Count pairs with given sum

Given an array of integers, and a number 'sum', find the number of pairs of integers in the array whose sum is equal to 'sum'.

```
Examples:
Input  :  arr[] = {1, 5, 7, -1},
          sum = 6
Output :  2
Pairs with sum 6 are (1, 5) and (7, -1)

Input  :  arr[] = {1, 5, 7, -1, 5},
          sum = 6
Output :  3
Pairs with sum 6 are (1, 5), (7, -1) &
                     (1, 5)

Input  :  arr[] = {1, 1, 1, 1},
          sum = 2
Output :  6
There are 3! pairs with sum 2.

Input  :  arr[] = {10, 12, 10, 15, -1, 7, 6,
                   5, 4, 2, 1, 1, 1},
          sum = 11
Output :  9
```

Expected time complexity O(n)

A **simple solution** is be traverse each element and check if there's another number in the array which can be added to it to give sum.

C++

```
// C++ implementation of simple method to find count of
// pairs with given sum.
#include <bits/stdc++.h>
using namespace std;

// Returns number of pairs in arr[0..n-1] with sum equal
// to 'sum'
int getPairsCount(int arr[], int n, int sum)
{
    int count = 0; // Initialize result

    // Consider all possible pairs and check their sums
    for (int i=0; i<n; i++)
        for (int j=i+1; j<n; j++)
            if (arr[i]+arr[j] == sum)
                count++;

    return count;
}

// Driver function to test the above function
int main()
{
    int arr[] = {1, 5, 7, -1, 5} ;
    int n = sizeof(arr)/sizeof(arr[0]);
    int sum = 6;
    cout << "Count of pairs is "
         << getPairsCount(arr, n, sum);
    return 0;
}
```

Java

```
// Java implementation of simple method to find count of
// pairs with given sum.
public class find
{
    public static void main(String args[])
    {
        int[] arr = { 1, 5, 7, -1, 5 };
        int sum = 6;
        getPairsCount(arr, sum);
    }

    // Prints number of pairs in arr[0..n-1] with sum equal
    // to 'sum'
    public static void getPairsCount(int[] arr, int sum)
    {

        int count = 0;// Initialize result

        // Consider all possible pairs and check their sums
        for (int i = 0; i < arr.length; i++)
            for (int j = i + 1; j < arr.length; j++)
                if ((arr[i] + arr[j]) == sum)
                    count++;

        System.out.printf("Count of pairs is %d",count);
    }
}
// This program is contributed by Jyotsna
```

Output :

```
Count of pairs is 3
```

Time Complexity : $O(n^2)$
Auxiliary Space : O(1)

A better solution is possible in O(n) time.

Below is the Algorithm.

1. Create a map to store frequency of each number in the array. (Single traversal is required)
2. In the next traversal, for every element check if it can be combined with any other element (other than itself!) to give the desired sum. Increment the counter accordingly.
3. After completion of second traversal, we'd have twice the required value stored in counter because every pair is counted two times. Hence divide count by 2 and return.

Below is the C++ program for the same:

```
// C++ implementation of simple method to find count of
// pairs with given sum.
#include <bits/stdc++.h>
using namespace std;

// Returns number of pairs in arr[0..n-1] with sum equal
// to 'sum'
int getPairsCount(int arr[], int n, int sum)
{
    unordered_map<int, int> m;

    // Store counts of all elements in map m
    for (int i=0; i<n; i++)
        m[arr[i]]++;

    int twice_count = 0;

    // iterate through each element and increment the
    // count (Notice that every pair is counted twice)
    for (int i=0; i<n; i++)
    {
        twice_count += m[sum-arr[i]];

        // if (arr[i], arr[i]) pair satisfies the condition,
        // then we need to ensure that the count is
        // decreased by one such that the (arr[i], arr[i])
        // pair is not considered
        if (sum-arr[i] == arr[i])
            twice_count--;
    }

    // return the half of twice_count
    return twice_count/2;
}

// Driver function to test the above function
int main()
{
    int arr[] = {1, 5, 7, -1, 5} ;
    int n = sizeof(arr)/sizeof(arr[0]);
    int sum = 6;
    cout << "Count of pairs is "
         << getPairsCount(arr, n, sum);
    return 0;
}
```

Output :

```
Count of pairs is 3
```