Final

(June 14, 2001)

Note

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

Problems

1. Below is an algorithm for solving a variant of the Towers of Hanoi puzzle with an additional fourth peg D; `Towers_Hanoi` is an algorithm for the original puzzle.

```
Algorithm Four_Towers_Hanoi(A,B,C,D,n);
begin
   if n<=2 then
      Towers_Hanoi(A,B,C,n);
   else
      Four_Towers_Hanoi(A,D,B,C,n-2);
      Towers_Hanoi(A,B,C,2);
      Four_Towers_Hanoi(D,B,C,A,n-2);
end;
```

   Let $T(n)$ denote the number of moves needed for n disks. Write a recurrence relation for $T(n)$ and solve it.

2. What is the *next* table, as in the KMP algorithm, for the string $B[1..12] = abbaabbaabab$? Please show the details of calculation for $next[10]$ and $next[12]$.

3. Below is an algorithm skeleton for depth-first search utilizing a stack; assume that the input graph is connected. Modify the algorithm so that it prints out (the edges of) 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
```

4. Design an algorithm for determining whether a given *acyclic* directed graph $G = (V, E)$ contains a directed Hamiltonian path. (Note: A directed *Hamiltonian path* of a directed graph is a simple directed path that includes all vertices of the graph. An acyclic directed graph is one without directed cycles.) The more efficient your algorithm is, the more points you get for this problem. Explain why the algorithm is correct and give an analysis of its time complexity.

5. Let $G = (V, E)$ be a connected weighted undirected graph and T be a minimum-cost spanning tree (MCST) of G. Suppose that the cost of one edge $\{u, v\}$ in G is *decreased*; $\{u, v\}$ may or may not belong to T. Design an algorithm either to find a new MCST or to determine that T is still an MCST. The more efficient your algorithm is, the more points you get for this problem. Explain why your algorithm is correct and analyze its time complexity.

6. Below is the main procedure for determining the biconnected components of an undirected graph. Is the algorithm still correct if we replace the last second line "$v.high := \max(v.high, w.DFS\_Number)$" by "$v.high := \max(v.high, w.high)$"? Why? Please explain.

```
procedure BC(v);
begin
    v.DFS_Number := DFS_N;
    DFS_N := DFS_N − 1;
    v.high := v.DFS_Number;
    for all edges (v, w) do
        if w is not the parent of v then
```

```
        insert (v, w) into Stack;
        if w.DFS_Number = 0 then
            BC(w);
            if w.high ≤ v.DFS_Number then
                remove all edges from Stack
                    until (v, w) is reached;
            v.high := max(v.high, w.high)
        else
            v.high := max(v.high, w.DFS_Number)
end
```

7. A connected undirected graph is called *edge-biconnected* if the graph remains connected after the removal of any edge. (Recall that a graph is *biconnected* if the graph remains connected after the removal of an arbitrary vertex and all edges incident to the vertex.)

   (a) Are biconnected graphs always edge-biconnected? Why? (3 points)

   (b) Draw a graph that is edge-biconnected but not biconnected. (3 points)

   (c) Describe the characteristics of edge-biconnected graphs that are useful for designing an algorithm that determines whether a graph is edge-biconnected. Is your characterization complete? (6 points)

   (d) Design an efficient algorithm, based on the characterization above, to determine whether a graph is edge-biconnected. It suffices to describe the main idea of the algorithm. (3 points)

8. Let $G = (V, E)$ be a directed graph and T be a DFS tree of G. Prove that the intersection of the edges of T with the edges of any strongly connected component of G form a subtree of T. (Hint: Assume toward contradiction that the intersection consists of two or more separate subtrees of T.)

9. Solve one of the following two problems. (Note: If you try to solve both problems, I will randomly pick one of them to grade.) (15 points)

   (a) The subgraph isomorphism problem is as follows.

   Given two graphs $G = (V_1, E_1)$ and $H = (V_2, E_2)$, does G contain a subgraph that is isomorphic to H? (Two graphs are isomorphic if there exists a one-one correspondence between the sets of vertices of the two graphs that preserve adjacency.)

Prove that the subgraph isomorphism problem is NP-complete.

(b) The bin packing problem is as follows.

> The input is a set of numbers $\{a_1, a_2, \cdots, a_n\}$ and two other numbers b and k. The problem is to determine whether the set can be partitioned into k subsets such that the sum of numbers in each subset is $\leq b$.

Prove that the bin packing problem is NP-complete.

Appendix

- 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.