DMA 离散数学复习笔记

郑文庭老师班

Rosen, *Discrete Mathematics and Its Applications*, 8th Edition

2026 年 6 月 27 日

目录

第一章 Chapter 1: Logic & Proofs (逻辑与证明) 15

1.1 Chapter 1.1: Propositional Logic (命题逻辑) . . . 15

1.1.1 1. Proposition (命题) . . . 15

1.1.2 2. Logical Connectives (逻辑连接词) . . . 15

1.1.3 3. Truth Tables for Compound Propositions (复合命题真值表) . . . 17

1.1.4 4. Precedence of Logical Operators (运算符优先级) . . . 17

1.1.5 5. Logic and Bit Operations (逻辑与位运算) . . . 17

1.1.6 6. Applications: Translating English to Logic . . . 17

1.1.7 7. System Specifications & Consistency (系统规格与一致性) . . . 17

1.1.8 8. Logic Puzzles (逻辑谜题) . . . 18

1.1.9 9. Muddy Forehead Puzzle (泥额头问题) . . . 18

1.1.10 Summary & Exam Tips (小结与考点提示) . . . 18

1.2 Chapter 1.3: Propositional Equivalences (命题等价式) . . . 19

1.2.1 1. Tautologies, Contradictions, and Contingencies . . . 19

1.2.2 2. Logical Equivalence (逻辑等价) . . . 19

1.2.3 3. De Morgan's Laws (德摩根律) . . . 19

1.2.4 4. Key Logical Equivalences (核心逻辑等价律) . . . 19

1.2.5 5. Equivalence Proofs (等价性证明) . . . 20

1.2.6 6. Dual (对偶式) . . . 20

1.2.7 7. Normal Forms (范式) . . . 20

1.2.8 8. Minterms (极小项) and Full DNF . . . 21

1.2.9 9. Maxterms (极大项) and Full CNF . . . 22

1.2.10 10. Functionally Complete Operators (功能完备集) . . . 22

1.2.11 11. Propositional Satisfiability (可满足性) . . . 22

1.2.12 Summary & Exam Tips (小结与考点提示) . . . 22

1.3 Chapter 1.4: Predicates and Quantifiers (谓词与量词) . . . 22

1.3.1 1. Why Predicate Logic? (为什么需要谓词逻辑) . . . 22

1.3.2 2. Predicates (谓词) and Propositional Functions . . . 22

1.3.3 3. Quantifiers (量词) . . . 23

1.3.4 4. Domain (论域) 的重要性 . . . 23

1.3.5 5. Precedence of Quantifiers (量词优先级) . . . 23

1.3.6 6. Binding Variables (变量绑定) . . . 23

1.3.7 7. Negating Quantifiers (量词的否定) – De Morgan's Laws for Quantifiers . . . 23

1.3.8 8. Logical Equivalences Involving Quantifiers (量词相关的逻辑等价式) . . . 24

1.3.9 9. Translating English to Predicate Logic (英译谓词逻辑) . . . 24

1.3.10 Summary & Exam Tips (小结与考点提示) . . . 24

1.4 Chapter 1.5: Nested Quantifiers (嵌套量词) . . . 25

1.4.1 1. Nested Quantifiers (嵌套量词) . . . 25

1.4.2 2. Thinking of Nested Quantification (类比为嵌套循环) . . . 25

1.4.3 3. Order of Quantifiers (量词顺序) . . . 25

1.4.4 4. Quantifications of Two Variables —Truth Table Summary . . . 25

1.4.5 5. Translating from Nested Quantifiers into English . . . 25

1.4.6 6. Translating English into Nested Quantifiers . . . 26

1.4.7 7. Translating Mathematical Statements . . . 26

1.4.8 8. Negating Nested Quantifiers . . . 26

1.4.9 9. Quantifier Equivalences —Common Questions . . . 27
1.4.10 Summary & Exam Tips (小结与考点提示) . . . 27
1.5 Chapter 1.6: Rules of Inference (推理规则) . . . 27
1.5.1 1. Valid Arguments (有效论证) . . . 27
1.5.2 2. Rules of Inference for Propositional Logic . . . 27
1.5.3 3. Building Valid Arguments (构建有效论证) . . . 29
1.5.4 4. Rules of Inference for Quantified Statements . . . 29
1.5.5 5. Using Rules of Inference with Quantifiers . . . 30
1.5.6 6. Universal Modus Ponens (全称假言推理) . . . 30
1.5.7 Summary & Exam Tips (小结与考点提示) . . . 31
1.6 Chapter 1.7: Introduction to Proofs (证明导论) . . . 31
1.6.1 1. Basic Terminology (基本术语) . . . 31
1.6.2 2. Forms of Theorems . . . 31
1.6.3 3. Proving Conditional Statements $p \to q$ . . . 31
1.6.4 4. Relation between Proof Methods . . . 32
1.6.5 5. Proving Biconditional Statements ($p \leftrightarrow q$) . . . 32
1.6.6 6. What is Wrong with This "Proof"? . . . 32
1.6.7 Summary & Exam Tips (小结与考点提示) . . . 33
1.7 Chapter 1.8: Proof Methods and Strategy (证明方法与策略) . . . 33
1.7.1 1. Proof by Cases (分情形证明法) . . . 33
1.7.2 2. Without Loss of Generality (WLOG, 不失一般性) . . . 33
1.7.3 3. Existence Proofs (存在性证明) . . . 33
1.7.4 4. Counterexamples (反例) . . . 34
1.7.5 5. Nonexistence Proofs (不存在性证明) . . . 34
1.7.6 6. Uniqueness Proofs (唯一性证明) . . . 34
1.7.7 7. Proof Strategies (证明策略) . . . 34
1.7.8 8. Open Problems (开放问题简介) . . . 34
1.7.9 Summary & Exam Tips (小结与考点提示) . . . 34

第二章 Chapter 2: Sets, Functions, Sequences (集合、函数、序列) 36
2.1 Chapter 2.1: Sets (集合) . . . 36
2.1.1 1. Definition of a Set (集合的定义) . . . 36
2.1.2 2. Describing Sets (描述集合) . . . 36
2.1.3 3. Important Sets in Mathematics (重要数集) . . . 36
2.1.4 4. Universal Set and Empty Set (全集与空集) . . . 37
2.1.5 5. Subsets (子集) . . . 37
2.1.6 6. Cardinality of Sets (集合的基数 / 势) . . . 37
2.1.7 7. Power Set (幂集) . . . 37
2.1.8 8. Tuples (元组) . . . 37
2.1.9 9. Cartesian Product (笛卡尔积) . . . 37
2.1.10 10. Truth Sets of Quantifiers (量词的真值集) . . . 38
2.1.11 Summary / 小结 . . . 38
2.2 Chapter 2.2: Set Operations (集合运算) . . . 39
2.2.1 1. Boolean Algebra Connection (布尔代数联系) . . . 39
2.2.2 2. Basic Set Operations (基本集合运算) . . . 39
2.2.3 3. Inclusion-Exclusion Principle (容斥原理) . . . 39
2.2.4 4. Set Identities (集合恒等式) . . . 39
2.2.5 5. Proving Set Identities (证明集合恒等式) . . . 40
2.2.6 6. Generalized Unions and Intersections (广义并集与交集) . . . 41
2.2.7 7. Computer Representation of Sets (集合的计算机表示) . . . 41
2.2.8 Summary / 小结 . . . 41
2.3 Chapter 2.3: Functions (函数) . . . 41
2.3.1 1. Definition of a Function (函数的定义) . . . 41
2.3.2 2. Domain, Codomain, Image, Preimage (定义域、陪域、像、原像) . . . 42
2.3.3 3. Types of Functions (函数的类型) . . . 42

2.3.4 4. Examples and Counterexamples . . . 42
2.3.5 5. Inverse Functions (反函数) . . . 42
2.3.6 6. Function Composition (函数组合) . . . 43
2.3.7 7. Graphs of Functions (函数图像) . . . 43
2.3.8 8. Floor and Ceiling Functions (取整函数) . . . 43
2.3.9 9. Factorial Function (阶乘) . . . 43
2.3.10 10. Partial Functions (偏函数, optional) . . . 44
2.3.11 Summary / 小结 . . . 44
2.4 Chapter 2.4: Sequences and Summations (序列与求和) . . . 44
2.4.1 1. Sequences (序列) . . . 44
2.4.2 2. Geometric Progression (等比数列 / 几何级数) . . . 44
2.4.3 3. Arithmetic Progression (等差数列 / 算术级数) . . . 44
2.4.4 4. Strings (字符串) . . . 45
2.4.5 5. Recurrence Relations (递推关系) . . . 45
2.4.6 6. Fibonacci Sequence (斐波那契数列) . . . 45
2.4.7 7. Solving Recurrence Relations (求解递推关系) . . . 45
2.4.8 8. Financial Application (金融应用) . . . 46
2.4.9 9. Special Integer Sequences (特殊整数序列, optional) . . . 46
2.4.10 10. Summations (求和) . . . 46
2.4.11 11. Geometric Series (几何级数求和) . . . 47
2.4.12 12. Useful Summation Formulae (常用求和公式) . . . 47
2.4.13 Summary / 小结 . . . 47
2.5 Chapter 2.5: Cardinality of Sets (集合的基数 / 势) . . . 47
2.5.1 1. Definition of Cardinality (基数的定义) . . . 47
2.5.2 2. Countable Sets (可数集) . . . 48
2.5.3 3. Hilbert's Grand Hotel (希尔伯特大旅馆) . . . 48
2.5.4 4. Examples of Countable Sets (可数集的例子) . . . 48
2.5.5 5. Properties of Cardinality of Infinite Sets (无限集基数的性质) . . . 48
2.5.6 6. Uncountable Sets: Real Numbers (不可数集：实数) . . . 48
2.5.7 Summary / 小结 . . . 49

第三章 Chapter 3: Algorithms (算法) 50
3.1 Chapter 3.1: Algorithms (算法) . . . 50
3.1.1 1. Definition of an Algorithm (算法的定义) . . . 50
3.1.2 2. Properties of Algorithms (算法的性质) . . . 50
3.1.3 3. Specifying Algorithms (算法的描述方式) . . . 50
3.1.4 4. Example: Finding the Maximum Element (找最大值) . . . 50
3.1.5 5. Searching Problems (搜索问题) . . . 50
3.1.6 6. Sorting Problems (排序问题) . . . 51
3.1.7 7. Greedy Algorithms (贪心算法) . . . 52
3.1.8 8. Halting Problem (停机问题) . . . 52
3.1.9 9. Summary of Important Algorithms (重要算法总结) . . . 53
3.1.10 10. Exam Tips (考点提示) . . . 53
3.2 Chapter 3.2: Growth of Functions (函数的增长) . . . 54
3.2.1 1. Big-O Notation (大 O 记号) . . . 54
3.2.2 2. Examples of Big-O (大 O 的例子) . . . 54
3.2.3 3. Big-O Estimates for Polynomials (多项式的大 O 估计) . . . 54
3.2.4 4. Big-O Estimates for Important Functions (重要函数的大 O 估计) . . . 55
3.2.5 5. Growth Hierarchy (增长阶层次) . . . 55
3.2.6 6. Combinations of Functions (函数的组合) . . . 55
3.2.7 7. Ordering Functions by Growth (按增长排序) . . . 55
3.2.8 8. Big-Omega Notation (Ω 记号) . . . 56
3.2.9 9. Big-Theta Notation (Θ 记号) . . . 56
3.2.10 10. Big-Theta for Polynomials (多项式的大 Theta 估计) . . . 57
3.2.11 11. 三种记号对比总结 . . . 57

3.2.12 12. Exam Tips (考点提示) . . . 57
3.3 Chapter 3.3: Complexity of Algorithms (算法复杂度) . . . 57
3.3.1 1. Introduction (引言) . . . 57
3.3.2 2. Worst-Case vs Average-Case Complexity (最坏情况 vs 平均情况) . . . 57
3.3.3 3. Complexity Analysis Examples (复杂度分析实例) . . . 58
3.3.4 4. Matrix-Chain Multiplication (矩阵链乘法) . . . 60
3.3.5 5. Algorithmic Paradigms (算法范式) . . . 60
3.3.6 6. Understanding Complexity (理解复杂度) . . . 60
3.3.7 7. Complexity of Problems (问题的复杂度分类) . . . 61
3.3.8 8. Summary of Time Complexities (时间复杂度总结) . . . 61
3.3.9 9. Exam Tips (考点提示) . . . 61

第四章 Chapter 4: Number Theory (数论) 63
4.1 Chapter 4.1: Divisibility and Modular Arithmetic (整除性与模运算) . . . 63
4.1.1 1. Divisibility (整除) . . . 63
4.1.2 2. Properties of Divisibility (整除的性质) . . . 63
4.1.3 3. Division Algorithm (除法算法) . . . 63
4.1.4 4. Congruence Relation (同余关系) . . . 64
4.1.5 5. Congruences of Sums and Products (同余的加法和乘法) . . . 64
4.1.6 6. Computing mod of Products and Sums . . . 64
4.1.7 7. Arithmetic Modulo m (模 m 算术) . . . 64
4.1.8 小结与考点提示 . . . 65
4.2 Chapter 4.2: Integer Representations (整数的表示) . . . 66
4.2.1 1. Base b Representations (基 b 展开) . . . 66
4.2.2 2. Binary Expansions (二进制展开) . . . 66
4.2.3 3. Octal Expansions (八进制展开) . . . 66
4.2.4 4. Hexadecimal Expansions (十六进制展开) . . . 66
4.2.5 5. Base Conversion Algorithm (进制转换算法) . . . 66
4.2.6 6. Conversion Between Binary, Octal, and Hexadecimal . . . 67
4.2.7 7. Binary Addition of Integers (二进制加法) . . . 67
4.2.8 8. Binary Multiplication of Integers (二进制乘法) . . . 67
4.2.9 9. Binary Modular Exponentiation (二进制模幂运算) . . . 67
4.2.10 小结与考点提示 . . . 68
4.3 Chapter 4.3: Primes and Greatest Common Divisors (素数与最大公约数) . . . 68
4.3.1 1. Primes (素数/质数) . . . 68
4.3.2 2. Fundamental Theorem of Arithmetic (算术基本定理) . . . 68
4.3.3 3. The Sieve of Eratosthenes (埃拉托斯特尼筛法) . . . 68
4.3.4 4. Infinitude of Primes (素数无穷定理) . . . 69
4.3.5 5. Mersenne Primes (梅森素数) . . . 69
4.3.6 6. Distribution of Primes (素数分布) . . . 69
4.3.7 7. Greatest Common Divisor (GCD, 最大公约数) . . . 69
4.3.8 8. Finding GCD Using Prime Factorizations (利用素因子分解求 GCD) . . . 69
4.3.9 9. Least Common Multiple (LCM, 最小公倍数) . . . 69
4.3.10 10. Euclidean Algorithm (欧几里得算法/辗转相除法) . . . 70
4.3.11 11. Bézout's Theorem (贝祖定理) and gcd as Linear Combinations . . . 70
4.3.12 12. Consequences of Bézout's Theorem . . . 70
4.3.13 13. Dividing Congruences by an Integer . . . 70
4.3.14 小结与考点提示 . . . 70
4.4 Chapter 4.4: Solving Congruences (解同余方程) . . . 71
4.4.1 1. Linear Congruences (线性同余方程) . . . 71
4.4.2 2. Finding Inverses (求逆元) . . . 71
4.4.3 3. Using Inverses to Solve Congruences (用逆元解同余方程) . . . 72
4.4.4 4. Chinese Remainder Theorem (CRT, 中国剩余定理) . . . 72
4.4.5 5. Back Substitution Method (回代法) . . . 73
4.4.6 6. Fermat's Little Theorem (费马小定理) . . . 73

4.4.7 7. Pseudoprimes (伪素数) . . . 73
4.4.8 8. Carmichael Numbers (卡米切尔数) [Optional] . . . 73
4.4.9 9. Primitive Roots (原根) . . . 74
4.4.10 10. Discrete Logarithms (离散对数) . . . 74
4.4.11 小结与考点提示 . . . 74
4.5 Chapter 4.5: Applications of Congruences and Cryptography (同余的应用与密码学) . . . 74
4.5.1 1. Hashing Functions (哈希函数) . . . 74
4.5.2 2. Pseudorandom Numbers (伪随机数) . . . 75
4.5.3 3. Check Digits (校验位) . . . 75
4.5.4 4. Public Key Cryptography (公钥密码学) . . . 75
4.5.5 5. RSA Cryptosystem (RSA 密码系统) . . . 75
4.5.6 6. Cryptographic Protocols: Key Exchange (密钥交换协议) . . . 76
4.5.7 7. Digital Signatures (数字签名) . . . 76
4.5.8 小结与考点提示 . . . 77

第五章 Chapter 5: Induction & Recursion (归纳与递归) 78
5.1 Chapter 5.1: Mathematical Induction (数学归纳法) . . . 78
5.1.1 1. Principle of Mathematical Induction (数学归纳法原理) . . . 78
5.1.2 2. Validity of Mathematical Induction —Well-Ordering Property (良序性质) . . . 78
5.1.3 3. Examples of Proof by Mathematical Induction (例题) . . . 78
5.1.4 4. An Incorrect "Proof" by Mathematical Induction (错误证明示例) . . . 80
5.1.5 5. Guidelines for Proofs by Mathematical Induction (证明指南) . . . 80
5.1.6 6. Summary and Exam Tips (小结与考点提示) . . . 80
5.2 Chapter 5.2: Strong Induction (强归纳法) . . . 82
5.2.1 1. Strong Induction (强归纳法) —Definition . . . 82
5.2.2 2. Strong Induction vs. Ordinary Induction . . . 82
5.2.3 3. Examples of Proof by Strong Induction (例题) . . . 82
5.2.4 4. Well-Ordering Property (良序性质) . . . 83
5.2.5 5. Applications of Well-Ordering Property . . . 83
5.2.6 6. Summary and Exam Tips (小结与考点提示) . . . 84
5.3 Chapter 5.3: Recursive Definitions, Structural Induction & Recursive Algorithms (递归定义、结构归纳法与递归算法) . . . 84
5.3.1 1. Recursively Defined Functions (递归定义的函数) . . . 84
5.3.2 2. Fibonacci Numbers (斐波那契数列) . . . 85
5.3.3 3. Recursively Defined Sets and Structures (递归定义的集合与结构) . . . 85
5.3.4 4. Strings (字符串) . . . 86
5.3.5 5. Balanced Parentheses (平衡括号) . . . 86
5.3.6 6. Well-Formed Formulae in Propositional Logic (命题逻辑的合式公式) . . . 86
5.3.7 7. Rooted Trees (根树) and Full Binary Trees (满二叉树) . . . 86
5.3.8 8. Structural Induction (结构归纳法) . . . 87
5.3.9 9. Generalized Induction (广义归纳法) . . . 87
5.3.10 10. Fractals (分形) . . . 88
5.3.11 11. Euclidean Algorithm (欧几里得算法) and Lame's Theorem (拉梅定理) . . . 88
5.3.12 12. Summary and Exam Tips (小结与考点提示) . . . 88

第六章 Chapter 6: Counting (计数) 90
6.1 Chapter 6.3: Permutations and Combinations (排列与组合) . . . 90
6.1.1 1. Section Overview . . . 90
6.1.2 2. Permutations (排列) . . . 90
6.1.3 3. Combinations (组合) . . . 91
6.1.4 4. Combinatorial Proofs (组合证明) . . . 91
6.1.5 5. Summary Table (总结表) . . . 92
6.1.6 6. 考点提示 (Exam Tips) . . . 92
6.1.7 Homework (from courseware) . . . 92
6.2 Chapter 6.4: Binomial Theorem & Pascal's Identity (二项式定理与帕斯卡恒等式) . . . 93
6.2.1 1. Section Overview . . . 93
6.2.2 2. Binomial Expression (二项式) . . . 93

6.2.3 3. Binomial Theorem (二项式定理) . . . 93
6.2.4 4. Useful Corollaries . . . 93
6.2.5 5. Pascal's Identity (帕斯卡恒等式) . . . 94
6.2.6 6. Pascal's Triangle (杨辉三角 / 帕斯卡三角) . . . 94
6.2.7 7. Vandermonde's Identity (范德蒙恒等式) . . . 94
6.2.8 8. Theorem 4: Another Identity . . . 94
6.2.9 9. 考点提示 (Exam Tips) . . . 95
6.2.10 Homework (from courseware) . . . 95
6.3 Chapter 6.5: Generalized Permutations and Combinations (广义排列与组合) . . . 95
6.3.1 1. Section Overview . . . 95
6.3.2 2. Permutations with Repetition (重复排列) . . . 95
6.3.3 3. Combinations with Repetition (重复组合) . . . 95
6.3.4 4. Summary: Four Types of Counting (四种计数总结) . . . 96
6.3.5 5. Permutations with Indistinguishable Objects (含相同对象的排列) . . . 96
6.3.6 6. Distributing Objects into Boxes (分配问题) . . . 96
6.3.7 7. 考点提示 (Exam Tips) . . . 97
6.3.8 Homework (from courseware) . . . 97
6.4 Chapter 6.6: Generating Permutations and Combinations (生成排列与组合) . . . 98
6.4.1 1. Section Overview . . . 98
6.4.2 2. Generating Permutations (生成排列) . . . 98
6.4.3 3. Generating Combinations (生成组合) . . . 98
6.4.4 4. 考点提示 (Exam Tips) . . . 99
6.4.5 Homework (from courseware) . . . 99

第七章 Chapter 8: Recurrence Relations (递推关系) 100
7.1 Chapter 8.1: Applications of Recurrence Relations . . . 100
7.1.1 1. 基本概念 (Basic Concepts) . . . 100
7.1.2 2. Fibonacci Numbers (斐波那契数列) . . . 100
7.1.3 3. Tower of Hanoi (汉诺塔) . . . 100
7.1.4 4. Counting Bit Strings (计数二进制串) . . . 101
7.1.5 5. Catalan Numbers (卡特兰数) —括号化问题 . . . 101
7.1.6 6. Codeword Enumeration (码字计数) . . . 102
7.1.7 7. 小结和考点提示 (Summary & Exam Tips) . . . 102
7.2 Chapter 8.2: Solving Linear Recurrence Relations . . . 103
7.2.1 1. 基本概念 (Basic Concepts) . . . 103
7.2.2 2. 求解线性齐次递推关系 (Solving Linear Homogeneous RR) . . . 103
7.2.3 3. 一般求解步骤 (General Solution Steps) . . . 103
7.2.4 4. 简单例子 (Simple Example) . . . 103
7.2.5 5. 求解方法概览 (All Solving Methods) . . . 103
7.2.6 6. 解的结构 (Structure of Solutions) . . . 103
7.2.7 7. 小结和考点提示 (Summary & Exam Tips) . . . 103
7.3 Chapter 8.3: Homogeneous Recurrence Relations (齐次递推关系) . . . 104
7.3.1 1. 求解线性齐次递推关系 (Solving Linear Homogeneous RR) . . . 104
7.3.2 2. Degree 2 —两个不同实根 (Theorem 1) . . . 104
7.3.3 3. Fibonacci 数的显式公式 (Explicit Formula for Fibonacci) . . . 105
7.3.4 4. Degree 2 —重根情况 (Theorem 2) . . . 105
7.3.5 5. 任意阶—互异根 (Theorem 3) . . . 105
7.3.6 6. 任意阶—允许重根 (Theorem 4) . . . 106
7.3.7 7. 定理总结 (Summary of Theorems) . . . 106
7.3.8 8. 小结和考点提示 (Summary & Exam Tips) . . . 106
7.4 Chapter 8.4: Nonhomogeneous Recurrence Relations & Divide-and-Conquer . . . 107
7.5 Part A: Nonhomogeneous Recurrence Relations (非齐次递推关系) . . . 107
7.5.1 1. 定义 (Definition) . . . 107
7.5.2 2. 解的结构 (Theorem 5) . . . 107
7.5.3 3. 求解步骤 (Solution Steps) . . . 107

7.5.4 4. 如何找特解 (Finding Particular Solution) —Theorem 6 . . . 108
7.5.5 5. 完整示例 (Complete Examples) . . . 108
7.5.6 6. 联立递推关系 (Simultaneous Recurrence Relations) . . . 109
7.6 Part B: Divide-and-Conquer Algorithms (分治算法) . . . 109
7.6.1 1. 分治算法范式 (Divide-and-Conquer Paradigm) . . . 109
7.6.2 2. 经典例子 (Classic Examples) . . . 109
7.6.3 3. 分治递推的估计 (Theorems) . . . 109
7.6.4 4. Master Theorem 应用示例 . . . 110
7.6.5 5. 小结和考点提示 (Summary & Exam Tips) . . . 110
7.7 Chapter 8.5: Generating Functions (生成函数) . . . 111
7.7.1 1. 定义 (Definition) . . . 111
7.7.2 2. 常见序列的生成函数 (Useful Generating Functions) . . . 111
7.7.3 3. 生成函数的运算性质 (Operations on Generating Functions) . . . 111
7.7.4 4. Extended Binomial Coefficient (广义二项式系数) . . . 112
7.7.5 5. 生成函数在计数中的应用 . . . 112
7.7.6 6. 用生成函数解递推关系 (Solving RR using Generating Functions) . . . 113
7.7.7 7. 用生成函数证明组合恒等式 . . . 113
7.7.8 8. 小结和考点提示 (Summary & Exam Tips) . . . 113
7.8 Chapter 8.6: Inclusion-Exclusion (容斥原理) . . . 114
7.9 Part A: The Principle of Inclusion-Exclusion (8.5) . . . 114
7.9.1 1. 两个集合的情况 (Two Sets) . . . 114
7.9.2 2. 三个集合的情况 (Three Sets) . . . 114
7.9.3 3. 一般形式 (General Form) . . . 114
7.9.4 4. 应用示例 . . . 115
7.10 Part B: Applications of Inclusion-Exclusion (8.6) . . . 115
7.10.1 1. 另一种形式 (Alternative Form) . . . 115
7.10.2 2. 带约束的方程求整数解 . . . 115
7.10.3 3. Eratosthenes 筛法 (Sieve of Eratosthenes) . . . 115
7.10.4 4. 满射函数的计数 (Counting Onto Functions) . . . 116
7.10.5 5. Derangements (错位排列) . . . 116
7.10.6 6. 小结和考点提示 (Summary & Exam Tips) . . . 116

第八章 Chapter 9: Relations (关系) 118
8.1 Chapter 9.1: Relations and Their Properties (关系及其性质) . . . 118
8.1.1 1. Binary Relations (二元关系) . . . 118
8.1.2 2. Properties of Relations (关系的性质) . . . 118
8.1.3 3. Combining Relations (关系的组合) . . . 119
8.1.4 4. Powers of a Relation (关系的幂) . . . 119
8.1.5 5. Inverse Relation (逆关系) . . . 120
8.1.6 6. Properties of Relation Operations (关系运算的性质) . . . 120
8.1.7 7. 小结与考点提示 (Summary & Exam Tips) . . . 120
8.2 Chapter 9.2: n-ary Relations (n 元关系) . . . 121
8.2.1 1. Definition of n-ary Relations (n 元关系的定义) . . . 121
8.2.2 2. Relational Databases (关系数据库) . . . 121
8.2.3 3. Operations on n-ary Relations (n 元关系的运算) . . . 121
8.2.4 4. n-ary Relations in the Course Context . . . 122
8.2.5 5. 小结与考点提示 (Summary & Exam Tips) . . . 122
8.3 Chapter 9.3: Representing Relations (关系的表示) . . . 122
8.3.1 1. Representing Relations Using Matrices (用矩阵表示关系) . . . 122
8.3.2 2. Matrix Properties and Relation Properties (矩阵与关系性质) . . . 123
8.3.3 3. Representing Relations Using Digraphs (用有向图表示关系) . . . 123
8.3.4 4. Properties from Digraphs (从有向图判断性质) . . . 123
8.3.5 5. Powers of a Relation in Digraphs (用图理解关系的幂) . . . 124
8.3.6 6. Inverse Relation and Matrix (逆关系与矩阵) . . . 124
8.3.7 7. 小结与考点提示 (Summary & Exam Tips) . . . 124

8.4 Chapter 9.4: Closures of Relations (关系的闭包) . . . 124
8.4.1 1. Concept of Closure (闭包的概念) . . . 124
8.4.2 2. Reflexive Closure (自反闭包) . . . 124
8.4.3 3. Symmetric Closure (对称闭包) . . . 125
8.4.4 4. Transitive Closure (传递闭包) . . . 125
8.4.5 5. Matrix Method for Transitive Closure . . . 126
8.4.6 6. Warshall's Algorithm (沃舍尔算法) . . . 126
8.4.7 7. Properties of Closure Operations (闭包运算的性质) . . . 127
8.4.8 8. 小结与考点提示 (Summary & Exam Tips) . . . 127
8.5 Chapter 9.5: Equivalence Relations (等价关系) . . . 127
8.5.1 1. Definition of Equivalence Relation (等价关系的定义) . . . 127
8.5.2 2. Examples of Equivalence Relations . . . 128
8.5.3 3. Equivalence Classes (等价类) . . . 128
8.5.4 4. Equivalence Classes and Partitions (等价类与划分) . . . 128
8.5.5 5. Example: Finding Partition from an Equivalence Relation . . . 129
8.5.6 6. Counting Equivalence Relations (计数等价关系) . . . 129
8.5.7 7. Combining Equivalence Relations (等价关系的组合) . . . 129
8.5.8 8. 小结与考点提示 (Summary & Exam Tips) . . . 130
8.6 Chapter 9.6: Partial Orderings (偏序) . . . 130
8.6.1 1. Definition of Partial Order (偏序的定义) . . . 130
8.6.2 2. Examples of Partial Orders . . . 130
8.6.3 3. Comparability (可比性) . . . 130
8.6.4 4. Lexicographic Order (字典序) . . . 131
8.6.5 5. Hasse Diagrams (哈斯图) . . . 131
8.6.6 6. Hasse Diagram Terminology (哈斯图术语) . . . 132
8.6.7 7. Upper and Lower Bounds (上下界) . . . 132
8.6.8 8. Lattices (格) . . . 133
8.6.9 9. Topological Sorting (拓扑排序) . . . 133
8.6.10 10. 小结与考点提示 (Summary & Exam Tips) . . . 134

第九章 Chapter 10: Graphs (图论) 135
9.1 Chapter 10.1: Graphs and Graph Models (图与图模型) . . . 135
9.1.1 1. 什么是图? (What is a Graph?) . . . 135
9.1.2 2. 无向图的类型 (Types of Undirected Graphs) . . . 135
9.1.3 3. 有向图 (Directed Graphs / Digraphs) . . . 135
9.1.4 4. 图模型 (Graph Models) . . . 135
9.1.5 5. 典型例题 . . . 136
9.1.6 小结 (Summary) . . . 136
9.2 Chapter 10.2: Graph Terminology and Special Types of Graphs (图术语与特殊图类) . . . 137
9.2.1 Part 1: 基本术语 (Basic Terminology) . . . 137
9.2.2 Part 2: 特殊简单图 (Special Simple Graphs) . . . 138
9.2.3 Part 3: Bipartite Graphs (二部图) . . . 138
9.2.4 Part 4: Matchings in Bipartite Graphs (二部图中的匹配) . . . 139
9.2.5 Part 5: New Graphs from Old (从旧图构造新图) . . . 140
9.2.6 小结 (Summary) . . . 140
9.3 Chapter 10.3: Representing Graphs and Graph Isomorphism (图的表示与同构) . . . 141
9.3.1 Part 1: 图的表示方法 (Representing Graphs) . . . 141
9.3.2 Part 2: Graph Isomorphism (图的同构) . . . 143
9.3.3 小结 (Summary) . . . 144
9.4 Chapter 10.4: Connectivity (连通性) . . . 145
9.4.1 Part 1: Paths (路径) . . . 145
9.4.2 Part 2: 计算路径数量 (Counting Paths) . . . 145
9.4.3 Part 3: 无向图的连通性 (Connectedness in Undirected Graphs) . . . 145
9.4.4 Part 4: 有向图的连通性 (Connectedness in Directed Graphs) . . . 146
9.4.5 Part 5: Paths and Isomorphism (路径与同构) . . . 147

9.4.6 小结 (Summary) . . . 147
9.5 Chapter 10.5: Euler and Hamilton Paths (欧拉路径与哈密顿路径) . . . 147
9.5.1 Part 1: Euler Paths and Circuits (欧拉路径与回路) . . . 147
9.5.2 Part 2: Hamilton Paths and Circuits (哈密顿路径与回路) . . . 149
9.5.3 小结: Euler vs Hamilton 对比 . . . 151
9.6 Chapter 10.6: Shortest Path Problems (最短路径问题) . . . 151
9.6.1 Part 1: 带权图与最短路径问题 (Weighted Graphs) . . . 151
9.6.2 Part 2: Dijkstra's Algorithm (Dijkstra 算法) . . . 151
9.6.3 Part 3: 补充算法—Floyd's Algorithm . . . 153
9.6.4 Part 4: The Traveling Salesman Problem (TSP, 旅行商问题) . . . 153
9.6.5 小结 (Summary) . . . 154
9.7 Chapter 10.7: Planar Graphs (平面图) . . . 154
9.7.1 Part 1: 平面图的基本概念 (What is a Planar Graph?) . . . 154
9.7.2 Part 2: Euler's Formula (欧拉公式) . . . 155
9.7.3 Part 3: Kuratowski's Theorem (库拉托夫斯基定理) . . . 157
9.7.4 小结 (Summary) . . . 158
9.8 Chapter 10.8: Graph Coloring (图的着色) . . . 158
9.8.1 Part 1: 地图着色问题 (Map Coloring) . . . 158
9.8.2 Part 2: 图着色的基本概念 . . . 159
9.8.3 Part 3: The Four Color Theorem (四色定理) . . . 159
9.8.4 Part 4: 一些简单图的色数 . . . 160
9.8.5 Part 5: 着色的应用 (Applications of Graph Coloring) . . . 161
9.8.6 小结 (Summary) . . . 162

第十章 Chapter 11: Trees (树) 163
10.1 Chapter 11.1: Introduction to Trees (树的基本介绍) . . . 163
10.1.1 1. Definition of a Tree (树的定义) . . . 163
10.1.2 2. Theorem 1: Unique Path Property (唯一路径定理) . . . 163
10.1.3 3. Rooted Trees (有根树) . . . 163
10.1.4 4. Basic Terminology in Rooted Trees (基本术语) . . . 163
10.1.5 5. Running Example: Jake's Pizza Shop Organization Tree . . . 164
10.1.6 6. m-ary Trees and Binary Trees (m 叉树与二叉树) . . . 164
10.1.7 7. Nonisomorphic Trees (非同构树) . . . 164
10.1.8 8. Tree Properties and Theorems (树的性质与定理) . . . 165
10.1.9 9. Every Tree is Bipartite (树是二分图) . . . 166
10.1.10 10. 小结与考点提示 . . . 166
10.2 Chapter 11.2: Applications of Trees (树的应用) . . . 167
10.2.1 Overview (概述) . . . 167
10.2.2 1. Binary Search Trees (二叉搜索树, BST) . . . 167
10.2.3 2. Decision Trees (决策树) . . . 168
10.2.4 3. Prefix Codes (前缀编码) . . . 169
10.2.5 4. 小结与考点提示 . . . 171
10.3 Chapter 11.3: Tree Traversal (树的遍历) . . . 171
10.3.1 Overview (概述) . . . 171
10.3.2 1. Preorder Traversal (前序遍历) . . . 172
10.3.3 2. Inorder Traversal (中序遍历) . . . 172
10.3.4 3. Postorder Traversal (后序遍历) . . . 173
10.3.5 4. Infix, Prefix, and Postfix Notation (中缀、前缀和后缀表达式) . . . 174
10.3.6 5. Evaluating Expression Trees (求值) . . . 175
10.3.7 6. 小结与考点提示 . . . 175
10.4 Chapter 11.4: Spanning Trees (生成树) . . . 176
10.4.1 1. Definition of Spanning Tree (生成树的定义) . . . 176
10.4.2 2. The Problem of Spanning Trees . . . 176
10.4.3 3. Theorem 1: Connected iff Spanning Tree Exists . . . 176
10.4.4 4. Algorithms for Constructing Spanning Trees . . . 177

10.4.5 5. Depth-First Search (DFS，深度优先搜索) . . . 177
10.4.6 6. Breadth-First Search (BFS，广度优先搜索) . . . 177
10.4.7 7. Backtracking Scheme (回溯法) . . . 178
10.4.8 8. DFS in Directed Graphs (有向图的深度优先搜索) . . . 179
10.4.9 9. 小结与考点提示 . . . 179
10.5 Chapter 11.5: Minimum Spanning Trees (最小生成树) . . . 180
10.5.1 1. The Problem (问题描述) . . . 180
10.5.2 2. Definition (定义) . . . 180
10.5.3 3. Greedy Algorithms (贪心算法) . . . 180
10.5.4 4. Prim's Algorithm (普里姆算法) . . . 180
10.5.5 5. Kruskal's Algorithm (克鲁斯卡尔算法) . . . 182
10.5.6 6. Prim vs Kruskal 对比 . . . 183
10.5.7 7. 小结与考点提示 . . . 184

第十一章 Network Flow (网络流) 185

11.1 Network Flow 网络流 . . . 185
11.1.1 1. Network Flow Definitions (网络流定义) . . . 185
11.1.2 2. Maximum Flow Problem (最大流问题) . . . 185
11.1.3 3. Cuts in a Graph (图中的割) . . . 185
11.1.4 4. Flows and Cuts —Key Lemmas (流与割的关键引理) . . . 186
11.1.5 5. Residual Graph (残差图) . . . 186
11.1.6 6. Augmenting Path (增广路径) . . . 187
11.1.7 7. Ford-Fulkerson Algorithm (福特-福克森算法) . . . 187
11.1.8 8. Max-Flow Min-Cut Theorem (最大流最小割定理) . . . 187
11.1.9 9. Performance and Improvements (性能与改进) . . . 188
11.1.10 10. Application –Bipartite Matching (应用——二分图匹配) . . . 189
11.1.11 11. Summary and Exam Tips (小结与考点提示) . . . 189

第一部分 历年真题考点分析 191

11.2 DMA 历年真题考点分析—Ch1-3 (郑文庭班) . . . 192
11.2.1 Ch1 Logic & Proofs . . . 192
11.2.2 Ch2 Sets, Functions, Sequences . . . 194
11.2.3 Ch3 Algorithms . . . 196
11.2.4 附录：各年份试卷对应题目速查 . . . 197
11.2.5 核心考点统计 . . . 198
11.3 Exam Review: Ch4 (Number Theory & Cryptography) + Ch5 (Induction & Recursion) . . . 199
11.3.1 Ch4: Number Theory & Cryptography (数论与密码学) . . . 199
11.3.2 Ch5: Induction & Recursion (归纳与递归) . . . 200
11.3.3 题型总结与答题技巧 . . . 201
11.3.4 附录：历年 Ch4/Ch5 题目索引 . . . 201
11.4 Ch6 & Ch8 考点笔记 – 郑文庭班历年小测真题整理 . . . 201
11.4.1 目录 . . . 201
11.4.2 Ch6: Counting（计数） . . . 202
11.4.3 Ch8: Recurrence Relations & Advanced Counting . . . 203
11.4.4 考试 Tips . . . 204
11.5 Ch9-Ch11 离散数学历年小测考点整理（郑文庭班） . . . 204
11.5.1 Ch9: Relations（关系） . . . 205
11.5.2 Ch10: Graphs（图） . . . 205
11.5.3 Ch11: Trees（树） . . . 206
11.5.4 综合高频考点一览 . . . 207
11.5.5 特别提醒 . . . 207

第二部分 模拟卷 208

11.6 DMA 离散数学摸底考试卷 . . . 209

11.6.1 Part 1: True/False (30%) . . . 209
11.6.2 Part 2: Short Problems (40%) . . . 209
11.6.3 Part 3: Long Problems (30%) . . . 210
11.6.4 Answer Quick-Check . . . 210
11.7 DMA 摸底卷—详细题解 . . . 211
11.7.1 Part 1: True/False 解析 . . . 211
11.7.2 Part 2: 详解 . . . 211
11.7.3 Part 3: 详解 . . . 212

第三部分 历年真题题解 215

11.8 DMA Quiz 1 题解—2022 年 . . . 216
11.8.1 一、判断题 (True/False) (12%) . . . 216
11.8.2 二、谓词逻辑翻译 (16%) . . . 216
11.8.3 三、命题等价变换 (12%) . . . 217
11.8.4 四、主析取范式与主合取范式 (16%) . . . 217
11.8.5 五、集合分配律的推广 (12%) . . . 217
11.8.6 六、Cantor 对角线法 (12%) . . . 218
11.8.7 七、算法复杂度分析 (8%) . . . 218
11.8.8 八、数学归纳法 (12%) . . . 218
11.9 DMA Quiz 1 题解—2023 年 . . . 219
11.9.1 一、判断题 (True/False) (30%) . . . 219
11.9.2 二、单项选择题 (5%) . . . 219
11.9.3 三、命题等价变换 (10%) . . . 219
11.9.4 四、函数阶的排序 (8%) . . . 220
11.9.5 五、主合取范式与主析取范式 (12%) . . . 220
11.9.6 六、有理数子集的证明 (12%) . . . 220
11.9.7 七、斐波那契数列的模与恒等式 (15%) . . . 221
11.9.8 八、取石子游戏 (8%) . . . 222
11.10 DMA Quiz 1 题解—2024 年 . . . 222
11.10.1 一、判断题 (True/False) (30%) . . . 222
11.10.2 二、谓词逻辑翻译 (10%) . . . 223
11.10.3 三、函数复合 (5%) . . . 223
11.10.4 四、命题等价变换 (7%) . . . 223
11.10.5 五、主析取范式与主合取范式 (14%) . . . 223
11.10.6 六、函数阶的排序 (7%) . . . 224
11.10.7 七、取整函数与集合基数 (10%) . . . 224
11.10.8 八、反证法与有理数 (10%) . . . 225
11.10.9 九、数学归纳法——幂平均不等式 (10%) . . . 225
11.11 DMA Quiz 1 题解—2025 年 . . . 225
11.11.1 一、判断题 (True/False) (35%) . . . 225
11.11.2 二、命题等价变换 (12%) . . . 227
11.11.3 三、主合取范式 (9%) . . . 227
11.11.4 四、函数的枚举与分类 (8%) . . . 227
11.11.5 五、简化剩余系 (9%) . . . 228
11.11.6 六、中国剩余定理 (9%) . . . 228
11.11.7 七、集合分配律的推广 (9%) . . . 228
11.11.8 八、斐波那契表示定理 (9%) . . . 229
11.12 DMA Quiz 2 题解—2022 年 . . . 229
11.12.1 题目 1 . . . 229
11.12.2 题目 2 . . . 230
11.12.3 题目 3 . . . 231
11.12.4 题目 4 . . . 231
11.12.5 题目 5 . . . 232
11.12.6 题目 6 . . . 232

11.12.7 题目 7 . . . 233
11.13 DMA Quiz 2 题解—2023 年 . . . 233
11.13.1 题目 1 (31% total) . . . 233
11.13.2 题目 2 (32% total, 4% each) . . . 235
11.13.3 题目 3 . . . 236
11.13.4 题目 4 . . . 236
11.13.5 题目 5 . . . 237
11.13.6 题目 6 . . . 237
11.14 DMA Quiz 2 题解—2024 年 . . . 238
11.14.1 说明 . . . 238
11.14.2 历年 Quiz 2 考点总结（供参考） . . . 238
11.15 DMA Quiz 3 题解—2022 年 . . . 238
11.15.1 题目 1 . . . 238
11.15.2 题目 2 . . . 239
11.15.3 题目 3 . . . 240
11.15.4 题目 4 . . . 241
11.15.5 题目 5 . . . 241
11.15.6 题目 6 . . . 242
11.15.7 题目 7 . . . 242
11.15.8 题目 8 . . . 243
11.16 DMA Quiz 3 题解—2024 年 . . . 243
11.16.1 题目 1 . . . 243
11.16.2 题目 2 . . . 244
11.16.3 题目 3 . . . 244
11.16.4 题目 4 . . . 245
11.16.5 题目 5 . . . 246
11.16.6 题目 6 . . . 246
11.16.7 题目 7 . . . 246
11.16.8 题目 8 . . . 247
11.16.9 题目 9 . . . 251
11.16.10 题目 10 . . . 252
11.16.11 题目 11 . . . 252
11.17 DMA Quiz 4 题解—2021-2022 春夏 (郑文庭班) . . . 254
11.17.1 题目 1 (18%)：谓词逻辑翻译 . . . 254
11.17.2 题目 2 (12%)：Almost-Palindromes 计数 . . . 254
11.17.3 题目 3 (15%)：递推关系求解 . . . 255
11.17.4 题目 4 (25%)：网格图 $G_{m,n}$ 的性质 . . . 255
11.17.5 题目 5 (20%)：正多面体与图论 . . . 256
11.17.6 题目 6 (10%)：斐波那契数的 Zeckendorf 表示 . . . 257
11.18 DMA Quiz 4 题解—2023-2024 春夏 (郑文庭班) . . . 257
11.18.1 题目 1 (40%，每题 5%)：填空题 . . . 257
11.18.2 题目 2 (6%)：中国剩余定理 . . . 258
11.18.3 题目 3 (30%，每题 5%)：扑克牌组合计数 . . . 259
11.18.4 题目 4 (12%)：生成函数求解递推 . . . 260
11.18.5 题目 5 (12%)：红绿点不相交连线（归纳法） . . . 260
11.19 DMA 期末回忆卷题解—2021-2022 春夏 (郑文庭班) . . . 260
11.19.1 一、判断题 . . . 260
11.19.2 二、填空题 . . . 261
11.19.3 三、关系与图论 . . . 262
11.19.4 四、进阶问题 . . . 262

第四部分　附录 **265**
11.20 DMA 期末复习规划 . . . 266
11.20.1 关键日期 . . . 266

11.20.2 6/22（今天）—摸底定位 . . . 266
11.20.3 6/23 —模拟考日 . . . 266
11.20.4 6/24-26 —真题冲刺（每天 3-4h） . . . 266
11.20.5 6/27 —总复习（约 3h） . . . 266
11.20.6 6/28 上午—考试 . . . 267
11.20.7 每日时间分配建议 . . . 267
11.20.8 章节优先级 . . . 267
11.21 DMA 薄弱点记录 . . . 268
11.21.1 今日消灭计划（6/27） . . . 268
11.21.2 2026-06-22 摸底卷暴露问题 . . . 268
11.21.3 消灭记录 . . . 269

第一章　Chapter 1: Logic & Proofs (逻辑与证明)

1.1　Chapter 1.1: Propositional Logic (命题逻辑)

DMA Review Notes – Bilingual (English/Chinese)

1.1.1　1. Proposition (命题)

Definition: A **proposition** is a declarative sentence that is either **true** or **false** (但不可同时为真且同时为假).

- 命题是一个具有真值的陈述句，非真即假。
- 命题变量 (propositional variables): $p, q, r, s, \ldots$
- $\mathbf{T}$ = True (真), $\mathbf{F}$ = False (假)

Examples of propositions:

- "The Moon is made of green cheese." (假)
- "Hangzhou is the capital of China." (假)
- $1 + 1 = 2$ (真)
- $0 + 0 = 2$ (假)

Not propositions (不是命题——因为没有真值):

- "Sit down!" (祈使句)
- "What time is it?" (疑问句)
- $x + 1 = 2$ (变量未赋值, 无法判断真假)

1.1.2　2. Logical Connectives (逻辑连接词)

名称	符号	英文	中文
Negation	$\neg$	NOT	否定
Conjunction	$\wedge$	AND	合取
Disjunction	$\vee$	OR	析取
Exclusive OR	$\oplus$	XOR	异或
Implication	$\rightarrow$	IF-THEN	蕴含
Biconditional	$\leftrightarrow$	IFF	双条件

2.1 Negation (否定) $\neg p$

- "It is not the case that p."
- 真值表:

p	$\neg p$
T	F
F	T

2.2 Conjunction (合取) $p \wedge q$

- "p AND q" —仅当 p 和 q 都为真时结果为真。
- 真值表:

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

2.3 Disjunction (析取) $p \vee q$

- "p OR q" —只要 p 或 q 至少一个为真即真。
- 真值表:

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

2.4 Exclusive OR (异或) $p \oplus q$

- "p XOR q" —恰好一个为真时为真（不能同时为真）。
- 真值表:

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

注意: 日常英语中的"or" 有两种含义:

- **Inclusive Or (兼或)**: 可同时成立，对应 $\vee$ (如"prerequisite: CS 120 or Math")
- **Exclusive Or (异或)**: 不可同时成立，对应 $\oplus$ (如"Soup or salad")

2.5 Implication (蕴含) $p \rightarrow q$

- **读法**: "if p, then q"
- p: hypothesis (假设/前件/antecedent), q: conclusion (结论/后件/consequent)
- 真值表:

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

关键理解: 只有当 p 真、q 假时，$p \rightarrow q$ 为假。前件为假时，蕴含式自动为真 (vacuous truth)。

多种英文表达方式: - if p, then q - p implies q - q follows from p - p only if q - q whenever p - p is **sufficient** for q (p 是 q 的充分条件) - q is **necessary** for p (q 是 p 的必要条件)

Example: "You can access the Internet from campus only if you are a CS major or you are not a freshman." - a: "you can access the Internet from campus" - c: "you are a CS major" - f: "you are a freshman" - 翻译: $a \rightarrow (c \vee \neg f)$

2.6 Converse, Inverse, Contrapositive (逆命题、反命题、逆否命题)

从 $p \rightarrow q$ 出发:

名称	形式	与原命题的关系
Converse (逆命题)	$q \rightarrow p$	**不等价**
Inverse (反命题)	$\neg p \rightarrow \neg q$	**不等价**
Contrapositive (逆否命题)	$\neg q \rightarrow \neg p$	**等价于原命题**

核心结论: $p \rightarrow q \equiv \neg q \rightarrow \neg p$ (原命题与逆否命题等价)

真值表验证:

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

可见 $p \rightarrow q$ 与 $\neg q \rightarrow \neg p$ 在所有情况下的真值一致，故等价。

Example: 对语句"It raining is a sufficient condition for my not going to town." - p: It is raining, q: I go to town - 原命题: $p \rightarrow \neg q$ (raining $\rightarrow$ not go) - Converse: $\neg q \rightarrow p$ (If I do not go to town, then it is raining) - Inverse: $\neg p \rightarrow q$ (If it is not raining, then I will go to town) - Contrapositive: $q \rightarrow \neg p$ (If I go to town, then it is not raining)

2.7 Biconditional (双条件) $p \leftrightarrow q$

- "p if and only if q" (iff, 当且仅当)
- 真值表:

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

- 等价表述: $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
- 其他英文表达: "p is necessary and sufficient for q", "if p then q, and conversely"

1.1.3 3. Truth Tables for Compound Propositions (复合命题真值表)

构造步骤:

1. **Rows**: 若有 n 个命题变量，则有 2^n 行。
2. **Columns**: 每个子表达式一列，最右列为最终结果。

Example: 构造 $(p \vee \neg q) \to r$ 的真值表

p	q	r	$\neg q$	$p \vee \neg q$	$(p \vee \neg q) \to r$
T	T	T	F	T	T
T	T	F	F	T	F
T	F	T	T	T	T
T	F	F	T	T	F
F	T	T	F	F	T
F	T	F	F	F	T
F	F	T	T	T	T
F	F	F	T	T	F

1.1.4 4. Precedence of Logical Operators (运算符优先级)

Operator	Precedence
$\neg$	1 (最高)
$\wedge$	2
$\vee$	3
$\to$	4
$\leftrightarrow$	5

注意: $p \vee q \to \neg r$ 等价于 $(p \vee q) \to (\neg r)$。若想要其他结合方式，必须使用括号。

1.1.5 5. Logic and Bit Operations (逻辑与位运算)

- **Bit** (比特): 0 (False), 1 (True)
- **Boolean variable**: 取值为 0 或 1 的变量

Bitwise operations (按位运算):

- Bitwise OR ($\vee$)
- Bitwise AND ($\wedge$)
- Bitwise XOR ($\oplus$)

Example: 对 01 1011 0110 和 11 0001 1101 做按位运算:

```
  01 1011 0110
  11 0001 1101
------------------
  11 1011 1111  (bitwise OR)
  01 0001 0100  (bitwise AND)
  10 1010 1011  (bitwise XOR)
```

1.1.6 6. Applications: Translating English to Logic

步骤:

1. 识别原子命题，赋命题变量。
2. 确定合适的逻辑连接词。

Example 1: "If I go to Harry's or to the country, I will not go shopping."

- p: I go to Harry's
- q: I go to the country
- r: I will go shopping
- 翻译: $(p \vee q) \to \neg r$

Example 2: "The automated reply cannot be sent when the file system is full."

- p: "The automated reply can be sent"
- q: "The file system is full"
- 翻译: $q \to \neg p$

1.1.7 7. System Specifications & Consistency (系统规格与一致性)

Consistency: 一个命题集合称为 **consistent** 如果存在一组真值赋值使得所有命题同时为真。**Example**: 判断以下规格是否一致:

1. "The diagnostic message is stored in the buffer or it is retransmitted." $(p \vee q)$
2. "The diagnostic message is not stored in the buffer." $(\neg p)$
3. "If the diagnostic message is stored in the buffer, then it is retransmitted." $(p \to q)$

Solution: 令 $p = F$, $q = T$，三条均为真，故一致。

1.1.8 8. Logic Puzzles (逻辑谜题)

Knights and Knaves 问题:

- **Knights** (骑士): 总是说真话
- **Knaves** (无赖): 总是说谎

Example: A 说"B is a knight." B 说"The two of us are of opposite types."

- p: "A is a knight", q: "B is a knight"
- B 说的话: $(p \wedge \neg q) \vee (\neg p \wedge q)$
- **推理**: 若 A 是骑士，则 q 为真 => B 的话应为真 => $(p \wedge \neg q) \vee (\neg p \wedge q)$ 为真，代入 $p = T, q = T$ 得假 (矛盾) => A 不是骑士 => A 是无赖 => B 不可能为骑士 => 两人均为无赖。

1.1.9 9. Muddy Forehead Puzzle (泥额头问题)

- 父亲说: "At least one of you has a muddy forehead." $(s \vee d)$
- 两个孩子都能看到对方的情况，但看不到自己的。
- **第一次问**: 两人都回答"No" (因为彼此看到对方有泥)
- **第二次问**: 两人都回答"Yes" (通过对方的回答推理出自己的情况)

1.1.10 Summary & Exam Tips (小结与考点提示)

考点	说明
命题定义	必须是陈述句且有真值；疑问句/祈使句/变量未绑定都不是命题
连接词真值表	熟记 6 个连接词的真值表，尤其 $\to$ 和 $\oplus$
蕴含式 $p \to q$	仅当前真后假时为假，其余均真
逆/反/逆否	原命题 $\equiv$ 逆否命题；converse 和 inverse 不等价
翻译 English → Logic	注意"only if" 引导的是必要条件，"unless" = "if not"
真值表行数	n 个变量需 2^n 行
运算符优先级	$\neg > \wedge > \vee > \to > \leftrightarrow$
Consistency 判断	找一组真值赋值使所有命题为真

易错提醒:

- p only if $q \iff p \to q$ (不是 $q \to p$!)
- p unless $q \iff \neg q \to p \iff p \vee q$
- 充分条件 (sufficient) = hypothesis, 必要条件 (necessary) = conclusion

1.2 Chapter 1.3: Propositional Equivalences (命题等价式)

DMA Review Notes – Bilingual (English/Chinese)

1.2.1 1. Tautologies, Contradictions, and Contingencies

类型	定义	示例
Tautology (永真式/重言式)	命题在所有真值赋值下均为真	$p \vee \neg p$
Contradiction (矛盾式)	命题在所有真值赋值下均为假	$p \wedge \neg p$
Contingency (可能式)	既不是永真也不是矛盾	p, $p \vee q$

Truth Table 验证:

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

- $p \vee \neg p$ 全为 T => Tautology
- $p \wedge \neg p$ 全为 F => Contradiction

1.2.2 2. Logical Equivalence (逻辑等价)

Definition: 两个复合命题 p 和 q 称为 **logically equivalent** (逻辑等价) 如果 $p \leftrightarrow q$ 是永真式。记作 $p \equiv q$ 或 $p \Leftrightarrow q$。

判定方法：真值表中 p 和 q 的所有对应行真值均相同。

Core Example: $p \to q \equiv \neg p \vee q$

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

两列结果完全一致，故 $p \to q \equiv \neg p \vee q$。

1.2.3 3. De Morgan's Laws (德摩根律)

两条核心律:

- $\neg(p \wedge q) \equiv \neg p \vee \neg q$
- $\neg(p \vee q) \equiv \neg p \wedge \neg q$

记忆口诀: 否定进入括号内，AND 变 OR，OR 变 AND。

Example: 用 De Morgan's Laws 表达以下句子的否定: - "Mike has a cellphone and he has a laptop computer" - 否定: "Mike does not have a cellphone **or** he does not have a laptop computer." - "Jim will go to the concert or Steve will go to the concert" - 否定: "Jim will **not** go to the concert **and** Steve will **not** go to the concert."

1.2.4 4. Key Logical Equivalences (核心逻辑等价律)

基本等价律

律名	形式
Identity Laws (恒等律)	$p \wedge \mathbf{T} \equiv p$, $p \vee \mathbf{F} \equiv p$
Domination Laws (支配律)	$p \vee \mathbf{T} \equiv \mathbf{T}$, $p \wedge \mathbf{F} \equiv \mathbf{F}$
Idempotent Laws (幂等律)	$p \vee p \equiv p$, $p \wedge p \equiv p$
Double Negation Law (双重否定律)	$\neg(\neg p) \equiv p$
Negation Laws (否定律)	$p \vee \neg p \equiv \mathbf{T}$, $p \wedge \neg p \equiv \mathbf{F}$

交换律、结合律、分配律

律名	形式
Commutative (交换律)	$p \vee q \equiv q \vee p$, $p \wedge q \equiv q \wedge p$
Associative (结合律)	$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
Distributive (分配律)	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$

吸收律与德摩根律

律名	形式
Absorption Laws (吸收律)	$p \vee (p \wedge q) \equiv p$, $p \wedge (p \vee q) \equiv p$
De Morgan's Laws	$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$

蕴含与等价

律名	形式
Implication Law (蕴含等值式)	$p \to q \equiv \neg p \vee q$
Equivalence Law (等价等值式)	$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$ $p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$

1.2.5 5. Equivalence Proofs (等价性证明)

方法一: 真值表 (truth table) —直观但变量多时耗时。**方法二**: 利用已知等价律逐步替换。

证明示例

Example 1: 证明 $\neg(p \vee (\neg p \wedge q)) \equiv \neg p \wedge \neg q$ **Solution**:

$$\begin{aligned}
\neg(p \vee (\neg p \wedge q)) &\equiv \neg p \wedge \neg(\neg p \wedge q) && \text{(De Morgan)} \\
&\equiv \neg p \wedge (p \vee \neg q) && \text{(De Morgan + Double Negation)} \\
&\equiv (\neg p \wedge p) \vee (\neg p \wedge \neg q) && \text{(Distributive)} \\
&\equiv \mathbf{F} \vee (\neg p \wedge \neg q) && \text{(Negation Law)} \\
&\equiv \neg p \wedge \neg q && \text{(Identity Law)}
\end{aligned}$$

Example 2: 证明 $(p \to q) \wedge (p \to r) \equiv p \to (q \wedge r)$ **Solution**:

$$\begin{aligned}
(p \to q) \wedge (p \to r) &\equiv (\neg p \vee q) \wedge (\neg p \vee r) && \text{(Implication Law)} \\
&\equiv \neg p \vee (q \wedge r) && \text{(Distributive)} \\
&\equiv p \to (q \wedge r) && \text{(Implication Law)}
\end{aligned}$$

Example 3: 证明 $[p \wedge (p \to q)] \to q$ 是永真式。**Solution**:

$$\begin{aligned}
[p \wedge (p \to q)] \to q &\equiv \neg[p \wedge (\neg p \vee q)] \vee q \\
&\equiv [\neg p \vee \neg(\neg p \vee q)] \vee q \\
&\equiv [\neg p \vee (p \wedge \neg q)] \vee q \\
&\equiv [(\neg p \vee p) \wedge (\neg p \vee \neg q)] \vee q \\
&\equiv [\mathbf{T} \wedge (\neg p \vee \neg q)] \vee q \\
&\equiv (\neg p \vee \neg q) \vee q \\
&\equiv \neg p \vee (\neg q \vee q) \\
&\equiv \neg p \vee \mathbf{T} \equiv \mathbf{T}
\end{aligned}$$

Non-Equivalence Example (非等价性证明)

Example 4: 证明 $(p \to (q \to r))$ 和 $((p \to q) \to r)$ **不**逻辑等价。**Solution**: 只需找到一组真值赋值使两式真值不同即可。取 $p = F$, $q = T$, $r = F$:

- $p \to (q \to r) \equiv F \to (T \to F) \equiv F \to F \equiv T$
- $(p \to q) \to r \equiv (F \to T) \to F \equiv T \to F \equiv F$

两式真值不同，故非等价。可见蕴含连接词 $\to$ 不可随意重新结合。

1.2.6 6. Dual (对偶式)

Definition: 一个只含 $\vee$, $\wedge$, $\neg$ 的复合命题的 **dual** 是将 $\vee$ 换为 $\wedge$、$\wedge$ 换为 $\vee$、**T** 换为 **F**、**F** 换为 **T** 得到的式子。记作 S^*. **Example**: $S = (p \vee \neg q) \wedge r \vee \mathbf{T} \implies S^* = (p \wedge \neg q) \vee r \wedge \mathbf{F}$ **Theorem**: 若 $s \equiv t$，则 $s^* \equiv t^*$.

1.2.7 7. Normal Forms (范式)

7.1 Literal (文字) 与 Clause (子句)

- **Literal**: 一个命题变量或其否定。如 p、$\neg p$。
- **Disjunctive clause** (析取子句): 文字的析取，如 $\neg p \vee q \vee \neg r$。
- **Conjunctive clause** (合取子句): 文字的合取，如 $p \wedge q \wedge \neg r$。

7.2 DNF - Disjunctive Normal Form (析取范式)

Definition: 合取子句的析取。形如:

$$(A_{11} \wedge \cdots \wedge A_{1n}) \vee \cdots \vee (A_{k1} \wedge \cdots \wedge A_{kn_k})$$

Examples:

- $(p \wedge q) \vee (p \wedge \neg q)$ —是 DNF
- $p \vee (q \wedge r)$ —是 DNF
- $\neg(p \wedge q) \vee r$ —**不是** DNF (括号在否定范围内)

7.3 CNF - Conjunctive Normal Form (合取范式)

Definition: 析取子句的合取。形如:

$$(A_{11} \vee \cdots \vee A_{1n}) \wedge \cdots \wedge (A_{k1} \vee \cdots \vee A_{kn_k})$$

Examples:

- $p \wedge (q \vee r)$ —是 CNF
- $\neg p \wedge (q \vee \neg r) \wedge (\neg q \vee r)$ —是 CNF
- $p \wedge (r \vee (p \wedge q))$ —**不是** CNF

7.4 转换为 CNF/DNF 的步骤

1. **消去** $\rightarrow, \leftrightarrow, |, \downarrow$:
- $p \rightarrow q \equiv \neg p \vee q$
- $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
1. **将否定向内移动** (利用 De Morgan 和双重否定):
- $\neg(p \wedge q) \equiv \neg p \vee \neg q$
- $\neg(p \vee q) \equiv \neg p \wedge \neg q$
- $\neg\neg p \equiv p$
1. **利用分配律**:
- 得到 CNF: $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
- 得到 DNF: $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$

Example: 将 $\neg((p \vee \neg q) \wedge \neg r)$ 转换为 CNF **Solution**:

$$\begin{aligned}
\neg((p \vee \neg q) \wedge \neg r) &\equiv \neg(p \vee \neg q) \vee \neg\neg r && \text{(De Morgan)} \\
&\equiv \neg(p \vee \neg q) \vee r && \text{(Double Negation)} \\
&\equiv (\neg p \wedge \neg\neg q) \vee r && \text{(De Morgan)} \\
&\equiv (\neg p \wedge q) \vee r && \text{(Double Negation)} \\
&\equiv (\neg p \vee r) \wedge (q \vee r) && \text{(Distributive)} \quad \text{(CNF)}
\end{aligned}$$

1.2.8 8. Minterms (极小项) and Full DNF

Definition

Minterm: 每个变量恰好出现一次（或其否定出现一次）的合取式。对于 n 个变量，共有 2^n 个不同的 minterm。**Example** ($n = 3$, 变量 p, q, r):

- $m_0 = \neg p \wedge \neg q \wedge \neg r$
- $m_1 = \neg p \wedge \neg q \wedge r$
- $m_2 = \neg p \wedge q \wedge \neg r$
- $m_3 = \neg p \wedge q \wedge r$
- $m_4 = p \wedge \neg q \wedge \neg r$
- $m_5 = p \wedge \neg q \wedge r$
- $m_6 = p \wedge q \wedge \neg r$
- $m_7 = p \wedge q \wedge r$

Minterm 的性质

1. 每个 minterm 恰好对一个真值赋值为真。
2. 不同 minterm 的合取恒为假 ($m_i \wedge m_k \equiv \mathbf{F}$)。
3. 所有 minterm 的析取为永真式 ($\bigvee_{i=0}^{2^n-1} m_i \equiv \mathbf{T}$)。

从真值表得到 Full DNF

对于真值表结果为真的行，取对应的 minterm 后做析取。**Example**: 给定真值表 f:

p	q	r	f
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	T
F	F	F	F

f 为真的行: (T, T, T), (T, F, T), (F, F, T)

$$f \equiv (p \wedge q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge \neg q \wedge r)$$

简写: $f = \sum m(1, 5, 7)$

从 DNF 转换为 Full DNF

对每个缺少变量的合取子句，使用 $(x \vee \neg x)$ 补全。

1.2.9 9. Maxterms (极大项) and Full CNF

- **Maxterm**: 每个变量恰好出现一次的析取式，记作 M_i。
- 若 $f = \sum m(\text{indexes})$，则 $\neg f$ 由不在该集合中的 minterm 组成。
- 利用 $f = \neg(\neg f)$ 可得 Full CNF。

1.2.10 10. Functionally Complete Operators (功能完备集)

Definition: 一组逻辑运算符称为 **functionally complete** (功能完备) 如果任意复合命题都可以用这组运算符等价表示。**常见功能完备集**:

- $\{\neg, \vee\}$
- $\{\neg, \wedge\}$
- $\{\downarrow\}$ (Peirce arrow, NOR —或非)
- $\{|\}$ (Sheffer stroke, NAND —与非)

1.2.11 11. Propositional Satisfiability (可满足性)

Definition:

- **Satisfiable** (可满足): 存在一组真值赋值使得复合命题为真。
- **Unsatisfiable** (不可满足): 不存在这样的赋值（即为矛盾式或永假式）。

Example: 判断以下命题的可满足性:

1. $(p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p)$ —**Satisfiable** (令 $p = q = r = T$)
2. $(p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee q) \wedge (\neg p \vee \neg q)$ —**Unsatisfiable**

应用: Sudoku 问题可编码为 SAT (satisfiability) 问题。共 $9 \times 9 \times 9 = 729$ 个命题变量 $p(i, j, n)$ 表示第 i 行第 j 列放数字 n。

1.2.12 Summary & Exam Tips (小结与考点提示)

考点	说明
Tautology vs Contradiction	永真 vs 永假；用真值表或等价推导判断
Logical Equivalence 证明	真值表法 / 等价推导法，重点掌握 De Morgan 和 Implication Law
De Morgan's Laws	$\neg(p \wedge q) \equiv \neg p \vee \neg q$ 和 $\neg(p \vee q) \equiv \neg p \wedge \neg q$
CNF 与 DNF 转换	先消 $\rightarrow$ 和 $\leftrightarrow$，再移动否定，最后用分配律
Minterm/Maxterm	从真值表直接构造 Full DNF/CNF
Satisfiability	判定是否存在真值赋值使命题为真
Dual	$\vee \leftrightarrow \wedge$, $\mathbf{T} \leftrightarrow \mathbf{F}$
功能完备集	$\{\neg, \vee\}$, $\{\neg, \wedge\}$, $\{\downarrow\}$, $\{|\}$

易错提醒:

- $p \rightarrow q \equiv \neg p \vee q$，千万不要混淆 $p \rightarrow q$ 和 $q \rightarrow p$!
- De Morgan 展开时，$\neg(p \vee q)$ 变为 $\neg p \wedge \neg q$ (连接词要变)
- 判断 CNF/DNF 时，检查每个子句内是否只含 $\vee$ (CNF) 或只含 $\wedge$ (DNF)

1.3 Chapter 1.4: Predicates and Quantifiers (谓词与量词)

DMA Review Notes – Bilingual (English/Chinese)

1.3.1 1. Why Predicate Logic? (为什么需要谓词逻辑)

命题逻辑无法表达以下推理:

- "All men are mortal."
- "Socrates is a man."
- $\therefore$ "Socrates is mortal."

需要一种能描述**对象**、**性质**和**关系**的语言 => **Predicate Logic (谓词逻辑)**

1.3.2 2. Predicates (谓词) and Propositional Functions

Definition: A **predicate** (谓词) is a statement that contains variables and becomes a proposition when the variables are assigned specific values.

- $P(x)$: 命题函数 (propositional function)，x 为变量，P 为谓词。
- 当 x 被赋予一个具体值，或 x 被量词绑定 (quantified) 时，$P(x)$ 成为命题。

Example: $P(x)$: "$x > 0$", 论域 (domain) 为整数。

- $P(-3)$ 假, $P(0)$ 假, $P(3)$ 真

多变量谓词: $R(x, y, z)$: "$x + y = z$"

- $R(2, -1, 5)$: $2 + (-1) = 5$? => False
- $R(3, 4, 7)$: $3 + 4 = 7$? => True
- $R(x, 3, z)$: 不是命题 (变量未绑定)

复合表达式: 命题连接词可直接用于谓词。

- $P(3) \vee P(-1)$: True (因为 $3 > 0$ 真)
- $P(3) \wedge P(-1)$: False
- $P(3) \to P(-1)$: False

1.3.3 3. Quantifiers (量词)

3.1 Universal Quantifier (全称量词) $\forall$

$\forall x P(x)$: "$P(x)$ is true for **every** x in the domain."

- 读作: "For all x, $P(x)$" 或"For every x, $P(x)$" - 域中使 $P(x)$ 为假的元素称为 **counterexample** (反例)

Truth: $\forall x P(x)$ 为真当且仅当论域中每个 x 都使 $P(x)$ 为真。

Examples: 1. $P(x)$: "$x > 0$", 论域为整数 => $\forall x P(x)$ **假** (负数为反例) 2. $P(x)$: "$x > 0$", 论域为正整数 => $\forall x P(x)$ **真** 3. $P(x)$: "$x < 3$", 论域为 $\{1, 2, 3\}$ => $\forall x P(x)$ **假** (因为 $P(3)$ 假)

与合取的关系 (有限域):

$$\forall x P(x) \equiv P(x_1) \wedge P(x_2) \wedge \cdots \wedge P(x_n)$$

3.2 Existential Quantifier (存在量词) $\exists$

$\exists x P(x)$: "There **exists** an x in the domain such that $P(x)$ is true."

Truth: $\exists x P(x)$ 为真当且仅当论域中至少有一个 x 使 $P(x)$ 为真。

Examples: 1. $P(x)$: "$x > 0$", 论域为整数 => $\exists x P(x)$ **真** 2. $P(x)$: "$x < 0$", 论域为正整数 => $\exists x P(x)$ **假** 3. $P(x)$: "$x < 3$", 论域为 $\{1, 2, 3\}$ => $\exists x P(x)$ **真** (因为 $P(1)$ 真)

与析取的关系 (有限域):

$$\exists x P(x) \equiv P(x_1) \vee P(x_2) \vee \cdots \vee P(x_n)$$

3.3 Uniqueness Quantifier (唯一性量词) $\exists!$

$\exists! x P(x)$: "There exists a **unique** x such that $P(x)$." (恰好存在一个)

Examples: 1. $P(x)$: "$x + 1 = 0$", 论域为整数 => $\exists! x P(x)$ **真** (仅 $x = -1$) 2. $P(x)$: "$x > 0$", 论域为整数 => $\exists! x P(x)$ **假** (不止一个)

等价表达 (无须独立量词):

$$\exists! x P(x) \equiv \exists x (P(x) \wedge \forall y (P(y) \to y = x))$$

1.3.4 4. Domain (论域) 的重要性

$\forall x P(x)$ 和 $\exists x P(x)$ 的真值同时取决于 $P(x)$ 和 **domain (论域)** U。**Example**:

论域	$P(x)$: $x < 2$	$\exists x P(x)$	$\forall x P(x)$
正整数	$x < 2$	True (x=1)	True (只有 1)
负整数	$x < 2$	True	True
$\{3, 4, 5\}$	$x < 2$	False	False

1.3.5 5. Precedence of Quantifiers (量词优先级)

$\forall$ 和 $\exists$ 的优先级**高于**所有逻辑运算符。

Example: $\forall x P(x) \vee Q(x)$ 意为 $(\forall x P(x)) \vee Q(x)$，**不是** $\forall x (P(x) \vee Q(x))$。

1.3.6 6. Binding Variables (变量绑定)

- **Bound variable** (约束变量): 被量词绑定或赋了具体值的变量。
- **Free variable** (自由变量): 未被量化或赋值的变量。
- **Scope** (作用域): 量词所作用的那部分逻辑表达式。

Example: $\exists x (P(x) \wedge Q(x)) \vee \forall x R(x)$

- 第一个 $\exists x$ 的作用域是 $P(x) \wedge Q(x)$
- 第二个 $\forall x$ 的作用域是 $R(x)$

命题函数中的所有变量必须被量化或赋值后才能成为命题。

1.3.7 7. Negating Quantifiers (量词的否定) – De Morgan's Laws for Quantifiers

原式	否定等价式
$\neg \forall x P(x)$	$\exists x \neg P(x)$
$\neg \exists x P(x)$	$\forall x \neg P(x)$

记忆口诀: 否定量词时，$\forall$ 变 $\exists$，$\exists$ 变 $\forall$，谓词取否定。

直观理解: - " 并非所有人都会 Java" $\iff$ " 存在某人不会 Java" - $\neg \forall x J(x) \equiv \exists x \neg J(x)$ - " 不存在某人会 Java" $\iff$ " 所有人都不会 Java" - $\neg \exists x J(x) \equiv \forall x \neg J(x)$

1.3.8 8. Logical Equivalences Involving Quantifiers (量词相关的逻辑等价式)

设 x 不出现于 A 中:

编号	等价式
1	$\forall x (P(x) \vee A) \equiv \forall x P(x) \vee A$
2	$\exists x (P(x) \wedge A) \equiv \exists x P(x) \wedge A$
3	$\forall x (P(x) \wedge A) \equiv \forall x P(x) \wedge A$
4	$\exists x (P(x) \vee A) \equiv \exists x P(x) \vee A$
5	$\forall x (A \to P(x)) \equiv A \to \forall x P(x)$
6	$\exists x (A \to P(x)) \equiv A \to \exists x P(x)$
7	$\forall x (P(x) \to A) \equiv \exists x P(x) \to A$
8	$\exists x (P(x) \to A) \equiv \forall x P(x) \to A$

注意: 第 7、8 条在 $\to$ 右侧的 A 不包含 x 时，量词会**翻转**。这是常见考点!

1.3.9 9. Translating English to Predicate Logic (英译谓词逻辑)

核心规则

英文	逻辑形式	要点
"All X are Y"	$\forall x (X(x) \to Y(x))$	用 $\to$
"Some X are Y"	$\exists x (X(x) \wedge Y(x))$	用 $\wedge$
"No X are Y"	$\neg \exists x (X(x) \wedge Y(x))$ 或 $\forall x (X(x) \to \neg Y(x))$	两种表达
"Some X are not Y"	$\exists x (X(x) \wedge \neg Y(x))$	用 $\wedge$ + $\neg$

关键提醒: - $\forall x (S(x) \to J(x)) \iff$ "All students have taken Java" - $\forall x (S(x) \wedge J(x)) \iff$ "Everything is a student who has taken Java" (错误!) - $\exists x (S(x) \wedge J(x)) \iff$ "Some student has taken Java" - $\exists x (S(x) \to J(x)) \iff$ 这几乎是永真式 (如果 x 不是学生，自动为真)，**不是我们想要的"Some" 表达**。

Examples

Example 1: "Every student in this class has taken a course in Java."

- 论域为所有学生: $\forall x J(x)$
- 论域为所有人: $\forall x (S(x) \to J(x))$ (不能用 $\wedge$!)

Example 2: "Some student in this class has taken a course in Java."

- 论域为所有学生: $\exists x J(x)$
- 论域为所有人: $\exists x (S(x) \wedge J(x))$ (不能用 $\to$!)

Example 3: "All lions are fierce." + "Some lions do not drink coffee." $\implies$ "Some fierce creatures do not drink coffee."

- $P(x)$: "x is a lion", $Q(x)$: "x is fierce", $R(x)$: "x drinks coffee"
- Premises:

1. $\forall x (P(x) \to Q(x))$
2. $\exists x (P(x) \wedge \neg R(x))$

- Conclusion: $\exists x (Q(x) \wedge \neg R(x))$

Example 4: 设 $C(x)$: "x is a CS student", $E(x)$: "x is a Math student", $S(x)$: "x is a smart student"

1. "Everyone is a CS student": $\forall x C(x)$
2. "Nobody is a Math student": $\forall x \neg E(x)$ 或 $\neg \exists x E(x)$
3. "All CS students are smart": $\forall x (C(x) \to S(x))$
4. "Some CS students are smart": $\exists x (C(x) \wedge S(x))$
5. "No CS student is a Math student": $\forall x (C(x) \to \neg E(x))$ 或 $\neg \exists x (C(x) \wedge E(x))$
6. "If any Math student is a smart student then he is also a CS student": $\forall x ((E(x) \wedge S(x)) \to C(x))$

Example 5: "Every mail message larger than one megabyte will be compressed."

- $L(m, y)$: "Mail message m is larger than y megabytes"
- $C(m)$: "Mail message m will be compressed"
- $\forall m (L(m, 1) \to C(m))$

1.3.10 Summary & Exam Tips (小结与考点提示)

考点	说明
谓词 vs 命题	含自由变量的谓词不是命题；变量被赋值或量化后才成为命题
$\forall$ 与 $\wedge$	$\forall x (S(x) \to P(x))$ —用 $\to$
$\exists$ 与 $\vee$	$\exists x (S(x) \wedge P(x))$ —用 $\wedge$
否定量词	$\neg \forall x P \equiv \exists x \neg P$, $\neg \exists x P \equiv \forall x \neg P$
量词优先级	$\forall$ 和 $\exists$ 高于 $\vee$, $\wedge$, $\to$, $\leftrightarrow$
量词等价式	注意第 7、8 条：$\forall$ 在 $\to$ 右侧会翻转为 $\exists$
论域影响真值	同样 $P(x)$ 在不同论域中真值不同

易错提醒:

- 别忘了量词的德摩根律: $\neg\forall = \exists\neg$, $\neg\exists = \forall\neg$
- "All X are Y" 翻译为 $\forall x(X(x) \to Y(x))$，不是 $\forall x(X(x) \wedge Y(x))$!
- "Some X are Y" 翻译为 $\exists x(X(x) \wedge Y(x))$，不是 $\exists x(X(x) \to Y(x))$!
- $\exists!$ 可以用 $\exists$ 和 $\forall$ 表达，无须额外记忆特殊规则

1.4 Chapter 1.5: Nested Quantifiers (嵌套量词)

DMA Review Notes – Bilingual (English/Chinese)

1.4.1 1. Nested Quantifiers (嵌套量词)

Definition: Two quantifiers are **nested** (嵌套) if one is within the scope of the other. **Example**: "Every real number has an additive inverse."

$$\forall x \exists y(x + y = 0)$$

其中论域为实数集。这里 $\exists y$ 在 $\forall x$ 的作用域内。**嵌套的理解**: 可将 $\forall x \exists y P(x, y)$ 视为 $\forall x Q(x)$，其中 $Q(x) = \exists y P(x, y)$。

1.4.2 2. Thinking of Nested Quantification (类比为嵌套循环)

将量词视为循环:

表达式	理解方式
$\forall x \forall y P(x, y)$	外层循环 x，内层循环 y，每一对 (x, y) 必须使 P 为真
$\forall x \exists y P(x, y)$	外层循环 x，对每个 x 至少要找到一个 y 使 P 为真
$\exists x \forall y P(x, y)$	找到一个 x，使它对**所有** y 都让 P 为真
$\exists x \exists y P(x, y)$	找到一对 (x, y) 使 P 为真

1.4.3 3. Order of Quantifiers (量词顺序)

量词的顺序至关重要，不同的顺序可能产生完全不同的真值!

Example 1: $P(x, y)$: "$x + y = y + x$", 论域为实数 - $\forall x \forall y P(x, y)$: **True** (加法交换律永远成立) - $\forall y \forall x P(x, y)$: **True** (等价)

Example 2: $Q(x, y)$: "$x + y = 0$", 论域为实数 - $\forall x \exists y Q(x, y)$: **True** (对每个 x，存在 $y = -x$) - $\exists y \forall x Q(x, y)$: **False** (不存在一个 y 使对所有 x 都有 $x + y = 0$)

Example 3: $P(x, y)$: "$x \cdot y = 0$", 论域为实数

| 表达式 | 真值 | 解释 | |:——|:—-:|:—-| | $\forall x \forall y P(x, y)$ | **F** | 非所有数对乘积为 0 | | $\forall x \exists y P(x, y)$ | **T** | 对任意 x，取 $y = 0$ 即得 0 | | $\exists x \forall y P(x, y)$ | **T** | 取 $x = 0$，对所有 y 乘积为 0 | | $\exists x \exists y P(x, y)$ | **T** | 取 $x = 0, y = 0$ 即可 |

Example 4: $P(x, y)$: "$x/y = 1$", 论域为正实数

| 表达式 | 真值 | 解释 | |:——|:—-:|:—-| | $\forall x \forall y P(x, y)$ | **F** | 非所有 $x/y = 1$ | | $\forall x \exists y P(x, y)$ | **T** | 对任意 x，取 $y = x$ | | $\exists x \forall y P(x, y)$ | **F** | 不存在一个 x 使对所有 y 都有 $x/y = 1$ | | $\exists x \exists y P(x, y)$ | **T** | 取 $x = 1, y = 1$ |

1.4.4 4. Quantifications of Two Variables —Truth Table Summary

Statement	When True?	When False?
$\forall x \forall y P(x, y)$ $\forall y \forall x P(x, y)$	$P(x, y)$ 对**每一对** (x, y) 为真	存在某对 (x, y) 使 $P(x, y)$ 为假
$\forall x \exists y P(x, y)$	对**每个** x，**存在**某个 y 使 P 为真	存在某个 x，对**所有** y P 为假
$\exists x \forall y P(x, y)$	存在某个 x，对**所有** y P 为真	对**每个** x，存在某个 y 使 P 为假
$\exists x \exists y P(x, y)$ $\exists y \exists x P(x, y)$	存在某对 (x, y) 使 P 为真	对**每一对** (x, y) P 为假

核心结论: $\forall$ 和 $\exists$ **交换顺序不一定等价**。相同量词交换顺序等价，不同量词交换顺序可能改变真值。

1.4.5 5. Translating from Nested Quantifiers into English

Example 1: 将以下表达式译为中文:

$$\forall x(C(x) \vee \exists y(C(y) \wedge F(x, y)))$$

其中 $C(x)$: "x has a computer", $F(x, y)$: "x and y are friends", 论域为 ZJU 全体学生。**Solution**: "Every student at ZJU has a computer or has a friend who has a computer." (对于 ZJU 的每个学生 x，要么 x 有电脑，要么存在某个学生 y 有电脑并且 x 和 y 是朋友。) **Example 2**: 将以下逻辑表达式译为中文:

$$\exists x \forall y \forall z((F(x, y) \wedge F(x, z) \wedge (y \neq z)) \to \neg F(y, z))$$

其中 $F(x, y)$: "x and y are friends", 论域为 ZJU 学生。**Solution**: "There is a student none of whose friends are also friends with each other." (存在一个学生 x，使得对于任意两个不同的 y 和 z，如果 x 和 y 是朋友且 x 和 z 是朋友，则 y 和 z 不是朋友。)

1.4.6 6. Translating English into Nested Quantifiers

Example 1: "Everyone has exactly one best friend."

- $B(x, y)$: "y is the best friend of x"
- 改写: For every person x, there is a person y who is the best friend of x, and **if z is not y, then z is not the best friend of x.**
- $$\forall x \exists y(B(x, y) \wedge \forall z((z \neq y) \to \neg B(x, z)))$$

Example 2: "If a person is female and is a parent, then this person is someone's mother."

- $F(x)$: "x is female", $P(x)$: "x is a parent", $M(x, y)$: "x is the mother of y"
- $$\forall x((F(x) \wedge P(x)) \to \exists y M(x, y))$$

Example 3: "There is a woman who has taken a flight on every airline in the world."

- $P(w, f)$: "w has taken flight f", $Q(f, a)$: "f is a flight on airline a"
- 论域: $w =$ 所有女性, $f =$ 所有航班, $a =$ 所有航空公司
- $$\exists w \forall a \exists f(P(w, f) \wedge Q(f, a))$$

Example 4: "Brothers are siblings."

- $B(x, y)$: "x and y are brothers", $S(x, y)$: "x and y are siblings"
- $$\forall x \forall y(B(x, y) \to S(x, y))$$

Example 5: "Siblinghood is symmetric."

- $$\forall x \forall y(S(x, y) \to S(y, x))$$

Example 6: "Everybody loves somebody." vs "There is someone who is loved by everyone."

- $\forall x \exists y L(x, y)$ vs $\exists y \forall x L(x, y)$ —**含义完全不同!**

1.4.7 7. Translating Mathematical Statements

Example 1: "The sum of two positive integers is always positive."

- 论域为整数
- $$\forall x \forall y((x > 0 \wedge y > 0) \to (x + y > 0))$$

Example 2: 极限的定义 $\lim_{x \to a} f(x) = L$:

$$\forall \varepsilon > 0\ \exists \delta > 0\ \forall x\ (0 < |x - a| < \delta \to |f(x) - L| < \varepsilon)$$

Example 3: Quantifiers with restricted domains (限制域):

- $\forall x < 0\ (x^2 > 0) \equiv \forall x(x < 0 \to x^2 > 0)$
- $\exists y > 0\ (y^2 = 2) \equiv \exists y(y > 0 \wedge y^2 = 2)$

1.4.8 8. Negating Nested Quantifiers

方法: 逐层应用 De Morgan's Laws for Quantifiers

Example 1: 否定 $\exists w \forall a \exists f(P(w, f) \wedge Q(f, a))$ **Solution**:

$$\begin{aligned}
&\neg \exists w \forall a \exists f(P(w, f) \wedge Q(f, a)) \\
&\equiv \forall w \neg \forall a \exists f(P(w, f) \wedge Q(f, a)) && \text{(De Morgan for } \exists) \\
&\equiv \forall w \exists a \neg \exists f(P(w, f) \wedge Q(f, a)) && \text{(De Morgan for } \forall) \\
&\equiv \forall w \exists a \forall f \neg(P(w, f) \wedge Q(f, a)) && \text{(De Morgan for } \exists) \\
&\equiv \forall w \exists a \forall f(\neg P(w, f) \vee \neg Q(f, a)) && \text{(De Morgan for } \wedge)
\end{aligned}$$

翻译回英文: "For every woman, there is an airline such that for all flights, this woman has not taken that flight or that flight is not on this airline." **Example 2**: 否定 $\forall x \exists y(xy = 1)$ (论域为实数) **Solution**:

$$\begin{aligned}
\neg \forall x \exists y(xy = 1) &\equiv \exists x \neg \exists y(xy = 1) \\
&\equiv \exists x \forall y \neg(xy = 1) \\
&\equiv \exists x \forall y(xy \neq 1)
\end{aligned}$$

Example 3: 证明极限 $\lim_{x\to a} f(x)$ **不存在**: 否定极限定义:

$$\neg[\forall\varepsilon > 0\ \exists\delta > 0\ \forall x\ (0 < |x-a| < \delta \to |f(x)-L| < \varepsilon)]$$

Applying negation step by step:

$$\exists\varepsilon > 0\ \forall\delta > 0\ \exists x\ (0 < |x-a| < \delta \wedge |f(x)-L| \geq \varepsilon)$$

即: " 存在某个 $\varepsilon > 0$，使得对任意 $\delta > 0$，总存在 x 使 $0 < |x-a| < \delta$ 但 $|f(x)-L| \geq \varepsilon$。"

1.4.9 9. Quantifier Equivalences —Common Questions

命题	是否等价?	解释
$\forall x\forall y P(x,y) \equiv \forall y\forall x P(x,y)$	**Yes**	相同量词可交换顺序
$\exists x\exists y P(x,y) \equiv \exists y\exists x P(x,y)$	**Yes**	同上
$\forall x\exists y P(x,y) \equiv \exists y\forall x P(x,y)$	**No**	**不同量词顺序不可交换!**
$\forall x(P(x)\wedge Q(x)) \equiv \forall x P(x)\wedge\forall x Q(x)$	**Yes**	$\forall$ 对 $\wedge$ 可分配
$\exists x(P(x)\vee Q(x)) \equiv \exists x P(x)\vee\exists x Q(x)$	**Yes**	$\exists$ 对 $\vee$ 可分配
$\forall x(P(x)\vee Q(x)) \equiv \forall x P(x)\vee\forall x Q(x)$	**No**	$\forall$ 不可分配 $\vee$ (如: " 所有动物是鱼或有鳞" 与" 所有动物是鱼或所有动物有鳞" 不等价)
$\exists x(P(x)\wedge Q(x)) \equiv \exists x P(x)\wedge\exists x Q(x)$	**No**	$\exists$ 不可分配 $\wedge$

1.4.10 Summary & Exam Tips (小结与考点提示)

考点	说明
量词顺序影响真值	$\forall x\exists y$ 和 $\exists y\forall x$ **不等价!** 这是最强考点
嵌套量词否定	逐层使用 De Morgan，每层 $\forall \leftrightarrow \exists$ 互换，谓词取反
英译逻辑	"Everyone has exactly one..." 需要 $\exists!$ 或 $\exists y\forall z$ 的组合表达
限制域量词	$(\forall x > 0)P(x) \equiv \forall x(x > 0 \to P(x))$; $(\exists x > 0)P(x) \equiv \exists x(x > 0 \wedge P(x))$
极限定义	$\forall\varepsilon > 0\ \exists\delta > 0\ \forall x\ (0 < \lvert x-a\rvert < \delta \to \lvert f(x)-L\rvert < \varepsilon)$

易错提醒:

- $\forall x\exists y P(x,y)$ 与 $\exists y\forall x P(x,y)$ **完全不同**：前者对每个 x 可能有不同的 y，后者要求同一个 y 适用于所有 x
- 否定嵌套量词时，**每层都要反转量词**，不能只反转最外层的
- "Everyone has exactly one..." 的翻译公式要熟记：$\forall x\exists y(P(x,y)\wedge\forall z(P(x,z)\to z=y))$
- $\forall$ 对 $\vee$ 不可分配，$\exists$ 对 $\wedge$ 不可分配

1.5 Chapter 1.6: Rules of Inference (推理规则)

DMA Review Notes – Bilingual (English/Chinese)

1.5.1 1. Valid Arguments (有效论证)

Definition: An **argument** (论证) in propositional logic is a sequence of propositions. All but the final proposition are **premises** (前提). The last statement is the **conclusion** (结论). **Valid**: The argument is valid if the premises imply the conclusion. i.e.,

$$(p_1 \wedge p_2 \wedge \cdots \wedge p_n) \to q$$

is a **tautology**.

有效性不要求前提为真，只要求" 如果前提为真，则结论必为真"。

Argument form (论证形式): 将命题变量替换为具体命题后依然有效的形式。

Inference rules (推理规则): 用于构建复杂论证的基本有效论证形式。

1.5.2 2. Rules of Inference for Propositional Logic

2.1 Modus Ponens (假言推理/肯定前件式)

$$\begin{array}{c} p \\ p \to q \\ \hline \therefore q \end{array}$$

- 对应的永真式: $(p \wedge (p \to q)) \to q$

Example:

- p: "It is snowing."
- q: "I will study discrete math."
- "If it is snowing, then I will study discrete math."
- "It is snowing."
- $\therefore$ "I will study discrete math."

2.2 Modus Tollens (取拒式/否定后件式)

$$\begin{array}{c} \neg q \\ p \to q \\ \hline \therefore \neg p \end{array}$$

- 对应的永真式: $(\neg q \wedge (p \to q)) \to \neg p$

Example:

- "If it is snowing, then I will study discrete math."
- "I will not study discrete math."
- $\therefore$ "It is not snowing."

2.3 Hypothetical Syllogism (假言三段论)

$$\begin{array}{c} p \to q \\ q \to r \\ \hline \therefore p \to r \end{array}$$

- 对应的永真式: $((p \to q) \wedge (q \to r)) \to (p \to r)$

Example:

- "If it snows, then I will study discrete math."
- "If I study discrete math, I will get an A."
- $\therefore$ "If it snows, I will get an A."

2.4 Disjunctive Syllogism (析取三段论)

$$\begin{array}{c} p \vee q \\ \neg p \\ \hline \therefore q \end{array}$$

- 对应的永真式: $((p \vee q) \wedge \neg p) \to q$

Example:

- "I will study discrete math or I will study English literature."
- "I will not study discrete math."
- $\therefore$ "I will study English literature."

2.5 Addition (附加律)

$$\begin{array}{c} p \\ \hline \therefore p \vee q \end{array}$$

- 对应的永真式: $p \to (p \vee q)$

Example:

- "I will study discrete math."
- $\therefore$ "I will study discrete math or I will visit Las Vegas."

2.6 Simplification (化简律)

$$\begin{array}{c} p \wedge q \\ \hline \therefore p \end{array}$$

- 对应的永真式: $(p \wedge q) \to p$

Example:

- "I will study discrete math and English literature."
- $\therefore$ "I will study discrete math."

2.7 Conjunction (合取律)

$$\begin{array}{c} p \\ q \\ \hline \therefore p \wedge q \end{array}$$

- 对应的永真式: $(p \wedge q) \to (p \wedge q)$ (即 $p \to (q \to (p \wedge q))$)

Example:

- "I will study discrete math."
- "I will study English literature."
- $\therefore$ "I will study discrete math and I will study English literature."

2.8 Resolution (消解律)

$$\begin{array}{c} p \vee q \\ \neg p \vee r \\ \hline \therefore q \vee r \end{array}$$

- 对应的永真式: $((p \vee q) \wedge (\neg p \vee r)) \to (q \vee r)$
- 在 AI 和 Prolog 中有重要应用。

Example:

- p: "I will study discrete math."
- q: "I will study databases."
- r: "I will study English literature."
- "I will not study discrete math or I will study English literature." ($\neg p \vee r$)
- "I will study discrete math or I will study databases." ($p \vee q$)
- $\therefore$ "I will study databases or I will study English literature." ($q \vee r$)

1.5.3 3. Building Valid Arguments (构建有效论证)

步骤:

1. 将自然语言命题翻译为命题逻辑符号。
2. 列出所有前提。
3. 逐行应用推理规则，直到得到结论。

Example: Swimsuit Problem

Premises:

- "It is not sunny this afternoon and it is colder than yesterday."
- "We will go swimming only if it is sunny."
- "If we do not go swimming, then we will take a canoe trip."
- "If we take a canoe trip, then we will be home by sunset."

Conclusion: "We will be home by sunset." **Solution**:

1. 定义命题变量:

- p: "It is sunny this afternoon."
- q: "It is colder than yesterday."
- r: "We will go swimming."
- s: "We will take a canoe trip."
- t: "We will be home by sunset."

1. 翻译前提:

- $\neg p \wedge q$
- $r \to p$ (注意: "p only if q" $\equiv p \to q$, 所以"go swimming only if sunny" $= r \to p$)
- $\neg r \to s$
- $s \to t$

1. 构建有效论证:

步骤	命题	推理规则
1	$\neg p \wedge q$	Premise
2	$\neg p$	Simplification from (1)
3	$r \to p$	Premise
4	$\neg r$	Modus Tollens from (2)(3)
5	$\neg r \to s$	Premise
6	s	Modus Ponens from (4)(5)
7	$s \to t$	Premise
8	t	Modus Ponens from (6)(7)

$\therefore$ "We will be home by sunset."

1.5.4 4. Rules of Inference for Quantified Statements

4.1 Universal Instantiation (UI, 全称实例)

$$\frac{\forall x P(x)}{\therefore P(c)}$$

(其中 c 是论域中的某个元素) **Example**: "All dogs are cuddly." $\implies$ "Fido is cuddly."

4.2 Universal Generalization (UG, 全称引入)

$$\frac{P(c) \text{ for an arbitrary } c}{\therefore \forall x P(x)}$$

(其中 c 是论域中的任意元素，在数学证明中经常隐式使用)

4.3 Existential Instantiation (EI, 存在实例)

$$\frac{\exists x P(x)}{\therefore P(c) \text{ for some element } c}$$

(将存在性陈述变为某个具体元素的性质，c 为尚未使用过的符号) **Example**: "There is someone who got an A in the course." $\implies$ "Let's call her a and say that a got an A."

4.4 Existential Generalization (EG, 存在引入)

$$\frac{P(c) \text{ for some } c}{\therefore \exists x P(x)}$$

Example: "Michelle got an A in the class." $\implies$ "Someone got an A in the class."

1.5.5 5. Using Rules of Inference with Quantifiers

Example 1: Socrates 论证

Premises:

- $\forall x(M(x) \to L(x))$ ("Every man has two legs")
- $M(\text{John Smith})$ ("John Smith is a man")

Conclusion: $L(\text{John Smith})$ ("John Smith has two legs") **Valid Argument**:

步骤	命题	推理规则
1	$\forall x(M(x) \to L(x))$	Premise
2	$M(\text{JS}) \to L(\text{JS})$	UI from (1)
3	$M(\text{JS})$	Premise
4	$L(\text{JS})$	Modus Ponens from (2)(3)

Example 2: Someone passed the exam but hasn't read the book

Premises:

- "A student in this class has not read the book." ($\exists x(C(x) \wedge \neg B(x))$)
- "Everyone in this class passed the first exam." ($\forall x(C(x) \to P(x))$)

Conclusion: "Someone who passed the first exam has not read the book." ($\exists x(P(x) \wedge \neg B(x))$) **Valid Argument**:

步骤	命题	推理规则
1	$\exists x(C(x) \wedge \neg B(x))$	Premise
2	$C(a) \wedge \neg B(a)$	EI from (1)
3	$C(a)$	Simplification from (2)
4	$\forall x(C(x) \to P(x))$	Premise
5	$C(a) \to P(a)$	UI from (4)
6	$P(a)$	Modus Ponens from (3)(5)
7	$\neg B(a)$	Simplification from (2)
8	$P(a) \wedge \neg B(a)$	Conjunction from (6)(7)
9	$\exists x(P(x) \wedge \neg B(x))$	EG from (8)

1.5.6 6. Universal Modus Ponens (全称假言推理)

结合 Universal Instantiation 和 Modus Ponens 为一体:

$$\begin{array}{c} \forall x(P(x) \to Q(x)) \\ P(a) \\ \hline \therefore Q(a) \end{array}$$

Example (Socrates):

- $\forall x(\text{Man}(x) \to \text{Mortal}(x))$
- Man(Socrates)
- $\therefore$ Mortal(Socrates)

1.5.7 Summary & Exam Tips (小结与考点提示)

考点	说明
8 条推理规则	Modus Ponens, Modus Tollens, Hypothetical Syllogism, Disjunctive Syllogism, Addition, Simplification, Conjunction, Reso
量词推理规则	UI (全称实例), UG (全称引入), EI (存在实例), EG (存在引入)
论证构建	将 premise 用符号表示，逐条应用推理规则，一步步推导出结论
全称假言推理	$\forall x(P(x) \to Q(x)),\ P(a) \implies Q(a)$ —最常用的推理模式

推理规则速查表:

规则	前提	结论
Modus Ponens	$p,\ p \to q$	q
Modus Tollens	$\neg q,\ p \to q$	$\neg p$
Hypothetical Syllogism	$p \to q,\ q \to r$	$p \to r$
Disjunctive Syllogism	$p \vee q,\ \neg p$	q
Addition	p	$p \vee q$
Simplification	$p \wedge q$	p
Conjunction	$p,\ q$	$p \wedge q$
Resolution	$p \vee q,\ \neg p \vee r$	$q \vee r$

易错提醒:

- Modus Tollens 是 $\neg q$ 和 $p \to q$ 推出 $\neg p$，**不是** $\neg p$ 和 $p \to q$ 推出 $\neg q$ (否定前件谬误!)
- EI 引入的 c 必须是**未曾用过的新符号**
- UI 和 EG 更安全，EI 和 UG 使用时有约束条件
- "A only if B" $\equiv A \to B$，在构建论证时特别注意

1.6 Chapter 1.7: Introduction to Proofs (证明导论)

DMA Review Notes – Bilingual (English/Chinese)

1.6.1 1. Basic Terminology (基本术语)

术语	定义
Theorem (定理)	能被证明为真的陈述
Proof (证明)	建立定理为真的有效论证
Axiom (公理)	被假定为真的前提（无须证明）
Lemma (引理)	用来证明其他定理的中间结果（"helper theorem"）
Corollary (推论)	从定理直接推出的结果
Conjecture (猜想)	被提议为真但尚未证明的陈述；找到证明后变为定理
Proposition (命题)	相对不重要的定理

Proof $= P_1 \wedge P_2 \wedge \cdots \wedge P_n \to C$ 是永真式，其中 P_i 为前提/公理/已知定理，C 为结论。**Informal proofs** (非形式化证明): 更简短、更易于理解，但可能隐含错误。

1.6.2 2. Forms of Theorems

大多数定理形如: $\forall x(P(x) \to Q(x))$，但由于数学惯例，全称量词常被省略。**Example**: "If $x > y$, where x and y are positive real numbers, then $x^2 > y^2$" 实际上意味着: "For all positive real numbers x and y, if $x > y$, then $x^2 > y^2$。" 证明方法: 取论域中的任意元素 c，证明 $P(c) \to Q(c)$，然后用 Universal Generalization 得到 $\forall x(P(x) \to Q(x))$。

1.6.3 3. Proving Conditional Statements $p \to q$

3.1 Trivial Proof (平凡证明)

若 q 已知为真，则 $p \to q$ 自动为真 (无论 p 的真值)。**Example**: "If it is raining then $1 = 1$." —结论恒真，证明平凡成立。

3.2 Vacuous Proof (空证明)

若 p 为假，则 $p \to q$ 自动为真。**Example**: "If I am both rich and poor then $2 + 2 = 5$." —前提永假，蕴含自动为真。

Trivial 和 Vacuous proof 常用于数学归纳法 (Chapter 5) 的证明中。

3.3 Direct Proof (直接证明法)

假定前提 p 为真，利用推理规则、公理、逻辑等价式，逐步推导出 q 为真。

Example: "If n is an odd integer, then n^2 is odd."

Proof: Assume n is odd. Then $n = 2k + 1$ for some integer k.

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

where $r = 2k^2 + 2k$ is an integer. Hence n^2 is odd. □

Example (Rational Numbers): 证明两个有理数的和是有理数。

Definition: 实数 r 称为 **rational** (有理数) 如果存在整数 p, q (其中 $q \neq 0$) 使得 $r = p/q$。

Proof: 设 r 和 s 是两个有理数，则存在整数 p, q, t, u (其中 $q \neq 0, u \neq 0$) 使得 $r = p/q$, $s = t/u$。

$$r + s = \frac{p}{q} + \frac{t}{u} = \frac{pu + qt}{qu}$$

由于 $pu + qt$ 和 qu 均为整数且 $qu \neq 0$，故 $r + s$ 是有理数。□

3.4 Proof by Contraposition (反证法 / 逆否证明)

假定 $\neg q$ 为真，证明 $\neg p$ 为真 (即证明 $\neg q \to \neg p$)。由于 $p \to q \equiv \neg q \to \neg p$，这就证明了原命题。

这是 **indirect proof** (间接证明) 的一种形式。

Example 1: "If n is an integer and $3n + 2$ is odd, then n is odd."

Proof (by contraposition): - Assume n is even (i.e., not odd). Then $n = 2k$ for some integer k. - Then $3n + 2 = 3(2k) + 2 = 6k + 2 = 2(3k + 1) = 2j$, which is even. - So if n is even, $3n + 2$ is even. By contraposition, if $3n + 2$ is odd, n is odd. □

Example 2: "If n^2 is odd, then n is odd."

Proof (by contraposition): Assume n is even. Then $n = 2k$, so $n^2 = 4k^2 = 2(2k^2)$ is even. Thus if n^2 is odd, n must be odd. □

3.5 Proof by Contradiction (归谬证明法)

假定 p 为假 (即 $\neg p$ 为真)，推导出矛盾 (如 $r \wedge \neg r$)。由于 $\neg p \to \mathbf{F}$ 为真，则 p 必为真。

形式: 要证 p，假设 $\neg p$，推导出矛盾。

Example 1: "If you pick 22 days from the calendar, at least 4 must fall on the same day of the week."

Proof (by contradiction): Assume that at most 3 of the 22 days fall on the same day. Since there are 7 days, we could only have picked $3 \times 7 = 21$ days. But we picked 22 days. Contradiction. Hence at least 4 must fall on the same day. □

Example 2: "$\sqrt{2}$ is irrational."

Proof (by contradiction): Assume $\sqrt{2}$ is rational. Then $\sqrt{2} = a/b$ where a and b are integers, $b \neq 0$, and a and b have no common factors. - Squaring: $2 = a^2/b^2$, so $a^2 = 2b^2$. - Hence a^2 is even, so a is even. Write $a = 2c$. - Then $(2c)^2 = 2b^2$, i.e., $4c^2 = 2b^2$, so $b^2 = 2c^2$. - Hence b^2 is even, so b is even. - Then 2 divides both a and b, contradicting the assumption that they have no common factors. - Therefore $\sqrt{2}$ is irrational. □

Example 3: "There is no largest prime number."

Proof (by contradiction): Assume there is a largest prime, p_n. - List all primes: $p_1, p_2, \ldots, p_n$. - Consider $r = p_1 \cdot p_2 \cdot \cdots \cdot p_n + 1$. - No prime on the list divides r (there is always remainder 1). - So either r is prime or a smaller prime divides r. - This contradicts the assumption that p_n is the largest prime. - Hence there is no largest prime. □

1.6.4 4. Relation between Proof Methods

对于 $p \to q$:

方法	假设	推导	依据
Direct Proof	p	q	直接推理
Proof by Contraposition	$\neg q$	$\neg p$	$p \to q \equiv \neg q \to \neg p$
Proof by Contradiction	$\neg(p \to q)$	contradiction	即 $p \wedge \neg q$ 推出矛盾

注意: Proof by contradiction 证明 $p \to q$ 时，假设 $p \wedge \neg q$ 推出矛盾即可。

1.6.5 5. Proving Biconditional Statements ($p \leftrightarrow q$)

同时证明两个方向:

1. $p \to q$
2. $q \to p$

Example: "If n is an integer, then n is odd if and only if n^2 is odd." **Proof**:

- ($\to$): We already proved "if n is odd, then n^2 is odd" by direct proof.
- ($\leftarrow$): We already proved "if n^2 is odd, then n is odd" by contraposition.
- Therefore, n is odd $\iff$ n^2 is odd. □

1.6.6 6. What is Wrong with This "Proof"?

常见错误: 证明 $1 = 2$ 的荒谬证明:

1. $a = b$ (给定前提)
2. $a^2 = ab$
3. $a^2 - b^2 = ab - b^2$
4. $(a - b)(a + b) = b(a - b)$

5. $a+b=b$
6. $2b=b$
7. $2=1$

错误: 第 5 步在 $a-b$ 上做了除法，但 $a=b$ 意味着 $a-b=0$，而**除以 0 是未定义的**!

1.6.7 Summary & Exam Tips (小结与考点提示)

考点	说明
Terminology	Theorem, Lemma, Corollary, Axiom, Conjecture 的定义需熟记
Direct Proof	假设前提，直接推导结论。最常见的方法
Proof by Contraposition	证明 $\neg q \to \neg p$。常用于前提不易直接使用的情况
Proof by Contradiction	假设否定，导出矛盾。适合证明否定性结论（如 $\sqrt{2}$ 无理数）
Biconditional Proof	分别证明两个方向 ($\to$ 和 $\leftarrow$)
Trivial/Vacuous Proof	结论恒真或前提恒假时，蕴含自动成立

选择证明方法的思路:

- 前提能直接推出结论? => **Direct Proof**
- 更难直接证明前提 $\to$ 结论? => **Contraposition** ($\neg$ 结论 $\to \neg$ 前提)
- 想证明某事物不存在 / 否定性结论? => **Proof by Contradiction**
- 想证明" 当且仅当"? => **Biconditional** (两个方向分别证)

易错提醒:

- 永远不要除以 0!
- Proof by contradiction 和 proof by contraposition 不同：contraposition 专门用于 $p \to q$ 且只证 $\neg q \to \neg p$；contradiction 更通用，假设任意命题的否定推出矛盾即可
- 反证法重点是找到**逻辑矛盾**，不一定是" 与前提矛盾"

1.7 Chapter 1.8: Proof Methods and Strategy (证明方法与策略)

DMA Review Notes – Bilingual (English/Chinese)

1.7.1 1. Proof by Cases (分情形证明法)

Tautology basis:

$$(p_1 \vee p_2 \vee \cdots \vee p_n) \to q \equiv (p_1 \to q) \wedge (p_2 \to q) \wedge \cdots \wedge (p_n \to q)$$

当命题可分解为若干互斥或穷举的 case 时，分别证明每个 case 下结论成立。

Example: 证明 max 运算满足结合律: $(a@b)@c = a@(b@c)$，其中 $a@b = \max(a,b)$。

Proof: 设 a,b,c 为任意实数，有 6 种可能的大小顺序（cases）: 1. $a \geq b \geq c$ 2. $a \geq c \geq b$ 3. $b \geq a \geq c$ 4. $b \geq c \geq a$ 5. $c \geq a \geq b$ 6. $c \geq b \geq a$

Case 1: $a \geq b \geq c$ - $(a@b) = a$, $a@c = a$, $b@c = b$ - LHS $= (a@b)@c = a@c = a$ - RHS $= a@(b@c) = a@b = a$ - 等式成立。

其余 5 种情况类似可证。□

1.7.2 2. Without Loss of Generality (WLOG, 不失一般性)

当多种情况在本质上对称时，只需证明其中一种，其余可类推。

Example: "If x and y are integers and both $x \cdot y$ and $x+y$ are even, then both x and y are even."

Proof (by contraposition + WLOG): 假设 x 和 y 不都是偶数（即至少一个为奇数）。**WLOG**，假设 x 是奇数。- $x = 2m+1$ 对某整数 m。

Case 1: y 是偶数，$y = 2n$。- $x \cdot y = (2m+1)(2n) = 2n(2m+1)$ 为偶数。- $x+y = (2m+1)+2n = 2(m+n)+1$ 为奇数。

Case 2: y 是奇数，$y = 2n+1$。- $x \cdot y = (2m+1)(2n+1) = 4mn+2m+2n+1 = 2(2mn+m+n)+1$ 为奇数。- $x+y = (2m+1)+(2n+1) = 2(m+n+1)$ 为偶数。

在两种情况下，至少有一个 $x \cdot y$ 或 $x+y$ 不为偶数。由反证法，原命题成立。□

这里假设 x 是奇数 WLOG，因为 x 和 y 对称，证明 y 为奇数的情况类似。

1.7.3 3. Existence Proofs (存在性证明)

形式: $\exists x P(x)$

3.1 Constructive Existence Proof (构造性存在证明)

给出一个具体的 c 使 $P(c)$ 为真。

Example: "Show that there is a positive integer that can be written as the sum of cubes in two different ways."

Proof: $1729 = 10^3 + 9^3 = 12^3 + 1^3$。（这就是著名的 Hardy-Ramanujan 数）□

3.2 Nonconstructive Existence Proof (非构造性存在证明)

不给出具体实例，而是通过推理（如反证法）证明某元素必然存在。

Example: "Show that there exist irrational numbers x and y such that x^y is rational."

Proof: 已知 $\sqrt{2}$ 是无理数。考虑 $\sqrt{2}^{\sqrt{2}}$: - 若 $\sqrt{2}^{\sqrt{2}}$ 是有理数，则取 $x = \sqrt{2}, y = \sqrt{2}$，得到 x^y 为有理数。- 若 $\sqrt{2}^{\sqrt{2}}$ 是无理数，则取 $x = \sqrt{2}^{\sqrt{2}}, y = \sqrt{2}$，则 $x^y = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{(\sqrt{2}\cdot\sqrt{2})} = \sqrt{2}^2 = 2$，是有理数。

无论如何，存在这样的无理数 x, y。但我们并不知道 $\sqrt{2}^{\sqrt{2}}$ 究竟是否无理数——这就是 **nonconstructive**。□

1.7.4 4. Counterexamples (反例)

要证明 $\forall x P(x)$ 为假，只需找到一个 c 使 $P(c)$ 为假，c 即为 **counterexample**。

Example: "Every positive integer is the sum of the squares of 3 integers."

Counterexample: 整数 7 不能表示为三个整数的平方和（$7 = 2^2 + 1^2 + 1^2 + 1^2$ 需四个）。故原命题为假。

1.7.5 5. Nonexistence Proofs (不存在性证明)

要证明 $\neg\exists x P(x)$，即证明 $\forall x \neg P(x)$。常使用 **proof by contradiction**：假设存在这样的 x，推出矛盾。

1.7.6 6. Uniqueness Proofs (唯一性证明)

形式: $\exists! x P(x)$ **两部分**:

1. **Existence** (存在性): 证明存在 x 使 $P(x)$ 为真。
2. **Uniqueness** (唯一性): 证明若 $y \neq x$，则 $\neg P(y)$（等价于：若 $P(y)$ 为真，则 $y = x$）。

Example: "If a and b are real numbers and $a \neq 0$, then there is a unique real number r such that $ar + b = 0$." **Proof**:

- **Existence**: $r = -b/a$ 是一个解 (因为 $a(-b/a) + b = 0$)。
- **Uniqueness**: 假设 s 也是 $as + b = 0$ 的解。则 $ar + b = as + b$，两边减 b 并除以 a (因为 $a \neq 0$)，得 $r = s$。□

1.7.7 7. Proof Strategies (证明策略)

7.1 Proving Universally Quantified Assertions

要证明 $\forall x P(x)$:

- **Exhaustive proof** (穷举法): 论域有限时，逐一验证每个元素。
- **Universal generalization** (全称引入): 选取一个 arbitrary element c，证明 $P(c)$，然后应用 UG。

7.2 常见策略总结

目标	推荐策略
$p \to q$	Direct Proof 或 Proof by Contraposition
p	Proof by Contradiction
$\forall x P(x)$	选任意 c 证明 $P(c) \to$ UG
$\exists x P(x)$	构造显式实例 (constructive) 或反证 (nonconstructive)
$\exists! x P(x)$	存在性 + 唯一性两部分
否定 $\forall x P(x)$	找 counterexample
$p \vee q$ 等	Proof by Cases
$p \leftrightarrow q$	分别证 $p \to q$ 和 $q \to p$

7.3 解题思路

当直接证明困难时:

1. 尝试 **proof by contraposition** (逆否命题可能更容易)
2. 尝试 **proof by contradiction** (假设更大的否定)
3. 尝试 **proof by cases** (分解情况)
4. 尝试证明其逆否命题或等价形式

1.7.8 8. Open Problems (开放问题简介)

数学中许多尚未解决的问题:

- Goldbach's Conjecture: 每个大于 2 的偶数可表示为两个素数之和
- 3x+1 问题 (Collatz Conjecture)
- 这些仍然是猜想 (conjectures)，尚未被证明。

1.7.9 Summary & Exam Tips (小结与考点提示)

考点	说明
Proof by Cases	将命题分解为穷举的 case，分别证明，注意要覆盖所有情况
WLOG	当情形对称时，只证一种，标注"without loss of generality"
Constructive Existence	给出具体实例，如 $1729 = 10^3 + 9^3 = 12^3 + 1^3$
Nonconstructive Existence	不给出具体值，通过推理证明必然存在
Counterexample	证明全称命题为假的最佳方法
Uniqueness Proof	先证存在，再证唯一

各证明方法的选择逻辑:

```
需要证明什么?

 p → q
    前提直接推出结论 → Direct Proof
    ¬q 推出 ¬p 更容易 → Contraposition
    假设 p  ¬q 推出矛盾 → Contradiction

 x P(x)
    能找到具体 x → Constructive
    无法给出具体 x → Nonconstructive

 x P(x) 为假
    找一个 counterexample

 p  q
    分别证明两个方向

 !x P(x)
     存在性 + 唯一性
```

易错提醒:

- Proof by cases 必须覆盖**所有**可能情况，遗漏任何 case 都会导致证明不完整
- WLOG 只在各种情况**真正对称**时才可使用
- Nonconstructive proof 可能让人感觉不" 直接"，但在数学上是完全有效的
- Counterexample 是**否定**全称命题的手段，不是证明手段
- 唯一性证明中，**存在**和**唯一**两者缺一不可

第二章　Chapter 2: Sets, Functions, Sequences (集合、函数、序列)

2.1　Chapter 2.1: Sets (集合)

2.1.1　1. Definition of a Set (集合的定义)

Set (集合): An unordered collection of distinct objects. 一组无序的、互不相同的对象。

- The objects in a set are called **elements (元素)** or **members (成员)** of the set.
- Notation: $a \in A$ means a is an element of set A; $a \notin A$ means a is not an element.

Naive Set Theory (朴素集合论): We use an informal approach to set theory without formal axioms.

2.1.2　2. Describing Sets (描述集合)

Roster Method (列举法)

List all elements inside curly braces . Examples:

- $V = \{a, e, i, o, u\}$ —vowels in English alphabet
- $O = \{1, 3, 5, 7, 9\}$ —odd positive integers less than 10
- $S = \{1, 2, 3, \ldots, 99\}$ —positive integers less than 100
- $S = \{\ldots, -3, -2, -1\}$ —all integers less than 0

Key points:

- Order does **not** matter: $\{a, b, c, d\} = \{b, c, a, d\}$
- Repeated elements do **not** change the set: $\{a, b, c, d\} = \{a, b, c, b, c, d\}$
- Ellipses $(\ldots)$ indicate a clear pattern.

Set-Builder Notation (构造法)

Specify properties that all members must satisfy:

$$S = \{x \mid P(x)\}$$

Examples:

- $S = \{x \mid x \text{ is a positive integer less than } 100\}$
- $O = \{x \in \mathbb{Z}^+ \mid x \text{ is odd and } x < 10\}$
- $\mathbb{Q}^+ = \{x \in \mathbb{R} \mid x = p/q \text{ for some positive integers } p, q\}$

Interval Notation (区间表示法)

- $[a, b] = \{x \mid a \leq x \leq b\}$ —closed interval
- $[a, b) = \{x \mid a \leq x < b\}$
- $(a, b] = \{x \mid a < x \leq b\}$
- $(a, b) = \{x \mid a < x < b\}$ —open interval

2.1.3　3. Important Sets in Mathematics (重要数集)

Notation	Name	Elements
$\mathbb{N}$	Natural numbers (自然数)	$\{0, 1, 2, 3, \ldots\}$
$\mathbb{Z}$	Integers (整数)	$\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
$\mathbb{Z}^+$	Positive integers (正整数)	$\{1, 2, 3, \ldots\}$
$\mathbb{R}$	Real numbers (实数)	—
$\mathbb{R}^+$	Positive real numbers (正实数)	—
$\mathbb{C}$	Complex numbers (复数)	—
$\mathbb{Q}$	Rational numbers (有理数)	—

2.1.4 4. Universal Set and Empty Set (全集与空集)

Universal Set (全集) U: The set containing everything currently under consideration. 上下文中所考虑的所有元素构成的集合。**Empty Set (空集)** $\emptyset$ **(or** $\{\}$**)**: The set with no elements. 不含任何元素的集合。**Venn Diagram (文氏图)**: A pictorial representation of sets and their relationships.

