

具体数论与代数

数论与代数的计算性导论

作者：王立斌

组织：华南师范大学

时间：3月10, 2020年

版本：0.10

自定义：信息



The purpose of computing is insight, not numbers. — Richard Hamming

前言

本书面向计算机专业的本科低年级学生，提供一本简单浅显的数论、代数的计算性导论。本书内容的裁剪与组合，陈述的启发性、清晰性与简洁性，以及所倡导的数学教学模式是本书的特色，希望这些特色能成为大家认可的优点。而缺点也非常明显，就是从知识的广度与深度上看都非常欠缺。所以，本书定位为计算机专业数学入门之入门，成为读者攻读其他经典著作的垫脚石。

本书书名的“具体”二字是为了向著名的《具体数学》致敬，同时也向数学的“抽象性”发起挑战，希望本书不要成为另一本抽象代数，而是具体代数。本书的英文名拟定为：A Concrete Introduction to Number Theory and Algebra，因其大写首写字母而简称为CINTA。希望大家可以享受阅读CINTA的乐趣。

王立斌

2020年8月10日

目录

1	整数与二进制	1
1.1	二进制	1
1.2	加法与乘法	3
1.3	除法算法	5
	第一章习题	7
2	欧几里德算法及相关定理	8
2.1	欧几里德算法	8
2.2	扩展欧几里德算法	11
	第二章习题	15
3	同余与模运算	16
3.1	同余与模算术运算	16
3.2	二进制补码	20
3.3	模指数运算	22
	第三章习题	24
4	费尔马小定理和欧拉定理	25
4.1	费尔马小定理	25
4.2	欧拉定理	28
4.3	欧拉 Φ 函数	30
	第四章习题	32
5	素数与素性测试	33
5.1	素数	33
5.2	素性测试	35
	第五章习题	40
6	群与子群	41
6.1	群	41
6.2	群的基本性质	42
6.3	子群	45
	第六章习题	45

7 循环群	47
7.1 循环群	47
7.2 原根	49
7.3 循环群的相关属性	50
7.4 寻找生成元	51
第七章习题	52
8 陪集与拉格朗日定理	53
8.1 陪集	53
8.2 拉格朗日定理	55
第八章习题	56
9 同构, 同态与商群	57
9.1 同构	57
9.2 同态	59
9.3 商群	61
9.4 同构定理	63
Chapter Note	66
第九章习题	66
10 中国剩余定理	67
10.1 数论视角下的中国剩余定理	67
10.2 代数视角下的中国剩余定理	70
Chapter Note	71
第十章习题	71
11 二次剩余	73
11.1 二次剩余与二次非剩余	73
11.2 勒让德符号与欧拉准则	76
11.3 计算平方根	79
Chapter Note	80
第十一章习题	80
12 环与域	82
12.1 环	82
12.2 域	89
第十二章习题	93
13 多项式与有限域	94
13.1 多项式	94

13.2 有限域	100
第十三章习题	102
14 椭圆曲线	103
14.1 初识椭圆曲线	103
14.2 椭圆曲线群的加法	105
14.3 椭圆曲线的点乘算法与离散对数问题	109
Chapter Note	110
第十四章习题	110
15 大整数分解	111
15.1 引言	111
15.2 Pollard 的 $p-1$ 法	112
15.3 Pollard 的 Rho 算法	114
第十五章习题	115
16 离散对数	117
16.1 导引	117
16.2 Pohlig-Hellman 算法	117
16.3 大步小步算法	120
16.4 Rho 算法	121
第十六章习题	123
A 快速乘法算法	124
B 斐波那契数列	126
C 生日悖论	128

第一章 整数与二进制

内容提要

□ 二进制的性质

□ 整数乘法

□ 整数加法

□ 整数除法

本书的数学历程开始于古老的且读者应该熟知的数学系统：整数，记为 \mathbb{Z} ，它包括了 $0, \pm 1, \pm 2, \pm 3, \dots$ 。而包含所有正整数的集合称为自然数，记为 \mathbb{N} 。当然，以下数学系统虽然不是本书重点，作为常识还是需要以下记号：

- \mathbb{Q} ：有理数
- \mathbb{R} ：实数
- \mathbb{C} ：复数

1.1 二进制

从计算机科学的角度去理解数学首先要认识二进制数，因为它是计算机系统中使用的数值表达系统。所谓二进制就是用 0 和 1 这两种数位（又称为比特）去表达所有数值的系统。本书并不会系统地去讲解二进制的知识点，仅通过实例展示若干关于二进制的整数运算规则，这些都是计算机专业学生（简记为 CSers）应该熟知但又往往被忽略的规则。为方便阅读，我们只在避免引起歧义时才在二进制数值前加上 0b 标识。

让我们先考虑所有的正整数。因为二进制中只有 0 和 1 两种数位，所以用 0 和 1 表达的自然数只能是：

$0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010 \dots$

显然，这些数字分别对应的十进制数就是：

$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \dots$

从以上数列可以归纳出：

性质 (二进制性质一)

偶数二进制最末尾的比特是 0；奇数二进制最末尾的比特是 1。

相信大多数 CSers 立即可以认可这个规律。接着，关注以下十进制数列的二进制表示：

$0, 2, 4, 8, 16, 32, 64, 128, 256 \dots$

发现它们分别是：

$0, 10, 100, 1000, 10000, 100000, 1000000, 10000000 \dots$

于是，可以有以下结论吗？

性质 (二进制性质二) 在一个二进制数末尾增加一个 0 等同于在十进制中对这个数乘 2。反过来说, 对一个十进制数进行乘 2 操作等同于对其二进制表达左移一个比特。

在稍微犹豫片刻之后, 我们也会认可它的。



思考 请问, 你认为对一个十进制数进行除 2 等于对其二进制表达右移一个比特吗?

考虑任意自然数 n , 所谓 2 的 n 次方 (2^n) 只是不断对 1 乘 n 次 2, 所以有以下规律。

性质 (二进制性质三)

给定任意自然数 n , 十进制数 2^n 的二进制数表达就是在 1 后加 n 个 0: $0b1\underbrace{000\cdots0}_n$ 。

如果性质三是对的, 那么我们还会有以下规律:

性质 (二进制性质四)

给定任意自然数 n , 十进制数 $2^n - 1$ 的二进制数表达就是 n 个 1: $0b\underbrace{111\cdots1}_n$ 。

例如, $2^7 - 1 = 127$, 它的二进制表达就是 $0b\underbrace{1111111}_7$ 。

例 1.1 请计算以下等式:

$$1 + 2 + 2^2 + 2^3 + \cdots + 2^{10}$$

考虑到也许我们并不具备高斯的头脑, 而且我们可能是熟悉二进制的 CSers, 所以我们用二进制“硬”算。显然, 我们有:

$$\begin{aligned} 1 &= 0b1 \\ 2 &= 0b10 \\ 2^2 &= 0b100 \\ \cdots &= \cdots \\ 2^9 &= 0b1000000000 \\ 2^{10} &= 0b10000000000 \end{aligned}$$

所以, 把上面的式子左右分别累加起来, 太幸运了, 没有任何进位, 于是我们得到:

$$\sum_{i=0}^{10} 2^i = 0b\underbrace{1111111111}_{11}$$

又因为 $0b\underbrace{1111111111}_{11}$ 等于十进制的 $2^{11} - 1$, 所以答案是 2047。



思考 请将以上计算过程的结论归纳成一个定理, 并证明。你可以使用任何的证明方法。请问, 以上“硬”算的方法能否推广为一种证明方法?

例子 1.1 有意思之处不在于其“炫技式”的蛮干, 而是在于提醒我们, 如果要将任意一个整数 b 表达为二进制数, 只需要对某些 2^i ($i \in [0, n-1]$, n 是 b 的比特长度) 进行累加, 合适的项加进来, 不合适的就舍弃。如下式所示, $b_i \in \{0, 1\}$ 可视为取舍的标识符, 也是整数 b 的二进制表达。

$$\begin{aligned}
 b &= b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_12 + b_0 \\
 &= \sum_{i=0}^{n-1} b_i 2^i
 \end{aligned}$$

这就是二进制表达中常用的位置计数法（position notation）。位置计数法不仅仅是一种概念性的符号表达法，而应该把它视为一种常用的有效工具。很快我们会看到这样的实例。

1.2 加法与乘法

毫无疑问，我的读者肯定会手算整数的加法和乘法，但是是否知道计算机如何做加法和乘法，那就不好说了。

先让我们检验以下实现整数加法的算法：

```

1 //输入：两个整数a和b
2 //输出：a与b的和
3 int add(int a, int b) {
4
5     if (b == 0) return a;
6     return add(a ^ b, (b & a) << 1);
7
8 }
```

算法 1.1: 整数加法

这个程序表达的主要思想是，计算机中的加法仅仅使用到比特的逻辑操作：异或操作、与操作及移位操作。



思考 要理解这个加法算法只需要正确地回答出以下三个问题：

1. $a \wedge b$ 得到的是什么？
2. $(b \& a) \ll 1$ 得到的是什么？
3. 该算法为什么会终止？

关于加法我们就讨论到这里，如果读者对以上问题还是没有答案，建议找相关入门教材自行阅读。接下来我们考察整数乘法的计算机实现。请阅读以下最朴素的（如果不能称为愚蠢的话）整数乘法算法。

```

1 # Input: two integers a and b, where a >= b.
2 # Output: the product of a and b.
3 def naive_multiply(a, b):
4     if b == 0:
```



```

5     return 0
6     return a + naive_multiply(a, b - 1)

```

算法 1.2: 朴素的整数乘法

一般而言, 当我们思考一个新的算法, 会重点考虑以下三个关键要素:

- **正确性.** 该算法是正确地达成既定目标?
- **效率.** 算法执行多少时间 (步骤) 才会终止?
- **优化.** 我们能否给出效率更佳的算法?

假定 a 和 b 都是 n 比特的整数, 该算法显然正确。因为:

$$a \cdot b = \underbrace{a + a + a + \cdots a}_b$$

该算法使用了 b 次加法, 每次加法使用 $O(n)$ 次比特操作, 因此粗略地说, 该算法的总体开销是 $O(bn)$ 次比特操作。如何提高效率? 考察下面递归式, 改进后的算法我们直接给出递归版本。

$$a \cdot b = \begin{cases} 2(a \cdot \lfloor b/2 \rfloor) & \text{如果 } b \text{ 是偶数;} \\ a + 2(a \cdot \lfloor b/2 \rfloor) & \text{如果 } b \text{ 是奇数.} \end{cases} \quad (1.1)$$

```

1 # Input: two integers a and b.
2 # Output: the product of a and b.
3 def multiply(a, b):
4     if b == 0:
5         return 0;
6     if is_even(b): # the last bit of b is 0;
7         return 2*multiply(a, b/2);
8     else:          # b is odd
9         return 2*multiply(a, b/2) + a;

```

简单乘法 (递归版)

正确性显然成立。为什么这个算法提高了效率? 分析递归调用的次数可知。每次递归调用 b 值减半, 即 b 除 2, 或者说 b 被右移了一个比特, 所以该算法在执行最多 n 次递归调用之后必然终止。算法的总体比特运算的开销是 $O(n^2)$ 。

该算法的直观思想还可以用下面的方式解释。对任意两个 n 比特的整数 a 和 b , 用位置记数法把 b 表示为二进制数, 可得以下等式:

$$\begin{aligned} a \cdot b &= a \cdot (b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_12 + b_0) \\ &= b_{n-1}(a \cdot 2^{n-1}) + b_{n-2}(a \cdot 2^{n-2}) + \cdots + b_1(a \cdot 2) + b_0a \end{aligned}$$

观察可知:



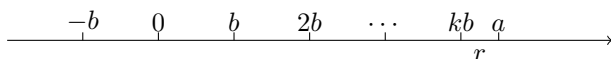


图 1.1: 除法的几何解释.

1. 上式的每一项都包含有 $a \cdot 2^i$, 也就是说我们必须重复地对 a 乘 2。
2. 每一项中的 b_i 比特可视为一个标识符, 当 $b_i = 1$ 时, 我们就把 $a \cdot 2^i$ 加到最终结果当中。

例 1.2 计算 3×10 。重复对 3 乘 2^i , $i \in [0, 3]$ (因为 10 是一个 4 比特数), 得到: 3, 6, 12 和 24。因为 10 的二进制表达是 1010, 所以要累加的项是 6 和 24, 因此最终结果是 30。

也许你不能立即看出所给出递归算法与这种思想的关系, 读者的任务就是要理解这种等价性。并且, 要求读者使用以上思想实现一种迭代式的简单乘法, 这是课后习题。

最后, 我们还必须问, 能否有更高效率的乘法算法? 答案是肯定的, 只是这并非本章的任务, 有兴趣的读者可以参考附录的第 A 节。

1.3 除法算法

乘法的直接对应操作就是除法。整数 a 除以非零整数 b 产生商值 q 和余数 r , 除法可形式化表达为以下定理。

定理 1.1. 除法算法

对任意给定的整数 a 和 b , 其中 $b > 0$, 存在唯一的整数对 q (商) 和 r (余数) 使得,

$$a = qb + r$$

且 $0 \leq r < b$ 。



图 1.1 展示了除法算法背后的几何直观。数轴上一系列的等距离点代表所有 b 的倍数, 而表达 a 值的点肯定落在某两个等距点的中间, 比如在 qb 和 $(q+1)b$ 之间。这就表示 $a - bq = r$, 其中 r 表示一段比 b 要小的长度。因此, 正如定理所要求, $0 \leq r < b$ 。

除法算法是数论的重要基础, 在以后的章节中该算法还会再次出现, 值得关注。而图 1.1 的直观也非常具有启发性。接下来我们要证明定理 1.1, 该证明使用了以下公理。

公理 1.1. 良序原则

自然数的非空子集必然存在一个最小元素。



证明 证明包括两部分: 存在性和唯一性。

- 存在性.
构造集合

$$S = \{a - bk : k \in \mathbb{Z} \text{ 且 } a - bk \geq 0\}.$$

显然, 集合 S 非空。由良序原则, 存在一个最小元 $r \in S$, 且 $r = a - qb$ 。因此, $a = qb + r, r \geq 0$ 。证明 $r < b$ 则留作课后习题。



- 唯一性证明留作课后习题。

以下是一些常用术语。设 a 和 b 是整数。如果存在整数 q 使得 $a = qb$, 则称 a 可被 b 整除, 又记为 $b \mid a$ 。如果整数 d 满足 $d \mid a$ 且 $d \mid b$, 则称 d 为 a 和 b 的公因子。如果整数 d 是 a 和 b 的公因子, 且对 a 和 b 的其他公因子 d' 都有 $d' \mid d$, 则称 d 是 a 和 b 的最大公因子 (greatest common divisor), 记为 $d = \gcd(a, b)$ 。如果 $\gcd(a, b) = 1$, 则称 a 和 b 互素。关于整除, 有些简单的结论, 证明留给读者作为练习。

命题 1.1. 整除性

设 $a, b, c \in \mathbb{Z}$, 如果 $a \mid b$, $b \mid c$, 则 $a \mid c$ 。如果 $c \mid a$, $c \mid b$, 则对任意 $m, n \in \mathbb{Z}$, 有 $c \mid (ma + nb)$ 。



讲到互素就必须提一下素数与合数这两个概念。关于素数, 我们将在第5章进一步进行讨论。

定义 1.1. 素数与合数

若 $p \in \mathbb{Z}$ 且 $p > 1$ 。如果整除 p 的正整数只有 1 和 p 自己本身, 则称 p 为素数。如果整数 $n > 1$ 不是素数, 则称之为合数。



为了与乘法比较, 下面展示两种除法算法, 读者可自行分析其正确性与效率, 这作为课后练习。

```
1 def Navie_Divide(a, b):
2     q, r = 0, 0
3     while(a >= b):
4         a, q = a - b, q + 1
5     r = a
6     return q, r
```

算法 1.3: 朴素除法

```
1 def Divide(a, b):
2     if(a == 0):
3         return 0, 0
4     q, r = Divide(a / 2, b)
5     q, r = 2 * q, 2 * r
6     if(a & 1): #if a is an odd number
7         r = r + 1
8     if(r >= b):
9         r, q = r - b, q + 1
10    return q, r
```

算法 1.4: 简单除法



从以上算法实现中我们看到：乘法的实现依靠加法与比特位移操作，而除法的实现依靠减法与比特位移操作。需要提醒的是，这其实不仅仅是算法实现上问题，而是关于计算思想的问题，或者说认识乘法、除法的本质有助于我们认识其他的算法问题。很快我们就会体会到这一点。

第一章习题

1. 用 C 语言编程实现判断输入为偶数的函数，即如果输入为偶数，返回 $True$ ，否则返回 $False$ 。
2. 给定一个整数 v ，如何判断 v 是否 2 的某次方？比如， $v = 4 = 2^2$ ，返回 $True$ ； $v = 9 = 2^3 + 1$ 并非 2 的次方，返回 $False$ 。请写一个 C 语言的函数来实现以上功能。
3. 用 C 语言编程实现一种迭代版本的简单乘法。
4. 证明命题 1.1。
5. 完成定理 1.1 的证明（除法算法）。
6. 检验朴素除法与简单除法的正确性，并分别给出它们的算法效率，假定输入为 n 比特整数。

第二章 欧几里德算法及相关定理

内容提要

□ gcd

□ binary gcd

□ egcd

2.1 欧几里德算法

除法算法是一个重要的开始，以除法算法为基础，定义整除性、公因子、最大公因子等概念。本章讨论与最大公因子相关的算法与理论。欧几里德算法 (*Euclidean algorithm*) 计算两个整数的最大公因子，也简记为 gcd 算法。这是一种古老的、著名的，且目前依然应用广泛的算法，其历史可以追溯到古希腊数学家欧几里德 (Euclid)。

定理 2.1. 欧几里德算法

给定两个整数 a 和 b ，设 $a \geq b$ ，则 a 和 b 的最大公因子等于 b 和 $a \bmod b$ 的最大公因子。即

$$gcd(a, b) = gcd(b, a \bmod b)$$

其中， $a \bmod b$ 表示用 a 除以 b 所得到的余数 r 。

a 除以 b 所得到的余数 r ，表示存在整数 k 使得 $a = kb + r$ ，图2.1展示了这种取余操作。

例 2.1 给定 $a = 12345$ 和 $b = 678$ ，求它们的最大公因子。

$$\begin{aligned} gcd(12345, 678) &= gcd(678, 141) = gcd(141, 114) \\ &= gcd(114, 27) = gcd(27, 6) = gcd(6, 3) = gcd(3, 0) \end{aligned}$$

因此， $gcd(12345, 678) = 3$ 。

例 2.2 使用 SageMath 命令gcd我们可以很容易计算两个整数的最大公因子。

```
1 sage: gcd(12345, 678)
2 3
3 sage: gcd(12^20, 18^20)
4 3656158440062976
```

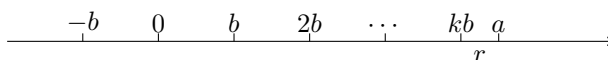


图 2.1: $a \bmod b$ 的几何解释.

例 2.3 证明, 对任意两个正整数 a 和 b , 有 $\gcd(2^a - 1, 2^b - 1) = 2^{\gcd(a,b)} - 1$ 。

首先, 我们证明,

$$(2^a - 1) \bmod (2^b - 1) = (2^r - 1),$$

其中 $r = (a \bmod b)$ 。如果考虑任意 $i \in \mathbb{N}$, $2^i - 1$ 的二进制表达, 不难看出:

$$2^a - 1 = 0b \underbrace{11 \dots 11}_{b} \underbrace{11 \dots 11}_{b} \dots \underbrace{11 \dots 11}_{b} \underbrace{11 \dots 11}_{r}$$

因此, 我们有 $\gcd(2^a - 1, 2^b - 1) = 2^{\gcd(a,b)} - 1$, 因为

$\gcd(2^a - 1, 2^b - 1) = \gcd(2^b - 1, 2^{r_1} - 1)$, 其中 $r_1 = a \bmod b$;

$\gcd(2^b - 1, 2^{r_1} - 1) = \gcd(2^{r_1} - 1, 2^{r_2} - 1)$, 其中 $r_2 = b \bmod r_1$;

...

$\gcd(2^{r_{k-1}} - 1, 2^{r_k} - 1) = 2^{r_k} - 1$, 其中 $r_k | r_{k-1}$, 或者, 等价地 $r_k = \gcd(a, b)$ 。



思考 很显然, 我是特意选择了这个例子。建议读者给出另一种数学家的证明。

以下直接给出一种计算 $\gcd(a, b)$ 的递归程序, 建议读者自己完成迭代版本的程序。

```

1 # Input: positive integers a and b, with a > b .
2 # Output: the greatest common divisor of a and b.
3 def gcd(a, b):
4     if(b == 0):
5         return a
6     else:
7         return gcd(b, a % b)

```

算法 2.1: 欧几里德算法

正确性. 要证明算法的正确性, 只需要验证以下计算成立即可:

$$\gcd(a, b) = \gcd(a - b, b)$$

然后, 利用这条规则, 我们能不断地从 a 中减去 b 而最终得到余数。请回忆算法 1.3 (朴素除法算法) 的过程, 不难导出以下计算过程:

$$\gcd(a, b) = \gcd(a - b, b) = \gcd((a - b) - b, b) = \dots = \gcd(a \bmod b, b)$$

要证明 $\gcd(a, b) = \gcd(a - b, b)$, 必须证明两个方向。首先, 任何整除 a 和 b 的整数必然整除 $a - b$, 因此 $\gcd(a, b) \leq \gcd(a - b, b)$ 。也就是说任意 a 与 b 的公因子必然小于等于 $a - b$ 和 b 的最大公因子, 而 $\gcd(a, b)$ 只是 a 与 b 的一个公因子而已。反方向类似, 容易证明任意整除 $a - b$ 和 b 的整数必然整除 a 和 b , 因此, $\gcd(a - b, b) \leq \gcd(a, b)$ 。这种证明方法频繁用于证明两个集合的相等, 本质上以上证明也是证明两个集合的相等, 值得初学者重视。

效率. 要理解 \gcd 算法的高效率, 我们必须意识到 $a \bmod b$ 这步操作对输入数据作出了重大的消减。容易验证:

引理 2.1

如果 $a \geq b$, 则 $(a \bmod b) < a/2$ 。



如果该引理成立, 那么算法连续两次迭代后, a 与 b 的值都至少消减了一半, 意味着算法在 $O(\log a)$ 步之后会终止。验证引理的正确性很容易, 因为如果 $b < a/2$, 由除法算法即得。如果 $b > a/2$, 则 $(a \bmod b) = a - b$, 证完。进一步, 容易看出 $O(1 + \log b)$, 或等价的 $O(\log b)$ 是一种更好的上界。

算法优化. 是否存在更高效的算法? 二进制欧几里德算法, 简称为二进制 \gcd 算法 (*binary gcd algorithm*, 简记为 *bgcd* 算法), 也被称为 *Stein* 算法, 它与 \gcd 算法功能相同, 但效率更好。该算法避免使用复杂的计算操作, 比如除法和乘法, 使用减法和更高效的比特移位操作来提高效率。*bgcd* 算法的分析留做课后练习, 以下直接给出代码。

```

1 # Function to implement Stein's Algorithm
2 # Input: positive integers a and b, with a > b .
3 # Output: the greatest common divisor of a and b.
4 def binary_gcd( a, b):
5     # Finding 'k', which is the greatest power of 2
6     # that divides both 'a' and 'b'.
7     k = 0
8     while(((a | b) & 1) == 0) :
9         a = a >> 1
10        b = b >> 1
11        k = k + 1
12    # Dividing 'a' by 2 until 'a' becomes odd
13    while ((a & 1) == 0) :
14        a = a >> 1
15    # From here on, 'a' is always odd.
16    while(b != 0) :
17    # If 'b' is even, remove all factor of 2 in 'b'
18        while ((b & 1) == 0) :
19            b = b >> 1
20    # Now 'a' and 'b' are both odd.
21    #Swap if necessary so a <= b
22    if (a > b) :
23        a, b = b, a # Swap 'a' and 'b'.
24    b = (b - a)      # b = b - a (which is even).
```

```

25 # restore common factors of 2
26 return (a << k)

```

算法 2.2: 二进制欧几里德算法

在本节的最后，补充关于 \gcd 算法的正确性的另一种证明。之前的证明非常简洁直观，而补充证明则更具有通用性。除法算法、欧几里德算法在以后章节（或者未来的学习中）会以另外的形式出现，希望大家能记住以下并不复杂的证明，并体会除法算法在证明中的应用。

证明 [\gcd 算法正确性的另一种证明.] 设 $d = \gcd(a, b)$, $r = (a \bmod b)$ 。根据除法算法，存在唯一的 $q \in \mathbb{Z}$ 使得 $r = a - qb$ 。根据命题 1.1，有 $d \mid r$ ，所以 d 是 b 和 r 的公因子。要证明 d 是 b 和 r 的最大公因子只需要证明对 b 和 r 任意的公因子 c 有 $c \leq d$ 。然而因为 $c \mid b$ 和 $c \mid r$ ，再次根据命题 1.1， $c \mid (qb + r)$ ，即 $c \mid a$ ，所以 c 是 a 和 b 的公因子。因为 d 是 a 和 b 的最大公因子，所以 $c \leq d$ 。

2.2 扩展欧几里德算法

欧几里德算法计算两个非零整数 a 和 b 的最大公因子，而扩展欧几里德算法 (*extended Euclidean algorithm*，简记为 *egcd* 算法) 不但计算出最大公因子，还同时计算出这个公因子如何表达为 a 和 b 的线性组合。*egcd* 算法在模算术运算中起到重要作用，很快我们就会看到。

整数 a 和 b 的公因子一定能表达为 a 和 b 的线性组合吗？直观上， \gcd 算法每一迭代步骤做了一次除法，也即若干次的减法，所得到的余数必为 a 和 b 的线性组合。迭代下去，无非就是得到 a 和 b 的线性组合的线性组合，当然还是 a 和 b 的线性组合。为了严谨性，需要以下定理。

定理 2.2. Bézout 定理

设 a 和 b 为非零整数，存在整数 r 和 s 使得：

$$\gcd(a, b) = ar + bs$$

而且， a 与 b 的最大公因子是惟一的。称 r 和 s 为 *Bézout* 系数。



Bézout 定理的证明可在标准数论教材中找到。以下给出证明思路。

证明 构造集合

$$S = \{am - bn : m, n \in \mathbb{Z} \text{ 且 } am + bn \geq 0\}.$$

显然，集合 S 非空，根据良序原理，取其中最小值 $d = ar + bs$ 。然后证明两个属性： d 是 a 和 b 的公因子；如果存在 a 和 b 的公因子 d' ，则 $d' \mid d$ 。这样就证明了 $d = \gcd(a, b)$ 。请读者完成细节，留作课后练习。

Bézout 定理说明两个非零整数 a 和 b 的公因子一定能表达为 a 和 b 的线性组合，强

调的是存在性，*egcd* 算法则高效地计算出 a 和 b 的最大公因子及其相应的 Bézout 系数。

例 2.4 使用 SageMath 命令 `xgcd` 可以容易计算两个整数的最大公因子及其相应的 Bézout 系数。

```
1 sage: xgcd(12345, 678)
2 (3, 101, -1839)
3 sage: xgcd(12^20, 18^20)
4 (3656158440062976, -1312176233, 394609)
```

在解释 *egcd* 算法的具体计算过程之前，请先理解以下手动计算的例子。

例 2.5 给定 $a = 19$, $b = 13$, 求 r 和 s , 使得 $ar + bs = \gcd(19, 13)$ 。为展示其计算过程，首先我们写出以下两条显然成立的等式：

$$a \cdot 1 + b \cdot 0 = 19 \quad (2.1)$$

$$a \cdot 0 + b \cdot 1 = 13 \quad (2.2)$$

让公式 2.1 减公式 2.2，可得新的等式如下：

$$a \cdot 1 + b \cdot (-1) = 6 \quad (2.3)$$

然后，让公式 2.2 减去公式 2.3 的两倍，得到：

$$a \cdot (-2) + b \cdot 3 = 1 \quad (2.4)$$

最终，得到答案： $r = -2$, $s = 3$ 和 $d = 1$ 。

要理解这个计算过程，只需要问：为什么等式的右边出现了 a 和 b 的最大公因子？让我们先简单归纳以上计算过程：

- 写出两个显然成立的等式；
- 不断用某个等式减去另一个等式的倍数，直到某个等式的右边出现 $\gcd(a, b)$ 。

至少要注意到两个事实。第一，迭代过程中，所有等式的右边一直在执行 *gcd* 算法的计算过程，表面上是做一次减法，本质上需要一次除法。其次，等式的左边根据等式右边的减法而不断在维持等式的平衡。实际上， a 和 b 只是占位符，并不参与计算，也不改变。既然如此，何不省略它们，把等式表达为矩阵的形式？请看使用这种表示法的例子。

例 2.6 给定 $a = 13$, $b = 9$, 计算 r 和 s , 使得 $ar + bs = \gcd(13, 9)$ 。写成以下矩阵：

$$\begin{pmatrix} 1 & 0 & 13 \\ 0 & 1 & 9 \end{pmatrix} \quad (2.5)$$

用第一行减第二行，得到一个新的行。接着，把第一行替换为第二行，而新的一行替换原来的第二行，得到一个新的矩阵：

$$\begin{pmatrix} 0 & 1 & 9 \\ 1 & -1 & 4 \end{pmatrix} \quad (2.6)$$

接着，第一行减去两倍的第二行，同样进行行调整得到：

$$\begin{pmatrix} 1 & -1 & 4 \\ -2 & 3 & 1 \end{pmatrix} \quad (2.7)$$

最终，我们得到答案： $r = -2$ ， $s = 3$ 和 $d = 1$ 。

这个例子让你想起了高斯消元法了吗？很类似，但并不是。高斯消元法的目标是从任意矩阵出发得到一个单位矩阵，而这里却是从一个单位矩阵出发。

以矩阵为表达工具，进一步抽象以上算法过程。对任意的非零整数 a 和 b ，构造矩阵 M ：

$$M = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \end{pmatrix} \quad (2.8)$$

M 满足以下矩阵乘法：

$$M \begin{pmatrix} a \\ b \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (2.9)$$

不断对 M 进行迭代“消元”，即某一行减去另一行的倍数，并进行行交换等操作，最终我们得到另一个矩阵 M'

$$M' = \begin{pmatrix} r & s & d \\ R & S & 0 \end{pmatrix} \quad (2.10)$$

d 是 a 和 b 的最大公因子， r 和 s 是对应的 Bézout 系数，且 M' 同样满足以下乘法：

$$M' \begin{pmatrix} a \\ b \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (2.11)$$

矩阵乘法表明： $ar + bs = d$ 。 $egcd$ 算法明确地描述了这个计算过程，请参看以下算法描述。

```

1 # Input: integers a and b, with a > b .
2 # Output: d=gcd(a,b), and
3 # r and s s.t. d = a*r + b*s
4 def egcd(a, b):
5     r0, r1, s0, s1 = 1, 0, 0, 1
6     while(b):
7         q, a, b = a//b, b, a%b
8         r0, r1 = r1, r0 - q*r1
9         s0, s1 = s1, s0 - q*s1
10    return a, r0, s0

```

算法 2.3: 扩展欧几里德算法

$egcd$ 算法的正确性显然，其效率与 gcd 算法相同。同样也存在 $binary\ egcd$ 算法，其分析留为课后练习，以下直接给出代码。


```

1 # Input: positive integers a and b, with a > b .
2 # Output: the greatest common divisor of a and b and Bezout coefficients
3 .
4 def binary_egcd( a, b):
5     # Finding 'k', which is the greatest power of 2
6     # that divides both 'a' and 'b'.
7     k = 0
8     while(((a | b) & 1) == 0) :
9         a = a >> 1
10        b = b >> 1
11        k = k + 1
12    r0, r1 = a, b
13    s0, s1, t0, t1 = 1, 0, 0, 1
14    while(r1 != 0):
15        #If r0 is even, remove all factor of 2 in r0
16        while ((r0 & 1) == 0) :
17            r0 = r0 >> 1
18            if (((s0 | t0) & 1) == 0) :
19                s0 = s0 >> 1
20                t0 = t0 >> 1
21            else:
22                s0 = (s0 + b) >> 1
23                t0 = (t0 - a) >> 1
24        #If r1 is even, remove all factor of 2 in r1
25        while ((r1 & 1) == 0) :
26            r1 = r1 >> 1
27            if (((s1 | t1) & 1) == 0) :
28                s1 = s1 >> 1
29                t1 = t1 >> 1
30            else:
31                s1 = (s1 + b) >> 1
32                t1 = (t1 - a) >> 1
33        if (r1 < r0) :
34            r0, s0, t0, r1, s1, t1 = r1, s1, t1, r0, s0, t0
35        r1 = r1 - r0
36        s1 = s1 - s0
37        t1 = t1 - t0

```

37

```
return (r0 << k), s0, t0
```

算法 2.4: 二进制扩展欧几里德算法

2020 年 9 月 20 日完成 begcd 的代码。补充进来。

❧ 第二章 习题 ❧

1. 写一个迭代版本的 gcd 算法程序和一个二进制 gcd 算法程序。
2. 写一个递归版本的 $egcd$ 程序和一个迭代版的 $egcd$ 程序。
3. 写一个迭代版本的二进制 $egcd$ 算法程序，并分析其算法的正确性和算法复杂度。
4. 请完成定理 2.2 的证明。
5. 对任意正整数 n ，记 $T_n = 2^{2^n} + 1$ ，这就是所谓的费尔马数 (*Fermat Number*)。请证明，对任意不同的正整数 m 和 n ， T_m 与 T_n 互素。
6. 证明：如果 $gcd(a, b) = d$ ，则 $gcd(a/d, b/d) = 1$ 。
7. 证明：如果 $gcd(a, b) = 1$ ，且 $gcd(a, c) = 1$ ，则 $gcd(a, bc) = 1$ 。
8. 证明：如果 $gcd(a, b) = 1$ ，则 $gcd(a, b^n) = 1$ ， n 是任意正整数。(提示：利用以上习题的结论。)
9. 证明：如果 $gcd(a, b) = 1$ ，则 $gcd(a^n, b^n) = 1$ ， n 是任意正整数。(提示：利用以上习题的结论。)
10. 证明：对任意整数 a ， b 和 d ， $gcd(a^d, b^d) = gcd(a, b)^d$ 。
11. 求满足方程 $4x + 8y + 15z = 1$ 的整数解 x ， y 和 z 。满足什么条件的整数 a ， b 和 c 使得等式 $ax + by + cz = 1$ 有解？请给出自己的解法，并用程序实现。

第三章 同余与模运算

内容提要

□ 同余

□ 模指数运算

□ 二进制补码

3.1 同余与模算术运算

在开始本章学习之前用以下 C 程序作为热身，先阅读程序，尝试给出程序的输出，然后编译运行它们，看看自己的答案是否对。答案与结论并不重要，只是邀请众位与数字玩耍，也许从中可以找到一些规律与乐趣。

```
1 unsigned char a = 255, b = 2;
2 a = a + b;
3 printf("a == %d \n", a);
```

热身练习-1.

```
1 int n = 32; /*try different values here;*/
2 unsigned int val = 1; /*or try a different type here;*/
3 for(int i = 1; i <= n; i++)
4 {
5     val = val * 2;
6     printf("What you get is %u \n.", val);
7 }
```

热身练习-2

通过以上练习，想强调的是，在计算机系统中每一个数值都表达为定长的二进制形式，其中的所有运算都是定长计算。定长的长度可能是 8 比特、16 比特、32 比特或者 64 比特，通常是 8 的倍数，但不大可能是 7 比特或者其他。计算机系统最基础的计算是二进制数值的与、或、非、异或、移位等逻辑操作，其次就是模算术运算，即二进制数值的加、减、乘、除等。在这一章我们将学习一种自然、方便的描述模算术运算的方法：同余 (congruence)。

如果整数 a 、 b 和 m 满足 $m \mid (a - b)$ ，我们称 a 与 b 模 m 同余，记为：

$$a \equiv b \pmod{m} \quad (3.1)$$

称整数 m 为同余式的模数 (*modulus*)。等价地, 等式 3.1 也意味着存在整数 k 使得:

$$b = a + km \quad (3.2)$$

等式 3.2 让我们想起定理 1.1 (除法算法), a 除 m 的余数 (表示为 $(a \bmod m)$) 等于 b 除 m 的余数 (表示为 $(b \bmod m)$), 也就是:

$$(a \bmod m) = (b \bmod m) \quad (3.3)$$

有理由相信, 这两种不同的同余的表示方式具有同等的重要性, 在不同的情形下选择正确的表示法会给我们带来方便。

例 3.1

$$26 \equiv 8 \pmod{9} \quad \text{且} \quad 6 \equiv 55 \pmod{7}$$

因为

$$9 \mid (26 - 8) \quad \text{且} \quad 7 \mid (6 - 55),$$

或者, 等价地:

$$8 = 26 - 2 * 9 \quad \text{且} \quad 55 = 6 + 7 * 7.$$

模加法、模乘法可以最直接的方式定义如下, 并写成以下命题的形式:

命题 3.1

如果 $a_1 \equiv b_1 \pmod{m}$ 而且 $a_2 \equiv b_2 \pmod{m}$, 则

$$a_1 \pm a_2 \equiv b_1 \pm b_2 \pmod{m} \quad \text{且} \quad a_1 a_2 \equiv b_1 b_2 \pmod{m}.$$

证明 容易, 作为课后练习。 □

命题 3.1 是说, 如果对两个整数 a 和 b 进行模 m 加法或者乘法, 我们可以首先算出 $a' = (a \bmod m)$ 和 $b' = (b \bmod m)$, 然后对 a' 和 b' 做模 m 的加法或者乘法, 得到最终答案。通过使用这种方法我们可以赢得效率, 因为我们总是在尽可能小的数值上进行计算。或者, 在某种意义上, 同余的值将被视为同一个值。

例 3.2 因为 $10000 \equiv 1 \pmod{3}$, 且 $998 \equiv 2 \pmod{3}$, 则

$$10000 * 998 \equiv 2 \pmod{3}$$

利用命题 3.1, 我们还可以解带一个未知变量的同余式。

例 3.3 求解以下带变量 x 的同余式:

$$x + 7 \equiv 3 \pmod{11}$$

因为

$$-7 \equiv -7 \pmod{11},$$

将上两式相加, 得到:

$$x \equiv 3 - 7 \equiv -4 \pmod{11}.$$

因此, 有一个好消息:

好消息.

