Count triplets with sum smaller than X

Friday, December 24, 2021 5:31 PM

Count triplets with sum smaller than X

Medium Accuracy: 49.96% Submissions: 25677 Points: 4

Given an array **arr[]** of distinct integers of size **N** and a value **sum**, the task is to find the count of triplets **(i, j, k)**, having **(i<j<k)** with the sum of **(arr[i] + arr[j] + arr[k])** smaller than the given value sum.

Example 1:

Input: N = 4, sum = 2
arr[] = {-2, 0, 1, 3}
Output: 2
Explanation: Below are triplets with
sum less than 2 (-2, 0, 1) and (-2, 0, 3).

Example 2:

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

Your Task:

This is a function problem. You don't need to take any input, as it is already accomplished by the driver code. You just need to complete the function countTriplets() that take array arr[], integer N and integer sum as parameters and returns the count of triplets.

From <https://practice.geeksforgeeks.org/problems/count-triplets-with-sum-smaller-than-x5549/1>

Expected Time Complexity: $O(N^2)$.
Expected Auxiliary Space: $O(1)$.

Constraints:
$3 \leq N \leq 10^3$
$-10^3 \leq arr[i] \leq 10^3$

View Bookmarked Problems

Company Tags

- Amazon
- SAP Labs

Topic Tags

- Arrays
- Two-pointer-algorithm

Related Interview Experiences

- Amazon interview experience set 367 campus internship
- Amazon interview experience set 383 campus internship

Given an array of distinct integers and a sum value. Find count of triplets with sum smaller than given sum value. The 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 12
(1, 3, 4), (1, 3, 5), (1, 3, 7) and
(1, 4, 5)
```

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 the triplet sum is smaller than the given sum.

From <https://practice.geeksforgeeks.org/problems/count-triplets-with-sum-smaller-than-x5549/1#>

// A Simple Java program to count triplets with sum smaller
// than a given value

```
class Test
{
    static int arr[] = new int[]{5, 1, 3, 4, 7};

    static int countTriplets(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 method to test the above function
    public static void main(String[] args)
    {
        int sum = 12;
        System.out.println(countTriplets(arr.length, sum));
    }
}
```

Output:

4

The time complexity of the 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 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 the implementation of the above idea.

From <https://practice.geeksforgeeks.org/problems/count-triplets-with-sum-smaller-than-x5549/1#>

// A Simple Java program to count triplets with sum smaller
// than a given value

```
import java.util.*;
import java.io.*;
public class countTriplet {
    public static long countTriplets(long arr[], int n, int sum) {

        Arrays.sort(arr);
        int count=0;
      System.out.println(Arrays.toString(arr));
        for (int i = 0; i < arr.length-2; i++) {

            System.out.println(" current i: "+i);
            int start =i+1;
            int end=n-1;



            while(start<end)
            {
                long  currentSum=arr[start]+arr[end]+arr[i];
                if (currentSum<sum)
```

```
                {
                    count+=end-start;


                    start++;


                }
                else
                {
                    end--;
                }
            }


        }

        return count;
    }
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String lineOne[] = br.readLine().trim().split("\\s");
        int n = Integer.parseInt(lineOne[0]);
        int k = Integer.parseInt(lineOne[1]);
        String[] lineTwo = br.readLine().trim().split("\\s");
        long arr[] = new long[n];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = Long.parseLong(lineTwo[i]);
        }
        System.out.println("Arr : " + Arrays.toString(arr));
        int sum = k;
        long ans = countTriplets(arr, n, sum);
        System.out.println(" ANs :" + ans);
    }
}
```