补充

数论初步之扩展的gcd算法

songyou@buaa.edu.cn

扩展欧几里得算法

- 给定 2 个不全为零的整数 a, b，令 gcd(a, b) 表示整数 a 和 b 的最大公约数
- x|y 表示 x 可以整除 y，即 y 是 x 的倍数
- 扩展欧几里得算法用于快速求解 gcd(a, b)，同时可以计算出 x_0 和 y_0，使得 $ax_0 + by_0 = gcd(a, b)$

扩展欧几里得算法

- 二元一次方程： $ax + by = c$，其中 $a, b, c, x, y \in Z$

- **裴蜀定理**
  - 结论一：设 a, b 是不全为零的整数，对任意整数 x, y，满足 $ax+by$ 能被 $gcd(a,b)$ 整除，即 $gcd(a, b) \mid ax + by$

  (即， $0 < gcd(a, b) \leq ax + by$；即， $k*gcd(a, b) = ax + by$，其中$1 \leq k$)

  - 结论二：存在整数 x, y，使得 $gcd(a, b) = ax + by$

  (即，$gcd(a, b) \leq ax + by$ 的等号可取得，即 gcd 是 a 和 b 的最小的正线性组合)

扩展欧几里得算法

- 裴蜀定理结论一的证明
  - 结论一：设 a, b 是不全为零的整数，对任意整数 x, y，满足 ax+by 能被 gcd(a,b) 整除，即 gcd(a, b) | ax + by

  （即， $0 < gcd(a, b) \le ax + by$；即， $k*gcd(a, b) = ax + by$，其中$1 \le k$）

证明结论一：

- 由于 gcd(a, b) | a, gcd(a, b) | b，因此有 gcd(a, b) | ax, gcd(a, b) | by
- 因此 gcd(a, b) | ax + by

显然，ax + by 所能得到的最小正整数不小于 gcd(a, b)

扩展欧几里得算法

结论二：存在整数 x, y，使得 gcd(a, b) = ax + by（即，等号可取得）

- 结论二的证明
  - 不妨设 $a \ge b > 0$，回顾欧几里得算法求 gcd(a, b) 的过程，设一共进行 r 次，即

$gcd(a_0, b_0) = gcd(b_0, a_0\%b_0) = gcd(a_1, b_1) = gcd(a_2, b_2) = \ldots = gcd(a_r, b_r)$

  - 令 $a_0 = a$, $b_0 = b$，则有 $a_i = b_{i-1}$, $b_i = a_{i-1} \% b_{i-1}$, 从假设和计算过程易知 $a_i \ge b_i$
  - 考虑构造 r + 1 个二元一次方程，方程组右侧均相等，设其为 g（即 gcd），结论二本质即为求解第 0 个方程（即第 0 个方程有整数解）

$$\begin{cases} a_0x_0 + b_0y_0 & = gcd(a_0, b_0) \\ a_1x_1 + b_1y_1 & = gcd(a_1, b_1) \\ & \vdots \\ a_rx_r + b_ry_r & = gcd(a_r, b_r) \end{cases} \iff \begin{cases} a_0x_0 + b_0y_0 & = g \\ a_1x_1 + b_1y_1 & = g \\ & \vdots \\ a_rx_r + b_ry_r & = g \end{cases}$$

  - 由于计算过程只进行 r 次，易知 $b_r = 0$（此时不用再辗转相除，递归中止）
  - 对于第 r 个方程，取 $x_r = 1$, $y_r = 0$ 即有: $a_rx_r + b_ry_r = a_r = g = gcd(a_r, b_r)$

扩展欧几里得算法

- 下面进行归纳法证明
  - 假设已知 x_i, y_i 满足第 i 个方程：$a_i x_i + b_i y_i = g$
  - 回顾 a_i, b_i 的定义，再根据模运算的性质可知：

$$\boxed{\begin{cases} a_i = b_{i-1} \\ b_i = a_{i-1} \% b_{i-1} \end{cases}} \Rightarrow \begin{cases} a_i = b_{i-1} \\ b_i = a_{i-1} - \lfloor \frac{a_{i-1}}{b_{i-1}} \rfloor \times b_{i-1} \end{cases}$$

