



# 网络空间安全数学基础讲义

## Lecture Notes of Essential Mathematics for Cyberspace Security

作者：李晓峰 (cy\_lxf@163.com)

组织：北京联合大学智慧城市学院

时间：November 29, 2021

版本：3.0

课程主页：[http://buuer\\_xxtxiaofeng.gitee.io/lxf/](http://buuer_xxtxiaofeng.gitee.io/lxf/)



*Victory won't come to us unless we go to it. — M. Moore*

## 前言

2018 年学院成立了信息安全专业并开始招生, 开始上“信息安全数学基础”一课, 由于在 2015 年网络空间安全成为一级学科, 故将此讲义命名为“网络空间安全数学基础”, 在备课中形成此讲义, 内容是汇编而来, 只是希望能够在内容安排上的逻辑结构更加适合教和学, 并把此讲义共享出来, 以供需要的人参考。

在编写此讲义的过程中, 基本思路, 或者预期达到的目的是:

1. 通过不吝啬加小标题, 力图更加清晰的表示概念之间的逻辑关系。
2. 在描述概念时, 尽量说明此概念引入的目的, 这里的“目的”也许并不是这些概念提出时的原始出发点, 因为没有去考证, 更多地是在本书的逻辑框架下描述概念的引入目的。
3. 为了使得结构更加紧凑, 减少篇幅, 省去了很多定理的证明, 在讲义中尽量将形成此讲义时定理内容出处标出(有可能有遗漏), 如果想看这些定理证明, 可以看这些参考文献和书籍。以后, 如果有时间, 会把所有定理证明做为书的附录列出。
4. 在习题中, 强调对基本概念的掌握, 不去强调和训练一些数学技巧。

在备课过程中, 参考了国外一些讲数论的授课计划, 看到都在使用 Maple、Sagemath 等工具, 所以在本书的前面, 也有一些 Sagemath 的例子, 但是在进入授课环节后没有使用 Sagemath, 只是用他来做一些习题的验证, 主要考虑就是利用工具, 不利于学生理解基本算法, 比如 gcd 的算法。

本课程在授课前, 在一次闲聊中, 华为(后去了阿里)的雷浩博士曾经提起, “现在的研究生、博士毕业, 让写个算法找素数都不会”, 由于他的这句话, 使我在授课中特别注意了实验课程的安排, 本课程实验让学生通过码云(gitee)提交, 主要是想同学们能够熟悉一些常用的工程工具, 另外课程实验是完成一个简单的命令程序, 可以执行 10 个基本的数论算法, 同时要求基本算法使用 GNU MP 库来实现。

此讲义参考了很多其他老师已经完成并发表的教材, 这些教材在参考文献里面列出, 再此感谢这些老师所做出的出色的工作。

由于水平有限, 在整理此讲义过程中, 难免会出现错误, 请大家不吝指出。

为了方便大家在讲课、学习中使用本讲义, 本讲义的 tex 文件上传到码云上, 网址为: [https://gitee.com/buuer\\_xtxtxiaofeng/FoCysMath](https://gitee.com/buuer_xtxtxiaofeng/FoCysMath), 希望对大家有所帮助。

有关课程的其他信息可以查看课程网页[http://buuer\\_xtxtxiaofeng.gitee.io/1xf/](http://buuer_xtxtxiaofeng.gitee.io/1xf/)。

Hope it works.

李晓峰

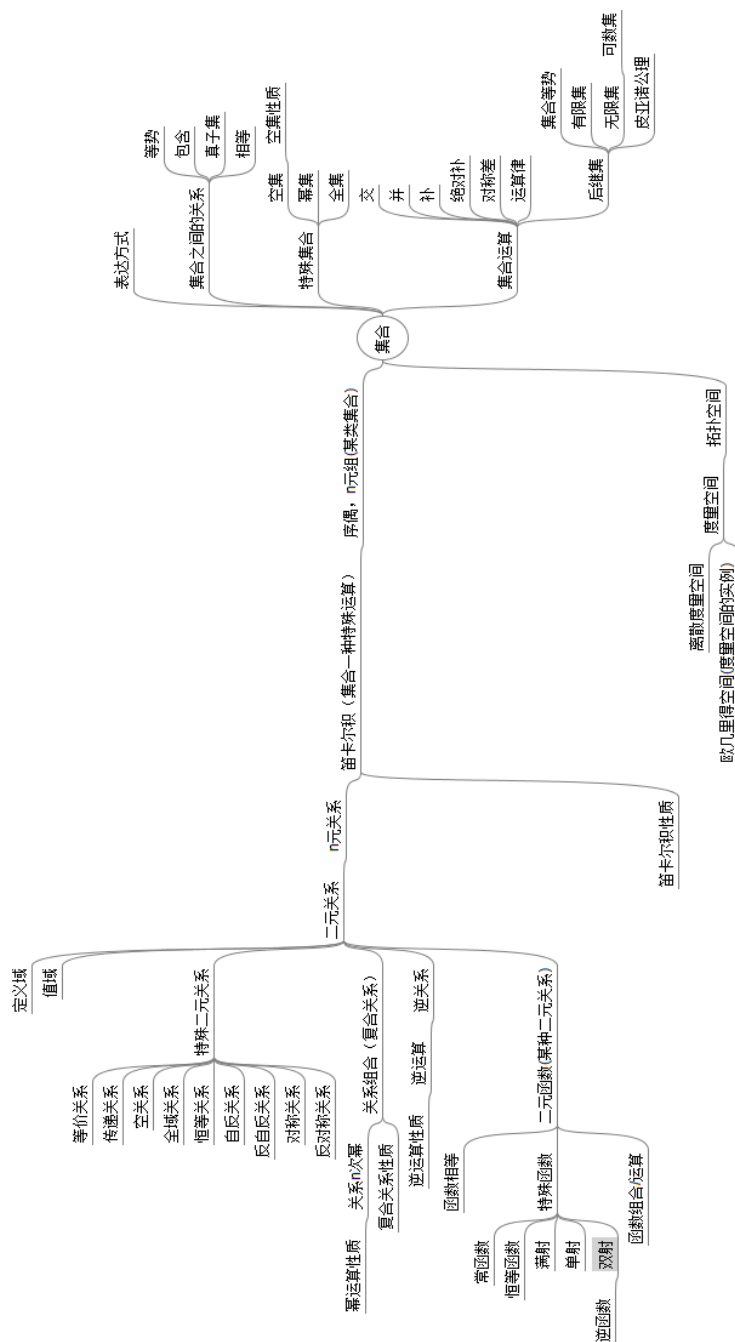


图 1: 集合

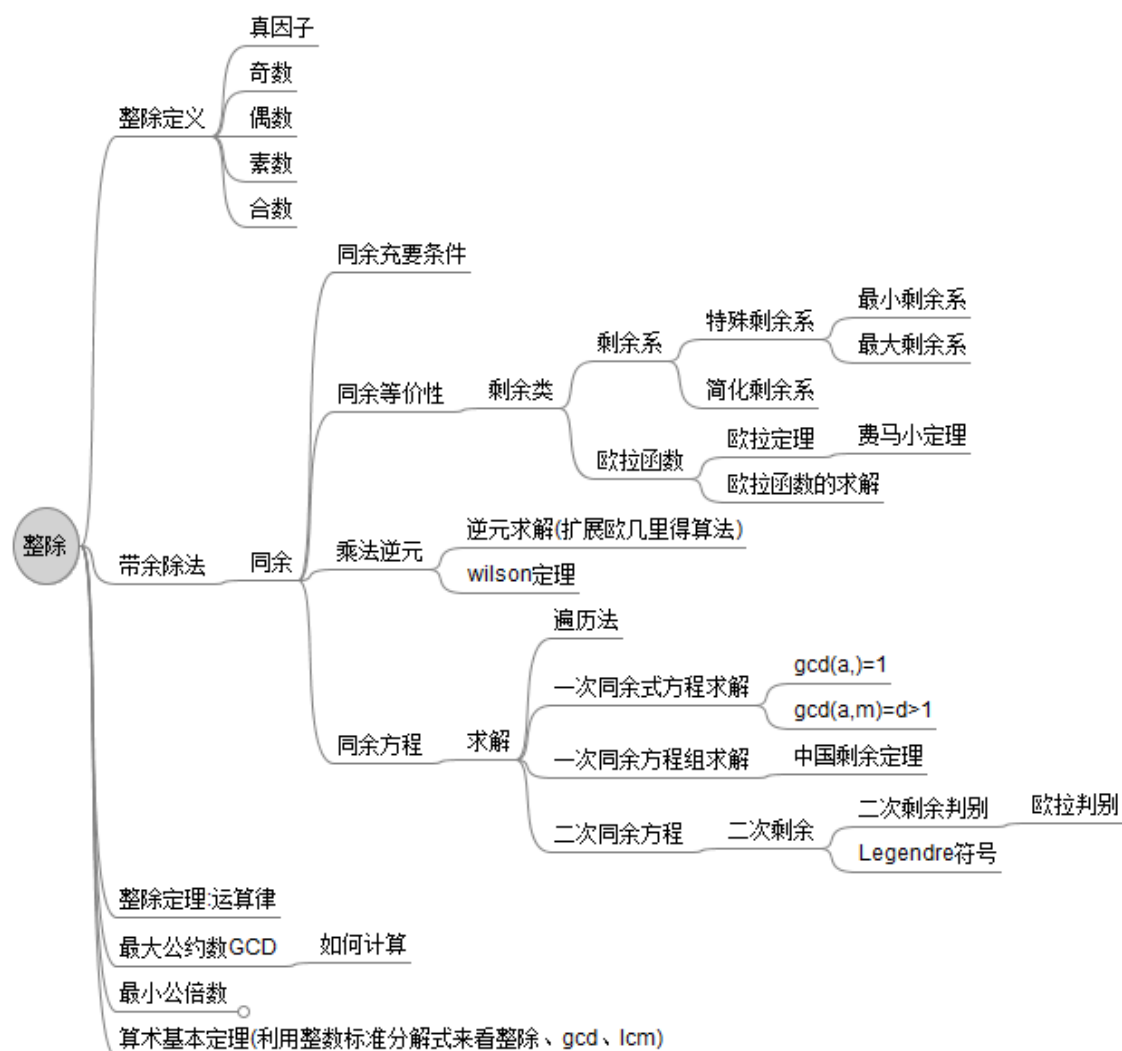


图 2: 整除

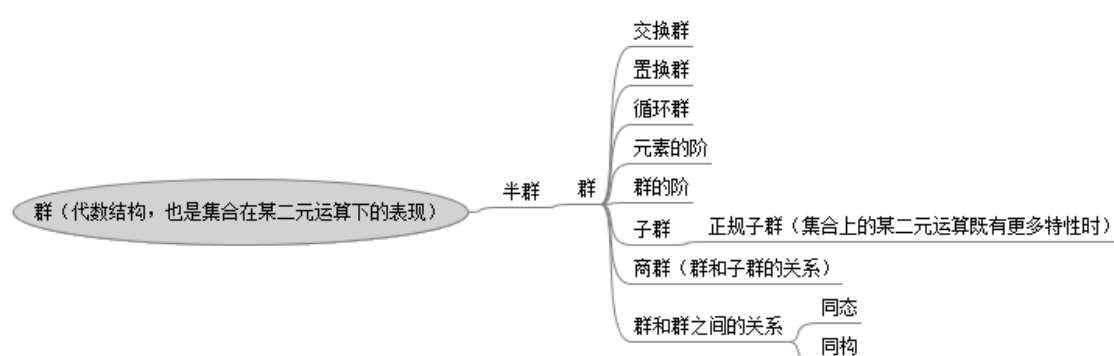


图 3: 群

# 目 录

<b>1 集合论 (set theory)</b>	<b>1</b>
1.1 集合 (set)	1
1.1.1 基本概念	1
1.1.2 集合之间的关系	2
1.1.3 特殊集合	2
1.1.4 集合运算	3
1.1.5 集合的划分	5
1.2 关系 (relation)	6
1.2.1 序偶和 $n$ 元组	6
1.2.2 笛卡尔积	6
1.2.2.1 基本概念	6
1.2.2.2 笛卡尔积运算律	6
1.2.3 关系	7
1.2.3.1 基本概念	7
1.2.3.2 特殊关系	8
1.2.3.3 关系运算	9
1.3 函数 (function)	10
1.3.1 定义	10
1.3.2 函数间关系	11
1.3.3 特殊的函数	11
1.3.4 函数的运算	12
1.4 势 (potential)	12
1.5 皮亚诺算术公理系统	13
1.6 习题	13
<b>2 拓扑学 (topology)</b>	<b>15</b>
2.1 拓扑空间 (topological space)	15
<b>3 组合数学 (combinational mathematics)</b>	<b>17</b>
3.1 排列与组合 (permutation and combination)	17
3.1.1 排列	17
3.1.2 组合	18
3.1.3 二项式定理	18
3.2 生成函数	18



3.2.1	基本定义 . . . . .	18
3.2.2	生成函数的代数运算 . . . . .	19
3.2.3	拆分问题 . . . . .	20
3.2.4	组合计数问题 . . . . .	21
3.2.5	序列 (sequence) . . . . .	21
3.2.5.1	periodic sequence . . . . .	22
3.2.5.2	递归关系 (recurrence relation) . . . . .	23
3.2.5.3	Recursive sequence(recurrent sequence) . . . . .	23
3.2.5.4	线性反馈移位寄存器 (LFSR) . . . . .	24
3.2.5.5	其他 . . . . .	24
<b>4</b>	<b>整除 (divisible)</b>	<b>25</b>
4.1	整除与带余除法 . . . . .	25
4.2	最大公因子 . . . . .	26
4.2.1	基本概念 . . . . .	26
4.2.2	性质 . . . . .	27
4.2.3	欧几里得除法 . . . . .	28
4.3	最小公倍数 . . . . .	29
4.3.1	基本概念 . . . . .	29
4.3.2	性质 . . . . .	29
4.4	算术基本定理 . . . . .	29
4.4.1	算术基本定理 . . . . .	29
4.4.2	整数分解方法 . . . . .	30
4.4.3	标准分解式的应用 . . . . .	30
4.5	完全数、梅森素数和费马素数 . . . . .	31
4.6	习题 . . . . .	33
<b>5</b>	<b>同余 (congruence)</b>	<b>34</b>
5.1	同余的基本概念和性质 . . . . .	34
5.1.1	基本概念 . . . . .	34
5.1.2	性质 . . . . .	34
5.2	剩余系 (residue system) . . . . .	35
5.3	欧拉定理 . . . . .	36
5.3.1	基本概念 . . . . .	36
5.3.2	欧拉函数的计算 . . . . .	37
5.3.3	简化剩余系 . . . . .	37
5.3.4	欧拉定理 . . . . .	38
5.4	乘法逆元 . . . . .	38
5.4.1	逆元 (inverse) 定义 . . . . .	38



5.4.2	逆元求解	38
5.5	费马小定理	40
5.6	威尔逊定理	40
5.7	RSA 12 位密码系统示例	41
5.7.1	加密系统	41
5.7.2	暴力破解	42
5.8	线性同余方程	42
5.8.1	基本概念	42
5.8.2	求解方法	42
5.8.3	线性 (or 一次) 同余方程	43
5.9	同余方程组的求解	43
5.9.1	中国剩余定理	43
5.9.2	一般一次同余方程组的解	45
5.10	高次同余方程的解	45
5.11	习题	46
6	二次剩余 (Quadratic residue)	49
6.1	二次同余方程	49
6.2	欧拉判别条件	50
6.3	勒让德符号	51
6.3.1	概念	52
6.3.2	勒让德符号的运算律	52
6.3.3	勒让德符号的计算	53
6.4	雅克比符号 (Jacobi Symbol)	54
6.4.1	概念	54
6.5	习题	54
7	原根与指数 (primitive root and index)	55
7.1	次数 (order)	55
7.1.1	基本概念	55
7.1.2	次数性质和计算	56
7.2	原根 (primitive root)	59
7.2.1	基本概念	59
7.2.2	原根的等价概念	59
7.2.3	原根存在性判断	59
7.2.4	原根的计算	60
7.3	指数/离散对数	61
7.4	$n$ 次剩余	62
7.5	习题	62



<b>8 群 (group)</b>	<b>64</b>
8.1 基本概念	64
8.2 子群	66
8.3 交换群/阿贝尔群	67
8.4 循环群	67
8.4.1 循环子群的构造	68
8.5 置换群	68
8.6 陪集和商群	70
8.7 同态和同构	72
<b>9 环 (ring)</b>	<b>74</b>
9.1 环 (ring)	74
9.2 子环	75
9.3 同态和同构	75
9.4 理想和商环	75
9.5 多项式环	76
<b>10 域 (field)</b>	<b>77</b>
10.1 基本概念	77
10.2 域上的多项式	77
10.3 同态和同构	78
10.4 域的代数扩张	78
<b>11 椭圆曲线 (elliptic curve)</b>	<b>80</b>
11.1 基本概念	80
11.2 有限域上的椭圆曲线	83
11.3 构造密码算法	85
11.3.1 消息到椭圆曲线上的映射	85
11.3.2 椭圆曲线上的计算困难问题	85
11.3.3 椭圆曲线上的密码算法	85
11.3.3.1 ECC(Elliptic Curve Cryptography) 加密算法	85
11.3.3.2 Diffie-Hellman 密钥交换	86
<b>12 信息论 (information theory)</b>	<b>87</b>
12.0.1 信源 (information source)	87
12.1 多信源	87
12.2 信道 (information channel)	88



<b>13 计算复杂性 (Computational complexity)</b>	<b>89</b>
13.1 简介 . . . . .	89
13.2 什么是计算? . . . . .	92
13.3 计算的数学定义 (形式化描述) . . . . .	93
13.3.1 确定型图灵机 (Deterministic Turing Machine) . . . . .	93
13.3.1.1 描述性定义 . . . . .	93
13.3.1.2 形式化定义 . . . . .	94
13.3.2 随机存取机 (Random-Access Machine, RAM) . . . . .	96
13.3.3 布尔电路 (Boolean Circuits) . . . . .	97
13.3.4 判定树 (Decision tree) . . . . .	98
13.3.5 $\lambda$ 演算 ( $\lambda$ Calculus) . . . . .	98
13.3.6 细胞自动机 (Cellular automata) . . . . .	99
13.3.6.1 能够识别轮廓的细胞自动机 . . . . .	100
13.3.7 随机计算 (Randomized Computation) . . . . .	101
13.3.7.1 概率型图灵机 (Probabilistic Turing machines) . . . . .	101
13.4 计算复杂性 . . . . .	101
13.4.1 计算复杂类 . . . . .	101
13.4.2 邱奇-图灵命题 (Church-Turing Thesis) . . . . .	103
13.5 Uniform and Non-uniform . . . . .	103
13.6 Hyper computation . . . . .	104
<b>A 习题参考解答</b>	<b>106</b>
<b>B 编程练习</b>	<b>114</b>
<b>C 特定标识说明</b>	<b>115</b>
<b>D 工具说明</b>	<b>116</b>
<b>E 用到的模板</b>	<b>117</b>
<b>F 课程中有关 SageMath 函数</b>	<b>118</b>
<b>G Latex 编写格式说明</b>	<b>119</b>
G.1 使用的环境 . . . . .	119
G.2 定义 . . . . .	119
G.3 定理 . . . . .	119
G.4 示例 . . . . .	119
G.5 示例解答 . . . . .	119
G.6 证明 . . . . .	119
G.7 SageMath 示例代码 . . . . .	120

G.8 贴图 . . . . .	120
<b>H 扩展阅读</b>	<b>121</b>
H.1 预言机 (Oracle Machine) . . . . .	121
H.2 布尔电路 (Boolean Circuits) . . . . .	122
H.3 弹跳桌球 (bouncing billiards ball) . . . . .	124
H.3.1 Quantum Algorithms . . . . .	124
H.3.2 Bouncing Billiards . . . . .	125
H.3.3 Historical Examples of Connections . . . . .	126
H.3.4 Further Reading . . . . .	127
H.4 康威生命游戏 (Conway's game of life) . . . . .	127
H.5 指针机 (pointer machine) . . . . .	128
H.6 欧式几何、罗氏几何和黎曼几何 . . . . .	128

# 第 1 章 集合论 (set theory)

## 1.1 集合 (set)

### 1.1.1 基本概念

**定义 1.1 (集合 (Set))** 具有共同性质的一些事物汇集成一个集体，就形成集合。这些事物称为集合的元素或成员。

通常用大写字母表示集合 (如: A, B, C), 用小写字母 (如: a, b, c) 表示元素。如何集合由有限个元素组成, 此集合称为有限集, 否则称为无限集。集合的表示方法通常有两种:

1. 列举法:  $1, 2, 3, 1, 2, 3, 4, 5, 6, \dots$ ;
2. 叙述法:  $x \mid x^2 - 1 = 0, x \mid x > 60$ ;

下面用 SageMath <sup>1</sup> 来进行集合定义方式的示例。

#### sagemath: 罗列方式定义有限集合

```
sage: #有限集的定义: 罗列的方式
.....: x=Set([1,2,3,4,5,6])
.....: x
.....: x.is_finite()
.....:
{1, 2, 3, 4, 5, 6}
True
```

#### sagemath: 描述方式定义有限集合

```
sage: #有限集的定义: 描述的方式
.....: x=range(0,10)
.....: type(x)
.....: #可以看到这时的x是list类型, 并非是set
.....: x=Set(x)
.....: x
.....: #可以看出其已经转换为一个set
.....:
<type 'list'>
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

<sup>1</sup>SageMath 是一个开源的数学计算软件

## sagemath: 无限集合定义

```

sage: #定义一个无限集:是一种描述方式, interpreter
.....: x=R(1,5)
.....: x
.....: #R Interpreter
.....: 2 in x
.....: #True
.....: 2.1 in x
.....: 2.111111 in x
.....:
R Interpreter
True
True
True

```

## 1.1.2 集合之间的关系

有了集合的定义,下面我们要展开对集合的研究,对于一个事物的研究,可以从两个大的方面展开,一个是研究其自身的构成,这是向内的方向,另外一个研究其于周边个体的关系,这是向外的方向,下面我们就从向外的方向看看记着之间有什么样的关系。

**定义 1.2 (包含 (contain))** A 包含于 B, 或 B 包含 A  $\xleftrightarrow{Def} A \subseteq B \Leftrightarrow (\forall x)(x \in A \rightarrow x \in B)$

**定义 1.3 (真子集 (real subset))** A 为 B 的真子集  $\xleftrightarrow{Def} A \subset B \Leftrightarrow (\forall x)(x \in A \rightarrow x \in B) \wedge (\exists x)(x \in B \wedge x \notin A)$

**定义 1.4 (相等 (equal))** A 和 B 相等  $\xleftrightarrow{Def} A = B \Leftrightarrow A \subseteq B \wedge B \subseteq A$

## 1.1.3 特殊集合

看完集合之间的关系,下面我们看看一些特殊的集合。

**定义 1.5 (空集 (empty set))** 不含任何元素的集合称为空集。形式化表示为  $\emptyset = \{x | p(x) \wedge \sim p(x)\}$  其中 p 表示任意谓词,  $\sim$  表示否。

**定义 1.6 (幂集 (power set))** 给定集合 A, 由集合 A 的所有子集组成的集合称为集合 A 的幂集, 记为  $\rho(A)$  或  $2^A$ , 即  $\rho(A) = \{B | B \subseteq A\}$ 。

## 例 1.1

$A = \{1, 2, 3\}$ , 求集合 A 的幂集。

**解**  $\rho(A) = \{\emptyset, A, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}\}$

**定义 1.7 (全集 (universal set))** 在一定范围内, 如果所有的集合均为某一个集合的子集, 则称该集合为全集, 记为  $E$ 。

#### 思考

以上定义中在一定范围是指的什么? 你研究的或者界定的。

全集的形式化表述是什么?  $E = \{x \mid p(x) \vee \sim p(x)\}$

#### 思考

定义完基本对象了 (此处为集合), 对象之间的关系 (或称为运算) 需要接着定义。

### 1.1.4 集合运算

**定义 1.8 (基本运算 (operation))** 1. 交集 (Intersection):  $A \cap B = \{x \mid x \in A \wedge x \in B\}$ 。

2. 并集 (Union):  $A \cup B = \{x \mid x \in A \vee x \in B\}$ 。

3. 补集 (或者成为“差”) (Difference):  $A - B = \{x \in A \wedge x \notin B\} = \{x \mid x \in A \wedge (x \notin B)\}$ 。

4. 绝对补 (absolute complement): 设  $E$  为全集, 对任一集合  $A$  关于  $E$  的补集  $E - A$ , 称为集合  $A$  的绝对补,  $\sim A = E - A = \{x \mid x \in E \wedge x \notin A\}$ 。

5. 对称差 (Symmetric Difference):  $A$  和  $B$  的对称差为集合  $S$ ,  $S = A \oplus B = (A - B) \cup (B - A) = \{x \mid (x \in A \wedge x \notin B) \vee (x \in B \wedge x \notin A)\}$ 。

#### sagemath: 基本运算

