Make Two Arrays Equal by Reversing Subarrays

Description

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

Algorithm

1.) **Check Lengths**:

- First, check if the lengths of the target and arr arrays are different. If they are, return false since arrays of different lengths cannot be made equal by sorting.

2.) **Sort Both Arrays**:

- Sort both the target and arr arrays. Sorting rearranges the elements in both arrays in ascending order.

3.) **Compare Arrays**:

- Use Arrays.equals to check if the sorted arrays are identical. If they are, return true; otherwise, return false.

Pseudocode

```
function canBeEqual(target: array of int, arr: array of int) -> boolean:
    if length of target != length of arr:
        return false
    sort(target)
    sort(arr)
    return arraysEqual(target, arr)
```

Code

```
class Solution {
    public boolean canBeEqual(int[] target, int[] arr) {
        if(target.length != arr.length){
            return false;
        }
        Arrays.sort(target);
        Arrays.sort(arr);
        return Arrays.equals(target, arr);
    }
}
```

Conclusion

The canBeEqual function determines whether two arrays can be made equal by sorting them. It first checks if the arrays have the same length, and if not, returns false. If they do, it sorts both arrays and then compares them. If the sorted arrays are identical, the function returns true, indicating that the arrays can be made equal by sorting; otherwise, it returns false. This approach leverages sorting and comparison to efficiently determine if the two arrays can be made identical.