Longest Common Prefix using Trie - GeeksforGeeks

Source: https://www.geeksforgeeks.org/longest-common-prefix-using-trie/

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 Longest Common Prefix using Trie Last Updated : 23 Jul, 2025 Given an array of strings arr[], the task is to return the longest common prefix among each and every strings present in the array. If there’s no prefix common in all the strings, return “”. Examples: Input: arr[] = [“geeksforgeeks”, “geeks”, “geek”, “geezer”] Output: “ gee” Explanation: “ gee” is the longest common prefix in all the given strings: “ gee ksforgeeks”, “ gee ks”, “ gee ks” and “ gee zer”. Input : arr[] = [“apple”, “ape”, “april”] Output : “ap” Explanation: “ap” is the longest common prefix in all the given strings: “ ap ple”, “ ap e” and “ ap ril”. Input: arr[] = [“hello”, “world”] Output: “” Explanation: There’s no common prefix in the given strings. Approach: The idea is to insert all the string one by one in the trie . After inserting we perform a walk on the trie. In this walk, we go deeper until we find a node having more than 1 children(branching occurs) or 0 children (one of the string gets exhausted). This is because the characters (nodes in trie) which are present in the longest common prefix must be the single child of its parent, i.e- there should not be branching in any of these nodes. C++ // C++ Program to find the Longest Common Prefix // of the given strings using Trie #include <iostream> #include <vector> using namespace std ; class TrieNode { public : vector < TrieNode *> children ; int childCount ; // isLeaf is true if the node represents // end of a word bool isLeaf ; TrieNode () { children = vector < TrieNode *> (26 , nullptr); childCount = 0 ; isLeaf = false ; } }; // If not present, inserts the key into the trie // If the key is a prefix of trie node, just mark leaf node void insert (TrieNode * root , string & key) { TrieNode * curr = root ; for (char ch : key) { int idx = ch - 'a' ; if (curr -> children [idx] == nullptr) { curr -> children [idx] = new TrieNode (); curr -> childCount ++ ; } curr = curr -> children [idx]; } // mark last node as leaf curr -> isLeaf = true ; } // Perform a walk on the trie and return the // longest common prefix string string walkTrie (TrieNode * root , string & s) { TrieNode * curr = root ; int i = 0 ; while (curr -> childCount == 1 && ! curr -> isLeaf) { int idx = s [i] - 'a' ; i ++ ; curr = curr -> children [idx]; } return s . substr (0 , i); } // A Function that returns the longest common prefix // from the array of strings string longestCommonPrefix (vector < string >& arr) { TrieNode * root = new TrieNode (); // Insert all strings to the trie for (string & s : arr) insert (root , s); // Perform a walk on the trie return walkTrie (root , arr [0]); } int main () { vector < string > arr = { "geeksforgeeks" , "geeks" , "geek" , "geezer" }; cout << longestCommonPrefix (arr) << endl ; return 0 ; } Java // Java Program to find the Longest Common Prefix // of the given strings using Trie import java.util.* ; class TrieNode { public List < TrieNode > children ; public int childCount ; public boolean isLeaf ; public TrieNode () { children = new ArrayList <> (26); for (int i = 0 ; i < 26 ; i ++) { children . add (null); } childCount = 0 ; isLeaf = false ; } } class GfG { // If not present, inserts the key into the trie // If the key is a prefix of trie node, just mark leaf node static void insert (TrieNode root , String key) { TrieNode curr = root ; for (char ch : key . toCharArray ()) { int idx = ch - 'a' ; if (curr . children . get (idx) == null) { curr . children . set (idx , new TrieNode ()); curr . childCount ++ ; } curr = curr . children . get (idx); } // mark last node as leaf curr . isLeaf = true ; } // Perform a walk on the trie and return the // longest common prefix string static String walkTrie (TrieNode root , String s) { TrieNode curr = root ; int i = 0 ; while (curr . childCount == 1 && ! curr . isLeaf) { int idx = s . charAt (i) - 'a' ; i ++ ; curr = curr . children . get (idx); } return s . substring (0 , i); } // A Function that returns the longest common prefix // from the array of strings static String longestCommonPrefix (String [] arr) { TrieNode root = new TrieNode (); // Insert all strings to the trie for (String s : arr) insert (root , s); // Perform a walk on the trie return walkTrie (root , arr [0]); } public static void main (String [] args) { String [] arr = { "geeksforgeeks" , "geeks" , "geek" , "geezer" }; System . out . println (

