7.7 Show that NP is closed under union and concatenation.

7.8 Let $CONNECTED = \{\langle G \rangle | \ G \text{ is a connected undirected graph}\}$. Analyze the algorithm given on page 157 to show that this language is in P.

7.9 A ***triangle*** in an undirected graph is a 3-clique. Show that $TRIANGLE \in$ P, where $TRIANGLE = \{\langle G \rangle | \ G \text{ contains a triangle}\}$.

7.10 Show that ALL_{DFA} is in P.

7.11 Call graphs G and H ***isomorphic*** if the nodes of G may be reordered so that it is identical to H. Let $ISO = \{\langle G, H \rangle | \ G \text{ and } H \text{ are isomorphic graphs}\}$. Show that $ISO \in$ NP.

PROBLEMS

7.12 Let

$$MODEXP = \{\langle a, b, c, p \rangle | \ a, b, c, \text{ and } p \text{ are binary integers such that } a^b \equiv c \pmod{p}\}.$$

Show that $MODEXP \in$ P. (Note that the most obvious algorithm doesn't run in polynomial time. Hint: Try it first where b is a power of 2.)

7.13 A ***permutation*** on the set $\{1, \ldots, k\}$ is a one-to-one, onto function on this set. When p is a permutation, p^t means the composition of p with itself t times. Let

$$PERM\text{-}POWER = \{\langle p, q, t \rangle | \ p = q^t \text{ where } p \text{ and } q \text{ are permutations on } \{1, \ldots, k\} \text{ and } t \text{ is a binary integer}\}.$$

Show that $PERM\text{-}POWER \in$ P. (Note that the most obvious algorithm doesn't run within polynomial time. Hint: First try it where t is a power of 2).

7.14 Show that P is closed under the star operation. (Hint: Use dynamic programming. On input $y = y_1 \cdots y_n$ for $y_i \in \Sigma$, build a table indicating for each $i \leq j$ whether the substring $y_i \cdots y_j \in A^*$ for any $A \in$ P.)

A7.15 Show that NP is closed under the star operation.

7.16 Let *UNARY-SSUM* be the subset sum problem in which all numbers are represented in unary. Why does the NP-completeness proof for *SUBSET-SUM* fail to show *UNARY-SSUM* is NP-complete? Show that *UNARY-SSUM* $\in$ P.

7.17 Show that, if P = NP, then every language $A \in$ P, except $A = \emptyset$ and $A = \Sigma^*$, is NP-complete.

***7.18** Show that $PRIMES = \{m | m \text{ is a prime number in binary}\} \in$ NP. (Hint: For $p > 1$ the multiplicative group $Z_p^* = \{x | \ x \text{ is relatively prime to } p \text{ and } 1 \leq x < p\}$ is both cyclic and of order $p - 1$ iff p is prime. You may use this fact without justifying it. The stronger statement $PRIMES \in$ P is now known to be true, but it is more difficult to prove.)

7.19 We generally believe that *PATH* is not NP-complete. Explain the reason behind this belief. Show that proving *PATH* is not NP-complete would prove P $\neq$ NP.

7.20 Let G represent an undirected graph. Also let

$$SPATH = \{\langle G, a, b, k\rangle \mid G \text{ contains a simple path of length at most } k \text{ from } a \text{ to } b\},$$

and

$$LPATH = \{\langle G, a, b, k\rangle \mid G \text{ contains a simple path of length at least } k \text{ from } a \text{ to } b\}.$$

a. Show that $SPATH \in$ P.

b. Show that *LPATH* is NP-complete. You may assume the NP-completeness of *UHAMPATH*, the Hamiltonian path problem for undirected graphs.

7.21 Let $DOUBLE\text{-}SAT = \{\langle \phi \rangle \mid \phi \text{ has at least two satisfying assignments}\}$. Show that *DOUBLE-SAT* is NP-complete.

A7.22 Let $HALF\text{-}CLIQUE = \{\langle G \rangle \mid G$ is an undirected graph having a complete subgraph with at least $m/2$ nodes, where m is the number of nodes in $G\}$. Show that *HALF-CLIQUE* is NP-complete.

