

素数幂次与p进数问题

记号约定

在本文中，采用习惯记号，素数 p 在 n 中的幂次记为：

$$v_p(n)$$

代表 p 的这个幂次恰好整除 n ，而比这个值更高的幂次无法整除 n 。由于“恰好整除”记号（双竖线）容易和C语言“或”运算混淆，故不采用恰好整除记号。

另外一个记号也在后文出现， p 进制下 n 的各位数字和：

$$S_p(n)$$

这个记号不一定要 p 是素数，只是后文的 p 均为素数。

阅读本文时，希望能提前大致了解模 p 的缩系乘法群的相关知识。

p进数

写在前面

因为 p 进数的主体部分是数论一个艰深的分支，这里只阐述 p 进数的初始观点，不做深入研究，仅为了方便理解后文的内容。

p 进数的基础，是 p 进制整数。所有 p 进制下正整数，改写为 p 进数时形式不变。这部分很简单，默认所有人都已经会了。

取模的观点

例如二进制数110110和101110。

如果我们模8的话，相当于取这两个数的最后三位，结果都为110。这也就是说，模8意义下无法区分两个数。

而模16的时候，相当于取这两个数的最后四位，分别为0110和1110。这个时候才能区分开两个数。

因此有这种感觉，模4就能区分的两个数，比模16才能区分的两个数，在二进数意义下距离更远。

p 进数重新定义了两个数的距离。如果模 p 的 a 次幂才能够区分两个数，那么这两个数的距离就是1除以 p 的 a 次幂，即写成 p 进数表示后从右往左第几位才开始不同。因此 p 进数表示的数无法像普通实数那样排在一条直线上，即没有通常意义的序关系。

因此，模数为素数幂次时，高幂次模数是对低幂次模数更加细化的划分。这样就理解了 p 进数中更靠左的位数更小，与一般的整数理解恰好相反。

也可以这样理解：多项式除法的时候，既可以使得余式的次数越除越高，也可以使得余式的次数越除越低。这依赖于除法的意义不同，导致除法的方向相反。

由于越靠左的位数表示距离越近，权重越小，因此p进制下的数左边可能会无限长，但右边一定有限长，即小数点后的位数有限。

p进制下的有理数

第一种解释，就是上述整数的除法。只是这次要求从右往左除。

类比一般的小数，这种解释也表明p进制下的有理数，左边要么有限，要么无限循环。

因为有取模的意义，采用p进制的时候，有理数本身和它的表示一定是一一对应的，而不是像普通的p进制存在一二对应。

例如7进制下，0.666666.....和1表示同一个数，即每个有限小数都是一二对应，存在两种表示方法，而7进制赋值下.....666666.0和1表示不同的数，至少在有理数范畴，实际的数与它的表示永远是一一对应的。

上面例子的.....666666.0事实上表示-1p进制里没有负号。

第二种解释，采用数论倒数。

例如模13意义下，无法区分1/3和9，那么有理数1/3在13进制里最后一位就是9，并且是小数点前一位。如果想知道有理数1/3在13进制赋值里倒数第二位是多少，就需要求解模169的时候3的倒数是多少，以此类推。

这两种解释完全等价。

采用p进制的时候，小数点后不为0的情况（合法p进制赋值数必然有限长），就是有理数分母中存在p的情况。例如0.1表示1/pp左边无限循环的情况，就是分母中存在非p素因子的情况，也有可能代表负数。

采用p进制方法记录的数，加减乘除的规则都与普通的整数完全相同，只是除法方向相反而已。

因此可以找一个计数方法巧妙记录有理数，例如将1/3就记录成.....10101010101等等，在数论方面的性质仍旧保留了。

p进制下的无理数

规定，每一个右端有限，左边无限不循环的合法p进制定义一个p进制意义下的无理数。那么全体合法p进制对应全体p进制意义下的实数。

数列的收敛：当且仅当随着p进制数列不断延伸pp进制从右往左各位逐一确定（不再改变），则p进制数列收敛于一个p进制。

数列发散于无穷：当且仅当随着p进制数列不断延伸pp进制的右边位数不断增长，可以增长到任意位数，则p进制数列发散于无穷。

其余情况为一般的发散（不收敛）。

定义了收敛和发散，自然所有可以收敛到的值，对应全体p进制意义下的实数。也就是说，可以用无穷的p进制有理数数列逼近任一个p进制实数。

p进制意义下的实数与正常的实数不同，往往不存在一一对应关系。一些正常的实数可能不对应于任何p进制意义下的实数，一些p进制意义下的实数也可能不表示任何正常实数。当然，也不保证两者一多对应的存在，因此事实上p进制创造了新的微积分体系。

