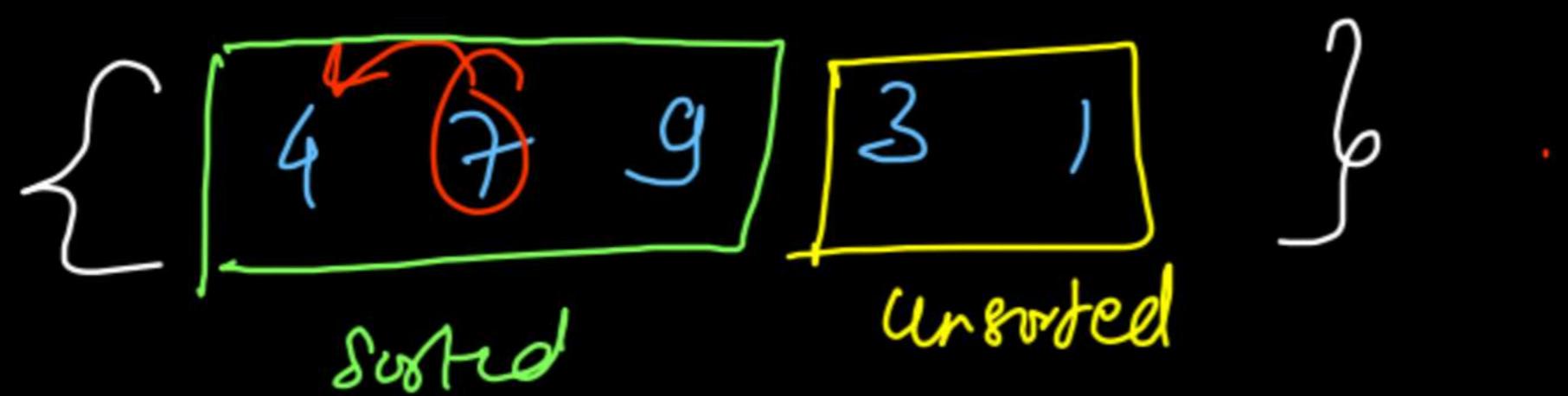
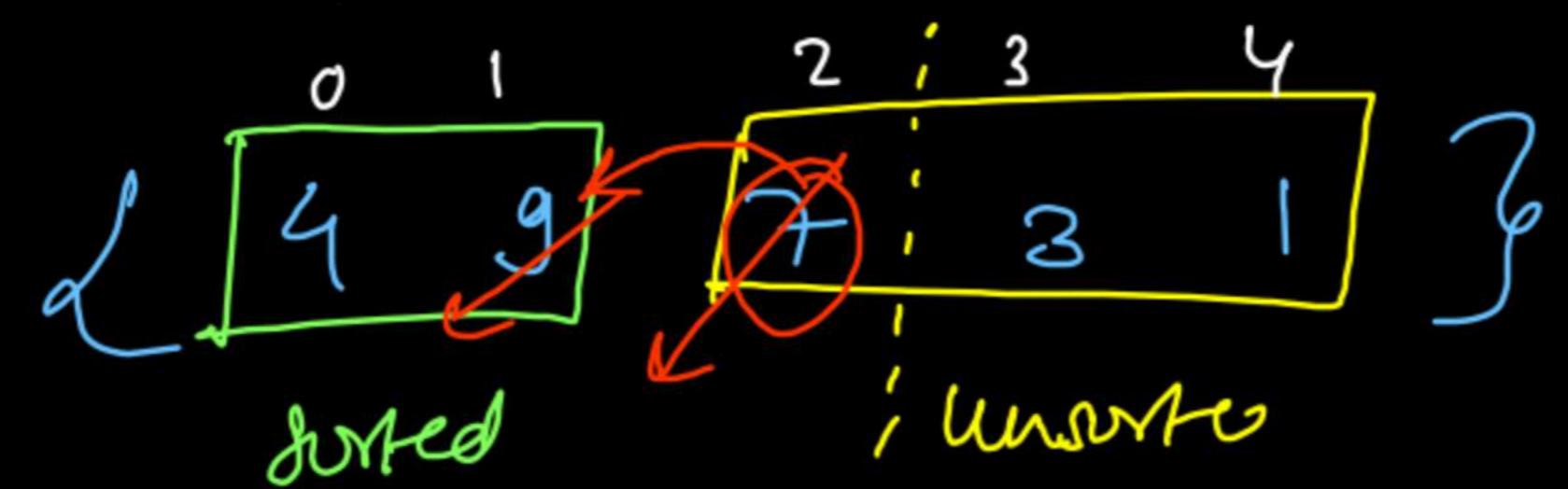
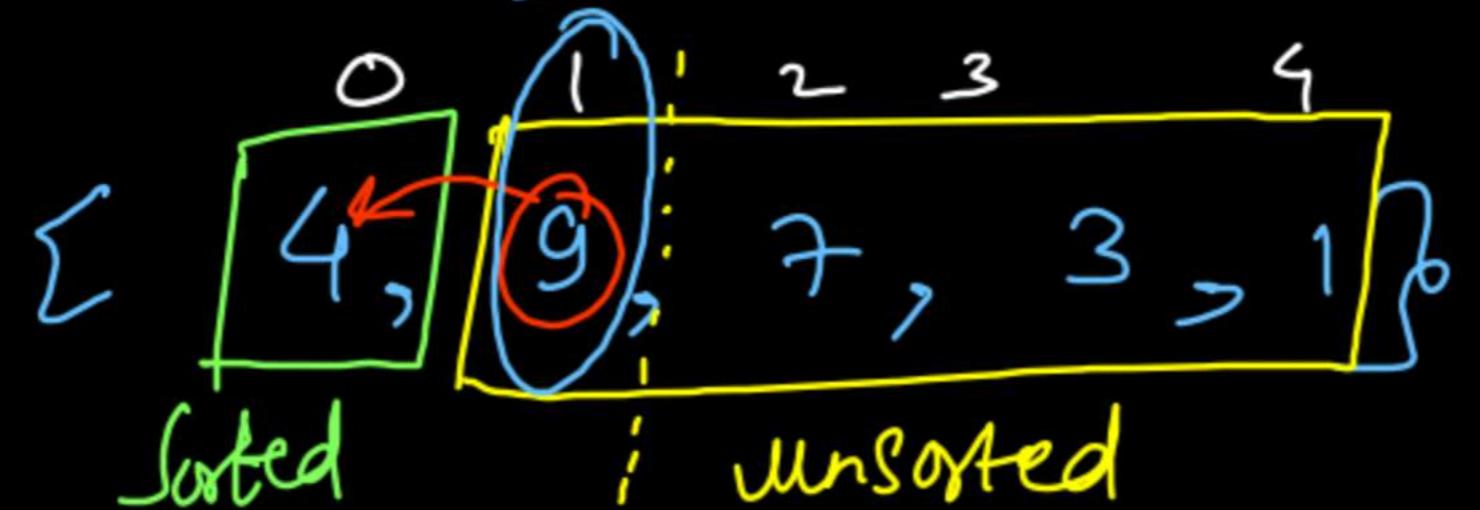
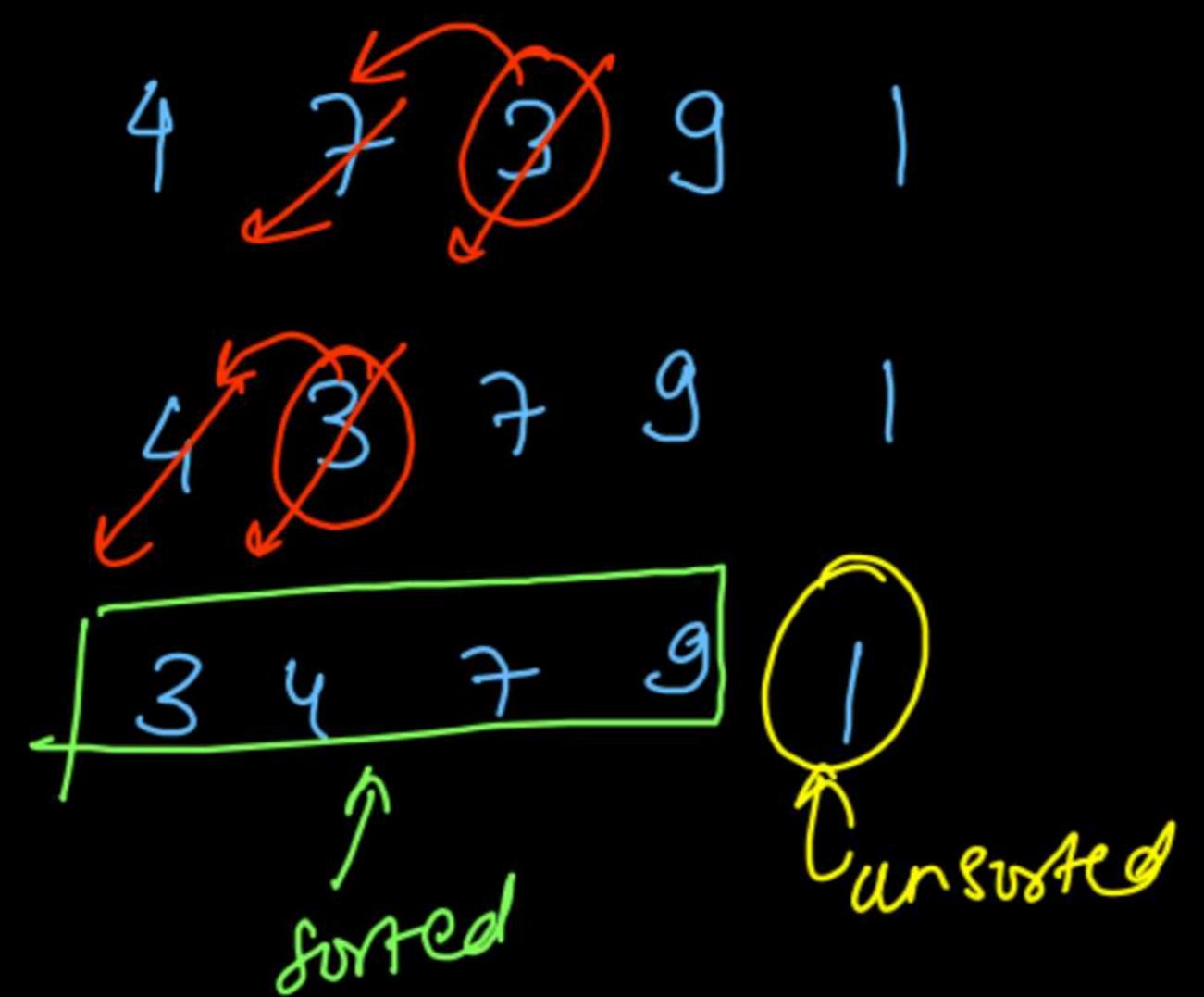
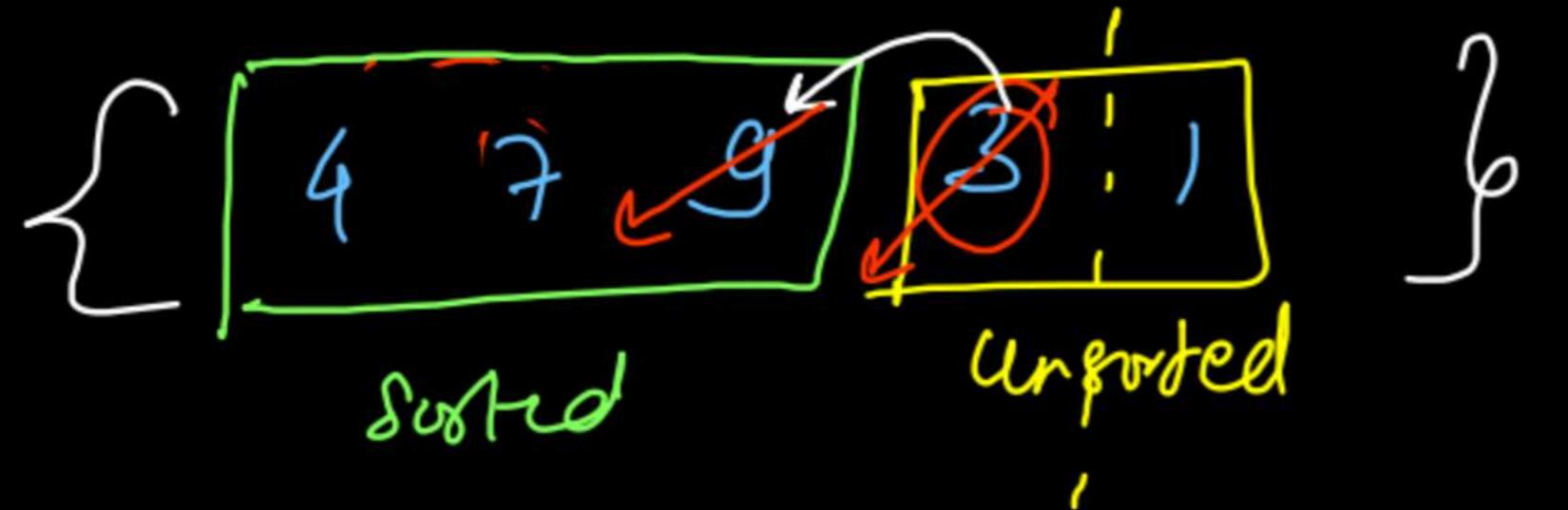
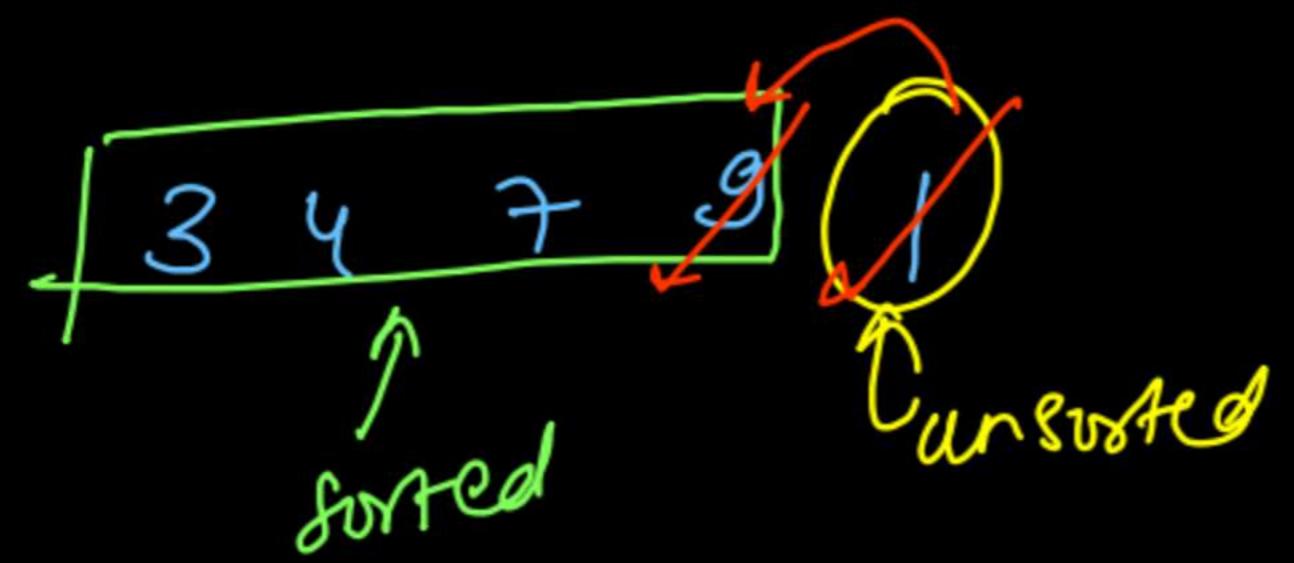


