



离散数学

Discrete Mathematics

极夜酱

目录

1	逻辑	1
1.1	命题	1
1.2	复合命题	5
1.3	逻辑等价	8
1.4	谓词与量词	11
1.5	证明	14
1.6	布尔代数	15
1.7	逻辑门电路	19
2	集合	20
2.1	集合	20
2.2	集合运算	23
2.3	集合恒等式	25
2.4	笛卡尔积	28
3	函数	30
3.1	函数	30
3.2	取整函数	32
3.3	函数分类	33
3.4	反函数	35
3.5	合成函数	36
3.6	指数函数与对数函数	37
4	数论	40
4.1	进制转换	40
4.2	素数	43
4.3	序列	45
4.4	递推关系	47
4.5	求和	50

4.6	数学归纳法	51
5	计数原理	53
5.1	计数原理	53
5.2	排列	54
5.3	组合	57
5.4	古典概型	58

Chapter 1 逻辑

1.1 命题

1.1.1 命题 (Proposition)

逻辑 (logic) 规则给出数学语句的准确含义，这些规则用来区分有效和无效的数学论证。逻辑不仅对理解数学推理十分重要，而且在计算机科学中有许多应用，逻辑可用于电路设计、程序构造、程序正确性证明等方面。

命题是逻辑的基本成分，一个命题是一个具有真值 (truth value) 的语句，命题可以为真也可以为假，但不能既为真又为假。

命题	非命题
I have a dog.	What day is today?
$1 + 2 = 3$	Shut the door!
Today is Wednesday.	$1 + 2$
It is snowing today.	$x + 1 = 2$

表 1.1: 命题与非命题

命题习惯上用字母 p, q, r, s 等来表示，如果一个命题是真命题，它的真值为真，用 T 表示；如果一个命题是假命题，它的真值为假，用 F 表示。

1.1.2 非运算符 (NOT, Negation Operator)

非运算符 \neg 只作用于一个命题，其作用是反转命题的真值。

真值表 (truth table) 可以给出命题真值之间的关系，在确定由简单命题组成的命题的真值时，真值表特别有用。

p	$\neg p$
T	F
F	T

表 1.2: NOT 真值表

Exercise $\neg p$

p : It snowed last night.

$\neg p$: It didn;t snow last night.

q : $2 + 3 = 6$

$\neg q$: $2 + 3 \neq 6$

1.1.3 合取运算符 (AND, Conjunction Operator)

命题 $p \wedge q$ 表示 p 并且 q , 当 p 和 q 都为真时命题为真, 否则为假。

p	$\neg p$	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

表 1.3: AND 真值表

Exercise $p \wedge q$

p : 今天是星期五。

q : 今天会下雨。

$p \wedge q$: 今天是星期五并且会下雨。

1.1.4 析取运算符 (OR, Disjunction Operator)

命题 $p \vee q$ 表示 p 或 q , 当 p 和 q 都为假时命题为假, 否则为真。

p	$\neg p$	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

表 1.4: OR 真值表

Exercise $p \vee q$

p : 开关坏了。

q : 灯泡坏了。

$p \vee q$: 开关坏了或者灯泡坏了。

1.1.5 异或运算符 (XOR, Exclusive Or)

命题 $p \oplus q$ 表示 p 和 q 的异或，当 p 和 q 中恰有一个为真时命题为真，否则为假。

p	$\neg p$	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

表 1.5: XOR 真值表

Exercise $p \oplus q$

p : 他现在在上海。

q : 他现在在北京。

$p \vee q$: 他现在在上海或北京。

Exercise 某地发生了一件谋杀案，警察通过排查确定杀人凶手必为 4 个嫌疑犯的一个，根据以下信息确定凶手。

A 说：不是我。

B 说：是 C。

C 说：是 D。

D 说：C 在胡说。

已知 3 个人说了真话，1 个人说的是假话。

```
1 def main():
2     for killer in ['A', 'B', 'C', 'D']:
3         if (killer != 'A') + (killer == 'C') \
4             + (killer == 'D') + (killer != 'D') == 3:
5             print(killer)
6
7 if __name__ == "__main__":
8     main()
```

运行结果 C

1.2 复合命题

1.2.1 复合命题 (Compound Proposition)

使用非运算符和已定义的各联结词可以构造复合命题。小括号用于规定复合命题中多个逻辑运算符的操作顺序，为了减少所需的小括号数量，规定了各联结词的优先级。

优先级	运算符
1	\neg
2	\wedge / \vee
3	$\rightarrow / \leftrightarrow$

表 1.6: 运算符优先级

1.2.2 蕴含运算符 (Implication Operator)

命题 $p \rightarrow q$ 表示 p 蕴含 q ，在 p 为真而 q 为假时命题为假，否则为真。其中 p 称为前提， q 称为结论。

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

表 1.7: 蕴含真值表

表示 $p \rightarrow q$ 的术语有很多种，常见的有：

- If p , then q .
- p only if q .
- q is necessary for p .

Exercise $p \rightarrow q$

p : 我去看电影。

q : 我买奶茶。

$p \rightarrow q$: 如果我去看电影，那么我会买奶茶。

如果地球是方的，那么猪会飞。



由 $p \rightarrow q$ 可以构造出几个相关的蕴含：

1. $q \rightarrow p$ 称为 $p \rightarrow q$ 的逆命题 (converse)。
2. $\neg q \rightarrow \neg p$ 称为 $p \rightarrow q$ 的逆否命题 (contrapositive)。

Exercise 逆命题与逆否命题

p : 今天是星期四。

q : 我今天有考试。

$p \rightarrow q$: 如果今天是星期四，那么我今天有考试。

$q \rightarrow p$: 如果我今天有考试，那么果今天是星期四。

$\neg q \rightarrow \neg p$: 如果我今天没有考试，那么今天不是星期四。

1.2.3 双向蕴含 (Biconditional Operation)

命题 $p \leftrightarrow q$ 表示 p 双向蕴含 q ，在 p 和 q 有相同的真值时命题为真，否则为假。

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

表 1.8: 双向蕴含真值表

双蕴含的真值表与异或的真值表正好相反，因此 $p \leftrightarrow q$ 与 $\neg(p \oplus q)$ 等价。

1.3 逻辑等价

1.3.1 逻辑等价 (Logical Equivalence)

两个不同的复合命题可能拥有完全相同的真值，则称这两个命题在逻辑上是等价的。如果无论复合命题中各个命题的真值是什么，复合命题的真值总是为真，这样的复合命题称为永真式 (tautology)。如果真值永远为假的复合命题称为矛盾 (contradiction)。

p	$\neg p$	$p \vee \neg p$	$p \wedge \neg p$
T	F	T	F
F	T	T	F

表 1.9: 逻辑等价

如果复合命题 s 和 r 逻辑等价的，可表示为 $s \equiv r$ 。只有当 $s \leftrightarrow r$ 是永真式时， s 和 r 才是逻辑等价的。

Exercise 使用真值表证明 $p \vee q \equiv \neg(\neg p \wedge \neg q)$

p	q	$p \vee q$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$	$\neg(\neg p \wedge \neg q)$
T	T	T	F	F	F	T
T	F	T	F	T	F	T
F	T	T	T	F	F	T
F	F	F	T	T	T	F

1.3.2 逻辑等价定理

幂等律 Idempotent Laws

$$p \wedge p \equiv p \quad (1.1)$$

$$p \vee p \equiv p \quad (1.2)$$

恒等律 Identity Laws

$$p \wedge T \equiv p \quad (1.3)$$

$$p \vee F \equiv p \quad (1.4)$$

支配律 Domination Laws

$$p \vee T \equiv T \quad (1.5)$$

$$p \wedge F \equiv F \quad (1.6)$$

双非律 Double Negation Law

$$\neg(\neg p) \equiv p \quad (1.7)$$

交换律 Commutative Laws

$$p \wedge q \equiv q \wedge p \quad (1.8)$$

$$p \vee q \equiv q \vee p \quad (1.9)$$

结合律 Associative Laws

$$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r) \quad (1.10)$$

$$(p \vee q) \vee r \equiv p \vee (q \vee r) \quad (1.11)$$

分配律 Distributive Laws

$$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r) \quad (1.12)$$

$$(p \vee q) \vee r \equiv p \vee (q \vee r) \quad (1.13)$$

德摩根律 De Morgan's Laws

$$\neg(p \wedge q) \equiv \neg p \vee \neg q \quad (1.14)$$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q \quad (1.15)$$