对形如以下的同余式:

$$x + a \equiv b \pmod{m},$$

其解是:

$$x \equiv b - a \pmod{m}.$$

然而坏消息就是, 要解以下带乘法的同余方程

$$ax \equiv b \pmod{m},$$

解必须是:

$$x \equiv \frac{b}{a} \pmod{m}.$$

一个显然的问题就是: $(\frac{b}{a} \pmod{m})$ 可能并不存在。到底什么是除法? 幸运的是, 我们还有以下规则:

命题 3.2. 消去律

设 $a, b, c \in \mathbb{Z}$, m 是一个正整数。如果 $\gcd(c, m) = 1$, 且 $ac \equiv bc \pmod{m}$, 则 $a \equiv b \pmod{m}$ 。

证明 根据同余的定义和条件, 有 $m | (ac - bc)$, 等价地, $m | (a - b)c$ 。因为 $\gcd(c, m) = 1$, 根据推论 5.1, 则 $m | (a - b)$, 即得。□

消去律本质是什么? 由方程 $ac \equiv bc \pmod{m}$ 推导出 $a \equiv b \pmod{m}$, 表面上看我们需要的是除法, 实际上需要的是乘法。其过程是, 先求出一个记为 c^{-1} 的值, 它满足

$$cc^{-1} \equiv 1 \pmod{m}$$

然后将它乘在同余式的两边:

$$acc^{-1} \equiv bcc^{-1} \pmod{m},$$

因此得到:

$$a \equiv b \pmod{m}.$$

这个 c^{-1} 被称为 c 在模 m 下的乘法逆元 (multiplicative inverse)。注意, 既然强调是模 m 下的乘法逆元, 就只考虑小于 m 的 c^{-1} , 或者所有同余的逆元都视为同一个逆元。在不引起误会的场合, 为了表述方便, 往往会省略模 m 这个限定语。

例 3.4 设 $m = 11$, $c = 3$, 那么 c 模 m 的乘法逆元 c^{-1} 是 4, 因为 $3 * 4 \pmod{11} = 1$ 。虽然, 15 也满足 $3 * 15 \pmod{11} = 1$, 但是 $15 \equiv 4 \pmod{11}$, 所以, 还是认为乘法逆元是 4。

给定互素的 c 和 m , c^{-1} 是一个唯一确定的值。但是, c^{-1} 会必然存在吗? 必然存在, 因为消去律的前提条件为 $\gcd(c, m) = 1$ 。根据 Bézout 定理 (定理 2.2), c^{-1} 必然存在且唯一。具体证明留给读者作为练习。这样, 至少我们已经可以部分地解决了同余方程 $ax \equiv b \pmod{m}$ 的求解问题。

例 3.5 求解 $3x \equiv 2 \pmod{11}$ 。首先, 使用 *egcd* 算法 (算法 2.3) 计算模 11 下 $3^{-1} = 4$ 。具

体而言，是算出 Bézout 系数 s 和 t 使得

$$3s + 11t = 1$$

此时， $s = 4$ ， $t = -1$ ，然后对以上等式两边同时模整数 11，则可得：

$$3 * 4 \equiv 1 \pmod{11}$$

即，3 模 11 的乘法逆元是 4。因此，对所需求解的同余方程两边同乘 3 的乘法逆元 4，得到 $x \equiv 8 \pmod{11}$ 。

如果要完全求解这一类同余方程，则需要更多的分析。由 $ax \equiv b \pmod{m}$ ，得到 $ax - b = ym$ ， $y \in \mathbb{Z}$ ，等价地有：

$$ax - my = b \quad (3.4)$$

希望以上等式能让你想到 Bézout 定理 (定理 2.2)。此时，存在 $x_0, y_0 \in \mathbb{Z}$ 使得：

$$\gcd(a, m) = ax_0 + my_0.$$

接着，需要考虑两种情形：



思考

- 什么情况下同余方程无解？
- 如果同余方程有解，会有多少个模 m 不同余的解？

建议读者自己完成以上提示的分析。

命题 3.3

设 p 为素数，且 $a \in \mathbb{N}$ 。则 $a^2 \equiv 1 \pmod{p}$ 当且仅当 $a \equiv 1 \pmod{p}$ 或 $a \equiv -1 \pmod{p}$ 。



证明 充份性的证明简单。对于必要性，如果 $a^2 \equiv 1 \pmod{p}$ 则 $p \mid (a^2 - 1)$ ，这意味着 $p \mid (a - 1)$ 或 $p \mid (a + 1)$ ，等价于 $a \equiv 1 \pmod{p}$ 或 $a \equiv -1 \pmod{p}$ 。□

命题 3.3 本质上是提醒我们，在模数为素数的情形下，什么数的乘法逆元是自己本身。或者，不准确地说，在模素数的情形下，对 1 开根号，得到一个 1 和一个 -1。难道如果模数不是素数会有什么不同？以后再说。

命题 3.4

同余关系是一种等价关系。



证明 容易，请读者自行完成。□

如果有一种等价关系定义在集合 S 上，则该等价关系会把集合 S 划分为不相交的子集，这些子集我们称为等价类 (equivalence classes)。集合中任意两个元素落在相同的等价类中当且仅当这两个元素等价。

考虑整数集合 \mathbb{Z} ，设 m 为正整数，则模 m 的同余关系把 \mathbb{Z} 划分为 m 个等价类。任意整数 a 所落在的等价类称为 $(a \bmod m)$ 的同余类 (congruence class)，或者称为剩余类 (residue class)，记为 $[a]_m$ ，即所有模 m 同余的整数形成的集合，或者所有具有以下形式的整数的集合： $a + km$ ， $k \in \mathbb{Z}$ 。

模 m 同余类的集合记为 $\mathbb{Z}/m\mathbb{Z}$, 该集合恰好有 m 个元素, 即:

$$\mathbb{Z}/m\mathbb{Z} = \{[0]_m, [1]_m, \dots, [m-1]_m\}.$$

例 3.6 如果 $m = 2$, $\mathbb{Z}/2\mathbb{Z} = \{[0]_2, [1]_2\}$ 。同余类 $[1]_2$ 就是所有模 2 余 1 的整数, 即奇数。同理, $[0]_2$ 则是所有偶数的集合。

称同余类 $[a]_m$ 中的任意一个元素 b 为代表元 (*representative*)。通常, 使用一个同余类中最小非负元素作为代表元来表示一个同余类, 但是在特定情况下选用某些特定的元素也许会更方便。 $\mathbb{Z}/m\mathbb{Z}$ 的所有最小非负代表元的集合 $\{0, 1, 2, \dots, m-1\}$ 被称为模 m 的最小剩余系 *least residue system*。任意 m 个整数的集合, 如果其中任意两个元素都不模 m 同余, 则称之为模 m 的完全剩余系 (*complete residue system*)。

例 3.7 设 $m = 7$, 模 m 的最小剩余系就是集合

$$\{0, 1, 2, 3, 4, 5, 6\},$$

下面这个集合则是模 m 的一个完全剩余系:

$$\{14, 8, 23, 46, 61, 13\}.$$

使用同余类的表达, 命题 3.1 可简洁地重写为如下形式:

命题 3.5

如果 $[a_1]_m = [a_2]_m$ 且 $[b_1]_m = [b_2]_m$, 则

$$[a_1 \pm b_1]_m = [a_2 \pm b_2]_m \quad \text{且} \quad [a_1 b_1]_m = [a_2 b_2]_m.$$

证明 把 $[a]_m = [b]_m$ 变形为 $a \equiv b \pmod{m}$, 然后使用命题 3.1, 立得。 \square

3.2 二进制补码

本节重点考虑模算术运算中的减法。对于整数 a, b 和 m , 以下等式:

$$a - b \pmod{m}$$

等价于

$$a + (-b) \pmod{m}$$

其中, $-b$ 为一个带符号数 (*signed number*)。

计算机中表示带符号数的最常用方法就是二进制补码 (*two's complement*)。使用模算术运算, 可以清晰地解释二进制补码的设计原理。

先回顾一下二进制补码的常规表述。假定需要表达一个 n 比特的带符号数, 最高位比特称为符号位, 用于表示正或负, 其余 $n-1$ 个比特用于表达数值。 n 比特带符号数的范围是 $[-2^{n-1}, 2^{n-1}-1]$, 其中正整数的表达范围是 $[0, 2^{n-1}-1]$, 而负整数的范围则是 $[-2^{n-1}, -1]$ 。相关表达规则如下:

- 正整数按常规二进制数表达存储在 $n-1$ 个低位比特中, 其符号位置为 0。
- 负整数的表达分为三个步骤。设 $1 \leq x \leq 2^{n-1}$, 要表达负整数 $-x$: 首先构造正整数 x 的二进制补码表达 (规则如上), 然后对 x 的所有比特进行一次比特翻转 (0

比特翻转为 1; 1 比特翻转为 0), 最后对所得的 n 比特加 1 (一次二进制加法)。

例 3.8 设带符号数的比特长度 $n = 8$, 计算 -1 和 -128 的二进制补码。

$$\begin{aligned} +1 &= 00000001 \\ \text{比特翻转} & 11111110 \\ \text{加 1} & 00000001 \\ -1 &= 11111111 \end{aligned}$$

$$\begin{aligned} +128 &= 10000000 \\ \text{比特翻转} & 01111111 \\ \text{加 1} & 00000001 \\ -128 &= 10000000 \end{aligned}$$

大部分刚接触二进制的初学者都会懂得这种“比特翻转再加 1”的二进制补码操作, 然而很少人会理解, 负数为何如此表达。常规的教科书会告诉读者, 这是为了效率, 为了避免出现 -0 等等。以下给出另一种解释二进制补码的方式。对 n 比特的正整数 x ($1 \leq x \leq 2^{n-1} - 1$) 取反所得的负整数 $-x$ 就是 $2^n - x$ 。

从目标出发, 所谓求正整数 x 的负值, 就是求得某个整数 x' 使得 $x + x' = 0$, 即:

$$x + x' \equiv 0 \pmod{2^n}$$

根据模算术运算的规则, 既然 $x < 2^n$, 那么 $x' = 2^n - x$ 。又因为 $2^n - x = \underbrace{111 \cdots 11}_n + 1 - x$, 且 $\underbrace{111 \cdots 11}_n - x$ 就等于对 x 的所有比特取反。所以, x' 的值就等于对 x 的所有比特取反之后再加 1。

有了二进制补码, 模算术加法和模算术减法的实现就完全地统一起来了。换言之, 使用模加法和某些逻辑操作可实例化出模减法。

例 3.9 计算 $(5 - 27) \bmod 2^8$ 。

- 分别写出 5 和 27 的二进制表达为: $0b00000101$ 和 $0b00011011$;
- -27 的二进制补码表达是 $0b11100101$; 检验可知 $0b00011011 + 0b11100101 = 0b00000000$;
- 则计算 $5 + (-27)$ 就是计算 $0b00000101 + 0b11100101 = 0b11101010$; 检验可知 $0b11101010$ 就是十进制的 -22 。

考虑到, 计算机内部的所有运算都是定长计算, n 比特整数的运算必然是模 2^n 的运算, 这是考虑表达负数时的前提条件。那么把负整数 $-x$ 理解为 $2^n - x$ 就不能算是对二进制补码的“翻转加 1”的解释, 而是正本清源罢了。

3.3 模指数运算

这一节关注模指数运算，一种非常重要的算术计算原语。所谓模指数运算，就是输入整数 x 、 y 和 m ，计算：

$$x^y \bmod m。$$

SageMath 给出命令 `pow` 来计算模指数，或者可以直接用 `^` 和 `%` 运算符。以下是具体实例。

```
1 sage: pow(2, 10, 1000)
2 24
3 sage: 2^10 % 1000
4 24
5 sage: pow(2, 1024, 3^100)
6 426135005318503491421173769021520445807198574681
```

完成模指数运算最简单的方法就是重复地乘 $x \bmod m$ 。但是，如果利用计算过程的中间数值，计算效率可以大大提高。该过程重复对 x 进行平方，例子3.10展示了这种思路。

例 3.10 求 $2^{16} \bmod 11$ 。计算过程：

$$2^2 \bmod 11 = 4$$

$$2^4 \bmod 11 = 4 * 4 \bmod 11 = 5$$

$$2^8 \bmod 11 = 5 * 5 \bmod 11 = 3$$

$$2^{16} \bmod 11 = 3 * 3 \bmod 11 = 9$$

以上例子中的指数恰好是 2 的乘方，如果不是怎么办？不难处理，该计算过程可以表达为以下递归过程，具体实现参见算法3.3的描述。由此，模指数运算的复杂性从 $O(y)$ 大大地提高到 $O(\log(y))$ 。

$$x^y = \begin{cases} (x^{\lfloor y/2 \rfloor})^2 & \text{如果 } y \text{ 是偶数;} \\ x \cdot (x^{\lfloor y/2 \rfloor})^2 & \text{如果 } y \text{ 是奇数。} \end{cases} \quad (3.5)$$

```
1 # Recursive Function to calculate
2 # (x^y)%p in O(log y)
3 def rec_mod_exp(x, y, p):
4     if (y == 0): return 1
5     z = rec_mod_exp(x, y/2, p)
6     if ((y & 1) == 0): #y is an even number
7         return z*z % p
8     else:             #y is an odd number
9         return x*z*z %p
```

算法 3.1: 模指数运算--递归版本

接下来展示如何把递归算法转换为迭代算法。首先，把整数 y 视为多项式（或者一个二进制数串）：

$$y = y_{n-1}2^{n-1} + y_{n-1}2^{n-1} \cdots + y_1 2 + y_0,$$

其中 $y_i \in \{0, 1\}$ 。

其次，将 x^y 变形为：

$$x^y = \prod_{i=0}^{n-1} x^{y_i 2^i}$$

最后，从 $i = 0$ 开始不断计算 $x^{2^i} \bmod m$ ，判断如果 $y_i = 1$ ，则把该值乘进最终结果。注意，如果 $y_i = 0$ ， $x^{y_i 2^i}$ 等于 1。以下例子展示了其具体计算过程。

例 3.11 设 $x = 7$ ， $y = 10$ ， $m = 11$ ，计算 $x^y \bmod m$ 。 y 的二进制表达为 1010，因此计算如下：

$$y_0 = 0, \quad x^{2^0} \equiv 7 \bmod m$$

$$y_1 = 1, \quad x^{2^1} \equiv 5 \bmod m$$

$$y_2 = 0, \quad x^{2^2} \equiv 3 \bmod m$$

$$y_3 = 1, \quad x^{2^3} \equiv 9 \bmod m$$

然后，将 $y_i = 1$ 的项乘起来，得到 $x^y = (5 * 9) \bmod 11 = 1$ 。

该过程可描述为以下算法。

```

1 # Iterative Function to calculate
2 # (x^y)%p in O(log y)
3 def mod_exp(x, y, p) :
4     # Initial result
5     res = 1
6     # view y as a binary string
7     # begin with y's least significant bit
8     while (y > 0) :
9         # If lsb of y is 1, multiply x with result
10        if ((y & 1) == 1) :
11            res = (res * x) % p
12        # Divide y (y = y >> 1) to get y's next bit.
13        y = y / 2
14        # Repeatedly compute x^{2^i}, 1 <= i <= log y
15        x = (x * x) % p
16    return res

```

算法 3.2: 模指数运算--迭代版本

该算法的正确性与效率都是显然的。值得一提的是，如果读者已经写出了简单乘法的迭代版本，这两种算法的类似之处就非常显然了。模指数运算的优化改进算法则不在本书讨论范围。

❧ 第三章 习题 ❧

1. 编写一个 C 语言程序，用于验证在 C 语言中，整数的加法、减法、乘法和除法都是模 2^n 的模运算， n 根据变量定义不同可能是 8、32 或者 64 等。
2. 请证明命题 3.1。
3. 给定整数 p 和 N ，且 $p \mid N$ 。证明：对任意整数 x ，

$$(x \bmod N) \bmod p = (x \bmod p)$$

给出反例说明 $(x \bmod p) \bmod N$ 并不一定等于 $(x \bmod N)$ 。

4. 请证明同余关系是等价关系，即证明命题 3.4。
5. 手动计算以下模 m 下 a 的乘法逆元。(a) $m = 11$, $a = 5$; (b) $m = 121$, $a = 13$; (c) $m = 1021$, $a = 131$ 。
6. 编写 C 语言程序完成模指数运算，即给定整数 x , y 和 m 为输入，计算并返回值 $x^y \bmod m$ 。
7. 给定互素的正整数 c 和 m ，请证明 c^{-1} 是存在且唯一确定的值，它使得 $cc^{-1} \equiv (\bmod m)$ 。
8. 思考另一个版本的消去律。设 $a, b, c \in \mathbb{Z}$, m 是一个正整数。如果 $\gcd(c, m) = g$, g 是大于 1 的整数，且 $ac \equiv bc (\bmod m)$ 。请问此时能消去 c 得到某个同余式 $a \equiv b (\bmod m')$ 吗？这个 m' 会是什么？给出证明。
9. 完成同余方程求解的一般性分析。设 $a, b, m \in \mathbb{Z}$ ，求解 $ax \equiv b (\bmod m)$ ，其中 $g = \gcd(a, m)$ 。
 - 同余方程在什么情况下无解？
 - 如果同余方程有解，会有多少个模 m 不同余的解？
10. 编写一个 Python 程序求解以下同余方程：

$$ax \equiv b (\bmod m).$$

即，给定整数 a , b 和 m 作为输入，返回所有满足方程的解 x ，或者给出无解告警。

11. 编写一个 Python 程序计算乘法逆元，即输入互素的正整数 c 和 m ，返回 c^{-1} ，它使得 $cc^{-1} \equiv (\bmod m)$ 。要求：只返回为正整数的 c^{-1} 。

第四章 费尔马小定理和欧拉定理

内容提要

□ 费尔马小定理

□ 欧拉 Phi 函数

□ 欧拉定理

4.1 费尔马小定理

这一节，我们试图从模乘法和模指数运算中发现一些规律。先看这一种情况，给定素数 p 和一个正整数 a ，然后观察 $a \bmod p, a^2 \bmod p, a^3 \bmod p, \dots$ 。你能找出其中的一些规律吗？也许你可以尝试使用以下代码。

```
1 # 输入：整数n；
2 # 输出：i^j mod n, 其中1 <= i, j <= n - 1
3 def prt_pow_list(n):
4     for i in range(1, n):
5         for j in range(1, n):
6             print(pow(i, j, n), ' ', end='') #输出 i^j mod n
7         print()
8     return
```

例 4.1 分别设 p 等于素数 3、5、7 等，运行以上代码，得到：

a	a^2
1	1
2	1

表 4.1: $a^i \bmod 3$

a	a^2	a^3	a^4
1	1	1	1
2	4	3	1
3	4	2	1
4	1	4	1

表 4.2: $a^i \bmod 5$

a	a^2	a^3	a^4	a^5	a^6
1	1	1	1	1	1
2	4	1	2	4	1
3	2	6	4	5	1
4	2	1	4	2	1
5	4	6	2	3	1
6	1	6	1	6	1

表 4.3: $a^i \bmod 7$

你能找到其中的一些规律吗？可通过设 $p = 11$ 或者其他数值，运行代码来验证你自己的猜想。在 $p = 11$ 的情形下，我们得到表 4.4。

请注意这些表格的最后一列，容易得到这样一种猜想：

$$a^{p-1} \equiv 1 \pmod{p} \quad (4.1)$$

在进一步分析之前，还可以尝试其他一些运算，比如，使用以下代码进行运算。

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
1	1	1	1	1	1	1	1	1	1
2	4	8	5	10	9	7	3	6	1
3	9	5	4	1	3	9	5	4	1
4	5	9	3	1	4	5	9	3	1
5	3	4	9	1	5	3	4	9	1
6	3	7	9	10	5	8	4	2	1
7	5	2	3	10	4	6	9	8	1
8	9	6	4	10	3	2	5	7	1
9	4	3	5	1	9	4	3	5	1
10	1	10	1	10	1	10	1	10	1

表 4.4: $a^i \bmod 11$

```

1 #输入：整数n;
2 # 输出： i*j mod n, 其中 1 <= i, j <= n - 1
3 def prt_mul_list(n):
4     for i in range(1, n):
5         for j in range(1, n):
6             print( i*j % n, ' ', end='') # i*j modulo n
7         print()
8     return

```

这个程序做的是模乘法而不是模指数，对所有的 $1 \leq i < p$ ，计算 $a * i \pmod{p}$ 。比如，设 $a = 2$ ， $p = 7$ ，得到表4.5。

$a * i \backslash i$	1	2	3	4	5	6
a						
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

表 4.5: $a * i \bmod 7$

观察表4.5，一种非常容易发现的规律就是：所有行的数值恰好是 1 到 6 之间的整数，即只是对 1 到 6 之间的整数进行一次位置置换。把这种初始的猜想写成以下引理。

引理 4.1

如果 p 是一个素数，任取整数 $1 < a < p$ ，对所有的 $1 \leq i < p$ ，计算 $a * i \bmod p$ 所得的数值是 1 到 $p - 1$ 之间所有整数的一次置换。



引理4.1本质上就是说，以下集合

$$\mathbb{S} = \{a * i \bmod p, 1 \leq i < p\}$$

与以下集合是同一个集合：

$$\{1, 2, 3, \dots, p-1\}$$

值得一提的是，集合 S 中的元素刚好形成一个模 p 的完全剩余系，也是最小剩余系。

证明 使用反证法。如果命题不成立，则存在两个不同的 i 和 j ，其中 $1 \leq i, j < p$ ，使得，

$$a * i \equiv a * j \pmod{p}$$

又因为 $1 < a < p$ ，且 p 是素数，所以 $\gcd(a, p) = 1$ 。所以，根据消去律（命题3.2），等式两边可以消去 a ，得到：

$$i \equiv j \pmod{p},$$

与假设矛盾！□

再接着分析之前的猜想，见等式4.1。此时我们需要计算 $a^{p-1} \bmod p$ ，即 $p-1$ 个 a 的模乘。注意到，我们计算得到的表格中的每一个元素都隐藏了一个 a 。这提示我们做以下简单的计算工作，将每一行中的 $a * i \bmod p$ 乘起来。引理4.1告诉我们，这本质上就是把所有的 $1 \leq i < p$ 乘起来。所以：

$$\prod_{i=1}^{p-1} i = \prod_{i=1}^{p-1} (a * i)$$

显然有：

$$\prod_{i=1}^{p-1} i \equiv \prod_{i=1}^{p-1} (a * i) \equiv a^{p-1} \prod_{i=1}^{p-1} i \pmod{p}$$

把这个大数字 $\prod_{i=1}^{p-1} i$ ，即 $(p-1)!$ ，从等式两边消去，得到：

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

当然，在这么做之前，请确保说服自己 $(p-1)!$ 这个大数字确实与 p 互素。最后，结果可以表达为以下定理，这就是颇著名的费尔马小定理。

定理 4.1. 费尔马小定理

设 p 是素数，设 a 为任意整数，且 $a \not\equiv 0 \pmod{p}$ ，则

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



例 4.2 设 $p = 17$ ， $a = 3$ ，计算 $a^{2018} \pmod{p}$ 。

$$3^{2018} \equiv 3^{65*16+2} \pmod{17},$$

因为 $65 * 16 = 65 * (17 - 1)$ ，所以

$$3^{65*16+2} \equiv 3^2 \equiv 9 \pmod{17}.$$



思考 以上分析、证明与最终的定理的表述有一点微小的差别，希望大家能察觉到。在引理4.1中，要求 $1 < a < p$ ；在定理4.1中， a 则是任意整数，只是要求 $a \not\equiv 0 \pmod{p}$ 。这两种要求是等价的吗？为什么是，或为什么不是呢？

4.2 欧拉定理

在上一节的分析中，模数是一个素数，借助消去律，得到了费尔马小定理。如果模数是合数，则消去律就可能失效了。比如，设 $n = 6$ ，借助之前的代码，可以得到以下数据。

$a * i \backslash i$ a	1	2	3	4	5
1	1	2	3	4	5
2	2	4	0	2	4
3	3	0	3	0	3
4	4	2	0	4	2
5	5	4	3	2	1

表 4.6: $a * i \bmod 6$

再看另一个实例，设 $n = 9$ ，得到：

$a * i \backslash i$ a	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2	4	6	8	1	3	5	7
3	3	6	0	3	6	0	3	6
4	4	8	3	7	2	6	1	5
5	5	1	6	2	7	3	8	4
6	6	3	0	6	3	0	6	3
7	7	5	3	1	8	6	4	2
8	8	7	6	5	4	3	2	1

表 4.7: $a * i \bmod 9$

观察可知，如果 n 是一个合数，则集合

$$\{a * i \bmod n, 1 \leq i < n\}$$

并不等于集合：

$$\{1, 2, 3, \dots, n-1\}$$

除非 a 与 n 互素，即 $\gcd(a, n) = 1$ 。即前一个集合是后一个集合的置换的这种属性在 a 与 n 不互素时被破坏了，表 4.7 中标记为绿色的两行展示了这个事实。请读者自己认真检验这一观察结果，或者做更多的计算并观察。

然而，从乐观的角度看，置换属性还是有可能得到保持的。将不互素于 n 的“坏家伙”从表格中剔除，将得到表 4.8。

$a * i \backslash i$ a	1	2	4	5	7	8
1	1	2	4	5	7	8
2	2	4	8	1	5	7
4	4	8	7	2	1	5
5	5	1	2	7	8	4
7	7	5	1	8	4	2
8	8	7	5	4	2	1

表 4.8: $a * i \bmod 9$ (要求 a 和 i 都与 9 互素)

设 n 为正整数, 记

$$\mathbb{S} = \{b : 1 \leq b < n \text{ 且 } \gcd(b, n) = 1\}$$

任取与 n 互素的整数 a , 记:

$$\mathbb{S}' = a * \mathbb{S} \bmod n$$

我们猜想:

$$\mathbb{S} = \mathbb{S}'$$

为方便讨论, 给出欧拉 *Phi* 函数¹ (Euler's phi function) 的定义。

定义 4.1. 欧拉 Phi 函数

欧拉 *Phi* 函数 ϕ 对任意正整数 n , 返回 1 到 n 之间且与 n 互素的整数个数, 即:

$$\phi(n) = |\{b : 1 \leq b < n \text{ 且 } \gcd(b, n) = 1\}|.$$



例如, 容易看出:

$$\phi(1) = |\{1\}| = 1,$$

$$\phi(2) = |\{1\}| = 1,$$

$$\phi(7) = |\{1, 2, 3, 4, 5, 6\}| = 6,$$

$$\phi(9) = |\{1, 2, 4, 5, 7, 8\}| = 6.$$

集合 \mathbb{S} 中的元素即为 1 到 n 之间且与 n 互素的整数, $\phi(n)$ 即为集合 \mathbb{S} 的阶 (order)。于是, 集合 \mathbb{S} 和 \mathbb{S}' 可分别表示为:

$$\mathbb{S} = \{b_1, b_2, \dots, b_{\phi(n)} : 1 \leq b_i < n \text{ and } \gcd(b_i, n) = 1\}$$

$$\mathbb{S}' = \{a * b_i \bmod n : b_i \in \mathbb{S} \text{ and } \gcd(a, n) = 1\}$$

我们的目标是要证明:

$$\mathbb{S} = \mathbb{S}'$$

与上一个证明类似, 如果存在不相同的 b_i 和 b_j 使得:

$$a * b_i \equiv a * b_j \pmod{n}$$

¹Phi 发音为 fai

则根据消去律得：

$$b_i \equiv b_j \pmod{n}$$

这与条件假设矛盾。

再次类似地，将集合 S 和 S' 中的元素分别累积，得到：

$$\prod_{i=1}^{\phi(n)} b_i = \prod_{i=1}^{\phi(n)} a * b_i$$

等式两边模 n ，得到：

$$\prod_{i=1}^{\phi(n)} b_i \equiv \prod_{i=1}^{\phi(n)} a * b_i \pmod{n}$$

利用乘法的结合律，简单整理变形得：

$$\prod_{i=1}^{\phi(n)} b_i \equiv a^{\phi(n)} \prod_{i=1}^{\phi(n)} b_i \pmod{n}$$

再次使用消去律，消去 $\prod_{i=1}^{\phi(n)} b_i$ ，得到：

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

最后，将该结论写成以下定理。

定理 4.2. 欧拉定理

设 n 和 a 为正整数，且 $\gcd(a, n) = 1$ ，则：

$$a^{\phi(n)} \equiv 1 \pmod{n}。$$



4.3 欧拉 Phi 函数

上一节，我们定义了欧拉 Phi 函数，因为该函数颇为重要，本节进一步讨论其相关性质。

SageMath 给出命令 `euler_phi` 来计算欧拉 Phi 函数，也许通过一些简单计算，读者也可看出某些规律。

```
1 sage: euler_phi(11)
2 10
3 sage: euler_phi(29)
4 28
5 sage: euler_phi(29*11)
6 280
```

比如，很容易可看出以下命题。



命题 4.1. 欧拉 Phi 函数性质

如果 p 是素数, 则 $\phi(p) = p - 1$, $\phi(p^k) = p^k - p^{k-1}$.



证明 容易, 简单计数可得, 因为大于 1, 小于等于 p^k 且与 p 不互素的数就是所有 p 的倍数: $p, 2p, \dots, (p^{k-1} - 1)p, p^{k-1}p$. \square

本节的主要任务要计算 $\phi(mn)$, 其中, m 和 n 是互素的正整数. 分析分为两大步骤. 第一步, 使用以下方式枚举并排列出所有不超过 mn 的正整数.

1	$m + 1$	$2m + 1$	\dots	$(n - 1)m + 1$
2	$m + 2$	$2m + 2$	\dots	$(n - 1)m + 2$
3	$m + 3$	$2m + 3$	\dots	$(n - 1)m + 3$
\dots	\dots	\dots	\dots	\dots
r	$m + r$	$2m + r$	\dots	$(n - 1)m + r$
\dots	\dots	\dots	\dots	\dots
m	$2m$	$3m$	\dots	mn

第二步, 找出所有同时与 n 和 m 互素的元素, 则它们与 mn 互素. 这种思想可表达为以下引理.

引理 4.2

对任意 $a \in \mathbb{Z}$, 如果 $\gcd(a, n) = 1$ 且 $\gcd(a, m) = 1$, 则 $\gcd(a, mn) = 1$.



证明 反证法. 假设 $\gcd(a, n) = 1$ 且 $\gcd(a, m) = 1$, 但是 $\gcd(a, mn) = d$, 且 $d > 1$. 则存在素数 $p > 1$ 使得 $p \mid d$, 因此 $p \mid a$ 且 $p \mid mn$. 因为 p 是素数, 则有 $p \mid m$ 或 $p \mid n$. 所以, p 是 a 与 m 的公因子或者是 a 和 n 的公因子. 矛盾! \square

为了完成第二步的分析, 继续使用计数法:

1. 对任意 $r \in [0..m]$, 有多少数字满足 $\gcd(r, m) = 1$? 答案是 $\phi(m)$. 注意, 如果 $\gcd(r, m) = 1$, 则对任意 $k \in [0..n - 1]$, 有 $\gcd(km + r, m) = 1$. 因此有 $\phi(m)$ 行数字与 m 互素.
2. 对任意第 r 行, 对任意 $k \in [0..n - 1]$, 有多少整数满足 $\gcd(km + r, n) = 1$? 答案是: $\phi(n)$. 因为, 第 r 行的元素是: $r, m + r, \dots, (n - 1)m + r$ 且 $\gcd(m, n) = 1$. $\forall k_i \neq k_j, k_i m + r \not\equiv k_j m + r \pmod{n}$. 否则根据消去律, 有 $k_i = k_j$, 矛盾. 这意味着第 r 行的 n 个元素恰好形成一个模 n 的完全剩余系, 即 $\{r, m + r, \dots, (n - 1)m + r\} \pmod{n} = \{0, 1, 2, \dots, n - 1\}$. 因此, 恰有 $\phi(n)$ 个元素与 n 互素.

综上分析, 有 $\phi(m)$ 行元素与 m 互素, 其中每一行有 $\phi(n)$ 个元素与 n 互素, 共有 $\phi(m)\phi(n)$ 个元素与 mn 互素. 该结论可表述为以下定理.

定理 4.3. 欧拉 Phi 函数公式

设 m 和 n 是互素的正整数, 则 $\phi(mn) = \phi(m)\phi(n)$.



因为这个属性, 欧拉 Phi 函数也被称为积性函数 (*multiplicative function*).

如果 p 和 q 都是素数, 则对所有正整数 i 和 j , 有 $\gcd(p^i, q^j) = 1$ 。根据定理4.3, 有 $\phi(p^i q^j) = \phi(p^i) \phi(q^j)$ 。所以, 有以下推论。

推论 4.1

设 m 为正整数, 且将其表达为素数指数的乘积, 即 $m = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$, 所有的 p_i 都是素数。则:

$$\phi(m) = \phi(p_1^{a_1}) \cdots \phi(p_k^{a_k})。$$



证明 留作课后习题。 □

定理 4.4

设 m 为正整数, 且将其表达为素数指数的乘积, 即 $m = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$, 所有的 p_i 都是素数。则:

$$\phi(m) = m \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_k}\right)。$$



证明 留作课后习题。 □

第四章 习题

1. 设 $p = 23$ 和 $a = 3$, 使用费尔马小定理计算 $a^{2019} \bmod p$?
2. 使用费尔马小定理求解同余方程 $x^{50} \equiv 2 \pmod{17}$ 。
3. 给定素数 p , 根据费尔马小定理, 任取与 p 互素的正整数 a , 则 $a^{p-1} \equiv 1 \pmod{p}$, 请问 $a^{(p-1)/2} \bmod p$ 等于什么? 为什么?
4. 使用欧拉定理计算 $2^{100000} \bmod 55$ 。
5. 心算回答: $\phi(3^3)$ 等于? $\phi(2^{10})$ 等于几?
6. 手动计算 7^{1000} 的最后两个数位等于什么?
7. 编写 Python 语言程序完成欧拉 Phi 函数的计算, 即输入正整数 n , 计算并返回 $\phi(n)$ 。
8. 设 p 是素数, 计算 $(p-1)! \bmod p$, 并找出规律, 写成定理, 并给出证明。
9. 分别给出推论4.1和定理4.4的完整证明。
10. 设整数 $n = p^k$ 是某个素数 p 的指数, n 的所有因子分别为 d_0, d_1, \cdots 和 d_r (包括 1 和 n)。请证明:

$$\sum_{i=0}^r \phi(d_i) = n。$$

第五章 素数与素性测试

内容提要

□ 素数的若干定理

□ 米勒测试

□ Carmichael 数

□ 米勒-拉宾算法

素数在数学中类似空气存在于地球，无处不在而且非常重要。本章首先简要地陈述素数的若干性质，为以后章节铺垫基础，然后重点讲解一种判断整数是否素数的算法。

5.1 素数

以下列举若干关于素数的定理与结论，都是常识。

定理 5.1. 素数整除性

设 $a, b \in \mathbb{Z}$, p 为素数。如果 $p \mid ab$, 则 $p \mid a$ 或 $p \mid b$ 。



证明 假设 $p \nmid a$, 证明 $p \mid b$ 。因为 $p \nmid a$ 且 p 是素数, 则 $\gcd(a, p) = 1$, 即存在 $r, s \in \mathbb{Z}$ 使得 $ar + ps = 1$ 。因此,

$$b = b(ar + ps) = abr + pbs.$$

既然 $p \mid ab$ (假设条件), 那么 p 同时整除 abr 和 pbs , 则 p 必然整除 b 。

推论 5.1

设 $a, b, n \in \mathbb{Z}$, $n > 0$ 。如果 $n \mid ab$ 且 $\gcd(a, n) = 1$, 则 $n \mid b$ 。



定理 5.2. 算术基本定理

每一个自然数都可唯一地表达为素数的乘积。



定理 5.3. 欧几里德定理

素数有无穷多。



证明 使用反证法。假设只存在有限多的素数, 比如 p_1, p_2, \dots, p_n , 记 $N = p_1 p_2 \cdots p_n + 1$ 。根据条件假设, N 必为合数, 所以存在某个 p_i , $1 \leq i \leq n$, 使得 $p_i \mid N$ 。则 p_i 必定整除 $N - p_1 p_2 \cdots p_n = 1$, 但是这是不可能的。因此, 或者 N 是素数, 或者存在另一个素数 $q \neq p_i$, $1 \leq i \leq n$, $q \mid N$ 。

简单的计算可知, 素数分为两类, 一类模 4 余 1, 一类模 4 余 3。

命题 5.1. 模 4 余 3 的素数

模 4 余 3 的素数有无穷多。



证明 反证法。假设模 4 余 3 的素数有限多，可枚举出所有模 4 余 3 的素数构成以下集合：

$$S = \{3, p_1, p_2, \dots, p_n\}$$

令

$$P = 4p_1p_2 \cdots p_n + 3 \quad (5.1)$$

则 P 可被分解为一系列素数的乘积，

$$P = q_1q_2 \cdots q_m \quad (5.2)$$

如果所有的 q_i 都模 4 余 1，则 P 必然模 4 余 1，然而从 P 的构造可知， P 模 4 余 3，所以，必然存在某个 q_i 模 4 余 3。最后要证明 $q_i \notin S$ 。因为等式 5.2 说明， q_i 整除 P ，如果 $q_i \in S$ ，则 P 模 q_i 余 3，矛盾。于是，找到了一个新的模 4 余 3 的素数，假设不成立！□



思考 利用以上证明方法能证明模 4 余 1 的素数有无穷多吗？请大家尝试，并发现证明的难点。

以下的两个定理都属于常识，但是，并非所有的常识都容易证明。

定理 5.4. 狄利克雷定理.

设 $a, m \in \mathbb{Z}$ ，且 $\gcd(a, m) = 1$ ，则存在无穷多个模 m 同余 a 的素数。即有无穷多素数满足：

$$p \equiv a \pmod{m}.$$



欧几里德定理告诉我们，素数有无穷多，需要进一步问，到底有多少，其分布如何？通过简单的 Sage 命令，我们可以稍微建立起关于素数分别的基本认识。

```
1 sage: prime_pi(1000)
2 168
3 sage: prime_pi(10000)
4 1229
5 sage: prime_pi(65537)
6 6543
7 sage: plot(prime_pi, 1, 10000, rgbcolor=(0,0,1))
```

以下是用 Sage 命令画出一万以内整数中的素数分布图，大家可以尝试其他参数，取 1000 或者 100,000 等得到的效果图基本相同。

定理 5.5. 素数定理

函数 $\pi(x)$ 渐进于 $x/\log(x)$ ，即：

$$\lim_{x \rightarrow \infty} \frac{\pi(x)}{x/\log(x)} = 1$$



以上定理是难证明，需要使用解析数论、复分析等知识，本书无法给出证明。以下常识则属于暂时尚无法证明也无法证伪的猜想。

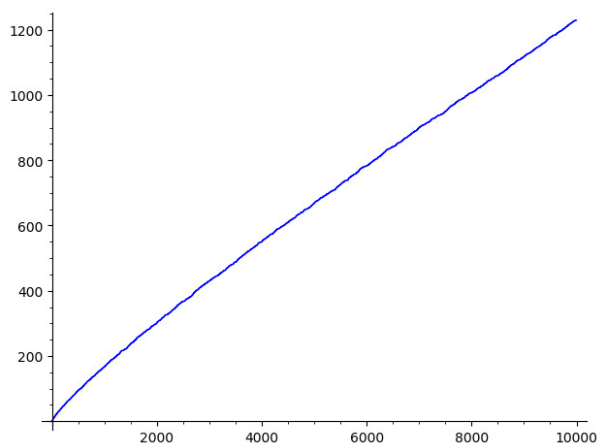


图 5.1: 一万以内整数的素数分布图

猜想 5.1. 歌德巴赫猜想任意的偶数 $n \geq 4$ 都是两个素数之和。**猜想 5.2. 孪生素数猜想**有无穷多的素数 p 使得 $p + 2$ 也是素数。**猜想 5.3. $N^2 + 1$ 猜想**有无穷多的素数形如 $N^2 + 1$, $N \in \mathbb{N}$ 。

5.2 素性测试

本节讨论素性测试问题，即如何判断一个大的奇整数 n 是否素数。先看以下 Sage 实例。

例 5.1

```

1 sage: is_prime(131)
2 True
3 sage: is_prime(12345678987654321)
4 False
5 sage: is_prime(969012308683094185386857784379)
6 True
7 sage: time is_prime(969012308683094185386857784379) #考察算法的执行时间
8 CPU times: user 6.18 ms, sys: 1.31 ms, total: 7.5 ms
9 Wall time: 7.57 ms
10 True

```

这个问题似乎并不难，因为大家都熟悉这样一种简单的方法—试除法 (*trial division*)，代码如下：

```

1 bool IsPrime(int n){
2     for(int i = 2; i * i <= n; i++){
3         if(n % i == 0) return false;
4     }
5     return n != 1
6 }

```

算法 5.1: 试除法

为什么我们对试除法并不满意？对特定的输入值 n ，试除法的复杂性是 $O(\sqrt{n})$ ，显然一个多项式时间算法。按常识而言，这是高效算法。然而，在这里需要用另一种方式看待算法的复杂性。

某些特定的算法，往往并不关心某个特定的输入值，而关注满足某一个规模的输入值，此时衡量算法的复杂性的输入就应该是输入值的数据长度。以素性测试问题为例，算法并不关心某个输入值 n 是否素数，而是关注某一类整数，比如 λ 比特的所有整数。此时衡量算法复杂性的参数应该是 λ 而不是 n 。因为试除法的复杂性是 $O(\sqrt{n})$ ，以 λ 为考量指标代入，则变成了 $O(\sqrt{2^\lambda})$ ，这是以 λ 为参量的指数式复杂性。

考虑具体的数值，如果要判断某个 100 比特的整数是否素数，用试除法的话，最多要做 2^{50} 次除法，使用一般的计算机很难算出答案。即使是超级计算机也不可能具有例子 5.1 中展示出来的秒杀速度。在算法理论中，把试除法这种貌似多项式时间的复杂性称为伪多项式时间 (*Pseudo-polynomial Time*)。所以，需要设计更高效的素性判定算法。

以下介绍一种最常用的高效的素性判定算法，其思想很简单，源自于费尔马小定理。费尔马小定理告诉我们，如果 p 是素数，对任意 $a \in \mathbb{Z}$ 且 $p \nmid a$ ，以下等式成立：

$$a^{p-1} \equiv 1 \pmod{p} \quad (5.3)$$

如果上式不成立，则 p 必然是合数。反过来，如果上式成立，不能论断 p 一定是素数，只能说没有发现证据证明 p 是合数，也就是， p 可能是素数。假设能以比较大的概率找到 p 是合数的证据，只需要选不同的 a ，不断判断等式 5.3 是否成立，如果多次判断而又找不到 p 是合数的证据，则说明 p 以很大的概率是一个素数。请大家相信概率！这样，通过判断等式 5.3 是否成立为判断一个整数是否素数提供了思路。

例 5.2 设 $p = 51$ ，计算可知：

$$2^{50} \equiv 4 \pmod{51}$$

所以，2 是 51 不是素数的一个证据。但是，

$$16^{50} \equiv 1 \pmod{51}$$

16 则提供一个“伪证”，让 51 看上去像一个素数。同时提供“伪证”还有 35。

存在伪证已经颇糟糕了，然而更糟糕的是，有可能存在大量的数是伪证。即对一个合数 n ，大部分的整数 $a < n$ 都使得等式 $a^{n-1} \equiv 1 \pmod{n}$ 成立。

定义 5.1. 伪素数与绝对伪素数

设 $n \in \mathbb{Z}$ 为合数, 如果存在 $a \in \mathbb{Z}$ 且 $\gcd(a, n) = 1$, 使得下式成立:

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

则称 n 是以 a 为基的伪素数 (Pseudoprime)。设 $n \in \mathbb{Z}$ 为合数, 对任意 $a \in \mathbb{Z}$ 且 $\gcd(a, n) = 1$, 上式成立, 则称 n 为 Carmichael 数, 或称绝对伪素数。



例 5.3 第一个 Carmichael 数是 $561 = 3 * 11 * 17$ 。可以验证, 对任意的 $a \in \mathbb{Z}$, 如果 $\gcd(a, 561) = 1$, 则 $a^{560} \equiv 1 \pmod{560}$ 。10000 之内的 Carmichael 数包括:

$$561, \quad 1105, \quad 1729, \quad 2465, \quad 2821, \quad 6601, \quad 8911$$

坏消息是, 存在无穷多个 Carmichael 数。

定理 5.6. Carmichael 数的判定性质

设 $n = q_0 q_1 \cdots q_k$ 是一个合数, 其中 $k > 1$, 对任意 i , q_i 是两两不同的素数且满足 $(q_i - 1) \mid (n - 1)$ 。则 n 是 Carmichael 数。



证明 使用中国剩余定理易得。留作第 10 章的作业。 □

伪素数的存在, 特别是 Carmichael 数的存在使得本节开始提供的思路遭到挑战。也许, 大家已经注意到, 对任意 Carmichael 数 n , 使得 $a^{n-1} \equiv 1 \pmod{n}$ 成立的整数 a 必须满足 $\gcd(a, n) = 1$ 。会不会存在这样的侥幸心理, 如果可以大概率地选中 $\gcd(a, n) \neq 1$ 的 a , 还是可以发现 Carmichael 数不是素数? 这也是一个概率问题。确实, 然而再稍微进一步分析, 满足 $\gcd(a, n) = 1$ 且小于 n 的整数有多少? 有 $\phi(n)$ 个, 其中 ϕ 是欧拉 Phi 函数。当 n 很大的时候, $\phi(n)$ 与 n 有可能很接近, 这样选中证据的概率就很小。算法有可能因此失效, 或者效率不高。所以, 如何设计算法避免接受 Carmichael 数为素数就是一个问题。

