A Compendium of Problems Complete for P

(Preliminary)
December 20, 1991
RCS Revision: 1.46

Raymond Greenlaw[*]

H. James Hoover[†]

Walter L. Ruzzo[‡]

Department of Computing Science
University of Alberta
Technical Report TR 91–11

Department of Computer Science
University of New Hampshire
Technical Report TR 91–14

Department of Computer Science and Engineering
University of Washington
Technical Report TR 91–05–01

Abstract

This paper serves two purposes. Firstly, it is an elementary introduction to the theory of P-completeness — the branch of complexity theory that focuses on identifying the problems in the class P that are "hardest," in the sense that they appear to lack highly parallel solutions. That is, they do not have parallel solutions using time polynomial in the logarithm of the problem size and a polynomial number of processors unless all problem in P have such solutions, or equivalently, unless $P = NC$. Secondly, this paper is a reference work of P-complete problems. We present a compilation of the known P-complete problems, including several unpublished or new P-completeness results, and many open problems.

This is a **preliminary** version, mainly containing the problem list. The latest version of this document is available in electronic form by anonymous ftp from `thorhild.cs.ualberta.ca` (`129.128.4.53`) as either a compressed dvi file (`TR91-11.dvi.Z`) or as a compressed postscript file (`TR91-11.ps.Z`), or from `cs.washington.edu` (`128.95.1.4`) as a compressed postscript file (`tr/1991/05/uw-cse-91-05-01.ps.Z`).

[*]Author's address: Department of Computer Science, University of New Hampshire, Durham, NH 03824, e-mail address: `greenlaw@unh.edu`

[†]Research supported by the Natural Sciences and Engineering Research Council of Canada grant OGP 38937. Author's address: Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2H1, e-mail address: `hoover@cs.ualberta.ca`

[‡]Research supported in part by NSF grants ECS-8306622, CCR-8703196, CCR-9002891, and by NSF/DARPA grant CCR-8907960. A portion of this work was performed while visiting the University of Toronto, whose hospitality is gratefully acknowledged. Author's address: Department of Computer Science and Engineering, FR-35, University of Washington, Seattle, WA 98195, e-mail address: `ruzzo@cs.washington.edu`

Additional Problems

The author's would be pleased to receive suggestions, corrections, and additional problems. Please send new problems, bibliography entries and/or copies of the relevant papers to Ray Greenlaw at the address on the title page.

Contents

Part I: Background and Theory 1

1 Introduction 1

1.1 Setting the Scene 1
1.2 Feasibility 2
1.3 Completeness 3
1.4 Overview of this Paper 3

2 Parallel Models and Complexity Classes 5

2.1 Parallel Machines 5
2.2 Computation Problems versus Decision Problems 6
2.3 Complexity Classes 7
2.4 The Complexity Classes P and NC 8
2.5 Reducibility 9
2.6 Completeness 10

3 Two Basic P-complete Problems 12

3.1 The Generic P-complete Problem 12
3.2 The Circuit Value Problem 13

4 Evidence That NC Does Not Equal P 15

4.1 General Simulations Are Not Fast 15
4.2 Fast Simulations Are Not General 16
4.3 Natural Approaches Provably Fail 19
4.4 Summary 20

5 The Circuit Value Problem 21

5.1 The Circuit Value Problem is P-Complete 21
5.2 Restricted Versions of the Circuit Value Problem 23

6 Parallel Versions of Sequential Paradigms 32

6.1 Strong P-completeness 32
6.2 Approximations to P-complete problems 33
6.3 Greedy Algorithms 36

6.3.1 Lexicographically Ordered Greedy Algorithms 36
6.3.2 Generic Greedy Algorithms 38

7 Boolean Circuits **41**
7.1 Boolean circuits 41
7.2 Uniform Circuit Families 42
7.3 The Class NC^k 43
7.4 NC^k Reducibility 43
7.5 Other Notions of Reducibility 44
7.6 Random NC 45

8 Acknowledgements **46**

Part II: P-Complete and Open Problems **47**

A List of P-Complete Problems **47**
A.1 Circuit Complexity 47
A.1.1 Circuit Value Problem (CVP) 47
A.1.2 Topologically Ordered Circuit Value Problem (TopCVP) 47
A.1.3 Monotone Circuit Value Problem (MCVP) 48
A.1.4 Alternating Monotone Fanout 2 CVP (AM2CVP) 48
A.1.5 NAND Circuit Value Problem (NANDCVP) 48
A.1.6 Synchronous Alternating Monotone Fanout 2 CVP (SAM2CVP) . . 48
A.1.7 Planar Circuit Value Problem (PCVP) 49
A.1.8 Arithmetic Circuit Value Problem (*) (ArithCVP) 49
A.1.9 Min-Plus Circuit Value Problem (MinPlusCVP) 50
A.1.10 Inverting an NC^0 Permutation (*) (InvNC0Perm) 50
A.2 Graph Theory 50
A.2.1 Lexicographically First Maximal Independent Set (LFMIS) 50
A.2.2 Lexicographically First Maximal Clique (LFMC) 51
A.2.3 Alternating Graph Accessibility Problem (AGAP) 51
A.2.4 Hierarchical Graph Accessibility Problem (HGAP) 51
A.2.5 Restricted Chromatic Alternating Graph Accessibility Problem (RCAGAP) 52
A.2.6 Lexicographically First $\Delta + 1$ Vertex Coloring (LFDVC) 52

A.2.7 High Degree Subgraph (HDS) 53
A.2.8 High Connectivity Subgraph (HCS) 54
A.2.9 Ordered High Degree Vertex Removal (OHDVR) 54
A.2.10 Ordered Low Degree Vertex Removal (OLDVR) 55
A.2.11 Ordered Vertices Remaining (OVR) 55
A.2.12 Nearest Neighbor Traveling Salesman Heuristic (NNTSH) 55
A.2.13 Lexicographically First Maximal Subgraph for π (LFMS(π)) 56
A.2.14 Minimum Feedback Vertex Set (MFVS) 56
A.2.15 Edge Maximal Acyclic Subgraph (EMAS) 57
A.2.16 General Graph Closure (GGC) 58
A.3 Search Problems 59
A.3.1 Lexicographically First Maximal Path (LFMP) 59
A.3.2 Lexicographically First Depth First Search Ordering (LFDFS) . . . 59
A.3.3 Breadth-Depth Search (BDS) 60
A.3.4 Stack Breadth First Search (SBFS) 60
A.3.5 Alternating Breadth First Search (ABFS) 61
A.4 Combinatorial Optimization and Flow Problems 62
A.4.1 Linear Inequalities (LI) 62
A.4.2 Linear Equalities (LE) 62
A.4.3 Linear Programming (*) (LP) 62
A.4.4 Maximum Flow (MaxFlow) 63
A.4.5 Homologous Flow (HF) 64
A.4.6 Lexicographically First Blocking Flow (LFBF) 64
A.4.7 First Fit Decreasing Bin Packing (FFDBP) 64
A.4.8 General List Scheduling (GLS) 65
A.5 Local Optimality 66
A.5.1 MAXFLIP Verification (MAXFLIPV) 66
A.5.2 Local Optimality Kernighan-Lin Verification (LOKLV) 66
A.5.3 Unweighted, Not-All-Equal Clauses, 3SAT/FLIP (U3NSATFLIP) . . 66
A.5.4 Unweighted MAXCUT/SWAP (UMS) 67
A.5.5 Unweighted 2SAT/FLIP (U2SATFLIP) 67
A.5.6 Unweighted SWAP (USWAP) 67
A.5.7 Unweighted STABLE NET (USN) 67

A.6 Logic 68
A.6.1 Unit Resolution (UNIT) 68
A.6.2 Horn Unit Resolution (HORN) 68
A.6.3 Propositional Horn Clause Satisfiability (PHCS) 68
A.6.4 Relaxed Consistent Labeling (RCL) 69
A.6.5 Generability (GEN) 69
A.6.6 Path Systems (PATH) 70
A.6.7 Unification (UNIF) 70
A.6.8 Logical Query Program (LQP) 70
A.7 Formal Languages 71
A.7.1 Context-Free Grammar Membership (CFGmem) 71
A.7.2 Context-Free Grammar Empty (CFGempty) 71
A.7.3 Context-Free Grammar Infinite (CFGinf) 72
A.7.4 Context-Free Grammar ϵ-Membership (CFGϵmem) 72
A.7.5 Straight-Line Program Membership (SLPmem) 72
A.7.6 Straight-Line Program Nonempty (SLPnonempty) 72
A.7.7 Labeled GAP (LGAP) 73
A.7.8 Two-Way DPDA Acceptance (2DPDA) 73
A.8 Algebraic Problems 74
A.8.1 Finite Horizon Markov Decision Process (FHMDP) 74
A.8.2 Discounted Markov Decision Process (DMDP) 74
A.8.3 Average Cost Markov Decision Process (ACMDP) 75
A.8.4 Gaussian Elimination with Partial Pivoting (GEPP) 75
A.8.5 Iterated Mod (IM) 76
A.8.6 Generalized Word Problem (GWP) 76
A.8.7 Subgroup Containment (SC) 77
A.8.8 Subgroup Equality (SE) 77
A.8.9 Subgroup Finite Index (SFI) 77
A.8.10 Group Independence (GI) 78
A.8.11 Group Rank (GR) 78
A.8.12 Group Isomorphism (SI) 78
A.8.13 Group Induced Isomorphism (GII) 78
A.8.14 Intersection of Cosets (IC) 79

A.8.15 Intersection of Subgroups (IS) 79
A.8.16 Group Coset Equality (GCE) 79
A.8.17 Conjugate Subgroups (CS) 79
A.8.18 Uniform Word Problem for Finitely Presented Algebras (UWPFPA) 80
A.8.19 Triviality Problem for Finitely Presented Algebras (TPFPA) 80
A.8.20 Finitely Generated Subalgebra (FGS) 80
A.8.21 Finiteness Problem for Finitely Presented Algebras (FPFPA) 81
A.8.22 Uniform Word Problem for Lattices (UWPL) 81
A.8.23 Generator Problem for Lattices (GPL) 82
A.8.24 Boolean Recurrence Equation (BRE) 82
A.9 Geometry 82
A.9.1 Plane Sweep Triangulation (PST) 82
A.9.2 3-Oriented Weighted Planar Partitioning (3OWPP) 83
A.9.3 Visibility Layers (VL) 84
A.10 Real Analysis 85
A.10.1 Real Analogue to CVP (RealCVP) 85
A.10.2 Fixed Points of Contraction Mappings (FPCM) 85
A.10.3 Inverting a One-to-One Real Functions (IOORF) 86
A.11 Miscellaneous 86
A.11.1 General Deadlock Detection (GDD) 86
A.11.2 Two Player Game (GAME) 87
A.11.3 Cat & Mouse (CM) 87
A.11.4 Longcake (LONGCAKE) 88

B Open Problems **89**
B.1 Algebraic Problems 89
B.1.1 Integer Greatest Common Divisor (IntegerGCD) 89
B.1.2 Extended Euclidean Algorithm (ExtendedGCD) 89
B.1.3 Relative Primeness (RelPrime) 89
B.1.4 Modular Inversion (ModInverse) 89
B.1.5 Modular Powering (ModPower) 89
B.1.6 Short Vectors (SV) 90
B.1.7 Short Vectors Dimension 2 (SV2) 90
B.1.8 Sylow Subgroups (SylowSub) 90

B.1.9 Two Variable Linear Programming (TVLP) 90
B.1.10 Univariate Polynomial Factorization over Q (UPFQ) 91
B.2 Graph Theory Problems 91
B.2.1 Edge-Weighted Matching (EWM) 91
B.2.2 Bounded Degree Graph Isomorphism (BDGI) 91
B.2.3 Graph Closure (GC) 91
B.2.4 Restricted Lexicographically First Independent Set (RLFMIS) . . . 91
B.2.5 Maximal Path (MP) 92
B.2.6 Maximal Independent Set Hypergraph (MISH) 92
B.3 Geometric Problems 92
B.3.1 2-Oriented Weighted Planar Partitioning (2OWPP) 92
B.3.2 Limited Reflection Ray Tracing (LRRT) 92
B.3.3 Restricted Plane Sweep Triangulation (SWEEP) 93
B.3.4 Successive Convex Hulls (SCH) 93
B.3.5 Unit Length Visibility Layers (ULVL) 93
B.4 *CC* Problems 93
B.4.1 Comparator Circuit Value Problem (CCVP) 94
B.4.2 Lexicographically First Maximal Matching (LFMM) 94
B.4.3 Stable Marriage (SM) 94
B.5 *RNC* Problems 95
B.5.1 Directed or Undirected Depth First Search (DFS) 95
B.5.2 0-1 Maximum Flow (0-1 MaxFlow) 95
B.5.3 Maximum Matching (MM) 95
B.5.4 Perfect Matching Existence (PME) 95

C Undigested Papers **97**

References **98**

Problem Index **113**

8 Acknowledgements

We wish to thank the following for contributing new problems and pointing us to appropriate literature: Richard Anderson, Paul Beame, Anne Condon, Steve Cook, Derek Corneil, Joachim von zur Gathen, Arvind Gupta, John Hershberger, David Johnson, Luc Longpré, Mike Luby, Pierre McKenzie, Andrew Malton, Satoru Miyano, John Reif, Jack Snoeyink, Paul Spirakis, Iain Stewart, Ashok Subramanian, Eva Tardos, Martin Tompa, and H. Venkateswaran.

Part II: P-Complete and Open Problems

A List of P-Complete Problems

This section contains a list of P-complete problems. For each entry we give a description of the problem and its input, references to source papers showing the problem P-complete, a hint illustrating the main idea of the completeness reduction, and mention related versions of the problem.

Problems marked by (*) are P-hard but are not known to be in P. Other problems are in P, although we usually omit the proofs that they are.

Many of the problems given here were originally shown P-complete with respect to log space reduction. Any log space computation is immediately in NC^2 by Borodin's simulation [Bor77]. Thus any problem log space complete for P is also $\leq_m^{NC^2}$ complete for P. In most cases the same reduction can be done in NC^1, i.e. the problem is $\leq_m^{NC^1}$ complete for P. We have noted some exceptions to this below, but have not been exhaustive.

The problems are divided into the following categories: circuit complexity, graph theory, search, combinatorial optimization and flow, local optimality, logic, formal languages, algebraic, geometry, real analysis, and miscellaneous.

A.1 Circuit Complexity

The circuit value problem (CVP) plays the same role in P-completeness theory that satisfiability does in NP-completeness theory. In this section we give many variants of CVP that are P-complete and are particularly useful for proving other problems are P-complete. See Section 5 of this paper for more details.

A.1.1 Circuit Value Problem (CVP)

Given: An encoding $\overline{\alpha}$ of a Boolean circuit α plus inputs $x_1, \ldots, x_n$.
Problem: Does α on input $x_1, \ldots, x_n$ output 1?
Reference: [Lad75]
Hint: A proof is given in Section 5 of this paper.
Remarks: For the two input basis of Boolean functions, it is known that CVP is P-complete except when the basis consists solely of OR, consists solely of AND, or consists of any or all of the following: XOR, EQUIVALENCE, and NOT [GP86, Par87].

A.1.2 Topologically Ordered Circuit Value Problem (TopCVP)

Given: An encoding $\overline{\alpha}$ of a Boolean circuit α plus inputs $x_1, \ldots, x_n$, with the additional assumption that the vertices in the circuit are numbered and listed in topological order.

Problem: Does α on input $x_1, \ldots, x_n$ output 1?
Reference: Folklore.
Hint: A proof is given in Theorem 5.3. Also see the remarks following the proof of Theorem 5.7.
Remarks: All of the reductions in Sections 5 and A.1 preserve topological ordering, so the restrictions of all of these variants of the Circuit Value Problem to topologically ordered instances remain P-complete.

A.1.3 Monotone Circuit Value Problem (MCVP)

Given: An encoding $\overline{\alpha}$ of a *monotone* Boolean circuit α, that is one constructed solely of AND and OR gates, plus inputs $x_1, \ldots, x_n$.
Problem: Does α on input $x_1, \ldots, x_n$ output 1?
Reference: [Gol77]
Hint: Reduce CVP to MCVP. A proof is given in Section 5.2.

A.1.4 Alternating Monotone Fanout 2 CVP (AM2CVP)

Given: An encoding $\overline{\alpha}$ of a monotone Boolean circuit α, plus inputs $x_1, \ldots, x_n$. On any path from an input to an output the gates are required to *alternate* between OR and AND gates. Inputs are required to be connected only to OR gates, and outputs must come directly from OR gates. The circuit is restricted to have fanout exactly two for inputs and internal gates, and to have a distinguished OR gate as output.
Problem: Does α on input $x_1, \ldots, x_n$ output 1?
Reference: Folklore.
Hint: A proof is given in Section 5.2.
Remarks: [GSS82] gave a completeness proof for monotone, fanout 2 CVP.

A.1.5 NAND Circuit Value Problem (NANDCVP)

Given: An encoding $\overline{\alpha}$ of a Boolean circuit α constructed only of NAND gates, plus inputs $x_1, \ldots, x_n$. The circuit is restricted to have fanout two for inputs and NAND gates.
Problem: Does α on inputs $x_1, \ldots, x_n$ output 1?
Reference: Folklore.
Hint: Reduction of AM2CVP to NANDCVP. A proof is given in Section 5.2.
Remarks: Any *complete* basis of gates suffices, by the obvious simulation of NAND gates in the other basis. For example, NOR gates form a complete basis. NOR CVP is defined analogously to NANDCVP. See Post [Pos41] for a characterization of complete bases. See the remarks for Problem A.1.1 for other bases, not necessarily complete, for which the associated circuit value problem is still complete.

A.1.6 Synchronous Alternating Monotone Fanout 2 CVP (SAM2CVP)

Given: An encoding $\overline{\alpha}$ of a monotone Boolean circuit α, plus inputs $x_1, \ldots, x_n$. In addition to the restrictions of Problem A.1.4, this version requires the circuit to be *synchronous*. That

is, each level in the circuit can receive its inputs only from gates on the preceding level.
Problem: Does α on input $x_1, \ldots, x_n$ output 1?
Reference: [GHR91]
Hint: A proof is given in Section 5.2. The reduction is from AM2CVP.

A.1.7 Planar Circuit Value Problem (PCVP)

Given: An encoding of a *planar* Boolean circuit α, that is one whose graph can be drawn in the plane with no edges crossing, plus inputs $x_1, \ldots, x_n$.
Problem: Does α on input $x_1, \ldots, x_n$ output 1?
Reference: [Gol77, McC81]
Hint: Reduce CVP to PCVP. Lay out the circuit and use cross-over circuits to replace crossing lines with a planar subcircuit. A planar XOR circuit can be built from two each $\wedge, \vee, \neg$ gates; a planar cross-over circuit can be built from three planar XOR circuits. An m gate CVP instance is embedded in an $m \times m$ grid as follows. Gate i will be in cell (i, i), with its value sent along a wire in the i^{th} row both to the left and to the right. Gate i's inputs are delivered to it through two wires in the i^{th} column, with data flowing down the wires from above and up from below. Let gate i's inputs be the outputs of gates j and k, and suppose j happens to be less than i. In cell (j, i) (which happens to be above (i, i)) at the point where j's horizontal (rightgoing) output wire crosses i's first (vertical, downgoing) input wire, insert a two input, two output planar subcircuit that discards the value entering from above and passes the value entering from the left both to the right and down. The input to i from k is treated similarly, with the obvious changes of orientation if $k > i$. At all other wire crossings, insert a copy of the planar crossover circuit. Note that given i and j, the wiring of cell (i, j) is easily determined based on whether $i < j$, $i = j$, i is an input to j, etc. Hence the reduction can be performed in NC^1, (even if the original circuit is not topologically sorted.)
Remarks: It is easy to see that monotone planar crossover networks do not exist, so the reduction above cannot be done in the monotone case. In fact, the monotone version of PCVP is in $LOGCFL \subseteq NC^2$ [DC80, DC89, Gol80] when all inputs appear on one face of the planar embedding. The more general problem where inputs may appear anywhere is also known to be in NC [Kos90, DK91].

A.1.8 Arithmetic Circuit Value Problem (*) (ArithCVP)

Given: An encoding of an *arithmetic circuit* α with dyadic operations $+, -, *$ and inputs $x_1, \ldots, x_n$ from a ring.
Problem: Does α on input $x_1, \ldots, x_n$ output 1?
Reference: [Ven83]
Hint: Reduce NANDCVP to ArithCVP as follows: TRUE $\rightarrow 1$, FALSE $\rightarrow 0$, and $\neg(u \wedge v) \rightarrow 1 - u * v$, where 0 denotes the additive identity and 1 denotes the multiplicative identity of the ring.
Remarks: The problem is not necessarily in FP for infinite rings like Z or Q, since intermediate values need not be of polynomial length. It will be in FP for any finite ring, and remains P-hard in any ring. It is also P-complete to decide whether all gates in an

arithmetic circuit over Z have "small" values, say values of a magnitude 2^{n^k} for some constant k. Is in NC for circuits of degree $2^{\log^{O(1)}}$, where the *degree* of a node is 1 for inputs, and $d_1 + d_2$ ($\max(d_1, d_2)$) when the node is product (respectively, sum) of the values computed by two nodes of degree d_1 and d_2 [VSBR83, MRK88].

A.1.9 Min-Plus Circuit Value Problem (MinPlusCVP)

Given: An encoding of a $(\min, +)$ circuit α and rational inputs $x_1, \ldots, x_n$.
Problem: Does α on input $x_1, \ldots, x_n$ output a non-zero value?
Reference: [Ven83]
Hint: Reduce MCVP to MinPlusCVP as follows: TRUE $\rightarrow 1$, FALSE $\rightarrow 0$, $u \wedge v \rightarrow \min(u, v)$, and $u \vee v \rightarrow \min(1, u + v)$.
Remarks: The above reduction works in any ordered semi-group with additive identity 0 and an element 1 such that $1 + 1 \geq 1 > 0$. If there is a non-zero element 1 such that such that $1 + 1 = 0$ (e.g., in Z_2) then reduce NANDCVP via $\neg(u \wedge v) \rightarrow 1 + \min(u, v)$. In a *well-ordered* semigroup where 0 is the minimum element, one or the other of these cases holds. If the semigroup is infinite, the problem may not be in P.

A.1.10 Inverting an NC^0 Permutation (*) (InvNC0Perm)

Given: An n-input, n-output NC^0 circuit computing a bijective function $f : \{0,1\}^n \rightarrow \{0,1\}^n$ and $y \in \{0,1\}^n$.
Problem: Is the last bit of $f^{-1}(y)$ equal to 1?
Reference: [Hås87, Section 2.5], [Hås88]
Hint:
Remarks: Not known to be in P in general, although the family of permutations used to show P-hardness is polynomial time invertible.

A.2 Graph Theory

A.2.1 Lexicographically First Maximal Independent Set (LFMIS)

Given: An undirected graph G with an ordering on the vertices and a designated vertex v.
Problem: Is vertex v in the lexicographically first maximal independent set of G?
Reference: [Coo85]
Hint: A proof is given in Section 6.3.1.
Remarks: This is an instance of Problem A.2.13. LFMIS is P-complete for bipartite or planar graphs restricted to degree at most 3 [Miy89]. Karp observed that the completeness of LFMIS implies that determining the i^{th} node chosen by any deterministic sequential algorithm for either LFMIS or LFMC (Problem A.2.2) is also complete [Kar84]. Computing or approximating the size of the lexicographically first maximal independent set is also P-complete; see Section 6.2. Karp and Wigderson gave the first NC algorithm for finding a maximal independent set [KW85], subsequently improved by Luby [Lub86], by Alon, Babai and Itai [ABI86], and by Goldberg and Spencer [GS89]. These algorithms do not compute the *lexicographically first* maximal independent set.

A.2.2 Lexicographically First Maximal Clique (LFMC)

Given: An undirected graph G with an ordering on the vertices and a designated vertex v.
Problem: Is vertex v in the lexicographically first maximal clique of G?
Reference: [Coo85]
Hint: Finding a maximal clique is equivalent to finding a maximal independent set in the complement graph of G (Problem A.2.1).

A.2.3 Alternating Graph Accessibility Problem (AGAP)

Given: A directed graph $G = (V, E)$, a partition $V = A \cup B$ of the vertices, and designated vertices s and t.
Problem: Is $apath(s, t)$ TRUE, where *apath* is defined as follows. Vertices in A are "universal," those in B are "existential." Such a graph is called an *alternating* graph or an $AND\backslash OR$ graph. Then $apath(x, y)$ holds if and only if

1. $x = y$, or
2. x is existential and there is a z with $(x, z) \in E$ and $apath(z, y)$, or
3. x is universal and for all z with $(x, z) \in E$, $apath(z, y)$ holds.

