欧几里得算法的应用

江苏省常州高级中学 金斌*

2009 年 1 月

摘 要

辗转相除法求两个数的最大公约数是最早被数学家研究的算法之一，并且和数论中如连分数，丢番图方程有着紧密的联系。本文从基本的欧几里得算法谈起，涉及了几个数论问题的解法，并受其思想的启发，研究并解决了了几个看起来与数论不相关的问题。

关键字： 欧几里得算法 辗转相除 连分数 Pell方程

目 录

*mailto:crazyb0y@126.com

1 欧几里得算法

这一节主要是简要介绍一些关于欧几里得算法的基础，如果您对其足够了解，请直接跳过

1.1 最大公约数

最大公约数问题是最早被研究的算法问题之一了[1]，并且是信息学竞赛中能涉及到的很多数论内容，比如模线性方程，模线性方程组的基础。欧几里得算法 ***(Euclidean algorithm[2])*** ，即大部分选手所知的**“辗转相除法”**，其核心在于不断将两数规模变小，最后实现对数时间[3]内把问题变换到能直接判定解的规模。

1.1.1 辗转相除

由于是数论的最基本算法，所以这里我直接给出伪代码：

Algorithm 1 Euclidean Algorithm

Input: $x \geq 0 \wedge y \geq 0$

Output: the greatest common divisor of x and y

```
while y ≠ 0 do
    t ⇐ x mod y
    x ⇐ y
    y ⇐ t
end while
return x
```

下面简要说明一下这个算法的正确性，对于任意两个数x和y，令$x = y \times a + b$，则对于任意$d \mid x$且$d \mid y$，有$x'd = y'd \times a + b$，$b = (x' - y' \times a) \times d$。所以$d \mid (x \bmod y)$，这说明了最后的结果$ans$一定是$x$和$y$的公约数。类似的对于任何$x$和$y$的公约数$d$，都有$d \mid ans$，这表明了$ans$为最大公约数。

1.1.2 复杂度分析

这里只给出简要的分析，说明其时间复杂度（忽略除法的复杂度）为$O(\log n)$，对于任意两个数a, b满足$a > b$，我们考虑如下这个语句的效果[4]

$$a \Leftarrow a \bmod b$$

- 如果$b \leq \frac{a}{2}$，则$a \bmod b < b \leq \frac{a}{2}$
- 如果$b > \frac{a}{2}$，则$a \bmod b \leq a - b < a - \frac{a}{2} = \frac{a}{2}$。

可以看出，每次这个操作都会将a减半，所以最后复杂度是$O(\log n)$。

[1] 初见于公元前300年的几何原本

[2] http://en.wikipedia.org/wiki/Euclidean_algorithm

[3] 假设所有基本的算术运算都是$O(1)$的

[4] 我们忽略中间可能的模零的错误

1.2 拓展欧几里得算法

拓展欧几里得算法（***Extended Euclidean Algorithm***[5]）是基于欧几里得算法而来解一类特殊的线性丢番图方程（***Diophantine equation***[6]）

$$ax + by = \gcd(a, b)$$

而当$\gcd(a, b) = 1$，即a, b互质的时候，这个方程的解实际上就对应了a关于模b的逆元。因此，这个算法成为解模线性方程和模线性方程组的基础，而在信息学竞赛中广泛传播。

还是先给出伪代码：

Algorithm 2 Extended Euclidean Algorithm

Input: $a,\ b$
Output: a solution to $ax + by = \gcd(a, b)$

```
1: function extended_gcd(a,b)
2: if b = 0 then
3:     return (1, 0)
4: else
5:     (x, y) ⇐ extended_gcd(b, a mod b)
6:     return (y, x − (a div b) × y)
7: end if
```

下边做个简要分析，这是个修改版的辗转相除算法，是可以算出最后的$\gcd(a, b)$的，考虑第3行，当当前情况是$a = \gcd(a, b), b = 0$时，则很容易看出$x = 1, y = 0$是一组解，所以在第5行的时候我们可以假定$x \times b + y \times (a \bmod b) = \gcd(a, b)$。对上式展开可得：

$$\begin{aligned}\gcd(a, b) &= xb + y \times (a - (a \operatorname{div} b) \times b) \\ &= xb + ya - y \times (a \operatorname{div} b) \times b \\ &= y \times a + (x - (a \operatorname{div} b) \times y) \times b\end{aligned}$$

