LC- 189 Reverse Array

n=7 → 7 unique rot.

n=5 → 5 unique rot

k % n

n = 7

k

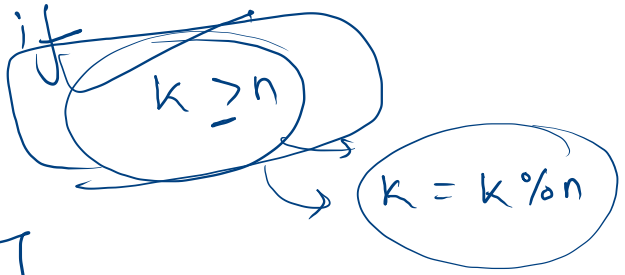
0 7
1 8
2 9
3 10
4 11
5
6

9 % 7 = 2

[0 6]

n = 5 k = 18

k				
0	5	10	15	18 % 5 = 3
1	6	11	16	
2	7	12	17	
3	8	13	18	
4	9	14		



$$7 \overline{)9} \quad 0$$

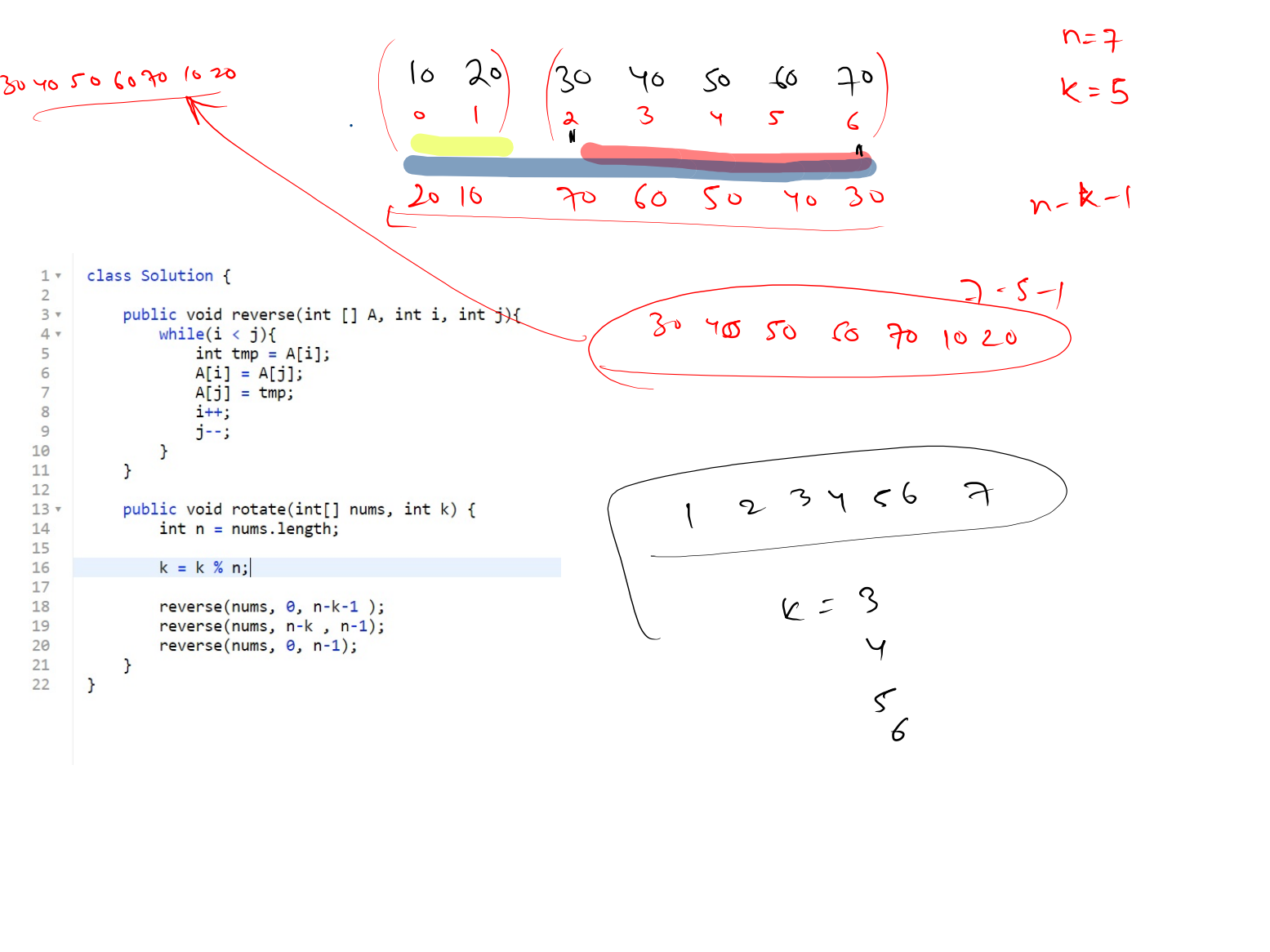
$$2$$

Eg. n = 7

k = 4

k = k % n

k = 4 % 7 = ? = 4




```java
class Solution {

    public void reverse(int [] A, int i, int j){
        while(i < j){
            int tmp = A[i];
            A[i] = A[j];
            A[j] = tmp;
            i++;
            j--;
        }
    }

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

        k = k % n;

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

Rotate by left. (HW).

K=3

K=-ve:

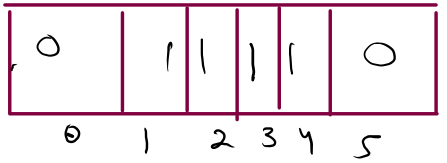
10 20 30 40 50 60 70

RBL 40 50 60 70 10 20 30

RBR 50 60 70 10 20 30 40

Sort 0 1

n = 6



Count Z = 2

Count one = n - 2 = 4

Sort → B, I, S } n^2

→ Arrays.sort O(n log n)

→ O(n).

2 pointer count

Sort by parity.

Sample Input 0

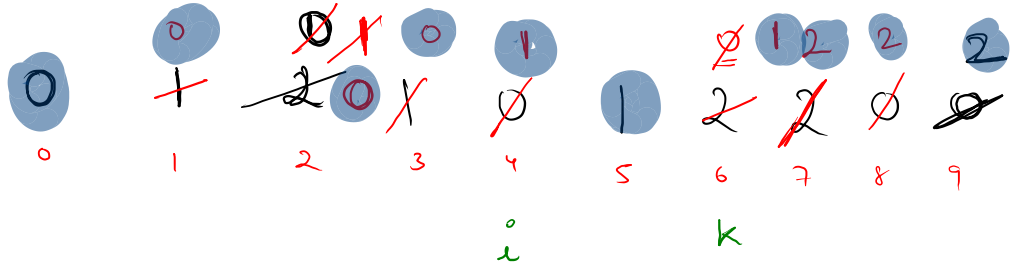
```
6
0 1 1 1 1 0
```

Sample Output 0

```
0 0 1 1 1 1
```

```java
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }

        int i = 0;
        int j = n-1;
        while(i < j){
            if(A[i] == 0){
                i++;
            }else if(A[j] == 1){
                j--;
            }else{
                int tmp = A[i];
                A[i] = A[j];
                A[j] = tmp;
                i++;
                j--;
            }
        }

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

Sort 0 1 2 / Dutch National Flag Algo / Sort Colors / LC-75



A[j] == 0
swap (i, j)
↳ j++
