Fraction to Recurring Decimal

Given two integers representing the numerator and denominator of a fraction, return the fraction in string format.

If the fractional part is repeating, enclose the repeating part in parentheses.

For example,

- Given numerator = 1, denominator = 2, return "0.5".
- Given numerator = 2, denominator = 1, return "2".
- Given numerator = 2, denominator = 3, return "0.(6)".

Credits:
Special thanks to @Shangrila for adding this problem and creating all test cases.

Solution 1

```
// upgraded parameter types
string fractionToDecimal(int64_t n, int64_t d) {
    // zero numerator
    if (n == 0) return "0";

    string res;
    // determine the sign
    if (n < 0 ^ d < 0) res += '-';

    // remove sign of operands
    n = abs(n), d = abs(d);

    // append integral part
    res += to_string(n / d);

    // in case no fractional part
    if (n % d == 0) return res;

    res += '.';

    unordered_map<int, int> map;

    // simulate the division process
    for (int64_t r = n % d; r; r %= d) {

        // meet a known remainder
        // so we reach the end of the repeating part
        if (map.count(r) > 0) {
            res.insert(map[r], 1, '(');
            res += ')';
            break;
        }

        // the remainder is first seen
        // remember the current position for it
        map[r] = res.size();

        r *= 10;

        // append the quotient digit
        res += to_string(r / d);
    }

    return res;
}
```

written by mzchen original link here

Solution 2

The important thing is to consider all edge cases while thinking this problem through, including: negative integer, possible overflow, etc.

Use HashMap to store a remainder and its associated index while doing the division so that whenever a same remainder comes up, we know there is a repeating fractional part.

Please comment if you see something wrong or can be improved. Cheers!

```
public class Solution {
    public String fractionToDecimal(int numerator, int denominator) {
        if (numerator == 0) {
            return "0";
        }
        StringBuilder res = new StringBuilder();
        // "+" or "-"
        res.append(((numerator > 0) ^ (denominator > 0)) ? "-" : "");
        long num = Math.abs((long)numerator);
        long den = Math.abs((long)denominator);

        // integral part
        res.append(num / den);
        num %= den;
        if (num == 0) {
            return res.toString();
        }

        // fractional part
        res.append(".");
        HashMap<Long, Integer> map = new HashMap<Long, Integer>();
        map.put(num, res.length());
        while (num != 0) {
            num *= 10;
            res.append(num / den);
            num %= den;
            if (map.containsKey(num)) {
                int index = map.get(num);
                res.insert(index, "(");
                res.append(")");
                break;
            }
            else {
                map.put(num, res.length());
            }
        }
        return res.toString();
    }
}
```

written by dchen0215 original link here

Solution 3

```
public String fractionToDecimal(int numerator, int denominator) {
    StringBuilder result = new StringBuilder();
    String sign = (numerator < 0 == denominator < 0 || numerator == 0) ? "" : "-"
;
    long num = Math.abs((long) numerator);
    long den = Math.abs((long) denominator);
    result.append(sign);
    result.append(num / den);
    long remainder = num % den;
    if (remainder == 0)
        return result.toString();
    result.append(".");
    HashMap<Long, Integer> hashMap = new HashMap<Long, Integer>();
    while (!hashMap.containsKey(remainder)) {
        hashMap.put(remainder, result.length());
        result.append(10 * remainder / den);
        remainder = 10 * remainder % den;
    }
    int index = hashMap.get(remainder);
    result.insert(index, "(");
    result.append(")");
    return result.toString().replace("(0)", "");
}
```

written by tusizi original link here