DATE : 29/12/2023

TOPIC : ARRAY QUESTIONS

TIME : 9:08 AM

Pattern to the solve:

**Problem Statement:**

**Brute Force Approach:**

**Code:**

**Dry Run:**

**Time And Space:**

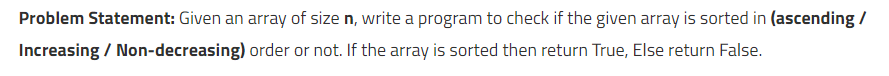
**Optimal Approach:**

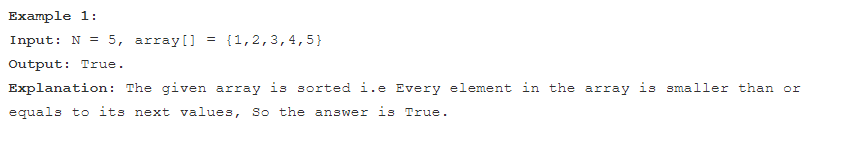
**Code:**

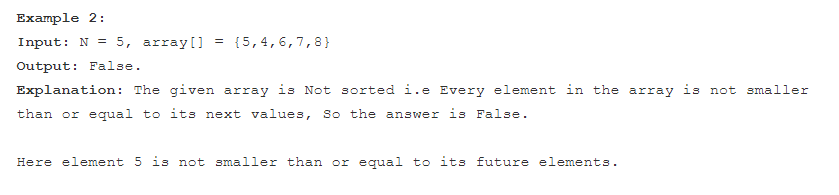
**Dry Run:**

**Time And Space:**

Q1) Check if the array is sorted or not.

**Problem Statement :**





**Brute Force Approach (2 for loop wala):**

**1. Keep an outer for loop , for index I , and keep a inner for loop for index j.**

**2. Since we need to check if it is sorted or not , the next number will always be bigger than the previous number. Thus , the condition if arr[j]>arr[i] , return true as this will be true.**

**3. Else at any case return 0.**

**Code**

**class TUF {**

**static boolean isSorted(int arr[], int n) {**

**for (int i = 0; i < n; i++) {**

**for (int j = i + 1; j < n; j++) {**

**if (arr[j] < arr[i])**

**return false;**

**}**

**}**

**return true;**

**}**

**public static void main(String args[]) {**

**int arr[] = {1, 2, 3, 4, 5}, n = 5;**

**System.out.println(isSorted(arr, n));**

**}**

**}**

**Dry Run:**

**No need for this one, it is easy.**

**Time Complexity Analysis: Since we are using two for loops, thus the time complexity turns out to be,**

**O(n^2).**

**Space Complexity Analysis: O(1)**

**Optimal Approach:**

**We can do the entire traversal in a single loop, reducing the time complexity.**

1. Keep a pointer i.

2. Apply a single for loop from i=0 till i=Size of array.

3. If arr[i]<arr[i-1] , piche wale se agar chota hogaya , to bhai , false return kardo.

4. Else true return kardo.

**Code:**

class TUF {

static boolean isSorted(int arr[], int n) {

for (int i = 1; i < n; i++) {

if (arr[i] < arr[i - 1])

return false;

}

return true;

}

public static void main(String args[]) {

int arr[] = {1, 2, 3, 4, 5}, n = 5;

System.out.println(isSorted(arr, n));