Reference: [CKS81, Imm81, Imm83]
Hint: Reduce AM2CVP (Problem A.1.4) to AGAP. Create two existential nodes 0 and 1. Put edge $(x_i, 0)$ into E if input x_i is 0, and edge $(x_i, 1)$ into E if input x_i is 1. AND gates are universal nodes and OR gates are existential nodes. Inputs to a gate correspond to children in the alternating graph. For output z, $apath(z, 1)$ holds if and only if the output z is 1.
Remarks: The original proof simulated an alternating Turing machine (ATM) directly to show that AGAP was complete for ATM log space [Imm81]. Since ASPACE($\log n$) = P [CKS81], this showed AGAP was P-complete too. When this problem is generalized to hierarchical graphs it remains in P, provided the graph is "breadth-first ordered;" see [LW87]. The proof sketched above also shows that the problem remains P-complete when the partition (A, B) induces a bipartite graph. When restricted to only existential nodes, the problem is equivalent to the "directed graph accessibility problem," variously called "GAP" and "STCON," and known to be complete for NL [Sav70]. Peterson (personal communication, 198x) shows that the undirected version of AGAP is also P-complete. When restricted to undirected graphs with only existential nodes, this problem is equivalent to the "undirected graph accessibility problem," called "UGAP" or "USTCON," which is known to be complete for the special case of nondeterministic log space known as symmetric log space (SL) [LP82].

A.2.4 Hierarchical Graph Accessibility Problem (HGAP)

Given: A *hierarchical graph* $G = (V, E)$ and two designated vertices s and t. A hierarchical graph $\Gamma = (G_1, \ldots, G_k)$ consists of k subcells G_i, $1 \leq i \leq k$. Each subcell is a graph that

contains three types of vertices called *pins*, *inner vertices*, and *nonterminals*. The pins are the vertices through which the subcell can be connected to from the outside. The inner vertices cannot be connected to from the outside. The nonterminals stand for previously defined subcells. A nonterminal inside G_i has a *name* and a *type*. The name is a unique number or string. The type is a number from $1, \ldots, i-1$. A nonterminal v of type j stands for a copy of subcell G_j. The neighbors of v are in a one-to-one correspondence with the pins of G_j via a mapping that is specified as part of Γ.
Problem: Is there a path between s and t in the *expansion graph* of G? An expansion graph is a hierarchical graph *expanded*. The graph is expanded by expanding cell G_k recursively. To expand subcell G_i expand its subcells $G_1, \ldots, G_{i-1}$ recursively and replace each nonterminal of v of type j with a copy of the expansion of subcell G_j.
Reference: [LW87]
Hint: The reduction is from the alternating graph accessibility problem, Problem A.2.3.
Remarks: Hierarchical versions of the following problems are also P-complete: graph accessibility in undirected graphs, determining whether a directed graph contains a cycle, and determining whether a given graph is bipartite [LW87]. There are several other very restricted versions of hierarchical graph problems that are P-complete [LW87].

A.2.5 Restricted Chromatic Alternating Graph Accessibility Problem (RCAGAP)

Given: An alternating graph $G = (V, E)$, two natural numbers k and m (where $k \leq m \leq \log |V|$), a coloring $c : E \to \{1, \ldots, m\}$, and two vertices s and t. (Note that the coloring is an unrestricted assignment of colors to the edges. It may assign the same color to several edges incident to a common vertex. See Problem A.2.3 for the definition of an alternating graph.)
Problem: Are there k different colors $i_1, \ldots, i_k \in \{1, \ldots, m\}$ such that $apath(s, t)$ holds in the subgraph of G induced by the edges with colors $i_1, \ldots, i_k$. (See Problem A.2.3 for the definition of *apath*.
Reference: [LW87]
Hint: There is a trivial reduction from the alternating graph accessibility problem, Problem A.2.3. Membership of RCAGAP in P follows from membership of AGAP in P, since there are at most $2^k \leq |V|$ possible sets of colors to try.
Remarks: The problem remains P-complete if the vertices are restricted to being breadth first ordered [LW87]. When generalized to hierarchical graphs, the problem becomes NP-complete [LW87].

A.2.6 Lexicographically First $\Delta + 1$ Vertex Coloring (LFDVC)

Given: A graph $G = (V, E)$ with Δ equal to the maximum degree of any vertex in V, and an ordering $v_1, \ldots, v_n$ on the vertices of V, a designated color c, and a vertex v.
Problem: Is vertex v colored with color c in the lexicographically least *coloring* of the vertices of G? The coloring uses at most $\Delta + 1$ colors. If c_i is the color of v_i, where $c_i \in \{1, \ldots, \Delta + 1\}$, then each coloring corresponds to a $(\Delta + 1)$-ary number, and the least coloring is well-defined.

Reference: [Lub84]
Hint: Computing the lexicographically least coloring is easily done in polynomial time by examining each vertex in order and coloring it with the smallest available color. To show completeness, reduce NANDCVP to LFDVC. The coloring will correspond to evaluating the circuit in topological order. Let $v_1, \ldots, v_n$ be the gates of a circuit α and assume without loss of generality that the gates are numbered in topological order. Each gate in the circuit will be represented by four vertices in G, with the order on the vertices induced from the order of the gates. Consider a NAND gate, $v_i \leftarrow \neg(v_j, v_k)$. Introduce three new vertices, v'_i, v'_j, v'_k where v'_j, v'_k appear after v_j, v_k, and v'_i appears after all these but before v_i in the topological ordering. The gate is then represented by the edges (v_j, v'_j), (v_k, v'_k), (v'_j, v'_i), (v'_k, v'_i), and (v'_i, v_i). One final fix is necessary. To keep the degree down to 3, a fanout tree may be required on the output of each gate. The resulting graph can be colored with only 3 colors in the order $\{T, F, X\}$ (even though 4 might be necessary for a different ordering).
Remarks: The problem of $\Delta - 1$ coloring is NP-complete [GJ79]. For graphs that are not an odd cycle or complete, a Δ coloring can be found in polynomial time (Brook's theorem, see [BM76]). However, this is not necessarily the lexicographically first. The $\Delta + 1$ vertex coloring is NC^1 reducible to finding a maximal independent set [Lub84], but the maximal independent set algorithm [KW85] does not produce the lexicographically first maximal independent set. It is possible to color a graph with Δ colors in NC [KN88, HS87, BK86], although the coloring produced is not the lexicographically first. There is an NC algorithm for 5 coloring planar graphs [Nao87].

A.2.7 High Degree Subgraph (HDS)

Given: A graph $G = (V, E)$ and an integer k.
Problem: Does G contain a vertex induced subgraph with minimum degree at least k?
Reference: [AM84a]
Hint: Reduce AM2CVP to HDS. The proof illustrated here is for $k = 3$, although it can be generalized to any fixed $k \geq 3$. A TRUE input k_1 connected to gate i is represented by a gadget with five vertices k_1, v_1, v_2, v_3, and k'_1. The edges in the gadget are (v_1, k_1), (v_1, k'_1), (v_1, v_2), (v_2, k'_1), (v_2, v_3), (v_3, k_1), (v_3, k'_1), and k'_1 is connected to a vertex in the gadget for gate i as described below. A FALSE input is represented by a single vertex. An AND gate i with inputs l_1 and l_2, and outputs l'_1 and l'_2 is represented by a fourteen vertex gadget. The gadget is composed of two of the gadgets used to represent TRUE inputs and an additional four vertices. l'_1 and l'_2 label the vertices corresponding to k'_1 in the TRUE input gadget. w_1 and w_2 label the positions corresponding to k_1 in their respective copy of the TRUE input gadget. The four additional vertices are labeled l_1, l_2, w_3, and w_4. l_1, l_2, and w_3 are connected into a three clique. w_4 is connected to w_1, w_2, and w_3. Inputs to gate i are connected to l_1 and l_2, and the outputs l'_1 and l'_2 are connected to the appropriate input positions of other gates. The representation of an OR gate is very similar to the AND gadget, omitting w_3 and connecting l_1, l_2 directly to w_4. Finally, there is a binary tree that has as its leaves the vertices corresponding to the k_1's of the TRUE inputs and has the vertex corresponding to the output vertex of the circuit as its root. The computation of a HDS of degree 3 proceeds on this new graph so that HDS is nonempty if and only if the output of the circuit is TRUE.

Remarks: Although not stated as a "lexicographically first" problem, the HDS in a graph is unique, hence this is another instance of Problem A.2.13. There is an NC algorithm computing a HDS for $k = 2$. Let $K(G)$ denote the largest k such that there is an induced subgraph of G with minimum degree k. For fixed $0 \leq c \leq 1$, consider finding an approximation d such that $K(G) \geq d \geq cK(G)$. For any $c < 1/2$ there is an NC algorithm for finding d, and for any $c > 1/2$ the problem of finding d is P-complete [AM84a]. A "complementary" low degree subgraph problem has also been studied and for several natural decision problems it is NP-complete [Gre89]. Decision problems based on ordered vertex removal relating to subgraph computations are also P-complete [Gre89]. A special case of HDS is the Color Index Problem [VS88]: given an undirected graph $G = (V, E)$, is the color index of G less than or equal to four? The *color index* is the maximum over all subgraphs H, of G, of the minimum degree of H. Asking if the color index is ≤ 4 is the complement to asking if there are any high degree subgraphs of order 5. The original reduction is from the ordered low degree vertex removal problem, Problem A.2.10. See remarks for Problem A.2.6.

A.2.8 High Connectivity Subgraph (HCS)

Given: A graph $G = (V, E)$ and an integer k.
Problem: Does G contain a vertex induced subgraph of vertex (edge) connectivity at least k?
Reference: [KSS89, Ser90]
Hint: The reduction is from MCVP and is similar to that used to prove Problem A.2.7, the high degree subgraph problem, is P-complete.
Remarks: Approximation algorithms for this problem exhibit a threshold type behavior. Below a certain value on the absolute performance ratio the problem remains P-complete for fixed k, and above that ratio there are NC algorithms to approximate the problem [SS89]. Specifically, let o be the maximum size of a k vertex-connected induced subgraph of G. Then for $0 \leq c \leq 1/4$ it is possible to find, in NC, a vertex induced subgraph of size $\geq co$, but for $1/4 < c \leq 1$ this not possible unless $NC = P$. For edge-connectivity, the threshold is $c = 1/2$.

A.2.9 Ordered High Degree Vertex Removal (OHDVR)

Given: An undirected graph $G = (V, E)$ with a numbering on the vertices in V and two designated vertices u and v.
Problem: Is there an *elimination order* on V, $v_1, \ldots, v_n$, satisfying the properties that u is eliminated before v and for $1 \leq i \leq n$, v_i is the lowest numbered vertex of maximum degree in the $(i-1)$-st remaining subgraph of G? An elimination order is a sequence of vertices ordered as they and their corresponding edges are to be deleted from the graph.
Reference: [Gre89]
Hint: The reduction is from NAND CVP with fanin and fanout restrictions to 2. The circuit is transformed directly into a graph. The vertices in the graph are ordered so that gates evaluating to FALSE in the circuit are deleted first in the instance of OHDVR. A special vertex of degree four is added and its removal order is compared with that of the

vertex corresponding to the output gate of the circuit.

A.2.10 Ordered Low Degree Vertex Removal (OLDVR)

Given: An undirected graph $G = (V, E)$ with a numbering on the vertices in V and two designated vertices u and v.
Problem: Is there an *elimination order* on V, $v_1, \ldots, v_n$, satisfying the properties that u is eliminated before v and for $1 \leq i \leq n$, v_i is the lowest numbered vertex of minimum degree in the $(i-1)$-st remaining subgraph of G?
Reference: [VS88, Gre89]
Hint: This is the complementary problem to Problem A.2.9. The problem defined in [VS88] is more restricted than the one presented here. Their graphs are also required to have the property that u appears before v in some minimum elimination sequence if and only if u appears before v in all minimum degree elimination sequences.

A.2.11 Ordered Vertices Remaining (OVR)

Given: An undirected graph $G = (V, E)$ with a numbering on the vertices in V, a designated vertex u, and an integer k.
Problem: Is there an *elimination order* on V, $v_1, \ldots, v_n$, satisfying the properties that $u = v_j$ for some $j < (n-k)$ and for $1 \leq i \leq n$, v_i is the lowest numbered vertex of maximum degree in the $(i-1)$-st remaining subgraph of G?
Reference: [Gre89]
Hint: The reduction is from Problem A.2.9.
Remarks: The ordered low degree subgraph membership problem is also P-complete [Gre89]. The problem here is to determine whether a designated vertex is in a remaining subgraph when all vertices in that remaining subgraph have small degree.

A.2.12 Nearest Neighbor Traveling Salesman Heuristic (NNTSH)

Given: A distance matrix D with entries (d_{ij}) and two distinguished vertices s and l.
Problem: Does the *nearest neighbor tour* starting at s visit l as the last vertex before completing the tour at s? The nearest neighbor tour is a greedy heuristic that always chooses the nearest unvisited vertex as the next vertex on the tour.
Reference: [KLS89]
Hint: Reduce NNTSH to NAND CVP. Without loss of generality, assume the gates are numbered in topological order. Gate k with inputs i_1 and i_2, and outputs o_1 and o_2 is replaced by the gadget described below. The gadget has vertices $A, i_1, i'_1, i_2, i'_2, o'_1, o_1, o_2, o'_2$, and B. Let the triple (x, y, z) mean the distance between x and y is d. The triples in the gadget are $(A, B, 3k+2), (A, i_1, 3k), (A, i_2, 3k+1), (i_1, i'_1, 0), (i_2, i'_2, 0), (i'_1, o'_1, 3k+1), (i'_2, o'_1, 3k), (o'_1, o_1, 0), (o_1, o_2, 3k), (o_2, o'_2, 0)$, and $(o'_2, B, 3k)$. Vertex B of gate k is connected to vertex A of gate $k+1$. The distances between vertices that have been left unspecified are assumed to be very large. The edges between "i" vertices and those between "o" vertices represent inputs and outputs respectively. An edge included (not included) in the tour represents a TRUE (FALSE) value. TRUE circuit inputs are "chained" together and

the tour begins at the first TRUE input. By inserting a new node C before vertex B in the last gadget, and connecting B and C to the first TRUE input in the input chain, the tour constructed by the NNTSH is such that B (C) is visited last if and only if the circuit evaluates to TRUE (FALSE).
Remarks: The nearest merger, nearest insertion, cheapest insertion, and farthest insertion heuristics are all P-complete [KLS89]. The double minimum spanning tree and nearest addition heuristics are in NC [KLS89].

A.2.13 Lexicographically First Maximal Subgraph for π (LFMS(π))

Given: A graph $G = (V, E)$ with an ordering on V, a designated vertex v, and a *polynomial time testable, nontrivial, hereditary* property π. A property is nontrivial if there are infinitely many graphs that satisfy the property and at least one graph that doesn't. A property π is hereditary on induced subgraphs if whenever G satisfies π so do all vertex induced subgraphs.
Problem: Is v in the lexicographically first maximal subgraph H of G that satisfies π?
Reference: [Miy89]
Hint: Given a property π that is nontrivial and hereditary, Ramsey's theorem implies that either π is satisfied by all cliques or by all independent sets of vertices. This observation combined with the facts that the lexicographically first maximal clique problem, Problem A.2.2, and the lexicographically first maximal independent set problem are P-complete are used to show LFMS(π) is P-complete.
Remarks: The following are examples of properties that meet the criteria stated in the problem: bipartite, chordal, clique, comparability graph, edge graph, forest, independent set, outerplanar, and planar. Not all problems computing a lexicographically first solution are P-complete. For example, the lexicographically first topological order problem is complete for NLOG [Sho89] and the lexicographic low degree subgraph membership problem is NP-complete [Gre89].

A.2.14 Minimum Feedback Vertex Set (MFVS)

Given: A directed graph $G = (V, E)$ that is *cyclically reducible* (defined below) and a designated vertex v.
Problem: Is v contained in the minimum feedback set of G that is computed by the algorithm given in [WLS85]?
Reference: [BDAP88]
Hint: We review some terminology [WLS85]. A node z of G is *deadlocked* if there is a directed path in G from z to a node y that lies on a directed cycle. The *associated graph of node x with respect to G, A(G,x)*, consists of node x and all nodes of G that are not deadlocked if x is removed from G. A directed graph is *cyclically reducible* if and only if there exists a sequence of nodes $(y_1, \ldots, y_k)$ such that each of the graphs $A(G_{i-1}, y_i)$ is cyclic, where $G_0 = G$ and $G_i = G_{i-1} - A(G_i, y_i)$, for $1 \leq i \leq k$.

A set is called a *feedback vertex set* if it contains at least one vertex from every cycle [Kar72]. It is *minimum* if no other feedback vertex set has fewer elements. Wang, Lloyd, and Soffa gave a polynomial time algorithm for computing feedback sets in cyclically reducible graphs [WLS85]. Thus, MFVSP is in P. The reduction is from MCVP. Let $(g_1, \ldots, g_k)$ denote an instance α of MCVP including inputs, where g_k is the output gate. From α a graph G is constructed as follows:

1. associate nodes g_i' and $g_{i''}$ with each g_i,
2. for each input g_i, if g_i is TRUE (FALSE) add a loop edge to g_i' ($g_{i''}$),
3. for each AND gate g with inputs i and j add edges (g', i''), (i'', j''), (j'', g'), (g'', i'), (i', g''), (g'', j'), and (j', g'') to G,
4. for each OR gate g with inputs i and j add edges (g'', i'), (i', j'), (j', g''), (g', i''), (i'', g'), (g', j''), and (j'', g') to G,
5. add edges $(g_{k'}, g_{k''})$ and $(g_{k''}, g_{k'})$.

It is easy to see that G is cyclically reducible. Set $v = g_k'$ to complete the reduction.
Remarks: The question of whether a graph has a feedback vertex set of size k is NP-complete [Kar72]. Bovet, De Agostino, and Petreshci give an algorithm for finding a minimum feedback set that requires $O(k \log^2 n)$ time and $O(n^4)$ processors on a CREW-PRAM, where k denotes the size of the minimum feedback set. Greenlaw proved that a related problem, the lexicographically first maximal acyclic subgraph problem, is P-complete [Gre90]. Ramachandran proved that finding a minimum weight feedback vertex set in a reducible flow graph with arbitrary weights is P-complete [Ram88]. She also proved the following four problems are NC equivalent: finding a minimum feedback arc set in an unweighted reducible flow graph, finding a minimum weight feedback arc set in a reducible flow graph with unary weights on the arcs, finding a minimum weight feedback vertex set in a reducible flow graph with unary weights on the vertices, and finding a minimum cut in a flow network with unary capacities [Ram88].

A.2.15 Edge Maximal Acyclic Subgraph (EMAS)

Given: A directed graph $G = (V, E)$ with an ordering on the edges and a designated edge e.
Problem: Is e contained in the edge maximal acyclic subgraph?
Reference: [Gre90]
Hint: The *edge maximal acyclic subgraph* is defined to be the subgraph computed by an algorithm that builds up the subgraph by processing edges in order. It adds an edge to the subgraph if its inclusion does not introduce a cycle. The reduction is from NOR CVP. A gadget is designed that replaces each gate in the circuit. A gadget for gate g with inputs i_1, i_2 and outputs o_1, o_2 has three nodes g(top), g(mid), and g(bot). The edges in the gadget are (g(top),i_1(mid)), (g(top),i_2(mid)), (g(mid),i_1(bot)), (g(mid),i_2(bot)), (g(bot),o_2(top)), (g(top),g(mid)), (g(mid),g(bot)), (g(bot),o_1(top)), and (g(bot),g(top)). The first five edges are upwarding pointing edges and are ordered first. The last four edges are ordered within the gadget as listed. TRUE input i to gate g is represented by a node with an edge to g(top). FALSE input i to gate g is represented by two nodes forming a cycle with g(top). The edges

are ordered so that the edge leading into $g(\text{top})$ is not put in the set the algorithm constructs. The edges of the output gate are "grounded." One of the edges leaving $g_{out}(\text{bot})$ is used as the designated edge e.
Remarks: The decision problem "Is the edge maximal subgraph of size k?" is also P-complete [Gre90]. The feedback arc set problem [Kar72] is an equivalent formulation of the maximum acyclic subgraph problem. Approximation algorithms for this problem are important because there are very few classes of graphs for which the problem is known to be in P [BS90]. Greenlaw proves decision problems based on several other natural approximation algorithms are P-complete [Gre90]. He also gives two approximation algorithms that are in NC. Berger has an NC approximation algorithm that, assuming the input graph does not contain two-cycles, generates a subgraph containing more than half the arcs [Ber91]. Ramachandran proved that finding a minimum weight feedback arc set in a reducible flow graph with arbitrary weights on the arcs in P-complete [Ram88].

A.2.16 General Graph Closure (GGC)

Given: An undirected graph $G = (V, E)$, a subset $E' \subseteq V \times V$ with $E' \cap E = \emptyset$, and a designated edge $e = (u, v) \in E'$.
Problem: Is e in the *general closure* $\mathcal{G}(G, E')$ of G? That is, the graph obtained from G by repeatedly joining non-adjacent pairs of vertices u and v whose degree sum is at least $|V|$ and such that $(u, v) \in E'$. The edges in E' are called *admissible edges.*
Reference: [Khu89]
Hint: An $O(n^3)$ algorithm solving the problem is given in [Khu89]. The reduction is from MCVP, Problem A.1.3. The idea is to replace gates in the circuit by gadgets. We will describe only the gadgets for inputs and AND gates. The gadgets for OR gates are similar. A TRUE (FALSE) value is associated with the output vertex of a gadget if its degree has increased by the addition of an admissible edge (remained the same). The gadget is constructed so that "values" do not propagate back up through the circuit. N will denote the total number of vertices contained in all the gadgets. An additional N vertices are added and connected so as to double the degrees of the vertices in the construction of the graph containing gadgets. The total degree of the graph constructed is $2N$. A TRUE input is represented by two vertices with an admissible edge between them and the degree of each vertex is N. A FALSE input is represented similarly except on the output side the vertex has degree $N - 1$. The gadget representing an AND gate α_k consists of thirteen vertices. We describe the upper left part, which consists of five vertices, first. We'll call the vertices 1, 2, 3, 4, and 5. Vertex 1 has a connection from one of α_k's inputs. Vertices 1 and 5 have degree $N - 1$ and vertices 2, 3, and 4 have degree N. The admissible edges are $(1,2), (1,3), (1,4), (2,5), (3,5)$, and $(4,5)$. The upper right part of the gadget is similar with vertices 6-10 playing the roles of 1-5. Vertices 5 and 10 are connected to vertex 11 via admissible edges. Vertex 11 has degree $N - 2$. Vertex 11 is connected to the outputs of the gadget. These are vertices 12 and 13. They both are of degree N. The gadgets are connected in the obvious manner. The circuit evaluates to TRUE if and only if the admissible edge $(11, 13)$ of the gadget corresponding to the output gate is added to G.
Remarks: The complexity of the general graph closure problem in which $E' = V \times V - E$ is open, see Problem B.2.3.

A.3 Search Problems

A.3.1 Lexicographically First Maximal Path (LFMP)

Given: A graph $G = (V, E)$ with a numbering on the vertices and two designated vertices s and t.
Problem: Is vertex t on the *lexicographically first maximal path* in G beginning at s? A *maximal path* is a path that cannot be extended because any attempt at extending it will result in an encounter with a node that is already on the path. The lexicographically first maximal path is the maximal path that would appear first in the "alphabetical" listing of all paths from s, where the alphabetizing is done with respect to the vertex numbers.
Reference: [AM87b]
Hint: The reduction is from a version of CVP consisting of NOT and OR gates. The key idea is the construction of a subgraph called a *latch*. A latch consists of six nodes connected in a rectangular fashion. The latches are hooked together and labeled in a clever manner. A latch that has been traversed (not traversed) in the construction of the lexicographically first maximal path indicates a TRUE (FALSE) value for the corresponding gate. Vertex t is a special vertex in a latch corresponding to the output gate.
Remarks: LFMP remains P-complete when restricted to planar graphs with maximum degree three. If the maximum degree of any vertex in G is at most Δ, then there is an algorithm that can find a maximal path in $O(\Delta \log^3 n)$ time using n^2 processors [AM87b]. There is also an NC algorithm for finding a maximal path in planar graphs [AM87b]. The complexity of the general problem of finding a maximal path is open [AM87b], although known to be in RNC [And87].

