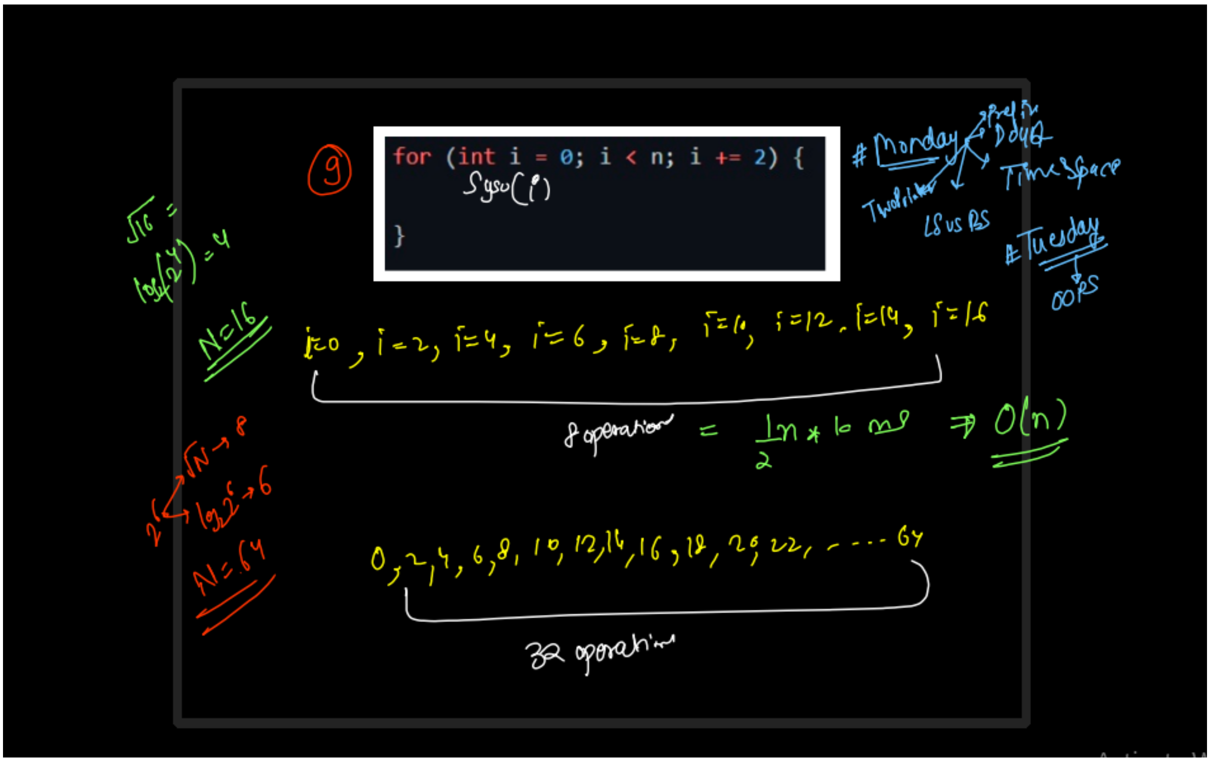
$O(\log_2 N)$


for (int i = 0; i * i < n; i++) {
}
O(√N)
i*i
1, 4, 9, 16, 25, 36, 49, ...
6
N = 16
2^4
i = 0 ⇒ 0*0 < 16
i = 1 ⇒ 1*1 < 16
i = 2 ⇒ 2*2 < 16
i = 3 ⇒ 3*3 < 16
4 operations
N = 32
2^5
6 operations
i=0 0*0 < 32
i=1 1*1 < 32
i=2 2*2 < 32
i=3 3*3 < 32
i=4 4*4 < 32
i=5 5*5 < 32
N = 64
2^6
Operations
i=0 0*0 < 64
i=1 1*1 < 64
i=2 2*2 < 64
i=3 9 < 64
i=4 16 < 64
i=5 25 < 64
i=6 36 < 64
i=7 49 < 64
i=8 64 = 64

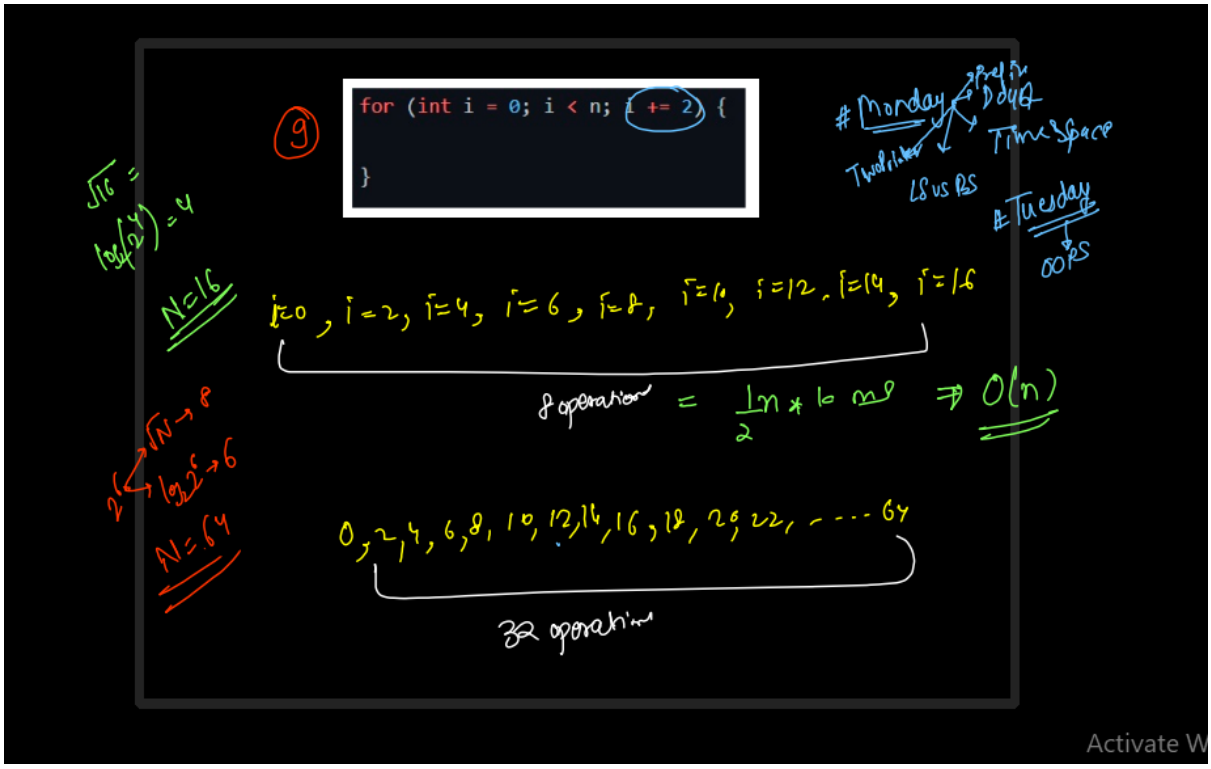

for (int i = 0; i < n; i++) {
Syso(i)
}
for (int j = 0; j < m; j++) {
Syso(j)
}
7
k+k+k+... n times
= n*k ⇒ O(n)
k+k+k+k+... m times
= m*k ⇒ O(m)
time → O(n+m)
Example
Merge 2 Sorted Arrays

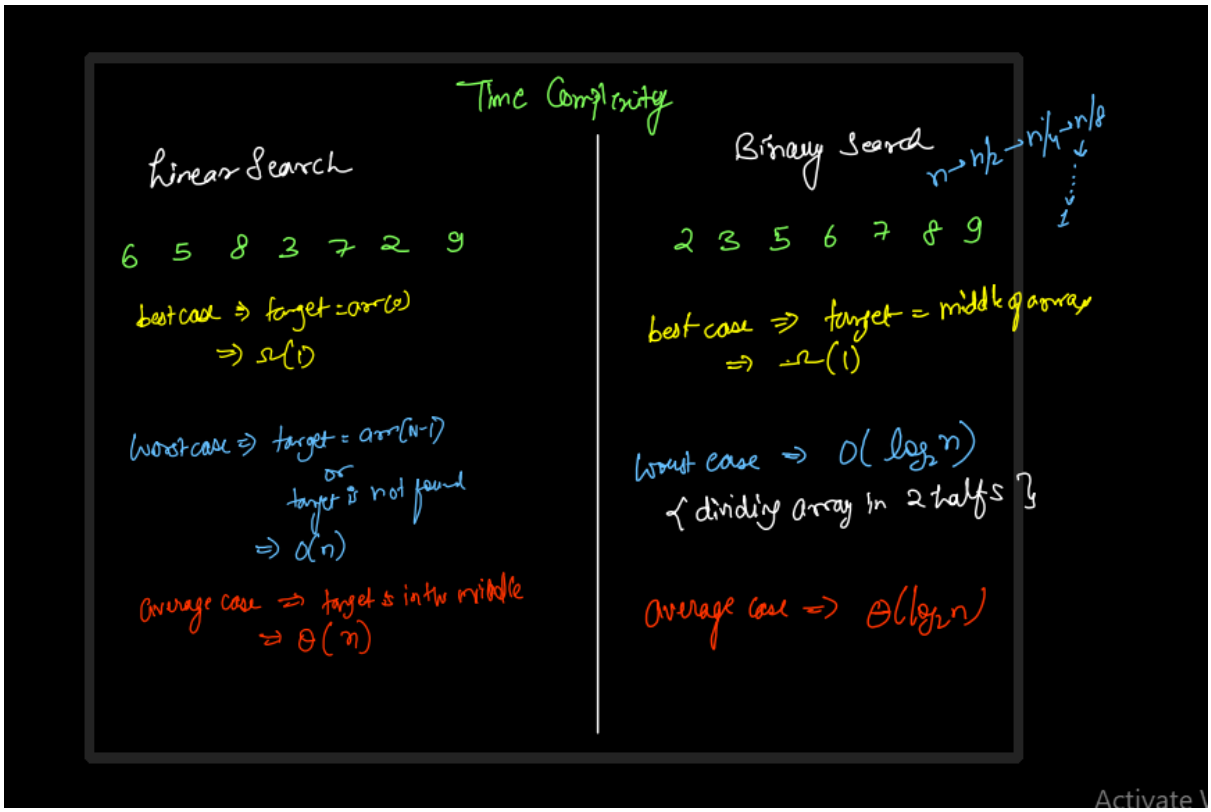


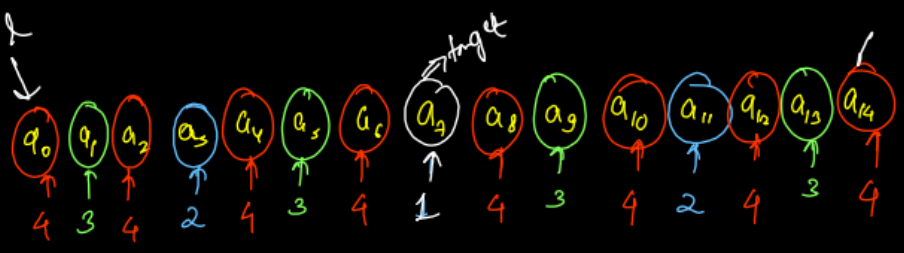
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
System.out.println(i + " " + j);
}
}
8
10
10 * k ms
10k + 10k + 10k + ... n times
= 10n k ms
⇒ O(n)



for (int i = 0; i < n; i += 2) {
Sysout(i)
}
9
N=16
i=0, i=2, i=4, i=6, i=8, i=10, i=12, i=14, i=16
8 operation
= 1/2 n * 1 ms ⇒ O(n)
N=64
0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, ... 64
32 operation
Monday
Tuesday
OOPS
Time Space







Best case → middle → $O(1)$

Worst case → 4 iterations → $O(\log_2 n)$

Average case $= \frac{4*7 + 3*4 + 2*2 + 1*1}{15}$ → 3 iterations

$\Rightarrow O(\log_2 n)$

Recurrence Relation

Linear Search

$n, n-1, n-2, n-3, \ldots 1$

N=7

6 5 8 3 7 2 9

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

Bigger Problem — Smaller Problem — Work done by you

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

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

$T(n-3) = T(n-4) + O(1)$

$T(1) = T(0) + O(1)$

$T(n) = T(0) + n * O(1)$

$T(0)$ = empty array

$\Rightarrow O(n)$

Binary search

$n, \frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \ldots 1$

2 3 5 6 7 8 9

$T(n) = T(n/2) + O(1)$

Bigger Problem — Smaller Problem — Work done by you

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

$T(n/4) = T(n/8) + O(1)$

$T(n/8) = T(n/16) + O(1)$

$T(1) = T(0) + O(1)$

$T(n) = T(0) + \log_2 n * O(1)$

$T(0)$ = empty array

$\Rightarrow O(\log_2 N)$

Recurrence Relations

Mock interview

O(n) ① $T(n) = T(n-1) + O(1)$ Linear Search

$O(n^2)$ ② $T(n) = T(n-1) + O(n)$ Bubble/Insertion Select

$O(\log n)$ ③ $T(n) = T(n/2) + O(1)$ Binary Search

O(n) ④ $T(n) = T(n/2) + O(n)$ Quick Select

$O(2^n)$ ⑤ $T(n) = 2T(n-1) + O(1)$ Tower of Hanoi, Subsets

O(n) ⑥ $T(n) = 2T(n/2) + O(1)$ ↳ Divide & Conquer

$O(n \log n)$ ⑦ $T(n) = 2T(n/2) + O(n)$ ↳ merge sort

② $T(n) = T(n-1) + O(n)$

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

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

⋮

$T(1) = T(0) + O(n)$

$T(n) = T(0) + n * O(n)$

$\Rightarrow O(n^2)$

④ $T(n) = T(n/2) + O(n)$

$T(n/2) = T(n/4) + O(n/2)$

$T(n/4) = T(n/8) + O(n/4)$

$T(n/8) = T(n/16) + O(n/8)$

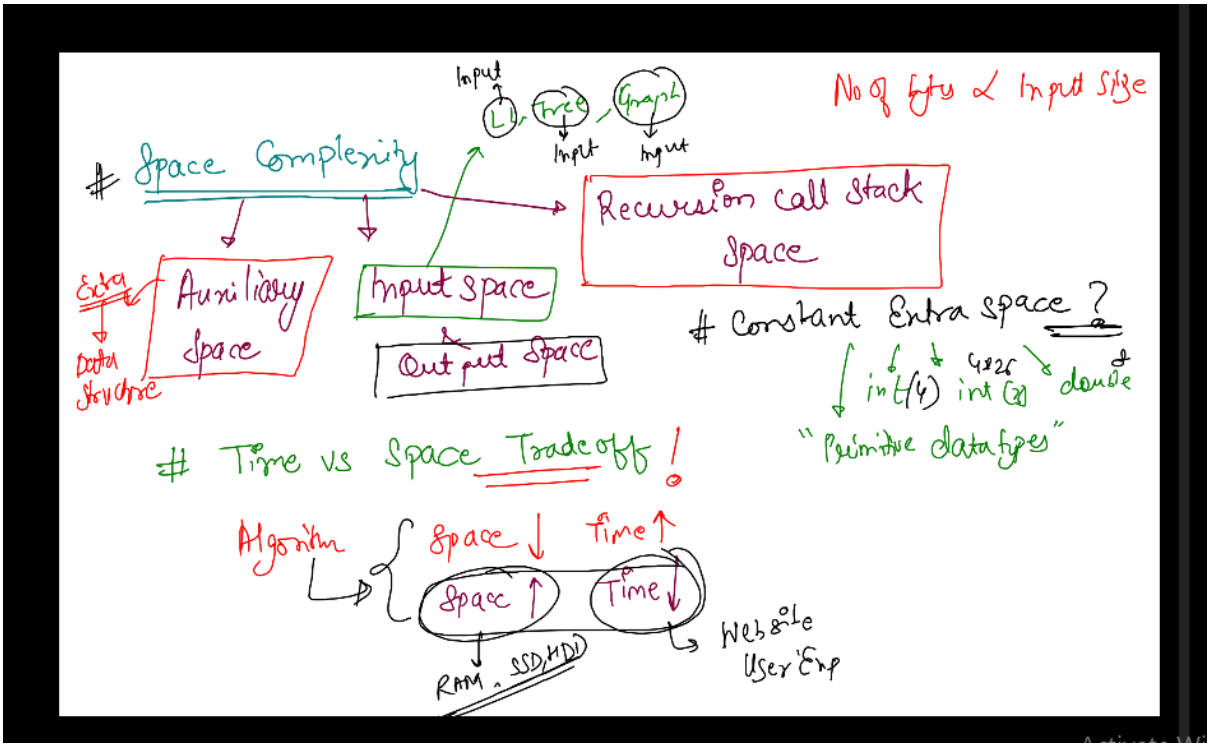
⋮

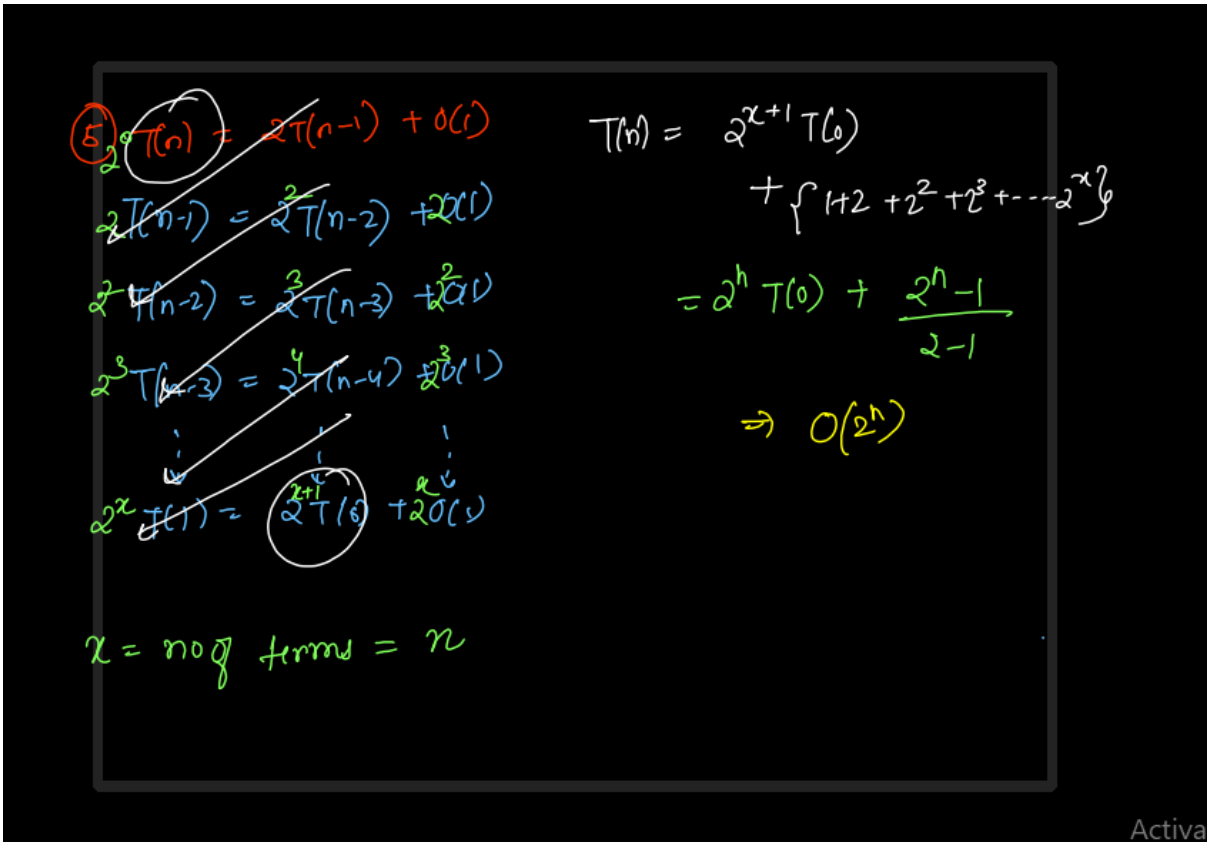
$T(1) = T(0) + O(1)$

$T(n) = T(0) + \{n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots 1\}$

$= T(0) + n\{1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots 0\}$

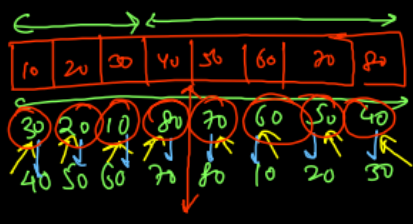
$= T(0) + 2*n$ $O(n)$


No of bytes ∝ Input Size
Space Complexity
Input
LL
Tree
Graph
Input
Input
Auxiliary Space
Extra
Data Structure
Input Space
Output Space
Recursion call stack Space
Constant Extra space ?
int i/4 int a/4 4 bytes double 8
"Primitive datatypes"
Time vs Space Tradeoff !
Algorithm
Space ↓ Time ↑
Space ↑
Time ↓
RAM, SSD, HDD
Web & Mobile User Exp


(5) T(n) = 2T(n-1) + O(1)
2T(n-1) = 2²T(n-2) + 2O(1)
2²T(n-2) = 2³T(n-3) + 2²O(1)
2³T(n-3) = 2⁴T(n-4) + 2³O(1)
2^x T(1) = 2^(x+1) T(0) + 2^x O(1)
x = no of terms = n
T(n) = 2^(x+1) T(0) + {1+2+2²+2³+---2^x}
= 2^n T(0) + (2^n - 1)/(2-1)
⇒ O(2^n)

Rotate Array by k positions

```
void reverse(int[] nums, int start, int end)
{
    while(start < end)
    {
        int temp = nums[start];
        nums[start] = nums[end];
        nums[end] = temp;

        start++;
        end--;
    }
}

public void rotate(int[] nums, int k) {
    int n = nums.length;
    k = k % n;

    reverse(nums, 0, n-1);
    reverse(nums, 0, k-1);
    reverse(nums, k, n-1);
}
```



Reversal

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

$\hookrightarrow O(n)$

Time → $O(n) + O(k) + O(n-k)$

$= O(n+n) = O(2n)$

$= O(n)$

Rotate Image

```
int row = mat.length, col = mat[0].length;

// Step 1: Transpose the matrix
for(int i=0; i<row; i++){
    for(int j=0; j<i; j++){
        int temp = mat[i][j];
        mat[i][j] = mat[j][i];
        mat[j][i] = temp;
    }
}
```

(n^2) + $i=0$ $j=$ $0+1+2+\cdots(n-1)$

$i=1$ $j=0$ $= \frac{n \times n-1}{2}$

$i=2$ $j=0,1$ $\Rightarrow O(n^2)$

$i=n$ $j=0,1,2,\ldots n-1$

```
// Step 2: Row by Row Reversal
for(int i=0; i<row; i++){
    int left = 0, right = col - 1;
    while(left < right){
        int temp = mat[i][left];
        mat[i][left] = mat[i][right];
        mat[i][right] = temp;

        left++; right--;
    }
}
```

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

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

String → Array of character
↳ O(no of characters)

Input: String str = scn.nextLine(); → O(length)

Output: System.out.println(str); → O(length)

```
for(int i=0; i<str.length(); i++){
    System.out.print(str.charAt(i));
}
```