命题 3.3 告诉我们: 如果 n 为素数, 任取 $a \in \mathbb{Z}$, 则 $a^2 \equiv 1 \pmod{n}$ 当且仅当 $a \equiv 1 \pmod{n}$ 或 $a \equiv -1 \pmod{n}$ 。如果 $a^{n-1} \not\equiv 1 \pmod{n}$, 那么 a 已经完成任务, 证明了 n 是合数。但是, 如果 $a^{n-1} \equiv 1 \pmod{n}$, n 还可能以 a 为基的伪素数。因此, 进一步要判断是否 $a^{(n-1)/2} \equiv \pm 1 \pmod{n}$ 。如果这个同余式不成立, 则可判断 n 为合数。

注 在第 11 章第 11.3 节将会讨论在模某个整数的情况下对 1 开平方的计算方法, 那时我们将会知道模任意 Carmichael 数 n 对 1 开平方会有多个解。因此, 对任意基 a , $a^{(n-1)/2} \equiv \pm 1 \pmod{n}$ 不是必然不成立, 而是可能不成立。

例 5.4 设 $n = 561$, 这是第一个 Carmichael 数。取 $a = 5$,

$$a^{560} \equiv 1 \pmod{n}$$

但是,

$$a^{560/2} = a^{280} \equiv 67 \pmod{n}$$

因此, 561 是合数。

以上判断某个数是否素数的思路归结起来就是米勒测试过程, 以下定义描述了这个过程。



定义 5.2. 米勒测试

设 $n > 2$ 是一个奇素数，则 $n - 1$ 可表达为：

$$n - 1 = 2^k q$$

其中， q 是一个奇数， k 是某个正整数。设 a 是任意选取的整数，且 $\gcd(a, n) = 1$ 。

如果以下两个条件其中之一得到满足：

1. $a^q \equiv 1 \pmod{n}$
2. 存在某个 $0 \leq j \leq k - 1$ ，使得 $a^{2^j q} \equiv -1 \pmod{n}$

则称 n 通过了以 a 为基的米勒测试。

**定理 5.7. 米勒测试**

设 n 是奇素数，对任意满足 $\gcd(a, n) = 1$ 的整数 a ， n 必然会通过以 a 为基的米勒测试。



证明 因为 n 是素数，且对任意满足 $\gcd(a, n) = 1$ 的整数 a ，根据费尔马小定理，有：

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

米勒测试中涉及判断数列如下：

$$a^q, a^{2q}, a^{4q}, \dots, a^{2^{k-1}q} \pmod{n}$$

而且只可能出现两种情况：

1. 首先判断 $a^q \equiv 1 \pmod{n}$ 或 $a^q \equiv -1 \pmod{n}$ 是否成立，如果成立，则 n 通过米勒测试；
2. 否则判断是否 $a^{2^j q} \equiv -1 \pmod{n}$ 成立，成立则 n 通过米勒测试。请注意，前一种情况已经确保了 $a^{2^q} \not\equiv 1 \pmod{n}$

如此，按顺序迭代计算并判断是否 $a^{2^j q} \equiv -1 \pmod{n}$ 成立， $1 < j \leq k - 1$ 。因为 $a^{n-1} \equiv 1 \pmod{n}$ 必然成立， a^{n-1} 即 $a^{2^k q}$ ，所以以上的某次判断必然成立，即 n 必然会通过以 a 为基的米勒测试。 \square

以上证明也就给出了米勒测试算法的基本过程，以下代码给出了具体描述。

```

1 #Input: a是随机选取的基, n是要测试的整数, 而q是整数满足n-1 = 2^k q
2 #Output: 如果n是合数, 输出0; 否则n可能是素数, 输出1, 即通过米勒测试;
3 def MillerTest(a, q, n):
4     x = pow(a, q, n)
5     if (x == 1) or (x == (n - 1)):
6         return 1
7     while q != (n - 1):
8         x = (x * x) % n
9         q *= 2
10        if x == (n - 1):
11            return 1

```

```
12     return 0
```

算法 5.2: 米勒测试

n 不通过米勒测试, 则 n 必然是合数, 但是如果 n 通过米勒测试, 则只说明 n 可能是素数。那么, 是否存在整数 a 使得一个合数 n 可以通过米勒测试? 答案是肯定的, 请看下例。

例 5.5 设 $n = 2047 = 23 * 89$, 则

$$2^{2046} \equiv (2^{11})^{186} \equiv 2048^{186} \equiv 1 \pmod{2047}$$

这说明, 2047 是以 2 为基的伪素数。更糟糕的是, 以下等式也成立:

$$2^{1023} \equiv (2^{11})^{93} \equiv 2048^{93} \equiv 1 \pmod{2047}$$

也就是说, 2047 可以通过以 2 为基的米勒测试。

例 5.6 第一个 Carmichael 数 $n = 561$, 它可以通过 50 为基的米勒测试。因为 $(n-1) = 2^4 3^5$, 而:

$$50^{35} \equiv 560 \pmod{561}$$

还可以再找出 7 个小于 n 的基使得 n 通过米勒测试。

以上例子说明, 米勒测试可能会误判, 这样的算法有用吗? 此时就要提及拉宾的贡献了。拉宾首先注意到伪证据的数量, 如果这个数量很小, 则某个基会成为伪证的概率很小。于是他建议如下做法: 选 k 个不同的且小于 n 的正整数 a , 以此为基对 n 执行米勒测试。如果某一次不通过, 则 n 是合数, 否则 n 就以一个大的概率是素数。米勒与拉宾的想法结合起来, 这就形成了米勒-拉宾素性测试算法, 代码如下:

```
1
2 #Input: 整数n和米勒测试的次数
3 #Output: 如果n通过k次米勒测试, 输出1, 否则输出0
4 def IsPrime(n, k):
5     q = n - 1
6     while is_even(p):
7         q /= 2
8     for i in range(k):
9         a = randint(2, n - 1)
10        if not MillerTest(a, q, n):
11            return 0
12    return 1
```

算法 5.3: 米勒-拉宾算法

定理 5.8. 伪证据的数量

如果 n 是一个奇合数，则最多有 $(n-1)/4$ 个整数 a , $1 \leq a \leq n-1$, 使得 n 可以通过以 a 为基的米勒测试。



定理5.8的证明并没有使用高级技巧，但是也颇复杂，在此不给出证明。有意思的是，利用群论的方法可以很容易得到一个类似的结论，而且非常直观、简洁。请参考第8的课后习题。

定理 5.9. 拉宾测试

如果 n 是一个奇合数，选取 k 个不同的整数 a , $1 \leq a \leq n-1$, 对 n 进行以 a 为基的米勒测试。则 n 通过所有 k 次米勒测试的概率小于 $(1/4)^k$ 。



第五章习题

1. 证明模 6 同余 5 的素数有无穷多。说明，为什么不能用同样的方法证明模 5 同余 4 的素数有无穷多。
2. 只有两种奇素数，模 4 同余 1 和模 4 同余 3。小于等于 n 且模 4 同余 1（或同余 3）的素数记为 $\pi_1(n)$ （或 $\pi_3(n)$ ）。请做以下工作：
 - (a). 编程统计在 n 分别等于 100, 1000, 10000 时 $\pi_1(n)$ 和 $\pi_3(n)$ 的值。并计算 $\pi_3(n)/\pi_1(n)$ 的值。
 - (b). 通过以上计算，得出一个关于 $\pi_1(n)$ 和 $\pi_3(n)$ 大小的猜想，哪一个集合更大？当 $n \rightarrow \infty$ 时， $\pi_3(n)/\pi_1(n)$ 会是什么？
3. 请编程找出所有使得 Carmichael 数 1105 可以通过米勒测试的所有基。
4. $f(x) = x^2 - x + 41$ 是一个有趣的函数，似乎对任意 $a \in \mathbb{N}$, $f(a)$ 都返回一个素数。请编程统计，对所有 $1 \leq a \leq 10000$, 能生成素数的个数是多少？请尝试比较分别使用试除法与米勒拉宾算法得到结果所用时间的差异。
5. 编写程序，输入整数 n , 均匀随机地生成一个 n 比特长的素数。

第六章 群与子群

内容提要

□ 群

□ 群的基本属性

□ 子群

6.1 群

前两章的内容主要针对整数的模算术运算，若干定理、引理的推导强烈依赖消去律(命题3.2)。所谓消去律即：

定理. 消去律

对整数 a, b, c 和 m ，且 $\gcd(c, m) = 1$ 。如果 $ac \equiv bc \pmod{m}$ ，则 $a \equiv b \pmod{m}$ 。

消去律意味着给定互素的整数 a 和 m ，可以计算 a 模 m 的乘法逆元 a^{-1} ，使得 $aa^{-1} \equiv 1 \pmod{m}$ 。我们利用消去律推导出费尔马小定理4.1和欧拉定理4.2。

在这一章的开始，首先需要思考的就是：在前两章的推导过程中，还使用了什么数学属性，但没有明确地定义或指出？比如，我们必须使用结合律：

定理. 结合律

对整数 a, b, c 和 m ， $(ab)c \equiv a(bc) \pmod{m}$ 。

其次，我们还需要封闭性 (closure)。也就是说，借助模运算，我们只需考虑 1 到 $m-1$ 之间的正整数。对于正整数和模算术运算，封闭性和结合律显然具备。最后，还需要强调，我们需要整数“1”，这是一个重要的且被严重依赖的隐式条件，没有“1”，逆元就无从说起了。

综上条件，并把它们推广到任意元素的集合 G 上，就形成一种新的概念—群 (group)。

定义 6.1. 群

群是集合 G 以及在其元素上所定义的操作符 \cdot ，并满足以下公理：

- 封闭性： $\forall a, b \in G, a \cdot b \in G$ ；
- 结合律： $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ ；
- 存在一个被称为单位元 (identity) 的元素 “ $e \in G$ ”，使得 $e \cdot a = a \cdot e = a$ ；
- $\forall a \in G$ ，存在逆元 (inverse) $a^{-1} \in G$ ，使得 $a \cdot a^{-1} = e = a^{-1} \cdot a$ 。

如果对任意 $a, b \in G$ ，满足 $a \cdot b = b \cdot a$ ，则群 G 被称为阿贝尔群或交换群。

以下给出几种重要的群（或非群）的实例。建议读者自行验证这些实例，明确某些代数结构为什么是或者不是群。

例 6.1 $(\mathbb{Z}, +)$ 是群, 但 (\mathbb{Z}, \times) 不是群。 (\mathbb{Q}, \times) 和 (\mathbb{R}, \times) 都是群。

例 6.2 所有的偶数在加法下成群, 记为 $2\mathbb{Z}$; 但是所有奇数加上 0 在加法上不是群, 因为奇数加奇数可能是偶数, 不满足封闭性。更一般地, 对任意整数 $n \in \mathbb{Z}$, n 的所有倍数形成的集合 $n\mathbb{Z} = \{kn : k \in \mathbb{Z}\}$ 在加法上成群。直观上, n 的倍数相加还是 n 的倍数, 0 是单位元, 对任意的 $a \in \mathbb{Z}$, na 的逆元是 $-na$ 。

例 6.3 设 n 为整数, $\mathbb{Z}_n = \{0, 1, 2, \dots, n-1\}$ 在模 n 的加法操作下形成群。然而, 在乘法上 (\mathbb{Z}_n, \times) 则不是群。

例 6.4 设 p 为素数, $\mathbb{Z}_p^* = \{1, 2, \dots, p-1\}$ 在模 p 的乘法下形成群。请适时地回忆一下费尔马小定理的证明。

例 6.5 设 n 为正整数, $\mathbb{Z}_n^* = \{a \in [1..n-1] \text{ 且 } \gcd(a, n) = 1\}$ 在模 n 的乘法下形成群。请回忆欧拉定理。

例 6.6 实数集合上 $n \times n$ 的可逆矩阵的集合在矩阵乘法下形成群, 称为一般线性群 (general linear group), 记为 $GL_n(\mathbb{R})$; 实数集合上行列式为 1 的 $n \times n$ 可逆矩阵集合在矩阵乘法下形成群, 称为特殊线性群 (special linear group), 记为 $SL_n(\mathbb{R})$ 。这两个群都不是阿贝尔群。

必须强调, 就严谨性而言, 群的定义必须包括两部分, 数据集与操作符。然而为了可读性与使用的方便, 符号误用无处不在。比如, 对某些约定俗成的群, 加法与乘法含义明确时, 我们就不再注明操作符, 比如 \mathbb{Z} 和 \mathbb{Z}_p 是加法群, \mathbb{Z}_p^* 和 \mathbb{Z}_n^* 是乘法群。再比如, 群元素的操作 $a \cdot b$ 往往在不引起误解的时候也写成 ab 。为了方便, 值得牺牲部分严谨性。

6.2 群的基本性质

命题 6.1. 单位元唯一性

群 \mathbb{G} 的单位元唯一; 即存在唯一的元素 $e \in \mathbb{G}$ 使得对所有 $g \in \mathbb{G}$, 有 $eg = ge = g$ 。♠

证明 假设 $\exists e, e' \in \mathbb{G}$ 为单位元, 则 $ee' = e'$, 且 $ee' = e$ 。联立以上两个等式, 得 $e = ee' = e'$ 。

命题 6.2. 逆元唯一性

$\forall g \in \mathbb{G}$, g 有唯一逆元 g^{-1} 。♠

证明 容易。□

命题 6.3

设 \mathbb{G} 为群, 任取 $a, b \in \mathbb{G}$, 则 $(ab)^{-1} = b^{-1}a^{-1}$ 。♠

证明 构造法。因为 $abb^{-1}a^{-1} = e$, 且 $b^{-1}a^{-1}ab = e$, 所以, 根据定义, ab 的逆元是 $b^{-1}a^{-1}$ 。□

命题 6.4

设 \mathbb{G} 是一个群, $\forall g \in \mathbb{G}$, $(g^{-1})^{-1} = g$ 。♠

证明 根据群公理, $gg^{-1} = e$, 且 $g^{-1}(g^{-1})^{-1} = e$ 。因此:

$$(g^{-1})^{-1} = (gg^{-1})(g^{-1})^{-1} = g(g^{-1}(g^{-1})^{-1}) = ge = g$$

□

命题 6.5

设 \mathbb{G} 为群, 任取两个元素 $a, b \in \mathbb{G}$, 则等式 $ax = b$ (或 $xa = b$) 在 \mathbb{G} 上有唯一解。♠

证明 对于存在性, 可找出满足要求的 x 。对唯一性, 反证法。假设存在 x_1 和 x_2 都是解, 推出矛盾。完整证明留作课后练习。

命题 6.6

设 \mathbb{G} 为群, 且 $a, b, c \in \mathbb{G}$ 。如果 $ba = ca$, 则 $b = c$; 并且, 如果 $ab = ac$, 则 $b = c$ 。♣

证明 留作课后练习。

命题6.6告诉我们, 对于群消去律必然存在。对于非交换群, 还分别有左消去律和右消去律。那么, 之前依赖消去律所得的若干属性就立即可以推广到任意的群上去。

例 6.7 在费尔马小定理和欧拉定理的证明中, 依赖消去律我们有: 对任意素数 p 和与 p 互素的正整数 a , $\mathbb{Z}_p^* = a\mathbb{Z}_p^* = \{ai : \forall i \in \mathbb{Z}_p^*\}$; 对任意合数 n 和与 n 互素的正整数 a , $a\mathbb{Z}_n^* = \mathbb{Z}_n^*$ 。请问, 对任意的群 \mathbb{G} 和群元 a , 是否有 $\mathbb{G} = a\mathbb{G} = \{ag : \forall g \in \mathbb{G}\}$? 为什么?

定义 6.2. 单射、满射与双射

给定映射 $\phi : A \mapsto B$, 如果对任意 $\phi(x_1) = \phi(x_2)$, 则有 $x_1 = x_2$, 则称映射 ϕ 为单射 (Injection), 也称为一一映射。如果对任意 $y \in B$, 都存在 $x \in A$, 使得 $\phi(x) = y$, 则称映射 ϕ 为满射 (Surjection)。如果映射 ϕ 同时满足单射性和满射性, 则称映射 ϕ 为双射 (Bijection)。♣

例 6.8 (置换群) 对任意给定的 n 阶有限群 \mathbb{G} , $\mathbb{G} = a\mathbb{G} = \{ag : \forall g \in \mathbb{G}\}$ 。当然, 结果并不出人意料。理解这种性质的一种方法是, 把任意群元素 a “乘” 上所有群元素的操作理解为一种映射 $\lambda_a : \mathbb{G} \mapsto \mathbb{G}$, 该映射被定义为 $\lambda_a : g \mapsto ag$, 其中, $\forall g \in \mathbb{G}$ 。直观上看, 映射 λ_a 就是一次置换 (Permutation) 操作, 它改变了群元素的位置, 但群并没有变化。容易证明, 映射 λ_a 是一种双射, 即同时为单射和满射, 证明依然依赖消去律。对于一个 n 阶有限群 \mathbb{G} , 有 $n!$ 种置换映射, 形成一个 $n!$ 阶集合:

$$\overline{\mathbb{G}} = \{\lambda_g : g \in \mathbb{G}\}$$

有趣的是, 在函数复合操作下, $\overline{\mathbb{G}}$ 是一个群。可验证所有群公理, 它满足封闭性, 因为:

$$(\lambda_a \circ \lambda_b)(g) = \lambda_a(bg) = abg = \lambda_{ab}(g) \in \overline{\mathbb{G}};$$

有单位元, 因为对任意 $\lambda_a \in \overline{\mathbb{G}}$:

$$(\lambda_e \circ \lambda_a)(g) = eag = ag = \lambda_a(g),$$

并且

$$(\lambda_a \circ \lambda_e)(g) = aeg = ag = \lambda_a(g);$$

最后，可验证该代数结构有逆元：

$$(\lambda_a \circ \lambda_{a^{-1}})(g) = \lambda_a(a^{-1}g) = aa^{-1}g = g = \lambda_e(g).$$

这种由置换构成的群被命名为置换群 (Permutation Group)，它在群论的具有重要的地位。可惜，这并非本书重点论述的知识点。

接下来引入一些常用的表达式。设 \mathbb{G} 是群，且 $g \in \mathbb{G}$ 。对任意 $n \in \mathbb{N}$ ，有以下表达式。

- $g^0 = e$
- $g^n = \underbrace{g \cdot g \cdots g}_{n-1 \text{ 次群运算}}$
- $g^{-n} = \underbrace{g^{-1} \cdot g^{-1} \cdots g^{-1}}_{n-1 \text{ 次群运算}}$

注 值得强调的是，任意群元 g^i 中的 i 不是群元素，是一个标量 (Scalar)。看上去这种强调没什么意思，其实不然，如果 g 刚好是落在整数上的群，往往会容易混淆。又偏巧，这种混淆是暂时无害的误解，习惯成自然，到了真要区分标量与群元素的时候往往会忘记了这里的强调。

另外，如果把群理解为加法群的时候，往往又把 g^i 写成 ig 。比如第14章的椭圆曲线群， n 个群元素 P 自相加就表示为 nP 。

命题 6.7

设 \mathbb{G} 是群， $\forall a, b \in \mathbb{G}$ ，以下性质成立。

- $\forall m, n \in \mathbb{Z}, g^m g^n = g^{m+n};$
- $\forall m, n \in \mathbb{Z}, (g^m)^n = g^{mn};$
- $\forall n \in \mathbb{Z}, (gh)^n = (h^{-1}g^{-1})^{-n};$ 如果 \mathbb{G} 是阿贝尔群，则 $(gh)^n = g^n h^n$ 。

证明 留做课后练习。

定义 6.3. 阶

有限群的阶 (order) 被定义为群元素的个数。如果一个群有无穷多元素，则它的阶称为无穷阶。如果群 \mathbb{G} 有 n 个元素，记为 $|\mathbb{G}| = n$ 。设 $g \in \mathbb{G}$ ， g 的阶被定义为最小正整数 m 使得 $g^m = e$ ，并记为 $\text{ord}(g) = m$ 。如果不存在最小正整数 m 使得 $g^m = e$ ，则称 g 有无穷阶。

例 6.9 群 \mathbb{Z} 是无穷阶群。设 p 是素数，则群 \mathbb{Z}_p 的阶是 p ，群 \mathbb{Z}_p^* 的阶是 $p-1$ 。设 n 是任意正整数，群 \mathbb{Z}_n 的阶是 n ，群 \mathbb{Z}_n^* 的阶是 $\phi(n)$ 。

例 6.10 请注意，群元素的阶与群操作相关。比如，加法群 \mathbb{Z}_p 中元素的阶称为加法阶，而乘法群 \mathbb{Z}_p^* 元素的阶则称为乘法阶。在 Sage 中分别给出了不同的操作。在上下文明确的且无歧义时，常常省略群操作而只称为阶。

```
1 sage: p = 17; a = 4
2 sage: Mod(a, p).multiplicative_order() # a模p的乘法阶，即a在乘法群中的阶
3 4
```



```

4 sage: Mod(a, p).additive_order()           # a模p的加法阶, 即a在加法
      群中的阶
5 17

```

例 6.11 设 $p = 11$, 请编程验证 \mathbb{Z}_p 中除单位元之外的每一个元素阶都为 11。尝试多个不同的素数, 发现规律: 如果 p 是素数, 则群 \mathbb{Z}_p 中除单位元之外的每一个元素阶都为 p 。可以验证, 2 在 \mathbb{Z}_{11}^* 中的阶是 10, 但是并非单位元除外的所有元素的阶都是 10, 比如 4 的阶是 5。我们会在第 7 章继续讨论这些问题。

6.3 子群

要分析一个大的数据集, 我们往往会先把数据划分为若干小的集合进行分析。同样地, 要考察一个群的属性, 我们也会先把群分成更小的群, 先考察小群的属性。这种想法引出了子群 (Subgroup) 的定义。

定义 6.4. 子群

设 G 为群。若 H 是 G 的子集, 且 H 在群 G 的操作符上也满足群公理, 则 H 被称为 G 的子群, 记为 $H \leq G$ 。



例 6.12 若干子群的实例。

- 在加法上, 偶数群 $2\mathbb{Z}$ 是整数群 \mathbb{Z} 的子群。对任意的 $n \in \mathbb{Z}$, $n\mathbb{Z}$ 是 \mathbb{Z} 的子群。
- 对任意群 G , 存在一个平凡子群 $\{e\}$ 。这是简单且平凡的结论, 但值得留意。
- 在加法上, $\mathbb{Z} \leq \mathbb{Q} \leq \mathbb{R} \leq \mathbb{C}$ 。

例 6.13 (\mathbb{Z}_p^* 的子群) 设 p 为素数, 对所有的 $i \in \mathbb{Z}_p^*$, 计算 $i^2 \bmod p$, 形成集合 $S = \{i^2 \bmod p, \forall i \in \mathbb{Z}_p^*\}$ 。检验 S 在模 p 的乘法下形成群, 即 $S \leq \mathbb{Z}_p^*$ 。 S 的阶是多少?

例 6.14 (\mathbb{Z}_n^* 的子群) 设 n 是一个正合数, 随机选取两个整数 $a, b \in \mathbb{Z}_n^*$, 形成集合 $S = \{a, b\}$ 。然后, 将集合 S 中两个元素相乘得到 c , 如果 $c \notin S$ 则更新集合 S 为 $S \cup \{c\}$ 。不断重复这种操作, 直到 S 不再增长。检验 S 在模 n 乘法操作下形成群, 即 $S \leq \mathbb{Z}_n^*$ 。请计算 S 的阶。

例 6.15 (对称群 S_n 的子群) 不难知道, 对任意 n 阶集合 S 中的元素进行置换, 一共有 $n!$ 种可能的映射, 把所有可能的置换放入集合 S_n , 在函数复合操作下, S_n 是一个群, 被称为对称群 (Symmetric Group), 且例子 6.8 中定义的置换群是对称群的子群。请读者自己验证以上结论。

命题 6.8

群 G 的非空子集 H 是 G 的子群, 当且仅当 $H \neq \emptyset$, 且对任意 $a, b \in H$, $ab^{-1} \in H$ 。



证明 充分性容易。对于必要性, 只需要检验 H 满足所有的群公理。

第六章习题

1. 给出命题 6.5 的完整证明。
2. 证明命题 6.6。

3. 证明命题6.7。
4. 设 \mathbb{G} 是群, 对任意 $n \in \mathbb{N}, i \in [0, n], g_i \in \mathbb{G}$ 。证明 $g_0 g_1 \cdots g_n$ 的逆元是 $g_n^{-1} \cdots g_1^{-1} g_0^{-1}$ 。
5. 设 n 是一个正整数。对所有的 $x \in \mathbb{Z}_n$, 定义 $|x|$ 为求 x 的绝对值, 其中 x 表达为 $\{-(n-1)/2, \dots, (n-1)/2\}$ 范围内的带符号整数。从群 \mathbb{Z}_n^* 出发, 定义集合 \mathbb{G}^+ 为

$$\mathbb{G}^+ = \{|x| : x \in \mathbb{Z}_n^*\}$$

对 $g, h \in \mathbb{G}^+$, 定义以下操作

$$g \circ h = |g \cdot h \bmod n|$$

请验证 (\mathbb{G}^+, \circ) 是一个群。如果 n 是一个素数, 该群的阶是多少? 如果 $n = pq$, p 和 q 是两个不同的素数, 该群的阶是多少?

6. 给出命题6.8的完整证明。
7. 证明: 任意群 \mathbb{G} 的两个子群的交集也是群 \mathbb{G} 的子群。
8. 证明或证伪: 任意群 \mathbb{G} 的两个子群的并集也是群 \mathbb{G} 的子群。
9. 证明: 阿贝尔群 \mathbb{G} 中的有限阶元素形成子群。该子群有一个特殊的名字, 称为挠子群 (*Torsion Subgroup*)。
10. 编程找规律: 给定一个正合数 n , 找出加法群 \mathbb{Z}_n 中阶为 n 的元素, 这些元素之间满足什么关系? 提示: 1 的阶为 n 。
11. 编程完成以下工作: 给定一个素数 p , 返回乘法群 \mathbb{Z}_p^* 的一个子群。据此, 能否找出子群的阶与 \mathbb{Z}_p^* 的阶之间的关系?
12. 编程完成以下工作: 给定一个合数 n , 返回乘法群 \mathbb{Z}_n^* 的一个子群。据此, 能否找出子群的阶与 \mathbb{Z}_n^* 的阶之间的关系?

第七章 循环群

内容提要

□ 循环群及循环子群

□ 循环群属性

□ 原根定理

□ 寻找原根

7.1 循环群

这一节考察这样的一种群，其所有的元素都能被特定的元素计算出来（或者生成出来）。考虑以下计算。给定素数 p ，从乘法群 Z_p^* 中任意选取元素 g ，构造集合 S ：

$$S = \{g, g^2, \dots, g^i, \dots\},$$

i 是任意整数。即不断对 g 做乘法，直到集合 S 不再增长。

然后，问以下问题：

- S 是否一个有限集合？
- S 在模 p 乘法下是否成群？理由？
- S 是否会等于 Z_p^* ？

在必要时，为什么不动手写个程序来帮忙呢？

例 7.1 设 $p = 11$ ，选取 $g = 4$ ，计算：

$$S = \{g, g^2, \dots, g^{10}, \dots\}$$

得到：

$$S = \{4, 5, 9, 3, 1\}$$

当然， S 是有限集，因为 g 是有限群的元素。 S 是一个群，然而它并不等于 Z_{11}^* 。继续问：

- 如果选择 $g = 2$ ，得到什么？
- 如果选择 $g = 3$ 呢？或者其他数值呢？

循环群的定义可以表述为以下命题，之所以是命题，因为需要验证群公理。

命题 7.1

设 G 是群， $\forall g \in G$ 。则集合

$$\langle g \rangle = \{g^k : k \in \mathbb{Z}\}$$

是 G 的子群。称 $\langle g \rangle$ 为由 g 生成的循环群 (Cyclic Group)，称 g 为群的生成元 (generator)。而且， $\langle g \rangle$ 是群 G 中包含元素 g 的最小子群。

证明 要证明 $\langle g \rangle$ 是群只需要检验所有群公理。由于群的封闭性，任何包括元素 g 的子群都必然包括所有 g^i ， $i \in \mathbb{Z}$ ，所以 $\langle g \rangle$ 是群 G 中包含元素 g 的最小子群。□

通过例7.1和命题7.1, 对循环群的第一印象就是, 给定一个元素 g , 整个群可以由 g 生成。其实, 这里还有一点微妙之处。请看下例。

例 7.2 \mathbb{Z} 是循环群, 生成元是 1。 $2\mathbb{Z} = \{\dots, -4, -2, 0, 2, 4, \dots\}$ 也是循环群, 2 为生成元。微妙之处在于, 如何理解群 \mathbb{Z} 中的负数由 1 生成呢? 同样, $2\mathbb{Z}$ 中的负数如何由 2 生成呢? 关键点在于, 根据命题7.1, 如果 g 落在群中, 必然有 g^{-1} 也落在群中。

例 7.3 \mathbb{Z}_n 是循环群, 生成元是 1。体会一下, 这个群元素的生成方式似乎与 \mathbb{Z} 不同。

例 7.4 完成例7.1的计算, 可知 \mathbb{Z}_{11}^* 是循环群, 2 和 6 都是群的生成元。也许生成元还有不少, 能找出其他生成元吗?

例 7.5 对于有限循环群而言, 群的生成元就是元素的阶等于群阶的群元素。以下 Sage 代码显示群 \mathbb{Z}_{17}^* 所有元素的阶。

```
1 sage: p = 17
2 sage: for i in range(1, p):
3     . . . . : print("The order of ", i, "is", Mod(i,p).
      multiplicative_order())
```

改进以上代码, 对任意素数 p , 统计群 \mathbb{Z}_p^* 生成元的个数, 能找出规律吗?

例 7.6 \mathbb{Z}_{11}^* 是循环群, 似乎是因为 11 是素数。如果给定一个合数 n , \mathbb{Z}_n^* 也会是循环群吗? 最好写个程序找找规律。

当然, 除了依赖整数的加法和乘法形成的循环群, 还有其他形式的循环群, 只是, 它们并不是本书关注的重点。请参看以下例子。

例 7.7 $\langle i \rangle$ 在乘法下是循环群, i 是复数, 满足 $i * i = -1$ 。 $\langle i \rangle = \{1, -1, i, -i\}$ 由 i 生成。

命题 7.2

所有的循环群都是阿贝尔群。



证明 设 $G = \langle g \rangle$ 是循环群, 且 $a, b \in G$ 。必然存在 $r, s \in \mathbb{Z}$ 使得, $a = g^r$ 和 $b = g^s$ 。因此,

$$ab = g^r g^s = g^{r+s} = g^{s+r} = g^s g^r = ba。$$

所以 G 是阿贝尔群。

命题 7.3

循环群 $G = \langle g \rangle$ 的每一个子群都是循环群。



证明 设 H 是 G 的子群。如果 $H = \{e\}$, 则这是循环群。假设 H 包含除 e 以外的元素, 则 H 中的元素必然是 $H = \{e\} \cup \{g^i, \forall i \in S\}$, 其中 S 是 \mathbb{Z} 的某个子集。设 m 为 S 中的最小正整数, 根据良序公理, 此 m 必然存在。注意, 对任意 $i \in \mathbb{Z}$, 如果 $g^i \in H$, 必然有 $g^{-i} \in H$, 所以集合 S 中必然有正整数。以下, 只需证 $h = g^m$ 为 H 的生成元。

为此, 证明任取 $h' \in H$, h' 由 h 生成。不失一般性, 假设 $h' = g^k$, k 是某个正整数。由除法算法, 存在整数 q 和 r 使得 $k = mq + r$, 其中 $0 \leq r < m$ 。因此:

$$h' = g^k = g^{mq+r} = (g^m)^q g^r = h^q g^r。$$

因为 $g^k, h^{-q} \in \mathbb{H}$, 所以, $g^r = g^k h^{-q}$ 是 \mathbb{H} 中元素。根据 \mathbb{H} 的定义, 只能是 $r = 0$, 即 $k = mq$ 。因此,

$$h' = g^k = g^{mq} = h^q。$$

□

7.2 原根

这一节暂时离开代数, 转向数论, 讨论一种与生成元相关的数论概念: 原根 (primitive root)。

定义 7.1. 原根

设 a 和 n 是互素的正整数。 a 模 n 的阶定义为最小的整数 $e \geq 1$ 使得 $a^e \equiv 1 \pmod{n}$ 。如果 a 模 n 的阶等于模 n 下最大可能的阶, 则称 a 是模 n 的原根。



“模 n 下最大可能的阶”是什么意思? 不要忘记费尔马小定理和欧拉定理, 对任意给定的互素的正整数 a 和 n , 必然有:

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

即 a 的阶只能小于等于 $\phi(n)$, 而 a 的阶最大也只能是 $\phi(n)$, 所以 $\phi(n)$ 就是最大可能的阶。

例 7.8 根据例 7.1 我们知道, 4 模 11 的阶是 5, 而 2 和 3 模 n 的阶都是 10。因为模 11 最大可能的阶是 $\phi(11) = 10$ 因此, 2 和 3 是模 11 的原根。使用群的语言, 则表述为 \mathbb{Z}_{11}^* 是由生成元 2 (或者 3) 生成的循环群。

对于模整数的原根, 有以下重要且通用的结论。

定理 7.1. 原根定理

对每一个素数 p 都有模 p 的原根, 且恰有 $\phi(p-1)$ 个模 p 原根。



例 7.9 群 \mathbb{Z}_{11}^* 有 $\phi(10) = 4$ 个生成元。群 \mathbb{Z}_{13}^* 有 $\phi(12) = 4$ 个生成元。如何找出它们所有的生成元呢?

定理 7.2. 广义原根定理

如果 $n \in \mathbb{Z}$ 形如 2、4、 p^e 和 $2p^e$, 对所有素数 $p > 2$ 和所有正整数 e , 则 \mathbb{Z}_n^* 是循环群。



用数论的语言表述广义原根定理, 则意味着对满足定理条件的整数 n , 必有模 n 的原根。换言之, 不满足条件的整数 n 则不存在原根。以上定理出现在标准的数论教材中, 本书不再给出证明。

7.3 循环群的相关属性

这一节接着讨论循环群的若干属性，特别是有限循环群的性质。所谓有限循环群，即循环群 $\mathbb{G} = \langle g \rangle$ 的阶为 n ， n 是一个正整数。先观察循环群元素之间的运算。任取群 $\mathbb{G} = \langle g \rangle$ 中的两个元素 $g_1 = g^{k_1}$ 和 $g_2 = g^{k_2}$ ，必然有：

$$g_1 g_2 = g^{k_1} g^{k_2} = g^{k_1+k_2} = g^{(k_1+k_2) \bmod n},$$

再考虑到，任取群 \mathbb{G} 中一个元素 $g_i = g^{k_i}$ ， k_i 会落在 0 到 $n-1$ 之间，即群元素的指数会遍历完从 0 到 $n-1$ 之间的所有整数。这意味，任取 k_i 在模 n 下有加法逆元，且逆元落在 0 到 $n-1$ 之间。这些观察直接导致一种直观认识：把群元素的操作理解为群指数上的加法，而且是落在群 \mathbb{Z}_n 的加法运算。这种认识没有立即带来新的定理，但是也许会带来理解上的方便。用这种认识去理解记忆以下命题是非常方便的，虽然我们还是需要严谨的证明。

命题 7.4

任取正整数 n ，设 $\mathbb{G} = \langle g \rangle$ 是阶为 n 的循环群，则 $g^k = e$ 当且仅当 n 整除 k 。



证明

1. 必要性证明：如果 n 整除 k ，则存在 $s \in \mathbb{Z}$ 使得 $k = ns$ ，所以有 $g^k = g^{ns} = e$ 。
2. 充分性证明：如果 $g^k = e$ ，根据除法算法， $\exists q, r \in \mathbb{Z}$ 使得 $k = nq + r$ ，其中 $0 \leq r < n$ 。因此，

$$e = g^k = g^{nq+r} = g^{nq} g^r = g^r.$$

又因为 n 是使得 $g^n = e$ 的最小正整数，所以 $r = 0$ 。

□

命题 7.5

设群 $\mathbb{G} = \langle g \rangle$ 是阶为 n 的循环群。如果 $h = g^k$ ，则 h 的阶为 n/d ，其中 $d = \gcd(k, n)$ 。



证明 设 $m \in \mathbb{N}$ 使得 $h^m = g^{km} = e$ 。

1. 根据命题 7.4 可知， $n \mid km$ ，即 $(n/d) \mid (k/d)m$ 。
2. 因为 $d = \gcd(k, n)$ ， n/d 与 k/d 互素。因此， $(n/d) \mid (k/d)m$ 蕴涵 $(n/d) \mid m$ 。可被 n/d 整除的最小的整数是 m 。

例 7.10 已知 2 是群 \mathbb{Z}_{11}^* 的生成元，群 \mathbb{Z}_{11}^* 的阶是 10， $2^3 = 8 \in \mathbb{Z}_{11}^*$ ，且 $\gcd(3, 10) = 1$ ，所以 8 的阶是 10，即 8 也是一个生成元。5 不是生成元，因为 $5 = 2^4 \bmod 11$ ， $\gcd(4, 10) = 2$ 。请读者自行验证以上结论。命题 7.5 告诉我们，在知道某个元是生成元时，如何找到另一个生成元。

推论 7.1


设群 $\mathbb{G} = \langle g \rangle$ 是阶为 n 的循环群，则群 \mathbb{G} 中恰有 $\phi(n)$ 个生成元。



证明 群 \mathbb{G} 中有 n 个元素，都可表达为 g^i ， $i \in \mathbb{Z}_n$ 。根据命题 7.5，对任意的 g^i ，其阶为 n/d ，其中 $d = \gcd(i, n)$ 。当 $d = 1$ 时 g^i 为生成元，此时 i 与 n 互素。 \mathbb{Z}_n 中有 $\phi(n)$ 个元

素与 n 互素，因此 \mathbb{G} 中有 $\phi(n)$ 个生成元。 \square

推论 7.2

设群 $\mathbb{G} = \langle g \rangle$ 是阶为 p 的循环群， p 是素数，则 \mathbb{G} 中的元素除了 e 之外都是生成元。 

证明 根据推论 7.1 可知。 \square

例 7.11 根据推论 7.2，如果 p 是素数，则群 \mathbb{Z}_p 中除单位元之外的每一个元素阶都是生成元，因为 \mathbb{Z}_p 是阶为 p 的循环群。群 \mathbb{Z}_p^* 中一共有 $\phi(p-1)$ 个生成元，因为群 \mathbb{Z}_p^* 的阶是 $p-1$ 。这些规律我们在例 6.11 已经有所提示，不知道读者当时发现了这些规律没有？


7.4 寻找生成元

原根定理告诉我们，给定一个素数 p ， \mathbb{Z}_p^* 是循环群，即必有至少一个生成元。通过命题 7.5 和例 7.10，我们知道，给定一个生成元可以很容易发现另一个生成元。然而我们更希望从无到有地直接求一个生成元。因为只考虑 \mathbb{Z}_p^* 的生成元，以下我们不加区别地把生成元表述为原根。

在此之前，先考虑一个相对“容易”的问题：给定一个素数 p 和一个正整数 a ，能否高效地判定 a 是否模 p 的原根？

首先，要明确所谓生成元，就是该元素的阶等于群阶的元素。而非生成元的元素的阶必然比群的阶要小，而且还必然整除群的阶，如以下推论。

推论 7.3

设群 $\mathbb{G} = \langle g \rangle$ 是阶为 n 的循环群，则群中任意元素 h 的阶必整除 n 。 

证明 该结论的正确性可直接由命题 7.5 得出。以后还会有一个更简洁的证明。 \square

这样，算法的思路就很明确了。给定 a 和 p ，只需要判断 a 是否有整除 $p-1$ 的阶。直观上，算法穷举 $p-1$ 的所有因子 f （注意，不是素因子，且 $f < p-1$ ），判断 $a^f \bmod p$ 是否等于 1，如果等则 a 不是原根，返回 False。如果对 $p-1$ 的所有因子 f ，都不返回 False，则返回 True，即 a 是原根。进一步改进这个直观思路，得到更好的效率。算法描述如下。

```

1 # 输入素数p和一个小于p的正整数a
2 # 输出True，如果a是模p的原根，否则输出False
3 def is_primitive_root(a, p):
4     flist = prime_fators_list(p-1) # 出求p-1的所有素因子
5     for f in flist:
6         if pow(a, (p-1)/f, p) == 1:
7             return False
8     return True

```

算法 7.1: 原根判定算法

算法的正确性是显然的, 其算法思想与之前描述的直观思路完全一样。不同之处在于, 它并不穷举 $p-1$ 的所有因子进行计算和判断, 而是穷举 $p-1$ 所有素因子 p_i , 计算 $a^{(p-1)/p_i} \bmod p$ 并进行判断, 但是达到了相同的目的。算法的复杂性则主要看求 $p-1$ 的素因子这一步。以后我们会知道, 这是一个困难问题。

关于算法的正确性与效率性, 不再继续证明。建议读者动手计算来体会其中的含义, 可以分几步走:

1. 设 $n = pq$, p 和 q 是不同的两个素数;
2. 设 $n = p^i q^j$, p 和 q 是不同的两个素数, 且 i 和 j 是大于 1 的整数;
3. 设 $n = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$, p_i 是不同的素数, 且 $e_i \geq 1$ 。

有了原根判定算法, 那么生成一个随机的原根则很容易。只需要随机在 \mathbb{Z}_p^* 中选取一个元素, 判定它是否原根, 如果是则返回, 否则重新选取一个随机元素, 直到找到一个原根为止。根据推论 7.1, 原根的数目还颇多, 所以, 一次随机选取元素找到原根的概率颇大。

需要强调一点, 当 p 很大时, 求 $p-1$ 的素因子并不容易。所以, 在使用 \mathbb{Z}_p^* 群的时候, 我们并不是真的随机选取素数 p , 而是通过算法构造出 p , 同时掌握 $p-1$ 的素因子。

