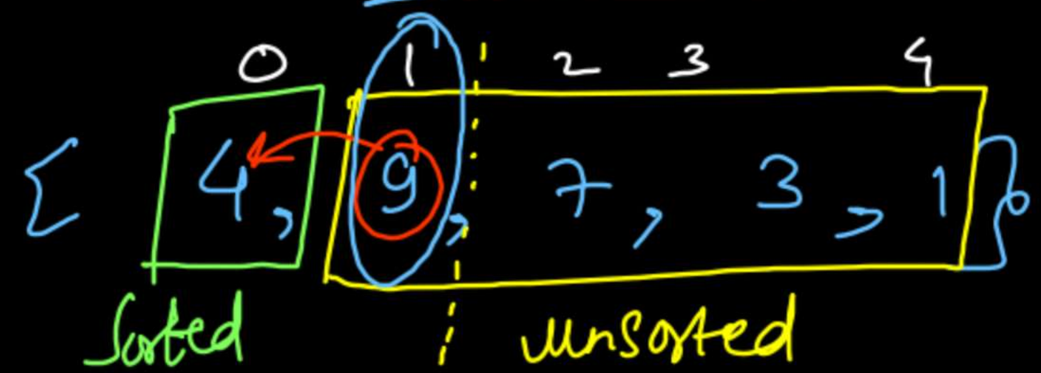
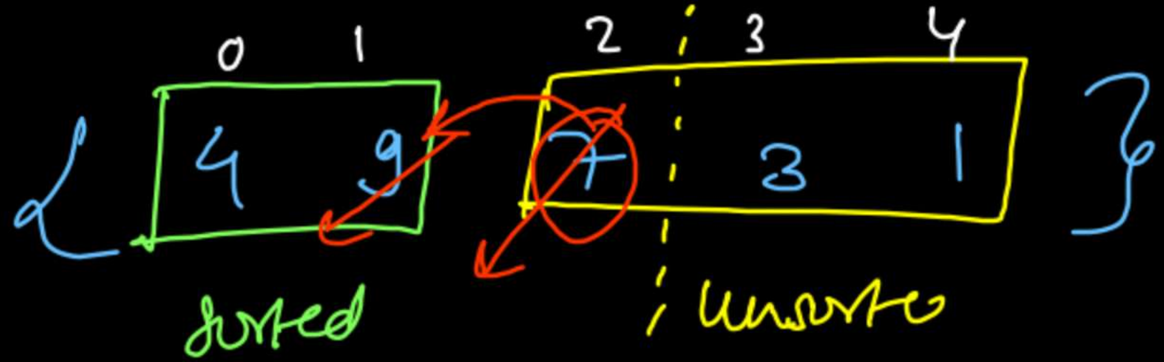
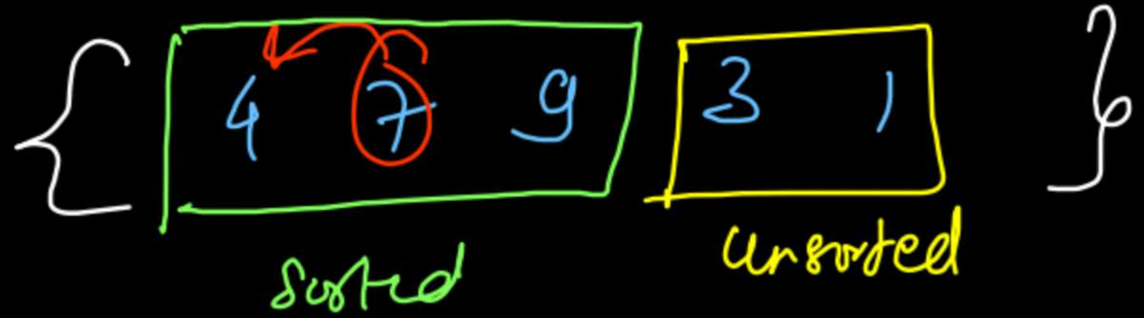
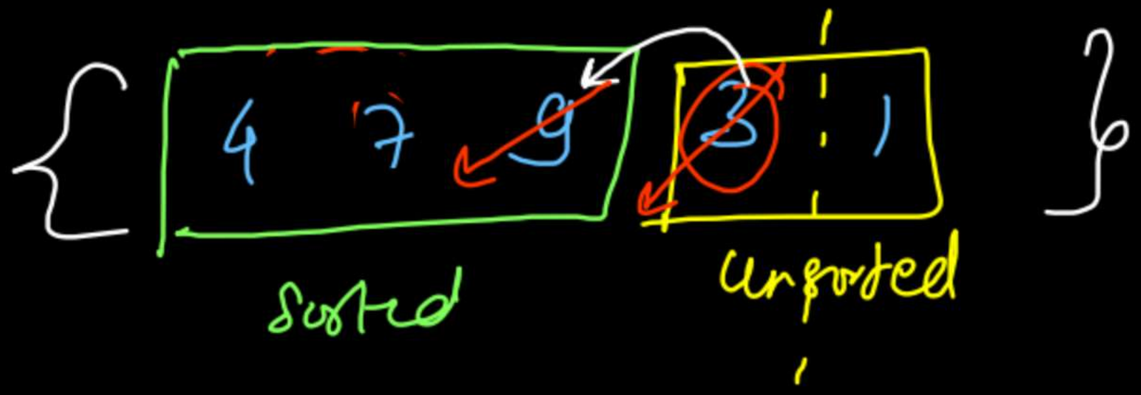
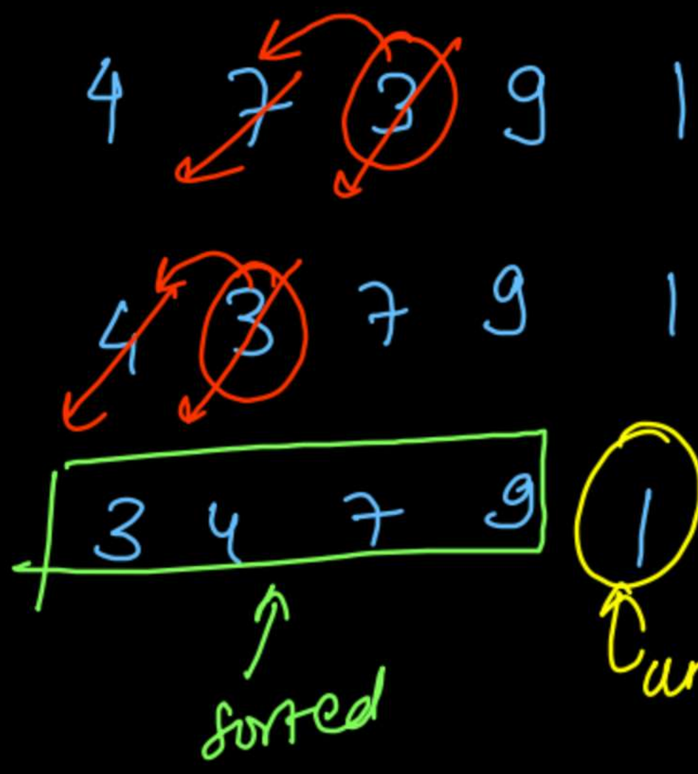
Insertion Sort