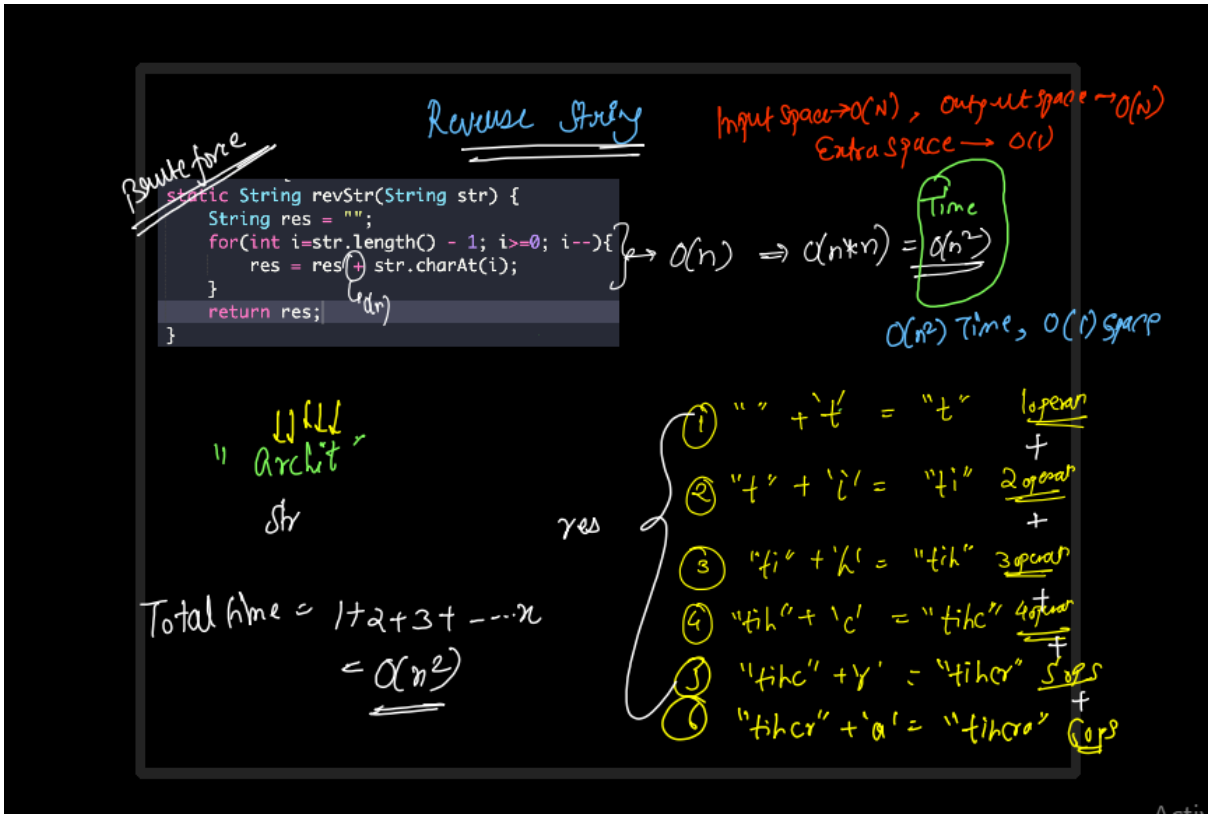
Check if String is Palindrome

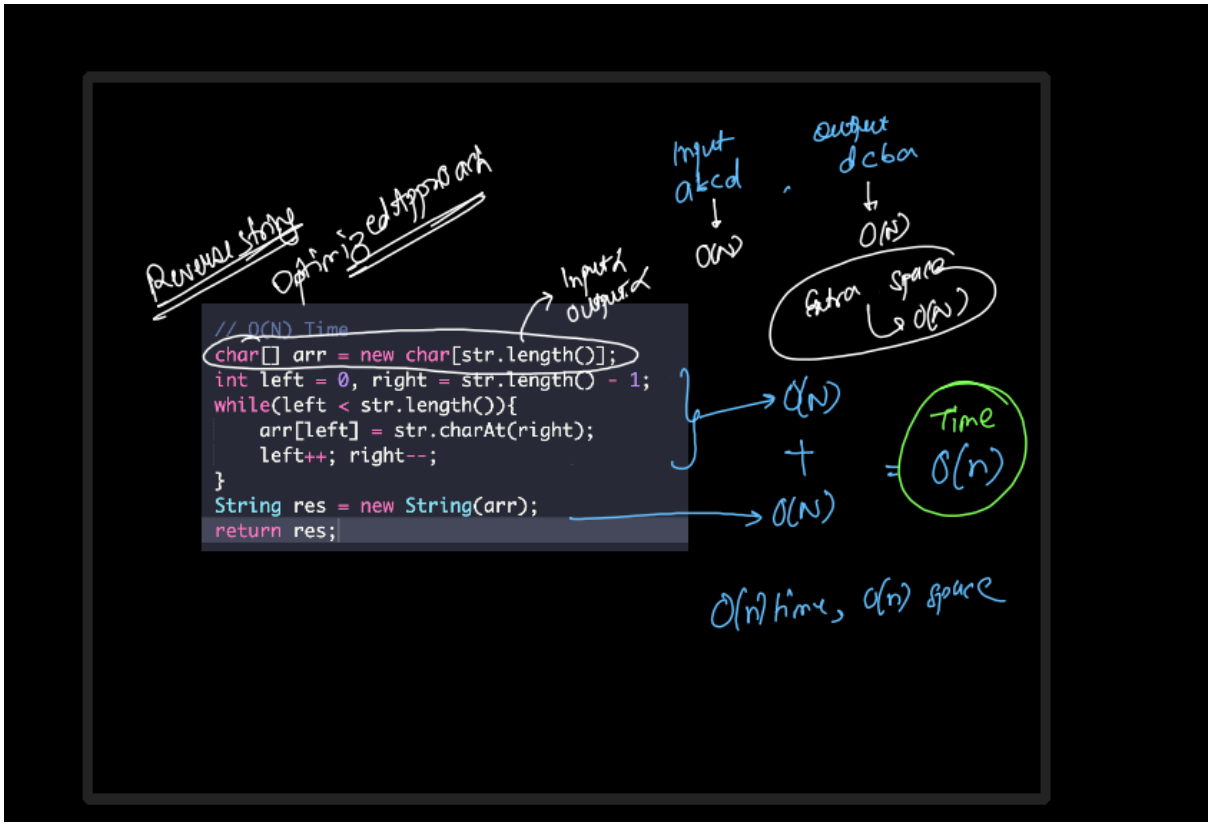
```
public boolean isPalindrome(String s) {
    int left = 0, right = s.length() - 1;
    while(left <= right){
        char chl = s.charAt(left);
        char chr = s.charAt(right);

        if(chl != chr){
            return false;
        }

        left++; right--;
    }

    return true;
}
```

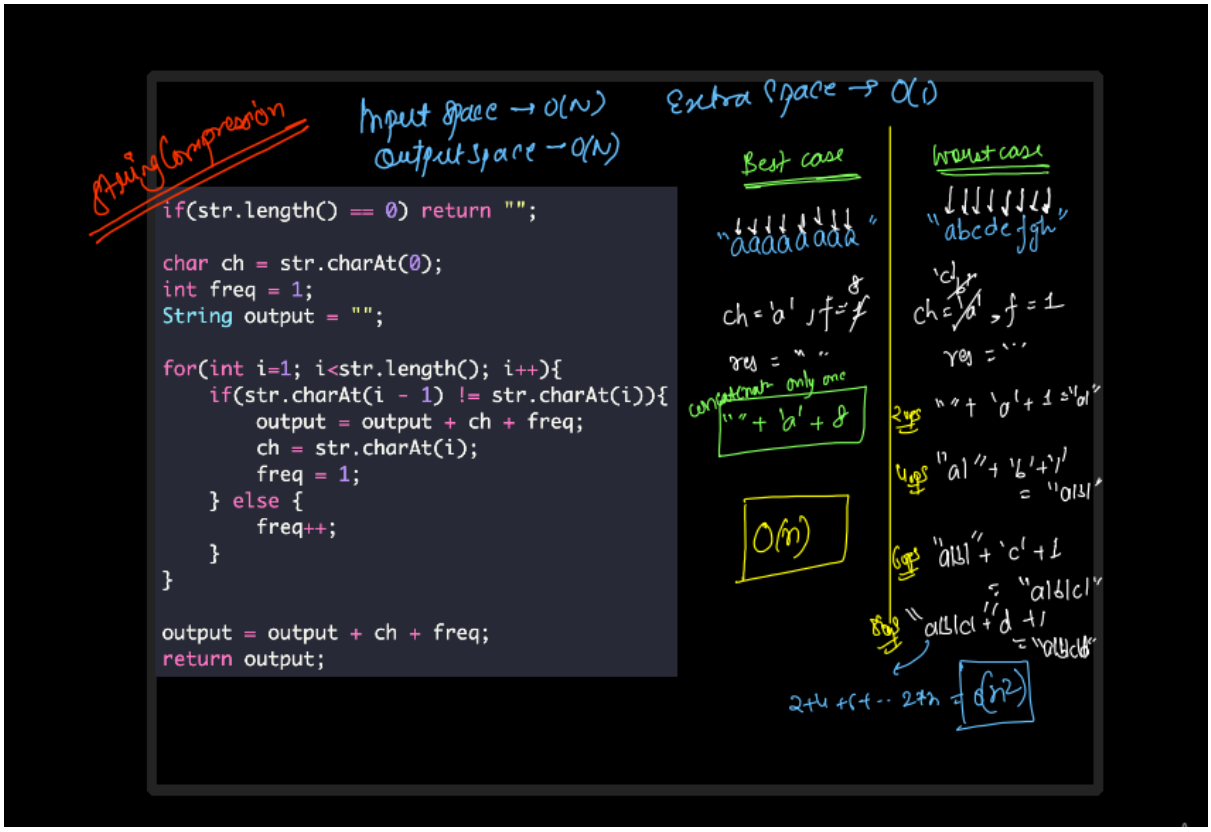
$T(n) = T(n-2) + O(1)$

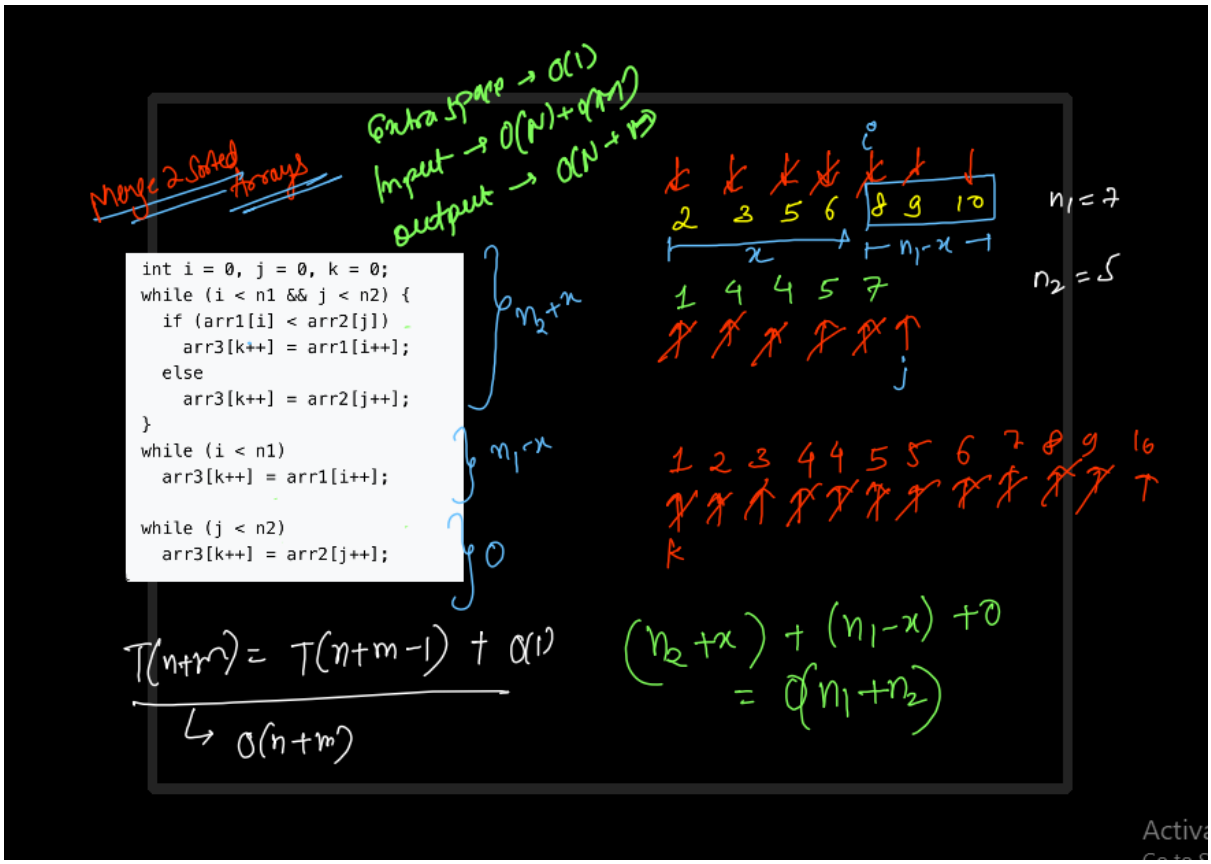
$\Rightarrow O(n)$

n = string's length


Reverse String
Input space → O(N), output space → O(N)
Extra space → O(1)
Bruteforce
static String revStr(String str) {
String res = "";
for(int i=str.length() - 1; i>=0; i--){
res = res + str.charAt(i);
}
return res;
}
O(n) ⇒ O(n*n) = O(n²)
Time
O(n²) Time, O(1) Space
"Hello Archit"
str
res
Total time = 1+2+3+ ....n
< O(n²)
① "" + 't' = "t" 1 operation
② "t" + 'i' = "ti" 2 operations
③ "ti" + 'h' = "tih" 3 operations
④ "tih" + 'c' = "tihc" 4 operations
⑤ "tihc" + 'r' = "tihcr" 5 ops
⑥ "tihcr" + 'a' = "tihcra" 6 ops


Reversal string
Optimized approach
// O(N) Time
char[] arr = new char[str.length()];
int left = 0, right = str.length() - 1;
while(left < str.length()){
arr[left] = str.charAt(right);
left++; right--;
}
String res = new String(arr);
return res;
Input & output
Input abcd
O(N)
output dcba
O(N)
Extra space
O(N)
O(N) + O(N) = Time O(n)
O(n) time, O(n) space

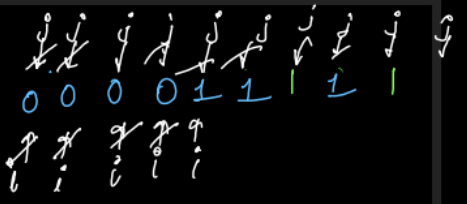

String Compression
Input space → O(N)
Output space → O(N)
Extra Space → O(1)
Best case
Worst case
"aaaaaaaaa"
"abcdefgh"
if(str.length() == 0) return "";
char ch = str.charAt(0);
int freq = 1;
String output = "";
for(int i=1; i<str.length(); i++){
if(str.charAt(i - 1) != str.charAt(i)){
output = output + ch + freq;
ch = str.charAt(i);
freq = 1;
} else {
freq++;
}
}
output = output + ch + freq;
return output;
O(n)
2+4+6+...2n = O(n^2)


Merge 2 Sorted Arrays
Extra space → O(1)
Input → O(N)+O(M)
Output → O(N+M)
int i = 0, j = 0, k = 0;
while (i < n1 && j < n2) {
if (arr1[i] < arr2[j])
arr3[k++] = arr1[i++];
else
arr3[k++] = arr2[j++];
}
while (i < n1)
arr3[k++] = arr1[i++];
while (j < n2)
arr3[k++] = arr2[j++];
n2+x
n1-x
0
T(n+m) = T(n+m-1) + O(1)
O(n+m)
(n2+x) + (n1-x) + 0
= O(n1+n2)
n1 = 7
n2 = 5

Sort 01

0s → left
1s → right

```
int i = 0;
for(int j=0; j<arr.length; j++){
    if(arr[j] == 0){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
        i++;
    }
}
return arr;
```



Best case, Average Case, Worst case → O(N)

Input → Binary Array → O(N)
Output → Binary Array → O(N)
Extra space → O(1) (i, j)

Equilibrium Point or Pivot Index

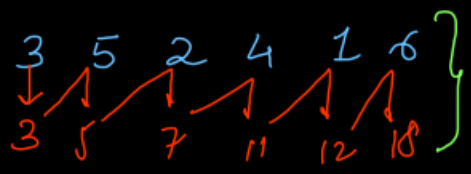
```
public int pivotIndex(int[] nums) {
    int[] pref = new int[nums.length];
    pref[0] = nums[0];
    for(int i=1; i<nums.length; i++){
        pref[i] = pref[i - 1] + nums[i];
    }

    for(int i=0; i<nums.length; i++){
        int leftSum = (i == 0) ? 0 : pref[i - 1];
        int rightSum = (i == nums.length - 1) ? 0 : (pref[nums.length - 1] - pref[i]);

        if(leftSum == rightSum) return i;
    }

    return -1;
}
```

Prefix Array → O(N)

→ O(N)

Input → 1D Array → O(N)
Output → Pivot Index → O(1)

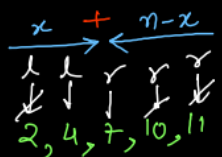
Extra Space → O(N) (Prefix Array)



Prefix Sum → O(N)

Two Pointer → Two Sum

```
int start = 0, end = nums.length - 1;
while(start < end){
    if(nums[start] + nums[end] == target){
        res[0] = start + 1; res[1] = end + 1;
        return res;
    } else if(nums[start] + nums[end] < target){
        start++;
    } else {
        end--;
    }
}
```



target ≠ 11

13 > 11
12 > 11
9 < 11
11 = 11

Total time = x + n - x

⇒ O(N)

Two Pointers ⇒ Dependent on each other.

Input Space → O(N)
Output space → 8byte → Pair → O(1)
Extra Space → 8byte → [L, R] → O(1)

Space Complexity

Input Space → Data Structures passed as parameters to the algorithm

Output Space → Data Structures returned as a output of an algorithm

Extra space or auxiliary space → Space which you are taking to design algorithm

O(1) Recursion calls Stack Space → Space taken by functions

1D array → O(N)
2D array → O(n*m)
String → O(x)