所以这个算法在正向计算出$\gcd(a, b)$之后，返回时就计算出了解，而复杂度显然和原先的一样。

乍看似乎欧几里得算法毫无神奇之处，但是当你发现了其逐步缩小规模的本质之后，这种思想往往在做某些题目（包括数学计算机...）会给你带来耳目一心的感觉。

2 “另类”欧几里得算法

容易想像，任何带“取模”操作并符合偏序的两个“数”[7]都可以运用欧几里得算法。比如下面即将提到的多项式。

而我们似乎可以做得更多，比如将取模放大到矩阵的初等变换，甚至是二维向量的操作。

[5] http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm
[6] http://en.wikipedia.org/wiki/Diophantine_equation
[7] 精确的说是一个环，这里所指的是欧几里得环(http://en.wikipedia.org/wiki/Euclidean_domain)

2.1 多项式

多项式的最大公因式在在计算机竞赛中用处不多，把它作为这节的开篇仅仅是概念上的意义。

这里给出一个不太正规的多项式“整除”的定义：即求一个新的多项式使“余数”次数小于“除数”。然后就可以在此基础上定义“取模”乃至后面的欧几里得过程。下面是一个例子。

$$x^4 + 8x^3 + 12x^2 + 17^x + 6 \tag{1}$$
$$x^4 - 4x^3 + 4x^2 - 3x + 14 \tag{2}$$
$$x^3 + \frac{2}{3}x^2 + \frac{5}{3}x - \frac{2}{3} \tag{3}$$
$$x^2 + x + 2 \tag{4}$$
$$0 \tag{5}$$

即可得到1与2的最大公因式为4。

多项式的最大公因式代表了两个多项式方程的公共根。

2.2 模意义下的矩阵行列式

题目来源 经典问题，一个例题是*SPOJ Find The Determinant III*[8][9]

题目大意 给定一个$n \times n$的整数矩阵A，求$\mathrm{Det}(A) \bmod M$，其中M为正整数。

分析与解答 这个题目在很多地方都会用到，当用行列式计算生成树个数的时候，大部分情况下会只求答案模一个数的值。如果是模素数或者模一个Square Free Number，则可以通过求逆元或者再用中国剩余定理解决，但是模一般的数用普通的高斯消元会有种种问题。

这个题目的解法显然还是基于高斯消元的，所以建议还不是很熟悉的读者停下先复习一下。

首先有两种很直接的解法：

1. 显然一种方法是一个高精度分数类，然后.....就可以了。但是由于高精度运算并没有在$O(1)$内完成，最后效果很不好，大概复杂度在$O(n^4)$左右。

2. 出题人的最初想法：首先把问题转化成模一个素数的幂次，比如a^p，最后用中国剩余定理合并。在具体实现的时候，如果当前需要可供选择的主元都是a的倍数，则先把这一列全部 $\mathrm{div}\ a$，然后算行列式 $\bmod a^{p-1}$的值之后再乘上a。这个方法比较复杂，最后的复杂度是$O(n^3 \log C)$。

以上两种方法如果实现得好的话，都可以解决本题，但是看起来似乎都不够优美。

根据代数知识，如果将矩阵的任意两行交换，则行列式变负，如果将任意一行加减另一行的倍数，则行列式不变。很容易发现，整数矩阵的初等变换和“取模”有惊人的相似：考虑矩阵$A_{n,n}$及i, j, k，我们可以通过$A_i \Leftarrow A_i - A_j \times (A_{i,k} \ \mathrm{div}\ A_{j,k})$这样一个操作实现$A_{i,k} \Leftarrow A_{i,k} \bmod A_{j,k}$而不改变行列式值。

[8]提交地址：http://www.spoj.pl/problems/DETER3/

[9]另一个是Timus 1627: Join http://acm.timus.ru/problem.aspx?space=1&num=1627

想到这里算法就很直接了，因为有了“取模”，我们就可以在$O(n \log C)$时间内将同一列中任意两个数中一个消成0，配合交换行操作即可完成消元过程。

下面是伪代码：

Algorithm 3 Determinant mod M

Input: $n \times n$ matrix A, M
Output: Det(A) mod M

```
for all element A_{i,j} in A do
    A_{i,j} ⇐ A_{i,j} mod M
end for
ans ⇐ 1
for i = 1 → n do
    for j = i + 1 → n do
        while A_{j,i} ≠ 0 do
            t ⇐ A_{i,i} div A_{j,i}
            A_i ⇐ A_i − A_j × t
            swap A_i and A_j
            ans ⇐ −ans
        end while
    end for
    if A_{i,i} = 0 then
        return 0
    end if
    ans ⇐ ans × A_{i,i}
end for
return ans
```

