2 Sum - Count pairs with given sum - GeeksforGeeks

Source: https://www.geeksforgeeks.org/count-pairs-with-given-sum/

Courses Tutorials Practice Jobs DSA Tutorial Interview Questions Quizzes Must Do Advanced DSA System Design Aptitude Puzzles Interview Corner DSA Python Technical Scripter 2026 Explore DSA Fundamentals Logic Building Problems Analysis of Algorithms Data Structures Array Data Structure String in Data Structure Hashing in Data Structure Linked List Data Structure Stack Data Structure Queue Data Structure Tree Data Structure Graph Data Structure Trie Data Structure Algorithms Searching Algorithms Sorting Algorithms Introduction to Recursion Greedy Algorithms Tutorial Graph Algorithms Dynamic Programming or DP Bitwise Algorithms Advanced Segment Tree Binary Indexed Tree or Fenwick Tree Square Root (Sqrt) Decomposition Algorithm Binary Lifting Geometry Interview Preparation Interview Corner GfG160 Practice Problem GeeksforGeeks Practice - Leading Online Coding Platform Problem of The Day - Develop the Habit of Coding DSA Course 90% Refund 2 Sum - Count pairs with given sum Last Updated : 6 Aug, 2025 Given an array arr[] of n integers and a target value , find the number of pairs of integers in the array whose sum is equal to target . Examples: Input: arr[] = [1, 5, 7, -1, 5], target = 6 Output: 3 Explanation: Pairs with sum 6 are (1, 5), (7, -1) & (1, 5). Input: arr[] = [1, 1, 1, 1], target = 2 Output: 6 Explanation: Pairs with sum 2 are (1, 1), (1, 1), (1, 1), (1, 1), (1, 1) and (1, 1). Input: arr[] = [10, 12, 10, 15, -1], target = 125 Output: 0 Explanation: There is no pair with sum = target Try it on GfG Practice Table of Content [Naive Approach] By Generating all Possible Pairs - O(n^2) time and O(1) space [Better Approach] Using Two Pointers Technique - O(nlogn) Time and O(1) Space [Expected Approach] Using Hash Map or Dictionary - O(n) Time and O(n) Space [Naive Approach] By Generating all Possible Pairs - O(n^2) time and O(1) space The very basic approach is to generate all the possible pairs and check if any pair exists whose sum is equals to given target value, then increment the count variable. C++ #include <iostream> #include <vector> using namespace std ; int countPairs (vector < int > & arr , int target) { int n = arr . size (); int cnt = 0 ; // Iterate through each element in the array for (int i = 0 ; i < n ; i ++) { // For each element arr[i], check every // other element arr[j] that comes after it for (int j = i + 1 ; j < n ; j ++) { // Check if the sum of the current pair // equals the target if (arr [i] + arr [j] == target) { cnt ++ ; } } } return cnt ; } int main () { vector < int > arr = { 1 , 5 , 7 , -1 , 5 }; int target = 6 ; cout << countPairs (arr , target) << endl ; return 0 ; } C #include <stdio.h> int countPairs (int arr [], int n , int target) { int cnt = 0 ; // Iterate through each element in the array for (int i = 0 ; i < n ; i ++) { // For each element arr[i], check every // other element arr[j] that comes after it for (int j = i + 1 ; j < n ; j ++) { // Check if the sum of the current pair // equals the target if (arr [i] + arr [j] == target) { cnt ++ ; } } } return cnt ; } int main () { int arr [] = { 1 , 5 , 7 , -1 , 5 }; int target = 6 ; int n = sizeof (arr) / sizeof (arr [0]); printf ("%d \n " , countPairs (arr , n , target)); return 0 ; } Java import java.util.Arrays ; class GfG { // Function to count all pairs whose sum is equal // to the given target value static int countPairs (int [] arr , int target) { int n = arr . length ; int cnt = 0 ; // Iterate through each element in the array for (int i = 0 ; i < n ; i ++) { // For each element arr[i], check every // other element arr[j] that comes after it for (int j = i + 1 ; j < n ; j ++) { // Check if the sum of the