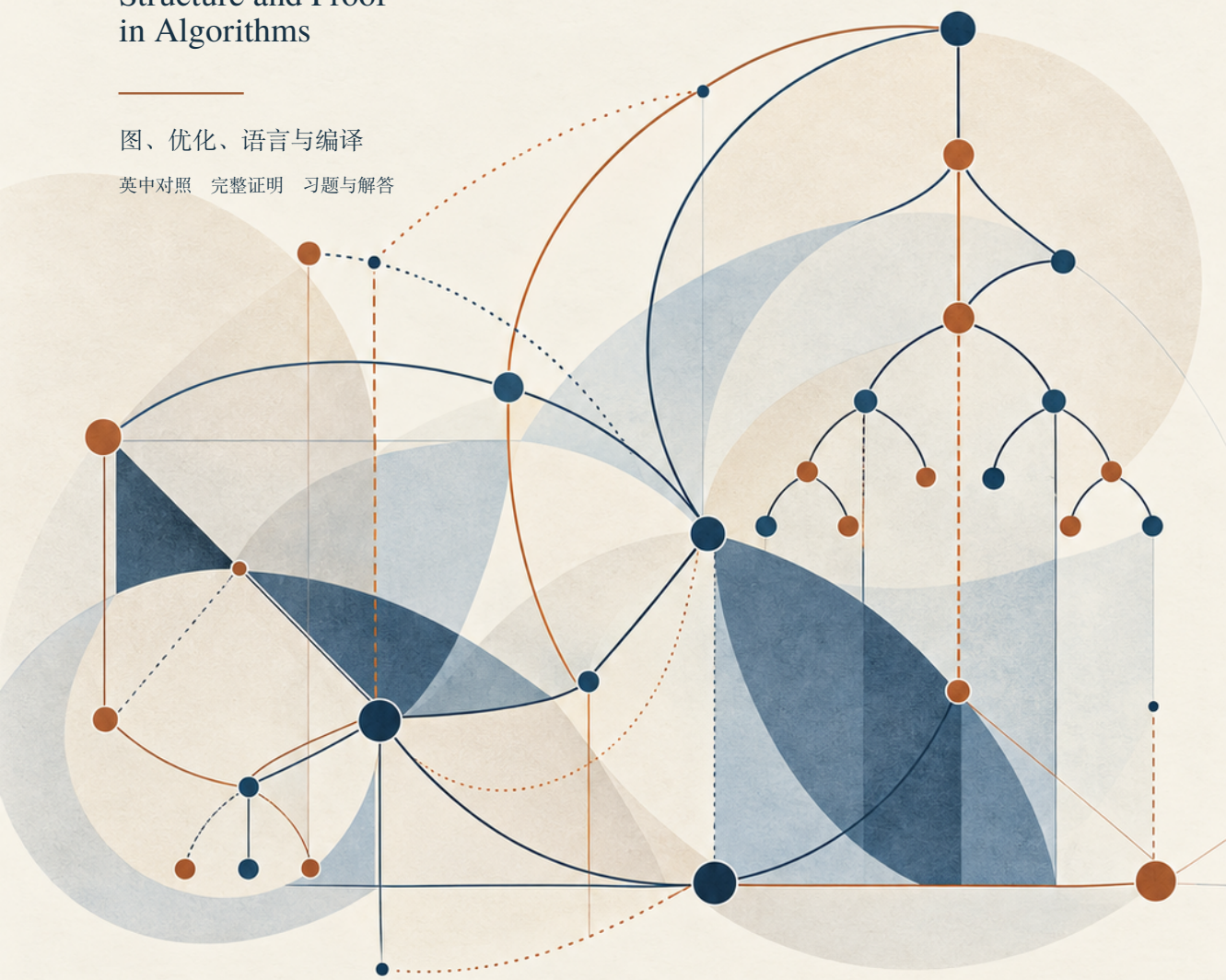
A STUDY TEXT ON TAOCP'S PUBLIC PLAN

算法的结构与证明

Structure and Proof in Algorithms

图、优化、语言与编译

英中对照　完整证明　习题与解答



全规划主题扩展 · 完整学习版

56 节完整合订本 / Complete study edition

2026-10-27　AI 辅助编写与独立审查

非官方续写 / UNOFFICIAL CONTINUATION

原创学习文本；未经 Donald E. Knuth 撰写、审阅或认可。

An independently written study text, not endorsed by Donald E. Knuth.

Contents / 目录

About this expanded edition / 关于扩展版 **xxiii**

7.2.2.5 Clique enumeration and maximum cliques / 团枚举与最大团 **1**

An answer that cannot be enlarged / 一个不能再扩大的答案 1

Graphs, neighborhoods, and two kinds of maximality / 图、邻域与两种最大性 . . 1

The information carried by three sets / 三个集合分别保留什么 3

A pivot selects branches, not answers / 枢轴筛选的是分支 5

Paying for calls as well as output / 调用和输出都要计费 7

Following the same graph through A and B / 在同一张图上追踪 A 与 B 8

Searching for one largest clique / 寻找一个最大团 13

Sparse graphs and a different counting question / 稀疏图与另一个计数问题 17

Exercises / 习题 18

Answers to Exercises / 习题答案 22

Historical Notes / 历史注记 37

7.2.2.6 Covering problems and complete search / 覆盖问题与完备搜索 **39**

A cover is not yet an answer / 找到覆盖之后还须辨明任务 39

Objects, labels, and five different requirements / 对象、标签与五种不同要求 . . . 40

A local certificate with a permanent failure / 局部证书与无法恢复的失败 41

One path per label set / 每个标签集恰由一条路径负责 42

Transposition and what an output means / 转置关系与输出究竟表示什么 45

Costs, ties, and actual incumbents / 费用、并列与真实可行的当前最优解 46

What the array implementation costs / 数组实现的实际代价 48

An eight-label instance / 一个八标签实例 50

Exercises / 习题 54

Answers to Exercises / 习题解答 59

Historical Notes / 历史注记 71

7.2.2.7 Squares: constraints, Latin squares, and symmetry / 方阵约束、拉丁方与对称 **72**

One square, all squares, or one representative? / 一个方阵、全部方阵，还是一个代表? 72

Objects and immutable input / 对象与不可改写的输入 73

A construction that can be checked cell by cell / 可以逐格复算的构造 74

Searching without losing the input / 不丢失输入条件的搜索 75

The order-four trace / 四阶追踪 77

Three kinds of requirements / 三种类型的要求 79

What is allowed to move? / 允许移动哪些对象? 80

Reduction has a particular multiplier / 约化只对应特定的乘数 82

Stabilizers and complete-square filtering / 稳定子与完整方阵过滤 83

Costs, including representations / 把表示方式也计入成本 86

Modular tools for the exercises / 习题所需的模运算工具 90

Exercises / 习题 92

Answers to Exercises / 习题解答 95

Historical Notes / 历史注记 107

7.2.2.9 Estimating search-tree size and backtracking cost / 搜索树规模与回溯成本估计 **109**

A short path can represent a large tree / 短路径也可能代表大树 109

The object being estimated / 先固定估计对象 109

Two path algorithms / 两种路径算法 111

Why the estimator is unbiased / 无偏性为什么成立 112

Where variance enters / 方差从哪里来 114

Repetition and an honest interval / 重复探测与有依据的区间 115

An original twelve-node tree / 一棵原创的十二结点树 119

What a probe costs / 一次探测本身要付多少成本 123

Executed experiments and a misleading zero / 已运行实验与会误导的零方差 . . 124

Exercises / 习题 128

Answers to Exercises / 习题解答 133

Historical Notes / 历史注记 144

7.2.3 Generating inequivalent patterns / 群作用与不等价模式生成 **145**

When a second answer is the same answer / 何时第二个答案仍是同一个答案 . . 145

Actions, fibers, and a counting identity / 作用、纤维与计数恒等式 145

A baseline that exposes the cost / 先把代价看清楚 148

Choosing a deletion orbit / 选择删除轨道 149

Growing one representative per orbit / 每条轨道只生长一个代表 151

Accounting for the work / 怎样计费 153

A rectangular board, all the way through / 把矩形棋盘完整走一遍 155

Reproducible finite evidence / 可复现的有限证据 160

Exercises / 习题 161

Answers to Exercises / 习题答案 166

Historical Notes / 历史注记 177

7.3.1 All-pairs shortest paths / 全源最短路径 **179**

Routes, costs, and three kinds of answer / 路线、代价与三种答案 179

Reachability before arithmetic / 先处理可达性 181

Numerical layers and negative cycles / 数值分层与负环 183

A trace with a zero-weight cycle / 含零权环的追踪 186

Recovering a path that must terminate / 必然终止的路径恢复 187

Cost model and bounded-length products / 计算模型与限长乘积 191

Exercises / 习题 193

Answers to exercises / 习题解答 197

Historical notes / 历史注记 209

7.3.2 Single-source shortest paths, potentials, and heuristics / 单源最短路径、势函数与启发式 **210**

When may a distance stop changing? / 距离何时可以不再改变? 210

Bounded walks and Bellman–Ford / 有长度上界的游走与 Bellman–Ford 算法 . . 211

A complete trace / 一个完整追踪 214

Nonnegative weights and Dijkstra's algorithm / 非负权与 Dijkstra 算法 215

Potentials change arc weights, not route comparisons / 势函数改变弧权，保持路线比较 218

A* and the cost of an estimate / A* 与估计值的作用 220
What an arithmetic operation costs / 一次算术操作的成本 224
Exercises / 习题 228
Answers to exercises / 习题答案 232
Historical notes / 历史注记 244

7.4.1 Graph traversal, strong components, and cut structure / 图遍历、强连通分量与割结构 246

A return from recursion is not a component boundary / 递归返回并不等于分量结束 246
What a depth-first forest records / 深度优先森林记录了什么 247
One search for the strongly connected components / 用一次搜索求强连通分量 . . 249
Following the two stacks / 逐步追踪两个栈 253
Removing an undirected vertex or edge / 删除无向图中的顶点或边 255
A multigraph trace and the cost model / 多重图追踪与计算成本 259
Exercises / 习题 261
Answers to exercises / 习题解答 264
Historical notes / 历史注记 277

7.4.1.1 Disjoint-set union and amortized analysis / 并查集与摊还分析 278

A question that survives each insertion / 每次插入之后都要回答的问题 278
The abstract contract and the machine / 抽象合同与机器模型 279
Root linking and complete compression / 连接根与完全路径压缩 280
What ranks remember / 秩保存了什么 283
Paying for explicit membership / 显式成员表怎样付费 285
A complete rank-bucket accounting / 完整的秩分桶核算 287
Two traces whose charges can be checked / 两组能够逐项核对的追踪 289
Inserting graph edges, and the deletion boundary / 插边应用与删除边界 295
Reproducing the numbers / 复算这些数字 296
Exercises / 习题 297
Answers to Exercises / 习题解答 303
Historical Notes / 历史注记 318

7.4.1.3 General vertex and edge connectivity and disjoint paths / 一般点边连通与不交路径 **320**

A route that survives a failure / 故障之后仍然可走的路 320

The integer flow kernel / 整数流核心 321

Returning to original edges and vertices / 返回原图的边和顶点 324

From one pair to the whole graph / 从一个点对到全图 327

What the reader can check / 读者可以核验什么 329

Four small computations / 四个小型计算 331

Cost and reproducibility / 成本与复现 334

Exercises / 习题 336

Answers to Exercises / 习题解答 341

Historical Notes / 历史注记 352

7.4.2 Special graphs: bipartiteness and Eulerian structure / 特殊图：二分性与欧拉结构 **353**

Two verifiable outputs / 两种可以核验的输出 353

Coloring with an odd-cycle certificate / 带奇环证书的着色 355

Closed trails and the stack construction / 闭迹与栈构造 358

Exact existence tests and open trails / 精确存在判据与开放迹 362

Cost models and verification / 计算模型与核验 364

Exercises / 习题 366

Answers to Exercises / 习题解答 371

Historical Notes / 历史注记 380

7.4.2.1 Chordal graphs, interval graphs, and planar embeddings / 弦图、区间图与平面嵌入 **381**

A representation must preserve the missing edges / 表示也必须保留缺失的边 . . 381

Local deletion and its global justification / 局部删除及其整体依据 382

Intervals and a complete, expensive recognition test / 区间及完整而昂贵的识别检验 387

An example with contact, a point, and an isolated component / 含接触、点区间与孤立分量的例子 391

Following a face requires an embedding / 沿面行走需要嵌入 394

A drawn example and its entire dart table / 一幅图及其完整半边表 397
Counting faces without losing bridges or components / 计数时保留桥与分量 . . . 399
Tools for prescribed integer lengths / 指定整数长度所需的工具 402
Exercises / 习题 407
Answers to Exercises / 习题解答 412
Historical Notes / 历史注记 427

7.4.3 Expander graphs, spectra, and random walks / 扩展图、谱与随机游走 429

A cut that is cheap, and a walk that does not mix / 廉价的割与不混合的游走 . . . 429
Definitions and the linear algebra actually needed / 定义与实际使用的线性代数 . 430
Energy, signs, and cuts / 能量、符号与割 431
Forty permutations suffice / 四十个置换已经足够 435
Convergence with a stated initial condition / 写明初态的收敛界 437
Algorithms and what their cost includes / 算法及其成本包含什么 440
Exact small instances / 精确的小规模实例 444
Exercises / 习题 449
Answers to Exercises / 习题解答 452
Historical Notes / 历史注记 462

7.4.4 Random graph models and thresholds / 随机图模型与阈值 463

The same mean, different graphs / 均值相同，分布不同 463
Two finite spaces / 两个有限空间 463
What moments can establish / 矩能证明什么 466
Triangles: count the pairs before taking limits / 三角形：先数指标对，再取极限 467
Exact sampling and its costs / 精确采样及其成本 470
Coupling two laws without equating them / 耦合两个分布而不混同它们 474
Elementary tools for the exercises / 习题所需的初等工具 476
Exercises / 习题 478
Answers to Exercises / 习题解答 481
Historical Notes / 历史注记 489

7.5.1 Bipartite matching and assignment / 二分匹配与指派 **491**

A. Matching, obstruction, and cost / 匹配、障碍与费用 491

1. What a matching can change / 匹配能够怎样改变 492

2. Paying for a perfect assignment / 完美指派的费用 497

3. Tools for the final exercise / 最后一题需要的工具 504

Exercises / 习题 505

Answers to Exercises / 习题解答 510

Historical Notes / 历史注记 522

7.5.3.1 Maximum flow and minimum cut / 最大流与最小割 **524**

7.5.3.1.1 A route may need to be undone / 一条已用路线可能需要撤销 524

7.5.3.1.2 Residual records and an optimality certificate / 残量记录与最优性证书 . 526

7.5.3.1.3 Choosing a shortest augmenting path / 选择最短增广路径 527

7.5.3.1.4 What the running time counts / 运行时间究竟计入什么 530

7.5.3.1.5 A trace that cancels an earlier choice / 真正撤销先前选择的追踪 532

7.5.3.1.6 Exercises / 习题 536

7.5.3.1.7 Answers to exercises / 习题解答 541

7.5.3.1.8 Historical notes / 历史注记 549

7.5.3.2 Minimum-cost flow / 最小费用流 **550**

7.5.3.2.1 Keeping the deliveries, changing the bill / 保持交付量，改变总费用 . . 550

7.5.3.2.2 What a residual record costs / 残量记录的费用 551

7.5.3.2.3 A certificate on every residual arc / 检查每条残量弧的证书 554

7.5.3.2.4 An integer cancellation algorithm / 整数负环消去算法 555

7.5.3.2.5 Storage, arithmetic, and submitted bits / 存储、算术与提交数据的位数 . 557

7.5.3.2.6 A complete cancellation trace / 完整的消去追踪 559

7.5.3.2.7 Exercises / 习题 562

7.5.3.2.8 Answers to exercises / 习题解答 567

7.5.3.2.9 Historical notes / 历史注记 578

7.5.4 Minimum spanning trees and optimal subtrees / 最小生成树与最优子树 **580**

Connecting everything / 把所有顶点连起来 580

The objects being optimized / 优化对象 581
What a light edge guarantees / 轻边究竟保证什么 583
Algorithm K (Kruskal's method) / 算法 K（Kruskal 方法） 584
Costs of this implementation / 这个实现的成本 586
A trace with ties and a disconnected graph / 含并列权重的不连通图追踪 587
Exercises / 习题 588
Answers to Exercises / 习题解答 593
Historical Notes / 历史注记 604

7.5.4.1 Optimum arborescences and the boundary of subtree problems / 最优树形图与子树边界 606

A cheap incoming arc can point the wrong way / 最便宜的入弧可能形成闭环 . . 606
Objects and admissible input / 对象与合法输入 607
Normalization and the exchange that makes contraction legal / 归一化与使收缩合法的交换 609
Algorithm A (Normalize, contract, expand) / 算法 A（归一化、收缩、展开） . . 612
What the implementation costs / 实现的成本 613
Two traces with original arc identities / 两个保留原弧身份的追踪例 614
A subtree inside an already given tree / 在已给树内寻找子树 618
Why the host matters / 主图条件为何重要 620
Tools for checking an optimum without solving again / 无须重新求解的最优性核验工具 622
Exercises / 习题 623
Answers to Exercises / 习题答案 627
Historical Notes / 历史注记 636

7.5.5.1 Maximum-cardinality matching in general graphs / 一般图最大基数匹配 638

When the search meets itself / 搜索遇见自己 638
A forest that returns three different answers / 返回三类结果的森林 640
What contraction preserves / 收缩究竟保留什么 643
One improvement at a time / 每次只改进一步 646
Two contractions worth tracing / 两个值得逐层追踪的例子 648

Before enumerating all maximum matchings / 枚举全部最大匹配之前 652
Exercises / 习题 653
Answers to Exercises / 习题答案 656
Historical Notes / 历史注记 663

