Leaders in an array - GeeksforGeeks

Source: https://www.geeksforgeeks.org/leaders-in-an-array/

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 Leaders in an array Last Updated : 22 Jan, 2026 Given an array arr[] of size n , the task is to find all the Leaders in the array. An element is a Leader if it is greater than or equal to all the elements to its right side. Note: The rightmost element is always a leader. Examples: Input: arr[] = [16, 17, 4, 3, 5, 2] Output: [17 5 2] Explanation: 17 is greater than all the elements to its right i.e., [4, 3, 5, 2], therefore 17 is a leader. 5 is greater than all the elements to its right i.e., [2], therefore 5 is a leader. 2 has no element to its right, therefore 2 is a leader. Input: arr[] = [1, 2, 3, 4, 5, 2] Output: [5 2] Explanation: 5 is greater than all the elements to its right i.e., [2], therefore 5 is a leader. 2 has no element to its right, therefore 2 is a leader. Try it on GfG Practice Table of Content [Naive Approach] Using Nested Loops - O(n^2) Time and O(1) Space [Expected Approach] Using Suffix Maximum - O(n) Time and O(1) Space [Naive Approach] Using Nested Loops - O(n^2) Time and O(1) Space: Use two loops. The outer loop runs from 0 to size - 1 and one by one pick all elements from left to right. The inner loop compares the picked element to all the elements on its right side. If the picked element is greater than all the elements to its right side, then the picked element is the leader. C++ #include <bits/stdc++.h> using namespace std ; // Function to find the leaders in an array vector < int > leaders (vector < int >& arr) { vector < int > res ; int n = arr . size (); for (int i = 0 ; i < n ; i ++) { int j ; // Check elements to the right // of the current element for (j = i + 1 ; j < n ; j ++) { // If a larger element is found, // break the loop if (arr [i] < arr [j]) break ; } // If no larger element was found, // the current element is a leader if (j == n) res . push_back (arr [i]); } return res ; } int main () { vector < int > arr = { 16 , 17 , 4 , 3 , 5 , 2 }; vector < int > result = leaders (arr); for (int res : result) { cout << res << " " ; } cout << endl ; return 0 ; } C #include <stdio.h> #include <stdlib.h> // Function to find the leaders in an array int * leaders (int arr [], int n , int * resSize) { int * result = (int *) malloc (n * sizeof (int)); int count = 0 ; for (int i = 0 ; i < n ; i ++) { int j ; // Check elements to the right for (j = i + 1 ; j < n ; j ++) { // If a larger element is found if (arr [i] < arr [j]) break ; } // If no larger element was found if (j == n) result [count ++] = arr [i]; } * resSize = count ; return result ; } int main () { int arr [] = { 16 , 17 , 4 , 3 , 5 , 2 }; int n = sizeof (arr) / sizeof (arr [0]); int resSize ; int * result = leaders (arr , n , & resSize); for (int i = 0 ; i < resSize ; i ++) { printf ("%d " , result [i]); } printf (" \n "); free (result); return 0 ; } Java import java.util.ArrayList ; class GfG { // Function to find the leaders in an array static ArrayList < Integer > leaders (int [] arr) { ArrayList < Integer > result = new ArrayList <> (); int n = arr . length ; for (int i = 0 ; i < n ; i ++) { int j ; // Check elements to the right for (j = i + 1 ; j < n ; j ++) { // If a larger element is found if (arr [i] < arr [j]) break ; } // If no larger element was found if (j == n) result . add (arr [i]); } return result ; } public static void main (String [] args) { int [] arr = { 16 , 17 , 4 , 3 , 5 , 2 }; ArrayList < Integer > result = leaders (arr); for (int res : result) { System . out . print (res + " "); } System . out . println (); } } Python # Function to find the leaders in an array def leaders (arr): result = [] n = len (arr) for i in range (n): # Check elements to the right for j in range (i + 1 , n): # If a larger element is found if arr [i] < arr [j]: break else : # If no larger element was found result . append (arr [i]) return result if __name__ == "__main__" : arr = [16 , 17 , 4 , 3 , 5 , 2] result = leaders (arr) print (" " . join (map (str , result))) C# using System ; using System.Collections.Generic ; class GfG { // Function to find the leaders in an array static List < int > Leaders (int [] arr) { List < int > result = new List < int > (); int n = arr . Length ; for (int i = 0 ; i < n ; i ++) { int j ; // Check elements to the right for (j = i + 1 ; j < n ; j ++) { // If a larger element is found if (arr [i] < arr [j]) break ; } // If no larger element was found if (j == n) result . Add (arr [i]); } return result ; } static void Main () { int [] arr = { 16 , 17 , 4 , 3 , 5 , 2 }; List < int > result = Leaders (arr); foreach (int res