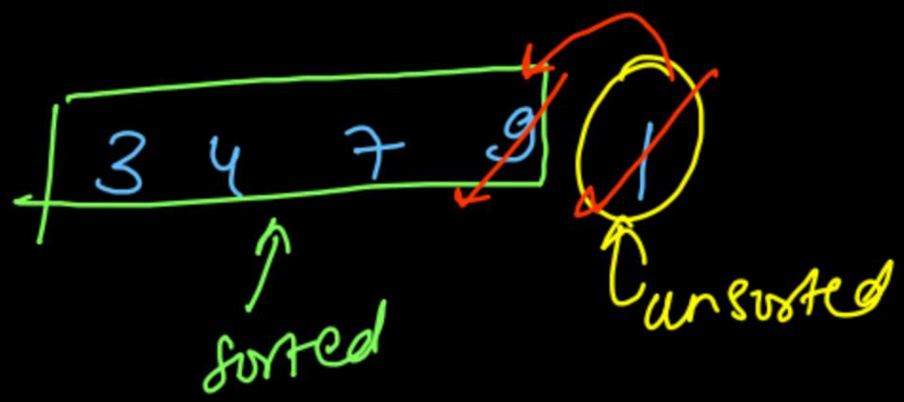


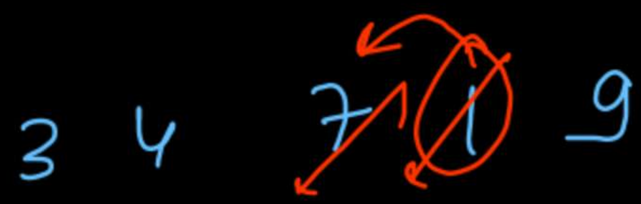


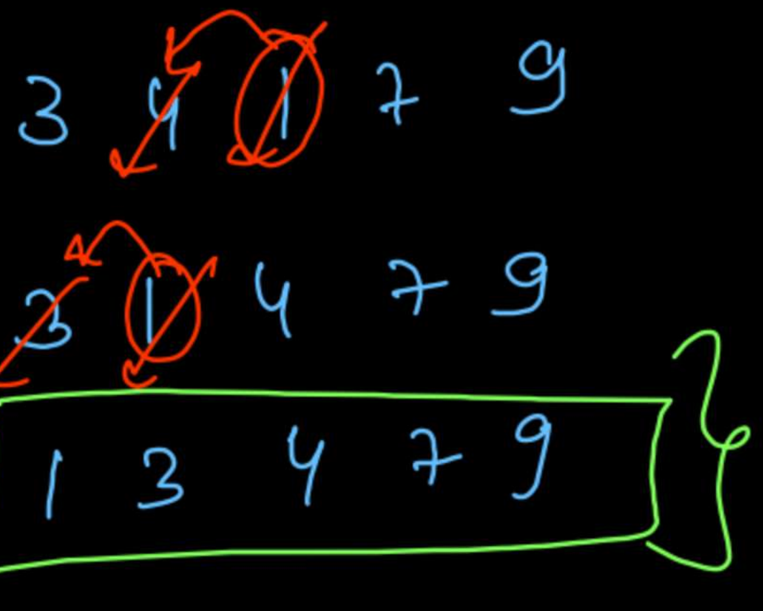



4 7 9 3 1
sorted
unsorted


4 7 3 9 1
4 3 7 9 1
3 4 7 9
1
sorted
unsorted


3 4 7 9 1
sorted
unsorted


3 4 7 1 9


3 4 1 7 9
3 1 4 7 9
1 3 4 7 9

```java
public void insertionSort(int arr[], int n)
{
    // Taking One Element From Unsorted Region and placing it
    // in the sorted region
    for(int i=1; i<n; i++){

        // Going to the left uptil you are smaller than
        // your left neighbour
        for(int j=i-1; j>=0; j--){
            if(arr[j] > arr[j + 1]){
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            } else break;
        }

    }
}
```

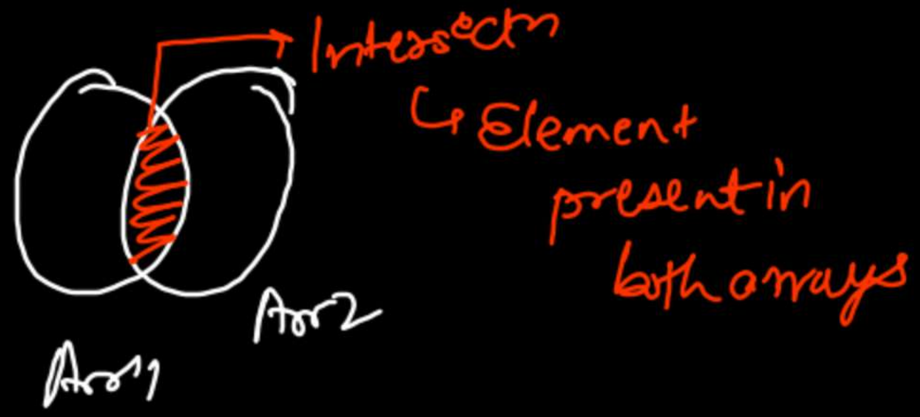
0.5 x

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}

< $O(N^2)$ ≥ $O(N\log N)$

MergeSort, QuickSort

Arr1 : { 10, 15, 20, 30, 45, 65, 70 }
ptr1

Arr2 : { 10, 17, 20, 23, 33, 40, 45, 65 }
ptr2

Sorting will be faster than the brute force

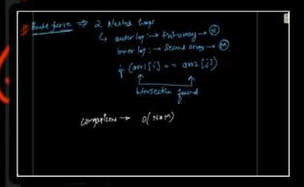
10, 20, 45, 65

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

O(N+M)



Merging 2 Sorted Arrays → O(N+M)

