Introduction to Algorithms	**Problem Set 6**
CS 4820 Spring 2016	**Due 11:59pm Thursday, March 24**

Your homework submissions need to be typeset (hand-drawn figures are OK). See the course web page for suggestions on typing formulas. Solution to each question needs to be uploaded to CMS as a separate pdf file.

Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time. The running time must be bounded by a polynomial function of the input size.

If A and B are two decision problems, proving $A \leq B$ often just consists of a polynomial-time algorithm that takes an instance of A and outputs an instance of B. To show that such a reduction is correct, you must prove two things. First, that it transforms YES instances of A to YES instances of B. Second, that it transforms NO instances of A to NO instances of B.

Problems 1 and 3 are asking you to prove a problem NP-complete. The list of problems we proved NP-complete in class so far are SAT, INDEPENDENT SET, VERTEX-COVER. You can use in your proof that these problems, as well as the CLIQUE problem defined below, are NP-complete.

(1) *(10 points)* Consider a strong version of SAT where we would need each clause to have at least 2 true variables. We say that a SAT formula is *very satisfiable* if there is a truth assignment so that there are at least two different true variables in each clause. The VERY-SAT problem is to decide if a given SAT formula Φ has a very satisfying truth assignment.

Show that the VERY-SAT problem is NP-complete.

Solution.

- Clearly VERY-SAT is in NP: given a truth assignment that has two true variables in each clause, this fact is easy to check.
- We prove that the problem is NP-complete by a reduction from SAT, which is known to be NP-complete.

Consider a SAT formula Φ. We need to show that we can decide if Φ is satisfiable by running an algorithm for VERY-SAT. To do this we add a new variable to Φ, let's call it y and now add y to every clause as an additional variable (so clause $x_1 \vee x_2 \vee \bar{x}_3$ becomes $y \vee x_1 \vee x_2 \vee \bar{x}_3$. Let's call the resulting formula Φ'. Clearly this transformation was done in polynomial time.

We claim that Φ is satisfiable if and only Φ' is very satisfiable, which establishes that SAT $\leq$ VERY-SAT.

- If Φ is satisfiable, we can use this satisfying assignment and add $y = \mathit{true}$ to make Φ' doubly satisfied.
- If Φ' doubly satisfied, then Φ must be satisfied by the part of the assignment on the original variables, as each clause only had one additional variable y and each clause in Φ' has two true variables.

Common mistakes. A few students did the reduction backwards, showing that VERY-SAT $\leq$ SAT, that is: a VERY-SAT problem can be solved using a SAT subroutine. This is also possible to show, but it is already implied by the fact that SAT is NP-complete, and hence doesn't show that VERY-SAT is NP-complete.

(2) *(10 points)* In class, we stated without proof the important fact that SAT is NP-hard, which means that every problem in NP reduces to SAT in polynomial time. We have seen more of these reductions, that SAT is at least as hard as matching and also independent set.

To gain more intuition about how these reductions work, this exercise asks you to give a polynomial-time reduction from a particular NP problem to SAT. **For this exercise, your solution *may not* use the fact that** SAT (or any other problem) **is NP-hard.**

The GRAPH ISOMORPHISM problem is defined by two undirected graphs $G = (V, E)$ and $G' = (V', E')$ with $|V| = |V'|$, and asks the question if the two graphs are isomorphic: that is, is there a one-to-one mapping of $\pi : V \to V'$, so that for all pairs of nodes v, w in V, the edge $(v, w) \in E$ if and only if $(\pi(v), \pi(w)) \in E'$. For example, the two graph on the figure below are isomorphic with the numbering showing how to identify the vertices.

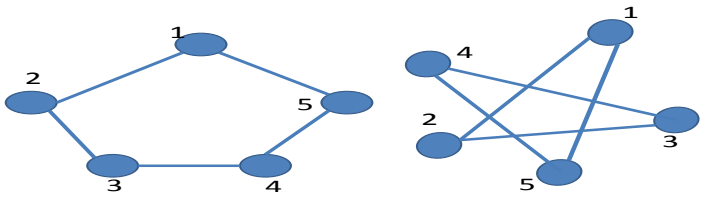

Figure 1: Two isomorphic graphs.