Russell's Paradox (罗素悖论)

Let S be the set of all sets which are **not** members of themselves. The question "Is S a member of itself?" leads to a paradox.

通俗理解：如果 S 包含自己，那它就不该包含自己；如果不包含自己，那它又该包含自己。这是一个逻辑矛盾。

Important Notes

- Sets can be elements of sets: $\{\{1,2,3\}, a, \{b,c\}\}$, $\{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$
- The empty set is **different** from a set containing the empty set: $\emptyset \neq \{\emptyset\}$ （空集和包含空集的集合是两回事！）

2.1.5 5. Subsets (子集)

Definition: Set A is a **subset (子集)** of B, denoted $A \subseteq B$, if and only if every element of A is also an element of B.

$$A \subseteq B \iff \forall x(x \in A \to x \in B)$$

- $\emptyset \subseteq S$ for every set S (空集是任何集合的子集)
- $S \subseteq S$ for every set S (任何集合是自身的子集)

Showing $A \subseteq B$: Show that if $x \in A$, then $x \in B$. **Showing** $A \not\subseteq B$: Find a **counterexample** $x \in A$ with $x \notin B$.

Set Equality (集合相等)

Two sets are equal if and only if they have the same elements:

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

Examples:

- $\{1,3,5\} = \{3,5,1\}$
- $\{1,5,5,5,3,3,1\} = \{1,3,5\}$

Proper Subset (真子集)

If $A \subseteq B$ but $A \neq B$, then A is a **proper subset** of B, denoted $A \subset B$.

$$A \subset B \iff \forall x(x \in A \to x \in B) \wedge \exists x(x \in B \wedge x \notin A)$$

2.1.6 6. Cardinality of Sets (集合的基数 / 势)

Definition: If there are exactly n distinct elements in S (where n is a nonnegative integer), we say S is **finite (有限集)**. Otherwise it is **infinite (无限集)**. **Cardinality (基数/势)**: The number of (distinct) elements in a finite set A, denoted $|A|$. Examples:

- $|\emptyset| = 0$
- $|\{1,2,3\}| = 3$
- $|\{\emptyset\}| = 1$ （注意：这不是空集！）
- The set of integers is infinite.

2.1.7 7. Power Set (幂集)

Definition: The set of all subsets of a set A, denoted $\mathcal{P}(A)$, is called the **power set (幂集)** of A. Example: If $A = \{a,b\}$, then

$$\mathcal{P}(A) = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$$

Key Formula: If a set has n elements, then the cardinality of its power set is 2^n.

$$|\mathcal{P}(A)| = 2^{|A|}$$

2.1.8 8. Tuples (元组)

Ordered n-tuple (有序 n 元组): $(a_1, a_2, \ldots, a_n)$ —an ordered collection where a_1 is the first element, a_2 the second, etc.

- Two n-tuples are equal iff their corresponding elements are equal.
- **Ordered pair (序偶)**: A 2-tuple. $(a,b) = (c,d) \iff a = c$ and $b = d$.

Important: Unlike sets, order **matters** in tuples! $(a,b) \neq (b,a)$ unless $a = b$.

2.1.9 9. Cartesian Product (笛卡尔积)

Definition: The **Cartesian product** of sets A and B, denoted $A \times B$, is the set of all ordered pairs (a,b) where $a \in A$ and $b \in B$.

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

Example: $A = \{a,b\}, B = \{1,2,3\}$

$$A \times B = \{(a,1),(a,2),(a,3),(b,1),(b,2),(b,3)\}$$

General Definition: The Cartesian product of $A_1, A_2, \ldots, A_n$:

$$A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in A_i \text{ for } i = 1, \ldots, n\}$$

Relation (关系): A subset $R \subseteq A \times B$ is called a relation from A to B.

2.1.10 10. Truth Sets of Quantifiers (量词的真值集)

Definition: Given a predicate P and a domain D, the **truth set (真值集)** of P is the set of elements in D for which $P(x)$ is true:

$$\{x \in D \mid P(x)\}$$

Example: The truth set of $P(x)$ where domain is integers and $P(x)$ is "$|x| = 1$" is $\{-1, 1\}$.

2.1.11 Summary / 小结

Concept	English	Chinese
Set	Set	集合
Element	Element	元素
Empty set	$\emptyset$	空集
Universal set	U	全集
Subset	$A \subseteq B$	子集
Proper subset	$A \subset B$	真子集
Cardinality	$\lvert A\rvert$	基数/势
Power set	$\mathcal{P}(A)$	幂集
Ordered pair	(a,b)	序偶
Cartesian product	$A \times B$	笛卡尔积
Truth set	$\{x \mid P(x)\}$	真值集

Exam Tips / 考点提示

1. **空集 vs 含空集的集合**: $\emptyset \neq \{\emptyset\}$，前者是空集（0 个元素），后者是含有一个元素（空集）的集合（1 个元素）。
2. **子集证明**: 要证 $A \subseteq B$，任取 $x \in A$ 推出 $x \in B$。要证 $A = B$，两边互证子集关系。
3. **幂集大小**: 若 $|A| = n$，则 $|\mathcal{P}(A)| = 2^n$，务必牢记。
4. **笛卡尔积**: $|A \times B| = |A| \cdot |B|$，如果 A 和 B 都是有限集。
5. **顺序**: 集合中的元素无序，元组中的元素有序。

2.2 Chapter 2.2: Set Operations (集合运算)

2.2.1 1. Boolean Algebra Connection (布尔代数联系)

Propositional calculus and set theory are both instances of a **Boolean Algebra**. The operators in set theory are analogous to those in propositional calculus.

- All sets are assumed to be subsets of a **universal set** U.
- 命题逻辑中的 $\wedge, \vee, \neg$ 分别对应集合论中的 $\cap, \cup, \overline{(\ )}$。

2.2.2 2. Basic Set Operations (基本集合运算)

Union (并集) $A \cup B$

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$

所有属于 A **或**属于 B 的元素组成的集合。Example: $\{1,2,3\} \cup \{3,4,5\} = \{1,2,3,4,5\}$

Intersection (交集) $A \cap B$

$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$

所有同时属于 A **且**属于 B 的元素组成的集合。Example: $\{1,2,3\} \cap \{3,4,5\} = \{3\}$ **Disjoint (不相交)**: If $A \cap B = \emptyset$, then A and B are said to be **disjoint** (互不相交). Example: $\{1,2,3\} \cap \{4,5,6\} = \emptyset$

Complement (补集) $\overline{A}$ **(or** A^c**)**

$$\overline{A} = \{x \in U \mid x \notin A\}$$

相对于全集 U，所有**不在** A 中的元素组成的集合。Example: If U is positive integers less than 100, the complement of $\{x \mid x > 70\}$ is $\{x \mid x \leq 70\}$.

Difference (差集) $A - B$ **(or** $A \setminus B$**)**

$$A - B = \{x \mid x \in A \wedge x \notin B\} = A \cap \overline{B}$$

所有属于 A **但不在** B 中的元素。Examples (using $U = \{0,1,2,3,4,5,6,7,8,9,10\}$, $A = \{1,2,3,4,5\}$, $B = \{4,5,6,7,8\}$):

- $A - B = \{1,2,3\}$
- $B - A = \{6,7,8\}$

Symmetric Difference (对称差) $A \oplus B$

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

属于 A 或属于 B，但**不同时**属于两者的元素。Example: $A = \{1,2,3,4,5\}, B = \{4,5,6,7,8\}$

$$A \oplus B = \{1,2,3,6,7,8\}$$

2.2.3 3. Inclusion-Exclusion Principle (容斥原理)

For the cardinality of the union of two sets:

$$|A \cup B| = |A| + |B| - |A \cap B|$$

理解: 计算 $A \cup B$ 的元素个数时，先加 $|A|$ 和 $|B|$，再减去重复计算的 $|A \cap B|$。

Example: 统计班里数学专业或计算机专业的学生人数 = 数学专业人数 + 计算机专业人数 - 同时修两个专业的人数。

2.2.4 4. Set Identities (集合恒等式)

Identity Laws (恒等律)

- $A \cup \emptyset = A$
- $A \cap U = A$

Domination Laws (支配律)

- $A \cup U = U$
- $A \cap \emptyset = \emptyset$

Idempotent Laws (幂等律)

- $A \cup A = A$
- $A \cap A = A$

Complementation Law (补律)

- $\overline{(\overline{A})} = A$

Commutative Laws (交换律)

- $A \cup B = B \cup A$
- $A \cap B = B \cap A$

Associative Laws (结合律)

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

Distributive Laws (分配律)

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

De Morgan's Laws (德摩根律)

- $\overline{A \cup B} = \overline{A} \cap \overline{B}$
- $\overline{A \cap B} = \overline{A} \cup \overline{B}$

Absorption Laws (吸收律)

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

Complement Laws (互补律)

- $A \cup \overline{A} = U$
- $A \cap \overline{A} = \emptyset$
- $\overline{U} = \emptyset$
- $\overline{\emptyset} = U$

2.2.5 5. Proving Set Identities (证明集合恒等式)

Three methods:

Method 1: Subset Proof (子集互证)

Prove each side is a subset of the other. **Example: Prove** $\overline{A \cap B} = \overline{A} \cup \overline{B}$ **(Second De Morgan's Law) Part 1:** Show $\overline{A \cap B} \subseteq \overline{A} \cup \overline{B}$

- Let $x \in \overline{A \cap B}$. Then $x \notin A \cap B$.
- So $x \notin A$ or $x \notin B$ (or both).
- If $x \notin A$, then $x \in \overline{A}$, so $x \in \overline{A} \cup \overline{B}$.
- If $x \notin B$, then $x \in \overline{B}$, so $x \in \overline{A} \cup \overline{B}$.
- Therefore $\overline{A \cap B} \subseteq \overline{A} \cup \overline{B}$.

Part 2: Show $\overline{A} \cup \overline{B} \subseteq \overline{A \cap B}$

- Let $x \in \overline{A} \cup \overline{B}$. Then $x \in \overline{A}$ or $x \in \overline{B}$.
- If $x \in \overline{A}$, then $x \notin A$, so $x \notin A \cap B$, so $x \in \overline{A \cap B}$.
- If $x \in \overline{B}$, then $x \notin B$, so $x \notin A \cap B$, so $x \in \overline{A \cap B}$.
- Therefore $\overline{A} \cup \overline{B} \subseteq \overline{A \cap B}$.

Since both subset relations hold, $\overline{A \cap B} = \overline{A} \cup \overline{B}$.

Method 2: Set-Builder Notation + Propositional Logic

Use logical equivalences to transform the set-builder notation.

$$\begin{aligned}
\overline{A \cap B} &= \{x \mid x \notin (A \cap B)\} \\
&= \{x \mid \neg(x \in A \cap B)\} \\
&= \{x \mid \neg(x \in A \wedge x \in B)\} \\
&= \{x \mid \neg(x \in A) \vee \neg(x \in B)\} \\
&= \{x \mid x \notin A \vee x \notin B\} \\
&= \{x \mid x \in \overline{A} \vee x \in \overline{B}\} \\
&= \overline{A} \cup \overline{B}
\end{aligned}$$

Method 3: Membership Table (成员关系表)

Use 1 to indicate an element is in the set, 0 to indicate it is not. **Example: Distributive Law** $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

A	B	C	$B \cap C$	$A \cup (B \cap C)$	$A \cup B$	$A \cup C$	$(A \cup B) \cap (A \cup C)$
1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	0	0	0	1	1	1	1
0	1	1	1	1	1	1	1
0	1	0	0	0	1	0	0
0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0

第 5 列和第 8 列完全相同，证明恒等式成立。

2.2.6 6. Generalized Unions and Intersections (广义并集与交集)

Let $A_1, A_2, \ldots, A_n$ be an indexed collection of sets:

$$\bigcup_{i=1}^{n} A_i = A_1 \cup A_2 \cup \cdots \cup A_n = \{x \mid x \in A_i \text{ for some } i\}$$

$$\bigcap_{i=1}^{n} A_i = A_1 \cap A_2 \cap \cdots \cap A_n = \{x \mid x \in A_i \text{ for all } i\}$$

These are well-defined because union and intersection are **associative**. Example: For $i = 1, 2, \ldots$, let $A_i = \{i, i+1, i+2, \ldots\}$.

2.2.7 7. Computer Representation of Sets (集合的计算机表示)

Using **bit strings (位串)** to represent sets:

1. Specify an arbitrary ordering of the n elements of universal set U.
2. Represent a subset $A \subseteq U$ with a bit string of length n, where the i-th bit is 1 if $a_i \in A$ and 0 if $a_i \notin A$.

Operations:

- **Union**: bitwise OR ($\vee$)
- **Intersection**: bitwise AND ($\wedge$)
- **Complement**: bitwise NOT

Example: $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $A = \{1, 2, 3, 4, 5\}$, $B = \{1, 3, 5, 7, 9\}$

- Bit string for A: `111110000`
- Bit string for B: `101010101`
- $A - B = A \cap \overline{B}$: `111110000` AND `010101010` = `010100000` = $\{2, 4\}$

2.2.8 Summary / 小结

Operation	Notation	Meaning
Union (并集)	$A \cup B$	$x \in A \vee x \in B$
Intersection (交集)	$A \cap B$	$x \in A \wedge x \in B$
Complement (补集)	$\overline{A}$	$x \notin A$
Difference (差集)	$A - B$	$x \in A \wedge x \notin B$
Symmetric Diff (对称差)	$A \oplus B$	$(A - B) \cup (B - A)$

Exam Tips / 考点提示

1. **容斥原理**: $|A \cup B| = |A| + |B| - |A \cap B|$ 是必考公式。若三个集合则有 $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$。
2. **De Morgan's Laws**: 非常常用，记住" 并的补 = 补的交，交的补 = 补的并"。
3. **分配律**: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ 和代数中的分配律形式不同，注意区别。
4. **位串表示法**: 能快速计算并、交、补，考试中遇到具体集合时优先使用。
5. **证明题**: Membership table (真值表法) 是最稳妥的证明恒等式方法。

2.3 Chapter 2.3: Functions (函数)

2.3.1 1. Definition of a Function (函数的定义)

Function (函数): Let A and B be nonempty sets. A **function** f from A to B, denoted $f : A \to B$, is an assignment of each element of A to **exactly one** element of B.

$$f(a) = b \text{ means } b \text{ is the unique element assigned to } a$$

Formally:

$$\forall a (a \in A \to \exists! b (b \in B \wedge f(a) = b))$$

- Functions are sometimes called **mappings (映射)** or **transformations (变换)**.
- A function $f : A \to B$ can also be defined as a subset of $A \times B$ (a relation), where no two ordered pairs have the same first element.

2.3.2 2. Domain, Codomain, Image, Preimage (定义域、陪域、像、原像)

For a function $f : A \to B$:

Term	Notation	Meaning
Domain (定义域)	A	The set of all inputs
Codomain (陪域)	B	The set of all possible outputs
Image (像) of a	$f(a) = b$	The output corresponding to input a
Preimage (原像) of b	a	An input whose output is b
Range (值域)	$f(A)$	The set of all actual outputs: $\{f(a) \mid a \in A\}$

Important: Codomain 是" 可能输出的集合"，Range 是" 实际输出的集合"。Range $\subseteq$ Codomain.

Example

```
A = {a,b,c,d}    B = {x,y,z}
f(a) = x, f(b) = y, f(c) = z, f(d) = z
```

- Domain $= A$, Codomain $= B$
- Image of $d = z$, Preimage(s) of $z = \{c, d\}$
- Range $f(A) = \{x, y, z\}$ (actually all of B here)

Image of a Subset (子集的像)

If $S \subseteq A$, then:

$$f(S) = \{f(s) \mid s \in S\}$$

In the example above: $f(\{a, b, c\}) = \{x, y, z\}$, $f(\{c, d\}) = \{z\}$.

2.3.3 3. Types of Functions (函数的类型)

Injective (One-to-One) —单射

Definition: A function f is **one-to-one** or **injective (单射)** iff $f(a) = f(b)$ implies $a = b$ for all a, b in the domain.

$$f \text{ is injective} \iff \forall a \forall b (f(a) = f(b) \to a = b)$$

等价说法：不同的输入映射到不同的输出。**How to prove injectivity**: Assume $f(a) = f(b)$ and prove $a = b$.

Surjective (Onto) —满射

Definition: A function $f : A \to B$ is **onto** or **surjective (满射)** iff for every element $b \in B$, there is an element $a \in A$ with $f(a) = b$.

$$f \text{ is surjective} \iff \forall b \in B, \exists a \in A \text{ such that } f(a) = b$$

等价说法：值域等于陪域，即 $f(A) = B$。**How to prove surjectivity**: For an arbitrary $b \in B$, find (construct) an $a \in A$ such that $f(a) = b$.

Bijective (One-to-One Correspondence) —双射

Definition: A function f is a **one-to-one correspondence** or a **bijection (双射)** if it is **both** injective and surjective.

2.3.4 4. Examples and Counterexamples

Example 1: $f : \{a, b, c, d\} \to \{1, 2, 3\}$ with $f(a) = 3, f(b) = 2, f(c) = 1, f(d) = 3$

- Is f onto? **Yes** —all three elements of codomain are images.
- If codomain were $\{1, 2, 3, 4\}$, f would **not** be onto.

Example 2: $f(x) = x^2$ from $\mathbb{Z}$ to $\mathbb{Z}$

- Not onto: no integer x with $x^2 = -1$.
- Not one-to-one: $f(2) = f(-2) = 4$ but $2 \neq -2$.

2.3.5 5. Inverse Functions (反函数)

Definition: Let f be a **bijection** from A to B. Then the **inverse** of f, denoted f^{-1}, is the function from B to A defined by:

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

No inverse exists unless f is a bijection. Why? - If f is not injective, then f^{-1} would have to map one output to multiple inputs —not a function. - If f is not surjective, then some elements of B have no preimage —f^{-1} would be undefined for them.

Examples:

1. $f : \{a, b, c\} \to \{1, 2, 3\}$, $f(a) = 2, f(b) = 3, f(c) = 1$ - f is bijective. $f^{-1}(1) = c, f^{-1}(2) = a, f^{-1}(3) = b$
2. $f : \mathbb{Z} \to \mathbb{Z}$, $f(x) = x + 1$ - f is bijective. $f^{-1}(y) = y - 1$
3. $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2$ - f is **not** invertible (not one-to-one, e.g., $f(2) = f(-2) = 4$).

2.3.6 6. Function Composition (函数组合)

Definition: Let $f : B \to C$ and $g : A \to B$. The **composition** of f with g, denoted $f \circ g$, is the function from A to C defined by:

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

注意顺序: $f \circ g$ 表示先应用 g，再应用 f。

Examples:

1. $f(x) = x + 1$, $g(x) = x^2$ - $(f \circ g)(x) = f(g(x)) = f(x^2) = x^2 + 1$ - $(g \circ f)(x) = g(f(x)) = g(x + 1) = (x + 1)^2 = x^2 + 2x + 1$
2. $g : \{a, b, c\} \to \{a, b, c\}$, $g(a) = b, g(b) = c, g(c) = a$ $f : \{a, b, c\} \to \{1, 2, 3\}$, $f(a) = 3, f(b) = 2, f(c) = 1$ - $f \circ g$: $(f \circ g)(a) = f(g(a)) = f(b) = 2$, $(f \circ g)(b) = f(g(b)) = f(c) = 1$, $(f \circ g)(c) = f(g(c)) = f(a) = 3$ - $g \circ f$ is **not defined** because range of f ($\{1, 2, 3\}$) is not a subset of domain of g ($\{a, b, c\}$).
3. $f(x) = 2x + 3$, $g(x) = 3x + 2$ (both $\mathbb{Z} \to \mathbb{Z}$) - $(f \circ g)(x) = f(g(x)) = f(3x + 2) = 2(3x + 2) + 3 = 6x + 7$ - $(g \circ f)(x) = g(f(x)) = g(2x + 3) = 3(2x + 3) + 2 = 6x + 11$

2.3.7 7. Graphs of Functions (函数图像)

The **graph** of a function $f : A \to B$ is the set of ordered pairs:

$$\{(a, b) \mid a \in A \text{ and } f(a) = b\}$$

Examples:

- $f(x) = x^2$ from $\mathbb{Z}$ to $\mathbb{Z}$: graph is $\{(n, n^2) \mid n \in \mathbb{Z}\}$
- $f(n) = 2n + 1$ from $\mathbb{Z}$ to $\mathbb{Z}$: graph is $\{(n, 2n + 1) \mid n \in \mathbb{Z}\}$

2.3.8 8. Floor and Ceiling Functions (取整函数)

Floor Function $\lfloor x \rfloor$

The **floor** of x is the **largest integer** less than or equal to x.

$$\lfloor x \rfloor = \max\{n \in \mathbb{Z} \mid n \leq x\}$$

" 向下取整": 不超过 x 的最大整数。

Ceiling Function $\lceil x \rceil$

The **ceiling** of x is the **smallest integer** greater than or equal to x.

$$\lceil x \rceil = \min\{n \in \mathbb{Z} \mid n \geq x\}$$

" 向上取整": 不小于 x 的最小整数。

x	$\lfloor x \rfloor$	$\lceil x \rceil$
2.3	2	3
-2.3	-3	-2
5	5	5
0.5	0	1
-0.5	-1	0

Proving Properties of Floor Functions

Example: Prove that for any real number x, $\lfloor 2x \rfloor = \lfloor x \rfloor + \lfloor x + 1/2 \rfloor$. **Solution**: Let $x = n + \varepsilon$, where n is an integer and $0 \leq \varepsilon < 1$.

Case 1: $\varepsilon < 1/2$

- $2x = 2n + 2\varepsilon$ and $\lfloor 2x \rfloor = 2n$ (since $0 \leq 2\varepsilon < 1$)
- $\lfloor x + 1/2 \rfloor = \lfloor n + (1/2 + \varepsilon) \rfloor = n$ (since $0 \leq 1/2 + \varepsilon < 1$)
- Hence $\lfloor 2x \rfloor = 2n$ and $\lfloor x \rfloor + \lfloor x + 1/2 \rfloor = n + n = 2n$.

Case 2: $\varepsilon \geq 1/2$

- $2x = 2n + 2\varepsilon = (2n + 1) + (2\varepsilon - 1)$ and $\lfloor 2x \rfloor = 2n + 1$ (since $0 \leq 2\varepsilon - 1 < 1$)
- $\lfloor x + 1/2 \rfloor = \lfloor n + (1/2 + \varepsilon) \rfloor = \lfloor n + 1 + (\varepsilon - 1/2) \rfloor = n + 1$ (since $0 \leq \varepsilon - 1/2 < 1$)
- Hence $\lfloor 2x \rfloor = 2n + 1$ and $\lfloor x \rfloor + \lfloor x + 1/2 \rfloor = n + (n + 1) = 2n + 1$.

Both cases satisfy the equation, so the statement is proved.

2.3.9 9. Factorial Function (阶乘)

Definition: $f : \mathbb{N} \to \mathbb{Z}^+$, denoted $f(n) = n!$, is the product of the first n positive integers.

$$n! = 1 \cdot 2 \cdot 3 \cdots (n - 1) \cdot n, \quad 0! = 1$$

Examples:

- $1! = 1$
- $2! = 1 \cdot 2 = 2$
- $6! = 720$
- $20! = 2,432,902,008,176,640,000$

Stirling's Formula (approximation for large n):

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

2.3.10 10. Partial Functions (偏函数, optional)

Definition: A **partial function** f from A to B assigns each element a in a **subset** of A (called the **domain of definition**) to a unique element b in B.

- f is **undefined** for elements in A not in the domain of definition.
- When the domain of definition equals A, we say f is a **total function (全函数)**.

Example: $f : \mathbb{Z} \to \mathbb{R}$ where $f(n) = \sqrt{n}$ is a partial function. Domain of definition is nonnegative integers. f is undefined for negative integers.

2.3.11 Summary / 小结

Concept	English	Chinese
Domain	Domain	定义域
Codomain	Codomain	陪域
Range / Image	Range	值域
One-to-one	Injective	单射
Onto	Surjective	满射
One-to-one correspondence	Bijective	双射
Inverse function	f^{-1}	反函数
Composition	$f \circ g$	复合函数
Floor	$\lfloor x \rfloor$	向下取整
Ceiling	$\lceil x \rceil$	向上取整
Factorial	$n!$	阶乘

Exam Tips / 考点提示

1. **判断函数类型**: 单射 ($f(a) = f(b) \to a = b$)，满射 (range = codomain)，双射 (both)。
2. **反函数存在的充要条件**: 函数必须是**双射**。
3. **复合函数的顺序**: $(f \circ g)(x) = f(g(x))$，先 g 后 f。$f \circ g$ 和 $g \circ f$ 一般不同。
4. **复合函数定义条件**: range of g 必须是 domain of f 的子集。
5. **Floor/Ceiling 证明技巧**: 将 x 分解为整数部分 + 小数部分 $x = n + \varepsilon$，分情况讨论 $\varepsilon < 1/2$ 和 $\varepsilon \geq 1/2$。
6. **负数的取整**: $\lfloor -2.3 \rfloor = -3$ (小于 -2.3 的最大整数是 -3)，$\lceil -2.3 \rceil = -2$。

2.4 Chapter 2.4: Sequences and Summations (序列与求和)

2.4.1 1. Sequences (序列)

Definition: A **sequence (序列)** is a function from a subset of the integers (usually $\{0, 1, 2, \ldots\}$ or $\{1, 2, 3, \ldots\}$) to a set S.

- We use the notation a_n to denote the image of integer n. a_n is called a **term (项)** of the sequence.
- The sequence is often denoted $\{a_n\}$.

Example: Consider the sequence $\{a_n\}$ where $a_n = 1/n$.

$$a_1 = 1,\ a_2 = \frac{1}{2},\ a_3 = \frac{1}{3},\ \ldots$$

2.4.2 2. Geometric Progression (等比数列 / 几何级数)

Definition: A **geometric progression (等比数列)** is a sequence of the form:

$$a,\ ar,\ ar^2,\ ar^3,\ \ldots,\ ar^n,\ \ldots$$

where the **initial term (初项)** a and the **common ratio (公比)** r are real numbers. Examples:

1. $a = 1,\ r = \frac{1}{3}$: $1, \frac{1}{3}, \frac{1}{9}, \frac{1}{27}, \ldots$
2. $a = -1,\ r = 2$: $-1, -2, -4, -8, \ldots$
3. $a = 6,\ r = 5$: $6, 30, 150, 750, \ldots$

2.4.3 3. Arithmetic Progression (等差数列 / 算术级数)

Definition: An **arithmetic progression (等差数列)** is a sequence of the form:

$$a,\ a + d,\ a + 2d,\ a + 3d,\ \ldots,\ a + nd,\ \ldots$$

where the initial term a and the **common difference (公差)** d are real numbers. Examples:

1. $a = -1,\ d = 4$: $-1, 3, 7, 11, 15, \ldots$
2. $a = 7,\ d = -3$: $7, 4, 1, -2, -5, \ldots$
3. $a = 1,\ d = 2$: $1, 3, 5, 7, 9, \ldots$

2.4.4 4. Strings (字符串)

Definition: A **string (字符串)** is a finite sequence of characters from a finite set (an **alphabet**).

- The **empty string (空串)** is represented by λ.
- The string `abcde` has length 5.

字符串在计算机科学中非常重要，如二进制位串。

2.4.5 5. Recurrence Relations (递推关系)

Definition: A **recurrence relation (递推关系)** for the sequence $\{a_n\}$ is an equation that expresses a_n in terms of one or more of the previous terms $(a_0, a_1, \ldots, a_{n-1})$, for all integers $n \geq n_0$.

- A sequence is called a **solution** of a recurrence relation if its terms satisfy the recurrence relation.
- The **initial conditions (初始条件)** specify the terms that precede the first term where the recurrence relation takes effect.

Example 1

Let $\{a_n\}$ satisfy $a_n = a_{n-1} + 3$ for $n = 1, 2, 3, \ldots$ with $a_0 = 2$.

$$a_1 = 2 + 3 = 5$$

$$a_2 = 5 + 3 = 8$$

$$a_3 = 8 + 3 = 11$$

Example 2

Let $\{a_n\}$ satisfy $a_n = a_{n-1} - a_{n-2}$ for $n = 2, 3, 4, \ldots$ with $a_0 = 3$, $a_1 = 5$.

$$a_2 = a_1 - a_0 = 5 - 3 = 2$$

$$a_3 = a_2 - a_1 = 2 - 5 = -3$$

2.4.6 6. Fibonacci Sequence (斐波那契数列)

Definition: The **Fibonacci sequence** $\{f_n\}$ is defined by:

- **Initial conditions**: $f_0 = 0,\ f_1 = 1$
- **Recurrence relation**: $f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$

Example: Find f_2, f_3, f_4, f_5, f_6.

$$\begin{aligned} f_2 &= f_1 + f_0 = 1 + 0 = 1 \\ f_3 &= f_2 + f_1 = 1 + 1 = 2 \\ f_4 &= f_3 + f_2 = 2 + 1 = 3 \\ f_5 &= f_4 + f_3 = 3 + 2 = 5 \\ f_6 &= f_5 + f_4 = 5 + 3 = 8 \end{aligned}$$

Fibonacci sequence: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$

2.4.7 7. Solving Recurrence Relations (求解递推关系)

Finding a formula for the n-th term of the sequence generated by a recurrence relation is called **solving the recurrence relation**. Such a formula is called a **closed formula (闭公式 / 通项公式)**.

Method 1: Forward Substitution (正向代入)

Start from initial condition and work forward. **Example**: $\{a_n\}$ with $a_n = a_{n-1} + 3$, $a_1 = 2$.

$$\begin{aligned} a_2 &= 2 + 3 \\ a_3 &= (2 + 3) + 3 = 2 + 3 \cdot 2 \\ a_4 &= (2 + 2 \cdot 3) + 3 = 2 + 3 \cdot 3 \\ &\vdots \\ a_n &= 2 + 3(n - 1) \end{aligned}$$

所以 $a_n = 2 + 3(n - 1) = 3n - 1$.

Method 2: Backward Substitution (反向代入)

Start from a_n and substitute backward until reaching initial condition. **Example**: Same sequence $a_n = a_{n-1} + 3$, $a_1 = 2$.

$$\begin{aligned} a_n &= a_{n-1} + 3 \\ &= (a_{n-2} + 3) + 3 = a_{n-2} + 3 \cdot 2 \\ &= (a_{n-3} + 3) + 3 \cdot 2 = a_{n-3} + 3 \cdot 3 \\ &\vdots \\ &= a_1 + 3(n - 1) = 2 + 3(n - 1) \end{aligned}$$

2.4.8 8. Financial Application (金融应用)

Example: Deposit $10,000.00$ in a savings account yielding 11% per year with interest compounded annually. How much after 30 years? Let P_n be the amount after n years.

$$P_n = P_{n-1} + 0.11P_{n-1} = 1.11P_{n-1}, \quad P_0 = 10,000$$

Forward substitution:

$$\begin{aligned} P_1 &= 1.11P_0 \\ P_2 &= 1.11P_1 = (1.11)^2 P_0 \\ P_3 &= 1.11P_2 = (1.11)^3 P_0 \\ &\vdots \\ P_n &= (1.11)^n P_0 \end{aligned}$$

$$P_{30} = (1.11)^{30} \cdot 10,000 \approx \$228,992.97$$

(Can be proved by induction —Chapter 5.)

2.4.9 9. Special Integer Sequences (特殊整数序列, optional)

Given a few terms of a sequence, try to:

1. Conjecture a formula, recurrence relation, or some other rule.
2. Useful questions to ask:

- Are there repeated terms?
- Can you obtain a term from the previous term by adding or multiplying?
- Can you obtain a term by combining previous terms?
- Are there cycles?

Examples:

1. $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$ —Geometric progression, $a_n = 1/2^{n-1}$ (or $a_n = 1/2^n$ starting from $n = 0$).
2. $1, 3, 5, 7, 9$ —Arithmetic progression, $a_n = 2n + 1$ (if n starts at 0).
3. $1, -1, 1, -1, 1$ —Alternating, $a_n = (-1)^n$ (if n starts at 0) or $a_n = (-1)^{n+1}$ (if n starts at 1).
4. $2, 5, 10, 17, 26, 37, \ldots$ —Pattern: $a_n = n^2 + 1$.
5. $1, 7, 25, 79, 241, 727, 2185, \ldots$ —After checking ratios (approx 3), conjecture $a_n = 3^n - 2$.

2.4.10 10. Summations (求和)

Summation Notation

The sum of terms $a_m, a_{m+1}, \ldots, a_n$ is denoted:

$$\sum_{j=m}^{n} a_j$$

- j is called the **index of summation (求和指标)**.
- m is the **lower limit (下限)** and n is the **upper limit (上限)**.

Examples:

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

$$\sum_{j=1}^{n} j = 1 + 2 + 3 + \cdots + n$$

More generally, for a set S:

$$\sum_{s \in S} f(s)$$

Product Notation (连乘)

$$\prod_{j=m}^{n} a_j = a_m \cdot a_{m+1} \cdots a_n$$

2.4.11 11. Geometric Series (几何级数求和)

The sum of terms of a geometric progression:

$$\sum_{j=0}^{n} ar^j = a + ar + ar^2 + \cdots + ar^n$$

Formula:

$$\sum_{j=0}^{n} ar^j = \begin{cases} a \cdot \frac{r^{n+1}-1}{r-1}, & r \neq 1 \\ a(n+1), & r = 1 \end{cases}$$

Proof: Let $S_n = \sum_{j=0}^{n} ar^j$.

$$\begin{aligned} S_n &= a + ar + ar^2 + \cdots + ar^n \\ rS_n &= ar + ar^2 + ar^3 + \cdots + ar^{n+1} \\ S_n - rS_n &= a - ar^{n+1} \\ S_n(1-r) &= a(1-r^{n+1}) \\ \therefore S_n &= a \cdot \frac{1-r^{n+1}}{1-r} = a \cdot \frac{r^{n+1}-1}{r-1} \quad (r \neq 1) \end{aligned}$$

2.4.12 12. Useful Summation Formulae (常用求和公式)

Formula	Closed Form
$\sum_{k=1}^{n} ar^{k-1}$ (geometric series)	$a \cdot \frac{r^n-1}{r-1},\ r \neq 1$
$\sum_{k=1}^{n} 1$	n
$\sum_{k=1}^{n} k$	$\frac{n(n+1)}{2}$
$\sum_{k=1}^{n} k^2$	$\frac{n(n+1)(2n+1)}{6}$
$\sum_{k=1}^{n} k^3$	$\frac{n^2(n+1)^2}{4}$
$\sum_{k=0}^{n} r^k,\ r \neq 1$	$\frac{r^{n+1}-1}{r-1}$
$\sum_{k=0}^{\infty} x^k,\ \lvert x\rvert < 1$	$\frac{1}{1-x}$
$\sum_{k=1}^{\infty} kx^{k-1},\ \lvert x\rvert < 1$	$\frac{1}{(1-x)^2}$

2.4.13 Summary / 小结

Concept	Formula
Arithmetic progression (等差数列)	$a_n = a + (n-1)d$
Geometric progression (等比数列)	$a_n = ar^{n-1}$
Fibonacci sequence	$f_n = f_{n-1} + f_{n-2},\ f_0 = 0,\ f_1 = 1$
Sum of first n integers	$\frac{n(n+1)}{2}$
Geometric series sum	$a \cdot \frac{r^n-1}{r-1}$

Exam Tips / 考点提示

1. **递推关系**: 给定初始条件和递推公式，能求出前几项。
2. **Closed formula**: 会用 forward/backward substitution 方法求解递推关系得到通项公式。
3. **等差数列和等比数列**的公式必须熟练记忆。
4. **求和公式**: $\sum k = n(n+1)/2$ 和 $\sum k^2 = n(n+1)(2n+1)/6$ 是最常用的两个求和公式。
5. **几何级数求和公式**的推导过程（乘公比错位相减）要掌握。
6. **Fibonacci 数列**: 初始条件 $f_0 = 0, f_1 = 1$，递推 $f_n = f_{n-1} + f_{n-2}$。

2.5 Chapter 2.5: Cardinality of Sets (集合的基数 / 势)

2.5.1 1. Definition of Cardinality (基数的定义)

Definition: The cardinality of a set A is equal to the cardinality of a set B, denoted $|A| = |B|$, if and only if there is a **one-to-one correspondence** (i.e., a **bijection**) from A to B.

$$|A| = |B| \iff \exists \text{ bijection } f : A \to B$$

- If there is an **injection** (one-to-one function) from A to B, then $|A| \leq |B|$.
- If $|A| \leq |B|$ and $|A| \neq |B|$, then $|A| < |B|$.

核心思想: 集合的大小通过是否能建立一一对应来判断，而不仅仅是数元素个数。

2.5.2 2. Countable Sets (可数集)

Definition: A set that is either **finite** or has the **same cardinality as the set of positive integers** $\mathbb{Z}^+$ is called **countable (可数的)**. A set that is **not** countable is **uncountable (不可数的)**.

- When an infinite set is countable (countably infinite), its cardinality is denoted $\aleph_0$ (aleph-null, 阿列夫零): $|S| = \aleph_0$.
- $\aleph$ is aleph, the first letter of the Hebrew alphabet.

Key insight: An infinite set is countable **iff** it is possible to **list** the elements of the set in a sequence (indexed by positive integers).

可数的无限集意味着我们可以将它的元素排成一个序列 $a_1, a_2, a_3, \ldots$，每个元素都会在序列中出现（允许重复）。

2.5.3 3. Hilbert's Grand Hotel (希尔伯特大旅馆)

David Hilbert 提出的著名思想实验，帮助理解可数无限集的性质。

The paradox: A hotel with **countably infinite** rooms (Room 1, Room 2, Room 3, ...), all occupied. Can we accommodate a new guest?

Solution: Yes! Move: - Guest in Room 1 → Room 2 - Guest in Room 2 → Room 3 - Guest in Room n → Room $n+1$

This frees up Room 1 for the new guest, and all current guests still have rooms.

Even more surprising: The hotel can also accommodate: - A **countable number** of new guests - All guests on a **countable number of buses**, each containing a countable number of guests

启示: 无限集可以和它的真子集一一对应，这与有限集完全不同！

2.5.4 4. Examples of Countable Sets (可数集的例子)

Example 1: Positive Even Integers

Show that the set of positive even integers $E = \{2, 4, 6, 8, \ldots\}$ is countable. **Solution**: Define $f : \mathbb{N} \to E$ by $f(n) = 2n$.

$$\begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & \ldots \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ 2 & 4 & 6 & 8 & 10 & 12 & \ldots \end{array}$$

- **One-to-one**: If $2n = 2m$, then $n = m$.
- **Onto**: For any even positive integer $t = 2k$, $f(k) = t$.

Since f is a bijection, $|E| = |\mathbb{N}| = \aleph_0$. The set is countably infinite.

Example 2: Integers $\mathbb{Z}$

Show that the set of integers $\mathbb{Z}$ is countable. **Solution**: List in a sequence: $0, 1, -1, 2, -2, 3, -3, \ldots$ Or define a bijection $f : \mathbb{N} \to \mathbb{Z}$:

- When n is even: $f(n) = n/2$
- When n is odd: $f(n) = -(n+1)/2$

n	0	1	2	3	4	5	6	...
$f(n)$	0	-1	1	-2	2	-3	3	...

This is a bijection, so $|\mathbb{Z}| = \aleph_0$.

Example 3: Ordered Pairs of Integers

The set of ordered pairs of integers is **countably infinite**. A one-to-one correspondence can be established (e.g., by traversing along diagonals in a 2D grid).

这个结论可能反直觉：二维网格可以一一映射到一维序列。

2.5.5 5. Properties of Cardinality of Infinite Sets (无限集基数的性质)

For every set A and B, the following hold: **(a) Trichotomy (三分律)**: Exactly one of the following holds:

$$|A| = |B|, \quad |A| < |B|, \quad |A| > |B|$$

(b) Cantor-Bernstein Theorem (康托尔-伯恩斯坦定理):

$$\text{If } |A| \leq |B| \text{ and } |B| \leq |A|, \text{ then } |A| = |B|$$

实用技巧: 要证 $|A| = |B|$，通常先证 $|A| \leq |B|$，再证 $|B| \leq |A|$。（构造两个单射即可，不需要直接构造双射。）

2.5.6 6. Uncountable Sets: Real Numbers (不可数集：实数)

The set of real numbers $\mathbb{R}$ is **uncountable**. This was proved by Cantor using his famous **diagonalization argument (对角线论证)**.

Cantor's Diagonalization (康托尔对角线法): 假设 $(0, 1)$ 区间内的实数可数，列出所有小数。然后构造一个在对角线上每一位都不同的新数，这个数不在原列表中，产生矛盾。

$\mathbb{R}$ 的基数被称为 $\aleph_1$ 或 c (continuum, 连续统的势)。

2.5.7 Summary / 小结

Concept	Meaning
$\lvert A\rvert = \lvert B\rvert$	Exists a bijection $f : A \to B$
$\lvert A\rvert \le \lvert B\rvert$	Exists an injection $f : A \to B$
Countable (可数)	Finite or $\lvert S\rvert = \aleph_0$
$\aleph_0$ (aleph-null)	Cardinality of $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$, positive even integers, etc.
Uncountable (不可数)	e.g., $\mathbb{R}$

Exam Tips / 考点提示

1. **判断可数性的基本方法**: 寻找一个从 $\mathbb{N}$ 或 $\mathbb{Z}^+$ 到该集合的双射，或证明可以按序列列出所有元素。
2. $\mathbb{Z}$ **是可数的!** 序列: $0, 1, -1, 2, -2, 3, -3, \ldots$。
3. **正偶数集合可数**: $f(n) = 2n$。
4. **有理数 $\mathbb{Q}$ 可数**: 按对角线顺序遍历 $\mathbb{Q}^+$（课件未详细展开，但是一个经典结论）。
5. **实数 $\mathbb{R}$ 不可数**: 康托尔对角线法证明。
6. **Cantor-Bernstein 定理**: 证明两集合基数相等时的利器——只需构造两个方向的单射。
7. **无限集的反直觉性质**: 可以与自己的真子集一一对应（如 $\mathbb{N}$ 与正偶数一一对应）。

第三章　Chapter 3: Algorithms (算法)

3.1 Chapter 3.1: Algorithms (算法)

3.1.1 1. Definition of an Algorithm (算法的定义)

Definition: An **algorithm** is a **finite set of precise instructions** for performing a computation or for solving a problem.

算法是一个**有限的精确指令集**，用于执行计算或解决问题。

词源：得名于波斯数学家 **Abu Ja'far Mohammed Ibin Musa Al-Khowarizmi** (780-850)。

3.1.2 2. Properties of Algorithms (算法的性质)

一个算法必须具备以下六个性质:

Property (性质)	Meaning (含义)
Input (输入)	从一个指定的集合中获取输入值
Output (输出)	为每组输入产生正确的输出值（即问题的解）
Correctness (正确性)	对每组输入都应产生正确的输出
Finiteness (有限性)	对任何输入，算法必须在**有限步**之后结束
Effectiveness (有效性)	每一步都必须在有限时间内被正确地执行
Generality (通用性)	算法应该适用于**同一类问题的所有实例**

3.1.3 3. Specifying Algorithms (算法的描述方式)

算法可以用多种方式描述:

- **自然语言** (English description)
- **伪代码** (Pseudocode) ——介于自然语言和编程语言之间的中间步骤
- 编程语言（如 C++, Java）

本课程使用 **pseudocode**（伪代码），详情见教材 Appendix 3。

3.1.4 4. Example: Finding the Maximum Element (找最大值)

Example 1: 描述一个算法，找到有限整数序列中的最大值。**思路**：将临时最大值设为第一个元素，然后遍历后续元素，遇到更大的就更新。**Pseudocode**:

```
procedure max(a1, a2, ...., an: integers)
    max := a1
    for i := 2 to n
        if max < ai then max := ai
    return max {max is the largest element}
```

分析:

- 该算法比较次数为 `n-1` 次（for 循环中每次比较一次）
- 满足所有算法性质（输入、输出、正确性、有限性、有效性、通用性）

3.1.5 5. Searching Problems (搜索问题)

Definition: The general **searching problem** is to locate an element `x` in the list of distinct elements `a1, a2, ..., an`, or determine that it is not in the list (返回 `0`).

5.1 Linear Search (线性搜索)

思路：从列表开头逐个比较元素，直到找到目标或遍历完整个列表。**Pseudocode**:

```
procedure linear search(x: integer, a1, a2, ..., an: distinct integers)
    i := 1
    while (i <= n and x != ai)
        i := i + 1
    if i <= n then location := i
    else location := 0
    return location {location is the subscript of the term that equals x, or is 0 if x is not found}
```

特点:

- 不需要列表有序
- 最坏情况需要检查所有 n 个元素
- 时间复杂度：$\Theta(n)$

5.2 Binary Search (二分搜索)

前提：列表必须是**升序排列**的 (increasing order)。**思路**:

1. 将目标与中间元素比较
2. 若目标大于中间元素，搜索上半部分
3. 否则搜索下半部分（包括中间位置）
4. 重复直到列表大小为 1
5. 若找到匹配则返回位置，否则返回 0

Pseudocode:

```
procedure binary search(x: integer, a1, a2, ..., an: increasing integers)
    i := 1          {i is left endpoint of interval}
    j := n          {j is right endpoint of interval}
    while i < j
        m := floor((i + j) / 2)
        if x > am then i := m + 1
        else j := m
    if x = ai then location := i
    else location := 0
    return location {location is the subscript i of the term ai equal to x, or 0 if x is not found}
```

Example: 在列表 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22 中搜索 19。**步骤**:

1. 列表有 16 个元素，中间位置是第 8 个（值为 10）。$19 > 10$，搜索位置 9 16。
2. 位置 9 16 的中间是第 12 个（值为 16）。$19 > 16$，搜索位置 13 16。
3. 位置 13 16 的中间是第 14 个（值为 19）。$19 \not> 19$，搜索位置 13 14。
4. 位置 13 14 的中间是第 13 个（值为 18）。$19 > 18$，搜索位置 14 14。
5. 此时只有一个元素，循环结束。`19 = 19`，返回位置 14。

特点:

- 时间复杂度：$\Theta(\log n)$，远优于线性搜索

3.1.6 6. Sorting Problems (排序问题)

排序是将列表中的元素按**递增顺序**排列。

6.1 Bubble Sort (冒泡排序)

思路：多次遍历列表，每次比较相邻元素，顺序不对就交换。每轮遍历将当前最大元素”冒泡”到正确位置。**Pseudocode**:

```
procedure bubblesort(a1, ..., an: real numbers with n >= 2)
    for i := 1 to n - 1
        for j := 1 to n - i
            if aj > aj+1 then interchange aj and aj+1
    {a1, ..., an is now in increasing order}
```

Example: 对 3 2 4 1 5 排序的步骤:

```
第一轮: 2 3 1 4 5   (5 已就位)
第二轮: 2 1 3 4 5   (4 已就位)
第三轮: 1 2 3 4 5   (3 已就位)
第四轮: 1 2 3 4 5   (2 已就位)
```

复杂度：比较次数为 $\frac{n(n-1)}{2}$，即 $\Theta(n^2)$。

6.2 Insertion Sort (插入排序)

思路：从第二个元素开始，将每个元素插入到它前面已排序序列的正确位置。**Pseudocode**:

```
procedure insertion sort(a1, ..., an: real numbers with n >= 2)
    for j := 2 to n
        i := 1
        while aj > ai
            i := i + 1
        m := aj
        for k := 0 to j - i - 1
            aj-k := aj-k-1
        ai := m
    {Now a1, ..., an is in increasing order}
```

Example: 对 3 2 4 1 5 排序的步骤:

```
i.  2 3 4 1 5    (前两个交换)
ii. 2 3 4 1 5    (第三个元素保持不动)
iii. 1 2 3 4 5   (第四个元素插入到开头)
iv. 1 2 3 4 5    (第五个元素保持不动)
```

复杂度：最坏情况比较次数为 $\frac{n(n-1)}{2}$，即 $\Theta(n^2)$。

3.1.7 7. Greedy Algorithms (贪心算法)

Greedy algorithm: 在每一步都做出当前”最佳”选择。

注意：局部最优选择**不一定**导致全局最优解，需要证明或找反例。

7.1 Greedy Change-Making Algorithm (找零问题)

Problem: 用最少的硬币数凑出 n 美分。硬币面额：quarters (25¢), dimes (10¢), nickels (5¢), pennies (1¢)。**Idea**: 每一步选择不超过剩余金额的最大面额硬币。**Pseudocode**:

```
procedure change(c1, c2, ..., cr: values of coins, where c1 > c2 > ... > cr; n: positive integer)
    for i := 1 to r
        di := 0  {di counts the coins of denomination ci}
        while n >= ci
            di := di + 1
            n := n - ci
    {di is the number of coins of denomination ci}
```

Example: 使用 US 硬币凑 67 美分:

1. $67 - 25 = 42$，用 1 个 quarter
2. $42 - 25 = 17$，再用 1 个 quarter
3. $17 - 10 = 7$，用 1 个 dime
4. $7 - 5 = 2$，用 1 个 nickel
5. $2 - 1 = 1$，用 1 个 penny
6. $1 - 1 = 0$，再用 1 个 penny

共使用 6 枚硬币。**Lemma 1** (US 硬币最优性引理)：用最少数量的 US 硬币找零时:

- 最多使用 2 个 dimes，1 个 nickel，4 个 pennies
- 不能有 2 个 nickels（可换成 1 个 dime）
- dimes、nickels 和 pennies 的总值不超过 24¢

Theorem: US 硬币的贪心算法产生最优解（使用最少硬币数）。**反例**：如果面额为 1, 7, 10 Asiaron:

- 贪心凑 15：$10 + 1 + 1 + 1 + 1 + 1 = 6$ 枚硬币
- 最优解：$7 + 7 + 1 = 3$ 枚硬币
- 此时贪心算法**不是最优的**。

7.2 Greedy Scheduling (任务调度)

Problem: 在一间演讲厅中安排尽可能多的演讲，演讲有开始和结束时间，不能重叠。**失败策略**:

- 选择开始最早的——不行
- 选择持续时间最短的——不行

正确策略：选择**结束时间最早**且与已选任务兼容的任务。**Pseudocode**:

```
procedure schedule(s1, s2, ..., sn: start times, e1, e2, ..., en: end times)
    sort talks by finish time and reorder so that e1 <= e2 <= ... <= en
    S := empty set
    for j := 1 to n
        if talk j is compatible with S then
            S := S union {talk j}
    return S {S is the set of talks scheduled}
```

3.1.8 8. Halting Problem (停机问题)

Halting Problem: 能否编写一个程序，判断任意程序 P 在给定输入 I 下是否会终止（halt）?

结论：**不存在**这样的程序。停机问题是**不可解的 (unsolvable)**。

证明思路（反证法）：1. 假设存在过程 `H(P, I)`，能判断程序 P 在输入 I 下是否停机 2. 构造程序 `K(P)`：- 若 `H(P, P)` 输出”loops forever”，则 `K(P)` 停机 - 若 `H(P, P)` 输出”halt”，则 `K(P)` 无限循环 3. 调用 `K(K)`：- 如果 `H(K, K)` 说”loops forever”，那么 `K(K)` 停机——**矛盾** - 如果 `H(K, K)` 说”halt”，那么 `K(K)` 无限循环——**矛盾** 4. 因此，停机问题不可解。

伪代码表示：

```
Bool DOES-HALT(P, I) {
    if (P will stop when run with input I)   // P(I) will stop
        return 1
    else
        return 0
}

Bool SELF-HALT(program) {
    if(DOES-HALT(program, program))
        infinite loop
    else
        halt
}
```

那么 `DOES-HALT(SELF-HALT, SELF-HALT)` 是什么？这将导致矛盾。

3.1.9 9. Summary of Important Algorithms (重要算法总结)

Algorithm	Type	Key Idea	Time Complexity
Max Finding	Search	遍历更新最大值	$\Theta(n)$
Linear Search	Search	逐个比较	$\Theta(n)$
Binary Search	Search	折半查找（要求有序）	$\Theta(\log n)$
Bubble Sort	Sort	相邻交换，冒泡	$\Theta(n^2)$
Insertion Sort	Sort	插入到正确位置	$\Theta(n^2)$
Greedy Change	Optimization	每次选最大面额	取决于硬币种类
Greedy Schedule	Optimization	选最早结束的	取决于排序

3.1.10 10. Exam Tips (考点提示)

1. **算法性质**：Input, Output, Correctness, Finiteness, Effectiveness, Generality —能够列举并解释
2. **Linear Search vs Binary Search**：理解差异，Binary Search 要求列表有序，复杂度 $O(\log n)$ vs $O(n)$
3. **Bubble Sort 和 Insertion Sort**：能手动模拟排序过程，能写出伪代码
4. **Greedy Algorithm**：理解”局部最优不一定全局最优”，能举反例（如 1, 7, 10 找零）
5. **Halting Problem**：理解反证法证明思路，知道它是 unsolvable problem
6. **伪代码**：能读懂和写出基本伪代码

3.2 Chapter 3.2: Growth of Functions (函数的增长)

3.2.1 1. Big-O Notation (大 O 记号)

1.1 Definition (定义)

Definition: Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ **is** $O(g(x))$ if there are constants C and k such that >

$$|f(x)| \leq C|g(x)| \quad \text{whenever } x > k$$

其中：- C 和 k 称为 **witnesses (凭证)** ——找到任意一对即可 - 读作：”$f(x)$ is big-O of $g(x)$” 或”g asymptotically dominates f” - **含义**：当 x 足够大时，$f(x)$ 的增长速度不超过 $g(x)$ 的常数倍

1.2 图示理解

```
f(x) is O(g(x)) 意味着：
              C · g(x)
               /
        f(x) /
           /
          /
         /
        /  （当 x > k 时，f(x) 始终在 C · g(x) 下方）
       /
```

1.3 Important Points (重要事项)

1. **一对 witnesses 找到后，就有无穷多对**——可以把 C 或 k 取得更大
2. 可以写成”$f(x) = O(g(x))$”，但这实际上是**滥用等号**，准确含义是 $f(x) \in O(g(x))$
3. 通常我们忽略绝对值符号，因为我们处理的函数都取正值
4. **目标**：找到尽可能小的 $g(x)$（在常数倍范围内）

3.2.2 2. Examples of Big-O (大 O 的例子)

Example 1: 证明 $7x^2$ 是 $O(x^3)$

Solution: 当 $x > 1$ 时，$7x^2 < 7x^3$。取 $C = 7, k = 1$ 作为 witnesses，即有 $7x^2 \leq 7x^3$ for $x > 1$。所以 $7x^2$ 是 $O(x^3)$。

Example 2: 证明 $x^2 + 2x + 1$ 是 $O(x^2)$

Solution: 当 $x > 1$ 时，$2x < 2x^2$，$1 < x^2$。因此 $x^2 + 2x + 1 < x^2 + 2x^2 + x^2 = 4x^2$。取 $C = 4, k = 1$，即得证。或者：当 $x > 2$ 时，$2x < x^2$，$1 < x^2$。因此 $x^2 + 2x + 1 < x^2 + x^2 + x^2 = 3x^2$。取 $C = 3, k = 2$，也是有效的 witnesses。

注意：两个不同的表达式都是 $O(x^2)$，它们是 **same order (同阶)**。

Example 3: 证明 x^2 不是 $O(x)$

Solution (反证法): 假设存在常数 C 和 k，使得当 $x > k$ 时 $x^2 \leq Cx$。则 $x \leq C$ 对所有的 $x > k$ 成立，这显然矛盾（因为 x 可以无限大）。所以 x^2 不是 $O(x)$。

3.2.3 3. Big-O Estimates for Polynomials (多项式的大 O 估计)

Theorem: Let

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

where $a_0, a_1, \ldots, a_n$ are real numbers with $a_n \neq 0$. Then $f(x)$ is $O(x^n)$.

证明思路（使用三角不等式 triangle inequality）：

$$|f(x)| = |a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0|$$

当 $x > 1$ 时：

$$|f(x)| \leq |a_n| x^n + |a_{n-1}| x^{n-1} + \cdots + |a_1| x + |a_0|$$

$$\leq x^n (|a_n| + |a_{n-1}| + \cdots + |a_1| + |a_0|)$$

取 $C = |a_n| + |a_{n-1}| + \cdots + |a_1| + |a_0|$，$k = 1$ 即可。

关键结论：多项式的**最高次项 (leading term)** 决定了其增长阶。

3.2.4 4. Big-O Estimates for Important Functions (重要函数的大 O 估计)

4.1 Sum of First n Positive Integers (前 n 个正整数之和)

$$1+2+3+\cdots+n=\frac{n(n+1)}{2}$$

所以:

- $1+2+\cdots+n$ 是 $O(n^2)$
- 实际上它是 $\Theta(n^2)$（见后文 Big-Theta）

4.2 Factorial Function (阶乘)

$$n!=1\cdot 2\cdot 3\cdots n$$

- $n!$ 是 $O(n^n)$（因为 $n!\le n^n$）
- 更精确地：$n!$ 是 $O(n^n)$，但 $n!$ 不是 $O(n^{n-1})$

4.3 Logarithm of Factorial

$$\log(n!)=\log 1+\log 2+\cdots+\log n$$

- $\log(n!)$ 是 $O(n\log n)$
- 事实上 $\log(n!)=\Theta(n\log n)$（斯特林公式 Stirling's approximation）

3.2.5 5. Growth Hierarchy (增长阶层次)

以下是从**慢**到**快**的增长顺序（$d>c>0, b>1$）:

Order	Example	Description
$O(1)$	constant	常数阶
$O(\log n)$	$\log_2 n$	对数阶
$O((\log n)^c)$	$(\log n)^2$	对数多项式阶
$O(n^c)$ for $0<c<1$	$\sqrt{n}$	次线性阶
$O(n)$	n	线性阶
$O(n\log n)$	$n\log n$	线性对数阶
$O(n^c)$ for $c>1$	n^2, n^3	多项式阶
$O(b^n)$	$2^n, e^n$	指数阶
$O(n!)$	$n!$	阶乘阶

核心比较关系

1. 如果 $d>c>0$，则 n^c 是 $O(n^d)$，但 n^d **不是** $O(n^c)$
2. 如果 $b>1$ 且 $c,d>0$，则 $(\log_b n)^c$ 是 $O(n^d)$，但 n^d **不是** $O((\log_b n)^c)$

- 即：**任意正幂次的对数增长慢于任意正幂次的多项式**

1. 如果 $b>1$ 且 $d>0$，则 n^d 是 $O(b^n)$，但 b^n **不是** $O(n^d)$

- 即：**任意多项式增长慢于任意指数函数**

1. 如果 $c>b>1$，则 b^n 是 $O(c^n)$，但 c^n **不是** $O(b^n)$

3.2.6 6. Combinations of Functions (函数的组合)

加法组合

如果 $f_1(x)$ 是 $O(g_1(x))$，$f_2(x)$ 是 $O(g_2(x))$，则 $(f_1+f_2)(x)$ 是 $O(\max(|g_1(x)|,|g_2(x)|))$。

直观理解：加法取增长较快的那一个。

乘法组合

如果 $f_1(x)$ 是 $O(g_1(x))$，$f_2(x)$ 是 $O(g_2(x))$，则 $(f_1f_2)(x)$ 是 $O(g_1(x)g_2(x))$。

3.2.7 7. Ordering Functions by Growth (按增长排序)

Example: 将下列函数按增长阶从小到大排序，使得每个函数都是后一个函数的 big-O:

Functions
$f_1(n)=(n^2+1)^n$
$f_2(n)=8^{\log n}$
$f_3(n)=(\log n)^3$
$f_4(n)=n^{\log n}$
$f_5(n)=10000$
$f_6(n)=\log(\log n)$
$f_7(n)=2^{n^2}$
$f_8(n)=n^{1.5}$
$f_9(n)=\log n$
$f_{10}(n)=n!$

Solution（从小到大排序）:

排序	函数	解释
1	$f_5(n)=10000$	常数，不随 n 增长
2	$f_6(n)=\log(\log n)$	增长最慢
3	$f_3(n)=(\log n)^3$	对数多项式
4	$f_9(n)=\log n$	对数（$(\log n)^3$ 实际比 $\log n$ 增长快，这里需要仔细——实际上 $(\log n)^3>\log n$ for large n，所以 $\log n$ 应该在 (lo

更正排序:

1. $f_5(n)=10000$（常数）
2. $f_6(n)=\log(\log n)$（增长最慢）
3. $f_9(n)=\log n$
4. $f_3(n)=(\log n)^3$
5. $f_8(n)=n^{1.5}$（多项式，阶 1.5）
6. $f_2(n)=8^{\log n}=n^{\log 8}=n^3$（次大，$8^{\log n}=n^{\log 8}=n^3$）
7. $f_4(n)=n^{\log n}$（比 n^3 增长快，因为 $\log n>3$ for large n）
8. $f_1(n)=(n^2+1)^n\approx n^{2n}$（指数型函数）
9. $f_{10}(n)=n!$（阶乘增长快于任何 c^n，但慢于 2^{n^2}）
10. $f_7(n)=2^{n^2}$（增长最快：$2^{n^2}=(2^n)^n$，比 $n!$ 更快，因为 2^n 远快于 n/e）

3.2.8 8. Big-Omega Notation (Ω 记号)

Definition (定义)

Definition: Let f and g be functions. We say that $f(x)$ **is** $\Omega(g(x))$ if there are constants C and k such that >

$$|f(x)|\ge C|g(x)|\quad \text{whenever } x>k$$

- 读作："$f(x)$ is big-Omega of $g(x)$" - **含义**：Big-O 给出上界 (upper bound)，Big-Omega 给出**下界 (lower bound)** - $f(x)$ 是 $\Omega(g(x))$ 当且仅当 $g(x)$ 是 $O(f(x))$

Example

证明 x^2 是 $\Omega(x)$:

- 对 $x>1$，有 $x^2\ge x$，取 $C=1, k=1$
- 所以 x^2 是 $\Omega(x)$（即 x^2 增长**至少**和 x 一样快）

3.2.9 9. Big-Theta Notation (Θ 记号)

Definition (定义)

Definition: Let f and g be functions. We say that $f(x)$ **is** $\Theta(g(x))$ if $f(x)$ is $O(g(x))$ **and** $f(x)$ is $\Omega(g(x))$.

- 读作："$f(x)$ is big-Theta of $g(x)$" - 也说："$f(x)$ is of order $g(x)$" 或"$f(x)$ and $g(x)$ are of the same order" - **等价定义**：存在 $C_1, C_2, k>0$ 使得

$$C_1|g(x)|\le|f(x)|\le C_2|g(x)|\quad \text{whenever } x>k$$

图示理解

```
    C2 · g(x)  (上界 - 来自 Big-O)
     /
f(x) /
   /   (f(x) 被夹在 C1 · g 和 C2 · g 之间)
  /
 C1 · g(x)  (下界 - 来自 Big-Omega)
```

Example 1: 前 n 个正整数之和

$$f(n) = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

证明 $f(n)$ 是 $\Theta(n^2)$：**Big-O 部分**：$f(n) = \frac{n^2+n}{2} \leq n^2$ (for $n \geq 1$)，所以 $f(n)$ 是 $O(n^2)$。**Big-Omega 部分**：

$$f(n) = \frac{n^2+n}{2} \geq \frac{n^2}{2} \quad (\text{for } n \geq 1)$$

所以 $f(n)$ 是 $\Omega(n^2)$。因此 $f(n)$ 是 $\Theta(n^2)$。

Example 2: 证明 $f(x) = 3x^2 + 8x\log x$ 是 $\Theta(x^2)$

Big-O 部分：$3x^2 + 8x\log x \leq 3x^2 + 8x^2 = 11x^2$ (for $x \geq 1$)，所以是 $O(x^2)$。**Big-Omega 部分**：$3x^2 + 8x\log x \geq 3x^2$ (for $x > 0$)，所以是 $\Omega(x^2)$。因此 $f(x)$ 是 $\Theta(x^2)$。

3.2.10 10. Big-Theta for Polynomials (多项式的大 Theta 估计)

Theorem: Let

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

where $a_n \neq 0$. Then $f(x)$ is $\Theta(x^n)$ (or $\Theta(x^n)$).

Example: - $f(x) = 3x^2 + 2x + 1$ 是 $\Theta(x^2)$ - $f(x) = 5x^3 + 100x^2 + 50$ 是 $\Theta(x^3)$

3.2.11 11. 三种记号对比总结

Notation	Type	Meaning	Inequality
Big-O $O(g)$	Upper bound (上界)	f grows no faster than g	$f(x) \leq C \cdot g(x)$
Big-Omega $\Omega(g)$	Lower bound (下界)	f grows at least as fast as g	$f(x) \geq C \cdot g(x)$
Big-Theta $\Theta(g)$	Tight bound (紧界)	f grows at the same rate as g	$C_1 \cdot g(x) \leq f(x) \leq C_2 \cdot g(x)$

关系图

```
f is Θ(g)  <==>  f is O(g) AND f is Ω(g)

f is O(g)   <==>  g is Ω(f)
f is Ω(g)   <==>  g is O(f)
```

3.2.12 12. Exam Tips (考点提示)

1. **Big-O 定义**必考：能写出 $|f(x)| \leq C|g(x)|$ for $x > k$，理解 witnesses 的概念
2. **多项式**的 Big-O 和 Big-Theta：最高次项决定增长阶
3. **比较增长阶**：$\log n \ll n^c \ll b^n \ll n!$（对数 < 多项式 < 指数 < 阶乘）
4. **三种记号的区别**：O 上界，Ω 下界，Θ 紧界——考试可能会要求证明
5. 能通过代数变换简化函数：如 $8^{\log n} = n^{\log 8} = n^3$
6. 注意 $n^{\log n}$ 增长快于任何 n^c，因为 $\log n$ 会超过任意常数 c
7. 能判断 f 是/不是 $O(g)$，会用反证法证明" 不是"
8. 熟练掌握三角不等式 (triangle inequality) 用于多项式估计

3.3 Chapter 3.3: Complexity of Algorithms (算法复杂度)

3.3.1 1. Introduction (引言)

给定一个算法，如何评估它在特定规模输入下的效率？我们需要回答两个问题：

1. **Time Complexity (时间复杂度)**：算法解决一个问题需要**多少时间**？
2. **Space Complexity (空间复杂度)**：算法解决一个问题需要**多少内存**？

本课程**主要关注时间复杂度**。空间复杂度在后续课程中学习。

如何度量时间复杂度？

- 不依赖具体实现细节（硬件、软件、编程语言）
- 使用算法执行的**基本操作次数**（如比较、算术运算）
- 使用 **Big-O / Big-Theta** 记号来估计

3.3.2 2. Worst-Case vs Average-Case Complexity (最坏情况 vs 平均情况)

Worst-Case Complexity (最坏情况复杂度)

- 对于规模为 n 的输入，算法所需操作数的**最大值**
- 提供了算法时间需求的**上界 (upper bound)**
- 相对容易分析
- **本课程主要关注最坏情况**

Average-Case Complexity (平均情况复杂度)

- 在所有规模为 n 的输入上，算法所需操作数的**平均值**
- 通常更难确定
- 某些情况下容易分析（如线性搜索）

3.3.3 3. Complexity Analysis Examples (复杂度分析实例)

3.1 Finding the Maximum Element (找最大值)

Algorithm:

```
procedure max(a1, a2, ..., an: integers)
    max := a1
    for i := 2 to n
        if max < ai then max := ai
    return max {max is the largest element}
```

分析:

- 每次循环执行一次比较 `max < ai`
- 循环执行 $n-1$ 次
- 外加最后一次判断 $i > n$ 的比较
- 总比较数：$2(n-1)+1 = 2n-1$，或简化计为 $n-1$ 次核心比较
- **时间复杂度**：$\Theta(n)$

3.2 Linear Search (线性搜索)

Algorithm:

```
procedure linear search(x: integer, a1, a2, ..., an: distinct integers)
    i := 1
    while (i <= n and x != ai)
        i := i + 1
    if i <= n then location := i
    else location := 0
    return location
```

Worst-Case 分析:

- 每一步进行 2 次比较：$i \leq n$ 和 $x \neq a_i$
- 如果 $x = a_i$，使用 $2i+1$ 次比较
- 如果 x 不在列表中，使用 $2n+2$ 次比较
- **最坏情况**：$2n+2$ 次比较
- **时间复杂度**：$\Theta(n)$

Average-Case 分析:

- 假设元素等可能出现在任一位置
- 若 $x = a_i$，比较次数为 $2i+1$
- 平均比较次数：

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

约为最坏情况的一半，但仍是线性阶。

3.3 Binary Search (二分搜索)

Algorithm:

```
procedure binary search(x: integer, a1, a2, ..., an: increasing integers)
    i := 1
    j := n
    while i < j
        m := floor((i + j) / 2)
        if x > am then i := m + 1
        else j := m
    if x = ai then location := i
    else location := 0
    return location
```

分析（假设 $n = 2^k$）:

- 每轮进行 2 次比较：$i < j$ 和 $x > a_m$

- 第一轮后，搜索范围减半为 2^{k-1}
- 第二轮后，为 2^{k-2}
- 以此类推，直到列表大小为 1
- 共进行 k 轮，$k = \log_2 n$
- 每轮 2 次比较，最后还有 1 次判断 $x = a_i$
- 总比较数：$2\log_2 n + 1$
- **时间复杂度**：$\Theta(\log n)$

对比：二分搜索 $\Theta(\log n)$ 远优于线性搜索 $\Theta(n)$，但要求输入有序。

3.4 Bubble Sort (冒泡排序)

Algorithm:

```
procedure bubblesort(a1, ..., an: real numbers with n >= 2)
    for i := 1 to n - 1
        for j := 1 to n - i
            if aj > aj+1 then interchange aj and aj+1
    {a1, ..., an is now in increasing order}
```

分析:

- 外层循环 $n-1$ 轮
- 第 i 轮内层循环 $n-i$ 次比较
- 总比较次数:

$$(n-1) + (n-2) + \cdots + 1 = \frac{n(n-1)}{2}$$

- **时间复杂度**：$\Theta(n^2)$

3.5 Insertion Sort (插入排序)

Algorithm:

```
procedure insertion sort(a1, ..., an: real numbers with n >= 2)
    for j := 2 to n
        i := 1
        while aj > ai
            i := i + 1
        m := aj
        for k := 0 to j - i - 1
            aj-k := aj-k-1
        ai := m
    {Now a1, ..., an is in increasing order}
```

Worst-Case 分析（输入为递减序列）:

- 第 j 个元素需要与前面 $j-1$ 个元素比较
- 总比较次数:

$$1 + 2 + \cdots + (n-1) = \frac{n(n-1)}{2}$$

- **时间复杂度**：$\Theta(n^2)$

3.6 Matrix Multiplication (矩阵乘法)

Algorithm:

```
procedure matrix multiplication(A, B: matrices)
    for i := 1 to m
        for j := 1 to n
            cij := 0
            for q := 1 to k
                cij := cij + aiq * bqj
    return C {C = [cij] is the product of A and B}
```

分析（$n \times n$ 方阵）:

- 内层循环：n 次乘法和 n 次加法
- 共有 n^2 个元素需要计算
- 总操作数：$n \cdot n^2 = n^3$ 次乘法和同样多次加法
- **时间复杂度**：$\Theta(n^3)$

3.7 Boolean Product (布尔积)

Algorithm:

```
procedure Boolean product(A, B: zero-one matrices)
    for i := 1 to m
        for j := 1 to n
            cij := 0
            for q := 1 to k
                cij := cij OR (aiq AND bqj)
    return C {C = [cij] is the Boolean product of A and B}
```

分析（$n \times n$ 零一矩阵）:

- 与矩阵乘法相同结构
- **时间复杂度**：$\Theta(n^3)$ 次 bit operations

3.3.4 4. Matrix-Chain Multiplication (矩阵链乘法)

Problem: 计算 $A_1 A_2 \cdots A_n$，其中 A_i 是 $m_i \times m_{i+1}$ 矩阵。矩阵乘法是**可结合的 (associative)**，不同的结合顺序会影响总乘法次数。

Example: A_1 是 30×1，A_2 是 1×40，A_3 是 40×10。有两种计算顺序:

顺序	步骤	乘法次数
$A_1(A_2A_3)$	A_2A_3: $1 \times 40 \times 10 = 400$	
	$A_1 \times (A_2A_3)$: $30 \times 1 \times 10 = 300$	**共 700**
$(A_1A_2)A_3$	A_1A_2: $30 \times 1 \times 40 = 1200$	
	$(A_1A_2) \times A_3$: $30 \times 40 \times 10 = 12000$	**共 13200**

结论：$A_1(A_2A_3)$ 更好，仅需要 700 次乘法，而 $(A_1A_2)A_3$ 需要 13200 次。

关键：选择最优的矩阵乘法顺序可以极大地减少计算量。这类问题可以用 **dynamic programming (动态规划)** 解决。

3.3.5 5. Algorithmic Paradigms (算法范式)

Algorithmic paradigm: 一种基于特定概念的通用算法设计方法。

本课程涉及的范式:

Paradigm	Description	Examples
—	—	—
Greedy Algorithms (贪心)	每一步做当前最优选择	找零、任务调度（3.1 节）
Brute-Force Algorithms (暴力)	最直接的解法，不加优化	线性搜索、冒泡排序、插入排序
Divide-and-Conquer (分治)	分而治之	归并排序（第 8 章）
Dynamic Programming (动态规划)	子问题重叠，记忆化	矩阵链乘法（第 8 章）
Backtracking (回溯)	试探性搜索	（第 11 章）
Probabilistic Algorithms (概率)	引入随机性	（第 7 章）

Brute-Force (暴力算法)

Brute-force algorithm: 以最直接的方式解决问题，不利用任何能使算法更高效的思想。

Example: Closest Pair Problem (最近点对问题)

Problem: 给定平面上的 n 个点，找到距离最近的两个点。

Distance formula:

$$\text{distance}((x_i, y_i), (x_j, y_j)) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

实际计算中无需开平方，比较平方距离即可。

Brute-Force Algorithm:

```
procedure closest pair((x1, y1), (x2, y2), ..., (xn, yn): xi, yi real numbers)
    min = infinity
    for i := 1 to n
        for j := 1 to i - 1
            if ((xj - xi)^2 + (yj - yi)^2) < min then
                min := ((xj - xi)^2 + (yj - yi)^2)
                closest pair := (xi, yi), (xj, yj)
    return closest pair
```

分析: - 共检查 $\frac{n(n-1)}{2}$ 对点 - **时间复杂度**：$\Theta(n^2)$

更优算法：使用分治法可达到 $O(n \log n)$（第 8 章）。

3.3.6 6. Understanding Complexity (理解复杂度)

各复杂度级别在实际中的表现

假设计算机每秒执行 10^8 次操作:

n	$\log n$	n	$n \log n$	n^2	2^n	$n!$
10	3.3 ns	0.1 μs	0.33 μs	1 μs	10 μs	3.6 ms
100	6.6 ns	1 μs	6.6 μs	0.1 ms	4×10^{14} yr	-
10^3	10 ns	10 μs	0.1 ms	10 ms	-	-
10^6	20 ns	10 ms	0.2 s	2.8 hr	-	-
10^9	30 ns	10 s	4.5 min	317 yr	-	-

可见：**指数级 2^n 和阶乘级 $n!$ 算法在实际中几乎不可行**。

Example: 行列式计算 (Determinant)

- **方法一**：用定义计算，复杂度 $O(n \cdot n!)$
- **方法二**：通过行消元化为上三角行列式，复杂度 $O(n^3)$

当 $n = 50$：

- 方法一需 1.5×10^{65} 次乘法，约 5×10^{46} 年
- 方法二仅需 6×10^6 次乘法，约 6×10^{-5} 秒

结论：选择合适的算法至关重要！

3.3.7 7. Complexity of Problems (问题的复杂度分类)

Tractable vs Intractable (易解 vs 难解)

Classification	Definition	Examples
Tractable (易解)	存在**多项式时间**算法	排序 $O(n^2)$, 搜索 $O(\log n)$
Intractable (难解)	不存在多项式时间算法	某些问题需要指数时间
Unsolvable (不可解)	根本不存在算法	**Halting Problem (停机问题)**

Class P, NP, NP-Complete

```
      NP-Complete
     /           \
    /             \
   P               NP
    \             /
     \           /
  All Decision Problems
```

Class	Meaning
Class P	**Polynomial time** —存在多项式时间算法的问题
Class NP	**Nondeterministic Polynomial time** —解可以在多项式时间内**验证**，但尚未找到多项式时间求解算法
NP-Complete	NP 中最难的一类问题——如果能解决其中一个，就可以解决所有 NP 问题
P vs NP Problem	问是否 $P = NP$？即所有可验证的问题是否也都可以在多项式时间内求解？

Key Points:

- P 中的问题 = 可以在多项式时间内**求解**
- NP 中的问题 = 解可以在多项式时间内**验证**
- 注意：”尚未找到多项式算法”与”证明不存在多项式算法”是两回事
- **Satisfiability (可满足性问题)** 是一个 NP-Complete 问题
- 目前普遍认为 $P \neq NP$，但尚未被证明

3.3.8 8. Summary of Time Complexities (时间复杂度总结)

Algorithm	Problem Type	Worst-Case	Average-Case
Find Max	Search	$\Theta(n)$	$\Theta(n)$
Linear Search	Search	$\Theta(n)$	$\Theta(n)$
Binary Search	Search	$\Theta(\log n)$	$\Theta(\log n)$
Bubble Sort	Sort	$\Theta(n^2)$	$\Theta(n^2)$
Insertion Sort	Sort	$\Theta(n^2)$	$\Theta(n^2)$
Matrix Multiplication	Arithmetic	$\Theta(n^3)$	$\Theta(n^3)$
Boolean Product	Boolean	$\Theta(n^3)$	$\Theta(n^3)$
Closest Pair (Brute)	Geometry	$\Theta(n^2)$	$\Theta(n^2)$

3.3.9 9. Exam Tips (考点提示)

1. **复杂度分析**：能分析给定伪代码的时间复杂度，计算基本操作次数
2. **最坏情况 vs 平均情况**：理解区别，考试通常问 worst-case
3. **常见算法复杂度**：要熟记（见上表）
4. **二分搜索复杂度**：为什么是 $\Theta(\log n)$？每次搜索范围减半
5. **P vs NP**：能解释 P、NP、NP-Complete 的含义和区别

6. **Tractable vs Intractable**：多项式时间 = tractable，指数时间 = intractable
7. **Halting Problem**：unsolvable problem 的经典例子
8. **算法范式**：区分 greedy、brute-force、divide-and-conquer、dynamic programming
9. **矩阵链乘法**：理解不同结合顺序对效率的影响

第四章　Chapter 4: Number Theory (数论)

4.1 Chapter 4.1: Divisibility and Modular Arithmetic (整除性与模运算)

4.1.1 1. Divisibility (整除)

Definition: If a and b are integers with $a \neq 0$, we say that a **divides** b if there exists an integer c such that $b = ac$.

- 当整数 $a \neq 0$ 且存在整数 c 使 $b = ac$ 时，称 a **整除** b。

Notation:

- $a \mid b$ denotes that a divides b. (a 整除 b)
- $a \nmid b$ denotes that a does NOT divide b. (a 不整除 b)
- When $a \mid b$, b/a is an integer.

Terminology:

- a is a **factor (因子)** or **divisor (除数)** of b
- b is a **multiple (倍数)** of a

Example: Determine whether $3 \mid 7$ and whether $3 \mid 12$.

- $3 \mid 7$? $7/3$ is not an integer, so $3 \nmid 7$.
- $3 \mid 12$? $12/3 = 4$ is an integer, so $3 \mid 12$.

4.1.2 2. Properties of Divisibility (整除的性质)

Theorem 1: Let a, b, c be integers, where $a \neq 0$.

Property	Statement
(i)	If $a \mid b$ and $a \mid c$, then $a \mid (b + c)$
(ii)	If $a \mid b$, then $a \mid bc$ for all integers c
(iii)	If $a \mid b$ and $b \mid c$, then $a \mid c$

Proof of (i):

- Suppose $a \mid b$ and $a \mid c$, then $\exists s, t \in \mathbb{Z}$ such that $b = as$ and $c = at$.
- Then $b + c = as + at = a(s + t)$, so $a \mid (b + c)$.

Corollary: If $a \mid b$ and $a \mid c$, then $a \mid (mb + nc)$ for all integers m, n.

- 推论：若 a 整除 b 和 c，则 a 整除 b 和 c 的任意整数线性组合。

4.1.3 3. Division Algorithm (除法算法)

Theorem (Division Algorithm): If a is an integer and d a positive integer, then there are **unique** integers q and r, with $0 \leq r < d$, such that:

$$a = dq + r$$

- d is called the **divisor (除数)**
- a is called the **dividend (被除数)**
- q is called the **quotient (商)**
- r is called the **remainder (余数)**

Definitions of div and mod:

- $q = a$ **div** d (quotient)
- $r = a$ **mod** d (remainder)

Examples:

1. What are the quotient and remainder when 101 is divided by 11?

- $101 = 11 \cdot 9 + 2$
- $q = 101 \text{ div } 11 = 9$, $r = 101 \bmod 11 = 2$

1. What are the quotient and remainder when -11 is divided by 3?

- $-11 = 3 \cdot (-4) + 1$ (注意：余数必须满足 $0 \leq r < d$)
- $q = -11 \text{ div } 3 = -4$, $r = -11 \bmod 3 = 1$

考点：当被除数为负数时，余数必须是非负且小于除数。不要写成 $-11 = 3 \cdot (-3) + (-2)$，因为余数 -2 不满足 $0 \leq r < d$。

4.1.4 4. Congruence Relation (同余关系)

Definition: If a and b are integers and m is a positive integer, then a **is congruent to** b **modulo** m if m divides $a - b$. 即 $m \mid (a - b)$。

- **Notation**: $a \equiv b \pmod{m}$
- m is called the **modulus (模数)**
- $a \not\equiv b \pmod{m}$ means a is NOT congruent to b modulo m

Key Insight: Two integers are congruent modulo m **iff** they have the same remainder when divided by m.

- 两个整数模 m 同余当且仅当它们除以 m 的余数相同。

Examples:

- $17 \equiv 5 \pmod{6}$ because $6 \mid (17 - 5) = 12$.
- $24 \not\equiv 14 \pmod{6}$ because $6 \nmid (24 - 14) = 10$.

Theorem 2: $a \equiv b \pmod{m}$ iff there is an integer k such that $a = b + km$. **Proof**:

- ($\Rightarrow$) If $a \equiv b \pmod{m}$, then $m \mid (a - b)$, so $\exists k \in \mathbb{Z}$ with $a - b = km$, i.e., $a = b + km$.
- ($\Leftarrow$) If $a = b + km$, then $a - b = km$, so $m \mid (a - b)$, hence $a \equiv b \pmod{m}$.

Relationship between $\equiv$ (mod m) and mod notation:

- $a \equiv b \pmod{m}$ is a **relation** (关系)
- $a \bmod m = b$ is a **function** (函数)
- **Theorem**: $a \equiv b \pmod{m}$ iff $a \bmod m = b \bmod m$.

4.1.5 5. Congruences of Sums and Products (同余的加法和乘法)

Theorem 3: If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then:

1. $a + c \equiv b + d \pmod{m}$ (加法同余)
2. $ac \equiv bd \pmod{m}$ (乘法同余)

Proof:

- Since $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, there exist integers s, t such that $b = a + sm$ and $d = c + tm$.
- $b + d = (a + sm) + (c + tm) = (a + c) + m(s + t)$, so $a + c \equiv b + d \pmod{m}$.
- $bd = (a + sm)(c + tm) = ac + m(at + cs + stm)$, so $ac \equiv bd \pmod{m}$.

Example: Because $7 \equiv 2 \pmod{5}$ and $11 \equiv 1 \pmod{5}$:

- $18 = 7 + 11 \equiv 2 + 1 = 3 \pmod{5}$
- $77 = 7 \cdot 11 \equiv 2 \cdot 1 = 2 \pmod{5}$

Algebraic Manipulation:

- Multiplying both sides by an integer preserves congruence: $a \equiv b \pmod{m} \Rightarrow ca \equiv cb \pmod{m}$
- Adding an integer to both sides preserves congruence: $a \equiv b \pmod{m} \Rightarrow a + c \equiv b + c \pmod{m}$
- **Dividing** a congruence by an integer does **NOT** always produce a valid congruence!
- 例：$14 \equiv 8 \pmod{6}$ 成立，但两边除以 2 得 $7 \equiv 4 \pmod{6}$，这**不成立**。

4.1.6 6. Computing mod of Products and Sums

Corollary: Let m be a positive integer and a, b be integers. Then:

- $(a + b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$
- $ab \bmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m$

用途：大数取模时，可以先对每个因子取模，再运算，最后再取模，避免大数溢出。

4.1.7 7. Arithmetic Modulo m (模 m 算术)

Definition: Let $\mathbb{Z}_m = \{0, 1, 2, \ldots, m - 1\}$ be the set of nonnegative integers less than m.

- **Addition modulo** m: $a +_m b = (a + b) \bmod m$
- **Multiplication modulo** m: $a \cdot_m b = (a \cdot b) \bmod m$

Properties (模 m 算术的性质，与普通算术类似):

Property	Description
Closure (封闭性)	If $a, b \in \mathbb{Z}_m$, then $a +_m b \in \mathbb{Z}_m$ and $a \cdot_m b \in \mathbb{Z}_m$
Associativity (结合律)	$(a +_m b) +_m c = a +_m (b +_m c)$, similarly for multiplication
Commutativity (交换律)	$a +_m b = b +_m a$, $a \cdot_m b = b \cdot_m a$
Identity (单位元)	0 for addition, 1 for multiplication: $a +_m 0 = a$, $a \cdot_m 1 = a$
Additive Inverse (加法逆元)	$a +_m (m - a) = 0$, i.e., the additive inverse of a is $m - a$
Distributivity (分配律)	$a \cdot_m (b +_m c) = (a \cdot_m b) +_m (a \cdot_m c)$

Example: Find $7 +_{11} 9$ and $7 \cdot_{11} 9$.

- $7 +_{11} 9 = (7 + 9) \bmod 11 = 16 \bmod 11 = 5$
- $7 \cdot_{11} 9 = (7 \cdot 9) \bmod 11 = 63 \bmod 11 = 8$

4.1.8 小结与考点提示

1. **Division Algorithm（除法算法）**：核心是 $a = dq + r, 0 \le r < d$。特别注意负数的除法，余数必须是非负的。
2. **Divisibility properties（整除性质）**：传递性、线性组合等，常用于证明题。
3. **Congruence（同余）**：$a \equiv b \pmod m \iff m \mid (a-b) \iff a = b + km$。
4. **模运算的加减乘**: 同余式可以在加减乘下保持，但不能随意" 除以" 一个数。
5. $\mathbb{Z}_m$ **上的算术**: 记住加法和乘法都在模 m 下封闭，有单位元、逆元等结构。
6. **常见题型**: 计算 $a \bmod m$、判断同余、利用同余性质化简计算。

Exam tips:

- **English terms to know**: divisibility, divisor, dividend, quotient, remainder, congruence, modulus, modulo, additive inverse, closure.
- 考试中遇到"Find the quotient and remainder when a is divided by d"，记得用 Division Algorithm。
- 验证同余时，检查 $m \mid (a-b)$ 是否成立。

4.2 Chapter 4.2: Integer Representations (整数的表示)

4.2.1 1. Base b Representations (基 b 展开)

Theorem 1: Let b be a positive integer greater than 1. Then if n is a positive integer, it can be expressed uniquely in the form:

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

where:

- k is a nonnegative integer
- a_j are nonnegative integers less than b $(0 \le a_j < b)$
- $a_k \neq 0$

The a_j's are called the **base-b digits (基 b 数码)**. **Notation**: The **base b expansion** of n is denoted by $(a_k a_{k-1} \ldots a_1 a_0)_b$.

- 通常我们使用十进制 (base 10, decimal)，但计算机科学中常用二进制 (base 2, binary)、八进制 (base 8, octal)、十六进制 (base 16, hexadecimal)。
- 玛雅人使用 base 20，巴比伦人使用 base 60。

4.2.2 2. Binary Expansions (二进制展开)

Binary expansion uses base 2, with digits only $\{0, 1\}$. 计算机内部使用二进制表示整数。**Example**: What is the decimal expansion of $(1\,0101\,1111)_2$? Solution:

$$(1\,0101\,1111)_2 = 1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

$$= 256 + 0 + 64 + 0 + 16 + 8 + 4 + 2 + 1 = 351$$

Example: What is the decimal expansion of $(11\,1011)_2$? Solution:

$$(11\,1011)_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^0$$

$$= 32 + 16 + 8 + 2 + 1 = 59$$

4.2.3 3. Octal Expansions (八进制展开)

Octal expansion uses base 8, with digits $\{0, 1, 2, 3, 4, 5, 6, 7\}$. **Example**: What is the decimal expansion of $(7016)_8$?

$$(7016)_8 = 7 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 + 6 \cdot 8^0 = 7 \cdot 512 + 0 + 8 + 6 = 3584 + 14 = 3598$$

Example: What is the decimal expansion of $(111)_8$?

$$(111)_8 = 1 \cdot 8^2 + 1 \cdot 8^1 + 1 \cdot 8^0 = 64 + 8 + 1 = 73$$

4.2.4 4. Hexadecimal Expansions (十六进制展开)

Hexadecimal expansion uses base 16, with digits $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$, where $A = 10, B = 11, C = 12, D = 13, E = 14, F = 15$. **Example**: What is the decimal expansion of $(2AE0B)_{16}$?

$$(2AE0B)_{16} = 2 \cdot 16^4 + 10 \cdot 16^3 + 14 \cdot 16^2 + 0 \cdot 16^1 + 11 \cdot 16^0$$

$$= 2 \cdot 65536 + 10 \cdot 4096 + 14 \cdot 256 + 0 + 11$$

$$= 131072 + 40960 + 3584 + 0 + 11 = 175627$$

Example: What is the decimal expansion of $(E5)_{16}$?

$$(E5)_{16} = 14 \cdot 16^1 + 5 \cdot 16^0 = 224 + 5 = 229$$

4.2.5 5. Base Conversion Algorithm (进制转换算法)

To construct the base b expansion of an integer n:

1. Divide n by b: $n = bq_0 + a_0$ (remainder a_0 is the **rightmost** digit)
2. Divide q_0 by b: $q_0 = bq_1 + a_1$ (a_1 is the second digit from right)
3. Continue until the quotient becomes 0.
4. The remainders from last to first give the base b expansion.

Algorithm (pseudocode):

```
procedure base b expansion(n, b: positive integers with b > 1)
q := n
k := 0
while (q  0)
    a_k := q mod b
    q := q div b
    k := k + 1
return (a_{k-1}, ..., a_1, a_0)  {(a_{k-1}...a_1a_0)_b is base b expansion}
```

Example: Find the octal expansion of $(12345)_{10}$. Solution: Successively divide by 8:

- $12345 = 8 \cdot 1543 + 1$
- $1543 = 8 \cdot 192 + 7$
- $192 = 8 \cdot 24 + 0$
- $24 = 8 \cdot 3 + 0$
- $3 = 8 \cdot 0 + 3$

Reading remainders from last to first: $(30071)_8$.

4.2.6 6. Conversion Between Binary, Octal, and Hexadecimal

Key observation:

- Each octal digit corresponds to a block of **3** binary digits.
- Each hexadecimal digit corresponds to a block of **4** binary digits.
- 因此二进制与八进制、十六进制之间的转换非常容易。

Example: Find the octal and hexadecimal expansions of $(11\,1110\,1011\,1100)_2$. **Solution**:

- **To octal**: Group into blocks of 3 (从右向左):
- $(011\,111\,010\,111\,100)_2$ (补齐前导零)
- $011_2 = 3$, $111_2 = 7$, $010_2 = 2$, $111_2 = 7$, $100_2 = 4$
- Answer: $(37274)_8$
- **To hexadecimal**: Group into blocks of 4:
- $(0011\,1110\,1011\,1100)_2$ (补齐前导零)
- $0011_2 = 3$, $1110_2 = E$, $1011_2 = B$, $1100_2 = C$
- Answer: $(3EBC)_{16}$

4.2.7 7. Binary Addition of Integers (二进制加法)

Algorithm for adding two n-bit integers:

```
procedure add(a, b: positive integers)
{the binary expansions of a and b are (a_{n-1}...a_1a_0)_2 and (b_{n-1}...b_1b_0)_2}
c := 0  (carry)
for j := 0 to n-1
    d := floor((a_j + b_j + c) / 2)
    s_j := a_j + b_j + c - 2d
    c := d
s_n := c
return (s_n s_{n-1} ... s_0)  {the binary expansion of the sum}
```

- 每一位的运算：$a_j + b_j + carry$，结果的个位为 s_j，进位给下一位。
- **Time complexity**: $O(n)$ additions of bits for adding two n-bit integers.

4.2.8 8. Binary Multiplication of Integers (二进制乘法)

- 二进制乘法通过移位 (shift) 和加法实现。
- 对乘数的每一位 b_j：
- 如果 $b_j = 1$，则 partial product 是被乘数左移 j 位
- 如果 $b_j = 0$，则 partial product 为 0
- 将所有 partial products 相加得到最终结果。
- **Time complexity**: $O(n^2)$ additions of bits.

4.2.9 9. Binary Modular Exponentiation (二进制模幂运算)

重要应用：密码学中需要高效计算 $b^n \bmod m$，其中 b, n, m 都是大整数。

核心思想：利用 n 的二进制展开 $n = (a_{k-1} \dots a_1 a_0)_2$。

$$b^n = b^{a_{k-1}2^{k-1} + \dots + a_1 2 + a_0} = b^{a_{k-1}2^{k-1}} \cdots b^{a_1 2} \cdot b^{a_0}$$

只需计算 $b, b^2, b^4, \dots, b^{2^{k-1}}$，然后挑选 $a_j = 1$ 的项相乘。

Algorithm (Modular Exponentiation):

```
procedure modular exponentiation(b: integer, n = (a_{k-1}...a_1a_0)_2, m: positive integers)
x := 1
power := b mod m
for i := 0 to k-1
    if a_i = 1 then x := (x * power) mod m
    power := (power * power) mod m
return x  {x equals b^n mod m}
```

- **Time complexity**: $O((\log m)^2 \log n)$ bit operations.

Example: Compute $3^{11} \bmod 5$ using this method.

Solution: - Binary of 11: $11 = (1011)_2$ - Compute powers: - $3^1 \equiv 3 \pmod 5$ (bit $0 = 1$) - $3^2 \equiv 4 \pmod 5$ - $3^4 \equiv 4^2 = 16 \equiv 1 \pmod 5$ - $3^8 \equiv 1^2 = 1 \pmod 5$ (bit $3 = 1$)

- $3^{11} = 3^{8+2+1} = 3^8 \cdot 3^2 \cdot 3^1 \equiv 1 \cdot 4 \cdot 3 = 12 \equiv 2 \pmod 5$

So $3^{11} \bmod 5 = 2$.

Another example: Compute $7^{644} \bmod 645$.

(教材有此例，用上述算法可高效计算)

4.2.10 小结与考点提示

1. **进制转换 (Base Conversion)**:
- 任意进制 → 十进制：按权展开求和
- 十进制 → 任意进制：反复除以基，取余数（从后往前读）
- 二进制 ↔ 八进制/十六进制：按 3 位/4 位分组
1. **Binary Modular Exponentiation** 是高频考点：
- 将指数写成二进制
- 反复平方 (repeated squaring)
- 只乘二进制位为 1 的项
1. **算法复杂度**:
- Binary addition: $O(n)$
- Binary multiplication: $O(n^2)$
- Modular exponentiation: $O((\log m)^2 \log n)$

Exam tips:

- **English terms**: base expansion, binary, octal, hexadecimal, digit, bit, quotient, remainder, modular exponentiation.
- Converting between bases is a common exam question – practice the division method.
- Modular exponentiation is used heavily in RSA; understand the repeated squaring method.

4.3 Chapter 4.3: Primes and Greatest Common Divisors (素数与最大公约数)

4.3.1 1. Primes (素数/质数)

Definition: A positive integer $p > 1$ is called **prime** if the only positive factors of p are 1 and p.

- 一个大于 1 的正整数 p，如果它的正因子只有 1 和它本身，则称为**素数 (质数)**。

A positive integer greater than 1 that is not prime is called **composite (合数)**. **Examples**:

- 7 is prime (factors: $1, 7$)
- 9 is composite because it is divisible by 3 (factors: $1, 3, 9$)

4.3.2 2. Fundamental Theorem of Arithmetic (算术基本定理)

Theorem: Every positive integer greater than 1 can be written **uniquely** as a prime or as the product of two or more primes where the prime factors are written in order of nondecreasing size.

- 每个大于 1 的正整数都可以唯一地表示为素数的乘积（素数按非递减顺序排列）。

Examples:

- $100 = 2^2 \cdot 5^2$
- $999 = 3^3 \cdot 37$
- $1024 = 2^{10}$

4.3.3 3. The Sieve of Eratosthenes (埃拉托斯特尼筛法)

A method to find all primes not exceeding a specified positive integer. **Procedure** (以找出 ≤ 100 的所有素数为例子):

1. 列出从 2 到 100 的所有整数。

2. 删除 2 的所有倍数（除 2 本身外）。
3. 删除 3 的所有倍数（除 3 本身外）。
4. 继续这个过程直到达到 $\sqrt{100} = 10$。
5. 所有剩下的数都是素数：$\{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97\}$

Key lemma: If n is composite, then it has a prime divisor $\leq \sqrt{n}$.

- 若 n 是合数，则它有一个素因子 $\leq \sqrt{n}$。
- 因此只需试除到 $\sqrt{n}$ 即可判断 n 是否为素数。

4.3.4 4. Infinitude of Primes (素数无穷定理)

Theorem: There are infinitely many primes. (Euclid's proof, 欧几里得证明) **Proof** (反证法):

- Assume only finitely many primes exist: $p_1, p_2, \ldots, p_n$.
- Let $q = p_1 p_2 \cdots p_n + 1$.
- Either q is prime, or by the Fundamental Theorem of Arithmetic, q is a product of primes.
- But none of the p_j divides q (since $q \div p_j$ leaves remainder 1).
- Thus there must be a prime not in the original list. Contradiction.
- Therefore, there are infinitely many primes.

证明被认为是数学中最优美的证明之一，收录于 Paul Erdos 所说的"The Book"。

4.3.5 5. Mersenne Primes (梅森素数)

Definition: Prime numbers of the form $2^p - 1$, where p is prime, are called **Mersenne primes**. **Examples**:

- $2^2 - 1 = 3$ (prime) – yes
- $2^3 - 1 = 7$ (prime) – yes
- $2^5 - 1 = 31$ (prime) – yes
- $2^7 - 1 = 127$ (prime) – yes
- $2^{11} - 1 = 2047 = 23 \cdot 89$ (not prime)

Note: $2^p - 1$ is not always prime even when p is prime. $2^{11} - 1$ is composite. The **Great Internet Mersenne Prime Search (GIMPS)** is a distributed computing project searching for Mersenne primes.

4.3.6 6. Distribution of Primes (素数分布)

Prime Number Theorem (素数定理): The ratio of the number of primes not exceeding x and $x/\ln x$ approaches 1 as $x \to \infty$.

- 不超过 x 的素数个数近似为 $x/\ln x$。
- 随机选择的一个小于 n 的正整数是素数的概率约为 $1/\ln n$。

4.3.7 7. Greatest Common Divisor (GCD, 最大公约数)

Definition: Let a and b be integers, not both zero. The largest integer d such that $d \mid a$ and $d \mid b$ is called the **greatest common divisor** of a and b, denoted $\gcd(a, b)$. **Examples**:

- $\gcd(24, 36) = 12$
- $\gcd(17, 22) = 1$

Definition: a and b are **relatively prime (互质/互素)** if $\gcd(a, b) = 1$. **Definition**: Integers $a_1, a_2, \ldots, a_n$ are **pairwise relatively prime (两两互质)** if $\gcd(a_i, a_j) = 1$ whenever $i \neq j$.

4.3.8 8. Finding GCD Using Prime Factorizations (利用素因子分解求 GCD)

If the prime factorizations of a and b are:

$$a = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}, \quad b = p_1^{f_1} p_2^{f_2} \cdots p_n^{f_n}$$

(all primes appearing in either factorization are included, with exponent 0 if not present), then:

$$\gcd(a, b) = p_1^{\min(e_1, f_1)} p_2^{\min(e_2, f_2)} \cdots p_n^{\min(e_n, f_n)}$$

Example:

- $120 = 2^3 \cdot 3 \cdot 5$
- $500 = 2^2 \cdot 5^3$
- $\gcd(120, 500) = 2^{\min(3,2)} \cdot 3^{\min(1,0)} \cdot 5^{\min(1,3)} = 2^2 \cdot 3^0 \cdot 5^1 = 4 \cdot 1 \cdot 5 = 20$

注意：对大整数做素因子分解是困难的（这正是 RSA 安全性的基础），因此这种方法只对小整数实用。

4.3.9 9. Least Common Multiple (LCM, 最小公倍数)

Definition: The **least common multiple** of positive integers a and b is the smallest positive integer divisible by both a and b, denoted $\operatorname{lcm}(a, b)$. From prime factorizations:

$$\operatorname{lcm}(a, b) = p_1^{\max(e_1, f_1)} p_2^{\max(e_2, f_2)} \cdots p_n^{\max(e_n, f_n)}$$

Theorem: For positive integers a and b:

$$ab = \gcd(a, b) \cdot \operatorname{lcm}(a, b)$$

两数之积等于它们的最大公约数和最小公倍数之积。这个关系常用于验证结果。

Example: $\gcd(120, 500) = 20$, so $\operatorname{lcm}(120, 500) = \frac{120 \cdot 500}{20} = \frac{60000}{20} = 3000$.

4.3.10 10. Euclidean Algorithm (欧几里得算法/辗转相除法)

最重要：计算 GCD 的最高效算法。必须熟练掌握！

Lemma: Let $a = bq + r$, where a, b, q, r are integers. Then $\gcd(a, b) = \gcd(b, r)$.

- a 和 b 的最大公约数等于 b 和 r（余数）的最大公约数。- Proof: Any common divisor of a and b also divides $r = a - bq$, and vice versa.

Euclidean Algorithm:

```
procedure gcd(a, b: positive integers)
x := a
y := b
while y  0
    r := x mod y
    x := y
    y := r
return x  {gcd(a,b) = x}
```

Example: Find $\gcd(252, 198)$.

1. $252 = 1 \cdot 198 + 54$ ($252 \div 198$, remainder 54) 2. $198 = 3 \cdot 54 + 36$ 3. $54 = 1 \cdot 36 + 18$ 4. $36 = 2 \cdot 18 + 0 \leftarrow$ 余数为 0，停止

$\gcd(252, 198) = 18$ (最后一个非零余数)

Time complexity: $O(\log b)$, where $a > b$. 即使对很大的数也非常快。

4.3.11 11. Bézout's Theorem (贝祖定理) and gcd as Linear Combinations

Bézout's Theorem: If a and b are positive integers, then there exist integers s and t such that:

$$\gcd(a, b) = sa + tb$$

- 存在整数 s, t 使得 $\gcd(a, b)$ 可以表示为 a 和 b 的**线性组合**。

Bézout coefficients: The integers s and t in the equation above. **Bézout's identity**: The equation $\gcd(a, b) = sa + tb$. **Example**: Express $\gcd(252, 198) = 18$ as a linear combination of 252 and 198. **Solution** (working backwards through the Euclidean algorithm):

Step 1 (from the Euclidean algorithm above):

- (iii) $54 = 1 \cdot 36 + 18 \Rightarrow 18 = 54 - 1 \cdot 36$
- (ii) $198 = 3 \cdot 54 + 36 \Rightarrow 36 = 198 - 3 \cdot 54$
- (i) $252 = 1 \cdot 198 + 54 \Rightarrow 54 = 252 - 1 \cdot 198$

Step 2 (back substitute):

- $18 = 54 - 1 \cdot 36 \quad = 54 - 1 \cdot (198 - 3 \cdot 54)$
  $= 4 \cdot 54 - 1 \cdot 198$
- $18 = 4 \cdot (252 - 1 \cdot 198) - 1 \cdot 198 \quad = 4 \cdot 252 - 5 \cdot 198$

Thus $\gcd(252, 198) = 4 \cdot 252 - 5 \cdot 198$. Bézout coefficients: $s = 4$, $t = -5$ (or vice versa).

4.3.12 12. Consequences of Bézout's Theorem

Lemma 1: If a, b, c are positive integers such that $\gcd(a, b) = 1$ and $a \mid bc$, then $a \mid c$. Proof:

- $\gcd(a, b) = 1 \Rightarrow \exists s, t$ such that $sa + tb = 1$.
- Multiply by c: $sac + tbc = c$.
- Since $a \mid sac$ and $a \mid tbc$ (from $a \mid bc$), we have $a \mid c$.

Lemma 2 (Euclid's Lemma): If p is prime and $p \mid a_1 a_2 \cdots a_n$, then $p \mid a_i$ for some i.

- 如果素数 p 能整除多个数的乘积，则 p 至少整除其中一个数。
- This lemma is crucial for proving the **uniqueness** part of the Fundamental Theorem of Arithmetic.

4.3.13 13. Dividing Congruences by an Integer

Theorem 7: If $ac \equiv bc \pmod{m}$ and $\gcd(c, m) = 1$, then $a \equiv b \pmod{m}$.

- 当 c 与 m 互质时，同余式两边可以同时" 除以" c。
- 如果 $\gcd(c, m) \neq 1$，则一般不能这样操作。

4.3.14 小结与考点提示

1. **Primes**:

- 定义与 Fundamental Theorem of Arithmetic
- 素数无穷的证明（欧几里得证明，反证法）
- Prime Number Theorem: $\pi(x) \sim x/\ln x$

1. **GCD and LCM**:
- 定义、互质 (relatively prime)、两两互质 (pairwise relatively prime)
- 通过素因子分解求 GCD/LCM
- $ab = \gcd(a,b) \cdot \mathrm{lcm}(a,b)$

1. **Euclidean Algorithm（重中之重）**:
- 反复做带余除法直到余数为 0
- 最后一个非零余数就是 GCD
- 会手算，也要理解算法流程

1. **Bézout's Theorem（高频考点）**:
- 能用 Extended Euclidean Algorithm 回代求出 s, t 使得 $sa + tb = \gcd(a,b)$
- 应用：求乘法逆元（用于 4.4 解同余方程 / RSA）

1. **Euclid's Lemma**: 素数整除乘积则必整除某个因子

Exam tips:
- **English terms**: prime, composite, relatively prime, pairwise relatively prime, greatest common divisor (GCD), least common multiple (LCM), Euclidean algorithm, Bézout coefficients.
- "Find $\gcd(a,b)$ using the Euclidean algorithm" – 写出每一步除法，最后指出最后一个非零余数。
- "Express $\gcd(a,b)$ as a linear combination of a and b" – 先用欧几里得算法，再回代。

4.4 Chapter 4.4: Solving Congruences (解同余方程)

4.4.1 1. Linear Congruences (线性同余方程)

Definition: A congruence of the form $ax \equiv b \pmod{m}$, where m is a positive integer, a and b are integers, and x is a variable, is called a **linear congruence (线性同余方程)**.
- 解就是所有满足该同余式的整数 x。

Definition: An integer $\bar{a}$ such that $\bar{a}a \equiv 1 \pmod{m}$ is said to be an **inverse of a modulo m (模 m 的乘法逆元)**.
- 例：5 是 3 模 7 的逆元，因为 $5 \cdot 3 = 15 \equiv 1 \pmod{7}$。

关键思想：我们不能直接" 除以" a，但可以两边乘以 a 的逆元 $\bar{a}$ 来解出 x。

Theorem 1: If a and m are relatively prime ($\gcd(a,m) = 1$) and $m > 1$, then an inverse of a modulo m exists. Furthermore, this inverse is **unique modulo** m.

- 当 a 与 m 互质时，a 模 m 的逆元存在且模 m 唯一。

Proof: - Since $\gcd(a,m) = 1$, by Bézout's Theorem there exist integers s and t such that $sa+tm = 1$. - Modulo m: $sa+tm \equiv sa \equiv 1 \pmod{m}$, since $tm \equiv 0 \pmod{m}$. - Hence s is an inverse of a modulo m.

4.4.2 2. Finding Inverses (求逆元)

方法：使用 Euclidean Algorithm 求出 Bézout coefficients，系数 s 即为逆元。**Example 1**: Find an inverse of 3 modulo 7. Solution: $\gcd(3,7) = 1$, so inverse exists.
- Euclidean algorithm: $7 = 2 \cdot 3 + 1 \Rightarrow 1 = 7 - 2 \cdot 3$
- So $(-2) \cdot 3 \equiv 1 \pmod{7}$, i.e., -2 is an inverse of 3 modulo 7.
- Usually we take the positive representative: $-2 \equiv 5 \pmod{7}$.
- So 5 is also an inverse of 3 modulo 7 (in fact, any integer $\equiv -2 \pmod{7}$ works: $5, -9, 12, \ldots$).

Example 2: Find an inverse of 101 modulo 4620. Solution: First use Euclidean algorithm to show $\gcd(101, 4620) = 1$.
- $4620 = 45 \cdot 101 + 75$
- $101 = 1 \cdot 75 + 26$
- $75 = 2 \cdot 26 + 23$
- $26 = 1 \cdot 23 + 3$
- $23 = 7 \cdot 3 + 2$
- $3 = 1 \cdot 2 + 1$
- $2 = 2 \cdot 1 + 0$

Working backwards:
- $1 = 3 - 1 \cdot 2$
- $2 = 23 - 7 \cdot 3 \Rightarrow 1 = 3 - 1 \cdot (23 - 7 \cdot 3) = 8 \cdot 3 - 1 \cdot 23$
- $3 = 26 - 1 \cdot 23 \Rightarrow 1 = 8 \cdot (26 - 23) - 1 \cdot 23 = 8 \cdot 26 - 9 \cdot 23$
- $23 = 75 - 2 \cdot 26 \Rightarrow 1 = 8 \cdot 26 - 9 \cdot (75 - 2 \cdot 26) = 26 \cdot 26 - 9 \cdot 75$

- $26 = 101 - 1 \cdot 75 \Rightarrow 1 = 26 \cdot (101 - 75) - 9 \cdot 75 = 26 \cdot 101 - 35 \cdot 75$
- $75 = 4620 - 45 \cdot 101 \Rightarrow 1 = 26 \cdot 101 - 35 \cdot (4620 - 45 \cdot 101) \quad = -35 \cdot 4620 + 1601 \cdot 101$

Thus Bézout coefficients are -35 and 1601.
- $(-35) \cdot 4620 + 1601 \cdot 101 = 1$
- $1601 \cdot 101 \equiv 1 \pmod{4620}$
- So 1601 is an inverse of 101 modulo 4620.

4.4.3 3. Using Inverses to Solve Congruences (用逆元解同余方程)

To solve $ax \equiv b \pmod{m}$:
1. Find $\bar{a}$, an inverse of a modulo m.
2. Multiply both sides by $\bar{a}$: $\bar{a}ax \equiv \bar{a}b \pmod{m}$.
3. Since $\bar{a}a \equiv 1 \pmod{m}$, we get $x \equiv \bar{a}b \pmod{m}$.

Example: Solve $3x \equiv 4 \pmod{7}$. Solution:
- We found that 5 is an inverse of 3 modulo 7 (since $5 \cdot 3 = 15 \equiv 1 \pmod{7}$).
- Multiply both sides by 5: $5 \cdot 3x \equiv 5 \cdot 4 \pmod{7}$.
- $15x \equiv 20 \pmod{7} \Rightarrow x \equiv 6 \pmod{7}$ (since $15 \equiv 1$ and $20 \equiv 6$).
- Check: $3 \cdot 6 = 18 \equiv 4 \pmod{7}$.
- All solutions: $x \equiv 6 \pmod{7}$, i.e., $\ldots, -8, -1, 6, 13, 20, \ldots$

4.4.4 4. Chinese Remainder Theorem (CRT, 中国剩余定理)

历史：出自中国古代数学著作《孙子算经》（约公元 3-5 世纪），后在 1247 年由秦九韶在《数书九章》中完善为" 大衍求一术"。

Problem (Sun-Tsu's puzzle): 有物不知其数，三三数之剩二，五五数之剩三，七七数之剩二。问物几何?

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

Chinese Remainder Theorem: Let $m_1, m_2, \ldots, m_n$ be **pairwise relatively prime** positive integers greater than 1, and $a_1, a_2, \ldots, a_n$ arbitrary integers. Then the system

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

has a **unique solution modulo** $M = m_1 m_2 \cdots m_n$ (i.e., there is a solution x with $0 \le x < M$, and all other solutions are congruent to it modulo M).

CRT Construction Method:
1. Compute $M = m_1 m_2 \cdots m_n$.
2. For each $k = 1, \ldots, n$:
- Compute $M_k = M/m_k$.
- Since $\gcd(m_k, M_k) = 1$, find y_k, an inverse of M_k modulo m_k (i.e., $M_k y_k \equiv 1 \pmod{m_k}$).

1. The solution is: $x = a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots + a_n M_n y_n$.
2. Reduce modulo M to get the unique solution in $\{0, 1, \ldots, M-1\}$.

Example: Solve Sun-Tsu's problem.
- $m_1 = 3, m_2 = 5, m_3 = 7$
- $M = 3 \cdot 5 \cdot 7 = 105$
- $M_1 = 105/3 = 35,\ M_2 = 105/5 = 21,\ M_3 = 105/7 = 15$

Find inverses:
- $35 \equiv 2 \pmod{3}$, inverse of 2 mod 3 is 2 (since $2 \cdot 2 = 4 \equiv 1$). So $y_1 = 2$.
- $21 \equiv 1 \pmod{5}$, inverse of 1 mod 5 is 1. So $y_2 = 1$.
- $15 \equiv 1 \pmod{7}$, inverse of 1 mod 7 is 1. So $y_3 = 1$.

Solution:

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + a_3 M_3 y_3$$

$$= 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1$$

$$= 140 + 63 + 30 = 233 \equiv 233 - 2 \cdot 105 = 23 \pmod{105}$$

Answer: $x \equiv 23 \pmod{105}$, so the smallest positive solution is 23. Verify: $23 \div 3 = 7$ rem 2, $23 \div 5 = 4$ rem 3, $23 \div 7 = 3$ rem 2.

4.4.5 5. Back Substitution Method (回代法)

另一种解同余方程组的方法，逐次代入。**Example**: Find all integers x such that:

$$x \equiv 1 \pmod 5, \quad x \equiv 2 \pmod 6, \quad x \equiv 3 \pmod 7$$

Solution:

1. From first congruence: $x = 5t + 1$, where t is an integer.
2. Substitute into second: $5t + 1 \equiv 2 \pmod 6$.
- $5t \equiv 1 \pmod 6 \Rightarrow 5t \equiv 1 \pmod 6$.
- Inverse of 5 mod 6 is 5 (since $5 \cdot 5 = 25 \equiv 1$).
- $t \equiv 5 \pmod 6$, so $t = 6u + 5$.
- $x = 5(6u + 5) + 1 = 30u + 26$.
1. Substitute into third: $30u + 26 \equiv 3 \pmod 7$.
- $30u \equiv -23 \equiv -2 \equiv 5 \pmod 7$.
- $30 \equiv 2 \pmod 7$, so $2u \equiv 5 \pmod 7$.
- Inverse of 2 mod 7 is 4, so $u \equiv 20 \equiv 6 \pmod 7$.
- $u = 7v + 6$.
- $x = 30(7v + 6) + 26 = 210v + 206$.
1. Solution: $x \equiv 206 \pmod{210}$.

4.4.6 6. Fermat's Little Theorem (费马小定理)

Pierre de Fermat (1601-1665) Theorem (Fermat's Little Theorem): If p is prime and a is an integer not divisible by p, then:

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

Furthermore, for every integer a:

$$a^p \equiv a \pmod p$$

- 费马小定理：若 p 为素数且 $p \nmid a$，则 $a^{p-1} \equiv 1 \pmod p$。

Example: Find 7^{222} mod 11. Solution:

- By Fermat's Little Theorem: $7^{10} \equiv 1 \pmod{11}$ (since 11 is prime and $11 \nmid 7$).
- $7^{222} = 7^{22 \cdot 10+2} = (7^{10})^{22} \cdot 7^2 \equiv 1^{22} \cdot 49 \equiv 5 \pmod{11}$ (since $49 \equiv 5 \pmod{11}$).
- So $7^{222} \bmod 11 = 5$.

  推广：Euler's Theorem（欧拉定理）。对于正整数 n，$\varphi(n)$ 表示小于 n 且与 n 互质的正整数个数（欧拉函数）。若 $\gcd(a, n) = 1$，则 $a^{\varphi(n)} \equiv 1 \pmod n$。- 费马小定理是欧拉定理在 n 为素数时的特例（$\varphi(p) = p - 1$）。

4.4.7 7. Pseudoprimes (伪素数)

- 由费马小定理可知：若 n 是素数，则 $2^{n-1} \equiv 1 \pmod n$。
- 但反过来不一定成立：合数也可能满足该同余式。

Definition: Composite integers n such that $2^{n-1} \equiv 1 \pmod n$ are called **pseudoprimes to the base 2 (以 2 为底的伪素数)**。**Example**: $341 = 11 \cdot 31$ is composite, but $2^{340} \equiv 1 \pmod{341}$. Hence 341 is a pseudoprime to base 2. **General definition**: Let b be a positive integer. If n is composite and $b^{n-1} \equiv 1 \pmod n$, then n is called a **pseudoprime to the base** b. **Use in primality testing**:

- If n fails the test (i.e., $2^{n-1} \not\equiv 1 \pmod n$), then n is definitely composite.
- If n passes, it might be prime or a pseudoprime.
- Testing with multiple bases increases confidence.

Pseudoprimes are relatively rare. Among positive integers $< 10^{10}$, there are $455,052,512$ primes but only $14,884$ pseudoprimes to base 2.

4.4.8 8. Carmichael Numbers (卡米切尔数) [Optional]

Definition: A composite integer n that satisfies $b^{n-1} \equiv 1 \pmod n$ for **all** positive integers b with $\gcd(b, n) = 1$ is called a **Carmichael number**. **Example**: $561 = 3 \cdot 11 \cdot 17$ is a Carmichael number.

- Verify: If $\gcd(b, 561) = 1$, then $\gcd(b, 3) = 1$, $\gcd(b, 11) = 1$, $\gcd(b, 17) = 1$.
- By Fermat's Little Theorem: $b^2 \equiv 1 \pmod 3$, $b^{10} \equiv 1 \pmod{11}$, $b^{16} \equiv 1 \pmod{17}$.
- $b^{560} = (b^2)^{280} \equiv 1 \pmod 3$, $b^{560} = (b^{10})^{56} \equiv 1 \pmod{11}$, $b^{560} = (b^{16})^{35} \equiv 1 \pmod{17}$.
- Hence $b^{560} \equiv 1 \pmod{561}$.

4.4.9 9. Primitive Roots (原根)

Definition: A **primitive root modulo a prime** p is an integer r in $\mathbb{Z}_p$ such that every nonzero element of $\mathbb{Z}_p$ is a power of r.

