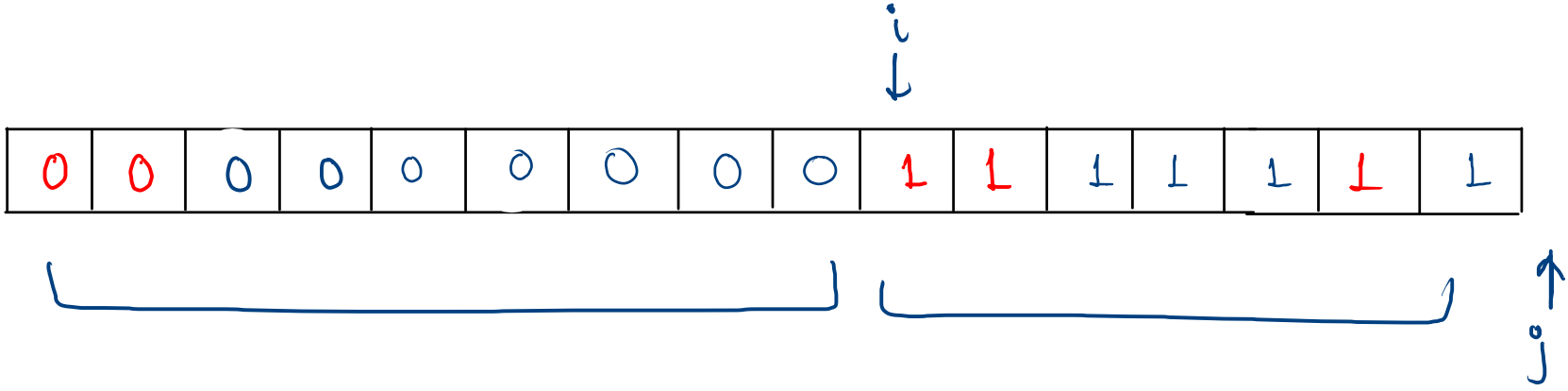
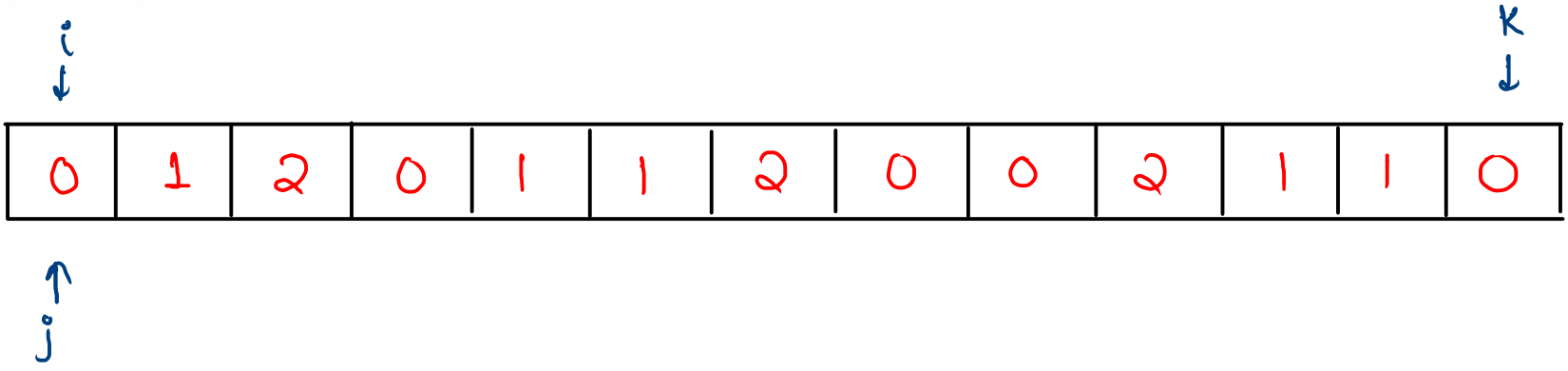
Zeroes and Ones