```
sage: a=Set([1,2,3,4,5,6])
....: b=Set([0,1,2])
....: c=union(a,b)
....: c
....: c=a.intersection(b)
....: c
....: c=a.difference(b)
....: c
....: c=a.symmetric_difference(b)
....: c
....:
[0, 1, 2, 3, 4, 5, 6]
{1, 2}
{3, 4, 5, 6}
{0, 3, 4, 5, 6}
```

**定理 1.1 (空集性质)** (1) 对于任意集合  $A$ ,  $\emptyset \subseteq A$ 。(2) 空集是唯一的。

**证明** (1) 假设结论不成立, 那么至少存在一个元素属于空集, 但不属于  $A$ , 按照空集的定义, 显然任何元素都不属于空集, 所以产生矛盾, 故假设错误, 结论得证。

(2) 反证法: 假设有两个不同的空集  $\phi_1, \phi_2$ , 根据空集性质  $\phi_1 \subseteq \phi_2$ ,  $\phi_2 \subseteq \phi_1$ , 根据集合相等的定义, 我们知道这两个集合相等, 与假设矛盾, 结论得证。

### 定理 1.2 (运算律 (Algorithm, Operational Law)) 1. 幂等律

$$A \cup A = A$$

$$A \cap A = A$$

#### 2. 交换律

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

$$A \oplus B = B \oplus A$$

#### 3. 结合律

$$(A \cup B) \cup C = A \cup (B \cup C)$$

$$(A \cap B) \cap C = A \cap (B \cap C)$$

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$

#### 4. 分配律

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

$$A \cap (B - C) = (A \cap B) - (A \cap C)$$

#### 5. 同一律

$$A \cup \emptyset = A$$

$$A \cap E = A$$

$$A - \emptyset = A$$

$$A \oplus \emptyset = A$$

#### 6. 零律

$$A \cup E = E$$

$$A \cap \emptyset = \emptyset$$

#### 7. 互补律

$$A \cup \sim A = E$$

$$A \cap \sim A = \emptyset$$

$$\sim E = \emptyset$$

$$\sim \emptyset = E$$

#### 8. 吸收律



$$A \cup (A \cap B) = A$$

$$A \cap (A \cup B) = A$$

### 9. 摩根定律

$$\sim (A \cup B) = \sim A \cap \sim B$$

$$\sim (A \cap B) = \sim A \cup \sim B$$

$$A - (B \cup C) = (A - B) \cap (A - C)$$

$$A - (B \cap C) = (A - B) \cup (A - C)$$

### 10. 双重否定律

$$\sim (\sim A) = A$$

### 11. 其他

$$A \oplus A = \emptyset$$

$$A - A = \emptyset$$

$$A \cap B \subseteq A$$

$$A \cap B \subseteq B$$

$$A \subseteq A \cup B$$

$$B \subseteq A \cup B$$

$$A - B \subseteq A$$

$$A - B = A \cap \sim B$$

$$A \oplus B = (A - B) \cup (B - A) = (A \cup B) - (A \cap B) = (A \cap B) \cup (\sim A \cap \sim B)$$

 **注意** 文氏图 (Venn Diagram) 用来表示集合间的运算很直观, 便于理解。

### 例 1.2

$$A - B = A \cap \sim B$$

**证明**  $A - B = \{x \mid x \in A \cap x \notin B\} = \{x \mid x \in A\} \cap \{x \mid x \notin B\} = A \cap \sim B$

### 例 1.3

证明幂等律  $A \cup A = A$ 。

**证明** 对于  $A \cup A$  中任意一个元素  $x$ , 有:

$x \in A \cup A \Leftrightarrow x \in A \vee x \in A \Leftrightarrow x \in A$ , 根据集合相等的定义,  $A \cup A = A$  成立。

## 1.1.5 集合的划分

**定义 1.9 (集合的一个划分 (partition))** 集合  $S_1, S_2, \dots, S_m$  是集合  $S$  的子集, 且  $S = S_1 \cup S_2 \cup \dots \cup S_m, S_i \cap S_j = \emptyset, i \neq j$ , 则称  $S_1, S_2, \dots, S_m$  是  $S$  的一个划分。

**定理 1.3 (划分的加法原理)** 设  $S_1, S_2, \dots, S_m$  是集合  $S$  的一个划分, 则  $|S| = |S_1| + |S_2| + \dots + |S_m|$ 。

**定理 1.4 (划分的乘法原理)** 设  $S_1, S_2, \dots, S_m$  是  $m$  个有限集, 则  $|S_1 \times S_2 \times \dots \times S_m| = |S_1| \cdot |S_2| \cdot \dots \cdot |S_m|$ 。

**定理 1.5 (划分的减法原理)** 设  $E$  为全集, 则  $|A| = |E| - |E - A|$ 。

**定理 1.6 (划分的除法原理)** 设  $S$  为有限集合, 它被划分为  $m$  个部分  $(S_1, S_2, \dots, S_m)$ , 且每个部分所包含元素数量相同 ( $|S_1| = |S_2| = \dots = |S_m|$ ), 则:

$$m = \frac{|S|}{|S_i|}, i = 1, 2, \dots, m$$

## 1.2 关系 (relation)

### 1.2.1 序偶和 $n$ 元组

**定义 1.10 (序偶 (ordered couple))** 由两个具有给定次序的个体  $x$  和  $y$  (允许  $x = y$ ) 所组成的序列, 称为序偶, 记作  $\langle x, y \rangle$ . 其中  $x$  称为第一分量,  $y$  称为第二分量。

**定义 1.11 (序偶相等)** 设  $\langle a, b \rangle, \langle x, y \rangle$  是两个序偶, 则  $\langle a, b \rangle = \langle x, y \rangle$  当且仅当  $a = x$  且  $b = y$ .

**定义 1.12 (有序  $n$  元组 (Ordered  $n$ -tuple))** 由  $n$  个具有给定次序的个体  $a_1, a_2, \dots, a_n$  组成的序列, 称为有序  $n$  元组, 记作  $\langle a_1, a_2, \dots, a_n \rangle$ 。

### 1.2.2 笛卡尔积

#### 1.2.2.1 基本概念

**定义 1.13 (笛卡儿积 (Cartesian product))** 设  $A_1, A_2, \dots, A_n$  是任意给定的  $n$  个集合, 若有序  $n$  元组  $\langle a_1, a_2, \dots, a_n \rangle$  的第一个分量是取自集合  $A_1$  里的元素, 第二个分量是取自集合  $A_2$  里的元素,  $\dots$ , 第  $n$  个分量是取自集合  $A_n$  里的元素, 则由所有这样的有序  $n$  元组所组成的集合称为集合  $A_1, A_2, \dots, A_n$  的笛卡儿积, 并用  $A_1 \times A_2 \times \dots \times A_n$  表示, 即  $A_1 \times A_2 \times \dots \times A_n = \{ \langle a_1, a_2, \dots, a_n \rangle \mid a_i \in A_i, i = 1, 2, \dots, n \}$ 。



**注意** 笛卡尔积是一种特殊集合, 什么是“特殊”? 所谓特殊是具有某种特点的一类集合。特点是从特定视角, 也就是从某个观测点所看到他们不同于其他的区别。

#### 例 1.4

$A = a, b, B = 0, 1, 2$ , 求  $A \times B$  和  $B \times A$ .

**解**  $A \times B = \langle a, 0 \rangle, \langle a, 1 \rangle, \langle a, 2 \rangle, \langle b, 0 \rangle, \langle b, 1 \rangle, \langle b, 2 \rangle$

$B \times A = \langle 0, a \rangle, \langle 0, b \rangle, \langle 1, a \rangle, \langle 1, b \rangle, \langle 2, a \rangle, \langle 2, b \rangle$

#### 1.2.2.2 笛卡尔积运算律

**定理 1.7 (笛卡尔积性质)** 1. 交换律不成立, 即  $A \times B \neq B \times A$ 。

2. 结合律不成立, 即  $(A \times B) \times C \neq A \times (B \times C)$ 。

3. 下列分配律是成立:

$$A \times (B \cup C) = (A \times B) \cup (A \times C)$$

$$A \times (B \cap C) = (A \times B) \cap (A \times C)$$

$$(A \cup B) \times C = (A \times C) \cup (B \times C)$$

$$(A \cap B) \times C = (A \times C) \cap (B \times C)$$

$$A \times (B - C) = (A \times B) - (A \times C)$$

$$(A - B) \times C = (A \times C) - (B \times C)$$

4. 若  $C \neq \emptyset$ , 则  $A \subseteq B \Leftrightarrow (A \times C \subseteq B \times C) \Leftrightarrow (C \times A \subseteq C \times B)$ 。

5. 设  $A, B, C, D$  是四个非空集合,  $A \times B \subseteq C \times D \Leftrightarrow A \subseteq C \& B \subseteq D$

### 例 1.5

证明  $(A \oplus B) \times C = (A \times C) \oplus (B \times C)$ .

**证明**  $(A \oplus B) \times C$

$$= ((A - B) \cup (B - A)) \times C$$

$$= (A - B) \times C \cup (B - A) \times C$$

$$= ((A \times C) - (B \times C)) \cup ((B \times C) - (A \times C))$$

$$= (A \times C) \oplus (B \times C)$$

## 1.2.3 关系

### 1.2.3.1 基本概念

**定义 1.14 (n 元关系 (n-ary relation))** 设  $A_1, A_2, \dots, A_n$  是任意给定的集合, 笛卡儿积  $A_1 \times A_2 \times \dots \times A_n$  的任何一个子集  $R$  称为  $A_1, A_2, \dots, A_n$  上的一个  $n$  元关系。

其实关系的本质仍然是一个集合。



**注意** 为什么是笛卡尔积的一个子集? 以笛卡尔二维坐标为例来思考关系的定义。X, Y 如果分别是两个实数集合, 他俩的笛卡尔积可以组成整个二维空间, 平面上的一条直线就是一个关系。

**定义 1.15 (二元关系 (Binary Relation))**  $A, B$  是任意两个集合, 则笛卡儿积  $A \times B$  的任意一个子集  $R$  称为从集合  $A$  到集合  $B$  的一个二元关系,  $\langle a, b \rangle \in R$  也可表示为  $aRb$ . 如果一个二元关系是从集合  $A$  到其自身的关系, 则这样的二元关系称为集合  $A$  上的关系。



**注意** 关系的本质仍为集合。

### 例 1.6

我们定义三个集合,  $A$  为全中国所有大学,  $B$  为所有中国人,  $A \times B$  是  $A, B$  的笛卡尔积, 而联大女生显然是其一个子集, 也就是说联大女生是  $A, B$  上的一个二元关系。

对于二元关系, 除了用序偶集合表示外, 还可以用矩阵表示, 通常也称为关系矩阵。

$A = \{a_1, a_2, \dots, a_m\}$ ,  $B = \{b_1, b_2, \dots, b_n\}$ ,  $R$  为  $A$  到  $B$  的一个二元关系, 则此二元关系  $R$  的关系矩阵  $M_R = [r_{ij}]_{m \times n}$  其中:

$$r_{ij} = \begin{cases} 1, \langle a_i, b_j \rangle \in R \\ 0, \langle a_i, b_j \rangle \notin R \end{cases} \quad (i = 1, 2, \dots, m; j = 1, 2, \dots, n) \quad (1.1)$$



**注意** (1) 概念的表达。

(2) 表达方式: 比如一个集合, 可以表达为  $A = 1, 2, 3$ , 同样也是这个集合在 C 语言中的表达方式会发生变化。体会同一概念在不同层次和体系或语境或上下文中的表达方式的不同。

**定义 1.16 (定义域 (Domain) 和与值域 (range))** 设  $R$  是从集合  $A$  到集合  $B$  的二元关系, 则  $R$  中所有序偶的第一个分量组成的集合称为关系  $R$  的定义域, 记作  $D(R)$ , 由  $R$  中所有序偶的第二个分量组成的集合称为关系  $R$  的值域, 记作  $V(R)$ , 即:

$$D(R) = \{a \mid a \in A \wedge (\exists b)(\langle a, b \rangle \in R)\},$$

$$V(R) = \{b \mid b \in B \wedge (\exists a)(\langle a, b \rangle \in R)\}.$$

### 例 1.7

$$A = \{0, 1\}, B = \{0, 1\}.$$

$$A \times B = \{\langle 0, 0 \rangle, \langle 0, 1 \rangle, \langle 1, 0 \rangle, \langle 1, 1 \rangle\};$$

$$\text{关系 } R_1 = \{\langle 0, 0 \rangle, \langle 1, 1 \rangle\}, R_1 \subset A \times B.$$

$$R_2 = \{\langle 0, 1 \rangle, \langle 1, 0 \rangle\}, R_2 \subset A \times B.$$

关系  $R_1, R_2$  的矩阵表示:

$$M_{R_1} = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}, M_{R_2} = \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix}$$

### 1.2.3.2 特殊关系

**定义 1.17 (空关系 (empty relation) 与全域关系 (universal relation))** 设  $R$  是从集合  $A$  到集合  $B$  的一个二元关系, 若  $R = \emptyset$ , 则称  $R$  为空关系, 若  $R = A \times B$ , 则称  $R$  为全域关系。

**定义 1.18 (恒等关系 (identity relation))** 设  $I_X$  是从集合  $X$  上的一个二元关系, 若  $I_X = \{\langle x, x \rangle \mid x \in X\}$ , 则称  $R$  为恒等关系。



**注意** 空关系、全域关系和恒等关系是唯一的。

**定义 1.19 (自反关系 (reflexive relation))**  $R$  在  $X$  上自反  $\iff \forall x, x \in X \longrightarrow xRx$ 。

有一些特殊关系具有一些特殊性质, 下面我们讨论一些特殊的关系。

**定义 1.20 (反自反关系 (anti-reflexive relation))**  $R$  在  $X$  上反自反  $\iff \forall x, x \notin X \longrightarrow xRx$ 。

**定义 1.21 (对称关系 (symmetrical relation))**  $R$  在  $X$  上对称  $\iff \forall x \forall y, x \in X \wedge y \in X \wedge xRy \longrightarrow yRx$ 。

**定义 1.22 (反对称关系 (antisymmetric relation))**  $R$  在  $X$  上反对称  $\iff \forall x \forall y, x \in X \wedge y \in X \wedge xRy \wedge yRx \longrightarrow y = x$ 。

**定义 1.23 (传递关系 (transitive relation))**  $R$  在  $X$  上传递  $\iff \forall x \forall y \forall z, x \in X \wedge y \in X \wedge z \in X \wedge xRy \wedge yRz \longrightarrow xRz$ 。

**定义 1.24 (等价关系 (equivalence relation))** 设  $R$  是集合  $X$  上的二元关系, 若  $R$  是自反、对称和传递的, 则称  $R$  为  $X$  上的等价关系。

### 1.2.3.3 关系运算

**定义 1.25 (复合关系 (composite relation))** 设  $R$  为  $X$  到  $Y$  的二元关系,  $S$  为  $Y$  到  $Z$  的二元关系, 则  $S \circ R$  称为  $R$  和  $S$  的复合关系。也可写为:

$$S \circ R = \{ \langle x, z \rangle \mid x \in X \wedge z \in Z \wedge (\exists y, y \in Y \wedge \langle x, y \rangle \in R \wedge \langle y, z \rangle \in S) \}$$

关系的复合运算或者合成运算就是求复合关系。复合运算是关系的二元运算, 可以由两个关系生成一个新的关系。

复合关系运算顺序是从右到左。

#### 例 1.8

$R$  表示父子关系,  $R \circ R$  生成一个新的关系, 新生成的关系是“祖孙关系”。

设  $B$  表示兄弟关系,  $R \circ B$  是叔侄关系,  $B \circ R$  是父子关系。

**定理 1.8 (关系复合运算的性质)** 1. 满足结合律,  $P \circ (S \circ R) = (P \circ S) \circ R$ 。

2. 不满足交换律,  $R \circ S \neq S \circ R$ 。

3. 并元素满足分配率,  $P \circ (S \cup R) = (P \circ S) \cup (P \circ R)$

$$(P \cup S) \circ R = (P \circ R) \cup (S \circ R)$$

4. 符合运算对交运算满足包含关系,  $R \circ (S \cap P) \subseteq (R \circ S) \cap (R \circ P)$

$$(S \cap P) \circ R \subseteq (S \circ R) \cap (P \circ R)$$

5. 设  $R$  是  $X$  到  $Y$  的关系,  $I_X$  是  $X$  中的恒等关系,  $I_Y$  是  $Y$  中的恒等关系, 则  $I_X \circ R = R \circ I_Y = R$ 。

**定义 1.26 (关系的  $n$  次幂 (power))** 设  $R$  是集合  $A$  上的二元关系,  $n$  为自然数, 则  $R$  的  $n$  次幂记为  $R^n$ , 定义为:

$$(1) R^0 = I_A;$$

$$(2) R^{n+1} = R^n \circ R;$$

#### 例 1.9

$X = \{0, 1\}$ ,  $X$  的二元关系  $R_1 = \{\langle 1, 1 \rangle\}$ ,  $R_2 = \{\langle 0, 1 \rangle, \langle 1, 0 \rangle\}$ , 分别求  $R_1, R_2$  各次幂。

$$\text{解 } R_1^0 = I_X = \{\langle 0, 0 \rangle, \langle 1, 1 \rangle\}$$

$$R_1^1 = R_1 = \{\langle 1, 1 \rangle\}$$

$$R_1^2 = R_1 \circ R_1 = R_1 = \{\langle 1, 1 \rangle\} = R_1$$

$$R_1^2 = R_1$$

...

$$R_2^0 = I_X = \{\langle 0, 0 \rangle, \langle 1, 1 \rangle\}$$

$$R_2^1 = R_2 = \{\langle 0, 1 \rangle, \langle 1, 0 \rangle\}$$

$$R_2^2 = R_2 \circ R_2 = I_X$$

$$R_2^3 = R_2 \circ R_2^2 = R_2$$

$$R_2^4 = R_2 \circ R_2^3 = I_X$$

**定理 1.9 (幂运算性质)** 设  $R$  是集合  $X$  中的二元关系,  $m, n \in \mathbb{N}$ , 则:

$$(1) R^m \circ R^n = R^{m+n};$$

$$(2) (R^m)^n = R^{m \times n}.$$

**证明** 数学归纳法

当  $n = 0$  时,  $R^m \circ R^n = R^m \circ R^0 = R^m \circ I_X = R^m = R^{m+0} = R^{m+n}$ .

假设  $n = k$  时,  $R^m \circ R^k = R^{m+k}$ .

当  $n = k + 1$  时,  $R^m \circ R^{k+1} = R^m \circ (R \circ R^k) = (R^m \circ R) \circ R^k = R^{m+1} \circ R^k = R^{m+k+1}$ .

由上可知, 对于所有  $m, n \in \mathbb{N}$ .

**定义 1.27 (逆关系 (inverse relation))** 设  $R$  是  $X$  到  $Y$  的二元关系, 若将  $R$  中的每一个序偶的元素顺序互换, 所得到的集合称为  $R$  的逆关系, 记做  $R^c$ 。即:

$$R^c = \{ \langle y, x \rangle \mid \langle x, y \rangle \in R \}.$$



**注意** 任何一个二元关系的逆关系总是存在的。

**定理 1.10 (逆运算性质)** 设  $R_1, R_2, R_3$  都是从  $A$  到  $B$  的二元关系, 则下列等式成立:

$$(1) (R_1 \cup R_2)^c = R_1^c \cup R_2^c;$$

$$(2) (R_1 \cap R_2)^c = R_1^c \cap R_2^c;$$

$$(3) (A \times B)^c = B \times A;$$

$$(4) (\bar{R})^c = \bar{R}^c, \text{ 也可以记为 } (\sim R)^c = \sim R^c;$$

$$(5) (R_1 - R_2)^c = R_1^c - R_2^c.$$

### 例 1.10

证明  $(R_1 \cup R_2)^c = R_1^c \cup R_2^c$ .

**证明**  $\langle x, y \rangle \in (R_1 \cup R_2)^c \Rightarrow$

$$\langle y, x \rangle \in R_1 \cup R_2 \Rightarrow$$

$$\langle y, x \rangle \in R_1 \vee \langle y, x \rangle \in R_2 \Rightarrow$$

$$\langle x, y \rangle \in R_1^c \vee \langle x, y \rangle \in R_2^c \Rightarrow$$


$$\langle x, y \rangle \in R_1^c \cup R_2^c.$$

## 1.3 函数 (function)

### 1.3.1 定义

在所有关系中有一类特殊二元关系, 这类关系定义域和值域有特殊的对应关系。下面我们讨论这种特殊的关系。

**定义 1.28 (函数 (function)/ 映射 (mapping))**  $f$  是任意两个集合  $X$  到  $Y$  的一个关系, 若对于  $\forall x \in X$ , 都有唯一的  $y \in Y$ , 使得  $\langle x, y \rangle \in f$ , 则称关系  $f$  为函数, 记做  $f: X \rightarrow Y$  或  $X \xrightarrow{f} Y$ 。  $x$  称为自变元,  $y$  称为在  $f$  作用下  $x$  的像, 也可记为  $y = f(x)$ 。函数也称为映射。

 **注意** 关系是笛卡尔积的一个子集，而这个子集再加上一定约束就是函数。而这个约束就是对于定义域中的任何一个元素  $x_0$ ，有且只存在一个形如  $\langle x_0, * \rangle$  的序偶。

### 例 1.11

$X = \{1, 2, 3, 4\}, Y = \{a, b, c, d\}$ , 下列哪些关系是函数。

$$f_1 = \{\langle 1, a \rangle, \langle 2, c \rangle, \langle 3, b \rangle, \langle 4, d \rangle\}$$

$$f_2 = \{\langle 1, a \rangle, \langle 2, b \rangle, \langle 3, d \rangle, \langle 4, b \rangle\}$$

$$f_3 = \{\langle 1, a \rangle, \langle 3, b \rangle, \langle 4, d \rangle\}$$

$$f_4 = \{\langle 1, a \rangle, \langle 1, b \rangle, \langle 2, b \rangle, \langle 3, c \rangle, \langle 4, d \rangle\}$$


**解**  $f_1, f_2$  是函数。 $f_3$  不是函数，因为  $f_3(x) \notin Y$ 。 $f_4$  不是函数，因为  $f_4(1)$  对应两个  $Y$  中元素。

### 1.3.2 函数间关系

**定义 1.29 (函数相等)**  $f, g$  都是从  $A$  到  $B$  的函数，若他们有相同的定义域和值域，并且  $\forall x \in A$  都有  $f(x) = g(x)$ ，则称函数  $f$  和  $g$  相等，记为  $f = g$ 。

### 1.3.3 特殊的函数

**定义 1.30 (满射 (surjection))**  $X \xrightarrow{f} Y$ ，如果  $Y$  中的每一个元素都是  $X$  中的一个或者多个元素的像，则称这个映射是满射。

 **注意** 满射我们也可以这样来描述，就是值域中的每个元素  $y$  都有对应的  $x$  存在，使得  $f(x) = y$ 。

**定义 1.31 (单射 (injective))** 从  $X$  到  $Y$  的映射中，若  $X$  中没有两个元素有相同的像，则称这个映射为单射。

**定义 1.32 (双射 (Bijection))** 从  $X$  到  $Y$  的映射中，若既是满射又是单射，称这个映射为双射，也称这样的映射是一一映射或一一对应。

**定义 1.33 (常函数 (constant function))**  $f$  称为常函数  $\iff \exists y_0 \in Y, \forall x \in X, f(x) = y_0$ 。

**定义 1.34 (恒等函数 (identity<sup>2</sup> function))** 如果  $I_X = \langle x, x \rangle \mid x \in X$ ，则称  $I_X = X \rightarrow X$  为恒等函数。

**定理 1.11**  $X$  和  $Y$  为有限集，且  $X$  和  $Y$  的元素个数相同 (记为  $|X| = |Y|$ )，则  $f: X \rightarrow Y$  是单射，当且仅当它是一个满射。

**证明** (1) 若  $f$  是单射，则  $|X| = |f(X)|$ ，已知  $|X| = |Y|$ ，所以  $|Y| = |f(X)|$ ，因为  $Y$  是有限集，因此  $f$  是满射。

(2) 若  $f$  是满射，则  $|Y| = |f(X)|$ ，已知  $|X| = |Y|$ ，所以  $|X| = |f(X)|$ ，因为  $X$  是有限集。因此  $f$  是单射。

<sup>2</sup>an equation that is satisfied for all values of the symbols



以上这个定理只有在有限集时才成立, 在无限集上不一定成立, 如  $f: \mathbb{I} \rightarrow \mathbb{I}, f(x)=2x$ , 这是将整数映射为偶整数的, 显然这是一个单射, 单不是满射。

**定理 1.12** 设  $f: X \rightarrow Y$  是一个双射函数, 那么  $f^c$  是  $Y \rightarrow X$  的双射函数。

### 1.3.4 函数的运算

**定义 1.35 (逆函数/反函数 (Inverse Function))** 设  $f: X \rightarrow Y$  是一个双射函数, 称  $Y \rightarrow X$  的双射函数  $f^c$  为  $f$  的逆函数, 记作  $f^{-1}$ 。

**定义 1.36 (左边可复合 (composite))**  $f: X \rightarrow Y, g: W \rightarrow Z$ , 若  $f(X) \subseteq W$  (或  $Y \subseteq W$ ), 则  $g \circ f = \langle x, z \rangle \mid x \in X \wedge z \in Z \wedge \exists y (y \in Y \wedge y = f(x) \wedge z = g(y))$ , 称  $g$  在函数  $f$  的左边可复合。

 **注意** 复合运算的顺序是从右向左。

**定理 1.13** 两个函数的复合是一个函数。

**定理 1.14** 令  $g \circ f$  是一个复合函数。

(1) 若  $g$  和  $f$  是满射, 则  $g \circ f$  是满射。

(2) 若  $g$  和  $f$  是单射, 则  $g \circ f$  是单射。

(3) 若  $g$  和  $f$  是双射, 则  $g \circ f$  是双射。


**定理 1.15**  $f: X \rightarrow Y \Rightarrow f = f \circ I_X = I_Y \circ f$ 。

**定理 1.16**  $f: X \rightarrow Y$  是双射  $\Rightarrow (f^{-1})^{-1} = f$ 。

**定理 1.17**  $f: X \rightarrow Y, g: Y \rightarrow Z$  都是双射  $\Rightarrow (g \circ f)^{-1} = f^{-1} \circ g^{-1}$ 。

## 1.4 势 (potential)

**定义 1.37 (集合等势 (equivalent set))** 当且仅当集合  $A$  和  $B$  之间存在一一对应的函数, 集合  $A$  与集合  $B$  称为等势, 记作  $A \sim B$ 。

 **注意** 集合等势和集合中元素个数相同是一个概念吗?

显然不是。

**例 1.12** 证明区间  $[0, 1]$  和  $(0, 1)$  等势。

**证明** 设集合  $A = 0, 1, \frac{1}{2}, \dots, \frac{1}{n}, \dots$ , 显然  $A \subseteq [0, 1]$ , 定义  $f: [0, 1] \rightarrow (0, 1)$ , 使得

$$f(x) = \begin{cases} \frac{1}{2} & x = 0 \\ \frac{1}{3} & x = 1 \\ \frac{1}{\frac{1}{x}+2} & x \in \frac{1}{2}, \frac{1}{3}, \dots, \frac{1}{n}, \dots \\ x & x \in (0, 1) - A \end{cases}$$

可知  $f$  是个双射函数, 命题得证。

**定义 1.38 (有限集合与无限集合, 可数集或可列集 (finite set, infinite set, countable set))** 如果存在一个从集合  $0, 1, \dots, n-1$  到集合  $A$  的双射, 那么称集合  $A$  是有限的, 如果  $A$  不


是有限的, 那么称  $A$  为无限的。若有从正整数集合  $1, 2, \dots, n, \dots$  到  $A$  的双射, 则称  $A$  是可数集或可列集。

**定理 1.18** 在集合族上等势关系是一个等价关系。

**定理 1.19** 自然数、正奇数、正偶数、整数集是可数集。

## 1.5 皮亚诺算术公理系统

**定义 1.39 (后继集 (follow set))** 给定集合  $A$ , 集合  $A$  的后继集记为  $A^+ \triangleq A \cup \{A\}$ 。

 **注意** 后继集也称为集合的后继 (successor of a set), 是集合的一种一元运算, 亦称为后继运算, 或后继函数。

**例 1.13**

对于空集  $\Phi, \Phi \triangleq 0, \Phi$  的后继集  $\Phi^+ = 0^+ = \Phi \cup \{\Phi\} = \{\Phi\} \triangleq 1$

$1^+ = \{\Phi, \{\Phi\}\} \triangleq 2$

皮亚诺公理是意大利皮亚诺所构造的算术公理系统中的公理。1889 年, 在数学家戴德金工作的基础上, 皮亚诺在《用一种新方法陈述的算术原理》一书中提出了一个算术公理系统, 这个公理系统有九条公理, 其中四条是关于“相等”的, 五条是刻画数的, 并且以 1 而不是 0 作为基本概念。在后来的著作中, 皮亚诺对这一算术系统作了修改, 去除了关于“相等”的四条公理, 并且以 0 取代 1 作为基本概念, 构造了沿用至今的皮亚诺算术公理系统。<sup>3</sup>


**定理 1.20 (皮亚诺公理 (G. Peano axioms))** (1)  $0 \in \mathbf{N}$

(2)  $x \in \mathbf{N} \rightarrow Sx \in \mathbf{N}$

(3)  $x \in \mathbf{N} \rightarrow Sx \neq 0$

(4)  $x \in \mathbf{N} \wedge y \in \mathbf{N} \wedge Sx = Sy \rightarrow x = y$

(5)  $0 \in M \wedge \forall x(x \in M \rightarrow Sx \in M) \rightarrow \mathbf{N} \subseteq M$  for any property  $M$  (axiom of induction).

 **注意** Cited from Encyclopedia of Mathematics <sup>4</sup>. A system of five axioms for the set of natural numbers  $\mathbf{N}$  and a function  $S$  (successor) on it, introduced by G. Peano (1889). In the first version of his system, Peano used 1 instead of 0 in axioms 1, 3, and 5. Similar axioms were proposed by R. Dedekind (1888).

## 1.6 习题

### Exercise 1

设  $A = \{a, \{a\}\}$ , 下列各式成立吗? 写出判断过程。

$\{a\} \in \rho(A); \{a\} \subseteq \rho(A); \{\{a\}\} \in \rho(A); \{\{a\}\} \subset \rho(A)$

<sup>3</sup>此段描述来自于 <https://baike.baidu.com/item/%E7%9A%AE%E4%BA%9A%E8%AF%BA%E5%85%AC%E7%90%86>, 在原链接的描述总, 此段话引自: 彭漪涟. 逻辑学大辞典: 上海辞书出版社, 2004 年 12 月

<sup>4</sup>Peano axioms. Encyclopedia of Mathematics. URL: [http://www.encyclopediaofmath.org/index.php?title=Peano\\_axioms&oldid=43518](http://www.encyclopediaofmath.org/index.php?title=Peano_axioms&oldid=43518)

**Exercise 2**

全集  $E = \{1, 2, 3, 4, 5\}$ ,  $A = \{1, 4\}$ ,  $B = \{1, 2, 5\}$ ,  $C = \{2, 4\}$ , 计算:

- 1)  $A \cap \sim B$
- 2)  $(A \cup B) \cap (A \cup C)$
- 3)  $\sim (A \cup B)$
- 4)  $\rho(A) - \rho(C)$

**Exercise 3**

$A, B, C$  是任意三个集合, 证明  $A \oplus (B \oplus C) = (A \oplus B) \oplus C$ .

**Exercise 4**

集合  $A = \{16, 17, 18, 19\}$ ,  $R$  为  $A$  上的关系  $R = \{< 16, 17 >, < 17, 18 >, < 18, 19 >\}$ , 请问  $R$  的逆关系存在吗? 如果存在请写出来。

**Exercise 5**

$\mathbb{Z}$  为整数集合, 请问此集合上的相等关系是等价关系吗? 如果是请证明。

**Exercise 6**

请证明笛卡尔积不满足结合律。

**Exercise 7**

$\mathbb{R}$  为实数集合, 二维向量定义为  $V_2 = \{(x, y) \mid x \in \mathbb{R}, y \in \mathbb{R}\}$ , 矩阵  $C = \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix}$ , 定义矩阵与二维向量的乘运算  $A \cdot v_2 = \begin{vmatrix} a & b \\ c & d \end{vmatrix} \cdot (x, y) = (a \times x + b \times y, c \times x + d \times y)$ , 请问  $C \cdot v, v \in V_2$  是从  $V_2$  到  $V_2$  的一个函数吗? 并说明理由。

**Exercise 8**

请分别给出实数域上的满射函数、单射函数、双射函数的例子, 并说出原因。

## 第 2 章 拓扑学 (topology)

### 2.1 拓扑空间 (topological space)



**注意** topology(摘抄自 bing 的英英解释)

1. the study of the properties of geometric figures that are independent of size or shape and are not changed by stretching, bending, knotting, or twisting
2. the family of all open subsets of a mathematical set, including the set itself and the empty set, which is closed under set union and finite intersection
3. the anatomy of a part of the body
4. the study of changes in topography that occur over time and, especially, of how such changes taking place in an area affect the history of that area
5. the relationships between parts linked together in a system such as a computer network

从  $X$  到  $Y$  的函数也称为映射。拓扑空间是一种数学结构 (mathematical structure), 数学结构是一种关系结构, 确切说, 拓扑是集合上的一种结构。度量空间是一种特殊的拓扑空间, 费雷歇 (Fréchet) 将欧几里得空间的距离概念抽象化, 于 1906 年定义了度量空间。

**定义 2.1 (度量空间 (metric space))**  $X$  是一个集合,  $\mathbb{R}$  是实数集,  $d: X \times X \rightarrow \mathbb{R}$  是  $X \times X$  到  $\mathbb{R}$  的映射, 如果对于任意  $x, y, z \in X$ , 有

(1) 正定性 (positive definite):  $d(x, y) \geq 0$ , 且  $d(x, y) = 0 \Leftrightarrow x = y$ ;

(2) 对称性 (symmetry):  $d(x, y) = d(y, x)$ ;

(3) 三角不等式 (the triangle inequality):  $d(x, y) + d(y, z) \geq d(x, z)$ ;

则称  $d$  是  $X$  的一个度量, 称  $(X, d)$  是一个度量空间, 称实数  $d(x, y)$  为  $x$  到  $y$  的距离。

**定义 2.2 (离散度量空间 (discrete spaces))**  $(X, d)$  是一个度量空间, 若对于每一个  $x \in X$  存在实数  $\delta_x > 0$ , 使得对于任意  $y \in X, y \neq x$ , 有  $d(x, y) > \delta_x$ , 则称此度量空间是离散度量空间。

**定义 2.3 (拓扑空间 (topological space))**  $X$  是一个集合,  $\tau$  是  $X$  的一个子集族, 如果  $\tau$  满足下列条件:

(1)  $X, \emptyset \in \tau$ ;

(2)  $A, B \in \tau \Rightarrow A \cap B \in \tau$ ;

(3)  $\tau_1 \subset \tau \Rightarrow \bigcup_{A \in \tau_1} A \in \tau$ ;

则称  $\tau$  是集合  $X$  的一个拓扑, 称  $(\tau, X)$  是一个拓扑空间。 $\tau$  中每一个元素称为  $(\tau, X)$  中的开集 (open set)。

开集定义就是拓扑空间定义。

**定义 2.4 (闭集和闭包 (closed set))** 开集的补集称为闭集。

**定义 2.5 (平凡拓扑 (trivial topology))**  $X$  是一个集合, 令  $\Gamma = \{X, \emptyset\}$ , 则  $\Gamma$  是  $X$  的一个拓扑, 称此类拓扑为平凡拓扑。

**定义 2.6 (离散拓扑 (discrete topology))**  $X$  是一个集合, 令  $\Gamma = \rho(X)$  为  $X$  的幂集, 则  $\Gamma$  是  $X$  的一个拓扑, 称此类拓扑为离散拓扑。

离散拓扑是最细拓扑, 平凡拓扑是最粗的拓扑。

## 第3章 组合数学 (combinational mathematics)

现代数学可以分为两大类：一类是研究连续对象的，如分析学、方程等，另一类就是研究离散对象的数学。

有人认为广义的组合数学就是离散数学，也有人认为离散数学是狭义的组合数学和图论、代数结构、数理逻辑等的总称。但这只是不同学者在叫法上的区别，随着计算机科学的日益发展，组合数学的重要性也日渐凸显，因为计算机科学的核心内容是使用算法处理离散数据。

组合数学不仅在基础数学研究中具有极其重要的地位，在其它的学科中也有重要的应用，如计算机科学、编码和密码学、物理、化学、生物学等学科中均有重要应用。微积分和近代数学的发展为近代的工业革命奠定了基础。而组合数学的发展则是奠定了本世纪的计算机革命的基础。

计算机之所以可以被称为电脑，就是因为计算机被人编写了程序，而程序就是算法，在绝大多数情况下，计算机的算法是针对离散的对象，而不是在做数值计算。确切地说，组合数学是计算机出现以后迅速发展起来的一门数学分支，主要研究离散对象的存在、计数以及构造等方面问题。由于计算机软件的促进和需求，组合数学已成为一门既广博又深奥的学科，其发展奠定了本世纪的计算机革命的基础，并且改变了传统数学中分析和代数占统治地位的局面。正是因为有了组合算法才使人感到，计算机好像是有思维的。

组合数学不仅在软件技术中有重要的应用价值，而且在企业管理、交通规划、战争指挥、金融分析等领域都有重要的应用。在美国有一家用组合数学命名的公司，他们用组合数学的方法来提高企业管理的效益，这家公司办得非常成功。此外，试验设计也是具有很大应用价值的学科，它的数学原理就是组合数学。用组合数学的方法解决工业界中的试验设计问题，在美国已有专门的公司开发这方面的软件。<sup>1</sup>

### 3.1 排列与组合 (permutation and combination)

#### 3.1.1 排列

**定义 3.1 (排列定义)** 从  $m$  个不同的元素中每次取一个元素，依次放到  $n(n \leq m)$  个位置上，称为一个  $n$  排列，所有的可能的排列数记为  $P_n^m$ 。

**定理 3.1 (排列数计算 [1])**  $P_n^m = m(m-1) \dots (m-n+1)$ ;  $P_m^m = m \cdot (m-1) \dots 1 = m!$ 。

<sup>1</sup><https://baike.baidu.com/item/组合数学/821134?fr=aladdin>

### 3.1.2 组合

**定义 3.2 (组合定义)** 从  $m$  个不同的元素中选  $n$  个不同元素, 这样一个选法称为一个  $n$  组合, 所有的可能的组合数记为  $C_n^m$  或  $\binom{m}{n}$ 。

**定理 3.2 (组合数计算)**  $C_n^m = \frac{P_n^m}{n!}$ 。

### 3.1.3 二项式定理

**定理 3.3 (二项式定理)**  $n$  为非负整数, 则  $(a+b)^n = \sum_{i=0}^n C_i^n a^{n-i} b^i$ , 特别  $(1+x)^n = \sum_{i=0}^n C_i^n x^i$ ,  $C_i^n$  称为二项式系数。

我国南宋时期数学家杨辉在他著作《详解九章算法》中提出了杨辉三角形来研究二项式系数 [1]。

## 3.2 生成函数

生成函数 (generation function) 即母函数, 是组合数学中尤其是计数方面的一个重要理论和工具。生成函数有普通型生成函数和指数型生成函数两种, 其中普通型用的比较多。形式上说, 普通型生成函数用于解决多重集的组合问题, 而指数型母函数用于解决多重集的排列问题。最早提出母函数的人是法国数学家 Laplace P.S. 在其 1812 年出版的《概率的分析理论》中明确提出“生成函数的计算”, 书中对生成函数思想奠基人——Euler L 在 18 世纪对自然数的分解与合成的研究做了延伸与发展。生成函数的理论由此基本建立。生成函数的应用简单来说在于研究未知 (通项) 数列规律, 用这种方法在给出递推式的情况下求出数列的通项, 生成函数是推导 Fibonacci 数列的通项公式方法之一, 另外组合数学中的 Catalan 数也可以通过生成函数的方法得到。另外生成函数也广泛应用于编程与算法设计、分析上, 运用这种数学方法往往对程序效率与速度有很大改进。<sup>2</sup>

生成函数对于研究序列问题具有重要意义, 我们看看 MIT 在其课件中的介绍<sup>3</sup>, 也就是说, 运用生产函数的, 可以把序列 (在本书的表述中序列和数列概念为同一概念) 研究变为函数性质的研究, 从而通过对单个函数的研究, 可以知道整个序列的性质, 而对于函数的研究有很多成果是可以使用的。

### 3.2.1 基本定义

**定义 3.3 (生成函数 (无限数列))[1])** 设数列  $a_{n=0}^\infty = a_0, a_1, a_2, \dots, a_n, \dots$  是一个无穷数列, 称以下形式的幂级数为此数列的普通生成函数或寻常生成函数, 简称普生成函数:

$$A(x) = a_0 + a_1 x^1 + a_2 x^2 + \dots + a_n x^n + \dots = \sum_{n=0}^\infty a_n x^n$$

<sup>2</sup><https://baike.baidu.com/item/生成函数>

<sup>3</sup>[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2010/readings/MIT6\\_042JF10\\_chap12.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2010/readings/MIT6_042JF10_chap12.pdf)



## 12 Generating Functions

Generating Functions are one of the most surprising and useful inventions in Discrete Math. Roughly speaking, generating functions transform problems about *sequences* into problems about *functions*. This is great because we've got piles of mathematical machinery for manipulating functions. Thanks to generating functions, we can then apply all that machinery to problems about sequences. In this way, we can use generating functions to solve all sorts of counting problems. They can also be used to find closed-form expressions for sums and to solve recurrences. In fact, many of the problems we addressed in Chapters 9–11 can be formulated and solved using generating functions.

图 3.1: 生成函数的重要意义

普生成函数有时也记为  $G(x)$ 。称以下形式的幂级数为指数生成函数，简称指生成函数：

$$B(x) = a_0 + a_1 \frac{x^1}{1!} + a_2 \frac{x^2}{2!} + \cdots + a_n \frac{x^n}{n!} + \cdots = \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}$$

**定义 3.4 (生成函数的相等 [1])** 设  $E(x)$  是数列  $e_{n=0}^{\infty}$  的生成函数， $F(x)$  是数列  $f_{n=0}^{\infty}$  的生成函数，若  $a_n = b_n (n = 0, 1, 2, \dots)$ ，则称  $E(x)$  和  $F(x)$  相等，记为  $E(x) = F(x)$ 。

**例 3.1 [1]** 由二项式定理，我们知道  $(1+x)^n = \sum_{i=0}^n C_i^n x^i$ ，根据生成函数的定义，我们知道  $(1+x)^n$  就是数列  $C_{i=0}^n$  的普生成函数，由于  $C_i^n = \frac{P_i^n}{i!}$ ，所以我们可以将  $(1+x)^n$  的展开式写为  $(1+x)^n = \sum_{i=0}^n P_i^n \frac{x^i}{i!}$ ，根据指生成函数的定义，我们知道  $(1+x)^n$  也是数列  $C_{i=0}^n$  的指生成函数。

我们可以将生成函数扩展到更一般形式的定义。

**定义 3.5 (函数序列线性无关 [1])** 对于函数序列  $g_n(x)_{n=0}^{\infty}$ ，不存在不全为零的常数  $k_0, k_1, k_2, \dots, k_n, \dots$ ，使  $k_0 g_0(x) + k_1 g_1(x) + \dots + k_n g_n(x) + \dots = 0$ ，我们称此函数序列线性无关。

**定义 3.6 (一般形式生成函数 [1])** 函数序列  $g_n(x)_{n=0}^{\infty}$  线性无关，称以下形式的函数为序列  $a_{n=0}^{\infty}$  的一般形式生成函数。  $G(x) = \sum_{n=0}^{\infty} a_n g_n(x) = a_0 g_0(x) + a_1 g_1(x) + a_2 g_2(x) + \dots + a_n g_n(x) + \dots$

**定义 3.7 (生成函数 (有限数列))** 对于数列  $a_0, a_1, a_2, \dots, a_n$ ，称  $G(x) = a_0 + a_1 x^1 + a_2 x^2 + \dots + a_n x^n = \sum_{i=0}^n a_i x^i$  为该数列的生成函数。

### 3.2.2 生成函数的代数运算

设  $a_{n=0}^{\infty}$  和  $b_{n=0}^{\infty}$  的生成函数分别为  $A(x)$  和  $B(x)$ 。

**定义 3.8 (数量乘积 [1])** 若  $\lambda$  是一个数，则称  $\lambda a_{n=0}^{\infty}$  的生成函数  $C(x)$  为数  $\lambda$  与  $a_{n=0}^{\infty}$  的生成函数  $A(x)$  的数量乘积，记为  $C(x) = \lambda A(x)$ 。

**定义 3.9 (生成函数的和 [1])** 若  $c_n = a_n + b_n (n = 0, 1, 2, \dots, n, \dots)$ ，则称  $c_{n=0}^{\infty}$  的生成函数  $C(x)$  为  $A(x)$  和  $B(x)$  的和，记为  $C(x) = A(x) + B(x)$ 。

**定义 3.10 (生成函数的积 [1])** 序列  $c_{n=0}^{\infty}$  的生成函数为  $C(x)$ ，元素  $c_n = \sum_{i+j=n} a_i b_j$ ，则称  $C(x)$  为  $a_{n=0}^{\infty}$  和  $b_{n=0}^{\infty}$  的普生成函数的积，记为  $C(x) = A(x)B(x)$ 。

序列  $c_{n=0}^{\infty}$  的生成函数为  $C(x)$ , 元素  $c_n = \sum_{i=0}^n \binom{n}{i} a_i b_{n-i}$ , 则称  $C(x)$  为  $a_{n=0}^{\infty}$  和  $b_{n=0}^{\infty}$  的指数生成函数的积, 记为  $C(x) = A(x)B(x)$ 。

**定义 3.11 (生成函数集合 [1])** 全体普生成函数组成集合  $\varepsilon$ , 在此集合中有一个元素  $A(x) = \sum_{n=0}^{\infty} a_n x^n$ , 对于所有的  $n \geq 0, a_n = 0$ , 我们称此生成函数  $A(x)$  为  $\varepsilon$  中的零元, 并记为 0; 若  $a_0 = 1$ , 且对于所有的  $n \geq 1, a_n = 0$ , 我们称此生成函数为  $\varepsilon$  中的幺元, 记为 1。

**定理 3.4 ( $\varepsilon$  上的运算)**  $\bullet$   $\varepsilon$  与加法和乘法组成一个整环。

- $\bullet$   $\varepsilon$  与数乘和加法组成数域上的一个向量空间。
- $\bullet$   $\varepsilon$  与数乘、加法和乘法组成数域上的一个代数。

**定义 3.12 (逆元 [1])** 对于生成函数  $A(x)$ , 若存在生成函数  $B(x)$ , 使得  $A(x)B(x) = 1$ , 称  $B(x)$  为  $A(x)$  的逆元, 记为  $A_{-1}(x)$ , 称  $A(x)$  可逆。

**定理 3.5 (逆元求解)** 普生成函数  $A(x) = \sum_{n=0}^{\infty} a_n x^n$  存在逆元的充分必要条件是  $a_0 \neq 0$ 。逆元  $A^{-1}(x) = \sum_{n=0}^{\infty} \hat{a}_n x^n$ , 其中:

$$\hat{a}_0 = a_0^{-1},$$

$$\hat{a}_n = (-1)^n a_0^{-n-1} \begin{vmatrix} a_1 & a_2 & a_3 & \cdots & a_n \\ a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ 0 & a_0 & a_1 & \cdots & a_{n-2} \\ 0 & 0 & 0 & \cdots & a_2 \\ 0 & 0 & 0 & \cdots & a_1 \end{vmatrix}, n = 1, 2, \dots$$

### 3.2.3 拆分问题

**定义 3.13 (n 的一个 k 拆分)** 将一个正整数  $n$  分解为  $k$  个正整数之和, 即  $n = n_1 + n_2 + \cdots + n_k$  ( $k \geq 1; n_i \geq 1, i = 1, 2, \dots, k$ ), 我们称该分解是  $n$  的一个  $k$  拆分, 并称  $n_i$  为分项。若考虑  $n_i$  之间的顺序, 称这样的拆分为有序拆分, 否则称为无序拆分。

**定义 3.14 (n 的拆分数)** 正整数  $n$  的所有无序拆分的个数称为  $n$  的拆分数, 记为  $p(n)$ , 即  $p(n) = \sum_{k=1}^n p_k(n)$ 。

整数分拆理论, 主要是研究各种类型的分拆函数的性质及其相互关系。早在中世纪, 就有关于特殊的整数分拆问题的研究。18 世纪 40 年代, L. 欧拉提出了用母函数法 (或称形式幂级数法) 研究整数分拆, 证明了不少有重要意义的定理, 为整数分拆奠定了理论基础。解析数论中的圆法的引进, 使整数分拆理论得到了进一步发展。整数分拆与模函数有密切关系, 并在组合数学、群论、概率论、数理统计学及质点物理学等方面都有重要应用。

根据是否考虑分拆部分之间的排列顺序, 我们可以将整数分拆问题分为有序分拆 (composition) 和无序分拆 (partition)。两者之间的区别如下: 在有序分拆中, 考虑分拆部分求和之间的顺序。

假定分拆之间不同的排序记为不同的方案, 称之为  $n$  的有序  $k$  拆分, 如 3 的有序 2 拆分为:  $3=1+2=2+1$ 。我们可以将这个问题建模为排列组合中的“隔板”问题, 即  $n$  个无

区别的球分为  $r$  份且每份至少有一个球, 则需要用  $r-1$  个隔板插入到球之间的  $n-1$  个空隙, 因此总共的方案数为  $C(n-1, r-1)$ 。

在无序拆分中, 不考虑其求和的顺序, 我们称之为  $n$  的无序  $k$  拆分, 如 3 的无序  $k$  拆分为:  $3=1+2$ 。这种拆分可以理解为将  $n$  个无区别的球分为  $r$  份且每份至少有一个球。

一般情况下, 无序拆分的个数用  $p(n)$  表示, 则  $p(2)=1$ ,  $p(3)=2$ ,  $p(4)=4$ 。

在通常情况下, 整数分拆是指整数的无序分拆。<sup>4</sup>

### 3.2.4 组合计数问题

**定理 3.6 (生成函数运算)** (1)  $A(x) = B(x) \iff a_k = b_k, k = 0, 1, 2, \dots$ 。生成函数相同, 生成序列相同。

(2)  $A(x) + B(x) = C(x) \iff a_k + b_k = c_k, k = 0, 1, 2, \dots$ 。

(3)  $A(x)B(x) = C(x) \iff c_0 = a_0b_0, c_1 = a_0b_1 + a_1b_0, \dots, c_k = a_0b_k + a_1b_{k-1} + \dots + a_{k-1}b_1 + a_kb_0, \dots$ 。

有了生成函数的概念, 就可以讨论他与组合计数的关系。

**例 3.2** 4 个相同球放入 5 个不同的盒子里, 要求 1, 2, 每盒最多不超过 1 个, 4, 5 最多不超过两个, 问有多少放法?

**解** 用  $x^k$  表示放  $k$  个球, 现设计一个符合题意的放法: 1, 2 盒各放一个, 3 盒放 0 个, 4 盒放 2 个, 5 盒放 0 个, 符号表示为:  $x^1x^1x^0x^2x^0$ 。

另外, 五个盒子的方法, 用多项式表示为,  $(x^0+x^1)(x^0+x^1)(x^0+x^1)(x^0+x^1+x^2)(x^0+x^1+x^2) = (1+x)^3(1+x+x^2)^2$ , 这个多项式中  $x^4 = x^1x^1x^0x^2x^0$  恰是题中描述的分配方案, 因此, 满足题意分配方案与多项式展开式中的  $x^4$  正好一一对应, 所以  $x^4$  项系数即为方案数目。

**例 3.3** 有 1 克, 2 克, 3 克, 4 克砝码各 1 枚, 问能称出几种重量? 每种重量有几种方案。

**解** 1 克砝码有不取和取, 不取用  $x^0 = 1$  表示, 取用  $x^1$  表示, 记做  $1+x$ 。2 克  $1+x^2$ , 3 克  $1+x^3$ , 4 克  $1+x^4$ , 生成函数  $g(x) = (1+x)(1+x^2)(1+x^3)(1+x^4) = 1+x+x^2+2x^3+2x^4+2x^5+2x^6+2x^7+x^8+x^9+x^{10}$ , 可见, 可称 10 种重量, 每种重量方案数为系数, 5g 方案有两种。

### 3.2.5 序列 (sequence)

我们在看一些参考书时, 会看到有些书说的是序列, 有些书说的是数列, 这两个名词对应的英文通常都是 sequence, 所以大家可以把这两个概念看成一个概念。

<sup>5</sup>A function defined on the set of positive integers whose range is contained in the set considered.

<sup>4</sup><https://baike.baidu.com/item/整数分拆>

<sup>5</sup><https://encyclopediaofmath.org/wiki/Sequence>



**注意** 序列其实就是一类特殊的函数，所以研究函数的方法和有关函数的结论是可以用在序列上的。

An element, or term, of a sequence  $f : \mathbf{N} \rightarrow X$ , where  $\mathbf{N}$  is the set of positive integers and  $X$  is the given set, is an ordered pair  $(n, x)$ ,  $x = f(n)$ ,  $n \in \mathbf{N}$ ,  $x \in X$ , denoted by  $x_n$ . The positive integer  $n$  is called the number (or index) of the term  $x_n$  and the element  $x \in X$  is called its value. The sequence  $f : \mathbf{N} \rightarrow X$  is usually denoted by  $\{x_n\}$  or  $x_n$ ,  $n = 1, 2, \dots$ .

The set of elements of a sequence is always countable; moreover, two different terms of a sequence are different at least with respect to their indices. The set of values of the elements of a sequence may be finite; e.g., the set of values of any stationary sequence, i.e. of a sequence  $\{x_n\}$  all elements of which have one and the same value  $x_n = a$ ,  $n = 1, 2, \dots$  consists of one element.

If  $n_1 < n_2$ , then the term  $x_{n_1}$  of a sequence  $\{x_n\}$  is called a predecessor of an element  $x_{n_2}$ , and the term  $x_{n_2}$  is called a successor of  $x_{n_1}$ . Thus, the set of elements of a sequence is ordered.

Various types of sequences are encountered in many branches of mathematics. They help to describe many properties of objects under study. For instance, if  $X$  is a *Topological space*, then among the sequences of points of it an important role is played by convergent sequences, i.e. by sequences that have a *Limit* in this space. Convergent sequences are convenient (at least when a countable base is available) for the description of such properties as compactness, existence of a limit of a mapping, continuity of a mapping, etc. If all elements of a sequence of some objects (points, sets, mappings, etc.) have a certain property, it is often important to find out whether this property is preserved at a limit point of this sequence. For example, to consider the behaviour of such properties as measurability, continuity, differentiability, and integrability under limit transition for different types of convergence of functions (pointwise convergence, convergence almost-everywhere, uniform convergence, convergence in measure, convergence in the mean, etc.).

Sometimes a mapping  $f : \overline{1, n} \rightarrow X$  from a finite set  $\overline{1, n} = \{1 \dots n\}$  of positive integers into a set  $X$  is called a finite sequence and is denoted by  $\{x_1 \dots x_n\}$ , where  $x_k = f(k)$ ,  $k = 1 \dots n$ . A sequence can be given by a formula for its general term (e.g. an arithmetical sequence), by a recurrence formula (e.g. the sequence of Bernoulli numbers) or simply by a verbal description with a certain degree of efficiency (e.g. the sequence of all positive prime integers in ascending order).

### 3.2.5.1 periodic sequence

A sequence  $\{a_i\}$  is said to be periodic with period  $p$  with if it satisfies  $a_i = a_{i+np}$  for  $n = 1, 2, \dots$ . For example,  $\{1, 2, 1, 2, 1, 2, 1, 2, \dots\}$  is a periodic sequence with least period 2. <sup>6</sup>

<sup>6</sup><https://mathworld.wolfram.com/PeriodicSequence.html>

### 3.2.5.2 递归关系 (recurrence relation)

**定义 3.15 (递归关系 [1])** 若  $k$  元整变量函数  $f(n_1, n_2, \dots, n_k)$  与某一确定的 1 元函数  $g(x_1, x_2, \dots, x_l)$  之间有如下关系:

$$g(f(m_1^{(1)}, m_2^{(1)}, \dots, m_k^{(1)}), \dots, f(m_1^{(l)}, m_2^{(l)}, \dots, m_k^{(l)})) = 0, \quad (3.1)$$

其中  $m_i^{(j)}$  为一函数  $m_i^{(j)} = m_i^{(j)}(n_1, n_2, \dots, n_k)$ , 我们称(3.1)为函数  $f$  的一个递归关系。若函数  $f$  满足递归关系(3.1), 则称  $f$  为此递归关系的一个解。若  $g$  是一个线性函数, 且(3.1)为等式关系, 称(3.1)为一个线性递归关系, 不是线性递归关系的称为非线性递归关系。

非线性递归关系既包括函数  $g$  是非线性函数的情况, 也包括  $g$  是线性函数, 而(3.1)为不等式的情况。

**定义 3.16 (一元线性递归关系 [1])** 设数列  $\{u_n\}$  满足一元线性递归关系  $u_{n+r} = a_1 u_{n+r-1} + a_2 u_{n+r-2} + \dots + a_r u_n (n \geq 0)$ , 其中  $a_i (i = 1, 2, \dots, r)$  为常数, 如果  $a_r \neq 0$ , 称其为  $r$  阶线性递归关系。

### 3.2.5.3 Recursive sequence (recurrent sequence)

<sup>7</sup> A sequence  $a_0, a_1, \dots$ , defined over a [[field]]  $K$  that satisfies a relation

$$a_{n+p} + c_1 a_{n+p-1} + \dots + c_p a_n = 0, \quad (3.2)$$

where  $c_1, \dots, c_p$  are constants. The relation permits one to compute the terms of the sequence one by one, in succession, if the first  $p$  terms are known. A classical example of such a sequence is the sequence of [[Fibonacci numbers]] 1, 1, 2, 3, 5, 8 defined by  $a_{n+2} = a_{n+1} + a_n$  with  $a_0 = 0$ ,  $a_1 = 1$ .

The sequences satisfying satisfying (3.2) form a vector space over  $K$  of dimension  $p$  with basis given by the impulse response sequence  $(0, 0, \dots, 1, \dots)$  and its left shifts.

The "characteristic polynomial" (also, companion or auxiliary polynomial) of the recurrence is the polynomial

$$F(X) = X^p + c_1 X^{p-1} + \dots + c_{p-1} X + c_p.$$

It is the characteristic polynomial of the left shift operator acting on the space of all sequences. If  $\alpha$  is a root of  $F$ , then the sequence  $(\alpha^n)$  satisfies (3.2).

A "recursive series" is a [[power series]]  $a_0 + a_1 x + a_2 x^2 + \dots$  whose coefficients form a recursive sequence. Such a series represents an everywhere-defined [[rational function]]: its denominator is the reciprocal polynomial  $X^p F(1/X)$ .

<sup>7</sup>[https://encyclopediaofmath.org/wiki/Recursive\\_sequence](https://encyclopediaofmath.org/wiki/Recursive_sequence)

### 3.2.5.4 线性反馈移位寄存器 (LFSR)

The linear complexity (LC) of a sequence is the size in bits of the shortest linear feedback shift register (LFSR) which can produce that sequence. The measure therefore speaks to the difficulty of generating – and perhaps analyzing – a particular sequence. Randomness can be seen as the size of the smallest program to produce a given sequence. But linear complexity is the size of a LFSR "processor" to produce a sequence, and there is an algorithm (Berlekamp-Massey) to measure the LC. So the resulting LC value might be used to measure one view of randomness. <sup>8</sup>

### 3.2.5.5 其他

关于序列的数学研究有很多，比如有关级数 (series) 的研究，特别是幂级数 (power series) 和傅里叶级数 (Fourier series) 等。

---

<sup>8</sup>Linear Complexity: A Literature Survey, Research Comments from Ciphers by Ritter, 网址<http://www.ciphersbyritter.com/RES/LINCOMPL.HTM>



## 第 4 章 整除 (divisible)

### 4.1 整除与带余除法

我们通常用  $\mathbb{N}$  表示正整数（自然数）集合，用  $\mathbb{Z}$  表示整数集合。

**定义 4.1 (整除 (to be divisible by))**  $a, b \in \mathbb{Z}, b \neq 0$ , 如果存在  $q \in \mathbb{Z}$ , 使得  $a = qb$ , 就称  $a$  可被  $b$  整除或者  $b$  整除  $a$ , 记为  $b \mid a$ 。 $a$  是  $b$  的倍数 (multiple),  $b$  是  $a$  的因子 (factor) (或约数、除数) (divisor)。若  $a$  不能被  $b$  整除, 记为  $b \nmid a$ 。

**注** 给定两个数, 如何判断是否是整除关系? 也就是说, 判断整除的算法是什么?

**定义 4.2 (真因子 (proper factor))**  $b \mid a, b \neq \pm 1, b \neq \pm a$ , 称  $b$  为  $a$  的真因子。

**定理 4.1 (整除定理)**  $a, b, c \in \mathbb{Z}$

1.  $b \mid a \Leftrightarrow -b \mid a \Leftrightarrow b \mid -a \Leftrightarrow |b| \mid |a|$ ;
2.  $a \neq 0, b \mid a \Rightarrow |b| \leq |a|$ ;
3.  $b \mid a, c \mid b \Rightarrow c \mid a$ ;
4.  $b \mid a \Rightarrow b \mid ac$ ;
5.  $c \neq 0, b \mid a \Leftrightarrow bc \mid ac$ ;
6.  $b \mid a, b \mid c \Leftrightarrow m, n \in \mathbb{Z}, b \mid ma + nc$ ;

**定理 4.2 (良序原理 (公理)(well ordering principle))** 每一个由非负整数组成的非空集合  $S$  必定含有一个最小元素。

良序原理与我们的直观感觉相同, 但是这个原理无法证明, 所以良序原理是一个公理。

**定理 4.3 (带余除法 (division algorithm))**  $a, b \in \mathbb{Z}, b > 0$ , 则存在唯一的一对整数  $q$  和  $r$ , 使  $a = qb + r$  ( $0 \leq r < b$ )。其中  $a$  称为被除数 (dividend),  $q$  称为商 (quotient),  $r$  称为余数 (remainder)。

**定义 4.3 (奇数 (odd number), 偶数 (even number))**  $a, b, r \in \mathbb{Z}, a = 2q + r, 0 \leq r < 2$ , 若  $r = 0$ , 称  $a$  为偶数, 若  $r = 1$ , 称  $a$  为奇数。

**定义 4.4 (素数 (prime), 合数 (composite number))**  $p \in \mathbb{Z}, p > 1$ , 如果  $p$  的真因子只有 1 和其自身, 称  $p$  为素数 (或质数)。除 1 以外所有非素数的正整数称为合数 (或复合数)。

**定理 4.4 (无穷素数)** 素数有无穷多个。

**证明** (欧几里德证明<sup>1)</sup>)

用反证法, 假定只有有限个素数  $p_1, p_2, \dots, p_k$ , 设  $a = p_1 p_2 \dots p_k + 1$ , 由于  $a$  是合数, 所以  $a$  必有素因子, 不失一般性, 假设这个素因子是  $p_j$  ( $1 \leq j \leq k$ ), 显然  $p_j \mid a$ , 因为  $a - p_1 p_2 \dots p_k = 1$ ,

<sup>1</sup>公元前 300 年左右, 古希腊数学家欧几里德写在《几何原本》中的一个古老定理 (欧几里德定理) 和它的证明, 距今已有两千多年的历史了。(来自于数学科普微信公众号“职业数学家在民间”(微信号: minjianshuxuejia), 其中的一片文章“【人人都能欣赏的数学证明】为什么有无限多个素数?”)



同时  $p_j \mid (p_1 p_2 \dots p_k)$ , 故  $p_j \mid 1$ , 但是因为素数  $p_j \geq 2$ , 与  $p_j \mid 1$  矛盾, 故假定错误, 定理得证。

**定理 4.5 (素数性质)**  $\bullet n \in \mathbb{N}$ , 存在素数  $p$ , 满足  $n < p \leq n! + 1$ 。

$\bullet n \in \mathbb{Z}, n \geq 2 \Rightarrow n! + 2$  与  $n! + n$  之间必没有素数。

$\bullet n$  为合数  $\Rightarrow n$  必有素数因子  $p$  满足  $p \leq \sqrt{n}$ 。

$\bullet$  若  $2^n - 1$  为素数, 则  $n$  必为素数。

**例 4.1** 证明:  $n$  为合数  $\Rightarrow n$  必有素数因子  $p$  满足  $p \leq \sqrt{n}$ 。

**证明** 设  $p$  为  $n$  的最小素因子, 如果  $n = r \cdot s$ ,  $r$  和  $s$  均为  $n$  的真因子, 那么  $p \leq r \wedge p \leq s$ , 所以  $p^2 \leq r \cdot s = n, p \leq \sqrt{n}$

**例 4.2**

质数判断用 2 到  $\sqrt{n}$  之间的所有整数去除正整数  $n$ , 均无法整除, 则  $n$  为质数, 这也是通常编程判断一个数是否为质数的方法。

**例 4.3** 证明: 若  $2^n - 1$  为素数, 则  $n$  必为素数。

**证明** 对于  $n > 1$ , 假设  $n$  为合数,  $n = bc$ ,  $b$  和  $c$  均为大于 1 的整数, 则  $2^b - 1 \mid 2^n - 1$ , 所以  $2^n - 1$  为合数, 这与条件相矛盾。

**注** (1) 如何判断一个数是否为素数? 也就是通常说的素性判定。

(2) 目前常用的素性判断方法是 Miller-Rabin 算法。

### The Sieve of Eratosthenes<sup>2</sup>

The Greek mathematician Eratosthenes (3rd-century B.C.E) designed a quick way to find all the prime numbers. It's a process called the Sieve of Eratosthenes. We're going to see how it works by finding all the prime numbers between 1 and 100. The idea is to find numbers in the table that are multiples of a number and therefore composite, to discard them as prime. The numbers that are left will be prime numbers. The Sieve of Eratosthenes stops when the square of the number we are testing is greater than the last number on the grid (in our case 100). Since  $11^2 = 121$  and  $121 > 100$ , when we get to the number 11, we can stop looking.

## 4.2 最大公因子

### 4.2.1 基本概念

**定义 4.5 (公因子 (common factor), 最大公因子 (greatest common factor), 互素 (relatively prime))** 设  $a_1, a_2, \dots, a_n$  是  $n$  个不全为零的整数, 若整数  $d$  是他们之中每一个数的因子, 那么  $d$  就称为  $a_1, a_2, \dots, a_n$  的一个公因子, 所有公因子中最大的称为最大公因子, 记为  $\gcd(a_1, a_2, \dots, a_n)$ 。若  $\gcd(a_1, a_2, \dots, a_n) = 1$ , 称  $a_1, a_2, \dots, a_n$  互素 (或互质)。

<sup>2</sup>cited from <https://www.smartickmethod.com/blog/math/operations-and-algebraic-thinking/divisibility/prime-numbers-sieve-eratosthenes/>, 在这个网页内有以动画方式表示的筛选过程, 可以参考阅读, 来理解此算法

## 4.2.2 性质

**定理 4.6 (最大公因子性质)** •  $a, b, c$  是任意三个不全为零的整数, 且  $a = bq + c$ ,  $q$  是整数  $\Rightarrow \gcd(a, b) = \gcd(b, c)$ 。

- 对于任意两个整数  $a$  和  $b$ , 一定存在两个整数  $m$  和  $n$ , 使得  $\gcd(a, b) = ma + nb$ , 也就是说  $\gcd(a, b)$  是  $a$  和  $b$  的整系数线性组合。
- $a, b, c \in \mathbb{Z}, c \mid a \wedge c \mid b \Rightarrow c \mid \gcd(a, b)$ .
- $a, b, c \in \mathbb{Z}, c > 0 \Rightarrow \gcd(ac, bc) = \gcd(a, b) \times c$ .
- $a, b, c \in \mathbb{Z}, a \mid bc \wedge \gcd(a, b) = 1 \Rightarrow a \mid c$ .
- $a_1, a_2, \dots, a_n \in \mathbb{Z}, \gcd(a_1, a_2) = d_2, \gcd(d_2, a_3) = d_3, \dots, \gcd(d_{n-1}, a_n) = d_n \Rightarrow \gcd(a_1, a_2, \dots, a_n) = d_n$ .

**例 4.4** 证明上面的性质:  $a, b, c$  是任意三个不全为零的整数, 且  $a = bq + c$ ,  $q$  是整数  $\Rightarrow \gcd(a, b) = \gcd(b, c)$ 。

**证明**  $\gcd(a, b) \mid a, \gcd(a, b) \mid b, c = a - bq \Rightarrow \gcd(a, b) \mid c$

所以,  $\gcd(a, b)$  是  $b$  和  $c$  的公因子

那么有,  $\gcd(a, b) \leq \gcd(b, c)$ 。

$$\left. \begin{array}{l} a = bq + c \Rightarrow \gcd(b, c) \mid a \\ \gcd(b, c) \mid b \end{array} \right\} \Rightarrow \gcd(b, c) \leq \gcd(a, b)$$

由上可知  $\gcd(a, b) = \gcd(b, c)$ 。

**例 4.5** 已知  $35 = 10 \times 3 + 5$ , 可知  $\gcd(35, 10) = 5, \gcd(10, 5) = 5$ 。

**例 4.6** 已知  $529 = 130 \times 4 + 9$

**解** 我们先用 sagemath 计算一下:

**sagemath: 求两数最大公约数**