- 原根 r 的幂可以生成 $\mathbb{Z}_p$ 中所有非零元素：$\{r^1, r^2, \ldots, r^{p-1}\} = \mathbb{Z}_p \setminus \{0\}$。

Example: 2 is a primitive root modulo 11. Powers of 2 modulo 11: $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 5$, $2^5 = 10$, $2^6 = 9$, $2^7 = 7$, $2^8 = 3$, $2^9 = 6$, $2^{10} = 1$. These cover all of $\{1, 2, \ldots, 10\}$, so 2 is a primitive root of 11. 3 is NOT a primitive root of 11: $3^1 = 3$, $3^2 = 9$, $3^3 = 5$, $3^4 = 4$, $3^5 = 1$, and then repeats – only 5 distinct values. **Fact**: For every prime p, there exists a primitive root modulo p. **General definition**: If the multiplicative order of g modulo m equals $\varphi(m)$ (the Euler totient function), then g is a primitive root modulo m.

4.4.10 10. Discrete Logarithms (离散对数)

Suppose p is prime and r is a primitive root modulo p. **Definition**: If a is an integer between 1 and $p - 1$ inclusive, and $r^e \equiv a \pmod p$, then e is called the **discrete logarithm of** a **modulo** p **to the base** r, denoted:

$$\log_r a = e$$

Example 1: $\log_2 3 = 8$ in modulo 11, because $2^8 = 256 \equiv 3 \pmod{11}$. **Example 2**: $\log_2 5 = 4$ in modulo 11, because $2^4 = 16 \equiv 5 \pmod{11}$. **Asymmetry (不对称性)**:

- 已知 p, r, e 求 a 容易（模幂运算是高效的）。
- 已知 p, r, a 求 e 困难（离散对数问题被认为在计算上是困难的）。
- 这一不对称性是许多密码系统（如 Diffie-Hellman 密钥交换）的安全基础。

4.4.11 小结与考点提示

1. **Linear Congruence** $ax \equiv b \pmod m$:
- 有解当且仅当 $\gcd(a, m) \mid b$
- 若 $\gcd(a, m) = 1$，解唯一模 m（可用逆元求解）
1. **Finding Inverses (求逆元)**:
- 用 Euclidean Algorithm + back substitution
- 是解同余方程和 RSA 的基础
1. **Chinese Remainder Theorem (CRT，重点)**:
- 模数两两互质时可用 CRT
- 构造方法：$M_k = M/m_k$，求 y_k 为 M_k 模 m_k 的逆元
- 解为 $x = \sum a_k M_k y_k$
- 回代法也要掌握
1. **Fermat's Little Theorem**:
- $a^{p-1} \equiv 1 \pmod p$（p 为素数，$p \nmid a$）
- 用于加速大数模幂计算
1. **Pseudoprimes & Carmichael Numbers**:
- 理解概念即可，考试较少考
1. **Primitive Roots & Discrete Logarithms**:
- 原根的定义，离散对数的定义
- 理解"已知底数和指数求幂容易，已知底数和幂求指数困难"（单向函数）

Exam tips:

- **English terms**: linear congruence, inverse modulo m, Chinese Remainder Theorem, Fermat's Little Theorem, pseudoprime, primitive root, discrete logarithm.
- CRT 的构造法是考试重点 – 记住 $x = \sum a_k M_k y_k$。
- 费马小定理常用于简化 $a^n \bmod p$ 的计算（p 为素数）。

4.5 Chapter 4.5: Applications of Congruences and Cryptography (同余的应用与密码学)

4.5.1 1. Hashing Functions (哈希函数)

Definition: A **hashing function** h assigns memory location $h(k)$ to the record that has k as its key. **Common hashing function**: $h(k) = k \bmod m$, where m is the number of memory locations.

- 常用哈希函数：用 key 模内存位置数来确定存储位置。

Example: Let $h(k) = k \bmod 111$ assign records with social security number keys:

- $h(064212848) = 064212848 \bmod 111 = 14$
- $h(037149212) = 037149212 \bmod 111 = 65$
- $h(107405723) = 107405723 \bmod 111 = 14$ – **collision (冲突)**!

Collision: When two different keys map to the same location.

- **Collision resolution**: assign the record to the next free location.
- **Linear probing**: $h(k,i) = (h(k) + i) \bmod m$, where i runs from 0 to $m-1$.

4.5.2 2. Pseudorandom Numbers (伪随机数)

Linear Congruential Method (线性同余法): One commonly used procedure for generating pseudorandom numbers. Four integers are needed:

- **Modulus** m $(m > 0)$
- **Multiplier** a $(0 \le a < m)$
- **Increment** c $(0 \le c < m)$
- **Seed** x_0 $(0 \le x_0 < m)$

Generate sequence $\{x_n\}$ recursively:

$$x_{n+1} = (ax_n + c) \bmod m$$

If pseudorandom numbers between 0 and 1 are needed, divide by m: x_n/m. **Example**: Find the sequence generated with $m = 9$, $a = 7$, $c = 4$, $x_0 = 3$. $x_1 = 7 \cdot 3 + 4 = 25 \equiv 7 \pmod 9$ $x_2 = 7 \cdot 7 + 4 = 53 \equiv 8 \pmod 9$ $x_3 = 7 \cdot 8 + 4 = 60 \equiv 6 \pmod 9$ $x_4 = 7 \cdot 6 + 4 = 46 \equiv 1 \pmod 9$ $x_5 = 7 \cdot 1 + 4 = 11 \equiv 2 \pmod 9$ $x_6 = 7 \cdot 2 + 4 = 18 \equiv 0 \pmod 9$ $x_7 = 7 \cdot 0 + 4 = 4 \pmod 9$ $x_8 = 7 \cdot 4 + 4 = 32 \equiv 5 \pmod 9$ $x_9 = 7 \cdot 5 + 4 = 39 \equiv 3 \pmod 9$ (repeats) Sequence: $3, 7, 8, 6, 1, 2, 0, 4, 5, 3, 7, 8, \ldots$ When $c = 0$, it's called a **pure multiplicative generator (纯乘性生成器)**.

- e.g., modulus $2^{31} - 1$, multiplier $7^5 = 16807$ – generates $2^{31} - 2$ numbers before repeating.

4.5.3 3. Check Digits (校验位)

UPC (Universal Product Code)

UPC 有 12 位十进制数字，最后一位是校验位，由以下同余式确定：

$$3x_1 + x_2 + 3x_3 + x_4 + 3x_5 + x_6 + 3x_7 + x_8 + 3x_9 + x_{10} + 3x_{11} + x_{12} \equiv 0 \pmod{10}$$

- 奇数位（从左边数第 1、3、5... 位）系数为 3，偶数位系数为 1。

Example (a): First 11 digits are 79357343104. Find the check digit. $3 \cdot 7 + 9 + 3 \cdot 3 + 5 + 3 \cdot 7 + 3 + 3 \cdot 4 + 3 + 3 \cdot 1 + 0 + 3 \cdot 4 + x_{12} \equiv 0 \pmod{10} \Rightarrow 21 + 9 + 9 + 5 + 21 + 3 + 12 + 3 + 3 + 0 + 12 + x_{12} \equiv 0 \pmod{10} \Rightarrow 98 + x_{12} \equiv 0 \pmod{10} \Rightarrow x_{12} \equiv 2 \pmod{10}$ So the check digit is 2. **Example (b)**: Is 041331021641 a valid UPC? $3 \cdot 0 + 4 + 3 \cdot 1 + 3 + 3 \cdot 3 + 1 + 3 \cdot 0 + 2 + 3 \cdot 1 + 6 + 3 \cdot 4 + 1 = 0 + 4 + 3 + 3 + 9 + 1 + 0 + 2 + 3 + 6 + 12 + 1 = 44 \equiv 4 \not\equiv 0 \pmod{10}$ Hence it is **not** a valid UPC.

ISBN-10 (International Standard Book Number)

10 位数字，第 10 位是校验位：

$$\sum_{i=1}^{10} i \cdot x_i \equiv 0 \pmod{11}$$

即 $1x_1 + 2x_2 + 3x_3 + \cdots + 10x_{10} \equiv 0 \pmod{11}$.

- 当校验位为 10 时，用 X 表示。

Example (a): First 9 digits are 007288008. Find the check digit x_{10}. $1 \cdot 0 + 2 \cdot 0 + 3 \cdot 7 + 4 \cdot 2 + 5 \cdot 8 + 6 \cdot 8 + 7 \cdot 0 + 8 \cdot 0 + 9 \cdot 8 + 10x_{10} \equiv 0 \pmod{11} \Rightarrow 0 + 0 + 21 + 8 + 40 + 48 + 0 + 0 + 72 + 10x_{10} \equiv 0 \pmod{11} \Rightarrow 189 + 10x_{10} \equiv 0 \pmod{11} \Rightarrow 2 + 10x_{10} \equiv 0 \pmod{11}$ (since $189 \equiv 2 \pmod{11}$) $\Rightarrow 10x_{10} \equiv 9 \pmod{11} \Rightarrow x_{10} \equiv 2 \pmod{11}$ (since inverse of 10 mod 11 is 10) So the check digit is 2. **Example (b)**: Is 084930149X a valid ISBN-10? ($X = 10$) $1 \cdot 0 + 2 \cdot 8 + 3 \cdot 4 + 4 \cdot 9 + 5 \cdot 3 + 6 \cdot 0 + 7 \cdot 1 + 8 \cdot 4 + 9 \cdot 9 + 10 \cdot 10 = 0 + 16 + 12 + 36 + 15 + 0 + 7 + 32 + 81 + 100 = 299 \equiv 2 \not\equiv 0 \pmod{11}$ Hence it is **not** a valid ISBN-10.

Note: Check digits can detect single-digit errors and transposition errors (数字交换错误).

4.5.4 4. Public Key Cryptography (公钥密码学)

Background

- **Private key (对称密钥)**: Encryption key = Decryption key. 发送方和接收方必须共享密钥。
- **Public key (公钥密码)**: Invented in 1970s. Encryption key is public; decryption key is private.
- 用 Alice 公钥加密，只能用 Alice 私钥解密。实现了定向发送密文。

4.5.5 5. RSA Cryptosystem (RSA 密码系统)

Invented by: Rivest, Shamir, Adleman (MIT, 1977). Also discovered earlier by Clifford Cocks (UK government). **RSA 的核心安全基础**：大整数因子分解的困难性 (factoring large integers into primes is computationally infeasible).

Key Generation (密钥生成)

1. Choose two large primes p and q (e.g., 300+ digits).
2. Compute $n = pq$ (the modulus).
3. Compute $\varphi(n) = (p-1)(q-1)$ (Euler's totient).

4. Choose an exponent e such that $\gcd(e, (p-1)(q-1)) = 1$.
5. Compute d, the inverse of e modulo $(p-1)(q-1)$ (i.e., $ed \equiv 1 \pmod{(p-1)(q-1)}$).

- **Public key (公钥)**: (n, e)
- **Private key (私钥)**: d (also keep p, q secret)

Encryption (加密)

To encrypt a message M:

1. Translate letters to numbers: $00 = A, 01 = B, \ldots, 25 = Z$.
2. Concatenate into digit strings. Divide into blocks of N digits, where N is the largest even number with $2N$ digits $\le n$.
3. Each block m (as integer) is encrypted: $c = m^e \bmod n$.

Example: Encrypt "STOP" using RSA with $n = 2537$ $(p = 43, q = 59)$, $e = 13$.

- $n = 43 \cdot 59 = 2537$, $e = 13$, $\gcd(13, 42 \cdot 58 = 2436) = 1$.
- "STOP" $\to$ 18 19 14 15.
- Blocks of 4 digits (since $2525 < 2537 < 252525$): 1819, 1415.
- Encrypt: $1819^{13} \bmod 2537 = 2081$, $1415^{13} \bmod 2537 = 2182$.
- Ciphertext: 2081 2182.

Decryption (解密)

To decrypt ciphertext c, use private key d:

$$m = c^d \bmod n$$

Why it works:

- $c \equiv m^e \pmod n$
- $c^d \equiv m^{ed} \equiv m^{k\varphi(n)+1} \equiv (m^{\varphi(n)})^k \cdot m \equiv 1^k \cdot m \equiv m \pmod n$
- By Euler's Theorem: if $\gcd(m, n) = 1$, then $m^{\varphi(n)} \equiv 1 \pmod n$.

Example: Decrypt 0981 0461 (encrypted with $n = 2537$, $e = 13$ from previous example).

- d is the inverse of 13 modulo $2436 = 42 \cdot 58$. $d = 937$ (since $13 \cdot 937 = 12181 = 5 \cdot 2436 + 1$).
- Decrypt: $0981^{937} \bmod 2537 = 0704$, $0461^{937} \bmod 2537 = 1115$.
- 0704 1115 $\to$ letters: $07 = H$, $04 = E$, $11 = L$, $15 = P$ $\to$ "HELP".

RSA 安全性：已知 n 但不知道 p, q，直接计算 $\varphi(n)$ 或 d，或从 $c = m^e \bmod n$ 反求 m，都被认为在计算上是困难的（依赖大整数分解难题）。

4.5.6 6. Cryptographic Protocols: Key Exchange (密钥交换协议)

Diffie-Hellman Key Agreement (迪菲-赫尔曼密钥交换)

Goal: Two parties (Alice and Bob) exchange a shared secret key over an insecure channel. **Protocol**:

1. Alice and Bob agree on a prime p and a primitive root a of p (public).
2. Alice chooses a secret integer k_1, sends $a^{k_1} \bmod p$ to Bob.
3. Bob chooses a secret integer k_2, sends $a^{k_2} \bmod p$ to Alice.
4. Alice computes $(a^{k_2})^{k_1} \bmod p = a^{k_1 k_2} \bmod p$.
5. Bob computes $(a^{k_1})^{k_2} \bmod p = a^{k_1 k_2} \bmod p$.
6. Shared key: $a^{k_1 k_2} \bmod p$.

Security:

- Eavesdropper sees $a^{k_1} \bmod p$ and $a^{k_2} \bmod p$, but cannot find k_1, k_2 (discrete logarithm problem).
- 窃听者只知道 a^{k_1} 和 a^{k_2}，但要从中求出 k_1, k_2 需要解离散对数问题，这在计算上不可行。

4.5.7 7. Digital Signatures (数字签名)

Goal: Ensure that a message came from the purported sender (认证来源). **RSA Digital Signature**:

- Alice's public key: (n, e); private key: d.
- To sign message M:

1. Alice applies her **decryption function** $D_{(n,e)}(M) = M^d \bmod n$ to the blocks.
2. She sends the result to recipients.

- To verify:

1. Recipients apply Alice's **encryption function** $E_{(n,e)}(x) = x^e \bmod n$.
2. Result is the original message since $E_{(n,e)}(D_{(n,e)}(M)) = M$.

Key idea:

- 用 Alice 私钥签名（加密），用 Alice 公钥验证（解密）。
- 因为私钥只有 Alice 知道，所以可以认证信息来源。
- 注意这里的" 加密" 和" 解密" 角色互换：签名时用的是私钥（通常用于解密），验证时用的是公钥（通常用于加密）。

Combining Digital Signatures and Encryption: For a message that is both **authenticated** (from Alice) and **confidential** (only for Bob):

1. Alice applies Bob's encryption function $E_{(n',e')}$ (用 Bob 公钥加密 = 只有 Bob 能解密).
2. Then applies her decryption function $D_{(n,e)}$ (用 Alice 私钥签名).
3. Bob applies Alice's encryption function $E_{(n,e)}$ (用 Alice 公钥验证签名).
4. Then applies his decryption function $D_{(n',e')}$ (用 Bob 私钥解密).

$$D_{(n',e')}(E_{(n,e)}(D_{(n,e)}(E_{(n',e')}(M)))) = M$$

4.5.8 小结与考点提示

1. **Hashing Functions**: $h(k) = k \bmod m$，处理冲突（collision）的方法
2. **Pseudorandom Numbers**: 线性同余法 $x_{n+1} = (ax_n + c) \bmod m$
3. **Check Digits**: UPC（模 10，奇数位乘 3）、ISBN-10（模 11，加权求和），会计算校验位和验证
4. **RSA Cryptosystem（重中之重）**：

Component	Description
Key generation	选大素数 p, q，$n = pq$，选 e 与 $(p-1)(q-1)$ 互质，$d = e^{-1} \bmod (p-1)(q-1)$
Encryption	$c = m^e \bmod n$
Decryption	$m = c^d \bmod n$
Public key	(n, e)
Private key	d

1. **Diffie-Hellman Key Exchange**:
- 共享 p 和原根 a，各自选秘密数，交换 $a^k \bmod p$，计算共同密钥 $a^{k_1 k_2} \bmod p$
- 安全性基于离散对数问题的困难性

1. **Digital Signatures**:
- 用私钥签名（加密），公钥验证（解密）
- 结合加密和签名：先公钥加密（保密），再私钥签名（认证）

Exam tips:

- **English terms**: hashing function, collision, pseudorandom numbers, linear congruential method, check digit, public key cryptography, RSA, modulus, encryption, decryption, digital signature, Diffie-Hellman key exchange.
- RSA 考试题典型模式：给 p, q, e，求 n 和 d，然后加解密消息。
- 理解 RSA 为什么安全："factoring large numbers is hard"。
- 数字签名中，"sign with private key, verify with public key" – 角色反转。

第五章 Chapter 5: Induction & Recursion (归纳与递归)

5.1 Chapter 5.1: Mathematical Induction (数学归纳法)

5.1.1 1. Principle of Mathematical Induction (数学归纳法原理)

Mathematical Induction (数学归纳法): A proof technique used to prove that a propositional function $P(n)$ is true for all positive integers n (or all integers $n \geq b$). 类比：**Climbing an Infinite Ladder (爬无限梯子)**：

1. 我们可以到达梯子的第一级。
2. 如果我们能到达某一级，那么就能到达下一级。

由 (1) 和 (2)，我们可以到达任意一级。

Formal Definition (正式定义)

To prove that $P(n)$ is true for all positive integers n, complete **two steps**:

Step	Description
Basis Step (基础步骤)	Show that $P(1)$ is true.
Inductive Step (归纳步骤)	Show that $P(k) \to P(k+1)$ is true for all positive integers k.

- **Inductive Hypothesis (归纳假设)**: The assumption that $P(k)$ is true for an arbitrary positive integer k.
- **Conclusion**: $P(n)$ is true for all positive integers n.

Rule of inference form:

$$P(1) \wedge \forall k(P(k) \to P(k+1)) \to \forall n P(n)$$

重要：在归纳步骤中，我们**并非**假设 $P(k)$ 对所有正整数都成立，而是**假设对于某一个特定的** k，$P(k)$ **成立**，然后证明 $P(k+1)$ 也成立。

Starting at an Integer Other than 1 (不从 1 开始)

证明可以从任意整数 b 开始：

- **Basis Step**: Prove $P(b)$ is true.
- **Inductive Step**: Prove $P(k) \to P(k+1)$ for all $k \geq b$.
- **Conclusion**: $P(n)$ is true for all integers $n \geq b$.

5.1.2 2. Validity of Mathematical Induction —Well-Ordering Property (良序性质)

Well-Ordering Property (良序性质): Every nonempty subset of the set of positive integers has a **least element (最小元素)**. 数学归纳法的正确性正是由良序性质保证的：

- 假设 $P(1)$ 真且 $\forall k(P(k) \to P(k+1))$ 真。
- 假设存在某个 n 使 $P(n)$ 假。设 S 是所有使 $P(n)$ 假的 n 的集合。
- 由良序性质，S 有最小元素 m。因为 $P(1)$ 真，所以 $m > 1$。
- 于是 $m-1$ 是正整数且 $m-1 \notin S$，所以 $P(m-1)$ 真。
- 但由归纳步骤，$P(m-1) \to P(m)$，所以 $P(m)$ 真，矛盾。
- 因此 $P(n)$ 对所有正整数 n 都成立。

结论：Mathematical Induction, Strong Induction, and Well-Ordering Property 三者是**等价的 (equivalent)**。

5.1.3 3. Examples of Proof by Mathematical Induction (例题)

Example 1: Summation Formula (求和公式)

Statement: $1+2+3+\cdots+n = \dfrac{n(n+1)}{2}$ for all positive integers n. **Solution**: Let $P(n)$ be the proposition that $1+2+\cdots+n = \frac{n(n+1)}{2}$.

- **Basis Step**: $P(1)$ is true, since $1 = \frac{1(1+1)}{2} = 1$.
- **Inductive Step**: Assume $P(k)$ is true (inductive hypothesis):

$$1 + 2 + \cdots + k = \frac{k(k+1)}{2}$$

Then show $P(k+1)$:

$$1+2+\cdots+k+(k+1)=\frac{k(k+1)}{2}+(k+1)$$

$$=(k+1)\left(\frac{k}{2}+1\right)=(k+1)\frac{k+2}{2}=\frac{(k+1)(k+2)}{2}$$

Thus $P(k+1)$ holds.

Therefore, $P(n)$ is true for all positive integers n.

Example 2: Sum of First n Positive Odd Integers (前 n 个正奇数的和)

Statement: $1+3+5+\cdots+(2n-1)=n^2$ for all positive integers n. **Observation**:

- $1=1^2$
- $1+3=4=2^2$
- $1+3+5=9=3^2$
- $1+3+5+7=16=4^2$
- $1+3+5+7+9=25=5^2$

Conjecture: $1+3+5+\cdots+(2n-1)=n^2$ **Proof by induction**:

- **Basis Step**: $P(1)$: $1=1^2$, true.
- **Inductive Step**: Assume $P(k)$: $1+3+5+\cdots+(2k-1)=k^2$. Then:

$$1+3+5+\cdots+(2k-1)+(2k+1)$$

$$=[1+3+\cdots+(2k-1)]+(2k+1)$$

$$=k^2+(2k+1)\quad\text{(by inductive hypothesis)}$$

$$=(k+1)^2$$

Thus $P(k+1)$ holds.

Therefore, $1+3+5+\cdots+(2n-1)=n^2$ for all positive integers n.

Example 3: Proving Inequalities (证明不等式)

Statement: $n<2^n$ for all positive integers n. **Solution**: Let $P(n)$ be the proposition $n<2^n$.

- **Basis Step**: $P(1)$: $1<2^1=2$, true.
- **Inductive Step**: Assume $P(k)$: $k<2^k$ for an arbitrary positive integer k. Show $P(k+1)$: $k+1<2^{k+1}$.

$$k+1<2^k+1\le 2^k+2^k=2\cdot 2^k=2^{k+1}$$

Therefore $P(k+1)$ holds.

Thus $n<2^n$ for all positive integers n.

Example 4: Proving $n<n!$

Statement: $n<n!$ for every integer $n\ge 4$. **Solution**: Let $P(n)$ be $n<n!$.

- **Basis Step**: $P(4)$: $4<4!=24$, true. (Note: $P(0),P(1),P(2),P(3)$ are all false, so basis starts at $n=4$.)
- **Inductive Step**: Assume $P(k)$: $k<k!$ for $k\ge 4$. Show $P(k+1)$: $k+1<(k+1)!$.

$$k+1<k!+1\le k!+k!\quad(\text{since } k!\ge 1)$$

$$=2\cdot k!<(k+1)k!\quad(\text{since } k+1>2 \text{ for } k\ge 4)$$

$$=(k+1)!$$

Thus $n<n!$ for all $n\ge 4$.

Example 5: Divisibility (整除性)

Statement: n^3-n is divisible by 3 for every positive integer n. **Solution**: Let $P(n)$: n^3-n is divisible by 3.

- **Basis Step**: $P(1)$: $1^3-1=0$, divisible by 3, true.
- **Inductive Step**: Assume $P(k)$: k^3-k is divisible by 3. Show $P(k+1)$:

$$(k+1)^3-(k+1)=(k^3+3k^2+3k+1)-(k+1)$$

$$=(k^3-k)+3(k^2+k)$$

By inductive hypothesis, (k^3-k) is divisible by 3, and $3(k^2+k)$ is clearly divisible by 3.

Therefore $(k+1)^3-(k+1)$ is divisible by 3.

Thus n^3-n is divisible by 3 for all positive integers n.

Example 6: Number of Subsets of a Finite Set (有限集的子集数)

Statement: If S is a finite set with n elements ($n\ge 0$), then S has 2^n subsets. **Solution**: Let $P(n)$ be the proposition: a set with n elements has 2^n subsets.

- **Basis Step**: $P(0)$: The empty set has only itself as a subset, and $2^0=1$, true.
- **Inductive Step**: Assume $P(k)$ is true (every set with k elements has 2^k subsets). Let T be a set with $k+1$ elements. Write $T=S\cup\{a\}$ where $a\in T$ and $S=T\setminus\{a\}$, so $|S|=k$.

  For each subset X of S, there are exactly **two** subsets of T: X and $X\cup\{a\}$.

  By inductive hypothesis, S has 2^k subsets, so T has $2\cdot 2^k=2^{k+1}$ subsets.

Thus a set with n elements has 2^n subsets.

Example 7: Tiling Checkerboards (铺棋盘)

Statement: Every $2^n\times 2^n$ checkerboard with one square removed can be tiled using **right triominoes (右三角板)** (L-shaped tiles covering 3 squares). **Solution**: Let $P(n)$: every $2^n\times 2^n$ board with one square removed can be tiled.

- **Basis Step**: $P(1)$: A 2×2 board with one square removed can be tiled with one right triomino. (Four possible cases, all work.)
- **Inductive Step**: Assume $P(k)$ holds for some $k\ge 1$. Consider a $2^{k+1}\times 2^{k+1}$ board with one square removed.

  Split it into four $2^k\times 2^k$ boards. One of them has the removed square; tile it by inductive hypothesis.

  For the other three, remove the square at the corner closest to the center. By inductive hypothesis, each can be tiled.

  Then cover the three adjacent squares at the center with one triomino.

  Thus the entire $2^{k+1}\times 2^{k+1}$ board is tiled.

Therefore $P(n)$ holds for all positive integers n.

5.1.4 4. An Incorrect "Proof" by Mathematical Induction (错误证明示例)

Example: "Every set of n lines in the plane, no two parallel, meet in a common point."

Flawed "proof": - **Basis Step** ($n=2$): Any two non-parallel lines meet at a point. True. - **Inductive Step**: Assume any k lines meet at a point. For $k+1$ lines, by inductive hypothesis the first k lines meet at p_1, and the last k lines meet at p_2. Since two points determine a line, $p_1=p_2$ must lie on all $k+1$ lines.

Error (错误原因): The inductive step fails for $k=2$ (i.e., proving $P(2)\to P(3)$). For three lines, the first two meet at p_1 and the last two meet at p_2 —they need not be the same point because only the second line is common to both sets. The argument assumes the lines are all distinct and each pair shares a different common point.

教训: 归纳步骤必须在从基础步骤开始的**所有** k 上都成立，不能有断裂。

5.1.5 5. Guidelines for Proofs by Mathematical Induction (证明指南)

1. **明确表述** $P(n)$ —明确写出命题 $P(n)$ 是什么。
2. **证明基础步骤**—$P(1)$ (或 $P(b)$) 通常很简单，但必须验证。
3. **明确写出归纳假设**—明确写出"Assume $P(k)$ is true for an arbitrary positive integer k"。
4. **证明** $P(k+1)$ —利用归纳假设推导 $P(k+1)$。在推导中明确指出使用归纳假设的位置。
5. **下结论**—最后说"Therefore, by mathematical induction, $P(n)$ is true for all positive integers n"。
6. **小心边界**—如果基础步骤不是 $n=1$，请明确指出。

5.1.6 6. Summary and Exam Tips (小结与考点提示)

概念	英文	要点
数学归纳法	Mathematical Induction	两步：Basis + Inductive
基础步骤	Basis Step	验证 $P(1)$ (或 $P(b)$) 为真
归纳假设	Inductive Hypothesis	假设 $P(k)$ 真
归纳步骤	Inductive Step	证明 $P(k)\to P(k+1)$
良序性质	Well-Ordering Property	正整数集的非空子集有最小元素
等价性	Equivalence	MI, Strong Induction, Well-Ordering 三者等价

考试重点

1. **写出完整的归纳证明**—Basis Step + Inductive Hypothesis + Inductive Step + Conclusion。每一步都要写清楚。
2. **求和公式**：$1+2+\cdots+n=\frac{n(n+1)}{2}$ 和奇数求和 $1+3+\cdots+(2n-1)=n^2$ 是经典例题。
3. **不等式证明**：常用技巧是利用归纳假设后放大 (如 $k+1<2^k+1\le 2^k+2^k$)。
4. **整除性证明**：将 $(k+1)^3-(k+1)$ 展开，分离出 k^3-k 项。

5. **错误辨析**：能识别归纳证明中的逻辑漏洞（如上述直线交点例子）。
6. **与 Strong Induction 的区别**：MI 只假设 $P(k)$，SI 假设 $P(1) \wedge P(2) \wedge \cdots \wedge P(k)$。

常见错误

- 跳过了 Basis Step（没有基础步骤，归纳无法启动）。
- 在归纳步骤中错误地假设了 $P(k)$ 对所有 k 都成立。
- 归纳步骤只在某些 k 上成立，而非所有 k。
- 基础步骤和归纳步骤之间的" 断裂"（如直线交点例子中 $P(2) \not\to P(3)$）。

Notes prepared for DMA course review. Key terms are kept in English for exam readiness.

5.2 Chapter 5.2: Strong Induction (强归纳法)

5.2.1 1. Strong Induction (强归纳法) —Definition

Strong Induction (强归纳法), also called the **Second Principle of Mathematical Induction (第二数学归纳原理)** or **Complete Induction (完全归纳法)**. To prove that $P(n)$ is true for all positive integers n, complete **two steps**:

Step	Description
Basis Step (基础步骤)	Verify that $P(1)$ is true.
Inductive Step (归纳步骤)	Show that $[P(1) \wedge P(2) \wedge \cdots \wedge P(k)] \to P(k+1)$ holds for all positive integers k.

Key Difference from Ordinary Mathematical Induction (与普通归纳法的关键区别)

Aspect	Ordinary Induction (MI)	Strong Induction (SI)
Inductive Hypothesis	Assume **only** $P(k)$	Assume **all** $P(1), P(2), \ldots, P(k)$
What you prove	$P(k) \to P(k+1)$	$[P(1) \wedge \cdots \wedge P(k)] \to P(k+1)$
When to use	When $P(k+1)$ depends only on $P(k)$	When $P(k+1)$ depends on some smaller $P(j)$ where $j \leq k$

Infinite Ladder Analogy (无限梯子类比)

Strong induction tells us we can reach all rungs if:

1. We can reach the **first** rung.
2. For every integer k, if we can reach the **first** k **rungs**, then we can reach the $(k+1)$st rung.

5.2.2 2. Strong Induction vs. Ordinary Induction

- 我们**总是可以**用 Strong Induction 代替 Ordinary Induction。
- 但如果 Ordinary Induction 已经够用，就没必要用 Strong Induction（它更简单）。
- 实际上，**Mathematical Induction, Strong Induction, and the Well-Ordering Property 三者是等价的**（见教材练习 41-43）。
- 有时只有其中一种方法能直接解决问题。

5.2.3 3. Examples of Proof by Strong Induction (例题)

Example 1: Reaching Rungs of a Ladder (梯子问题)

Problem: Suppose we can reach the first and second rungs of an infinite ladder, and we know that if we can reach a rung, then we can reach two rungs higher. Prove that we can reach every rung. **Solution** (using Strong Induction): Let $P(n)$: we can reach the nth rung.

- **Basis Step**: $P(1)$ is true (we can reach the first rung). Also $P(2)$ is true.
- **Inductive Step**: Assume $P(1), P(2), \ldots, P(k)$ are true for $k \geq 2$. To reach the $(k+1)$st rung: since we can reach the $(k-1)$st rung by the inductive hypothesis, and the rule says "if we can reach a rung, we can reach two rungs higher", we can reach $(k-1)+2 = k+1$. Thus $P(k+1)$ holds.

Therefore, we can reach all rungs of the ladder.

Note: Ordinary Induction cannot easily prove this because $P(k+1)$ depends on $P(k-1)$, not $P(k)$.

Example 2: Fundamental Theorem of Arithmetic (算术基本定理)

Statement: Every integer $n > 1$ can be written as a product of primes. **Solution**: Let $P(n)$: n can be written as a product of primes.

- **Basis Step**: $P(2)$ is true since 2 itself is prime.
- **Inductive Step**: Assume $P(j)$ is true for all integers j with $2 \leq j \leq k$. Show $P(k+1)$ is true.
- **Case 1**: $k+1$ is prime. Then $P(k+1)$ is trivially true.
- **Case 2**: $k+1$ is composite. Then $k+1 = a \cdot b$ where $2 \leq a \leq b < k+1$. By the inductive hypothesis, both a and b can be written as products of primes.

  Therefore $k+1 = a \cdot b$ can also be written as a product of primes.

Thus every integer $n > 1$ can be expressed as a product of primes. (唯一性在 Section 4.3 中证明。)

关键：这里必须用 Strong Induction，因为 a 和 b 是比 $k+1$ 更小的数，但不一定是 k。

Example 3: Postage Stamp Problem (邮票问题)

Problem: Prove that every amount of postage of **12 cents or more** can be formed using just **4-cent** and **5-cent** stamps.

Solution 1: Using Strong Induction Let $P(n)$: postage of n cents can be formed using 4-cent and 5-cent stamps.

- **Basis Step**: Verify $P(12), P(13), P(14), P(15)$.
- $P(12)$: three 4-cent stamps.
- $P(13)$: two 4-cent stamps + one 5-cent stamp.
- $P(14)$: one 4-cent stamp + two 5-cent stamps.
- $P(15)$: three 5-cent stamps.

- **Inductive Step**: Assume $P(j)$ holds for all j with $12 \le j \le k$, where $k \ge 15$. Show $P(k+1)$: Since $k+1 = (k-3)+4$ and $k-3 \ge 12$, by the inductive hypothesis $P(k-3)$ holds.
  Add one 4-cent stamp to the postage for $k-3$ cents to get $k+1$ cents.
  Thus $P(k+1)$ holds.

Therefore $P(n)$ holds for all $n \ge 12$.

Solution 2: Using Ordinary Induction (alternative) Let $P(n)$: postage of n cents can be formed.

- **Basis Step**: $P(12)$: three 4-cent stamps.
- **Inductive Step**: Assume $P(k)$ holds for $k \ge 12$. Show $P(k+1)$.
- If a 4-cent stamp was used, replace it with a 5-cent stamp (net +1 cent).
- If no 4-cent stamps (all 5-cent stamps), then at least three 5-cent stamps were used (since $k \ge 12$). Replace three 5-cent stamps with four 4-cent stamps (net +1 cent). Thus $P(k+1)$ holds.

比较：Strong Induction 的解法更直观自然。

Example 4: Triangulation of a Simple Polygon (简单多边形的三角形化)

Lemma 1: Every simple polygon has an **interior diagonal (内部对角线)**. **Theorem 1**: A simple polygon with n sides ($n \ge 3$) can be **triangulated (三角形化)** into $n-2$ triangles. **Proof** (using Strong Induction): Let $T(n)$: a simple polygon with n sides can be triangulated into $n-2$ triangles.

- **Basis Step**: $T(3)$ is true (a triangle is already triangulated into $3-2=1$ triangle).
- **Inductive Step**: Assume $T(j)$ is true for all j with $3 \le j \le k$. Show $T(k+1)$. Let P be a simple polygon with $k+1$ sides.
  By Lemma 1, P has an interior diagonal ab. This diagonal splits P into two simple polygons Q (with s sides, $3 \le s \le k$) and R (with t sides, $3 \le t \le k$).
  By the inductive hypothesis, Q can be triangulated into $s-2$ triangles and R into $t-2$ triangles.
  Total triangles: $(s-2)+(t-2) = (s+t)-4 = (k+3)-4 = k-1$.
  Note that $s+t = (k+1)+2$ because the diagonal ab is counted in both Q and R.

Thus every simple polygon with n sides ($n \ge 3$) can be triangulated into $n-2$ triangles.

5.2.4 4. Well-Ordering Property (良序性质)

Definition

Well-Ordering Property: Every nonempty set of **nonnegative integers** has a **least element (最小元素)**.

Properties

- $\mathbb{N}$ (natural numbers) is **well ordered** under $\le$.
- The set of finite strings over an alphabet using **lexicographic ordering (字典序)** is well ordered.
- $\mathbb{Z}$ (integers) is **NOT** well ordered under $\le$ (no smallest element).
- The open interval $(0,1)$ is **NOT** well ordered (no least element).

Equivalence

- Mathematical Induction, Strong Induction, and Well-Ordering Property are **all equivalent (三者等价)**.
- 任意一个成立，其余两个也必然成立。

5.2.5 5. Applications of Well-Ordering Property

Example 1: Division Algorithm (除法算法)

Statement: If a is an integer and d is a positive integer, then there exist **unique** integers q and r with $0 \le r < d$ such that $a = dq + r$.

Proof using Well-Ordering: Let S be the set of nonnegative integers of the form $a - dq$, where q is an integer.

- S is nonempty (choose q appropriately so the expression is ≥ 0).
- By the Well-Ordering Property, S has a least element $r = a - dq_0$.
- $r \ge 0$ by definition. Must show $r < d$:
- If $r \ge d$, then $r - d = a - d(q_0+1)$ is a smaller nonnegative element in S, contradicting minimality.
- Therefore $r < d$.
- Uniqueness: exercise.

Thus $a = dq + r$ with $0 \le r < d$.

Example 2: Round-Robin Tournament Cycle (循环赛中的循环)

Problem: In a round-robin tournament, every player plays every other player exactly once, and each match has a winner and loser. Players $p_1, p_2, \ldots, p_m$ form a **cycle (循环)** if p_1 beats p_2, p_2 beats p_3, …, p_m beats p_1. Prove: If there is a cycle of length m ($m \ge 3$), then there must be a **cycle of length 3**. **Proof using Well-Ordering**: Assume there is **no** cycle of three players. Since there is at least one cycle, the set of all positive integers n for which a cycle of length n exists is nonempty. By the Well-Ordering Property, this set has a **least element** k, which by assumption must be > 3. Consider a cycle $p_1, p_2, p_3, \ldots, p_k$. Examine the match between p_1 and

p_3:

- **Case 1**: p_3 beats p_1. Then p_1, p_2, p_3 form a cycle of length 3, contradiction.
- **Case 2**: p_1 beats p_3. Then $p_1, p_3, p_4, \ldots, p_k$ form a cycle of length $k-1$, contradicting the minimality of k.

Therefore there must be a cycle of length 3.

关键技巧：Well-Ordering 常用于反证法，通过取" 最小的反例" 来推导矛盾。

5.2.6 6. Summary and Exam Tips (小结与考点提示)

概念	英文	要点
强归纳法	Strong Induction	假设 $P(1) \wedge \cdots \wedge P(k)$ 证明 $P(k+1)$
完全归纳法	Complete Induction	Strong Induction 的别名
良序性质	Well-Ordering Property	非空子集有最小元素
等价原理	Equivalent Principles	MI, SI, Well-Ordering 三者等价
算术基本定理	Fundamental Theorem of Arithmetic	每个整数可唯一分解为质数乘积

考试重点

1. **何时使用 Strong Induction**：
- 当 $P(k+1)$ 的证明依赖于某个**小于** k **但不是** k 的值时（如 $P(k-1)$，或任意 $j \le k$）。
- 典型例子：质因数分解 $P(k+1)$ 依赖于 $P(a)$ 和 $P(b)$，而 $a, b < k+1$ 但不一定等于 k。

1. **与 Ordinary Induction 的比较**：
- Ordinary Induction：假设 $P(k)$，证明 $P(k+1)$。
- Strong Induction：假设 $P(1), P(2), \ldots, P(k)$，证明 $P(k+1)$。
- SI 更强但不更" 正确"——两者等价。

1. **基础步骤可能不止一个**：如邮票问题验证了 $P(12), P(13), P(14), P(15)$ 四个基本情况。
2. **Well-Ordering 证明模板**：
- 设 S 为反例集合。
- 由 Well-Ordering，S 有最小元素 m。
- 证明 m 不可能是最小的（构造更小的反例或证明 m 不是反例）。
- 导出矛盾，因此 S 为空。

常见错误

- 混淆 Ordinary Induction 和 Strong Induction 的假设范围。
- 基础步骤不充分（如邮票问题只验证了 $P(12)$ 是不够的）。
- 在需要 Strong Induction 时用 Ordinary Induction（导致归纳步骤无法推导）。
- Well-Ordering 证明中忘记验证集合非空。

Notes prepared for DMA course review. Key terms are kept in English for exam readiness.

5.3 Chapter 5.3: Recursive Definitions, Structural Induction & Recursive Algorithms (递归定义、结构归纳法与递归算法)

5.3.1 1. Recursively Defined Functions (递归定义的函数)

Definition: A **recursive (or inductive) definition** of a function consists of two steps:

Step	Description
Basis Step (基础步骤)	Specify the value of the function at **zero** (or some base).
Recursive Step (递归步骤)	Give a rule for finding its value at an integer from its values at **smaller integers**.

递归定义与 Section 2.4 中的 recurrence relations（递推关系）是同一概念，$f(n)$ 等价于序列 $a_0, a_1, a_2, \ldots$ 中的 a_n。

Example 1: Simple Recursive Function

Define f by:

$$f(0) = 3$$

$$f(n+1) = 2f(n) + 3$$

Compute:

- $f(1) = 2 \cdot f(0) + 3 = 2 \cdot 3 + 3 = 9$
- $f(2) = 2 \cdot 9 + 3 = 21$
- $f(3) = 2 \cdot 21 + 3 = 45$
- $f(4) = 2 \cdot 45 + 3 = 93$

Example 2: Factorial Function (阶乘)

Recursive definition:

$$f(0) = 1$$

$$f(n+1) = (n+1) \cdot f(n)$$

Remember: $n! = n \times (n-1) \times \cdots \times 2 \times 1$, and $0! = 1$.

Example 3: Summation

Recursive definition of $\sum_{i=0}^{n} a_i$:

$$\sum_{i=0}^{0} a_i = a_0$$

$$\sum_{i=0}^{n+1} a_i = \left(\sum_{i=0}^{n} a_i\right) + a_{n+1}$$

5.3.2 2. Fibonacci Numbers (斐波那契数列)

Fibonacci Numbers (Leonardo Fibonacci, c.1170–1250): **Recursive Definition**:

$$f_0 = 0$$

$$f_1 = 1$$

$$f_n = f_{n-1} + f_{n-2} \quad \text{for } n \geq 2$$

Sequence: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$

Example: Proving a Property of Fibonacci Numbers (with Strong Induction)

Statement: Show that $f_n > \alpha^{n-2}$ for $n \geq 3$, where $\alpha = \frac{1+\sqrt{5}}{2}$ (the **golden ratio (黄金比例)**). **Proof**: Let $P(n)$: $f_n > \alpha^{n-2}$.

- **Basis Step**:
- $P(3)$: $f_3 = 2 > \alpha^1 = (1+\sqrt{5})/2 \approx 1.618$, true.
- $P(4)$: $f_4 = 3 > \alpha^2 = ((1+\sqrt{5})/2)^2 \approx 2.618$, true.
- **Inductive Step**: Assume $P(j)$ holds for all j with $3 \leq j \leq k$, where $k \geq 4$. Show $P(k+1)$: $f_{k+1} > \alpha^{(k+1)-2} = \alpha^{k-1}$.

$$f_{k+1} = f_k + f_{k-1}$$

$$> \alpha^{k-2} + \alpha^{k-3} \quad \text{(by inductive hypothesis)}$$

$$= \alpha^{k-3}(\alpha + 1)$$

Since α satisfies $\alpha^2 = \alpha + 1$ (because α is a root of $x^2 - x - 1 = 0$):

$$\alpha + 1 = \alpha^2$$

Therefore $f_{k+1} > \alpha^{k-3} \cdot \alpha^2 = \alpha^{k-1}$.

Thus $P(k+1)$ holds.

Therefore $f_n > \alpha^{n-2}$ for all $n \geq 3$.

Note: $\alpha = \frac{1+\sqrt{5}}{2} \approx 1.618$ is the golden ratio. This example shows that Fibonacci numbers grow exponentially.

Fibonacci Numbers in Nature

Fibonacci numbers appear in:

- Rabbit population growth (Fibonacci's original application)
- Flower petal counts
- Spiral patterns in shells and pinecones

5.3.3 3. Recursively Defined Sets and Structures (递归定义的集合与结构)

Definition: A **recursive definition of a set** has two parts:

Step	Description
Basis Step (基础步骤)	Specify an **initial collection** of elements in the set.
Recursive Step (递归步骤)	Give **rules** for forming new elements from those already known to be in the set.
Exclusion Rule (排除规则)	The set contains **nothing other than** those elements specified in the basis step and generated by the recu

Example 1: Subset of Integers

Define $S \subseteq \mathbb{Z}$:

- **Basis Step**: $3 \in S$.
- **Recursive Step**: If $x \in S$ and $y \in S$, then $x + y \in S$.

Initially: $3 \in S$. Then: $3 + 3 = 6 \in S$, $3 + 6 = 9 \in S$, $6 + 6 = 12 \in S$, etc. Result: $S = \{3, 6, 9, 12, \ldots\}$ = all positive multiples of 3.

Proof that $S = \{3n \mid n \in \mathbb{Z}^+\}$: Let $A = \{3n \mid n \in \mathbb{Z}^+\}$.

- $A \subseteq S$ **(by MI)**: Show $3n \in S$ for all n.
- Basis: $3 \in S$ (basis step of definition), so $3 \cdot 1 \in S$.
- Inductive: Assume $3k \in S$. Since $3 \in S$ as well, by the recursive step $3k + 3 = 3(k+1) \in S$.
- Thus $A \subseteq S$.
- $S \subseteq A$: The basis step puts 3 (which is $3 \cdot 1$) in A. The recursive step adds $x + y$ where $x, y \in S$; if x and y are both multiples of 3, then $x + y$ is also a multiple of 3. Hence every element of S is a multiple of 3, i.e., $S \subseteq A$.

Therefore $S = A$.

Example 2: Natural Numbers $\mathbb{N}$

- **Basis Step**: $0 \in \mathbb{N}$.
- **Recursive Step**: If $n \in \mathbb{N}$, then $n + 1 \in \mathbb{N}$.

5.3.4 4. Strings (字符串)

Definition: The set Σ^* of **strings** over the alphabet Σ:

- **Basis Step**: $\lambda \in \Sigma^*$ (λ is the **empty string (空串)**).
- **Recursive Step**: If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$.

Example: $\Sigma = \{0, 1\}$. Σ^* contains all bit strings: $\lambda, 0, 1, 00, 01, 10, 11, \ldots$ **Example**: Show $aab \in \Sigma^*$ where $\Sigma = \{a, b\}$.

- $\lambda \in \Sigma^*$ (basis)
- $a \in \Sigma^*$ (since $\lambda \in \Sigma^*$ and $a \in \Sigma$)
- $aa \in \Sigma^*$ (since $a \in \Sigma^*$ and $a \in \Sigma$)
- $aab \in \Sigma^*$ (since $aa \in \Sigma^*$ and $b \in \Sigma$)

String Concatenation (字符串连接)

Definition: The concatenation of two strings w_1 and w_2, denoted $w_1 \cdot w_2$ (or just $w_1 w_2$):

- **Basis Step**: $w \cdot \lambda = w$.
- **Recursive Step**: If $w_1 \in \Sigma^*$, $w_2 \in \Sigma^*$, and $x \in \Sigma$, then $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$.

Length of a String (字符串长度)

Recursive definition of $l(w)$, the length of string w:

- $l(\lambda) = 0$
- $l(wx) = l(w) + 1$ if $w \in \Sigma^*$ and $x \in \Sigma$

5.3.5 5. Balanced Parentheses (平衡括号)

Definition: The set P of **balanced parentheses**:

- **Basis Step**: $() \in P$.
- **Recursive Step**: If $w \in P$, then $()w \in P$, $(w) \in P$, and $w() \in P$.

Example: Show $(()()) \in P$.

- $() \in P$ (basis)
- $(()) \in P$ (by recursive step on $w = ()$: (w))
- $(()()) \in P$ (by recursive step on $w = ()$: $w()$, then on $(())$: (w))

Why is `))(()` **not in** P**?** Any string formed by the rules will have equal numbers of `(` and `)`, and at any prefix, the number of `(` is $\geq$ the number of `)`.

5.3.6 6. Well-Formed Formulae in Propositional Logic (命题逻辑的合式公式)

Definition: The set of **well-formed formulae (合式公式, WFF)** in propositional logic:

- **Basis Step**: T, F, and s (where s is a propositional variable) are well-formed formulae.
- **Recursive Step**: If E and F are well-formed formulae, then $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \to F)$, $(E \leftrightarrow F)$ are well-formed formulae.

Example: $((p \vee q) \to (q \wedge r))$ is well-formed. $p \wedge q \to r$ is **not** (missing parentheses).

5.3.7 7. Rooted Trees (根树) and Full Binary Trees (满二叉树)

Rooted Trees (根树)

Recursive Definition:

- **Basis Step**: A single vertex r is a rooted tree.

- **Recursive Step**: Suppose $T_1, T_2, \ldots, T_n$ are disjoint rooted trees with roots $r_1, r_2, \ldots, r_n$. Then the graph formed by starting with a new root r (not in any T_i) and adding an edge from r to each r_i is also a rooted tree.

Full Binary Trees (满二叉树)

Recursive Definition:

- **Basis Step**: A single vertex r is a full binary tree.
- **Recursive Step**: If T_1 and T_2 are disjoint full binary trees, then $T_1 \cdot T_2$ (consisting of a new root r with edges to the roots of T_1 and T_2) is a full binary tree.

Height of a Full Binary Tree

Recursive definition of **height** $h(T)$:

- **Basis Step**: $h(T) = 0$ for a tree consisting of only a root.
- **Recursive Step**: If $T = T_1 \cdot T_2$, then $h(T) = 1 + \max(h(T_1), h(T_2))$.

Number of Vertices of a Full Binary Tree

Recursive formula for **number of vertices** $n(T)$:

- **Basis Step**: $n(T) = 1$ for a tree consisting of only a root.
- **Recursive Step**: If $T = T_1 \cdot T_2$, then $n(T) = 1 + n(T_1) + n(T_2)$.

5.3.8 8. Structural Induction (结构归纳法)

Definition: **Structural Induction** is a proof technique used to prove properties of elements of a **recursively defined set**.

Steps

Step	Description
Basis Step (基础步骤)	Show that the result holds for all elements specified in the **basis step** of th
Recursive Step (递归步骤)	Show that if the statement is true for each of the elements used to construct new elements in the **recursiv**

Structural Induction 的正确性可以由 Mathematical Induction 推导出来。

Theorem: Structural Induction on Full Binary Trees

Theorem: If T is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$. **Proof** (by Structural Induction):

- **Basis Step**: For a full binary tree consisting of only a root: $n(T) = 1$, $h(T) = 0$.
  $2^{h(T)+1} - 1 = 2^1 - 1 = 1$.
  Therefore $n(T) \leq 2^{h(T)+1} - 1$ holds.
- **Recursive Step**: Assume the statement holds for two disjoint full binary trees T_1 and T_2. That is: $n(T_1) \leq 2^{h(T_1)+1} - 1$ and $n(T_2) \leq 2^{h(T_2)+1} - 1$.
  For $T = T_1 \cdot T_2$:

$$n(T) = 1 + n(T_1) + n(T_2)$$

$$\leq 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) \quad \text{(by inductive hypothesis)}$$

$$= 2^{h(T_1)+1} + 2^{h(T_2)+1} - 1$$

$$\leq 2 \cdot 2^{\max(h(T_1), h(T_2))} - 1$$

$$= 2^{\max(h(T_1), h(T_2))+1} - 1$$

Since $h(T) = 1 + \max(h(T_1), h(T_2))$, we have $\max(h(T_1), h(T_2)) = h(T) - 1$. Therefore:

$$n(T) \leq 2^{(h(T)-1)+1} - 1 = 2^{h(T)} - 1 \leq 2^{h(T)+1} - 1$$

Thus $n(T) \leq 2^{h(T)+1} - 1$ holds.

Therefore, for every full binary tree T, $n(T) \leq 2^{h(T)+1} - 1$.

5.3.9 9. Generalized Induction (广义归纳法)

Generalized Induction is used to prove results about sets other than the integers that have the **well-ordering property**.

Lexicographic Ordering on $\mathbb{N} \times \mathbb{N}$ (字典序)

Define an ordering on $\mathbb{N} \times \mathbb{N}$: $(x_1, y_1) \leq (x_2, y_2)$ if either $x_1 < x_2$, or $x_1 = x_2$ and $y_1 \leq y_2$. This is called **lexicographic ordering (字典序)**.

Example

Suppose $a_{m,n}$ is defined for $(m, n) \in \mathbb{N} \times \mathbb{N}$ by:

- $a_{0,0} = 0$
- $a_{m,n} = \begin{cases} a_{m,n-1} + 1 & \text{if } n > 0 \\ a_{m-1,n} + 1 & \text{if } n = 0 \end{cases}$

Show that $a_{m,n} = m + \frac{n(n+1)}{2}$. **Proof** (using generalized induction on lexicographic ordering):

- **Basis Step**: $a_{0,0} = 0 = 0 + \frac{0 \cdot 1}{2}$.
- **Inductive Step**: Assume the formula holds for all (m', n') less than (m, n) in lexicographic order.
- If $n > 0$: $a_{m,n} = a_{m,n-1} + 1 = [m + \frac{(n-1)n}{2}] + 1 = m + \frac{n(n+1)}{2}$.
- If $n = 0$ and $m > 0$: $a_{m,0} = a_{m-1,0} + 1 = [(m-1) + 0] + 1 = m = m + \frac{0 \cdot 1}{2}$.

Thus $a_{m,n} = m + \frac{n(n+1)}{2}$ for all $(m, n) \in \mathbb{N} \times \mathbb{N}$.

5.3.10 10. Fractals (分形)

Fractal: A pattern that uses recursion —the pattern itself repeats indefinitely in different sizes and orientations.

Koch Curve

- A curve of **order 0** is a straight line.
- A curve of **order** n consists of 4 curves of order $n - 1$.

Koch Snowflake

- Formed from 3 Koch curves of order 4.

Sierpinski Triangle

- A confined recursion of triangles forming a geometric lattice.

5.3.11 11. Euclidean Algorithm (欧几里得算法) and Lame's Theorem (拉梅定理)

Euclidean Algorithm

An efficient method for computing $\gcd(a, b)$ (greatest common divisor, 最大公约数). Based on: $\gcd(a, b) = \gcd(b, c)$ where c is the remainder when a is divided by b. **Pseudocode**:

```
procedure gcd(a, b: positive integers)
    x := a
    y := b
    while y != 0
        r := x mod y
        x := y
        y := r
    return x  {gcd(a,b) is x}
```

Example: Find $\gcd(287, 91)$:

- $287 = 91 \cdot 3 + 14$
- $91 = 14 \cdot 6 + 7$
- $14 = 7 \cdot 2 + 0$
- $\gcd(287, 91) = \gcd(91, 14) = \gcd(14, 7) = 7$

Lame's Theorem (拉梅定理)

Statement: Let a and b be positive integers with $a \geq b$. The number of divisions used by the Euclidean algorithm to find $\gcd(a, b)$ is **less than or equal to five times the number of decimal digits in** b. **Consequence**: Time complexity of Euclidean algorithm is $O(\log b)$ where $a > b$.

Lame's Theorem was the first result in **computational complexity (计算复杂度理论)**.

5.3.12 12. Summary and Exam Tips (小结与考点提示)

概念	英文	要点
递归定义	Recursive Definition	Basis Step + Recursive Step
斐波那契数列	Fibonacci Numbers	$f_n = f_{n-1} + f_{n-2}$, $f_0 = 0, f_1 = 1$
结构归纳法	Structural Induction	对递归定义集合的归纳证明
满二叉树	Full Binary Tree	每个节点有 0 或 2 个子树
欧几里得算法	Euclidean Algorithm	$\gcd(a, b) = \gcd(b, a \bmod b)$
拉梅定理	Lame's Theorem	算法复杂度 $O(\log b)$
广义归纳法	Generalized Induction	对字典序等良序集的归纳

考试重点

1. **递归定义函数的计算**：给定递归公式，能求出若干项的值（如 $f(0) = 3, f(n+1) = 2f(n) + 3$）。

2. **Fibonacci 数列与 Strong Induction 的结合**：经典证明 $f_n > \alpha^{n-2}$（需要 Strong Induction）。
3. **递归定义集合与结构归纳法**：

- 对满二叉树证明 $n(T) \leq 2^{h(T)+1} - 1$ 是标准例题。
- Structural Induction 的步骤：Basis（单节点树）+ Recursive（假设左右子树成立，证明合并后成立）。

1. **字符串的递归定义**：$\lambda \in \Sigma^*$，$w \in \Sigma^*, x \in \Sigma \implies wx \in \Sigma^*$。

- 连接 (concatenation) 和长度 (length) 的递归定义。

1. **欧几里得算法**：能用辗转相除法求 gcd，理解 Lame's Theorem 的含义。
2. **平衡括号与合式公式**：能根据递归定义判断字符串是否合法。

常见题型

- 给递归定义，求值（Fibonacci 数、阶乘等）。
- 用 Strong Induction 证明 Fibonacci 的性质。
- 用 Structural Induction 证明二叉树的性质。
- 用 Euclidean Algorithm 计算 gcd。
- 写递归定义（如字符串长度、平衡括号等）。

Notes prepared for DMA course review. Key terms are kept in English for exam readiness.

第六章　Chapter 6: Counting (计数)

6.1 Chapter 6.3: Permutations and Combinations (排列与组合)

6.1.1 1. Section Overview

本节介绍两个最基本的计数工具：**排列 (Permutation)** 和 **组合 (Combination)**。它们是后续所有计数问题的基础，也是离散概率中计算样本空间大小的核心手段。**Exam Tip:** 考试中遇到"how many ways" 类型的问题，第一步永远是判断 **order matters?** (顺序是否重要)。若顺序重要 → 排列；若顺序不重要 → 组合。

6.1.2 2. Permutations (排列)

2.1 Definition

Permutation (排列): A permutation of a set of distinct objects is an **ordered arrangement** of these objects. > **r-permutation**: An ordered arrangement of **r** elements of a set.

- 排列强调 **顺序 (order)**。同一个集合中的元素，顺序不同即为不同的排列。- 记号: $P(n, r)$ 表示从 n 个不同元素中选取 r 个进行排列的总数。

Example: 设 $S = \{1, 2, 3\}$。- 全部排列 (permutations of S): 123, 132, 213, 231, 312, 321 ($3! = 6$ 种) - 2-permutations of S: 12, 13, 21, 23, 31, 32 → $P(3, 2) = 6$

2.2 Formula

Theorem 1: If n is a positive integer and r is an integer with $1 \leq r \leq n$, then the number of **r-permutations** of a set with n distinct elements is: >

$$P(n, r) = n(n-1)(n-2)\cdots(n-r+1) = \frac{n!}{(n-r)!}$$

Proof (乘积法则): - 第 1 个位置有 n 种选择 - 第 2 个位置有 $n-1$ 种选择（已选走 1 个）- ... - 第 r 个位置有 $n-r+1$ 种选择 - 由 Product Rule (乘积法则)，相乘得 $n(n-1)\cdots(n-r+1)$

Special Case: $P(n, 0) = 1$ (只有一种方式排列 0 个元素) **Special Case:** $P(n, n) = n!$ (全排列)

2.3 Examples

Example 1: Prize Winners (奖品分配)

How many ways are there to select a first-prize winner, a second-prize winner, and a third-prize winner from **100** different people who have entered a contest?

Solution: - 顺序重要（一等奖、二等奖、三等奖是不同的位置）- 即从 100 人中选 3 人排顺序

$$P(100, 3) = 100 \cdot 99 \cdot 98 = 970{,}200$$

Example 2: Tour Route (旅行路线)

A saleswoman must visit eight different cities. She must begin her trip in a specified city, but can visit the other seven cities in any order. How many possible orders?

Solution: - 起点固定，其余 7 个城市可以任意排列顺序

$$7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5040$$

- 如果要找最短路径，她需要考虑 5040 种路径!

Example 3: Permutations with Constraint (带约束的排列)

How many permutations of the letters ABCDEFGH contain the string **ABC**?

Solution: - 把"ABC" 看作一个整体对象，则我们有 6 个对象: ABC, D, E, F, G, H

$$6! = 720$$

6.1.3 3. Combinations (组合)

3.1 Definition

r-combination (r-组合): An **unordered** selection of r elements from a set. - 组合不关心顺序，只关心选了哪些元素 - 一个 r-combination 就是集合的一个大小为 r 的 **子集 (subset)** > 记号为 $C(n,r)$ 或 $\binom{n}{r}$ (称为 **binomial coefficient / 二项式系数**)

Example: 设 $S = \{a,b,c,d\}$。- $\{a,c,d\}$ 是一个 3-combination - $\{a,c,d\} = \{d,c,a\}$ (顺序不重要) - 所有的 2-combinations: $\{a,b\},\{a,c\},\{a,d\},\{b,c\},\{b,d\},\{c,d\} \to C(4,2) = 6$

3.2 Formula

Theorem 2: The number of **r-combinations** of a set with n elements, where $0 \le r \le n$, is: >

$$C(n,r) = \binom{n}{r} = \frac{P(n,r)}{r!} = \frac{n!}{r!(n-r)!}$$

Proof (by Product Rule): - 先选 r-permutation: $P(n,r)$ 种方式 - 每个 r-combination 对应 $r!$ 种不同的排列 - 所以 $C(n,r) = \frac{P(n,r)}{r!}$

3.3 Key Property

Corollary: Let n and r be nonnegative integers with $r \le n$. Then >

$$C(n,r) = C(n,n-r)$$

Proof:

$$C(n,r) = \frac{n!}{r!(n-r)!} = \frac{n!}{(n-r)!r!} = \frac{n!}{(n-r)!(n-(n-r))!} = C(n,n-r)$$

理解: 选 r 个元素 == 选 $n-r$ 个元素留下 → 是一一对应的。

3.4 Examples

Example 1: Poker Hands (扑克手牌)

How many poker hands of five cards can be dealt from a standard deck of 52 cards?

Solution: - 手牌的顺序不重要 → 组合

$$C(52,5) = \frac{52!}{5!47!}$$

How many ways to select 47 cards from 52?

$$C(52,47) = C(52,5) \quad \text{(由对称性)}$$

Example 2: Team Selection (组队选人)

How many ways to select five players from a 10-member tennis team?

$$C(10,5) = \frac{10!}{5!5!} = 252$$

Example 3: Mars Mission (火星任务)

A group of 30 people, how many ways to select a crew of six?

$$C(30,6)$$

Example 4 (Combined): Soccer Club

A soccer club has 8 female and 7 male members. (1) Select 6 female and 5 male players (2) Select 11 players with **at most** 5 male players

Solution (1): 乘法原理: 先选女生，再选男生

$$C(8,6) \cdot C(7,5) = 28 \cdot 21 = 588$$

Solution (2): "at most 5 male players" 意味着男球员数 ≤ 5，分情况: - 6 女 5 男: $C(8,6)C(7,5)$ - 7 女 4 男: $C(8,7)C(7,4)$ - 8 女 3 男: $C(8,8)C(7,3)$

$$C(8,6)C(7,5) + C(8,7)C(7,4) + C(8,8)C(7,3)$$

6.1.4 4. Combinatorial Proofs (组合证明)

4.1 Definition

Combinatorial proof (组合证明): A proof that uses counting arguments to show an identity holds.

两种主要方法: 1. **Double counting proof** (双重计数): 用两种不同方式计数同一组对象，证明等式两边相等 2. **Bijective proof** (双射证明): 证明两个集合之间存在一一对应 (bijection)，因此元素个数相同

4.2 Example: Proving $C(n,r) = C(n,n-r)$

Bijective Proof:

- 设 S 有 n 个元素
- 映射 $f(A) = \overline{A}$ (取补集) 是从 S 的 r-子集到 (n-r)-子集的双射
- 由于双射的两个集合元素个数相等，所以 $C(n,r) = C(n,n-r)$

Double Counting Proof:

- S 的 r-子集的数量 $= C(n,r)$
- 每个 r-子集 A 也可以由其补集 $\overline{A}$ 描述 (含 $n-r$ 个元素)
- 所以 r-子集的数量也等于 $C(n,n-r)$
- 因此 $C(n,r) = C(n,n-r)$

6.1.5 5. Summary Table (总结表)

Concept	Order Matters?	Formula
Permutation (排列)	**Yes** (顺序重要)	$P(n,r) = \frac{n!}{(n-r)!}$
Combination (组合)	**No** (顺序不重要)	$C(n,r) = \binom{n}{r} = \frac{n!}{r!(n-r)!}$

6.1.6 6. 考点提示 (Exam Tips)

1. **Order test (顺序测试):** 做题第一步判断是否有序。Prize (一等奖/二等奖) → 有序；Team member (队员) → 无序
2. **"At most / at least" 问题:** 分情况讨论，逐一相加
3. **Combinatorial proof:** 理解双射思想和双重计数法，不要求代数推导
4. **P(0,0) = 1, C(n,0) = 1:** 边界情况不要忘
5. **对称性:** $C(n,r) = C(n,n-r)$ 可以简化计算

6.1.7 Homework (from courseware)

Sec 6.3: 20, 44, 46

6.2 Chapter 6.4: Binomial Theorem & Pascal's Identity (二项式定理与帕斯卡恒等式)

6.2.1 1. Section Overview

本节围绕 **二项式系数 (binomial coefficients)** $\binom{n}{k}$ 展开，介绍：

- **Binomial Theorem** (二项式定理) —展开 $(x+y)^n$ 的公式
- **Pascal's Identity** (帕斯卡恒等式) —$\binom{n}{k}$ 的递推关系
- **Pascal's Triangle** (杨辉三角)
- **Vandermonde's Identity** 及其他常用恒等式

Exam Tip: 二项式定理是必考内容！需要熟练展开并找到特定项的系数。

6.2.2 2. Binomial Expression (二项式)

Binomial expression (二项式): The sum of two terms, such as $x+y$.

展开 $(x+y)^n$ 时，每个项都是从每个 $(x+y)$ 因子中选 x 或 y 相乘的结果。- $(x+y)^n = (x+y)(x+y)\cdots(x+y)$ (n 个因子) - 展开后形如 $x^{n-j}y^j$，系数 = 从 n 个因子中选 j 个取 y 的方式数 = $\binom{n}{j}$

Intuition: $(x+y)^3$ **展开**

$$(x+y)^3 = (x+y)(x+y)(x+y)$$

- x^3: 全部选 x, 1 种方式 → 系数 1
- x^2y: 选 2 个 x 和 1 个 y → $\binom{3}{1} = 3$ 种方式 → 系数 3
- xy^2: 选 1 个 x 和 2 个 y → $\binom{3}{2} = 3$ 种方式 → 系数 3
- y^3: 全部选 y, 1 种方式 → 系数 1

$$(x+y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$$

6.2.3 3. Binomial Theorem (二项式定理)

Binomial Theorem: Let x and y be variables, and n a nonnegative integer. Then: >

$$(x+y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j} y^j = \binom{n}{0} x^n + \binom{n}{1} x^{n-1} y + \binom{n}{2} x^{n-2} y^2 + \cdots + \binom{n}{n} y^n$$

Proof (组合论证): - 展开 $(x+y)^n = (x+y)(x+y)\cdots(x+y)$ (n 个因子) - 项 $x^{n-j}y^j$ 需要从 j 个因子中选 y，其余选 x - 选法数 = $\binom{n}{j}$ → 这就是系数

Example 1: Find Coefficient

What is the coefficient of $x^{12}y^{13}$ in the expansion of $(x-3y)^{25}$?

Solution: - 把表达式看作 $(x+(-3y))^{25}$ - 由 Binomial Theorem:

$$(x+(-3y))^{25} = \sum_{j=0}^{25} \binom{25}{j} x^{25-j}(-3y)^j = \sum_{j=0}^{25} \binom{25}{j} (-3)^j x^{25-j} y^j$$

- 要找 $x^{12}y^{13}$ → $j = 13$ - 系数 = $\binom{25}{13}(-3)^{13}$

6.2.4 4. Useful Corollaries

Corollary 1: Sum of Binomial Coefficients

With $n \geq 0$:

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Proof (Binomial Theorem): 令 $x = 1, y = 1$:

$$(1+1)^n = \sum_{k=0}^{n} \binom{n}{k} 1^{n-k} 1^k = \sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Proof (Combinatorial): - $\binom{n}{k}$ = 从 n 个元素中选 k 个的方式数 - 所有子集总数 = $\sum_{k=0}^{n} \binom{n}{k}$ - 已知 n 个元素的集合有 2^n 个子集 - 因此 $\sum_{k=0}^{n} \binom{n}{k} = 2^n$

理解: 所有二项式系数之和 = 2^n，即 n 元素集合的子集总数。

Corollary 2: Alternating Sum

Let n be a positive integer. Then:

$$\sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0$$

Proof: 令 $x = 1, y = -1$:

$$(1+(-1))^n = \sum_{k=0}^{n} \binom{n}{k} 1^{n-k} (-1)^k = \sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0^n = 0$$

理解: 正负交替的二项式系数之和 = 0。意味着 $\sum_{k \text{ even}} \binom{n}{k} = \sum_{k \text{ odd}} \binom{n}{k}$ (偶数项之和 = 奇数项之和)。

6.2.5 5. Pascal's Identity (帕斯卡恒等式)

Pascal's Identity: If n and k are integers with $n \geq k \geq 0$, then: >

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$$

Proof (组合论证): - 设 T 是一个集合，$|T| = n+1$，取 $a \in T$，$S = T \setminus \{a\}$ - $\binom{n+1}{k}$ = T 的 k-子集总数 - 分类讨论: - **包含 a 的子集:** 从 S 中再选 k-1 个 → $\binom{n}{k-1}$ 种 - **不包含 a 的子集:** 从 S 中选 k 个 → $\binom{n}{k}$ 种 - 因此 $\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$

6.2.6 6. Pascal's Triangle (杨辉三角 / 帕斯卡三角)

```
            1                    (n=0)
          1   1                  (n=1)
        1   2   1                (n=2)
      1   3   3   1              (n=3)
    1   4   6   4   1            (n=4)
  1   5  10  10   5   1          (n=5)
1   6  15  20  15   6   1        (n=6)
```

- 第 n 行: $\binom{n}{0}, \binom{n}{1}, \ldots, \binom{n}{n}$
- 每个数等于它上方两数之和 (Pascal's Identity)
- 第 n 行之和 = 2^n

6.2.7 7. Vandermonde's Identity (范德蒙恒等式)

Vandermonde's Identity: Let m, n, r be nonnegative integers with r not exceeding either m or n. Then: >

$$\binom{m+n}{r} = \sum_{k=0}^{r} \binom{m}{r-k} \binom{n}{k}$$

Proof (组合论证): - 设 A 和 B 是两个不相交的集合，$|A| = m, |B| = n$ - $\binom{m+n}{r}$ = 从 $A \cup B$ 中选 r 个元素的方式数 - 另一种方式: 从 A 中选 $r-k$ 个，从 B 中选 k 个，$k = 0, 1, \ldots, r$

$$\sum_{k=0}^{r} \binom{m}{r-k} \binom{n}{k}$$

Corollary 4 (Special Case of Vandermonde's)

If n is a nonnegative integer, then:

$$\binom{2n}{n} = \sum_{k=0}^{n} \binom{n}{k}^2$$

Proof: 在 Vandermonde's Identity 中令 $m = n, r = n$:

$$\binom{2n}{n} = \sum_{k=0}^{n} \binom{n}{n-k} \binom{n}{k} = \sum_{k=0}^{n} \binom{n}{k}^2$$

6.2.8 8. Theorem 4: Another Identity

Let n and r be nonnegative integers with $r \leq n$. Then:

$$\binom{n+1}{r+1} = \sum_{j=r}^{n} \binom{j}{r}$$

Proof (组合论证): - LHS: 从 $n+1$ 个元素的集合中选 $r+1$ 个元素的子集数 - RHS: 考虑长度为 $n+1$ 的 bit string 中最后一个 1 的位置 - 若最后一个 1 在位置 $k+1$，则前 k 位中有 r 个 1 → $\binom{k}{r}$ - k 从 r 到 n 求和，即 $\sum_{j=r}^{n} \binom{j}{r}$

6.2.9 9. 考点提示 (Exam Tips)

Identity	Formula	When to Use
Binomial Theorem	$(x+y)^n = \sum \binom{n}{j} x^{n-j} y^j$	展开二项式，找特定项系数
Sum of binom coeffs	$\sum \binom{n}{k} = 2^n$	证明子集个数
Alternating sum	$\sum (-1)^k \binom{n}{k} = 0$	奇偶子集个数相等
Pascal's Identity	$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$	递推计算
Vandermonde's	$\binom{m+n}{r} = \sum \binom{m}{r-k}\binom{n}{k}$	从两个集合中选元素

1. **Binomial Theorem 是核心:** 务必熟练掌握，尤其是找指定项的系数
2. **注意符号:** 如 $(x-3y)^{25}$，y 的系数含 $(-3)^j$
3. **Pascal's Triangle** 各行和 $= 2^n$
4. **组合证明**是本课程重点—用" 数数" 方式证明恒等式，不需要代数推导
5. **Corollary 4:** $\binom{2n}{n} = \sum \binom{n}{k}^2$ 经常出现在组合恒等式中

6.2.10 Homework (from courseware)

Sec 6.4: 18, 26, 30, 34

6.3 Chapter 6.5: Generalized Permutations and Combinations (广义排列与组合)

6.3.1 1. Section Overview

本节将 6.3 中的排列组合推广到更一般的情形:

- **Permutations with Repetition** (重复排列)
- **Combinations with Repetition** (重复组合) —**Stars and Bars** 方法
- **Permutations with Indistinguishable Objects** (含相同对象的排列)
- **Distributing Objects into Boxes** (分配问题)

Exam Tip: 本节是考试的重点和难点！特别是 **Stars and Bars** 方法解决" 方程非负整数解" 问题几乎必考。

6.3.2 2. Permutations with Repetition (重复排列)

Theorem 1: The number of **r-permutations** of a set of n objects with **repetition allowed** is: >

$$n^r$$

Proof: - 每个位置有 n 种选择 - 共 r 个位置 - 由 Product Rule: $n \times n \times \cdots \times n = n^r$

Example: How many strings of length r can be formed from the uppercase English alphabet (26 letters)?

Solution: 26^r (每个位置有 26 种选择)

对比 6.3: 不允许重复时是 $P(n,r) = \frac{n!}{(n-r)!}$；允许重复时是 n^r。

6.3.3 3. Combinations with Repetition (重复组合)

3.1 Motivating Example

How many ways to select **five bills** from a box containing at least five of each of the following denominations: 1,2, 5,10, 20,50, $100?

Analysis: - 有 7 种面额 (n = 7)，选 5 张 (r = 5) - 同种面额可以重复选 → **组合 + 重复** - 顺序不重要 → 组合

Key Insight (Stars and Bars): - 把问题转化为: 把 5 个 **star** ($*$) 分配到 7 种面额中，用 6 个 **bar** (|) 分隔 - 例如: 2,2, 10,20, $100 → $** | * | | * | *$ | (5 stars, 6 bars) - 问题等价于: 排列 5 个 star 和 6 个 bar → $\binom{5+6}{5} = \binom{11}{5}$

3.2 Theorem

Theorem 2: The number of **r-combinations** from a set with n elements when **repetition is allowed** is: >

$$C(n+r-1,\ r) = C(n+r-1,\ n-1)$$

Proof (Stars and Bars): - 每个 r-combination 对应一个长度为 $n+r-1$ 的序列，包含 r 个 star 和 $n-1$ 个 bar - 选 r 个位置放 star (其余放 bar) - 方法数 $= \binom{n+r-1}{r}$

直观理解: " 把 r 个无区别的球放入 n 个有区别的盒子" 的方式数。

3.3 Examples

Example 1: Cookie Shop (饼干店)

A cookie shop has **four** different kinds of cookies. How many different ways can **six** cookies be chosen?

Solution: - n = 4 (四种饼干), r = 6 (选 6 块) - 重复组合:

$$C(4+6-1,6) = C(9,6) = C(9,3) = 84$$

Example 2: Equation Solutions (方程非负整数解)

How many solutions does the equation $x_1 + x_2 + x_3 = 11$ have, where x_i are **nonnegative** integers?

Solution: - n = 3 (三个变量), r = 11 (和为 11) - 每个解对应: 从 3 种类型中选 11 个 (每种类型可以重复选) - 方法数:

$$C(3+11-1,11) = C(13,11) = C(13,2) = 78$$

Example 3: More Variables

$x_1 + x_2 + x_3 + x_4 = 16$, $x_i \geq 0$ integer.

Solution: - n = 4, r = 16

$$C(4+16-1,16) = C(19,16) = C(19,3) = 969$$

Example 4: With Constraints (带约束的方程)

$x_1 + x_2 + x_3 + x_4 = 16$, $x_i \geq 2$ for all i.

Solution: - 令 $y_i = x_i - 2 \geq 0$, 则 $y_1 + y_2 + y_3 + y_4 = 8$

$$C(4+8-1,8) = C(11,8) = C(11,3) = 165$$

$x_1 + x_2 + x_3 + x_4 \leq 16$, $x_i \geq 0$.

Solution: - 引入辅助变量 $x_5 \geq 0$, 使 $x_1 + x_2 + x_3 + x_4 + x_5 = 16$

$$C(5+16-1,16) = C(20,16) = C(20,4) = 4845$$

6.3.4 4. Summary: Four Types of Counting (四种计数总结)

Type	Repetition Allowed?	Order Matters?	Formula
Permutation	No	Yes	$\frac{n!}{(n-r)!}$
Permutation	**Yes**	Yes	n^r
Combination	No	No	$\binom{n}{r}$
Combination	**Yes**	No	$\binom{n+r-1}{r}$

6.3.5 5. Permutations with Indistinguishable Objects (含相同对象的排列)

5.1 Example: SUCCESS

How many different strings can be made by reordering the letters of the word **SUCCESS**?

Breakdown: S × 3, C × 2, U × 1, E × 1 (共 7 个字母)

Solution: 1. 选 3 个位置放 S: $C(7,3)$ 2. 从剩余 4 个位置选 2 个放 C: $C(4,2)$ 3. 从剩余 2 个位置选 1 个放 U: $C(2,1)$ 4. 最后 1 个放 E: $C(1,1)$

乘积法则:

$$C(7,3) \cdot C(4,2) \cdot C(2,1) \cdot C(1,1) = \frac{7!}{3!2!1!1!}$$

5.2 Theorem

Theorem 3: The number of different permutations of n objects, where there are n_1 indistinguishable objects of type 1, n_2 of type 2, ..., n_k of type k, is: >

$$\frac{n!}{n_1!\,n_2!\cdots n_k!}$$

Proof: 先放 type 1 ($C(n,n_1)$), 再放 type 2 ($C(n-n_1,n_2)$), ..., 乘积化简即得。

5.3 More Examples

Example: MISSISSIPPI

How many different strings can be made from the letters in MISSISSIPPI, using all the letters?

Breakdown: M × 1, I × 4, S × 4, P × 2 (共 11 个字母)

Solution:

$$\frac{11!}{1!\,4!\,4!\,2!}$$

Example: Class Leading Group

A class of 50 students: (1) Select 7 students for a leading group → $C(50,7)$ (2) Select a monitor and a vice monitor from the 7 → $P(7,2)$ (3) Assign 7 different tasks to 7 students → $P(50,7)$

6.3.6 6. Distributing Objects into Boxes (分配问题)

这是计数问题的终极概括。根据 **对象是否可区分**和 **盒子是否可区分**，有四种情况:

6.1 Distinguishable Objects + Distinguishable Boxes

The number of ways to distribute n **distinguishable** objects into k **distinguishable** boxes is: >

$$\frac{n!}{n_1!\,n_2!\cdots n_k!}$$

> where box i receives n_i objects ($\sum n_i = n$).

实际上就是含相同对象排列的公式—把" 对象" 分配到" 位置/盒子" 是等价的。

Example: Deal 5 cards each to 4 players from 52 cards:

$$\frac{52!}{5!\,5!\,5!\,5!\,32!}$$

(最后 32 张牌未发)

6.2 Indistinguishable Objects + Distinguishable Boxes

The number of ways to place r **indistinguishable** objects into n **distinguishable** boxes is: >

$$C(n+r-1,\ r)$$

这其实就是重复组合公式！跟 Stars and Bars 等价。

Example: Place 10 indistinguishable objects into 8 distinguishable boxes:

$$C(8+10-1, 10) = C(17, 10) = 19{,}448$$

6.3 Distinguishable Objects + Indistinguishable Boxes

没有简单的闭式公式。涉及第二类 Stirling 数 (Stirling numbers of the second kind)。> 考试中通常通过 **枚举**方式解决此类问题。

Example: 4 Employees into 3 Indistinguishable Offices

How many ways to put 4 different employees into 3 indistinguishable offices?

Solution (按分组方式枚举):

(1) **4 人一屋:** A,B,C,D → 1 种

(2) **3+1 分组:** - A,B,C,D; A,B,D,C; A,C,D,B; B,C,D,A → 4 种

(3) **2+2 分组:** - A,B,C,D; A,C,B,D; A,D,B,C → 3 种

(4) **2+1+1 分组:** - A,B,C,D; A,C,B,D; A,D,B,C B,C,A,D; B,D,A,C; C,D,A,B → 6 种

Total: $1+4+3+6=14$ 种

6.4 Indistinguishable Objects + Indistinguishable Boxes

同样没有简单闭式公式。> 数量 $= p_k(n)$: 将 n 拆分成不超过 k 个正整数之和的方式数 (不计顺序)。> 通过 **枚举**解决。

Example: 6 Books into 4 Identical Boxes

How many ways to pack 6 copies of the same book into 4 identical boxes?

Solution (枚举所有拆分数):

| 分配方式 | 表示 | |———|——| | 6 | 6 | | 5+1 | 5,1 | | 4+2 | 4,2 | | 4+1+1 | 4,1,1 | | 3+3 | 3,3 | | 3+2+1 | 3,2,1 | | 3+1+1+1 | 3,1,1,1 | | 2+2+2 | 2,2,2 | | 2+2+1+1 | 2,2,1,1 |

共 **9** 种方式。

6.3.7 7. 考点提示 (Exam Tips)

1. **Stars and Bars 是必考内容:** 重复组合公式 $C(n+r-1, r)$ 务记熟
2. **方程非负整数解:** $x_1 + x_2 + \cdots + x_n = r \ \Rightarrow\ C(n+r-1, r)$
3. **带约束条件:** $x_i \geq c \to$ 令 $y_i = x_i - c \geq 0$ 转化
4. **不等式:** $x_1 + \cdots + x_n \leq r \to$ 加辅助变量转化为等式
5. **含相同对象的排列:** $\frac{n!}{n_1!n_2!\cdots n_k!}$ —分母除以下标阶乘
6. **分配问题判断流程:**

- Objects distinguishable? → Boxes distinguishable? → 用 $\frac{n!}{n_1!\cdots n_k!}$
- Objects indistinguishable? → Boxes distinguishable? → 用 $C(n+r-1, r)$
- Boxes indistinguishable? → 枚举

6.3.8 Homework (from courseware)

Sec 6.5: 10, 16, 28, 34, 48, 52, 56, 63

6.4 Chapter 6.6: Generating Permutations and Combinations (生成排列与组合)

6.4.1 1. Section Overview

前面几节我们学习了**计数**排列和组合的数量，本节学习**生成** (generate/list) 所有排列和组合的算法。**Exam Tip:** 本节内容偏算法实现，考试重点在于理解 **lexicographic order (字典序)** 和生成算法的步骤，而不是直接套公式计算。

6.4.2 2. Generating Permutations (生成排列)

2.1 Problem

List (generate) all permutations of any set with n elements.

Approach: 1. 将任意 n 元素集合与 $\{1, 2, \ldots, n\}$ 建立一一对应 2. 生成 $\{1, 2, \ldots, n\}$ 的所有排列 3. 将整数映射回原集合的元素

2.2 Lexicographic Order (字典序)

Lexicographic ordering (字典序): The permutation $a_1a_2\ldots a_n$ **precedes** $b_1b_2\ldots b_n$ if for some k ($1 \leq k \leq n$): - $a_1 = b_1, a_2 = b_2, \ldots, a_{k-1} = b_{k-1}$ - $a_k < b_k$

通俗理解: 和字典排序一样—从左到右逐位比较，遇到第一位不同的数字，较小的那个排列在前。

Example:

```
123465  precedes  124635
```

比较过程: 前两位 12 相同，第三位 $3 < 4$，所以 123465 排在 124635 之前。

2.3 Algorithm: Finding the Next Permutation

Goal: Given a permutation $a_1a_2\ldots a_n$, find the **next larger permutation** in lexicographic order. **Algorithm:**

1. **Find the largest** j such that $a_j < a_{j+1}$ (从右向左找第一个递增相邻对)
- 此时 $a_{j+1} > a_{j+2} > \cdots > a_n$ (后缀是递减的)
1. **Find the smallest integer** among $a_{j+1}, a_{j+2}, \ldots, a_n$ that is **greater than** a_j
- 即后缀中比 a_j 大的最小元素
1. **Swap** a_j with that element
2. **Reverse** (or sort in increasing order) the suffix $a_{j+1}, a_{j+2}, \ldots, a_n$

直观理解: 找到最后一个可以" 增大的位置"，将该位置的数换成后缀中稍大的数，然后将后缀反转为最小顺序。

2.4 Example

What is the next permutation after **124653** in lexicographic order?

Step-by-step:

1. 从右向左找 $a_j < a_{j+1}$: - 65: $6 > 5$ - 46: $4 < 6$ → $j = 3$, $a_j = 4$ - 后缀 = 653 (递减)
2. 在后缀 653 中找比 4 大的最小数: **5**
3. Swap 4 和 5 → **125643**
4. 反置后缀 643 → **346** - (实际是排序后缀为递增: 346)

Result: 125346

Verification: 124653 → 125346 → 125364 → 125436 → ...

6.4.3 3. Generating Combinations (生成组合)

3.1 Problem 1: Generate All Subsets

Generate all combinations (subsets) of a finite set.

Approach: - 用长度为 n 的 **bit string** 表示子集: 第 i 位 = 1 表示包含第 i 个元素 - 所有 2^n 个 bit strings 按 **二进制递增**顺序列出

Algorithm: 1. Start with $000\ldots00$ (n zeros) 2. Repeatedly find the next binary expansion until $111\ldots11$

How to find the next bit string: Locate the **first position from the right** that is **not a 1**, change it to **1**, and change **all 1s to its right** to **0**.

Example:

```
1000110011  →  1000110100
```

- 从右向左找第一个 0: 位置 (从右数第 3 位) - 该位 0→1 - 右侧所有 1→0

3.2 Problem 2: Generate All r-Combinations

Generate all r-combinations of the set $\{1, 2, \ldots, n\}$.

Key Idea: 将每个 r-combination 表示为递增序列 $a_1 < a_2 < \cdots < a_r$。

Algorithm: 1. Start with the smallest r-combination: $\{1, 2, \ldots, r\}$ 2. Given current combination $\{a_1, a_2, \ldots, a_r\}$, find the next: - Find the **last** element a_i such that $a_i \neq n-r+i$ - (即可以递增而不超出范围的最右元素) - Replace a_i with a_i+1 - For $j = i+1, \ldots, r$: set $a_j = a_i + (j-i)$ - (将后面的元素设为最小的连续递增序列)

Example

Given $S_i = \{2, 3, 5, 6, 9, 10\}$ (a 6-combination from 1..10), find S_{i+1}.

Solution: - 检查每个位置的最大可能值 (n - r + i): - $a_1 = 2$, max $= 10 - 6 + 1 = 5 \to$ OK - $a_2 = 3$, max $= 10 - 6 + 2 = 6 \to$ OK - $a_3 = 5$, max $= 10 - 6 + 3 = 7 \to$ OK - $a_4 = 6$, max $= 10 - 6 + 4 = 8 \to$ OK - $a_5 = 9$, max $= 10 - 6 + 5 = 9 \to a_5 = 9 = 9$ (已达上限) - $a_6 = 10$, max $= 10 - 6 + 6 = 10 \to a_6 = 10 = 10$ (已达上限)

- 从右向左找最后一个 $a_i \neq n - r + i$: $a_4 = 6 \neq 8 \to i = 4$

- $a_4 \leftarrow 6 + 1 = 7$ - 后面接最小递增: $a_5 = 8, a_6 = 9$

Result: $S_{i+1} = \{2, 3, 5, 7, 8, 9\}$

6.4.4 4. 考点提示 (Exam Tips)

1. **Lexicographic Order 定义:** 按字典序比较排列—从左到右逐位比较
2. **Next Permutation 算法:**
- 从右向左找 $a_j < a_{j+1}$
- 在后缀中找比 a_j 大的最小数
- 交换 + 反置后缀
1. **Next Bit String 算法:** 从右找第一个 0 → 改 1，右侧清 0
2. **Next r-Combination 算法:**
- 从右向左找最后一个可以增大的位置 ($a_i \neq n - r + i$)
- 该位 +1，后面置为最小递增序列
1. 这些算法在考试中可能以 **手工演算**的形式出现，理解步骤比背代码更重要

6.4.5 Homework (from courseware)

Sec 6.6: 6(f), 7, 9

第七章 Chapter 8: Recurrence Relations (递推关系)

7.1 Chapter 8.1: Applications of Recurrence Relations

7.1.1 1. 基本概念 (Basic Concepts)

Recurrence Relation (递推关系)

Definition: A **recurrence relation** for the sequence a_n is an equation that expresses a_n in terms of one or more of the previous terms $a_0, a_1, \ldots, a_{n-1}$, for all integers $n \geq n_0$, where n_0 is a nonnegative integer.

- **Solution:** A sequence is called a **solution** of a recurrence relation if its terms satisfy the recurrence relation.
- **Initial conditions (初始条件):** specify the terms that precede the first term where the recurrence relation takes effect.

Example (判断一个序列是否为解): Determine whether the sequence a_n is a solution of $a_n = 2a_{n-1} - a_{n-2}$ for $n = 2, 3, 4, \ldots$:

Sequence	Result
$a_n = 3n$	Yes
$a_n = 2^n$	No
$a_n = 5$	Yes

关键概念 (Key Concepts):

- 同一个递推关系可以有**无穷多个不同的序列**满足它
- **初始条件** uniquely determine a sequence
- 递推关系的 **degree (阶数)** 是指 a_n 表达式中涉及的最早项与 n 的差值。例如 $a_n = a_{n-1} + a_{n-8}$ 是 degree 8 的递推关系。

7.1.2 2. Fibonacci Numbers (斐波那契数列)

兔子问题 (Rabbits Problem)

原始问题 (Leonardo Pisano, 13 世纪): 一对幼兔被放在岛上。兔子到**两个月大**才开始繁殖。之后每个月每对成年兔子产生一对新兔子。兔子从不死亡。**建模 (Modeling):** Let f_n = number of pairs of rabbits after n months.

- $f_1 = 1$ (第一个月只有最初的一对)
- $f_2 = 1$ (第二个月仍只有最初的一对，尚未繁殖)

递推关系:

$$f_n = f_{n-1} + f_{n-2}, \quad n \geq 3$$

解释:

- f_{n-1}: 上个月已有的兔子对数
- f_{n-2}: 新出生的兔子对数（因为只有至少两个月大的兔子才能繁殖）

初值:

$$f_1 = 1, \quad f_2 = 1$$

Explicit formula (显式公式, Binet's formula):

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

7.1.3 3. Tower of Hanoi (汉诺塔)

Problem (问题描述)

有三根柱子，n 个大小不同的圆盘按从大到小顺序叠在第一根柱子上。**规则 (Rules):**

1. 每次只能移动一个圆盘
2. 大盘不能放在小盘上面

目标: 将所有圆盘移动到第二根柱子。

Recurrence Relation (递推关系)

Let H_n = number of moves needed to solve the puzzle with n disks.

$$H_n = 2H_{n-1} + 1, \quad H_1 = 1$$

思路:

1. 将上面 $n-1$ 个盘子从 peg 1 移到 peg 3: H_{n-1} 步
2. 将最大的盘子从 peg 1 移到 peg 2: 1 步
3. 将 $n-1$ 个盘子从 peg 3 移到 peg 2: H_{n-1} 步

Iterative Solution (迭代求解)

$$\begin{aligned}
H_n &= 2H_{n-1}+1 \\
&= 2(2H_{n-2}+1)+1 = 4H_{n-2}+2+1 \\
&= 8H_{n-3}+4+2+1 \\
&\vdots \\
&= 2^{n-1}H_1 + 2^{n-2}+2^{n-3}+\cdots+2+1 \\
&= 2^{n-1}\cdot 1 + (2^{n-1}-1) \quad \text{(geometric series)} \\
&= 2^n - 1
\end{aligned}$$

封闭形式 (Closed form): $H_n = 2^n - 1$

神话与现实 (Myth and Reality)

- 传说中僧人每天移动一个圆盘，需要移动 64 个金盘
- $H_{64} = 2^{64}-1 = 18,446,744,073,709,551,615$ 天
- 超过 5000 亿年

Reve's Puzzle: 4 根柱子的汉诺塔变体，最小移动次数仍未完全解决（有未证猜想）。

7.1.4 4. Counting Bit Strings (计数二进制串)

Example: Find a recurrence relation for the number of bit strings of length n with **no two consecutive 0s**. Let a_n = number of bit strings of length n with no two consecutive 0s. **初值:**

- $a_1 = 2$ (strings: `0`, `1`)
- $a_2 = 3$ (strings: `01`, `10`, `11`; `00` is invalid)

递推关系:

$$a_n = a_{n-1} + a_{n-2}, \quad n \geq 3$$

推导思路:

- 以 `1` 结尾的合法串: 前面 $n-1$ 位必须是合法串 $\to a_{n-1}$ 个
- 以 `0` 结尾的合法串: 倒数第二位必须是 `1`，所以末尾是 `10` $\to$ 前面 $n-2$ 位必须是合法串 $\to a_{n-2}$ 个

计算结果:

- $a_3 = a_2 + a_1 = 3+2 = 5$
- $a_4 = a_3 + a_2 = 5+3 = 8$
- $a_5 = a_4 + a_3 = 8+5 = 13$

注意: 这个序列与 Fibonacci 数相同！由于 $a_1 = 3 = f_4$ 且 $a_2 = 4 = f_5$，所以 $a_n = f_{n+2}$。

7.1.5 5. Catalan Numbers (卡特兰数) —括号化问题

Example: Find a recurrence relation for C_n, the number of ways to parenthesize the product of $n+1$ numbers $x_0 \cdot x_1 \cdot x_2 \cdots x_n$ to specify the order of multiplication. **例子（4 个数的括号化）:** $C_3 = 5$:

$$((x_0x_1)x_2)x_3, \quad (x_0(x_1x_2))x_3, \quad (x_0x_1)(x_2x_3), \quad x_0((x_1x_2)x_3), \quad x_0(x_1(x_2x_3))$$

递推关系:

$$C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k}, \quad C_0 = 1, C_1 = 1$$

推导: 考虑最后一次乘法运算的位置。它将乘积分成两部分:

- 左半部分有 $k+1$ 个数 $\to C_k$ 种括号化方式
- 右半部分有 $n-k$ 个数 $\to C_{n-1-k}$ 种括号化方式

所以 $C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k}$ **Catalan Numbers 的封闭形式:**

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

7.1.6 6. Codeword Enumeration (码字计数)

Example: A computer system considers a string of decimal digits a valid codeword if it contains an **even number of 0 digits**. Let a_n be the number of valid n-digit codewords. Find a recurrence relation for a_n. **递推关系:**

$$a_n = 9a_{n-1} + (10^{n-1} - a_{n-1})$$

简化:

$$a_n = 8a_{n-1} + 10^{n-1}$$

推导思路:

- 若前 $n-1$ 位已经有偶数个 0: 第 n 位可以选 1-9（非 0 数字）$\to 9a_{n-1}$ 种
- 若前 $n-1$ 位有奇数个 0: 第 n 位必须选 0 $\to 10^{n-1} - a_{n-1}$ 种

7.1.7 7. 小结和考点提示 (Summary & Exam Tips)

核心公式 (Core Formulas)

Problem	Recurrence	Initial Conditions	Closed Form
Fibonacci	$f_n = f_{n-1} + f_{n-2}$	$f_1 = 1, f_2 = 1$	$f_n = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n)$
Tower of Hanoi	$H_n = 2H_{n-1} + 1$	$H_1 = 1$	$H_n = 2^n - 1$
Bit Strings	$a_n = a_{n-1} + a_{n-2}$	$a_1 = 2, a_2 = 3$	$a_n = f_{n+2}$
Catalan	$C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k}$	$C_0 = 1$	$C_n = \frac{1}{n+1}\binom{2n}{n}$

考点提示 (Exam Tips)

1. **必背递推关系:**

- Fibonacci: $f_n = f_{n-1} + f_{n-2}$
- Tower of Hanoi: $H_n = 2H_{n-1} + 1$
- 结果 $2^n - 1$ 及其推导过程（iterative method）

1. **建立递推关系的思路:**

- 将问题分解为**互斥且完备**的几种情况
- 使用 **case analysis**: "ending with 0" vs "ending with 1"
- 考虑"first step" (如汉诺塔的最大盘片移动)

1. **英文术语必须掌握:**

- Recurrence relation
- Initial condition
- Degree of recurrence
- Fibonacci numbers
- Tower of Hanoi
- Catalan numbers
- Bit strings
- Codeword

1. **考试常见题型:**

- 给定实际问题，建立递推关系
- 给定递推关系，验证某个序列是否是解
- 用迭代法求解简单递推关系
- 计算 Fibonacci 数的值或 Catalan 数的值

7.2 Chapter 8.2: Solving Linear Recurrence Relations

7.2.1 1. 基本概念 (Basic Concepts)

Linear Homogeneous Recurrence Relation (线性齐次递推关系)

Definition: A **linear homogeneous recurrence relation of degree k with constant coefficients** is a recurrence relation of the form:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

where $c_1, c_2, \ldots, c_k$ are real numbers, and $c_k \neq 0$. **三大特性 (Three Characteristics):**

- **Linear (线性):** 右边是序列前项的线性组合（每项乘以一个只依赖于 n 的系数后求和）
- **Homogeneous (齐次):** 没有不包含 a_j 的独立项
- **Constant coefficients (常系数):** 每个系数都是常数（不依赖于 n）

Degree (阶数): k，因为 a_n 用前 k 项表示。**由 Strong Induction 可证:** 递推关系 + k 个初始条件唯一确定一个序列。

判断示例 (Classification Examples)

Recurrence	Linear?	Constant Coeff?	Homogeneous?	Degree
$a_n = (1.02)a_{n-1}$	Yes	Yes	Yes	1
$a_n = (1.02)a_{n-1} + 2^{n-1}$	Yes	Yes	**No**	1
$a_n = a_{n-1} + a_{n-2} + a_{n-3} + 2^{n-3}$	Yes	Yes	**No**	3
$a_n = na_{n-1} + n^2 a_{n-2} + a_{n-1}a_{n-2}$	**No**	**No**	**No**	2

7.2.2 2. 求解线性齐次递推关系 (Solving Linear Homogeneous RR)

核心思想 (Core Idea)

寻找形式为 $a_n = r^n$ 的解，其中 r 是常数。代入递推关系 $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$ 得到:

$$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k r^{n-k}$$

两边除以 r^{n-k}:

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_k = 0$$

这就是 **characteristic equation (特征方程)**，它的解称为 **characteristic roots (特征根)**。

7.2.3 3. 一般求解步骤 (General Solution Steps)

Step 1: 写出递推关系对应的特征方程 **Step 2:** 求解特征方程的根 **Step 3:** 根据根的情况写出通解形式 **Step 4:** 用初始条件确定常数

7.2.4 4. 简单例子 (Simple Example)

Example: Solve $a_{n+2} = 3a_{n+1}$ with $a_0 = 4$. **Solution:** **(1)** 特征方程: $r - 3 = 0$ **(2)** 特征根: $r = 3$ **(3)** 通解: $a_n = \alpha \cdot 3^n$ **(4)** 代入初值: $a_0 = \alpha \cdot 3^0 = 4 \implies \alpha = 4$ **(5)** 特解: $a_n = 4 \cdot 3^n$

7.2.5 5. 求解方法概览 (All Solving Methods)

三大求解方法:

1. **Iterative approach (迭代法):** 反复代入简化，如汉诺塔
2. **Characteristic equation method (特征方程法):** 系统求解齐次递推
3. **Generating functions (生成函数法):** 通用方法，也适用于非齐次

7.2.6 6. 解的结构 (Structure of Solutions)

Nonhomogeneous Case (非齐次情况)

对于非齐次递推:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + F(n)$$

解的结构为:

$$a_n = a_n^{(p)} + a_n^{(h)}$$

其中:

- $a_n^{(h)}$ 是对应的**齐次递推**的通解 (homogeneous solution)
- $a_n^{(p)}$ 是原非齐次递推的**特解** (particular solution)

7.2.7 7. 小结和考点提示 (Summary & Exam Tips)

必掌握概念

- Linear homogeneous recurrence relation
- Characteristics equation (特征方程)
- Characteristic roots (特征根)
- Degree of recurrence (递推关系的阶)

求解步骤

1. Write characteristic equation
2. Find roots
3. Write general solution form
4. Use initial conditions to find constants

考试提示

- 注意区分 homogeneous 和 nonhomogeneous
- 初始条件的个数必须等于 degree
- 求解时一定**先写特征方程，再求根**
- Degree 1 是最简单的情况，不要出错
- 会判断递推关系是否为 linear, homogeneous, constant coefficients

英文关键术语

- Linear homogeneous recurrence relation
- Constant coefficients
- Degree
- Characteristic equation
- Characteristic roots
- General solution / Particular solution
- Initial conditions
- Associated homogeneous recurrence relation

7.3 Chapter 8.3: Homogeneous Recurrence Relations (齐次递推关系)

7.3.1 1. 求解线性齐次递推关系 (Solving Linear Homogeneous RR)

核心思路 (Core Idea)

寻找形式为 $a_n = r^n$ 的解，其中 r 是常数。代入递推关系 $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$ 得到:

$$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k r^{n-k}$$

两边除以 r^{n-k} 得到 **characteristic equation (特征方程)**:

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_k = 0$$

7.3.2 2. Degree 2 —两个不同实根 (Theorem 1)

Theorem 1 (Distinct Roots)

Let c_1, c_2 be real numbers. Suppose that $r^2 - c_1 r - c_2 = 0$ has **two distinct roots** $r_1 \neq r_2$. > Then the sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ **if and only if** >

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$$

> for $n = 0, 1, 2, \ldots$, where α_1, α_2 are constants.

证明思路 (Proof Sketch):

(1) ” 如果” 方向 (If): 若 $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$，代入验证:

$$c_1 a_{n-1} + c_2 a_{n-2} = c_1(\alpha_1 r_1^{n-1} + \alpha_2 r_2^{n-1}) + c_2(\alpha_1 r_1^{n-2} + \alpha_2 r_2^{n-2})$$

$$= \alpha_1 r_1^{n-2}(c_1 r_1 + c_2) + \alpha_2 r_2^{n-2}(c_1 r_2 + c_2)$$

$$= \alpha_1 r_1^{n-2} r_1^2 + \alpha_2 r_2^{n-2} r_2^2 = \alpha_1 r_1^n + \alpha_2 r_2^n = a_n$$

(2) ” 仅当” 方向 (Only if): 给定初始条件 $a_0 = C_0, a_1 = C_1$，可解出常数:

$$\alpha_1 = \frac{C_1 - C_0 r_2}{r_1 - r_2}, \quad \alpha_2 = \frac{C_0 r_1 - C_1}{r_1 - r_2}$$

示例 (Example)

Question: Solve $a_n = a_{n-1} + 2a_{n-2}$ with $a_0 = 2, a_1 = 7$. **Solution:** **(1)** 特征方程: $r^2 - r - 2 = 0$ **(2)** 因式分解: $(r-2)(r+1) = 0$, 特征根: $r_1 = 2, r_2 = -1$ **(3)** 通解形式: $a_n = \alpha_1 \cdot 2^n + \alpha_2 \cdot (-1)^n$ **(4)** 代入初值:

$$\begin{cases} a_0 = \alpha_1 + \alpha_2 = 2 \\ a_1 = 2\alpha_1 - \alpha_2 = 7 \end{cases}$$

(5) 求解得: $\alpha_1 = 3, \alpha_2 = -1$ **(6)** 特解: $a_n = 3 \cdot 2^n - (-1)^n$

7.3.3 3. Fibonacci 数的显式公式 (Explicit Formula for Fibonacci)

Fibonacci 递推

$$f_n = f_{n-1} + f_{n-2}, \quad f_0 = 0, f_1 = 1$$

求解过程

(1) 特征方程: $r^2 - r - 1 = 0$ **(2)** 特征根:

$$r_1 = \frac{1+\sqrt{5}}{2}, \quad r_2 = \frac{1-\sqrt{5}}{2}$$

(3) 通解: $f_n = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$ **(4)** 代入 $f_0 = 0, f_1 = 1$:

$$\begin{cases} \alpha_1 + \alpha_2 = 0 \\ \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right) + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right) = 1 \end{cases}$$

(5) 解得: $\alpha_1 = \frac{1}{\sqrt{5}}, \alpha_2 = -\frac{1}{\sqrt{5}}$ **(6)** Binet's formula:

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

记法: 令 $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$ (golden ratio), $\hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -0.618$，则 $f_n = \frac{\phi^n - \hat{\phi}^n}{\sqrt{5}}$.

7.3.4 4. Degree 2 —重根情况 (Theorem 2)

Theorem 2 (Repeated Root)

Let c_1, c_2 be real numbers with $c_2 \neq 0$. Suppose that $r^2 - c_1 r - c_2 = 0$ has **one repeated root** r_0. > Then the sequence $\{a_n\}$ is a solution of $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ **if and only if** >

$$a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n$$

> for $n = 0, 1, 2, \ldots$, where α_1, α_2 are constants.

注意: 重根时，通解中第二项多了一个因子 n。

示例 1 (Example 1)

Question: Solve $a_n = 6a_{n-1} - 9a_{n-2}$ with $a_0 = 1, a_1 = 6$. **Solution:** **(1)** 特征方程: $r^2 - 6r + 9 = 0$ **(2)** 特征根: $(r-3)^2 = 0$, $r_0 = 3$ (repeated, multiplicity 2) **(3)** 通解: $a_n = \alpha_1 \cdot 3^n + \alpha_2 \cdot n \cdot 3^n$ **(4)** 代入初值:

$$\begin{cases} a_0 = \alpha_1 = 1 \\ a_1 = 3\alpha_1 + 3\alpha_2 = 6 \implies \alpha_2 = 1 \end{cases}$$

(5) 特解: $a_n = 3^n + n \cdot 3^n$

示例 2 (Example 2)

Question: Solve $a_n = 6a_{n-1} - 9a_{n-2}$ with $a_0 = 1, a_1 = 3$. **Solution:** **(1)** 特征方程: $r^2 - 6r + 9 = 0$, $r_0 = 3$ (重根) **(2)** 通解: $a_n = \alpha_1 \cdot 3^n + \alpha_2 \cdot n \cdot 3^n$ **(3)** 代入初值:

$$\begin{cases} a_0 = \alpha_1 = 1 \\ a_1 = 3\alpha_1 + 3\alpha_2 = 3 \implies \alpha_2 = 0 \end{cases}$$

(4) 特解: $a_n = 3^n$

7.3.5 5. 任意阶—互异根 (Theorem 3)

Theorem 3 (Arbitrary Degree, Distinct Roots)

Let $c_1, c_2, \ldots, c_k$ be real numbers. Suppose the characteristic equation >

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_k = 0$$

> has k **distinct roots** $r_1, r_2, \ldots, r_k$. > Then $\{a_n\}$ is a solution of $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$ **if and only if** >

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \cdots + \alpha_k r_k^n$$

> for $n = 0, 1, 2, \ldots$, where $\alpha_1, \alpha_2, \ldots, \alpha_k$ are constants.

示例 (Example)

Question: Solve $a_n = 6a_{n-1} - 11a_{n-2} + 6a_{n-3}$ with $a_0 = 2, a_1 = 5, a_2 = 15$. **Solution:** **(1)** 特征方程: $r^3 - 6r^2 + 11r - 6 = 0$ **(2)** 因式分解: $(r-1)(r-2)(r-3) = 0$ 特征根: $r_1 = 1, r_2 = 2, r_3 = 3$ **(3)** 通解: $a_n = \alpha_1 \cdot 1^n + \alpha_2 \cdot 2^n + \alpha_3 \cdot 3^n$ **(4)** 代入初值:

$$\begin{cases} a_0 = \alpha_1 + \alpha_2 + \alpha_3 = 2 \\ a_1 = \alpha_1 + 2\alpha_2 + 3\alpha_3 = 5 \\ a_2 = \alpha_1 + 4\alpha_2 + 9\alpha_3 = 15 \end{cases}$$

(5) 解得: $\alpha_1 = 1, \alpha_2 = -1, \alpha_3 = 2$ **(6)** 特解: $a_n = 1 - 2^n + 2 \cdot 3^n$

7.3.6 6. 任意阶—允许重根 (Theorem 4)

Theorem 4 (General Case with Repeated Roots)

Suppose the characteristic equation has t distinct roots $r_1, r_2, \ldots, r_t$ with **multiplicities** $m_1, m_2, \ldots, m_t$ respectively (where $m_1 + m_2 + \cdots + m_t = k$). > Then the general solution is: >

$$a_n = \sum_{i=1}^{t} \left(\sum_{j=0}^{m_i - 1} \alpha_{i,j} n^j \right) r_i^n$$

即每个根 r_i 对应一个多项式因子 $\alpha_{i,0} + \alpha_{i,1} n + \cdots + \alpha_{i,m_i-1} n^{m_i - 1}$，乘以 r_i^n。

举例: 特征根为 2, 2, 2, 5, 5, 9

$$a_n = (\alpha_{1,0} + \alpha_{1,1} n + \alpha_{1,2} n^2) \cdot 2^n + (\alpha_{2,0} + \alpha_{2,1} n) \cdot 5^n + \alpha_{3,0} \cdot 9^n$$

示例 1: 三重根

Question: Solve $a_n = -3a_{n-1} - 3a_{n-2} - a_{n-3}$ with $a_0 = 1, a_1 = -2, a_2 = -1$. **Solution:** **(1)** 特征方程: $r^3 + 3r^2 + 3r + 1 = 0$ **(2)** 因式分解: $(r+1)^3 = 0$, $r = -1$ (multiplicity 3) **(3)** 通解: $a_n = (\alpha_{1,0} + \alpha_{1,1} n + \alpha_{1,2} n^2)(-1)^n$ **(4)** 代入初值:

$$\begin{cases} a_0 = \alpha_{1,0} = 1 \\ a_1 = -\alpha_{1,0} - \alpha_{1,1} - \alpha_{1,2} = -2 \\ a_2 = \alpha_{1,0} + 2\alpha_{1,1} + 4\alpha_{1,2} = -1 \end{cases}$$

(5) 解得: $\alpha_{1,0} = 1, \alpha_{1,1} = 3, \alpha_{1,2} = -2$ **(6)** 特解: $a_n = (1 + 3n - 2n^2)(-1)^n$

示例 2: 三重根 + 单根

Question: Solve $a_n = 3a_{n-1} + 6a_{n-2} - 28a_{n-3} + 24a_{n-4}$ with $a_0 = -2, a_1 = 12, a_2 = 22, a_3 = 222$. **Solution:** **(1)** 特征方程: $r^4 - 3r^3 - 6r^2 + 28r - 24 = 0$ **(2)** 因式分解: $(r-2)^3(r+3) = 0$

- $r_1 = 2$ (multiplicity 3)
- $r_2 = -3$ (multiplicity 1)

(3) 通解:

$$a_n = (\alpha_{1,0} + \alpha_{1,1} n + \alpha_{1,2} n^2) \cdot 2^n + \alpha_{2,0} \cdot (-3)^n$$

(4) 代入初值:

$$\begin{cases} a_0 = \alpha_{1,0} + \alpha_{2,0} = -2 \\ a_1 = 2\alpha_{1,0} + 2\alpha_{1,1} + 2\alpha_{1,2} - 3\alpha_{2,0} = 12 \\ a_2 = 4\alpha_{1,0} + 8\alpha_{1,1} + 16\alpha_{1,2} + 9\alpha_{2,0} = 22 \\ a_3 = 8\alpha_{1,0} + 24\alpha_{1,1} + 72\alpha_{1,2} - 27\alpha_{2,0} = 222 \end{cases}$$

(5) 解得: $\alpha_{1,0} = 0, \alpha_{1,1} = 1, \alpha_{1,2} = 2, \alpha_{2,0} = -2$ **(6)** 特解: $a_n = (n + 2n^2) \cdot 2^n - 2(-3)^n$

7.3.7 7. 定理总结 (Summary of Theorems)

Theorem	Degree	Roots	General Solution Form
Theorem 1	2	Two distinct: $r_1 \neq r_2$	$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$
Theorem 2	2	One repeated: r_0	$a_n = (\alpha_1 + \alpha_2 n) r_0^n$
Theorem 3	k	k distinct: $r_1, \ldots, r_k$	$a_n = \sum_{i=1}^{k} \alpha_i r_i^n$
Theorem 4	k	Repeated roots with multi. m_i	$a_n = \sum_{i=1}^{t} (\sum_{j=0}^{m_i-1} \alpha_{i,j} n^j) r_i^n$

7.3.8 8. 小结和考点提示 (Summary & Exam Tips)

核心考点

1. **特征方程法三步走:**

- 写特征方程 → 求根 → 写通解形式

- 代入初值 → 解常数 → 得特解

1. **重根的处理:**
- 重根 r_0 会带来 $nr_0^n, n^2r_0^n, \ldots$ 等项
- 不要忘记 n 的因子!

1. **Fibonacci 显式公式:**
- 必须能推导
- 公式中的 $\frac{1\pm\sqrt{5}}{2}$ 是关键

常见错误 (Common Mistakes)
- 忘写特征方程，直接猜解
- 重根时忘记加 n 项
- 初始条件代入时次数搞错
- degree 为 k 时只给了少于 k 个初始条件

英文术语 (English Terms)
- Homogeneous recurrence relation
- Characteristic equation / characteristic roots
- Linear homogeneous with constant coefficients
- Degree / Multiplicity
- Distinct roots / Repeated roots
- General solution / Particular solution
- Fibonacci numbers / Binet's formula
- Golden ratio $\phi = \frac{1+\sqrt{5}}{2}$

7.4 Chapter 8.4: Nonhomogeneous Recurrence Relations & Divide-and-Conquer

7.5 Part A: Nonhomogeneous Recurrence Relations (非齐次递推关系)

7.5.1 1. 定义 (Definition)

Linear Nonhomogeneous Recurrence Relation

Definition: A **linear nonhomogeneous recurrence relation with constant coefficients** is of the form:

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k} + F(n)$$

where $c_1, c_2, \ldots, c_k$ are real numbers, and $F(n)$ is a function **not identically zero** depending only on n.

Associated Homogeneous Recurrence (相伴齐次递推)

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k}$$

这是去掉 $F(n)$ 后得到的齐次递推关系。

例子 (Examples)

Nonhomogeneous RR	Associated Homogeneous RR
$a_n = a_{n-1} + 2^n$	$a_n = a_{n-1}$
$a_n = a_{n-1} + a_{n-2} + n^2 + n + 1$	$a_n = a_{n-1} + a_{n-2}$
$a_n = a_{n-1} + 2^n$	$a_n = a_{n-1}$
$a_n = a_{n-1} + a_{n-2} + a_{n-3} + n!$	$a_n = a_{n-1} + a_{n-2} + a_{n-3}$

7.5.2 2. 解的结构 (Theorem 5)

Theorem 5 (Structure of Solutions)

If $\{a_n^{(p)}\}$ is a **particular solution** of the nonhomogeneous linear recurrence relation >

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k} + F(n)$$

> then **every solution** is of the form $\{a_n^{(p)} + a_n^{(h)}\}$, where $\{a_n^{(h)}\}$ is a solution of the associated homogeneous recurrence relation.

证明思路: 若 $\{b_n\}$ 是另一个解，则 $\{b_n - a_n^{(p)}\}$ 满足齐次递推。

7.5.3 3. 求解步骤 (Solution Steps)

Step A: 求出对应齐次递推的通解 $g(n)$ (即 $a_n^{(h)}$) **Step B:** 找一个非齐次的特解 $f(n)$ (即 $a_n^{(p)}$) **Step C:** 代入初始条件，确定 $f(n)+g(n)$ 中的参数

7.5.4 4. 如何找特解 (Finding Particular Solution) —Theorem 6

Theorem 6 (Form of Particular Solution)

For the recurrence $a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k} + F(n)$ where >

$$F(n) = (b_tn^t + b_{t-1}n^{t-1} + \cdots + b_1n + b_0) \cdot s^n$$

> **(Case 1)** If s is **NOT** a root of the characteristic equation: >

$$f(n) = (p_tn^t + p_{t-1}n^{t-1} + \cdots + p_1n + p_0) \cdot s^n$$

> **(Case 2)** If s **IS** a root of multiplicity m: >

$$f(n) = n^m \cdot (p_tn^t + p_{t-1}n^{t-1} + \cdots + p_1n + p_0) \cdot s^n$$

核心思想: 特解的形式与 $F(n)$ 形式相同，但如果 s 是特征根，则需要乘上 n^m（m 是重数）。

示例: 判断特解形式

Recurrence: $a_n = 6a_{n-1} - 9a_{n-2} + F(n)$ 特征方程: $r^2 - 6r + 9 = (r-3)^2 = 0$, 所以 $r = 3$ 是重根 (multiplicity 2).

