字节 7. 整数反转

7. 整数反转

难度 简单 2665

给你一个 32 位的有符号整数 x ，返回 x 中每位上的数字反转后的结果。

如果反转后整数超过 32 位的有符号整数的范围 $[-2^{31}, 2^{31}-1]$ ，就返回 0。

假设环境不允许存储 64 位整数（有符号或无符号）。

示例 1：

```
输入：x = 123
输出：321
```

示例 2：

```
输入：x = -123
输出：-321
```

示例 3：

```
输入：x = 120
输出：21
```

示例 4：

```
输入：x = 0
输出：0
```

提示：

- $-2^{31} <= x <= 2^{31}-1$

题解

一、字符串反转

① 将数字转换为字符串

② 反转字符串

```
/**
 * @param {number} x
 * @return {number}
 */
var reverse = function(x) {
    if (x > 0) {
        let result = rev(x.toString());
        if (result > Math.pow(2, 31) - 1) {
            return 0;
        }
        return result;
    } else if (x === 0) {
        return 0;
    } else if (x < 0) {
        let result = rev(x.toString().slice(1));
        result -= 2 * result;
        if (result < Math.pow(2, 31) * (-1)) {
            return 0;
        }
        return result;
    }

    function rev(x) {
        return parseInt(x.split('').reverse().join(''));
    }
};
```

复杂度分析

时间复杂度：与数字的长度相关，x 的长度 $\log_{10} x$

$O(\log x)$

空间复杂度：$O(1)$

二、数字思路

构建一个反转的数字 rev，x 每"推出"一个尾数，便附加到 rev 后。

"弹出" `pop = x % 10;`

`x /= 10;`

"推入" `temp = rev * 10 + pop;`

`rev = temp`

"溢出" 当 $rev > \frac{MAX\_INT}{10}$，则 temp 溢出

当 $rev = \frac{MAX\_INT}{10}$，$pop > 7$ 时溢出，因为 $2^{31}-1$ 是 2147483647

负数同理，$pop < -8$ 时溢出。

```
/**
 * @param {number} x
 * @return {number}
 */
var reverse = function(x) {
    const MAX_INT_D_10  = parseInt((Math.pow(2, 31) - 1) / 10);
    const N_MAX_INT_D_10 = parseInt((Math.pow(2, 31) * (-1)) / 10);
    let rev = 0;
    while(x !== 0) {
        const pop = x % 10;
        x = parseInt(x / 10);
        if (rev > MAX_INT_D_10 || rev < N_MAX_INT_D_10) {
            return 0;
        } else if (rev === MAX_INT_D_10 && pop > 7) {
            return 0;
        } else if (rev === N_MAX_INT_D_10 && pop < -8) {
            return 0;
        } else {
            rev = rev * 10 + pop;
        }
    }
    return rev;
};
```

三、位运算

运用位运算的规则

① (x / 10) | 0，当小数做位运算的时候，正数向下取整，负数向上取整

② (result | 0) === result.

JavaScript 做位运算的时候，将 64 位转化为 32 位，运算完再转回 64 位，

result | 0 只截取 32 位部分，依此可判断 result 是否溢出 32 位.

```
/**
 * @param {number} x
 * @return {number}
 */
var reverse = function (x) {
    let result = 0;
    while (x !== 0) {
        const pop = x % 10;
        result = result * 10 + pop;
        x = (x / 10) | 0;
    }
    return (result | 0) === result ? result : 0;
};
```