[Click on the logo to find **Problem Statement**]

Intuition

Sorting the Strings: Sorting the list of strings lexicographically ensures that any common prefix shared by multiple strings will be grouped together. After sorting, strings with similar prefixes will be adjacent in the sorted list.

Comparing Characters: The code then compares the characters of the first string (start) and the last string (end) after sorting. By iterating through these characters, it checks if they are the same in both strings.

Finding the Common Prefix: As long as the characters at the current position i in start and end are the same, it adds that character to the ans variable, which represents the common prefix found so far.

Early Exit on Mismatch: If the characters at position i are not the same, it means the common prefix ends at the previous position. The code returns the ans variable immediately, indicating that the common prefix among all strings is up to index i-1.

Approach

Edge Case Check: If the input list strs is empty, return an empty string as there is no common prefix.

Sorting: Sort the list of strings strs in lexicographical order. Sorting is done to bring the strings with the common prefix together, making it easier to find the common prefix.

Comparison: Take the first string after sorting (i.e., start) and the last string after sorting (i.e., end). Iterate through the characters of start and end simultaneously until you find a mismatch at any position i. If you find a mismatch, return the current ans as the longest common prefix found so far. If all characters match up to the length of the shorter string, append the character to the ans variable.

Return ans: After the loop, return ans as the longest common prefix among all strings in the input list.

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
```