First negative integer in every window of size k - GeeksforGeeks

Source: https://www.geeksforgeeks.org/first-negative-integer-every-window-size-k/

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 First negative integer in every window of size k Last Updated : 14 May, 2025 Given an array and a positive integer k, find the first negative integer for each window(contiguous subarray) of size k. If a window does not contain a negative integer, then print 0 for that window. Examples: Input: arr[] = [-8, 2, 3, -6, 1] , k = 2 Output: [-8, 0, -6, -6] Explanation: First negative integer for each window of size 2 [-8, 2] = -8 [2, 3] = 0 (does not contain a negative integer) [3, -6] = -6 [-6, 10] = -6 Input: arr[] = [12, -1, -7, 8, -15, 30, 16, 28], k = 3 Output: [-1, -1, -7, -15, -15, 0] Explanation : First negative integer for each window of size 3 [12, -1, -7] = -1 [-1,-7, 8] = -1 [-7, 8, -15] = -7 [8, -15, 30] = -15 [-15, 30, 16] = -15 [30, 16, 28] = 0 Try it on GfG Practice Table of Content [Naive Approach] Nested Loops - O(n*k) time and O(1) space [Better Approach] Sliding Window with Deque technique - O(n) time and O(k) space [Expected Approach] Sliding Window with Index Tracking - O(n) time and O(1) space [Naive Approach] Nested Loops - O(n*k) time and O(1) space The idea is to loop through the array, and for each window of size k, check each element to find the first negative number. If a negative number is found, it is printed immediately, and the inner loop breaks. If no negative number is found in the window, it prints 0 . This ensures that each window is processed individually, and the result is output for all windows in sequence. C++ #include <bits/stdc++.h> using namespace std ; // function to find the first negative integer // in every window of size k vector < int > firstNegInt (vector < int >& arr , int k) { vector < int > res ; int n = arr . size (); // Loop for each subarray(window) of size k for (int i = 0 ; i <= (n - k); i ++) { bool found = false ; // traverse through the current window for (int j = 0 ; j < k ; j ++) { // if a negative integer is found, then // it is the first negative integer for // the current window. Set the flag and break if (arr [i + j] < 0) { res . push_back (arr [i + j]); found = true ; break ; } } // if the current window does not contain // a negative integer if (! found) { res . push_back (0); } } return res ; } int main () { vector < int > arr = { 12 , -1 , -7 , 8 , -15 , 30 , 16 , 28 }; int k = 3 ; vector < int > res = firstNegInt (arr , k); cout << "[" ; for (int i = 0 ; i < res . size (); i ++) { cout << res [i]; if (i < res . size () - 1) { cout << ", " ; } } cout << "]" ; return 0 ; } Java import java.util.* ; // function to find the first negative integer // in every window of size k public class GfG { public static int [] firstNegInt (int [] arr , int k) { List < Integer > res = new ArrayList <> (); int n = arr . length ; // Loop for each subarray(window) of size k for (int i = 0 ; i <= (n - k); i ++) { boolean found = false ; // traverse through the current window for (int j = 0 ; j < k ; j ++) { // if a negative integer is found, then // it is the first negative integer for // the current window. Set the flag and break if (arr [i + j] < 0) { res . add (arr [i + j]); found = true ; break ; } } // if the current window does not contain // a negative integer if (! found) { res . add (0); } } // Convert List to int[] return res . stream (). mapToInt (i -> i). toArray (); } public static void main (String [] args) { int [] arr = { 12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28 }; int k = 3 ; int [] res = firstNegInt (arr , k); System . out . print (Arrays . toString (res)); } } Python # function to find the first negative integer # in every window of size k def firstNegInt (arr , k): res = [] n = len (arr) # Loop for each subarray(window) of size k for i in range (n - k + 1): found = False # traverse through the current window for j in range (k): # if a negative integer is found, then # it is the first negative integer for # the current window. Set the flag and break if arr [i + j] < 0 : res . append (arr [i + j]) found = True break # if the current window does not contain # a negative integer if not found : res . append (0) return res arr = [12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28] k = 3 res = firstNegInt (arr , k) print (res) C# using System ; using System.Collections.Generic ; class GfG { // function to find the first

