produces a solution with cost at most c times the optimum, it produces one of cost n if and only if G has a spanning cycle.

Thus our polynomial algorithm for HAMILTONIAN CYCLE generates an instance of TSP as stated and runs A on this instance. The graph G is Hamiltonian if and only if A produces a cycle of cost n. ∎

Approximation algorithms do exist for some special classes of TSP problems. To prove that an algorithm is an approximation algorithm, we need a lower bound on the optimal solution.

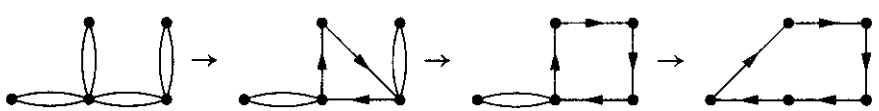
Let M be the cost of a minimum spanning tree in a weighted graph G. If we delete an edge from the optimal cycle in G, we obtain a spanning path. Since this is a spanning tree, its cost is at least M. The cost of the optimal cycle is at least M plus the minimum cost of an edge not in some tree with cost M. We can run Kruskal's Algorithm for minimum spanning trees to compute this bound.

B.4. Theorem. On the class of Traveling Salesman Problems where the input satisfies the triangle inequality, there is an approximation algorithm that finds a spanning cycle with cost at most twice the optimum.

Proof: Satisfying the triangle inequality means that the matrix of costs satisfies $w_{i,j} + w_{j,k} \geq w_{i,k}$ for all i, j, k. We know that the cost of the optimal cycle is at least M, where M is the cost of the minimum spanning tree. We use the triangle inequality and the minimum spanning tree to obtain a spanning cycle with cost at most $2M$.

Begin by duplicating each edge in a minimum spanning tree. As on the left below. This makes all degrees even, so there is an Eulerian circuit; the circuit has $2n$ edges and total cost $2M$. We successively reduce the number of edges without increasing the cost until only n edges remain. By maintaining the property that the circuit visits all vertices, we ensure that the circuit at the end is a spanning cycle with cost at most $2M$.

If a circuit has more than n edges, then it visits some vertex more than once, say via edges $v_i \to v_j \to v_k$ and $v_r \to v_j \to v_s$. Replace the edges v_iv_j and v_jv_k with v_iv_k. The result is still a circuit visiting all the original vertices. Furthermore, the triangle inequality guarantees that the total cost of the edges is no larger than before. In the figure below, we orient edges to suggest a particular succession of circuits. ∎



The algorithm of Theorem B.4 was rediscovered many times. Christofides [1976] improved the performance ratio to 3/2. After finding a minimum spanning tree, we needn't double all the edges to obtain an Eulerian circuit. It suffices to add edges to pair vertices that have odd degree in the tree. The resulting graph has an Eulerian circuit, and the subsequent part of the algorithm

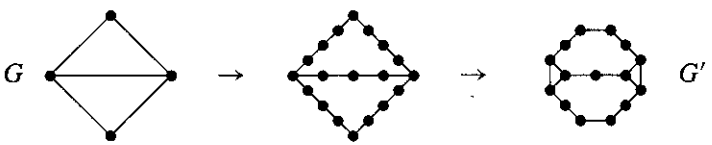
produces a spanning cycle with cost at most M plus the cost of the matching. To obtain the performance ratio 3/2, it then suffices to show that there is such a matching with cost at most half the cost of the optimal cycle (Exercise 8). The improvement has value because a minimum weight matching saturating the vertices of odd degree in the tree can be found in polynomial time.

NP-COMPLETENESS PROOFS

A **transformation** of problem A to problem B is a procedure that converts instances of A into instances of B so that the answer to A on the original instance is determined by the answer to B on the transformed instance. If we have an efficient (polynomial-time) transformation of A to B and an efficient algorithm for B, then we have an efficient algorithm for A. We say that A **reduces to** or **transforms to** B.

If A is NP-hard and reduces to B by a polynomial-time transformation, then B is also NP-hard (a polynomial algorithm for B yields a polynomial algorithm for A and hence for all of NP). If also B is in NP, then we say that B is NP-complete by **reduction from** or **transformation from** A.

The direction of the reduction is crucial. For example, EULERIAN CIRCUIT reduces easily to HAMILTONIAN CYCLE. Given an input graph G, replace each edge with a path of four edges through three new vertices, add edges to make the neighbors of each original vertex pairwise adjacent, and delete $V(G)$. The graph G is Eulerian if and only if the resulting graph G' is Hamiltonian. Applying an algorithm for HAMILTONIAN CYCLE to G' would determine whether G is Eulerian. This tells use that HAMILTONIAN CYCLE is at least as hard as EULERIAN CIRCUIT, up to a polynomial factor. Since EULERIAN CIRCUIT is easy (in P), we learn nothing of use about the complexity of HAMILTONIAN CYCLE.



The reduction technique requires an initial NP-complete problem. Cook [1971] provided SATISFIABILITY as such a problem. SATISFIABILITY takes as input a logical formula expressed as a list of clauses; each clause is a collection of literals (variables or their negations). A clause is considered true when at least one of its literals is true. A formula is **satisfiable** if there is an assignment of truth values to the variables that makes every clause true. The question is whether such an assignment exists.

Cook proved that for every problem A in NP, an instance of SATISFIABILITY can be produced in polynomial time from an instance of A such that the answer to the SATISFIABILITY instance is the same as the answer to the instance of A. Thus every problem in NP reduces to SATISFIABILITY.