7.5.5.2 Weighted general matching and odd-set certificates / 带权一般匹配与奇集证书 664

The triangle that defeats degree bounds / 度约束挡不住的三角形 664
Definitions and objectives / 定义与目标 665
Finite-dimensional tools / 有限维工具 667
The perfect matching theorem / 完美匹配定理 672
From perfect to arbitrary matchings / 从完美匹配到一般匹配 675
Exact construction and independent checking / 精确构造与独立检查 676
A perfect-matching optimization reduction / 完美匹配优化归约 681
Five reproducible examples / 五组可复算例子 682
Tools for the robustness exercise / 稳健半径习题的工具 684
Exercises / 习题 687
Answers to Exercises / 习题答案 691
Historical Notes / 历史注记 699

7.5.6 Permutation optimization and scheduling / 排列最优化与调度 701

Jobs, orders, and adjacent exchanges / 任务、次序与相邻交换 701
Remembering the completed set / 保存已经完成的集合 704
Edge lengths as prefix cuts / 把边长写成前缀割 708
What the exact algorithms cost / 精确算法的成本 711
A reduction with strictly positive times / 处理时长严格为正的归约 713
Tools for a more structured precedence relation / 更有结构的先后关系所需工具 . 717
Exercises / 习题 718
Answers to Exercises / 习题解答 723
Historical Notes / 历史注记 735

7.6.1 Independence structures / 独立结构 **737**

When the first choice blocks two better ones / 第一次选择怎样挡住两个更好的选择 . . . 737

Feasible sets and augmentation / 可行集与增广 . . . 738

Bases inside a specified set / 指定集合内部的基 . . . 739

Rank records the usable size / 秩记录可用的大小 . . . 741

Closure and a witness for rejection / 闭包与拒绝的见证 . . . 742

Algorithm X (Maximum-weight independent set) / 算法 X（最大权独立集） . . . 744

What the operation count pays for / 运算计数究竟支付什么 . . . 747

A hand trace with two rejections / 包含两次拒绝的手算追踪 . . . 747

Exercises / 习题 . . . 749

Answers to Exercises / 习题答案 . . . 754

Historical Notes / 史注 . . . 763

7.6.2 Efficient matroid algorithms / 高效拟阵算法 **765**

When a negative edge must be taken / 必须接受负权边的时候 . . . 765

Bases, order, and a complete scan / 基、次序与完整扫描 . . . 766

Algorithm X (Maximum-weight basis) / 算法 X（最大权基） . . . 766

A certificate that does not remember the run / 不依赖运行记录的证书 . . . 768

Three insertion tests / 三种插入测试 . . . 769

Algorithm Y (Graphic insertion by component size) / 算法 Y（按分量大小合并的图插入） . . . 771

Algorithm Z (Incremental exact elimination) / 算法 Z（增量精确消元） . . . 772

Costs that remain outside the test / 测试以外的成本 . . . 773

A forest, its trace, and its certificate / 一片森林、它的追踪与证书 . . . 774

The boundary at two constraints / 两组约束处的边界 . . . 777

Exercises / 习题 . . . 777

Answers to Exercises / 习题答案 . . . 782

Historical Notes / 史注 . . . 790

7.6.2.1 Matroid intersection, union, and partition algorithms / 拟阵交、并与划分算法 **791**

Two approvals can still block progress / 两次批准仍可能挡住后续选择 791
The objects queried / 查询的对象 791
Why a whole path may be flipped / 为什么整条路径可以翻转 792
A failed search is an upper bound / 搜索失败给出的上界 794
Algorithm X (Shortest exchange augmentation) / 算法 X（最短交换增广） 796
Copying elements to assign them / 用复制底集完成分配 797
Algorithm Y (Allocation by intersection) / 算法 Y（通过交算法分配） 798
Seven labels and five path vertices / 七个标签与五顶点路径 800
A forest exchange and a real reassignment / 森林交换与实际再分配 803
Exercises / 习题 805
Answers to Exercises / 习题解答 808
Historical Notes / 历史注记 817

7.7 Discrete dynamic programming / 离散动态规划 **818**
What must a state remember? / 状态必须记住什么? 818
Why an evaluation order exists / 为什么存在求值次序 819
A capacity-indexed selection problem / 以容量为下标的选择问题 821
Algorithm X (Layered 0–1 knapsack) / 算法 X（分层 0–1 背包） 822
What a rolling row preserves / 滚动单行保留了什么 824
Comparisons, arithmetic, and input length / 比较、算术与输入长度 825
A complete trace / 一次完整追踪 826
Exercises / 习题 827
Answers to Exercises / 习题解答 832
Historical Notes / 历史注记 843

7.7.1 Dynamic programming with several state structures / 多类状态结构的动态规划 **845**
The information at a cut / 切开一个解以后留下的信息 845
Subsets with an endpoint / 带末点的子集 847
Algorithm X (Subset paths) / 算法 X（子集路径） 848
Intervals without an external-depth coordinate / 不必记录外部深度的区间 851
Algorithm Y (Interval roots) / 算法 Y（区间选根） 852

One bit at a subtree boundary / 子树边界上的一个二进制位 854
Algorithm Z (Forest interfaces) / 算法 Z（森林接口） 855
Counting histories that can no longer be distinguished / 对未来相同的历史仍然分别计数 857
Algorithm P (Count by binary exponentiation) / 算法 P（二进制快速幂计数） . . 859
Algorithm W (Lexicographically smallest witness) / 算法 W（字典序最小见证） 860
Tools for a cardinality constraint / 固定基数所需的工具 863
What the runnable record includes / 可运行记录的成本范围 864
Exercises / 习题 865
Answers to Exercises / 习题解答 869
Historical Notes / 历史注记 879

7.8 Branch and bound with checkable relaxations / 分支定界与可核查松弛 880

A solution costing 13 / 一个费用为 13 的解 880
Definitions and arithmetic / 定义与运算 881
Solving the relaxation / 求出松弛问题的最优值 882
Branching without losing a solution / 不丢解的分支 883
Algorithm X (Prefix branch and bound) / 算法 X（前缀分支定界） 885
Correctness, stopping, and certificates / 正确性、停止与证书 886
Cost of search and cost of the record / 搜索成本与记录成本 890
Eleven generated prefixes / 十一个实际生成前缀 891
Tools for a different certificate / 另一种证书的工具 894
Exercises / 习题 896
Answers to Exercises / 习题解答 901
Historical Notes / 历史注记 913

7.9 Computational complexity and NP-completeness reductions / 计算复杂性与 NP 完全归约 914

A lost label changes the answer / 丢掉一个标签，答案就变了 914
Definitions and encoding / 定义与编码 914
Signals, formulas, and the foundation / 信号、公式与基础定理 917
From clauses to vertices / 从子句到顶点 921

Encoding a cover in digits / 把覆盖写进数位 923
A table whose width is a number / 宽度由数值决定的表 926
Worked traces / 手算追踪 929
Tools for the final exercise / 最后一题所需工具 932
Exercises / 习题 933
Answers to Exercises / 习题解答 938
Historical Notes / 历史注记 949

7.10 Approximation algorithms and schemes / 近似算法与近似方案 951

A cheap answer needs a lower bound / 便宜的答案还需要下界 951
Guarantees, encodings, and zero / 保证、编码与零值 951
Prices on edges / 在边上定价 953
Paying for new elements / 为新覆盖元素付费 956
Rounding profits while keeping capacity exact / 缩放利润，保持容量精确 960
A trace that exposes the preprocessing order / 能暴露预处理顺序的追踪 964
Tools for the two-capacity question / 双容量问题所需工具 966
Exercises / 习题 967
Answers to Exercises / 习题解答 971
Historical Notes / 历史注记 983

8 Recursion and recurrence analysis / 递归与递推分析 985

Part A: Specifications, executions, and charged trees / A 部分：规格、执行与节点收费 985
Part B: Exercises, answers, and historical notes / B 部分：习题、答案与历史注记 995
Exercises / 习题 995
Answers to Exercises / 习题解答 999
Historical Notes / 历史注记 1010

8.1 Mutual recursion, well-foundedness, and call DAGs / 互递归、良基性与调用有向无环图 1011

Part A: Which calls may share an answer? / A 部分：哪些调用能够共享答案？ . 1011
Part B: Exercises, answers, and historical notes / B 部分：习题、解答与历史注记 1029
Exercises / 习题 1030

Answers to Exercises / 习题解答 1035
Historical Notes / 历史注记 1047

9.1 Regular languages and finite automata / 正则语言与有限自动机 1048
A. Remembering only what the suffix can test / 只记住后缀能够检验的差别 . . . 1048
Exercises / 习题 1061
Answers to exercises / 习题解答 1065
Historical notes / 历史注记 1075

9.2 Scanner construction and implementation / 扫描器构造与实现 1076
A. Finding where a word ends / 确定一个词素在哪里结束 1076
Exercises / 习题 1087
Answers to Exercises / 习题解答 1091
Historical Notes / 历史注记 1102

9.3 Single-pattern and dictionary matching / 单模式与多模式字符串匹配 1103
A. Keeping a useful suffix / 留住仍然有用的后缀 1103
Exercises / 习题 1123
Answers to Exercises / 习题解答 1128
Historical Notes / 历史注记 1139

9.4 Suffix indexes and text queries / 后缀索引与文本查询 1141
The indexed objects / 索引的对象 1141
Building the order without copying suffixes / 不复制后缀的构建 1142
Queries as two boundaries / 查询化为两个边界 1145
Reusing common prefixes / 复用公共前缀 1147
Nonadjacent suffixes and a supplied RMQ tool / 非相邻后缀与区间最小查询工具 1149
Two applications / 两项应用 1150
An exact trace / 精确追踪 1151
Cost boundaries and executable evidence / 成本边界与可执行证据 1154
Exercises / 习题 1155
Answers to Exercises / 习题解答 1159
Historical Notes / 历史注记 1172

9.5 Prefix codes, entropy, and optimal coding / 前缀码、熵与最优编码 **1173**

The model and its boundary / 模型及其边界 1173

Recovering the boundaries / 恢复码字边界 1174

Which lengths fit? / 哪些长度放得下？ 1175

Why merging is legitimate / 合并为什么成立 1177

Algorithm H and its bill / 算法 H 及其成本 1179

The seven-symbol calculation / 七符号计算 1182

How entropy enters / 熵如何进入问题 1186

Exercises / 习题 1188

Answers to Exercises / 习题解答 1193

Historical Notes / 历史注记 1205

9.6 Dictionary compression and arithmetic coding / 词典压缩与算术编码 **1207**

Messages, bits, and a length header / 消息、比特与长度帧头 1207

A dictionary that both sides can build / 双方能够同步建立的词典 1209

Exact probability intervals / 精确概率区间 1216

A whole binary cell, rather than a point / 选择完整二进制单元 1217

Frames and midpoint reconstruction / 帧与中点重建 1219

Precision, work, and the executable experiment / 精度、工作量与可执行实验 . . 1222

Exercises / 习题 1225

Answers to Exercises / 习题解答 1230

Historical Notes / 历史注记 1247

10.1 Top-down parsing / 自顶向下语法分析 **1248**

A choice before the next token / 在下一个记号之前作出选择 1248

Building and executing the table / 构造与执行分析表 1252

A complete small table / 一张完整的小分析表 1257

Rewriting a grammar without changing its language / 不改变语言的文法改写 . . 1259

When choice itself has an order / 当选择本身具有顺序 1261

Exercises / 习题 1262

Answers to Exercises / 习题解答 1267

Historical notes / 历史注记 1283

10.2 Bottom-up parsing / 自底向上语法分析 **1284**

A completed phrase is not yet a decision / 短语已经读完，决定仍未作出 1284

What an item proves about a stack / 一个项目究竟证明了栈的什么性质 1288

The parser and its stopping argument / 分析器及其停止论证 1291

A table that keeps two contexts apart / 把两个上下文分开的一张表 1294

Counting the actual work / 计算实际执行的工作 1297

Keeping fewer states / 减少状态之后 1298

Exercises / 习题 1302

Answers to Exercises / 习题解答 1307

Historical Notes / 历史注记 1322

11.1 Grammar normal forms and ambiguity / 文法范式与二义性 **1324**

Part A. Trees, normalization, and recognition / A 篇：树、规范化与识别 1324

Part B. Exercises, answers, and historical notes / B 篇：习题、解答与历史注记 . 1334

11.2 Pumping lemmas and decision boundaries / 泵引理与判定边界 **1352**

Part A. Repeated labels and finite decisions / A 篇：重复标号与有限判定 1352

Part B. Exercises, answers, and historical notes / B 篇：习题、解答与历史注记 . 1364

Exercises / 习题 1365

Answers to Exercises / 习题解答 1369

Historical Notes / 历史注记 1379

11.3 Parsing complexity and determinism / 解析复杂度与确定性 **1380**

A completion can arrive too early / 完成项也可能来得太早 1380

Objects that remain distinct / 哪些对象必须保留区别 1380

An implementation with two arrival orders / 同时处理两种到达顺序 1382

Two traces / 两个完整追踪 1386

What the cubic bound charges for / 三次界究竟计算了什么 1387

A deterministic linear example / 一个确定的线性实例 1389

Exercises / 习题 1391

Answers to Exercises / 习题解答 1397

Historical Notes / 历史注记 1415

11.4 Pushdown automata and grammar equivalence / 下推自动机与文法等价 **1416**

An empty input is not an empty stack / 输入读完，不等于栈已清空 1416

What a transition is allowed to inspect / 转移能检查什么 1416

Changing the acceptance condition requires another machine / 改接受条件，需要另造机器 1418

Keeping the unfinished tree on the stack / 把未完成的树放在栈上 1420

Making long replacements local / 把长替换拆成局部步骤 1422

A variable records the disappearance of one layer / 一个变量记录一层栈的消失 . 1423

Explicit output has a cost / 显式输出需要付费 1429

Finite relations for an infinite collection of stacks / 用有限关系表示无穷多个栈 . 1430

Exercises / 习题 1432

Answers to Exercises / 习题解答 1436

Historical Notes / 历史注记 1446

11.5 Language operations, closure properties, and normal forms / 语言运算、封闭性与范式 **1447**

A record can disappear while its input still has to be read / 记录可以消失，输入却仍须读取 1447