## Insertion Sort



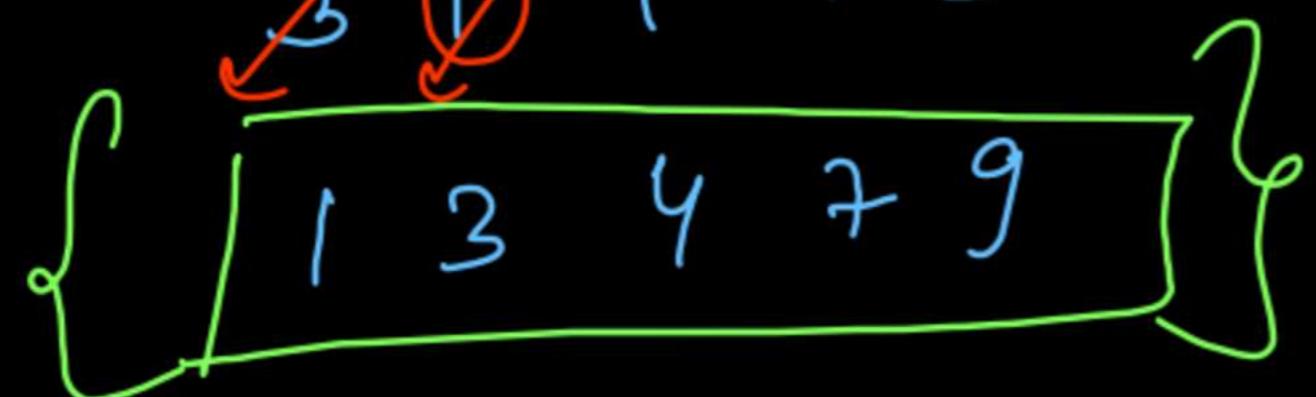




~~3 4 7 10 9~~

~~3 4 10 7 9~~

~~2 10 4 7 9~~



```
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 upto 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

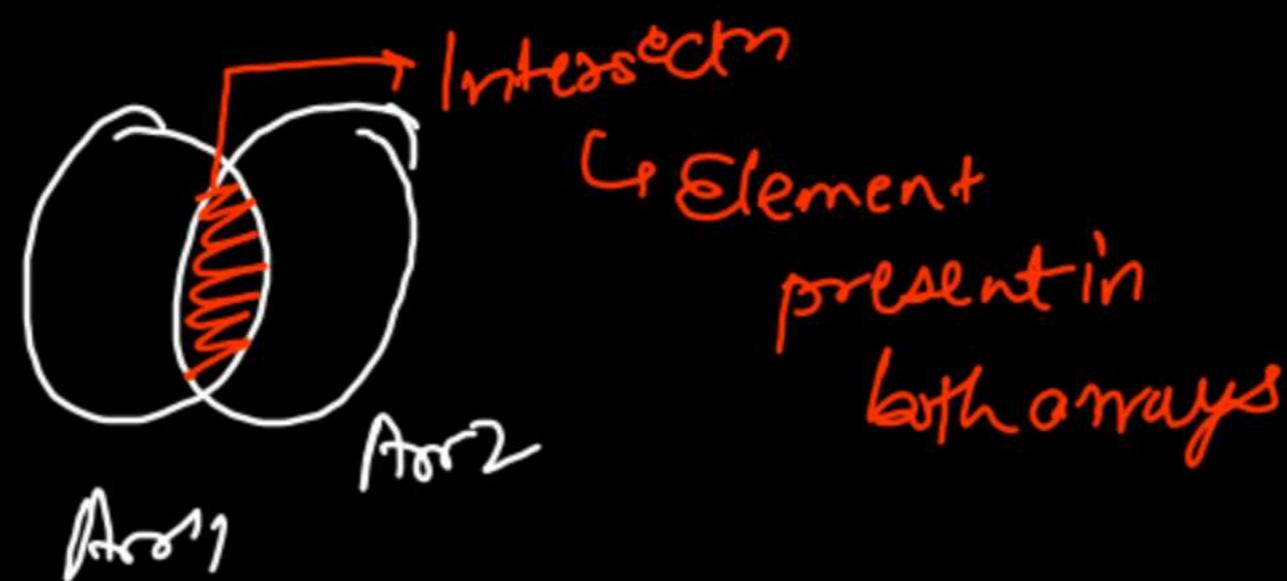
# # Intersection of 2 Arrays

# constraint

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

Each array contains  
distinct elements

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



{ 10, 45, 20, 65 }

→ Answer order  
does not matter

# Brute force  $\Rightarrow$  2 nested loops  
↳ outer loop:  $\rightarrow$  first array  $\rightarrow N$   
inner loop:  $\rightarrow$  Second array  $\rightarrow M$

$\text{if } (\text{arr1}[i] == \text{arr2}[j])$

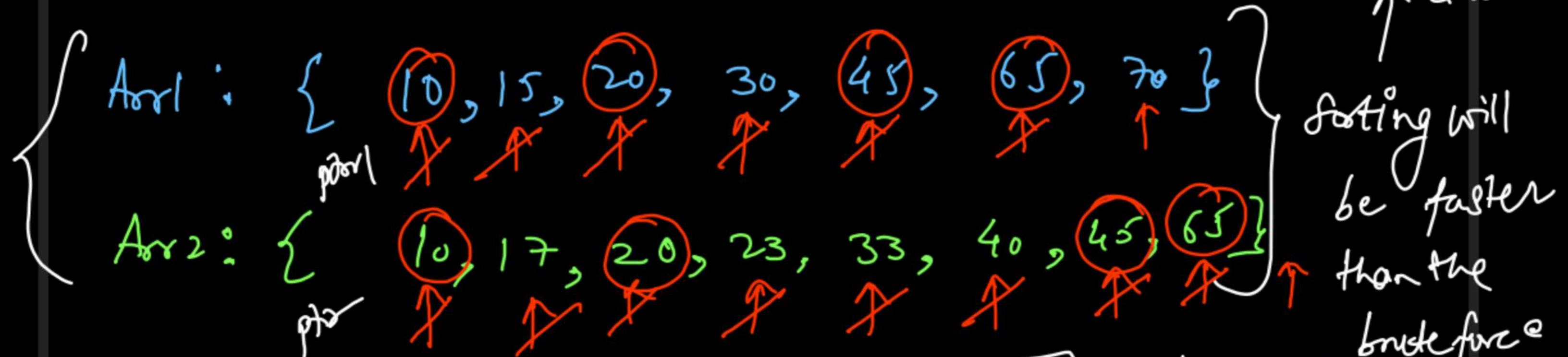
$\uparrow$                              $\uparrow$   
*intersection found*

Comparisons  $\rightarrow O(N * M)$

# Optimized Approach → {Sorting + Two Pointers}

<  $O(N^2) \approx O(N \log N)$

# MergeSort,  
↑ QuickSort



10, 20, 45, 65

$pt1 < arr1.length$   
20  
 $pt2 < arr2.length$   
while

# Comparisons

10-10 45-33

15-17 45-40

20-17 45-45

20-20

30-23

30-33

$O(N+M)$

Brute force  $\Rightarrow 2$  nested loops  
Outer loop  $\rightarrow$  First array  $\rightarrow \Theta$   
Inner loop  $\rightarrow$  Second array  $\rightarrow \Theta$   
 $\therefore O(N \cdot M) = O(M \cdot N)$   
Intersection found  
Comparison  $\rightarrow O(1)$

#  
arr1 and arr2 sorted  
already  
one  
array  
(min)  
arr2

# Merging 2 Sorted Arrays  $\underbrace{\qquad\qquad\qquad}_{O(N+M)}$

# modified array  
should also  
be freed

{ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100,  
110, 110, 120, 120, 130, 140, 150 } } Output

res:  $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 110, \cancel{120}, \cancel{120}, \cancel{120}, 130, 140, 150\}$

# Optimized Approach → Divide & Conquer

Area 1: { 10, 15, 20, 30, 40, 50 }  
 Area 2: { 1, 2, 5, 8, 20, 25, 30, 40, 50, 60 }

$\text{pivot} = \frac{\text{min}(A_1) + \text{max}(A_2)}{2}$   
 $\text{pivot} = \frac{10 + 60}{2}$   
 $\text{pivot} = 35$

while

divide will be faster than the break function