  - 将其带入第 i 个方程，然后进行化简可得：

$$b_{i-1} x_i + (a_{i-1} - \lfloor \frac{a_{i-1}}{b_{i-1}} \rfloor \times b_{i-1}) y_i = g$$

$$\Downarrow$$

$$a_{i-1} \underline{y_i} + b_{i-1} \underline{(x_i - \lfloor \frac{a_{i-1}}{b_{i-1}} \rfloor \times y_i)} = g$$

→ 这不就是第 i-1 个方程的系数吗！

$$\begin{cases} a_0 x_0 + b_0 y_0 & = g \\ a_1 x_1 + b_1 y_1 & = g \\ & \vdots \\ a_r x_r + b_r y_r & = g \end{cases}$$

设一共进行 r 次，即

$\gcd(a_0, b_0) = \gcd(b_0, a_0 \% b_0)$

$= \gcd(a_1, b_1) = \gcd(a_2, b_2)$

$= \ldots = \gcd(a_r, b_r)$

令 $a_0 = a$, $b_0 = b$，则有

$\boxed{a_i = b_{i-1},\ b_i = a_{i-1} \% b_{i-1}}$

扩展欧几里得算法

$$a_{i-1}\underline{y_i} + b_{i-1}\underline{(x_i - \lfloor \frac{a_{i-1}}{b_{i-1}} \rfloor \times y_i)} = g$$

- 观察化简后的式子容易发现可以作如下的对应使得第 i - 1 个方程成立：

$$\begin{cases} x_{i-1} = y_i \\ y_{i-1} = x_i - \lfloor \frac{a_{i-1}}{b_{i-1}} \rfloor \times y_i \end{cases}$$

- 因此在已知 $x_r = 1, y_r = 0$ 的情况下可以逆推得到 x_0, y_0 使得第 0 个方程 $a_0x_0 + b_0y_0 = g$ 成立
- 结论二证毕

$$\begin{cases} a_0x_0 + b_0y_0 & = g \\ a_1x_1 + b_1y_1 & = g \\ & \vdots \\ a_rx_r + b_ry_r & = g \end{cases}$$

设一共进行 r 次，即

$\gcd(a_0, b_0) = \gcd(b_0, a_0\%b_0)$

$= \gcd(a_1, b_1) = \gcd(a_2, b_2)$

$= \ldots = \gcd(a_r, b_r)$

令 $a_0 = a, b_0 = b$，则有

$a_i = b_{i-1}, b_i = a_{i-1} \% b_{i-1}$

扩展欧几里得算法

- 容易发现在证明过程中已经求解出一个可行的 x, y 使得 ax + by = gcd(a, b)，这种方法由于基于欧几里得算法，因此被称作扩展欧几里得算法。
- 一种可行的代码实现如下：

```
#define LL long long
int Exgcd(int a, int b, LL *x, LL *y) //songyou
{
    if(b==0)
    {
        *x=1, *y=0;
        return a;
    }
    else
    {
        LL xx, yy;
        int d = Exgcd(b, a%b, &xx, &yy);
        *x=yy, *y=xx-(a/b)*yy;
        return d;
    }
}
```

a = 99, b = 78
gcd(a, b) = ?

a	b	$\lfloor a/b \rfloor$	d	x	y
99	78	1	3	−11	14
78	21	3	3	3	−11
21	15	1	3	−2	3
15	6	2	3	1	−2
6	3	2	3	0	1
3	0	—	3	1	0

gcd(a, b) = 3 = 99*(-11) + 78*(14)

$$\begin{cases} x_{i-1} = y_i \\ y_{i-1} = x_i - \lfloor \frac{a_{i-1}}{b_{i-1}} \rfloor \times y_i \end{cases}$$

扩展欧几里得算法

- 扩展欧几里得算法与欧几里得算法时间复杂度相同，下面对其时间复杂度做个简单的估计。
- 考虑证明中的方程组有：

$$a_0 \geq b_0 = a_1 \geq b_1 \geq \cdots \geq b_{r-1} = a_r \geq b_r = 0$$

- 下面证明 $a_i > 2*(a_i\%b_i)$
  - 若 $a_i < 2*b_i$, $a_i\%b_i = a_i - b_i < a_i - a_i/2 = a_i/2$
  - 若 $a_i \geq 2*b_i$, 由于取模的性质 $a_i\%b_i < b_i \leq a_i/2$
  - 证毕
- 因此有 $a_i \geq 2b_{i+1}$，易算得 $r = O(\log(a_0))$