$F(n)$	特解形式
$F(n) = 3^n$	$f(n) = p_0n^23^n$
$F(n) = n^23^n$	$f(n) = n^2(p_2n^2 + p_1n + p_0)3^n$
$F(n) = (n^2+1)3^n$	$f(n) = n^2(p_2n^2 + p_1n + p_0)3^n$
$F(n) = n^22^n$	$f(n) = (p_2n^2 + p_1n + p_0)2^n$ (2 不是特征根)

7.5.5 5. 完整示例 (Complete Examples)

Example 1

Question: Find all solutions of $a_n = 3a_{n-1} + 2n$. Find the solution with $a_1 = 3$. **Solution: (1) 齐次解 (Homogeneous solution):** Associated homogeneous: $a_n = 3a_{n-1}$ 特征方程: $r - 3 = 0$, 根: $r = 3$ 通解: $a_n^{(h)} = \alpha \cdot 3^n$ **(2) 特解 (Particular solution):** $F(n) = 2n = (2n+0) \cdot 1^n$, $s = 1$ 不是特征根设 $f(n) = cn + d$ 代入原方程:

$$cn + d = 3(c(n-1) + d) + 2n$$

$$cn + d = 3cn - 3c + 3d + 2n$$

$$cn + d = (3c+2)n + (3d - 3c)$$

比较系数:

$$\begin{cases} c = 3c + 2 \implies c = -1 \\ d = 3d - 3c \implies d = 3d + 3 \implies d = -\frac{3}{2} \end{cases}$$

特解: $a_n^{(p)} = -n - \frac{3}{2}$ **(3)** 通解: $a_n = -n - \frac{3}{2} + \alpha \cdot 3^n$ **(4)** 代入 $a_1 = 3$:

$$3 = -1 - \frac{3}{2} + 3\alpha \implies 3 = -\frac{5}{2} + 3\alpha \implies \alpha = \frac{11}{6}$$

(5) 特解: $a_n = -n - \frac{3}{2} + \frac{11}{6} \cdot 3^n$

Example 2

Question: Solve $a_n = a_{n-1} + n$ with $a_1 = 1$. **Solution: (1) 齐次解:** $a_n^{(h)} = c$ (常数) 特征方程: $r - 1 = 0$, $r = 1$ **(2) 特解:** $F(n) = n = n \cdot 1^n$, $s = 1$ 是特征根 (multiplicity 1) 设 $f(n) = n(p_1n + p_0) = p_1n^2 + p_0n$ 代入:

$$p_1n^2 + p_0n = p_1(n-1)^2 + p_0(n-1) + n$$

展开:

$$p_1n^2 + p_0n = p_1n^2 - 2p_1n + p_1 + p_0n - p_0 + n$$

比较系数:

$$\begin{cases} n^2 : p_1 = p_1 \\ n^1 : p_0 = -2p_1 + p_0 + 1 \implies p_1 = \frac{1}{2} \\ n^0 : 0 = p_1 - p_0 \implies p_0 = \frac{1}{2} \end{cases}$$

特解: $f(n) = \frac{1}{2}n^2 + \frac{1}{2}n = \frac{n(n+1)}{2}$ 注意这里其实不需要加齐次解部分，因为初始条件 $a_1 = 1$ 代入 $f(n)$ 刚好成立。但完整做法应该是通解 $a_n = \frac{n(n+1)}{2} + c$，代入 $a_1 = 1$ 得 $c = 0$。**结果:** $a_n = \frac{n(n+1)}{2}$

Example 3: 非齐次项为两项之和

Question: Solve $a_n = 5a_{n-1} - 6a_{n-2} + 2^n + 3^n$, with $a_0 = 0, a_1 = 1$. **Solution: (1)** 特征方程: $r^2 - 5r + 6 = (r-2)(r-3) = 0$, 根 $r_1 = 2, r_2 = 3$ **(2) 齐次解:** $a_n^{(h)} = c_1 \cdot 2^n + c_2 \cdot 3^n$ **(3)** $F(n) = 2^n + 3^n$, 其中 2 和 3 都是特征根 (single root) 对 2^n 部分: 特解形式为 $p_0 n \cdot 2^n$ 对 3^n 部分: 特解形式为 $p_1 n \cdot 3^n$ 所以: $f(n) = p_0 n \cdot 2^n + p_1 n \cdot 3^n$ **(4)** 代入原方程求解 p_0, p_1，然后加上齐次解，再用初始条件确定所有常数。

7.5.6 6. 联立递推关系 (Simultaneous Recurrence Relations)

Example:

$$\begin{cases} a_n = 3a_{n-1} + 2b_{n-1} \\ b_n = a_{n-1} + 2b_{n-1} \end{cases} \qquad a_0 = 1, b_0 = 2$$

解法: 通过代入消元，转化为单一递推关系。

7.6 Part B: Divide-and-Conquer Algorithms (分治算法)

7.6.1 1. 分治算法范式 (Divide-and-Conquer Paradigm)

三步策略

1. **Divide:** 将问题分成两个或多个更小的子问题
2. **Conquer:** 递归地求解子问题
3. **Combine:** 将子问题的解合并为原问题的解

分治递推关系的一般形式

$$f(n) = af(n/b) + g(n)$$

其中:

- a = 子问题的个数
- n/b = 每个子问题的大小
- $g(n)$ = 分解和合并所需的额外操作数

7.6.2 2. 经典例子 (Classic Examples)

Binary Search (二分查找)

描述: 在一个大小为 n 的有序序列中查找元素，每次比较后问题规模减半。

$$f(n) = f(n/2) + 2$$

其中 2 次比较：一次判断是否在上下半区，一次判断序列是否还有元素。**复杂度:** $f(n) = O(\log n)$

Merge Sort (归并排序)

描述: 将列表分成两个大小为 $n/2$ 的子列表，分别排序，然后合并。

$$M(n) = 2M(n/2) + n$$

其中 n 次比较用于合并两个有序列表。**复杂度:** $M(n) = O(n \log n)$

Fast Integer Multiplication (快速整数乘法)

描述: 两个 n 位二进制整数相乘，利用以下恒等式减少乘法次数。**技巧:** 将 n 位整数分成两个 $n/2$ 位的块: 设 $a = A_1 2^{n/2} + A_0$, $b = B_1 2^{n/2} + B_0$

$$ab = A_1 B_1 2^n + (A_1 B_0 + A_0 B_1) 2^{n/2} + A_0 B_0$$

优化为:

$$ab = A_1 B_1 2^n + ((A_1 + A_0)(B_1 + B_0) - A_1 B_1 - A_0 B_0) 2^{n/2} + A_0 B_0$$

只需要 **3 次** $n/2$ 位的乘法（而不是 4 次）。**递推关系:**

$$f(n) = 3f(n/2) + Cn$$

复杂度: $f(n) = O(n^{\log_2 3}) \approx O(n^{1.6})$，比直接乘法的 $O(n^2)$ 更快。

7.6.3 3. 分治递推的估计 (Theorems)

Theorem 1 (Simple Divide-and-Conquer)

Let f be an increasing function satisfying $f(n) = af(n/b) + c$ whenever n is divisible by b, where $a \geq 1$, $b \geq 2$, and $c > 0$. > If $n = b^k$: - **If** $a = 1$**:** $f(n) = f(1) + c \log_b n \implies f(n) \in \Theta(\log n)$ - **If** $a > 1$**:** $f(n) \in \Theta(n^{\log_b a})$

推导过程 ($n = b^k$)**:**

$$\begin{aligned} f(n) &= af(n/b) + C \\ &= a^2 f(n/b^2) + aC + C \\ &= a^3 f(n/b^3) + a^2 C + aC + C \\ &\vdots \\ &= a^k f(1) + C(a^{k-1} + a^{k-2} + \cdots + a + 1) \end{aligned}$$

当 $a = 1$: $f(n) = f(1) + Ck = f(1) + C \log_b n$ 当 $a > 1$: $f(n) = a^k f(1) + C \frac{a^k - 1}{a - 1} = C_1 n^{\log_b a} - C_2$

Theorem 2 (Master Theorem)

Let f be an increasing function satisfying >

$$f(n) = af(n/b) + cn^d$$

> whenever $n = b^k$, where $a \geq 1$, $b \geq 2$, $c > 0$, $d \geq 0$. > Then: >

$$> f(n) \in \begin{cases} > \Theta(n^d) & \text{if } a < b^d \\ > \Theta(n^d \log n) & \text{if } a = b^d \\ > \Theta(n^{\log_b a}) & \text{if } a > b^d > \end{cases} >$$

7.6.4 4. Master Theorem 应用示例

Binary Search

$f(n) = f(n/2) + 2 = 1 \cdot f(n/2) + 2 \cdot n^0$ $a = 1, b = 2, d = 0$ $a = b^d = 1 \implies f(n) = \Theta(n^0 \log n) = \Theta(\log n)$

Merge Sort

$M(n) = 2M(n/2) + n = 2 \cdot f(n/2) + 1 \cdot n^1$ $a = 2, b = 2, d = 1$ $a = b^d = 2 \implies M(n) = \Theta(n^1 \log n) = \Theta(n \log n)$

Fast Integer Multiplication

$f(n) = 3f(n/2) + Cn = 3 \cdot f(n/2) + C \cdot n^1$ $a = 3, b = 2, d = 1$ $a = 3 > b^d = 2 \implies f(n) = \Theta(n^{\log_2 3}) \approx \Theta(n^{1.585})$

7.6.5 5. 小结和考点提示 (Summary & Exam Tips)

Part A: Nonhomogeneous RR

三步法:

1. 求齐次通解 $a_n^{(h)}$
2. 找非齐次特解 $a_n^{(p)}$ (注意 s 是否是特征根!)
3. 合起来 + 初值确定常数

Theorem 6 是重点:

- $F(n) = (polynomial) \cdot s^n$
- s 不是特征根 → 直接用多项式 × s^n
- s 是 m 重根 → 乘 n^m
- 如果 $F(n)$ 是多项式的和，分别处理再相加 (superposition)

Part B: Divide-and-Conquer

Master Theorem 三种情况:

- $a < b^d$: $\Theta(n^d)$
- $a = b^d$: $\Theta(n^d \log n)$
- $a > b^d$: $\Theta(n^{\log_b a})$

必知例子及其复杂度:

Algorithm	Recurrence	Complexity
Binary Search	$f(n) = f(n/2) + 2$	$O(\log n)$
Merge Sort	$M(n) = 2M(n/2) + n$	$O(n \log n)$
Fast Multiplication	$f(n) = 3f(n/2) + Cn$	$O(n^{\log_2 3}) \approx O(n^{1.6})$

英文术语 (English Terms)

- Nonhomogeneous recurrence relation
- Associated homogeneous recurrence
- Particular solution
- Particular solution form (Theorem 6)
- Superposition
- Divide-and-conquer

- Master Theorem
- Binary search / Merge sort
- Fast multiplication of integers
- Big-O / Theta notation

7.7 Chapter 8.5: Generating Functions (生成函数)

7.7.1 1. 定义 (Definition)

Generating Function

Definition: The **generating function (生成函数)** for the sequence $a_0, a_1, a_2, \ldots, a_k, \ldots$ of real numbers is the infinite series:

$$G(x) = a_0 + a_1x + a_2x^2 + \cdots + a_kx^k + \cdots = \sum_{k=0}^{\infty} a_kx^k$$

Herbert Wilf 的名言: "A generating function is a clothesline on which we hang up a sequence of numbers for display." **注意:** 通常忽略收敛性问题，主要关注形式幂级数的代数运算。

有限序列的生成函数

对于有限序列 $a_0, a_1, \ldots, a_n$，可以扩展为无限序列（令 $a_{n+1} = a_{n+2} = \cdots = 0$），生成函数就是一个 n 次多项式:

$$G(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

7.7.2 2. 常见序列的生成函数 (Useful Generating Functions)

核心公式表

序列 (Sequence)	生成函数 (Generating Function)
(1) $a_k = \binom{n}{k}$	$(1+x)^n$
(2) $a_k = \binom{n}{k}a^k$	$(1+ax)^n$
(3) $1, 1, 1, \ldots$	$\frac{1}{1-x}$
(4) $1, 1, \ldots, 1$ ($n+1$ 个 1)	$\frac{1-x^{n+1}}{1-x} = 1 + x + \cdots + x^n$
(5) $a_k = a^k$	$\frac{1}{1-ax}$
(6) $a_k = k+1$	$\frac{1}{(1-x)^2}$
(7) $a_k = \binom{n+k-1}{k}$	$\frac{1}{(1-x)^n}$
(8) $a_k = (-1)^k\binom{n+k-1}{k}$	$(1+x)^{-n}$
(9) $a_k = \binom{n+k-1}{k}a^k$	$\frac{1}{(1-ax)^n}$
(10) $a_k = \frac{1}{k!}$	e^x
(11) $a_k = \frac{(-1)^k}{k+1}$	$\frac{\ln(1+x)}{x}$

常用公式推导示例

Example 1: 序列 $1, 1, 1, \ldots$ (所有项都是 1)

$$G(x) = \sum_{k=0}^{\infty} x^k = \frac{1}{1-x}, \quad |x| < 1$$

Example 2: 序列 $a_k = a^k$ (几何级数)

$$G(x) = \sum_{k=0}^{\infty} a^kx^k = \frac{1}{1-ax}, \quad |ax| < 1$$

Example 3: 序列 $0, 1, 2, 3, 4, 5, \ldots$ (即 $a_k = k$)

$$G(x) = \sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

推导: $\sum_{k=0}^{\infty} kx^k = x \cdot \left(\frac{1}{1-x}\right)' = x \cdot \frac{1}{(1-x)^2}$ **Example 4:** 有限序列 $1, 1, 1, 1, 1, 1$ (6 个 1)

$$G(x) = 1 + x + x^2 + x^3 + x^4 + x^5 = \frac{x^6 - 1}{x - 1} = \frac{1 - x^6}{1 - x}$$

Example 5: 序列 $a_k = \binom{m}{k}$ (二项式系数)

$$G(x) = \sum_{k=0}^{m} \binom{m}{k} x^k = (1+x)^m$$

7.7.3 3. 生成函数的运算性质 (Operations on Generating Functions)

Theorem 1 (Power Series Properties)

设 $f(x) = \sum_{k=0}^{\infty} a_kx^k$ 和 $g(x) = \sum_{k=0}^{\infty} b_kx^k$，则:

运算	结果
(1) $f(x) + g(x)$	$\sum_{k=0}^{\infty}(a_k + b_k)x^k$
(2) $\alpha f(x)$	$\sum_{k=0}^{\infty}(\alpha a_k)x^k$, $\alpha \in \mathbb{R}$
(3) $x \cdot f'(x)$	$\sum_{k=0}^{\infty} ka_kx^k$
(4) $f(\alpha x)$	$\sum_{k=0}^{\infty} \alpha^k a_kx^k$
(5) $f(x) \cdot g(x)$	$\sum_{k=0}^{\infty}\left(\sum_{j=0}^{k} a_jb_{k-j}\right)x^k$ (卷积, convolution)

性质 (5) 的详细解释:

$$f(x)g(x) = (a_0 + a_1x + a_2x^2 + \ldots)(b_0 + b_1x + b_2x^2 + \ldots)$$

$$= a_0b_0 + (a_0b_1 + a_1b_0)x + (a_0b_2 + a_1b_1 + a_2b_0)x^2 + \ldots$$

即 x^k 的系数是 $\sum_{j=0}^{k} a_jb_{k-j}$。

应用示例: 求 $b_k = \sum_{i=0}^{k} a_i$ 的生成函数

若序列 $\{a_k\}$ 的生成函数为 $G(x)$，则序列 $\{b_k\}$ 其中 $b_k = \sum_{i=0}^{k} a_i$ 的生成函数为:

$$\frac{G(x)}{1-x}$$

因为 $\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k$ 对应序列 $1, 1, 1, \ldots$。

7.7.4 4. Extended Binomial Coefficient (广义二项式系数)

Definition (定义)

Let u be a real number and k a nonnegative integer. The **extended binomial coefficient** is defined as: >

$$> \binom{u}{k} = \begin{cases} > \frac{u(u-1)\cdots(u-k+1)}{k!} & \text{if } k > 0 \\ > 1 & \text{if } k = 0 > \end{cases} >$$

例子

(1) $\binom{1/2}{3} = \frac{(1/2)(-1/2)(-3/2)}{3!} = \frac{3/8}{6} = \frac{1}{16}$ **(2)** $\binom{-n}{k} = \frac{(-n)(-n-1)\cdots(-n-k+1)}{k!} = (-1)^k\binom{n+k-1}{k}$

Theorem 2 (Extended Binomial Theorem)

Let x be a real number with $|x| < 1$ and u be a real number. Then: >

$$(1+x)^u = \sum_{k=0}^{\infty} \binom{u}{k} x^k$$

重要推论

$$(1-x)^{-n} = \sum_{k=0}^{\infty} \binom{n+k-1}{k} x^k = \sum_{k=0}^{\infty} \binom{n+k-1}{n-1} x^k$$

$$(1+x)^{-n} = \sum_{k=0}^{\infty} (-1)^k \binom{n+k-1}{k} x^k$$

7.7.5 5. 生成函数在计数中的应用

5.1 基本思路

生成函数将计数问题转化为求幂级数展开中特定项的系数。

5.2 k-Combinations (k 组合)

无重复选取:

$$(1+x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k$$

x^k 的系数就是 $C(n, k)$。

5.3 Combinations with Repetitions (有重复选取)

Example: 从 n 个元素中可重复地选取 r 个元素（允许重复的组合数）。每个元素可以选择 0 次、1 次、2 次、…次，所以每个元素贡献因子:

$$1 + x + x^2 + x^3 + \cdots = \frac{1}{1-x}$$

n 个元素的生成函数:

$$G(x) = \left(\frac{1}{1-x}\right)^n = \sum_{k=0}^{\infty} \binom{n+k-1}{k} x^k$$

所以结果就是 $\binom{n+r-1}{r} = C(n+r-1, r)$。

5.4 带约束条件的组合计数

Example: 求 $e_1+e_2+e_3=17$ 的非负整数解个数，其中 $2\le e_1\le 5$, $3\le e_2\le 6$, $4\le e_3\le 7$。每个变量的生成函数:

- e_1: $x^2+x^3+x^4+x^5$
- e_2: $x^3+x^4+x^5+x^6$
- e_3: $x^4+x^5+x^6+x^7$

乘积中 x^{17} 的系数即为答案（结果为 3）。

5.5 购物问题 (Token Problem)

Problem: 用 1,2, 5　r 商品。**(1) 不计顺序 (Order does NOT matter):**

$$G(x)=(1+x+x^2+\dots)(1+x^2+x^4+\dots)(1+x^5+x^{10}+\dots)=\frac{1}{1-x}\cdot\frac{1}{1-x^2}\cdot\frac{1}{1-x^5}$$

x^r 的系数即答案。**(2) 考虑顺序 (Order matters):** 使用恰好 n 个代币: $(x+x^2+x^5)^n$ 所有可能:

$$\sum_{n=0}^{\infty}(x+x^2+x^5)^n=\frac{1}{1-(x+x^2+x^5)}$$

x^r 的系数即答案。

7.7.6　6. 用生成函数解递推关系 (Solving RR using Generating Functions)

方法 (Method)

1. 用递推关系列出生成函数 $G(x)=\sum_{n=0}^{\infty}a_nx^n$
2. 将递推代入，得到关于 $G(x)$ 的方程
3. 解出 $G(x)$ 的封闭形式
4. 展开 $G(x)$ 为幂级数，读出 a_n 的表达式

完整示例

Question: Solve $a_n=2a_{n-1}+3a_{n-2}+4n+6$ with $a_0=20, a_1=60$. **Solution: (1)** 设 $G(x)=\sum_{n=0}^{\infty}a_nx^n$，递推乘以 x^n 并对 $n\ge 2$ 求和:

$$\sum_{n=2}^{\infty}a_nx^n=2\sum_{n=2}^{\infty}a_{n-1}x^n+3\sum_{n=2}^{\infty}a_{n-2}x^n+4\sum_{n=2}^{\infty}nx^n+6\sum_{n=2}^{\infty}x^n$$

(2) 用 $G(x)$ 表示:

$$G(x)-a_0-a_1x=2x(G(x)-a_0)+3x^2G(x)$$
$$+4\left[\frac{x}{(1-x)^2}-x\right]+6\left[\frac{1}{1-x}-1-x\right]$$

(3) 代入 $a_0=20, a_1=60$，化简:

$$G(x)(1-2x-3x^2)=20-80x+\frac{4x}{(1-x)^2}+\frac{6}{1-x}-6-6x+\dots$$

(4) 解出 $G(x)$:

$$G(x)=\frac{20-80x+40x^2-40x^3}{(1-x)^2(1-2x-3x^2)}$$

(5) 分解为部分分式:

$$G(x)=\frac{5/16}{1-4x}-\frac{2/3}{1-x}+\frac{31/20}{(1-x)^2}+\frac{67/4}{1-3x}$$

(6) 展开:

$$a_n=\frac{5}{16}\cdot 4^n-\frac{2}{3}\cdot 1^n+\frac{31}{20}\cdot(n+1)+\frac{67}{4}\cdot 3^n$$

7.7.7　7. 用生成函数证明组合恒等式

Example: 用生成函数证明 Pascal 恒等式: $\binom{n}{r}=\binom{n-1}{r}+\binom{n-1}{r-1}$ **Proof:** 考虑 $(1+x)^n=(1+x)(1+x)^{n-1}$ 左边: $\sum_{r=0}^{n}\binom{n}{r}x^r$ 右边: $(1+x)\sum_{r=0}^{n-1}\binom{n-1}{r}x^r=\sum_{r=0}^{n-1}\binom{n-1}{r}x^r+\sum_{r=0}^{n-1}\binom{n-1}{r}x^{r+1}=\sum_{r=0}^{n-1}\binom{n-1}{r}x^r+\sum_{r=1}^{n}\binom{n-1}{r-1}x^r$ 比较 x^r 的系数:

$$\binom{n}{r}=\binom{n-1}{r}+\binom{n-1}{r-1}$$

7.7.8　8. 小结和考点提示 (Summary & Exam Tips)

核心考点

1. **常见序列的生成函数表必须熟记** (Section 2 中的表格)
2. **Extended Binomial Theorem** 及其应用
3. **生成函数解递推关系**的完整步骤
4. **用生成函数做组合计数**:

- 无重复选取: $(1+x)^n$
- 可重复选取: $\frac{1}{(1-x)^n}$
- 带约束: 为每个变量设对应生成函数

解题步骤 (生成函数解递推)

1. 设 $G(x)=\sum a_nx^n$
2. 递推两边乘 x^n 并对 n 从某值开始求和
3. 将所有求和用 $G(x)$ 表示
4. 解出 $G(x)$ 的封闭形式
5. 展开为部分分式
6. 读出 a_n 表达式

英文关键术语

- Generating function
- Power series
- Extended binomial coefficient
- Extended binomial theorem
- Convolution
- Partial fractions (部分分式)
- Coefficient extraction (提取系数)
- Combinatorial identity
- Combinations with repetitions

7.8　Chapter 8.6: Inclusion-Exclusion (容斥原理)

7.9　Part A: The Principle of Inclusion-Exclusion (8.5)

7.9.1　1. 两个集合的情况 (Two Sets)

$$|A\cup B|=|A|+|B|-|A\cap B|$$

Example: 离散数学课上，主修 CS 的学生有 25 人，主修 Math 的有 13 人，同时主修两门的有 8 人。班上有多少人?

$$|CS\cup Math|=25+13-8=30$$

7.9.2　2. 三个集合的情况 (Three Sets)

$$|A\cup B\cup C|=|A|+|B|+|C|-|A\cap B|-|A\cap C|-|B\cap C|+|A\cap B\cap C|$$

图示记忆: 加单个、减两两交、加三三交。**Example:** 选西班牙语的学生 1232 人，法语 879 人，俄语 114 人。其中同时选西法 103 人，西俄 23 人，法俄 14 人。至少选一门的有 2092 人。问三门都选的有多少人?

$$2092=1232+879+114-103-23-14+|S\cap F\cap R|$$

$$|S\cap F\cap R|=7$$

7.9.3　3. 一般形式 (General Form)

Theorem (Principle of Inclusion-Exclusion)

Let $A_1,A_2,\dots,A_n$ be finite sets. Then: >

$$>\left|\bigcup_{i=1}^{n}A_i\right|=\sum_{1\le i\le n}|A_i|-\sum_{1\le i<j\le n}|A_i\cap A_j|+\sum_{1\le i<j<k\le n}|A_i\cap A_j\cap A_k|-\dots+(-1)^{n+1}|A_1\cap A_2\cap\dots\cap A_n|>$$

规律: - 奇数个集合的交 (1, 3, 5, ...): **加号** - 偶数个集合的交 (2, 4, 6, ...): **减号**

证明思路 (Proof Sketch)

考虑一个属于恰好 r 个集合的元素 a。它在右边被计数的次数为:

$$\binom{r}{1}-\binom{r}{2}+\binom{r}{3}-\dots+(-1)^{r+1}\binom{r}{r}=1$$

由二项式定理推论: $\sum_{k=0}^{r}(-1)^k\binom{r}{k}=0$，所以 $\sum_{k=1}^{r}(-1)^{k+1}\binom{r}{k}=1$. 因此每个元素被恰好计数一次。

7.9.4 4. 应用示例

Example: 不被 5、6、8 整除的数

Question: 求不超过 1000 的正整数中，不能被 5、6、8 中任何一个整除的数的个数。**Solution:** 设 $U = \{1, 2, \ldots, 1000\}$，

- A: 能被 5 整除的数
- B: 能被 6 整除的数
- C: 能被 8 整除的数

所求 $= |\overline{A} \cap \overline{B} \cap \overline{C}| = |U| - |A \cup B \cup C|$

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

$$\begin{aligned} &= \left\lfloor \frac{1000}{5} \right\rfloor + \left\lfloor \frac{1000}{6} \right\rfloor + \left\lfloor \frac{1000}{8} \right\rfloor \\ &\quad - \left\lfloor \frac{1000}{\mathrm{lcm}(5,6)} \right\rfloor - \left\lfloor \frac{1000}{\mathrm{lcm}(5,8)} \right\rfloor - \left\lfloor \frac{1000}{\mathrm{lcm}(6,8)} \right\rfloor \\ &\quad + \left\lfloor \frac{1000}{\mathrm{lcm}(5,6,8)} \right\rfloor \end{aligned}$$

重要: 交集的元素个数要用 **最小公倍数 (LCM, least common multiple)** 计算!

- $\mathrm{lcm}(5,6) = 30$, $\mathrm{lcm}(5,8) = 40$, $\mathrm{lcm}(6,8) = 24$
- $\mathrm{lcm}(5,6,8) = \mathrm{lcm}(5, \mathrm{lcm}(6,8)) = \mathrm{lcm}(5,24) = 120$

$$= 200 + 166 + 125 - 33 - 25 - 41 + 8 = 400$$

$$|\overline{A} \cap \overline{B} \cap \overline{C}| = 1000 - 400 = 600$$

7.10 Part B: Applications of Inclusion-Exclusion (8.6)

7.10.1 1. 另一种形式 (Alternative Form)

无某组性质的元素计数

设 $P_1, P_2, \ldots, P_n$ 是 n 个性质 (properties)，A_i 是具有性质 P_i 的元素集合。$N(P_1'P_2' \cdots P_n') =$ 同时不具有任何 P_i 的元素个数。由容斥原理:

$$\begin{aligned} N(P_1'P_2' \cdots P_n') = N &- \sum_{1 \le i \le n} N(P_i) + \sum_{1 \le i < j \le n} N(P_iP_j) \\ &- \sum_{1 \le i < j < k \le n} N(P_iP_jP_k) + \cdots + (-1)^n N(P_1P_2 \cdots P_n) \end{aligned}$$

其中 N 是元素总数，$N(P_i)$ 是具有性质 P_i 的元素个数，$N(P_iP_j)$ 是同时具有 P_i 和 P_j 的元素个数，依此类推。

7.10.2 2. 带约束的方程求整数解

Example 1: 求 $x_1 + x_2 + x_3 = 13$ 的非负整数解个数，其中 $x_i < 6$ $(i = 1, 2, 3)$。**Solution:** 设性质 P_i 为 $x_i \ge 6$。所求为 $N(P_1'P_2'P_3')$。$N =$ 无约束时非负整数解总数 $= C(3 + 13 - 1, 13) = C(15, 13) = C(15, 2) = 105$ $N(P_i) = x_i \ge 6$ 时: 令 $y_i = x_i - 6 \ge 0$，则 $y_1 + x_2 + x_3 = 7$，解数为 $C(3 + 7 - 1, 7) = C(9, 7) = 36$ $N(P_iP_j) = x_i \ge 6$ 且 $x_j \ge 6$: 解数为 $C(3 + 1 - 1, 1) = C(3, 1) = 3$ $N(P_1P_2P_3) = x_1 \ge 6, x_2 \ge 6, x_3 \ge 6$: 此时 $x_1 + x_2 + x_3 \ge 18 > 13$，解数为 0。

$$N(P_1'P_2'P_3') = 105 - 3 \times 36 + 3 \times 3 - 0 = 105 - 108 + 9 = 6$$

7.10.3 3. Eratosthenes 筛法 (Sieve of Eratosthenes)

Example 2: 求不超过 100 的素数个数。**思路:**

- 合数必有不超过其平方根的素因子
- 不超过 100 的合数必有不超过 10 的素因子
- 不超过 10 的素数只有: 2, 3, 5, 7
- 所以不超过 100 的素数 = 4 (即 2,3,5,7) + 不能被 2,3,5,7 中任何一个整除的大于 1 的整数

设 P_1: 被 2 整除, P_2: 被 3 整除, P_3: 被 5 整除, P_4: 被 7 整除

$$\begin{aligned} N(P_1'P_2'P_3'P_4') = N &- [N(P_1) + N(P_2) + N(P_3) + N(P_4)] \\ &+ [N(P_1P_2) + N(P_1P_3) + N(P_1P_4) + N(P_2P_3) + N(P_2P_4) + N(P_3P_4)] \\ &- [N(P_1P_2P_3) + N(P_1P_2P_4) + N(P_1P_3P_4) + N(P_2P_3P_4)] \\ &+ N(P_1P_2P_3P_4) \end{aligned}$$

从 1 到 100 共有 100 个数，去掉 1（既不是素数也不是合数），所以 $N = 99$。最后结果 $= 4 + N(P_1'P_2'P_3'P_4') = 25$

7.10.4 4. 满射函数的计数 (Counting Onto Functions)

问题

Example: 从 6 个元素的集合到 3 个元素的集合，有多少个满射函数 (onto functions)? **Solution:** 设值域中的元素为 b_1, b_2, b_3。P_i: b_i 不在值域中。函数是满射当且仅当 P_1, P_2, P_3 都不成立。

- 函数总数: $N = 3^6 = 729$
- $N(P_i) = b_i$ 不在值域中 $= 2^6 = 64$（从剩下 2 个元素中选）
- $N(P_iP_j) = b_i, b_j$ 都不在值域中 $= 1^6 = 1$（只能映射到剩下 1 个元素）
- $N(P_1P_2P_3) = 0$（值域为空不可能）

$$\begin{aligned} \text{Onto} &= 729 - 3 \cdot 64 + 3 \cdot 1 - 0 \\ &= 729 - 192 + 3 = 540 \end{aligned}$$

Theorem 1 (Onto Functions)

Let m and n be positive integers with $m \ge n$. Then there are: $>$

$$n^m - \binom{n}{1}(n-1)^m + \binom{n}{2}(n-2)^m - \cdots + (-1)^{n-1}\binom{n}{n-1}1^m$$

$>$ onto functions from a set with m elements to a set with n elements.

等价形式:

$$\sum_{k=0}^{n} (-1)^k \binom{n}{k} (n-k)^m$$

7.10.5 5. Derangements (错位排列)

定义 (Definition)

A **derangement (错位排列)** is a permutation of n objects that **leaves no object in its original position**.

Example: - 21453 是 12345 的一个 derangement (没有一个数字在原来位置) - 21543 不是 derangement，因为 4 还在原位置

Theorem 2 (Number of Derangements)

The number of derangements of a set with n elements is: $>$

$$D_n = n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}\right)$$

推导 (Derivation)

设 P_i 为排列中元素 i 在原位的性质。Derangement 就是所有 P_i 都不成立的排列。

$$\begin{aligned} D_n &= N(P_1'P_2' \cdots P_n') \\ &= n! - \binom{n}{1}(n-1)! + \binom{n}{2}(n-2)! - \binom{n}{3}(n-3)! + \cdots + (-1)^n \binom{n}{n}(n-n)! \\ &= n! - \frac{n!}{1!} + \frac{n!}{2!} - \frac{n!}{3!} + \cdots + (-1)^n \frac{n!}{n!} \\ &= n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}\right) \end{aligned}$$

帽子检查问题 (Hatcheck Problem)

餐馆员工忘记给 n 顶帽子贴标签。顾客随机取回帽子，所有人都没拿到自己帽子的概率为:

$$\frac{D_n}{n!} = 1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}$$

当 $n \to \infty$ 时，这个概率趋近于 $e^{-1} \approx 0.368$。

7.10.6 6. 小结和考点提示 (Summary & Exam Tips)

核心公式速记

容斥原理 (一般形式):

$$|\bigcup A_i| = \sum |A_i| - \sum |A_i \cap A_j| + \sum |A_i \cap A_j \cap A_k| - \cdots$$

无性质元素计数:

$$N(P_1'P_2' \cdots P_n') = N - \sum N(P_i) + \sum N(P_iP_j) - \sum N(P_iP_jP_k) + \cdots$$

满射函数:

$$m \to n \text{ onto}: \sum_{k=0}^{n} (-1)^k \binom{n}{k} (n-k)^m$$

Derangements:

$$D_n = n! \sum_{k=0}^{n} \frac{(-1)^k}{k!}$$

重要注意事项

1. **交集的元素个数用 LCM (最小公倍数) 计算**，不是直接相乘！
2. **三个集合的容斥:** 先加单、再减两两交、再加三交
3. **Derangement 公式:** 首项是 $n!$，然后交替加减
4. **满射函数公式:** 从 m 个元素到 n 个元素，要求 $m \geq n$

常见错误 (Common Mistakes)

- 忘记 $(-1)^n$ 的符号
- 两两交时只计算了部分组合
- 计算 $\lfloor N/\text{lcm}(a,b) \rfloor$ 时用了乘积代替 lcm
- 满射函数公式中符号搞反

英文术语 (English Terms)

- Inclusion-Exclusion principle (容斥原理)
- Principle of inclusion-exclusion
- Union / Intersection
- Finite sets
- Sieve of Eratosthenes
- Onto functions (满射函数 / 映上函数)
- Derangement (错位排列)
- Hatcheck problem
- Probability of derangement $\to 1/e$
- Properties (性质)
- LCM (Least common multiple)

第八章　Chapter 9: Relations (关系)

8.1 Chapter 9.1: Relations and Their Properties (关系及其性质)

8.1.1 1. Binary Relations (二元关系)

Definition

A **binary relation** R from a set A to a set B is a subset of $A \times B$.

$$R \subseteq A \times B$$

- If $(a,b) \in R$, we write aRb (读作"a relates to b").
- Relations are **more general than functions**: a function is a relation where exactly one element of B is related to each element of A.

Example

Let $A = \{0,1,2\}$, $B = \{a,b\}$. Then

$$R = \{(0,a),(0,b),(1,a),(2,b)\}$$

is a relation from A to B.

Binary Relation on a Set (集合上的二元关系)

A **binary relation on a set** A is a subset of $A \times A$. Example: $A = \{a,b,c\}$, $R = \{(a,a),(a,b),(a,c)\}$ is a relation on A.

Counting Relations (关系的计数)

Question: How many relations are there on a set A with n elements?

Solution: A relation on A is a subset of $A \times A$. Since $|A \times A| = n^2$, and a set with m elements has 2^m subsets, there are:

$$2^{|A|^2} = 2^{n^2}$$

relations on A.

8.1.2 2. Properties of Relations (关系的性质)

Reflexive (自反的)

Definition: R is **reflexive** iff $(a,a) \in R$ for every element $a \in A$. Symbolically: $\forall x \in A\ [(x,x) \in R]$

注意：空关系 $\emptyset$ 只有在 A 是空集时才 reflexive。若 A 非空，则空关系**不是** reflexive，因为没有包含任何 (a,a)。

Example (reflexive): - $R_1 = \{(a,b) \mid a \leq b\}$ —reflexive, 因为 $a \leq a$ 对任意整数成立 - $R_3 = \{(a,b) \mid a = b \text{ or } a = -b\}$ —reflexive - $R_4 = \{(a,b) \mid a = b\}$ —reflexive

Example (not reflexive): - $R_2 = \{(a,b) \mid a > b\}$ —因为 $3 \not> 3$ - $R_5 = \{(a,b) \mid a = b+1\}$ —因为 $3 \neq 3+1$ - $R_6 = \{(a,b) \mid a+b \leq 3\}$ —因为 $4+4 \not\leq 3$

Symmetric (对称的)

Definition: R is **symmetric** iff $(b,a) \in R$ whenever $(a,b) \in R$, for all $a,b \in A$. Symbolically: $\forall x \forall y\ [(x,y) \in R \longrightarrow (y,x) \in R]$

Example (symmetric):

- $R_3 = \{(a,b) \mid a = b \text{ or } a = -b\}$
- $R_4 = \{(a,b) \mid a = b\}$
- $R_6 = \{(a,b) \mid a+b \leq 3\}$

Example (not symmetric):

- $R_1 = \{(a,b) \mid a \leq b\}$ —$3 \leq 4$ 但 $4 \not\leq 3$
- $R_2 = \{(a,b) \mid a > b\}$ —$4 > 3$ 但 $3 \not> 4$
- $R_5 = \{(a,b) \mid a = b+1\}$ —$4 = 3+1$ 但 $3 \neq 4+1$

Antisymmetric (反对称的)

Definition: A relation R on a set A is **antisymmetric** iff for all $a, b \in A$, if $(a, b) \in R$ and $(b, a) \in R$, then $a = b$. Symbolically: $\forall x \forall y\ [((x, y) \in R \wedge (y, x) \in R) \longrightarrow (x = y)]$

区分 Symmetric 和 Antisymmetric：它们**不是**互斥的。- Symmetric: 有 (a, b) 必须有 (b, a) - Antisymmetric: 有 (a, b) 且 (b, a) 只能发生在 $a = b$ 时 - 一个关系可以既 symmetric 又 antisymmetric（如 $R = \{(a, a)\}$）- 一个关系可以既不是 symmetric 也不是 antisymmetric

Example (antisymmetric): - $R_1 = \{(a, b) \mid a \leq b\}$ —若 $a \leq b$ 且 $b \leq a$，则 $a = b$ - $R_2 = \{(a, b) \mid a > b\}$ - $R_4 = \{(a, b) \mid a = b\}$ - $R_5 = \{(a, b) \mid a = b + 1\}$

Example (not antisymmetric): - $R_3 = \{(a, b) \mid a = b \text{ or } a = -b\}$ —$(1, -1)$ 和 $(-1, 1)$ 都属于 R_3 - $R_6 = \{(a, b) \mid a + b \leq 3\}$ —$(1, 2)$ 和 $(2, 1)$ 都属于 R_6

Transitive (传递的)

Definition: R is **transitive** iff whenever $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$, for all $a, b, c \in A$. Symbolically: $\forall x \forall y \forall z\ [((x, y) \in R \wedge (y, z) \in R) \longrightarrow (x, z) \in R]$ **Example (transitive):**

- $R_1 = \{(a, b) \mid a \leq b\}$ —$a \leq b$ 且 $b \leq c \Rightarrow a \leq c$
- $R_2 = \{(a, b) \mid a > b\}$
- $R_4 = \{(a, b) \mid a = b\}$

Example (not transitive):

- $R_5 = \{(a, b) \mid a = b + 1\}$ —$(3, 2)$ 和 $(4, 3)$ 属于 R_5，但 $(3, 3) \notin R_5$
- $R_6 = \{(a, b) \mid a + b \leq 3\}$ —$(2, 1)$ 和 $(1, 2)$ 属于 R_6，但 $(2, 2) \notin R_6$

Important Question: Symmetric + Transitive => Reflexive?

陷阱题：如果一个关系是 symmetric 和 transitive 的，它一定是 reflexive 吗?

答案：不一定!

错误推理：

$$\left.\begin{array}{c}(a, b) \in R \\ R \text{ is symmetric} \\ \hline (b, a) \in R\end{array}\right\} \xrightarrow{\text{transitivity}} (a, a) \in R$$

问题：这个推理假设了存在某个 b 使得 $(a, b) \in R$。如果对于某个 a，不存在这样的 b，那么 (a, a) 不一定属于 R。

结论： Symmetry 和 transitivity 不足以保证 reflexivity。

8.1.3 3. Combining Relations (关系的组合)

Set Operations (集合运算)

Given two relations R_1 and R_2 on the same sets, we can combine them using basic set operations:

- $R_1 \cup R_2$ (并集)
- $R_1 \cap R_2$ (交集)
- $R_1 - R_2$ (差集)
- $R_2 - R_1$ (差集)

Composition (复合)

Definition: Suppose R_1 is a relation from A to B, and R_2 is a relation from B to C. Then the **composition** $R_2 \circ R_1$ is a relation from A to C where:

$$R_2 \circ R_1 = \{(x, z) \mid \exists y \in B : (x, y) \in R_1 \wedge (y, z) \in R_2\}$$

Example: Family Relations (家庭关系举例)

- Let $M =$ "is mother of" (是... 的母亲)
- Let $F =$ "is father of" (是... 的父亲)
- $M \circ F =$ "maternal grandfather" (外公): a 是 b 的父亲, b 是 c 的母亲
- $F \circ M =$ "paternal grandmother" (奶奶): a 是 b 的母亲, b 是 c 的父亲
- $M \circ M =$ "maternal grandmother" (外婆)
- 注意：M 和 F **都不是 transitive** 的关系!

易错提醒：复合运算的顺序很重要！$R_2 \circ R_1$ 的意思是先应用 R_1 再应用 R_2。

8.1.4 4. Powers of a Relation (关系的幂)

Definition: Let R be a binary relation on A. The powers R^n of R are defined inductively:

- **Basis Step:** $R^1 = R$
- **Inductive Step:** $R^{n+1} = R^n \circ R$

Example: $R = \{(1, 1), (2, 1), (3, 2), (4, 3)\}$

- $R^2 = R \circ R = \{(1, 1), (2, 1), (3, 1), (4, 2)\}$

- $R^3 = R^2 \circ R = \{(1, 1), (2, 1), (3, 1), (4, 1)\}$
- $R^4 = R^3 \circ R = \{(1, 1), (2, 1), (3, 1), (4, 1)\}$

Interpretation: Paths of Length n

$(x, y) \in R^n$ 当且仅当存在一条从 x 到 y 的长度为 n 的路径。

Theorem 1: Transitivity and Powers

Theorem: A relation R on a set A is **transitive** if and only if $R^n \subseteq R$ for $n = 1, 2, 3, \ldots$

Proof: - **If part (⇐):** If $R^2 \subseteq R$, and $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R^2 \subseteq R$, so R is transitive. - **Only if part (⇒):** If R is transitive and $(a, c) \in R^2$, then $\exists b : (a, b) \in R \wedge (b, c) \in R$, so $(a, c) \in R$. Thus $R^2 \subseteq R$. By induction, $R^n \subseteq R$ for all n.

8.1.5 5. Inverse Relation (逆关系)

Definition: Let R be a relation from A to B. The **inverse** of R, denoted R^{-1}, is a relation from B to A:

$$R^{-1} = \{(a, b) \mid (b, a) \in R\}$$

How to find R^{-1}:

1. **直接使用定义**：颠倒每个有序对
2. **从有向图**：反转所有边的方向
3. **从矩阵**：对连接矩阵 M_R 做转置 M_R^T

8.1.6 6. Properties of Relation Operations (关系运算的性质)

Let R, S be relations from A to B, T from B to C, P from C to D.

1. $(R \cup S)^{-1} = R^{-1} \cup S^{-1}$
2. $(R \cap S)^{-1} = R^{-1} \cap S^{-1}$
3. $(\overline{R})^{-1} = \overline{R^{-1}}$
4. $(R - S)^{-1} = R^{-1} - S^{-1}$
5. $(A \times B)^{-1} = B \times A$
6. $\overline{R} = A \times B - R$
7. $(S \circ T)^{-1} = T^{-1} \circ S^{-1}$
8. $(R \circ T) \circ P = R \circ (T \circ P)$ (复合的结合律)
9. $(R \cup S) \circ T = R \circ T \cup S \circ T$ (分配律)

8.1.7 7. 小结与考点提示 (Summary & Exam Tips)

核心考点

1. **四种性质的判断**—给定一个关系（定义、矩阵或有向图），判断它是否 reflexive / symmetric / antisymmetric / transitive
2. **对称与反对称的区别**—不是互斥的！可以同时成立，也可以都不成立
3. **Symmetric + Transitive ⇏ Reflexive** —经典陷阱题
4. **复合运算**—理解 $R_2 \circ R_1$ 的方向和含义
5. **关系的幂**—R^n 与路径长度的关系
6. **逆关系**—特别是矩阵转置和图的边反向

判断技巧速查表

性质	矩阵特征	有向图特征
Reflexive	主对角线全为 1	每个顶点都有 loop
Symmetric	M_R 是对称矩阵 ($m_{ij} = m_{ji}$)	每条边都有反向边
Antisymmetric	若 $m_{ij} = 1$ 且 $i \neq j$，则 $m_{ji} = 0$	没有双向边（loop 除外）
Transitive	M_R^2 中为 1 的位置在 M_R 中也必须为 1	若存在 $a \to b$ 和 $b \to c$，则必须有 $a \to c$

中文记忆口诀

- **Reflexive:** 每人都有自环
- **Symmetric:** 有来必有往
- **Antisymmetric:** 有来无往（除非是自己）
- **Transitive:** 两步能到，一步也能到

Notes prepared for DMA final exam.

8.2 Chapter 9.2: n-ary Relations (n 元关系)

8.2.1 1. Definition of n-ary Relations (n 元关系的定义)

What is an n-ary Relation?

A **binary relation** involves two sets. An **n-ary relation** (n 元关系) generalizes this to n sets. **Definition:** Let $A_1, A_2, \ldots, A_n$ be sets. An **n-ary relation** R on these sets is a subset of the Cartesian product:

$$R \subseteq A_1 \times A_2 \times \cdots \times A_n$$

- Each element of R is an **n-tuple** $(a_1, a_2, \ldots, a_n)$, where $a_i \in A_i$.
- Binary relation is a special case where $n = 2$.
- The **degree** (度) of the relation is n (the number of domains).
- The **number of entries** (基数) is $|R|$ (the number of tuples).

Example

Let $A_1 = \{1, 2\}$, $A_2 = \{a, b\}$, $A_3 = \{x, y\}$. Then

$$R = \{(1, a, x), (1, b, y), (2, a, x)\}$$

is a 3-ary (ternary) relation.

8.2.2 2. Relational Databases (关系数据库)

Basic Concepts

n-ary relations are the mathematical foundation of **relational databases** (关系数据库).

- **Table** (表) = an n-ary relation
- **Record / Tuple** (记录/元组) = an n-tuple
- **Field / Attribute** (字段/属性) = a column (one of the n positions)
- **Domain** (域) = the set of allowable values for a field
- **Primary Key** (主键) = a set of fields that uniquely identifies a tuple
- **Composite Key** (复合键) = a key consisting of multiple fields

Example: Student Database

Student ID	Name	Major	Year
001	Alice	CS	3
002	Bob	Math	2
003	Carol	CS	4

This is a 4-ary relation:

$$R \subseteq \text{ID} \times \text{Name} \times \text{Major} \times \text{Year}$$

represented as:

$$R = \{(001, \text{Alice}, \text{CS}, 3), (002, \text{Bob}, \text{Math}, 2), (003, \text{Carol}, \text{CS}, 4)\}$$

Key Concepts (关键概念)

- **Domain** (域): Each column has a domain (e.g., Year $\in \{1, 2, 3, 4\}$)
- **Primary Key** (主键): A field (or combination) whose values are unique across all tuples (e.g., Student ID)
- **Selection** (选择): Picking tuples satisfying a condition
- **Projection** (投影): Picking a subset of columns

8.2.3 3. Operations on n-ary Relations (n 元关系的运算)

Selection (选择)

Select tuples that satisfy a condition P:

$$\sigma_P(R) = \{(a_1, \ldots, a_n) \in R \mid P(a_1, \ldots, a_n) \text{ is true}\}$$

Example: Select all CS students from the database:

$$\sigma_{\text{Major = CS}}(R) = \{(001, \text{Alice}, \text{CS}, 3), (003, \text{Carol}, \text{CS}, 4)\}$$

Projection (投影)

Project onto a subset of the n components (columns):

$$\pi_{i_1, i_2, \ldots, i_m}(R) = \{(a_{i_1}, a_{i_2}, \ldots, a_{i_m}) \mid (a_1, \ldots, a_n) \in R\}$$

Example: Project onto Name and Major:

$$\pi_{\text{Name, Major}}(R) = \{(\text{Alice}, \text{CS}), (\text{Bob}, \text{Math}), (\text{Carol}, \text{CS})\}$$

Join (连接)

Combine two relations based on a common domain (field).

- **Natural Join** (自然连接 $\bowtie$): Combines tuples from two relations that have equal values on the common attributes.

Example:

- R_1: (StudentID, Name, Major)
- R_2: (StudentID, Course, Grade)
- $R_1 \bowtie R_2$: (StudentID, Name, Major, Course, Grade) —matched by StudentID.

Cartesian Product (笛卡尔积)

$$R_1 \times R_2 = \{(a_1, \ldots, a_n, b_1, \ldots, b_m) \mid (a_1, \ldots, a_n) \in R_1, (b_1, \ldots, b_m) \in R_2\}$$

8.2.4 4. n-ary Relations in the Course Context

Relationship to Binary Relations

在本课程中，重点研究的是 **binary relations**（二元关系）。n-ary relation 是更一般的形式，用于处理多个对象之间的关系。大部分重要的性质（reflexive, symmetric, transitive 等）都是在 binary relation 的框架下定义的。

Complexity Comparison

- Binary relation: 两个集合之间的关系，可用矩阵、有向图表示
- n-ary relation: n 个集合之间的关系，可用数据库表表示
- 课程中提到的"N-ary relationships (complex): relationships among many objects" 表明 n-ary relations 是多对象关系的数学形式化

8.2.5 5. 小结与考点提示 (Summary & Exam Tips)

核心考点

1. **n-ary relation 的定义**—是 $A_1 \times A_2 \times \cdots \times A_n$ 的子集
2. **与 binary relation 的关系**—binary relation 是 $n = 2$ 的特例
3. **数据库应用**—Selection, Projection, Join 等运算
4. **主键的概念**—唯一标识一个元组的字段组合

中文总结

- **n 元关系**是二元关系的推广，涉及 n 个集合
- 在关系数据库中，表就是 n-ary relation
- Selection 选行，Projection 选列，Join 合并表
- 本课程重点仍在 binary relation，n-ary relation 只需掌握基本概念

Notes prepared for DMA final exam.

8.3 Chapter 9.3: Representing Relations (关系的表示)

8.3.1 1. Representing Relations Using Matrices (用矩阵表示关系)

Zero-One Matrix (0-1 矩阵)

A relation between **finite sets** can be represented using a **zero-one matrix** (0-1 矩阵 / 布尔矩阵). Suppose R is a relation from $A = \{a_1, a_2, \ldots, a_m\}$ to $B = \{b_1, b_2, \ldots, b_n\}$. The relation R is represented by the matrix $M_R = [m_{ij}]$, where:

$$m_{ij} = \begin{cases} 1 & \text{if } (a_i, b_j) \in R \\ 0 & \text{if } (a_i, b_j) \notin R \end{cases}$$

- M_R is an $m \times n$ matrix
- Row i corresponds to a_i, column j corresponds to b_j
- When $A = B$, we use the same ordering for both sets (M_R is $n \times n$)

Example 1

$A = \{1, 2, 3\}$, $B = \{1, 2\}$, R contains (a, b) if $a > b$. $R = \{(2, 1), (3, 1), (3, 2)\}$, so:

$$M_R = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Example 2: Reading a Matrix

Let $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3, b_4, b_5\}$.

$$M_R = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$R = \{(a_1, b_2), (a_2, b_1), (a_2, b_3), (a_2, b_4), (a_3, b_1), (a_3, b_3), (a_3, b_5)\}$

8.3.2 2. Matrix Properties and Relation Properties (矩阵与关系性质)

When R is a relation on a set A (so M_R is square):

Property	Matrix Criterion
Reflexive	All diagonal entries $m_{ii} = 1$
Symmetric	M_R is symmetric: $m_{ij} = m_{ji}$ for all i, j
Antisymmetric	If $m_{ij} = 1$ and $i \neq j$, then $m_{ji} = 0$
Transitive	If M_R^2 has a 1 in position (i, j), then M_R must also have a 1 there

Example 3: Matrix Properties

Suppose R on a set of 3 elements is represented by:

$$M_R = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Reflexive?** Yes, all diagonal entries are 1.
- **Symmetric?** Yes, M_R is symmetric ($m_{12} = m_{21} = 1$, $m_{13} = m_{31} = 0$, $m_{23} = m_{32} = 0$).
- **Antisymmetric?** No, because $m_{12} = m_{21} = 1$ and $1 \neq 2$.

8.3.3 3. Representing Relations Using Digraphs (用有向图表示关系)

Directed Graph / Digraph (有向图)

Definition: A **directed graph** (有向图 / digraph) consists of:

- A set V of **vertices** (顶点 / nodes)
- A set E of **edges** (有向边 / arcs) —ordered pairs of elements of V
- The vertex a is the **initial vertex** (起点) of edge (a, b)
- The vertex b is the **terminal vertex** (终点) of edge (a, b)
- An edge of the form (a, a) is called a **loop** (自环)

关键理解：一个集合上的二元关系等价于一个有向图。每个有序对 (a, b) 对应一条从 a 到 b 的边。

Example 4: Reading a Digraph

Vertices: a, b, c, d Edges: $(a, b), (a, d), (b, b), (b, d), (c, a), (c, b), (d, b)$ This can be drawn as a diagram with arrows between labeled vertices.

Example 5: Reading a Digraph Back to Pairs

Given a digraph with vertices $\{1, 2, 3, 4\}$ and edges as drawn, determine the ordered pairs. **Solution:** The ordered pairs are:

$$(1, 3), (1, 4), (2, 1), (2, 2), (2, 3), (3, 1), (3, 3), (4, 1), (4, 3)$$

8.3.4 4. Properties from Digraphs (从有向图判断性质)

Property	Digraph Criterion
Reflexive	Every vertex has a **loop**
Symmetric	For every edge (x, y), the reverse edge (y, x) also exists
Antisymmetric	If (x, y) is an edge with $x \neq y$, then (y, x) is **not** an edge
Transitive	If edges (x, y) and (y, z) exist, then (x, z) must also exist

Example 6: Trivial Cases (平凡情况)

Consider a digraph with 4 vertices and **no edges at all** (只有顶点，没有边):

Property	Answer	Reason
Reflexive?	**No**	No vertex has a loop
Symmetric?	**Yes** (vacuously)	没有边，所以条件”若有边则应有反向边”自动成立
Antisymmetric?	**Yes** (vacuously)	没有边，所以条件自动成立
Transitive?	**Yes** (vacuously)	没有边，所以条件自动成立

关键词：**vacuously true (空洞真)** —当条件的前提永远不成立时，结论自动为真。

Example 7: Partial Digraph Analysis

Scenario	Reflexive	Symmetric	Antisymmetric	Transitive
No loops, edge $a \to b$	No	No (no $b \to a$)	No ($d \to b$ and $b \to d$)	No (missing $a \to d$)
No loops, some edges	No	No	Yes	No
Some edges, no loops	No	No	Yes	Yes (trivially, no two consecutive edges)

8.3.5 5. Powers of a Relation in Digraphs (用图理解关系的幂)

The pair (x, y) is in R^n if and only if there is a **path of length** n from x to y in the digraph of R.

- R^2: paths of exactly 2 edges
- R^3: paths of exactly 3 edges
- etc.

Example

Given a digraph of R, we find:

- R: original edges
- R^2: all pairs reachable via a path of length 2
- R^3: all pairs reachable via a path of length 3

This is visually intuitive: trace all paths of the given length following arrow directions.

8.3.6 6. Inverse Relation and Matrix (逆关系与矩阵)

For a relation R with matrix M_R:

- The **inverse relation** R^{-1} has matrix M_R^T (the **transpose**, 转置矩阵)
- Row and column indices are swapped

In digraph terms: R^{-1} is obtained by reversing all arrows in the digraph of R.

8.3.7 7. 小结与考点提示 (Summary & Exam Tips)

核心考点

1. **矩阵表示**—给定关系和集合，写出矩阵；给定矩阵，读出关系
2. **矩阵判断性质**—通过矩阵对角线、对称性判断 reflexive / symmetric / antisymmetric
3. **有向图表示**—给定关系画图，或看图写出关系
4. **从有向图判断性质**—最重要！看好 loops, 双向边, 传递性
5. **平凡情况 (vacuously true)** —没有边时的性质判断是经典考点
6. **矩阵转置与逆关系**

快速判断表

性质	矩阵	有向图
Reflexive	对角线全 1	所有顶点有 loop
Symmetric	矩阵对称	每条边有反向边
Antisymmetric	$m_{ij} = 1 \wedge i \neq j \Rightarrow m_{ji} = 0$	无双向边（loop 除外）
Transitive	$M_R^2 \leq M_R$ (元素级)	路径 $a \to b \to c \Rightarrow a \to c$

中文记忆

- **Reflexive:** 矩阵对角全是 1，图里每人有自环
- **Symmetric:** 矩阵对称，图里箭头成对
- **Antisymmetric:** 上三角或下三角全 0（非对角），图里没有双向箭头
- **Transitive:** 有中转就能直达

Notes prepared for DMA final exam.

8.4 Chapter 9.4: Closures of Relations (关系的闭包)

8.4.1 1. Concept of Closure (闭包的概念)

Definition: The **closure** (闭包) of a relation R with respect to property P is the relation obtained by adding the **minimum number** of ordered pairs to R to obtain property P.

通俗理解：给定一个关系 R，它可能不具备某种性质（如 reflexive）。我们通过添加**最少的**有序对，使它变成具有该性质的关系。这个新关系就是 R 的闭包。

8.4.2 2. Reflexive Closure (自反闭包)

Definition

The **reflexive closure** of R, denoted $r(R)$, makes R reflexive by adding the minimum number of pairs.

$$r(R) = R \cup \Delta$$

where $\Delta = \{(a, a) \mid a \in A\}$ (对角线关系 / diagonal relation)

In Matrix Terms

Put 1's on the diagonal:

$$M_{r(R)} = M_R \vee I_n$$

where I_n is the $n \times n$ identity matrix. **Example:**

$$M_R = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad \Rightarrow \quad M_{r(R)} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

In Digraph Terms

Add **loops** (自环) to all vertices that don't already have one.

8.4.3 3. Symmetric Closure (对称闭包)

Definition

The **symmetric closure** of R, denoted $s(R)$, makes R symmetric by adding the minimum number of pairs.

$$s(R) = R \cup R^{-1}$$

In Matrix Terms

Add 1's to make the matrix symmetric (reflect across the diagonal):

$$M_{s(R)} = M_R \vee M_R^T$$

In Digraph Terms

For every existing edge (a, b), add the reverse edge (b, a) if it isn't already there. **Example:**

$$M_R = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad \Rightarrow \quad M_{s(R)} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

8.4.4 4. Transitive Closure (传递闭包)

Definition

The **transitive closure** of R, denoted $t(R)$, makes R transitive by adding the minimum number of pairs.

Challenge

Finding the transitive closure is **harder** than reflexive or symmetric closures.

- Reflexive closure: just add loops
- Symmetric closure: just add reverse edges
- Transitive closure: must add all pairs reachable via paths of any length

Connectivity Relation (连通关系)

Definition: The **connectivity relation** R^* consists of all pairs (a, b) such that there is a **path** (路径) from a to b in the digraph of R:

$$R^* = \bigcup_{n=1}^{\infty} R^n$$

Theorem 2: The transitive closure $t(R) = R^* =$ the connectivity relation.

Paths (路径)

Definition: A **path** from a to b in a digraph G is a sequence of one or more edges:

$$(x_0, x_1), (x_1, x_2), \ldots, (x_{n-1}, x_n)$$

where $x_0 = a$ and $x_n = b$.

- The **length** of the path is n.
- If $a = b$, the path is called a **circuit** (回路) or **cycle** (圈).

Theorem 1: There is a path of length n from a to b iff $(a, b) \in R^n$.

Proof by induction: - Basis ($n = 1$): An edge is a path of length 1, which belongs to $R^1 = R$. **- Inductive step:** A path of length $n+1$ exists iff there exists x with a path of length 1 from a to x and a path of length n from x to b. By induction hypothesis, $(a, x) \in R$ and $(x, b) \in R^n$, so $(a, b) \in R^n \circ R = R^{n+1}$.

Lemma 1 (Bounded Path Length)

If A has n elements and R is a relation on A, then if there is a path from a to b, there is a path of length **not exceeding** n. If $a \neq b$, the length does not exceed $n - 1$.

This gives us a practical bound:

$$t(R) = \bigcup_{i=1}^{n} R^i$$

8.4.5 5. Matrix Method for Transitive Closure

Theorem 3: The matrix of R^* is:

$$M_{R^*} = M_R \vee M_R^2 \vee M_R^3 \vee \cdots \vee M_R^n$$

Algorithm 1 (Basic Algorithm)

```
procedure transitive_closure(M_R: zero-one n×n matrix)
    A := M_R
    B := A
    for i := 2 to n
        A := A   M_R        // Boolean matrix multiplication
        B := B   A           // OR operation
    end
    // B is the zero-one matrix for R*
end
```

Complexity: $O(n^4)$ bit operations.

8.4.6 6. Warshall's Algorithm (沃舍尔算法)

Warshall's algorithm is a much more efficient method —$O(n^3)$ bit operations.

Key Idea

Let the vertices be $v_1, v_2, \ldots, v_n$. Define W_k as the matrix where $W_k[i][j] = 1$ iff there is a path from v_i to v_j whose **interior vertices** (内部顶点) come only from $\{v_1, v_2, \ldots, v_k\}$.

- $W_0 = M_R$ (no interior vertices allowed)
- $W_n = M_{R^*}$ (all vertices allowed as interior)

Recurrence (递推公式)

$$w_{ij}^{(k)} = w_{ij}^{(k-1)} \vee \left(w_{ik}^{(k-1)} \wedge w_{kj}^{(k-1)} \right)$$

Meaning: There is a path using vertices $\{v_1, \ldots, v_k\}$ as interiors if either:

- (a) There is already such a path using only $\{v_1, \ldots, v_{k-1}\}$, OR
- (b) There is a path $v_i \to v_k$ and a path $v_k \to v_j$, both using only $\{v_1, \ldots, v_{k-1}\}$

Algorithm 2 (Warshall)

```
procedure warshall(M_R: n×n zero-one matrix)
    W := M_R
    for k := 1 to n
        for i := 1 to n
            for j := 1 to n
                W[i][j] := W[i][j]   (W[i][k]   W[k][j])
    end
    // W is the zero-one matrix for R*
end
```

Complexity: $O(n^3)$ —three nested loops.

Optimization Trick

在 Warshall 算法中，如果 $W[i][k] == 1$，则第 i 行的第 j 列等于原来的 $W[i][j]$ 或 $W[k][j]$：

$$\text{If } W[i][k] = 1, \text{ then } W[i][j] := W[i][j] \vee W[k][j] \text{ for all } j$$

Example: Warshall's Algorithm Step by Step

$A = \{1, 2, 3, 4, 5\}$, $R = \{(1, 1), (1, 2), (2, 4), (3, 5), (4, 2)\}$ **Initial** M_R **($k = 0$):**

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$k = 1$ (add vertex 1 as interior):

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$k = 2$ (add vertex 2): Discover paths through vertex 2.

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Explanation: $W[1][4]$ becomes 1 because $W[1][2] = 1$ and $W[2][4] = 1$. Similarly $W[4][4] = 1$ because $W[4][2] = 1$ and $W[2][4] = 1$. $k = 3$ (add vertex 3): No significant change (vertex 3 only connects to 5). $k = 4$ (add vertex 4):

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Explanation: $W[2][2] = 1$ because $W[2][4] = 1$ and $W[4][2] = 1$. This creates a cycle $2 \to 4 \to 2$. $k = 5$ (add vertex 5): No significant change. **Final $t(R)$:**

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

As ordered pairs: $t(R) = \{(1,1), (1,2), (1,4), (2,2), (2,4), (3,5), (4,2), (4,4)\}$

8.4.7 7. Properties of Closure Operations (闭包运算的性质)

For relations R, S on a set A:

- $r(R) = R \cup \Delta$ (reflexive closure)
- $s(R) = R \cup R^{-1}$ (symmetric closure)
- $t(R) = R^* = \bigcup_{i=1}^{\infty} R^i = \bigcup_{i=1}^{n} R^i$ (transitive closure, for finite A with $|A| = n$)

8.4.8 8. 小结与考点提示 (Summary & Exam Tips)

核心考点

1. **三种闭包的定义和求法**—加自环、加反向边、加传递路径
2. **传递闭包与连通关系**—$t(R) = R^* = \bigcup R^n$
3. **Warshall 算法**—三步循环，$O(n^3)$，必须掌握手工计算过程
4. **有限集合上的路径长度不超过** n —Lemma 1 是关键理论

中文口诀

- **Reflexive 闭包:** 缺环补环，对角变 1
- **Symmetric 闭包:** 缺对补对，转置加法
- **Transitive 闭包:** 路径可达，传递闭包

算法对比

算法	思路	复杂度
基本算法 (Algorithm 1)	$M_R \vee M_R^2 \vee \cdots \vee M_R^n$	$O(n^4)$
Warshall (Algorithm 2)	动态规划，逐个加入中间顶点	$O(n^3)$

Warshall 算法手动计算步骤

1. 从 $W_0 = M_R$ 开始
2. 对每个 $k = 1, 2, \ldots, n$：检查第 k 列中为 1 的行 i，将第 i 行与第 k 行 OR 后赋给第 i 行（即 W[i][j] = W[i][j] OR W[k][j]）
3. W_n 就是传递闭包的矩阵

Notes prepared for DMA final exam.

8.5 Chapter 9.5: Equivalence Relations (等价关系)

8.5.1 1. Definition of Equivalence Relation (等价关系的定义)

Definition 1: A relation on a set A is called an **equivalence relation** (等价关系) if it is:

1. **Reflexive** (自反的)
2. **Symmetric** (对称的)
3. **Transitive** (传递的)

Definition 2: Two elements a and b that are related by an equivalence relation are called **equivalent** (等价的). We write $a \sim b$ to denote that a and b are equivalent.

核心理解：等价关系将集合中的元素按照某种标准" 等同" 起来。满足自反、对称、传递这三个条件的关系才能做到这一点。

8.5.2 2. Examples of Equivalence Relations

Example 1: Strings of Equal Length (等长的字符串)

R is the relation on strings of English letters where aRb iff $l(a) = l(b)$ （长度相同）.

- **Reflexive:** $l(a) = l(a)$ for all strings a
- **Symmetric:** If $l(a) = l(b)$, then $l(b) = l(a)$
- **Transitive:** If $l(a) = l(b)$ and $l(b) = l(c)$, then $l(a) = l(c)$

Therefore, R is an equivalence relation.

Example 2: Congruence Modulo m (模 m 同余)

Let $m > 1$ be an integer. Define:

$$R = \{(a, b) \mid a \equiv b \pmod{m}, a, b \in \mathbb{Z}\}$$

where $a \equiv b \pmod{m}$ means $m \mid (a - b)$.

- **Reflexive:** $a - a = 0 = 0 \cdot m$, so $a \equiv a \pmod{m}$
- **Symmetric:** If $a \equiv b \pmod{m}$, then $a - b = km$, so $b - a = (-k)m$, hence $b \equiv a \pmod{m}$
- **Transitive:** If $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$, then:
- $a - b = km$ and $b - c = lm$
- $(a - b) + (b - c) = a - c = (k + l)m$
- So $a \equiv c \pmod{m}$

Therefore, R is an equivalence relation.

Example 3: "Divides" Relation (整除关系)

The relation $a \mid b$ on positive integers:

- **Reflexive:** $a \mid a$ for all a
- **Symmetric: NO** —$2 \mid 4$ but $4 \nmid 2$
- **Transitive:** $a \mid b$ and $b \mid c \Rightarrow a \mid c$

结论："divides" 不是等价关系，因为不满足 symmetric。

8.5.3 3. Equivalence Classes (等价类)

Definition

Let R be an equivalence relation on a set A. The **equivalence class** (等价类) of an element $a \in A$ is the set of all elements related to a:

$$[a]_R = \{s \mid (a, s) \in R\}$$

or simply $[a]$ when the relation is clear from context.

- $b \in [a]_R$ is called a **representative** (代表元) of this equivalence class.
- Any element of a class can serve as its representative.

Example: Congruence Classes (同余类)

For congruence modulo m, the equivalence class of a is:

$$[a]_m = \{\ldots, a - 2m, a - m, a, a + m, a + 2m, \ldots\}$$

For $m = 4$:

- $[0]_4 = \{\ldots, -8, -4, 0, 4, 8, \ldots\}$
- $[1]_4 = \{\ldots, -7, -3, 1, 5, 9, \ldots\}$
- $[2]_4 = \{\ldots, -6, -2, 2, 6, 10, \ldots\}$
- $[3]_4 = \{\ldots, -5, -1, 3, 7, 11, \ldots\}$

8.5.4 4. Equivalence Classes and Partitions (等价类与划分)

Theorem 1

Let R be an equivalence relation on A. For $a, b \in A$, the following statements are **equivalent**: 1. aRb 2. $[a] = [b]$ 3. $[a] \cap [b] \neq \emptyset$

Proof:

(1) ⇒ (2): Assume aRb. - To show $[a] \subseteq [b]$: Take $c \in [a]$, then aRc. Since aRb and R is symmetric, bRa. By transitivity, bRa and aRc gives bRc, so $c \in [b]$. - To show $[b] \subseteq [a]$: Similar argument. - Therefore $[a] = [b]$.

(2) ⇒ (3): Since R is reflexive, $a \in [a]$. If $[a] = [b]$, then $a \in [b]$, so $[a] \cap [b] \neq \emptyset$ (至少包含 a).

(3) ⇒ (1): Assume $[a] \cap [b] \neq \emptyset$. Then $\exists x$ such that $x \in [a]$ and $x \in [b]$. So $(a, x) \in R$ and $(b, x) \in R$. By symmetry, $(x, b) \in R$. By transitivity, $(a, x) \in R$ and $(x, b) \in R$ gives $(a, b) \in R$.

Partition of a Set (集合的划分)

Definition: A **partition** (划分) of a set S is a collection of **disjoint** (互不相交) **nonempty** subsets of S whose **union** is S. Formally, a collection $\{A_i \mid i \in I\}$ forms a partition of S iff:

1. $A_i \neq \emptyset$ for all $i \in I$ （非空）
2. $A_i \cap A_j = \emptyset$ when $i \neq j$ （两两不交）
3. $\bigcup_{i \in I} A_i = S$ （并集为整个集合）

Notation: $\mathrm{pr}(A) = \{A_i \mid i \in I\}$

Theorem 2: The Fundamental Theorem of Equivalence Relations

Let R be an equivalence relation on S. Then the equivalence classes of R form a **partition** of S.

Conversely, given a partition $\{A_i \mid i \in I\}$ of S, there is an **equivalence relation** R whose equivalence classes are exactly the A_i's.

Proof (second part): Define R as: $(x, y) \in R$ iff x and y belong to the same subset A_i in the partition. - **Reflexive:** Every a is in the same subset as itself. - **Symmetric:** If a and b are in the same subset, then b and a are in the same subset. - **Transitive:** If a, b are in the same subset and b, c are in the same subset, since the subsets are disjoint and b belongs to both, the two subsets must be identical. So a and c are in the same subset.

核心结论：等价关系 $\longleftrightarrow$ 划分（一一对应关系!）

8.5.5 5. Example: Finding Partition from an Equivalence Relation

$A = \{0, 1, 2, 3, 4, 5\}$

$$R = \{(0,0), (1,1), (2,2), (3,3), (1,2), (1,3), (2,1), (2,3), (3,1), (3,2), (4,4), (4,5), (5,4), (5,5)\}$$

Find the equivalence classes:

- $[0] = \{0\}$ (只有 0 和自己相关)
- $[1] = \{1, 2, 3\}$ (1,2,3 相互相关)
- $[4] = \{4, 5\}$ (4 和 5 相关)

The partition:

$$\mathrm{pr}(A) = \{[0], [1], [4]\} = \{\{0\}, \{1, 2, 3\}, \{4, 5\}\}$$

Example: Partition from Congruence Modulo m

$$R = \{(a, b) \mid a \equiv b \pmod{m}, a, b \in \mathbb{Z}\}$$

The equivalence classes are:

$$\mathrm{pr}(\mathbb{Z}) = \{[0]_m, [1]_m, \ldots, [m-1]_m\}$$

There are exactly m distinct equivalence classes, each containing integers with the same remainder when divided by m.

8.5.6 6. Counting Equivalence Relations (计数等价关系)

Question: If $|A| = 3$, how many different equivalence relations are there on A?

Solution: An equivalence relation on $A \Longleftrightarrow$ a partition of A.

The number of partitions of a 3-element set = **Bell number** $B_3 = 5$.

For $A = \{a, b, c\}$, the 5 partitions are: 1. $\{\{a\}, \{b\}, \{c\}\}$ —3 individual classes 2. $\{\{a, b\}, \{c\}\}$ 3. $\{\{a, c\}, \{b\}\}$ 4. $\{\{a\}, \{b, c\}\}$ 5. $\{\{a, b, c\}\}$ —1 class containing all

Each partition corresponds to exactly one equivalence relation, so there are **5** equivalence relations.

8.5.7 7. Combining Equivalence Relations (等价关系的组合)

Intersection (交集)

Theorem: If R and S are equivalence relations on A, then $R \cap S$ is also an equivalence relation.

- **Reflexive:** $(a, a) \in R$ and $(a, a) \in S$, so $(a, a) \in R \cap S$ for all a. - **Symmetric:** $R^{-1} = R$ and $S^{-1} = S$, so $(R \cap S)^{-1} = R^{-1} \cap S^{-1} = R \cap S$. - **Transitive:** $R^2 \subseteq R$ and $S^2 \subseteq S$, and $(R \cap S)^2 \subseteq R^2 \cap S^2 \subseteq R \cap S$.

Union (并集)

Theorem: If R and S are equivalence relations on A, $R \cup S$ is **NOT** necessarily an equivalence relation.

Counterexample: $A = \{a, b, c\}$ - $R_1 = \{(a, a), (b, b), (c, c), (a, b), (b, a)\}$ - $R_2 = \{(a, a), (b, b), (c, c), (b, c), (c, b)\}$ - $R_1 \cup R_2 = \{(a, a), (b, b), (c, c), (a, b), (b, a), (b, c), (c, b)\}$ - This is reflexive and symmetric, but **not transitive**: $(a, b) \in R$ and $(b, c) \in R$, but $(a, c) \notin R$.

Theorem on Union Closure

Theorem: If R_1, R_2 are equivalence relations on A, then: - $R_1 \cup R_2$ is reflexive and symmetric (但不一定 transitive) - $(R_1 \cup R_2)^*$ (transitive closure of the union) is an equivalence relation

8.5.8 8. 小结与考点提示 (Summary & Exam Tips)

核心考点

1. **判断等价关系**—检查 reflexive + symmetric + transitive 三个条件
2. **等价类的计算**—能找出给定等价关系的所有等价类
3. **等价关系 $\longleftrightarrow$ 划分**—最重要的对应关系！给定一个就能求另一个
4. **代表元**—等价类的任一元素都可以做代表
5. **同余关系**—模 m 同余是最经典的等价关系，有 m 个等价类
6. **等价关系的交集还是等价关系，并集不一定**

Exam Tips

- 判断一个关系是否为等价关系时，三个条件**缺一不可**
- "Divides" 关系的反例：不满足 symmetric
- 等价类的并集 = 整个集合，等价类两两不相交
- 一个 n 元集合上的等价关系数 = 第 n 个 **Bell number**

常见陷阱

陷阱	说明
认为"divides" 是等价关系	错！它不满足 symmetric
认为等价类的代表元必须唯一	错！任意元素都可以
认为 $R \cup S$ 是等价关系	错！可能不满足 transitive

中文总结

- **等价关系** = 自反 + 对称 + 传递
- **等价类** = 相互等价的元素构成的集合
- **划分** = 将集合拆成不相交的非空子集，子集之并等于原集合
- **等价关系与划分一一对应**：给定等价关系，等价类是划分；给定划分，可定义等价关系

Notes prepared for DMA final exam.

8.6 Chapter 9.6: Partial Orderings (偏序)

8.6.1 1. Definition of Partial Order (偏序的定义)

Definition 1: A relation R on a set S is called a **partial ordering** (偏序) or **partial order** if it is:

1. **Reflexive** (自反的)
2. **Antisymmetric** (反对称的)
3. **Transitive** (传递的)

Definition: A set S together with a partial ordering R is called a **partially ordered set** (偏序集), or **poset**, denoted by (S, R).

注意区分：等价关系需要 symmetric（对称），偏序关系需要 antisymmetric（反对称）。这是两者的关键区别。

8.6.2 2. Examples of Partial Orders

Example 1: "Greater than or equal" ($\geq$)

$(\mathbb{Z}, \geq)$ is a poset.

- **Reflexive:** $a \geq a$ for all a
- **Antisymmetric:** If $a \geq b$ and $b \geq a$, then $a = b$
- **Transitive:** If $a \geq b$ and $b \geq c$, then $a \geq c$

Example 2: Divisibility ($|$)

$(\mathbb{Z}^+, |)$ is a poset.

- **Reflexive:** $a \mid a$ for all positive integers
- **Antisymmetric:** If $a \mid b$ and $b \mid a$, then $a = b$
- **Transitive:** If $a \mid b$ and $b \mid c$, then $a \mid c$

Example 3: Set Inclusion ($\subseteq$)

$(\mathcal{P}(S), \subseteq)$ is a poset, where $\mathcal{P}(S)$ is the power set of S.

- **Reflexive:** $A \subseteq A$ for all subsets A
- **Antisymmetric:** If $A \subseteq B$ and $B \subseteq A$, then $A = B$
- **Transitive:** If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$

8.6.3 3. Comparability (可比性)

Definitions

In a poset $(S, \preccurlyeq)$:

- **Comparable** (可比的): Elements a and b are **comparable** if $a \preccurlyeq b$ or $b \preccurlyeq a$.
- **Incomparable** (不可比的): Neither $a \preccurlyeq b$ nor $b \preccurlyeq a$.
- **Total order** (全序) / **Linear order** (线序): A poset where **every** two elements are comparable. Also called a **chain** (链).
- **Well-ordered set** (良序集): A total order where **every nonempty subset** has a **least element**.

Examples

- $(\mathbb{Z}, \leq)$ is a **total order** —any two integers are comparable.
- $(\mathbb{Z}^+, |)$ is **NOT** a total order —e.g., 2 and 3 are incomparable (2 does not divide 3, 3 does not divide 2).

8.6.4 4. Lexicographic Order (字典序)

Definition: Given two posets $(A_1, \preccurlyeq_1)$ and $(A_2, \preccurlyeq_2)$, the **lexicographic ordering** (字典序) on $A_1 \times A_2$ is defined by:

$$(a_1, a_2) \prec (b_1, b_2)$$

if either:

1. $a_1 \prec_1 b_1$, OR
2. $a_1 = b_1$ and $a_2 \prec_2 b_2$

理解：类似字典中的排序。先比较第一个分量，如果相等再比较第二个分量。

Example

Let $A_1 = A_2 = \mathbb{Z}^+$ and $R_1 = R_2 = |$ (divisibility):

- $(2, 4) < (2, 8)$ because $x_1 = x_2$ (both 2) and $4 \mid 8$
- $(2, 4)$ is **not related** to $(2, 6)$ because $x_1 = x_2$ but 4 does not divide 6
- $(2, 4) < (4, 5)$ because $2 \mid 4$

Lexicographic Order on Strings

The same idea extends to strings of arbitrary length:

- "discreet" $\prec$ "discrete" —第七个字母不同，e $\prec$ t
- "discreet" $\prec$ "discreetness" —前八个字母相同，但第二个字符串更长

8.6.5 5. Hasse Diagrams (哈斯图)

Purpose

A **Hasse diagram** (哈斯图) is a simplified visual representation of a poset that omits:

1. **Loops** (自环) —implied by reflexivity
2. **Transitive edges** (传递边) —implied by transitivity
3. **Arrow heads** —edges always point **upward**

Construction Procedure

1. Start with the directed graph of the relation
2. Remove all loops (reflexive)
3. Remove edges that are implied by transitivity (if $x \prec z \prec y$, remove $x \to y$)
4. Arrange vertices so edges go **upward**, remove arrow heads

Example: $(\{1, 2, 3, 4, 6, 8, 12\}, |)$

```
  8       12
 / \      / \
4       6
|      / |
2     3
 \   /
  1
```

逐步构建：

1. 原始关系包含所有整除对（如 1|2, 1|3, 1|4, 2|4, 2|8, 4|8 等）
2. 去掉自环（1|1, 2|2 等）
3. 去掉传递边（如 1|8 可以通过 1|2|4|8 传递得到，所以去掉 1→8 的直接边；保留 1→2, 2→4, 4→8）
4. 将所有边方向朝上

Example: $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $R = \{(a, b) \mid a \mid b\}$
After removing loops and transitive edges:

```
  8       9
 / \
4   6
| / |
2   3    5    7
 \ |    |    |
  1
```


8.6.6 6. Hasse Diagram Terminology (哈斯图术语)

Extremal Elements (极值元素)

Term	Definition	Chinese
Maximal (极大元)	No $b \in S$ such that $a \prec b$	在图中是" 顶部" 元素，没有元素在它上面
Minimal (极小元)	No $b \in S$ such that $b \prec a$	在图中是" 底部" 元素，没有元素在它下面
Greatest (最大元)	$b \preccurlyeq a$ for all $b \in S$	唯一，大于或等于所有元素
Least (最小元)	$a \preccurlyeq b$ for all $b \in S$	唯一，小于或等于所有元素

Examples

Example Set 1: $(\{1, 2, 3, 4, 6, 8, 12, 24, 36\}, |)$

```
24   36
 |    |
 12
/  \
4    6
|   / \
2  3
 \ /
  1
```

- **Maximal:** 24, 36
- **Minimal:** 2, 3
- **Greatest:** / (不存在，24 和 36 无法比较)
- **Least:** / (不存在，2 和 3 无法比较)

Example Set 2: $(\mathcal{P}(\{a, b, c\}), \subseteq)$

```
      {a,b,c}
     /   |   \
{a,b} {a,c} {b,c}
  |   X   X   |
{a}   {b}   {c}
  \    |    /
```

- **Greatest:** $\{a, b, c\}$ (全集)
- **Least:** $\emptyset$ (空集)

Theorem: Uniqueness of Greatest/Least Element

The greatest and least elements of a poset $(A, \preccurlyeq)$ are **unique** when they exist.

Proof: Suppose a_1 and a_2 are both greatest elements. Then $a_1 \preccurlyeq a_2$ (since a_2 is greatest) and $a_2 \preccurlyeq a_1$ (since a_1 is greatest). By antisymmetry, $a_1 = a_2$.

8.6.7 7. Upper and Lower Bounds (上下界)

Let A be a subset of a poset $(S, \preccurlyeq)$.

Term	Definition	Chinese
Upper bound (上界)	u such that $a \preccurlyeq u$ for all $a \in A$	大于等于 A 中所有元素的元素
Lower bound (下界)	l such that $l \preccurlyeq a$ for all $a \in A$	小于等于 A 中所有元素的元素
Least upper bound (lub, 最小上界)	The **smallest** upper bound	所有上界中最小的
Greatest lower bound (glb, 最大下界)	The **largest** lower bound	所有下界中最大的

Example

```
      h       j
    /   \   /
   g     f
  / \   / \
 d   e
/ \ / \
a   b   c
```

- **Maximal elements:** h, j
- **Minimal elements:** a
- **Greatest element:** none
- **Least element:** a
- **Upper bound of** $\{a, b, c\}$**:** $\{e, f, j, h\}$
- **Least upper bound of** $\{a, b, c\}$**:** e
- **Lower bound of** $\{a, b, c\}$**:** $\{a\}$
- **Greatest lower bound of** $\{a, b, c\}$**:** a

Another Example

$A = \{2, 3, 6, 12, 24, 36\}$, $B_1 = \{2, 3, 6\}$, $B_2 = \{12, 24, 36\}$, relation: $|$

```
24   36
  \  /
   12
   |
   6
  / \
 2   3
```

For $B_1 = \{2, 3, 6\}$**:**

- Upper bounds: $\{6, 12, 24, 36\}$
- Lower bounds: / (none, 2 和 3 没有公共下界)
- lub: 6
- glb: / (none)

For $B_2 = \{12, 24, 36\}$**:**

- Upper bounds: / (none, 24 和 36 没有公共上界)
- Lower bounds: $\{12, 6, 2, 3\}$
- lub: /
- glb: 12

8.6.8 8. Lattices (格)

Definition: A poset in which **every pair of elements** has both a **least upper bound** (lub) and a **greatest lower bound** (glb) is called a **lattice** (格).

Examples

1. $(\mathbb{Z}, \leq)$: lub = max(a,b), glb = min(a,b) —**是 lattice**
2. $(\mathbb{Z}^+, |)$: lub = lcm(a,b), glb = gcd(a,b) —**是 lattice**
3. $(\mathcal{P}(S), \subseteq)$: lub = $A \cup B$, glb = $A \cap B$ —**是 lattice**

Note: Every totally ordered set is a lattice (因为全序集中任意两个元素都可比，lub 和 glb 就是两者中的较大者和较小者).

8.6.9 9. Topological Sorting (拓扑排序)

Motivation

Given a partial order (e.g., task dependencies), find a **total order** that is **compatible** with the partial order:

A total ordering $\preccurlyeq$ is compatible with the partial ordering R if $a \preccurlyeq b$ whenever aRb.

Lemma 1

Every finite nonempty poset $(S, \preccurlyeq)$ has at least one **minimal element**.

Algorithm: Topological Sorting

```
procedure topological_sort(S, : finite poset)
    k := 1
    while S
        a_k := a minimal element of S
        S := S - {a_k}
        k := k + 1
    end
end
```

Steps:

1. 在当前 poset 中找一个 **minimal element**（极小元）
2. 把它从集合中移除，放在排序结果中
3. 重复直到所有元素都被移除

Example

Poset $(\{1, 2, 4, 5, 12, 20\}, |)$:

```
  12     20
 /  \   /
4    5
|    |
2
|
1
```

One possible topological sort (有多个可能): $1, 2, 4, 5, 12, 20$ or $1, 2, 5, 4, 20, 12$ 只要保证：如果 $a \mid b$，那么 a 在 b **之前**出现即可。

8.6.10 10. 小结与考点提示 (Summary & Exam Tips)

核心考点

1. **偏序的三条件**—Reflexive + Antisymmetric + Transitive（与等价关系对比）
2. **Hasse 图的绘制和分析**—去掉自环、传递边、箭头向上
3. **极大/极小元 vs 最大/最小元**—前者不唯一，后者唯一
4. **上下界、lub、glb 的计算**
5. **判断是否为 lattice** —每对元素是否有 lub 和 glb
6. **拓扑排序**—每次取 minimal element

易混淆概念对比

概念	偏序 (Partial Order)	等价关系 (Equivalence)
核心条件	Reflexive + Antisymmetric + Transitive	Reflexive + Symmetric + Transitive
目的	排序/比较	归类/等同
典型例子	$\leq, \mid, \subseteq$	$= \bmod m$, 等长字符串

极大元 vs 最大元

- **极大元** (Maximal): 没有元素比它大（可以有多个）
- **最大元** (Greatest): 它比所有元素都大（唯一，如果存在）

Hasse 图判断技巧

- **极大元** = 图中最顶层（没有边从它往上指）
- **极小元** = 图中最底层（没有边从它往下指）
- **最大元** = 所有元素都能走到它（唯一顶层）
- **最小元** = 它能走到所有元素（唯一底层）

中文口诀

- **偏序**：自反对称传递一打住！是**反对称**不是对称！
- **Hasse 图**：去自环，去传递，箭头朝上全不见
- **拓扑排序**：每次找个极小元，拿走放在结果前

Notes prepared for DMA final exam.

第九章　Chapter 10: Graphs (图论)

9.1　Chapter 10.1: Graphs and Graph Models (图与图模型)

9.1.1　1. 什么是图? (What is a Graph?)

【**Definition 1**】 A **graph** $G = (V, E)$ consists of:

- V: a nonempty set of **vertices** (or **nodes**) —顶点
- E: a set of **edges** —边
- Each edge has either one or two vertices associated with it, called its **endpoints** (端点). An edge is said to **connect** its endpoints.

注意：这里的"graph" 与函数图像无关。图只关心顶点之间的连接关系（connections），而不关心具体的几何画法。

分类：- **Finite graph** (有限图)：顶点数和边数有限 - **Infinite graph** (无限图)：顶点数或边数无限

9.1.2　2. 无向图的类型 (Types of Undirected Graphs)

类型	英文	特点
简单图	**Simple graph**	每条边连接两个不同顶点；没有平行边（multiple edges）
多重图	**Multigraph**	允许平行边（multiple edges），但不允许自环（loops）
伪图	**Pseudograph**	允许自环（loops）和平行边

图示说明：

- **Simple graph**: 顶点 a, b, c, d 之间，每对顶点至多一条边，没有自环
- **Multigraph**: 顶点之间可以有两条或更多平行边，无自环
- **Pseudograph**: 既有平行边，也有自环（自环从一个顶点出发回到自身）

课件图 10.1-1 详解：课件第 4 页将三种无向图并列对比，每张图使用相同的四个顶点布局——a 在左、b 在上、c 在右、d 在下，呈菱形排列：1. **Simple graph（左上）**：边为 $\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}$，构成一个四边形环。没有平行边或自环。2. **Multigraph（左下）**：在上述四边形环的基础上，a 与 c 之间有两条平行边（对角线方向）。没有自环。3. **Pseudograph（右下）**：既有平行边（如 a 与 d 之间有两条平行边），也包含自环（如顶点 b 有一条从 b 回到 b 的环状边）。

包含关系：

```
Pseudographs
    Multigraphs
        Simple Graphs
```

9.1.3　3. 有向图 (Directed Graphs / Digraphs)

【**Definition 2**】 A **directed graph** (or **digraph**) (V, E) consists of:

- V: a nonempty set of vertices
- E: a set of **directed edges** (or **arcs**)
- Each directed edge is associated with an **ordered pair** (u, v): starts at u and ends at v

有向图的类型：

类型	英文	特点
简单有向图	**Simple directed graph**	无自环，无平行有向边
有向多重图	**Directed multigraph**	允许从一顶点到另一顶点的多条有向边，允许自环

课件图 10.1-2 详解：课件第 7 页对比展示了两种有向图，顶点布局均为 a, b, c, d：1. **Simple directed graph（简单有向图，左图）**：有向边为 $a \to b,\ b \to c,\ b \to d,\ c \to a,\ d \to c$。没有自环，没有平行有向边。2. **Directed multigraph（有向多重图，右图）**：允许从同一顶点到另一顶点的多条有向边（如从 a 到 d 有两条平行有向边），也可能包含自环。

9.1.4　4. 图模型 (Graph Models)

图可以用来建模现实世界中的各种问题。以下是几个经典例子：

社交网络 (Social Networks)

- **Friendship graph** (无向图)：顶点代表人，边代表朋友关系

课件图示：一个 Facebook 好友关系示例图。多个顶点代表不同的用户，无向边连接互为好友的两个用户，直观展示社交网络中人与人之间的连接关系。- **Collaboration graph** (无向图)：顶点代表合作者，边代表合作关系 - **The Hollywood Graph**: 顶点代表演员，边表示两人曾出演同一部电影（即 Kevin Bacon 数问题的背景）- **Academic collaboration graph**: 顶点代表研究者，边表示共同发表论文（即 Erdos 数问题的背景）- **Influence graph** (有向图)：从 A 到 B 的边表示 A 能影响 B **课件图示**：一个影响关系有向图。顶点代表不同个体，有向箭头从影响者指向被影响者。例如 A 影响 B 和 C，而 B 又影响 D，形成一条影响链。

信息网络 (Information Networks)

- **Web graph** (有向图)：顶点代表网页，边代表超链接
- **Citation network** (有向图)：顶点代表论文，边代表引用关系

交通网络 (Transportation Networks)

- 航线网络：顶点为机场，有向边代表航班
- 道路网络：顶点为交叉路口，无向边代表双向道路，有向边代表单向道路

软件设计 (Software Design)

- **Module dependency graph** (有向图)：顶点为软件模块，边表示依赖关系

  课件图示：一个 Web 浏览器的 7 个模块依赖图。7 个顶点代表 7 个软件模块，有向边从被依赖模块指向依赖它的模块。例如模块 M_1 依赖于 M_2 和 M_3，则有一条从 M_2 到 M_1 和从 M_3 到 M_1 的有向边。- **Precedence graph** (有向图)：顶点为程序语句，边表示执行先后顺序 **课件图示**：一个包含 6 条程序语句 $(S_1, S_2, \ldots, S_6)$ 的前驱图。有向边 $S_i \to S_j$ 表示 S_i 必须在 S_j 之前执行。例如 $S_1 \to S_2 \to S_3$ 链表示按顺序执行。

生物网络 (Biological Networks)

- **Niche overlap graph** (生态位重叠图)：顶点代表物种，边代表竞争关系

  课件图示：一个森林生态系统中 9 个物种的生态位重叠图。每个顶点代表一个物种，两个顶点之间有无向边当且仅当这两个物种竞争相同的食物资源。该图展示了生态系统中物种之间的食物竞争关系网络。

9.1.5　5. 典型例题

Example 1: 铁路网络

How can we represent a network of (bi-directional) railways connecting a set of cities?

Solution: 使用 **simple graph**。顶点代表城市，边 $\{a, b\}$ 代表城市 a 和 b 之间有直通铁路。因为铁路是双向的，所以用无向边。

课件图示：图中展示了 5 个城市顶点——Beijing（北京）、Shanghai（上海）、Xi'an（西安）、Hangzhou（杭州）、Haebin（哈尔滨），大致按地理位置分布。无向边连接实际有铁路直连的城市对——例如 Beijing 同时与 Xi'an、Shanghai 和 Haebin 相连，而 Shanghai 与 Hangzhou 相连，构成一张铁路交通图。

Example 2: 循环赛结果

In a round-robin tournament, each team plays against each other team exactly once. How can we represent the results?

Solution: 使用 **directed graph**。有向边 (a, b) 表示队伍 a 击败了队伍 b。

课件图示：图中 4 个顶点代表 4 支冰球队——Maple Leafs、Bruins、Penguins、Lubeck Giants。有向边从获胜队伍指向战败队伍。例如若 Maple Leafs 击败了 Bruins，则有边 Maple Leafs $\to$ Bruins。每个比赛结果都对应一条有向边，构成一个完整的循环赛结果有向图。

9.1.6　小结 (Summary)

概念	英文术语	关键点
图	Graph	$G = (V, E)$，顶点集 + 边集
简单图	Simple graph	无平行边，无自环
多重图	Multigraph	有平行边，无自环
伪图	Pseudograph	有平行边和自环
有向图	Directed graph / Digraph	边有方向，用有序对 (u, v) 表示
图模型	Graph model	用图来建模实际问题（社交、交通、信息等）

考点提示 (Exam Tips)

1. 能区分 **simple graph / multigraph / pseudograph** 和 **directed / undirected**
2. 能根据实际问题描述，选择合适的图类型进行建模
3. 注意：自环（loop）在一个无向图中对顶点的度数贡献为 2（将在 10.2 中详述）

9.2 Chapter 10.2: Graph Terminology and Special Types of Graphs (图术语与特殊图类)

9.2.1 Part 1: 基本术语 (Basic Terminology)

无向图 (Undirected Graphs)

对于无向图 $G = (V, E)$:

术语	英文	含义
相邻	**Adjacent (Neighbors)**	顶点 u, v 之间有边 $\{u, v\}$，则称 u 和 v 相邻
关联	**Incident**	边 e 连接 u 和 v，称 e 与 u, v 关联
端点	**Endpoints**	边 $\{u, v\}$ 的端点是 u 和 v
自环	**Loop**	连接同一顶点的边
度数	**Degree** $\deg(v)$	与顶点 v 关联的边的数目。**自环对度数贡献 2**

特殊顶点:

- **Isolated vertex** (孤立顶点): $\deg(v) = 0$
- **Pendant vertex** (悬挂顶点): $\deg(v) = 1$

课件插图：无向图基本示例：课件顶部展示了一个 6 个顶点的无向图，顶点为 a, b, c, d, e, f。图的连接如下：

- a 与 b, d 相连
- b 与 a, c, e 相连
- c 与 b, f 相连
- d 与 a, e 相连
- e 与 b, d, f 相连
- f 与 c, e 相连

此图用于演示 adjacent（相邻）、incident（关联）、degree（度数）等基本概念。图中 deg(a)=2, deg(b)=3, deg(c)=2, deg(d)=2, deg(e)=3, deg(f)=2，该图显示没有 isolated vertex，也没有 pendant vertex。

【Theorem 1】Handshaking Theorem (握手定理)

Let $G = (V, E)$ be an undirected graph with e edges. Then:

$$> \sum_{v \in V} \deg(v) = 2e >$$

所有顶点的度数之和等于边数的两倍。

Proof：每条边对其两个端点各贡献 1 度（自环贡献 2 度），因此总度数为 $2e$。

推论：对于有向图，类似地有 $\sum \deg^+(v) = \sum \deg^-(v) = |E|$。

常见考题：- 一个图有 5 个顶点，每个顶点度数为 3 是否可能？→ 总和 $3 \times 5 = 15$ 为奇数，不可能。- 每个顶点度数为 4 是否可能？→ 总和 $20 = 2|E|$，$|E| = 10$，可能。

【Theorem 2】偶数个奇度顶点

An undirected graph has an **even number** of vertices of odd degree. 无向图中奇度顶点的个数必为偶数。

Proof：设 V_1 为偶度顶点集，V_2 为奇度顶点集。

$$\sum_{v \in V_1} \deg(v) + \sum_{v \in V_2} \deg(v) = 2m$$

左边第一项为偶数，右边为偶数，所以第二项也必为偶数，因此 $|V_2|$ 为偶数。

有向图 (Directed Graphs)

对于有向图，边 (u, v) 中:

- u 是 **initial vertex** (起点)，u is **adjacent to** v
- v 是 **terminal vertex** (终点)，v is **adjacent from** u

术语	英文	含义
入度	**In-degree** $\deg^-(v)$	以 v 为终点的边的数目
出度	**Out-degree** $\deg^+(v)$	以 v 为起点的边的数目

【Theorem 3】

For a directed graph $G = (V, E)$:

$$> \sum_{v \in V} \deg^+(v) = \sum_{v \in V} \deg^-(v) = |E| >$$

所有顶点的出度之和 = 入度之和 = 边数。

Underlying undirected graph (基础无向图)：将有向图中的每条有向边去掉方向后得到的无向图。

课件插图：有向图示例：课件展示了一个 5 个顶点的有向图，顶点为 a, b, c, d, e。各顶点的连接关系：- a → b, a → d - b → a, b → c - c → a - d → b - e → c, e → d

该图用于演示 initial vertex（起点）、terminal vertex（终点）、in-degree（入度）和 out-degree（出度）的概念。

9.2.2 Part 2: 特殊简单图 (Special Simple Graphs)

(1) Complete Graphs —K_n (完全图)

- 有 n 个顶点
- 每对不同顶点之间恰有一条边
- 边数：$|E| = \binom{n}{2} = \frac{n(n-1)}{2}$
- K_n 是 $(n-1)$-regular（正则图）

课件插图：K_1 到 K_5：课件展示了五个完全图并列排列。

- K_1：单个孤立顶点（无任何边）
- K_2：两个顶点由一条边连接
- K_3：三个顶点两两相连，构成三角形
- K_4：四个顶点两两相连，画成正方形加两条对角线（共 6 条边）
- K_5：五个顶点两两相连，画成五边形加内部五角星形状（共 $\binom{5}{2} = 10$ 条边）

(2) Cycles —C_n ($n \geq 3$) (圈图)

- n 个顶点排成一个环
- 每个顶点度数均为 2
- C_n 是 2-regular

课件插图：C_3 到 C_6：课件展示了四个圈图并列排列，每个顶点度为 2。

- C_3：三个顶点排成三角形（3 条边）
- C_4：四个顶点排成正方形（4 条边）
- C_5：五个顶点排成正五边形（5 条边）
- C_6：六个顶点排成正六边形（6 条边）

所有圈图中每个顶点的度数均为 2。

(3) Wheels —W_n ($n \geq 3$) (轮图)

- 在 C_n 的基础上增加一个中心顶点（hub），与 C_n 中每个顶点相连
- W_n 有 $n+1$ 个顶点
- 中心顶点度数为 n，外围顶点度数为 3

课件插图：W_3 到 W_6：课件展示了四个轮图并列排列，每个都是在对应圈图 C_n 的基础上增加一个中心顶点并连接到圈上所有顶点。

- W_3：三角形加中心点（中心连 3 条边到外围），共 4 个顶点、6 条边，外围顶点度数为 3、中心度数为 3
- W_4：正方形加中心点，共 5 个顶点、8 条边，外围顶点度数为 3、中心度数为 4
- W_5：五边形加中心点，共 6 个顶点、10 条边
- W_6：六边形加中心点，共 7 个顶点、12 条边

(4) n-Cubes —Q_n ($n \geq 1$) (超立方体)

- 顶点：长度为 n 的所有 bit 串（共 2^n 个）
- 边：两个顶点对应的 bit 串恰好有 **1 bit 不同**
- 每个顶点度数为 n

课件插图：Q_1、Q_2、Q_3：课件展示了三个超立方体图并列排列，顶点标记为对应的 bit 串。

- Q_1：两个顶点 0 和 1 由一条边连接
- Q_2：四个顶点 00, 01, 10, 11 排成正方形。00 与 01 和 10 相连，01 与 00 和 11 相连，10 与 00 和 11 相连，11 与 01 和 10 相连（每条边对应一个 bit 不同）
- Q_3：八个顶点 000, 001, 010, 011, 100, 101, 110, 111 形成立方体结构。画成立方体的八个顶角，每条棱连接恰有 1 bit 不同的两个顶点（如 000 连接 001, 010, 100）。每个顶点度数为 3

9.2.3 Part 3: Bipartite Graphs (二部图)

【Definition】 A simple graph G is **bipartite** if V can be partitioned into two disjoint subsets V_1 and V_2 such that every edge connects a vertex in V_1 and a vertex in V_2.

即顶点可以分成两组，所有边都只在两组之间，组内没有边。

课件插图：二部图示例：课件展示了一个 5 个顶点的二部图，顶点为 v1, v2, v3, v4, v5。该图分为两组：$V_1 = \{v_1, v_2, v_3\}$ 和 $V_2 = \{v_4, v_5\}$。连接关系为：- v1 与 v4, v5 相连 - v2 与 v4 相连 - v3 与 v4, v5 相连组内（V_1 内部和 V_2 内部）没有任何边。该图演示了二部图的标准结构。

判断二部图的方法

【Theorem 4】 A simple graph is bipartite **iff** it is possible to assign one of two different colors to each vertex so that no two adjacent vertices have the same color.

一个简单图是二部图当且仅当可以用两种颜色给顶点染色，使得相邻顶点颜色不同。即二部图等价于 **2-colorable**（2-可着色）。

判断技巧：从一个顶点开始用两种颜色交替染色，看是否会出现冲突。如果出现冲突，则不是二部图。

课件示例

〖Example 1〗C_3 是二部图吗？ 课件展示了三角形 C_3，顶点为 v1, v2, v3，三条边为 v1,v2, v2,v3, v3,v1。尝试 2-染色：从 v1 染红色，v2 染蓝色，v3 必须与 v1 和 v2 都不同色 → 需要第三色，因此 C_3 **不是**二部图（奇数长度圈都不是二部图）。**〖Example 2〗C_6 是二部图吗？** 课件首先展示 C_6 的标准六边形（顶点为 v1 到 v6 依次相连），然后将其重新排列成两行：

- 上排：v1, v5, v3
- 下排：v6, v2, v4

此时所有边都在上下两排之间，因此 C_6 是二部图（偶数长度圈都是二部图），可交替染色。

Complete Bipartite Graph —$K_{m,n}$ (完全二部图)

- 顶点集划分为 V_1（m 个顶点）和 V_2（n 个顶点）
- V_1 中每个顶点与 V_2 中每个顶点都相连
- 边数：$m \times n$

课件插图：完全二部图 $K_{3,2}$ 与星型网络 $K_{1,n}$：

- **星型网络** $K_{1,n}$：一个中心顶点（如中央服务器）位于中心，周围 n 个外围顶点（如工作站）呈放射状排列，中心与每个外围相连，外围之间互不相连。这是最常见的星型拓扑结构。
- $K_{3,2}$：左侧 3 个顶点排成一列（V_1），右侧 2 个顶点排成一列（V_2），每个左侧顶点连接所有右侧顶点（共 $3 \times 2 = 6$ 条边），组内无连接。

2-染色判断二部图

课件插图：Theorem 4 的 2-染色示例：课件展示了同一个图（顶点 a-g）的两种布局。

- **左图（原始布局）**：顶点 a 位于顶部，b, c 位于第二层左右，d, e, f 位于第三层，g 位于底部。边交错连接（难以一眼看出是否为二部图）。
- **右图（染色后布局）**：将顶点重排为两行，上排为 a, b, d, c，下排为 e, f, g。所有边均连接上排与下排的顶点，说明该图是二部图（2-colorable）。

Regular Graph (正则图)

- 每个顶点的度数都相同，称为 n-**regular**
- K_n 是 $(n-1)$-regular
- 思考题：$K_{m,n}$ 在什么情况下是正则图？→ 当 $m = n$ 时。

【Example 3】Local Area Networks (局域网拓扑)

LAN 的三种常见拓扑结构可以用特殊图表示：

1. **Star topology (星型)**：对应 $K_{1,n}$（星型网络 = 完全二部图 $K_{1,n}$），中心是中央服务器，外围是工作站 2. **Ring topology (环型)**：对应 C_n（圈图），每个设备连接两个邻居形成环 3. **Hybrid topology (混合型)**：结合星型和环型的结构

课件插图：三种 LAN 拓扑：课件展示了三个网络拓扑示意图。- **Star**：中央一个服务器节点，周围 4 个工作站节点呈放射状连接到中心（$K_{1,4}$ 结构）- **Ring**：4 个节点围成环形，每个节点仅与左右两个邻居相连（C_4 结构）- **Hybrid**：结合了星型和环型——外圈是环形骨干网（4 个节点围成环），其中部分节点下挂星型子网（每个子网有一个中心连接若干叶子节点）

9.2.4 Part 4: Matchings in Bipartite Graphs (二部图中的匹配)

- **Matching** (匹配)：边集 $M \subseteq E$，M 中任意两条边没有公共顶点
- **Maximum matching** (最大匹配)：边数最多的匹配
- **Complete matching** (完全匹配 / 完美匹配)：从 V_1 到 V_2 的匹配，使得 V_1 中每个顶点都被匹配

【Theorem 5】Hall's Marriage Theorem (霍尔婚姻定理)

A bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a **complete matching** from V_1 to V_2 **iff** $|N(A)| \geq |A|$ for all $A \subseteq V_1$.

其中 $N(A)$ 是 A 的邻居集（与 A 中顶点相邻的顶点）。

直观含义：V_1 中任意 k 个顶点必须至少与 V_2 中的 k 个不同顶点相邻，否则不可能有完全匹配。

证明思路（⇒）：显然每个 V_1 中的顶点在匹配中有一个不同的邻居。（⇐）：使用强归纳法，分两种情况讨论。

详细证明（⇐ 方向） 对 $|V_1|$ 做**强归纳 (Strong Induction)**。**Base Case（$|V_1| = 1$）**：V_1 只有一个顶点 v。由条件 $|N(\{v\})| \geq 1$，v 至少有一个邻居，任选一个匹配即可。**Inductive Hypothesis**：假设对所有 $|V_1| < k$ 的图，定理成立。**Inductive Step（$|V_1| = k$）**：分两种情况。

Case 1：所有非空真子集都是严格大于即对任意 $\emptyset \neq A \subsetneq V_1$，都有 $|N(A)| \geq |A| + 1$。

- 任选 $v \in V_1$，任意匹配一个 $w \in N(v)$
- 将 v 和 w 从图中删去，得到 G'（左侧剩 $k-1$ 个顶点）
- 对 G' 中的任意 $B \subseteq V_1 \setminus \{v\}$：原图中 $|N(B)| \geq |B| + 1$，删去 w 最多损失 1 个邻居，故 $|N'(B)| \geq |B|$

- Hall 条件仍满足！由归纳假设，G' 有 complete matching，加上 $\{v, w\}$ 即得原图 complete matching

直观：邻居都比自己多 → 随便配一对、删掉继续，条件不会坏。

Case 2：存在一个子集恰好相等即存在 $\emptyset \neq A \subsetneq V_1$ 使得 $|N(A)| = |A|$。令 $B = N(A)$。|A| = |B|，且 $|A| < k$。**Step 1（切出左半）**：考虑子图 $G_1 = (A \cup B)$。A 的任意子集在原图中的邻居是 B 的子集，Hall 条件成立。$|A| < k$，由归纳假设，A 与 B 之间存在 complete matching。**Step 2（验证右半）**：考虑剩余子图 $G_2 = ((V_1 \setminus A) \cup (V_2 \setminus B))$。需验证 G_2 也满足 Hall 条件。反证：若存在 $C \subseteq V_1 \setminus A$ 使得 $|N'(C)| < |C|$（在 G_2 中邻居不够），则在**原图**中考虑 $A \cup C$：

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

$$|A \cup C| = |A| + |C| > |B| + |N'(C)| = |N(A \cup C)|$$

得出 $|N(A \cup C)| < |A \cup C|$，违反原图 Hall 条件，矛盾！因此 G_2 满足 Hall 条件，且 $|V_1 \setminus A| < k$，由归纳假设存在 complete matching。

直观：找到一个" 刚好够" 的子集 → 把它和它的邻居切出去 → 剩下部分 Hall 条件自动成立，不会崩。

两者合并：G_1 的匹配 + G_2 的匹配 = 原图的 complete matching。□

核心 trick：Case 2 中切出 A 和 $B = N(A)$ 后，右半不可能出现 Hall 条件违反，因为那会连累 A 在原图中也不满足条件。

9.2.5 Part 5: New Graphs from Old (从旧图构造新图)

Subgraph (子图)

- $H = (W, F)$ 是 $G = (V, E)$ 的 **subgraph** 如果 $W \subseteq V$ 且 $F \subseteq E$
- **Proper subgraph**：$H \neq G$
- **Spanning subgraph** (生成子图)：$W = V$（顶点集不变，只取部分边）

课件插图：K_5 及其子图：左侧是 K_5（完全图，5 个顶点两两相连，共 10 条边），右侧是 K_5 的一个子图（取 K_5 的 5 个顶点，但只保留部分边，例如只保留外围五边形的 5 条边，删去了内部对角线），用来说明子图的概念——保留原图的全部顶点但只取部分边。

Induced Subgraph (诱导子图)

- 由顶点子集 W **诱导**的子图：边集 F 包含 E 中两端点都在 W 中的所有边
- 记作 $G[W]$

课件插图：K_5 的诱导子图：左侧是 K_5，右侧是 K_5 由 $W = \{a, b, c, e\}$ 诱导的子图。该子图仅包含顶点 a, b, c, e 以及 K_5 中这四个顶点之间原有的所有边（即 a-b, a-c, a-e, b-c, b-e, c-e，共 6 条边），与原图在这四个顶点上的连接完全相同。展示诱导子图与普通子图的区别——诱导子图必须保留选定顶点之间的所有原边。

Union of Graphs (图的并)

- $G_1 = (V_1, E_1)$ 和 $G_2 = (V_2, E_2)$ 的并：$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$

课件插图：两个子图并成 K_5：展示了 G_1（包含顶点 a, b, c, d, e，边集为外围五边形）、G_2（包含相同顶点集，边集为内部五角星形），以及它们的并 $G_1 \cup G_2 = K_5$（5 个顶点，合并后包含所有边，形成完全图）。用来说明两个简单图的并运算。

【Example 4】W_3 的非空子图计数

How many subgraphs with at least one vertex does W_3 have?

课件中的 W_3 图：W_3 是轮图，由三角形 C_3（3 个顶点围成三角形）加上一个中心顶点（hub）构成，中心与三角形的 3 个顶点各连一条边。因此 W_3 共有 4 个顶点（中心 + 三个外围顶点）和 6 条边（三角形 3 条边 + 中心到三个外围的 3 条边）。本题用此图计算其所有非空子图的个数。

Solution：W_3 有 4 个顶点，需要按选取的顶点数分类计算（每种顶点数下还需考虑边选取的不同方式）：- 选 1 个顶点：$C(4, 1) = 4$ 种 - 选 2 个顶点：$2 \times C(4, 2) = 2 \times 6 = 12$ 种（边存在与否取决于顶点对的位置）- 选 3 个顶点：$2 \times C(4, 3) = 2 \times 4 = 8$ 种 - 选 4 个顶点：$2 \times C(4, 4) = 2 \times 1 = 2$ 种（选边方式不同）

合计：$4 + 12 + 8 + 2 = 26$ 个非空子图。

9.2.6 小结 (Summary)

概念	英文术语	关键点
握手定理	Handshaking Theorem	$\sum \deg(v) = 2e$，总度数为偶数
奇度顶点	Odd-degree vertices	个数必为偶数
出度/入度	In-degree / Out-degree	$\sum \deg^+ = \sum \deg^- = \lvert E \rvert$
完全图	Complete graph K_n	每对不同顶点都有边
圈图	Cycle C_n	n 个顶点环，2-regular
轮图	Wheel W_n	C_n 加中心顶点
超立方体	n-Cube Q_n	2^n 个 bit 串顶点
二部图	Bipartite graph	顶点可二分，边只在两组之间
完全二部图	Complete bipartite $K_{m,n}$	$m \times n$ 条边
霍尔定理	Hall's Marriage Theorem	$\lvert N(A) \rvert \geq \lvert A \rvert$ 是存在完全匹配的充要条件
正则图	Regular graph	所有顶点度数相同

考点提示 (Exam Tips)

1. **Handshaking Theorem** 和奇度顶点个数为偶数的证明是高频考点
2. 判断一个图是否为二部图：尝试 2-染色，看是否冲突
3. C_n 是二部图 iff n 是偶数
4. Hall's Theorem 条件 $|N(A)| \geq |A|$ 是充要条件
5. 区分 subgraph, spanning subgraph, induced subgraph
6. 计算 W_3 的非空子图个数等组合问题

9.3 Chapter 10.3: Representing Graphs and Graph Isomorphism (图的表示与同构)

9.3.1 Part 1: 图的表示方法 (Representing Graphs)

方法一：Adjacency Lists (邻接表)

对每个顶点，列出所有与其相邻的顶点。**无向简单图示例**（课件原图：5 个顶点 a, b, c, d, e 的简单图）：

```
      a
     /|\
    / | \
   /  |  \
  b   e   c
   \  |  /
    \ | /
     \|/
      d
```

边集：*a-b*, *a-c*, *a-e*, *b-c*, *b-d*, *b-e*, *c-d*, *d-e*

```
Vertex  Adjacent Vertices
a       b, c, e
b       a, c, d, e
c       a, b, d
d       b, c, e
e       a, b, d
```

有向图示例（课件原图：5 个顶点 a, b, c, d, e 的有向图）：

```
a   → c

 \
↓  \     ↓
e   ←   d ←   b
```

```
Initial Vertex  Terminal Vertices
a               c, e
b               a, c, d
c               e
d               a, c, e
e               b, c, d
```

方法二：Adjacency Matrices (邻接矩阵)

简单图的邻接矩阵 对于 n 个顶点的简单图 $G = (V, E)$，邻接矩阵 $A = [a_{ij}]$ 是一个 $n \times n$ 的 0-1 矩阵：

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G \\ 0 & \text{otherwise} \end{cases}$$

重要性质：

- 无向图的邻接矩阵是 **对称的**（symmetric）
- 对角线上的元素表示自环，简单图对角线为 0

【Example 1】无向图的邻接矩阵 **课件原图结构**：4 个顶点 a, b, c, d，5 条边。

```
      a
     /|\
    / | \
   /  |  \
  b   |   c
   \  |  /
    \ | /
     \|/
      d
```

边集：$\{a, b\}, \{a, c\}, \{a, d\}, \{b, d\}, \{c, d\}$（$b$ 和 c 之间没有边直连）对顶点顺序 a, b, c, d，邻接矩阵为：

$$A_G = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

矩阵是对称的。第 i 行之和 = 顶点 i 的度数。

多重图/伪图的邻接矩阵 使用 **非负整数矩阵**：

- a_{ij} = 顶点 v_i 和 v_j 之间的边数（平行边的数量）
- 自环计入对角线元素

【Example 2】多重图的邻接矩阵 **课件原图结构**：与 Example 1 相同的骨架（a, b, c, d 蝴蝶形），但增加了：

- a 和 b 之间有 **2 条平行边**
- a 处有 **1 个自环 (loop)**
- c 和 d 之间有 **3 条平行边**
- d 处有 **2 个自环**

对顶点顺序 a, b, c, d：

$$A_G = \begin{bmatrix} 1 & 2 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 1 & 0 & 0 & 3 \\ 1 & 1 & 3 & 2 \end{bmatrix}$$

矩阵依然对称（无向图）。对角线元素 = 自环数，非对角线 = 平行边数。

有向图的邻接矩阵

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \text{ is an edge of } G \\ 0 & \text{otherwise} \end{cases}$$

【Example 3】有向图的邻接矩阵 **课件原图结构**：4 个顶点 a, b, c, d，有向边如下：

```
a    → b

 \
  \
↓      ↓
c ←   d
```

边集：$a \to b,\ a \to c,\ a \to d,\ b \to d,\ d \to a,\ d \to c$ 对顶点顺序 a, b, c, d：

$$A_G = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

不对称！ 第 i 行之和 = $\deg^+(v_i)$（出度），第 j 列之和 = $\deg^-(v_j)$（入度）。

有向图邻接矩阵的性质：- **不一定对称** - 第 i 行之和 = $\deg^+(v_i)$（出度）- 第 j 列之和 = $\deg^-(v_j)$（入度）

行和与列和的意义 问题：无向图邻接矩阵一行之和是什么？

等于与该顶点关联的边的数目（即度数减去自环数）。

问题：有向图邻接矩阵一行之和？一列之和？行和 = $\deg^+(v_i)$，列和 = $\deg^-(v_j)$。

方法三：Incidence Matrices (关联矩阵)

对于无向图 $G=(V,E)$，$|V|=n$, $|E|=m$，关联矩阵 $M=[m_{ij}]$ 是 $n\times m$ 矩阵：

$$m_{ij}=\begin{cases}1 & \text{when edge } e_j \text{ is incident with } v_i\\ 0 & \text{otherwise}\end{cases}$$

性质：

- 每列恰好有两个 1（连接两个不同顶点的边），或一个 1（自环）

【Example 4】关联矩阵示例 **课件原图**：4 个顶点 a,b,c,d，6 条边 e_1,e_2,e_3,e_4,e_5,e_6。

```
          e1
   a        b

     e5

e6                 e2



   c        d
      e3     e4
```

（e_3 和 e_4 是 c 和 d 之间的平行边）对顶点顺序 a,b,c,d，边顺序 e_1,e_2,e_3,e_4,e_5,e_6：

$$M=\begin{bmatrix}1&0&0&0&1&1\\1&1&0&0&0&0\\0&0&1&1&0&1\\0&1&1&1&1&0\end{bmatrix}$$

每列恰好有两个 1（每条边连接两个顶点）。自环列只有一个 1。

9.3.2 Part 2: Graph Isomorphism (图的同构)

同构的定义

【Definition】 Two simple graphs $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$ are **isomorphic** if there exists a **bijection** $f:V_1\to V_2$ such that for all $a,b\in V_1$:

$$\{a,b\}\in E_1 \iff \{f(a),f(b)\}\in E_2$$

直观含义：两个图结构相同（只是顶点的名字不同，画法不同）。

这样的函数 f 称为 **isomorphism**（同构映射）。

Invariants (不变量)

如果两个图同构，它们必须具有以下共同性质（不变量）。这些性质可以用来**证明两个图不同构**：

不变量	说明
顶点数	number of vertices
边数	number of edges
对应顶点的度数	degrees of corresponding vertices
是否二部图	if one is bipartite, the other must be
是否完全图	if one is complete, the other must be
是否轮图	if one is a wheel, the other must be
连通分量数	number of connected components
简单回路的长度	presence of simple circuits of the same length

注意：不变量相同是**必要条件**，但不是**充分条件**。即使所有已知不变量都相同，也不一定能保证同构。

判断同构的技巧

1. **先检查不变量**：顶点数、边数、度数序列
2. **对应度数列**：确定可能的一一对应关系
3. **验证邻接关系**：用邻接矩阵或手动验证是否保持邻接关系
4. **若邻接矩阵经过行列置换后相同**，则同构

【Example 5】判断同构 **课件左图**（5 顶点，8 条边）：五边形 a-b-c-d-e-a + 弦 b-d

```
  a
 / \
/   \
e     b
|\   /|
| \ / |
|  X  |
| / \ |
|/   \|
d     c
```

边集：$\{a,b\},\{b,c\},\{c,d\},\{d,e\},\{e,a\}$（外圈五边形）+ $\{b,d\},\{d,b\}$ 即弦 b-d + $\{a,e\}$ 就是外圈边。实际额外边是 b-d 连接。**课件右图**：结构完全相同，只是画法不同——顶点 b 被移到了 $\{a,c\}$ 的左侧，使得看起来不一样。

Solution：它们同构。如果你把右图的顶点 b 移到边 $\{a,c\}$ 的左边，两图看起来完全一样。

同构映射：

$$f(a)=e,\quad f(b)=a,\quad f(c)=b,\quad f(d)=c,\quad f(e)=d$$

【Example 6】证明同构 **课件原图**：两个 5 顶点图。**左图**（五边形星形）：

```
      v1
     /  \
    /    \
   v5    v2
   |      |
   |      |
   v4----v3
```

边集：$\{v_1,v_2\},\{v_2,v_3\},\{v_3,v_4\},\{v_4,v_5\},\{v_5,v_1\}$（外圈五边形）**右图**（正五边形）：

```
      u1
     /  \
    /    \
   u5    u2
    \    /
     \  /
   u3--u4
```

边集与左图相同（五边形），只是排列方式不同，看起来一个是" 星" 一个是" 正五边形"。

Proof：找双射 f 并验证邻接关系保持不变。两图五边形一一对应。

验证技巧：把左图的邻接矩阵按 f 重排行列后，与右图邻接矩阵完全相同。

【Example 7】判断是否同构 课件中包含 3 组图（(1)(2)(3)），每组给出两幅图，判断是否同构。**套路**：

步骤	检查内容
1	顶点数、边数是否相同？不同 → 不同构
2	度数序列是否相同？不同 → 不同构
3	回路结构：如一个有 3-回路（三角形），另一个没有 → 不同构
4	二部性：一个二部图、一个不是 → 不同构
5	以上都通过 → 可能是同构，需构造双射 f 并验证邻接关系

典型考点：两图有相同的顶点数、边数、度数列，但一个包含三角形（C_3），另一个没有 → 不同构。因为" 是否包含给定长度的回路" 是不变量。

9.3.3 小结 (Summary)

概念	英文术语	关键点
邻接表	Adjacency list	列出每个顶点的邻居
邻接矩阵	Adjacency matrix	$n\times n$ 0-1 矩阵（或非负整数矩阵）
关联矩阵	Incidence matrix	$n\times m$ 矩阵，每列两个 1
同构	Isomorphism	双射 f 保持邻接关系
不变量	Invariant	同构的必要条件，用于排除不同构

考点提示 (Exam Tips)

1. **邻接矩阵**：理解无向图对称、有向图不一定对称
2. **行和与列和**的含义（出度/入度）
3. 判断同构：先检查不变量（顶点数、边数、度数列、回路结构），再用邻接矩阵验证
4. 同构需要 **双射（bijection）** 且 **保持邻接关系**
5. K_5 **的子图计数**等问题需结合组合数学

9.4 Chapter 10.4: Connectivity (连通性)

9.4.1 Part 1: Paths (路径)

路径的定义

无向图中的路径: A **path of length** n in a simple graph is a sequence of vertices $v_0, v_1, \ldots, v_n$ such that $\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{n-1}, v_n\}$ are n edges in the graph.

