0150. 逆波兰表达式求值

ITCharge 大约 2 分钟

- 标签：栈、数组、数学
- 难度：中等

题目链接

- 0150. 逆波兰表达式求值 - 力扣

题目大意

描述：给定一个字符串数组 `tokens` ，表示「逆波兰表达式」。

要求：求解表达式的值。

说明：

- **逆波兰表达式**：也称为后缀表达式
  - 中缀表达式 `( 1 + 2 ) * ( 3 + 4 )` ，对应的逆波兰表达式为 `( ( 1 2 + ) ( 3 4 + ) * )` 。
- $1 \leq tokens.length \leq 10^4$。
- `tokens[i]` 是一个算符（`+`、`-`、`*` 或 `/`），或是在范围 $[-200, 200]$ 内的一个整数。

示例：

- 示例 1：

```py
输入：tokens = ["4","13","5","/","+"]
输出：6
解释：该算式转化为常见的中缀算术表达式为：(4 + (13 / 5)) = 6
```

- 示例 2：

```py
输入：tokens = ["10","6","9","3","+","-11","*","/","*","17","+","5","+"]
输出：22
解释：该算式转化为常见的中缀算术表达式为：
  ((10 * (6 / ((9 + 3) * -11))) + 17) + 5
= ((10 * (6 / (12 * -11))) + 17) + 5
= ((10 * (6 / -132)) + 17) + 5
= ((10 * 0) + 17) + 5
= (0 + 17) + 5
= 17 + 5
= 22
```

解题思路

思路 1：栈

这道题是栈的典型应用。我们先来简单介绍一下逆波兰表达式。

逆波兰表达式，也叫做后缀表达式，特点是：没有括号，运算符总是放在和它相关的操作数之后。 我们平常见到的表达式是中缀表达式，可写为： `A 运算符 B` 。其中 `A` 、 `B` 都是操作数。 而后缀表达式可写为： `A B 运 `。

逆波兰表达式的计算遵循从左到右的规律。我们在计算逆波兰表达式的值时，可以使用一个栈来存放当前的操作数，从左到右依次遍历逆波兰表达式，计算出对应的值。具体操作步骤如下：

1. 使用列表 `stack` 作为栈存放操作数，然后遍历表达式的字符串数组。
2. 如果当前字符为运算符，则取出栈顶两个元素，在进行对应的运算之后，再将运算结果入栈。
3. 如果当前字符为数字，则直接将数字入栈。
4. 遍历结束后弹出栈中最后剩余的元素，这就是最终结果。

思路 1：代码

```py
class Solution:
    def evalRPN(self, tokens: List[str]) -> int:
        stack = []
        for token in tokens:
            if token == '+':
```

```
            stack.append(stack.pop() + stack.pop())
        elif token == '-':
            stack.append(-stack.pop() + stack.pop())
        elif token == '*':
            stack.append(stack.pop() * stack.pop())
        elif token == '/':
            stack.append(int(1 / stack.pop() * stack.pop()))
        else:
            stack.append(int(token))
    return stack.pop()
```

思路 1：复杂度分析

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