current pair // equals the target if (arr [i] + arr [j] == target) { cnt ++ ; } } } return cnt ; } public static void main (String [] args) { int [] arr = { 1 , 5 , 7 , - 1 , 5 }; int target = 6 ; System . out . println (countPairs (arr , target)); } } Python def countPairs (arr , target): n = len (arr) cnt = 0 # Iterate through each element in the array for i in range (n): # For each element arr[i], check every # other element arr[j] that comes after it for j in range (i + 1 , n): # Check if the sum of the current pair # equals the target if arr [i] + arr [j] == target : cnt += 1 return cnt if __name__ == "__main__" : arr = [1 , 5 , 7 , - 1 , 5] target = 6 print (countPairs (arr , target)) C# using System ; class GfG { // Function to count all pairs whose sum is // equal to the given target value static int countPairs (int [] arr , int target) { int n = arr . Length ; int cnt = 0 ; // Iterate through each element in the array for (int i = 0 ; i < n ; i ++) { // For each element arr[i], check every // other element arr[j] that comes after it for (int j = i + 1 ; j < n ; j ++) { // Check if the sum of the current pair // equals the target if (arr [i] + arr [j] == target) { cnt ++ ; } } } return cnt ; } static void Main () { int [] arr = { 1 , 5 , 7 , - 1 , 5 }; int target = 6 ; Console . WriteLine (countPairs (arr , target)); } } JavaScript function countPairs (arr , target) { const n = arr . length ; let cnt = 0 ; // Iterate through each element in the array for (let i = 0 ; i < n ; i ++) { // For each element arr[i], check every // other element arr[j] that comes after it for (let j = i + 1 ; j < n ; j ++) { // Check if the sum of the current pair // equals the target if (arr [i] + arr [j] === target) cnt ++ ; } } return cnt ; } // Driver Code const arr = [1 , 5 , 7 , - 1 , 5]; const target = 6 ; console . log (countPairs (arr , target)); Output 3 [Better Approach] Using Two Pointers Technique - O(nlogn) Time and O(1) Space The idea is to sort the input array and use two-pointer technique . Maintain two pointers, say left and right and initialize them to the first and last element of the array respectively. According to the sum of left and right pointers, we can have three cases: arr[left] + arr[right] < target: Increase the pair sum by moving the left pointer towards right. arr[left] + arr[right] > target: Decrease the pair sum by moving the right pointer towards left. arr[left] + arr[right] = target: We have found a pair whose sum is equal to target. We can find the product of the count of both the elements and add them to the result. C++ #include <iostream> #include <vector> #include <algorithm> using namespace std ; int countPairs (vector < int > & arr , int target) { int res = 0 ; int n = arr . size (); int left = 0 , right = n - 1 ; // Sort the array before applying // two-pointer technique sort (arr . begin (), arr . end ()); while (left < right) { // If sum is greater if (arr [left] + arr [right] < target) left ++ ; // If sum is lesser else if (arr [left] + arr [right] > target) right -- ; // If sum is equal else { int cnt1 = 0 , cnt2 = 0 ; int ele1 = arr [left], ele2 = arr [right]; // Count frequency of first element of the pair while (left <= right and arr [left] == ele1) { left ++ ; cnt1 ++ ; } // Count frequency of second element of the pair while (left <= right and arr [right] == ele2) { right -- ; cnt2 ++ ; } // If both the elements are same, then count of // pairs = the number of ways to choose 2 // elements among cnt1 elements if (ele1 == ele2) res += (cnt1 * (cnt1 - 1)) / 2 ; // If the elements are different, then count of // pairs = product of the count of both elements else res += (cnt1 * cnt2); } } return res ; } int main () { vector < int > arr = { 1 , 5 , 7 , -1 , 5 }; int target = 6 ; cout << countPairs (arr , target); return 0 ; } Java import java.util.Arrays ; class GfG { static int countPairs (int [] arr , int target) { int res = 0 ; int n = arr . length ; int left = 0 , right = n - 1 ; // Sort the array before applying // two-pointer technique Arrays . sort (arr); while (left < right) { // If sum is less than target if (arr [left] + arr [right] < target) { left ++ ; } // If sum is more than target else if (arr [left] + arr [right] > target) { right -- ; } // If sum is equal to target else { int cnt1 = 0 , cnt2 = 0 ; int ele1 = arr [left] , ele2 = arr [right] ; // Count frequency of first element while (left <= right && arr [left] == ele1) { cnt1 ++ ; left ++ ; } // Count frequency of second element while (left <= right && arr [right] == ele2) { cnt2 ++ ; right -- ; } // If both elements are same if (ele1 == ele2) { res += (cnt1 * (cnt1 - 1)) / 2 ; } else { res += cnt1 * cnt2 ; } } } return res ; } public static void main (String [] args) { int [] arr = { 1 , 5 , 7 , - 1 , 5 }; int target = 6 ; System . out . println (countPairs (arr , target)); } } Python def countPairs (arr , target): res = 0 n = len (arr) left = 0 right = n - 1 # Sort the array before applying # the two-pointer technique arr . sort () while left < right : # If sum is less than target if arr [left] + arr [right] < target : left += 1 # If sum is more than target elif arr [left] + arr [right] > target : right -= 1 # If sum is equal to target else : cnt1 = 0 cnt2 = 0 ele1 = arr [left] ele2 = arr [right] # Count frequency of arr[left] while left <= right and arr [left] == ele1 : left += 1 cnt1 += 1 # Count frequency of arr[right] while left <= right and arr [right] == ele2 : right -= 1 cnt2 += 1 # If both elements are same if ele1 == ele2 : res += (cnt1 * (cnt1 - 1)) // 2 else : res += cnt1 * cnt2 return res if __name__ == "__main__" : arr = [1 , 5 , 7 , - 1 , 5] target = 6 print (countPairs (arr , target)) C# using System ; class GfG { static int countPairs (int [] arr , int target) { int res = 0 ; int n = arr . Length ; int left = 0 , right = n - 1 ; // Sort the array before applying // the two-pointer approach Array . Sort (arr); while (left < right) { // If sum is less than target if (arr [left] + arr [right] < target) left ++ ; // If sum is more than target else if (arr [left] + arr [right] > target) right -- ; // If sum is equal to target else { int cnt1 = 0 , cnt2 = 0 ; int ele1 = arr [left], ele2 = arr [right]; // Count frequency of first element while (left <= right && arr [left] == ele1) { left ++ ; cnt1 ++ ; } // Count frequency of second element while (left <= right && arr [right] == ele2) { right -- ; cnt2 ++ ; } // If both elements are same if (ele1 == ele2) res += (cnt1 * (cnt1 - 1)) / 2 ; else res += (cnt1 * cnt2); } } return res ; } static void Main (string [] args) { int [] arr = { 1 , 5 , 7 , - 1 , 5 }; int target = 6 ; Console . WriteLine (countPairs (arr , target)); } } JavaScript function countPairs (arr , target) { let res = 0 ; const n = arr . length ; let left = 0 , right = n - 1 ; // Sort the array before using // two-pointer approach arr . sort ((a , b) => a - b); while (left < right) { // If sum is less than target if (arr [left] + arr [right] < target) { left ++ ; } // If sum is more than target else if (arr [left] + arr [right] > target) { right -- ; } // If sum is equal to target else { let cnt1 = 0 , cnt2 = 0 ; const ele1 = arr [left], ele2 = arr [right]; // Count frequency of arr[left] while (left <= right && arr [left] === ele1) { left ++ ; cnt1 ++ ; } // Count frequency of arr[right] while (left <= right && arr [right] === ele2) { right -- ; cnt2 ++ ; } // If both elements are the same if (ele1 === ele2) res += (cnt1 * (cnt1 - 1)) / 2 ; else res += cnt1 * cnt2 ; } } return res ; } // Driver Code const