吸收律 Absorption Laws

$$p \wedge (p \vee q) \equiv p \quad (1.16)$$

$$p \vee (p \wedge q) \equiv p \quad (1.17)$$

条件恒等

$$p \rightarrow q \equiv \neg p \vee q \quad (1.18)$$

$$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p) \quad (1.19)$$

Exercise 证明 $(p \vee q) \rightarrow p$ 永真

$$\begin{aligned} & (p \vee q) \rightarrow p \\ & \equiv \neg(p \wedge q) \vee p \\ & \equiv (\neg p \vee \neg q) \vee p \\ & \equiv (\neg q \vee \neg p) \vee p \\ & \equiv \neg q \vee (\neg p \vee p) \\ & \equiv \neg q \vee T \\ & \equiv T \end{aligned}$$

1.4 谓词与量词

1.4.1 谓词 (Predicate)

命题逻辑并不能表达数学语言和自然语言中所有语句的确切含义。在数学表达式和计算机程序中经常可以看到含有变量的语句，例如 $x > 3$ 、 $x = y + 3$ 、程序 x 正在运行等。当变量值未指定时，这些语句既不为真也不为假。

利用 $P(x)$ 可以表示语句，其中 x 是变量，语句 $P(x)$ 可以说是命题函数 P 在 x 的值。一旦给变量 x 赋一个值，语句 $P(x)$ 就称为命题并具有真值。

通常使用大写字母 P, Q, R 等表示谓词，小写字母 x, y, z 等表示变量。

Exercise 谓词

谓词	真值
$P(x) : x + 3 = 6$	$P(3)$ 为 True
$Q(x, y) : x = y + 2$	$Q(4, 1)$ 为 False

1.4.2 量词 (Quantifier)

量词用量化表示在何种程度上谓词对于一定范围的个体成立。量词有全称量词 (universal quantifier) 和存在量词 (existential quantifier)。

全称量词 \forall 表示 all。 $\forall_x P(x)$ 是一个命题，当范围内所有的 x 都能使语句 $P(x)$ 为真时，命题为真。

$$\forall_x P(x) = P(a_1) \wedge P(a_2) \wedge \cdots \wedge P(a_k)$$

Exercise 全称量词

假设 x 表示全班所有学生， $P(x)$ 表示 x 完成了作业。

$\forall_x P(x)$: 全班所有学生都完成了作业。

存在量词 \exists 表示 exists。 $\exists_x P(x)$ 是一个命题，当范围内存在至少一个 x 能够语句 $P(x)$ 为真时，命题为真。

$$\exists_x P(x) = P(a_1) \vee P(a_2) \vee \cdots \vee P(a_k)$$

Exercise 存在量词

假设 x 表示全班所有学生， $P(x)$ 表示 x 完成了作业。

$\exists_x P(x)$: 班里存在有一个学生完成了作业。

Exercise 嵌套量词

假设 x 表示某个人， $P(x)$ 表示有父母。

$\forall_x P(x)$: 所有人都有父母。

$\exists_x \neg P(x)$: 存在至少有一个人没有父母。

$\exists_x \exists_y (P(x) \wedge P(y))$: 至少存在一个人 x 和一个人 y 有父母。

Exercise $P(x)$: x 是偶数, $Q(x)$: x 能被 3 整除, $x \in \mathbb{Z}^+$

语句	真值
$\exists_x (P(x) \wedge Q(x))$	True
$\forall_x (P(x) \rightarrow \neg Q(x))$	False

1.4.3 全称量词的否定

否定运算符可以使用在全称量词上。

$$\neg \forall_x P(x) \equiv \exists_x \neg P(x) \quad (1.20)$$

$$\neg \exists_x P(x) \equiv \forall_x \neg P(x) \quad (1.21)$$

Exercise 全称量词的否定

$P(x)$: x will pass the course (x is a student).

$\neg \forall x P(x)$: Not all students will pass the course.

$\forall x \neg P(x)$: No student will pass the course.

$\neg \exists x P(x)$: There does not exist a student that will pass the course.

$\exists x \neg P(x)$: There exists a student that will not pass the course.

1.5 证明

1.5.1 证明 (Proof)

证明方法非常重要，不仅因为它们可用于证明数学定理，而且在计算机科学中也有许多应用，包括验证程序正确性、建立安全的操作系统、人工智能领域做推论等。证明就是建立定理真实性的有效论证。

证明定理有很多方法：

1. 直接证明法 (direct proof)

证明 如果 n 是奇数，那么 n^2 也是奇数， $n \in \mathbb{Z}$ 。

$$\begin{aligned}n^2 &= (2k + 1)^2 \\&= 4k^2 + 4k + 1 \\&= 2(2k^2 + 2k) + 1\end{aligned}$$

2. 反证法 (proof by contrapositive): 由于 $p \rightarrow q \equiv \neg q \rightarrow \neg p$ ，因此可以通过证明原命题的逆否命题来反证原命题。

证明 如果 xy 是偶数，那么 x 是偶数或 y 是偶数， $x, y \in \mathbb{Z}$ 。

逆否命题：如果 x 是奇数并且 y 是奇数，那么 xy 是奇数， $x, y \in \mathbb{Z}$ 。

$$\begin{aligned}xy &= (2m + 1)(2n + 1) \\&= 4mn + 2m + 2n + 1 \\&= 2(2mn + m + n) + 1\end{aligned}$$

1.6 布尔代数

1.6.1 布尔代数 (Boolean Algebra)

计算机和其它电子设备中的电路都有输入和输出，输入是 0 或 1，输出也是 0 或 1。电路可以用任何具有两个不同状态的基本元件来构造，例如开关和光学装置就是这样的原件，开关可位于开或关的位置，光学装置可能是点亮或未点亮。18 世纪，乔治·布尔 (George Boole) 给出了逻辑的基本规则。

电路的操作可以用布尔函数来定义，这样的布尔函数对任意一组输入都能指出其输出的值。

布尔代数提供的是集合 $\{0, 1\}$ 上的运算和规则，布尔代数的规则类似于命题逻辑的规则。1 相当于逻辑中的真，0 相当于逻辑中的假。

布尔代码运算主要有三种：

1. 补 (complement)

x	\bar{x}
1	0
0	1

2. 布尔积 (boolean multiplication)

x	y	$x \cdot y$
1	1	1
1	0	0
0	1	0
0	0	0

3. 布尔和 (boolean addition)

x	y	$x + y$
1	1	1
1	0	1
0	1	1
0	0	0

Exercise 当 $x = y = 1$, $w = z = 0$ 时,

$$x \cdot y + (w + z) = 1 \cdot 1 + (0 + 0) = 1 + 0 = 1$$

$$x \cdot \bar{y} + z \cdot \overline{(w + z)} = 1 \cdot \bar{1} + 0 \cdot \overline{(0 + 0)} = 0 + 0 = 0$$

$$x \cdot \bar{y} + \overline{(x + y + \bar{y}z)} = 1 \cdot \bar{1} + \overline{(\bar{1} + 1 + \bar{1} \cdot 0)} = 0 + \bar{1} = 0$$