- **Length** (长度)：路径中边的数目
- **Circuit** (回路 / 圈)：路径起点和终点相同（长度 > 0）
- **Simple path** (简单路径)：不包含重复边
- **Path of length 0**：只包含一个顶点

有向图中的路径：A path of length n is a sequence $v_0, v_1, \ldots, v_n$ such that $(v_0, v_1), (v_1, v_2), \ldots, (v_{n-1}, v_n)$ are n directed edges.

- **Simple path**：不重复使用边
- **Circuit** (回路)：起点和终点相同

路径的应用

Acquaintanceship Graphs (熟人图)：两人之间存在路径当且仅当有一条熟人链条连接他们。著名的"六度分隔"（Six Degrees of Separation）理论就是基于这个概念。

9.4.2 Part 2: 计算路径数量 (Counting Paths)

【Theorem 2】用邻接矩阵计算路径

The number of different paths of length r from v_i to v_j is equal to the (i, j)th entry of A^r, where A is the adjacency matrix.

从 v_i 到 v_j 长度为 r 的不同路径的数量 = 邻接矩阵 A 的 r 次幂的 (i, j) 元。

注意：这里的幂运算是**普通的矩阵乘法**（不是布尔积）。

证明思路：用归纳法。A^2 的 (i, j) 元为 $\sum_{k=1}^{n} a_{ik} a_{kj}$，表示从 v_i 到 v_j 经过某个中间顶点 v_k 的长度为 2 的路径数。依次类推。

【Example 2】路径计数 给定一个图（6 个顶点）：

```
(1) How many paths of length 2 from v5 to v4?
    → 查 A^2 中 a54 = 1（1 条路径）

(2) Number of paths not exceeding length 6 from v4 to v5?
    → 计算 A + A^2 + A^3 + A^4 + A^5 + A^6 中 a45 = 2（2 条路径）

(3) Number of circuits starting at v5 with length  6?
    → 计算 A + A^2 + ... + A^6 中 a55
```

9.4.3 Part 3: 无向图的连通性 (Connectedness in Undirected Graphs)

连通图

【Definition】 An undirected graph is called **connected** if there is a path between every pair of distinct vertices.

如果任意两个不同顶点之间都有路径，则称为连通图。

【Example 3】判断连通性 课件中的两个示例图对比：**图 1（不连通）**：5 个顶点 a, b, c, d, e，分为两个分量——

```
a --- b --- c          d --- e
```

左侧 $\{a, b, c\}$ 由边 ab, bc 构成一条链；右侧 $\{d, e\}$ 由边 de 连接。c 与 d 之间没有路径，因此图不连通。**图 2（连通）**：5 个顶点 a, b, c, d, e 连成一条路径——

```
a --- b --- c --- d --- e
```

任意两个顶点之间都有路径，因此图连通。

【Theorem 1】简单路径的存在性

There is a **simple path** between every pair of distinct vertices of a connected undirected graph.

连通无向图中任意两个不同顶点之间存在一条简单路径。

Proof：因为图连通，存在路径。去掉路径中的所有回路（冗余部分），得到简单路径。

Connected Components (连通分量)

Connected components (or just **components**) are the **maximally connected subgraphs** of G.

连通分量是图的极大连通子图。

示例图（10 个顶点，3 个连通分量）：


```
    i --- h

    d
    |
a --- e
    |
    g

b --- c --- f --- j
```

三个分量：- 分量 1：$\{i, h\}$（一条边相连）- 分量 2：$\{a, d, e, g\}$（a 与 e 相连，d 和 g 分别连接到 e，构成十字形结构）- 分量 3：$\{b, c, f, j\}$（b 连 c，c 连 f，f 连 j，形成一条链）

Cut Vertices and Cut Edges (割点与割边)

- **Cut vertex** (or **articulation point**)：删除该顶点及其关联边后，连通分量数增加
- **Cut edge** (or **bridge**)：删除该边后，连通分量数增加

示例：图 1）星型网络 (Star network)：

```
    v
    |
v --- v
    |
    v
```

中心顶点 $\star$ 是**割点**——删除后所有叶子成为孤立顶点，连通分量数从 1 变为 4。每条边 $\{\star, v_1\}, \{\star, v_2\}, \{\star, v_3\}, \{\star, v_4\}$ 都是**割边（桥）**——删除其中任意一条都会使对应的叶子变为孤立顶点，增加连通分量数。**图 2）既无割点也无割边的图** G：

```
  a --- b
 /       \
d         c
 \       /
  e --- f
```

每个顶点的度数均为 2，整个图构成一个环状结构。删除任意一个顶点或任意一条边后图仍然连通，连通分量数不变，因此没有割点和割边。

9.4.4 Part 4: 有向图的连通性 (Connectedness in Directed Graphs)

强连通 vs 弱连通

类型	英文	定义
强连通	**Strongly connected**	对任意顶点 a, b，存在从 a 到 b 和从 b 到 a 的路径
弱连通	**Weakly connected**	基础无向图（去掉方向后）是连通的

注意：强连通 $\Rightarrow$ 弱连通，反之不成立。

图示（4 个顶点的有向图）：

弱连通但不强连通：

```
a ---→ b
↑      ↓
d ←--- c
```

基础无向图（去掉箭头后）是连通的：所有 4 个顶点均通过无向边相连。但在有向意义下，并非所有顶点对之间都有双向路径——例如存在 $a \to b$，但不存在 $b \to a$ 的路径；存在 $c \to d$，但不存在 $d \to c$ 的路径（仅 $d \leftarrow c$）。因此图是弱连通但不是强连通。

强连通：

```
a → b
↑  ↓
d ← c
```

每个顶点都能通过有向路径到达任何其他顶点。例如存在回路 $a \to b \to c \to d \to a$，所有顶点之间双向可达。因此是强连通图。

Strongly Connected Components (强连通分量)

The **strongly connected components** (强连通分量) are the maximal strongly connected subgraphs of a directed graph. **常见算法**：

- **Kosaraju's algorithm**
- **Tarjan's algorithm**

SCC 示例图（5 个顶点）：

```
a ↔ b
```

```
↓
c →→→ e
↓
d
```

边连接：$a \leftrightarrow b$（双向），$a \to c$，$c \to d$，$c \to e$，$e \to b$ 该有向图的强连通分量分解：

- **SCC1:** $\{a, b\}$ ——a 与 b 双向可达（$a \to b, b \to a$）
- **SCC2:** $\{c, e\}$ ——c 可到达 e（$c \to e$），且 e 可通过 $e \to b \to a \to c$ 回到 c（经过 SCC1 中的顶点），因此在缩点后的 DAG 中 $\{c, e\}$ 属于同一 SCC。注意 b 和 a 属于已识别出的另一个 SCC。
- **SCC3:** $\{d\}$ ——$c \to d$ 但 d 没有指向其他顶点的出边，无法回到 c，因此 d 单独构成一个 SCC

9.4.5 Part 5: Paths and Isomorphism (路径与同构)

路径可以用作判断同构的**不变量**：

1. **回路长度**：两个图同构仅当它们有相同长度的简单回路（simple circuits）
2. **路径结构**：可以通过路径找到顶点的对应关系
3. **度数序列**结合路径信息

【Example 4】用回路长度判断不同构 两个图：

- 左侧图：只有长度为 4 的回路（四边形的各边）
- 右侧图：包含长度为 3 的回路（三角形）

结论：不同构。因为右侧图有 3-回路而左侧没有。

9.4.6 小结 (Summary)

概念	英文术语	关键点
路径	Path	顶点序列，相邻顶点之间有边
回路	Circuit / Cycle	起点 = 终点的路径
简单路径	Simple path	不重复使用同一条边
连通图	Connected graph	任意两顶点都有路径
连通分量	Connected component	极大连通子图
割点	Cut vertex / Articulation point	删除后增加连通分量数
割边	Cut edge / Bridge	删除后增加连通分量数
强连通	Strongly connected	有向图中双向可达
弱连通	Weakly connected	基础无向图连通
强连通分量	Strongly connected component	极大强连通子图
用 A^r 计数路径	Counting paths	A^r 的 (i, j) 元是长度为 r 的路径数

考点提示 (Exam Tips)

1. **用邻接矩阵幂计算路径数量**是重要计算题型
2. 区分 **connected**（无向图）、**strongly connected** 和 **weakly connected**（有向图）
3. 理解 **cut vertex** 和 **bridge** 的概念及其应用（如网络可靠性）
4. 路径和回路结构可以作为判断 **graph isomorphism** 的不变量
5. 一个奇数长度的圈（$C_3, C_5, \ldots$）包含三角形等回路结构，是判断同构的重要依据

9.5 Chapter 10.5: Euler and Hamilton Paths (欧拉路径与哈密顿路径)

9.5.1 Part 1: Euler Paths and Circuits (欧拉路径与回路)

1.1 历史背景：Konigsberg Seven Bridge Problem (哥尼斯堡七桥问题)

在哥尼斯堡（Konigsberg），有一条河穿过城市，河中有两个岛，通过七座桥连接。问题：**能否从某处出发，恰好穿过每座桥一次并返回起点？**欧拉将此问题抽象为**图论问题**：

- 将陆地抽象为顶点（A, B, C, D）
- 将桥抽象为边（多重边）

问题转化为：图中是否存在一条**包含所有边的简单回路？原始地图示意**：

```
   ~~~ 河流 ~~~
  北岸 A
    |  \
[桥1]   [桥2]
    |      \
~~岛 C~~   ~~岛 D~~
    | \    / |
[桥3][桥4][桥5]
    |   \   |
  南岸 B
    [桥6] [桥7]
```

四块陆地：北岸 A、南岸 B、岛 C、岛 D，之间有七座桥连接。**抽象为多重图**：

```
    A
   /|\
  / | \
 /  |  \
C---*---D
 \  |  /
  \ | /
   \|/
    B
```

顶点 A, B, C, D 代表四块陆地，每条边代表一座桥。四个顶点的度数均为 3（奇数）。

1.2 术语定义

术语	英文	定义
欧拉路径	**Euler path**	包含图中每条边恰好一次的**简单路径** (simple path containing every edge)
欧拉回路	**Euler circuit**	包含图中每条边恰好一次的**简单回路** (simple circuit containing every edge)
欧拉图	**Euler graph**	存在欧拉回路的图

通俗理解：**一笔画问题**—能否不重复地一笔画出整个图，并回到起点（欧拉回路）或不回到起点（欧拉路径）。

1.3 判定条件 (Necessary and Sufficient Conditions)

【Theorem 1】A connected multigraph has an Euler circuit iff every vertex has even degree.

连通多重图存在欧拉回路 $\iff$ 每个顶点的度数均为偶数。

Proof (必要性的直观理解)：- 考虑欧拉回路，每次进入一个顶点后必须离开 - 起点：离开一次 + 返回一次 = 贡献 2 度（偶数）- 中间顶点：进入一次 + 离开一次 = 贡献 2 度（偶数）- 因此所有顶点度数必为偶数

Proof (充分性的大致思路)：1. 从任意顶点开始构造一个简单回路 2. 如果所有边已用完，就得到了欧拉回路 3. 否则，从剩余子图中找一个与当前回路有公共顶点的回路 4. 将两个回路" 拼接" 在一起 5. 重复直到所有边用完

课件中的逐步构造示例（图 G，顶点 a, b, c, d, e, f）：

```
a --- f
|     |
|     |
b --- c
     / \
    d   e
```

具体构建步骤：1. 从 a 出发构造回路：$a \to f \to c \to b \to a$（路径边：$af, fc, cb, ba$）2. 删除已用边后，剩余子图 H 包含顶点 c, d, e 及边 cd, de, ec 3. 从公共顶点 c 出发，在 H 中构造子回路：$c \to d \to e \to c$ 4. 拼接两条回路得到欧拉回路：$a \to f \to c \to d \to e \to c \to b \to a$

【Theorem 2】A connected multigraph has an Euler path but not an Euler circuit iff it has exactly two vertices of odd degree.

连通多重图存在欧拉路径但不存在欧拉回路 $\iff$ 恰好有两个奇度顶点。

重要：这两个奇度顶点就是欧拉路径的**起点和终点**。

【Example 1】Konigsberg Seven Bridge Problem 七桥问题的抽象图有 4 个顶点，每个顶点的度数：

- A：度 3（奇数）
- B：度 3（奇数）
- C：度 3（奇数）
- D：度 3（奇数）

结论：有 4 个奇度顶点，既不满足欧拉回路条件（需要全偶数），也不满足欧拉路径条件（需要恰好 2 个奇度顶点）。因此**无解**。

【Example 2】判断欧拉路径 课件中的图（10 个顶点 $A, B, C, D, E, F, G, H, I, J$）：

```
  B
  |
  C
 / \
A   J
```

```
\ /
 D
 |
 E -- F -- G
      / \
     H   I
```

度数分析：该图有 2 个奇度顶点，其余顶点均为偶度。根据 Theorem 2，存在欧拉路径（从其中一个奇度顶点开始，到另一个奇度顶点结束）。**一条欧拉路径**：

```
A → C → E → F → G → I → J → E → A → B → C → D → E → G → H → I → G → J
```

这条路径经过了图中的每条边恰好一次。

1.4 有向图的欧拉回路与路径

有向图存在欧拉回路的条件：

1. The graph is **weakly connected**（弱连通）
2. $\deg^-(v) = \deg^+(v)$ for every vertex v（每个顶点的入度 = 出度）

有向图存在欧拉路径（无回路）的条件：

1. The graph is weakly connected
2. 恰好有两个顶点满足 $|\deg^+(v) - \deg^-(v)| = 1$（一个入度比出度大 1，一个出度比入度大 1），其余顶点入度 = 出度

【Example 3】有向图欧拉路径判定 顶点度数表：

Vertex	deg-	deg+
a	1	2
b	2	2
c	2	2
d	3	2

- $\deg^+(a) - \deg^-(a) = 1$，$\deg^-(d) - \deg^+(d) = 1$
- 其余顶点入度 = 出度
- 弱连通
- **结论**：存在欧拉路径

1.5 应用 (Applications)

1. **一笔画问题**：能否不提起笔、不重复地画出图形
2. **Chinese Postman Problem (中国邮递员问题)**：由管梅谷于 1962 年提出。邮递员需要走遍所有街道至少一次并返回邮局，求最短路径。如果图有欧拉回路，则总路程为所有边权之和
3. 网络路由、分子生物学等领域

9.5.2 Part 2: Hamilton Paths and Circuits (哈密顿路径与回路)

2.0 历史背景：Hamilton 的十二面体游戏

Hamilton 提出过一个谜题：沿着正十二面体（dodecahedron）的棱行走，从某个城市（顶点）出发，访问其余 19 个城市各**恰好一次**，最后返回起点。**正十二面体展开示意（20 个顶点）**：

```
  A --- B --- C
 / \   / \   / \
D   E-F---G-H   I
|   |     |   |   |
J---K     L---M---N
 \ / \   / \ / \ /
  O   P-Q---R   S
   \ /     \ /
    T       U
```

每个顶点代表一个城市，每条边代表十二面体的棱。问题等价于：该图中是否存在经过每个顶点恰好一次的**哈密顿回路**？这个谜题直接引出了**哈密顿路径**和**哈密顿回路**的概念。

2.1 定义

术语	英文	定义
哈密顿路径	**Hamilton path**	经过每个顶点**恰好一次**的路径
哈密顿回路	**Hamilton circuit / cycle**	经过每个顶点**恰好一次**的回路（起点终点相同）
哈密顿图	**Hamilton graph**	存在哈密顿回路的连通图

关键区别：Euler 关注**边**（每条边恰好一次），Hamilton 关注**顶点**（每个顶点恰好一次）。

2.2 存在性的充分条件 (Sufficient Conditions)

【Theorem 3】Dirac's Theorem (迪拉克定理)

If G is a simple graph with $n \geq 3$ vertices such that the degree of **every vertex** in G is at least $n/2$, then G has a Hamilton circuit.

如果简单图 G 有 $n \geq 3$ 个顶点且每个顶点的度数 $\geq n/2$，则 G 存在哈密顿回路。

【Theorem 4】Ore's Theorem (奥尔定理)

If G is a simple graph with $n \geq 3$ vertices such that $\deg(u) + \deg(v) \geq n$ for **every pair of nonadjacent vertices** u and v, then G has a Hamilton circuit.

如果简单图 G 有 $n \geq 3$ 个顶点且任意两个不相邻顶点的度数之和 $\geq n$，则 G 存在哈密顿回路。

注意：这些是**充分条件**（满足则一定有），但不是必要条件（不满足也可能有）。

【Example 4】K_n $(n \geq 3)$ 总有哈密顿回路 因为 K_n 中任意两个顶点都相邻，且每条边都存在，我们可以从任意顶点开始按任意顺序访问其他顶点，最后回到起点。

2.3 存在性的必要条件 (Necessary Conditions)

注意：这些条件是**必要**的（没有则不成立），但不是充分的（满足也未必有）。

哈密顿路径的必要条件：- G 是连通的 - 最多有两个顶点的度数小于 2

哈密顿回路的必要条件：- 每个顶点的度数 ≥ 2

重要性质：- 如果某个顶点的度数为 2，则与该顶点关联的两条边**必须**在哈密顿回路中 - 构造哈密顿回路时，一旦经过一个顶点，该顶点其他未使用的边可以忽略

另一个重要的必要条件：- If G is a Hamilton graph, then for any nonempty subset $S \subseteq V$, the number of connected components in $G - S \leq |S|$. - 即**删除 S 后得到的分量数不能超过 S 的大小**。这是证明某些图没有哈密顿回路的重要工具。

【Example 5】判断是否有哈密顿回路 课件中的示例图：**图 1（无哈密顿回路）**：

```
a --- b --- c
      |
d --- e --- f
```

- 顶点 b 和 e 的度数为 3，a, c, d, f 的度数为 2
- 虽然所有顶点度数 ≥ 2，但删除顶点集 $S = \{b, e\}$ 后，图分裂为 $\{a, c\}$、$\{d\}$、$\{f\}$ 三个分量，$3 > |S| = 2$，不满足必要条件
- 因此**不存在**哈密顿回路

图 2（可能有哈密顿回路）：

```
 a --- b
/|     |\
d |     | c
 \|     |/
  e --- f
```

- 每个顶点度数 ≥ 2，结构紧凑
- 存在哈密顿回路如 $a \to b \to c \to f \to e \to d \to a$

判断技巧：

- 如果图中有度数为 1 的顶点 → 无哈密顿回路
- 如果图结构在删除某些顶点后分裂出太多分量 → 无哈密顿回路
- 如果所有顶点度数 $\geq n/2$（Dirac 定理）→ 一定有哈密顿回路

【Example 6】圆桌座位安排问题 7 个人 A, B, C, D, E, F, G，每人会一些语言：

```
A: English
B: English, Chinese
C: English, Italian, Russian
D: Japanese, Chinese
E: German, Italian
F: French, Japanese, Russian
G: French, German
```

问能否安排圆桌座位使得每两个相邻的人都能交流？**Solution**：

1. **构造图**：顶点代表人，如果两人有共同语言则连边
2. 寻找**哈密顿回路**：例如 $A \to B \to D \to F \to G \to E \to C \to A$
3. 如果存在哈密顿回路，则按回路顺序安排座位即可

2.4 应用 (Applications)

1. 旅行商问题 **TSP** (Traveling Salesman Problem)：寻找最短的哈密顿回路（带权完全图）
2. 通信网络中的路由规划
3. DNA 测序等生物信息学问题

9.5.3 小结：Euler vs Hamilton 对比

对比项	Euler 路径/回路	Hamilton 路径/回路
关注对象	每条边恰好一次 (edges)	每个顶点恰好一次 (vertices)
判定条件	充要条件明确（度数判定）	没有简单的充要条件
Euler 回路条件	所有顶点度数为偶数	—
Euler 路径条件	恰好两个奇度顶点	—
Hamilton 充分条件	—	Dirac: $\deg(v) \geq n/2$; Ore: $\deg(u) + \deg(v) \geq n$
难度	容易判定	很难（NP-complete 问题）

考点提示 (Exam Tips)

1. **必须能区分 Euler 和 Hamilton**！考试中常见的混淆点
2. Euler: 看**边**，条件简单明确（奇偶度）
3. Hamilton: 看**顶点**，条件复杂（Dirac / Ore / 必要条件的组合应用）
4. 判断欧拉回路/路径：先数每个顶点的度数
5. 使用 $G - S$ 的分量数 $\leq |S|$ 的必要条件来排除哈密顿回路
6. 度数为 2 的顶点在哈密顿回路中必须使用其两条边

9.6 Chapter 10.6: Shortest Path Problems (最短路径问题)

9.6.1 Part 1: 带权图与最短路径问题 (Weighted Graphs)

Weighted Graph (带权图)

A **weighted graph** $G = (V, E, W)$ assigns a weight (weight) to each edge. **图示**：

```
  b ----- d
 /| \    /|\
 4/ |  \5 / |6\
a  |   \ /  |  z
 3\ |   /\  | /2
   \| /2  \1|/
    c ----- e
       10
```

边上的数字表示权重（如距离、时间、费用等）。

The Length of a Path (路径长度)

The **length** of a path in a weighted graph = the **sum of the weights** of the edges on this path.

Shortest Path Problem (最短路径问题)

Given a weighted graph $G = (V, E, W)$ where $w(x, y)$ is the weight of edge (x, y) (if $(x, y) \notin E$, $w(x, y) = \infty$), and two vertices a and z, find the **shortest path** between a and z (the path with minimum total weight).

9.6.2 Part 2: Dijkstra's Algorithm (Dijkstra 算法)

算法简介

- 由荷兰数学家 Edsger Dijkstra 于 1959 年提出
- 用于解决**带正权无向连通简单图**中的单源最短路径问题
- 是一个**迭代算法**（iterative procedure）
- 每次找到一个离起点最近的新顶点，直到找到目标顶点 z

核心思想

Dijkstra 算法通过维护一个**已区分顶点集** S（distinguished set）来工作：

- 在第 k 次迭代中，从未加入 S 的顶点中找出标签最小的顶点加入 S
- 然后更新其他未加入 S 的顶点的标签

符号说明

符号	含义
$L(v)$	从起点 a 到 v 的当前最短路径长度
S	已确定最短路径的顶点集合
$w(u, v)$	边 (u, v) 的权重

算法伪代码

```
Procedure Dijkstra(G: weighted connected simple graph, with all weights positive)
{G has vertices a = v0, v1, ..., vn = z and weights w(vi, vj)
 where w(vi, vj) = ϖ if {vi, vj} is not an edge in G}

For i := 1 to n
    L(vi) := ϖ
L(a) := 0
S :=

While z   S
Begin
    u := a vertex not in S with L(u) minimal
    S := S   {u}
    For all vertices v not in S
        If L(u) + w(u, v) < L(v) then L(v) := L(u) + w(u, v)
End
{L(z) = length of shortest path from a to z}
```

更新规则（关键公式）：

$$L_k(v) = \min\{L_{k-1}(v),\ L_{k-1}(u) + w(u, v)\}$$

其中 u 是第 k 次迭代加入 S 的顶点。

【Example 1a】Dijkstra 算法初步示例

课件中的第一个示例图（顶点 a, b, c, d, e, z）：

```
   b ----- c
    \     /
     \   /
      \ /
a ----- d ----- e ----- z
```

边权：$a - b = 4$, $a - d = 1$, $b - c = 3$, $d - e = 2$, $c - z = 2$, $e - z = 3$ **逐步求解**：

1. 从 a 出发最近的顶点是 d（路径 $a \to d$，长度 1）
2. 第二近的是 b（考虑已通过 a 和 d 的路径：$a \to b$ 长度 4）
3. 第三近的是 e（通过 a, d 和 b 可达）
4. 第四近的是 z（最终找到目标）

这个简单示例展示了 Dijkstra 算法逐步扩展搜索范围的核心思想。

【Example 1b】Dijkstra 算法完整演示

第二个示例图（顶点 a, b, c, d, e, z）：

```
    b ----- d
  4/ \     / \6
a   5\   /8   z
  3\  \ /    /2
    c -- e --
       10
```

逐步演示：Step 0 (初始化)：

```
S =
L(a)=0, L(b)=ϖ, L(c)=ϖ, L(d)=ϖ, L(e)=ϖ, L(z)=ϖ
```

Step 1 (加入 a)：

```
S = {a}
L(b)=4(a), L(c)=2(a), L(d)=ϖ, L(e)=ϖ, L(z)=ϖ
```

- 选择最小的 L(c)=2 加入 S

Step 2 (加入 c)：

```
S = {a, c}
L(b)=min(4, 2+1)=3(a→c), L(d)=min(ϖ, 2+8)=10(a,c), L(e)=min(ϖ, 2+10)=12(a,c), L(z)=ϖ
```

- 选择最小的 L(b)=3 加入 S

Step 3 (加入 b)：

```
S = {a, c, b}
L(d)=min(10, 3+5)=8(a,c,b), L(e)=min(12, 3+ϖ)=12(a,c), L(z)=ϖ
```

- 选择最小的 L(d)=8 加入 S

Step 4 (加入 d):

```
S = {a, c, b, d}
L(e)=min(12, 8+1)=10(a,c,b,d), L(z)=min(ϖ, 8+6)=14(a,c,b,d)
```

- 选择最小的 L(e)=10 加入 S

Step 5 (加入 e):

```
S = {a, c, b, d, e}
L(z)=min(14, 10+2)=13(a,c,b,d,e)
```

- 选择最小的 L(z)=13 加入 S

Step 6 (找到 z):

```
S = {a, c, b, d, e, z}
L(z)=13
```

结果：最短路径长度为 **13**，路径为 $a \to c \to b \to d \to e \to z$。

算法正确性

【Theorem 1】 Dijkstra's algorithm finds the length of a shortest path between two vertices in a connected simple undirected weighted graph. (仅适用于正权图) **证明思路（归纳法）**:

- **归纳假设**：在第 k 次迭代结束时：

1. S 中每个顶点的标签就是从 a 到该顶点的最短路径长度
2. 不在 S 中的每个顶点的标签是从 a 到该顶点的、只经过 S 中顶点的最短路径长度

- 基础情况 $k = 0$ 显然成立
- 归纳步骤证明每次加入的顶点 v 的标签确实是最短路径长度

时间复杂度

【Theorem 2】 Dijkstra's algorithm uses $O(n^2)$ operations (additions and comparisons), where n is the number of vertices.

- 最多 $n-1$ 次迭代
- 每次迭代：$O(n)$ 找最小标签 + $O(n)$ 更新标签

9.6.3 Part 3: 补充算法—Floyd's Algorithm

基本思想

Floyd 算法可以**一次性求出所有顶点对之间的最短路径**。

- 允许负权边，但不能有**负权回路**（negative-weight cycle）
- 时间复杂度 $O(n^3)$

核心思想:

```
For each vertex vi:
    For each pair (vj, vk):
        If d(vj, vi) + d(vi, vk) < d(vj, vk) then
            update d(vj, vk) = d(vj, vi) + d(vi, vk)
```

即检查从 v_j 到 v_k 的路径是否可以通过走 v_i 变得更短。**注意**:

- Floyd 算法可以找出最短路径的**长度**，但不能直接构造最短路径本身
- 可以处理有负权边的图（但不能有负权回路）

9.6.4 Part 4: The Traveling Salesman Problem (TSP, 旅行商问题)

问题描述

The traveling salesman problem asks for the **circuit of minimum total weight** in a weighted, complete, undirected graph that visits each vertex exactly once and returns to its starting point.

在带权完全无向图中，寻找**总权重最小的哈密顿回路**。

求解方法

(1) 暴力枚举法 (Brute Force)

- 检查所有可能的哈密顿回路
- 哈密顿回路数量：$(n-1)!/2$（因为顺逆时针重复）
- 时间复杂度：$O(n!)$，**不可行**（NP-hard）

(2) 近似算法 (Approximation Algorithm)

- 不一定找到精确解，但保证找到接近最优解的解
- 例如最近邻算法（nearest neighbor）等
- 时间复杂度较低：$O(n^2)$

课件中的 TSP 示例图（5 顶点完全带权图）:

```
      a
     /|\
    / | \
   /  |  \
  b---c---d
   \ / \ /
    e---?
```

边权（部分示例如下，完整边权见课件）：a 到 b 权重 14，a 到 e 权重 12，b 到 c 权重 9，b 到 e 权重 5，c 到 d 权重 11，c 到 e 权重 8，d 到 e 权重 6，等等。

- **最优解**（精确哈密顿回路）：$a \to c \to e \to b \to d \to a$，总长度 = **37**
- **近似解**（例如最近邻算法）：总长度 = 40

9.6.5 小结 (Summary)

概念	英文术语	关键点
带权图	Weighted graph	每条边有一个权重
路径长度	Length of path	路径上边权之和
Dijkstra 算法	Dijkstra's algorithm	单源最短路径，$O(n^2)$，**仅正权**
标签更新	Label update	$L(v) = \min(L(v), L(u) + w(u,v))$
Floyd 算法	Floyd's algorithm	所有点对最短路径，$O(n^3)$，允许负权（无负环）
旅行商问题	TSP	最短哈密顿回路，NP-hard

考点提示 (Exam Tips)

1. **Dijkstra 算法是必考内容**—要能手动模拟每一步，记录标签变化
2. 注意 Dijkstra 算法**不能处理负权边**
3. 区分 Dijkstra（单源）和 Floyd（全源）
4. TSP 的本质是**带权哈密顿回路**问题
5. Dijkstra 每一步选择标签最小的顶点加入 S 是关键操作
6. 学会用表格形式记录每次迭代的标签更新（考试中常见方式）

9.7 Chapter 10.7: Planar Graphs (平面图)

9.7.1 Part 1: 平面图的基本概念 (What is a Planar Graph?)

【Definition】 A graph is called **planar** if it can be drawn in the plane **without any edges crossing** (没有边交叉). Such a drawing is called a **planar representation** of the graph. **平面图的重要性**:

- 地图（maps）的表示都是平面图
- 电子电路（electronic circuits）通常用平面图表示（PCB 板的布线不能交叉）

【Example 1】判断是否是平面图 (1) 有些图看起来有交叉边，但可以通过**重新画法**避免交叉 → 是平面图 (2) 有些图无论如何都无法避免边交叉 → 非平面图 (3) 需要反复尝试不同的画法

关键：判断一个图是否为平面图，不等价于"当前画法是否有交叉"，而是**是否存在一种画法使得边不交叉**。

图示——平面图与非平面图示例:

(1) **K4（完全图 4 顶点）**——虽然是平面图，但不同画法效果不同:

```
有交叉的画法:                无交叉的平面画法:

a  b                         a
|\  /|                       /|\
| \/ |                      / | \
| /\ |                     b   c
|/  \|                      \ | /
c  d                        \|
                               d
```

K4 虽然可以画成对角线交叉的形式，但存在无交叉的平面画法，因此是平面图。这说明：判断平面图不等价于"当前画法没有交叉"，而是**是否存在一种无交叉的画法**。

(2) 其他平面图：通过移动顶点、重新绘制边可避免交叉。

(3) **非平面图**（如 K5、K_3,3）：无论如何调整，总会有边交叉。

9.7.2 Part 2: Euler's Formula (欧拉公式)

Regions (区域)

A **region** is a part of the plane completely disconnected off from other parts of the plane by the edges of the graph.

- **Bounded region** (有界区域)：四周被边包围
- **Unbounded region** (无界区域)：图形最外面的无限区域
- 每个平面图有且仅有一个**无界区域**

区域划分图示：

```
一个平面图将平面划分为若干区域：

  a   b   c

   R1   R2     ← R1、R2 为有界区域（被边包围）

  d   e   f

    R3（外部）      ← R3 为无界区域
                      （最外面的无限区域）
```

每个区域由边围成，有界区域的边界完全由图的边组成，无界区域的边界部分由图的边构成、部分延伸至无限远。

【Example 2】数区域 (1) 简单三角形图：内部 1 个有界区域 + 外部 1 个无界区域 = 共 2 个区域 (2) 较复杂的图：每个" 封闭面" 是一个区域，包括最外面的无限面

- 示例图有 4 个有界区域 + 1 个无界区域 = 共 5 个区域

【Theorem 1】Euler's Formula (欧拉公式)

Let G be a **connected planar simple graph** with e edges and v vertices. Let r be the number of regions in a planar representation of G. Then:

$$> r = e - v + 2 >$$

连通平面简单图满足：顶点数 - 边数 + 面数 = 2

Proof (构造性证明)：1. 从 G 中任意选一条边得到 $G_1 \to e_1 = 1, v_1 = 2, r_1 = 1$，满足 $r_1 = e_1 - v_1 + 2$ 2. 逐步添加边，每次添加分为两种情况：- **情况一**：新边的两个端点都已存在 → 在新旧顶点之间形成一个新的区域，r 加 1，e 加 1，v 不变 - **情况二**：新边的一个端点是新顶点 → 不产生新区域，r 不变，e 加 1，v 加 1 3. 归纳可得公式始终成立

两种情况的图示：

情况一：新边的两个端点都已存在（两个顶点都在当前子图中）

```
      Gn                          Gn+1
a     b                    a     b
              添加边
    R           →          R1   R2
            (a  , b  )
c     d                    c     d

添加前：区域 R 存在              添加后：R 被新边分成 R1 和 R2
r  = r, e  = e, v  = v        r   = r+1, e   = e+1, v   = v
```

情况二：新边的一个端点是新顶点（只有一个端点在当前子图中）

```
      Gn                          Gn+1
a     b                    a     b

添加边 (a, x)    ←新增顶点 x
      →                   x

添加前：                        添加后：
r  = r, e  = e, v  = v        r   = r, e   = e+1, v   = v+1
                              不产生新区域
```

注意：- 欧拉公式是**必要条件**，可以用来证明某些图不是平面图 - 对于**不连通**的平面图，公式需修改为 $r = e - v + c + 1$（其中 c 是连通分量数）

【Example 3】应用欧拉公式

A connected planar simple graph has 20 vertices, each of degree 3. How many regions does this planar graph have?

Solution：- 握手定理：$3v = 2e \Rightarrow 3 \times 20 = 2e \Rightarrow e = 30$ - 欧拉公式：$r = e - v + 2 = 30 - 20 + 2 = 12$ - 该平面图有 **12 个区域**

Degree of a Region (区域的度数)

【Definition】 The **degree** of a region R, denoted $\mathrm{Deg}(R)$, is the number of edges on the boundary of R. **图示**：

```
R1     Deg(R1) = 12（12 条边围成的区域）


R2     Deg(R2) = 4（4 条边围成的区域）
```

重要关系：$\sum \mathrm{Deg}(R_i) = 2e$（每条边被两个区域共享，或被同一区域计两次）

【Corollary 1】边数上界

If G is a connected planar simple graph with e edges and v vertices where $v \geq 3$, then:

$$> e \leq 3v - 6 >$$

Proof：- 每个区域的度数至少为 3 - 所以 $2e = \sum \mathrm{Deg}(R_i) \geq 3r \Rightarrow r \leq \frac{2}{3}e$ - 代入欧拉公式：$e - v + 2 \leq \frac{2}{3}e \Rightarrow e \leq 3v - 6$

等号成立条件：当且仅当每个区域恰好有 3 条边。

应用：用此不等式可以证明某些图是非平面图。

【Example 4】证明 K_5 是非平面图 K_5 有 $v = 5$ 个顶点，$e = 10$ 条边。**K5 图形结构**（5 个顶点两两相连的完全图）：

```
K5 试图画在平面上时必然产生边交叉：

        a
       /|\
      / | \
     b c d
      \ | /
       \|/
        e

说明：K5 有 5 个顶点、10 条边。
任意两个顶点之间都有边直接相连，
因此任何平面画法都至少有一条边交叉。
用不等式验证：e = 10 > 3v - 6 = 9，故非平面。
```

K5 中每对顶点之间都有边相连（完全图），因此任何画法中总会有边交叉。

检查不等式：$e \leq 3v - 6 \Rightarrow 10 \leq 3 \times 5 - 6 = 9$? 不成立！因为 $10 \not\leq 9$，所以 K_5 **不是平面图**。

【Corollary 2】最小度数定理

If G is a connected planar simple graph, then G has a vertex of degree not exceeding five ($\delta(G) \leq 5$).

任何连通平面简单图至少有一个顶点的度数不超过 5。

Proof：- 若 $v \leq 2$，显然成立 - 若 $v \geq 3$，由 Corollary 1：$e \leq 3v - 6 \Rightarrow 2e \leq 6v - 12$ - 假设所有顶点度数 ≥ 6，则 $2e = \sum \deg(v) \geq 6v$，矛盾 - 因此至少有一个顶点度数 ≤ 5

【Corollary 3】没有 3-回路的平面图

If a connected planar simple graph has e edges and v vertices with $v \geq 3$ and **no circuits of length 3** (即没有三角形), then:

$$> e \leq 2v - 4 >$$

Proof：- 没有 3-回路意味着每个区域的度数至少为 4 - $2e = \sum \mathrm{Deg}(R_i) \geq 4r \Rightarrow r \leq \frac{e}{2}$ - 代入欧拉公式：$e - v + 2 \leq \frac{e}{2} \Rightarrow e \leq 2v - 4$

一般形式：如果每个区域至少有 k 条边，则：

$$e \leq \frac{k(v-2)}{k-2}$$

【Example 4】证明 $K_{3,3}$ 是非平面图 $K_{3,3}$ 有 $v = 6$ 个顶点，$e = 9$ 条边。**K3,3 图形结构**（二部图，两组各 3 个顶点，组间完全连接）：

```
第一组:    a      b      c
            \    |    /
             \   |   /
第二组:        d   e   f

a 连接 d, e, f    (3 条边)
b 连接 d, e, f    (3 条边)
c 连接 d, e, f    (3 条边)
共 9 条边，任意平面画法必然有边交叉
```

因为 $K_{3,3}$ 是二部图，它**没有 3-回路**（没有三角形），所以适用 Corollary 3：检查：$e \le 2v-4 \Rightarrow 9 \le 2\times 6-4=8$? 不成立！因为 $9 \not\le 8$，所以 $K_{3,3}$ **不是平面图**。

【Example 5】正十二面体 (Dodecahedron) 正十二面体是一个平面图（可投影到平面），每个顶点度数为 3，每个区域的度数为 5。

```
正十二面体的平面投影示意图（简化）:

      a  b
     /    \
    c  d  e
   / \   / \
  f  g h  i
   \ /   \ /
    j  k  l
     \    /
      m  n

- 每个顶点度数 = 3
- 每个面（区域）由 5 条边围成
```

由关系：

$$2e=3v,\quad 2e=5r,\quad r=e-v+2$$

解得：$v=20,\ e=30,\ r=12$。（20 个顶点、30 条边、12 个面）

这是一个欧拉公式的应用范例：利用顶点度数、区域度数与欧拉公式联立求解。

9.7.3 Part 3: Kuratowski's Theorem (库拉托夫斯基定理)

基本概念

Elementary subdivision (基本细分)：在一条边上插入一个新顶点，将一条边分成两条边。**Homeomorphic** (同胚)：两个图 G_1 和 G_2 称为 **homeomorphic** 如果它们可以通过一系列的基本细分从同一个图得到。**基本细分 (Elementary Subdivision) 示例**：

```
原边:                    一次基本细分后:

a     b                  a  x  b

一条边 (a, b)            在边 (a, b) 中间插入新顶点 x
                         原边被分成两条边 (a, x) 和 (x, b)
```

同胚 (Homeomorphic) 示例——两个图可以通过一系列基本细分从同一个图得到：

```
原图 G:          G1（在边 a-b 上细分）:    G2（在边 c-d 上细分）:

a  b             a  x  b                  a  b


c  d             c  d                     c  y  d

G1 与 G2 同胚（都可以从 G 通过基本细分得到）
G1 与 G2 不一定同构，但它们是同胚的
```

【Theorem 2】Kuratowski's Theorem

A graph is **nonplanar** iff it contains a subgraph homeomorphic to $K_{3,3}$ or K_5.

一个图是非平面图 $\iff$ 它包含一个与 $K_{3,3}$ 或 K_5 同胚的子图。

意义：- K_5 和 $K_{3,3}$ 是**最小的非平面图** - 任何非平面图必定"包含"（通过基本细分变成的）K_5 或 $K_{3,3}$

【Example 6】用 Kuratowski 定理判断平面性 (1) 检查图是否包含与 K_5 同胚的子图：

- 找到 5 个顶点，它们之间有类似完全图 K_5 的连接结构
- 如果存在 $\to$ 非平面

(2) 检查图是否包含与 $K_{3,3}$ 同胚的子图：

- 找到可以分成两组各 3 个顶点的结构，组间完全连接
- 如果存在 $\to$ 非平面

(3) 通过删除"多余"的顶点（度为 2 的顶点可以"压缩"掉），检查剩下的结构是否为 K_5 或 $K_{3,3}$。**图示说明——反向执行基本细分（压缩度为 2 的顶点）**：

```
原图（可能非平面）:            压缩度 2 顶点后:


  a                            a
 / \                          / \
x  b       删除顶点 x        b
 \ /      ============>      |    → 压缩后是否为 K5 或 K3,3 的形式?
  c                            c
                               |
                               d

- 将度数为 2 的顶点"压缩"掉（反向基本细分）
- 检查压缩后的结构是否为 K5 或 K3,3
- 若是，则原图非平面
```

应用示例：(1) 检查一个图是否包含与 K5 同胚的子图：寻找 5 个顶点，它们之间通过路径（而非直接边）连接成 K5 的结构。(2) 检查一个图是否包含与 K3,3 同胚的子图：寻找两组各 3 个顶点，组间通过路径连接成完全二部图的结构。(3) 判断方法：

```
复杂图 → 移除度 2 的顶点（反向细分）→ 简化图
→ 检查简化图是否为 K5 或 K3,3
→ 如果是，则原图非平面
```

9.7.4 小结 (Summary)

概念	英文术语	关键点
平面图	Planar graph	可以画在平面上且边不交叉
区域	Region	边将平面划分成的区域（含一个无界区域）
欧拉公式	Euler's formula	$r=e-v+2$（连通平面图）
区域的度数	Degree of region	区域边界上的边数
边数上界	$e \le 3v-6$	$v \ge 3$ 的连通平面简单图
无三角形上界	$e \le 2v-4$	无 3-回路的平面图
库拉托夫斯基定理	Kuratowski's Theorem	非平面图 $\iff$ 包含 K_5 或 $K_{3,3}$ 同胚子图
两个经典非平面图	K_5, $K_{3,3}$	最小的非平面图

考点提示 (Exam Tips)

1. **欧拉公式** $r=e-v+2$ 必须牢记
2. 用 $e \le 3v-6$ 证明 K_5 非平面；用 $e \le 2v-4$ 证明 $K_{3,3}$ 非平面
3. 欧拉公式与握手定理结合使用（如 Example 3）
4. Kuratowski 定理：寻找与 K_5 或 $K_{3,3}$ 同胚的子图
5. "同胚"的含义：通过基本细分（插入度为 2 的顶点）得到
6. 注意 Corollary 2：平面图至少有一个顶点度数 ≤ 5

9.8 Chapter 10.8: Graph Coloring (图的着色)

9.8.1 Part 1: 地图着色问题 (Map Coloring)

问题的起源

Determining the **least number of colors** that can be used to color a map so that **adjacent regions** (相邻区域) never have the **same color**.

示例（各类型地图）：- 简单地图（4-5 个区域）：需要多少种颜色？- 随着地图复杂度的增加，所需颜色数如何变化？

Dual Graph (对偶图)

地图着色问题可以转化为图论问题。每个地图都可以用一个**对偶图 (dual graph)** 表示：

- 地图的每个区域 $\to$ 对偶图的一个**顶点**
- 两个区域有公共边界 $\to$ 对偶图中两个顶点之间有**边**
- **注意**：只在一个点接触（touch at only one point）不算相邻

构造示例——地图到对偶图的转化:

```
地图（五个区域）:                对偶图（五个顶点）:

                    A  B

A    B
                         C  D

C    D                   E    （E 为外部区域）

     E              ← 外部无限区域
   （外部）
```

```
转化规则:
- 每个区域（A, B, C, D, E）→ 一个顶点
- 相邻区域（共享边界）→ 顶点之间连边
- E 是地图外部的无限区域，也需要着色
```

注意：地图到对偶图的转化是**图着色问题的核心技巧**。

关键转化：Coloring a map is **equivalent** to coloring the vertices of its dual graph so that no two adjacent vertices have the same color.

9.8.2 Part 2: 图着色的基本概念

定义

术语	英文	定义
着色	**Coloring**	给每个顶点分配一个颜色，使得相邻顶点颜色不同
色数	**Chromatic number** $\chi(G)$	对图进行着色所需的**最少颜色数**

图示——简单图的着色示例:

```
(a) 路径图 P4 的着色:            (b) 偶圈 C4 的着色:

（红）（蓝）（红）（蓝）

需要 2 种颜色（交替着色）
(P4) = 2                       （蓝）（红）

                               需要 2 种颜色（交替着色）
                               (C4) = 2

(c) 奇圈 C5 的着色:              (d) 完全图 K4 的着色:

  / \                            \  /
 /   \                            \/
                               /\
 \   /                          /  \
  \ /

需要 3 种颜色（奇数圈无法交替）    需要 4 种颜色（每对不同顶点都相邻）
(C5) = 3                       (K4) = 4
```

9.8.3 Part 3: The Four Color Theorem (四色定理)

【Theorem 1】Four Color Theorem

The **chromatic number** of a **planar graph** is no greater than **four**.

任何平面图都可以用不超过 4 种颜色进行着色，使得相邻区域颜色不同。

历史：- 1850s：作为猜想提出 - 1976：由 **Haken 和 Appel** 通过计算机穷举搜索证明 - 这是数学史上第一个借助计算机证明的重要定理

四色定理示意图——平面地图可只用 4 种颜色着色:

```
一个平面地图的对偶图着色示例:

区域 A（红）          区域 B（蓝）

   A        B

   C        D
 （绿）     （黄）

   E （红）        ← 外部区域可复用 A 的颜色
                     （不相邻即可同色）

A(红), B(蓝), C(绿), D(黄), E(红)
只用了 4 种颜色，相邻区域颜色均不同
```

注意：四色定理只保证平面图　4。对于非平面图（如 K5），色数可以任意大。

重要局限：- 四色定理**只适用于平面图 (planar graphs)** - 非平面图的色数可以任意大（例如 K_n 的色数为 n）

9.8.4 Part 4: 一些简单图的色数

计算色数的方法

要证明一个图的色数为 n，需要：

1. **上界**：构造一个使用 n 种颜色的着色方案
2. **下界**：证明不能用 $n-1$ 种颜色（如存在 K_n 子图等）

常见图的色数

图	色数 $\chi(G)$	说明
仅含孤立顶点的图	1	所有顶点互不相邻
路径 (Path)	2	交替两种颜色即可
偶圈 C_n (n 为偶数)	2	交替两种颜色
奇圈 C_n (n 为奇数)	3	需要三种颜色（因为奇数个顶点无法交替）
完全图 K_n	n	每个顶点都与其他所有顶点相邻
二部图 (Bipartite graph)	2	二部图等价于 2-colorable
平面图	≤ 4	四色定理

常见图的色数图示:

```
(1) 孤立顶点图（ = 1）:

                 所有顶点互不相邻，1 种颜色足够

(2) 路径图 Pn（ = 2）:


   交替使用两种颜色即可

(3) 偶圈 C4（ = 2）与奇圈 C5（ = 3）:

  偶圈 C4:          奇圈 C5:
                          /   \

   交替着色即可            \     /   需要 3 种颜色


(4) 完全图 K4（ = 4）:


  /|\
      每对不同顶点都相邻，需要 4 种颜色
```

```
    \|/


(5) 二部图 K2,3（ = 2）：


    / \  / \
            两组内部无边，组间全连接，交替着色即可
```

特别重要：

A simple graph with chromatic number **2** is **bipartite**. A **connected** bipartite graph has chromatic number **2**. 色数为 2 的简单图就是二部图。

9.8.5 Part 5: 着色的应用 (Applications of Graph Coloring)

应用一：考试时间安排 (Scheduling Exams)

问题：大学如何安排考试，使得没有学生同时参加两门考试？**建模方法**：

1. 顶点 = 课程
2. 边 = 两门课有共同学生
3. 每种颜色 = 一个考试时间段
4. 图的一个着色方案 → 一个可行的考试安排

【Example】考试时间安排设有课程：1007, 3137, 3157, 3203, 3261, 4115, 4118, 4156 冲突对（有共同学生的课程对）：

```
1007-3137, 1007-3157, 3137-3157
1007-3203
1007-3261, 3137-3261, 3203-3261
1007-4115, 3137-4115, 3203-4115, 3261-4115
1007-4118, 3137-4118
1007-4156, 3137-4156, 3157-4156
```

步骤与图示：步骤 1：画出**冲突图**（conflict graph）——顶点是课程，边表示两门课有共同学生（不能同时考）：冲突边列表（共 15 条冲突边）：

```
1007 与所有其他 7 门课程冲突：
1007-3137, 1007-3157, 1007-3203, 1007-3261,
1007-4115, 1007-4118, 1007-4156

其他冲突：
3137-3157, 3137-3261, 3137-4115, 3137-4118, 3137-4156
3157-4156
3203-3261, 3203-4115
3261-4115
```

注意 1007 与所有其他课程都有冲突（必须单独一个时间段）。同时存在三角形冲突结构（如 3137-3157-4156 两两冲突）。

步骤 2：冲突图包含一个**完全二部量子图**——无法用 2 色着色剩余的 7 个顶点，因为存在三角形。

步骤 3：计算冲突图的**补图**（complementary graph）——补图中的边表示**可以**同时考试的课程对：

```
补图边（无共同学生的课程对，共 12 条）：
3137-3203
3157-3203, 3157-3261, 3157-4115, 3157-4118
3203-4118, 3203-4156
3261-4118, 3261-4156
4115-4118, 4115-4156
4118-4156
```

步骤 4：对冲突图进行着色——每种颜色代表一个考试时间段，最小色数即为最优时间段数：

```
冲突图中，1007 与所有顶点冲突 → 单独占一种颜色
剩余 7 个顶点包含三角形 3137-3157-4156 → 至少需要 3 种颜色
总色数 = 1 + 3 = 4

可行着色方案（4 种颜色 / 4 个时间段）：

时间段 1：{1007}                    （1007 与所有人都冲突）
时间段 2：{3137, 3203}              （3137 与 3203 无共同学生）
时间段 3：{3157, 3261, 4118}        （三者互不冲突）
时间段 4：{4115, 4156}              （4115 与 4156 无共同学生）
```


验证：同色课程对之间均无冲突边。4 为最小色数（因为 1007 必独占一色 + 三角形必占三色），故最少需要 4 个考试时间段。

最终结果：通过图着色找到需要的最少考试时间段数（即色数）。

应用二：动物园动物栖息地安排

问题：动物园中某些动物不能放在同一栖息地（因为捕食关系），如何安排？**建模方法**：

1. 顶点 = 动物
2. 边 = 两种动物不能在同一栖息地（如捕食关系）
3. 颜色 = 不同的栖息地
4. 图的一个着色方案 → 栖息地分配方案

9.8.6 小结 (Summary)

概念	英文术语	关键点
图着色	Graph coloring	相邻顶点不同色
色数	Chromatic number $\chi(G)$	最少颜色数
四色定理	Four Color Theorem	平面图 $\chi(G) \leq 4$
二部图	Bipartite graph	$\chi(G) = 2$
对偶图	Dual graph	地图 → 图着色的转化工具
考试安排	Exam scheduling	冲突图着色
K_n 的色数	$\chi(K_n) = n$	每对不同顶点都相邻
C_n 的色数	$\chi(C_n) = 2$ (n even), 3 (n odd)	奇圈需要 3 色

考点提示 (Exam Tips)

1. **色数的证明**：要同时证明上界（构造着色）和下界（证明不能更少）
2. C_n **的色数**：偶数 → 2，奇数 → 3（重要!）
3. **二部图 = 2-可着色图**，这是两者的等价关系
4. **四色定理**：只适用于平面图，非平面图色数可以任意大
5. **实际应用建模**：将实际问题转化为图着色问题，尤其考试安排
6. 区分**四色定理**（平面图色数 ≤ 4）与一般图的色数
7. 注意：K_n 的色数是 n，$K_{m,n}$（二部图）的色数是 2

第十章　Chapter 11: Trees (树)

10.1　Chapter 11.1: Introduction to Trees (树的基本介绍)

10.1.1　1. Definition of a Tree (树的定义)

Definition 1: A **tree** is a **connected undirected graph** with **no simple circuits**.

- 树是一个连通无向图，且没有简单回路（no simple circuits）。
- **Forest (森林):** An undirected graph with no simple circuits. 森林是没有简单回路的无向图，它的每个连通分量都是一棵树。

Notes: (1) Any tree must be a simple graph (任何树都必须是简单图，即没有平行边和自环). (2) Each connected component of a forest is a tree (森林的每个连通分量都是一棵树).

Example 1: Which graphs are trees?
判断以下哪个图是树：

```
a)              b)              c)
  a            a---b           a---b    d
 / \            |  |            |
b   c           c--d            c
```

- 图 a: 连通且无环 (connected, no circuits) –> **是树**
- 图 b: 有回路 (contains a cycle a-b-d-c-a) –> **不是树**
- 图 c: 不连通 (disconnected, 两个连通分量 a-b-c 和 d) –> **不是树**

关键检查点： 连通性 (connected) + 无简单回路 (no simple circuits)

10.1.2　2. Theorem 1: Unique Path Property (唯一路径定理)

Theorem 1: An undirected graph is a tree **if and only if** there is a **unique simple path** between any two of its vertices. (一个无向图是树当且仅当它的任意两个顶点之间存在唯一的一条简单路径。) **Proof (证明思路): (1) Forward direction (=>):** 假设 G 是树。

- G 是连通的（connected），所以任意两点间存在简单路径。
- 假设存在两条不同的简单路径，则这两条路径的并会形成回路，与树无环矛盾。因此路径唯一。

(2) Backward direction (<=): 假设任意两点间存在唯一简单路径。

- 连通性（connected）显然成立。
- 如果存在简单回路，则回路上任意两点就有两条不同路径，矛盾。因此无简单回路。

10.1.3　3. Rooted Trees (有根树)

在许多应用中，我们会指定树中的一个特殊顶点为 **root (根)**。

- 一旦指定了根，每条边的方向就被定为 **away from the root (远离根)**。
- 树 + 根 = **rooted tree (有根树)**，形成了一个有向图。

Example 2: 将无根树转化为有根树
选择不同的顶点作为根，会得到不同结构的有根树。

```
原无根树:        选择根后:
  a                a (root)
 / \              / \
b   c            b   c
 / \            /     \
d   e          d       e
```

选择不同的根会改变树的结构和层次关系。

10.1.4　4. Basic Terminology in Rooted Trees (基本术语)

Parent and Child (父节点与子节点)

- **Parent (父节点):** 对于非根顶点 v，有且只有一个顶点 u 存在从 u 到 v 的有向边，则 u 是 v 的 parent.
- **Child (子节点):** 若 u 是 v 的 parent，则 v 是 u 的 child.

Sibling (兄弟节点)

Vertices with the **same parent** are called **siblings**.（拥有相同父节点的顶点互为兄弟。）

Ancestor and Descendant (祖先与后代)

- **Ancestor (祖先):** The ancestors of a non-root vertex v are all the vertices in the path from the root to v.（从根到 v 的路径上的所有顶点都是 v 的祖先。）
- **Descendant (后代):** The descendants of v are all vertices that have v as an ancestor.（以 v 为祖先的所有顶点都是 v 的后代。）

Leaf, Internal Vertex, Root (叶节点、内部顶点、根)

- **Leaf (叶节点 / 树叶):** A vertex with **no children**.（没有子节点的顶点。）
- **Internal vertex (内部顶点):** A vertex that **has children**.（有子节点的顶点。）
- **Root (根):** The unique vertex with no parent.（唯一一个没有父节点的顶点。）

Subtree (子树)

The **subtree at vertex v** is the subgraph consisting of v and all its descendants and edges incident to these descendants.（以 v 及其所有后代构成的子图称为 v 的子树。）

10.1.5　5. Running Example: Jake's Pizza Shop Organization Tree

课件中使用 Jake's Pizza Shop 的人员组织树贯穿 11.1，用于直观展示 parent, child, sibling, ancestor, descendant, leaf, internal vertex, subtree 等有根树术语。

```
              Owner Jake (Level 0)
             /                    \
   Manager Brad (Level 1)    Chef Carol (Level 1)
                            /       |        \
                         Joyce     Chris      Max (Level 2)
                          (Level 2) (Level 2)    |
                                                 Len (Level 3)
```

该树中各术语的直观说明：

- **Root (根):** Owner Jake —唯一没有父节点的顶点。
- **Parent-Child (父子):** Owner Jake 是 Manager Brad 和 Chef Carol 的 parent；Chef Carol 是 Joyce, Chris, Max 的 parent；Max 是 Len 的 parent。
- **Siblings (兄弟):** Manager Brad 与 Chef Carol 互为 siblings（同为 Owner Jake 的孩子）；Joyce, Chris, Max 也互为 siblings（同为 Chef Carol 的孩子）。
- **Ancestors (祖先):** Len 的 ancestors 包括 Owner Jake, Chef Carol, Max。
- **Descendants (后代):** Chef Carol 的 descendants 包括 Joyce, Chris, Max, Len。
- **Leaf (叶):** Manager Brad, Joyce, Chris, Len —没有子节点的顶点。
- **Internal vertex (内部顶点):** Owner Jake, Chef Carol, Max —有子节点的顶点。
- **Subtree (子树):** 以 Chef Carol 为根的子树包含 Chef Carol, Joyce, Chris, Max, Len 及它们之间的所有边。

10.1.6　6. m-ary Trees and Binary Trees (m 叉树与二叉树)

Definition: A rooted tree is called an **m-ary tree** if every internal vertex has **no more than m** children.（每个内部顶点最多有 m 个子节点的有根树称为 m 叉树。）

- **Binary tree (二叉树):** m = 2 的 m 叉树。
- **Full m-ary tree (满 m 叉树):** 每个内部顶点恰好有 m 个子节点。

Ordered Rooted Tree (有序有根树)

Definition: An **ordered rooted tree** is a rooted tree where the children of each internal vertex are **ordered** (有序的). 在 ordered binary tree 中：

- **Left child (左子节点)** 和 **right child (右子节点)**
- **Left subtree (左子树)** 和 **right subtree (右子树)**

10.1.7　7. Nonisomorphic Trees (非同构树)

Example 3: 非同构树的计数

(1) How many nonisomorphic unrooted trees are there with n = 5? Solution: 一棵有 5 个顶点的树一定是连通无环的，且有 4 条边（Theorem 2）。n = 5 时共有 3 种非同构的无根树：

```
1)  o        2)  o        3)  o
    |            |            |
    o            o            o
    |           / \           |
    o          o   o          o
```

```
|                          |
o                          o
                            |
                            o
```

(2) How many nonisomorphic rooted trees are there with n = 5? Solution: 当考虑根的区别时，n = 5 共有 **9** 种非同构的有根树（比无根树更多，因为选择不同顶点作为根会产生不同的有根树结构）。

易错点：无根树同构与有根树同构的计数不同。相同的无根树结构，选择不同根可能产生不同的有根树。

10.1.8 8. Tree Properties and Theorems (树的性质与定理)

Theorem 2: A tree with n vertices has n-1 edges.
（一棵有 n 个顶点的树有 n-1 条边。）**Proof 1 (通过根):** 选择 r 为根，建立边与除 r 外的顶点之间的一一对应关系。每条边关联到它的末端顶点（非根），因为共有 n-1 个非根顶点，所以有 n-1 条边。

```
    r (root)
   / \
  a   b        边 (r,a) ↔ 顶点 a
 /   / \       边 (r,b) ↔ 顶点 b
c   d   e      边 (a,c) ↔ 顶点 c
               边 (b,d) ↔ 顶点 d
               边 (b,e) ↔ 顶点 e

每条边都有一个唯一的非根末端顶点，
5 条边对应 5 个非根顶点（r 除外）。
```

Proof 2 (通过平面图公式): 使用欧拉公式 $n - e + r = 2$（其中 r 是 face 数）。树是平面且连通的，且没有回路，所以 $r = 1$。代入得 $n - e + 1 = 2$，所以 $e = n - 1$。

Example 4: 已知顶点度数求叶节点数

A tree has two vertices of degree 2, one vertex of degree 3, three vertices of degree 4. How many leaves does this tree have? **Solution:** 设叶节点数为 x。总顶点数: $v = 2+1+3+x = 6+x$ 握手定理: $\sum deg(v) = 2e$ $2\times2+1\times3+3\times4+x\times1 = 4+3+12+x = 19+x = 2e$ $\Rightarrow e = \frac{19+x}{2}$ 由 Theorem 2: $e = v - 1$ $\frac{19+x}{2} = (6+x) - 1 = 5+x$ $19+x = 10+2x$ $x = 9$ **Answer:** 9 leaves.

Theorem 3: Full m-ary Tree Vertex Count

A **full m-ary tree** with i internal vertices contains $n = mi + 1$ vertices. **Proof:** 除根以外，每个顶点都是某个内部顶点的子节点。i 个内部顶点各有 m 个子节点，所以除了根外有 mi 个顶点。因此 $n = mi + 1$。

Theorem 4: Relationships Between n, i, l

对于一个 full m-ary tree:

已知条件	公式
已知 n	$i = (n-1)/m$, $l = \frac{(m-1)n+1}{m}$
已知 i	$n = mi + 1$, $l = (m-1)i + 1$
已知 l	$n = \frac{ml-1}{m-1}$, $i = \frac{l-1}{m-1}$

基础关系:

- $n = mi + 1$ （Theorem 3）
- $n = i + l$ （顶点由内部顶点 + 叶节点组成）

对于 full binary tree（满二叉树）: $l = i + 1$, $e = v - 1$

Example 5: Chain Letter Problem

A chain letter starts when a person sends a letter to 5 others. Each person who receives the letter either sends it to 5 new people or does not send it. Suppose 10000 people send out the letter before the chain ends and no one receives more than one letter. How many people receive the letter, and how many do not send it out? **Solution:** 这是一个 full 5-ary tree。

- $i = 10000$ (发送信件的人数 = 内部顶点数)
- $n = 5i + 1 = 50001$ (总人数)
- $n = i + l \implies l = n - i = 50001 - 10000 = 40001$
- 不发送信件的人数 = 叶节点数 $l = 40001$

Answer: 50001 people receive the letter, 40001 do not send it out.

Level and Height (层数与高度)

- **Level (层数):** The level of vertex v in a rooted tree is the length of the unique path from the root to v. （从根到该顶点的路径长度。）根在第 0 层。
- **Height (高度):** The height of a rooted tree is the **maximum** of the levels of all vertices. （所有顶点层数的最大值。）

Example hierarchy (在 Jake's Pizza Shop 树上标注 levels):


```
Level 0:         Owner Jake
                /         \
Level 1:  Manager Brad     Chef Carol
                          /    |    \
Level 2:            Joyce  Chris  Max
                                    |
Level 3:                           Len
```

Balanced Tree (平衡树)

Book definition: A rooted m-ary tree of height h is called **balanced** if all its leaves are at levels **h or h-1**. **For binary trees (更常见的定义):** A tree is balanced if:

1. The left and right subtrees' heights differ by at most one, AND
2. The left subtree is balanced, AND
3. The right subtree is balanced

Theorem 5: Maximum Number of Leaves

Theorem 5: There are at most m^h leaves in an **m-ary tree** of height h. （高度为 h 的 m 叉树最多有 m^h 个叶节点。）**Proof (induction on h):**

- Base case h=1: 最多 m 个叶节点，即 m^1。
- Inductive step: 假设所有高度小于 h 的 m 叉树满足条件。T 的高度为 h，根的子树高度均为 h-1，每棵子树最多有 m^{h-1} 个叶节点，总共最多 $m \cdot m^{h-1} = m^h$ 个叶节点。

```
      root (高度 h)
    /   |    ...   \
  T1    T2          Tm
 (h-1) (h-1)       (h-1)
  m^{h-1} leaves each


 Total   m × m^{h-1} = m^h
```

Corollary (推论)

If an **m-ary tree** of height h has l leaves, then:

$$h \geq \lceil \log_m l \rceil$$

If the m-ary tree is **full and balanced**, then:

$$h = \lceil \log_m l \rceil$$

10.1.9 9. Every Tree is Bipartite (树是二分图)

Question: Every tree is a bipartite graph? **Yes. Proof:** 选择根并着色为红色（red），所有奇数层的顶点着蓝色（blue），所有偶数层的顶点着红色（red）。由于树中所有边都连接不同层（父子之间），所以没有边连接同色顶点。因此树可以用两种颜色着色。

10.1.10 10. 小结与考点提示

考试重点:

1. **Tree 的定义**—连通 + 无简单回路
2. **Theorem 1** —任意两点间唯一简单路径
3. **Theorem 2** —n 个顶点，$n - 1$ 条边（最重要的性质之一）
4. **Theorem 3 & 4** —full m-ary tree 中 n, i, l 的关系（公式要熟记）
5. **Level 和 Height** 的概念及计算
6. **Balanced tree** 的定义
7. **Tree is bipartite** —2-colorable
8. **Example 4** 类型题：已知度数求叶节点数——使用握手定理 + $e = n - 1$
9. **Example 5** 类型题：full m-ary tree 应用题
10. **术语辨析**—前序考试常见：identify root, leaves, internal vertices, parent-child relationships

易错点:

- 不要混淆"level" 和"height" —level 从根（0）开始计数，高度是最大 level
- Full m-ary tree balanced m-ary tree
- 区分 rooted tree 和 unrooted tree 的同构计数

10.2 Chapter 11.2: Applications of Trees (树的应用)

10.2.1 Overview (概述)

本节讨论树的三个主要应用：

1. **Binary Search Trees (二叉搜索树)** —高效查找和插入
2. **Decision Trees (决策树)** —通过一系列决策解决问题
3. **Prefix Codes (前缀编码)** —高效编码字符（Huffman Coding）

10.2.2 1. Binary Search Trees (二叉搜索树, BST)

The Problem of Search (查找问题)

如何在一个列表中存储项目，以便能够快速定位某个项目？二叉搜索树提供了高效的解决方案。

The Concept of Binary Search Tree

Binary search tree 是一种 ordered rooted binary tree，满足：

- 每个顶点包含一个 **distinct key value (唯一键值)**
- 键值之间可以比较大小（使用"greater than" 和"less than"）
- 每个顶点的键值 **大于其左子树中所有键值**（所以左子树中所有节点都小于根）
- 每个顶点的键值 **小于其右子树中所有键值**（所以右子树中所有节点都大于根）

  简而言之：左小右大。

Constructing a Binary Search Tree (构建二叉搜索树)

BST 的形状取决于键值及其插入顺序。**Example:** Insert elements 'J', 'E', 'F', 'T', 'A' in that order.

```
Step 1: 插入 'J' 作为根
        J

Step 2: 插入 'E', 'E' < 'J', 向左走, 成为左子节点
        J
       /
      E

Step 3: 插入 'F', 'F' < 'J' 向左, 'F' > 'E' 向右, 成为右子节点
        J
       /
      E
       \
        F

Step 4: 插入 'T', 'T' > 'J' 向右, 成为右子节点
        J
       / \
      E   T
       \
        F

Step 5: 插入 'A', 'A' < 'J' 向左, 'A' < 'E' 向左, 成为左子节点
        J
       / \
      E   T
     / \
    A   F
```

不同的插入顺序产生不同的树形：如果按顺序'A', 'E', 'F', 'J', 'T' 插入：

```
A
 \
  E
   \
    F
     \
      J
       \
        T
```


这退化成了链表（degenerate tree），查找效率会降低。这说明 **BST 的形状依赖于插入顺序**。**更复杂的 BST 示例**：按顺序 J, E, T, A, H, M, K, P 插入后，再插入 D, B, L, Q, S, V, Z，得到以下树形：

```
          J
         / \
        E   T
       / \ / \
      A  H M  V
       \ / \   \
       D K  P   Z
      / \   \
     B   L   Q
              \
               S
```

在 BST 中查找'F' 的路径 (课件演示)：F < J → 向左到 E （第 1 次比较）F > E → 向右到 H （第 2 次比较）F < H → H 无左子节点 → 查找结束，**'F' 不在树中**（第 3 次比较）该搜索路径说明 BST 查找的过程：每次比较将搜索范围缩小到左子树或右子树。

Binary Search Algorithm (查找算法)

Algorithm 1: Locating and Adding Items to a Binary Search Tree

```
Procedure insertion (T: binary search tree, x: item)
v := root of T
While v  null and label(v)  x
begin
    if x < label(v) then
        if left child of v  null then v := left child of v
        else add new vertex as a left child of v and set v := null
    else
        if right child of v  null then v := right child of v
        else add new vertex as a right child of v and set v := null
end
if root of T = null then add a vertex r to the tree and label it with x
else if v is null or label(v)  x then label new vertex with x
```

核心思路：从根开始比较，小于当前节点向左走，大于当前节点向右走，直到找到空位插入或找到目标值。

Computational Complexity (计算复杂度)

- 为形成完整的二叉树进行分析，通常给 BST 添加无标记的外部节点（external/unlabeled vertices），形成 **full binary tree**。例如：

```
原始 BST:          添加外部节点后:
    J                  J
   / \               /   \
  E   T             E     T
 / \               / \   / \
A   H             A   H ?   ?
 \                 \  / \  / \
  D                D ?  ?? ?  ?
 /
B                 / \
                  B ?
                 / \
                ? ?

每个 ? 代表一个外部(空)节点, 使树变为 full binary tree
```

- 添加一个新项目所需的最多比较次数 = 从根到叶的最长路径长度（即树的高度）。
- 如果 BST 是 **balanced (平衡的)**，则定位或添加一个项目最多需要 $\lceil \log_2(n+1) \rceil$ 次比较。
- 在最坏情况下（degenerate tree），复杂度为 $O(n)$。

10.2.3 2. Decision Trees (决策树)

Definition: A **decision tree** is a rooted tree where each internal vertex corresponds to a **decision**, with a subtree at these vertices for each possible outcome of the decision. （决策树是一种有根树，每个内部顶点对应一个决策，每个可能的结果对应一个子树。）

Example: Counterfeit Coin Detection (假币检测)

Problem: 有 8 枚硬币，其中 1 枚是假币（可能较轻或较重）。用天平称重找出假币。**决策树结构**：

```
第一轮: 称 (1,2,3)
```

```
                    vs
                  (4,5,6)


   左边轻                平衡                右边轻
(1,2,3)有轻币        (7,8)有假币        (4,5,6)有轻币


 第二轮:              第二轮:             第二轮:
 称 1 vs 2            称 7 vs 1           称 4 vs 5


1轻    3轻          7轻    7重          4轻    6轻
=1     =3           =7     =7           =4     =6


2轻=2               不可能              5轻=5

                   8是假币
```

说明：左侧三列对应 (1,2,3) 有轻币的分支（称 1 vs 2 找出轻币）；中间列对应第一次平衡的情况（需另找标准币 1 与 7 比较来判断 7/8 中谁是假币及轻重）；右侧三列对应 (4,5,6) 有轻币的分支（对称于左分支）。

决策树的每个内部节点是一个称量决策，每个叶节点是一个结论（哪枚硬币是假币，轻还是重）。

10.2.4 3. Prefix Codes (前缀编码)

The Problem (编码问题)

如何用 bit strings（比特串）编码英文字母?

- **等长编码 (Fixed-length):** 每个字母用相同长度的 bit 串编码（如 ASCII）。简单但效率低。
- **变长编码 (Variable-length):** 常用字母用短编码，不常用字母用长编码，提高效率。但需要确保编码可以被唯一解码。

问题示例：若 e: 0, a: 1, t: 01 则 0101 可以解释为：eat, tea, eaea, tt —有歧义!

The Concept of Prefix Codes (前缀码概念)

Definition: A **prefix code** is a code where the bit string for a letter **never occurs as the first part** of the bit string for another letter.（前缀码：任何一个字符的编码都不是另一个字符编码的前缀。）**Example:** e: 0, a: 10, t: 110

- 0101 只能解释为 e a e a（唯一解码）
- 因为 0（e）不是 10（a）的前缀？实际上这里搞混了——让我们澄清：如果 e=0, a=10, t=110，那么 0110 只能是 e t，因为 0 后如果跟 1 就继续读。前缀码保证了唯一可解码性。

关键性质：前缀码保证唯一可解码性 (uniquely decodable)。

Constructing Prefix Codes Using Binary Trees

使用二叉树构造前缀码：

- 每条左边缘标记为 **0**
- 每条右边缘标记为 **1**
- 叶节点标记为被编码的字符
- 每个字符的编码 = 从根到叶的路径上的边标签拼接

```
          root
         /    \
        0      1
       / \    / \
      0   1  0   1
     / \  |  |  / \
    0   1 ?  ? 1   1
    |   |      |   |
    ?   ?      ?   ?
  (000)(001)  (101)(111)
     ↑
   "01" 也是叶节点（路径：root→0→1）
```


其中几个叶节点对应的编码：

- 路径 0→0→0 的叶编码为 **000**
- 路径 0→0→1 的叶编码为 **001**
- 路径 0→1 的叶编码为 **01**
- 路径 1→0→1 的叶编码为 **101**
- 路径 1→1→1 的叶编码为 **111**

每个字符的编码 = 从根到该字符叶节点的路径上所有边标签（0 或 1）的拼接。由于没有编码是另一个编码的前缀，所以可以唯一解码。

Huffman Coding (霍夫曼编码)

Problem: 如何根据字符出现频率生成最有效的前缀码? **目标**：最小化 $\sum_{i=1}^{26} w_i \cdot l_i$ 其中 w_i 是字符频率，l_i 是编码长度。**General problem:** 树 T 有 t 个叶节点，$w_1, w_2, \ldots, w_t$ 是权重，$l_i = l(w_i)$ 是路径长度。树的权重定义为：

$$W(T) = \sum_{i=1}^{t} w_i \cdot l_i$$

目标是 **minimize** $W(T)$。

Huffman Coding Algorithm (霍夫曼编码算法)

Algorithm 2: Huffman Coding

```
Procedure Huffman (C: symbols a_i with frequencies w_i, i=1,...,n)
F := forest of n rooted trees, each consisting of the single vertex a_i
      and assigned weight w_i

While F is not a tree
begin
    Replace the rooted trees T and T' of least weights from F
    (with w(T) >= w(T')) with a tree having a new root that has
    T as its left subtree and T' as its right subtree.
    Label the new edge to T with 0 and the new edge to T' with 1.
    Assign w(T) + w(T') as the weight of the new tree.
end
```

核心思路：每次合并两个权重最小的树，形成新树，直到所有树合并为一棵。

Example 1: Huffman Coding Example

Use Huffman coding to encode the following symbols with frequencies:

- A: 0.08, B: 0.10, C: 0.12, D: 0.15, E: 0.20, F: 0.35

Solution (Step-by-Step): Step 1: 初始森林：6 棵单节点树

```
A(0.08)  B(0.10)  C(0.12)  D(0.15)  E(0.20)  F(0.35)
```

Step 2: 合并 A(0.08) 和 B(0.10) -> 新树 AB(0.18)，A 为左子树（标记 0），B 为右子树（标记 1）

```
    AB(0.18)   C(0.12)   D(0.15)   E(0.20)   F(0.35)
    /    \
  0/      \1
 A(0.08) B(0.10)
```

Step 3: 合并 C(0.12) 和 D(0.15) -> 新树 CD(0.27)

```
    AB(0.18)     CD(0.27)    E(0.20)    F(0.35)
    /    \       /    \
  0/      \1   0/      \1
 A(0.08) B(0.10) C(0.12) D(0.15)
```

Step 4: 合并 AB(0.18) 和 E(0.20) -> 新树 ABE(0.38)

```
       ABE(0.38)          CD(0.27)    F(0.35)
       /      \           /    \
     0/        \1       0/      \1
   AB(0.18)   E(0.20)  C(0.12) D(0.15)
   /    \
 0/      \1
A(0.08) B(0.10)
```

Step 5: 合并 CD(0.27) 和 F(0.35) -> 新树 CDF(0.62)

```
      ABE(0.38)         CDF(0.62)
      /      \          /      \
    0/        \1      0/        \1
  AB(0.18)   E(0.20) CD(0.27) F(0.35)
  /    \            /   \
 0/      \1        0/     \1
A(0.08) B(0.10)  C(0.12) D(0.15)
```

Step 6: 合并 ABE(0.38) 和 CDF(0.62) -> 最终树 (1.00)

```
                  root(1.00)
                 /         \
               0/           \1
          ABE(0.38)       CDF(0.62)
          /      \        /      \
        0/        \1    0/        \1
      AB(0.18)   E(0.20) CD(0.27) F(0.35)
      /    \            /   \
    0/      \1        0/     \1
   A(0.08) B(0.10)  C(0.12) D(0.15)
```

结果编码（从根到叶的路径标签）:

字符	编码	长度
A	000	3
B	001	3
E	01	2
C	100	3
D	101	3
F	11	2

Average number of bits per character: $0.08\times3+0.10\times3+0.12\times3+0.15\times3+0.20\times2+0.35\times2 = 0.24+0.30+0.36+0.45+0.40+0.70$ $= 2.45$ bits per character

10.2.5 4. 小结与考点提示

考试重点:

1. **Binary Search Tree**
- BST 的定义：左小右大
- 构建过程（不同插入顺序产生不同形状）
- 查找/插入的时间复杂度：平衡时 $O(\log n)$，最坏 $O(n)$
- **可能考：给定一组键值，画出 BST 的构造过程**

1. **Decision Tree**
- 概念：每个内部节点是一个决策，叶节点是结论
- 典型应用：假币检测、排序问题
- **可能考：设计决策树解决某个问题**

1. **Prefix Codes**
- 前缀码定义和性质（唯一可解码）
- 用二叉树构造前缀码的方法
- **Huffman Coding：核心算法（必须掌握）**
- 给定频率表，完整地执行 Huffman 算法步骤
- 计算每个字符的编码和平均编码长度
- **Huffman 算法是 greedy algorithm 的经典例子**

易错点:

- Huffman 编码的边标签：**左 0 右 1**（或相反，但需保持一致）
- 合并时权重小的树放哪边？算法要求 w(T) >= w(T')，T' 是权重较小的树
- BST 中，左子树所有节点 < 根 < 右子树所有节点（是严格的大小关系）
- Prefix code 和 non-prefix code 的区别要能通过例子判断

10.3 Chapter 11.3: Tree Traversal (树的遍历)

10.3.1 Overview (概述)

Traversal algorithm (遍历算法): A procedure for systematically visiting every vertex of an ordered rooted tree. 树的遍历有三种基本方式，都是**递归定义**的:

1. **Preorder traversal (前序遍历)**
2. **Inorder traversal (中序遍历)**
3. **Postorder traversal (后序遍历)**

10.3.2 1. Preorder Traversal (前序遍历)

Definition

Let T be an ordered tree with root r.

- If T has only r, then r is the preorder traversal.
- Otherwise:

1. **Visit r** (先访问根)
2. Traverse T1 in preorder (递归前序遍历左子树)
3. Traverse T2 in preorder (递归前序遍历右子树) ...
4. Traverse Tn in preorder

For Binary Trees (二叉树的前序遍历)

1. **Visit the root.** (访问根)
2. **Visit the left subtree**, using preorder. (前序遍历左子树)
3. **Visit the right subtree**, using preorder. (前序遍历右子树)

记忆口诀：**根 -> 左 -> 右**

Example: Preorder Traversal

```
      'J'
     /   \
   'E'   'T'
  /   \    \
'A'  'H'   'M'
             \
             'Y'
```

Preorder order: J, E, A, H, T, M, Y **解析:**

1. 访问 J (根)
2. 前序遍历左子树: 访问 E -> 前序遍历 E 的左子树 (A) -> 前序遍历 E 的右子树 (H)
3. 前序遍历右子树: 访问 T -> 前序遍历 T 的左子树 (空) -> 前序遍历 T 的右子树 -> 访问 M -> 前序遍历 M 的右子树 (Y)

10.3.3 2. Inorder Traversal (中序遍历)

Definition

Let T be an ordered tree with root r.

- If T has only r, then r is the inorder traversal.
- Otherwise:

1. Traverse T1 in inorder (递归中序遍历左子树)
2. **Visit r** (访问根)
3. Traverse T2 in inorder (递归中序遍历右子树) ...
4. Traverse Tn in inorder

For Binary Trees (二叉树的中序遍历)

1. **Visit the left subtree**, using inorder. (中序遍历左子树)
2. **Visit the root.** (访问根)
3. **Visit the right subtree**, using inorder. (中序遍历右子树)

记忆口诀：**左 -> 根 -> 右**

Example: Inorder Traversal

```
      'J'
     /   \
   'E'   'T'
  /   \    \
'A'  'H'   'M'
             \
             'Y'
```

Inorder order: A, E, H, J, T, M, Y **解析:**

1. 中序遍历 J 的左子树: 中序遍历 E 的左子树 (A) -> 访问 E -> 中序遍历 E 的右子树 (H)
2. 访问 J

3. 中序遍历 J 的右子树: 中序遍历 T 的左子树 (空) -> 访问 T -> 中序遍历 T 的右子树 -> 中序遍历 M 的左子树 (空) -> 访问 M -> 中序遍历 M 的右子树 (Y)

对于二叉搜索树 (BST)，中序遍历结果就是 **升序排列**。

10.3.4 3. Postorder Traversal (后序遍历)

Definition

Let T be an ordered tree with root r.

- If T has only r, then r is the postorder traversal.
- Otherwise:

1. Traverse T1 in postorder (递归后序遍历左子树)
2. Traverse T2 in postorder (递归后序遍历右子树) ...
3. Traverse Tn in postorder
4. **Visit r** (最后访问根)

For Binary Trees (二叉树的后序遍历)

1. **Visit the left subtree**, using postorder. (后序遍历左子树)
2. **Visit the right subtree**, using postorder. (后序遍历右子树)
3. **Visit the root.** (访问根)

记忆口诀：**左 -> 右 -> 根**

Example: Postorder Traversal

```
      'J'
     /   \
   'E'   'T'
  /   \    \
'A'  'H'   'M'
             \
             'Y'
```

Postorder order: A, H, E, Y, M, T, J

遍历顺序标注对比 (Annotated Traversal Order Comparison)

同一棵树在三种遍历下的访问顺序（节点上的数字表示第几个被访问）：**Preorder (前序: 根 -> 左 -> 右)**

```
        J
      /   \
    E     T
   /  \     \
  A    H     M
                \
                 Y
```

Inorder (中序: 左 -> 根 -> 右)

```
        J
      /   \
    E     T
   /  \     \
  A    H     M
                \
                 Y
```

Postorder (后序: 左 -> 右 -> 根)

```
        J
      /   \
    E     T
   /  \     \
  A    H     M
                \
                 Y
```

解析:

1. 后序遍历 J 的左子树: 后序遍历 E 的左子树 (A) -> 后序遍历 E 的右子树 (H) -> 访问 E
2. 后序遍历 J 的右子树: 后序遍历 T 的右子树 -> 后序遍历 M 的右子树 (Y) -> 访问 M -> 访问 T
3. 访问 J

Example 1: 对一般有序树的遍历

对于下面的 ordered rooted tree，分别给出三种遍历顺序：

```
      a
     / \
    b   c
   /   / \
  d   e   f
     /   / \
    g   h   i
```

Preorder (前序): a, b, d, c, e, g, f, h, i **Inorder (中序):** d, b, a, e, g, c, h, f, i **Postorder (后序):** d, b, g, e, h, i, f, c, a **(注：对于一般的 ordered tree（非二叉树），中序遍历的定义有所不同——先遍历 T1，然后访问根，再遍历 T2..Tn。但上面的例子中，树的结构允许我们根据自然顺序遍历。)**

10.3.5 4. Infix, Prefix, and Postfix Notation (中缀、前缀和后缀表达式)

Binary Expression Tree (二叉树表达式树)

A special kind of binary tree representing arithmetic expressions:

1. Each **leaf** contains a single **operand** (操作数)
2. Each **nonleaf node** contains a single **operator** (运算符)
3. The left and right subtrees represent subexpressions to be evaluated before applying the operator at the root

Example 2: 构造表达式树

What is the ordered tree that represents the expression:

$$\frac{x}{a} + 3 \times \ln(x+1)$$

Solution:

```
          +
         / \
        /   \
       /     \
      /       \
     /         \
    ÷           ×
   / \         / \
  x   a       3  ln
                  |
                  +
                 / \
                x   1
```

注意：运算符在内部节点，操作数在叶节点。运算优先级决定了树的结构（加减在较高层，乘除在较低层）。

Infix Form (中缀形式)

The expression obtained by an **inorder traversal** (中序遍历) of the binary tree, with parentheses added for clarity. 对于上面的表达式树，中序遍历加上括号：

$$(x/a) + (3 \times \ln(x+1))$$

中序遍历结果加上括号就是标准的数学表达式格式。

Prefix Form (前缀形式 / Polish Notation)

The expression obtained by a **preorder traversal** (前序遍历) of the binary tree. **特点**：运算符在操作数之前。不需要括号，无歧义。对于上面的表达式树：

```
+ / x a × 3 ln + x 1
```

Evaluation method: 从右向左扫描。

Postfix Form (后缀形式 / Reverse Polish Notation)

The expression obtained by a **postorder traversal** (后序遍历) of the binary tree. **特点**：运算符在操作数之后。不需要括号，无歧义。对于上面的表达式树：

```
x a / 3 x 1 + ln × +
```

Evaluation method: 从左向右扫描，使用栈 (stack) 计算。

三种形式的对比

遍历方式	所得表达式	运算符位置	特点
Inorder (中序)	Infix (中缀)	操作数之间	需要括号确定优先级
Preorder (前序)	Prefix (前缀)	操作数之前	从右向左求值，无括号
Postorder (后序)	Postfix (后缀)	操作数之后	从左向右求值，无括号

10.3.6 5. Evaluating Expression Trees (求值)

在表达式中，树中节点所在的层次决定了运算的优先级：

- **较高层（靠近根）的运算后执行**（即优先级较低）
- **较低层的运算先执行**（即优先级较高）
- **根节点的运算是最后执行的**

Example: Evaluate Expression Tree

```
        '*'
       /   \
     '-'   '/'
    /  \   / \
  '8'  '5' '+' '3'
          / \
        '4' '2'
```

Prefix form: `* - 8 5 / + 4 2 3` 求值（从右向左）：

1. 遇到 3 → 压入栈
2. 遇到 2 → 压入栈
3. 遇到 4 → 压入栈
4. 遇到 + → 弹出 4 和 2 → 4+2=6 → 压入 6
5. 遇到 / → 弹出 6 和 3 → 6/3=2 → 压入 2
6. 遇到 5 → 压入栈
7. 遇到 8 → 压入栈
8. 遇到 - → 弹出 8 和 5 → 8-5=3 → 压入 3
9. 遇到 → 弹出 *3* 和 *2* → *3*2=6

Answer: 6 Postfix form: `8 5 - 4 2 + 3 / *` 求值（从左向右）：

1. 8 → 压入栈
2. 5 → 压入栈
3. - → 弹出 8 和 5 → 8-5=3 → 压入 3
4. 4 → 压入栈
5. 2 → 压入栈
6. + → 弹出 4 和 2 → 4+2=6 → 压入 6
7. 3 → 压入栈
8. / → 弹出 6 和 3 → 6/3=2 → 压入 2
9. → 弹出 *3* 和 *2* → *3*2=6

Answer: 6

10.3.7 6. 小结与考点提示

考试重点：

1. **三种遍历的递归定义和顺序**

- Preorder: **根 -> 左 -> 右**
- Inorder: **左 -> 根 -> 右**
- Postorder: **左 -> 右 -> 根**
- 必须能熟练写出任意给定树的三种遍历结果

1. **已知遍历结果还原树**

- 已知 Preorder + Inorder → 唯一确定二叉树
- 已知 Postorder + Inorder → 唯一确定二叉树
- **典型考题：给出两种遍历结果，要求还原二叉树**

1. **Infix / Prefix / Postfix 表达式**

- 理解 binary expression tree 的结构
- 能从表达式构造表达式树
- 能从表达式树写出三种形式的表达式

- 能对 Prefix / Postfix 表达式求值

1. **Prefix 和 Postfix 的求值方法**

- Prefix: **从右向左**扫描，遇操作数压栈，遇运算符弹出两个操作数计算后压栈
- Postfix: **从左向右**扫描，遇操作数压栈，遇运算符弹出两个操作数计算后压栈

易错点：

- 后序遍历时，操作数的顺序和原表达式一致（都是从左到右），改变的是运算符的位置
- 从表达式树写 Prefix/Postfix 时，注意左右子树的顺序不能颠倒
- 用栈求值时，注意第二个弹出操作数的位置（先弹出的是右操作数，后弹出的是左操作数）

10.4 Chapter 11.4: Spanning Trees (生成树)

10.4.1 1. Definition of Spanning Tree (生成树的定义)

Definition 1: Let G be a simple graph. A **spanning tree** of G is a **subgraph** of G that is a **tree** containing **every vertex** of G.（生成树是图 G 的一个子图，它是一棵树，且包含 G 的所有顶点。）**Key properties:**

- 包含原图的所有顶点 (contains every vertex)
- 是树（连通且无环）
- 是原图的子图
- 边数 = 顶点数 - 1 ($e = n - 1$)

Example:

```
原图:                生成树之一:
  A---B---C            A---B---C
  |   |   |                |   |
  D---E---F            D---E---F
```

多个生成树 (Multiple Spanning Trees)

一个连通图通常有多个不同的生成树。以上图为例，可以移除不同的边得到不同的生成树：

```
原图:                  生成树 1 (移除 A-D, C-F):   生成树 2 (移除 B-E, C-F):
  A---B---C              A---B---C                  A---B---C
  |   |   |                  |                      |
  D---E---F              D---E---F                  D---E---F
                         边: A-B, B-C, B-E,         边: A-B, B-C, A-D,
                             D-E, E-F                   D-E, E-F
```

注意：不同的删除回路方式会得到不同的生成树，但所有生成树都包含 n-1=5 条边。

10.4.2 2. The Problem of Spanning Trees

为什么要研究生成树？考虑一个道路系统：如果道路太多（有环），维护成本高；但如果道路太少（不连通），则无法到达所有地方。生成树在**连通性**和**最小边数**之间取得平衡。**Example:**

```
原始道路网络 (有环):          生成树 (无环连通):
    B---C                       B---C
   /|  /|\                      |  /
  / | / | \                     | /
 A--E--D--F                    A--E--D--F
  \ | /                          \ |
   \|/                            \|
    G                              G
```

生成树移除了冗余道路，但仍保证所有节点可达。

10.4.3 3. Theorem 1: Connected iff Spanning Tree Exists

Theorem 1: A simple graph is **connected** if and only if it has a **spanning tree**.（简单图是连通的当且仅当它存在一棵生成树。）

Proof: (1) Forward (=>): 若 G 有生成树 T。

- T 包含 G 的所有顶点。
- T 中任意两点间存在路径（因为树是连通的）。
- 由于 T 是 G 的子图，所以 G 中任意两点间也存在路径。
- 因此 G 是连通的。

(2) Backward (<=): 若 G 是连通的。

- 不断删除 G 中简单回路上的边，直到没有回路为止。
- 由于每次删除边时保持连通性，最终得到的子图是连通的且无回路，即为生成树。

重要推论：每个连通图至少有一个生成树。

10.4.4 4. Algorithms for Constructing Spanning Trees

Method 1: Remove Edges from Circuits (删除回路法)

基于 Theorem 1：不断找出回路并删除回路上的一条边，直到无回路。

Method 2: Add Edges (添加边法) —两种经典搜索算法

10.4.5 5. Depth-First Search (DFS，深度优先搜索)

DFS is also called **backtracking (回溯法)**。

Algorithm Procedure

1. **任意选一个顶点**作为根 (root)
2. **形成一条路径**：从这个顶点开始，不断添加边，每条新边连接当前路径的最后一个顶点和一个**未访问过**的顶点
3. **继续添加边**直到无法继续（到达死胡同）
4. 如果路径已经覆盖所有顶点 → 该路径就是生成树
5. 如果未覆盖所有顶点 → 回退到路径中的前一个顶点，尝试从那里走新的路径
6. 重复此过程直到所有顶点都被访问

DFS 的核心思想：先尽可能深地探索一条路径，走不通时回溯，再探索下一条路径。

Example 2: DFS 寻找生成树

Graph:

```
a --- b --- c
|     |     |
d --- e --- f
|     |     |
g --- h --- i --- j
                |
                k
```

DFS Process (文字描述):

1. 从 a 开始: a -> b -> c -> f -> i -> j -> k (到达叶节点，无法继续)
2. 回溯到 j: 无未访问邻居，继续回溯
3. 回溯到 i: 访问未访问的 h -> 回溯到 h -> 访问 g -> 回溯到 g -> 访问 d -> 回溯到 d -> 访问 e
4. 所有顶点已访问

得到的生成树根为 a:

```
    a
    |
    b
    |
    c
    |
    f
    |
    i
   / \
  h   j
  |   |
  g   k
  |
  d
  |
  e
```

10.4.6 6. Breadth-First Search (BFS，广度优先搜索)

Algorithm Procedure

1. **任意选一个顶点**作为根，添加所有与该顶点关联的边
2. 这些新顶点成为第 1 层的顶点（任意排序）
3. **按顺序访问第 1 层的每个顶点**，对每个顶点，添加与该顶点关联的、**不会产生回路**的边
4. 这些新顶点成为第 2 层
5. 重复直到所有顶点都被加入树中

BFS 的核心思想：一层一层地向外扩展，先访问离根最近的所有顶点。

Example 3: BFS 寻找生成树

Graph:

```
a --- b --- c --- l
|     |     |     |
d --- e --- f --- m
|     |     |
g --- h --- i --- j
                |
                k
```

BFS Process (文字描述):

```
Level 0:    a (根)
Level 1:    b, d (a的邻居)
Level 2:    c, e, g (b和d的邻居)
Level 3:    f, h, l (c, e, g的邻居)
Level 4:    i, m (f, h, l的邻居)
Level 5:    j (i的邻居)
Level 6:    k (j的邻居)
```

得到的 BFS 生成树:

```
        a (L0)
       / \
      b   d (L1)
     /   / \
    c   e   g (L2)
   /   /     \
  l   f       h (L3)
  |   |       |
  m   i       ... (L4)
      |
      j (L5)
      |
      k (L6)
```

对比：DFS vs BFS

特性	DFS	BFS
搜索策略	深度优先（纵向）	广度优先（横向）
数据结构	栈（stack）	队列（queue）
生成树形状	狭长（长路径）	扁平（宽而浅）
应用	回溯法、拓扑排序	最短路径（无权图）

10.4.7 7. Backtracking Scheme (回溯法)

Backtracking is DFS applied to decision trees where each internal vertex represents a decision and each leaf represents a possible solution. 有些问题只能通过穷举搜索（exhaustive search）来解决，backtracking 提供了一种系统化的搜索方法。

Application 1: Graph Coloring (图着色问题)

Example 4: 能否用 3 种颜色对以下图着色?

```
a --- b
|     |
c --- d --- e
```

Backtracking 过程（文字描述）:

1. a 着红色 (red)
2. b 着蓝色 (blue)
3. c 尝试红色 → 不行（与 a 同色且相邻）→ 尝试绿色 (green) → 可行
4. d 尝试红色 → 不行（与 a/c 相邻）→ 尝试蓝色 → 不行（与 b 相邻）→ 尝试绿色 → 不行（与 c 相邻）→ 回溯到 c，改 c 为... 继续尝试
5. 最终：a=red, b=blue, c=green, d=red, e=green

Backtracking 决策树（搜索路径示意）:

```
a=red
  |
b=blue
```

```
        /      \
c=red        c=green  ← 红色与 a 冲突
               |
             d=red  ← 与 b=blue, c=green 均无冲突
               |
             e=green  ← 与 d=red 无冲突
                 找到解
```

决策树中每个分支代表一次尝试， 表示冲突（回溯）， 表示找到可行解。

Application 2: The n-Queens Problem (n 皇后问题)

Example 5: 在 4×4 棋盘上放置 4 个皇后，使它们互不攻击。在每一行逐列放置皇后，若当前位置与已放置的皇后冲突，则尝试下一列；若无列可用，则回溯到上一行，移动上一行的皇后。**Backtracking 过程（文字描述）**：

```
Row 1: 放置皇后在 col 1
Row 2: 尝试 col 1(冲突) -> col 2(冲突) -> col 3(可行) -> 放置
Row 3: 尝试 col 1(冲突) -> col 2(冲突) -> col 3(冲突) -> col 4(冲突) -> 回溯
Row 2: 移动到 col 4(可行)
Row 3: 尝试 col 1(冲突) -> col 2(可行) -> 放置
Row 4: 尝试 col 1(冲突) -> col 2(冲突) -> col 3(冲突) -> col 4(冲突) -> 回溯
Row 3: 移动到 col ... 无可用 -> 继续回溯
... 继续搜索直到找到解
```

Backtracking 过程中的部分棋盘状态 (Q= 皇后, ×= 冲突位置)：

```
第1行col1放皇后:        第2行col3可行:          第3行无位置 → 回溯:
col: 1 2 3 4            col: 1 2 3 4            col: 1 2 3 4
r1:  Q . . .            r1:  Q . . .            r1:  Q . . .
r2:  . . . .            r2:  . . Q .            r2:  . . Q .
r3:  . . . .            r3:  . . . .            r3:  x x x x  ← 全冲突
r4:  . . . .            r4:  . . . .            r4:  . . . .

回溯后改第2行列4:       第3行列2可行:          最终找到的解:
col: 1 2 3 4            col: 1 2 3 4            col: 1 2 3 4
r1:  Q . . .            r1:  Q . . .            r1:  . . Q .
r2:  . . . Q            r2:  . . . Q            r2:  Q . . .
r3:  . . . .            r3:  . Q . .            r3:  . . . Q
r4:  . . . .            r4:  . . . .            r4:  . Q . .
```

解的位置（一种可能的解）：

```
col: 1 2 3 4
row 1:  X . . .
row 2:  . . X .
row 3:  . . . X
row 4:  . X . .
```

10.4.8 8. DFS in Directed Graphs (有向图的深度优先搜索)

Example 6: 对有向图进行 DFS。

```
a ----> b ----> c
|       |       |
v       v       v
d ----> e ----> f
        |       |
        v       v
        g ----> h
```

DFS output (按照访问顺序):

```
a -> b -> c -> f -> h
                -> ...
```

DFS 在有向图中的输出顺序取决于具体的图结构和边的方向。

10.4.9 9. 小结与考点提示

考试重点：

1. **Spanning Tree 的定义和性质**

- 包含所有顶点的树（连通 + 无环）
- Theorem 1: 连通图 <=> 有生成树
- 一个连通图可以有多个不同的生成树

1. **DFS 构造生成树**

- 必须能手动执行 DFS 算法
- 能画出生成的树结构
- 理解 backtracking 的本质

1. **BFS 构造生成树**

- 必须能手动执行 BFS 算法
- 能画出生成的树结构（分层展示）

1. **Backtracking 应用**

- Graph Coloring（图着色）
- n-Queens Problem（n 皇后问题）
- 理解回溯的过程：尝试 → 失败 → 回溯 → 重试

1. **DFS vs BFS 的对比**

易错点：

- DFS 是" 一条路走到黑" 再回溯，BFS 是" 层层推进"
- 生成树中**不包含原图的所有边**，只包含 $n-1$ 条边
- 不同起始顶点会产生不同的生成树
- 回溯时要注意恢复状态（撤销之前的决策）

10.5 Chapter 11.5: Minimum Spanning Trees (最小生成树)

10.5.1 1. The Problem (问题描述)

考虑一个计算机网络，不同计算机之间的连接成本不同（月租费用不同）：

```
        1200
    A --------- B
    | \         |
    |  \        |
 2000|   \      |1300
    |    \      |
    |     \     |
    |      \    |
    |       \   |
    |        \  |
    |         \ |
    C --------- D --------- E
       1600    700    800
```

问题：哪些链路应该开通，才能保证任意两台计算机之间都有路径，同时使总成本最小？**答案**：找到一棵 **生成树 (spanning tree)**，使得树中边的权重之和最小。

10.5.2 2. Definition (定义)

Definition 1: A **minimum spanning tree (MST)** in a connected weighted graph is a spanning tree that has the **smallest possible sum of weights** of its edges. （在连通加权图中，**最小生成树**是所有生成树中边权重之和最小的一棵生成树。）

注意：最小生成树可能不唯一（当有相同权重的边时）。

10.5.3 3. Greedy Algorithms (贪心算法)

构造 MST 的两种经典算法都是 **greedy algorithms (贪心算法)**：

Greedy strategy: 每一步都选择当前最优（权重最小）的边，且不破坏约束条件。

两种算法：1. **Prim's Algorithm (普里姆算法)** 2. **Kruskal's Algorithm (克鲁斯卡尔算法)**

10.5.4 4. Prim's Algorithm (普里姆算法)

Algorithm Description

```
Procedure Prim (G: weighted connected undirected graph with n vertices)
T := a minimum-weight edge                        // 从权重最小的边开始
for i := 1 to n-2
begin
    e := an edge of minimum weight incident to a vertex in T
         and not forming a simple circuit in T if added to T
```

```
    T := T with e added
end
{T is a minimum spanning tree of G}
```

核心思路：

- 从一棵只有一条边的树开始（该边为权重最小的边）
- 每次添加一条**与当前树中顶点相邻**且**权重最小**且**不形成回路**的边
- 重复直到树中有 $n-1$ 条边

Prim's Algorithm 的关键特征：始终保持树是连通的（从一个顶点集合逐步扩展）。

Example 1: Prim's Algorithm

用 Prim's Algorithm 在以下图中求最小生成树：

```
          1200
     A --------- B
     |\          |
     | \         |
     |  \        |
     |   \       |
 2000|    \      |1300
     |     \     |
     |      \    |
     |       \   |
     |        \  |
     |         \ |
     C --------- D --------- E
        1600    700    800
```

Step-by-step process: Step 1: 权重最小的边是 **B-E (700)**。T = B-E **Step 2:** 与 T 中顶点 B, E 相邻的边：

- B-A: 1200
- B-C: 1300
- B-D: 1200
- E-C: 800
- E-D: 1400

最小的是 **E-C (800)**。T = B-E, E-C **Step 3:** 与 T 中顶点 B, E, C 相邻的边：

- B-A: 1200
- B-D: 1200
- C-A: 2000
- C-D: 1600
- E-D: 1400

最小的是 **B-A (1200)**（或 **B-D (1200)**，此处选 B-A）。T = B-E, E-C, B-A **Step 4:** 与 T 中顶点 B, E, C, A 相邻的边：

- A-D: 900
- B-D: 1200
- C-D: 1600
- E-D: 1400

最小的是 **A-D (900)**。T = B-E, E-C, B-A, A-D 现在树中有 5 个顶点和 4 条边，已达到 $n-1$ 条边，算法结束。**最终的 MST (Prim's Algorithm):** 边集：B-E(700), E-C(800), B-A(1200), A-D(900)

```
  A
 / \
B   D
|
E
|
C
```

总成本： $700 + 800 + 1200 + 900 = \mathbf{3600}$

Prim's Algorithm 执行过程图解

以下逐步展示 Prim 算法中 MST 的生长过程（加粗边 = 当前已选边）：

```
原始加权图：
          1200
     A --------- B
     |\          |
     | \         |
     |  \        |
     |   \       |
 2000|    \      |1300
     |     \     |
     |      \    |
     |       \   |
     |        \  |
     |         \ |
     C --------- D --------- E
        1600    700    800

Step 1: 选最小边 B-E (700)

     A          B
                          700
     C          D --- E
     T = {B-E}

Step 2: 加 E-C (800)

     A          B
     |
     C          D --- E
                          800
     T = {B-E, E-C}

Step 3: 加 B-A (1200)

     A --------- B
                 |
                 |
     C          D --- E
                 |
                 |
     T = {B-E, E-C, B-A}

Step 4: 加 A-D (900) → MST完成!

     A --------- B
     /           |
    /            |
     C          D---E
                  |
                  |
     T = {B-E, E-C, B-A, A-D}
      总权重 = 3600
```

注意：Prim 算法始终保持一棵连通的树，每一步从树中顶点出发向外延伸一条最小权边。

10.5.5 5. Kruskal's Algorithm (克鲁斯卡尔算法)

Algorithm Description

```
Procedure Kruskal (G: weighted connected undirected graph with n vertices)
T := empty graph
for i := 1 to n-1
begin
    e := any edge in G with smallest weight that does not
         form a simple circuit when added to T
    T := T with e added
end
{T is a minimum spanning tree of G}
```

核心思路：

- 从空图开始
- 每次选择**全局权重最小**且**不形成回路**的边加入
- 重复直到树中有 $n-1$ 条边

Kruskal's Algorithm 的关键特征：允许树在过程中是不连通的（形成森林），最后合并成一棵树。

Example 2: Kruskal's Algorithm

用 Kruskal's Algorithm 在相同的图中求 MST：**图的边按权重排序**：

边	权重
B-E	700
E-C	800
A-D	900
A-B	1200
B-D	1200
B-C	1300
D-E	1400
C-D	1600
A-C	2000

Step-by-step process:

Step	Edge	Weight	Action	Explanation
1	**B-E**	**700**	**Add**	权重最小，加入。森林：B,E
2	**E-C**	**800**	**Add**	不形成回路，加入。森林：B,E,C
3	**A-D**	**900**	**Add**	不形成回路，加入。森林：B,E,C, A,D
4	**A-B**	**1200**	**Add**	连接两个连通分量 B,E,C 和 A,D，不形成回路，加入
5	B-D	1200	**Skip**	B 和 D 已在同一棵树中，形成回路，跳过
⋯	其余边	⋯	**Skip**	所有顶点已连通，已有 n-1=4 条边，算法结束

现在树中有 5 个顶点和 4 条边，算法结束。**最终的 MST (Kruskal's Algorithm):**

```
T = {B-E(700), E-C(800), A-D(900), A-B(1200)}
```

总成本： $700 + 800 + 900 + 1200 = \mathbf{3600}$

Kruskal's Algorithm 执行过程图解

以下逐步展示 Kruskal 算法中森林的合并过程（加粗边 = 当前已选边）：

```
Step 1: 选全局最小边 B-E (700)     Step 2: 选 E-C (800)

    A          B                     A          B
                                     |
    C          D --- E               C          D --- E
                     700                                800     800

    森林: {B,E}                      森林: {B,E,C}

Step 3: 选 A-D (900)               Step 4: 选 A-B (1200) → 两森林合并!

    A          B                     A --------- B
    |                                |           |
    |                                |           |
    C          D --- E               C          D---E
               |                                 |
               |                                 |
    森林: {B,E,C}, {A,D}            森林合并为一棵树, MST 完成!
                                      总权重 = 3600
```

注意：Kruskal 算法允许中间状态有多个连通分量（森林），通过不断加入全局最小边逐步合并各分量，最终形成一棵树。

两种算法的结果对比

```
Prim's MST:                Kruskal's MST:
       A                          A
      / \                        / \
     B   D                      B   D
     |                          |
     E                          E
     |                          |
     C                          C
Cost: 3600                  Cost: 3600
边序: BE->EC->BA->AD      边序: BE->EC->AD->AB
```

两种算法得到了**相同的 MST 边集** B-E, E-C, A-B, A-D，但边的添加顺序不同。

10.5.6 6. Prim vs Kruskal 对比

特性	Prim's Algorithm	Kruskal's Algorithm
策略	从点出发，每次选到当前树最近的边	从边出发，每次选全局最小边
树的状态	始终保持连通	开始时是森林，结束时成树
选择依据	选与当前树中顶点相邻的最小边	选全局最小且不形成回路的边
判断回路	检查边的端点是否都在树中	使用 Union-Find 等数据结构
适用于	稠密图 (dense graph)	稀疏图 (sparse graph)
时间复杂度	$O(\|V\|^2)$ 或 $O(\|E\|\log\|V\|)$	$O(\|E\|\log\|E\|)$
数据结构	优先队列 (Priority Queue)	并查集 (Union-Find)

10.5.7 7. 小结与考点提示

考试重点：

1. **MST 的定义**
- 生成树中权重之和最小的一棵
- 可能不唯一

1. **Prim's Algorithm**
- 从权重最小的边开始（或从任意顶点开始，课件版本是从最小权重边开始）
- 每次选与当前树中顶点相邻的最小权重边
- **必须能手动模拟完整的算法过程**

1. **Kruskal's Algorithm**
- 从空图开始
- 每次选全局最小权重且不形成回路的边
- **必须能手动模拟完整的算法过程**

1. **两种算法的异同**
- 都是 greedy algorithm
- 都要求图是连通的
- 选择边的策略不同

易错点：

- Prim 的初始边选择：课件中是从 **a minimum-weight edge** 开始（与一些教材从任意顶点开始不同）
- Kruskal 中一定要先对边按权重排序
- 两种算法都要**检查是否形成回路**——这是最常见的失分点
- 当遇到权重相同的边时，任意选择一条即可，MST 可能不唯一
- 一定要确认最终树中的边数为 $n-1$

第十一章　Network Flow (网络流)

11.1　Network Flow 网络流

考试语言: 英文 | **概念解释**: 中文本笔记覆盖: Flow network definition, Max flow problem, Min cut, Ford-Fulkerson algorithm, Residual graph, Augmenting path

11.1.1　1. Network Flow Definitions (网络流定义)

1.1 Flow Network (流网络)

A **flow network** (流网络) is a directed graph $G = (V, E)$ with:

- A **source** (源点) s —where flow originates
- A **sink** (收点/汇点) t —where flow terminates
- Each edge $e \in E$ has a **capacity** (容量) $c(e) \geq 0$, representing the maximum amount of flow that can pass through it

图示描述: 在一个流网络中，每条边上标注的是容量值 $c(e)$。箭头方向表示允许流动的方向。源点 s 只有出边，收点 t 只有入边。

1.2 Flow (可行流)

Definition. An **s-t flow** (可行流) is a function $f : E \to \mathbb{R}$ that satisfies:

1. **Capacity constraint** (容量约束): For each $e \in E$:

$$0 \leq f(e) \leq c(e)$$

2. **Flow conservation** (守恒约束): For each vertex $v \in V - \{s, t\}$:

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$

流入量 = 流出量（除了源点和收点）

Definition. The **value** of a flow f is:

$$v(f) = \sum_{e \text{ out of } s} f(e)$$

即从源点 s 流出的总流量。

特例: **零流** (zero flow) —所有 $f(e) = 0$，显然是一个可行流。

1.3 Multiple Sources or Sinks (多源点多收点)

当问题中有多个源点或多个收点时，可以通过添加一个 **supersource** (超级源点) 和一个 **supersink** (超级收点) 来转化：

- 从 supersource s 到每个原始源点连一条容量为 ∞ 的边
- 从每个原始收点到 supersink t 连一条容量为 ∞ 的边

图示描述: 多个源点 a, b, c 通过容量 ∞ 的边连接到新的超级源点 s；多个收点 k, l 通过容量 ∞ 的边连接到新的超级收点 t。这样就把多源多收问题转化为标准单源单收问题。

11.1.2　2. Maximum Flow Problem (最大流问题)

Max flow problem: Find an s-t flow of maximum value. 在所有可行流中，找出流量值 $v(f)$ 最大的流。

图示描述: - 一个可行流示例中，每条边上有两个数字：灰色标注的是**流量** $f(e)$，黑色标注的是**容量** $c(e)$。例如边 $s \to a$ 上写有 $\frac{10}{10}$ 表示 $f = 10, c = 10$。- 随着不断调整流量分配，可以从初始流值 $v(f) = 4$ 逐步增大到 $v(f) = 24$，最终达到最大流 $v(f) = 28$。

11.1.3　3. Cuts in a Graph (图中的割)

3.1 Definition of Cut (割的定义)

Definition. An **s-t cut** is a partition (A, B) of V with $s \in A$ and $t \in B$. 即将顶点集 V 划分为两部分 A 和 B，其中源点 s 必须在 A 中，收点 t 必须在 B 中。**Definition**. The **capacity of a cut** (割的容量) (A, B) is:

$$\text{cap}(A, B) = \sum_{e \text{ out of } A} c(e)$$

即从 A 指向 B 的所有边的容量之和。

图示描述: - 第一个割例: A 包含 s 及其相邻的一些顶点，从 A 到 B 的边容量为 $10 + 5 + 15 = 30$ - 第二个割例: 另一个不同的划分，容量为 $9 + 15 + 8 + 30 = 62$ - 第三个割例: 更优化的划分，容量为 $10 + 8 + 10 = 28$

3.2 Cap(S,T) and Flow(S,T)

对于一个割 (A, B):

- **Cap(S,T)**: 从 A 到 B 的所有边的容量之和
- **Flow(S,T)**: 从 A 到 B 的净流量（流出 A 的流量减去流入 A 的流量）

关系: $\text{Flow}(S, T) \leq \text{Cap}(S, T)$ —净流量不可能超过割容量。

3.3 Minimum Cut Problem (最小割问题)

Min s-t cut problem: Find an s-t cut of minimum capacity. 在所有可能的 s-t 割中，找出容量最小的割。

11.1.4　4. Flows and Cuts —Key Lemmas (流与割的关键引理)

4.1 Flow Value Lemma (流值引理)

Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then the net flow sent across the cut is equal to the value of the flow:

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$

解释: 对于任意一个割，从 A 侧净流出的流量（流出 A 减去流入 A）总是等于整个流的流量 $v(f)$。

Proof:

$$\begin{aligned} v(f) &= \sum_{e \text{ out of } s} f(e) \\ &= \sum_{v \in A} \left(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to } v} f(e) \right) \\ &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \end{aligned}$$

- **从顶点角度**: 除了 s 外，A 中所有其他顶点的出入流量在求和中相互抵消 - **从边角度**: 位于 A 内部的边，其出入流量在求和中相互抵消，只有 A 和 B 之间的边才起作用

图示验证: - 一个 $v(f) = 24$ 的流，取一个割 (A, B)，- 计算: $10 + 3 + 11 = 24$（直接从 A 流出的边）- 或计算: $6 + 0 + 8 - 1 + 11 = 24$（考虑反向流量后净流出）

4.2 Weak Duality (弱对偶性)

Weak duality. Let f be any flow, and let (A, B) be any s-t cut. Then the value of the flow is at most the capacity of the cut:

$$v(f) \leq \text{cap}(A, B)$$

Proof:

$$v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \leq \sum_{e \text{ out of } A} f(e) \leq \sum_{e \text{ out of } A} c(e) = \text{cap}(A, B)$$

直觉: 割容量是流必须经过的”瓶颈”，流的流量不可能超过任何割的容量。

4.3 Certificate of Optimality (最优性凭证)

Corollary. Let f be any flow, and let (A, B) be any cut. If $v(f) = \text{cap}(A, B)$, then f is a **max flow** and (A, B) is a **min cut**.

意义: 当我们找到一个流的流量等于某个割的容量时，我们就同时证明了该流是最大流、该割是最小割。这是最优性的充要条件。

11.1.5　5. Residual Graph (残差图)

5.1 Definition (定义)

Given a flow network G with flow f, the **residual graph** (残差图) $G_f = (V, E_f)$ is constructed as follows: For each original edge $e = (u, v) \in E$:

- If $f(e) < c(e)$: add a **forward edge** $e = (u, v)$ with residual capacity $c_f(e) = c(e) - f(e)$

  前向边：剩余容量 = 容量 - 已用流量 - If $f(e) > 0$: add a **reverse edge** $e_R = (v, u)$ with residual capacity $c_f(e_R) = f(e)$ 反向边：容量 = 已用流量（允许”撤销”已发送的流量）

  Formally:

$$c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e_R \in E \end{cases}$$

$$E_f = \{e : f(e) < c(e)\} \cup \{e_R : f(e) > 0\}$$

5.2 Intuition (直观理解)

残差图告诉我们**还能发送多少额外流量**以及**可以撤销多少已有流量**。

- **前向边** (forward edge): 表示还可以在原方向上发送 $c(e) - f(e)$ 单位流量
- **反向边** (reverse edge): 表示可以撤销 $f(e)$ 单位已发送的流量（即把流量”退回去”）

图示描述: - 原图: 从 u 到 v 的一条边，容量 c，流量 f，记为 $\frac{f}{c}$ - 残差图: 拆成两条边——- 前向边 $u \to v$，残差容量 $c - f$ - 反向边 $v \to u$，残差容量 f

11.1.6 6. Augmenting Path (增广路径)

6.1 Definition (定义)

An **augmenting path** (增广路径) is a path $P = v_1, v_2, \ldots, v_k$ in the residual graph G_f where:

- $v_1 = s$ (起点是源点)
- $v_k = t$ (终点是收点)
- For each consecutive pair (v_j, v_{j+1}), the edge has **positive** residual capacity $c_f(v_j, v_{j+1}) > 0$

找到一条从 s 到 t 的路径，路径上的每条边在残差图中都有正的剩余容量。

6.2 Bottleneck Capacity (瓶颈容量)

The **bottleneck** of an augmenting path P is:

$$b = \min_{(u,v)\in P} c_f(u, v)$$

即路径上所有边的最小残差容量，也就是通过这条路径最多能增加的流量。

11.1.7 7. Ford-Fulkerson Algorithm (福特-福克森算法)

7.1 Algorithm Description (算法描述)

Ford-Fulkerson (1956) is a greedy-based algorithm for finding the maximum flow.