This effort need not be repeated for each NP-complete problem. To prove that B is NP-hard, we can reduce SATISFIABILITY to B. Any NP-complete problem may be reduced to show that B is NP-hard. With more options, in principle the proofs become easier to find, but in practice a few fundamental NP-complete problems serve as the known NP-complete problem in most NP-completeness proofs.

Starting from SATISFIABILITY, Karp [1972] provided 21 such problems. These include many fundamental problems of graph theory, including HAMILTONIAN CYCLE and MAXIMUM INDEPENDENT SET. It helps to have as restrictive a version of an NP-complete problem as possible while still remaining NP-complete. Since a restricted version has less flexibility, it is easier to reduce it to the problem we are trying to prove NP-complete.

For example, SATISFIABILITY remains NP-complete when restricted by requiring that every clause have three literals. The restricted problem is called 3-SATISFIABILITY or 3-SAT. This is proved NP-complete by considering an arbitrary instance of SATISFIABILITY and replacing each clause by an equivalent collection of clauses with three literals (at the cost of introducing some additional variables). 3-SAT is sufficiently restrictive that many, NP-completeness proofs use reduction from 3-SAT. It is easier yet to reduce the more restrictive 2-SAT, but this is so restrictive that 2-SAT is solvable in polynomial time.

We start from 3-SAT because the NP-completeness of SATISFIABILITY and the reduction of it to 3-SAT do not involve graph theory. Our reductions to 3-COLORABILITY and DIRECTED HAMILTONIAN PATH follow the presentation of Gibbons [1985].

B.5. Definition. 3-SATISFIABILITY (3-SAT)

Instance: A set of logical **variables** $U = \{u_j\}$ and a set $\mathbf{C} = \{C_i\}$ of **clauses**, where each clause consists of three literals, a **literal** being a variable u_i or its negation $\overline{u}_i$.

Question: Can the value of each variable be set to True or False so that each clause is "satisfied" (contains at least one true literal)?

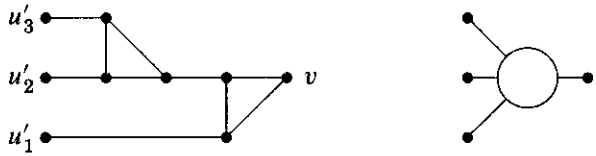
B.6. Theorem. (Karp [1972]) 3-SAT is NP-complete.

Proof: (Exercise 14). ∎

B.7. Theorem. (Stockmeyer [1973]) 3-COLORABILITY is NP-complete.

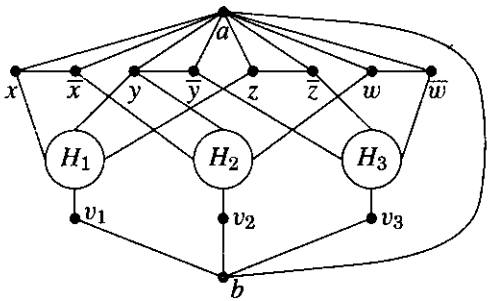
Proof: In this problem, we are given a graph and asked whether is it 3-colorable. If the answer is YES, then there exists a proper 3-coloring, and we can verify in quadratic time that the coloring is proper. Hence 3-COLORABILITY is in NP. To prove that it is NP-hard, we reduce 3-SAT to 3-COLORABILITY.

Consider an instance of 3-SAT with variables $U = \{u_j\}$ and clauses $\mathbf{C} = \{C_i\}$. We transform this into a graph G that is 3-colorable if and only if the instance of 3-SAT is satisfiable. We use the auxiliary graph H below, calling $\{u'_1, u'_2, u'_3\}$ the **inputs** and v the **output**. When we use H in the transformation, we will attach it to a larger graph at the inputs, as suggested on the right.



We consider 3-colorings using the color set $\{0, 1, 2\}$. Every proper 3-coloring of H in which the inputs all have color 0 also assigns color 0 to v. On the other hand, if the inputs receive colors that are not all 0, then this coloring extends to a proper 3-coloring of H in which v does not have color 0.

From our instance of 3-SAT, we construct a graph G having vertices u_j and $\overline{u}_j$ for each variable in U, a copy H_i of H for each clause C_i, and two special vertices a, b. For each j, the vertices $a, u_j, \overline{u}_j$ form a triangle. For each clause C_i, the subgraph H_i attaches to the graph formed thus far at the vertices for the literals in C_i. The vertex for the jth literal in C_i plays the role of u'_j in H_i. Except for these attachment vertices, the subgraphs H_i are pairwise disjoint. Finally, the vertex b is adjacent to a and to the output vertex v_i for each H_i. Below we draw the graph G resulting from an instance of 3-SAT having four variables and three clauses.



A satisfying truth assignment for $\{c_i\}$ yields a proper 3-coloring f of G. When u_i is true in the assignment, set $f(u_i) = 1$ and $f(\overline{u}_i) = 0$; otherwise, set $f(u_i) = 0$ and $f(\overline{u}_i) = 1$. For each clause, some literal is true; hence for each H_i at least one of $\{u'_1, u'_2, u'_3\}$ has color 1. By our observation about H, we can extend f so that each v_i has a color other than 0. We complete the proper 3-coloring by setting $f(b) = 0$ and $f(a) = 2$.