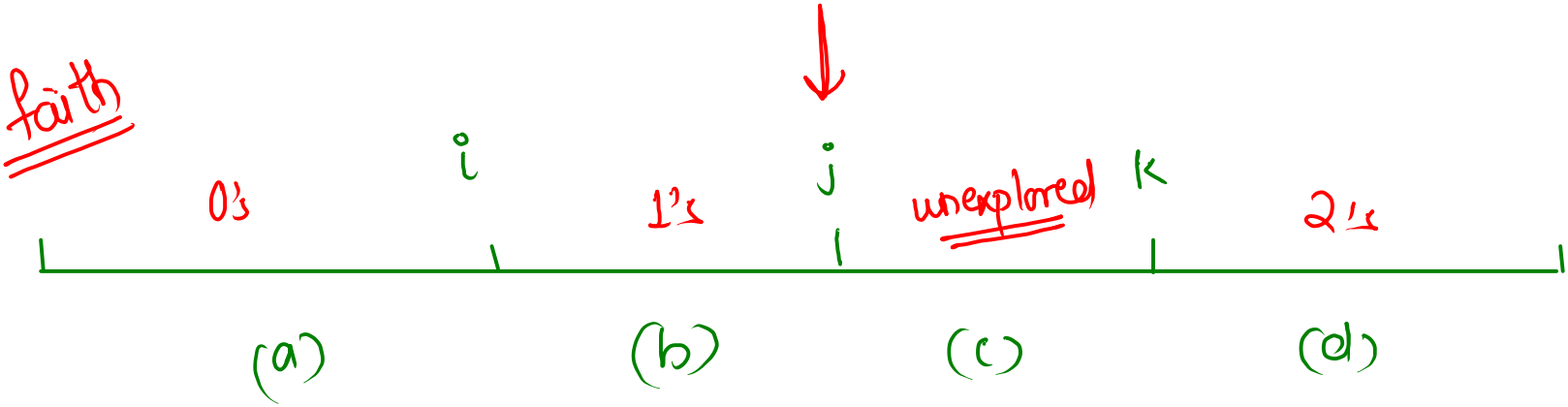


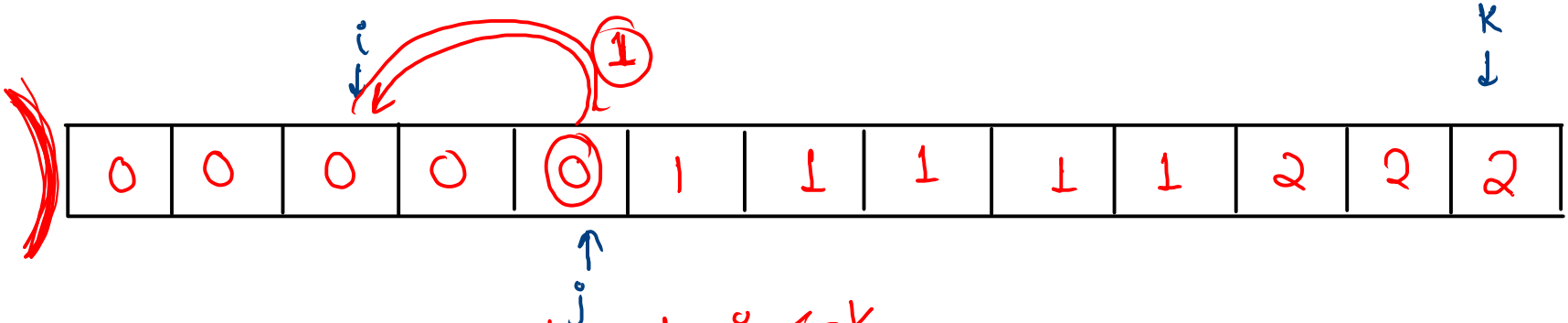
psudo code

- iterate j^{th} pointer till end
  - if j^{th} element is 1, j++
  - else j^{th} element is 0, swap(i,j) i++, j++