arr1 and arr2 are already sorted

Array (or) vector

arr1: { 10, 30, 50, 60, 70, 100, 110, 120, 120 }

arr2: { 20, 40, 80, 90, 110, 130, 140, 150 }

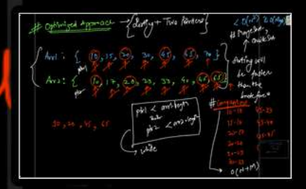
Input

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, 120, 130, 140, 150 }

output

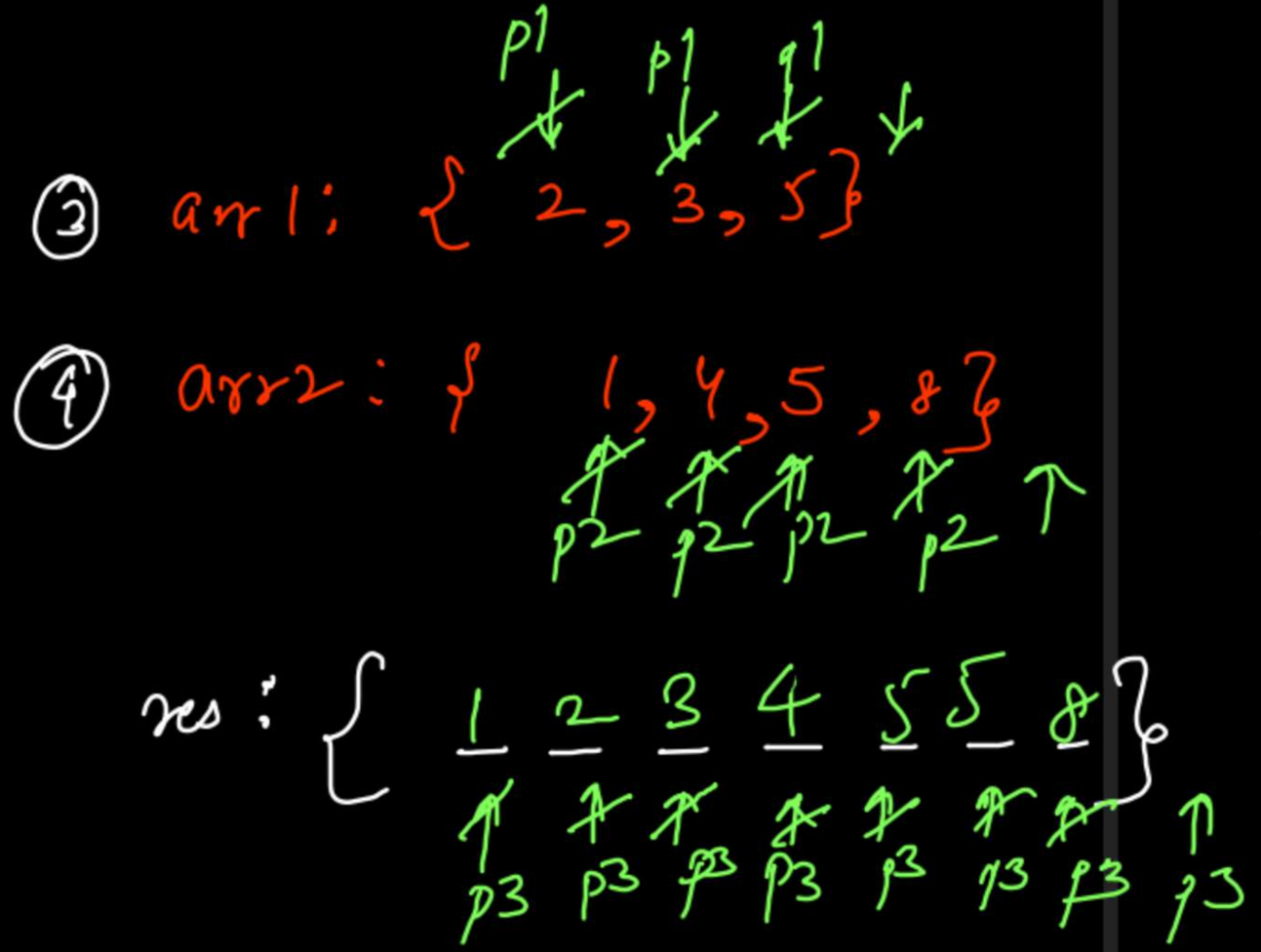


```
int ptr1 = 0, ptr2 = 0, ptr3 = 0;
int[] res = new int[arr1.length + arr2.length];

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++;
    }
}
```

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

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



Rotate Array {Left rotation}

k = 0, 7, 14, 21, 28

All multiples of 7

arr: { 10, 20, 30, 40, 50, 60, 70 }

k = 4, 11, 18

After 4th rotation

{ 50, 60, 70, 10, 20, 30, 40 }

k = 1, 8, 15, 22, 28 → All multiples of 7 + 1

After 1st rotation

{ 20, 30, 40, 50, 60, 70, 10 }

k = 5, 12, 19

After 5th rotation

{ 60, 70, 10, 20, 30, 40, 50 }

k = 2, 9, 16 → All multiples of 7 + 2

After 2nd rotation

{ 30, 40, 50, 60, 70, 10, 20 }

k = 6, 13

After 6th rotation

{ 70, 10, 20, 30, 40, 50, 60 }

k = 3, 10, 17 → All ..... + 3

After 3rd rotation

{ 40, 50, 60, 70, 10, 20, 30 }

Eg: size $n = 7$

Any variable C not rotating $k \in [0, \infty)$

$k = 0$ → 7, 14, 21, 28, ... = $7 * k \Rightarrow 0$

$k = 1$ → 8, 15, 22, 29 = $7 * k + 1 \Rightarrow 1$

$k = 2$ → 9, 16, 23, 30 = $7 * k + 2 \Rightarrow 2$

$k = 3$ → 10, 17, 24, 31 = $7 * k + 3 \Rightarrow 3$

$k = 4$ → 11, 18, 25, 32 = $7 * k + 4 \Rightarrow 4$

$k = 5$ → 12, 19, 26, 33 = $7 * k + 5 \Rightarrow 5$

$k = 6$ → 13, 20, 27, 34, = $7 * k + 6 \Rightarrow 6$

(rotations) % size

minimum rotations

Array Rotation { Left }

6*m+0

R=0, 6, 12, 18, 24

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

6*m+1

R=1, 7, 13

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

6*m+2

R=2, 8, 14

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

6*m+3

R=3, 9, 15

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

6*m+4

R=4, 10, 16

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

6m+5

R=5, 11, 17

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

Smaller Rotation for any

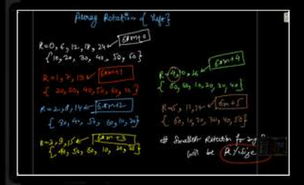
will be R%size

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

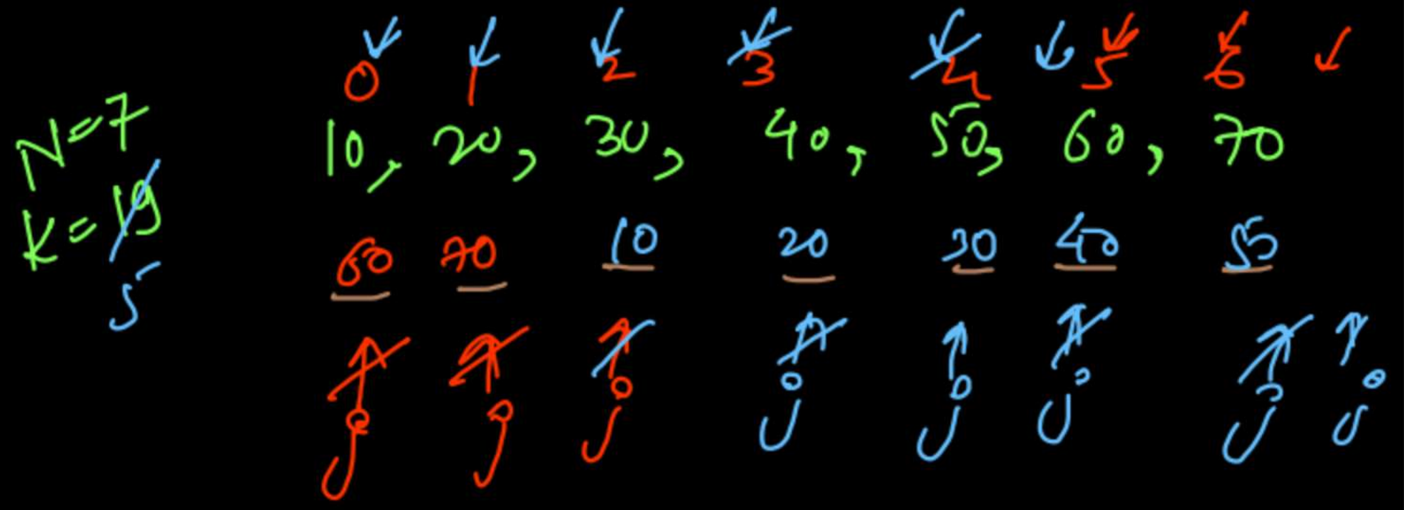


```
k = k % n; // Reduce the va

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++;
}
```