A.3.2 Lexicographically First Depth First Search Ordering (LFDFS)

Given: A graph $G = (V, E)$ with fixed ordered adjacency lists, and two designated vertices u and v.
Problem: Is vertex u visited before vertex v in the depth first search of G induced by the order of the adjacency lists?
Reference: [Rei85]
Hint: Follows easily from Problem A.3.1, since the leftmost path in the lexicographically first depth first search tree is the lexicographically first maximal path [And85]. Reif [Rei85] gives a direct reduction from NOR CVP to DFS, taking advantage of the fixed order by which the adjacency lists are examined. We present the directed case from which the undirected case is easily derived. Without loss of generality, assume gates are numbered in topological order. The gadget described below replaces NOR gate i having inputs i_1 and i_2, and outputs to gates j_1 and j_2. The gadget has 8 vertices $enter(i), in(i, i_1), in(i, i_2), s(i), out(i, 1), out(i, 2), t(i)$, and $exit(i)$. Let the triple (x, y, z) denote a directed edge (x, y) with y appearing z^{th} on x's adjacency list. The gadget has triples $(enter(i), in(i, i_1), 1)$, $(in(i, i_1), in(i, i_2), 2)$, $(in(i, i_2), s(i), 2)$, $(s(i), out(i, 1), 1)$, $(out(i, 1), s(i), 1)$, $(out(i, 1), in(j_1, i), 2)$, $(out(i, 2), out(i, 1), 1)$, $(out(i, 2), in(j_2, i), 2)$, $(t(i), out(i, 2), 1)$, $(enter(i), t(i), 2)$, $(t(i), exit(i), 2)$, $(s(i), exit(i), 2)$, and $(exit(i), enter(i + 1), 1)$. Additionally, $(in(j_1, i), out(i, 2), 1)$ and $(in(j_2, i), t(i), 1)$ are triples connected to the gadget. TRUE inputs are "chained" together. The lexicographic

DFS of the graph constructed visits vertex $s(n)$, where n corresponds to the output gate, before (after) $t(n)$ if and only if the circuit evaluates to TRUE (FALSE).

Remarks: The directed case can be easily reduced to the undirected case. The reduction is dependent on the adjacency lists fixing the order in which the adjacent nodes are examined. The problem remains open if this constraint is relaxed. For example, the problem remains open for graphs presented with all adjacency lists sorted in order of increasing vertex number. The problem remains P-complete if the input is specified by a fixed vertex numbering [And85, Gre88b]. Anderson showed that computing just the first branch of the lexicographically first DFS tree, called the lexicographically first maximal path, is P-complete [And85] (see Problem A.3.1). Computing the LFDFS tree in planar graphs is P-complete as well [And85]. In RNC, it is possible to find *some* depth-first vertex numbering and the depth-first spanning tree corresponding to it; see Problem B.5.1 [AA88, AAK90]. Computing a depth-first vertex numbering for planar graphs is in NC [Smi86, HY87]. Computing the lexicographically first depth-first numbering for DAGs is in NC [Grear, dlTK88, dlTK89]. Determining whether a directed spanning tree of a general graph has a valid DFS numbering is in NC [SV85].

A.3.3 Breadth-Depth Search (BDS)

Given: A graph $G = (V, E)$ with a numbering on the vertices and two designated vertices u and v.

Problem: Is vertex u visited before vertex v in the *breadth-depth first search* [HS84] of G induced by the vertex numbering? A breadth-depth first search starts at a node s and visits all children of s pushing them on a stack as the search proceeds. After all of s's children have been visited, the search continues with the node on the top of the stack playing the role of s.

Reference: [Gre88b, Grear]

Hint: The proof sketched below is due to Anderson [And88]. Reduce LFDFS to BDS. Insert a new vertex between every pair of connected nodes in the original graph. Suppose in the original graph corresponding to the circuit that vertex u_i has children $v_1, \ldots, v_k$ and that these vertices are visited in this order by the greedy DFS algorithm. Let $c_1, \ldots, c_k$ be the nodes inserted by the reduction between (u_i and v_1), ..., (u_i and v_k) respectively. The c_i's are assigned numbers so that in increasing order they are listed as $c_k, \ldots, c_1$. The vertex numbers assigned to these nodes are specified to "reverse" the breadth-depth search in the new levels. In this way the DFS order of the original graph can be maintained. Vertex u is visited before vertex v in an instance of LFDFS if and only if the vertex corresponding to u is visited before the vertex corresponding to v in the constructed instance of BDS.

A.3.4 Stack Breadth First Search (SBFS)

Given: A graph $G = (V, E)$ with a numbering on the vertices and two designated vertices u and v.

Problem: Is vertex u visited before vertex v in the *stack breadth first search* of G induced by the vertex numbering? A stack breadth first search is a breadth first search that is implemented on a stack. The nodes most recently visited on a new level are searched from

first at the next level.
Reference: [Gre88b, Grear]
Hint: The reduction is from SAM2CVP to SBFS. Sort the inputs nodes and assign FALSE inputs lower numbers than TRUE inputs. In increasing order let $f_1, \ldots, f_k, t_1, \ldots, t_m$ denote the ordering induced by this numbering. A new vertex e_1 is introduced and given a number between f_k and t_1. For each gate a vertex is introduced and its connections in the circuit are maintained in the graph being constructed. A start vertex s is added and connected to all inputs. Additionally, a new chain of vertices starting from s is added. The vertices in this chain are $s, e_1, e_2, \ldots, e_D$, where D denotes the depth of the circuit. The search order specified by stack breadth first search is such that vertex e_l for l odd (even) corresponding to an OR (AND) level in the instance of SAM2CVP is visited before vertex v if and only if the gate corresponding to vertex v evaluates to FALSE (TRUE) in the circuit.
Remarks: The *lexicographic breadth first search problem*, which has a natural implementation on a queue, is defined as follows: given a graph G with fixed ordered adjacency lists is vertex u visited before vertex v in the breadth first search of G induced by the order of the adjacency lists. This problem is in *NC* [Gre88b, dlTK88, dlTK89].

A.3.5 Alternating Breadth First Search (ABFS)

Given: A graph $G = (V, E)$ with E partitioned into two sets M and U, a designated vertex v, and a designated start vertex s.
Problem: Does vertex v get visited along an edge from the set M during an *alternating breadth first search* of G? An *alternating breadth first search*, which has applications in some matching algorithms, is a breadth first search in which only edges in the set U (M) can be followed in going from even (odd) to odd (even) levels.
Reference: [And85, And88]
Hint: Anderson's proof was from a version of CVP composed of OR and NOT gates. The reduction we present is from NAND CVP. Let $t_1, \ldots, t_k$ denote TRUE inputs, $f_1, \ldots, f_l$ denote FALSE inputs, and $g_1, \ldots, g_m$ denote NAND gates. A new vertex s is created from where the search will originate. For each t_i, $1 \leq i \leq k$, two new vertices t_i' and t_i'' are introduced, for each f_i, $1 \leq i \leq l$, a new vertex f_i' is introduced, and for each g_i, $1 \leq i \leq m$, a vertex v_i is introduced. f_i' is connected to s by an edge in U and to v_j, where j is such that f_i' was input to gate g_j, by an edge in M. t_i'' is connected to s by an edge in U, t_i'' is connected to t_i' by an edge in M, and t_i' is connected to v_j, where j is such that t_i' was input to gate g_j. If gate g_i has outputs to gates g_j and g_h, then (v_i, v_j) and (v_i, v_h) are edges in M. For each of these gates receiving inputs from g_i there are two additional vertices. For g_j they are called v_{ij} and v_{ij}'. (v_i, v_{ij}) is an edge in U, (v_{ij}, v_{ij}') is in M, and (v_{ij}', v_j) is in U. For gate g_h similar vertices and edges are added. The circuit evaluates to TRUE (FALSE) if and only if the vertex corresponding to the output gate of the circuit is visited along an edge in M (U).
Remarks: The matching constructed by the search is not necessarily maximal. The problem of finding a maximum matching is in *RNC* [MVV87]. The problem of finding a perfect matching is also in *RNC* [KUW85, MVV87].

A.4 Combinatorial Optimization and Flow Problems

A.4.1 Linear Inequalities (LI)

Given: An integer $n \times d$ matrix A and an integer $n \times 1$ vector b.
Problem: Is there a rational $d \times 1$ vector $x > 0$ such that $Ax \leq b$? (It is not required to find such an x.)
Reference: [Coo82, Val82a, Kha79]
Hint: LI is in P by [Kha79]. The following reduction of CVP to LI is due to [Coo82].

1. If input x_i is TRUE (FALSE) it is represented by the equation $x_i = 1$ ($x_i = 0$).
2. A NOT gate with input u and output w, computing $w \leftarrow \neg u$ is represented by the inequalities $w = 1 - u$ and $0 \leq w \leq 1$.
3. An AND gate with inputs u, v computing $w \leftarrow u \wedge v$ is represented by the inequalities $0 \leq w \leq 1$, $w \leq u$, $w \leq v$, and $u + v - 1 \leq w$.
4. An OR gate is represented by the inequalities $0 \leq w \leq 1$, $u \leq w$, $v \leq w$, and $w \leq u + v$.

Note for any gate, if the inputs are 0 or 1 the output will be 0 or 1. To determine the output z of the circuit, add the inequalities required to force $z = 1$. If the system has a solution then the output is TRUE and otherwise the output is FALSE.

A.4.2 Linear Equalities (LE)

Given: An integer $n \times d$ matrix A and an integer $n \times 1$ vector b.
Problem: Is there a rational $d \times 1$ vector $x > 0$ such that $Ax = b$?
Reference: [Coo82, Val82a, Kha79]
Hint: LE is NC^1 reducible to LI since $Ax \leq b$ and $-Ax \leq -b$ if and only if $Ax = b$. Thus LE is in P. For completeness an instance of LI can be reduced to LE as follows: for each inequality in LI there is a corresponding equality in LE with an additional "slack" variable that is used to make the inequality into an equality.
Remarks: If LE is restricted so the coefficients of A and b are either $-1, 0$, or 1 then LE is still P-complete. This follows from the reduction given by Itai [Ita78]. The restricted version of LE is denoted [-1,1] LE.

A.4.3 Linear Programming (*) (LP)

Given: An integer $n \times d$ matrix A, an integer $n \times 1$ vector b, and an integer $1 \times d$ vector c.
Problem: Find a rational $d \times 1$ vector x such that $Ax \leq b$ and cx is maximized.
Reference: [DLR79, Kha79, DR80, Val82a]
Hint: LP is not in P, but is in FP by [Kha79]. Reduce LI to LP by picking any cost vector c, say $c = \vec{0}$, and checking whether the resulting linear program is feasible.
Remarks: The original reduction in [DLR79] is from HORN, Problem A.6.2, to LP. In [DR80], LP and LI are shown to be log space equivalent by reducing LP to LI using rational binary search [Pap78, Rei78] to find the value of the maximum and an x that

yields it. However, it is not clear how to perform this reduction in NC^1. Since LP and LI are complete via NC^1 reductions though, there must be a NC^1 reduction between the two problems. Although we know that LP and LI are NC^1 equivalent, the NC^1 reduction between them is not an obvious one. It is also P-hard to approximate cx to within any constant fraction, even given a feasible solution x'. (Anne Condon, personal communication, April 1991; reduction is from Problem A.8.2.)

A.4.4 Maximum Flow (MaxFlow)

Given: A directed graph G with each edge labeled with a capacity $c_i \geq 0$ two distinguished vertices, source s and sink t, and a value f.
Problem: Is there a feasible flow of value k, i.e. is the value of the maximum flow into the sink $\geq f$?
Reference: [GSS82, LW87]
Hint: The first P-completeness proof for a decision problem derived from maximum flow was for the problem of determining whether the flow is odd or even [GSS82]. We give this reduction. (The proof given in [LW87] for the more natural threshold problem stated above is similar.) The reduction is from AM2CVP (Problem A.1.4) to MaxFlow. Gates v_i and connections e_{ij} of α are associated with nodes v_i' and edges e_{ij}' of G. G has additional nodes s, t and an overflow edge for each v_i'. Each edge e_{ij} of α has a capacity and a flow associated with it. This capacity is 2^i, and the flow is 2^i if gate v_i is TRUE and 0 otherwise. A node v_i' with inputs v_j' and v_k' has a maximum possible inflow of $2^j + 2^k$, and outflow to other gates of $d2^i$ (d is the outdegree of v_i). The remaining flow is absorbed by the overflow edge from v_i' with capacity $2^j + 2^k - d2^i$. This overflow edge is directed toward t in case v_i is an AND gate and toward s in case v_i is an OR gate. Thus the nodes must be topologically ordered with the output first and the inputs last, and the output gate must be an OR gate. Note that all edge capacities are even except the one from v_0 to t. Thus the maximum flow for G is odd if and only if α outputs TRUE.
Remarks: This reduction produces exponential edge capacities in G. In a network with edge capacities expressed in unary, computing the magnitude of the maximum flow is in RNC^2 [Fea84] and a method for finding the flow in RNC is also known [KUW86]. If the network is restricted to being acyclic MaxFlow remains P-complete [Ram87]. Flows in planar networks can be computed in NC [JV82]. Two commodity flow (2CF) is defined like MaxFlow except there are two sources and two sinks, and there are two separate flows functions for the commodities. Since MaxFlow is a special case of 2CF and 2CF is in P by results of Itai [Ita78] and Khachian [Kha79], it follows that 2CF is P-complete. Itai defined several other variants of 2CF that are also P-complete. They are defined below. (l, u)-2CF is a case of 2CF in which there are lower and upper bounds on the capacity of each edge. Selective (l, u)-2CF is defined to be a 2CF with lower and upper bounds on the sum of the two flows on each edge. Stein and Wein [SW91] show that, althought there is an RNC algorithm to approximate maximum flow, approximating the minimum cost maximum flow is P-complete.

A.4.5 Homologous Flow (HF)

Given: A directed graph G with each edge (v, w) labeled with a lower and upper bound on flow capacity $l(v, w), u(v, w) \geq 0$ and two distinguished vertices, source s and sink t.
Problem: Is there a feasible flow in the network? A *feasible flow* is one in which the flow assigned to each arc falls within the lower and upper bounds for the arc. A *homologous flow* is a flow in which pairs of edges are required to have the same flow.
Reference: [Ita78]
Hint: It follows that HF is in P by the results of Itai [Ita78] and Khachian [Kha79]. We describe the reduction given by Itai [Ita78] and note that it is a log space reduction. The reduction is from [−1,1] LE. Let $\sum_{j=1}^{m} a_{ij}x_j = b_i$, for $i = 1, \ldots, n$ be an instance of [−1,1] LE. For $\sigma \in \{-1, 0, 1\}$, let $J_\sigma^i = \{j \mid a_{ij} = \sigma\}$. Another formulation of the original equations is $\sum_{j \in J_1^i} x_j - \sum_{j \in J_{-1}^i} x_j = b_i$, for $i = 1, \ldots, n$. There are n sections in the flow network constructed and each one has $m + 5$ vertices $\{v_1^i, \ldots, v_m^i, y^i, z^i, J_{-1}^i, J_0^i, J_1^i\}$. For $\sigma = -1, 0, 1$, if $j \in J_0^i$ then add (v_j^i, J_σ^i) as a *nonrestricted edge*, one with lower bound 0 and upper bound ∞, to the network. Add (J_1^i, z^i) with lower and upper capacities equal to b_i. (J_1^i, y^i) and (J_{-1}^i, y^i) are homologous nonrestricted edges. (J_1^i, z^i) and (J_1^i, y^i) are nonrestricted edges. An additional node z_0 is added as a source and z_n is the sink. For each j, $(z^0, v_j^1), (z^1, v_j^2), \ldots, (z^{n-1}, v_j^n)$ are pairwise nonrestricted homologous edges. Let f denote the flow, given a solution x to the equations a feasible flow is $x_j = f(z_0, v_j^1) = \cdots = f(z^{n-1}, v_j^{n-1})$ and given a feasible flow it is easy to construct solution x.

A.4.6 Lexicographically First Blocking Flow (LFBF)

Given: A directed acyclic graph $G = (V, E)$ represented by fixed ordered adjacency lists with each edge labeled with a capacity $c_i \geq 0$ and two distinguished vertices, source s and sink t.
Problem: Is the value of the lexicographically first *blocking flow* odd? A blocking flow is a flow in which every path from s to t has a *saturated edge* — an edge whose flow is equal to its capacity. The lexicographically first blocking flow is the flow resulting from the standard sequential depth first search blocking flow algorithm.
Reference: [AM87a]
Hint: The reduction given in Problem A.4.4 can be easily modified to show this problem is P-complete.
Remarks: The problem of finding the lexicographically first blocking flow in a *3 layered network* is also P-complete. A 3 layered network is one in which all source to sink paths have length 3. Cheriyan and Maheshwari also give an *RNC* algorithm for finding a blocking flow in a 3 layered network [CM89].

A.4.7 First Fit Decreasing Bin Packing (FFDBP)

Given: A list of n items $v_1, \ldots, v_n$, where each v_i is rational number between 0 and 1, and two distinguished indices i and b.
Problem: Is the i^{th} item packed into the b^{th} bin by the first fit decreasing bin packing heuristic?

Reference: [AMW89]
Hint: Reduce AM2CVP (Problem A.1.4) to FFDBP. Without loss of generality, we can assume the gates $\beta_1, \ldots, \beta_n$ are numbered in topological order. The reduction transforms the sequence $\beta_1, \ldots, \beta_n$ into a list of items and bins. Let $\delta_i = 1 - i/(n+1)$ and $\epsilon = 1/(5(n+1))$, and T_i (F_i) denote any item of size δ_i ($\delta_i - 2\epsilon$). We describe how to construct the list of items and bins. For AND gate β_i with outputs to gates β_j and β_k construct bins of size $\delta_i, 2\delta_i - 4\epsilon, \delta_i + \delta_j - 3\epsilon$, and $\delta_i + \delta_k - 4\epsilon$ and items of size $\delta_i, \delta_i, \delta_i - 2\epsilon, \delta_i - 2\epsilon, \delta_i - 3\epsilon$, and $\delta_i - 4\epsilon$. For an OR gate β_i with outputs to gates β_j and β_k construct bins of size $2\delta_i - 4\epsilon, \delta_i, \delta_i + \delta_j - 3\epsilon$, and $\delta_i + \delta_k - 4\epsilon$ and the same items as for the AND gate. The output gate β_n is treated specially and has bins of size δ_n and δ_n, and items of size $\delta_n, \delta_n, \delta_n - 2\epsilon$, and $\delta_n - 2\epsilon$. For gates receiving a constant circuit input, a T_i (F_i) is removed if the gate receives a FALSE (TRUE) input. The lists of bins are concatenated in the order of their corresponding gate numbers and similarly for the items. To get unit size bins let $u_1, \ldots, u_q$ be the non-increasing list of item sizes and let $b_1, \ldots, b_r$ be the list of variable bin sizes as constructed above. Let $B = \max_i b_i$ and $C = (2r+1)B$. For $1 \leq i \leq 2r$, set $v_i = C - iB - b$, if $i \leq r$ and $C - ib$, otherwise. Packing these $2r$ items into r bins of size C has the affect of leaving b_i space in the i^{th} bin. By concatenating the "u" and "v" item lists and normalizing the bin sizes, a first fit decreasing bin packing of the items will place the item corresponding to the second T_n in β_n's list into the last bin if and only if the circuit evaluates to TRUE.
Remarks: The problem remains P-complete even if unary representations are used for the numbers involved. This is one of the first such problem where large numbers do not appear to be required for P-completeness (in contrast see MaxFlow, Problem A.4.4). The problem of determining if I is the packing produced by the best fit decreasing algorithm is also P-complete [AMW89]. In [AMW89] there is an *NC* algorithm that produces a packing within 11/9 of optimal. This is the same performance as for first fit decreasing.

A.4.8 General List Scheduling (GLS)

Given: An ordered list of n jobs $\{J_1, \ldots, J_n\}$, a positive integer execution time $T(J_i)$ for each job, and a non-preemptive schedule L. The jobs are to be scheduled on two identical processors.
Problem: Is the *final offset* produced by the list scheduling algorithm non-zero? The final offset is the difference in the total execution time of the two processors.
Reference: [HM87]
Hint: Reduce NOR CVP to GLS. Without loss of generality, assume the gates in the instance of NOR CVP are numbered in reverse topological order. The input wires to gate i are numbered 4^{2i} and 4^{2i+1}. The output wire of gate 1, the overall circuit output, is labeled 4. Let V_i be the sum of the labels on the output wires of gate i. For gate i, 17 jobs are introduced with the following execution times — 1 job at $2 \cdot 4^{2i+1}$, 14 jobs at $4^{2i}/2$, and 2 jobs at $(4^{2i} + V_i)/2$. The initial job has execution time equal to the sum of the labels of all TRUE input wires. The remaining jobs are listed in descending order of gate number. The final offset will be 4 (0) if and only if the output gate i is TRUE (FALSE).
Remarks: The problem is in *NC* if the job times are small, that is $n^{O(1)}$. *NC* algorithms for scheduling problems with either intree or outtree precedence constraints are known [HM86, HM87].

A.5 Local Optimality

A.5.1 MAXFLIP Verification (MAXFLIPV)

Given: An encoding of a Boolean circuit α constructed of AND, OR, and NOT gates, plus inputs $x = x_1, \ldots, x_n$. The circuit's m output values $y = y_1, \ldots, y_m$.
Problem: Is the circuit's output a local maximum among the *neighbors of* x when y is viewed as a binary number? The neighbors of x are vectors of length n whose Hamming distance differs from x by 1. That is, they can be obtained from x by flipping one bit.
Reference: [JPY88]
Hint: The problem is easily seen to be in P. The reduction is from the MCVP, Problem A.1.3. Let α denote an instance of MCVP with input $x_1, \ldots, x_n$. Construct an instance of MAXFLIP as follows. The new circuit is the same as α except for a modification to the input. Add a "latch" input that is AND'd with each of α's inputs before they are fed into later gates of α. Set the latch input to value 0. The output of the circuit constructed will be 0. The input $x_1, \ldots, x_n, 0$ will be locally optimal if and only if the output is 0 when the latch input is 1. This is true if and only if the output of α on its input is 0.
Remarks: The complementary problem, called the *FLIP verification problem*, in which the output is minimized is also P-complete [JPY88]. The general problems MAXFLIP and FLIP are *PLS*-complete [JPY88]. PLS stands for polynomially local search.

A.5.2 Local Optimality Kernighan-Lin Verification (LOKLV)

Given: A graph $G = (V, E)$ with weights $w(e)$ on the edges and a partition of V into two equal size subsets A and B.
Problem: Is the *cost of the partition*, $c(A, B)$, a local optimum among the *neighbors of the partition*? The cost of the partition is defined to be the sum of the costs of all edges going between the sets A and B. We follow the presentation in [JPY88] to define the neighbors. A *swap* of partition (A, B) is a partition (C, D) such that (C, D) is obtained from (A, B) by swapping one element of A with an element of B. The swap (C, D) is a *greedy* swap if $c(C, D) - c(A, B)$ is minimized over all swaps of (A, B). If (C, D) is the lexicographically smallest over all greedy swaps, then (C, D) is said to be the *lexicographically greedy swap of* (A, B). A sequence of partitions (A_i, B_i), each obtained by a swap from the preceding partition, is *monotonic* if the differences $A_i - A_0$ and $B_i - B_0$ are monotonically increasing. A partition (C, D) is a neighbor of (A, B) if it occurs in the unique maximal monotonic sequence of lexicographic greedy swaps starting with (A, B).
Reference: [JPY88]
Hint: The reduction is from MAXFLIPV, Problem A.5.1.
Remarks: A problem called the *weak local optimum for Kernighan-Lin verification problem* in which the *neighborhoods* are larger is also P-complete [JPY88]. The general versions of these problems, *local optimality Kernighan-Lin* and *weak local optimality Kernighan-Lin* are both *PLS*-complete [JPY88].

A.5.3 Unweighted, Not-All-Equal Clauses, 3SAT/FLIP (U3NSATFLIP)

Given: A Boolean formula F in CNF with 3 literals per clause and a truth assignment s.

Each clause has a weight of 1. The clauses are *not-all-equals* clauses with positive literals. A truth assignment "satisfies" a clause C under the not-all-equals criterion if it is such that C has at least one TRUE and one FALSE literal.
Problem: Is the assignment s the maximum *cost assignment* of F over all *neighbors* of s? The cost of the assignment is the sum of the weights of the clauses it satisfies. The neighbors of s are assignments that differ from s in one bit position.
Reference: [PSY90, SY]
Hint: The reduction is from NOR CVP.
Remarks: The weighted version of the problem is *PLS*-complete [PSY90].

A.5.4 Unweighted MAXCUT/SWAP (UMS)

Given: A graph $G = (V, E)$ with weights of size 1 on the edges and a subset $S \subseteq V$.
Problem: Is S the maximum cost subset of nodes, where the cost is the sum of the weights of the edges leaving nodes in S, over all *neighbors* of S? A neighbor of S is a set of size $|S|$ whose symmetric difference with S contains one node.
Reference: [PSY90, SY]
Hint: The reduction is from U3NSATFLIP, Problem A.5.3.
Remarks: The weighted version of the problem is *PLS*-complete [PSY90].

A.5.5 Unweighted 2SAT/FLIP (U2SATFLIP)

Given: A Boolean formula F in CNF with 2 literals per clause and a truth assignment s. Each clause has a weight of 1.
Problem: Is the assignment s the maximum *cost assignment* of F over all *neighbors* of s? The cost of the assignment is the sum of the weights of the clauses it satisfies. The neighbors of s are assignments that differ from s in one bit position.
Reference: [PSY90, SY]
Hint: The reduction is from UMS, Problem A.5.4.
Remarks: The weighted version of the problem is *PLS*-complete [PSY90].

