Subarray with Given Sum - GeeksforGeeks

Source: https://www.geeksforgeeks.org/find-subarray-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 Subarray with Given Sum Last Updated : 27 Jan, 2026 Given a 1-based indexing array arr[] of non-negative integers and an integer sum . Find the left and right indexes(1-based indexing) of that subarray that is equal to the given sum. In case of multiple subarrays, find the subarray indexes which come first on moving from left to right. If no such subarray exists return an array consisting of element -1 . Examples: Input : arr[] = [15, 2, 4, 8, 9, 5, 10, 23], target = 23 Output : [2, 5] Explanation: Sum of subarray arr[2...5] is 2 + 4 + 8 + 9 = 23. Input : arr[] = [1, 10, 4, 0, 3, 5], target = 7 Output : [3, 5] Explanation: Sum of subarray arr[3...5] is 4 + 0 + 3 = 7. Input : arr[] = [1, 4], target = 0 Output : [-1] Explanation: There is no subarray with 0 sum. Try it on GfG Practice Table of Content [Naive Approach] Using Nested loop - O(n2) Time and O(1) Space [Expected Approach] Sliding Window - O(n) Time and O(1) Space [Alternate Approach] Hashing + Prefix Sum - O(n) Time and O(n) Space [Naive Approach] Using Nested loop - O(n^2) Time and O(1) S pace The very basic idea is to use a nested loop where the outer loop picks a starting element, and the inner loop calculates the cumulative sum of elements starting from this element. For each starting element, the inner loop iterates through subsequent elements and adding each element to the cumulative sum until the given sum is found or the end of the array is reached. If at any point the cumulative sum equals the given sum , then return starting and ending indices (1-based). If no such sub-array is found after all iterations, then return -1. C++ #include <iostream> #include <vector> using namespace std ; vector < int > subarraySum (vector < int > arr , int target) { vector < int > res ; int n = arr . size (); // Pick a starting point for a subarray for (int s = 0 ; s < n ; s ++) { int curr = 0 ; // Consider all ending points // for the picked starting point for (int e = s ; e < n ; e ++) { curr += arr [e]; if (curr == target) { res . push_back (s + 1); res . push_back (e + 1); return res ; } } } // If no subarray is found return { -1 }; } int main () { vector < int > arr = { 15 , 2 , 4 , 8 , 9 , 5 , 10 , 23 }; int target = 23 ; vector < int > res = subarraySum (arr , target); for (int ele : res) cout << ele << " " ; return 0 ; } Java import java.util.ArrayList ; import java.util.List ; class GfG { static ArrayList < Integer > subarraySum (int [] arr , int target) { ArrayList < Integer > res = new ArrayList <> (); int n = arr . length ; // Pick a starting point for a subarray for (int s = 0 ; s < n ; s ++) { int curr = 0 ; // Consider all ending points // for the picked starting point for (int e = s ; e < n ; e ++) { curr += arr [e] ; if (curr == target) { res . add (s + 1); res . add (e + 1); return res ; } } } // If no subarray is found res . add (- 1); return res ; } public static void main (String [] args) { int [] arr = { 15 , 2 , 4 , 8 , 9 , 5 , 10 , 23 }; int target = 23 ; ArrayList < Integer > res = subarraySum (arr , target); for (int ele : res) System . out . print (ele + " "); } } Python def subarraySum (arr , target): res = [] n = len (arr) # Pick a starting point for a subarray for s in range (n): curr = 0 # Consider all ending points # for the picked starting point for e in range (s , n): curr += arr [e] if curr == target : res . append (s + 1) res . append (e + 1) return res # If no subarray is found return [- 1] if __name__ == "__main__" : arr = [15 , 2 , 4 , 8 , 9 , 5 , 10 , 23] target = 23 res = subarraySum (arr , target) for ele in res : print (ele , end = " ") C# using System ; using System.Collections.Generic ; class GfG { static List < int > subarraySum (int [] arr , int target) { List < int > res = new List < int > (); int n = arr . Length ; // Pick a starting point for a subarray for (int s = 0 ; s < n ; s ++) { int curr = 0 ; // Consider all ending points // for the picked starting point for (int e = s ; e < n ; e ++) { curr += arr [e]; if (curr == target) { res . Add (s + 1); res . Add (e + 1); return res ; } } } // If no subarray is found res . Add (- 1); return res ; } static void Main () { int [] arr = { 15 , 2 , 4 , 8 , 9 , 5 , 10 , 23 }; int target = 23 ; List < int > res = subarraySum (arr , target); foreach (var ele in res) Console . Write (ele + " "); } } JavaScript function subarraySum (arr , target) { let res = []; let n = arr . length ; //

