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

Assignment 8

Due on Wednesday, November 23, by 11.59pm

Matt Gmitro 51430 MTG759

Problem 1

(11 points) Given a set $A = \{a_1, \ldots, a_n\}$, a collection of subsets of A, namely $\mathcal{B}$, where $\mathcal{B} = \{B_1, \ldots, B_m\}$, and a number k. The **Hitting Set Problem** asks whether there is a set $H \subset A$ such that $H \cap B_i$ is not empty for each $i \in [m]$ and $|H| \leq k$. For example, given $A = \{1, 2, 3\}$, $\mathcal{B} = \{\{1, 2\}, \{2\}, \{2, 3\}\}$, and k=1, then H=$\{2\}$ since $H \subset A$ and $H \cap B_i = \{2\}$.

Prove that:

$$\text{Hitting set} \leq_p \text{Set Cover.}$$

Note that Set Cover is NOT Vertex Cover. The Set Cover problem is described in the text book page 456. It is also discussed during the discussion section on Nov 16 & 18.
You need to do the following three steps:

1. **Construct a reduction algorithm.** That is, an algorithm to solve an arbitrary instance of hitting set problem using a polynomial number of computational steps, plus a polynomial number of calls to a black box that solves the set cover problem problem.
   Answer: For the input to a set cover problem we need a set S and a collection of subsets Q. For a hitting set problem the input includes a set A, collection of subsets B, and a number k. To reduce a hitting set problem into a set cover problem we can say that the collection Q = A and let S = [s_i such that s_i in each X in B_i]. We consider each X in B_i because A can be thought of as a set of sets made from elements in S. Then there is a hitting set size k if there is a set cover size k.

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: First, if you have a hitting set of A and B, transform this into S and Q using the algorithm. We need to show there is a set cover of S from Q. If an a_i hits a B_j then we know a_i is in Q and in S. Thus, because each B_j is hit, we know the set is covered.

On the other hand, given a set cover S and Q, we can transform this into the hitting set putting A = Q and B_j = [X in Q such that s_j in X]. Then if s_j is in Q_i, a_i will hit B_j.

3. **Show that your reduction is achieved in polynomial time.**
Answer: The reduction only requires transforming each element in the set B. So the size of all elements within all sets in B is the maximum number of steps that will need to be taken. And we know the size of B is constant.

Problem 2

(12 points) The Clique problem can be defined as follows: Given a graph $G = (V, E)$, is there a clique (i.e., a subset S $\subseteq$ V such that for every vertex v $\in$ S, there is an edge to every other vertex that exists in S) of size at least k?. Clique-3 is a restricted Clique to graphs in which every vertex has a degree of at most 3.

1. Prove that Clique is NP-Complete, given that the Independent Set is NP-Complete. You need to do the following steps:

   a. **Argue that Clique is in NP.**
   Answer: Clique is a decision problem because there is either a clique or there is not. If the answer is yes, then it is verifiable by taking each vertex in S and searching each possible path for the other vertices in S. This will never be more than the size of the set of edges E.

   b. **Choose an NP-Complete problem (in this case the problem is Independent Set).**

   c. **Construct a reduction algorithm.** That is, an algorithm to solve an arbitrary instance of Independent Set problem using a polynomial number of computational steps, plus a polynomial number of calls to a black box that solves the Clique problem.
   Answer: Assume we are given a graph G = (V,E) to solve independent set size k on. Then we construct a new graph H = (V', E') where V' = V. For each edge (u,v) not in E, add (u,v) in E'. Then call Clique(H, k).

   d. **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: First, assume we have an Indepedent Set of size k on a graph G and we use the reduction algorithm to turn G into H. We need to show H includes a clique of size k. Since we know by the definition of independent set that in G we had a group of vertices size k that were not connected to each other by edges, and our algorithm connects all unconnected vertices. We know all of these vertices in the IS will now be connected to each other vertex in H. Thus, H includes a clique of size k.

Suppose instead we have a Clique of size k on a graph H, where H is transformed from G. We nee to show from the clique in H, we can find an IS in G. Since each edge in H is not in G, we know none of the vertices in the clique are connected by edges in G. Thus, we know that there is an independent set in G made from all the vertices of the clique in H.

e. **Show that your reduction is achieved in polynomial time.**
Answer: My reduction is bounded by the number of vertices in G. For each vertex, check to see if there is an edge with each other vertex. This can be done in size of V squared time.

2. Argue that Clique-3 is in NP.
Answer: Clique-3 is solved with a set S of k vertices and can be verified by checking if each vertex has edges to the k-1 other vertices. Thus, it is bounded by the number of edges in the graph + k.

3. Is Clique-3 NP-complete? If yes, prove it. If not, give a polynomial time algorithm to solve Clique-3.
Answer: No. If $k > 4$, we return NO since each vertex can be connected to 3 others at most. Then we can check all k-element subsets of V to see if each is a clique in polynomial time. There will be at most 6 edges per subset and there should be n choose k subsets, which is $O(n^4)$, where n is the size of V.

Problem 3

(12 points) Suppose you're acting as a consultant for the Port Authority of a small Pacific Rim nation. They're currently doing a multi-billion dollar business per year, and their revenue is constrained almost entirely by the rate at which they can unload ships that arrive in the port.

Here's a basic sort of problem they face. A ship arrives, with n containers of weight $w_1, w_2, ..., w_n$. Standing on the dock is a set of trucks, each of which can hold K units of weight. (You can assume that K and each w_i is an integer.) You can stack multiple containers in each truck, subject to the weight restriction of K; the goal is to minimize the number of trucks that are needed in order to carry all the containers. This problem is NP-complete (you don't have to prove this).

A greedy algorithm you might use for this is the following. Start with an empty truck, and begin piling containers 1, 2, 3, ... into it until you get to a container that would overflow the weight limit. Now declare this truck "loaded" and send it off; then continue the process with a fresh truck.

a. Give an example of a set of weights, and a value of K, where this algorithm does not use the minimum possible number of trucks.
Answer: Suppose the containers w_1, w_2, w_3 brought in have weights [1,3,1] respectively and K = 3. Then the greedy algorithm will use 3 trucks when it only needed 2.

Problem 3 continued on next page. . .

b. Show that the number of trucks used by this algorithm is within a factor of 2 of the minimum possible number, for any set of weights and any value of K.
Answer: If we take the sum of all weights to be W, then the minimum number of trucks would be W/K. Assume we use y trucks such that y = 2x + 1. Divide the trucks into groups of 2 so that we get x+1 groups. The last group will not necessarily have total weight greater than K, but all the prior ones will. This means $W > Kx$, thus $W/K > x$. Then the optimal must use at least x+1 trucks, and y is in a factor of 2 because y = 2x + 1.