Objects and promises / 对象与承诺 1447

Graft trees, then cut them back / 嫁接树，再把它切回来 1448

A finite filter shares the input, not the stack / 有限过滤器共享输入 1451

Loading an image without another unbounded tape / 不增设无界纸带地加载字母像 1452

Why two unrestricted stacks do not follow from a product / 任意两个语言不能照此求交 1454

One terminal per production / 每条产生式产出一个终结符 1457

Cost is paid for the text actually written / 按实际写出的文本计费 1461

Two worked records / 两组可手算的记录 1464

What the finite checks certify / 有限检查证明了哪一类事实 1468

Exercises / 习题 1470

Answers to Exercises / 习题解答 1474

Historical Notes / 历史注记 1485

11.6 Deterministic context-free languages / 确定性上下文无关语言 **1487**

The last input symbol does not end the computation / 最后一个输入符号并不结束计算 1487

Four acceptance conditions / 四种接受条件 1488

A finite calculation about an unbounded stack / 关于无界栈的有限计算 1489

Removing endless empty computations / 消除无尽空计算 1491

The bit that must be reset / 必须重置的一位 1495

Regular filters and the empty-stack boundary / 正则过滤与空栈边界 1499

Putting on a marker, then taking it off / 加上标记，再删除标记 1500

Unique runs give an unambiguous grammar / 用唯一运行构造无二义文法 1504

The inclusions are strict / 两个包含都是真包含 1505

Cost, and tools for the last exercise / 成本与末题工具 1507

Exercises / 习题 1509

Answers to Exercises / 习题解答 1513

Historical Notes / 历史注记 1521

11.7 Ambiguity, undecidability, and reductions / 二义性、不可判定性与归约 **1522**

Two trees are only the beginning / 两棵树只是起点 1522

Positions inside an actual tree / 实际树中的位置 1522

Forcing two crossing phrases / 迫使两个短语交叉 1525

A machine that can read its own description / 能读取自身描述的机器 1528

The local relation is finite-state / 局部关系只需有限状态 1530

Two online checks / 两份在线检查 1533

Which answer does a reduction preserve? / 归约保持哪一个答案？ 1536

Closure tools that preserve unique trees / 保持树唯一性的工具 1539

A small history with two different errors / 一条短历史与两种错误 1540

Cost and the executable boundary / 成本与可执行边界 1541

Exercises / 习题 1543

Answers to Exercises / 习题解答 1545

Historical Notes / 历史注记 1551

12.1 Intermediate representations and dataflow optimization / 中间表示与数据流优化 **1553**

A value that will never be used / 一个不会再被使用的值 1553

Instructions and observations / 指令与观察 1554

What must remain equal / 哪些值必须保持一致 1555

A worklist whose initialization matters / 初始化不可省略的工作列表 1558

A loop calculated to completion / 把一个循环算到结束 1561

Representation costs and other directions / 表示成本与其他方向 1563

Exercises / 习题 1565

Answers to Exercises / 习题答案 1570

Historical Notes / 历史注记 1579

12.2 Instruction selection and register allocation / 指令选择与寄存器分配 **1580**

12.2.1 One expression, different questions / 同一表达式，不同问题 1580

12.2.2 Covering a tree with instruction patterns / 用指令模式覆盖树 1581

12.2.3 How many registers does a tree need? / 一棵树需要多少寄存器? 1585

12.2.4 Keeping subtraction in order / 保持减法的操作数次序 1587

12.2.5 A fixed schedule gives intervals / 固定调度给出区间 1590

12.2.6 When there are only k registers / 只有 k 个寄存器时 1591

12.2.7 Exercises / 习题 1594

12.2.8 Answers to exercises / 习题解答 1598

12.2.9 Historical notes / 历史注记 1611

12.3 Runtime, storage, and garbage collection / 运行时、存储与垃圾回收 **1613**

Moving an object without changing its meaning / 搬动对象而不改变其含义 . . . 1613

The copied prefix is also a queue / 复制前缀同时充当队列 1615

A trace with sharing and a cycle / 含共享与环的追踪 1620

Cost, allocation, and failure / 成本、分配与失败 1622

Charging variable-size requests / 为变长请求计费 1624

Exercises / 习题 1626

Answers to Exercises / 习题解答 1631

Historical Notes / 历史注记 1643

12.4 Names, scope, and type checking / 名字、作用域与类型检查 **1645**

Declarations before values / 先分清声明与值 1645

The rules a checker must implement / 检查器必须实现的规则 1648

A store that survives a block / 穿过局部块的存储 1651

Cost and what the laboratory counts / 成本与附件的计量范围 1656

Two executions with the same names / 相同名字的两次执行 1658

Preparing an initialization question / 为初始化问题准备工具 1662

Exercises / 习题 1664

Answers to Exercises / 习题解答 1668

Historical Notes / 历史注记 1685

12.5 Syntax-directed translation and intermediate code / 语法制导翻译与中间代码 **1686**

The fragment must say where it ends / 片段必须说明在哪里结束 1686

Constructing the graph once / 一次构造控制流图 1689

The program counter and the source focus / 程序计数器与源焦点 1693

Static size is not execution time / 静态大小不是执行时间 1699

P, with every target instruction exposed / P 的每条目标指令 1701

Exercises / 习题 1710

Answers to Exercises / 习题解答 1715

Historical Notes / 历史注记 1728

12.6 Dominance, SSA, and proofs of optimization / 支配、SSA 及优化证明 **1729**

Where two executions meet / 两条执行路径在哪里汇合 1729

The version that crosses an edge / 穿过一条边的版本 1733

One meter, three ways back to the header / 三条边怎样回到同一个计数块 1737

Crossing a merge without φ / 用普通指令穿过合流处 1743

Paying for the cells / 每个格子的代价 1747

Tools for the last problem / 最后一题的工具 1748

Exercises / 习题 1750

Answers to Exercises / 习题解答 1753

Historical Notes / 历史注记 1762

12.7 Procedures, closures, calls, and stack frames / 过程、闭包、调用与栈帧 1763

The three identities / 三种身份 1763

Heap layouts and construction boundaries / 堆布局与构造边界 1766

A source machine that can stop inside an expression / 可以停在表达式内部的源机 1770

Compiling control into instructions and frames / 把控制编译成指令与帧 1774

Matching finite steps without assuming calls return / 不预设调用返回的有限步对应 1778

What the experiments expose / 实验具体展示什么 1784

A finite graph for trimming / 用于裁剪的有限图 1787

Costs that survive the representation choice / 表示选择之后仍需核算的成本 . . . 1789

Exercises / 习题 1791

Answers to Exercises / 习题解答 1795

Historical Notes / 史注 1809

12.8 General register allocation, spilling, and emission / 一般寄存器分配、溢出与代码发出 1811

A machine small enough to specify / 先把机器规定完整 1811

What may be read before it is replaced / 被覆盖之前可能读取什么 1814

An edge for a write that nobody wants / 无用写入也要有边 1815

Two scratch registers, including the awkward subtraction / 两个暂存寄存器也能做减法 1818

Optimize before allocating, then analyze again / 先优化，再重新分析和分配 . . . 1822

Costs that the fallback actually pays / 回退算法实际支付的成本 1824

A finite interface for exact weighted spilling / 精确加权溢出的有限接口 1827

Exercises / 习题 1828

Answers to Exercises / 习题解答 1833

Historical Notes / 历史注记 1846

12.9 A complete compiler / 一个完整的编译器 1848

Bytes, declarations, and values / 字节、声明和值 1848

An interpreter that does not compile / 不经过编译的解释器 1851

Connecting every boundary / 把每个边界接起来 1854

A record that can be checked / 可以回查的记录 1859
Finite experiments and their cost / 有限实验及其成本 1862
Exercises / 习题 1865
Answers to Exercises / 习题解答 1870
Historical Notes / 历史注记 1883

Appendix A. Terminology / 附录 A 术语表 **1884**

Appendix B. Public material and reading routes / 附录 B 已公开原作与阅读路径 **2290**

About this expanded edition / 关于扩展版

UNOFFICIAL CONTINUATION. This is an original, independently written study text organized around publicly announced TAOCP topics. It was not written, reviewed, or endorsed by Donald E. Knuth or the original publisher.

非官方续写。本书按 TAOCP 公开选题组织原创讲解，未经 Donald E. Knuth 或原出版方撰写、审阅或认可。

The first edition contains twenty-two sections. The expanded edition grows from those unchanged, reviewed sources to address the remaining public topics in combinatorial algorithms, syntactic algorithms, context-free language theory, and compilation. Its coverage table records what each section actually teaches; a topic appearing in a table of contents does not by itself count as completed coverage.

第一版包含二十二节，扩展版在这些已经审阅、正文保持不变的材料上，继续补充组合算法、语法算法、上下文无关语言理论和编译技术中的公开选题；覆盖表记录每节实际讲授的内容，仅仅把题目写进目录，不算完成这一主题。

The public plan supplies topic boundaries, not an unpublished manuscript. Where no detailed plan is available, the subdivision is ours. The remaining parts of Volume 4 are not assigned invented official subvolume boundaries. Existing books and public prefascicles remain separate works: references to them identify available material and do not claim it as new writing in this project.

公开规划提供的是主题边界，并非尚未公开的书稿，没有详细纲目的部分由本书自行编排；第四卷后续各部分也不会被擅自分配到某个官方分卷。已出版著作与已公开预分册仍是独立作品，对它们的引用用于说明哪些材料已经存在，不把这些材料记为本项目新增写作。

English paragraphs are followed by corresponding Chinese paragraphs; formulas, algorithms, and numerical traces are shared. Each new instructional section has a stated model, complete arguments, worked examples, exercises with answers, and a source record distinguishing bibliographic verification from actual reading. The difficulty labels are editorial estimates made for this book.

英文段落后紧接内容对应的中文段落，公式、算法和数值追踪共用一份，每个新增教学单元都说明计算模型，给出完整论证、算例、习题及答案，并在来源记录中区分书目核验与实际阅读；难度标记是本书的教学估计。

This expanded edition contains fifty-six reviewed instructional sections and 846 exercises with complete answers. Mathematical review, notation checks, compilation, visual inspection, and final packaging are recorded separately. The first edition remains available with its original verification record, so that new work does not change the meaning of an earlier delivery.

本扩展版包含五十六节已经审阅的教学单元和 846 道附完整答案的习题，数学审阅、记号检查、编译、实际页面检查和最终打包分别记录；第一版继续保留原有文件及核验记录，因此新增工作不会改变上一次交付的含义。

7.2.2.5 Clique enumeration and maximum cliques / 团枚举与最大团

UNOFFICIAL CONTINUATION. This is an original, independently written study text organized around publicly announced TAOCP topics. It was not written, reviewed, or endorsed by Donald E. Knuth or the original publisher.

非官方续写。本文按 TAOCP 公开选题组织原创讲解，未经 Donald E. Knuth 或原出版方撰写、审阅或认可。

An answer that cannot be enlarged / 一个不能再扩大的答案

Suppose a graph consists of a triangle on a, b, c and a separate edge joining d, e. Both components give vertex sets to which no further vertex can be added while retaining pairwise adjacency. Their sizes differ. A program that lists both has solved a different problem from a program that returns only the largest size, and a program that lists every pairwise adjacent set must report many additional sets. We will give each output contract a precise meaning before using backtracking.

设一张图由顶点 a, b, c 上的三角形和另一条连接 d, e 的独立边构成，两个连通部分的顶点集都不能再添入顶点而保持两两邻接，但它们的大小不同；列出这两个集合、只返回最大的大小，以及列出所有两两邻接的点集，分别要求程序给出不同的答案。使用回溯之前，必须先把这些输出合同说清楚。

Bron and Kerbosch published Algorithm 457 in *Communications of the ACM* in 1973 [BK1973]. The three-set organization associated with that algorithm will let us retain a fact that a simple candidate list forgets: a vertex may still extend the current clique even though the present branch is no longer allowed to choose it. That fact decides whether a terminal set is maximal in the whole graph.

Bron 与 Kerbosch 于 1973 年在 *Communications of the ACM* 发表了 Algorithm 457 [BK1973]。与该算法相关的三集合组织方法保留了一项单纯候选列表容易丢失的信息：某个顶点虽然已经不许在当前分支选择，却仍可能扩充当前团，而这个事实决定了搜索终点是否在整个图中极大。

Graphs, neighborhoods, and two kinds of maximality / 图、邻域与两种最大性

Definition D1 (graph and induced subgraph). Throughout this section, $G = (V, E)$ is a finite simple undirected graph: every edge is an unordered pair of distinct vertices, and no pair occurs twice. Write $n = |V|$ and $m = |E|$. For $W \subseteq V$, the induced subgraph $G[W]$ retains precisely the vertices in W and the edges whose two endpoints lie in W. Vertices have a fixed total order;

letters in examples stand for ordered vertex identifiers. Invalid input, such as a self-loop or an asymmetric adjacency matrix, is rejected. There are no weights in this section.

定义 D1（图与诱导子图）。本节始终使用有限简单无向图 $G = (V, E)$，每条边都是两个不同顶点构成的无序对，同一对不会重复出现，并记顶点数为 $n = |V|$、边数为 $m = |E|$。对于 $W \subseteq V$，诱导子图 $G[W]$ 恰好保留 W 中的顶点，以及两端都在 W 中的原边；顶点带有固定全序，例子中的字母代表有序顶点标识。包含自环或邻接矩阵不对称等非法输入应当被拒绝，本节不引入权重。

Definition D2 (open and common neighborhoods). The open neighborhood $N(v)$ contains the vertices adjacent to v, so it never contains v itself. The degree of v is $|N(v)|$; when the graph is H, write $\deg_H(v)$ for that degree. For any subset R of V, the common neighborhood consists of the vertices outside R adjacent to every member of R. The universal condition over an empty set is true, giving the empty-set convention in (1). This definition can be evaluated even when R is not a clique; such an R will nevertheless be an invalid search state.

定义 D2（开邻域与公共邻域）。开邻域 $N(v)$ 由与 v 邻接的顶点组成，因此不含 v 本身；顶点的度就是开邻域的大小，在图 H 中记为 $\deg_H(v)$。点集 R 的公共邻域则由位于 R 外部、同时邻接 R 中每个顶点的点组成。对空集的全称条件成立，所以得到式（1）中的空集约定；即使 R 不是团，这一定义仍可计算，只是这样的 R 不能作为合法搜索状态。

$$
\begin{aligned}
&N(v) = \{w \in V : \{v, w\} \in E\},\\
&\Gamma(R) = \{w \in V \setminus R : \ \forall r \in R,\ \{w, r\} \in E\},\\
&\Gamma(\varnothing) = V.
\end{aligned}
\tag{1}
$$

Definition D3 (clique, maximal clique, and maximum clique). A clique is a subset of vertices whose distinct members are pairwise adjacent. The empty set is a clique. A clique K is maximal if no strictly larger clique contains it; equivalently, no vertex outside K can be added to it. A maximum clique has the largest cardinality among all cliques. That cardinality is the clique number $\omega(G)$; the term does not mean the number of cliques. There can be several maximum cliques.

定义 D3（团、极大团与最大团）。团是其中任意两个不同顶点都邻接的点集，空集也算团。若没有更大的团包含 K，就称团 K 为极大团；等价地，不能再从外部加入任何一个顶点。最大团则是在所有团中顶点数最多的团，其大小称为团数，记作 $\omega(G)$，这里的“团数”不是团的个数，而且最大团可以有多个并列解。

The two formulations of maximality agree because any strict containing clique supplies at least one vertex that can be added, while a single admissible addition already gives a strict containing clique. Thus a clique is maximal exactly when its common neighborhood is empty. Every maximum clique is maximal; the triangle and the separate edge at the start show why the converse fails. For an empty vertex set the only clique, the only maximal clique, and a maximum clique are all the empty set, and the clique number is zero. A nonempty edgeless graph has one maximal clique for each singleton, and clique number one.

极大性的两种说法等价，因为严格包含原团的团必定提供至少一个可加入的顶点，而一次合法添加本身就给出了严格包含关系；因此，一个团极大当且仅当其公共邻域为空。每个最大团都是极大团，但开头的三角形与独立边说明反方向不成立。空顶点图的唯一团、唯一极大团以及最大团都是空集，团数为零；非空无边图的极大团恰为各个单点集，团数为一。

The information carried by three sets / 三个集合分别保留什么

Definition D4 (legal enumeration state). A state consists of a current clique R, a candidate set P, and an excluded set X. It is legal when the three sets are pairwise disjoint and satisfy (2). The responsibility of the state is to enumerate exactly the maximal cliques of the whole graph lying in the indicated interval. The word "excluded" refers to this responsibility: X need not be the set of all vertices ever printed by the program.

定义 D4（合法枚举状态）。一个状态由当前团 R、候选集 P 和排除集 X 构成；当三者两两不交且满足式（2）时，称它合法。这个状态负责枚举的，是下述区间内属于整个图的极大团；"排除" 指的是不能进入本分支负责的答案，而不是说 X 等于程序此前输出过的所有顶点。

$$\begin{aligned} P \cup X &= \Gamma(R), \\ \mathcal{F}(R,P) &= \{K :\ K \text{ is a maximal clique of } G, \\ &\qquad R \subseteq K \subseteq R \cup P\}. \end{aligned} \tag{2}$$

Lemma L1 (state preservation). The root state with empty current and excluded sets and candidate set V is legal. In a legal state, choosing $v \in P$ gives the legal child in (3). After that child returns, moving v from the parent's candidate set to its excluded set preserves legality at the parent.

引理 L1（状态保持）。以空集作为当前团与排除集、以 V 作为候选集的根状态合法；在合法状态中选取 $v \in P$，会得到式（3）中的合法子状态，而在子调用返回后，把 v 从父状态的候选集移入排除集，也保持父状态合法。

$$(R', P', X') = (R \cup \{v\},\ P \cap N(v),\ X \cap N(v)). \tag{3}$$

Proof. Equation (1) establishes the root case. For a child, membership of v in $\Gamma(R)$ makes the enlarged current set a clique. A vertex extends that enlarged clique exactly when it extends R and is adjacent to v, so its common neighborhood is $\Gamma(R) \cap N(v)$. Intersecting (2) with the open neighborhood gives the required union; the child sets remain disjoint, and v is absent from both intersections. At the parent, transferring v changes neither the union of the two sets nor their disjointness, and does not change R. This proves all parts.

证明。根状态直接由式(1)成立；对子状态而言，v 属于 $\Gamma(R)$，因此加入它后仍为团，而一个顶点能够扩充新团，当且仅当它既能扩充 R 又邻接 v，所以新公共邻域就是 $\Gamma(R) \cap N(v)$。

将式（2）与这个开邻域取交，便得到子状态所需的并集等式；两集合仍然不交，且开邻域排除了 v 本身。回到父状态时，移动 v 既不改变两个集合的并集，也不破坏它们的不交性，R 则保持不变。■

Algorithm A (Three-set enumeration / 三集合枚举). Input is a valid graph; output is a stream of all its maximal cliques, each once. Every recursive call owns its copies of the sets. Within a call, the loop order is fixed before the loop starts, but recursive arguments use the parent's current sets.

算法 A（三集合枚举）。输入为合法图，输出流依次给出全部极大团，每个恰好一次；每次递归调用都持有自己的集合副本，同一次调用的循环次序在循环开始前固定，但递归实参必须取父状态当时的集合。

- **A1 (Start / 启动).** Call the procedure with $R = \varnothing$, $P = V$, $X = \varnothing$. / 以这三个集合作根调用。
- **A2 (Terminal test / 终止检查).** If both P and X are empty, emit R and return immediately. If only P is empty, return without output. / 两者皆空时输出当前团并立即返回；仅候选集为空时不输出而返回。
- **A3 (Snapshot / 快照).** Let S be an ordered copy of P. / 把候选集复制成有序快照。
- **A4 (Branch / 分支).** For each v in S, in order, recurse on the state in (3), using the current P, X. / 按快照次序逐点建立式（3）的子调用，交集使用实时集合。
- **A5 (Exclude and continue / 排除并继续).** After each child returns, set $P \leftarrow P \setminus \{v\}$ and $X \leftarrow X \cup \{v\}$, then continue A4. When the snapshot is exhausted, return. / 子调用返回后移动所选顶点，快照处理完即返回。

Theorem T1 (correctness of A). From any legal state, Algorithm A terminates and outputs exactly the family in (2), without repetitions. In particular, the root call enumerates all maximal cliques, including the empty clique when the vertex set is empty.

定理 T1（算法 A 的正确性）。算法 A 从任意合法状态出发都会终止，并无重复地输出式（2）中的整个集合族；特别地，根调用枚举全部极大团，在空顶点图上也会输出空团。

Proof. Each child increases the size of R by one, so a path has at most n edges. Every loop is finite. By L1, the state remains legal, and A2 emits a clique only when its entire common neighborhood is empty; every emitted set is therefore globally maximal. If P is empty and X is nonempty, the only possible set in the interval of (2) is R, and a vertex of X extends it, so the responsible family is empty.

证明。每个子调用都使 R 的大小增加一，所以一条递归路径至多有 n 条边，每个循环本身也有限。由 L1，沿途状态始终合法，而 A2 只在整个公共邻域为空时输出当前团，因此输出的团在全图中极大；若 P 为空而 X 非空，式（2）的区间里只可能有 R，但 X 中的点可以扩充它，所以该状态负责的集合族为空。

For completeness, induct on $|P|$. The terminal cases have just been settled. Fix a target K in the responsible family at a nonterminal state. It must contain a vertex outside R, since P is

nonempty and hence R is not maximal. Choose the first member v of $K \setminus R$ in the snapshot. Earlier loop vertices do not belong to K, so removing them has not removed a needed vertex. Every remaining member of $K \setminus (R \cup \{v\})$ is still in the current P and is adjacent to v. Thus K lies in the responsibility of the child (3), whose candidate set is smaller because the open neighborhood omits v. Induction places K in that child's output.

