2570. Merge Two 2D Arrays by Summing Values

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

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

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.

Example 1:

```
Input: nums1 = [[1,2],[2,3],[4,5]], nums2 = [[1,4],[3,2],[4,1]]
Output: [[1,6],[2,3],[3,2],[4,6]]
Explanation: The resulting array contains the following:
- id = 1, the value of this id is 2 + 4 = 6.
- id = 2, the value of this id is 3.
- id = 3, the value of this id is 2.
- id = 4, the value of this id is 5 + 1 = 6.
```

Example 2:

```
Input: nums1 = [[2,4],[3,6],[5,5]], nums2 = [[1,3],[4,3]]
Output: [[1,3],[2,4],[3,6],[4,3],[5,5]]
Explanation: There are no common ids, so we just include each id with its value in
the resulting list.
```

Constraints:

- `1 <= nums1.length, nums2.length <= 200`
- `nums1[i].length == nums2[j].length == 2`
- `1 <= idi, vali <= 1000`
- Both arrays contain unique ids.
- Both arrays are in strictly ascending order by id.

2570. Merge Two 2D Arrays by Summing Values

Overview

We are given two arrays, `nums1` and `nums2`, each containing pairs of the form `{id, value}`. These pairs represent mappings where `id` is unique within each array, and both arrays are sorted in ascending order based on `id`.

Our goal is to merge these two arrays of pairs into a single sorted array of pairs. Each entry in the result should correspond to an `id` that appears in either input array. If an `id` is present in both arrays, we sum the associated values; otherwise, we keep the existing pair as is. The final output must be sorted by `id`.

Examples:

1. If `nums1 = [(id1, val1)]` and `nums2 = [(id2, val2)]` with `id1 < id2`, then the final array should be `[(id1, val1), (id2, val2)]`.
2. If `nums1 = [(id1, val1)]` and `nums2 = [(id1, val2)]`, then the final array should be `[(id1, val1 + val2)]`.
3. If `nums1 = [(id1, val1), (id2, val2)]` and `nums2 = [(id2, val3)]` with `id1 < id2`, then the final array should be `[(id1, val1), (id2, val2 + val3)]`.

2570. Merge Two 2D Arrays by Summing Values

```
/*
    Two pointers
    Time complexity: O(2(n+m))
    Space complexity: O(n+m)
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;
class Solution {
    public:
        vvi mergeArrays(vvi& nums1, vvi& nums2) {
            int n=nums1.size();
            int m=nums2.size();
            vvi ans;
            int i=0,j=0;
            while(i<n && j<m){
                if(nums1[i][0]<nums2[j][0]){
                    ans.push_back(nums1[i]);
                    i++;
                }
                else if(nums1[i][0]>nums2[j][0]){
                    ans.push_back(nums2[j]);
                    j++;
                }
                else{
                    ans.push_back({nums1[i][0],nums1[i][1]+nums2[j][1]});
                    i++;
                    j++;
                }
            }

            while(i<n) ans.push_back(nums1[i++]);
            while(j<m) ans.push_back(nums2[j++]);

            return ans;
        }
};
```