算法分析和复杂性理论

第 2 次作业

张瀚文 2201212865

1 整数反转

题目描述：给你一个 32 位的有符号整数 x ，返回将 x 中的数字部分反转后的结果。如果反转后整数超过 32 位的有符号整数的范围[−231,231−1]，就返回 0。假设环境不允许存储 64 位整数（有符号或无符号）。

解题思路：用取模运算逐个数位取出数字，按顺序就可以反向拼接出一个数字了，达到反转的效果。题目需要判断反转后的数字是否超过 32 位有符号整数的范围，因此需要在每次拼接反转数字前，先做一个判断。

运行结果：

测试代码：

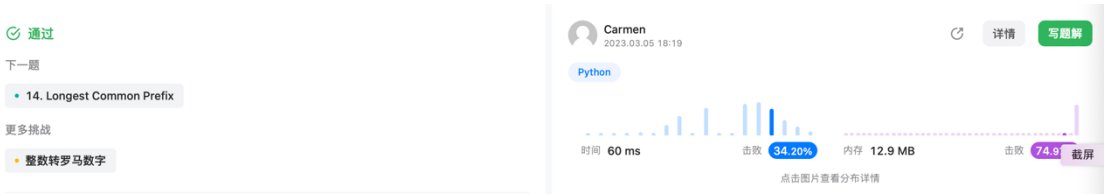
```
class Solution {
public:
    int reverse(int x) {
        int rev = 0;
        while (x != 0) {
            if (rev < INT_MIN / 10 || rev > INT_MAX / 10) {
                return 0;
            }
            int digit = x % 10;
            x /= 10;
            rev = rev * 10 + digit;
        }
        return rev;
    }
};
```

2 罗马数字转整数

题目描述：给定一个罗马数字，将其转换成整数。

解题思路：通常情况下，罗马数字中小的数字在大的数字的右边。若输入的字符串满足该情况，那么可以将每个字符视作一个单独的值，累加每个字符对应的数值即可。若存在小的数字在大的数字的左边的情况，根据规则需要减去小的数字。对于这种情况，我们也可以将每个字符视作一个单独的值，若一个数字右侧的数字比它大，则将该数字的符号取反。

运行结果：



测试代码：

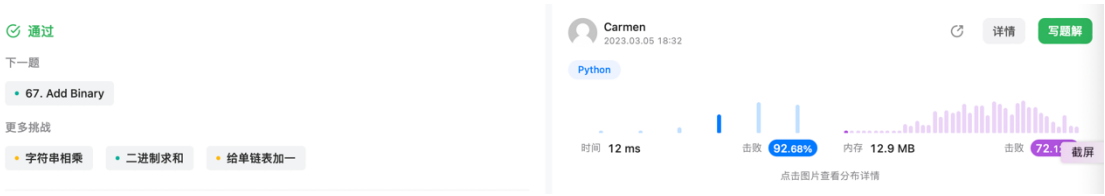
```
class Solution:

    SYMBOL_VALUES = {
        'I': 1,
        'V': 5,
        'X': 10,
        'L': 50,
        'C': 100,
        'D': 500,
        'M': 1000,
    }

    def romanToInt(self, s: str) -> int:
        ans = 0
        n = len(s)
        for i, ch in enumerate(s):
            value = Solution.SYMBOL_VALUES[ch]
            if i < n - 1 and value < Solution.SYMBOL_VALUES[s[i + 1]]:
                ans -= value
            else:
                ans += value
        return ans
```

3 加一

题目描述：给定一个由整数组成的非空数组所表示的非负整数，在该数的基础上加一。最高位数字存放在数组的首位，数组中每个元素只存储单个数字。你可以假设除了整数 0 之外，这个整数不会以零开头。

解题思路：只需要对数组进行一次逆序遍历，找出第一个不为 9 的元素，将其加一并将后续所有元素置零即可。如果 digits 的所有元素都是 9，我们只需要构造一个长度比 digits 多 1 的新数组，将首元素置为 1，其余元素置 0 即可。

运行结果：



测试代码：

```
class Solution:
    def plusOne(self, digits: List[int]) -> List[int]:
        n = len(digits)
        for i in range(n - 1, -1, -1):
            if digits[i] != 9:
                digits[i] += 1
                for j in range(i + 1, n):
                    digits[j] = 0
                return digits

        # digits 中所有的元素均为 9
        return [1] + [0] * n
```