longestCommonPrefix (arr)); } } Python # Python Program to find the Longest Common Prefix # of the given strings using Trie class TrieNode : def __init__ (self): self . children = [None] * 26 self . childCount = 0 self . isLeaf = False # If not present, inserts the key into the trie # If the key is a prefix of trie node, just mark leaf node def insert (root , key): curr = root for ch in key : idx = ord (ch) - ord ('a') if curr . children [idx] is None : curr . children [idx] = TrieNode () curr . childCount += 1 curr = curr . children [idx] # mark last node as leaf curr . isLeaf = True # Perform a walk on the trie and return the # longest common prefix string def walkTrie (root , s): curr = root i = 0 while curr . childCount == 1 and not curr . isLeaf : idx = ord (s [i]) - ord ('a') i += 1 curr = curr . children [idx] return s [: i] # A Function that returns the longest common prefix # from the array of strings def longestCommonPrefix (arr): root = TrieNode () # Insert all strings to the trie for s in arr : insert (root , s) # Perform a walk on the trie return walkTrie (root , arr [0]) if __name__ == "__main__" : arr = ["geeksforgeeks" , "geeks" , "geek" , "geezer"] print (longestCommonPrefix (arr)) C# // C# Program to find the Longest Common Prefix // of the given strings using Trie using System ; using System.Collections.Generic ; class TrieNode { public List < TrieNode > children ; public int childCount ; public bool isLeaf ; public TrieNode () { children = new List < TrieNode > (26); for (int i = 0 ; i < 26 ; i ++) { children . Add (null); } childCount = 0 ; isLeaf = false ; } } class GfG { // If not present, inserts the key into the trie // If the key is a prefix of trie node, just mark leaf node static void insert (TrieNode root , string key) { TrieNode curr = root ; foreach (char ch in key) { int idx = ch - 'a' ; if (curr . children [idx] == null) { curr . children [idx] = new TrieNode (); curr . childCount ++ ; } curr = curr . children [idx]; } // mark last node as leaf curr . isLeaf = true ; } // Perform a walk on the trie and return the // longest common prefix string static string walkTrie (TrieNode root , string s) { TrieNode curr = root ; int i = 0 ; while (curr . childCount == 1 && ! curr . isLeaf) { int idx = s [i] - 'a' ; i ++ ; curr = curr . children [idx]; } return s . Substring (0 , i); } // A Function that returns the longest common prefix // from the array of strings static string longestCommonPrefix (string [] arr) { TrieNode root = new TrieNode (); // Insert all strings to the trie foreach (string s in arr) { insert (root , s); } // Perform a walk on the trie return walkTrie (root , arr [0]); } static void Main (string [] args) { string [] arr = { "geeksforgeeks" , "geeks" , "geek" , "geezer" }; Console . WriteLine (longestCommonPrefix (arr)); } } JavaScript // JavaScript Program to find the Longest Common Prefix // of the given strings using Trie class TrieNode { constructor () { this . children = new Array (26). fill (null); this . childCount = 0 ; this . isLeaf = false ; } } // If not present, inserts the key into the trie // If the key is a prefix of trie node, just mark leaf node function insert (root , key) { let curr = root ; for (let ch of key) { let idx = ch . charCodeAt (0) - 'a' . charCodeAt (0); if (curr . children [idx] === null) { curr . children [idx] = new TrieNode (); curr . childCount ++ ; } curr = curr . children [idx]; } // mark last node as leaf curr . isLeaf = true ; } // Perform a walk on the trie and return the // longest common prefix string function walkTrie (root , s) { let curr = root ; let i = 0 ; while (curr . childCount === 1 && ! curr . isLeaf) { let idx = s . charCodeAt (i) - 'a' . charCodeAt (0); i ++ ; curr = curr . children [idx]; } return s . substring (0 , i); } // A Function that returns the longest common prefix // from the array of strings function longestCommonPrefix (arr) { let root = new TrieNode (); // Insert all strings to the trie for (let s of arr) { insert (root , s); } // Perform a walk on the trie return walkTrie (root , arr [0]); } // Driver Code const arr = ["geeksforgeeks" , "geeks" , "geek" , "geezer"]; console . log (longestCommonPrefix (arr)); Output gee Time Complexity: O(n*m), where n is the number of strings and m is the length of the largest string. Auxiliary Space: O(n*m) , to store all the strings in Trie. Other Approaches Longest Common Prefix using Sorting Longest Common Prefix Word by Word Matching Longest Common Prefix Character by Character Matching Longest Common Prefix Divide and Conquer Longest Common Prefix Binary Search Comment Article Tags: Article Tags: Strings Advanced Data Structure DSA Arrays Trie VMWare Longest Common Prefix + 3 More