Click on the logo to find Problem Statement

Intuition

1. **Initialization:**
    - Start by assuming that the first string in the list is the longest common prefix. This assumption is based on the fact that there is no information available yet to limit the prefix.
2. **Iterative Comparison:**
    - Iterate through the rest of the strings in the list. For each string, compare its characters with the characters of the assumed common prefix.
3. **Character Comparison:**
    - Compare characters of the assumed common prefix and the current string until a mismatch is found or until the end of either string is reached.
    - If a mismatch is found or if the index exceeds the length of the current string, update the assumed common prefix to be the substring up to the current index. This is because any characters beyond this index cannot be part of the common prefix.
4. **Finding the Longest Common Prefix:**
    - By iteratively comparing characters and shortening the assumed common prefix, the algorithm effectively finds the longest common prefix among all strings.
    - If at any point the assumed common prefix becomes empty, it means there is no common prefix among the strings, and the algorithm can stop.
5. **Final Output:**
    - The final assumed common prefix after iterating through all strings is the longest common prefix. Return this prefix as the result.

Approach

1. Edge Case Check:
    - If the input list strs is empty, return an empty string as there is no common prefix.
2. Initialize Prefix:
    - Initialize a variable prefix with the first string in the list strs[0].This string will serve as the initial assumed common prefix.
3. Iterate through Strings:
    - Iterate through the remaining strings in the list (from strs[1] onwards).
4. Find Common Prefix:
    - For each string in the list, iterate through the characters of the assumed common prefix and the current string.
    - Compare characters at the same position.
    - If a mismatch is found or if the index exceeds the length of the current string, update the prefix variable to be the substring up to the current index and break out of the loop. This is because any characters beyond this index cannot be part of the common prefix.
5. Return Result:

- After iterating through all strings, the prefix variable will contain the longest common prefix. Return prefix as the result.

Complexity

- Time complexity: O(n·m·log(m))
- Space complexity: O(n·m)

Code

```
class Solution:
    def longestCommonPrefix(self, strs: List[str]) -> str:
        ans = ""
        strs = sorted(strs)
        start = strs[0]
        end = strs[-1]
        for i in range(min(len(start), len(end))):
            if start[i] != end[i]:
```

```
            return ans
        ans += start[i]
    return ans
```