Pick a starting point for a subarray for (let s = 0 ; s < n ; s ++) { let curr = 0 ; // Consider all ending points // for the picked starting point for (let e = s ; e < n ; e ++) { curr += arr [e]; if (curr === target) { res . push (s + 1); res . push (e + 1); return res ; } } } // If no subarray is found return [- 1]; } // Driver Code let arr = [15 , 2 , 4 , 8 , 9 , 5 , 10 , 23]; let target = 23 ; let res = subarraySum (arr , target); console . log (res . join (' ')); Output 2 5 [Expected Approach] Sliding Window - O(n) Time and O(1) S pace The idea is simple, as we know that all the elements in subarray are positive so, If a subarray has sum greater than the given sum then there is no possibility that adding elements to the current subarray will be equal to the given sum. So the Idea is to use a similar approach to a sliding window . Start with an empty window add elements to the window while the current sum is less than sum If the sum is greater than sum , remove elements from the start of the current window. If current sum becomes same as sum, return the result C++ #include <iostream> #include <vector> using namespace std ; vector < int > subarraySum (vector < int >& arr , int target) { // Initialize window int s = 0 , e = 0 ; vector < int > res ; int curr = 0 ; for (int i = 0 ; i < arr . size (); i ++) { curr += arr [i]; // If current sum becomes more or equal, // set end and try adjusting start if (curr >= target) { e = i ; // While current sum is greater, // remove starting elements of current window while (curr > target && s < e) { curr -= arr [s]; ++ s ; } // If we found a subraay if (curr == target) { res . push_back (s + 1); res . push_back (e + 1); return res ; } } } // If no subarray is found return { -1 }; } int main () { vector < int > arr = { 15 , 2 , 4 , 8 , 9 , 5 , 10 , 23 }; int target = 23 ; vector < int > res = subarraySum (arr , target); for (int ele : res) cout << ele << " " ; return 0 ; } Java import java.util.ArrayList ; import java.util.List ; class GfG { static ArrayList < Integer > subarraySum (int [] arr , int target) { // Initialize window int s = 0 , e = 0 ; ArrayList < Integer > res = new ArrayList <> (); int curr = 0 ; for (int i = 0 ; i < arr . length ; i ++) { curr += arr [i] ; // If current sum becomes more or equal, // set end and try adjusting start if (curr >= target) { e = i ; // While current sum is greater, // remove starting elements of current window while (curr > target && s < e) { curr -= arr [s] ; ++ s ; } // If we found a subarray if (curr == target) { res . add (s + 1); res . add (e + 1); return res ; } } } // If no subarray is found res . add (- 1); return res ; } public static void main (String [] args) { int [] arr = { 15 , 2 , 4 , 8 , 9 , 5 , 10 , 23 }; int target = 23 ; ArrayList < Integer > res = subarraySum (arr , target); for (int ele : res) System . out . print (ele + " "); } } Python def subarraySum (arr , target): # Initialize window s , e = 0 , 0 res = [] curr = 0 for i in range (len (arr)): curr += arr [i] # If current sum becomes more or equal, # set end and try adjusting start if curr >= target : e = i # While current sum is greater, # remove starting elements of current window while curr > target and s < e : curr -= arr [s] s += 1 # If we found a subarray if curr == target : res . append (s + 1) res . append (e + 1) return res # If no subarray is found return [- 1] if __name__ == "__main__" : arr = [15 , 2 , 4 , 8 , 9 , 5 , 10 , 23] target = 23 res = subarraySum (arr , target) print (" " . join (map (str , res))) C# using System ; using System.Collections.Generic ; class GfG { static List < int > subarraySum (int [] arr , int target) { // Initialize window int s = 0 , e = 0 ; List < int > res = new List < int > (); int curr = 0 ; for (int i = 0 ; i < arr . Length ; i ++) { curr += arr [i]; // If current sum becomes more or equal, // set end and try adjusting start if (curr >= target) { e = i ; // While current sum is greater, // remove starting elements of current window while (curr > target && s < e) { curr -= arr [s]; ++ s ; } // If we found a subarray if (curr == target) { res . Add (s + 1); res . Add (e + 1); return res ; } } } // If no subarray is found res . Add (- 1); return res ; } static void Main () { int [] arr = { 15 , 2 , 4 , 8 , 9 , 5 , 10 , 23 }; int target = 23 ; List < int > res = subarraySum (arr , target); foreach (var ele in res) Console . Write (ele + " "); } } JavaScript function subarraySum (arr , target) { // Initialize window let s = 0 , e = 0 ; let res = []; let curr = 0 ; for (let i = 0 ; i < arr . length ; i ++) { curr += arr [i]; // If current sum becomes more or equal, // set end and try adjusting start if (curr >= target) { e = i ; // While current sum is greater, // remove starting elements of current window while (curr > target && s < e) { curr -= arr [s]; s ++ ; } // If we found a subarray if (curr === target) { res . push (s + 1); res . push (e + 1); return res ; } } } // If no subarray is found return [- 1]; } // Driver Code let arr = [15 , 2 , 4 , 8 , 9 , 5 , 10 , 23]; let target = 23 ; let res = subarraySum (arr , target); console . log (res . join (' ')); Output 2 5 [Alternate Approach] Hashing + Prefix Sum - O(n) Time and O(n) S pace The above solution does not work for arrays with negative numbers. To handle all cases, we use hashing and prefix sum. The idea is to store the sum of elements of every prefix of the array in a hashmap, i.e, every index stores the sum of elements up to that index hashmap. So to check if there is a subarray with a sum equal to target , check for every index i , and sum up to that index as currSum . If there is a prefix with a sum equal to (currSum – target) , then the subarray with the given sum is found. To know more about the implementation, please refer Subarray with Given Sum – Handles Negative Numbers . Comment Article Tags: Article Tags: DSA Arrays Amazon Google Morgan Stanley Facebook Zoho Visa FactSet subarray sliding-window subarray-sum + 8 More