

初等数论三大定理和缩系乘法群

前言

这篇和算法没什么关系，纯粹是基础知识。

初等数论三大定理，指将整个初等数论框架支撑起来的三个定理，分别是Fermat-Euler（费马欧拉）定理、Wilson（威尔逊）定理和Chinese-Residue（中国剩余）定理。

其中FE定理说明取模意义下缩系（简化剩余系/缩剩余系）集合的乘法构成群，Wilson定理揭示了模为素数的乘法群的结构，而CR定理阐述了怎样将群和群结合起来，即多素因子模数乘法群的结构问题。

它们三者的本质，都是解释缩系乘法群的结构问题。而研究缩系乘法群的结构，最终结论的形式是：奇素数幂次群结构、2的幂次群结构、CR定理，三个定理作为最终的最高结论。

Fermat-Euler定理

内容

设欧拉函数 $\varphi(n)$ 是0到 $n-1$ 里与 n 互素的数（缩剩余系）的个数，即缩系乘法群的阶。对于缩系中任一元素 a 有：

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

特别地，当 n 是单个素数 p 的时候 $\varphi(p)$ 是 $p-1$ 即：（费马小定理）

$$a^{p-1} \equiv 1 \pmod{p}$$

这其实是群论里的定理。任意一个群，群里任意一个元素，自乘群的阶次，一定会回到单位元。即：元素的阶整除群的阶。

证明也简单：对缩系所有元素同时进行乘法操作，构成缩系元素的一个置换。（也可以采用群论中陪集的方法）

这个定理在数学题或者算法中，一般用于简化幂次。例如快速幂函数。

推广

将研究对象转移到缩系以外。在完系（完全剩余系）中，任一元素 a 有相似结论：

$$a^{t + \varphi\left(\frac{n}{(a^t, n)}\right)} \equiv a^t \pmod{n}$$

对于足够大的整数 t 成立。意思是 a 本身自乘很多次后，也会落入循环中，循环节是 n 去除 a^t 与 n 最大公约数的缩系元素个数的约数。

并且这个足够大的 t 一般要求 a 与 n 重合的那部分素因数被“消除”干净了，即 a^t 这部分素因数的幂次已

经达到或超过了n中的相应幂次。

这个证明是显然的，分素因数讨论即可。

由于欧拉函数的积性，循环节显然是 $\varphi(n)$ 的约数。因此弱化一下就是这样：

$$a^{t+\varphi(n)} \equiv a^t \pmod n$$

这个更方便理解和使用。

Wilson定理

内容

对于任一素数 $p \geq 1$ 到 $p-1$ 的乘积，模 p 余-1。即：

$$(p-1)! \equiv -1 \pmod p$$

或写为比较常见（方便使用）的形式：

$$(n-2)! \equiv \begin{cases} 1 \pmod n & n \text{ is prime} \\ 0 \pmod n & \text{others} \end{cases}$$

等价条件，显然可以用于判定素数，像费马小定理都还有无数个特例存在。但是由于阶乘太大了，且判断余数没有速算法，导致时间复杂度比正常因数分解还要高，所以没人选择这么做。

既然要研究缩系乘法群，那么缩系所有元素乘积自然很重要。Wilson定理说明它是-1。

证明也特别简单：数论倒数两两配对即可。只有两个无法配对的数，1和-1，因此最终结果是-1。

这个定理常用于解决剩余问题，在算法中基本不会遇到。

推广

模不是素数的时候，缩系中所有元素的乘积如何？

对于奇素数的幂次：

$$\prod_{(a,p)=1} a \equiv (-1) \pmod{p^t}$$

对于2的幂次：4以下仍然是-1，但是8以上全是1。

对于一般的整数 n 情形如何？只要8不整除 n 结论仍然是-1。当8整除 n 的时候，情形就非常复杂了，这需要借助中国剩余定理。

设2在 n 中的幂次为 v 下面的不定方程有整数解 x 和 y

$$\frac{n}{2^v} x - 2^v y = 1$$

那么最终结果为：

$$\prod_{(a,n)=1} a \equiv \begin{cases} \frac{n}{2^v} x + 2^v y = 1 + 2^{v+1} y \pmod n & n \text{ is odd} \\ 0 \pmod n & 8 \mid n \end{cases}$$