1.6.2 布尔代数定理

幂等律 Idempotent Laws

$$x \cdot x = x \quad (1.22)$$

$$x + x = x \quad (1.23)$$

恒等律 Identity Laws

$$x \cdot 1 = x \quad (1.24)$$

$$x + 0 = x \quad (1.25)$$

支配律 Domination Laws

$$x \cdot 0 = 0 \quad (1.26)$$

$$x + 1 = 1 \quad (1.27)$$

双非律 Double Negation Law

$$\overline{\overline{x}} = x \quad (1.28)$$

交换律 Commutative Laws

$$x \cdot y = y \cdot x \quad (1.29)$$

$$x + y = y + x \quad (1.30)$$

结合律 Associative Laws

$$(x \cdot y) \cdot z = x \cdot (y \cdot z) \quad (1.31)$$

$$(x + y) + z = x + (y + z) \quad (1.32)$$

分配律 Distributive Laws

$$x \cdot (y + z) = x \cdot y + x \cdot z \quad (1.33)$$

$$x + (y \cdot z) = (x + y) \cdot (x + z) \quad (1.34)$$

德摩根律 De Morgan's Laws

$$\overline{x \cdot y} = \overline{x} + \overline{y} \quad (1.35)$$

$$\overline{x + y} = \overline{x} \cdot \overline{y} \quad (1.36)$$

吸收律 Absorption Laws

$$x \cdot (x + y) = x \quad (1.37)$$

$$x + (x \cdot y) = x \quad (1.38)$$

证明 $xy + x\bar{y} = x$

$$xy + x\bar{y} \quad (1.39)$$

$$= x \cdot (y + \bar{y}) \quad (1.40)$$

$$= x \cdot 1 \quad (1.41)$$

$$= x \quad (1.42)$$

1.6.3 布尔函数 (Boolean Function)

含有 n 个变量的布尔函数能够构造出 2^n 行的输入输出表。

Exercise 计算 $F(x, y, z) = xy + \bar{z}$

x	y	z	$F(x, y, z)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

1.6.4 NAND 与 NOR

NAND 运算符用 \uparrow 表示 Not And:

$$x \uparrow y = \overline{x \cdot y}$$

NOR 运算符用 \downarrow 表示 Not Or:

$$x \downarrow y = \overline{x + y}$$

1.7 逻辑门电路

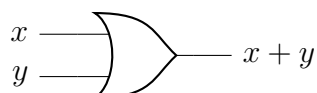
1.7.1 逻辑门电路 (Logical Gate Circuit)

基础的逻辑门主要有三种：

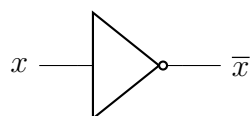
1. 与门 (AND gate)



2. 或门 (OR gate):

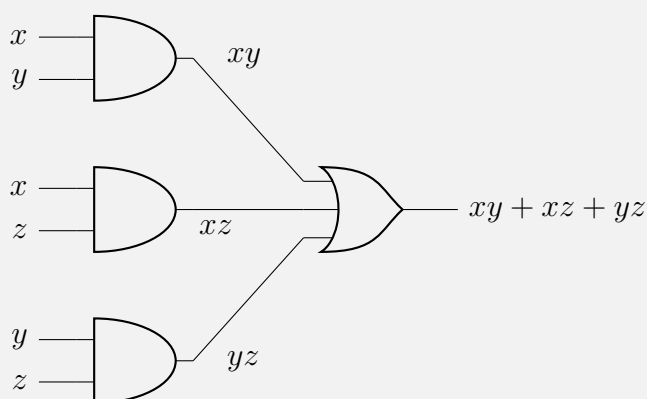


3. 非门 (NOT gate):



Exercise 设计一个投票表决电路，三个人中有两人赞成即通过。赞成票为 1，否决票为 0。

$$F(x, y, z) = xy + xz + yz$$



Chapter 2 集合

2.1 集合

2.1.1 集合 (Set)

集合是对象的唯一的、无序的聚集，通常一个集合中的对象都具有相似的性质。对象也称为集合的元素 (element) 或成员 (member)。

通常用大写字母表示集合，小写字母表示元素。 $a \in A$ 表示是 a 集合 A 中的元素， $a \notin A$ 表示 a 不是集合 A 中的元素。

使用花名册方法 (roster method) 列出集合中的元素，可以用于描述集合。

Exercise 花名册方法

小写元音字母集合 $V = \{a, e, i, o, u\}$

小于 10 的正奇数集合 $O = \{1, 3, 5, 7, 9\}$

小于 100 的非负整数集合 $A = \{0, 1, 2, 3, \dots, 99\}$

集合构造器 (set builder) 通过描述元素具有的形式来描述集合。

Exercise 集合构造器

小于 10 的正整数 $A = \{x \mid x < 10\}$

一些常用的特殊符号可用于描述指定的集合：

符号	含义
\mathbb{N}	自然数集 $\{0, 1, 2, 3, \dots\}$
\mathbb{Z}	整数集 $\{\dots, -2, -1, 0, 1, 2, \dots\}$
\mathbb{Z}^+	正整数集 $\{1, 2, 3, \dots\}$
\mathbb{Q}	有理数集 $\{\frac{p}{q} \mid p \in \mathbb{Z}, q \in \mathbb{Z} (q \neq 0)\}$
\mathbb{Q}^+	正有理数集
\mathbb{R}	实数集
\mathbb{R}^+	正实数集
\mathbb{C}	复数集
\emptyset	空集 $\{\}$

2.1.2 基数 (Cardinality)

基数表示有限集合中元素的个数，集合 A 的基数记为 $|A|$ 。

Exercise 基数

英语字母集合 A , $|A| = 26$

空集 \emptyset , $|\emptyset| = 0$

2.1.3 韦恩图 (Venn Diagram)

集合还可以使用韦恩图来表示。

全集 (universal set) 包含所研究问题中所有的元素，用符号 U 表示。假设 A 是一个集合，由全集 U 中所有不属于 A 的元素组成的集合，称为 A 的补集，表示为 \bar{A} 。

假设有两个集合 A 和 B ，如果 A 中的所有元素都在 B 中，那么 A 就是 B 的子集，表示为 $A \subseteq B$ 。如果 A 中有一个元素不在 B 中，那么 A 就不是 B 的子集，表示为 $A \not\subseteq B$ 。只有当两个集合互相为对方的子集时，那么这两个集合相等，即：

$$A = B \text{ iff } A \subseteq B \text{ and } B \subseteq A$$

如果 $A \subseteq B$ ，并且 B 中有一个元素不是 A 的元素，那么称 A 是 B 的真子集

(proper subset), 表示为 $A \subset B$ 。

2.1.4 幂集 (Power Set)

一个集合中是可以包含另一个集合的, 如 $\{\{1\}, \{1, 2\}, \{1, 2, 3\}\}$ 。需要注意, $1 \neq \{1\} \neq \{\{1\}\}$ 。

幂集用于表示一个集合所有子集的集合, 集合 A 的幂集表示为 $P(A)$ 。

Exercise 计算 $A = \{1, 2, 3\}$ 的幂集

$$P(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

如果集合 A 的基数为 n , 那么 A 的幂集的基数为 2^n , 即 $|P(A)| = 2^n$ 。

2.2 集合运算

2.2.1 交集 (Intersection)

假设 A 和 B 是两个集合，由所有属于 A 并且属于 B 的元素所组成的集合，称为 A 与 B 的交集，表示为 $A \cap B$ 。

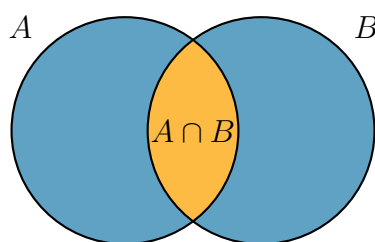


图 2.1: 交集

如果两个集合没有公共元素，那么它们的交集为空集。

2.2.2 并集 (Union)

假设 A 和 B 是两个集合，由它们所有元素合并在一起组成的集合，称为 A 与 B 的并集，表示为 $A \cup B$ 。

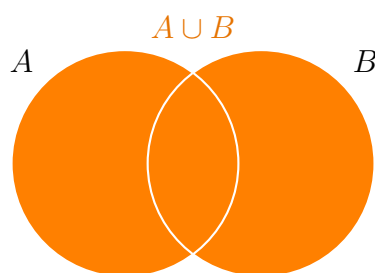


图 2.2: 并集

2.2.3 差集 (Difference)

