537. Complex Number Multiplication

Medium Math String Simulation Leetcode Link

Problem Description

In this problem, we are given two complex numbers in the form of strings, where the format of each string is "real+imaginaryi". The task is to multiply these two complex numbers and output the result as a string in the same format. Complex numbers are numbers that include a real part and an imaginary part. The imaginary part is represented by 'i', and by definition, `i^2 = -1`. Multiplication of two complex numbers follows the distributive property from algebra and the special rule of `i^2`.

The inputs:

- `num1`: a string representation of the first complex number.
- `num2`: a string representation of the second complex number.

Each part of the complex numbers (real and imaginary) are within the range of `[-100, 100]`.

The output:

- A string that represents the multiplication of `num1` and `num2` following the structure "real+imaginaryi".

Intuition

To solve this problem, we can break it down by:

1. Parsing the input strings to extract the real and imaginary parts as integers.
2. Performing the multiplication of two complex numbers using the following formula from complex number arithmetic: `(a+bi)(c+di) = ac + adi + bci + bdi^2`
3. Simplifying the equation by considering that `i^2 = -1`. This will give us: `(a+bi)(c+di) = (ac - bd) + (ad + bc)i`
4. The terms `ac - bd` and `ad + bc` give us the real and imaginary parts of the result, respectively.
5. Convert these parts back into the string format, following the required output format.

The implementation of this intuition is straightforward algebra. Given this structure, we split the real and imaginary parts of each input number, perform the arithmetic, and combine the results into a string formatted as a complex number.

Solution Approach

The implementation of the solution follows the mathematical approach detailed in the previous section. Here, we discuss how the provided Python code adheres to the formula `(a+bi)(c+di) = ac - bd + (ad + bc)i` for multiplication of complex numbers.

1. Parsing the Input: The first step in the solution is to parse the input strings `num1` and `num2`. We want to extract the real part (a or c) and the imaginary part (b or d) separately for each number.

- We use the `split('+')` function to separate the real and imaginary parts from the `num1` and `num2` strings since they follow the format "real+imaginaryi". It splits the string into two parts wherever the '+' sign appears.
- To ignore the imaginary unit 'i' at the end of the string, we use the slicing operation `[:-1]` which gives us all the characters in the string except the last one.
- The `map` function is then applied to convert both parts into integers. The real and imaginary parts are stored in variables `a, b` for `num1` and `c, d` for `num2`.

2. Multiplying the Complex Numbers: Next, we calculate the multiplication of the complex numbers using the distributive property and by considering `i^2 = -1`. The actual multiplication is performed using the following components:

- `a * c - b * d` is the real part of the result. This comes from multiplying the real parts and subtracting the product of the imaginary parts (since `i^2 = -1`).
- `a * d + b * c` is the imaginary part of the result. This represents the sum of the products of the real part of one number with the imaginary part of the other.

3. Formatting the Result: Finally, we need to format the result into the required string format "real+imaginaryi". Using an f-string (formatted string literal) in Python, we can directly embed expressions inside a string literal:

- The expression `f'{a * c - b * d}+{a * d + c * b}i'` converts the real and imaginary parts back into the required string format. The placeholders `{}` are replaced with the evaluated result of the expressions inside them.

In conclusion, the algorithm doesn't use complex data structures or patterns as the solution is a direct application of the formula for multiplying complex numbers. The parsing of strings and arithmetic operations are the main aspects of the solution.

Example Walkthrough

Let's illustrate the solution approach with an example. Consider the two complex numbers represented by the strings `num1 = "1+3i"` and `num2 = "2+4i"`. We want to multiply these two complex numbers as per the formula and return the result in the standard complex number format.

1. Parsing the Input:
    - We split `num1` and `num2` by '+' to get the real and imaginary parts separately.
        - `num1`: "1+3i" `->` real part `1` and imaginary part "3i".
        - `num2`: "2+4i" `->` real part `2` and imaginary part "4i".
    - We remove the trailing 'i' from the imaginary parts.
        - Imaginary part of `num1`: "3i" `->` "3".
        - Imaginary part of `num2`: "4i" `->` "4".
    - Convert these strings into integers.
        - `num1`: real part `a = 1`, imaginary part `b = 3`.
        - `num2`: real part `c = 2`, imaginary part `d = 4`.
2. Multiplying the Complex Numbers:
    - We use the distributive property: `(a+bi)(c+di) = ac - bd + (ad + bc)i`.
    - Multiply the real parts and subtract the product of the imaginary parts, since `i^2 = -1`: `1 * 2 - 3 * 4 = 2 - 12 = -10`.
    - Add the product of the real part of one number with the imaginary part of the other: `1 * 4 + 3 * 2 = 4 + 6 = 10`.
    - The resulting real part is `-10` and the imaginary part is `+10i`.
3. Formatting the Result:
    - We combine the real and imaginary parts into one string: `"-10+10i"`.
    - This string follows the format "real+imaginaryi", which is our final output.

Therefore, multiplying `num1 = "1+3i"` and `num2 = "2+4i"` using the given approach yields the result `"-10+10i"`. This demonstrates the implementation of the mathematical multiplication of complex numbers in the provided Python code, and how the parsed parts are used in the distributive property to get the desired output.