为证明完备性，对 $|P|$ 归纳，终止情形已经处理。固定非终止状态负责的任意目标团 K，因为 P 非空时 R 尚不极大，K 必须含有 R 外的顶点；取快照中第一个属于 $K \setminus R$ 的点 v，此前处理的点都不属于 K，因此移走它们没有丢失目标团所需的点。$K \setminus (R \cup \{v\})$ 中每一点仍在实时 P 中，并且邻接 v，故 K 落在式（3）子调用的责任范围内；这个子调用的候选集因为开邻域不含 v 而严格变小，归纳假设保证它输出 K。

Finally, consider two different child branches at their first divergence. The earlier branch selects a vertex v that remains in every descendant's current clique. Before the later branch begins, v has moved into X and can never enter a descendant's current clique there. Their outputs cannot coincide. Applying the induction hypothesis inside each child proves uniqueness throughout the call. The root family contains every maximal clique by its definition.

最后，考察两个搜索分支第一次分叉的位置：较早的分支选入某个顶点 v，它会一直留在该分支后代的当前团中；在较晚的分支开始之前，v 已移入 X，以后不可能在这个分支选入，因此两边不可能输出同一集合。再由每个子调用内部的归纳假设，整个调用都没有重复，而根状态的责任族按定义就是全部极大团。■

A pivot selects branches, not answers / 枢轴筛选的是分支

Lemma L2 (pivot intersection). In a legal nonterminal state with P nonempty, choose any $u \in P \cup X$, and form $S = P \setminus N(u)$ at that moment. Every clique in the state's responsible family contains at least one member of S.

引理 L2（枢轴命中）。在候选集非空的合法非终止状态中，任取 $u \in P \cup X$，并在此时构造 $S = P \setminus N(u)$，则该状态负责的每个极大团都含有 S 中的至少一个顶点。

Proof. Let K be a target. If u belongs to K, it cannot be in R or X, so it belongs to P. Because the neighborhood is open, u itself is in S, giving the claimed intersection. If u is not in K and K misses S, all members of $K \setminus R$ belong to $N(u)$. The members of R are also adjacent to u, because the pivot belongs to $\Gamma(R)$. Then adding u would enlarge K to a clique, contradicting its global maximality.

证明。设目标团为 K；若 u 属于 K，它不可能在 R 或 X 中，因而属于 P，又因为采用开邻域，u 本身就在 S 中，命中关系成立。若 u 不属于 K 而 K 与 S 不交，则 $K \setminus R$ 中的每个点都邻接 u，R 中的每个点也因为枢轴属于 $\Gamma(R)$ 而邻接它；这样就能把 u 加入 K，与全图极大性矛盾。■

Algorithm B (Pivoted enumeration / 枢轴枚举). The input, ownership of sets, root state, and streamed output contract are those of A. The following steps replace its recursive procedure. Choosing the smallest identifier breaks all pivot ties unless stated otherwise.

算法 B（枢轴枚举）。输入、集合副本的归属、根状态和流式输出合同均与 A 相同，递归过程改用下列步骤；除非另有说明，枢轴并列时选择标识最小者。

- **B1 (Terminal test / 终止检查).** If $P = X = \varnothing$, emit R and return immediately. If $P = \varnothing$ but $X \neq \varnothing$, return without output. / 两者皆空时输出并立即返回；仅候选集为空时不输出而返回。
- **B2 (Pivot / 枢轴).** Select $u \in P \cup X$ maximizing $|P \cap N(u)|$, breaking ties by the fixed order. / 在候选集与排除集的并集中，选取邻接当前候选点最多的枢轴，并按固定次序打破并列。
- **B3 (Snapshot / 快照).** Copy $P \setminus N(u)$ into an ordered immutable sequence S. / 把差集复制为有序且不再改变的循环快照。
- **B4 (Branch / 分支).** For each v in S, recurse on (3) using the current P, X. / 按快照次序，用实时集合建立式（3）的子调用。
- **B5 (Exclude and continue / 排除并继续).** After that child, move v from P to X. Return when all members of S have been processed. / 子调用返回后将所选点从候选集移入排除集，快照处理完即返回。

Theorem T2 (correctness of B). Algorithm B has the same output and termination guarantees as A. Correctness does not require the maximizing choice in B2; that choice minimizes the number of immediate child branches for the current state, but need not minimize the whole search tree.

定理 T2（算法 B 的正确性）。算法 B 具有与 A 相同的输出和终止保证；正确性不依赖 B2 的最大化选择，这一选择仅使当前状态的直接子分支数最少，并不保证整棵搜索树最小。

Proof. Legality, termination, and validity of every output follow as in L1 and T1. For any target in the entry responsibility, L2 supplies a nonempty intersection with the fixed snapshot. Let v be the first vertex of that intersection. Earlier snapshot vertices do not belong to the target; their moves to X therefore leave all its needed vertices in the current P. After intersecting with $N(v)$, the child still contains the target in its responsible family. Its smaller candidate set allows induction. Later siblings cannot select v, so the first-divergence argument again prevents repetition. If the snapshot is empty, L2 says the responsible family is empty, justifying a return without children. The branch count is $|P| - |P \cap N(u)|$, so maximizing the second term proves only the stated local minimization.

证明。合法性、终止性和每项输出的有效性与 L1、T1 相同；对入口责任族中的任意目标团，L2 保证它与固定快照的交集非空，取该交集中最早的点 v，则此前移入 X 的快照顶点都不属于目标团，目标团需要的其余顶点仍在实时 P 中。与 $N(v)$ 取交以后，目标团仍处于该子调用的责任族内，候选集严格变小，因而可以归纳；后来的兄弟分支不能再选 v，第一次分叉论证也就再次排除了重复。若快照为空，L2 表明责任族为空，所以

直接返回合法；当前分支数等于 $|P| - |P \cap N(u)|$，最大化第二项只证明了所述局部最小化。■

Freezing the loop sequence does not freeze the state. If earlier vertices remain available to later children because old copies of P, X are reused, the uniqueness argument no longer applies. Replacing the open neighborhood by a closed one is another error: when the pivot is itself a needed candidate, it must be allowed to start a branch. Exercise 8 isolates both faults on small graphs.

冻结循环序列并不意味着冻结状态；如果建立后续子调用时仍使用旧的 P, X，让已经处理的顶点继续可选，无重复性的论证就失去了依据。把开邻域换成闭邻域也会出错，因为当枢轴本身就是目标团需要的候选点时，它必须有机会启动分支，习题 8 将分别用小图定位这两类故障。

Paying for calls as well as output / 调用和输出都要计费

Assume vertices are numbered from one to n, adjacency is stored in a Boolean matrix with constant-time entry access, and every set is a Boolean array of length n or an ordered list scanned in at most linear time. Set construction, copying, intersections, and iteration are charged. An adjacency-list input can be converted and checked in $O(n^2+m)$ time and $O(n^2)$ space; a supplied matrix still takes quadratic time to validate. The empty vertex set is handled separately in constant time.

假定顶点编号为一至 n，邻接关系存为可常数时间访问单项的布尔矩阵，每个集合则使用长度为 n 的布尔数组或至多在线性时间扫描的有序列表；集合构造、复制、求交与迭代均计入成本。邻接表输入可以在 $O(n^2 + m)$ 时间和 $O(n^2)$ 空间内转换并核验，已有矩阵仍需要二次时间验证；空顶点图单独以常数时间处理。

Call-count argument. In either A or B, different recursive calls have different current sets R. Calls on one path have strictly increasing cardinalities. Calls on distinct paths have a first divergence: the earlier child adds a vertex that the later child can never add, by the same transfer used in T1. Thus their final current sets cannot be equal. There are at most as many calls as subsets of V, including the root; this bound counts failed and internal calls as well as output leaves.

调用计数论证。在 A 或 B 中，不同递归调用具有不同的当前集合 R：同一路径上的集合大小严格增加，而不同路径存在第一次分叉，较早子分支加入的顶点在较晚子分支中永远不能加入，这正是 T1 使用的移动性质，所以它们的当前集合不可能相等。因此，包含根在内的调用数不超过 V 的子集个数；这个界同时计算了失败调用、内部调用和输出叶，并非只数输出。

Let L be the total number of vertex identifiers in the output lists. For $n \geq 1$, the following conservative bound applies to both algorithms. At a B-call, testing all pivot candidates against all vertices of P costs at most quadratic time; at an A-call this work is absent. Loop setup costs linear time, and the linear copying and intersection work of each tree edge can be charged to its child. Therefore a quadratic charge per call suffices. Nonempty graphs have no empty maximal

clique, so record overhead is also covered by the output term. The empty graph emits one empty record in constant time even though its L is zero.

令 L 表示输出列表中顶点标识的总数，当 $n \geq 1$ 时，两种算法都满足下面的保守界。B 的每次调用把所有枢轴候选与 P 中顶点逐一核对，至多花二次时间，A 没有这项工作；循环准备花线性时间，每条递归树边上的线性复制与求交可以记在子调用账上，因此每次调用收取二次成本已经足够。非空顶点图没有空的极大团，输出记录本身的开销也能计入输出项；空顶点图则以常数时间输出一个空记录，即使此时 L 为零。

$$\begin{aligned} \text{calls} &\leq 2^n, \\ \text{time} &= O(n^2 2^n + L), \\ \text{auxiliary space} &= O(n^2). \end{aligned} \tag{4}$$

The space bound follows from at most $n + 1$ active levels, each holding a constant number of linear-size sets and a snapshot. The adjacency matrix also uses quadratic space. Output is streamed; storing every result would add space proportional to L and its record overhead. Equation (4) makes no claim of polynomial delay between outputs. A bitset implementation must specify a word width w and charge for processing up to $\lceil n/w \rceil$ words per set operation, rather than treating an arbitrarily large bitset as one constant-time value.

空间界来自至多 $n + 1$ 个活动层，每层持有常数个线性大小的集合与一个快照，邻接矩阵自身也占二次空间。这里采用流式输出，若把全部结果留在内存，还须增加与 L 及记录开销对应的空间；式 (4) 也没有给出相邻两次输出之间的多项式延迟保证。若改用位集，就必须说明机器字宽 w，并对一次集合操作至多处理的 $\lceil n/w \rceil$ 个机器字计费，不能把任意大的位集当成常数时间的单值。

Following the same graph through A and B / 在同一张图上追踪 A 与 B

Use the following original instance, with alphabetical order for branching and pivot ties. A string such as bc in a table means the set consisting of those letters; the empty-set symbol means an empty set. Figure 1, the edge set, and Table 1 specify the same graph, including the isolated vertex g.

使用下面这个原创实例，分支和枢轴并列都按字母顺序处理；表中的 bc 之类字符串表示由这些字母构成的集合，空集符号表示空集合。图 1、边集和表 1 描述的是同一张图，其中包括孤立点 g。

$$V = \{a, b, c, d, e, f, g\}, \qquad E = \{ab, ac, bc, bd, cd, de, ef\}.$$

Figure 1. Worked graph, with isolated vertex g / 图 1. 含孤立点 g 的主例图

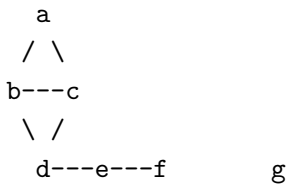


Table 1. Adjacency list of the worked graph / 表 1. 主例邻接表

Vertex / 顶点	Open neighbors / 开邻域
a	b, c
b	a, c, d
c	a, b, d
d	b, c, e
e	d, f
f	e
g	∅

The first table for each algorithm records entry states, not states after sibling calls. An arrow identifies a selected vertex and its child call. The second table gives every return-time move in chronological order. A dash means that the algorithm returns before constructing that field. These conventions distinguish the fixed snapshot from the changing parent sets.

每种算法的第一张表记录调用入口状态，而不是兄弟调用完成后的状态，箭头指出所选顶点与对应子调用编号；第二张表按实际发生顺序列出每次返回后的移动。短横线表示算法已提前返回，因而没有构造相应字段；这样可以明确区分快照与不断变化的父状态集合。

A: all candidate branches / A：遍历候选分支

Table 2. Algorithm A entry states / 表 2. 算法 A 入口状态

Call / 调用	Depth / 深度	R	P	X	u	S	Children / 子调用	Result / 结果
1	0	∅	abcdefg	∅	—	abcdefg	a→2, b→6, c→10, d→12, e→14, f→16, g→17	return / 返回
2	1	a	bc	∅	—	bc	b→3, c→5	return / 返回
3	2	ab	c	∅	—	c	c→4	return / 返回
4	3	abc	∅	∅	—	—	—	output / 输出
5	2	ac	∅	b	—	—	—	excluded / 被排除
6	1	b	cd	a	—	cd	c→7, d→9	return / 返回
7	2	bc	d	a	—	d	d→8	return / 返回
8	3	bcd	∅	∅	—	—	—	output / 输出
9	2	bd	∅	c	—	—	—	excluded / 被排除
10	1	c	d	ab	—	d	d→11	return / 返回
11	2	cd	∅	b	—	—	—	excluded / 被排除
12	1	d	e	bc	—	e	e→13	return / 返回
13	2	de	∅	∅	—	—	—	output / 输出
14	1	e	f	d	—	f	f→15	return / 返回
15	2	ef	∅	∅	—	—	—	output / 输出

Call / 调用	Depth / 深度	R	P	X	u	S	Children / 子调用	Result / 结果
16	1	f	$\varnothing$	e	—	—	—	excluded / 被排除
17	1	g	$\varnothing$	$\varnothing$	—	—	—	output / 输出

Table 3. Algorithm A return-time moves / 表 3. 算法 A 返回后的移动

Parent / 父调用	Returned vertex / 返回点	New P / 新候选集	New X / 新排除集
3	c	$\varnothing$	c
2	b	c	b
2	c	$\varnothing$	bc
1	a	bcdefg	a
7	d	$\varnothing$	ad
6	c	d	ac
6	d	$\varnothing$	acd
1	b	cdefg	ab
10	d	$\varnothing$	abd
1	c	defg	abc
12	e	$\varnothing$	bce
1	d	efg	abcd
14	f	$\varnothing$	df
1	e	fg	abcde
1	f	g	abcdef
1	g	$\varnothing$	abcdefg

Call 5 does not output ac, because b is still a common neighbor even though this branch excludes it. Call 17 does output the isolated singleton. The five outputs are abc, bcd, de, ef, g. There are 17 calls: one for the empty set, seven for singletons, seven for edges, and two for triangles.

调用 5 不输出 ac，因为虽然这个分支排除了 b，它仍是公共邻居；调用 17 则输出孤立单点。五项输出分别为 abc, bcd, de, ef, g，总共有 17 次调用，分别对应一个空集、七个单点、七条边和两个三角形。

B: a smaller set of starting vertices / B：缩小分支起点集合

Table 4. Algorithm B entry states / 表 4. 算法 B 入口状态

Call / 调用	Depth / 深度	R	P	X	u	S	Children / 子调用	Result / 结果
1	0	∅	abcdefg	∅	b	befg	b→2, e→6, f→9, g→10	return / 返回
2	1	b	acd	∅	c	c	c→3	return / 返回
3	2	bc	ad	∅	a	ad	a→4, d→5	return / 返回
4	3	abc	∅	∅	—	—	—	output / 输出
5	3	bcd	∅	∅	—	—	—	output / 输出
6	1	e	df	∅	d	df	d→7, f→8	return / 返回
7	2	de	∅	∅	—	—	—	output / 输出
8	2	ef	∅	∅	—	—	—	output / 输出
9	1	f	∅	e	—	—	—	excluded / 被排除
10	1	g	∅	∅	—	—	—	output / 输出

Table 5. Algorithm B return-time moves / 表 5. 算法 B 返回后的移动

Parent / 父调用	Returned vertex / 返回点	New P / 新候选集	New X / 新排除集
3	a	d	a
3	d	$\varnothing$	ad
2	c	ad	c
1	b	acdefg	b
6	d	f	d
6	f	$\varnothing$	df
1	e	acdfg	be
1	f	acdg	bef
1	g	acd	befg

At the root, b, c, d all have three candidate neighbors, so the tie rule selects b. Its open neighborhood omits b itself, leaving the snapshot $befg$. After all these branches, the root's P still contains acd; returning is correct because L2 has already assigned every target to a snapshot branch. B uses 10 calls on this graph and produces the same five outputs. These counts describe this fixed implementation and instance, rather than a universal speed ratio.

根状态中 b, c, d 都有三个候选邻居，因而按并列规则选择 b；其开邻域不含自身，所以快照为 $befg$。所有这些分支完成后，根状态的 P 仍含 acd，此时返回依然正确，因为 L2 已经把每个目标团交给某个快照分支负责；B 在这张图上使用 10 次调用，并给出相同的五项输出。这些数字仅描述固定实现与固定实例，不代表普遍的速度比例。

Searching for one largest clique / 寻找一个最大团

For optimization, a globally maximal output is unnecessary at every search leaf. Keep a feasible incumbent clique B_*, initially empty, and let a state (R, P) represent all clique extensions between R and $R \cup P$, with $P \subseteq \Gamma(R)$. This time P need not cover the whole common neighborhood: vertices omitted by earlier choices simply belong to other search branches. The incumbent is an actual stored clique, never a guessed objective value.

在优化问题中，没有必要要求每个搜索叶都对应全图极大团。保存一个真实可行的当前最好团 B_*，初值为空集，并用状态 (R, P) 表示位于 R 与 $R \cup P$ 之间的所有团扩充，其中 $P \subseteq \Gamma(R)$；这里不要求 P 覆盖整个公共邻域，因为此前选择排除的顶点归属于其他搜索分支。当前最好解必须是实际保存的团，不能只是猜出的目标值。

Lemma L3 (coloring upper bound). A proper coloring assigns a color to each vertex so that adjacent vertices receive different colors. Its color classes are therefore independent sets, meaning that no two vertices within one class are adjacent. If $G[P]$ has a proper coloring with q nonempty

classes, every clique represented by (R, P) has size at most the bound in (5). Without a coloring, replacing q by $|P|$ gives a weaker valid bound.