In-place

② Rotating the Array 1 place at a time

→ call this for (k) time

```
Rotate by 1 ( )
```

performing the algorithm on the input array

reduced value

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

k=1

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

k = 16 ⇒ 16 % 6 = ④

k % n

① Reduce the value of k

Rotate by 1 at k

② Store 0th index value in temp

③ Shift all indexes

④ Place temp at the last index

```java
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;
}
```

```java
k = k % n; /
```

```java
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)

N − k = 9 − 3 = 6

B

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

reverse(0, 8)

N − k = 6

B

k = 3

A

Output

40	50	60	70	80	90	10	20	30

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

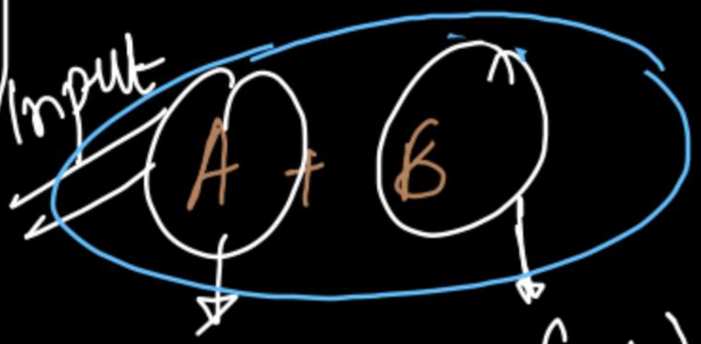
```
public static void reverse(int[] arr, int start, int end){
    while(start < end){
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++; end--;
    }
}
```

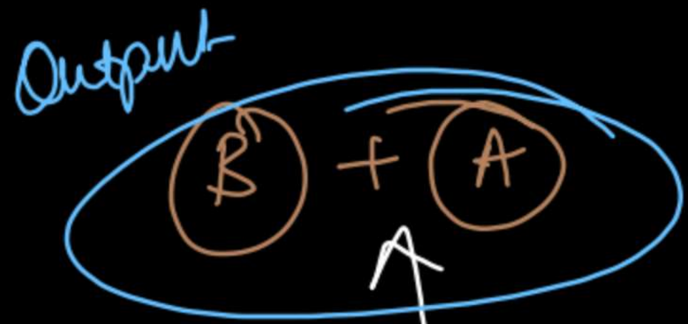
```
// 1. Reverse the starting k  elements
reverse(input, 0, k - 1);
// 2. Reverse the ending n - k elements
reverse(input, k, n - 1);
// 3. Reverse the entire array
reverse(input, 0, n - 1);

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

WHY

Mathematical Proof





$$(A^T + B^T)^T = (B^T)^T + (A^T)^T = B + A$$

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

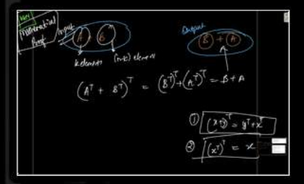
(17+14) ≠ 16

(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 end

(1+20) = ↓21 > 16

(1+17) = ↓18 > 16

(1+12) = ↑13 < 16

(3+12) = ↑15 < 16

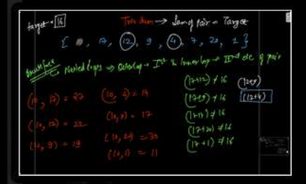
start ++ / end --

sum ↑ sum ↓

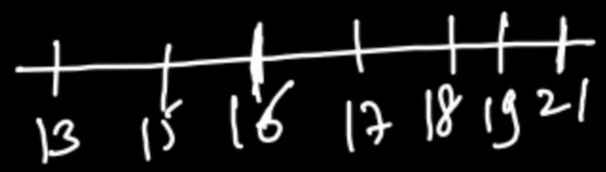
(7+12) = ↓19 > 16

(7+10) = 17 > 16

(7+9) = 16 = 16



2 Sum

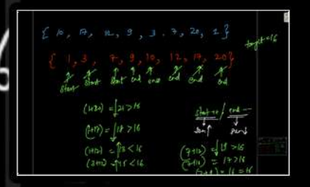


target = 16

{ 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;
```

20 > 16
1+17 > 16
1+12 < 16
3+12 < 16
7+12 > 16
7+10 > 16
7+9 = 16



28 m

{ 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

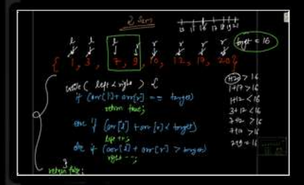
7+12 = 19 > 16

7+10 = 17 > 16

7+8 = 15 < 16

unsuccessful search

because there is no pair with distinct element summing to target



1+16 > 16
1+15 = 16
2+15 > 16
2+14 = 16
4+14 > 16
4+12 = 16
5+12 > 16
5+9 < 16

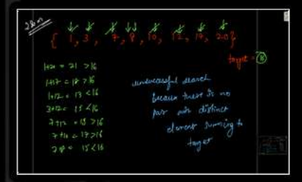
2 Sum → All the pairs equal to the target

{ 1, 2, 4, 5, 7, 8, 9, 12, 14, 15, 16 }

target = 16

(1, 15) (4, 12)
(2, 14) (7, 9)

```
while ( left < right ) {
  if (arr[l] + arr[r] == target)
      Syso(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 }

target = 30

First No + pair with target = 29

(1) → (10, 19), (12, 19)

(2) → (8, 19), (10, 17)

First No + pair with target = 27

(3) → 8 + 10 + 12 = 30

arr[i] + arr[j] + arr[k] == target

triplet(i, j, k) i ≠ j ≠ k

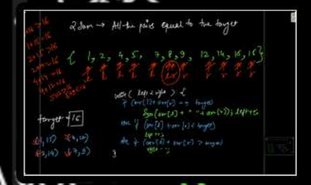
2 Sum → TP → 1 loop

3 Sum → TP → one

for 2 loops → 3 loops

for 3 loops → 4 loops

4 loops → 3 loops



ent has just been saved...

Hint

Sorted Squares of Sorted Array

Naive < Sort < 2 loops

↑ Two pointer

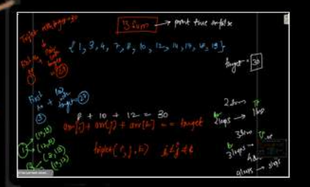
left is moving to left
right is moving to right

-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



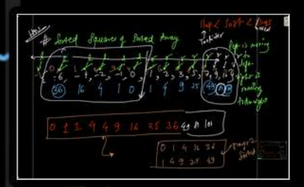
Frequency Array

array of characters → it will contain only lowercase english alphabets

Input :→ { 'g', 'e', 'e', 'k', 's', 't', 'e', 's', 'a', 'r', 'c', '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, ...