$a_0 \geq 2a_2 \geq 2^2a_4 \geq \ldots \geq 2^{r/2}a_r \geq 2^{r/2}a_{r+1} = 2^{r/2}g \rightarrow r = O(\log(a_0))$

($a_0 \geq 2b_1 = 2a_2$，后面以此类推)

$$\begin{cases} a_0x_0 + b_0y_0 &= g \\ a_1x_1 + b_1y_1 &= g \\ &\vdots \\ a_rx_r + b_ry_r &= g \end{cases}$$

设一共进行 r 次，即

$\gcd(a_0, b_0) = \gcd(b_0, a_0\%b_0)$

$= \gcd(a_1, b_1) = \gcd(b_1, a_1\%b_1)$

$= \gcd(a_2, b_2) = \gcd(b_2, a_2\%b_2)$

$= \gcd(a_3, b_3) \ldots = \gcd(a_r, b_r)$

令 $a_0 = a$, $b_0 = b$，则有

$a_i = b_{i-1}$, $b_i = a_{i-1} \% b_{i-1}$

扩展欧几里得算法的重要意义

- 扩展欧几里得算法用于快速求解 gcd(a, b)，同时可以计算出 x 和 y，使得 ax + by = gcd(a, b)

- 这个公式 ax + by = gcd(a, b) 的重要意义在于，当 b = n, gcd(a, n) = 1 时，公式为 ax + ny = 1，Exgcd可以求出 x，x就是 ax = 1 mod n 方程的解，即 x 是 a 模 n 的逆，这是数论中的一个重要算法，在许多领域有广阔的应用。

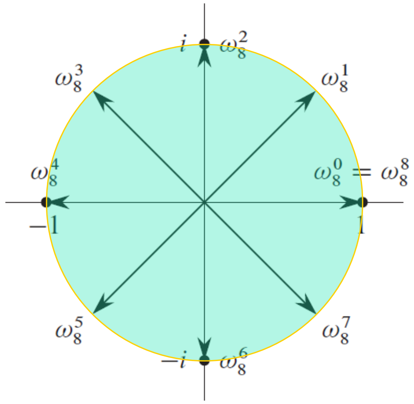
- 数论的更多知识，参考《算法导论》。

```
// ex-gcd 的另一种实现，songyou
struct GCDxy
{
    int d, x, y;
} dxy;

struct GCDxy gcd(int a, int b)
{
    struct GCDxy gcdxy, gcdxy2;
    if(!b) // b == 0
    {
        gcdxy.d = a;
        gcdxy.x = 1;
        gcdxy.y = 0;
    }
    else
    {
        gcdxy2 = gcd(b, a%b);
        gcdxy.d = gcdxy2.d;
        gcdxy.x = gcdxy2.y;
        gcdxy.y = gcdxy2.x - (a/b)*gcdxy2.y;
    }
    return gcdxy;
}
```

数论知识的广阔应用举例

30.2.1 Complex roots of unity



some properties

(1) $\omega_n^n = \omega_n^0 = 1$

($\omega_n^n = (e^{2\pi i/n})^n = \cos 2\pi + i \sin 2\pi$)

(2) $\omega_n^j \omega_n^k = \omega_n^{j+k} = \omega_n^{(j+k) \bmod n}$

(if $j + k = mn + r$, then

$\omega_n^{j+k} = \omega_n^{mn+r} = \omega_n^{mn} \omega_n^r = (\omega_n^n)^m \omega_n^r = \omega_n^r = \omega_n^{(j+k) \bmod n}$)

群的快读

集合S以及定义在S上的运算，满足性质：

①封闭性，②单位元，③结合律，④逆元。

如，整数及加法构成一个群。

进一步定义，交换群，有限群等。

(3) $\omega_n^{-1} = \omega_n^{n-1}$

n 次单位复根的相关运算，其实就是整数加法的模 n 运算。

(proof by using $\omega_n^k = e^{2\pi ik/n}$, $k = 0, 1, \cdots, n-1$)

致谢

感谢闵家旭博士整理了本部分内容的初稿。