}

}

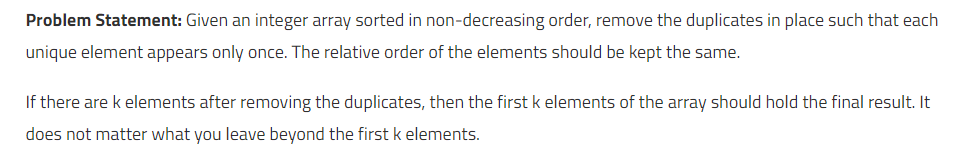
**Dry Run:**

**Time** : O(N) since using only one for loop

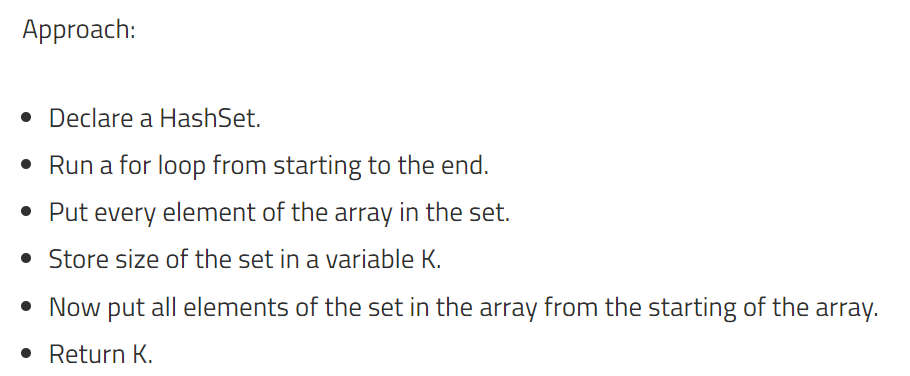
**Space** : O(1)

Q2. Remove Duplicates in-place from Sorted Array.

**Problem Statement:**

****

**Brute Force Approach:**

****

**Code:**

import java.util.HashSet;

// Time : 0(n\*logn) + O(n)

public class FindDuplicates {

    public static int[] duplicates(int[] arr)

    {

        // Brute approach: Hash set

        HashSet<Integer> set = new HashSet<>();

        for(int i=0;i<arr.length;i++)

        {

            set.add(arr[i]);

        }

        // Take all the values of the set into the array

        int k=set.size();

        int result[]=new int[k];

        int index=0;

        for(int value : set){

            result[index++]=value;

        }

        return result;

    }

    public static void main(String[] args) {

        int arr[] = {1,1,2,2,2,3,3};

        int[] uniques=duplicates(arr);

        for(int i=0;i<uniques.length;i++)

        {

            System.out.println(uniques[i]);

        }

    }

}

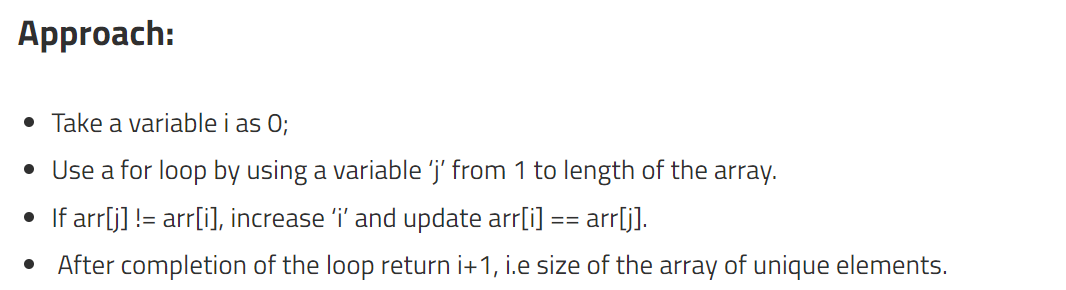
**Dry Run:**

**Time And Space:**

**Time : O(n\*Logn)+O(n)**

**Space : O(n)**

**Optimal Approach(Two Pointer)**

****

**Code:**

public class FindDuplicateOptimal {

    public static int[] removeDuplicates(int[] arr)

    {

        int i=0;

        for(int j=i+1;j<arr.length;j++)

        {

            if(arr[i]!=arr[j])

            {

                i++;

                arr[i]=arr[j];

            }

        }

        int result[]=new int[i+1];

        for(int k=0;k<result.length;k++)

        {

            result[k]=arr[k];

        }

        return result;

    }

    public static void main(String[] args) {

        //

        int arr[] = {1,1,2,2,2,3,3};

        int result[]= removeDuplicates(arr);

        for(int i=0;i<result.length;i++)