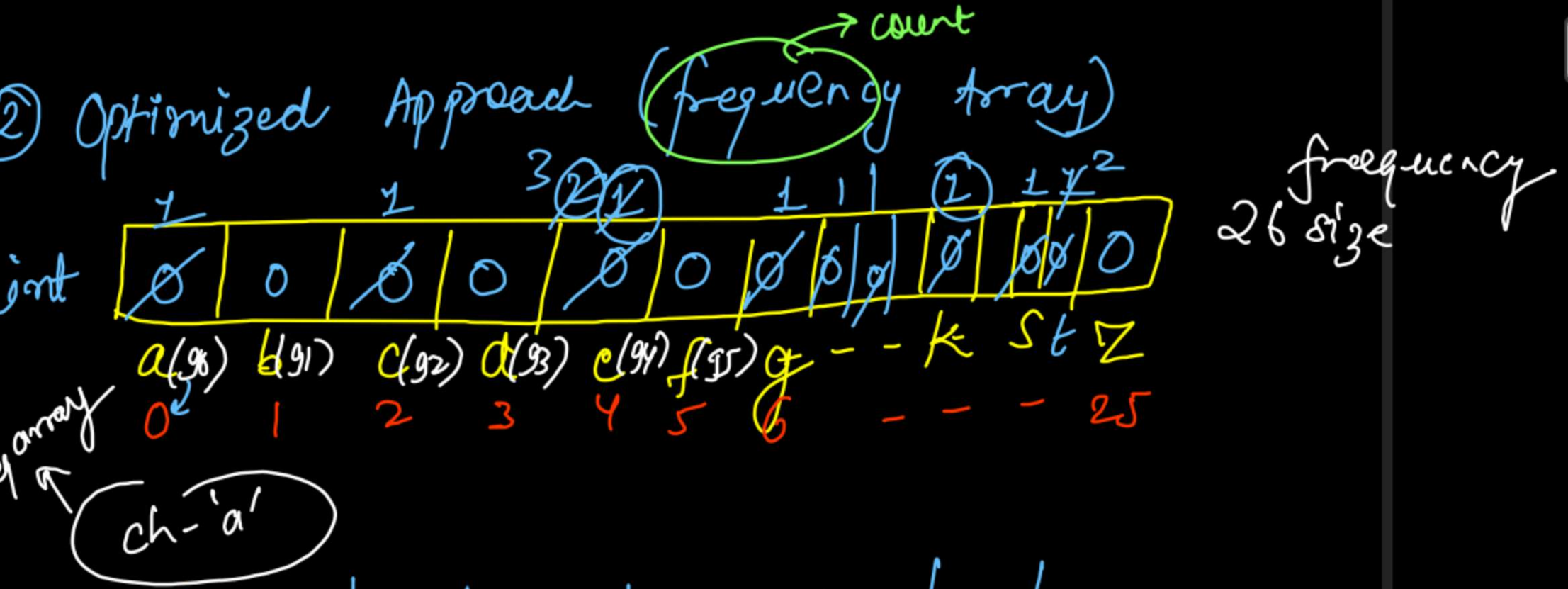


```java
// Brute Force
// Outer Loop: Character
for(char i='a'; i<='z'; i++){

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

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

② Optimized Approach (frequency array)



```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] + " ");
}
```

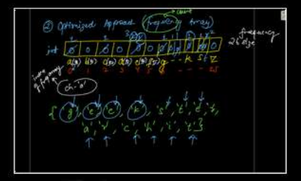
O(N)

+

O(26)

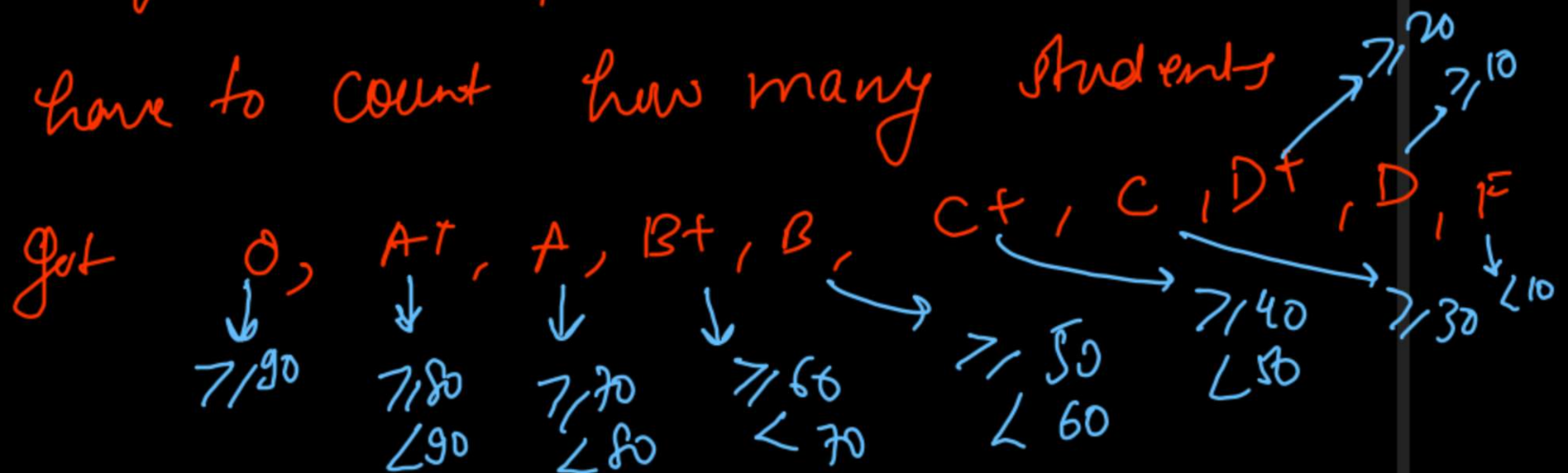
O(N + 26)

These are not nested loops



Eg 2 There are N students. Each students have got marks from 0 to 100.

You have to count how many students got



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

countof O = 2, countof A+ = 1, countof 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)

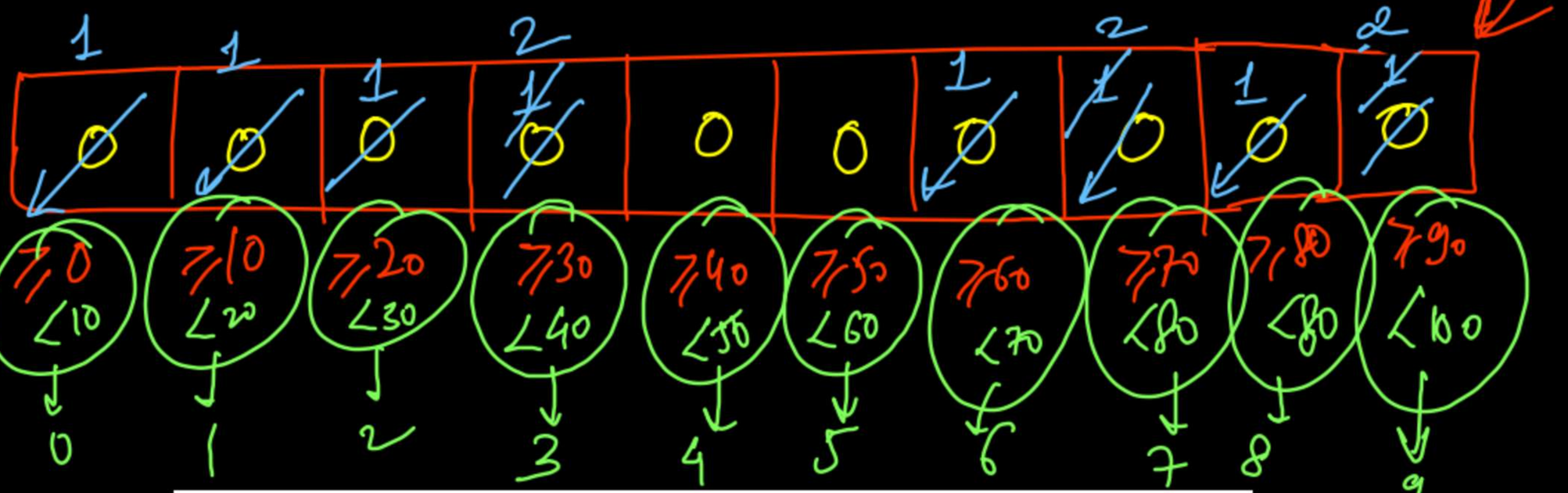
```
Finished in 125 ms
 >= 90 -> 2
 >= 80 -> 1
 >= 70 -> 2
 >= 60 -> 1
 >= 50 -> 0
 >= 40 -> 0
 >= 30 -> 2
 >= 20 -> 1
 >= 10 -> 1
 >= 0 -> 1
```

stdin

