Find the Missing Number - GeeksforGeeks

Source: https://www.geeksforgeeks.org/find-the-missing-number/

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 Find the Missing Number Last Updated : 19 Apr, 2025 Given an array arr[] of size n-1 with distinct integers in the range of [1, n]. This array represents a permutation of the integers from 1 to n with one element missing. Find the missing element in the array. Examples: Input: arr[] = [8, 2, 4, 5, 3, 7, 1] Output: 6 Explanation: All the numbers from 1 to 8 are present except 6. Input: arr[] = [1, 2, 3, 5] Output: 4 Explanation: Here the size of the array is 4, so the range will be [1, 5]. The missing number between 1 to 5 is 4 Try it on GfG Practice Table of Content [Naive Approach] Linear Search for Missing Number - O(n^2) Time and O(1) Space [Better Approach] Using Hashing - O(n) Time and O(n) Space [Expected Approach 1] Using Sum of n terms Formula - O(n) Time and O(1) Space [Expected Approach 2] Using XOR Operation - O(n) Time and O(1) Space [Naive Approach] Linear Search for Missing Number - O(n^2) Time and O(1) Space This approach iterates through each number from 1 to n (where n is the size of the array + 1) and checks if the number is present in the array. For each number, it uses a nested loop to search the array. If a number is not found, it is returned as the missing number. C++ #include <iostream> #include <vector> using namespace std ; int missingNum (vector < int >& arr) { int n = arr . size () + 1 ; // Iterate from 1 to n and check // if the current number is present for (int i = 1 ; i <= n ; i ++) { bool found = false ; for (int j = 0 ; j < n - 1 ; j ++) { if (arr [j] == i) { found = true ; break ; } } // If the current number is not present if (! found) return i ; } return -1 ; } int main () { vector < int > arr = { 8 , 2 , 4 , 5 , 3 , 7 , 1 }; cout << missingNum (arr) << endl ; return 0 ; } Java public class GfG { public static int missingNum (int [] arr) { int n = arr . length + 1 ; // Iterate from 1 to n and check // if the current number is present for (int i = 1 ; i <= n ; i ++) { boolean found = false ; for (int j = 0 ; j < n - 1 ; j ++) { if (arr [j] == i) { found = true ; break ; } } // If the current number is not present if (! found) return i ; } return - 1 ; } public static void main (String [] args) { int [] arr = { 8 , 2 , 4 , 5 , 3 , 7 , 1 }; System . out . println (missingNum (arr)); } } Python def missingNum (arr): n = len (arr) + 1 # Iterate from 1 to n and check # if the current number is present for i in range (1 , n + 1): found = False for j in range (n - 1): if arr [j] == i : found = True break # If the current number is not present if not found : return i return - 1 if __name__ == '__main__' : arr = [8 , 2 , 4 , 5 , 3 , 7 , 1] print (missingNum (arr)) C# using System ; class GfG { static int missingNum (int [] arr) { int n = arr . Length + 1 ; // Iterate from 1 to n and check // if the current number is present for (int i = 1 ; i <= n ; i ++) { bool found = false ; for (int j = 0 ; j < n - 1 ; j ++) { if (arr [j] == i) { found = true ; break ; } } // If the current number is not present if (! found) return i ; } return - 1 ; } static void Main () { int [] arr = { 8 , 2 , 4 , 5 , 3 , 7 , 1 }; Console . WriteLine (missingNum (arr)); } } JavaScript function missingNum (arr) { const n = arr . length + 1 ; // Iterate from 1 to n and check // if the current number is present for (let i = 1 ; i <= n ; i ++) { let found = false ; for (let j = 0 ; j < n - 1 ; j ++) { if (arr [j] === i) { found = true ; break ; } } // If the current number is not present if (! found) return i ; } return - 1 ; } // drvier code const arr = [8 , 2 , 4 , 5 , 3 , 7 , 1]; console . log (missingNum (arr)); Output 6 [Better Approach] Using Hashing - O(n) Time and O(n) Space This approach uses a hash array (or frequency array) to track the presence of each number from 1 to n in the input array. It first initializes a hash