Count triplets with sum smaller than a given value

Given an array of distinct integers and a sum value. Find count of triplets with sum smaller than given sum value. Expected Time Complexity is $O(n^2)$.

Examples:

```
Input : arr[] = {-2, 0, 1, 3}
        sum = 2.
Output : 2
Explanation :  Below are triplets with sum less than 2
               (-2, 0, 1) and (-2, 0, 3)

Input : arr[] = {5, 1, 3, 4, 7}
        sum = 12.
Output : 4
Explanation :  Below are triplets with sum less than 4
               (1, 3, 4), (1, 3, 5), (1, 3, 7) and
               (1, 4, 5)
```

We strongly recommend you to minimize your browser and try this yourself first.

A **Simple Solution** is to run three loops to consider all triplets one by one. For every triplet, compare the sums and increment count if triplet sum is smaller than given sum.

```
// A Simple C++ program to count triplets with sum smaller
// than a given value
#include<bits/stdc++.h>
using namespace std;

int countTriplets(int arr[], int n, int sum)
{
    // Initialize result
    int ans = 0;

    // Fix the first element as A[i]
    for (int i = 0; i < n-2; i++)
    {
       // Fix the second element as A[j]
       for (int j = i+1; j < n-1; j++)
       {
           // Now look for the third number
```

```
            for (int k = j+1; k < n; k++)
                if (arr[i] + arr[j] + arr[k] < sum)
                    ans++;
        }
    }

    return ans;
}

// Driver program
int main()
{
    int arr[] = {5, 1, 3, 4, 7};
    int n = sizeof arr / sizeof arr[0];
    int sum = 12;
    cout << countTriplets(arr, n, sum) << endl;
    return 0;
}
```

Run on IDE

Output:

```
4
```

Time complexity of above solution is $O(n^3)$. An **Efficient Solution** can count triplets in $O(n^2)$ by sorting the array first, and then using method 1 of this post in a loop.

```
1) Sort the input array in increasing order.
2) Initialize result as 0.
3) Run a loop from i = 0 to n-2.  An iteration of this loop finds all
   triplets with arr[i] as first element.
     a) Initialize other two elements as corner elements of subarray
        arr[i+1..n-1], i.e., j = i+1 and k = n-1
     b) Move j and k toward each other until they meet, i.e., while (j < k)
            (i) if (arr[i] + arr[j] + arr[k] >= sum), then do k--

            // Else for current i and j, there can (k-j) possible third elements
            // that satisfy the constraint.
            (ii) Else Do ans += (k - j) followed by j++
```

Below is C++ implementation of above idea.

```
// C++ program to count triplets with sum smaller than a given value
#include<bits/stdc++.h>
using namespace std;

int countTriplets(int arr[], int n, int sum)
{
    // Sort input array
    sort(arr, arr+n);

    // Initialize result
    int ans = 0;

    // Every iteration of loop counts triplet with
```

```
    // first element as arr[i].
    for (int i = 0; i < n - 2; i++)
    {
        // Initialize other two elements as corner elements
        // of subarray arr[j+1..k]
        int j = i + 1, k = n - 1;

        // Use Meet in the Middle concept
        while (j < k)
        {
            // If sum of current triplet is more or equal,
            // move right corner to look for smaller values
            if (arr[i] + arr[j] + arr[k] >= sum)
                k--;

            // Else move left corner
            else
            {
                // This is important. For current i and j, there
                // can be total k-j third elements.
                ans += (k - j);
                j++;
            }
        }
    }
    return ans;
}

// Driver program
int main()
{
    int arr[] = {5, 1, 3, 4, 7};
    int n = sizeof arr / sizeof arr[0];
    int sum = 12;
    cout << countTriplets(arr, n, sum) << endl;
    return 0;
}
```

Run on IDE

Output:

```
4
```

Thanks to Gaurav Ahirwar for suggesting this solution.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

16 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length
- Convert array into Zig-Zag fashion

(Login to Rate and Mark)

2.5 Average Difficulty : **2.5/5.0**
Based on **13** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.