DSA SERIES

- Learn Coding

Topic to be Covered today

Two pointers Approach

LETS START TODAY'S LECTURE

Two pointers are like two soldiers who march from either:

- **Start and end** of the array (opposite directions), or
- **Start and next** element (same direction)

They help **efficiently search or solve problems** that involve:

- Sorted arrays
- Linked lists
- Strings
- Searching for pairs/subarrays, etc.

When to use Two pointers

1. The array or string is **sorted** (mostly).
2. You’re trying to find:
    1. A **pair or subarray** with a given condition.
    2. **Duplicates** or something related to **gap**, **sum**, **difference**, etc.
3. You want to **reduce nested loops** into a single loop.

Types of Two Pointer Techniques

1. Opposite Direction

2. Same Direction

Problem : 2570

Merge Two 2D Arrays by Summing Values

You are given two **2D** integer arrays nums1 and nums2.

- nums1[i] = [id_i, val_i] indicate that the number with the id id_i has a value equal to val_i.
- nums2[i] = [id_i, val_i] indicate that the number with the id id_i has a value equal to val_i.

Each array contains **unique** ids and is sorted in **ascending** order by id.

Merge the two arrays into one array that is sorted in ascending order by id,
respecting the following conditions:

- Only ids that appear in at least one of the two arrays should be included in the resulting array.
- Each id should be included **only once** and its value should be the sum of the values of this id in the two arrays.
  If the id does not exist in one of the two arrays, then assume its value in that array to be 0.

Return *the resulting array*. The returned array must be sorted in ascending order by id.

```
class Solution {
public:
    vector<vector<int>> mergeArrays(vector<vector<int>>&nums1,
                                    vector<vector<int>>& nums2) {
        int i = 0;
        int j = 0;

        vector<vector<int>> result;

        while (i < nums1.size() && j < nums2.size()) {
            if (nums1[i][0] == nums2[j][0]) {
                result.push_back({nums1[i][0], nums1[i][1] + nums2[j][1]});
                i++;
                j++;
            }

            else if (nums1[i][0] < nums2[j][0]) {
                result.push_back(nums1[i]);
                i++;
            } else {
                result.push_back(nums2[j]);
                j++;
            }
        }

        while (i < nums1.size()) {
            result.push_back(nums1[i]);
            i++;
        }

        while (j < nums2.size()) {
            result.push_back(nums2[j]);
            j++;
        }

        return result;
    }
};
```

Problem : 905

Sort Array By parity

Given an integer array nums, move all the even integers at the beginning of the array followed by all the odd integers.

Return ***any array*** *that satisfies this condition*.

```
class Solution {
public:
    vector<int> sortArrayByParity(vector<int>& nums) {
        int n  = nums.size();
        int i = 0;
        int j = 0;
        while(j<n){
            if(nums[j] % 2==0){
                swap(nums[i],nums[j]);
                i++;
            }
            j++;
        }
        return nums;
    }
};
```

Problem : 922

Sort Array By parity II

Given an array of integers nums, half of the integers in nums are **odd**, and the other half are **even**.
Sort the array so that whenever nums[i] is odd, i is **odd**, and whenever nums[i] is even, i is **even**.
Return *any answer array that satisfies this condition*.

```cpp
class Solution {
public:
    vector<int> sortArrayByParityII(vector<int>& nums) {
        int n = nums.size();
        int i = 0;
        int j = 1;
        while (i < n) {
            while (i < n && nums[i] % 2 == 0) {
                i += 2;
            }
            while (j < n && nums[j] % 2 == 1) {
                j += 2;
            }
            if (i < n) {
                swap(n ums[i], nums[j]);
                i += 2;
                j += 2;
            }
        }
        return nums;
    }
};
```

Problem : 2149

Rearrange Array Elements by Sign

You are given a **0-indexed** integer array nums of **even** length consisting of an **equal** number of positive and negative integers.

You should return the array of nums such that the the array follows the given conditions:

1. Every **consecutive pair** of integers have **opposite signs**.
2. For all integers with the same sign, the **order** in which they were present in nums is **preserved**.
3. The rearranged array begins with a positive integer.

Return *the modified array after rearranging the elements to satisfy the aforementioned conditions*

```cpp
class Solution {
public:
    vector<int> rearrangeArray(vector<int>& nums) {
        int n = nums.size();
        int pi = 0;
        int ni = 1;

        vector<int> result(n);
        for (int& num : nums) {
            if (num > 0) {
                result[pi] = num;
                pi += 2;
            } else {
                result[ni] = num;
                ni += 2;
            }
        }

        return result;
    }
};
```

Learn coding

THANK YOU