Common Elements

Easy Accuracy: 35.98% Submissions: 29K+ Points: 2

Given two **lists V1 and V2 of sizes n and m** respectively. Return the list of elements common to both the lists and return the list in sorted order. Duplicates may be there in the output list.

Example:

Input:
n = 5
v1[] = {3, 4, 2, 2, 4}
m = 4
v2[] = {3, 2, 2, 7}
Output:
2 2 3
Explanation:
The common elements in sorted order are {2 2 3}

C++ (g++ 5.4) Start Timer

```
// } Driver Code Ends
//Back-end complete function Template for C++

class Solution{
    public:
        vector<int> common_element(vector<int>v1,vector<int>v2)
    {
        // Your code here
        sort(v1.begin() , v1.end());
        sort(v2.begin() , v2.end());

        int n1 = v1.size();
        int n2 = v2.size();
        int i = 0 , j = 0 ;
        vector<int>v;

        while( i <n1 && j<n2){
            if(v1[i] == v2[j]){
                v.push_back(v1[i]);
                i++; j++;
            }else if(v1[i] < v2[j]){
                i++;
            }else{
                j++;
            }
        }
        return v;
    }
};
} // } Driver Code Ends
```

Custom Input | Compile & Run | Submit

Description | Discussion (58) | Solutions (4.1K) | Submissions

31. Next Permutation

Medium | 14.4K | 4K

Companies

A **permutation** of an array of integers is an arrangement of its members into a sequence or linear order.

- For example, for `arr = [1,2,3]`, the following are all the permutations of `arr`: `[1,2,3]`, `[1,3,2]`, `[2, 1, 3]`, `[2, 3, 1]`, `[3,1,2]`, `[3,2,1]`.

The **next permutation** of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the **next permutation** of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

- For example, the next permutation of `arr = [1,2,3]` is `[1,3,2]`.
- Similarly, the next permutation of `arr = [2,3,1]` is `[3,1,2]`.
- While the next permutation of `arr = [3,2,1]` is `[1,2,3]` because `[3,2,1]` does not have a lexicographical larger rearrangement.

Given an array of integers `nums`, *find the next permutation of* `nums`.

The replacement must be **in place** and use only constant extra memory.

C++ | Auto

```cpp
class Solution {
public:
    void nextPermutation(vector<int>& a) {
        int index = -1;
        int n = a.size();
        for(int i=n-1;i>0;i--){
            if(a[i]>a[i-1]){
                index = i;
                break;
            }
        }
        if(index == -1){
            reverse(a.begin(),a.end());
        }
        else{
            int prev = index;
            for(int i=index+1;i<n;i++){
                if(a[i] > a[index-1] and a[i] <= a[prev]){
                    prev = i;
                }
            }
            swap(a[index-1],a[prev]);
            reverse(a.begin()+index,a.end());
        }

    }
```


Console | Run | Submit

Array Subset of another array

Easy Accuracy: 44.05% Submissions: 175K+ Points: 2

Given two arrays: **a1[0..n-1]** of size **n** and **a2[0..m-1]** of size **m**. Task is to check whether a2[] is a subset of a1[] or not. Both the arrays can be sorted or unsorted.

Example 1:

Input:
a1[] = {11, 1, 13, 21, 3, 7}
a2[] = {11, 3, 7, 1}
Output:
Yes
Explanation:
a2[] is a subset of a1[]

C++ (g++ 5.4) Average Time: 20m Start Timer

```cpp
// } Driver Code Ends


string isSubset(int a1[], int a2[], int n, int m) {
    unordered_map<int , int>mp;
    for(int i=0;i<n;i++){
        mp[a1[i]]++;
    }
    for(int i=0;i<m;i++){
        if(mp[a2[i]]==0){
            return "No";
        } else {
            mp[a2[i]]--;
        }
    }
    return "Yes";
}
```

Custom Input Compile & Run Submit