引理 L3（着色上界）。合法着色要求相邻顶点具有不同颜色，所以每个颜色类都是独立集，即同类内部没有两个顶点邻接；若 $G[P]$ 具有包含 q 个非空颜色类的合法着色，则状态 (R, P) 表示的每个团都不超过式（5）的上界。没有着色时，用 $|P|$ 代替 q 也能得到有效但较弱的界。

$$|K| \le |R| + q \le |R| + |P|. \tag{5}$$

Proof. A clique can use at most one vertex of each independent color class; its vertices outside R all belong to P, so there are at most q of them. A partition of P into nonempty classes has at most $|P|$ classes. Neither statement says the coloring uses the fewest possible colors, or that its upper bound is attained by a clique.

证明。团在每个独立的颜色类中至多取一个顶点，而 R 以外的顶点全部来自 P，因此至多增加 q 个顶点；将 P 分为非空类时，类的个数也不超过 $|P|$。这两个论断都没有声称着色使用了最少颜色，也没有声称某个团一定达到该上界。■

We use first-fit greedy coloring: process the vertices of P in fixed order, place each in the first existing class containing no neighbor of it, and create a new class if no existing class works. Each insertion preserves independence of its class. Store the classes as disjoint lists; for each new vertex, testing all attempted classes inspects each previously colored vertex at most once. Thus a coloring costs $O(n^2)$ in the stated model, with linear space for the class memberships. The certificate is the partition itself, which can be checked directly.

这里采用首次适配的贪心着色：按固定次序处理 P 中的顶点，将每个点放入第一个不含其邻居的已有颜色类，若没有合适的类就新建一类；每次放入都保持该类独立。把颜色类存为互不相交的列表，每处理一个新点，即使检查了所有尝试过的类，对每个先前着色顶点也至多检查一次，因此在既定模型下每次着色的成本为 $O(n^2)$，记录类成员需要线性空间，而划分本身就是可以直接核对的上界证书。

Algorithm C (Maximum-clique search / 最大团搜索). Return one maximum clique and its cardinality. A recursive call owns its own R, P, while all calls share the feasible incumbent. Ties leave the earlier incumbent unchanged.

算法 C（最大团搜索）。返回一个最大团及其大小；每次递归持有自己的 R, P，所有调用共享真实可行的当前最好解，大小并列时保留较早得到的解。

- **C1 (Start / 启动).** Set $B_* = \varnothing$ and call the procedure with $R = \varnothing, P = V$. Return the final B_* and its size. / 初始化并启动根调用，最后返回保存的团与大小。
- **C2 (Update and stop / 更新与终止).** If $|R| > |B_*|$, copy R into B_*. If $P = \varnothing$, return. / 仅在当前团严格较大时复制更新，候选集为空即返回。
- **C3 (Color and bound / 着色与定界).** Greedily color $G[P]$ into q classes. If $|R| + q \le |B_*|$, return. / 若着色上界不超过已有可行解大小，结束此分支。

- **C4 (Include / 选入).** Let v be the first vertex of P. Recurse on $(R \cup \{v\}, P \cap N(v))$. / 选取首个候选点，搜索包含它的扩充。
- **C5 (Exclude / 排除).** Recurse on $(R, P \setminus \{v\})$, using the parent's unchanged sets, and return. / 使用父调用原有集合，搜索不包含该点的扩充，然后返回。

Theorem T3 (correctness of C). Algorithm C terminates with a feasible maximum clique, including the empty-vertex and tied-optimum cases under its one-solution contract.

定理 T3（算法 C 的正确性）。算法 C 会终止并返回一个真实最大团；在只求一个解的合同下，空顶点图以及多个并列最优解也包含在结论中。

Proof. The root current set is a clique. Inclusion uses a common neighbor and intersects remaining candidates with its open neighborhood; exclusion leaves the clique unchanged. Hence every current set and every copied incumbent is feasible. Every child removes at least one candidate, giving termination. For a state not pruned, its represented extensions split into disjoint families according to whether they contain v: C4 represents exactly the first family and C5 exactly the second. If P is empty, C2 has considered the only represented extension, namely R.

证明。根状态当前集合是团，选入操作使用公共邻居并把剩余候选与它的开邻域取交，排除操作则不改变当前团，因此每个当前集合以及每次复制得到的最好解都可行；每个子调用至少删去一个候选点，所以搜索终止。对没有被剪去的状态，其全部团扩充按是否含 v 分成不交的两族，C4 与 C5 恰好分别表示这两族；若 P 为空，C2 已经考察了唯一可能的扩充，也就是 R 本身。

At a pruned state, L3 bounds every represented extension by the size of a feasible incumbent already found. Later updates can only increase that size, so no pruned state could improve the final answer. Induct over the finite search tree, using the partition at unpruned nodes and this inequality at pruned nodes: no clique represented at the root is larger than the final incumbent. Since that incumbent is itself a clique, it is maximum. For an empty graph the initial current set is the sole clique and is returned; if several maxima exist, the inequality permits discarding tied branches because one feasible optimum suffices.

在剪枝状态，L3 保证它所表示的每个扩充都不大于已经找到的真实最好解，而后续更新只会使这个大小增加，所以被剪去的分支不可能改善最终答案。在有限搜索树上归纳，对未剪枝节点使用两族划分，对剪枝节点使用上述不等式，就知道根状态表示的任何团都不大于最终最好解；后者自身可行，故它就是最大团。空顶点图直接返回初始空团；存在多个最大团时，剪去相等上界的分支仍然合法，因为合同只要求一个最优解。■

C has at most $2^{n+1}-1$ calls: if at most k candidates remain, a call has at most two children with at most $k-1$ candidates, so induction on k, starting with one call at zero, gives this bound. Unlike A and B, C can revisit the same R after exclusions, so the earlier distinct-current-set argument is not used. A quadratic charge for coloring and set work gives $O(n^2 2^n)$ time for $n \geq 1$, quadratic auxiliary space including the matrix, and constant time at $n = 0$. Pruning can reduce the actual tree without changing this worst-case guarantee.

C 的调用数至多为 $2^{n+1}-1$，因为剩余候选至多为 k 时，一个调用至多产生两个候选数不超过 $k-1$ 的子调用，从零候选时的一次调用开始归纳即可得到该界。C 在排除顶点后可能再次使用同一个 R，因此这里不能套用 A、B 的当前集合互异论证；把着色与集合处理按每次调用二次成本计费，便得到 $n \geq 1$ 时的 $O(n^2 2^n)$ 时间，以及包含矩阵的二次辅助空间，$n = 0$ 时则为常数时间。剪枝可以缩小实际搜索树，但不改变这个最坏情形保证。

For a worked optimization trace, take vertices a, b, c, d, e, f and edges ab, ac, bc, bd, be, bf. Its triangle shares b with three extra leaves. The table records the incumbent after C2 and uses the same letter-set notation. At a leaf, the bound column shows the exact size of the sole represented extension; no coloring is performed there.

优化追踪另取顶点 a, b, c, d, e, f 和边 ab, ac, bc, bd, be, bf，也就是在三角形的 b 点上再接三个叶子；下表记录 C2 更新后的当前最好团，并沿用字母集合记法。叶子行的上界栏给出唯一扩充的准确大小，这些行没有执行着色。

Table 6. Algorithm C coloring and pruning trace / 表 6. 算法 C 着色与剪枝追踪

Call / 调用	R	P	B_*	Classes / 颜色类	Bound / 上界	Action / 动作
1	∅	abcdef	∅	adef; b; c	3	branch / 分支 a
2	a	bc	a	b; c	3	branch / 分支 b
3	ab	c	ab	c	3	branch / 分支 c
4	abc	∅	abc	—	3	leaf / 叶
5	ab	∅	abc	—	2	leaf / 叶
6	a	c	abc	c	2	prune / 剪枝
7	∅	bcdef	abc	b; cdef	2	prune / 剪枝

At call 7, the remaining five candidates form a star, with color classes b and $cdef$. A bound based only on candidate count would be five and would not prune against the stored triangle. The coloring gives two, so the branch is pruned. The triangle was actually constructed at call 4; without that feasible incumbent, the same upper bound would not justify the same decision.

调用 7 的五个剩余候选构成星形图，颜色类为 b 与 $cdef$；若只按候选数定界，上界为五，还不能凭已有三角形剪枝，而着色把上界降为二，因此可以结束分支。作为下界的三角形是在调用 4 实际构造出来的，若没有这个真实可行解，相同的上界并不支持相同的剪枝决定。

Sparse graphs and a different counting question / 稀疏图与另一个计数问题

Definition D5 (degeneracy and degeneracy ordering). The degeneracy d of a nonempty graph is the least nonnegative integer such that every nonempty subgraph has a vertex of degree at most d within that subgraph. Set $d = 0$ for the empty vertex graph. A degeneracy ordering is an ordering of the vertices in which each vertex has at most d neighbors occurring later. It suffices to test the defining property on induced subgraphs: deleting further edges can only decrease degrees.

定义 D5（退化度与退化度排列）。非空图的退化度 d 是满足下述条件的最小非负整数：每个非空子图都含有一个在该子图内度数不超过 d 的顶点；空顶点图约定退化度为零。退化度排列是每个顶点至多具有 d 个后继邻居的顶点排列，定义中的子图条件只须对诱导子图检查，因为继续删边只能降低度数。

Lemma L4 (ordering characterization). The degeneracy condition with a nonnegative bound d holds exactly when such an ordering exists. Every nonempty graph of degeneracy d has clique number at most $d + 1$; the empty graph satisfies the same inequality under our convention.

引理 L4（排列刻画）。给定非负界 d，退化度定义中的度数条件成立，当且仅当存在这样的排列；每个退化度为 d 的非空图，其团数不超过 $d+1$，空顶点图在既定约定下也满足此不等式。

Proof. Repeatedly remove a vertex of degree at most d from the remaining induced graph, and list vertices in removal order. At the moment a vertex is removed, its remaining neighbors are exactly its later neighbors, proving one direction. Conversely, in a nonempty subgraph take the earliest vertex of the ordering present in that subgraph. All its subgraph neighbors occur later, so it has at most d of them. For a clique of positive size, its earliest vertex has every other clique member as a later neighbor; the size is therefore at most $d + 1$. In the empty graph the clique number is zero.

证明。从剩余诱导图中反复删除一个度数不超过 d 的顶点，并按删除次序排列；每个点被删除时尚存的邻居恰好就是它的后继邻居，因此得到一个方向。反过来，对任意非空子图，取它在排列中最早的顶点，其子图邻居全部位于后方，数量便不超过 d；再对非空团取最早点，其余团成员全是后继邻居，所以团的大小至多为 $d+1$，空顶点图的团数则为零。■

Small clique number does not mean few maximal cliques. For a multiple of three d and $n \geq d+3$, Exercise 15 proves that the largest possible number of maximal cliques in an n-vertex graph of degeneracy d is $(n-d)3^{d/3}$, and constructs a graph attaining it. This is the extremal counting statement on page 8 of [ELS2010-v1], developed here through a complete deletion argument. It counts cliques, not identifiers in the output lists, and it is not a running-time theorem for our elementary implementation.

团数较小并不意味着极大团较少；当 d 是三的倍数且 $n \geq d+3$ 时，习题 15 将通过完整的删除论证，证明退化度为 d 的 n 点图最多可以有 $(n-d)3^{d/3}$ 个极大团，并构造达到此

数目的图。这是 [ELS2010-v1] 第 8 页的极值计数结论，计算的是团的个数，不是输出列表中的顶点标识总数，也不是本节朴素实现的运行时间定理。

> **UNOFFICIAL CONTINUATION.** This is an original, independently written study text organized around publicly announced TAOCP topics. It was not written, reviewed, or endorsed by Donald E. Knuth or the original publisher.
>
> **非官方续写**。本文按 TAOCP 公开选题组织原创讲解，未经 Donald E. Knuth 或原出版方撰写、审阅或认可。

Exercises / 习题

The difficulty labels are this project's teaching estimates, not ratings assigned or certified by the TAOCP author. M marks an exercise centered on mathematical proof. References D1–D5, L1–L4, T1–T3, and equations (1)–(5) belong to this section. Exercise 15 uses the complement correspondence established in Exercise 13 and the ordering characterization in L4; all additional tools are specified in its statement.

难度标签是在本工程内作出的教学估计，并非 TAOCP 作者给出或认证的评级，M 表示以数学证明为主；D1—D5、L1—L4、T1—T3 以及式（1）至（5）均指本节内容。习题 15 使用习题 13 建立的补图对应和 L4 的排列刻画，所需的其余工具均在题干中给出。

1. [05] Two kinds of maximality / 两种最大性. In the graph with edges ab, ac, bc, de and vertex set $\{a, b, c, d, e\}$, classify the sets $\varnothing, \{a, b\}, \{a, b, c\}, \{d, e\}, \{a, d\}$ as cliques, maximal cliques, and maximum cliques. For each negative classification, give the missing edge, available extension, or larger competing clique that proves it.

在顶点集为 $\{a, b, c, d, e\}$、边为 ab, ac, bc, de 的图中，把题列各集合分别按团、极大团与最大团分类；每次否定都要给出作为证据的缺边、可扩充顶点或更大竞争团。

2. [10] Empty sets and isolated vertices / 空集与孤立点. Specify the exact outputs of A, B, and C on an empty vertex graph and on a nonempty edgeless graph. If a new isolated vertex is added to an arbitrary graph, describe the new maximal-clique family and clique number, distinguishing an originally empty graph from a nonempty one.

写出 A、B、C 在空顶点图与非空无边图上的准确输出；再给任意图加入一个新孤立点，描述极大团族与团数如何变化，并区分原图为空和非空的情况。

3. [15] Common neighborhoods do not certify a clique / 公共邻域不能证明当前集合是团. On the seven-vertex worked graph, compute the common neighborhoods of the empty set, $\{b\}$, $\{b, c\}$, $\{a, d\}$, and $\{g\}$. Which proposed current sets are illegal, and why would the neighborhood calculation alone miss the fault?

在七点主例上，计算题列五个集合的公共邻域；指出哪些集合作为当前团不合法，并说明为什么只计算公共邻域发现不了这个错误。

4. [20] Every return-time move / 每次返回后的移动. Run A on the path with vertices a, b, c and edges ab, bc, in alphabetical order. List every entry state and every parent move from P to X. Explain exactly why the last singleton is not output.

在顶点为 a, b, c、边为 ab, bc 的路径上按字母顺序运行 A，列出每个入口状态与父调用中每次从 P 到 X 的移动，并准确解释最后一个单点为什么没有输出。

5. [20] Removing the excluded set / 删去排除集. Modify A by discarding X, retaining the removal of each processed vertex from P, and outputting whenever P is empty. Give a smallest graph on which a nonmaximal clique is output, and decide whether this particular variant can output the same set twice. Then analyze the further modification that restores all original common neighbors for every child.

将 A 改成不保存 X、仍从 P 中删去已处理顶点、并在 P 为空时输出；给出它会输出非极大团的最小图，判断这个特定变体是否还会输出重复集合，再分析进一步将每个子调用候选集恢复为全部原始公共邻居的修改。

6. [20] Disjoint but incomplete / 不交却不完整. On a two-vertex edge, give pairwise disjoint triples with a clique as R for which omitting a common neighbor causes a false output and omitting it at the root causes a missed maximal clique. State the actual responsible family in each case, and identify the failed clause of D4.

在只有两个端点的一条边上，给出 R 为团且三集合两两不交的状态，分别使漏掉一个公共邻居导致错误输出，以及在根部遗漏它导致漏报极大团；写出每个状态实际应负责的集合族，并指出 D4 中失效的条件。

7. [25] Pivots from either set / 枢轴的两种来源. On a triangle a, b, c, trace B from the root with alphabetical ties, explaining why a pivot in P belongs to its own branch snapshot. Then start from the legal state $R = \{c\}, P = \{b\}, X = \{a\}$, choose pivot a, and explain the snapshot and the correct output responsibility.

在三角形 a, b, c 上从根按字母并列规则追踪 B，解释位于 P 的枢轴为什么属于自己的分支快照；然后从题给合法状态出发选取枢轴 a，说明快照内容与准确的输出责任。

8. [25] Three broken pivot routines / 三种错误枢轴过程. Supply a graph and a failing execution for each fault: **(a)** using the closed neighborhood instead of the open neighborhood in B3; **(b)** deleting the immediate return after an output and proceeding to B2; **(c)** using entry-time copies of P, X for every child in B4, though moving vertices in the parent's live sets. For **(c)** you may choose any legal pivot, as permitted by T2; give both the erroneous repeated output and the correct output family.

分别给出图与具体失败过程：**(a)** 在 B3 用闭邻域替代开邻域；**(b)** 删除输出后的立即返回并继续进入 B2；**(c)** 虽然仍在父状态的实时集合中移动顶点，却在 B4 每次建立子调用时都使用入口处的 P, X 副本。第三小问可按 T2 选任意合法枢轴，必须写出错误的重复输出与正确的输出族。

9. [30] Ties and the cost of a search tree / 并列规则与搜索树成本. **(a)** On the seven-vertex worked graph, retain alphabetical branch order but reverse only the pivot tie rule. List the re-

cursive current sets in depth-first entry order and compare the call count and outputs with the original B trace. Explain why this comparison proves no universal optimality of either tie rule.

(a) 在七点主例上保留字母分支顺序，只把枢轴并列规则反转；按深度优先入口次序列出递归当前集合，与正文 B 的调用数和输出比较，并说明这个比较为什么不能证明任何一种并列规则普遍最优。

(b) Let K_n be the complete graph on n ordered vertices, with an edge between every pair of distinct vertices. For $n \geq 1$, prove that A makes exactly 2^n calls and B exactly $n+1$, although each emits only one clique and has total output-identifier length $L = n$. Use the Boolean adjacency matrix of the text and length-n Boolean arrays for sets; generate ordered member lists by scanning these arrays, and score every pivot candidate against every current candidate, without shortcuts for complete graphs. Charge input validation, set copies and intersections, snapshots, pivot scoring, and output. Derive time and depth-first auxiliary-space upper bounds for each algorithm, and explain why one output does not imply linear running time. Treat $n = 0$ separately.