```
sage: gcd(529,130)
1
sage: gcd(130,9)
1
sage:
```

然后手工计算一下:

$$529 = 130 \times 4 + 9, \gcd(529, 130) = \gcd(130, 9)$$

$$130 = 9 \times 14 + 4, \gcd(130, 9) = \gcd(9, 4)$$

$$9 = 4 \times 2 + 1, \gcd(9, 4) = \gcd(4, 1)$$

$$4 = 1 \times 4 + 0, \gcd(4, 1) = 1$$

所以, 529、130 互质。

## 4.2.3 欧几里得除法

给定任意两个正整数  $a$  和  $b$  (任意两个整数呢?), 假设  $a \geq b$ , 如何求解  $a$  和  $b$  的最大公因数呢?

## 辗转相除法/欧几里得除法 (Euclid's algorithm)

求  $a, b$  最大公因数的方法 (设  $a \geq b$ )

$$a = bq_1 + r_1, 0 < r_1 < b$$

$$b = r_1q_2 + r_2, 0 < r_2 < r_1$$

$$r_1 = r_2q_3 + r_3, 0 < r_3 < r_2$$

...

$$r_{n-1} = r_nq_{n+1} + r_{n+1}, r_{n+1} = 0.$$

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

## 例 4.7

$a=1560, b=1200$ , 求  $a, b$  的最大公因子。

解  $1560 = 1200 \times 1 + 360$

$$1200 = 360 \times 3 + 120$$

$$360 = 120 \times 3 + 0$$

$$\gcd(1560, 1200) = 120$$

为了使得过程更加清晰, 我们做一张表, 展示其过程, 初始  $q_0 = a = 1560, r_0 = b = 1200$ :

$i$	$q_i$	$r_i$	$m_i$	$r_{i+1}$
0	1560	1200	1	360
1	1200	360	3	120
2	360	120	3	0

**定理 4.7 (最大公因子性质)** • 任给两个正整数  $a$  和  $b$ , 一定存在两个整数  $m, n$ , 使得

$\gcd(a, b) = ma + nb$ , 即  $\gcd(a, b)$  是  $a$  和  $b$  的线性组合.

- 设整数  $a, b, c$  满足  $c \mid a \wedge c \mid b$ , 则  $c \mid \gcd(a, b)$ .
- $a, b, c \in \mathbb{Z}, c > 0 \Rightarrow \gcd(ac, bc) = \gcd(a, b)c$ .
- 整数  $a, b$  互素的充分必要条件是存在整数  $x, y$ , 使得  $xa + yb = 1$ .
- 设有整数  $a, b, c$ , 若  $a \mid bc$  且  $\gcd(a, b) = 1$ , 则  $a \mid c$ .
- 设  $a_1, a_2, \dots, a_n \in \mathbb{Z}$ , 其中  $a_1 \neq 0$ . 令  $\gcd(a_1, a_2) = d_2, \gcd(d_2, a_3) = d_3, \dots, \gcd(d_{n-1}, a_n) = d_n \Rightarrow \gcd(a_1, a_2, \dots, a_n) = d_n$ .

## 4.3 最小公倍数

### 4.3.1 基本概念

**定义 4.6 (最小公倍数 (least common multiple))** 设  $a_1, a_2, \dots, a_n$  是  $n$  个整数, 若  $m$  是这  $n$  个数中每一个数的倍数, 则  $m$  就称为这  $n$  个数的一个公倍数. 在  $a_1, a_2, \dots, a_n$  的所有公倍数中最小的正整数称为最小公倍数, 记作  $[a_1, a_2, \dots, a_n]$  或者  $\text{lcm}(a_1, a_2, \dots, a_n)$ .

已知最小公倍数的定义, 如何来求解最小公倍数, 下面我们进行简单的讨论.

**定理 4.8 (互素数的最小公倍数)** 设  $a$  和  $b$  为任意两个互素正整数, 则其乘积即为最小公倍数.

#### 例 4.8

5 和 7 的最小公倍数是  $5 \times 7 = 35$ , 4 和 8 的最小公倍数数是 8, 而不是  $4 \times 8 = 32$ .

### 4.3.2 性质

**定理 4.9 (最小公倍数性质)** 设  $a$  和  $b$  为任意正整数, 则

(1) 若  $m$  是  $a, b$  的任一公倍数, 则  $\text{lcm}(a, b) \mid m$ ;

(2)  $\text{lcm}(a, b) = \frac{ab}{\gcd(a, b)}$ .

(3)  $\gcd(a, b) \times \text{lcm}(a, b) = a \times b$

#### 例 4.9

4 和 8 的最大公约数为 4, 所以  $\text{lcm}(4, 8) = \frac{4 \times 8}{4}$

**定理 4.10 (最小公倍数性质)**  $a_1, a_2, \dots, a_n$  是  $n$  个整数,

- $\text{lcm}(a_1, a_2) = m_2, \text{lcm}(m_2, a_3) = m_3, \dots, \text{lcm}(m_{n-1}, a_n) = m_n \Rightarrow \text{lcm}(a_1, a_2, \dots, a_n) = m_n$ .
- $a_1 \mid m, a_2 \mid m, \dots, a_n \mid m \Rightarrow \text{lcm}(a_1, a_2, \dots, a_n) \mid m$ .

由上面的定理我们可以得出, 对于一组整数, 求其最小公倍数时, 可以转换为求解一系列两两之数的最小公倍数. 也就是我们可以利用两个数最小公倍数的算法做为运算单元, 求解多个数的最小公倍数.

## 4.4 算术基本定理

### 4.4.1 算术基本定理

**定理 4.11 (算术基本定理 (Fundamental Theorem of arithmetic))** 任一大于 1 的整数都可以表示成素数的乘积, 且在不考虑乘积顺序的情况下, 该表达式是唯一的. 即  $n = p_1 p_2 \dots p_s, p_1 \leq p_2 \leq \dots \leq p_s$ , 其中  $p_1 p_2 \dots p_s$  是素数, 并且若  $n = q_1 q_2 \dots q_t, q_1 \leq q_2 \leq \dots \leq q_t$ , 其中  $q_1 q_2 \dots q_t$  是素数, 则  $s = t, p_i = q_i (i = 1, 2, \dots, s)$ .

算术基本定理也被称为整数的唯一分解定理.

**定理 4.12 (标准分解式)** 任一大于 1 的整数都能够唯一地表示成  $n = p_1^{a_1} p_2^{a_2} \dots p_s^{a_s}$ ,  $a_i > 0, i = 1, 2, \dots, s$ , 其中  $p_i < p_j (i < j)$  是素数。

### 4.4.2 整数分解方法

给定一个整数, 计算此整数的分解式的一般方法是, 用小于此整数的素数, 从大到小, 去除此数, 能整除, 则此素数是一个素因子, 然后继续用此方法找商的第一个素因子, 依次类推。

#### 例 4.10

计算 360 的唯一分解。

**解**  $360 = 2^3 \times 3^2 \times 5$

#### 例 4.11

大数分解 6596783, 6596784。

**解**

#### sagemath: 整数分解

```
sage: factor(6596783)
6596783

sage: factor(6596784)
2^4 * 3^2 * 61 * 751

sage:
```

#### 注意

(1) 素数表。做整数分解, 最直观的方法就是依次用素数去除, 可以整除就是其一个因子, 可见先有一张素数表很重要, 那么怎么来准备这张素数表呢?

(2) 大数运算。编程语言通常的数的范围有限, 所以通常在进行密码学上的计算或者一些数据计算时, 需要有大数运算能力, 这需要专有运算库来支持。

(3) C 语言 int 占 2 bytes, 符号位占一位, 正数最大为  $2^{15} = 32768$

(4) 开源项目 GMP(GNU multiple precision arithmetic library), 网站 <https://gmplib.org/>

### 4.4.3 标准分解式的应用

**定理 4.13 (正因子个数)** 设正整数  $n$  的标准分解式为  $n = p_1^{a_1} p_2^{a_2} \dots p_s^{a_s}$ ,  $a_i \geq 0, i = 1, 2, \dots, s$ ,  $\tau(n)$  表示  $n$  的所有正因子的个数, 则  $\tau(n) = \tau(p_1^{a_1})\tau(p_2^{a_2})\tau(p_s^{a_s}) = (a_1 + 1)(a_2 + 1) \dots (a_s + 1)$

**例 4.12**  $84 = 2^2 \times 3 \times 7$ , 求  $\tau(84)$

**解**  $\tau(84) = (2+1)(1+1)(1+1) = 12$ , 也就是说 84 共有 12 个正因子, 下面我们罗列一下这 12 个正因子:

1,84,2,3,4,6,7,12,14,21,28,42.

### 例 4.13

计算 360 的所有正因子的个数。

**解**  $360 = 2^3 \times 3^2 \times 5$ ,  $\tau(360) = \tau(2^3)\tau(3^2)\tau(5) = (3+1)(2+1)(1+1) = 24$

**定理 4.14 (int)** 数  $a$  和  $b$  的整数分解式为：

$$a = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_s^{\alpha_s}, \alpha_i > 0, i = 1, 2, \dots, s$$

$$b = p_1^{\beta_1} p_2^{\beta_2} \dots p_s^{\beta_s}, \beta_i > 0, i = 1, 2, \dots, s$$

那么：

$$\gcd(a, b) = p_1^{\gamma_1} p_2^{\gamma_2} \dots p_s^{\gamma_s}, \gamma_i = \min(\alpha_i, \beta_i), i = 1, 2, \dots, s.$$

$$\text{lcm}(a, b) = p_1^{\delta_1} p_2^{\delta_2} \dots p_s^{\delta_s}, \delta_i = \max(\alpha_i, \beta_i), i = 1, 2, \dots, s.$$

## 4.5 完全数、梅森素数和费马素数

**定理 4.15 (完全数)** 若正整数  $n$  的所有正因子之和等于  $2n$ ，则  $n$  称为完全数。

### 例 4.14

6 的正因子有 1、2、3、6， $1+2+3+6=12$ ， $2 \times 6=12$ ，根据完全数定义，6 是一个完全数。

我们用  $\sigma(n)$  表示正整数  $n$  的所有正因子之和，所以如果  $n$  是完全数，那么  $\sigma(n) = 2n$ 。

完全数 (Perfect number)，又称完美数或完备数，是一些特殊的自然数。它所有的真因子（即除了自身以外的约数）的和（即因子函数），恰好等于它本身。例如：第一个完全数是 6，它有约数 1、2、3、6，除去它本身 6 外，其余 3 个数相加， $1+2+3=6$ 。第二个完全数是 28，它有约数 1、2、4、7、14、28，除去它本身 28 外，其余 5 个数相加， $1+2+4+7+14=28$ 。第三个完全数是 496，有约数 1、2、4、8、16、31、62、124、248、496，除去其本身 496 外，其余 9 个数相加， $1+2+4+8+16+31+62+124+248=496$ 。后面的完全数还有 8128、33550336 等等。

公元前 6 世纪的毕达哥拉斯是最早研究完全数的人，他已经知道 6 和 28 是完全数。毕达哥拉斯曾说：“6 象征着完满的婚姻以及健康和美丽，因为它的部分是完整的，并且其和等于自身。”有些《圣经》注释家认为 6 和 28 是上帝创造世界时所用的基本数字，因为上帝创造世界花了六天，二十八天则是月亮绕地球一周的日数。圣·奥古斯丁说：6 这个数本身就是完全的，并不因为上帝造物用了六天；事实上，因为这个数是一个完全数，所以上帝在六天之内把一切事物都造好了。

在中国文化里：有六谷、六畜、战国时期的六国、秦始皇以六为国数、六常（仁、义、礼、智、信、孝）、天上四方有二十八宿等等，6 和 28，在中国历史长河中，之所以熠熠生辉，是因为它是一个完全数。难怪有的学者说，中国发现完全数比西方还早呢。

完全数诞生后，吸引着众多数学家与业余爱好者像淘金一样去寻找。它很久以来就一直对数学家和业余爱好者有着一种特别的吸引力，他们没完没了地找寻这一类数字。

接下去的两个完数看来是公元 1 世纪，毕达哥拉斯学派成员尼克马修斯发现的，他在其《数论》一书中有一段话如下：也许是这样，正如美的、卓绝的东西是罕有的，是容易计数的，而丑的、坏的东西却滋蔓不已；是以盈数和亏数非常之多，杂乱无章，它们的发现也毫无系统。但是完全数则易于计数，而且又顺理成章：因为在个位数里只有一个 6；十位数里也只有一个 28；第三个在百位数的深处，是 496；第四个却在千位数的尾巴颈部上，是 8128。它们具有一致的特性：尾数都是 6 或 8，而且永远是偶数。但在茫茫数海中，第五个完全数要大得多，居然藏在千万位数的深处！它是 33550336，它的寻求之路也更加扑朔迷离，直到十五世纪才由一位无名氏给出。这一寻找完全数的努力从来没有停止。电子计算机问世后，人们借助这一有力的工具继续探索。笛卡尔曾公开预言：“能找出完全数是不会多的，好比人类一样，要找一个完美人亦非易事。”时至今日，人们一直没有发现有奇完全数的存在。于是是否存在奇完全数成为数论中的一大难题。只知道即便有，这个数也是非常之大，并且需要满足一系列苛刻的条件。<sup>3</sup>

**定理 4.16 (梅森数 (Mersenne number))**  $p$  是一个素数，形如  $2^p - 1$  的数叫做梅森数，记作  $M_p = 2^p - 1$ ，当  $M_p$  是素数时，则称其为梅森素数。

古希腊数学家欧几里得在名著《几何原本》中证明了素数有无穷多个，并论述完全数时提出：如果  $2^p - 1$  是素数 (其中指数  $p$  也是素数)，则  $2^{p-1}(2^p - 1)$  是完全数。瑞士数学家和物理学家欧拉证明所有的偶完全数都有这种形式。因此，人们只要找到  $2^p - 1$  型素数，就可以发现偶完全数了。数学界将  $2^p - 1$  型素数称为“梅森素数” (Mersenne prime)，因为法国数学家和法兰西科学院奠基人梅森在这方面的研究成果较为卓著。梅森素数貌似简单，但探究难度却极大。它不仅需要高深的理论和纯熟的技巧，而且还需要进行艰巨的计算。到 2013 年 2 月 6 日为止，人类仅发现 48 个梅森素数。值得提出的是：在梅森素数的基础研究方面，法国数学家鲁卡斯和美国数学家雷默都做出了重要贡献；以他们命名的“鲁卡斯-雷默方法” (Lucas-Lehmer primality test) 是目前已知的检测梅森素数素性的最佳方法。此外，中国数学家和语言学家周海中给出了梅森素数分布的精确表达式，为人们寻找梅森素数提供了方便；这一研究成果被国际上命名为“周氏猜测”。

美国中央密苏里大学数学家库珀领导的研究小组通过参加一个名为“互联网梅森素数大搜索” (GIMPS)<sup>4</sup>项目，于 2016 年 1 月 7 日发现了第 49 个梅森素数—— $2^{74207281} - 1$ 。该素数也是目前已知的最大素数，有 22,338,618 位。这是库珀教授第四次通过 GIMPS 项目发现新的梅森素数，刷新了他的记录。他上次发现第 48 个梅森素数  $2^{57885161} - 1$  是在 2013 年 1 月，有 17425170 位。

梅森素数在当代具有重大意义和实用价值。它是发现已知最大素数的最有效途径，其探究推动了“数学皇后”——数论的研究，促进了计算技术、密码技术、程序设计技术和计算机检测技术的发展。难怪许多科学家认为，梅森素数的研究成果，在一定程度上反映了一个国家的科技水平。英国数学协会主席马科斯索托伊甚至认为它的研究进展不但是人类智力发展在数学上的一种标志，也是整个科技发展的里程碑之一。

所有的奇素数都是准梅森数  $2^N - 1$  的因子数，凡是一个素数是四倍金字塔数  $A$  的因

<sup>3</sup><https://baike.baidu.com/item/完全数>

<sup>4</sup>GIMPS 是 great internet Mersenne Prime search, 网站在 [www.mersenne.org](http://www.mersenne.org)



子数，都不是以后梅森合数的因子数，则留下部份素数可能都是梅森合数的因子数。

**定理 4.17 (费马数 (Fermat number))**  $n$  是非负整数，则  $F_n = 2^{2^n} + 1$  称为费马数。当  $F_n$  为素数时，称其为费马素数。

法国数学家费马于 1640 年提出了以下猜想：

$F_0, F_1, F_2, F_3, F_4$  都是质数，因为第六个数实在太太大，费马没有计算出来，他猜测第六个数也是素数。由此提出费马数都是素数的猜想。

1732 年，欧拉算出  $F_5 = 641 \times 6700417$ ，也就是说  $F_5$  不是质数，宣布了费马的这个猜想不成立，它不能作为一个求质数的公式。以后，人们又陆续找到了不少反例，如  $n=6$  时， $F_6 = 2^{2^6} + 1 = 27417767280421310721$  不是质数。至今这样的反例共找到了 243 个，却还没有找到第 6 个正面的例子，也就是说只有  $n=0, 1, 2, 3, 4$  这 5 个情况下， $F_n$  才是质数。

## 4.6 习题

### Exercise 9

证明，若  $2 \mid n, 5 \mid n, 7 \mid n$ ，那么  $70 \mid n$ 。

### Exercise 10

证明任意三个连续的正整数的乘积都被 6 整除。

### Exercise 11

证明每个奇数的平方都具有  $8k+1$  的形式。

### Exercise 12

求如下整数对的最大公因子，并写出求解过程。

(1)(55,85)      (2)(202,282)      (3)(666,1414)      (4)(20785,44350)

### Exercise 13

求如下整数对的最小公倍数，并写出求解过程。

(1)(231,732)      (2)(-871,728)

### Exercise 14

求以下整数的标准分解式，并写出求解过程。

(1)36      (2)69      (3)200      (4)289

### Exercise 15

求以下整数的标准分解式，并写出求解过程。

(1)625      (2)2154      (3)2838      (4)3288



## 第5章 同余 (congruence)

### 5.1 同余的基本概念和性质

#### 5.1.1 基本概念

**定义 5.1 (同余 (congruence))** 定一个正整数  $m$ , 如果用  $m$  去除两个整数  $a$  和  $b$  所得的余数相同, 称  $a$  和  $b$  模 (module)  $m$  同余, 记为  $a \equiv b(\text{mod } m)$ 。否则, 称  $a$  和  $b$  模  $m$  不同余, 记为  $a \not\equiv b(\text{mod } m)$ 。

#### 例 5.1

$$1(\text{mod } 5)=1=6(\text{mod } 5)=11(\text{mod } 5)$$

$$2(\text{mod } 5)=2=7(\text{mod } 5)=12(\text{mod } 5)$$

$$3(\text{mod } 5)=3=8(\text{mod } 5)=13(\text{mod } 5)$$

$$4(\text{mod } 5)=4=9(\text{mod } 5)=14(\text{mod } 5)$$

$$5(\text{mod } 5)=0=10(\text{mod } 5)=15(\text{mod } 5)$$

#### 5.1.2 性质

**定理 5.1 (同余的等价性)** 同余关系是等价关系。

等价是自反、传递、对称, 显而易见, 自己和自己同余,  $x$  和  $y$  同余,  $y$  也一定和  $x$  同余, 如果同时  $y$  和  $z$  同余, 那么  $x$  和  $z$  也同余。



**注意** 当给一个对象贴上一个有明确定义的标签时, 其实就是对这个对象有了一个深刻的说明, 因为这个标签后面的一切属性此对象都会具有。

**定理 5.2 (同余性质)** 1. 数  $a$  和  $b$  模  $m$  同余的充要条件是  $m \mid a - b$ 。

2. 数  $a$  和  $b$  模  $m$  同余的充要条件是, 存在一个整数  $k$  使得  $a = b + km$ 。[2]

3.  $a + b \equiv c(\text{mod } m) \Rightarrow a \equiv c - b(\text{mod } m)$ 。

4.  $a \equiv b(\text{mod } m) \Rightarrow \gcd(a, m) = \gcd(b, m)$ 。

5.  $a = a_1d, b = b_1d, \gcd(d, m) = 1, a \equiv b(\text{mod } m) \Rightarrow a_1 \equiv b_1(\text{mod } m)$ 。[2]<sup>1</sup>

6.  $m, a_1, a_2, b_1, b_2 \in \mathbb{Z}, a_1 \equiv b_1(\text{mod } m), a_2 \equiv b_2(\text{mod } m) \Rightarrow$

(a).  $a_1x + a_2y \equiv b_1x + b_2y(\text{mod } m), x, y \in \mathbb{Z}$

(b).  $a_1a_2 \equiv b_1b_2(\text{mod } m)$

(c).  $a_1^n \equiv b_1^n(\text{mod } m), n > 0$

7.  $f(t) = a_nt^n + a_{n-1}t^{n-1} + \dots + a_1t + a_0, g(t) = b_nt^n + b_{n-1}t^{n-1} + \dots + b_1t + b_0$  是两个整系数多项式, 且  $a_i \equiv b_i(\text{mod } m)$ , 若  $x \equiv y(\text{mod } m)$ , 则  $f(x) \equiv g(y)(\text{mod } m)$ 。

我们可以设  $a = k_1m + r_1, b = k_2m + r_2$ , 然后去讨论证明以上定理。

<sup>1</sup>公式后面这种标识, 表示此公式在整理本书时的出处

设  $m=5$ , 11 和 1 同余吗? 由于  $11 - 1 = 10$ , 10 可以整除 5, 由上面的定理可知, 11 和 1 模 5 同余。或者说, 由于  $11 = 1 + 2 \times 5$ , 所以 11 和 1 模 5 同余。

那么 3879 和 3657 模 5 同余吗?  $3879 - 3657 = 222$ ,  $5 \nmid 222$ , 所以 3879 和 3657 不同余。

前面的同余性质都是在模不变的情况下进行讨论, 下面我们看看另外一些同余的性质, 这些性质是模在变化的时候所保持的。

**定理 5.3**     •  $ac \equiv bc \pmod{m}, \gcd(c, m) = d \Rightarrow a \equiv b \pmod{\frac{m}{d}}$ .

- $a \equiv b \pmod{m} \Rightarrow ak \equiv bk \pmod{mk}, k \in \mathbb{Z}$ .
- $a \equiv b \pmod{m}, d \mid m, d \in \mathbb{Z} \Rightarrow a \equiv b \pmod{d}$ .
- $a \equiv b \pmod{m_i}, i = 1, 2, \dots, n \Rightarrow a \equiv b \pmod{\text{lcm}(m_1, m_2, \dots, m_n)}$

**例 5.2**  $m = 9, 5 \times 3 \equiv 2 \times 3 \pmod{9}, \gcd(3, 9) = 3$ , 根据定理 5.3,  $5 \equiv 2 \pmod{\frac{9}{3}} \Rightarrow 5 \equiv 2 \pmod{3}$   
 $m = 9, 5 \times 6 \equiv 2 \times 6 \pmod{9}, \gcd(6, 9) = 3$ , 根据定理 5.3,  $5 \equiv 2 \pmod{\frac{9}{3}} \Rightarrow 5 \equiv 2 \pmod{3}$

## 5.2 剩余系 (residue system)

同余为整数集合上的等价关系, 所以可以利用同余关系把整数集合  $\mathbb{Z}$  划分为若干等价类。

**定理 5.4 ([2])**  $m$  是一个给定的正整数, 则全部整数可分成  $m$  个集合, 记作  $C_0, C_1, \dots, C_{m-1}$ , 其中  $C_r (r = 0, 1, \dots, m-1)$  是由一切形如  $qm + r (q = 0, \pm 1, \pm 2, \dots)$  的整数所组成, 这些集合具有下列性质:

- (1) 每一个整数必包含在且仅在上述的一个集合里。
- (2) 两个整数在一个集合中的充要条件是这两个整数对模  $m$  同余。

**定义 5.2 (剩余类 (residue class))**  $m$  是一个给定的正整数,  $C_r$  表示所有与整数  $r$  模  $m$  同余的整数组成的集合,  $C_r$  叫做模  $m$  的一个剩余类, 一个剩余类中的任一元素叫做该类的代表元。

### 例 5.3

$$1 \pmod{5} = 11 \pmod{5}, 11 \pmod{5} = 16 \pmod{5} \Rightarrow 1 \pmod{5} = 16 \pmod{5}$$

$$\text{模 } 5, r=1, C_1 = \{1, 6, 11, 16, 21, \dots\}; r = 2, C_2 = \{2, 7, 12, 17, 22, \dots\}; r = 3, C_3 = \{3, 8, 13, 18, 23, \dots\}$$

**定义 5.3 (完全剩余系 (complete system of residues))** 在模  $m$  的剩余类  $C_0, C_1, C_2, \dots, C_{m-1}$  中各取一代表元  $a_i \in C_i, i = 0, 1, \dots, m-1$ , 则此  $m$  个数称为模  $m$  的一个完全剩余系 (又称完系)。

### 例 5.4

模 5 的三个完全剩余系:

$$0, 1, 2, 3, 4$$

1,2,3,4,5  
2,3,4,5,6  
...

**定义 5.4 (特殊剩余系定义)** 对于正整数  $m$

- (1)  $0, 1, \dots, m-1$  为模  $m$  的一个完全剩余系, 并且叫做模  $m$  的最小非负完全剩余系。
- (2)  $1, 2, \dots, m-1, m$  为模  $m$  的一个完全剩余系, 并且叫做模  $m$  的最小正完全剩余系。
- (3)  $-(m-1), \dots, -1, 0$  为模  $m$  的一个完全剩余系, 并且叫做模  $m$  的最大非正完全剩余系。
- (4)  $-m, -(m-1), \dots, -1$  为模  $m$  的一个完全剩余系, 并且叫做模  $m$  的最大负完全剩余系。

**例 5.5**

模  $m=5$ ,  
0,1,2,3,4 是模 5 的最小非负完全剩余系。  
1,2,3,4,5 是模 5 的最小正完全剩余系。  
0,-1,-2,-3,-4 是模 5 的最大非正完全剩余系。  
-1,-2,-3,-4,-5 是模 5 的最大负完全剩余系。

**定理 5.5 ([2])**  $m$  是正整数,  $\gcd(a, m)=1$ ,  $b$  是任意整数, 若  $x$  遍历模  $m$  的一个完全剩余系, 则  $ax+b$  也遍历模  $m$  的完全剩余系。

**定理 5.6 ([2])**  $m_1, m_2$  是两个互质的正整数,  $x_1, x_2$  分别遍历模  $m_1, m_2$  的完全剩余系, 则  $m_2x_1 + m_1x_2$  也遍历模  $m_1 \times m_2$  的完全剩余系。

## 5.3 欧拉定理

### 5.3.1 基本概念

**定义 5.5 (剩余类与  $m$  互素)** 在模  $m$  的一个剩余类中, 若有一个数与  $m$  互素, 则该剩余类中所有数都与  $m$  互素, 此时称该剩余类与  $m$  互素。

**例 5.6**  $m = 26, C_5 = 5, 31, 57, \dots, \gcd(5, 26) = 1, \gcd(31, 26) = 1, \dots$

**例 5.7**  $m = 6, C_5 = 5, 11, 17, \dots, \gcd(5, 6) = 1, \gcd(11, 6) = 1, \dots$

**定义 5.6 (欧拉函数 (Euler's totient function 或称 totient function))** 设  $m$  是正整数, 在  $m$  的所有剩余类中, 与  $m$  互素的剩余类的个数称为  $m$  的欧拉函数, 记为  $\varphi(m)$ 。

我们也可以将欧拉函数定义为, 集合  $\{0, 1, \dots, m-1\}$  中与  $m$  互素的整数个数。

**例 5.8**

$\varphi(6) = 2$ , 这是因为  $\{0, 1, 2, 3, 4, 5\}$  中与 6 互素的只有 1, 5。

$\varphi(12) = 4$ , 这是因为  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$  中与 12 互素的只有 1, 5, 7, 11。

$\varphi(1) = 1$

如果  $p$  为素数,  $\varphi(p) = p - 1$

### 5.3.2 欧拉函数的计算

求解欧拉函数, 我们可以用定义来求解, 这需要我们遍历一个完全剩余系, 并且判断和其互素的数, 运算量还是挺大的。

下面看一种利用整数标准分解式求解欧拉函数的方法。

**定理 5.7 (欧拉函数求解定理)** 设  $m$  有标准分解式  $m = p_1^{a_1} p_2^{a_2} \dots p_s^{a_s}, a_i > 0, i = 1, 2, \dots, s$ , 则  $\varphi(m) = m \prod_{i=1}^s (1 - \frac{1}{p_i})$

**例 5.9**  $360 = 2^2 \times 3^2 \times 5$ ,  $\varphi(360) = 360(1 - \frac{1}{2})(1 - \frac{1}{3})(1 - \frac{1}{5}) = 2^2 \times 3^2 \times 5 \times (1 - \frac{1}{2})(1 - \frac{1}{3})(1 - \frac{1}{5}) = (2^3 - 2^2)(3^2 - 3)(5 - 5^0) = 4 \times 6 \times 4 = 96$

**定理 5.8 ([2])** 若  $m_1, m_2$  是互质正整数, 则,  $\varphi(m_1 m_2) = \varphi(m_1) \varphi(m_2)$ .

### 5.3.3 简化剩余系

**定义 5.7 (缩剩余系/简化剩余系 (reduced residue system))** 设  $m$  是正整数, 在与模  $m$  互素的  $\varphi(m)$  个剩余类中, 各取一个代表元组成一个集合, 此集合叫做模  $m$  的缩剩余系 (有时简称缩系) 或者简化剩余系。

简化剩余系中元素的个数为  $\varphi(m)$ .

#### 例 5.10

$\{1, 5\}$  是  $m=6$  的一个缩系;

$\{1, 2, 3, 4, 5, 6\}$  是  $m=7$  的一个缩系;

$\{1, 5, 7, 11\}$  是  $m=12$  的一个缩系。

利用以下定理可以构造简化剩余系。

**定理 5.9 (简化剩余系构造定理)** 设  $m$  是正整数, 整数  $a$  满足  $\gcd(a, m)=1$ 。若  $x$  遍历模  $m$  的一个简化剩余系, 则  $ax$  也遍历模  $m$  的一个简化剩余系。

#### 例 5.11

整数 5 与 6 互素,  $a=5, m=6$ ,  $x$  遍历  $\{1, 5\}$ ,  $ax$  遍历  $\{5, 25\}$ ,  $\{5, 25\}$  是  $m=6$  的一个缩系;

整数 3 与 7 互素,  $a=3, m=7$ ,  $x$  遍历  $\{1, 2, 3, 4, 5, 6\}$ ,  $ax$  遍历  $\{3, 6, 9, 12, 15, 18\}$ ,  $\{3, 6, 9, 12, 15, 18\}$  是  $m=7$  的一个缩系;

整数 5 与 12 互素,  $a=5, m=12$ ,  $x$  遍历  $\{1, 5, 7, 11\}$ ,  $x$  遍历  $\{5, 25, 35, 55\}$ ,  $\{5, 25, 35, 55\}$  是  $m=12$  的一个缩系。

**定理 5.10 ([2])**  $\gcd(m_1, m_2) = 1, x_1, x_2$  分别遍历  $m_1, m_2$  的简化剩余系, 则  $m_2 x_1 + m_1 x_2$  遍历模  $m_1 \times m_2$  的简化剩余系。

### 5.3.4 欧拉定理

**定理 5.11 (欧拉定理 (Euler Theorem))** 设  $m$  是大于 1 的整数, 若  $a$  是满足  $\gcd(a, m) = 1$  的整数, 则  $a^{\varphi(m)} \equiv 1 \pmod{m}$ 。

#### 例 5.12

$\varphi(6) = 2, \gcd(5, 6) = 1$ , 计算  $a^{\varphi(6)} = 5^2 = 25 \equiv 1 \pmod{6}$ 。

$\varphi(12) = 4, \gcd(5, 12) = 1$ , 计算  $a^{\varphi(12)} = 5^4 = 625 \equiv 1 \pmod{12}$ 。

## 5.4 乘法逆元

本节研究模正整数的乘法运算的可逆性问题, 先看一个例子。

模 10 的最小非负完全剩余系  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$  中, 在模 10 的运算下有:

$$1 \times 1 \equiv 1 \pmod{10}$$

$$3 \times 7 \equiv 1 \pmod{10}$$

$$9 \times 9 \equiv 1 \pmod{10}$$

也就是说当  $a \in \{1, 3, 7, 9\}$  时, 存在  $a' \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ , 使得  $aa' \equiv 1 \pmod{10}$ , 而对于  $\{0, 2, 4, 5, 6, 8\}$  集合中的数, 不具有这样的性质, 而集合  $\{1, 3, 7, 9\}$  恰好是 10 的简化剩余系。

### 5.4.1 逆元 (inverse) 定义

**定理 5.12 (逆元存在性)** 若  $a$  是满足  $\gcd(a, m) = 1$  的整数, 则存在唯一的整数  $a', 1 \leq a' < m$ , 且  $\gcd(a', m) = 1$ , 使得  $aa' \equiv 1 \pmod{m}$ 。

#### 例 5.13

$\{1, 5\}$  是  $m=6$  的一个简化剩余系;  $1 \times 1 \equiv 1 \pmod{6}, 5 \times 5 \equiv 1 \pmod{6}$

$\{1, 2, 3, 4, 5, 6\}$  是  $m=7$  的一个简化剩余系;  $1 \times 1 \equiv 1 \pmod{7}, 2 \times 4 \equiv 1 \pmod{7}, 3 \times 5 \equiv 1 \pmod{7}, 4 \times 2 \equiv 1 \pmod{7}, 5 \times 3 \equiv 1 \pmod{7}, 6 \times 6 \equiv 1 \pmod{7}$

$\{1, 5, 7, 11\}$  是  $m=12$  的一个简化剩余系;  $1 \times 1 \equiv 1 \pmod{12}, 5 \times 5 \equiv 1 \pmod{12}, 7 \times 5 \equiv 1 \pmod{12}, 7 \times 5 \equiv 1 \pmod{12}, 11 \times 1 \equiv 1 \pmod{12}$ 。

**定义 5.8 (乘法逆元 (multiplicative inverse))** 对于正整数  $m$  和整数  $a$ , 满足  $\gcd(a, m) = 1$ , 存在唯一一个  $m$  的剩余类, 其中对于每一个元素  $a'$ , 都会使  $aa' \equiv 1 \pmod{m}$  成立, 此时称  $a'$  为  $a$  模  $m$  的乘法逆元, 记作  $a' \pmod{m}$  或  $a^{-1} \pmod{m}$ 。

### 5.4.2 逆元求解

逆元的求解在公钥密码学中非常重要, 但是当  $m$  和  $a$  比较大时, 直接利用定义很难求解, 所以如何快速求解逆元, 是一个重要的研究内容。扩展欧几里得算法就是一种快

速求解逆元的算法。

**定理 5.13 (贝祖定理 (Bézout's identity))** 任何整数  $a, b$ , 一定存在整数  $x, y$ , 使  $ax + by = \gcd(a, b)$  成立。

欧几里得算法的数学基础是  $a = bq + r \Rightarrow \gcd(a, b) = \gcd(b, r), r = a - [\frac{a}{b}]b$ , 根据贝祖定理, 我们有,  $\gcd(a, b) = ax_1 + by_1, \gcd(b, r) = bx_2 + (a - [\frac{a}{b}]b)y_2$ , 对于第二个式子我们进行重新整理,  $bx_2 + (a - [\frac{a}{b}]b)y_2 = bx_2 + ay_2 - [\frac{a}{b}]by_2 = ay_2 + b(x_2 - [\frac{a}{b}]y_2)$ , 由于  $\gcd(a, b) = \gcd(b, r)$ , 所以有  $ax_1 + by_1 = ay_2 + b(x_2 - [\frac{a}{b}]y_2) \Rightarrow x_1 = y_2, y_1 = x_2 - [\frac{a}{b}]y_2$ , 一直计算下去, 就可以得出扩展欧几里得算法。

**定理 5.14 (扩展欧几里得算法) extend Euclid's Algorithm** 设  $r_0, r_1$  是两个正整数, 且  $r_0 > r_1$ , 设  $r_i (i = 1, 2, \dots, n)$  是使用欧几里得算法计算  $\gcd(r_0, r_1)$  时所得到的余数序列且  $r_{n+1} = 0$ , 则可以使用如下算法求整数  $s_n$  和  $t_n$ , 使得  $\gcd(r_0, r_1) = s_n r_0 + t_n r_1$ 。  $s_n, t_n$  是如下递归定义的序列的第  $n$  项。

$$s_0 = 1, t_0 = 0$$

$$s_1 = 0, t_1 = 1$$

$$s_i = s_{i-2} - q_{i-1}s_{i-1}, t_i = t_{i-2} - q_{i-1}t_{i-1}, \text{ 其中 } q_i = r_{i-1}/r_i, i = 2, 3, \dots, n.$$

扩展欧几里得算法是在计算贝祖 (Bézout, 也有翻译为“裴蜀”的) 等式  $ax + by = \gcd(a, b)$  的整数解, 而贝祖等式是  $a, b$  和它们的最大公约数  $d$  之间的线性丢番图方程。当  $\gcd(r_0, r_1) = 1$  时, 有  $s_n r_0 + t_n r_1 = 1$ , 等式两边同取模, 等式不变, 所以有  $s_n r_0 + t_n r_1 \equiv s_n r_0 \pmod{r_1}$ , 由逆元定义知  $r_0^{-1} = s_n \pmod{r_1}$ , 同理, 可以知道  $r_1$  的逆元,  $r_1^{-1} = t_n \pmod{r_0}$ 。

#### 例 5.14

$a=529, m=130$ , 求  $a$  模  $m$  的乘法逆元。

**解** 由前面最大公约数的计算 (见下), 可知  $a$  和  $m$  互素。

$$529 = 130 \times 4 + 9$$

$$130 = 9 \times 14 + 4$$

$$9 = 4 \times 2 + 1$$

$$4 = 1 \times 4 + 0$$

$$\gcd(529, 130) = 1$$

下面我们求解模 130 时, 529 的逆元, 或者模 529, 130 的逆元。

为了使得过程更加清晰, 我们做一张表, 展示其过程:

i	$r_i$	$q_i$	$s_i = s_{i-2} - q_{i-1}s_{i-1}$	$t_i = t_{i-2} - q_{i-1}t_{i-1}$	
0	529	-	$s_0 = 1$	$t_0 = 0$	
1	130	4	$s_1 = 0$	$t_1 = 1$	
2	9	14	$s_2 = s_0 - q_1s_1 = 1 - 4 \times 0 = 1$	$t_2 = t_0 - q_1t_1 = 0 - 4 \times 1 = -4$	529×
3	4	2	$s_3 = s_1 - q_2s_2 = 0 - 14 \times 1 = -14$	$t_3 = t_1 - q_2t_2 = 1 - 14 \times (-4) = 57$	
4	1	4	$s_4 = s_2 - q_3s_3 = 1 - 2 \times (-14) = 29$	$t_4 = t_2 - q_3t_3 = -4 - 2 \times 57 = -118$	
5	0	-	-	-	

$$29 \equiv 1(\text{mod } 130)$$

$$130 \times (-118) \equiv -528 \equiv 1(\text{mod } 529).$$

## 5.5 费马小定理

**定理 5.15 (费马小定理 (Fermat's little theorem, 1963))**  $p$  是素数,  $a$  为任意整数, 则  $a^p \equiv a(\text{mod } p)$ 。 [3]

上面的定理也有写作, 若  $p$  是素数, 则对任意整数  $a$ , 如果  $a$  和  $p$  互素, 有  $a^{p-1} \equiv 1(\text{mod } p)$ 。这种写法要求  $a$  和  $p$  互素

费马小定理是欧拉定理的一种特殊情况。

### 例 5.15

3 是素数,  $2^3 = 8 \equiv 2(\text{mod } 3), 4^3 = 64 \equiv 1 \equiv 4(\text{mod } 3), 6^3 = 216 \equiv 0 \equiv 6(\text{mod } 3);$

7 是素数,  $2^7 = 128 \equiv 2(\text{mod } 7), 4^7 = 16384 \equiv 4(\text{mod } 7), 6^7 = 279936 \equiv 6(\text{mod } 7);$

可以用费马小定理来求逆元呢。

由费马小定理  $a^{p-1} \equiv 1$ , 变形得  $aa^{p-2} \equiv 1(\text{mod } p)$ , 很明显了, 若  $a, p$  互质, 因为  $aa^{p-2} \equiv 1(\text{mod } p)$ , 则  $a^{-1} = a^{p-2}(\text{mod } p)$ , 用快速幂可快速求之。

## 5.6 威尔逊定理

威尔逊定理、欧拉定理、孙子定理 (中国剩余定理)、费马小定理并称数论四大定理。下面我们看看威尔逊定理。

**定理 5.16 (威尔逊定理 (Wilson's theorem))**  $p$  是一个素数  $\Leftrightarrow (p-1)! \equiv -1(\text{mod } p)$ 。

威尔逊定理给出了素数的充要条件, 那么我们可以用  $p-1$  的阶乘来判断  $p$  是否为素数, 但由于阶乘的计算量太大, 通常不会使用。

### 例 5.16

$$(3-1)! = 2 \equiv 2(\text{mod } 3) \equiv -1(\text{mod } 3);$$

$$(5-1)! = 24 \equiv 4(\text{mod } 5) \equiv -1(\text{mod } 5);$$

$$(7-1)! = 720 \equiv 6(\text{mod } 7) \equiv -1(\text{mod } 7);$$

$$(11-1)! = 3628800 \equiv 10(\text{mod } 11) \equiv -1(\text{mod } 11);$$

## 5.7 RSA 12 位密码系统示例

RSA 加密算法是一种非对称加密算法。在公开密钥加密和电子商业中 RSA 被广泛使用。RSA 是 1977 年由罗纳德·李维斯特 (Ron Rivest)、阿迪·萨莫尔 (Adi Shamir) 和伦纳德·阿德曼 (Leonard Adleman) 一起提出的, 后来这三个人创立了有名的安全公司 RSA(网站 [www.rsa.com](http://www.rsa.com))。当时他们三人都在麻省理工学院工作。RSA 就是他们三人姓氏开头字母拼在一起组成的。对极大整数做因数分解的难度决定了 RSA 算法的可靠性。到目前为止, 世界上还没有任何可靠的攻击 RSA 算法的方式。只要其密钥的长度足够长, 用 RSA 加密的信息实际上是不能被解破的。<sup>2</sup>


下面以 12 位演算 RSA 的加密解密与暴力破方法, 讲解 RSA 的基本原理<sup>3</sup>。

### 5.7.1 加密系统


#### (1) 生成公私钥对

RSA 是非对称加密, 有公钥和私钥, 公钥公开给加密方加密, 私钥留给自己解密, 是不公开的。

1. 随机选两个素数, 用  $p$ 、 $q$  来代替 (素数的数值越大, 位数就越多, 可靠性就越高。假设我们取  $p = 47$ ,  $q = 59$ 。

 **注意** 如何实现一个算法实现随机选两个素数?

2. 计算这两个素数的乘积,  $n = p \times q = 47 \times 59 = 2773$ ,  $n$  的长度就是公钥长度。2773 写成二进制是 101011010101, 一共有 12 位, 所以这个密钥就是 12 位。实际应用中, RSA 密钥一般是 1024 位, 重要场合则为 2048 位。

 **注意** 在实际的 RSA 系统中, 比如密钥是 1024 位, 也就是意味着要在限定一个数的位数情况下, 如何选取两个素数, 乘积满足这样的要求。

3. 计算  $n$  的欧拉函数  $\phi(n)$ ,  $\phi(n) = (p-1)(q-1)$ ,  $\phi(2773) = (47-1) \times (59-1) = 46 \times 58 = 2668$ 。

4. 随机选择一个整数  $e$ ,  $1 < e < \phi(n)$ , 且  $e$  与  $\phi(n)$  互素 (我们知道, 此时  $e^{\phi(n)} \equiv 1 \pmod{n}$ )。例如我们在 1 到 2668 之间, 随机选择了 17,  $e = 17$ 。

5. 计算  $e$  对于  $\phi(n)$  的模乘法逆元  $d$ , 当  $\gcd(e, \phi(n)) = 1$ ,  $ed \equiv 1 \pmod{\phi(n)} \Rightarrow d = (1 + k\phi(n))/e, k \in \mathbb{Z}$ , 代入各值,  $d = (1 + 2668k)/17$ , 可以依次给  $k$  赋值, 取  $d$  为整数的序偶, 得到一系列  $(k, d)$ ,  $(1, 157)$ 、 $(18, 2825)$ 、 $(35, 5493)$  ..., 随机选一个序偶, 比如  $(1, 157)$ , 也就是  $d=157$ 。

6. 将  $n$  和  $e$  封装成公钥,  $n$  和  $d$  封装成私钥, 即公钥为:  $n = 2773$ ,  $e = 17$ , 私钥为:  $n = 2773$ ,  $d = 157$ 。

#### (2) 用公钥加密字符串

<sup>2</sup>本段内容摘自 [https://baike.baidu.com/item/RSA 算法/263310?fr=aladdin](https://baike.baidu.com/item/RSA%20算法/263310?fr=aladdin), 可以从网页上获得更多的介绍信息, 并且此条信息由百度的“科普中国科学百科词条编写与应用工作项目”审核

<sup>3</sup>此节采用的例子素材来源于 <https://www.jianshu.com/p/4e302869d057>



假设我们加密一个字符"A", 首先字符要用数值表示 (这就是编码, 信源编码), 一般用 Unicode 或 ASCII 码表示, 此处我们用 ASCII 码表示, "A" 的 ASCII 码十进制为 65 (十六进制 0x41), 我们用  $m$  来代替明文 (message),  $c$  来代替密文 (cipher),  $m = 65$ , RSA 加密公式:  $m^e \equiv c \pmod{n}$ , 代入各值  $65^{17} \pmod{2773} \equiv 6,599,743,590,836,592,050,933,837,890,625 \pmod{2773} \equiv 332 \pmod{2773}$ ,  $c = 332$

(3) 用私钥解密密文

RSA 解密公式:  $c^d \equiv m \pmod{n}$ , 代入各值,  $c^d \equiv 332^{157} \equiv 6.5868707484014117339891253968203e+395 \equiv 65 \pmod{2773}$ ,  $m = 65$ .

## 5.7.2 暴力破解

RSA 的可靠性在于因数分解的难度, 因为  $p$ 、 $q$ 、 $n$ 、 $\phi(n)$ 、 $e$ 、 $d$  这六个数字之中, 公钥用到了两个  $n$  和  $e$ , 其余四个数字都是不公开的。其中最关键的是  $d$ , 因为  $n$  和  $d$  组成了私钥, 一旦  $d$  泄漏, 就等于私钥泄漏。

(1)  $ed \equiv 1 \pmod{\phi(n)}$ 。只有知道  $e$  和  $\phi(n)$ , 才能算出  $d$ 。

(2)  $\phi(n) = (p-1)(q-1)$ 。只有知道  $p$  和  $q$ , 才能算出  $\phi(n)$ 。

(3)  $n = pq$ 。只有将  $n$  因数分解, 才能算出  $p$  和  $q$ 。

由此可见推导出  $d$  就必须因数分解出  $p$  和  $q$ , 公钥里面有  $n = 2773$ , 那么暴力破解的方法就是把 2773 因数分解出两个相乘的素数。可以编程对此例进行暴力破解。

## 5.8 线性同余方程

### 5.8.1 基本概念

**定义 5.9 (同余方程)** 多项式  $f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ ,  $n > 0$ ,  $a_i$  ( $i = 0, 1, \dots, n$ ) 是整数, 设  $m > 0$ , 则同余式  $f(x) \equiv 0 \pmod{m}$  称为模  $m$  的同余方程, 若  $a_n$  不能被  $m$  整除, 则  $n$  称为  $f(x)$  的次数, 记为  $\deg f(x)$ 。若  $x_0$  满足  $f(x_0) \equiv 0 \pmod{m}$ , 则  $x \equiv x_0 \pmod{m}$  叫做此同余方程的解。不同的解是指互不同余的解。

### 5.8.2 求解方法

求解同余方程, 最直观的方法是采用遍历所有取值的方法。

#### 例 5.17

求  $x^4 + 3x^2 - 2x + 1 \equiv 0 \pmod{5}$

**解**  $x = 0, 0 + 0 - 0 + 1 = 1 \equiv 1 \pmod{5}$

$x = 1, 1 + 3 \times 1 - 2 \times 1 + 1 = 3 \equiv 3 \pmod{5}$

$x = 2, 16 + 12 - 4 + 1 = 25 \equiv 0 \pmod{5}$

$x = 3, 729 + 27 - 6 + 1 = 751 \equiv 1 \pmod{5}$

$x = 4, 256 + 48 - 8 + 1 = 297 \equiv 2 \pmod{5}$



**注意** 是不是遍历一个完全剩余系, 就算遍历了? 我们以上面为例, 先看模 5 的  $C_2 = 2, 7, 12, 17, \dots, x=7$  带入  $x^4+3x^2-2x+1$  有,  $7^4+3\times 7^2-2\times 7+1=2535 \equiv 0(\text{mod } 5)$ ,  $x=12$  带入, 有  $12^4+3\times 12^2-2\times 12+1=21145 \equiv 0(\text{mod } 5)$ , 再看  $C_0 = 0, 5, 10, 15, 20, \dots, x=5$  带入, 有  $5^4+3\times 5^2-2\times 5+1=690 \equiv 1(\text{mod } 5)$ ,  $x=10$  带入, 有  $10^4+3\times 10^2-2\times 10+1=10281 \equiv 1(\text{mod } 5)$ .

如何想证明, 我们就要看各个运算是否同余相等, 比如  $a \equiv b(\text{mod } m), c \in \mathbb{Z} \Rightarrow a+c \equiv b+c(\text{mod } m)$ .

### 例 5.18

求  $x^2+1 \equiv 0(\text{mod } 7)$

**解** 遍历后无解。

## 5.8.3 线性 (or 一次) 同余方程

**定理 5.17** 设  $m > 1, \gcd(a, m) = 1$ , 同余方程  $ax \equiv b(\text{mod } m)$  有且仅有一个解  $x_0, x_0 \equiv ba^{\varphi(m)-1}(\text{mod } m)$ .

**证明**  $1, 2, \dots, m$  组成模  $m$  的一个完全剩余系, 因为  $\gcd(a, m)=1$ , 故  $a, 2a, \dots, ma$  也组成一个模  $m$  的完全剩余系, 那么有且只有一个数  $aj$ , 满足  $aj \equiv b(\text{mod } m)$ , 所以  $x=j$  是上式的唯一解。

因为  $\gcd(a, m)=1$ , 由欧拉定理, 有  $a^{\varphi(m)} \equiv 1(\text{mod } m)$ , 根据同余的性质, 我们有  $a^{\varphi(m)}b \equiv b(\text{mod } m) \Rightarrow aa^{\varphi(m)-1}b \equiv b(\text{mod } m) \Rightarrow x = a^{\varphi(m)-1}b$  是上式同余方程唯一解。

### 例 5.19

求  $5x \equiv 3(\text{mod } 6)$  的解。

**解**

已知  $\gcd(5, 6) = 1, \varphi(6) = 2$ , 由以上定理知其有且只有一个解  $x \equiv 3 \cdot 5 \equiv 15(\text{mod } 6) \equiv 3(\text{mod } 6)$ .

**定理 5.18 ([3])** 设  $m > 1, \gcd(a, m) = d > 1$ , 同余方程  $ax \equiv b(\text{mod } m)$  有解的充要条件是  $d \mid b$ , 并且在此同余方程有解时, 其解的个数是  $d$ , 若  $x \equiv x_0(\text{mod } m)$  是此方程的特解, 则它的  $d$  个解为  $x \equiv x_0 + \frac{m}{d}t (\text{mod } m) t = 0, 1, \dots, d-1$ 。

### 例 5.20

求解  $28x \equiv 21(\text{mod } 35)$

**解**  $\gcd(28, 35) = 7$ , 且  $7 \mid 21$ , 故此方程有解。

$4x \equiv 3(\text{mod } 5), x_0 = 2$  是一个特解。

$x \equiv 2 + \frac{35}{7}t, t = 0, 1, 2, \dots, 6$ , 可求得解为 2, 7, 12, 17, 22, 17, 32.

## 5.9 同余方程组的求解

### 5.9.1 中国剩余定理

我国古代的一部数学著作《孙子算经》中, 有一类叫做“物不知数”的问题, 原文为: 今有物不知其数 (四声), 三三数 (三声) 之剩二, 五五数之剩三, 七七数之剩二,

问物几何?

这个问题就是求解同余方程组:

$$\begin{cases} x \equiv 2 \pmod{3} \\ x \equiv 3 \pmod{5} \\ x \equiv 2 \pmod{7} \end{cases}$$

我国明代数学家程大位在《算法统宗》这部著作中,把此题解法用一首优美的诗来总结:

三人同行七十稀, 五树梅花廿一枝,  
七子团圆整半月, 除百零五便得知。

这首诗翻译为现代汉语的意识就是,此未知数除 3 所得余数乘 70, 除 5 所得余数乘 21, 除 7 所得余数乘 15, 然后相加除 105 便是此未知数的取值。用算式表示为:

$$(2 \times 70 + 3 \times 21 + 2 \times 15) \equiv 23 \pmod{105}$$

计算知此数为 23。

由上计算方法可见, 2, 3, 2 为余, 7, 21, 15 分别为模 7, 模 3×模 7, 模 3×模 5, 是否有规律可循, 对于此思路的详细阐述, 见 [3] 中相应部分的描述 (第 71 页)。

**定理 5.19 (中国剩余定理 (chinese remainder theorem, CRT)[3])** 设  $m_1, m_2, \dots, m_k$  是  $k$  个两两互素的正整数, 若令  $m = m_1 m_2 \dots m_k$ ,  $M_i = m_1 m_2 \dots m_{i-1} m_{i+1} \dots m_k$ ,  $m_i = m_i M_i$ , 则对于任意的整数  $b_1, b_2, \dots, b_k$ , 同余方程组

$$\begin{cases} x \equiv b_1 \pmod{m_1} \\ x \equiv b_2 \pmod{m_2} \\ \dots \\ x \equiv b_k \pmod{m_k} \end{cases}$$

有唯一解  $x \equiv M'_1 M_1 b_1 + M'_2 M_2 b_2 + \dots + M'_k M_k b_k \pmod{m}$ , 其中  $M'_i$  为  $M_i$  的逆元,  $M'_i M_i \equiv 1 \pmod{m_i}$ ,  $i = 1, 2, \dots, k$ 。

### 例 5.21

韩信点兵, 有兵一队, 若列五行纵队, 末行一人, 列六行纵队, 末列五人, 列七行纵队, 末行四人, 列十一行纵队, 末行十人, 求兵数。

**解** 转化为同余方程组:

$$\begin{cases} x \equiv 1 \pmod{5} \\ x \equiv 5 \pmod{6} \\ x \equiv 4 \pmod{7} \\ x \equiv 10 \pmod{11} \end{cases}$$

应用 CRT,  $m = 5 \times 6 \times 7 \times 11 = 2310$ 。

$$M_1 = 2310/5 = 462, M_1^{-1} \equiv 3 \pmod{5}$$

$$M_2 = 2310/6 = 385, M_2^{-1} \equiv 1 \pmod{6}$$

$$M_3 = 2310/7 = 330, M_3^{-1} \equiv 1 \pmod{7}$$

$$M_4 = 2310/11 = 210, M_4^{-1} \equiv 1 \pmod{11}$$

$$x \equiv 462 \cdot 3 \cdot 1 + 385 \cdot 1 \cdot 5 + 330 \cdot 1 \cdot 4 + 210 \cdot 1 \cdot 10 \equiv 6731 \equiv 2111 \pmod{2310}$$

### 5.9.2 一般一次同余方程组的解

在中国剩余定理中, 要求  $m_1, m_2, \dots, m_k$  两两互素, 如果不互素, 如何求解同余方程组的解? 我们看下面的定理。

**定理 5.20** 同余方程组 
$$\begin{cases} x \equiv b_1 \pmod{m_1} \\ x \equiv b_2 \pmod{m_2} \end{cases}$$
 有解的充要条件是  $\gcd(m_1, m_2) \mid (b_1 - b_2)$ 。如

果这个条件成立, 则此同余方程组模  $\text{lcm}(m_1, m_2)$  有唯一解。[3]

对于一次同余方程组

$$\begin{cases} x \equiv b_1 \pmod{m_1} \\ x \equiv b_1 \pmod{m_1} \\ \dots \\ x \equiv b_k \pmod{m_k} \end{cases}$$

$k \geq 3$ , 若  $\gcd(m_1, m_2) \mid (b_1 - b_2)$ , 可先求得前两个方程解  $x \equiv b'_2 \pmod{\text{lcm}(m_1, m_2)}$ 。若  $\gcd(\text{lcm}(m_1, m_2), m_3) \mid (b'_2 - b_3)$ , 则  $x \equiv b'_2 \pmod{\text{lcm}(m_1, m_2)}$  与  $x \equiv b_3 \pmod{m_3}$  进行联立求解, 以此类推, 可以求得最后的解。

**例 5.22** [3] 判断以下方程是否有解

$$\begin{cases} x \equiv 11 \pmod{36} \\ x \equiv 7 \pmod{40} \\ x \equiv 32 \pmod{75} \end{cases}$$

**解**  $\gcd(36, 40) = 4, \gcd(36, 75) = 3, \gcd(40, 75) = 5$

$$b_1 - b_2 = 11 - 7 = 4, b_1 - b_3 = 11 - 32 = -21, b_2 - b_3 = 7 - 32 = -25$$

因为  $4 \mid 4, 3 \mid -21, 5 \mid -25$ , 所以此方程组一定有解, 且解的模数为  $\text{lcm}(36, 40, 75) = 1800$ 。

依次联立解方程组, 最后可得  $x \equiv 407 \pmod{1800}$ 。

## 5.10 高次同余方程的解

本节只是初步讨论一下高次同余方程的解的情况。

**定理 5.21 ([2]<sup>4</sup>)** 若  $m_1, m_2, \dots, m_k$  是  $k$  个两两互质的正整数,  $m = m_1 m_2 \dots m_k$ , 则同余方程  $f(x) \equiv 0 \pmod{m}$  与同余方程组  $f(x) \equiv 0 \pmod{m_i}, i = 1, 2, \dots, k$  等价。用  $T_i$  表示方程  $f(x) \equiv 0 \pmod{m_i}, i \in \{1, 2, \dots, k\}$  的解数,  $T$  表示  $f(x) \equiv 0 \pmod{m}$  的解数, 则有  $T = T_1 \times T_2 \times \dots \times T_k$ .

**例 5.23** 求解同余式  $x^4 + 2x^3 + 8x + 9 \equiv 0 \pmod{35}$ . [2]

**解**

$35 = 5 \times 7$ , 且  $\gcd(5, 7) = 1$ , 由此可知题中方程与以下方程组等价:

$$\begin{cases} x^4 + 2x^3 + 8x + 9 \equiv 0 \pmod{5} \\ x^4 + 2x^3 + 8x + 9 \equiv 0 \pmod{7} \end{cases}$$

采用遍历的方法可知方程组中第一个方程有两个解  $x \equiv 1, 4 \pmod{5}$ , 第二个方程有三个解  $x \equiv 3, 5, 6 \pmod{7}$ , 根据上述定理, 题中同余方程解有  $2 \times 3 = 6$  个解。

下面我们看看如何求出题中方程的解。

方程等价于以下方程组:

$$\begin{cases} x \equiv 1 \pmod{5} \\ x \equiv 4 \pmod{5} \\ x \equiv 3 \pmod{7} \\ x \equiv 5 \pmod{7} \\ x \equiv 6 \pmod{7} \end{cases}$$

我们有  $7 \times 3 \equiv 1 \pmod{5}, 5 \times 3 \equiv 1 \pmod{7}$  利用中国剩余定理有  $x \equiv 21b_1 + 15b_2, b_1 \in \{1, 4\}, b_2 \in \{3, 5, 6\}$ , 计算所有组合, (例如,  $b_1 = 1, b_2 = 3, x = 21 \times 1 + 15 \times 3 = 66 \equiv 31 \pmod{35}$ ), 可得题中方程全部解为  $x \equiv 31, 26, 6, 24, 19, 34 \pmod{35}$ .

## 5.11 习题

### Exercise 16

设模  $m=16$ , 求解  $1, 9, 16, 17, 25, 160$  模 16 余数, 并找出同余的数来。

### Exercise 17

求  $7^{2046}$  写成十进制数时的个位数。

### Exercise 18

求  $2^{1000}$  的十进制表示中的末尾两位数字。

<sup>4</sup>为了便于初学者理解, 在闵嗣鹤先生的《初等数论》一书中此定理的描述上做了一点修改

**Exercise 19**

已知 2019 年 9 月 29 日是星期天, 问之后的  $2^{100}$  天是星期几? 第  $2^{200}$  天呢?

**Exercise 20**

求  $1^5 + 2^5 + 3^5 + \dots + 99^5$  之和被 4 除的余数。

**Exercise 21**

写出模 9 的一个完全剩余系, 它的每个数都是奇数。

**Exercise 22**

写出模 9 的一个完全剩余系, 它的每个数都是偶数。

**Exercise 23**

用模 5 和模 6 的完全剩余系, 表示模 30 的完全剩余系。

**Exercise 24**

写出 12 的最小正缩系。

**Exercise 25**

用模 5 和模 6 的缩系, 表示模 30 的缩系。

**Exercise 26**

计算以下整数的欧拉函数。

(1)24      (2)64      (3)187      (4)360

**Exercise 27**

计算  $8 \times 9 \times 10 \times 11 \times 12 \times 13 \pmod{7}$ .

**Exercise 28**

求  $229^{-1} \pmod{281}$

**Exercise 29**

求  $3169^{-1} \pmod{3571}$ .

**Exercise 30**

解方程  $105x + 121y = 1 (x, y \in \mathbb{Z})$ .

**Exercise 31**

求解一次同余方程  $27x \equiv 12 \pmod{15}$

**Exercise 32**

求解一次同余方程  $24x \equiv 6 \pmod{81}$

**Exercise 33**

求解一次同余方程  $91x \equiv 26 \pmod{169}$

**Exercise 34**

确定以下同余式不同解的个数，无须求出具体的解。

1.  $72x \equiv 47 \pmod{200}$
2.  $4183x \equiv 5781 \pmod{15087}$
3.  $1537x \equiv 2863 \pmod{6731}$

**Exercise 35**

求解同余方程组：
$$\begin{cases} x \equiv 9 \pmod{12} \\ x \equiv 6 \pmod{25} \end{cases}$$

**Exercise 36**

求解同余方程组：
$$\begin{cases} x \equiv 5 \pmod{7} \\ x \equiv 12 \pmod{15} \\ x \equiv 18 \pmod{22} \end{cases}$$

**Exercise 37**

求解同余方程组：
$$\begin{cases} x \equiv 5 \pmod{9} \\ 3x \equiv 12 \pmod{5} \\ 4x \equiv 18 \pmod{7} \end{cases}$$

**Exercise 38**

有总数不满 50 人的一队士兵，一至三报数，最后一人报一，一至五报数，最后一人报二，一至七报数，最后一人报二，这支队伍共有士兵多少人。

## 第 6 章 二次剩余 (Quadratic residue)

中学学过的一元二次方程理论，讨论了实系数的一元二次方程的根如何求解。但到目前为止，人们还没有找到一般方法求解一般的多项式同余方程。除了求根问题，还有一个相关问题就是判断是否有解。二次同余方程有着较多的研究成果，这就是本节所涉及到的核心内容。

### 6.1 二次同余方程

二次同余方程的一般表达式为： $ax^2 + bx + c \equiv 0 \pmod{m}, a \not\equiv 0 \pmod{m}$ ，下面我们讨论一般二次同余方程解的问题，讨论过程主要参考 [2](第 73 页)。

对于一个二次同余方程，有可能有解，有可能无解，例如  $x^2 - 3 \equiv 0 \pmod{7}$  就无解，所以首先讨论方程什么时候有解。

设  $m$  的标准分解式是  $m = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ ，根据定理 5.21，二次同余方程等价于方程组  $f(x) \equiv 0 \pmod{p_i^{\alpha_i}}, i = 1, 2, \dots, k$ 。

对于方程  $f(x) \equiv 0 \pmod{p_i^{\alpha_i}}, i \in \{1, 2, \dots, k\}$ ，根据同余的性质 ( $a \equiv b \pmod{m}, d \mid m, d > 0 \Rightarrow a \equiv b \pmod{d}$ )，等价于  $f(x) \equiv 0 \pmod{p_i}, i \in \{1, 2, \dots, k\}$ ，我们将问题转换为考虑此种形式 ( $ax^2 + bx + c \equiv 0 \pmod{p}, p$  为素数) 的二次同余方程的解。下面我们对此种形式进行讨论。

1. 若  $p \mid \gcd(a, b, c)$ ，可知  $x$  的任何取值都满足方程。
2. 若  $p \nmid \gcd(a, b, c)$ 
  - (a). 若  $p \mid a, p \mid b$ ，则  $p \nmid c, ax^2 + bx + c \equiv 0 \pmod{p}$  无解。
  - (b). 若  $p \mid a, p \nmid b$ ，同余方程等价于  $bx + c \equiv 0 \pmod{p}$ ，因为  $\gcd(b, p) = 1$ ，所以此方程一定有解。具体解法参见前面线性同余方程的求解方法。
  - (c). 若  $p \nmid a, p > 2$ ，则  $\gcd(4a, p) = 1$ <sup>1</sup>， $ax^2 + bx + c \equiv 0 \pmod{p}$  两边同乘  $4a$ ，配方后得  $(2ax + b)^2 - (b^2 - 4ac) \equiv 0 \pmod{p}$ ，设  $y \equiv 2ax + b \pmod{p}, d \equiv b^2 - 4ac \pmod{p}$ ，原方程可写为  $y^2 \equiv d \pmod{p}$ ，如果  $y_0$  是其一个解，则  $y_0 \equiv 2ax + b \pmod{p}, x_0 \equiv (2a)^{-1}(y_0 - b) \pmod{p}$  为原方程的一个解。

通过以上讨论，我们可以得出，对于一般的二次同余方程，我们都可以转化为讨论形如  $x^2 \equiv a \pmod{m}, \gcd(a, m) = 1$  的二次同余方程的讨论。

**例 6.1** 求解  $5x^2 - 6x + 2 \equiv 0 \pmod{13}$ 。

**解**  $\gcd(20, 13) = 1, a = 5, b = -6, c = 2$

$$(2ax + b)^2 \equiv b^2 - 4ac \pmod{p}$$

$$y = 2ax + b = 10x - 6$$

$$d = b^2 - 4ac = 36 - 4 \times 5 \times 2 \equiv -4$$

<sup>1</sup> $p > 2, p$  为素数， $a$  和  $p$  互素，如果  $4a$  和  $p$  不互素，那么  $4$  和  $p$  一定不互素，显然不成立，所以  $4a$  和  $p$  互素。



$y^2 \equiv -4 \equiv 9 \pmod{13} \Rightarrow y \equiv 3 \text{ or } 10 \pmod{13}$ , 我们可得

$10x - 6 \equiv 3 \pmod{13}$  or  $10x - 6 \equiv 10 \pmod{13}$ , 这两个方程可以用前面介绍的方法, 也可以用 10 的逆元为 4 来求解:

$$10x \times 4 \equiv 9 \times 4 \pmod{13} \Rightarrow x \equiv 36 \pmod{13} \equiv 10 \pmod{13}$$

$$10x \times 4 \equiv 16 \times 4 \pmod{13} \Rightarrow x \equiv 36 \pmod{13} \equiv 12 \pmod{13}$$

**定义 6.1 (二次剩余 (quadratic residue))** 设  $m$  为正整数, 若同余式  $x^2 \equiv a \pmod{m}$ ,  $\gcd(a, m) = 1$ , 有解, 则称  $a$  为模  $m$  的平方剩余, 也叫二次剩余; 否则称  $a$  为模  $m$  的二次非剩余或平方非剩余。

### 例 6.2

求模 7 的平方剩余。

解

$$1^2 \equiv 1 \pmod{7}, 2^2 \equiv 4 \pmod{7}, 3^2 \equiv 2 \pmod{7}, 4^2 \equiv 2 \pmod{7}, 5^2 \equiv 4 \pmod{7}, 6^2 \equiv 1 \pmod{7}, 7^2 \equiv 0 \pmod{7}, 8^2 \equiv 1 \pmod{7}, 9^2 \equiv 4 \pmod{7}, 10^2 \equiv 2 \pmod{7}, 11^2 \equiv 2 \pmod{7}$$

计算所得余数中与 7 互素的有 1, 4, 2, 可得模 7 的二次剩余是 1, 2, 4。

## 6.2 欧拉判别条件

有时候我们往往需要判断二次同余方程是否有解, 欧拉判别条件就是解决这个判定的。

**定理 6.1 (欧拉判别条件 (Euler's Criterion)[3])** 设  $p$  是奇素数,  $\gcd(a, p) = 1$ , 则:

(1)  $a$  是模  $p$  的二次剩余的充要条件是  $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$ ;

(2)  $a$  是模  $p$  的二次非剩余的充要条件是  $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$ ;

并且当  $a$  是模  $p$  的二次剩余时, 同余方程  $x^2 \equiv a \pmod{p}$  恰有两个解。

### 例 6.3

$i = 1, 2, \dots, 13$ , 我们计算  $i^2 \pmod{13}$ , 但是要注意在二次剩余的定义中互素的条件。然后我们可以检验一下上面的欧拉判别条件的正确性。

### 例 6.4

8 是不是模 17 的平方剩余?

解

$\gcd(8, 17) = 1$ , 17 为奇素数

$$8^{(17-1)/2} = 8^8 = 16777216 \equiv 1 \pmod{17}, \text{ 因此 } 8 \text{ 是模 } 17 \text{ 的平方剩余。}$$

### 例 6.5

137 是不是模 227 的平方剩余?

解

227 是奇素数,  $\gcd(137, 227)=1$

$$137^{(227-1)/2} = 137^{113} \pmod{227}$$

$$\therefore 137^2 = 18769 \pmod{227} \equiv 155 \pmod{227}$$

$$\therefore 137^{113} \pmod{227} \equiv 155^{56} \cdot 137 \pmod{227}$$

$$\therefore 155^2 = 24025 \pmod{227} \equiv 190 \pmod{227}$$

$$\therefore 137^{113} \pmod{227} \equiv 190^{28} \cdot 137 \pmod{227}$$

$$\therefore 190^2 = 36100 \pmod{227} \equiv 7 \pmod{227}$$

$$\therefore 137^{113} \pmod{227} \equiv 7^{14} \cdot 137 \pmod{227}$$

$$\therefore 7^{14} = 49^7 = 49^6 \cdot 49 = 131^3 \cdot 49 \equiv 136 \cdot 131 \cdot 49 \pmod{227}$$

$$\therefore 137^{113} \equiv 136 \cdot 131 \cdot 49 \cdot 137 \pmod{227}$$

$$136 \cdot 131 \equiv 17816 \equiv 110 \pmod{227}$$

$$110 \cdot 49 = 5390 \equiv 169 \pmod{227}$$

$$169 \cdot 137 = 23153 \equiv 226 \equiv -1 \pmod{227}$$

由于余数为-1, 故 137 是模 227 的平方非剩余。

我们也可以用 SageMath 编写程序来判断 137 是否是模 227 的二次剩余。

#### sagemath: 判断二次剩余

