Nic Carchio
Adam Romano
Professor Ames
Homework 4

1. TSP-OPT can be solved in polynomial time by making a polynomial number of calls to TSP. TSP-OPT asks for the shortest tour for a given graph G and TSP indicates whether a tour exists in a given graph G within a budget B. Therefore, with the use of TSP, it is possible to solve TSP-OPT in polynomial time. For any particular graph, a tour is at least size 0 and at most the sum of the edge weights in the graph. Therefore, using 0 as a lower bound and the sum of the edge weights of G as an upper bound, as well as binary search, it is possible to determine the shortest tour in a graph G, solving TSP-OPT. Repeated calls to TSP using a binary search approach with bounds of (0, sum of edge weights) will ultimately yield the minimum budget in which a tour exists in graph G, which can be used to then directly find the tour. The number of calls to TSP will be log(T) since the budget is repeatedly being cut in half until the minimum is found, and log(T) is polynomial in the input size. Therefore, TSP-OPT can be solved in polynomial time.

2. The procedure that runs in polynomial time indicates whether a given graph G has a Hamiltonian path. Therefore, by looking at each individual edge in G, it is possible to get the actual path. For every edge, there are two possible options. The edge is either a part of the Hamiltonian path, or the edge is not a part of the Hamiltonian path. Therefore, the actual path can be returned by doing the following for every edge:
    a. Removing the edge from the graph
    b. Running the polynomial-time procedure on the graph
    c. If the procedure returns true, move on to the next edge. The procedure returning true means that a Hamiltonian cycle *still* exists, so the edge that was removed was not necessarily a part of the cycle.
    d. If the procedure returns false, that means that the edge that was removed was a part of the Hamiltonian cycle, and should be added to the output that will be returned.

    Since every edge is examined, the Hamiltonian path can be returned by running the procedure |E| times. Therefore, the total running time is some polynomial running time * |E|, which is still polynomial.

3. If a problem Π is in NP, that means that a solution to the problem can be verified as correct in polynomial time and that the solution space to the problem is of size p(n). In order for an algorithm to solve the problem Π, it will have to go through the solution space to verify the possible solutions. Since there are at most p(n) solutions, then the number of possible inputs is $2^{p(n)}$. Therefore, for an algorithm to solve Π, it will have to go through all $2^{p(n)}$ possible solutions, meaning that the algorithm has a runtime of $O(2^{p(n)})$.

4a. This problem can be solved in polynomial time. Remove all the nodes in the set L from the graph. Find a minimum spanning tree of the graph. This MST will **NOT** contain any nodes

Nic Carchio
Adam Romano
Professor Ames

from the set L. Now, after finding a MST, for every vertex in the set L, add the vertex to a neighbor present in the MST. By the end, you will have a minimum spanning tree of the graph that contains the set L of nodes.

4b. Finding a MST that contains precisely the set L is NP-complete. It is not possible to find a MST of a graph that is exactly the set L, but it is possible to verify that a MST given is exactly the set L. If given an MST, it is possible to verify in polynomial time that the MST is exactly the set of nodes in L. This problem is a generalization of the Hamiltonian path problem, which can be reduced to this problem.

4c. Finding a MST whose leaves are included in the set L is NP-complete. Once again, this problem is a generalization of the Hamiltonian path problem.

4d. This is a generalization of the Hamiltonian path problem, which is both NP and NP-hard. Therefore, finding a spanning tree with K or fewer leaves is NP-complete.

4e. This is a generalization of the Hamiltonian path problem, which is both NP and NP-hard. Therefore, finding a spanning tree with K or fewer leaves is NP-complete.

4f. This is a generalization of the Hamiltonian path problem, which is both NP and NP-hard. Therefore, finding a spanning tree with K or fewer leaves is NP-complete.