```
12
64 20 34 92 97 88 73 75 0 33 10 100
```

Approach 2: Frequency Array

types of grades



11

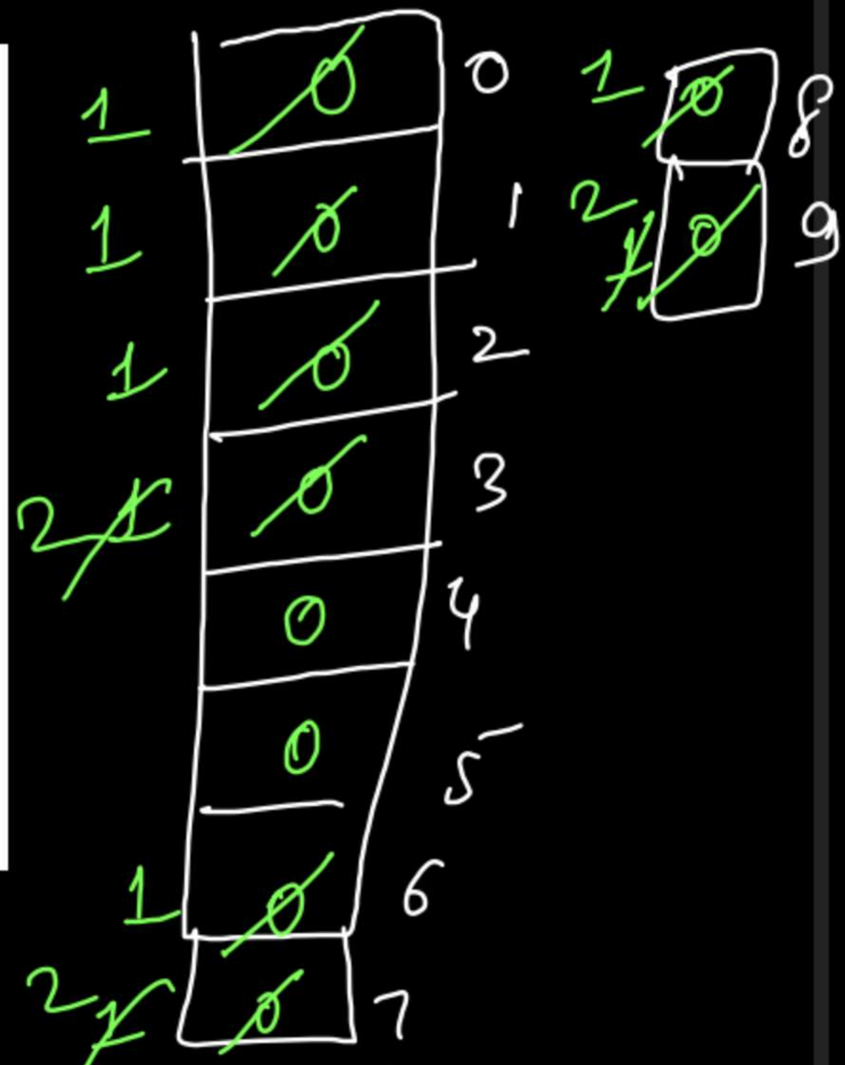
64 20 34 92 97 88 73 75 0 33 10 ~~100~~

64/10 =6 20/10 =2 34/10 =3 92/10 =9 97/10 =9