```

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++;
    }
}

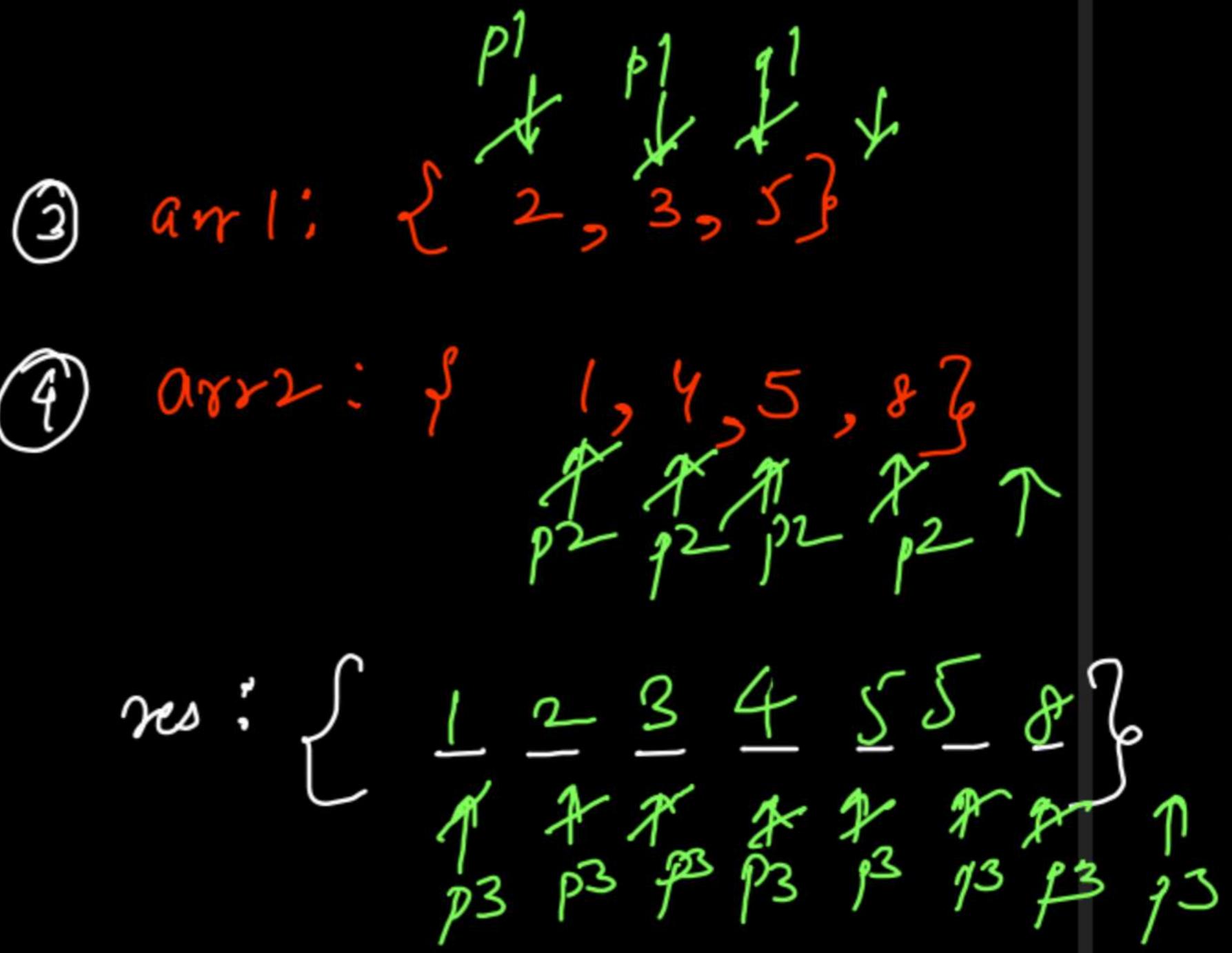
```

```

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

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

```



# Rotate Array {Left rotation}

Initial Array  $k=0, 7, 14, 21, 28$   
 $\text{arr} = \{10, 20, 30, 40, 50, 60, 70\}$

All multiples of 7

After 4th rotation  $k=4, 11, 18$

After 1st rotation  $k=1, 8, 15, 22, 29$  → All multiples of 7 + 1

$\{56, 60, 70, 10, 20, 30, 40\}$

$\{20, 30, 40, 50, 60, 70, 10\}$   
 After 2nd rotation  $k=2, 9, 16$  → All multiples of 7 + 2

After 5th rotation  $k=5, 12, 19$

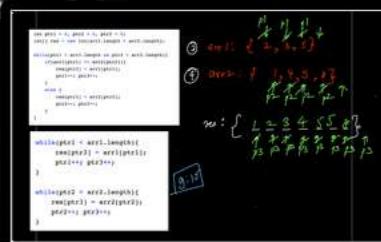
$\{30, 40, 50, 60, 70, 10, 20\}$

$\{60, 70, 19, 20, 30, 40, 50\}$

After 3rd rotation  $k=3, 10, 17$  → A '--- + 3

After 6th rotation  $k=6, 13$   
 $\{20, 10, 20, 30, 40, 50, 60\}$

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



~~Eg size~~  $n=7$

Any variable (not rotation)  
 $k \in [0, \infty)$

$$R = \textcircled{3} \quad 7, 14, 21, 28, \dots = 7 * k \Rightarrow 0$$

$$k = \textcircled{1} \quad 8, 15, 22, 29 = 7 * k + 1 \Rightarrow 1$$

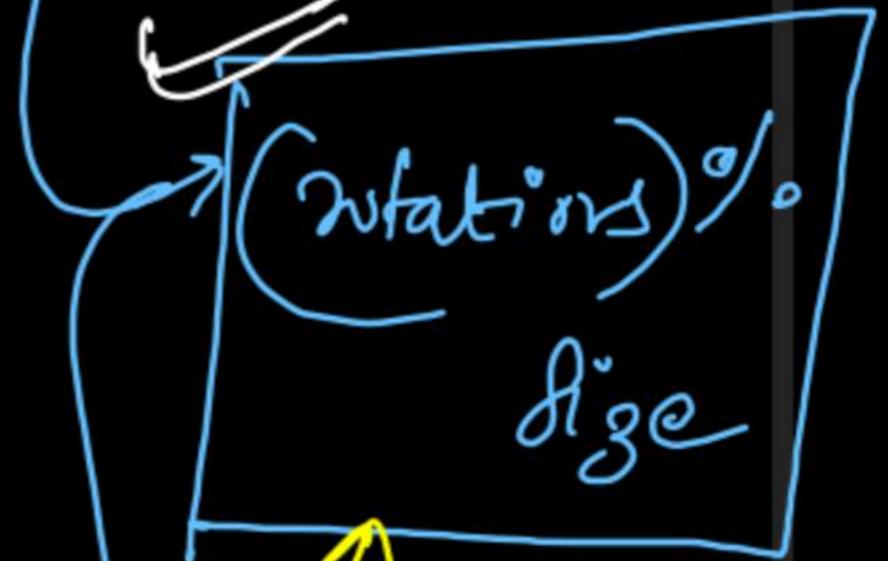
$$k = \textcircled{2}, 9, 16, 23, 30 = 7 * k + 2 \Rightarrow 2$$

$$k = \textcircled{3}, 10, 17, 24, 31 = 7 * k + 3 \Rightarrow 3$$

$$k = \textcircled{4}, 11, 18, 25, 32 = 7 * k + 4 \Rightarrow 4$$

$$k = \textcircled{5}, 12, 19, 26, 33 = 7 * k + 5 \Rightarrow 5$$

$$k = \textcircled{6}, 13, 20, 27, 34, = 7 * k + 6 \Rightarrow 6$$



Initial Array (Left to right)	Rotations	Final Array
{1, 2, 3, 4, 5, 6, 7}	0	{1, 2, 3, 4, 5, 6, 7}
{1, 2, 3, 4, 5, 6, 7}	1	{2, 3, 4, 5, 6, 7, 1}
{1, 2, 3, 4, 5, 6, 7}	2	{3, 4, 5, 6, 7, 1, 2}
{1, 2, 3, 4, 5, 6, 7}	3	{4, 5, 6, 7, 1, 2, 3}
{1, 2, 3, 4, 5, 6, 7}	4	{5, 6, 7, 1, 2, 3, 4}
{1, 2, 3, 4, 5, 6, 7}	5	{6, 7, 1, 2, 3, 4, 5}
{1, 2, 3, 4, 5, 6, 7}	6	{7, 1, 2, 3, 4, 5, 6}

minimum rotations

# Array Rotation ⌈ Left ⌉

$R = 0, 6, 12, 18, 24 \leftarrow [6 * m + 0]$

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

$R = 1, 7, 13 \leftarrow [6 * m + 1]$

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

$R = 2, 8, 14 \leftarrow [6 * m + 2]$

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

$R = 3, 9, 15 \leftarrow [6 * m + 3]$

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

$R = 4 \leftarrow [6 * m + 4]$

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

$R = 5, 11, 17 \leftarrow [6 * m + 5]$

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

# Smaller Rotation for any  
will be  $R / \text{size}$

Key Points	
$R = 0$	$2, 10, 21, 22, \dots$
$R = 1$	$8, 15, 16, 17, 18, \dots$
$R = 2$	$9, 16, 23, 20, \dots$
$R = 3$	$10, 17, 24, 21, \dots$
$R = 4$	$11, 18, 25, 22, \dots$
$R = 5$	$12, 19, 26, 23, \dots$
$R = 6$	$13, 20, 27, 24, \dots$

① Brute force  $\Rightarrow$  By taking another array of same size

Taking another array  $\rightarrow$

- Reduce the value of  $k$  by  $k/\text{size}$
- First skipping the starting  $k$  elements
- Store the remaining elements
- Store the starting  $k$  elements

② Rotating 1 one by one

## ② Reversal algorithm

Algebra Rotation of Vectors

$R = 0, \pi/2, \pi, 3\pi/2, 2\pi$        $\sqrt{6+5}$

 $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ 

$R = 0, \pi/2, \pi, 3\pi/2, 2\pi$        $\sqrt{6+5}$

 $\{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ 

$R = 2\pi/3, 4\pi/3$        $\sqrt{6+5}$

 $\{3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ 

$R = 2\pi/3, 4\pi/3$        $\sqrt{6+5}$

 $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ 

# Smaller Rotation for  $\frac{\pi}{3}$  degree