中国剩余定理

很简单，不同素因子幂乘起来，对应于缩系乘法群的笛卡尔积。因此缩系乘法群的总体构成一个空间，各个素因子的缩系乘法群互不相干，分别构成相应的维度。

当已知这个数在各个维度的坐标，想求这个数的时候，利用线性代数的知识，先求各个维度上的单位向量，然后向量点乘即可。

单位向量的求法，就是一次不定方程。

缩系乘法群的结构

有个经典事实：群的结构与这个素数是不是2有关，当素数是2的时候群的结构会更加复杂。

模为奇素数幂

构成循环群。生成元叫做原根。

不止这类模有原根，事实上1、2、4、奇素数的幂、2倍奇素数的幂都有，也就是说这些缩系乘法群也是循环群，而其余的模都没有。

模为2的幂

当为1、2、4的时候，仍旧是循环群。

当大于等于8的时候，变为一个循环群（元素数为这个数除以4）与 $\{-1,1\}$ 乘法群的笛卡尔积。

著名的Klein四元群与模8的缩系乘法群同构。

离散对数

写在前面

这是一个天坑。关于离散对数的算法数不胜数，甚至是一个P与NP问题。如果未来的您能找到一个多项式时间求解离散对数问题的算法，那么今天的加密算法将半数失效，您不仅可以凭借这个算法轻松拿到图灵奖和菲尔兹奖，甚至可以改写世界历史。当然，如果您证明了不存在多项式时间的求解离散对数问题算法，相当于找到了P与NP问题的有效反例，照样可以拿到图灵奖和菲尔兹奖，只是无法改写历史的进程了而已。

由于本页面不打算涉及算法，那么这部分的算法计划将于暑假再开一个页面（这是因为烤漆实在没时间）。这里仅谈谈离散对数是怎么来的。

定义

离散对数，就来源于循环群。我们知道，原根是缩系乘法群的生成元，那么每个元素是原根的多少次幂呢？

求解幂次，就是标准的对数运算。

(当c为3, 即模数为8的时候, 定义为 $1+i$ ——当然这实在没什么用, 因为它同构于Klein四元群, 习惯采用别的处理方法)

例如模16, 有4个“生成元”(只能跑遍半个缩系) 3、5、11、13, 可以列表验证换底公式(验算不妨将除法改为计算乘法)仍然成立:

$n \pmod{16}$	1	9	5	13	3	11	7	5
$\pmod{16} \quad \log_3 n$	0	2	$3+2i$	$1+2i$	1	3	$2+2i$	$2i$
$\pmod{16} \quad \log_{11} n$	0	2	$1+2i$	$3+2i$	3	1	$2+2i$	$2i$
$\pmod{16} \quad \log_5 n$	0	2	1	3	$3+2i$	$1+2i$	$2+2i$	$2i$
$\pmod{16} \quad \log_{13} n$	0	2	3	1	$1+2i$	$3+2i$	$2+2i$	$2i$

利用复平面上两个维度同时取模(取模构成矩形)意义下的除法, 换底公式仍旧成立。虽然完备, 只是这么定义没什么实际用途罢了。

计算模2的幂以5为底给定元素的对数

这里给一个算法, 计算模2的t次幂缩系中一个元素a以5为底的对数。要求a必须为 $4k+1$ 形式, 因为5的幂在这个缩系乘法群中只能跑遍一半。

(如果a是 $4k+3$ 形式, 就计算 $-a$)

首先, 类似于快速幂的思想, 先计算 $5, 5^2, 5^4, \dots$ 在模2的t次幂意义下的值, 总共有 $t-3$ 个, 因为 $5^{2^{t-2}} \equiv 1 \pmod{2^t}$

有规律: 5 是 $4k+1$ 形式, 5^2 是 $8k+1$ 形式, 5^4 是 $16k+1$ 形式.....。

在模2的幂缩系中 $4k+1$ 形式的数以5为底的对数是奇数 $8k+1$ 形式的数以5为底的对数恰好被2整除(不被4整除) $16k+1$ 形式以5为底的对数恰好被4整除。

因此, 对数计算算法设计非常简单:

第一步: 计算 $a-1$ 在二进制下末尾含多少个0, 假设含 h 个。由于a是 $4k+1$ 形式 h 至少为2。这意味着a以5为底的对数恰好被 2^{h-2} 整除。

第二步: 因为取模下除法(数论倒数)不好计算, 因此改算取模乘法。用a乘上 $5^{2^{h-2}}$ 得到b那么 $b-1$ 当中2的幂次一定比 $a-1$ 要高。

第三步: 如果新的b为1, 则跳出循环, 否则用b代替a回到第一步重新执行。

循环中记录下每一个 $h-2$ 这些 $h-2$ 是单调递增的。

循环结束后, 我们得到一个二进制数: 在每个得到的 $h-2$ 处为1, 其它处为0。因为我们使用了乘法而不是除法, 最后用 2^{t-2} 减去得到的二进制数, 就得到了所求的对数。

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: <https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:namespace:%E5%88%9D%E7%AD%89%E6%95%B0%E8%BA%E4%B8%B9%E5%A4%A7%E5%AE%9A%E7%90%B6%E5%92%B8%E7%B8%A9%E7%B3%BB%E4%B9%98%E6%B3%95%E7%BE%A4&rev=1591342007>

Last update: 2020/06/05 15:26