Conversely, suppose that G has a proper 3-coloring f. By renaming colors if necessary, we may assume that $f(a) = 2$ and $f(b) = 0$. Since $f(a) = 2$, for each variable we have one literal colored 0 and one colored 1. Consider the truth assignment in which variable u_j is true or false when $f(u_j)$ is 1 or 0, respectively. We claim that this is a satisfying truth assignment. Since $f(b) = 0$, every output vertex v_i has nonzero color. By our observation about H, the vertices corresponding to the inputs in H_i cannot all have color 0. Therefore, each clause contains at least one true literal. ∎

Exercise 2 extends this to k-COLORABILITY for fixed $k \geq 3$. For each k it is a special case of CHROMATIC NUMBER, which thus is also NP-complete.

B.8. Theorem. (Karp [1972]) INDEPENDENT SET, CLIQUE, and VERTEX COVER are NP-complete.

Proof: A YES answer to the question of whether an input graph G has an independent set as large as an input integer k can be verified by exhibiting the set and checking (in quadratic time) that its vertices are independent. Hence INDEPENDENT SET is in NP.

Exercise 5.1.31 states that G is m-colorable if and only if $G \square K_m$ has an independent set of size $n(G)$. This transformation reduces CHROMATIC NUMBER to INDEPENDENT SET. The construction of $G \square K_m$ is quadratic in $n(G)$, since CHROMATIC NUMBER is trivial if $m > n$. We conclude that INDEPENDENT SET is NP-hard.

Since cliques in G are independent in $\overline{G}$, CLIQUE and INDEPENDENT SET are polynomially equivalent. Since $\alpha(G) + \beta(G) = n(G)$ (Lemma 3.1.21), INDEPENDENT SET and VERTEX COVER are polynomially equivalent. ■

We next consider problems of traversing graphs via spanning paths and cycles. Digraph problems are more general than graph problems, since we can model graphs by using symmetric digraphs. Thus, it may be easiest to prove a digraph version of a problem NP-hard and then obtain the graph version by a simple restriction.

B.9. Definition. Given vertices x, y in a digraph D, the DIRECTED HAMILTONIAN PATH problem asks whether D has a spanning x, y-path.

B.10. Theorem. (Karp [1972]) DIRECTED HAMILTONIAN PATH is NP-complete.

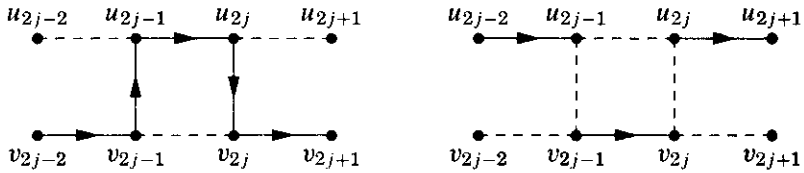
Proof: A spanning x, y-path in a digraph D can be verified in linear time. Thus DIRECTED HAMILTONIAN PATH is in NP. To show that it is NP-hard, we reduce VERTEX COVER to DIRECTED HAMILTONIAN PATH.

Consider an instance of VERTEX COVER, consisting of a graph G and an integer k; we want to know whether G has a vertex cover of size (at most) k. We construct a digraph D such that D has a Hamiltonian path if and only if G has a vertex cover of size at most k. We index the edges incident to each vertex arbitrarily. When the edge $e = uv$ is the ith edge incident to u and the jth edge incident to v, we write $e_i(u) = e = e_j(v)$.

To build D, we start with $k+1$ special vertices $z_0, \ldots, z_k$. For each $v \in V(G)$, we add a path $P(v) = v_1, \ldots, v_{2r}$ to D, where $r = d_G(v)$. We add edges from each of $z_0, \ldots, z_{k-1}$ to the start of each $P(v)$ and from the end of each $P(v)$ to each of $z_1, \ldots, z_k$. Also, for each edge $e = e_i(u) = e_j(v)$ in $E(G)$, we create the edges $u_{2i-1}v_{2j-1}$, $v_{2j-1}u_{2i-1}$, $u_{2i}v_{2j}$, and $v_{2j}u_{2j}$.

Suppose that G has a vertex cover of size k, consisting of vertices $v^1, \ldots, v^k$. We form a z_0, z_k-path in D by following $z_0, P(v^1), z_1, \ldots, P(v^k), z_k$. This path omits all of $P(u)$ for each uncovered vertex u. We absorb these vertices in pairs, absorbing each pair $u_{2i-1}u_{2i}$ by making a detour from the path $P(v)$ for the vertex that covers the edge $uv = e_i(u) = e_j(v)$. The detour is $v_{2j-1}u_{2j-1}u_{2j}v_{2j}$, as

shown below. Because the vertices v_{2j-1}, v_{2j} are associated with only one edge, each such detour is requested at most once. After implementing all the detours, we have a Hamiltonian z_0, z_k-path in D.



Conversely, suppose that there is such a path Q. Note that every vertex of $P(u)$ except the first and last has indegree and outdegree 2, for each $u \in V(G)$. We show first for $i \geq 1$ and $u \in V(G)$ that $u_{2i}u_{2i+1} \in E(Q)$ if and only if $u_{2i-2}u_{2i-1} \in E(Q)$, where for $i = 1$ we take u_0 to mean some element of $\{z_r\}$. Similarly, when $i = d(u)$ we take u_{2i+1} to mean some element of $\{z_r\}$ Since the ith edge incident to u is well-defined, we can let the vertex v and index j be such that $e_i(u) = e_j(v)$.

