3 Sum - Triplet Sum in Array - GeeksforGeeks

Source: https://www.geeksforgeeks.org/find-a-triplet-that-sum-to-a-given-value/

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 3 Sum - Triplet Sum in Array Last Updated : 13 Aug, 2025 Given an array arr[] and an integer sum , check if there is a triplet in the array which sums up to the given target sum. Examples: Input: arr[] = [1, 4, 45, 6, 10, 8], target = 13 Output: true Explanation: The triplet [1, 4, 8] sums up to 13 Input: arr[] = [1, 2, 4, 3, 6, 7], target = 10 Output: true Explanation: The triplets [1, 3, 6] and [1, 2, 7] both sum to 10. Input: arr[] = [40, 20, 10, 3, 6, 7], sum = 24 Output: false Explanation: No triplet in the array sums to 24. Try it on GfG Practice Table of Content [Naive Approach] Generating All Triplets - O(n^3) Time and O(1) Space [Better Approach] - Hash Set - O(n^2) Time and O(n) Space [Expected Approach] - Sorting and Two Pointer - O(n^2) Time and O(1) Space [Naive Approach] Generating All Triplets - O(n^3) Time and O(1) Space A simple method is to generate all possible triplets and compare the sum of every triplet with the given target. If the sum is equal to target, return true. Otherwise, return false. C++ #include <iostream> #include <vector> using namespace std ; bool hasTripletSum (vector < int >& arr , int target) { int n = arr . size (); // Fix the first element as arr[i] for (int i = 0 ; i < n - 2 ; i ++) { // Fix the second element as arr[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] == target) return true ; } } } return false ; } int main () { vector < int > arr = { 1 , 4 , 45 , 6 , 10 , 8 }; int target = 13 ; if (hasTripletSum (arr , target)) cout << "true" ; else cout << "false" ; return 0 ; } C #include <stdio.h> #include <stdbool.h> bool hasTripletSum (int arr [], int n , int target) { // Fix the first element as arr[i] for (int i = 0 ; i < n - 2 ; i ++) { // Fix the second element as arr[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] == target) return true ; } } } return false ; } int main () { int arr [] = { 1 , 4 , 45 , 6 , 10 , 8 }; int target = 13 ; int n = sizeof (arr) / sizeof (arr [0]); if (hasTripletSum (arr , n , target)) printf ("true"); else printf ("false"); return 0 ; } Java class GfG { static boolean hasTripletSum (int [] arr , int target) { int n = arr . length ; // Fix the first element as arr[i] for (int i = 0 ; i < n - 2 ; i ++) { // Fix the second element as arr[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] == target) return true ; // If a triplet is found } } } return false ; } public static void main (String [] args) { int [] arr = { 1 , 4 , 45 , 6 , 10 , 8 }; int target = 13 ; if (hasTripletSum (arr , target)) System . out . println ("true"); else System . out . println ("false"); } } Python def hasTripletSum (arr , target): n = len (arr) # Fix the first element as arr[i] for i in range (n - 2): # Fix the second element as arr[j] for j in range (i + 1 , n - 1): # Now look for the third number for k in range (j + 1 , n): if arr [i] + arr [j] + arr [k] == target : return True return False if __name__ == "__main__" : arr = [1 , 4 , 45 , 6 , 10 , 8] target = 13 if hasTripletSum (arr , target): print ("true") else : print ("false") C# using System ; class GfG { static bool hasTripletSum (int [] arr , int target) { int n = arr . Length ; // Fix the first element as arr[i] for (int i = 0 ; i < n - 2 ; i ++) { // Fix the second element as arr[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] == target) return true ; } } } return false ; } static void Main () { int [] arr = { 1 , 4 , 45 , 6 , 10 , 8 }; int target = 13 ; if (hasTripletSum (arr , target)) Console . WriteLine ("true"); else Console . WriteLine ("false"); } } JavaScript function hasTripletSum (arr , target) { let n = arr . length ; // Fix the first element as arr[i] for (let i = 0 ; i < n - 2 ; i ++) { // Fix the second element as arr[j] for (let j = i + 1 ; j < n - 1 ; j ++) { // Now look for the third number for (let k = j + 1 ; k < n ; k ++) { if (arr [i] + arr [j] + arr [k] === target) return true ; } } } return false ; } // Driver code let arr = [1 , 4 , 45 , 6 , 10 , 8]; let target = 13 ; if (hasTripletSum (arr , target)) console . log ("true"); else console . log ("false"); Output true [Better Approach] - Hash Set - O(n^2) Time and O(n) Space The idea is to traverse every