```
Ford-Fulkerson(G, s, t, c) {
    foreach e in E
        f(e) <- 0
    Gf <- residual graph

    while (there exists augmenting path P in Gf) {
        f <- Augment(f, c, P)
        update Gf
    }
    return f
}

Augment(f, c, P) {
    b <- bottleneck(P)
    foreach e in P {
        if (e in E)             // forward edge: 前向边
            f(e) <- f(e) + b
        else                    // reverse edge: 反向边
            f(e_R) <- f(e_R) - b
    }
    return f
}
```

7.2 Manual Steps (手动步骤)

1. **初始化**: 所有边流量 $f(e) = 0$
2. **构建残差图**: 根据当前流量构造 G_f
3. **找增广路径**: 在 G_f 中找一条从 s 到 t 的路径 P（每条边残差容量 > 0）
4. **增广**: 计算瓶颈 b，沿 P 增加 b 单位流量（前向边加，反向边减）
5. **更新残差图**: 重新计算残差容量
6. **重复**: 直到残差图中不存在 s 到 t 的路径

图示描述: 算法执行过程示例——- 初始所有边流量为 0 - 第一次迭代选择某条路径，增广后流值变为 20 - 由于贪心选择的局部最优性，可能无法直接达到全局最优 - 但在残差图中利用反向边可以”纠正”之前的决策，最终达到最大流 30

7.3 Greedy vs. Ford-Fulkerson (贪心 vs. FF 算法)

- **贪心算法**: 只使用前向边，一旦选择路径就不会改变 → 可能陷入局部最优
- **Ford-Fulkerson**: 使用残差图，通过反向边允许”撤销”之前的选择 → 可以达到全局最优

核心洞见: 反向边是 FF 算法的关键创新。它允许算法”反悔”之前做出的次优决策，从而保证能找到全局最大流。

11.1.8 8. Max-Flow Min-Cut Theorem (最大流最小割定理)

8.1 Statement (陈述)

Augmenting path theorem. Flow f is a max flow iff there are **no augmenting paths** in the residual graph G_f. **Max-flow min-cut theorem** (Elias-Feinstein-Shannon 1956, Ford-Fulkerson 1956). The value of the **maximum flow** equals the capacity of the **minimum cut**:

$$\max_f v(f) = \min_{(A,B)} \mathrm{cap}(A, B)$$

即最大流的值等于最小割的容量。

8.2 Proof (证明)

We prove the following three statements are equivalent (TFAE): **(i) There exists a cut (A, B) such that $v(f) = \mathrm{cap}(A, B)$.** 存在一个割的容量等于流 f 的值。**(ii) Flow f is a max flow.** f 是最大流。**(iii) There is no augmenting path relative to f.** 对于 f 没有增广路径。

(i) ⇒ (ii): 由弱对偶性 (Weak Duality) 的直接推论。若存在割 (A, B) 使得 $v(f) = \mathrm{cap}(A, B)$，则对于任意其他流 f' 有 $v(f') \le \mathrm{cap}(A, B) = v(f)$，所以 f 是最大流。

(ii) ⇒ (iii): 采用逆否命题证明 (contrapositive)。若存在增广路径，则我们可以沿该路径增广，得到更大的流值，因此 f 不是最大流。逆否命题成立。

(iii) ⇒ (i): 设 f 是一个没有增广路径的流。定义集合 A 为在残差图 G_f 中从 s 出发可达的所有顶点。

- $s \in A$ (显然)
- $t \notin A$ (因为没有增广路径)

令 $B = V - A$，则 (A, B) 是一个 s-t 割。对于从 A 到 B 的边: 在残差图中不可达，所以前向边必须是满的 $(f(e) = c(e))$，反向边流量必须为 0 $(f(e) = 0)$。对于从 B 到 A 的边: 在残差图中反向边不可达，所以 $f(e) = 0$。因此:

$$\begin{aligned} v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\ &= \sum_{e \text{ out of } A} c(e) - 0 \\ &= \mathrm{cap}(A, B) \end{aligned}$$

图示描述: 在证明中，A 集合由残差图中从 s 可达的所有顶点组成。由于没有增广路径，t 不在 A 中。从 A 到 B 的所有前向边必须已满容量（否则残差图上仍有边，B 中的顶点就可达了），从 B 到 A 的所有边流量必须为 0（否则反向边会让 B 中顶点可达）。由此可推导出 $v(f) = \mathrm{cap}(A, B)$。

8.3 Key Consequences (关键推论)

- **Ford-Fulkerson 算法正确性**: FF 算法找到的流使得残差图不再连通（s 到 t 不可达），因此由定理，该流是最大流
- **找最小割**: 先运行 FF 算法找到最大流，然后在最终残差图中，从 s 出发可达的顶点集合就是最小割的 A 侧

  找最小割的方法: 先求最大流，再在残差图中做 BFS/DFS 从 s 出发找所有可达顶点，即为最小割的 A 侧。

11.1.9 9. Performance and Improvements (性能与改进)

9.1 Integer Capacity Assumption (整数容量假设)

Assumption: All capacities are integers between 1 and C. **Invariant**: Every flow value $f(e)$ and every residual capacity $c_f(e)$ remains an **integer** throughout the algorithm. **Theorem**: The algorithm terminates in at most $v(f^*) \le nC$ iterations.

证明: 每次增广至少增加 1 单位流量，而最大流值不超过 nC。

9.2 Worst-Case Performance (最坏情况性能)

Ford-Fulkerson 算法的最坏情况非常糟糕，可能需要进行 C 次迭代（指数级）。

图示描述: 经典反例——- 一个简单的三角网络，$s \to u$ 容量 C，$s \to v$ 容量 C，$u \to v$ 容量 1，$u \to t$ 容量 C，$v \to t$ 容量 C - 如果每次选择的增广路径都经过中间容量为 1 的边，则每次只能增加 1 单位流量 - 总共需要 C 次增广，当 C 很大时效率极低

Q: Is generic Ford-Fulkerson algorithm polynomial in input size $(m, n, \log C)$? **A**: No. If max capacity is C, the algorithm can take C iterations.

9.3 Integrality Theorem (整数性定理)

Integrality theorem. If all capacities are integers, then there exists a max flow f for which every flow value $f(e)$ is an integer.

所有容量为整数时，存在一个最大流使得每条边上的流量也是整数。FF 算法保证输出整数流。

Corollary: If $C = 1$, Ford-Fulkerson runs in $O(mn)$ time. 当所有容量为 1 时，每次增广至少增加 1，最大流值 $\le n$，故最多 n 次增广。

9.4 Choosing Good Augmenting Paths (选择好的增广路径)

重要注意: 如果容量是**无理数** (irrational capacities)，Ford-Fulkerson 算法不保证终止！因此在选择增广路径时需要谨慎。

使用不同的策略选择增广路径可以显著改善性能:

Strategy	Description	Time Complexity
Max bottleneck capacity	选瓶颈容量最大的路径	$O(m^2 \log C)$
Shortest path (Edmonds-Karp/Dinitz)	选边数最少的路径 (BFS)	$O(m^2 n)$
Blocking flow	在残差图中找阻塞流	$O(mn \log n)$

9.5 Capacity Scaling (容量缩放算法)

Intuition: 选择瓶颈容量大的路径，每次尽可能多地增加流量。**Algorithm** (Scaling Max-Flow):

```
Scaling-Max-Flow(G, s, t, c) {
    foreach e in E
```

```
        f(e) <- 0
    Δ <- smallest power of 2 >= C
    Gf <- residual graph

    while (Δ >= 1) {
        Gf(Δ) <- Δ-residual graph (only edges with capacity >= Δ)
        while (there exists augmenting path P in Gf(Δ)) {
            f <- Augment(f, c, P)
            update Gf(Δ)
        }
        Δ <- Δ / 2
    }
    return f
}
```

Theorem: The scaling max-flow algorithm finds a max flow in $O(m \log C)$ augmentations, and can be implemented to run in $O(m^2 \log C)$ time.

核心思想: 先处理大容量边（Δ 较大时），确保快速增加流量；然后逐渐减小 Δ，精细化调整。整体迭代次数从 C 减少到 $O(\log C)$ 轮。

11.1.10 10. Application –Bipartite Matching (应用——二分图匹配)

10.1 Bipartite Graph (二分图)

A **bipartite graph** (二分图) is an undirected graph $G = (V, E)$ in which V can be partitioned into two sets V_1 and V_2 such that every edge $(u, v) \in E$ implies $u \in V_1$ and $v \in V_2$ (or vice versa). 即所有边都连接两个集合之间的顶点，集合内部没有边。

10.2 Matching Problem (匹配问题)

- **问题**: 有 n 个男人和 m 个女人，已知某些配对是兼容的。最大化成功配对的数量（不允许一夫多妻或一妻多夫）
- **数学模型**: 在二分图中找到最大匹配 (maximum matching)，即最多的边数，使得没有两条边共享顶点

10.3 Solution Using Max Flow (用最大流求解)

将二分图匹配问题转化为最大流问题：**Construction**:

1. 添加一个 **supersource** s 连接到所有左侧顶点（男人），每条边容量为 1
2. 添加一个 **supersink** t，所有右侧顶点（女人）连接到 t，每条边容量为 1
3. 将原二分图中的每条无向边替换为容量为 1 的有向边（从男人指向女人）
4. 所有边容量均为 1

Why it works:

- 容量为 1 的边确保每个顶点最多匹配一次（流量守恒约束）
- 从 s 到 t 的最大流值即为最大匹配数

图示描述: - 左侧: 男人集合 $\{A, B, C, D\}$，右侧: 女人集合 $\{X, Y, Z\}$，兼容关系用边连接 - 加入 s 和 t 后，所有从 s 出发和进入 t 的边容量为 1 - 最大流的整数性保证每个单位的流量对应一对匹配 - 例子中最大匹配为 3（$A-X, B-Y, C-Z$，或 $A-Y, B-X, C-Z$ 等）

11.1.11 11. Summary and Exam Tips (小结与考点提示)

11.1 Key Concepts Checklist (关键概念检查表)

中文	English	Importance
流网络	Flow Network	基础定义
容量	Capacity $c(e)$	基础定义
可行流	Flow $f(e)$	基础定义
容量约束	Capacity constraint $0 \le f(e) \le c(e)$	基础定义
守恒约束	Flow conservation	基础定义
流值	Flow value $v(f)$	基础定义
源点/收点	Source s / Sink t	基础定义
割	s-t Cut (A, B)	**核心概念**
割容量	Cut capacity $\text{cap}(A, B)$	**核心概念**
残差图	Residual Graph G_f	**核心概念**
增广路径	Augmenting Path	**核心概念**
瓶颈	Bottleneck	**核心概念**
Ford-Fulkerson	Ford-Fulkerson Algorithm	**核心算法**
最大流最小割定理	Max-Flow Min-Cut Theorem	**核心定理**
弱对偶性	Weak Duality	重要引理
流值引理	Flow Value Lemma	重要引理
整数性定理	Integrality Theorem	重要定理
容量缩放	Capacity Scaling	算法改进
二分图匹配	Bipartite Matching	重要应用

11.2 Typical Exam Questions (典型考题)

1. **Find max flow**: 给定一个流网络，用 Ford-Fulkerson 算法计算最大流
- 步骤: 初始化 → 画残差图 → 找增广路径 → 增广 → 重复 → 直到无增广路径
- 每次迭代都要记录流量和残差图的变化

1. **Find min cut**: 在最大流的基础上找出最小割
- 方法: 在最终残差图中从 s 出发做 BFS/DFS，所有可达顶点构成 A，其余为 B

1. **Prove optimality**: 证明找到的流是最大流
- 方法: 找到一个割，使得 $v(f) = \text{cap}(A, B)$

1. **Apply to bipartite matching**: 将实际问题建模为二分图匹配，再转化为最大流问题

11.3 Common Pitfalls (常见错误)

- **忘记反向边**: 在构建残差图时必须同时考虑前向边和反向边
- **割容量计算错误**: 只计算从 A 到 B 的边，不计算从 B 到 A 的边
- **流值 vs 割容量的混淆**: 流值是净流量（考虑双向），割容量只考虑单向
- **增广路径必须用残差图**: 在原图中可能找不到路径，但在残差图中能找到

11.4 Summary of Key Relationships (关键关系总结)

```
弱对偶性: v(f)  cap(A, B)  对于任意流 f 和任意割 (A,B)

推理: 若 v(f) = cap(A,B) → f 是最大流, (A,B) 是最小割 (Certificate of Optimality)

最大流最小割定理: max v(f) = min cap(A,B)
                  (最大流值)  (最小割容量)

算法: Ford-Fulkerson 通过不断在残差图中找增广路径逼近这个值
```

11.5 Formula Reference Card

Formula	Description
$0 \le f(e) \le c(e)$	Capacity constraint
$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$	Flow conservation
$v(f) = \sum_{e \text{ out of } s} f(e)$	Flow value
$\text{cap}(A, B) = \sum_{e \text{ out of } A} c(e)$	Cut capacity
$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$	Flow value lemma
$v(f) \le \text{cap}(A, B)$	Weak duality
$c_f(e) = c(e) - f(e)$ (forward), $c_f(e_R) = f(e)$ (reverse)	Residual capacity
$b = \min_{(u,v) \in P} c_f(u, v)$	Bottleneck capacity
$\max_f v(f) = \min_{(A,B)} \text{cap}(A, B)$	Max-flow min-cut theorem

Final tip: 考试中如果要求"Prove that the flow is maximum"，最直接的方法就是构造一个割使得流值等于割容量。找最小割的方法是：在最终残差图中从 s 做 BFS。

第一部分

历年真题考点分析

11.2 DMA 历年真题考点分析—Ch1-3 (郑文庭班)

分析范围：Quiz 1 (2022-2025)、Quiz 4 中涉及 Ch1-3 的题目、期末回忆卷排除：Quiz 2 / Quiz 3（大部分为 Ch4-8 内容）、纯数论/同余/中国剩余定理题

11.2.1 Ch1 Logic & Proofs

题型 1：命题逻辑等价性与永真式判断（True/False）

出现年份：2022, 2023, 2024, 2025 **题目示例**：

- 2022 Q1a：判断 `p → (¬q  r)` 与 `¬p  ¬(r → q)` 是否逻辑等价
- 2022 Q1c：判断 `((p → q)  ¬p) → ¬q` 是否为永真式
- 2023 Q1e：判断 `¬(p → q)  q` 是否为永真式
- 2024 Q1a：判断 `p → (q → r)` 与 `(p → q) → r` 是否逻辑等价
- 2024 Q1c：判断 `(8+3=9) iff (8-3=7)` 的真值

考点：

- 命题逻辑的基本等价律（蕴含式改写、德摩根律、结合律）
- 永真式（tautology）的定义与判定（真值表或等价变形）
- 条件语句的蕴涵关系与逻辑等价

解题思路：

- 对等价性判断：分别写出两个命题的真值表，或使用已知等价律逐步转换。关键在于 `p→q  ¬p q` 和德摩根律的熟练运用。
- 对永真式判断：检查是否所有真值指派下都为真。常用方法：假设前提为真，看结论是否必然为真；或列真值表。
- 注意 `p→(q→r)` 与 `(p→q)→r` 不等价（前者等价于 `(p q)→r`）。

题型 2：谓词逻辑——自然语言翻译为符号

出现年份：2022, 2024 **题目示例**：

- 2022 Q2：变量 `x` 表示学生，`y` 表示课程，`T(x,y)` 表示"x 在修 y"。翻译：
- (a) 没有课程被所有学生修
- (b) 每个学生至少修一门课
- (c) 有些课程没有学生修
- (d) 没有学生修所有课程
- 2024 Q2 类似的题目（" 有一门课被所有学生修"、" 没有学生修所有课程"）

考点：

- 全称量词 与存在量词 的混合使用
- " 没有……" 的翻译方式：`¬ x` 或 `x¬`
- 量词顺序对语义的影响

解题思路：

- " 所有 A 都是 B"：`x(A(x) → B(x))`
- " 存在 A 是 B"：`x(A(x)  B(x))`
- " 没有 A 是 B"：`¬ x(A(x)  B(x))` `x(A(x) → ¬B(x))`
- 示例：(a) No course is being taken by all students → `¬ y x T(x,y)` 或 `y x ¬T(x,y)`
- (d) No student is taking all courses → `¬ x y T(x,y)` 或 `x y ¬T(x,y)`

题型 3：量词等价与谓词逻辑真值判断（True/False）

出现年份：2023, 2025 **题目示例**：

- 2023 Q1a：判断 `xP(x)  xQ(x)  x y(P(x)  Q(y))`（前提：量词域相同且非空）
- 2023 Q2：单选题——下列哪个等价式正确？(考察 / 对 / /→ 的分配律)
- 2025 Q1a：若 x 不在 A 中出现，则 `x(P(x)→A)  xP(x)→A`
- 2025 Q1d：给定 P(x,y) 在 1,2,3,4 上的真值表，判断 `x y((x y) P(x,y))`

考点：

- 量词的分配律：`x(P(x) Q(x))  xP(x) xQ(x)` 有效,但 `x(P(x) Q(x))` 不能分配；`x(P(x) Q(x))  xP(x) xQ(x)` 有效,但 `x(P(x) Q(x))` 不能分配
- 量词与 → 的关系：`x(P(x)→Q(x))` 不等价于 `xP(x)→ xQ(x)`
- 换名规则与辖域扩张：当被量化的变量不在另一部分中出现时，可进行辖域扩张
- 给定真值表判断量化语句：对每个 x 逐一检查是否存在 y

解题思路：

- 对于 `x(P(x)→A)  xP(x)→A`（x 不在 A 中）：将 `P(x)→A` 改写为 `¬P(x) A`，则 `x(¬P(x) A)  ( x¬P(x)) A  ¬ xP(x) A  xP(x)→A`
- 2023 Q2：A 选项 `x(P(x) Q(x))  xP(x) xQ(x)` 正确；其他选项的 / /→ 分配均可能失效
- 2025 Q1d：对每个 x=1,2,3,4，找到对应的 y 使 (x y) P(x,y) 为真即可

题型 4：用受限联结词表达命题

出现年份：2022, 2023, 2024, 2025（每年必考！）**题目示例**：

- 2022 Q3a：用 ¬、 表达 p ¬q
- 2022 Q3b：仅用 NAND(|) 表达 p q
- 2023 Q3a：用 ¬、 表达 p q
- 2023 Q3b：仅用 NOR(↓) 表达 p q
- 2024 Q4：仅用 NAND(|) 表达 p ¬q
- 2025 Q2a：用 ¬、 表达 p q
- 2025 Q2b：仅用 NAND(|) 表达 p q

考点：

- 联结词的完备集：¬, 、¬, 、NAND、NOR 都是功能完备的
- NAND 和 NOR 的定义与基本性质
- 命题等价变换技巧

解题思路：

- 先用德摩根律消去不需要的联结词，逐步转化为目标联结词
- NAND：p|q ¬(p q)，¬p p|p，p q ¬(p|q) (p|q)|(p|q)
- NOR：p↓q ¬(p q)，¬p p↓p，p q ¬(p↓q) (p↓q)↓(p↓q)
- p q (p ¬q) (¬p q)，再用德摩根律转化为 ¬(¬(p ¬q) ¬(¬p q))，最后用 ¬ 和 表达

题型 5：主析取范式 (Full DNF) 与主合取范式 (Full CNF)

出现年份：2022, 2023, 2024, 2025（每年必考！）**题目示例**：

- 2022 Q4：三变量 p,q,r，命题在至多一个变量为真时取真，求 Full DNF 和 Full CNF
- 2023 Q5：求 (p ¬r)→(q r) 的 Full CNF 和 Full DNF
- 2024 Q5：求 p (q r) 的 Full DNF 和 Full CNF
- 2025 Q3：求 (p q) r 的 Full CNF

考点：

- DNF（析取范式）：各子句是合取式（极小项），子句之间用 连接
- CNF（合取范式）：各子句是析取式（极大项），子句之间用 连接
- Full DNF：每个合取子句包含所有变量（或其否定）恰好一次
- Full CNF：每个析取子句包含所有变量（或其否定）恰好一次
- 极小项 (minterm) 与极大项 (maxterm) 的对应关系

解题思路：

- 列真值表，找出所有使公式为真的行 → 每行对应一个极小项 → 析取得 DNF
- 找出所有使公式为假的行 → 每行取反得一个极大项 → 合取得 CNF
- 或利用等价变换：先写出一般形式，再补充缺失的变量（如 p p (q ¬q)）
- DNF 与 CNF 互为对偶，可通过德摩根律互换
- 注意：Full DNF 和 Full CNF 唯一的充要条件是公式不是永真/永假

题型 6：谓词逻辑的否定

出现年份：2024 (Quiz 4 Q1b) **题目示例**：

- 2024 Quiz4 Q1b：写出 x yP(x,y) x yQ(x,y) 的否定形式，且否定符号仅直接出现在谓词之前

考点：

- 量词否定的对偶律：¬ xP(x) x¬P(x)，¬ xP(x) x¬P(x)
- 德摩根律在谓词逻辑中的应用

解题思路：

- ¬(x yP(x,y) x yQ(x,y))
- ¬ x yP(x,y) ¬ x yQ(x,y)（德摩根律）
- x y¬P(x,y) x y¬Q(x,y)（量词对偶律）

题型 7：数学归纳法证明

出现年份：2022, 2023, 2024, 2025（每年必考！）**题目示例**：

- 2022 Q8：用数学归纳法证明 n ¹ > (n+1) 对所有正整数 n 3 成立
- 2023 Q7a：证明 f 0 (mod 5)（Fibonacci 数）
- 2023 Q7b：证明 f² + f² = f
- 2024 Q9：证明若 x>0, y>0，则 (x +y)/2 ((x+y)/2)
- 2025 Q8：证明每个大于 2 的正整数都可表示为不同 Fibonacci 数之和

考点：

- 标准数学归纳法（奠基 + 归纳假设 + 归纳步骤）

- 强归纳法（Strong Induction）
- 归纳法中代数不等式放缩技巧
- Fibonacci 数的性质

解题思路：

- **标准归纳法**：证明 P(1) 成立；假设 P(k) 成立，证明 P(k+1) 成立
- **2022 Q8**：n 3 时，用二项式展开和放缩。注意 n ¹ > (n+1) 等价于 n > (1+1/n)，利用 (1+1/n) < 3 及 n 3 得证
- **2023 Q7a**：利用 Fibonacci 数模 5 的周期性，或直接归纳 f = 5 · f · g 形式
- **2023 Q7b**：从 f² +f² = f 出发，利用递推关系 f = f + f 进行归纳
- **2024 Q9**：用均值的归纳形式，(x ¹+y ¹)/2 = (x +y)(x+y)/4 - (x-y)²(x ¹-y ¹)/4，进行放缩
- **2025 Q8**：使用强归纳法，找到小于 n 的最大 Fibonacci 数 f，考虑 n-f 的情况

题型 8：其他证明方法（反证法/逆否命题/构造）

出现年份：2023, 2024 **题目示例**：

- 2024 Q8：用逆否命题证明——若 x^3 是无理数，则 x 是无理数
- 2023 Q6：给定有理数子集 S 满足封闭性和三分性，证明 S = Q
- 2023 Q8：两堆石子取石子游戏，先手必胜策略（不变量的构造）

考点：

- 逆否命题证明法（Prove by contrapositive）：p→q ¬q→¬p
- 反证法（Proof by contradiction）
- 不变量（invariant）与策略偷换（strategy stealing）
- 集合性质证明

解题思路：

- **2024 Q8**（逆否）：假设 x 是有理数，则 x = a/b（a,b Z,b 0），则 $x^3 = a^3/b^3$ 也是有理数，证毕。逆否命题等价于原命题。
- **2023 Q6**（构造性证明）：利用条件证明对所有正有理数 r，r S；同时对负有理数 r，¬(r S)。由三分性，对任意有理数 r，要么 r S, 要么 -r S, 要么 r=0。先证所有正整数都在 S 中（用 1 封闭性和归纳），再证所有正有理数在 S 中。
- **2023 Q8**（取石子策略）：用 Nim 和的思路，两堆石子数 (115,125)，计算二进制异或和。先手可通过一步操作使异或和为 0，然后模仿对手操作。

11.2.2 Ch2 Sets, Functions, Sequences

题型 1：集合恒等式判断（True/False）

出现年份：2022, 2023, 2025 **题目示例**：

- 2022 Q1b：A - (B C) = (A - B) (A - C) 是否成立
- 2022 Q1d：P(A) = P(B) 当且仅当 A = B（幂集性质）
- 2023 Q1b：(A - C) - (B - C) = A - B 是否成立
- 2023 Q1d：若 P(A) P(B)，则 A B 是否成立
- 2025 Q1b：同 2022 Q1b

考点：

- 集合的差、交、并运算律（特别是德摩根律在集合上的形式）
- 幂集（power set）的性质
- 集合相等的证明方式

解题思路：

- 用文氏图快速验证，或用元素分析法：x 左式 ... x 右式
- 2022 Q1b 正确：A - (B C) = A (B C)^c = A (B^c C^c) = (A B^c) (A C^c) = (A-B) (A-C)
- 2023 Q1d 错误：反例 A=1, B=1，则 P(A)= ,1 ,1 = P(B)，但 A B

题型 2：对称差运算

出现年份：2024 **题目示例**：

- 2024 Q1b：判断 A (B C) = (A B) C 是否成立

考点：

- 对称差（symmetric difference）的定义：A B = (A-B) (B-A)
- 对称差满足结合律

解题思路：

- 对称差实际上就是集合的 XOR 运算，满足结合律和交换律。因此 A (B C) = (A B) C 成立。
- 可以用元素分析法或真值表法验证：对任意元素 x，它在左式中的隶属情况与在右式中一致。

题型 3：集合恒等式的归纳法证明

出现年份：2022, 2025 **题目示例**：

- 2022 Q5：用数学归纳法证明分配律 A (A ... A) = (A A) ... (A A) 对任意 n>2 成立

- 2025 Q7：证明 A (A ... A) = (A A) ... (A A)

考点：

- 集合运算的归纳证明
- 分配律的推广形式
- 用元素分析法证明集合相等

解题思路：

- 两条分配律互为对偶。证明框架：
- 奠基 n=2：就是基本的 对 （或 对 ）的分配律
- 归纳步骤：假设 n=k 成立，证明 n=k+1
- A (A ... A A) = A ((A ... A) A)
- 对括号内部先使用 n=2 分配律，再使用归纳假设
- 也可用元素分析法直接展开证明

题型 4：康托尔对角化——不可数性证明

出现年份：2022, 2024, 2025 **题目示例**：

- 2022 Q6：证明 (0,1) 中十进制表示只含 0 和 1 的实数集合不可数
- 2024 Q1d：判断——(0,1) 中十进制表示只含 6 和 8 的实数集合不可数 [TRUE]
- 2025 Q1f：判断——(0,1) 中十进制表示只含 0 和 1 的实数集合可数 [FALSE]

考点：

- 康托尔对角线论证法（Cantor diagonalization argument）
- 可数集与不可数集的区别
- 对角化构造：对任意可数列表构造一个不在列表中的新数

解题思路：

- 假设集合 S 可数，列出 S 中所有元素 r , r , r , ...
- 每个 r 写成十进制小数 0.a a a ...
- 构造新数 r = 0.b b b ...，其中 b a （且 b 在允许的数码集合中）
- 则 r 不在原列表中（因为与每个 r 至少在第 i 位不同），但 r S，矛盾
- 2024 Q1d 的数码是 6 和 8，依然可以用对角化证明不可数

题型 5：可数性判断（True/False）

出现年份：2023, 2024 **题目示例**：

- 2023 Q1c：整系数三次方程 $ax^3+bx^2+cx+d=0$ 的实数解集合不可数 [FALSE——可数]
- 2023 Q1f：存在从 R 到 Z×Z 的单射 [FALSE]
- 2024 Q1e：整系数二次方程 $ax^2+bx+c=0$ 的实数解集合可数 [TRUE]
- 2024 Quiz4 Q1d：三维空间中三个坐标都是有理数的点的个数 [可数无穷多]

考点：

- 可数集的性质：有限个可数集的笛卡尔积可数；可数集的可数并仍可数
- 代数数（整系数多项式方程的根）的集合是可数的
- 实数集 R 不可数，Z×Z 可数，故不存在从 R 到 Z×Z 的单射

解题思路：

- 2023 Q1c：每个整系数三次方程至多有 3 个实根；整系数三次方程可数（(a,b,c,d) Z , Z 可数），故解集可数
- 2023 Q1f：若存在单射 f: R → Z×Z，则 |R| |Z×Z|。但 |R| = c（连续统势），|Z×Z| = ，矛盾
- 2024 Q1e：同理，整系数二次方程可数，每个至多 2 个根，解集可数
- 2024 Quiz4 Q1d：Q^3 可数（Q 可数，有限笛卡尔积保持可数）， 个点

题型 6：函数——定义、复合与映射计数

出现年份：2024, 2025 **题目示例**：

- 2024 Q3：已知 g: A→B 和 f: B→C（具体映射集合给出），求 f g
- 2025 Q4：构造从 A=1,2 到 B=a,b 的所有函数，指出哪些是双射、满射

考点：

- 函数复合的定义：(f g)(x) = f(g(x))
- 函数作为映射集合的表示
- 单射 (injection)、满射 (surjection)、双射 (bijection) 的判定
- 从 m 元集到 n 元集的映射总数：n

解题思路：

- **2024 Q3**：对每个 x A，找 g(x)，再代入 f。例如 g(1)=b，f(b)=7，所以 (f g)(1)=7。最终 f g = (1,7),(2,10),(3,10),(4,7)
- **2025 Q4**：A→B 的函数共 2^2=4 个。双射要求一一对应（a b 时 f ,f 是双射）；若允许 a=b 则可重集，4 个都满射但无双射

题型 7：集合基数计算

出现年份：2024 **题目示例**：

- 2024 Q7：A = x + 2x + 3x | x , B = x is positive integer < 2024，求 |A B|

考点：

- 取整函数（floor/ceiling）的性质
- 通过分段分析确定集合的元素分布

解题思路：

- 对 x 在不同区间上分析 x 、 2x 、 3 x 的值，找出 A 中所有正整数
- 例如当 x (0,1] 时， x =1， 2x =1 或 2， 3x =1,2 或 3，子集和为 3 6
- 综合所有区间段，找出 A 与 1,2,...,2023 的交集大小

题型 8：序列与求和

出现年份：期末回忆卷 **题目示例**：

- 生成函数的展开（来自期末回忆）
- 递推关系（来自期末回忆）

考点：

- 序列的概念
- 求和记号与运算

解题思路：这些内容在 Quiz 2 中大量出现，但属于 Ch2 的基础概念。基本题型涉及生成函数的展开、求通项公式等。

11.2.3 Ch3 Algorithms

题型 1：时间复杂度 Big-O/Big-Theta 判断（True/False）

出现年份：2022, 2024, 2025 **题目示例**：

- 2022 Q7a：打印 1,...,n 的所有大小为 3 的子集的算法复杂度
- 2022 Q7b：线性搜索的最佳情况复杂度
- 2024 Q1f：线性搜索找最小数的复杂度是 Θ(n log n) [FALSE，应为 Θ(n)]
- 2025 Q1e：n · ¹ 是 O((log . n)) [FALSE]

考点：

- Big-O、Big-Omega、Big-Theta 的定义
- 常见复杂度的大小比较：常数 < 对数 < 多项式 < 指数 < 阶乘
- 多项式函数始终占优对数函数：n = Ω((log n)) 对任意 >0, k>0
- 算法分析中的最佳/最差/平均情况复杂度

解题思路：

- 2022 Q7a：从 n 个元素中选 3 个的组合数为 C(n,3) = n(n-1)(n-2)/6 = O(n^3)
- 2022 Q7b：线性搜索的最佳情况是第一个元素就是目标，只需 1 次比较，复杂度 O(1)（常数时间）
- 2024 Q1f：找最小数需要扫描整个数组一次，共 n-1 次比较，复杂度 Θ(n)
- 2025 Q1e：n · ¹ 是多项式函数，(log . n) 是对数函数。任何 >0 次幂的多项式增长速度都超过任意高次的对数函数，故 n · ¹ 不是 O((log n))

题型 2：按 Big-O 增长率排序

出现年份：2023, 2024 **题目示例**：

- 2023 Q4：将下列 10 个函数按 Big-O 从小到大的顺序排列：(1.2) , 7n +n+323, (log n)³, 3 , log(log n), n²(log n)³, 3 (n³+1), n³+n(log n)², 1000000, 10n!
- 2024 Q6：将 9 个函数排序：(1.01) , 10n!, (log n)³, 2 , log(log n), 999n²(log n)³, (n +1)/(n³+3), n³+n(log n)², 9999

考点：

- 常见函数按增长率排序：常数 < 对数 < 多对数 < 线性 < nlogn < 多项式 < 指数 < 阶乘
- 函数化简与渐近等价比较
- 有理函数的约化

解题思路：

- 按增长率从低到高排序：常数 < log log n < (log n)³ < 多项式（按指数排序）< 指数（按底数排序）< n! < ...
- 注意化简：(n +1)/(n³+3) n（大 n 时趋近于 n），所以与 n 同级
- 使用比值法或极限法比较两个函数的增长率

排序模板（由慢到快）：常数 < log log n < (log n) < n < n (log n) < n (c>a) < b (b>1) < n!

题型 3：具体算法复杂度分析

出现年份：2022 **题目示例**：

- 2022 Q7：分析具体算法复杂度（子集枚举、线性搜索）

考点：

- 将算法步骤转化为数学计数
- 最好情况、最坏情况、平均情况分析

解题思路:

- 子集枚举：组合计数。大小为 k 的子集个数为 C(n,k)，打印每个需 O(k) 时间
- 线性搜索最佳情况：O(1)，目标在第一个位置

11.2.4 附录：各年份试卷对应题目速查

年份	试卷	题目编号	章节	题型
2022	Quiz 1	Q1a,c	Ch1	命题逻辑等价/永真式判断
2022	Quiz 1	Q1b,d	Ch2	集合恒等式判断
2022	Quiz 1	Q2	Ch1	谓词逻辑翻译
2022	Quiz 1	Q3	Ch1	受限联结词表达
2022	Quiz 1	Q4	Ch1	主析取/合取范式
2022	Quiz 1	Q5	Ch2	集合分配律归纳证明
2022	Quiz 1	Q6	Ch2	康托尔对角化 (不可数)
2022	Quiz 1	Q7	Ch3	Big-O 时间复杂度分析
2022	Quiz 1	Q8	Ch1	数学归纳法
2023	Quiz 1	Q1a,e,f	Ch1/Ch2	逻辑等价/真值/函数存在性判断
2023	Quiz 1	Q1b,d	Ch2	集合恒等式/幂集判断
2023	Quiz 1	Q1c	Ch2	可数性判断
2023	Quiz 1	Q2	Ch1	量词等价选择题
2023	Quiz 1	Q3	Ch1	受限联结词 (含 NOR)
2023	Quiz 1	Q4	Ch3	按 Big-O 排序
2023	Quiz 1	Q5	Ch1	主析取/合取范式
2023	Quiz 1	Q6	Ch1	有理数子集证明
2023	Quiz 1	Q7	Ch1	Fibonacci 归纳法
2023	Quiz 1	Q8	Ch1	取石子策略证明
2024	Quiz 1	Q1a,c	Ch1	命题逻辑等价/真值判断
2024	Quiz 1	Q1b	Ch2	对称差结合律
2024	Quiz 1	Q1d,e	Ch2	不可数/可数判断
2024	Quiz 1	Q1f	Ch3	Big-Theta 判断
2024	Quiz 1	Q2	Ch1	谓词逻辑翻译
2024	Quiz 1	Q3	Ch2	函数复合
2024	Quiz 1	Q4	Ch1	NAND 表达命题
2024	Quiz 1	Q5	Ch1	主析取/合取范式
2024	Quiz 1	Q6	Ch3	按 Big-O 排序
2024	Quiz 1	Q7	Ch2	集合基数计算
2024	Quiz 1	Q8	Ch1	逆否命题证明
2024	Quiz 1	Q9	Ch1	数学归纳法
2024	Quiz 4	Q1a,b	Ch1	受限联结词/谓词否定
2024	Quiz 4	Q1d	Ch2	可数性判断 (Q^3)
2025	Quiz 1	Q1a,c,d,g	Ch1	谓词等价/整性/真值/同余
2025	Quiz 1	Q1b	Ch2	集合恒等式
2025	Quiz 1	Q1e,f	Ch3/Ch2	Big-O/不可数判断
2025	Quiz 1	Q2	Ch1	受限联结词（含 ）
2025	Quiz 1	Q3	Ch1	主合取范式
2025	Quiz 1	Q4	Ch2	函数映射枚举
2025	Quiz 1	Q7	Ch2	集合分配律归纳证明
2025	Quiz 1	Q8	Ch1	Fibonacci 和归纳法

11.2.5 核心考点统计

考点	出现频次	重要度
命题等价/永真式判断 (T/F)	4 年	
受限联结词表达（NAND/NOR/¬+ /¬+ ）	4 年	
Full DNF / Full CNF	4 年	
数学归纳法证明	4 年	
谓词逻辑翻译（自然语言 → 符号）	2 年	
量词等价与真值判断	2 年	
集合恒等式判断 (T/F)	3 年	
康托尔对角化/不可数性	3 年	
可数性判断	2 年	
Big-O 概念判断 (T/F)	3 年	
Big-O 增长率排序	2 年	
集合分配律归纳证明	2 年	
函数概念（复合/映射）	2 年	
集合基数计算	1 年	

11.3 Exam Review: Ch4 (Number Theory & Cryptography) + Ch5 (Induction & Recursion)

整理自郑文庭班历年小测真题（2022-2025），仅收录 Ch4-Ch5 相关题目。

11.3.1 Ch4: Number Theory & Cryptography (数论与密码学)

4.1 整除与模运算 / 4.3 素数、GCD、Euler 函数

Euler's Totient Function (欧拉函数) | Quiz 1 2025 Q5 **题目**: If all the positive integers that are relatively prime with 77 are arranged into a strictly increasing sequence, find the 600th term of this sequence. **考点**:

- 与 n 互质的正整数构成一个**缩系 (reduced residue system / reduced set of residues)**，其元素个数为 $\varphi(n)$（欧拉函数）。
- $\varphi(77) = \varphi(7 \times 11) = (7-1) \times (11-1) = 60$.
- 每 77 个连续整数中恰有 60 个与 77 互质，即序列以 60 为**周期**，且每增加 60 项，数值增加 77：$a_{n+60} = a_n + 77$.
- 求第 600 项：$a_{600} = a_{60} + 77 \times \left(\frac{600}{60} - 1\right) = a_{60} + 77 \times 9 = 769$.

相关知识点:

- $\varphi(p) = p - 1$ 当 p 为素数。
- $\varphi(p^k) = p^k - p^{k-1}$.
- $\varphi(mn) = \varphi(m)\varphi(n)$ 当 $\gcd(m, n) = 1$（积性函数）。
- **缩系**：模 n 的缩系是与 n 互质的剩余类构成的集合。

4.4 解同余方程（中国剩余定理）

Chinese Remainder Theorem (CRT) | Quiz 1 2025 Q6 **题目**: Use the construction in the proof of the Chinese Remainder Theorem to find all solutions to the system of congruences:

$$\begin{cases} x \equiv 1 \pmod 3 \\ x \equiv 2 \pmod 5 \\ x \equiv 3 \pmod 8 \end{cases}$$

解法步骤:

1. $m_1 = 3,\ m_2 = 5,\ m_3 = 8,\ m = 3 \times 5 \times 8 = 120$.
2. $M_1 = \frac{m}{m_1} = 40,\ M_2 = \frac{m}{m_2} = 24,\ M_3 = \frac{m}{m_3} = 15$.
3. 求 y_k 满足 $M_k y_k \equiv 1 \pmod{m_k}$:

- $40y_1 \equiv 1 \pmod 3 \Rightarrow y_1 = 1$
- $24y_2 \equiv 1 \pmod 5 \Rightarrow y_2 = 4$
- $15y_3 \equiv 1 \pmod 8 \Rightarrow y_3 = 7$

1. $x \equiv \sum a_k M_k y_k = 1 \times 40 \times 1 + 2 \times 24 \times 4 + 3 \times 15 \times 7 = 40 + 192 + 315 = 547 \equiv 67 \pmod{120}$.

答案: $x \equiv 67 \pmod{120}$.

CRT 应用：连续整除问题 | Quiz 4 2024 Q2 **题目**: There are three consecutive positive integers that are divisible by 5, 7, and 11 respectively （依次能被 5, 7, 11 整除）. Find all the solutions. **分析**: 设第一个数为 n，则:

$$\begin{cases} n \equiv 0 \pmod 5 \\ n + 1 \equiv 0 \pmod 7 \Rightarrow n \equiv 6 \pmod 7 \\ n + 2 \equiv 0 \pmod{11} \Rightarrow n \equiv 9 \pmod{11} \end{cases}$$

考点: 将连续整除条件转化为标准 CRT 形式。$m = 5 \times 7 \times 11 = 385$，解出 n 模 385 的唯一解。

4.5 密码学（费马小定理）

Fermat's Little Theorem (费马小定理) | Quiz 1 2025 Q1(g) **题目**:（判断）$2026^{2025} \equiv 1 \pmod{2027}$. **分析**:

- 2027 是素数，2025 与 2027 互质。
- 由费马小定理：若 p 为素数且 $p \nmid a$，则 $a^{p-1} \equiv 1 \pmod p$.
- 这里 $a = 2025$，$p = 2027$，$p - 1 = 2026$。
- 注意指数是 2025 而不是 2026！$2025^{2026} \equiv 1 \pmod{2027}$ 才成立。
- 但是 $2025 \equiv -2 \pmod{2027}$，$2026 \equiv -1 \pmod{2027}$，计算 $2026^{2025} \equiv (-1)^{2025} = -1 \equiv 2026 \pmod{2027}$.
- 实际上题目可能标记为 True（答案中说" 注意到 2027 是素数，且 2025<2027，所以...2025^(2027-1)　1 (mod 2027)"——但原题是 2026^{2025}，答案可能写的是 2025^{2026}）。

重要:

- 费马小定理：$a^{p-1} \equiv 1 \pmod p$ 对素数 p 且 $p \nmid a$ 成立。
- 易错点：指数必须是 $p - 1$，底数 a 不能被 p 整除。

11.3.2 Ch5: Induction & Recursion (归纳与递归)

5.1 数学归纳法（基本归纳法）

不等式归纳 | Quiz 1 2022 Q8 **题目**: Use mathematical induction to prove that $n^{n+1} > (n+1)^n$ for all positive integer $n \geq 3$. **考点**:

- Basis step: $n = 3$，验证 $3^4 = 81 > 4^3 = 64$，成立。
- Inductive step: 假设 $k^{k+1} > (k+1)^k$，证 $(k+1)^{k+2} > (k+2)^{k+1}$.
- 关键技巧：从 $k^{k+1} > (k+1)^k$ 两边同乘适当因子，或等价变形为 $\frac{k^{k+1}}{(k+1)^k} > 1$，然后证明 $\frac{(k+1)^{k+2}}{(k+2)^{k+1}} > 1$.

平均不等式归纳 | Quiz 1 2024 Q9 **题目**: Use induction to prove that: if $x > 0, y > 0$, then

$$\frac{x^n + y^n}{2} \geq \left(\frac{x+y}{2}\right)^n$$

for all positive integers n. **考点**:

- Basis step: $n = 1$ 时取等号。
- Inductive step: 假设 $n = k$ 成立，证明 $n = k + 1$ 成立。
- 常用技巧：$\frac{x^{k+1} + y^{k+1}}{2} = \frac{x \cdot x^k + y \cdot y^k}{2}$，结合归纳假设和 AM-GM 或排序不等式。

分配律的推广 | Quiz 1 2022 Q5 / Quiz 1 2025 Q7 **Quiz 1 2022 Q5**: Prove that the distributive law

$$A \cap (A_1 \cup A_2 \cup \cdots \cup A_n) = (A \cap A_1) \cup (A \cap A_2) \cup \cdots \cup (A \cap A_n)$$

is true for all $n > 2$. **Quiz 1 2025 Q7**: Prove that $A_1 \cup (A_2 \cap \cdots \cap A_n) = (A_1 \cup A_2) \cap \cdots \cap (A_1 \cup A_n)$ for all $n > 2$. **考点**: 集合运算分配律（De Morgan / 分配律）的归纳推广。

- Basis step: $n = 2$（或 $n = 3$，视题目要求）时的分配律已知成立。
- Inductive step: 假设对 $n = k$ 成立，将 $n = k + 1$ 的情形分成 $(A_1 \cup A_2 \cup \cdots \cup A_k) \cup A_{k+1}$ 再应用归纳假设。
- 本质是反复应用二元分配律。

几何配对问题 | Quiz 4 2024 Q5 **题目**: There are n red points and n green points on the plane, any three of them are not collinear. Use induction to prove: These $2n$ points can be connected in pairs to form n non-intersecting line segments. Each line segment connects a red point and a green point. **考点**:

- 归纳法在几何组合中的应用。
- Inductive step: 找到一条直线将点集分为非空的两部分（各含 k 个红点和 k 个绿点），分别应用归纳假设。
- 关键思想：取最左下的红点，找一个绿点与之配对使连线不与其他点相交。

5.2 强归纳法（Strong Induction）

斐波那契数表示 | Quiz 1 2025 Q8 / Quiz 4 2022 Q6 **题目**: Prove that every positive integer ($n > 2$) can be expressed as the sum of different Fibonacci numbers. **考点**:

- **Fibonacci 数定义**：$f_1 = 1, f_2 = 2, f_3 = 3, f_4 = 5, \ldots$（注意此处 $f_1 = 1, f_2 = 2$，区别于标准定义 $F_1 = 1, F_2 = 1$）。
- **方法一（普通归纳法）**:
- Basis: $n = 3 = f_1 + f_2$，成立。
- 假设 n 可表示为 $f_{i_1} + f_{i_2} + \cdots + f_{i_k}$（互异），考虑 $n + 1$.
- 若 $i_1 > 1$，直接加 $f_1 = 1$ 即可。
- 若 $i_1 = 1$，利用 Fibonacci 性质 $f_m + f_{m+1} = f_{m+2}$ 合并相邻项。
- **方法二（强归纳法）**:
- 假设所有小于 n 且大于 2 的正整数均可表示。
- 设 f 为小于 n 的最大 Fibonacci 数，考虑 $n - f$.
- 若 $n - f = 1$ 或 2，则 $n = f + 1$ 或 $n = f + 2$，是两不同 Fib 数之和。
- 若 $n - f \geq 3$，由强归纳假设 $n - f$ 可表示，且所有加数都小于 f，合并即得 n 的表示。

球合并问题 | 期末回忆 **题目**: 把 2^k 个球放到若干个包里。可以通过下面的规则合并包:

- 若两个包的球的个数相同，则可以直接将其合并为一个包。
- 若两个包的球个数不同，分别为 m, n ($m > n$)，则可将两个包的个数变为 $m - n, 2n$。
- 利用归纳法证明：这里存在一个合适的算法将 2^k 个球合并到一个包上。

考点:

- Strong induction on k.
- Basis: $k = 0$（只有一个球）或 $k = 1$（两个球，直接合并）显然成立。
- Inductive step: 将 2^{k+1} 个球分成两堆各 2^k 个，分别用归纳假设合并成一堆，再处理两堆的合并。

5.3 递归定义 / 母函数与递推

递推关系 | Quiz 4 2022 Q3 / Quiz 4 2024 Q4 **Quiz 4 2024 Q4**: Use generating functions to solve the recurrence relation $a_k = 5a_{k-1} - 6a_{k-2}$ with initial conditions $a_0 = 6$ and $a_1 = 30$. **考点**:

- 特征方程：$r^2 - 5r + 6 = 0$，根为 $r = 2, 3$.
- 通解形式：$a_n = \alpha \cdot 2^n + \beta \cdot 3^n$.
- 代入初值解 α, β.

Quiz 4 2022 Q3: $a_n = c_1 a_{n-1} + c_2 a_{n-2} + c_3$，已知 $a_0 = 0, a_1 = 1, a_2 = 4, a_3 = 11, a_4 = 26$，求通解。**考点**:

- 先由前几项反解系数 c_1, c_2, c_3.
- 然后解非齐次线性递推（齐次解 + 特解）。

11.3.3 题型总结与答题技巧

Ch4 常见题型

题型	频次	关键方法
CRT 解同余方程组	高频	$M_k y_k \equiv 1 \pmod{m_k}$ 求逆元
Euler 函数 + 缩系	高频	$\varphi(n)$ 周期性，$a_{n+\varphi(n)} = a_n + n$
费马小定理判断	中频	$a^{p-1} \equiv 1 \pmod p$ 条件
RSA 相关	未考	但需掌握密钥生成、加密/解密过程

Ch5 常见题型

题型	频次	关键方法
不等式归纳证明	高频	Basis step 验证 + 代数变形
集合/分配律的归纳推广	高频	反复应用二元分配律
强归纳法（Fibonacci 表示等）	高频	取最大小于 n 的 Fib 数
几何组合归纳	中频	构造分割线，分治归纳
递推关系求解	中频	特征方程 / 母函数

易错点提醒

1. **CRT 求逆元**：$M_k y_k \equiv 1 \pmod{m_k}$ 中的 y_k 是 M_k 在模 m_k 下的乘法逆元，可用扩展欧几里得算法或试凑法求解。
2. **费马小定理指数**：指数必须是 $p-1$ 而非 p，且底数不能是 p 的倍数。
3. **归纳法的 Basis step**：务必检查题目要求的起始值（可能从 $n = 3$ 而不是 $n = 1$ 开始）。
4. **强归纳假设**：假设所有小于 n 的命题成立，而不仅仅是 $n - 1$。
5. **Fibonacci 数表示**：注意题目定义的 f_1 值（有时 $f_1 = 1, f_2 = 1$，有时 $f_1 = 1, f_2 = 2$）。

11.3.4 附录：历年 Ch4/Ch5 题目索引

试卷	题号	章节	考点
Quiz 1 2022	Q5	Ch5	归纳法证明分配律推广
Quiz 1 2022	Q8	Ch5	不等式归纳 $n^{n+1} > (n+1)^n$
Quiz 1 2024	Q9	Ch5	平均不等式归纳
Quiz 1 2025	Q1(g)	Ch4	费马小定理判断
Quiz 1 2025	Q5	Ch4	缩系 + Euler 函数求第 600 项
Quiz 1 2025	Q6	Ch4	中国剩余定理（CRT）
Quiz 1 2025	Q7	Ch5	归纳法证明分配律推广
Quiz 1 2025	Q8	Ch5	强归纳法：Fibonacci 数表示
Quiz 4 2022	Q3	Ch5	线性递推关系求解
Quiz 4 2022	Q6	Ch5	强归纳法：Fibonacci 数表示
Quiz 4 2024	Q2	Ch4	CRT 应用（连续整除）
Quiz 4 2024	Q4	Ch5	母函数解递推关系
Quiz 4 2024	Q5	Ch5	归纳法：红绿点配对连线
期末回忆	-	Ch5	强归纳法：球合并问题

11.4 Ch6 & Ch8 考点笔记 – 郑文庭班历年小测真题整理

11.4.1 目录

（计数） (#ch6-counting 计数)

- 1.1 排列组合基础（Permutations & Combinations）
- 1.2 鸽巢原理（Pigeonhole Principle）
- 1.3 容斥原理（Inclusion-Exclusion Principle）
- 1.4 综合计数问题

Counting (#ch8-recurrence-relations–advanced-counting)

- 2.1 递推关系求解（Linear Recurrence Relations）

- 2.2 生成函数（Generating Functions）
- 2.3 斐波那契数列相关证明

考试 Tips (# 考试 tips)

11.4.2 Ch6: Counting（计数）

排列组合基础

题型 1: 基础计数（乘法原理 + 组合数） Quiz 4 (2023-2024) Q3(e) – 忽略花色，从 52 张牌中取 3 张：

把数字相同但花色不同的牌视为不可分辨，即从 13 个数字中选 3 个数字。直接组合数：

$$\binom{13}{3} = 286$$

期末回忆 – 升序字符串计数：给定 $C_5^3 C_2^2 C_4^1$ 等，要求升序字符串有多少个。> 思路：先选择数字组合，升序排列唯一确定，即组合数问题。

题型 2: 扑克牌牌型计数（xxxyy 型、xxyyz 型） Quiz 4 (2023-2024) Q3(a) – xxyyy 牌型（四条）：

How many poker hands of 5 cards contain four cards of the same value? > 解法：1. 选 4 张相同数字：先选数字 $C(13, 1)$，4 张花色已经确定（必须全取）2. 剩 1 张从其余 12 个数字中选，4 种花色任选

$$\binom{13}{1} \times \binom{12}{1} \times \binom{4}{1} = 13 \times 12 \times 4 = 624$$

Quiz 4 (2023-2024) Q3(b) – xxyyz 牌型（两对）：How many poker hands of 5 cards contain 2 pairs (but not 3 of same value)? > 解法：1. 选两个数字作为对子：$\binom{13}{2}$ 2. 每个对子从 4 花色选 2：$\binom{4}{2} \times \binom{4}{2}$ 3. 剩 1 张从其他 11 个数字选，4 花色任选：$\binom{11}{1} \times \binom{4}{1}$

$$\binom{13}{2} \times \binom{4}{2}^2 \times \binom{11}{1} \times \binom{4}{1}$$

鸽巢原理（Pigeonhole Principle）

题型 3: 鸽巢原理基础应用 Quiz 4 (2023-2024) Q3(c) – 确保出现对子：

How many playing cards do you need to take at random to ensure that there is a pair among them? > 解法：13 个数字，最坏情况取 13 张各不同。第 14 张必然与某张重复。

$$\text{答案} = 13 + 1 = 14$$

Quiz 4 (2023-2024) Q3(d) – 确保出现顺子：How many playing cards do you need to take to ensure that there is a straight（顺子）? (A-2-3-4-5 和 10-J-Q-K-A 都算) > 解法：10 种可能的顺子，每种 4 种花色。最坏情况取到 9 种顺子 × 4 花色 = 36 张而没有第 10 种顺子的完整一套，再取 1 张即可。> 或更严谨：最坏情况取 13 个数字各 3 张（共 39 张），仍未凑出顺子。再取 1 张就必然凑出顺子。

题型 4: 鸽巢原理 + 组合设计 Quiz 3 (2023-2024) Q11 – 8 个学生 8 道判断题：

8 students take a test with 8 true/false questions. No two students make exactly the same choice. Prove that we can remove one question, and still no two students make exactly the same choice. > 证明思路（鸽巢原理 / 组合论证）：- 共有 $2^8 = 256$ 种可能的答题模式，但只有 8 个学生 - 若去掉一题，则可能模式降为 $2^7 = 128$ 种 - 关键：考虑任意两学生，他们在 8 题中的答案模式不同。去掉一题后，若某两学生变成相同，说明他们只在被去掉的那题上有差异 - 若存在两学生只在第 i 题上不同，则第 i 题被去掉时他们就会相同 - 由于只有 8 个学生，最多 $\binom{8}{2} = 28$ 对学生，而题目有 8 道 - 需要论证存在一个题目不被任何一对学生独占为唯一差异

容斥原理（Inclusion-Exclusion Principle）

题型 5: 容斥原理基础 期末回忆 – 容斥原理应用：

求包含所有数字的字符串有多少个（使用容斥原理，总字符串数减去缺某个数字的字符串数，再加回缺两个的...）

期末回忆 – 多条件计数：求出满足下列条件之一的字符串：以"66" 开头；以"5" 开头；以"88" 结尾。> 设集合 A = 以"66" 开头的字符串，B = 以"5" 开头的字符串，C = 以"88" 结尾的字符串。

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

综合计数问题

题型 6: 4-digit Almost-palindromes Quiz 4 (2021-2022) Q2 – 4-digit almost-palindrome 计数：

4-digit palindrome: 千位 = 个位，百位 = 十位（且首位不能为 0）almost-palindrome: 交换恰好一位数字可变成 4-digit palindrome。> 分类：- 情况 1: 千位 = 个位　百位且百位　十位，且改变百位或十位中的一个使其变成回文 - 情况 2: 千位　个位且百位 = 十位，改变千位或个位中的一个 - 注意排除本身就是回文的、以及首位为 0 的情况 > (参考答案方法：分千位 = 个位且十位　百位、千位　个位且十位 = 百位两种情况)

题型 7: Combination with Constraints　Quiz 4 (2023-2024) Q3(f) – 卡牌匹配:

52 cards into a 4x13 grid, prove that by selecting one card from each column, you can always get all 13 values. > 思路：Hall's marriage theorem（霍尔婚姻定理），看作二分图完全匹配问题。每列选一张，要覆盖所有 13 个数字，证明存在完美匹配。

11.4.3 Ch8: Recurrence Relations & Advanced Counting

递推关系求解

题型 1: 齐次线性递推求解　Quiz 4 (2023-2024) Q4 – 使用生成函数解齐次递推:

Use generating functions to solve: $a_k = 5a_{k-1} - 6a_{k-2}$ with $a_0 = 6, a_1 = 30$. > 求解步骤：1. 特征方程：$r^2 - 5r + 6 = 0 \to r = 2, 3$ 2. 通解形式：$a_k = \alpha \cdot 2^k + \beta \cdot 3^k$ 3. 代入初值:

$$> \begin{cases} > a_0 = \alpha + \beta = 6 \\ > a_1 = 2\alpha + 3\beta = 30 > \end{cases} >$$

4. 解得 $\alpha = -12, \beta = 18$ 5. 最终解：$a_k = -12 \cdot 2^k + 18 \cdot 3^k = 18 \cdot 3^k - 12 \cdot 2^k$

题型 2: 非齐次线性递推求解　Quiz 4 (2021-2022) Q3 – 非齐次递推（含常数项）:

$a_n = c_1 a_{n-1} + c_2 a_{n-2} + c_3$，已知 $a_0 = 0, a_1 = 1, a_2 = 4, a_3 = 11, a_4 = 26$。> 求解步骤：1. 先代入求出系数:

$$> \begin{cases} > a_2 = c_1 a_1 + c_2 a_0 + c_3 \Rightarrow 4 = c_1 + c_3 \\ > a_3 = c_1 a_2 + c_2 a_1 + c_3 \Rightarrow 11 = 4c_1 + c_2 + c_3 \\ > a_4 = c_1 a_3 + c_2 a_2 + c_3 \Rightarrow 26 = 11c_1 + 4c_2 + c_3 > \end{cases} >$$

2. 解得：$c_1 = 3, c_2 = -2, c_3 = 2$ 3. 递推：$a_n = 3a_{n-1} - 2a_{n-2} + 2$ 4. 齐次部分特征方程：$r^2 - 3r + 2 = 0 \Rightarrow r = 1, 2$ 5. 齐次通解：$a_n^{(h)} = A \cdot 1^n + B \cdot 2^n = A + B \cdot 2^n$ 6. 非齐次项为常数 2，且 1 是特征根（重数 1），设特解形式 $a_n^{(p)} = Cn$ 7. 代入求 C: $Cn = 3C(n-1) - 2C(n-2) + 2 \Rightarrow C = 1$ (验证：$n = 3(n-1) - 2(n-2) + 2 = 3n - 3 - 2n + 4 + 2 = n + 3 \to$ 需仔细计算) 8. 通解：$a_n = A + B \cdot 2^n + n$ 9. 代入初值 $a_0 = 0, a_1 = 1$ 得:

$$> \begin{cases} > A + B = 0 \\ > A + 2B + 1 = 1 > \end{cases} >\Rightarrow A = 0, B = 0 >$$

10. 最终解：$a_n = n$

验证：$a_0 = 0, a_1 = 1, a_2 = 2, a_3 = 3, a_4 = 4$　与给定值不符，说明上述答案需要修正? > 重新核对：已知 $a_2 = 4$，若 $a_n = n$ 则 $a_2 = 2 \neq 4$。> 从参考资料看，参考答案如下：- 解得 $c_1 = 3, c_2 = -2, c_3 = 2$，递推式 $a_n = 3a_{n-1} - 2a_{n-2} + 2$ - 继续求：齐次部分通解 $a_n^{(h)} = \alpha + \beta \cdot 2^n$ - 由于 1 是特征根，特解设 $a_n^{(p)} = pn$ - 代入：$pn = 3p(n-1) - 2p(n-2) + 2 \to$ 化简得 $pn = pn + p + 2 \to p = -2$ - 特解：$a_n^{(p)} = -2n$ - 通解：$a_n = \alpha + \beta \cdot 2^n - 2n$ - 代入初值:

$$> \begin{cases} > a_0 = \alpha + \beta = 0 \\ > a_1 = \alpha + 2\beta - 2 = 1 > \end{cases} >\Rightarrow \alpha = -3, \beta = 3 >$$

- 最终解：$a_n = -3 + 3 \cdot 2^n - 2n$ > **验证**：$a_2 = -3 + 12 - 4 = 5$　仍不对（已知 $a_2 = 4$）> 我们重新检查，实际上原题中给定 $a_2 = 4$，可以通过直接解方程组:

$$> \begin{cases} > a_2 = c_1 a_1 + c_2 a_0 + c_3 \Rightarrow 4 = c_1 + c_3 \\ > a_3 = c_1 a_2 + c_2 a_1 + c_3 \Rightarrow 11 = 4c_1 + c_2 + c_3 \\ > a_4 = c_1 a_3 + c_2 a_2 + c_3 \Rightarrow 26 = 11c_1 + 4c_2 + c_3 > \end{cases} >$$

由　得 $c_3 = 4 - c_1$，代入　:

$$> \begin{cases} > 11 = 4c_1 + c_2 + 4 - c_1 \Rightarrow 7 = 3c_1 + c_2 \Rightarrow c_2 = 7 - 3c_1 \\ > 26 = 11c_1 + 4(7 - 3c_1) + 4 - c_1 \Rightarrow 26 = 11c_1 + 28 - 12c_1 + 4 - c_1 = 32 - 2c_1 > \end{cases} >$$

$\Rightarrow 2c_1 = 6 \Rightarrow c_1 = 3, c_2 = -2, c_3 = 1 >$ 所以递推式为：$a_n = 3a_{n-1} - 2a_{n-2} + 1 >$ 特征根 r=1, 2。齐次解 $a_n^{(h)} = \alpha + \beta \cdot 2^n$ 非齐次项为常数 1，1 是特征根，设特解 $a_n^{(p)} = pn$ 代入：$pn = 3p(n-1) - 2p(n-2) + 1 \to pn = pn - 3p + 2p + 1 = pn - p + 1 \to p = 1$ 通解：$a_n = \alpha + \beta \cdot 2^n + n$ 代入 $a_0 = 0, a_1 = 1$:

$$> \begin{cases} > \alpha + \beta = 0 \\ > \alpha + 2\beta + 1 = 1 > \end{cases} >\Rightarrow \alpha = 0, \beta = 0 >$$

$a_n = n$? 但 $a_2 = 2 \neq 4$，矛盾。原题数据可能有问题，或者系数计算有误。> **建议**：考试时严格按照" 代入求系数 → 解特征方程 → 齐次解 → 特解 → 初值定系数" 的流程做，数据以考场实际为准。

生成函数（Generating Functions）

题型 3: 生成函数解递推　Quiz 4 (2023-2024) Q4 – 使用生成函数:

$a_k = 5a_{k-1} - 6a_{k-2}, a_0 = 6, a_1 = 30$ > 生成函数法步骤：1. 设 $G(x) = \sum_{k\geq 0} a_k x^k$ 2. 由递推：$\sum_{k\geq 2} a_k x^k = 5\sum_{k\geq 2} a_{k-1}x^k - 6\sum_{k\geq 2} a_{k-2}x^k$ 3. $G(x) - a_0 - a_1 x = 5x(G(x) - a_0) - 6x^2 G(x)$ 4. 代入初值：$G(x) - 6 - 30x = 5xG(x) - 30x - 6x^2 G(x)$ 5. 整理：$G(x) - 6 = 5xG(x) - 6x^2 G(x)$ 6. $G(x)(1 - 5x + 6x^2) = 6$ 7. $G(x) = \frac{6}{1-5x+6x^2} = \frac{6}{(1-2x)(1-3x)}$ 8. 部分分式：$\frac{6}{(1-2x)(1-3x)} = \frac{A}{1-2x} + \frac{B}{1-3x}$ 9. 解得 $A = -12, B = 18$ 10. $G(x) = \frac{-12}{1-2x} + \frac{18}{1-3x} = -12\sum_{k\geq 0} 2^k x^k + 18\sum_{k\geq 0} 3^k x^k$ 11. 所以 $a_k = -12 \cdot 2^k + 18 \cdot 3^k$

期末回忆 – 生成函数综合题：给定序列递推关系或生成函数形式，求通项或特定系数。> 常见形式：- $G(x) = \frac{1}{1-x}$ 对应 $a_n = 1$ - $G(x) = \frac{1}{(1-x)^k}$ 对应 $a_n = \binom{n+k-1}{k-1}$ - $G(x) = \frac{1}{1-ax}$ 对应 $a_n = a^n$

斐波那契数列相关证明

题型 4: 正整数表示为不同 Fibonacci 数之和　Quiz 1 (2025) Q8 – 强归纳法证明:

Prove that every positive integer (n > 2) can be expressed as the sum of different Fibonacci numbers. > 证法一（强归纳法）：1. **Base**: n=3 时，$3 = f_1 + f_2 = 1 + 2$，成立。2. **Inductive step**: 假设对所有 $3 \leq m < n$ 成立。设 f_k 为小于 n 的最大 Fibonacci 数。- 若 $n - f_k = 1$ 或 2，则 $n = f_k + 1$ 或 $n = f_k + 2$，可表示。- 若 $n - f_k \geq 3$，由归纳假设 $n - f_k$ 可表示为不同 Fibonacci 数之和。由于 f_k 是小于 n 的最大 Fibonacci 数，这些 Fibonacci 数均小于 f_k，所以 n 可表示为互异的 Fibonacci 数之和。

证法二（构造型归纳）：处理 n+1 = 1 + (f_i_1 + f_i_2 + ... + f_i_k)：- 若 $i_1 > 1$，则 1 = f_1 与所有已有 Fibonacci 数不同。- 若 $i_1 = 1$，则利用 Fibonacci 性质 $f_m + f_{m+1} = f_{m+2}$，找到连续下标区间合并。

题型 5: 二元递推归纳　期末回忆 – 2^k 个球合并问题:

将 2^k 个球放到若干个包中，通过规则合并包：- 若两个包的球个数相同，可直接合并 - 若两个包球数不同（m > n），可将两包变为 m-n 和 2n > 证明：存在算法将所有球合并到一个包中。> 思路：对 k 使用归纳法。

11.4.4 考试 Tips

Ch6 重点题型

题型	考点	出现年份
基础排列组合	乘法原理、组合数直接计算	2024 Quiz4
扑克牌型计数	xxxyy, xxyyz 型（选数字 × 选花色）	2024 Quiz4
鸽巢原理	最坏情况 + 1 张	2024 Quiz4
容斥原理	字符串计数、多条件并集	期末回忆
Almost-palindrome	分类讨论 + 排除首位 0	2022 Quiz4

Ch8 重点题型

题型	考点	出现年份
齐次线性递推	特征方程 → 通解 → 初值定系数	2024 Quiz4
非齐次线性递推	含常数项（特解形式注意特征根）	2022 Quiz4
生成函数	设 G(x) → 利用递推 → 部分分式展开	2024 Quiz4
斐波那契证明	强归纳法、构造法	2025 Quiz1, 2022 Quiz4

解题模板

递推求解流程:

```
Step 1: 写特征方程 r^k - c_1 r^{k-1} - ... - c_k = 0
Step 2: 求特征根 → 齐次通解
Step 3: 非齐次项 → 设特解（注意特征根导致的重数）
Step 4: 齐通解 + 特解 → 代入初值定系数
```

生成函数解递推流程:

```
Step 1: 设 G(x) = Σ a_n x^n
Step 2: 利用递推关系写出 G(x) 方程
Step 3: 解出 G(x) 有理函数形式
Step 4: 部分分式展开
Step 5: 展开为幂级数，提取系数
```

排列组合解题框架:

```
Step 1: 识别问题类型（排列/组合/容斥/鸽巢）
Step 2: 分类讨论（互斥情况分开算）
Step 3: 乘法原理分步计算
Step 4: 排除重复/非法情况
```

11.5 Ch9-Ch11 离散数学历年小测考点整理（郑文庭班）

整理自 2021-2022、2022-2023、2023-2024、2024-2025 学年 Quiz 2/3/4 及期末回忆卷, 仅包含 Ch9-Relations, Ch10-Graphs, Ch11-Trees.

11.5.1 Ch9: Relations（关系）

9.1 关系及其性质 / 9.3 关系的表示

- **(2022 Quiz3 Q1)** 已知集合 `a,b,c,d` 上的关系 `R = (a,a),(b,a),(b,c),(c,a),(c,c),(c,d),(d,a),(d,c)`, 求其**传递闭包**。用 Warshall 算法或关系矩阵求传递闭包。
- **(2024 Quiz3 Q1b)** `R = (a,a),(a,b),(b,d),(a,d)` 是 `a,b,c,d` 上的关系。求包含 R 的**最小的对称且传递的关系**。
- **(期末回忆)** 给一个有向图, 求**传递闭包**、强连通分量。

9.5 等价关系

- **(2024 Quiz3 Q6)** `|A| = 4`, 求 A 上**不同等价关系的数量** (Bell number / Stirling numbers of the second kind)。
- `B_4 = 15` (或 `S(4,1)+S(4,2)+S(4,3)+S(4,4) = 1+7+6+1=15`)

9.6 偏序关系

- **(2022 Quiz3 Q2)** 在 `a,b,c,d,e,f` 上求包含 `(a,c),(c,c),(c,b),(c,d),(b,e),(b,f)` 的**最小偏序关系**（加上自反、反对称、传递所需的对）, 然后:
- (b) 画 **Hasse 图**
- (c) 列出**极大元** (maximal elements)
- (d) 列出**极小元** (minimal elements)
- (e) 找**最大元** (greatest element) ——可能不存在
- (f) 找**最小元** (least element) ——可能不存在
- (g) 求 `d,e` 的**最小上界** (least upper bound)
- (h) **拓扑排序** (topological sorting)
- **(2024 Quiz3 Q7)** 偏序集 `(2,3,5,6,12,20,27,36,60, |)`（整除关系）:
- (a) 画 Hasse 图
- (b) 求极大元
- (c) 是否有最小元
- (d) 求 `2,3` 的所有上界
- **(2024 Quiz3 Q1a)** `R = (a,a),(a,b),(b,d),(a,d)` 是 `a,b,c,d` 上的关系。求包含 R 的**最小的偏序关系**（补充自反、传递、反对称所需的元素）。
- **(期末回忆)** n=3 的偏序关系有多少种? (偏序的计数问题)

11.5.2 Ch10: Graphs（图）

10.1 图模型 / 10.2 图术语

- **(2024 Quiz3 Q10 / 期末回忆 Q_n) 超立方体 Q_n**:
- 顶点数 = `2^n`
- 边数 = `n * 2^(n-1)`
- 是否为二分图? 是（按奇偶性划分）
- 色数 = `2`（因为超立方体是二分图）
- 是否有欧拉回路/欧拉通路?
- 是否有哈密顿回路/通路?
- 是否为平面图?
- **(2024 Quiz4 Q1e) 支配集** (dominating set): 给定图, 求**最小支配集的顶点数**。
- **(2024 Quiz4 Q1f)** 判断图是否存在**哈密顿回路** (Hamilton circuit)。
- **(2024 Quiz4 Q1h)** 给定有向图, 计算长度为某值的**不同路径数** (cycles allowed)。

10.3 图的表示与同构

- **(2024 Quiz3 Q9)** 判断两个图是否**同构** (isomorphic), 给出理由。
- 常用的同构不变量: 顶点数、边数、度数序列、环长等。
- **(2022 Quiz3 Q8b)** 判断 Fig.4 是否是 Petersen 图（**同构**判定）。

10.4 连通性

- **(2024 Quiz3 Q11)** 8 个学生做 8 道 True/False 题, 已知没有两个人的答案完全相同。证明可以删去一道题, 使得仍然没有两个人的答案完全相同。(鸽巢原理/图的度序列/二分图匹配思想)

10.5 欧拉与哈密顿路径

- **(2022 Quiz4 Q4) 网格图 G_m,n**（m 行 n 列的矩形网格）:
- (a) 哪些 m, n 有**欧拉回路**? （所有顶点度数为偶数——当 m,n 均为奇数时不成立, 因为角点度数为 2, 边点度数可能为奇数。分析: 内点度数为 4, 角点度数为 2, 非角边界点度数为 3（若 m,n>2）。所以只有当 m=1 或 n=1 或 m=n=2 等特殊情况才可能所有点度数为偶数。一般地, G_m,n 有欧拉回路当且仅当 m 和 n 均为偶数或 m=1,n=1 或 (m,n)=(1,2)/(2,1) 等。)
- (b) 哪些有**欧拉通路**但没有欧拉回路? （恰好两个奇度数顶点）

- (c) 哪些有**哈密顿回路**? （m,n 不全为偶数时总有哈密顿回路; 当 m 和 n 均为偶数时没有, 可用" 删去一个点二分图两边不平衡" 证明）
- (d) 哪些有**哈密顿通路**但没有哈密顿回路? （m,n 均为偶数时, 有哈密顿路径但无哈密顿回路）
- (e) G_1,2 有多少棵**生成树**?
- **(2022 Quiz3 Q7 / 答案详解) 循环赛** (round-robin tournament):
- n 个选手每两人比赛一次, 必有胜负。证明可以排成序列 `p1, p2, ..., pn` 使得 `p1` 胜 `p2`, `p2` 胜 `p3`, ..., `p_n-1` 胜 `pn`。即**竞赛图中必存在哈密顿路径**。
- 证明方法: 归纳法。n=2 时显然。假设 n=k 时有哈密顿路径。加入第 k+1 个选手 v: 如果有边从 v 指向 p1 或从 pk 指向 v, 直接加在首尾; 否则一定存在某个 i 使得 p_i 指向 v 且 v 指向 p_i+1, 插入 v 即可。
- **(2022 Quiz3 Q8d / 答案详解) Petersen 图是否是哈密顿图?**
- 答案: **不是**。证明: 用反证法, 假设存在哈密顿回路, 则回路是 10 个点的环, 加上 5 条弦使每个顶点度数为 3。通过分析环上任一边的连接方式, 只能形成两个六元环的连法, 但尝试所有情况都会导出长度为 4 的环（Petersen 图无 4 环）, 矛盾。
- **(2024 Quiz3 Q10c)** Q_n 是否有**哈密顿回路/路径**?
- Q_n (n >= 2) 有哈密顿回路（格雷码构造）。

10.6 最短路径问题

- **(2022 Quiz3 Q4)** 用 **Dijkstra 算法**求顶点 1 到顶点 6 的最短路径长度。
- **(2024 Quiz4 Q1g)** Dijkstra 算法在 n 个顶点的加权图中的**最坏时间复杂度** = `O(n^2)`（使用数组）或 `O((n+m)log n)`（使用堆）。
- **(期末回忆)** 求从源点到各个点的最短路中最长的一条。

10.7 平面图

- **(2022 Quiz3 Q6)** 确定所有使得 **K_r,s 是平面图**的 r, s。
- K_r,s 是平面图当且仅当 r <= 2 或 s <= 2（即 K_2,n, K_n,2, K_1,n 等）。当 r >= 3 且 s >= 3 时 K_r,s 包含 K_3,3 子图, 不是平面图。
- **(2022 Quiz3 Q8c / 答案详解) Petersen 图是非平面图**。
- 用欧拉公式推论: 对于无三角形的平面图, `e <= 2v - 4`。Petersen 图 `v=10, e=15`, 满足 `15 > 2*10 - 4 = 16`? 不对, 15 < 16... 实际上 Petersen 图最小环长为 5, 可以推导出 `e <= (5/3)(v-2)  13.3`, 但 e=15 > 13.3, 矛盾, 故不是平面图。
- **(2022 Quiz4 Q5) 正则多面体** (regular polyhedron):
- (a) 正四面体、正方体、正八面体是否都是**平面图**? 是, 可以画在平面上无交叉边。
- (b) 它们的**色数**:
- 正四面体（4 个顶点, 完全图 K4）-> 色数 = 4
- 正方体（8 个顶点, 立方体图 Q3）-> 色数 = 2（二分图）
- 正八面体（6 个顶点）-> 色数 = 3（含三角形, 但不是 K4）
- (c) 用**欧拉定理**证明正多面体只有 5 种: 正四面体、正方体、正八面体、正十二面体、正二十面体。
- 利用 `v - e + f = 2`, 每个面有 p 条边, 每个顶点有 q 条边, 则 `pf = 2e = qv`, 代入欧拉公式求解整数解。
- **(2024 Quiz4 Q1c)** G 是 20 个顶点的平面连通图, 每个顶点度数为 3, 求**区域数**。
- 握手定理: `2e = sum deg(v) = 20*3 = 60 => e = 30`
- 欧拉公式: `v - e + f = 2 => 20 - 30 + f = 2 => f = 12`

10.8 图的着色

- **(2022 Quiz3 Q8a) Petersen 图的色数** = 3。(Petersen 图是 3-色的, 但不是二分图, 因为包含奇环)
- **(2024 Quiz3 Q10b)** Q_n 的**色数** = 2（Q_n 是二分图, 因为所有边连接奇偶性不同的顶点）。

其他图论考点

- **(2024 Quiz3 Q8) 最大流** (maximum flow):
- 在给定网络中找从源点 A 到汇点 J 的最大流, 计算流量值, 并证明是最大流（用最小割定理）。
- **(期末回忆)** 给一个有向图, 求强连通部分。
- **(期末回忆)** 从 A 点开始, 按字典序找出使各点尽可能联通的树/森林。

11.5.3 Ch11: Trees（树）

11.1 树的定义与性质

- **(2024 Quiz3 Q4) 满 7 叉树** (full 7-ary tree) 有 2024 个顶点, 求叶子数。
- 公式: 对于满 m 叉树, `i = (n-1)/m` 个内点, `l = n - i` 个叶子。
- 或: `n = m*i + 1`, `l = n - i`。若 `2024` 个顶点, 则 `2024 = 7i + 1 => i = 289`, `l = 2024 - 289 = 1735`。
- **(2024 Quiz3 Q5)** 确定所有正整数 r, s 使得 **K_r,s 是树**。
- 树是连通且无环的。K_r,s 是树当且仅当 `r + s = rs + 1`? 不对, 树有 `v-1` 条边。
- K_r,s 有 `r+s` 个顶点, `r*s` 条边。树要求 `rs = (r+s) - 1 => rs - r - s + 1 = 0 => (r-1)(s-1) = 0`。
- 所以 r = 1 或 s = 1, 即 K_1,n 或 K_n,1（星形图）。
- **(期末回忆)** 一棵树有 7 个叶子节点, 3 个内点度数为 3, 其他内点度数为 4, 求度数为 4 的内点数。

- 设度数为 4 的节点有 x 个, 总顶点数 `n = 7 + 3 + x`, 总度数 `sum deg = 7*1 + 3*3 + 4x = 16 + 4x`。
- 树的性质: `sum deg = 2(n-1)` => `16 + 4x = 2(10 + x - 1) = 18 + 2x` => `2x = 2` => `x = 1`。

11.2 树的应用——Huffman 编码

- **(2022 Quiz3 Q5)** 字符频率: a:0.15, b:0.22, c:0.26, d:0.19, e:0.08, f:0.1。用 **Huffman 编码**求平均比特数。
- **(2024 Quiz3 Q3)** 六个字符的频率: 0.09, 0.05, 0.2, 0.25, 0.3, 0.11。用 Huffman 编码求**平均比特数**。
- **(期末回忆)** 哈夫曼编码。

11.2 树的应用——BST / 决策树

- **(2024 Quiz3 Q11)** 8 个学生做 8 道 True/False 题, 证明可以删去一道题后任意两人答案仍不同。(可利用树/二分图思想)

11.3 树的遍历

- **(2024 Quiz3 Q2b)** 给定图, 使用**字母序的深度优先搜索 (DFS)** 求生成树。

11.4 / 11.5 生成树与最小生成树

- **(2022 Quiz3 Q3)** 给加权图, 求**最小生成树 (MST)**。(可能用 Kruskal 或 Prim)
- **(2024 Quiz3 Q2a)** 用 **Kruskal 算法**求最小生成树, 列出边加入顺序。
- **(2024 Quiz3 Q2b) DFS 生成树** (字母序)。
- **(期末回忆)** 求最小生成树。
- **(期末回忆)** 从 A 点开始, 找出使各点尽可能联通的树/森林, 其中边的大小按照字典序比较。

11.5.4 综合高频考点一览

章节	考点	出现次数	年份
Ch9	传递闭包 (Warshall)	2	2022Q3, 期末
Ch9	偏序 / Hasse 图 / 极值元 / 拓扑排序	3	2022Q3, 2024Q3x2
Ch9	等价关系计数	1	2024Q3
Ch9	最小偏序/对称传递关系	1	2024Q3
Ch10	欧拉回路/通路判定 (网格图等)	2	2022Q4, 2024Q4
Ch10	哈密顿回路/通路判定 (竞赛图, Petersen)	3	2022Q3, 2022Q4, 2024Q4
Ch10	超立方体 Q_n (顶点/边/色数/欧拉/哈密顿)	2	2024Q3, 期末
Ch10	最短路径 (Dijkstra)	3	2022Q3, 2024Q4, 期末
Ch10	平面图判定/欧拉公式	3	2022Q3, 2022Q4, 2024Q4
Ch10	图的着色 (色数)	2	2022Q3, 2024Q3
Ch10	图同构	2	2022Q3, 2024Q3
Ch10	最大流 / 最小割	1	2024Q3
Ch10	Petersen 图综合	1	2022Q3
Ch11	Huffman 编码 / 平均比特数	3	2022Q3, 2024Q3, 期末
Ch11	最小生成树 (Kruskal/Prim)	3	2022Q3, 2024Q3, 期末
Ch11	树的基本性质 (树叶/内点计算)	2	2024Q3, 期末
Ch11	K_r,s 树/平面图判定	2	2022Q3 (平面), 2024Q3 (树)
Ch11	DFS/BFS 生成树	1	2024Q3

11.5.5 特别提醒

1. **Petersen 图**几乎是必考综合题: 色数 =3, 非平面 (欧拉公式), 非哈密顿 (详细构造法反证), 同构判定, 10 顶点 15 边 3-正则。
2. **竞赛图** (tournament) 必有哈密顿路径——归纳法构造, 是经典证明题。
3. **网格图 G_m,n**: 欧拉回路/通路和哈密顿回路/通路的判定条件要记清楚, 尤其是 m,n 奇偶性。
4. **欧拉公式**应用极广: 平面图 `v - e + f = 2`, 常用推论 `e <= 3v - 6` (简单平面图) 和 `e <= 2v - 4` (无三角形平面图), 推广到最小环长 g 时 `e <= g(v-2)/(g-2)`。
5. **满 m 叉树**公式: `n = m*i + 1`, 叶子数 = `n - i`。
6. **最小生成树**的两个算法 (Kruskal 和 Prim) 都要会。

第二部分

模拟卷

11.6 DMA 离散数学摸底考试卷

考试范围：Ch1-6, Ch8-11 + Network Flow | 满分 100 | 时间 90 分钟

11.6.1 Part 1: True/False (30%)

评分：答对 +3 分，空白 0 分，答错 -1 分（倒扣制）

Ch1 Logic & Proofs

Q1 . Let $P(x)$ be a predicate. Then $\forall x \exists y P(x,y) \equiv \exists y \forall x P(x,y)$. ( ) **Q2** . The negation of $\forall x(P(x) \to Q(x))$ is $\exists x(P(x) \wedge \neg Q(x))$. ( ) **Q3** . $(p \to q) \wedge (p \to r) \equiv p \to (q \wedge r)$. ( )

Ch2 Sets & Functions

Q4 . $|A \times B| = |A| \cdot |B|$ for any finite sets A and B. ( ) **Q5** . If $f : A \to B$ and $g : B \to C$ are both injective, then $g \circ f$ is injective. ( ) **Q6** . The set of all finite subsets of $\mathbb{N}$ is countable. ( )

Ch3 Algorithms

Q7 . If $f(n) = 3n^2 + 2n + 1$, then $f(n) = \Theta(n^2)$. ( ) **Q8** . $n! = O(2^n)$. ( )

Ch4 Number Theory

Q9 . If $a \equiv b \pmod m$ and $c \equiv d \pmod m$, then $ac \equiv bd \pmod m$. ( ) **Q10** . $\gcd(a,b) \cdot \mathrm{lcm}(a,b) = ab$ for all positive integers a, b. ( )

Ch5 Induction

Q11 . The Well-Ordering Principle is equivalent to the Principle of Mathematical Induction. ( ) **Q12** . In strong induction, to prove $P(n)$ we assume $P(k)$ is true for all $k < n$, not just $P(n-1)$. ( )

Ch6 Counting

Q13 . $C(n,r) = P(n,r)/r!$ for all $0 \le r \le n$. ( ) **Q14** . By the pigeonhole principle, among any 13 people, at least 2 were born in the same month. ( )

Ch8 Recurrence Relations

Q15 . The characteristic equation of $a_n = 3a_{n-1} - 2a_{n-2}$ has roots 1 and 2. ( ) **Q16** . The number of derangements of n objects is approximately $n!/e$. ( )

Ch9 Relations

Q17 . Every equivalence relation on a set A partitions A. ( ) **Q18** . A relation that is symmetric and transitive must also be reflexive. ( )

Ch10 Graphs

Q19 . K_5 is a planar graph. ( ) **Q20** . A connected multigraph has an Euler circuit iff every vertex has even degree. ( ) **Q21** . The chromatic number of a bipartite graph is at most 2. ( ) **Q22** . By the Handshaking Theorem, every graph has an even number of vertices of odd degree. ( )

Ch11 Trees

Q23 . Every tree with n vertices has exactly $n-1$ edges. ( ) **Q24** . In a full binary tree, the number of leaves equals the number of internal vertices plus 1. ( )

Network Flow

Q25 . By the Max-Flow Min-Cut Theorem, the value of a maximum flow equals the capacity of a minimum cut. ( )

11.6.2 Part 2: Short Problems (40%)

Ch1 (6%)

Q26 . Write an equivalent proposition of $p \oplus q$ using only the NAND operator $|$.

Ch2 (6%)

Q27 . Let $A = \{1,2,3\}$, $B = \{2,3,4\}$. Compute: (a) $A \oplus B$ (symmetric difference) (b) $\mathcal{P}(A \cap B)$ (power set)

Ch3 (4%)

Q28 . Order the following functions by growth rate (slowest first):

$$(\log n)^2, \quad n \log n, \quad n^2, \quad 2^n, \quad n!, \quad n^{1/\log n}$$

Ch4 (6%)

Q29 . (a) Compute $\gcd(252, 198)$ using the Euclidean algorithm. (b) Express $\gcd(252, 198)$ as a linear combination of 252 and 198 (Bézout coefficients).

Ch5 (6%)

Q30 . Prove by induction: $\sum_{i=1}^{n}(2i-1) = n^2$ for all $n \ge 1$.

Ch6 (6%)

Q31 . How many strings of 4 lowercase letters: (a) contain at least one 'a'? (b) have all letters distinct? (c) start with a vowel (a, e, i, o, u) and end with a consonant?

Ch8 (6%)

Q32 . Find the general solution to the recurrence relation:

$$a_n = 5a_{n-1} - 6a_{n-2} \quad (n \ge 2)$$

with initial conditions $a_0 = 2, a_1 = 7$.

11.6.3 Part 3: Long Problems (30%)

Ch9 Relations (10%)

Q33 . Let $A = \{1,2,3,4,6,12\}$ and define a relation R on A by:

$$aRb \iff a \mid b \quad \text{(a divides b)}$$

(a) (30%) Prove that R is a partial order. (b) (30%) Draw the Hasse diagram. (c) (20%) Find all maximal, minimal, maximum, and minimum elements. (d) (20%) Is (A, R) a lattice? Justify.

Ch10 Graphs (10%)

Q34 . Consider the following weighted graph:

```
Edges with weights:
A-B: 4, A-C: 2, B-C: 1, B-D: 5, C-D: 8, C-E: 10, D-E: 2, D-F: 6, E-F: 3
```

(a) (50%) Use Dijkstra's algorithm to find the shortest path from A to F. Show each step. (b) (50%) Does this graph have an Euler circuit? An Euler path? An Hamilton circuit? Justify each answer.

Ch11 Trees (10%)

Q35 . A file contains 6 symbols with frequencies: A: 15%, B: 25%, C: 5%, D: 20%, E: 10%, F: 25%. (a) (50%) Construct a Huffman code for these symbols. Show the merging steps. (b) (30%) What is the average number of bits per symbol using your Huffman code? (c) (20%) If we used a fixed-length code, how many bits per symbol would be needed?

11.6.4 Answer Quick-Check

Q	Ans	Q	Ans	Q	Ans	Q	Ans
1	F	8	F	15	T	22	T
2	T	9	T	16	T	23	T
3	T	10	T	17	T	24	T
4	T	11	T	18	F	25	T
5	T	12	T	19	F		
6	T	13	T	20	T		
7	T	14	T	21	T		

| Q26 | $(p \mid (q \mid q)) \mid ((p \mid p) \mid q)$ | | Q27a | $\{1,4\}$ | | Q27b | $\{\emptyset, \{2\}, \{3\}, \{2,3\}\}$ | | Q28 | $n^{1/\log n} < (\log n)^2 < n \log n < n^2 < 2^n < n!$ | | Q29a | $\gcd(252, 198) = 18$ | | Q29b | $18 = 4 \cdot 252 - 5 \cdot 198$ | | Q30 | Basis $n = 1$: $1 = 1^2$; Inductive: $\sum_{i=1}^{k+1}(2i-1) = k^2 + (2k+1) = (k+1)^2$ | | Q31a | $26^4 - 25^4 = 66351$ | | Q31b | $P(26, 4) = 358800$ | | Q31c | $5 \cdot 21 \cdot 26^2 = 70980$ | | Q32 | $a_n = 3 \cdot 2^n - 3^n$ | | Q33 | R is reflexive, antisymmetric, transitive; Hasse: 1→2→6→12, 1→3→6, 1→4→12; Max=Maximal=12, Min=Minimal=1; Not a lattice (e.g. lub$(2,4)$ does not exist in the "2,4 don't both divide anything below 12" sense —actually lub=12, glb(2,4)=1, but check: glb(6,4) = 2, lub(4,3)... 3 and 4 have no common upper bound less than 12, so lub(3,4)=12 exists, glb always exists? Actually (2,4) has glb 2... this is actually a lattice because it's the divisor poset of 12). Actually the divisor poset of any positive integer IS always a lattice (lub = lcm, glb = gcd). So yes, it is a lattice. | | Q34a | A-C-B-D-F = 2+1+5+6 = 14 or A-C-B-D-E-F = 2+1+5+2+3 = 13. Shortest: A-B-D-E-F = 4+5+2+3 = 14. Wait: A-C=2, from C: C-B=1(B=3), from B: B-D=5(D=8), from D: D-E=2(E=10), from E: E-F=3(F=13). Path: A→C→B→D→E→F, length 2+1+5+2+3=13. | | Q34b | All degrees even? deg(A)=2, B=3, C=3, D=3, E=3, F=2. Four odd-degree vertices → no Euler circuit, no Euler path. Dirac: min deg = 2 < 6/2 → insufficient for Hamilton guarantee, but doesn't prove non-existence. Try: A-C-B-D-F-E-A is not a circuit. Check: A-B-D-F-E-C-A: yes! All 6 vertices exactly once and back to A. Yes, Hamiltonian circuit exists. | | Q35a | Merge: C+E=15% → A+CE=30% → D+B=45% → (A_C_E)+(B_D)=30%+45%=N/A... Actually step by step: C(5)+E(10)=15, A(15)+15=30, D(20)+B(25)=45, 30+F(25)=55, 45+55=100. Codes: B:00, D:01, F:10, A:110, E:1110, C:1111. | | Q35b | Avg bits = 2×0.25 + 2×0.20 + 2×0.25 + 3×0.15 + 4×0.10 + 4×0.05 = 0.5+0.4+0.5+0.45+0.4+0.2 = 2.45 | | Q35c | Fixed-length for 6 symbols needs ceil(log2 6) = 3 bits. |

11.7 DMA 摸底卷—详细题解

11.7.1 Part 1: True/False 解析

Ch1

Q1 (F). $\forall x\exists y$ 和 $\exists y\forall x$ 一般不可交换。反例：$P(x,y)=$ "$x<y$"，论域 $\mathbb{N}$。$\forall x\exists y(x<y)$ 为真（每个数都有更大的数），但 $\exists y\forall x(x<y)$ 为假（不存在比所有数都大的数）。**Q2 (T)**. 量词否定：$\neg\forall x(P\to Q)\equiv\exists x\neg(P\to Q)\equiv\exists x\neg(\neg P\vee Q)\equiv\exists x(P\wedge\neg Q)$。**Q3 (T)**. 左边 $\equiv(\neg p\vee q)\wedge(\neg p\vee r)\equiv\neg p\vee(q\wedge r)\equiv p\to(q\wedge r)=$ 右边。

Ch2

Q4 (T). 笛卡尔积基数 = 两个集合基数的乘积，这是组合数学基本结论。**Q5 (T)**. f injective: $f(x)=f(y)\Rightarrow x=y$. g injective: $g(u)=g(v)\Rightarrow u=v$. Then $(g\circ f)(x)=(g\circ f)(y)\Rightarrow g(f(x))=g(f(y))\Rightarrow f(x)=f(y)\Rightarrow x=y$。**Q6 (T)**. 所有有限子集 $=\bigcup_{n=0}^{\infty}\mathcal{P}_n(\mathbb{N})$，每个 $\mathcal{P}_n$ 与 $\mathbb{N}^n$ 的一个子集等势，可数个可数集的并仍是可数的。

Ch3

Q7 (T). $f(n)=3n^2+2n+1\le 3n^2+2n^2+n^2=6n^2$ for $n\ge 1$，且 $f(n)\ge 3n^2$，所以 $f(n)=\Theta(n^2)$。**Q8 (F)**. $n!$ 的增长比任何指数函数 c^n 都快（Stirling 公式：$n!\sim\sqrt{2\pi n}(n/e)^n$）。所以 $n!=\Omega(2^n)$ 而非 $O(2^n)$。

Ch4

Q9 (T). 同余式的基本性质：$a\equiv b\pmod m\iff a=b+km$，$c\equiv d\pmod m\iff c=d+\ell m$，相乘 $(b+km)(d+\ell m)=bd+m(b\ell+dk+k\ell m)$，所以 $ac\equiv bd\pmod m$。**Q10 (T)**. gcd 和 lcm 的基本关系，由素因子分解 $\gcd=\prod p^{\min(e,f)}$, $\mathrm{lcm}=\prod p^{\max(e,f)}$，乘积 $=\prod p^{e+f}=ab$。

Ch5

Q11 (T). 良序原理（$\mathbb{N}$ 的每个非空子集有最小元）与数学归纳法原理等价。**Q12 (T)**. 强归纳法的定义：假设 $P(1),P(2),\dots,P(k)$ 都成立来证明 $P(k+1)$，而非只假设 $P(k)$。

Ch6

Q13 (T). $P(n,r)=n!/(n-r)!$, $C(n,r)=n!/(r!(n-r)!)$，所以 $C(n,r)=P(n,r)/r!$。**Q14 (T)**. 13 个人分配到 12 个月 = 鸽巢原理的标准例子。

Ch8

Q15 (T). 特征方程 $r^2-3r+2=0$，$(r-1)(r-2)=0$，$r=1,2$。**Q16 (T)**. $D_n=n!\sum_{i=0}^{n}(-1)^i/i!\approx n!/e$。

Ch9

Q17 (T). 等价关系 → 等价类 → 划分。这是等价关系和划分的一一对应定理。**Q18 (F)**. 对称 + 传递不能推出自反。反例：$R=\emptyset$ 在 $A=\{1,2\}$ 上。R 对称且传递（vacuously true），但不自反（$(1,1)\notin R$）。

Ch10

Q19 (F). K_5 有 5 个顶点、10 条边。平面图的推论 $e\le 3v-6$：$10\le 3\cdot 5-6=9$，不成立。所以 K_5 非平面。**Q20 (T)**. Euler 定理：连通多重图有欧拉回路 $\iff$ 所有顶点度数为偶数。**Q21 (T)**. 二部图可以用 2 种颜色着色（每组一种），所以 $\chi\le 2$。如果至少有一条边，$\chi=2$；如果没有边，$\chi=1$。**Q22 (T)**. Handshaking Theorem: $\sum\deg(v)=2|E|$。奇度顶点度数和为奇数个奇数和 = 偶数，所以奇度顶点个数必为偶数。

Ch11

Q23 (T). 树的基本性质。**Q24 (T)**. 满二叉树的性质：$n_0=n_2+1$，其中 n_0 是叶节点数，n_2 是内部节点数。

Network Flow

Q25 (T). Max-Flow Min-Cut Theorem 的核心陈述。

11.7.2 Part 2: 详解

Q26 —NAND 表达 XOR

解：$p\oplus q\equiv(p\wedge\neg q)\vee(\neg p\wedge q)$（XOR 定义）利用 NAND 的性质：

- $\neg x\equiv x\mid x$
- $x\wedge y\equiv\neg(x\mid y)\equiv(x\mid y)\mid(x\mid y)$

所以：

$$\begin{aligned}p\oplus q&\equiv\neg(\neg(p\wedge\neg q)\wedge\neg(\neg p\wedge q))\\&\equiv(p\wedge\neg q)\mid(\neg p\wedge q)\quad(\text{最外层 NAND}=\neg(\text{AND}))\\&\equiv\big(p\mid(q\mid q)\big)\mid\big((p\mid p)\mid q\big)\end{aligned}$$

快捷记法：$p\oplus q\equiv(p\mid(q\mid q))\mid((p\mid p)\mid q)$

Q27 —对称差与幂集

(a) $A\oplus B=(A-B)\cup(B-A)=\{1\}\cup\{4\}=\{1,4\}$ (b) $A\cap B=\{2,3\}$，$\mathcal{P}(\{2,3\})=\{\emptyset,\{2\},\{3\},\{2,3\}\}$

Q28 —函数增长率排序

$n^{1/\log n}$：注意到 $\log_n n=1$，$n^{1/\log n}=n^{\log_n 2}=2$。实际上 $n^{1/\log_2 n}=2$ 是常数！排序（慢 → 快）：

$$n^{1/\log n}\quad(\text{常数})<(\log n)^2<n\log n<n^2<2^n<n!$$

Q29 —欧几里得算法

(a)

$$\begin{aligned}252&=1\cdot 198+54\\198&=3\cdot 54+36\\54&=1\cdot 36+18\\36&=2\cdot 18+0\end{aligned}$$

最后非零余数：$\gcd(252,198)=18$ **(b)** 回代求 Bézout 系数：

$$\begin{aligned}18&=54-1\cdot 36\\36&=198-3\cdot 54\\54&=252-1\cdot 198\end{aligned}$$

代入：

$$\begin{aligned}18&=54-1\cdot(198-3\cdot 54)=4\cdot 54-1\cdot 198\\&=4\cdot(252-1\cdot 198)-1\cdot 198\\&=4\cdot 252-5\cdot 198\end{aligned}$$

所以 $18=4\cdot 252-5\cdot 198$（$s=4,t=-5$）

Q30 —归纳法证奇数求和

Basis Step $(n=1)$：$\sum_{i=1}^{1}(2i-1)=1=1^2$ **Inductive Hypothesis**：假设 $\sum_{i=1}^{k}(2i-1)=k^2$ 成立。**Inductive Step** $(n=k+1)$：

$$\begin{aligned}\sum_{i=1}^{k+1}(2i-1)&=\sum_{i=1}^{k}(2i-1)+(2(k+1)-1)\\&=k^2+(2k+1)\quad(\text{by IH})\\&=(k+1)^2\end{aligned}$$

证毕。□

Q31 —字符串计数

(a) 容斥原理：总数 - 不含'a' 的 $=26^4-25^4=456976-390625=66351$ **(b)** $P(26,4)=26\cdot 25\cdot 24\cdot 23=358800$ **(c)** 第一位：5 种（元音），第四位：21 种（辅音），中间两位：26^2。$5\cdot 21\cdot 26^2=105\cdot 676=70980$

Q32 —解齐次线性递推

特征方程：$r^2-5r+6=0$，$(r-2)(r-3)=0$，$r=2,3$ 通解：$a_n=\alpha\cdot 2^n+\beta\cdot 3^n$ 代入初始条件：

$$\begin{aligned}n=0&:\alpha+\beta=2\\n=1&:2\alpha+3\beta=7\end{aligned}$$

解得：$\alpha=-1+2=3$? 重新算：$\beta=2-\alpha$，$2\alpha+3(2-\alpha)=7\Rightarrow 2\alpha+6-3\alpha=7\Rightarrow-\alpha=1\Rightarrow\alpha=-1$? 等一等：$2\alpha+6-3\alpha=7\Rightarrow-\alpha+6=7\Rightarrow-\alpha=1\Rightarrow\alpha=-1$，$\beta=2-(-1)=3$。校验：$n=0$: $-1+3=2$ ，$n=1$: $2(-1)+3(3)=-2+9=7$ 所以 $a_n=-2^n+3\cdot 3^n=3\cdot 3^n-2^n=3^{n+1}-2^n$

11.7.3 Part 3: 详解

Q33 —偏序关系

(a) 自反：$a\mid a$ 对任意 $a\in A$ 成立 反对称：若 $a\mid b$ 且 $b\mid a$ 则 $a=b$ 传递：若 $a\mid b$ 且 $b\mid c$ 则 $a\mid c$ 所以 R 是偏序。**(b)** Hasse 图（省略传递边和自环边）：

```
      12
     /  \
    6    4
   / \   |
  2   3  |
   \ /   |
    1 ---
```

实际上：1→2→6→12, 1→3→6, 1→4→12, 1→2→4→12

```
      12
     /  \
    6    4
   / \   /
  2   3 /
   \ / /
    1
```

(c) 极大元 $= 12$（没有元素能整除 12 或 4 或 6 之后得到更大值... wait. 极大元：没有其他元素”大于”它。12 是唯一极大元，因为 $12 \mid a \Rightarrow a = 12$）。实际上 4 和 6 呢？$4 \mid 12$ 所以 $12 > 4$，4 不是极大元。极大元 $= 12$。最大元 $= 12$。极小元：没有其他元素”小于”它。1 是唯一极小元。最小元 $= 1$。**(d)** 在 $(A, |)$ 中，任意两个元素 a, b：$\text{lub}(a, b) = \text{lcm}(a, b)$，$\text{glb}(a, b) = \gcd(a, b)$。A = 1, 2, 3, 4, 6, 12 对 lcm/gcd 封闭：任意两个元素的 lcm 和 gcd 都在 A 中吗？

- $\text{lcm}(4, 6) = 12 \in A$
- $\text{lcm}(3, 4) = 12 \in A$
- $\gcd(4, 6) = 2 \in A$

所有组合的 lcm 和 gcd 都在 A 中！所以是 lattice。事实上 $(\{\quad\}, |)$ 总是 lattice。

Q34 —图论综合

(a) Dijkstra 算法（图：A=0 起点）

```
    2      1      5      2      3
A --- C --- B --- D --- E --- F
 \         \               /
  4         8             6
   \         \           /
    +---------+---------+
```

实际上边是：A-B:4, A-C:2, B-C:1, B-D:5, C-D:8, C-E:10, D-E:2, D-F:6, E-F:3

Step	已确定	A	B	C	D	E	F
0	—	0	∞	∞	∞	∞	∞
1	A	**0**	4	2	∞	∞	∞
2	C	0	3	**2**	10	12	∞
3	B	0	**3**	2	8	12	∞
4	D	0	3	2	**8**	10	14
5	E	0	3	2	8	**10**	13
6	F	0	3	2	8	10	**13**

最短路径：A → C → B → D → E → F，长度 $= 2 + 1 + 5 + 2 + 3 =$ **13** 路径标记：A→C (2), C→B 通过 B=3 (via C-B=1), B→D=8 (via B-D=5), D→E=10 (via D-E=2), E→F=13 (via E-F=3) **(b)** 度数：deg(A)=2, deg(B)=3, deg(C)=3, deg(D)=3, deg(E)=3, deg(F)=2

- **Euler circuit?** 有 4 个奇度顶点（B,C,D,E）→ 不存在 Euler circuit。
- **Euler path?** 要求恰好 2 个奇度顶点 → 不存在。
- **Hamilton circuit?** Dirac 定理：$\deg(v) \geq 3 \geq 6/2 = 3$，刚好满足 Dirac 条件！所以存在 Hamilton circuit。一条路径：A → B → D → F → E → C → A（验证：A-B, B-D, D-F, F-E, E-C, C-A，全部是图中的边 ）

Q35 —Huffman 编码

(a) 频率：A=0.15, B=0.25, C=0.05, D=0.20, E=0.10, F=0.25 合并过程（每次选频率最小的两个）：

Step	合并	新节点
1	C(0.05) + E(0.10)	CE =
2	A(0.15) + CE(0.15)	A-CE =
3	D(0.20) + B(0.25)	D-B = 0.45... 等等，B=0.25, D=0.20, F=0.25。最小两个是 D(0.20) + F(0.25)？不对，B(0.25)=F(0.2
3	D(0.20) + F(0.25)	DF =
4	A-CE(0.30) + B(0.25)	A-CE-B
5	A-CE-B(0.55) + DF(0.45)	ROOT =

编码（左 =0, 右 =1，从根向叶走）：

```
ROOT
  0: A-CE-B (0.55)
     0: B (0.25)          → B: 00
     1: A-CE (0.30)
        0: A (0.15)       → A: 010
        1: CE (0.15)
           0: C (0.05)  → C: 0110
```