假设 A 和 B 是两个集合，由属于 A 而不属于 B 的元素组成的集合，称为 A 与 B 的差集，表示为 $A - B$ 。

差集运算不满足交换律，即 $A - B \neq B - A$ 。

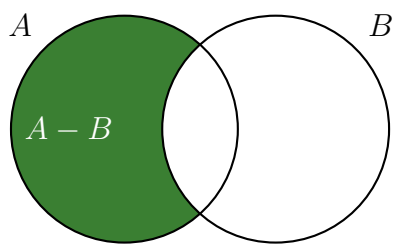


图 2.3: 差集 $A - B$

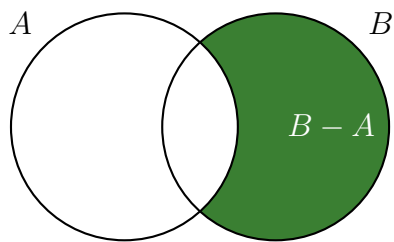


图 2.4: 差集 $B - A$

2.3 集合恒等式

2.3.1 集合恒等式

集合恒等式可以直接由对应的逻辑等价式证明。

幂等律 Idempotent Laws

$$A \cap A = A \quad (2.1)$$

$$A \cup A = A \quad (2.2)$$

恒等律 Identity Laws

$$A \cap U = A \quad (2.3)$$

$$A \cup \emptyset = A \quad (2.4)$$

支配律 Domination Laws

$$A \cap \emptyset = \emptyset \quad (2.5)$$

$$A \cup U = U \quad (2.6)$$

双非律 Double Negation Law

$$\overline{\overline{A}} = A \quad (2.7)$$

交换律 Commutative Laws

$$A \cap B = B \cap A \quad (2.8)$$

$$A \cup B = B \cup A \quad (2.9)$$

结合律 Associative Laws

$$(A \cap B) \cap C = A \cap (B \cap C) \quad (2.10)$$

$$(A \cup B) \cup C = A \cup (B \cup C) \quad (2.11)$$

分配律 Distributive Laws

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \quad (2.12)$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \quad (2.13)$$

德摩根律 De Morgan's Laws

$$\overline{A \cap B} = \bar{A} \cup \bar{B} \quad (2.14)$$

吸收律 Absorption Laws

$$A \cap (A \cup B) = A \quad (2.15)$$

$$A \cup (A \cap B) = A \quad (2.16)$$

证明 $\overline{A \cup (B \cap C)} = (\bar{C} \cup \bar{B}) \cap \bar{A}$

$$\begin{aligned} & \overline{A \cup (B \cap C)} \\ &= \bar{A} \cap \overline{B \cap C} \\ &= \bar{A} \cap (\bar{B} \cup \bar{C}) \\ &= (\bar{B} \cup \bar{C}) \cap \bar{A} \\ &= (\bar{C} \cup \bar{B}) \cap \bar{A} \end{aligned}$$

Exercise 一共有 40 个学生，有 3 门课程可供学生选择（C 语言、离散数学、软件工程）。

7 人没有选任何课程；

16 人选软件工程；

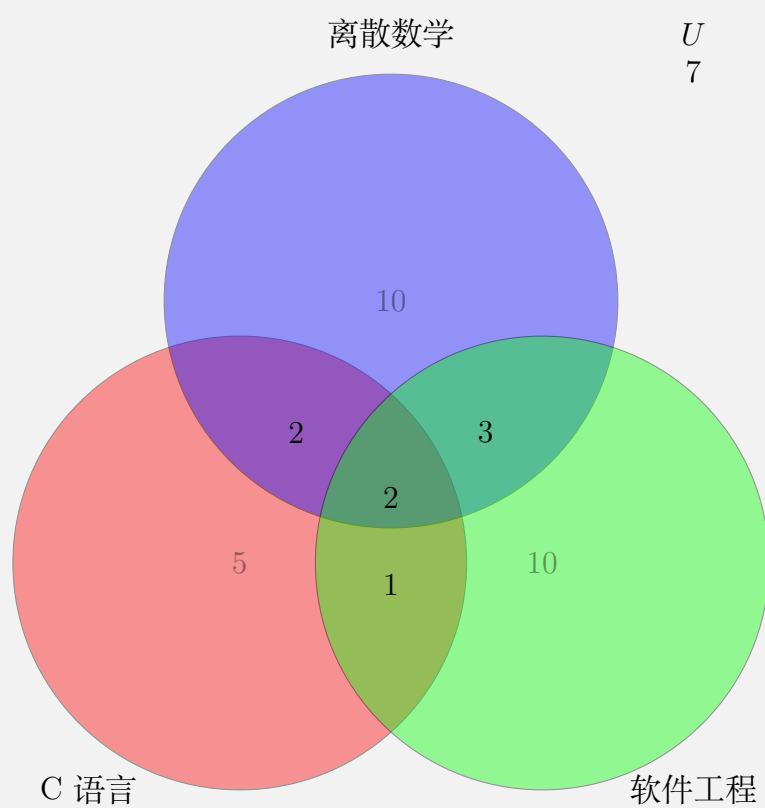
10 人选 C 语言；

5 人同时选离散数学和软件工程；

4 人同时选离散数学和 C 语言；

3 人同时选软件工程和 C 语言；

2 人同时选离散数学、软件工程和 C 语言。



2.4 笛卡尔积

2.4.1 元组 (Tuple)

有时候元素聚集中次序是很重要的，由于集合是无序的，所以就需要同一种不同的结构表示有序的聚集，这就是有序 n 元组 (ordered- n -tuple)。

有序 n 元组 (a_1, a_2, \dots, a_n) 是以 a_1 为第 1 个元素， a_2 为第 2 个元素， a_n 为第 n 个元素的有序聚集。

只有两个有序 n 元组的每一对对应的元素都相等，那么这两个有序 n 元组是相等的，即：

$$(a_1, a_2, \dots, a_n) = (b_1, b_2, \dots, b_n) \text{ iff } i = 1, 2, \dots, n$$

需要注意， (a, b) 与 (b, a) 不相等，除非 $a = b$ 。

2.4.2 笛卡尔积 (Cartesian Product)

假设有两个集合 A 和 B ， A 和 B 的笛卡尔积用 $A \times B$ 表示，笛卡尔积是所有序偶 (a, b) 的集合，其中 $a \in A$ 且 $b \in B$ 。

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

笛卡尔积 $A \times B$ 和 $B \times A$ 是不相等的，除非 $A = \emptyset$ 或 $B = \emptyset$ 或 $A = B$ 。

Exercise 笛卡尔积

学生集合 $S = \{s1, s2\}$

课程集合 $C = \{c1, c2, c3\}$

$S \times C = \{(s1, c1), (s1, c2), (s1, c3), (s2, c1), (s2, c2), (s2, c3)\}$

笛卡尔积 $S \times C$ 表示学生选课的所有可能情况

Exercise 笛卡尔积 $A \times B \times C$

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

$$B = \{1, 2\}$$

$$C = \{0, 1, 2\}$$

$$A \times B \times C = \{(0, 1, 0), (0, 1, 1), (0, 1, 2), (0, 2, 0), (0, 2, 1), (0, 2, 2), \\ (1, 1, 0), (1, 1, 1), (1, 1, 2), (1, 2, 0), (1, 2, 1), (1, 2, 2)\}$$

一个集合与自身的笛卡尔积，如 $A \times A$ 可表示为 A^2 。

Exercise 笛卡尔积 A^2

$$A = \{1, 2\}$$

$$A^2 = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$$

Chapter 3 函数

3.1 函数

3.1.1 函数 (Function)

函数在数学和计算机科学中的概念非常重要，在离散数学中函数用于定义像序列和字符串这样的离散结构。

利用一个函数 f ，可以将一个值 $x \in \mathbb{R}$ 映射 (mapping) 到一个特定的值 $y = f(x)$, $y \in \mathbb{R}$ 上。

假设有两个非空集合 X 和 Y ，从 X 到 Y 的函数 f 是指对于 X 的每个元素恰好都对应 Y 的一个元素，即 $f(x) = y$, $x \in X$, $y \in Y$ ，那么就写成 $f: X \rightarrow Y$ 。

集合 X 被称为函数 f 的定义域 (domain)，集合 Y 被称为函数 f 的陪域 (co-domain)。

如果 $f(x) = y$ ，那么 y 是 x 在函数 f 下的像 (image)， x 是 y 在函数 f 下的原像 (pre-image)。函数 f 的值域 (range) 是集合 X 中所有像的集合。

当两个函数 f 和 g 有相同的定义域和陪域，并且对于定义域中所有元素 x 都满足 $f(x) = g(x)$ ，那么函数 f 和 g 相等，表示为 $f = g$ 。

