Union Of Two Sorted Arrays With Distinct Elements

solved by	Senan
Platform	GeeksForGeeks
difficulty	Easy
tags	Sorting Union
language	C++
solved on	@10/11/2024
link	https://www.geeksforgeeks.org/problems/union-of-two-sorted-arrays-with-distinct-elements/1
Completion	✓

Intuition

The goal is to find the union of two sorted arrays, `a` and `b`, without including any duplicates in the result. Since the arrays are sorted, we can use a two-pointer approach to efficiently traverse and compare elements in both arrays.

Approach

1. **Two-pointer Technique**: Use two pointers, `i` and `j`, initialized to the start of arrays `a` and `b`, respectively. The pointers will move through each array, comparing elements and adding them to the result vector.
2. **Handling Duplicates**: While adding elements to the result, we check if the last element in the result vector (`answer.back()`) is the same as the current element. If it is, we skip adding to avoid duplicates.
3. **Traverse Both Arrays**:
   - If `a[i]` is less than `b[j]`, add `a[i]` to `answer` and move pointer `i` forward.
   - If `a[i]` is greater than `b[j]`, add `b[j]` to `answer` and move pointer `j` forward.
   - If `a[i]` equals `b[j]`, add only one of them to `answer` to avoid duplicates, then move both pointers forward.
4. **Remaining Elements**: After one array is fully traversed, add any remaining elements from the other array to `answer`, ensuring no duplicates.

Complexity

Time Complexity:

The time complexity is **O(m + n)**, where **m** and **n** are the lengths of arrays `a` and `b`. This is because each element in both arrays is processed once.

Space Complexity:

The space complexity is **O(m + n)**, where **m** and **n** are the lengths of arrays `a` and `b`. This is due to the storage needed for the result vector `answer`.

Code

```
class Solution {
public:
    vector<int> findUnion(vector<int>& a, vector<int>& b) {
        vector<int> answer;
        int i = 0, j = 0;

        while (i < a.size() && j < b.size()) {
            if (a[i] < b[j]) {
                if (answer.empty() || answer.back() < a[i]) {
                    answer.push_back(a[i]);
                }
                i++;
            } else if (a[i] > b[j]) {
                if (answer.empty() || answer.back() < b[j]) {
                    answer.push_back(b[j]);
                }
                j++;
            } else {
                if (answer.empty() || answer.back() < a[i]) {
                    answer.push_back(a[i]);
                }
                i++;
                j++;
            }
        }

        while (i < a.size()) {
            if (answer.empty() || answer.back() < a[i]) {
                answer.push_back(a[i]);
            }
            i++;
        }

        while (j < b.size()) {
            if (answer.empty() || answer.back() < b[j]) {
                answer.push_back(b[j]);
            }
            j++;
        }

        return answer;
    }
};
```