(b) 设 K_n 是 n 个有序顶点构成的完全图，其中任意两个不同顶点之间都有边；对 $n \geq 1$，证明 A 恰有 2^n 次调用、B 恰有 $n+1$ 次调用，虽然两者都只输出一个团，输出顶点标识的总长度也都为 $L = n$。使用正文的布尔邻接矩阵，以长度为 n 的布尔数组存储集合，并通过扫描数组生成有序成员列表；给每个枢轴候选评分时，都逐一检查当前候选点，不针对完全图使用捷径。把输入核验、集合复制与求交、快照、枢轴评分及输出全部计费，分别推出时间和深度优先辅助空间的上界，说明为什么只有一个输出不能推出线性运行时间，并单独处理 $n = 0$。

(c) For C on an arbitrary n-vertex graph, derive the bound $2^{n+1} - 1$ on the number of calls from a recurrence in the number of remaining candidates. Give the corresponding time and auxiliary-space bounds in the same model, including $n = 0$, and explain why A and B's distinct-current-set argument cannot be transferred to C.

(c) 对任意 n 点图上的 C，以剩余候选数为参数建立递推，推出调用数上界 $2^{n+1} - 1$；在同一模型下给出相应的时间与辅助空间界，包括 $n = 0$，并解释为什么不能把 A、B 的当前集合互异论证搬到 C。

10. [25] A feasible lower bound / 一个真实可行的下界. Prove that C preserves feasibility of its incumbent and that the cardinality bound alone justifies pruning when it cannot exceed that incumbent. On a single edge, show how an invented initial incumbent size can suppress the optimum even if all upper bounds are valid.

证明 C 始终保存可行的当前最好团，并证明单凭顶点数上界，只要不能超过该最好团就足以剪枝；再在一条边上说明，即使上界全都有效，虚构一个初始最好解大小也能使最优解被剪去。

11. [25] All tied maxima / 全部并列最大团. Change C's contract to list every maximum clique exactly once. Give an explicit two-edge graph showing why the original non-strict pruning test loses a tie. Describe a correct strict pruning test, when to collect solutions, and how to clear or append the solution list without counting the same R repeatedly. Include the empty vertex graph.

将 C 的合同改为恰好一次地列出所有最大团，用明确的两条边图说明原来的非严格剪枝条件为什么会丢掉并列解；给出正确的严格剪枝条件、收集解的时机，以及清空或追加解列表的规则，使重复出现的当前集合不会被重复计算，并处理空顶点图。

12. [30] A coloring certificate may be loose / 着色证书可以不紧. Verify the two color classes in call 7 of the optimization example and explain the strict improvement over the candidate-count bound. Then greedily color the five-cycle with edges ab, bc, cd, de, ea in alphabetical order. Determine its clique number and prove that the resulting color bound is valid but unattained.

核验优化示例调用 7 的两个颜色类，解释它怎样严格改进候选数上界；再按字母顺序给边为 ab, bc, cd, de, ea 的五边形贪心着色，确定其团数，并证明所得颜色上界虽然有效，却没有团达到它。

13. [30] Taking complements / 取补图. Define the complement $\overline{G}$ on the same vertices by making two distinct vertices adjacent exactly when they are not adjacent in G. An independent set has no adjacent pair; it is maximal when it cannot be enlarged, and maximum when its size is largest. Prove the clique–independent-set correspondence for ordinary, maximal, and maximum sets, including the empty vertex graph.

在相同顶点上定义补图 $\overline{G}$，两个不同顶点在补图中邻接当且仅当它们在原图中不邻接；独立集内部没有相邻点对，按包含关系不能扩大时称为极大，大小达到最大时称为最大。证明普通、极大与最大三种层次的团—独立集对应，并包含空顶点图。

14. [30] Degeneracy is not maximum degree / 退化度不是最大度. On the seven-vertex worked graph, repeatedly delete a vertex of minimum current degree, breaking ties alphabetically. Record the deletion degrees and obtain the exact degeneracy. Prove the ordering characterization from this deletion process and give a family whose maximum degree grows without bound while degeneracy stays one.

在七点主例上反复删除当前度数最小的顶点，按字母顺序打破并列，记录删除时的度数并确定准确退化度；由这个删除过程证明排列刻画，再给出最大度无界增长而退化度始终为一的图族。

15. [M40] From local deletion to an extremal clique count / 从局部删除到极大团极值. Establish the following result by proving all three parts; citing a general maximal-clique counting theorem does not solve the exercise. For a graph H with vertex set $V(H)$, write $N_H[v] = \{v\} \cup N_H(v)$ for its closed neighborhood, and let $f(H)$ count its maximal independent sets, with value one on an empty vertex set. You may use induction, the fact that a union of finite families has size at most the sum of their sizes, Exercise 13, and L4. A complete multipartite graph $K_{a_1,\dots,a_t}$ has disjoint nonempty parts of the indicated sizes, no edges within a part, and all edges between different parts.

通过完成下面三个相互依赖的小问证明极值结论，直接引用一般极大团计数定理不算解答。对顶点集为 $V(H)$ 的图 H，以 $N_H[v] = \{v\} \cup N_H(v)$ 表示闭邻域，以 $f(H)$ 表示极大独立集个数，空顶点图上的值约定为一；可以使用归纳法、有限集合族之并的基数不

超过各族基数之和、习题 13 以及 L4。完全多部图 $K_{a_1,\dots,a_t}$ 由相应大小的互不相交非空部组成，同部没有边，不同部之间全部邻接。

(a) Prove $k \le 3^{k/3}$ for every positive integer k without calculus. Next prove that every n-vertex graph has at most $3^{n/3}$ maximal cliques, including $n = 0$. For the independent-set formulation, choose a minimum-degree vertex v, partition or cover the maximal independent sets according to their intersection with $N_H[v]$, and justify exactly what remains after deleting the closed neighborhood of a chosen member. Overlapping families must not be counted as disjoint.

(a) 不使用微积分，证明每个正整数 k 都满足题列数值不等式，再证明每个 n 点图的极大团个数不超过 $3^{n/3}$，包括 $n = 0$。转为独立集问题时，可选择最小度顶点 v，按极大独立集与 $N_H[v]$ 的交集作分族或覆盖，但必须准确证明删去所选成员的闭邻域以后还剩下什么，不能把可能重叠的族当成不交划分。

(b) Let d be a nonnegative multiple of three and $n \ge d + 3$. Prove that a graph of degeneracy d has at most $(n - d)3^{d/3}$ maximal cliques. Use a degeneracy ordering, treat the first $n - d - 3$ vertices by earliest-vertex classes, and treat the last $d+3$ vertices together. Explain why a globally maximal clique assigned to one part is maximal in the particular induced graph used for counting, and why the reverse implication is unnecessary.

(b) 设 d 是非负的三的倍数且 $n \ge d + 3$，证明退化度为 d 的图至多有 $(n - d)3^{d/3}$ 个极大团；使用退化度排列，把前 $n - d - 3$ 个顶点按最早顶点分族，把最后 $d + 3$ 个顶点作为整体处理。必须解释分到某部分的全图极大团为什么会在计数所用的指定诱导图中极大，以及为什么不需要反方向成立。

(c) Prove that $K_{n-d,3,\dots,3}$, with exactly $d/3$ parts of size three, attains the bound. Prove maximality of every counted clique, prove that there are no others, and establish both bounds needed for degeneracy to equal d. Cover $d = 0$ and $n = d + 3$ explicitly. The mathematical result is the one recorded in [ELS2010-v1, p. 8]; the task is to supply its full reasoning.

(c) 证明大小为三的部恰有 $d/3$ 个时，题给完全多部图达到上界；证明数到的每个团都极大、没有其他极大团，并同时证明退化度等于 d 所需的上下界，明确覆盖 $d = 0$ 和 $n = d + 3$。这一数学结果见 [ELS2010-v1，第 8 页]，本题要求补齐完整推理。

Answers to Exercises / 习题答案

References to algorithms, definitions, lemmas, and theorems refer to the companion text of this section. All graph inputs are finite, simple, and undirected. Set strings in traces use the notation defined in the worked examples. A finite computation can check a trace or reveal a counterexample; the general claims below require the proofs that accompany them.

本稿中的算法、定义、引理与定理编号均指本节正文，所有输入图都有限、简单且无向，追踪中的字母集合沿用正文记法；有限计算可以核对轨迹或发现反例，下面的一般性结论则由随附证明承担。

1. The classifications are as follows. An affirmative maximality claim is checked against every vertex outside the set, not only against a particular unused vertex.

分类如下；肯定极大性时必须检查集合外的每个顶点，而不是只试某个尚未使用的点。

Table 7. Classifying five proposed sets / 表 7. 五个点集的分类

Set / 集合	Clique / 团	Maximal / 极大	Maximum / 最大	Evidence / 证据
$\varnothing$	yes / 是	no / 否	no / 否	add a / 可添 a
$\{a, b\}$	yes / 是	no / 否	no / 否	add c / 可添 c
$\{a, b, c\}$	yes / 是	yes / 是	yes / 是	size 3 / 大小为 3
$\{d, e\}$	yes / 是	yes / 是	no / 否	size $2 < 3$ / 小于三角形
$\{a, d\}$	no / 否	no / 否	no / 否	edge ad absent / 缺边 ad

For the triangle, both outside vertices have no edge to any triangle vertex; for the edge, no outside vertex is adjacent to both endpoints. Thus both sets are maximal. A clique cannot use vertices from both components, because all cross-component pairs are nonedges; within the components the largest sizes are three and two. This proves that the triangle is maximum, rather than merely exhibiting a clique of size three.

对三角形而言，两个外部顶点都不邻接其中任何顶点；对独立边而言，外部各点都不能同时邻接它的两个端点，因此两者都极大。由于不同连通部分之间的点对全部缺边，一个团不能跨越两个部分，而各部分内部的最大大小分别为三和二，所以三角形确实最大，不只是找到了一个三点团。

2. On the empty vertex graph, A and B each emit one record containing the empty set; C returns that set and size zero. On an edgeless graph with $n > 0$, the maximal-clique outputs are exactly the n singleton sets. C returns the first vertex in its fixed order and size one: its first inclusion creates that singleton, and no later branch can improve the incumbent.

在空顶点图上，A 与 B 各输出一条包含空集的记录，C 返回空集及大小零；在 $n > 0$ 的无边图上，极大团输出恰好是 n 个单点集。C 返回固定次序中的第一个顶点及大小一，因为首次选入就产生该单点，此后没有分支能够改善当前最好解。

Add a new isolated vertex z. If the old graph is nonempty, its old maximal cliques remain maximal: they are nonempty, and z cannot extend any of them. The new singleton is maximal, and no clique containing it can contain any other vertex, so these are all the new outputs. If the old graph is empty, its former maximal empty set is no longer maximal and is replaced by the singleton. The new clique number is the larger of the old clique number and one, which covers both cases.

加入新孤立点 z 时，若原图非空，原有极大团也都非空，z 无法扩充其中任何一个，所以它们仍极大；新单点本身极大，且任何含它的团都不能再含别的顶点，因此全部新答

案就是原答案再加这个单点。若原图为空，原先极大的空集已经不再极大，需要被新单点替代；新团数等于原团数与一中的较大者，这也同时覆盖两种情况。

3. Intersect the listed neighborhoods, excluding members of the proposed current set as in D2. The result is below.

按 D2 对相应邻域取交，并排除当前集合本身的成员，结果如下。

Table 8. Common neighborhoods in Exercise 3 / 表 8. 习题 3 的公共邻域

R	$\Gamma(R)$	Legal as a current clique? / 可作当前团?
$\varnothing$	$abcdefg$	yes / 可以
b	acd	yes / 可以
bc	ad	yes / 可以
ad	bc	no: $ad \notin E$ / 不可：缺边
g	$\varnothing$	yes / 可以

The set ad has two common neighbors, but its own pair is not adjacent. A common-neighborhood calculation asks whether an outside vertex meets all required adjacencies; it does not check the adjacencies inside R. Therefore the condition that R is a clique in D4 is independent of the union condition for P, X. A legal current clique alone also does not make an arbitrary triple legal: the other clauses must still be checked.

集合 ad 虽有两个公共邻居，其内部的一对顶点却不邻接；公共邻域计算检查的是外部点是否满足所需邻接关系，并不检查 R 内部的边。因此，D4 中 "R 是团" 的条件独立于 P, X 的并集条件，而且即使当前集合确实为团，任意三集合状态也不一定合法，其余条件仍须核对。

4. The complete entry and return trace is as follows. Return-time moves occur in the order of the second table, including those made inside a child before its parent resumes.

完整的入口与返回追踪如下，第二张表按实际时间顺序记录移动，也包括父调用恢复之前在子调用内部完成的移动。

Table 9. Path entry states for Exercise 4 / 表 9. 习题 4 的路径入口状态

Call / 调用	Depth / 深度	R	P	X	u	S	Children / 子调用	Result / 结果
1	0	$\varnothing$	abc	$\varnothing$	—	abc	a→2, b→4, c→6	return / 返回
2	1	a	b	$\varnothing$	—	b	b→3	return / 返回

Call / 调用	Depth / 深度	R	P	X	u	S	Children / 子调用	Result / 结果
3	2	ab	∅	∅	—	—	—	output / 输出
4	1	b	c	a	—	c	c→5	return / 返回
5	2	bc	∅	∅	—	—	—	output / 输出
6	1	c	∅	b	—	—	—	excluded / 被排除

Table 10. Path return-time moves for Exercise 4 / 表 10. 习题 4 的路径返回移动

Parent / 父调用	Returned vertex / 返回点	New P / 新候选集	New X / 新排除集
2	b	∅	b
1	a	bc	a
4	c	∅	ac
1	b	c	ab
1	c	∅	abc

At the final call, $R = c$, P is empty, and $X = b$. The singleton has no unused extension in this branch, but b still extends it in the graph. Accordingly, the two outputs are ab and bc, and no singleton is output.

最后一次调用中，当前集合为 c，候选集为空而排除集为 b；这个单点在本分支虽已没有尚未使用的扩充点，但在图中仍可加入 b，所以两项输出是 ab 和 bc，没有单点被误报。

5. A single edge ab is a smallest counterexample. The root first chooses a, whose candidate set contains b, and outputs ab after the second inclusion. Back at the root, a has been removed. Choosing b now gives an empty candidate set and outputs b, although a extends it. Graphs with zero or one vertex do not fail under the stated convention, proving minimality in vertex count.

单独一条边 ab 就是顶点数最少的反例：根先选 a，其候选集中含 b，再次选入后输出 ab；回根后 a 已被删去，再选 b 时候选集为空，于是输出 b，但在原图中它仍能加入 a。零点图和单点图在既定约定下都不会出错，因此这个反例在顶点数上最小。

This variant need not duplicate a set, and in fact never does if the parent removals and inherited candidate intersections are retained. At the first divergence, the earlier selected vertex remains in one descendant current set and is unavailable to the other, just as in the call-count proof. The failure is loss of global maximality information. If every child instead resets its candidates to all current common neighbors, previously processed vertices can reappear: on the same edge, the

paths choosing a then b and choosing b then a both output ab. The correct output family has just one member, that edge. This restored-neighbor variant has a different defect from the first one.

只要保留父状态删除以及继承候选集的求交，这个变体实际上不会重复输出同一集合，因为第一次分叉时，先选的顶点仍留在一边的后代当前集合中，另一边已经不能再选它，和调用计数证明相同；它的问题是丢失了全图极大性的信息。若进一步把每个子调用的候选恢复为全部当前公共邻居，兄弟选择中已经删过的点便能重新出现，于是在同一条边上，先选 a 再选 b 与先选 b 再选 a 都输出 ab，而正确输出族只应包含这一条边一次；第二个变体的错误因此与第一个不同。

6. Take $R = \{a\}$ and both other sets empty on the edge ab. The sets are pairwise disjoint and R is a clique, but its common neighborhood is $\{b\}$, so the covering equation of D4 fails. A2 falsely emits $\{a\}$. The family defined by the interval $R \subseteq K \subseteq R \cup P$ contains no globally maximal clique; it should emit nothing from this restricted state.

在边 ab 上取 $R = \{a\}$，另外两集都空，三者两两不交且 R 是团，但其公共邻域为 $\{b\}$，所以 D4 的覆盖等式不成立。A2 会错误输出 $\{a\}$，而由当前区间定义的责任族中并没有全图极大团，这个受限状态原本不该输出任何东西。

For a malformed root, take $R = X = \varnothing$ and $P = \{a\}$. Its union misses b, whereas the root common neighborhood is all of V. The procedure again outputs $\{a\}$ and misses the actual maximal clique $\{a, b\}$. The formula for the restricted responsibility still gives an empty family, while the required full-root responsibility is the one-member family containing the edge. This example shows both the false local certificate and the lost global coverage; disjointness alone establishes neither.

作为非法根状态，取 $R = X = \varnothing$、$P = \{a\}$，其并集漏掉了 b，而根的公共邻域本应等于整个 V；程序同样输出 $\{a\}$，并漏掉真正的极大团 $\{a, b\}$。对这个受限区间套用责任族公式仍得到空族，真正完整根状态应该负责的却是只含该边的一成员族，因此反例同时暴露了错误的局部判断与丢失的全局覆盖，两两不交不能保证其中任何一项。

7. The root pivot is a and its snapshot is just a. The child with current set a has candidates bc, pivot b, and snapshot b; the next child has current set ab, candidate c, pivot c, and snapshot c. The final call has current set abc and two empty auxiliary sets, so it outputs once. Each pivot is absent from its own open neighborhood and is therefore retained as a branch candidate while it belongs to P.

根枢轴为 a，快照只有 a；当前集合为 a 的子调用具有候选 bc，选择枢轴 b、快照 b，下一子调用的当前集合为 ab、候选为 c，便选择枢轴 c、快照 c。最终调用的当前集合是 abc，两个辅助集合皆空，所以输出一次；每个枢轴都不在自己的开邻域中，只要它位于 P，就会作为分支候选保留。

In the second state, the common neighborhood of c is ab, so the stated union and disjointness conditions hold. Choosing pivot a gives an empty snapshot, because b is adjacent to a. The only maximal clique in the graph is abc, which contains an excluded vertex and is outside this state's

responsibility. Returning without output is correct even though P is nonempty. A pivot in X need not start a child itself.

