Problem | Editorial | Submissions

Count all triplets with given sum in sorted array

Difficulty: **Medium** Accuracy: **48.57%** Submissions: **48K+** Points: **4**

Given a sorted array **arr[]** and a **target** value, the task is to count triplets **(i, j, k)** of valid indices, such that **arr[i] + arr[j] + arr[k] = target** and **i < j < k**.

Examples:

Input: arr[] = [-3, -1, -1, 0, 1, 2], target = -2
Output: 4
Explanation: Four triplets that add up to -2 are:
arr[0] + arr[3] + arr[4] = (-3) + 0 + (1) = -2
arr[0] + arr[1] + arr[5] = (-3) + (-1) + (2) = -2
arr[0] + arr[2] + arr[5] = (-3) + (-1) + (2) = -2
arr[1] + arr[2] + arr[3] = (-1) + (-1) + (0) = -2

Input: arr[] = [-2, 0, 1, 1, 5], target = 1
Output: 0
Explanation: There is no triplet whose sum is equal to 1.

Constraints:

$3 < arr.size() < 10^4$

Java (1.8) | Start Timer

```java
// } Driver Code Ends
class Solution {
    public int countTriplets(int[] arr, int target) {
        Map<Integer, Integer> freq = new HashMap<>();
        for (int e : arr) {
            freq.put(e, freq.getOrDefault(e, 0) + 1);
        }
        int ans = 0;
        for (int i = 0; i < arr.length; i++) {
            freq.put(arr[i], freq.get(arr[i]) - 1);
            for (int j = 0; j < i; j++) {
                int lookfor = target - arr[i] - arr[j];
                if (freq.containsKey(lookfor)) {
                    ans += freq.get(lookfor);
                }
            }
        }
        return ans;
    }
}
```

Custom Input | Compile & Run | Submit