Exercise 判断是否为函数

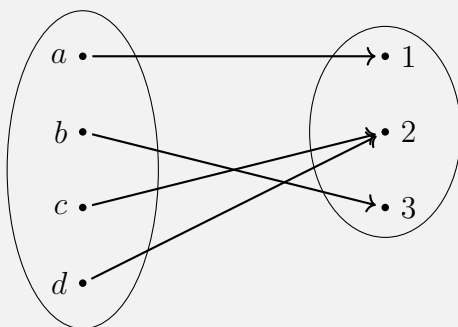


图 3.1: 函数

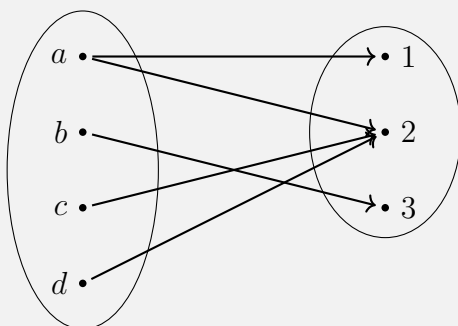
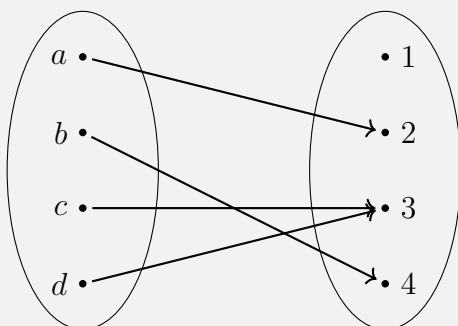


图 3.2: 非函数

Exercise



是否为函数: 是

定义域 (domain): a, b, c, d

陪域 (co-domain): $1, 2, 3, 4$

值域 (range): $2, 3, 4$

3.2 取整函数

3.2.1 上取整函数 (Ceiling Function)

取整函数包括上取整和下取整，可以将实数映射到整数 ($\mathbb{R} \rightarrow \mathbb{Z}$)，它们以不同的方式将实数近似到相邻的整数。

上取整函数将实数 x 向上取到大于或等于 x 的最小整数，表示为 $\lceil x \rceil$ 。

Exercise 上取整函数

$$\lceil 3.2 \rceil = 4$$

$$\lceil 2.6 \rceil = 3$$

$$\lceil -0.5 \rceil = 0$$

3.2.2 下取整函数 (Floor Function)

下取整函数将实数 x 向下取到小于或等于 x 的最大整数，表示为 $\lfloor x \rfloor$ 。

Exercise 下取整函数

$$\lfloor 3.2 \rfloor = 3$$

$$\lfloor 5.9 \rfloor = 5$$

$$\lfloor -0.5 \rfloor = -1$$

3.3 函数分类

3.3.1 一对一函数 (One-to-one) / 单射函数 (Injection)

一对一函数 / 单射函数是指对于函数 f 的定义域中所有的 a 和 b ，如果 $a \neq b$ ，那么 $f(a) \neq f(b)$ 。

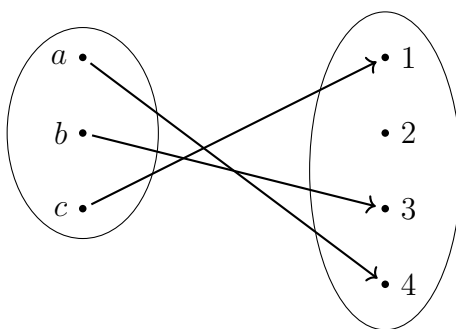


图 3.3: 一对一函数 / 单射函数

Exercise 一对一函数 / 单射函数

$f(x) = x + 1$ 是一对一函数。

$f(x) = x^2$ 不是一对一函数，因为 $f(1) = f(-1) = 1$ 。

3.3.2 映上函数 (Onto) / 满射函数 (Surjection)

映上函数 / 满射函数是指对于函数 $f: A \rightarrow B$ ，每个 $b \in B$ 都有元素 $a \in A$ 使得 $f(a) = b$ 。

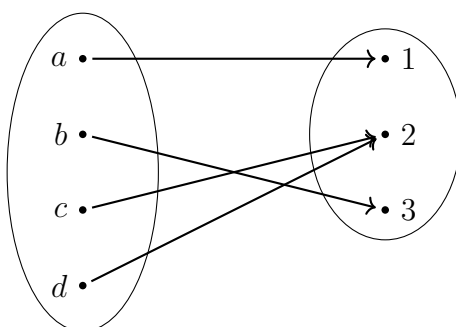


图 3.4: 映上函数 / 满射函数

Exercise 映上函数 / 满射函数

$f: \mathbb{Z} \rightarrow \mathbb{Z}, f(x) = x + 1$ 是映上函数。

$f: \mathbb{Z} \rightarrow \mathbb{Z}, f(x) = x^2$ 不是映上函数，因为没有整数 x 使 $x^2 = -1$ 。

3.3.3 一一对应函数 / 双射函数 (Bijection)

如果一个函数既是一对一函数又是映上函数，那么这个函数就被称为一一对应函数 / 双射函数。

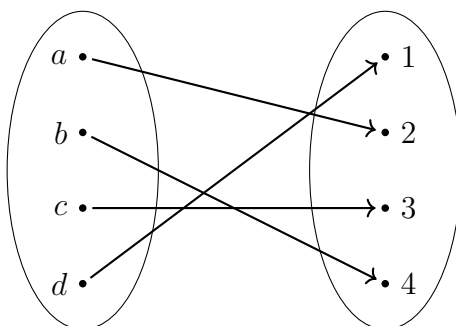


图 3.5: 一一对应函数 / 双射函数

Exercise 一一对应函数 / 双射函数

f 是从 $\{a, b, c, d\}$ 到 $\{1, 2, 3, 4\}$ 的函数，定义 $f(a) = 4, f(b) = 2, f(c) = 1, f(d) = 3$ 。

函数 f 是单射函数，因为没有两个值映射到相同的函数值。

函数 f 是满射函数，因为陪域的个数与值域的个数相同。

因此，函数 f 是双射函数。

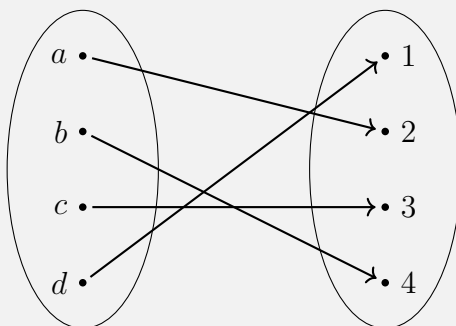
3.4 反函数

3.4.1 反函数 (Inverse Function)

假设有一个从集合 A 到集合 B 的双射函数 f 。由于 f 是满射函数，所以 B 中的每个元素都是 A 中某些元素的像；又由于 f 还是单射函数，所以 B 的每个元素都是 A 中唯一一个元素的像。

于是，通过把 f 的对应关系颠倒，获得的从 B 到 A 的新函数被称为 f 的反函数，用 f^{-1} 表示。当 $f(a) = b$ 时， $f^{-1}(b) = a$ 。需要注意，不要将 f^{-1} 与 $\frac{1}{f}$ 混淆。

Exercise 反函数



是否有反函数：是

$$f^{-1}(2) = a$$

$$f^{-1}(1) = d$$

Exercise 计算 $f(x) = x + 3$ 的反函数

$$f^{-1}(x) = x - 3$$

3.5 合成函数

3.5.1 合成函数 (Composition Function)

假设 g 是从集合 A 到集合 B 的函数, f 是从集合 B 到集合 C 的函数。函数 f 和 g 的合成, 记作 $f \circ g$ 。

$$(f \circ g)(x) = f(g(x))$$

函数合成的顺序很重要, $f \circ g$ 与 $g \circ f$ 并不相等。

Exercise 合成函数

$$f: \mathbb{R}^+ \rightarrow \mathbb{R}^+, f(x) = x^3$$

$$g: \mathbb{R}^+ \rightarrow \mathbb{R}^+, g(x) = x + 2$$

$$(f \circ g)(x) = f(g(x)) = (x + 2)^3$$

$$(g \circ f)(x) = g(f(x)) = x^3 + 2$$

3.5.2 恒等函数 (Identity Function)