array to store the frequency of each element. Then, it iterates through the hash array to find the number that is missing (i.e., the one with a frequency of 0). C++ #include <iostream> #include <vector> using namespace std ; int missingNum (vector < int > & arr) { int n = arr . size () + 1 ; // Create hash array of size n+1 vector < int > hash (n + 1 , 0); // Store frequencies of elements for (int i = 0 ; i < n - 1 ; i ++) { hash [arr [i]] ++ ; } // Find the missing number for (int i = 1 ; i <= n ; i ++) { if (hash [i] == 0) { return i ; } } return -1 ; } int main () { vector < int > arr = {

8 , 2 , 4 , 5 , 3 , 7 , 1 }; int res = missingNum (arr); cout << res << endl ; return 0 ; } Java import java.util.Arrays ; public class GfG { public static int missingNum (int [] arr) { int n = arr . length + 1 ; // Create hash array of size n+1 int [] hash = new int [n + 1] ; // Store frequencies of elements for (int i = 0 ; i < n - 1 ; i ++) { hash [arr [i]]++ ; } // Find the missing number for (int i = 1 ; i <= n ; i ++) { if (hash [i] == 0) { return i ; } } return - 1 ; } public static void main (String [] args) { int [] arr = { 8 , 2 , 4 , 5 , 3 , 7 , 1 }; int res = missingNum (arr); System . out . println (res); } } Python def missingNum (arr): n = len (arr) + 1 # Create hash array of size n+1 hash = [0] * (n + 1) # Store frequencies of elements for i in range (n - 1): hash [arr [i]] += 1 # Find the missing number for i in range (1 , n + 1): if hash [i] == 0 : return i return - 1 if __name__ == '__main__' : arr = [8 , 2 , 4 , 5 , 3 , 7 , 1] res = missingNum (arr) print (res) C# using System ; class GfG { public static int missingNum (int [] arr) { int n = arr . Length + 1 ; // Create hash array of size n+1 int [] hash = new int [n + 1]; // Store frequencies of elements for (int i = 0 ; i < n - 1 ; i ++) { hash [arr [i]] ++ ; } // Find the missing number for (int i = 1 ; i <= n ; i ++) { if (hash [i] == 0) { return i ; } } return - 1 ; } static void Main () { int [] arr = { 8 , 2 , 4 , 5 , 3 , 7 , 1 }; int res = missingNum (arr); Console . WriteLine (res); } } JavaScript function missingNum (arr) { let n = arr . length + 1 ; // Create hash array of size n+1 let hash = new Array (n + 1). fill (0); // Store frequencies of elements for (let i = 0 ; i < n - 1 ; i ++) { hash [arr [i]] ++ ; } // Find the missing number for (let i = 1 ; i <= n ; i ++) { if (hash [i] === 0) { return i ; } } return - 1 ; } // driver code const arr = [8 , 2 , 4 , 5 , 3 , 7 , 1]; const res = missingNum (arr); console . log (res); Output 6 [Expected Approach 1] Using Sum of n terms Formula - O(n) Time and O(1) Space The sum of the first n natural numbers is given by the formula (n * (n + 1)) / 2 . The idea is to compute this sum and subtract the sum of all elements in the array from it to get the missing number. C++ #include <iostream> #include <vector> using namespace std ; int missingNum (vector < int > & arr) { int n = arr . size () + 1 ; // Calculate the sum of array elements int sum = 0 ; for (int i = 0 ; i < n - 1 ; i ++) { sum += arr [i]; } // Calculate the expected sum long long expSum = (n * 1L L * (n + 1)) / 2 ; // Return the missing number return expSum - sum ; } int main () { vector < int > arr = { 8 , 2 , 4 , 5 , 3 , 7 , 1 }; cout << missingNum (arr); return 0 ; } Java import java.util.* ; public class GfG { public static int missingNum (int [] arr) { long n = arr . length + 1 ; // Calculate the sum of array elements long sum = 0 ; for (int i = 0 ; i < arr . length ; i ++) { sum += arr [i] ; } // Use long for expected sum to avoid overflow long expSum = n * (n + 1) / 2 ; // Return the missing number return (int)(expSum - sum); } public static void main (String [] args) { int [] arr = { 8 , 2 , 4 , 5 , 3 , 7 , 1 }; System . out . println (missingNum (arr)); } } Python def missingNum (arr): n = len (arr) + 1 # Calculate the sum of array elements totalSum = sum (arr) # Calculate the expected sum expSum = n * (n + 1) // 2 # Return the missing number return expSum - totalSum if __name__ == '__main__' : arr = [8 , 2 , 4 , 5 , 3 , 7 , 1] print (missingNum (arr)) C# using System ; using System.Linq ; class GfG { static int missingNum (int [] arr) { int n = arr . Length + 1 ; // Calculate the sum of array elements int sum = arr . Sum (); // Calculate the expected sum using long to avoid overflow long expSum = (long) n * (n + 1) / 2 ; // Return the missing number return (int)(expSum - sum); } static void Main () { int [] arr = { 8 , 2 , 4 , 5 , 3 , 7 , 1 }; Console . WriteLine (missingNum (arr)); } } JavaScript function missingNum (arr) { let n = arr . length + 1 ; // Calculate the sum of array elements let sum = 0 ; for (let i = 0 ; i < n - 1 ; i ++) { sum += arr [i]; } // Calculate the expected sum let expSum = (n * (n + 1)) / 2 ; // Return the missing number return expSum - sum ; } // driver code let arr = [8 , 2 , 4 , 5 , 3 , 7 , 1]; console . log (missingNum (arr)); Output 6 [Expected Approach 2] Using XOR Operation - O(n) Time and O(1) Space XOR of a number with itself is 0 i.e. x ^ x = 0 and the given array arr[] has numbers in range [1, n] . This means that the result of XOR of first n natural numbers with the XOR of all the array elements will be the missing number. To do so, calculate XOR of first n natural numbers and XOR of all the array arr[] elements, and then our result will be the XOR of both the resultant values. C++ #include <iostream> #include <vector> using namespace std ; int missingNum (vector < int >& arr) { int n = arr . size () + 1 ; int xor1 = 0 , xor2 = 0 ; // XOR all array elements for (int i = 0 ; i < n - 1 ; i ++) { xor2 ^= arr [i]; } // XOR all numbers from 1 to n for (int i = 1 ; i <= n ; i ++) { xor1 ^= i ; } // Missing number is the XOR of xor1 and xor2 return xor1 ^ xor2 ; } int main () { vector < int > arr = { 8 , 2 , 4 , 5 , 3 , 7 , 1 }; int res = missingNum (arr); cout << res << endl ; return 0 ; } Java import java.util.Arrays ; public class GfG { public static int missingNum (int [] arr) { int n = arr . length + 1 ; int xor1 = 0 , xor2 = 0 ; // XOR all array elements for (int i = 0 ; i < n - 1 ; i ++) { xor2 ^= arr [i] ; } // XOR all numbers from 1 to n for (int i = 1 ; i <= n ; i ++) { xor1 ^= i ; } // Missing number is the XOR of xor1 and xor2 return xor1 ^ xor2 ; } public static void main (String [] args) { int [] arr = { 8 , 2 , 4 , 5 , 3 , 7 , 1 }; int res = missingNum (arr); System . out . println (res); } } Python def missingNum (arr): n = len (arr) + 1 xor1 = 0 xor2 = 0 # XOR all array elements for i in range (n - 1): xor2 ^= arr [i] # XOR all numbers from 1 to n for i in range (1 , n + 1): xor1 ^= i # Missing number is the XOR of xor1 and xor2 return xor1

^ xor2 if __name__ == '__main__' : arr = [8 , 2 , 4 , 5 , 3 , 7 , 1] res = missingNum (arr) print (res) C# using System ; class GfG { static int missingNum (int [] arr) { int n = arr . Length + 1 ; int xor1 = 0 , xor2 = 0 ; // XOR all array elements for (int i = 0 ; i < n - 1 ; i ++) { xor2 ^= arr [i]; } // XOR all numbers from 1 to n for (int i = 1 ; i <= n ; i ++) { xor1 ^= i ; } // Missing number is the XOR of xor1 and xor2 return xor1 ^ xor2 ; } static void Main () { int [] arr = { 8 , 2 , 4 , 5 , 3 , 7 , 1 }; int res = missingNum (arr); Console . WriteLine (res); } } JavaScript function missingNum (arr) { const n = arr . length + 1 ; let xor1 = 0 , xor2 = 0 ; // XOR all array elements for (let i = 0 ; i < n - 1 ; i ++) { xor2 ^= arr [i]; } // XOR all numbers from 1 to n for (let i = 1 ; i <= n ; i ++) { xor1 ^= i ; } // Missing number is the XOR of xor1 and xor2 return xor1 ^ xor2 ; } // driver code const arr = [8 , 2 , 4 , 5 , 3 , 7 , 1]; const res = missingNum (arr); console . log (res); Output 6 Comment Article Tags: Article Tags: Searching DSA Arrays Microsoft Amazon Morgan Stanley Qualcomm Samsung Cisco Accolite Payu Visa Ola Cabs Bitwise-XOR limited-range-elements + 11 More