Final

Note

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

Problems

1. Below is a theorem from Manber's book:

   > For all constants $c > 0$ and $a > 1$, and for all monotonically increasing functions $f(n)$, we have $(f(n))^c = O(a^{f(n)})$.

   Prove, by using the above theorem, that $n(\log n)^4 = O(n^{1.5})$.

2. 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 is a subset of the numbers whose sum is exactly k. Show how to use this black box to find the subset whose sum is k, if it exists. You should use the black box $O(n)$ times (where n is the size of the sequence).

3. Given two strings *aabcb* and *acabb*, compute the minimal cost matrix $C[0..5, 0..5]$ for changing the first string character by character to the second one. Show the detail of your calculation for the entry $C[5, 5]$.

4. Give a binary de Bruijn sequence of 2^4 bits. Explain how you can systematically produce the sequence.

5. Why Dijkstra's algorithm for single source shortest paths does not work when the input graph contains negative weight edges? (5 points)

6. Given as input a connected undirected graph G, a spanning tree T of G, and a vertex v, design an algorithm to determine whether T is a valid DFS tree of G rooted at v. In other words, determine whether T can be the output of DFS under some order of the edges starting with v.

7. Prove that if the costs of all edges in a given connected graph are distinct, then the graph has an unique minimum-cost spanning tree.

8. Give a weighted connected undirected graph whose minimum bottleneck weight spanning tree and minimum weight spanning tree are different. A bottleneck weight is defined as the maximum weight of an edge in the subgraph.

9. Consider the DFS-based algorithm for determining biconnected components of an undirected graph (see the appendix). Is the stack always empty when the algorithm terminates? Explain. (5 points)

10. The independent set problem is as follows.

    > An independent set in an undirected graph is a set of vertices no two of which are adjacent. The problem is to determine, given a graph G and an integer k, whether G contains an independent set with $\geq k$ vertices.

    Prove that the independent set problem is NP-complete.

11. 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

- Dijkstra's algorithm for single source shortest paths:

```
Algorithm Single_Source_Shortest_Paths (G, v);
begin
    for all vertices w do
        w.mark := false;
        w.SP := ∞;
    v.SP := 0;
    while there exists an unmarked vertex do
        let w be an unmarked vertex such that w.SP is minimal;
        w.mark := true;
```

for all edges (w, z) such that z is unmarked **do**
if $w.SP + length(w, z) < z.SP$ **then**
$z.SP := w.SP + length(w, z)$

end

- A DFS-based algorithm for determining biconnected components:

Algorithm Biconnected_Components (G, v, n);
begin
for every vertex w **do** $w.DFS\_Number := 0$;
$DFS\_N := n$;
$BC(v)$
end

procedure BC (v);
begin
$v.DFS\_Number := DFS\_N$; $DFS\_N := DFS\_N - 1$;
insert v into $Stack$;
$v.high := v.DFS\_Number$;
for all edges (v, w) **do**
insert (v, w) into $Stack$;
if w is not the parent of v **then**
if $w.DFS\_Number = 0$ **then**
$BC(w)$;
if $w.high \leq v.DFS\_Number$ **then**
remove all edges and vertices from $Stack$ until v
is reached (they form a biconnected component);
insert v back into $Stack$;
$v.high := \max(v.high, w.high)$
else $v.high := \max(v.high, w.DFS\_Number)$
end

- A binary de Bruijn sequence is a cyclic sequence of 2^n bits $a_1 a_2 \cdots a_{2^n}$ such that each binary code s of size n is represented somewhere in the sequence; that is, there exists a unique index i such that $s = a_i a_{i+1} \cdots a_{i+n-1}$ (where the indices are taken modulo 2^n).

- The clique problem: given an undirected graph $G = (V, E)$ and an integer k, does G contain a clique of size $\geq k$?
  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 exactly once.)
  The problem is NP-complete.