```
sage: if mod(227,2)==0:
.....: print "227 is odd"
.....: elif is_prime(227):
.....: print "227 is prime"
.....: if mod(137^((227-1)/2),227)==1:
.....:   print "137 is quadratic residue of module 227"
.....: else:
.....:   print "137 is not quadratic residue of module 227"
.....: else:
.....:   print "227 is not prime"
.....:
227 is prime
137 is not quadratic residue of module 227
sage:
```

## 6.3 勒让德符号

从上面的计算我们基本可以看出, 当  $p$  比较大时, 用欧拉判定方法来判定一个数是否为  $p$  的二次剩余时, 计算量很大。引入 Legendre 符号, 可以给出一种更有效的判定方法。

### 6.3.1 概念

**定义 6.2 (勒让德符号 (Legendre symbol)[3])**  $p$  是奇素数,  $\gcd(a, p) = 1$ , 定义勒让德 (legendre) 符号为:

$$\left[ \frac{a}{p} \right] = \begin{cases} 1, & \text{若 } a \text{ 是模 } p \text{ 的二次剩余} \\ -1, & \text{若 } a \text{ 是模 } p \text{ 的二次非剩余} \end{cases} \quad (6.1)$$

下面给出 [4] 中的 Legendre 符号定义的方法, 请注意比较。

**定义 6.3 (勒让德符号 (Legendre symbol))**  $p$  是奇素数,  $a$  是整数, 定义勒让德 (legendre) 符号为:

$$\left[ \frac{a}{p} \right] = \begin{cases} 1, & \text{若 } a \text{ 是模 } p \text{ 的二次剩余} \\ -1, & \text{若 } a \text{ 是模 } p \text{ 的二次非剩余} \\ 0, & \text{若 } p|a \end{cases} \quad (6.2)$$

#### 例 6.6

求 13 的 Legendre 符号。

**解**

可以通过计算得知 1, 3, 4, 9, 10, 12 是模 13 的二次剩余, 2, 5, 6, 7, 8, 11 为模 13 的二次非剩余。所以我们有:

$$\left[ \frac{1}{13} \right] = \left[ \frac{3}{13} \right] = \left[ \frac{4}{13} \right] = \left[ \frac{9}{13} \right] = \left[ \frac{10}{13} \right] = \left[ \frac{12}{13} \right] = 1 \quad (6.3)$$

$$\left[ \frac{2}{13} \right] = \left[ \frac{5}{13} \right] = \left[ \frac{6}{13} \right] = \left[ \frac{7}{13} \right] = \left[ \frac{8}{13} \right] = \left[ \frac{11}{13} \right] = -1 \quad (6.4)$$

利用勒让德符号, 我们可以将欧拉判别条件利用勒让德符号进行改写。

**定理 6.2 (欧拉判别条件的勒让德符号表示 [3])** 设  $p$  是奇素数,  $a$  是与  $p$  互素的整数, 则  $\left[ \frac{a}{p} \right] \equiv a^{\frac{p-1}{2}} \pmod{p}$ . 显然  $\left[ \frac{1}{p} \right] = 1$ .

### 6.3.2 勒让德符号的运算律

**定理 6.3 (Legendre 符号的性质 [3])** 设  $p$  是奇素数,  $a, b$  都是与  $p$  互素的整数, 则有:

$$(1) \text{ 若 } a \equiv b \pmod{p} \Rightarrow \left[ \frac{a}{p} \right] = \left[ \frac{b}{p} \right]$$

$$(2) \left[ \frac{ab}{p} \right] = \left[ \frac{a}{p} \right] \left[ \frac{b}{p} \right]$$

$$(3) \left[ \frac{a^2}{p} \right] = 1$$

**定理 6.4 ([3])** 设  $p$  是奇素数, 我们有  $\left[ \frac{-1}{p} \right] = (-1)^{\frac{p-1}{2}} = \begin{cases} 1, & p \equiv 1 \pmod{4} \\ -1, & p \equiv 3 \pmod{4} \end{cases}$ .

**定理 6.5 ([4])** 设  $p$  是奇素数, 则  $\left[\frac{2}{p}\right] = (-1)^{\frac{p^2-1}{8}}$ .

### 6.3.3 勒让德符号的计算

**定理 6.6 (二次剩余的高斯引理 [3])**  $p$  是奇素数,  $a$  是与  $p$  互素的整数, 如果  $\frac{p-1}{2}$  个整数  $a \times 1, a \times 2, a \times 3, \dots, a \times \frac{p-1}{2}$  模  $p$  后得到的最小正剩余中大于  $\frac{p}{2}$  的个数是  $m$ , 则  $\left[\frac{a}{p}\right] = (-1)^m$

**例 6.7 [3]**

利用高斯引理判断 5 是否为模 13 的二次剩余。

**解**

13 是奇素数, 5 和 13 互素,  $(13-1)/2 = 6$ , 可得整数序列 5, 10, 15, 20, 25, 30, 这些数模 13 后得: 5, 10, 2, 7, 12, 4, 其中大于  $13/2$  的数有 10, 7, 12, 共 3 个 ( $m=3$ ), 那么根据高斯引理有:

$$\left[\frac{5}{13}\right] = (-1)^3 = -1$$

5 是模 13 的二次非剩余。

**定理 6.7 (二次互反率 (quadratic reciprocity law) [3])**  $p, q$  是奇素数,  $p \neq q$ , 则  $\left[\frac{p}{q}\right] \left[\frac{q}{p}\right] = (-1)^{\frac{p-1}{2} \frac{q-1}{2}}$ 。

在实际应用中, 我们有时也把二次互反率写为:  $\left[\frac{q}{p}\right] = (-1)^{\frac{p-1}{2} \frac{q-1}{2}} \left[\frac{p}{q}\right]$

**例 6.8 [3]**

3 是否为模 17 的二次剩余。

**解**

由二次互反率, 有:

$$\left[\frac{3}{17}\right] = (-1)^{\frac{3-1}{2} \frac{17-1}{2}} \left[\frac{17}{3}\right] = \left[\frac{17}{3}\right]$$

$$\because 17 = 6 \times 3 - 1 \equiv -1 \pmod{3}$$

$$\therefore \left[\frac{17}{3}\right] = \left[\frac{-1}{3}\right]$$

根据定理 6.4, 我们有:

$$\left[\frac{-1}{3}\right] = (-1)^{\frac{3-1}{2}} = -1$$

故 3 是模 17 的二次非剩余。

**例 6.9 [4]**

同余方程  $x^2 \equiv 137 \pmod{227}$  是否有解?

**解** 227 是素数, 则:

$$\left[\frac{137}{227}\right] = \left[\frac{-90}{227}\right] = \left[\frac{-1}{227}\right] \left[\frac{2 \cdot 3^2 \cdot 5}{227}\right] = - \left[\frac{2}{227}\right] \left[\frac{5}{227}\right]$$

$$\text{根据定理 6.5, 有 } \left[\frac{2}{227}\right] = (-1)^{\frac{227^2-1}{8}} = (-1)^{\frac{226 \cdot 228}{8}} = -1$$

$$\text{根据二次互反律有, } \left[\frac{5}{227}\right] = (-1)^{\frac{5-1}{2} \frac{227-1}{2}} \left[\frac{227}{5}\right] = \left[\frac{227}{5}\right] = \left[\frac{2}{5}\right] = (-1)^{\frac{5^2-1}{8}} = -1$$

因此,  $\left[\frac{137}{227}\right] = -1$ , 也就是说原同余方程无解。

## 6.4 雅可比符号 (Jacobi Symbol)

我们在利用勒让德符号判断二次同余方程是否有解时，通常需要将符号上方的数分解成标准分解式，而把一数分解成标准分解式是没有什么一般方法的，因此，利用勒让德符号来判断实际计算时，还有一定程度的缺点。引入雅可比符号就是为了去掉这一缺点。[2]

### 6.4.1 概念

**定义 6.4 (雅可比符号 (Jacobi Symbol))** 正奇数  $m = p_1 p_2 \dots p_r$  是奇素数  $p_i (i = 1, 2, \dots, r)$  的乘积，定义雅可比符号为  $\left(\frac{a}{m}\right) = \left[\frac{a}{p_1}\right] \left[\frac{a}{p_2}\right] \dots \left[\frac{a}{p_r}\right]$ 。

雅可比符号是用勒让德符号定义的，但是，雅可比符号将 Legendre 符号从奇素数的限制，扩大到正奇数。“扩大后其意义也发生了变化，如果  $a$  对  $p$  的勒让德符号为 1，可知  $a$  是模  $p$  的二次剩余，但当  $a$  对  $m$  的雅可比符号为 1 时，却不能得到  $a$  是模  $m$  的二次剩余这个结论。

雅可比符号的好处就是它一方面具有很多与勒让德符号一样的性质，当  $r=1$  是，雅可比符号和勒让德符号相等，另一方面，他并没有限制  $m$  必需是素数，因此要想计算勒让德符号的值，只须把他看成雅可比符号来计算。在计算雅可比符号值时，由于不用考虑  $m$  是不是素数，所以在实际计算上就非常方便了，并且利用雅可比符号最后一定能把勒让德符号的值计算出来。” [2](第 88 页)

## 6.5 习题

### Exercise 39

求模 23 的二次剩余和非二次剩余

### Exercise 40

求满足方程  $E: y^2 \equiv x^2 - 2x + 1 \pmod{7}$  的所有点。

### Exercise 41

利用欧拉判别条件判断 2 是否是 29 的二次剩余。

### Exercise 42

利用勒让德符号判断 2 是否是 73 的二次剩余。

### Exercise 43

计算勒让德符号  $\left(\frac{17}{37}\right)$ 。

### Exercise 44

计算勒让德符号  $\left(\frac{37}{25411}\right)$  (备注: 25411 为素数)

## 第7章 原根与指数 (primitive root and index)

本章讨论同余方程  $x^n \equiv a \pmod{m}$  在什么条件下有解，在讨论过程中引入原根和指数的概念，通过讨论，对某些特殊的  $m$  有解的条件利用指数表达出来。[2]

### 7.1 次数 (order)

#### 7.1.1 基本概念

**定义 7.1 (次数 (order))**  $m$  是大于 1 的整数， $a$  是与  $m$  互素的整数，使  $a^l \equiv 1 \pmod{m}$  成立的最小正整数  $l$  叫做  $a$  对模  $m$  的次数，记作  $\text{ord}_m(a)$  或  $\delta_m(a)$ 。

对于任意大于 1 的整数  $m$ ，我们可以知道  $\text{ord}_m(1) = 1, \text{ord}_m(-1) = 2$ 。

#### 例 7.1

求  $\text{ord}_{11}(a)$ ，其中  $a = 1, 2, 3, \dots, 10$ 。

**解** 我们可以把所有可能列出来，从而进行解答。

	a=1	a=2	a=3	a=4	a=5	a=6	a=7	a=8	a=9	a=10
$a^1 \pmod{11}$	1	2	3	4	5	6	7	8	9	10
$a^2 \pmod{11}$	1	4	9	5	3	3	5	9	4	1
$a^3 \pmod{11}$	1	8	5	9	4	7	2	6	3	
$a^4 \pmod{11}$	1	5	4	3	9	9	3	4	5	
$a^5 \pmod{11}$	1	10	1	1	1	10	10	10	1	
$a^6 \pmod{11}$	1	9				5	4	3		
$a^7 \pmod{11}$	1	7				8	6	2		
$a^8 \pmod{11}$	1	3				4	9	5		
$a^9 \pmod{11}$	1	6				2	8	7		
$a^{10} \pmod{11}$	1	1				1	1	1		

由上表可知  $\text{ord}_{11}(1) = 1, \text{ord}_{11}(2) = 10, \text{ord}_{11}(3) = 5, \text{ord}_{11}(4) = 5, \text{ord}_{11}(5) = 5, \text{ord}_{11}(6) = 10, \text{ord}_{11}(7) = 10, \text{ord}_{11}(8) = 10, \text{ord}_{11}(9) = 5, \text{ord}_{11}(10) = 2$ 。

在次数定义时，只考虑与  $m$  互素的整数  $a$ ，如果  $a$  和  $m$  不互素，不可能存在一个正整数  $l$ ，使得  $a^l \equiv 1 \pmod{m} (l \geq 1)$ ，所以通常只要谈到  $a$  对模  $m$  的次数，隐含条件都是  $a$  和  $m$  互素。

sagemath:  $a$  与  $m$  不互素时, 不存在次数

```
sage: a=5
.....: m=15
.....: for l in range(20):
.....:     print a,"^",l,"=",mod(a^l,m),"(mod ",m,")"
.....:
5 ^ 0 = 1 (mod 15 )
5 ^ 1 = 5 (mod 15 )
5 ^ 2 = 10 (mod 15 )
5 ^ 3 = 5 (mod 15 )
5 ^ 4 = 10 (mod 15 )
5 ^ 5 = 5 (mod 15 )
5 ^ 6 = 10 (mod 15 )
5 ^ 7 = 5 (mod 15 )
5 ^ 8 = 10 (mod 15 )
5 ^ 9 = 5 (mod 15 )
5 ^ 10 = 10 (mod 15 )
5 ^ 11 = 5 (mod 15 )
5 ^ 12 = 10 (mod 15 )
5 ^ 13 = 5 (mod 15 )
5 ^ 14 = 10 (mod 15 )
5 ^ 15 = 5 (mod 15 )
5 ^ 16 = 10 (mod 15 )
5 ^ 17 = 5 (mod 15 )
5 ^ 18 = 10 (mod 15 )
5 ^ 19 = 5 (mod 15 )
sage:
```

### 7.1.2 次数性质和计算

那么如何计算某个整数对某个模的次数呢?

**定理 7.1**  $n$  为非负整数,  $a^n \equiv 1 \pmod{m} \Leftrightarrow \text{ord}_m(a) \mid n$

**定理 7.2 ([2])**  $a^0, a^1, \dots, a^{\text{ord}_m(a)-1}$  对模  $m$  两两不同余, 其中  $a^0 = 1$ .

**定理 7.3 ([2])** 若  $a$  对模  $m$  的指数是  $\delta$ , 则  $a^\gamma \equiv a^{\gamma'} \pmod{m}$  成立的充要条件是  $\gamma \equiv \gamma' \pmod{\delta}$ , 特别地,  $a^\gamma \equiv 1 \pmod{m} \Leftrightarrow \delta \mid \gamma$ .

**定理 7.4 (次数性质 [3])** 设  $a$  对模  $m$  的次数是  $\text{ord}_m(a)$ , 则有:

(1)  $\text{ord}_m(a) \mid \varphi(m)$ ;

$$(2) b \equiv a \pmod{m} \Rightarrow \text{ord}_m(b) = \text{ord}_m(a).$$

利用上面的次数性质，可以再求次数时减少计算量。

### 例 7.2

计算 5 对模 17 的次数。

解

由于  $\varphi(17) = 16$ ，根据次数性质 7.4，次数可以被  $\varphi(17)$  整除 (也就是 16 的因子)，而 16 的因子有 1, 2, 4, 8, 16，所以只需计算 5 的 1, 2, 4, 8, 16 次方：

$$5^1 \equiv 5 \pmod{17}$$

$$5^2 \equiv 10 \pmod{17}$$

$$5^4 \equiv 13 \pmod{17}$$

$$5^8 \equiv 16 \pmod{17}$$

$$5^{16} \equiv 1 \pmod{17}$$

可见  $\text{ord}_{17}(5) = 16$ .

**定理 7.5 (幂的周期性 [3])**  $s, t$  是任意非负整数， $a^s \equiv a^t \pmod{m} \Leftrightarrow s \equiv t \pmod{\text{ord}_m(a)}$

以上定理揭示了一个事实，当  $m > 1, \gcd(a, m) = 1$ ，序列  $a^i \pmod{m}$  ( $i = 1, 2, 3, \dots$ ) 是周期序列，周期为  $\text{ord}_m(a)$ 。

### 例 7.3

我们先看  $5^x \pmod{12}$  序列的周期性，我们知道  $\text{ord}_{12}(5) = 2$ ：

#### sagemath: 幂的周期性示例

```
sage: a=5
.....: m=12
.....: for l in range(50):
.....:     ^Iprint a,"^",l,"=",mod(a^l,m),"(mod ",m,")"
.....:
5 ^ 0 = 1 (mod 12 )
5 ^ 1 = 5 (mod 12 )
5 ^ 2 = 1 (mod 12 )
5 ^ 3 = 5 (mod 12 )
5 ^ 4 = 1 (mod 12 )
5 ^ 5 = 5 (mod 12 )
5 ^ 6 = 1 (mod 12 )
5 ^ 7 = 5 (mod 12 )
5 ^ 8 = 1 (mod 12 )
```



$5^9 = 5 \pmod{12}$   
 $5^{10} = 1 \pmod{12}$   
 $5^{11} = 5 \pmod{12}$   
 $5^{12} = 1 \pmod{12}$   
 $5^{13} = 5 \pmod{12}$   
 $5^{14} = 1 \pmod{12}$   
 $5^{15} = 5 \pmod{12}$   
 $5^{16} = 1 \pmod{12}$   
 $5^{17} = 5 \pmod{12}$   
 $5^{18} = 1 \pmod{12}$   
 $5^{19} = 5 \pmod{12}$   
 $5^{20} = 1 \pmod{12}$   
 $5^{21} = 5 \pmod{12}$   
 $5^{22} = 1 \pmod{12}$   
 $5^{23} = 5 \pmod{12}$   
 $5^{24} = 1 \pmod{12}$   
 $5^{25} = 5 \pmod{12}$   
 $5^{26} = 1 \pmod{12}$   
 $5^{27} = 5 \pmod{12}$   
 $5^{28} = 1 \pmod{12}$   
 $5^{29} = 5 \pmod{12}$   
 $5^{30} = 1 \pmod{12}$   
 $5^{31} = 5 \pmod{12}$   
 $5^{32} = 1 \pmod{12}$   
 $5^{33} = 5 \pmod{12}$   
 $5^{34} = 1 \pmod{12}$   
 $5^{35} = 5 \pmod{12}$   
 $5^{36} = 1 \pmod{12}$   
 $5^{37} = 5 \pmod{12}$   
 $5^{38} = 1 \pmod{12}$   
 $5^{39} = 5 \pmod{12}$   
 $5^{40} = 1 \pmod{12}$   
 $5^{41} = 5 \pmod{12}$   
 $5^{42} = 1 \pmod{12}$   
 $5^{43} = 5 \pmod{12}$   
 $5^{44} = 1 \pmod{12}$   
 $5^{45} = 5 \pmod{12}$