第七章 习题

1. 请心算列举出群 \mathbb{Z}_{10} 的所有生成元。
2. 群 \mathbb{Z}_{17}^* 有多少个生成元? 已知 3 是其中一个生成元, 请问 9 和 10 是否生成元?
3. p 和 q 是素数, 请问 \mathbb{Z}_{pq} 有多少个生成元? r 是任意正整数, 请问 \mathbb{Z}_{p^r} 有多少个生成元?
4. 证明: 如果 p 是素数, 则 \mathbb{Z}_p 没有非平凡子群。
5. 证明: 设 n 为任意正整数, 群 \mathbb{Z}_n 的生成元 r 满足以下条件: $1 \leq r \leq n$, 且 $\gcd(r, n) = 1$ 。
6. 证明: 如果群 G 没有非平凡子群, 则群 G 是循环群。
7. 证明推论 7.3, 即群 G 中任意元素的阶都整除群 G 的阶。
8. 编程完成以下工作: 给定一个素数 p , 找出 \mathbb{Z}_p^* 的最小生成元。对于素数 $1 < p < 10000$, 哪一个素数 p 使得 \mathbb{Z}_p^* 的最小生成元最大?
9. 设 q 是素数且 $p = 2 * q + 1$ 也是素数。选取 $g = h^2$ 且 $g \neq 1$, 其中 h 是 \mathbb{Z}_p^* 中的元素。显然, $\langle g \rangle$ 是循环群。
 - (a). 群 $\langle g \rangle$ 的阶是多少, 为什么?
 - (b). 群 $\langle g \rangle$ 中有多少个生成元, 为什么?
 - (c). 写一个 Python (或者 Sage) 程序生成群 $\langle g \rangle$ 。

第八章 陪集与拉格朗日定理

内容提要

□ 陪集

□ 拉格朗日定理

8.1 陪集

在第4章，费尔马小定理的证明中使用这样一种操作，选取一个群元素 $a \in \mathbb{Z}_p^*$ ，对群 \mathbb{Z}_p^* 中的每一个元素都乘上 a ，得到一个“新”的集合。这种操作被称为元素 a 作用于群 \mathbb{Z}_p^* 。同样的操作也出现在欧拉定理的证明中。在这一节，将定义一种群元素作用于子群的操作，并利用这种定义和相关属性来证明一个重要的定理。

定义 8.1. 陪集

设 G 是群， H 是群 G 的子群。定义 H 的左陪集为

$$gH = \{gh : h \in H\},$$

其中，元素 $g \in G$ 被称为代表元 (Representative)。右陪集的定义类似：

$$Hg = \{hg : h \in H\}.$$



例 8.1 回忆费尔马小定理的证明，随机选取元素 $a \in \mathbb{Z}_p^*$ ，然后证明：

$$a\mathbb{Z}_p^* = \mathbb{Z}_p^*.$$

在欧拉定理的证明中，类似地，证明： $\forall a \in \mathbb{Z}_n^*$,

$$a\mathbb{Z}_n^* = \mathbb{Z}_n^*.$$

例 8.2 设 $p = 11$, $g = 4$ ，则 $H = \{g^i : i \in \mathbb{Z}\}$ 是 \mathbb{Z}_p^* 的子群。实际上， $H = \{1, 3, 4, 5, 9\}$ 。做以下计算：

- $\forall a \in H$, aH 等于什么？
- $\forall a \notin H$, 但 $a \in \mathbb{Z}_p^*$, aH 等于什么？

希望大家通过以上例子熟悉陪集的操作。 H 是群，但是一般情况下，陪集 gH 并不是群，除非 $g \in H$ 。此时 g 会被 H 吸收掉，即 $gH = H$ 。这种简单的性质在下一章很有用。

以下的证明统一使用左陪集，实际上左陪集与右陪集并没有本质区别，使用右陪集也会得到相同的属性、命题和定理。需要强调的是，一般情况下，任取群元 $g \in G$ ，左陪集 gH 与右陪集 Hg 并不相等。以下考察陪集的若干属性。首先，陪集的代表元可以有多个，或者说，陪集中任意元素都可以作为代表元。

命题 8.1. 陪集属性

设 G 是群, H 是 G 的子群。任取 $g_1, g_2 \in G$, 则 $g_1H = g_2H$ 当且仅当 $g_2 \in g_1H$ 。

证明 如果 $g_1H = g_2H$, 则存在 $h_1, h_2 \in H$ 使得 $g_1h_1 = g_2h_2$, 即得 $g_2 = g_1h_1h_2^{-1}$, 也即 $g_2 \in g_1H$ 。必要性证明留为课后练习。 \square

命题 8.2. 陪集的阶

设 G 是群, H 是 G 的子群。 $\forall g \in G$, 群 H 的阶与集合 gH 的阶相同。

证明 定义映射 $\psi: H \rightarrow gH$ 为, $\psi(h) = gh, g \in G, h \in H$ 。然后证明该映射是单射和满射。请回忆在费尔马小定理的证明中, 我们做过什么。要证明单射, 只需要证明任取 $h, h' \in H$, 且 $h \neq h'$, 则 $gh \neq gh'$ 。映射 ψ 显然是一种满射。

命题 8.3. 同一或不相交

设 G 为群, H 是 G 的子群。任取不同的两个群元素 $g_1, g_2 \in G$, 则 $g_1H = g_2H$ 或 $g_1H \cap g_2H = \emptyset$ 。

证明 假设存在 $h_1, h_2 \in H$ 使得 $g_1h_1 = g_2h_2$, 可证明 $g_1H \subseteq g_2H$ 。注意, 任取 $g_1h \in g_1H$, 有

$$g_1h = g_1(h_1h_1^{-1})h = g_2(h_2h_1^{-1}h) \in g_2H,$$

即 $g_1h \in g_2H$ 。类似, 可证 $g_2H \subseteq g_1H$ 。因此, $g_1H = g_2H$ 。

命题 8.4. 群 G 的划分

设 G 是群, H 是群 G 的子群。则子群 H 的所有左陪集将划分群 G 。

证明 首先, 要确信 H 的左陪集覆盖了群 G , 即对任意 $g \in G$, 存在 $g' \in G$ 且存在 $h \in H$, 使得 $g = g'h$ 。这显然正确。然后, 使用命题 8.3 即得 \square

定义 8.2. 指标

设 G 是群, H 是群 G 的子群。 H 在群 G 上的指标 (Index) 定义为 H 在 G 上左陪集的个数, 并记为 $[G:H]$ 。

例 8.3 $H = \{1, 3, 4, 5, 9\}$ 是 \mathbb{Z}_{11}^* 的子群。 H 在 G 有两个左陪集, 即 $[G:H] = 2$ 。

既然提到集合 (群) 的“划分”, 就必须重提第 3 章所提及的等价关系和等价类等定义。我们知道, 如果有一种等价关系定义在集合 S 上, 则该等价关系会把集合 S 划分为不相交的等价类。毫不意外, 陪集相等关系是一种等价关系, 表述为以下命题。

命题 8.5

设 G 是群, H 是群 G 的子群。则子群 H 的所有左陪集的相等关系是这样一种等价关系 \sim , 即对任意 $g_1, g_2 \in G$, $g_1 \sim g_2$ 当且仅当 $g_1H = g_2H$ 。

证明 容易, 直接验证自反、对称、传递三个属性。 \square

8.2 拉格朗日定理

命题 8.6. 拉格朗日定理

设 G 是有限群, H 是 G 的子群, 则 $|G|/|H| = [G:H]$, H 在 G 上不同左陪集的个数。



证明 群 G 被划分为 $[G:H]$ 个不同的左陪集, 每一个左陪集有 $|H|$ 个元素。因此 $|G| = [G:H]|H|$ 。□

等价地, 拉格朗日定理还可表述为: 子群 H 的阶必然整除群 G 的阶。这是一个重要结论, 以下推论和定理显示, 拉格朗日定理是分析有限群的有力工具。

推论 8.1

设 G 是有限群, 对任意 $g \in G$, g 的阶必然整除群 G 的阶, 即 $\text{ord}(g) \mid |G|$ 。



推论 8.2

设 G 是素数阶有限群, 即 $|G| = p$, p 是素数, 则 G 是循环群且任意非单位元元素 $g \in G$ 是生成元。



证明 任取 $g \in G$ 且 $g \neq e$ 。根据推论 8.1, g 的阶必然整除 $|G| = p$, p 是素数。因为 $\text{ord}(g) > 1$, 则 $\text{ord}(g) = p$ 。因此 g 是群 G 的一个生成元。□

推论 8.3

设 H 和 K 是有限群 G 的子群, 并且 $K \subset H \subset G$, 则

$$[G:K] = [G:H][H:K]。$$



证明 观察可得:

$$[G:K] = \frac{|G|}{|K|} = \frac{|G|}{|H|} \frac{|H|}{|K|} = [G:H][H:K]$$



使用群的语言和拉格朗日定理, 可对第 4 章的两个重要定理, 费尔马小定理和欧拉定理, 给出简洁的证明。

推论 8.4. 费尔马小定理

设 p 是素数, a 是整数且 $p \nmid a$ 。则

$$a^{p-1} \equiv 1 \pmod{p}。$$



证明 留作课后练习。□

推论 8.5. 欧拉定理

设 n 是正合数, a 是正整数且 $\gcd(a, n) = 1$ 。则

$$a^{\phi(n)} \equiv 1 \pmod{n}。$$



证明 作为课后习题。 □

使用群论的语言，还可以把费尔马小定理和欧拉定理变形为以下“群论”版本。

定理 8.1. 群论版费尔马小定理

\mathbb{G} 是阶为 n 的有限群，则对任意群元 $a \in \mathbb{G}$ ，有 $a^n = e$ 。 ♡

必须强调，虽然该定理被称为“群论版费尔马小定理”，但它也适用于欧拉定理，因为欧拉定理中的 $\phi(n)$ 是群 \mathbb{Z}_n^* 的阶。

证明 直接由推论 8.1 可得。 □

注意，看上去还可以使用之前的技巧，通过证明任取群元 $a \in \mathbb{G}$ ，有 $a\mathbb{G} = \mathbb{G}$ ，从而得到结论。但是，如果这样的话，就必须依赖交换律，定理就只能在阿贝尔群下成立了。

第八章 习题

1. 设 \mathbb{G} 是群， \mathbb{H} 是 \mathbb{G} 的子群。任取 $g_1, g_2 \in \mathbb{G}$ ，则 $g_1\mathbb{H} = g_2\mathbb{H}$ 当且仅当 $g_1^{-1}g_2 \in \mathbb{H}$ 。
2. 如果 \mathbb{G} 是群， \mathbb{H} 是群 \mathbb{G} 的子群，且 $[\mathbb{G} : \mathbb{H}] = 2$ ，请证明 $g\mathbb{H} = \mathbb{H}g$ 。
3. 如果群 \mathbb{H} 是群 \mathbb{G} 的真子群，即存在 $g \in \mathbb{G}$ 但是 $g \notin \mathbb{H}$ 。请证明 $|\mathbb{H}| \leq |\mathbb{G}|/2$ 。
4. 设 $n \in \mathbb{N}$ 为合数，群 \mathbb{Z}_n^* 中至少一半的元素是 n 为合数的证据。提示： \mathbb{Z}_n^* 中不是 n 为合数证据的元素是否构成群？
5. 使用群论的方法重新证明费尔马小定理和欧拉定理。

第九章 同构，同态与商群

内容提要

□ 同构

□ 同态

□ 商群

□ 第一同构定理

9.1 同构

虽然许多不同的群表现为不同的形式，但本质上也许它们是同一种代数结构。如何刻画所谓本质上相同的群结构，这引出了同构 (*Isomorphism*) 这个概念。给定两个不同的群，如果能在它们所代表的两类群元素中找到一种双射 (单射且为满射)，并且在该映射下群元素的群操作得以保持，则称这两个群同构。

定义 9.1. 同构

给定群 (\mathbb{G}, \cdot) 和群 (\mathbb{H}, \circ) ，如果存在一种双射 $\phi: \mathbb{G} \mapsto \mathbb{H}$ ，使得群操作得以保持，即：任取 $a, b \in \mathbb{G}$ ，

$$\phi(a \cdot b) = \phi(a) \circ \phi(b)$$

称群 \mathbb{G} 与群 \mathbb{H} 同构，并记为 $\mathbb{G} \cong \mathbb{H}$ 。映射 ϕ 被称为一种同构映射。



注 同构映射和之后的同态映射是人为造出来的字眼。本来，英文的 *Isomorphism* (同构) 和 *Homomorphism* (同态) 就是指满足特定条件的映射 ϕ ，而表述两个群同构用的是形容词 *Isomorphic* (同构的)，表示两个群同态则用的是形容词 *Homomorphic* (同态的)。因为中文不大容易区分这里的名词与形容词，所以在表示名词的同构 (或同态) 时强调它们是映射。

例 9.1 $\mathbb{Z}_4 \cong \langle i \rangle$ ，因为可以定义以下双射 $\phi: \mathbb{Z}_4 \mapsto \langle i \rangle$ 为 $\phi(n) = i^n$ 。映射 ϕ 是单射且为满射，因为：

$$\phi(0) = 1$$

$$\phi(1) = i$$

$$\phi(2) = -1$$

$$\phi(3) = -i.$$

并且， ϕ 保持了群操作，因为：

$$\phi(m+n) = i^{m+n} = i^m i^n = \phi(m)\phi(n).$$

例 9.2 给定群 $\mathbb{Z}_8^* = \{1, 3, 5, 7\}$ 和群 $\mathbb{Z}_{12}^* = \{1, 5, 7, 11\}$ ，可证明 $\mathbb{Z}_8^* \cong \mathbb{Z}_{12}^*$ ，因为可以定义

双射 $\phi: \mathbb{Z}_8^* \mapsto \mathbb{Z}_{12}^*$ 如下:

$$\begin{aligned} 1 &\mapsto 1 \\ 3 &\mapsto 5 \\ 5 &\mapsto 7 \\ 7 &\mapsto 11 \end{aligned}$$

还可以找到这两个群的另一个同构映射吗?

命题 9.1

设 $\phi: \mathbb{G} \mapsto \mathbb{H}$ 从群 \mathbb{G} 到群 \mathbb{H} 的一种同构映射, 则以下命题为真:

1. $\phi^{-1}: \mathbb{H} \mapsto \mathbb{G}$ 也是同构;
2. $|\mathbb{G}| = |\mathbb{H}|$;
3. 如果 \mathbb{G} 是阿贝尔群, 则 \mathbb{H} 也是阿贝尔群;
4. 如果 \mathbb{G} 是循环群, 则 \mathbb{H} 也是循环群;
5. 如果 \mathbb{G} 有阶为 n 的子群, 则 \mathbb{H} 也有阶为 n 的子群。



证明 留作课后练习。

命题 9.2

所有无限阶的循环群都同构于群 \mathbb{Z} 。



证明 设群 \mathbb{G} 是一个无限阶的循环群, $g \in \mathbb{G}$ 是生成元。定义 $\phi: \mathbb{Z} \mapsto \mathbb{G}$ 为 $\phi: n \mapsto g^n$ 。则

$$\phi(m+n) = g^{m+n} = g^m g^n = \phi(m)\phi(n)。$$

然后, 证明 ϕ 是双射, 留作课后练习。



命题 9.3

如果 \mathbb{G} 是阶为 n 的循环群, 则 \mathbb{G} 同构于 \mathbb{Z}_n 。



证明 设 \mathbb{G} 是阶为 n 的循环群, g 是生成元。定义 $\phi: \mathbb{Z}_n \mapsto \mathbb{G}$ 为 $\phi: k \mapsto g^k$, 其中, $0 \leq k < n$ 。证明 ϕ 是一种同构映射, 留作课后练习。



推论 9.1

如果 \mathbb{G} 是阶为 p 的循环群, 其中 p 是素数, 则 \mathbb{G} 同构于 \mathbb{Z}_p 。



证明 由命题 9.3 和推论 8.2 直接可得。



命题 9.4

群同构关系决定了一种等价关系, 将所有的群划分为不同的等价类。



证明 留作课后练习。

命题 9.4 的证明是简单的, 然而其内涵颇深刻, 它展示了研究群同构的基本意义。例如, 阶为素数的循环群表现形式有多种, 在实际应用中包括 \mathbb{Z}_p^* 的素数阶子群, 或者素数

阶椭圆曲线群等，利用推论9.1的结果，无论素数阶群表现有多大差异，我们都可以通过研究加法群 \mathbb{Z}_p ，得到所有素数阶群的属性。

以下是一个有趣的、值得大家知道的结论。

定理 9.1. 凯莱定理

每一个群都同构于一个置换群。



证明 给定任意一个 n 阶有限群 \mathbb{G} ，例子6.8展示如何构造一个 n 阶置换群 $\overline{\mathbb{G}}$ 。定义同态映射 $\phi: \mathbb{G} \mapsto \overline{\mathbb{G}}$ 为 $\phi: g \mapsto \lambda_g$ 。首先验证这是一种双射，即为单射且为满射。 ϕ 显然是满射。要证单射，只需要证如果给定两个元素 $a, b \in \mathbb{G}$ ，如果 $\phi(a) = \phi(b)$ ，则 $a = b$ 。 $\phi(a) = \phi(b)$ 意味着，对任意群元素 $g \in \mathbb{G}$ ，有 $\phi(a)(g) = \phi(b)(g)$ ，即

$$ag = \lambda_a(g) = \lambda_b(g) = bg,$$

因此，根据消去律， $a = b$ 。最后，可验证群操作得以保持，因为，

$$\phi(ab) = \lambda_{ab} = \lambda_a \circ \lambda_b = \phi(a) \circ \phi(b).$$

凯莱定理暗示，要研究群需要重点研究对称群和置换群。实际上，在许多现有的计算代数系统中都实现了对称群和置换群，比如 Sage 和 GAP 等。

9.2 同态

放松同构的定义，不要求映射为双射，就有了同态 (*homomorphism*) 的定义。

定义 9.2. 同态

给定两个群 (\mathbb{G}, \cdot) 和 (\mathbb{H}, \circ) ，如果存在映射 $\phi: \mathbb{G} \mapsto \mathbb{H}$ 使得对任意的 $a, b \in \mathbb{G}$ ，群操作得以保持，即：

$$\phi(a \cdot b) = \phi(a) \circ \phi(b)$$

则称群 (\mathbb{G}, \cdot) 同态于群 (\mathbb{H}, \circ) 。映射 ϕ 被称为同态映射。



例 9.3 \mathbb{Z} 是加法群，定义映射 $\phi: \mathbb{Z} \mapsto \mathbb{Z}$ 为 $\phi(i) = 2i, \forall i \in \mathbb{Z}$ 。可以验证 ϕ 是一种群同态，它把整数映射为偶数。偶数在加法下成群。

例 9.4 设 \mathbb{G} 是群， $g \in \mathbb{G}$ 。定义映射 $\phi: \mathbb{Z} \mapsto \mathbb{G}$ 为 $\phi(n) = g^n, \forall n \in \mathbb{Z}$ 。则 ϕ 是一种群同态映射，因为

$$\phi(m+n) = g^{m+n} = g^m g^n = \phi(m)\phi(n).$$

ϕ 不是一种从 \mathbb{Z} 到 \mathbb{G} 的满射，也不是单射。它把 \mathbb{Z} 满射到由 g 生成的 \mathbb{G} 的循环子群。

例 9.5 设 p 为素数，任取 $g \in \mathbb{Z}_p^*$ 。请定义映射 $\phi: \mathbb{Z} \mapsto \mathbb{Z}_p^*$ 使得 ϕ 是一种群同态映射，并找出 ϕ 将 \mathbb{Z} 满射到由 g 生成的 \mathbb{Z}_p^* 的循环子群。可参考例7.1。

在深入讨论同态之前，先给出一种重要的定义。

定义 9.3. 正规子群

群 \mathbb{H} 是群 \mathbb{G} 的子群。如果对任意 $g \in \mathbb{G}$, 有 $g\mathbb{H} = \mathbb{H}g$, 则称 \mathbb{H} 是群 \mathbb{G} 的正规子群 (Normal subgroup)。



正规子群就是一种使得左陪集与右陪集相等的子群。等式 $g\mathbb{H} = \mathbb{H}g$ 表达某种意义上的“交换律”，然而对于阿贝尔群，该性质显然成立。请注意，等式 $g\mathbb{H} = \mathbb{H}g$ 并意味说，对任意的 $g \in \mathbb{G}$, 存在 $h \in \mathbb{H}$ 使得 $gh = hg$ 。而是说对任意的 $g \in \mathbb{G}$ 和 $h \in \mathbb{H}$, 存在 $h' \in \mathbb{H}$ 使得 $gh = h'g$ 。

命题 9.5

设 \mathbb{G} 是群, \mathbb{H} 是 \mathbb{G} 的子群。以下命题等价。

1. 子群 \mathbb{H} 是 \mathbb{G} 的正规子群, 即对任意 $g \in \mathbb{G}$, $g\mathbb{H} = \mathbb{H}g$ 。
2. 对任意 $g \in \mathbb{G}$, 有 $g\mathbb{H}g^{-1} = \mathbb{H}$ 。



证明 (1) \implies (2). 因为 \mathbb{H} 是 \mathbb{G} 的正规子群, 所以对任意 $g \in \mathbb{G}$, $g\mathbb{H} = \mathbb{H}g$; 即对任意给定的 $g \in \mathbb{G}$ 和 $h \in \mathbb{H}$, 存在 $h' \in \mathbb{H}$ 使得 $gh = h'g$ 。根据消去律, $ghg^{-1} = h' \in \mathbb{H}$, 即得 $g\mathbb{H}g^{-1} \subset \mathbb{H}$ 。对任意 $h \in \mathbb{H}$, 根据以上结论, 对任意群元 $g^{-1} \in \mathbb{G}$, 有 $g^{-1}hg = g^{-1}h(g^{-1})^{-1} \in \mathbb{H}$ 。因此, 存在 $h' \in \mathbb{H}$, 使得 $g^{-1}hg = h'$ 。所以, $h = gh'g^{-1} \in g\mathbb{H}g^{-1}$, 即 $\mathbb{H} \subset g\mathbb{H}g^{-1}$ 。

(2) \implies (1). 假设对任意 $g \in \mathbb{G}$, $g\mathbb{H}g^{-1} = \mathbb{H}$ 。那么, 对任意 $h \in \mathbb{H}$, 存在 $h' \in \mathbb{H}$ 使得 $ghg^{-1} = h'$ 。即 $gh = h'g$, 这意味着 $g\mathbb{H} \subset \mathbb{H}g$ 。同理, 可证 $\mathbb{H}g \subset g\mathbb{H}$ 。

命题 9.6

设 $\phi: \mathbb{G}_1 \mapsto \mathbb{G}_2$ 为一种从群 \mathbb{G}_1 到群 \mathbb{G}_2 的群同态映射, 则以下命题为真。

1. 如果 e 是 \mathbb{G}_1 的单位元, 则 $\phi(e)$ 是 \mathbb{G}_2 的单位元;
2. 对任意群元 $g \in \mathbb{G}_1$, $\phi(g^{-1}) = [\phi(g)]^{-1}$;
3. 如果 \mathbb{H}_1 是 \mathbb{G}_1 的子群, 则 $\phi(\mathbb{H}_1)$ 是 \mathbb{G}_2 的子群;
4. 如果 \mathbb{H}_2 是群 \mathbb{G}_2 的子群, 则 $\phi^{-1}(\mathbb{H}_2) = \{g \in \mathbb{G}_1 : \phi(g) \in \mathbb{H}_2\}$ 是群 \mathbb{G}_1 的子群。而且, 如果 \mathbb{H}_2 是群 \mathbb{G}_2 的正规子群, 则 $\phi^{-1}(\mathbb{H}_2)$ 是 \mathbb{G}_1 的正规子群。

**证明**

1. 假设 e 和 e' 分别是群 \mathbb{G}_1 和 \mathbb{G}_2 的单位元, 则:

$$e'\phi(e) = \phi(e) = \phi(ee) = \phi(e)\phi(e)$$

然后根据消去律, 得到 $\phi(e) = e'$ 。

2. 因为 $\phi(g^{-1})\phi(g) = \phi(g^{-1}g) = \phi(e) = e'$, 所以, $\phi(g^{-1}) = [\phi(g)]^{-1}$ 。
3. 首先, $\phi(\mathbb{H}_1)$ 是非空集合, 因为它至少包括群 \mathbb{G}_2 的单位元。任取 $a', b' \in \phi(\mathbb{H}_1)$, 存在 $a, b \in \mathbb{H}_1$, 使得 $\phi(a) = a'$ 且 $\phi(b) = b'$ 。又因为 \mathbb{H}_1 是 \mathbb{G}_1 的子群, 所以有:

$$a'b'^{-1} = \phi(a)[\phi(b)]^{-1} = \phi(ab^{-1}) \in \phi(\mathbb{H}_1)$$

最后, 根据命题 6.8, $\phi(\mathbb{H}_1)$ 是群 \mathbb{G}_2 的子群。

4. 设 \mathbb{H}_2 是群 \mathbb{G}_2 的子群, 记 $\mathbb{H}_1 = \phi^{-1}(\mathbb{H}_2)$ 。首先, \mathbb{H}_1 是非空集合, 因为 $\phi(e) = e'$,

所以, \mathbb{H}_1 至少包括单位元。任取 $a, b \in \mathbb{H}_1$, 因为 \mathbb{H}_2 是群, 则有:

$$\phi(ab^{-1}) = \phi(a)[\phi(b)]^{-1} \in \mathbb{H}_2.$$

因此, $ab^{-1} \in \mathbb{H}_1$, \mathbb{H}_1 是 \mathbb{G}_1 的子群。

设 \mathbb{H}_2 是群 \mathbb{G}_2 的正规子群, 要证 \mathbb{H}_1 是 \mathbb{G}_1 的正规子群, 即需要证对任意的 $g \in \mathbb{G}_1$, $h \in \mathbb{H}_1$, 有 $g^{-1}hg \in \mathbb{H}_1$ 。因为 \mathbb{H}_2 是群 \mathbb{G}_2 的正规子群, 所以有:

$$\phi(g^{-1}hg) = \phi(g^{-1})\phi(h)\phi(g) \in \mathbb{H}_2$$

因此, $g^{-1}hg \in \mathbb{H}_1$ 。

定义 9.4. Kernel

设 $\phi: \mathbb{G} \mapsto \mathbb{H}$ 是一种群同态映射, e 是群 \mathbb{H} 的单位元。根据命题9.6, $\phi^{-1}(\{e\})$ 是 \mathbb{G} 的子群, 该子群称为 ϕ 的 Kernel, 记为 $\text{Ker } \phi$ 。



命题 9.7. Kernel

设 $\phi: \mathbb{G} \mapsto \mathbb{H}$ 是一种群同态映射, 则 ϕ 的 Kernel 是 \mathbb{G} 的正规子群。



证明 容易, 因为 \mathbb{H} 的平凡子群是正规子群。

9.3 商群

如果群 \mathbb{H} 是群 \mathbb{G} 的正规子群, 则 \mathbb{H} 在 \mathbb{G} 中的陪集形成一个群 \mathbb{G}/\mathbb{H} , 群操作是 $(a\mathbb{H})(b\mathbb{H}) = ab\mathbb{H}$ 。此群被称为 \mathbb{G} 在 \mathbb{H} 上的商群 (quotient group) 或因子群 (factor group)。通常把 \mathbb{G}/\mathbb{H} 读成 \mathbb{G} 模 \mathbb{H} , 没错, 这里的“模”就是我们之前讲整数除法的那个“mod”。该结论可表述为以下定理。

定理 9.2. 商群

若群 \mathbb{H} 是群 \mathbb{G} 的正规子群, 则 \mathbb{H} 在群 \mathbb{G} 的陪集形成商群 \mathbb{G}/\mathbb{H} , 且群的阶为 $[\mathbb{G} : \mathbb{H}]$ 。



现在要证明 \mathbb{G}/\mathbb{H} 确实是一个群, 即要给出定理9.2的完整证明。证明分为三步: 首先, 要理解商群的群操作, 即要理解 $(a\mathbb{H})(b\mathbb{H}) = ab\mathbb{H}$ 的含义。其次, 要证明群操作是一种良定义 (well-defined) 操作。最后, 验证群公理, 这一步最简单。以下分别对这三步骤进行解释。

理解和证明商群定理的第一步, 必须理解商群的群操作, 即如何理解 $(a\mathbb{H})(b\mathbb{H}) = ab\mathbb{H}$? $(a\mathbb{H})(b\mathbb{H})$ 是集合的乘法, 就是集合中所有元素分别进行群操作所得到的集合。比如对任意集合 A 和集合 B 进行相乘, 得到 $AB = \{ab : \forall a \in A \text{ 且 } \forall b \in B\}$ 。把对群操作的解释写成以下引理。

引理 9.1

群 \mathbb{H} 是群 \mathbb{G} 的正规子群, 群 \mathbb{H} 的任意两个陪集 $a\mathbb{H}$ 和 $b\mathbb{H}$ 的乘积依然是一个陪集, 即 $(a\mathbb{H})(b\mathbb{H}) = ab\mathbb{H}$ 。



证明 因为 \mathbb{H} 是正规子群, 所以 $a\mathbb{H} = \mathbb{H}a$ 。又因为 \mathbb{H} 是群, 所以 $\mathbb{H}\mathbb{H} = \mathbb{H}$ 。因此,

$$(a\mathbb{H})(b\mathbb{H}) = (\mathbb{H}a)(b\mathbb{H}) = (ab\mathbb{H})\mathbb{H} = ab\mathbb{H}.$$

□

其次, 要证明 \mathbb{G}/\mathbb{H} 中的操作是一种良定义操作, 这对于代数的初学者是陌生的。所谓良定义的操作, 就是要求操作独立于所参与操作的代表元。比如, 对任意群 \mathbb{G} 和其上的某种操作 $\psi: \mathbb{G} \mapsto \mathbb{G}$, 要求 ψ 良定义就是要求对任意的群元 $a, b \in \mathbb{G}$, 如果 $a = b$, 则 $\psi(a) = \psi(b)$ 。一眼看上去, 这个要求很无理, 毫无意义, 但是对于商群来说就必不可少。请注意, 商群中操作的是陪集, $a\mathbb{H} = b\mathbb{H}$ 并不意味 $a = b$ 。而且命题8.1强调, 每一个陪集中的元素都可以是代表元。那么群操作时使用不同代表元表示的陪集 $a\mathbb{H}$ 或者 $b\mathbb{H}$ 会不会有不同结果呢? 确保群操作的良定义就是要确保群操作独立于陪集的代表元。这种思想表达为以下引理。

引理 9.2. 商群操作的良定义性

\mathbb{G}/\mathbb{H} 中的群操作是一种良定义操作, 即如果对任意的群元 $a, b, c, d \in \mathbb{G}$, 有 $a\mathbb{H} = b\mathbb{H}$ 且 $c\mathbb{H} = d\mathbb{H}$, 则有:

$$(a\mathbb{H})(c\mathbb{H}) = ac\mathbb{H} = bd\mathbb{H} = (b\mathbb{H})(d\mathbb{H})$$



证明 根据条件 $a\mathbb{H} = b\mathbb{H}$ 且 $c\mathbb{H} = d\mathbb{H}$, 则存在 $n_1, n_2 \in \mathbb{H}$, 使得 $a = bn_1$, $c = dn_2$ 。因此,

$$\begin{aligned} ac\mathbb{H} &= bn_1dn_2\mathbb{H} \\ &= bn_1d\mathbb{H} \\ &= bn_1\mathbb{H}d \\ &= b\mathbb{H}d \\ &= bd\mathbb{H} \end{aligned}$$

又根据引理9.1, 有 $(a\mathbb{H})(c\mathbb{H}) = ac\mathbb{H}$ 和 $(b\mathbb{H})(d\mathbb{H}) = bd\mathbb{H}$, 得证。

□

最后, 完成一些简单工作就可以给出定理9.2的完整证明。

证明 验证 \mathbb{G}/\mathbb{H} 的群公理, 引理9.1证明了群操作的封闭性。结合律是显然的。群的单位元是 $e\mathbb{H} = \mathbb{H}$, 群元 $a\mathbb{H}$ 的逆元是 $a^{-1}\mathbb{H}$ 。群的阶就是陪集的个数, 即 $[\mathbb{G} : \mathbb{H}]$ 。 □

以下通过一些简单的例子来熟悉商群。

例 9.6 有两种平凡的商群。对任意群 \mathbb{G} , e 是它的单位元。商群 \mathbb{G}/\mathbb{G} 同构于平凡子群 $\{e\}$ 。商群 $\mathbb{G}/\{e\}$ 同构于群 \mathbb{G} 。请读者自行验证。

例 9.7 (\mathbb{Z} 的商群) 群 \mathbb{Z} 是阿贝尔群, 所有子群都是正规子群。考虑 \mathbb{Z} 的子群 $3\mathbb{Z}$, 它在 \mathbb{Z} 中的所有陪集是:

$$0 + 3\mathbb{Z} = \{\dots, -3, 0, 3, 6, \dots\}$$

$$1 + 3\mathbb{Z} = \{\dots, -2, 1, 4, 7, \dots\}$$

$$2 + 3\mathbb{Z} = \{\dots, -1, 2, 5, 8, \dots\}.$$

商群 $\mathbb{Z}/3\mathbb{Z}$ 的群操作可表述为以下乘法表。



+	$0 + 3\mathbb{Z}$	$1 + 3\mathbb{Z}$	$2 + 3\mathbb{Z}$
$0 + 3\mathbb{Z}$	$0 + 3\mathbb{Z}$	$1 + 3\mathbb{Z}$	$2 + 3\mathbb{Z}$
$1 + 3\mathbb{Z}$	$1 + 3\mathbb{Z}$	$2 + 3\mathbb{Z}$	$0 + 3\mathbb{Z}$
$2 + 3\mathbb{Z}$	$2 + 3\mathbb{Z}$	$0 + 3\mathbb{Z}$	$1 + 3\mathbb{Z}$

以上操作可以推广到对任意给定的整数 n 和子群 $n\mathbb{Z}$, 可得商群 $\mathbb{Z}/n\mathbb{Z}$, 请读者自行完成构造。

例 9.8 (\mathbb{Z}_n^* 的商群) 设 $n = 15$, 则 $\mathbb{Z}_n^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$. 设 $g = 2$, 记 $S = \langle g \rangle = \{1, 2, 4, 8\}$, 这 \mathbb{Z}_n^* 的子群。则 $\mathbb{Z}_n^*/7S = \{S, 7S\}$, 请验证 S 是群的单位元, $7S$ 的逆元是自己本身, 即 $(7S)(7S) = 4S = S$ 。这个例子中有一个特别需要大家注意的要点, 同一个群元素可能有不同的表现形式, 比如, $7S$ 和 $4S$ 。这也说明了确保群操作良定义的必要性。

9.4 同构定理

上述知识点可以形成这样一条知识链: 从一个群同态 $\phi: \mathbb{G}_1 \mapsto \mathbb{G}_2$ 出发。首先可以找到 ϕ 的 Kernel, 于是得到一个 \mathbb{G}_1 的正规子群; 接着, 构造 \mathbb{G}_1 上的商群 $\mathbb{G}_1/\text{Ker}\phi$ 。这一节接着讨论所得商群的性质。有教材说, 这里是通过商群来研究同态。有趣的是, 本节所有的例子似乎都是在用同态来研究商群, 或者得到商群相关的性质。而这种性质竟然主要是关于同构, 本章第一节的内容。于是, 同构定理把同构、同态、Kernel、正规子群和商群等概念紧密地联系起来, 成为这条知识链的最重要一环。

定义 9.5. 标准同态

设 \mathbb{H} 是群 \mathbb{G} 的正规子群, 定义同态映射 $\phi: \mathbb{G} \mapsto \mathbb{G}/\mathbb{H}$ 为 $\phi(g) = g\mathbb{H}$, $\forall g \in \mathbb{G}$ 。并称该映射为标准同态或者自然同态 (Cannonical Homomorphism), 且 $\text{Ker } \phi = \mathbb{H}$ 。

可以验证, ϕ 确实是一种同态, 因为:

$$\phi(g_1 g_2) = g_1 g_2 \mathbb{H} = g_1 \mathbb{H} g_2 \mathbb{H} = \phi(g_1) \circ \phi(g_2)$$

定理 9.3. 第一同构定理

如果 $\psi: \mathbb{G} \mapsto \mathbb{H}$ 是一种群同态映射, 且 $\mathbb{K} = \text{Ker } \psi$, 则 \mathbb{K} 是 \mathbb{G} 的正规子群。设 $\phi: \mathbb{G} \mapsto \mathbb{G}/\mathbb{K}$ 是标准同态, 则存在唯一同构映射 $\eta: \mathbb{G}/\mathbb{K} \mapsto \psi(\mathbb{G})$ 使得 $\psi = \eta\phi$ 。

让我们先从表面上理解这个定理。首先, 通过群同态映射 $\psi: \mathbb{G} \mapsto \mathbb{H}$ 得到一个 \mathbb{G} 的正规子群 \mathbb{K} , 这是最容易理解的部分。接着, 用群 \mathbb{G} 模 \mathbb{K} 得到一个商群 \mathbb{G}/\mathbb{K} , 我们的任务是要求出该商群的阶。要达到这个目标, 定理最后告诉我们, 可以找到商群 \mathbb{G}/\mathbb{K} 与 \mathbb{H} 上的子群 $\psi(\mathbb{G})$ 之间的一个同构映射。注意一点, 映射 ψ 并不要求是满射, 根据命题 9.6, ψ 打到 \mathbb{H} 上的像形成子群 $\psi(\mathbb{G})$ 。以上思想可以表达为如图 9.1 的直观图形。

证明分为三步。首先, 定理既然是要找一个从 \mathbb{G}/\mathbb{K} 到 $\psi(\mathbb{G})$ 同态映射, 那就使用构造法, 先构造一个同态映射 η 。第二步, 证明映射 η 是良定义映射。第三步, 证明映射 η 是同态且是双射, 即为同构映射。以下给出具体证明。

证明

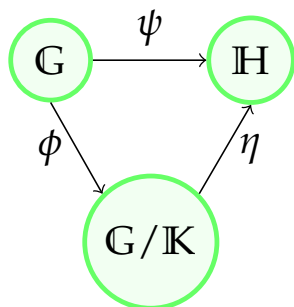


图 9.1: 第一同构定理的图形化表达

1. 定义映射 $\eta: \mathbb{G}/\mathbb{K} \mapsto \psi(\mathbb{G})$ 为 $\eta(g\mathbb{K}) = \psi(g)$ 。
2. 证明 η 是良定义映射，即证明如果对于两个不同的 $g_1, g_2 \in \mathbb{G}$ ，有 $g_1\mathbb{K} = g_2\mathbb{K}$ ，则 $\eta(g_1\mathbb{K}) = \eta(g_2\mathbb{K})$ 。因为 $g_1\mathbb{K} = g_2\mathbb{K}$ ，则存在 $k \in \mathbb{K}$ 使得 $g_1 = g_2k$ ，这仅仅利用了陪集相等的定义。因此：

$$\eta(g_1\mathbb{K}) = \psi(g_1) = \psi(g_2k) = \psi(g_2)\psi(k) = \psi(g_2) = \eta(g_2\mathbb{K})$$

请注意，上式中第四个等号之所以成立是因为 $k \in \mathbb{K}$ ，而 \mathbb{K} 是 ψ 的 **Kernel**， ψ 必然映射到 \mathbb{H} 的单位元。这就证明了 η 映射独立于陪集代表元的选择。之所以 η 是唯一定义的，因为给定了 ψ 和 ϕ ，而 $\psi = \eta\phi$ 。

3. 可证明 η 是同态且是双射。 η 是同态，因为：

$$\begin{aligned} \eta(g_1\mathbb{K}g_2\mathbb{K}) &= \eta(g_1g_2\mathbb{K}) \\ &= \psi(g_1g_2) \\ &= \psi(g_1)\psi(g_2) \\ &= \eta(g_1\mathbb{K})\eta(g_2\mathbb{K}) \end{aligned}$$

η 显然是满射，最后，只需证明 η 是单射。假设 $\eta(g_1\mathbb{K}) = \eta(g_2\mathbb{K})$ ，则 $\psi(g_1) = \psi(g_2)$ 。 g_1 和 g_2 是 \mathbb{G} 中两个不同的元素，所以

$$\psi(g_1) = \psi(g_2g_2^{-1}g_1) = \psi(g_2)\psi(g_2^{-1}g_1)$$

即 $g_2^{-1}g_1 = e$ 。所以 $g_2^{-1}g_1\mathbb{K} = \mathbb{K}$ ，最后得到 $g_1\mathbb{K} = g_2\mathbb{K}$ 。

□

例 9.9 (循环群) 设 \mathbb{G} 是由生成元 g 生成的循环群。定义映射 $\phi: \mathbb{Z} \mapsto \mathbb{G}$ 为 $n \mapsto g^n$ ， $\forall n \in \mathbb{Z}$ 。 ϕ 是同态映射，因为：

$$\phi(m+n) = g^{m+n} = g^m g^n = \phi(m)\phi(n)。$$

ϕ 显然是满射。如果 \mathbb{G} 的阶为 m ，因为 g 是生成元，则 $\text{ord}(g) = m$ 。于是， $g^m = e$ ，且有 $\text{Ker } \phi = m\mathbb{Z}$ 。根据第一同构定理，则有：

$$\mathbb{Z}/\text{Ker } \phi = \mathbb{Z}/m\mathbb{Z} \cong \mathbb{G}。$$

如果 \mathbb{G} 是无限阶，则 g 也是无限阶，则 $\text{Ker } \phi = \{0\}$ ，则 \mathbb{Z} 与 \mathbb{G} 同构。因此，两个循环

群同构当且仅当它们有相同的阶。在同构的意义上, 只有两种循环群: \mathbb{Z} 和 \mathbb{Z}_n 。

例 9.10 (从 \mathbb{Z}_p^* 到 \mathbb{Z}_p^* 的同态.) 设 p 是素数, \mathbb{Z}_p^* 是循环群。定义映射 $\phi: \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$ 为 $\phi(g) = g^2, \forall g \in \mathbb{Z}_p^*$ 。 ϕ 是一种群同态映射, 因为:

$$\phi(g_1 g_2) = (g_1 g_2)^2 = g_1^2 g_2^2 = \phi(g_1) \phi(g_2)$$

显然, ϕ 不是满射, 且 $\text{Ker } \phi = \{1, p-1\}$ 是 \mathbb{Z}_p^* 的一个正规子群。知道 $\text{Ker } \phi$ 是因为根据命题 3.3, 以下公式

$$x^2 \equiv 1 \pmod{p}$$

有且仅有两个解, 即 1 和 $p-1$ 。验证可知 $S = \{\phi(g) : \forall g \in \mathbb{Z}_p^*\}$ 是群。群 S 的阶是多少? 根据第一同构定理, $|S| = |\mathbb{Z}_p^* / \text{Ker } \phi| = |\mathbb{Z}_p^*|/2$ 。

例 9.11 (从 \mathbb{Z}_n^* 到 \mathbb{Z}_n^* 的同态.) 设 $n = pq$ 是合数, p 和 q 是素数, 则 \mathbb{Z}_n^* 是群。定义映射 $\phi: \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^*$ 为 $\phi(g) = g^2, \forall g \in \mathbb{Z}_n^*$ 。可验证 ϕ 是一种群同态映射。记 $S = \{\phi(g) : \forall g \in \mathbb{Z}_n^*\}$ 。如果知道 $\text{Ker } \phi$ 的阶, 则根据第一同构定理, 可知 $|S| = |\mathbb{Z}_n^*|/|\text{Ker } \phi|$, 要完成这个目标, 需要知道以下公式有多少个解:

$$x^2 \equiv 1 \pmod{n}$$

可惜, 命题 3.3 并不能完成这个任务, 需留待第 10 章再解决这个问题。

例 9.12 设 n 是一个正整数。对所有的 $x \in \mathbb{Z}_n$, 定义 $|x|$ 为求 x 的绝对值, 其中 x 表达为 $\{-(n-1)/2, \dots, (n-1)/2\}$ 范围内的带符号整数。从群 \mathbb{Z}_n^* 出发, 定义集合 \mathbb{G}^+ 为

$$\mathbb{G}^+ = \{|x| : x \in \mathbb{Z}_n^*\}$$

对 $g, h \in \mathbb{G}^+$, 定义以下操作

$$g \circ h = |g \cdot h \bmod n|$$

可验证 (\mathbb{G}^+, \circ) 是一个群。这是第 6 章的课后习题。现在, 利用第一同构定理, 可以解决求该群的阶是多少的问题。

观察可知, 取绝对值的操作 (不妨记为 ϕ) 是一种从 \mathbb{Z}_n^* 到 \mathbb{G}^+ 的同态映射, 因为:

$$\phi(x \cdot y) = |x \cdot y| = |x| \circ |y| = \phi(x) \circ \phi(y)$$

又因为 $-1 \in \mathbb{Z}_n^*$, 所以 $\text{Ker } \phi = \{1, -1\}$ 。最后, 根据第一同构定理, \mathbb{G}^+ 的阶是 $|\mathbb{Z}_n^*|/2$ 。

定义 9.6

给定两个群 \mathbb{G} 和 \mathbb{H} , 记 $\mathbb{G} \times \mathbb{H}$ 为集合 \mathbb{G} 与集合 \mathbb{H} 的笛卡尔乘积, 并定义集合 $\mathbb{G} \times \mathbb{H}$ 中元素的乘法操作 \cdot 为:

$$(g_1, h_1) \cdot (g_2, h_2) = (g_1 g_2, h_1 h_2)。$$

可验证, $\mathbb{G} \times \mathbb{H}$ 是群, 并称为 \mathbb{G} 与 \mathbb{H} 的直积 (direct product)。



定义 9.7

给定一个阿贝尔群 \mathbb{G} 和它两个子群 \mathbb{H}_1 和 \mathbb{H}_2 , 且满足 $\mathbb{H}_1 \cap \mathbb{H}_2 = \{e\}$ 。定义 $\mathbb{H}_1 \oplus \mathbb{H}_2$

为:

$$\mathbb{H}_1 \oplus \mathbb{H}_2 = \{h_1 h_2 : h_1 \in \mathbb{H}_1, h_2 \in \mathbb{H}_2\}$$

可立即验证 $\mathbb{H}_1 \oplus \mathbb{H}_2$ 是一个群, 并称之为 \mathbb{H}_1 与 \mathbb{H}_2 的直和 (*direct sum*)。



命题 9.8

给定一个阿贝尔群 \mathbb{G} , 设 \mathbb{H}_1 和 \mathbb{H}_2 是群 \mathbb{G} 的任意两个子群, 且 $\mathbb{H}_1 \cap \mathbb{H}_2 = \{e\}$ 。

定义以下映射:

$$\phi : \mathbb{H}_1 \times \mathbb{H}_2 \mapsto \mathbb{H}_1 \oplus \mathbb{H}_2$$

为

$$(h_1, h_2) \mapsto h_1 h_2。$$

则映射 ϕ 是一种同构映射。



证明 需证明映射 ϕ 是同态映射, 且是双射。映射 ϕ 是同态映射, 因为:

$$\phi((g_1, g_2) \cdot (h_1, h_2)) = \phi(g_1 h_1, g_2 h_2) = g_1 h_1 g_2 h_2 = \phi(g_1, g_2) \phi(h_1, h_2)。$$

读者可自行体会, 为什么需要 \mathbb{G} 是阿贝尔群。由定义, 映射 ϕ 显然是满射, 只需证它是单射。为此, 需证 $\text{Ker } \phi = \{(e, e)\}$ 。对任意 $(g_1, g_2) \in \mathbb{H}_1 \times \mathbb{H}_2$, 有 $\phi(g_1, g_2) = e$ 当且仅当 $g_1 g_2 = e$, 即 $g_1 = g_2^{-1}$ 。又因为 $\mathbb{H}_1 \cap \mathbb{H}_2 = \{e\}$, 因此只能是 $g_1 = g_2 = e$ 。□



思考 为什么要通过第一同构定理来证明映射 ϕ 的单射性, 而不直接利用单射的属性?