想出这个算法的关键在于能洞悉整数除法和矩阵的初等变换之间的关系，巧妙的把消元这个过程对应到欧几里得过程，并最后在$O(n^3 \log C)$的时间内解决这题。

值得一提的是当M为奇数的时候，欧几里得算法的二进制实现[10]也可以运用到这道题目，思想与刚才叙述的算法类似，而且效率更高（当然复杂度还是没变）。

2.3 二维欧几里得算法

题目来源 *SPb IFMO* 的*WiNGeR*提供的题目，来自 *Maloyaroslavets Summer Camp 2008*

题目大意 给出两个向量a, b，求整数x, y满足不同时为0，使$|ax + by|$最小。

分析与解答 首先观察一个很特殊的情况：就是a及b共线。很容易发现，最后的答案实际上就是gcd($|a|, |b|$)，这启示我们，可能这道题目和欧几里得算法有关。

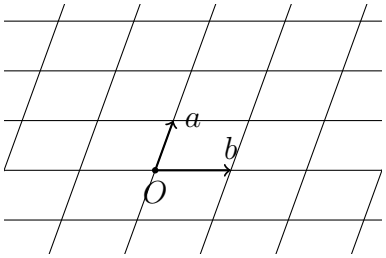
考虑一个极端的情况，如下图，如果a与b的夹角很小，足以忽略不计，那么实际上答案还是gcd($|a|, |b|$)。

[10] http://en.wikipedia.org/wiki/Binary_GCD_algorithm

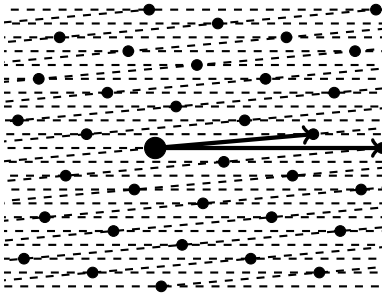
这或许也可以说明，本题需要用欧几里得算法解决。

为了方便讨论，我们约定$a \cdot b \geq 0$，即a与b同向，且$|a| \leq |b|$。

考虑如下一张图，按a和b作为坐标轴画出的一幅图，很显然，这道题目是要求该图中除了O之外的所有格点到O的最小值。



如果你觉得这样子过于简单了？请看下图：



所以直观感受应该是，夹角越小问题越“不平凡”。

所以现在再考虑一个特殊情况，设a与b的夹角α大于$\frac{\pi}{3}$，即$\cos\alpha < \frac{1}{2}$。

结论1. 如果a, b夹角大于$\frac{\pi}{3}$，则要求的答案就是$\min(|a|, |b|)$。

Proof. 令$p = |a|,\ q = |b|$，不妨设$p < q$，则

$$\begin{aligned} |ax + by| &= \sqrt{(px)^2 + (qy)^2 - 2pxqy\cos\alpha} \\ &\geq \sqrt{|px|^2 + |qy|^2 - 2|px||qy|\cos\alpha} \\ &\geq \sqrt{|px|^2 + |qy|^2 - |px||qy|} \\ &\geq \sqrt{(|px| - |qy|)^2 + |px||qy|} \end{aligned}$$

- 若$x = 0$，则$y \neq 0$，$(|px| - |qy|)^2 = |qy|^2 \geq q^2 \geq p^2$
- 若$y = 0$，则$x \neq 0$，$(|px| - |qy|)^2 = |px|^2 \geq p^2$
- 否则$|px||qy| \geq |p||q| \geq p^2$