如果一个从集合 A 到集合 B 的函数 f 有反函数, 那么 f 与 f^{-1} 的合成函数得到的是恒等函数。

如果 $f(a) = b$, 那么 $f^{-1}(b) = a$ 。

$$(f \circ f^{-1})(a) = f^{-1}(f(a)) = f^{-1}(b) = a$$

3.6 指数函数与对数函数

3.6.1 指数函数 (Exponential Function)

指数函数的定义为 $y = a^x$ ($a > 0 \mid a \neq 1$), 其中 a 称为底数 (base), x 称为指数 (exponent)。

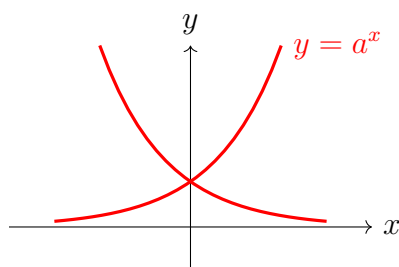


图 3.6: 指数函数

指数函数

$$a^{-x} = \frac{1}{a^x} \quad (3.1)$$

$$\frac{1}{a^{-x}} = a^x \quad (3.2)$$

$$(ab)^x = a^x b^x \quad (3.3)$$

$$\left(\frac{a}{b}\right)^x = \frac{a^x}{b^x} \quad (3.4)$$

$$a^{kx} = (a^k)^x = (a^x)^k \quad (3.5)$$

$$a^m a^n = a^{m+n} \quad (3.6)$$

$$\frac{a^m}{a^n} = a^{m-n} \quad (3.7)$$

$$a^{1/n} = \sqrt[n]{a} \quad (3.8)$$

$$a^{m/n} = \sqrt[n]{a^m} = (\sqrt[n]{a})^m \quad (3.9)$$

Exercise 指数函数

$$(6^{2k})^3 = 6^{6k}$$

$$6^{k^2} \times 6 = 6^{k^2+1}$$

$$\frac{3^k}{9} = \frac{3^k}{3^2} = 3^{k-2}$$

$$3^k \times 27 = 3^k \times 3^3 = 3^{k+3}$$

3.6.2 对数函数 (Logarithm Function)

对于函数 $f: \{1, 2, 3, 4\} \rightarrow \{1, 4, 8, 16\}$, $f(x) = 2^x$, 指数函数是双射函数, 因此它是有反函数的。

对数函数是指数函数的反函数, 对数函数的定义为 $y = \log_a x$ ($a > 0 \mid a \neq 1$), 其中 a 称为底数。

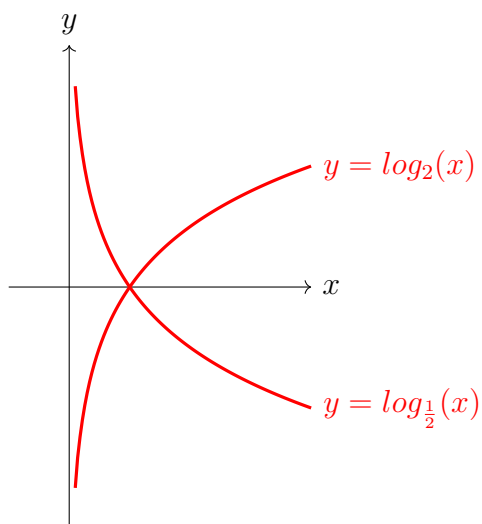


图 3.7: 对数函数

对数函数

$$\log_a(a^x) = x \quad (3.10)$$

$$a^{\log_a(x)} = x \quad (3.11)$$

$$\log_a(xy) = \log_a(x) + \log_a(y) \quad (3.12)$$

$$\log_a\left(\frac{x}{y}\right) = \log_a(x) - \log_a(y) \quad (3.13)$$

$$\log_a(x^n) = n\log_a(x) \quad (3.14)$$

$$\log_a(x) = \frac{\log_b(x)}{\log_b(a)} \quad (3.15)$$

Exercise 对数函数

$$\log_5 k + \log_5 2 = \log_5 2k$$

$$\log_2 5^2 = 2 \times \log_2 5$$

$$\frac{\log_3 k^2}{\log_3 25} = \frac{2 \times \log_3 k}{\log_3 5^2} = \frac{2 \times \log_3 k}{2 \times \log_3 5} = \log_5 k$$

Chapter 4 数论

4.1 进制转换

4.1.1 进制

日常生活中都用十进制(decimal)来表示整数,十进制数由 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 这 10 个字符组成。一个十进制整数的第 k 位的值可以由 10^{k-1} 计算得到。

Exercise 十进制

$$\begin{aligned} 256 &= 200 + 50 + 6 \\ &= 2 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 \end{aligned}$$

二进制 (binary)、八进制 (octal)、十六进制 (hexadecimal) 也是非常常用的表示法,例如计算机通常用二进制来做算术运算,而用八进制或十六进制来表示字符。

4.1.2 进制转换

一个 b 进制的正整数 n 可以唯一地构造展开式:

$$n = a^k \times b^k + a_{k-1} \times b^{k-1} + \cdots + a_1 \times b^1 + a_0 \times b^0$$

Exercise b 进制转十进制

$$(1011)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = (11)_{10}$$

$$(21022)_3 = 2 \times 3^4 + 1 \times 3^3 + 0 \times 3^2 + 2 \times 3^1 + 2 \times 3^0 = (197)_{10}$$

十进制	二进制	八进制	十六进制
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

表 4.1: 进制转换

十进制转 b 进制还可以使用短除法的方式。

Exercise 十进制转四进制

$4 \overline{) 637}$ remainder 1

$4 \overline{) 159}$ remainder 3

$4 \overline{) 39}$ remainder 3 $(637)_{10} = (21331)_4$

$4 \overline{) 9}$ remainder 1

$4 \overline{) 2}$ remainder 2

Exercise 十进制转二进制 $2 \overline{) 637}$ remainder 1 $2 \overline{) 318}$ remainder 0 $2 \overline{) 159}$ remainder 1 $2 \overline{) 79}$ remainder 1 $2 \overline{) 39}$ remainder 1 $2 \overline{) 19}$ remainder 1 $2 \overline{) 9}$ remainder 1 $2 \overline{) 4}$ remainder 0 $2 \overline{) 2}$ remainder 0 $2 \overline{) 1}$ remainder 1

$$(637)_{10} = (1001111101)_2$$

Exercise 十进制转八进制 $8 \overline{) 1000}$ remainder 0 $8 \overline{) 125}$ remainder 5 $8 \overline{) 15}$ remainder 7 $8 \overline{) 1}$ remainder 1

$$(1000)_{10} = (1750)_8$$

4.2 素数

4.2.1 素数 (Prime Numbers)

基于整除性的一个重要概念就是素数，素数是大于 1 的且不能被 1 和它自身以外的正整数整除的整数。素数是现代密码学中必不可少的一部分，密码学中的大素数就用在信息加密的某些方法中。

Exercise 判断素数

7 是素数，因子有 1 和 7。

9 是合数，因为 9 能被 3 整除。

每个大于 1 的整数都可以唯一地写成多个素数的乘积。

Exercise 素因子分解

$$100 = 2 \times 2 \times 5 \times 5 = 2^2 \times 5^2$$

$$999 = 3 \times 3 \times 3 \times 37 = 3^3 \times 37$$

$$1024 = 2^{10}$$

如果 n 是一个合数，那么 n 必有一个素因子小于或等于 \sqrt{n} 。

Exercise 证明 101 是素数

不超过 $\sqrt{101}$ 的素数只有 2, 3, 5, 7，因为 101 不能被 2, 3, 5, 7 整除，所以 101 是素数。

```
1 import math
2
3 def is_prime(num):
4     for i in range(2, int(math.sqrt(num)) + 1):
5         if num % i == 0:
6             return False
7     return True
8
9 def main():
```

```

10     print(is_prime(13))
11     print(is_prime(18))
12
13 if __name__ == "__main__":
14     main()

```

埃拉托斯特尼筛法 (Sieve of Eratosthenes) 可以用来寻找不超过一个给定整数的所有素数。

步骤:

1. 建立包含所有给定整数以内的表格
2. 从 $i = 2$ 开始
3. 移除所有整数 $n\%i == 0$ (除 i 以外)
4. $i = i + 1$
5. 重复第 3 步和第 4 步

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30

表 4.2: 埃拉托斯特尼筛法

4.3 序列

4.3.1 序列 (Sequence)

序列是一种用来表示有序列表的离散结构。例如 1, 2, 3, 5, 8 是一个含有五项的序列, 而 1, 3, 9, 27, 81, ... 是一个无穷序列。序列可以用记号 $\{a_n\}$ 表示。

- 递增序列 (increasing sequence): 一个序列任意相邻的两项满足 $a_k < a_{k+1}$
- 非递减序列 (non-decreasing sequence): 一个序列任意相邻的两项满足 $a_k \leq a_{k+1}$
- 递减序列 (decreasing sequence): 一个序列任意相邻的两项满足 $a_k > a_{k+1}$
- 非递增序列 (non-increasing sequence): 一个序列任意相邻的两项满足 $a_k \geq a_{k+1}$

Exercise 序列

$\{a_n\}$, $a_n = \frac{1}{n}$: 1, $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, ...

