然后 1~1345 的数字上出现 1 的个数可以按照如上方式递归求解。

具体过程请参看如下代码中的 solution2 方法。

```
public int solution2(int num) {
        if (num < 1) {
                return 0;
        }
        int len = getLenOfNum(num);
        if (len == 1) {
                return 1;
        }
        int tmp1 = powerBaseOf10(len - 1);
        int first = num / tmp1;
        int firstOneNum = first == 1 ? num % tmp1 + 1 : tmp1;
        int otherOneNum = first * (len - 1) * (tmp1 / 10);
        return firstOneNum + otherOneNum + solution2(num % tmp1);
}

public int getLenOfNum(int num) {
        int len = 0;
        while (num != 0) {
                len++;
                num /= 10;
        }
        return len;
}

public int powerBaseOf10(int base) {
        return (int) Math.pow(10, base);
}
```

仅通过分析如上代码就可以知道，n 一共有多少位，递归函数最多就会被调用多少次，即 log$N$ 次。在递归函数内部 getLenOfNum 方法和 powerBaseOf10 方法的复杂度分别为 $O(\log N)$和 $O(\log(\log N))$。求一个数的 A 次方的问题在系统内部实现的复杂度为 $O(\log A)$，A 为 N 的位数(A=log$N$)，所以 powerBaseOf10 方法的时间复杂度为 $O(\log(\log N))$。所以方法二的总时间复杂度为 $O(\log N \times \log N)$。

从 N 个数中等概率打印 M 个数

【题目】

给定一个长度为 N 且没有重复元素的数组 arr 和一个整数 n，实现函数等概率随机打印 arr 中的 M 个数。

【要求】

1．相同的数不要重复打印。

2．时间复杂度为 $O(M)$，额外空间复杂度为 $O(1)$。

3．可以改变 arr 数组。

【难度】

士 ★☆☆☆

【解答】

如果没有空间复杂度的限制，可以用哈希表标记一个数之前是否被打印过，就可以做到不重复打印。解法的关键点是利用要求 3 改变数组 arr。打印过程如下：

1．在[0,*N*-1]中随机得到一个位置 a，然后打印 arr[a]。

2．把 arr[a]和 arr[N-1]交换。

3．在[0,N-2]中随机得到一个位置 b，然后打印 arr[b]，因为打印过的 arr[a]已被换到了 *N*-1 位置，所以这次打印不可能再次出现。

4．把 arr[b]和 arr[N-2]交换。

5．在[0,*N*-3]中随机得到一个位置 c，然后打印 arr[c]，因为打印过的 arr[a]和 arr[b]已被换到了 *N*-1 位置和 *N*-2 位置，所以这次打印都不可能再出现。

6．依此类推，直到打印 *M* 个数。

总之，就是把随机选出来的数打印出来，然后将打印的数交换到范围中的最后位置，再把范围缩小，使得被打印的数下次不可能再被选中，直到打印结束。很多有关等概率随机的面试题都是用这种和最后一个位置交换的解法，希望这种小技巧能引起读者的重视。具体过程请参看如下代码中的 printRandM 方法。

```
public void printRandM(int[] arr, int m) {
        if (arr == null || arr.length == 0 || m < 0) {
                return;
        }
        m = Math.min(arr.length, m);
        int count = 0;
        int i = 0;
        while (count < m) {
                i = (int) (Math.random() * (arr.length - count));
                System.out.println(arr[i]);
                swap(arr, arr.length - count++ - 1, i);
        }
```

```
    }

    public void swap(int[] arr, int index1, int index2) {
            int tmp = arr[index1];
            arr[index1] = arr[index2];
            arr[index2] = tmp;
    }
```

判断一个数是否是回文数

【题目】

定义回文数的概念如下：

- 如果一个非负数左右完全对应，则该数是回文数，例如：121，22 等。
- 如果一个负数的绝对值左右完全对应，也是回文数，例如：-121，-22 等。

给定一个 32 位整数 num，判断 num 是否是回文数。

【难度】

士 ★☆☆☆

【解答】

本题的实现方法当然有很多种，本书介绍一种仅用一个整型变量就可以实现的方法，步骤如下：

1．假设判断的数字为非负数 n，先生成变量 help，开始时 help=1。

2．用 help 不停地乘以 10，直到变得与 num 的位数一样。例如：num 等于 123321 时，help 就是 100000。num 如果是 131，help 就是 100，总之，让 help 与 num 的位数一样。

3．那么 num/help 的结果就是最高位的数字，num%10 就是最低位的数字，比较这两个数字，不相同则直接返回 false。相同则令 num=(num%help)/10，即 num 变成除去最高位和最低位两个数字之后的值。令 help/=100，即让 help 变得继续和新的 num 位数一样。

4．如果 num==0，表示所有的数字都已经对应判断完，返回 true，否则重复步骤 3。

上述方法就是让 num 每次剥掉最左和最右两个数，然后逐渐完成所有对应的判断。需要注意的是，如上方法只适用于非负数的判断，如果 n 为负数，则先把 n 变成其绝对值，然后用上面的方法进行判断。同时还需注意，32 位整数中的最小值为-2147483648，它是转不成相应的绝对值的，可这个数也很明显不是回文数。所以，如果 n 为-2147483648，直接