Sort 0 1 2







condition ⇒ j <= K

psudo code ⟶ traverse j^{th} pointer from start to end

a ↳ if j^{th} ele. is 1, j++

b ↳ if j^{th} ele. is 0, swap(i, j) i++, j++

c ↳ if j^{th} ele. is 2, swap(j, k). k--

```java
public static void zeroOneTwo(int[] arr, int n) {
    int i = 0;
    int j = 0;
    int k = n - 1;
    while (j <= k) {
        if ( arr[j] == 1 ) {
            j++;
        } else if ( arr[j] == 0 ) {
            swap(arr, i, j);
            i++;
            j++;
        } else {
            swap(arr, j, k);
            k--;
        }
    }
}

public static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

Rotate Right

$n = 7$

arr

1	2	3	4	5	6	7

$k = 3$

rotation $k = 1$, 7 1 2 3 4 5 6

rotation $k = 2$, 6 7 1 2 3 4 5

rotation $k = 3$, 5 6 7 1 2 3 4

arr | 1 | 2 | 3 | 4 | 5 | 6 | 7 | , n = 7

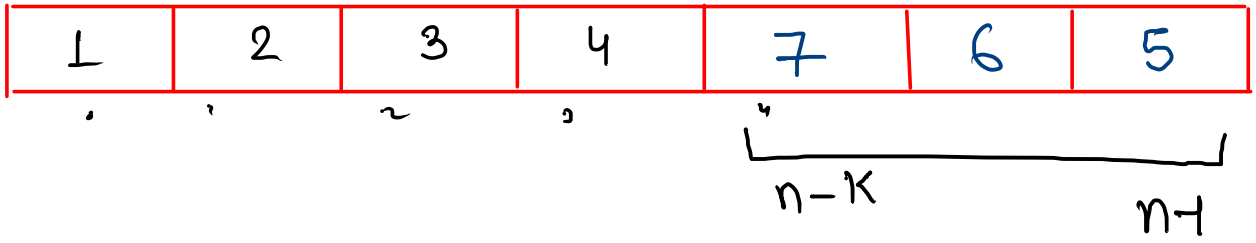
k = 3

n - k = 4

approch

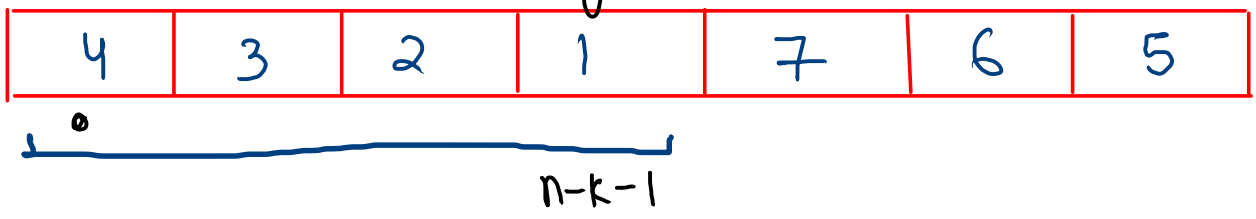
reverse k elements from last

Step1



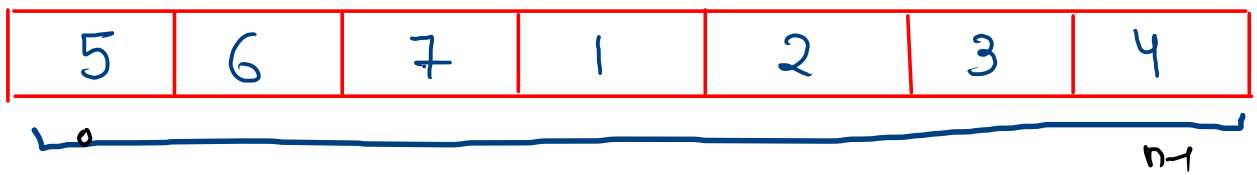
reverse remaining element

Step2



reverse all elements

Step3



input

```java
public static void main(String[] args) {
    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 k = scn.nextInt();
    rotateArray(arr, n, k);

