



Leetcode: 14 Longest Common Prefix



BHONE



14. Longest Common Prefix

Easy

Topics

Companies

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string `""`.



Example 1:

Input: strs = ["flower", "flow", "flight"]
Output: "fl"

Example 2:

Input: strs = ["dog", "racecar", "car"]
Output: ""
Explanation: There is no common prefix among the input strings.



Constraints:

- `1 <= strs.length <= 200`
- `0 <= strs[i].length <= 200`
- `strs[i]` consists of only lowercase English letters if it is non-empty.



Constraints:

- $1 \leq \text{nums.length} \leq 3 * 10^4$
- $-100 \leq \text{nums}[i] \leq 100$
- `nums` is sorted in **non-decreasing** order.



Explanation





longest common prefix

flower, flow, flight

flower, flow, flight

shortest = flow

shortest = fl



Code





```
class Solution:
    def longestCommonPrefix(self, strs: List[str]) -> str:
        shortest = min(strs, key=len)
        shortestLength = len(shortest)

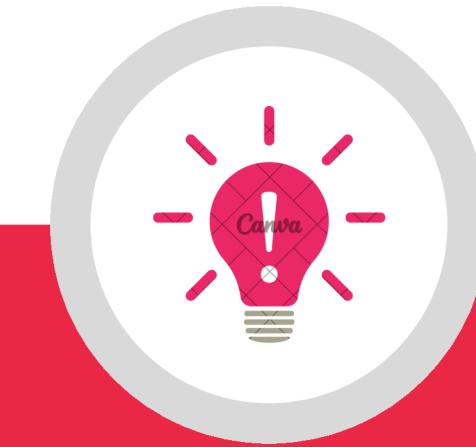
        for i in strs:
            for j in range(shortestLength):
                if i[j] == shortest[j]:
                    continue
                else:
                    shortest = i[:j]
                    shortestLength = len(shortest)
                    break
        return shortest
```



</> Code

Python3 ▾ 🔒 Auto

```
1 class Solution:
2     def longestCommonPrefix(self, strs: List[str]) -> str:
3         if not strs:
4             return ""
5
6         # Compare character by character across all strings
7         for i in range(len(strs[0])):
8             char = strs[0][i]
9             for j in range(1, len(strs)):
10                 # Check if we've reached end of current string or found mismatch
11                 if i >= len(strs[j]) or strs[j][i] != char:
12                     return strs[0][:i]
13
14         return strs[0]
15
16
```



Time and Space Complexity Analysis





```
class Solution:  
    def longestCommonPrefix(self, strs: List[str]) -> str:  
        shortest = min(strs, key=len)      Space - O(m)  
        shortestLength = len(shortest)  
  
        for i in strs:  
            for j in range(shortestLength):  
                if i[j] == shortest[j]:  
                    continue  
                else:  
                    shortest = i[:j]  
                    shortestLength = len(shortest)  
                    break  
        return shortest
```



Time - $O(nxm)$