If $u_{2i-2}u_{2i-1} \notin E(Q)$, then Q must enter u_{2i-1} from v_{2j-1}. This implies that Q can only leave u_{2i-1} on $u_{2i-1}u_{2i}$ and can only enter v_{2j} on $u_{2i}v_{2j}$. This in turn implies that $u_{2i}u_{2i+1} \notin E(Q)$.

If $u_{2i-2}u_{2i-1} \in E(P)$, then P cannot leave v_{2j-1} on $v_{2j-1}u_{2i-1}$ and must leave v_{2j-1} on $v_{2j-1}v_{2j}$. This implies that Q enters v_{2j} on $v_{2j-1}v_{2j}$ and not on $u_{2i}v_{2j}$. Hence Q does not leave u_{2i} on $u_{2i}v_{2j}$ and must leave Q on $u_{2i}u_{2i+1}$. (In this case, Q may include $\{u_{2i-1}u_{2i}, v_{2i-2}v_{2i-1}, v_{2i}v_{2i+1}\}$ or $\{u_{2i-1}v_{2i-1}, v_{2i}u_{2i}\}$.)

Now let $S = \{v \in V(G)\colon z_iv_1 \in Q\}$; these are the k vertices in G whose initial copies are entered from $z_0, \ldots, z_{k-1}$ by Q. Our argument shows for each edge uv that $u \notin S$ implies $v \in S$. Hence S is a vertex cover, and we have the desired reduction of VERTEX COVER to DIRECTED HAMILTONIAN PATH. ∎

B.11. Corollary. (Karp [1972]) DIRECTED HAMILTONIAN CYCLE, HAMILTONIAN PATH, HAMILTONIAN CYCLE are NP-complete.

Proof: All these problems are in NP. To reduce DIRECTED HAMILTONIAN PATH to DIRECTED HAMILTONIAN CYCLE, add one vertex z and edges vz and zu to an instance requesting a spanning u, v-path in D.

The reduction of HAMILTONIAN PATH (with specified endpoints) to HAMILTONIAN CYCLE is the same.

To reduced DIRECTED HAMILTONIAN PATH to HAMILTONIAN PATH, consider an instance that requests a u, v-path in D. To form an instance G of HAMILTONIAN PATH, first split each vertex x into a path x^-, x^0, x^+. Let x^- inherit all edges with heads at x, and let x^+ inherit all edges with tails at x. A spanning u, v-path in D becomes a spanning u^-, v^+-path in G by replacing each vertex x with the sequence x^-, x^0, x^+.

Conversely, since a spanning u^-, v^+-path in G must visit each x^0, it must visit traverse all sequences of the form x^-, x^0, x^+, forwards or backwards. Since no vertices of the same sign are adjacent, these traversals must all be in the same direction, and then they collapse to the desired u, v-path in D. Thus G has a spanning u^-, v^+-path if and only if D has a spanning u, v-path. ∎

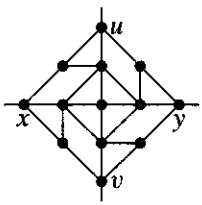
Exploration of the boundary between P and NP-complete seeks more restricted problems that remain NP-complete and larger classes of inputs on which there are polynomial-time solution algorithms. The former provides easier NP-completeness proofs and places limits on the successes of the latter type. The latter aim is that of extending the applicability of good algorithms.

We illustrate this process by proving that 3-COLORABILITY remains NP-complete for planar graphs.

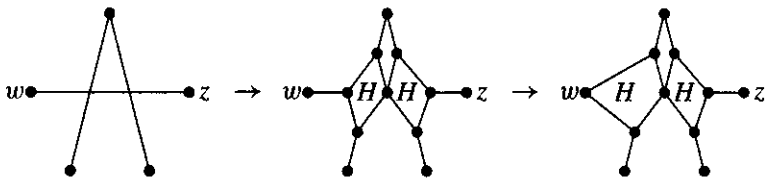
B.12. Theorem. (Stockmeyer [1973], see also Garey–Johnson–Stockmeyer [1976]) PLANAR 3-COLORABILITY is NP-complete.

Proof: As usual, it is easy to verify that a 3-coloring is proper. We reduce 3-COLORABILITY to PLANAR 3-COLORABILITY. Given an arbitrary graph G, we construct a planar graph G' such that G' is 3-colorable if and only if G is 3-colorable.

Consider a drawing of G in the plane. We replace each crossing by a planar "gadget" that has the effect of propagating color across the crossing in a 3-coloring. In every proper 3-coloring of the graph H below, u and v have the same color, as do x and y (Exercise 17). The partial edges indicate where copies of H will be linked to the rest of the graph.



In the drawing of G, each edge involved in k crossings is cut into $k + 1$ segments by the crossings. On each segment, add a new vertex. Replace each crossing by a copy of H attached by its terminals to the new vertices on the four segments incident to the crossing. Finally, for each original edge wz, choose one endpoint and contract the edge between it and the vertex on the segment of wz incident to it. An edge involved in no crossings returns to its original state.



In a proper 3-coloring of the new graph G', the propagation of color across the gadgets requires the endpoints of each original edge to have different colors. Hence restricting this coloring to the original vertices yields a 3-coloring of G.

