15-451 Algorithms, Fall 2012

Homework # 6 **Due: Sun-Wed, Nov 18-21, 2012**

Ground rules:

- This is an oral presentation assignment. You should work in groups of three. At some point before **Saturday, Nov 17 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. [NP-completeness and approximation algorithms]

   (a) Let $\mathcal{A}$ be the set of pairs (G, k) such that G is a graph with a vertex cover of size k or less. Let $\mathcal{C}$ be the set of pairs (G, k) such that G has a vertex cover of size $k/2$ or less. Notice that $\mathcal{A} \supseteq \mathcal{C}$ because if $(G, k) \in \mathcal{C}$ then clearly $(G, k) \in \mathcal{A}$ also.

   Determining whether a given input (G, k) belongs to $\mathcal{A}$ is NP-Complete (this is the Vertex-Cover problem), and also determining whether a given input (G, k) belongs to $\mathcal{C}$ is NP-complete (since this is really the same problem).

   Describe a set $\mathcal{B}$ such that $\mathcal{A} \supseteq \mathcal{B} \supseteq \mathcal{C}$ but membership in $\mathcal{B}$ can be decided in polynomial time (and explain why membership in $\mathcal{B}$ can be decided in polynomial time).

   (b) As a related example, let $\mathcal{A}$ be the set of CNF formulas that have at least one satisfying assignment. Let $\mathcal{B}$ be the set of CNF formulas that have an assignment satisfying an *odd* number of literals per clause. Let $\mathcal{C}$ be the set of CNF formulas that have an assignment satisfying *exactly one* literal per clause. Clearly $\mathcal{A} \supseteq \mathcal{B} \supseteq C$. We know that determining whether a formula belongs to $\mathcal{A}$ is NP-complete. It also turns out that determining whether a formula belongs to $\mathcal{C}$ is NP-complete. However, determining whether a formula belongs to $\mathcal{B}$ can be done in polynomial time! How? Hint: think linear algebra (which all works great mod 2 as well).

2. [Euler tours] 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?

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.

   (d) Combine the above facts with your algorithm from 2(b) to get a 1.5 approximation to the TSP.

   The above algorithm is due to Christofides [CMU Tech Report, 1976]. Extra credit and PhD thesis: Find an algorithm that approximates the TSP to a factor of 1.49.[1]

[1]For the case of *points in the plane* it is known how to get a $1+\epsilon$ approximation for any constant $\epsilon > 0$. For the case that G is *unweighted*, two recent results give improvements over 1.5: a $1.5-\epsilon$ for (very small) constant $\epsilon > 0$ by Gharan, Saberi, and Singh `http://www.stanford.edu/~saberi/tsp.pdf`, then improved to a 1.461 approximation by Momke and Svensson `http://arxiv.org/abs/1104.3090`. Both papers appear in the 2011 Symposium on Foundations of Computer Science.