0227. 基本计算器 II

ITCharge 大约 2 分钟

- 标签：栈、数学、字符串
- 难度：中等

题目链接

- 0227. 基本计算器 II - 力扣

题目大意

描述：给定一个字符串表达式 `s`，表达式中所有整数为非负整数，运算符只有 `+`、`-`、`*`、`/`，没有括号。

要求：实现一个基本计算器来计算并返回它的值。

说明：

- $1 \leq s.length \leq 3 * 10^5$。
- `s` 由整数和算符（`+`、`-`、`*`、`/`）组成，中间由一些空格隔开。
- `s` 表示一个有效表达式。
- 表达式中的所有整数都是非负整数，且在范围 $[0, 2^{31} - 1]$ 内。
- 题目数据保证答案是一个 32-bit 整数。

示例：

- 示例 1：

```py
输入：s = "3+2*2"
输出：7
```

- 示例 2：

```py
输入：s = " 3/2 "
输出：1
```

解题思路

思路 1：栈

计算表达式中，乘除运算优先于加减运算。我们可以先进行乘除运算，再将进行乘除运算后的整数值放入原表达式中相应位置，再依次计算加减。

可以考虑使用一个栈来保存进行乘除运算后的整数值。正整数直接压入栈中，负整数，则将对应整数取负号，再压入栈中。这样最终计算结果就是栈中所有元素的和。

具体做法：

1. 遍历字符串 `s`，使用变量 `op` 来标记数字之前的运算符，默认为 `+`。
2. 如果遇到数字，继续向后遍历，将数字进行累积，得到完整的整数 num。判断当前 op 的符号。
    1. 如果 `op` 为 `+`，则将 `num` 压入栈中。
    2. 如果 `op` 为 `-`，则将 `-num` 压入栈中。
    3. 如果 `op` 为 `*`，则将栈顶元素 `top` 取出，计算 `top * num`，并将计算结果压入栈中。
    4. 如果 `op` 为 `/`，则将栈顶元素 `top` 取出，计算 `int(top / num)`，并将计算结果压入栈中。
3. 如果遇到 `+`、`-`、`*`、`/` 操作符，则更新 `op`。
4. 最后将栈中整数进行累加，并返回结果。

思路 1：代码

py

```
class Solution:
    def calculate(self, s: str) -> int:
        size = len(s)
        stack = []
        op = '+'
        index = 0
        while index < size:
            if s[index] == ' ':
                index += 1
                continue
            if s[index].isdigit():
```

```
                num = ord(s[index]) - ord('0')
                while index + 1 < size and s[index+1].isdigit():
                    index += 1
                    num = 10 * num + ord(s[index]) - ord('0')
                if op == '+':
                    stack.append(num)
                elif op == '-':
                    stack.append(-num)
                elif op == '*':
                    top = stack.pop()
                    stack.append(top * num)
                elif op == '/':
                    top = stack.pop()
                    stack.append(int(top / num))
            elif s[index] in "+-*/":
                op = s[index]
            index += 1
        return sum(stack)
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$。
- **空间复杂度**：$O(n)$。