Chapter Note

To be done...

第九章习题

1. 证明命题9.1。
2. 给出命题9.2的完整证明。
3. 如果 \mathbb{H}_1 和 \mathbb{H}_2 是群 \mathbb{G} 的正规子群, 证明 $\mathbb{H}_1 \oplus \mathbb{H}_2$ 也是群 \mathbb{G} 的正规子群。
4. 对任意给定素数 p , 利用第一同构定理构造与 Z_p^* 同构的群。

第十章 中国剩余定理

内容提要

中国剩余定理

中国剩余定理的代数版本

10.1 数论视角下的中国剩余定理

中国剩余定理当然是数论中的一个重要的知识点，为什么要强调数论的视角下呢？本节的小标题确实有点怪。这里只是预示我们将用另外的视角去讨论中国剩余定理，数论的视角则是常规的视角。

讨论中国剩余定理，我们先从一元同余方程组的高效解法说起。请看以下例子。

例 10.1 假定需要解以下方程求 x ：

$$x \equiv 2 \pmod{5}$$

$$x \equiv 3 \pmod{7}$$

有一种平凡的解决方法如下：

1. 从一个同余式中得到 $x = 5t + 2$, $t \in \mathbb{N}$;
2. 把 x 代入第二个同余式得到: $5t + 2 \equiv 3 \pmod{7}$;
3. 整理可得: $5t \equiv 1 \pmod{7}$;
4. 上式两边同乘 5^{-1} 得到 $t = 7s + 3$, $s \in \mathbb{N}$;
5. 最后, $x = 35s + 17$, 即 $x \equiv 17 \pmod{35}$ 。

对任意的一元同余方程组, 中国剩余定理 (*Chinese Remainder Theorem*, 简记为 CRT), 告诉我们, 对特定的模数总是有唯一解, 并给出了一种高效的求解方法。先看以下定理的描述。

定理 10.1. 中国剩余定理

设 p 和 q 是互素的两个正整数, $n = pq$ 。对任意的 $a \in \mathbb{Z}_p$ 和 $b \in \mathbb{Z}_q$, 存在唯一解 x , $0 \leq x < n$ 使得:

$$x \equiv a \pmod{p}$$

$$x \equiv b \pmod{q}$$

然而, 在给出整个定理证明之前, 让我们试图重新“发明”这个定理的证明。希望在这一节当中, 读者可以享受到发明定理的乐趣, 这比读懂一个证明要快乐得多。所以, 在我提示大家要停下来思考的时候, 请尽量跟我的节奏走。

我们的任务是解以上两个同余方程, 让我们先考察其中一个同余式。比如, 设 p 为

一个整数, 对任意的 $a \in \mathbb{Z}_p$, 所求解的 x 要满足方程

$$x \equiv a \pmod{p}$$

那么, x 应该“长”什么模样? 答案之一, 当然是存在 $k \in \mathbb{Z}$, 使得 $x = a + kp$ 。这是例子10.1的方法。还有没有其他答案?

大家能回顾一下线性代数中相似矩阵的定义吗? 对两个 $n \times n$ 矩阵 A 和 B , 如果存在一个可逆的 $n \times n$ 矩阵 C 使得:

$$A = C^{-1}BC$$

则称 A 和 B 为相似矩阵。回到同余式 $x \equiv a \pmod{p}$, 我们能发现 x 与 a 是“相似数”吗? 即存在某个“可逆数” c 使得 $c^{-1}ac = a$? 建议读者在此处暂停阅读, 自己思考一段时间。

相似矩阵的内涵很丰富, 本章基本用不上, 只是一种横行联想。而与中国剩余定理更相关的是正规子群的知识点。回顾第9章关于正规子群的定义(定义9.3), 群 \mathbb{H} 是群 \mathbb{G} 的正规子群意味着, 如果对任意 $g \in \mathbb{G}$, 有 $g\mathbb{H} = \mathbb{H}g$ 。等价地, 即对任意的 $g \in \mathbb{G}$ 和 $a \in \mathbb{H}$, 存在 $a' \in \mathbb{H}$ 使得 $g^{-1}ag = a'$ 。此时也称 a 与 a' 共轭 (Conjugate)。如果群 \mathbb{G} 是阿贝尔群, 因为交换律, 很显然有 $g^{-1}ag = g^{-1}ga = a$ 。提示到这里, 读者能想到什么吗?

回到刚才的问题, 其实我们是要问给定 a 和 p , 能构造什么样的数可与 a 模 p 同余? 或者说, 试图找一个值 a' , 使得

$$aa' \equiv a \pmod{p}$$

第一印象应该是选一个模 p 等于 1 的数 a' 。对于整数 a , 在可逆矩阵、正规子群的启发下, 能联想到模 p 下互为乘法逆元的整数 c 和 c^{-1} 吗? 也就是说, 对任意的 $c \in \mathbb{N}$, 如果 $\gcd(c, p) = 1$, 则存在唯一乘法逆元 $c^{-1} \in \mathbb{N}$, 使得 $cc^{-1} \equiv 1 \pmod{p}$ 。如果 $x = cc^{-1}a$, 借用线性代数的概念, 称整数 x 和 a 相似。为什么相似, 因为 x 和 a 模 p 同余且不等。注意, 给定 a , 可以构造多个不同的 x 与之相似。另外, 如果从抽象代数的角度出发, 可称整数 x 和 a 共轭。无论相似或者共轭, 都只是一种联想法, 以下并不会使用到与之相关的理论与内涵, 它们只是在提醒我们应该想到利用乘法逆元。

对于满足两个同余式 $x \equiv a \pmod{p}$ 和 $x \equiv b \pmod{q}$ 的解 x , 大致形式应该是既与 a 相似又与 b 相似。分别相似并不难构造, 难点是同时满足“相似性”。比如考虑两个相似数的线性组合, 令 $x = c_1c_1^{-1}a + c_2c_2^{-1}b$, 其中 $c_1c_1^{-1} \equiv 1 \pmod{p}$, $c_2c_2^{-1} \equiv 1 \pmod{q}$ 。现在 x 模 p 会得到 a , 但是 $c_2c_2^{-1}b$ 多出来的部分不知道是什么。 x 模 q 会得到 b , 但是又多出了 $c_1c_1^{-1}a$ 。怎么解决? 也就是说, 怎么能做到在模 p (或者模 q) 的同时要使那些讨厌的部分“消失”? 建议读者在此处暂停阅读, 自己思考一段时间。

其实, 刚才的问题不难, 如果把 $c_2c_2^{-1}b$ 这一项中的 c_2 人为设置为 p , 在模 p 的时候, 这一项就被“消去”了。同理, 把 $c_1c_1^{-1}a$ 这一项中的 c_1 人为设置为 q 即可。这也大致解释了, 为什么定理的条件需要 p 与 q 互素。如果以上的问题读者在提示下都能准确找到答案, 那么恭喜你, 你已经重新地“发明”了中国剩余定理。请看以下证明。

证明 使用构造法。因为 p 与 q 互素, 则存在 p^{-1} 和 q^{-1} 分别使得 $pp^{-1} \equiv 1 \pmod{q}$ 和

$qq^{-1} \equiv 1 \pmod{p}$ 。令整数 x 为：

$$x = aqq^{-1} + bpp^{-1}$$

容易验证, x 同时满足两个同余式。最后需要证明在模 $n = pq$ 情况下, x 是唯一解。即需要证明如果存另一个解, 它也必然与 x 模 n 同余。假设 $\exists y \neq x$ 是另一个解, 则存在 $t \in \mathbb{N}$ 使得 $(y - x) = tp$ 。此时是把 x 和 y 理解为 $a + kp$ 的形式。类似, 如果把 x 和 y 理解为 $b + kq$ 的形式, 则存在 $s \in \mathbb{N}$ 使得 $(y - x) = sq$ 。因为 p 与 q 互素, 则存在某个 $k \in \mathbb{N}$, 使得 $(y - x) = kpq$ 。因此, $y \equiv x \pmod{n}$ 。

例 10.2 求解以下同余方程组：

$$x \equiv 2 \pmod{5}$$

$$x \equiv 3 \pmod{7}$$

利用中国剩余定理, 解法如下：

1. 记 $a = 2$, $b = 3$, $p = 5$, $q = 7$ 和 $n = pq = 35$;
2. 使用 *egcd* 算法求解 p^{-1} 和 q^{-1} , 它们分别使得 $pp^{-1} \equiv 1 \pmod{q}$ 和 $qq^{-1} \equiv 1 \pmod{p}$ 。得到 $p^{-1} = 3$, $q^{-1} = 3$;
3. 令 $y \equiv aqq^{-1} + bpp^{-1} \pmod{n}$, $y = 17$;
4. 验证可知, y 是正确解。

例 10.3 利用 SageMath 的 *crt* 命令, 可以方便利用 CRT 求解同余方程组。比如求解例 10.2:

```
1 sage: a = 2; b = 3; p = 5; q = 7;
2 sage: crt([a,b],[p,q])
3 17
```

对于多于两个同余式的同余方程组, 有一个更一般化的 CRT 版本, 描述如下：

定理 10.2. 中国剩余定理--推广版

设 m_1, m_2, \dots, m_n 是两两互素的整数, 对以下 n 个同余式的同余方程组

$$x \equiv a_1 \pmod{m_1}$$

...

$$x \equiv a_n \pmod{m_n}$$

存在模 M 的唯一解, 其中 $M = m_1 m_2 \cdots m_n$ 。



证明 构造法。令 $M = \prod_{i=1}^n m_i$, 记 $b_i = M/m_i$ 。存在 b_i^{-1} 使得 $b_i b_i^{-1} \equiv 1 \pmod{m_i}$ 。则

$$x = \sum_{i=1}^n a_i b_i b_i^{-1} \pmod{M}$$

是方程组的唯一解。

稍微把中国剩余定理的要求增强一点点, 要求同余方程组的模数分别是不同的素数, 然后可以继续大开脑洞, 请跟上思路。在证明中国剩余定理的时候我们问: 给定 a 和 p , 找一个值 a' , 使得

$$aa' \equiv a \pmod{p}$$

乘法逆元并不是唯一选择。费尔马小定理告诉我们，如果 p 是素数，会有很多的数 $c \in \mathbb{Z}$ 且 $\gcd(c, p) = 1$ 使得：

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

所以，某个数 b 的 $p-1$ 次方也能构造出一个 a 的相似数。因此，我们构造出来的 x 大致可以是这个形式：

$$ac_1^{p-1} + bc_2^{q-1}$$

请问， c_1 和 c_2 分别选择什么可以使得同余方程组得到满足呢？答案不难想，分别取 q 和 p 即可。于是我们很可能会得到一个试头剩余定理。

定理 10.3. 试头剩余定理

设 p 和 q 是不等的两个素数， $n = pq$ 。对任意的 $a \in \mathbb{Z}_p$ 和 $b \in \mathbb{Z}_q$ ，存在唯一解 x ， $0 \leq x < n$ 使得：

$$x \equiv a \pmod{p}$$

$$x \equiv b \pmod{q}$$

且 $x = (aq^{p-1} + bp^{q-1}) \bmod n$ 。



请验证该定理是否成立？

10.2 代数视角下的中国剩余定理

中国剩余定理是一种高效的工具，在特定的时候，它也是一种抽象的表达，从代数的角度认识某些数字系统的方法。在很多情况下，当我们说“哦，这是中国剩余定理!!”的时候我们并不一定在说“这里有一种高效的计算方法”，而往往是说“这类数值可看为具有这样的形式”。比如，设 $n = pq$ ， p 和 q 是两个互素的正整数。给定一个任意正整数 x ，它可以表达为一个唯一的序对： $([x \bmod p], [x \bmod q])$ 。这种思想可表达为以下定理。

定理 10.4. 中国剩余定理的代数版本

设 $n = pq$ ， $p > 1$ 和 $q > 1$ 是两个互素的正整数。则

$$\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q \quad \text{且} \quad \mathbb{Z}_n^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*。$$



证明

1. 定义从 \mathbb{Z}_n 到 $\mathbb{Z}_p \times \mathbb{Z}_q$ 的映射 ϕ 为：

$$\phi(x) = ([x \bmod p], [x \bmod q])$$

2. 证明映射 ϕ 是一种双射，即证明 ϕ 是满射且单射。满射显然，因为根据中国剩余定理（定理10.1），任意序对中的两个同余式在模 n 下有唯一解。证明单射即证明，如果对任意正整数 $a, b < n$ ，有 $([a \bmod p], [a \bmod q]) = ([b \bmod p], [b \bmod q])$ ，则 $a = b$ 。再次根据中国剩余定理（定理10.1），可得。

3. 证明映射 ϕ 保持群操作, 即需要证明:

$$\begin{aligned}\phi(a+b) &= ([(a+b) \bmod p], [(a+b) \bmod q]) \\ &= ([a \bmod p], [a \bmod q]) + ([b \bmod p], [b \bmod q]) \\ &= \phi(a) + \phi(b)\end{aligned}$$

这就证明了 $\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q$, 证明 \mathbb{Z}_n^* 与 $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ 同构类似, 留作课后作业。



思考 请思考这个问题: 如何使用第一同构定理来证明映射 ϕ 的单射性? 留为课后作业。

例 10.4 设 $15 = 5 \cdot 3$. $\mathbb{Z}_n^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$ 同构于 $\mathbb{Z}_5^* \times \mathbb{Z}_3^*$ 因为有以下一一对应关系:

$$1 \leftrightarrow (1, 1) \quad 2 \leftrightarrow (2, 2) \quad 4 \leftrightarrow (4, 1) \quad 7 \leftrightarrow (2, 1)$$

$$8 \leftrightarrow (3, 2) \quad 11 \leftrightarrow (1, 2) \quad 13 \leftrightarrow (3, 1) \quad 14 \leftrightarrow (4, 2)$$

例 10.5 求解 $14 \cdot 13 \bmod 15$. 因为 $14 \leftrightarrow (4, 2)$ 且 $13 \leftrightarrow (3, 1)$, 有:

$$(4, 2) \cdot (3, 1) = ([4 \cdot 3 \bmod 5], [2 \cdot 1 \bmod 3]) = (2, 2)$$

注意到, $(2, 2) \leftrightarrow 2$, 所以 2 是最终答案。

例 10.6 求解 $11^{53} \bmod 15$. 因为 $11 \leftrightarrow (1, 2)$ 且 $2 \equiv -1 \pmod{3}$, 所以:

$$(1, 2)^{53} = ([1^{53} \bmod 5], [-1^{53} \bmod 3]) = (1, -1 \bmod 3) = (1, 2)$$

最后, $11^{53} \bmod 15 = 11$.

例 10.7 设 $n = pq$ 为合数, p 和 q 是两个不同的素数。以下等式在模 n 的意义上有多少个解?

$$x^2 \equiv 1 \pmod{n}$$

为此, 需要解以下两个同余方程

$$x^2 \equiv 1 \pmod{p}$$

$$x^2 \equiv 1 \pmod{q}$$

根据命题 3.3, 以上两式分别有两个不同的解。因此在模 n 的意义上总共有 4 个解。同时, 这也就解决了第 9 章例 9.11 的一个遗留问题。

在需要大量使用 \mathbb{Z}_n^* 群元素的操作时, 比如 RSA 算法的加密解密和签名, 可以把群元 $x \in \mathbb{Z}_n^*$ 转换为 $(a, b) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$ 上的元素。然后群 \mathbb{Z}_n^* 中元素的操作就变为对群 \mathbb{Z}_p^* 和群 \mathbb{Z}_q^* 元素的操作。这样通常比较高效, 因为输入数据长度减半导致运行时间不仅仅减半。

Chapter Note

To do...

第十章习题

1. 手动计算 $2000^{2019} \pmod{221}$, 不允许使用电脑或者其他电子设备。
2. 运用 CRT 求解:

$$x \equiv 8 \pmod{11}$$

$$x \equiv 3 \pmod{19}$$

3. 运用 CRT 求解:

$$x \equiv 1 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

$$x \equiv 3 \pmod{9}$$

$$x \equiv 4 \pmod{11}$$

4. 设 p 和 q 为互素的正整数, $a > 0$ 为一个正整数, 如果

$$x \equiv a \pmod{p}$$

$$x \equiv a \pmod{q}$$

x 模 pq 等于什么? 为什么?

5. 实现一个利用 CRT 求解同余方程的程序 (Python 或者 C 语言都可以)。
6. 使用 CRT 证明第5的定理5.6。
7. 请使用第一同构定理证明定理10.4中定义的映射 ϕ 的单射性。
8. 完成定理10.4的证明, 即证明 \mathbb{Z}_n^* 与 $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ 同构。



第十一章 二次剩余

内容提要

□ 二次剩余

□ 勒让德符号

□ 欧拉准则

□ 二次互反律

11.1 二次剩余与二次非剩余

设 p 是一个奇素数, a 是与 p 互素的整数, a 是否是模 p 的平方数? 换言之, 需要判断是否存在 $x \in \mathbb{Z}$ 使得:

$$x^2 \equiv a \pmod{p}.$$

这就是本章的重点主题。

表面上看, 这似乎很容易, 正如我们之前做过的, 只需要把 1 到 $p-1$ 之间的数做平方, 然后看是否有一个数会等于 a 。

例 11.1 设 $p = 11$, $a = 5$, 判断 a 是否模 p 的平方数。只需要写一个简单的 Python 程序, 对 1 到 10 之间的数做平方。

```
1 p = 11
2 for i in range(1, p):
3     print(pow(i, 2, p), end=' ')
4 print(end='\n')
```

得到的输出是:

1, 4, 9, 5, 3, 3, 5, 9, 4, 1

根据输出可以知道:

$$4^2 \equiv 5 \pmod{p}, \text{ 并且 } 7^2 \equiv 5 \pmod{p}.$$

当然, 穷举法不是我们想要的, 知道求解的命令也远远不够, 我们想要讨论的比这复杂得多。然而通过这些简单代码和命令我们可以获取数据, 从而发现某些模式或者规律, 再从中发现更有价值的解决方案。例如, 如果分别设 $p = 5, 7, 13$, 获得的数据可显示如下:

a	a^2
1	1
2	4
3	4
4	1

表 11.1: $a^2 \bmod 5$

a	a^2
1	1
2	4
3	2
4	2
5	4
6	1

表 11.2: $a^2 \bmod 7$

a	a^2
1	1
2	4
3	9
4	3
5	12
6	10
7	10
8	12
9	3
10	9
11	4
12	1

表 11.3: $a^2 \bmod 13$

从上表中我们可以发现什么规律呢？例如，考虑所有表格最后一行，会得到这样一个结论：

$$(p-1)^2 \equiv 1 \pmod{p}.$$

为什么呢？很容易证明：

$$(p-1)^2 = p^2 - 2p + 1 \equiv 1 \pmod{p}.$$

这个规律意味着同余式 $x^2 \equiv 1 \pmod{p}$ 有两个解 1 and $p-1$ 。这是我们已知结论，参看命题 3.3。

另一个有趣的结论是，所有表格的第二列都呈现一种“翻转对称性”，也就是说，如果你把上面表格从中间对折，平方数部分竟然是重合的。用公式表达如下：

$$(p-a)^2 \equiv a^2 \pmod{p}$$

证明与上类似，展开 $(p-a)^2$ ，

$$(p-a)^2 = p^2 - 2pa + a^2$$

记 $b = a^2 \bmod p$ ，此时同余式 $x^2 \equiv b \pmod{p}$ 有解，则必然是两个不同余的解： a 和 $p-a$ 。注意， $p-a \equiv a \pmod{p}$ 当且仅当 $a = 0$ 。把这个结论写成命题如下。

命题 11.1

设 p 是一个奇素数， b 是不被 p 整除的整数。则同余式

$$x^2 \equiv b \pmod{p}$$

或者没有解，或者有且仅有两个模 p 非同余解。



证明 根据以上分析可知，同余式 $x^2 \equiv b \pmod{p}$ 若有解则必有两个解，设为 x_0 和 x_1 。

现在只需要证明它不会有多于两个的不同余解。已知：

$$x_0^2 \equiv x_1^2 \equiv b \pmod{p}$$

整理可得：

$$x_0^2 - x_1^2 = (x_0 - x_1)(x_0 + x_1) \equiv 0 \pmod{p}$$

即 $p \mid (x_0 - x_1)$ 或 $p \mid (x_0 + x_1)$ 。也就是说， $x_0 \equiv x_1 \pmod{p}$ 或者 $x_0 \equiv -x_1 \pmod{p}$ 。因此，如果存在解，只有两个不同余解。□

在进一步讨论之前，需要以下定义。

定义 11.1. 二次剩余

设 $m, n \in \mathbb{Z}$ ，且 $\gcd(m, n) = 1$ 。如果存在 $x \in \mathbb{Z}$ ，使得 $m \equiv x^2 \pmod{n}$ ，则称 m 为模 n 的二次剩余 (Quadratic Residue，简记为 **QR**)，否则称 m 为模 n 的二次非剩余 (Quadratic Non-Residue，简记为 **QNR**)。



例 11.2 根据表 11.2 和表 11.3 我们知道，模 7 的 **QR** 是 $\{1, 2, 4\}$ ；模 13 的 **QR** 是 $\{1, 3, 4, 9, 10, 12\}$ ，模 7 的 **QNR** 是 $\{3, 5, 6\}$ ；模 13 的 **QNR** 是 $\{2, 5, 6, 7, 8, 11\}$ 。如果使用 Sage 的 `quadratic_residues` 命令可以帮助我们做计算。代码如下：

```
1 sage: quadratic_residues(7)
2 [0, 1, 2, 4]
3 sage: quadratic_residues(13)
4 [0, 1, 3, 4, 9, 10, 12]
5
```

根据命题 11.1，可以得到以下定理。

定理 11.1

设 p 为奇素数，则刚好存在 $(p-1)/2$ 个模 p 的 **QR** 和 $(p-1)/2$ 个模 p 的 **QNR**。♡

证明 对 \mathbb{Z}_p^* 中的所有整数做平方，映射到模 p 的 **QR**。命题 11.1 告诉我们，恰好有两个不同余的整数映射到一个 **QR**。现在我们有 $p-1$ 个平方要考虑，则恰好有 $(p-1)/2$ 个模 p 的 **QR**。剩下的 $(p-1)/2$ 整数就是模 p 的 **QNR**。□

命题 11.2

用 \mathbb{QR}_p 表示模 p 的 **QR** 的集合， \mathbb{QR}_p 在乘法上成群。♠

证明 容易，检验所有的群公理即得。□

从 \mathbb{Z}_p^* 到 \mathbb{QR}_p 的映射 ϕ 定义为： $\forall a \in \mathbb{Z}_p^*, a \mapsto a^2$ 。容易验证 ϕ 是群同态，根据以上观察可知， $\text{Ker } \phi = \{1, p-1\}$ 。再根据第 9 章的第一同构定理 (定理 9.3)，可以对定理 11.1 给出一种新的证明：

$$|\mathbb{QR}_p| = |\mathbb{Z}_p^*| / |\text{Ker } \phi| = (p-1)/2$$

请读者完成其中具体细节，留作课后作业。

既然 \mathbb{QR}_p 是一种乘法群, 那么根据群的封闭性, 将两个模 p 的 **QR** 相乘还会得到一个 **QR**, 表示为:

$$\mathbf{QR} \times \mathbf{QR} = \mathbf{QR}.$$

然而, 现在还不清楚 **QR** 乘 **QNR**, 或者两个 **QNR** 相乘会得到什么。建议读者写一个程序进行相关计算, 并猜测结果。相信大家不难找到答案, 请看以下命题。

命题 11.3

设 p 为奇素数, 则

1. 两个模 p 的 **QR** 相乘得到一个 **QR**, 记为:

$$\mathbf{QR} \times \mathbf{QR} = \mathbf{QR},$$

2. 一个模 p 的 **QR** 和一个模 p 的 **QNR** 相乘得到一个模 p 的 **QNR**, 记为:

$$\mathbf{QR} \times \mathbf{QNR} = \mathbf{QNR},$$

3. 两个模 p 的 **QNR** 相乘得到一个 **QR**, 记为:

$$\mathbf{QNR} \times \mathbf{QNR} = \mathbf{QR}$$

证明 第一个命题已证, 只需考虑后两个命题。设 b 是一个 **QNR**, a 是一个 **QR**, 则存在 $a_1 \in \mathbb{Z}$ 使得 $a \equiv a_1^2 \pmod{p}$ 。如果 ab 是一个 **QR**, 则存在 $c \in \mathbb{Z}$ 使得:

$$c^2 \equiv ab \equiv a_1^2 b \pmod{p}$$

已知 $\gcd(a_1, p) = 1$, 则存在 $a_1^{-1} \in \mathbb{Z}$ 使得 $a_1 a_1^{-1} \equiv 1 \pmod{p}$, 因此有:

$$c^2 (a_1^{-1})^2 \equiv b \pmod{p}$$

这表示 b 是一个 **QR**, 与假设矛盾。

考虑最后一个命题。设 a 是一个 **QNR**, 且 $p \nmid a$, 则根据群的性质可知:

$$a\mathbb{Z}_p^* = \mathbb{Z}_p^*.$$

我们还知道在 \mathbb{Z}_p^* 中恰有 $(p-1)/2$ 个 **QR** 和 $(p-1)/2$ 个 **QNR**。已证明, 用一个 **QR** 乘 a 会得到一个 **QNR**, 所以用 $(p-1)/2$ 个 **QR** 乘 a 得到了 \mathbb{Z}_p^* 中所有 $(p-1)/2$ 个 **QNR**。因此, 用 a 乘一个 **QNR**, 唯一的可能就是得到 \mathbb{Z}_p^* 中的一个 **QR**。□

11.2 勒让德符号与欧拉准则

定义 11.2. 勒让德符号

设 p 是一个奇素数, 并设 $n \in \mathbb{Z}$ 。定义勒让德符号 (Legendre symbol) $\left(\frac{a}{p}\right)$ 为:

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{若 } a \text{ 是模 } p \text{ 的 } \mathbf{QR} \\ -1 & \text{若 } a \text{ 是模 } p \text{ 的 } \mathbf{QNR} \\ 0 & \text{若 } p \mid a. \end{cases}$$

Sage 命令 `legendre_symbol` 可以计算勒让德符号。请看以下代码

例 11.3

```

1 sage: a = 1; p = 17
2 sage: legendre_symbol(a, p)
3 1
4 sage: a = 131; p = 65537
5 sage: legendre_symbol(a, p)
6 -1
7

```

命题 11.4. 勒让德符号的若干属性

设 p 是奇素数, $a, b \in \mathbb{Z}$ 且不被 p 整除。则有:

1. 如果 $a \equiv b \pmod{p}$, 则 $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$;
2. $\left(\frac{a}{p}\right) \left(\frac{b}{p}\right) = \left(\frac{ab}{p}\right)$;
3. $\left(\frac{a^2}{p}\right) = 1$ 。



证明 容易, 留作课后练习。

从抽象代数的视角去审视勒让德符号。首先注意到 $\{1, -1\}$ 在模素数 p 的乘法下成群。然后, 定义映射 $\psi: \mathbb{Z}_p^* \rightarrow \{\pm 1\}$ 为 $\psi(a) = \left(\frac{a}{p}\right)$, $\forall a \in \mathbb{Z}_p^*$ 。可以验证, 这是一个满同态。具体细节请读者自行完成, 留作课后练习。

欧拉给出了计算勒让德符号的简单方法。在讲解定理与证明之前让我们思考以下问题, 并寻找发现欧拉准则的思路。给定奇素数 p , 如果 $a \in \mathbb{Z}_p^*$, 则勒让德符号 $\left(\frac{a}{p}\right)$ 不是等于 1 就是等于 -1 , 1 和 -1 提示我们的是什么? 难道不应该是模 p 下对 1 开根号? 这不难想到。模 p 下什么数值等于 1? 根据费尔马小定理, 想到 $a^{p-1} \bmod p$ 应该也不难。 $a^{p-1} \bmod p$ 可以开根号吗? 注意到, $p-1$ 必为偶数, 所以, $a^{(p-1)/2} \bmod p$ 就是开根号了, 不是等于 1 就是等于 -1 。接下来只需要分别检验 $a^{(p-1)/2} \bmod p = 1$ 和 $a^{(p-1)/2} \bmod p = -1$ 与 $\left(\frac{a}{p}\right)$ 之间的关系即可。在看以下证明之前, 建议读者先自行检验。

定理 11.2. 欧拉准则

设 p 是奇素数, $a \in \mathbb{Z}$, 且 $\gcd(a, p) = 1$ 。那么,

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}.$$



证明 需要考虑两种情形: $\left(\frac{a}{p}\right) = 1$ 和 $\left(\frac{a}{p}\right) = -1$ 。

1. 假设 $\left(\frac{a}{p}\right) = 1$, 即存在 $x \in \mathbb{Z}$ 使得 $x^2 \equiv a \pmod{p}$ 。根据费尔马小定理, 有:

$$x^{(p-1)} \equiv 1 \pmod{p}$$

和

$$x^{(p-1)} \equiv (x^2)^{(p-1)/2} \equiv a^{(p-1)/2} \pmod{p}$$

因此,

$$a^{(p-1)/2} \equiv 1 \equiv \left(\frac{a}{p}\right) \pmod{p}.$$

2. $\left(\frac{a}{p}\right) = -1$, 这说明 $x^2 \equiv a \pmod{p}$ 无解。根据第一种情形的结论, 有

$$a^{(p-1)/2} - 1 \not\equiv 0 \pmod{p}$$

再次使用费尔马小定理,

$$0 \equiv a^{(p-1)} - 1 \equiv (a^{(p-1)/2} - 1)(a^{(p-1)/2} + 1) \pmod{p}$$

因为 $a^{(p-1)/2} - 1 \not\equiv 0 \pmod{p}$, 所以必须是

$$a^{(p-1)/2} + 1 \equiv 0 \pmod{p}$$

即,

$$a^{(p-1)/2} \equiv -1 \equiv \left(\frac{a}{p}\right) \pmod{p}$$

□

根据欧拉准则, 计算勒让德符号的程序如下。当然, 做一次指数运算并不能算是难题, 只是需要思考, 效率能否得到提高呢?

```
1 def legendre_symbol(a, p):
2     if pow(a, (p-1)/2, p) == 1:
3         return 1
4     else:
5         return -1
6
```

以上从数论的角度理解了欧拉准则, 同时也可以从抽象代数的角度去理解。注意到, 欧拉准则考虑从 \mathbb{Z}_p^* 到 $a^{(p-1)/2}$ 的映射, $\forall a \in \mathbb{Z}_p^*$, 记为 ϕ 。可验证, ϕ 是一种群同态。结合勒让德符号的满同态 $\psi(a) = \left(\frac{a}{p}\right)$ 一起考虑, 如果 $\text{Ker } \psi = \text{Ker } \phi$, 则欧拉准则得证。两个方向, $\text{Ker } \psi \subset \text{Ker } \phi$ 和 $\text{Ker } \phi \subset \text{Ker } \psi$ 。具体细节请读者自行完成, 留作课后练习。

推论 11.1

设 p 是一个奇素数, 则:

$$\left(\frac{-1}{p}\right) = \begin{cases} 1 & \text{如果 } p \equiv 1 \pmod{4}; \\ -1 & \text{如果 } p \equiv -1 \pmod{4}. \end{cases}$$



证明 因为 $p \equiv 1 \pmod{4}$ 意味着存在 $k \in \mathbb{Z}$ 使得 $p = 4k + 1$ 。根据欧拉准则,

$$\left(\frac{-1}{p}\right) \equiv (-1)^{(p-1)/2} \equiv (-1)^{(4k+1-1)/2} \equiv 1 \pmod{p}.$$

$p \equiv -1 \pmod{4}$ 的情形类似, 请读者自己完成。 □

以下描述著名的定理二次互反律 (Law of Quadratic Reciprocity), 并不给出证明, 因为其中第三部分的证明不容易, 所以不在本书的讨论范围之内。需要强调的是, 二次互反律非常重要, 是一个优美且高效实用的定理, 最好大家能知道并会使用它。

定理 11.3. 二次互反--版本 1

设 p, q 是两个不同的奇素数, 则

$$\left(\frac{-1}{p}\right) = \begin{cases} 1 & \text{如果 } p \equiv 1 \pmod{4}; \\ -1 & \text{如果 } p \equiv -1 \pmod{4}. \end{cases}$$

$$\left(\frac{2}{p}\right) = \begin{cases} 1 & \text{如果 } p \equiv 1 \text{ 或 } 7 \pmod{8}; \\ -1 & \text{如果 } p \equiv 3 \text{ 或 } 5 \pmod{8}. \end{cases}$$

$$\left(\frac{q}{p}\right) = \begin{cases} \left(\frac{p}{q}\right) & \text{如果 } p \equiv 1 \pmod{4} \text{ 或 } q \equiv 1 \pmod{4}; \\ -\left(\frac{p}{q}\right) & \text{如果 } p \equiv q \equiv 3 \pmod{4}. \end{cases}$$



以下是二次互反律的简洁版本, 它与之前的版本等价。

定理 11.4. 二次互反律--版本 2

设 p, q 是两个不同的奇素数, 则

$$\left(\frac{p}{q}\right) \left(\frac{q}{p}\right) = (-1)^{((p-1)/2)((q-1)/2)}.$$

**例 11.4**

$$\left(\frac{13}{17}\right) = \left(\frac{4}{13}\right) = \left(\frac{2^2}{13}\right) = 1$$

因此 13 是模 17 的 **QR**。

$$\left(\frac{14}{137}\right) = \left(\frac{2}{137}\right) \left(\frac{7}{137}\right) = \left(\frac{7}{137}\right) = \left(\frac{137}{7}\right) = \left(\frac{4}{7}\right) = 1$$

因此 14 是模 137 的 **QR**。

11.3 计算平方根

在这一节, 我们考虑这样一个问题。给定任意整数 n , 如何求解以下同余方程:

$$x^2 \equiv a \pmod{n}.$$

从简单情形开始考虑。如果同余式的模数是一个素数, 则需要解的是:

$$x^2 \equiv a \pmod{p}$$

首先, 需要判定该方程是否有解。运用欧拉准则, 如果勒让德符号 $\left(\frac{a}{p}\right) = 1$, 方程有解, 进入下一步骤。否则, 返回“无解”提示, 程序结束。

接下来, 我们要做的是“凑”一个平方数, 它模 p 与 a 同余。思路是, 给 a 乘上一个模 p 等于 1 的数, 并且得到的值还能“凑”成一个平方数。首先想到的应该是 aa^{-1} , 但是很快就会否决掉这个想法, 因为这样乘上去什么没有得到, 还是一个 a 。那么找任意不等于 a 的其他整数呢, 比如, 任取 $b \in \mathbb{Z}$, 乘 bb^{-1} ? 也不行, 因为不知道怎么凑出平方数。还是要乘一个与 a 相关, 且模 p 等于 1 的数, 应该是什么? 建议大家在此掩卷思

考几分钟，回忆最近的知识点中有没有这样的数值。



思考 根据费尔马小定理, $a^{p-1} \equiv 1 \pmod{p}$, 似乎也满足我们的需求。给 a 乘上 a^{p-1} 能不能带来好的结果呢? 为什么或者为什么不?

现在已知, 勒让德符号 $\left(\frac{a}{p}\right) = 1$, 根据欧拉准则, 即有 $a^{(p-1)/2} \equiv 1 \pmod{p}$ 。于是很高兴地发现:

$$a \equiv a^{(p-1)/2} \cdot a \equiv a^{(p+1)/2} \equiv (a^{(p+1)/4})^2 \pmod{p}$$

$(a^{(p+1)/4})^2$ 似乎就是我们需要的答案。不过, 且慢, 请问 $(p+1)/4$ 一定是整数? 如果是, 那么求解就成功了。所以, 倒不如问, 在什么情况下 $(p+1)/4$ 一定是整数?