A.5.6 Unweighted SWAP (USWAP)

Given: A graph $G = (V, E)$ with $2n$ nodes and a set S of n nodes.
Problem: Is S the minimum cost set of n nodes, where the cost is the sum of the weights of the edges leaving S, over all *neighbors* of S? A neighbor of S is a set of n nodes whose symmetric difference with S consists of two nodes.
Reference: [PSY90, SY]
Hint: The reduction is from UMS, Problem A.5.4.
Remarks: The weighted version of the problem is *PLS*-complete [PSY90].

A.5.7 Unweighted STABLE NET (USN)

Given: A graph $G = (V, E)$ whose edges all have weights of -1 and an assignment B of labels $x_i \in \{-1, +1\}$ to each node i.

Problem: Does the assignment B yield maximum cost; $\sum_{i,j} x_i x_j$ over all i, j; over all *neighbors* of B? A neighbor of B is an assignment of labels that differs from B on only one node.
Reference: [PSY90, SY]
Hint: The reduction is from UMS, Problem A.5.4.
Remarks: The weighted version of the problem is *PLS*-complete [PSY90].

A.6 Logic

A.6.1 Unit Resolution (UNIT)

Given: A Boolean formula F in conjunctive normal form.
Problem: Can the empty clause $\Box$ be deduced from F by *unit* resolution? A unit is a clause with only one term. For example, the unit resolvent of $F = A \vee B_1 \vee \cdots \vee B_m$ and the unit $G = \neg A$ is $B_1 \vee \cdots \vee B_m$.
Reference: [JL76]
Hint: Jones and Laaser provide a poly-time algorithm for unit resolution [JL76]. To show B follows from the assumption $A_1 \wedge \cdots \wedge A_m$, negate B, add it to the set of clauses and derive the empty clause. Reduce CVP to UNIT as described below. A gate in the circuit $v_k \leftarrow v_i \wedge v_j$ is represented by the clauses of $v_k \Leftrightarrow v_i \wedge v_j$, that is, $(\neg v_k \vee v_i) \wedge (\neg v_k \vee v_j) \wedge (v_k \vee \neg v_i \vee \neg v_j)$. Similarly, $v_k \leftarrow v_i \vee v_j$ is represented by the clauses of $v_k \Leftrightarrow v_i \vee v_j$ and $v_k \leftarrow \neg v_i$ is represented by the clauses of $v_k \Leftrightarrow \neg v_i$.
Remarks: Under the appropriate definitions, it is known that approximating this problem is also P-complete [SS89].

A.6.2 Horn Unit Resolution (HORN)

Given: A Horn formula F, that is, a conjunctive normal form (CNF) formula with each clause a disjunction of literals having at most one positive literal per clause.
Problem: Can the empty clause $\Box$ be deduced from F by *unit* resolution?
Reference: [DLR79, JL76]
Hint: Reduce an arbitrary Turing machine to a CNF formula as in [Coo71b]. All of the clauses are Horn clauses. Most clauses are of the form $\neg P^a_{i-1,t} \vee \neg P^b_{i,t} \vee \neg P^c_{i+1,t} \vee P^{f(a,b,c)}_{i,t+1}$, where P_i, t^a is true if at time t tape cell i contains symbol a (or symbol (a, s) if the tape head is over the cell and the Turing machine is in state s). The function $f(a, b, c)$ depends on the Turing machine. For cells $i-1, i, i+1$ containing symbols a, b, c the value of cell i at the next time step is $f(a, b, c)$. Alternatively, reduce CVP to HORN as for UNIT. The clauses for $v_k \leftarrow v_i \wedge v_j$ and for $v_k \leftarrow \neg v_i$ are already in Horn form. For $v_k \leftarrow v_i \vee v_j$ the clauses of $v_k \Leftrightarrow v_i \vee v_j$ are not in Horn form, but replacing the OR by an AND gate and associated NOT gates using DeMorgan's laws results in a set of Horn clauses.

A.6.3 Propositional Horn Clause Satisfiability (PHCS)

Given: A set S of Horn clauses in the propositional calculus.
Problem: Is S satisfiable?

Reference: [Pla84, Kas86]
Hint: The reduction is straightforward from the alternating graph accessibility problem, Problem A.2.3.
Remarks: The problem remains P-complete when there are at most 3 literals per clause [Pla84]. Plaisted has shown that two problems involving proofs of restricted depth are also P-complete. They are the two literal Horn clause unique matching problem and the three literal Horn clause problem [Pla84].

A.6.4 Relaxed Consistent Labeling (RCL)

Given: A relaxed consistent labeling problem G consisting of a set of variables $V = \{v_1, \ldots, v_n\}$ and a set of labels $L = \{L_1, \ldots, L_n\}$, where L_i consists of the possible labels for v_i. A binary predicate P, where $P_{ij}(x, y) = 1$ if and only if the assignment of label x to v_i is compatible with the assignment of label y to v_j. A designated variable P_0 and a designated label f.
Problem: Is there a valid assignment of the label f to P_0 in G?
Reference: [Kas86, GHR91]
Hint: The original reduction is from the propositional Horn clause satisfiability problem, Problem A.6.3 [Kas86]. The reduction we sketch is from NAND CVP. A variable is introduced for each circuit input. The variable must have label 1 (0) if the circuit input is TRUE (FALSE). We view each NAND gate as being represented by three variables. Consider NAND gate k with inputs i and j, and outputs s and t. The variables for k will be denoted k, L_k, and R_k. The possible labels for a NAND gate (variable k) are 0, 1, T, T', and F. T and T' are used to denote TRUE values, and F is used to denote a FALSE value. The possible labels for variables L_k and R_k are 0 and 1. The constraints for variable k with its inputs are as follows: $P_{ik}(0, 1) = 1$, $P_{ik}(1, T) = 1$, $P_{ik}(0, T') = 1$, $P_{ik}(1, F) = 1$, $P_{jk}(0, 1) = 1$, $P_{jk}(0, T) = 1$, $P_{jk}(1, T') = 1$, and $P_{jk}(1, F) = 1$. The constraints on L_k and R_k are the same. When k has any label l from $\{1, T, T'\}$, then $P_{kL_k}(l, 1) = 1$. When k has a label l from $\{0, F\}$, then $P_{kL_k}(l, 0) = 1$. All other possible labelings are not allowed. The constraints involving NAND gate s (t) use L_k (R_k). Notice since inputs have only one possible label, they must be assigned this label. The remainder of the labeling is done so that the circuit gets evaluated in a topological manner. Let o denoted the number of the output gate of the NAND circuit instance. There is a valid assignment of label 1 to node L_o if and only if the output of gate o is TRUE.
Remarks: The general consistent labeling problem is NP-complete.

A.6.5 Generability (GEN)

Given: A finite set W, a binary operation $\bullet$ on W (presented as a table), a subset $V \subseteq W$, and $w \in W$.
Problem: Is w contained in the smallest subset of W that contains V and is closed under the operation $\bullet$?
Reference: [JL76]
Hint: The reduction is from unit resolution, Problem A.6.1. Define $a \bullet b$ to be the unit resolution of clauses a and b. Let W be all the subclauses of a formula F in an instance of

UNIT, and V be all of its clauses. Let w be the empty clause.
Remarks: If $\bullet$ is associative, GEN is complete for NSPACE($\log n$). The problem remains in P even with more than one operation. Under the appropriate definitions, it is known that approximating this problem is also P-complete [SS89].

A.6.6 Path Systems (PATH)

Given: A path system $P = (X, R, S, T)$ where $S \subseteq X$, $T \subseteq X$, and $R \subseteq X \times X \times X$.
Problem: Is there an *admissible* node in S? A node x is admissible if and only if $x \in T$, or there exists admissible $y, z \in X$ such that $(x, y, z) \in R$.
Reference: [Coo74, JL76]
Hint: Reduce GEN to PATH by defining $(x, y, z) \in R$ if and only if $x = y \bullet z$.
Remarks: This is the first problem shown to be log space complete for P. The original proof by Cook does a direct simulation of a Turing machine [Coo74]. Under the appropriate definitions, it is known that approximating this problem is also P-complete [SS89].

A.6.7 Unification (UNIF)

Given: Two symbolic terms s and t. Each term is composed of variables and function symbols. A *substitution* for x in a term u is the replacement of all occurrences of a variable x in u by another term v.
Problem: Is there a series of substitutions σ that unify s and t? That is, gives $\sigma(s) = \sigma(t)$. The two terms are called *unifiable* if such a σ exists.
Reference: [DKM84, Yas84, DKS88]
Hint: The reduction given in [DKM84] is from MCVP. The reductions given in [DKS88] are from NAND CVP.
Remarks: *Unrestricted* unification is also P-complete [DKM84]. Unrestricted unification is where we allow substitutions to map variables to infinite terms. It is convenient to represent terms as labeled directed acyclic graphs. A term is *linear* if no variable appears more than once in the term. The following two restricted versions of unification are also both P-complete: (a) both terms are linear, are represented by trees, and have all function symbols with arity less than or equal to two; (b) both terms are represented by trees, no variable appears in both terms, each variable appears at most twice in some term, and all function symbols have arity less than or equal to two [DKS88]. A restricted problem called *term matching* can be solved on a CREW-PRAM in randomized time $O(\log^2 n)$ using $M(n)$ processors, where $M(n)$ denotes the complexity of an $n \times n$ matrix multiplication [DKS88]. A term s matches a term t if there exists a substitution σ with $\sigma(s) = t$. Vitter and Simons give $\sqrt{n}$ time parallel algorithms for unification and some other P-complete problems [VS86].

A.6.8 Logical Query Program (LQP)

Given: An extended logic program P, and a ground clause C of the form $p(\bar{x}) :\!\!- q_1(\bar{y}_1), \ldots, q_k(\bar{y}_k)$. A *ground clause* is one in which all arguments (e.g., $\bar{x}$ and $\bar{y}_i$ above) are constants. An *extended logic program* is a basic logic program plus an extensional data base instance. A *basic logic program* is a finite set of rules. A *rule* is a disjunction

of literals having exactly one positive literal, called the *head*. The negative literals in the clause are called *subgoals*. The set of predicate symbols that appear only in subgoals is called the *Extensional Database*, or *EDB*. An *EDB fact* is an EDB predicate with constants as arguments. An *EDB instance* is a finite set of EDB facts. C is a *theorem* of P if the head of C is derivable from its subgoals in P.
Problem: Is C a theorem of P?
Reference: [UVG88]
Hint: Reduce PATH, Problem A.6.6, to LPQ. Let (X, R, S, T) be an instance of PATH. Without loss of generality, assume $S = \{s\}$. Let $t(Y)$ be an EDB relation specifying that node Y is in T. Let $r(U, V, W)$ be an EDB relation specifying that the triple of nodes (U, V, W) is in R. The basic logic program consists of the two rules $a(Y) :\!\!- t(Y)$ and $a(Y) :\!\!- r(Y, V, W), a(V), a(W)$. The relation a models "admissibility" in the path system, so $a(s)$ is a theorem of P if and only if s is admissible in the path system.
Remarks: Remains P-complete even for very restricted programs. In NC for programs with the "polynomial fringe property." See [UVG88] for details. See also [AP87] for related results.

A.7 Formal Languages

A.7.1 Context-Free Grammar Membership (CFGmem)

Given: A context-free grammar $G = (N, T, P, S)$ and a string $x \in T^*$.
Problem: Is $x \in L(G)$?
Reference: [JL76]
Hint: Reduce GEN to CFGmem. Let $(W, \bullet, V, w)$ be an instance of GEN. Construct the grammar $G = (W, \{a\}, P, w)$, where $P = \{x \rightarrow yz \mid y \bullet z = x\} \cup \{x \rightarrow \epsilon \mid x \in V\}$. It follows that $\epsilon \in L(G)$ if and only if w is generated by V.
Remarks: Goldschlager remarks it is the presence of ϵ-productions in the input grammar that make the membership question difficult [Gol81]. Lewis, Stearns, and Hartmanis' $\log^2 n$ space algorithm [LSH65] and Ruzzo's AC^1 (hence NC^2) algorithm [Ruz80] for general context free language recognition can both be modified to work with an ϵ-free grammar given as part of the input.

A.7.2 Context-Free Grammar Empty (CFGempty)

Given: A context-free grammar $G = (N, T, P, S)$.
Problem: Is $L(G)$ is empty?
Reference: [JL76, Gol81]
Hint: The reduction given in Problem A.7.1 suffices. The following reduction of MCVP to CFGempty (due to Martin Tompa, private communication) is also of interest. Given a circuit α construct the grammar $G = (N, T, P, S)$ such that $N = \{i \mid v_i \text{ is a vertex in } \alpha\}$, $T = \{a\}$, and $S = n$, where v_n is the output of α. Let $\nu(g)$ denote the value of gate g. The productions in P are of the following form:

1. For input v_i, $i \rightarrow a$ if $\nu(v_i)$ is TRUE,

2. $i \rightarrow jk$ if $v_i \leftarrow v_j \wedge v_k$,
3. $i \rightarrow j \mid k$ if $v_i \leftarrow v_j \vee v_k$.

Then $\nu(v_i)$ is TRUE if and only if $i \Rightarrow^* \gamma$, where $\gamma \in \{a\}^*$.
Remarks: Note, this reduction and the one for CFGinf, have no ϵ-productions yet remain complete. The original proof of Jones and Laaser reduced GEN to CFGempty. Their proof used the reduction for CFGmem and instead checked if $L(G)$ is empty [JL76].

A.7.3 Context-Free Grammar Infinite (CFGinf)

Given: A context-free grammar $G = (N, T, P, S)$.
Problem: Is $L(G)$ is infinite?
Reference: [Gol81, JL76]
Hint: Use a grammar similar to G in the proof for CFGempty, Problem A.7.2, except production $i \rightarrow a$ is replaced by $i \rightarrow x$, and the productions $x \rightarrow a$ and $x \rightarrow ax$ are also added.

A.7.4 Context-Free Grammar ϵ-Membership (CFGϵmem)

Given: A context-free grammar $G = (N, T, P, S)$.
Problem: Is $\epsilon \in L(G)$?
Reference: [Gol81, JL76]
Hint: Use a grammar similar to G in the proof for CFGempty, Problem A.7.2, except production $i \rightarrow a$ is replaced by $i \rightarrow \epsilon$.

A.7.5 Straight-Line Program Membership (SLPmem)

Given: A straight-line program over alphabet Σ, $|\Sigma| \geq 1$, with operations taken from $\Phi = \Sigma \cup \{\{\epsilon\}, \emptyset, \cup, \cdot\}$, and a string x.
Problem: Is x a member of the set constructed by the program?
Reference: [Goo83]
Hint: By noting there is a log space alternating Turing machine that "parses" x relative to the program, the problem is easily seen to be in P. Reduce MCVP to SLPmem by the following: TRUE $\rightarrow \{\epsilon\}$, FALSE $\rightarrow \emptyset$, $\wedge \rightarrow \cdot$, and $\vee \rightarrow \cup$.
Remarks: The original reduction was from GEN. Remains in P if $\cap$ is allowed. The analogous membership question for regular languages presented as regular expressions (NFAs) is complete for NSPACE($\log n$).

A.7.6 Straight-Line Program Nonempty (SLPnonempty)

Given: A straight-line program over alphabet Σ, $|\Sigma| \geq 1$, with operations taken from $\Phi = \Sigma \cup \{\{\epsilon\}, \emptyset, \cup, \cdot\}$, and a string x.
Problem: Is the set constructed by the program non-empty?
Reference: [Goo83]

Hint: Reduce SLPmem to SLPnonempty. Change non-empty constants to $\{\epsilon\}$, and test membership of ϵ.
Remarks: With $\cap$ added, SLPnonempty becomes complete for nondeterministic exponential time [Goo83].

A.7.7 Labeled GAP (LGAP)

Given: A fixed context free language L over alphabet Σ, a directed graph $G = (V, E)$ with edges labeled by strings in Σ^*, and two vertices s and t.
Problem: Is there a path from s to t such that the concatenation of its edge labels is in L?
Reference: [Ruz79, GHR91]
Hint: Reduce the 2-way DPDA acceptance problem, Problem A.7.8 to LGAP. Let $M = (Q, \Sigma', \Gamma, \delta, q_0, Z_0)$ be a 2-way DPDA [HU79] and let $x \in \Sigma'$ be an input string. Without loss of generality, the PDA has a unique final configuration q_f and accepts with empty stack with its head at the right end of the input. Let $\Sigma = \Gamma \cup \{\overline{Z} \mid Z \in \Gamma\}$. Let V be the set containing the special vertex s, together with all "surface configurations" of the PDA, i.e. $Q \times \{1, \ldots, |x|\}$. There is an edge from $\langle p, i\rangle$ to $\langle q, j\rangle$ labeled $\alpha \in \Sigma^*$ if and only if when reading x_i the PDA has a move from p to q that moves its input head $j - i \in \{-1, 0, 1\}$ cells to the right, pops $Z \in \Gamma$, and pushes $\beta \in \Gamma^*$, where $\alpha = \overline{Z}\beta$. Additionally, there is an edge from the special vertex s to the initial surface configuration $\langle q_0, 1\rangle$, labeled Z_0 (the initial stack symbol). The designated vertex t is $\langle q_f, |x|\rangle$. Finally, $L = L(G)$, where $G : \{S \rightarrow aS\overline{a}S \mid a \in \Gamma\} \cup \{S \rightarrow \epsilon\}$, i.e., the semi-Dyck language $D_{|\Gamma|}$ on $|\Gamma|$ letters [Har79, Section 10.4].
Remarks: Remains P-complete when L is D_2. An equivalent statement is that it is P-complete to decide, given a deterministic finite state automaton M, whether $D_2 \cap L(M) = \emptyset$. If G is acyclic then the problem is complete for $SAC^1 = LOGCFL$ [Ruz79].

A.7.8 Two-Way DPDA Acceptance (2DPDA)

Given: A two-way deterministic pushdown automaton M and a string x.
Problem: Is x accepted by M?
Reference: [Coo71a, Gal74, Gal77, Lad75]
Hint: See, e.g., [HU79] for a definition of 2DPDAs. Cook [Coo71a] gives a direct simulation of a polynomial time Turing machine by a logarithmic space *auxiliary* pushdown automaton. Galil [Gal74, Gal77] shows existence of a P-complete language accepted by a 2DPDA, in effect showing that the logarithmic space worktape isn't crucial to Cook's simulation. (See also [Sud78] for a general reduction of auxiliary PDAs to ordinary PDAs.) Ladner [Lad75] gives a much more direct proof by observing that a suitably encoded version of CVP is solvable by a 2DPDA, basically by doing a depth first search of the circuit, using the stack for backtracking.
Remarks: Remains in P when generalized to nondeterministic and/or logarithmic space auxiliary PDAs [Coo71a]. When restricted to polynomial time PDAs, or one-way PDAs, even with a logarithmic space worktape, the problem is in NC; specifically it is complete for $LOGDCFL$ in the deterministic case, and for $LOGCFL = SAC^1$ in the nondeterministic

case.

A.8 Algebraic Problems

A.8.1 Finite Horizon Markov Decision Process (FHMDP)

Given: A *nonstationary* Markov decision process $M = (S, c, p)$ and an integer T. Before defining the problem, we present some background on Markov decision processes. The term *finite horizon* refers to the time bound T. S is a finite set of states and contains a designated initial state s_0. Let s_t denote the current state of the system for each time $t = 1, 2, \ldots$ Associated with each state $s \in S$ is a finite set of decisions D_s. A cost of $c(s, i, t)$ is incurred at time t by making decision $i \in D_{s_t}$. The next state s' has probability distribution given by $p(s, s', i, t)$. If c and p are independent of t then the process is said to be *stationary.* A *policy* δ is a mapping that assigns to each time step t and each state s a decision $\delta(s, t)$. A policy is *stationary* if δ is independent of time, and can then be suplied as an input.
Problem: Is the expectation of the cost, $\sum_{t=0}^{T} c(s_t, \delta(s_t, t), t)$, equal to 0?
Reference: [PT87]
Hint: There is a polynomial time algorithm for the problem that uses dynamic programming. The reduction is from the monotone circuit value problem, Problem A.1.3. Let $c = ((a_i, b_i, c_i), i = 1, \ldots, k)$ denote an encoding of MCVP, where a_i denotes gate type, and b_i and c_i denote the numbers of gate i's inputs. A *stationary* Markov process $M = (S, c, p)$ is constructed from the circuit instance as follows. S has one state q_i for each i, $1 \leq i \leq k$. There is an additional state in S called q. If (a_i, b_i, c_i) corresponds to a circuit input, then the corresponding state q_i has a single decision 0 with $p(q_i, q, 0) = 1$, and cost $c(q_i, 0) = 1$ (0) if a_i is a FALSE (TRUE) input. All other costs of this process are 0. There is one decision 0 for state q and $p(q, q, 0) = 1$. If a_i is an OR gate then there are two decisions 0 and 1 from state q_i. The associated probabilities are $p(q_i, q_{b_i}, 0) = 1$ and $p(q_i, q_{c_i}, 1) = 1$. The associated costs are both 0. If a_i is an AND gate then there are two decisions 0 and 1 from state q_i. $p(q_i, q_{b_i}, 0) = 1/2$ and $p(q_i, q_{c_i}, 1) = 1/2$, with associated costs 0. The initial state is k corresponding to the output gate of the circuit; the time horizon T is k. It is easy to verify that the expected cost of the process is 0 if and only if the circuit evaluates to TRUE.
Remarks: The reduction shows that the finite horizon stationary version of the problem is P-hard. This problem is not known to be in P. The *deterministic* version of the FHMDP, which requires that p has only values 0 or 1, is in NC [PT87]. Note, this last result holds for both the stationary and nonstationary versions of the problem.

A.8.2 Discounted Markov Decision Process (DMDP)

Given: A stationary Markov decision process $M = (S, c, p)$ and a real number $\beta \in (0, 1)$. See Problem A.8.1 for definitions. This problem is *infinite horizon*, that is, there is no time bound.
Problem: Is the expectation of the cost, $\sum_{t=0}^{\infty} c(s_t, \delta(s_t, t))\beta^t$, equal to 0?
Reference: [PT87]
Hint: The problem can be phrased as a linear programming problem and solved in polynomial time. The same construction as used in Problem A.8.1 can be used to show this

problem is P-complete.
Remarks: The infinite horizon, discounted, *deterministic* problem is in NC [PT87]. The deterministic problem requires that p has values only 0 or 1.

A.8.3 Average Cost Markov Decision Process (ACMDP)

Given: A stationary Markov decision process $M = (S, c, p)$. See Problem A.8.1 for definitions. This problem is *infinite horizon*, that is, there is no time bound.
Problem: Is the $\lim_{T\to\infty} \left(\sum_{i=0}^{T} c(s_t, \delta(s_t, t))/T\right) = 0$?
Reference: [PT87]
Hint: The problem can be phrased as a linear programming problem and solved in polynomial time. The reduction is from a synchronous variant of MCVP, Problem A.1.6. The construction is a modification to that given in Problem A.8.1. Instead of having states corresponding to circuit inputs going to a new state q, they have transitions to the initial state. The limit is 0 if and only if the circuit instance evaluates to TRUE.
Remarks: The infinite horizon, average cost, *deterministic* problem is in NC [PT87]. The deterministic problem requires that p has values only 0 or 1.

A.8.4 Gaussian Elimination with Partial Pivoting (GEPP)

Given: An $n \times n$ matrix A with entries over the reals or rationals and an integer l.
Problem: Is the pivot value for the l^{th} column positive when Gaussian elimination with *partial pivoting* is performed on A? Partial pivoting is a technique used to obtain numerical stability in which rows of the matrix are exchanged so that the largest value in a given column can be used to perform the elimination.
Reference: [Vav89]
Hint: The standard Gaussian elimination algorithm requires $O(n^3)$ operations. Since the size of the numbers involved can be bounded by a polynomial in n (see [Vav89]), the problem is in P. To show completeness reduce NAND CVP to GEPP. Without loss of generality, assume the inputs and gates of the circuit are numbered in topological order from 1 to G, where G numbers the output gate. A $2G \times 2G$ matrix $A = (a_{i,j})$ is constructed from the instance of CVP. The entries of A are described below. A TRUE circuit input i contributes entry -3.9 in position $a_{2i-1,i}$ and entry 0 in position $a_{2i,i}$. For FALSE input i or NAND gate i, A has entry -3.9 in position $a_{2i-1,i}$ and 4.0 in position $a_{2i,i}$. If gate i is an input to gate k, then A has entry 0 in position $a_{2k-1,i}$ and entry 1 in position $a_{2k,i}$. For $1 \leq i \leq G$, A has entry $a_{2i,G+i} = 1$. All unspecified matrix entries have value 0. The pivot value used in eliminating column G is positive (negative) if and only if the circuit evaluates to FALSE (TRUE).
Remarks: The reduction does not rely on large numbers, and therefore it shows that the problem is strongly P-complete. Another decision problem that is strongly complete for P based on Gaussian elimination with partial pivoting is as follows: given matrix A, and integers i and j, is the pivot used to eliminate the j^{th} column taken from the initial i^{th} row? Vavasis also shows that Gaussian elimination with *complete pivoting* is P-complete. In complete pivoting both rows and columns are interchanged so that the largest remaining

matrix entry can be used as a pivot. The reduction for complete pivoting does not show the problem is strongly P-complete. This question is open.