Space Complexity → Rated growth of bytes wrt to the input size

Datatype

float/int → 4 byte
long/double → 8byte
Char → 2 byte
boolean → 1 byte

primitive variables → O(1) Constant space

Process Memory Layout

Runtime vs Extra space Tradeoff

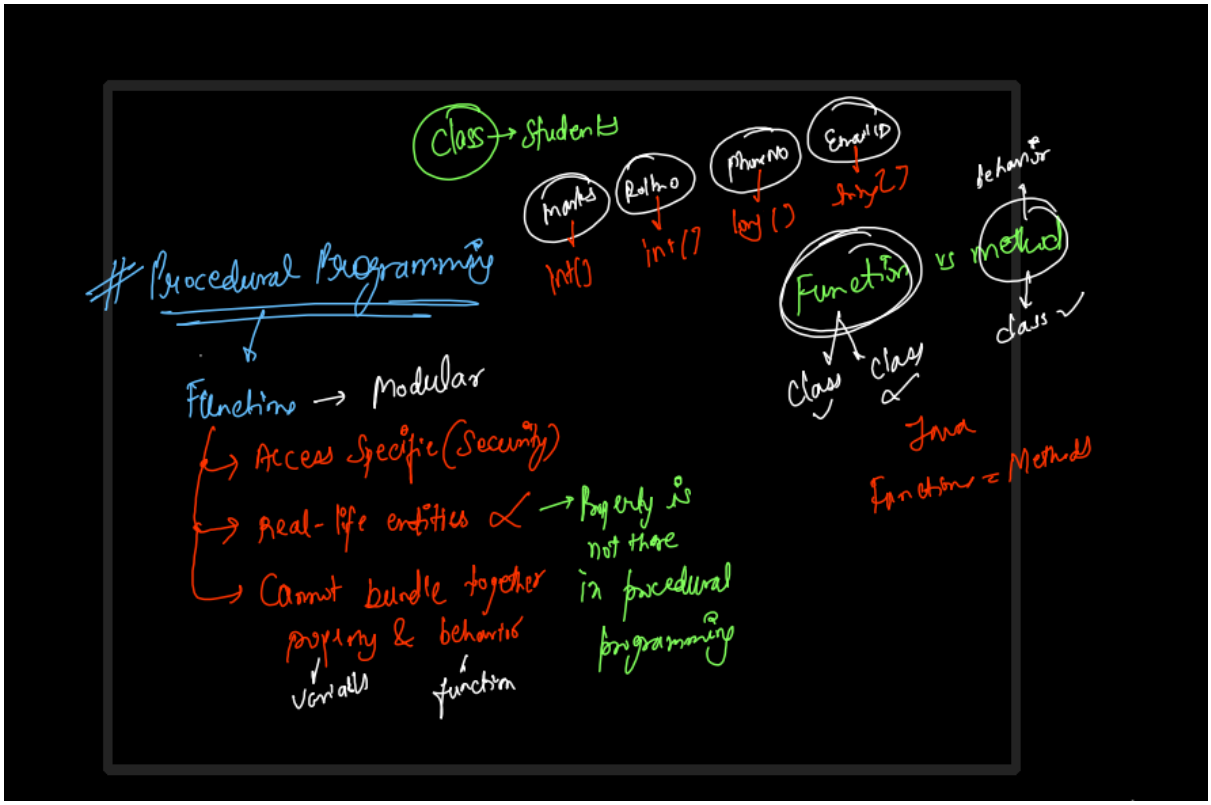
Time vs Space Complexity Tradeoff

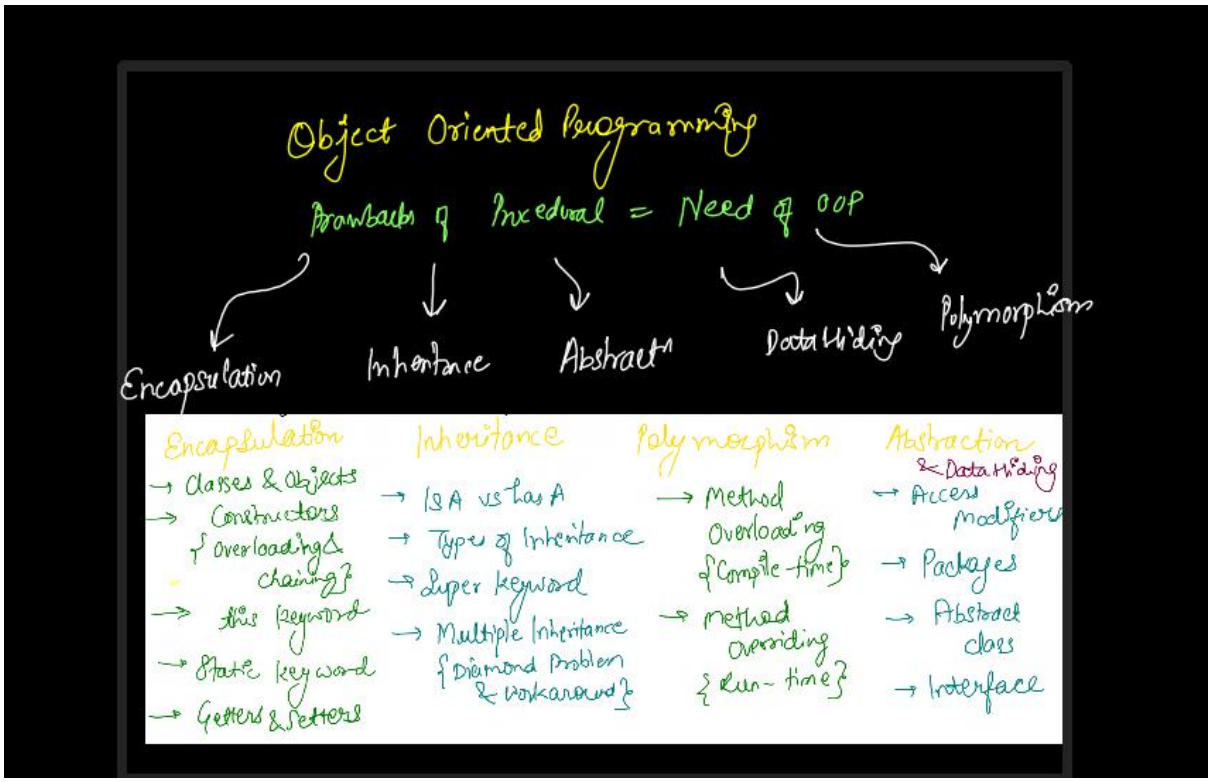
Bruteforce (1) Time ↑ Space ↓

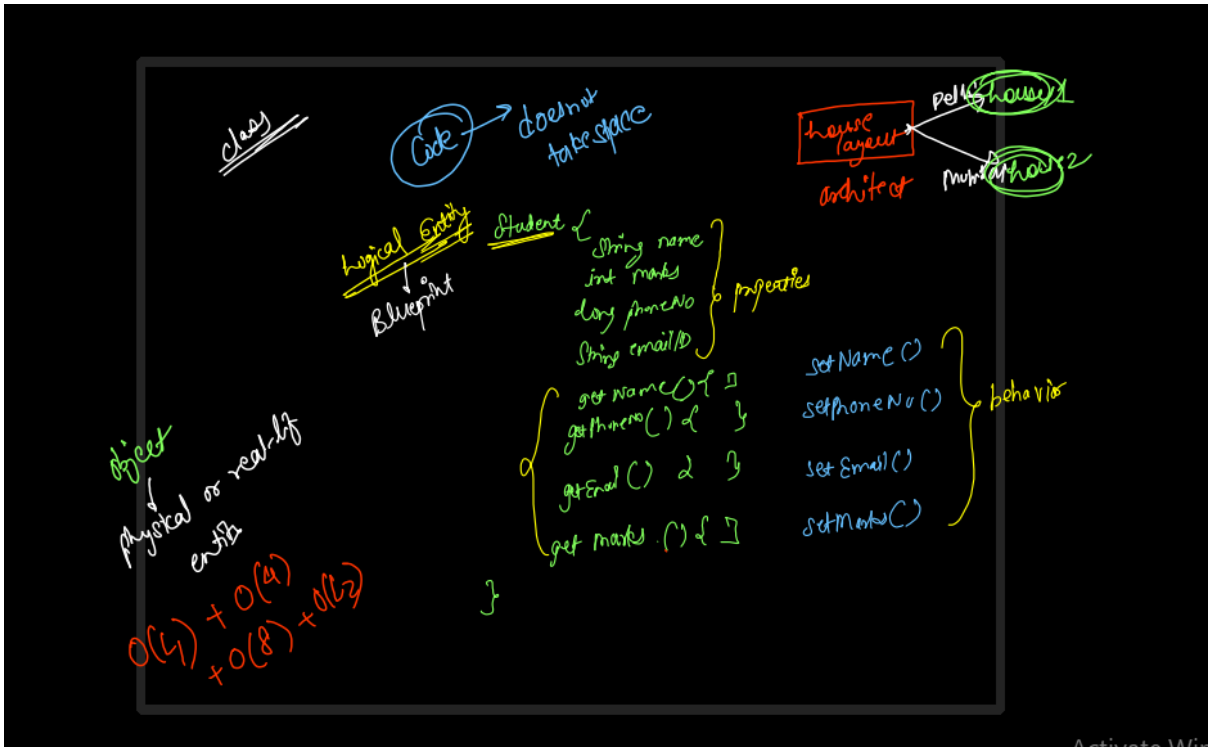
Optimized Approach (2) Time ↓ Space ↑

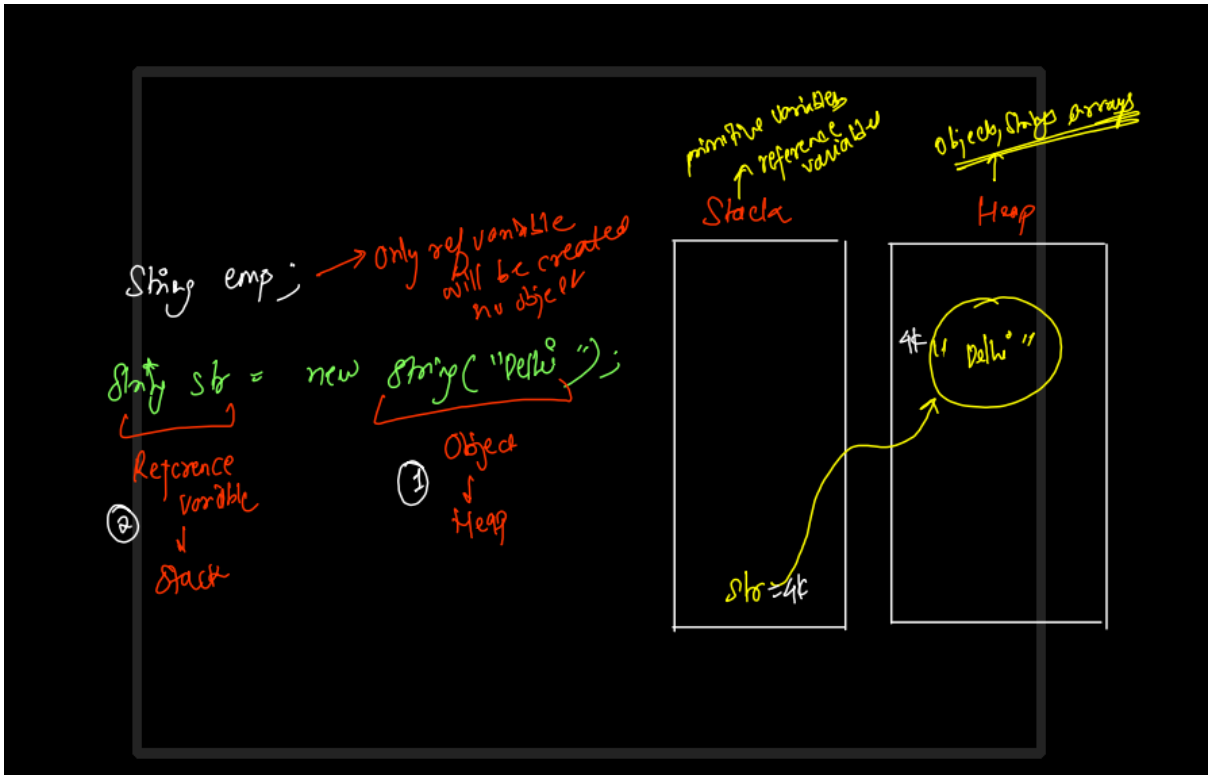
Time → Expensive

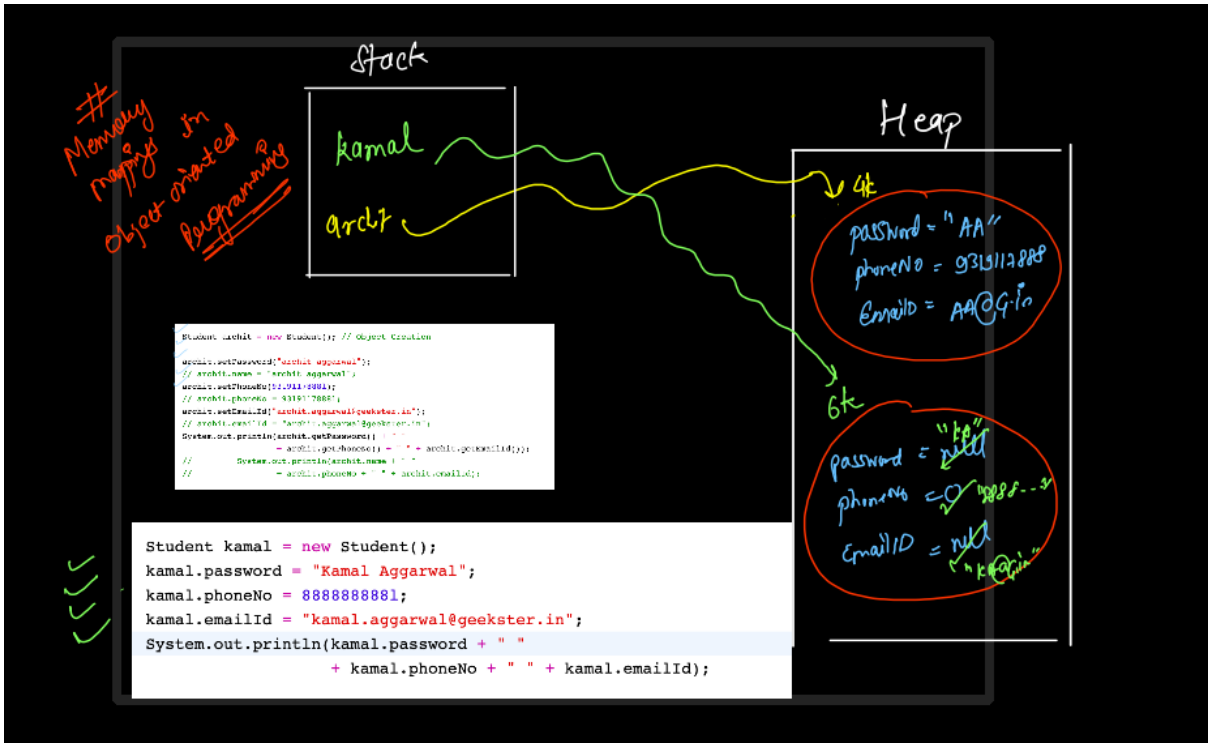
Space → Cheap

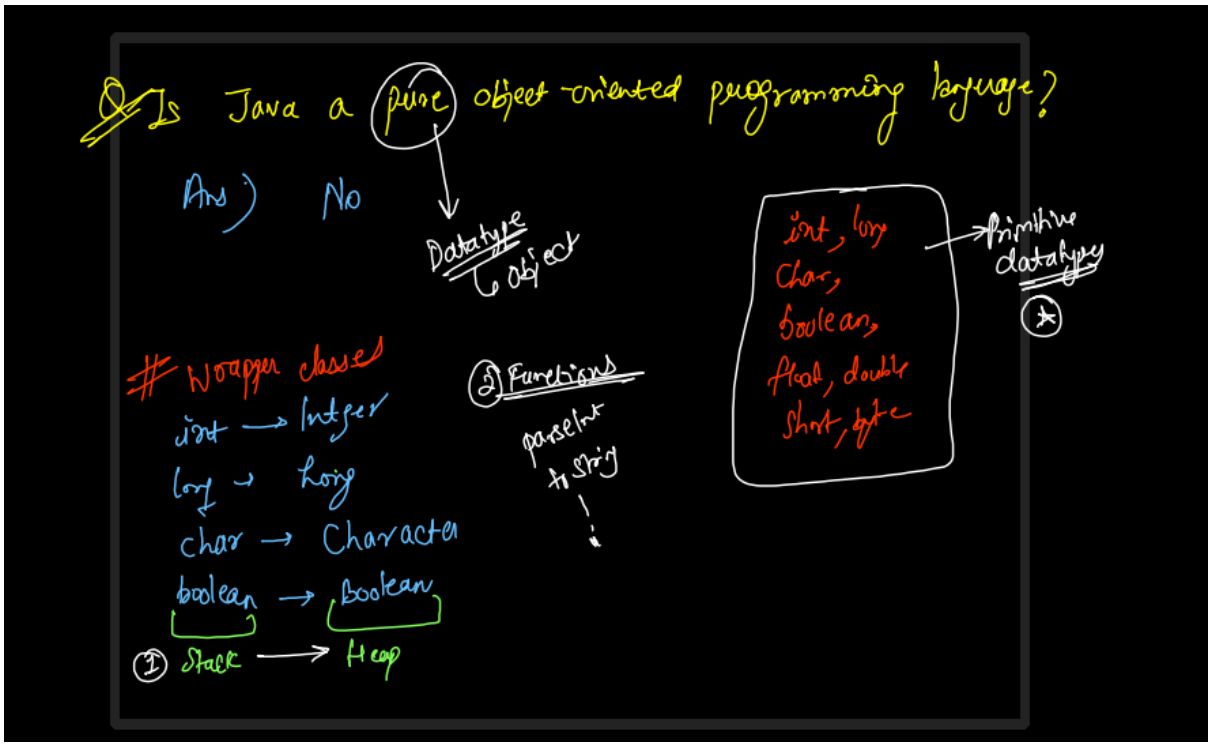

Class → Student
Marks
Roll no
Phone no
Email ID
int[]
int[]
long[]
String[]
behavior
Procedural Programming
Function
vs
method
class
class
class
Function → Modular
Access Specific (Security)
Real-life entities ∝ → Property is not there in procedural programming
Cannot bundle together property & behavior
variables
function
Java
Function = Methods


Object Oriented Programming
Drawbacks of Procedural = Need of OOP
Encapsulation
Inheritance
Abstraction
Data Hiding
Polymorphism
Encapsulation
→ Classes & Objects
→ Constructors {overloading & chaining}
→ this keyword
→ Static keyword
→ Getters & Setters
Inheritance
→ IS A vs Has A
→ Type of Inheritance
→ Super keyword
→ Multiple Inheritance {Diamond Problem & workaround}
Polymorphism
→ Method Overloading {Compile-time}
→ method overriding {Run-time}
Abstraction & Data Hiding
→ Access modifiers
→ Packages
→ Abstract class
→ Interface


class
Cat
doesn't take space
house layout
architect
Pehla
Dusra
Logical Entity
Blueprint
Student {
String name
int marks
long phoneNo
String emailID
properties
getName(){ }
getPhoneNo(){ }
getEmail(){ }
getMarks(){ }
setName()
setPhoneNo()
setEmail()
setMarks()
behavior
}
Object
physical or real life entity
O(1) + O(4) + O(8) + O(12)


String emp; → Only ref variable will be created no object
String sb = new String("Delhi");
Reference variable
Stack
Object
Heap
primitive variables
reference variables
Stack
Objects, Strings, arrays
Heap
4k
"Delhi"
sb = 4k


Stack
kamal
archit
Heap
4k
password = "AA"
phoneNo = 9315112888
EmailID = AA@G.in
6k
password = "kkk"
EmailID = null
Memory mapping in Object oriented programming
Student kamal = new Student();
kamal.password = "Kamal Aggarwal";
kamal.phoneNo = 8888888881;
kamal.emailId = "kamal.aggarwal@geekster.in";
System.out.println(kamal.password + " "
+ kamal.phoneNo + " " + kamal.emailId);


Is Java a pure object oriented programming language?
Ans) No
Datatype
Object
int, long
Char,
boolean,
float, double
Short, byte
Primitive datatypes
Wrapper classes
int → Integer
long → long
char → Character
boolean → Boolean
Stack → Heap
Functions
parseInt
toString

Reference variables are getting swapped not the objects

```
public static void swap(Student s1, Student s2){
    Student temp = s1;
    s1 = s2;
    s2 = temp;
}
```

```
Student archit = new Student();
archit.password = "archit aggarwal";
archit.phoneNo = 9319117888l;
archit.emailId = "archit.aggarwal@geekster.in";
System.out.println(archit.password + " "
                   + archit.phoneNo + " " + archit.emailId);

Student kamal = new Student();
kamal.password = "Kamal Aggarwal";
kamal.phoneNo = 8888888881;
kamal.emailId = "kamal.aggarwal@geekster.in";
System.out.println(kamal.password + " "
                   + kamal.phoneNo + " " + kamal.emailId);

swap(archit, kamal);

System.out.println(archit.password + " "
                   + archit.phoneNo + " " + archit.emailId);

System.out.println(kamal.password + " "
                   + kamal.phoneNo + " " + kamal.emailId);
```

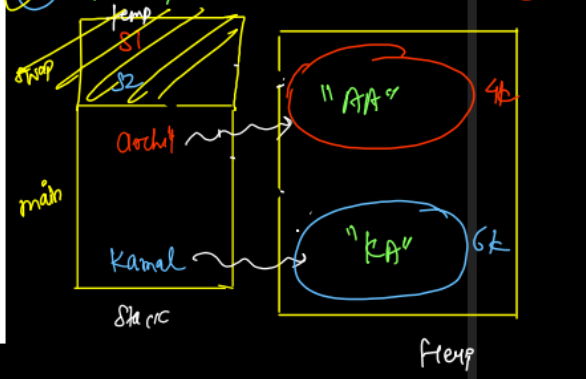
(1) (2) (3) (4)

A. Archit, Kamal, Kamal, Archit (Incorr. Swap)

B. Archit, Kamal, Archit, Archit

C. Archit, Kamal, Kamal, Kamal

D. Archit, Kamal, Archit, Kamal (Corr. Swap)



Reference variables (Stack) are getting updated/copied, then you are performing a shallow copy ☆

⇒ Objects are intact (They are not getting copied/updated)

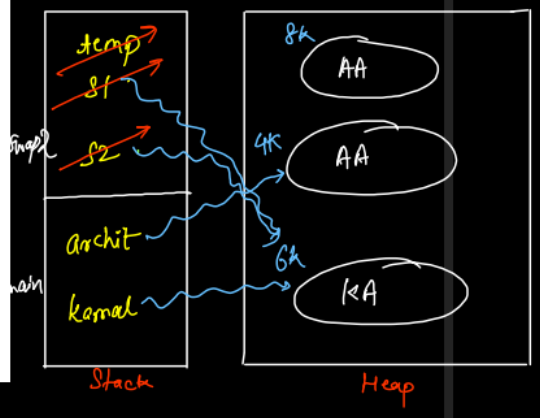
Swap 1

```
public static void swap(Student s1, Student s2){
    Student temp = s1;
    s1 = s2;
    s2 = temp;
}
```

```
public static void swap2(Student s1, Student s2){
    Student temp = new Student();
    temp.password = s1.password;
    temp.phoneNo = s1.phoneNo;
    temp.emailId = s1.emailId;

    s1 = s2;
    s2 = temp;
}
```

(A) Archit, Kamal, Kamal, Archit
(B) Archit, Kamal, Archit, Archit
(C) Archit, Kamal, Kamal, Kamal
(D) Archit, Kamal, Archit, Kamal

```
Student archit = new Student();
archit.password = "archit aggarwal";
archit.phoneNo = 93191178881;
archit.emailId = "archit.aggarwal@geekster.in";
System.out.println(archit.password + " "
                   + archit.phoneNo + " " + archit.emailId);

Student kamal = new Student();
kamal.password = "Kamal Aggarwal";
kamal.phoneNo = 8888888881;
kamal.emailId = "kamal.aggarwal@geekster.in";
System.out.println(kamal.password + " "
                   + kamal.phoneNo + " " + kamal.emailId);

swap2(archit, kamal);

System.out.println(archit.password + " "
                   + archit.phoneNo + " " + archit.emailId);

System.out.println(kamal.password + " "
                   + kamal.phoneNo + " " + kamal.emailId);
```

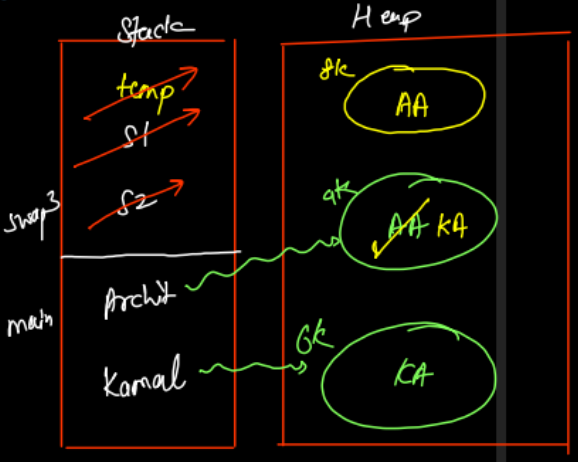



```
public static void swap3(Student s1, Student s2){
    Student temp = new Student();
    temp.password = s1.password;
    temp.phoneNo = s1.phoneNo;
    temp.emailId = s1.emailId;

    s1.password = s2.password;
    s1.phoneNo = s2.phoneNo;
    s1.emailId = s2.emailId;

    s2 = temp;
}
```

updating the object

(A) Archit, Kamal, Kamal, Archit
(B) Archit, Kamal, Archit, Archit
(C) Archit, Kamal, Kamal, Kamal
(D) Archit, Kamal, Archit, Kamal

```
Student archit = new Student();
archit.password = "archit aggarwal";
archit.phoneNo = 9319117888;
archit.emailId = "archit.aggarwal@geekster.in";
System.out.println(archit.password + " "
                   + archit.phoneNo + " " + archit.emailId);

Student kamal = new Student();
kamal.password = "Kamal Aggarwal";
kamal.phoneNo = 8888888881;
kamal.emailId = "kamal.aggarwal@geekster.in";
System.out.println(kamal.password + " "
                   + kamal.phoneNo + " " + kamal.emailId);

swap(archit, kamal);

System.out.println(archit.password + " "
                   + archit.phoneNo + " " + archit.emailId);

System.out.println(kamal.password + " "
                   + kamal.phoneNo + " " + kamal.emailId);
```



If the actual object is getting copied into a new object (and not just the reference variables), it is known as deep copy

```
public static void swap4(Student s1, Student s2){
    Student temp = new Student();
    temp.password = s1.password;
    temp.phoneNo = s1.phoneNo;
    temp.emailId = s1.emailId;

    s1.password = s2.password;
    s1.phoneNo = s2.phoneNo;
    s1.emailId = s2.emailId;

    s2.password = temp.password;
    s2.phoneNo = temp.phoneNo;
    s2.emailId = temp.emailId;
}
```

Initialization

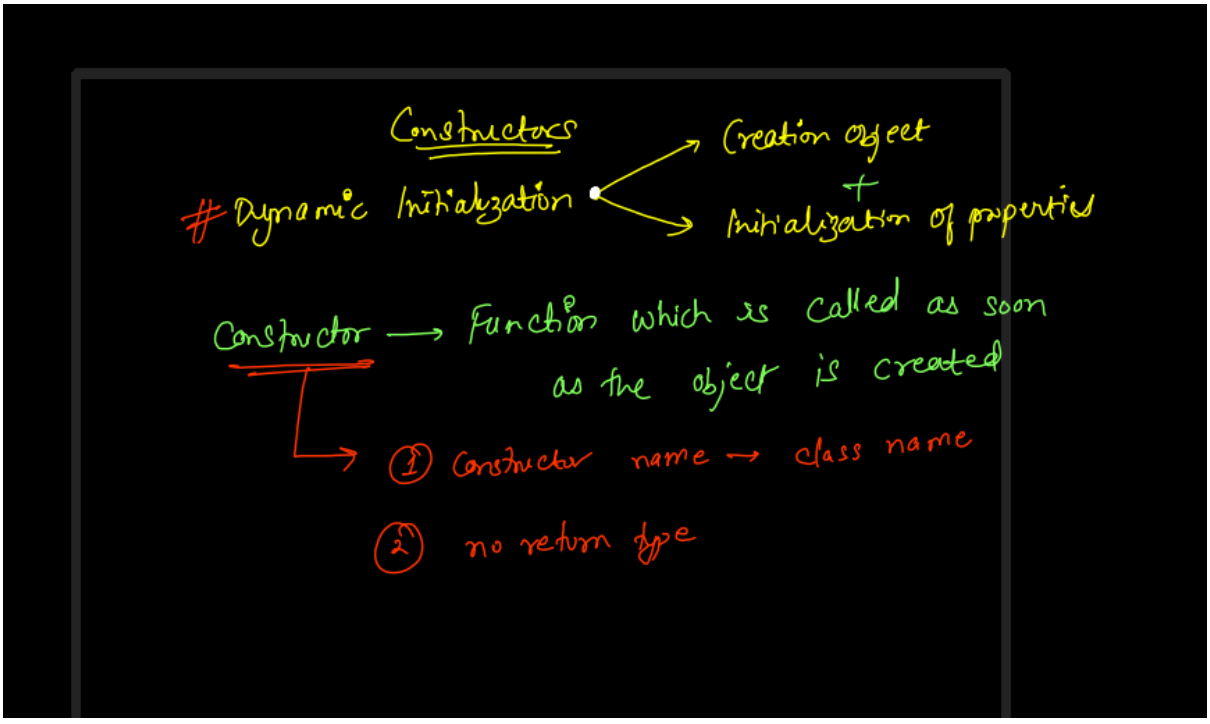
```
public void initialize(String newPassword, long newPhoneNo, String newEmailId){
    setPassword(newPassword);
    setPhoneNo(newPhoneNo);
    setEmailId(newEmailId);
}
```

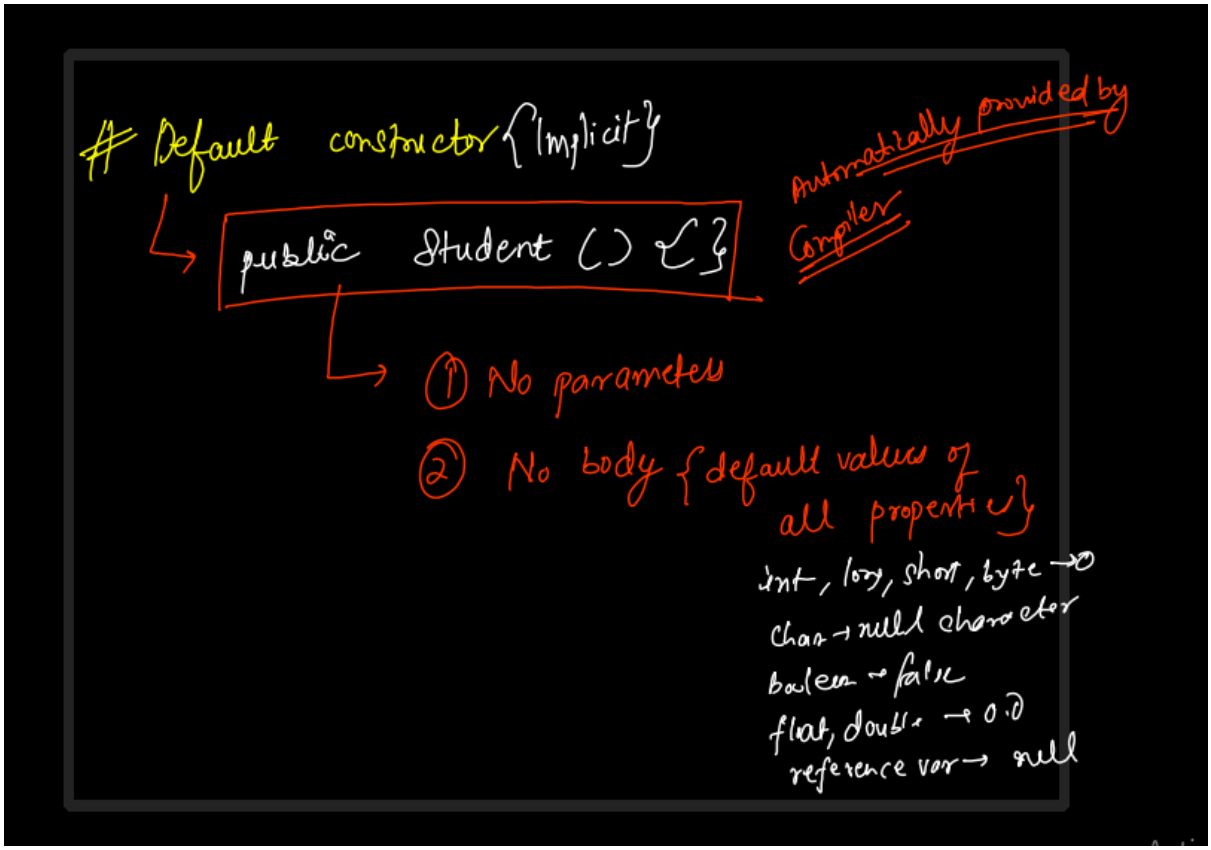
```
Student archit = new Student();
archit.initialize("archit aggarwal", 93191178881, "archit@geekster.in");

System.out.println(archit.password + " "
                   + archit.phoneNo + " " + archit.emailId);

Student kamal = new Student();
kamal.initialize("kamal aggarwal",8888888888881, "kamal@geekster.in");
System.out.println(kamal.password + " "
                   + kamal.phoneNo + " " + kamal.emailId);
```