要使得 $(p+1)/4$ 为整数, 就是需要存在 $k \in \mathbb{Z}$, 使得 $(p+1)/4 = 4k$ 。简单的计算可知, 其实就是要求 $p \equiv -1 \pmod{4}$, 或者写成 $p \equiv 3 \pmod{4}$ 。所以, 如果 p 满足模 4 余 3, 则同余方程有解 $\pm a^{(p+1)/4}$ 。使用代码表示以上分析如下:

```
1 Input: 整数a和p
2 Output: 求x使得x^2 模p与a同余
3 def find_sqrt(a, p):
4     assert (p + 1) % 4 == 0    # p mod 4 余 3;
5     assert is_prime(p)        # p 为素数;
6     if legendre_symbol(a, p) != 1: return []    #勒让德符号不为1则无解;
7     b = (p + 1) // 4
8     return [a^b % p, -a^b % p]
```

如果 $n = pq$, 且 p 和 q 都是满足 $p \equiv q \equiv 3 \pmod{4}$ 的素数, 解同余方程 $x^2 \equiv a \pmod{n}$ 等价于先解以下两个模素数的同余方程。

$$x^2 \equiv a \pmod{q}$$

$$x^2 \equiv a \pmod{p}$$

然后, 使用中国剩余定理可以找到所有解。至于模 4 余 1 的情形, 则暂且不讨论。

Chapter Note

To be done.

第十一章习题

1. 证明命题11.2。
2. 使用群论的方法证明定理11.1。
3. 定义映射 $\psi: \mathbb{Z}_p^* \rightarrow \{\pm 1\}$ 为 $\psi(a) = \left(\frac{a}{p}\right)$, $\forall a \in \mathbb{Z}_p^*$ 。请证明这是一个满同态。
4. 设 p 是奇素数, 请证明 \mathbb{Z}_p^* 的所有生成元都是模 p 的二次非剩余。
5. 证明命题11.4。
6. 给出推论11.1的完整证明。

7. 使用抽象代数的语言重新证明欧拉准则。
8. 利用二次互反律，写程序完成 $\left(\frac{q}{p}\right)$ 的计算， p 和 q 是任意的奇素数。
9. 设 $n = 7 * 11$ ， $a = 5$ ，请手动计算求 x 使得 $x^2 \equiv a \pmod{n}$ 。
10. 写一个程序求解同余方程 $x^2 \equiv a \pmod{n}$ ，其中要求 $n = pq$ ，且 pq 是模 4 余 3 的两个不同素数。



第十二章 环与域

内容提要

- 环
- 整环
- 理想

- 多项式环
- 域
- 有限域

代数结构群针对一个特定的集合，定义该集合中元素的一种操作，不是“加法”就是“乘法”，比如整数的加法群 \mathbb{Z}_n 和整数的乘法群 \mathbb{Z}_n^* 。然而常识告诉我们，对于整数应该是同时具有加法与乘法，为什么不定义一种同时具有两种元素操作的代数结构呢？这一章讨论两种代数结构：环（*Ring*）和域（*Field*），它们都定义两种元素操作。

12.1 环

在计算机系统中，最自然的代数结构就是环。环是一种包括两种二元操作的代数结构，具体定义如下：

定义 12.1. 环

环 R 是一个非空集合，在 R 上有两种封闭的二元操作：加法（记为 $+: R \times R \mapsto R$ ）和乘法（记为 $*: R \times R \mapsto R$ ），并且满足以下条件：

1. 在加法（ $+$ ）上 R 是一个阿贝尔群，加法的单位元记为 0 ，加法上的逆元记为 $-a$ ；
2. R 在乘法（ $*$ ）上满足结合律；
3. 乘法在加法上满足分配律。



具体地表示为公式，对任意 $a, b, c \in R$ ，环 R 满足以下公理：

$$a + (b + c) = (a + b) + c \quad (+ \text{ 的结合律}) \quad (12.1)$$

$$a + b = b + a \quad (+ \text{ 的交换律}) \quad (12.2)$$

$$a + 0 = 0 + a \quad (+ \text{ 的单位元}) \quad (12.3)$$

$$a + (-a) = (-a) + a = 0 \quad (+ \text{ 的逆元}) \quad (12.4)$$

$$a * (b * c) = (a * b) * c \quad (* \text{ 的结合律}) \quad (12.5)$$

$$a * 1 = 1 * a = a \quad (* \text{ 的单位元}) \quad (12.6)$$

$$(a + b) * c = (a * c) + (b * c) \quad (\text{右分配律}) \quad (12.7)$$

$$a * (b + c) = (a * b) + (a * c) \quad (\text{左分配律}) \quad (12.8)$$

如果 R 在乘法上也满足交换律, 即 $\forall a, b \in R$, 有

$$a * b = b * a$$

则称 R 为交换环, 否则称为非交换环。以后, 在不发生歧义时, 表述环的乘法往往省略乘法符号, 比如把 $a * b$ 写成 ab 。

如果 R 在乘法上具有单位元, 即存在 $1 \in R$, $1 \neq 0$, 且对任意的 $a \in R$, 有 $a1 = 1a = a$, 则称环 R 为带单位元的环。下面我们重点讨论带单位元的环, 不特别声明都假定环带单位元。

例 12.1

1. $R = \{0\}$ 是只有一个元素的最小环, 称为平凡环或零环。如果一个环中, $1 \neq 0$, 则这个环是非平凡环。一个最小的非平凡环就是 $R = \{0, 1\}$, 或者记为 \mathbb{Z}_2 。
2. 在普通意义的加法和乘法上, 容易验证 $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ 都是交换环。比如, 考虑 \mathbb{Z} , 它在加法上是群, 在乘法上满足结合律, 有单位元 1, 交换律和分配律则是大家熟知的事实。
3. 整数中所有的偶数在一般的加法与乘法上形成环, 同样记为 $2\mathbb{Z}$, 这是一个不带单位元的环。更一般地, 对任意整数 $n \in \mathbb{Z}$, $n\mathbb{Z}$ 在一般的加法与乘法上形成环。
4. 对任意正整数 $n \in \mathbb{N}$, \mathbb{Z}_n 是模 n 的加法群。如果给加法群 \mathbb{Z}_n 配置上乘法 $*$, 乘法 $*$ 被定义为整数上的模 n 乘法, 即 $\forall a, b \in \mathbb{Z}_n$, $a * b \triangleq ab \bmod n$ 。容易验证 \mathbb{Z}_n 是一个交换环。注意, 此时 \mathbb{Z}_n 的元素在乘法上不一定存在逆元。 \mathbb{Z}_n 是计算机系统中最基本的代数结构, 因为计算机系统里的算术都是模 n 的加法和模 n 的乘法, 并且要求 n 是 2 的某个指数。
5. 不难想到, 如果取 p 为任意素数, \mathbb{Z}_p 在模 p 的加法与模 p 的乘法下成环, 而且 \mathbb{Z}_p 的所有元素在乘法上都有逆元。以后我们将重点讨论这种特殊的环。
6. 对任意环 R , R 的直积 $R \times R$ (参考群论的定义 9.6) 形成环, 环的加法与乘法定义为 $\forall (a, b), (c, d) \in R \times R$ 序对, $(a, b) + (c, d) = (a + c, b + d)$ 和 $(a, b)(c, d) = (ac, bd)$ 。
7. 实数上的 $n \times n$ 矩阵在普通的矩阵加法和矩阵乘法上形成环, 也称为矩阵环, 记为 $M_n(\mathbb{R})$ 。这是一种非交换环。

命题 12.1

设 R 是一个环, 且 $a, b \in R$, 则有:

1. $a0 = 0a = 0$
2. $a(-b) = (-a)b = -ab$
3. $(-a)(-b) = ab$



证明 根据分配律,

$$a0 = a(0 + 0) = a0 + a0$$

则 $a0 = 0$, 同理 $0a = 0$ 。同样根据分配律

$$ab + a(-b) = a(b + (-b)) = a0 = 0$$

所以, $-(ab) = a(-b)$, 同理 $-(ab) = (-a)b$ 。最后, 根据以上结论, $(-a)(-b) = -(a(-b)) =$

$$-(-(ab)) = ab.$$

注 命题12.1中的性质似乎非常平凡, 考虑环 \mathbb{Z} 或 \mathbb{Z}_n , 这些都是我们熟知的性质。然而, 考虑到抽象代数的“抽象性”, 则不可想当然地认为这些属性必然存在了。

环并不要求环元素在乘法上存在逆元, 一个显然的问题就是, 在乘法上, 消去律不必然存在! 如果修补定义而获得乘法的消去律呢? 最自然的想法就是让环中非零元素在乘法上成群。进而再问, 这是否必须的呢? 以上问题引出了整环 (Integral Domain) 的概念。

定义 12.2. 整环

给定一个环 R , 对任意元素 $a \in R$, 如果存在元素 $b \in R$ 使得 $b \neq 0$ 且 $a * b = 0$, 则称 a 为一个零因子 (zero divisor)。如果交换环 R 中没有除 0 以外的零因子, 即 $\forall a, b \in R$, 如果 $ab = 0$ 则有 $a = 0$ 或 $b = 0$, 则称 R 为整环。



例 12.2

1. 在普通意义的加法和乘法上, $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ 都是整环。比如, 考虑 $a, b \in \mathbb{Z}$, $ab = 0$ 当且仅当 $a = 0$ 或者 $b = 0$ 。其他实例, 验证类似。
2. 可验证偶数环 $2\mathbb{Z}$ 是整环。更一般地, 对任意的 $n \in \mathbb{Z}$, $n\mathbb{Z}$ 都是整环。
3. 已知 \mathbb{Z}_n 是一个交换环。但是, \mathbb{Z}_n 并不是整环。比如, \mathbb{Z}_{15} 中, $(3 * 5) \bmod 15 = 0$, 3 和 5 都是 \mathbb{Z}_{15} 中的零因子。
4. 矩阵环 $M_n(\mathbb{R})$ 不是整环, 因为存在 $A, B \in M_n(\mathbb{R})$, 使得 $AB = 0$ 但是 A 和 B 都不为 0。

命题 12.2. 消去律

设 D 是一个交换环, D 是整环当且仅当对任意元素 $a, b, c \in D$, 且 $a \neq 0$, 若 $ab = ac$, 则 $b = c$ 。



证明

1. 充分性. D 是整环, 则 D 中无零因子。若 $a \neq 0$ 且 $ab = ac$, 则 $a(b - c) = 0$, 则 $b - c = 0$, 即 $b = c$ 。
2. 必要性. 假设 D 中消去律成立, 即若 $ab = ac$, 则 $b = c$ 。设 $ab = 0$, 有 $ab = a0$ 。如果 $a \neq 0$, 根据消去律, 得到 $b = 0$ 。因此, a 不可能是零因子。

命题12.2解答了之前提出的问题, 整环可确保在较弱的条件下环的乘法消去律存在。

定义 12.3. 子环

给定环 R , R' 是 R 的子集, 如果 R' 在环 R 的加法和乘法上也形成环, 则称 R' 是 R 的子环, 记为 $R' \subset R$ 。



例 12.3

1. 容易验证以下子环序列: $\mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$ 。
2. 偶数环是整数环的子环, 即 $2\mathbb{Z} \subset \mathbb{Z}$ 。由此可知, 子环并不自然继承母环的单位元。当然, 对任意的 $n \in \mathbb{Z}$, 有 $n\mathbb{Z} \subset \mathbb{Z}$ 。

命题 12.3

给定环 R , R' 是 R 的子集。 R' 是 R 的子环, 当且仅当以下条件满足:

1. $R' \neq \emptyset$;
2. $\forall a, b \in R'$, 有 $ab \in R'$;
3. $\forall a, b \in R'$, 有 $a - b \in R'$ 。



证明 根据命题6.8易得, 留作课后练习。

第9章已经建立了一条紧密结合的知识链: 同构、同态、Kernel、正规子群、商群和同构定理。同样的, 在环论的知识体系中也存在类似的对应物, 也包括: 同构、同态、Kernel、商环、同构定理等等。以下简要给出相关定义与定理, 并重点讨论在环的语境下与群论中正规子群对应的概念: 理想 (Ideal)。

定义 12.4. 环同态与环同构

给定两个环 R 和 R' , 若映射 $\phi: R \mapsto R'$ 满足: $\forall a, b \in R$,

$$\phi(a + b) = \phi(a) + \phi(b)$$

$$\phi(ab) = \phi(a)\phi(b)$$

则称 ϕ 为一个环同态。如果 ϕ 是一个双射, 则 ϕ 是一个环同构。环同态 ϕ 的 Kernel 定义为以下集合:

$$\text{Ker } \phi = \{r \in R : \phi(r) = 0\}。$$

**命题 12.4. 环同态的性质**

设映射 $\phi: R \mapsto R'$ 是环同态, 则:

1. 如果 R 是交换环, 则 $\phi(R)$ 也是交换环。
2. 分别记 0 和 $0'$ 是 R 和 R' 的加法单位元, $\phi(0) = 0'$ 。
3. 分别记 1 和 $1'$ 是 R 和 R' 的乘法单位元, 如果 ϕ 是满射, 则 $\phi(1) = 1'$ 。



证明 留作课后练习。

例 12.4 命题12.4的证明与群论中对应的证明类似, 依葫芦画瓢很容易得到证明。必须提醒大家要思考的是, 命题为什么要求 ϕ 是满射才会有 $\phi(1) = 1'$? 请验证以下同态实例, 体会环同态与群同态的异同点。

1. 已知 \mathbb{Z} 是环, 定义映射 $\phi: \mathbb{Z} \mapsto \mathbb{Z}$ 为 $\phi(k) = 2k, \forall k \in \mathbb{Z}$, 即把所有整数映射到偶数 $2\mathbb{Z}$ 。已知 ϕ 是 \mathbb{Z} 到 \mathbb{Z} 的群同态, 但是可以验证 ϕ 不是一种环同态。
2. 对任意整数 $n \in \mathbb{Z}$, 已知 \mathbb{Z}_n 为环, 定义映射 $\phi: \mathbb{Z}_n \mapsto \mathbb{Z}_n$ 为 $\phi(a) = a^2 \bmod n, \forall a \in \mathbb{Z}_n$, 即把所有 \mathbb{Z}_n 的元素映射到平方数。可以验证 ϕ 不是一种环同态, 尽管这种映射在 \mathbb{Z}_n^* 中是一种群同态。
3. 考虑一种特殊的环同态, 定义 $\phi: 2\mathbb{Z} \mapsto \mathbb{Z}_2$ 为 $\forall k \in 2\mathbb{Z}, \phi(k) = k \bmod 2$ 。明显, ϕ 是同态, 但不是满同态, 它把所有偶数都映射到了 0 。将这种把任意 R 映射到零环 0 的环同态称为零同态。
4. 考虑一种更特殊的环同态, 对任意的环 R , 定义映射 $\phi: R \mapsto \mathbb{Z}_2$ 为 $\forall a \in R$,

$\phi(a) = 0$ 。可验证，这是一种零同态，显然不是满同态。特别提醒注意，如果环 R 有乘法单位元 1 ， $\phi(1) = 0$ 。并不是我们期望的映射到 \mathbb{Z}_2 的 1 。

5. 对任意环，已知 $R \times R$ 是环。定义映射 $\phi: R \times R \mapsto R \times R$ 为 $\forall(a, b) \in R \times R$, $\phi(a, b) = (a, 0)$ 。可验证，这是一个环同态，但并非满同态。注意，在 $\phi(R \times R)$ 中的单位元是 $(1, 0)$ ，但是 $R \times R$ 中的单位元是 $(1, 1)$ 。
6. 设 p 是任意一个奇素数，考虑 $2\mathbb{Z}$ 与 \mathbb{Z}_p 之间的映射 $\phi: 2\mathbb{Z} \mapsto \mathbb{Z}_p$ ，定义为 $\forall k \in 2\mathbb{Z}$, $\phi(k) = k \bmod p$ 。直观上看，该映射把所有的偶数做模 p 操作满射到环 \mathbb{Z}_p 。容易验证 ϕ 是满同态。值得注意的是，偶数环 $2\mathbb{Z}$ 没有乘法单位元，环 \mathbb{Z}_p 的乘法单位元是 1 ， $2\mathbb{Z}$ 中有无穷多的元素映射到 \mathbb{Z}_3 的乘法单位元 1 上。也就是说，即使乘法单位元必然映射为乘法单位元，也并非只有乘法单位元才映射为乘法单位元。

环同态的定义显得很自然直接，与群同态的定义、属性也非常类似，但是，只要揣摩一下以上实例还是能明显感觉到环同态与群同态的巨大区别。接下来介绍的这个定义则与群论的对应物有明显区别。回顾群论中的正规子群，它性质特殊，与 Kernel 有密切关系，并用于定义商群，有着非常重要的作用。在环论中与之对应的概念是理想。根据定义，子群 \mathbb{H} 是群 \mathbb{G} 的正规子群，则满足 $\forall g \in \mathbb{G}$,

$$g^{-1}\mathbb{H}g = \mathbb{H};$$

或者等价地，

$$g\mathbb{H} = \mathbb{H}g.$$

由于环的定义并不要求元素在乘法上有逆元，所以，这就决定了环中的理想有着与正规子群不同的形式。

定义 12.5. 理想

给定环 R ， I 是 R 的子环，如果对任意的 $r \in R$ 有 $rI \subset I$ 和 $Ir \subset I$ ，则称 I 是 R 的理想。



从表面上看，所谓环 R 的理想 I ，首先它是环 R 的子环，其次它具有“吸收性”，即对任意的环元素 $r \in R$ ，无论它是否落在 I 中，用 r 左乘或者右乘 I ，所得到的元素都会落回到 I 中。换一种形式说，对任意的 $r \in R$ 和任意的 $i \in I$ ，必然有 $ri \in I$ 和 $ir \in I$ 。在此，需要读者去体会理想的“吸收性”与正规子群的“左陪集与右陪集相等”的异同之处。

例 12.5

1. 所有的环 R 都有两个平凡理想： $\{0\}$ 和 R 。
2. 如果 R 的理想 I 中包括 1 ，则 $R = I$ 。
3. 对任意整数 $n \in \mathbb{Z}$ ，集合 $n\mathbb{Z}$ 是环 \mathbb{Z} 的理想。直观上看，集合 $n\mathbb{Z}$ 包含了所有 n 的倍数， $n\mathbb{Z}$ 在加法上成群，而用任意整数乘 n 的倍数还是得到一个 n 的倍数，虽然此时 $n\mathbb{Z}$ 中并不必然有单位元 1 ，也不必然有乘法逆元。

命题 12.5. 主理想

设 R 是一个交换环且有单位元, 任取 $a \in R$, 则集合

$$\langle a \rangle = \{ar : r \in R\}$$

是环 R 的一个理想, 称之为主理想 (Principal Ideal)。



证明 首先, 验证 $\langle a \rangle$ 是非空集合, 至少包括 0 和 a 两个元素。其次, 验证 $\langle a \rangle$ 在加法上成群。最后, 验证 $\langle a \rangle$ 具有吸收性, 即任取 $s \in R$ 乘上 $\langle a \rangle$ 中任意元素 ar , 必然有

$$s(ar) = a(sr) \in \langle a \rangle$$

注意, 上式成立需要依赖交换律。所以, $\langle a \rangle$ 是 R 的理想。

例 12.6 对任意整数 n , 集合 $n\mathbb{Z}$ 是整数环 \mathbb{Z} 的理想, 也是主理想, $n\mathbb{Z} = \langle n \rangle$ 。

命题 12.6

整数环 \mathbb{Z} 的所有理想都是主理想。



证明 首先, 零理想也是主理想, 因为 $\langle 0 \rangle = \{0\}$ 。设 I 是整数环 \mathbb{Z} 的一个非零理想, 则 I 中必然包括某些正整数, 根据良序原则 (公理 1.1), 则 I 中必然存在一个最小正整数 n 。对任意的元素 $a \in I$, 根据除法算法, 存在整数 q 和 r , $0 \leq r < n$, 使得:

$$a = qn + r$$

也就是, $r = a - qn$, 利用理想的属性, 可知 $r \in I$ 。又因为 n 是 I 中最小正整数, 所以, $r = 0$ 。因此, $a = nq$, 即 $I = \langle n \rangle$ 。

环论中的主理想对应于群论中的循环群, 循环群 $\langle g \rangle$ 是包含 g 的最小群, 而理想 $\langle a \rangle$ 则是包含 a 的最小理想。命题 7.3 告诉我们, 循环群 $\mathbb{G} = \langle g \rangle$ 的每一个子群都是循环群, 但是, 针对主理想, 就没有类似的结论了。即如果 I 是 R 的主理想, I' 是 I 的子集, 且是 R 的理想, 则 I' 不一定是主理想。

命题 12.7

环同态 $\phi: R \rightarrow R'$ 的 Kernel 是 R 的理想。



证明 根据群论的结论, $K = \text{Ker } \phi$ 是 R 的加法子群 (并且是正规子群), 只需要证明 K 具有理想的“吸收性”, 即对任意的 $r \in R$ 和 $a \in K$ 有 $ar \in K$ 和 $ra \in K$ 。显然如此, 因为:

$$\phi(ar) = \phi(a)\phi(r) = 0\phi(r) = 0$$

且

$$\phi(ra) = \phi(r)\phi(a) = \phi(r)0 = 0$$

例 12.7 对任意整数 $n \in \mathbb{Z}$, 定义映射 $\phi: \mathbb{Z} \rightarrow \mathbb{Z}_n$ 为 $\forall a \in \mathbb{Z}, \phi(a) = a \bmod n$ 。容易验证这是一个环同态, 而 $\text{Ker } \phi$ 就是 $n\mathbb{Z}$ 。

讲完理想, 是时候开始讨论商环了。在证明商环定理之前, 让我们先回顾、理解、熟悉相关的思路。首先, 理解商环必须从商群出发。因为一个环 R , 本身在加法上是阿贝

尔群, 而其理想 I 则是 R 的正规加法子群, 因此 R/I 在加法上就是一个商群, 其中元素就是加法上 I 的陪集。比如, 任取 $r \in R$, $r + I$ 就是 R/I 中的群元素。至此, 我们没有引入任何新知识, 已经对商环的元素和结构有了大致的了解。

接着, R/I 要形成环, 必须定义 R/I 群元素的乘法。无论乘法是什么, 根据环的定义, 该乘法必须是封闭的, 具有结合律和分配律。在给出乘法定义之后, 这些都需要证明。除此之外, 特别强调的是, 既然 R/I 的乘法是对陪集的操作, 千万要记得证明良定义属性, 因为陪集的代表元不唯一。

引理 12.1. 商环乘法

设 R 是环, I 是 R 中的理想。商群 R/I 中元素的乘法定义为: 对任意的群元 $r, s \in R$,

$$(r + I)(s + I) = rs + I$$

该乘法是一种良定义操作, 且具有封闭性、结合律和对加法具有分配律。



要证明乘法是一种良定义操作, 就是要证明乘法独立于陪集代表元的选择。即证明, 如果 $r + I = r' + I$, $s + I = s' + I$, 则 $(r + I)(s + I) = (r' + I)(s' + I)$ 。根据乘法定义, 即要证 $rs + I = r's' + I$ 。根据命题 8.1, 也就是要证明 $r's' \in rs + I$ 。另外, 同样根据命题 8.1, $r + I = r' + I$ 和 $s + I = s' + I$ 分别代表 $r' \in r + I$, $s' \in s + I$ 。这就是证明思路, 具体证明如下。

证明 首先证明乘法是一种良定义操作, 即假设 $r' \in r + I$, $s' \in s + I$, 证明 $r's' \in (rs + I)$ 。因为 $r' \in r + I$, $s' \in s + I$, 即存在 $i_1, i_2 \in I$ 使得 $r' = r + i_1$ 和 $s' = s + i_2$, 因此,

$$r's' = (r + i_1)(s + i_2) = rs + ri_2 + i_1s + i_1i_2$$

根据理想的吸收性, $ri_2 + i_1s + i_1i_2 \in I$, 所以, $r's' \in rs + I$ 。商环乘法的封闭性、结合律和分配律留作课后练习。 \square

定理 12.1. 商环

设 R 是环, I 是 R 中的理想。商群 R/I 在陪集加法与引理 12.1 中定义的乘法上形成环, 称为 R 模 I 的商环, 同样记为 R/I 。



例 12.8 任取 $n \in \mathbb{Z}$, $n\mathbb{Z} = \langle n \rangle$ 是整数环 \mathbb{Z} 的主理想, 则 $\mathbb{Z}/n\mathbb{Z}$ 是商环, 其中元素刚好构成模 n 的完全剩余系。

定义 12.6. 环的标准同态

设 I 是环 R 的理想, 定义环同态映射 $\phi: R \mapsto R/I$ 为 $\phi(r) = r + I$, $\forall r \in R$ 。并称该映射为环的标准同态或者自然同态, 且 $\text{Ker } \phi = I$ 。



定理 12.2. 第一同构定理

设 $\psi: R \mapsto S$ 是环同态, 记 $K = \text{Ker } \psi$ 是 R 的理想。如果 $\phi: R \mapsto R/K$ 是标准同态, 则存在唯一同构 $\eta: R/K \mapsto \psi(R)$ 使得 $\psi = \eta\phi$ 。



证明 根据群论的第一同构定理 (定理 9.3), 在 R 的加法群与 R 模 K 的加法商群之间,

存在唯一的良定义的群同态 $\eta: R/K \mapsto \psi(R)$ 。该映射定义为, 对任意的 $r \in R$, 有

$$\eta(r + K) = \psi(r)$$

要证明 η 是一种环同态, 只需要证明, 对任意的 $r, s \in R$, 有

$$\eta((r + K)(s + K)) = \eta(r + K)\eta(s + K)$$

然而, 这是容易的, 因为

$$\begin{aligned}\eta((r + K)(s + K)) &= \eta(rs + K) \\ &= \psi(rs) \\ &= \psi(r)\psi(s) \\ &= \eta(r + K)\eta(s + K)\end{aligned}$$

□

例 12.9 任取 $n \in \mathbb{Z}$, 构造映射 $\phi: \mathbb{Z} \mapsto \mathbb{Z}_n$ 为, 任取 $a \in \mathbb{Z}$, $\phi(a) = a \bmod n$ 。可验证, 这是一个环同态映射。 $\text{Ker } \phi = n\mathbb{Z}$, 因为所有 n 的倍数都映射为 0。根据第一同构定理, $\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}_n$ 。

12.2 域

定义 12.7. 域

如果一个带单位元的交换环 R 中的非 0 元素都存在唯一的乘法逆元, 即 $\forall a \in R$ 且 $a \neq 0$, 则存在唯一的 $a^{-1} \in R$ 使得 $a * a^{-1} = a^{-1}a = 1$, 则称这种代数结构为域。♣

一般用 F 表示域, 而 F^* 表示域中所有的非 0 元素。由定义可知, 域 F 在加法上是阿贝尔群, 同时 F^* 在乘法上是阿贝尔群。通常可以把域看为特殊的环。但是, 这样说并不公平, 从域的发展史看, 域的提出本身具有其特定的意义与内涵。很少有人去定义有限元素的环, 但是有限元素的域被定义为有限域 (*Finite Field*), 这是非常重要的概念, 得到广泛的重视与研究, 而且往往称有限域为伽罗瓦域 (*Galois Field*), 以此纪念该理论的先驱之一法国数学天才伽罗瓦 (Évariste Galois)。

本书不打算全面展开讨论域的相关理论, 建议读者在此先把域看为环的特例, 从讨论环与域的关系入手, 逐步熟悉域的属性。以此为基础再去学习域的理论则收获更大。

例 12.10

1. 在普通的加法与乘法上, $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ 都是域。 \mathbb{Z} 不是域, 因为乘法上 \mathbb{Z} 不是群。
2. 任意给定素数 p , 在模 p 的加法与乘法上 \mathbb{Z}_p 是域。因为 \mathbb{Z}_p 在加法上是阿贝尔群, 而 \mathbb{Z}_p^* 在乘法上也是阿贝尔群。对任意合数 $n \in \mathbb{Z}$, \mathbb{Z}_n 则不是域, 因为 $\mathbb{Z}_n - \{0\}$ 乘法上不成群。

根据定义, 域 F 中的非 0 元素都存在唯一的乘法逆元, 而这意味着 F 中没有零因子, 表述为以下命题。

命题 12.8. 乘法逆元与零因子

对任意的域 F , 任取 $a, b \in F$, 如果 $ab = 0$ 则 $a = 0$ 或者 $b = 0$ 。即域中不存在零因子。



证明 不妨设 $a \neq 0$, 否则证完。因为 $a \neq 0$, 则存在 a 的乘法逆元 $a^{-1} \in F$ 且 $a^{-1} \neq 0$, 使得 $aa^{-1} = 1$ 。等式 $ab = 0$ 两边乘上 a^{-1} , 根据命题 12.1, 则 $b = 0$ 。

因此, 所有的域都是整环。但是显然并非所有的整环都是域, 比如, \mathbb{Z} 是整环, 但并非域。以下结论则颇有点出人意料。

命题 12.9. 整环与域

每一个有限整环都是域。



证明 证明的思路就是利用有限整环的性质, 为每一个非 0 元素找到乘法逆元。设 D 是一个有限整环, 记 D^* 为环中所有非 0 元素的集合。对任意的 $a \in D^*$, 构造映射 $\lambda_a : D^* \mapsto D^*$ 为 $\lambda_a(d) = ad, \forall d \in D^*$ 。首先, 这确实是一个合理的映射, 因为根据整环的性质, 若 $a \neq 0$ 和 $d \neq 0$, 则 $ad \neq 0$ 。其次, 可证明 λ_a 是单射, 因为对任意的 $d_1, d_2 \in D^*$, 如果:

$$ad_1 = \lambda_a(d_1) = \lambda_a(d_2) = ad_2$$

根据整环的消去律, 则有 $d_1 = d_2$ 。然后, 因为 D^* 是有限集且 λ_a 是从 D^* 到 D^* 的单射, 所以 λ_a 必然是满射。因此, 必然存在某个 $d \in D^*$ 使得 $ad = 1$, 又因为 D 是交换环, 所以这个 d 就是 a 的乘法逆元。结论: 可为 D^* 中每一个元素都找到乘法逆元, 所以 D 是一个域。

定义 12.8. 特征

环 R 的特征 (characteristic) 定义为最小的正整数 n 使得对任意的 $r \in R$, $\underbrace{r + r + \cdots + r}_{n\text{个}} = nr = 0$ 。如果不存在这样的 n , 则 R 的特征定义为 0。

**例 12.11**

1. 环 $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ 的特征都是 0。
2. 对任意素数 p , 域 \mathbb{Z}_p 的特征是 p 。因为 \mathbb{Z}_p 加法群的阶为 p , 即对任意的 $a \in \mathbb{Z}_p$, $pa = 0$ 。

引理 12.2

设 R 为环, 若 1 在加法群的阶为 n , 则 R 的特征为 n 。



证明 若 1 在加法群的阶为 n , 则 n 是最小正整数使得 $n1 = 0$ 。那么, 任取 $r \in R$, 有:

$$nr = n(1r) = (n1)r = 0r = 0$$

即 n 是 R 的特征。

命题 12.10

整环的特征或者为素数，或者为 0。



证明 设整环 D 的特征为 n ，且 $n \neq 0$ 。如果 n 不是素数，则 $n = ab$ ，且 $1 < a, b < n$ 。根据引理 12.2，有

$$0 = n1 = (ab)1 = (a1)(b1)$$

因为 D 是整环， D 中无零因子，所以必然 $a1 = 0$ 或者 $b1 = 0$ 。但是这都意味着 D 的特征小于 n ，矛盾。

因为所有的域都是整环，所以域的特征也必然或者为素数，或者为 0。例 12.11 已经展示了相关域的特征，其中域 \mathbb{Z}_p 的特征与 \mathbb{Z}_p 的阶都是 p 有没有引起你的联想呢？域的特征与阶是什么样的关系呢？以下考虑有限域。

命题 12.11. 有限域的特征

有限域 F 的特征是一个素数。



证明 考虑任意 n 阶有限域 F ，因为 F 是加法群，所以对任意的 $a \in F$ ，有 $na = 0$ 。所以， F 的特征必然是素数 p ，且 $p \mid n$ 。

以下不加证明给出另一个重要结论。

命题 12.12. 有限域的阶

如果有限域 F 的特征是素数 p ，则 F 的阶是 p^n ， n 是某个正整数。进一步，对任意的素数 p 和正整数 n ，存在阶为 p^n 的有限域，并且所有的 p^n 阶有限域都同构。



p^n 阶的有限域也记为 $\text{GF}(p^n)$ ， GF 是 Galois Field 的缩写。这个结论最吸引计算机人的地方在于，因为 2 是素数，所以必然可以构造有 2^n 个元素的有限域。如果说 \mathbb{Z}_n 是计算机系统最自然的代数结构，那么 $\text{GF}(2^n)$ 则是计算机系统的最理想化的代数结构。

例 12.12 Sage 中有可轻松定义 GF 域，请看以下简单例子。虽然简单，但是，有些输出结果应该看不明白。解释这些结果是这接下来的任务，请先建立直观印象。

```

1 \label{exm:GF_field}
2 sage: GF17 = GF(17)
3 sage: GF17.is_field()
4 True
5 sage: GF17.characteristic() #域的特征
6 17
7 sage: GF17.order() #域的阶
8 17
9 sage: GF17_2 = GF(17^2)
10 GF17.is_field()
11 True
12 sage: GF17_2.characteristic()

```



```

13 17
14 sage: GF17_2.order()
15 289
16 sage: GF17.random_element()    #输出一个随机域元素
17 11
18 sage: GF_17_2.random_element()
19 4*z2 + 3

```

是时候讨论域的同态、同构、同构定理、“商域”这些知识点了，然而如果视域为特殊的环，其实可讨论的并不多。因为任何一个从域 F_1 到域 F_2 的域同态都只是从环 F_1 到环 F_2 的环同态。所以，命题12.4、定理12.1和定理12.2等对于域同样成立。特殊之处在于，每一个域同态都是单射，或者是零同态。

在没有证明之前，仅仅知道结论，“域同态是单射”会让读者立即想到什么呢？比如，考虑一个从域 F_1 到域 F_2 的域单射同态 ϕ ， $\text{Ker } \phi$ 会是什么样？单射当然意味着 $\text{Ker } \phi$ 只有一个元素 0。 $\text{Ker } \phi$ 是什么？域的理想！所以，需要讨论任意域 F 的理想所呈现的状态，有以下命题了。

命题 12.13. 域的理想

任何一个域 F 的理想只有 0 和自己本身 F 。



证明 首先，已知 0 和 F 都是 F 的理想。设 I 是域 F 的非 0 理想，则存在非零元素 $a \in I$ 。因为 F 是域，则存在 a 的乘法逆元 $a^{-1} \in F$ 。根据理想的吸收性， $a^{-1}a = 1 \in I$ 。包含 1 的理想 I 等于 F ，即 $I = F$ 。

命题 12.14. 域同态是单射

任何一个域同态或者是单射或者是零同态。



证明 域 F_1 到域 F_2 的域同态 ϕ 是单射，当且仅当 $\text{Ker } \phi = \{0\} \subset F_1$ 。因为 F_1 的理想只有 0 和 F_1 本身，所以当 $\text{Ker } \phi = \{0\}$ 时 ϕ 是单射，而当 $\text{Ker } \phi = F_1$ 时， ϕ 是零同态。

既然域是特殊的环，有商环自然就有商域？但是，考虑到任何一个域 F 的理想只有 0 和自己本身 F 。 F 模 0 就是 F 本身，而 F 模 F 就同构于 0。也就是说，“商域”其实不太值得讨论。值得思考的是，给定一个环 R ，什么样的商环 R/I 会是域？

虽然暂时还没有答案，但是环的第一同构定理已经给出了具体的实例。例12.9告诉我们，任取 $n \in \mathbb{Z}$ ， $\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}_n$ 。任取素数 $p \in \mathbb{N}$ ，立即有 $\mathbb{Z}/p\mathbb{Z} \cong \mathbb{Z}_p$ ，而已知 \mathbb{Z}_p 是域。所以，讨论理想 $p\mathbb{Z}$ 的属性将会帮助我们找到正确答案。

定义 12.9. 极大理想与素理想

设 R 是环， M 是 R 的真子集且是 R 的理想，则称 M 是 R 的真理想。设 M 是 R 的真理想，如果 M 不是 R 的任意真理想的真子集，则称 M 是极大理想 (*maximal ideal*)。即如果 M 是 R 的极大理想，则对 R 的任意理想 I ，若 $M \subset I$ ，则 $I = R$ 。设 P 是交换环 R 的真理想，如果对任意 $ab \in P$ ，则或者 $a \in P$ ，或者 $b \in P$ ，就

称 P 为素理想 (Prime Ideal)。



例 12.13 素理想的“素”确实有“素数”的意味。请回顾第5章的定理5.1：设 p 为素数，如果 $p \mid ab$ ，则 $p \mid a$ 或 $p \mid b$ 。请体会素理想定义中要求与之类似之处：对任意 $ab \in P$ ，则 $a \in P$ 或 $b \in P$ 。令 $P = p\mathbb{Z}$ ， p 是任意素数， $a \in P$ 当且仅当 $p \mid a$ 。所以，从整数的角度上看，素理想确实是素数的倍数形成的理想。当然，理想不能仅停留于此，还需要进一步的抽象，但是这个例子告诉我们，抽象代数的“抽象”并非凭空而出，往往源自于具体的实例。

例 12.14 设 $P = \{0, 2, 4, 6\}$ 为环 \mathbb{Z}_8 的理想，可验证 P 是极大理想，也是素理想。任取素数 p ，则 $p\mathbb{Z}$ 是 \mathbb{Z} 的素理想。

定理 12.3. 极大理想与域

设 R 是交换环， M 是 R 的理想，则 M 是 R 的极大理想，当且仅当 R/M 是域。



例 12.15 任取素数 p ，已知 $p\mathbb{Z}$ 是 \mathbb{Z} 的素理想。因为 $\mathbb{Z}/p\mathbb{Z} \cong \mathbb{Z}_p$ ， \mathbb{Z}_p 是域，所以 $p\mathbb{Z}$ 是 \mathbb{Z} 的极大理想，

定理 12.4. 素理想与整环

设 R 是交换环， P 是 R 的理想，则 P 是 R 的素理想，当且仅当 R/P 是整环。



例 12.16 设 p 是素数，则 $p\mathbb{Z}$ 是 \mathbb{Z} 的理想。可知， $p\mathbb{Z}$ 是 \mathbb{Z} 的极大理想，因为 $\mathbb{Z}/p\mathbb{Z} \cong \mathbb{Z}_p$ 是域。

推论 12.1. 极大理想与素理想

交换环的每一个极大理想都是素理想。



第十二章习题

1. 如果 I 和 J 都是交换环 R 的理想，则 $I + J = \{i + j : i \in I, j \in J\}$ 也是 R 的理想。
2. 证明命题12.4。
3. 任取 $n \in \mathbb{Z}$ ，构造映射 $\phi : \mathbb{Z} \rightarrow \mathbb{Z}_n$ 为，任取 $a \in \mathbb{Z}$ ， $\phi(a) = a \bmod n$ 。请验证 ϕ 是一个环同态映射。
4. 请验证例子12.4的同态实例，并回答，为什么命题12.4要求 ϕ 是满射才会有 $\phi(1) = 1'$ ？
5. 请补充完成引理12.1的证明。

第十三章 多项式与有限域

内容提要

- 多项式环
- 多项式环的 $egcd$ 算法
- 多项式环的 gcd 算法
- GF 域的构造

13.1 多项式

在阅读本书之前,读者应该非常熟悉多项式,因为大多数中国学生从小就熟练掌握多项式的加减乘除。以下内容当然也不外乎是多项式的加减乘除,希望不会有什么难度。

多项式通常表达为如下形式:

$$f(x) = \sum_{i=0}^n a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \quad (13.1)$$

其中,所有的 a_i 称为多项式的系数 (*coefficient*), n 是使得 $a_n \neq 0$ 的最大非负整数,称为多项式的次数,记为 $\deg f(x) = n$,而系数 a_n 就称为首项系数。首项系数为 1 的多项式称为首一多项式 (*monic*)。如果多项式所有的系数都是 0,即多项式 $f(x) = 0$,称为零多项式,其次数定义为 $-\infty$ 。如果多项式的所有系数除了 a_0 之外都是 0,即 $f(x) = a_0$, $a_0 \neq 0$,此多项式称为常数多项式。多项式中的占位符 x 称为不定元 (*indeterminate*),如果习惯于把 x 称为变量的,也许需要纠正。

本节的精彩内容开始于给系数设定范围。设 R 为交换环, x 为不定元,限定多项式的所有系数 $a_i \in R, \forall i \in [0, n]$,形如等式 13.1 的多项式 $f(x)$ 称为环 R 上的多项式,记所有多项式的集合为 $R[x]$ 。为什么说 x 不是变量? 其中一个理由就是, x 的值域并没有任何限定,甚至可以说,对其还一无所知。

然后,就可以定义 $R[x]$ 中多项式的加法与乘法,尽管在普通意义上说,这两个操作都是大家熟知的。

定义 13.1. 多项式的加法与乘法

设 R 为交换环, $a(x), b(x) \in R[x]$, 且

$$a(x) = \sum_{i=0}^n a_i x^i, \quad b(x) = \sum_{i=0}^m b_i x^i, \quad n \geq m.$$

加法定义为:

$$a(x) + b(x) = \sum_{i=0}^n (a_i + b_i) x^i$$

其中,当 $i > m$ 时, b_i 视为 0。乘法定义为:

$$a(x) \cdot b(x) = \sum_{i=0}^{n+m} c_i x^i$$

其中, 对所有的 i , $c_i = \sum_{k=0}^i a_k b_{i-k}$ 。



例 13.1 无论以上定义你看了感觉有多陌生, 本质上就是中学常用的操作而已。以下实例展示在 Sage 当中, 如何定义多项式, 并进行多项式运算。

```

1 sage: R.<x> = PolynomialRing(ZZ) #定义R[x], 其中x是不定元, ZZ是整数环
2 sage: f = R.random_element(); g = R.random_element() # 初始化两个多项式
3 sage: f = 7*x^3 - 4*x^2 + 4*x + 2; g = x^2 - 2*x + 1 # 定义两个多项式f和g
4 sage: f + g
5 7*x^3 - 3*x^2 + 2*x + 3
6 sage: f * g
7 7*x^5 - 18*x^4 + 19*x^3 - 10*x^2 + 2
8 sage: f - f
9 0
10 sage: f / g
11 .....