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

main logic

```java
public static void rotateArray(int[] arr, int n, int k) {
    reverse( arr, n - k, n - 1 );
    reverse( arr, 0, n - k - 1 );
    reverse( arr, 0, n - 1 );
}
```

prev. ques

```java
public static void reverse(int[] arr, int si, int ei) {
    while ( si < ei ) {
        swap(arr, si, ei);
        si++;
        ei--;
    }
}
```

swap

```java
public static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

→ Variation of 2 pointer

TLE

T.C = $O(N^2)$

= $O(N)$

Reach Target

	0	1	2	3	4	5
arr	1	2	3	5	7	8

target = 8 ;

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

print i , j

ans) 2 , 3

0 , 4

Sorted

arr

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

i j

target = 8

sum = 1 + 8 = 9

= 1 + 7 = 8 (0, 4) ✓

= 2 + 5 = 7

= 3 + 5 = 8 (2, 3) ✓

psudo code

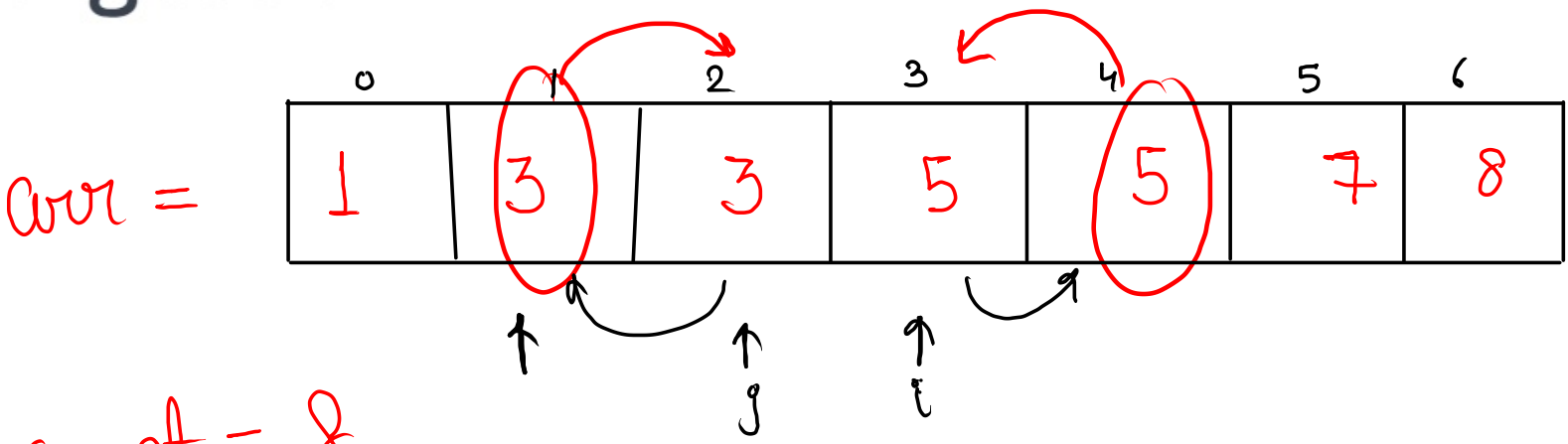
- pointer $i = 0$, $j = n-1$
- iterate until $i < j$
  - if sum == target
    - print i, j
    - i++, j--
  - elseif sum > target ; j--
  - else sum < target ; i++

Code

