Linear Search Algorithm - GeeksforGeeks

Source: https://www.geeksforgeeks.org/linear-search/

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 Linear Search Algorithm Last Updated : 22 Oct, 2025 Given an array, arr[] of n integers, and an integer element x , find whether element x is present in the array. Return the index of the first occurrence of x in the array, or -1 if it doesn't exist. Input : arr[] = [1, 2, 3, 4], x = 3 Output : 2 Explanation : There is one test case with array as [1, 2, 3 4] and element to be searched as 3. Since 3 is present at index 2, the output is 2. Input : arr[] = [10, 8, 30, 4, 5], x = 5 Output : 4 Explanation : For array [10, 8, 30, 4, 5], the element to be searched is 5 and it is at index 4. So, the output is 4. Input : arr[] = [10, 8, 30], x = 6 Output : -1 Explanation : The element to be searched is 6 and its not present, so we return -1. Try it on GfG Practice In Linear Search, we iterate over all the elements of the array and check if it the current element is equal to the target element. If we find any element to be equal to the target element, then return the index of the current element. Otherwise, if no element is equal to the target element, then return -1 as the element is not found. Linear search is also known as sequential search . Table of Content Time and Space Complexity of Linear Search Algorithm: Applications of Linear Search Algorithm: Advantages of Linear Search Algorithm: Disadvantages of Linear Search Algorithm: When to use Linear Search Algorithm? For example: Consider the array arr[] = {10, 50, 30, 70, 80, 20, 90, 40} and key = 30 Implementation: C++ #include <iostream> #include <vector> using namespace std ; int search (vector < int >& arr , int x) { // Iterate over the array in order to // find the key x for (int i = 0 ; i < arr . size (); i ++) if (arr [i] == x) return i ; return -1 ; } int main () { vector < int > arr = { 2 , 3 , 4 , 10 , 40 }; int x = 10 ; int res = search (arr , x); if (res == -1) cout << "Element is not present in the array" ; else cout << "Element is present at index " << res ; return 0 ; } C #include <stdio.h> int search (int arr [], int n , int x) { // Iterate over the array in order to // find the key x for (int i = 0 ; i < n ; i ++) if (arr [i] == x) return i ; return -1 ; } // Driver code int main (void) { int arr [] = { 2 , 3 , 4 , 10 , 40 }; int x = 10 ; int n = sizeof (arr) / sizeof (arr [0]); // Function call int result = search (arr , n , x); (result == -1) ? printf ("Element is not present in array") : printf ("Element is present at index %d" , result); return 0 ; } Java import java.io.* ; class GFG { public static int search (int arr [] , int N , int x) { // Iterate over the array in order to // find the key x for (int i = 0 ; i < N ; i ++) { if (arr [i] == x) return i ; } return - 1 ; } public static void main (String args []) { int arr [] = { 2 , 3 , 4 , 10 , 40 }; int x = 10 ; int result = search (arr , arr . length , x); if (result == - 1) System . out . print ("Element is not present in array"); else System . out . print ("Element is present at index " + result); } } Python def search (arr , x): n = len (arr) # Iterate over the array in order to # find the key x for i in range (0 , n): if (arr [i] == x): return i return - 1 if __name__ == "__main__" : arr = [2 , 3 , 4 , 10 , 40] x = 10 result = search (arr , x) if (result == - 1): print ("Element is not present in array") else : print ("Element is present at index" , result) C# using System ; class GFG { public static int search (int [] arr , int x) { int n = arr . Length ; // Iterate over the array in order to // find the key x for (int i = 0 ; i < n ; i ++) { if (arr [i] == x) return i ; } return - 1 ; } public static void Main () { int [] arr = { 2 , 3 , 4 , 10 , 40 }; int x = 10 ; int result = search (arr , x); if (result == - 1) Console . WriteLine ("Element is not present in array"); else Console . WriteLine ("Element is present at index " + result); } } JavaScript function search (arr , x) { const n = arr . length ; // Iterate over the array in order to // find the key x for (let i = 0 ; i < n ; i ++) if (arr [i] == x) return i ; return - 1 ; } // Driver Code let arr = [2 , 3 , 4 , 10 , 40]; let x = 10 ; let result = search (arr , x); (result == - 1) ? console . log ("Element is not present in array") : console . log ("Element is present at index " + result); PHP <?php function search ($arr , $x) { $n = sizeof ($arr); // Iterate over the array in order to // find the key x for ($i = 0 ; $i < $n ; $i ++) { if ($arr [$i] == $x) return $i ; } return - 1 ; } $arr = array

(2 , 3 , 4 , 10 , 40); $x = 10 ; $result = search ($arr , $x); if ($result == - 1) echo "Element is not present in array" ; else echo "Element is present at index " , $result ; ?> Output Present at Index 3 Time and Space Complexity of Linear Search Algorithm: Time Complexity: Best Case: In the best case, the key might be present at the first index. So the best case complexity is O(1) Worst Case: In the worst case, the key might be present at the last index i.e., opposite to the end from which the search has started in the list. So the worst-case complexity is O(N) where N is the size of the list. Average Case: O(N) Auxiliary Space: O(1) as except for the variable to iterate through the list, no other variable is used. Applications of Linear Search Algorithm: Unsorted Lists: When we have an unsorted array or list, linear search is most commonly used to find any element in the collection. Small Data Sets: Linear Search is preferred over binary search when we have small data sets with Searching Linked Lists: In linked list implementations, linear search is commonly used to find elements within the list. Each node is checked sequentially until the desired element is found. Simple Implementation: Linear Search is much easier to understand and implement as compared to Binary Search or Ternary Search. Advantages of Linear Search Algorithm : Linear search can be used irrespective of whether the array is sorted or not. It can be used on arrays of any data type. Does not require any additional memory. It is a well-suited algorithm for small datasets. Disadvantages of Linear Search Algorithm : Linear search has a time complexity of O(N), which in turn makes it slow for large datasets. Not suitable for large arrays. When to use Linear Search Algorithm ? When we are dealing with a small dataset. When you are searching for a dataset stored in contiguous memory. Comment Article Tags: Article Tags: Searching DSA Arrays school-programming CBSE - Class 11 + 1 More