golang 整数常量INT_MAX INT_MIN最大值最小值

在C语言中，有标准库limits.h定义了一些最大最小值常量，例如int类型的最大值常量INT_MAX，最小值常量INT_MIN，无符号整型uint类型的最大值常量UINT_MAX golang的标准库里没有定义这些变量。不过可以用位操作运算，轻松定义这些常量。

无符号整型uint

其最小值是0，其二进制表示的所有位都为0，

```
const UINT_MIN uint = 0
```

其最大值的二进制表示的所有位都为1，那么，

```
const UINT_MAX = ^uint(0)
```

有符号整型int

根据补码，其最大值二进制表示，首位0，其余1，那么，

```
const INT_MAX = int(^uint(0) >> 1)
```

根据补码，其最小值二进制表示，首位1，其余0，那么，

```
const INT_MIN = ^INT_MAX
```

反转整数：

方法：弹出和推入数字 & 溢出前进行检查

我们想重复“弹出” x 的最后一位数字，并将它“推入”到 rev 的后面。最后，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 $\text{temp} = \text{rev} \cdot 10 + \text{pop}$ 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 rev 是正数。

1. 如果 $temp = \text{rev} \cdot 10 + \text{pop}$ 导致溢出，那么一定有 $\text{rev} \geq \frac{INTMAX}{10}$。
2. 如果 $\text{rev} > \frac{INTMAX}{10}$，那么 $temp = \text{rev} \cdot 10 + \text{pop}$ 一定会溢出。
3. 如果 $\text{rev} == \frac{INTMAX}{10}$，那么只要 $\text{pop} > 7$，$temp = \text{rev} \cdot 10 + \text{pop}$ 就会溢出。

当 rev 为负时可以应用类似的逻辑。

```
package main

import (
    "fmt"
)

//0值取反向右移动一位，即首位0，其余1，最大
const INT_MAX = int(^uint(0) >> 1) //9223372036854775807
const INT_MIN = ^INT_MAX          //-9223372036854775808

func main() {
    a := ReverseInt(1223372036854775899)
    fmt.Println(a)
}

//方法：弹出和推入数字 & 溢出前进行检查
func ReverseInt(x int) int {
    var rev int = 0
    for x != 0 {
```

```
        pop := x % 10
        x = x / 10
        if rev > INT_MAX/10 || (rev == INT_MAX/10 && pop > 7) {
            return 0
        }
        if rev < INT_MIN/10 || (rev == INT_MIN/10 && pop < -8) {
            return 0
        }
        rev = rev*10 + pop
    }

    return rev

}
```