```
① k = k % n; // Reduce the va  
② int[] output = new int[n];  
int j = 0;  
  
③ for(int i=k; i<n; i++){  
    output[j] = input[i];  
    j++;  
}  
  
for(int i=0; i<k; i++){  
    output[j] = input[i];  
    j++;  
}  
}
```

$N=7$   
 $k=13$   
 $j=5$

0 ↘ 1 ↗ 2 ↙ 3 ↖ 4 ↗ 5 ↙ 6 ↘ 7 ↗

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

60 20 10 20 20 40 50

X X X X X X X X

10 20 30 40 50 60 70

8:20

In-place  
② Rotating the Array 1 place at a time  
↓  
performing the  
algorithm  
or the  
input array

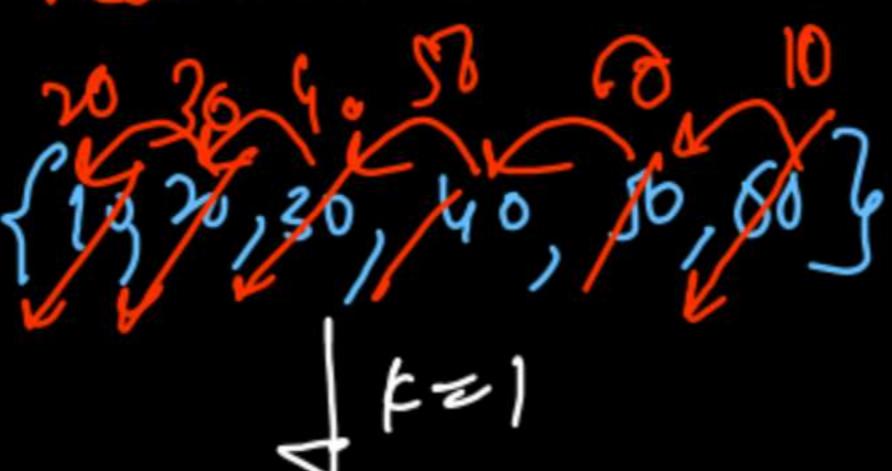
call this fn  $\text{③}$  times

Rotate by 1 ()

$$k = 16 \Rightarrow 16 \div 6 = ④$$

$\frac{k}{n}$

reduced value



eg

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

- ① Reduce the value of  $k$
- ② Place 0th index value in temp
- ③ Shift all indices
- ④ Place temp at the  $k^{th}$  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 }

↓ rotate by 1()

{ 30, 40, 50, 10, 20 }

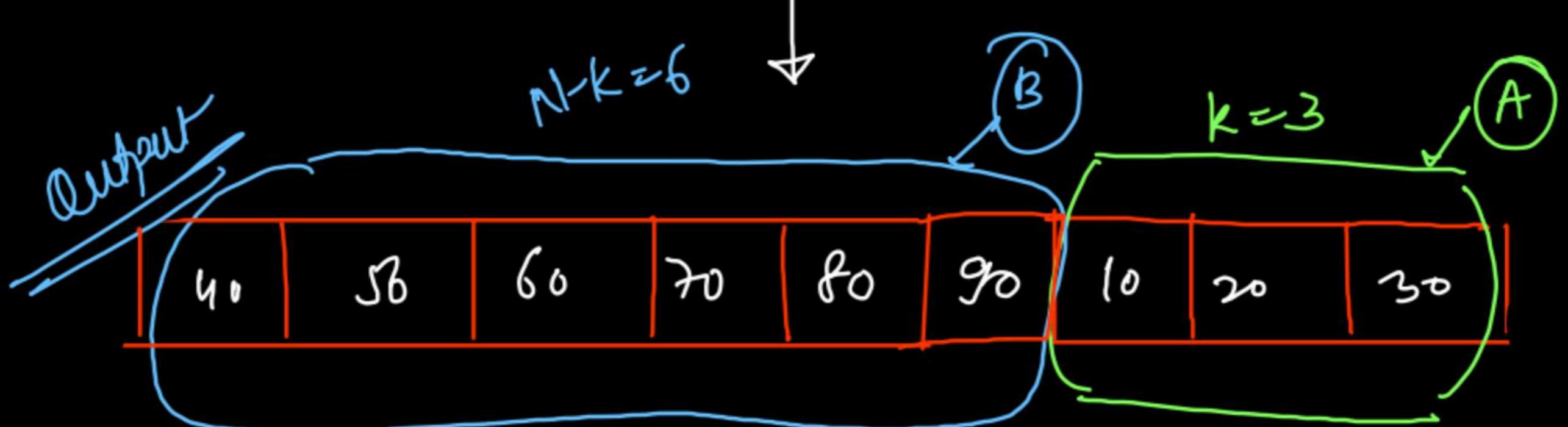
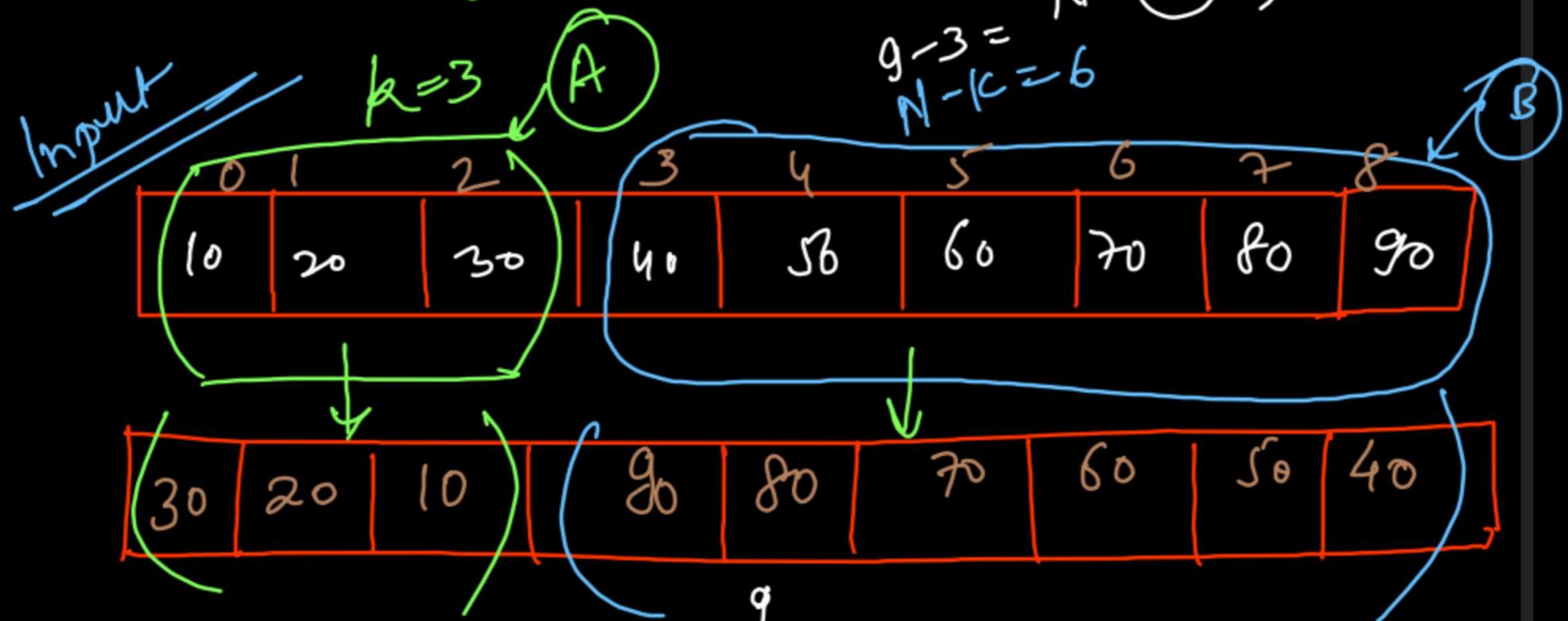
↓ rotate by 1()

{ 40, 50, 10, 20, 30 }

rotations  
k = 3%5  
= 3

### ③ Reversal algorithm

$N = 9$ ,  $k = 3$