```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

0 → 0 1 2 3 4 5 6 7 8 9 — >=0, <10

1 → 10 11 12 13 14 15 16 17 18 19 — 10*1+K — >=10, <20

2 → 20 21 22 23 24 25 26 27 28 29 — >=20, <30

30

39

40

49

51

59

61

69

71

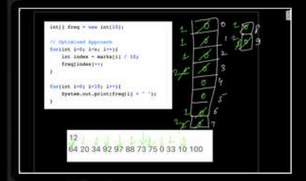
79

81

89

91

99



Array

Sum of all ranges

subarray $[l \le r]$

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

0

[1,1] = 12 ✓

[1,2] = 12+8 ✓

[1,3] = 12+8+16 = 36 ✓

[1,4] = 12+8+16+2 = 38 ✓

[2,2] = 8

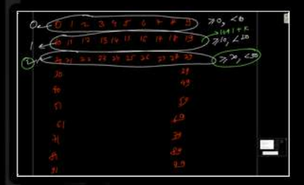
[2,3] = 8+16 = 24

[2,4] = 8+16+2 = 26

[3,3] = 16

[3,4] = 16+2 = 18

[4,4] = 2



Sort 01

```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);
    }
}
```

Brute force

↓

Sum of all the ranges

Diagonal Traversal

N=5

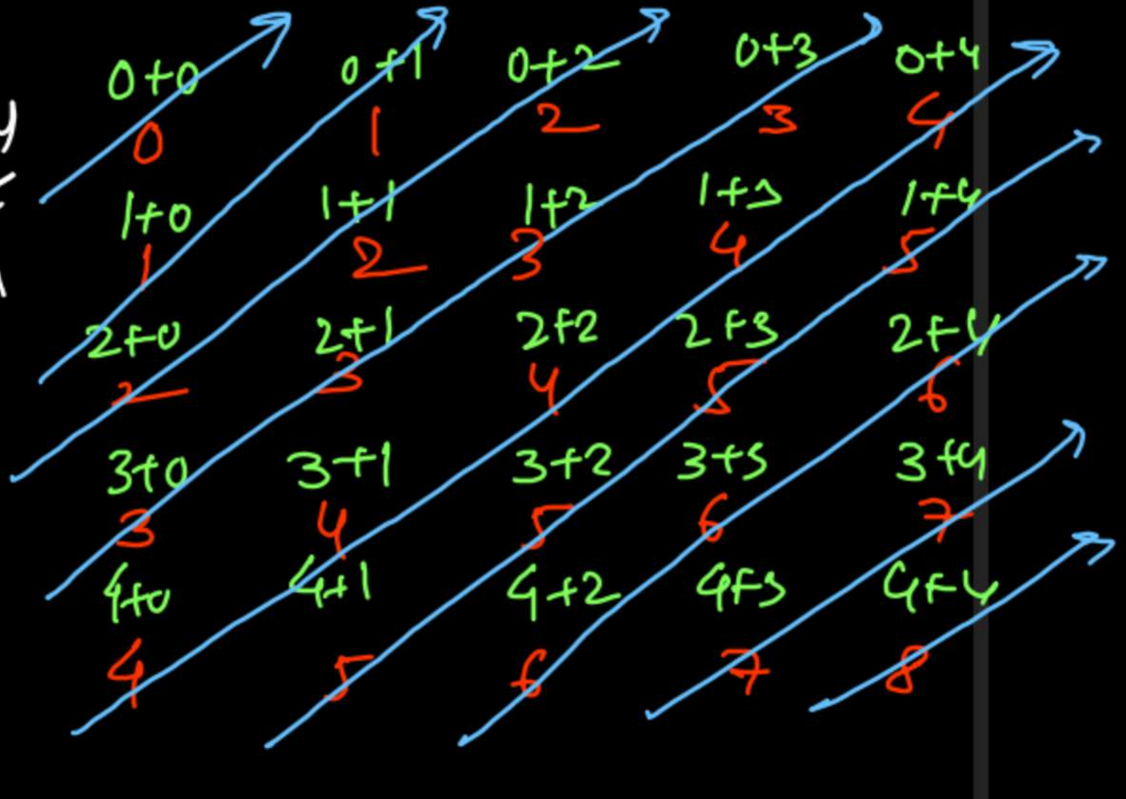
col →

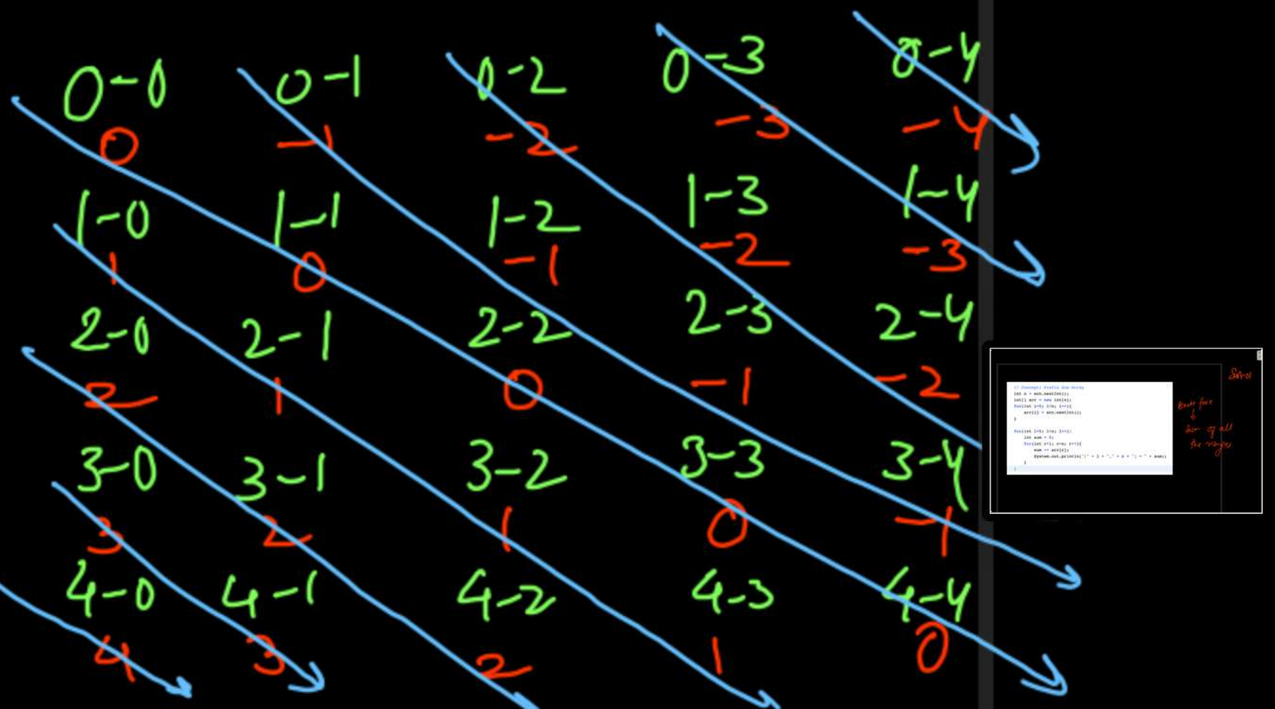
00	01	02	03	04
10	11	12	13	14
20	21	22	23	24
30	31	32	33	34
40	41	42	43	44