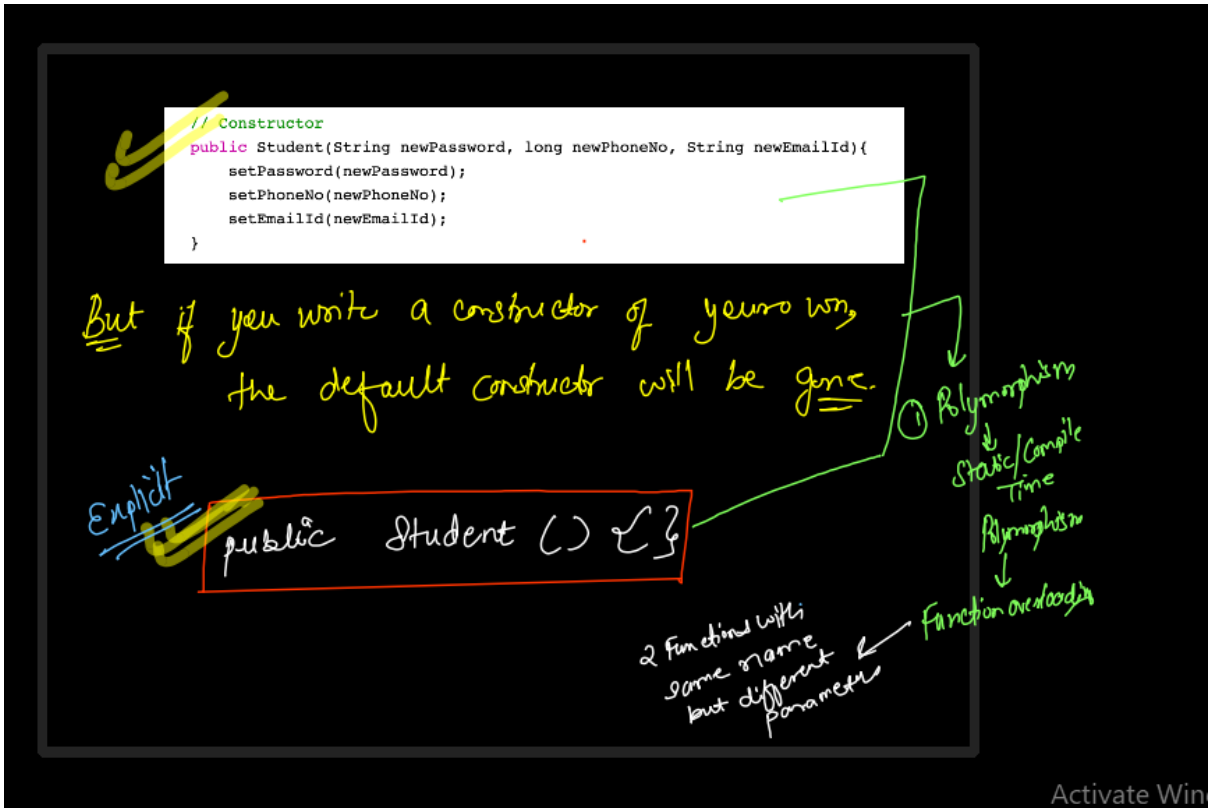
Dynamic Initialization

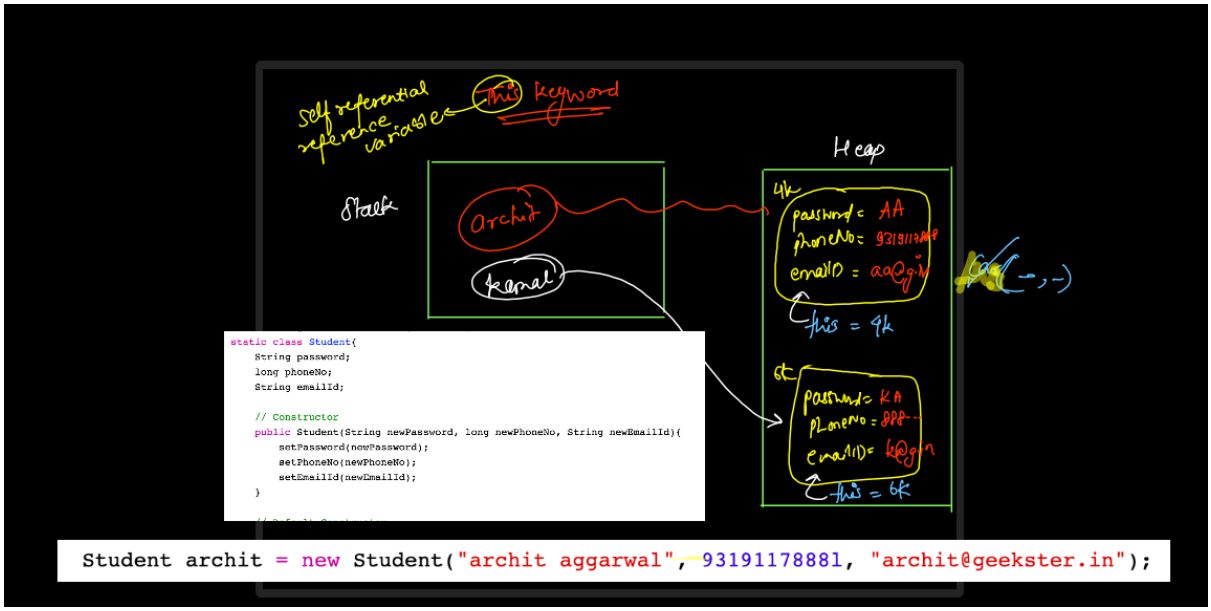
```
// Constructor
public Student(String newPassword, long newPhoneNo, String newEmailId){
    setPassword(newPassword);
    setPhoneNo(newPhoneNo);
    setEmailId(newEmailId);
}
```

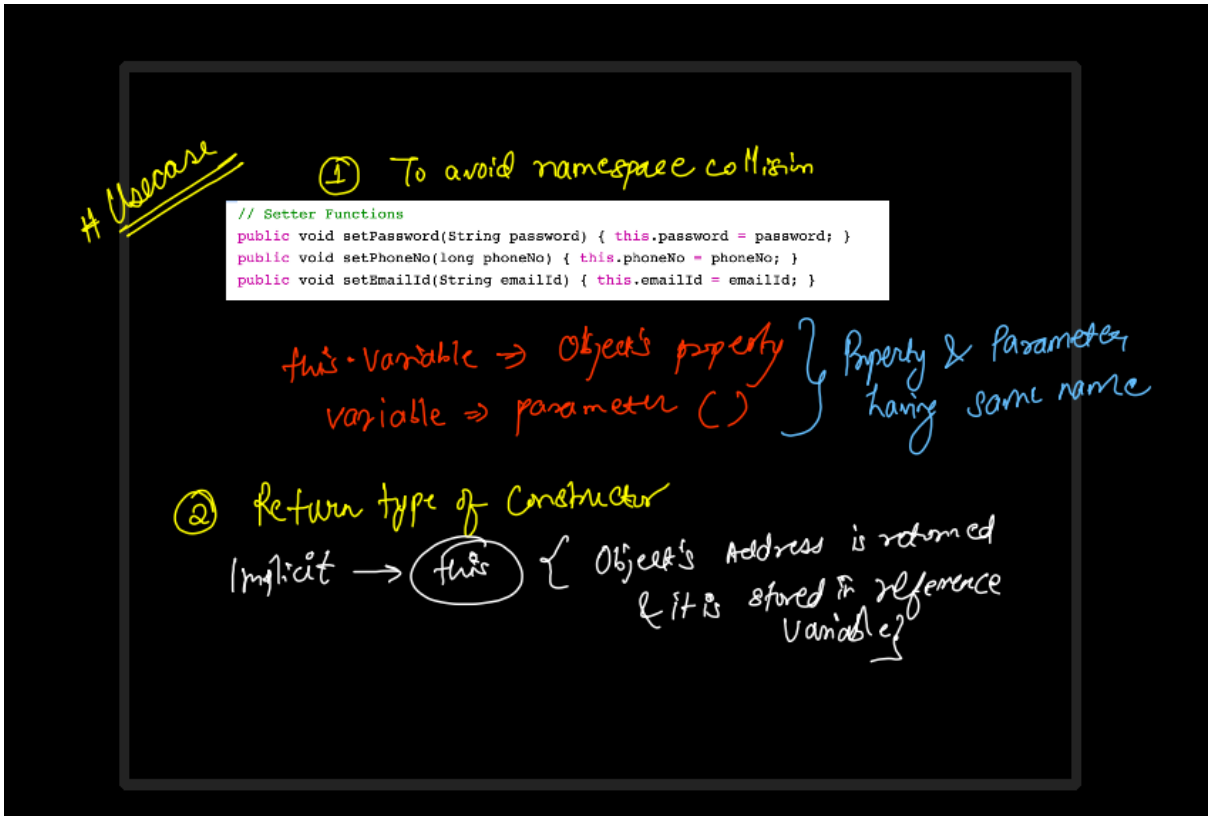
```
Student archit = new Student("archit aggarwal", 93191178881, "archit@geeksster.in");
// archit.initialize("archit aggarwal", 93191178881, "archit@geeksster.in");

System.out.println(archit.password + " "
                   + archit.phoneNo + " ".+ archit.emailId);

Student kamal = new Student("kamal aggarwal",8888888888881, "kamal@geeksster.in");
// kamal.initialize("kamal aggarwal",8888888888881, "kamal@geeksster.in");
System.out.println(kamal.password + " "
                   + kamal.phoneNo + " " + kamal.emailId);
```








Self referential reference variable
this keyword
Stack
Heap
Archit
Kamal
4k
password = AA
phoneNo = 9319117...
emailID = aa@g...
this = 4k
6k
password = KA
phoneNo = 999...
emailID = k@g...
this = 6k
static class Student{
String password;
long phoneNo;
String emailId;
// Constructor
public Student(String newPassword, long newPhoneNo, String newEmailId){
setPassword(newPassword);
setPhoneNo(newPhoneNo);
setEmailId(newEmailId);
}
Student archit = new Student("archit aggarwal", 93191178881, "archit@geekster.in");


Usecase
① To avoid namespace collision
// Setter Functions
public void setPassword(String password) { this.password = password; }
public void setPhoneNo(long phoneNo) { this.phoneNo = phoneNo; }
public void setEmailId(String emailId) { this.emailId = emailId; }
this.variable ⇒ Object's property
variable ⇒ parameter ( )
} Property & Parameter having same name
② Return type of Constructor
Implicit → this { Object's Address is returned & it is stored in reference Variable

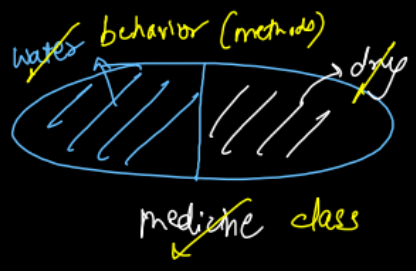
③ Constructor chaining

→ One constructor calls another constructor

```
static class Student{
    String password;
    long phoneNo;
    String emailId;

    // Constructor
    public Student(String newPassword, long newPhoneNo, String newEmailId){
        setPassword(newPassword);
        setPhoneNo(newPhoneNo);
        setEmailId(newEmailId);
    }

    // Default Constructor
    public Student(){
        this("admin", 88888888881, "admin@gmail.com");
```

① Encapsulation

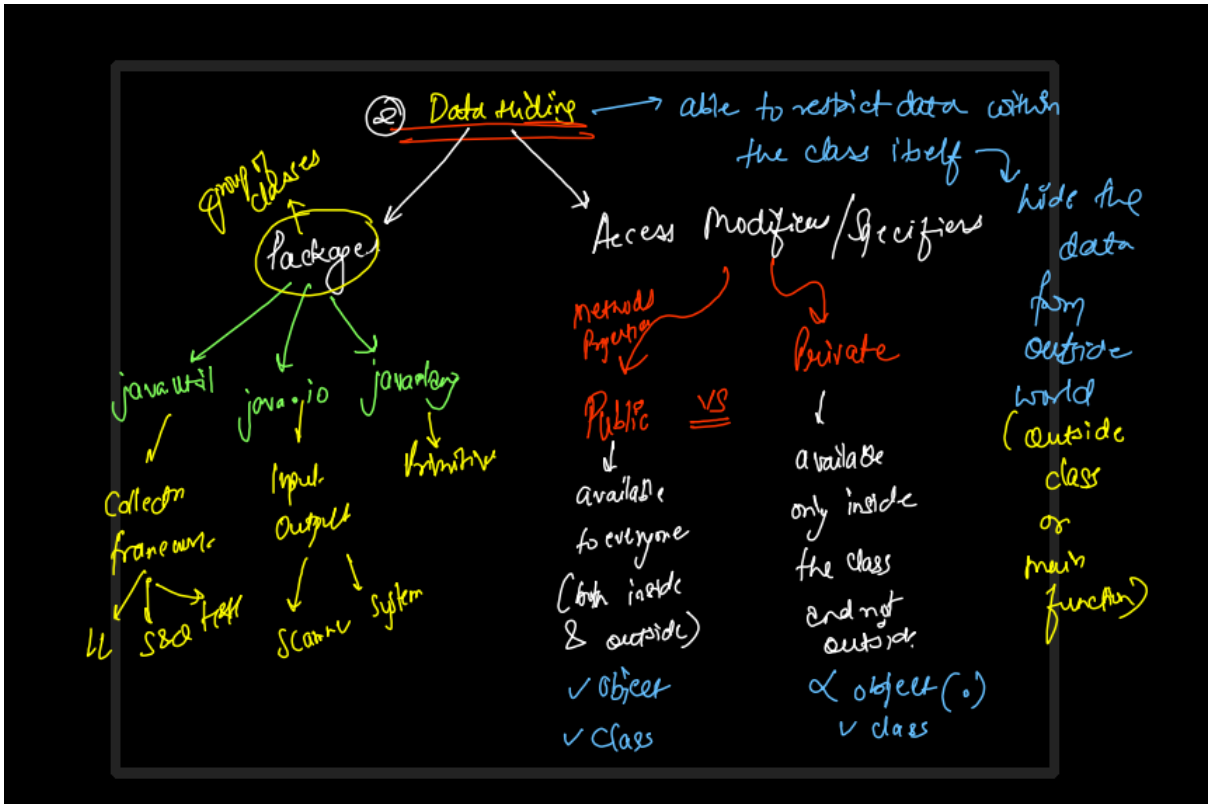
" Wrapping up data members (properties) with the behaviors (methods) into a single entity (class), this is known as encapsulation.

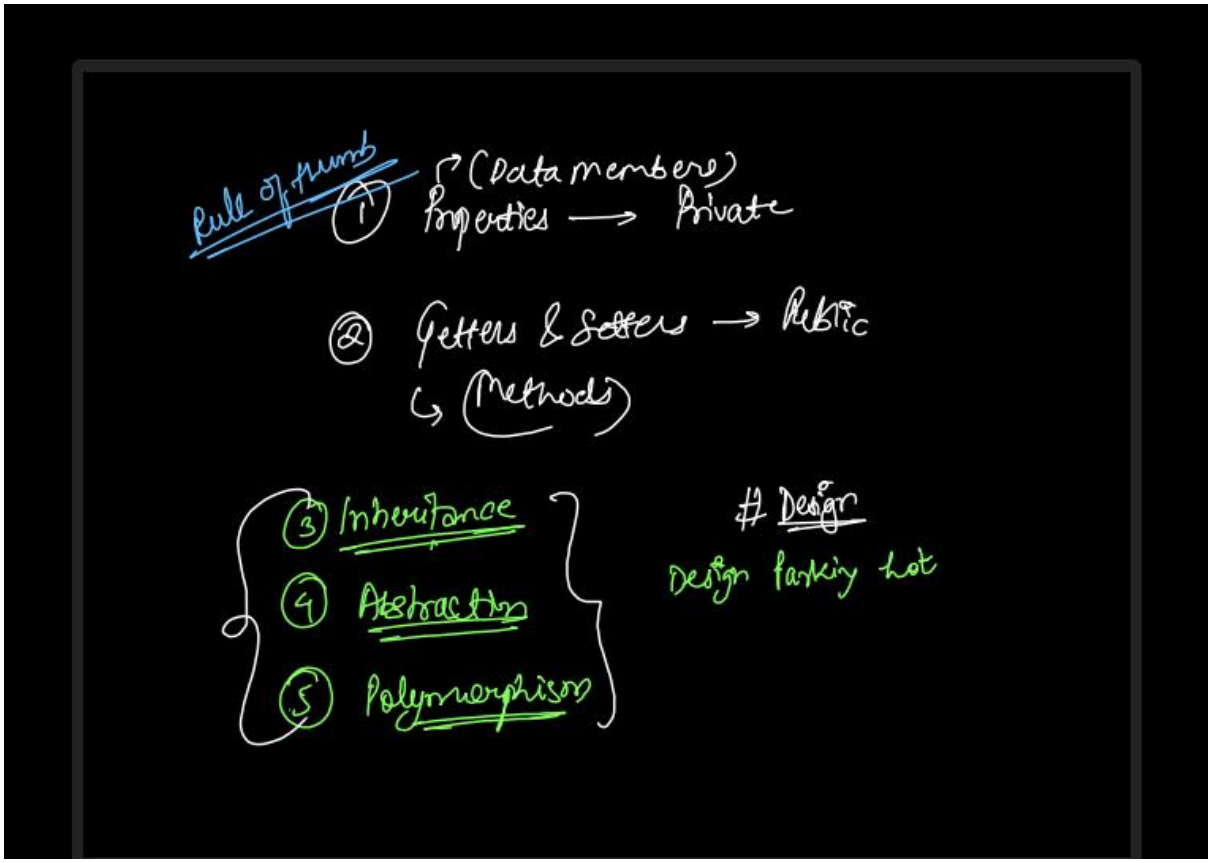


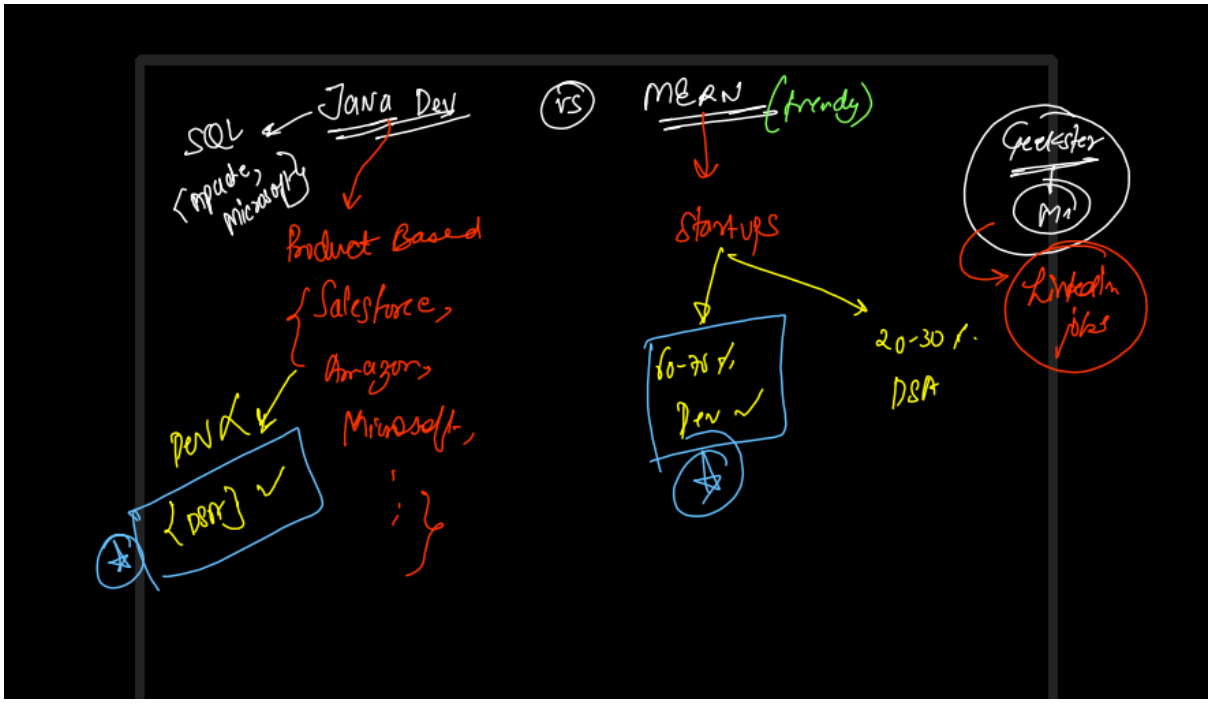
Implementation of Encapsulation

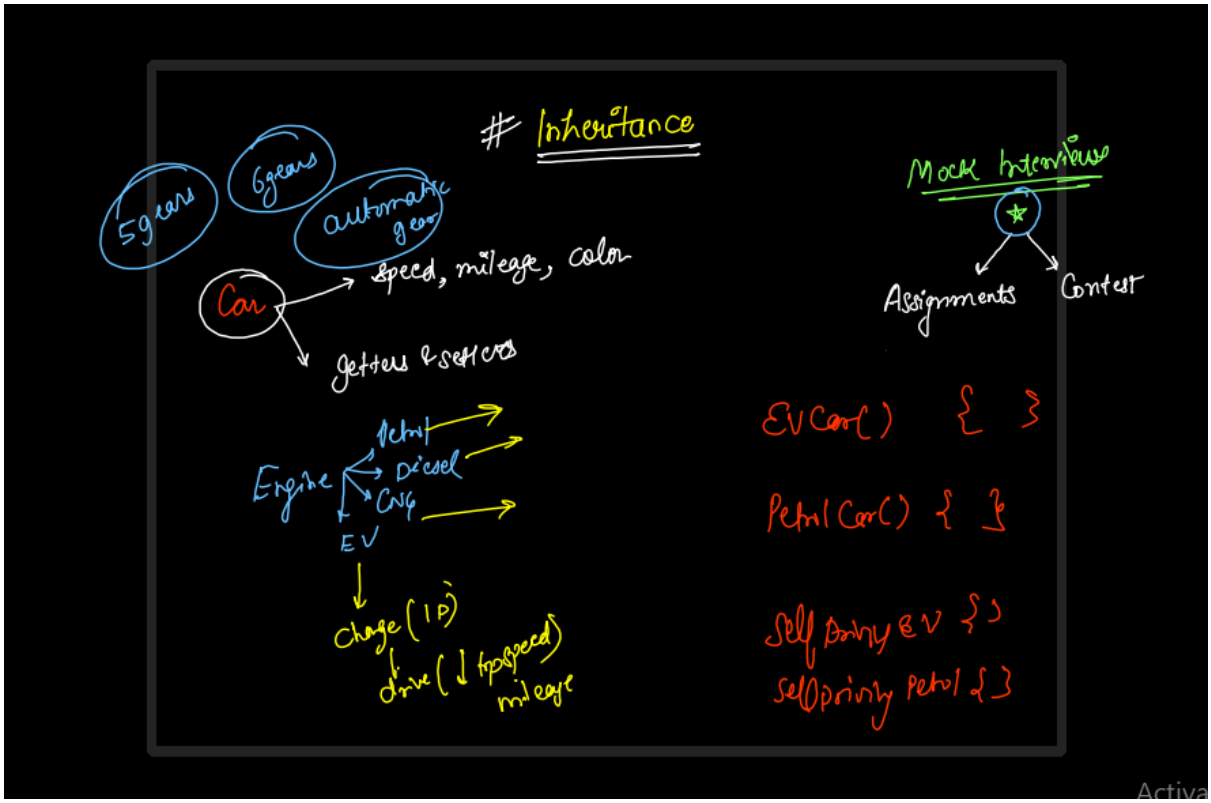
↓

Writing a class & creating its object.