```java
public static void reachTarget(int[] arr, int n, int target) {
    int i = 0;
    int j = n - 1;
    while ( i < j ) {
        int sum = arr[i] + arr[j];
        if ( sum == target ) {
            System.out.println(i + " " + j);
            i++;
            j--;
        } else if ( sum > target ) {
            j--;
        } else {
            i++;
        }
    }
}
```

Target Sum

$O(N^2)$ is not acceptable



target = 8 ,

left = 1 3

right = 5

Sum = arr[i] + arr[j]

Sum = 9

Sum = 8 (1 , 7)

Sum = 8 (3 , 5)

Sum = 8

psudo code

→ Arrays.sort(arr)

↳ pointer i = 0, j = n-1

↳ iterate until i < j

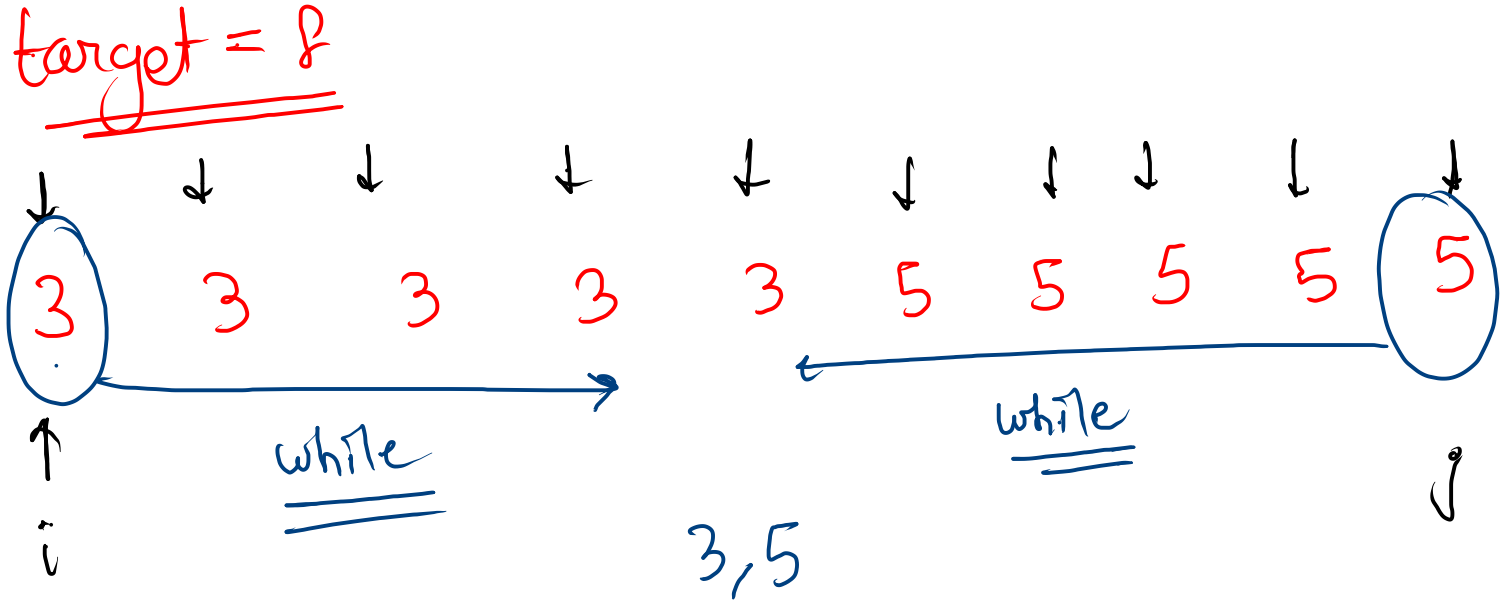
↳ if sum == target

print i, j → loop to skip all repetation

i++, j--

↳ elseif sum > target ; j--

↳ else sum < target ; i++



↳ how many times we are visiting each element

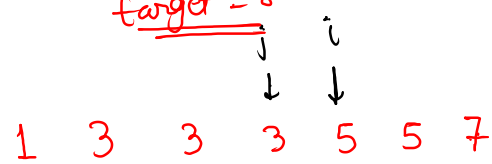
```java
public static void targetSum(int[] arr, int n, int target) {
    Arrays.sort(arr);
    int i = 0;
    int j = n - 1;
    while ( i < j ) {
        int sum = arr[i] + arr[j];
        if ( sum == target ) {
            System.out.println( arr[i] + " " + arr[j] );

            while ( i < j && arr[i] == arr[i + 1] ) {
                i++;
            }

            while ( i < j && arr[j] == arr[j - 1] ) {
                j--;
            }

            i++;
            j--;
        } else if (sum > target) {
            j--;
        } else {
            i++;
        }
    }
}
```

O(N log N)

O(N)

Important

target = 8



sum = ~~8~~ 8

1 , 7

3 , 5

T.C = O(N log N + N) ≅ O(N log N)