辗转相除（数学证明及算法）

1.概述

辗转相除，又称欧几里得算法，用于求最大公约数
Java实现如下：

```java
//循环实现
private static int gcd(int a, int b) {
        int temp = 0;
        while(b != 0){
          // 换位
            temp = a % b;
            a = b;
            b = temp;
        }
        return a;
    }
```

```java
//递归实现
 int gcd(int a, int b) {
      return b == 0 ? a : gcd(b, a % b);
 }
```

这个算法实现很简单，但是这么难理解可是让人"辗转"呀

那到底是什么样子呢？我们以递归的代码来分析。
首先我们证明一个东西

> 下面的证明参考：http://www.cnblogs.com/zwffff/archive/2010/08/25/1808178.html

求证：gcd(m,n)=gcd(n,b)

其中

- m和n是两个数，gcd(m,n)表示m和n的最大公约数
- 设a，b分别是m除以n的商和余数，即m=n * a+b。

证明：

设c=gcd(m,n), d=gcd(n,b)

∵ c为m和n的公约数，
∴ m能被c整除，n也能被c整除
∴ n * a可以被c 整除 【注1】
∴ m - n * a 也能被c整除 【注2】
∵ 题目条件m=n * a+b
∴ m - n * a = b
∴ c为n和b的公约数
∵d为n和b的最大公约数
∴ c <= d 【注3】

∵ d为n和b的公约数
∴ n能被d整除，b也能被d整除
∴ n * a 能被d整除 【注1】
∴ n * a + b 能被d整除，根据题目条件，也即m能被d整除
∴ d为m和n的公约数
∵ c为m和n的最大公约数
∴ d <= c

∴ 综上，d = c
即gcd(m,n)=gcd(n,b) 证毕

【注1】：

n / c = x（其中x为整数）
n * a / c = x * a （a也为整数）
即n能被c整除，n*a也能

【注2】

m ÷ c = x1（x1为整数）
n * a ÷ c = x * a（x，a为整数）
m ÷ c - n * a ÷ c = x1 - x * a （结果也为整数）

【注3】

注意公约数与最大公约数之间的区别。
如第一段证明，我们所求只能的c是n和b的公约数，但不能下定论说c是最大公约数。

2.求解算法

有了上面的证明过程，我相信你已经明白了算法的内含。

```java
//递归实现
 int gcd(int m, int n) {
      return b == 0 ? m : gcd(n, m % n);
 }
```

因为gcd(m,n)=gcd(n,b)啊，
所以我们求m和n的公约数，只需要求n和b（即m % n）的即可
那什么时候是个头呢？ 应该是 b = 0的时候，b是个余数，余数都等于0了就是最简状态了，而且下一步0不能被当作分母来运算，这就标志着算法的结束。

而此时最简状态的 m（形参）即是最大公约数。
更深入的理解 当 m % n = 0，（这里的m，n指的是形参）即 m 是 n 的倍数时，此时，n 就是 m 的最大公约数 。（形参中对应m，好绕好绕……）

同样，循环算法也是类似的道理。

> 注：你不必管m ，n谁大谁小，因为有个辗转的过程，他们会自动调整。