A.8.5 Iterated Mod (IM)

Given: Integers $a, b_1, \ldots, b_n$.
Problem: Is $((\cdots((a \bmod b_1) \bmod b_2)\cdots) \bmod b_n) = 0$?
Reference: [KR89]
Hint: Reduce NAND CVP to IM. Without loss of generality, assume the gates are numbered in reverse topological order from G down to 1, where the output gate is numbered 1. Let $y_1, \ldots, y_r$ denote the inputs and $Y_l \in \{0,1\}$ denote the value of input y_l. The input wires to gate g are numbered $2g$ and $2g-1$. Let a be a bit vector of length $2G+1$ whose j^{th} bit is Y_l if edge j is incident from input y_l, and 1 otherwise. Let O_g represent the set of out-edge labels from gate g. For $1 \le g \le G$, we construct moduli $b_1, \ldots, b_{2G}$ as follows:

$$b_{2g} = 2^{2g} + 2^{2g-1} + \sum_{j \in O_g} 2^j \text{ and } b_{2g-1} = 2^{2g-1}.$$

The output gate in the NAND CVP instance has value 0 if and only if

$$((\cdots((a \bmod b_1) \bmod b_2)\cdots) \bmod b_{2G}) = 0.$$

Remarks: The polynomial iterated mod problem is the problem in which $a(x), b_1(x), \ldots, b_n(x)$ are univariate polynomials over a field F and the question is to determine if

$$((\cdots((a(x) \bmod b_1(x)) \bmod b_2(x))\cdots) \bmod b_n(x)) = 0.$$

This problem is in NC [KR89]. The proof technique used to show IM is P-complete can be modified to show the *superincreasing knapsack problem* is also P-complete [KR89]. The superincreasing knapsack problem is defined analogously to the knapsack problem [GJ79] with weights $w_1, \ldots, w_n$, except for $2 \le i \le n$, $w_i > \sum_{j=1}^{i-1} w_j$.

A.8.6 Generalized Word Problem (GWP)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let $U = \{u_1, \ldots, u_m\} \subseteq \underline{S}^*$, where $m \in N$ and let $x \in \underline{S}^*$.
Problem: Is $x \in \langle U \rangle$? That is, is x in the subgroup of F generated by U?
Reference: [AM84b, Ste89]
Hint: Stewart reported an error in the result contained in [AM84b]. The reduction in [AM84b] is a generic one from a normal form Turing machine. However, it reduces a Turing machine computation to a version of GWP where S is a countably infinite set [Ste89]. Stewart shows using the Nielsen reduction algorithm that this problem is still in P. Stewart calls this P-complete problem the *generalized word problem for countably-generated free groups* (GWPC). He shows that GWPC is log space reducible to GWP thus proving GWP is P-complete as well [Ste89].

Remarks: For a natural number k, GWPC(k) and GWPC($\leq k$) are the generalized word problems for finitely-generated subgroups of countably-generated free groups where all words involved are of length exactly k and at most k respectively. Stewart shows GWPC(k) and GWPC($\leq k$) are P-complete for $k > 2$ [Ste90]. When $k = 2$ the problems are complete for symmetric log space (SL). The *word problem for a free group* is to decide whether x equals the empty word in F. This problem is solvable in deterministic log space [LZ77].

A.8.7 Subgroup Containment (SC)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let $U = \{u_1, \ldots, u_m\}, V = \{v_1, \ldots, v_p\} \subseteq \underline{S}^*$, where $m, p \in N$.
Problem: Is the group generated by U a subgroup of the group generated by V?
Reference: [AM84b]
Hint: A variant of the Nielsen reduction algorithm can be used to show the problem is in P. The reduction is from the generalized word problem, Problem A.8.6. Observe that for any $x \in \underline{S}^*$ and $U \subseteq \underline{S}^*$, $x \in \langle U \rangle$ if and only if $\langle x \rangle$ is a subgroup of $\langle U \rangle$.
Remarks: Since $\langle U \rangle$ is a subgroup of $\langle V \rangle$ if and only if $\langle U \cup V \rangle$ is normal in $\langle V \rangle$, it follows that the *normal subgroup problem*, which is also in P, is P-complete. The problem of determining whether $\langle U \rangle$ is normal in $\langle U, x \rangle$ is also P-complete [AM84b].

A.8.8 Subgroup Equality (SE)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let $U = \{u_1, \ldots, u_m\}, V = \{v_1, \ldots, v_p\} \subseteq \underline{S}^*$, where $m, p \in N$.
Problem: Is $\langle U \rangle = \langle V \rangle$?
Reference: [AM84b]
Hint: A variant of the Nielsen reduction algorithm can be used to show the problem is in P. The reduction is from the subgroup containment problem, Problem A.8.7. Observe $\langle U \rangle$ is a subgroup of $\langle V \rangle$ if and only if $\langle U \cup V \rangle = \langle V \rangle$.

A.8.9 Subgroup Finite Index (SFI)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let $U = \{u_1, \ldots, u_m\}, V = \{v_1, \ldots, v_p\} \subseteq \underline{S}^*$, where $m, p \in N$.
Problem: Is $\langle U \rangle$ a subgroup of $\langle V \rangle$ with finite *index* in $\langle V \rangle$? The index of U in V is the number of distinct right cosets of U in V.
Reference: [AM84b]
Hint: A variant of the Nielsen reduction algorithm can be used to show the problem is in P. The reduction is from the subgroup containment problem, Problem A.8.7. Note, $\langle U \rangle$ is a subgroup of $\langle V \rangle$ if and only if $\langle U \cup V \rangle$ has finite index in $\langle V \rangle$. Let $x \in \underline{S}^*$. The problem of determining whether $\langle U \rangle$ has finite index in $\langle U, x \rangle$ is also P-complete [AM84b].

A.8.10 Group Independence (GI)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let $U = \{u_1, \ldots, u_m\} \subseteq \underline{S}^*$, where $m \in N$.
Problem: Is U *independent*? That is, does each $x \in \langle U \rangle$ have a unique *freely reducible* representation. A word w is freely reducible if it contains no segment of the form ss^{-1} or $s^{-1}s$.
Reference: [AM84b]
Hint: A variant of the Nielsen reduction algorithm can be used to show the problem is in P. The reduction is from an arbitrary polynomial time Turing machine [AM84b].

A.8.11 Group Rank (GR)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let $k \in N$. Let $U = \{u_1, \ldots, u_m\} \subseteq \underline{S}^*$, where $m \in N$.
Problem: Does $\langle U \rangle$ have *rank* k? The rank is the number of elements in a minimal generating set.
Reference: [AM84b]
Hint: A variant of the Nielsen reduction algorithm can be used to show the problem is in P. The reduction is from the group independence problem, Problem A.8.10. Observe U is independent if and only if $\langle U \rangle$ has rank the number of elements in U.

A.8.12 Group Isomorphism (SI)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let $U = \{u_1, \ldots, u_m\}, V = \{v_1, \ldots, v_p\} \subseteq \underline{S}^*$, where $m, p \in N$.
Problem: Is $\langle U \rangle$ isomorphic to $\langle V \rangle$?
Reference: [AM84b]
Hint: A variant of the Nielsen reduction algorithm can be used to show the problem is in P. The reduction is from the group independence problem, Problem A.8.10. U is independent if and only if $\langle U \rangle$ and $\langle \{s_1, \ldots, s_{|U|}\} \rangle$ are isomorphic.

A.8.13 Group Induced Isomorphism (GII)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let $U = \{u_1, \ldots, u_m\}, V = \{v_1, \ldots, v_p\} \subseteq \underline{S}^*$, where $m = p \in N$.
Problem: Does the mapping ϕ defined by $\phi(u_i) = v_i$ for $i = 1, \ldots, m$, induce an isomorphism from $\langle U \rangle$ to $\langle V \rangle$?
Reference: [AM84b]
Hint: A variant of the Nielsen reduction algorithm can be used to show the problem is in P. The reduction is from the group independence problem, Problem A.8.10. U is independent if and only if ϕ as defined above induces an isomorphism from $\langle U \rangle$ to $\langle \{s_1, \ldots, s_{|U|}\} \rangle$.

A.8.14 Intersection of Cosets (IC)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let $x, y \in \underline{S}^*$. Let $U = \{u_1, \ldots, u_m\}, V = \{v_1, \ldots, v_p\} \subseteq \underline{S}^*$, where $m = p \in N$.
Problem: Is $\langle U \rangle x \cap y \langle V \rangle$ non-empty?
Reference: [AM84c]
Hint: A polynomial time algorithm for the problem is given in [AM84c]. The reduction is from an arbitrary polynomial time Turing machine [AM84c].
Remarks: The *intersection of right cosets problem* and the *intersection of left cosets problem* are subproblems of the intersection of cosets problems and they are both P-complete as well. For example, the right coset problem is P-complete since $\langle U \rangle x \cap \langle V \rangle y$ is non-empty if and only if $\langle U \rangle x y^{-1} \cap e \langle V \rangle$ is non-empty, where e denotes the empty word.

A.8.15 Intersection of Subgroups (IS)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let e denote the empty word. Let $U = \{u_1, \ldots, u_m\}, V = \{v_1, \ldots, v_p\} \subseteq \underline{S}^*$, where $m = p \in N$.
Problem: Is $\langle U \rangle \cap \langle V \rangle \neq \langle e \rangle$?
Reference: [AM84c]
Hint: A polynomial time algorithm for the problem is given in [AM84c]. The reduction is straightforward from the intersection of right cosets problem, see Problem A.8.14.

A.8.16 Group Coset Equality (GCE)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let $x, y \in \underline{S}^*$. Let $U = \{u_1, \ldots, u_m\}, V = \{v_1, \ldots, v_p\} \subseteq \underline{S}^*$, where $m = p \in N$.
Problem: Is $\langle U \rangle x = y \langle V \rangle$?
Reference: [AM84c]
Hint: A polynomial time algorithm for the problem is given in [AM84c]. The reduction is from Problem A.8.15.
Remarks: The following three decision problems are also P-complete: *equality of right cosets* — Does $\langle U \rangle x = \langle V \rangle y$?, *equivalence of cosets* — Are there x, y such that $\langle U \rangle x = y \langle V \rangle$?, and *equivalence of right cosets* — Are there x, y such that $\langle U \rangle x = \langle V \rangle y$? [AM84c].

A.8.17 Conjugate Subgroups (CS)

Given: Let S be a finite set and F be the free group generated by S. Let $\underline{S} = \{s, s^{-1} | s \in S\}$, where s^{-1} denotes the inverse of s. Let $\underline{S}^*$ denote the set of all finite words over $\underline{S}$. Let $U = \{u_1, \ldots, u_m\}, V = \{v_1, \ldots, v_p\} \subseteq \underline{S}^*$, where $m = p \in N$.
Problem: Is there an $x \in \underline{S}^*$ such that $x^{-1} \langle U \rangle x = \langle V \rangle$?
Reference: [AM84c]
Hint: A polynomial time algorithm for the problem is given in [AM84c]. The reduction is

from the equivalence of right cosets problem, see Problem A.8.16.
Remarks: The problem of determining whether $x^{-1}\langle U\rangle x$ is a subgroup of $\langle V\rangle$ is also P-complete [AM84c].

A.8.18 Uniform Word Problem for Finitely Presented Algebras (UWPFPA)

Given: A finitely presented algebra $\mathcal{A} = (M, A, \Gamma)$ and a pair of terms x, y. M is a finite set of *symbols* and $A : M \to N$ defines the *arity* of each symbol. N represents the non-negative integers. M is partitioned into two sets: $G = \{a \in M \mid A(a) = 0\}$ consists of *generator symbols* and $O = \{a \in M \mid A(a) > 0\}$ consists of *operator symbols*. The set of *terms* over M is the smallest subset of M^* such that:

1. all elements of G are terms,
2. if θ is m-ary and $x_1, \ldots x_m$ are terms, then $\theta(x_1, \ldots, x_m)$ is a term.

Let τ denote the set of terms. Γ is a set of unordered pairs of terms called *axioms*. $\equiv$ is the smallest congruence relation on τ satisfying the axioms of Γ.
Problem: Is $x \equiv y$?
Reference: [Koz77]
Hint: A polynomial time algorithm for the problem is given in [Koz77]. The reduction is from the monotone circuit value problem, Problem A.1.3. Let $\mathcal{B}$ be an instance MCVP represented as a list of assignments to variables $C_1, \ldots, C_n$ of the form $C_i = 0, C_i = 1, C_i = C_j \vee C_k$, or $C_i = C_j \wedge C_k$, where $i > j, k$. $\mathcal{B}$ is in MCVP provided value$(C_n) = 1$, where n denotes the output gate of the circuit. The reduction is as follows: $G = \{C_1, \ldots, C_n, 0, 1\}$, $O = \{\vee, \wedge\}$, $\Gamma = \mathcal{B} \cup \{0 \vee 0 \equiv 0, 0 \vee 1 \equiv 1, 1 \vee 0 \equiv 1, 1 \vee 1 \equiv 1, 0 \wedge 0 \equiv 0, 0 \wedge 1 \equiv 0, 1 \wedge 0 \equiv 0, 1 \wedge 1 \equiv 1\}$. $\mathcal{B}$ is in MCVP if and only if $C_n \equiv 1$.

A.8.19 Triviality Problem for Finitely Presented Algebras (TPFPA)

Given: A finitely presented algebra $\mathcal{A} = (M, A, \Gamma)$. See Problem A.8.18 for definitions.
Problem: Is $\mathcal{A}$ *trivial*? That is, does it contain only one element.
Reference: [Koz77]
Hint: A polynomial time algorithm for the problem is given in [Koz77]. The reduction is from MCVP. Construct Γ as done in the proof hint for Problem A.8.18. Let $\Gamma' = \Gamma \cup \{C_n \equiv 0\}$. Using notation from Problem A.8.18, it follows that $\mathcal{B}$ is an instance of MCVP if and only if $C_n \equiv 1$, that is, if and only if $1 \equiv_{\Gamma'} 0$ that is, if and only if $\tau / \equiv_{\Gamma'}$ is trivial.

A.8.20 Finitely Generated Subalgebra (FGS)

Given: A finitely presented algebra $\mathcal{A} = (M, A, \Gamma)$ and terms $x_1, \ldots, x_n, y$. Let $[x] = \{y \in \tau \mid x \equiv y\}$. See Problem A.8.18 for definitions.
Problem: Is $[y]$ contained in the subalgebra generated by $[x_1], \ldots, [x_n]$?
Reference: [Koz77]

Hint: A polynomial time algorithm for the problem is given in [Koz77]. This problem is a general formulation of GEN, Problem A.6.5, and so it follows that it is also P-complete.

A.8.21 Finiteness Problem for Finitely Presented Algebras (FPFPA)

Given: A finitely presented algebra $\mathcal{A} = (M, A, \Gamma)$. See Problem A.8.18 for definitions.
Problem: Is $\mathcal{A}$ finite?
Reference: [Koz77]
Hint: A polynomial time algorithm for the problem is given in [Koz77]. The reduction is from MCVP and is similar to that used in Problem A.8.19. The algebra constructed in that proof is modified as follows: add another generator b to G, and the axioms $\{b \wedge b \equiv 0,\ b \wedge 0 \equiv 0,\ 0 \wedge b \equiv 0,\ b \vee b \equiv 0,\ b \vee 0 \equiv 0,\ 0 \vee b \equiv 0\}$ to Γ' to obtain Γ''. Γ'' is finite if Γ' is trivial and otherwise it is infinite.

A.8.22 Uniform Word Problem for Lattices (UWPL)

Given: Let E be a set of equations and $e_1 = e_2$ an equation. We present some preliminary definitions before defining the problem. A *lattice* is a set L with two binary operations $\{+, \cdot\}$ that satisfies the *lattice axioms*. Let $x, y, z \in L$. The lattice axioms are as follows:

1. associativity: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$, $(x + y) + z = x + (y + z)$,
2. commutativity: $x \cdot y = y \cdot x$, $x + y = y + x$,
3. idempotence: $x \cdot x = x$, $x + x = x$, and
4. absorption: $x + (x \cdot y) = x$, $x \cdot (x + y) = x$.

Let $\mathcal{U}$ be a countably infinite set of *symbols*. The set of *terms* over $\mathcal{U}$, $W(\mathcal{U})$, is defined inductively as follows:

1. If α is in $\mathcal{U}$, then α is in $W(\mathcal{U})$.
2. If p, q are in $W(\mathcal{U})$, then $(p + q)$, $(p \cdot q)$ are in $W(\mathcal{U})$.

Let e_1 and e_2 be terms over $\mathcal{U}$. An *equation* is a formula of the form $e_1 = e_2$. A *valuation* for a given lattice L is a mapping $\mu : \mathcal{U} \to L$. The valuation is extended to $W(\mathcal{U})$ by defining $\mu(p + q) = \mu(p) + \mu(q)$, and $\mu(p \cdot q) = \mu(p) \cdot \mu(q)$. A lattice satisfies an equation $e_1 = e_2$ under a valuation μ, denoted $L \models_\mu e_1 = e_2$, if and only if $\mu(e_1) = \mu(e_2)$. A lattice L satisfies a set of equations E, denoted $L \models_\mu E$, if and only if L satisfies every member of E under μ. E *implies* $e_1 = e_2$, denoted $E \models e_1 = e_2$, if and only if for every lattice L and valuation μ such that $L \models_\mu E$, it follows that $L \models_\mu e_1 = e_2$.
Problem: Does $E \models e_1 = e_2$?
Reference: [Cos88]
Hint: A polynomial time algorithm for the problem is given in [Cos88]. The reduction is from the *implication problem for propositional Horn clauses* [JL76]. See Problems A.6.1 and A.6.2. Let Σ be a set of propositional formulas of the form $x_i \wedge x_j \Rightarrow x_k$, where $x_1, x_2, \ldots$

are propositional variables. Let σ be the formula $x_1 \wedge x_2 \Rightarrow x_3$. The problem is to test if Σ implies σ. Let ϕ represent the formula $x_j \wedge x_j \Rightarrow x_k$. In the instance of UWPL we construct the equation ϵ_ϕ as follows $\alpha_i \cdot \alpha_j \cdot \alpha_k = \alpha_i \cdot \alpha_k$. Let $E_\Sigma = \{\epsilon_\phi \mid \phi \in \Sigma\}$. It follows that Σ implies σ if and only if $E_\Sigma \models \epsilon_\sigma$.
Remarks: The problem remains P-complete if we use inequalities instead of equations. Furthermore, the problem remains P-complete when $E = \emptyset$ and the terms are represented by dags instead of trees. However, if $E = \emptyset$ and the terms are represented as trees the problem is in *DLOG* [Cos88]. This problem is called the *identity problem for lattices.*

A.8.23 Generator Problem for Lattices (GPL)

Given: Let L be a lattice. Let E be a set of equations and $e, g_1, \ldots, g_n$ be terms over $\mathcal{U}$. Let $\mu : \mathcal{U} \rightarrow L$ be a valuation (see Problem A.8.22). We present some preliminary definitions first. Let $X \subseteq L$. The *sublattice generated* by X is the smallest subset of L that contains X and is closed under the operations of L. e is *generated by* $g_1, \ldots, g_n$ in L under μ, denoted $L \models_\mu \text{gen}(e, g_1, \ldots, g_n)$, if and only if $\mu(e)$ is in the sublattice of L generated by the set $\{\mu(g_i) \mid i = 1, \ldots, n\}$. *$E$ implies that e is generated by* $g_1, \ldots, g_n$, denoted $E \models \text{gen}(g_1, \ldots, g_n)$, if and only if for every lattice L and valuation μ such that $L \models_\mu E$, it follows that $L \models_\mu \text{gen}(e, g_1, \ldots, g_n)$.
Problem: Does $E \models \text{gen}(e, g_1, \ldots, g_n)$?
Reference: [Cos88]
Hint: A polynomial time algorithm for the problem is given in [Cos88]. The reduction is a continuation of the reduction used in Problem A.8.22. Since $E_\Sigma \models \epsilon_\sigma$ if and only if $E_\Sigma \models \text{gen}(\alpha_1 \cdot \alpha_2 \cdot \alpha_3, \alpha_1, \alpha_2)$, it follows that GPL is also P-complete.
Remarks: The problem remains P-complete when $E = \emptyset$ and the terms are represented by dags instead of trees. However, if $E = \emptyset$ and the terms are represented as trees the problem is in *DLOG* [Cos88]. This problem is called the *generator problem for free lattices.*

A.8.24 Boolean Recurrence Equation (BRE)

Given: (M, B, F, j), where M is an $m \times n$ Boolean matrix, B is an $n \times n$ Boolean matrix, F is an $n \times 1$ Boolean vector, and j is an integer in the range 0 to n.
Problem: Is the first entry of $M \cdot Y_j$ a 1? Y_j is defined recursively as $Y_0 = F$, $Y_k = B \cdot Y_{k-1}$ for $k \geq 1$.
Reference: [BBMS88]
Hint: The reduction is from an alternating Turing machine that uses $O(\log n)$ space.
Remarks: If $0 \leq j \leq \log^k n$, then the problem is complete for AC^k [BBMS88]. That is, the class of problems accepted by alternating Turing machines in $O(\log n)$ space and $O(\log^k n)$ alternations.

A.9 Geometry

A.9.1 Plane Sweep Triangulation (PST)

Given: An n vertex polygon P that may contain holes, and a designated vertex u.

Problem: Is a vertical edge connecting to u in the *plane sweep triangulation* of P? The plane sweep triangulation is the triangulation produced by sweeping from top to bottom a horizontal line L. When L encounters a vertex v of P, each diagonal from v to another vertex in P, which does not cross a previously drawn diagonal, is added to the triangulation.
Reference: [ACG90]
Hint: It is easy to see the plane sweep triangulation algorithm runs in polynomial time. The reduction is from a variant of planar CVP, Problem A.1.7. The new version of PCVP consists of NOT gates of fanout 1, OR gates of fanout 1, routing gates, and fanout gates that take one value and produce two copies of it. This instance of PCVP is required to be laid out on a planar grid in a special manner with alternating layers of routing and logic. The reduction involves constructing "geometric" gadgets for routing (left and right shifts in the grid), "vertical" wires, fanout 1 OR gates, and NOT gates. The presence (absence) of a vertical edge in the triangulation denotes a TRUE (FALSE) value. The vertex u is a special "target" vertex in the output gate of the circuit. A vertical line is connected to u in the triangulation if and only if the circuit evaluates to TRUE.
Remarks: The problem of finding some arbitrary triangulation is in NC [Goo89]. If the polygon P is not allowed to have holes then the complexity of the problem is open. In [ACG90] they conjecture that this restricted version is in NC.

A.9.2 3-Oriented Weighted Planar Partitioning (3OWPP)

Given: A set of non-intersecting line segments $s_1, \ldots, s_n$ in the plane, a set of associated integer weights $w_1, \ldots, w_n$, and two designated segments r and t. The segments are *3-oriented* meaning that there are only three different possible slopes for the segments.
Problem: Do segments r and t "touch" in the partitioning of the plane constructed by extending segments in the order of their weights? Segments are extended until they reach another segment or a previous segment extension.
Reference: [ACG90]
Hint: It is easy to see that the process of extending the segments can be performed in polynomial time. The reduction is from the same version of PCVP as used in Problem A.9.1. The gates of the instance of PCVP are numbered in topological order. Gadgets are constructed for routing and for logic. There are gadgets for right and left shifts, fanout gates, OR gates, NOT gates, and TRUE inputs. TRUE values in the circuit are transmitted as vertical extensions of segments. The most interesting gadget is the one for the NOT gate and we describe this. The gadgets for the other constructs all involve only two different sloped segments, whereas to simulate NOT gates three different slopes are required. The instance of PCVP was laid out on a grid so we consider a NOT gate numbered i that receives its input on channel j and has its output on channel k. A *blocking segment* is one that is used to prevent the extension of another line segment and whose weight is very large. These segments don't play an active role in simulating the gate. The NOT gadget consists of six blocking segments and three additional segments — called "one," "two," and "three" indicating their relative weights. That is, segment one is processed first within the gadget followed by two and then three. Segment two is a horizontal segment whose potential extension spans channels j and k, and is blocked on both ends. Two lies directly to the left of channel j. Segment three is a vertical segment on channel k blocked "above" segment two but with the possibility of

being extended downward across two's extension. Channel j, the input to NOT gate i, is blocked above segment two. Segment one has a slope of -1 and its potential extension is blocked at both ends. Segment one lies completely to the left of channel j. Its rightward extension would cross channel j as well as segment two's extension. We now describe how this gadget simulates a NOT gate. If the input to gate i is TRUE (FALSE) then the vertical segment on channel j has (has not) been extended to where it is blocked. This prevents (allows) segment one from being extended across segment two. Thus, segment two can be (cannot be) extended towards the right across channel k. This prevents (allows) segment three from being extended across segment two's extension indicating a FALSE (TRUE) value for the output of the gate. The assignments of weights and the construction of all gadgets can be accomplished in log space. Segment r is the vertical segment corresponding to the output wire of the circuit. Segment t is a special output "pad." r will touch t when extended if and only if the circuit evaluates to TRUE.
Remarks: The complexity of the 2-oriented version of the problem is open [ACG90]. In [ACG90] they remark that the problem has been reduced to an instance of MCVP that has a very restricted topology, although not planar. Thus, it is open whether or not this version of the problem is in NC.

