Documentation: Fraction to Recurring Decimal Conversion

Overview

This document explains the process of converting a fraction, represented by a numerator and denominator, into its decimal representation. It details the steps required to handle both terminating and repeating decimals, ensuring that the output is correctly formatted according to the problem's requirements.

Problem Statement

Given two integers representing the numerator and denominator of a fraction, return the fraction as a string in decimal format. If the decimal is a repeating decimal, the repeating part should be enclosed in parentheses.

Example 1:

- **Input:** numerator = 1, denominator = 2
- **Output:** "0.5"

Example 2:

- **Input:** numerator = 2, denominator = 1
- **Output:** "2"

Example 3:

- **Input:** numerator = 4, denominator = 333
- **Output:** "0.(012)"

Constraints

- The numerator and denominator are both within the range ([-2^{31}, 2^{31} 1]).
- The denominator is non-zero.

Steps to Solve the Problem

1. Handling the Sign of the Result

- The sign of the final result is determined based on the signs of the numerator and denominator.
- If either (but not both) the numerator or denominator is negative, the result should be negative.
- This step is crucial to ensure that the final output is correctly formatted with or without a negative sign.

2. Calculating the Integer Part

- The integer part of the decimal is obtained by performing integer division of the numerator by the denominator.
- This gives the portion of the fraction that does not involve any remainder or fractional part.

3. Handling the Remainder

- After calculating the integer part, the remainder is computed by taking the modulus of the numerator and denominator.
- If the remainder is zero, the decimal part terminates, and the final result is simply the integer part.
- If the remainder is non-zero, it indicates that there is a fractional part to be computed.

4. **Calculating the Fractional Part**
    - The fractional part is calculated by continuously multiplying the remainder by 10 and dividing by the denominator to get each successive digit.
    - The remainder is then updated by taking the modulus of the new remainder and the denominator.
    - This process continues until the remainder becomes zero (indicating a terminating decimal) or repeats (indicating a repeating decimal).

5. **Detecting Repeating Cycles**
    - To detect repeating decimals, a dictionary (or hash map) is used to store each remainder and its corresponding index in the fractional part.
    - If a remainder reappears during the calculation of the fractional part, it means that a repeating cycle has been detected.
    - The decimal digits corresponding to this cycle should be enclosed in parentheses.

6. **Constructing the Final Result**
    - The result is constructed by combining the sign (if any), the integer part, and the fractional part.
    - If a repeating cycle was detected, the digits corresponding to the cycle are enclosed in parentheses to indicate the repeating section.
    - Finally, the result is returned as a string.

Edge Cases and Considerations

- **Zero Numerator:** If the numerator is zero, the result is always "0", regardless of the denominator.

- **Denominator of 1:** If the denominator is 1, the result is simply the integer value of the numerator.

- **Repeating Decimals:** Properly handling repeating decimals is critical, as failure to detect and correctly format repeating cycles will result in incorrect output.

- **Large Numbers:** Given the constraints, the solution must efficiently handle large values of the numerator and denominator without running into performance issues or overflow errors.

Summary

This solution efficiently converts a fraction into its decimal representation by handling signs, computing the integer and fractional parts, and detecting repeating cycles. The use of a dictionary to track remainders ensures that the algorithm can quickly identify repeating decimals and format the output accordingly. The approach ensures that the final result is accurate, concise, and correctly formatted, meeting all problem constraints.