```

public static void main(String[] args) {
    int arr[] = {10, 20, 30, 40, 50, 60, 70, 80, 90};
    int k = 3;
    reverseK(arr, k);
    System.out.println(Arrays.toString(arr));
}
    public static void reverseK(int arr[], int k) {
        int n = arr.length;
        int temp = arr[0];
        for (int i = 0; i < k; i++) {
            arr[i] = arr[n - i - 1];
            arr[n - i - 1] = temp;
            temp = arr[i];
        }
    }
}

```

~~#WHAT~~

- ① Reverse starting  $k$  elements → single loop
- ② Reverse last  $\rightarrow$  single loop  
 $n-k$  elements +
- 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] + " ");  
}
```



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

①

$$(x+y)^T = y^T + x^T$$

②

$$(x^T)^T = x$$

```

POINT
    if (numRows == 1) & (numCols == 1)
        return x[0][0];
    else if (numRows == 1)
        return x[0];
    else if (numCols == 1)
        return x;
    else
        return x;
    
```

target = 16

Two sum  $\rightarrow$  Sum of pair = Target

{ 10, 17, 12, 9, 4, 7, 20, 1 }

~~Brute force~~, Nested loops  $\Rightarrow$  Outer loop  $\rightarrow$  1<sup>st</sup> & Inner loop  $\rightarrow$  2<sup>nd</sup> ele of pair

$$(10, 17) = 27$$

$$(10, 4) = 14$$

$$(10, 7) = 17$$

$$(10, 12) = 22$$

$$(10, 20) = 30$$

$$(10, 9) = 19$$

$$(10, 1) = 11$$

$$(17+12) \neq 16$$

$$(17+9) \neq 16$$

$$(17+4) \neq 16$$

$$(17+20) \neq 16$$

$$(17+1) \neq 16$$

$$(12+9)$$

$$(12+4)$$

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

①  $(x+y)^T = y^T + x^T$   
②  $(xy)^T = y^T x^T$

{ 10, 17, 12, 9, 3, 7, 20, 1 }

target = 16

The diagram shows an array of integers: 1, 3, 7, 9, 10, 12, 17, 20. Above the array, indices are labeled: start (at 1), end (at 8), and several green arrows pointing to specific elements (1, 3, 7, 9, 10, 12, 17, 20). Below the array, the indices start, end, and current pointers are also labeled.

$$(1+20) = \boxed{21 > 16}$$

$$(1+17) = \boxed{18 > 16}$$

$$(1+12) = \boxed{13 < 16}$$

$$(3+12) = \boxed{15 < 16}$$

start++ / end--  
↓      ↓  
sum ↑      sum ↓

$$(7+12) = \boxed{19 > 16}$$

$$(7+10) = \boxed{17 > 16}$$

$$(7+9) = \boxed{16 = 16}$$

target = 16		Treatment → sum of pair = target
$\{1, 3, 7, 9, 10, 12, 17, 20\}$	$\{1, 3, 7, 9, 10, 12, 17, 20\}$	$\{1, 3, 7, 9, 10, 12, 17, 20\}$
$(1+20) = 21$	$(1+20) = 21$	$(1+20) > 16$
$(1+17) = 18$	$(1+17) = 18$	$(1+17) > 16$
$(1+12) = 13$	$(1+12) = 13$	$(1+12) < 16$
$(3+12) = 15$	$(3+12) = 15$	$(3+12) < 16$
		$\boxed{(7+10) = 17}$
		$\boxed{(7+9) = 16}$
		$\boxed{(7+7) = 14}$
		$\boxed{(1+1) = 2}$

2 Sum

+-----  
| 13 | 15 | 16 | 17 | 18 | 19 | 21 |

{ 1, 3, [ 7, 9 ], 10, 12, 17, 20 }

target = 16

```

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;

```

$\text{H20} > 16$   
 $1+17 > 16$   
 $1+12 < 16$   
 $3+12 < 16$   
 $7+12 > 16$   
 $7+10 > 16$   
 $7+9 = 16$

$\{ 1, 3, 7, 9, 10, 12, 17, 20 \}$	$\{ 1, 3, 7, 9, 10, 12, 17, 20 \}$
$\{ 1, 3, 7, 9, 10, 12, 17, 20 \}$	$\{ 1, 3, 7, 9, 10, 12, 17, 20 \}$
$(1+3) = 4 > 16$	$\frac{1+3+7}{3} > 16$
$(1+7) = 8 > 16$	$\frac{1+3+9}{3} > 16$
$(1+9) = 10 < 16$	$\frac{1+3+10}{3} > 16$
$(3+10) = 13 < 16$	$\frac{1+3+12}{3} > 16$
$(3+12) = 15 < 16$	$\frac{1+3+17}{3} > 16$
$(7+9) = 16 = 16$	$\frac{1+3+17}{3} = 16$

~~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

elements summing to

target

<del>l</del>	<del>r</del>	<del>mid</del>	<del>arr[mid]</del>	<del>arr[l] + arr[r] &gt; target</del>
1	3	2	7	<del>arr[l] + arr[r] &gt; target</del>
1	3	2	7	<del>arr[l] + arr[r] &gt; target</del>
1	3	2	7	<del>arr[l] + arr[r] &gt; target</del>
1	3	2	7	<del>arr[l] + arr[r] &gt; target</del>

$$1+15 \rightarrow 16$$

$$1+15 = 16$$

$$2+15 \rightarrow 16$$

$$2+14 \rightarrow 16$$

$$4+14 \rightarrow 16$$

$$4+12 \rightarrow 16$$

$$5+12 \rightarrow 16$$

target  $\boxed{16}$

~~(1, 15)~~ ~~(4, 12)~~

~~(2, 14)~~ ~~(7, 9)~~

2 sum  $\rightarrow$  All the pairs equal to the target



```
while (left < right) {  
    if (arr[left] + arr[right] == target)  
        cout << arr[left] + " " + arr[right]; left++;  
    else if (arr[left] + arr[right] < target)  
        left++;  
    else if (arr[left] + arr[right] > target)  
        right--;
```

Step	left	right	arr[left] + arr[right]	target
1	1	15	16	16
2	2	14	16	16
3	3	13	16	16
4	4	12	16	16
5	5	11	16	16
6	6	10	16	16
7	7	9	16	16
8	8	8	16	16
9	9	7	16	16
10	10	6	16	16
11	11	5	16	16
12	12	4	16	16
13	13	3	16	16
14	14	2	16	16
15	15	1	16	16



Triplet with target = 30

**3 Sum** → point true or false

first no + pair with target = 29

target = 30

first no + pair with target = 29

③

(10, 19)

$$8 + 10 + 12 = 30$$

$$\text{arr}[i] + \text{arr}[j] + \text{arr}[k] = \text{target}$$

2 sum TP  
1 loop

for  
2 loops →

3 sum TP... etc

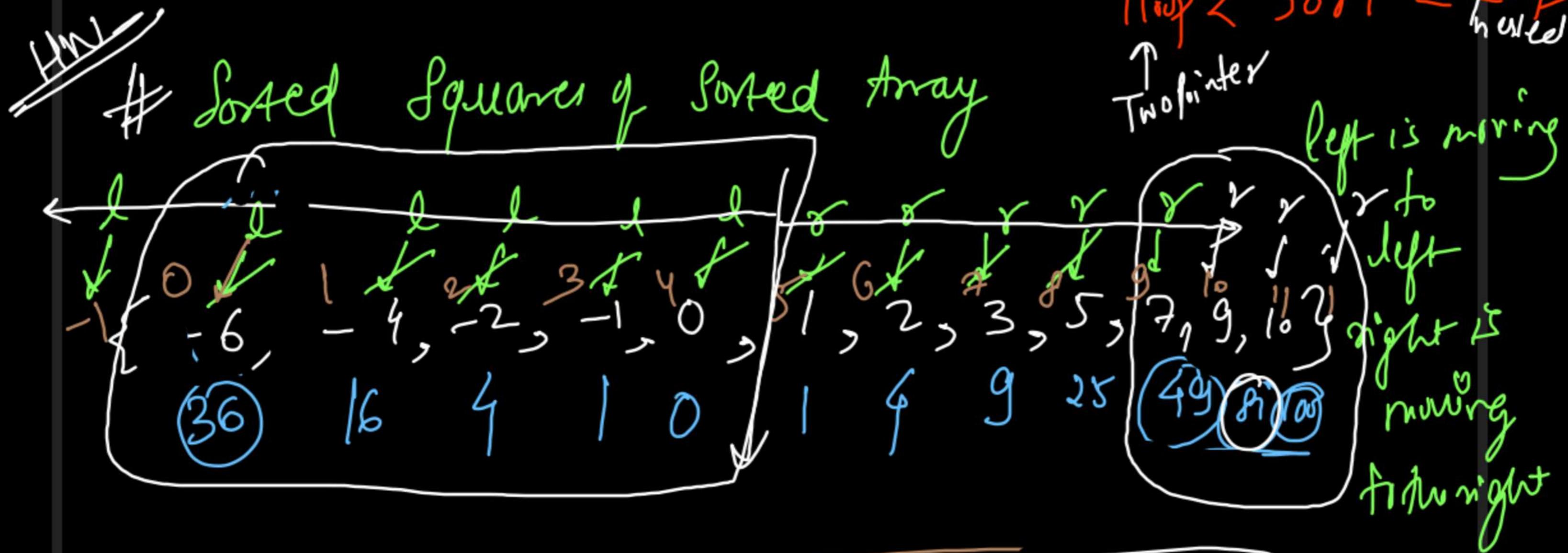
3 sum	
target	All the pairs equal to the target
loop	for i = 0 to n - 2
loop	for j = i + 1 to n - 1
loop	for k = j + 1 to n
loop	if arr[i] + arr[j] + arr[k] == target
loop	print(i, j, k)
loop	else if arr[i] + arr[j] + arr[k] > target
loop	break
loop	else if arr[i] + arr[j] + arr[k] < target
loop	j++

triplet(i, j, k)

i ≠ j ≠ k

for  
3 loops → 4 loops  
4 loops → 3 loops

loop < sort < 2 loops  
handled



0 1 4 9 16 25 36 49 81 100

0 1 4 16 36 → merge<sup>2</sup>  
1 4 9 25 49 → sorted

→ print this ans  
ans = [0, 1, 4, 9, 16, 25, 36, 49, 81, 100]

