CSci 421
Introduction to Algorithms

Homework Assignment 4
Due: Thursday, 29 Jul 2004

Summer 2004 Handout 4

W. L. Ruzzo July 23, 2004

Reading Assignment:

Reading in Chapter 7: 7.1–7.4, 7.6 (although I gave a different algorithm in lecture), 7.9–7.11, 7.13.

Homework:

1. 7.32. Use breadth-first search.
2. Let $G = (V, E)$ be a connected undirected graph, and v a designated vertex of G. A depth first search of G starting at v may produce different results depending on the order in which edges are examined at each vertex. In particular, the set $T \subseteq E$ of edges categorized as "tree edges" by DFS (defining a spanning tree of G) may depend on the edge order.
   (a) Give an example of this, on a graph with $|V| \leq 5$.
   (b) We say a spanning tree T (with root v) "is a DFS tree for v" if there is an ordering of the edges of E incident to each vertex such that T is the tree constructed by DFS(v).
   Disprove: "For every connected undirected graph $G = (V, E)$, every vertex v in V and every spanning tree T with root v, 'T is a DFS tree for v' (as defined in part b)."
   Extra credit: Disprove the following slightly different statement: "For every connected undirected graph $G = (V, E)$ and every spanning tree T, there is a vertex v in V such that 'T is a DFS tree for v' (as defined in part b)."
   (c) Give a necessary and sufficient condition for a spanning tree T rooted at v to be a DFS tree for v.
   (d) Give an efficient algorithm ($O(|E| + |V|)$ if possible) to determine, given G, v and a spanning tree T, whether T is a DFS tree for v.
3. 7.14. You may use *in*creasing DFS numbers and LOW values (as in lecture) rather than decreasing numbers/HIGH values (as in text) if you prefer. Say which you're doing. *Note: Fig. 7.44 shows increasing DFS numbers.*
4. 7.83
5. 7.86

A couple of leftover dynamic programming/greedy problems, in case you're looking for something to do with all your spare time; both extra credit:

6. (Extra Credit) You are given a binary tree with n leaves, with the ith leaf labeled by a letter S_i from a fixed alphabet Σ. You are also given a function c, which assigns a cost $c(S, S') \geq 0$ to each pair of letters S, S' in Σ. Assume $c(S, S') = c(S', S)$ and $c(S, S) = 0$ for all $S, S' \in \Sigma$. The problem is to give each internal node x of the tree a label $S_x \in \Sigma$ so as to minimize the total over all tree edges of the cost of that edge, where the cost of an edge whose end points are labeled U and V is $c(U, V)$. Give an efficient algorithm to solve this problem, explain why it is correct, and analyze its running time as a function of n and $|\Sigma|$. Assume that each evaluation of the cost function c costs $O(1)$ operations; e.g., via table lookup using a table built in to your algorithm. Hint: use dynamic programming. As we've seen before, you might need a stronger induction hypothesis; i.e., you'll calculate more at each internal node than just whether to put letter "S" there.

   [Although it doesn't matter for purposes of this homework, this algorithm is sometimes used in estimating evolutionary trees. The S_i are letters at corresponding positions in DNA or protein sequences from n different modern species, the tree is their presumed evolutionary tree, and the goal is to try to reconstruct the corresponding letters in the ancestral sequences. For example, with the cost function $c(S, S') = (\text{ if } S == S' \text{ then } 0 \text{ else } 1)$, the min cost is simply counting the minimum number of "mutation" events in the evolutionary tree necessary to explain the observed variability in the modern species with respect to their presumed common ancestors.

7. (Extra Credit) Let G denote a connected undirected graph with weighted edges. The weights need not be distinct. A *cut* of G is simply a partition of its vertices into two nonempty subsets; an edge *crosses the cut* if one end point is in one part of the partition and the other endpoint is in the other part.

   Prove or disprove the following:
   Let e be an edge of G. Then the following statements are equivalent:

   (a) There exists a minimum spanning tree containing the edge e;

   (b) Every simple cycle containing e contains another edge at least as heavy as e;

   (c) There exists a cut crossed by e for which every edge crossing the cut is at least as heavy as e;