□

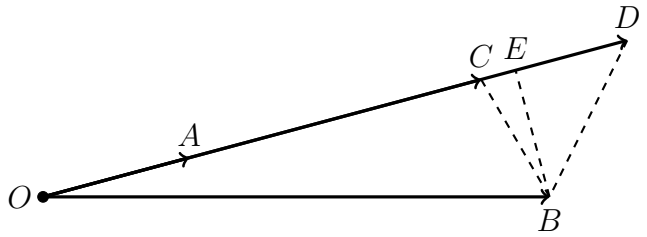
现在我们的目标就是不断通过变换两边增加其夹角。

结论2. (a, b)所对应的答案，和$(a, b + ka)$一致，其中k为整数。

Proof. 令$|ax+by|$为所求答案，则$|a(x-ky)+(b+ka)y|$也是，即(x,y)对应着答案$(x-ky,y)$。并且若$(x,y)\neq(0,0)$，则$(x-ky,y)\neq(0,0)$。同理可证相反方向。 □

这提示了我们变换的方向，但是我们还有很多细节考虑，比如k如何选取，如何保证有限步内结束？

这里我们重申上面的约定，$|a|\leq|b|$。下面请看下图



令$\overrightarrow{OA}=a,\overrightarrow{OB}=b$，$BE\perp OA$，且$\overrightarrow{OC}=k\overrightarrow{OA},\overrightarrow{OD}=(k+1)\overrightarrow{OA}$并且$E$在线段$CD$上。

如果此时$|OA|>|OE|$，则容易证明$\angle OAB>\angle AOB$（做A关于E的对称点A'易得）。

否则如果$|CE|<|DE|$，则用(CB,OA)代替(OA,OB)，否则用(DB,OA)代替(OA,OB)。

结论3. $\max(\angle BCE,\angle BDE)>\angle AOB$

Proof. 直接可以证明$\angle DCB=\angle AOB+\angle CBO>\angle AOB$。 □

综上，这种变换保持着向量夹角增加。

所以一个基于欧几里得算法的算法就成型了，每次用二次函数计算上面的k值，之后进行变换，直到夹角超过$\frac{\pi}{3}$。

限于知识水平，这个算法的确切复杂度我不能给出，但是根据实验结果，基本能在几步之内结束，应该是$O(\log C)$级别的。

2.4 小结

欧几里得算法本质就是通过一个特定的运算，将规模变小，最后将问题变成一个平凡的问题。到目前为止，我们讨论的所有运算还都是基于“取模”的层次，在最后一道例题里，我们甚至自己构造了一种取模的方法，这样可以套用欧几里得的例子很少很少，更多的时候，我们只是在不经意间推得了一个将问题规模变小的式子，但是没有到能直接算出的地步，这是我们就可以尝试着去反复这个过程。更多的例子请参见第四节。

3 连分数和Pell方程

3.1 连分数

本节很多内容参考了杨哲同学2006年的集训队作业《一类分数问题的研究》[3]中第四章简单连分数和最优渐进分数

一个实数ω可以用一个正整数数列$[a_0;a_1,a_2,\cdots]$表示，其中

$$\omega=a_0+\cfrac{1}{a_1+\cfrac{1}{a_2+\cfrac{1}{a_3+\cdots}}}$$

这种表示方法称之为连分数(***Continued fraction***[11])表示。
很显然可以按照如下方法计算出ω对应的$[a_0; a_1, a_2, \cdots]$:

$$\omega_0 \Leftarrow \omega,\ a_n = \lfloor \omega_n \rfloor,\ \omega_{n+1} = \frac{1}{\omega_n - a_n}$$

观察这个过程不难发现，如果ω为有理数，则这个过程实际上就对应了欧几里得算法，所以对于任意有理数，这个序列总是有限的，而对于无理数，这个序列则一定是无限的。更进一步，如果序列有些的话禁止最后一个数a_n为1，则这个序列是显然唯一确定的。

如果给定了任意一个有限的连分数$[a_0; a_1, a_2, \cdots, a_n]$，我们显然可以通过直接按照表达式计算出原分数，这里我们只考虑一个简单情况，还原$a + \frac{1}{p/q}$:

$$a + \frac{1}{\frac{p}{q}} = \frac{ap + q}{p} = \frac{p'}{q'}$$

这里有几点可以观察出:

- 如果p,q互质，则p',q'也互质，所以这个过程只要简单地写成乘法就行
- 这个方法不能直接用来求有理数来逼近特定的无理数，因为每次都要重新计算
- 这个过程实际上是一个矩阵乘法过程:

$$\begin{bmatrix} p' \\ q' \end{bmatrix} = \begin{bmatrix} a & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} p \\ q \end{bmatrix}$$