Conversely, given a proper 3-coloring of G, we can start along each edge from the endpoint used in a copy of H and propagate the colors to properly 3-color G'. We can do this because H has both a proper 3-coloring in which u, v, x, y all get the same color and a proper 3-coloring in which x and y have a different color from u and v (Exercise 17). ∎

HAMILTONIAN CYCLE also remains NP-complete for planar graphs. Indeed, it remains NP-complete for graphs that are planar, 3-regular, 3-connected, and have no face of length less than 5 (Garey–Johnson–Tarjan [1976]). Also it is NP-complete for bipartite graphs (Krishnamoorthy [1975]) and for line graphs (Bertossi [1981]; see also Exercise 14). The restriction of 3-COLORABILITY to line graphs is the same as 3-EDGE-COLORABILITY for graphs; this is NP-complete even for 3-regular graphs, by reduction from 3-SAT (Holyer [1981]).

EXERCISES

B.1. (–) Using algorithms developed earlier in this text, describe a good algorithm to compute $\alpha(G)$ when G is bipartite.

B.2. (!) Use Theorem B.7 to prove that k-COLORABILITY is NP-complete for each fixed value of k that is at least 3.

B.3. Give a polynomial-time algorithm for 2-COLORABILITY.

B.4. Prove that HAMILTONIAN CYCLE and HAMILTONIAN PATH are polynomially equivalent. That is, show that a polynomial-time algorithm for either one can be used to obtain a polynomial-time algorithm for the other.

B.5. Testing for a cycle of fixed length k in an input graph with n vertices can be done in time bounded by a multiple of $k!n^k$: look at each of the $\binom{n}{k}$ vertex subsets of size k in turn and test all possible orderings. Since k is a constant, this is polynomial time. For a 4-cycle, this runs in time $O(n^4)$. Devise an algorithm that will test for the presence of a 4-cycle in time $O(n^2)$. (Richards–Liestman [1985])

B.6. Given a graph G and an integer k, the MINIMUM DEGREE SPANNING TREE problem asks whether G has a spanning tree T such that $\Delta(T) \leq k$, and the LONGEST PATH problem asks whether G has a path of length at least k. In the k-PATH problem, k is not part of the input, and the question is whether G has a path of length at least k.

a) Prove that MINIMUM DEGREE SPANNING TREE and LONGEST PATH are NP-complete.

b) Prove the k-PATH is in P for each fixed k.

B.7. Construct a family of examples to prove that the performance ratio of the nearest-insertion heuristic for the TSP is not bounded by any constant.

B.8. (!) Consider an instance of TSP where the costs satisfy the triangle inequality. Prove that some matching of the odd-degree vertices in a minimum spanning tree uses edges with total cost at most half the minimum cost of a spanning cycle. Conclude that Christofides' algorithm has performance ratio at most 3/2.

B.9. Prove that in order to solve the TSP exactly, it suffices to have an algorithm that solves the TSP when the edge weights satisfy the triangle inequality. (Hint: Given an arbitrary instance of the TSP, produce in polynomial time an instance of the TSP where the edge weights satisfy the triangle inequality ($w_{ij} + w_{jk} \geq w_{ik}$) and the set of optimal tours is the same as in the original instance.)

B.10. Prove that 2-SAT belongs to P.

B.11. A road system has one snowplow. The narrow city streets must be plowed; the plow can clear such streets completely in one traversal. There are also rural roads that the plow can use to change position, but these don't need to be plowed. Thus we have a weighted graph with edges of two types; those of type 1 must be traversed, those of type 2 need not be. The state wants an algorithm that will find a minimum-length circuit traversing all type 1 edges in such a graph. Prove that this problem is NP-hard, by reduction from the HAMILTONIAN CYCLE problem.

B.12. (!) *Heuristic algorithms for vertex covering.* Algorithm 1: include a vertex of maximum remaining degree, delete, iterate until the remaining graph is a stable set. Algorithm 2: choose an arbitrary edge, include both endpoints, delete them, iterate until the remaining graph is a stable set. The heuristic in Algorithm 1 may seem more powerful, but Algorithm 2 has a better performance guarantee!

a) Prove that algorithm 2 always produces a vertex cover with size at most twice the minimum size $\beta(G)$.

b) Prove that algorithm 1 may produce a vertex cover of size about $\log \beta(G)$ times the minimum. (Hint: Construct a bipartite graph G for which Algorithm 1 chooses about $\beta(G)/i$ vertices of degree i for each $1 \leq i \leq \beta(G)$.)

B.13. (+) A graph G is **α-critical** if $\alpha(G-e) > \alpha(G)$ for every $e \in E(G)$. Prove that a connected α-critical graph has no cut-vertex. (Hint: If e_1, e_2 are edges incident to a cut-vertex x, use the maximum independent sets in $G-e_1$ and $G-e_2$ to build an independent set in G with more than $\alpha(G)$ vertices.)

B.14. SATISFIABILITY differs from 3-SAT in that clauses may have arbitrary size. Prove that a clause containing more than three literals can be replaced by clauses with three literals (allowing the addition of a few new variables) so that the original clause is satisfiable if and only if the new instance of 3-SAT is satisfiable. Conclude that SATISFIABILITY reduces to 3-SAT. (Karp [1972])

B.15. (!) Given that HAMILTONIAN CYCLE is NP-complete for 3-regular graphs, prove that COVERING CIRCUIT is NP-complete. This is the question of whether the input graph G contains a closed trail that includes at least one endpoint of every edge. Prove that the line graph of G is Hamiltonian if and only if G has a covering circuit. Conclude that HAMILTONIAN CYCLE is NP-complete for line graphs.