```

5 ^ 46 = 1 (mod 12 )
5 ^ 47 = 5 (mod 12 )
5 ^ 48 = 1 (mod 12 )
5 ^ 49 = 5 (mod 12 )
sage:

```

## 7.2 原根 (primitive root)

### 7.2.1 基本概念

**定义 7.2 (原根 (primitive root))**  $m$  是大于 1 的整数,  $a$  是与  $m$  互素的整数, 如果  $\text{ord}_m(a) = \varphi(m)$ , 则  $a$  称为  $m$  的原根。

设  $a$  对模  $m$  的次数为  $l$ , 符号化描述为  $\text{ord}_m(a) = l$ , 欧拉函数  $\varphi(m)$  表示完全剩余系中与  $m$  互素的元素个数。也就是说欧拉函数只和  $m$  有关, 而次数与  $m$  和  $a$  有关。

[2] 中对原根的定义为: 若  $a$  对模  $m$  的次数<sup>1</sup>是  $\varphi(m)$ , 则  $a$  叫做模  $m$  的一个原根。

#### 例 7.4

5 是否是 6 的原根? 是否是 8 的原根?

**解**

5, 6 互素, 已知  $\varphi(6) = 2$ , 且  $5^1 \equiv 5 \pmod{6}$ ,  $5^2 \equiv 1 \pmod{6}$ , 得  $\text{ord}_6(5) = 2$ , 所以  $\text{ord}_6(5) = \varphi(6)$ , 由定义可知 5 是 6 的原根。

由于 5, 8 互素,  $\varphi(8) = 4$ , 且  $5^1 \equiv 5 \pmod{8}$ ,  $5^2 \equiv 1 \pmod{8}$ , 得  $\text{ord}_8(5) = 2$ , 所以  $\text{ord}_8(5) \neq \varphi(8)$ , 由定义可知 5 不是 8 的原根。

### 7.2.2 原根的等价概念

**定理 7.6 (原根充要条件)**  $\text{ord}_m(a) = \varphi(m) \Leftrightarrow 1, a, a^2, \dots, a^{\varphi(m)}$  是模  $m$  的一个缩系。

也可叙述为:  $a$  是  $m$  原根的充要条件是  $1, a, a^2, \dots, a^{\varphi(m)}$  是模  $m$  的一个缩系。

**定理 7.7 (原根充要条件)**  $a$  是  $m$  的一个原根,  $t$  是非负整数, 则  $a^t$  也是  $m$  的原根的充要条件是  $\gcd(t, \varphi(m)) = 1$ 。

### 7.2.3 原根存在性判断

对于任意模数  $m$ , 不一定存在原根。所以探讨什么情况下原根存在是有意义的。

**定理 7.8 (奇素数原根存在性 [3])** 设  $p$  是奇素数, 则  $p$  的原根存在。

<sup>1</sup>在闵先生的原书中称为指数, 把本书中的指数称为指标, 这是对英文的不同翻译造成的。

**定理 7.9 (奇素数次幂原根存在性 [3])** 设  $p$  是奇素数, 则对于任意正整数  $l$ , 存在  $p^l$  的原根。

**定理 7.10 (奇素数次幂倍数原根存在性 [3])** 设  $p$  是奇素数, 则对于任意正整数  $l$ , 存在  $2p^l$  的原根。

**定理 7.11 (无原根整数 [3])** 设  $a$  是一个奇数, 则对任意整数  $k \geq 3$ , 有  $a^{\frac{1}{2}\varphi(2^k)} \equiv a^{2^{k-2}} \equiv 1 \pmod{2^k}$ , 即  $2^k (k \geq 3)$  没有原根。

有了上面这些定理, 可以推出原根存在的充要条件。

**定理 7.12 (原根存在性判定 [3])** 设  $m$  是大于 1 的整数, 则  $m$  的原根存在的充要条件是  $m$  为 2, 4,  $p^l, 2p^l$  之一, 其中  $l \geq 1, p$  为奇素数。

### 7.2.4 原根的计算

上面我们讨论了原根的存在性判定, 还有两个重要问题就是判定有几个原根和这几个原根是什么。

**定理 7.13 ([3])** 设  $m$  是大于 2 的整数,  $\varphi(m)$  的所有不同的素因子是  $q_1, q_2, \dots, q_s$ , 则与  $m$  互素的正整数  $g$  是  $m$  的一个原根的充要条件是  $g^{\frac{\varphi(m)}{q_i}} \not\equiv 1 \pmod{m}, i = 1, 2, \dots, s$ 。

**例 7.5 [3]**

求 41 的原根。

**解**

$\varphi(m) = \varphi(41) = 40 = 2^3 \times 5$ , 所以  $\varphi(m)$  的素因子是  $q_1 = 5, q_2 = 2$ , 可计算得:

$$\frac{\varphi(m)}{q_1} = \frac{40}{5} = 8, \frac{\varphi(m)}{q_2} = \frac{40}{2} = 20$$

与  $m$  互素的正整数  $g=2,3,4,5,6$ , 分别验算  $g^8, g^{20}$ , 得:

$2^8 \equiv 10 \pmod{41}, 2^{20} \equiv 1 \pmod{41}$ , 不符合原根条件;

$3^8 \equiv 1 \pmod{41}$ , 不符合原根条件;

$4^8 \equiv 18 \pmod{41}, 4^{20} \equiv 1 \pmod{41}$ , 不符合原根条件;

$5^8 \equiv 10 \pmod{41}, 5^{20} \equiv 1 \pmod{41}$ , 不符合原根条件;

$6^8 \equiv 10 \pmod{41}, 6^{20} \equiv 40 \pmod{41}$ , 符合原根条件;

$t$  遍历  $\varphi(m) = 40$  的缩系 1, 3, 7, 9, 11, 13, 17, 19, 21, 23, 27, 29, 31, 33, 37, 39,  $6^t$  遍历 41 的原根, 可得:

$6^1 \equiv 6 \pmod{41}; 6^3 \equiv 11 \pmod{41}; 6^7 \equiv 29 \pmod{41};$

$6^9 \equiv 19 \pmod{41}; 6^{11} \equiv 28 \pmod{41}; 6^{13} \equiv 24 \pmod{41};$

$6^{17} \equiv 26 \pmod{41}; 6^{19} \equiv 34 \pmod{41}; 6^{21} \equiv 35 \pmod{41};$

$6^{23} \equiv 30 \pmod{41}; 6^{27} \equiv 12 \pmod{41}; 6^{29} \equiv 22 \pmod{41};$

$6^{31} \equiv 13 \pmod{41}; 6^{33} \equiv 17 \pmod{41}; 6^{37} \equiv 15 \pmod{41};$

$6^{39} \equiv 7 \pmod{41};$

所以, 41 的原根为: 6, 7, 11, 12, 13, 15, 17, 19, 22, 24, 26, 28, 29, 30, 34, 35

## 7.3 指数/离散对数

“如果  $m$  是一个原根  $g$ , 根据定理 7.6 可知,  $1, g, g^2, \dots, g^{\varphi(m)-1}$  是模  $m$  的一个缩系, 因此, 对任何一个与  $m$  互素的整数  $a$ , 存在唯一的非负整数  $r$ ,  $0 \leq r < \varphi(m)$ , 使得  $g^r \equiv a \pmod{m}$ , 由于原根具有以上性质, 我们可以给出下面的定义”。[3](第 103 页)

**定义 7.3 (指数 (index)[3])** 设  $m$  是大于 1 的整数,  $g$  是  $m$  的一个原根,  $a$  是与  $m$  互素的整数, 则存在唯一的非负整数  $r$ ,  $0 \leq r < \varphi(m)$ , 满足  $a \equiv g^r \pmod{m}$ , 称  $r$  为以  $g$  为底  $a$  对模  $m$  的指数, 记作  $\text{ind}_g a, a \equiv g^{\text{ind}_g a} \pmod{m}$ , 有时也把指数称为离散对数, 记为  $\log_g a, a \equiv g^{\log_g a} \pmod{m}$ 。

**注** 对于实数来说  $a^c = b \rightarrow \log_a b = c$ ,  $c$  叫做以  $a$  为底  $b$  的对数 (logarithm)。所以我们通常也把  $\text{ind}_g a$  叫做离散对数, 记作  $\log_g a$ 。

离散对数的计算问题是一个计算上困难的问题, 目前没有找到一个有效的算法。

### 例 7.6

$m=7$ , 有原根  $g=3$ , 计算指数表。

**解**

$3^x \equiv a \pmod{7}$ , 可知  $x$  和  $a$  的关系为:

$$x = 1, a = 3^1 = 3 \equiv 3 \pmod{7}$$

$$x = 2, a = 3^2 = 9 \equiv 2 \pmod{7}$$

$$x = 3, a = 3^3 = 27 \equiv 6 \pmod{7}$$

$$x = 4, a = 3^4 = 81 \equiv 4 \pmod{7}$$

$$x = 5, a = 3^5 = 243 \equiv 5 \pmod{7}$$

$$x = 6, a = 3^6 = 729 \equiv 1 \pmod{7}$$

整理以上计算结果, 可得指数表。

**定理 7.14 (原根的幂)**  $g$  是  $m$  的一个原根,  $g^x \equiv g^y \pmod{m} \Leftrightarrow x \equiv y \pmod{\varphi(m)}$

**定理 7.15 (原根的次幂)**  $g$  是  $m$  的一个原根, 整数  $a, b$  均与  $m$  互素,  $a \equiv b \pmod{m} \Leftrightarrow \text{ind}_g a \equiv \text{ind}_g b \pmod{\varphi(m)}$

可见离散指数与实数中的指数的性质很相似, 我们可以利用原根做出指数表。

### 例 7.7 [3]

做出模 41 的指数表

**解**

已知 6 是 41 的一个原根, 所以  $g=6$ , 又知  $\varphi(41) = 40$ , 直接计算  $g^r \pmod{m}$ ,  $g = 6, m = 41, r = 0, 1, \dots, 39$ : (下表中的值为  $6^{a+b} \pmod{41}$ ,  $a \in \{0, 1, 2, 3, 4, 5\}, b \in \{0, 6, 12, 18, 24, 30, 36\}$ )

	a=0	a=1	a=2	a=3	a=4	a=5
b=0	$6^{(a+b)} \equiv 1(mod\ 41)$	6	36	11	25	27
b=6	39	29	10	19	32	28
b=12	4	24	21	3	18	26
b=18	33	34	40	35	5	30
b=24	16	14	2	12	31	22
b=30	9	13	37	17	20	38
b=36	23	15	8	7	-	-

根据计算结果，我们构造指数表，第一行表示  $g^r(mod\ m)$  的个位数，第一列表示  $g^r(mod\ m)$  的十位数，交叉位置即为 r:

模 41 的指数表

	0	1	2	3	4	5	6	7	8	9
0	-	0	26	15	12	22	1	39	38	30
1	8	3	27	31	25	37	24	33	16	9
2	34	14	29	36	13	4	17	5	11	7
3	23	28	10	18	19	21	2	32	35	6
4	20	-	-	-	-	-	-	-	-	-

## 7.4 n 次剩余

**定义 7.4 (n 次剩余)** m 是大于 1 的整数，a 是与 m 互素的整数，若  $n(n \geq 2)$  次同余方程  $x^n \equiv a(mod\ m)$  有解，则把 a 称为模 m 的 n 次剩余，否则（无解），a 叫作模 m 的 n 次非剩余。

**定理 7.16 (高次同余方程有解充要条件)** g 是 m 的一个原根，a 是与 m 互素的整数，则同余方程  $x^n \equiv a(mod\ m)$  有解  $\Leftrightarrow gcd(n, \varphi(m)) \mid ind_g a$ , 且解的个数为  $gcd(n, \varphi(m))$ .

### 例 7.8

求解同余方程  $x^{12} \equiv 37(mod\ 41)$ .

**解**

$\varphi(41) = 40, gcd(12, 40) = 4$ , 查模 41 的指数表，知  $ind_g 37 = 32$ , 所以  $4 \mid 32$ , 可知同余方程有解。

原同余方程与  $12ind_g x \equiv ind_g 37 = 32(mod\ 40)$  等价，有  $3ind_g x \equiv 8(mod\ 10)$ , 3 模 10 的逆元是 7，两边同乘 7，得到  $ind_g x \equiv 56 \equiv 6(mod\ 10)$ , 可解得  $ind_g x \equiv 6, 16, 26, 36(mod\ 40)$ , 通过查模 41 的指数表可得到原同余方程的解为  $x \equiv 39, 18, 2, 23(mod\ 41)$ .

## 7.5 习题

**Exercise 45**

34 对模 37 的次数是多少？

**Exercise 46**

判断 47, 55, 59 的原根是否存在。若存在，求出其所有原根。

**Exercise 47**

求 47 所有原根。



## 第8章 群 (group)

### 8.1 基本概念

**定义 8.1 (二元运算 (binary operation))** 集合  $G$  上的二元运算是一个如下的函数:  $*$  :  $G \times G \rightarrow G$ .

在不引起歧义的情况下, 二元运算  $*(a, b)$  通常写成  $a * b$ , 这里需要注意的是, 二维及以上向量的点积运算不是我们这里定义的二元运算, 如  $a = (x_1, y_1), b = (x_2, y_2), a \cdot b = x_1x_2 + y_1y_2$ , 因为参与运算的是两个向量, 结果是一个数, 显然不是同一个集合, 不符合定义。

**定义 8.2 (代数系统 (algebra system))** 对于非空集合  $S$  以及  $S$  上的运算  $\star$ , 若运算满足封闭性 (closure), 则称  $S$  和运算构成一个代数系统, 记为  $(S, \star)$ .

**定义 8.3 (半群 (semigroup))**  $G$  是一个非空集合,  $*$  是定义在集合  $G$  上的一个二元运算,  $(G, *)$  被称为半群, 如果  $(G, *)$  满足以下条件:

- (1) 封闭 (closure): 对于任意  $a, b \in G \Rightarrow a * b \in G$ .
- (2) 结合律 (associative): 对于任意  $a, b, c \in G \Rightarrow a * (b * c) = (a * b) * c$ .

**例 8.1** 整数集  $\mathbb{Z}$ , 有理数集  $(\mathbb{Q})$ , 实数集  $\mathbb{R}$ , 复数集  $\mathbb{C}$  在普通加法下满足封闭性和结合律, 所以构成半群, 同样在普通乘法下满足封闭性和结合律, 所以也构成半群。

**定义 8.4 (幺半群或单位半群 (monoid))** 具有单位元 (identity element) 的半群称为独异点, 也叫幺半群。

**定义 8.5 (群 (group))** 半群  $(G, *)$  被称为群, 如果满足以下条件:

- (1) 单位元 (identity element):  $\exists e \in G, \forall a \in G \Rightarrow e * a = a$ , 此时我们称  $e$  为  $G$  的左幺元 (member of the upper-left)。
- (2) 逆元 (inverse element):  $\forall a \in G, \exists a' \in G, a' * a = e$ , 此时我们称元素  $a'$  为  $a$  的左逆元 (left inverse element)。

群的另外一个定义是:  $G$  中每个元素都有逆元的独异点叫群。



**注意** 以上各个定义是渐进定义, 也就是不断加入新的限制, 如封闭 (closure)、结合律 (associative), 还有以后要用到的消去律 (cancellation law) 和交换律 (commutative law)。

**例 8.2**  $(\mathbb{Z}, +)$ , 单位元是 0, 任何一个整数  $a$  都存在逆元  $-a$ , 所以  $(\mathbb{Z}, +)$  为群。同样, 有理数集  $\mathbb{Q}$ , 实数集  $\mathbb{R}$ , 复数集  $\mathbb{C}$  在普通加法下构成群。  $\mathbb{Q}^* = \mathbb{Q} \setminus \{0\}$ ,  $(\mathbb{Q}^*, \times)$ , 单位元是 1, 任何一个有理数  $a$ , 存在逆元  $\frac{1}{a}$ , 所以  $(\mathbb{Q}^*, \times)$  为群, 同样  $\mathbb{R}^* = \mathbb{R} \setminus \{0\}$   $\mathbb{C}^* = \mathbb{C} \setminus \{0\}$  在普通乘法下也构成群。  $(\mathbb{Z}^*, \times)$  存在单位元, 但是不存在逆元, 所以  $(\mathbb{Z}^*, \times)$  仅能构成半群。

**定理 8.1 (群的幺元和逆元性质)**  $G$  是一个群,  $e$  是  $G$  的左幺元, 则有:

- (1) 任意  $a \in G$ ,  $b$  是  $a$  的左逆元, 则  $b$  也是  $a$  的右逆元, 称  $b$  是  $a$  的逆元。

(2)  $e$  也是  $G$  的右幺元, 称  $e$  是  $G$  的幺元。

(3) 任意  $a \in G$ , 其逆元是唯一的。

**证明** (1) 设  $c$  为  $b$  的左逆元,  $a * b = e * (a * b) = (c * b) * (a * b) = c * (b * a) * b = c * e * b = c * b = e$ , 可见  $b$  也是  $a$  的右逆元。

(2) 设  $b$  为  $a$  的逆元,  $a * e = a * (b * a) = (a * b) * a = e * a = a$ , 可见  $e$  也是右逆元。

(3) 设  $b, d$  均为  $a$  的逆元,  $b = b * e = b * (a * d) = (b * a) * d = e * d = d$ , 可见逆元唯一。

**定义 8.6 (群的阶 (group order))** 群  $(G, *)$  的元素个数称为此群的阶, 记为  $|G|$ , 如果  $|G|$  有限, 则称  $(G, *)$  为有限群 (finite group), 否则称为无限群 (infin group)。

### 例 8.3

$\mathbb{Z}_n = \{0, 1, 2, \dots, n-1\}$ , 定义  $(\mathbb{Z}_n, +)$ , 二元运算  $+$  为模  $n$  的加法,  $(\mathbb{Z}_n, +)$  为群, 为有限群。

**定义 8.7 (元素的阶 (order))** 群  $(G, *)$  中的元素  $a$ , 使  $a^n = 1$  的最小正整数  $n$  称为元素  $a$  的阶, 记为  $ord(a)$ 。如果不存在这样的正整数, 我们称  $a$  为无限阶元素。

**注** 在谈到任意群的幺元时, 通常用  $1$  来表示, 但是其并不是  $1$ , 只是一个记号, 例如整数加法群里面幺元是  $0$ 。用  $a^{-1}$  表示  $a$  的逆元。

### 例 8.4

(1) 在任何群中, 只有单位元的阶为  $1$ , 即  $ord(1) = 1$ 。

(2) 普通加法下,  $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$  中, 每个非零数都是无限阶。整数中幺元为  $0$ , 任何一个非零数连加都不会是  $0$ , 所以是无限阶。

(3) 普通乘法下,  $\mathbb{Q}^*, \mathbb{R}^*, \mathbb{C}^*$  中,  $ord(1) = 1, ord(-1) = 2$ , 幺元为  $1$ , 非  $1$  和  $-1$ , 没有任何数连乘会是  $1$ , 所以其他都是无限阶。

**定理 8.2 (关于方程解)**  $G$  是一个群,  $a, b \in G$ , 则方程  $ax = b$  和  $ya = b$  有唯一的解。

**定理 8.3 (群中元素幂的性质)** 对于正整数  $m$  和  $n$ , 群中元素  $a$  的幂满足:

$$(1) (a^{-1})^n = (a^n)^{-1};$$

$$(2) a^{n+m} = a^n a^m;$$

$$(3) (a^n)^m = a^{nm};$$

**定理 8.4 (元素阶的性质)** 有限群  $G$  中元素  $a$  的阶必为有限数。

### 例 8.5

Klein<sup>1</sup>四元群为集合  $G = \{a, b, c, e\}$ , 其上二元运算  $\cdot$  定义如下表 (通常有限群的运算关系用以下表格方式给出, 此表称为  $G$  的群表):

$\cdot$	$e$	$a$	$b$	$c$
$e$	$e$	$a$	$b$	$c$
$a$	$a$	$e$	$b$	$c$
$b$	$b$	$c$	$e$	$a$
$c$	$c$	$b$	$a$	$e$

<sup>1</sup>Klein 是个德国人, 中文又是音译为克莱因, Klein 群是一个最小非循环群, 有时常用  $V$  来表示, 因为德文的四元群单词为 Vierergruppe



解

观察群表, 可知,  $\forall x \in G, x \cdot e = e \cdot x = x$ , 所以  $e$  是单位元或么元,  $\forall x \in G, x \cdot x = e$ , 所以  $x$  的逆元就是  $x$  自身。同样可以验证  $\forall x, y, z \in G, (x \cdot y) \cdot z = x \cdot (y \cdot z)$ , 即满足结合律。

$(G, \cdot)$  对于二元运算是封闭的, 满足结合律, 所以是一个半群, 由于存在单位元和逆元, 所以其是群。

## 8.2 子群

**定义 8.8 (子群 (subgroup), 平凡子群 (trivial subgroup), 真子群 (proper subgroup))** 设  $(G, *)$  是一个群,  $H \subset G$ , 如果  $H$  对于运算  $*$  也构成群, 那么称  $H$  是  $G$  的子群, 记为  $H \leq G$ . 由于  $\{1\}$  和  $G$  都是  $G$  的子群, 我们称  $\{1\}$  和  $G$  为  $G$  的平凡子群, 否则称为非平凡子群。如果子群  $H \neq G$ , 我们认为  $H$  是真子群, 记为  $H < G$ 。

### 例 8.6

在普通加法下,  $\mathbb{Z} \leq \mathbb{Q} \leq \mathbb{R} \leq \mathbb{C}$ 。

### 例 8.7

在普通加法下  $\mathbb{R}$  是群, 普通乘法下  $\mathbb{Q}^*$  是群, 而且  $\mathbb{Q}^* \subset \mathbb{R}$ , 但  $(\mathbb{Q}^*, \times)$  不是  $(\mathbb{R}, +)$  子群, 因为这两个群的二元运算不同。

**定理 8.5 (子群的等价条件)**  $G$  是一个群,  $H$  是它的非空子集, 则:

$$H \leq G \Leftrightarrow$$

$$1 \in H; a \in H \text{ then } a^{-1} \in H; a, b \in H \text{ then } ab \in H \Leftrightarrow$$

$$a, b \in H \text{ then } ab \in H, a^{-1} \in H \Leftrightarrow$$

$$\forall a, b \in H, ab^{-1} \in H$$

由于这些条件都是与  $H$  是  $G$  子群的等价条件, 所以也可以用于子群的判断。

### 例 8.8

$n \in \mathbb{Z}, n\mathbb{Z} = \{n \times k \mid k \in \mathbb{Z}\}$ , 则  $(n\mathbb{Z}, +)$  是  $(\mathbb{Z}, +)$  的子群。

**证明**  $n\mathbb{Z} \subset \mathbb{Z}, \forall a, b \in n\mathbb{Z}$ , 存在  $i, j \in \mathbb{Z}$ , 使得  $a = n \times i, b = n \times j$ , 我们有  $a + b = n \times i + n \times j = n \times (i + j) \in n\mathbb{Z}$ , 单位元为 0,  $a^{-1} = -n \times i \in n\mathbb{Z}$ , 根据上述定理  $(a, b \in H \text{ then } ab \in H, a^{-1} \in H)$ , 我们可知  $(n\mathbb{Z}, +)$  是  $(\mathbb{Z}, +)$  的子群。

**定理 8.6 (有限群子群判定)**  $G$  是一个有限群, 他的非空子集  $H$  是子群  $\Leftrightarrow H$  在  $G$  的二元运算下是封闭的。

上述定理只有在  $G$  是有限群时才成立, 下面我们给出一个示例。

### 例 8.9

在普通加法下  $\mathbb{Z}$  是一个无限群,  $\mathbb{N} \subset \mathbb{Z}$ , 且加法下封闭, 但其不是  $\mathbb{Z}$  的子群。我们如果从群的定义出发, 会发现自然数集合在加法下是一个半群, 但是不是一个群。

**定义 8.9 (正规子群 (normal subgroup))**  $K$  是  $G$  的子群, 如果对于任意  $k \in K, g \in G$ , 有  $gkg^{-1} \in K$ , 则称  $K$  是  $G$  的正规子群, 记为  $K \triangleleft G$ 。

**例 8.10**

$$n > 1, (n\mathbb{Z}, +) \triangleleft (\mathbb{Z}, +).$$

**解**

任意  $g \in \mathbb{Z}, k \in n\mathbb{Z}$ , 加法满足交换律,  $g$  逆元为  $-g$ , 所以  $g + k + (-g) = g + (-g) + k = 1 + k = 0 + k = k \in n\mathbb{Z}$ , 所以  $n\mathbb{Z}$  是  $\mathbb{Z}$  的正规子群。

**定理 8.7 (正规子群的等价条件)**  $H \leq G$ , 则:

$$H \triangleleft G \Leftrightarrow$$

$$\forall g \in G, gHg^{-1} = H \Leftrightarrow$$

$$\forall g \in G, gH = Hg; \Leftrightarrow$$

$$\forall g_1, g_2 \in G, g_1Hg_2H = g_1g_2H.$$

**例 8.11**  $(n\mathbb{Z}, +) \triangleleft (\mathbb{Z}, +)$ , 举例来验证以上等价条件。

**8.3 交换群/阿贝尔群**

**定义 8.10 (阿贝尔群 (Abelian groups))** 如果群  $(G, *)$  中的二元运算  $*$  满足交换律 (commutative law), 那么群  $(G, *)$  称为阿贝尔群 (Abelian groups) 或交换群 (commutative groups)。

**定理 8.8 (交换群子群的正规性)** 任意交换群的子群都是正规子群。

**8.4 循环群**

**定义 8.11 (循环子群 (cyclic sub-group))**  $G$  是一个群,  $a \in G$ , 集合  $\{a^n \mid n \in \mathbb{Z}\}$  称为由元素  $a$  生成的  $G$  的循环子群, 记为  $\langle a \rangle$ 。



**注意** 在循环子群的定义中注意理解  $a^n$  的含义, 其是指对于群  $(G, *)$   $\underbrace{a * a * \dots * a}_n$ 。

**定理 8.9 ( $\langle a \rangle$  是子群)**  $\langle a \rangle$  是一个群, 且为  $G$  的子群。

**定义 8.12 (循环群 (cyclic group))**  $G$  是一个群, 如果  $\exists a \in G, G = \langle a \rangle$ , 则称  $G$  为循环群, 称  $a$  为  $G$  的生成元。

由前面的定义和定理可知, 任何循环群的子群必定是循环群。一个循环群可以有不止一个生成元, 例如, 集合  $G = \langle a \rangle = \{a^n \mid n \in \mathbb{Z}\}$ , 因为  $\{a^n \mid n \in \mathbb{Z}\} = \{(a^{-1})^n \mid n \in \mathbb{Z}\} = \langle a^{-1} \rangle$ , 所以  $G = \langle a^{-1} \rangle$ 。

**例 8.12**

$(\mathbb{Z}, +)$  是交换群, 任取  $a \in \mathbb{Z}, \langle a \rangle = \{na \mid n \in \mathbb{Z}\}, (\langle a \rangle, +)$  是  $(\mathbb{Z}, +)$  的循环子群。

当  $a = 0$  时, 这个子群只有一个元素  $0$  组成。

当  $a = 1$  时,  $\langle 1 \rangle = \mathbb{Z}$ , 所以  $(\mathbb{Z}, +)$  是循环群,  $1$  是  $\mathbb{Z}$  的生成元。

当  $a = 2$  or  $-2$  时,  $\langle 2 \rangle = \langle -2 \rangle =$  偶数集合, 所以  $2$  和  $-2$  是偶数集合的生成元。

奇数集合不是  $\mathbb{Z}$  的循环子群, 奇数集合根本不是群, 因为不包含单位元  $\mathbb{K}$ 。

**例 8.13**

集合  $\mathbb{Z}_6 = 0, 1, 2, \dots, 5$  (此处二元运算是模 6 加法<sup>2</sup>), 1 是  $\mathbb{Z}_6$  的生成元, 另一个明显的生成元是 5<sup>3</sup>。他的子集  $\{0, 3\}$  是一个循环群, 该子群的生成元只有一个是 3, 另一个子集  $\{0, 2, 4\}$  也是循环群, 生成元是 2, 4。

**例 8.14**

$\mathbb{Z}_5^* = \{1, 2, 3, 4\}$ , 2 和 3 都是他的生成元<sup>4</sup>。该集合的子集  $\{1, 4\}$  是  $(\mathbb{Z}_5^*, \cdot)$  的一个循环子群, 生成元是 4。

**定理 8.10 (循环群的生成元)**  $G = \langle a \rangle$  是一个循环群, 且  $|G| = n$ , 则当且仅当  $\gcd(k, n) = 1$  时,  $a^k$  是  $G$  的生成元。

**定理 8.11 (循环群的生成元)**  $n$  阶循环群共有  $\varphi(n)$  个生成元。

**定理 8.12 (循环群的子群生成元)**  $G = \langle a \rangle$  是一个循环群,  $S \leq G$ , 则  $S$  必定是一个循环群, 且如果  $k$  是使得  $a^k \in S$  的最小正整数, 则  $a^k$  是  $S$  的生成元。

**定理 8.13 (有限循环群的阶)**  $G$  是有限群, 且  $a \in G$ , 则  $\text{ord}(a) = |\langle a \rangle|$ 。

**定理 8.14 (有限循环群的子群)**  $G = \langle a \rangle$  是一个有限循环群,  $|G| = n$ , 则对于任意整除  $n$  的正整数  $d$ , 一定存在一个唯一的阶为  $d$  的循环子群  $\langle a^{\frac{n}{d}} \rangle$ 。

**8.4.1 循环子群的构造**

**定理 8.15 (子群的交集)** 子群的交集还是子群。

**定义 8.13 (集合生成的子群)**  $(G, \cdot)$  是群,  $S$  是  $G$  的子集, 设  $(H_i \mid i \in I, \cdot)$  是  $(G, \cdot)$  的所有包含集合  $S$  的子群, 即  $S \subset H_i (i \in I)$ , 则  $(H_i \mid i \in I, \cdot)$  称为由集合  $S$  生成的子群, 记为  $\langle S \rangle$ ,  $S$  中的元素叫子群  $\langle S \rangle$  的生成元。

**8.5 置换群**

**定义 8.14 (置换 (permutation))** 给定非空集合  $X$ , 我们将任意一个双射  $\alpha: X \rightarrow X$  称作集合  $X$  的一个置换。

把函数的复合“ $\circ$ ”看作一种置换间的二元运算 (注意: 此处群的二元运算是置换的复合), 那么非空集合  $X$  的所有置换组成的集合  $S_X$  就是一个群, 我们把这个群记为  $(S_X, \circ)$ , 满足:

(1) 封闭性 (closure): 任意两个置换的复合也是置换, 所以“ $\circ$ ”是  $S_X$  上的封闭二元运算。

<sup>2</sup>模 6 加

<sup>3</sup>此处  $5^n$  表示  $n$  个 5 相加模 6, 所以有  $5 \equiv 5 \pmod{6}$ ;  $5^2 = 10 \equiv 4 \pmod{6}$ ;  $5^3 = 15 \equiv 3 \pmod{6}$ ;  $5^4 = 20 \equiv 2 \pmod{6}$ ;  $5^5 = 10 \equiv 1 \pmod{6}$ ;  $5^6 = 30 \equiv 0 \pmod{6}$ ; 5 生成所有元素。

<sup>4</sup>这种说法其实很迷惑人, 因为并没有说明其二元运算是模 5 乘法, 根据本题后面的描述, 我们可以假设其二元运算是模 5 乘法, 验算一下,  $2^1 = 2 \equiv 2 \pmod{5}$ ;  $2^2 = 4 \equiv 4 \pmod{5}$ ;  $2^3 = 8 \equiv 3 \pmod{5}$ ;  $2^4 = 16 \equiv 1 \pmod{5}$ , 同样验证 3 是生成元,  $3^1 = 3 \equiv 3 \pmod{5}$ ;  $3^2 = 9 \equiv 4 \pmod{5}$ ;  $3^3 = 27 \equiv 2 \pmod{5}$ ;  $3^4 = 81 \equiv 1 \pmod{5}$ , 但是我们可以看出如果二元运算是模 5 加法, 也成立, 读者可以自行验算一下

(2) 结合律 (associative): 一般函数的复合满足结合律, 所以  $\circ$  满足结合律。

(3) 单位元 (identity element): 定义恒等置换  $I_X: X \rightarrow X \Leftrightarrow x \in X, I_X(x) = x$ , 则对于任意置换  $\alpha, \alpha \circ I_X = I_X \circ \alpha = \alpha$ , 所以  $I_X$  是单位元。

(4) 逆元 (inverse element): 对于任意置换  $\alpha: X \rightarrow X$ , 因为是双射, 所以存在逆函数  $\alpha^{-1}: X \rightarrow X, \alpha \circ \alpha^{-1} = \alpha^{-1} \circ \alpha = I_X$ , 所以置换  $\alpha^{-1}$  是  $\alpha$  的逆元。

由上可见  $(S_X, \circ)$  满足所有的群条件, 所以其是一个群。

**定义 8.15 (全变换群 (transformation group))**  $(S_X, \circ)$  称为集合  $X$  上的全变换群或对称群, 当  $X = 1, 2, \dots, n$  时, 称  $S_X$  为  $n$  次变换群 ( $n$  次对称群), 记作  $S_n$ 。

利用排列组合的知识我们知道  $S_n$  的元素数量是  $n!$ 。

**定义 8.16 (r-轮换)** 设  $\alpha \in S_n, A = \{i_1, i_2, \dots, i_r\} \subset \{1, 2, \dots, n\}, B = \{1, 2, \dots, n\} - A$ , 如果置换  $\alpha$  满足:

(1) 对  $A$  中的元素有  $\alpha(i_1) = i_2, \alpha(i_2) = i_3, \dots, \alpha(i_{r-1}) = i_r, \alpha(i_r) = i_1$ <sup>5</sup>; (2)  $i \in B, \alpha(i) = i$ ; 我们称置换  $\alpha$  是一个  $r$ -轮换, 记为  $\alpha(i_1, i_2, \dots, i_r)$ , 我们也把“2-轮换”称为对换。

### 例 8.15

一个“3-轮换”  $\alpha = (2, 1, 3) \in S_5$ , 其置换的含义是  $\alpha(2) = 1, \alpha(1) = 3, \alpha(3) = 2, \alpha(4) = 4, \alpha(5) = 5$ . 也可以表示为:

$$\alpha = \begin{pmatrix} 2 & 1 & 3 & 4 & 5 \\ 1 & 3 & 2 & 4 & 5 \end{pmatrix}$$

任意置换可以分解为多个轮换的复合, 为了称呼简单一些, 我们以后将置换的复合称为“乘积”, 以此来简化轮换的复合表示。

### 例 8.16

如下置换  $\alpha \in S_5$  可以用两种不同的轮换乘积来表示:  $\alpha = (1, 2)(1, 3, 4, 2, 5)(2, 5, 1, 3) = (1, 4)(3, 5)(2)$

复合运算是从右向左, 我们可以验证上面是相等的:

$$\begin{pmatrix} 2 & 5 & 1 & 3 & 4 \\ 5 & 1 & 3 & 2 & 4 \\ - & - & - & - & - \\ 1 & 3 & 4 & 5 & 2 \\ - & - & - & - & - \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 5 & 1 & 3 & 4 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix} \text{rearrange} \begin{pmatrix} 2 & 5 & 3 & 1 & 4 \\ 2 & 3 & 5 & 4 & 1 \end{pmatrix} = (2)(5, 3)(1, 4)$$

**定义 8.17 (轮换不相交)**  $\alpha, \beta \in S_n$  是两个轮换, 且两个轮换记号中没有共同的数字, 称轮换  $\alpha, \beta$  不相交, 如果一组轮换中任意两个轮换都不相交, 称改组轮换不相交。

1-轮换都等于恒等置换, 对于恒等置换写作  $1_n$ , 对于任意轮换  $\alpha \in S_n$ , 如果已知他的轮换分解, 求出逆置换的方法是将轮换分解的每一个轮换中的数字倒排, 例如  $\alpha = (1, 4)(5, 3) \rightarrow \alpha^{-1} = (4, 1)(5, 3), \alpha\alpha^{-1} = 1_{S_5}$ , 可以计算验证一下。

<sup>5</sup>这是按照变换进行了排序, 前一个变换结果是下一个变换的变量, 这样  $A$  的变换其实是形成了一个循环, 画图更容易看一些。

$S_2$  只有两个元素, 明显是个交换群, 但是  $S_n$  ( $n \geq 3$ ) 是非交换群。

### 例 8.17

对于  $S_n$  ( $n \geq 3$ ) 有:

$$(1, 2)(1, 3) = (1, 3, 2)$$

$$(1, 3)(1, 2) = (1, 2, 3)$$

可见  $(1, 2)(1, 3) \neq (1, 3)(1, 2)$ .

尽管  $S_n$  ( $n \geq 3$ ) 是非交换群, 但是不相交的轮换是可交换的。

**定理 8.16 (不相交轮换可交换)** 不相交的轮换是可交换的。

对于一个置换的不相交轮换分解来说, 随意调整其中各个轮换的次序不会改变该轮换。

**定理 8.17 (置换唯一分解)**  $S_n$  中的任意置换一定能够分解为不相交轮换的乘积, 且这种分解式唯一的<sup>6</sup>。

**定义 8.18 (轮换和置换的阶)** 对于一个  $r$ -轮换  $\alpha$ ,  $\text{ord}(\alpha) = r$ , 对于任意置换, 先将置换进行不相交轮换分解, 该置换的阶就等于所有轮换因子长度的最小公倍数。

### 例 8.18

3-轮换  $(1, 2, 3) \in S_3$ , 我们用定义求解其阶。

$$(1, 2, 3) = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix},$$

$$(1, 2, 3)(1, 2, 3) = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix},$$

$$(1, 2, 3)(1, 2, 3)(1, 2, 3) = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix},$$

可见  $(1, 2, 3)$  的阶是 3.

### 例 8.19

求  $\alpha = (1, 2, 3)(4, 5) \in S_5$  的阶。

解

$$\text{ord}(\alpha) = \text{lcm}(3, 2) = 6.$$

**定义 8.19 (置换群 (Permutation group))** 将任意全变换群的任意子群称为一个置换群。

## 8.6 陪集和商群

群和她的子群之间有一定的关系, 陪商和商群就是研究群和子群之间关系的。

<sup>6</sup>这种唯一性是要有约束条件的, 分解的形式必须满足以下条件:

- (1) 对于一个置换的不相交轮换分解, 随意调整其中各轮换的次序, 把这些记法看做同一个轮换分解。
- (2) 把一个轮换的不同记法看做一个轮换。
- (3) 一个置换的不相交轮换分解中去掉任何 1-轮换。

**定义 8.20 (左陪集 (Coset))** 设  $(G, \bullet)$  为群,  $H \leq G, a \in G, a \bullet H = \{a \bullet h \mid h \in H\}$ , 我们称  $a \bullet H$  这样的子集为群  $G$  关于子群  $H$  的左陪集,  $a$  称为代表元。

**例 8.20** 令  $\langle 3 \rangle = \{n \times 3 \mid n \in \mathbb{Z}\}, (\langle 3 \rangle, +) \leq (\mathbb{Z}, +)$ , 求  $(\langle 3 \rangle, +)$  的所有左陪集。

**解** 令  $a=0$ , 则相应的左陪集为  $\{0 + n \times 3 \mid n \in \mathbb{Z}\}$ , 也就是说这个集合是所有 3 的倍数组成的集合, 用同余的概念, 我们也可以表示为  $\{k \mid k \equiv 0(\text{mod } 3)\}$ 。

令  $a=1$ , 则相应的左陪集为  $\{1 + n \times 3 \mid n \in \mathbb{Z}\}$ , 用同余的概念, 我们也可以表示为  $\{k \mid k \equiv 1(\text{mod } 3)\}$ 。

令  $a=3$ , 则相应的左陪集为  $\{2 + n \times 3 \mid n \in \mathbb{Z}\}$ , 用同余的概念, 我们也可以表示为  $\{k \mid k \equiv 2(\text{mod } 3)\}$ 。

我们再设  $a$  为其他整数, 不难发现,  $a$  为其他整数时, 都是以上三个集合之一。所以  $(\langle 3 \rangle, +)$  的左陪集为  $\{k \mid k \equiv 0(\text{mod } 3)\}$ 、 $\{k \mid k \equiv 1(\text{mod } 3)\}$ 、 $\{k \mid k \equiv 2(\text{mod } 3)\}$ 。

**定义 8.21 (左陪集关系 (Coset relation))** 设  $(H, \bullet) \leq (G, \bullet)$ , 我们确定  $G$  上的一个关系  $\equiv^7, a \equiv b \Leftrightarrow a^{-1} \bullet b \in H$ , 这个关系叫  $G$  上关于  $H$  的左陪集关系。也可以写为  $\{ \langle a, b \rangle \mid a, b \in G \wedge (a^{-1} \bullet b) \in H \}$ 。

**例 8.21** 对于群  $(\mathbb{Z}, +)$ , 么元为 0,  $\forall a, a^{-1} = -a$ , 此时我们看子群  $(\langle 3 \rangle, +)$ , 可知其关于  $(\langle 3 \rangle, +)$  的左陪集关系满足  $\{ \langle a, b \rangle \mid a, b \in \mathbb{Z} \wedge (b - a) \in \langle 3 \rangle \}$ , 也就是说  $b$  和  $a$  的差是 3 的倍数, 那这个关系根据前面的同余关系讨论, 显然是模 3 的同余关系。

**定理 8.18 (左陪集关系的等价性)** 设  $(H, \bullet) \leq (G, \bullet)$ , 则  $G$  上关于  $H$  的左陪集关系是等价关系。

因为左陪集关系是一个等价关系, 所以可以利用左陪集关系对  $G$  进行划分。

**定义 8.22 (a 为代表元的等价类)** 群  $(G, \bullet)$  的子群  $(H, \bullet)$  所确定的左陪集关系  $\equiv$  对  $G$  划分等价类, 将  $[a] = \{x \mid x \in G, a \equiv x\}$  等价类叫作以  $a$  为代表元的等价类。

**例 8.22** 在前面一个例子的讨论中, 我们知道, 同余关系是群  $(\mathbb{Z}, +)$  上关于  $(\langle 3 \rangle, +)$  左陪集关系, 以  $a$  为代表元的等价类  $[a] = C_a, C_a$  就是前面定义的同余等价类。

**定理 8.19 (等价类与左陪集关系)** 设  $(H, \bullet) \leq (G, \bullet)$ , 则  $[a] = a \bullet H$ 。

**定理 8.20 (拉格朗日定理)** 设  $(H, \bullet) \leq (G, \bullet), (G, \bullet)$  是有限群, 则  $|H|$  是  $|G|$  的因子。

**定义 8.23 (商集 (quotient set))** 设群  $(G, \bullet)$  有一个子群  $(H, \bullet)$ , 则  $H$  在  $G$  中的两两不相交左陪集组成的集合称为  $H$  在  $G$  中的商集, 记为  $G/H, G/H$  中两两不想交的左陪集个数叫做  $H$  在  $G$  中的指标, 记为  $[G : H]$ 。

**例 8.23** 同余关系是群  $(\mathbb{Z}, +)$  上关于  $(\langle 3 \rangle, +)$  左陪集关系, 那么关于同余关系子群  $(\langle 3 \rangle, +)$  的左陪集根据上面的定理, 我们知道有  $[0], [1], [2]$ , 所以  $\langle 3 \rangle$  在  $\mathbb{Z}$  中的商集为  $\{[0], [1], [2]\}$ 。

**例 8.24** 更一般的,  $(n\mathbb{Z}, +)$  在整数加法群  $(\mathbb{Z}, +)$  中的商集为  $\mathbb{Z}/n\mathbb{Z} = \{[0], [1], \dots, [n-1]\}$ 。

<sup>7</sup>此处只是使用了与同余相等同样的符号, 两者没有任何关系

**定理 8.21 (正规子群的充要条件)** 群  $(G, \bullet)$  的子群  $(H, \bullet)$  是正规子群的充要条件是,  $\forall a \in G, aN = Na$ 。

根据以上定理可知, 正规子群形成的陪集没有左右之分。

**定理 8.22 (商群 (quotient group) 的定义)** 设群  $(G, \bullet)$  有一个正规子群  $(N, \bullet)$ ,  $T = G/N$  是  $N$  在  $G$  中的商集, 在商集  $T$  上定义二元运算  $\odot$ :

对于任意  $aN, bN \in T$  ( $a, b \in G$ ),  $aN \odot bN = (a \bullet b)N$

则  $(T, \odot)$  构成群, 并称为群  $(G, \bullet)$  对正规子群  $(N, \bullet)$  的商群, 记为  $(T, \odot) = (G, \bullet)/(N, \bullet) = (G/N, \bullet)^8$ 。

**例 8.25** 同余关系是群  $(\mathbb{Z}, +)$  上关于  $\langle 3 \rangle, +$  左陪集关系,  $\langle 3 \rangle$  在  $\mathbb{Z}$  中的商集  $T = \{[0], [1], [2]\}$ , 因为整数上的加法运算满足分配律 (distributive law), 根据商群定义, 群  $(\mathbb{Z}, +)$  对正规子群  $\langle 3 \rangle, +$  的商群为  $(T, +)$

 **注意** 商群  $(T, \odot)$  是群  $(G, \bullet)$  的子群吗?

显然不是, 集合不是子集关系, 二元运算也可能不同。在代数中, 商的概念是"元素划分的集合"而非元素的一部分, 这个需要体会一下。

## 8.7 同态和同构

"在代数学中, 我们主要关心的是代数结构的抽象运算, 而元素用什么表示, 运算用何符号无关紧要, 因此, 我们需要建立两个代数结构的比较方法"[5]。

看看下面两个代数结构  $(A, \cdot), (B, \circ)$

$\cdot$	N	Y
N	N	N
Y	N	Y

$\circ$	0	1
0	0	0
1	0	1

显然, 凭直觉, 我们可以看出这两个结构相同, 但对于复杂代数结构, 我

们如何判断? 当两个代数结构不完全相同时, 我们又如何探讨他们某个侧面的相同性? 同态和同构的概念就是解决这个问题的。

**定义 8.24 (同态 (homomorphism))** 设  $(X, \bullet), (Y, *)$  是两个群, 如果存在一个映射  $f: X \rightarrow Y$ , 使得对任意  $x_1, x_2 \in X$ , 都有  $f(x_1 \bullet x_2) = f(x_1) * f(x_2)$ , 称  $f$  是一个从  $(X, \bullet)$  到  $(Y, *)$  的同态映射, 或称这两个群同态, 记为  $(X, \bullet) \sim (Y, *)$ , 简记为  $X \sim Y$ 。如果  $f$  是单射, 称为单同态,  $f$  是满射, 称为满同态。

**定义 8.25 (同构 (isomorphism))** 如果  $(X, \bullet) \sim (Y, *)$ , 并且映射  $f$  是双射, 则称这两个群同构, 记为  $(X, \bullet) \cong (Y, *)$ , 简记为  $X \cong Y$ 。

一个群到自身的同态叫自同态, 到自身的同构叫自同构。当我们根据定义判断两个

<sup>8</sup>在不引起混淆的情况下, 有时将  $\odot$  也写为  $\bullet$ 。



代数结构是否同构或者同态时，关键是，是否能找到或者构造出一个映射？

### 例 8.26

群  $(\mathbb{Z}, +)$  到群  $(\mathbb{Z}_n, +)$  的映射  $f: \mathbb{Z} \rightarrow \mathbb{Z}_n$  为  $f(a) = a \bmod n$  是一个同态映射。

### 例 8.27

群  $(\mathbb{R}, +)$  和  $(\mathbb{R}^+, \times)$ <sup>9</sup> 同构。在此构造映射  $f(x) = e^x, f(a+b) = e^{a+b} = f(a) \times f(b)$ , 其存在函数,  $g(x) = \ln x, g(a \times b) = \ln(a \times b) = \ln a + \ln b = g(a) + g(b)$ ,  $\mathbb{R}, \mathbb{R}^+$  存在双射。

在此类教科书中，通常介绍完同态和同构定义后，都会后继进行展开讨论，但我们往往看到，后面的讨论都是围绕同态展开，为什么？难道是遗忘了？其实这个原因是，如果两个代数结构同构，那么从结构的角度看，他们是一样的。

这也就是通常在不同的领域，如果两个结构式同构的，那么一个结构中成立的命题在另一个结构中也成立，比如实数序偶加法和平面向量加法，是两个同构的代数结构，这也就是为什么在计算这类问题时，我们可以根据需要随意的切换到序偶运算和向量运算，而不用担心计算结果的一致性。

**定理 8.23 (同态性质)** 两个群满足  $(S, \bullet) (G, \odot)$ ,  $e$  和  $e'$  分别是他们的单位元，同态映射为  $f: S \rightarrow G$ ，则有：

- (1)  $f(e) = e'$ ;
- (2)  $\forall a \in S, f(a^{-1}) = f(a)^{-1}$ ;
- (3)  $\forall n \in \mathbb{Z} \quad a \in S, f(a^n) = f(a)^n$ ;

**定义 8.26 (同态映射的核、像集合)** 两个群满足  $(S, \bullet) (G, \odot)$ ,  $e$  和  $e'$  分别是他们的单位元，同态映射为  $f: S \rightarrow G$ ，令集合  $\ker f = \{a \mid a \in S, f(a) = e'\}$ ，称集合  $\ker f$  为同态映射  $f$  的核；令集合为  $\text{im } f = f(S) = \{f(a) \mid a \in S\}$ ，称集合  $\text{im } f$  为同态映射  $f$  的像。

**定理 8.24 (核子群和像子群)** 两个群满足  $(S, \bullet) (G, \odot)$ ,  $e$  和  $e'$  分别是他们的单位元，同态映射为  $f: S \rightarrow G$ ，则有：

- (1)  $\ker f \leq S$ ,  $\ker f$  称为同态映射  $f$  的核子群，且  $f$  是单同态的充要条件是  $\ker f = \{e\}$ ;
- (2)  $\text{im } f \leq G$ ,  $\text{im } f$  称为同态映射  $f$  的像子群，且  $f$  是满同态的充要条件是  $f(S) = G$ ;
- (3) 如果  $G' \leq G, f^{-1}(G') = \{a \mid a \in S, f(a) \in G'\}$ ，则  $f^{-1}(G') \leq S$ 。

**定理 8.25 (核子群的正规性)** 两个群满足  $(S, \bullet) (G, \odot)$ ,  $e$  和  $e'$  分别是他们的单位元，同态映射为  $f: S \rightarrow G$ ，则有  $\ker f \triangleleft S$ 。

**定理 8.26 (同态商群构造)** 两个群满足  $(N, \bullet) \triangleleft (S, \odot)$ ，构造商群  $(S/N, \odot)$ ，且定义映射  $f: S \rightarrow S/N, f(a) = aN$ ，则  $f$  是一个同态映射，且  $\ker f = N$ 。

**定理 8.27 (同态基本定理)** 设  $f: S \rightarrow G$  是群  $(S, \bullet)$  到群  $(G, \times)$  的同态映射，则存在  $S/\ker f$  到  $\text{im } f$  的一一映射  $h: S/\ker f \rightarrow \text{im } f$ ，使得  $(S/\ker f, \odot) = (\text{im } f, \times)$ 。

<sup>9</sup> $\mathbb{R}^+$  表示正实数集合



## 第9章 环 (ring)

### 9.1 环 (ring)

群时一种由集合和集合上的一个二元运算构成的数学结构，显然他并不能包含所有的代数结构，例如，我们如果考虑有两个二元运算呢？环就是有两个二元运算的代数结构。

**定义 9.1 (环 (ring) 和交换环 (commutative ring))** 设  $R$  是一个给定的集合，在其上定义了两种二元运算  $+$ ,  $\circ$ ，且满足以下条件：

(1)  $(R, +)$  是一个交换群。

(2)  $(R, \bullet)$  是一个半群。

(3)  $a, b, c \in R, a \bullet (b + c) = (a \bullet b) + (a \bullet c)$ 。

我们称  $(R, +, \bullet)$  为环，若  $(R, \bullet)$  是一个交换群，则称其为交换环。

为了后面描述方便，将运算  $+$  下的单位元称为环的零元，记为  $0$ ，元素  $a$  在  $+$  运算下的逆元称为元素  $a$  的负元，记为  $-a$ 。如果  $R$  中存在运算  $\bullet$  的单位元<sup>1</sup>，称为环的么元，记为  $1$ ，如果元素  $a$  在运算  $\bullet$  下存在逆元，称该逆元为元素  $a$  的逆元，记为  $a^{-1}$ ,  $a$  称为可逆元素。

#### 例 9.1

可以根据定义验证， $(\mathbb{Z}, +, \times)$  是一个交换环，零元是  $0$ ，么元是  $1$ ，可逆元素只有  $-1$  和  $1$ 。 $(\mathbb{Q}, +, \times), (\mathbb{R}, +, \times), (\mathbb{C}, +, \times)$  都是交换环，零元都是  $0$ ，么元都是  $1$ ，除了  $0$  以外，所有其它元素都是可逆元素。

**定义 9.2 (环特征 (characteristic of ring))** 环特征也称为环特征数，对于环  $(R, +, \bullet)$ ，若存在自然是  $n$ ，对于  $\forall a \in R$ ，有  $n \bullet a = 0$ ，则称具有这种性质的最小自然数  $n$  称为此环的特征，记为  $\text{Char}(R) = n$ 。若不在这样的自然数满足此性质，则称此环的特征数为零元“ $0$ ”。

#### 例 9.2

$R$  是一个布尔环，求  $\text{Char}(R)$ 。

**解** 我们有： $0 + 0 = 0, 1 + 1 = 0$ ，所以我们可以说  $\forall x \in R, x + x = 2x = 0$ ，根据环特征的定义，我们有  $\text{Char}(R) = 2$ 。

**定义 9.3 (零因子 [4])**  $R = (S, +, \times)$  是一个环，如果存在  $a, b \in S, b \neq 0, a \neq 0$ ，但  $ab = 0$  ( $0$  为  $+$  运算的单位元) 成立，称环  $R$  是有零因子环， $a$  为  $R$  的左零因子 (left zero divisor)， $b$  是  $R$  的右零因子 (right zero divisor)，否则  $R$  是无零因子环。若  $a$  既是左零因子又是右零因子，称  $a$  为零因子。环内既不是左零因子，也不是右零因子的元素称为正则元。

**定义 9.4 (么环)**  $(R, +, \times)$  是一个环，若  $(R, \times)$  是一个含么元的半群， $(R, +, \times)$  称为么环。

<sup>1</sup>由于  $(R, \bullet)$  是半群，所以不一定有单位元和逆元，所以在描述中使用“如果”。

**定义 9.5 (无零因子环)**  $(R, +, \times)$  是一个环, 若任意两个元素  $a, b \in R, ab \neq 0$ ,  $(R, +, \times)$  称为无零因子环。

**定义 9.6 (整环)**  $(R, +, \times)$  是无零因子的幺环, 则称为整环。

**定义 9.7 (体)**  $(R, +, \times)$  是一个环, 若非零元对  $\times$  构成群, 则称为体 (或除环)。

## 9.2 子环

**定义 9.8 (子环)** 如果环  $R$  的一个子集  $S$  满足以下三个条件:

(1)  $0 \in S$ ;

(2)  $a, b \in S \Rightarrow a - b \in S$ ;<sup>2</sup>

(3)  $a, b \in S \Rightarrow ab \in S$ ;<sup>3</sup>

称  $S$  是  $R$  的子环,  $R$  是  $S$  的扩环 (或扩张)。如果  $S=R$  或  $S=\{0\}$ , 显然  $S$  是  $R$  的子环, 称为平凡子环, 其他子环称为真子环。

## 9.3 同态和同构

**定义 9.9 (同态和同构)**  $X$  和  $Y$  是两个环, 如果存在一个映射  $f: X \rightarrow Y$ , 使得  $\forall x_1, x_2 \in X$ , 有  $f(x_1 + x_2) = f(x_1) + f(x_2)$ ,  $f(x_1 \bullet x_2) = f(x_1) \bullet f(x_2)$ , 则称  $f$  是一个从  $X$  到  $Y$  的同态映射或称环  $X$  和  $Y$  同态, 记作  $XY$ , 其中  $+$  和  $\bullet$  是两个环中相应的“加法”和“乘法”。如果  $f$  是单射, 称为单同态,  $f$  是满射, 称为满同态, 如果  $f$  是双射, 称此同态为同构记为  $X = Y$ 。

## 9.4 理想和商环

**定义 9.10 (理想)**  $I$  是环  $R$  的子环, 如果满足  $RI \subset I (\forall i \in I, r \in R, ri \in I)$ , 则  $I$  是  $R$  的左理想, 类似可以定义右理想, 同时左理想和右理想的子环称为理想。

理想是一类特殊的子环, 对于交换环来说, 左理想就是右理想就是理想。

### 例 9.3

证明  $n\mathbb{Z}$  是交换环  $(\mathbb{Z}, +, \times)$  的一个理想。

**定理 9.1 (商环)**  $R'$  是环  $(R, +, \bullet)$  的子环, 则可在  $R$  中定义等价关系:  $a, b \in R, a \sim b \Leftrightarrow a - b \in R'$ .  $a$  所在的等价类记为  $a + R'$ . 若  $R'$  是  $R$  的理想, 则可在商集合  $R/ = R/R'$  中定义  $+, \bullet$  为:  $(a + R') + (b + R') = a + b + R', (a + R') \bullet (b + R') = ab + R'$ . 可知集合  $R/$  对上述定义的  $+, \bullet$  构成环, 称为  $R$  对  $R'$  的商环。

**定义 9.11 (理想的生成元)**  $(R, +, \bullet)$  是一个交换环,  $H$  是  $R$  的非空子集,  $\{H_i \mid i \in \mathbb{N}\}$  是  $R$  的所有包含集合  $H$  的理想, 即  $H \subseteq H_i (i \in \mathbb{N})$ , 则  $\bigcap_{i \in \mathbb{N}} H_i$  称为由子集  $H$  生成的理想, 记

<sup>2</sup> $a-b$  是  $a+(-b)$  的简写, 否则-这个运算并没有定义, 后面都使用相同的简写方式。

<sup>3</sup> $ab$  是  $a \bullet b$  的简写, 后面使用相同的简写方式。

为  $(H)$ ,  $H$  中的元素叫做理想  $(H)$  的生成元。如果  $H = \{a_1, a_2, \dots, a_n\} (n \in \mathbb{N})$ , 则理想  $(H)$  记为  $(a_1, a_2, \dots, a_n)$ , 并称为有限生成的理想, 由一个元素生成的理想  $\langle a \rangle$  叫主理想。

**定义 9.12 (主理想环)** 如果交换环  $R$  的所有理想都是主理想, 则交换环  $R$  称为主理想环。

#### 例 9.4

求证  $(\mathbb{Z}, +, \times)$  是主理想环。

## 9.5 多项式环

**定义 9.13 (环上的一元多项式)**  $(R, +, \times)$  交换环,  $x$  是一个变元,  $n$  是非负整数,  $a_0, a_1, \dots, a_n \in R$ , 则  $f(x) = a_0 + a_1x + \dots + a_nx^n$ <sup>4</sup> 称为交换环  $R$  上的一元多项式,  $a_0, a_1, \dots, a_n$  称为此多项式的系数,  $a_0$  称为常数项; 如果  $a_n \neq 0$ ,  $a_n$  称为首项系数,  $n$  称为一元多项式  $f(x)$  的次数, 记为  $\deg f(x) = n$ 。所有交换环  $R$  上的一元多项式组成的集合记为  $R[x]$ 。

**定义 9.14 ( $R$  上的一元多项式环)**  $(R, +, \times)$  是交换环, 称  $(R[x], +, \times)$  为  $R$  上的一元多项式环, 也称为  $R$  上添加  $x$  生成的环。

**定义 9.15 ( $R$  上的  $n$  元多项式环)**  $(R, +, \times)$  是交换环, 称  $(R[x_1, \dots, x_n], +, \times)$  为  $R$  上的  $n$  元多项式环。

**定义 9.16 (代数元)** 如果在交换幺环  $R$  中存在有限多个元素  $a_1, \dots, a_n, a_n \neq 0$ , 使得  $a_nu^n + \dots + a_1u + a_0 = 0$ , 称  $u$  为  $R$  上的代数元, 使上述关系成立的最小正整数  $n$  称为代数元的次数, 记为  $\deg(u, R)$ , 称  $f(x) = a_nx^n + a_1x + a_0 \in R[x]$  为  $u$  在  $R$  上的不可约多项式, 记为  $\text{Irr}(u, R)$ 。

**定义 9.17 (超越元)** 如果在交换幺环  $R$  中任意不全为零元素  $a_1, \dots, a_n$ , 均有  $a_nu^n + \dots + a_1u + a_0 \neq 0$ , 称  $u$  为  $R$  上的超越元。

<sup>4</sup>这里的  $+$  和  $\times$  只是一个运算的记号, 不代表任何特定的运算。

## 第 10 章 域 (field)

### 10.1 基本概念

**定义 10.1 (域 (field))**  $(R, +, \times)$  是一个环, 若非零元对  $\times$  构成阿贝尔群 (交换群), 则称域。

**定义 10.2 (有限域 (finite field 或 Galois field))** 包含有限个元素的域称为有限域或伽罗瓦域。

**定义 10.3 (有限域的阶 (Order))** 有限域的元素个数称为它的阶 (order), 如果一个有限域阶为  $q$ , 通常这个有限域我们记为  $GF(q)$ 。

有限域最常见的例子是当  $p$  为素数时, 整数对  $p$  取模就可以形成此有限域,  $GF(p)$  是一个素域。

- 取大质数  $p$ , 则有限域中有  $p$  个有限元:  $0, 1, 2, \dots, p-1$
- $GF(p)$  上的加法为模  $p$  加法  $a + b = c \pmod{p}$
- $GF(p)$  上的乘法为模  $p$  乘法  $a \times b = c \pmod{p}$
- $GF(p)$  上的除法就是乘除数的乘法逆元  $a \div b = c \pmod{p}$ , 即  $a \times b^{-1} = c \pmod{p}$
- $GF(p)$  的乘法单位元为 1, 零元为 0
- $GF(p)$  域上满足交换律, 结合律, 分配律

**定义 10.4 (域特征 (characteristic of field))** 对于域  $F$ , 若其乘法群单位元为 1, 加法群单位元为 0, 若存在正整数  $p$ , 是的  $p$  个 1 相加为 0, 那么满足这个条件的最小  $p$  称为域  $F$  的特征。

可以证明, 域的特征为 0 或素数。

### 10.2 域上的多项式

域的概念产生于方程求解。

**定义 10.5 (域上的一元多项式)**  $F$  是域, 若  $a_0, a_1, \dots, a_n \in F, a_n \neq 0$ , 则  $f(x) = a_0 + a_1x + \dots + a_nx^n$  为域  $F$  上的一元多项式或多项式, 称  $n$  为该多项式的次数, 记为  $\deg f = n$ . 若记  $F[x] = \{a_0 + a_1x + \dots + a_nx^n \mid a_0, a_1, \dots, a_n \in F\}$ , 则  $F[x]$  构成环, 称之为  $F$  上的一元多项式环或多项式环。

**定义 10.6 (整除, 可约或不可约 [5])** 若  $F$  上的多项式  $f(x)$  等于  $F$  上其他两个非零次多项式  $g(x), h(x)$  的乘积, 即  $f(x) = g(x)h(x)$ , 且  $\deg g$  和  $\deg h$  均不为 0, 则称多项式  $f(x)$  是可约的,  $g(x), h(x)$  称为  $f(x)$  的因式或  $g(x), h(x)$  整除  $f(x)$ , 记为  $g(x) \mid f(x)$ ; 否则称之为不可约, 记为  $g(x) \nmid f(x)$ 。

**定理 10.1 (带余除法 [5])** 若  $f(x), g(x) \in F[x], g(x) \neq 0$ , 则存在唯一的  $q(x), r(x) \in F[x]$ , 使  $f(x) = q(x)g(x) + r(x)$ , 其中  $\deg r < \deg g$  或  $r(x) = 0$ 。这里, 称  $q(x)$  为  $g(x)$  除  $f(x)$

的商式,  $r(x)$  为  $g(x)$  除  $f(x)$  的余式。

**定义 10.7 (最大公因式 [5])** 若  $d(x)$  为  $f(x)$  和  $g(x)$  公因式, 且  $d(x)$  能被  $f(x), g(x)$  任何一个公因式整除, 则称  $d(x)$  是  $f(x)$  和  $g(x)$  的最大公因式, 记为  $\gcd(f(x), g(x))$  或  $(f(x), g(x))$ 。

**定理 10.2 (最大公因式表示)** 若  $d(x)$  为  $f(x)$  和  $g(x)$  公因式, 且  $d(x)$  能被  $f(x), g(x)$  任何一个公因式整除, 则称  $d(x)$  是  $f(x)$  和  $g(x)$  的最大公因式, 记为  $\gcd(f(x), g(x))$  或  $(f(x), g(x))$ 。

**定义 10.8 (互素多项式 [5])** 若  $\gcd(f(x), g(x)) = 1$ , 则称  $f(x), g(x)$  互素, 记为  $\gcd(f(x), g(x)) = 1$ 。

**定理 10.3 (互素多项式性质 [5])**  $\gcd(f(x), g(x)) = 1 \Leftrightarrow \exists u(x), v(x) \in F[x], u(x)f(x) + v(x)g(x) = 1$ 。

**定理 10.4 (互素多项式性质 [5])**  $\gcd(f(x), g(x)) = 1, f(x) \mid g(x)h(x) \Rightarrow f(x) \mid h(x)$ 。

**定理 10.5 (因式分解唯一定理)** 设  $F$  是一个域,  $F$  上的任何一个次数大于等于 1 的多项式  $f(x)$  都可分解成  $F[x]$  中若干不可约多项式的乘积, 若不考虑因式的次序, 分解式唯一的。

**定义 10.9 (多项式的根 [5])**  $f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \in F[x]$ , 若  $\alpha \in F$ , 使  $f(\alpha) = a_n \alpha^n + a_{n-1} \alpha^{n-1} + \dots + a_1 \alpha + a_0 = 0$ , 则称  $\alpha$  为  $f(x)$  在  $F$  中的根。

### 10.3 同态和同构

**定义 10.10 (同构、自同构)** 设  $F_1, F_2$  为域,  $\delta: F_1 \rightarrow F_2$ , 满足  $\forall a, b \in F_1, \delta(a+b) = \delta(a) + \delta(b), \delta(ab) = \delta(a)\delta(b)$ , 则称  $\delta$  为  $F_1$  到  $F_2$  的同构, 若  $F_1 = F_2$ , 则称  $\delta$  自同构。

### 10.4 域的代数扩张

**定义 10.11 (扩域 [4])** 设集合  $K'$  是域  $K$  集合的非空子集, 如果对于域  $K$  的运算,  $K'$  可构成一个域, 则  $K'$  叫作  $K$  的子域,  $K$  叫作域  $K'$  的扩域。

**定义 10.12 (素域)** 不包含任何非平凡子域的域称为素域。

**定义 10.13 (添加  $S$  所得的域)** 设  $K$  为域  $F$  的扩域,  $S$  为  $K$  的子集,  $K$  中所有包含  $F \cup S$  的子域的交, 即由  $F$  与  $S$  生成的子域, 称为  $F$  上添加  $S$  所得的域, 记为  $F(S)$ 。

**定义 10.14 (代数元和超越元)** 设  $K$  为域  $F$  的扩域,  $\alpha \in K$ , 若存在域上的非零多项式  $f(x)$  满足  $f(\alpha) = 0$ , 则称  $\alpha$  为  $F$  上的代数元, 否则称  $\alpha$  为  $F$  上的超越元。  $K$  包含的  $F$  上的代数元的集合, 称为  $F$  在  $K$  中的代数闭包,  $F$  上所有代数元的集合称为  $F$  的代数闭包, 记为  $\bar{F}$ 。

**定义 10.15 (代数闭域)** 域  $K$  是自身的代数闭包, 即  $K$  上多项式的根均在  $K$  中, 则称  $K$  为代数闭域。

**定义 10.16 (扩张)** 设  $K$  为域  $F$  的扩域, 且  $\alpha \in K, K = F(\alpha)$ , 则称  $K$  为  $F$  的单扩张。若  $\alpha$  为  $F$  上的代数元, 则称  $K$  为  $F$  的单代数扩张; 若  $\alpha$  为  $F$  上的超越元, 则称  $K$  为  $F$  的单超越扩张。

**定义 10.17 (不可约多项式)** 设  $K$  为域  $F$  的扩域, 且  $\alpha \in K$  为  $F$  上的代数元,  $F[x]$  中以  $\alpha$  为根的不可约的首一多项式称为  $\alpha$  在  $F$  上的不可约多项式, 记为  $\text{Irr}(\alpha, F)$ , 其次数称为  $\alpha$  在  $F$  上的次数, 记为  $\deg(\alpha, F)$ 。

**定义 10.18 (等价扩张)** 设  $K_1, K_2$  为域  $F$  的扩域, 若存在  $K_1, K_2$  同构  $\phi$ , 使得  $\phi|_F = \text{id}_F$ , 则称  $K_1, K_2$  为  $F$  的等价扩张, 称  $\phi$  为  $F$  同构, 若  $K_1 = K_2, \phi$  为  $F$  自同构。

**定义 10.19 (代数扩张)** 设  $K$  为域  $F$  的扩域, 若  $K$  中的每个元素都是  $F$  上的代数元, 则称  $K$  为  $F$  的代数扩张。

**定义 10.20 (有限扩张, 维数)** 设  $K$  为域  $F$  的扩域,  $K$  作为  $F$  上的线性空间是有限维的, 则称  $K$  为  $F$  的有限扩张, 该维数称为  $K$  在  $F$  上的维数, 记为  $[K : F]$ ; 若  $K$  作为  $F$  上的线性空间是无限维的, 则称  $K$  为  $F$  的无限扩张。

## 第 11 章 椭圆曲线 (elliptic curve)

椭圆曲线是黎曼几何的一个研究内容，也是代数几何中重要的一类研究对象，一个椭圆曲线是一个连通黎曼曲面，关于椭圆曲线的详细介绍，可以参考 Joseph H. Silverman 的 “The Arithmetic of Elliptic Curves” (椭圆曲线算数) 一书。下面我们引用一段椭圆曲线的英文介绍。

*Classically in complex geometry, an elliptic curve is a connected Riemann surface (a connected compact 1-dimensional complex manifold) of genus 1, hence it is a torus equipped with the structure of a complex manifold, or equivalently with conformal structure.*

*The curious term “elliptic” is a remnant from the 19th century, a back-formation which refers to elliptic functions (generalizing circular functions, i.e., the classical trigonometric functions) and their natural domains as Riemann surfaces.*

*In more modern frameworks and in the generality of algebraic geometry, an elliptic curve over a field  $k$  or indeed over any commutative ring may be defined as a complete irreducible non-singular algebraic curve of arithmetic genus-1 over  $k$ , or even as a certain type of algebraic group scheme.*

*Elliptic curves have many remarkable properties, and their deeper arithmetic study is one of the most profound subjects in present-day mathematics.<sup>1</sup>*

### 11.1 基本概念

椭圆曲线 (elliptic curve) 是指由 Weierstrass 韦尔斯特拉 (中文翻译有韦尔斯特拉、魏尔斯特拉斯) 方程确定的平面，韦尔斯特拉方程为： $E: y^2 + axy + by = x^3 + cx^2 + dx + e$ ,  $E$  是 Elliptic curve 的缩写，表示这个方程描述了一个椭圆曲线，其中  $a, b, c, d$  和  $e$  属于域  $F$ ， $F$  可以是有理数域、复数域、有限域，密码学中通常采用有限域。

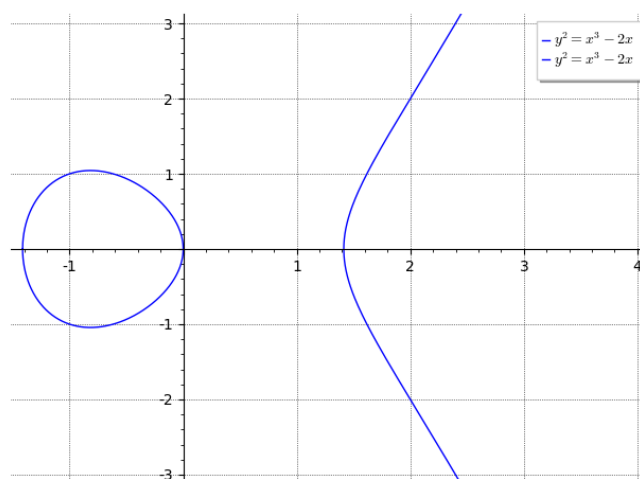
下面我们使用 SageMath 绘制几个实数域上的椭圆曲线。

**例 11.1** 椭圆曲线： $y^2 = x^3 - 2x$

sagemath: 椭圆曲线

