WIKIPEDIA

扩展欧几里得算法

扩展欧几里得算法（英语：Extended Euclidean algorithm）是欧几里得算法（又叫辗转相除法）的扩展。已知整数a、b，扩展欧几里得算法可以在求得a、b的最大公约数的同时，能找到整数x、y（其中一个很可能是负数），使它们满足贝祖等式

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

如果a是负数，可以把问题转化成

$$|a|\,(-x) + by = \gcd(|a|, b)$$ （$|a|$为a的绝对值），然后令$x' = (-x)$。

通常谈到最大公约数时，我们都会提到一个非常基本的事实：给予二个整数**a、b，必存在整数x、y使得ax + by = gcd(a,b)**[1]。

有两个数a,b，对它们进行辗转相除法，可得它们的最大公约数——这是众所周知的。然后，收集辗转相除法中产生的式子，倒回去，可以得到ax+by=gcd(a,b)的整数解。

扩展欧几里得算法可以用来计算模反元素(也叫模逆元)，而模反元素在RSA加密算法中有举足轻重的地位。

目录

例子

用类似辗转相除法，求二元一次不定方程$47x + 30y = 1$的整数解。

- $47 = 30 \times 1 + 17$
- $30 = 17 \times 1 + 13$
- $17 = 13 \times 1 + 4$
- $13 = 4 \times 3 + 1$

然后把它们改写成“余数等于”的形式

- $17 = 47 \times 1 + 30 \times (-1)$ //式1
- $13 = 30 \times 1 + 17 \times (-1)$ //式2
- $4 = 17 \times 1 + 13 \times (-1)$ //式3
- $1 = 13 \times 1 + 4 \times (-3)$

然后把它们“倒回去”

- $1 = 13 \times 1 + 4 \times (-3)$
- $1 = 13 \times 1 + [17 \times 1 + 13 \times (-1)] * (-3)$ //应用式3
- $1 = 17 \times (-3) + 13 \times 4$
- $1 = 17 \times (-3) + [30 \times 1 + 17 \times (-1)] \times 4$//应用式2
- $1 = 30 \times 4 + 17 \times (-7)$
- $1 = 30 \times 4 + [47 \times 1 + 30 \times (-1)] \times (-7)$ //应用式1
- $1 = 47 \times (-7) + 30 \times 11$

得解 $x = -7, y = 11$。

这个过程可以用矩阵表示（其中q表示商，r表示余数）

$$\begin{pmatrix} a \\ b \end{pmatrix} = \prod_{i=0}^{N} \begin{pmatrix} q_i & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} r_{N-1} \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 47 \\ 30 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 4 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 47 & 11 \\ 30 & 7 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -7 & 11 \\ 30 & -47 \end{pmatrix} \begin{pmatrix} 47 \\ 30 \end{pmatrix}$$

或者用初等变换

$$\begin{pmatrix} 47 & 30 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \to \begin{pmatrix} 17 & 30 \\ 1 & 0 \\ -1 & 1 \end{pmatrix} \to \begin{pmatrix} 17 & 13 \\ 1 & -1 \\ -1 & 2 \end{pmatrix} \to \begin{pmatrix} 4 & 13 \\ 2 & -1 \\ -3 & 2 \end{pmatrix} \to \begin{pmatrix} 4 & 1 \\ 2 & -7 \\ -3 & 11 \end{pmatrix} \Rightarrow 1 = 47(-7) + 30(11)$$ [2]

实现

以下是扩展欧几里德算法的Python实现：

```python
def ext_euclid(a, b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, q = ext_euclid(b, a % b) # q = gcd(a, b) = gcd(b, a%b)
        x, y = y, (x - (a // b) * y)
        return x, y, q
```

扩展欧几里得算法C语言实现：

```c
int gcdEx(int a, int b, int *x, int *y)
{
    if(b==0)
    {
        *x = 1,*y = 0;
        return a;
    }
    else
    {
        int r = gcdEx(b, a%b, x, y); /* r = GCD(a, b) = GCD(b, a%b) */
        int t = *x;
        *x = *y;
        *y = t - a/b * *y;
        return r;
    }
}
```

参考资料

1. 沈渊源. 数论轻松游 (PDF). [2017-09-25]（中文（台湾））.
2. 张慧玲. 介绍多项式带余除法的矩阵形式及其应用. 太原大学教育学院学报. 2014, (1): 第103—105页.