# 题目

给定一个表示分数加减运算的字符串 expression ，你需要返回一个字符串形式的计算结果。

这个结果应该是不可约分的分数，即最简分数。 如果最终结果是一个整数，例如 2，你需要将它转换成分数形式，其分母为 1。所以在上述例子中, 2 应该被转换为 2/1。

示例 1:

输入: expression = "-1/2+1/2"

输出: "0/1"

示例 2:

输入: expression = "-1/2+1/2+1/3"

输出: "1/3"

示例 3:

输入: expression = "1/3-1/2"

输出: "-1/6"

提示:

输入和输出字符串只包含 '0' 到 '9' 的数字，以及 '/', '+' 和 '-'。

输入和输出分数格式均为 ±分子/分母。如果输入的第一个分数或者输出的分数是正数，则 '+' 会被省略掉。

输入只包含合法的最简分数，每个分数的分子与分母的范围是 [1,10]。 如果分母是1，意味着这个分数实际上是一个整数。

输入的分数个数范围是 [1,10]。

最终结果的分子与分母保证是 32 位整数范围内的有效整数。

# 分析

## 方法一：模拟

思路：

代码：

class Solution {

public:

string fractionAddition(string expression) {

long long x = 0, y = 1; // 分子，分母

int index = 0, n = expression.size();

while (index < n) {

// 读取分子

long long x1 = 0, sign = 1;

if (expression[index] == '-' || expression[index] == '+') {

sign = expression[index] == '-' ? -1 : 1;

index++;

}

while (index < n && isdigit(expression[index])) {

// 计算分子的数

x1 = x1 \* 10 + expression[index] - '0';

index++;

}

x1 = sign \* x1;

index++;

// 读取分母

long long y1 = 0;

while (index < n && isdigit(expression[index])) {

y1 = y1 \* 10 + expression[index] - '0';

index++;

}

// 根据通分的方法实现分数运算

x = x \* y1 + x1 \* y;

y \*= y1;

}

if (x == 0) {

return "0/1";

}

long long g = gcd(abs(x), y); // 获取最大公约数

return to\_string(x / g) + "/" + to\_string(y / g);

}

};

复杂度分析：

时间复杂度：O(n+logC)，其中n是字符串expression的长度，C为化简前结果分子分母的最大值。求最大公约数需要O(logC)。

空间复杂度：O(1)。