```
public int[] getRanMSysNumLessN(int[] nMSys, int m) {
        int[] res = new int[nMSys.length];
        int start = 0;
        while (nMSys[start] == 0) {
                start++;
        }
        int index = start;
        boolean lastEqual = true;
        while (index != nMSys.length) {
                res[index] = randlToM(m) - 1;
                if (lastEqual) {
                        if (res[index] > nMSys[index]) {
                                index = start;
                                lastEqual = true;
                                continue;
                        } else {
                                lastEqual = res[index] == nMSys[index];
                        }
                }
                index++;
        }
        return res;
}

// 把 m 进制数转成十进制数
public int getNumFromMSysNum(int[] mSysNum, int m) {
        int res = 0;
        for (int i = 0; i != mSysNum.length; i++) {
                res = res * m + mSysNum[i];
        }
        return res;
}
```

一行代码求两个数的最大公约数

【题目】

给定两个不等于 0 的整数 M 和 N，求 M 和 N 的最大公约数。

【难度】

士 ★★☆☆

【解答】

一个很简单的求两个数最大公约数的算法是欧几里得在其《几何原本》中提出的欧几

里得算法，又称为辗转相除法。

具体做法为：如果 q 和 r 分别是 m 除以 n 的商及余数，即 $m=nq+r$，那么 m 和 n 的最大公约数等于 n 和 r 的最大公约数。详细证明略。

具体请参看如下代码中的 gcd 方法。

```
public int gcd(int m, int n) {
        return n == 0 ? m : gcd(n, m % n);
}
```

有关阶乘的两个问题

【题目】

给定一个非负整数 N，返回 $N!$结果的末尾为 0 的数量。

例如：3!=6，结果的末尾没有 0，则返回 0。5!=120，结果的末尾有 1 个 0，返回 1。1000000000!，结果的末尾有 249999998 个 0，返回 249999998。

【进阶题目】

给定一个非负整数 N，如果用二进制数表达 $N!$的结果，返回最低位的 1 在哪个位置上，认为最右的位置为位置 0。

例如：1!=1，最低位的 1 在 0 位置上。2!=2，最低位的 1 在 1 位置上。1000000000!，最低位的 1 在 999999987 位置上。

【难度】

原问题　尉 ★★☆☆

进阶问题　校 ★★★☆

【解答】

无论是原问题还是进阶问题，通过算出真实的阶乘结果后再处理的方法无疑是不合适的，因为阶乘的结果通常很大，非常容易溢出，而且会增加计算的复杂性。

先来介绍原问题的一个普通解法。对原问题来说，$N!$结果的末尾有多少个 0 的问题可以转换为 1，2，3，…，N-1，N 的序列中一共有多少个因子 5。这是因为 $1\times2\times3\times\cdots\times N$ 的过程中，因子 2 的数目比因子 5 的数目多，所以不管有多少个因子 5，都有足够的因子 2