B.16. The DOMINATING SET problem considers an input graph H and an input integer k and asks whether H has a dominating set of size at most k (Definition 3.1.26).

a) Given a graph G, let G' be the graph obtained by adding an extra copy of each edge of G and subdividing one copy of each edge (thus each edge is replaced by a triangle involving a new vertex). Prove that G has a vertex cover of size at most k if and only if G' has a dominating set of size at most k.

b) Use the NP-completeness of VERTEX COVER to prove the NP-completeness of DOMINATING SET.

B.17. Prove the claims made in Theorem B.12 about 3-colorings of the gadget H.

B.18. (*) Use HAMILTONIAN PATH in directed graphs to prove that 3-MATROID INTERSECTION is NP-complete.

B.19. (*) Use 3-D MATCHING to prove that 3-MATROID INTERSECTION is NP-complete. Given a collection of triples of the form (x_1, x_2, x_3) with $x_i \in V_i$ and sets V_1, V_2, V_3 disjoint, 3-D MATCHING is the problem of finding the maximum number of triples such that each element appears in at most one of the triples selected (by comparison, bipartite matching is the 2-D matching problem with the sets being pairs).

Appendix C

Hints for Selected Exercises

In this appendix we provide some general guidelines and some specific suggestions for selected exercises. This should help students who have trouble getting started in finding and writing proofs.

GENERAL DISCUSSION

The first step is making sure that one understands exactly what the problem is asking. Some problems request a verification of a mathematical statement. Definitions may provide a road map for what needs to be verified. Sometimes, the desired statement follows from a theorem already proved, and then one needs to verify that its hypotheses hold.

Other problems may involve some experimentation to discover the mathematical statement that needs to be proved. Sometimes one examines small cases to discover a general pattern and then proves that pattern by induction. In other problems, exploration of examples can help one understand why the claim is true.

Understand definitions and use them carefully. A disconnected graph need not have an isolated vertex. Loops and cycles are different concepts. Understand the difference between *maximal* and *maximum*. A vertex cover is a set of vertices, and an edge cover is a set of edges. A graph with connectivity 3 is 2-connected, and a 3-chromatic graph is 17-colorable.

When seeking a direct proof of a conditional statement, one can work from both ends. List statements that follow from the hypothesis. List statements that suffice to imply the conclusion. When some statement appears in both lists, the problem is solved.

When unsuccessful with the direct method, list what would follow if the conclusion were not true. If something in this list contradicts something in the list of statements that follow from the hypothesis (or other known true statements), then again the problem is solved, using the method of contradiction.

Contradiction is particularly appropriate for statements of impossibility. To prove that something exists, often one can construct an example and prove that it has the desired properties; this is the direct method.

Most conditional statements can be interpreted as universally quantified statements. The proof of a universally quantified statement must be valid for every value of the variable in the given universe. Examples can help one understand or explain a proof, but an example by itself does not provide a proof. Explaining why the example has the desired property, in language that applies to all possible examples, can lead to the proof of the desired statement.

Induction often works to prove statements that have a natural number parameter. Beware of the induction trap|—(Example 1.3.26, Example A.28). Remember the template for proving conditional statements by induction (Remark 1.3.25). Exploration of small cases can help one understand either the statement to be proved or the way to go from one value of the parameter to the next, but such discussion is *not* part of the final proof except as needed in the basis step.

Other techniques include extremality and the pigeonhole principle. Sometimes one considers a smallest counterexample to a desired statement and then uses its existence to obtain a smaller counterexample. This can be viewed as induction or contradiction or extremality.

It may not be obvious what technique works in a particular problem. Sometimes many different techniques work and produce different proofs. Mathematicians find proofs by working hard; both stubbornness and flexibility are virtues. One tries all imaginable techniques to solve a problem. Practice increases understanding and speed in finding proofs.

The final step is to produce a careful and complete exposition of the solution. Writing out a proof can reveal hidden subtleties or cases that have been overlooked. It can also expose thoughts that turn out to be irrelevant. Producing a well-written solution often involves repeated revision. It is helpful to write a solution early, put it aside, and read it again before submission to see whether it is still complete, convincing, and comprehensible. The process of writing solutions helps develop a useful skill: the ability to express oneself concisely, clearly, and accurately.

SUPPLEMENTAL SPECIFIC HINTS

1.1.14. Contradiction often helps with a nonexistence conclusion. Suppose that such a decomposition exists; what conclusions can be drawn about the board?

1.1.25. Another nonexistence conclusion. Assume that a 7-cycle exists, and use the properties of the Petersen graph to obtain a contradiction.

1.1.26. Consider an edge in G (one can also start with a vertex).

1.1.27. Start with a vertex in G.

1.1.29. Consider the acquaintances or nonacquaintances (whichever set is larger) of one particular person.

1.1.32. Consider the parity of the sizes of the partite sets.

1.1.34. Since the three subgraphs are to be pairwise isomorphic, it will help to examine a drawing of the graph that has three-fold symmetry.