```

命题 13.1. 多项式环

设 R 为交换环, x 为不定元, 环 R 上的多项式 $R[x]$ 在加法、乘法下形成交换环, 并称之为多项式环。



命题13.1的证明留给读者。多项式在加法上成加法群是显然的, 单位元是 0, $f(x)$ 的逆元是 $-f(x)$ 。乘法有封闭性也显然。重点需要验证的是结合律和分配律, 这是两种我们一直都在用却一直没有证明的定律。

例 13.2

1. $\mathbb{Z}[x]$ 和 $\mathbb{Q}[x]$ 分别是整数上的多项式环和有理数上的多项式环。
2. 设 p 为素数, $\mathbb{Z}_p[x]$ 是域 \mathbb{Z}_p 上的多项式环。
3. $\mathbb{Z}_2[x]$ 是域 \mathbb{Z}_2 上的多项式环。如果大家没有忘记二进制的位置计数法的话, 应该知道 $\mathbb{Z}_2[x]$ 中每一个多项式都代表了一个二进制数字。

以下, 延续第一章讨论整数的加减乘除的思路, 讨论多项式的各种操作属性。首先

考虑的是除法。先看以下例子，动手计算找回对除法计算的回忆。

例 13.3

$$\begin{array}{r}
 3x^2 + 4x + 5 \\
 x-1 \overline{) 3x^3 + x^2 + x - 1} \\
 \underline{- 3x^3 + 3x^2} \\
 4x^2 + x \\
 \underline{- 4x^2 + 4x} \\
 5x - 1 \\
 \underline{- 5x + 5} \\
 4
 \end{array}$$

例 13.4

$$\begin{array}{r}
 \frac{3}{2}x^2 + \frac{5}{4}x + \frac{9}{8} \\
 2x-1 \overline{) 3x^3 + x^2 + x - 1} \\
 \underline{- 3x^3 + \frac{3}{2}x^2} \\
 \frac{5}{2}x^2 + x \\
 \underline{- \frac{5}{2}x^2 + \frac{5}{4}x} \\
 \frac{9}{4}x - 1 \\
 \underline{- \frac{9}{4}x + \frac{9}{8}} \\
 \frac{1}{8}
 \end{array}$$

必须强调，多项式环 $R[x]$ 上的加法与乘法严重依赖其基础环 R 上的加法与乘法。考虑到 $R[x]$ 上的除法，问题来了，环 R 上元素的不一定能做“除法”，因为环没有要求每一个元素都有乘法逆元。比如，例13.4中的所有“分数”在 $\mathbb{Z}[x]$ 中是不一定存在的，但是 $\mathbb{Q}[x]$ 中则没有问题。不难想到，域可以满足除法要求。所以要求能做除法的多项式环都建立在域的基础上。于是，有以下定理为多项式理论奠定基础。

定理 13.1. 多项式的除法算法

设 \mathbb{F} 为域， $a(x), b(x) \in \mathbb{F}[x]$ ，且 $b(x)$ 不为零多项式。那么，存在唯一的多项式 $q(x), r(x) \in \mathbb{F}[x]$ 使得，

$$a(x) = q(x)b(x) + r(x)$$

其中， $\deg r(x) < \deg b(x)$ 或者 $r(x)$ 为零多项式。



证明 使用归纳法证明。

1. 归纳起始步，如果 $a(x)$ 是零多项式，则

$$0 = 0b(x) + 0$$

因此， $q(x)$ 和 $r(x)$ 都是零多项式。

2. 归纳假设，假设次数小于 n 的多项式 $a(x)$ 都满足定理要求。
3. 归纳步，假设 $a(x)$ 为非零多项式，且 $\deg a(x) = n$ ， $\deg b(x) = m$ 。如果 $m > n$ ，则令 $q(x) = 0$ ， $r(x) = a(x)$ ，证完。所以，可以假设 $m \leq n$ ，对 $a(x)$ 的次数 n 进

行归纳。现在,

$$a(x) = \sum_{i=0}^n a_i x^i, \quad b(x) = \sum_{i=0}^m b_i x^i, \quad n \geq m.$$

用 $a(x)$ 除 $b(x)$, 除法第一步的结果是多项式

$$a'(x) = a(x) - \frac{a_n}{b_m} x^{n-m} b(x)$$

并且, $a'(x)$ 的次数小于 n 或者为 0。根据归纳假设, 存在多项式 $q'(x)$ 和 $r(x)$ 使得

$$a'(x) = q'(x)b(x) + r(x)$$

且 $r(x)$ 的次数小于 $b(x)$ 的次数或者为 0。令

$$q(x) = q'(x) + \frac{a_n}{b_m} x^{n-m}$$

则 $a(x) = q(x)b(x) + r(x)$ 。唯一性的证明留给读者, 提示: 用反证法。

多项式的除法算法对于多项式理论的重要性与基础性类似除法算法 (定理1.1) 对整数理论的重要性与基础性。这两个除法算法的证明都非常具有代表性, 值得大家学习。其中, 定理13.1的证明使用了结构归纳法。虽然与大家熟知的算术归纳法一样, 结构归纳法也分为归纳起始、归纳假设和归纳三个步骤。不同的是, 结构归纳针对的不是某个等式中的整数值 n , 而是针对代数结构进行归纳。就多项式除法算法的证明而言, 归纳针对的是多项式的次数 n 。这种方法非常值得大家关注。

至于实现多项式除法的算法则比较简单, 请看以下伪代码, 这也是本书中极少数不能运行的代码之一。整体结构上看, 这个算法就是把中学的长除法代码化而已, 没什么特殊的, 也不需要更多的解释。只有两点需要提醒, 代码中要用到域的“除法”和多项式的加法, 所以, 要把代码运行起来还需要一些细化工作, 这项细化工作留给读者。

```

1 #多项式除法
2 #Input: f and g != 0 are two polynomials over field with indeterminate x
   ;
3 #Output: q, the quotient, and r, the remainder;
4 def poly_div(f, g):
5     q = 0
6     r = f
7     d = deg(g) #g的阶
8     c = lc(g)  #lc返回g的首项系数
9     while deg(r) >= d:
10         temp_c = lc(r) / c #注意, 此为域的除法
11         temp_pow = deg(r) - d
12         s = temp_c*x^temp_pow #构成多项式的某一阶的项
13         q = q + s #多项式加法
14         r = r - s*g #多项式减法
15     return (q, r)

```

算法 13.1: 多项式除法

有了除法, 则有以下类似整数下的定义。设 \mathbb{F} 为域, $a(x), b(x) \in \mathbb{F}[x]$, 如果存在 $q(x) \in \mathbb{F}[x]$ 使得 $a(x) = q(x)b(x)$, 则称 $a(x)$ 可被 $b(x)$ 整除, 记为 $b(x) \mid a(x)$, 此时也称 $b(x)$ 是 $a(x)$ 的一个因子。如果多项式 $d(x) \in \mathbb{F}[x]$ 满足 $d(x) \mid a(x)$ 且 $d(x) \mid b(x)$, 则称 $d(x)$ 为 $a(x)$ 和 $b(x)$ 的公因子。如果首一多项式 $d(x)$ 是 $a(x)$ 和 $b(x)$ 的公因子, 且对 $a(x)$ 和 $b(x)$ 的其他公因子 $d'(x)$ 都有 $d'(x) \mid d(x)$, 则称 $d(x)$ 是 $a(x)$ 和 $b(x)$ 的最大公因子, 记为 $d(x) = \gcd(a(x), b(x))$ 。如果 $\gcd(a(x), b(x)) = 1$, 则称 $a(x)$ 和 $b(x)$ 互素。如果非常数多项式 $a(x) \in \mathbb{F}[x]$ 不能表达为任意两个非常数多项式 $b(x), c(x) \in \mathbb{F}[x]$ 的乘积, 且 $b(x)$ 和 $c(x)$ 的次数都比 $a(x)$ 的次数要小, 则称 $a(x)$ 为不可约多项式 (*irreducible polynomial*)。不可约多项式类似整数中的素数, 它是多项式环中的素多项式。

类似整数的整除性 (命题 1.1), 多项式环中也有关于整除性的简单定理。

命题 13.2. 整除性--多项式环版本

设 F 为域, $a(x), b(x), c(x) \in F[x]$, 则有以下结论:

1. 如果 $a(x) \mid b(x)$, $b(x) \mid c(x)$, 则 $a(x) \mid c(x)$ 。
2. 如果 $c(x) \mid a(x)$, $c(x) \mid b(x)$, 则对任意 $m(x), n(x) \in F[x]$, 有 $c(x) \mid (m(x)a(x) + n(x)b(x))$ 。



证明

1. 因为 $a(x) \mid b(x)$, $b(x) \mid c(x)$, 则存在 $u(x), v(x) \in F[x]$ 使得 $b(x) = u(x)a(x)$ 和 $c(x) = v(x)b(x)$, 所以 $c(x) = v(x)(u(x)a(x)) = (v(x)u(x))a(x)$, 即 $a(x) \mid c(x)$ 。
2. 因为 $c(x) \mid a(x)$, $c(x) \mid b(x)$, 则存在 $u(x), v(x) \in F[x]$ 使得 $a(x) = u(x)c(x)$, $b(x) = v(x)c(x)$ 。对任意 $m(x), n(x) \in F[x]$, 有

$$(m(x)a(x) + n(x)b(x)) = m(x)u(x)c(x) + n(x)v(x)c(x) = (m(x)u(x) + n(x)v(x))c(x)$$

所以, $c(x) \mid (m(x)a(x) + n(x)b(x))$ 。

命题 13.2 虽然简单, 但是它确保了多项式环中欧几里德算法 (\gcd 算法) 的成立。此时就必须重新提及第二章中 \gcd 算法正确性的证明了。首先, 第一种证明在此不再有效, 因为简单验算可知:

$$\gcd(a(x), b(x)) = \gcd(a(x), (a(x) - b(x)))$$

不再必然成立了。其次, 好消息是第二种证明只依赖命题 1.1, 而现在多项式环中对应的命题 13.2 成立, 多项式环中 \gcd 算法就应该成立。实际情况确实如此, 以下给出定理。证明留给读者, 请参考第二章的证明进行完成。

定理 13.2. 欧几里德算法--多项式环版本

设 \mathbb{F} 是域, 给定两个多项式 $a, b \in \mathbb{F}[x]$, 设 $\deg a(x) \geq \deg b(x)$, 则 $a(x)$ 和 $b(x)$ 的最大公因子等于 $b(x)$ 和 $a(x) \bmod b(x)$ 的最大公因子。即

$$\gcd(a(x), b(x)) = \gcd(b(x), a(x) \bmod b(x))$$

其中, $a(x) \bmod b(x)$ 表示用 $a(x)$ 除以 $b(x)$ 所得到的余数 $r(x)$ 。



例 13.5 计算有理数域 \mathbb{Q} 上多项式 $a(x) = x^5 + x^4 + x + 1$ 和 $b(x) = x^4 + x^3 + x + 1$ 的最大公因子。

$$\begin{aligned} \gcd(a(x), b(x)) &= \gcd(b(x), -x^2 + 1) \\ &= \gcd(-x^2 + 1, 2x + 2) \\ &= \gcd(2x + 2, x + 1) \\ &= \gcd(x + 1, 0) = x + 1 \end{aligned}$$

颇令人欣喜的是, 以下代码在 SageMath 下竟然可以求两个多项式的最大公因子, 而代码与求整数的最大公因子一模一样。这当然归功于 SageMath 的 % 操作对多项式同样有效。

```
1 #Input: 多项式f和g
2 #Output: f和g的最大公因子
3 def poly_gcd(f, g):
4     while g != 0:
5         r = f % g #求f除g得到的余数
6         f = g
7         g = r
8     return f
```

算法 13.2: 欧几里德算法--多项式环版本

然而, 算法 poly_gcd 并没有正确完成任务! 因为多项式的公因子要求是首一多项式。请读者自行完成这个简单任务。进而要问, 多项式环中 \gcd 成立, 难道 egcd 算法会不成立吗? 当然成立, 以下给出结论, 证明与算法实现留给读者完成。

定理 13.3. 扩展欧几里德算法--多项式环版本

设 F 是域, 设 $d(x)$ 是两个多项式 $a, b \in \mathbb{F}[x]$ 的最大公因子, 则存在 $r(x), s(x) \in \mathbb{F}[x]$ 使得:

$$d(x) = r(x)a(x) + s(x)b(x).$$



命题 13.3

设 \mathbb{F} 为域, $p(x) \in \mathbb{F}[x]$ 是不可约多项式, 则对任意 $f(x) \in \mathbb{F}[x]$ 且 $p(x) \nmid f(x)$, 有:

$$\gcd(p(x), f(x)) = 1$$



证明 设 $d(x) = \gcd(p(x), f(x))$, 则 d 或者是一个非零常量, 或者是一个 $\mathbb{F}[x]$ 中非零多项式。如果是前者, 则 $d(x)$ 只能是 1, 因为最大公因子必须是首一多项式。如果是后者,

因为 $p(x)$ 是不可约多项式, 且 $d(x) \mid p(x)$, 则存在某个常量 $a \in \mathbb{F}$, 使得 $d(x) = ap(x)$ 。但是, 同时 $d(x) \mid f(x)$, 则存在某个 $g(x) \in \mathbb{F}[x]$, 使得 $f(x) = d(x)g(x) = ap(x)g(x)$, 说明 $p(x) \mid f(x)$, 与条件假设矛盾。□

命题13.3说明, 不可约多项式确实与整数中的素数类似, 环 $F[x]$ 中的多项式在模某个不可约多项式的前提下存在乘法逆元, egcd 算法实现了这种求解。

13.2 有限域

准确来说, 本节并不打算深入讨论有限域的理论。而是通过构造出一种特殊的有限域, 让读者先建立起对有限域的认识。实在是入门之入门。

域 \mathbb{F} 是一种特殊的环, 它在加法上是阿贝尔群, \mathbb{F}^* 在乘法上是阿贝尔群。有限域则是具有有限个元素的域。先归纳以上章节中关于有限域的几点结论:

1. 有限域的特征必为素数。
2. 如果有限域 \mathbb{F} 的特征是素数 p , 则 \mathbb{F} 的阶是 p^n , n 是某个正整数。
3. 对任意的素数 p 和正整数 n , 存在阶为 p^n 的有限域。
4. R 是交换环, R/M 是域当且仅当 M 是 R 的极大理想。

目前大家最熟悉的有限域是 $\text{GF}(p)$, 又称为素域 (*prime field*), 也就是 \mathbb{Z}_p , 其中 p 是任意的素数。而关于 \mathbb{Z}_p 则暂时再没有太多值得探讨之处。对 \mathbb{Z}_p 不满意之处在于, p 是素数, 它不可能等于 2^n , n 为某个正整数。计算机系统对 2^n 情有独钟, 这也是大家熟知的事实。诱人之处在于, 以上结论说, 一定会存在阶为 2^n 的有限域, 且特征为 2。不要忘记 2 是素数。有意思的是, 上面结论最后一点似乎已经提示了构造的方法。

虽然目前我们基本上具备了足够的知识可以从一个环出发, 构造其极大理想 (或者素理想), 从而构造商环得到一个域。然而以下的构造方法则比较“低级”: 按有限域的定义去构造。其出发点当然还是一个环, 但是暂时不提理想, 而是构造环中元素的乘法逆元。其实, 两种方法原理一样, 说法不同而已, 只是以下做法可以暂时摆脱定理的纠缠。

构造的基本思路如下。首先, 计算机系统关注 2^n 阶的域, 其实是关注 n 比特的某个代数结构。 $\mathbb{Z}_2[x]$ 是构造的出发点, 它是一种交换环, 它里面的元素是多项式, 多项式的系数落在 \mathbb{Z}_2 , 也就是说每一个多项式都唯一代表了一个二进制数值, 这也就是所谓的位置计数法。因为要得到一个 n 比特代数结构, 立即想到的是对 $\mathbb{Z}_2[x]$ 取模, 即用一个次数为 n 的多项式 $p(x)$ 去除 $\mathbb{Z}_2[x]$ 中所有的元素, 取其余数, 于是根据除法算法, 得到一系列次数小于 $n-1$ 的多项式, 这个集合记为 S 。即定义映射 $\phi: \mathbb{Z}_2[x] \mapsto S$ 为:

$$\phi(f(x)) = (f(x) \bmod p(x)), \forall f(x) \in \mathbb{Z}_2[x]$$

其中 $\deg p(x) = n$ 。

第一个问题是, 映射 ϕ 是否满射到所有次数小于 $n-1$ 的多项式? 即任取 $g(x) \in \mathbb{Z}_2[x]$ 且 $g(x)$ 的次数小于 n , 是否存在 $f(x) \in \mathbb{Z}_2[x]$ 使得 $\phi(f(x)) = g(x)$?

设定作为模数的多项式 $p(x)$ 为次数 n 的不可约多项式, 则所有问题迎刃而解, 这就是构造的关键点。因为, 如果 $p(x)$ 是不可约多项式, $g(x)$ 的次数小于 n , 则 $p(x) \nmid g(x)$ 。

根据命题13.3, 存在 $r(x), s(x) \in \mathbb{Z}_2[x]$ 使得:

$$r(x)g(x) + s(x)p(x) = 1$$

令 $f(x) = r(x)g(x)g(x)$, 则 $\phi(f(x)) = (r(x)g(x)g(x) \bmod p(x)) = g(x)$ 。

集合 S 包括了所有的次数小于 $n-1$ 的多项式, 即有 2^n 个元素, 每一个元素唯一代表一个二进制数值。现在定义集合 S 上的加法和乘法如下。

将加法定义为 $\mathbb{Z}_2[x]$ 上的多项式加法。需要指出的是, 因为系数落在 \mathbb{Z}_2 上, 所有项对位相加, 其实等价于所有系数按比特异或。此加法显然封闭, 单位元为 0, 所有元素有逆元。即 S 在加法上成群。

将乘法定义为 $\mathbb{Z}_2[x]$ 上多项式的模 $p(x)$ 的乘法, 即对任意多项式 $a(x), b(x) \in S$, 令 $a(x)b(x) = (a(x)b(x) \bmod p(x))$ 。容易验证此乘法封闭, 单位元为 1。当然, 每一个非零多项式都有逆元, 而且满足分配律, 这留给读者证明。即 S 在乘法上成群。

因此, S 在所定义的加法、乘法上是一个域。这个域有一个特殊的名字, 就是 $\text{GF}(2^n)$ 。因为它与二进制密切相关, 也称之为二进制域, 该域的特征为 2。

例 13.6 有限域 $\text{GF}(2^4)$ 由 16 个次数小于 4 的多项式组成, 或者想象为所有 4 比特的二进制数。令不可约多项式 $p(x) = x^4 + x + 1$ 。以下列举 $\text{GF}(2^4)$ 上若干域运算实例。

1. 加法: $(x^3 + x^2 + 1) + (x^2 + x + 1) = x^3 + x$, 等同于 $1101 \oplus 0111 = 1010$ 。
2. 减法: $(x^3 + x^2 + 1) - (x^2 + x + 1) = x^3 + x$, 等同于加法!
3. 乘法: $(x^3 + x^2 + 1) \cdot (x^2 + x + 1) = x^5 + x + 1$, 因为:

$$(x^3 + x^2 + 1) \cdot (x^2 + x + 1) = x^5 + x + 1$$

并且

$$(x^5 + x + 1) \bmod (x^4 + x + 1) = x^2 + 1$$

4. 求乘法逆元: $(x^3 + x^2 + 1)^{-1} = x^2$, 因为 $(x^3 + x^2 + 1) \cdot x^2 \bmod (x^4 + x + 1) = 1$

例 13.7 Sage 提供 $\text{GF}(p^n)$ 域的定义与操作, 比如:

```

1 sage: p = 137
2 sage: F1 = GF(p)    # 定义有限域
3 sage: a = F1.random_element(); a #取一个随机元素
4 51
5 sage: F2 = GF(2^4) #定义二进制域
6 sage: b = F2.random_element(); b #取一个随机元素
7 z4^2 + 1
8 sage: c = F2.random_element(); c #取一个随机元素
9 z4^3 + z4^2 + z4
10 sage: b + c
11 z4^3 + z4 + 1
12 sage: b*c
13 z4 + 1
14 sage:

```

15

例 13.8 计算有限域 $\text{GF}(2^8)$ 上多项式 $a(x) = x^5 + x^4 + x + 1$ 和 $b(x) = x^4 + x^3 + x + 1$ 的最大公因子。

$$\begin{aligned} \gcd(a(x), b(x)) &= \gcd(b(x), x^2 + 1) \\ &= \gcd(x^2 + 1, 0) \\ &= x^2 + 1 \end{aligned}$$

请对比例13.5的结果，思考为什么结果不同。

第十三章 习题

1. 证明命题13.1。
2. 将算法13.1编程实现为一个可真正进行多项式除法的程序。
3. 证明定理13.2。
4. 改进算法poly_gcd，使之输出为首一多项式。
5. 证明定理13.3。
6. 编程实现多项式环中的 egcd 算法。
7. 证明所定义的 $\text{GF}(2^n)$ 的所有非零元素都有乘法逆元。
8. 证明所定义的 $\text{GF}(2^n)$ 满足分配律。
9. 分析对比例13.5与例13.8，为什么相同的多项式在不同的域上，最大公因子不同？提示：把 $a(x)$ 和 $b(x)$ 进行多项式分解，然后发现某些因子多项式在 \mathbb{Q} 上是素多项式，而在 $\text{GF}(2^8)$ 上却不是素多项式。
10. 编程实现 $\text{GF}(2^n)$ 的乘法。
11. 编程实现求 $\text{GF}(2^n)$ 中非零元素的乘法逆元。

第十四章 椭圆曲线

内容提要

□ 椭圆曲线

□ 椭圆曲线上的点乘算法

□ 椭圆曲线上的加法

14.1 初识椭圆曲线

简单而言，所谓椭圆曲线就是满足以下方程的解，即序对 (x, y) 形成的集合：

$$y^2 = x^3 + ax + b$$

其中，要求 $\Delta = 4a^3 + 27b^2 \neq 0$ ，请大家先接受并记得这个常量。如果 a 和 b 是实数，则 x 和 y 自然是实数。先看这样的点形成什么样的曲线，以建立第一印象。使用 SageMath 做以下工作，令 $a = -5$ 和 $b = 4$ ，得到的椭圆曲线如图 14.1 所示。

```
1 sage: a = -5; b = 4
2 sage: E1 = EllipticCurve(RR, [a, b]) #定义实数域上的椭圆曲线
3 sage: show(plot(E1, hue=.9))         #作图
```

如果改变 a 和 b 的值，可以得到其他的曲线，比如令 $a = 2$ 和 $b = 3$ ，得到的椭圆曲线如图 14.2 所示。

```
1 sage: a = 2; b = 3
2 sage: E2 = EllipticCurve(RR, [a, b])
3 sage: show(plot(E2, hue=.9))
```

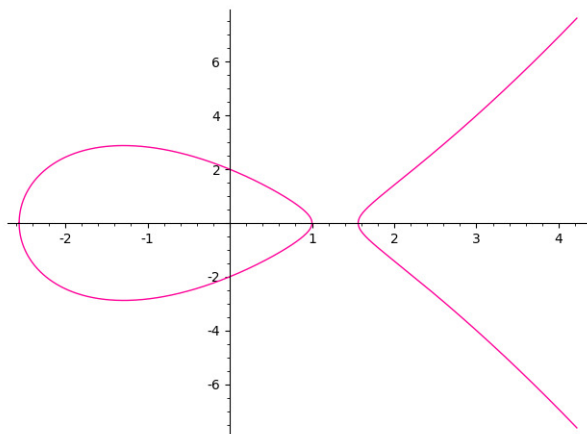
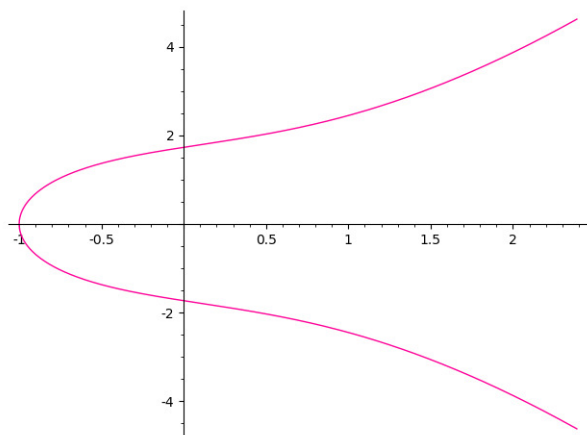


图 14.1: 实数域上的椭圆曲线 $y^2 = x^3 - 5x + 4$

图 14.2: 实数域上的椭圆曲线 $y^2 = x^3 + 2x + 3$

还可以选取随机的 a 和 b , 去看看曲线长什么样。比如我们完全无法预测以下曲线的模样:

```
1 sage: E3 = EllipticCurve(RR, [RR.random_element(), RR.random_element()])
   #RR代表有理数域
2 sage: show(plot(E3, hue=.9))
```

通过观察以上的图, 加上大家也可以利用代码构造不同的图, 然后可以得到对椭圆曲线的第一印象: 长得很不椭圆! 可能有一个要点大家已经注意到: 曲线沿 X 轴对称。

但是, 这些曲线还不是我们最关心的, 我们更关心的是, 如果 a 和 b 落在有限域 \mathbb{F} 上的情形。本节主要考虑的有限域是素域 \mathbb{Z}_p , p 是素数。

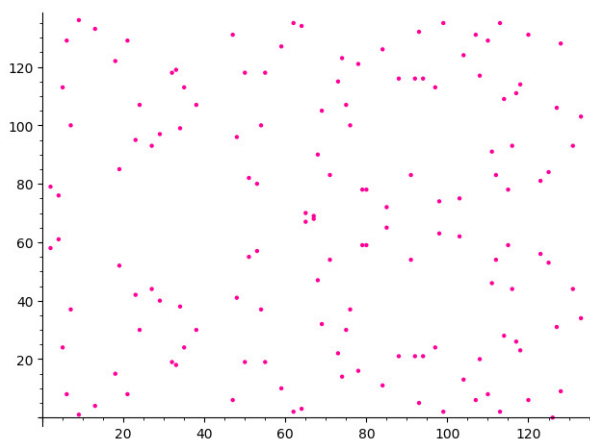
比如, 令 $p = 137$, 定义有限域 $GF(p)$ 。我们运行以下代码, 得到图14.3。

```
1 sage: p = 137 #p是一个素数
2 sage: F = GF(p) #一种有限域, 也可以用 F = FiniteField(p)命令
3 sage: E137 = EllipticCurve(F, [F.random_element(), F.random_element()])
4 sage: show(plot(E137, hue=.9))
```

图14.3所展示的椭圆曲线既不“曲”, 也不成线, 而是离散点的集合。到底有多少个点呢? 这是椭圆曲线研究关心的大问题。对我们而言, 暂时只需要知道曲线大概的点数即可。可以使用以下命令。

```
1 sage: len(E137.points()) # 所有点集合的大小
2 124
3 sage: E137.order() #群的阶
4 124
```

看到这里, 你至少知道椭圆曲线就是一堆“乱七八糟”点的集合, 这有什么用呢? 暂时还没什么用, 直到我们说, 椭圆曲线上的点在加法上成群。在此先给出椭圆曲线的定义, 在此把它写成如下定理。

图 14.3: 有限域 $GF(137)$ 上的椭圆曲线**定理 14.1. 椭圆曲线**

设 \mathbb{F} 为域, $a, b \in \mathbb{F}$ 为常数, 它们使得 $\Delta = 4a^3 + 27b^2 \neq 0$ 。非奇异椭圆曲线 $E(\mathbb{F})$ 是满足方程 $y^2 = x^3 + ax + b$ 上的解集合 $\{(x, y) \in \mathbb{F} \times \mathbb{F}\}$ 加上一个称为无穷远点 (Point at infinity) 的特殊点 \mathcal{O} , 并且 E 在加法上形成一个有限交换群。



所谓无穷远点 \mathcal{O} 就是加法的单位元。这里的内涵比较丰富, 稍后再进一步讨论。当前迫切的任务是要把点的加法定义出来。

注 以上定义还要求椭圆曲线都落在特征不为 2 和 3 的域上。因为, 特征为 2 或者 3 的域会使得以下计算不成立。特征为 2 或者 3 的域上的椭圆曲线要分开独立讨论。

14.2 椭圆曲线群的加法

在定义群加法之前, 先规定一些表示法。 $P, Q \in E$ 表示曲线 E 上的两个点, 注意, 点是一个序对 (x, y) 。所谓群上点的加法就是给定 P 和 Q 求 R , 使得 $R = P + Q$ 。以下定义加法的方法从几何的角度出发, 把所有的椭圆曲线都想象为如图 14.3 或图 14.2 那样的连续的曲线。点的加法分两种情形。

第一种情形考虑两个不同的点 P 和 Q , 有称为点加 (Point addition)。如图 14.4 所示, 给定两个不同的点 P 和 Q , 作 P 和 Q 的连线, 该直线与 E 相交于某个点, 把这个点沿 X 轴翻转就得到了点 R 。令 $R = P + Q$ 。

具体到数值计算, 设 $P = (x_1, y_1)$ 和 $Q = (x_2, y_2)$, $R = (x_3, y_3)$, 则

1. P 与 Q 的连线 L 的斜率 $\lambda = (y_2 - y_1)/(x_2 - x_1)$ 。则直线 L 可记为:

$$y = \lambda(x - x_1) + y_1$$

2. 求 L 与 E 相交的另一个点。将 L 的方程代入椭圆曲线方程, 即用斜率 λ 与 x 表达的 y 代入方程:

$$(\lambda(x - x_1) + y_1)^2 = x^3 + ax + b$$

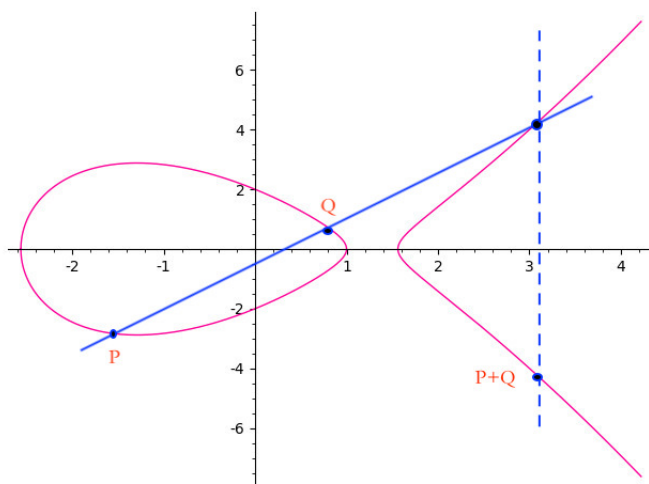


图 14.4: 椭圆曲线加法-1

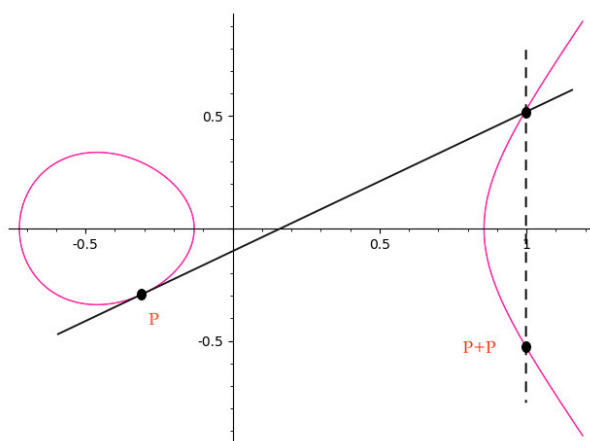


图 14.5: 椭圆曲线加法-2

整理可得

$$0 = x^3 - \lambda^2 x^2 + \dots$$

3. 因为以上方程是一个三次方程，它有三个不同的根，所以：

$$x^3 + \lambda^2 x^2 + \dots = (x - x_1)(x - x_2)(x - x_3) = x^3 - (x_1 + x_2 + x_3)x^2 + \dots = 0$$

4. 可得： $x_3 = \lambda^2 - x_1 - x_2$ ，进而可得 $y_3 = \lambda(x_1 - x_3) - y_1$ 。注意，这里已经做了值的翻转！

注意，难道所有三次方程都一定有三个不同的根，不能是重根吗？好问题，答案是：三次方程的判别式 $\Delta = 4a^3 + 27b^2 \neq 0$ 已经确保了重根不会出现。这个讨论请参考课后习题。

第二种情形，考虑 $P = Q$ 时的加法，又称为倍点 (Point doubling)。如图14.5所示，此时，只给定了一个点 P ，我们要计算 $R = P + P$ 。

首先，沿点 P 作曲线的切线，与 E 相交于某点，然后沿 X 轴翻转该点就得到 R 。具体计算公式如下：

$$1. \lambda = (3x_1^2 + a)/2y_1$$

$$2. x_3 = \lambda^2 - 2x_1, y_3 = \lambda(x_1 - x_3) - y_1$$

以上计算最重要是要考虑 λ 。 λ 就是经过点 P 在曲线 E 上的切线的斜率，根据隐式微分：

$$2y \frac{dy}{dx} = 3x^2 + a$$

所以

$$\lambda = \frac{dy}{dx} = \frac{3x_1^2 + a}{2y_1} =$$

余下推导过程作课后练习。



思考 之前强调过此时域的特征不为 2，请思考，如果此时域的特征为 2， $\frac{3x_1^2+a}{2y_1}$ 会是什么情况？此时需要重新提及第 6 章不大容易引起注意的注 6.2。

讲到这里大家是否会提出这样的异议：没错，如果椭圆曲线是在实数域上 \mathbb{R} ，以上推导基本没什么问题，但是我们知道，如果椭圆曲线落在 F 上，曲线是离散点的集合，那么做连线、切线、求斜率等等，这一系列我们都做不到啊！说得没错，这一系列几何意义确实很难体现。但是，不要忘记，在实数域 \mathbb{R} 上曲线的点加法无非就是用了域上的加法与乘法。所有的域都可以做加法、乘法运算。所以，定义任意域上的椭圆曲线的点加法，只需要把以上定义的所有相关操作“平移”到特定域上即可！

例 14.1 设 $a = 3$, $b = 4$ ，定义有限域 $GF(137)$ 上的椭圆曲线方程： $y^2 = x^3 + 3x + 4b$ 。给定两个点 $P = (x_1, y_1) = (18, 100)$, $Q = (x_2, y_2) = (20, 114)$ ，求 $R = P + Q$ 。

解法如下：

- 求 $\lambda = (y_2 - y_1)/(x_2 - x_1) = (114 - 100)/(20 - 18) = 7$ 。
- $x_3 = \lambda^2 - x_1 - x_2 = 7^2 - 18 - 20 = 11$
- $y_3 = \lambda(x_1 - x_3) - y_1 = 7(18 - 11) - 100 = 86$

注意，所有加减乘除都是域上运算。所以， $R = (x_3, y_3) = (11, 86)$ 。大家可以通过 Sage 进行验证：

```
1 sage: p = 137
2 sage: F = GF(p)
3 sage: E137 = EllipticCurve(F, [3,4])
4 sage: P = E137(18,100)
5 sage: Q = E137(20,114)
6 sage: R = P + Q; R
```

请注意，椭圆曲线群的定义之所以写为定理，是因为其中的群结构属性需要证明。即需要证明椭圆曲线 E 的点在以上加法上满足：封闭性、存在单位元、存在逆元、结合律和交换律。其中，结合律的证明并不显然，而是较为复杂的证明，本书略过。

无穷远点到底是什么？我们知道 \mathcal{O} 是群的单位元，也就是说如果 P 与 Q 互为逆元， $P + Q = \mathcal{O}$ 。而 P 与 Q 互为逆元意味着，它们曲线上是沿 X 轴对称的两个点，那么它们的连线永远是垂直于 X 轴的线。与 X 轴垂直的线与曲线在哪里还能相交出一个点呢？理论上就只能在无穷远的地方。直观上， \mathcal{O} 就代表所有垂直于 X 轴的线在曲线上的交

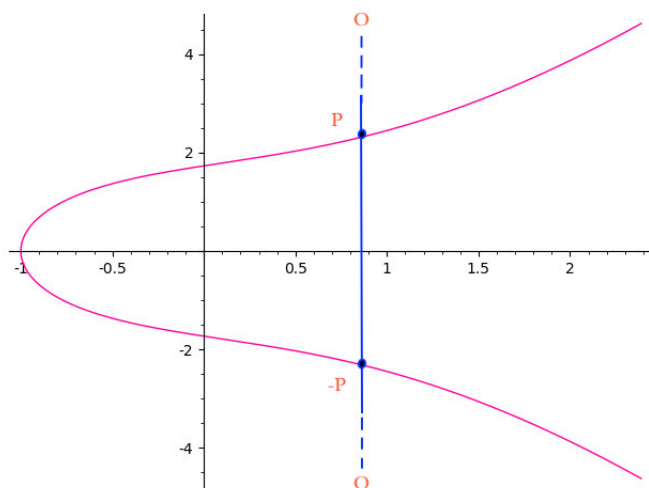


图 14.6: 椭圆曲线群的逆元与无穷远点

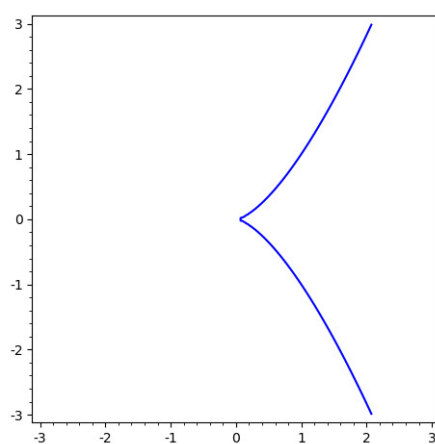


图 14.7: 带奇异点的曲线-1

点。图14.6展示了这种直观。

要求三次方程的判别式 $\Delta = 4a^3 + 27b^2 \neq 0$ 是为了确保不会出现重根，此时椭圆曲线不会出现奇异点 (*singular points*)。要理解所谓奇异点，可先直观地认识当 $\Delta = 0$ 是曲线的形状。比如，通过 SageMath 可以画以下曲线：

```
1 sage: x, y = var('x, y') #定义两变量
2 sage: implicit_plot(x^3 - y^2 == 0, (x,-3,3), (y,-3,3)) # 画曲线y^2 = x
   ^3
3 sage: implicit_plot(x^3 - 3*x + 2 - y^2 == 0, (x,-3,3), (y,-3,3)) # 画
   曲线y^2 = x^3 - 3x + 2
```

图14.7显示的曲线有一个不平滑的折点，而图14.8的曲线则自己与自己相交，产生一个交点，这些点就是所谓的奇异点。存在奇异点的曲线都不符合我们对椭圆曲线的要求。

给定椭圆曲线 $E(\mathbb{F})$ 上的一个点 P ，根据以上计算法则，理论上可以计算出 $E(\mathbb{F})$ 上所有的点。问题是，在最开始如何找到 $E(\mathbb{F})$ 上的一个点？

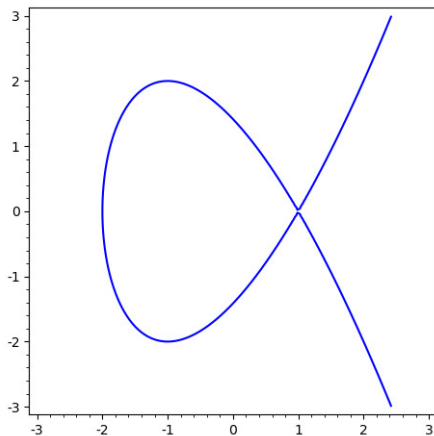


图 14.8: 带奇异点的曲线-2

定理 14.2. Hasse 定理

设 p 为素数, $E(GF(p))$ 是有限域 $GF(p)$ 上的椭圆曲线, 则 $p + 1 - 2\sqrt{p} \leq |E(GF(p))| \leq p + 1 + 2\sqrt{p}$ 。



Hasse 定理暗示说椭圆曲线 $E(GF(p))$ 上有很多点, 那么随机挑选一个序对 (x, y) 都以很大的概率是椭圆曲线上的点。要相信概率! 做法如下: 均匀随机选取 $x \in \mathbb{Z}_p$, 根据椭圆曲线方程计算出 y^2 (注意这里只是表示一个值), 然后判断 y^2 是否模 p 的二次剩余, 如果是, 则求平方根得到 y 。上述过程使用的算法都出现在第11章, 具体实现留作课后练习。

14.3 椭圆曲线的点乘算法与离散对数问题

对任意的 $k \in \mathbb{N}$, 给定 P 为椭圆曲线 $E(F)$ 上的一个点, 计算

$$kP = \underbrace{P + P + \cdots + P}_k$$

即求 k 个 P 相加得到的点, 这种计算称为点乘 (Point multiplication), 或称为标量乘 (scalar multiplication)。

看上去, 这是群元素的指数运算, 实际上, 点乘算法更类似整数的乘法。因为, 把 k 看为二进制数 $k = \sum_{i=0}^n k_i 2^i$, n 是某个整数, $k_i \in \{0, 1\}$ 代表 k 二进制表达的一个比特。则:

$$\begin{aligned} kP &= (k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \cdots + k_12 + k_0)P \\ &= k_{n-1}(2^{n-1}P) + k_{n-2}(2^{n-2}P) + \cdots + k_1(2P) + k_0P \end{aligned}$$

观察可知:

1. 上式的每一项都包含有 $2^i P$, 也就是说必须重复地对 a 进行倍点运算。
 2. 每一项中的 k_i 比特可视为一个标识符, 当 $k_i = 1$ 时, 就把 $2^i P$ 加到最终结果当中。
- 整个过程与第1章的简单整数乘法算法基本相同, 不断进行倍点与加法。所以算法也命名

为 Double-and-Add 算法，具体描述如下：

```

1 # Input: n是一个标量, P是椭圆曲线上的一个点
2 # Output: nP, 即n个P的点加
3 def DoubleAdd(n, P):
4     R = 0
5     Q = P
6     while(n > 0):
7         if is_odd(n):
8             R = R + Q
9             Q = 2 * Q
10            n = n // 2
11    return R

```

算法 14.1: 椭圆曲线的点乘算法

定义 14.1. 椭圆曲线上的离散对数问题

设 $E(F)$ 是有限域 F 上的椭圆曲线，且 $P \in E(F)$ 。假定 Q 是 P 的倍数，则所谓椭圆曲线离散对数问题 (ECDLP) 就是要计算 $n \in \mathbb{F}$ 使得 $nP = Q$ 。



Chapter Note

When placed at the end of a chapter or contribution (as opposed to at the end of the book), the numbering of tables, figures, and equations in the appendix section continues on from that in the main text. Hence please *do not* use the `appendix` command when writing an appendix at the end of your chapter or contribution. If there is only one the appendix is designated “Appendix”, or “Appendix 1”, or “Appendix 2”, etc. if there is more than one.