```
           1: E (0.10)  → E: 0111
  1: DF (0.45)
     0: D (0.20)          → D: 10
     1: F (0.25)          → F: 11
```

符号	编码	长度
B	00	2
D	10	2
F	11	2
A	010	3
C	0110	4
E	0111	4

(b) 平均比特数：

$$2 \cdot (0.25 + 0.20 + 0.25) + 3 \cdot 0.15 + 4 \cdot (0.05 + 0.10) = 2 \cdot 0.70 + 3 \cdot 0.15 + 4 \cdot 0.15 = 1.40 + 0.45 + 0.60 = 2.45$$

(c) 定长编码：6 个符号需要 $\lceil \log_2 6 \rceil = 3$ bits。Huffman 节省了 $(3 - 2.45)/3 \approx 18.3\%$。

第三部分

历年真题题解

11.8 DMA Quiz 1 题解—2022 年

11.8.1 一、判断题 (True/False) (12%)

题目 1a

题目：判断以下两个命题是否逻辑等价：$p \to (\neg q \wedge r)$ 与 $\neg p \vee \neg(r \to q)$ **答案**：T **考点**：第一章命题逻辑（Propositional Logic）——逻辑等价（Logical Equivalence），蕴含律（Implication Law），德摩根律（De Morgan's Law）**解析**：

$$\begin{aligned}
\text{LHS} &= p \to (\neg q \wedge r) \equiv \neg p \vee (\neg q \wedge r) && \text{(蕴含律)} \\
\text{RHS} &= \neg p \vee \neg(r \to q) \equiv \neg p \vee \neg(\neg r \vee q) && \text{(蕴含律)} \\
&\equiv \neg p \vee (r \wedge \neg q) && \text{(德摩根律)}
\end{aligned}$$

由于 $\neg q \wedge r \equiv r \wedge \neg q$（合取交换律），LHS 与 RHS 完全一致，故逻辑等价。答案为 **T**。

题目 1b

题目：若 A, B, C 是集合，则 $A - (B \cap C) = (A - B) \cup (A - C)$ **答案**：T **考点**：第二章集合（Sets）——集合运算（Set Operations），德摩根律的集合版本 **解析**：

$$\begin{aligned}
x \in A - (B \cap C) &\iff x \in A \wedge x \notin (B \cap C) \\
&\iff x \in A \wedge (x \notin B \vee x \notin C) && \text{(德摩根律)} \\
&\iff (x \in A \wedge x \notin B) \vee (x \in A \wedge x \notin C) && \text{(分配律)} \\
&\iff x \in (A - B) \cup (A - C)
\end{aligned}$$

由集合外延公理，两集合相等。答案为 **T**。

题目 1c

题目：判断以下命题是否为重言式（tautology）：$((p \to q) \wedge \neg p) \to \neg q$ **答案**：F **考点**：第一章命题逻辑——重言式（Tautology），真值表法 **解析**：考虑赋值 $p = \mathrm{F}, q = \mathrm{T}$：

$$((\mathrm{F} \to \mathrm{T}) \wedge \neg \mathrm{F}) \to \neg \mathrm{T} = (\mathrm{T} \wedge \mathrm{T}) \to \mathrm{F} = \mathrm{T} \to \mathrm{F} = \mathrm{F}$$

存在使公式为假的赋值，故不是重言式。答案为 **F**。或者化简：$((\neg p \vee q) \wedge \neg p) \to \neg q \equiv \neg((\neg p \vee q) \wedge \neg p) \vee \neg q \equiv (p \wedge \neg q) \vee p \vee \neg q$，当 $p = \mathrm{F}, q = \mathrm{T}$ 时值为假。

题目 1d

题目：$\mathrm{P}(A) = \mathrm{P}(B)$ 当且仅当 $A = B$，其中 $\mathrm{P}(X)$ 是 X 的幂集 **答案**：T **考点**：第二章集合——幂集（Power Set）**解析**：

- ($\Leftarrow$) 若 $A = B$，则显然 $\mathrm{P}(A) = \mathrm{P}(B)$。
- ($\Rightarrow$) 若 $\mathrm{P}(A) = \mathrm{P}(B)$，则 $A \in \mathrm{P}(A) = \mathrm{P}(B) \Rightarrow A \subseteq B$。同理 $B \subseteq A$。故 $A = B$。

答案为 **T**。

11.8.2 二、谓词逻辑翻译 (16%)

题目：设变量 x 表示学生，y 表示课程，$T(x, y)$ 表示"x 正在修 y"。将下列语句翻译为符号形式。

题目 2a

题目：没有课程被所有学生修读（No course is being taken by all students）**答案**：$\neg\exists y \forall x\, T(x, y)$ 或 $\forall y \exists x\, \neg T(x, y)$ **考点**：第一章谓词与量词（Predicates and Quantifiers）——量词的翻译与否定 **解析**：

- " 存在一门课程，所有学生都在修"：$\exists y \forall x\, T(x, y)$
- " 没有……" 即否定：$\neg\exists y \forall x\, T(x, y)$
- 等价形式：$\forall y \neg\forall x\, T(x, y) \equiv \forall y \exists x\, \neg T(x, y)$

题目 2b

题目：每个学生至少修一门课（Every student is taking at least one course）**答案**：$\forall x \exists y\, T(x, y)$ **考点**：谓词与量词——全称量词与存在量词的嵌套 **解析**：" 每个学生" 对应 $\forall x$，" 存在一门课" 对应 $\exists y$，" 正在修" 即 $T(x, y)$。

题目 2c

题目：有些课程没有被任何学生修读（Some courses are being taken by no students）**答案**：$\exists y \forall x\, \neg T(x, y)$ **考点**：谓词与量词——量词的翻译 **解析**：" 存在课程 y，使得对所有学生 x，x 没有在修 y"。

题目 2d

题目：没有学生修读所有课程（No student is taking all courses）**答案**：$\neg\exists x \forall y\, T(x, y)$ 或 $\forall x \exists y\, \neg T(x, y)$ **考点**：谓词与量词——量词的翻译与否定 **解析**：

- " 存在学生修读所有课程"：$\exists x \forall y\, T(x, y)$
- " 没有" 即否定：$\neg\exists x \forall y\, T(x, y)$
- 等价形式：$\forall x \exists y\, \neg T(x, y)$

11.8.3 三、命题等价变换 (12%)

题目 3a

题目：写出一个与 $p \vee \neg q$ 等价的命题，只使用 $p, q, \neg$ 和联结词 $\wedge$（4%）**答案**：$\neg(\neg p \wedge q)$ **考点**：第一章命题逻辑——联结词归约，德摩根律 **解析**：

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

应用德摩根律：$\neg(A \wedge B) \equiv \neg A \vee \neg B$，取 $A = \neg p, B = q$ 即得。

题目 3b

题目：写出一个与 $p \wedge q$ 等价的命题，只使用 p, q 和联结词 $|$（NAND，与非）（8%）**答案**：$(p \mid q) \mid (p \mid q)$ **考点**：第一章命题逻辑——功能完全联结词（Functional Completeness），NAND **解析**：由定义 $p \mid q \equiv \neg(p \wedge q)$。利用 NAND 可以构造非和合取：

- $\neg p \equiv p \mid p$（因为 $p \mid p \equiv \neg(p \wedge p) \equiv \neg p$）
- $p \wedge q \equiv \neg(p \mid q) \equiv (p \mid q) \mid (p \mid q)$

11.8.4 四、主析取范式与主合取范式 (16%)

题目：有一个包含三个变量 p, q, r 的命题公式，在最多一个变量为真时取真，否则取假。

- (a) 表达为全析取范式（full DNF）（8%）
- (b) 表达为全合取范式（full CNF）（8%）

答案：

- (a) $(\neg p \wedge \neg q \wedge \neg r) \vee (p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r)$
- (b) $(p \vee q \vee \neg r) \wedge (p \vee \neg q \vee r) \wedge (\neg p \vee q \vee r) \wedge (p \vee q \vee r)$

考点：第一章命题逻辑——主析取范式（DNF）、主合取范式（CNF）**解析**：公式为真的情况（0 或 1 个变量为真）：

p	q	r	取值	极小项
F	F	F	T	$\neg p \wedge \neg q \wedge \neg r$
T	F	F	T	$p \wedge \neg q \wedge \neg r$
F	T	F	T	$\neg p \wedge q \wedge \neg r$
F	F	T	T	$\neg p \wedge \neg q \wedge r$

(a) 全 DNF：将所有极小项析取即得：

$$(\neg p \wedge \neg q \wedge \neg r) \vee (p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r)$$

公式为假的情况（2 或 3 个变量为真）：

p	q	r	取值	极大项
T	T	F	F	$\neg p \vee \neg q \vee r$
T	F	T	F	$\neg p \vee q \vee \neg r$
F	T	T	F	$p \vee \neg q \vee \neg r$
T	T	T	F	$\neg p \vee \neg q \vee \neg r$

(b) 全 CNF：将所有极大项合取得：

$$(\neg p \vee \neg q \vee r) \wedge (\neg p \vee q \vee \neg r) \wedge (p \vee \neg q \vee \neg r) \wedge (\neg p \vee \neg q \vee \neg r)$$

11.8.5 五、集合分配律的推广 (12%)

题目：证明分配律 $A_1 \cap (A_2 \cup \cdots \cup A_n) = (A_1 \cap A_2) \cup \cdots \cup (A_1 \cap A_n)$ 对所有 $n > 2$ 成立。**考点**：第二章集合——集合运算分配律，数学归纳法 **解析**：**方法一（元素法）**：

$$\begin{aligned} x \in A_1 \cap (A_2 \cup \cdots \cup A_n) &\iff x \in A_1 \wedge x \in (A_2 \cup \cdots \cup A_n) \\ &\iff x \in A_1 \wedge (x \in A_2 \vee \cdots \vee x \in A_n) \\ &\iff (x \in A_1 \wedge x \in A_2) \vee \cdots \vee (x \in A_1 \wedge x \in A_n) \\ &\iff x \in (A_1 \cap A_2) \cup \cdots \cup (A_1 \cap A_n) \end{aligned}$$

方法二（数学归纳法）：归纳基础：$n = 2$ 时，$A_1 \cap (A_2 \cup A_3) = (A_1 \cap A_2) \cup (A_1 \cap A_3)$ 是集合的基本分配律，成立。归纳假设：设 $n = k$ $(k \geq 2)$ 时命题成立。归纳步骤（$n = k + 1$）：

$$\begin{aligned} A_1 \cap (A_2 \cup \cdots \cup A_k \cup A_{k+1}) &= A_1 \cap ((A_2 \cup \cdots \cup A_k) \cup A_{k+1}) \\ &= (A_1 \cap (A_2 \cup \cdots \cup A_k)) \cup (A_1 \cap A_{k+1}) && \text{(基本分配律)} \\ &= ((A_1 \cap A_2) \cup \cdots \cup (A_1 \cap A_k)) \cup (A_1 \cap A_{k+1}) && \text{(归纳假设)} \\ &= (A_1 \cap A_2) \cup \cdots \cup (A_1 \cap A_k) \cup (A_1 \cap A_{k+1}) \end{aligned}$$

由数学归纳法，命题对所有 $n > 2$ 成立。

11.8.6 六、Cantor 对角线法 (12%)

题目：改编 Cantor 对角线论证，证明 $(0, 1)$ 中十进制表示只由 0 和 1 组成的正实数集合是不可数的。**考点**：第二章集合的基数（Cardinality of Sets）——Cantor 对角线论证，不可数集 **解析**：设 $S = \{r \in (0, 1) \mid r$ 的十进制表示中只包含数码 0 和 1$\}$。假设 S 是可数的，则可以将 S 中的所有数排成一个序列 $r_1, r_2, r_3, \ldots$。将每个 r_i 写成十进制小数形式：

$$r_1 = 0.a_{11}a_{12}a_{13}\cdots r_2 = 0.a_{21}a_{22}a_{23}\cdots r_3 = 0.a_{31}a_{32}a_{33}\cdots \vdots$$

其中每个 $a_{ij} \in \{0, 1\}$。构造一个新数 $r = 0.b_1b_2b_3\cdots$，其中

$$b_i = \begin{cases} 1 & 若 a_{ii} = 0 \\ 0 & 若 a_{ii} = 1 \end{cases}$$

即 $b_i = 1 - a_{ii}$（在二进制意义下的" 翻转"）。那么 $r \in S$（因为每位都是 0 或 1），但 r 与每个 r_i 都在第 i 位上不同，所以 $r \notin \{r_1, r_2, r_3, \ldots\}$，与假设矛盾。因此 S 是不可数的。

注意：这里需要处理 0.01111... = 0.1000... 的表示歧义问题。但因为我们限制只使用 0 和 1，且构造的 r 也不是有限小数（无限不循环），所以表示是唯一的。

11.8.7 七、算法复杂度分析 (8%)

题目 7a

题目：求打印集合 $\{1, 2, \ldots, n\}$ 的所有三元子集的算法的复杂度（用最佳 Big-O 表示）（4%）**答案**：$O(n^3)$ **考点**：第三章算法复杂度（Complexity of Algorithms）——Big-O 记号 **解析**：三元子集的个数为：

$$\binom{n}{3} = \frac{n(n-1)(n-2)}{6} = \frac{n^3 - 3n^2 + 2n}{6} = O(n^3)$$

打印每个子集的时间是常数，因此总复杂度为 $O(n^3)$。

题目 7b

题目：线性查找在最佳情况下的比较次数（对规模为 n 的列表）（4%）**答案**：$O(1)$ **考点**：第三章算法复杂度——Big-O 记号，最佳情况分析 **解析**：线性查找的最佳情况是目标元素恰好是列表的第一个元素，只需要 1 次比较即结束。因此复杂度为 $O(1)$（常数时间）。

11.8.8 八、数学归纳法 (12%)

题目：用数学归纳法证明 $n^{n+1} > (n+1)^n$ 对所有正整数 $n \geq 3$ 成立。（注：试卷原题写为 $n \leq 3$，但该不等式实际对 $n = 1, 2$ 不成立，应为 $n \geq 3$ 的笔误。）**考点**：第五章数学归纳法（Mathematical Induction）——不等式归纳证明 **解析**：**归纳基础**：$n = 3$：

$$3^4 = 81 > 4^3 = 64$$

成立。**归纳假设**：设 $n = k$（$k \geq 3$）时成立，即 $k^{k+1} > (k+1)^k$。**归纳步骤**：需证 $(k+1)^{k+2} > (k+2)^{k+1}$。由归纳假设：

$$k^{k+1} > (k+1)^k \iff k > \left(\frac{k+1}{k}\right)^k = \left(1 + \frac{1}{k}\right)^k$$

已知数列 $\left(1 + \frac{1}{n}\right)^n$ 单调递增且收敛到 $e < 3$。因此对 $k \geq 3$：

$$\left(1 + \frac{1}{k}\right)^k < e < 3 \leq k$$

于是：

$$(k+1)^{k+2} = (k+1) \cdot (k+1)^{k+1} = (k+1) \cdot (k+1)^k \cdot (k+1)$$

也可以间接证明：考虑比率：

$$\frac{(k+1)^{k+2}}{(k+2)^{k+1}} = (k+1) \cdot \left(\frac{k+1}{k+2}\right)^{k+1} = (k+1) \cdot \left(1 - \frac{1}{k+2}\right)^{k+1}$$

由归纳假设和伯努利不等式或已知的单调性可证该比率 > 1。具体地：因为序列 $\left(1 + \frac{1}{n}\right)^n$ 单调递增趋于 e，所以：

$$\left(1 + \frac{1}{k+1}\right)^{k+1} > \left(1 + \frac{1}{k}\right)^k$$

但我们需要比较的是 $(1 - \frac{1}{k+2})^{k+1}$。利用不等式：

$$\left(1 - \frac{1}{m}\right)^{m-1} > \frac{1}{e} \text{ 对} m > 1$$

代入 $m = k + 2$：$\left(1 - \frac{1}{k+2}\right)^{k+1} > \frac{1}{e}$ 所以：

$$(k+1) \cdot \left(1 - \frac{1}{k+2}\right)^{k+1} > (k+1) \cdot \frac{1}{e} > \frac{k+1}{3} \geq \frac{4}{3} > 1$$

故 $(k+1)^{k+2} > (k+2)^{k+1}$ 成立。由数学归纳法，$n^{n+1} > (n+1)^n$ 对所有 $n \geq 3$ 成立。

11.9 DMA Quiz 1 题解—2023 年

11.9.1 一、判断题 (True/False) (30%)

评分规则：答对 +5 分，空白 0 分，答错 -2 分（倒扣机制，防止乱猜）

题目 1a

题目：判断以下两个命题是否逻辑等价（所有量词有相同的非空论域）：

$$\forall x P(x) \vee \exists x Q(x), \quad \forall x \exists y\,(P(x) \vee Q(y))$$

答案：T **考点**：第一章嵌套量词（Nested Quantifiers）——量词的逻辑等价 **解析**：从左到右：若 $\forall x P(x) \vee \exists x Q(x)$ 为真，则有两种情况：

- 若 $\forall x P(x)$ 为真，则对任意 x，$P(x)$ 为真，从而 $P(x) \vee Q(y)$ 对任意 y 为真，故 $\forall x \exists y\,(P(x) \vee Q(y))$ 为真。
- 若 $\exists x Q(x)$ 为真，设 c 使 $Q(c)$ 为真，则对任意 x，取 $y = c$ 有 $P(x) \vee Q(c)$ 为真，故原式成立。

从右到左：若 $\forall x \exists y\,(P(x) \vee Q(y))$ 为真而 $\forall x P(x) \vee \exists x Q(x)$ 为假，则 $\forall x P(x)$ 假且 $\exists x Q(x)$ 假。即存在 x_0 使 $P(x_0)$ 为假，且对所有 x，$Q(x)$ 为假。那么对 $x = x_0$，$\forall y\,(P(x_0) \vee Q(y))$ 为假（取任意 y，$P(x_0) \vee Q(y) \equiv \mathrm{F} \vee \mathrm{F} \equiv \mathrm{F}$），与假设矛盾。因此两个命题逻辑等价。答案为 **T**。

题目 1b

题目：若 A, B, C 是集合，则 $(A - C) - (B - C) = A - B$ **答案**：F **考点**：第二章集合运算 **解析**：取反例即可。设 $A = \{1, 2, 3\}, B = \{1\}, C = \{1, 2\}$：

$$A - C = \{3\}, \quad B - C = \varnothing, \quad (A - C) - (B - C) = \{3\} A - B = \{2, 3\}$$

$\{3\} \neq \{2, 3\}$，等式不成立。更精确地，$(A - C) - (B - C) = A - (B \cup C)$（可化简验证），而非 $A - B$。答案为 **F**。

题目 1c

题目：满足整系数三次方程 $ax^3 + bx^2 + cx + d = 0$ 的实数解集合是不可数的。**答案**：F **考点**：第二章集合的基数——可数集的性质 **解析**：

- 每个三次方程由整数系数 $(a, b, c, d) \in \mathbb{Z}^4$ 确定，$\mathbb{Z}^4$ 是可数集（可数集的笛卡尔积仍是可数的）。
- 每个三次方程最多有 3 个实根。
- 因此所有实根组成的集合是" 可数多个有限集" 的并，仍然是可数的。

故该集合是可数的，不是不可数的。答案为 **F**。

题目 1d

题目：若 $\mathrm{P}(A) \in \mathrm{P}(B)$，则 $A \in B$ **答案**：T **考点**：第二章集合——幂集的概念 **解析**：$\mathrm{P}(A) \in \mathrm{P}(B)$ 意味着 $\mathrm{P}(A)$ 是 B 的一个子集，即 $\mathrm{P}(A) \subseteq B$。由于 $A \in \mathrm{P}(A)$（任何集合都是其自身的子集），可得 $A \in B$。答案为 **T**。

题目 1e

题目：公式 $\neg(p \to q) \wedge q$ 是重言式（tautology）。**答案**：F **考点**：第一章命题逻辑——重言式 **解析**：

$$\neg(p \to q) \wedge q \equiv \neg(\neg p \vee q) \wedge q \equiv (p \wedge \neg q) \wedge q \equiv p \wedge (\neg q \wedge q) \equiv p \wedge \mathrm{F} \equiv \mathrm{F}$$

这是一个矛盾式（contradiction），而非重言式。答案为 **F**。

题目 1f

题目：存在一个从 $\mathbb{R}$ 到 $\mathbb{Z} \times \mathbb{Z}$ 的单射（one-to-one function）。**答案**：F **考点**：第二章集合的基数——单射与基数比较 **解析**：

- $|\mathbb{Z} \times \mathbb{Z}| = |\mathbb{Z}| = \aleph_0$（可数无穷）
- $|\mathbb{R}| = \mathfrak{c}$（不可数连续统势）
- 若存在单射 $f : \mathbb{R} \to \mathbb{Z} \times \mathbb{Z}$，则 $|\mathbb{R}| \leq |\mathbb{Z} \times \mathbb{Z}|$，即 $\mathfrak{c} \leq \aleph_0$，矛盾。

故不存在这样的单射。答案为 **F**。

11.9.2 二、单项选择题 (5%)

题目：下列哪个命题等价关系成立？A. $\forall x(P(x) \wedge Q(x)) \equiv \forall x P(x) \wedge \forall x Q(x)$ B. $\exists x(P(x) \wedge Q(x)) \equiv \exists x P(x) \wedge \exists x Q(x)$ C. $\forall x(P(x) \to Q(x)) \equiv \forall x P(x) \to \forall x Q(x)$ D. $\exists x(P(x) \to Q(x)) \equiv \exists x P(x) \to \exists x Q(x)$ **答案**：A **考点**：第一章谓词与量词——量词的分配律 **解析**：

- **A** 正确：$\forall$ 对 $\wedge$ 可分配，两边都表示" 所有 x 同时满足 P 和 Q"。
- **B** 错误：$\exists$ 对 $\wedge$ 不可分配。右式可能由不同 x 分别满足 P 和 Q 而得真，但左式要求同一个 x 同时满足两者。反例：论域 $\{a, b\}$，$P(a) = \mathrm{T}, P(b) = \mathrm{F}, Q(a) = \mathrm{F}, Q(b) = \mathrm{T}$。此时 $\exists x(P(x) \wedge Q(x)) = \mathrm{F}$，但 $\exists x P(x) \wedge \exists x Q(x) = \mathrm{T} \wedge \mathrm{T} = \mathrm{T}$。
- **C** 错误：反例略。一般而言 $\forall$ 不分配于 $\to$。
- **D** 错误：反例：论域 $\{a, b\}$，$P(a) = \mathrm{T}, P(b) = \mathrm{F}, Q(a) = \mathrm{F}, Q(b) = \mathrm{F}$。$\exists x(P(x) \to Q(x)) = (P(a) \to Q(a)) \vee (P(b) \to Q(b)) = \mathrm{F} \vee \mathrm{T} = \mathrm{T}$，但 $\exists x P(x) \to \exists x Q(x) = \mathrm{T} \to \mathrm{F} = \mathrm{F}$。

故选 **A**。

11.9.3 三、命题等价变换 (10%)

题目 3a

题目：写出一个与 $p \wedge q$ 等价的命题，只使用 $p, q, \neg$ 和联结词 $\vee$（5%）**答案**：$\neg(\neg p \vee \neg q)$ **考点**：第一章命题逻辑——德摩根律 **解析**：

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

由德摩根律，合取可用" 非" 和" 析取" 表示。

题目 3b

题目：写出一个与 $p \wedge q$ 等价的命题，只使用 p, q 和联结词 $\downarrow$（NOR，或非）（5%）**答案**：$(p \downarrow p) \downarrow (q \downarrow q)$ **考点**：第一章命题逻辑——功能完全联结词，NOR **解析**：$p \downarrow q \equiv \neg(p \vee q)$，即只有当 p, q 都为假时 $p \downarrow q$ 才为真。利用 NOR 构造否定：$p \downarrow p \equiv \neg(p \vee p) \equiv \neg p$ 利用 NOR 构造合取：

$$p \wedge q \equiv \neg(\neg p \vee \neg q) \equiv \neg p \downarrow \neg q \equiv (p \downarrow p) \downarrow (q \downarrow q)$$

11.9.4 四、函数阶的排序 (8%)

题目：将以下函数按顺序排列，使得每个函数是下一个函数的 Big-O（即从小到大排列）。

$$\begin{aligned}
&f_1(n) = (1.2)^n && f_2(n) = 7n^6 + n + 323 && f_3(n) = (\log n)^3 \\
&f_4(n) = 3^n && f_5(n) = \log(\log n) && f_6(n) = n^2(\log n)^3 \\
&f_7(n) = 3^n(n^3 + 1) && f_8(n) = n^3 + n(\log n)^2 && f_9(n) = 1000000 \\
&f_{10}(n) = 10n!
\end{aligned}$$

答案：$f_9 \to f_5 \to f_3 \to f_6 \to f_8 \to f_2 \to f_1 \to f_4 \to f_7 \to f_{10}$ **考点**：第三章函数的增长（Growth of Functions）——常见函数增长阶比较 **解析**：按增长速度从慢到快排列：

函数	阶	排序
$f_9(n) = 1000000$	$O(1)$ 常数	1
$f_5(n) = \log\log n$	对数-对数	2
$f_3(n) = (\log n)^3$	多对数（polylog）	3
$f_6(n) = n^2(\log n)^3$	多项式 × 对数	4
$f_8(n) = n^3 + n(\log n)^2$	$\Theta(n^3)$ 多项式	5
$f_2(n) = 7n^6 + n + 323$	$\Theta(n^6)$ 多项式	6
$f_1(n) = (1.2)^n$	指数（底数 1.2）	7
$f_4(n) = 3^n$	指数（底数 3）	8
$f_7(n) = 3^n(n^3 + 1)$	指数 × 多项式	9
$f_{10}(n) = 10n!$	阶乘	10

说明：

- 多项式增长：$\log\log n \prec (\log n)^3 \prec n^2(\log n)^3 \prec n^3 \prec n^6$
- 指数增长：$(1.2)^n \prec 3^n$ 因为 $(1.2/3)^n \to 0$
- $3^n \prec 3^n n^3$ 因为 $n^3 \to \infty$
- 阶乘增长快于指数：$n! \succ c^n$ 对所有常数 c

11.9.5 五、主合取范式与主析取范式 (12%)

题目：求 $(p \leftrightarrow \neg r) \to (q \leftrightarrow r)$ 的：

- (a) 全合取范式（full CNF）（6%）
- (b) 全析取范式（full DNF）（6%）

考点：第一章命题逻辑——主范式 **解析**：先列出真值表：

p	q	r	$p \leftrightarrow \neg r$	$q \leftrightarrow r$	$(p \leftrightarrow \neg r) \to (q \leftrightarrow r)$
F	F	F	T	T	T
F	F	T	T	F	F
F	T	F	T	F	F
F	T	T	T	T	T
T	F	F	F	T	T
T	F	T	F	F	T
T	T	F	F	F	T
T	T	T	F	T	T

公式为真的行：$(F, F, F), (F, T, T), (T, F, F), (T, F, T), (T, T, F), (T, T, T)$ —共 6 行公式为假的行：$(F, F, T), (F, T, F)$ —共 2 行 **(a) 全 DNF**（取所有为真的极小项的析取）：

$$(\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge r) \vee (p \wedge \neg q \wedge \neg r) \vee (p \wedge \neg q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (p \wedge q \wedge r)$$

(b) 全 CNF（取所有为假的极大项的合取）：

$$(p \vee q \vee \neg r) \wedge (p \vee \neg q \vee r)$$

11.9.6 六、有理数子集的证明 (12%)

题目：设 S 是 $\mathbb{Q}$ 的子集，满足：

1. 若 $a \in S, b \in S$，则 $a + b \in S, ab \in S$

2. 对任意有理数 r，以下三种关系恰有一种成立：$r \in S$，$-r \in S$，$r = 0$

证明 $S = \mathbb{Q}^+$（正有理数集）。**考点**：第二章集合、第四章数论——有理数的封闭性与序 **解析**：**第一步：证明** $1 \in S$。对 $r = 1$，由条件 (2)：要么 $1 \in S$，要么 $-1 \in S$，要么 $1 = 0$（不可能）。

- 若 $1 \in S$，成立。
- 若 $-1 \in S$，则 $(-1) \times (-1) = 1 \in S$（条件 (1) 的乘法封闭性）。

故 $1 \in S$。**第二步：证明所有正整数都在 S 中**。由条件 (1) 加法封闭性：$1 \in S \Rightarrow 1 + 1 = 2 \in S \Rightarrow 2 + 1 = 3 \in S \Rightarrow \cdots$。由归纳法，$\mathbb{N}^+ \subseteq S$。**第三步：证明** $\mathbb{Q}^+ \subseteq S$。设正有理数 $r = \frac{p}{q} > 0$，其中 $p, q \in \mathbb{N}^+$。

- 由第二步，$p, q \in S$。
- 对 r 应用条件 (2)：要么 $r \in S$，要么 $-r \in S$。
- 若 $-r \in S$，则 $(-r) \times q = -p \in S$（条件 (1) 乘法封闭性）。但 $p \in S$，由条件 (2) 知 $-p \notin S$，矛盾。

故只能是 $r \in S$，所以 $\mathbb{Q}^+ \subseteq S$。**第四步：证明** $S \subseteq \mathbb{Q}^+$。任取 $s \in S$。由条件 (2) 知 $s \neq 0$。假设 $s < 0$，则 $-s > 0$。由第三步 $\mathbb{Q}^+ \subseteq S$ 得 $-s \in S$。但条件 (2) 要求 $s \in S$ 和 $-s \in S$ 不能同时成立，矛盾。故 $s > 0$，即 $s \in \mathbb{Q}^+$。因此 $S \subseteq \mathbb{Q}^+$。综上 $S = \mathbb{Q}^+$，证毕。

11.9.7 七、斐波那契数列的模与恒等式 (15%)

题目：用归纳法证明对所有非负整数 n：

- (a) $f_{5n} \equiv 0 \pmod 5$（6%）
- (b) $f_n^2 + f_{n+1}^2 = f_{2n+1}$（9%）

其中 f_i 为第 i 个斐波那契数（$f_0 = 0, f_1 = 1, f_{n+2} = f_{n+1} + f_n$）。**考点**：第五章数学归纳法——斐波那契数列的性质

题目 7a 解析

归纳基础：$n = 0$，$f_0 = 0 \equiv 0 \pmod 5$。成立。**归纳假设**：设 $n = k$ 时成立，即 $f_{5k} \equiv 0 \pmod 5$。**归纳步骤**：需证 $f_{5(k+1)} = f_{5k+5} \equiv 0 \pmod 5$。利用递推关系展开：

$$\begin{aligned}
f_{5k+5} &= f_{5k+4} + f_{5k+3} \\
&= (f_{5k+3} + f_{5k+2}) + (f_{5k+2} + f_{5k+1}) \\
&= f_{5k+3} + 2f_{5k+2} + f_{5k+1} \\
&= (f_{5k+2} + f_{5k+1}) + 2f_{5k+2} + f_{5k+1} \\
&= 3f_{5k+2} + 2f_{5k+1} \\
&= 3(f_{5k+1} + f_{5k}) + 2f_{5k+1} \\
&= 5f_{5k+1} + 3f_{5k}
\end{aligned}$$

由归纳假设 $f_{5k} \equiv 0 \pmod 5$，所以：

$$f_{5k+5} \equiv 5f_{5k+1} + 3 \cdot 0 \equiv 0 \pmod 5$$

由数学归纳法，对所有 $n \geq 0$，$f_{5n} \equiv 0 \pmod 5$ 成立。

题目 7b 解析

归纳基础：

- $n = 0$：$f_0^2 + f_1^2 = 0^2 + 1^2 = 1 = f_1 = f_{2 \cdot 0 + 1}$。成立。
- $n = 1$：$f_1^2 + f_2^2 = 1^2 + 1^2 = 2 = f_3 = f_{2 \cdot 1 + 1}$。成立。

归纳假设（强归纳法）：设当 $n = k - 1$ 和 $n = k$（$k \geq 1$）时成立：

$$f_{k-1}^2 + f_k^2 = f_{2k-1}, \quad f_k^2 + f_{k+1}^2 = f_{2k+1}$$

归纳步骤：需证 $n = k + 1$ 时成立：$f_{k+1}^2 + f_{k+2}^2 = f_{2k+3}$。关键引理（斐波那契加法公式）：$f_{m+n} = f_m f_{n+1} + f_{m-1} f_n$ 用该引理，取 $m = k + 1, n = k + 1$：

$$f_{2k+3} = f_{(k+1)+(k+2)} = f_{k+1} f_{k+3} + f_k f_{k+2}$$

计算右边：

$$\begin{aligned}
f_{k+1} f_{k+3} + f_k f_{k+2} &= f_{k+1}(f_{k+2} + f_{k+1}) + f_k(f_{k+1} + f_k) \\
&= f_{k+1} f_{k+2} + f_{k+1}^2 + f_k f_{k+1} + f_k^2 \\
&= f_{k+1}(f_{k+2} + f_k) + (f_{k+1}^2 + f_k^2) \\
&= f_{k+1} \cdot 2f_{k+1} + f_{2k+1} \quad (\text{因为} f_{k+2} + f_k = f_{k+1} + f_k + f_k\text{? 稍作修正})
\end{aligned}$$

另一种更简洁的方式：使用斐波那契的加法公式 $f_{m+n} = f_m f_{n+1} + f_{m-1} f_n$。取 $m = n + 1$：

$$f_{2n+1} = f_{n+1} f_{n+2} + f_n f_{n+1} = f_{n+1}(f_n + f_{n+1} + f_n)$$

等等，让我直接计算：由引理 $f_{m+n} = f_m f_{n+1} + f_{m-1} f_n$：令 $m = n, n = n + 1$：

$$f_{2n+1} = f_n f_{n+2} + f_{n-1} f_{n+1}$$

用 $f_{n-1} = f_{n+1} - f_n$ 和 $f_{n+2} = f_{n+1} + f_n$：

$$\begin{aligned}
f_{2n+1} &= f_n(f_{n+1} + f_n) + (f_{n+1} - f_n) f_{n+1} \\
&= f_n f_{n+1} + f_n^2 + f_{n+1}^2 - f_n f_{n+1} \\
&= f_n^2 + f_{n+1}^2
\end{aligned}$$

因此恒等式成立，不需归纳（直接用加法公式即可证明）。但如果一定要用归纳法，可以基于加法公式进行归纳证明，或使用矩阵幂的归纳法。**完整归纳法证明**（使用强归纳）：需要先证明引理 $f_{m+n} = f_m f_{n+1} + f_{m-1} f_n$（对 m 归纳）。引理证明略。有了引理后，直接取 $m = n + 1$ 即得恒等式。

11.9.8 八、取石子游戏 (8%)

题目：有两堆石子，分别有 115 颗和 125 颗。两人轮流从其中一堆中取走 1-3 颗石子。取走最后一颗石子的人获胜。证明先手有必胜策略。**考点**：第五章数学归纳法或博弈论——取石子游戏，Grundy 数 / Nim 和 **解析**：这是一个 impartial combinatorial game。从一堆 n 颗石子中每次可取 1-3 颗的游戏的 Grundy 数为 $n \bmod 4$。**理由**：

- 0 颗石子：无法取，先手输，Grundy 数 $= 0$
- 1 颗石子：可取到 0，Grundy 数 $= \text{mex}\{0\} = 1$
- 2 颗石子：可取到 0 或 1，Grundy 数 $= \text{mex}\{0, 1\} = 2$
- 3 颗石子：可取到 0、1 或 2，Grundy 数 $= \text{mex}\{0, 1, 2\} = 3$
- 4 颗石子：可取到 1、2 或 3（状态 1、2、3），Grundy 数 $= \text{mex}\{1, 2, 3\} = 0$
- 以此类推，周期为 4。

对于两堆石子，整个游戏的 Nim 和为两堆 Grundy 数的异或（XOR）：

$$\text{Grundy}(115) = 115 \bmod 4 = 3$$
$$\text{Grundy}(125) = 125 \bmod 4 = 1$$
$$\text{Nim-sum} = 3 \oplus 1 = 2 \neq 0$$

Nim-sum 不为 0，说明先手处于必胜位置。**必胜策略**：先手将 Nim-sum 变为 0，即让两堆的 Grundy 数相等。

- 从 115 颗的堆中取 2 颗，使其变为 113 颗（Grundy 数 $= 113 \bmod 4 = 1$），则堆状态变为 $(113, 125)$，Nim-sum $= 1 \oplus 1 = 0$。
- 或者从 125 颗的堆中取 2 颗，使其变为 123 颗（Grundy 数 $= 123 \bmod 4 = 3$），则 Nim-sum $= 3 \oplus 3 = 0$。

此后无论后手如何操作（从任一堆取走 1-3 颗），都会破坏 Nim-sum $= 0$ 的状态，先手总可以再次恢复 Nim-sum $= 0$，最终取走最后一颗石子获胜。**不依赖 Grundy 数的直接论证**：若两堆石子数模 4 同余，则后手有对称策略（模仿先手在另一堆做相同操作）。115 和 125 模 4 不同余（3 和 1），故先手可以取走 2 颗石子使两堆模 4 同余（113 和 125 模 4 都余 1，或 115 和 123 模 4 都余 3），此后模仿后手的操作即可获胜。

11.10 DMA Quiz 1 题解—2024 年

11.10.1 一、判断题 (True/False) (30%)

题目 1a

题目：判断以下两个命题是否逻辑等价：

$$p \to (q \to r), \quad (p \to q) \to r$$

答案：F **考点**：第一章命题逻辑——逻辑等价，蕴含律 **解析**：

$$p \to (q \to r) \equiv \neg p \vee (\neg q \vee r) \equiv \neg p \vee \neg q \vee r$$
$$(p \to q) \to r \equiv \neg(\neg p \vee q) \vee r \equiv (p \wedge \neg q) \vee r$$

取反例：$p = \text{F}, q = \text{T}, r = \text{F}$：

- LHS：$\text{F} \to (\text{T} \to \text{F}) \equiv \text{F} \to \text{F} \equiv \text{T}$
- RHS：$(\text{F} \to \text{T}) \to \text{F} \equiv \text{T} \to \text{F} \equiv \text{F}$

两者不等价。答案为 **F**。

题目 1b

题目：若 A, B, C 是集合，则 $A \oplus (B \oplus C) = (A \oplus B) \oplus C$（其中 $\oplus$ 为对称差）**答案**：T **考点**：第二章集合运算——对称差（Symmetric Difference）**解析**：对称差定义为 $A \oplus B = (A - B) \cup (B - A) = (A \cup B) - (A \cap B)$。对称差运算满足结合律，即 $A \oplus (B \oplus C) = (A \oplus B) \oplus C$。一种理解方式：$x \in A \oplus B$ 当且仅当 x 恰好在 A 和 B 中的一个中。因此 $x \in A \oplus (B \oplus C)$ 当且仅当 x 属于 A, B, C 中的奇数个（1 个或 3 个），这与 $(A \oplus B) \oplus C$ 等价。答案为 **T**。

题目 1c

题目：$8 + 3 = 9$ 当且仅当 $8 - 3 = 7$。**答案**：T **考点**：第一章命题逻辑——双条件（biconditional）的真值 **解析**：

- $8 + 3 = 9$ 为假
- $8 - 3 = 7$ 为假

- 双条件 $\mathrm{F} \leftrightarrow \mathrm{F} \equiv \mathrm{T}$

因此整个命题为真。答案为 **T**。

题目 1d

题目：$(0,1)$ 中十进制表示只由 6 和 8 组成的正实数集合是不可数的。**答案**：T **考点**：第二章集合的基数——Cantor 对角线论证 **解析**：此题与 2022 年第六题本质相同，只是将数码从 $\{0,1\}$ 换成了 $\{6,8\}$。通过 Cantor 对角线论证：假设该集合可数，列出所有元素 $r_1, r_2, r_3, \ldots$，构造新数 r 使得

$$r \text{ 的第} i \text{ 位} = \begin{cases} 8 & \text{若} r_i \text{ 的第} i \text{ 位} = 6 \\ 6 & \text{若} r_i \text{ 的第} i \text{ 位} = 8 \end{cases}$$

则 r 属于该集合但不等于任何 r_i，矛盾。故不可数。答案为 **T**。

题目 1e

题目：满足整系数二次方程 $ax^2 + bx + c = 0$ 的实数解集合是可数的。**答案**：T **考点**：第二章集合的基数——可数集的性质 **解析**：

- 每个二次方程由整数系数 $(a,b,c) \in \mathbb{Z}^3$ 确定，$\mathbb{Z}^3$ 是可数的。
- 每个二次方程最多有 2 个实根。
- 所有实根的集合是" 可数多个有限集" 的并，故可数。

答案为 **T**。

题目 1f

题目：在 n 个数的列表中线性查找最小数的时间复杂度是 $\Theta(n \log n)$。**答案**：F **考点**：第三章算法复杂度——线性查找 **解析**：查找最小数需要逐个比较每个元素，总共 $n-1$ 次比较，时间复杂度为 $\Theta(n)$，而非 $\Theta(n \log n)$。答案为 **F**。

11.10.2 二、谓词逻辑翻译 (10%)

题目：设变量 x 表示学生，y 表示课程，$T(x,y)$ 表示"x 正在修 y"。翻译为符号形式。

题目 2a

题目：有一门课程被所有学生修读（There is a course that is being taken by all students）**答案**：$\exists y \forall x\, T(x,y)$ **考点**：第一章谓词与量词——嵌套量词 **解析**：" 存在一门课程 y，使得对所有学生 x，x 在修 y。"

题目 2b

题目：没有学生修读所有课程（No student is taking all courses）**答案**：$\neg\exists x \forall y\, T(x,y)$ 或 $\forall x \exists y\, \neg T(x,y)$ **考点**：第一章谓词与量词——量词的否定 **解析**：

- " 存在一个学生修读所有课程"：$\exists x \forall y\, T(x,y)$
- " 没有" 即否定：$\neg\exists x \forall y\, T(x,y)$
- 等价形式（将否定内移）：$\forall x \neg\forall y\, T(x,y) \equiv \forall x \exists y\, \neg T(x,y)$

11.10.3 三、函数复合 (5%)

题目：设 $g : A \to B$ 和 $f : B \to C$，其中 $A = \{1,2,3,4\}, B = \{a,b,c\}, C = \{2,7,10\}$，且

$$g = \{(1,b),(2,a),(3,a),(4,b)\}, \quad f = \{(a,10),(b,7),(c,2)\}$$

求 $f \circ g$。**考点**：第二章函数（Functions）——函数复合 **解析**：

$$\begin{aligned}
(f \circ g)(1) &= f(g(1)) = f(b) = 7 \\
(f \circ g)(2) &= f(g(2)) = f(a) = 10 \\
(f \circ g)(3) &= f(g(3)) = f(a) = 10 \\
(f \circ g)(4) &= f(g(4)) = f(b) = 7
\end{aligned}$$

因此：

$$f \circ g = \{(1,7),(2,10),(3,10),(4,7)\}$$

11.10.4 四、命题等价变换 (7%)

题目：写出一个与 $p \wedge \neg q$ 等价的命题，只使用 p, q 和联结词 $\mid$（NAND）。**答案**：$(p \mid (q \mid q)) \mid (p \mid (q \mid q))$ **考点**：第一章命题逻辑——NAND 功能完全性 **解析**：由 NAND 定义：$p \mid q \equiv \neg(p \wedge q)$ 逐步构造：

- 否定：$\neg q \equiv q \mid q$
- 合取逆用：$\neg(p \wedge \neg q) \equiv p \mid \neg q \equiv p \mid (q \mid q)$
- 再否定回来：$p \wedge \neg q \equiv \neg(p \mid (q \mid q)) \equiv (p \mid (q \mid q)) \mid (p \mid (q \mid q))$

11.10.5 五、主析取范式与主合取范式 (14%)

题目：求 $p \oplus (q \oplus r)$（其中 $\oplus$ 表示异或 XOR）的：

- (a) 全析取范式（full DNF）（7%）
- (b) 全合取范式（full CNF）（7%）

考点：第一章命题逻辑——异或，主范式 **解析**：异或运算满足结合律：$p \oplus (q \oplus r) \equiv (p \oplus q) \oplus r \equiv p \oplus q \oplus r$。$p \oplus q \oplus r$ 为真当且仅当 p, q, r 中有奇数个为真（即 1 个或 3 个为真）。真值表：

p	q	r	$p \oplus q \oplus r$
F	F	F	F
F	F	T	T
F	T	F	T
F	T	T	F
T	F	F	T
T	F	T	F
T	T	F	F
T	T	T	T

公式为真的行（4 行）：$(F,F,T),(F,T,F),(T,F,F),(T,T,T)$ 公式为假的行（4 行）：$(F,F,F),(F,T,T),(T,F,T),(T,T,F)$ **(a) 全 DNF**：

$$(\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge \neg r) \vee (p \wedge q \wedge r)$$

(b) 全 CNF：

$$(p \vee q \vee r) \wedge (p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r)$$

11.10.6 六、函数阶的排序 (7%)

题目：将以下函数按顺序排列，使得每个函数是下一个函数的 Big-O。

$$\begin{aligned}
&f_1(n) = (1.01)^n && f_2(n) = 10n! && f_3(n) = (\log n)^3 \\
&f_4(n) = 2^n && f_5(n) = \log\log n && f_6(n) = 999n^2(\log n)^3 \\
&f_7(n) = \frac{n^4+1}{n^3+3} && f_8(n) = n^3 + n(\log n)^2 && f_9(n) = 9^{999}
\end{aligned}$$

答案：$f_9 \to f_5 \to f_3 \to f_7 \to f_6 \to f_8 \to f_1 \to f_4 \to f_2$ **考点**：第三章函数的增长——常见函数增长阶比较 **解析**：

函数	简化阶	排序
$f_9(n) = 9^{999}$	$O(1)$ 常数	1
$f_5(n) = \log\log n$	对数-对数	2
$f_3(n) = (\log n)^3$	多对数	3
$f_7(n) = \frac{n^4+1}{n^3+3} \sim n$	$\Theta(n)$	4
$f_6(n) = 999n^2(\log n)^3$	$\Theta(n^2(\log n)^3)$	5
$f_8(n) = n^3 + n(\log n)^2$	$\Theta(n^3)$	6
$f_1(n) = (1.01)^n$	指数（底 1.01）	7
$f_4(n) = 2^n$	指数（底 2）	8
$f_2(n) = 10n!$	阶乘	9

说明：

- $f_7(n) = \frac{n^4+1}{n^3+3} = n - \frac{3n}{n^3+3} + \frac{1}{n^3+3} \sim n$，为线性增长
- $\log\log n \prec (\log n)^3 \prec n \prec n^2(\log n)^3 \prec n^3$
- $(1.01)^n \prec 2^n$ 因为 $(1.01/2)^n \to 0$
- $n!$ 比任何指数函数增长都快

11.10.7 七、取整函数与集合基数 (10%)

题目：设集合

$$A = \{\lceil x \rceil + \lceil 2x \rceil + \lceil 3x \rceil \mid x \in \mathbb{R}\}, \quad B = \{x \mid x \text{ 是小于 2024 的正整数}\}$$

求 $|A \cap B|$。**考点**：第二章集合与函数——取整函数（Ceiling Function），集合的计数 **解析**：将 x 写成 $x = n + \varepsilon$，其中 $n \in \mathbb{Z}$，$\varepsilon \in [0,1)$。当 $\varepsilon = 0$（x 为整数）时：

$$\lceil x \rceil = n,\ \lceil 2x \rceil = 2n,\ \lceil 3x \rceil = 3n \implies f(x) = 6n$$

当 $\varepsilon > 0$ 时，$\lceil x \rceil = n + 1$。按 ε 范围分段讨论：

ε 范围	$\lceil 2x \rceil$	$\lceil 3x \rceil$	$f(x)$
$(0, \frac{1}{3}]$	$2n+1$	$3n+1$	$6n+3$
$(\frac{1}{3}, \frac{1}{2}]$	$2n+1$	$3n+2$	$6n+4$
$(\frac{1}{2}, \frac{2}{3}]$	$2n+2$	$3n+2$	$6n+5$
$(\frac{2}{3}, 1)$	$2n+2$	$3n+3$	$6n+6$

所以对每个整数 n，A 包含的值为：$6n, 6n+3, 6n+4, 6n+5, 6n+6$。即 $A = \mathbb{Z} \setminus \{6k+1, 6k+2 \mid k \in \mathbb{Z}\}$（缺少形如 $6k+1$ 和 $6k+2$ 的整数）。$B = \{1,2,3,\ldots,2023\}$。$A \cap B$ 即 B 中不被排除的数。排除的数形如 $6k+1$ 或 $6k+2$ 且小于 2024：

- $6k + 1 < 2024 \Rightarrow k \le 337$，共 338 个（$k = 0, 1, \ldots, 337$）

- $6k+2<2024 \Rightarrow k \le 336$，共 337 个（$k=0,1,\ldots,336$）

排除总数 $=338+337=675$ 因此：

$$|A\cap B| = 2023-675=1348$$

11.10.8 八、反证法与有理数 (10%)

题目：证明：若 x^3 是无理数，则 x 是无理数。**考点**：第一章证明方法——逆否命题 **解析**：采用**逆否命题**进行证明。原命题的逆否命题为：

若 x 是有理数，则 x^3 是有理数。

证明：设 x 是有理数，则可写为 $x=\dfrac{p}{q}$，其中 $p,q\in\mathbb{Z}$，$q\neq 0$，且 $\gcd(p,q)=1$（既约分数）。则：

$$x^3=\left(\frac{p}{q}\right)^3=\frac{p^3}{q^3}$$

由于 $p^3,q^3\in\mathbb{Z}$，且 $q^3\neq 0$，所以 x^3 是有理数。

原命题与其逆否命题等价，故原命题成立。

11.10.9 九、数学归纳法——幂平均不等式 (10%)

题目：用归纳法证明：若 $x>0,y>0$，则

$$\frac{x^n+y^n}{2}\ge\left(\frac{x+y}{2}\right)^n$$

对所有正整数 n 成立。**考点**：第五章数学归纳法——不等式证明；幂平均不等式（Power Mean Inequality）特例 **解析**：**归纳基础**：$n=1$：

$$\frac{x+y}{2}\ge\frac{x+y}{2}$$

等号成立。**归纳假设**：设 $n=k$ 时命题成立，即：

$$\frac{x^k+y^k}{2}\ge\left(\frac{x+y}{2}\right)^k$$

归纳步骤：需证 $n=k+1$ 时：

$$\frac{x^{k+1}+y^{k+1}}{2}\ge\left(\frac{x+y}{2}\right)^{k+1}$$

由归纳假设两边乘以 $\dfrac{x+y}{2}$：

$$\left(\frac{x+y}{2}\right)^{k+1}\le\frac{x+y}{2}\cdot\frac{x^k+y^k}{2}$$

只需证明：

$$\frac{x+y}{2}\cdot\frac{x^k+y^k}{2}\le\frac{x^{k+1}+y^{k+1}}{2}$$

等价于：

$$(x+y)(x^k+y^k)\le 2(x^{k+1}+y^{k+1})$$

展开左边：

$$x^{k+1}+xy^k+yx^k+y^{k+1}\le 2x^{k+1}+2y^{k+1}$$

移项：

$$0\le x^{k+1}+y^{k+1}-xy^k-yx^k$$

因式分解：

$$x^{k+1}+y^{k+1}-xy^k-yx^k=x^k(x-y)+y^k(y-x)=(x-y)(x^k-y^k)$$

由于 $x>0,y>0$：

- 若 $x\ge y$，则 $x-y\ge 0$ 且 $x^k-y^k\ge 0$，乘积 ≥ 0
- 若 $x<y$，则 $x-y<0$ 且 $x^k-y^k<0$，乘积 >0

因此 $(x-y)(x^k-y^k)\ge 0$ 恒成立。于是：

$$\frac{x+y}{2}\cdot\frac{x^k+y^k}{2}\le\frac{x^{k+1}+y^{k+1}}{2}$$

结合归纳假设：

$$\left(\frac{x+y}{2}\right)^{k+1}\le\frac{x+y}{2}\cdot\frac{x^k+y^k}{2}\le\frac{x^{k+1}+y^{k+1}}{2}$$

由数学归纳法，对所有正整数 n 成立。证毕。

11.11 DMA Quiz 1 题解—2025 年

11.11.1 一、判断题 (True/False) (35%)

评分规则：答对 +5 分，空白 0 分，答错 -2 分（倒扣机制，防止乱猜）

题目 1a

题目：若 x 不在 A 中出现，则 $\exists x(P(x)\to A)\equiv\forall xP(x)\to A$。**答案**：T **考点**：第一章谓词与量词——量词的分配律，蕴含律 **解析**：

$$\begin{aligned}\exists x(P(x)\to A)&\equiv\exists x(\neg P(x)\vee A) && \text{(蕴含律)}\\ &\equiv(\exists x\neg P(x))\vee A && (\exists\text{ 对 }\vee\text{ 分配，}x\text{ 在}A\text{ 中不自由})\\ &\equiv\neg(\forall xP(x))\vee A && \text{(量词否定)}\\ &\equiv\forall xP(x)\to A && \text{(蕴含律)}\end{aligned}$$

关键步骤：由于 x 在 A 中不出现（即 A 是闭公式或不含变量 x），存在量词可以直接分配于析取式。答案为 **T**。

题目 1b

题目：若 A,B,C 是集合，则 $A-(B\cap C)=(A-B)\cup(A-C)$。**答案**：T **考点**：第二章集合运算——德摩根律的集合版本 **解析**：

$$\begin{aligned}x\in A-(B\cap C)&\iff x\in A\wedge x\notin(B\cap C)\\ &\iff x\in A\wedge(x\notin B\vee x\notin C) && \text{(德摩根律)}\\ &\iff(x\in A\wedge x\notin B)\vee(x\in A\wedge x\notin C) && \text{(分配律)}\\ &\iff x\in(A-B)\cup(A-C)\end{aligned}$$

集合相等得证。答案为 **T**。

题目 1c

题目：若 n 是整数，则 $\lceil n/2\rceil+\lfloor n/2\rfloor=n$。**答案**：T **考点**：第二章序列与求和——取整函数（Ceiling & Floor）**解析**：

- 若 $n=2k$（偶数）：$\lceil k\rceil+\lfloor k\rfloor=k+k=2k=n$
- 若 $n=2k-1$（奇数）：$\lceil k-0.5\rceil+\lfloor k-0.5\rfloor=k+(k-1)=2k-1=n$

答案为 **T**。

题目 1d

题目：设 $P(x,y)$ 是一个谓词，论域为 $\{1,2,3,4\}$。已知 $P(1,3),P(2,1),P(2,4),P(3,2),P(3,4),P(4,1),P(4,4)$ 为真，其余为假。则 $\forall x\exists y((x\le y)\wedge P(x,y))$ 为真。**答案**：T **考点**：第一章嵌套量词——量词真值判定 **解析**：逐 x 验证：

x	可选择 y	$(x\le y)\wedge P(x,y)$
1	$y=3$	$(1\le 3)\wedge P(1,3)\equiv\mathrm{T}\wedge\mathrm{T}\equiv\mathrm{T}$
2	$y=4$	$(2\le 4)\wedge P(2,4)\equiv\mathrm{T}\wedge\mathrm{T}\equiv\mathrm{T}$
3	$y=4$	$(3\le 4)\wedge P(3,4)\equiv\mathrm{T}\wedge\mathrm{T}\equiv\mathrm{T}$
4	$y=4$	$(4\le 4)\wedge P(4,4)\equiv\mathrm{T}\wedge\mathrm{T}\equiv\mathrm{T}$

每个 x 都能找到满足条件的 y，故全称量化命题为真。答案为 **T**。

题目 1e

题目：$n^{0.01}$ 是 $O(\log n)$。（注：原卷文字略有损毁，根据参考答案推断为此题）**答案**：F **考点**：第三章函数的增长——Big-O 记号，多项式与对数的比较 **解析**：对于任意正常数 $a>0,b>0$，有 $n^a=\Omega((\log n)^b)$，即多项式函数增长严格快于任何对数函数的幂。特别地，$n^{0.01}/\log n\to\infty$ 当 $n\to\infty$。因此 $n^{0.01}$ 不是 $O(\log n)$。答案为 **F**。

注意：若函数趋于 0（如 $(0.01)^n$），则它会是 $O(\log n)$，但这里 $n^{0.01}$ 是增长函数。

题目 1f

题目：$(0,1)$ 中十进制表示只由 0 和 1 组成的正实数集合是可数的。**答案**：F **考点**：第二章集合的基数——Cantor 对角线论证 **解析**：此题与 2022 年第六题完全相同。该集合实际上是**不可数**的（由 Cantor 对角线法可证）。假设可数，列出所有元素 $r_1,r_2,r_3,\ldots$，构造新数 $r=0.b_1b_2b_3\cdots$，其中：

$$b_i=\begin{cases}1 & \text{若}r_i\text{ 的第}i\text{ 位}=0\\ 0 & \text{若}r_i\text{ 的第}i\text{ 位}=1\end{cases}$$

则 r 的每个数码都是 0 或 1，故 r 属于该集合；但 r 与每个 r_i 都至少在第 i 位不同，矛盾。故该集合不可数，原题为假。答案为 **F**。

题目 1g

题目：$2025^{2026}\equiv 1\pmod{2027}$。**答案**：T **考点**：第四章数论——费马小定理（Fermat's Little Theorem）**解析**：

- 2027 是素数（可验证：$\sqrt{2027}\approx 45$，不能被小于 45 的任何素数整除）。
- 2025 与 2027 互素（$\gcd(2025,2027)=1$，因为 $2027-2025=2$，而 2025 是奇数）。
- 由费马小定理：若 p 为素数且 $p\nmid a$，则 $a^{p-1}\equiv 1\pmod p$。
- 这里 $p=2027$，$a=2025$，$p-1=2026$，故：

$$2025^{2026}\equiv 1\pmod{2027}$$

答案为 **T**。

11.11.2 二、命题等价变换 (12%)

题目：写出一个与 $p \oplus q$（异或）等价的命题：

- (a) 只使用 $p, q, \neg$ 和联结词 $\wedge$（6%）
- (b) 只使用 p, q 和联结词 $|$（NAND）（6%）

题目 2a 解析

$$p \oplus q \equiv (p \wedge \neg q) \vee (\neg p \wedge q)$$

用 $\wedge$ 和 $\neg$ 表示 $\vee$（德摩根律）：

$$p \oplus q \equiv \neg(\neg(p \wedge \neg q) \wedge \neg(\neg p \wedge q))$$

答案：$\neg(\neg(p \wedge \neg q) \wedge \neg(\neg p \wedge q))$

题目 2b 解析

NAND 定义：$p \mid q \equiv \neg(p \wedge q)$ 逐步构造：

- $\neg q \equiv q \mid q$
- $\neg p \equiv p \mid p$
- $p \wedge \neg q \equiv \neg(p \mid (q \mid q)) \equiv (p \mid (q \mid q)) \mid (p \mid (q \mid q))$
- $\neg p \wedge q \equiv \neg((p \mid p) \mid q) \equiv ((p \mid p) \mid q) \mid ((p \mid p) \mid q)$
- $(A) \vee (B) \equiv \neg(\neg A \wedge \neg B)$

代入化简后得简洁形式：**答案**：$(p \mid (q \mid q)) \mid ((p \mid p) \mid q)$ 验证：

$$\begin{aligned}(p \mid (q \mid q)) \mid ((p \mid p) \mid q) &= \neg(p \wedge \neg q) \mid \neg(\neg p \wedge q) \\ &= \neg(\neg(p \wedge \neg q) \wedge \neg(\neg p \wedge q)) \\ &= (p \wedge \neg q) \vee (\neg p \wedge q) \\ &= p \oplus q\end{aligned}$$

11.11.3 三、主合取范式 (9%)

题目：求 $(p \oplus q) \vee r$ 的全合取范式（full CNF）。**答案**：$(p \vee q \vee r) \wedge (\neg p \vee \neg q \vee r)$ **考点**：第一章命题逻辑——主合取范式，异或 **解析**：计算真值表：

p	q	r	$p \oplus q$	$(p \oplus q) \vee r$
F	F	F	F	F
F	F	T	F	T
F	T	F	T	T
F	T	T	T	T
T	F	F	T	T
T	F	T	T	T
T	T	F	F	F
T	T	T	F	T

公式为假的行：(F, F, F) 和 (T, T, F)。对应的极大项（maxterm）：

- (F, F, F)：$p \vee q \vee r$
- (T, T, F)：$\neg p \vee \neg q \vee r$

全 CNF（所有极大项的合取）：

$$(p \vee q \vee r) \wedge (\neg p \vee \neg q \vee r)$$

11.11.4 四、函数的枚举与分类 (8%)

题目：列出所有从 $A = \{1, 2\}$ 到 $B = \{a, b\}$ 的函数，并指出哪些是双射（bijection），哪些是满射（surjection）。**考点**：第二章函数——映射的种类：单射、满射、双射 **解析**：从 A 到 B 共有 $|B|^{|A|} = 2^2 = 4$ 个函数：

函数	$f(1)$	$f(2)$	单射?	满射?	双射?
f_1	a	a	否（1 和 2 都映射到 a）	否（b 无原像）	否
f_2	a	b	是	是	**是**
f_3	b	a	是	是	**是**
f_4	b	b	否	否（a 无原像）	否

结果：

- 所有函数：f_1, f_2, f_3, f_4
- 双射：f_2, f_3
- 满射：f_2, f_3
- 无双射以外的满射（因为 $|A| = |B|$，此时单射等价于满射等价于双射）

11.11.5 五、简化剩余系 (9%)

题目：将所有与 77 互质的正整数排成严格递增序列，求第 600 项。**答案**：769 **考点**：第四章数论——欧拉函数 φ，简化剩余系（Reduced Residue System），周期性 **解析**：

$$77 = 7 \times 11$$

与 77 互质的正整数就是不被 7 整除也不被 11 整除的数。每 77 个连续整数中，与 77 互质的个数为欧拉函数 $\varphi(77)$：

$$\varphi(77) = \varphi(7 \times 11) = 77 \times \left(1 - \frac{1}{7}\right) \times \left(1 - \frac{1}{11}\right) = 77 \times \frac{6}{7} \times \frac{10}{11} = 60$$

即序列以 60 为周期，每增加 77，序列的值增加 77：

$$a_{n+60} = a_n + 77$$

因此：

$$a_{600} = a_{60} + 77 \times \left(\frac{600}{60} - 1\right) = a_{60} + 77 \times 9 = a_{60} + 693$$

a_{60} 是小于 77 且与 77 互质的最大正整数。由于 $77 - 1 = 76$ 与 77 互质，$a_{60} = 76$。故：

$$a_{600} = 76 + 693 = 769$$

11.11.6 六、中国剩余定理 (9%)

题目：利用中国剩余定理证明中的构造方法，求解同余方程组：

$$x \equiv 1 \pmod 3, \quad x \equiv 2 \pmod 5, \quad x \equiv 3 \pmod 8$$

考点：第四章求解同余式——中国剩余定理（Chinese Remainder Theorem）**解析**：**步骤 1**：确定模数及总模。

$$m_1 = 3,\ m_2 = 5,\ m_3 = 8, \quad M = 3 \times 5 \times 8 = 120$$

步骤 2：计算各 $M_k = M/m_k$。

$$M_1 = 120/3 = 40, \quad M_2 = 120/5 = 24, \quad M_3 = 120/8 = 15$$

步骤 3：求解 y_k 使 $M_k y_k \equiv 1 \pmod{m_k}$。

- $40y_1 \equiv 1 \pmod 3$：$40 \equiv 1 \pmod 3$，所以 $1 \cdot y_1 \equiv 1 \pmod 3 \Rightarrow y_1 = 1$
- $24y_2 \equiv 1 \pmod 5$：$24 \equiv 4 \pmod 5$，$4y_2 \equiv 1 \pmod 5 \Rightarrow y_2 = 4$（因为 $4 \times 4 = 16 \equiv 1$）
- $15y_3 \equiv 1 \pmod 8$：$15 \equiv 7 \pmod 8$，$7y_3 \equiv 1 \pmod 8 \Rightarrow y_3 = 7$（因为 $7 \times 7 = 49 \equiv 1$）

步骤 4：构造解。

$$\begin{aligned}x &\equiv \sum_{k=1}^{3} a_k M_k y_k \pmod M \\ &= 1 \times 40 \times 1 + 2 \times 24 \times 4 + 3 \times 15 \times 7 \\ &= 40 + 192 + 315 \\ &= 547 \\ &\equiv 547 - 4 \times 120 = 67 \pmod{120}\end{aligned}$$

答案：$x \equiv 67 \pmod{120}$，即 $x = 67 + 120k$，$k \in \mathbb{Z}$。**验证**：

- $67 \bmod 3 = 1$
- $67 \bmod 5 = 2$
- $67 \bmod 8 = 3$

11.11.7 七、集合分配律的推广 (9%)

题目：证明分配律

$$A_1 \cup (A_2 \cap \cdots \cap A_n) = (A_1 \cup A_2) \cap \cdots \cap (A_1 \cup A_n)$$

对所有 $n > 2$ 成立。**考点**：第二章集合运算——分配律，元素法证明 **解析**：用元素法直接证明：

$$\begin{aligned}x \in A_1 \cup (A_2 \cap \cdots \cap A_n) &\iff x \in A_1 \vee x \in (A_2 \cap \cdots \cap A_n) \\ &\iff x \in A_1 \vee (x \in A_2 \wedge \cdots \wedge x \in A_n) \\ &\iff (x \in A_1 \vee x \in A_2) \wedge \cdots \wedge (x \in A_1 \vee x \in A_n) \qquad \text{(分配律)} \\ &\iff x \in (A_1 \cup A_2) \cap \cdots \cap (A_1 \cup A_n)\end{aligned}$$

由集合外延公理，等式成立。也可用数学归纳法证明：

- 基础 $n = 2$ 即基本分配律 $A_1 \cup (A_2 \cap A_3) = (A_1 \cup A_2) \cap (A_1 \cup A_3)$。
- 归纳步骤与 2022 年第五题类似。

11.11.8 八、斐波那契表示定理 (9%)

题目：证明每个大于 2 的正整数都可以表示为不同的斐波那契数之和。**注**：此处斐波那契数定义为 $f_1 = 1, f_2 = 2, f_3 = 3, f_4 = 5, f_5 = 8, \ldots$（即略去 $f_0 = 0$，直接从 $f_1 = 1, f_2 = 2$ 开始）。**考点**：第五章数学归纳法与强归纳法——菲波那契数的性质，Zeckendorf 定理 **解析**：**方法一（普通归纳法）**：**归纳基础**：$n = 3$。$3 = f_1 + f_2 = 1 + 2$，成立。**归纳假设**：设 n 可表示为不同斐波那契数之和：$n = f_{i_1} + f_{i_2} + \cdots + f_{i_k}$，其中 $i_1 < i_2 < \cdots < i_k$。**归纳步骤**：考虑 $n+1$。$n + 1 = 1 + f_{i_1} + f_{i_2} + \cdots + f_{i_k}$。

- 若 $i_1 > 1$（即 $f_{i_1} \neq 1$），则 1 与所有 f_{i_j} 均不同，故 $n+1$ 已被表示为不同斐波那契数之和。
- 若 $i_1 = 1$，则 $f_{i_1} = f_1 = 1$，此时出现重复的 1。设最大的 m 满足 $i_j = j$（即 $i_1 = 1, i_2 = 2, \ldots, i_m = m$），且 $i_{m+1} \geq m + 2$。利用斐波那契性质 $f_j + f_{j+1} = f_{j+2}$，将连续的项逐对合并：
- 若 m 为偶数：

$$n + 1 = 1 + (f_1 + f_2) + (f_3 + f_4) + \cdots + (f_{m-1} + f_m) + \sum_{j=m+1}^{k} f_{i_j}$$

$$= f_3 + f_5 + \cdots + f_{m+1} + \sum_{j=m+1}^{k} f_{i_j}$$

  由于 $f_{m+1} < f_{m+2} \leq f_{i_{m+1}}$（关键的不等关系保证了合并后的项互不相同且大于前面的项），所有项互异。
- 若 m 为奇数：

$$n + 1 = 1 + f_1 + (f_2 + f_3) + \cdots + (f_{m-1} + f_m) + \sum_{j=m+1}^{k} f_{i_j}$$

$$= f_2 + f_4 + \cdots + f_{m+1} + \sum_{j=m+1}^{k} f_{i_j}$$

  同样所有项互异。

因此 $n+1$ 可表示为不同斐波那契数之和。由归纳法，命题对所有 $n > 2$ 成立。

方法二（强归纳法）：**归纳基础**：$n = 3 = f_1 + f_2$，成立。**强归纳假设**：假设所有大于 2 且小于 n 的整数都可以表示为不同斐波那契数之和。**归纳步骤**：设 f_k 是小于 n 的最大斐波那契数（由于 $n > 2$，$f_k \geq 3$）。

- 若 $n - f_k = 1$ 或 2，则 $n = f_k + 1$ 或 $n = f_k + 2$。由于 1 和 2 本身就是斐波那契数（$f_1 = 1, f_2 = 2$），且 $f_k > f_2 = 2$（因为 $n > 3$），所以这些斐波那契数互异。
- 若 $n - f_k \geq 3$，由强归纳假设，$n - f_k$ 可表示为不同斐波那契数之和：$n - f_k = f_{i_1} + f_{i_2} + \cdots + f_{i_t}$。由于 f_k 是小于 n 的最大斐波那契数，所有 $f_{i_j} < f_k$，因此这些斐波那契数与 f_k 互异。

故 $n = f_k + f_{i_1} + \cdots + f_{i_t}$ 是不同斐波那契数之和。由强归纳法，命题对所有 $n > 2$ 成立。证毕。

11.12 DMA Quiz 2 题解—2022 年

11.12.1 题目 1

题目：(a) Find the next larger permutation in lexicographic order after 76154238. (5%) (b) Find the next larger 5-combinations of the set $\{1, 2, 3, 4, 5, 6, 7, 8\}$ after $\{3, 4, 5, 7, 8\}$. (5%) **考点**：第六章排列与组合（Chapter 6.6 Generating Permutations and Combinations）——字典序后继 **解析**：**(a)** 字典序找下一个排列的算法：

1. 从右向左找第一个满足 $a_i < a_{i+1}$ 的位置（即最后一个升序位置）。
2. 在 a_i 右边找到比 a_i 大的最小元素 a_j。
3. 交换 a_i 和 a_j。
4. 将 a_{i+1} 到末尾反转。

对 76154238：

- 从右向左：$8 > 3$，继续；$3 < 8$，找到 $a_i = 3$（第 6 位，从 1 开始数）。实际上，排列索引：$p_1 = 7, p_2 = 6, p_3 = 1, p_4 = 5, p_5 = 4, p_6 = 2, p_7 = 3, p_8 = 8$
- 从右找第一个 $p_i < p_{i+1}$：$p_8 = 8, p_7 = 3, p_7 < p_8$，所以 $i = 7$。
- 在 p_7 右边（即 p_8）比 3 大的最小元素：8。
- 交换 $p_7 = 3$ 和 $p_8 = 8$：得 76154283。
- 反转 p_{i+1} 到末尾（即 p_8 到 p_8，只有一个元素，不变）。

结果为 76154283。验证：76154238 之后的排列是 76154283，再下一个是 76154328，正确。**(b)** 找字典序下一个 r-组合的算法：

1. 从右向左找第一个满足 $a_i < n - r + i$ 的位置。
2. 将该位置的元素加 1。
3. 后续所有元素在前一个基础上加 1。

对 $\{3, 4, 5, 7, 8\}$，$n = 8, r = 5$：

- 从左到右：位置 1（3），最大可取 $8 - 5 + 1 = 4$，$3 < 4$，可以继续。
- 从右向左找：
- 位置 5（8），最大可取 8，$8 = 8$，跳过。
- 位置 4（7），最大可取 7，$7 = 7$，跳过。

- 位置 3（5），最大可取 6，$5 < 6$，找到。
- 位置 3 的元素 5 加 1 得 6。
- 位置 $4 = 6 + 1 = 7$，位置 $5 = 7 + 1 = 8$。
- 得 $\{3, 4, 6, 7, 8\}$。

结果为 $\{3, 4, 6, 7, 8\}$。

11.12.2 题目 2

题目：Allowing repeated characters, how many strings of six characters from $\{`0' \sim `9'\}$: (24%) (a) contain '1'? (b) contain exactly one '1'? (c) contain `1' and 2', where `1' is somewhere to the left of 2' in the string? (d) contain `1' and 2', where `1' is somewhere to the left of 2' in the string, with all the characters distinct? (e) begin and end with the same character? (f) consists of exactly 3 different characters? **考点**：第六章排列与组合（Chapter 6 Counting）——容斥原理、排列组合计数 **解析**：总共有 10^6 个长度为 6 的字符串（每位可从 0-9 中选择，可重复）。**(a)** 包含至少一个'1'。用补集：不含'1' 的字符串每位有 9 种选择，共 9^6 个。

$$10^6 - 9^6 = 1\,000\,000 - 531\,441 = 468\,559$$

(b) 恰好包含一个'1'。先选'1' 出现的位置：$\binom{6}{1} = 6$ 种。其余 5 位每位从 0,2-9 中选（排除'1'），每位置 9 种选择。

$$6 \times 9^5 = 6 \times 59\,049 = 354\,294$$

(c) 包含'1' 和'2'，且'1' 在'2' 的左边。先计算同时包含至少一个'1' 和至少一个'2' 的字符串总数，再除以对称性（'1' 左'2' 右与'2' 左'1' 右各占一半）。用容斥原理计算包含 1 和 2 的字符串数：

- 总：10^6
- 不含'1'：9^6
- 不含'2'：9^6
- 不含'1' 也不含'2'：8^6

由容斥原理，至少含一个'1' 和一个'2' 的数量：

$$N = 10^6 - 2 \times 9^6 + 8^6$$

由对称性，其中'1' 在'2' 左侧的占一半：

$$\frac{N}{2} = \frac{10^6 - 2 \times 9^6 + 8^6}{2}$$

数值计算：$10^6 = 1\,000\,000$ $9^6 = 531\,441$ $8^6 = 262\,144$ $N = 1\,000\,000 - 1\,062\,882 + 262\,144 = 199\,262$ $\frac{N}{2} = 99\,631$ **(d)** 包含'1' 和'2'，'1' 在'2' 左侧，且所有字符互不相同（不可重复）。选 6 个不同的字符，其中必须包含'1' 和'2'，其余 4 个从剩下的 8 个数字中选：$\binom{8}{4}$ 种选择。选定 6 个字符后，要求'1' 在'2' 左边。在 6 个位置的排列中，'1' 和'2' 的相对顺序各占一半。总排列数为 6!，其中'1' 在'2' 左边的占 $\frac{1}{2}$。

$$\binom{8}{4} \times \frac{6!}{2} = 70 \times 360 = 25\,200$$

(e) 首尾字符相同。首位和末位必须选同一个字符，有 10 种选择。中间 4 位任意，每位 10 种。

$$10 \times 10^4 = 10^5 = 100\,000$$

(f) 恰好由 3 种不同字符组成。这是一个有难度的计数问题。需要先选 3 种字符，然后计算仅由这 3 种字符组成且每种至少出现一次的 6 位字符串个数，再乘以选择数。步骤：

1. 从 10 种字符中选 3 种：$\binom{10}{3} = 120$。
2. 对于选定的 3 种字符，计算长度为 6、每位来自这 3 种字符、且每种至少出现一次的字符串数。这就是将 6 个位置分配给 3 种字符，每种至少 1 个，且同种字符不可区分（但位置重要）。

   用容斥原理：

- 总数（仅用这 3 种，但允许不出现某些字符）：3^6
- 减去恰好缺了 1 种字符的：$\binom{3}{1} \times 2^6$
- 加上恰好缺了 2 种字符的（即全为同一种）：$\binom{3}{2} \times 1^6$

  由容斥原理：

$$3^6 - \binom{3}{1} 2^6 + \binom{3}{2} 1^6 = 729 - 3 \times 64 + 3 = 729 - 192 + 3 = 540$$

1. 所以总数为：

$$\binom{10}{3} \times 540 = 120 \times 540 = 64\,800$$

11.12.3 题目 3

题目：Find the coefficient of x^9 in the expansion of $(x/2-3/x)^{15}$. (6%) **考点**：第六章二项式定理（Chapter 6.4 Binomial Theorem）——广义二项式展开的系数计算 **解析**：展开式的一般项为：

$$\binom{15}{k}\left(\frac{x}{2}\right)^{15-k}\left(-\frac{3}{x}\right)^{k}=\binom{15}{k}\left(\frac{1}{2}\right)^{15-k}(-3)^k\cdot x^{15-k}\cdot x^{-k}=\binom{15}{k}\frac{(-3)^k}{2^{15-k}}\cdot x^{15-2k}$$

要求 x^9 的系数，即 $15-2k=9$，解得 $k=3$。代入 $k=3$：

$$\binom{15}{3}\frac{(-3)^3}{2^{12}}\cdot x^9=455\cdot\frac{(-27)}{4096}\cdot x^9=-\frac{455\times 27}{4096}x^9=-\frac{12\,285}{4096}x^9$$

系数为 $-\dfrac{12\,285}{4096}$。

11.12.4 题目 4

题目：How many relations are there on a set with n elements that are: (24%) (a) both reflexive and symmetric? (b) both symmetric and antisymmetric? (c) neither reflexive nor irreflexive? (d) transitive? (if $n=2$) (e) equivalence relation? (if $n=4$) (f) partial order relation? (if $n=2$) (Tips: A relation R on the set A is **irreflexive** if for every $a\in A,(a,a)\notin R$.) **考点**：第九章关系（Chapter 9 Relations）——关系的性质、等价关系、偏序关系 **解析**：集合 A 有 n 个元素，则 $A\times A$ 共有 n^2 个有序对。关系是 $A\times A$ 的子集。**(a)** 自反且对称的关系。

- 自反要求所有 (a_i,a_i) 必须属于关系：n 个有序对已确定。
- 对称要求：对 $i\neq j$，(a_i,a_j) 与 (a_j,a_i) 要么同时出现，要么同时不出现。所以 $i\neq j$ 的 $n(n-1)$ 个有序对可以分成 $\frac{n(n-1)}{2}$ 组。
- 每组（一个无序对的两个方向）独立选择是否在关系中：$2^{n(n-1)/2}$ 种选择。

故自反且对称的关系数为：

$$2^{n(n-1)/2}$$

(b) 对称且反对称的关系。

- 对称要求：(a_i,a_j) 和 (a_j,a_i) 同时出现或同时不出现。
- 反对称要求：若 (a_i,a_j) 和 (a_j,a_i) 同时出现，则必须有 $a_i=a_j$。即当 $i\neq j$ 时，不能同时有 (a_i,a_j) 和 (a_j,a_i)。
- 所以对 $i\neq j$，只能有一个都不出现（满足对称和反对称），不能出现单个（违反对称）。所以所有 $i\neq j$ 的 (a_i,a_j) 都不在关系中。
- 自反对：(a_i,a_i) 可以自由选择在或不在（对称和反对称对自反元素自动满足）。因此每个 (a_i,a_i) 有 2 种选择。

故对称且反对称的关系数为：

$$2^n$$

即关系只能是 A 的子集的对角线关系。**(c)** 既不自反也不反自反的关系。

- 自反要求所有 n 个自反对都在关系中。
- 反自反要求所有 n 个自反对都不在关系中。
- " 既不自反也不反自反" 意味着：有些自反对在关系中，有些不在。即至少有一个自反对在关系中，也至少有一个不在。

自反的总数：2^n（每个自反对 2 种选择）。其中自反的：只有 1 种（全部选" 在"）。反自反的：只有 1 种（全部选" 不在"）。所以既不自反也不反自反的选择数（只考虑自反对部分）：对于非自反对（$i\neq j$ 的 $n(n-1)$ 个有序对），没有任何限制，可任意在或不在，共 $2^{n(n-1)}$ 种。故答案为：

$$(2^n-2)\times 2^{n(n-1)}$$

(d) 传递关系（$n=2$）。设 $A=\{a,b\}$。$A\times A$ 有 4 个有序对：$(a,a),(a,b),(b,a),(b,b)$。列出所有可能的传递关系（共 $2^4=16$ 个关系），检查哪些是传递的。传递性要求：若 $(x,y)\in R$ 且 $(y,z)\in R$，则 $(x,z)\in R$。逐一检查所有 16 个子集：包含 (a,b) 和 (b,a) 时必须包含 (a,a) 和 (b,b)：

- 包含 $(a,b),(b,a),(a,a),(b,b)$：4 个都在——传递
- 加上其他组合如果已经有 $(a,b),(b,a)$，缺少 (a,a) 或 (b,b) 则不传递。

我们系统分类（考虑是否包含 (a,b) 和 (b,a)）：情况 1：不包含 (a,b) 且不包含 (b,a)。此时传递性条件的前件永远不会满足，任何子集都传递。包含 (a,a) 和 (b,b) 各有 2 种选择：$2\times 2=4$ 种。情况 2：包含 (a,b) 但不包含 (b,a)。此时唯一需要检查的是：若 (b,a) 不在则无链。但需要：

- 若 $(a,b)\in R$ 且 $(b,b)\in R$，需 $(a,b)\in R$（已满足）。
- 没有其他限制。$(a,a),(b,b)$ 自由。

共：(a,a) 2 种 $\times$ (b,b) 2 种 $=4$ 种。情况 3：包含 (b,a) 但不包含 (a,b)。对称于情况 2：4 种。情况 4：同时包含 (a,b) 和 (b,a)。此时需要 $(a,a)\in R$ 且 $(b,b)\in R$。(a,a) 和 (b,b) 都必须选。共：1 种（(a,a) 和 (b,b) 都选）。总计：4+4+4+1 $=13$ 种。另一种方法，列出所有 16 种关系并逐项检查：传递关系有：$\varnothing,\{(a,a)\},\{(b,b)\},\{(a,a),(b,b)\},\{(a,b)\},\{(a,b),(a,a)\},\{(a,b),(b,b)\},\{(a,b),(a,a),(b,b)\},\{(b,a)\},\{(b,a),(a,a)\},\{(b,a),(b,b)$ 共 13 种。**(e)** 等价关系（$n=4$）。集合 A 上的等价关系与 A 的划分一一对应。等价关系的数量等于 4 元集合的划分数，即 Bell 数 B_4。列出所有划分方式：

1. 1 个块（所有元素在一起）：1 种 $\{a,b,c,d\}$
2. 分成 3+1（一个 3 元块加一个单元素块）：$\binom{4}{3}=4$ 种

3. 分成 2+2（两个 2 元块）：$\frac{\binom{4}{2}}{2}=3$ 种
4. 分成 2+1+1：$\binom{4}{2}=6$ 种
5. 分成 1+1+1+1（全部分开）：1 种

总计：$1+4+3+6+1=15$ 种。所以 $n=4$ 时的等价关系数为 **15**。**(f)** 偏序关系（$n=2$）。偏序关系要求自反、反对称、传递。设 $A=\{a,b\}$。自反要求 $(a,a),(b,b)$ 必须在关系中。反对称要求：不能同时有 (a,b) 和 (b,a)。传递要求：若 (a,b) 且 (b,a) 同时出现则需 $(a,a),(b,b)$（但反对称已经禁止了同时出现）。所以可能的偏序关系：

1. 只有自反对：$\{(a,a),(b,b)\}$
2. 加上 (a,b)：$\{(a,a),(b,b),(a,b)\}$
3. 加上 (b,a)：$\{(a,a),(b,b),(b,a)\}$

共 **3** 种。

11.12.5 题目 5

题目：Write the first 6 terms of the sequence determined by the generating function: $(1+x)/(1-x)$. (10%) **考点**：第八章生成函数（Chapter 8.5 Generating Functions）——从生成函数到序列 **解析**：

$$\frac{1+x}{1-x}=(1+x)\cdot\frac{1}{1-x}$$

已知 $\dfrac{1}{1-x}=\displaystyle\sum_{n=0}^{\infty}x^n$，即序列 $\{1,1,1,1,1,1,\ldots\}$。所以：

$$(1+x)\sum_{n=0}^{\infty}x^n=\sum_{n=0}^{\infty}x^n+x\sum_{n=0}^{\infty}x^n=\sum_{n=0}^{\infty}x^n+\sum_{n=0}^{\infty}x^{n+1}$$

写成以 x^n 为基准：

$$=1+\sum_{n=1}^{\infty}x^n+\sum_{n=1}^{\infty}x^n=1+2\sum_{n=1}^{\infty}x^n$$

即序列为：$a_0=1,a_1=2,a_2=2,a_3=2,a_4=2,a_5=2$ 前 6 项：$1,2,2,2,2,2$。或者直接展开：

$$\frac{1+x}{1-x}=(1+x)(1+x+x^2+x^3+x^4+x^5+\cdots)=1+2x+2x^2+2x^3+2x^4+2x^5+\cdots$$

11.12.6 题目 6

题目：Find the solution to the following iteration relation: (16%)

$$a_n=5a_{n-1}-6a_{n-2}+2^n+3\quad\text{for } n\geq 2,\quad a_0=1,a_1=1.$$

考点：第八章递推关系（Chapter 8.2-8.4 Solving Recurrence Relations）——非齐次线性递推 **解析**：这是一个非齐次线性递推关系。解由齐次通解 + 特解组成。**Step 1: 齐次通解**齐次部分：$a_n^{(h)}-5a_{n-1}^{(h)}+6a_{n-2}^{(h)}=0$。特征方程：$r^2-5r+6=0$，解得 $r=2,3$。齐次通解：

$$a_n^{(h)}=\alpha\cdot 2^n+\beta\cdot 3^n$$

Step 2: 特解非齐次项为 2^n+3，包含两项。由于 2^n 与齐次解中的 2^n 项冲突（齐次解已有 2^n），需要乘 n 修正。设特解形式：

$$a_n^{(p)}=An\cdot 2^n+B$$

（常数项 3 对应常数特解 B。）代入递推式：

$$An\cdot 2^n+B=5[A(n-1)\cdot 2^{n-1}]-6[A(n-2)\cdot 2^{n-2}]+2^n+3$$

先处理含 2^n 的项：

$$An\cdot 2^n=5A(n-1)\cdot 2^{n-1}-6A(n-2)\cdot 2^{n-2}+2^n$$

除以 2^{n-2}：

$$4An=5A(n-1)\cdot 2-6A(n-2)+4$$

$$4An=10A(n-1)-6A(n-2)+4$$

$$4An=10An-10A-6An+12A+4$$

$$4An=4An+2A+4$$

所以 $2A+4=0$，即 $A=-2$。再处理常数项：

$$B=5B-6B+3\implies B=-B+3\implies 2B=3\implies B=\frac{3}{2}$$

特解为：

$$a_n^{(p)} = -2n \cdot 2^n + \frac{3}{2}$$

Step 3: 通解

$$a_n = \alpha \cdot 2^n + \beta \cdot 3^n - 2n \cdot 2^n + \frac{3}{2}$$

Step 4: 代入初始条件 $n=0$: $a_0 = \alpha+\beta+\frac{3}{2} = 1 \implies \alpha+\beta = -\frac{1}{2}$ $n=1$: $a_1 = 2\alpha+3\beta-2\cdot 2^1+\frac{3}{2} = 2\alpha+3\beta-4+\frac{3}{2} = 2\alpha+3\beta-\frac{5}{2} = 1$ $\implies 2\alpha+3\beta = \frac{7}{2}$ 解方程组：

$$\begin{cases} \alpha+\beta = -\frac{1}{2} \\ 2\alpha+3\beta = \frac{7}{2} \end{cases}$$

由第 1 式：$\alpha = -\frac{1}{2} - \beta$。代入第 2 式：$2(-\frac{1}{2}-\beta)+3\beta = -1-2\beta+3\beta = -1+\beta = \frac{7}{2} \implies \beta = \frac{9}{2} \implies \alpha = -\frac{1}{2}-\frac{9}{2} = -5$ 所以：

$$a_n = -5 \cdot 2^n + \frac{9}{2} \cdot 3^n - 2n \cdot 2^n + \frac{3}{2}$$

化简：

$$a_n = \frac{1}{2}\left[9 \cdot 3^n - (10+4n) \cdot 2^n + 3\right] = \frac{9 \cdot 3^n - (4n+10)2^n + 3}{2}$$

11.12.7 题目 7

题目：Let A be a set consisting 10 integers $10 \le a_i \le 99$ that are different from each other. Prove that A must have two different subsets without common elements, and the sums of the elements in these two subsets are equal. (10%) **考点**：第六章/第八章组合数学（Combinatorics）——鸽巢原理（Pigeonhole Principle）**解析**：考虑集合 A 的所有子集（包括空集）。共有 $2^{10} = 1024$ 个子集。每个子集的和的范围：

- 最小和：空集，和为 0。
- 最大和：全部 10 个元素的和，最大值不会超过 $10 \times 99 = 990$。
- 更精确地，因为 $10 \le a_i \le 99$ 且元素互异，最大和不会超过 $99+98+\cdots+90 = 945$。但就算粗略估计也够。

但我们需要两个不相交的子集和相等，所以我们应该考虑" 子集和" 的取值范围：

- 最小：0（空集）
- 最大：所有元素和 $\le 99+98+97+96+95+94+93+92+91+90 = 945$

所以 1024 个子集的和落在区间 $[0, 945]$ 中，只有至多 946 种可能的和值。但我们需要两个**不相交**的子集（即没有公共元素），这和鸽巢原理的常规应用稍有不同。更好的方法是：考虑所有子集（包括空集），共 1024 个。和的范围在 0 到 945 之间，有 946 种可能。由鸽巢原理，存在两个不同的子集 B, C（可能有交）使得 $sum(B) = sum(C)$。然后移除它们的公共部分：令 $B' = B \setminus C$，$C' = C \setminus B$。则 B' 与 C' 不相交，且：

$$sum(B') = sum(B) - sum(B \cap C) = sum(C) - sum(B \cap C) = sum(C')$$

所以 B' 和 C' 即为所需的一对不相交子集。但还需保证 B' 和 C' 不都是空集。如果 B 和 C 是不同的子集但和相等，则它们的对称差非空，所以 B' 和 C' 不会同时为空（若同时为空，则 $B = C$）。**更严谨的鸽巢原理版本**：共有 $2^{10} = 1024$ 个子集。最大可能和：$99+98+97+96+95+94+93+92+91+90 = \frac{10\times(90+99)}{2} = 5 \times 189 = 945$。最小和：0。所以可能的和值最多有 $945+1 = 946$ 种。由鸽巢原理，$1024 > 946$，所以存在两个不同子集 S_1, S_2 和相等。令 $S_1' = S_1 \setminus S_2$，$S_2' = S_2 \setminus S_1$，则 $S_1' \cap S_2' = \varnothing$，且：

$$\sum S_1' = \sum S_1 - \sum(S_1 \cap S_2) = \sum S_2 - \sum(S_1 \cap S_2) = \sum S_2'$$

因为 $S_1 \ne S_2$，所以 S_1' 和 S_2' 不同时为空。若 $S_1' = \varnothing$，则 $S_1 \subseteq S_2$，但 $S_1 \ne S_2$，所以 $S_2' \ne \varnothing$，反之亦然。故我们找到了一对不相交的非空子集 S_1', S_2' 和相等。**证毕**。

11.13 DMA Quiz 2 题解—2023 年

11.13.1 题目 1 (31% total)

1(a)

题目：Find the next larger permutation in lexicographic order after 276154389. (4%) **考点**：第六章排列与组合（Chapter 6.6 Generating Permutations）——字典序后继 **解析**：从右向左找第一个满足 $a_i < a_{i+1}$ 的位置：

- 276154389，从右向左：$9 > 8 > 3 > 4 > 5 > 1$？详细检查：位置：$p_1 = 2, p_2 = 7, p_3 = 6, p_4 = 1, p_5 = 5, p_6 = 4, p_7 = 3, p_8 = 8, p_9 = 9$
- 从右：$9 > 8$，继续；$8 > 3$，继续；$3 < 4$，找到 $i = 7$，$a_i = 3$。
- 在 $a_7 = 3$ 右边（$p_8 = 8, p_9 = 9$）找比 3 大的最小元素：8。
- 交换 $p_7 = 3$ 与 $p_8 = 8$：得 276154839。
- 反转 p_8 到末尾（即 $p_8 = 3, p_9 = 9 \to$ 反转后 $p_8 = 9, p_9 = 3$）：得 276154893。

结果为 276154893。

1(b)

题目：Find the next 5-combination of the set $\{1,2,3,4,5,6,7,8,9\}$ after $\{1,3,5,7,9\}$. (4%) **考点**：第六章排列与组合（Chapter 6.6 Generating Combinations）——字典序组合后继 **解析**：找下一个 r-组合的算法：从右向左找第一个满足 $a_i < n-r+i$ 的位置，将该位置元素加 1，后续元素依次递增。$n = 9, r = 5$，组合 $\{1,3,5,7,9\}$：

- 从右向左：
- $a_5 = 9$，最大可取 $9-5+5 = 9$，$9 = 9$，跳过。
- $a_4 = 7$，最大可取 $9-5+4 = 8$，$7 < 8$，找到。
- $a_4 = 7$ 加 1 得 8。
- $a_5 = 8+1 = 9$。
- 得 $\{1,3,5,8,9\}$。

结果为 $\{1,3,5,8,9\}$。

1(c)

题目：Find $3^{2023} \bmod 1997$. (4%) **考点**：第四章数论（Chapter 4.3 Primes and GCD）——模幂运算、费马小定理 **解析**：1997 是否为素数？检查：$\sqrt{1997} \approx 44.7$，用不超过 44 的素数试除知 1997 为素数。由费马小定理：若 p 为素数且 $p \nmid a$，则 $a^{p-1} \equiv 1 \pmod p$。这里 $p = 1997$，$1997 \nmid 3$，所以 $3^{1996} \equiv 1 \pmod{1997}$。计算 $2023 \bmod 1996$：$2023 = 1996+27$，所以 $3^{2023} = 3^{1996} \cdot 3^{27} \equiv 1 \cdot 3^{27} \pmod{1997}$。计算 $3^{27} \bmod 1997$：

$$\begin{aligned}
&3^1 = 3 \\
&3^2 = 9 \\
&3^4 = 81 \\
&3^8 = 81^2 = 6561 \equiv 6561 - 3 \times 1997 = 6561 - 5991 = 570 \\
&3^{16} \equiv 570^2 = 324\,900 \equiv 324\,900 - 162 \times 1997 \\
&\quad 162 \times 1997 = 162 \times 2000 - 162 \times 3 = 324\,000 - 486 = 323\,514 \\
&\quad 324\,900 - 323\,514 = 1\,386
\end{aligned}$$

所以：

$$3^{27} = 3^{16} \times 3^8 \times 3^2 \times 3^1 \equiv 1386 \times 570 \times 9 \times 3 \pmod{1997}$$

先算 1386×570：$1386 \times 570 = 1386 \times 500 + 1386 \times 70 = 693\,000 + 97\,020 = 790\,020$ $790\,020 \div 1997$：$1997 \times 395 = 1997 \times 400 - 1997 \times 5 = 798\,800 - 9\,985 = 788\,815$ $790\,020 - 788\,815 = 1\,205$ 再乘以 9：$1205 \times 9 = 10\,845$ $10\,845 \div 1997$：$1997 \times 5 = 9\,985$，余 $10\,845 - 9\,985 = 860$ 再乘以 3：$860 \times 3 = 2\,580$ $2\,580 \bmod 1997 = 2\,580 - 1\,997 = 583$ 所以 $3^{2023} \equiv 583 \pmod{1997}$。

1(d)

题目：Find the value of $\binom{-3}{4}$. (4%) **考点**：第六章广义组合数（Chapter 6.5 Generalized Combinations）——上指标为负数的二项式系数 **解析**：广义二项式系数的定义：

$$\binom{u}{k} = \frac{u(u-1)(u-2)\cdots(u-k+1)}{k!}$$

其中 u 可为任意实数，k 为非负整数。代入 $u = -3, k = 4$：

$$\binom{-3}{4} = \frac{(-3)(-4)(-5)(-6)}{4!} = \frac{360}{24} = 15$$

结果为 **15**。

1(e)

题目：Use the construction in the proof of the Chinese remainder theorem to find all solutions to the system of congruences:

$$x \equiv 1 \pmod 2,\ x \equiv 2 \pmod 3,\ x \equiv 3 \pmod 5,\ x \equiv 4 \pmod 7.$$

(5%) **考点**：第四章同余方程（Chapter 4.4 Solving Congruences）——中国剩余定理 **解析**：模数 $m_1 = 2, m_2 = 3, m_3 = 5, m_4 = 7$ 两两互素。$M = 2 \times 3 \times 5 \times 7 = 210$。对每个 i，计算 $M_i = M/m_i$ 和 y_i 满足 $M_i y_i \equiv 1 \pmod{m_i}$：

- $i = 1$：$M_1 = 210/2 = 105$。解 $105y_1 \equiv 1 \pmod 2$，即 $105 \equiv 1 \pmod 2$，所以 $y_1 = 1$。
- $i = 2$：$M_2 = 210/3 = 70$。解 $70y_2 \equiv 1 \pmod 3$。$70 \equiv 1 \pmod 3$，所以 $y_2 = 1$。
- $i = 3$：$M_3 = 210/5 = 42$。解 $42y_3 \equiv 1 \pmod 5$。$42 \equiv 2 \pmod 5$，解 $2y_3 \equiv 1 \pmod 5$，得 $y_3 = 3$（因为 $2 \times 3 = 6 \equiv 1 \pmod 5$）。
- $i = 4$：$M_4 = 210/7 = 30$。解 $30y_4 \equiv 1 \pmod 7$。$30 \equiv 2 \pmod 7$，解 $2y_4 \equiv 1 \pmod 7$，得 $y_4 = 4$（因为 $2 \times 4 = 8 \equiv 1 \pmod 7$）。

由中国剩余定理，解为：

$$x = \sum_{i=1}^{4} a_i M_i y_i \pmod M$$

其中 $a_1=1, a_2=2, a_3=3, a_4=4$。

$$\begin{aligned} x &= 1\times 105\times 1+2\times 70\times 1+3\times 42\times 3+4\times 30\times 4 \pmod{210} \\ &= 105+140+378+480 \pmod{210} \\ &= 1\,103 \pmod{210} \end{aligned}$$

$1\,103 \div 210 = 5\times 210 = 1\,050$，余 $1\,103-1\,050=53$。所以 $x\equiv 53 \pmod{210}$，即解为 $x=53+210k,\ k\in\mathbb{Z}$。验证：$53\equiv 1 \pmod 2$ $53\equiv 2 \pmod 3$ （$53=3\times 17+2$）$53\equiv 3 \pmod 5$ （$53=5\times 10+3$）$53\equiv 4 \pmod 7$ （$53=7\times 7+4$）

1(f)

题目：Find the coefficient of x^6y^4 in the expansion of $(x/3-2/x+y)^{12}$. (5%) **考点**：第六章多项式系数（Multinomial Theorem）**解析**：使用多项式定理：

$$(x/3-2/x+y)^{12} = \sum_{a+b+c=12} \frac{12!}{a!\,b!\,c!}\left(\frac{x}{3}\right)^a\left(-\frac{2}{x}\right)^b (y)^c$$

其中 $a,b,c\geq 0$ 且 $a+b+c=12$。单项式为：$x^{a-b}y^c$，系数为 $\frac{12!}{a!b!c!}\left(\frac{1}{3}\right)^a(-2)^b$。要求 x^6y^4，所以：

- y 的指数 $c=4$
- x 的指数 $a-b=6$
- $a+b+c=12 \implies a+b+4=12 \implies a+b=8$

解方程组：

$$\begin{cases} a-b=6 \\ a+b=8 \end{cases} \implies a=7, b=1$$

代入系数公式：

$$\frac{12!}{7!\,1!\,4!}\left(\frac{1}{3}\right)^7(-2)^1 = \frac{12!}{7!\,4!}\times\frac{1}{3^7}\times(-2)$$

计算 $\frac{12!}{7!\,4!} = \frac{12\times 11\times 10\times 9\times 8}{4\times 3\times 2\times 1} = \frac{95\,040}{24} = 3\,960$ 所以系数为：

$$-3\,960\times\frac{2}{3^7} = -7\,920\times\frac{1}{2\,187} = -\frac{7\,920}{2\,187} = -\frac{2\,640}{729} = -\frac{880}{243}$$

化简：$\frac{7\,920}{2\,187}$，分子分母同时除以 $3=\frac{2\,640}{729}$，再除以 $3=\frac{880}{243}$。系数为 $-\dfrac{880}{243}$。

1(g)

题目：Write the first 6 terms of the sequence determined by the generating function: $(1-x)/(1+x)$. (5%) **考点**：第八章生成函数（Chapter 8.5 Generating Functions）——从生成函数到序列 **解析**：

$$\frac{1-x}{1+x} = (1-x)\cdot\frac{1}{1+x} = (1-x)\cdot\frac{1}{1-(-x)}$$

$\frac{1}{1-(-x)} = \sum_{n=0}^{\infty}(-x)^n = \sum_{n=0}^{\infty}(-1)^n x^n = 1-x+x^2-x^3+x^4-x^5+\cdots$ 所以：

$$\frac{1-x}{1+x} = (1-x)(1-x+x^2-x^3+x^4-x^5+\cdots)$$

$$\begin{aligned} &= 1-x+x^2-x^3+x^4-x^5+\cdots \\ &\quad -x+x^2-x^3+x^4-x^5+x^6-\cdots \\ &= 1-2x+2x^2-2x^3+2x^4-2x^5+\cdots \end{aligned}$$

前 6 项：$1,-2,2,-2,2,-2$。另一种方法：$\frac{1-x}{1+x} = -1+\frac{2}{1+x} = -1+2\sum_{n=0}^{\infty}(-1)^n x^n$。结果一样。

11.13.2 题目 2 (32% total, 4% each)

题目：Ball distribution problems. **考点**：第六章排列与组合（Chapter 6.5 Generalized Permutations and Combinations）——球盒模型 **解析**：以下是所有 8 个小题的解答。**(a)** 4 个有标号的球放入 6 个有标号的盒子，无限容量的典型分配。每个球独立选择 6 个盒子之一：$6^4=1\,296$。**(b)** 4 个无标号的球放入 6 个有标号的盒子，每个盒子至多一个球。无标号球，每个盒子要么有球要么没有，选 4 个盒子放球即可：$\binom{6}{4}=15$。**(c)** 6 个有标号的球放入 6 个有标号的盒子，无空盒。每个盒子恰好一个球，即球到盒的双射：$6!=720$。**(d)** 6 个有标号的球放入 6 个有标号的盒子，恰好 3 个盒子非空。这等价于将 6 个有标号球分配到 3 个有标号盒子（选出的 3 个盒子），每个盒子至少一个球。

1. 选哪 3 个盒子非空：$\binom{6}{3}=20$ 种。
2. 将 6 个有标号球放入 3 个有标号盒子，无空盒：即满射函数数，用第二类 Stirling 数。

$$3!\times S(6,3) = 6\times 90 = 540$$

或者用容斥原理：$3^6-\binom{3}{1}2^6+\binom{3}{2}1^6 = 729-3\times 64+3 = 540$。

总数：$20\times 540 = 10\,800$。**(e)** 6 个无标号的球放入 6 个有标号的盒子，恰好 3 个盒子非空。

1. 选哪 3 个盒子非空：$\binom{6}{3}=20$ 种。
2. 将 6 个无标号球放入 3 个有标号盒子，无空盒：相当于将整数 6 拆分成 3 个正整数之和（顺序重要，因为盒子有标号），即方程 $x_1+x_2+x_3=6$ 的正整数解数。$\binom{6-1}{3-1}=\binom{5}{2}=10$。

总数：$20\times 10=200$。**(f)** 4 种颜色的球（红、绿、蓝、白），每种无限，同色球无标号。将 6 个球放入 6 个有标号盒子，要求至少有一个红球，且无空盒。先不考虑颜色限制，将 6 个无标号球放入 6 个有标号盒子（无空盒），只有 1 种方式（每个盒子恰好一个球）。再考虑颜色。6 个盒子各有 1 个球，每个球可涂 4 种颜色，要求至少 1 个红色：总着色数：4^6，不含红色：3^6。至少一个红色：$4^6-3^6=4\,096-729=3\,367$。但注意：这里的球是同种颜色无标号，但不同颜色视为不同。所以问题实际上是：每个盒子的球可以涂 4 种颜色，要求至少一个红球。总数就是 4^6-3^6。这是简化版理解。更严谨地说：6 个盒子各有一个球，每个球独立选择颜色，至少 1 个红球 = 总数 - 无红球 $=4^6-3^6=3\,367$。**(g)** 4 种球，每种无限，同种球有标号。将 6 个球放入 6 个有标号盒子，恰好用 3 种球，无空盒。”同种球有标号”意味着同种颜色的球之间可区分（例如红球 1、红球 2……）。但我们仍然是在放置球，且无空盒。恰好使用 3 种球的颜色。这其实是：6 个有标号盒子，每个盒子放 1 个球（无空盒），每个球的颜色从 4 种中选，恰好用 3 种不同颜色。

1. 选 3 种颜色：$\binom{4}{3}=4$。
2. 将 6 个有标号盒子分配到 3 种颜色，每种颜色至少一次，且同种颜色的球之间不可区分（因为同色球有标号但这里每个盒子只有一个球，所以标不标号不重要）：满射数 $3!S(6,3)=6\times 90=540$。

总数：$4\times 540=2\,160$。**(h)** 3 种球各 3 个，同种球无标号。取 6 个球，求方案数。3 种球各有 3 个，取 6 个球，同种球之间无区别。问题转化为：求非负整数解 $x_1+x_2+x_3=6$，且 $0\leq x_i\leq 3$ 的个数。先不考虑上限：$\binom{6+3-1}{3-1}=\binom{8}{2}=28$。减去违反上限的情况（某一种球取的超过 3 个，即 ≥ 4）。用容斥原理：

- 设 A_1 为 $x_1\geq 4$ 的情况，令 $y_1=x_1-4\geq 0$，则 $y_1+x_2+x_3=2$，解数为 $\binom{2+3-1}{3-1}=\binom{4}{2}=6$。同理 $|A_2|=|A_3|=6$。
- $A_1\cap A_2$ 要求 $x_1\geq 4$ 且 $x_2\geq 4$，则 $x_1+x_2\geq 8$，但总和为 6，不可能。所以 $|A_i\cap A_j|=0$。
- 同理三个同时违反也不可能。

所以有效解数：$28-3\times 6=28-18=10$。枚举验证：$(3,3,0),(3,2,1),(3,1,2),(3,0,3),(2,3,1),(2,2,2),(2,1,3),(1,3,2),(1,2,3),(0,3,3)$ 共 10 种。

11.13.3 题目 3

题目：Given 2023 cups numbered from 1 to 2023, all are placed with opening upwards. Starting from $k=2$ to 2023 each time (that is: for $k=2; k\leq 2023; k++$), we invert all cups whose number is a multiple of k. Question: In the end, how many cups are still opening upwards? (7%) **考点**：第四章数论（Number Theory）——因子个数、完全平方数 **解析**：这是一个经典的”翻杯子”问题，等价于找 1 到 2023 中每个数的因子数量（排除 1，因为从 $k=2$ 开始）。杯子 n 被翻转的次数等于 n 在 $[2,2023]$ 中的因子个数，即 n 的因子总数减 1（排除因子 1）。因子总数为 $d(n)$，则翻转次数为 $d(n)-1$。杯子最终朝上当且仅当翻转次数为偶数，即 $d(n)-1$ 为偶数 $\iff$ $d(n)$ 为奇数。我们知道 $d(n)$ 为奇数当且仅当 n 是完全平方数（因为因子成对出现，完全平方数有一对相同的因子）。所以最终朝上的杯子是编号为完全平方数的杯子，且在该范围内。在 1 到 2023 中：$\lfloor\sqrt{2023}\rfloor$。$44^2=1\,936$，$45^2=2\,025>2\,023$。所以完全平方数有 44 个（从 1^2 到 44^2）。但注意：我们是从 $k=2$ 开始翻转，即 $k=1$ 不翻转。所以：

- 对于杯子 1：因子只有 1，所以不被翻转（因为 $k\geq 2$），最终朝上。1 是完全平方数，$d(1)=1$ 为奇数，$d(1)-1=0$ 为偶数，朝上，符合。
- 一般地，完全平方数 m 有奇数个因子，减去因子 1 后的翻转次数为偶数，朝上。
- 非完全平方数有偶数个因子，减去因子 1 后的翻转次数为奇数，朝下。

所以朝上的杯子数 = 完全平方数的个数 $=\lfloor\sqrt{2023}\rfloor=44$。答案为 **44**。

11.13.4 题目 4

题目：Find the solution to the following iteration relation: (10%)

$$a_n = 4a_{n-1}-3a_{n-2}+2^n+1, \quad a_0=1,\ a_1=3$$

考点：第八章递推关系（Chapter 8.4 Nonhomogeneous Recurrence Relations）**解析**：**Step 1: 齐次通解**特征方程：$r^2-4r+3=0$，解得 $r=1,3$。齐次通解：

$$a_n^{(h)} = \alpha\cdot 1^n+\beta\cdot 3^n = \alpha+\beta\cdot 3^n$$

Step 2: 特解非齐次项为 2^n+1。

- 2^n 与齐次解无冲突（齐次解有 1^n 和 3^n，无 2^n），设特解形式 $A\cdot 2^n$。
- 常数项 1 与齐次解的常数项 α 冲突，需乘 n 修正。设形式 Bn。

设特解 $a_n^{(p)}=A\cdot 2^n+Bn$。代入递推式：

$$A\cdot 2^n+Bn = 4(A\cdot 2^{n-1}+B(n-1))-3(A\cdot 2^{n-2}+B(n-2))+2^n+1$$

分别处理：**对 2^n 项**：

$$A\cdot 2^n = 4A\cdot 2^{n-1}-3A\cdot 2^{n-2}+2^n$$

除以 2^{n-2}：

$$4A = 8A-3A+4 \implies 4A=5A+4 \implies A=-4$$

对线性项：

$$Bn = 4B(n-1) - 3B(n-2) + 1$$

$$Bn = 4Bn - 4B - 3Bn + 6B + 1$$

$$Bn = Bn + 2B + 1$$

所以 $2B + 1 = 0 \implies B = -\frac{1}{2}$。特解为：

$$a_n^{(p)} = -4 \cdot 2^n - \frac{n}{2}$$

Step 3: 通解

$$a_n = \alpha + \beta \cdot 3^n - 4 \cdot 2^n - \frac{n}{2}$$

Step 4: 代入初始条件 $n = 0$：$a_0 = \alpha + \beta - 4 = 1 \implies \alpha + \beta = 5$ $n = 1$：$a_1 = \alpha + 3\beta - 8 - \frac{1}{2} = 3 \implies \alpha + 3\beta = 11 + \frac{1}{2} = \frac{23}{2}$ 解方程组：

$$\begin{cases} \alpha + \beta = 5 \\ \alpha + 3\beta = \frac{23}{2} \end{cases}$$

相减：$2\beta = \frac{23}{2} - 5 = \frac{13}{2} \implies \beta = \frac{13}{4}$ $\alpha = 5 - \frac{13}{4} = \frac{7}{4}$ 所以：

$$a_n = \frac{7}{4} + \frac{13}{4} \cdot 3^n - 4 \cdot 2^n - \frac{n}{2}$$

化简：

$$a_n = \frac{7 + 13 \cdot 3^n - 16 \cdot 2^n - 2n}{4} = \frac{13 \cdot 3^n - 16 \cdot 2^n - 2n + 7}{4}$$

11.13.5 题目 5

题目：Let p be a prime number. Prove that there are infinite terms a_k in the sequence $1, 11, 111, 1111, \ldots, 11\ldots1$ (a_k has k '1's) such that $p \mid a_k$. (Hint: Fermat's little theorem) (10%) **考点**：第四章数论（Number Theory）——模运算、费马小定理、鸽巢原理 **解析**：序列中第 k 项为：

$$a_k = \underbrace{11\cdots1}_{k个1} = \frac{10^k - 1}{9}$$

需证：对任意素数 p，存在无穷多个 k 使得 $p \mid a_k$。**当** $p = 2$ **或** $p = 5$ **时**：

- $p = 2$：a_k 都是奇数，不可能被 2 整除。实际上题目可能有隐含条件 $p \neq 2, 5$，因为 a_k 的末位是 1。
  或者我们可以理解为 $p = 2, 5$ 时结论不成立，但题目通常针对 $p \neq 2, 5$。
  但让我们重新审题：$a_k = \frac{10^k - 1}{9}$。若 $p = 3$，有 3 和 9 的关系需小心。

更准确的表述：对素数 $p \neq 2, 3, 5$，结论成立。**对** $p \neq 2, 3, 5$：

$$p \mid a_k \iff p \mid \frac{10^k - 1}{9} \iff 10^k \equiv 1 \pmod{p} \quad (\text{由于} \gcd(p, 9) = 1)$$

由费马小定理：若 $p \nmid 10$（即 $p \neq 2, 5$），则 $10^{p-1} \equiv 1 \pmod{p}$。所以 $k = p - 1$ 是一个解，且对任意 $t \in \mathbb{N}$，$k = t(p-1)$ 也是解（因为 $10^{t(p-1)} \equiv (10^{p-1})^t \equiv 1 \pmod{p}$）。故有无穷多个 k（所有 $p - 1$ 的倍数）使得 $p \mid a_k$。**对** $p = 3$：$a_k = \underbrace{11\cdots1}_{k个1}$，其各位数字和为 k。一个数被 3 整除当且仅当各位数字和能被 3 整除。所以 k 是 3 的倍数时，$3 \mid a_k$。有无穷多个。**对** $p = 2$ **或** $p = 5$：a_k 的末位为 1，不可能被 2 或 5 整除。所以原命题对 $p = 2, 5$ 不成立。但题目中 p 为素数通常默认排除了 2 和 5（因为序列全部是奇数且不以 0 或 5 结尾）。若强行要求，则可表述为" 对素数 $p \neq 2, 5$"。**证毕**。

11.13.6 题目 6

题目：Given any positive integer m, prove that a multiple of m can be found in the Fibonacci sequence. (10%) **考点**：第五章/第八章数学归纳法、鸽巢原理 **解析**：Fibonacci 数列定义为 $F_0 = 0, F_1 = 1$，且 $F_n = F_{n-1} + F_{n-2}$ 对 $n \geq 2$。需证：对任意正整数 m，存在 n 使得 $m \mid F_n$。**方法：鸽巢原理**考虑 Fibonacci 数列模 m 后的余数对：

$$(F_0 \bmod m, F_1 \bmod m),\ (F_1 \bmod m, F_2 \bmod m),\ \ldots,\ (F_k \bmod m, F_{k+1} \bmod m),\ \ldots$$

每个余数在 $[0, m-1]$ 中取值，所以有序对共有 m^2 种可能。考虑前 $m^2 + 1$ 个余数对（从 (F_0, F_1) 到 (F_{m^2}, F_{m^2+1})）。由鸽巢原理，至少有两个相同的余数对。设 $(F_i \bmod m, F_{i+1} \bmod m) = (F_j \bmod m, F_{j+1} \bmod m)$，其中 $i < j$。由于 Fibonacci 数列具有**确定性**和**可逆性**：

- 由 $F_i \equiv F_j \pmod{m}$ 和 $F_{i+1} \equiv F_{j+1} \pmod{m}$，可推得 $F_{i-1} \equiv F_{j-1} \pmod{m}$。
- 往前递推得 $F_0 \equiv F_{j-i} \pmod{m}$。
- 而 $F_0 = 0$，所以 $F_{j-i} \equiv 0 \pmod{m}$，即 $m \mid F_{j-i}$。

且 $j - i > 0$，故找到了 Fibonacci 数列中被 m 整除的项。**另一种更简单的证明**：Fibonacci 数列模 m 是周期数列（因为状态有限，由鸽巢原理必然有重复，从而进入循环）。只要证明 $(0, 1)$ 这个状态会出现即可。但因为 $F_0 = 0, F_1 = 1$ 是初始状态，如果周期包含它，那么一定有某个 $F_k \equiv 0 \pmod{m}$。实际上，由上述证明，我们找到了 $(F_0, F_1) = (0, 1)$ 会在后续重现（的平移），所以有一个 Fibonacci 数模 m 余 0。**证毕**。

11.14 DMA Quiz 2 题解—2024 年

11.14.1 说明

2024 年（23 级）离散数学 Quiz 2 的试卷内容在当前资料库中不可用。在 TuringCourses 网站上，2024 年（23 级）仅收录了第一次小测、第三次小测和第四次小测，**并没有第二次小测（Quiz 2）**。本地文件夹中的 `Discrete_Mathematics_Quiz_2_2024.md` 和 `Discrete_Mathematics_Quiz_2_2` 均为同名文件的 HTML 页面存档，不包含实际试卷内容。建议参考其他年份的 Quiz 2 试卷（2022 年、2023 年）进行复习。

11.14.2 历年 Quiz 2 考点总结（供参考）

根据 2022 年和 2023 年的试卷，Quiz 2 通常覆盖以下章节：

第六章计数（Counting）

- 排列与组合的字典序生成（下一个排列/组合）
- 二项式系数、多项式系数
- 广义组合数（负上指标）
- 球盒模型（有/无标号、是否可重复、是否允许空盒）
- 容斥原理

第四章数论（Number Theory）

- 模幂运算、费马小定理
- 中国剩余定理
- 整除性质

第八章递推与生成函数（Recurrence & Generating Functions）

- 线性递推关系的求解（齐次 + 非齐次）
- 生成函数到序列的转换
- 鸽巢原理的应用

第九章关系（Relations）

- 关系的性质计数（自反、对称、传递、等价、偏序等）

11.15 DMA Quiz 3 题解—2022 年

11.15.1 题目 1

题目：Find the transitive closure of R on $\{a, b, c, d\}$, where

$$R = \{(a, a), (b, a), (b, c), (c, a), (c, c), (c, d), (d, a), (d, c)\}.$$

(6%) **考点**：第九章关系（Chapter 9.4 Closures of Relations）——传递闭包、Warshall 算法 **解析**：**方法一：直接计算传递闭包**传递闭包 $R^* = R \cup R^2 \cup R^3 \cup \cdots$（因为集合大小有限，最多到 R^{n-1}）。写出关系矩阵 M_R，其中行为起点，列为终点（顺序 a, b, c, d）：

$$M_R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

计算 $M_{R^2} = M_R \odot M_R$（布尔矩阵乘法：$\vee$ 代替 $+$，$\wedge$ 代替 $\times$）：$(M_R^2)_{ij} = \bigvee_{k=1}^{4} (M_R)_{ik} \wedge (M_R)_{kj}$ 逐项计算：

- 第 1 行：$(a, a) \wedge (a, a) = 1$，$(a, a) \wedge (a, b) = 0$，$(a, a) \wedge (a, c) = 0$，$(a, a) \wedge (a, d) = 0$ 得 $(1, 0, 0, 0)$
- 第 2 行 (b)：经 a: $1 \to (1, 0, 0, 0)$；经 c: $1 \to (1, 0, 1, 1)$ 取并：$(1, 0, 1, 1)$
- 第 3 行 (c)：经 a: $1 \to (1, 0, 0, 0)$；经 c: $1 \to (1, 0, 1, 1)$ 取并：$(1, 0, 1, 1)$
- 第 4 行 (d)：经 a: $1 \to (1, 0, 0, 0)$；经 c: $1 \to (1, 0, 1, 1)$ 取并：$(1, 0, 1, 1)$

$$M_{R^2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

继续计算 $M_{R^3} = M_{R^2} \odot M_R$：

- 第 1 行 (a): $1 \to (1, 0, 0, 0) \to (1, 0, 0, 0)$
- 第 2 4 行 (b,c,d): 已有到 a,c,d 的路径，经 a 得 $(1, 0, 0, 0)$，经 c 得 $(1, 0, 1, 1) \to (1, 0, 1, 1)$

$M_{R^3} = M_{R^2}$，说明已经收敛。所以 $M_{R^*} = M_R \vee M_{R^2}$：

$$M_{R^*} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

传递闭包：

$$R^* = \{(a,a),(b,a),(b,c),(b,d),(c,a),(c,c),(c,d),(d,a),(d,c),(d,d)\}$$

注意：(b,b) 和 (c,b) 不存在（没有到 b 的边），(d,b) 也不存在。**方法二：Warshall 算法** Warshall 算法逐轮引入中间节点：初始矩阵 $W_0 = M_R$：

$$W_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

第 1 轮（以 a 为中间节点，k=1）：检查哪些行有到 a 的路径，将那些行的第 1 行加到该行：

- 有到 a 的行：第 2,3,4 行
- 矩阵的第 1 行 $= (1,0,0,0)$，加到第 2,3,4 行不变（已含 $(1,0,0,0)$）

$$W_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

第 2 轮（以 b 为中间节点，k=2）：没有行有到 b 的路径（第 2 列全 0），所以无变化。

$$W_2 = W_1$$

第 3 轮（以 c 为中间节点，k=3）：有到 c 的行：第 2 行（b→c）、第 3 行（c→c）、第 4 行（d→c）。将第 3 行（c 的行）$(1,0,1,1)$ 加到这些行：第 2 行原为 $(1,0,1,0)$ 或 $(1,0,1,0) \to$ 加上 $(1,0,1,1)$ 得 $(1,0,1,1)$。第 3 行原为 $(1,0,1,1)$，不变。第 4 行原为 $(1,0,1,0) \to$ 加上 $(1,0,1,1)$ 得 $(1,0,1,1)$。

$$W_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

第 4 轮（以 d 为中间节点，k=4）：这一轮可能增加路径。有到 d 的行：第 2,3,4 行。将第 4 行加到这些行：第 4 行 $= (1,0,1,1)$，已有的第 2,3 行也是 $(1,0,1,1)$，不变。

$$W_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

结论与前面一致。

11.15.2 题目 2

题目：(a) Find the smallest partial order relation R on $\{a,b,c,d,e,f\}$ that contains $(a,c),(c,c),(c,b),(c,d),(b,e),(b,f)$. (b) Draw the Hasse diagram of R. (c) List the maximal elements. (d) List the minimal elements. (e) Find the greatest element. (f) Find the least element. (g) Find the least upper bound of $\{d,e\}$. (h) Use topological sorting to order the elements of the poset. (24%) **考点**：第九章偏序关系（Chapter 9.6 Partial Orderings）——偏序闭包、Hasse 图、拓扑排序 **解析**：**(a)** 偏序关系要求自反、反对称、传递。给定集合

$$S = \{(a,c),(c,c),(c,b),(c,d),(b,e),(b,f)\}$$

需要添加最少的有序对使其成为偏序。**Step 1: 自反闭包**添加所有 (x,x) 对每个 $x \in \{a,b,c,d,e,f\}$：$R_1 = S \cup \{(a,a),(b,b),(d,d),(e,e),(f,f)\}$（$(c,c)$ 已存在）**Step 2: 传递闭包**从已知关系出发，添加传递性所需的有序对：

- (a,c) 且 $(c,b) \to$ 添加 (a,b)
- (a,c) 且 $(c,d) \to$ 添加 (a,d)
- (a,c) 且 $(c,c) \to$ 已有 (a,c)
- (c,b) 且 $(b,e) \to$ 添加 (c,e)
- (c,b) 且 $(b,f) \to$ 添加 (c,f)
- (a,b) 且 $(b,e) \to$ 添加 (a,e)（通过 $a \to c \to b \to e$）
- (a,b) 且 $(b,f) \to$ 添加 (a,f)（通过 $a \to c \to b \to f$）
- (a,c) 且 $(c,e) \to$ 已有 (a,e)
- (a,c) 且 $(c,f) \to$ 已有 (a,f)

另外还要检查通过更多步的传递：

- $a \to c \to b \to e$：(a,e) 已加
- $a \to c \to b \to f$：(a,f) 已加
- $a \to c \to d$：(a,d) 已加
- $c \to b \to e$：(c,e) 已加
- $c \to b \to f$：(c,f) 已加

Step 3: 检查反对称现有关系统没有同时包含 (x,y) 和 (y,x) 对 $x \neq y$ 的情况，所以反对称自动满足。最小偏序关系：

$$R = \{(a,a),(b,b),(c,c),(d,d),(e,e),(f,f),(a,c),(c,b),(c,d),(b,e),(b,f),(a,b),(a,d),(c,e),(c,f),(a,e),(a,f)\}$$

(b) Hasse 图（去掉自反边和传递边，箭头向上或向下）：关系链：

- $a \to c \to b \to e$
- $a \to c \to b \to f$
- $a \to c \to d$

Hasse 图：

```
e     f     d
 \   /      |
  \ /       |
   b        |
   |        |
   c        |
   |        |
   a
```

更清晰的层次（从下到上）：

- 底层：a
- 第二层：c
- 第三层：b 和 d
- 顶层：e 和 f

覆盖关系（Hasse 图中的边）：

- $a \prec c$（无中间元素）
- $c \prec b$（无中间元素）
- $c \prec d$（无中间元素）
- $b \prec e$（无中间元素）
- $b \prec f$（无中间元素）

注意 $a \prec b$ 不是覆盖关系，因为 $a \prec c \prec b$。**(c)** 极大元（没有比它更大的元素）：d,e,f **(d)** 极小元（没有比它更小的元素）：a **(e)** 最大元（大于等于所有元素）：不存在，因为 d,e,f 之间没有可比性，没有一个元素同时大于它们三个。**(f)** 最小元（小于等于所有元素）：不存在，因为 a 与 d 不相比较（$a \leq d$ 成立吗？我们有 $a \to c \to d$，即 $a \preceq d$，成立！实际上 a 小于等于所有元素，因为从 a 出发可到达所有其他节点。所以 a 是**最小元**。检查：$a \preceq a$（自反），$a \preceq c$（$a \to c$），$a \preceq b$（$a \to c \to b$），$a \preceq d$（$a \to c \to d$），$a \preceq e$（$a \to c \to b \to e$），$a \preceq f$（$a \to c \to b \to f$）。所以 a 确实是最小元。**(g)** $\{d,e\}$ 的最小上界（LUB）：

- d 的上界：比 d 大或等于 d 的元素。由于 d 是极大元，只有 d 自己是上界。
- e 的上界：比 e 大或等于 e 的元素。e 也是极大元，只有 e 自己是上界。
- $\{d,e\}$ 的共同上界：存在一个元素同时大于等于 d 和 e 吗？没有，因为 d 和 e 是不可比的，且没有元素同时大于它们两个。

所以 $\{d,e\}$ 没有上界，也就**没有最小上界**。**(h)** 拓扑排序（偏序的线性拓展）：需要满足偏序关系，即若 $x \preceq y$，则 x 在 y 之前。偏序关系：$a \to c \to b \to e$，$a \to c \to b \to f$，$a \to c \to d$ 一个可能的拓扑排序顺序：a,c,b,d,e,f 另一个：a,c,d,b,e,f 另一个：a,c,b,e,d,f 只要满足 a 在 c 前，c 在 b 前，c 在 d 前，b 在 e 前，b 在 f 前即可。

11.15.3 题目 3

题目：Find a minimum spanning tree for the weighted graph in Fig.1. You can just draw out the answer. (6%) **考点**：第十一章树（Chapter 11.5 Minimum Spanning Trees）——最小生成树 **解析**：Fig.1 的图包含节点 0-8，权重如下：

- 0-1: 4, 0-7: 8
- 1-2: 8, 1-7: 11
- 2-3: 7, 2-5: 4, 2-8: 2
- 3-4: 9, 3-5: 14
- 4-5: 10
- 5-6: 2
- 6-7: 1, 6-8: 6
- 7-8: 7

使用 **Kruskal 算法**（按边权从小到大选择，不形成环）：排序边（升序）：

1. 6-7: 1
2. 2-8: 2
3. 5-6: 2
4. 0-1: 4
5. 2-5: 4

6. 2-3: 7
7. 7-8: 7 → 会形成环（6-7-8-2-5-6），跳过
8. 0-7: 8 → 会形成环（0-1-?），实际检查连通性后在 0-7 加入？需要检查。

按算法逐步：

1. 6-7 (1)：选择
2. 2-8 (2)：选择
3. 5-6 (2)：选择
4. 0-1 (4)：选择
5. 2-5 (4)：选择，此时 2,5,6,7,8 连通。
6. 2-3 (7)：选择，连通 3。
7. 7-8 (7)：跳过（2-5-6-7-8-2 形成环）
8. 0-7 (8)：选择，连通 0-1 分组与 2-5-6-7-8-3 分组。
9. 1-2 (8)：跳过（连通 0-1-7-6-5-2-8-3 形成环）
10. 3-4 (9)：选择，连通 4。
11. 4-5 (10)：跳过，4 已连通。
12. 1-7 (11)：跳过
13. 3-5 (14)：跳过
14. （剩余边都会被跳过）

选择的边：6-7 (1), 2-8 (2), 5-6 (2), 0-1 (4), 2-5 (4), 2-3 (7), 0-7 (8), 3-4 (9) MST 总权重：$1+2+2+4+4+7+8+9=37$ MST 边集合：$\{(6,7),(2,8),(5,6),(0,1),(2,5),(2,3),(0,7),(3,4)\}$ **另一种 MST（使用 Prim 算法）**：Prim 从节点 0 出发，逐次添加最小权重的边连接到已选集合。可能会得到不同的 MST 但总权重相同。

11.15.4 题目 4

题目：Use Dijkstra's Algorithm to find the shortest path length between the vertices 1 and 6 in the weighted graph in Fig.2. (10%) **考点**：第十章最短路径（Chapter 10.6 Shortest Path Problems）——Dijkstra 算法 **解析**：Fig.2 的有向图（带权）：

- 1→2: 1, 1→3: 12
- 2→3: 9, 2→4: 3
- 3→5: 5
- 4→3: 4, 4→5: 13, 4→6: 15
- 5→6: 4

运行 Dijkstra 算法求从 1 到 6 的最短路径：

轮次	当前节点	dist[1]	dist[2]	dist[3]	dist[4]	dist[5]	
初始	-	0	∞	∞	∞	∞	
1	1	0	**1** (1→2)	12 (1→3)	∞	∞	
2	2	0	1	min(12, 1+9=10) = **10** (1→2→3)	1+3=4	∞	
3	4	0	1	min(10, 4+4=8) = **8** (1→2→4→3)	4	min(∞, 4+13=17)=17	
4	3	0	1	8	4	min(17, 8+5=13) = **13** (1→2→4→3→5)	mi
5	5	0	1	8	4	13	min(19,

最短路径长度为 **17**。最短路径：1→2→4→3→5→6，总长 $=1+3+4+5+4=17$。另一种可能路径：1→2→4→6 $=1+3+15=19$，不是最短。

11.15.5 题目 5

题目：Use Huffman coding to encode these symbols with given frequencies: a: 0.15, b: 0.22, c: 0.26, d: 0.19, e: 0.08, f: 0.1. What is the average number of bits required to encode a character? (8%) **考点**：第十一章树（Chapter 11.2 Applications of Trees）——Huffman 编码 **解析**：Huffman 编码构建过程（每次合并两个最小的频率）：

1. 初始频率：a(0.15), b(0.22), c(0.26), d(0.19), e(0.08), f(0.10)
2. 排序：e(0.08), f(0.10), a(0.15), d(0.19), b(0.22), c(0.26)
3. 合并 e(0.08) 和 f(0.10) → 节点 ef(0.18)
4. 排序：a(0.15), ef(0.18), d(0.19), b(0.22), c(0.26)
5. 合并 a(0.15) 和 ef(0.18) → aef(0.33)
6. 排序：d(0.19), b(0.22), aef(0.33), c(0.26)
7. 合并 d(0.19) 和 b(0.22) → bd(0.41)
8. 排序：c(0.26), aef(0.33), bd(0.41)
9. 合并 c(0.26) 和 aef(0.33) → caef(0.59)
10. 排序：bd(0.41), caef(0.59)
11. 合并 bd(0.41) 和 caef(0.59) → root(1.00)

构建的 Huffman 树（分配码字：左 0 右 1）：

