443. String Compression

Oct. 27, 2017 | 89.8K views

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","
```

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. "bbbbbbbbbbb" is repla
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`.

Approach #1: Read and Write Heads [Accepted]

Intuition

We will use separate pointers `read` and `write` to mark where we are reading and writing from. Both operations will be done left to right alternately: we will read a contiguous group of characters, then write the compressed version to the array. At the end, the position of the `write` head will be the length of the answer that was written.

Algorithm

Let's maintain `anchor`, the start position of the contiguous group of characters we are currently reading.

Now, let's read from left to right. We know that we must be at the end of the block when we are at the last character, or when the next character is different from the current character.

When we are at the end of a group, we will write the result of that group down using our `write` head. `chars[anchor]` will be the correct character, and the length (if greater than 1) will be `read - anchor + 1`. We will write the digits of that number to the array.

Python

```
class Solution(object):
    def compress(self, chars):
        anchor = write = 0
        for read, c in enumerate(chars):
            if read + 1 == len(chars) or chars[read + 1] != c:
                chars[write] = chars[anchor]
                write += 1
                if read > anchor:
                    for digit in str(read - anchor + 1):
                        chars[write] = digit
                        write += 1
                anchor = read + 1
        return write
```

Java

```
class Solution {
    public int compress(char[] chars) {
        int anchor = 0, write = 0;
        for (int read = 0; read < chars.length; read++) {
            if (read + 1 == chars.length || chars[read + 1] != chars[read]) {
                chars[write++] = chars[anchor];
                if (read > anchor) {
                    for (char c: ("" + (read - anchor + 1)).toCharArray()) {
                        chars[write++] = c;
                    }
                }
                anchor = read + 1;
            }
        }
        return write;
    }
}
```

Complexity Analysis

- Time Complexity: $O(N)$, where N is the length of `chars`.
- Space Complexity: $O(1)$, the space used by `read`, `write`, and `anchor`.

Rate this article:

Previous Next

Comments: 37

Sort By

Type comment here... (Markdown is supported)

Preview Post

qliang ★ 108 September 10, 2019 3:33 AM

The description of this problem is so poorly written! It doesn't mention that the characters in output are also in order of group in input, and even duplicates are allowed! The given 3 examples are just not enough, at least should add this one as input ["a","a","b","b","a","a"] and ["a","2","b","2","a","3"] as output.

Read More

103 Share Reply

SHOW 6 REPLIES

canadianczar ★ 175 August 11, 2019 11:17 PM

IMHO this question should be medium difficulty because of the trickier edge cases.

65 Share Reply

spanlabs ★ 115 July 12, 2018 7:53 AM

Convert `read - anchor + 1` to string uses extra space O(lgN).

Further more, this problem is somehow worthless, The compressed strings are ambiguous. For example, ["1", "2", "2"]. Algorithms should have practical value. One star for this question.

28 Share Reply

SHOW 2 REPLIES

payalarya ★ 66 November 4, 2017 12:25 PM

Hi,My code in eclipse is working fine,giving me correct output, but it is showing incorrect in leetcode. how to fix this

24 Share Reply

SHOW 3 REPLIES

monil1511 ★ 27 August 21, 2018 8:39 PM

in the worst case the internal for loop and the string will take O(N) space. This is not constant time solution.

9 Share Reply

SHOW 2 REPLIES

_voyageur ★ 153 February 19, 2019 2:40 AM

Python code is poorly written in this solution.

Why write

```
for read, c in enumerate(chars):
```

Read More

20 Share Reply

phoang03 ★ 6 August 12, 2019 4:56 AM

how come you don't have to delete the repeated characters? For eg if it's "a", "a", "a" why don't we have to delete the last 2 "a"s?

6 Share Reply

SHOW 1 REPLY

tinaaa ★ 7 February 16, 2018 3:43 AM

@avice

The line here is confusing at first.

```
if (read > anchor) {
```

Read More

5 Share Reply

kevin37 ★ 10 October 25, 2018 8:28 PM

The above solution sucks. see my two-pointer-approach solution in C#.
Time: O(N) , Space: O(1)

```
public int Compress(char[] str) {
    int len = 0;
```

Read More

9 Share Reply

SHOW 2 REPLIES

mmkhatkhatay ★ 4 October 10, 2019 2:02 PM

The one-pass constant space Python code I ended up writing:

```
        if not chars:
            return 0
```

Read More

3 Share Reply

SHOW 2 REPLIES