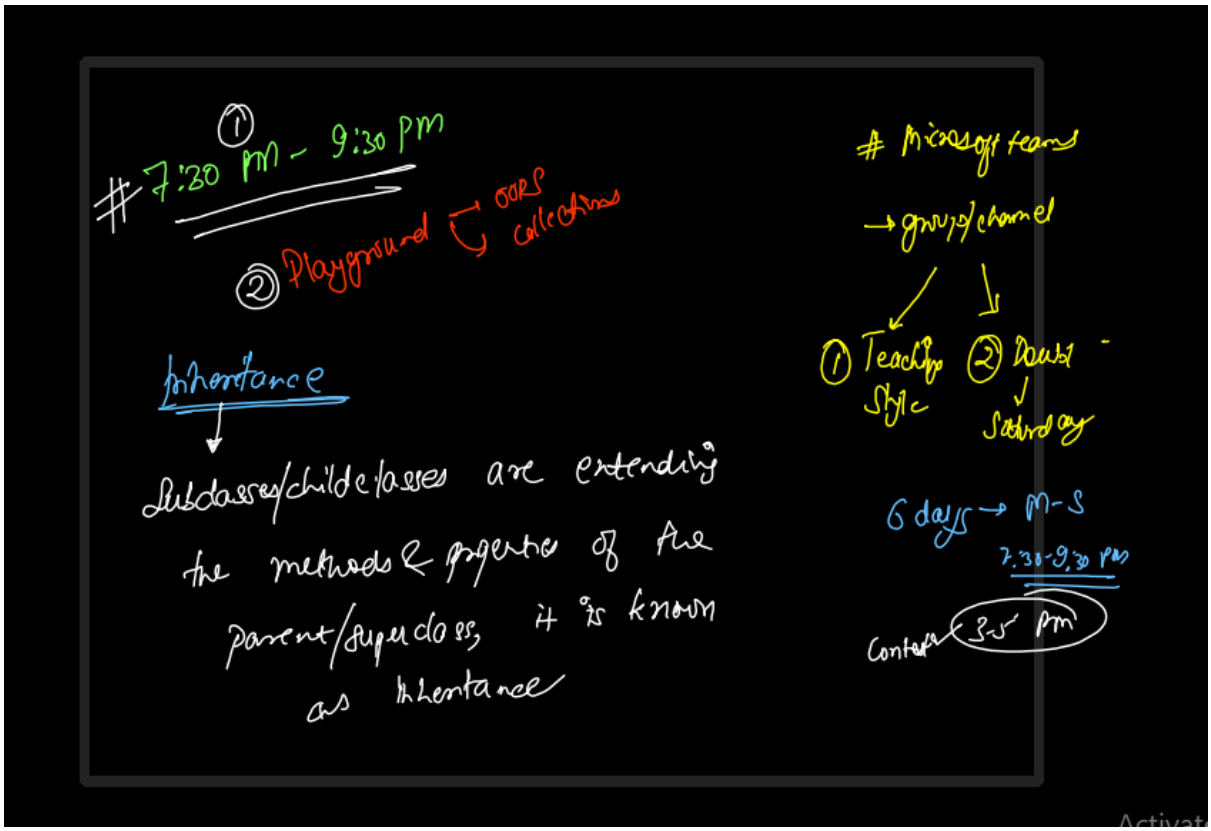

② Data hiding → able to restrict data within the class itself
group classes
Package
Access Modifiers / Specifiers
hide the data from outside world (outside class or main function)
java.util
java.io
java.lang
Collection framework
LL
SBC
Input Output
Scanner
System
Primitive
Method Arguments
Public
vs
Private
available to everyone (both inside & outside)
✓ Object
✓ Class
available only inside the class and not outside
✗ object (.)
✓ class

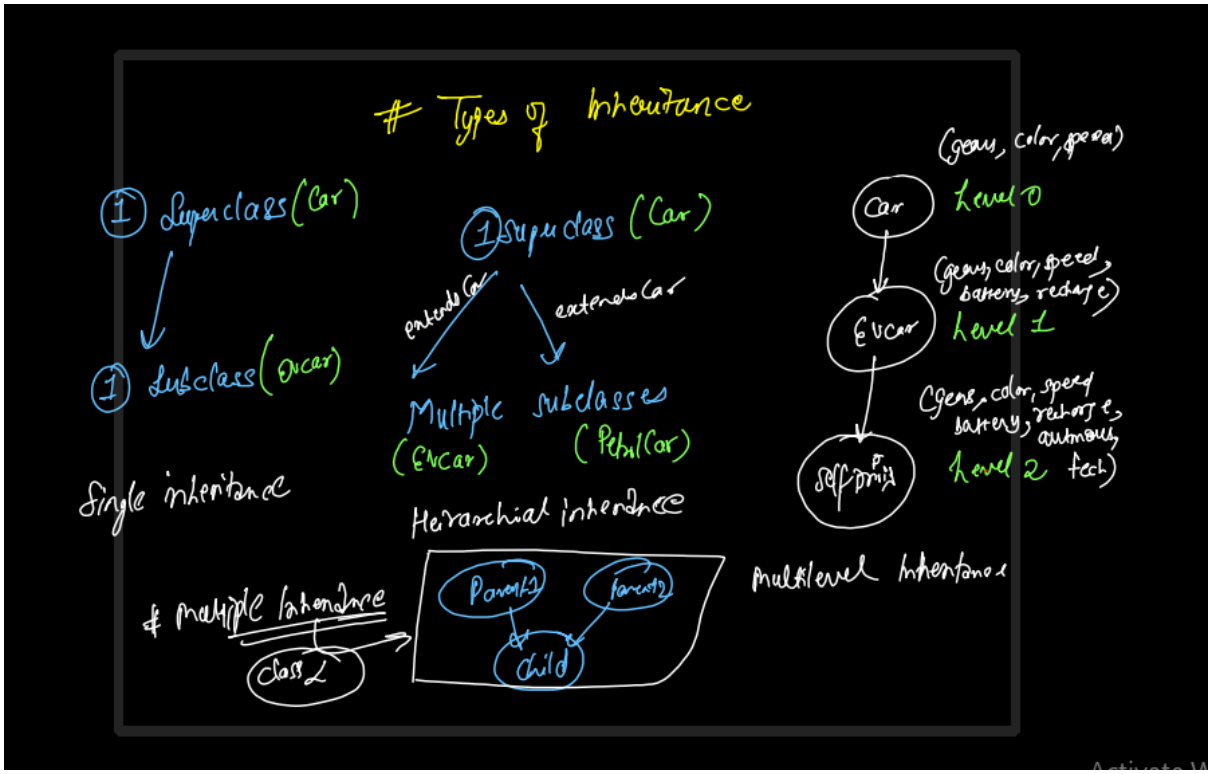

Rule of thumb
① (Data members) Properties → Private
② Getters & Setters → Public
(Methods)
③ Inheritance
④ Abstraction
⑤ Polymorphism
Design
Design Parking lot

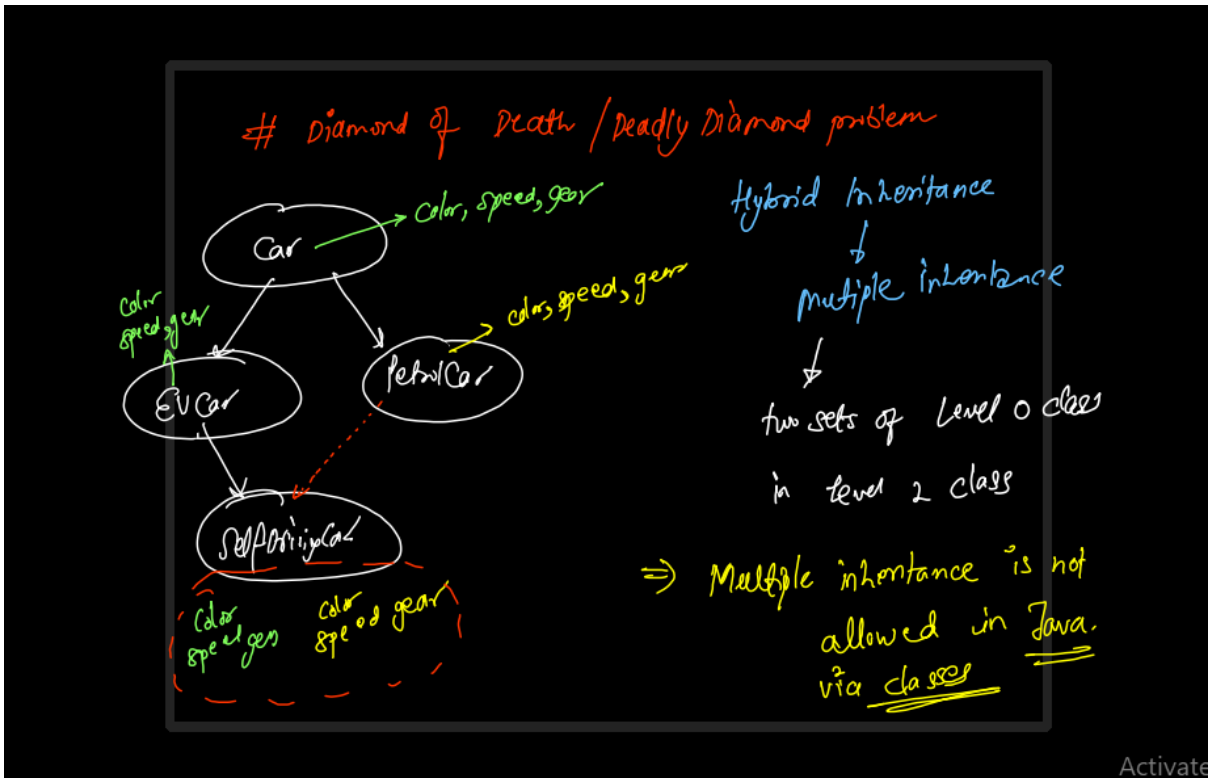

SQL
Java Dev
vs
MERN (trendy)
{oracle, microsoft}
Product Based
{Salesforce, Amazon, Microsoft, ... }
DevX
{DSA}
Startups
60-70 % Dev
20-30 % DSA
Geeksforgeeks
Linkedin jobs


Inheritance
5 gears
6 gears
automatic gear
Car
speed, mileage, color
getters & setters
Engine
Petrol
Diesel
CNG
EV
charge (1D)
drive (↓ topspeed, mileage)
Mock Interviews
Assignments
Contest
EVCar() { }
PetrolCar() { }
selfDrivingEV { }
selfDrivingPetrol { }


#Inheritance
Car
parent class/Super class
Speed, milage, color, gears
self Driving Car
Child classes
/Sub classes
EV Car
AI
Fully autonomous/semi-autonomous.
Charger
Time to charge
Recharge
batt


#7:30 pm - 9:30 pm
①
② Playground
OOPS
collections
Inheritance
Subclasses/child classes are extending the methods & properties of the parent/superclass, it is known as Inheritance
Microsoft teams
→ group/channel
① Teaching Style
② Doubt
Solving
6 days → M-S
7.30-9.30 pm
Contest 3.30 pm


Types of Inheritance
1 Superclass (Car)
1 Subclass (EVCar)
Single inheritance
1 Superclass (Car)
extends Car
extends Car
Multiple (EVCar)
subclasses (PetrolCar)
Hierarchical inheritance
Car
Level 0
(Gears, color, speed)
EVCar
Level 1
(Gears, color, speed, battery, recharge)
SelfDriving
Level 2
(Gears, color, speed, battery, recharge, automous, tech)
Multilevel Inheritance
multiple Inheritance
Class 2
Parent1
Parent2
Child


Diamond of Death / Deadly Diamond problem
Car
Color, speed, gear
EVCar
Color speed gear
PetrolCar
color, speed, gear
SelfDrivingCar
Color speed gear
Color speed gear
Hybrid Inheritance
Multiple Inheritance
two sets of Level 0 class in level 2 class
⇒ Multiple inheritance is not allowed in Java. via classes

Super Keyword

Polymorphism

Compile Time

Run-time

Self referential variable → This

Reference pointing to parent class object → super

Usecase ①

To avoid name-space collision

b/w 2 properties/ methods

b/w child class object

& parent class object

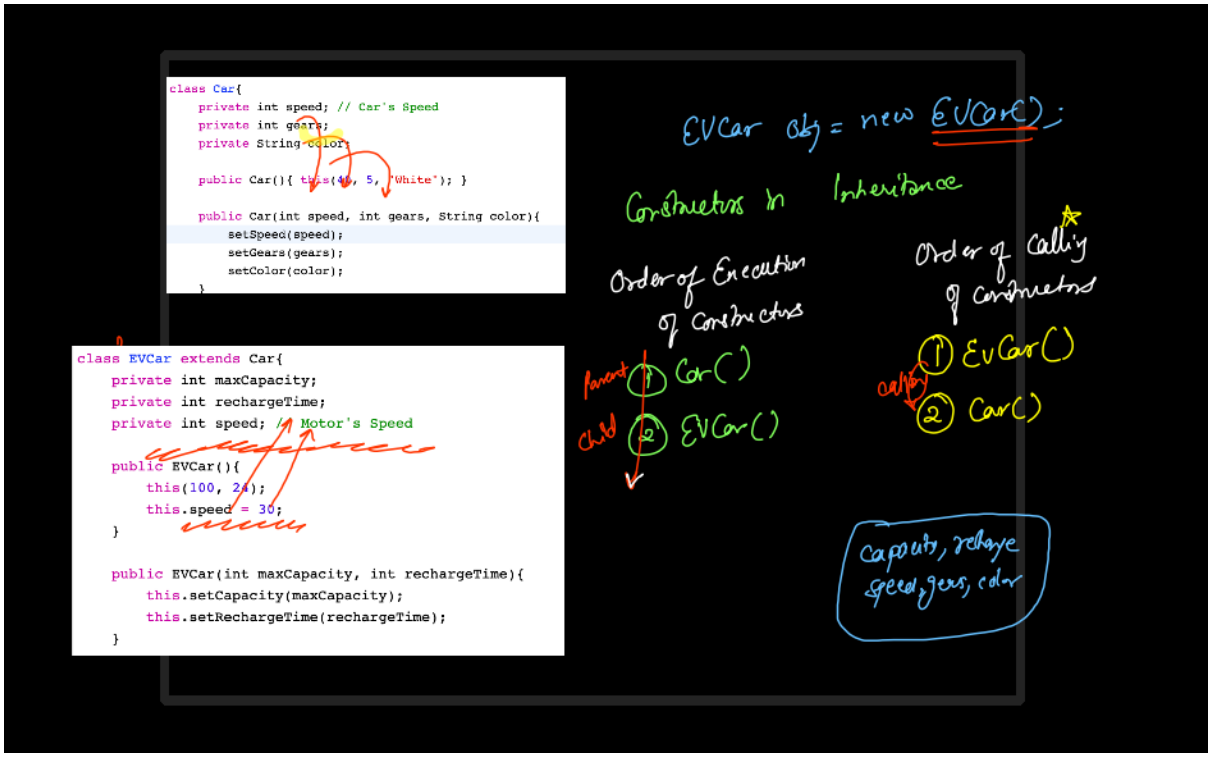
speed ← Car's speed (super)

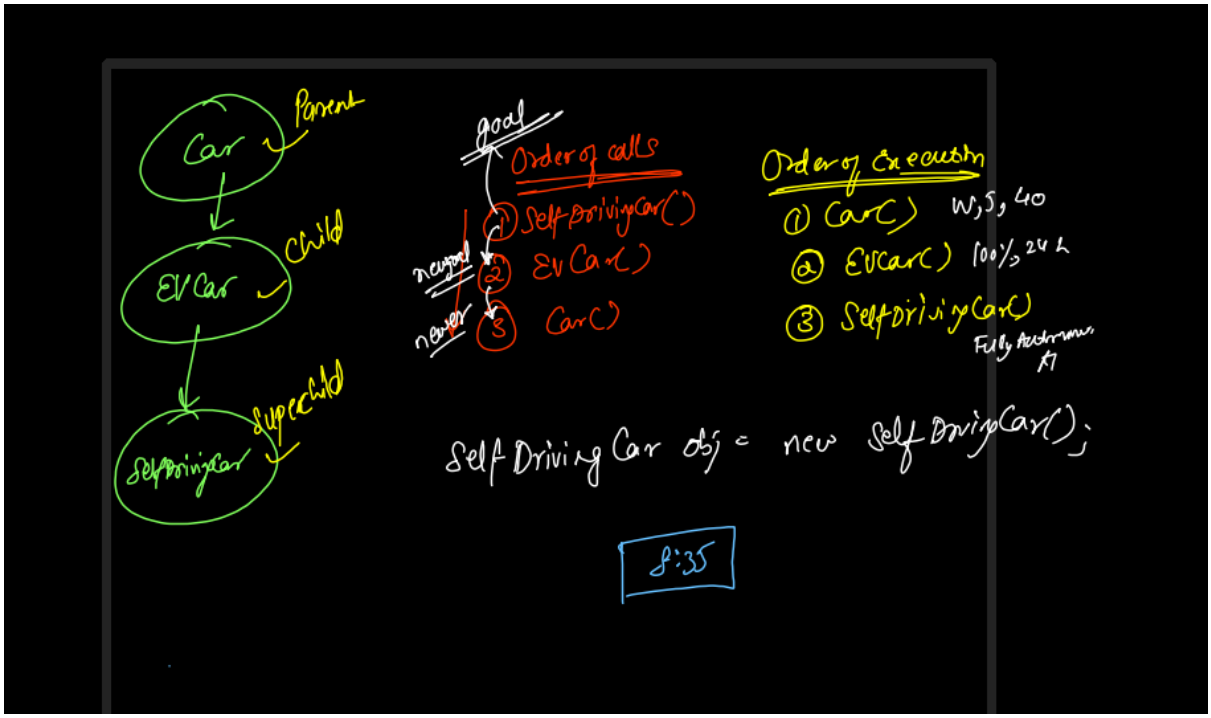
Motor's speed (this)

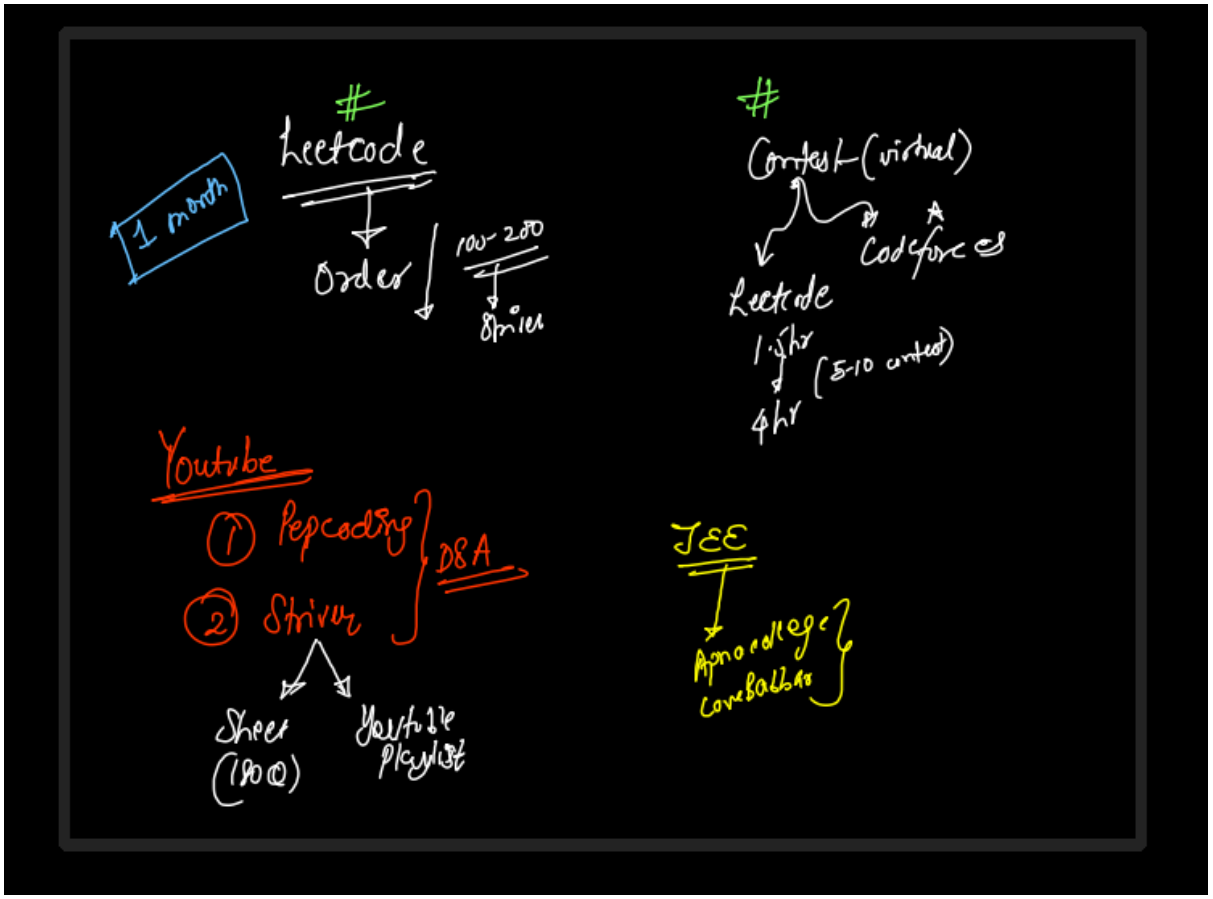
Usecase ②

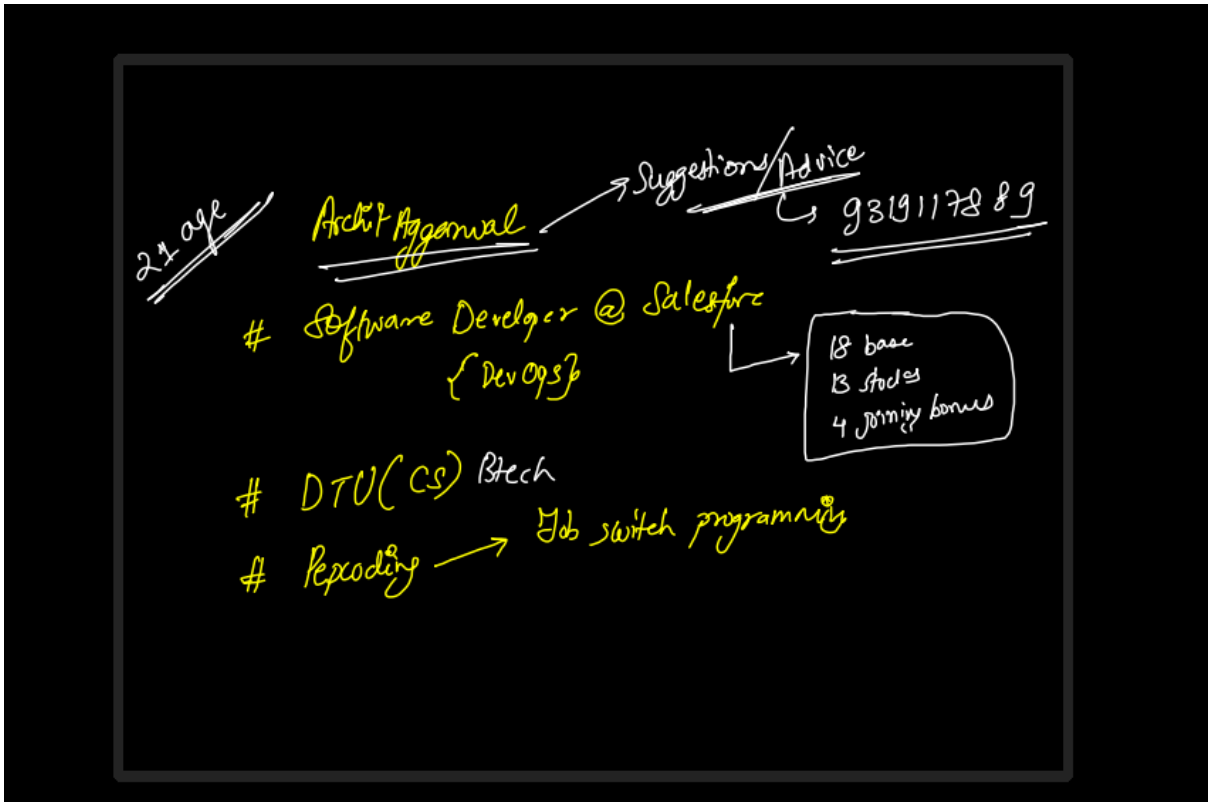
```
public EVCar(int maxCapacity, int rechargeTime, String color, int gears, int speed){
    super(gears, speed, color);
    this.setCapacity(maxCapacity);
    this.setRechargeTime(rechargeTime);
}
```

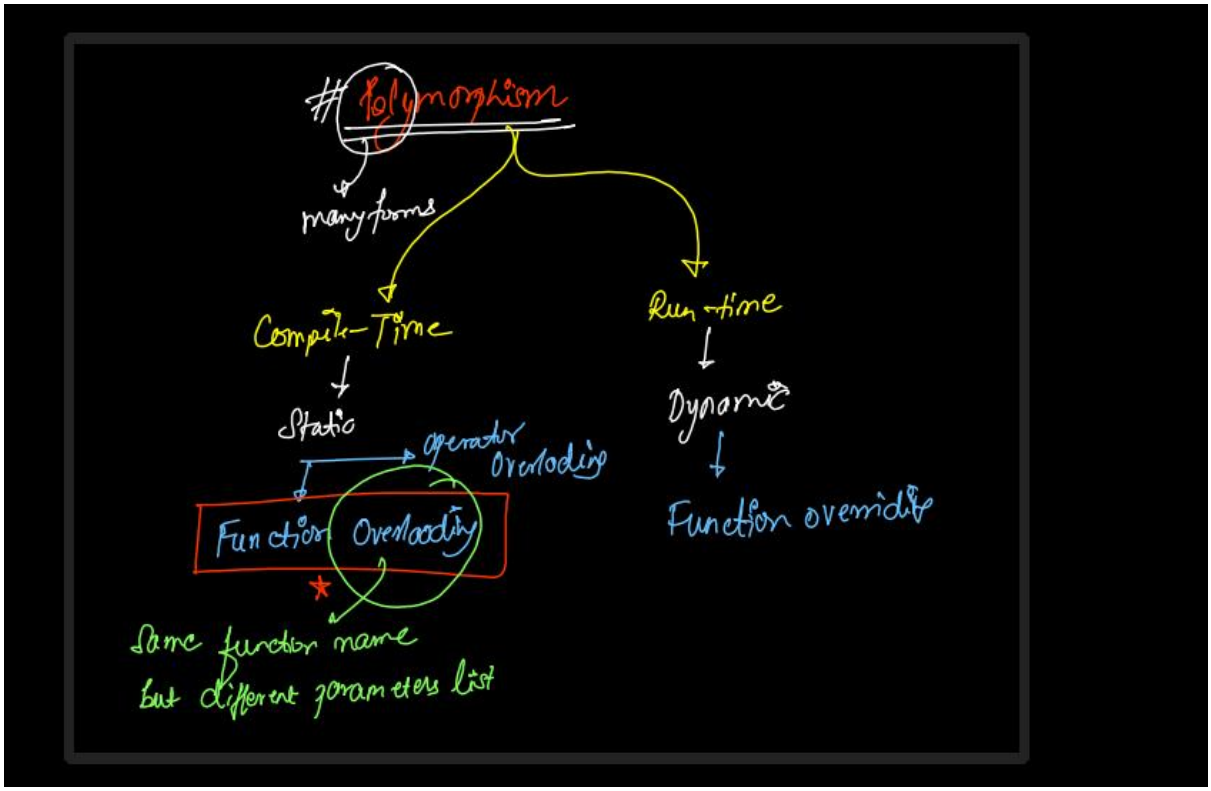
You can call parent's constructor (parameter), super can be used