## # Frequency Array

array of characters → it will contain only lowercase alphabets

Input:  $\{ 'g', 'e', 'e', 'k', 's', 't', 'e', 's,$   
 $'a', 'r', 'c', 'h', 'i', 't' \}$

Count the frequency of each English alphabet

Output:  $a \rightarrow 1, b \rightarrow 0, c \rightarrow 1, d \rightarrow 0, e \rightarrow 3,$   
 $f \rightarrow 0, g \rightarrow 0, h \rightarrow 1, \dots$

for sorted frequency of sorted array	
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14
14	15
15	16
16	17
17	18
18	19
19	20
20	21
21	22
22	23
23	24
24	25
25	26
26	27
27	28
28	29
29	30
30	31
31	32
32	33
33	34
34	35
35	36
36	37
37	38
38	39
39	40
40	41
41	42
42	43
43	44
44	45
45	46
46	47
47	48
48	49
49	50
50	51
51	52
52	53
53	54
54	55
55	56
56	57
57	58
58	59
59	60
60	61
61	62
62	63
63	64
64	65
65	66
66	67
67	68
68	69
69	70
70	71
71	72
72	73
73	74
74	75
75	76
76	77
77	78
78	79
79	80
80	81
81	82
82	83
83	84
84	85
85	86
86	87
87	88
88	89
89	90
90	91
91	92
92	93
93	94
94	95
95	96
96	97
97	98
98	99
99	100

```
// 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);  
}
```



Comparisons or operations



## ② Optimized Approach (frequency array)

frequency  
26 size

int	1	1	3	22	1 1 1	1	$\gamma^2$
a[0]	0	b[1]	c[2]	d[3]	e[4]	f[5]	g - - k s t z
0	1	2	3	4	5	6	- - - 25

index  
of freq array  
ch-'a'

{ 'g', 'e', 'e', 'k', 's', 't', 'e', 'z',  
 'a', 'r', 'c', 'h', 'i', 't' }

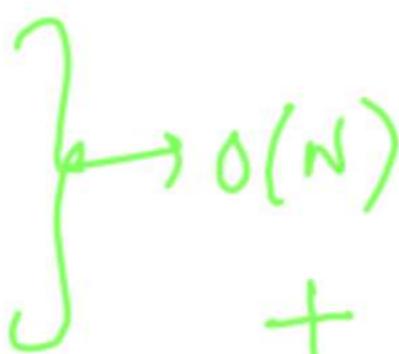
```

// 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]++;
}

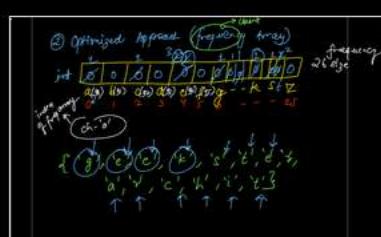
// Loop will run for 26 iterations
for(int i=0; i<26; i++){
    System.out.print(freq[i] + " ");
}

```


 $\rightarrow O(N)$   
 +
   

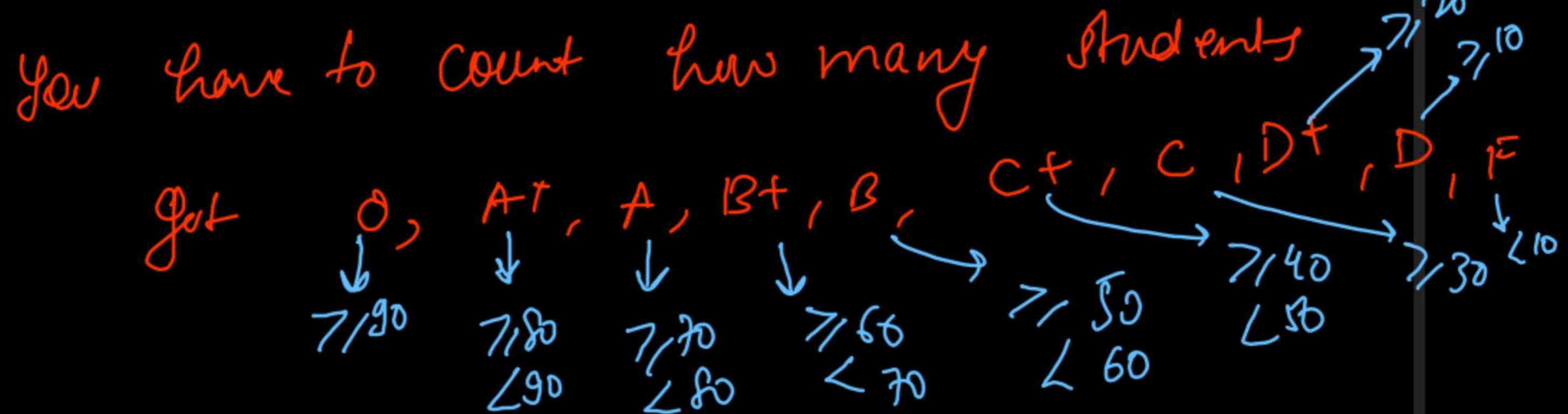
 $\rightarrow O(26)$

$O(N + 26)$   
*# These are not nested loops*



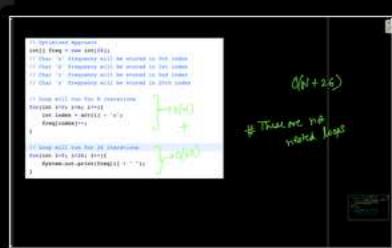
Eg 2

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



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

count of 0 = 2, count of  $A^+$  = 1, count of  $A$  = 0, --  
--



```

// 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

```

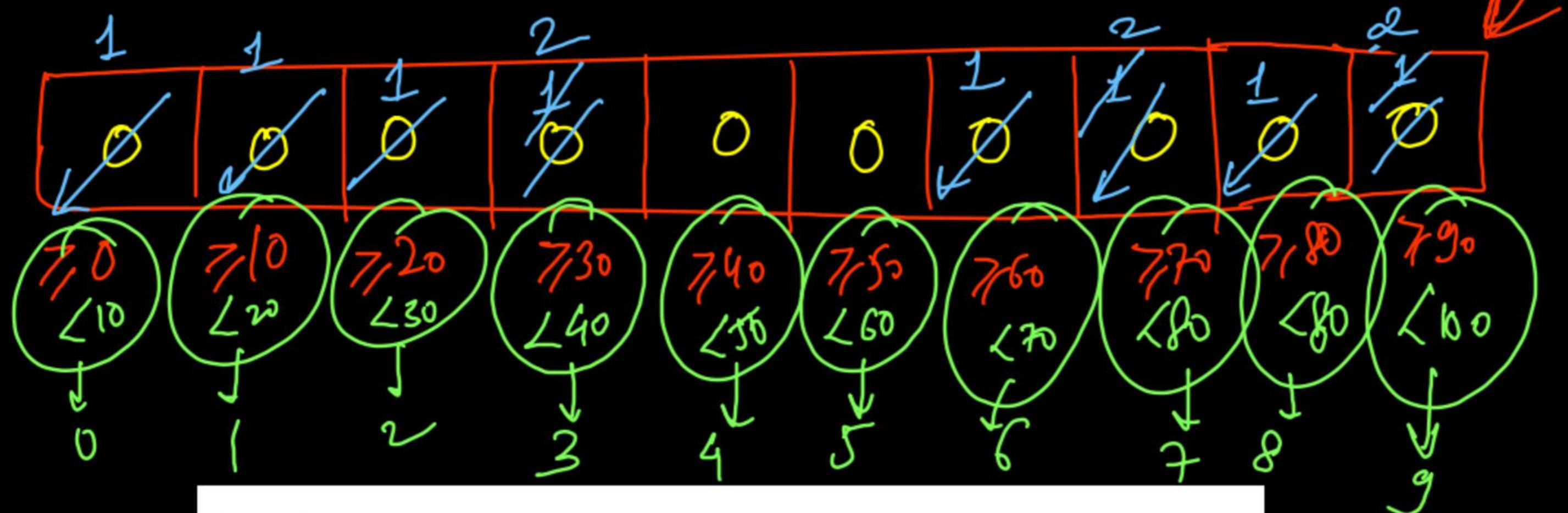
stdin ▾

12

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

## # Approach 2: Frequency Array

[types of grades]

