CS 331: Algorithms and Complexity (Fall 2016)
Unique numbers: 51420, 51425, 51430, 51435

Assignment 7

Due on Wednesday, November 16, by 11:59pm

Matt Gmitro 51430 MTG759

Problem 1

(6 points) Decide whether the answer is "yes", "no", or "unknown since it would decide P versus NP". Give a brief explanation.

1. Define the decision version of the interval scheduling problem as deciding, for a collection of intervals, whether there is a subset of non-overlapping intervals of size at least k.

   Is Interval Scheduling reducible in polynomial time to Independent Set?
   Answer: Yes because the interval scheduling problem is solvable using the greedy algorithm in O(nlogn). Thus, we can solve it using a polynomial number of computational steps plus zero calls to a black box which would solve Independent Set (and zero counts as a polynomial or less number of calls). So Interval Scheduling $<=_P$ Independent Set.

2. Is Vertex Cover reducible in polynomial time to Interval Scheduling?
   Answer: This is unknown because it would resolve the question of whether P=NP. Vertex cover is NP-complete right now. If vertex cover is reducible to interval scheduling then vertex cover would be in P and P=NP.

Problem 2

(15 points) Suppose that $G = (V, E)$ is a graph with integer edge weights. Given a target number W and G, the W-Weight-Cycle problem is to find a simple cycle in G such that the sum of the edge weights on this cycle is W. Show that the W-Weight-Cycle problem is at least as hard as the subset sum problem.

To show this, you have to do the following:

1. **Construct a reduction algorithm.** That is, an algorithm to solve an arbitrary instance of subset sum problem using a polynomial number of computational steps, plus a polynomial number of calls to a black box that solves the W-Weight-Cycle problem. (**Hint:** Consider the concept of a bipartite graph)

Answer: For subset sum we have a set S = $[s_1, s_2, ..., s_n]$ and a weight W. We can generate G with 2n vertices such that each s_i has vertices v_i and u_i connected by an edge with the weight of s_i. Then we add zero weight edges to each u_i from all v_j and each v_i from all u_j, where i does not equal j. In order for a W-weight-cycle to exist in G, the weights from each v_i to u_i in a cycle must sum to W and all of the other edges will be zero weight edges. Summing a traversal of this graph is equivalent to finding a subset in subset sum. Taking a subset S_k which sums to W, we can find a cycle which sums to W by picking edges (v_i, u_i) corresponding to each s_i in S_k and connecting those zero weight edges to form a cycle.

2. **Prove the correctness of your algorithm.** That is, you have to show that these two problems are closely related, i.e., you have to prove an if and only if statement that shows the equivalence of these two problems.
Answer: Proof: First assume we have a subset sum of weight W on some set S, and we transform the set into a graph G using the reduction algorithm. We need to show the graph contains a W-Weight-Cycle. We know this is the case because taking each edges (v_i, u_i) for each s_i in the subset and connecting the zero weight edges between each set of vertices which correlate to an element in the subset forms a cycle of the weights from each s_i, thus forming a W-weight-cycle.

   On the other hand, assume we have a graph built like the one from our algorithm which contains a cycle of weight W. We need to show the set of edge weights create a subset sum equal to W. For each non-zero edge, we can generate an element in a set S with the edge weight. Then taking the subset formed by the elements corresponding with the non-zero weights in the cycle, we get a subset of weight W. Thus, the algorithm is correct.

3. **Show that your reduction is achieved in polynomial time.**
Answer: Let n be the number of elements in a subset. The reduction is done by creating edges between 2n vertices on a graph. We need 2n vertices, and then (n-1) zero weight edges for each vertex plus n edges between the vertices that relate to the same element in the subset. Thus we can do the reduction in, 2n(n-1) + n steps. Which is $O(n^2)$.

Problem 3

(14 points) Suppose the algorithm A takes an undirected graph G and a number k and reports the following in polynomial time: "Not connected" if G is not connected, "Yes" if G is connected and has an independent set of size k, and "No" otherwise.

Construct an algorithm using A that solves the independent set problem in polynomial time.

Answer: We can use A to break down the graph into smaller problems. First, we run A on G with k as the number of vertices in G because this is the maximum possible size

of an independent set from G. If A returns "not connected", then we break the graph into two subgraphs and run A on each graph with k = size of the subgraphs. Repeat this until A returns no. On each case of A, decrement k by one until either A returns yes or k=1. Finally, we can integrate the solution sets for each case of A, which solves the independent set problem.