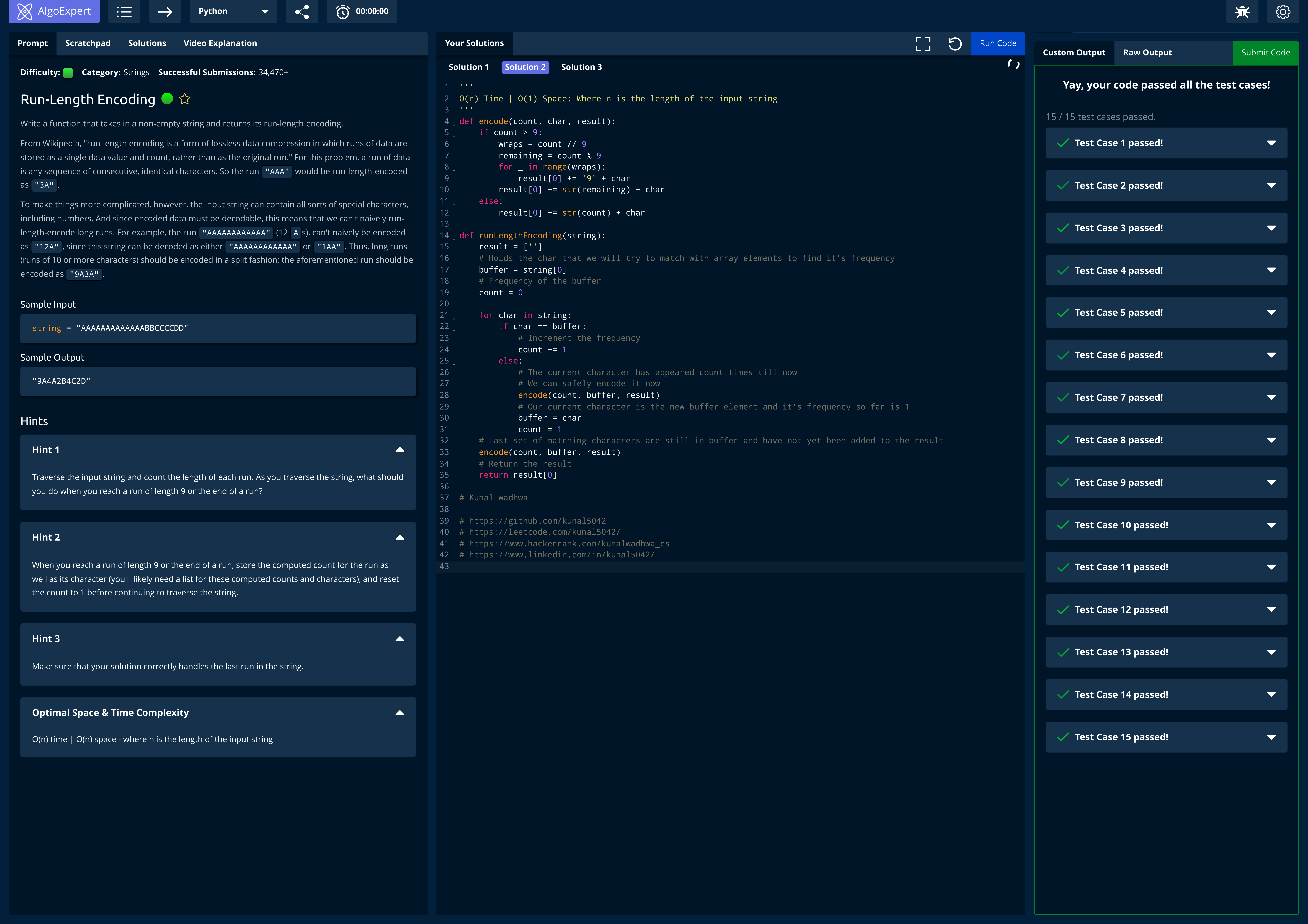

Difficulty: 🟩 **Category:** Strings **Successful Submissions:** 34,470+

Run-Length Encoding

Write a function that takes in a non-empty string and returns its run-length encoding.

From Wikipedia, "run-length encoding is a form of lossless data compression in which runs of data are stored as a single data value and count, rather than as the original run." For this problem, a run of data is any sequence of consecutive, identical characters. So the run `"AAA"` would be run-length-encoded as `"3A"`.

To make things more complicated, however, the input string can contain all sorts of special characters, including numbers. And since encoded data must be decodable, this means that we can't naively run-length-encode long runs. For example, the run `"AAAAAAAAAAAA"` (12 `A` s), can't naively be encoded as `"12A"`, since this string can be decoded as either `"AAAAAAAAAAAA"` or `"1AA"`. Thus, long runs (runs of 10 or more characters) should be encoded in a split fashion; the aforementioned run should be encoded as `"9A3A"`.

Sample Input

```
string = "AAAAAAAAAAAAABBCCCCDD"
```

Sample Output

```
"9A4A2B4C2D"
```

Hints

Hint 1

Traverse the input string and count the length of each run. As you traverse the string, what should you do when you reach a run of length 9 or the end of a run?

Hint 2

When you reach a run of length 9 or the end of a run, store the computed count for the run as well as its character (you'll likely need a list for these computed counts and characters), and reset the count to 1 before continuing to traverse the string.

Hint 3

Make sure that your solution correctly handles the last run in the string.

Optimal Space & Time Complexity

O(n) time | O(n) space - where n is the length of the input string

```python
'''
O(n) Time | O(1) Space: Where n is the length of the input string
'''
def encode(count, char, result):
    if count > 9:
        wraps = count // 9
        remaining = count % 9
        for _ in range(wraps):
            result[0] += '9' + char
        result[0] += str(remaining) + char
    else:
        result[0] += str(count) + char

def runLengthEncoding(string):
    result = ['']
    # Holds the char that we will try to match with array elements to find it's frequency
    buffer = string[0]
    # Frequency of the buffer
    count = 0

    for char in string:
        if char == buffer:
            # Increment the frequency
            count += 1
        else:
            # The current character has appeared count times till now
            # We can safely encode it now
            encode(count, buffer, result)
            # Our current character is the new buffer element and it's frequency so far is 1
            buffer = char
            count = 1
    # Last set of matching characters are still in buffer and have not yet been added to the result
    encode(count, buffer, result)
    # Return the result
    return result[0]

# Kunal Wadhwa

# https://github.com/kunal5042
# https://leetcode.com/kunal5042/
# https://www.hackerrank.com/kunalwadhwa_cs
# https://www.linkedin.com/in/kunal5042/
```

Yay, your code passed all the test cases!

15 / 15 test cases passed.

✓ Test Case 1 passed!
✓ Test Case 2 passed!
✓ Test Case 3 passed!
✓ Test Case 4 passed!
✓ Test Case 5 passed!
✓ Test Case 6 passed!
✓ Test Case 7 passed!
✓ Test Case 8 passed!
✓ Test Case 9 passed!
✓ Test Case 10 passed!
✓ Test Case 11 passed!
✓ Test Case 12 passed!
✓ Test Case 13 passed!
✓ Test Case 14 passed!
✓ Test Case 15 passed!