        {

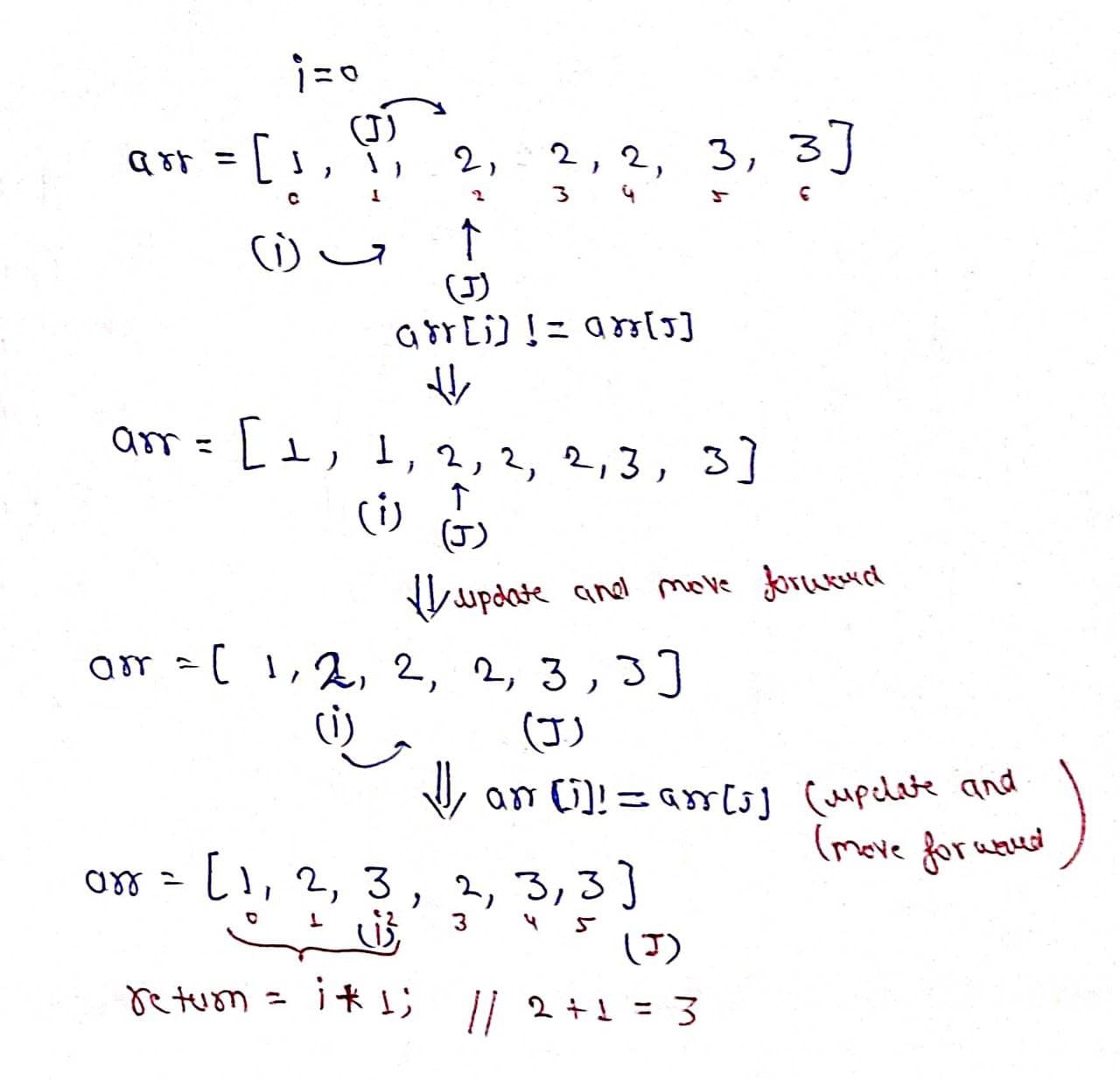
            System.out.print(result[i]);

        }

    }

}

**Dry Run:**



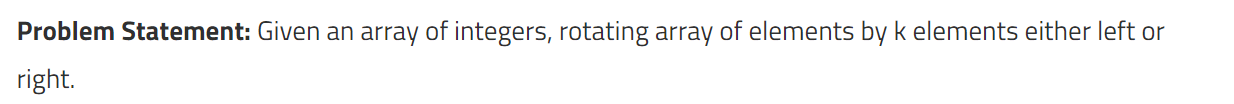
**Time And Space:**

**Time : O(N)**

**Space: O(1)**

**Q3) Rotate Elements by K elements.**

**Problem Statement:**

****

**Brute Force Approach:**

**1. Take an outer loop , and then take an inner loop inside the loop , the outer loop takes an k iterations, the inner loop shifts the element by one.**