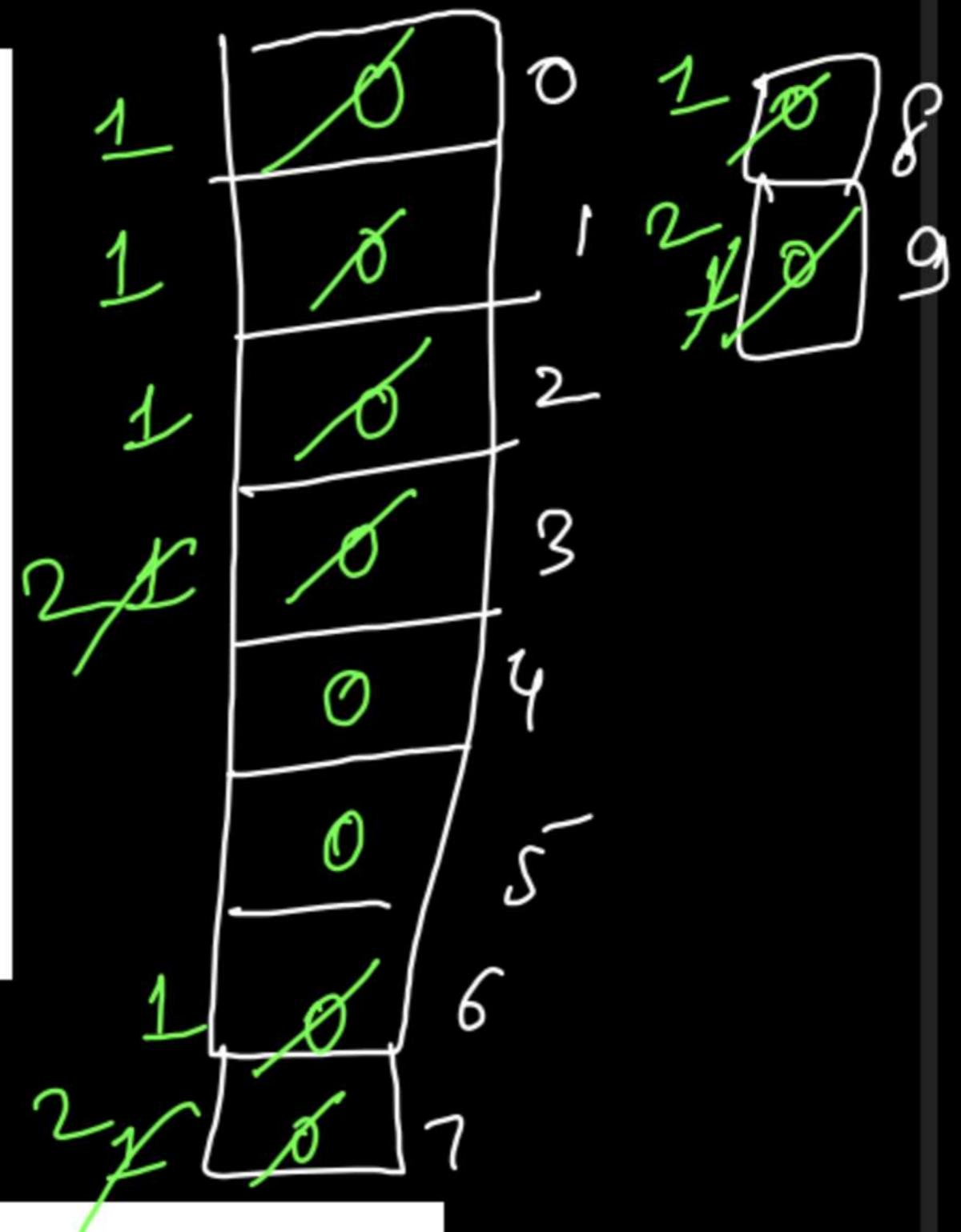


② 11

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  
88/10 = 8  
73/10 = 7  
75/10 = 7  
0/10 = 0  
33/10 = 3  
10/10 = 1

```
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



30

40

51

61

71

81

91

39

49

59

69

79

89

99

```

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

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

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

Array

Sum of all ranges  
Subarray  
 $\{l \leq r\}$

Array:  $\{10, 12, 8, 16, 2\}$

$$[0,0] = 10 \checkmark$$

$$[0,1] = 10+12 = 22 \checkmark$$

$$[0,2] = 10+12+8 = 30 \checkmark$$

$$[0,3] = 10+(2+8+16) = 46 \checkmark$$

$$[0,4] = 10+12+8+(6+2) = 48 \checkmark$$

$$[1,1] = 12 \checkmark$$

$$[1,2] = 12+8 \checkmark$$

$$[1,3] = 12+8+16 = 36 \checkmark$$

$$[1,4] = 12+8+16+2 = 38 \checkmark$$

$$[2,2] = 8$$

$$[2,3] = 8+16 = 24$$

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

Out	0	10	22	30	46	48	8	24	26
l	0	1	2	3	4	5	6	7	8
r	0	1	2	3	4	5	6	7	8
sum	0	10	22	30	46	48	8	24	26
range	0	10	22	30	46	48	8	24	26

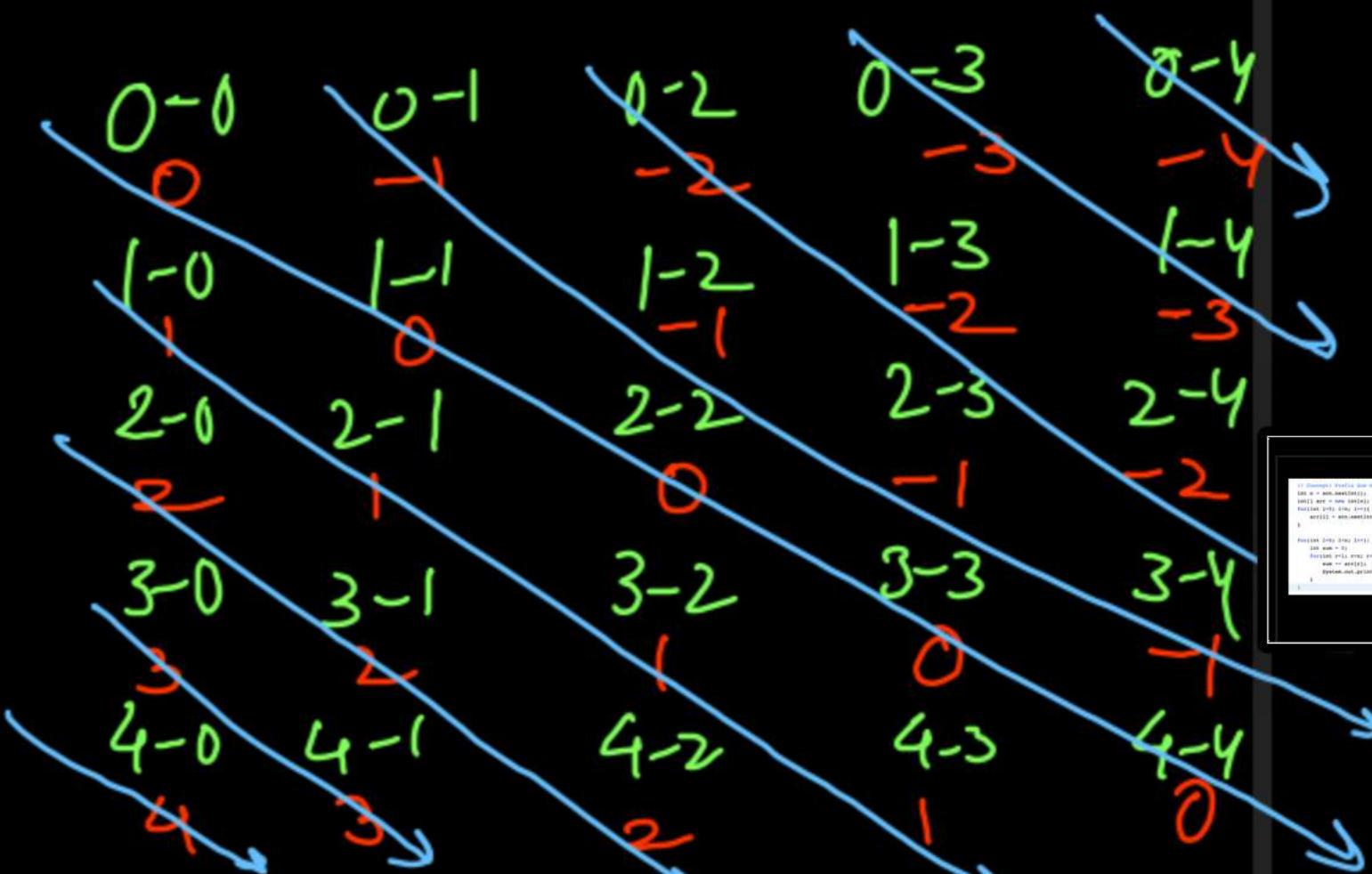
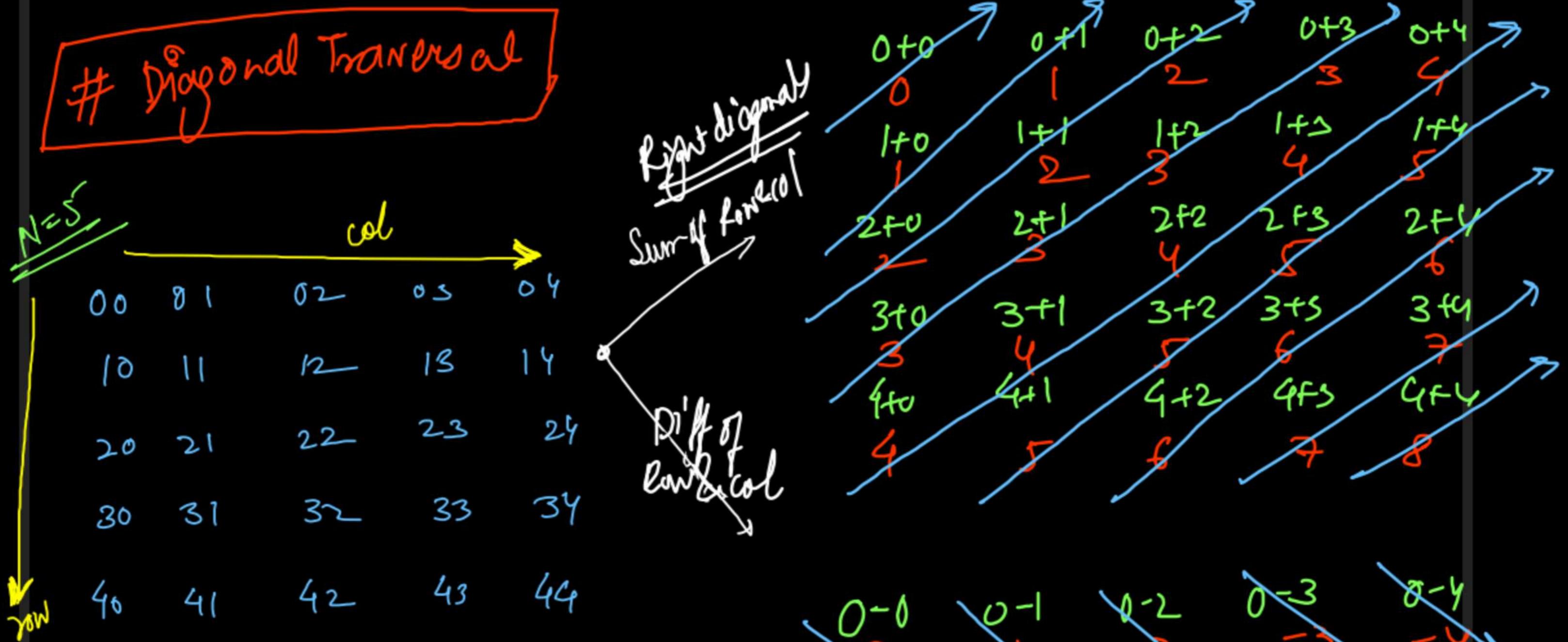
Surf 01

```
// 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