7.23 Let $CNF_k = \{\langle \phi \rangle \mid \phi$ is a satisfiable cnf-formula where each variable appears in at most k places$\}$.

a. Show that $CNF_2 \in$ P.

b. Show that CNF_3 is NP-complete.

7.24 Let ϕ be a 3cnf-formula. An ***≠-assignment*** to the variables of ϕ is one where each clause contains two literals with unequal truth values. In other words, an $\neq$-assignment satisfies ϕ without assigning three true literals in any clause.

a. Show that the negation of any $\neq$-assignment to ϕ is also an $\neq$-assignment.

b. Let $\neq SAT$ be the collection of 3cnf-formulas that have an $\neq$-assignment. Show that we obtain a polynomial time reduction from *3SAT* to $\neq SAT$ by replacing each clause c_i

$$(y_1 \vee y_2 \vee y_3)$$

with the two clauses

$$(y_1 \vee y_2 \vee z_i) \quad \text{and} \quad (\overline{z_i} \vee y_3 \vee b),$$

where z_i is a new variable for each clause c_i and b is a single additional new variable.

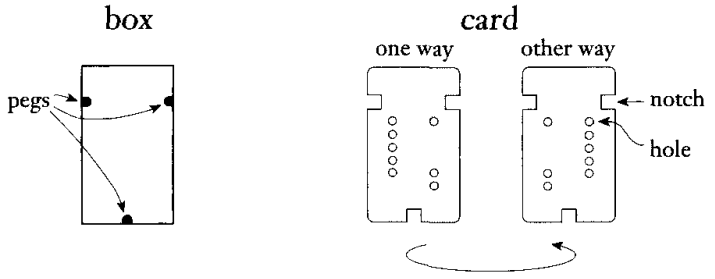
c. Conclude that $\neq SAT$ is NP-complete.

7.25 A ***cut*** in an undirected graph is a separation of the vertices V into two disjoint subsets S and T. The size of a cut is the number of edges that have one endpoint in S and the other in T. Let

$$MAX\text{-}CUT = \{\langle G, k\rangle \mid G \text{ has a cut of size } k \text{ or more}\}.$$

Show that *MAX-CUT* is NP-complete. You may assume the result of Problem 7.24. (Hint: Show that $\neq SAT \leq_P MAX\text{-}CUT$. The variable gadget for variable x is a collection of $3c$ nodes labeled with x and another $3c$ nodes labeled with $\overline{x}$, where c is the number of clauses. All nodes labeled x are connected with all nodes labeled $\overline{x}$. The clause gadget is a triangle of three edges connecting three nodes labeled with the literals appearing in the clause. Do not use the same node in more than one clause gadget. Prove that this reduction works.)

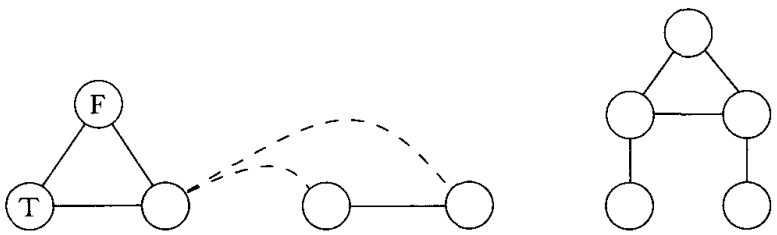
7.26 You are given a box and a collection of cards as indicated in the following figure. Because of the pegs in the box and the notches in the cards, each card will fit in the box in either of two ways. Each card contains two columns of holes, some of which may not be punched out. The puzzle is solved by placing all the cards in the box so as to completely cover the bottom of the box, (i.e., every hole position is blocked by at least one card that has no hole there.) Let $PUZZLE = \{\langle c_1, \ldots, c_k\rangle |$ each c_i represents a card and this collection of cards has a solution$\}$. Show that *PUZZLE* is NP-complete.



7.27 A ***coloring*** of a graph is an assignment of colors to its nodes so that no two adjacent nodes are assigned the same color. Let

$3COLOR = \{\langle G\rangle |$ the nodes of G can be colored with three colors such that no two nodes joined by an edge have the same color$\}$.

Show that *3COLOR* is NP-complete. (Hint: Use the following three subgraphs.)



