

统计一

1 题目描述

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？

1.1 输入描述：

输入有多组数据，每组数据包含一个正整数 n ，($1 \leq n \leq 2147483647$)。

1.2 输出描述：

对应每组输入，输出从 1 到 n （包含 1 和 n ）之间包含数字 1 的个数。例如 11 与 101 包含 2 个 1、1101 包含 3 个 1。

1.3 输入例子：

```
1
9
10
20
```

1.4 输出例子：

```
1
1
2
12
```

2 解题思路

2.1 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 $O(N \lg N)$ ， N 比较大的时候，需要耗费很长的时间。

2.2 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 $n \geq 1$ ，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 $x_1 x_2 x_3 \dots x_m$ ）中“1”的个数可以分解为两个 $m-1$ 位数（这两个数是 $10^m - 1$ 和 $x_2 x_3 \dots x_m$ ）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 x_1 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 10^{bit} ，其中 bit 为 N 的位数-1，即 $m-1$ 。如 $n=12$ 时， $\text{bit}=1$ ， $n=232$ 时， $\text{bit}=2$ 。

于是，我们可以列出递归方程如下（其中 $n = x_1 x_2 x_3 \dots x_m$ ， $\text{bit} = m-1$ ）：

$$f(n) = \begin{cases} f(10^{\text{bit}} - 1) + f(n - 10^{\text{bit}}) + n - 10^{\text{bit}} + 1 & n \geq 10, x_1 = 1 \\ x_1 * f(10^{\text{bit}} - 1) + f(n - x_1 * 10^{\text{bit}}) + 10^{\text{bit}} & n \geq 10, x_1 \neq 1 \end{cases}$$

说明：

- 对于(1)， $f(n)$ （其中 $n = x_1 x_2 x_3 \dots x_m$ ）可以表示为数字序列 $(0, 1, 2, \dots, n)$ 中 1 出现的次数，因为最高位为 1， $n - 10^{\text{bit}} + 1$ 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 $(0, 1, 2, \dots, 10^{\text{bit}} - 1)$ 的数字序列，要求其这个序列 1 出现的次数，使用 $f(10^{\text{bit}} - 1)$ 求得。
- 数字已经到了 bit 位，最高位为 1，有数字序列 $(10^{\text{bit}}, 10^{\text{bit}} + 1, \dots, x_1 x_2 x_3 \dots x_{\text{bit}})$ ，因为最高位出现的 1 的次数已经统计了为 $n - 10^{\text{bit}} + 1$ ，所有只要统计其它位出现的次数，实际就是 $(0, 1, 2, \dots, x_2 x_3 \dots x_{\text{bit}})$ 序列中 1 出现的次数，。

1 出现的次数为： $f(n) = f(10^{\text{bit}} - 1) + f(n - 10^{\text{bit}}) + n - 10^{\text{bit}} + 1$

2. 对于 (2)，最高位为 1 出现的次数为 10^{bit} ，不考虑最高位为 x_1 同时忽略最高位， $(0, 1, 2, \dots, 10^{\text{bit}} - 1)$ 序列一共出现了 x_1 次，则 1 出现的次数为 $x_1 * f(10^{\text{bit}} - 1)$ 。如果最高位为数字 x_1 ，则有序列 $(0, 1, 2, \dots, x_2 x_3 \dots x_{\text{bit}})$ ，1 出现的次数为： $f(n - x_1 * 10^{\text{bit}})$ 。

1 出现的次数为： $f(n) = x_1 * f(10^{\text{bit}} - 1) + f(n - x_1 * 10^{\text{bit}}) + 10^{\text{bit}}$

递归的出口条件为：

$$f(n) = \begin{cases} 1 & 0 < n < 10 \\ 0 & n \leq 0 \end{cases}$$

综合即为：

$$f(n) = \begin{cases} 0 & n \leq 0 \\ 1 & 0 < n < 10 \\ f(10^{\text{bit}} - 1) + f(n - 10^{\text{bit}}) + n - 10^{\text{bit}} + 1 & n \geq 10, x_1 = 1 \\ x_1 * f(10^{\text{bit}} - 1) + f(n - x_1 * 10^{\text{bit}}) + 10^{\text{bit}} & n \geq 10, x_1 \neq 1 \end{cases}$$

2.3 解法三：

解法二的优点是不用遍历 $1 \sim N$ 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 $1 \sim N$ 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 $1 \sim N$ 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 $(12+1) \times 100$ 。

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 $n = x_1 x_2 x_3 \dots x_i \dots x_n$ ，当前处理第 i 位， $f(i)$ 为第 i 位上 1 出现的次数则有：

$$f(i) = \begin{cases} (x_1 \dots x_{i-1}) * 10^{n-i} & x_i = 0 \\ (x_1 \dots x_{i-1}) * 10^{n-i} + (x_{i+1} \dots x_n + 1) & x_i = 1 \\ (x_1 \dots x_{i-1} + 1) * 10^{n-i} & x_i > 1 \end{cases}$$

[0,n] 的序列中 1 出现的总次数是：

$$\sum_i^n f(i)$$