Question 9: Longest Common Prefix Among Strings

Problem

Find the longest common prefix among a list of strings.

Algorithm 1: Horizontal Scanning

1. Take the first string as the initial prefix.
2. Iterate through the remaining strings: a. Compare the current prefix with each string. b. Update the prefix to the common part. c. If at any point the prefix becomes empty, return empty string.
3. Return the final prefix.

Program Implementation

cpp

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;

string longestCommonPrefix(vector<string>& strs) {
    if (strs.empty()) return "";

    string prefix = strs[0];

    for (int i = 1; i < strs.size(); i++) {
        int j = 0;
        while (j < prefix.length() && j < strs[i].length() && prefix[j] == strs[i][j]) {
            j++;
        }

        prefix = prefix.substr(0, j);
        if (prefix.empty()) return "";
    }

    return prefix;
}

int main() {
    vector<string> strs1 = {"flower", "flow", "flight"};
    cout << "Longest common prefix: " << longestCommonPrefix(strs1) << endl;

    vector<string> strs2 = {"dog", "racecar", "car"};
    cout << "Longest common prefix: " << longestCommonPrefix(strs2) << endl;

    vector<string> strs3 = {"apple", "app", "application"};
    cout << "Longest common prefix: " << longestCommonPrefix(strs3) << endl;

    return 0;
}
```

Algorithm 2: Vertical Scanning

1. Scan characters from left to right for all strings simultaneously.
2. For each position, check if the character is the same across all strings.
3. If not, or if we reached the end of any string, return the prefix found so far.

cpp

```cpp
string longestCommonPrefixVertical(vector<string>& strs) {
    if (strs.empty()) return "";

    for (int i = 0; i < strs[0].length(); i++) {
        char c = strs[0][i];
        for (int j = 1; j < strs.size(); j++) {
            if (i == strs[j].length() || strs[j][i] != c) {
                return strs[0].substr(0, i);
            }
        }
    }

    return strs[0];
}
```

Algorithm 3: Divide and Conquer

1. Divide the array of strings into two halves.
2. Find the longest common prefix of each half.
3. Merge by finding the common prefix of the two results.
4. Base case: If there's only one string, return it.

cpp

```cpp
string commonPrefix(const string& str1, const string& str2) {
    int minLength = min(str1.length(), str2.length());
    for (int i = 0; i < minLength; i++) {
        if (str1[i] != str2[i]) {
            return str1.substr(0, i);
        }
    }
    return str1.substr(0, minLength);
}

string longestCommonPrefixDC(vector<string>& strs, int start, int end) {
    if (start == end) {
        return strs[start];
    }

    int mid = (start + end) / 2;
    string leftPrefix = longestCommonPrefixDC(strs, start, mid);
    string rightPrefix = longestCommonPrefixDC(strs, mid + 1, end);

    return commonPrefix(leftPrefix, rightPrefix);
}

string longestCommonPrefixDivideConquer(vector<string>& strs) {
    if (strs.empty()) return "";
    return longestCommonPrefixDC(strs, 0, strs.size() - 1);
}
```

Time Complexity

- Horizontal Scanning: O(S) where S is the sum of all characters in the array
- Vertical Scanning: O(S) in the worst case, but can terminate early
- Divide and Conquer: O(S log n) where n is the number of strings

Space Complexity

- Horizontal and Vertical Scanning: O(1) extra space
- Divide and Conquer: O(m log n) where m is the length of the longest string

Example Explanation

Let's trace through the example ["flower", "flow", "flight"] using horizontal scanning:

1. Initial prefix = "flower"
2. Compare with "flow":

- Common characters: "flow"
- Update prefix to "flow"

3. Compare with "flight":
   - Common characters: "fl"
   - Update prefix to "fl"

The algorithm returns "fl" as the longest common prefix.