

14.01.2025

## Problem:

1. two integers A and B of length n
2. a prefix common array of A and B is an array C such that **C[i] is equal to count of numbers** that are present at or before the index in both A and B.
3. return the prefix common array of A and B.
4. A[i] and B[i] belongs to [1, n].

### Leetcode daily problem 14th jan

14 January 2025 13:13

0 1 2 3  
A = [1, 3, 2, 4]  
B = [3, 1, 2, 4]

index	C[i]
0	0
1	[1, 3] = 2
2	[1, 3, 2] = 3
3	[1, 3, 2, 4] = 4

Count of common

0

2

3

4

result

we have to return an array which stores the count of common elements before its index

# Brute-force approach

```
for(int i = 0; i < n; i++){
    count = 0;
    for(int j = 0; j <= i; j++){
        for(int k = 0; k <= i; k++){
            if(B[k] == A[j]){
                count++;
                break;
            }
        }
    }
    res.push_back(count);
}
```

time complexity:  $O(n^3)$

space complexity:  $O(1)$

## Better approach

let us say that if  $i = 2$ , that is in third iteration, if we have to search 1 of A array in B array, we will store that we had found 1 in range 0 to  $i$  index.

using boolean array

isPresentA  
isPresentB

initially both the arrays will be marked as `false`

size of both arrays would be  $n+1$  where  $n$  is the size of A and B, kyuki elements would range from 1 to  $n$ .

for  $i = 0$ , mark isPresenA[A[0]] as `true` and also mark isPresenB[B[0]] as `true`

now traverse both the boolean arrays and check whether both arrays are marked as `true`.

for  $i = 1$ , mark isPresenA[A[1]] as `true` and also mark isPresenB[B[1]] as `true`

now we will have,

isPresentA = {F, T, F, T, F}  
isPresentB = {F, T, F, T, F}

therefore traversing both the array again and incrementing the count where `true` in both.  
count = 2

for  $i = 2$ , mark isPresenA[A[2]] as `true` and also mark isPresenB[B[2]] as `true`

now we will have,

isPresentA = {F, T, T, T, F}  
isPresentB = {F, T, T, T, F}

therefore traversing both the array again and incrementing the count where `true` in both.

count = 3

for i = 3, mark isPresentA[A[3]] as `true` and also mark isPresentB[B[3]] as `true`

now we will have,

`isPresentA` = {F, T, F, T, T}

`isPresentB` = {F, T, F, T, T}

therefore traversing both the array again and incrementing the count where `true` in both.

count = 4

```
for (int i = 0; i < n; i++){
    isPresentA[A[i]] = true;
    isPresentB[B[i]] = true;
    int count = 0;
    for(int j = 1; j <= n; j++){
        if(isPresentA[j] == true && isPresentB[j] == true){
            count++;
        }
    }
    result[i] = count;
}
// time complexity: O(n^2)
// space complexity: O(2 * (n + 1))
```

## Optimal Solution:

the logic behind the solution:

since we are given that there would be no duplicates in both the arrays also the elements would also belong to 1 to n both inclusive, this ensures that the count of an element would be at max 2.

therefore, after adding both the elements to the map, we would check that if its frequency is equal to 2 or not, if it is then we would increase the count.

we will use map data structure to store count of element with key as element.

dry run

Optimal

A = <1, 3, 4, 4>      B = <3, 1, 2, 4>

i=0; A[0] & B[0] both does not exist in map  $\therefore$  just add them

i=1; A[1] & B[1] already exist in the map  $\therefore$  increment the frequency also increment the counter & store it to array.

At i: count

(0, 2) present

i=2; A[2] = 4, B[2] = 2

add 4 maps increment in map as well as in count

Count = 3

<0, 2, 3>

i=3; A[3] = 4, B[3] = 4

count = 4

4 exists in both & it would be added to map

<0, 2, 3, 4>

using map:

```
class Solution {
public:
    vector<int> findThePrefixCommonArray(vector<int>& A, vector<int>& B) {
        int n = A.size();
        unordered_map<int, int> ump; // to store the element and frequency.
        int count = 0;
        vector<int> res(n);
        for(int i = 0; i < n; i++){
            // add current element to map.
            ump[A[i]]++;
            // no duplicates in an array therefore at max an element could be
            // 2 times
            if(ump[A[i]] == 2){
                count++;
            }
            ump[B[i]]++;
            if(ump[B[i]] == 2){
                count++;
            }
            res[i] = count;
        }
        return res;
    }
};
```

using frequency array:

```
class Solution {
public:
    vector<int> findThePrefixCommonArray(vector<int>& A, vector<int>& B) {
        int n = A.size();
        int count = 0;
```

```
vector<int>res(n), freq(n+1, 0);
for(int i = 0; i < n; i++){
    // add current element to map.
    freq[A[i]]++;
    // no duplicates in an array therefore at max an element could be
    // 2 times
    if(freq[A[i]] == 2){
        count++;
    }
    freq[B[i]]++;
    if(freq[B[i]] == 2){
        count++;
    }
    res[i] = count;
}
return res;
}
};
```