0008. 字符串转换整数 (atoi)

ITCharge 大约 3 分钟

- 标签：字符串
- 难度：中等

题目链接

- 0008. 字符串转换整数 (atoi) - 力扣

题目大意

描述：给定一个字符串 `s` 。

要求：实现一个 `myAtoi(s)` 函数。使其能换成一个 32 位有符号整数（类似 C / C++ 中的 `atoi` 函数）。需要检测有效性，无法读取返回 0。

说明：

- 函数 `myAtoi(s)` 的算法如下：
  1. 读入字符串并丢弃无用的前导空格。
  2. 检查下一个字符（假设还未到字符末尾）为正还是负号，读取该字符（如果有）。 确定最终结果是负数还是正数。 如果两者都不存在，则假定结果为正。
  3. 读入下一个字符，直到到达下一个非数字字符或到达输入的结尾。字符串的其余部分将被忽略。
  4. 将前面步骤读入的这些数字转换为整数（即，`"123" -> 123` ，`"0032" -> 32` ）。如果没有读入数字，则整数为 `0` 。必要时更改符号（从步骤 2 开始）。
  5. 如果整数数超过 32 位有符号整数范围 $[-2^{31}, 2^{31} - 1]$ ，需要截断这个整数，使其保持在这个范围内。具体来说，小于 -2^{31} 的整数应该被固定为 -2^{31} ，大于 $2^{31} - 1$ 的整数应该被固定为 $2^{31} - 1$。
  6. 返回整数作为最终结果。
- 本题中的空白字符只包括空格字符 `' '` 。
- 除前导空格或数字后的其余字符串外，请勿忽略任何其他字符。
- $0 \leq s.length \leq 200$。
- `s` 由英文字母（大写和小写）、数字（`0-9`）、`' '`、`'+'`、`'-'` 和 `'.'` 组成

示例：

- 示例 1：

```py
输入：s = "42"
输出：42
解释：加粗的字符串为已经读入的字符，插入符号是当前读取的字符。
第 1 步："42"（当前没有读入字符，因为没有前导空格）
         ^
第 2 步："42"（当前没有读入字符，因为这里不存在 '-' 或者 '+'）
         ^
第 3 步："42"（读入 "42"）
           ^
解析得到整数 42 。
由于 "42" 在范围 [-231, 231 - 1] 内，最终结果为 42 。
```

- 示例 2：

```py
输入：s = "   -42"
输出：-42
解释：
第 1 步："   -42"（读入前导空格，但忽视掉）
            ^
第 2 步："   -42"（读入 '-' 字符，所以结果应该是负数）
             ^
第 3 步："   -42"（读入 "42"）
               ^
解析得到整数 -42 。
由于 "-42" 在范围 [-231, 231 - 1] 内，最终结果为 -42 。
```

解题思路

思路 1：模拟

1. 先去除前后空格。
2. 检测正负号。
3. 读入数字，并用字符串存储数字结果。
4. 将数字字符串转为整数，并根据正负号转换整数结果。
5. 判断整数范围，并返回最终结果。

思路 1：代码

```py
class Solution:
    def myAtoi(self, s: str) -> int:
        num_str = ""
        positive = True
        start = 0

        s = s.lstrip()
        if not s:
            return 0

        if s[0] == '-':
            positive = False
            start = 1
        elif s[0] == '+':
            positive = True
            start = 1
        elif not s[0].isdigit():
            return 0

        for i in range(start, len(s)):
            if s[i].isdigit():
                num_str += s[i]
            else:
                break
        if not num_str:
            return 0
        num = int(num_str)
        if not positive:
            num = -num
            return max(num, -2 ** 31)
        else:
            return min(num, 2 ** 31 - 1)
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$，其中 n 是字符串 `s` 的长度。

- **空间复杂度**：$O(1)$。