**Code:**

**Dry Run:**

**Time And Space:**

**Optimal Approach:**

1. We will apply the concept of reversing the array.

How to reverse an array ?

public void reverse(int arr[],int start,int end)

    {

        while(start<end)

        {

            int temp=arr[start];

            arr[start]=arr[end];

            arr[end]=temp;

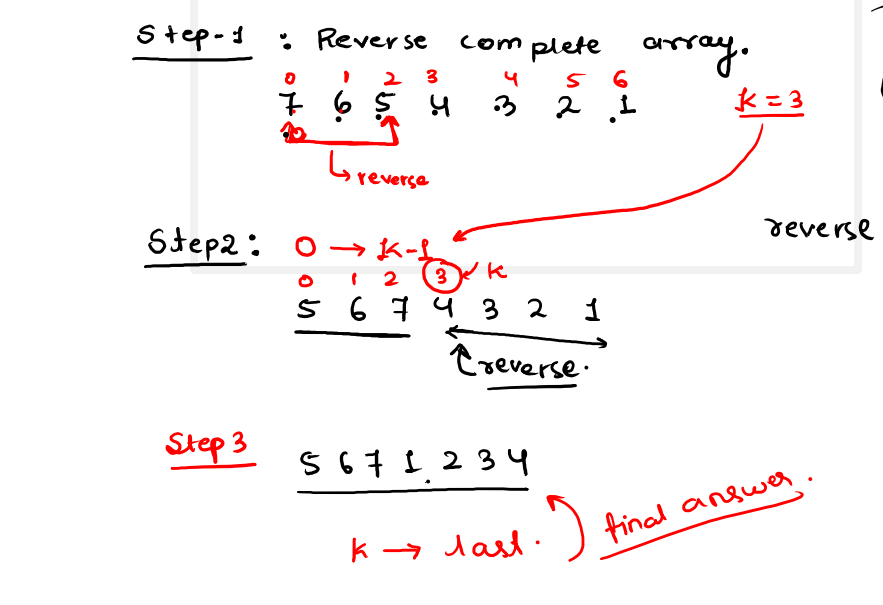
            start++;

            end--;

        }

    }

}



**Code:**

class Solution {

    public void rotate(int[] nums, int k) {

        k=k%nums.length;

        reverse(nums,0,nums.length-1);

        reverse(nums,0,k-1);

        reverse(nums,k,nums.length-1);

    }

    public void reverse(int arr[],int start,int end)

    {

        while(start<end)

        {

            int temp=arr[start];

            arr[start]=arr[end];

            arr[end]=temp;

            start++;

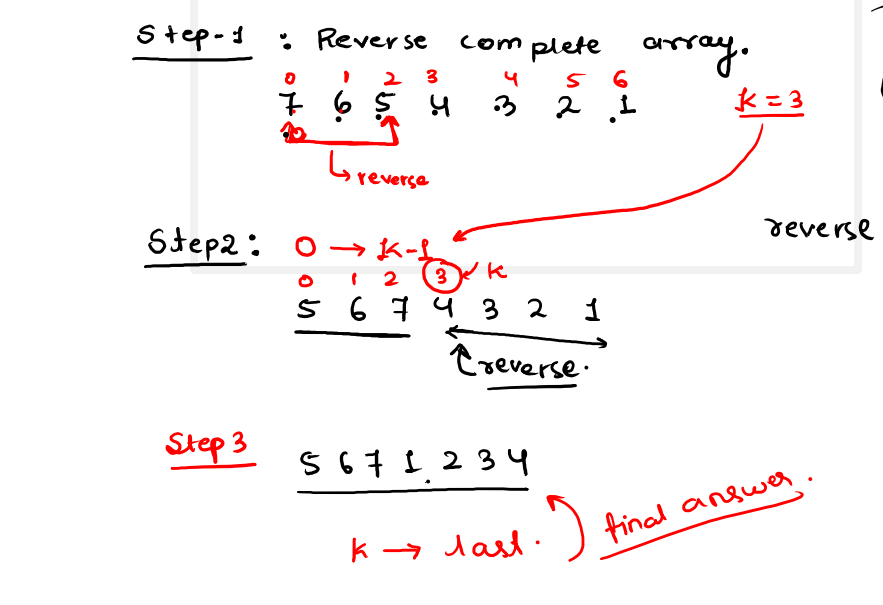
            end--;