```
                 root
               /      \
          bd(0.41)   caef(0.59)
         /    \      /       \
    d(0.19) b(0.22) c(0.26) aef(0.33)
                              /     \
                         a(0.15)  ef(0.18)
                                  /     \
                              e(0.08) f(0.10)
```

码字长度：

- d: 2 位
- b: 2 位
- c: 2 位
- a: 3 位
- e: 4 位
- f: 4 位

平均码长：

$$0.15\times3+0.22\times2+0.26\times2+0.19\times2+0.08\times4+0.10\times4$$

$$=0.45+0.44+0.52+0.38+0.32+0.40=2.51$$

平均需要 **2.51** 位/字符。

11.15.6 题目 6

题目：Determine all positive integers r and s for which $K_{r,s}$ is planar. Explain your answer. (8%) **考点**：第十章平面图（Chapter 10.7 Planar Graphs）——完全二分图的平面性 **解析**：$K_{r,s}$ 是完全二分图，两部分分别有 r 和 s 个顶点。**结论**：$K_{r,s}$ 是平面图当且仅当 $r\le2$ 或 $s\le2$。**证明**：**(充分性)**：

- $K_{1,s}$ 是星形图，显然是平面图（一个中心连接 s 个叶子）。
- $K_{2,s}$ 也是平面图：画法为将两个节点放在中间，s 个节点围绕它们排列，每条边从一侧到另一侧可以不交叉。具体构造：将两个节点放在内圈（或同一侧），s 个节点并列排列，可画出无交叉的二分图。

(必要性)：若 $r\ge3$ 且 $s\ge3$，则 $K_{r,s}$ 包含 $K_{3,3}$ 作为子图。而 $K_{3,3}$ 是已知的非平面图（由 Kuratowski 定理或直接使用 Euler 公式证明）。包含非平面子图的图必然非平面。故 $K_{r,s}$ 非平面。**Euler 公式证明 $K_{3,3}$ 非平面**：假设 $K_{3,3}$ 可平面嵌入，有 $n=6$ 个顶点，$m=9$ 条边。因为 $K_{3,3}$ 是二分图，无奇环，所以每个面的边界长度至少为 4。由 Euler 公式 $n-m+f=2$，得 $f=2-6+9=5$。面度数和 $\sum\deg(f)=2m=18$。但每个面至少 4 条边，所以面度数和 $\ge4f=20$。矛盾（$18<20$）。故 $K_{3,3}$ 非平面。Kuratowski 定理也直接指出：$K_{3,3}$ 和 K_5 是极小非平面图。因此 $K_{r,s}$ 平面当且仅当 $r\le2$ 或 $s\le2$。

11.15.7 题目 7

题目：In a round-robin tournament every player plays every other player exactly once and each match has a winner and a loser. There are total n players. Prove that we can sort the players in a certain order $p_1,p_2,\ldots,p_n$, so that p_1 beats p_2, p_2 beats p_3, …, and p_{n-1} beats p_n. (10%) **考点**：第十章图论（Graph Theory）——竞赛图、哈密顿路径 **解析**：**建模**：将 n 个选手看作 n 个顶点，若选手 i 击败选手 j，则画一条从 i 到 j 的有向边。这样得到的图是**竞赛图**（tournament）：每对顶点之间恰有一条有向边。原命题等价于：**任意竞赛图都有哈密顿路径**。用数学归纳法证明。**归纳基础**：$n=1$ 时，只有一个选手，平凡路径成立。$n=2$ 时，两个选手之间有一条边，该边即构成哈密顿路径。成立。**归纳假设**：假设对 $n=k$（$k\ge1$）的任意竞赛图都存在哈密顿路径。**归纳步骤**：考虑 $n=k+1$ 个选手的竞赛图。去掉选手 v（及与之相连的所有边），剩下 k 个选手的竞赛图。由归纳假设，存在一条哈密顿路径 $p_1\to p_2\to\cdots\to p_k$。现在考虑将 v 插入到路径中。

- 若 v 击败了 p_1，则可放在开头：$v\to p_1\to p_2\to\cdots\to p_k$，形成新哈密顿路径。
- 若 p_k 击败了 v，则可放在末尾：$p_1\to p_2\to\cdots\to p_k\to v$，形成新哈密顿路径。
- 否则，p_1 击败了 v 且 v 击败了 p_k（因为竞赛图每对顶点间必有一条边）。此时，存在 i 使得 p_i 击败 v 且 v 击败 p_{i+1}。
- 因为 $p_1\to v$（p_1 击败 v），且 $v\to p_k$（v 击败 p_k）。
- 从 $i=1$ 开始，如果 $v\to p_{i+1}$，则检查 $p_i\to v$ 是否成立。
- 随着 i 从 1 增加到 k-1，由于 $p_1\to v$ 且 $v\to p_k$，一定存在某个 i 使得 $p_i\to v$ 且 $v\to p_{i+1}$（否则如果对所有 i 要么 $v\to p_i$ 要么 $p_{i+1}\to v$，会产生矛盾）。

  找到这样的 i 后，将 v 插入在 p_i 和 p_{i+1} 之间：

$$p_1\to\cdots\to p_i\to v\to p_{i+1}\to\cdots\to p_k$$

  这得到了新的哈密顿路径。

由数学归纳法，对所有 $n \geq 1$，任意竞赛图都有哈密顿路径，即原命题成立。**证毕**。

11.15.8 题目 8

题目：Fig.3 is the Petersen graph. (28%) (a) Find the chromatic number of the Petersen graph. (b) Determine whether the graph in Fig.4 is also a Petersen graph. (c) Prove that the Petersen Graph is non-planar using Euler's formula. (d) Determine whether the Petersen graph is Hamilton graph. Prove or disprove it. **考点**：第十章图论（Graph Theory）——图染色、图同构、平面图、哈密顿图 **解析**：**(a)** Petersen 图的色数。Petersen 图是 3-正则图（每个顶点的度为 3），且包含奇环（长度为 5），所以色数至少为 3。构造 3-着色是可行的（Petersen 图是 3-可着色的，实际上它的色数为 3）。因此，**色数为 3**。一种 3-着色：外圈 5 个顶点交替涂 2 种颜色（需 3 种因为 5 是奇数，比如 1,2,1,2,3），内圈 5 个顶点对应使用适当的第 3 种颜色。具体地，可以对内外各 5 个顶点分配三种颜色。**(b)** Fig.4 是否也是 Petersen 图。Fig.4 描述：一个顶点 0 位于中心，连接到外圈的 1, 4, 7；外圈有 9 个顶点 1-9 按圆形排列，相邻顶点之间有边：1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 7-8, 8-9, 9-1；0 还与 1, 4, 7 相连；另外还有 3-8, 2-6, 9-5。标准的 Petersen 图有 10 个顶点，每个顶点度为 3，不含三角形和 4 环。Fig.4 的图中：顶点 1-9 在外圈（9 个顶点）+ 中心顶点 0 = 10 个顶点。度为 3 的顶点？中心 0 连接 1,4,7 三个顶点，度 = 3。外圈顶点：每个外圈顶点与相邻两个外圈顶点相连，且有些还与 0 或与其他外圈顶点相连。具体看：

- 1 连接 0, 2, 9 → 度 3
- 2 连接 1, 3, 6 → 度 3
- 3 连接 2, 4, 8 → 度 3
- 4 连接 3, 5, 0 → 度 3
- 5 连接 4, 6, 9 → 度 3
- 6 连接 5, 7, 2 → 度 3
- 7 连接 6, 8, 0 → 度 3
- 8 连接 7, 9, 3 → 度 3
- 9 连接 8, 1, 5 → 度 3

每个顶点度都是 3，且没有三角形（最短环为 5）。这个图是 Petersen 图的另一种画法，**是同构的**。**验证同构**：存在一个双射使得边结构相同。标准 Petersen 图的顶点可以标注为 (i, j)（$i \in \{0, 1\}, j \in \{0, 1, 2, 3, 4\}$），其中 $(0, j)$ 连接 $(1, j), (1, j+1), (1, j+2)$（模 5）。Fig.4 的标注与标准标注可以建立一一对应。因此，**Fig.4 也是 Petersen 图**。**(c)** 用 Euler 公式证明 Petersen 图非平面。Petersen 图有 $n = 10$ 个顶点，$m = 15$ 条边。假设它是平面图，则由 Euler 公式 $n - m + f = 2$，面数 $f = 2 - 10 + 15 = 7$。Petersen 图不含三角形（最小环长度为 5），因此每个面的边界至少包含 5 条边。面度数和 $= \sum \deg(f) = 2m = 30$（每条边被两个面共享）。但每个面至少 5 条边，所以面度数和 $\geq 5f = 5 \times 7 = 35$。矛盾（$30 < 35$）！因此 Petersen 图**不是平面图**。**(d)** Petersen 图是否是哈密顿图。**结论**：**Petersen 图不是哈密顿图**（不包含哈密顿回路）。**证明**：假设 Petersen 图有哈密顿回路 C（长度为 10 的环）。Petersen 图是 3-正则图（每个顶点度 = 3）。在哈密顿回路中，每个顶点用了两条边（回路中的两条邻边），还剩一条边（不在回路中的边）。去掉回路 C 上的所有边，剩下 5 条边（因为 $15 - 10 = 5$，每条边对应一个顶点上未被回路使用的边）。这 5 条边构成一个完美匹配（每个顶点恰好关联一条剩余边）。现在 C 是一个 10 环。在这 10 环上添加 5 条完美匹配边。由于 Petersen 图不含 3 环和 4 环，且 10 环的弦只能在相隔至少 3 个顶点的位置添加（否则会产生短环）。可以论证：无论怎么添加这 5 条匹配边，都无法得到 Petersen 图（即会导出短环或度不一致等矛盾）。构造性证明比较复杂，但标准结论是 Petersen 图没有哈密顿回路。Petersen 图**有哈密顿路径**（实际上去掉任何一个顶点后剩余的子图有哈密顿回路，但整体没有哈密顿回路）。因此，Petersen 图**不是哈密顿图**（没有哈密顿回路），但有哈密顿路径。**证毕**。

11.16 DMA Quiz 3 题解—2024 年

11.16.1 题目 1

题目：$R = \{(a, a), (a, b), (b, d), (a, d)\}$ is a relation on $\{a, b, c, d\}$. Find the smallest relation containing the relation R that is: (a) (6%) partial order relation. (b) (6%) symmetric and transitive. **考点**：第九章关系（Chapter 9.4 & 9.6）——偏序关系、自反对称传递闭包 **解析**：**(a)** 偏序关系要求自反、反对称、传递。给定 R，需添加最少的有序对满足这三条。**Step 1: 自反闭包**添加所有 (x, x)：

$$R_1 = R \cup \{(b, b), (c, c)\} = \{(a, a), (b, b), (c, c), (a, b), (b, d), (a, d)\}$$

Step 2: 传递闭包

- (a, b) 且 $(b, d) \to$ 添加 (a, d)（已存在）
- 没有其他传递路径需要添加。

检查新对：$(a, b), (b, d), (a, d)$ 已覆盖。**Step 3: 检查反对称**没有同时包含 (x, y) 和 (y, x) 对 $x \neq y$ 的情况，满足反对称。因此最小偏序关系为：

$$R' = \{(a, a), (b, b), (c, c), (a, b), (b, d), (a, d)\}$$

其中 c 是孤立点（只与自身相关）。注意偏序关系不需要是连通的，所以 c 孤立没问题。**(b)** 满足对称性和传递性的最小关系。给定的 R 需要包含在关系中，关系需要对称且传递。**几何方法**：对称传递闭包 = 传递闭包 + 对称闭包，反复进行直到稳定。**代数方法**：对称且传递的关系等价于"等价关系"但不要求自反（实际上对称 + 传递可推出对任何存在于关系中的元素有自反性）。先取对称闭包：添加所有反向边。从 R 添加 $(b, a), (d, b), (d, a)$：

$$R_1 = R \cup \{(b, a), (d, b), (d, a)\} = \{(a, a), (a, b), (a, d), (b, a), (b, d), (d, a), (d, b)\}$$

再取传递闭包：现有关系构成一个团：a, b, d 两两相连，且每个自反。检查传递性：

- (a, b) 且 $(b, a) \to$ 已有 (a, a)
- (a, b) 且 $(b, d) \to$ 已有 (a, d)
- (b, a) 且 $(a, d) \to$ 需加 (b, d)
- (d, b) 且 $(b, a) \to$ 需加 (d, a)
- (d, a) 且 $(a, b) \to$ 需加 (d, b)
- 其他类似组合都已满足。

所以对称且传递的最小关系为：

$$R'' = \{(a, a), (b, b), (d, d), (a, b), (b, a), (a, d), (d, a), (b, d), (d, b)\}$$

注意：c 不在任何对中（除了它自己可能被要求）。但对称 + 传递不要求自反（除非元素出现在关系中）。这里 c 不在任何对中，所以不需要添加。如果要求最小等价关系，还需加 (c, c)。但题目只要求对称和传递，没有要求自反，所以不需要。

11.16.2 题目 2

题目：Given the undirected graph G as shown in Fig.1. (a) (6%) Use Kruskal's algorithm to find the minimum spanning tree of graph G. What is the order in which the edges are added to the minimum spanning tree? (b) (6%) Using alphabetical ordering, find a spanning tree for this graph by depth-first search. **考点**：第十一章树（Chapter 11.4 & 11.5）——Kruskal 算法、DFS 生成树 **解析**：Fig.1 的节点：a, b, c, d, e, f。边权重：

- a-b: 20, a-c: 12, a-e: 9
- b-e: 11, b-d: 6, b-f: 5
- c-e: 10, c-d: 18
- d-e: 14, d-f: 7

(a) Kruskal 算法（按边权升序选择，不形成环）：排序边：

1. b-f: 5
2. b-d: 6
3. a-e: 9
4. c-e: 10
5. b-e: 11 → 会形成环（a-e-b-f 不形成环，但 a-e-b 再加 b-f: a-e-b 已连通？检查：目前连通分量：a,e,c, b,f,d。b-e 连接两个分量，所以选！）等等让我重新仔细检查。

实际上：

1. b-f (5) → 连通分量：b,f
2. b-d (6) → 连通分量：b,f,d
3. a-e (9) → 连通分量：a,e
4. c-e (10) → 连通分量：a,e,c
5. b-e (11) → 连接分量 a,e,c 和 b,f,d，选择连通分量：a,b,c,d,e,f 全部连通，MST 完成。

选择的边（添加顺序）：b-f (5), b-d (6), a-e (9), c-e (10), b-e (11) 或另一种可能的顺序（如果有相同权重的边，不同实现可能顺序不同）。MST 权重 $= 5 + 6 + 9 + 10 + 11 = 41$。**另一种可能的顺序**：当然也可以选 a-e (9) 在 c-e (10) 之前这是一定的。但 10 和 11 以及后面的边顺序固定。**(b) DFS 生成树**（按字母顺序，从 a 开始）DFS 从 a 开始，按字母顺序选择邻接顶点：

- 从 a 出发，邻接：b(20), c(12), e(9)。按字母顺序：先 b。
- a → b，邻接：a(20), d(6), e(11), f(5)。除 a 外字母序最小的：d。
- b → d，邻接：b(6), c(18), e(14), f(7)。除 b 外：c。
- d → c，邻接：a(12), d(18), e(10)。除 d 外：a(已访问)，e。
- c → e，邻接：a(9), b(11), c(10), d(14)。除 c 外：a, b 都已访问，唯一未访问的是无。但 e 有到 f 吗？e 不直接连接 f。回溯：从 e 返回 c，从 c 返回 d，从 d 返回 b。
- 在 b：还有 f 未访问。
- b → f。

DFS 生成树的边（按访问顺序）：a-b, b-d, d-c, c-e, b-f 这 5 条边构成了生成树（所有 6 个节点连通，5 条边）。

11.16.3 题目 3

题目：The frequencies of six characters are 0.09, 0.05, 0.2, 0.25, 0.3 and 0.11, respectively. If Huffman coding is used for optimal encoding, the average number of bits required to encode a character is _____. (6%) **考点**：第十一章树（Chapter 11.2 Applications of Trees）——Huffman 编码 **解析**：符号频率：设 $f_1 = 0.09, f_2 = 0.05, f_3 = 0.2, f_4 = 0.25, f_5 = 0.3, f_6 = 0.11$。构建 Huffman 树的过程：

1. 排序：0.05, 0.09, 0.11, 0.2, 0.25, 0.3
2. 合并 0.05 和 0.09 → 0.14
3. 排序：0.11, 0.14, 0.2, 0.25, 0.3
4. 合并 0.11 和 0.14 → 0.25
5. 排序：0.2, 0.25, 0.25, 0.3

6. 合并 0.2 和 0.25(第一个) → 0.45
7. 排序：0.25(第二个), 0.3, 0.45
8. 合并 0.25 和 0.3 → 0.55
9. 合并 0.45 和 0.55 → 1.0

Huffman 树结构：

```
                root
              /       \
          0.45         0.55
         /   \        /    \
    0.2    0.25    0.25    0.3
           /  \    /  \
        0.09  0.14  f?   f?
              /  \
           0.05  0.09
```

等一下，让我重新仔细构建，注意相同的值需要区分。频率：a=0.09, b=0.05, c=0.2, d=0.25, e=0.3, f=0.11

1. 排序：b(0.05), a(0.09), f(0.11), c(0.2), d(0.25), e(0.3)
2. 合并 b(0.05) + a(0.09) = n1(0.14)
3. 排序：f(0.11), n1(0.14), c(0.2), d(0.25), e(0.3)
4. 合并 f(0.11) + n1(0.14) = n2(0.25)
5. 排序：c(0.2), n2(0.25), d(0.25), e(0.3) （有两个 0.25）
6. 合并 c(0.2) + n2(0.25) = n3(0.45)
7. 排序：d(0.25), e(0.3), n3(0.45)
8. 合并 d(0.25) + e(0.3) = n4(0.55)
9. 合并 n3(0.45) + n4(0.55) = root(1.0)

树结构（左 0 右 1）：

```
                root
              /       \
          n3(0.45)      n4(0.55)
          /      \      /      \
    c(0.2)  n2(0.25) d(0.25) e(0.3)
            /     \
        f(0.11) n1(0.14)
                /     \
            b(0.05) a(0.09)
```

码字：

- c: 00 → 2 位
- f: 010 → 3 位
- b: 0110 → 4 位
- a: 0111 → 4 位
- d: 10 → 2 位
- e: 11 → 2 位

平均码长：

$$0.2 \times 2 + 0.11 \times 3 + 0.05 \times 4 + 0.09 \times 4 + 0.25 \times 2 + 0.3 \times 2$$

$$= 0.4 + 0.33 + 0.2 + 0.36 + 0.5 + 0.6 = 2.39$$

平均需要 **2.39** 位/字符。

11.16.4 题目 4

题目：How many leaves does a full 7-ary tree with 2024 vertices have? (6%) **考点**：第十一章树（Chapter 11.1 Introduction to Trees）——m-ary 树的顶点与叶子关系 **解析**：对于满 m-ary 树（每个内部节点恰好有 m 个孩子）：

- 总顶点数 $n = m \cdot i + 1$，其中 i 是内部节点数。
- 叶子数 $l = n - i$。

由 $n = m \cdot i + 1$ 得 $i = \frac{n-1}{m}$。叶子数 $l = n - i = n - \frac{n-1}{m} = \frac{mn-n+1}{m} = \frac{(m-1)n+1}{m}$。代入 $m = 7, n = 2024$：

$$l = \frac{(7-1) \times 2024 + 1}{7} = \frac{6 \times 2024 + 1}{7} = \frac{12\,144 + 1}{7} = \frac{12\,145}{7} = 1\,735$$

验证：$i = \frac{2024-1}{7} = \frac{2023}{7} = 289$，叶子数 $= 2024 - 289 = 1\,735$。答案为 **1735**。

11.16.5 题目 5

题目：Determine all positive integers r and s for which the complete bipartite graph $K_{r,s}$ is a tree. (6%) **考点**：第十一章树（Chapter 11.1 Introduction to Trees）——树的性质、完全二分图 **解析**：$K_{r,s}$ 有 $n = r + s$ 个顶点，$m = r \cdot s$ 条边。树满足 $m = n - 1$（连通无环）。所以需要：

$$r \cdot s = r + s - 1$$

整理：

$$r \cdot s - r - s + 1 = 0$$

$$(r-1)(s-1) = 0$$

所以 $r = 1$ 或 $s = 1$（另一个为正整数）。验证：

- $K_{1,s}$（星形图）：$r = 1, s \geq 1$，有 $1 + s$ 个顶点，$1 \times s = s$ 条边。$m = s = (1 + s) - 1 = n - 1$，是树。
- $K_{r,1}$：同理。

因此 $K_{r,s}$ 是树当且仅当 $r = 1$ 或 $s = 1$（且另一个为正整数）。

11.16.6 题目 6

题目：Suppose $|A| = 4$. Find the number of different equivalence relations on A. (6%) **考点**：第九章等价关系（Chapter 9.5 Equivalence Relations）——等价关系与划分、Bell 数 **解析**：集合 A 上的等价关系与 A 的划分一一对应（一个等价类对应一个划分块）。所以 $|A| = 4$ 时等价关系数 = 4 元集合的划分数 = Bell 数 B_4。列出所有划分：

1. 1 个块（所有元素在一起）：$\{\{a, b, c, d\}\}$ —1 种
2. 分成 3+1（一个 3 元块 + 一个单元素块）：选 3 个元素组成块：$\binom{4}{3} = 4$ 种
   如 $\{\{a, b, c\}, \{d\}\}$
3. 分成 2+2（两个 2 元块）：先选 2 个元素组成第一个块，剩下 2 个自然成块。
   但两个 2 元块无序，所以：$\frac{\binom{4}{2}}{2} = \frac{6}{2} = 3$ 种
   如 $\{\{a, b\}, \{c, d\}\}$
4. 分成 2+1+1（一个 2 元块 + 两个单元素块）：选 2 个元素组成 2 元块：$\binom{4}{2} = 6$ 种
   如 $\{\{a, b\}, \{c\}, \{d\}\}$
5. 分成 1+1+1+1（全部单独）：$\{\{a\}, \{b\}, \{c\}, \{d\}\}$ —1 种

总计：$1 + 4 + 3 + 6 + 1 = 15$ 种。答案为 **15**。

11.16.7 题目 7

题目：Answer these questions for the poset $(\{2, 3, 5, 6, 12, 20, 27, 36, 60\}, |)$. (a) (4%) Draw the Hasse diagram. (b) (2%) Find the maximal elements. (c) (2%) Is there a least element? (d) (2%) Find all upper bound of $\{2, 3\}$. **考点**：第九章偏序关系（Chapter 9.6 Partial Orderings）——哈斯图、极值元素 **解析**：集合 $S = \{2, 3, 5, 6, 12, 20, 27, 36, 60\}$，偏序关系为整除 |。**(a)** Hasse 图。首先找到所有整除关系：

- 2 | 6, 2 | 12, 2 | 20, 2 | 36, 2 | 60
- 3 | 6, 3 | 12, 3 | 27, 3 | 36, 3 | 60
- 5 | 20, 5 | 60
- 6 | 12, 6 | 36, 6 | 60
- 12 | 36, 12 | 60
- 20 | 60
- 27 无（27 的倍数 54, 81... 不在集合中）
- 36 无
- 60 无

覆盖关系（去掉传递边后的直接整除关系）：底层（极小元）：2, 3, 5 第二层：

- 6 覆盖 2, 3（即 2→6, 3→6 是覆盖，因为 2 和 3 之间没有中间元素）
- 20 覆盖 2, 5
- 27 覆盖 3

第三层：

- 12 覆盖 6（2→6→12，2→12 不是覆盖因为有 6 在中间）
- 36 覆盖 6, 12（3→6→36 和 3→12→36，3→36 不是覆盖）实际上 36 覆盖 12 吗？12 | 36，中间有 6 吗？6|12 且 6|36，但 12 和 36 之间没有整除关系的中间元素：6 不能整除 12 且被 36 整除（6 不在 12 和 36 之间，因为 12/6=2 是整数，但 6 不整除 12... 等等，6 确实整除 12！所以 2|6|12|36，所以 2→36 不是覆盖，但 12→36 也不是覆盖？因为 6|12|36 但 6 在 12 下面，所以 12 和 36 之间没有中间元素吗？如果有 x 使得 12 | x | 36，需要 12|x 且 x|36，且 x S。在 S 中，6 不满足 12|6。所以 12 和 36 之间没有中间元素。所以 12 覆盖 36 ）

重新审查覆盖关系：

关系	覆盖?	理由
2→6	是	2
2→12	否	2
2→20	是	2
2→36	否	2
2→60	否	2
3→6	是	3
3→12	否	3
3→27	是	3
3→36	否	3
3→60	否	3
5→20	是	5
5→60	否	5
6→12	是	6
6→36	是	6
6→60	否	6
12→36	是	12
12→60	是	12
20→60	是	20
27	-	极大元，无向上覆盖

更新 6→36：因为 6|12|36，所以 6→36 不是覆盖。36 的覆盖是 12→36。Hasse 图（从下到上，下层整除上层）：

```
27        36        60
 |       /  \      /  \
 3    12    |   20  /
  \   / \   |  /   /
   \ /   \  | /   /
    6     \ |/   /
   / \     \|   /
  2   3     5
```

更清晰的分层：顶层（极大元）：27, 36, 60 第三层：12, 20 第二层：6 底层（极小元）：2, 3, 5 覆盖边：2→6, 3→6, 3→27, 5→20, 6→12, 12→36, 12→60, 20→60 **(b)** 极大元（没有其他元素能整除它，即在偏序中它不被任何其他元素覆盖）：**27, 36, 60**（集合中没有比它们大的倍数了）**(c)** 最小元（小于等于所有元素）：需要找到一个元素能整除所有其他元素。2 不能整除 3 或 5 或 27。3 不能整除 2 或 5 或 20。5 不能整除 2 或 3 或 6 或 12 或 27 或 36。所以**没有最小元**。**(d)** $\{2, 3\}$ 的所有上界（能同时被 2 和 3 整除的元素，即 6 的倍数）：在 S 中，6 的倍数有：6, 12, 36, 60。所以上界为：**6, 12, 36, 60**。

11.16.8 题目 8

题目：In the network below (Fig.2), find a maximum flow from A to J, calculate its flow value, and prove that it is the maximum flow. (10%) **考点**：网络流（Network Flow）——最大流、最小割定理 **解析**：Fig.2 的网络图（有向边，括号内为容量）：A→B: 13, A→C: 7 B→C: 7, B→D: 10, B→F: 2 C→F: 9 D→G: 2, D→H: 9 F→H: 3, F→I: 3 G→H: 7, G→J: 9 H→I: 2, H→J: 9 I→J: 4 寻找最大流。尝试使用增广路径法：**路径 1**: A→B→D→G→J 瓶颈：min(13, 10, 2, 9) = 2 增广 2 单位。剩余容量：A→B:11, B→D:8, D→G:0, G→J:7 **路径 2**: A→B→D→H→J 瓶颈：min(11, 8, 9, 9) = 8 增广 8 单位。剩余容量：A→B:3, B→D:0, D→H:1, H→J:1 **路径 3**: A→B→F→H→J（B→F 容量 2，但 A→B 只有 3）瓶颈：min(3, 2, 3, 1) = 1 增广 1 单位。剩余容量：A→B:2, B→F:1, F→H:2, H→J:0 **路径 4**: A→B→F→I→J 瓶颈：min(2, 1, 3, 4) = 1 增广 1 单位。剩余容量：A→B:1, B→F:0, F→I:2, I→J:3 **路径 5**: A→C→F→I→J 瓶颈：min(7, 9, 2, 3) = 2 增广 2 单位。剩余容量：A→C:5, C→F:7, F→I:0, I→J:1 **路径 6**: A→C→F→H→I→J（H→I 容量 2）瓶颈：min(5, 7, 2, 2, 1) = 1 增广 1 单位。剩余容量：A→C:4, C→F:6, F→H:1, H→I:1, I→J:0 **路径 7**: A→B→C→F→H→J（H→J 已满）尝试其他路径。A→C→F→H→I→J（I→J 已满）。A→B→C→F→H→G→J? H→G 是反向允许吗（H→G 无直接边）。现在检查是否有其他增广路径：

- A→B: 剩余 1
- A→C: 剩余 4
- 尝试 A→C→F→H→I（I→J 已满），无法到 J。
- 尝试 A→B→C→F→H→G→J：B→C 可用，C→F 可用，F→H 可用，H→G? 没有 H→G 的边。只有 G→H 的边，但方向是 G→H，不能反向用（除非有之前的流可以抵消）。
- G→J 还有 7 容量。
- 找从 A 到 J 的路径（允许反向边）：A→C→F→H，但 H 之后无法到 J（H→J 满，H→I→J 中 I→J 满）。
- 通过反向 H→G? 不行，因为只有 G→H。

当前流值 = 2+8+1+1+2+1 = 15。**最小割**：找 s-t 割。考虑割 $S = \{A, B, C, D, F\}$，$T = \{G, H, I, J\}$：跨越割的边：

- D→G: 2（已满）
- D→H: 9（已用 8，还余 1）
- B→F: 2（已用 2，满）
- F→H: 3（已用 2，余 1）

- F→I: 3（已用 3，满）
- C→F: 9（但 C 在 S 中，F 在 S 中——等等 F 在 S 中，所以 C→F 不在割中）

重新确定 S。考虑割 $S = \{A, B, C, D\}$，$T = \{F, G, H, I, J\}$：跨越边：

- B→F: 2（满）
- D→G: 2（满）
- D→H: 9（已用 8）
- C→F: 9（还有剩余）

容量和 = 2+2+9+9 = 22。更好的割：$S = \{A, B, C, D, F, G, H\}$，$T = \{I, J\}$：跨越边：

- F→I: 3（满）
- H→I: 2（已用 1）
- G→J: 9（已用 2）
- H→J: 9（已用 9，满）
- I→J: 4（已用 4，满）

容量和 = 3+2+9+9+4 = 27。寻找最小割：考虑割 $S = \{A, B, C\}$，$T = \{D, F, G, H, I, J\}$：

- A→B 在 S 内
- B→D: 10
- B→F: 2
- C→F: 9

容量 = 10+2+9 = 21 割 $S = \{A, C\}$，$T = \{B, D, F, G, H, I, J\}$：

- A→B: 13
- C→F: 9
- C→B? A→C 在 S 内，C→B? 没有直接边。

容量 = 13+9 = 22 割 $S = \{A, B, C, D, F\}$，$T = \{G, H, I, J\}$：

- D→G: 2
- D→H: 9
- F→H: 3
- F→I: 3

容量 = 2+9+3+3 = 17 割 $S = \{A, B, C, D, F, G\}$（在 T 中加入 J 的路径检查）：其实更简单的方法是找到最小割容量为 15 的割来证明最大流为 15。考虑割：$S = \{A, B, C, D, F, G, H\}$（所有无到 J 的饱和路径的点）从 S 到 T = {I, J} 的边：

- F→I: 3
- H→I: 2
- G→J: 9（但 G 在 S 中，嗯 G 在 S 中? G 不能到 J 了吗? 还有 7 容量未用?）

让我们更仔细地找。其实从我们增广后的残留网络看：从源点 A 出发，用残留容量可以到达：

- A→B（剩余 1）
- A→C（剩余 4）
- B→C（剩余 7，B→C 容量 7，已用 0）
- C→F（剩余 6）
- F→H（剩余 1）
- H→I（剩余 1）
- （H→J 满，I→J 满，D→G 满，D→H 剩余 1 但 D...）

实际上我需要系统地做：在残留网络中（正向边剩余容量 > 0 或反向边容量 > 0），从 A 可到达的顶点：A → B (13-12=1>0) → C (7>0) → F (9-3=6>0) → H (3-2=1>0) → I (2-1=1>0) 同时 A → C (7-3=4>0) 直接 D: B→D 已满（10-10=0），所以 D 不可达。G: D→G 已满，H→G 没有正向边（反向可以有但 H→G 不存在，且 G→H 反向边容量 2? 残留网络中如果 G→H 有流 2，则反向边 H→G 可走）。是的！在残留网络中，D→G 流了 2，所以反向 G→D 边有容量 2。但 G 本身是否可达? 从 A 出发，不能到 D（B→D 满），不能到 G。所以 S = A, B, C, F, H, I。但等一下，H→I 可以，I→J 满。所以从 A 出发可达的顶点：A, B, C, F, H, I。检查 S 到 T 的边（其中 T = 所有其他顶点 = D, G, J）：

- B→D: 容量 10，但正向边已满（流 10），所以这个边在残留网络中不存在（正向满，反向在 D→B 有 10）。但原始割容量只看正向边！所以从 S 到 T 的边：B→D（容量 10），F→? F 没有到 D,G,J 的边，H→J（容量 9），I→J（容量 4）。等等，H 在 S 中，H→J 容量 9，还有流向 J 吗?
  修正：S 包含从 A 在残留网络中可达的所有顶点。如果 I 在 S 中但 I→J 满（4/4），则 I→J 正向满，不能从 I 到 J。但 J 不在 S 中。H 在 S 中，H→J 满（9/9），不能从 H 到 J。

所以 S = A, B, C, F, H, I，T = D, G, J。从 S 到 T 的边（原始容量）：

- B→D: 10
- H→J: 9
- I→J: 4

- 还有 C→? C 在 S，C 到 T? C 到 D? 没有边。
- F 在 S，F 到 T? F 到 H 在 S 内，F 到 I 在 S 内。
- B 在 S，B 到 D 已算，B 到 F 在 S 内。

割容量 = 10 + 9 + 4 = 23。太大了。让我重新考虑增广路径，也许我没有找到最优流。其实，更好的方法是先不管增广顺序，直接找最小割。在图中有 6 个节点（A, B, C, D, F, G, H, I, J）共 9 个节点? 节点列表：A, B, C, D, F, G, H, I, J（9 个节点）让我们尝试找最小割。直观上，瓶颈可能在 B→F (2), D→G (2), 和某个连接。实际上，我们可以直接用 min-cut = max-flow 来推理。从 A 出发的总出容量 = 13+7=20。从各个方向汇聚到 J 的入边有：G→J(9), H→J(9), I→J(4)，共 22。考虑割：S=A,B,D，T=C,F,G,H,I,J

- A→C: 7
- B→F: 2
- D→G: 2
- D→H: 9

容量 = 7+2+2+9 = 20 割：S=A,B,D,F，T=C,G,H,I,J

- A→C: 7
- B→F? F 在 T 中... 不 F 在 S 中
- B→D? D 在 S 中
- D→G: 2
- D→H: 9
- F→H: 3
- F→I: 3

容量 = 7+2+9+3+3 = 24 割：S=A,B,C,D，T=F,G,H,I,J

- A→C? C 在 S 中
- B→F: 2
- D→G: 2
- D→H: 9
- C→F: 9

容量 = 2+2+9+9 = 22 割：S=A,B,D，T=C,F,G,H,I,J

- A→C: 7
- B→F: 2
- D→G: 2
- D→H: 9

容量 = 7+2+2+9 = 20 割：S=A,B,C,D,F，T=G,H,I,J

- D→G: 2
- D→H: 9
- F→H: 3
- F→I: 3

容量 = 2+9+3+3 = 17 割：S=A,B,D,F,H，T=C,G,I,J

- A→C: 7
- D→G: 2
- F→I: 3
- H→I: 2
- H→J: 9

容量 = 7+2+3+2+9 = 23 割：S=A,B,D,G，T=C,F,H,I,J

- A→C: 7
- B→F: 2
- D→H: 9
- G→H: 7
- G→J: 9

容量 = 7+2+9+7+9 = 34 割：S=A,B,C,D,G，T=F,H,I,J

- B→F: 2
- D→H: 9
- C→F: 9
- G→H: 7
- G→J: 9

容量 = 2+9+9+7+9 = 36 割：S=A,B,C,D,G,H，T=F,I,J

- B→F: 2

- C→F: 9
- G→J: 9
- H→I: 2
- H→J: 9

容量 = 2+9+9+2+9 = 31 继续找最小：割：S=A,C,F,H，T=B,D,G,I,J

- A→B: 13
- C→B? 没有
- C→F? F 在 S 中
- F→? F 只有到 H 和 I，H 在 S
- H→I: 2
- H→J: 9
- H→G? 没有 H→G

容量 = 13+2+9 = 24 割：S=A,B,C,F,H,I，T=D,G,J

- B→D: 10
- H→J: 9
- I→J: 4

容量 = 10+9+4 = 23 割：S=A,B,C,D,H，T=F,G,I,J

- C→F: 9
- B→F: 2
- D→G: 2
- H→I: 2
- H→J: 9

容量 = 9+2+2+2+9 = 24 割：S=A,B,C,D,H,I，T=F,G,J

- C→F: 9
- B→F: 2
- D→G: 2
- H→J: 9

容量 = 9+2+2+9 = 22 看来最小割可能在 17。让我验证一下是否可以达到 17 的流。尝试最大流 17：先想好哪些边应该饱和。如果最小割是 A,B,C,D,F/G,H,I,J，容量 17，则从 S 到 T 的边全部饱和（D→G 2, D→H 9, F→H 3, F→I 3）。那么从 A 出发到 B, C 应该有足够的流。A→B 需要到 D（12）+ 到 F（B→F 2 + 通过其他）= 至少 12+2 = 14? 实际上 B→D 有 10 容量，B→F 有 2 容量，所以 B 至少需要收 12。A→B 容量 13，可以发 12。A→C 需要到 F 9（C→F），所以 A→C 发 9。剩余 A→B 的 1 还可以发到别处。等等让我重新考虑。如果割是 A,B,C,D,F/G,H,I,J 容量 17，则从 S 到 T 的四条边都必须满才能达到 17 的流值。即需要 D→G=2, D→H=9（需要 D 收 11），F→H=3, F→I=3（需要 F 收 6）。D 的入边：只有 B→D（容量 10）。所以 B→D 最多 10，但需要 11... 这是个问题。重新考虑：D→H 9, D→G 2，所以 D 需要 11 的入流。但 B→D 只有 10，且没有其他入边到 D。所以这个割的容量虽然为 17，但无法实现总流 17（因为从 A 到 T 的路径的中间能力不足）。这说明最小割可能不是这个。让我找最小割更仔细。考虑割 S=A,B,C,D,F,H, T=G,I,J:

- D→G: 2
- F→I: 3
- H→I: 2
- H→J: 9

容量 = 2+3+2+9 = 16 如果这个割是可行的（即所有 S 到 T 的边都能被饱和），那最大流可能是 16。验证：S 到 T 的边是 D→G(2), F→I(3), H→I(2), H→J(9)。需要 D 出 2，F 出 3，H 出 11（2+9）。D 入：B→D(10) （只需要 2）F 入：B→F(2), C→F(9) （只需要 3）H 入：D→H(9), F→H(3), G→H(7), 但 G 在 T 中，不能从 G 入 H。D→H(9) 给 H 9，F→H(3) 给 H 3，共 12。H 需要出 11，可以。所以流安排：

- B→D: 2（给 D，D→G 用 2）
- D→H: 2（D 收 2 全部给 H? 但 H 还需要更多。D 还可以从别处收? D 仅有 B→D 一条入边，所以 D 最多收 10。）

等等，我们安排：

- B→D: 2 → D→G: 2（刚好）
- B→F: 2, C→F: 1 或 D→H 需要更多...

H 需要 11（H→I 2 + H→J 9）= 11。H 的入边：D→H(9), F→H(3), G→H(7)——但 G 在 T 中不能。所以 D→H + F→H = 11。

- D 需要出 2（D→G）+ 给 H 一些 = B→D 的总出最多 10。D→H 最多 8（因为 B→D 最大 10，D→G 用 2 剩下 8 给 D→H）。
- F→H 最少 3（因为 H 需要 11，D→H 最多 8）。

所以：B→D: 10，其中 D→G: 2，D→H: 8。F 需要出 3（F→I）+ 给 H。F 入：B→F(2), C→F(9)。B→F: 2, C→F: 最小 1（F→I 3 中 2 来自 B→F，1 来自 C→F）。F→H: 3（H 需要 11，D→H 给 8，F→H 给 3）。A→B 需要 10+2 = 12。A→B 容量 13 。A→C 需要 1（C→F）。A→C 容量 7 。剩余 6 可以闲置或通过 C→? C 还可到... C 只有到 F 的边。H→I: 2, H→J: 9 。H 总入 8+3=11 。I 出：H→I(2) + F→I(3) = 5，I→J(4)，多出 1 闲置或需要调整。F→I=3，但 I→J 只有 4。H→I=2。总共入 I = 5，出 I 最多 4（I→J=4）。所

以 I 不能收 5，只能收 4！调整：F→I=3, H→I=1。则 H 出：H→I=1, H→J=9，H 总出 =10。H 入：D→H(8) + F→H(3) = 11，但出只需 10。多 1 闲置或可以加到 I 中？额，H 入 = 11，出 = 10，违反流量守恒。需要 H 入 = H 出。所以：H→J=9, H→I=1，共 10。入也需要 10。D→H=7, F→H=3 共 10。调整：D→H=7, D→G=2，B→D=9。B→F=2, C→F=... F→H=3, F→I=3，所以 F 出 =6。F 入：B→F=2, C→F=4。A→B=9+2=11, A→C=4。现在检查所有节点流量守恒：A: 出 11+4=15（A 是源点）B: 入 11, 出 9+2=11　C: 入 4, 出 4　D: 入 9, 出 2+7=9　F: 入 2+4=6, 出 3+3=6　H: 入 7+3=10, 出 1+9=10　I: 入 1+3=4, 出 4　G: 入 2, 出... G→J=2（G 入 2 出 2，G→H 不用）或 G→H(0) J: 入 G→J(2) + H→J(9) + I→J(4) = 15　总流 = 15。这与之前找的 15 一致。所以最大流 = 15。更简洁的最小割：割 S=A,B,C,D,F, T=G,H,I,J：容量 = D→G:2 + D→H:9 + F→H:3 + F→I:3 = 17 但上面的分析说明由于 I→J=4 的限制，实际上最大只能到 15。割 S=A,B,C,D,F,H,I,J... 这太大了。真正的最小割应该在 15。让我找容量为 15 的割。割 S=A,B,C,D,F,H: T=G,I,J S→T: D→G(2), F→I(3), H→I(2), H→J(9) = 16 S=A,B,C,D,F：T=G,H,I,J S→T: D→G(2), D→H(9), F→H(3), F→I(3) = 17 让我检查流量分配能否达到 16。如上面分析，H 和 I 之间存在约束。I 的入不能超过 4（I→J 容量 4）。F→I=3, H→I=1 时 H 的出 =10，但 H 的入不够... 实际上更系统的分析：最大流的瓶颈路径中，A→J 的所有路径集合：

1. A→B→D→G→J：容量 min(13,10,2,9)=2
2. A→B→D→H→J：容量 min(13,10,9,9)=9
3. A→B→F→H→J：容量 min(13,2,3,9)=2
4. A→B→F→I→J：容量 min(13,2,3,4)=2
5. A→C→F→H→J：容量 min(7,9,3,9)=3
6. A→C→F→I→J：容量 min(7,9,3,4)=3
7. A→B→C→F→H→J：容量 min(13,7,9,3,9)=3 但受 B→C 的 7 限制

考虑 I→J 只有 4 容量，且所有到 I 的流都必须通过 I→J 出去。F→I=3, H→I=2，所以 I→J 的最大流可以处理 4（F→I 3 + H→I 1 或各 2）。所以 I 相关路径的总和受限于 min(F→I+H→I, I→J)=min(5,4)=4。G→J=9, H→J=9。再看更全面的约束。哪些边的容量是整体瓶颈？B→D 10, B→F 2, C→F 9, D→G 2。路径约束：

- 经过 D→G 的流最多 2
- 经过 B→F 的流最多 2
- 经过 I→J 的流最多 4
- 经过 D→H 的流最多 9（B→D 剩余）

最大流不超过 2+9+2+2 = 15（不考虑 I→J 约束为 4）。更准确：min-cut 分析。割 S=A, T=B,C,D,F,G,H,I,J：容量 = 13+7=20 割 S=A,B,C,D,F,H,I, T=G,J：S→T: D→G(2), H→J(9), I→J(4) = 15 ！验证这个割是否有意义（即 S 中的所有顶点在残留网络中从源点可达？不一定需要，我们只是在找最小割）。S=A,B,C,D,F,H,I 中，所有顶点通过已有边可达 A。在流量分配中，这个割的边 D→G(2), H→J(9), I→J(4) 的容量和为 15。所以 **最大流 = 15**。最大流的分配方式（如上推导）：

- A→B: 11, A→C: 4
- B→D: 9, B→F: 2
- C→F: 4
- D→G: 2, D→H: 7
- F→H: 3, F→I: 3
- G→J: 2
- H→I: 1, H→J: 9
- I→J: 4

总流 = 15。最小割容量 = 15。用最小割定理：最大流值等于最小割容量。我们找到了容量为 15 的割 A,B,C,D,F,H,I/G,J，其容量为 D→G(2)+H→J(9)+I→J(4)=15。且找到了流值为 15 的可行流。所以最大流为 **15**。

11.16.9　题目 9

题目：Determine if the given pair of graphs (Fig.3) is isomorphic. Give the reason. (8%) **考点**：第十章图的同构（Chapter 10.3 Representing Graphs and Isomorphism）**解析**：Fig.3 左侧图：顶点 1-8 和标号（这是一个 8 个顶点的图）。右侧图：顶点 a-h，也是 8 个顶点。左侧图的边（从图中看）：1-2, 1-4, 1-6 2-3, 2-5 3-4, 3-8 4-7 5-6, 5-8 6-7 7-8（需要从 Typst 代码中确认边）从 Typst 代码看左侧图：

```
redge("1", "2")
redge("1", "4")
redge("1", "6")
redge("3", "2")
redge("3", "4")
redge("3", "8")
redge("5", "2")
redge("5", "6")
redge("5", "8")
redge("7", "4")
redge("7", "6")
redge("7", "8")
```

左侧图边：1-2, 1-4, 1-6 2-3, 2-5 3-4, 3-8 4-7 5-6, 5-8 6-7 7-8 右侧图边：a-b, b-h, h-g, g-a, c-d, d-f, f-e, e-c, a-c, b-d, h-f, g-e 为了判断是

否同构，我们需要检查是否有一个一一对应关系保持边结构。注意到左侧图是 4-正则图（每个顶点度 = 3），右侧图也是每个顶点度 = 3。实际上，这个图是两个重叠的 K_4 再连接？让我们仔细分析。实际上，从结构来看，左侧图由两个 4-环通过匹配边连接而成，类似于立方体。右侧图也一样。尝试建立同构映射：左侧：1→a, 2→b, 3→c, 4→d, 5→e, 6→f, 7→g, 8→h 检查对应边：1-2 → a-b　1-4 → a-d？右侧没有 a-d。努力中... 其实这两个图是同构的，都是**立方体图 Q_3**（3-立方体）的另一种画法？检查：Q_3 有 8 个顶点，每个度 = 3，有 12 条边。左侧有 12 条边（计数上面列表 = 12 条边）。右侧也有 12 条边。是的，这两个都是 Q_3，所以是同构的。标准同构映射可以通过匹配顶点度序列和邻接模式来建立。直观上，两个图都是将 8 个顶点排列成两个 4-环，然后在环之间连接匹配边。所以它们是**同构的**。

11.16.10　题目 10

题目：Q_n is the graph with 2^n vertices representing bit strings of length n. An edge exists between two vertices that differ in exactly one bit position. (a) (3%) Find the number of edges of Q_5. (b) (3%) Find the chromatic number of Q_5. Give the reason. (c) (6%) Determine if Q_5 has Hamilton circuit / path. Give the reason. **考点**：第十章图论（Graph Theory）——超立方体图、图染色、哈密顿图 **解析**：**(a)** Q_n 的边数。每个顶点对应一个 n-位二进制串，共有 2^n 个顶点。每个顶点有 n 个邻居（翻转任意一位），所以总度数为 $2^n \cdot n$。每条边贡献 2 个度，所以边数为：

$$|E(Q_n)| = \frac{2^n \cdot n}{2} = n \cdot 2^{n-1}$$

对于 $n = 5$：

$$|E(Q_5)| = 5 \times 2^4 = 5 \times 16 = 80$$

(b) Q_n 的色数。Q_n 是二分图（bipartite graph）：将顶点按二进制串中 1 的个数的奇偶性分为两部分。每条边翻转一位，所以 1 的个数的奇偶性必然改变。因此可以 2-着色（一部分涂黑，一部分涂白）。所以色数为 **2**。对于 Q_5 也是 2。**(c)** Q_n 的哈密顿回路/路径。Q_n 对 $n \geq 2$ 总有哈密顿回路。证明：归纳法。

- Q_1 有 2 个顶点，有哈密顿路径但没有回路（是 K_2）。
- Q_2 是 4-环，显然有哈密顿回路。
- Q_n 可以由两个 Q_{n-1} 的副本通过在对应顶点之间加边构成。

归纳步骤：Q_n 由两个 Q_{n-1}（记作 Q^0_{n-1} 和 Q^1_{n-1}）组成，Q^0_{n-1} 的顶点是首位为 0 的串，Q^1_{n-1} 的顶点是首位为 1 的串。对应顶点（后 $n-1$ 位相同）之间用边连接。假设 Q_{n-1} 有哈密顿回路。设 $v_0^{(1)}, v_0^{(2)}, ..., v_0^{(2^{n-1})}$ 是 Q^0_{n-1} 中的哈密顿回路。从回路中找一条边 $v_0^{(i)} v_0^{(i+1)}$，将其替换为两条边：$v_0^{(i)} - v_1^{(i)} - v_1^{(i+1)} - v_0^{(i+1)}$，其中 $v_1^{(i)}$ 和 $v_1^{(i+1)}$ 是 Q^1_{n-1} 中对应的顶点。这样就可以构造出 Q_n 的哈密顿回路。因此，对 $n \geq 2$，Q_n 有哈密顿回路（当然也有哈密顿路径）。对于 Q_5：有哈密顿回路，也有哈密顿路径。

11.16.11　题目 11

题目：8 students take a test with 8 true/false questions. It is known that no two students make exactly the same choice. Prove that we can remove one of the 8 questions, and still no two students make exactly the same choice. (12%) **考点**：组合数学、鸽巢原理、向量空间 **解析**：我们将每个学生的答题情况表示为一个长度为 8 的二进制向量，第 i 位表示第 i 题的答案（例如 1 表示 True，0 表示 False）。已知 8 个学生的向量两两不同，即我们有 8 个不同的 8-位二进制向量。**问题**：证明存在一列（一道题），去掉该列后（即移除该分量后），剩下的 7 个分量形成的 8 个 7-位向量仍然两两不同。**等价表述**：在 8 个不同的 8-位二进制向量中，存在一个坐标位置，使得去掉该坐标后，剩下的 7 个坐标仍然能区分所有 8 个向量。**证明**：假设结论不成立，即去掉任意一列后，至少有两个向量在该列以外其他列完全相同。对每个列位置 i（$1 \leq i \leq 8$），存在两个不同的学生 s, t，使得他们在除第 i 题外的所有题目上答案相同，只在第 i 题上不同。也就是说，对于每个 i，存在一对向量 v, w，它们仅在坐标 i 上不同。现在我们可以将 8 个向量看作 $\mathbb{F}_2^8$ 中的 8 个点。如果一对向量仅在坐标 i 上不同，则它们的差是第 i 个单位向量 e_i。因此对于每个 i（$1 \leq i \leq 8$），存在一对向量，其差为 e_i。这意味着 $e_1, e_2, ..., e_8$ 都在这组向量的差集中。但 8 个向量在 $\mathbb{F}_2^8$ 中的差可以生成最多 $\binom{8}{2} = 28$ 个不同的差向量，且这些差向量都是非零的。如果每个 e_i 都出现，则这 8 个单位向量互不相同，这是可能的。现在用另一种思路：考虑这 8 个向量组成的 8×8 矩阵（行代表学生，列代表题目）。该矩阵的 8 行两两不同。我们要证明存在一列，去掉后行仍然两两不同。注意到如果去掉第 i 列后，某两行变得相同，则原来的两行只在第 i 列不同。所以，假设对所有列 i，去掉第 i 列后都有某两行相同，则对每个 i，存在一对行（学生）只在第 i 列不同。由于只有 $\binom{8}{2} = 28$ 对学生对，而每列对应一个学生对，所以可能存在 8 个学生对分别对应 8 列。但这要求每列都有一对唯一的行。考虑向量的**奇偶性**。对每个向量，计算其所有分量的和（模 2），即汉明重量的奇偶性。如果两个向量只在第 i 位不同，则它们的分量和差 1（模 2），即一个为偶数另一个为奇数。所以在每列对应的一对向量中，两个向量属于不同的奇偶类。但 8 个向量中，奇偶类（偶/奇）的分布最多有 8+0, 7+1, 6+2, 5+3, 4+4 这几种方式。每对只在一列不同的向量必须是一奇一偶，所以总共有多少对这样的向量对呢？考虑所有可能的配对数量限制。**更简洁的证明**（用鸽巢原理和组合推理）：设 8 个学生的答案向量为 $v_1, ..., v_8 \in \{0,1\}^8$，两两不同。对每个学生 s，将其答案向量表示为二进制数（0-255 之间的一个整数）。考虑所有 8 个向量，其和向量（逐位 XOR）记为 $S = v_1 \oplus v_2 \oplus \cdots \oplus v_8$。如果去掉第 i 列后，有两位学生变得相同，设他们是 p 和 q，则 v_p 和 v_q 只在第 i 位不同，即 $v_p \oplus v_q = e_i$。对 p 和 q 两个学生，它们在所有其他列上都相同。这意味着在去掉第 i 列的 7 列中，他们是相同的，所以只剩下第 i 列可以区分。因此，对每个列 i，至多能有一对学生在去掉第 i 列后相同（否则如果有两对，则那两对的区分度不够）。但 8 个不同向量之间的两两差异至少需要若干列来区分。考虑二进制向量空间 $\mathbb{F}_2^8$。8 个两两不同的向量一定能在某个坐标位置被区分。具体地，考虑 Hamming 距离的概念。其实可以用一个简单论断：在 8 个不同的 8 位向量中，必然存在一个坐标，该坐标上 8 个向量的取值不全为 0 也不全为 1（否则所有向量在该坐标相同，则该坐标是冗余的，可以直接移除）。如果每个坐标都既有 0 又有 1，那是否可以保证去掉任何一列都仍然能区分所有向量？不一定。考虑以下反例模式：8 个向量恰好成对，每对只在某一列不同。如果这种模式能构造出来，则结论可能不成立。但我们需要证明这种模式不可能存在。**严格证明**：假设结论不成立，即对每个 $i \in \{1, \ldots, 8\}$，存在两个学生（向量）$x^{(i)}, y^{(i)}$ 满足：

- $x^{(i)}$ 和 $y^{(i)}$ 只在第 i 坐标上不同

- 在其他 7 个坐标上完全相同

现在，对固定的 i，$x^{(i)}$ 和 $y^{(i)}$ 由它们在其余 7 个坐标上的公共值唯一确定（一个是 $v \cup \{0_i\}$，另一个是 $v \cup \{1_i\}$）。所以实际上对每个 i，对应于一个 7-位向量 $u^{(i)} \in \{0,1\}^7$，产生两个 8-位向量 $(u_1^{(i)}, ..., u_{i-1}^{(i)}, 0, u_i^{(i)}, ..., u_7^{(i)})$ 和 $(u_1^{(i)}, ..., u_{i-1}^{(i)}, 1, u_i^{(i)}, ..., u_7^{(i)})$。由于所有 8 个学生对应 8 个不同的向量，且每个 i 产生两个向量，所以这 8 个向量必须是 4 对（即 4 个不同的 i 值各产生一对），或者 8 个向量由全部 8 个 i 各产生一个向量（共 8 对，但只有 8 个向量，不可能）。等等，8 个向量，每个 i 需要产生两个不同的向量（一对），但总共有 8 个 i 和 8 个向量，所以实际上是：对每一个 i，存在一对向量只在第 i 位不同。但这意味着 8 个向量可分成 4 对，每对只在某一位不同。也就是 8 个向量由 4 个轴各自产生两个向量。但 8 个不同的 i 需要 8 对（16 个向量），而我们只有 8 个向量。矛盾? 不一定，因为同一对向量可能在多个坐标上相同（只在某一列不同，但可能同时在多个列不同? 不，条件是说对每个 i，存在某对向量只在第 i 列不同，它们在其他列相同）。让我们重新理解：对每个 $i \in \{1, \ldots, 8\}$，存在一对 (a_i, b_i)，它们只在第 i 位不同。但这 8 对可能不是互斥的——同一对向量可能在多个 i 上被选中（比如一对向量只在第 1 和第 2 位不同，则这对向量在去掉第 3-8 列都无法区分，但去掉第 1 列或第 2 列时都可以区分）。实际上，如果一对向量只在第 i 位不同，则去掉第 i 列后它们相同。所以" 去掉第 i 列后相同" 的对就是只在第 i 列不同的对。现在，一个向量对可能同时在多个列上不同。如果向量对 (p, q) 在 k 个列上不同，则去掉这些列中的任何一个都会使它们相同。所以这对向量同时是多个 i 的" 罪魁祸首"。所以 8 个向量的 28 个可能对中，可以覆盖全部的 8 个列。要使每列都被覆盖，需要足够的" 不同位置"。由于每个向量对有至少 1 个不同位置，最多 8 个不同位置。关键是：如果 8 个向量中没有两个是相同的，则任意两个向量有至少 1 个不同位置。但要求对每列 i，有一对向量只在第 i 位不同（即 Hamming 距离为 1 且恰好是第 i 位）。更简洁的证明（使用奇偶性）：将每个学生的答案向量写成长度为 8 的二进制串。定义第 s 个学生的" 答案和" 为 $f(s) = (v_s)_1 + (v_s)_2 + \cdots + (v_s)_8 \pmod 2$（即二进制表示的奇偶性）。如果有两个学生 p, q 只在第 i 题上答案不同，则 $f(p) \neq f(q)$（因为其他题相同，只有第 i 题一真一假，所以奇偶性不同）。所以对于每个 i，存在一对学生，其中一个是奇数答题和，另一个是偶数。但 8 个学生中，奇偶性数量分布只可能是 0+8, 1+7, 2+6, 3+5, 4+4, 5+3, 6+2, 7+1, 8+0。最多只能形成 奇个数 × 偶个数 个奇偶性不同的学生对。为了使每列 i 都有一个奇偶性不同的" 只在第 i 列不同" 的学生对，需要奇偶对的数量至少为 8。但 奇个数 × 偶个数 在奇偶分布为 4-4 时最大，等于 16，所以这是可能的。这种方法不能直接解决问题。**改用线性代数方法**：8 个向量在 $\mathbb{F}_2^8$ 中。如果对每列 i 都存在一对向量只在第 i 列不同，则单位向量 $e_1, ..., e_8$ 中每一个都可以表示为某对向量的差（在 $\mathbb{F}_2^8$ 中）。如果某对向量 (v, w) 只在第 i 列不同，则 $v - w = e_i$（在 $\mathbb{F}_2$ 中，减等于加）。由 8 个两两不同的向量可以形成最多 $\binom{8}{2} = 28$ 个差向量。每个 e_i 至少出现一次。这不能直接推出矛盾。**标准答案可能使用**：考虑所有 8 个向量的集合 S。对任意子集 $T \subseteq S$，定义其 XOR 和 $X(T) = \bigoplus_{v \in T} v$（逐位 XOR）。有 $2^8 = 256$ 种可能的 XOR 值（但当然这里只有 8 个向量）。实际上，有一个更简洁的证明：** 对每个学生，将其第 i 题的答案记为 a_{si}。构造 8×8 矩阵 $A = (a_{si})$。每行是一个学生的答案，行互不相同。题目说，对任意第 k 题 $k = 1, \ldots, 8$，去掉第 k 列后，存在两行相同。这意味着，列 k 是关键区分列，即对任意 k，存在行 s, t 使得它们在所有 $j \neq k$ 的列上相同，但在列 k 上不同。这意味着，矩阵中第 s 行与第 t 行的差（Hamming 距离）为 1（仅在列 k 不同）。因此，对所有 8 列 k，存在对应的行对 (s_k, t_k) 使得行差 $= e_k$。现在考虑由 8 个行向量张成的线性空间。由于行互不相同，它们不都是全 0。但更重要的是，所有行向量都属于某个仿射子空间。实际上有一个简洁的结论：8 个不同的行向量不可能满足" 对每列 k，存在一对只在 k 列不同的行" 当 k=8 是列数。反证：根据鸽巢原理，8 个向量只能形成 28 个不同的对。每个对在第 d 列不同（d 是它们不同的列数，至少为 1）。要能覆盖 8 列，需要 8 个这样的对（可能有些对覆盖多列，但" 只在第 i 列不同" 意味着该对只在一列不同）。所以需要至少 8 个不同的**距离为 1** 的向量对，且这些对覆盖了全部 8 列。现在考虑 8 个向量，形成 8 个距离为 1 的" 关键" 对（每个关键对只在某一列不同）。观察：如果 (v, w) 是只在第 i 列不同的关键对，那么它们在其他 7 列上完全相同。这意味着所有 8 个向量中，第 i 列以外的 7 列构成的投影中，这两个向量被映射到同一个点。考虑 8 个向量在 8 列上的投影。其实这个问题等价于：**一个 8×8 的 0-1 矩阵，行互不相同。证明存在一列，去掉该列后行仍然互不相同。**有标准结论：矩阵的秩（在 $\mathbb{F}_2$ 上）至少是去掉一列还能保持行互不相同的最大列数。但这里行只有 8 个，所以只要 8 个向量线性无关（或至少秩 7），就能保证去掉任何一列后仍有 8 个互不相同的投影。但 8 个向量在 8 维空间中，秩最多为 8。如果秩 $= 8$，则任何一列去掉后剩下的 7 列仍然能区分所有向量（因为秩 7，7 维空间中有 8 个不同的点）。矛盾只发生在秩 < 7 时。但题设没有保证秩 $= 8$。实际上 8 个不同的向量在 $\mathbb{F}_2^8$ 中可能线性相关。但题目要求的是" 存在一列可以移除"，不是" 所有列都可以"。所以我们只需要证明不可能出现" 每一列都是关键列" 的情况。实际上有一个著名的组合结果：对于任意 8 个互不相同的长度为 8 的二进制向量，总存在一个坐标位置，使得去掉该坐标后剩下的 7 个向量仍互不相同。这个结论对任意 m 个长度 m 的二进制向量都成立，当 m 是 2 的幂时尤其容易证明。**简洁证明**（使用鸽巢原理）：假设否，即对每列 i，去掉第 i 列后存在两行变得相同。则每列 i 对应一对行，它们在去掉第 i 列后相同。用 P_i 表示对应第 i 列的学生对（无序对）。由于只有 $\binom{8}{2} = 28$ 个可能的学生对，而我们要分配 8 个列给这些对（不同列可以分配给相同对）。如果存在两个不同的列 $i \neq j$ 分配给相同的对 (p, q)，那么 p 和 q 去掉第 i 列后相同且去掉第 j 列后相同。这意味着 p 和 q 在第 i 列和第 j 列上都不同（因为去掉它们各自后都相同），而在其他列上都相同。所以 p 和 q 在恰好两个列上不同。这没问题。但关键问题是：无论如何分配，去掉的列必须对应一个关键对，该对只在那一列不同（否则去掉该列后它们不会相同）。等等，我可能需要回归更基础的推理。标准解法：8 个学生，8 道题。每个学生答案是一个长度为 8 的 0-1 串。有 8 个互不相同的串。反证：假设去掉任何一题（列），都会导致某两个学生答案相同。那么对每个第 i 题，存在学生对 (s_i, t_i)，他们在除第 i 题外的所有题上答案相同。即 s_i 和 t_i 的答案只在第 i 位不同，其他位完全相同。记学生 s 的答案向量为 $v_s \in \{0,1\}^8$。那么 $v_{s_i} \oplus v_{t_i} = e_i$（第 i 个单位向量），对所有 i = 1,...,8。考虑这 8 个向量 $v_1, ..., v_8$。它们的集合为 S。对每个 i，存在 $p_i, q_i \in S$ 使得 $p_i \oplus q_i = e_i$。这意味着 S 包含了覆盖所有单位向量的差。但这可能吗? 考虑 S 的仿射包。如果 S 是某个 3 维仿射子空间（包含 8 个向量），则其中的差向量构成一个 3 维子空间（共 8 个差向量），不可能包含全部 8 个单位向量（需要 8 维）。更一般地，8 个向量的差集最多能形成 $\binom{8}{2}$ 个向量，但它们的线性张成最多是 7 维（因为 8 个向量在 8 维空间中）。通过更细致的分析：8 个向量在 $\mathbb{F}_2^8$ 中，如果它们的差包含了 8 个单位向量 $e_1, ..., e_8$，则这些单位向量张成了整个 $\mathbb{F}_2^8$。但 8 个向量最多能生成一个秩为 8 的差空间（如果这 8 个向量仿射无关）。但 8 个向量生成的差空间的维数最多为 7（因为差空间是原始向量空间的平移，在 $\mathbb{F}_2$ 上，n 个向量最多生成秩为 $n-1$ 的差空间）。等等，在 $\mathbb{F}_2$ 上，一组向量 $\{v_1, ..., v_n\}$ 的差集 $\{v_i - v_j : i \neq j\}$ 生成的子空间等于这些向量的线性张成（但去掉平移）。实际上，固定 v_1，则集合 $\{v_i - v_1 : i = 2, ..., n\}$ 张成的子

空间即 span$\{v_2 - v_1, ..., v_n - v_1\}$。如果这 n-1 个差向量张成了整个 $\mathbb{F}_2^8$，则 n-1 8，即 n 9。但我们只有 8 个向量，所以最大秩为 7。因此，8 个向量的差空间最多是 7 维的，不能包含全部 8 个单位向量。但我们的假设要求 $e_1, ..., e_8$ 都在差集中，而 $e_1, ..., e_8$ 张成了整个 $\mathbb{F}_2^8$（8 维），这要求差空间至少是 8 维。矛盾!（需要小心：" 差空间" 即由差向量张成的线性子空间。8 个 $\mathbb{F}_2^8$ 中的向量，差集最多张成 7 维子空间，因为 $v_i - v_1, i = 2, ..., 8$ 这 7 个向量张成了差空间。）具体地，对任意 i，$e_i = v_{s_i} \oplus v_{t_i} = v_{s_i} \oplus v_{t_i}$。那么所有 e_i 都在由 $\{v_s - v_t\}$ 张成的空间中。但 8 个向量最多产生 7 个线性无关差（差空间维数 7），却需要包含 8 个线性无关的向量 $e_1, ..., e_8$。矛盾。因此假设不成立，原命题得证。**证毕**。

11.17 DMA Quiz 4 题解—2021-2022 春夏 (郑文庭班)

11.17.1 题目 1 (18%)：谓词逻辑翻译

题目：给定谓词 `taller(x, y)`（"x 比 y 高"）和 `successful(x)`（"x 是成功的领袖"），用谓词逻辑表达以下语句。**(a)** Lincoln was the tallest leader. **考点**：一阶谓词逻辑—全称量词与唯一性表达 **解析**："Lincoln 是最高的领袖" 意味着 Lincoln 比所有其他领袖都高。注意这里不需要表达唯一性（"the tallest" 暗示 Lincoln 是唯一最高的），用全称量词即可。设 Lincoln 为常量 l，对任意领袖 y，只要 y 不是 Lincoln，则 taller(l, y) 成立。但需要注意论域是所有领袖，我们需要一个谓词 leader(x) 来限定。不过题目未提供 leader 谓词，我们可以直接对任意人选言，或者引入 leader 谓词。标准写法（引入 leader 谓词辅助，或默认论域为领袖集合）：若论域为所有历史人物，需引入 leader(x)：

$$\forall x\,(\text{leader}(x) \wedge x \neq \text{Lincoln} \to \text{taller}(\text{Lincoln}, x))$$

若默认论域就是所有领袖，则简写为：

$$\forall x\,(x \neq \text{Lincoln} \to \text{taller}(\text{Lincoln}, x))$$

(b) Napoleon was at least as tall as any unsuccessful leader. **考点**：全称量词 + 蕴含 **解析**：" 拿破仑至少和任何不成功的领袖一样高" 意味着：对所有 x，如果 x 是不成功的领袖，则拿破仑不矮于 x (即 taller(Napoleon, x) 为真或同样高?)。"at least as tall as" 即 taller(Napoleon, x) Napoleon = x 的某种表达。但题目只有 taller(x,y) 表示严格高于。我们可以用 ¬taller(x, Napoleon) 来表示 x 不比拿破仑高，即拿破仑 x 的高度。设拿破仑为常量 n：

$$\forall x\,(\text{leader}(x) \wedge \neg\text{successful}(x) \to \neg\text{taller}(x, n))$$

这表示：所有不成功的领袖都不比拿破仑高，即拿破仑至少和他们一样高。

(c) No two leaders had the same height. **考点**：全称量词 + 否定 + 等词 **解析**：" 没有两个领袖有相同的高度" ——即任意两个不同的领袖，一个不会和另一个一样高。给定只有 taller(x,y) 谓词（严格高于），两人一样高意味着既不是 x 比 y 高也不是 y 比 x 高：

$$\forall x \forall y\,(\text{leader}(x) \wedge \text{leader}(y) \wedge x \neq y \to \neg(\neg\text{taller}(x, y) \wedge \neg\text{taller}(y, x)))$$

等价于：

$$\forall x \forall y\,(\text{leader}(x) \wedge \text{leader}(y) \wedge x \neq y \to \text{taller}(x, y) \vee \text{taller}(y, x))$$

即任意两个不同的领袖，一个比另一个高。

11.17.2 题目 2 (12%)：Almost-Palindromes 计数

题目：一个四位数称为 **almost-palindrome**，如果恰通过改变一位数字可以变成回文数。例如 1337 → 1331 或 7337；1501 → 1001 或 1551。但 1234、0991、1331 不是。求有多少个 4-digit almost-palindromes。**考点**：组合计数—分类讨论 **解析**：四位回文数形式为 $\overline{abba}$，其中 $a \in \{1, \ldots, 9\}$，$b \in \{0, \ldots, 9\}$。一个四位非回文数 $\overline{d_1 d_2 d_3 d_4}$（$d_1 \neq 0$）要成为 almost-palindrome，必须存在一种改变恰好一位数字使之成为回文数的方法。**分类讨论**：

- **Type A**：$d_1 \neq d_4$ 且 $d_2 = d_3$
- 改变 d_1 为 d_4：得到 $d_4 d_2 d_2 d_4$（回文数）
- 或改变 d_4 为 d_1：得到 $d_1 d_2 d_2 d_1$（回文数）
- 两种方式都可行，保证了 almost-palindrome 的性质
- **Type B**：$d_1 = d_4$ 且 $d_2 \neq d_3$
- 改变 d_2 为 d_3：得到 $d_1 d_3 d_3 d_1$（回文数）
- 或改变 d_3 为 d_2：得到 $d_1 d_2 d_2 d_1$（回文数）

验证非 almost-palindrome 的情况：

- 回文数本身（$d_1 = d_4, d_2 = d_3$）：改任一位都破坏回文性
- $d_1 \neq d_4$ 且 $d_2 \neq d_3$：如 1234，无法通过改一位变成回文

计数：

- **Type A**（$d_1 \neq d_4, d_2 = d_3$）：
- d_1：9 种（1-9）
- d_4：9 种（0-9 中除 d_1）
- $d_2 = d_3$：10 种（0-9）
- 小计：$9 \times 9 \times 10 = 810$
- **Type B**（$d_1 = d_4, d_2 \neq d_3$）：

- $d_1 = d_4$：9 种（1-9）
- d_2：10 种
- d_3：9 种（0-9 中除 d_2）
- 小计：$9 \times 10 \times 9 = 810$

总数：$810 + 810 = \boxed{1620}$

11.17.3 题目 3 (15%)：递推关系求解

题目：已知递推关系 $a_n = c_1 a_{n-1} + c_2 a_{n-2} + c_3$，其中 c_1, c_2, c_3 为未知常数，且 $a_0 = 0, a_1 = 1, a_2 = 4, a_3 = 11, a_4 = 26$。求通项公式。**考点**：常系数线性非齐次递推关系—待定系数法 **解析**：**第一步：确定系数** c_1, c_2, c_3 代入已知值，列方程组：

$$\begin{cases} n=2: \quad 4 = c_1 \cdot 1 + c_2 \cdot 0 + c_3 \quad \Rightarrow \quad c_1 + c_3 = 4 \\ n=3: \quad 11 = c_1 \cdot 4 + c_2 \cdot 1 + c_3 \quad \Rightarrow \quad 4c_1 + c_2 + c_3 = 11 \\ n=4: \quad 26 = c_1 \cdot 11 + c_2 \cdot 4 + c_3 \quad \Rightarrow \quad 11c_1 + 4c_2 + c_3 = 26 \end{cases}$$

由第一式得 $c_3 = 4 - c_1$，代入第二式：

$$4c_1 + c_2 + 4 - c_1 = 11 \ \Rightarrow \ c_2 = 7 - 3c_1$$

代入第三式：

$$11c_1 + 4(7 - 3c_1) + (4 - c_1) = 26$$

$$11c_1 + 28 - 12c_1 + 4 - c_1 = 26$$

$$32 - 2c_1 = 26 \ \Rightarrow \ c_1 = 3$$

从而 $c_3 = 1$，$c_2 = -2$。故递推关系为：

$$a_n = 3a_{n-1} - 2a_{n-2} + 1$$

第二步：求解齐次部分特征方程：$r^2 - 3r + 2 = 0 \ \Rightarrow \ (r-1)(r-2) = 0$，特征根 $r_1 = 1, r_2 = 2$。齐次通解：

$$a_n^{(h)} = A \cdot 1^n + B \cdot 2^n = A + B \cdot 2^n$$

第三步：求特解非齐次项为常数 $1 = 1 \cdot 1^n$。由于 1 是特征根（单根），设特解形式 $a_n^{(p)} = Cn$。代入递推：

$$Cn = 3C(n-1) - 2C(n-2) + 1$$

$$Cn = 3Cn - 3C - 2Cn + 4C + 1$$

$$Cn = Cn + C + 1$$

$$C = -1$$

故 $a_n^{(p)} = -n$。**第四步：全解**

$$a_n = A + B \cdot 2^n - n$$

利用初始条件：

$$\begin{cases} a_0 = A + B = 0 \\ a_1 = A + 2B - 1 = 1 \ \Rightarrow \ A + 2B = 2 \end{cases}$$

解得 $B = 2, A = -2$。**最终通项**：

$$\boxed{a_n = 2^{n+1} - n - 2}$$

验证：

- $a_0 = 2 - 2 = 0$
- $a_1 = 4 - 1 - 2 = 1$
- $a_2 = 8 - 2 - 2 = 4$
- $a_3 = 16 - 3 - 2 = 11$
- $a_4 = 32 - 4 - 2 = 26$

11.17.4 题目 4 (25%)：网格图 $G_{m,n}$ 的性质

题目：网格图 $G_{m,n}$ 是 $m \times n$ 的矩形网格图（$m \le n$）。讨论以下性质：**(a)** 哪些 m, n 有欧拉回路？**(b)** 哪些 m, n 有欧拉通路但没有欧拉回路？**(c)** 哪些 m, n 有哈密顿回路？**(d)** 哪些 m, n 有哈密顿通路但没有哈密顿回路？**(e)** $G_{1,2}$ 有多少棵生成树？**考点**：图论—欧拉图、哈密顿图、生成树

解析

顶点度分析 $G_{m,n}$ 有 mn 个顶点。

- **角点（4 个）**：度数 2
- **边点（非角）**：
- 上边非角：$n-2$ 个，度数 3
- 下边非角：$n-2$ 个，度数 3
- 左边非角：$m-2$ 个，度数 3
- 右边非角：$m-2$ 个，度数 3
- **内点**：$(m-2)(n-2)$ 个，度数 4

奇度顶点总数：$2(n-2) + 2(m-2) = 2(m+n-4)$（当 $m, n \ge 2$ 时）。对于 $m = 1$ 的情况，$G_{1,n}$ 是 n 个顶点的路径图：两端点度 1（奇），内部点度 2（偶），共 2 个奇度顶点。

(a) 欧拉回路 欧拉回路存在当且仅当所有顶点度数为偶数。

- $(m, n) = (1, 1)$：单顶点，平凡成立
- $(m, n) = (2, 2)$：四个顶点度数均为 2，均为偶数

其他情况均有奇度顶点。**答案**：欧拉回路存在当且仅当 $(m, n) = (1, 1)$ 或 $(2, 2)$（$m \le n$ 下即为这两个解）。

(b) 欧拉通路但没有欧拉回路 恰有 2 个奇度顶点。

- $m = 1$：$G_{1,n}$（$n \ge 2$）恰有 2 个奇度端点
- 即 $(1, n)$ 对任意 $n \ge 2$
- $m, n \ge 2$：需要 $2(m + n - 4) = 2 \ \Rightarrow \ m + n = 5$，且 $m \le n$，$m \ge 2$
- 得 $(m, n) = (2, 3)$

答案：欧拉通路存在但没有回路当且仅当 $(m, n) = (1, n)$ $(n \ge 2)$ 或 $(2, 3)$。

(c) 哈密顿回路 $G_{m,n}$ 是二分图。若 $m, n \ge 2$，两部顶点数分别为 $\lceil mn/2 \rceil$ 和 $\lfloor mn/2 \rfloor$。哈密顿回路需要两部顶点数相等。

- $(1, 1)$：平凡成立
- $m, n \ge 2$：需要 mn 为偶数，即 m, n 至少一个为偶数
- 蛇形遍历可构造回路

答案：哈密顿回路存在当且仅当 $(m, n) = (1, 1)$ 或 $m, n \ge 2$ 且 m, n 至少一个为偶数。

(d) 哈密顿通路但没有哈密顿回路

- $m = 1, n \ge 2$：路径本身就是哈密顿通路，但无回路（端点度 1）
- $m, n \ge 2$ 且 m, n 均为奇数：两部顶点数差 1，有哈密顿通路但无回路

答案：哈密顿通路存在但没有回路当且仅当 $(m, n) = (1, n)$ $(n \ge 2)$ 或 $m, n \ge 2$ 且 m, n 均为奇数。

(e) $G_{1,2}$ 的生成树数量 $G_{1,2}$ 是两个顶点一条边，本身就是树。**答案**：$\boxed{1}$ 棵生成树。

11.17.5 题目 5 (20%)：正多面体与图论

题目：**(a)** 判断正四面体、正方体、正八面体是否为平面图。若是，画出示意图。**(b)** 求正四面体、正方体、正八面体的色数。**(c)** 用欧拉定理证明凸正多面体只有 5 种，列出它们的面数、顶点数、边数。**考点**：平面图、着色、欧拉公式

解析

(a) 平面性 三者都是平面图（均可画在平面上无边交叉）。

- **正四面体** $(= K_4)$：三角锥展开为平面
- **正方体**：展开为 4×4 的平面图（十字展开）
- **正八面体**：中心一个四边形，上下各接两个三角面

（此处略去绘图，考场上展开绘制即可）

(b) 色数

- **正四面体** (K_4)：$\chi = 4$（完全图 K_4 需 4 色）
- **正方体**（三维立方体骨架图）：为二分图，$\chi = 2$
- **正八面体**：$\chi = 3$（可 3-着色：顶底同色、中间四边形对角同色）

(c) 证明只有 5 种正多面体 设每个面为正 p 边形（$p \ge 3$），每个顶点关联 q 条棱（$q \ge 3$）。计数关系：

- 棱数 E 满足 $pF = 2E$（每个面 p 条边，每条棱被 2 个面共享）
- 顶点数 V 满足 $qV = 2E$（每个顶点 q 条棱，每条棱关联 2 顶点）

代入欧拉公式 $V-E+F=2$：

$$\frac{2E}{q}-E+\frac{2E}{p}=2$$

$$E\left(\frac{2}{q}-1+\frac{2}{p}\right)=2$$

$$E=\frac{2}{\frac{2}{q}+\frac{2}{p}-1}=\frac{2pq}{2p+2q-pq}$$

由于 $E>0$，分母必须为正：

$$2p+2q-pq>0 \;\Rightarrow\; (p-2)(q-2)<4$$

$p,q\geq 3$ 为整数，故可能的 (p,q) 组合：

(p,q)	名称	V	E	F
$(3,3)$	正四面体 (Tetrahedron)	4	6	4
$(3,4)$	正方体 (Cube)	8	12	6
$(4,3)$	正八面体 (Octahedron)	6	12	8
$(3,5)$	正十二面体 (Dodecahedron)	20	30	12
$(5,3)$	正二十面体 (Icosahedron)	12	30	20

证毕。这 5 种就是全部凸正多面体（柏拉图立体）。

11.17.6 题目 6 (10%)：斐波那契数的 Zeckendorf 表示

题目：证明每个大于 2 的正整数 n 都可以表示为不同斐波那契数之和。**考点**：斐波那契数、强归纳法、Zeckendorf 定理 **注**：完整 Zeckendorf 定理还要求表示中不含连续斐波那契数且表示唯一，本题仅要求证明存在性。**解析**：**定义**：斐波那契数列 $F_1=1,F_2=1,F_3=2,F_4=3,F_5=5,\ldots$，即 $F_1=F_2=1$，$F_k=F_{k-1}+F_{k-2}$（$k\geq 3$）。**证明（强归纳法）**：**归纳假设**：对任意 m 满足 $2<m<n$，m 可表示为不同斐波那契数之和。**基始**：$n=3$ 时，$3=1+2=F_2+F_3$（或 $3=F_4$），成立。**归纳步骤**：对 $n>3$，取最大的斐波那契数 $F_k\leq n$。若 $n=F_k$，则直接表示为 F_k 本身，成立。若 $n>F_k$，令 $m=n-F_k$。由于 $F_k\leq n<F_{k+1}=F_k+F_{k-1}$，有：

$$m=n-F_k<F_{k-1}$$

故 $m<F_{k-1}$，且 $m>0$（因为 $n>F_k$）。

- 若 $m>2$，由归纳假设 m 可表示为不同斐波那契数之和，且这些数均 $\leq m<F_{k-1}$，故与 F_k 不重复且不连续
- 若 $m\leq 2$，即 $m=1$ 或 2，两者都是斐波那契数且 $<F_{k-1}$，直接加入和即可

因此 $n=F_k+m$ 可表示为不同斐波那契数之和。由强归纳法，命题对所有 $n>2$ 成立。□

11.18 DMA Quiz 4 题解—2023-2024 春夏 (郑文庭班)

11.18.1 题目 1 (40%，每题 5%)：填空题

1(a) 等值命题变换

题目：写出与 $(p\wedge\neg q)$ 等价的命题，只使用 $p,q,\neg$ 和联结词 $\vee$。**考点**：命题逻辑—De Morgan 律 **解析**：利用 De Morgan 律：$\neg(\neg p\vee q)\equiv p\wedge\neg q$。

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

验证真值表：

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

答案：$\boxed{\neg(\neg p\vee q)}$

1(b) 否定词内移

题目：写出 $\exists x\exists yP(x,y)\wedge\forall x\forall yQ(x,y)$ 的否定，使否定符号紧邻谓词。**考点**：谓词逻辑—量词否定 **解析**：

$$\begin{aligned}&\neg[\exists x\exists yP(x,y)\wedge\forall x\forall yQ(x,y)]\\&\equiv\neg\exists x\exists yP(x,y)\vee\neg\forall x\forall yQ(x,y)\quad\text{(De Morgan)}\\&\equiv\forall x\forall y\neg P(x,y)\vee\exists x\exists y\neg Q(x,y)\end{aligned}$$

答案：$\boxed{\forall x\forall y\neg P(x,y)\vee\exists x\exists y\neg Q(x,y)}$

1(c) 平面图区域数

题目：G 是连通的平面图，有 20 个顶点，每个顶点度数为 3，求 G 的区域数。**考点**：平面图—握手定理 + 欧拉公式 **解析**：握手定理：$2E=\sum\deg(v)=20\times 3=60\;\Rightarrow\;E=30$ 欧拉公式（连通平面图）：$V-E+F=2$

$$20-30+F=2\;\Rightarrow\;F=12$$

答案：$\boxed{12}$ 个区域。

1(d) 有理数坐标点的个数

题目：三维直角坐标系中，三个坐标均为有理数的点有多少个？**考点**：集合论—可数性 **解析**：有理数集 $\mathbb{Q}$ 是可数无穷（$\aleph_0$）。$\mathbb{Q}^3=\mathbb{Q}\times\mathbb{Q}\times\mathbb{Q}$ 是可数无穷个可数集的笛卡尔积，仍为可数无穷。**答案**：$\boxed{\text{可数无穷多}}$（$\aleph_0$，与自然数集等势）。

1(e) 最小支配集

题目：下图为 12 个顶点的图（网格图加额外边 i-l），求最小支配集的顶点数。**考点**：图论—支配集 **解析**：图的结构为 3×4 网格（顶点 a-l），外加一条边 i-l（左下到右下）。顶点度分布：

- a,d：度 2
- b,c,e,h,j,k：度 3
- i,l：度 3（i-l 额外边使角点度数增加）
- f,g：度 4

最小支配集可以取 $\{b,e,h,k\}$：

- b 支配：a,b,c,f
- e 支配：a,e,i
- h 支配：d,h,l
- k 支配：g,k,j

12 个顶点全部被覆盖。可验证 3 个顶点无法支配全部 12 个顶点。**答案**：$\boxed{4}$ 个顶点。

1(f) 哈密顿回路

题目：判断上图是否存在哈密顿回路。若存在，给出一个；若不存在，说明理由。**考点**：图论—哈密顿回路 **解析**：存在哈密顿回路。例如：

$$a\to b\to f\to j\to k\to g\to c\to d\to h\to l\to i\to e\to a$$

验证各边存在：a-b, b-f, f-j, j-k, k-g, g-c, c-d, d-h, h-l, l-i（额外边），i-e, e-a。全部 12 个顶点各出现一次，且首尾相接。**答案**：存在，如上所示。

1(g) Dijkstra 算法时间复杂度

题目：Dijkstra 算法在 n 个顶点的带权图中求最短路径的最坏情况时间复杂度是多少？**考点**：算法分析—Dijkstra 算法 **解析**：

- 使用**简单数组**实现的未优化版本：每次选最小距离顶点需 $O(n)$，共 n 次，总 $O(n^2)$
- 使用**二叉堆**优化：$O((n+E)\log n)$，最坏 $O(n^2\log n)$（稠密图）
- 使用**斐波那契堆**优化：$O(E+n\log n)$

教材一般讲授简单数组实现。**答案**：$\boxed{O(n^2)}$（简单数组实现）。

1(h) 有向图路径计数

题目：下图为 4 个顶点的有向图，求图中长度为 _ 的不同路径数（允许环路）。**考点**：图论—邻接矩阵、路径计数 **解析**：邻接矩阵 M：

$$M=\begin{pmatrix}0&1&1&0\\1&0&0&1\\0&0&0&1\\1&0&1&0\end{pmatrix}$$

路径数由 M^n 的全体元素之和给出，其中 n 是路径长度。（注：原题中路径长度数值在转录中缺失。考试时根据给定长度 L，计算 M^L 中所有元素之和即可。）

11.18.2 题目 2 (6%)：中国剩余定理

题目：求三个连续的正整数，依次能被 5, 7, 11 整除。**考点**：初等数论—中国剩余定理 (CRT) **解析**：设这三个连续整数为 $n,n+1,n+2$：

$$\begin{cases}n\equiv 0\pmod 5\\n\equiv 6\pmod 7\quad(\text{即}n+1\equiv 0\pmod 7)\\n\equiv 9\pmod{11}\quad(\text{即}n+2\equiv 0\pmod{11})\end{cases}$$

由中国剩余定理求解：由 $n\equiv 0\pmod 5$ 设 $n=5k$，代入第二式：

$$5k\equiv 6\pmod 7$$

$5\pmod 7$ 的逆元为 3（因 $5\times 3=15\equiv 1\pmod 7$）：

$$k\equiv 6\times 3=18\equiv 4\pmod 7\;\Rightarrow\;k=7m+4$$

$$n = 5(7m+4) = 35m + 20$$

代入第三式 $n \equiv 9 \pmod{11}$：

$$35m + 20 \equiv 9 \pmod{11}$$

$35 \equiv 2 \pmod{11}$，$20 \equiv 9 \pmod{11}$：

$$2m + 9 \equiv 9 \pmod{11} \Rightarrow 2m \equiv 0 \pmod{11}$$

$\gcd(2, 11) = 1$，所以 $m \equiv 0 \pmod{11}$，即 $m = 11t$。

$$n = 35 \times 11t + 20 = 385t + 20$$

答案：

$$\boxed{n = 385t + 20, \quad t \geq 0,\ t \in \mathbb{Z}}$$

即 $20, 21, 22$；$405, 406, 407$；...

11.18.3 题目 3 (30%，每题 5%)：扑克牌组合计数

题目：一副 52 张牌（4 花色 × 13 点数），回答以下问题。

(a) 四条 (Four of a kind, xxxx y)

考点：组合计数—基本乘法原理 **解析**：

- 选四张同点数的点数：$\binom{13}{1} = 13$
- 选该点数的 4 张牌（全取）：$\binom{4}{4} = 1$
- 选第五张的点数：$\binom{12}{1} = 12$
- 选第五张的花色：$\binom{4}{1} = 4$

答案：$13 \times 12 \times 4 = \boxed{624}$

(b) 两对 (Two pairs, xx yy z)

题目：两个对子（同点不同花），不含三条。**考点**：组合计数 **解析**：

- 选两个对子的点数：$\binom{13}{2} = 78$
- 对子 1 选花色：$\binom{4}{2} = 6$
- 对子 2 选花色：$\binom{4}{2} = 6$
- 选第五张的点数（不同于两个对子）：$\binom{11}{1} = 11$
- 选第五张的花色：$\binom{4}{1} = 4$

答案：$78 \times 6 \times 6 \times 11 \times 4 = \boxed{123,552}$

(c) 鸽笼原理—保证有对子

题目：至少取多少张牌才能保证其中必有一个对子（同点数）？**考点**：鸽笼原理（抽屉原理）**解析**：共有 13 种点数。最坏情况下取到的 13 张牌点数各不相同（每种点数取 1 张）。再取第 14 张必与某张重复。**答案**：$\boxed{14}$ 张。

(d) 鸽笼原理—保证有顺子

题目：至少取多少张牌才能保证其中必有顺子（5 张点数连续，A-2-3-4-5 和 10-J-Q-K-A 都算，花色不限）？**考点**：鸽笼原理 **解析**：共有 10 种可能的顺子：$A-2-3-4-5,\ 2-3-4-5-6,\ \ldots,\ 10-J-Q-K-A$。要避免任何顺子，不能拥有 5 张连续点数的牌。最坏情况是取所有不构成 5 连续的牌：

- 取点数 $A, 2, 3, 4$ 各 4 张（共 16 张）
- 取点数 $6, 7, 8, 9$ 各 4 张（共 16 张）
- 取点数 J, Q, K 各 4 张（共 12 张）

共计 $16 + 16 + 12 = 44$ 张，仍无顺子。再取任意一张（点数为 5 或 10 中的某张），必构成顺子。**答案**：$\boxed{45}$ 张。

(e) 花色不可分辨的三张牌组合

题目：不考虑花色差异（同点数不同花色的牌视为相同），取 3 张牌有多少种不同方式？**考点**：组合计数—多重集组合 **解析**：相当于从 13 种点数中选 3 张（可重复），即多重集组合数：

$$\binom{13+3-1}{3} = \binom{15}{3} = 455$$

答案：$\boxed{455}$ 种。

(f) 4×13 网格中的完全匹配

题目：52 张牌排成 4 行 13 列的网格（每张牌一个格子）。证明：从每列各选一张牌，总能使选出的 13 张牌点数各不相同。**考点**：Hall 婚姻定理—完全匹配 **解析**：构造二分图 $G = (C \cup V, E)$，其中：

- 左部 C：13 列，每列是一个顶点
- 右部 V：13 种点数
- 边 $c_i \to v_j$：第 i 列中存在点数为 j 的牌

需要证明 G 有一个完美匹配，即存在从列到点数的双射使每列选出不同点数。**应用 Hall 定理**：对任意 k 列（$1 \leq k \leq 13$），这 k 列共有 $4k$ 张牌。由于每种点数至多出现 4 次（整副牌共有 4 张每种点数），这 $4k$ 张牌至少覆盖 k 种不同的点数（否则如果只覆盖 $\leq k-1$ 种点数，则最多只有 $4(k-1)$ 张牌，矛盾于共有 $4k$ 张）。因此对任意 $k \leq 13$ 列，相邻的点数种类 $\geq k$，Hall 条件成立。由 Hall 定理，存在完美匹配，即存在从每列选一张牌使得点数互不相同。□

11.18.4 题目 4 (12%)：生成函数求解递推

题目：使用生成函数求解递推关系 $a_k = 5a_{k-1} - 6a_{k-2}$，初始条件 $a_0 = 6, a_1 = 30$。**考点**：生成函数—常系数线性齐次递推 **解析**：设 $A(x) = \sum_{k=0}^{\infty} a_k x^k$。

$$\sum_{k\geq 2} a_k x^k = 5\sum_{k\geq 2} a_{k-1}x^k - 6\sum_{k\geq 2} a_{k-2}x^k$$

$$A(x) - a_0 - a_1 x = 5x(A(x) - a_0) - 6x^2 A(x)$$

$$A(x) - 6 - 30x = 5xA(x) - 30x - 6x^2 A(x)$$

$$A(x)(1 - 5x + 6x^2) = 6$$

$$A(x) = \frac{6}{(1-2x)(1-3x)}$$

部分分式展开：

$$\frac{6}{(1-2x)(1-3x)} = \frac{A}{1-2x} + \frac{B}{1-3x}$$

$$6 = A(1-3x) + B(1-2x)$$

比较系数：

$$\begin{cases} A + B = 6 \\ -3A - 2B = 0 \end{cases} \Rightarrow A = -12,\ B = 18$$

因此：

$$A(x) = -\frac{12}{1-2x} + \frac{18}{1-3x} = -12\sum_{k\geq 0} 2^k x^k + 18\sum_{k\geq 0} 3^k x^k$$

$$a_k = -12 \cdot 2^k + 18 \cdot 3^k$$

答案：$\boxed{a_k = 18 \cdot 3^k - 12 \cdot 2^k}$ **验证**：

- $a_0 = 18 - 12 = 6$
- $a_1 = 54 - 24 = 30$
- $a_2 = 162 - 48 = 114$，代入递推：$5 \times 30 - 6 \times 6 = 150 - 36 = 114$

11.18.5 题目 5 (12%)：红绿点不相交连线（归纳法）

题目：平面上有 n 个红点和 n 个绿点，任意三点不共线。用归纳法证明：这些 $2n$ 个点可以两两配对连成 n 条线段，每条线段连接一个红点和一个绿点，且线段互不相交。**考点**：数学归纳法—组合几何 **解析**：**归纳基础（$n = 1$）**：1 红 1 绿两点相连，唯一线段，无相交问题。成立。**归纳步骤**：假设对 $n-1$ 成立（$n \geq 2$），证明对 n 成立。给定 n 红 n 绿共 $2n$ 个点，任意三点不共线。考虑这些点的凸包（包含所有点的最小凸多边形）。**情况 1**：凸包上既有红点也有绿点。则凸包上必存在相邻的两个顶点颜色不同（因为沿凸包走，颜色变化至少一次）。设红点 R 和绿点 G 在凸包上相邻。连接 RG，该线段在凸包边界上，不穿过任何其他点。移除 R 和 G，剩余 $n-1$ 红 $n-1$ 绿，由归纳假设可连成 $n-1$ 条不相交线段，且均不会与凸包边界上的 RG 相交。**情况 2**：凸包上全是同一颜色（全红或全绿）。不妨设凸包全为红点。取一个红点 R 在凸包上。由于绿点都在凸包内部，任意连接 R 和某个绿点 G 得到的线段 RG 不会接触任何其他凸包顶点。选取绿点 G 使得直线 RG 两侧各有相同数量的红绿点对。具体构造：对 R 固定，将绿点按 R 出发的极角排序，依次检查过 R 和 G_i 的直线两侧的红绿点数。由于总红点数 = 总绿点数，必存在某个 G 使两侧红绿数分别相等。连接 RG，两侧各为 $n-1$ 和 n' 对... 但更简洁的方法如下：**标准证法（旋转直线法）**：取任意一个红点 R。考虑一条过 R 且不穿过任何其他点的直线 L。将 L 以 R 为中心连续旋转 $180°$。记 $f(\theta)$ 为直线 L_θ 某一侧中红点数减去绿点数。$f(0) = -f(180°)$。由于 f 是整数阶梯函数且每次变化 ± 1，存在 θ 使 $f(\theta) = 0$。此时 L_θ 上必恰有一个绿点 G（否则两侧红绿数不可能相等）。连接 RG，两侧红绿点数各自相等，由归纳假设可分别处理。因此，对任意 n 均可用归纳法构造出 n 条不相交的红-绿线段。□

11.19 DMA 期末回忆卷题解—2021-2022 春夏 (郑文庭班)

说明：本卷为回忆版，题目内容不完整，部分表述有缺失。以下根据现有回忆内容还原并解答，标注了不确定之处。

11.19.1 一、判断题

1. 平面图的点度数之和 = 图的度数之和

考点：握手定理 **解析**：对任意图（包括平面图），握手定理 $\sum \deg(v) = 2E$ 恒成立。图的度数之和就是 $2E$，也就是所有顶点度数之和。该命题 **真**（trivially true）。**答案**：$\boxed{\text{True}}$

2. 三维单位立方体内包含的点的个数与二维单位正方形内包含的点的个数比较

考点：集合论—无穷集合的势 **解析**：三维单位立方体 $[0,1]^3$ 和二维单位正方形 $[0,1]^2$ 都是**不可数无穷**，势均为 $\mathfrak{c} = |\mathbb{R}|$（连续统基数）。两者同样多。（回忆题可能有几种变体，若有具体问法如" 大于/小于/等于"，则为等于。）**答案**：$\boxed{\text{相等}}$（势相同，均为 $\mathfrak{c}$）。

3. 对称差恒等式：$(A \cap B) \oplus C = (A \oplus C) \cap (B \oplus C)$

考点：集合论—对称差运算 **解析**：用真值表（成员关系表）验证：

A	B	C	$A \cap B$	$(A \cap B) \oplus C$	$A \oplus C$	$B \oplus C$	$(A \oplus C) \cap (B \oplus C)$
0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	1
0	1	0	0	0	0	1	0
0	1	1	0	1	1	0	0
1	0	0	0	0	1	0	0
1	0	1	0	1	0	1	0
1	1	0	1	1	1	1	1
1	1	1	1	0	0	0	0

两列不完全一致（在第 3, 6 行不同），恒等式不成立。**答案**：$\boxed{\text{False}}$

11.19.2 二、填空题

1. 树的内部节点度数

题目：一棵树有 7 个叶子，3 个内点度数为 3，其余内点度数为 4。求度数为 4 的内点个数。**考点**：树的性质—握手定理 **解析**：设度数为 4 的内点有 x 个。

- 顶点数 $V = 7 + 3 + x = 10 + x$
- 边数 $E = V - 1 = 9 + x$
- 度数之和 $= 7 \times 1 + 3 \times 3 + 4x = 16 + 4x$

握手定理：$2E = \sum \deg(v)$

$$2(9 + x) = 16 + 4x \;\Rightarrow\; 18 + 2x = 16 + 4x \;\Rightarrow\; x = 1$$

答案：$\boxed{1}$ 个。

2. 3 元集上的偏序关系个数

题目：$n = 3$ 时，一个 3 元集上有多少种不同的偏序关系？**考点**：偏序集计数 **解析**：n 元集上偏序关系的个数是 Dedekind 数。$n = 3$ 时为 19 种。（按：可枚举所有可能的 Hasse 图结构：1 个反链（全无序）、3 种含 1 个可比对、3 种链长 2 加孤立点、3 种 V 形、3 种 Λ 形、3 种全序的线性扩展变形等，总和 19。）**答案**：$\boxed{19}$

3. 集合计数

题目：$A \cup B \cup C = \{1, 2, 3, 4, 5\}$，$A \cap B = \{1, 2\}$，求 (A, B, C) 的可能组合数。**考点**：组合计数—包含排斥 **解析**：元素 $1, 2$ 必在 $A \cap B$ 中，可能亦在 C 中：各 2 种选择（在 C 或不在），共 $2 \times 2 = 4$ 种。元素 $3, 4, 5$ 必须至少出现在 A, B, C 之一中，且不能同时出现在 A 和 B 中（否则会扩大 $A \cap B$）。对每个元素，合法状态为以下 5 种：

$$\text{仅在}A,\ \text{仅在}B,\ \text{仅在}C,$$
$$\text{在}A\ \text{和}C,\ \text{在}B\ \text{和}C$$

（不允许在 $A \cap B$ 中，也不允许全不在）每个元素 5 种选择，3 个元素：$5^3 = 125$ 种。总数：$4 \times 125 = 500$。**答案**：$\boxed{500}$ 种。

4. 生成函数求数列

题目：$G(x) = \dfrac{1}{1 - x - x^2} = \sum_{n \geq 0} a_n x^n$，求 a_n。**考点**：生成函数—斐波那契数列 **解析**：$G(x) = \dfrac{1}{1 - x - x^2}$ 是斐波那契数列的生成函数变体。展开：$\dfrac{1}{1 - x - x^2} = 1 + x + 2x^2 + 3x^3 + 5x^4 + 8x^5 + 13x^6 + \cdots$ 数列符合斐波那契递推 $a_n = a_{n-1} + a_{n-2}$，初始 $a_0 = 1, a_1 = 1$。$a_n = F_{n+2}$，其中 $F_1 = F_2 = 1$ 为标准斐波那契数列。具体通项（Binet 公式）：

$$a_n = \frac{1}{\sqrt{5}} \left(\varphi^{n+2} - \psi^{n+2}\right), \quad \varphi = \frac{1 + \sqrt{5}}{2},\ \psi = \frac{1 - \sqrt{5}}{2}$$

答案：$\boxed{a_n = F_{n+2}}$（$F_1 = F_2 = 1$ 的斐波那契数）。

5. 最长最短路径

题目：求从源点到各点的最短路中最长的一条。**考点**：图论—Dijkstra 算法、离心率 **解析**：该问题即求源点 s 的**离心率**（eccentricity）：

$$e(s) = \max_{v \in V} \{\text{dist}(s, v)\}$$

对给定图运行 Dijkstra 算法，得到 s 到所有顶点的最短距离，取最大值即可。（注：回忆卷未提供具体图结构，此处只给出一般方法。）

6. 生成函数的变换

题目：设 $G(x) = \sum_{n \geq 0} a_n x^n$，求数列 $a_0, 2a_1, 3a_2, \ldots$ 的生成函数。**考点**：生成函数运算—导数 **解析**：$b_n = (n + 1)a_n$ 的生成函数：

$$\sum_{n \geq 0} (n + 1) a_n x^n = \sum_{n \geq 0} n a_n x^n + \sum_{n \geq 0} a_n x^n = x G'(x) + G(x)$$

答案：$\boxed{G(x) + xG'(x)}$

7. 非降序列计数

题目：给定序列 $C_5 C_5 C_3 C_2 C_1$，其中 $C_i \in \{5, 6, 7, 8\}$，且若 $m < n$ 则 $C_m \leq C_n$。求可能的序列个数。**考点**：组合计数—多重集组合 **解析**：从 4 个元素 $\{5, 6, 7, 8\}$ 中选 5 个构成非降序列，等价于 5 个相同球放入 4 个盒子的组合数（允许空盒），即多重集组合：

$$\binom{5 + 4 - 1}{5} = \binom{8}{5} = 56$$

答案：$\boxed{56}$

8. 包含所有数字的字符串

题目：（推测）求 n 位十进制字符串中包含全部 10 个数字的个数。**考点**：组合计数—包含排斥原理 **解析**：用包含排斥原理：

$$\sum_{i=0}^{10} (-1)^i \binom{10}{i} (10 - i)^n$$

（首位可为 0 时；若首位不可为 0 需额外处理。）

9. 字符串前缀/后缀条件

题目：求满足下列至少一个条件的字符串个数：

- 以"66" 开头
- 以"5" 开头
- 以"88" 结尾

考点：组合计数—包含排斥原理 **解析**：用包含排斥原理计算三个集合的并集。设 $A =$ 以"66" 开头，$B =$ 以"5" 开头，$C =$ 以"88" 结尾。字符串总长为 n。

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

（具体数值取决于字符串长度和字符集，考场上根据给定长度代入即可。）

11.19.3 三、关系与图论

1. 关系传递闭包

题目：给定关系 $R = \{(C, A), (A, B), (B, D), (D, E), (E, B)\}$，求相关性质或传递闭包。**考点**：关系—传递闭包 **解析**：关系图：$C \to A \to B \to D \to E \to B$（有环 $B \to D \to E \to B$）。传递闭包 R^+ 为所有可达对：

$$R^+ = \{(C, A), (C, B), (C, D), (C, E), (A, B), (A, D), (A, E), (B, B), (B, D), (B, E), (D, B), (D, D), (D, E), (E, B), (E, D), (E, E)\}$$

（具体题目要求可能有所不同，此为一般求解方法。）

2. 递推关系求通项

题目：（内容不全）给定递推关系和部分初始条件，求通项。**考点**：常系数线性递推 **解析**：回忆版中出现"12 + 4n"、"−n−1"、"16a"、"7a" 等片段，推测涉及形如 $a_n = 16a_{n-1} - 7a_{n-2} + \cdots$ 的递推。**一般方法**：

1. 写出特征方程 $r^2 - c_1 r - c_2 = 0$
2. 解出特征根
3. 齐次通解 $= Ar_1^n + Br_2^n$
4. 非齐次项代入设特解（常数项视为 1^n，多项式视为 n 的多项式）
5. 利用初始条件定系数

3. 有向图的强连通分量与传递闭包

题目：给一个有向图，求强连通分量（SCC）和传递闭包。**考点**：有向图—Kosaraju / Tarjan 算法、Warshall 算法 **解析**：**强连通分量**：可使用 Kosaraju 算法或 Tarjan 算法，运行 DFS 找到极大强连通子图。**传递闭包**：可用 Warshall 算法（Floyd-Warshall 变体）计算，$O(n^3)$。

4. 最小生成树（字典序）

题目：从 A 点出发，找使各点尽可能连通的树/森林，边权重按字典序比较。**考点**：最小生成树—字典序最小生成树 **解析**：当边权按字典序定义时（如边标记为字符串），可用 Kruskal 算法或 Prim 算法，始终选择当前字典序最小的合法边。标准 MST 算法仍适用，只需将比较运算改为字典序比较。

11.19.4 四、进阶问题

1. 球合并问题（归纳法）

题目：将 2^k 个球放到若干包中。可通过以下规则合并：

- 规则 1：若两包球数相同，直接合并（数量翻倍，包数减一）
- 规则 2：若两包球数分别为 $m > n$，则变为 $m - n$ 和 $2n$

用归纳法证明：存在算法可将所有球合并到一个包中。**考点**：数学归纳法 **解析**：对 k 做归纳。**归纳基础**：$k = 0$ 时只有 1 个球，已经在一个包中。成立。**归纳步骤**：假设对 2^{k-1} 个球结论成立。考虑 2^k 个球的任意分布。**策略**：先将所有包的大小变为偶数，然后整体除以 2 降阶。

- 若所有包都是偶数大小：全部除以 2，得到 2^{k-1} 个球的分布，由归纳假设可合并成一包，再乘以 2 还原。
- 若有奇数大小的包：由于总数 2^k 为偶数，奇数大小的包必有偶数个。将它们两两配对：
- 若两奇数相同：规则 1 合并为偶数
- 若两奇数不同：规则 2 得 $(m - n)$ 偶和 $2n$ 偶
- 处理后所有包均为偶数，回到上一步

重复此过程最终所有球合并到一包中。□

2. 超立方体 Q_n

题目：Q_n 是 n 维超立方体图，顶点为所有长度 n 的 01 串，边连接恰有一位不同的串。回答：**(a)** 有多少个顶点？**(b)** 有多少条边？**(c)** 是否有欧拉回路/通路？为什么？**(d)** 是否有哈密顿回路/通路？为什么？**(e)** 最少着色数是多少？为什么？**(f)** 是否为平面图？为什么？**考点**：超立方体图的性质 **解析**：**(a) 顶点数**：2^n（所有 n 位 01 串）。**(b) 边数**：每个顶点度为 n，总度数和 $= n \cdot 2^n$，故边数 $= \dfrac{n \cdot 2^n}{2} = n \cdot 2^{n-1}$。

(c) 欧拉回路/通路：

- Q_n 是 n-正则图（所有顶点度数均为 n）。
- **欧拉回路**：需要所有顶点度数为偶数，即 n 为偶数。
- **欧拉通路（无回路）**：需要恰 2 个奇度顶点。n 为奇数时所有 2^n 个顶点度数为奇（$\neq 2$ 对 $n > 1$），故无欧拉通路。
- Q_1 恰 2 个奇度顶点，有欧拉通路但无回路。

n	欧拉回路	欧拉通路 (无回路)
$n = 1$	无	有
n 偶且 $n \geq 2$	有	—
n 奇且 $n \geq 3$	无	无

(d) 哈密顿回路/通路：

- Q_n 对所有 $n \geq 2$ 存在哈密顿回路（Gray code 构造：$n = 2$ 为正方形，$n \geq 3$ 归纳构造）。
- Q_n 对所有 $n \geq 1$ 存在哈密顿通路。

(e) 色数：Q_n 是二分图（按串中 1 的个数的奇偶性划分），故 $\chi(Q_n) = 2$（$n \geq 1$）。**(f) 平面性**：Q_1（边）、Q_2（正方形）、Q_3（立方体展开）是平面图。Q_4 及以上含有 $K_{3,3}$ 或 K_5 的细分，非平面。

3. 和差整除子集

题目：$S = \{x \mid 1 \leq x \leq 2022,\ x \in \mathbb{Z}\}$。求 S 的最大子集 T，满足对任意不同 $a, b \in T$，$a + b$ 不能被 $|a - b|$ 整除。**考点**：数论—整除性、构造法 **解析**：**条件分析**：对 $a < b$，设 $d = b - a$。则 $a + b = 2a + d$。$d \mid (a + b)$ 当且仅当 $d \mid 2a$。即：$b - a \mid 2a$ 是禁止条件。**构造**：取 $T = \{1, 4, 7, 10, \ldots, 2020\}$，即所有 $\equiv 1 \pmod 3$ 的数。**验证**：对任意 $a = 1 + 3i, b = 1 + 3j$（$j > i$），$d = 3(j - i)$。计算 $2a = 2 + 6i$。$3(j - i) \mid (2 + 6i)$ 要求 $3 \mid (2 + 6i)$，但 $2 + 6i \equiv 2 \pmod 3$，不可能。因此条件满足。**大小**：从 1 到 2022 中 $\equiv 1 \pmod 3$ 的数共有 $\lceil 2022/3 \rceil = 674$ 个。**猜想最优性**：贪心法从 1 开始构造，每次添加不冲突的最小数，恰好得到该集合。这暗示 674 是最优值。**答案**：$|T|_{\max} = \boxed{674}$（可取所有 $\equiv 1 \pmod 3$ 的数）。

4. 球队分组计数

题目：n 名球员，从中选 k 人（k 为偶数），将 k 人分为红/蓝两队（一队可为空），求和所有偶数 k 的分组方式总数。**考点**：组合计数—二项式定理 **解析**：对固定的偶数 k：

- 选人：$\binom{n}{k}$
- 分两队（每人有红/蓝 2 种选择）：2^k 种
- 小计：$\binom{n}{k} \cdot 2^k$

总和：

$$\sum_{\substack{0 \leq k \leq n \\ k \text{ 偶}}} \binom{n}{k} 2^k$$

利用二项式定理：

$$(1 + 2)^n = \sum_{k=0}^{n} \binom{n}{k} 2^k = 3^n$$

$$(1 - 2)^n = \sum_{k=0}^{n} \binom{n}{k} (-2)^k = (-1)^n$$

相加得：

$$3^n + (-1)^n = 2 \sum_{k \text{ 偶}} \binom{n}{k} 2^k$$

$$\sum_{k \text{ 偶}} \binom{n}{k} 2^k = \frac{3^n + (-1)^n}{2}$$

答案：$\boxed{\dfrac{3^n + (-1)^n}{2}}$

5. 欧拉公式应用的补充题

（回忆卷还出现了以下题型，因信息不全列出大纲：）

- 哈夫曼编码（Huffman coding）
- 图的最大流（Max-flow）
- 图的最小生成树（MST）
- 字符串排列组合（非降序、包含全部数字等）

这些为课程标准题型，按教材方法求解即可。

（注：本卷为回忆版，题目细节可能不完整。实际考试时请仔细阅读原题条件。所有解题方法均为课程标准内容。）

第四部分

附录

11.20 DMA 期末复习规划

当前日期：6 月 22 日 | 期末考试：6 月 28 日上午

11.20.1 关键日期

日期	事件
6/22 (今天)	摸底卷 + 定位薄弱点
6/23 上午	突击薄弱章节
6/23 下午	学校期末模拟考（范围 = 期末）
6/24-26	分章节刷题 + 历年卷
6/27	总复习 + 错题回顾
6/28 上午	期末考试

11.20.2 6/22（今天）—摸底定位

上午/下午（约 3h）

做摸底卷（90 min 限时）—`notes/Practice-Exam.md`

对答案，记录每道错题对应的章节

晚上（约 1.5h）

统计错题分布，确定 Top 3 薄弱章节

快速过一遍薄弱章节的笔记

11.20.3 6/23 —模拟考日

上午（约 2h）

针对昨天薄弱章节，看笔记 + 做对应 Quiz 题

快速浏览 Ch10-11 图论/树的定理清单

下午—学校模拟考

认真对待，当作真实考试

记录不会做的题、时间分配问题

晚上（约 1h）

复盘模拟考错题

更新薄弱点清单

11.20.4 6/24-26 —真题冲刺（每天 3-4h）

6/24：数理逻辑 + 数论 + 归纳

Ch1 弱点题型回顾（NAND/NOR 表达、Full CNF/DNF、嵌套量词）

Ch4 弱点：Euclidean 算法、CRT、费马小定理

Ch5 弱点：强归纳法证明模板

做 **Quiz1-2024 / Quiz1-2025**（选一套未做过的）

6/25：计数 + 递推 + 生成函数

Ch6：排列组合分类总结（4 种情况速查表）

Ch8：齐次/非齐次递推求解模板、生成函数套路

做 **Quiz2-2022 / Quiz4-2024 前 3 题**

6/26：关系 + 图论 + 树

Ch9：等价关系/偏序/Hasse 图/Warshall 算法

Ch10：Euler/Hamilton 判定、Dijkstra、平面图（$e \leq 3v - 6$）、着色

Ch11：Prim/Kruskal、Huffman、遍历

做 **Quiz3-2024 / Quiz4-2022**

11.20.5 6/27 —总复习（约 3h）

错题重做：之前所有错题重新做一遍

公式默写：在一张白纸上默写所有核心定理

- Handshaking Theorem, Euler's formula ($r = e - v + 2$), $\chi(G)$ bounds
- $P(n, r) = n!/(n - r)!$, $C(n, r) = n!/(r!(n - r)!)$
- Characteristic equation solution forms, Master Theorem
- Hall's Marriage Theorem, Dijkstra algorithm steps

快速浏览：Network Flow 笔记（Ford-Fulkerson, Max-Flow Min-Cut）

11.20.6 6/28 上午—考试

考前 30 min：看一眼公式速查表

带齐计算器、证件、笔

做题策略：先易后难，判断题不会就空着（不倒扣也别乱猜）

11.20.7 每日时间分配建议

	DMA	其他科目
6/22	4.5h	5-6h
6/23	3h	5-6h
6/24	3.5h	5-6h
6/25	3.5h	5-6h
6/26	3.5h	5-6h
6/27	3h	5-6h
6/28	考试	其他科目

DMA 集中在每天上午/下午各一段，其余时间留给其他科目。每天 DMA 不超过 4h。

11.20.8 章节优先级

优先级	章节	原因
	Ch10 Graphs	分值最高，题型多变，Euler/Hamilton/Dijkstra/平面图必考
	Ch1 Logic	Quiz 1 核心，NAND/CNF/DNF/量词每年必出
	Ch5 Induction	至少一道大题，强归纳法/普通归纳法都要会
	Ch11 Trees	Prim/Kruskal/Huffman 算法题必有一道
	Ch4 Number Theory	CRT/费马小定理可能出在 Quiz 1 或 4
	Ch9 Relations	Hasse 图/Warshall/等价类几乎必考
	Ch6 Counting	排列组合题难度适中但易错
	Ch8 Recurrence	递推求解套路固定，会模板就行
	Ch2 Sets	概念题为主，不难
	Ch3 Algorithms	Big-O 判断 + 排序，1-2 道小题
	Network Flow	可能只有 1 题，Ford-Fulkerson 基本概念

11.21 DMA 薄弱点记录

6/22 摸底暴露 → 6/27 逐个消灭 → 6/28 考试

11.21.1 今日消灭计划（6/27）

```
09:00-10:30  [1] Ch9 Relations        整章 2h
10:30-12:00  [2] Ch4 Number Theory     Euclidean+Bézout+CRT+Fermat 1.5h
12:00-13:00  午饭
13:00-14:00  [3] Ch11 Trees           Huffman+Prim+Kruskal 1h
14:00-15:00  [4] Ch10 Graphs          平面图/色数/Euler/Hamilton 1h
15:00-15:30  [5] Ch8 Recurrence       通解+生成函数+derangement 0.5h
15:30-16:00  [6] Ch2+Ch3              术语+Big-O 0.5h
16:00-16:30  [7] Ch1 Logic            NAND+CNF/DNF 0.5h
16:30-17:30  [8] 真题实战             做一套 Quiz3 或 Quiz4
17:30-18:30  晚饭
18:30-20:00  [9] 错题回顾+公式默写    1.5h
20:00-21:00  [10] 快速过所有笔记      扫一遍防遗忘
```

11.21.2 2026-06-22 摸底卷暴露问题

Ch9 Relations —整章空白

- **Q17**: 不认识 equivalence relation（等价关系），不认识 partition（划分）
- **Q18**: 忘记 symmetric / transitive / reflexive 的定义
- **Q33**: 不知道 partial order, Hasse diagram, maximal/minimal/maximum/minimum, lattice

知识点	对应笔记	消灭?
Relations, reflexive/symmetric/antisymmetric/transitive	Ch09-9.1	
Representing relations, closures	Ch09-9.2, 9.3, 9.4	
Equivalence relations, partitions	Ch09-9.5	
Partial order, Hasse diagram, lattice	Ch09-9.6	

Ch4 Number Theory —Euclidean 算法空白

- **Q29**: 不知道 Euclidean algorithm 和 Bézout coefficients

知识点	对应笔记	消灭?
Divisibility, modular arithmetic	Ch04-4.1	
Euclidean algorithm, Bézout's theorem	Ch04-4.3	
Chinese Remainder Theorem	Ch04-4.4	
Fermat's Little Theorem, RSA	Ch04-4.4, 4.5	

Ch11 Trees —Huffman 完全不会

- **Q35**: 完全不知道 Huffman coding

知识点	对应笔记	消灭?
Tree basics, rooted trees	Ch11-11.1	
Huffman coding	Ch11-11.2	
Tree traversal (pre/in/post order)	Ch11-11.3	
Spanning trees, Prim, Kruskal	Ch11-11.4, 11.5	

Ch10 Graphs —术语遗忘

- **Q19**: 忘记 planar graph（平面图，Euler 公式 $r = e - v + 2$，$e \leq 3v - 6$）
- **Q21**: 不知道 chromatic number（色数 $\chi(G)$，四色定理）
- **Q20/Q34**: Euler circuit / Euler path / Hamilton circuit 判定会但会算错度数

知识点	对应笔记	消灭?
Planar graphs, Euler's formula, $K_5/K_{3,3}$	Ch10-10.7	
Graph coloring, chromatic number	Ch10-10.8	
Euler/Hamilton 判定（练 3 道题）	Ch10-10.5	

Ch8 Recurrence —通解形式遗忘

- **Q15**: roots 不知道有什么用 → $a_n = r_1^n + r_2^n$
- **Q16**: 不认识 derangements（错排 $D_n \approx n!/e$）
- **Q32**: 求出 roots 后忘记通解形式

知识点	对应笔记	消灭?
Linear homogeneous RR, characteristic eqn	Ch08-8.2, 8.3	
Nonhomogeneous RR, divide-and-conquer	Ch08-8.4	
Generating functions	Ch08-8.5	
Derangements, inclusion-exclusion	Ch08-8.6	

Ch2 + Ch3 —英文术语不熟

- **Q4**: $|A|$ = cardinality（集合元素个数）
- **Q5**: injective = 单射, surjective = 满射, bijective = 双射
- **Q7**: O = 上界, Ω = 下界, Θ = 紧界, o = 严格上界

知识点	对应笔记	消灭?
Functions, injective/surjective/bijective	Ch02-2.3	
Cardinality, countability	Ch02-2.5	
Big-O / Big-Ω / Big-Θ / little-o	Ch03-3.2	

Ch1 Logic —特殊符号

- **Q26**: 不知道 NAND = |, XOR =
- 需要复习：NAND/NOR 表达、Full CNF/DNF 计算

知识点	对应笔记	消灭?
NAND/NOR, CNF/DNF, minterm/maxterm	Ch01-1.3	

次优先

知识点	对应笔记	消灭?
Well-ordering principle	Ch05-5.2	
Strong induction form	Ch05-5.2	
Pigeonhole principle	Ch05-5.2	
Counting formulas P(n,r), C(n,r)	Ch06-6.3	
Network Flow, Ford-Fulkerson	Network-Flow	

11.21.3 消灭记录

日期	消灭项	验证方式
6/22	摸底卷 35 题对答案	单次