Give a polynomial-time reduction from GRAPH ISOMORPHISM to SAT, i.e., show that GRAPH ISOMORPHISM $\leq$ SAT.

Comment. You may find it interesting to hear about a recent breakthrough of a sub-exponential graph isomorphism algorithm `http://arxiv.org/abs/1512.03547`. While this is not a polynomial time algorithm, it suggests that graph isomorphism may really be easier than NP-complete.

Solution. We'll use $v, w \in V$ to denote nodes in V and $i, j \in V'$ to denote nodes in V'. We'll use variables x_{vi} that will be *true* if node v is mapped to node i in the isomorphism, *false* otherwise. This is n^2 variables. For clauses, we need the following.

- $\vee_{i \in V'} x_{vi}$ for all nodes $v \in V$ and $\vee_{v \in V} x_{vi}$ for all nodes $i \in V'$
- for all nodes v and all pairs of nodes $i, j \in V'$ we add $\bar{x}_{vi} \vee \bar{x}_{vj}$ and for all nodes $i \in V'$ and all pair of nodes $v, w \in V$ we add $\bar{x}_{vi} \vee \bar{x}_{wi}$. The set of constraints so far expressing that the isomorphism should be a one-to-one matching between nodes in V and V'.
- Note that it is enough to add the first kind of clauses on one side of the graph, and the second kind on the other side (rather than adding both kinds on both sides). This is true, as we assumed the two graphs have the same number of nodes, so the only way to have each node of G assigned a node of G' and no node in G' getting two assignment is to have the assignment one-to-one.
- For each edge $(v, w) \in E$, we need to ensure that the matched nodes in V' are also forming an edge, and vice versa: which can be done by a constraint for each pair of nodes $v, w \in V$ and $i, j \in V'$ if $(v, w) \in E$ and $(i, j) \notin E'$ or $(v, w) \notin E$ and $(i, j) \in E'$ then we add $\bar{x}_{vi} \vee \bar{x}_{wj}$ to guarantee that v, w are not mapped to i, j.
- There is another solution for this last set of clauses: adding a clause for each edge in E and node $v \in V'$, as well as for edges in E' and nodes in V. For the edge $(ij) \in E$ and node $k \in V'$ we add

the clause: $(\bar{x}_{ik} \vee x_{j\ell_1} \vee x_{j\ell_2} \vee \cdots)$, where $\ell_1, \ell_2, \ldots$ are the list of k's neighbors in G'. This clause means, either i is not mapped to k, or (if i is mapped to k) the neighbor j of i must be mapped to some ℓ which is a neighbor of k.

To prove correctness of this reduction we need to argue

- the reduction runs in polynomial time. For a graph with n nodes and m edges, we created n^2 variables, $2n$ clauses of the first kind, and at most $2n^2m \leq n^4$ of the second kind.
- if the two graphs are isomorphic, with mapping π, then the setting of the variables $x_{v\pi(v)} = 1$ for all v, and setting all others to 0 satisfies all constraints.
- If the formula is satisfiable, then for all nodes $v \in V$ there is exactly one node $i \in V'$ with x_{vi} true by the first set of constraint. This defines the mapping $\pi(v) = i$. Due to the same set of constraints for nodes in V' this mapping is one-to-one. The final set of constraints guarantees that edges of E are matched to edges of E' and vice versa.

Common Mistakes Some students had trouble expressing the last constraint as a set of clauses: that two nodes $v, w \in V$ are matched to nodes $i, j \in V'$ connected by an edges if and only if (v, w) is an edge in G.

