String Compression III

solved by	Senan
Platform	LeetCode
difficulty	Medium
# Serial	3163
tags	String Manipulation
language	C++
solved on	@04/11/2024
link	https://leetcode.com/problems/string-compression-iii/description/
Completion	☑

Intuition

The goal is to compress a given string by counting consecutive occurrences of each character and representing them as a digit followed by the character itself. This approach helps in minimizing the storage required for strings with repetitive sequences.

Approach

1. Initialize an empty vector `temp` to store pairs representing the count and character.
2. Push a dummy pair `{-1, '#'}` to `temp` to facilitate comparison.
3. Traverse each character in the input `word`:
    - If the current character is different from the last character in `temp`, push a new pair `{1, ch}` to `temp`.
    - If the count of the last character in `temp` reaches `9`, push a new pair `{1, ch}` to ensure digit limits.
    - Otherwise, increment the count of the last character in `temp`.
4. Compute the size of the new compressed string (`newSize`) and resize `answer` accordingly.
5. Populate `answer` by iterating over `temp` from the second element, appending counts as digits and characters.
6. Return the compressed string.

Complexity

Time Complexity:

- **O(n)**, where `n` is the length of the input string `word`. We traverse the string once to build the `temp` vector and once to construct the `answer`.

Space Complexity:

- **O(n)**, for the `temp` vector and the resulting compressed string `answer`.

Code

```
class Solution {
public:
    string compressedString(string word) {
        string answer;
        int count = 0;
        vector<pair<int, char>> temp;
        temp.push_back({-1, '#'});
        for (auto ch : word) {
            if (temp.back().second != ch)
                temp.push_back({1, ch});
            else if (temp.back().first == 9)
                temp.push_back({1, ch});
            else
                temp.back().first++;
        }
        int newSize = 2 * (temp.size() - 1);
        answer.resize(newSize);
        int j = 0;
        for (int i = 1; i < temp.size(); i++) {
            answer[j++] = '0' + temp[i].first;
            answer[j++] = temp[i].second;
        }
        return answer;
    }
};
```