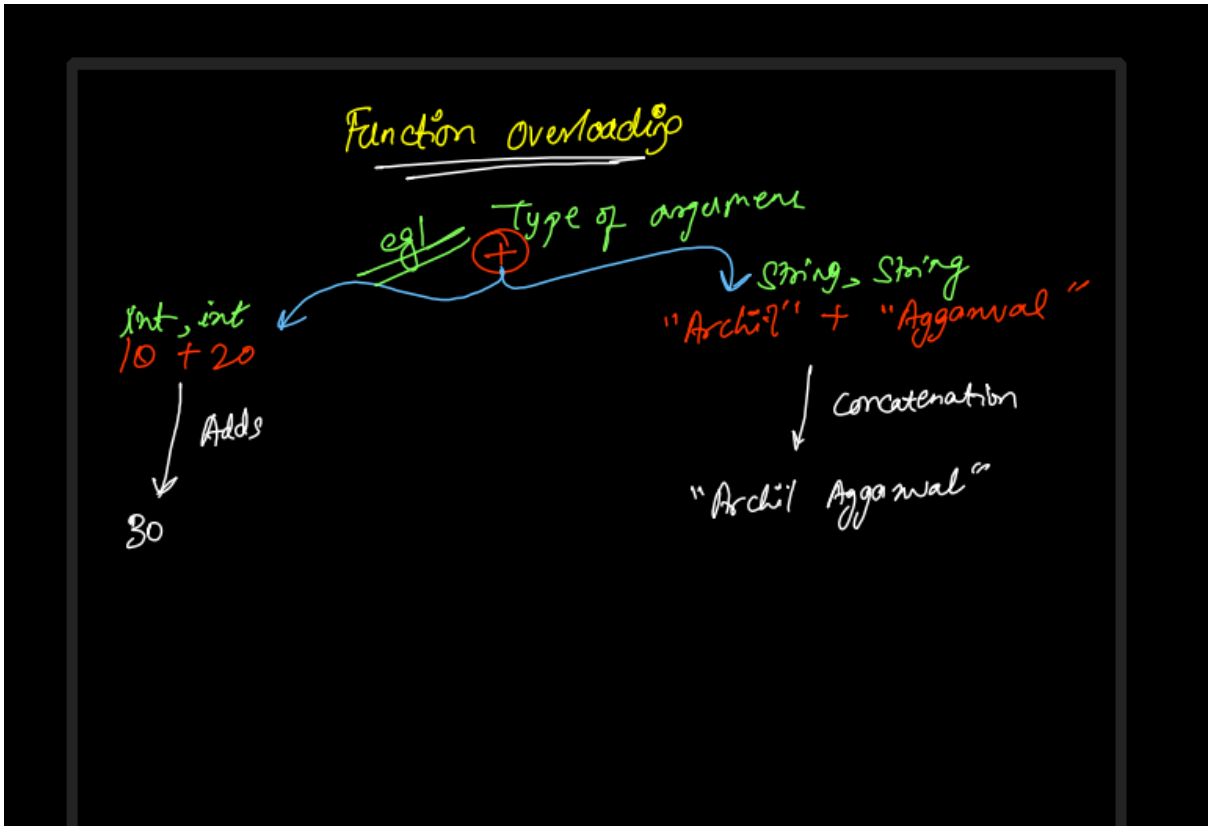

class Car{
private int speed; // Car's Speed
private int gears;
private String color;
public Car(){ this(40, 5, "White"); }
public Car(int speed, int gears, String color){
setSpeed(speed);
setGears(gears);
setColor(color);
}
class EVCar extends Car{
private int maxCapacity;
private int rechargeTime;
private int speed; // Motor's Speed
public EVCar(){
this(100, 24);
this.speed = 30;
}
public EVCar(int maxCapacity, int rechargeTime){
this.setCapacity(maxCapacity);
this.setRechargeTime(rechargeTime);
}
EVCar obj = new EVCar();
Constructors in Inheritance
Order of Execution of Constructors
Order of Calling of Constructors
Parent
Child
Car()
EVCar()
calling
EVCar()
Car()
capacity, recharge, speed, gears, color

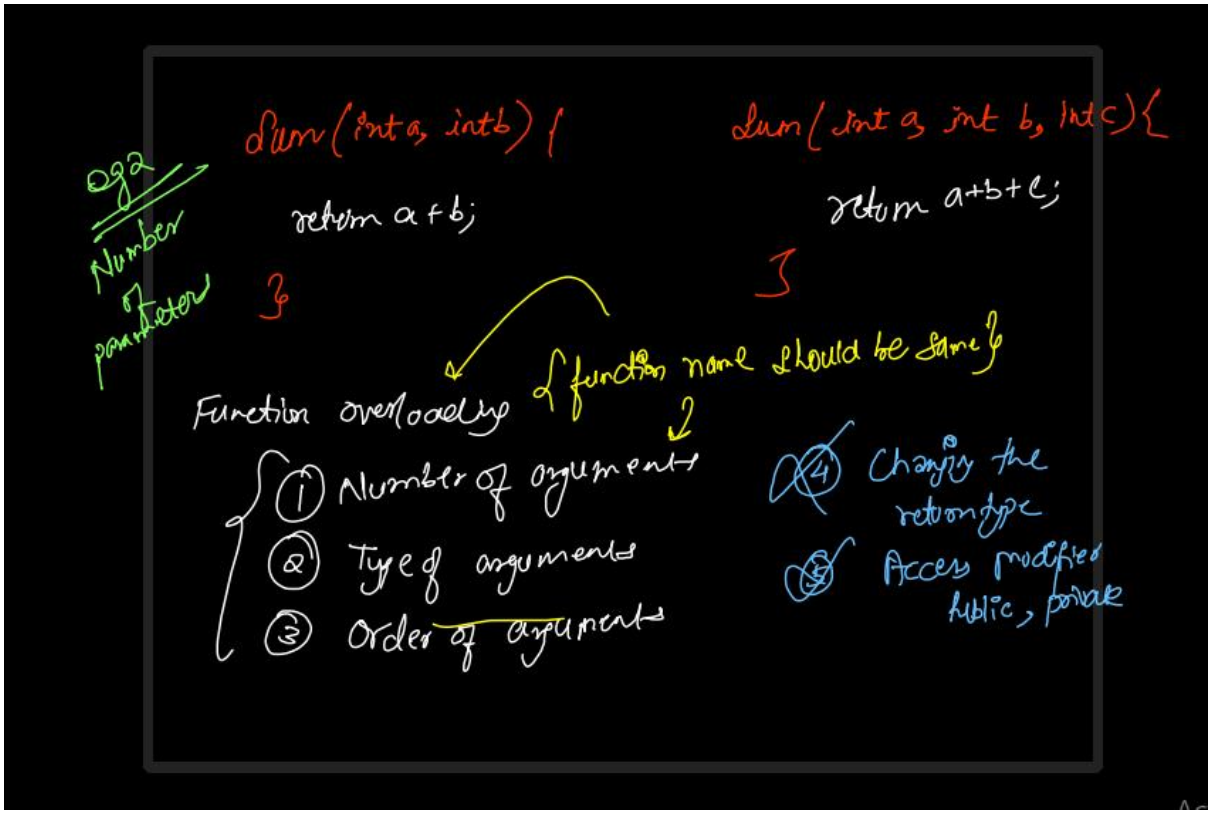

Car
Parent
EVCar
Child
SelfDrivingCar
Superchild
good
Order of calls
SelfDrivingCar()
EVCar()
Car()
newest
never
Order of Execution
Car() W,5,40
EVCar() 100%, 24 h
SelfDrivingCar()
Fully Autonomous
AI
SelfDrivingCar obj = new SelfDrivingCar();
8:35


#
leetcode
1 month
Order
100-200
Striver
#
Contest (virtual)
Codeforces
Leetcode
1.5hr
4hr
(5-10 contest)
Youtube
1) Pepcoding
2) Striver
DSA
Sheet
(180Q)
Youtube playlist
JEE
Apna college
CoreBallers


21 age
Archit Aggarwal
Suggestions/Advice
9319117889
Software Developer @ Salesforce
{DevOps}
18 base
13 stocks
4 joining bonus
DTU (CS) Btech
Pepcoding → Job switch programming

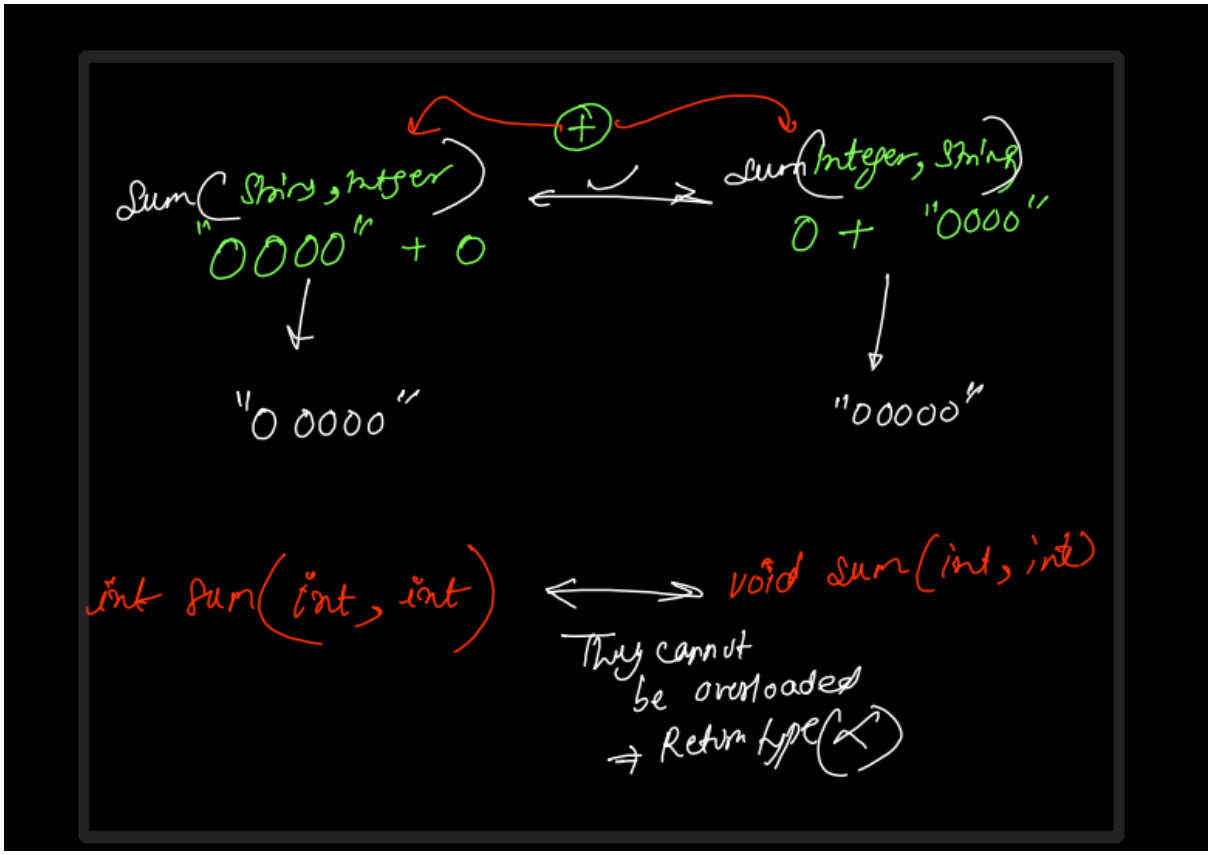

#Polymorphism
many forms
Compile-Time
Static
operator Overloading
Function Overloading
Same function name but different parameters list
Run-time
Dynamic
Function overriding

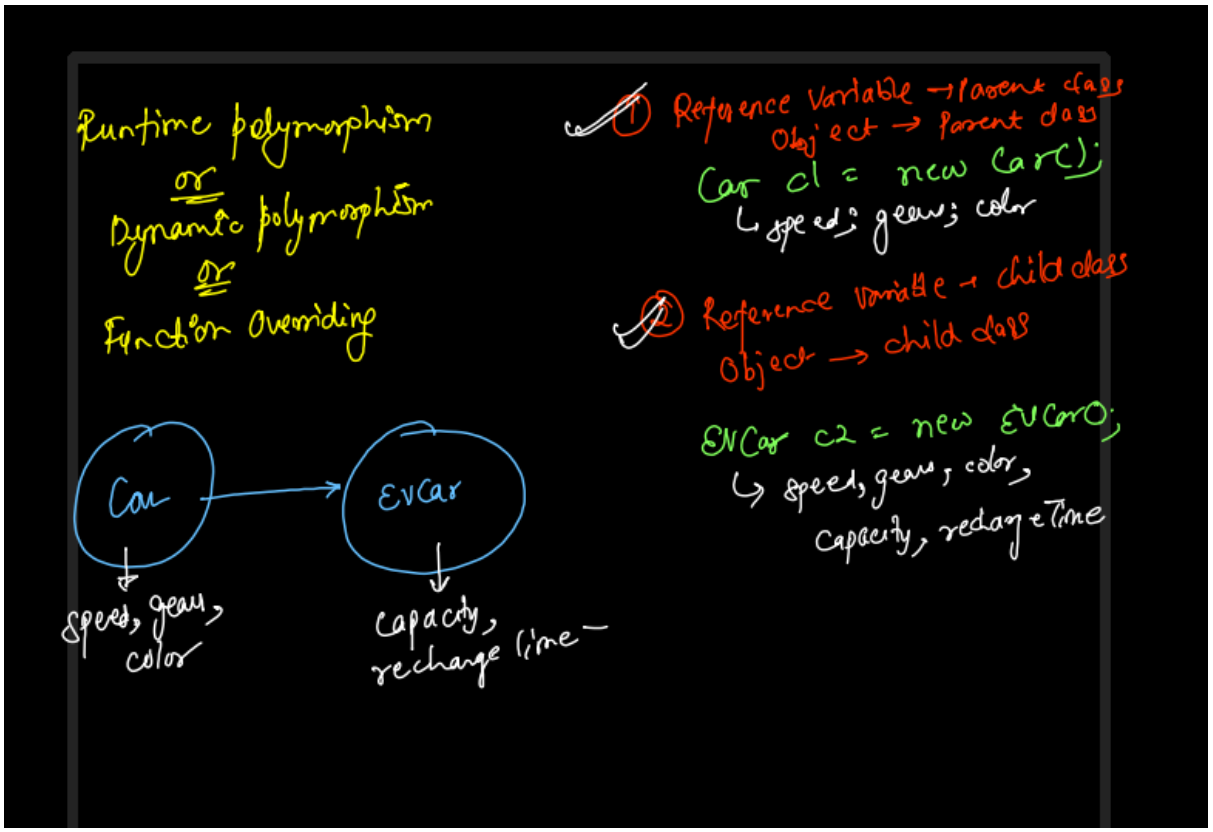

Function Overloading
eg!
Type of arguments
int, int
10 + 20
Adds
30
String, String
"Archi" + "Aggarwal"
Concatenation
"Archi Aggarwal"

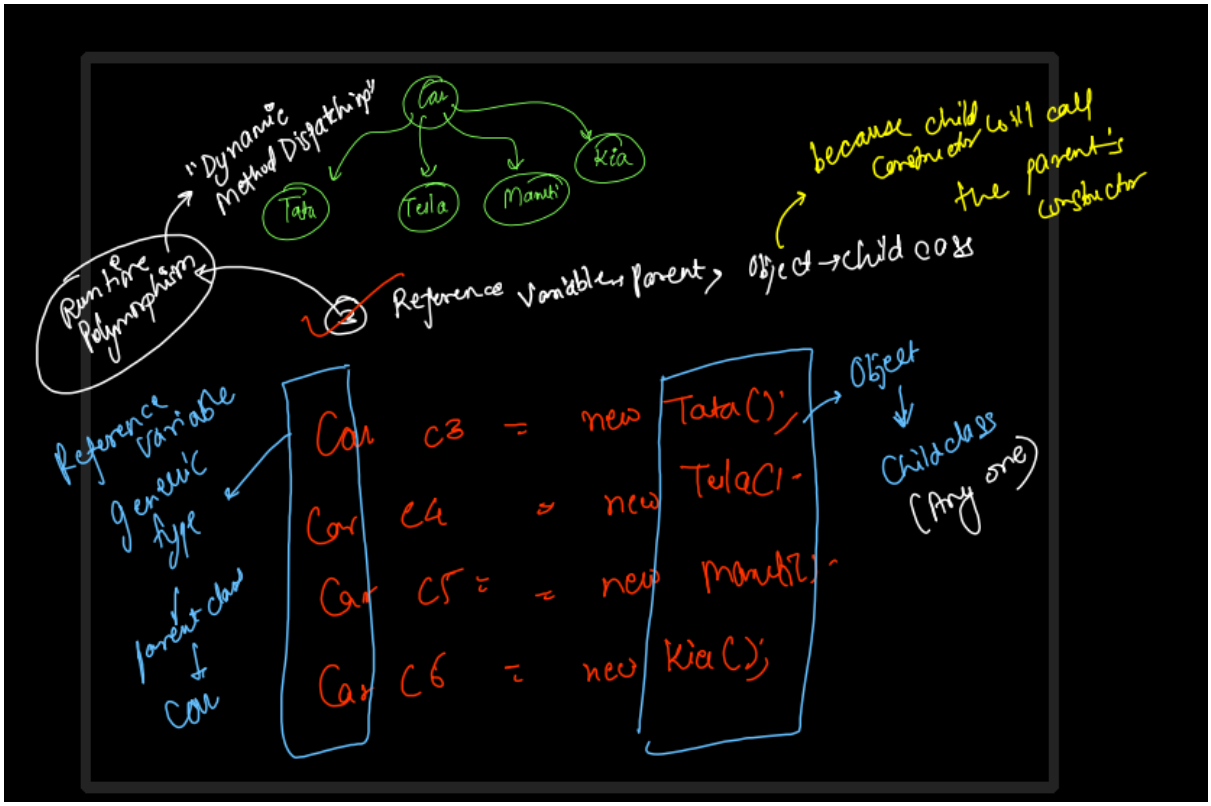


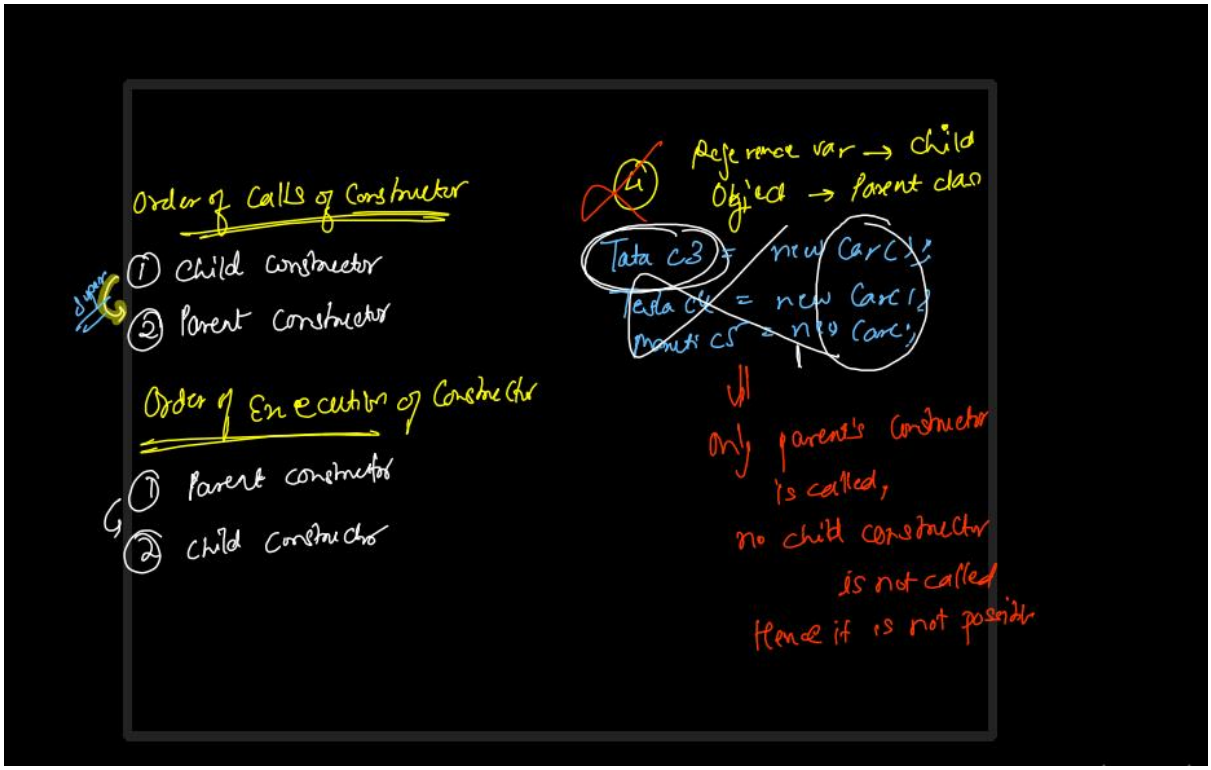

```java
// Type 1: DataType of arguments
public static int sum(int a, int b){
    return a + b;
}

public static String sum(String a, String b){
    return a + b;
}

// Type 2: Number of arguments
public static int sum(int a, int b, int c){
    return a + b + c;
}

// Type 3: Order of Arguments
public static String sum(int a, String b){
    return a + b;
}

public static String sum(String a, int b){
    return a + b;
}
```

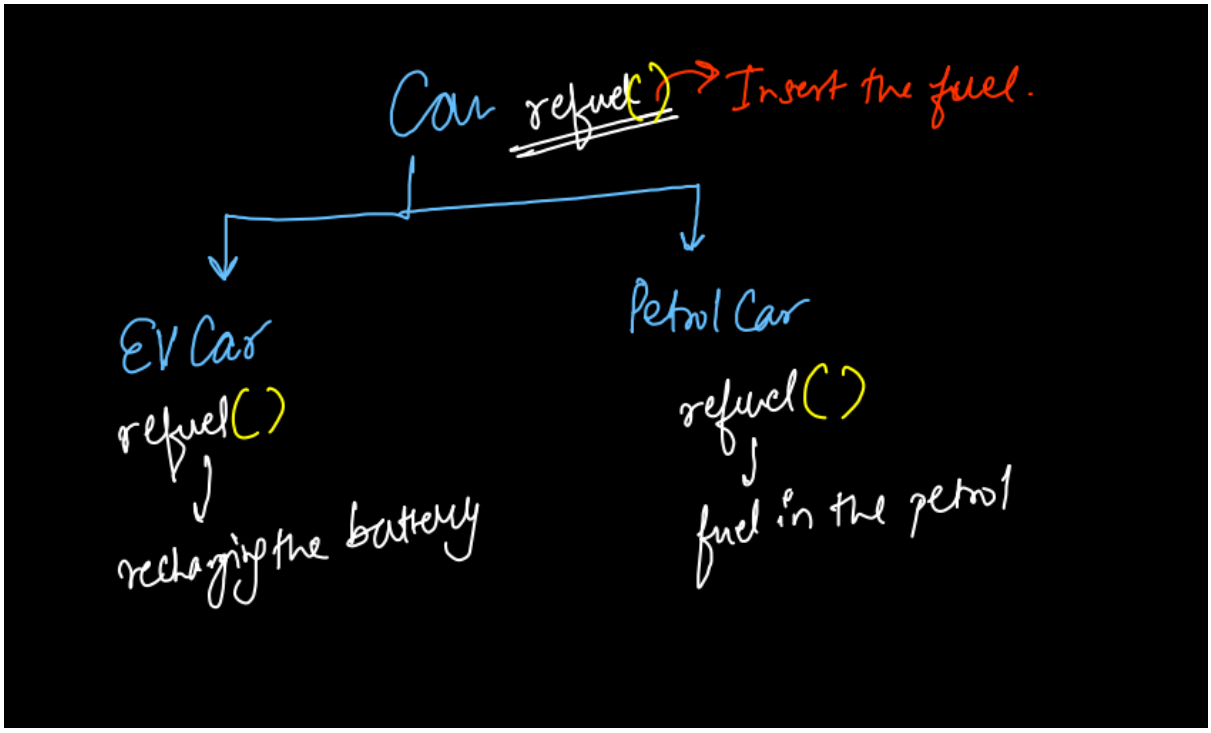
```java
public static void functionOverloadingDemo(){
    System.out.println(sum(10, 20));
    System.out.println(sum("Archit", "Aggarwal"));
    System.out.println(sum(10, 20, 30));
    System.out.println(sum("Archit", 10));
    System.out.println(sum(10, "Archit"));
}
```