Python Solution

```python
class Solution:
    def complexNumberMultiply(self, num1: str, num2: str) -> str:
        # Split the real and imaginary parts of the first complex number 'num1'
        real1, imaginary1 = map(int, num1[:-1].split('+'))
        # Split the real and imaginary parts of the second complex number 'num2'
        real2, imaginary2 = map(int, num2[:-1].split('+'))

        # Perform multiplication of two complex numbers using the formula:
        # (a + bi) * (c + di) = (ac - bd) + (ad + bc)i
        # Calculate the real part as (real1 * real2 - imaginary1 * imaginary2)
        real_part = real1 * real2 - imaginary1 * imaginary2
        # Calculate the imaginary part as (real1 * imaginary2 + real2 * imaginary1)
        imaginary_part = real1 * imaginary2 + real2 * imaginary1

        # Return the result as a string in the format 'real_part+imaginary_parti'
        result = f'{real_part}+{imaginary_part}i'
        return result

```

Java Solution

```java
class Solution {

    // Function to multiply two complex numbers given as strings
    public String complexNumberMultiply(String num1, String num2) {
        // Split each input string into real and imaginary parts
        String[] complexOneComponents = num1.split("\\+|i");
        String[] complexTwoComponents = num2.split("\\+|i");

        // Parse the real and imaginary parts of the first complex number
        int realPartOne = Integer.parseInt(complexOneComponents[0]);
        int imaginaryPartOne = Integer.parseInt(complexOneComponents[1]);

        // Parse the real and imaginary parts of the second complex number
        int realPartTwo = Integer.parseInt(complexTwoComponents[0]);
        int imaginaryPartTwo = Integer.parseInt(complexTwoComponents[1]);

        // Apply the formula for multiplication of two complex numbers:
        // (a + bi) * (c + di) = ac + adi + bci - bd = (ac - bd) + (ad + bc)i
        int realResult = realPartOne * realPartTwo - imaginaryPartOne * imaginaryPartTwo;
        int imaginaryResult = realPartOne * imaginaryPartTwo + imaginaryPartOne * realPartTwo;

        // Construct the resulting complex number as a string and return it
        return String.format("%d+%di", realResult, imaginaryResult);
    }
}

```

C++ Solution

```cpp
#include <string>
using namespace std;

class Solution {
public:
    // Function to multiply two complex numbers given as strings
    string complexNumberMultiply(string num1, string num2) {
        // Initializing integer variables to contain real and imaginary parts
        int real1, imag1, real2, imag2;

        // Parse the first complex number string and extract real and imaginary parts
        sscanf(num1.c_str(), "%d+%di", &real1, &imag1);
        // Parse the second complex number string and extract real and imaginary parts
        sscanf(num2.c_str(), "%d+%di", &real2, &imag2);

        // Calculate the real part of the product of the two complex numbers
        int realProduct = real1 * real2 - imag1 * imag2;
        // Calculate the imaginary part of the product of the two complex numbers
        int imagProduct = real1 * imag2 + real2 * imag1;

        // Construct the result string in the format "real+imagi"
        string result = to_string(realProduct) + "+" + to_string(imagProduct) + "i";

        // Return the resulting string of the complex number multiplication
        return result;
    }
};

```

Typescript Solution

```typescript
function complexNumberMultiply(num1: string, num2: string): string {
    // Split the complex numbers into real and imaginary parts.
    let partsOfNum1 = num1.split('+'),
        partsOfNum2 = num2.split('+');

    // Parse the real parts of the complex numbers from the strings.
    let realPart1 = Number(partsOfNum1[0]),
        realPart2 = Number(partsOfNum2[0]);

    // Parse the imaginary parts of the complex numbers from the strings.
    // The 'i' character is stripped from the end of the string.
    let imaginaryPart1 = Number(partsOfNum1[1].slice(0, -1)),
        imaginaryPart2 = Number(partsOfNum2[1].slice(0, -1));

    // Calculate the real part of the product by using the formula:
    // (a + bi) * (c + di) = (ac - bd) + (ad + bc)i
    // where a and b are the real and imaginary parts of the first complex number,
    // and c and d are the same for the second number.
    let productReal = realPart1 * realPart2 - imaginaryPart1 * imaginaryPart2;

    // Calculate the imaginary part of the product.
    let productImaginary = realPart1 * imaginaryPart2 + realPart2 * imaginaryPart1;

    // Construct the string representation of the complex product.
    return `${productReal}+${productImaginary}i`;
}

```

Time and Space Complexity

Time complexity

The time complexity of the code is `O(1)`. This is because the code involves a constant number of operations regardless of the input size. Splitting the strings and performing arithmetic operations does not depend on the length of the input, as the format of the input is fixed (it represents a complex number).

Space complexity

The space complexity of the code is also `O(1)`. Only a fixed number of additional variables are used for storing the parsed integers (`a`, `b`, `c`, `d`) and for constructing the final result. The storage required does not scale with input size, confirming that the space complexity is constant.