Our Solution(s) Run Code

Solution 1 Solution 2

```python
# Copyright © 2020 AlgoExpert, LLC. All rights reserved.

# O(nm) time | O(min(n, m)) space
def levenshteinDistance(str1, str2):
    small = str1 if len(str1) < len(str2) else str2
    big = str1 if len(str1) >= len(str2) else str2
    evenEdits = [x for x in range(len(small) + 1)]
    oddEdits = [None for x in range(len(small) + 1)]
    for i in range(1, len(big) + 1):
        if i % 2 == 1:
            currentEdits = oddEdits
            previousEdits = evenEdits
        else:
            currentEdits = evenEdits
            previousEdits = oddEdits
        currentEdits[0] = i
        for j in range(1, len(small) + 1):
            if big[i - 1] == small[j - 1]:
                currentEdits[j] = previousEdits[j - 1]
            else:
                currentEdits[j] = 1 + min(previousEdits[j - 1],
    return evenEdits[-1] if len(big) % 2 == 0 else oddEdits[-1]

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```python
def levenshteinDistance(str1, str2):
    # Write your code here.
    pass

```

Our Tests

Custom Output Submit Code

```
import unittest


class TestProgram(unittest.TestCase):
    def test_case_1(self):
        self.assertEqual(program.lengthOfLastWord(""), 0)

    def test_case_2(self):
        self.assertEqual(program.lengthOfLastWord(""), 0),

    def test_case_3(self):
        self.assertEqual(program.lengthOfLastWord("abc"), 3

    def test_case_4(self):
        self.assertEqual(program.lengthOfLastWord("abc"), 3

    def test_case_5(self):
        self.assertEqual(program.lengthOfLastWord("abc"), 3

    def test_case_6(self):
        self.assertEqual(program.lengthOfLastWord("abc"), 3
```

**Run or submit code when you're ready.**