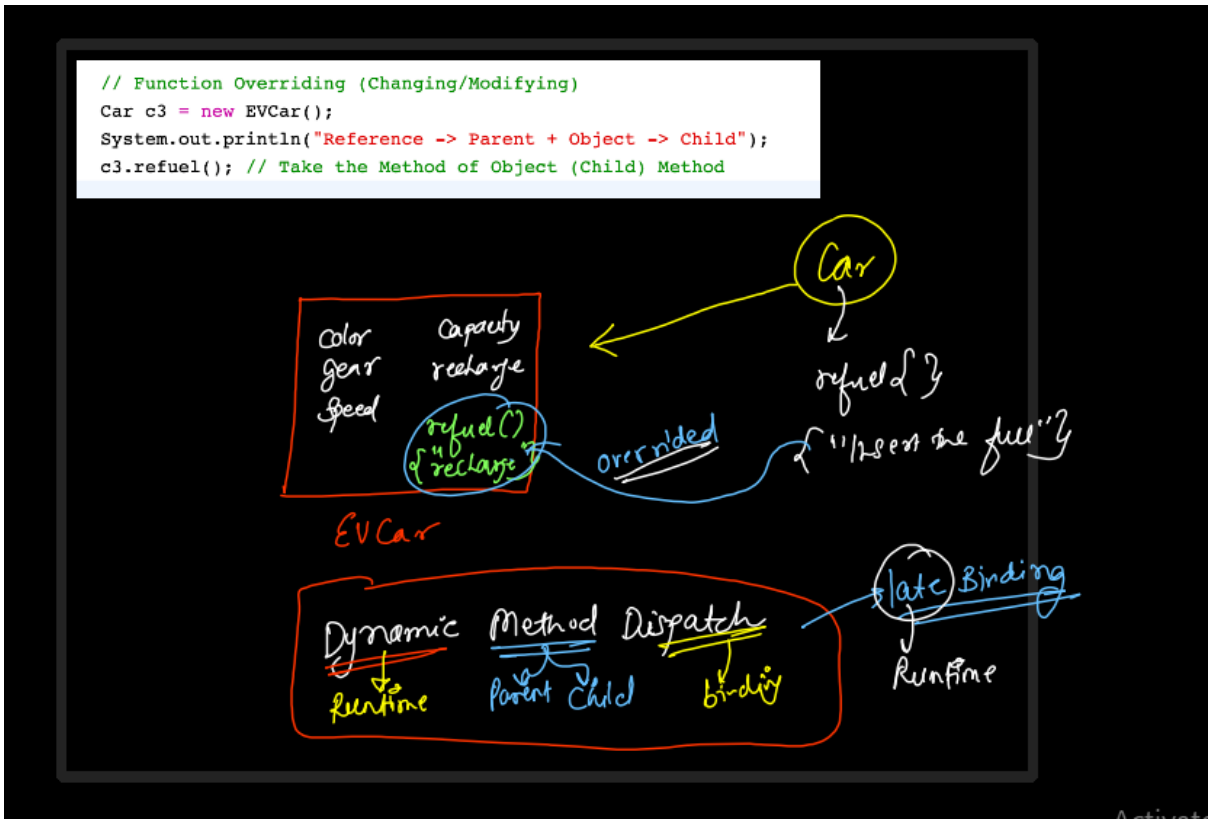

+
Sum(String, Integer)
"0000" + 0
"0 0000"
Sum(Integer, String)
0 + "0000"
"00000"
int sum(int, int)
void sum(int, int)
They cannot be overloaded
→ Return type(x)


Runtime polymorphism
or
Dynamic polymorphism
or
Function overriding
Car
EVCar
speed, gear, color
capacity, recharge time
① Reference Variable → Parent class
Object → Parent class
Car c1 = new Car();
↳ speed; gears; color
② Reference Variable → child class
Object → child class
EVCar c2 = new EVCar();
↳ speed, gears, color,
capacity, recharge time


"Dynamic Method Dispatching"
Car
Tata
Tesla
Maruti
Kia
Run time Polymorphism
Reference Variable (Parent) → Object → Child class
because child constructor will call the parent's constructor
Reference variable generic type
parent class i.e Car
Car c3 = new Tata();
Car c4 = new TeslaCl-
Car c5 = new Maruti();
Car c6 = new Kia();
Object
Childclass (Any one)


Order of Calls of Constructor
① Child constructor
② Parent constructor
Order of Execution of Constructor
① Parent constructor
② Child constructor
Reference var → Child
Object → Parent class
Tata c3 = new Car();
Tesla c4 = new Car();
Maruti c5 = new Car();
Only parent's constructor is called,
no child constructor is not called
Hence it is not possible


Car refuel()
Insert the fuel.
EV Car
refuel()
recharging the battery
Petrol Car
refuel()
fuel in the petrol


// Function Overriding (Changing/Modifying)
Car c3 = new EVCar();
System.out.println("Reference -> Parent + Object -> Child");
c3.refuel(); // Take the Method of Object (Child) Method
Car
refuel()
Color
Gear
Speed
Capacity
recharge
refuel()
"recharge"
overrided
EV Car
Dynamic Method Dispatch
Runtime
Parent Child
binding
late Binding
Runtime

Function overriding

Relatn

1st criteria → 1 is in the parent, other is in child class
(Insert the full) (Recharge the battery)

2nd criteria → Function name, Parameter, Return type } same

eg1
```
Car() {
  → void drive();
  → void drive();
}
```
Overriding ✗ (same class)

Overloaded ✗ (same parameters)

Error

eg2
```
Car() {
  → void drive();
  → int drive();
}
```
Overriding ✗ (same class)

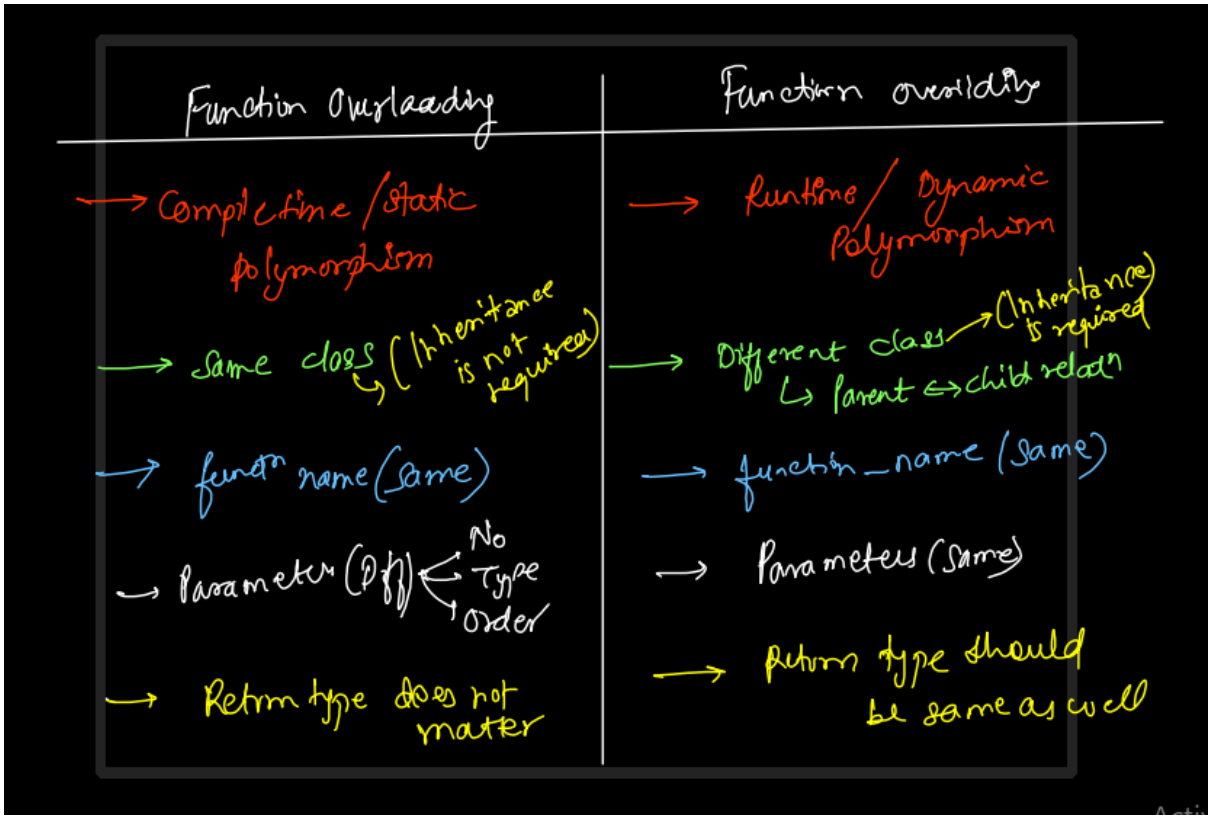
Overloading ✗ (only the return type is diff)

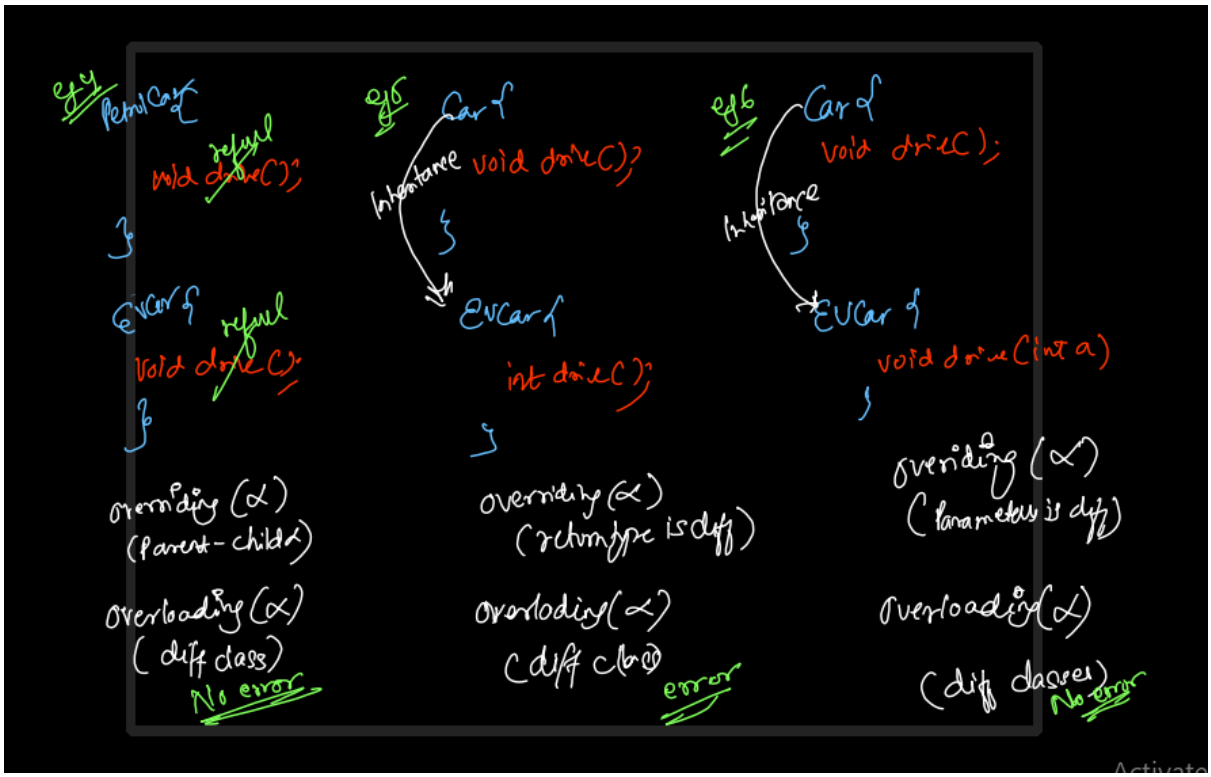
Error

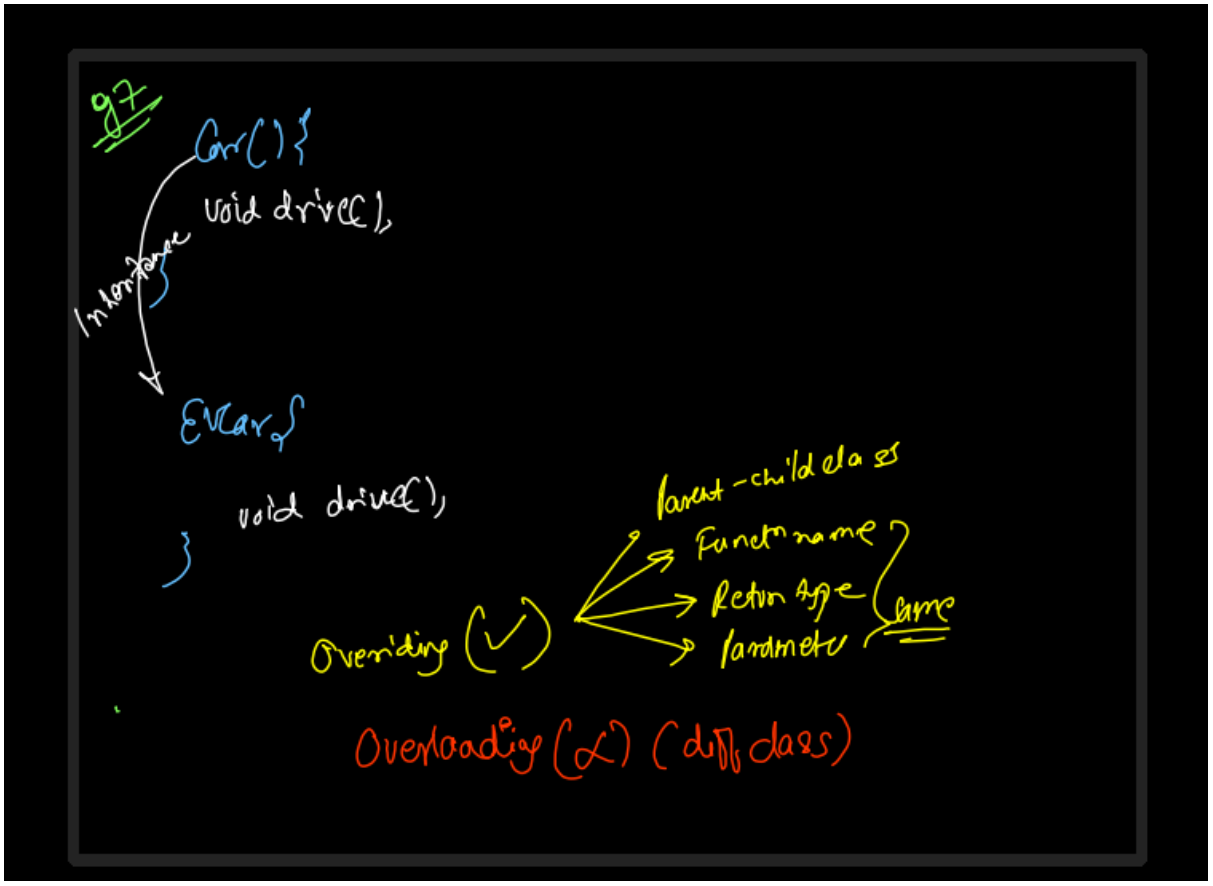
eg3
```
Car() {
  → void drive(int a)
  → void drive(int a, int b)
}
```
Overriding (✓) same class

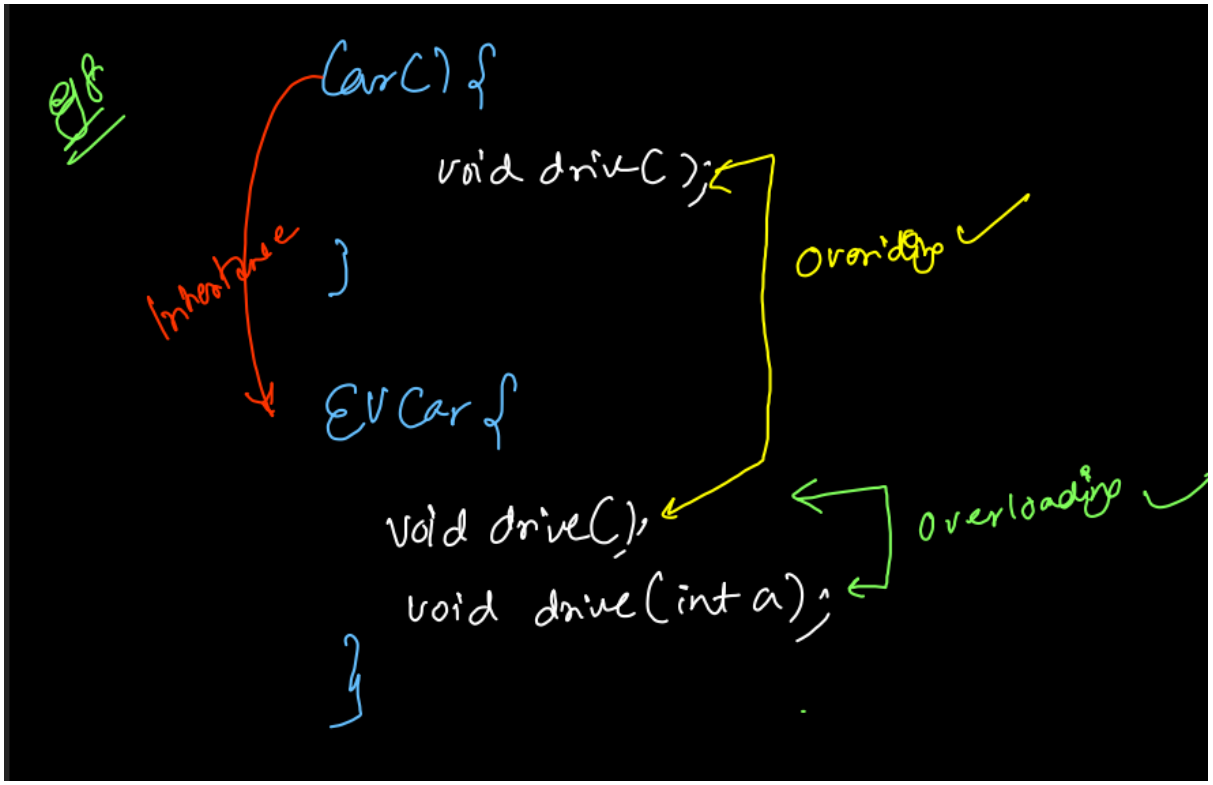
Overloading (✓) (parameters ✓)

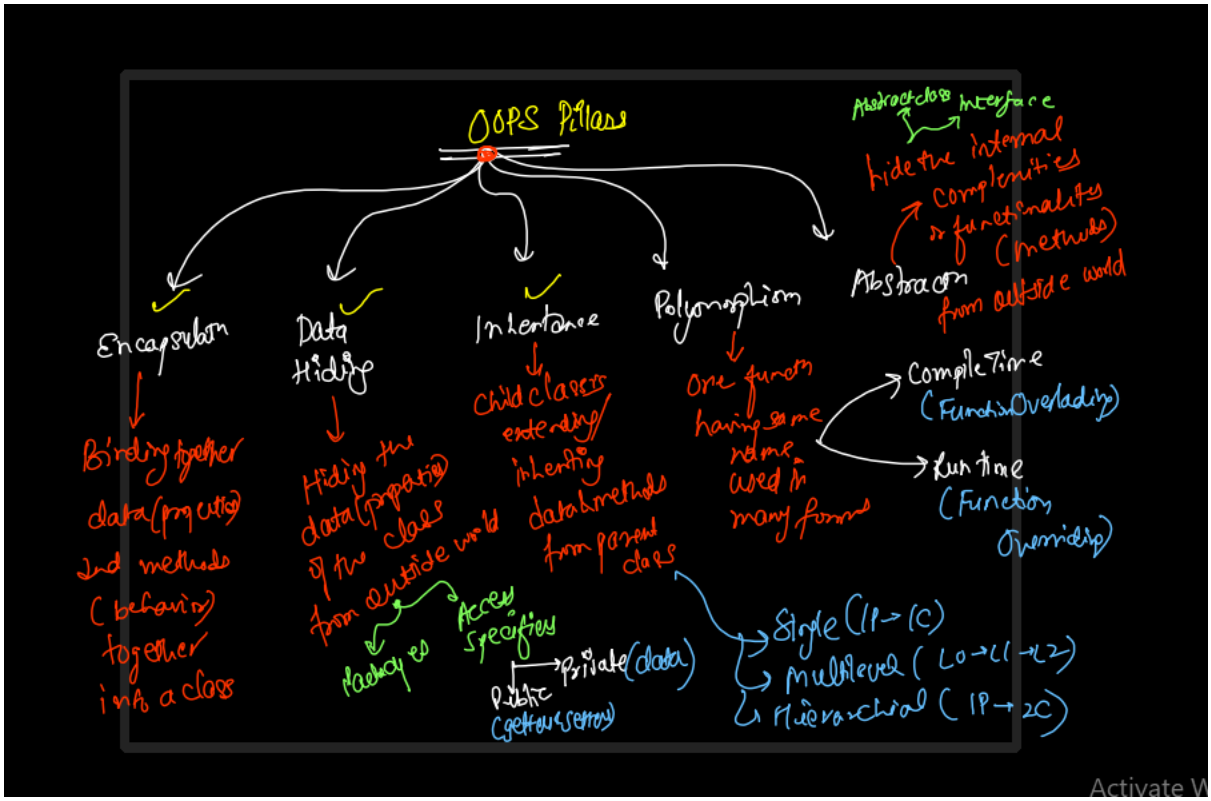
No error

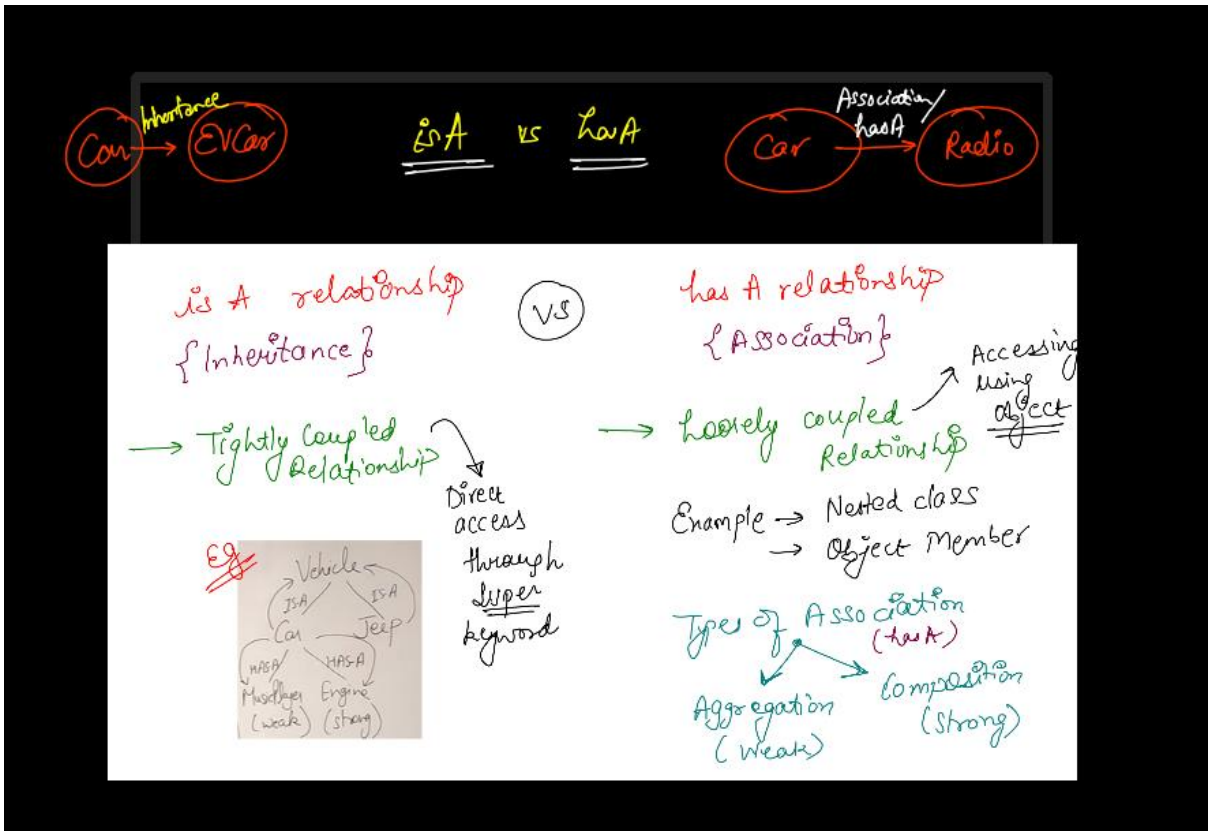

Function Overloading
Function overriding
→ Compile time / static polymorphism
→ Runtime / Dynamic Polymorphism
→ same class (Inheritance is not required)
→ Different class → (Inheritance is required)
↳ Parent ↔ child relation
→ function name (same)
→ function_name (same)
→ Parameters (Diff) ← No Type order
→ Parameters (Same)
→ Return type does not matter
→ Return type should be same as well