例如熟悉的指对数计算，采用 e^x 和 $\ln x$ 引入，计算时采用展开成无穷级数的方法。那么很多正常的实数就没办法表示，因为两个无穷级数在p进赋值里存在收敛域（收敛域不难计算），一个普通的x在正常实数范畴可以存在指对数，到p进数里就可能落在收敛域之外。又有可能在正常实数里负数无法计算对数（暂不考虑复数意义下），但是将负数表示为p进数之后可能落在收敛域里，就存在对数。

后文将探讨收敛域问题。

升幂定理

总述

又叫LTE（Lifting The Exponent）现在统称升幂定理。本来是用于解决p进赋值问题的工具，但是由于它超好用而基础，已经深入到高中竞赛之中。

由于缩系乘法群的结构不同，根据素数为奇素数或2，分为LTEP定理和LTE2定理两部分。

LTEP

这是p为奇素数的情况。

使用本定理的前提：

第一条 p 整除 $a-b$ 即模 p 意义下无法区分 a 和 b

第二条 p 既不整除 a 也不整除 b 即 a 与 b 在 p 进制下末位不为0。

那么 a^n 和 b^n 距离有多近呢？即究竟要取多大的模才能区分 a^n 和 b^n 呢？下面这个公式恒成立：

$$v_p(a^n - b^n) = v_p(a - b) + v_p(n)$$

即贡献分为两部分：原本的距离部分和指数的部分。

LTE2

这是p为2的情况。

使用本定理的前提：

条件一 a 和 b 都是奇数。

条件二 n 为偶数，或者4整除 $a-b$

由于群的结构不同，最终的结论也略有不同。事实上二进赋值与其他的p进赋值的性质不太一样。

$$v_2(a^n - b^n) = v_2(a - b) + v_2(a + b) + v_2(n) - 1$$

贡献实际上也分为两部分：原本的距离部分和指数的部分，只是原本距离部分变复杂了。

条件二比较严格。当4不整除 $a-b$ 且 n 为奇数的时候，定理就不适用了，即此时无法借助本定理计

算 $v_2(a^n-b^n)$

素数在阶乘中的幂次

一般在解析数论研究中偏爱这个式子，最早是由Gauss研究的：

$$v_p(n!) = \sum_{k=0}^{\infty} \left\lfloor \frac{n}{p^k} \right\rfloor$$

这里推荐使用更加流行而简单的公式，替代上面这个繁杂的式子。它用到了文章开头的p进制下各位数字和：

$$v_p(n!) = \frac{n - S_p(n)}{p-1}$$

与等比数列求和公式很相似。由于涉及各位数字和，利用数学归纳法可以轻松证明。

特别地，阶乘中2的幂次是：

$$v_2(n!) = n - S_2(n)$$

p进数的指对数

定义

利用展开式定义：

$$e^x = 1 + x + \frac{1}{2}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \dots$$

$$\ln x = (x-1) - \frac{1}{2}(x-1)^2 + \frac{1}{3}(x-1)^3 - \frac{1}{4}(x-1)^4 + \dots$$

根据上文收敛和发散的定义，利用上面的升幂定理和素数在阶乘中的幂次，可以证得下面的结论。

p为奇素数

此时的p进数中，仅当被p整除的数可以计算指数，被p除余1的数可以计算对数，即两种运算的收敛域。

2进数

在2进数中，仅当被4整除的数可以计算指数，被4除余1的数可以计算对数。这比奇素数时要严格。

其实所有奇数均在对数收敛域内。但是全体被4除余3的数算出来的对数不满足后文的双射条件，算出的对数值再代入指数无法得到原数，而是它的相反数——被4除余1的数。这是由于如果代入计算 $\ln(-1)$ 会算出它等于0。

我们可以计算2进数下的 e^4 和 $\ln 5$ 进而直观了解指对数：任意指数都是 e^4 的指数，利用换底公式可以转换成以5为底的对数，从而与前一篇缩系乘法群中以5为底的对数对比。

e^4 的例子参见：

[OEIS A320815 Digits of the 2-adic integer \$\exp\(4\)\$.](#)

[OEIS A320814 Approximation of the 2-adic integer \$\exp\(4\)\$ up to \$2^n\$.](#)

$\ln 5$ 的例子参见：

[OEIS A152228 2-adic expansion of \$\log\(5\)\$.](#)

[OEIS A321690 Approximations up to \$2^n\$ for the 2-adic integer \$\log\(5\)\$.](#)

同样还能找到更多有趣的数列，可以在OEIS上搜索“2-adic integer”例如 $\ln -3$ 等等
[A321694](#)[A321691](#)