        }

    }

}

**Dry Run:**



**Why have we taken k=k%nums.length?**

**Certainly! The line `k = k % nums.length;` is used to handle cases where the rotation value `k` is greater than the length of the array `nums`. This ensures that the rotation is within the bounds of the array, preventing unnecessary rotations that would bring the array back to its original state.**

**Here's an explanation of why this line is necessary:**

**1. If `k` is greater than or equal to `nums.length`, it means you have more rotations than there are elements in the array.**

**2. Taking the modulus (`%`) with `nums.length` effectively reduces `k` to the range `[0, nums.length - 1]`.**

**3. For example, if `nums.length` is 5 and `k` is 7, then `k % nums.length` becomes 2, which is the equivalent rotation within the array bounds.**

**By performing this modulus operation, you ensure that `k` is always within the valid range, preventing unnecessary rotations and making the rotation operation more efficient.**

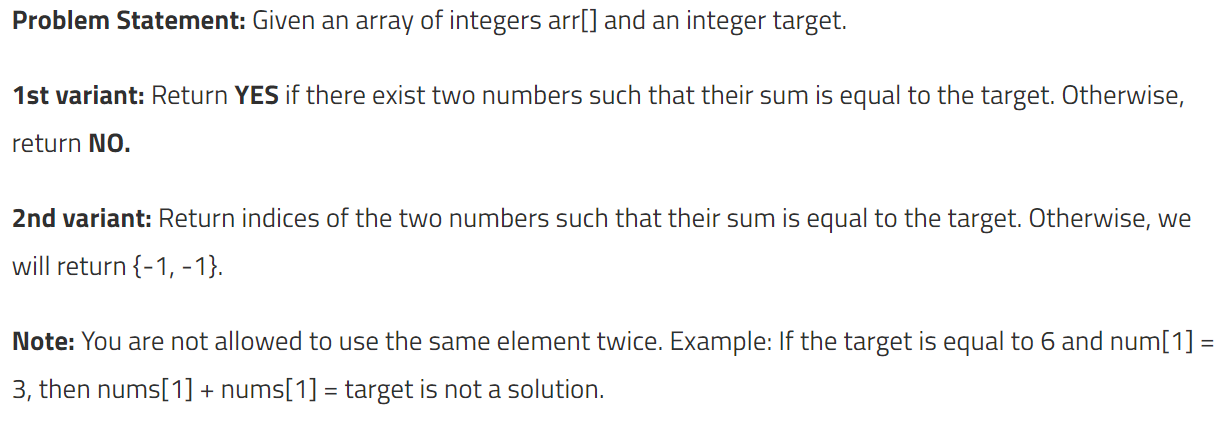
**Time And Space:**

**Time : O(n)**

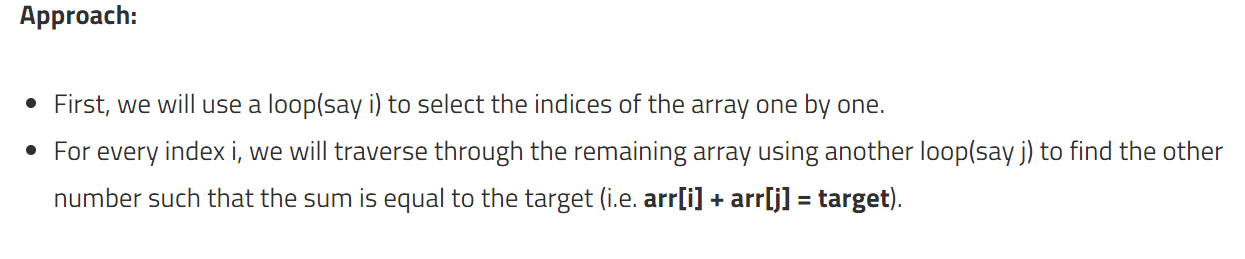
**Space: O(1)**

**Q4) Two Sum problem**

**Problem Statement:**

****

**Brute Force Approach:**

****

**Code:**

import java.util.\*;