```

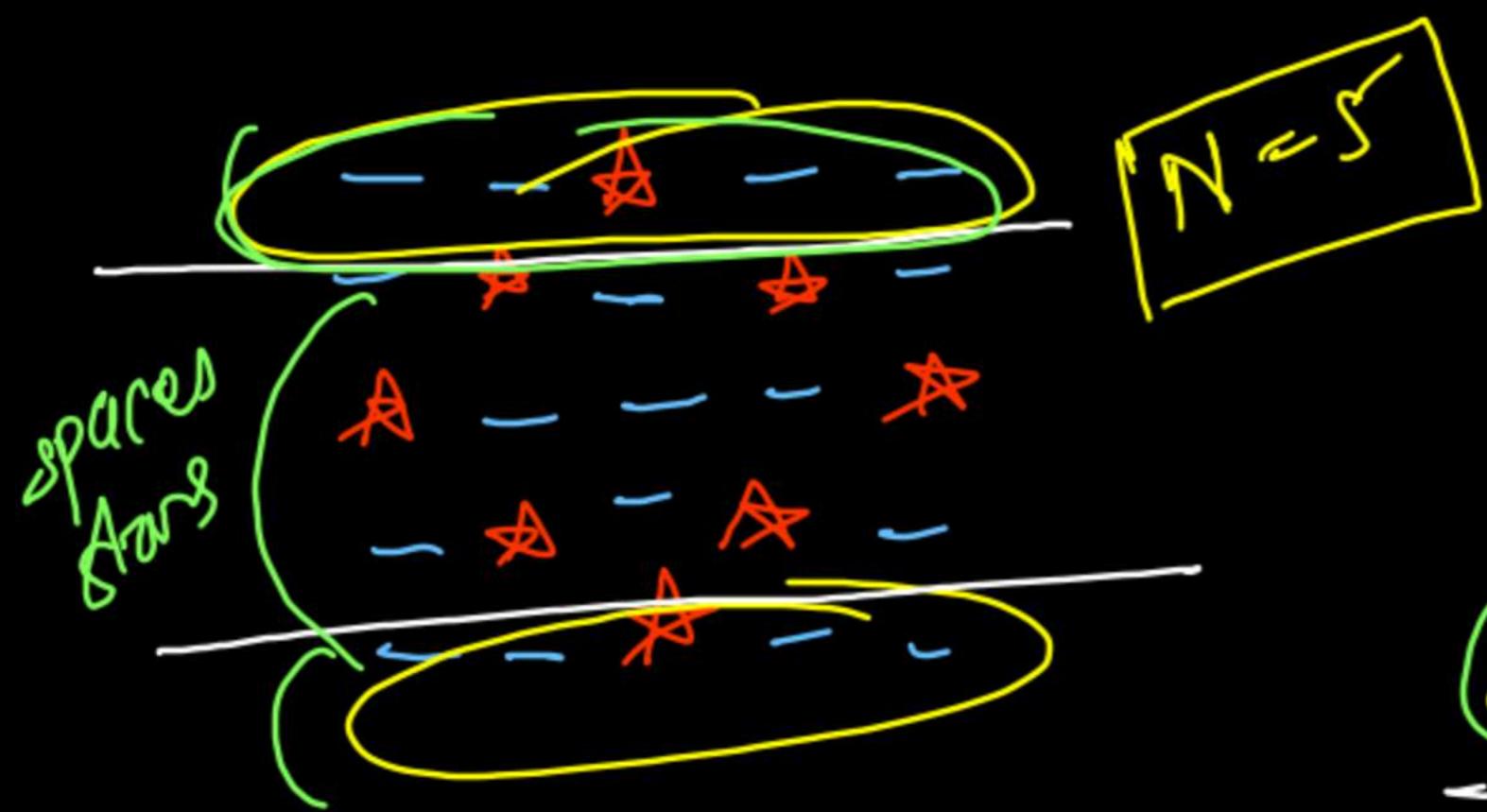
1. int main() {
2.     int arr[5][5];
3.     int i, j;
4.     for(i = 0; i < 5; i++) {
5.         for(j = 0; j < 5; j++) {
6.             cout << arr[i][j] << " ";
7.         }
8.     }
9. }
```

main function  
sum of all the integers

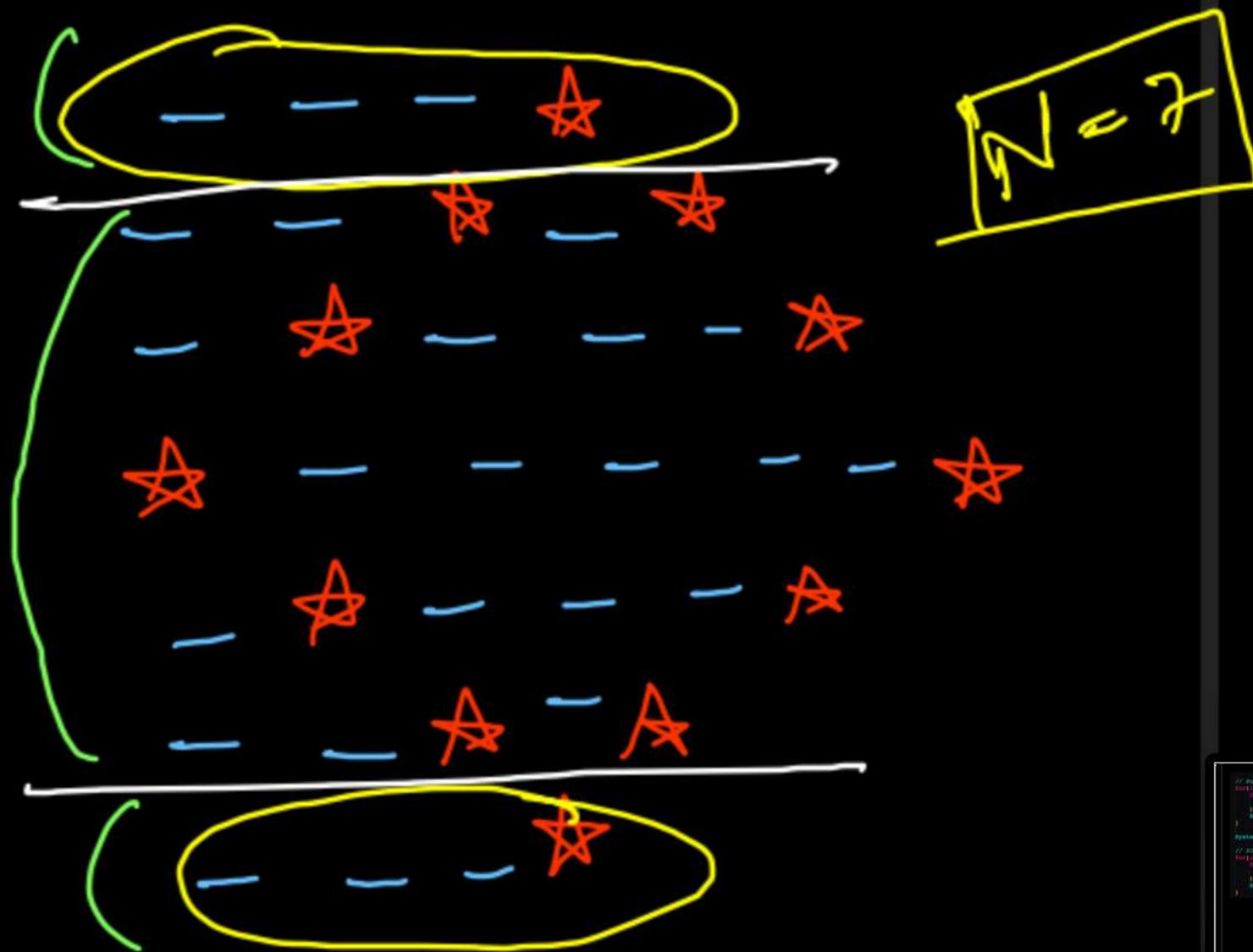
```
// Sum of Row & Col Will Give Equal Elements on Right Diagonals
for(int row=0; row<n; row++){
    for(int col=0; col<n; col++){
        System.out.print(row + col + " ");
    }
    System.out.println();
}

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();
}
```



$$N=5$$



$$N=7$$

```
// sum of last N will give Right Elements on Right Diagonals
int sumRightDiag(int n)
{
    int sum = 0;
    for (int i = 0; i < n; i++)
        sum += arr[i][n - 1 - i];
    return sum;
}

System.out.println();
```

```
// sum of last N + 1 will give Right Elements on Left Diagonals
int sumLeftDiag(int n)
{
    int sum = 0;
    for (int i = 0; i < n; i++)
        sum += arr[n - 1 - i][i];
    return sum;
}
```

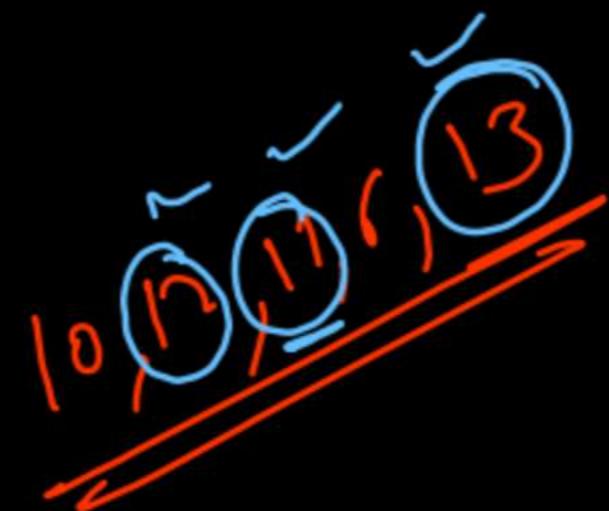
# 25 horses

In one race → 5 horses can participate

↳ fastest horse

3 fastest ones

minimum races



H<sub>a</sub>



H<sub>a</sub>

H<sub>1</sub>

H<sub>5</sub>

H<sub>b</sub>

H<sub>6</sub>

H<sub>10</sub>

H<sub>c</sub>

H<sub>11</sub>

H<sub>15</sub>

H<sub>d</sub>

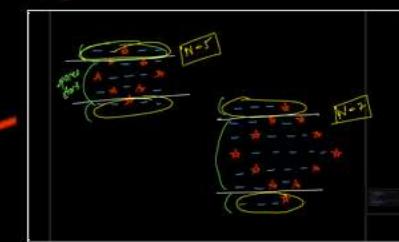
H<sub>16</sub>

H<sub>20</sub>

H<sub>e</sub>

H<sub>21</sub>

H<sub>25</sub>



Target sum pair (2 sum)  $\rightarrow$  Closest Target

closest  $\Rightarrow$   $\cancel{50}$   $\cancel{52}$   $\cancel{55}$   $\cancel{53}$   
target  $\approx 54$

10, 22, 25, 28, 30, 40  
~~X~~ ~~X~~ ↑ ↑ ~~X~~ ~~X~~

$$|5| = 5$$

$$|-3| = 3$$

$$|+\infty| = \infty$$

$$|-\infty| = -\infty$$

$$\uparrow 10 + 40 < 54$$

$$22 + 40 > 54$$

$$22 + 30 = 52 < 54$$

$$\cancel{25 + 30 = 55} > 54$$

$$25 + 28 = 53 < 54$$