```
# y^2=x^3-2x
p= plot(EllipticCurve([0,0,0,-2,0]),gridlines='true', xmin=-4, xmax=4,
        ymin=-3, ymax=3,legend_label='$y^2=x^3-2x$')
show(p)
```

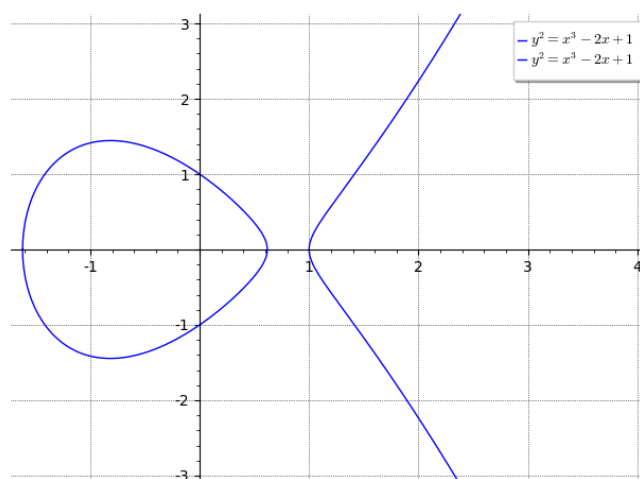
<sup>1</sup>来源于 <https://ncatlab.org/nlab/show/elliptic+curve>

图 11.1: 椭圆曲线  $y^2 = x^3 - 2x$ 

例 11.2 椭圆曲线:  $y^2 = x^3 - 2x + 1$

**sagemath: 椭圆曲线**

```
# y^2=x^3-2x
p= plot(EllipticCurve([0,0,0,-2,1]),gridlines='true', xmin=-4, xmax=4,
        ymin=-3, ymax=3,legend_label='$y^2=x^3-2x+1$')
show(p)
```

图 11.2: 椭圆曲线  $y^2 = x^3 - 2x + 1$ 

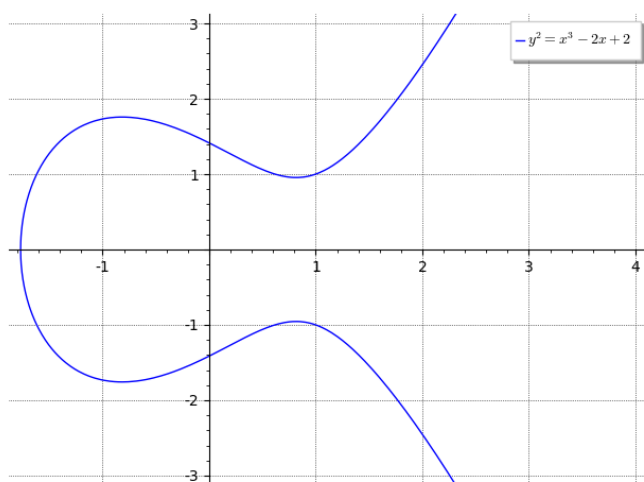
例 11.3 椭圆曲线:  $y^2 = x^3 - 2x + 2$

**sagemath: 椭圆曲线**

```
# y^2=x^3-2x+2
p= plot(EllipticCurve([0,0,0,-2,2]),gridlines='true', xmin=-4, xmax=4,
        ymin=-3, ymax=3,legend_label='$y^2=x^3-2x+2$')
show(p)
```



show(p)

图 11.3: 椭圆曲线  $y^2 = x^3 - 2x + 2$ 

通过定义恰当的“加法”运算，椭圆曲线上的点全体构成一个加法群，正因为椭圆曲线存在加法结构，所以它包含了很多重要的数论信息。下面我们看看椭圆曲线上的加法群的构建。

- 单位元： $O$  为单位元，椭圆曲线上的所有点  $P$  有  $P + O = P$ ， $O$  也是一个椭圆曲线上的一个点，是一个无穷远的点。
- 逆元：对点  $P = (x, y)$ ，其加法逆元为  $(x, -y)$ ，记为  $-P$ ， $P + (-P) = O$ ，由此也可以定义减法  $P - P = O$
- 加法：对于两个不同且不互逆的点  $P, Q$ ，我们画一条通过  $P, Q$  的直线，与椭圆曲线交于一点，这个交点是唯一的（除非所做的直线是  $P$  或  $Q$  的切线），此交点的逆元为  $R$ ，定义  $P + Q = R$ 。加法定义如图 11.4 所示。
- 倍数：点  $P$  的倍数定义为，在  $P$  点做椭圆曲线的一条切线，设切线与椭圆曲线交于  $n$  个点  $Q$

一点， $R$  为此交点的逆元，定义  $2P = P + P = R$ ，一般将  $\overbrace{Q + Q + \dots + Q}^{n \text{ 个 } Q}$  记为  $nQ$ 。可以证明以上定义的加法运算具有交换律和结合律等一般性质。

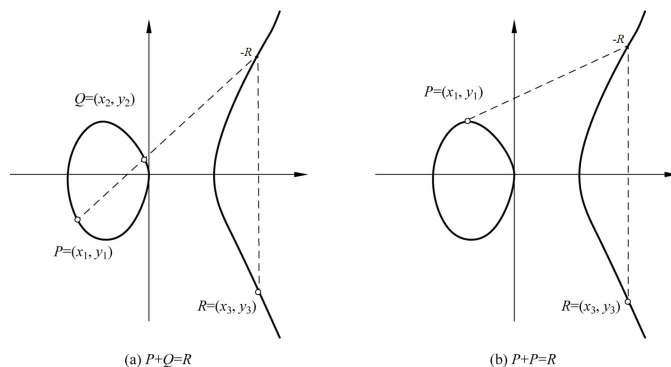


图 11.4: 椭圆曲线加法定义图示

## 11.2 有限域上的椭圆曲线

为了简单起见, 我们通常考虑在有限域  $GF(p)$  上的椭圆曲线,  $p$  为大于 3 的素数, 在有限域  $GF(p)$  上的曲线  $y^2 = x^3 + ax + b$  ( $a, b \in GF(p), 4a^3 + 27b^2 \neq 0$ ) 称为有限域上的椭圆曲线, 通常记为  $E_p(a, b)$ 。

椭圆曲线的定义也要求曲线是非奇异的 (即处处可导的)。几何上来说, 这意味着图像里面没有尖点、自相交或孤立点。代数上来说, 这成立当且仅当判别式  $\delta = 4a^3 + 27b^2$  不等于 0, 这里主要是满足其可导性。

我们看看同一方程在不同域上的曲线。

**例 11.4** 椭圆曲线:  $y^2 = x^3 - 2x$  在有限域  $GF(11)$  和实数域上表示的曲线。

### sagemath: 椭圆曲线