A.9.3 Visibility Layers (VL)

Given: A set of n non-intersecting line segments in the plane and a designated segment s.
Problem: Is the label assigned to segment s by the *visibility layering process* congruent to one mod three? The visibility layering process is repeatedly to compute and delete the *upper envelope* of the remaining set of segments and label those segments with the current depth. The upper envelope consists of those segments *visible* from the point $(0, +\infty)$. A segment is visible from a point p if a ray cast from p can hit the segment before hitting any other segment.
Reference: [ACG90, Her90]
Hint: The visibility layering process is polynomial time [Her90]. The reduction presented in [Her90] is from the monotone CVP. We sketch this reduction. The gates in the instance of CVP are assumed to be numbered in topological order. A grid is constructed that consists of $V + 1$ rows and E columns, where V (E) is the number of nodes (edges) in the dag corresponding to the circuit. Gadgets are constructed for the AND and OR gates. Gadgets consist of horizontal line segments of varying lengths. The gadget for AND gate k with inputs i and j ($i < j$), and outputs l and m ($l < m$) consists of three horizontal segments situated in row k of the grid. One segment spans column i, one segment spans column j, and another segment spans from column i through column m. The gadget for OR gate k with inputs i and j ($i < j$), and outputs l and m ($l < m$) consists of three horizontal segments situated in row k of the grid. One segment spans column j, one segment spans columns i through j, and another segment spans columns j through m. If an input associated with a given column is TRUE (FALSE) then a horizontal segment is put (is not put) in to span that column in row 0. "Deepeners," which are horizontal line segments spanning single columns of the grid, are used to make sure gate input values arrive at the "right time" and also to make sure that once a gate has been evaluated its outputs affect only the desired gates. The output of the circuit is TRUE if and only if a s has a label whose value is congruent to

one mod three.
Remarks: The reduction given in [ACG90] is similar and is also from a variant of MCVP. The main difference is in the way fanout is treated. The version of MCVP used in [ACG90] consists of *crossing fanout gates*, single output AND gates, and single output OR gates. An instance consists of alternate routing and logic layers. Gadgets are constructed for the three types of gates and a similar decision problem to the one in [Her90] is posed to determine the output of the circuit. If the length of all segments is required to be the same then the complexity of the problem is not known [ACG90]. In [ACG90] they conjecture that this version of the problem is in NC.

A.10 Real Analysis

A.10.1 Real Analogue to CVP (RealCVP)

Given: A *feasible* real function V defined on $(-\infty, +\infty)$. A real function f is *feasible* if, given a sufficiently accurate fixed-point binary approximation to $x \in [-2^n, 2^n]$, a fixed-point binary approximation to $f(x)$ with absolute error $< 2^{-n}$, can be computed in time $n^{O(1)}$. (Sufficiently accurate means that the error in approximating the input x is $< 2^{-n^{O(1)}}$. This fixes the number of input bits, the continuity of f limits its range, and thus fixes the number of output bits, both are polynomial in n.)
Problem: Compute $V(x)$ with absolute error $< 2^{-n}$.
Reference: [Hoo91] [Hoo90]
Hint: The function V computes the continuous analog of the circuit value function by mapping circuit descriptions, along with their possible inputs, onto the real line. To evaluate the circuit α on input x, treat the encoding $\overline{\alpha}$ as an integer, and the bits $\overline{x}$ as an n-bit fixed-point binary fraction, and add the two. The value of $V(\overline{\alpha}.\overline{x})$ is then a rational number that encodes the values of the gates of α on the input x. To make V continuous between these evaluation points, V is simply linearly interpolated.
Remarks: The same function yields a family of P-complete polynomials $\{p_n\}$ computable by *feasible-size-magnitude circuits*. A $+,-,\times$ arithmetic circuit family is feasible-size-magnitude if the n^{th} member is polynomial size and its output over the interval $[-2^n, +2^n]$ can be computed without generating any intermediate values with magnitude exceeding $2^{n^{O(1)}}$.

A.10.2 Fixed Points of Contraction Mappings (FPCM)

Given: An NC real function C that behaves as a contractor on some interval I (with integer endpoints) contained in $(-\infty, +\infty)$. The end points of I are specified as integers. A real function f is in NC if an approximation to $f(x)$ with absolute error $< 2^{-n}$, for $x \in [-2^n, +2^n]$, can be computed in NC (with the same input/output conventions as for RealCVP).
Problem: Compute the fixed point of C in I with absolute error $< 2^{-n}$.
Reference: [Hoo91]
Hint: Same basic technique of RealCVP, but the function C evaluates the circuit level by level, thus converging to a fixed point which encodes the final state of the circuit. Finding

the fixed point is in P since each iteration of the contraction mapping reduces the width of the interval by some constant factor.
Remarks: This provides an argument that fast numerical methods based on fixed points probably have to use contraction maps with better than linear rates of convergence, such as Newton's method.

A.10.3 Inverting a One-to-One Real Functions (IOORF)

Given: An NC real function f on $[0,1]$, that is increasing, and has the property that $f(0) < 0 < f(1)$. Thus there is a unique root x_0 such that $f(x_0) = 0$. A real function f is in NC if an approximation to $f(x)$ with error $< 2^{-n}$, for $x \in [-2^n, +2^n]$, can be computed in NC.
Problem: Compute x_0 with error $< 2^{-n}$.
Reference: [Ko90]
Hint: Map intermediate configurations of a log space DTM onto the real line.
Remarks: This problem was expressed originally in terms of log space computability and reductions — if f is log space computable then $f^{-1}(0)$ is not log space computable unless $DLOG = P$. The problem remains hard even if f is required to be differentiable.

A.11 Miscellaneous

A.11.1 General Deadlock Detection (GDD)

Given: A multigraph $D = (V, E)$ with $V = \pi \cup \rho$, where $\pi = \{p_1, \ldots, p_n\}$ is the set of *process nodes* and $\rho = \{r_1, \ldots, r_m\}$ is the set of *resource nodes*; and a set $T = \{t_1, \ldots, t_m\}$, where t_i denotes the number of units of r_i. The bipartite multigraph D represents the state of the system. The edges in V are of the form (p_i, r_j) denoting a *request* of process p_i for resource r_j or of the form (r_j, p_i) denoting an allocation of resource r_j to process p_i.
Problem: Is D a *deadlock state*? A deadlock state is one in which there is a non-trivial subset of the processes that cannot change state and will never change state in the future.
Reference: [Spi87]
Hint: The reduction is from MCVP. Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ denote an instance of MCVP. There will be one process p_i associated with each α_i and one additional special process p_0. The representation of gates is as follows:

1. If α_i is an input with a FALSE (TRUE) value then the edge (p_i, rf_i) $((rt_i, p_i))$ is in D.
2. Suppose $\alpha_{i_1} = \alpha_{i_2} = \alpha_j$ AND α_k, where $j, k < i_1, i_2$ (the AND gate has fanout 2). Then edges $(p_{i_1}, r_{i_1 j})$, $(p_{i_1}, r_{i_1 k})$, $(r_{i_1 j}, p_j)$, and $(r_{i_1 k}, p_k)$ are added to D for α_{i_1}, and edges $(p_{i_2}, r_{i_2 j})$, $(p_{i_2}, r_{i_2 k})$, $(r_{i_2 j}, p_j)$, and $(r_{i_2 k}, p_k)$ are added to D for α_{i_2}.
3. Suppose $\alpha_{i_1} = \alpha_{i_2} = \alpha_j$ OR α_k, where $j, k < i_1, i_2$ (the OR gate has fanout 2). Then edges $(p_{i_1}, r_{i_1 jk})$, $(r_{i_1 jk}, p_j)$, and $(r_{i_1 jk}, p_k)$ are added to D for α_{i_1}, and edges $(p_{i_2}, r_{i_2 jk})$, $(r_{i_2 jk}, p_j)$, and $(r_{i_2 jk}, p_k)$ are added to D for α_{i_2}.

Edges are also added for the special process p_0 as follows: for $1 \leq i \leq n$ add (rf_i, p_0) and (p_0, rt_i), and for $1 \leq j, k \leq n-1$ add (p_0, r_{nj}) and (p_0, r_{njk}). The graph D is not in a deadlock state if and only if α_n is TRUE.
Remarks: Notice, in the reduction the maximum number of units of any resource is two. The problem is in NC if $t_i = 1$ for all i [Spi87]. That is, if there is only one unit of each resource. If the system states are *expedient*, the resource allocator satisfies them as soon as possible, and at most one request can be connected to any process at a given time then the problem is in NC [Spi87].

A.11.2 Two Player Game (GAME)

Given: A two player game $G = (P_1, P_2, W_0, s, M)$ defined by $P_1 \cap P_2 = \emptyset, W_0 \subseteq P_1 \cup P_2$, $s \in P_1$, and $M \subseteq P_1 \times P_2 \cup P_2 \times P_1$. P_i is the set of positions in which it is player i's turn to move. W_0 is the set of immediate *winning positions* (defined below) for player one, and s is the starting position. M is the set of allowable moves; if $(p, q) \in M$ and $p \in P_1$ (or P_2) then player one (or two) may move from position p to position q in a single step. A position x is winning for player one if and only if $x \in W_0$, or $x \in P_1$ and $(x, y) \in M$ for some winning position y, or $x \in P_2$ and y is winning for every move (x, y) in M.
Problem: Is s a winning position for the first player?
Reference: [JL76]
Hint: Reduce AM2CVP to GAME. OR gates in the circuit correspond to winning positions for player one, AND gates to winning positions for player two. W_0 is the set of all inputs having value TRUE and s is the output. s is winning if and only if the output of the circuit is TRUE.
Remarks: The original reduction by Jones and Laaser was from GEN to GAME [JL76]. Given $W, \bullet, V$, and w construct the game $G = (W, W \times W, V, w, M)$, where the allowable moves are given by

$$M = \{(p, (q, r)) \mid q \bullet r = p\} \cup \{((p, q), p) \mid p, q \in W\} \cup \{((p, q), q) \mid p, q \in W\}.$$

Player one attempts to prove that a node p is generated by V. They do this by exhibiting two elements q and r, also claimed to be generated by V, such that $p = q \bullet r$. Player two attempts to exhibit an element of the pair that is not generated by V. Since GAME is an instance of AND/OR graph solvability it follows that determining whether an AND/OR graph has a solution is also P-complete [Kas86].

A.11.3 Cat & Mouse (CM)

Given: A directed graph $G = (V, E)$ with three distinguished nodes c, m, and g.
Problem: Does the mouse have a *winning strategy* in the game? The game is played as follows. The cat starts on node c, the mouse on node m and g represents the goal node. The cat and mouse alternate moves with the mouse moving first. Each move consists of following a directed edge in the graph. Either player has the option to pass by remaining on the same node. The cat is not allowed to occupy the goal node. The mouse wins if it reaches the goal node without being caught. The cat wins if the mouse and cat occupy the

same node.
Reference: [CS76a, Sto84]
Hint: The reduction is from a log space alternating Turing machine M. Assume that M starts in an existential configuration I, has a unique accepting configuration A that is existential, and each existential (universal) configuration has exactly two immediate successors, both of which are universal (existential). A directed graph is constructed with the number of nodes in the graph proportional to the number of configurations of M. We illustrate only how existential configurations can be simulated and do not account for the cat or mouse being able to pass on a move. There are two copies of each configuration C, denoted C and C', in the graph we construct. The graph has additional nodes as well. Consider an existential configuration X of M with its two succeeding universal configurations Y and Z. Assume that the cat is on X, the mouse is on X' and it is the mouse's turn to move. X is directly connected to Y and Z, whereas X' is connected to two intermediate nodes Y_1 and Y_2. Y_1 (Z_1) is connected to Y' (Z') and Y_2 (Z_2). Both Y_2 and Z_2 are connected to g. The mouse simulates an existential move of M by moving to either Y_1 of Z_1. If the mouse moves to Y_1 (Z_1) then the cat must move to Y (Z). Otherwise, the mouse's next move can be to Y_2 (Z_2) and from there onto g uncontested. From Y_1 (Z_1) the mouse must move to Y' (Z') and an universal move is ready to be simulated. The simulation of universal moves is fairly similar with the cat moving first. The game starts with the cat on I and the mouse on I'. There is an edge from A' to g. M will accept its input if and only if the mouse has a winning strategy.

A.11.4 Longcake (LONGCAKE)

Given: Two players H and V, a token, and an $m \times n$ Boolean matrix M. We describe how the game is played. Initially, the token is on placed on position m_{11} of M, it is H's turn to move, and the *current submatrix* is M. The term current submatrix denotes the portion of M that the game is currently being played on. H's turn consists of moving the token horizontally within the current submatrix to some entry $m_{ij} = 1$. At this point, either all columns to the left of j or all columns to the right of j are removed from the current submatrix, depending on which causes fewer columns to be removed. Notice, the token occupies a corner of the current submatrix again. V's turn is similar except V moves vertically and rows are removed. The first player with no moves left loses.
Problem: Does H have a winning strategy on M?
Reference: [CT90]
Hint: The reduction is performed in two stages. First, MCVP is reduced to an acyclic version of Schaefer's Geography game [Sch78] called ACYCLICGEO. This proves that ACYCLICGEO is P-complete. Second, ACYCLICGEO is reduced to LONGCAKE.
Remarks: The game SHORTCAKE is the same as LONGCAKE except the larger portion of the current submatrix is thrown away. SHORTCAKE is complete for AC^1 [CT90]. Another variant of these games called SEMICAKE is complete for $LOGCFL = SAC^1$ [CT90].

B Open Problems

This section contains a list of open problems. The goal is to find an NC algorithm or a P-completeness proof for the problem. Many of these open questions are stated as computation problems to be as general as possible. The problems listed are divided into the following categories: algebraic problems, graph theoretic problems, and miscellaneous problems.

B.1 Algebraic Problems

B.1.1 Integer Greatest Common Divisor (IntegerGCD)

Given: Two n-bit positive integers a and b.
Problem: Compute $\gcd(a, b)$.
Reference: [Gat84]
Remarks: For n^{th} degree polynomials $p, q \in Q[x]$, computing $\gcd(p, q)$ is in NC^2 via an NC^1 reduction to determinant [CS76b, BCP83]. IntegerGCD is NC^1 reducible to short vectors dimension 2 (Problem B.1.7) [Gat84].

B.1.2 Extended Euclidean Algorithm (ExtendedGCD)

Given: Two n-bit positive integers a and b.
Problem: Compute integers s and t such that $as + bt = \gcd(a, b)$.
Reference: [BGH82]
Remarks: The analogous problem for n^{th} degree polynomials is in NC [BGH82].

B.1.3 Relative Primeness (RelPrime)

Given: Two n-bit positive integers a and b.
Problem: Are a and b relatively prime?
Reference: [Tom83]
Remarks: Is a special case of IntegerGCD, Problem B.1.1.

B.1.4 Modular Inversion (ModInverse)

Given: An n-bit prime p and an n-bit positive integer a, such that p does not divide a.
Problem: Compute b such that $ab \bmod p = 1$.
Reference: [Tom83]
Remarks: ModInverse is reducible to ExtendedGCD (compute s, t such that $as + pt = gcd(a, p) = 1$), and also to ModPower, even restricted to prime moduli (compute $a^{p-2} \bmod p$ and apply Fermat's Little Theorem).

B.1.5 Modular Powering (ModPower)

Given: Positive n-bit integers a, b, and c.
Problem: Compute $a^b \bmod c$.

Reference: [Coo85]
Remarks: The complexity of the problem is open even for finding a single bit of the desired output. The problem is in NC for "smooth" c, that is, for c having only small prime factors [Gat87]. In the interesting special case when c is a prime (the antithesis of smooth) the problem is open. ModInverse, Problem B.1.4 is reducible to this restriced case. The analogous problems when a and c are n^{th} degree polynomials over a finite field of small characteristic, *modular polynomial exponentiation* and *polynomial exponentiation*, are in NC^2 [FT88]. They are open for finite fields having large (superpolynomial) characteristic. Note that ModPower can be reduced to the polynomial version with exponential characteristic, simply by considering degree 0 polynomials.

B.1.6 Short Vectors (SV)

Given: Input vectors $a_1, \ldots, a_n \in Z^n$ that are linearly independent over Q.
Problem: Find a nonzero vector x in the Z-module (or "lattice") $M = \sum a_i Z \subseteq Z^n$ such that $\|x\| \leq 2^{(n-1)/2}\|y\|$ for all $y \in M - \left\{\vec{0}\right\}$, where $\|y\| = \left(\sum y_i^2\right)^{1/2}$ is the L_2 norm.
Reference: [Gat84]
Remarks: Lenstra, Lenstra, and Lovász [LLL82] show that the problem is in P. IntegerGCD, Problem B.1.1, is NC^1 reducible to SV [Gat84].

B.1.7 Short Vectors Dimension 2 (SV2)

Given: Input vectors $a_1, a_2 \in Z^2$ that are linearly independent over Q, and an arbitrary number $c \geq 1$.
Problem: Find a nonzero vector x such that $\|x\| \leq c\|y\|$ for all $y \in M - \left\{\vec{0}\right\}$.
Reference: [Gat84]
Remarks: IntegerGCD, Problem B.1.1, is NC^1 reducible to SV2 [Gat84].

B.1.8 Sylow Subgroups (SylowSub)

Given: A group G.
Problem: Find the Sylow subgroups of G.
Reference: [BSL87]
Remarks: The problem is known to be in P [Kan85], however, the NC question is open even for solvable groups [BSL87]. For a permutation group G, testing membership in G, finding the order of G, finding the center of G, and finding a composition series of G are all known to be in NC [BSL87]. Babai, Seres, and Luks present several other open questions involving group theory [BSL87].

B.1.9 Two Variable Linear Programming (TVLP)

Given: A linear system of inequalities $Ax \leq b$ over the rationals, where each row of A has at most two nonzero elements.
Problem: Find a feasible solution if one exists.
Reference: [LMR86]

Remarks: There is a known poly-log algorithm that uses $n^{\log^k n}$ processors on a CREW-PRAM [LMR86].

B.1.10 Univariate Polynomial Factorization over Q (UPFQ)

Given: An n^{th} degree polynomial $p \in Q[x]$.
Problem: Compute the factorization of p over Q.
Reference: [Gat84]
Remarks: UPFQ is NC^1 reducible to short vectors (Problem B.1.6) [Gat84].

B.2 Graph Theory Problems

B.2.1 Edge-Weighted Matching (EWM)

Given: A graph G with positive integer weights on its edges.
Problem: Find a matching of maximum weight.
Reference: [KUW86]
Remarks: EWM is in RNC (but not known to be in NC) if all weights are polynomially bounded [KUW86]. LFMM, Problem B.4.2 (and hence all of CC), is NC reducible to EWM, by assigning weight $2^{\mathrm{rank}(e)}$ to edge e.

B.2.2 Bounded Degree Graph Isomorphism (BDGI)

Given: Two graphs G and H. The vertices in G and H have maximum degree at most k, a constant independent of the sizes of G and H.
Problem: Are G and H isomorphic?
Remarks: Luks showed the problem is in P [FHL80]. Without the degree bound, the problem is in NP but not known to be in P, nor is it known to be either P-hard or NP-complete.

B.2.3 Graph Closure (GC)

Given: An undirected graph $G = (V, E)$ and a designated edge $e = (u, v)$.
Problem: Is e in the *closure* of G? That is, the graph obtained from G by repeatedly joining non-adjacent pairs of vertices u and v whose degree sum is at least $|V|$.
Reference: [Khu89]
Remarks: With the following modification, the problem becomes P-complete (Problem A.2.16): add a set of designated edges E' such that only vertices whose degree sum is at least $|V|$ and whose corresponding edge is in E' may be added to the closure [Khu89].

B.2.4 Restricted Lexicographically First Independent Set (RLFMIS)

Given: A planar, bipartite graph G with a numbering on the vertices.
Problem: Find the lexicographically first maximal independent set.
Reference: [Miy89]

Remarks: See Problem A.2.1. Finding the lexicographically first maximal subgraph of maximum degree one in planar, bipartite graphs of degree at most three is P-complete [Miy89].

B.2.5 Maximal Path (MP)

Given: A graph G with a numbering on the vertices and a designated vertex r.
Problem: Find a *maximal path* originating from r. That is, a path that cannot be extended without encountering a node already on the path.
Reference: [AM87b]
Remarks: The lexicographically first maximal path problem, Problem A.3.1, is P-complete even when restricted to planar graphs with maximum degree three. If the maximum degree of any vertex in G is at most Δ then there is an algorithm that can find a maximal path in $O(\Delta \log^3 n)$ time using n^2 processors [AM87b]. There is also an NC algorithm for finding a maximal path in planar graphs [AM87b].

B.2.6 Maximal Independent Set Hypergraph (MISH)

Given: A finite collection $C = \{C_1, \ldots, C_n\}$ of finite sets.
Problem: Find a subcollection $D = D_1, \ldots, D_k$ of pairwise disjoint members of C such that for every set $S \in C - D$ there is a $T \in D$, such that $S \cap T \neq \emptyset$.
Reference: [BL90, KR90]
Remarks: If C is a collection of two element sets ("dimension 2") then the problem becomes the maximal independent set problem, which is known to be in NC (see Problem A.2.1). Beame and Luby give an RNC algorithm for dimension $O(1)$, as well as an algorithm for the general problem that is conjectured to be RNC.

B.3 Geometric Problems

B.3.1 2-Oriented Weighted Planar Partitioning (2OWPP)

Given: A set of non-intersecting segments $s_1, \ldots, s_n$ in the plane, a set of associated integer weights $w_1, \ldots, w_n$, and two designated segments r and t. The segments are *2-oriented* meaning that there are only two different possible slopes for the segments.
Problem: Do segments r and t "touch" in the partitioning of the plane constructed by extending segments in the order of their weights? Segments are extended until they reach another segment or a previous segment extension.
Reference: [ACG90]
Remarks: The 3-oriented version, Problem A.9.2, in which three different slopes are allowed, is P-complete [ACG90].

B.3.2 Limited Reflection Ray Tracing (LRRT)

Given: A set of n flat mirrors of lengths $l_1, \ldots, l_n$; their placements at rational points in the plane; a source point S; the trajectory of a single beam emitted from S; and a designated

mirror M.
Problem: Determine if M is hit by the beam within n reflections. At the mirrors the angle of incident of the beam equals the angle of reflection.
Reference: [GHR91]
Remarks: The general ray tracing problem is to determine if a mirror is ever hit by the beam. When the mirrors are points, that is have no length, the general problem is in *NC* [dlTG90]. In two or more dimensions, the general problem is in *PSPACE*. In three dimensions, with mirrors placed at rational points, the general problem is *PSPACE*-hard. The general problem is open for all mirrors of a fixed size as well. See [RTY90] for a detailed discussion.

B.3.3 Restricted Plane Sweep Triangulation (SWEEP)

Given: An n vertex polygon P without holes.
Problem: Find the triangulation computed by the *plane sweep triangulation algorithm.*
Reference: [ACG90]
Remarks: The problem of finding some arbitrary triangulation is in *NC* [Goo89]. If the polygon is allowed to have holes then the problem is P-complete [ACG90]. See Problem A.9.1

B.3.4 Successive Convex Hulls (SCH)

Given: A set S of n points in dimension d and a designated point p.
Problem: Determine if p is in the i^{th} remaining convex hull that is formed by repeatedly finding and removing convex hulls from S.
Reference: [Cha85, Ata87, Cha89]
Remarks: The problem is open for two dimensions and up.

B.3.5 Unit Length Visibility Layers (ULVL)

Given: A set of n unit length, horizontal, non-intersecting line segments in the plane, a designated segment s, and an integer d.
Problem: Is the label assigned to segment s by the *visibility layering process* d? The visibility layering process is repeatedly to compute and delete the *upper envelope* of the remaining set of segments and label those segments with the current depth. The upper envelope consists of those segments visible from the point $(0, +\infty)$.
Reference: [ACG90]
Remarks: The problem is P-complete if the restriction on unit lengths is removed [ACG90]. See Problem A.9.3.

