15-451 Algorithms, Fall 2003

Homework # 6 **due: Mon–Tues, Nov 18–19, 2003**

Some Reminders:

- This is an oral presentation assignment. You should work in groups of three. At some point before Sunday Nov 16 at midnight you should sign up for a 1 hour time slot on the signup sheet on the course web page.
- You are not required to hand anything in at your presentation, but you may if you choose.

Problems:

1. [Set Cover] The *set-cover* problem is the following: Given n points labeled $1, 2, \ldots, n$, and m subsets of these points $s_1, s_2, \ldots, s_m$, and an integer k, is it possible to cover all the points using only k of the sets s_i? (I.e., every point should be in at least one of these k sets). For instance, if $n = 6$ and the sets are $s_1 = \{1, 2, 3\}$, $s_2 = \{1, 4\}$, $s_3 = \{2, 5\}$, and $s_4 = \{3, 6\}$, then it is possible to cover all the points with three sets (namely, s_2, s_3, and s_4) but not with two of them.

   (a) Prove that the set-cover problem is NP-complete by reducing from the vertex-cover problem (vertex cover is defined in Chapter 34). Also, say why set-cover is in NP.

   (b) Show how to reduce the search version of the set-cover problem to the decision version. That is, show how you can use an oracle for the set-cover decision problem defined above to actually *find* an optimal set cover (a cover that uses the fewest sets).

   (c) The *fractional set cover* problem is like the set-cover problem, except instead of choosing or not choosing each set, you instead assign each set a fraction between 0 and 1. The requirement is that for each point i, if you add up the fractions assigned to sets that cover that point, you must get a total of ≥ 1. The goal is to minimize the sum of all the fractions.

   For example, if there are 3 points and the sets are $s_1 = \{1, 2\}$, $s_2 = \{2, 3\}$, and $s_3 = \{3, 1\}$, then the best fractional set cover is to assign each set the fraction $1/2$; this covers all the points and the total sum is $1/2 + 1/2 + 1/2 = 3/2$. In contrast, the best standard set cover requires 2 sets. (Note that a standard set cover is also a legal fractional cover, but not necessarily vice-versa.)

   Show how to solve the fractional set cover problem using *linear programming*. Be sure to specify what the variables are, what the constraints are, and what you are trying to minimize or maximize.

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? This problem is given as problem 22-3 in the book.

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 (just a few sentences): prove that 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 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 at all 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 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).