1460. Make Two Arrays Equal by Reversing Subarrays

You are given two integer arrays of equal length `target` and `arr`. In one step, you can select any **non-empty subarray** of `arr` and reverse it. You are allowed to make any number of steps.

Return `true` *if you can make* `arr` *equal to* `target` *or* `false` *otherwise*.

Example 1:

```
Input: target = [1,2,3,4], arr = [2,4,1,3]
Output: true
Explanation: You can follow the next steps to convert arr to target:
1- Reverse subarray [2,4,1], arr becomes [1,4,2,3]
2- Reverse subarray [4,2], arr becomes [1,2,4,3]
3- Reverse subarray [4,3], arr becomes [1,2,3,4]
There are multiple ways to convert arr to target, this is not the only way to do
so.
```

Example 2:

```
Input: target = [7], arr = [7]
Output: true
Explanation: arr is equal to target without any reverses.
```

Example 3:

```
Input: target = [3,7,9], arr = [3,7,11]
Output: false
Explanation: arr does not have value 9 and it can never be converted to target.
```

Constraints:

- `target.length == arr.length`
- `1 <= target.length <= 1000`
- `1 <= target[i] <= 1000`
- `1 <= arr[i] <= 1000`

1460. Make Two Arrays Equal by Reversing Subarrays

```
/*
    Time complexity: O(1000+1000+n+1000)=O(n)
    Space complexity: O(1000+1000)=O(1)
*/
class Solution {
public:
    bool canBeEqual(std::vector<int>& target, std::vector<int>& arr) {
        int n=arr.size();

        std::vector<int> count_arr(1001,0);
        std::vector<int> count_target(1001,0);

        for(int i=0;i<n;++i){
            count_arr[arr[i]]++;
            count_target[target[i]]++;
        }

        return count_arr==count_target;
    }
};
```

1460. Make Two Arrays Equal by Reversing Subarrays

```
/*
    Time complexity: O(1000+1000+n+n+1000)=O(n)
    Space complexity: O(1000+1000)=O(1)
*/
class Solution {
public:
    bool canBeEqual(std::vector<int>& target, std::vector<int>& arr) {
        int n=arr.size();

        std::vector<int> count(1001,0);
        std::vector<int> should_be(1001,0);

        for(int i=0;i<n;++i){
            count[arr[i]]++;
            should_be[arr[i]]=count[arr[i]]*2;
        }

        for(int i=0;i<n;++i){
            count[target[i]]++;
        }

        for(int i=0;i<=1000;++i){
            if(count[i]!=should_be[i]) return false;
        }

        return true;

    }
};
```