public class tUf {

public static String twoSum(int n, int []arr, int target) {

for (int i = 0; i < n; i++) {

for (int j = i + 1; j < n; j++) {

if (arr[i] + arr[j] == target) return "YES";

}

}

return "NO";

}

public static void main(String args[]) {

int n = 5;

int[] arr = {2, 6, 5, 8, 11};

int target = 14;

String ans = twoSum(n, arr, target);

System.out.println("This is the answer for variant 1: " + ans);

}

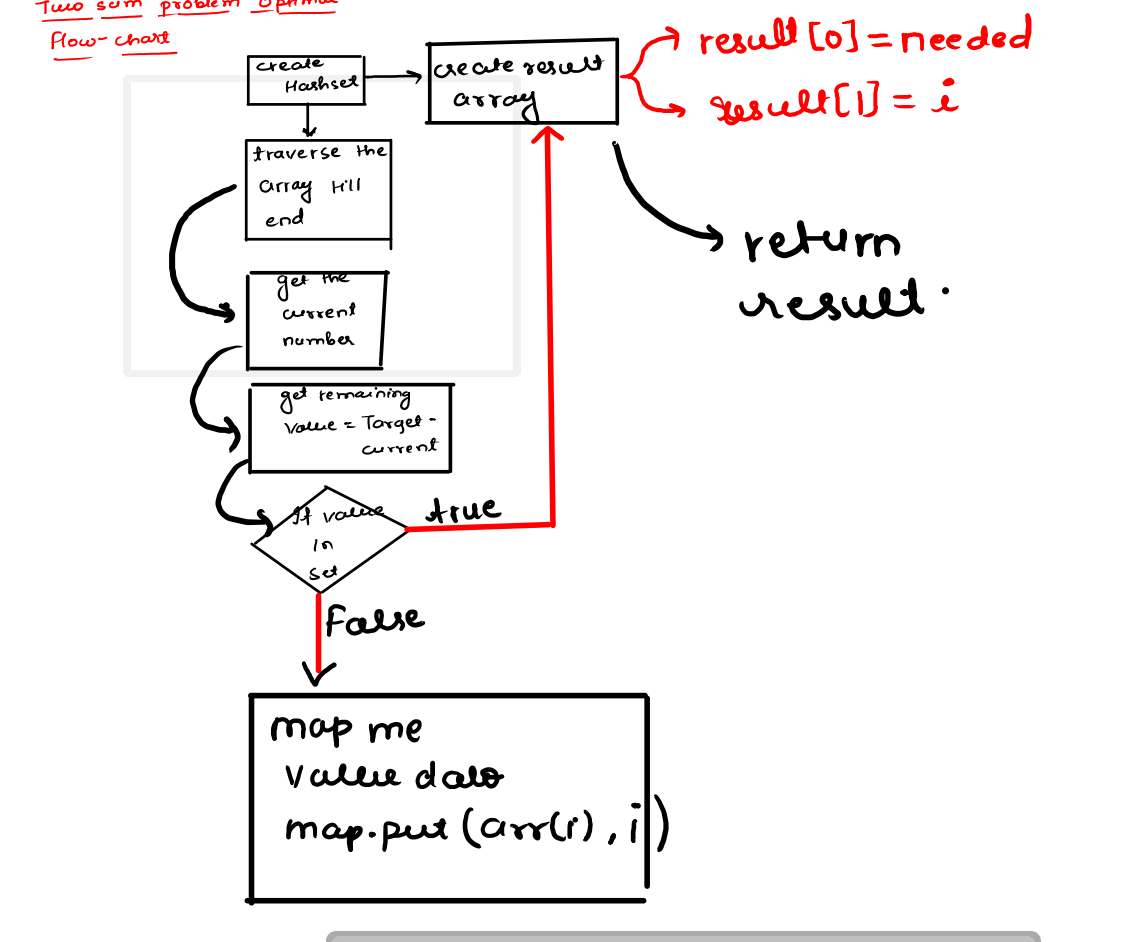
**Dry Run:**

**Time And Space:**

**Time: O(n^2)**

**Space:O(1)**

**Optimal Approach:**

****

**Code:**

        HashMap<Integer,Integer> map = new HashMap<>();