注意，无论奇素数或2的情形 e 和 $\ln 2$ 的相应p进制表示均无法计算，即落在收敛域外，找不到对应的表示。

定义的根据与意义

上面的定义依赖展开式，即只与多项式本身有关，不依赖于x的取值。因此，只用多项式理论即可证得，这种定义满足性质：

互反性：用一个嵌套另一个，会得到1。即两种运算互逆。

乘加转换：对于任意的a和b满足：

$$e^{a+b} = e^a e^b$$

$$\ln ab = \ln a + \ln b$$

因此指对数定义是完备的。借助公式：

$$a^b = e^{b \ln a}$$

可以定义任意二元幂运算，只要相应变量均落入指对数收敛域即可。同时借助换底公式可以计算任意二元对数，也要求落入收敛域。

这里的指对数与之前群论的指对数有何异同？之前群论中的指对数，要求幂次（指数）是普通的整数，只有底数和真数（幂）是取模意义下的。

这里p进制理论中的指对数，底数、指数（幂次）和幂（真数）均在p进制中。它建立了一个双射：模p或4）余1的乘法，与模p或4）余0的加法之间的双射。

素数在组合数中的幂次

组合数对一个数取模的结果，往往构成分形结构，例如著名的谢尔宾斯基三角形就可以通过组合数模2得到。

$$v_p(C_m^n) = \frac{S_p(n) + S_p(m-n) - S_p(m)}{p-1}$$

如果仔细分析p是否整除组合数其实和上下标在p进制下减法是否需要借位有关。这就有了下面的定理。

p在组合数 C_m^n 中的幂次，恰好是p进制下m减掉n需要借位的次数。

特别地，组合数中2的幂次是：

$$v_2(C_m^n) = S_2(n) + S_2(m-n) - S_2(m)$$

Lucas定理

定理

结合上面“素数在组合数中的幂次”一同分析。上面的部分用于计算当组合数被p整除时，一共能被多少个p整除（仅判断模p的幂是否为0）；而这里则研究当组合数不被p整除时，模p余多少。

对于组合数，有同余式：

$$C_{m_1p+m_2}^{n_1p+n_2} \equiv C_{m_1}^{n_1} C_{m_2}^{n_2} \pmod p$$

至于到算法层面，还有与中国剩余定理结合的扩展卢卡斯算法exlucas用于解决模p的幂的余数问题。由于本文注重数学部分，这里不再讲解。

它的证明也不难。就以下两步，然后用到二项展开。

$$(x+1)^{n_1p+n_2} = (x+1)^{n_1p} (x+1)^{n_2}$$

$$(x+1)^p \equiv x^p + 1 \pmod p$$

二项式展开的部分和

给定 $n \leq m \leq x$ 计算多项式取模意义下的值：

$$\sum_{k=0}^m C_n^k x^k$$

记带余除法：

$$n = n_1p + n_2$$

$$m = m_1p + m_2$$

将幂和组合数同时用Lucas定理拆开，有：

$$\sum_{k=0}^m C_n^k x^k \equiv \sum_{t=0}^{m_1-1} C_{n_1}^{t_1} x^{t_1} \sum_{r=0}^{p-1} C_{n_2}^{r_2} x^{r_2} + C_{n_1}^{m_1} x^{m_1} \sum_{r=0}^{m_2} C_{n_2}^{r_2} x^{r_2} = ((1+x)^{n_2})^{m_1} C_{n_1}^{m_1} x^{m_1} \sum_{r=0}^{m_2} C_{n_2}^{r_2} x^{r_2} \pmod p$$

可以对新的 n_1 继续迭代，最终只需预处理若干个 $C_{n_2}^{r_2} x^{r_2}$ 即可。

p进制中的平方元

首先要在p进制中引入类似于二次剩余的概念。根据p进数的定义，即取p的幂次模后能区分的程度作为p进制右侧的数位，模p的幂中的平方元放在p进制中就变为了p进制中相应的平方元。

首先来对比一下整数、有理数和实数中的平方元：

整数的平方元是离散的，分别为0，1，4，9，.....。

有理数的平方元，就是整数平方元之比。

实数中的平方元是全体非负数，并且全体负数都是非平方元。

注意，在这里就出现了负数和非负数的区分。这一点其实很重要。

首先p进数的书写方法与p进制数相同，但是右边有限，左边无穷，这点已经提到了。

于是，每一个p进制都可以唯一表示为 $p^n \cdot u$ 的形式。其中u小数点后没有内容，并且小数点前一位非0。

（相当于将p的因子全部提出来）

于是p进制为平方元的充要条件为n是偶数，并且u小数点前一位是模p的二次剩余。

这体现了p进制中的平方元具有某种聚集的特征，除了含p偶数条件以外，只与一位有效数字有关，与再左边全没关系。

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

Permanent link:
<https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:namespace:%E7%B4%A0%E6%95%B0%E5%B9%82%E6%AC%A1%E4%B8%8E%E8%BF%9B%E6%95%B0%E9%97%AE%E9%A2%98>

Last update: 2020/08/01 23:12