从最后一点可以推出很多有趣的结论，令

$$A_i = \begin{bmatrix} a_i & 1 \\ 1 & 0 \end{bmatrix}$$

则实际上最后可以通过计算

$$A_0 \times A_1 \times \cdots \times A_n \times \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

来求该连分数对应的有理数。由于矩阵乘法具有结合律，所以我们大可以直接从前往后计算，这样第二个问题（即上边列出三点中的第二点）在数学上已经解决了。

但实际上，在这个基础上可以把结论做得更优美，令

$$S_i = A_0 \times A_1 \times \cdots \times A_i = \begin{bmatrix} x & y \\ z & w \end{bmatrix}$$

，则

$$S_{i+1} = S_i \times A_{i+1} = \begin{bmatrix} a_{i+1}x + y & x \\ a_{i+1}z + w & z \end{bmatrix}$$

[11] http://en.wikipedia.org/wiki/Continued_fraction

很容易观察出，分子分母都是独立的线性递归数列！！！而最后乘上的矩阵$\begin{bmatrix}1\\0\end{bmatrix}$即表示，只取最后的$\frac{x_n}{y_n}$为所求分数—细节也做到了完美。

下面是完整的计算方法：

$$p_n = a_n p_{n-1} + p_{n-2}, \quad p_{-1} = 1, \quad p_{-2} = 0$$
$$q_n = a_n q_{n-1} + q_{n-2}, \quad q_{-1} = 0, \quad q_{-2} = 1$$

这里给出更规范的几个定义，并不加证明的给出几个结论[12]：

定义1. *一个实数ω的连分数表示为$[a_0; a_1, a_2, \cdots]$，则称$[a_0; a_1; a_2, \cdots, a_n]$为$\omega$的$n$渐进分数。*

定义2. *称一个分数$\frac{p}{q}$是一个数ω的最优渐进分数，当且仅当在所有分母不大于q的分数中，$|\frac{p}{q} - \omega|$最小。*

结论4. *ω的所有渐进分数都是最优渐进分数。*

这个结论将在下一节介绍Pell方程的时候用到。

3.2 Pell方程

题目来源 经典问题，*SPOJ EQU2*[13]

题目大意 对于给定正整数N，其中N不是完全平方数，求方程

$$x^2 - Ny^2 = 1$$

的最小正整数根。

分析与解答 这题要解的就是***Pell*方程*(Pell's equation*[14]*)***。实际上，任何形如

$$ax^2 + bx^2 + cxy + dx + ey + f = 0$$

的二元二次丢番图方程都可以通过解Pell方程解决。

这里直接引用 Lagrange 和 Euler 发现的结论。

令$\frac{p_k}{q_k}$为$\sqrt{N}$的渐进分数，则

结论5 (Lagrange)**.** *存在$n > 0$，使得*

$$p_n^2 - Nq_n^2 = 1$$

结论6 (Euler)**.** *在结论5中提到的所有n中，最小的n代表了这个Pell方程对应的最小解(p_n, q_n)。*

所以，我们可以通过计算序列$\{(p_n, q_n)\}$直到我们找到答案。

据此可以直接写出第一个伪代码（算法4）。

遗憾的是，这个算法由于浮点数的精度误差，最后可能会出错甚至陷入死循环，我们需要发掘更多的信息。

[12] 证明请参照杨哲的作业

[13] http://www.spoj.pl/problems/EQU2/

[14] http://mathworld.wolfram.com/PellEquation.html

Algorithm 4 Naïve Implementation of solving Pell's Equation

Input: a postive integer N, where N is not a perfect square number

Output: the minimum non-trivial solution of equation $x^2 - Ny^2 = 1$。

$\omega_{-1} \Leftarrow \sqrt{N}$
$p_{-1} \Leftarrow 1, \quad p_{-2} \Leftarrow 0$
$q_{-1} \Leftarrow 0, \quad q_{-2} \Leftarrow 1$
for $i = 0 \rightarrow \infty$ **do**
  $a_i \Leftarrow \lfloor \omega_{i-1} \rfloor$
  $\omega_i \Leftarrow \frac{1}{\omega_{i-1} - a_i}$
  $p_i \Leftarrow p_{i-1} \times a_i + p_{i-2}$
  $q_i \Leftarrow q_{i-1} \times a_i + q_{i-2}$
  if $p_i^2 - Nq_i^2 = 1$ **then**
    return (p_i, q_i)
  end if
end for