7.28 Let $SET\text{-}SPLITTING = \{\langle S, C\rangle |$ S is a finite set and $C = \{C_1, \ldots, C_k\}$ is a collection of subsets of S, for some $k > 0$, such that elements of S can be colored *red* or *blue* so that no C_i has all its elements colored with the same color.$\}$ Show that *SET-SPLITTING* is NP-complete.

7.29 Consider the following scheduling problem. You are given a list of final exams $F_1, \ldots, F_k$ to be scheduled, and a list of students $S_1, \ldots, S_l$. Each student is taking some specified subset of these exams. You must schedule these exams into slots so that no student is required to take two exams in the same slot. The problem is to determine if such a schedule exists that uses only h slots. Formulate this problem as a language and show that this language is NP-complete.

7.30 This problem is inspired by the single-player game *Minesweeper*, generalized to an arbitrary graph. Let G be an undirected graph, where each node either contains a single, hidden *mine* or is empty. The player chooses nodes, one by one. If the player chooses a node containing a mine, the player loses. If the player chooses an empty node, the player learns the number of neighboring nodes containing mines. (A neighboring node is one connected to the chosen node by an edge.). The player wins if and when all empty nodes have been so chosen.

In the *mine consistency problem* you are given a graph G, along with numbers labeling some of G's nodes. You must determine whether a placement of mines on the remaining nodes is possible, so that any node v that is labeled m has exactly m neighboring nodes containing mines. Formulate this problem as a language and show that it is NP-complete.

A**7.31** In the following solitaire game, you are given an $m \times m$ board. On each of its n^2 positions lies either a blue stone, a red stone, or nothing at all. You play by removing stones from the board so that each column contains only stones of a single color and each row contains at least one stone. You win if you achieve this objective. Winning may or may not be possible, depending upon the initial configuration. Let $SOLITAIRE = \{\langle G\rangle \mid G \text{ is a winnable game configuration}\}$. Prove that *SOLITAIRE* is NP-complete.

7.32 Let $U = \{\langle M, x, \#^t\rangle \mid \text{TM } M \text{ accepts input } x \text{ within } t \text{ steps on at least one branch}\}$. Show that U is NP-complete.

7.33 Recall, in our discussion of the Church–Turing thesis, that we introduced the language $D = \{\langle p\rangle \mid p \text{ is a polynomial in several variables having an integral root}\}$. We stated, but didn't prove, that D is undecidable. In this problem you are to prove a different property of D—namely, that D is NP-hard. A problem is ***NP-hard*** if all problems in NP are polynomial time reducible to it, even though it may not be in NP itself. So, you must show that all problems in NP are polynomial time reducible to D.

7.34 A subset of the nodes of a graph G is a ***dominating set*** if every other node of G is adjacent to some node in the subset. Let

$$DOMINATING\text{-}SET = \{\langle G, k\rangle \mid G \text{ has a dominating set with } k \text{ nodes}\}.$$

Show that it is NP-complete by giving a reduction from *VERTEX-COVER*.

*7.35 Show that the following problem is NP-complete. You are given a set of states $Q = \{q_0, q_1, \ldots, q_l\}$ and a collection of pairs $\{(s_1, r_1), \ldots, (s_k, r_k)\}$ where the s_i are distinct strings over $\Sigma = \{0, 1\}$, and the r_i are (not necessarily distinct) members of Q. Determine whether a DFA $M = (Q, \Sigma, \delta, q_0, F)$ exists where $\delta(q_0, s_i) = r_i$ for each i. Here, $\delta(q, s)$ is the state that M enters after reading s, starting at state q. (Note that F is irrelevant here).

*7.36 Show that if P = NP, a polynomial time algorithm exists that produces a satisfying assignment when given a satisfiable Boolean formula. (Note: The algorithm you are asked to provide computes a function, but NP contains languages, not functions. The P = NP assumption implies that *SAT* is in P, so testing satisfiability is solvable in polynomial time. But the assumption doesn't say how this test is done, and the test may not reveal satisfying assignments. You must show that you can find them anyway. Hint: Use the satisfiability tester repeatedly to find the assignment bit-by-bit.)