$\{b_n\}$, $b_n = 2^n$: 1, 2, 4, 8, 16, 32, ...

4.3.2 算术级数 (Arithmetic Sequence)

算术级数也称等差级数, 序列形式如下:

$$a, a + d, a + 2d, \dots, a + nd, \dots \quad (a, d \in \mathbb{R})$$

Exercise 算术级数

$\{a_n\}$, $a_n = -1 + 4n$: -1, 3, 7, 11, ...

$\{b_n\}$, $b_n = 7 - 3n$: 7, 4, 1, -2, ...

4.3.3 几何级数 (Geometric Sequence)

几何级数也称等比级数, 序列形式如下:

$$a, ar, ar^2, \dots, ar^n, \dots, \quad (a, r \in \mathbb{R})$$

Exercise 几何级数

$$\{a_n\}, a_n = (-1)^n : 1, -1, 1, -1, 1, \dots$$

$$\{b_n\}, b_n = -2 \times 5^n : 2, 10, 50, 250, 1250, \dots$$

$$\{c_n\}, c_n = 6 \times \left(\frac{1}{3}\right)^n : 6, 2, \frac{2}{3}, \frac{2}{9}, \frac{2}{27}, \dots$$

4.4 递推关系

4.4.1 递推 (Recurrence)

如果数列 $\{a_n\}$ 的第 n 项与它前一项的关系可以用一个公式来表示，那么这个公式就叫做这个数列的递推方程。

算术级数的递推关系：

$$a_0 = a$$

$$a_n = a_{n-1} + d$$

几何级数的递推关系：

$$a_0 = a$$

$$a_n = a_{n-1} \times r$$

Exercise 银行储蓄账户上有 10000 元，年利率为 5.8%，7 年后账户中将有多少钱？

$$\begin{aligned} P_n &= P_{n-1} + 0.058P_{n-1} \\ &= (1.058)P_{n-1} \end{aligned}$$

$$P_0 = 10000$$

$$P_1 = (1.058)P_0$$

$$P_2 = (1.058)P_1 = (1.058)^2 P_0$$

...

$$P_7 = (1.058)P_6 = (1.058)^7 P_0 \approx 14838.83$$

4.4.2 斐波那契数列 (Fibonacci Sequence)

斐波那契数列 f_0, f_1, f_2, \dots 的递推公式为：

$$f(n) = \begin{cases} 1 & n = 1 \\ 1 & n = 2 \\ f(n-1) + f(n-2) & n > 3 \end{cases}$$

斐波那契数列的通项公式为：

$$f_n = \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^{n+1} - \frac{1}{\sqrt{5}} \left(\frac{1 - \sqrt{5}}{2} \right)^{n+1}$$

斐波那契数列（递归）

```

1 int fibonacci(int n) {
2     if(n == 1 || n == 2) {
3         return 1;
4     }
5     return fibonacci(n-2) + fibonacci(n-1);
6 }

```

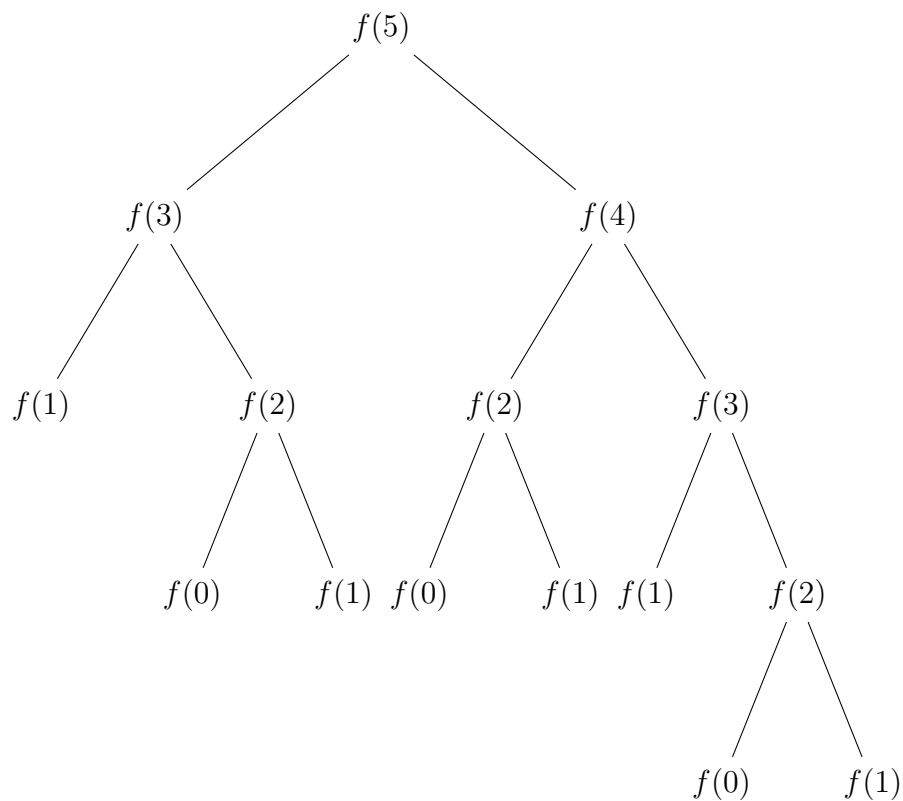


图 4.1: 递归树

斐波那契数列（迭代）

```
1 int fibonacci(int n) {  
2     int f[n];  
3     f[0] = f[1] = 1;  
4     for(int i = 2; i < n; i++) {  
5         f[i] = f[i-2] + f[i-1];  
6     }  
7     return f[n-1];  
8 }
```

4.5 求和

4.5.1 求和 (Summation)

求和符号 \sum 可以用于表示序列中所有项的累加和。

$$\sum_{i=lower}^{upper} a_i$$

Exercise 求和

$$\sum_{i=1}^{100} i = 1 + 2 + 3 + \cdots + 99 + 100 = 5050$$

$$\sum_{j=1}^5 j^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55$$

$$\sum_{k=4}^6 (-1)^k = (-1)^4 + (-1)^5 + (-1)^6 = 1 - 1 + 1 = 1$$

4.5.2 双重求和

很多情况下需要使用双重求和，比如在计算机程序中嵌套循环的分析中。

计算双重求和的方法是先展开内层求和，再继续计算外层求和。

Exercise 双重求和

$$\begin{aligned} & \sum_{i=1}^4 \sum_{j=1}^3 ij \\ &= \sum_{i=1}^4 (i + 2i + 3i) \\ &= \sum_{i=1}^4 6i \\ &= 6 + 12 + 18 + 24 \\ &= 60 \end{aligned}$$

4.6 数学归纳法

4.6.1 数学归纳法 (Mathematical Induction)