in result) { Console . Write (res + " "); } Console . WriteLine (); } } JavaScript // Function to find the leaders in an array function leaders (arr) { const result = []; const n = arr . length ; for (let i = 0 ; i < n ; i ++) { let j ; // Check elements to the right for (j = i + 1 ; j < n ; j ++) { // If a larger element is found if (arr [i] < arr [j]) break ; } // If no larger element was found if (j === n) result . push (arr [i]); } return result ; } const arr = [16 , 17 , 4 , 3 , 5 , 2]; const result = leaders (arr); console . log (result . join (" ")); Output 17 5 2 [Expected Approach] Using Suffix Maximum - O(n) Time and O(1) Space: The idea is to scan all the elements from right to left in an array and keep track of the maximum till now. When the maximum changes its value, add it to the result. Finally reverse the result C++ #include <bits/stdc++.h> using namespace std ; // Function to find the leaders in an array vector < int > leaders (vector < int >& arr) { vector < int > res ; int n = arr . size (); // Start with the rightmost element int maxRight = arr [n - 1]; // Rightmost element is always a leader res . push_back (maxRight); // Traverse the array from right to left for (int i = n - 2 ; i >= 0 ; i --) { if (arr [i] >= maxRight) { maxRight = arr [i]; res . push_back (maxRight); } } // Reverse the result array to maintain // original order reverse (res . begin (), res . end ()); return res ; } int main () { vector < int > arr = { 16 , 17 , 4 , 3 , 5 , 2 }; vector < int > res = leaders (arr); for (int x : res) { cout << x << " " ; } cout << endl ; return 0 ; } C #include <stdio.h> #include <stdlib.h> // Function to find the leaders in an array int * leaders (int arr [], int n , int * resSize) { int * result = (int *) malloc (n * sizeof (int)); int count = 0 ; // Start with the rightmost element int maxRight = arr [n - 1]; // Rightmost element is always a leader result [count ++] = maxRight ; // Traverse the array from right to left for (int i = n - 2 ; i >= 0 ; i --) { if (arr [i] >= maxRight) { maxRight = arr [i]; result [count ++] = maxRight ; } } // Reverse the result array for (int i = 0 ; i < count / 2 ; i ++) { int temp = result [i]; result [i] = result [count - i - 1]; result [count - i - 1] = temp ; } * resSize = count ; return result ; } int main () { int arr [] = { 16 , 17 , 4 , 3 , 5 , 2 }; int n = sizeof (arr) / sizeof (arr [0]); int resSize ; int * result = leaders (arr , n , & resSize); for (int i = 0 ; i < resSize ; i ++) { printf ("%d " , result [i]); } printf (" \n "); return 0 ; } Java import java.util.ArrayList ; import java.util.Collections ; class GfG { // Function to find the leaders in an array static ArrayList < Integer > leaders (int [] arr) { ArrayList < Integer > result = new ArrayList <> (); int n = arr . length ; // Start with the rightmost element int maxRight = arr [n - 1] ; // Rightmost element is always a leader result . add (maxRight); // Traverse the array from right to left for (int i = n - 2 ; i >= 0 ; i --) { if (arr [i] >= maxRight) { maxRight = arr [i] ; result . add (maxRight); } } // Reverse the result list to maintain // original order Collections . reverse (result); return result ; } public static void main (String [] args) { int [] arr = { 16 , 17 , 4 , 3 , 5 , 2 }; ArrayList < Integer > result = leaders (arr); for (int res : result) { System . out . print (res + " "); } System . out . println (); } } Python # Function to find the leaders in an array def leaders (arr): result = [] n = len (arr) # Start with the rightmost element maxRight = arr [- 1] # Rightmost element is always a leader result . append (maxRight) # Traverse the array from right to left for i in range (n - 2 , - 1 , - 1): if arr [i] >= maxRight : maxRight = arr [i] result . append (maxRight) # Reverse the result list to maintain # original order result . reverse () return result if __name__ == "__main__" : arr = [16 , 17 , 4 , 3 , 5 , 2] result = leaders (arr) print (" " . join (map (str , result))) C# using System ; using System.Collections.Generic ; class GfG { // Function to find the leaders in an array static List < int > Leaders (int [] arr){ List < int > result = new List < int > (); int n = arr . Length ; // Start with the rightmost element int maxRight = arr [n - 1]; // Rightmost element is always a leader result . Add (maxRight); // Traverse the array from right to left for (int i = n - 2 ; i >= 0 ; i --) { if (arr [i] >= maxRight) { maxRight = arr [i]; result . Add (maxRight); } } // Reverse the result list to maintain original // order result . Reverse (); return result ; } static void Main (){ int [] arr = { 16 , 17 , 4 , 3 , 5 , 2 }; List < int > result = Leaders (arr); foreach (int res in result){ Console . Write (res + " "); } Console . WriteLine (); } } JavaScript // Function to find the leaders in an array function leaders (arr) { const result = []; const n = arr . length ; // Start with the rightmost element let maxRight = arr [n - 1]; // Rightmost element is always a leader result . push (maxRight); // Traverse the array from right to left for (let i = n - 2 ; i >= 0 ; i --) { if (arr [i] >= maxRight) { maxRight = arr [i]; result . push (maxRight); } } // Reverse the result array to maintain // original order result . reverse (); return result ; } // Driver code const arr = [16 , 17 , 4 , 3 , 5 , 2]; const result = leaders (arr); console . log (result . join (" ")); Output 17 5 2 Illustration: arr[] = {16, 17, 4, 3, 5, 2} Initially : maxRight = 2, res[] = { 2 } i = 4, maxRight = 5, res[] = { 2, 5 } i= 3, maxRight = 5, res[] = { 2, 5 } i = 2, maxRight = 5, res[] = { 2, 5 } i = 1, maxRight = 17, res[] = { 2, 5, 17 } i = 0, maxRight = 17, res[] = { 2, 5, 17 } Reverse res[] = {17, 5, 2} and return Comment Article Tags: Article Tags: DSA Arrays Amazon Payu