链滴

【leetCode】整数反转

作者：zeekling

原文链接：https://ld246.com/article/1605713691332

来源网站：链滴

题目

给出一个 32 位的有符号整数，你需要将这个整数中每位上的数字进行反转。

示例 1:

输入: 123

输出: 321

示例 2:

输入: -123

输出: -321

示例 3:

输入: 120

输出: 21

注意:

假设我们的环境只能存储得下 32 位的有符号整数，则其数值范围为 [−231，231 − 1]。请根据这个设，如果反转后整数溢出那么就返回 0。

链接：https://leetcode-cn.com/problems/reverse-integer

题目解答

方法：弹出和推入数字 & 溢出前进行检查

思路

我们可以一次构建反转整数的一位数字。在这样做的时候，我们可以预先检查向原整数附加另一位数是否会导致溢出。

算法

反转整数的方法可以与反转字符串进行类比。

我们想重复“弹出” x 的最后一位数字，并将它“推入”到 \text{rev}的后面。最后， \text{rev} 将与 x 相反。

要在没有辅助堆栈 / 数组的帮助下 “弹出” 和 “推入” 数字，我们可以使用数学方法。

```
//pop operation:
pop = x % 10;
x /= 10;//push operation:
temp = rev * 10 + pop;
rev = temp;
```

但是，这种方法很危险，因为当 temp=rev□10+pop 时会导致溢出。

幸运的是，事先检查这个语句是否会导致溢出很容易。

为了便于解释，我们假设 \text{rev} 是正数。

如果 temp=rev□10+pop 导致溢出，那么一定有 \text{rev} \geq \frac{INTMAX}{10}。
如果 \text{rev} > \frac{INTMAX}{10}

，那么 temp=rev□10+pop 一定会溢出。
如果 \text{rev} == \frac{INTMAX}{10} ，那么只要 \text{pop} > 7，temp=rev□10+pop 就会溢出。
当 \text{rev} 为负时可以应用类似的逻辑。

```
public int reverse(int x) {
       int rev = 0;
        while (x != 0) {
            int pop = x % 10;
            x /= 10;
            if (rev > Integer.MAX_VALUE/10 || (rev == Integer.MAX_VALUE / 10 && pop > 7)) retu
n 0;
            if (rev < Integer.MIN_VALUE/10 || (rev == Integer.MIN_VALUE / 10 && pop < -8)) retu
n 0;
            rev = rev * 10 + pop;
        }
        return rev;
 }
```

复杂度分析

时间复杂度：O(\log(x))，x 中大约有 \log_{10}(x) 位数字。

空间复杂度：O(1)。