1.1.37. Compare contributions from the ends of the paths with contributions from the internal vertices.

1.1.38. Get a decomposition from a bipartition and a bipartition from a decomposition.

1.2.15. In this problem the starting vertex is specified, so having the first and last edge be the same is not enough.

1.2.17. Use the transitivity of the connection relation.

1.2.18. Draw G for small values of n to see what the answer should be; then prove it.

1.2.19. For the upper bound, use the fact that when a and b are relatively prime, there are integers p and q such that $pa + qb = 1$.

1.2.26. The characterization of bipartite graphs makes this easy.

1.2.28. The problem does not restrict attention to induced subgraphs.

1.2.38. Use induction and Lemma 1.2.25.

1.2.40. If P and Q are disjoint, then consider a shortest path from $V(P)$ to $V(Q)$, and obtain a contradiction.

1.3.12. It may help to construct the example first in the case $k = 1$ and then generalize.

1.3.15. For part (b), consider the complements.

1.3.18. Assuming that e is a cut-edge and H is a component of $G - e$, count the edges of H from the viewpoint of each partite set. Setting these formulas equal leads to a contradiction.

1.3.19. For the second part, make the desired graphs correspond to 3-regular graphs.

1.3.22. In part (a), what happens if an outside vertex has three neighbors in $V(C)$? For part (b), part (a) provides one bound on the number of edges between $V(C)$ and $V(G) - V(C)$; the hypothesis on minimum degree provides another.

1.3.28. When k is even, exhibit an isomorphism. When k is odd, find an odd cycle in Q'_k.

1.3.31. For part (a), consider Example 1.3.18.

1.3.33. For part (a), establish a one-to-one correspondence between the nonneighbors of x and the pairs of neighbors of x.

1.3.34. Consider two adjacent vertices x, y, and establish a one-to-one correspondence between $N(x)$ and $N(y)$.

1.3.43. For the construction, a regular graph can't work. Vertices of high and low degrees are needed, but both must be adjacent to high-degree vertices.

1.3.50. Construct for each n an example with few edges, and use induction on n to prove that it is optimal. The degree-sum formula implies that an n-vertex simple graph with fewer than $3n/2$ edges has a vertex of degree at most 2.

1.3.53. Define a graph to model pairs of people who can still play together. What is the condition that permits an additional game to be played?

1.3.55. For part (a), show first that $\Delta(G) \geq n(G)/2$. For part (b), show that $\overline{G}$ must be disconnected.

1.3.57. Keep the paradigm of Remark 1.3.25 in mind.

1.3.63. Any inductive proof of sufficiency must verify that the "smaller object" satisfies the condition before the induction hypothesis can be applied.

1.4.16. For part (a), keep the definition of l in mind.

1.4.23. The shortest proof uses a graph transformation.

1.4.25. Given an orientation with more than two vertices of odd outdegree, make an appropriate alteration.

1.4.29. Use the strong connectedness of D and the reference odd cycle of G to build an odd cycle in D.

1.4.34. Show that in the subgraph F of G consisting of edges oriented oppositely in H, indegree equals outdegree at every vertex. Bring G closer to H by finding a 3-cycle reversal involving a vertex of maximum outdegree in F.

1.4.37. Use strong induction on the order of the tournament.

1.4.38. In one direction, show that there is such a tournament with n vertices if there is one with $n-2$ vertices. Be careful about $n = 6$.

2.1.2. In part (b), the statement includes the possibility of adding multiple copies of edges already present.

2.1.17. Compare with the proof of A⇒B,C in Theorem 2.1.4.

2.1.25. Use induction on n; for the induction step, delete a leaf.

2.1.27. Keep the quantifier in mind. Prove that the condition on the list of numbers is both necessary and sufficiency for the *existence* of a tree with vertex degrees $d_1, \ldots, d_n$. Two implications must be proved.

2.1.29. Count the edges in two ways.

2.1.31. Consider the contrapositive.

2.1.33. Two implications must be proved. Each concerns a connected n-vertex graph.

2.1.34. Use induction on n.

2.1.40. Express G as a union of the right number of paths. What can be done if they are not pairwise intersecting?

2.1.41. Use a spanning tree of some component.

2.1.47. In part (a), remember that the union of a u, v-path and a v, w-path need not be a u, w-path.

2.1.59. In the problem, both n and k are fixed. The answer must be given in terms of these two parameters.

2.1.61. Form G' from $G - x - y$ by adding k disjoint edges joining $N_G(x)$ to $N_G(y)$.

2.2.5. Consider for each 5-cycle the number of edges that will be used.

2.2.7. By symmetry, each edge of K_n lies in the same number of spanning trees of K_n.

2.2.9. Use the Prüfer code.

2.2.19. In a tree with vertex set $[n]$, cut the edge at vertex n on the path from n to 1.

2.2.24. Build the edges in decreasing order of the difference between their endpoints.

2.2.29. If a tree is not a caterpillar, then it contains the tree Y of Example 2.2.18.

2.2.33. Consider the set of vertices reachable by paths from the root.

2.3.11. What happens if a minimum-weight spanning tree contains an edge of weight larger that the maximum weight in a bottleneck spanning tree?

