1到N的所有整数的1出现的次数\_剑指Offer43

# 1到N的所有整数的1出现的次数\_剑指Offer43

## 问题描述

/\*\*\*\*\*\*剑指Offer43:1到N正整数中1出现的次数\*\*\*\*\*/

/\*\*\*

\* 题目描述

\* 求出1~13的整数中1出现的次数,并算出100~1300的整数中1出现的次数？

\* 为此他特别数了一下1~13中包含1的数字有1、10、11、12、13因此共出现6次,

\* 但是对于后面问题他就没辙了。ACMer希望你们帮帮他,并把问题更加普遍化,

\* 可以很快的求出任意非负整数区间中1出现的次数（从1 到 n 中1出现的次数）。

## 思路分析：

\* 蛮力法：编写一个方法，参数为数值，返回该数值的1的个数；

\* 然后遍历1到N，调用上面的方法，统计累加1的总个数。

\* 存在问题：如100-199，对于百位1，蛮力法需要通过100次求余统计出来，

\* 是不是只要是知道百位是1，就直接返回为1呢？并且十位和个位

\* 分别为1时，有2\*10 = 20个。然后递归调用两位数、一位数即可。

\* 鉴于这种思路，利用递归思路实现统计。

\* 巧妙方法：探查数学规律，降低复杂度。

\* 思路：一次递归只统计当前位数的1出现个数，如五位数，需要执行递归方法5次，

\* 第一次统计五位数1出现个数；第二次统计四位数1出现个数；...第五次统计个数1出现个数。

\* 递归终止条件：一位数，若是0返回0，否则返回1；

\* 以五位数为例，分成三部分：

\* A. 最高位如果大于1，则直接返回10^4,即10000-19999之间万位为1的个数；

\* 若等于1，则返回后四位数+1，即10000-n之间的万位为1的个数；

\* 如21234，返回10000；而11234，返回1235；

\* B. 后四位判断：任何一个为1，其余任意，则有4\*10^3;

\* 若万位为1,则只有1xxxx,若万位为2，则有1xxxx,2xxxx；因此返回first\*4\*10^3;\

\* 注意：11234统计的是1235到11234范围，C步调用递归只需要统计1234及之下1出现次数；

\* C. 递归调用后四位如11234的1234以内1出现的次数即可。

\* 最后返回以上三部分的统计结果之和。

\*/

## Java代码

### 蛮力法

/\*\*

\* 蛮力法：计算每一个数中1的个数，累加在一起。

\*/

public int NumberOf1Between1AndN\_Solution1(int n) {

int count = 0;

while(n > 0){

count += countEveryNumber(n--);

}

return count;

}

public int countEveryNumber(int num){

if(num <= 0) return 0;

int count = 0;

while(num != 0){

if(num%10 == 1) count++;

num /= 10;

}

return count;

}

### 分段递归法

/\*\*

\* 巧妙方法：分段。

\*/

public int NumberOf1Between1AndN\_Solution(int n) {

if(n <= 0) return 0;

return countRecursive(n);

}

public int countRecursive(int num){

if(num/10 == 0) return num>=1?1:0;//个位数

//位数有必要每次都计算一次，因为10000,四位数0000其从位数5直接变为位数1

int weiShu = weiShu(num),base = (int)Math.pow(10,weiShu-1);

//处理最高位

int first = num/base;

int count = 0;

if(first > 1) count += base;

else count += num%base + 1;

//处理后几位

count += first\*(weiShu-1)\*base/10;

//递归调用weiShu-1位数据

count += countRecursive(num%base);

return count;

}

public int weiShu(int num){

if(num < 0) return 0;

int weiShu = 1;

while((num = num/10) != 0) weiShu++;

return weiShu;

}

/\* \*//\*\*

\* 巧妙方法：分段法。

\* 通过传递位数参数有**错误实例**，因为10000,位数是5，而下次位数就是1，不是4.

\* 必须每次都要计算位数。

\*//\*

public int NumberOf1Between1AndN\_Solution2(int num) {

if(num <= 0) return 0;

int weiShu = 1,copy = num;

while((copy = copy/10) != 0) weiShu++;

return countRecursive(num,weiShu);

}

public int countRecursive(int num,int weiShu){

if(weiShu == 1) return num>=1?1:0;//个位数

int base = (int)Math.pow(10,weiShu-1);

//处理最高位

int first = num/base;

int count = 0;

if(first > 1) count += base;

else count += num%base + 1;

//处理后几位

count += first\*(weiShu-1)\*base/10;

//递归调用weiShu-1位数据

count += countRecursive(num%base,weiShu-1);

return count;

}\*/