Find the first repeating element in an array of integers - GeeksforGeeks

Source: https://www.geeksforgeeks.org/find-first-repeating-element-array-integers/

Find the first repeating element in an array of integers Last Updated : 15 Jul, 2025 Given an array of integers arr[] , The task is to find the index of first repeating element in it i.e. the element that occurs more than once and whose index of the first occurrence is the smallest. Examples: Input: arr[] = {10, 5, 3, 4, 3, 5, 6} Output: 5 Explanation: 5 is the first element that repeats Input: arr[] = {6, 10, 5, 4, 9, 120, 4, 6, 10} Output: 6 Explanation: 6 is the first element that repeats Try it on GfG Practice Naive Approach - O(n^2) Time and O(1) Space Run two nested loops, the outer loop picks an element one by one, and the inner loop checks whether the element is repeated or not. Once a repeating element is found, break the loops and print the element. C++ // Cpp to find first repeating element // using Naive approach in O(n^2) Time and O(1) Space #include <bits/stdc++.h> using namespace std ; // Function to find the index of the first // repeating element in a vector int firstRepeatingElement (vector < int > & arr) { int n = arr . size (); // Nested loop to check for repeating elements for (int i = 0 ; i < n ; i ++) { for (int j = i + 1 ; j < n ; j ++) { if (arr [i] == arr [j]) { return i ; } } } // If no repeating element is found, return -1 return -1 ; } int main () { vector < int > arr = { 10 , 5 , 3 , 4 , 3 , 5 , 6 }; int index = firstRepeatingElement (arr); if (index == -1) cout << "No repeating found!" << endl ; else cout << "First repeating is " << arr [index] << endl ; return 0 ; } C #include <stdio.h> int firstRepeatingElement (int arr [], int n) { // Nested loop to check for repeating elements for (int i = 0 ; i < n ; i ++) { for (int j = i + 1 ; j < n ; j ++) { if (arr [i] == arr [j]) { return i ; } } } // If no repeating element is found return -1 ; } int main () { int arr [] = { 10 , 5 , 3 , 4 , 3 , 5 , 6 }; int n = sizeof (arr) / sizeof (arr [0]); int index = firstRepeatingElement (arr , n); if (index == -1) printf ("No repeating found! \n "); else printf ("First repeating is %d \n " , arr [index]); return 0 ; } Java // Java program to find first repeating element // using Naive approach in O(n^2) Time and O(1) Space import java.util.* ; class GfG { // Function to find the index of first repeating element in an array static int firstRepeatingElement (int [] arr , int n) { // Nested loop to check for repeating elements for (int i = 0 ; i < n ; i ++) { for (int j = i + 1 ; j < n ; j ++) { // If a repeating element is found, return its index if (arr [i] == arr [j]) { return i ; } } } // If no repeating element is found, return -1 return - 1 ; } // Driver code public static void main (String [] args) { // Initializing an array and its size int [] arr = { 10 , 5 , 3 , 4 , 3 , 5 , 6 }; int n = arr . length ; // Finding the index of first repeating element int index = firstRepeatingElement (arr , n); // Checking if any repeating element is found or not if (index == - 1) { System . out . println ("No repeating element found!"); } else { System . out . println ("First repeating element is " + arr [index]); } } } Python # Python program to find first repeating element # using Naive approach in O(n^2) Time and O(1) Space # Function to find the index of first repeating element in an array def firstRepeatingElement (arr , n): # Nested loop to check for repeating elements for i in range (n): for j in range (i + 1 , n): # If a repeating element is found, return its index if arr [i] == arr [j]: return i # If no repeating element is found, return -1 return - 1 # Driver code if __name__ == '__main__' : # Initializing an array and its size arr = [10 , 5 , 3 , 4 , 3 , 5 , 6] n = len (arr) # Finding the index of first repeating element index = firstRepeatingElement (arr , n) # Checking if any repeating element is found or not if index == - 1 : print ("No repeating element found!") else : print ("First repeating element is" , arr [index]) C# // C# program to find first repeating element // using Naive approach in O(n^2) Time and O(1) Space using System ; class GfG { static void Main () { // Initializing an array and its size int [] arr = { 10 , 5 , 3 , 4 , 3 , 5 , 6 }; int n = arr . Length ; // Initializing variables to keep track of the minimum index and // minimum value of the repeating element int minIndex = n ; int minValue = int . MaxValue ; // Creating a dictionary to store the last seen index of each element var dict = new System . Collections . Generic . Dictionary < int , int > (); // Iterating over the array from left to right for (int i = 0 ; i < n ; i ++) { int val = arr [i]; // If the element is not in the dictionary, add it with its index if (! dict . ContainsKey (val)) { dict [val] = i ; } // If the element is already in the dictionary, update the minimum index // and minimum value if necessary else { int index = dict [val]; if (index < minIndex || (index == minIndex && val < minValue)) { minIndex = index ; minValue = val ; } } } // Checking if any repeating element is found or not if (minIndex == n) { Console . WriteLine ("No repeating element found!"); } else { Console . WriteLine ("First repeating element is " + minValue + " at index " + minIndex); } } } JavaScript // JavaScript program to find first repeating element // using Naive approach in O(n^2) Time and O(1) Space function