Next Greater Element in Array - GeeksforGeeks

Source: https://www.geeksforgeeks.org/next-greater-element/

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 Next Greater Element in Array Last Updated : 10 Sep, 2025 Given an array arr[] of integers, determine the Next Greater Element (NGE) for every element in the array, maintaining the order of appearance. The Next Greater Element for an element x is defined as the first element to the right of x in the array that is strictly greater than x. If no such element exists for an element, its Next Greater Element is -1. Examples: Input : arr[] = [1, 3, 2, 4] Output : [3, 4, 4, -1] Explanation : The next larger element to 1 is 3, 3 is 4, 2 is 4 and for 4, since it doesn't exist, it is -1. Input : arr[] = [6, 8, 0, 1, 3] Output : [8, -1, 1, 3, -1] Explanation : The next larger element to 6 is 8, for 8 there is no larger elements hence it is -1, for 0 it is 1 , for 1 it is 3 and then for 3 there is no larger element on right and hence -1. Try it on GfG Practice Table of Content [Naive Approach] Using Nested Loops - O(n^2) Time and O(1) Space [Expected Approach] Using Stack - O(n) Time and O(n) Space [Naive Approach] Using Nested Loops - O(n 2) Time and O(1) Space The idea is to look at all the elements to its right and check if there's a larger element for each element in the array. If we find one, we store it as the next greater element. If no greater element is found, we store -1. This is done using two nested loops: the outer loop goes through each element, and the inner loop searches the rest of the array for a greater element. C++ #include <iostream> #include <vector> using namespace std ; vector < int > nextLargerElement (vector < int > & arr) { int n = arr . size (); vector < int > res (n , -1); // iterate through each element in the array for (int i = 0 ; i < n ; i ++) { // check for the next greater element // in the rest of the array for (int j = i + 1 ; j < n ; j ++) { if (arr [j] > arr [i]) { res [i] = arr [j]; break ; } } } return res ; } int main () { vector < int > arr = { 6 , 8 , 0 , 1 , 3 }; vector < int > res = nextLargerElement (arr); for (int x : res) { cout << x << " " ; } return 0 ; } Java import java.util.ArrayList ; class GfG { static ArrayList < Integer > nextLargerElement (int [] arr) { int n = arr . length ; ArrayList < Integer > res = new ArrayList <> (); // initialize res with -1 for all elements for (int i = 0 ; i < n ; i ++) { res . add (- 1); } // iterate through each element in the array for (int i = 0 ; i < n ; i ++) { // check for the next greater element // in the rest of the array for (int j = i + 1 ; j < n ; j ++) { if (arr [j] > arr [i]) { res . set (i , arr [j]); break ; } } } return res ; } public static void main (String [] args) { int [] arr = { 6 , 8 , 0 , 1 , 3 }; ArrayList < Integer > res = nextLargerElement (arr); for (int x : res) { System . out . print (x + " "); } } } Python def nextLargerElement (arr): n = len (arr) res = [- 1] * n # Iterate through each element in the array for i in range (n): # Check for the next greater element # in the rest of the array for j in range (i + 1 , n): if arr [j] > arr [i]: res [i] = arr [j] break return res if __name__ == "__main__" : arr = [6 , 8 , 0 , 1 , 3] res = nextLargerElement (arr) print (" " . join (map (str , res))) C# using System ; using System.Collections.Generic ; class GfG { static List < int > nextLargerElement (int [] arr) { int n = arr . Length ; List < int > res = new List < int > (new int [n]); // Initialize res with -1 for all elements for (int i = 0 ; i < n ; i ++) { res [i] = - 1 ; } // Iterate through each element in the array for (int i = 0 ; i < n ; i ++) { // Check for the next greater element // in the rest of the array for (int j = i + 1 ; j < n ; j ++) { if (arr [j] > arr [i]) { res [i] = arr [j]; break ; } } } return res ; } static void Main (string [] args) { int [] arr = { 6 , 8 , 0 , 1 , 3 }; List < int > res = nextLargerElement (arr); foreach (int x in res) { Console . Write (x + " "); } } } JavaScript function nextLargerElement (arr) { let n = arr . length ; let res = new Array (n). fill (- 1); // Iterate through each element in the array for (let i = 0 ; i < n ; i ++) { // Check for the next greater element // in the rest of the array for (let j = i + 1 ; j < n ; j ++) { if (arr [j] > arr [i]) { res [i] = arr [j]; break ; } } } return res ; } // Driver Code let arr = [6 , 8 , 0 , 1 , 3]; let res = nextLargerElement (arr); console . log (res . join (" ")); Output 8 -1 1 3 -1 [Expected Approach] Using Stack - O(n) Time and O(n) Space The idea is to use a monotonic decreasing stack (stack that maintains elements in decreasing order). We traverse the array from right to left. For each element, we pop elements from the stack that are smaller than or equal to it, since they cannot be the next greater element. If the stack is not empty, the top of the stack is the next greater element. Finally, we push the current element onto the stack. Illustration: C++ #include <iostream> #include <vector> #include <stack> using namespace std ; vector < int > nextLargerElement (vector < int > & arr) { int n = arr . size (); vector < int > res (n , -1); stack < int > stk ; for (int i = n - 1 ; i >= 0 ; i --) { // Pop elements from the stack that are less // than or equal to the current element while (! stk . empty () && stk . top () <= arr [i]) { stk . pop (); } // If the stack is not empty, the top element // is the next greater element if (! stk . empty ()) { res [i] = stk . top (); } // Push the current element onto the stack stk . push (arr [i]); } return res ; } int main () { vector < int > arr = { 6 , 8 , 0 , 1 , 3 }; vector < int > res = nextLargerElement (arr); for (int x : res) { cout << x << " " ; } return 0 ; } Java import java.util.ArrayList ; import java.util.Stack ; class GfG { static ArrayList < Integer > nextLargerElement (int [] arr) { int n = arr . length ; ArrayList < Integer > res = new ArrayList <> (); Stack < Integer > stk = new Stack <> (); // Initialize res with -1 for all elements for (int i = 0 ; i < n ; i ++) { res . add (- 1); } for (int i = n - 1 ; i >= 0 ; i --) { // Pop elements from the stack that are less // than or equal to the current element while (! stk . isEmpty () && stk . peek () <= arr [i]) { stk . pop (); } // If the stack is not empty, the top element // is the next greater element if (! stk . isEmpty ()) { res . set (i , stk . peek ()); } // Push the current element onto the stack stk . push (arr [i]); } return res ; } public static void main (String [] args) { int [] arr = { 6 , 8 , 0 , 1 , 3 }; ArrayList < Integer > res = nextLargerElement (arr); for (int x : res) { System . out . print (x + " "); } } } Python def nextLargerElement (arr): n = len (arr) res = [- 1] * n stk = [] # Traverse the array from right to left for i in range (n - 1 , - 1 , - 1): # Pop elements from the stack that are less # than or equal to the current element while stk and arr [stk [- 1]] <= arr [i]: stk . pop () # If the stack is not empty, the element at the # top of the stack is the next greater element if stk : res [i] = arr [stk [- 1]] # Push the current index onto the stack stk . append (i) return res if __name__ == "__main__" : arr = [6 , 8 , 0 , 1 , 3] res = nextLargerElement (arr) print (" " . join (map (str , res))) C# using System ; using System.Collections.Generic ; class GfG { static List < int > nextLargerElement (int [] arr) { int n = arr . Length ; List < int > res = new List < int > (new int [n]); Stack < int > stk = new Stack < int > (); // Initialize res with -1 for all elements for (int i = 0 ; i < n ; i ++) { res [i] = - 1 ; } // Traverse the array from right to left for (int i = n - 1 ; i >= 0 ; i --) { // Pop elements from the stack that are less // than or equal to the current element while (stk . Count > 0 && stk . Peek () <= arr [i]) { stk . Pop (); } // If the stack is not empty, the top element // is the next greater element if (stk . Count > 0) { res [i] = stk . Peek (); } // Push the current element onto the stack stk . Push (arr [i]); } return res ; } static void Main (string [] args) { int [] arr = { 6 , 8 , 0 , 1 , 3 }; List < int > res = nextLargerElement (arr); foreach (int x in res) { Console . Write (x + " "); } } } JavaScript function nextLargerElement (arr) { let n = arr . length ; let res = new Array (n). fill (- 1); let stk = []; // Traverse the array from right to left for (let i = n - 1 ; i >= 0 ; i --) { // Pop elements from the stack that are less // than or equal to the current element while (stk . length > 0 && stk [stk . length - 1] <= arr [i]) { stk . pop (); } // If the stack is not empty, the top element // is the next greater element if (stk . length > 0) { res [i] = stk [stk . length - 1]; } // Push the current element onto the stack stk . push (arr [i]); } return res ; } // Driver Code let arr = [6 , 8 , 0 , 1 , 3]; let res = nextLargerElement (arr); console . log (res . join (" ")); Output 8 -1 1 3 -1 Comment Article Tags: Article Tags: Stack DSA Arrays Amazon Samsung Snapdeal Payu Zoho Informatica CouponDunia + 6 More