数学归纳法是一种数学证明方法，通常被用于证明某个给定命题在一个给定范围内成立。

数学归纳法分为三个步骤：

1. 归纳基础
2. 归纳假设
3. 归纳递推

证明 $\sum_{i=1}^n i = \frac{n(n+1)}{2}, n \in \mathbb{Z}^+$

1. 归纳基础：当 $n = 1$,

$$\sum_{i=1}^1 i = \frac{1(1+1)}{2} = 1$$

2. 归纳假设：假设 $n = k$,

$$\sum_{i=1}^k i = \frac{k(k+1)}{2} \text{ 成立}$$

3. 归纳递推：证明 $n = k + 1$ 时,

$$\begin{aligned}\sum_{i=1}^{k+1} i &= \sum_{i=1}^k i + k + 1 \\&= \frac{k(k+1)}{2} + k + 1 \\&= \frac{k(k+1) + 2(k+1)}{2} \\&= \frac{(k+1)(k+2)}{2}\end{aligned}$$

证明 $2^n \geq 3n, n \geq 4$

1. 归纳基础：当 $n = 4$,
 $2^4 \geq 3 \times 4$
2. 归纳假设：假设 $n = k (n > 4)$,
 $2^k \geq 3k$ 成立
3. 归纳递推：证明 $n = k + 1$ 时,

$$\begin{aligned} 2^{k+1} &= 2 \times 2^k \\ &\geq 2 \times 3k \\ &= 3k + 3k \\ &\geq 3k + 3 \\ &\geq 3(k + 1) \end{aligned}$$

Chapter 5 计数原理

5.1 计数原理

5.1.1 分类加法计数原理

完成一件事有 n 种不同的方案，其中第 1 种方案有 m_1 种不同方法，第 2 种方案有 m_2 种不同方法， \dots ，第 n 种方案有 m_n 种不同方法，那么完成这件事共有 $m_1 + m_2 + \dots + m_n$ 种不同方法。

Exercise 从 A 地到 B 地，可以乘火车、汽车、飞机。火车有 4 班、汽车 2 班、飞机 3 班，那么一天中乘坐这些交通工具从 A 地到 B 地有多少种不同的走法？

$$4 + 2 + 3 = 9$$

5.1.2 分步乘法计数原理

完成一件事需要 n 个步骤，其中第 1 个步骤有 m_1 种不同方法，第 2 个步骤有 m_2 种不同方法， \dots ，第 n 个步骤有 m_n 种不同方法，那么完成这件事共有 $m_1 \times m_2 \times \dots \times m_n$ 种不同方法。

Exercise 一个书架的第 1 层有 4 本不同的计算机书，第 2 层有 3 本不同的经济书，第 3 层有 2 本不同的数学书。从书架的每一层各取一本书，有多少种不同取法？

$$4 \times 3 \times 2 = 24$$

5.2 排列

5.2.1 排列 (Permutation)

从 n 个不同元素中取出 m ($m \leq n$) 个元素, 按照一定次序排成一行, 称为从 n 个不同元素中取出 m 个元素的一个排列。

$$P_n^m = \frac{n!}{(n-m)!} \quad (5.1)$$

例如一共有 8 个人, A、B、C、D、E、F、G、H。现在有 3 个奖杯, 分别为金牌、银牌和铜牌。将这 3 个奖牌颁发给 8 个人中的 3 个, 问颁发奖牌的不同方式总共有几种?

很明显这是一个排列的问题, 因为把金牌先颁给 A, 再把银牌颁给 B, 跟把金牌先颁给 B, 再把银牌颁给 A 这是两种不同的颁奖方式。

- 第一步颁发金牌, 金牌可以颁发给 8 个人中的 1 个, 共有 8 种选择。
- 第二步颁发银牌, 银牌可以颁发剩下 7 个人中的 1 个, 共有 7 种选择。
- 第三步颁发铜牌, 铜牌可以颁发剩下 6 个人中的 1 个, 共有 6 种选择。

那么总共的颁奖方式共有 $8 \times 7 \times 6 = 336$ 种。

Exercise 用 0-9 这 10 个数字可以组成多少个没有重复数字的三位数?

方法一

由于 0 没有排在百位上, 那么百位只能是 1-9 这 9 个数字任选 1 个, 有 P_9^1 种选法。

对于十位和个位, 从余下的 9 个数字种选 2 个, 有 P_9^2 种选法。

$$P_9^1 \times P_9^2 = 9 \times 9 \times 8 = 648$$

方法二

符合条件的三位数可以分为 3 大类:

1. 每一位数字都不是 0 的三位数, 也就是从 1-9 中选 3 个, 有 P_9^3 种选法。
2. 个位数字为 0, 那么需要从剩下 9 个数字中选 2 个作为十位和百位, 有 P_9^2 种选法。
3. 十位数字为 0, 那么需要从剩下 9 个数字中选 2 个作为个位和百位, 有 P_9^2 种选法。

$$P_9^3 + P_9^2 + P_9^2 = 648$$

方法三

利用容斥法。从 0-9 这 10 个数字任取 3 个数字的排列数为 P_{10}^3 , 其中 0 在百位上 (也就是从 1-9 中选 2 个作为十位和个位) 的排列数是 P_9^2 。

$$P_{10}^3 - P_9^2 = 648$$

Exercise 打印字符串 math 的全排列。

全排列

```
1 #include <stdio.h>
2 #include <string.h>
3
```

```
4 void swap(char *a, char *b) {
5     char temp = *a;
6     *a = *b;
7     *b = temp;
8 }
9
10 void permutation(char *s, int start, int end) {
11     if(start >= end) {
12         printf("%s\n", s);
13     } else {
14         for(int i = start; i < end; i++) {
15             swap(s + i, s + start);
16             permutation(s, start + 1, end);
17             swap(s + i, s + start);
18         }
19     }
20 }
21
22 int main() {
23     char s[] = "math";
24     int len = strlen(s);
25     permutation(s, 0, len);
26     return 0;
27 }
```

5.3 组合

5.3.1 组合 (Combination)

从 n 个不同元素中取出 m ($m \leq n$) 个元素，称为从 n 个不同元素中取出 m 个元素的一个组合。

排列与组合的不同点在于，排列与元素的顺序有关，组合与元素的顺序无关。

$$C_n^m = \frac{P_n^m}{P_m^m} = \frac{n!}{(n-m)!m!} \quad (5.2)$$

Exercise 在 100 件产品中，有 98 件合格品，2 件次品，从这 100 件产品中任意抽出 3 件。

(1) 有多少种不同的抽法？

$$C_{100}^3 = \frac{100 \times 99 \times 98}{3 \times 2 \times 1} = 161700$$

(2) 抽出的 3 件中恰好有 1 件事次品的抽法有多少种？

2 件次品抽出 1 件有 C_2^1 种，再从 98 件合格品种抽出 2 件合格品有 C_{98}^2 种。

$$C_2^1 \times C_{98}^2 = 9506$$

(3) 抽出的 3 件中至少有 1 件事次品的抽法有多少种？

方法一

恰好有 1 件次品有 $C_2^1 \times C_{98}^2$ 种，恰好有 2 件次品有 $C_2^2 \times C_{98}^1$ 种。

$$C_2^1 \times C_{98}^2 + C_2^2 \times C_{98}^1 = 9604$$

方法二

利用容斥法。先从 100 件抽出 3 件有 C_{100}^3 种，其中 3 件都是合格品有 C_{98}^3 种。

$$C_{100}^3 - C_{98}^3 = 9604$$

5.4 古典概型

5.4.1 古典概型

如果一个随机试验所包含的单位事件是有限的，且每个单位事件发生的可能性均相等，则这个随机试验叫做拉普拉斯试验，这种条件下的概率模型就叫古典概型。古典概型是概率论中最直观和最简单的模型，概率的许多运算规则，也首先是在这种模型下得到的。

单位事件的特点是两两互斥的，例如抛一枚质地均匀的硬币时，正面朝上和背面朝上不会同时出现。

在古典概型中，概率的计算公式为：

$$P(A) = \frac{A \text{ 包含的单位事件个数 } m}{\text{单位事件的总数 } n}$$

Exercise 掷两个质地均匀的骰子。

(1) 一共有多少种不同的结果？

$$6 \times 6 = 36$$

(2) 点数之和为 9 的结果有多少种？

一共有 4 种：(3, 6), (6, 3), (4, 5), (5, 4)

(3) 点数之和为 9 的概率是多少？

$$P(A) = \frac{4}{36} = \frac{1}{9}$$