B.4 *CC* Problems

Several reseachers independently suggested looking at the complexity of the Comparator Circuit Value Problem, or CCVP (Problem B.4.1). *CC* is the class of problems reducible to CCVP (Mayr and Subramanian [MS89]). Problems in this section are all equivalent to, i.e. NC interreducible with, CCVP. While the evidence is less compelling than that for RNC problems (Section B.5), it is generally considered unlikely that these problems are P-complete, because of the lack of fanout in comparator circuits. On the other hand, no fast parallel algorithms are known for them, with the partial exception of $\sqrt{n}\log^{O(1)} n$ algorithms for CCVP and some related problems. (Such algorithms were independently discovered by D. Soroker, unpublished, and by Mayr and Subramanian [MS89]. Richard Anderson, also unpublished, has improved the algorithms so as to use only $\sqrt{n}$ processors. Mayr and Subramanian note that these algorithms are P-complete, in the sense of Section **??**.) It is known that $NL \subseteq CC \subseteq P$ [MS89].

Most of the results described in this subsection come from [MS89]. See also [Sub90, Sub89, Fed89].

B.4.1 Comparator Circuit Value Problem (CCVP)

Given: An encoding $\overline{\alpha}$ of a circuit α composed of comparator gates plus inputs $x_1, \ldots, x_n$. A *comparator* gate outputs the minimum of its two inputs on its first output wire and outputs the maximum of its two inputs on its second output wire. The gate is further restricted so that each output has fanout at most one.
Problem: Does α on input $x_1, \ldots, x_n$ output 1?
Reference: [Coo, MS89]
Remarks: Cook shows that CCVP is NC equivalent to computing the lexicographically first maximal matching in a bipartite graph. Mayr and Subramanian [MS89] show that this problem is NC equivalent to stable marriage (Problem B.4.3).

B.4.2 Lexicographically First Maximal Matching (LFMM)

Given: A graph G with an ordering on its edges.
Problem: Find the lexicographically first maximal matching.
Reference: [Coo, MS89]
Remarks: LFMM is NC equivalent to Comparator CVP (Problem B.4.1) [MS89]. This problem resembles the lexicographically first maximal independent set problem, Problem A.2.1, known to be P-complete. A P-completeness proof for LFMM would imply that edge-weighted matching (Problem B.2.1) is also P-complete.

B.4.3 Stable Marriage (SM)

Given: For each member of a community of n men and n women, a ranking of the opposite sex according to their preference for a marriage partner.
Problem: Find n marriages such that the set of marriages is stable. The set is unstable if a man and woman exist who are not married to each other but prefer each other to their actual mates.
Reference: [MS89]
Remarks: See, e.g., [Gib85] for background on the Stable Marriage problem. SM is NC equivalent to Comparator CVP (Problem B.4.1) [MS89]. Several variations on Stable Marriage are also known to be equivalent to CCVP; see [MS89].

B.5 *RNC* Problems

The problems in this subsection are all known to be in *RNC* or *FRNC*, but not known to be in *NC*. A proof that any of them is *P*-complete would be almost as unexpected as a proof that $NC = P$. See Section 7.6 for more discussion.

B.5.1 Directed or Undirected Depth First Search (DFS)

Given: A graph G and a vertex s.
Problem: Construct the depth first search numbering of G starting from vertex s.
Reference: [AA88, AAK90]
Remarks: *RNC* algorithms are now known for both the undirected [AA88] and directed [AAK90] cases, subsuming earlier *RNC* results for planar graphs [Smi86]. For DAGs, DFS is in *NC* [Gre88a].

B.5.2 0-1 Maximum Flow (0-1 MaxFlow)

Given: Directed graph G with each edge labeled in unary with a capacity $c_i \geq 0$ and two distinguished vertices, source s and sink t.
Problem: Find a maximum flow.
Reference: [Fea84, KUW86]
Remarks: [Fea84] shows the problem of finding the *value* of the maximum flow to be in *RNC*. [KUW86] show how to construct a maximum flow, also in *RNC*. Both problems remain in *RNC* when capacities are polynomially bounded, but are *P*-complete when capacities are arbitrary (Problem A.4.4). Venkateswaran [Ven90] shows that 0-1 MaxFlow is hard for AC^1.

B.5.3 Maximum Matching (MM)

Given: A graph G.
Problem: Find a maximum matching of G. A *matching* is a subset of the edges of G such that no two edges of the subset are adjacent in G. A matching is a *maximum* matching if no matching of larger cardinality exists.
Reference: [Fea84, KUW86, MVV87]
Remarks: Feather [Fea84] shows that the problem of finding the *size* of maximum matching is in *RNC*. Karp, Upfal, and Wigderson [KUW86] gave the first *RNC* algorithm for *finding* the maximum matching. A more efficient algorithm was given by Mulmuley, Vazirani, and Vazirani [MVV87]. Karloff shows how any *RNC* algorithm for matching can be made errorless [Kar86]. Maximum edge-weighted matching for unary edge weights and maximum vertex-weighted matching for binary vertex weights are also known to be in *RNC* [KUW86, MVV87].

B.5.4 Perfect Matching Existence (PME)

Given: A graph G.

Problem: Does G have a perfect matching? A *perfect matching* is a matching where each vertex is incident to one edge in the matching.
Reference: [KUW86, MVV87]
Remarks: See remarks for Problem B.5.3. PME seems to be the simplest of the matching problems not known to be in NC.

C Undigested Papers

Listed below are a few papers we are aware of, but have not yet added to the main problem lists. They will appear in the final version.

[KKS91, KS88]

References

[BGS90] J. L. Balcázar, J. Gabarró, and M. Santha. Deciding bisimilarity is P-complete. Technical Report LSI-90-25, Universitat Politècnica de Catalunya, 1990.

[Den84] L. Denenberg. *Computational Complexity of Logical Problems.* PhD thesis, Harvard University, 1984.

[DL84] L. Denenberg and H. R. Lewis. The complexity of the satisfiablility problem for Korn formulas. *Theoretical Computer Science*, 30(3):319–341, 1984.

[God87] G. Godbeer. The computational complexity of the stable configuration problem for connectionist models. Master's thesis, University of Toronto, 1987.

[GR88] A. M. Gibbons and W. Rytter. *Efficient Parallel Algorithms.* Cambridge University Press, Cambridge, UK, 1988.

[Lin91] Y. Lin. *Parallel Computational Methods in Integer Linear Programming.* PhD thesis, The City University of New York, 1991.

[Lip87] J. Lipscomb. On the computational complexity of finding a connectionist model's stable state vectors. Master's thesis, University of Toronto, 1987.

[LKPar] Y. Lin-Kriz and V. Pan. On the parallel complexity of integer linear programming, GCD and the iterated mod function. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, to appear.

[SH90] R. Sarnath and X. He. A P-complete graph partition problem. *Theoretical Computer Science*, 76:343–351, 1990.

[Ste91a] I.A. Stewart. On a greedy heuristic for finding small dominating sets. Technical report, University of Newcastle upon Tyne, 1991. Submitted for publication.

[Ste91b] I.A. Stewart. On two approximation algorithms for the clique problem. Technical report, University of Newcastle upon Tyne, 1991.

References

[AA88] A. Aggarwal and R. Anderson. A random *NC* algorithm for depth first search. *Combinatorica*, 8(1):1–12, 1988.

[AAK90] A. Aggarwal, R. Anderson, and M. Kao. Parallel depth-first search in general directed graphs. *SIAM Journal on Computing*, 19(2):397–409, 1990.

[ABI86] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986.

[ACG90] M. J. Atallah, P. Callahan, and M. T. Goodrich. *P*-complete geometric problems. In *Proceedings of the 1990 ACM Symposium on Parallel Algorithms and Architectures*, pages 317–326, Crete, Greece, July 1990.

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[All89] Eric W. Allender. *P*-uniform circuit complexity. *Journal of the ACM*, 36(4):912–928, October 1989.

[AM84a] R. J. Anderson and E. Mayr. Parallelism and greedy algorithms. Technical Report STAN-CS-84-1003, Computer Science Department, Stanford University, April 1984.

[AM84b] J. Avenhaus and K. Madlener. The Nielsen reduction and *P*-complete problems in free groups. *Theoretical Computer Science*, 32(1,2):61–76, 1984.

[AM84c] J. Avenhaus and K. Madlener. On the complexity of intersection and conjugacy in free groups. *Theoretical Computer Science*, 32(3):279–295, 1984.

[AM87a] R. Anderson and E. Mayr. Parallelism and greedy algorithms. In *Advances in Computing Research*, volume 4, pages 17–38. JAI Press, 1987. Also Stanford Technical Report STAN-CS-84-1003, 1984.

[AM87b] R. Anderson and E. Mayr. Parallelism and the maximal path problem. *Information Processing Letters*, 24(2):121–126, 1987.

[AMW89] R. Anderson, E. Mayr, and M. Warmuth. Parallel approximation algorithms for bin packing. *Information and Computation*, 82(3):262–277, 1989.

[And85] R. J. Anderson. *The Complexity of Parallel Algorithms*. PhD thesis, Stanford University, 1985. Stanford Computer Science Department TR STAN-CS-86-1092.

[And87] R. J. Anderson. A parallel algorithm for the maximal path problem. *Combinatorica*, 7(4):315–326, 1987.

[And88] Richard Anderson. Reduction of lexicographic depth-first search to breadth-depth search. Private communication, 1988.

[AP87] F. Afrati and C. H. Papadimitriou. The parallel complexity of simple chain queries. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Database Systems*, pages 210–213, 1987.

[Ata87] M. J. Atallah. Posed the successive convex hull problem. Cited in [ACG90], 1987.

[BBMS88] A. Bertoni, M. C. Bollina, G. Mauri, and N. Sabadini. On characterizing classes of efficiently parallelizable problems. In *Proceedings of the International Workshop on Parallel Computing and VLSI*, pages 13–26. VLSI: Algorithms and Architectures, North-Holland, Amsterdam, 1988.

[BCMV83] B. von Braunmühl, S. A. Cook, K. Mehlhorn, and R. Verbeek. The recognition of deterministic CFL's in small time and space. *Information and Control*, 56(1-2):34–51, January/February 1983.

[BCP83] A. Borodin, S. Cook, and N. Pippenger. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Information and Control*, 58(1-3):113–136, 1983.

[BDAP88] D. P. Bovet, S. De Agostino, and R. Petreschi. Parallelism and the feedback vertex set problem. *Information Processing Letters*, 28(2):81–85, June 1988.

[Ber91] Bonnie Berger. The fourth moment method. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 373–383. ACM, 1991.

[BGH82] A. Borodin, J. von zur Gathen, and J. E. Hopcroft. Fast parallel matrix and GCD computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 65–71, Chicago, IL, November 1982. IEEE.

[BHC86] P. W. Beame, H. J. Hoover, and S. A. Cook. Log depth circuits for division and related problems. *SIAM Journal on Computing*, 15(4):994–1003, November 1986.

[BIS90] D. A. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274–306, December 1990.

[BK86] J. Boyar and H. Karloff. Coloring planar graphs in parallel. Manuscript, 1986.

[BL90] Paul Beame and Michael Luby. Parallel search for maximal independence given minimal dependence. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 212–218. ACM, 1990.

[BM76] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. MacMillan, 1976. Revised paperback edition, 1977.

[Bor77] A. Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6(4):733–744, December 1977.

[Bor82] Allan Borodin. Structured vs. general models in computational complexity. *L'Enseignement Mathématique*, XXVIII(3-4):171–190, July-December 1982. Also in [L'E82, pages 47–65].

[BS90] B. Berger and P. Shor. Approximation algorithms for the maximum acyclic subgraph problem. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 236–243. ACM, 1990.

[BSL87] L. Babai, S. Seres, and E. M. Luks. Permutation groups in *NC*. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 409–420, New York, NY, May 1987.

[Cha85] B. Chazelle. On the convex layers of a planar set. *IEEE Transactions on Information Theory*, IT-31(4):509–517, 1985.

[Cha89] B. Chazelle. Posed the successive convex hull problem. Cited in [ACG90], 1989.

[CKS81] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981.

[CM89] J. Cheriyan and S. N. Maheshwari. The parallel complexity of finding a blocking flow in a 3-layer network. *Information Processing Letters*, 31(3):157–161, 1989.

[Coo] S. A. Cook. The comparator circuit value problem. personal communication.

[Coo71a] S. A. Cook. Characterizations of pushdown machines in terms of time-bounded computers. *Journal of the ACM*, 18(1):4–18, January 1971.

[Coo71b] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, OH, May 1971.

[Coo74] S. A. Cook. An observation on time-storage trade off. *Journal of Computer and System Sciences*, 9(3):308–316, December 1974.

[Coo79] S. A. Cook. Deterministic CFL's are accepted simultaneously in polynomial time and log squared space. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 338–345, Atlanta, GA, April-May 1979.

[Coo81] S. A. Cook. Towards a complexity theory of synchronous parallel computation. *L'Enseignement Mathématique*, XXVII(1–2):99–124, January-June 1981. Also in [L'E82, pages 75–100].

[Coo82] S. A. Cook. Reduction of the circuit value problem to feasible linear programming. Private communication, 1982.

[Coo85] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(1–3):2–22, January/February/March 1985.

[Cos88] S. S. Cosmadakis. The word and generator problems for lattices. *Information and Computation*, 77(3):192–217, 1988.

[CS76a] A. K. Chandra and L. J. Stockmeyer. Alternation. In *17th Annual Symposium on Foundations of Computer Science*, pages 98–108, Houston, TX, October 1976. IEEE. Preliminary Version.

[CS76b] S. A. Cook and R. Sethi. Storage requirements for deterministic polynomial time recognizable languages. *Journal of Computer and System Sciences*, 13(1):25–37, 1976.

[CT90] Ashok K. Chandra and Martin Tompa. The complexity of short two-person games. *Discrete Applied Mathematics*, 29:21–33, 1990.

[DC80] P. W. Dymond and S. A. Cook. Hardware complexity and parallel computation. In *21st Annual Symposium on Foundations of Computer Science*, pages 360–372, Syracuse, NY, October 1980. IEEE.

[DC89] P. W. Dymond and S. A. Cook. Complexity theory of parallel time and hardware. *Information and Computation*, 80(3):205–226, March 1989.

[DK91] Arthur L. Delcher and S. Rao Kosaraju. An $\mathcal{NC}$ algorithm for evaluating monotone planar circuits. manuscript, 1991.

[DKM84] C. Dwork, P. C. Kanellakis, and J. C. Mitchell. On the sequential nature of unification. *Journal of Logic Programming*, 1:35–50, 1984.

[DKS88] C. Dwork, P. C. Kanellakis, and L. J. Stockmeyer. Parallel algorithms for term matching. *SIAM Journal on Computing*, 17(4):711–731, 1988.

[DLR79] D. Dobkin, R. J. Lipton, and S. Reiss. Linear programming is log-space hard for *P*. *Information Processing Letters*, 8(2):96–97, February 1979.

[dlTG90] P. de la Torre and R. Greenlaw. Discrete ray-tracing is in *NC*, 1990. Personal communication.

[dlTK88] P. de la Torre and C. Kruskal. Fast parallel algorithms for all sources lexicographic search and path finding problems. Manuscript (appeared as University of Maryland technical report CS-TR 2283, 1989), 1988.

[dlTK89] P. de la Torre and C. Kruskal. Fast and efficient parallel algorithms for single source lexicographic depth-first search, breadth-first search and topological-first search. Manuscript, 1989.

[DR80] D. P. Dobkin. and S. Reiss. The complexity of linear programming. *Theoretical Computer Science*, 11(1):1–18, 1980.

[DT85] Patrick W. Dymond and Martin Tompa. Speedups of deterministic machines by synchronous parallel machines. *Journal of Computer and System Sciences*, 30(2):149–161, April 1985.

[Dym80] P. W. Dymond. *Simultaneous Resource Bounds and Parallel Computation.* PhD thesis, University of Toronto, August 1980. Technical Report 145/80.

[Fea84] T. Feather. The parallel complexity of some flow and matching problems. Master's thesis, Department of Computer Science, University of Toronto, 1984. TR 174/84.

[Fed89] T. Feder. A new fixed point approach to stable networks and stable marriages. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, Seattle, WA, May 1989.

[FF62] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[FHL80] M. Furst, J. Hopcroft, and E. Luks. Polynomial time algorithms for permutation groups. In *21st Annual Symposium on Foundations of Computer Science*, pages 36–41, Syracuse, NY, October 1980. IEEE.

[Fic91] Faith E. Fich. The parallel random access machine. In John H. Reif, editor, *Synthesis of Parallel Algorithms.* Morgan Kaufman, San Mateo, CA, 1991. To appear.

[FT88] Faith E. Fich and Martin Tompa. The parallel complexity of exponentiating polynomials over finite fields. *Journal of the ACM*, 35(3):651–667, July 1988.

[FW78] S. Fortune and J. Wyllie. Parallelism in random access machines. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pages 114–118, San Diego, CA, May 1978.

[Gal74] Zvi Galil. Two way deterministic pushdown automaton languages and some open problems in the theory of computation. In *15th Annual Symposium on Switching and Automata Theory*, pages 170–177, 1974. Published as [Gal77].

[Gal77] Zvi Galil. Some open problems in the theory of computation as questions about two-way deterministic pushdown automaton languages. *Mathematical Systems Theory*, 10:211–228, 1977.

[Gat84] J. von zur Gathen. Parallel algorithms for algebraic problems. *SIAM Journal on Computing*, 13(4):802–824, November 1984.

[Gat87] J. von zur Gathen. Computing powers in parallel. *SIAM Journal on Computing*, 16(5):930–945, October 1987.

[GHR91] R. Greenlaw, H. Hoover, and W. L. Ruzzo. A compendium of problems complete for P, 1991. This work.

[Gib85] A. Gibbons. *Algorithmic Graph Theory.* Cambridge University Press, Cambridge, 1985.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[Gol77] L. M. Goldschlager. The monotone and planar circuit value problems are log space complete for *P*. *SIGACT News*, 9(2):25–29, Summer 1977.

[Gol80] L. M. Goldschlager. A space efficient algorithm for the monotone planar circuit value problem. *Information Processing Letters*, 10(1):25–27, 1980.

[Gol81] L. M. Goldschlager. ϵ-productions in context-free grammars. *Acta Informatica*, 16(3):303–308, 1981.

[Gol82] L. M. Goldschlager. A universal interconnection pattern for parallel computers. *Journal of the ACM*, 29(4):1073–1086, October 1982.

[Goo83] G. B. Goodrich. *The Complexity of Finite Languages.* PhD thesis, University of Washington, 1983.

[Goo89] M. T. Goodrich. Triangulating a polygon in parallel. *Journal of Algorithms*, 10(3):327–351, 1989.

[GP86] L. M. Goldschlager and I. Parberry. On the construction of parallel computers from various bases of Boolean functions. *Theoretical Computer Science*, 43(1):43–58, 1986.

[Gre88a] R. Greenlaw. Parallel complexity results about greedy breadth and depth first search. Technical Report 88–07–05, Department of Computer Science, University of Washington, July 1988. Submitted for publication.

[Gre88b] Raymond Greenlaw. *The Complexity of Parallel Computations: Inherently Sequential Algorithms and P-Complete Problems.* PhD thesis, University of Washington, December 1988. Also, TR 88–12–01.

[Gre89] Raymond Greenlaw. Ordered vertex removal and subgraph problems. *Journal of Computer and System Sciences*, 39(3):323–342, December 1989.

[Gre90] R. Greenlaw. The parallel complexity of approximation algorithms for the maximum acyclic subgraph problem. Technical Report 90-61, University of New Hampshire, 1990. To appear *Mathematical Systems Theory* .

[Grear] Raymond Greenlaw. A model classifying algorithms as inherently sequential with applications to graph searching. *Information and Computation*, to appear.

[GS89] M. Goldberg and T. Spencer. A new parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 18:419–427, 1989.

[GSS82] L. M. Goldschlager, R. A. Shaw, and J. Staples. The maximum flow problem is log space complete for *P*. *Theoretical Computer Science*, 21:105–111, 1982.

[Har79] M. Harrison. *Introduction to Formal Language Theory.* Addison Wesley, 1979.

[Hås87] Johan Håstad. *Computational Limitations of Small-Depth Circuits.* ACM doctoral dissertation award, 1986. MIT Press, 1987.

[Hås88] Johan Håstad. One-way permutations in NC^0. *Information Processing Letters*, 26:153–155, 1988.

[Her90] J. Hershberger. Upper envelope onion peeling. In G. Goos and J. Hartmanis, editors, *2nd Scandinavian Workshop on Algorithm Theory*, volume 447 of *Lecture Notes in Computer Science*, pages 368–379. Springer-Verlag, Berlin, New York, 1990.

[HM86] D. Helmbold and E. Mayr. Perfect graphs and parallel algorithms. In *International Conference on Parallel Processing*, pages 853–860. IEEE Computer Society, 1986.

[HM87] D. Helmbold and E. Mayr. Fast scheduling algorithms on parallel computers. *Advances in Computing Research*, 1987.

[Hon84] J. W. Hong. Similarity and duality in computation. *Information and Control*, 62:109–128, 1984.

[Hoo90] H. J. Hoover. Feasible real functions and arithmetic circuits. *SIAM Journal on Computing*, 19(1):182–204, 1990.

[Hoo91] H. J. Hoover. Real functions, contraction mappings, and P-completeness. *Information and Computation*, 93(2):333–349, 1991.

[HS84] E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms*. Computer Science Press, Rockville, Md., 1984.

[HS87] P. Hajnal and E. Szemeredi. Brooks coloring in parallel. Manuscript, 1987.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[HY87] X. He and Y. Yesha. A nearly optimal parallel algorithm for constructing depth first spanning trees in a planar graph, 1987. Manuscript.

[Imm81] N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.

[Imm83] N. Immerman. Languages which capture complexity classes (preliminary report). In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 347–354, Boston, MA, April 1983.

[Ita78] A. Itai. Two–commodity flow. *Journal of the ACM*, 25(4):596–611, 1978.

[JL76] N. D. Jones and W. T. Laaser. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3(2):105–117, 1976.

[JPY88] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

[JV82] D. B. Johnson and S. M. Venkatesan. Parallel algorithms for minimum cuts and maximum flows in planar networks (preliminary version). In *23rd Annual Symposium on Foundations of Computer Science*, pages 244–254, Chicago, IL, November 1982. IEEE.

[Kan85] W. M. Kantor. Sylow's theorem in polynomial time. *Journal of Computer and System Sciences*, 30(3):359–394, 1985.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

[Kar84] R. M. Karp. Talk at the University of Toronto, 1984.

[Kar86] Howard Karloff. A las vegas *RNC* algorithm for maximum matching. *Combinatorica*, 6(4):387–391, 1986.

[Kas86] S. Kasif. On the parallel complexity of some constraint satisfaction problems. In *Proceedings AAAI-86*, pages 349–353, 1986.

[Kha79] L. G. Khachian. A polynomial time algorithm for linear programming. *Doklady Akademii Nauk SSSR, n.s.*, 244(5):1093–1096, 1979. English translation in *Soviet Math. Dokl. 20*, 191–194.

[Khu89] S. Khuller. On computing graph closures. *Information Processing Letters*, 31(5):249–255, 1989.

[KKS91] Dimitris Kavadias, Lefteris M. Kirousis, and Paul Spirakis. the complexity of the reliable connectivity problem. *Information Processing Letters*, 39:245–252, 1991.

[KLS89] G. A. P. Kindervater, J. K. Lenstra, and D. B. Shmoys. The parallel complexity of TSP heuristics. *Journal of Algorithms*, 10(2):249–270, 1989.

[KN88] M. Karchmer and J. Naor. A fast parallel algorithm to color a graph with Δ colors. *Journal of Algorithms*, 9(1):83–91, 1988.

[Ko90] K. Ko. Inverting a one-to-one real function is inherently sequential. In S. Buss and P. Scott, editors, *In Feasible Mathematics, A Mathematical Sciences Institute Workshop*, pages 239–257. Birkhauser, Ithaca, New York, 1990.

[Kos90] S. Rao Kosaraju. On the parallel evaluation of classes of circuits. In K. V. Nori and C. E. Veni Madhavan, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 472 of *Lecture Notes in Computer Science*, pages 232–237. Springer, 1990.

[Koz77] D. Kozen. Complexity of finitely presented algebras. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, pages 164–177, Boulder, CO, May 1977.

[KR89] H. J. Karloff and W. L. Ruzzo. The iterated mod problem. *Information and Computation*, 80(3):193–204, March 1989.

[KR90] R. M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines. In Jan van Leeuwan, editor, *Handbook of Theoretical Computer Science*, volume A: Algorithms and Complexity, chapter 17, pages 869–941. M.I.T. Press/Elsevier, 1990.

[KS88] Lefteris M. Kirousis and Paul Spirakis. Probabilistic log-space reductions and problems prababilistically hard for *p*. In *Proceedings, SWAT*, volume 318 of *Lecture Notes in Computer Science*, pages 163–175. Springer, 1988.

[KSS89] L. Kirousis, M. Serna, and P. Spirakis. The parallel complexity of the subgraph connectivity problem. In *30th Annual Symposium on Foundations of Computer Science*, pages 294–299, Research Triangle Park, NC, October 1989. IEEE.