arr = [1 , 5 , 7 , - 1 , 5]; const target = 6 ; console . log (countPairs (arr , target)); Output 3 [Expected Approach] Using Hash Map or Dictionary - O(n) Time and O(n) Space HashMap or Dictionary provides a more efficient solution to the 2Sum problem. Instead of checking every pair of numbers, we keep each number in a map as we go through the array. For each number, we calculate its complement (i.e., target - current number) and check if it’s in the map. If it is, increment the count variable by the occurrences of complement in map. C++ #include <iostream> #include <vector> #include <unordered_map> using namespace std ; // Returns number of pairs in arr[0...n-1] with sum // equal to 'target' int countPairs (vector < int >& arr , int target) { unordered_map < int , int > freq ; int cnt = 0 ; for (int i = 0 ; i < arr . size (); i ++) { // Check if the complement (target - arr[i]) // exists in the map. If yes, increment count if (freq . find (target - arr [i]) != freq . end ()) { cnt += freq [target - arr [i]]; } // Increment the frequency of arr[i] freq [arr [i]] ++ ; } return cnt ; } int main () { vector < int > arr = { 1 , 5 , 7 , -1 , 5 }; int target = 6 ; cout << countPairs (arr , target); return 0 ; } Java import java.util.Map ; import java.util.HashMap ; class GfG { // Returns number of pairs in arr[0...n-1] with // sum equal to 'target' static int countPairs (int [] arr , int target) { Map < Integer , Integer > freq = new HashMap <> (); int cnt = 0 ; for (int i = 0 ; i < arr . length ; i ++) { // Check if the complement (target - arr[i]) // exists in the map. If yes, increment count if (freq . containsKey (target - arr [i])) { cnt += freq . get (target - arr [i]); } // Increment the frequency of arr[i] freq . put (arr [i] , freq . getOrDefault (arr [i] , 0) + 1); } return cnt ; } public static void main (String [] args) { int [] arr = { 1 , 5 , 7 , - 1 , 5 }; int target = 6 ; System . out . println (countPairs (arr , target)); } } Python def countPairs (arr , target): freq = {} cnt = 0 for i in range (len (arr)): # Check if the complement (target - arr[i]) # exists in the map. If yes, increment count if (target - arr [i]) in freq : cnt += freq [target - arr [i]] # Increment the frequency of arr[i] freq [arr [i]] = freq . get (arr [i], 0) + 1 return cnt if __name__ == "__main__" : arr = [1 , 5 , 7 , - 1 , 5] target = 6 print (countPairs (arr , target)) C# using System ; using System.Collections.Generic ; class GfG { static int countPairs (int [] arr , int target) { Dictionary < int , int > freq = new Dictionary < int , int > (); int cnt = 0 ; for (int i = 0 ; i < arr . Length ; i ++) { // Check if the complement (target - arr[i]) // exists in the map. If yes, increment count if (freq . ContainsKey (target - arr [i])) { cnt += freq [target - arr [i]]; } // Increment the frequency of arr[i] if (freq . ContainsKey (arr [i])) freq [arr [i]] ++ ; else freq [arr [i]] = 1 ; } return cnt ; } public static void Main () { int [] arr = { 1 , 5 , 7 , - 1 , 5 }; int target = 6 ; Console . WriteLine (countPairs (arr , target)); } } JavaScript function countPairs (arr , target) { const freq = new Map (); let cnt = 0 ; for (let i = 0 ; i < arr . length ; i ++) { // Check if the complement (target - arr[i]) // exists in the map. If yes, increment count if (freq . has (target - arr [i])) { cnt += freq . get (target - arr [i]); } // Increment the frequency of arr[i] freq . set (arr [i], (freq . get (arr [i]) || 0) + 1); } return cnt ; } const arr = [1 , 5 , 7 , - 1 , 5]; const target = 6 ; console . log (countPairs (arr , target)); Output 3 Related Problems: 2 Sum – Count Pairs with given Sum in Sorted Array 2 Sum - Complete Tutorial Comment Article Tags: Article Tags: Hash DSA Arrays Amazon Accolite Hike FactSet 2Sum + 4 More