# 题目

给定一个非负整数num，反复将各个位上的数字相加，直到结果为一位数。返回这个结果。

示例 1:

输入: num = 38

输出: 2

解释: 各位相加的过程为：

38 --> 3 + 8 --> 11

11 --> 1 + 1 --> 2

由于 2 是一位数，所以返回 2。

示例 2:

输入: num = 0

输出: 0

提示：

0 <= num <= 231 - 1

进阶：你可以不使用循环或者递归，在 O(1) 时间复杂度内解决这个问题吗？

# 分析

## 方法一：模拟

思路：

代码：

class Solution {

public:

int addDigits(int num) {

while (num >= 10) {

int sum = 0;

while (num > 0) {

sum += num % 10;

num /= 10;

}

num = sum;

}

return num;

}

};

复杂度分析

时间复杂度：O(lognum)，其中num是给定的整数。对于num计算一次各位相加需要O(lognum)的时间，由于num≤231−1，因此对于 num计算一次各位相加的最大可能结果是82，对于任意两位数最多只需要计算两次各位相加的结果即可得到一位数。

空间复杂度：O(1)。

## 方法二：数学