第十四章习题

1. 证明如果三次方程 $x^3 + ax + b = 0$ 有重根当且仅当 $\Delta = 4a^3 + 27b^2 = 0$ 。提示：如果方程的解分别是 e_1 、 e_2 和 e_3 ，必然有 $x^3 + ax + b = (x - e_1)(x - e_2)(x - e_3) = 0$ ，这样可以得到解与三次方程系数 a 、 b 之间的关系。然后，方程判别式 $\Delta = ((e_1 - e_2)(e_2 - e_3)(e_3 - e_1))^2$ 。计算出 Δ ，并观察到 $\Delta = 0$ 当且仅当方程有重根。
2. 编写程序，对任意的椭圆曲线 $E(GF(p))$ ， p 是一个素数，输入序对 (x, y) ，判断该序对是否是椭圆曲线上的点。

第十五章 大整数分解

内容提要

□ 大整数分解

□ $p-1$ 算法

□ Rho 算法

□

15.1 引言

在之前的许多算法中都需要对给定的整数求解该整数的素数因子，比如，求欧拉 Φ 函数的值、求 \mathbb{Z}_p^* 的生成元等，甚至所有需要使用中国剩余定理的算法都需要使用某个整数的素因子。本章的核心主题就是：给定一个大整数 n ，如果求出 n 的所有素因子？

以下这种被称为“试除法”的平凡方法广泛用于计算机专业的编程入门练习，相信大家都会懂。但是，这是一种效率不高的算法，不能用于大整数的分解。

```
1 int NaiveFactoring(int n){
2     for(int i = 2; i * i <= n; i++){
3         if(n % i == 0) return i;
4     }
5     return 0; //Not found.
6 }
```

算法 15.1: " 整数分解--试除法"

表面上看，这是一个效率是 $O(\sqrt{n})$ 的多项式时间算法，为什么说它不高效呢？其实，这是一种典型的伪多项式时间 (*Pseudo-polynomial time*) 算法。通常，某些算法对特定的输入值并不感兴趣，而重点关注的是某一类数值，比如，大整数分解算法对某个整数 n 并不关心，而关心的是 1024 比特这个规模的所有整数的分解。此时，整数分解算法的输入就不是某个整数 n 而是整数 n 的长度 $l = \log(n)$ ，于是算法的效率就不是 $O(\sqrt{n})$ 而是 $O(2^{\log(n)/2})$ 。如果考虑 1024 比特那么长的整数，算法复杂性就具体化为 $O(2^{512})$ ，很显然的指数式时间。

SageMath 里面实现了若干整数算法，例子如下，也请大家善于利用它们，以后会得上。

例 15.1

```
1 sage: p = random_prime(2**20); q=random_prime(2**20); n = p*q
2 sage: factor(n)
3 628267 * 689713
```

```

4 sage: p = random_prime(2**70); q = random_prime(2**70); n = p*q
5 sage: qsieve(n)
6 ([71778121402821018943, 833225181856404536623], '')
7 sage: p = random_prime(2**80); q = random_prime(2**80); n = p*q
8 sage: time qsieve(n)
9 CPU times: user 3.05 ms, sys: 8.71 ms, total: 11.8 ms
10 Wall time: 2.16 s
11 ([746515468832919223969213, 961436815326002097539293], '')
12 sage: time factor(n)
13 CPU times: user 670 ms, sys: 93 ms, total: 763 ms
14 Wall time: 877 ms
15 746515468832919223969213 * 961436815326002097539293

```

以上计算实例从直观上说明, 个人计算机分解一两百比特的整数还是比较容易的, 当然不能使用上面那个NaiveFactoring算法。不同的算法效率差异很大, 比如, SageMath实现的factor算法要比qsieve慢很多。目前最高效的大整数分解工具是Cado-nfs, 它创造了多个分解RSA模数的记录, 强烈推荐大家下载使用。

目前, 尚不存在高效的大整数分解算法。以下介绍的方法只能让读者稍微窥探大整数分解的奥秘。

15.2 Pollard 的 $p-1$ 法

我们介绍的第一种方法是Pollard的 $p-1$ 法, 本质上这是一种构造法。为方便描述, 考虑一种非常简洁的大整数 $N = pq$, p 和 q 是 N 的两个不相同的素因子。从一个简单的想法出发: 假设可以找到某个整数 m (也许通过交好运) 使得 N 的素因子 p 和 q (都是未知的值) 满足

$$(p-1) \mid m, \text{ 且 } (q-1) \nmid m$$

那么, 选取任意的整数 $x \in \mathbb{Z}_N^*$, 通常取 $x = 2$, 计算可得 y :

$$\begin{aligned}
 y &= ((x^m - 1) \bmod N) \leftrightarrow [(x^m - 1) \bmod p], [(x^m - 1) \bmod q] \\
 &= (0, [(x^m - 1) \bmod q])
 \end{aligned}$$

以上计算使用了CRT和费尔马小定理。所得的 y 大概率会被 p 整除, 但是不被 q 整除。 y 不被 q 整除的理由也很简单, 因为任取 $x \in \mathbb{Z}_n^*$ 都以大概率是一个 \mathbb{Z}_q^* 的生成元。所以 $p \mid ((x^m - 1) \bmod N)$ 且 $p \mid N$ 。那么, 通过使用gcd算法就得到了这个素因子 p , 即计算 $\gcd(y, N)$ 可得 p 。

接下来再讨论以上思路的几个关键点。首先, 怎么样才能找到符合条件的 m ? 如果 $p-1$ 恰好是某些小素数的乘积 (又是一种假设), 则 $p-1$ 会整除某个 $n!$, n 是某个不太大的整数值。所以, 只需要取 $m = n!$ 即可。

但是, 即使 n 不大, $n!$ 也可能是一个巨大的数值。幸运的是, 算法只需要模 N 下

的计算, 实际上参加计算的数值并不超过 N 。并且, 也不需要直接去计算 $(x^{n!} \bmod N)$, 因为

$$x^{(n+1)!} \equiv (x^{n!})^{n+1} \pmod{N}$$

算法的实际操作是, 对 $n = 2, 3, \dots$ 不断迭代计算:

$$\gcd((x^{n!} - 1) \bmod N, N)$$

直到计算出一个不为 1 的返回值。请看算法的具体实现。

```

1 def PollardPM1(N, B=10^5):
2     a = 2 # or some other value;
3     for i in range(2, B):
4         a = a^i % N
5         divisor = gcd(a - 1, N)
6         if divisor != 1 and divisor != N:
7             return divisor
8     return 1

```

算法 15.2: "Pollard 的 $p-1$ 法--版本 1."

例 15.2 令 $a = 2$, $N = 121$ 。迭代做以下计算:

1. $a^2 \bmod N = 4$
2. $a^3 \bmod N = 64$
3. $a^4 \bmod N = 82$
4. $a^5 \bmod N = 56$
5. $\gcd(56 - 1, N) = 11$

得到 11 是 121 的一个素因子。

例 15.3 设 $a = 2$,

1. 如果 $N = 13927189$, 则 $\gcd(a^{14!} - 1, N) = 3823$ 。
2. 如果 $N = 168441398857$, 则 $\gcd(a^{53!} - 1, N) = 350437$

关于 m 的取法, 不同的教科书会有稍微不同的做法, 比如不用阶乘, 而是使用素数的乘积, 那么就需要维护一张素数的列表。在此不再详细讨论, 请看算法 15.3 的具体实现。

```

1 def PollardPM1(N, B=10^5, stop=10):
2     m = prod([p^int(math.log(B)/math.log(p))
3     for p in prime_range(B + 1)])
4     # p^x, where x = ceiling(log_p B)
5     for a in range(2, stop):
6         x = (Mod(a, N)^m - 1).lift()
7         if x == 0: continue

```

```

8     divisor = gcd(x, N)
9     if divisor != 1 and divisor != N: return divisor
10    return 1

```

算法 15.3: "Pollard 的 $p-1$ 法--版本 2."

Pollard 的 $p-1$ 法需要一个非常重要的假设: $p-1$ 恰好是某些小素数的乘积。如果要分解的大整数 N 不配合表演怎么办? 没办法, 只能承认, 分解失败。换言之, $p-1$ 法不必然成功。如果成功, 其效率还是颇高的, 只是 n 次模 N 的指数运算的开销。

15.3 Pollard 的 Rho 算法

还是从最简单的情形出, 发给定一个大整数 $N = pq$, p 和 q 分别是 N 的两个不同的素因子。如果可以幸运地找到两个整数 $x, y \in \mathbb{Z}_N^*$ 使得 $x \equiv y \pmod{p}$ 成立, 则称发生了一次碰撞 (*collision*)。当然, 不可能同时有 $x \equiv y \pmod{q}$ 成立, 否则 x 和 y 将大于 N 。于是使用 \gcd 算法, 计算 $\gcd(x - y, N)$ 则必然得到 N 的某个非平凡的素因子。

如何能求出那两个满足条件的幸运整数 x 和 y 呢? 根据生日悖论, 如果均匀随机选取 $O(\sqrt{p})$ 个 \mathbb{Z}_N^* 元素, 那么以很大的概率必然会有两个数发生碰撞。但是, 如果只是均匀随机选取 $O(\sqrt{p})$ 个 \mathbb{Z}_N^* 元素, 并对元素两两测试判断是否一次碰撞, 则需 $O(p)$ 次计算, 这并不比试除法好到哪里去。

基于以上观察, Pollard 设计了这样一种方法, 使用一个特定的随机函数 $F: \mathbb{Z}_N^* \mapsto \mathbb{Z}_N^*$, 从某个随机值 $x \in \mathbb{Z}_N^*$ 出发, 进行一次龟兔赛跑:

乌龟过程: 这是一次慢计算,

$$x \equiv F(x) \pmod{N}$$

兔子过程: 这是一次快计算, 算法开始前, 令 $x' = x$,

$$x' \equiv F(F(x')) \pmod{N}$$

然后计算并判断:

$$d = \gcd(x - x', N)$$

如果 d 是一个非平凡因子, 则返回 d , 否则继续龟兔赛跑过程。具体算法描述如下:

```

1 def PollardRho(N):
2     x1 = x2 = randint(1, N)
3     while True:
4         x1 = F(x1, N)
5         x2 = F(F(x2, N), N)
6         p = gcd(x1 - x2, N)
7         if p != 1 and p != N:
8             return p

```

算法 15.4: "通用 Pollard 的 Rho 算法."

著名算法教科书《算法导论》使用以下随机函数：

$$F(x) = x^2 - 1 \pmod{N}$$

并且在满足某个特定条件时，改变兔子比乌龟跑得更快的速度，即不总是跑多一步，而是 2^k 步， k 是一个变量。具体过程见算法 15.5。

```

1 def PollardRho(N):
2     i, k = 1, 2
3     x = randint(0, N - 1)
4     y = x
5     while True:
6         i = i + 1
7         x = ((x^2) - 1) % N
8         divisor = gcd(y - x, N)
9         if divisor != 1 and divisor != N:
10            return divisor
11        if i == k:
12            y, k = x, 2*k

```

算法 15.5: "Pollard 的 Rho 算法--算法导论版本."

以下例子用的是通用 Rho 算法，随机函数定义为 $F(x) = x^2 - 1 \pmod{N}$ 。

例 15.4 设 $N = 77$ ，令 $x = 51$ 和 $x' = 51$ 。

- **乌龟过程：** $x = F(x) = 59$;
- **兔子过程：** $x' = F(F(x')) = 15$;
- 计算 $\gcd(x - x', N) = 11$ 。

例 15.5 设 $N = 3013$ ，令 $x = x' = 752$ 。

- **乌龟过程：** $x = F(x)$ ，得到以下数字序列：

2072, 2671, 2469, 661, 35, 1224, 714, 598

- **兔子过程：** $x' = F(F(x'))$ ，得到以下数字序列：

2671, 661, 1224, 598, 2300, 276, 2392, 575

- 最后， $\gcd(598 - 575, N) = 23$ ，总共迭代了 8 次。

Pollard 的 Rho 算法的期望开销是 $\Theta(\sqrt{N})$ ，依然是一个指数式的复杂度。

❧ 第十五章 习题 ❧

1.

2. 设 $N = pq$, p 和 q 是两个不同的素数。如果可以求得一个对 1 模 N 开根号的非平凡解 x , 即 $x^2 \equiv 1 \pmod{N}$, 且 $x \neq \pm 1$ 。证明 $\gcd(x-1, N)$ 和 $\gcd(x+1, N)$ 都是 N 的非平凡因子。如果证明成立, 似乎找到了一个分解 N 的算法, 请问, 这样分解 N 的算法复杂度是多少?
- 3.



第十六章 离散对数

16.1 导引

本章讨论求离散对数的算法，先看定义。

定义 16.1. 离散对数问题

设 \mathbb{G} 为群。给定任意元素 $g \in \mathbb{G}$ （不要求 g 为群 \mathbb{G} 的生成元）和 $y \in \langle g \rangle$ 。所谓离散对数问题（*Discrete Logarithm Problem*，简记为 DLP）即要计算 x 使得 $g^x = y$ 。称解 x 为 y 以 g 为基底的离散对数值。



以上定义不要求 g 为群 \mathbb{G} 的生成元，这使得离散对数问题的定义更具有普遍意义。实际应用中我们也确实需要在某些非循环群中求离散对数，比如第14章的椭圆曲线群。也有教科书或者科技论文要求基底 g 为群的生成元，这个要求并不影响以下算法的计算。

求离散对数最朴素的方法就是穷举法。已知 g 和 y ，不断求 g^i 并判断是否等于 y 。显然，如果群 \mathbb{G} 很大，这是一个不高效的算法。例如，给定一个长为 λ 比特的素数 p ，群 \mathbb{Z}_p^* 的大小是 $p - 1$ ，是 $O(2^\lambda)$ 的规模。穷举法求离散对数则需要 $O(2^\lambda)$ 次乘法，是一个指数式的效率。

我们猜想，对给定 g ，如果 g^i 会满足某种规律，那么也许可以利用这种规律去寻找更好的计算方法。图16.1则显示出 \mathbb{Z}_p^* 群中的 g^i 呈现出一种均匀分布性，至少看上去如此。

16.2 Pohlig-Hellman 算法

Pohlig-Hellman 算法是一种特殊的分治法，把一个 DLP 问题实例分解为若干个小的 DLP 问题，对小问题进行求解，然后通过小问题的解得到目标问题的解。

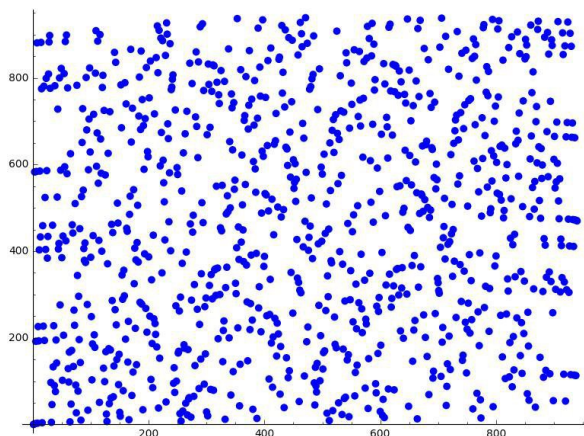


图 16.1: $g = 2, p = 941$ 时 $g^x \pmod{p}$ 的点集.

考虑任意群 \mathbb{G} , 给定基底 $g \in \mathbb{G}$, 且 g 的阶为 $n \in \mathbb{N}$, DLP 实例是 $y = g^x$, 目标是计算出 x 。如果读者感觉把这里的任意群 \mathbb{G} 实例化为乘法群 \mathbb{Z}_p^* 更容易理解, 也可以如此操作。算法分为以下步骤:

1. 首先对 n 进行整数分解, 即求得 $n = p_1^{k_1} p_2^{k_2} \cdots p_t^{k_t}$, 所有的 p_i 都是素数;
2. 令 $g_i = (g^{n/p_i^{k_i}})$, $y_i = y^{n/p_i^{k_i}}$, 得到了 t 个小的 DLP 问题的实例, 因为此时对所有的 $0 \leq i \leq t$, $g_i^x = y_i$ 。
3. 对以上 DLP 实例进行求解, 得到 t 个解 x_i , $0 \leq i \leq t$;
4. 任意一个 $g_i^x = y_i$ 的解 x_i 与目标问题的解 x 满足以下关系:

$$x \equiv x_i \pmod{p_i^{k_i}}$$

因此, 得到 t 个同余方程 $x \equiv x_i \pmod{p_i^{k_i}}$, 借助 CRT 可求出 x 。

具体算法执行过程写成如下代码:

```

1 #Problem: Given base g, y \in <g>, y = g^x mod p, to find x.
2 def Pohlig_Hellman(y, g, p):
3     prime_list = factor(p - 1)
4     n = len(prime_list)
5     q_list = [prime_list[i][0]^prime_list[i][1] for i in range(n)]
6     G = [ g^((p - 1) // q_list[i]) for i in range(n) ]
7     Y = [ y^((p - 1) // q_list[i]) for i in range(n) ]
8     val_list = [discrete_log(Y[i], G[i]) for i in range(n)]
9     modulus_list = [q_list[i] for i in range(n)]
10    x = crt(val_list, modulus_list) # solve the CRT problem.
11    return x

```

算法 16.1: "Pohlig-Hellman 算法."

例 16.1 设 $p = 131$, $g = 2$ 是 \mathbb{Z}_p^* 的生成元, 给定 $y = 80$, 求 x 使得 $y = g^x$ 。首先, 分解 $p - 1 = 2 * 5 * 13$ 。第二步, 因为 g 的阶是 130, 求得以下三个小的 DLP 问题实例:

$$\begin{aligned}
 (g^{130/13})^x &\equiv y^{130/13} \pmod{131} \\
 (g^{130/5})^x &\equiv y^{130/5} \pmod{131} \\
 (g^{130/2})^x &\equiv y^{130/2} \pmod{131}
 \end{aligned}$$

代入 g 和 y 并计算可得:

$$\begin{aligned}
 107^x &\equiv 63 \pmod{131} \\
 53^x &\equiv 1 \pmod{131} \\
 130^x &\equiv 1 \pmod{131}
 \end{aligned}$$

第三步，计算以上 DLP 实例，得到：

$$x_1 \equiv 11 \pmod{13}$$

$$x_2 \equiv 0 \pmod{5}$$

$$x_3 \equiv 0 \pmod{2}$$

最后，对以上同余方程组运用 CRT，求得 $x = 50$ 。

可以使用 SageMath 验证以上计算。

```
1 sage: p = 131
2 sage: g = Mod(2, p) #表示g是模p下的值
3 sage: y = g^50      #因此，所有与g相关的计算都模p
4 sage: discrete_log(y, g, p)
5 50
```

算法的正确性. 要证明算法的正确性，需要证明以下两个性质。

第一点，所构造出的小 DLP 实例确实是以 g_i 为基底、离散对数值是目标问题的离散对数值，即 $g_i^x = y_i$ 成立。因为 $g^x = y$ ，所以对任意 $0 \leq i \leq t$

$$g_i^x = (g^{n/p_i^{k_i}})^x = (g^x)^{n/p_i^{k_i}} = y^{n/p_i^{k_i}} = y_i$$

即 $g_i^x = y_i$ 。

第二点，要证明任意一个 $g_i^x = y_i$ 的解 x_i 与目标问题的解 x 满足关系 $x \equiv x_i \pmod{p_i^{k_i}}$ ，只需要证明任意一个 g_i 的阶是 $p_i^{k_i}$ 。首先要注意到

$$g_i^{p_i^{k_i}} = (g^{n/p_i^{k_i}})^{p_i^{k_i}} = g^n = 1$$

即 g_i 的阶整除 $p_i^{k_i}$ 。假设 g_i 的阶是 $p_i^{k'_i}$ ， $k'_i \leq k_i$ 。则有：

$$g_i^{p_i^{k'_i}} = g^{(n/p_i^{k_i})p_i^{k'_i}} = g^{n/p_i^{k_i-k'_i}} = 1$$

因为 g 的阶是 n ，所以，只能是 $k'_i = k_i$ ，即 g_i 的阶就是 $p_i^{k_i}$ 。

算法的复杂性. Pohlig-Hellman 算法的计算主要包括三部分：分解基底 g 的阶 n ；求解 t 个小的 DLP 问题；计算 CRT 问题。大整数分解是难题，这个问题在第 15 章详细讨论，所以，Pohlig-Hellman 算法通常是假设 n 的素因子已知的情况下进行 DLP 求解。求解 CRT 问题是容易的，请参考第 10 的结论。比较奇怪的是在求解 DLP 问题的时候借助了 DLP 问题的求解算法。其实，这在算法理论当中是常见的手段。读者在此需要体会的是，构造出的所有 DLP 问题实例 $g_i^x = y_i$ 确实是更“小”的问题。这种小体现在 g_i 的阶变小了，从某个 n 变成了某个 $p_i^{k_i}$ 。即使从最肤浅的层面看，阶变小了，穷举都会变容易很多。当然，实际上我们还可以借助下面章节的算法得到更好的效率。

会不会某些 n 的素因子还是足够大，使得相应的 DLP 问题求解依然很难呢？当然有这种情况。所以，综上所述，Pohlig-Hellman 算法并不是一种高效的 DLP 求解方法，但是为 DLP 问题求解提供了一种可供参考的研究思路。

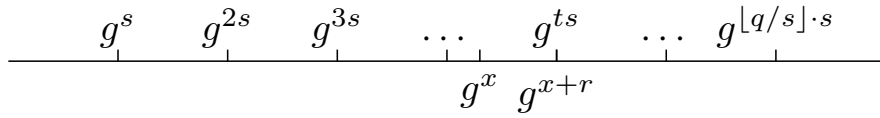


图 16.2: BSGS 算法的几何解释.

16.3 大步小步算法

以下这种称为大步小步算法 (*Baby-Step/Giant-Step algorithm*, 简记为 BSGS) 的 DLP 求解算法是一种通过牺牲存储来换取效率的方法, 思路很简单: 通过两种不同的计算方式得到两个形式不同但是值相同的群元素, 从而算出离散对数值。

对任意的群 \mathbb{G} , 选取 $g \in \mathbb{G}$, 设 g 的阶为 p , DLP 实例是 $g^x = y$ 。算法过程如下:

- 大步过程: 将群 \mathbb{G} 分割成若干部分, 每部分之间的区间大小为 $s = \lfloor \sqrt{p} \rfloor$ 。具体计算如下:

$$g^0, g^s, g^{2s}, \dots, g^{\lfloor p/s \rfloor \cdot s}$$

- 小步过程: y 必然落在某个区间, 即存在某个 t 使得 y 落在 $g^{(t-1)s}$ 与 g^{ts} 之间。因此, 计算:

$$y \cdot g, y \cdot g^2, \dots, y \cdot g^s$$

- 将以上计算结果分别存储, 通过搜索发现如果存在某个 r 和某个 t 使得 $y \cdot g^r = g^{ts}$, 此时也称两种不同的计算发生了一次碰撞 (*Collision*)。于是 $x = (ts - r) \pmod{p}$, 简单计算可得 x 。

图16.2展示了 BSGS 算法的直观思路。图中上层标识了大步过程的群分割, 下层标识小步过程的逼近计算。可能需要强调, 这个图是为了展示算法的直观, 实际上每一个区间并不是连续的而是离散的点集, g^{rs} 也不是按顺序排列的值。具体算法展示如下。

```

1 # Input: g,y,n, 其中g是群的基底, y = g^x, p是g的阶.
2 def BSGS(g, y, p):
3     s = floor(sqrt(p))
4     A = [(y*(g^r) % p) for r in range(0, s)] #Baby Step.
5     B = [(g^(t*s) % p) for t in range(1, s+1)] #Giant Step.
6     r,t = 0,0
7     for u in A:
8         for v in B:
9             if u == v: #Collision.
10                 r, t = A.index(u), B.index(v)
11     return ((t+1)*s - r) % p # Return x

```

算法 16.2: "大步小步算法."

例 16.2 设群为 \mathbb{Z}_p^* , $p = 37$, 设 $g = 2$, $y = 15$, 求 x 使得 $g^x = y$ 。首先, $s = \lfloor \sqrt{p} \rfloor = 6$ 大步过程计算:

$$g^s = 27, g^{2s} = 26, g^{3s} = 36, g^{4s} = 10, g^{5s} = 11, g^{6s} = 1$$

小步过程计算:

$$y \cdot g^1 = 30, y \cdot g^2 = 23, y \cdot g^3 = 9, y \cdot g^4 = 18, y \cdot g^5 = 36, y \cdot g^6 = 35,$$

观察可知, $g^{3s} = y \cdot g^s$, 即 $t = 3$, $r = 5$, 计算 $x = (ts - r) \pmod{p} = 13$ 。用 SageMath 验证计算结果。

```
1 sage: p = 37
2 sage: g = Mod(2, p)          #表示g是模p下的值
3 sage: y = g^13               #因此, 所有与g相关的计算都模p
4 sage: bsgs(g, y, (0,p))     #这里要求提供的是一个区间
5 13
```

BSGS 算法的正确性主要体现在大步过程与小步过程所得计算结果必然会发生碰撞。图 16.2 已经做了很好的说明, 其实质就是除法算法。BSGS 算法的计算开销重点包括: 大步、小步过程, 分别是 $O(\sqrt{p})$ 次群乘法; 排序与搜索的开销是 $O(\sqrt{p} \log \sqrt{p})$ 。所以, BSGS 算法的复杂性就是 $O(\sqrt{p} \log p)$ 。

16.4 Rho 算法

Rho 算法与 BSGS 算法有类似的之处, 都是希望通过计算出两个形式不同但是值相同的群元素, 即通过找到碰撞来求解 DLP。Rho 算法不同于 BSGS 算法, 它并不存储大量的数据, 而是通过两个不同的随机计算过程 (理论上称为随机游动) 来寻找碰撞。这是一种通用的思路, 也可以用于分解大整数, 正如我们在第 15 章看到的。

与之前一样, 考虑任意的群 \mathbb{G} , 选取 $g \in \mathbb{G}$, 设 g 的阶为 q , DLP 实例是 $g^x = y$ 。算法过程是一次“龟兔赛跑”过程:

1. 设定一个随机函数 f , 给定初始值 $lg, ly, rg, ry \in \mathbb{N}$, 这些值也称为指标值。计算 $left = g^{lg} y^{ly}$ 和 $right = g^{rg} y^{ry}$ 。左值 $left$ 扮演乌龟的角色, 右值 $right$ 扮演兔子的角色。
2. 迭代做以下计算: 左值 $left$ (乌龟) 进行一次慢计算 $left = f(left)$, 右值 $right$ (兔子) 进行一次快计算 $right = f(f(right))$ 。所谓“快计算”指的是它比“慢计算”多做了一次随机函数 f 的求值。注意, 随机函数 f 的计算除了更新 $left$ 和 $right$ 之外, 还要同步更新 lg, ly, rg, ry 等指标值。
3. 如果在某个时刻左值与右值发生碰撞, 即 $left = right$, 则 $x = (lg - rg)/(ry - ly)$ 。

```
1 #Input: y = g^x mod p; Output: x
2 def PollardRhoDLOG(y):
3     left, lg, ly = 1, 0, 0 # left = g^lg * y^ly
```

```

4   right, rg, ry = y, 0, 1 # right = g^rg * y^ry
5   while left != right:
6       left, lg, ly = F(left, lg, ly)
7       right, rg, ry = F(right, rg, ry)
8       right, rg, ry = F(right, rg, ry)
9   s, t = lg - rg, ry - ly
10  if s == 0:
11      return 'fail'
12  return s * (t^(-1))

```

算法 16.3: "Rho 算法."

Rho 算法的过程写成代码如算法 16.3 所示, 其中, 为了方便, 设 g 的阶为素数, 读者能看出原因吗? 随机函数 f 的设定是算法的关键, Rho 算法的发明者 Pollard 建议使用以下函数:

$$f(a) = \begin{cases} ga & \text{如果 } 0 \leq a \leq |\mathbb{G}|/3, \\ a^2 & \text{如果 } |\mathbb{G}|/3 \leq a < 2|\mathbb{G}|/3, \\ ya & \text{如果 } 2|\mathbb{G}|/3 \leq a \leq |\mathbb{G}|. \end{cases}$$

如果 f 设定如上, 则指标值 lg, ly 的更新如下所示, rg, ry 的更新类似。

$$f(lg) = \begin{cases} lg + 1 & \text{如果 } 0 \leq a \leq |\mathbb{G}|/3, \\ 2lg & \text{如果 } |\mathbb{G}|/3 \leq a < 2|\mathbb{G}|/3, \\ lg & \text{如果 } 2|\mathbb{G}|/3 \leq a \leq |\mathbb{G}|. \end{cases}$$

$$f(ly) = \begin{cases} ly & \text{如果 } 0 \leq a \leq |\mathbb{G}|/3, \\ 2ly & \text{如果 } |\mathbb{G}|/3 \leq a < 2|\mathbb{G}|/3, \\ ly + 1 & \text{如果 } 2|\mathbb{G}|/3 \leq a \leq |\mathbb{G}|. \end{cases}$$

需要指出, 以上随机函数及其相关更新规则的设定不具通用性, 对于特定的群应该考虑做某些相应的改变。

例 16.3 设 $q = 23$, $p = 2 * p + 1 = 47$, 都是素数。令 $a = 5$, 这是 \mathbb{Z}_q^* 的生成元, 令 $g = a^2 = 25$, 则 g 是一个阶为 q 的群元。令 $y = g^x = 16$ 是 DLP 实例。请读者留意这里参数设定的含义。初始指标值 $lg = 1, ly = 0, rg = 0, ry = 1$, 即 $left = g = 25$, $right = y = 16$ 。

乌龟过程:

1. $left = left * g = 14$, $lg = 2, ly = 0$
2. $left = left * g = 21$, $lg = 3, ly = 0$
3. $left = left * left = 18$, $lg = 6, ly = 0$

兔子过程:

1. $right = 18, rg = 0, ry = 4$
2. $right = 14, rg = 0, ry = 9$

3. $right = 18, rg = 2, ry = 18$

乌龟跑了三步，兔子其实是跑了六步，所以以上兔子的一步其实是两步，具体每一步的细节就不再详细列出，读者可以自行验证。到了算法第三次迭代，发现 $left = right$ ，非常幸运。而且更幸运的是， $x = (lg - rg)/(ry - ly) = (6 - 2)/(18 - 0)$ ，在模 q 的意义下求得 $x = 13$ 。

Rho 算法的分析不再次展开，能成功求解离散对数问题的概率就是碰撞发生的概率，根据生日悖论，碰撞发生的期望计算步骤是 $\Theta(\sqrt{q})$ ，这就是 Rho 算法的复杂度。

第十六章 习题

1. 本题要求读者重新证明 Pohlig-Hellman 算法的正确性，但是要求有所降低，旨在让读者通过简单的实例去体会 Pohlig-Hellman 算法的发现思路。给定素数 p ，满足 $(p - 1) = p_1 p_2$ ，即 $p - 1$ 只有两个素因子。请证明针对 \mathbb{Z}_p^* 群，Pohlig-Hellman 算法能正确求解离散对数问题。
2. 请用 Pohlig-Hellman 算法解决以下 \mathbb{Z}_p^* 群中的离散对数问题：
 - (a). $p = 71, g = 7, y = 53$;
 - (b). $p = 433, g = 5, y = 392$;
3. 请用 BSGS 算法解决以下 \mathbb{Z}_p^* 群中的离散对数问题：
 - (a). $p = 71, g = 7, y = 32$;
 - (b). $p = 433, g = 5, y = 38$;
4. 对算法 16.3 进行完善，并编程实现求 \mathbb{Z}_p^* 群中离散对数问题。

附录 A 快速乘法算法

设 a 和 b 是连个 n 比特整数，为了方便，也与实际应用相符，假定 $n = 2^k$ ， k 是某个正整数。该算法使用分治法（*divide-and-conquer*）为策略计算 ab 。首先，分别把 a 和 b 写为两部分，每部分 $n/2$ 比特：

$$a = 2^{n/2}a_L + a_R$$

$$b = 2^{n/2}b_L + b_R$$

那么， a 与 b 的乘积可写为：

$$ab = (2^{n/2}a_L + a_R)(2^{n/2}b_L + b_R) = 2^n a_L b_L + 2^{n/2}(a_L b_R + a_R b_L) + a_R b_R.$$

使用等式最右边的表达计算 ab ，正确性是显然的。

为了分析算法的效率，先得到算法开销的递归式如下：

$$T(n) = 3T(n/2) + O(n)$$

即表示存在一个常数 C 使得

$$T(n) \leq 3T(n/2) + Cn \quad (\text{A.1})$$

以下我们要证明：

$$T(2^n) = c(3^n - 2^n), \quad (\text{A.2})$$

其中， c 是 $T(2)$ 和 C （这是等式 A.1 中的常量）中的最大值。

证明 使用数学归纳法证明。

归纳初始. 当 $k = 1$ ，有 $T(2) \leq c(3^1 - 2^1) = c$ ，因为 c 是 $T(2)$ 和 C 中的最大值。

归纳假设. 假设

$$T(2^k) \leq c(3^k - 2^k)$$

归纳步. 根据等式 A.1，有：

$$\begin{aligned} T(2^{k+1}) &\leq 3T(2^k) + C2^k \\ &\leq 3c(3^k - 2^k) + C2^k \\ &\leq c3^{k+1} - c \cdot 3 \cdot 2^k + c2^k \\ &\leq c(3^{k+1} - 2^{k+1}). \end{aligned}$$

□

定理 A.1

两个 n 比特整数的乘法的计算开销是 $O(n^{\log_2 3})$ 比特运算。（注意： $\log_2 3$ 约等于 1.585。）



证明 根据等式A.2得到：

$$\begin{aligned} T(n) &= T(2^{\lceil \log_2 n \rceil}) \leq T(2^{\lceil \log_2 n \rceil + 1}) \\ &\leq c(3^{\lceil \log_2 n \rceil + 1} - 2^{\lceil \log_2 n \rceil + 1}) \\ &\leq 3c \cdot 3^{\lceil \log_2 n \rceil} \leq 3c \cdot 3^{\log_2 n} \\ &= 3cn^{\log_2 3}. \end{aligned}$$

因为 $3^{\log_2 n} = n^{\log_2 3}$ ，最后那个等号成立。所以， $T(n) = O(n^{\log_2 3})$ 。

附录 B 斐波那契数列

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}.$$

```
1 # Function for nth fibonacci number
2 def fibonacci(n):
3     if (n == 0):
4         return 0
5     if (n == 1):
6         return 1
7     a, b = 0, 1
8     for i in range(1,n):
9         c = a + b
10        a = b
11        b = c
12    return b
```

算法 B.1: "Fibonacci Number."

```
1 n = 17
2 for i in range(3, n):
3     print "F%u / F%u = %.9f" %(i, i-1, float(fibonacci(i)) / float(
4         fibonacci(i-1)))
5     print
```

算法 B.2: "Measure the ratio."

$$\begin{array}{ll} F_3/F_2 = 2.000000000 & F_4/F_3 = 1.500000000 \\ F_5/F_4 = 1.666666667 & F_6/F_5 = 1.600000000 \\ F_7/F_6 = 1.625000000 & F_8/F_7 = 1.615384615 \\ F_9/F_8 = 1.619047619 & F_{10}/F_9 = 1.617647059 \\ F_{11}/F_{10} = 1.618181818 & F_{12}/F_{11} = 1.617977528 \\ F_{13}/F_{12} = 1.618055556 & F_{14}/F_{13} = 1.618025751 \\ F_{15}/F_{14} = 1.618037135 & F_{16}/F_{15} = 1.618032787 \end{array}$$

$$F_n \approx \alpha F_{n-1}$$



附录 C 生日悖论

所谓生日悖论问题即：问在 n 个人当中，有两个或以上的人生日相同的概率为多少？准确来说，这不是悖论，只是稍微违反人类直觉罢了。结论是，如果 $n = 23$ ，则有两个人生日相同的概率超过 50%。这个结论是假定每个人的生日都是从一年 365 天这个范围内均匀随机选取的日子。 $n = 23$ 这个答案是否出乎你的预料呢？以下讨论这个概率如何得到。

假设我们有一个盒子里面放了 m 个球，每一球都用标签唯一标注一个号码，范围从 1 到 m 。然后抽取 k 个球，如抽彩票一般，抽完了球还放回去。问以下问题：抽取到同一个球的概率有多大？抽取到同一个球这个事件也称发生了一次碰撞。

注意到，抽取第二个球不与第一个球相同的概率的是 $1 - 1/m$ ，而第三个球不与前面抽到的球相同的概率是 $1 - 2/m$ 。以这种方式思考，得到：

$$Pr(\text{不发生碰撞}) = \prod_{i=0}^{k-1} (1 - i/m) \quad (\text{C.1})$$

如果 $k \leq m$ ，则可以把 $1 - i/m$ 换为 $e^{-i/m}$ ，然后公式 C.1 可以重写为：

$$\begin{aligned} Pr(\text{不发生碰撞}) &= \prod_{i=0}^{k-1} (1 - i/m) \\ &\approx \prod_{i=0}^{k-1} e^{-i/m} \\ &= e^{-\sum_{i=0}^{k-1} (i/m)} \\ &= e^{-k(k-1)/2m} \\ &\approx e^{-k^2/2m}. \end{aligned}$$

$$Pr(\text{发生碰撞}) = 1 - Pr(\text{不发生碰撞}) \quad (\text{C.2})$$

$$= 1 - e^{-k^2/2m} \quad (\text{C.3})$$

希望以 50% 的概率发生一次碰撞，则 $k = \sqrt{2 \ln 2 m} \approx 1.18\sqrt{m}$ 。原始的生日悖论问题中 $m = 365$ ，则得到 $k = \sqrt{(2 \ln 2) \cdot 365} \approx 23$ 。下图展示了 $m = 365$ ， k 取不同值时发生碰撞的概率，可知当 $k = 50$ 时概率基本为 1。也就是说一个 50 人的班级大概率有两位同学会同一天过生日。

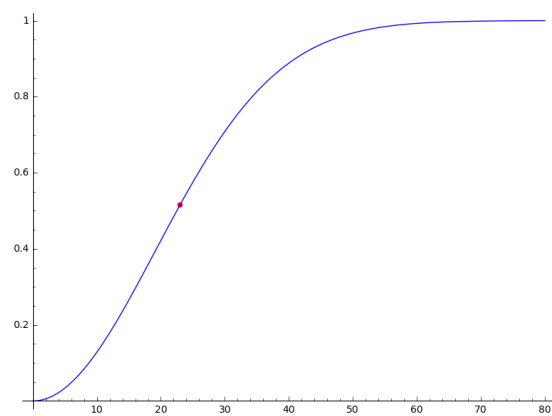


图 C.1: 生日悖论问题的概率.

索引

- Abelian group, 38
- Bézout 系数, 11
- Bézout 定理, 11
- Baby-Step/Giant-Step algorithm, 115
- Bijection, 40
- binary gcd algorithm, 10
- Cancellation Law, 15
- Canonical Homomorphism, 60
- Cayley Theorem, 56
- Characteristic, 87
- closure, 38
- common divisor, 5
- Commutative group, 38
- complete residue system, 17
- composite number, 6
- congruence, 14
- congruence class, 17
- Conjugate, 65
- cyclic group, 44
- direct product, 62
- direct sum, 63
- divide-and-conquer, 120
- divisible, 5
- Division Algorithm, 5
- egcd 算法, 11
- equivalence classes, 16
- Euler's phi function, 26
- extended Euclidean algorithm, 11
- Factor Group, 58
- Field, 79
- Finite Field, 86
- First Isomorphism Theorem, 60
- fixed-size computation, 14
- Fundamental Theorem of Arithmetic, 30
- Galois Field, 86
- gcd algorithm, 8
- generator, 44
- greatest common divisor, 6
- Group, 38
- Homomorphisms, 56
- identity, 38
- Injection, 40
- Integral Domain, 81
- Irreducible Polynomials, 95
- Isomorphism, 54
- Law of Quadratic Reciprocity, 75
- least residue system, 17
- Maximal Ideal, 89
- multiplicative function, 28
- multiplicative inverse, 16
- Normal Subgroup, 57
- Order, 41
- Permutation Group, 41
- Point addition, 102
- Point doubling, 103
- Point multiplication, 106
- position notation, 3
- Prime Ideal, 89
- prime number, 6
- Primitive root, 46
- Principle of Well-Ordering, 5
- Pseudo-polynomial Time, 33
- pseudo-polynomial time, 108
- Quadratic Non-Residue, 72

- quadratic residue, 72
Quotient Groups, 58

relatively prime, 6
Representative, 50
representative, 17
residue class, 17
Ring, 79

scalar multiplication, 106
signed number, 17
Subgroup, 42
Surjection, 40

The Euclidean Algorithm, 8
Two's Complement, 17

Well-defined, 58

不可约多项式, 95
二次互反律, 75
二次剩余, 72
二次非剩余, 72
二进制于, 98
二进制补码, 17
代表元, 50
伪多项式时间, 33, 108
伽罗瓦域, 86
位置计数法, 3

倍点, 103
分治法, 120
单射, 40
双射, 40
合数, 6
同态, 56
同构, 54
域, 79
大步小步算法, 115
封闭性, 38
带符号数, 17

整环, 81
有限域, 86
极大理想, 89
标准同态, 60
标量乘, 106
欧拉准则, 74
满射, 40

点乘, 106
点加, 102
特征, 87
环, 79
直和, 63
直积, 62
第一同构定理, 60
等价类, 16
算术基本定理, 30
素数, 6
素理想, 89
置换群, 41

自然同态, 60
除法算法, 5