05 整数反转

更新时间：2019-08-09 10:02:29

> 对自己不满是任何真正有才能的人的根本特征之一。
>
> ——契诃夫

刷题内容

> **难度: Easy**
>
> 原题连接：https://leetcode-cn.com/problems/reverse-integer/。

内容描述

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]。请根据这个假设，如果反转后整数溢出那么就返回 0。

题目详解

这道题目要求我们**将一个有符号整数进行反转**，也就是说 64 这个数字再反转后会变为 46 ，个位数反转到十位数，十位数反转到个位数。如果给出的整数是负数的话，整数的符号不能变。在这道题中我们要注意以下几点：

- 整数会有负数的情况；
- 题目中要求的有符号整数是一个 32 位整数(32 位整数的取值范围：-2147483648~2147483647，超出此范围即为溢出)，如果将整数反转后发生了溢出情况，那么要返回 0 ；
- 最后一位是 0 的情况下要舍弃，例如 120 反转后为 21 。

解题方案

思路1：时间复杂度: O(lgx) 空间复杂度: O(1)

翻转数字问题需要注意的就是溢出问题，为什么会存在溢出问题呢？

我们 int 型的数值范围是 -2147483648～2147483647 (-2^31 ~ 2^31 - 1)， 那么如果我们要翻转 1000000009 这个在数值范围内的数，得到 9000000001，但翻转后的数就超过了范围，这个情况就是**溢出**，这个时候程序返回 0 即可。

如果输入的是负数，就递归调用原函数，参数变成 -x 即可，代码大致执行流程如下：

- 首先判断 x 是否为负数，如果是负数先取绝对值然后递归取反，最后再将结果转换为负数即可；
- res 为最后结果，最开始等于 0 。x % 10 (例如： 123 % 10 = 3) 取出最后一位数。每次得到最后一位数字，并将其作为结果中的当前最高位；
- 判断结果 res 是否溢出，如果溢出则返回 0 。

下面来看详细代码：

Python beats 69.76%

```python
class Solution:
    def reverse(self, x: int) -> int:
        if x < 0: # 判断是否为负数
            return -self.reverse(-x) # 如果是负数则取绝对值调用自身，最后将结果转为负数
        res = 0
        while x: # 每次得到最后一位数字，并将其作为结果中的当前最高位
            res = res * 10 + x % 10
            x //= 10
        return res if res <= 0x7fffffff else 0 # 如果溢出就返回0
```

Java beats 100%

```
class Solution {
    public int reverse(int x) {
        if (x == -2147483648) { //如果不做这个判断，下面的x=-x将会报错
            return 0;
        }
        if (x < 0) { // 判断是否为负数
            return -reverse(-x); // 如果是负数则取绝对值调用自身，最后将结果转为负数
        }
        int res = 0;
        while (x != 0) { // 每次得到最后一位数字，并将其作为结果中的当前最高位
            if (res > 214748364) { // 处理溢出
                return 0;
            }
            res = res * 10 + x % 10;
            x /= 10;
        }
        return res <= 0x7fffffff ? res : 0; // 如果溢出就返回0
    }
}
```

Go beats 96.66%

```
func reverse(x int) int {
    if x < 0 { // 如果是负数就取反递归再取反
        return -reverse(-x)
    }

    var res int
    for x != 0 { // 每次得到最后一位数字，并将其作为结果中的当前最高位
        res = res * 10 + x % 10
        x /= 10
    }
    if res <= 0x7fffffff { // 如果溢出就返回0
        return res
    }
    return 0
}
```

c++

```
class Solution {
public:
    int reverse(int x) {
        if (x == -2147483648) { //如果不做这个判断，下面的x=-x将会报错
            return 0;
        }
        if (x < 0) { // 判断是否为负数
            return -reverse(-x); // 如果是负数则取绝对值调用自身，最后将结果转为负数
        }
        int res = 0;
        while (x != 0) { // 每次得到最后一位数字，并将其作为结果中的当前最高位
            if (res > 214748364) { // 处理溢出
                return 0;
            }
            res = res * 10 + x % 10;
            x /= 10;
        }
        return res <= 0x7fffffff ? res : 0; // 如果溢出就返回0
    }
};
```

各版本代码虽然语法不同，但是核心代码还是一样的。即使你看不懂上面的代码，那就看一下思路就好，不管用哪种语言实现都是可以的。

小结

- 时刻注意特殊情况——为负数怎么办；为0怎么办；为空怎么办。
- 合理运用写完的函数，机智一点，不要碰到负数又去实现一个负数的版本，直接递归调用原函数就可以了。

}

04 两数之和

06 回文数