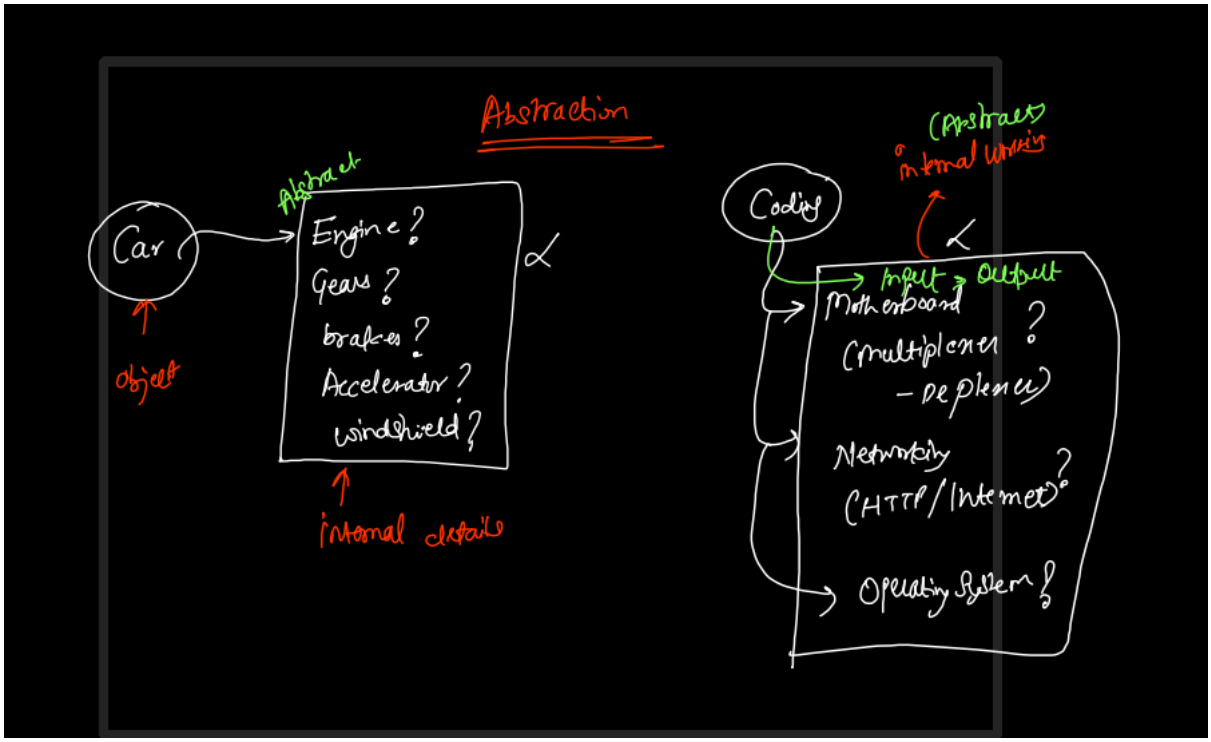

eg1
PetrolCar
void drive();
refuel
EVCar
void drive();
refuel
overriding (∝)
(Parent-child)
Overloading (∝)
(diff class)
No error
eg2
Car
Inheritance
void drive();
EVCar
int drive();
overriding (∝)
(return type is diff)
overloading (∝)
(diff class)
error
eg3
Car
void drive();
Inheritance
EVCar
void drive(int a)
overriding (∝)
(Parameters is diff)
Overloading (∝)
(diff classes)
No error

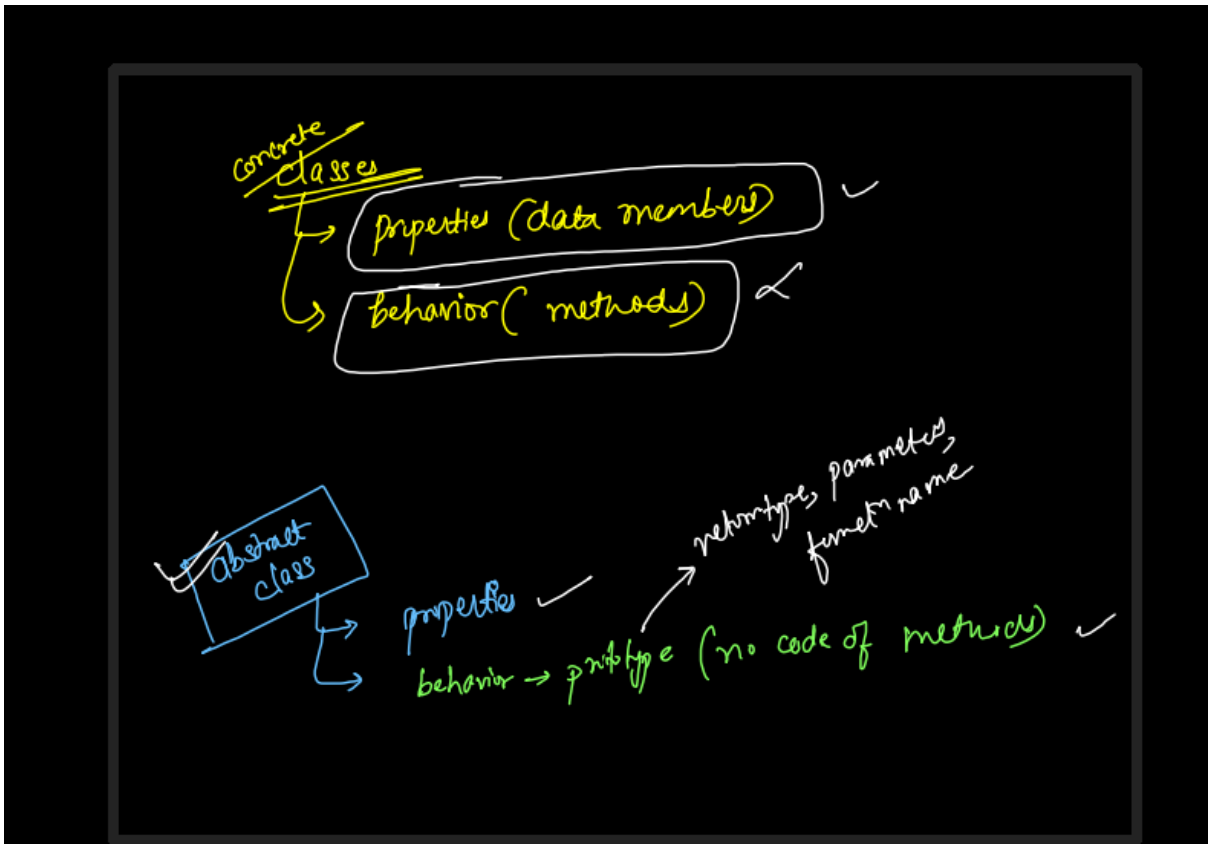

Q7
Car(){
void drive();
Inheritance
EVCar{
void drive();
}
Parent-child class
Function name
Return type
Parameter
same
Overriding (✓)
Overloading (✗) (diff. class)

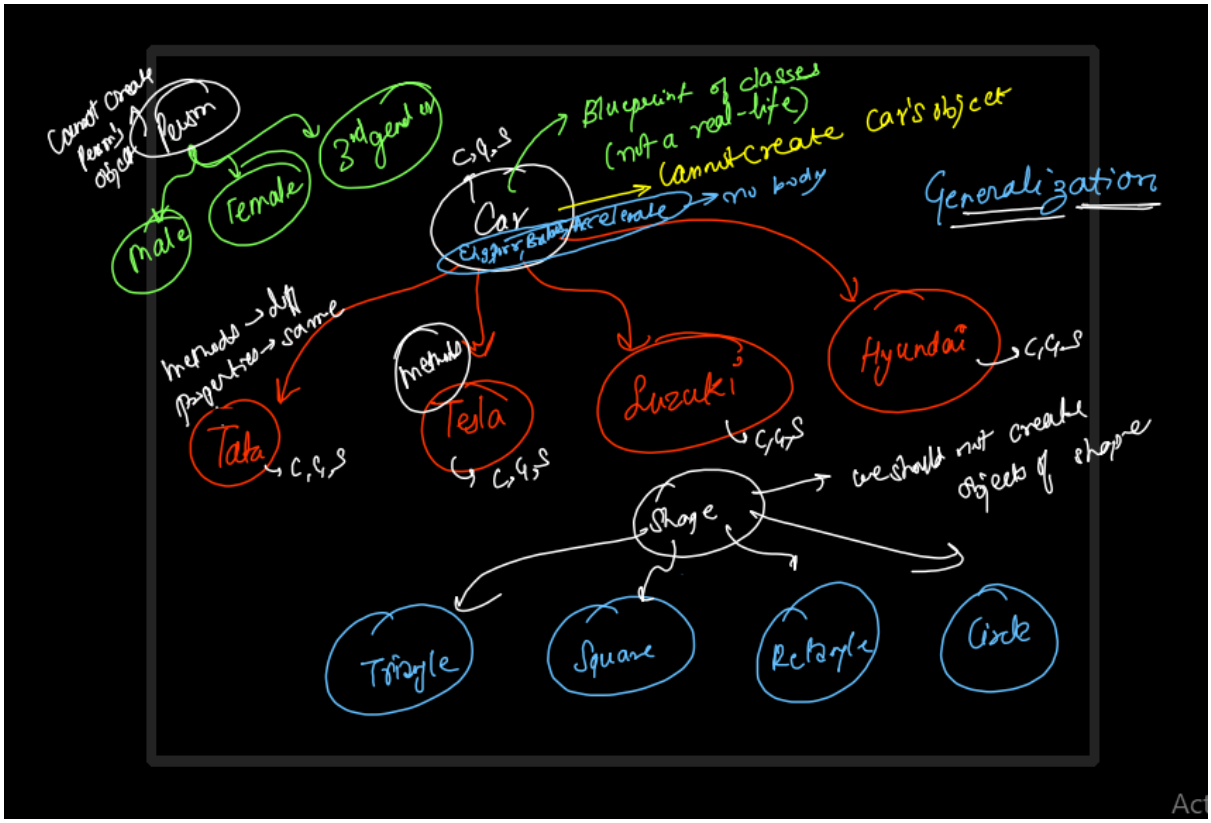

Q8
Car(){
void drive();
}
Inheritance
Overriding
EVCar{
void drive();
void drive(int a);
}
Overloading


OOPS Pillars
Encapsulation
Binding together data (properties) and methods (behaviour) together in a class
Data Hiding
Hiding the data (property) of the class from outside world
Access Specifiers
Private (data)
Public (getter & setter)
Inheritance
Child class extending/inheriting data & methods from parent class
Single (1P → 1C)
Multilevel (1P → 1C → 1C)
Hierarchical (1P → 2C)
Polymorphism
One function having same name used in many forms
Compile Time (Function Overloading)
Run time (Function Overriding)
Abstraction
Abstract class
Interface
Hide the internal complexities & functionalities (methods) from outside world


Inheritance
Car
EVCar
IsA vs HasA
Association/ hasA
Car
Radio
is A relationship
{Inheritance}
VS
has A relationship
{Association}
→ Tightly Coupled Relationship
Direct access through super keyword
→ Loosely coupled Relationship
Accessing using Object
Example → Nested class
→ Object Member
Eg
Vehicle
ISA
ISA
Car
Jeep
HAS-A
HAS-A
Steering (weak)
Engine (strong)
Types of Association (has A)
Aggregation (weak)
Composition (strong)


Abstraction
Abstract
Car
Engine ?
Gears ?
brakes ?
Accelerator ?
windshield ?
Object
internal details
Coding
(Abstract)
internal working
input
output
Motherboard ?
(multiplexer
- depleneer)
Networking
(HTTP/Internet)?
Operating System ?


concrete classes
properties (data members)
behavior (methods)
Abstract class
properties
behavior → prototype (no code of methods)
return type, parameters, function name

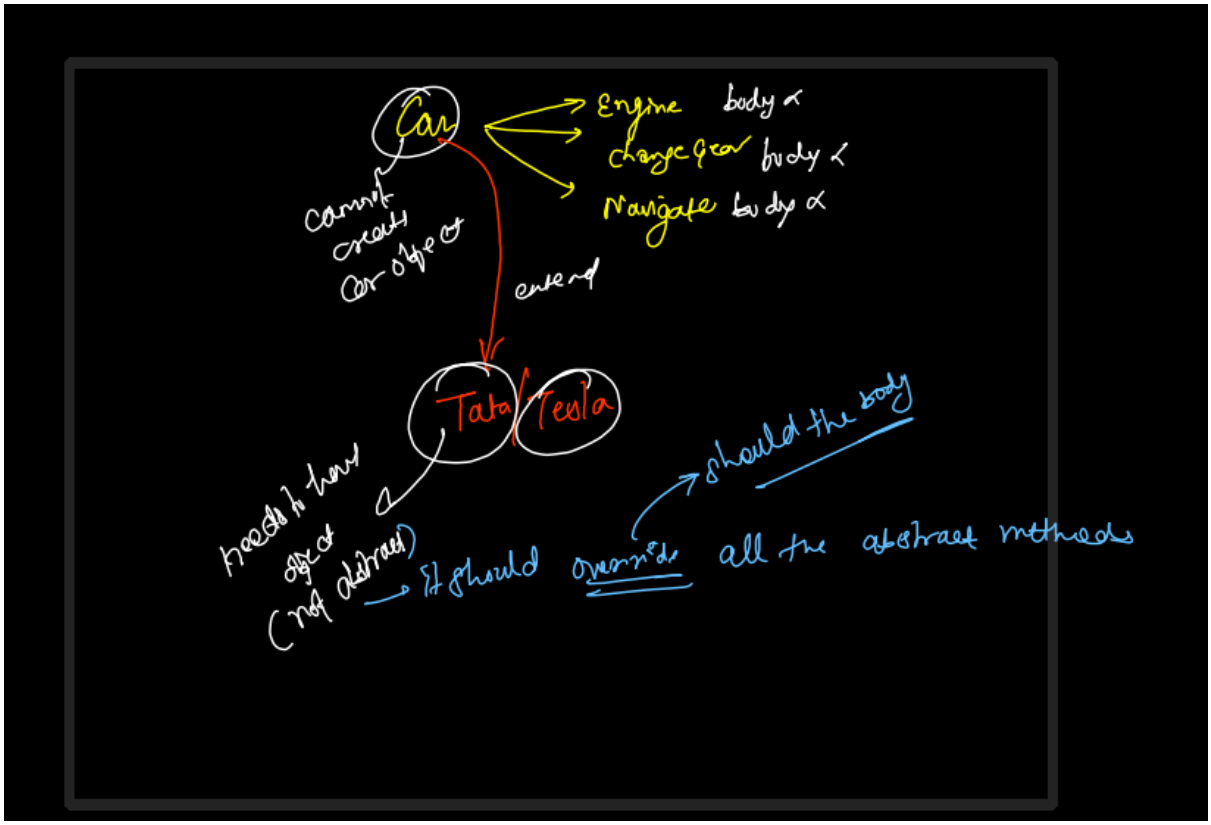


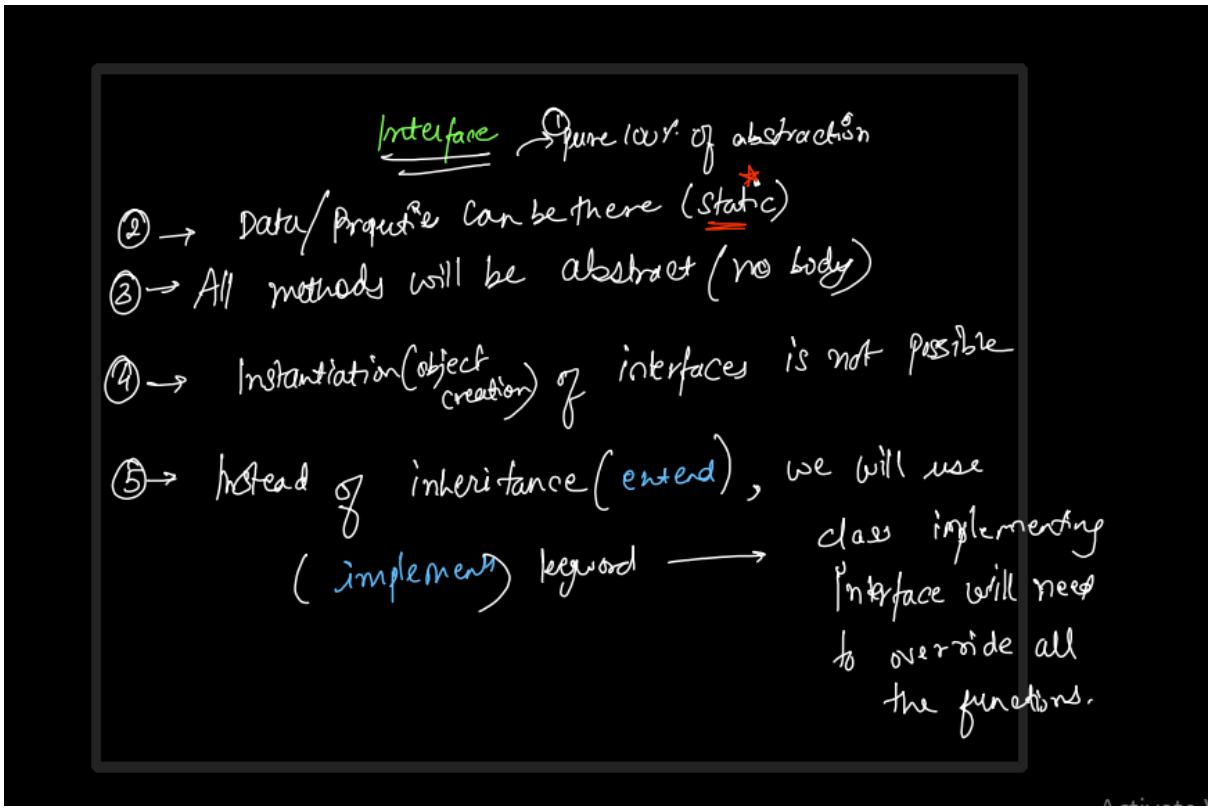
Abstract class → 0-100% abstraction

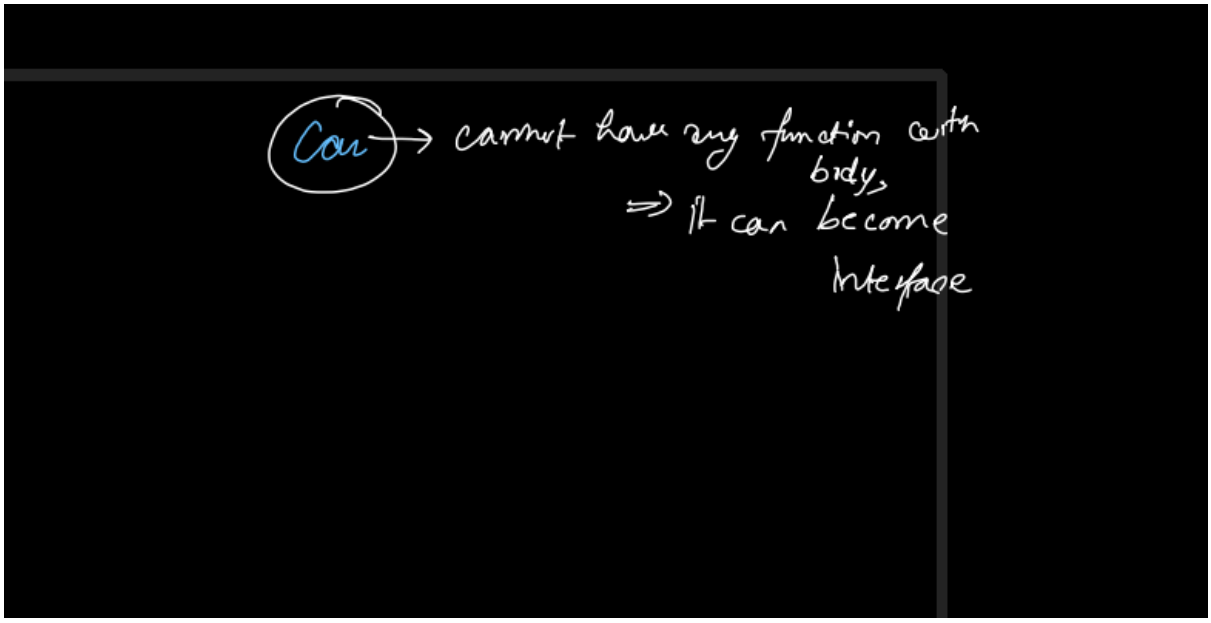
1. → class with properties (data), but the methods will have no body / with body

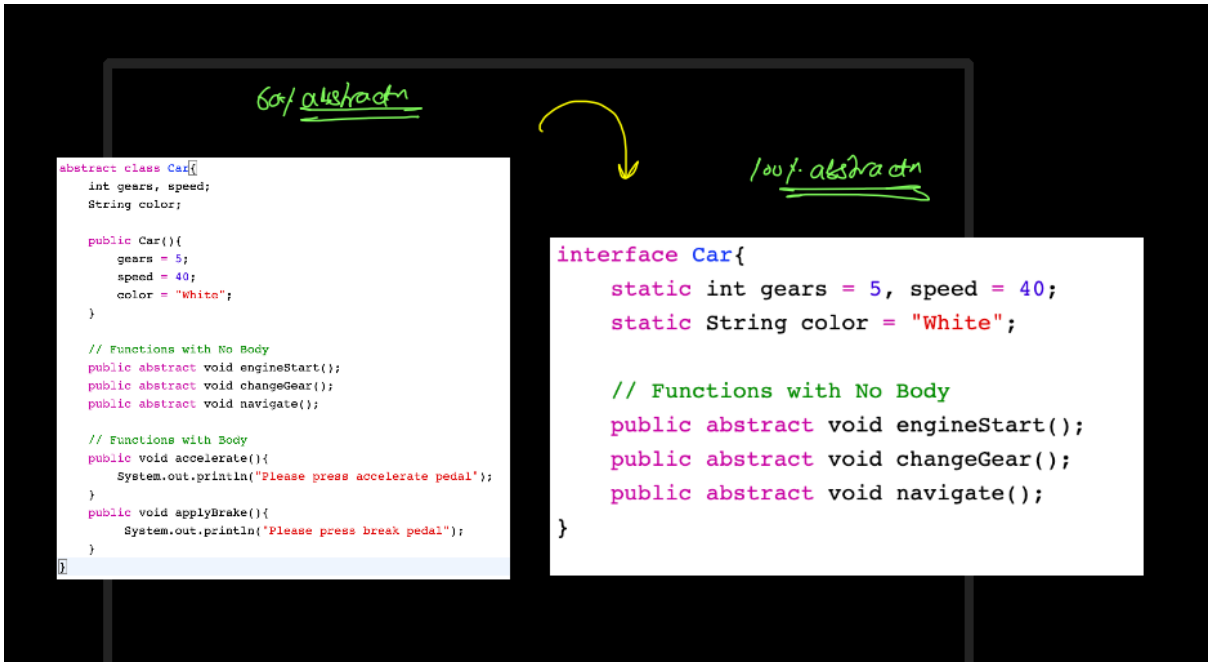
2. → We cannot instantiate (create objects) of abstract classes!

3. → To inherit the abstract class, you should override the abstract methods


Car
Engine body ×
Change Gear body ×
Navigate body ×
cannot create Car object
extend
Tata
Tesla
needs to have body (not abstract)
→ It should override all the abstract methods
should the body


Interface → pure level of abstraction
② → Data/Properties can be there (static)
③ → All methods will be abstract (no body)
④ → Instantiation (object creation) of interfaces is not possible
⑤ → Instead of inheritance (extend), we will use (implements) keyword → class implementing Interface will need to override all the functions.






```java
abstract class Car{
    int gears, speed;
    String color;

    public Car(){
        gears = 5;
        speed = 40;
        color = "White";
    }

    // Functions with No Body
    public abstract void engineStart();
    public abstract void changeGear();
    public abstract void navigate();

    // Functions with Body
    public void accelerate(){
        System.out.println("Please press accelerate pedal");
    }
    public void applyBrake(){
        System.out.println("Please press break pedal");
    }
}
```

```java
interface Car{
    static int gears = 5, speed = 40;
    static String color = "White";

    // Functions with No Body
    public abstract void engineStart();
    public abstract void changeGear();
    public abstract void navigate();
}
```