Longest Consecutive Subsequence - GeeksforGeeks

Source: https://www.geeksforgeeks.org/longest-consecutive-subsequence/

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 Longest Consecutive Subsequence Last Updated : 20 Aug, 2025 Given an array of integers, the task is to find the length of the longest subsequence such that elements in the subsequence are consecutive integers, the consecutive numbers can be in any order. Examples : Input: arr[] = [2, 6, 1, 9, 4, 5, 3] Output: 6 Explanation: The longest consecutive subsequence [2, 6, 1, 4, 5, 3]. Input : arr[] = [1,1,1,2,2,3] Output : 3 Explanation: The subsequence [1, 2,3] is the longest subsequence of consecutive elements Try it on GfG Practice Table of Content [Naive Approach] Using Sorting - O(n*log n) Time and O(1) Space [Expected Approach] Using Hashing - O(n) Time and O(n) Space [Naive Approach] Using Sorting - O(n*log n) Time and O(1) Space The idea is to sort the array and find the longest subarray with consecutive elements. Initialize the consecutive count with 1 and start iterating over the sorted array from the second element. Follow the steps below for implementation: Sort the array in ascending order. For each element arr[i], we can have three cases: arr[i] = arr[i - 1] , then the ith element is simply a duplicate element so skip it. . arr[i] = arr[i - 1] + 1 , then increase the consecutive count and update result if consecutive count is greater than result. arr[i] > arr[i - 1] , then reset the consecutive count to 1. After iterating over all the elements, return the result . C++ #include <iostream> #include <vector> #include <algorithm> using namespace std ; int longestConsecutive (vector < int >& arr) { // base case: if array is empty if (arr . empty ()) return 0 ; // sort the array sort (arr . begin (), arr . end ()); int res = 1 , cnt = 1 ; // find the maximum length by traversing the array for (int i = 1 ; i < arr . size (); i ++) { // Skip duplicates if (arr [i] == arr [i -1]) continue ; // Check if the current element is equal // to previous element + 1 if (arr [i] == arr [i - 1] + 1) { cnt ++ ; } else { cnt = 1 ; } res = max (res , cnt); } return res ; } int main () { vector < int > arr = {}; cout << longestConsecutive (arr); return 0 ; } C #include <stdio.h> #include <stdlib.h> #include <stdbool.h> int longestConsecutive (int * arr , int arrSize) { if (arrSize == 0) return 0 ; qsort (arr , arrSize , sizeof (int), (int (*) (const void * , const void *))) (int compare (const void * a , const void * b) { return (* (int *) a - * (int *) b); }); int res = 1 , cnt = 1 ; for (int i = 1 ; i < arrSize ; i ++) { if (arr [i] == arr [i -1]) continue ; if (arr [i] == arr [i -1] + 1) { cnt ++ ; } else { cnt = 1 ; } if (cnt > res) res = cnt ; } return res ; } int main () { int arr [] = {}; int arrSize = sizeof (arr) / sizeof (arr [0]); printf ("%d" , longestConsecutive (arr , arrSize)); return 0 ; } Java import java.util.Arrays ; class GfG { static int longestConsecutive (int [] arr) { if (arr . length == 0) return 0 ; // Sort the array Arrays . sort (arr); int res = 1 , cnt = 1 ; // Find the maximum length by traversing the array for (int i = 1 ; i < arr . length ; i ++) { // Skip duplicates if (arr [i] == arr [i - 1]) continue ; // Check if the current element is equal // to previous element + 1 if (arr [i] == arr [i - 1] + 1) { cnt ++ ; } else { // Reset the count cnt = 1 ; } // Update the result res = Math . max (res , cnt); } return res ; } public static void main (String [] args) { int [] arr = { 2 , 2 , 3 , 1 , 4 , 5 , 6 }; System . out . println (longestConsecutive (arr)); } } Python def longestConsecutive (arr): if not arr : return 0 # Sort the array arr . sort () res = 1 cnt = 1 # Find the maximum length by traversing the array for i in range (1 , len (arr)): # Skip duplicates if arr [i] == arr [i - 1]: continue # Check if the current element is equal # to previous element + 1 if arr [i] == arr [i - 1] + 1 : cnt += 1 else : # Reset the count cnt = 1 # Update the result res = max (res , cnt) return res if __name__ == "__main__" : arr = [2 , 2 , 3 , 1 , 4 , 5 , 6] print (longestConsecutive (arr)) C# using System ; using System.Linq ; class GfG { static int longestConsecutive (int [] arr) { if (arr . Length == 0) return 0 ; // Sort the array Array . Sort (arr); int res = 1 , cnt = 1 ; // Find the maximum length by traversing the array for (int i = 1 ; i < arr . Length ; i ++) { // Skip duplicates if (arr [i] == arr [i - 1]) continue ; // Check if the current element is equal // to previous element + 1 if (arr [i] == arr [i - 1] + 1) { cnt ++ ; } else { // Reset the count cnt = 1 ; } // Update the result res = Math . Max (res , cnt); } return res ; } static void Main (string [] args) { int [] arr = { 2 , 2 , 3 , 1 , 4 , 5 , 6 }; Console . WriteLine (longestConsecutive (arr)); } } JavaScript function longestConsecutive (arr) { if (arr . length === 0) return 0 ; // Sort the array arr . sort ((a , b) => a - b); let res = 1 , cnt = 1 ; // Find the maximum