2.3.13. Consider the heaviest edge of T' among those not in T.

2.3.31. In the induction step, partition the set of words in an optimal code into two sets according to the first bit.

3.1.8. Consider the symmetric difference of two perfect matchings.

3.1.9. Use vertex covers, or compare a maximal matching with a maximum matching using symmetric difference.

3.1.16. Use induction on k.

3.1.24. Transform this into a graph problem.

3.1.25. Find one appropriate permutation matrix and then adjust what remains by a constant factor to apply the induction hypothesis.

3.1.26. Use Corollary 3.1.13 for part (a) and induction on n for part (b).

3.1.29. What does the König–Egerváry theorem say about the vertex cover problem when G has no matching of size k.

3.1.30. This requires a bound and an example that achieves the bound.

3.1.39. Consider the edges joining a maximum independent set to its complement.

3.2.11. Suppose that the first occurrence of such a rejection in the Proposal Algorithm is a rejecting x even though xa is a pair in some stable matching M. If a rejects x for y, note that y must be paired with some women b in M. What can be deduced about the preferences of these people?

3.3.2. Vertex cover is not strong enough to prove optimality of the matching.

3.3.7. This is easy when k is even. For odd k, construct an example for $k = 3$ and generalize it.

3.3.11. Use Tutte's Theorem.

3.3.12. To prove that there is a matching whose size is the weight of some generalized cover, let T be a maximal set of vertices maximizing the quantity $o(G - T) - |T|$ (the deficiency).

3.3.14. Show for each $S \subseteq V(G)$ that $o(G - S) - |S|$ is sufficiently small. A different approach is to let X be the k-vertex set inducing the fewest edges and prove that there is a matching from X into $\overline{X}$.

3.3.16. Generalize the argument of Corollary 3.3.8.

3.3.17. Given adjacent vertices x, y, verify Tutte's condition for $G - x - y$.

3.3.18. Think of an appropriate size of a Tutte set S and an appropriate number of odd components in $G - S$.

3.3.19. Use Petersen's Theorem to obtain a 1-factor. Assemble copies of P_4 by considering a consistent orientation of the cycles in the remaining 2-factor.

4.1.5. Show that G' is connected and has no cut-vertex. There is also a short proof using internally disjoint paths:

4.1.10. Theorem 4.1.11 and Corollary 1.3.5 yield a short proof.

4.1.14. Prove the contrapositive.

4.1.17. Show first that if $\left|[S, \overline{S}]\right| = 3$, then S and $\overline{S}$ have odd size.

4.1.18. For the first part, show that the subgraph induced by the smaller side of an edge cut with at most two edges has too many edges to avoid having a triangle.

4.1.23. Verify Tutte's Condition. Keep in mind that the forbidden condition is large stars as induced subgraphs, not just any large stars.

4.1.26. Necessity is proved by following individual cycles. For sufficiency, define an appropriate auxiliary graph whose vertices are the components of $G - F$. Show that this graph is bipartite to obtain the desired partition of the components of $G - F$.

4.1.27. Recall that an edge cut is a bond if and only if on each side of the cut the vertices induce a connected subgraph.

4.2.6. Use ear decomposition.

4.2.14. This is essentially an edge version of Theorem 4.2.2, with the added conclusion that the common points happen in the same order on the two paths.

4.2.21. What do we know about connected graphs having at most two vertices of odd degree?

4.2.23. From a given bipartite graph G, design a graph H so that applying Menger's Theorem to H will yield the needed result on G.

4.2.28. Use the Expansion Lemma and Menger's Theorem.

4.3.13. Design a network so that there is an assignment of participants to groups if and only if the network has a flow of value $\sum m_i$. Use the Ford–Fulkerson Theorem to express the condition for such a flow in terms of cuts. Show that the given condition on the data is equivalent to that condition on the source/sink cuts.

5.1.20. One approach is to prove the contrapositive. Another is to delete an odd cycle.

5.1.22. An ordering is needed so that when each vertex is encountered, it has at most two neighbors among the earlier vertices.

5.1.23. The lower bounds involve counting and/or the pigeonhole principle. The interesting part is providing a construction to show that the graph is $k+2$-colorable with $k+1$ does not divide n.

5.1.26. (also next problem) Obtain a clique and a proper coloring of the same size.

5.1.30. Given a proper r-coloring of G_n, produce a collection of distinct subsets of the colors, each subset associated with an element of n. Conversely, show how to use such a collection to produce a proper coloring.

5.1.31. From a proper m-coloring, build an independent set of the same size. From a big enough independent set, construct a proper m-coloring.

5.1.32. As in Theorem 1.2.23, one can use induction or encode the colors as binary k-tuples and apply the pigeonhole principle.

5.1.39. Obtain a quadratic inequality for k in terms of m by using upper and lower bounds on $e(G)$.

5.1.41. Using the induction hypothesis, a new graph can't violate the bound unless we delete a vertex and find that the chromatic numbers of both the graph and its complement decrease. Is it possible for this to happen when these chromatic numbers already sum to the maximum?

5.1.44. Obtain the upper bound from part (a). For the lower bound, use the same orientation as in Theorem 5.1.21.

5.1.51. Modify some proper k-coloring so that it becomes a proper $k+1$-coloring that has the pre-specified values on the vertices of P.

5.2.2. Consider the complement.

5.2.9. Since G' is connected, it suffices to consider deletion of edges from G'; see Remark 5.2.12.

5.2.15. Use large neighborhoods as color classes while there remain vertices of high degree; then apply Brooks' Theorem.

5.2.17. For part (b), consider complements.