DSA SERIES

- Learn Coding

Topic to be Covered today

Two pointers

Problem

LETS START TODAY'S LECTURE

Problem : 2161

Partition Array According to Given Pivot

You are given a **0-indexed** integer array nums and an integer pivot. Rearrange nums such that the following conditions are satisfied:

- Every element less than pivot appears **before** every element greater than pivot.
- Every element equal to pivot appears **in between** the elements less than and greater than pivot.
- The **relative order** of the elements less than pivot and the elements greater than pivot is maintained.
- More formally, consider every p_i, p_j where p_i is the new position of the i^{th} element and p_j is the new position of the j^{th} element. If $i < j$ and **both** elements are smaller (*or larger*) than pivot, then $p_i < p_j$.

Return nums *after the rearrangement.*

Input: nums = [9,12,5,10,14,3,10], pivot = 10

Output: [9,5,3,10,10,12,14]

```cpp
class Solution {
public:
    vector<int> pivotArray(vector<int>& nums, int pivot) {
        int n = nums.size();
        int countLess =0 ;
        int countEqual =0;

        for(int num :nums){
            if(num<pivot){
                countLess++;
            } else if(num==pivot){
                countEqual++;
            }
        }

        int i = 0;
        int j = countLess;
        int k = countLess + countEqual;
```

```
            vector<int> result(n);
            for(int num : nums){
                if(num<pivot){
                    result[i]=num;
                    i++;
                } else if(num == pivot){
                    result[j]=num;
                    j++;
                }else{
                    result[k]=num;
                    k++;
                }
            }

            return result;

        }
    };
```

Problem : 2460

Apply Operations to an Array

You are given a **0-indexed** array nums of size n consisting of **non-negative** integers.

You need to apply n - 1 operations to this array where, in the i^{th} operation (**0-indexed**), you will apply the following on the i^{th} element of nums:

- If nums[i] == nums[i + 1], then multiply nums[i] by 2 and set nums[i + 1] to 0. Otherwise, you skip this operation.

After performing **all** the operations, **shift** all the 0's to the **end** of the array.

- For example, the array [1,0,2,0,0,1] after shifting all its 0's to the end, is [1,2,1,0,0,0].

Return *the resulting array*.

Note that the operations are applied **sequentially**, not all at once.

Input: nums = [1,2,2,1,1,0]
Output: [1,4,2,0,0,0]

```cpp
class Solution {
public:
    vector<int> applyOperations(vector<int>& nums) {
        int n = nums.size();

        // Apply the operation

        for (int i = 0; i < n - 1; i++) {
            if (nums[i] == nums[i + 1]) {
                nums[i] = nums[i] * 2;
                nums[i + 1] = 0;
            }
        }
            int pos=0;
        for(int i =0;i<n;i++){
            if(nums[i] != 0){
                nums[pos]=nums[i];
                pos++;
            }
        }

        // remaining place if it is not completely filled
        while (pos< n) {
            nums[pos]=0;
            pos++;
        }

        return nums;
    }
};
```

Problem : 2200

Find All K-distant Indices in an array

You are given a **0-indexed** integer array nums and two integers key and k.
A **k-distant index** is an index i of nums for which there exists at least one index j such that |i - j| <= k and nums[j] == key.
Return *a list of all k-distant indices sorted in* ***increasing order***.

Input: nums = [3,4,9,1,3,9,5], key = 9, k = 1

Output: [1,2,3,4,5,6]

```cpp
class Solution {
public:
    vector<int> findKDistantIndices(vector<int>& nums, int key, int k) {
        int n = nums.size();
        vector<int> position;
        for(int i =0;i<n;i++){
            if(nums[i]==key){
                position.push_back(i);
            }
        }
        vector<int> result;
        for(int i =0 ;i<n;i++){
            for(int j =0;j<position.size();j++){
                if((abs(i-position[j]) <= k) && nums[position[j]]==key){
                    result.push_back(i);
                    break;
                }
            }
        }
        return result;
    }
};
```

Problem : 75

Sort Colors

```cpp
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int n = nums.size();
        int i =0;
        int j= 0;
        int k =n-1;

        while(j<=k){
            int x = nums[j];
            if(x == 1) j++;
            else if(x==0){
                swap(nums[i],nums[j]);
                i++;
                j++;
            } else {
                swap(nums[j],nums[k]);
                k--;
            }
        }
    }
};
```

Problem : 151

Reverse Words In a string

```cpp
class Solution {
public:
    string reverseWords(string s) {
        reverse(begin(s),end(s));
        int i =0;
        int l = 0,r=0;
        int n =s.length();

        while(i<n){
            while(i<n && s[i] != ' '){
                s[r++]=s[i++];
            }

            if(l<r){
                reverse(s.begin()+l ,s.begin()+r);
                s[r]=' ';
                r++;
                l=r;
            }
            i++;
        }

        s= s.substr(0,r-1);

        return s;
    }
};
```

Problem : 125

Valid Palindrome

```cpp
class Solution {
public:
    bool isPalindrome(string s) {
        int left = 0;
        int right = s.length() - 1;
        while (left < right) {
            while (left < right && !isalnum(s[left]))
                left++;
            while (left < right && !isalnum(s[right]))
                right--;

            if (tolower(s[left]) != tolower(s[right])) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
};
```

Learn coding

THANK YOU