        // For storing the result creating a dummy array

        int res[] = new int[2];

        res[0] = res[1] =-1;

        for(int i=0 ; i<arr.length ; i++)

        {

            int number = arr[i];

            int needed = target - number;

            if(map.containsKey(needed)){

                res[0] = map.get(needed);

                res[1] = i;

                return res;

            }

            map.put(arr[i],i);

        }

        return res;

    }

}

**Dry Run:**

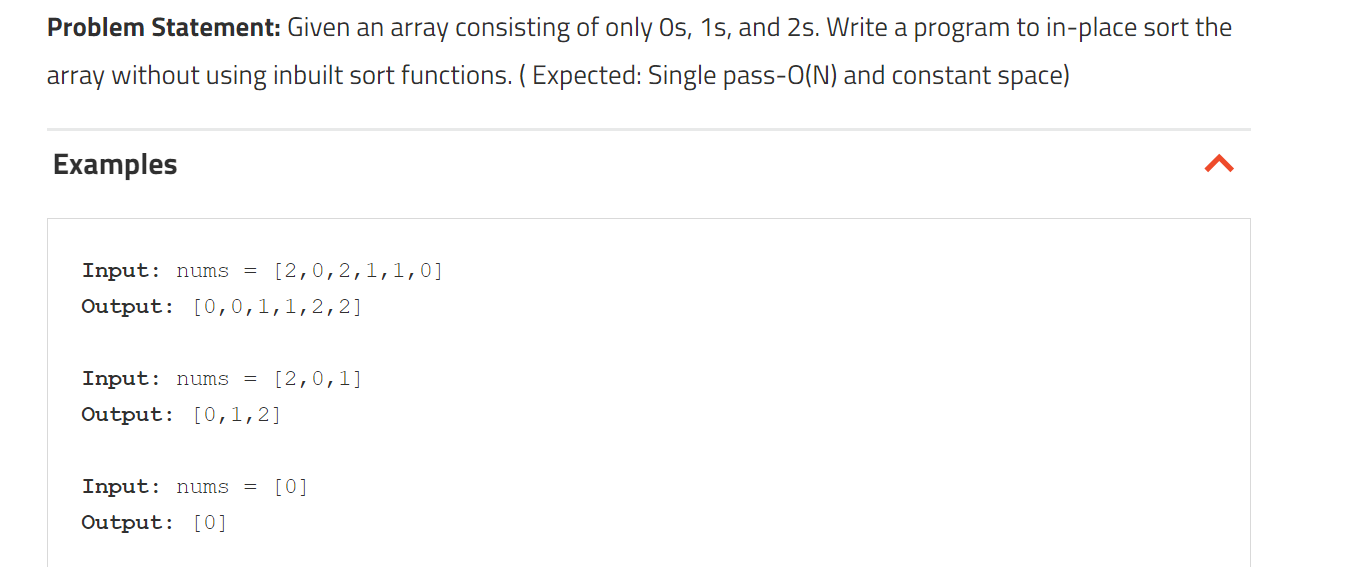
**Time And Space:**

**Time :O(n)**

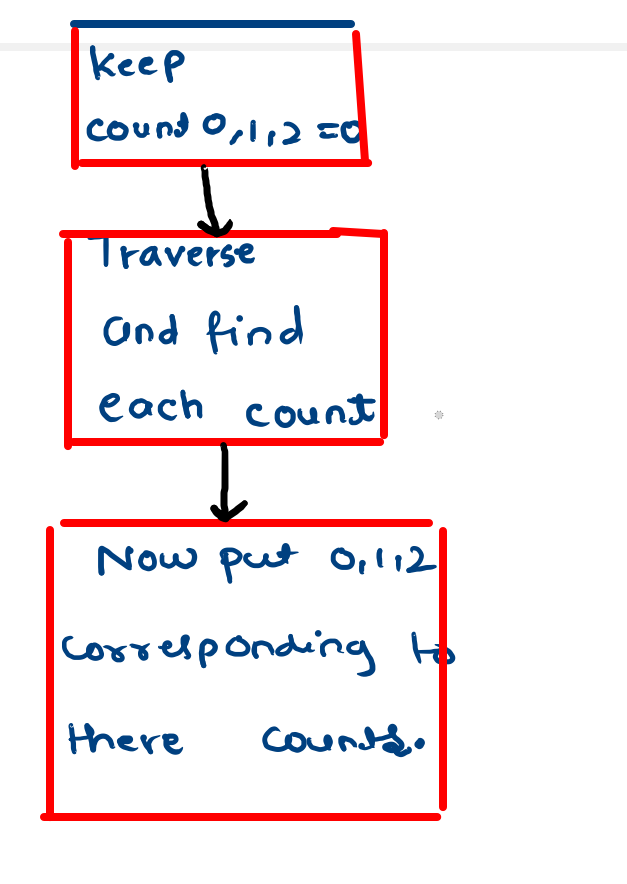
**Space:O(1)**

**Q5) Sort an Array of 0s 1s 2s**

**Problem Statement:**

****

**Brute Force Approach:**

****

**Code:**

**import java.util.\*;**

**public class Main {**

**public static void sortArray(ArrayList<Integer> arr, int n) {**

**int cnt0 = 0, cnt1 = 0, cnt2 = 0;**

**for (int i = 0; i < n; i++) {**

**if (arr.get(i) == 0) cnt0++;**

**else if (arr.get(i) == 1) cnt1++;**

**else cnt2++;**

**}**

**//Replace the places in the original array:**

**for (int i = 0; i < cnt0; i++) arr.set(i, 0); // replacing 0's**

**for (int i = cnt0; i < cnt0 + cnt1; i++) arr.set(i, 1); // replacing 1's**

**for (int i = cnt0 + cnt1; i < n; i++) arr.set(i, 2); // replacing 2's**

**}**

**public static void main(String args[]) {**

**int n = 6;**

**ArrayList<Integer> arr = new ArrayList<>(Arrays.asList(new Integer[] {0, 2, 1, 2, 0, 1}));**

**sortArray(arr, n);**

**System.out.println("After sorting:");**

**for (int i = 0; i < n; i++) {**

**System.out.print(arr.get(i) + " ");**

**}**

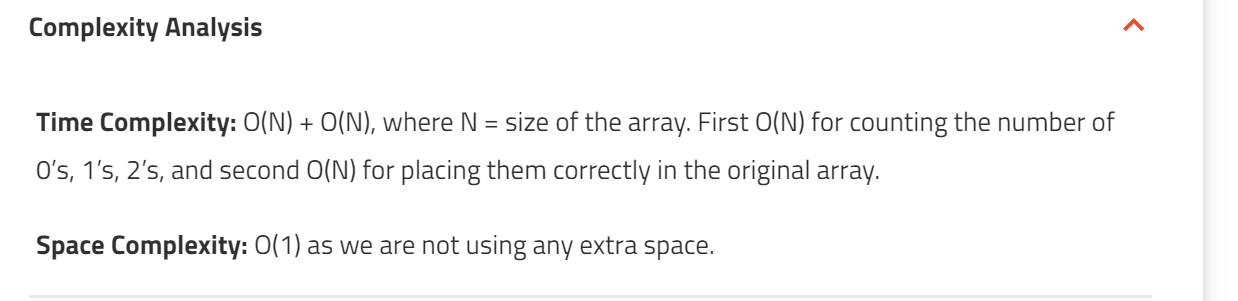
**System.out.println();**

**}**

**}**

**Dry Run:**

**Time And Space:**

****

**Optimal Approach:**

**Code:**

**Dry Run:**

**Time And Space:**