element arr[i] in a loop. For every arr[i], use the hashing based solution of 2 Sum Problem to check if there is a pair with sum equal to given sum - arr[i]. Step by Step Approach: Iterate through the array, fixing the first element (arr [i]) for the triplet. For each arr [i] , use a Hash Set to store potential second elements and run another loop inside it for j from i+1 to n-1. Inside a nested loop, check if given sum - arr[i] - arr[j] is present in the hash set. If yes, then print the triplet. If no triplet is found in the entire array, the function returns false . C++ #include <iostream> #include <vector> #include <unordered_set> using namespace std ; bool hasTripletSum (vector < int >& arr , int target) { int n = arr . size (); // Fix the first element as arr[i] for (int i = 0 ; i < n - 2 ; i ++) { // Hash set to store potential second elements unordered_set < int > st ; // Fix the third element as arr[j] for (int j = i + 1 ; j < n ; j ++) { int second = target - arr [i] - arr [j]; // Search for second element in hash set if (st . find (second) != st . end ()) { return true ; } // Add arr[j] as a potential second element st . insert (arr [j]); } } return false ; } int main () { vector < int > arr = { 1 , 4 , 45 , 6 , 10 , 8 }; int target = 13 ; if (hasTripletSum (arr , target)) cout << "true" ; else cout << "false" ; return 0 ; } Java import java.util.HashSet ; import java.util.Set ; class GfG { static boolean hasTripletSum (int [] arr , int target) { int n = arr . length ; // Fix the first element as arr[i] for (int i = 0 ; i < n - 2 ; i ++) { // Hash set to store potential second elements Set < Integer > st = new HashSet <> (); // Fix the third element as arr[j] for (int j = i + 1 ; j < n ; j ++) { int second = target - arr [i] - arr [j] ; // Search for second element in hash set if (st . contains (second)) { return true ; } // Add arr[j] as a potential second element st . add (arr [j]); } } return false ; } public static void main (String [] args) { int [] arr = { 1 , 4 , 45 , 6 , 10 , 8 }; int target = 13 ; if (hasTripletSum (arr , target)) System . out . println ("true"); else System . out . println ("false"); } } Python def hasTripletSum (arr , target): n = len (arr) # Fix the first element as arr[i] for i in range (n - 2): # Hash set to store potential second elements st = set () # Fix the third element as arr[j] for j in range (i + 1 , n): second = target - arr [i] - arr [j] # Search for second element in hash set if second in st : return True # Add arr[j] as a potential second element st . add (arr [j]) return False if __name__ == "__main__" : arr = [1 , 4 , 45 , 6 , 10 , 8] target = 13 if hasTripletSum (arr , target): print ("true") else : print ("false") C# using System ; using System.Collections.Generic ; class GfG { static bool hasTripletSum (int [] arr , int target) { int n = arr . Length ; // Fix the first element as arr[i] for (int i = 0 ; i < n - 2 ; i ++) { // Hash set to store potential second elements HashSet < int > st = new HashSet < int > (); // Fix the third element as arr[j] for (int j = i + 1 ; j < n ; j ++) { int second = target - arr [i] - arr [j]; // Search for second element in hash set if (st . Contains (second)) return true ; // Add arr[j] as a potential second element st . Add (arr [j]); } } return false ; } static void Main (string [] args) { int [] arr = { 1 , 4 , 45 , 6 , 10 , 8 }; int target = 13 ; if (hasTripletSum (arr , target)) Console . WriteLine ("true"); else Console . WriteLine ("false"); } } JavaScript function hasTripletSum (arr , target) { let n = arr . length ; // Fix the first element as arr[i] for (let i = 0 ; i < n - 2 ; i ++) { // Hash set to store potential second elements let st = new Set (); // Fix the third element as arr[j] for (let j = i + 1 ; j < n ; j ++) { let second = target - arr [i] - arr [j]; // Search for second element in hash set if (st . has (second)) { return true ; } // Add arr[j] as a potential second element st . add (arr [j]); } } return false ; } // Driver code let arr = [1 , 4 , 45 , 6 , 10 , 8]; let target = 13 ; if (hasTripletSum (arr , target)) console . log ("true"); else console . log ("false"); Output true [Expected Approach] - Sorting and Two Pointer - O(n^2) Time and O(1) Space We first sort the array. After sorting, we traverse every element arr[i] in a loop. For every arr[i], use the Two Pointer Technique based solution of 2 Sum Problem to check if there is a pair with sum equal to given sum - arr[i]. C++ #include <iostream> #include <vector> #include <algorithm> using namespace std ; bool hasTripletSum (vector < int >& arr , int target) { int n = arr . size (); sort (arr . begin (), arr . end ()); // Fix the first element as arr[i] for (int i = 0 ; i < n - 2 ; i ++) { // Initialize left and right pointers with // start and end of remaining subarray int l = i + 1 , r = n - 1 ; int requiredSum = target - arr [i]; while (l < r) { if (arr [l] + arr [r] == requiredSum) return true ; if (arr [l] + arr [r] < requiredSum) l ++ ; else if (arr [l] + arr [r] > requiredSum) r -- ; } } return false ; } int main () { vector < int > arr = { 1 , 4 , 45 , 6 , 10 , 8 }; int target = 13 ; if (hasTripletSum (arr , target)) cout << "true" ; else cout << "false" ; return 0 ; } C #include <stdio.h> int compare (const void * a , const void * b) { return (* (int *) a - * (int *) b); } int hasTripletSum (int arr [], int n , int target) { qsort (arr , n , sizeof (int), compare); // Fix the first element as arr[i] for (int i = 0 ; i < n - 2 ; i ++) { // Initialize left and right pointers with // start and end of remaining subarray int l = i + 1 , r = n - 1 ; int requiredSum = target - arr [i]; while (l < r) { if (arr [l] + arr [r] == requiredSum) return 1 ; if (arr [l] + arr [r] < requiredSum) l ++ ; else if (arr [l] + arr [r] > requiredSum) r -- ; } } return 0 ; } int main () { int arr [] = { 1 , 4 , 45 , 6 , 10 , 8 }; int target = 13 ; int n = sizeof (arr) / sizeof (arr [0]); if (hasTripletSum (arr , n , target)) printf ("true"); else printf ("false"); return 0 ; } Java import java.util.Arrays ; class GfG { static boolean hasTripletSum (int [] arr , int target) { int n = arr . length ; Arrays . sort (arr); // Fix the first element as arr[i] for (int i = 0 ; i < n - 2 ; i ++) { //

Initialize left and right pointers with // start and end of remaining subarray int l = i + 1 , r = n - 1 ; int requiredSum = target - arr [i] ; while (l < r) { if (arr [l] + arr [r] == requiredSum) return true ; if (arr [l] + arr [r] < requiredSum) l ++ ; else if (arr [l] + arr [r] > requiredSum) r -- ; } } return false ; } public static void main (String [] args) { int [] arr = { 1 , 4 , 45 , 6 , 10 , 8 }; int target = 13 ; if (hasTripletSum (arr , target)) System . out . println ("true"); else System . out . println ("false"); } } Python def hasTripletSum (arr , target): n = len (arr) arr . sort () # Fix the first element as arr[i] for i in range (n - 2): # Initialize left and right pointers with # start and end of remaining subarray l = i + 1 r = n - 1 requiredSum = target - arr [i] while l < r : if arr [l] + arr [r] == requiredSum : return True if arr [l] + arr [r] < requiredSum : l += 1 else : r -= 1 return False if __name__ == "__main__" : arr = [1 , 4 , 45 , 6 , 10 , 8] target = 13 if hasTripletSum (arr , target): print ("true") else : print ("false") C# using System ; class GfG { static bool hasTripletSum (int [] arr , int target) { int n = arr . Length ; Array . Sort (arr); // Fix the first element as arr[i] for (int i = 0 ; i < n - 2 ; i ++) { // Initialize left and right pointers with // start and end of remaining subarray int l = i + 1 , r = n - 1 ; int requiredSum = target - arr [i]; while (l < r) { if (arr [l] + arr [r] == requiredSum) return true ; if (arr [l] + arr [r] < requiredSum) l ++ ; else if (arr [l] + arr [r] > requiredSum) r -- ; } } return false ; } static void Main (string [] args) { int [] arr = { 1 , 4 , 45 , 6 , 10 , 8 }; int target = 13 ; if (hasTripletSum (arr , target)) Console . WriteLine ("true"); else Console . WriteLine ("false"); } } JavaScript function hasTripletSum (arr , target) { let n = arr . length ; arr . sort ((a , b) => a - b); // Fix the first element as arr[i] for (let i = 0 ; i < n - 2 ; i ++) { // Initialize left and right pointers with // start and end of remaining subarray let l = i + 1 , r = n - 1 ; let requiredSum = target - arr [i]; while (l < r) { if (arr [l] + arr [r] == requiredSum) return true ; if (arr [l] + arr [r] < requiredSum) l ++ ; else if (arr [l] + arr [r] > requiredSum) r -- ; } } return false ; } // Driver code let arr = [1 , 4 , 45 , 6 , 10 , 8]; let target = 13 ; if (hasTripletSum (arr , target)) console . log ("true"); else console . log ("false"); Output true Related Problems 3 Sum – Find All Triplets with Zero Sum 3 Sum – Find all Triplets with Given Sum 3 Sum – Triplet Sum Closest to Target 3 Sum – Pythagorean Triplet in an array 3 Sum – All Distinct Triplets with given Sum Pythagorean Triplet with given sum Count triplets with sum smaller than a given value Please refer 3Sum - Complete Tutorial for all list of problems on triplets in an array. Comment Article Tags: Article Tags: Sorting DSA Arrays Amazon Morgan Stanley Samsung Accolite CarWale two-pointer-algorithm 3Sum + 6 More