结论7 (Euler, 1765)**.** 存在整数g_n和h_n使得

$$\omega_n = \frac{g_n + \sqrt{N}}{h_n}$$

即这个算法执行到任何时候的ω都可以写成如上形式（二次无理数形式[15]）。
那么将$\omega_n = \frac{1}{\omega_{n-1} - a_n}$带入上式：

$$\begin{aligned}
\omega_n =& \frac{1}{\omega_{n-1} - a_n} \\
=& \frac{1}{\frac{g_{n-1} + \sqrt{N}}{h_{n-1}} - a_n} \\
=& \frac{h_{n-1}}{\sqrt{N} + g_{n-1} - a_n h_{n-1}} \\
=& \frac{h_{n-1}(\sqrt{N} - g_{n-1} + a_n h_{n-1})}{(\sqrt{N} + g_{n-1} - a_n h_{n-1})(\sqrt{N} - g_{n-1} + a_n h_{n-1})} \\
=& \frac{\sqrt{N} - g_{n-1} + a_n h_{n-1}}{\frac{N - (-g_{n-1} + a_n h_{n-1})^2}{h_{n-1}}}
\end{aligned}$$

即可得出g_n和h_n的递推关系：

$$\begin{cases} g_{-1} = 0, h_{-1} = 1 \\ g_n = -g_{n-1} + a_n h_{n-1}, h_n = \frac{N - g_n^2}{h_{n-1}} \end{cases}$$

[15] http://en.wikipedia.org/wiki/Quadratic_irrational

更进一步，a_n其实也可以不要$\sqrt{N}$参与而直接算出：

$$a_n = \lfloor \omega_{n-1} \rfloor = \left\lfloor \frac{g_{n-1} + \sqrt{N}}{h_{n-1}} \right\rfloor$$
$$= \left\lfloor \frac{g_{n-1} + \lfloor \sqrt{N} \rfloor}{h_{n-1}} \right\rfloor = \left\lfloor \frac{g_{n-1} + a_0}{h_{n-1}} \right\rfloor$$

经过了上边的分析，我们可以写出一个在运算过程中完全不包含浮点误差的程序，完美解决了这道题目：

Algorithm 5 Advanced Algorithm For Pell's Equation

Input: a postive integer N, where N is not a perfect square number
Output: the minimum non-trivial solution of equation $x^2 - Ny^2 = 1$。

$p_{-1} \Leftarrow 1, \quad p_{-2} \Leftarrow 0$
$q_{-1} \Leftarrow 0, \quad q_{-2} \Leftarrow 1$
$a_0 \Leftarrow \lfloor \sqrt{N} \rfloor$ // we can implement it without float-number operations
$g_{-1} \Leftarrow 0, \quad h_{-1} \Leftarrow 1$
for $i = 0 \to \infty$ **do**
  $g_i \Leftarrow -g_{i-1} + a_i h_{i-1}$
  $h_i \Leftarrow \frac{N - g_i^2}{h_{i-1}}$
  $a_{i+1} \Leftarrow \lfloor \frac{g_i + a_0}{h_i} \rfloor$
  $p_i \Leftarrow a_i p_{i-1} + p_{i-2}$
  $q_i \Leftarrow a_i q_{i-1} + q_{i-2}$
  if $p_i^2 - N q_i^2 = 1$ **then**
    return (p_i, q_i)
  end if
end for

