University of Waterloo
CS 466 — Advanced Algorithm
Spring 2013
Problem Set 8
Siwei Yang - 20258568

1. [10 marks:] Given a set P of n points in the plane, a minimum Steiner tree is a tree that connects the points of P and has minimum total Euclidean (i.e. L_2) length. For example, for 3 points p_1, p_2, p_3 forming an acute triangle, the minimum Steiner tree has leaves p_1, p_2, p_3 plus a node of degree 3 in the middle of the triangle at the Fermat point where the 3 edges form angles of 120°(Look at Wikipedia for examples).

   Prove that there is a polynomial time 2-approximation algorithm for the minimum Steiner tree problem in the plane. In particular, show that the minimum spanning tree of the points(using Euclidean distances as edge weights in the complete graph) is a 2-approximation.

2. [10 marks] Given a directed graph, G, represented by its adjacency matrix. (Lets assume there are no loops i.e. no edges (i,i).)

   - [6 marks]Give an efficient algorithm to determine what pairs of nodes have directed paths of length exactly $n-1$. Give the runtime of your method and justify this runtime.

     Let M be the adjacency matrix.

     – Calculate series M^{2^i} up to $i = \lceil \ln(n-1) \rceil$. By using Dynamic Programming, we only need to spend time calculate $M^{2^{\lceil \ln(n-1) \rceil}}$ which takes roughly $O(\lceil \ln(n-1) \rceil)$ time assuming size of M is constant.

     – Then find $I \subset \{1, 2, \ldots, \lceil \ln(n-1) \rceil\}$ such that $\sum_{i \in I} 2^i = n-1$.

     – Calculate $M^* = \prod_{i \in I} M^{2^i}$ which takes roughly $O(\lceil \ln(n-1) \rceil)$ time as well.

     Inspect the entries of M^*, whenever $M^*_{i,j} > 0$, there is a path of length $n-1$ from node i to node j.

- [4 marks]We know the Hamiltonian cycle problem is NP-Complete. The Hamiltonian path problem, of having a path go though each node exactly once, is also NP-hard. Explain this apparent anomaly, given that you have already given an efficient algorithm to find paths of length n-1.

  The path found using the algorithm given doesn't take into account a node can only be visit once. If the checking were to be incorporated into the algorithm, then each entries of the matrix have to keep track of all the paths associated with it. And, during the multiplication step, the paths from two entries have to be cross checked to filter out the bad paths. These extra overhead will drive up the runtime to reflect it's NP-hardness.