第二个状态中，c 的公共邻域为 ab，题给集合确实满足并集和不交条件；选枢轴 a 后，由于 b 邻接它，快照为空。全图唯一极大团是 abc，它含有被排除顶点，已经不在该状态的责任范围内，所以即使 P 非空，不输出而返回仍然正确；位于 X 中的枢轴也无须亲自启动子分支。

8. (a) On a one-vertex graph, the root has $P = \{a\}$ and pivot a. Subtracting the closed neighborhood removes a, leaving an empty snapshot and no output. The correct maximal-clique family is $\{\{a\}\}$. With the open neighborhood, the snapshot contains a and its child emits the required singleton.

(a) 对单点图，根候选集为 $\{a\}$，枢轴为 a；若减去闭邻域，就会把自身删掉，留下空快照而完全不输出，但正确极大团族应当只含单点 $\{a\}$。使用开邻域时，快照保留 a，其子调用就能输出所需单点。

(b) On the empty vertex graph, B1 emits the empty clique. If it then proceeds to B2, it must choose a pivot from an empty union, an undefined operation. The record emitted before the fault is mathematically correct, but the execution has not satisfied the algorithm's termination contract. The same fault occurs after every ordinary output leaf. An implementation may throw an exception instead of duplicating output; that is still a failed algorithm.

(b) 在空顶点图上，B1 会输出空团；若随后继续进入 B2，就必须从空的并集中选枢轴，这个操作没有定义。故障之前发出的记录虽然数学上正确，执行过程却没有履行算法的正常终止合同，而且每个普通输出叶也会遇到同样问题；实现可能抛出异常而不是重复输出，但仍然是算法故障。

(c) Use the graph with edge ab and isolated vertex c. Choose the legal root pivot c; the snapshot is abc. Let the erroneous routine retain the root entry copies with all three candidates and no exclusions for every child. The branch at a receives candidate b and outputs ab. After returning, the live root moves a to X, but the branch at b is built from the stale copies, receives candidate a and no excluded vertex, and outputs ab again. The final branch outputs c. The erroneous stream is ab, ab, c; the correct family is ab, c.

(c) 使用边 ab 与孤立点 c 构成的图，选择合法根枢轴 c，快照便为 abc；让错误过程对每个子调用都保留根入口处 "三点全可选、无排除" 的旧副本。a 分支收到候选 b 并输出 ab，返回后实时根虽然把 a 移入 X，但 b 分支仍由旧副本建立，收到候选 a 且没有排除点，于是再次输出 ab，最后的分支再输出 c。错误输出流为 ab, ab, c，正确输出族则是 ab, c。

This counterexample deliberately uses a permitted nonmaximizing pivot, so it tests T2's general contract, not the specific tie rule of the worked example. In the correct live-state child at b, P would be empty and X would contain a, and no second copy of the edge would be emitted.

这个反例有意使用 T2 允许的非最大化枢轴，检验的是一般正确性合同，而不是正文实例的特定并列规则；使用正确实时状态时，b 子调用的候选集为空、排除集含 a，所以不会再次输出该边。

9. (a) Reversing pivot ties selects d at the root, with snapshot $adfg$. The depth-first entry sequence and chosen pivots are below; a dash means a terminal return before pivot selection.

(a) 反转枢轴并列规则后，根选 d，快照为 $adfg$；按深度优先列出的入口序列与枢轴如下，短横线表示在挑选枢轴之前已经终止返回。

Table 11. Reversed pivot ties in Exercise 9 / 表 11. 习题 9 的反向枢轴并列规则

Call / 调用	Current set / 当前集合	Pivot / 枢轴	Output? / 是否输出
1	∅	d	no / 否
2	a	c	no / 否
3	ac	b	no / 否
4	abc	—	yes / 是
5	d	c	no / 否
6	cd	b	no / 否
7	bcd	—	yes / 是
8	de	—	yes / 是
9	f	e	no / 否
10	ef	—	yes / 是
11	g	—	yes / 是

The original tie rule uses 10 calls; this one uses 11 and produces the same five maximal cliques. Both root pivots have three candidate neighbors and hence four immediate branches. Equal immediate counts can lead to different descendant trees. The computation compares two rules on one labeled input only; it neither quantifies over other graphs nor proves that 10 is the minimum possible count among all pivot choices even here.

原并列规则使用 10 次调用，这个规则使用 11 次，但输出仍是同样的五个极大团；两个根枢轴都具有三个候选邻居，因而都有四个直接分支，直接分支数相同并不意味着后代树也相同。这个计算只比较了一个带标号输入上的两种规则，既没有覆盖其他图，也没有证明即使在此图上，10 已经是所有枢轴选择中的最小调用数。

(b) In K_n, every subset is a clique. In A, the vertices selected along a path occur in increasing order: before a later sibling is visited, all earlier snapshot vertices have left the parent's P, and intersections can only remove candidates. Conversely, every nonempty subset is reached by choosing its vertices in increasing order, because all its remaining vertices are adjacent to the next choice and have not yet been removed. The empty subset labels the root. These two facts put the calls in bijection with all 2^n subsets. Only V is maximal, since every proper subset can be extended by any missing vertex, so only the call with $R = V$ outputs.

(b) 在 K_n 中，每个子集都是团；A 沿一条路径选入的顶点按递增次序出现，因为轮到较晚兄弟分支之前，较早的快照顶点都已离开父状态的 P，而后续求交只能继续删除候选。

反过来，每个非空子集都能按其顶点递增次序到达，因为它尚未选择的顶点既邻接下一个选点，也尚未被移走；空子集则对应根调用。这两点使调用与全部 2^n 个子集构成双射。只有 V 极大，因为每个真子集都能加入任意一个缺少的顶点，所以只有 $R = V$ 的调用输出。

B starts with $X = \varnothing$. At a call on its first descent, suppose $X = \varnothing$ and $P = V \setminus R$ has $k > 0$ members. Every pivot candidate lies in P, has score $k - 1$, and leaves the singleton snapshot $\{u\}$ because its open neighborhood omits only itself. Its sole child again has empty X and all remaining vertices in P. Induction gives one call for each current-set size from zero through n, hence $n + 1$ calls and a single output V. Moving the selected vertex after that child returns creates no further child, since the snapshot was a singleton. Thus both output lengths are $L = n$.

B 从 $X = \varnothing$ 出发；假定首次向下搜索到某个调用时，$X = \varnothing$ 且 $P = V \setminus R$ 含 $k > 0$ 个点，那么每个枢轴候选都在 P 中，评分均为 $k - 1$，其开邻域只漏掉自身，所以快照只有 $\{u\}$。唯一子调用仍具有空的 X，且 P 含全部剩余顶点，由归纳可知，当前团大小从零到 n 各有一次调用，总数为 $n + 1$，唯一输出为 V。子调用返回后虽然还要移动所选点，但单点快照已经耗尽，不会产生新孩子，因此两种算法的输出长度都为 $L = n$。

Now charge the specified implementation for $n \geq 1$. Matrix validation takes $O(n^2)$ time. For A, each call needs at most $O(n)$ work for terminal tests and its snapshot, and each of its $2^n - 1$ tree edges incurs $O(n)$ work for copies, intersections, and the return-time move. The loop work is charged to those edges, rather than multiplied into the per-call bound a second time. For B, at the unique call with k remaining candidates, building member lists and the snapshot costs $O(n)$, and the full pivot scoring performs k^2 adjacency tests, including the diagonal tests. Its n edges each incur $O(n)$ set work. This yields the following upper bounds; they concern this implementation, not an optimal algorithm for recognizing complete graphs.

现在对 $n \geq 1$ 的指定实现计费，矩阵核验需要 $O(n^2)$ 时间。A 的每次调用至多花 $O(n)$ 时间进行终止检查和准备快照，而 $2^n - 1$ 条递归树边各承担 $O(n)$ 的复制、求交与返回移动成本；循环工作记入这些树边，不能再把子调用数重复乘进每次调用的成本。对 B，剩余 k 个候选的唯一调用需要 $O(n)$ 时间生成成员列表和快照，完整枢轴评分则进行 k^2 次邻接检查，其中包含对角项；它的 n 条树边又各需要 $O(n)$ 的集合处理。因此得到下面这些上界，它们描述指定实现，并不声称是识别完全图的最优算法。

$$\text{time}_A = O(n^2 + n2^n + L) = O(n2^n + L),$$
$$\text{time}_B = O\left(n^2 + \sum_{k=1}^{n}(n + k^2) + L\right) = O(n^3 + L).$$

For the last simplification, the sum has n terms, each with $k^2 \leq n^2$; for A, $n \leq 2^n$ absorbs the matrix term. Both depth-first runs have at most $n+1$ active calls, each holding a constant number of length-n arrays and linear-size lists. Their active auxiliary storage is therefore $O(n^2)$, and including the adjacency matrix leaves the same bound. The output is streamed. One output does not imply linear time: A already makes exponentially many calls, while B's root pivot scoring alone performs n^2 adjacency tests, in addition to matrix validation. On K_0, both algorithms

make one call, emit one empty record with $L = 0$, and use constant time and space under the separate empty-input convention.

最后一个化简使用了求和共有 n 项、每项的 $k^2 \le n^2$；A 则由 $n \le 2^n$ 吸收矩阵成本。两种深度优先运行都至多同时保留 $n+1$ 个调用，每层持有常数个长度为 n 的数组和线性大小的列表，所以活动辅助存储为 $O(n^2)$，把邻接矩阵也计入后仍为同一上界，输出则采用流式处理。只有一个输出并不能推出线性时间：A 的调用数已经是指数级，B 除了核验矩阵，仅根调用的完整枢轴评分就要进行 n^2 次邻接检查。对于 K_0，两种算法都只调用一次，输出一条空记录且 $L = 0$，按独立的空输入约定使用常数时间和空间。■

(c) For a fixed input size n, let $Q(k)$ be the largest possible number of calls in a C-subtree whose root has at most k candidates. With no candidates there is just one call. A pruned call also contributes only one; otherwise there are at most two children, each with at most $k-1$ candidates. Induction therefore gives the recurrence and bound below: if the preceding bound is $2^k - 1$, the new bound is at most $1 + 2(2^k - 1) = 2^{k+1} - 1$.

(c) 固定输入规模 n，令 $Q(k)$ 为 C 的子树调用数可能达到的最大值，其中子树根至多有 k 个候选；没有候选时只有一次调用，被剪去的调用也只贡献一次，否则至多有两个孩子，且各至多剩下 $k-1$ 个候选。因此可以用下面的递推归纳：若前一项上界为 $2^k - 1$，新一项就至多为 $1 + 2(2^k - 1) = 2^{k+1} - 1$。

$$Q(0) = 1, \qquad Q(k) \le 1 + 2Q(k-1) \le 2^{k+1} - 1 \quad (k \ge 1).$$

For $n \ge 1$, first-fit coloring costs $O(n^2)$ per call; set construction, copying the incumbent, and the work on at most two outgoing edges fit within that charge. Matrix validation and the final output of at most n identifiers fit within $O(n^2 2^n)$ total time. At most $n+1$ active levels each require $O(n)$ storage, so auxiliary space, including the matrix and incumbent, is $O(n^2)$. At $n = 0$, the single root returns the empty clique in constant time and space. C's exclusion child keeps its parent's R, so distinct calls can have the same current set even on a one-vertex graph. The binary recurrence is valid; an injection of C's calls into subsets via R is not.

对 $n \ge 1$，首次适配着色每次调用至多花 $O(n^2)$ 时间，集合构造、最好团复制以及至多两条出边的工作也能计入这一成本，矩阵核验和最后至多 n 个标识的输出则都能计入总时间 $O(n^2 2^n)$。至多 $n+1$ 个活动层各需 $O(n)$ 存储，因此包含矩阵与当前最好团的辅助空间为 $O(n^2)$；$n = 0$ 时，唯一根调用以常数时间和空间返回空团。C 的排除子调用保留父状态的 R，所以即使在单点图上，不同调用也能具有相同的当前集合；二叉递推依然成立，但把 C 的调用按 R 单射到子集的论证不成立。■

10. Induct over calls. The empty initial incumbent is a clique. An inclusion child uses a vertex adjacent to every current member, an exclusion child keeps the current set, and the only incumbent update copies such a current clique. Therefore the stored size is always the size of a feasible clique. Every represented extension adds at most $|P|$ vertices, so if the resulting cardinality upper bound does not exceed the incumbent size, it cannot improve the answer. The monotonicity of incumbent sizes makes the conclusion remain true after later updates.

对调用归纳，初始最好解为空团，选入子调用使用邻接全部当前成员的顶点，排除子调用保持当前集合，唯一的最好解更新又只是复制这样一个当前团，因此保存的大小始终来自可行团。任意被表示的扩充至多再添 $|P|$ 个点，所以顶点数上界若不超过最好解大小，就不可能改善答案，而最好解大小的单调性保证后续更新之后这一结论仍成立。

On a single edge, give the empty incumbent a fictitious recorded size three. The root upper bound is two, both by cardinality and by a valid two-coloring, and the procedure prunes immediately. It returns neither the real optimum edge nor a valid clique of recorded size three. Even assigning a fictitious size two would allow equality pruning before any size-two clique was constructed. Upper bounds restrict what is possible; they cannot certify that a lower bound has been achieved.

在一条边上，若给空的最好解虚记大小三，根的顶点数上界和合法两色上界都等于二，程序便立即剪枝，既没有返回真实最优边，也没有返回任何大小为三的可行团；即使虚记为二，也会在尚未构造二点团之前触发相等剪枝。上界只能限制可能性，不能证明某个下界已经实现。

11. Consider two disjoint edges ab and cd, with alphabetical order. After the first branch finds ab, the incumbent size is two. In the later root exclusion branch, the remaining graph on b, c, d has a proper coloring with two classes, so C3's original test can prune it at equality and lose cd. Both edges are maximum; the one-solution contract permits this, but the all-solutions contract does not.

考察互不相交的两条边 ab 与 cd，按字母顺序处理；第一分支找到 ab 后，最好解大小为二，而后来的根排除分支在剩余 b, c, d 上具有两类合法着色，C3 原有条件便可能在相等时剪掉它，丢失 cd。两条边都是最大团，只求一个解的合同允许这种处理，全部解合同则不允许。

A correct modification collects solutions only when P is empty, and prunes only when the upper bound is strictly smaller than the best size. Initialize the best size to zero and the solution list to empty. At a terminal call, if $|R|$ is larger than the best size, set the best size to it and replace the entire list with a one-element list containing the clique R; if it is equal, append R; if smaller, do nothing. For simplicity, do not update or collect at internal calls. This may postpone a useful lower bound but preserves correctness.

一种正确修改只在 P 为空时收集解，而且仅当上界严格小于当前最好大小时剪枝；把最好大小初始化为零、解列表初始化为空，到终止调用时，如果 $|R|$ 更大，就更新最好大小并将解列表替换为仅含团 R 的单元素列表，如果相等就追加，如果更小则不处理。为使规则明确，内部调用既不更新也不收集；这样可能推迟有效下界的获得，却不会影响正确性。

To prove absence of duplicates at terminal calls, take two different root-to-leaf paths and their first binary split at vertex v. One path includes it permanently and the other excludes it permanently, so their terminal sets differ. Repeated internal appearances of the same R along an exclusion chain are never collected. Every globally maximum clique has a unique path through these splits; no containing state can be pruned while the best size is at most the true optimum, because its upper bound is at least that optimum. If a smaller clique was collected earlier, the first larger

terminal clique clears it. Thus the final list is precisely all maxima. In the empty vertex graph the root is terminal and appends the empty set once.

为证明终止调用不重复，取两条不同的根到叶路径，在它们首次按顶点 v 二分的地方，一条永久选入 v，另一条永久排除 v，因此终止集合不同；同一个 R 沿排除链在内部重复出现时不收集。每个全局最大团都按这些二分获得唯一路径，而且只要当前最好大小不超过真实最优值，包含这个最大团的状态就不会被剪去，因为它的上界至少等于最优值；此前若曾收集较小团，遇到第一个更大的终止团就会清空旧解，因此最终列表恰为全部最大团。空顶点图的根直接终止，并恰好追加一次空集。■

12. In call 7 the graph on b, c, d, e, f has edges from b to each of the other four vertices and no edges among those four. Thus the classes b and $cdef$ are independent, cover all candidates, and give two as the upper bound. The candidate-count bound is five. The stored triangle has size three, which lies strictly above the valid color bound and below the unhelpful count bound.

调用 7 中，b, c, d, e, f 上的边恰好从 b 连向另外四点，那四点之间没有边，所以颜色类 b 和 $cdef$ 都独立、覆盖全部候选，并给出上界二；候选数上界则为五。已经保存的三角形大小为三，严格超过有效颜色界，却小于不能帮助剪枝的候选数界。

On the five-cycle, first-fit coloring in order gives classes ac, bd, and e. The first four assignments alternate between the first two classes; e is adjacent to a in the first and d in the second, so it needs the third. These are independent classes, proving a bound of three. There is an edge, so the clique number is at least two. For every vertex its two neighbors are not adjacent to one another, so no triangle exists; any clique of size at least three would contain a triangle. The clique number is therefore exactly two, and this color bound is not attained.

在五边形上，首次适配的字母顺序着色得到 ac、bd、e 三类：前四次依次交替放入前两类，而 e 同时邻接第一类的 a 和第二类的 d，只好使用第三类。这些类都独立，所以三确实是上界；图中有边，团数至少为二，而每个顶点的两个邻居互不邻接，图中没有三角形，任何至少三点的团却必须包含三角形，因此团数恰为二，颜色上界没有被达到。

13. For any subset S, each distinct pair in S is an edge of G exactly when it is a nonedge of $\overline{G}$. Hence S is a clique in G exactly when it is independent in the complement. This identity preserves the actual subsets and their inclusion relation. A strict clique superset exists in one graph exactly when a strict independent superset exists in the other, proving the maximal correspondence. The families have identical cardinalities member by member, proving the maximum correspondence as well. On an empty vertex set, both families contain just the empty set, which is both maximal and maximum of size zero.