4 另外几个其他的运用

4.1 两分数间分母最小的分数

题目来源 杨哲作业中第三章

题目大意 满足$\frac{a}{b} < \frac{p}{q} < \frac{c}{d}$的分数$\frac{p}{q}$，分母最小的是多少。

分析与解答 杨哲作业[3]中的解法已经很完整了，这里简要叙述一遍

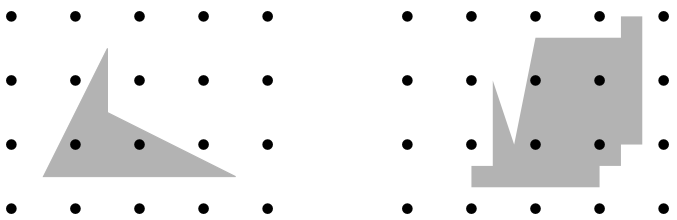
首先如果存在整数n满足$\frac{a}{b} < n < \frac{c}{d}$，则问题直接就可以被直接解决，否则可以直接将问题转化成求$\frac{p}{q}$满足$0 \le \frac{a \bmod b}{b} < \frac{p}{q} < \frac{c \bmod d}{d} < 1$。可以证明满足条件的$q$唯一，且此时$p$也最小，所以可以直接取倒数把问题转化成$\frac{d}{c \bmod d} < \frac{q}{p} < \frac{b}{a \bmod b}$。

上式显然就是辗转相除，我们通过取倒数问题不变的方式，意外地踏入了取模这个阶段，从而构造出了算法。

4.2 有理点组成的简单多边形包含的格点数

题目来源 TopCoder *SRM 410 WifiPlanet*[16]

题目大意 在整数格点的平面上有一个简单多边形（顶点坐标均为有理数），问其内部有多少格点。（题目保证不会有格点出现在边界上）



分析与解答 这是那次比赛的压轴题，比赛现场无人正确解答。

联系普通多边形面积的做法，很容易想到这道题目需要对多边形进行剖分。但是普通的三角剖分现在在这里不适用（至少解法不会很简单），因为考虑一个纯粹的三角形内部的格点数是很复杂的一件事。所以想到梯形剖分应该是相对直接的一件事。

形式化些，我们需要计算一条线段下方有多少格点（感谢题设，边上不会有点，问题简单了些）。即给出$(x_1, y_1), (x_2, y_2), x_1 < x_2$，问格点$(x, y)$的个数，满足

$$x_1 \le x < x_2, \quad 0 \le y < \frac{y_1(x_2 - x) + y_2(x - x_1)}{x_2 - x_1}$$

由于x, y都是整数，所以问题可以转化成求$(x_1, y_1), (x_2, y_2)$这条线段下方的节点数，其中$x_1, x_2 \in \mathbb{Z}, y_1, y_2 \in \mathbb{Q}$。

而实际上求这个数目等价于求如下这个式子的值:

$$\sum_{i=0}^{n-1} \left\lfloor \frac{a + di}{m} \right\rfloor$$

问题到了这里就有了两种解法

- 第一种是比较容易想到的，但是复杂度为$O(\sqrt{n})$，且需要考虑太多的细节。
- 另外一个算法是官方做法，也就是欧几里得算法变种，复杂度为$O(\log n)$。

首先介绍$O(\sqrt{n})$的算法。令$A_i = \lfloor \frac{a+di}{m} \rfloor$，考虑$A_i - A_{i-1}$，有两种可能:

- 如果$(a + d(i-1)) \bmod m + d \bmod m >= m$，则$A_i - A_{i-1} = 1 + \lfloor \frac{d}{m} \rfloor$。
- 如果$(a + d(i-1)) \bmod m + d \bmod m < m$，则$A_i - A_{i-1} = \lfloor \frac{d}{m} \rfloor$。

想要快速的完成计算，我们最好要能尽可能得减少$d \bmod m$，这样在之后的计算中，我们可以一下计算个数为$\frac{m}{d}$左右的值。一种方法是，选定一个数c，然后把$0 \ldots n-1$这n个数按模c的值归类，这样每类就可以直接加$dc \bmod m$了。这里有一种实现，令$c \in [1, C]$，且$dc \bmod m$最小，假定$dk \bmod m$是当$k \in [1, n]$时是均匀分布的，那么“期望”的$dc \bmod m < \frac{m}{C}$。所以我们对于所有$i \in [0, c)$，直接枚举累和A_{i+jc}，这个过程实际上的复杂度是$O(C +$

Algorithm 6 $O(\sqrt{n})$ implementation

Input: a, d, m, n
Output: $\sum_{i=0}^{n-1} \lfloor \frac{a+di}{m} \rfloor$

```
C ⇐ √n, c = 1
for i = 1 → C do
    if di mod m < dc mod m then
        c ⇐ i
    end if
end for
for i = 0 → c − 1 do
    j ⇐ i
    while j < n do
        let t be the maximum number that (dj mod m) + (dc mod m) × t < m
        apply changes , add the answer to ret
        j ⇐ j + t
    end while
end for
return ret
```

n/C)，当$C = \sqrt{n}$的时候，复杂度到最小，为$O(\sqrt{n})$。这里给出伪代码，但是容易想像，这个算法存在着很多细节问题，很多地方要认真处理才能解决这题。