[KUW85] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random *NC*. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 22–32, Providence, RI, May 1985.

[KUW86] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a maximum matching is in random NC. *Combinatorica*, 6(1):35–48, 1986.

[KUW88] R. M. Karp, E. Upfal, and A. Wigderson. The complexity of parallel search. *Journal of Computer and System Sciences*, 36:225–253, 1988.

[KW85] R. M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. *Journal of the ACM*, 32(4):762–773, 1985.

[Lad75] R. E. Ladner. The circuit value problem is log space complete for *P*. *SIGACT News*, 7(1):18–20, January 1975.

[Law76] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, New York, 1976.

[L'E82] *Logic and Algorithmic,* An International Symposium Held in Honor of Ernst Specker, Zürich, February 5–11, 1980. Monographie No. 30 de L'Enseignement Mathématique, Université de Genève, 1982.

[Lei92] F. Thomas Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, 1992.

[LF80] R. E. Ladner and M. J. Fischer. Parallel prefix computation. *Journal of the ACM*, 27(4):831–838, October 1980.

[LLL82] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:513–534, 1982.

[LMR86] G. S. Lueker, N. Megiddo, and V. Ramachandran. Linear programming with two variables per inequality in poly-log time. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 196–205, Berkeley, CA, May 1986.

[LP82] H. R. Lewis and C. H. Papadimitriou. Symmetric space-bounded computation. *Theoretical Computer Science*, 19:161–187, 1982.

[LR89] Tak Wah Lam and Walter L. Ruzzo. The power of parallel pointer manipulation. In *Proceedings of the 1989 ACM Symposium on Parallel Algorithms and Architectures*, pages 92–102, Santa Fe, NM, June 1989.

[LSH65] P. M. Lewis, R. E. Stearns, and J. Hartmanis. Memory bounds for recognition of context–free and context–sensitive languages. In *Proc.* 6^{th} *IEEE Symp. on Switching Theory and Logic Design*, pages 191–202. IEEE Computer Society, 1965.

[LT82] T. Lengauer and R. E. Tarjan. Asymptotically tight bounds on time-space trade-offs in a pebble game. *Journal of the ACM*, 29(4):1087–1130, October 1982.

[Lub84] M. Luby. Private communication, 1984.

[Lub86] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986.

[LW87] T. Lengauer and K. W. Wagner. The correlation between the complexities of the nonhierarchical and hierarchical versions of graph problems. In *4th Annual Symposium on Theoretical Aspects of Computer Science*, pages 100–113, 1987.

[LZ77] Richard J. Lipton and Yechezkel Zalcstein. Word problems solvable in log space. *Journal of the ACM*, 24(3):522–526, 1977.

[McC81] W. F. McColl. Planar crossovers. *IEEE Transactions on Computers*, C-30(3), 1981.

[Miy89] Satoru Miyano. The lexicographically first maximal subgraph problems: *P*-completeness and *NC* algorithms. *Mathematical Systems Theory*, 22(1):47–73, 1989.

[MR85] G. L. Miller and J. H. Reif. Parallel tree contraction and its applications. In *26th Annual Symposium on Foundations of Computer Science*, pages 478–489, Portland, OR, October 1985. IEEE.

[MRK88] G. L. Miller, V. Ramachandran, and E. Kaltofen. Efficient parallel evaluation of straight-line code and arithmetic circuits. *SIAM Journal on Computing*, 17:687–695, 1988.

[MS89] E. W. Mayr and A. Subramanian. The complexity of circuit value and network stability. In *Proceedings, Structure in Complexity Theory, Fourth Annual Conference*, pages 114–123, Eugene, OR, June 1989. IEEE. Full version appears as Stanford Technical Report STAN-CS-89-1278.

[MVV87] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[Nao87] J. Naor. A fast parallel coloring of planar graphs with five colors. *Information Processing Letters*, 25(1):51–53, 1987.

[Ofm63] Yu. P. Ofman. On the algorithmic complexity of discrete functions. *Soviet Physics Doklady*, 7:589–591, 1963.

[Pap78] C. H. Papadimitriou. Efficient search for rationals. *Information Processing Letters*, 8(1):1–4, 1978.

[Par86] Ian Parberry. Parallel speedup of sequential machines: A defense of the parallel computation thesis. *SIGACT News*, 18(1):54–67, Summer 1986.

[Par87] Ian Parberry. *Parallel Complexity Theory.* Research Notes in Theoretical Computer Science. Pitman/Wiley, 1987.

[PF79] N. Pippenger and M. J. Fischer. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, April 1979.

[Pip77] N. Pippenger. Fast simulation of combinational logic networks by machines without random-access storage. *Allerton Conference on Communication, Control and Computing*, 15:25–33, September 1977. Also available as Research Report RC6582, IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY, June 1977.

[Pip79] N. Pippenger. On simultaneous resource bounds. In *20th Annual Symposium on Foundations of Computer Science*, pages 307–311, San Juan, Puerto Rico, October 1979. IEEE.

[Pip80] N. Pippenger. Pebbling. In *Proceedings of the Fifth IBM Symposium on Mathematical Foundations of Computer Science*. IBM Japan, May 1980.

[Pip81] N. Pippenger. Pebbling with an auxiliary pushdown. *Journal of Computer and System Sciences*, 23(2):151–165, October 1981.

[Pla84] D. A. Plaisted. Complete problems in the first-order predicate calculus. *Journal of Computer and System Sciences*, 29(1):8–35, 1984.

[Pos41] E. Post. *The Two-Valued Iterative Systems of Mathematical Logic.* Number 5 in Annals of Math. Studies. Princeton University Press, 1941.

[PS76] V. R. Pratt and L. J. Stockmeyer. A characterization of the power of vector machines. *Journal of Computer and System Sciences*, 12:198–221, 1976.

[PSY90] C. H. Papadimitriou, A. A. Schäffer, and M. Yannakakis. On the complexity of local search. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 438–445, Baltimore, MD, May 1990.

[PT87] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[PTC77a] W. J. Paul, R. E. Tarjan, and J. R. Celoni. Correction to "Space bounds for a game on graphs". *Mathematical Systems Theory*, 11:85, 1977.

[PTC77b] W. J. Paul, R. E. Tarjan, and J. R. Celoni. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977. See also [PTC77a].

[PV76] M. S. Paterson and L. G. Valiant. Circuit size is nonlinear in depth. *Theoretical Computer Science*, 2:397–400, 1976.

[Ram87] V. Ramachandran. The complexity of minimum cut and maximum flow problems in an acyclic network. *Networks*, 17(4):387–392, 1987.

[Ram88] V. Ramachandran. Fast and processor-efficient parallel algorithms for reducible flow graphs. Technical Report UILU-ENG-88-2257, ACT-103, University of Illinois at Urbana-Champaign, 1988.

[Rei78] S. Reiss. Rational search. *Information Processing Letters*, 8(2):87–90, 1978.

[Rei84] J. H. Reif. On synchronous parallel computations with independent probabilistic choice. *SIAM Journal on Computing*, 13(1):46–56, February 1984.

[Rei85] J. Reif. Depth-first search is inherently sequential. *Information Processing Letters*, 20(5):229–234, 1985.

[RTY90] John H. Reif, J. D. Tygar, and Akitoshi Yoshida. The computability and complexity of optical beame tracing. In *31st Annual Symposium on Foundations of Computer Science*, pages 106–114, St. Louis, MO, October 1990. IEEE.

[Ruz79] W. L. Ruzzo. Complete pushdown languages. Unpublished manuscript, 1979.

[Ruz80] W. L. Ruzzo. Tree-size bounded alternation. *Journal of Computer and System Sciences*, 21(2):218–235, October 1980.

[Ruz81] W. L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383, June 1981.

[Sav70] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[Sch78] T. J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.

[Ser90] M. J. Serna. *The Parallel Approximability of P-complete Problems.* PhD thesis, Universitat Politecnica de Catalunya, Barcelona, 1990.

[Sho89] T. Shoudai. The lexicographically first topological order problem is NLOG-complete. *Information Processing Letters*, 33(3):121–124, 1989.

[Smi86] J. R. Smith. Parallel algorithms for depth first searches I: Planar graphs. *SIAM Journal on Computing*, 15(3):814–830, 1986.

[Spi87] P. Spirakis. The parallel complexity of deadlock detection. *Theoretical Computer Science*, 52(1,2):155–163, 1987.

[SS79] W. J. Savitch and M. J. Stimson. Time bounded random access machines with parallel processing. *Journal of the ACM*, 26(1):103–118, 1979.

[SS89] M. Serna and P. Spirakis. The approximability of problems complete for P. In G. Goos and J. Hartmanis, editors, *International Symposium on Optimal Algorithms*, volume 401 of *Lecture Notes in Computer Science*, pages 193–204. Springer-Verlag, Berlin York, 1989.

[Ste89] I. A. Stewart. Complete problems for symmetric logspace involving free groups. Technical Report 300, University of Newcastle upon Tyne, 1989. To appear, *Information Processing Letters.*

[Ste90] I. A. Stewart. The complexity of restricted versions of the generalized word problem for free groups. Manuscript, University of Newcastle upon Tyne, submitted for publication, 1990.

[Sto84] L. J. Stockmeyer. Unpublished notes on the Cat and Mouse Game, 1984.

[Sub89] Ashok Subramanian. A new approach to stable matching problems. Technical Report STAN-CS-89-1275, Stanford, 1989.

[Sub90] A. Subramanian. *The Computational Complexity of the Circuit Value and Network Stability Problems.* PhD thesis, Department of Computer Science, Stanford University, 1990.

[Sud78] I. H. Sudborough. On the tape complexity of deterministic context-free languages. *Journal of the ACM*, 25(3):405–414, 1978.

[SV84] L. Stockmeyer and U. Vishkin. Simulation of parallel random access machines by circuits. *SIAM Journal on Computing*, 13(2):409–422, May 1984.

[SV85] C. A. Schevon and J. S. Vitter. A parallel algorithm for recognizing unordered depth-first search. Technical Report CS-85-21, Brown University, Department of Computer Science, 1985.

[SW91] Clifford Stein and Joel Wein. Approximating the minimum-cost maximum flow is $\mathcal{P}$-complete. Manuscript, September 1991.

[SY] A. A. Schäffer and M. Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing.* To appear.

[Tom82] M. Tompa. A pebble game that models alternation. Unpublished manuscript, 1982.

[Tom83] M. P. Tompa. Unpublished notes on *NC* reductions among problems in *RP*, 1983.

[UVG88] J. D. Ullman and A. Van Gelder. Parallel complexity of logical query programs. *Algorithmica*, 3:5–52, 1988.

[Val82a] L. G. Valiant. Reducibility by algebraic projections. *L'Enseignement Mathématique*, XXVIII:253–268, 1982. Also in [L'E82, pages 365–380].

[Val82b] Leslie G. Valiant. Parallel computation. Technical Report TR-16-82, Harvard, Center of Research in Computing Technology, April 1982. To be presented at the 7th IBM Symposium on Mathematical Foundations of Computer Science, Hakone, Kanagawa, Japan, May 24–26, 1982.

[Vav89] S. Vavasis. Gaussian elimination with pivoting is *P*-complete. *SIAM Journal on Discrete Mathematics*, 2(3):413–423, 1989.

[Ven83] H. Venkateswaran. Private communication, 1983.

[Ven89] H. Venkateswaran. Two natural circuits for which interpreted pebbling helps. Technical Report GIT-ICS-89-23, Georgia Institute of Technology, June 1989. Submitted for publication.

[Ven90] H. Venkateswaran. The unary maximum flow problem is hard for ac^1. Technical Report GIT-ICS-90-xx, Georgia Institute of Technology, School of Information and Computer Science, January 1990.

[VS86] J. S. Vitter and R. A. Simons. New classes for parallel complexity: A study of unification and other complete problems for $\mathcal{P}$. *IEEE Transactions on Computers*, TC-35:403–418, May 1986.

[VS88] S. Vishwanathan and M. A. Sridhar. Some results on graph coloring in parallel. In *Proceedings International Conference on Parallel Processing III*, pages 299–303, 1988.

[VSBR83] Leslie G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing*, 12(4):641–644, November 1983.

[VT89] H. Venkateswaran and M. Tompa. A new pebble game that characterizes parallel complexity classes. *SIAM Journal on Computing*, 18(3):533–549, June 1989.

[Wil88] Robert Wilber. White pebbles help. *Journal of Computer and System Sciences*, 36(2):108–124, April 1988.

[WLS85] C. C. Wang, E. L. Lloyd, and M. L. Soffa. Feedback vertex sets and cyclically reducible graphs. *Journal of the ACM*, 32(2):296–313, 1985.

[Yas84] H. Yasuura. On parallel computational complexity of unification. In *Proceedings International Conference on Fifth Generation Computer Systems*, pages 235–243, 1984.

Problem Index

0-1 MaxFlow .... (B.5.2) 95
0-1 Maximum Flow .... (B.5.2) 95
2-Oriented Weighted Planar Partitioning .... (B.3.1) 92
2DPDA .... (A.7.8) 73
2OWPP .... (B.3.1) 92
3-Oriented Weighted Planar Partitioning .... (A.9.2) 83
3OWPP .... (A.9.2) 83
ABFS .... (A.3.5) 61
ACMDP .... (A.8.3) 75
AGAP .... (A.2.3) 51
Alternating Breadth First Search .... (A.3.5) 61
Alternating Graph Accessibility Problem .... (A.2.3) 51
Alternating Monotone Fanout 2 CVP .... (A.1.4) 48
AM2CVP .... (A.1.4) 48
ArithCVP .... (A.1.8) 49
Arithmetic Circuit Value Problem (*) .... (A.1.8) 49
Average Cost Markov Decision Process .... (A.8.3) 75
BDGI .... (B.2.2) 91
BDS .... (A.3.3) 60
Boolean Recurrence Equation . (A.8.24) 82
Bounded Degree Graph Isomorphism .... (B.2.2) 91
Breadth-Depth Search .... (A.3.3) 60
BRE .... (A.8.24) 82
Cat & Mouse .... (A.11.3) 87
CCVP .... (B.4.1) 94
CFGϵmem .... (A.7.4) 72
CFGempty .... (A.7.2) 71
CFGinf .... (A.7.3) 72
CFGmem .... (A.7.1) 71
Circuit Value Problem .... (A.1.1) 47
CM .... (A.11.3) 87
Comparator Circuit Value Problem .... (B.4.1) 94
Conjugate Subgroups .... (A.8.17) 79
Context-Free Grammar ϵ-Membership .... (A.7.4) 72
Context-Free Grammar Empty (A.7.2) 71
Context-Free Grammar Infinite (A.7.3) 72
Context-Free Grammar Membership .... (A.7.1) 71
CS .... (A.8.17) 79
CVP .... (A.1.1) 47
DFS .... (B.5.1) 95
Directed or Undirected Depth First Search .... (B.5.1) 95
Discounted Markov Decision Process .... (A.8.2) 74
DMDP .... (A.8.2) 74
Edge Maximal Acyclic Subgraph .... (A.2.15) 57
Edge-Weighted Matching ...... (B.2.1) 91
EMAS .... (A.2.15) 57
EWM .... (B.2.1) 91
Extended Euclidean Algorithm (B.1.2) 89
ExtendedGCD .... (B.1.2) 89
FFDBP .... (A.4.7) 64
FGS .... (A.8.20) 80
FHMDP .... (A.8.1) 74
Finite Horizon Markov Decision Process .... (A.8.1) 74
Finitely Generated Subalgebra (A.8.20) 80
Finiteness Problem for Finitely Presented Algebras .... (A.8.21) 81
First Fit Decreasing Bin Packing .... (A.4.7) 64
Fixed Points of Contraction Mappings .... (A.10.2) 85
FPCM .... (A.10.2) 85
FPFPA .... (A.8.21) 81
GAME .... (A.11.2) 87
Gaussian Elimination with Partial Pivoting .... (A.8.4) 75
GCE .... (A.8.16) 79
GC .... (B.2.3) 91
GDD .... (A.11.1) 86
Generability .... (A.6.5) 69
General Deadlock Detection ... (A.11.1) 86

General Graph Closure .... (A.2.16) 58
General List Scheduling .... (A.4.8) 65
Generalized Word Problem .... (A.8.6) 76
Generator Problem for Lattices (A.8.23) 82
GEN .... (A.6.5) 69
GEPP .... (A.8.4) 75
GGC .... (A.2.16) 58
GII .... (A.8.13) 78
GI .... (A.8.10) 78
GLS .... (A.4.8) 65
GPL .... (A.8.23) 82
Graph Closure .... (B.2.3) 91
Group Coset Equality .... (A.8.16) 79
Group Independence .... (A.8.10) 78
Group Induced Isomorphism .. (A.8.13) 78
Group Isomorphism .... (A.8.12) 78
Group Rank .... (A.8.11) 78
GR .... (A.8.11) 78
GWP .... (A.8.6) 76
HCS .... (A.2.8) 54
HDS .... (A.2.7) 53
HF .... (A.4.5) 64
HGAP .... (A.2.4) 51
Hierarchical Graph
  Accessibility Problem .... (A.2.4) 51
High Connectivity Subgraph .. (A.2.8) 54
High Degree Subgraph .... (A.2.7) 53
Homologous Flow .... (A.4.5) 64
Horn Unit Resolution .... (A.6.2) 68
HORN .... (A.6.2) 68
IC .... (A.8.14) 79
IM .... (A.8.5) 76
Integer Greatest Common
  Divisor .... (B.1.1) 89
IntegerGCD .... (B.1.1) 89
Intersection of Cosets .... (A.8.14) 79
Intersection of Subgroups ..... (A.8.15) 79
Inverting a One-to-One Real
  Functions .... (A.10.3) 86
Inverting an NC^0 Permutation
  (*) .... (A.1.10) 50
InvNC0Perm .... (A.1.10) 50
IOORF .... (A.10.3) 86
IS .... (A.8.15) 79
Iterated Mod .... (A.8.5) 76
Labeled GAP .... (A.7.7) 73
Lexicographically First $\Delta + 1$
  Vertex Coloring .... (A.2.6) 52
Lexicographically First
  Blocking Flow .... (A.4.6) 64
Lexicographically First Depth
  First Search Ordering .... (A.3.2) 59
Lexicographically First
  Maximal Clique .... (A.2.2) 51
Lexicographically First
  Maximal Independent Set ... (A.2.1) 50
Lexicographically First
  Maximal Matching .... (B.4.2) 94
Lexicographically First
  Maximal Path .... (A.3.1) 59
Lexicographically First
  Maximal Subgraph for π .... (A.2.13) 56
LE .... (A.4.2) 62
LFBF .... (A.4.6) 64
LFDFS .... (A.3.2) 59
LFDVC .... (A.2.6) 52
LFMC .... (A.2.2) 51
LFMIS .... (A.2.1) 50
LFMM .... (B.4.2) 94
LFMP .... (A.3.1) 59
LFMS(π) .... (A.2.13) 56
LGAP .... (A.7.7) 73
Limited Reflection Ray Tracing (B.3.2) 92
Linear Equalities .... (A.4.2) 62
Linear Inequalities .... (A.4.1) 62
Linear Programming (*) ...... (A.4.3) 62
LI .... (A.4.1) 62
Local Optimality
  Kernighan-Lin Verification .. (A.5.2) 66
Logical Query Program .... (A.6.8) 70
LOKLV .... (A.5.2) 66
LONGCAKE .... (A.11.4) 88
Longcake .... (A.11.4) 88
LP .... (A.4.3) 62
LQP .... (A.6.8) 70
LRRT .... (B.3.2) 92
MAXFLIP Verification .... (A.5.1) 66
MAXFLIPV .... (A.5.1) 66
MaxFlow .... (A.4.4) 63
Maximal Independent Set
  Hypergraph .... (B.2.6) 92
Maximal Path .... (B.2.5) 92

Maximum Flow .... (A.4.4) 63
Maximum Matching .... (B.5.3) 95
MCVP .... (A.1.3) 48
MFVS .... (A.2.14) 56
Min-Plus Circuit Value
  Problem .... (A.1.9) 50
Minimum Feedback Vertex Set (A.2.14) 56
MinPlusCVP .... (A.1.9) 50
MISH .... (B.2.6) 92
MM .... (B.5.3) 95
ModInverse .... (B.1.4) 89
ModPower .... (B.1.5) 89
Modular Inversion .... (B.1.4) 89
Modular Powering .... (B.1.5) 89
Monotone Circuit Value
  Problem .... (A.1.3) 48
MP .... (B.2.5) 92
NAND Circuit Value Problem .(A.1.5) 48
NANDCVP .... (A.1.5) 48
Nearest Neighbor Traveling
  Salesman Heuristic .... (A.2.12) 55
NNTSH .... (A.2.12) 55
OHDVR .... (A.2.9) 54
OLDVR .... (A.2.10) 55
Ordered High Degree Vertex
  Removal .... (A.2.9) 54
Ordered Low Degree Vertex
  Removal .... (A.2.10) 55
Ordered Vertices Remaining ...(A.2.11) 55
OVR .... (A.2.11) 55
Path Systems .... (A.6.6) 70
PATH .... (A.6.6) 70
PCVP .... (A.1.7) 49
Perfect Matching Existence ... (B.5.4) 95
PHCS .... (A.6.3) 68
Planar Circuit Value Problem .(A.1.7) 49
Plane Sweep Triangulation .... (A.9.1) 82
PME .... (B.5.4) 95
Propositional Horn Clause
  Satisfiability .... (A.6.3) 68
PST .... (A.9.1) 82
RCAGAP .... (A.2.5) 52
RCL .... (A.6.4) 69
Real Analogue to CVP .... (A.10.1) 85
RealCVP .... (A.10.1) 85
Relative Primeness .... (B.1.3) 89
Relaxed Consistent Labeling .. (A.6.4) 69
RelPrime .... (B.1.3) 89
Restricted Chromatic
  Alternating Graph
  Accessibility Problem .... (A.2.5) 52
Restricted Lexicographically
  First Independent Set .... (B.2.4) 91
Restricted Plane Sweep
  Triangulation .... (B.3.3) 93
RLFMIS .... (B.2.4) 91
SAM2CVP .... (A.1.6) 48
SBFS .... (A.3.4) 60
SCH .... (B.3.4) 93
SC .... (A.8.7) 77
SE .... (A.8.8) 77
SFI .... (A.8.9) 77
Short Vectors Dimension 2 .... (B.1.7) 90
Short Vectors .... (B.1.6) 90
SI .... (A.8.12) 78
SLPmem .... (A.7.5) 72
SLPnonempty .... (A.7.6) 72
SM .... (B.4.3) 94
Stable Marriage .... (B.4.3) 94
Stack Breadth First Search .... (A.3.4) 60
Straight-Line Program
  Membership .... (A.7.5) 72
Straight-Line Program
  Nonempty .... (A.7.6) 72
Subgroup Containment .... (A.8.7) 77
Subgroup Equality .... (A.8.8) 77
Subgroup Finite Index .... (A.8.9) 77
Successive Convex Hulls .... (B.3.4) 93
SV2 .... (B.1.7) 90
SV .... (B.1.6) 90
SWEEP .... (B.3.3) 93
Sylow Subgroups .... (B.1.8) 90
SylowSub .... (B.1.8) 90
Synchronous Alternating
  Monotone Fanout 2 CVP .... (A.1.6) 48
TopCVP .... (A.1.2) 47
Topologically Ordered Circuit
  Value Problem .... (A.1.2) 47
TPFPA .... (A.8.19) 80
Triviality Problem for Finitely
  Presented Algebras .... (A.8.19) 80
TVLP .... (B.1.9) 90

Two Player Game ....(A.11.2) 87
Two Variable Linear
  Programming ....(B.1.9) 90
Two-Way DPDA Acceptance ..(A.7.8) 73
U2SATFLIP ....(A.5.5) 67
U3NSATFLIP ....(A.5.3) 66
ULVL ....(B.3.5) 93
UMS ....(A.5.4) 67
Unification ....(A.6.7) 70
Uniform Word Problem for
  Finitely Presented Algebras .(A.8.18) 80
Uniform Word Problem for
  Lattices ....(A.8.22) 81
UNIF ....(A.6.7) 70
Unit Length Visibility Layers .(B.3.5) 93
Unit Resolution ....(A.6.1) 68
UNIT ....(A.6.1) 68
Univariate Polynomial
  Factorization over Q ....(B.1.10) 91
Unweighted 2SAT/FLIP ......(A.5.5) 67
Unweighted MAXCUT/SWAP (A.5.4) 67
Unweighted STABLE NET ....(A.5.7) 67
Unweighted SWAP ....(A.5.6) 67
Unweighted, Not-All-Equal
  Clauses, 3SAT/FLIP ....(A.5.3) 66
UPFQ ....(B.1.10) 91
USN ....(A.5.7) 67
USWAP ....(A.5.6) 67
UWPFPA ....(A.8.18) 80
UWPL ....(A.8.22) 81
Visibility Layers ....(A.9.3) 84
VL ....(A.9.3) 84