```
# y^2=x^3-2x在有限域GF(11)
p=plot(EllipticCurve(GF(11),[0,0,0,-2,0]),gridlines='true',
xmin=-11, xmax=11, ymin=-30, ymax=30,
legend_label='$y^2=x^3-2x, GF(11)$')
# y^2=x^3-2x在实数域
p+=plot(EllipticCurve([0,0,0,-2,0]),gridlines='true', color=hue(1),
xmin=-11, xmax=11, ymin=-30, ymax=30, legend_label='$y^2=x^3-2x$')
show(p)
```

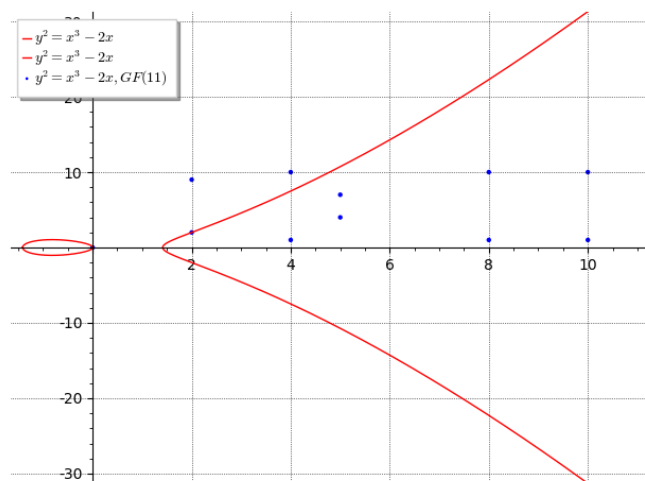


图 11.5: 椭圆曲线  $y^2 = x^3 - 2x$  在有限域  $GF(11)$  和实数域上的图

有限域椭圆曲线构成的群如下:

- 单位元:  $O$  为单位元, 椭圆曲线上的所有点  $P$  有  $P + O = P$ ,  $O$  也是一个椭圆曲线上的一个点, 是一个无穷远的点。
- 逆元: 对点  $P = (x, y)$ , 其加法逆元为  $(x, -y)$ , 记为  $-P$ ,  $P + (-P) = O$ , 由此也可以定义减法  $P - P = O$

- 加法: 对于两个不同且不互逆的点  $P(x_1, y_1), Q(x_2, y_2), x_1 \neq x_2, P(x_1, y_1) + Q(x_2, y_2) = S(x_3, y_3)$ , 其中:

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p}$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

- 倍数: 点  $P$  的倍数定义为,  $2P = P(x_1, y_1) + P(x_1, y_1) = S(x_3, y_3)$ , 其中:

$$x_3 = \lambda^2 - 2x_1 \pmod{p}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p}$$

$$\lambda = \frac{3x_1^2 + a}{2y_1}, \text{ 其中 } a \text{ 是方程中的常数。}$$

假设  $E/F_p$  是一个椭圆曲线,  $e \geq 1$ , 如果  $(x_1, y_1), x_1, y_1 \in F_{p^e}$ , 满足曲线方程  $E$ , 我们说点  $(x_1, y_1)$  在椭圆曲线上。当  $e = 1$  时, 点  $(x_1, y_1)$  定义在基础域 (base field)  $F_p$  上, 当  $e > 1$  时, 点  $(x_1, y_1)$  定义在域  $F_p$  的扩展上。在域  $F_{p^e}$  上曲线  $E$  上的所有点, 包括无穷远点 (the point of infinity)  $O$ , 我们记为  $E(F_{p^e})$ , 用  $|E(F_{p^e})|$  表示椭圆曲线上点的个数。

根据哈赛 (Hase) 的研究结果, 我们有  $|E(F_{p^e})| = p^e + 1 - t$ , 其中  $|t| \leq 2\sqrt{p^e}$ , 这表明  $|E(F_{p^e})|$  非常接近  $p^e - 1$ 。对于  $E(F_p)$  来说, 这个值为  $p + 1$ .[6]

**例 11.5** [7]  $GF(11)$  上的一个椭圆曲线  $E_{11}(1, 6) : y^2 = x^3 + x + 6 \pmod{11}$  构成的交换群。

**解** 1、计算椭圆曲线上所有的点

对于  $GF(11)$  上的每一个点  $x$  计算  $s = x^3 + x + 6 \pmod{11}$ , 然后求解  $y^2 = s \pmod{11}$ , 如果此方程有解 (即  $s$  为模 11 的平方剩余), 解为  $y$ , 则  $(x, \pm y)$  是  $E_{11}(1, 6)$  上的点, 按照这个方法, 我们可以获得  $E_{11}(1, 6)$  上的点, 共 12 个点:

$(2, 4), (2, 7), (3, 5), (3, 6), (5, 2), (5, 9), (7, 2), (7, 9), (8, 3), (8, 8), (10, 2), (10, 9)$

2、交换群

我们可以看到  $(2, 4)$  和  $(2, 7)$  互为逆元, 下面我们计算  $(2, 4) + (3, 5)$ :

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{5 - 4}{3 - 2} = 1$$

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{11} = 1^2 - 2 - 3 \pmod{11} = 7$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{11} = (2 - 7) - 4 = 2$$

$(2, 4) + (3, 5) = (7, 2)$ , 可以看到  $(7, 2)$  仍然是椭圆曲线上的点, 可见加法运算在  $E_{11}(1, 6)$  上是封闭的。

**例 11.6** [8] 在  $E_{23}(1, 1)$  上计算  $P + Q, P = (3, 10), Q = (9, 7)$ 。

**解**  $\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{7 - 10}{9 - 3} = \frac{-3}{6} = \frac{-1}{2} = 11 \pmod{23}$ <sup>2</sup>

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{23} = 11^2 - 3 - 9 \pmod{23} = 17$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{23} = 11(3 - 17) - 10 = -164 = 20$$

$P + Q = (17, 20)$ , 结果仍然为  $E_{23}(1, 1)$  中的点。

<sup>2</sup>此处计算解释, 我们需要找到一个数  $a, 2a = 1 \pmod{23}$ , 可知  $a = 12, -12 \pmod{23} = 11$

## 11.3 构造密码算法

要想利用椭圆曲线构造密码算法,从大的方面来说,要解决以下几个问题:

1. 将一个信息,也就是一个二进制串或者一个数,映射到椭圆曲线的一个点,同时也可将椭圆曲线上的一个点映射到一个信息。
2. 然后在椭圆曲线上找到一个计算困难问题,但这个困难问题当有某个信息时要变得不困难。
3. 设计一个信息变换或计算过程。

下面我们依次看看以上这几个问题的考虑。

### 11.3.1 消息到椭圆曲线上的映射

设消息是  $m, 0 \leq m \leq M, E: y^3 = x^2 + ax + b$ , 给定一个整数  $k$ , 实际中  $k$  在 30 50 之间取值, 在这里我们取  $k=30$ , 对明文  $m$  计算一系列  $x, x = \{mk + j, j = 0, 1, 2, \dots\} = \{30m + j, j = 0, 1, 2, \dots\}$ , 直到  $x^2 + ax + b \pmod{p}$  是平方根, 那么就将  $m$  映射到椭圆曲线上这一点  $(x, \sqrt{x^2 + ax + b})$ 。

反之, 有一个椭圆曲线上的一点  $(x, y)$ , 我们将其还原为消息  $m$  的过程为  $m = \lfloor \frac{x}{30} \rfloor$ 。

### 11.3.2 椭圆曲线上的计算困难问题

在  $E_p(a, b)$  上考虑方程  $Q = kP, Q, P \in E_p(a, b), k < p$ , 我们知道了  $k, P$  很容易求解  $Q$ , 但是知道了  $P, Q$ , 求确  $k$  是一个困难问题, 这就是椭圆曲线上的离散对数问题。

### 11.3.3 椭圆曲线上的密码算法

#### 11.3.3.1 ECC(Elliptic Curve Cryptography) 加密算法

通常软件实现采用的域为  $GF(p)$  域, 在硬件实现的采用  $GF(2^m)$  域, 下面我们看一个经典的  $GF(p)$  域上的 ECC 算法。[9]

ECC 属于公钥密码体制, 下面假设用 Alice 给 Bob 准备发送一个秘密信息的实例来说明整个过程。

#### (1) Bob 生成公私钥对

- 选择椭圆曲线  $E: y^2 = x^3 + ax + b \pmod{p}$ , 构造群  $E_p(a, b)$ 。
- 在  $E_p(a, b)$  上挑选生成元  $g = (x_0, y_0)$ ,  $g$  应使得满足  $ng = O$  的最小  $n$  是一个非常大的素数。
- 选择一个随机数  $\alpha, \alpha \in [1, n-1]$ , 计算  $\beta = \alpha g$ 。
- Bob 的公钥为  $(E_p(a, b), n, g\beta)$ , 私钥为  $\alpha$ 。

#### (2) Alice 对消息 $m$ 加密

- 选择一个随机数  $k, k \in [1, n-1]$ 。
- 计算点  $C_1 = (x_1, y_1) = kg$ 。

- 随机选择一个点  $P_t = (x_t, y_t)$ , 计算  $C_2 = P_t + k\beta$ .
- 计算密文  $C_3 = mx_t + y_t$ .
- 将  $(C_1, C_2, C_3)$  做为密文发给 Bob。

### (3)Bob 对密文 $(C_1, C_2, C_3)$ 解密

- 使用私钥  $\alpha$  计算  $C_2 - \alpha C_1 = (x'_t, y'_t) = P'_t$ .
- 计算  $m = \frac{(C_3 - y'_t)}{x'_t}$ ,  $m$  即为解密后的明文。

我们简单看看以上过程的正确性:

$$\alpha C_1 = \alpha kg = k\alpha g = k\beta$$

$$(x'_t, y'_t) = C_2 - \alpha C_1 = P_t + k\beta - k\beta = P_t = (x_t, y_t)$$

$$\frac{C_3 - y'_t}{x'_t} = \frac{mx_t + y_t - y'_t}{x'_t} = m$$

### 11.3.3.2 Diffie-Hellman 密钥交换

下面我们介绍一下如何用椭圆曲线进行 DH 密钥交换.

- 选一个素数  $p \approx 2^{180}$  和两个参数  $a, b$ , 构造  $E_p(a, b)$ 。
- 取  $E_p(a, b)$  的一个生成元  $G_1(x_1, y_1)$ , 使  $G$  的阶  $n$  是一个非常大的素数。<sup>3</sup>
- $E_p(a, b), G, n$  公开。
- 用户 A 任选  $n_A, n_A \in [1, n-1]$ ,  $n_A$  为 A 的私钥,  $P_A = n_A G$  是 A 的公钥。
- 用户 B 任选  $n_B, n_B \in [1, n-1]$ ,  $n_B$  为 B 的私钥,  $P_B = n_B G$  是 B 的公钥。
- A、B 双方利用对方的公钥和自身的私钥, 即可产生双方共享的密钥  $K$ , A 计算  $K = n_A P_B$ , B 计算  $K = n_B P_A$

<sup>3</sup>G 的阶是满足  $nG = O$  的最小正整数  $n$ 。

## 第 12 章 信息论 (information theory)

1948 年 C. E. Shannon 发表的 “A Mathematical Theory of Communication” 是公认的信息论、现代通信的奠基之作，其中文译本可以去[https://gitee.com/buuer\\_xxtxiaofeng/InfSecClaT](https://gitee.com/buuer_xxtxiaofeng/InfSecClaT)下载。

### 12.0.1 信源 (information source)

信源，故名思意就是信息的来源，但是如何用数学的语言来定义，或者我们说如何形式化地描述这个概念呢？

**定义 12.1 (信源形式化定义 [1])** 设  $(S, \mathbb{F}, P)$  为任意概率空间， $S$  为基本基本事件集， $\mathbb{F}$  是  $S$  的某些子集构成的集族， $P(A), A \in \mathbb{F}$ ，表示事件  $A$  发生的概率，如果  $S$  是信源输出的符号集，则称此概率空间为一个信源，若  $S$  是离散集，称此信源为离散信源，若  $S$  与实数集等势或是其他连续集，则称此信源为连续信源，若  $S$  的符号中，有一些在离散集中，有一些在连续集中，称此信源为混合信源。

**定义 12.2 (信息量 [1])** 设信源输出的符号取值于  $A = \{a_1, a_2, \dots, a_n\}$ ，若每个信源符号等概率出现，即  $p = \frac{1}{n}$ ，则称  $I = \log n$  为具有  $n$  个等概率值的信源符号的信息量。

**定义 12.3 (信源的熵 [1])** 设随机变量  $X$  取值于  $A$ ， $p_i = P(X = a_i)$ ，称  $I(a_i) = \log \frac{1}{p_i} = -\log p_i$  为符号  $a_i$  所产生的信息量， $I(a_i)$  称为  $a_i$  的自信息量。信息量的数学期望值 (平均值)  $EI(a_i)$  称为信源的平均信息量或信源的信息熵，简称为信息熵，记为  $H(X) = EI(a_i) = E \log \frac{1}{p_i} = \sum_{i=1}^n p_i \log \frac{1}{p_i} = -\sum_{i=1}^n p_i \log p_i$ 。

如果对数取 2 为底，此时熵的单位为比特 (bits)。取 3 为铁特，取 10 为笛特。

**定理 12.1 (最大离散熵定理 [1])** 信源所有符号等概率出现时，信源的熵最大，形式化描述为，对于任意信源  $X$ ， $H(X) \leq \log n$ 。

## 12.1 多信源

**定义 12.4 (两个联合信源的熵 [1])** 给定两个离散信源：

$$[A, p_i] = \left( a_1, a_2, \dots, a_n \right), \sum_{i=1}^n p_i = 1 \quad (12.1)$$

$$[B, q_i] = \left( b_1, b_2, \dots, b_m \right), \sum_{i=1}^m q_i = 1 \quad (12.2)$$

若分别取值于  $A$  和  $B$  的随机变量  $X$  与  $Y$  的联合概率分布为  $r_{ij} = P(X = a_i, Y = b_j)$  ( $i = 1, 2, \dots, n; j = 1, 2, \dots$



## 第 13 章 计算复杂性 (Computational complexity)

### 13.1 简介

这里我们引用 Arora 在其 *Computational Complexity: A Modern Approach*(计算复杂性: 现代方法) 一书 [10] 中的序言部分内容作为本节内容<sup>1</sup>。

如果一个科学分支还能找到大量的研究问题, 则这个分支还活着; 研究问题的缺失则预示着这个科学分支的消亡或者独立发展的终结。——戴维·希尔伯特 (David Hilbert), 1900

我演讲的主题或许可以直接用两个简单的问题来揭示。第一个问题是, 乘法比加法更难吗? 而第二个问题则是, 为什么? ……我(想) 要证明不存在乘法的算法在计算上同加法的算法一样简单, 这就证明了乘法计算中确实存在某种绊脚石。——阿兰·科巴姆 (Alan Cobham), 1964

几千年来, 人们在账目管理和天文学中不断地进行着各种计算, 因此计算的概念以这种形式一直存在着。例如, 利用计算可以求解下面的任务。

- 给定两个整数, 计算它们的乘积。
- 给定  $n$  个变量上  $n$  个方程构成的方程组, 找出它的一个解 (如果解存在的话)。
- 给定一个熟人列表和这些人中彼此不能和睦相处的一些人员对, 找出你在宴会中打算邀请的最大熟人子集使得他们彼此都能够和睦相处。

在历史长河中, 人们总结得出: 在概念上, 计算就是在给定的输入上用有限个步骤得到输出的过程。他们认为, “计算” 就是 “在演草纸上根据一定规则写出一些数字的人”。

20 世纪中叶, 科学上的一项重要突破就是对 “计算” 的精确定义。根据这个定义, 人们马上清楚了计算其实就存在于各种物理系统和数学系统中——图灵机 (Turing machine), 演算 (Lambda calculus), 细胞自动机 (cellular automata), 指针机 (pointer machine), 弹跳桌球 (bouncing billiards ball), 康韦生命游戏 (Conway's game of life), 等等。出人意料的是, 这些形式的计算都是等价的, 也就是说, 其中一个模型能够实现的所有计算也能在其他模型上完成。在这种认识的基础上, 人们马上发明了能够执行所有程序的硬件, 这就是标准的通用电子计算机。在接下来的几十年中, 计算机迅速被社会接纳, 这使得计算融入了现代生活的方方面面, 也使得计算问题在设计、计划、工程、科学发现等人类活动中变得越来越重要, 于是计算机算法 (亦即, 求解计算问题的各种方法) 变得无处不在。

然而, 计算并不 “仅仅” 是一种用于实践的工具, 它也是一个主要的科学概念。事实上, 科学家现在将许多自然现象都视为一种计算过程, 这些计算过程实际上是细胞自动

<sup>1</sup>此书的网页在<https://theory.cs.princeton.edu/complexity/>



机的推广。例如，对生物繁衍过程的理解曾经导致了自体复制计算机的发现。再比如，物理学家薛定谔 (Schrodinger) 曾在他的书中预言细胞中肯定存在类似于 DNA 的物质，后来沃森 (Watson) 和克里克 (Crick) 发现了这种物质，克里克坦承他们的研究正是受到了薛定谔的工作的启发。如今，各种计算模型已经成为生物学和神经科学中许多领域研究的基础。电子电动力学 (QED) 等几种物理理论的本征刻画非常类似于计算的定义，这种现象甚至还促使一些科学家相信整个宇宙就是一台超级计算机。更有趣的是，这样的物理理论在过去的十年中已经被用于设计量子计算机。

### 可计算性与计算复杂性

研究者在成功地给出计算的概念之后，开始研究什么样的问题是可计算的。他们证明了几个有趣的问题本质上都不是可计算的，也就是说，没有计算机能够求解这些问题而不在任何输入上陷入无限循环（即不停机）。可计算性理论是一个很重要的专题。计算复杂性理论，它关注计算的效率，也就是量化地研究求解给定计算任务所需的计算资源的数量。

### 计算效率的量化

我们用前面提到的三个计算任务来阐释计算效率的含义。首先，考虑两个整数的乘法。我们可以用两种不同的方法（或算法）来完成这项任务。第一种方法是累加算法；也就是说，为了计算  $a \cdot b$ ，只需用  $b-1$  次加法将  $b$  个  $a$  进行累加。第二种方法是小学列式算法。虽然累加算法比小学列式算法简单，但我们总觉得后者比前者更好。事实上，小学列式算法的效率更高。例如，计算 577 乘以 423 时，累加算法需要计算 422 次加法；但小学列式算法却只需将其中一个数分别与 3 个一位数相乘，再计算 3 次加法。

算法效率的量化就是研究算法所执行的基本操作个数随着输入规模的增长是如何变化的。在整数乘法中，单个数位相加或相乘就是基本操作（在其他场合中，除法也可以是基本操作），而两个因数中的数位个数就是输入规模。在计算两个  $n$  位整数的乘法时（也就是说，两个因数都大于  $10^n - 1$  且小于  $10^n$ ），小学列式算法执行的基本操作数不超过  $2n^2$ ，而累加算法执行的基本操作数至少是  $n10^n - 1$ 。经过这样的分析，两种算法之间的巨大差异就显而易见了。即使计算两个 11 位整数的乘法，执行小学列式算法的便携式计算器也会快于执行累加算法的超级计算机。对于稍大一些的整数，小学五年级学生用笔和纸执行列式算法也会胜于执行累加算法的超级计算机。由此可见，算法的效率明显比运行算法所用的技术要重要得多。

更令人意外的是，借助傅里叶变换可以设计出更快的乘法算法。这个算法 40 年前才被人们发现，在这个算法中，两个  $n$  位整数的乘法仅用  $cn \log n \times \log \log n$  个操作就可以完成，其中  $c$  是独立于  $n$  的绝对常数。随着  $n$  的增大，该算法执行的基本操作数将远小于  $n^2$ 。

对于线性方程组的求解，经典的高斯消元法用  $O(n^3)$  个基本的算术操作就可以求解  $n$  个变量上  $n$  个方程构成的方程组（虽然这一算法用高斯的名字命名，但早在公元 1 世纪中国数学家就已经掌握了这种算法的某种形式）。20 世纪 60 年代末，斯特拉森 (Strassen) 找到了更加高效的算法，该算法大约只需要执行  $O(n^{2.81})$  个操作就能求解这个问题。目前，求解这一问题的最佳算法需要执行  $O(n^{2.376})$  个操作。

宴会问题也有一段有趣的历史。同乘法的情况一样，宴会问题也存在显而易见的简单低效算法——从大到小地依次枚举  $n$  个人的每个子集，直到找到一个子集使得其中不含任何两个无法和睦相处的人。这个算法需要执行的计算步骤数可能与  $n$  个人的所有子集数一样多，亦即  $2^n$ 。这使得该算法根本无法用于实践，因为如果某人用这个算法来安排一个 70 人参加的宴会，即使她用超级计算机来进行处理，也需要提前一千年开始筹备。但出乎意料的是，人们至今仍没有为宴会问题找到效率显著更优的算法。事实上，我们有理由怀疑宴会问题不存在高效的算法，因为我们可以证明它等价于独立集这个计算问题，而独立集问题以及其他成千上万个计算问题都是 NPC 完全问题。著名的  $P \stackrel{?}{=} NP$  问题是问：有没有哪个 NPC 完全问题存在高效的算法？

### 证明高效算法的不存在性

我们已经看到，某些计算任务存在非平凡的算法使得其效率比几千年来人们一直使用的算法更高。一件特别有意义的事情是：证明某些组合任务的当前算法是最佳的。也就是说，这些计算任务不存在更有效的算法。例如，我们可以证明整数乘法的  $O(n \log n \log \log n)$  步算法无法进一步改进，这就说明乘法在本质上确实比加法更难，因为加法存在  $O(n)$  步算法。再比如，我们还可以证明，没有算法能用少于  $2^{n/10}$  个计算步骤来求解宴会问题。证明这样的结论是计算复杂性的一个核心任务。

如何才能证明这种不存在性呢？求解计算任务的算法可能有无穷个！因此，我们只能用数学手段证明其中的每个算法都比已知的算法更低效。这种方法之所以可行，是因为计算本身也是一个精确的数学概念。事实上，一旦这样一个结果被证明，则它必然吻合于数学上的某个不可能性结果，例如，几何中其他公理无法推导出欧几里得平行公理、尺规作图无法三等分一个角等。这些结论都是数学上最有价值、最可靠和最出人意料的结果。

在复杂性理论中，我们很少能证明这种不存在性结果。但是，在能力弱于一般计算机的计算模型上，我们确实已经证得了一些重要的不存在性结果。由于在一般的计算机上我们还缺少这样的好结果，因此复杂性理论在一般计算机上获得的重要结果指的是在不同复杂性之间建立的相互联系。人们在这方面获得了很多漂亮的结果。

### 关于计算效率的几个有趣问题

现在，我们概述关于计算复杂性的几个重要问题。

1. 生命科学、社会科学和运筹学等学科中的很多计算任务都通过搜索海量的解空间来找出问题的解。比如，前面已经提到的线性方程组的求解和宴会问题中找出最大的受邀者集合都属于这种情况。这种搜索通常称为穷举搜索，因为搜索过程穷举了所有的可能。这种穷举搜索能替换为更有效的算法吗？

2. 用随机性（即硬币投掷）能加快算法的计算速度吗？

3. 如果允许算法在小部分输入上出错，或者只要求算法求得问题的近似解，那么难解的问题会变得容易一些吗？

4. 计算上的难解问题对实践有什么帮助呢？例如，我们能借助这些难解问题构造出牢不可破的密码协议吗（至少相对于大家认可的敌手而言）？

5. 我们能利用物质的违背直觉的量子力学性质建造出更快的计算机吗？



6. 只有人才能证明数学定理吗? 换句话说, 数学证明能被自动生成吗? 能在不完整阅读数学证明的情况下验证数学证明的真伪吗? 证明者和验证者通过对话来完成的交互式证明比标准的“静态”数学证明更有效力吗?

证明是数学上的核心概念。事实证明, 它也是计算复杂性的核心概念。而且, 计算复杂性已经对数学证明的含义赋予了新的解释。数学证明能否自动生成将取决于  $P \stackrel{?}{=} NP$  问题的答案。概率可验证证明 (Probabilistic Checkable Proof) 是一种健壮的数学证明, 要查验这种证明的真伪, 只需概率地选取证明中的少数几个位置进行查验即可。相比之下, 传统的证明则需要逐行阅读才能查验其真伪。类似地, 用交互式证明的概念得出了一些出人意料的结果。还有研究证明复杂性的, 它是复杂性的一个子领域, 研究各种命题的最小证明长度。

历经近 40 年<sup>2</sup>的发展, 复杂性理论仍是一门年轻的科学, 许多重要结果的发现还不到 20 年。上述这些问题还没有被完全解决。一个令人意外的转折是, 复杂性理论被用于某些数学定理的证明中: 它们提供的证据表明计算复杂性中某些问题是难解的。

## 13.2 什么是计算?

所谓计算, 是指通过重复使用一种预先制定的规则来改变环境的过程。这里有一个关键性限制条件是规则必须是简单的: 在每次应用时, 只依赖于且只影响环境的一(小)部分。虽然每次使用规则时只产生极其有限的影响, 但多次使用之后, 结果将异常复杂。换言之, 计算会把与之相关的环境变得非常复杂, 但其过程只是重复简单的规则。

可以用计算这一概念对自然现象的“机械化”部分建立模型, 就是说, 确定现实(而非现象在某个特定时刻的特殊状态)演变的规则。此时, 研究的出发点是自然现象的实际演化过程, 而研究的目的是找到这个自然过程背后的规则。在某种意义上, 科学研究的目标总体上可以描述为寻找支配各种自然现象(或者这些现象的抽象)的规则。

计算机算法是由人类设计的计算规则, 借此可以对相应的人造环境产生特定的期望效果。

为了严格地定义计算, 需要指定计算模型, 即对于计算环境及可应用于该环境中的一类规则提供具体定义。这种模型对应着实际计算中计算机的抽象。常使用的一个简单抽象模型是图灵机 (Turing Machine), 因此算法通常由相应的图灵机来形式化定义。然后, 需要强调的是, 计算理论中的许多结论, 并不考虑特定的计算模型, 只要模型是“合理”的即可。[11]<sup>3</sup>

计算其实是一个过程, 算法是一个实例化的计算过程, 一个算法或者一个计算过程最后其实是确定了一个函数。要严格定义计算, 需要设计一个计算模型, 图灵在 1936 年其发表的论文中, 第一次创造性地提出了一个抽象的一般化的计算模型, 为计算研究或者说自动计算打开了一扇大门。

<sup>2</sup>本书出版在 2016 年左右

<sup>3</sup>此书的网站上有电子版可以下载, 网址<https://www.wisdom.weizmann.ac.il/~oded/cc-drafts.html>

## 13.3 计算的数学定义 (形式化描述)

自从图灵给出人类第一个计算模型图灵机后，陆续又提出几个不同的计算模型，这些计算模型在讨论模型问题时都有其优势，同时这些图灵机均被证明与图灵机计算能力等价，也就是一个模型能计算的，另外一个模型也能计算，一个模型不能计算，另外一个模型也不能计算。

### 13.3.1 确定型图灵机 (Deterministic Turing Machine)

#### 13.3.1.1 描述性定义

1936 年，英国数学家阿兰·麦席森·图灵 (1912-1954 年) 提出了一种抽象的计算模型——图灵机 (Turing machine)。图灵机，又称图灵计算机，即将人们使用纸笔进行数学运算的过程进行抽象，由一个虚拟的机器替代人类进行数学运算。

所谓的图灵机就是指一个抽象的机器，它有一条无限长的纸带，纸带分成了一个一个小方格，每个方格有不同的颜色。有一个机器头在纸带上移来移去。机器头有一组内部状态，还有一些固定的程序。在每个时刻，机器头都要从当前纸带上读入一个方格信息，然后结合自己的内部状态查找程序表，根据程序输出信息到纸带方格上，并转换自己的内部状态，然后进行移动。

图灵的基本思想是用机器来模拟人们用纸笔进行数学运算的过程，他把这样的过程看作下列两种简单的动作：

- 1、在纸上写上或擦除某个符号；
- 2、把注意力从纸的一个位置移动到另一个位置。

而在每个阶段，人要决定下一步的动作，依赖于 (1) 此人当前所关注的纸上某个位置的符号和 (2) 此人当前思维的状态。

为了模拟人的这种运算过程，图灵构造出一台假想的机器，该机器由以下几个部分组成：

1、一条无限长的纸带 **TAPE**。纸带被划分为一个接一个小格子，每个格子上包含一个来自有限字母表的符号，字母表中有一个特殊的符号表示空白。纸带上的格子从左到右依此被编号为 0, 1, 2, ...，纸带的右端可以无限伸展。

2、一个读写头 **HEAD**。该读写头可以在纸带上左右移动，它能读出当前所指的格子上的符号，并能改变当前格子上的符号。

3、一套控制规则 **TABLE**。它根据当前机器所处的状态以及当前读写头所指的格子上的符号来确定读写头下一步的动作，并改变状态寄存器的值，令机器进入一个新的状态。

4、一个状态寄存器。它用来保存图灵机当前所处的状态。图灵机的所有可能状态的数目是有限的，并且有一个特殊的状态，称为停机状态。参见停机问题。

注意这个机器的每一部分都是有限的，但它有一个潜在的无限长的纸带，因此这种机器只是一个理想的设备。图灵认为这样的一台机器就能模拟人类所能进行的任何计算



过程。

### 13.3.1.2 形式化定义

我们这里描述的是一个最简单的图灵机，确定型图灵机 (Deterministic Turing Machine)，通常缩写为 DTM，下面我们对图灵机进行形式化描述，即使是图灵机的形式化定义，不同资料上给出的也略有不同，但是都是等价定义。

一台图灵机是一个七元组， $\{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$ ，其中  $\Sigma, \Gamma, \delta$  都是有限集合，且满足：

- 1、 $Q$  是状态集合；
- 2、 $\Sigma$  是输入字母表或者是输入符号的有穷集合，其中不包含特殊的空白符  $B$ ；
- 3、 $\Gamma$  是带字母表，也有称为带上可用字符集合，其中  $\Sigma \subset \delta$ ；
- 4、 $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$  是转移函数，其中  $L, R$  表示读写头是向左移还是向右移；
- 5、 $q_0 \in Q$  起始状态， $q_0 \in Q$ ；
- 6、 $B$  空白符号<sup>4</sup>， $B \in \Gamma, B \notin \Sigma$ ，开始时  $B$  出现在除包含输入符号的有穷多个初始单元之外的所有单元中。
- 7、 $F$  是终结状态或拒绝状态  $q_{reject}$  和接受状态  $q_{accept}$  的集合，且  $q_{reject} \neq q_{accept}$ ， $F \subset Q$ 。

图灵机  $\{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$  将以如下方式运作：

开始的时候将输入符号串从左到右依此填在纸带上，其他格子保持空白（即填以空白符）。 $M$  的读写头指向第 0 号格子， $M$  处于状态  $q_0$ 。机器开始运行后，按照转移函数  $\delta$  所描述的规则进行计算。例如，若当前机器的状态为  $q$ ，读写头所指的格子中的符号为  $x$ ，设  $\delta(q, x) = (q', x', L)$ ，则机器进入新状态  $q'$ ，将读写头所指的格子中的符号改为  $x'$ ，然后将读写头向左 ( $L$ ) 移动一个格子。若在某一时刻，读写头所指的是第 0 号格子，但根据转移函数它下一步将继续向左移，这时它停在原地不动。换句话说，读写头始终不移出纸带的左边界。若在某个时刻  $M$  根据转移函数进入了状态  $q_{accept}$ ，则它立刻停机，并接受输入的字符串；若在某个时刻  $M$  根据转移函数进入了状态  $q_{reject}$ ，则它立刻停机并拒绝输入的字符串。

注意，转移函数  $\delta$  是一个部分函数，换句话说对于某些  $q, x, \delta(q, x)$  可能没有定义，如果在运行中遇到下一个操作没有定义的情况，机器将立刻停机。

为了形式化地描述图灵机的运转过程，引入格局 (configuration) 或瞬时描述 (instantaneous description)，格局对当前图灵机的状态进行了完整描述，格局是一个三元组  $(q, w, u)$ ， $q$  是当前状态， $w$  是读写头左边字符串， $u$  是读写头右边字符串，读写头正扫描  $u$  的最左面的符号，我们也通常记为  $wqu$ ，而用  $\vdash$  表示格局变化。

**例 13.1** [12] 我们来描述一个接受语言  $L = \{a^i b a^j | 0 \leq i \leq j\}$  的图灵机  $M$ ， $M$  有状态集合

<sup>4</sup>有的参考资料上写的是空格符号，但是这个叫法有时候会和我们键盘上的空格在理解上混淆，但其实两个含义完全不一样，因为键盘上空格依然是一个符号，但是图灵机里这个空格符号没有任何实际意义，所以此处我们称为空白符号。

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$ , 初始状态  $q_0$ , 接受状态集合为  $\{q_5\}$ , 输入符号集合  $\Sigma = \{a, b\}$ , 带用符号集合  $\Gamma = \{a, c, c, B\}$ ,  $B$  为空格符号, 图灵机的转移函数  $\delta$  如下表定义:

在描述状态转移函数的状态转移表中, 第一个参数表示转移的状态, 第二个参数表示在

$\delta$	a	b	c	B
$q_0$	$q_1, c, R$	$q_4, B, R$	$q_0, B, R$	
$q_1$	$q_1, a, R$	$q_2, b, R$		
$q_2$	$q_3, c, L$		$q_2, c, R$	$q_2, B, R$
$q_3$	$q_3, a, L$	$q_3, b, L$	$q_3, c, L$	$q_0, B, R$
$q_4$	$q_4, a, R$		$q_4, B, R$	$q_5, B, R$

当前位置写入的符号, 第三个表示移动方向, 比如第二行第二列的  $q_1, c, R$ , 表示当前状态为  $q_0$ , 读头下是  $a$  (也就是说读取  $a$ ) 是, 转移到状态  $q_1$ , 并且写入  $c$  (也就是把原来的  $a$  改写为  $c$ ), 然后向右移动。

容易看出, 对所有  $L$  中符号行,  $M$  都能停在状态  $q_5$  中, 对于符号行  $a^i b a^j$  ( $i \geq j \geq 0$ ),  $M$  无法停机, 对于不含符号  $b$  或者含两个以上  $b$  的符号行  $w$ ,  $M$  也能停机, 但是停在非接受状态中, 下面是三种运算过程的例子:

(1) 在输入  $abaa$  时 (停机, 接受):  $q_0 abaa \vdash cq_1 baa \vdash cbq_2 aa \vdash cq_3 bca \vdash q_3 cbca \vdash q_3 Bcbca \vdash q_0 cbca \vdash q_0 bca \vdash q_4 ca \vdash q_4 a \vdash aq_4 B \vdash aBq_5 B$ 。

(2) 在输入  $aaba$  时 (不停机):  $q_0 aaba \vdash cq_1 aba \vdash caq_1 ba \vdash cabq_2 a \vdash caq_3 bc \vdash cq_3 abc \vdash q_3 cabcb \vdash q_3 Bcabcb \vdash q_0 cabcb \vdash cq_1 bcb \vdash cbq_2 c \vdash cbcq_2 B \vdash cbcBq_2 B \vdash cbcBBq_2 B \vdash \dots$ 。

(3) 在输入  $abab$  时 (停机, 但不可接受):  $q_0 abab \vdash cq_1 bab \vdash cbq_2 ab \vdash cq_3 bcb \vdash q_3 cbcb \vdash q_3 Bcbab \vdash q_0 cbcb \vdash q_0 bcb \vdash q_4 cb \vdash q_4 b$ 。

**定义 13.1** [12] (a) 如果对  $\Sigma$  上的语言  $A$ , 存在图灵机  $M$ ,  $A$  是  $M$  可接受语言, 那么称  $A$  为图灵机可接受集, 或递归可枚举集, 简称为  $r.e.$  集。

(b) 如果对  $\Sigma$  上的语言  $A$  和它的补集  $\bar{A} = \Sigma^* - A$  都是图灵可接受集, 那么称其为图灵可判定集, 或者说它是递归集。

(c) 如果对部分函数  $f: \Sigma^* \rightarrow \Sigma^*$ , 如果存在图灵机  $M$  使得 (i) 将  $f$  的定义域中  $w$  输入  $M$  时所得输出为  $f(w)$ , (ii) 将  $f$  的定义域外之  $w$  输入时,  $M$  无法停机, 那么称  $f$  为可计算函数, 或者说部分递归函数。

(d) 如果一个部分递归函数  $f$  的定义域等于  $\Sigma^*$ , 则称  $f$  为全递归函数, 或递归函数。

在大多数研究者阐明图灵机的时候都会结合语言来讲计算模型, 但是也有部分学者不建议在介绍图灵机概念时引入语言, 其认为这对于理解图灵机的实质并无益处。

**例 13.2** 设计一个二进制的非操作图灵机。

**解** 输入字符集  $\Sigma = \{0, 1\}$ , 带字母表  $\Gamma = \{0, 1, N\}$ , 其中  $N$  是空白符 (null), 这是一个

3-symbol 图灵机，此图灵机执行过程为：从一个小方格子里读出 symbol；清除一个小方格子里的内容，或者直接写入新的 symbol 覆盖原有的数据；向左（右）移动。总共有三种状态  $\{state_0, state_1, stop\}$  状态转移函数为：

$\delta$	0	1	N
$state_0$	$state_1, 1, R$	$state_1, 0, R$	$state_1, N, R$
$state_1$	$state_1, 1, R$	$state_1, 0, R$	$stop, N, R$
$stop$			

如果输入 010，其计算过程为（为了表达方便，用  $s_i$  表示状态  $state_i$ ）：

$s_0 010 \vdash 1s_1 10 \vdash 10s_1 0 \vdash 101s_1 \vdash stop$

注意，因为传输带是无限长，所以  $101s_1 = 101s_1 N$ 。

**例 13.3** 设计一个二进制加法图灵机。

**解** 在这里我们把一个二进制数直接表示为 0 的个数，比如 101(十进制的 5)，我们表示为 "00000"，加法用 "c" 来表示，比如 00c000，就是表示 2+3，对于这个计算带的输入就是 "00c000"，我们可以看到对于这样的编码，加法图灵机的状态转移函数变的很简单  $(q_0, 0) \rightarrow (q_1, 0, R), (q_0, c) \rightarrow (q_{reject}, c, R), (q_1, 0) \rightarrow (q_1, 0, R), (q_1, c) \rightarrow (q_1, 0, R), (q_1, B) \rightarrow (q_{accept}, B, R)$ 。

用自然语言来描述就是把 c 换成 0 即可。

### 13.3.2 随机存取机 (Random-Access Machine, RAM)

随机存取机 (Random-Access Machine, RAM) 是冯·诺依曼在其 1945 年的一个技术报告<sup>5</sup>中提出的一个计算模型，下面我们简单描述以下这个模型 [11]。一个抽象的 RAM 包含任意多个存储单元，数量有限的寄存器，其中一个程序计数器，以及由有限个指令构成的程序，可能的指令集合包含以下指令：

- $reset(r)$ :  $r$  为寄存器编号，此指令将寄存器  $r$  的值置 0。
- $inc(r)$ :  $r$  为寄存器编号，此指令为增加  $r$  的值。类似  $dec(r)$ , 使寄存器  $r$  的值减小。
- $load(r_1, r_2)$ :  $r_1, r_2$  为寄存器编号，此指令将存储位置  $m$  处的值存入寄存器  $r_1$  中，而  $m$  为寄存器  $r_2$  中存储的内容。
- $store(r_1, r_2)$ : 将寄存器  $r_1$  中的值存储到内容中，此内容位置记录在  $r_2$  中。
- $cond - goto(r, l)$ :  $r$  为寄存器序号， $l$  为不大于程序长度的整数，此指令表示，如果寄存器  $r$  中保存的值非负，则程序计数器的值置为  $l-1$ 。

每条指令执行后，程序计数器的值加 1，程序计数器的指针移到下一条要执行的指令（当程序计数器的值超过程序长度时停机），可以将前  $n$  个存储单元的内容定义为机器的输入，其中  $n$  的值保存于一个特殊的寄存器中。

<sup>5</sup>First Draft of a Report on the EDVAC

可以证明这个抽象的 RAM 可以由一个图灵机来模仿,也就是说 RAM 的计算能力不超过图灵机。反之如果我们证明用 RAM 来模拟图灵机,那么就是说明两种模型计算能力等价。

### 13.3.3 布尔电路 (Boolean Circuits)

布尔电路计算模型比图灵机模型更加简单,在证明复杂性下界时比图灵机证明更加容易一些。

**定义 13.2 (布尔线路)** [10] 对任意  $n \in N$ , 一个  $n$  输入单输出的布尔线路是具有  $n$  个源顶点和 1 个汇顶点的有向图。源顶点也称输入顶点,指的是入度为 0 的顶点。汇顶点也称输出顶点,指的是出度为 0 的顶点,每个非源顶点称为一个逻辑门,并用逻辑操作  $\wedge$ (与)、 $\vee$ (或)、 $\neg$ (非) 中的一个操作进行标记。顶点的扇入度指的是进入该顶点的边的条数。标价为和的顶点的扇入度等于 2,而标记为  $\neg$  的顶点的扇入度等于 1。如果  $C$  是一个布尔线路,而  $x \in \{0,1\}^n$  是他的一个输入,则将  $C$  在  $x$  上输出记为  $C(x)$ ,定义每个顶点  $v$  的输出值为  $val(v)$ ,如果  $v$  是第  $i$  个源顶点,则  $val(v) = x_i$ 。 $C(x)$  是  $C$  的输出顶点的值。

在有些书籍里布尔电路定义是  $n$  输入  $m$  输出,这个没有大的影响,因为我们可以证明,可以用  $n$  输入 1 输出的布尔电路来模拟  $n$  输入  $m$  输出。

研究布尔线路的动机之一是为了给硅芯片设计一个数学模型,但是后来作为研究计算复杂性的计算模型了。

布尔电路的规模是指其边的个数,对于计算函数  $f : (0,1)^* \rightarrow \{0,1\}$  的电路簇  $(C_n)_{n \in N}$ ,可以将  $C_n$  的规模看做是关于  $n$  的函数,  $s(n), s : N \rightarrow N$ , 函数  $f$  的电路复杂度记为  $s_f$ ,是所有可计算  $f$  的电路簇的规模复杂度下确界。

电路复杂度具有如下一些定理 [11]:

1. 任意一个布尔函数都可由某类电路计算,并且电路复杂度最多为指数级。也就是说函数  $f : \{0,1\}^* \rightarrow \{0,1\}$  可由规模为  $O(n2^n)$  的电路计算,这个电路实现的是查表运算。
2. 一些函数具有多项式的电路复杂度,特别是任意一个时间复杂度为  $t$  的函数(即可由时间复杂度为  $t$  的算法计算的函数)具有  $\text{poly}(t)$  的电路复杂度,也就是说具有多项式的电路复杂度。(证明思路:考虑计算该函数的图灵机,及其对于  $n$  比特输入的计算,相关计算过程可以用一个  $t(n)$  层电路模仿,电路的每一层代表机器的一种中间格局。)
3. 几乎所有的布尔函数都有指数级的电路复杂度,特别地,可由规模为  $s$  的电路计算的  $\{0,1\}^n \rightarrow \{0,1\}$  的映射个数小于  $s^{2s}$ 。

多项式规模的电路簇被称为一致的,如果一个函数可以由一致的多项式规模电路簇计算,则他也可以由多项式时间算法计算。



### 13.3.4 判定树 (Decision tree)

目前图灵计算能力相关的一些基本问题仍然远未解决，于是，为了用其他方式深刻地理解“高效计算”这一难以琢磨的概念，人们转而研究更简单的限制性更强的计算模型，在这些模型中最简单的就是判定树。[10]

**定义 13.3 (判定树)** [10] 设  $f: \{0,1\}^n \rightarrow \{0,1\}$  是一个函数，构造计算  $f$  的判定树如下，每个内部节点标记为形如  $x_i$ ，并且其恰有两条分别被标记为 0 和 1 的出边，每个叶节点被标记为输出 0 或 1。在输入  $x = x_1x_2 \dots x_n$  时，计算过程为，更加节点自身的标记来检查输入中相应的二进制位  $x_i$ ，如果  $x_i = 1$ ，则计算从该顶点出发，沿 1 边移动到下一节点，如果  $x_i = 0$ ，则计算从该顶点出发，沿 0 边移动到下一节点，一直到叶节点，输出结果。

**例 13.4** 构造一个判定树，计算 3 位的多数函数 (majority function)。多数函数是这样的函数，如果 3 个二进制中至少有两个 1，结果为 1，否则结果为 0。

**解** 3 位多数函数的判定树如图 13.1 所示。

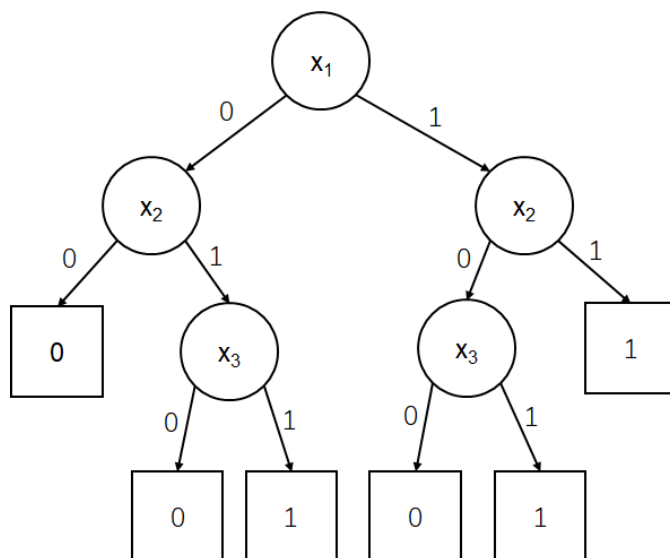


图 13.1: 3 位多数函数的判定树

### 13.3.5 $\lambda$ 演算 ( $\lambda$ Calculus)

$\lambda$  演算 (Lambda Calculus 或  $\lambda$  Calculus) 是阿隆佐·邱奇 (Alonzo Church) 在 20 世纪 30 年代其指导学生陆续提出的概念<sup>6</sup>，在 1941 年对这部分工作进行总结出版了一书 “The Calculi of Lambda-Conversion”。

邱奇和图灵是生活在一个年代的人，其提出的  $\lambda$  演算与图灵机一样，都是想探寻计算的本质，讨论机械/自动计算。关于演算的基本介绍可以参考英文介绍 “The Lambda

<sup>6</sup>关于 Church 的详细生平可以看 Manzano, Maía 于 1997 发表的论文, Alonzo church: his life, his work and some of his miracles. History and Philosophy of Logic, 18(4), 211-232. doi:10.1080/01445349708837290

Calculus for Absolute Dummies (like myself)”<sup>7</sup>,Brilliant 网站关于 Lambda Calculus 的介绍<sup>8</sup>, 以及一个外文介绍的中文翻译 “Good Math/Bad Math 的 Lambda 演算系列的中文翻译”<sup>9</sup>。

$\lambda$  演算表达式由 3 个元素组成, 变量, 函数和应用, 演算过程就是将表达式进行简单的查找替换, 而这样的模型计算能力与图灵机等价。

我们可以将  $\lambda$  表达式用以下 BNF 范式 (Backus-Naur form) 表达的上下文无关文法描述:

1.  $\langle \text{expression} \rangle := \langle \text{name} \rangle \mid \langle \text{function} \rangle \mid \langle \text{application} \rangle$
2.  $\langle \text{function} \rangle := \lambda \langle \text{name} \rangle . \langle \text{expression} \rangle$
3.  $\langle \text{application} \rangle := \langle \text{expression} \rangle \langle \text{expression} \rangle$

其中 name 也称为变量 (variable), 可以是任何字母 a,b,c 等等。

**例 13.5** 函数  $f(x) = x$  的  $\lambda$  表达式为:  $\lambda x . x$

**例 13.6** 当对表达式  $(\lambda x . x)a$  求值时, 就是把函数体中所有出现的  $x$  都替换为  $a$ , 如  $(\lambda x . x)a$  求值结果为  $a$ 。虽然 lambda 演算传统上只支持单参数函数, 但我们可以用一种被称为柯里化 (currying) 的技巧来创造多参数函数, 如  $(\lambda x . \lambda y . \lambda z . xyz)$  表达的函数是  $f(x, y, z) = ((xy)z)$ 。

必须意识到经典的  $\lambda$  演算没有数, 字符, 或任何非函数的数据类型。

**例 13.7** 在  $\lambda$  演算中没有 True 或 False, 也没有 1 或 0。布尔逻辑的表达方式为: T 表示为  $\lambda x . \lambda y . x$ , F 表示为  $\lambda x . \lambda y . y$ 。然后我们定义一个 IF 函数  $\lambda b t f$ , 如果  $b$  为 True 返回  $t$ , 如果  $b$  为 False 返回  $f$ , IF 等价于  $\lambda b . \lambda t . \lambda f . b t f$ , 使用 IF 函数, 我们可以定义基本的布尔逻辑运算符:  $a \text{ AND } b$  等价于  $\lambda a b . \text{IF} a b F$ ;  $a \text{ OR } b$  等价于  $\lambda a b . \text{IF} a T b$ ;  $\text{NOT } a$  等价于  $\lambda a . \text{IF} a F T$ 。此处  $\text{IF} a b c$  也可写为  $\text{IF}((ab)c)$ 。

**例 13.8** 虽然 Lambda 演算没有数字, 但我们可以用 Church 数来对数字编码。对于任意数  $n$ :  $n = \lambda f . f^n$ , 所以有  $0 = \lambda f . \lambda x . x$ ,  $1 = \lambda f . \lambda x . f x$ ,  $2 = \lambda f . \lambda x . f(fx)$ ,  $3 = \lambda f . \lambda x . f(f(fx))$ , 我们定义一个后继函数 (successor function)  $S(n) = n + 1$ , 即  $S = \lambda n . \lambda f . \lambda x . f((nf)x)$ , 使用后继函数我们可以定义加法  $\text{ADD} = \lambda a b . (aS)b$

### 13.3.6 细胞自动机 (Cellular automata)

细胞自动机 (cellular automata) 是为模拟包括自组织结构在内的复杂现象提供的一个强有力的方法, 也称为元胞自动机 (Cellular Automaton)。细胞自动机模型的基本思想是: 自然界里许多复杂结构和过程, 归根到底只是由大量基本组成单元的简单相互作用所引起。细胞自动机主要研究由小的计算机或部件, 按邻域连接方式连接成较大的、并行工作的计算机或部件的理论模型。它分为固定值型、周期型、混沌型以及复杂型。<sup>10</sup>

<sup>7</sup>The Lambda Calculus for Absolute Dummies (like myself), 网络链接: <http://bach.ai/lambda-calculus-for-absolute-dummies/>

<sup>8</sup>Lambda Calculus, <https://brilliant.org/wiki/lambda-calculus/>

<sup>9</sup>Good Math/Bad Math 的 Lambda 演算系列的中文翻译, 网络链接: <http://cgnail.github.io/academic/lambd-index/>

<sup>10</sup><https://baike.baidu.com/item//2765689>

细胞自动机是 20 世纪 50 年代初由计算机之父冯·诺依曼 (J.von Neumann) 为了模拟生命系统所具有的自复制功能而提出来的。此后, 史蒂芬·沃尔夫勒姆 (Stephen Wolfram) 对元胞自动机理论进行了深入的研究。例如, 他对一维初等元胞机全部 256 种规则所产生的模型进行了深入研究, 并将元胞自动机分为平稳型、周期型、混沌型和复杂型 4 种类型。有关细胞自动机的系统介绍可以查看斯坦福大学斯坦福哲学百科全书网站上的 “Cellular Automata” 介绍<sup>11</sup>。

### 13.3.6.1 能够识别轮廓的细胞自动机

下面我么举一个例子"能够识别轮廓的细胞自动机"<sup>12</sup>, 看看细胞自动机的有趣之处, 其可以用看似简单的规则, 完成一些负责功能。

我们用一个矩形网状细胞群作为执行某些有用操作的细胞自动机示例, 其中每个细胞都为黑色或白色并处于初始设置状态。每个细胞在下一个步骤都将遵守如下规则: 如果一个细胞为黑色, 与之相邻的细胞为白色, 那么此细胞将保持黑色, 反之, 细胞将变为白色。这里, 相邻细胞指的是水平或垂直相邻的任何细胞。

图13.2是细胞自动机的初始状态, 我们按照上面描述的规则运行, 稳定后, 也就是计算结束后, 我们发现这个细胞自动机运行的结果是实现了边缘检测, 如图??所示, 换句话说, 此规则成功地筛选出了形状的轮廓。请注意, 任何细胞都只使用了本地信息, 但却成功地提取了我们可能认为是全局的东西, 即轮廓。应用本地规则可以产生全局模式。。

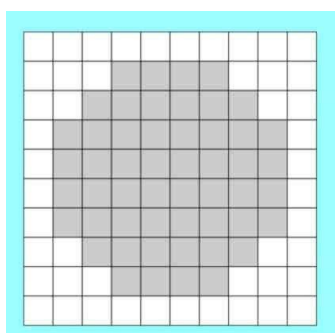


图 13.2: 细胞自动机初始状态

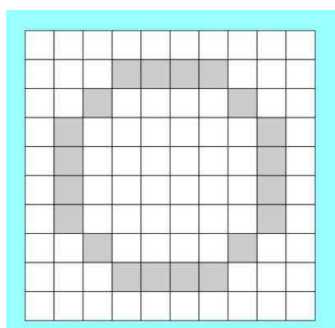


图 13.3: 细胞自动机停机状态

<sup>11</sup> “Cellular Automata” ,<https://plato.stanford.edu/entries/cellular-automata/>

<sup>12</sup>能够识别轮廓的细胞自动机, 链接<https://baijiahao.baidu.com/s?id=1629331712004653022&wfr=spider&for=pc>

### 13.3.7 随机计算 (Randomized Computation)

随机计算在实践中是客观存在的, 这些随机方法被成功应用到许多问题的解决中, 并且得到了更简单或更高效的算法, 前面提出的确定型计算模型都可以扩展到随机计算中。

#### 13.3.7.1 概率型图灵机 (Probabilistic Turing machines)

随机算法能以某种方式进行随机选择, 比如随机地将变量初始化为某个范围的一个整数。在实践中, 随机算法通过随机数发生器来实现。同用标准图灵机为确定型算法建模一样, 我们用概率型图灵机为随机算法建模。

**定义 13.4 (概率型图灵机 (Probabilistic Turing machines, 简称 PTM))** [10] PTM 是有两个转移函数  $\delta_0, \delta_1$  的图灵机, 概率型图灵机  $M$  在输入  $x$  上运行时, 每个步骤以  $1/2$  的概率选用转移函数  $\delta_0$ , 以  $1/2$  的概率选用转移函数  $\delta_1$ 。每一步的选择均独立于之前所做的所有选择。PTM 只能输出 1(接受) 或 0(拒绝), PTM 在输入  $x$  上运行结束时的输出结果是一个随机变量, 记为  $M(x)$ , 对于函数  $T: N \rightarrow N$ , 如果概率型图灵机  $M$  在任意输入  $x$  上运行时, 无论  $M$  做任何随机选择, 他都在  $T(|x|)$  步内停机, 则称  $M$  的运行时间为  $T(n)$ 。

**例 13.9** 寻找数集  $\{a_1, \dots, a_n\}$  的中位数。

**解** 这里给一个找  $k$  位数的一般算法  $FINDKTHELEMENT(k, a_1, \dots, a_n)$ , 输出数集中第  $k$  小的数, 算法如下:

1. 随机选择  $i \in [n]$ , 并令  $x = a_i$ 。
2. 扫描列表  $\{a_1, \dots, a_n\}$ , 统计满足  $a_i \leq x$  的  $a_i$  个数  $m$ 。
3. 如果  $m = k$ , 则输出  $x$ 。
4. 否则, 如果  $m > k$ , 则将满足  $a_i \leq x$  的所有  $a_i$  拷贝到新列表  $L$  中, 执行  $FINDKTHELEMENT(k, L)$ 。
5. 否则, 如果  $m < k$ , 则将满足  $a_i > x$  的所有  $a_i$  拷贝到新列表  $H$  中, 执行  $FINDKTHELEMENT(k - m, H)$ 。

此算法运行时间  $T(n) = O(n)$ 。我们通常用的确定型算法, 就是先排序, 然后在找中位数, 其时间复杂度为  $O(n \log n)$ 。

## 13.4 计算复杂性

### 13.4.1 计算复杂类

计算复杂性是描述计算所需资源的情况, 目前我们看到的基本资源有时间和空间, 但不仅仅是这两种资源, 还有很多其他资源比如 Blum 复杂度<sup>13</sup>, 计算复杂性类就是在给定资源界限下能被计算的所有函数构成的集合。

**定义 13.5 (DTIME 类 (Deterministic time class))** [10] 设  $T: N \rightarrow N$  是一个函数, 称语言  $L \in DTIME(T(n))$ , 当且仅当存在运行时间为  $c \cdot T(n)$  的确定型 (Deterministic) 图灵机可以判定语言  $L$ , 其中  $c > 0$  是常数。

<sup>13</sup>M. Blum. "A Machine-independent theory of the complexity of recursive functions", J. ACM 14, 2, pp.322-336, 1967.

DTIME 中的 D(Deterministic) 是指确定型图灵机的意思。

**定义 13.6 (P 类 (polynomial class))** [10]  $P = \bigcup_{c \geq 1} DTIME(n^c)$ , 其中  $c$  为常数。

**定义 13.7 (NP 类 (nondeterministic polynomial class))** [12] 语言  $A$  属于 NP, 当且仅当存在语言  $B \in P$  和多项式  $p$ , 使得  $x \in A \Leftrightarrow (\exists y, |y| \leq p(|x|)) \langle x, y \rangle \in B$ 。

非确定型计算是一个猜测加验证的计算。

复杂性类 P(polynomial)、NP(nondeterministic polynomial) 类和 NPC(NP complete) 之间的关系如图 13.4 所示。

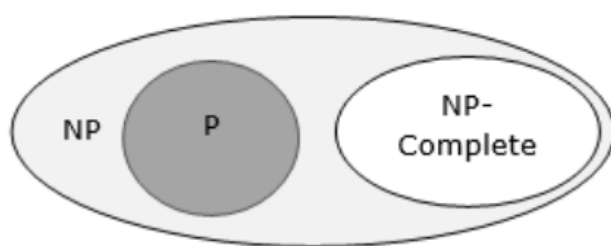


图 13.4: P 类, NP 类和 NPC 类之间的关系

除上面讲的时间复杂性, 还可以研究空间复杂性, 可以对图灵机执行计算过程中使用的存储单元个数进行限制, 由此定义空间受限计算的概念, 定义  $SPACE(n)$ , 依次可定义 PSAPCE 和 NPSpace。具体概念可以参考 Arora 的“计算复杂性现代方法” [10]。

关于复杂性分类, 下面我们引用 Oded Goldreich 在其“计算复杂性” [11] 中“1.2.5 Complexity Classes”的描述。

Complexity classes are sets of computational problems. Typically, such classes are defined by fixing three parameters:

1. A type of computational problems (see Section 1.2.2). Indeed, most classes refer to decision problems, but classes of search problems, promise problems, and other types of problems will also be considered.

2. A model of computation, which may be either uniform (see Section 1.2.3) or non-uniform (see Section 1.2.4).

3. A complexity measure and a limiting function (or a set of functions), which put together limit the class of computations of the previous item; that is, we refer to the class of computations that have complexity not exceeding the specified function (or set of functions). For example, in §1.2.3.5, we mentioned time complexity and space complexity, which apply to any uniform model of computation. We also mentioned polynomial-time computations, which are computations in which the time complexity (as a function) does not exceed some polynomial (i.e., a member of the set of polynomial functions).

The most common complexity classes refer to decision problems, and are sometimes defined as classes of sets rather than classes of the corresponding decision problems. That is, one often says that a set  $S \subseteq \{0, 1\}^*$  is in the class  $C$  rather than saying that the problem of deciding

membership in  $S$  is in the class  $C$ . Likewise, one talks of classes of relations rather than classes of the corresponding search problems (i.e., saying that  $R \subseteq \{0, 1\} \times \{0, 1\}$  is in the class  $C$  means that the search problem of  $R$  is in the class  $C$ ).

### 13.4.2 邱奇-图灵命题 (Church-Turing Thesis)

我们前面介绍了不同的计算模型，那么就存在这么个问题？不同计算模型的计算能力是否相同？或者换句话说来说，不同的某一个问题的属于那个计算复杂类，是否与你选择的计算模型有关？

邱奇-图灵命题就是来说明这个问题的。这个命题其实无法被证明，所以很多时候也称为邱奇-图灵断言。

“There are various equivalent formulations of the Church-Turing thesis. A common one is that every effective computation can be carried out by a Turing machine.”<sup>14</sup>

**定义 13.8 (Church-Turing 命题)** [12] 如果一个函数在某个合理的计算模型上可计算，那么它在图灵机上也是可计算的。

**定义 13.9 (广义 Church-Turing 命题)** [12] 如果一个函数在某个合理的计算模型上使用合理时间复杂性度量是多项式时间可计算的，那么它在图灵机上也是多项式时间可计算的。

其实在这个描述是不严格的，比如对“合理的计算模型”就没有一个准确定义。

邱奇-图灵命题断言任何可被物理实现的计算装置均可被图灵机模拟，也就是说任何其他模型上的可计算问题的集合不会比图灵机上可计算问题的集合更大，所以，我可以理解这个问题属于什么复杂类和你选择什么样的模型研究无关，他虽然不是一个可以被证明的命题，他更多是对计算或者世界本质的哲学思考<sup>15</sup>。

## 13.5 Uniform and Non-uniform

在我们看一些介绍资料时通过会看到 uniform model of computation 和 non-uniform model of computation 这两个概念。有些将 uniform model 翻译为“一致性模型”，将 non-uniform model 翻译为“非一致性模型”，uniform model 是指所有的计算都可以用这个模型来表示，而 non-uniform model 则不行，所以将其翻译为“统一模型”个“非统一模型”也许更利于理解。

我们看看 Oded Goldreich 在其“计算复杂性”[11]一书中的一段描述。

By a non-uniform model of computation we mean a model in which for each possible input length a different computing device is considered, while there is no “uniformity” requirement

<sup>14</sup>copy from <https://plato.stanford.edu/entries/church-turing/>

<sup>15</sup>有兴趣的可以参考“The Church-Turing Thesis”，网址链接<https://plato.stanford.edu/entries/church-turing/>



relating devices that correspond to different input lengths. Furthermore, this collection of devices is infinite by nature, and (in the absence of a uniformity requirement) this collection may not even have a finite description. Nevertheless, each device in the collection has a finite description. In fact, the relationship between the size of the device (resp., the length of its description) and the length of the input that it handles will be of major concern.

## 13.6 Hyper computation

Hyper computation 通常我们翻译为“超计算”，是研究比图灵机有着更强计算能力的计算模型，如，图灵自己提出的“预言机”(也有翻译为“喻示机”、“神喻图灵机”)(oracle machine)等。有很多超计算模型并没有对应的物理实现。想进一步了解相关研究的可以阅读文献"The many forms of hypercomputation" <sup>16</sup>

"Hypercomputation: computing more than the Turing machine" <sup>17</sup>

---

<sup>16</sup>Toby Ord, The many forms of hypercomputation, Applied Mathematics and Computation, Volume 178, Issue 1, 2006, Pages 143-153, url <http://www.amirrorclear.net/files/the-many-forms-of-hypercomputation.pdf>

<sup>17</sup>url [http://www.hypercomputation.net/download/2002a\\_ord.pdf](http://www.hypercomputation.net/download/2002a_ord.pdf)



## 参考文献

---

- [1] 沈永欢. 实用数学手册 [M]. 北京: 科学出版社, 1992.
- [2] 闵嗣鹤, 严士健. 初等数论 (第二版)[M]. 北京: 高等教育出版社, 2003.
- [3] 贾春福. 信息安全数学基础 [M]. 北京: 机械工业出版社, 2017.
- [4] 任伟. 信息安全数学基础——算法、应用于实践 (第 2 版)[M]. 北京: 清华大学出版社, 2018.
- [5] 范崇金. 近世代数基础 [M]. 哈尔滨: 哈尔滨工程大学出版社, 2003.
- [6] DAN BONEH V S. A graduate course in applied cryptography[M]. USA: MIT.
- [7] 李浪, 邹伟, 郭迎. 密码工程学 [M]. 北京: 清华大学出版社, 2014.
- [8] 杨波. 现代密码学 (第 4 版)[M]. 北京: 清华大学出版社, 2017.
- [9] 高胜朱建明等编著. 区块链技术与实践 [M]. 北京: 机械工业出版社.
- [10] Sanjeev Arora(译) 骆吉洲. 计算复杂性: 现代方法 [M]. 北京: 机械工业出版社, 2016.
- [11] Oded Goldreich 韩益亮 杨晓元. 计算复杂性 [M/OL]. 北京: 国防工业出版社, 2015. <https://www.wisdom.weizmann.ac.il/~oded/cc-drafts.html>.
- [12] 堵丁柱王洁. 计算复杂性导论 [M]. 北京: 高等教育出版社, 2002.
- [13] 杨思熈. 数论与密码 [M]. 上海: 华东师范大学出版社, 2010.
- [14] STEIN W. Sage Quick Reference[EB/OL]. 2009. <http://wiki.sagemath.org/quickref>.
- [15] Christos H. Papadimitriou 彭超 卜天明. 计算复杂性 [M]. 北京: 机械工业出版社, 2004.
- [16] TURING A. On computable numbers, with an application to the entscheidungsproblem[J]. Alan Turing His Work and Impact, 1936, s2-42(1): 13-115.
- [17] John E. HopcroftJeffrey D. Ullman [译] 刘田姜晖 王捍贫. 自动机理论、语言和计算导论 [M]. 北京: 机械工业出版社, 中信出版社, 2004.



## 附录 A 习题参考解答

### Answer of exercise 1

先计算幂集，然后判断幂集和各个各部分的关系。

### Answer of exercise 2

$$\sim B = \{3, 4\}, \rho(A) = \{\phi, \{1, 4\}, \{1\}, \{4\}\}, \rho(C) = \{\phi, \{2, 4\}, \{2\}, \{4\}\}$$

- 1)  $A \cap \sim B = \{4\}$
- 2)  $(A \cup B) \cap (A \cup C) = \{1, 2, 4\}$
- 3)  $\sim (A \cup B) = \{3\}$
- 4)  $\rho(A) - \rho(C) = \{\{1, 4\}, \{1\}\}$

### Answer of exercise 3

先按照对称差的定义，用基本运算来表示，然后在组合为右边形式。

### Answer of exercise 4

这道题来源于 16 级的学生做 17 级的班助，17 级的做 18 级的，18 级做 19 级的这种关系，二元逆关系总是存在的，将有序偶倒着写就是二元逆关系。

### Answer of exercise 5

首先要知道什么是等价关系：自反，传递，对称，很容易验证整数集上的相等关系符合等价定义。

### Answer of exercise 6

证明不满足，其实只需要给出一个反例即可，设  $A = \{1\}, B = \{a, b\}, C = \{x, y\}$ ,  $(A \times B) \times C = \{ \langle \langle 1, a \rangle, x \rangle, \langle \langle 1, a \rangle, y \rangle, \langle \langle 1, b \rangle, x \rangle, \langle \langle 1, b \rangle, y \rangle \}, A \times (B \times C) = \{ \langle 1, \langle a, x \rangle \rangle, \langle 1, \langle a, y \rangle \rangle, \langle 1, \langle b, x \rangle \rangle, \langle 1, \langle b, y \rangle \rangle \}$ , 显然这两个集合是不相等的。

### Answer of exercise 7

判断关系是不是一个函数，就要看是否每个  $x$  是否有唯一的  $y$  对应，显然这是符合定义的。

### Answer of exercise 8

满射就是值域中所有的元素都有一个像对应，单射是定义域中没有两个不同的元素像相同，双射就是既是满射又是单射。

### Answer of exercise 9

$$2 \mid n, 5 \mid n, 7 \mid n \Rightarrow \text{lcm}(2, 5, 7) \mid m \Rightarrow 70 \mid n$$

### Answer of exercise 10

$$m(m+1)(m+2)$$

$$m = 1, 1 \times 2 \times 3 = 6, \text{ 成立。}$$

设  $m=k$  也成立。

$m = k + 1, (k+1)(k+2)(k+3) = (k+1)(k+2)k + 3(k+1)(k+2)$ , 可见式子的第一部分可被 6 整除, 要想证明  $3(k+1)(k+2)$  可被 6 整除, 可证明  $(k+1)(k+2)$  可被 2 整除, 不管  $k$  是奇数还是偶数, 两个连续项一定有个偶数项, 所以可以被 2 整除。

### Answer of exercise 11

奇数可以写为  $2k+1$ , 奇数的平方是  $(2k+1)^2 = 4k^2 + 4k + 1 = 4k(k+1) + 1$ , 可知  $k(k+1)$  是偶数, 所以可以写成  $2m$  的形式,  $4k(k+1) + 1$  可以写成  $8m+1$  形式。

### Answer of exercise 12

$\text{gcd}(55, 85) = 5, \text{gcd}(55, 85) = 2, \text{gcd}(666, 1414) = 2, \text{gcd}(20785, 44350) = 5$ , 下面给出第一对数的实际计算过程:

$$85 = 55 \times 1 + 30$$

$$55 = 30 \times 1 + 25$$

$$30 = 25 \times 1 + 5$$

$$25 = 5 \times 5$$

$$\text{gcd}(55, 85) = 5$$

### Answer of exercise 13

先求  $\text{gcd}(231, 732), \text{lcm}(231, 732) = (231 \times 732) \div \text{gcd}(231, 732)$ , 计算结果如下:  
 $\text{lcm}(231, 732) = 56364, \text{lcm}(-871, 728) = 48776$

### Answer of exercise 14

利用小于次数的素数去依次除次数, 可以整除, 则找到一个素因子, 依次可以找到所有素因子。分解结果如下 (sagemath factor):

$$36 = 2^2 * 3^2, 69 = 3 * 23, 200 = 2^3 * 5^2, 289 = 17^2$$

### Answer of exercise 15

利用小于次数的素数去依次除次数, 可以整除, 则找到一个素因子, 依次可以找到所有素因子。分解结果如下:

$$625 = 5^4, 2154 = 2 * 3 * 359, 2838 = 2 * 3 * 11 * 43, 3288 = 2^3 * 3 * 137$$

### Answer of exercise 16

依次计算上面各数模 16 的余数, 余数分别为 1、9、0、1、9、0, 根据同余定义, 我们有  $1 \equiv 17(\text{mod } 16), 9 \equiv 25(\text{mod } 16), 16 \equiv 160(\text{mod } 16)$ 。

**Answer of exercise 17**

答案是：9, 计算过程如下：

$$7^2 = 49 \equiv 9(\text{mod } 10), 9^2 = 81 \equiv 1(\text{mod } 10), 2046 = 4 \times 511 + 2 \Rightarrow 7^{2046} \equiv ((7^2)^2)^{511} \times 7^2 \equiv 9(\text{mod } 10)$$

**Answer of exercise 18**

答案为：76, 计算过程如下：

$$\begin{aligned} 2^0 &= 1(\text{mod } 100), 2^1 = 2(\text{mod } 100), 2^2 = 4(\text{mod } 100), 2^3 = 8(\text{mod } 100), 2^4 = 16(\text{mod } 100) \\ 2^5 &= 32(\text{mod } 100), 2^6 = 64(\text{mod } 100), 2^7 = 28(\text{mod } 100), 2^8 = 56(\text{mod } 100), 2^9 = 12(\text{mod } 100) \\ 2^{10} &= 24(\text{mod } 100), 2^{11} = 48(\text{mod } 100), 2^{12} = 96(\text{mod } 100), 2^{13} = 92(\text{mod } 100), 2^{14} = 84(\text{mod } 100) \\ 2^{15} &= 68(\text{mod } 100), 2^{16} = 36(\text{mod } 100), 2^{17} = 72(\text{mod } 100), 2^{18} = 44(\text{mod } 100), 2^{19} = 88(\text{mod } 100) \\ 2^{20} &= 76(\text{mod } 100), 2^{21} = 52(\text{mod } 100), 2^{22} = 4(\text{mod } 100), 2^{23} = 8(\text{mod } 100), 2^{24} = 16(\text{mod } 100) \\ 2^{25} &= 32(\text{mod } 100), 2^{26} = 64(\text{mod } 100), 2^{27} = 28(\text{mod } 100), 2^{28} = 56(\text{mod } 100), 2^{29} = 12(\text{mod } 100) \\ 2^{30} &= 24(\text{mod } 100), 2^{31} = 48(\text{mod } 100), 2^{32} = 96(\text{mod } 100), 2^{33} = 92(\text{mod } 100), 2^{34} = 84(\text{mod } 100) \\ 2^{35} &= 68(\text{mod } 100), 2^{36} = 36(\text{mod } 100), 2^{37} = 72(\text{mod } 100), 2^{38} = 44(\text{mod } 100), 2^{39} = 88(\text{mod } 100) \\ 1000 &= 20 \times 50 \Rightarrow 2^{1000} = 2^{20 \times 50} \equiv 2^{20} \equiv 76(\text{mod } 100) \end{aligned}$$

**Answer of exercise 19**

由于  $2^1 \equiv 2(\text{mod } 7), 2^2 \equiv 4(\text{mod } 7), 2^3 \equiv 1(\text{mod } 7)$

所以  $2^{3 \times 33} \equiv 1(\text{mod } 7)$

又因为  $100 = 3 \times 33 + 1$ , 所以  $2^{100} = 2^{3 \times 33 + 1} = 2^{3 \times 33} \times 2 \equiv 2(\text{mod } 7)$ , 所以第  $2^{100}$  天是星期三。同理  $200 = 3 \times 66 + 2, 2^{200} = 2^{3 \times 66 + 2} = 2^{3 \times 66} \times 2^2 \equiv 4(\text{mod } 7)$ , 所以第  $2^{200}$  天是星期五。

**Answer of exercise 20**

$$4^5 + 5^5 + 6^5 + 7^5 \equiv 0 + 1^5 + 2^5 + 3^5(\text{mod } 4)$$

...

$$96^5 + 97^5 + 98^5 + 99^5 \equiv 0 + 1^5 + 2^5 + 3^5(\text{mod } 4) \text{ 共有 } 24 \text{ 组}, 24 \times (1 + 32 + 3^5) \equiv 0(\text{mod } 4)$$

**Answer of exercise 21**

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8\}, 2k + 1, \{1, 3, 5, 7, 9, 11, 13, 15, 17\}$$

**Answer of exercise 22**

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8\}, 2k, \{0, 2, 4, 6, 8, 10, 12, 14, 16\}$$

**Answer of exercise 23**

模 5 的完全剩余系  $\{0, 1, 2, 3, 4\}$

模 6 的完全剩余系  $\{0, 1, 2, 3, 4, 5\}$

5, 6 互质,  $30 = 5 \times 6$

$$m_i \times 5 + n_j \times 6, \{0, 1, 2, 3, 4, 5, 11, 17, 23, 29, 35, 10, 16, 22, 28, 34, 40, 15, 21, 27, 31, 39, 45, 20, 26, 32, 38, 44, 50\}$$

#### Answer of exercise 24

最小正完全系  $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ , 与 12 互素的有  $\{1, 5, 7, 11\}$

#### Answer of exercise 25

5 和 6 互素,  $x$  和  $y$  遍历 5 和 6 的缩系,  $5x + 6y$  也遍历模  $5 \times 6$  的缩系。模 5 的缩系  $\{1, 2, 3, 4\}$ , 模 6 的缩系  $\{1, 5\}$ , 模 30 的缩系  $11, 16, 21, 30, 35, 40, 45, 50$

#### Answer of exercise 27

$$\begin{aligned} \because 8 &\equiv 1 \pmod{7} \\ 9 &\equiv 2 \pmod{7} \\ 10 &\equiv 3 \pmod{7} \\ 11 &\equiv 4 \pmod{7} \\ 12 &\equiv 5 \pmod{7} \\ 13 &\equiv 6 \pmod{7} \\ \therefore 720 &\equiv 6 \pmod{7} \end{aligned}$$

#### Answer of exercise 28

229 模 281 逆元是 27.

具体解法是首先判断  $\gcd(229, 281) = 1$ , 逆元存在, 然后利用扩展欧几里得算法求逆元

$$s_i = s_{i-2} + q_{i-1}s_{i-1}, t_i = t_{i-2} + q_{i-1}t_{i-1}:$$

i	$r_i$	$q_i$	$s_i$	$t_i$
0	281	-	1	0
1	229	1	0	1
2	52	4	1	-1
3	21	2	-4	5
4	10	2	9	-11
5	1	10	-22	27
5	0	-	-	-

#### Answer of exercise 29

利用欧几里得扩展算法计算, 计算结果为: 2887.

sagemath 验证语句: `3169.inverse_mod(3571)`

#### Answer of exercise 30

根据欧几里得扩展算法，我们有  $\gcd(r_0, r_1) = s_n r_0 + t_n r_1$ ，我们设  $r_0 = 121, r_1 = 105$ ，同时我们可知 105 与 121 互素 ( $\gcd(121, 105) = 1$ )，可见欧几里得扩展算法最后的等式为  $121s_n + 105t_n = 1$ ，与所求的方程相同，我们利用扩展欧几里得算法进行计算：

	$r_i$	$q_i$	$s_i$	$t_i$
0	121	-	1	0
1	105	1	0	1
2	16	6	1	-1
3	9	1	-6	7
4	7	1	7	-8
5	2	3	-13	15
6	1	2	46	-53
7	0			

#### Answer of exercise 31

$\gcd(27, 15) = 3$ ，所以同余方程共有 3 个解，同余方程  $9x \equiv 4 \pmod{5}$  的一个特解是  $x=1$ ，所以原方程全部解为  $x \equiv 1 + \frac{15}{3}t \pmod{15}, t \in 0, 1, 2$ ，求得其解为 1, 6, 11.

#### Answer of exercise 32

$\gcd(24, 81) = 3$ ，所以同余方程共有 3 个解，同余方程  $8x \equiv 2 \pmod{27}$  的一个特解是  $x=7$ ，所以原方程全部解为  $x \equiv 7 + \frac{81}{3}t \pmod{81}, t \in 0, 1, 2$ ，求得其解为 7, 34, 61.

#### Answer of exercise 33

$\gcd(91, 169) = 13$ ，并且  $13 \mid 26$ ，所以同余方程共有 13 个解，同余方程  $7x \equiv 2 \pmod{13}$  的一个特解是  $x=4$  (可以通过遍历的方法求得)，所以原方程全部解为  $x \equiv 4 + \frac{169}{13}t \pmod{169}, t \in 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12$ ，求得其解为 4, 17, 30, 43, 56, 69, 82, 95, 108, 121, 134, 147, 160.

#### Answer of exercise 34

$\gcd(72, 200) = 8$ ，但是  $8 \nmid 47$ ，所以第一个方程无解。 $\gcd(4183, 15087) = 47$ ，且  $47 \mid 5781$ ，故第二方程有解，且解的个数为 47。 $\gcd(1537, 6731) = 53$ ，但是  $53 \nmid 2863$ ，故此方程无解。

#### Answer of exercise 35

$\gcd(12, 25) = 1$ ，利用中国剩余定理，唯一解为  $x \equiv 25 \times 1 \times 9 + 12 \times 23 \times 6 \pmod{12 \times 25} \equiv 1881 \equiv 81 \pmod{300}$

#### Answer of exercise 36

$\gcd(7, 15) = 1, \gcd(15, 22) = 1, \gcd(7, 22) = 1$ ，可以看出以上方程组的模两两互素，根据中国剩余定理，我们有：

$$x \equiv 5 \pmod{7}, m = 7 \times 15 \times 22 = 2310$$

$$M_1 = 15 \times 22 = 330, M'_1 = 1 \pmod{7}$$

$$M_2 = 7 \times 22 = 154, M'_2 = 4 \pmod{15}$$

$$M_3 = 7 * 15 = 105, M'_3 = 13(mod 22)$$

$$x = M_1 M'_1 b_1 + M_2 M'_2 b_2 + M_3 M'_3 b_3 = 15 \times 1 \times 5 + 154 \times 4 \times 12 + 105 \times 13 \times 18(mod 2310) \equiv 1272(mod 2310)$$

### Answer of exercise 37

先看  $3x \equiv 12(mod 5)$  的解, 由于  $gcd(3, 5) = 1$ , 所以  $x \equiv 12 \times 3^{\varphi(5)-1} \equiv 12 \times 3^3 \equiv 324 \equiv 4(mod 5)$

再看  $4x \equiv 18(mod 7)$  的解, 由于  $gcd(4, 7) = 1$ , 所以  $x \equiv 4 \times 18^{\varphi(7)-1} \equiv 4 \times 18^5 \equiv 1(mod 7)$

也可以直接化简为以下方程组。

方程组等价于:

$$\begin{cases} x \equiv 5(mod 9) \\ x \equiv 4(mod 5) \\ x \equiv 1(mod 7) \end{cases}$$

由于 9,5,7 两两互素, 根据中国剩余定理计算可得  $x=239$ .

### Answer of exercise 38

根据题意列出方程:

$$\begin{cases} x \equiv 1(mod 3) \\ x \equiv 2(mod 5) \\ x \equiv 2(mod 7) \end{cases}$$

3,5,7 两两互素, 根据中国剩余定理,  $m = 3 \times 5 \times 7 = 105, M_1 = 35, M'_1 = 2, M_2 = 21, M'_2 = 1, M_3 = 15, M'_3 = 1, x \equiv 35 \times 2 \times 1 + 21 \times 1 \times 2 + 15 \times 1 \times 2 \equiv 142 \equiv 37(mod 105)$

### Answer of exercise 39

23 的一个完全剩余系为 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23, 依次计算  $i^2(mod 23)$ , 计算结果组成的集合就是模 23 二次剩余组成的集合, 完全剩余系中不包含在此集合中的元素就是非二次剩余。

$$\begin{aligned} 1^2 &= 1(mod 23), 2^2 = 4(mod 23), 3^2 = 9(mod 23), 4^2 = 16(mod 23) \\ 5^2 &= 2(mod 23), 6^2 = 13(mod 23), 7^2 = 3(mod 23), 8^2 = 18(mod 23) \\ 9^2 &= 12(mod 23), 10^2 = 8(mod 23), 11^2 = 6(mod 23), 12^2 = 6(mod 23) \\ 13^2 &= 8(mod 23), 14^2 = 12(mod 23), 15^2 = 18(mod 23), 16^2 = 3(mod 23) \\ 17^2 &= 13(mod 23), 18^2 = 2(mod 23), 19^2 = 16(mod 23), 20^2 = 9(mod 23) \\ 21^2 &= 4(mod 23), 22^2 = 1(mod 23), 23^2 = 0(mod 23) \end{aligned}$$

二次剩余为 0, 1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18

非二次剩余 5, 7, 10, 11, 14, 15, 17, 19, 20, 21, 22

### Answer of exercise 40

首先观察方程的左边是  $y^2 \pmod{7}$  的形式，其最终结果是模 7 的二次剩余，我们计算模 7 的二次剩余，可知为：0, 1, 2, 4。

$$1^2 \equiv 1 \pmod{7}, 2^2 \equiv 4 \pmod{7}, 3^2 \equiv 2 \pmod{7}, 4^2 \equiv 2 \pmod{7}, 5^2 \equiv 4 \pmod{7}, 6^2 \equiv 1 \pmod{7}, 7^2 \equiv 0 \pmod{7}$$

$$(7k)^2 \equiv 0 \pmod{7}, (7k+1)^2 \equiv 1 \pmod{7}, (7k+3)^2 \equiv 2 \pmod{7}, (7k+5)^2 \equiv 4 \pmod{7} \text{ 以上方程等价于:}$$

$$x^2 - 2x + 1 \equiv 0 \pmod{7} \text{ or } x^2 - 2x + 1 \equiv 1 \pmod{7} \text{ or } x^2 - 2x + 1 \equiv 2 \pmod{7} \text{ or } x^2 - 2x + 1 \equiv$$

$4 \pmod{7}$ , 对这四个同余式进一步变形，得

$$(x-1)^2 \equiv 0 \pmod{7} \text{ or } (x-1)^2 \equiv 1 \pmod{7} \text{ or } (x-1)^2 \equiv 2 \pmod{7} \text{ or } (x-1)^2 \equiv 4 \pmod{7},$$

根据模 7 的二次剩余，进一步知道：

$$x \equiv 1 \pmod{7}, x \equiv 9 \pmod{7}, x \equiv 4 \pmod{7}, x \equiv 6 \pmod{7}$$

所以满足以上方程的点为 (0,1)(1,9)(6,9)(3,4)(4,4)(2,6)(5,6) 只要对应的 y 和 x 于这些点同余就满足方程。

#### Answer of exercise 41

29 是奇素数,  $\gcd(2, 29)=1$ , 根据欧拉判别条件  $a^{\frac{p-1}{2}} = 2^{\frac{29-1}{2}} = 2^14 = 16384 \equiv -1 \pmod{29}$ , 所以 2 不是 29 的二次剩余。

#### Answer of exercise 42

73 为奇素数，根据书中定理我们有

$$\left(\frac{2}{73}\right) = \begin{cases} 1, & \text{if } p \equiv \pm 1 \pmod{8} \\ -1, & \text{if } p \equiv \pm 3 \pmod{8} \end{cases},$$

$\therefore 73 \pmod{8} \equiv 1, \therefore \left(\frac{2}{73}\right) = 1, 2$  是 73 的二次剩余。

#### Answer of exercise 44

$$\gcd(37, 25411) = 1$$

根据二次互反定律，我们有：

$$\left(\frac{37}{25411}\right) = (-1)^{\frac{37-1}{2} \frac{25411-1}{2}} \left(\frac{25411}{37}\right) = \left(\frac{29}{37}\right) = (-1)^{\frac{29-1}{2} \frac{37-1}{2}} \left(\frac{37}{29}\right) = \left(\frac{8}{29}\right), \text{ 然后可以根据二次剩余的} \\ \text{定义, 知 } \left(\frac{8}{29}\right) = 8^{\frac{29-1}{2}} \pmod{29} \equiv 28 \equiv -1 \pmod{29}$$

#### Answer of exercise 45

解法一：可以按照定义去求，遍历一个完全系： $34^0 = 1, 34^1 = 34, 34^2 = 9, 34^3 = 10, 34^4 = 7, 34^5 = 16, 34^6 = 26, 34^7 = 33, 34^8 = 12, 34^9 = 1, 34^{10} = 34, 34^{11} = 9, 34^{12} = 10, 34^{13} = 7, 34^{14} = 16, 34^{15} = 26, 34^{16} = 33, 34^{17} = 12, 34^{18} = 1, 34^{19} = 34, 34^{20} = 9, 34^{21} =$

$10, 34^{22} = 7, 34^{23} = 16, 34^{24} = 26, 34^{25} = 33, 34^{26} = 12, 34^{27} = 1, 34^{28} = 34, 34^{29} = 9, 34^{30} = 10, 34^{31} = 7, 34^{32} = 16, 34^{33} = 26, 34^{34} = 33, 34^{35} = 12, 34^{36} = 1$

由计算结果可知 34 对模 37 的次数为 9.

解法二:

可以证明,  $\text{ord}_m(a) \mid \varphi(m)$ , 所以, 37 的欧拉函数是 36, 6 的所有因子是 1, 2, 3, 4, 6, 9, 12, 18, 36, 依次计算  $a^l \pmod{37}, l \in 1, 2, 3, 4, 6, 9, 12, 18, 36$ , 结果为 1 的  $l$  最小取值为次数.

#### Answer of exercise 46

47 是奇素数, 根据定理, 我们知道 47 的原根存在。

55 的标准分解式  $5 \times 11$ , 根据定理其不是  $2, 4, p^l, 2p^l$  的形式, 故 55 没有原根。

59 是奇素数, 根据定理, 我们知道 59 的原根存在。

#### Answer of exercise 47

47 是奇素数, 根据定理, 我们知道 47 的原根存在。

$\varphi(47) = 46$ , 46 的标准分解式为  $2 \times 23$ , 46 互素的因子  $g$  为 3, 5, 7, 9, ..., 计算  $g^{23}, g^2$ :

$3^{23} \equiv 1, 3^2 \equiv 9, 3$  不是原根

$4^{23} \equiv 1, 4^2 \equiv 16, 4$  不是原根

$5^{23} \equiv 46, 5^2 \equiv 25, 5$  是原根

$5^l, l$  遍历 46 的缩系,  $\{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45\}$ , 得到所有原根为  $\{5, 10, 11, 13, 15, 19, 20, 22, 23, 26, 29, 30, 31, 33, 35, 38, 39, 40, 41, 43, 44, 45\}$ .



## 附录 B 编程练习

- 1、实现一个简单的交互式命令行界面，可以输入要计算的函数，输入 `exti` 退出。参考工程为 gitee 上的 `calculator` 项目（<https://github.com/btmills/calculator>）。
- 2、利用 GNU MP 实现以下算法（不能直接使用 GNU MP 已有函数）。

序号	命令行接口	输出示例	说明
1	<code>prime_erat(n,m)</code>	3,5,7	利用 Eratosthenese 筛选法，从屏幕输出从 $n$ 到 $m$ 的所有素数，如果此范围内不存在素数，输出 <code>none</code>
2	<code>gcd(n,m)</code>	5	利用欧几里得算法实现求解 $n$ 和 $m$ 的最大公约数的算法
3	<code>lcm(n,m)</code>	12	求解 $n$ 和 $m$ 的最小公倍数的算法
4	<code>factor(n)</code>	$22*3*5$	求解整数 $n$ 的标准分解式
5	<code>eulerfun(n)</code>	2	计算 $n$ 的欧拉函数
6	<code>inverse(n,m)</code>	550	实现扩展欧几里得算法，求解 $n$ 模 $m$ 的逆元，如果不存在，输出 <code>none</code> 。
7	<code>crt(a,b,c,d,e,f)</code>	28	实现 CRT，求解 $\begin{cases} x \equiv a \pmod{b} \\ x \equiv c \pmod{d} \\ x \equiv e \pmod{f} \end{cases}$
8	<code>order(a,m)</code>	5	$a$ 对模 $m$ 的次数
9	<code>primroot(m)</code>	6,7,11,12,13,15,17,19,22,24,26,28,29,30,34,35	求 $m$ 的原根
10	<code>legendresym(a,p)</code>	-1	利用二次互反律，计算 $a$ 对 $p$ 的 Legendre 符号

- 3、实验报告中要有框架和功能测试，要编写测试用例。测试用例的基础知识可以在网上查阅相关资料了解。
- 4、实验报告中的流程图绘制要求参考百度百科“程序流程图”。

## 附录 C 特定标识说明

---

$\mathbb{N}$  自然数集合

$\mathbb{Z}$  整数集合

$\mathbb{Q}$  有理数集合

$\mathbb{C}$  复数集合

gcd      greatest common divider

lcm      Lowest Common Multiple

$[\frac{a}{b}]$       a 除 b 的余数

[2]      这个表示引自的参考文献，如果是一条定理有此标识，表示在整理此书时，此定理的表述参考的文献。

## 附录 D 工具说明

---

本书编写环境是 TexStudio+MikTeX。

TexStudio 是一个 Tex 的编辑环境，并且集成了编译环境环境，其目前是一个开源项目，网址为 <http://texstudio.sourceforge.net/>。

MikTeX 是 TeX/Latex 的一个具体实现的引擎（也就是可以将 tex 文件转换为便于阅读的 pdf 或 ps 文件），其中还包含包管理功能，目前是一个开源项目，网址为 <https://miktex.org/>。

书中的脑图是用 freemind 绘制，其为一个开源项目，java 开发，跨平台性好，网址为：[http://freemind.sourceforge.net/wiki/index.php/Main\\_Page](http://freemind.sourceforge.net/wiki/index.php/Main_Page)

在书的编写前期，还使用了 Mathpix Snipping Tool，这个工具可以将看到的任何公式，转换为 Tex 的公式描述，识别率很高，一个不错的工具，网址：<https://mathpix.com/>。

在本书使用到了 SageMath 工具软件，SageMath 是一个开源的数学软件，整合了许多开源的 Python 包（如：NumPy, SciPy, matplotlib, Sympy, Maxima, GAP, FLINT, R 等），采用 Python 语言编写，同时也是一个编程环境，使用 Python 语言可以编写计算程序，官网为：<http://www.sagemath.org/>，官网上有很多说明文档，可以参考。本课程选择他，首先是因为他是开源软件，另外他是基于 Python 的，比较容易上手，另外其对数论和代数支持比较好。

## 附录 E 用到的模板

---

首先本书的模板来自于 <https://github.com/ElegantLaTeX/ElegantBook>, 此模板的信息来自于“Latex 开源小屋”网址上的一个帖子 <https://www.latexstudio.net/archives/51589.html>。本书使用的试卷模板信息也是来自于这个网站, 从下载的模板信息中可知, 模板作者“高星”, 作者来自于湖南潇湘技师学院, 湖南九嶷 (yi) 职业技术学院, 试卷模板地址是 <https://github.com/gnixoag/myworks2017/tree/master/16jidazhuan>。

谢谢!

## 附录 F 课程中有关 SageMath 函数

---

本部分内容主要参考的是 SageMath 网站上的参考文献，其中参考最多的是一个快速参考 [14].

1. 求余 remainder,  $n \% m$
2. 最大公约数, gcd(n,m), gcd(list)
3. extended gcd(扩展欧几里得算法) g,s,t=xgcd(a,b)
4. 最小公倍数, lcm(a,b), lcm(list)
5. 是否能整除, n.divides(m), 返回 true, 表示  $n \mid m$
6. 某数的所有因子, n.divisors()
7. 整数标准分解, factor(n), 整数 n 的标准分解
8. 素性检测, is\_prime(n), n 是素数返回 true, 否则返回 false
9. 欧拉方程, euler\_phi(n)
10. 次数, Mod(a,m).multiplicative\_order(), 求 a 模 m 的次数
11. 二次剩余, quadratic\_residues(n), 示例:  $Q = \text{quadratic\_residues}(23)$ ; Q
12. 原根, primitive\_root(n), 求 n 的原根
13. 逆元, n.inverse\_mod(m), n 的模 m 逆元
14. power\_mod(a,n,m), 计算的是  $a^n \pmod m$
15. 中国剩余定理, x=crt(a,b,m,n), 表示的是满足  $x \equiv a \pmod m, x \equiv b \pmod n$  的 x

## 附录 G Latex 编写格式说明

### G.1 使用的环境

### G.2 定义

整除定义示例。

