15-451 Algorithms, Spring 2008

Homework # 6

Ground rules:

- This is an oral presentation assignment. You should work in groups of three. At some point before **Saturday, April 19 at 11:59pm** your group should sign up for a 1-hour time slot on the signup sheet on the course web page.
- Each person in the group must be able to present every problem. The TA/Professor will select who presents which problem. The other group members may assist the presenter.
- You are not required to hand anything in at your presentation, but you may if you choose.

Problems:

1. [Euler tours]

   Warmup: During the eighteenth century the city of Konigsberg in East Prussia was divided into four sections by the Pregel river. Seven bridges connected these regions, as shown below. It was said that residents spent their Sunday walks trying to find a way to walk about the city so as to cross each bridge exactly once and then return to their starting point.

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
         |    |        |
      xxxxxxxxxx----xxxxxxxx
         |    |        |
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

   (xxx : ground. lines: bridges. blanks: river)

   Show how the residents of the city could accomplish such a walk or prove no such walk exists.

   An Euler tour in a graph is a cycle that traverses each edge exactly once (it may visit some vertices multiple times — i.e., it doesn't have to be a *simple* cycle). In this problem we will assume the graph is undirected.

   (a) Suppose the graph has some node of odd degree. Then there cannot be an Euler tour. Why?

   (b) On the other hand, if all nodes have even degree (and the graph is connected) then there always does exist an Euler tour. Prove this by giving a polynomial-time algorithm that finds an Euler tour in any such graph. Your algorithm should work for multigraphs too (multiple edges allowed between any two vertices).

   Hint: Suppose you start at some node x and just arbitrarily take a walk around the graph, never going on any edge you've traversed before. Where will you end up? Now, what about parts of the graph you haven't visited?

2. [Vertex-Cover] A vertex cover for an undirected graph $G = (V, E)$ is a subset S of its vertices such that each edge has at least one endpoint in S. In other words, for each edge (u, v) in E, one of u or v must be an element of S. The decision problem (does there exist a vertex-cover of size k?) is known to be NP-complete.

   Consider the following decision problem:

   INSTANCE: graph G over $2n$ vertices. A list that partitions V into n pairs of vertices (in other words, every vertex belongs exactly to one pair).

   QUESTION: Is there a vertex cover that uses exactly one vertex from each pair? (if (v_1, v_2) a pair in the list, both cannot participate in the cover).

   Is the problem NP-complete as well? Prove or disprove.

3. [TSP approximation] Given a weighted undirected graph G, a *traveling salesman tour* for G is the shortest tour that starts at some node, visits all the vertices of G, and then returns to the start. We will allow the tour to visit vertices multiple times (so, our goal is the shortest cycle, not the shortest simple cycle). This version of the TSP that allows vertices to be visited multiple times is sometimes called the *metric* TSP problem, because we can think of there being an implicit complete graph H defined over the nodes of G, where the length of edge (u, v) in H is the length of the shortest path between u and v in G. (By construction, edge lengths in H satisfy the triangle inequality, so H is a metric. We're assuming that all edge weights in G are positive.)

   (a) Briefly: show why we can get a factor of 2 approximation to the TSP by finding a minimum spanning tree T for H and then performing a depth-first traversal of T. (If you get stuck, the CLRS book does this in a lot more sentences in section 35.2.1.)

   (b) The minimum spanning tree T must have an even number of nodes of odd degree (only considering the edges in T). In fact, *any* (undirected) graph must have an even number of nodes of odd degree. Why?

   (c) Let M be a minimum-cost perfect matching (in H) between the nodes of odd degree in T. I.e., if there are $2k$ nodes of odd degree in T, then M will consist of k edges in H, no two of which share an endpoint. Prove that the total length of edges in M is at most one-half the length of the optimal TSP tour.[1]

   (d) Combine the above facts with your algorithm from 2(b) to get a 1.5 approximation to the TSP. Hint: think about the (multi)graph you get from the union of edges in T and M.

   The above algorithm is due to Christofides [1976]. Extra credit and PhD thesis: Find an algorithm that approximates the TSP to a factor of 1.49.

[1] We didn't prove it in class, but there are efficient algorithms for finding minimum cost perfect matchings in *arbitrary* graphs (not just bipartite graphs).