negative integer // in every window of size k public static List < int > FirstNegInt (int [] arr , int k) { List < int > res = new List < int > (); int n = arr . Length ; // Loop for each subarray(window) of size k for (int i = 0 ; i <= (n - k); i ++) { bool found = false ; // traverse through the current window for (int j = 0 ; j < k ; j ++) { // if a negative integer is found, then // it is the first negative integer for // the current window. Set the flag and break if (arr [i + j] < 0) { res . Add (arr [i + j]); found = true ; break ; } } // if the current window does not contain // a negative integer if (! found) { res . Add (0); } } return res ; } static void Main () { int [] arr = { 12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28 }; int k = 3 ; List < int > res = FirstNegInt (arr , k); Console . WriteLine ("[" + string . Join (", " , res) + "]"); } } JavaScript // function to find the first negative integer // in every window of size k function firstNegInt (arr , k) { let res = []; let n = arr . length ; // Loop for each subarray(window) of size k for (let i = 0 ; i <= (n - k); i ++) { let found = false ; // traverse through the current window for (let j = 0 ; j < k ; j ++) { // if a negative integer is found, then // it is the first negative integer for // the current window. Set the flag and break if (arr [i + j] < 0) { res . push (arr [i + j]); found = true ; break ; } } // if the current window does not contain // a negative integer if (! found) { res . push (0); } } return res ; } const arr = [12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28]; const k = 3 ; const res = firstNegInt (arr , k); console . log (res); Output [-1, -1, -7, -15, -15, 0] Time Complexity : The outer loop runs n–k+1 times, and for each iteration, the inner loop runs k times. This gives us a time complexity of O((n–k+1)×k), which simplifies to O(n*k) when k is much smaller than n. If k is close to n, the complexity becomes O(k). Auxiliary Space : O(1) as it is using constant space for variables [Better Approach] Sliding Window with Deque technique - O(n) time and O(k) space We create a Dequeue, dq of capacity k, that stores only useful elements of the current window of k elements. An element is useful if it is in the current window and it is a negative integer. We process all array elements one by one and maintain dq to contain useful elements of current window and these useful elements are all negative integers. For a particular window, if dq is not empty then the element at front of the dq is the first negative integer for that window, else that window does not contain a negative integer. It is a variation of the problem of Sliding Window Maximum . C++ #include <iostream> #include <deque> #include <vector> using namespace std ; // Function to find the first negative integer // in every window of size k vector < int > firstNegInt (vector < int >& arr , int k) { deque < int > dq ; vector < int > res ; int n = arr . size (); // Process the first window of size k for (int i = 0 ; i < k ; i ++) { if (arr [i] < 0) { dq . push_back (i); } } // Process the rest of the elements for (int i = k ; i < n ; i ++) { // If there is any negative number in the window, add it to the result if (! dq . empty ()) { res . push_back (arr [dq . front ()]); } else { res . push_back (0); } // Remove elements which are out of this window while (! dq . empty () && dq . front () <= i - k) { dq . pop_front (); } // Add the current element if it is negative if (arr [i] < 0) { dq . push_back (i); } } // For the last window, process it separately if (! dq . empty ()) { res . push_back (arr [dq . front ()]); } else { res . push_back (0); } return res ; } int main () { vector < int > arr = { 12 , -1 , -7 , 8 , -15 , 30 , 16 , 28 }; int k = 3 ; // Get the result from the function vector < int > result = firstNegInt (arr , k); // Print the result in the required format cout << "[" ; for (size_t i = 0 ; i < result . size (); i ++) { cout << result [i]; if (i != result . size () - 1) cout << ", " ; } cout << "]" << endl ; return 0 ; } Java import java.util.