```
\begin{definition}{整除}{int}
 $a, b \in \mathbb{Z}, b \neq 0$ , 如果存在  $q \in \mathbb{Z}$ , 使得  $a = qb$ ,
就称  $a$  可被  $b$  整除 或者  $b$  整除  $a$ , 记为  $b \mid a$ 。
 $a$  是  $b$  的倍数,  $b$  是  $a$  的因子 (或约数、除数)。若  $a$  不能被  $b$  整除, 记为  $b \nmid a$ 。
\end{definition}
```

### G.3 定理

```
\begin{theorem}{}{set}
 $X$  和  $Y$  为有限集, 且  $X$  和  $Y$  的元素个数相同 (记为  $|X| = |Y|$ ), 则  $f: X \rightarrow Y$  是单射, 当且仅当它
\end{theorem}
```

### G.4 示例

```
\begin{example}
证明幂等律  $A \cup A = A$ 。
\end{example}
```

### G.5 示例解答

```
\begin{solution}
    bla bla bla。
\end{solution}
```

### G.6 证明

```
\begin{proof}
(1) 假设结论不成立, 那么至少存在一个元素属于空集, 但不属于  $A$ , 按照空集的定义, 显然任何元素
(2) 反证法: 假设有两个不同的空集  $\phi_1, \phi_2$ , 根据空集性质  $\phi_1 \subseteq \phi_2$ ,
\end{proof}
```

## G.7 SageMath 示例代码

```
\begin{SageMath}{罗列方式定义有限集合}
\begin{lstlisting}
sage: #有限集的定义：罗列的方式
.....: x=Set([1,2,3,4,5,6])
.....: x
.....: x.is_finite()
.....:
{1, 2, 3, 4, 5, 6}
True
\end{lstlisting}
\end{SageMath}
```

## G.8 贴图

```
\begin{figure}[!htbp]
\centering
\includegraphics[width=0.6\textwidth]{mpg.png}
\caption{MPG 和 Weight 的关系图\label{fig:mpg}}
\end{figure}
```

## 附录 H 扩展阅读

---

### H.1 预言机 (Oracle Machine)

Loosely speaking, an oracle machine is a machine that is augmented such that it may pose questions to the outside. We consider the case in which these questions, called queries, are answered consistently by some function  $f : \{0, 1\}^* \rightarrow \{0, 1\}$ , called the oracle. That is, if the machine makes a query  $q$  then the answer it obtains is  $f(q)$ . In such a case, we say that the oracle machine is given access to the oracle  $f$ . For an oracle machine  $M$ , a string  $x$  and a function  $f$ , we denote by  $M^f(x)$  the output of  $M$  on input  $x$  when given access to the oracle  $f$ . (Re-examining the second part of the proof of Theorem 1.5, observe that we have actually described an oracle machine that computes  $d'$  when given access to the oracle  $d$ .)

The notion of an oracle machine extends the notion of a standard computing device (machine), and thus a rigorous formulation of the former extends a formal model of the latter. Specifically, extending the model of Turing machines, we derive the following model of oracle Turing machines.

Definition 1.11 (using an oracle):

- An oracle machine is a Turing machine with a special additional tape, called the oracle tape, and two special states, called oracle invocation and oracle spoke.
- The computation of the oracle machine  $M$  on input  $x$  and access to the oracle  $f : \{0, 1\}^* \rightarrow \{0, 1\}$  is defined based on the successive configuration function. For configuration with state different from oracle invocation the next configuration is defined as usual. Let  $\gamma$  be a configuration in which the machine's state is oracle invocation and suppose that the actual contents of the oracle tape is  $q$  (i.e.,  $q$  is the contents of the maximal prefix of the tape that holds bit values). Then, the configuration following  $\gamma$  is identical to  $\gamma$ , except that the state is oracle spoke, and the actual contents of the oracle tape is  $f(q)$ . The string  $q$  is called  $M$ 's query and  $f(q)$  is called the oracle's reply.
- The output of the oracle machine  $M$  on input  $x$  when given oracle access to  $f$  is denoted  $M^f(x)$ .

We stress that the running time of an oracle machine is the number of steps made during its (own) computation, and that the oracle's reply on each query is obtained in a single step.<sup>1</sup>

---

<sup>1</sup>以上文字来自<https://www.wisdom.weizmann.ac.il/~oded/CC/r1.pdf>



## H.2 布尔电路 (Boolean Circuits)

布尔电路模型是一个非一致性计算模型，布尔电路在开始使用时是为了描述实际电路的逻辑操作，然而，喜剧性的是，该模型现在为复杂性理论中某些与实用最无关的研究提供的研究工具。

A Boolean circuit is a collection of gates and wires that performs a mapping from Boolean inputs to Boolean outputs. The accepted wisdom is that such circuits must have acyclic (i.e., loop-free or feed-forward) topologies. In fact, the model is often defined this way—as a directed acyclic graph (DAG). And yet simple examples suggest that this is incorrect. We advocate that Boolean circuits should have cyclic topologies (i.e., loops or feedback paths). In other work, we demonstrated the practical implications of this view: digital circuits can be designed with fewer gates if they contain cycles. In this paper, we explore the theoretical underpinnings of the idea. We show that the complexity of implementing Boolean functions can be lower with cyclic topologies than with acyclic topologies. With examples, we show that certain Boolean functions can be implemented by cyclic circuits with as little as one-half the number of gates that are required by equivalent acyclic circuits. We also show a quadratic upper bound: given a cyclic Boolean circuit with  $m$  gates, there exists an equivalent acyclic Boolean circuit with  $m^2$  gates.<sup>2</sup>

An  $n$ -ary Boolean function  $f(x_1, \dots, x_n)$  is a function from  $\{0, 1\}^n$  with values in  $\{0, 1\}$ . The values 0, 1 are called Boolean values, or bits, or truth values if they are identified with false and true (cf. also Boolean function). Elements of  $\{0, 1\}$  are Boolean strings of length  $n$  (or bit strings).  $B_n$  is the set of all  $2^{2^n}$   $n$ -ary Boolean functions.

A basis of connectives is any non-empty set  $\Omega$  of Boolean functions, not necessarily of the same arity. An  $\Omega$ -formula for  $f(x_1, \dots, x_n)$  is a formula built from Boolean variables  $x_1, \dots, x_n$  using connectives from  $\Omega$ . A basis  $\Omega$  is complete if all Boolean functions are definable by an  $\Omega$ -formula. Examples of complete bases are:  $B_2$ , the set of all binary Boolean functions, or the DeMorgan basis  $\{0, 1, \neg, \vee, \wedge\}$ .

A more general way to **define (equivalently, compute) Boolean functions is by circuits** (straight line programs) that are simplified mathematical models of integrated circuits. A Boolean circuit over a basis  $\Omega$  for  $f \in B_n$  is a sequence  $g_1, \dots, g_k$  of functions from  $B_n$  such that  $g_k = f$  and such that any  $g_i$  is either one of the projections  $x_1, \dots, x_n$  or is equal to  $h(g_{j_1}, \dots, g_{j_r})$ , for some  $h \in \Omega$  and  $j_1, \dots, j_r < i$ . Drawing directed edges from all such  $j_1, \dots, j_r$  to  $i$ , and labelling  $i$  by  $h$ , defines a labelled directed acyclic graph (cf. also Graph, oriented). The size of the circuit is  $k$ . The maximum length of a path from a vertex corresponding to one of the inputs  $x_1, \dots, x_n$  to  $g_k$  is the depth of the circuit. Note that formulas are circuits whose graphs are trees.

<sup>2</sup>文字来源<https://experts.umn.edu/en/publications/cyclic-boolean-circuits>

There are three basic ways to measure the complexity of a Boolean function, given a basis  $\Omega$ :

the minimum size  $L_\Omega(f)$  of an  $\Omega$ -formula defining  $f$ ,

the minimum size  $C_\Omega(f)$  of a circuit over  $\Omega$  defining  $f$ , and

the minimum depth  $D_\Omega(f)$  of a circuit over  $\Omega$  defining  $f$ . For two complete bases  $\Omega, \Omega'$ , the measures  $C_\Omega(f)$ ,  $C_{\Omega'}(f)$  and  $D_\Omega(f)$ ,  $D_{\Omega'}(f)$  are proportional to each other, and  $L_\Omega(f)$ ,  $L_{\Omega'}(f)$  are polynomially related. A relation exists between the formula size and the circuit depth (see [a13]):  $\log(L_\Omega(f))$  and  $D_\Omega(f)$  are proportional to each other, provided  $\Omega$  is finite.

By estimating the number of circuits of a bounded size, C.E. Shannon [a11] proved (cf. [a14] for an optimal computation) that for almost-all  $f \in B_n$ ,  $C_{B_2}(f) \geq 2^n/n$ . On the other hand,

$$C_{B_2}(f) \leq \frac{2^n}{n}(1 + o(1)),$$

for all  $f \in B_n$ , see [a8].

For a formal language  $L \subseteq \{0, 1\}^*$ , let  $L_n$  be the characteristic function of  $L \cap \{0, 1\}^n$ . The main open problem in Boolean complexity theory (cf. [a4]) is whether there is a language  $L$  accepted by a non-deterministic polynomial-time Turing machine (cf. also Turing machine) such that  $C_\Omega(L_n)$  is not bounded by any polynomial in  $n$ . The importance of this question stems from the fact that the affirmative answer implies that the computational complexity classes  $P$  and  $NP$  are different, as  $C_{B_2}(L_n)$  is bounded by a polynomial for any  $L$  decidable in polynomial time (cf. [a10] and also  $NP$ ).

No super-linear lower bounds on  $C_{B_2}(L_n)$  are known for any  $L \in NP$ . However, if additional restrictions are put on the circuits, then strong lower bounds are known. Most notable are two results: any circuit of depth  $d \geq 2$  over a DeMorgan-like basis  $\{0, 1, \neg, \vee, \wedge\}$ , allowing  $\vee, \wedge$  of unbounded arity, computing the parity function  $\oplus_n$  must have size at least  $\exp(\omega(n^{1/d}))$  (see [a1], [a5], [a15], [a6]), and any circuit in the monotone basis  $\{0, 1, \vee, \wedge\}$  computing whether or not a graph on  $n$  vertices contains a clique of size at least  $k \leq n^{1/4}$  must have size at least  $n\omega(\sqrt{k})$  (see [a9], [a3]).

Interesting upper bounds are also known: a DeMorgan circuit of size for multiplication of two  $n$ -bit long numbers, [a12] (it is still open (1998) if there are linear size circuits), and a monotone DeMorgan sorting network of simultaneous size and depth  $O(\log n)$ , see [a2].

Various parts of computational complexity (cf. also Complexity theory) are directly related to lower-bound problems for Boolean functions; for example, communication complexity, cf. [a7].<sup>3</sup>

<sup>3</sup>以上文字来自[https://encyclopediaofmath.org/wiki/Boolean\\_circuit](https://encyclopediaofmath.org/wiki/Boolean_circuit)

## H.3 弹跳桌球 (bouncing billiards ball)

Bouncing billiards ball 也是一个计算模型，下面是美国计算机协会 (ACM) 的一个介绍性文章 “Bouncing Balls and Quantum Computing, Don Monroe Communications of the ACM, October 2020, Vol. 63 No. 10, Pages 10-12 DOI:10.1145/3416076 ”<sup>4</sup>。

The history of science and mathematics includes many examples of surprising parallels between seemingly unrelated fields. Sometimes these similarities drive both fields forward in profound ways, although often they are just amusing.

In December, Adam Brown, a physicist at Google, described a surprisingly precise relationship between a foundational quantum-computing algorithm and a whimsical method of calculating the irrational number  $\pi$ . "It's just a curiosity at the moment," but "the aspiration might be that if you find new ways to think about things, that people will use that to later make connections that they'd not previously been able to make," Brown said. "It's very useful to have more than one way to think about a given phenomenon."

In a preprint posted online (but not yet peer-reviewed at press time), Brown showed a mathematical correspondence between two seemingly unconnected problems. One is the well-known Grover search algorithm proposed for quantum computers, which should be faster than any classical equivalent. The other is a surprising procedure in which counting the number of collisions between idealized billiard balls produces an arbitrarily precise value for the  $\pi$ .

### H.3.1 Quantum Algorithms

Quantum computing exploits quantum bits, or qubits, such as ions or superconducting circuits, that can simultaneously represent two distinct states. In principle, a modest number of qubits can represent and manipulate an exponentially larger number of combinations. Exploiting this possibility for computing seemed like a pipe dream, however, until researchers devised algorithms to extract useful information from the qubits. The first such algorithm, described in 1994 by Peter Shor, then at Bell Labs in New Jersey, efficiently finds the prime factors of a number, potentially cracking important cryptography schemes. The trick is to frame the problem as determining the repetition period of a sequence, essentially a Fourier transform, which can be found using global operations on an entire set of qubits.

The second fundamental algorithm, devised in 1996 by Lov Grover working independently at Bell Labs, operates quite differently. "Shor and Grover are the two most canonical quantum algorithms," according to Scott Aaronson of the University of Texas at Austin. "Even today, the vast majority of quantum algorithms that we know are recognizably either 'Shor-inspired' or 'Grover-inspired', or both."

Grover's algorithm is often described as a database search, examining a list of  $N$  items to find

---

<sup>4</sup>网址<https://cacm.acm.org/magazines/2020/10/247584-bouncing-balls-and-quantum-computing/fulltext>

the item that has a desired property. If the list is ordered by some label (for example, alphabetized), any label can be found by repeatedly dividing the list in successive halves, eventually requiring  $\log_2 N$  queries. For an unsorted list, however, checking each item in turn requires, on average  $N/2$  steps (and possibly as many as  $N$ ).

Like other quantum algorithms, Grover's manipulates the entire set of qubits simultaneously, while preserving the relationships between them (prematurely querying any qubit to determine its state turns it into an ordinary bit, squandering any quantum advantage). However, Grover showed the desired item can generally be found with only  $\frac{\pi}{4} \sqrt{N}$  global operations.

This improvement is less than that seen in Shor-style algorithms, which typically are exponentially faster than their classical counterparts. The Grover approach, however, can be applied to more general, unstructured problems, Brown notes.

Grover's algorithm manipulates the entire set of qubits simultaneously, while preserving the relationships between them.

The calculation starts with an equal admixture of all  $N$  qubits. The algorithm then repeatedly subjects all the qubits to two alternating manipulations. The first operation embodies the target: it inverts the state of a specific, but unknown, bit. The task is to determine which bit is altered, but not by measuring them all. The second operation does not require any information about the target. Grover found that each time this sequence is repeated, the weight of the target in the admixture increases (although this cannot be measured). After the correct number of repetitions, there is an extremely high chance a measurement will yield the correct result.

### H.3.2 Bouncing Billiards

These sophisticated quantum manipulations may seem to have little relationship to bouncing billiard balls. Yet Brown, while working on issues related to Grover's algorithm, came across an animation by math popularizer Grant Sanderson that made him notice the similarities. In his paper, Brown shows there is a precise mapping between the two problems.

Sanderson's animation illustrates a surprising observation described in 2003 by Gregory Galperin, a mathematician at Eastern Illinois University in Charleston. In the paper "Playing Pool with ," he imagined two billiard balls moving without friction along a horizontal surface, bouncing off each other and off a wall on the left side in completely elastic collisions (which preserve their combined kinetic energy).

If the right-hand ball is sent leftward toward a second stationary ball that is much lighter, the smaller ball will be sent back toward the left-hand wall without slowing the larger ball much. The small ball will bounce off the wall, and then collide with the large one again, repeating this multiple times. Eventually the collisions will turn the large ball around until it finally escapes to the right faster than the small ball can pursue it.

The number of collisions needed before this escape can occur grows larger with the ratio of the mass of the large ball compared to the small one. If the masses are equal, it will take three



bounces: the first transfers all motion from the right ball to the left one, which bounces off the wall and then transfers its momentum back to the right ball again. If the large ball is 100 times as massive, the process will take 31 bounces. If the mass ratio is 10,000, there will be 314 bounces. In a spectacularly impractical computation, for every increase of a factor of 100 in the mass ratio, the number of collisions (divided by the square root the mass ratio) includes another digit to the digital representation of  $\pi$ , 3.141592654 ...

Brown fortuitously encountered Sanderson's animation (which uses blocks instead of balls) when Grover's algorithm was fresh in his mind, and recognized significant similarities between the two situations. The two quantum operations, for example, correspond respectively to collisions between the balls and between the lighter ball and the wall. The mass ratio corresponds to the size of the database. Moreover, the final result was that the number of operations (or bounces) is proportional to and to the square root of this size or mass ratio. (There are also two factors of two that reflect simple bookkeeping differences between the problems.)

Beyond the surprising connection between such different systems, what on earth is the number doing in both cases? This irrational number is of course best known as the ratio of the circumference of a circle to its diameter, although it also appears in the proportions of ellipses, as well as higher-dimensional objects like spheres. One way to define a circle is through an algebraic constraint on the horizontal and vertical coordinates,  $x$  and  $y$ : The points of a circle with radius  $r$  are constrained to satisfy  $x^2 + y^2 = r^2$ .

As it turns out, both the billiard problem and the Grover algorithm have constraints of this form. Collisions of the balls or manipulations of the quantum system correspond to rotations along the circle defined by these constraints.

For example, for two billiards of mass  $m$  (with velocity  $v_m$ ) and  $M$  (with velocity  $v_M$ ), an elastic collision preserves their total kinetic energy,  $\frac{1}{2}mv_m^2 + \frac{1}{2}Mv_M^2$ . Completely reversing the velocity of the larger ball requires a total "rotation" by  $180^\circ$  ( $\pi$  radians) in the plane with coordinates  $v_m$  and  $v_M$ .

Similarly, for quantum systems, the probability of observing a particular outcome is proportional to the square of the "wave function" corresponding to that outcome. The sum of the probability (squared amplitude) for the target and all other outcomes must be one.

### H.3.3 Historical Examples of Connections

There is still the question, "Is this profound insight into the nature of reality, or is it just a sort of curiosity?" Brown said. "Maybe Grover search is telling us something profound about the nature of reality, and maybe the bouncing-ball thing is more of a curiosity, and maybe connecting them is more in the spirit of the second one than the first one."

Still, there have been numerous cases in physics, and especially in mathematics, where such connections have contributed profoundly to progress. For example, physicists have spent more than two decades exploring a surprising correspondence between strongly interacting multi-

particle quantum systems and gravitational models incorporating curved spacetime with one higher dimension. There is even hope the wormholes in spacetime can help resolve paradoxes associated with quantum-mechanical "entanglement" of distant particles.

For quantum systems, the probability of seeing a particular outcome is proportional to the square of the wave function corresponding to that outcome.

Mathematics has frequently advanced through connections between disparate fields. For example, Fermat's "last theorem," involving integer solutions of a simple equation, was only proved centuries later using methods from "elliptic curves." In another example, in January, computer scientists proved a theorem relating entanglement to Alan Turing's notion of decidable computations, which continues to shake up other seemingly unrelated fields.

For his part, Aaronson suspects the Grover-billiard correspondence, although "striking in its precision," is probably "just a cute metaphor (in the sense that I don't know how to use it to deduce anything about Grover's algorithm that we didn't already know). And that's fine."

### H.3.4 Further Reading

Galperin, G., "Playing Pool with : The Number from a Billiard Point of View)," Regular and Chaotic Dynamics 8, p. 375 (2003).

Brown, A.R., "Playing Pool with |: from Bouncing Billiards to Quantum Search," arXiv.org/abs/1912.02207 (2019).

Sanderson, G., "How Pi Connects Colliding Blocks to a Quantum Search Algorithm," Quanta (2020)

## H.4 康威生命游戏 (Conway's game of life)

康威生命游戏 (Conway's Game of Life), 又称康威生命棋, 是英国数学家约翰·何顿·康威 1970 年发明的细胞自动机, 康威生命游戏引起了各行各业研究者的关注, 这使得细胞机引起了大家的关注。约翰·何顿·康威的这篇论文发表在 "Scientific American", 文章的题目为 "MATHEMATICAL GAMES: The fantastic combinations of John Conway's new solitaire game "life""<sup>5</sup>。下面我们简单介绍一下康威生命游戏。

康威生命游戏, 对于任意细胞, 规则如下:

- 每个细胞有两种状态 - 存活或死亡, 每个细胞与以自身为中心的周围八格细胞产生互动 (通常用图形表示时, 黑色为存活, 白色为死亡)
- 当前细胞为存活状态时, 当周围的存活细胞低于 2 个时 (不包含 2 个), 该细胞变成死亡状态。(模拟生命数量稀少)
- 当前细胞为存活状态时, 当周围有 2 个或 3 个存活细胞时, 该细胞保持原样。
- 当前细胞为存活状态时, 当周围有超过 3 个存活细胞时, 该细胞变成死亡状态。(模拟生命数量过多)

---

<sup>5</sup>文章链接<https://www.ibiblio.org/lifepatterns/october1970.html>



- 当前细胞为死亡状态时，当周围有 3 个存活细胞时，该细胞变成存活状态。（模拟繁殖）

可以把最初的细胞结构定义为种子，当所有在种子中的细胞同时被以上规则处理后，可以得到第一代细胞图。按规则继续处理当前的细胞图，可以得到下一代的细胞图，周而复始。

科学写作者王咏刚写了一篇帖子“康威生命游戏是如何搭建计算机的？”<sup>6</sup>，这篇文章比较完整的描述的康威生命游戏自动机理论到实践的过程。

## H.5 指针机 (pointer machine)

指针机也是一种理论计算模型，他是两种理论计算模型 (计数器机和随机存取 (RAM) 机) 的混合，而这些模型都是寄存器机 (Register machine) 的子类，可以证明任何正确定义的寄存器机都是图灵等价的。

## H.6 欧式几何、罗氏几何和黎曼几何

古希腊大数学家欧几里德是与他的巨著《几何原本》一起名垂千古的。欧几里德把人们公认的一些事实列成定义和公理，以形式逻辑的方法，用这些定义和公理来研究各种几何图形的性质，从而建立了一套从公理、定义出发，论证命题得到定理得几何学论证方法，形成了一个严密的逻辑体系——几何学，形成了欧式几何的奠基之作。

在欧式几何中，代表作是“几何原本”中，有如下公理：

1. 过相异两点，能作且只能作一直线。
2. 线段 (有限直线) 可以任意地延长。
3. 以任一点为圆心、任意长为半径，可作一圆。
4. 凡是直角都相等。
5. 两直线被第三条直线所截，如果同侧两内角和小于两个直角，则两直线则会在该侧相交。这条公理等价于：过直线之外一点有唯一的一条直线和已知直线平行。

在此公理上衍生出整个欧式几何。

俄国的尼古拉·伊万诺维奇·罗巴切夫斯基将欧式几何中的第五条公理代替为“双曲平行公理”，即“过直线之外的一点至少有两条直线和已知直线平行”，并在此公理基础上，推理出一套自治的集合体系，我们称为罗巴切夫斯基几何 (Lobachevskian geometry)，罗巴切夫斯基几何从发表之处一直得不到认同，甚至得到当时学术权威的打压和嘲讽。

1856 年 2 月 12 日，伟大的学者罗巴切夫斯基在苦闷和抑郁中走完了他生命的最后一段路程。喀山大学师生为他举行了隆重的追悼会。在追悼会上，他的许多同事和学生高度赞扬他在建设喀山大学、提高民族教育水平和培养数学人材等方面的卓越功绩，可是谁也不提他的非欧几何研究工作，因为此时，人们还普遍认为非欧几何纯属“无稽之谈”。

---

<sup>6</sup>康威生命游戏是如何搭建计算机的？，链接<https://zhuanlan.zhihu.com/p/144162012>

罗巴切夫斯基为非欧几何的生存和发展奋斗了三十多年，他从来没有动摇过对新几何远大前途的坚定信念。为了扩大非欧几何的影响，争取早日取得学术界的承认，除了用俄文外，他还用法文、德文发行了自己的著作，同时还精心设计了检验大尺度空间几何特性的天文观测方案。

不仅如此，他还发展了非欧几何的解析和微分部分，使之成为一个完整的、有系统的理论体系。在身患重病，卧床不起的困境下，他也没停止对非欧几何的研究。他的最后一部巨著《论几何学》，就是在他双目失明，临去世的前一年，口授他的学生完成的。<sup>7</sup>

直到 1868 年，意大利数学家贝特拉米发表了一篇著名论文《非欧几何解释的尝试》，证明非欧几何可以在欧氏空间的曲面上实现。这就是说，非欧几何命题可以“翻译”成相应的欧氏几何命题，如果欧氏几何没有矛盾，非欧几何也就自然没有矛盾。直到此时，长期无人问津的非欧几何才开始获得学术界的普遍注意和深入研究，罗巴切夫斯基的独创性研究也由此得到学术界的高度评价和一致赞美，这时的罗巴切夫斯基则被人们赞誉为“几何学中的哥白尼”。

黎曼几何是德国数学家黎曼创立的，他在 1851 年所作的一篇论文《论几何学作为基础的假设》中明确的提出另一种几何学的存在，开创了几何学的一片新的广阔领域。黎曼几何是对欧式几何第五条公理修改为：在同一平面内任何两条直线都有公共点(交点)。在黎曼几何学中不承认平行线的存在，它的另一条公设讲：直线可以无限延长，但总的长度是有限的。黎曼几何的模型是一个经过适当“改进”的球面。1845 年，黎曼在哥廷根大学发表了题为《论作为几何基础的假设》的就职演讲，标志着黎曼几何的诞生。黎曼把这三种几何统一起来，统称为黎曼几何，并用这一工作，在哥廷根大学的数学系作报告，谋求一个讲师的位置。后经 E. B. Christoffel, L. Bianchi 及 C. G. Ricci 等人进一步完善和拓广，成为 A. Einstein 创立广义相对论(1915 年)的有力数学工具。此后黎曼几何得到了蓬勃发展，特别是 E. Cartan，他建立的外微分形式和活动标架法，沟通了 Lie 群与黎曼几何的联系，为黎曼几何的深入发展开辟了广阔的前景，影响极为深远。近半个世纪来，黎曼几何的研究从局部发展到整体，产生了许多深刻的并在其他数学分支(如代数拓扑学，偏微分方程，多复变函数论等)及现代物理学中有重要作用的结果。

总之，非欧几何的出现，为天体研究、现代物理、航天航空提供适用的数学工具，极大推进了相关研究工作的进展。

---

<sup>7</sup>摘自 <https://baike.baidu.com/item/>