CS1510 Reduction Problems 5, 8, & 12

Rebecca Negley, Sean Myers

October 14, 2011

5. Show that if one of the following three problems has a polynomial time algorithm then they all do.

   - The Independent Set Problem: The input is a graph G. The problem is to find the largest independent set in G. In an independent set all vertices are mutually nonadjacent.
   - The Clique Problem: The input is a graph G. The problem is to find the largest clique in G. In a clique all vertices are mutually adjacent.
   - The Vertex Cover Problem: The input is a graph G. The problem is to find the smallest vertex cover in G. A set S is a vertex cover if each edge in G is incident to a vertex in S.

   Solution: First, observe that, for some graph G, the complement of any independent set of G is a vertex cover of G and vice versa. Suppose not. Suppose there exists some independent set S whose complement $\bar{S}$ is not a vertex cover. There must exist some edge e in G that is not incident to any vertex in $\bar{S}$. Then, e must be incident to vertices in S on both ends, so all vertices in S are not mutually nonadjacent, a contradiction. Then, we need that there exists some vertex cover S' of G whose complement $\bar{S}'$ is not an independent set in G. This means that there are two vertices in $\bar{S}'$ that are not mutually nonadjacent, so some edge e goes between them. Neither of these edges can be in S', so e is incident to no vertices in S', a contradiction. Hence, we have that some set is a vertex cover if and only if its complement is an independent set.

   From the above statement it is trivial to prove that a polynomial time algorithm for the Independent Set Problem exists if and only if a polynomial time algorithm exists for the Vertex Cover Problem. If a polynomial time algorithm exists for the Independent Set Problem, we can reduce the Vertex Cover Problem to the Independent Set Problem by taking its input graph G and inputting it into the Independent Set Problem. We take the complement of the independent set we get back (which takes linear time) and return that as our vertex cover. Obviously, the complement of the largest independent set will be the smallest vertex cover. If there exists some smaller vertex cover, its complement will be an independent set larger than the one returned by the Independent Set Problem, a contradiction. Similarly, we can reduce the Indepent Set Problem to the Vertex Cover Problem by taking its input graph G and inputting it into the Vertex Cover Problem. We return the complement of the smallest vertex cover (linear time), which must be the largest independent set by the same reasoning as above. Since we can reduce each of these problems to the other in linear (polynomial) time, a polynomial time algorithm exists for one if and only if a polynomial time algorithm exists for the other.

   Now we will show that there exists a polynomial time algorithm for the Independent Set Problem if and only if there exists a polynomial time algorithm for the Clique Problem. First, we will reduce the Clique Problem to the Independent Set Problem. For any input graph G, input its complement $\bar{G}$ (where an edge exists between two vertices if and only if no edge exists between the vertices in G) into the Independent Set Problem. We return the set of vertices returned by the Independent Set Problem as our largest clique. It is clear that the set returned will be a clique because, since all vertices are mutually nonadjacent in $\bar{G}$, they must be mutually adjacent in G. Suppose that it is not the largest clique. Then, there exists some larger set of mutually adjacent vertices. However, these vertices will be mutually nonadjacent in $\bar{G}$, so a larger independent set will exist than the one returned by the Independent Set Problem, a contradiction. Hence, our output will be valid. This can easily go the other way. We reduce the Independent Set Problem to the Clique Problem by inputting $\bar{G}$, the complement of the input G, into the Clique Problem. We return the set of points in the largest clique in $\bar{G}$ as our largest independent set in G. By the same reasoning as above, this is valid. Since we can easily create $\bar{G}$ in $O(n^2)$ time (because there are $O(n^2)$ pairs of edges), we have that a polynomial time algorithm for the Independent Set Problem exists if and only if a polynomial time algorithm exists for the Clique Problem.

8. Show that the subset sum problem is self-reducible. The decision problem is to take a collection of positive integers $x_1, \ldots, x_n$ and an integer L and decide if there is a subset of the x_i's that sum to L. The optimization problem asks you to return the actual subset if it exists. So you must show that if the decision problem has a polynomial time algorithm then the optimization problem also has a polynomial time algorithm.

   Solution: The first thing we do is set up a decision tree of whether to add a node at a particular level or not. At every single point of this decision tree, the first thing we do is don't take the x_i at level i. We then set up the input into the decision problem, so that the input is whatever our current subset is, S, so that $S, x_{i+1}, ..., x_n$ and L. If the output to that input is 1, then we know that there is still a possible solution without adding x_i to the subset. If it returns 0, we add x_i to the subset, since it is impossible to get a feasibly solution without adding x_i. Setting up the input will take a maximum of n time (actually less that if we keep a collection of S, and all the current nodes we need to check). If the decision algorithm is polynomial some n^k, and we are running it n times, then the optimization algorithm will run in $n^{k+1} + n^2$ time, where n^2 is setting up the input n times. Therein, if the decision is polynomial time, so is the optimization.

12. Consider the following 2Clique problem:
    INPUT: An undirected graph G and an integer k.
    OUTPUT: 1 if G has two vertex disjoint cliques of size k, and 0 otherwise.
    Show that this problem is NP-hard. Use the fact that the clique problem is NP-complete. The input to the clique problem is an undirected graph H and an integer j. The output should be 1 if H contains a clique of size j and 0 otherwise. Note that a clique is a mutually adjacent collectioin of vertices. Two cliques are disjoint if they do not share any vertices in common.

    Solution: To properly show that this problem is NP-Hard, we must have it so that $Clique < 2Clique$, in other words we have to show that we can create an algorithm for Clique, using 2Clique. If we can show that setting up the input so that it if and only if 2Clique is correct, then Clique is correct in polynomial time, then we will have shown that if there exists a polynomial time solution for 2Clique, then all NP complete problems will have a polynomial time solution (the definition of NP-Hard) The first step in doing that is converting the input of Clique to the input of 2Clique. This is very trivial to do. All we have to do is use the undirected graph for clique, and add a Clique of size j to the graph. In order to do that, create j vertices, and add an edge to every vertex that you just made. If there is one Clique already there, then 2Clique will find that one and the one you just made. If there was already 2Cliques of size j there, then it would ouput 1 anyways, so creating an extra one will do no harm. If there are no cliques there to begin with of size j, then 2Clique will output a zero, since we only introduced 1 clique, hence it is a one to one relationship between the output of 2Clique and the output of Clique.

    If we added n vertices ($n = j$), then to be adjacent, we must add roughly n^2 edges, which is a polynomial conversion from one input to the other. So if 2Clique can find an output in polynomial time, so can Clique, and since Clique is an NP-Complete problem, we have shown that 2Clique is an NP-Hard problem.