* ; public class GfG { public static int [] firstNegInt (int [] arr , int k) { Deque < Integer > dq = new LinkedList <> (); List < Integer > res = new ArrayList <> (); int n = arr . length ; // Process first k (or first window) elements for (int i = 0 ; i < k ; i ++) if (arr [i] < 0) dq . addLast (i); // Process rest of the elements, i.e., // from arr[k] to arr[n-1] for (int i = k ; i < n ; i ++) { if (! dq . isEmpty ()) res . add (arr [dq . peekFirst ()]); else res . add (0); // Remove the elements which are out of // this window while (! dq . isEmpty () && dq . peekFirst () < (i - k + 1)) dq . pollFirst (); // Add current element at the rear // of dq if it is a negative integer if (arr [i] < 0) dq . addLast (i); } // Print the first negative integer of // the last window if (! dq . isEmpty ()) res . add (arr [dq . peekFirst ()]); else res . add (0); return res . stream (). mapToInt (i -> i). toArray (); } public static void main (String [] args) { int [] arr = { 12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28 }; int k = 3 ; int [] result = firstNegInt (arr , k); // Print the result in the required format System . out . print (Arrays . toString (result)); } } Python # Function to find the first negative integer # in every window of size k def firstNegInt (arr , k): from collections import deque dq = deque () res = [] n = len (arr) # Process first k (or first window) elements for i in range (k): if arr [i] < 0 : dq . append (i) # Process rest of the elements, i.e., # from arr[k] to arr[n-1] for i in range (k , n): if dq : res . append (arr [dq [0]]) else : res . append (0) # Remove the elements which are out of # this window while dq and dq [0] < (i - k + 1): dq . popleft () # Add current element at the rear # of dq if it is a negative integer if arr [i] < 0 : dq . append (i) # Print the first negative integer of # the last window if dq : res . append (arr [dq [0]]) else : res . append (0) return res # Driver program to test the above function arr = [12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28] k = 3 result = firstNegInt (arr , k) # Print the result in the required format print (result) C# using System ; using System.Collections.Generic ;

class GfG { public static int [] FirstNegInt (int [] arr , int k) { Queue < int > dq = new Queue < int > (); List < int > res = new List < int > (); int n = arr . Length ; // Process first k (or first window) elements for (int i = 0 ; i < k ; i ++) if (arr [i] < 0) dq . Enqueue (i); // Process rest of the elements, i.e., // from arr[k] to arr[n-1] for (int i = k ; i < n ; i ++) { if (dq . Count > 0) res . Add (arr [dq . Peek ()]); else res . Add (0); // Remove the elements which are out of // this window while (dq . Count > 0 && dq . Peek () < (i - k + 1)) dq . Dequeue (); // Add current element at the rear // of dq if it is a negative integer if (arr [i] < 0) dq . Enqueue (i); } // Print the first negative integer of // the last window if (dq . Count > 0) res . Add (arr [dq . Peek ()]); else res . Add (0); return res . ToArray (); } static void Main () { int [] arr = { 12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28 }; int k = 3 ; int [] result = FirstNegInt (arr , k); // Print the result in the required format Console . WriteLine ("[" + string . Join (", " , result) + "]"); } } JavaScript // Function to find the first negative integer // in every window of size k function firstNegInt (arr , k) { let dq = []; let res = []; let n = arr . length ; // Process first k (or first window) elements for (let i = 0 ; i < k ; i ++) { if (arr [i] < 0) { dq . push (i); } } // Process rest of the elements, i.e., // from arr[k] to arr[n-1] for (let i = k ; i < n ; i ++) { if (dq . length > 0) { res . push (arr [dq [0]]); } else { res . push (0); } // Remove the elements which are out of // this window while (dq . length > 0 && dq [0] < (i - k + 1)) { dq . shift (); } // Add current element at the rear // of dq if it is a negative integer if (arr [i] < 0) { dq . push (i); } } // Print the first negative integer of // the last window if (dq . length > 0) { res . push (arr [dq [0]]); } else { res . push (0); } return res ; } // Driver program to test the above function let arr = [12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28]; let