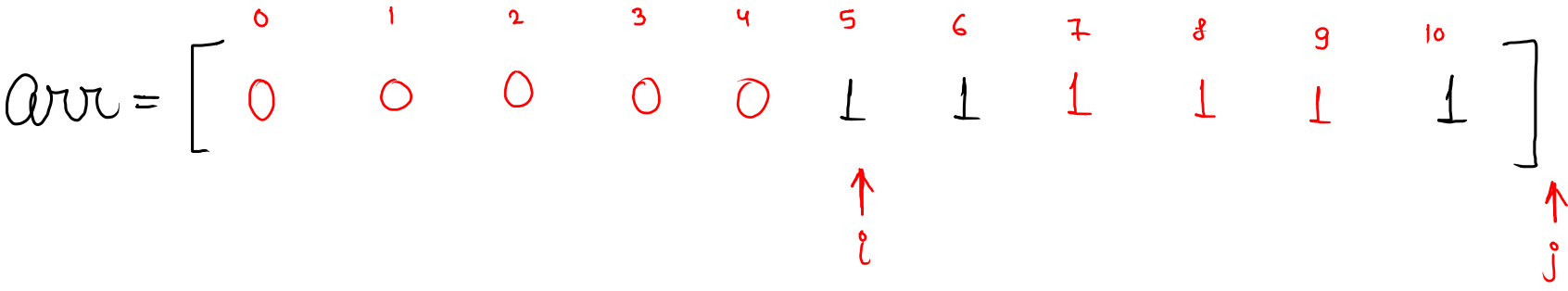
Zeroes and Ones (Inplace) S.C = O(1), T.C = O(n)



2 pointers logic

i → all elements before i are zero

j → all elements after i and before j are one

(rest of the elements are unexplored)

psudo code

1) make i = 0 and j = 0

2) loop until j < n

2.1) if arr[j] is 1

2.1.1) j++

2.2) else if arr[j] is 0

2.2.1) swap(i, j)

i++

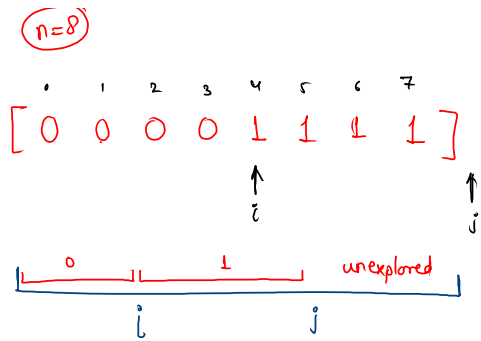
j++

main logic

```java
public static int[] count01(int[] arr, int n) {
    int i = 0;
    int j = 0;
    while ( j < n ) {
        if ( arr[j] == 1 ) {
            j++;
        } else {
            swap(arr, i, j);
            i++;
            j++;
        }
    }
    return arr;
}
public static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

a

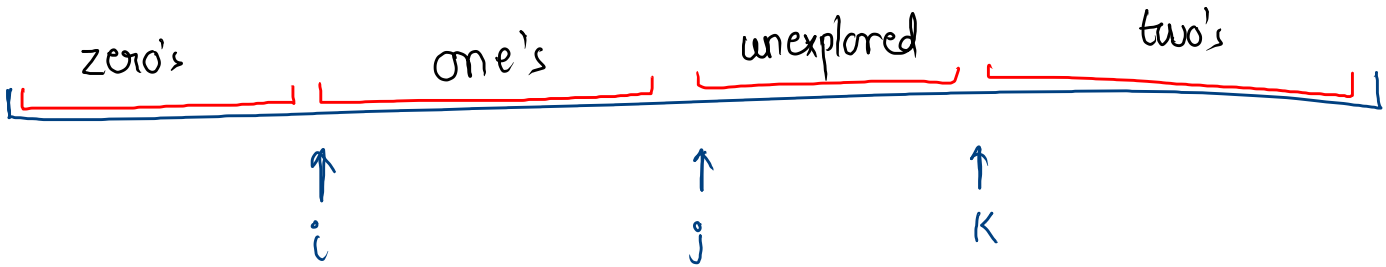
b



Sort 0 1 2

arr =

0	1	1	0	0	2	1	2	0	1	2



Logic :-

start

i → before i all zero's → 0

j → in b/w i and j all one's → 0

k → after k all two's → n-1

(rest of it is unexplored)

psudo code

1) make pointer i=0, j=0, K=n-1;

2) traverse until **j <= K**

2.1) if arr[j] == 1

j++

2.2) else if arr[j] == 0

swap(i, j)

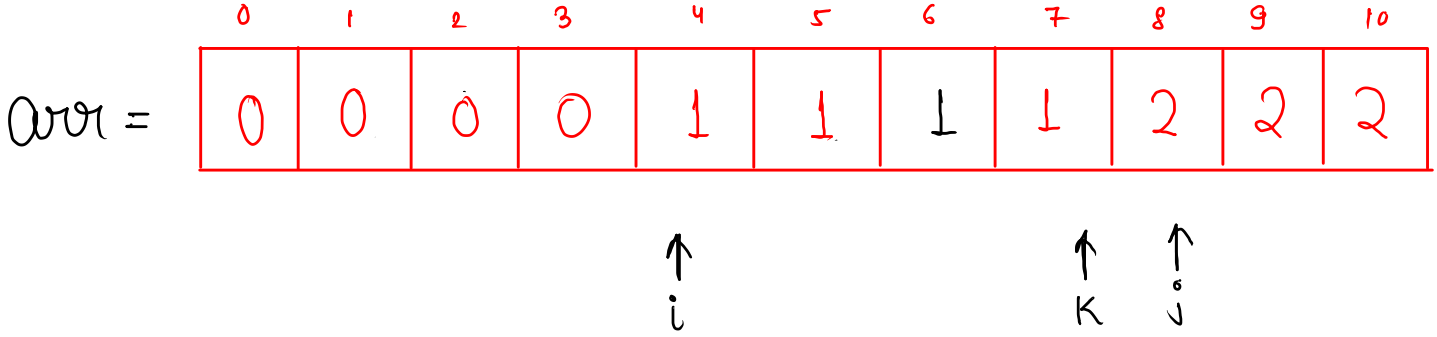
i++

j++

2.3) else arr[j] == 2

swap(j, K)

K--;



code

```java
public static int[] count012(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--;
        }
    }
    return arr;
}

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

T.C = O(N)

S.C = O(1)

⇒ Variation of 2 pointer

Reach Target (sorted)

while (i < j)

n = 6 target = 4

	0	1	2	3	4	5
arr = [	-1	1	2	3	4	5]

i j

sum = arr[i] + arr[j]

sum = ~~4 5~~ 4

o/p

0 5
1 3

psudo code

1) make i = 0, j = n-1 ;

2) loop until i < j

sum = arr[i] + arr[j]

2.1) sum == target

Syso(i + " " + j);

i++, j--;

2.2 sum < target

i++ ;

2.3) sum > target

j-- ;

Code

```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 target = scn.nextInt();
    targetSum(arr, n, target);
}
public static void targetSum(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 ) {
            i++;
        } else {
            j--;
        }
    }
}
```