Final

(June 15, 2000)

Note

This is a closed-book exam. Each problem accounts for 10 points, unless otherwise marked.

Problems

1. Design an efficient algorithm to determine whether two sorted arrays have a common element. Present your algorithm in a reasonable pseudocode. State the time complexity of your algorithm in terms of the sizes m and n of the given arrays. The more efficient your algorithm is, the more points you get for this problem.

2. Compute the *next* table as in the KMP algorithm for the string $B[1..10] = bababbabaa$. Show the details of calculation for $B[9]$ and $B[10]$.

3. Given two strings $A = aabcb$ and $B = acabbb$, compute the minimal cost matrix $C[0..5, 0..6]$ for changing A character by character to B. Show the detail of calculation for the entry $C[5, 6]$. (15 points)

4. Below is an algorithm skeleton for depth-first search utilizing a stack; assume that the input graph is connected. Modify it to obtain an algorithm for recording a DFS tree of the input graph. You should try not to change the overall structure of the original algorithm.

   Algorithm Simple_Nonrecursive_DFS (G, v);
   begin
       push v to $Stack$;
       while $Stack$ is not empty **do**
           pop vertex w from $Stack$;
           if w is unmarked **then**
               mark w;
               for all edges (w, x) such that x is unmarked **do**

push x to *Stack*

end

5. Design an algorithm that, given as input a directed acyclic graph G, finds a simple path in G with the maximum number of edges among all simple paths of G. Your algorithm should run in linear time.

6. Let $G = (V, E)$ be a connected undirected graph. A set $F \subseteq E$ is called a *feedback-edge set* if every cycle of G has at least one edge in F. Design an algorithm to find a minimum-size feedback-edge set. It is sufficient to describe the main idea of the algorithm. (Hint: think about $E - F$.)

7. A connected undirected graph is called *edge-biconnected* if the graph remains connected after the removal of any edge.

   (a) Find a graph that is edge-biconnected but not biconnected. (5 points)

   (b) Design a linear-time algorithm to determine whether a graph is edge-biconnected. It is sufficient to describe the main idea of the algorithm. (10 points)

8. Solve any two of the following three problems. (Note: If you try to solve all three problems, I will randomly pick two of them to grade.) (20 points)

   (a) A *dominating set* D of a graph $G = (V, E)$ is a subset of V such that every member of V is either in D or is adjacent to some vertex in D. The dominating set problem is as follows.

      Given an undirected graph G and an integer k, determine whether G has a dominating set containing $\leq k$ vertices.

   Prove that the dominating set problem is NP-complete.

   (b) The traveling salesman problem is as follows.

      The input includes a set of cities, the distances between all pairs of cities, and a number D. The problem is to determine whether there exists a (traveling-salesman) tour of all the cities having total length $\leq D$.

   Prove that the traveling salesman problem is NP-complete.

   (c) The knapsack problem is as follows.

      Given a set X, where each element $x \in X$ has an associated size $s(x)$ and value $v(x)$, and two other numbers S and V, is there a subset $B \subseteq X$ whose total size is $\leq S$ and whose total value is $\geq V$?

Prove that the knapsack problem is NP-complete.

Appendix

- The vertex cover problem: given an undirected graph $G = (V, E)$ and an integer k, determine whether G has a vertex cover containing $\leq k$ vertices. A *vertex cover* of G is a set of vertices such that every edge in G is incident to at least one of these vertices.
  The problem is NP-complete.

- The partition problem: given a set X where each element $x \in X$ has an associated size $s(x)$, is it possible to partition the set into two subsets with exactly the same total size?
  The problem is NP-complete.

- The Hamiltonian cycle problem: given a graph G, does G contain a Hamiltonian cycle? (A Hamiltonian cycle in a graph is a cycle that contains each vertex, except the starting vertex of the cycle, exactly once.)
  The problem is NP-complete.