443 String Compression

2018年4月11日 15:28

Given an array of characters, compress it **in-place**.

The length after compression must always be smaller than or equal to the original array.

Every element of the array should be a **character** (not int) of length 1.

After you are done **modifying the input array in-place**, return the new length of the array.

Follow up:

Could you solve it using only O(1) extra space?

Example 1:

Input:

["a","a","b","b","c","c","c"]

Output:

Return 6, and the first 6 characters of the input array should be: ["a","2","b","2","c","3"]

Explanation:

"aa" is replaced by "a2". "bb" is replaced by "b2". "ccc" is replaced by "c3".

Example 2:

Input:

["a"]

Output:

Return 1, and the first 1 characters of the input array should be: ["a"]

Explanation:

Nothing is replaced.

Example 3:

Input:

["a","b","b","b","b","b","b","b","b","b","b","b","b"]

Output:

Return 4, and the first 4 characters of the input array should be: ["a","b","1","2"].

Explanation:

Since the character "a" does not repeat, it is not compressed. "bbbbbbbbbbbb" is replaced by "b12".

Notice each digit has it's own entry in the array.

Note:

1. All characters have an ASCII value in [35, 126].
2. 1 <= len(chars) <= 1000.

来自 <https://leetcode.com/problems/string-compression/description/>

给定一组字符，使用原地算法将其压缩。

压缩后的长度必须始终小于或等于原数组长度。

数组的每个元素应该是长度为1 的**字符**（不是 int 整数类型）。

在完成原地**修改输入数组**后，返回数组的新长度。

进阶：

你能否仅使用O(1) 空间解决问题?

注意：

1. 所有字符都有一个ASCII值在[35, 126]区间内。
2. 1 <= len(chars) <= 1000。

Solution for Python3:

```python
class Solution:
    def compress(self, chars):
        """
        :type chars: List[str]
        :rtype: int
        """
        anchorindex, curindex = 0, 0
        while curindex < len(chars):
            curChar = chars[curindex]
            charNum = 0
            while curindex < len(chars) and chars[curindex] == curChar:
                charNum += 1
                curindex += 1
            chars[anchorindex] = curChar;
            if charNum > 1:
                chars[anchorindex+1:anchorindex+1+len(str(charNum))] = list(str(charNum))
                anchorindex += 1 + len(str(charNum))
            else:
                anchorindex += 1
        print(chars)
        return anchorindex


```

```cpp
class Solution {
public:
    int compress(vector<char>& chars) {
        int anchorindex = 0, curindex = 0;
        while (curindex < chars.size()) {
            char curChar = chars[curindex];
            int charNum = 0;
            while (curindex < chars.size() &&
```

```
                int charNum = 0;
4               while (curindex < chars.size() &&
    chars[curindex] == curChar) {
5                   charNum++;
                    curindex++;
6               }
                chars[anchorindex] = curChar;
7               if (charNum > 1) {
                    string s = to_string(charNum);
8                   for (int i = 1; i <= s.length();
    i++) {
9                       chars[anchorindex + i] = s[i -
    1];
10                  }
                    anchorindex += 1 + s.length();
11              } else {
                    anchorindex++;
12              }
            }
13          return anchorindex;
        }
14  };
15
16
17
18
19
20
21
22
23
```

24

25