k = 3 ; let result = firstNegInt (arr , k); // Print the result in the required format console . log (JSON . stringify (result)); Output [-1, -1, -7, -15, -15, 0] [Expected Approach] Sliding Window with Index Tracking - O(n) time and O(1) space This approach uses a sliding window technique to find the first negative integer in each window of size k . It keeps track of the index of the first negative integer within the current window and skips over positive elements or those that have moved out of the window. This ensures that the first negative element is efficiently identified for each window as it slides. C++ #include <iostream> #include <vector> using namespace std ; vector < int > firstNegInt (vector < int >& arr , int k) { int fstNegIdx = 0 ; vector < int > res ; int n = arr . size (); // Use size() for vectors for (int i = k - 1 ; i < n ; i ++) { // Skip out of window and positive elements while ((fstNegIdx < i) && (fstNegIdx <= i - k || arr [fstNegIdx] >= 0)) { fstNegIdx ++ ; } // Check if a negative element is found, // otherwise use 0 if (fstNegIdx < n && arr [fstNegIdx] < 0) { res . push_back (arr [fstNegIdx]); } else { res . push_back (0); } } return res ; } int main () { vector < int > arr = { 12 , -1 , -7 , 8 , -15 , 30 , 16 , 28 }; int k = 3 ; vector < int > res = firstNegInt (arr , k); cout << "[" ; for (size_t i = 0 ; i < res . size (); i ++) { cout << res [i]; if (i != res . size () - 1) cout << ", " ; } cout << "]" ; return 0 ; } Java import java.util.ArrayList ; import java.util.List ; public class Main { public static List < Integer > firstNegInt (int [] arr , int k) { int fstNegIdx = 0 ; List < Integer > res = new ArrayList <> (); int n = arr . length ; for (int i = k - 1 ; i < n ; i ++) { // Skip out of window and positive elements while (fstNegIdx < i && (fstNegIdx <= i - k || arr [fstNegIdx] >= 0)) { fstNegIdx ++ ; } // Check if a negative element is found, // otherwise use 0 if (fstNegIdx < n && arr [fstNegIdx] < 0) { res . add (arr [fstNegIdx]); } else { res . add (0); } } return res ; } public static void main (String [] args) { int [] arr = { 12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28 }; int k = 3 ; List < Integer > res = firstNegInt (arr , k); System . out . println (res); } } Python def first_neg_int (arr , k): fst_neg_idx = 0 res = [] n = len (arr) for i in range (k - 1 , n): # Skip out of window and positive elements while fst_neg_idx < i and (fst_neg_idx <= i - k or arr [fst_neg_idx] >= 0): fst_neg_idx += 1 # Check if a negative element is found, # otherwise use 0 if fst_neg_idx < n and arr [fst_neg_idx] < 0 : res . append (arr [fst_neg_idx]) else : res . append (0) return res # Driver code arr = [12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28] k = 3 res = first_neg_int (arr , k) print (res) C# using System ; using System.Collections.Generic ; class GfG { public static List < int > FirstNegInt (int [] arr , int k) { int fstNegIdx = 0 ; List < int > res = new List < int > (); int n = arr . Length ; for (int i = k - 1 ; i < n ; i ++) { // Skip out of window and positive elements while (fstNegIdx < i && (fstNegIdx <= i - k || arr [fstNegIdx] >= 0)) { fstNegIdx ++ ; } // Check if a negative element is found, // otherwise use 0 if (fstNegIdx < n && arr [fstNegIdx] < 0) { res . Add (arr [fstNegIdx]); } else { res . Add (0); } } return res ; } static void Main () { int [] arr = { 12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28 }; int k = 3 ; List < int > res = FirstNegInt (arr , k); Console . WriteLine (string . Join (", " , res)); } } JavaScript function firstNegInt (arr , k) { let fstNegIdx = 0 ; let res = []; let n = arr . length ; for (let i = k - 1 ; i < n ; i ++) { // Skip out of window and positive elements while (fstNegIdx < i && (fstNegIdx <= i - k || arr [fstNegIdx] >= 0)) { fstNegIdx ++ ; } // Check if a negative element is found, // otherwise use 0 if (fstNegIdx < n && arr [fstNegIdx] < 0) { res . push (arr [fstNegIdx]); } else { res . push (0); } } return res ; } // Driver code let arr = [12 , - 1 , - 7 , 8 , - 15 , 30 , 16 , 28]; let k = 3 ; let res = firstNegInt (arr , k); console . log (res); Output [-1, -1, -7, -15, -15, 0] Comment Article Tags: Article Tags: Queue DSA Arrays Amazon sliding-window + 1 More