这道题目的n, m设到了10^9，这个算法才勉强通过了测试，但如果n的规模更大，我们也就无能为力了，所以，我们需要寻求更加高效简洁优美的算法。

考虑式子

$$\sum_{i=0}^{n-1} \left\lfloor \frac{a+di}{m} \right\rfloor \tag{6}$$

则我们可以做出几点约定：

- $0 \le a < m$, 否则直接可以把$\lfloor \frac{a}{m} \rfloor$提到外面。从直观感觉看，这个变换实际上是把这个“梯形”，变成了一个“三角形”。
- $0 < d < m$，通过和上边式子一样的方法，可以保证这一点。注意，首先当$d = 0$时，这个数列变成常数数列就可以直接计算。而这个变换的几何本质实际上可以理解成如果另一条直角边更长，那就可以通过这个变换使得当前直角边为最长边。

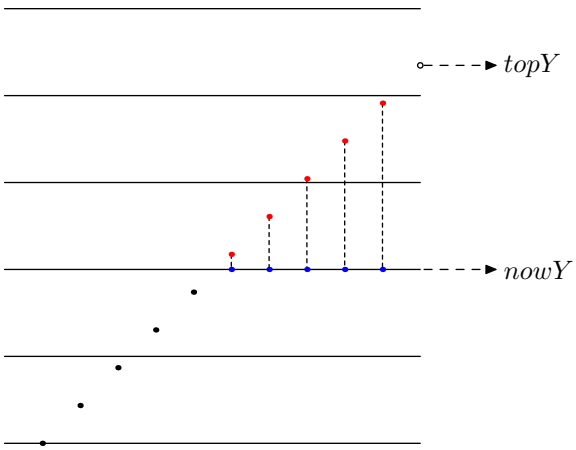
以上第二个观察提示我们，我们只要能在互换两条“直角边”，就可以顺利的运用欧几里得算法了。

我们换个角度来看表达式6，可以这么理解，有一串点$(i, a + di)$，然后每个点向正下方做射线。令$l = \lfloor \frac{a+dn}{m} \rfloor$，则一共有$l$条水平的直线$y = km$，则表达式6的意思上就是所有射线与所有直线的交点总数。

我们再换个角度，考虑蓝点，其实也可以理解成每条水平直线上方的点数之和（红色点），即

$$\left\lfloor \frac{topY - nowY}{d} \right\rfloor$$

[16] writer是bmerry，引用该题目仅为教育目的



其中$topY$为$a+dn$，$nowY$为当前的直线高度。

想到这里就能直接写出公式了

$$\sum_{i=0}^{n-1}\left\lfloor\frac{a+di}{m}\right\rfloor=\sum_{k=0}^{l-1}\left\lfloor\frac{(a+dn)\bmod m+mk}{d}\right\rfloor \tag{7}$$

根据7，我们已经可以写出一个完整的程序了。回到刚才的“直观分析”，实际上有了这个式子，我们就可以不断讲两条“直角边”取模之后对调，显然这个复杂度是$O(\log n)$的，

至此，问题得到解决。

5 总结

欧几里得虽然被大部分信息学竞赛选手所熟知，但是在实际的运用中却很难被想到，实际上我们只要能挑出思维的定势，从一些并不相关的等式把问题简单化，最后都是能到达我们的目的的。本次论文主要从思想方面介绍了欧几里得算法的几个运用，可以算是一种开拓视野。而最后可以发现，所有的题目都是纯数学题，也能在一定程度上表明数学思想在计算机竞赛中的作用。

参考文献

[1] http://www.wikipedia.org/ 维基百科

[2] http://mathworld.wolfram.com/ MathWorld

[3] 杨哲，《一类分数问题的研究》

感谢

感谢我的老师常州中学的曹文老师，没有他就没有现在的我

感谢SPb IFMO的Vladislav Isenbaev(WiNGeR)提供题目，感谢上海交通大学的吴卓杰和中国科技大学的朱泓丞的帮助。