↳ i++

A[j] == 2
swap (j, k)
k--

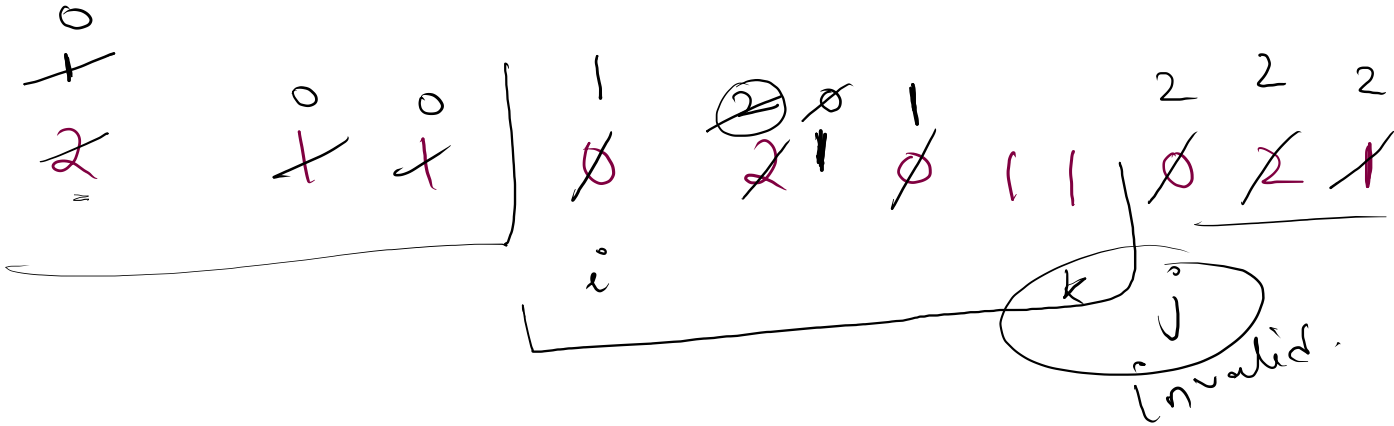
A[j] == 1
↳ j++

j → move

0 ⟶ i-1 = 0
i ⟶ j-1 = 1
j ⟶ k = unexplored

Vanished invalid.

k+1 ⟶ n-1 = 2



A[j] = 0 → swap(i, j) i++ j++

A[j] → 1 → j++

A[j] → 2 → swap(j, k) k--

```java
class Solution {

    public void swap(int x, int y, int [] A){
        int tmp = A[x];
        A[x] = A[y];
        A[y] = tmp;
    }
    public void sortColors(int[] A) {
        int i = 0;
        int j = 0;
        int k = A.length-1;


        while(j <= k){
            if (A[j] == 0){
                swap(i, j, A);
                i++;
                j++;

            }else if(A[j] == 1){
                j++;

            }else{
                swap(j, k,A);
                k--;
            }
        }
    }
}
```

0 → i-1 = 0

i → j-1 = 1

k+1 → n-1 = 2

j to k → unknown

Reach Target

Take the target as an integer input. Then print the **indices** of the two numbers such that they add to the **target**.
Note that the array is sorted here.

Use **Two pionter**, answer must be **unique**.

n=6
tar = 4

Sample Input 0

```
6
-1 1 2 3 4 5
4
```

-1 1 2 3 4 5
0 1 2 3 4 5
i
j

Sample Output 0

```
0 5
1 3
```

s == t
↳ print
i++ j--

0 5
1 3

s > t
j--

s < t
i++

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        int tar = scn.nextInt();

        int i = 0;
        int j = n-1;

        while(i < j){
            int sum = A[i] + A[j];
            if(sum == tar){
                System.out.println(i + " " +j);
                i++;
                j--;
            }
            else if(sum > tar){
                j--;
            }else if(tar > sum){
                i++;
            }
        }
    }
}
```