**LC#26:REMOVE DUPLICATES FROM SORTED ARRAY**

**TWO POINTER(OPTIMAL)=>track pointer(i) separately when a new ele occurs->inc(i) and assign nums[i] =nums[j]**

class Solution {

    public int removeDuplicates(int[] nums) {

        int i = 0 , len = nums.length ;

        for(int j = 1 ;j< len;j++)

        {

            if(nums[i]!=nums[j])

            {

                nums[++i] = nums[j];

            }

        }

        return i+1;

    }

}

**LC#27:REMOVE ELEMENT(OPTIMAL) if val == nums[i] increment count and assign which r not equal**

class Solution {

    public int removeElement(int[] nums, int val) {

        int i = 0 , len = nums.length;

        for(int j=0;j< len ;j++)

        {  if(nums[j]!=val)

            {

                nums[i] = nums[j];

                i++;

            }

        }

        return i; }}

**LC#80: REMOVE DUPLICATES FROM SORTED ARRAY II**

**APPROACH:**

**->have two pointers one loop other one updates only when**

**count<2 or nums[i] != nums[j-2]**

**ex:[1,1,2,2,2,3,3,3,4,5,5,5]**

**count only valid for first 2 elements afters that the second condition takes cares of the duplicates**

**CODE:**

class Solution {

    public int removeDuplicates(int[] nums) {

        int count = 0 , j = 0;

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

        {

            if(count<2 || nums[i]!=nums[j-2])

            {

                count++;

                nums[j++] = nums[i];

            }

        }

        return j;

    }

}

**LC#75: SORT COLORS**

**APPROACH 1: count all the 0,1,2 and assign the values accordingly**

**4loops ( 1 count , 3 for putting each number)**

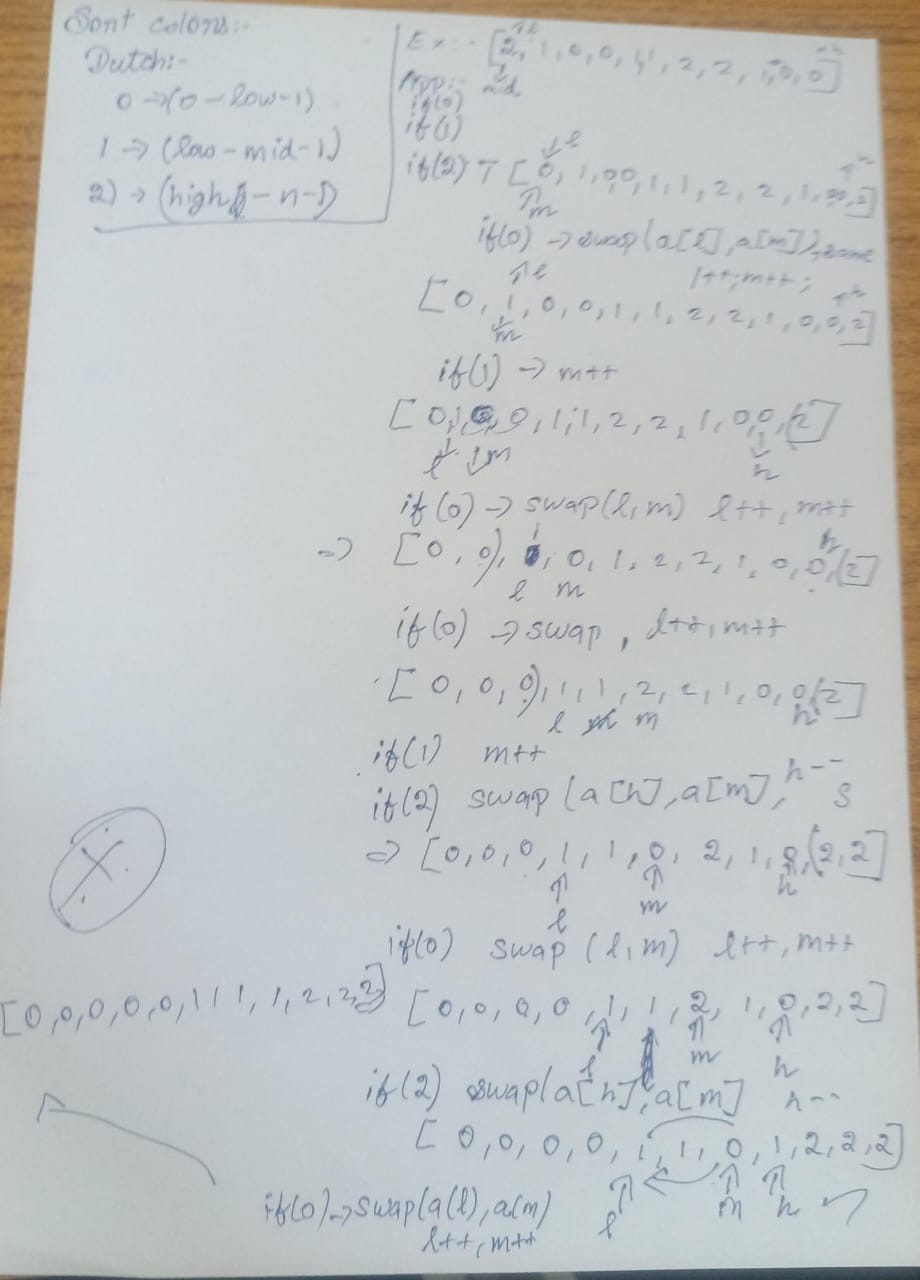
**APPROACH 2:two pointer**

**->DUTCH NATIONAL FLAG PROBLEM**

* **0->low-1 sorted**
* **Low->mid-1 sorted**
* **Mid->high unsorted (use this)**
* **High->h-1 sorted**

**App:**

* **If(arr[mid])==0) -> swap arr[low] and arr[mid] and increment low and mid**
* **Else if(arr[mid]==1) -> increment mid only**
* **Else -> swap(arr[high],arr[mid]) and decrement high**



**Code:**

class Solution {

    public void sortColors(int[] arr) {

        int left = 0 , right = arr.length-1 , mid = 0;

        while(!(mid>right))

        {

            if(arr[mid]==0)

            {

                int temp = arr[mid];

                arr[mid] = arr[left];

                arr[left] = temp;

                left++;

                mid++;

            }

             else if(arr[mid]==1)

            {

                mid++;}

            else

            {   int temp = arr[mid];

                arr[mid] = arr[right];

                arr[right] = temp;

                right--;

            }

}

}

}

**LC#283:MOVE ZEROS**

* **Same as above , if not zero swap and increment otherwise nothing**

class Solution {

    public void moveZeroes(int[] nums) {

        int j = 0;

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

        {

            if(nums[i]!=0)

            {

               int temp = nums[i];

               nums[i] = nums[j];

               nums[j] = temp;

               j++;

            }

        }

    }

}

**LC2337:MOVE PIECES TO OBTAIN STRING:**

class Solution {

    public boolean canChange(String start, String target) {

        int n = start.length();

        int i = 0, j = 0;

        while (i < n || j < n) {

            while (i < n && start.charAt(i) == '\_') i++;

            while (j < n && target.charAt(j) == '\_') j++;

            if (i == n || j == n) return i == n && j == n;

            if (start.charAt(i) != target.charAt(j)) return false;

            if (start.charAt(i) == 'L' && j > i) return false; // L can't move right

            if (start.charAt(i) == 'R' && j < i) return false; // R can't move left

            i++;

            j++;

        }

        return true;

    }

}