firstRepeatingElement (arr) { // Nested loop to check for repeating elements for (let i = 0 ; i < arr . length ; i ++) { for (let j = i + 1 ; j < arr . length ; j ++) { // If a repeating element is found, return its index if (arr [i] === arr [j]) { return i ; } } } // If no repeating element is found, return -1 return - 1 ; } const arr = [10 , 5 , 3 , 4 , 3 , 5 , 6]; // Finding the index of first repeating element const index = firstRepeatingElement (arr); if (index === - 1) { console . log ("No repeating element found!"); } else { console . log ("First repeating element is" , arr [index]); } Output First repeating is 5 Expected Approach - Hash Set - O(n) Time and O(n) Space Follow the steps below to implement the idea: Initialize an empty Hash Set s Run a for loop for each index i If the current element is present in the hash set, then return i Else insert arr[i] in s. Return -1 if no repeating element found Below is the implementation of the above approach. C++ // Cpp program to find first repeating element // using Hash Set in O(n) Time and O(n) Space #include <iostream> #include <vector> #include <unordered_set> #include <climits> using namespace std ; int firstRepeatingElement (const vector < int >& arr) { unordered_set < int > s ; // If an element is already present, return it // else insert it int minEle = INT_MAX ; for (int i = arr . size () - 1 ; i >= 0 ; i --) { if (s . find (arr [i]) != s . end ()) { minEle = min (minEle , i); } s . insert (arr [i]); } // If no element repeats return minEle == INT_MAX ? -1 : minEle ; } int main () { vector < int > arr = { 10 , 5 , 3 , 4 , 3 , 5 , 6 }; int index = firstRepeatingElement (arr); if (index == -1) cout << "No repeating found!" << endl ; else cout << "First repeating is " << arr [index] << endl ; return 0 ; } C #include <stdio.h> #include <limits.h> #define MAX 100001 // Assuming values in arr[i] are in range 0 to 100000 int firstRepeatingElement (int arr [], int n) { int seen [MAX] = { 0 }; // Initialize all to 0 int minIndex = INT_MAX ; for (int i = n - 1 ; i >= 0 ; i --) { if (seen [arr [i]] == 1) { if (i < minIndex) minIndex = i ; } seen [arr [i]] = 1 ; } return (minIndex == INT_MAX) ? -1 : minIndex ; } int main () { int arr [] = { 10 , 5 , 3 , 4 , 3 , 5 , 6 }; int n = sizeof (arr) / sizeof (arr [0]); int index = firstRepeatingElement (arr , n); if (index == -1) printf ("No repeating found! \n "); else printf ("First repeating is %d \n " , arr [index]); return 0 ; } Java // Java program to find first repeating element // using Hash Set in O(n) Time and O(n) Space import java.util.* ; class GfG { static int firstRepeatingElement (int [] arr) { HashSet < Integer > s = new HashSet <> (); // If an element is already present, return it // else insert it int minEle = Integer . MAX_VALUE ; for (int i = arr . length - 1 ; i >= 0 ; i --) { if (s . contains (arr [i])) { minEle = Math . min (minEle , i); } s . add (arr [i]); } // If no element repeats return minEle == Integer . MAX_VALUE ? - 1 : minEle ; } public static void main (String [] args) { int [] arr = { 10 , 5 , 3 , 4 , 3 , 5 , 6 }; int index = firstRepeatingElement (arr); if (index == - 1) System . out . println ("No repeating found!"); else System . out . println ("First repeating is " + arr [index]); } } Python # Python program to find first repeating element # using Hash Set in O(n) Time and O(n) Space import sys def firstRepeatingElement (arr): s = set () # If an element is already present, return it # else insert it minEle = sys . maxsize for i in range (len (arr) - 1 , - 1 , - 1): if arr [i] in s : minEle = min (minEle , i) s . add (arr [i]) # If no element repeats return - 1 if minEle == sys . maxsize else minEle arr = [10 , 5 , 3 , 4 , 3 , 5 , 6] index = firstRepeatingElement (arr) if index == - 1 : print ("No repeating found!") else : print ("First repeating is" , arr [index]) C# // C# program to find first repeating element // using Hash Set in O(n) Time and O(n) Space using System ; using System.Collections.Generic ; class GfG { static int firstRepeatingElement (int [] arr) { HashSet < int > s = new HashSet < int > (); // If an element is already present, return it // else insert it int minEle = int . MaxValue ; for (int i = arr . Length - 1 ; i >= 0 ; i --) { if (s . Contains (arr [i])) { minEle = Math . Min (minEle , i); } s . Add (arr [i]); } // If no element repeats return minEle == int . MaxValue ? - 1 : minEle ; } static void Main () { int [] arr = { 10 , 5 , 3 , 4 , 3 , 5 , 6 }; int index = firstRepeatingElement (arr); if (index == - 1) Console . WriteLine ("No repeating found!"); else Console . WriteLine ("First repeating is " + arr [index]); } } JavaScript // JavaScript program to find first repeating element // using Hash Set in O(n) Time and O(n) Space function firstRepeatingElement (arr) { let s = new Set (); // If an element is already present, return it // else insert it let minEle = Number . MAX_SAFE_INTEGER ; for (let i = arr

. length - 1 ; i >= 0 ; i --) { if (s . has (arr [i])) { minEle = Math . min (minEle , i); } s . add (arr [i]); } // If no element repeats return minEle === Number . MAX_SAFE_INTEGER ? - 1 : minEle ; } let arr = [10 , 5 , 3 , 4 , 3 , 5 , 6]; let index = firstRepeatingElement (arr); if (index === - 1) console . log ("No repeating found!"); else console . log ("First repeating is " + arr [index]); Output First repeating is 5 Comment Article Tags: Article Tags: Searching Hash DSA Arrays Amazon Oracle + 2 More