Final

Note

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

Problems

1. Suppose that you are given an algorithm as a *black box* (you cannot see how it is designed) that has the following properties: If you input any sequence of real numbers and an **integer** k, the algorithm will answer "yes" or "no," indicating whether there exists a subset of the numbers whose sum is exactly k (an integer!). Show how to use this black box to find the subset (so as to list its members) whose sum is k, if it exists. You should use the black box $O(n)$ times (where n is the size of the sequence).

2. Compute the *next* table as in the KMP algorithm for the string $B[1..10] = bbabbabbba$. Please show how $next[9]$ and $next[10]$ are computed from using preceding entries of the table.

3. Solve the single-source shortest path problem using the dynamic programming approach (which we have described in class). You need only to define precisely a recurrence relation. Please explain why (or why not) your solution allows edges with a negative weight. (15 points)

4. Consider an $m \times n$ matrix of non-negative integers. A path through the matrix consists of a sequence of cells of the matrix, starting anywhere in the first column (Column 1) and terminating in the last column (Column n). Two consecutive cells in a path constitute a step of the path and should go from one column to the next column and either remain on the same row or go up or down one row. The weight of a path is the sum of the integers in the cells along the path. Design an algorithm to determine the minimum weight of such paths. Please present your algorithm in an adequate pseudo code and make assumptions wherever necessary. Give an analysis of its time complexity. The more efficient your algorithm is, the more points you get for this problem. (Hint: one possible approach is to reduce the problem into the standard shortest path problem in graphs.) (15 points)

5. For an arbitrary weighted graph G, prove that, if the weights of all its edges are distinct, then G has an unique minimum-cost spanning tree. (Note: the fact that a particular algorithm always finds just one minimum-cost spanning tree for a graph does not mean that the graph does not have another spanning tree of the same cost.)

6. The celebrated Prim's algorithm we discussed in class for finding the minimum-cost spanning tree of a graph is based on the following property:

   > Consider a weighted graph G and an arbitrary subgraph G' of graph G. Let $E(G', G)$ be the set of edges connecting nodes of G' to nodes in G but not in G'. If $E(G', G)$ is not empty, then the edge with the minimum weight in $E(G', G)$ must be in the minimum-cost spanning tree of G.

   Prove the correctness of the above property.

7. 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 $w.high \leq v.DFS\_Number$ 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
```

8. Describe how the maximum matching problem for bipartite graphs can be reduced to the network flow problem.

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.)

   (a) The hitting set problem is as follows.

   > Given a collection C of subsets of a set S and a positive integer k, does S contain a hitting set for C of size k or smaller, that is, a subset $S' \subseteq S$ with $|S'| \leq k$ such that S' contains at least one element from each subset in C?

   Prove that the hitting set 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 partition 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 partition problem is NP-complete.

- 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 vertex cover problem is complete.