roman to integer leetcode solution

Roman to Integer LeetCode Solution is a classic problem that challenges programmers to convert a given Roman numeral string into its corresponding integer value. This problem is not only an excellent way to practice algorithmic thinking but also serves as a useful exercise in understanding how to manipulate and interpret different numeral systems. While the problem may seem straightforward at first glance, it requires careful consideration of the rules governing Roman numeral formation and their respective values.

Understanding Roman Numerals

Roman numerals are a numeral system originating in ancient Rome, utilizing combinations of letters from the Latin alphabet. The basic Roman numerals and their values are as follows:

- I = 1
- V = 5
- X = 10
- L = 50
- C = 100
- D = 500
- M = 1000

Roman numerals are typically written from largest to smallest from left to right. However, there are exceptions known as subtractive combinations, where a smaller numeral precedes a larger numeral, indicating subtraction. These combinations include:

- IV = 4 (5 - 1)
- IX = 9 (10 - 1)
- XL = 40 (50 - 10)
- XC = 90 (100 - 10)
- CD = 400 (500 - 100)
- CM = 900 (1000 - 100)

Understanding these rules is crucial for correctly converting Roman numerals into integers.

The Problem Statement

The LeetCode problem typically asks you to implement a function that takes a string representing a Roman numeral and returns its integer value. You are expected to handle valid inputs, as outlined by the constraints provided in the problem statement.

Example Input and Output

- Input: "III"
- Output: 3

- Input: "IV"
- Output: 4

- Input: "IX"
- Output: 9

- Input: "LVIII"
- Output: 58 (50 + 5 + 3)

- Input: "MCMXCIV"
- Output: 1994 (1000 + 900 + 90 + 4)

Approach to the Solution

To solve the Roman to Integer problem, we can employ a straightforward algorithm that iterates through the characters of the Roman numeral string. The key steps in this approach are as follows:

1. Create a mapping of Roman numerals to their integer values.
2. Iterate through the string of Roman numerals:
- Compare the value of the current numeral with the value of the next numeral.
- If the current numeral is less than the next numeral, subtract its value from the total.
- If the current numeral is greater than or equal to the next numeral, add its value to the total.
3. Return the final calculated total.

Pseudocode

Here is a simple pseudocode representation of the above approach:

```
function romanToInt(s):
romanMap = { 'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000
}
total = 0

for i from 0 to length(s) - 1:
if i < length(s) - 1 and romanMap[s[i]] < romanMap[s[i + 1]]:
total -= romanMap[s[i]]
else:
total += romanMap[s[i]]
```

```
return total
```

Implementation of the Solution

Let’s look at a Python implementation of the above pseudocode. This implementation utilizes a dictionary to map Roman characters to their integer values and iterates through the input string to calculate the final integer value.

```python
def romanToInt(s: str) -> int:
Mapping of Roman numerals to integers
romanMap = {
'I': 1,
'V': 5,
'X': 10,
'L': 50,
'C': 100,
'D': 500,
'M': 1000
}

total = 0
n = len(s)

for i in range(n):
If the current numeral is less than the next one, we subtract its value
if i < n - 1 and romanMap[s[i]] < romanMap[s[i + 1]]:
total -= romanMap[s[i]]
else:
total += romanMap[s[i]]

return total
```

Explanation of the Code

- The function `romanToInt` takes a string `s` as input.
- A dictionary `romanMap` is created to hold the integer values of each Roman numeral.
- A variable `total` is initialized to zero to keep track of the cumulative integer value.
- The loop iterates over each character in the string:
- If the current numeral is less than the next numeral, its value is subtracted from the total.
- Otherwise, its value is added to the total.
- Finally, the total is returned as the output.

Complexity Analysis

The time complexity of this solution is O(n), where n is the length of the input string. This is because we are iterating through the string once. The space complexity is O(1) since we are using a fixed amount of additional space for the mapping dictionary.

Testing the Function

To ensure the function works correctly, it’s essential to test it with various cases, including edge cases. Here are some test cases:

```python
assert romanToInt("III") == 3
assert romanToInt("IV") == 4
assert romanToInt("IX") == 9
assert romanToInt("LVIII") == 58
assert romanToInt("MCMXCIV") == 1994
assert romanToInt("MMXXIII") == 2023 Testing with a recent year
```

Conclusion

The Roman to Integer problem on LeetCode is an excellent exercise for sharpening algorithmic skills and understanding numeral systems. By implementing a systematic approach to convert Roman numerals into integers, we not only solve the problem effectively but also gain insights into handling various data representations. This solution can be extended and adapted to handle more complex numeral systems or variations in input requirements, making it a versatile tool in a programmer's toolkit.

Frequently Asked Questions

What is the problem statement for the 'Roman to Integer' LeetCode problem?

The problem asks you to convert a given Roman numeral string into its corresponding integer value, following the rules of Roman numeral representation.

What is the time complexity of the optimal solution for the 'Roman to Integer' problem?

The optimal solution has a time complexity of O(n), where n is the length of

the Roman numeral string, as it requires a single pass through the string.

What are the valid Roman numeral symbols and their corresponding integer values?

The valid symbols are: I (1), V (5), X (10), L (50), C (100), D (500), M (1000).

How do you handle subtractive combinations in Roman numerals?

Subtractive combinations like IV (4) and IX (9) are handled by checking if a smaller numeral precedes a larger numeral; if so, subtract the smaller from the larger.

What is an example input and output for the 'Roman to Integer' problem?

Input: 'MCMXCIV' (1994), Output: 1994.

What is a common mistake when converting Roman numerals to integers?

A common mistake is not correctly handling the cases where a smaller numeral appears before a larger numeral, which indicates subtraction.

What data structure can be used to store the values of Roman numerals?

A hash map (or dictionary) can be used to map Roman numeral characters to their corresponding integer values for quick lookup.

Can you provide a simple algorithm to solve the 'Roman to Integer' problem?

1. Create a map of Roman symbols to integers. 2. Initialize a total variable. 3. Loop through the string, comparing each numeral with the next one. 4. Add or subtract the values based on comparisons. 5. Return the total.

What are some edge cases to consider when solving the 'Roman to Integer' problem?

Edge cases include handling empty strings, invalid characters, and the maximum valid Roman numeral (MMMCMXCIX for 3999).

Roman To Integer Leetcode Solution

Roman To Integer Leetcode Solution