# 统计一

## 问题描述

NewCode总是力争上游，凡事都要拿第一，所以他对“1”这个数情有独钟。爱屋及乌，他也很喜欢包含1的数，例如10、11、12、……。

例如：N=2，1、2出现了1个“1”。N=12，1、2、3、4、5、6、7、8、9、10、11、12。出现了5个“1”。你能帮他统计一下整数里有多少个1吗？

## 解题思路

### 解法一：

最直接的方法就是从1开始遍历到N，将其中每一个数中含有”1”的个数加起来，就得到了问题的解。

|  |
| --- |
| **private static long** countOne3(**long** n) {  **long** count = **0**;  **for** (**int** i = **0**, j; i <= n; i++) {  j = i;  **while** (j != **0**) {  **if** (j % **10** == **1**) {  count++;  }  j = j / **10**;  }  }  **return** count; } |

此方法简单，容易理解，但它的问题是效率，时间复杂度为，N比较大的时候，需要耗费很长的时间。

### 解法二：

我们重新分析下这个问题，对于任意一个个位数n，只要n≥1,它就包含一个“1”；n<1，即n=0时，则包含的“1”的个数为0。于是我们考虑用分治的思想将任意一个n位数不断缩小规模分解成许多个个位数，这样求解就很方便。

但是，我们该如何降低规模？仔细分析，我们会发现，任意一个m位数其值为n（假设这个数为）中“1”的个位可以分解为两个m-1位数（这两个数是和）中“1”的个数的和加上一个与最高位数相关的常数C。例如，f(12)=f(10-1)+f(12-10)+3，其中3是表示最高位为1的数字个数，这里就是10、11、12；f(132)=f(100-1)+f(132-100)+33，33代表最高位为1的数字的个数，这里就是100~132；f(232)=2\*f(100-1)+f(32)+100，因为232大于199，所以它包括了所有最高位为1的数字即100~199，共100个。

综上，我们分析得出，最后加的常数C只跟最高位是否为1有关。

当最高位为1时，常数C为原数字n去掉最高位后剩下的数字+1。

当最高位不为1时，常数C为，其中bit为N的位数-1,即m-1。如n=12时，bit=1，n=232时，bit=2。

于是，我们可以列出递归方程如下（其中n=，bit=m-1）：

说明：

1. 对于(1)，f(n)（其中n=）可以表示为数字序列(0,1,2,…,n)中1出现的次数，因为最高位为1，表示最高位为1的**出现次数（不是1的个数）**。对于**其它部分**1的出现次数还未求出。其它部分成**两个部分**:

* 数字还未到bit位，有(0,1,2,…,)的数字序列，要求其这个序列1出现的次数，使用求得。
* 数字已经到了bit位，最高位为1，有数字序列(,…,)，因为最高位出现的1的次数已经统计了为，所有只要统计其它位出现的次数，实际就是(0,1,2,…,)序列中1出现的次数，。

1出现的次数为：

1. 对于(2)，最高位为1出现的次数为，不考虑最高位为同时忽略最高位，(0,1,2,…,)序列一共出现了次，则1出现的次数为。如果最高位为数字，则有序列(0,1,2,…,)，1出现的次数为：。

1出现的次数为：

递归的出口条件为：

综合即为：

### 解法三：

解法二的优点是不用遍历1~N就可以得到f(N)。经过测试，此算法的运算速度比解法一快了许多许多，。但算法二有一个显著的**缺点就是当数字超过某个值时会导致堆栈溢出，无法计算**。

解法二告诉我们1~N中“1”的个数跟最高位有关，那我们换个角度思考，给定一个N，我们分析1~N中的数在每一位上出现1的次数的和，看看每一位上“1”出现的个数的和由什么决定。

1位数的情况：在解法二中已经分析过，大于等于1的时候，有1个，小于1就没有。

2位数的情况：N=13，个位数出现的1的次数为2，分别为1和11，十位数出现1的次数为4，分别为10、11、12、13，所以f(N)=2+4。N=23，个位数出现的1的次数为3，分别为1、11、21，十位数出现1的次数为10，分别为10~19，f(N)=3+10。

由此我们发现，个位数出现1的次数不仅和个位数有关，和十位数也有关，如果个位数大于等于1，则个位数出现1的次数为十位数的数字加1；如果个位数为0，个位数出现1的次数等于十位数数字。而十位数上出现1的次数也不仅和十位数相关，也和个位数相关：如果十位数字等于1，则十位数上出现1的次数为个位数的数字加1，假如十位数大于1，则十位数上出现1的次数为10。

3位数的情况：N=123，个位出现1的个数为13:1、11、21、…、91、101、111、121。十位出现1的个数为20:10~19、110~119。百位出现1的个数为24:100~123。

我们可以继续分析4位数，5位数，推导出下面一般情况：假设N，我们要计算百位上出现1的次数，将由三部分决定：百位上的数字，百位以上的数字，百位以下的数字。

**如果百位上的数字为0**，**则百位上出现1的次数仅由更高位决定**，比如12013，百位出现1的情况为100~199、1100~1199、2100~2199、…、11100~11199，共1200个。**等于更高位数字乘以当前位数**，即12\*100。

**如果百位上的数字大于1**，**则百位上出现1的次数仅由更高位决定**，比如12213，百位出现1的情况为100~199、1100~1199、2100~2199、…、11100~11199、12100~12199共1300个。**等于更高位数字加1乘以当前位数**，即(12+1)\*100。

**如果百位上的数字为1**，**则百位上出现1的次数不仅受更高位影响，还受低位影响**。例如12113，受高位影响出现1的情况（**等于更高位数字乘以当前位数**）：100~199、1100~1199、2100~2199、…、11100~11199，共1200个，但它还受低位影响（），出现1的情况是12100~12113，共14个，等于低位数字13+1。

假设n=，当前处理第i位，f(i)为第i位上1出现的次数则有：

[0,n]的序列中1出现的总次数是：