对任意点集 S，其中每个不同点对在 G 中是边，当且仅当它在 $\overline{G}$ 中不是边，所以 S 在原图中为团，当且仅当它在补图中为独立集。这个对应保留实际子集及其包含关系，一边存在严格的团超集，恰好等价于另一边存在严格的独立超集，极大性因此对应；两族中逐项成员的大小也完全相同，所以最大性同样对应。空顶点图两边都只有空集，它既极大又最大，大小为零。■

14. The minimum-degree deletion sequence and degrees are below. Degrees are recomputed in the graph that remains, not read from the original adjacency table.

最小度删除序列及删除时的度数如下；每次度数都在剩余图中重新计算，而不是沿用原邻接表的度数。

Table 12. Minimum-degree deletion in Exercise 14 / 表 12. 习题 14 的最小度删除

Step / 步骤	Removed / 删除顶点	Degree on removal / 删除时度数	Remaining / 余下顶点
1	g	0	abcdef
2	f	1	abcde
3	e	1	abcd
4	a	2	bcd
5	b	2	cd
6	c	1	d
7	d	0	$\varnothing$

The largest deletion degree is two, so the order gives at most two later neighbors per vertex and implies degeneracy at most two by L4. The original graph contains the induced triangle abc, in which every degree is two, so no bound smaller than two can satisfy the definition. Its degeneracy is exactly two.

最大删除度为二，因此所得排列中每点至多有两个后继邻居，由 L4 得退化度至多为二；原图又含有诱导三角形 abc，其中每点度数都是二，所以任何小于二的界都不能满足定义，准确退化度便为二。

More generally, if a degree bound holds for every nonempty remaining induced graph, successively remove an admissible vertex. Its degree on removal counts precisely its later neighbors, giving an order. Conversely, in any nonempty subgraph, choose the earliest vertex from such an order. Every neighbor retained in that subgraph is later, giving the required degree bound even if some original edges were deleted. For a star with $t \geq 1$ leaves, delete all leaves before the center. The resulting later-neighbor bound is one, and the existence of an edge gives degeneracy at least one. The maximum degree is t, which is unbounded as the number of leaves grows.

更一般地，如果每个非空剩余诱导图都满足某个度数界，就可以依次删除满足条件的顶点，每次删除时的度数恰好计数其后继邻居，从而得到排列；反过来，在任意非空子图中选择这种排列最早出现的顶点，其保留的邻居都在后方，即使删去了一部分原边，也仍满足同一个度数界。对含 $t \geq 1$ 个叶子的星形图，先删全部叶子再删中心，后继邻居界为一，而存在一条边又迫使退化度至少为一；其最大度却等于 t，随着叶子数增加而无界增长。■

15. (a) Begin with the numerical inequality. For $k = 1, 2, 3, 4$, cube both sides; the required comparisons are respectively $1 \leq 3$, $8 \leq 9$, $27 = 27$, and $64 \leq 81$. If the inequality holds for an

integer $k \geq 2$, then $k+3 \leq 3k$ and hence the following implication holds. Starting from two, three, and four proves every integer at least two by steps of three; one was checked separately. This argument uses no differentiation or optimization over real numbers.

(a) 先证数值不等式。对 $k = 1, 2, 3, 4$，将两边立方，所需比较依次为 $1 \leq 3$、$8 \leq 9$、$27 = 27$ 和 $64 \leq 81$；若不等式对某个 $k \geq 2$ 成立，因为 $k+3 \leq 3k$，就得到下列推导。从二、三、四分别以步长三归纳，覆盖所有至少为二的整数，一则已单独检查，整个论证不需要求导或在实数域优化。

$$k \leq 3^{k/3},\ k \geq 2 \quad \Longrightarrow \quad k+3 \leq 3k \leq 3\,3^{k/3} = 3^{(k+3)/3}.$$

We now prove the independent-set count by induction on the number n of vertices of an arbitrary graph H. When $n = 0$, the empty set is the sole maximal independent set, and the bound is one. For $n > 0$, let v have minimum degree δ in H. Every maximal independent set intersects $N_H[v]$: if it did not, it would contain neither v nor any neighbor of v, and adding v would preserve independence, contradicting maximality.

现在对任意图 H 的顶点数 n 归纳，证明极大独立集计数界。$n = 0$ 时，空集是唯一极大独立集，上界也为一；$n > 0$ 时，取 H 中最小度为 δ 的顶点 v，则每个极大独立集都与 $N_H[v]$ 相交，否则它既不含 v 又不含 v 的任何邻居，可以直接加入 v 而保持独立，与极大性矛盾。

For each $w \in N_H[v]$, let $\mathcal{A}_w$ be the family of maximal independent sets containing w. These families cover all maximal independent sets, but may overlap. Set $H_w = H[V(H) \setminus N_H[w]]$. If $I \in \mathcal{A}_w$, independence implies that $I \setminus \{w\}$ lies in this induced graph. It is maximal there: a vertex of H_w that could be added to it is not adjacent to w, by construction of H_w, and would therefore also extend I in H. The map deleting w is injective on $\mathcal{A}_w$, because adding it back recovers I. We consequently have the following inequalities.

对每个 $w \in N_H[v]$，令 $\mathcal{A}_w$ 为包含 w 的极大独立集族，这些族覆盖全部极大独立集，但可能彼此重叠；再令 $H_w = H[V(H) \setminus N_H[w]]$。若 $I \in \mathcal{A}_w$，独立性保证删去 w 后的集合落在这个诱导图中，而且它在其中极大：如果 H_w 中还有某点能加入剩余集合，因为该点按构造不邻接 w，它也能加入原来的 I，与极大性矛盾。删去 w 的映射在 $\mathcal{A}_w$ 上是单射，因为添回 w 就能恢复原集，因此得到下面的不等式。

$$\begin{aligned} f(H) &\leq \sum_{w \in N_H[v]} |\mathcal{A}_w| \\ &\leq \sum_{w \in N_H[v]} f(H_w) \\ &\leq \sum_{w \in N_H[v]} 3^{(n - \deg_H(w) - 1)/3} \\ &\leq (\delta+1) 3^{(n-\delta-1)/3} \\ &\leq 3^{(\delta+1)/3} 3^{(n-\delta-1)/3} = 3^{n/3}. \end{aligned}$$

Here every H_w has fewer than n vertices, so the third line follows from the induction hypothesis, including an empty H_w. Minimum degree gives $\deg_H(w) \geq \delta$, and the closed neighborhood of v has exactly $\delta + 1$ vertices, giving the fourth line. The last line applies the numerical inequality with positive integer $\delta + 1$. No disjointness of the covering families was assumed. Exercise 13 now transfers this count to maximal cliques of an arbitrary n-vertex graph by applying it to the complement. The empty-vertex case transfers as well.

这里每个 H_w 的顶点数都严格小于 n，所以第三行适用归纳假设，也包括 H_w 为空的情况；最小度条件给出 $\deg_H(w) \geq \delta$，而 v 的闭邻域恰含 $\delta + 1$ 个顶点，因此得到第四行，最后一行则把数值不等式应用于正整数 $\delta + 1$。论证没有假设覆盖族互不相交，最后再对任意 n 点图的补图使用习题 13，就将计数界转为原图的极大团计数界，空顶点情形也随对应一并成立。■

(b) Let $v_1, \ldots, v_n$ be a degeneracy ordering. Partition the globally maximal cliques into the classes whose earliest vertex is one of the first $n - d - 3$ vertices and the class of cliques wholly contained in the final set T of $d + 3$ vertices. The conditions in the question ensure that the prefix length is nonnegative and that T has exactly that size. Every maximal clique is nonempty because $n \geq 3$, so every clique belongs to exactly one of these classes.

(b) 取退化度排列 $v_1, \ldots, v_n$，把全图极大团分为若干类：最早顶点分别是前 $n - d - 3$ 个顶点之一的各类，以及完全包含在末尾 $d + 3$ 个顶点集合 T 中的一类。题设保证前缀长度非负、尾集恰有指定大小，而 $n \geq 3$ 又保证极大团非空，所以每个极大团恰好属于其中一类。

Fix a prefix index i, and let W_i be the neighbors of v_i occurring later in the ordering. There are at most d of them. If K is a globally maximal clique whose earliest vertex is v_i, then $K \setminus \{v_i\}$ is a clique in $G[W_i]$. It is maximal in that induced graph: if $w \in W_i$ could extend it, w would also be adjacent to v_i, and would extend the original K in the whole graph. Removing v_i is injective for this fixed class, including the case of an empty remainder. Part (a) therefore bounds the number in the class by $3^{|W_i|/3} \leq 3^{d/3}$.

固定一个前缀下标 i，令 W_i 为 v_i 的后继邻居集，它的大小至多为 d；若全图极大团 K 的最早顶点为 v_i，则删去这个顶点后得到 $G[W_i]$ 中的团，而且它在该诱导图中极大，因为若 W_i 中还有某点能扩充剩余集合，这个点也邻接 v_i，从而能在全图中扩充 K。对固定这一类，删去 v_i 是单射，即使余下集合为空也成立，因此由 (a) 知道该类的团个数至多为 $3^{|W_i|/3} \leq 3^{d/3}$。

A globally maximal clique contained in T is also maximal in $G[T]$, because an extension inside T would be an extension in the whole graph. Part (a) bounds the entire tail class by $3^{(d+3)/3}$. It is unnecessary for every maximal clique in either induced graph to lift to a globally maximal clique. We need only an injection from the globally maximal cliques being counted into the larger family of induced-graph maximal cliques. Adding these bounds gives the requested expression.

完全包含在 T 中的全图极大团也一定在 $G[T]$ 中极大，否则尾集内的扩充就是全图中的扩充，所以 (a) 将整个尾类限制在 $3^{(d+3)/3}$ 以内。并不需要任何一个诱导图中的所有极大

团都能还原为全图极大团，我们只需要把正在计数的全图极大团单射到可能更大的诱导图极大团族中；将这些界相加，就得到所要求的表达式。

$$\begin{aligned}\#\{\text{maximal cliques of } G\} &\le (n-d-3)3^{d/3} + 3^{(d+3)/3} \\ &= (n-d-3)3^{d/3} + 3\,3^{d/3} \\ &= (n-d)3^{d/3}.\end{aligned}$$

For $n = d+3$, the prefix is empty and the tail estimate alone yields the same bound. For $d = 0$, every W_i is empty, its induced graph has one maximal empty clique, and the calculation gives n. In fact a zero-degeneracy graph cannot contain an edge, because that two-vertex subgraph would have minimum degree one. Its n singleton maximal cliques are consistent with the bound. The divisibility assumption is part of the requested sharp statement and will make the construction in part (c) possible; the upper-bound argument itself does not require that divisibility.

当 $n = d+3$ 时前缀为空，仅用尾类估计就得到相同上界；当 $d = 0$ 时，各 W_i 都空，其诱导图各有一个极大的空团，计算给出的上界为 n。事实上，零退化度图不能含边，否则两个端点组成的子图最小度为一，因此它的 n 个单点极大团也与该界一致。整除条件属于题目要求的可达到极值结论，并将用于 (c) 的构造，而上界论证本身不依赖这个整除条件。■

(c) Put $s = n - d$ and $t = d/3$, so $s \ge 3$ and t is a nonnegative integer. Make one part of size s and t further parts each of size three, with every cross-part edge and no within-part edge. A clique contains at most one vertex from each part. If it omits a part, any vertex of that part is adjacent to all vertices already chosen, so the clique is not maximal. Conversely, choosing one vertex from every part gives a clique, and any outside vertex lies in a part already represented and is not adjacent to the chosen vertex of that part. Such a clique is maximal. Thus the maximal cliques are exactly these choices.

(c) 记 $s = n - d$、$t = d/3$，则 $s \ge 3$ 且 t 为非负整数；构造一个大小为 s 的部，再加 t 个大小为三的部，异部之间全连、同部内部无边。团在每部至多选一个顶点，若漏掉某部，该部任一点都邻接已经选入的所有点，因此这个团尚不极大；反过来，每部恰选一点得到团，而任何外部顶点所在的部已经被代表，它与该部所选顶点不邻接，所以这样的团确实极大。极大团因而恰好对应逐部选点。

$$\#\{\text{maximal cliques}\} = s\,3^t = (n-d)3^{d/3}.$$

For the upper bound on degeneracy, order every vertex in the part of size s first. Each of them has exactly d later neighbors, all outside that part. The remaining graph has only d vertices, so every later vertex has at most $d-1$ later neighbors when $d > 0$; when $d = 0$ there are no remaining vertices. L4 proves degeneracy at most d. For the lower bound, inspect the whole graph: each vertex in the large part has degree d, while each vertex in a size-three part has degree $n - 3 \ge d$. Thus the whole graph has minimum degree d, so any bound smaller than d fails the definition. Degeneracy is exactly d.

先证明退化度上界，把大小为 s 的部中全部顶点排在最前，它们各有恰好 d 个后继邻居，全部来自该部外部；随后只剩 d 个点，若 $d>0$，每点的后继邻居至多为 $d-1$，若 $d=0$ 则已经没有剩余顶点，因此 L4 给出退化度至多为 d。再证下界，在整张图中，大部顶点的度都是 d，三点部顶点的度为 $n-3\geq d$，所以全图最小度恰为 d，任何更小的界都违反退化度定义，从而准确退化度等于 d。

If $d=0$, there is just one part of size n, giving an edgeless graph with degeneracy zero and exactly n singleton maximal cliques. If $n=d+3$, the large part also has size three, so all $t+1$ parts have size three; the same degree argument gives d, and the number of choices is $3^{t+1}=3\,3^{d/3}$. This agrees with the tail-only estimate in part (b). This construction proves attainability and covers the boundary cases above. Together with the universal upper bound proved in part (b), it completes the extremal statement under all its stated hypotheses.

若 $d=0$，图中只有一个大小为 n 的部，得到的就是退化度零、恰有 n 个单点极大团的无边图；若 $n=d+3$，大部也只有三个点，全部 $t+1$ 个部都为三点部，同一度数论证仍给出 d，选点数则为 $3^{t+1}=3\,3^{d/3}$，与 (b) 仅有尾类时的估计一致。该构造证明了可达到性，并覆盖上述边界情况；结合 (b) 已证的普适上界，便完成题设全部条件下的极值结论。■

Historical Notes / 历史注记

[BK1973] Coen Bron and Joep Kerbosch, “Algorithm 457: finding all cliques of an undirected graph,” *Communications of the ACM* 16(9), 575–577, September 1973. The DOI record identifies the publication; a university-hosted scan supplies readable source pages. The source review for this section covered printed pages 575–577, which are the first three PDF pages, stopping before the different algorithm note at the bottom of the third page. The modern set procedures, proofs, graphs, and exercises above are independently written teaching material.

[BK1973] Coen Bron 与 Joep Kerbosch 的《Algorithm 457: finding all cliques of an undirected graph》发表于 1973 年 9 月的 *Communications of the ACM* 16 卷 9 期，页码 575—577；上列 DOI 记录用于确认出版信息，大学网站扫描件用于读取正文。本节来源核查覆盖印刷页 575—577，也就是 PDF 前三页，并在第三页底部开始另一篇算法短文处停止；本文的现代集合过程、证明、图例与习题均为独立编写的教学内容。

The scan’s old Algol interface uses a true diagonal in its adjacency matrix together with its own candidate and counting conventions. Our finite simple graph has a false diagonal, and every pivot difference uses an open neighborhood. These conventions cannot be interchanged by copying a matrix initialization line. The publication date supports the introductory historical statement; it is not used to claim historical priority or performance on modern machines.

扫描件中的旧 Algol 接口采用邻接矩阵对角为真的约定，并与它自身的候选及计数处理配套；本节有限简单图的矩阵对角为假，枢轴差集始终使用开邻域，所以不能只复制一行矩阵初始化就互换两套约定。该出版日期支持正文的历史钩子，不用于推断历史首次地位或现代机器上的性能。

[ELS2010-v1] David Eppstein, Maarten Löffler, and Darren Strash, *Listing All Maximal Cliques in Sparse Graphs in Near-optimal Time*, author preprint arXiv:1006.5440v1, June 28, 2010; versioned PDF. The admitted reading scope is pages 1–5 and 8–9, with visual checks of pages 4, 5, and 8. Page 8 states the sharp degeneracy-based count used in Exercise 15 with both the divisibility and size conditions retained. The exercise develops the necessary general count from closed-neighborhood deletion, rather than asking the reader to cite an additional paper.

[ELS2010-v1] David Eppstein、Maarten Löffler 与 Darren Strash 的 *Listing All Maximal Cliques in Sparse Graphs in Near-optimal Time* 是 2010 年 6 月 28 日的作者预印本，本节固定使用 arXiv:1006.5440v1；准入阅读范围为第 1—5 页与第 8—9 页，并对第 4、5、8 页作过视觉核对。第 8 页给出了习题 15 使用的退化度极值计数，本节保留其中的整除条件与顶点数条件，并通过闭邻域删除完整建立所需的一般计数界，没有让读者借用另外一篇论文的结论。

The set movement in one sentence on page 4 is inconsistent with the recursive arguments on page 5; our movement is justified by L1 instead of inheriting that sentence. We also write an explicit return immediately after output, so the procedure never selects a pivot from an empty set. The preprint's dedicated implementation analysis is not reproduced here: its pages 6–7 were outside the admitted reading scope. Accordingly, the elementary procedures are charged by the separately proved call bound (4), not by importing the preprint's specialized running-time result. A maximal-clique count alone cannot charge internal calls, neighborhood work, or the length of printed vertex lists.

第 4 页有一句集合移动叙述与第 5 页的递归实参不一致，本节的移动方向由 L1 自行证明，不沿用那句话；输出之后也明确立即返回，避免从空集合中选择枢轴。预印本的专门实现分析没有在这里复建，第 6—7 页也不在准入阅读范围内，因此本文对朴素过程使用单独证明的调用界式（4）计费，没有移植预印本的专用运行时间结果；极大团个数本身无法计算内部调用、邻域处理或打印顶点列表的长度。