length by traversing the array for (let i = 1 ; i < arr . length ; i ++) { // Skip duplicates if (arr [i] === arr [i - 1]) continue ; // Check if the current element is equal // to previous element + 1 if (arr [i] === arr [i - 1] + 1) { cnt ++ ; } else { // Reset the count cnt = 1 ; } // Update the result res = Math . max (res , cnt); } return res ; } // Driver Code const arr = [2 , 2 , 3 , 1 , 4 , 5 , 6]; console . log (longestConsecutive (arr)); Output 6 [Expected Approach] Using Hashing - O(n) Time and O(n) Space The idea is to use Hashing. We first insert all elements in a Hash Set. Then, traverse over all the elements and check if the current element can be a starting element of a consecutive subsequence. If it is then start from X and keep on removing elements X + 1, X + 2 .... to find a consecutive subsequence. To check if the current element, say X can be a starting element, check if (X - 1) is present in the set. If (X - 1) is present in the set, then X cannot be starting of a consecutive subsequence. C++ #include <iostream> #include <unordered_set> #include <vector> using namespace std ; int longestConsecutive (vector < int > & arr) { unordered_set < int > st ; int res = 0 ; // Hash all the array elements for (int val : arr) st . insert (val); // check each possible sequence from the start then update optimal length for (int val : arr) { // if current element is the starting element of a sequence if (st . find (val) != st . end () && st . find (val -1) == st . end ()) { // Then check for next elements in the sequence int cur = val , cnt = 0 ; while (st . find (cur) != st . end ()) { // Remove this number to avoid recomputation st . erase (cur); cur ++ ; cnt ++ ; } // update optimal length res = max (res , cnt); } } return res ; } int main () { vector < int > arr = { 2 , 6 , 1 , 9 , 4 , 5 , 3 }; cout << longestConsecutive (arr); return 0 ; } Java import java.util.* ; class GfG { static int longestConsecutive (int [] arr) { Set < Integer > st = new HashSet <> (); int res = 0 ; // Hash all the array elements for (int val : arr) st . add (val); // Check each possible sequence from the start then update optimal length for (int val : arr) { // If current element is the starting element of a sequence if (st . contains (val) && ! st . contains (val - 1)) { // Then check for next elements in the sequence int cur = val , cnt = 0 ; while (st . contains (cur)) { // Remove this number to avoid recomputation st . remove (cur); cur ++ ; cnt ++ ; } // Update optimal length res = Math . max (res , cnt); } } return res ; } public static void main (String [] args) { int [] arr = { 2 , 6 , 1 , 9 , 4 , 5 , 3 }; System . out . println (longestConsecutive (arr)); } } Python def longestConsecutive (arr): st = set () res = 0 # Hash all the array elements for val in arr : st . add (val) # Check each possible sequence from the start # then update length for val in arr : # If current element is the starting element of a sequence if val in st and (val - 1) not in st : # Then check for next elements in the sequence cur = val cnt = 0 while cur in st : # Remove this number to avoid recomputation st . remove (cur) cur += 1 cnt += 1 # Update optimal length res = max (res , cnt) return res if __name__ == "__main__" : arr = [2 , 6 , 1 , 9 , 4 , 5 , 3] print (longestConsecutive (arr)) C# using System ; using System.Collections.Generic ; class GfG { static int longestConsecutive (int [] arr) { HashSet < int > st = new HashSet < int > (); int res = 0 ; // Hash all the array elements foreach (int val in arr) st . Add (val); // Check each possible sequence from the start then update optimal length foreach (int val in arr) { // If current element is the starting element of a sequence if (st . Contains (val) && ! st . Contains (val - 1)) { // Then check for next elements in the sequence int cur = val , cnt = 0 ; while (st . Contains (cur)) { // Remove this number to avoid recomputation st . Remove (cur); cur ++ ; cnt ++ ; } // Update optimal length res = Math . Max (res , cnt); } } return res ; } static void Main (string [] args) { int [] arr = { 2 , 6 , 1 , 9 , 4 , 5 , 3 }; Console . WriteLine (longestConsecutive (arr)); } } JavaScript function longestConsecutive (arr) { let st = new Set (); let res = 0 ; // Hash all the array elements for (let val of arr) { st . add (val); } // Check each possible sequence from the start then update // optimal length for (let val of arr) { // If current element is the starting element of a sequence if (st . has (val) && ! st . has (val - 1)) { // Then check for next elements in the sequence let cur = val , cnt = 0 ; while (st . has (cur)) { // Remove this number to avoid recomputation st . delete (cur); cur ++ ; cnt ++ ; } // Update optimal length res = Math . max (res , cnt); } } return res ; } // Driver Code const arr = [2 , 6 , 1 , 9 , 4 , 5 , 3]; console . log (longestConsecutive (arr)); Output 6 Comment Article Tags: Article Tags: Sorting Hash DSA Arrays Amazon Walmart Zoho Linkedin subsequence Arrays Hash + 7 More