Problem Set 6
Due: Thursday, April 29, 1999
Half credit if turned on Tuesday, May 4, 1999

1. (1.5 Approximation algorithm for the TSP)

   The TSP (traveling salesmen problem) is, given a complete undirected graph G with non-negative edge weights, to find a cycle that contains all the vertices and has minimum total edge weight. (A graph is *complete* if it contains all possible edges.) We will assume that the edge weights satisfy the triangle inequality; namely, $w(x, z) \leq w(x, y) + w(y, z)$ for all vertices x, y, z.

   Our goal is to give a polynomial-time algorithm to find a cycle whose total edge weight is at most 1.5 times minimum. Here is the algorithm (described by Sanjeev Arora in class):

   1. Find a minimum spanning tree T.
   2. Let S be the set of all odd-degree vertices in T.
   3. Find a minimum-cost perfect matching M on S and add its edges to T.
   4. Find an Euler tour E on $T \cup S$ (a tree that traverses each edge exactly once).

   Let OPT be the weight of a minimum-weight tour.

   a) Prove that any graph has an even number of odd-degree vertices, and thus that S is of even size and has a perfect matching.

   b) Prove that $weight(M) \leq$ OPT/2 and $weight\ (E) \leq 1.5$ OPT.

   c) Assuming that a minimum-cost perfect matching on an n-vertex graph can be found in $O(n^3)$ time, describe an $O(n^3)$-time algorithm to compute E.

   d) The tour E may contain repeated vertices. Describe an $O(n^2)$-time algorithm to convert E into a cycle E' that contains each vertex exactly once, with $weight(E') \leq weight(E)$. Hint: use the triangle inequality.

2. Consider the following problem: Given a graph with non-negative integer edge weight, does it contain a biconnected spanning subgraph of total edge weight at most k? Prove that this problem is NP-complete. Hint: you may use the NP-completeness of any of the problems listed in Figure 26-11 of the text.

CS 423: Theory of Algorithms
Spring, 1999
Tarjan

Extra Credit Problems ~ Due Thursday, May 13

1. Red-black trees require two pointers per node plus a data item and a balance bit (red or black) per node. Describe a way to implement red-black trees with two pointers and one data item per node, without using explicit balance bits, and without increasing the time of any operation by more than constant factor. Hint: Find a way to encode the balance bits into the pointers.
2. Problem 27-2 text, p. 626
3. Problem 36.4-7 text, p. 946