row ↓

Right diagonals → Sum of row & col

Diff of row & col

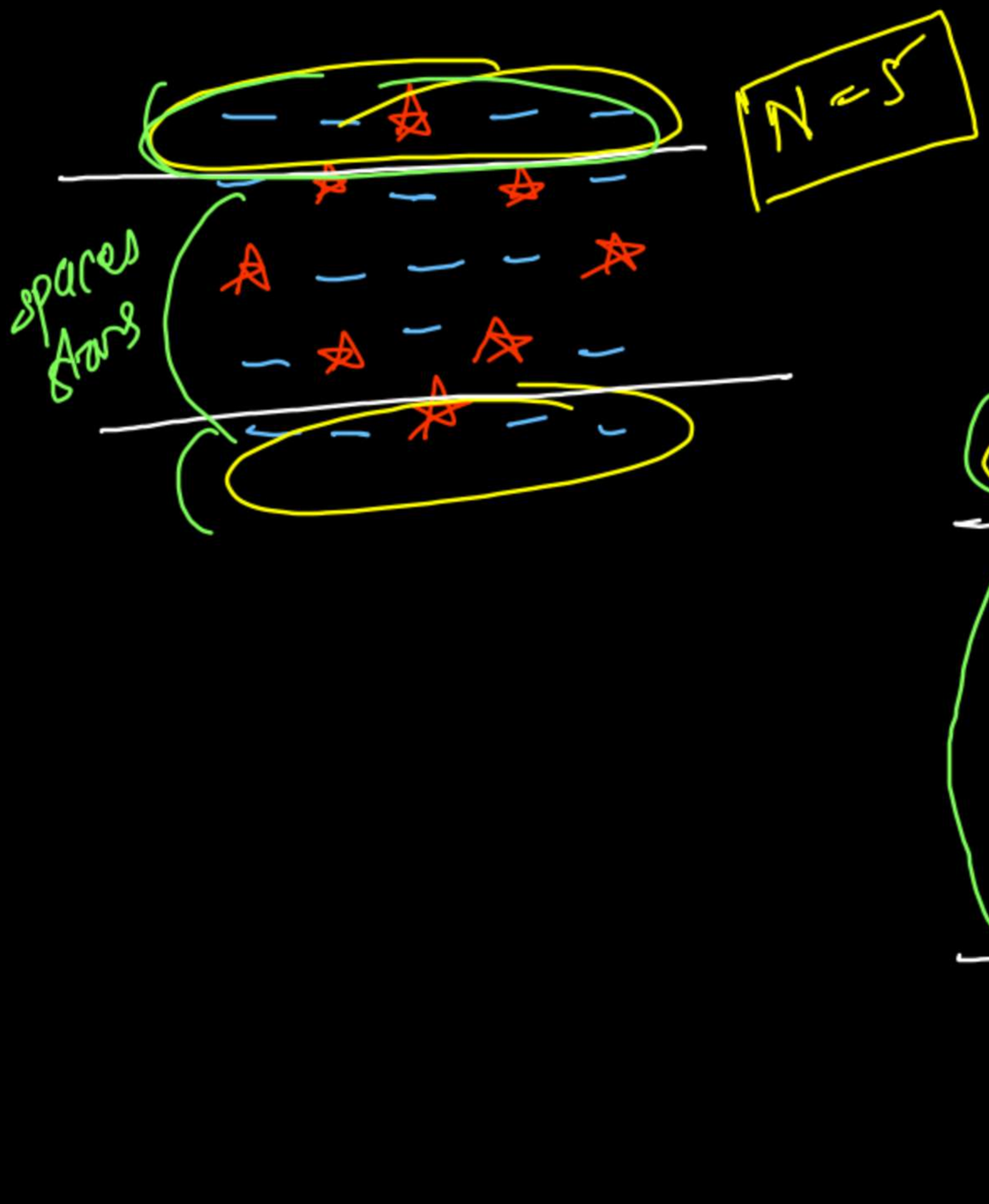


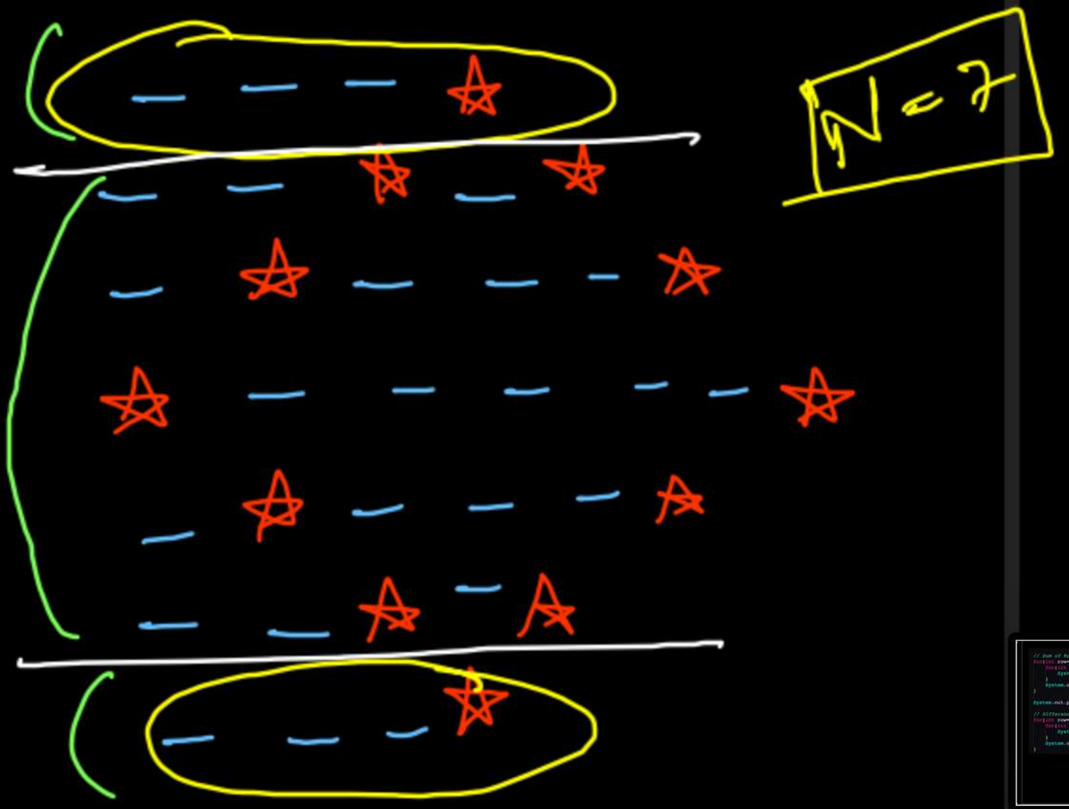


```java
// Sum of Row & Col Will Give Equal Elements on Right Diagonals
for(int row=0; row<n; row++){
    for(int col=0; col<n; col++){
        System.out.print(row + col + " ");
    }
    System.out.println();
}

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

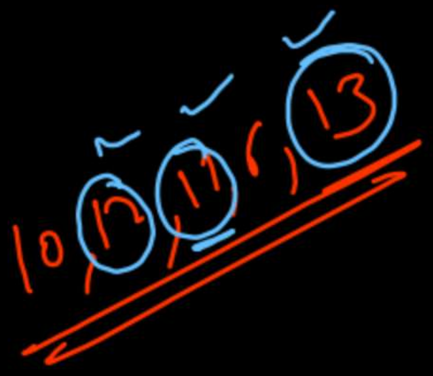

N=7

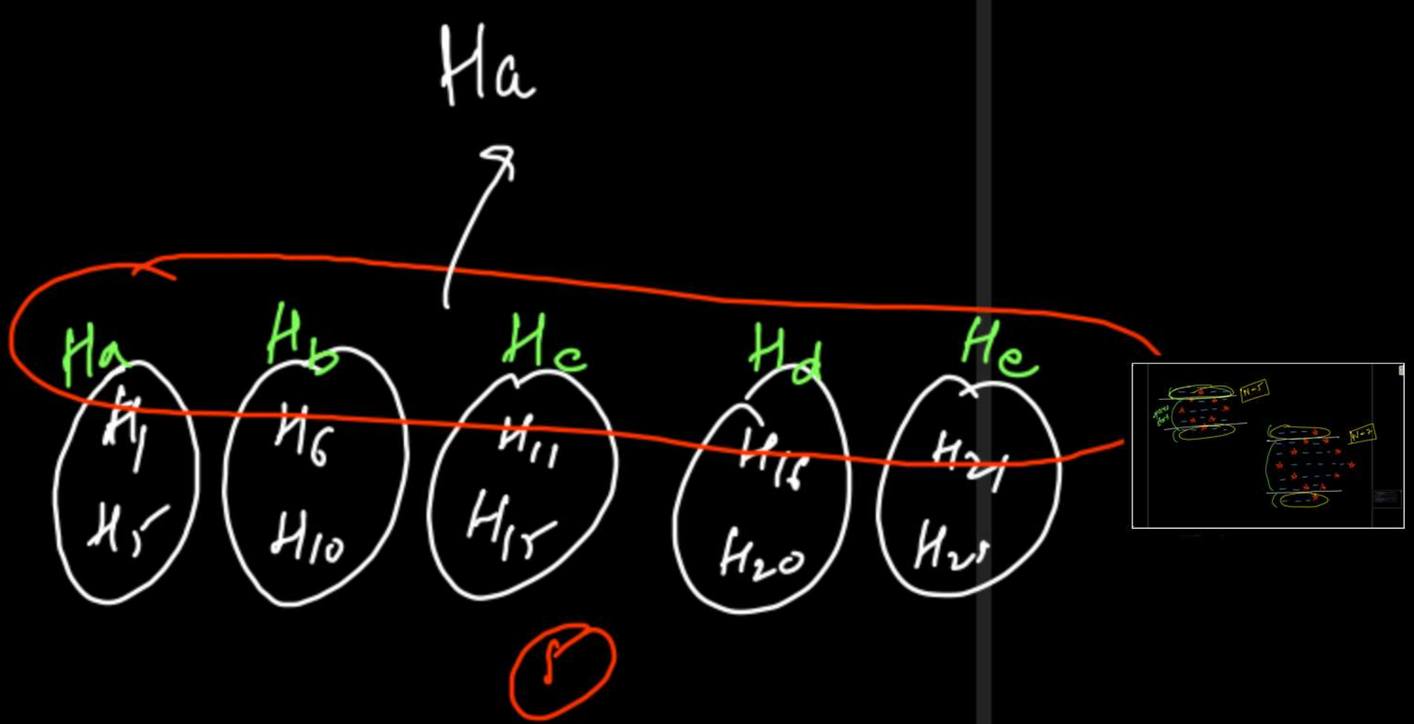
25 horses

In one race → 5 horses can participate

3 fastest ones ↳ fastest horse

Minimum races





Target Sum Pair (2 Sum) → Closest Target

Closest ⇒ 50 52 55 53

target ⇒ 54

10, 22, 25, 28, 30, 40

l r l r

10 + 40 < 54

22 + 40 > 54

22 + 30 = 52 < 54

25 + 30 = 55 > 54

25 + 28 = 53 < 54

|5| = 5

|-3| = 3

|+∞| = ∞

|-∞| = -∞