(3) *(10 points)* Consider the (undirected) graph of friendships $G = (V, E)$, where nodes correspond to people, and two nodes are connected by an edge if they are friends (we will assume friendships are symmetric). You are also one node in the graph, let's call this node y. You would like to organize a giant party to subset of friend, i.e., a subset of the neighbors of your node in the graph. Ideally you want all pairs of people invited to be friends. Such a group of nodes with all edges between them is called a *clique* in the graph. It turns out that this would make your party too small. So you decide that it is OK to invite any pair of people, who have at least one friend in common beyond yourself, where this other shared friend $x \neq y$ is not necessarily one of your friends. For this problem, we will call such a set of nodes an *almost clique*, that is, a set $S \subset V$ all of which are neighbors of one node y (representing you in the graph), such that for all pairs of nodes $v, w \in S$, either $(v, w) \in E$ or there is a third node $s \neq y \in V$ (doesn't have to be in S or even be connected to y) such that both $(v, s) \in E$ and $(w, s) \in E$. Given a desired size k, and a graph G with special node y the ALMOST CLIQUE problem is to decide if the graph has an almost-clique among y's neighbors of size k.

Show that the ALMOST CLIQUE problem is NP-complete.

Hint: We have seen that the INDEPENDENT SET problem is NP-complete. A similar problem is the CLIQUE problem: given a graph G and a target size k, does the graph have a clique of size k? Notice that the CLIQUE problem is also NP-complete (as the clique problem on graph G is the same as the independent set problem on the complement graph of G, where the edges in the complement are exactly the pairs that are not edges in G). You can use that the CLIQUE problem is NP-complete without writing down a proof for this.

Solution. First note a few facts:

- the only part of the graph that matters is the set of neighbors of node y. Let's call the graph restricted to just this set of nodes G.
- Now notice that the clique problem on G is the same as the independent set problem on the complement (adding edges if and only if G didn't have an edge). Suggesting that independent set may be the most relevant NP-complete problem.

Now consider the formal proof.

- First note that the ALMOST CLIQUE problem is in NP, as given an almost clique, it is easy to verify that it satisfies the constraints.
- We will use the CLIQUE problem in our reduction, which is NP-complete.

To show that the CLIQUE $\leq$ ALMOST CLIQUE, consider an input graph G for independent set, and do the following construction. Let $G = (V, E)$. We add a new node y and edges (y, v) for all $v \in V$. In addition, we add nodes n_{vw} for all pairs of nodes $v, w \in V$ that form an edge, and replace edge (v, w) with edges (v, n_{vw}) and (w, n_{vw}) for each such pair of nodes (or more informally, we add a new node n_{vw} in the middle of every edge (v, w)). See an example on the figure below. Let G' be the resulting graph.

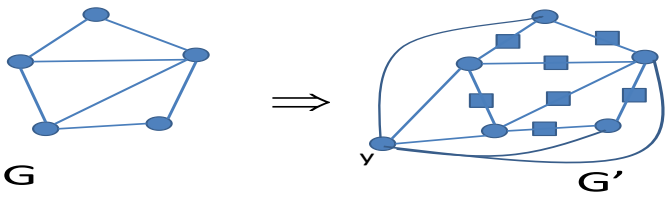

Figure 2: Example of the reduction from CLIQUE to ALMOST-CLIQUE.

We claim that G' has an almost clique of size k among the neighbors of node y, if and only if the original graph G has an clique of size k. This is true, as the neighbors of y are the original nodes in V, and two are connected by an edge in G if and only if they have another shared neighbor in G' (other than y).

Alternate Solution

Rather than adding a new node y, we can try all nodes y of the original graph as y, and do a special construction G'_y with y as the special node. Then we claim that G has a clique of size k containing y if and only if G'_y has al almost clique of size $k - 1$ (-1 as y is also part of the clique).. In this version we subdivide all edges not adjacent to y, so the neighbors of y in G'_y are the same as the neighbors of y in G.

Common mistakes. A few students did the reduction backwards, showing that ALMOST-CLIQUE $\leq$ CLIQUE, that is: a ALMOST-CLIQUE problem can be solved using a CLIQUE subroutine. This is also possible to show (and in fact, easier to show), but it is already implied by the fact that CLIQUE is NP-complete, and hence doesn't show that ALMOST-CLIQUE is NP-complete.

Some people are trying the above construction trying all nodes y without adding the extra nodes in the middle of edges. Rather: let G' be just the neighbors of y in G. The idea is this: consider two neighbors v and w of y. The construction hopes that since we deleted all nodes not adjacent to y, we cannot have $s \neq y$ such that (v, s) and (s, w) in E, so if v and w are both in the almost clique, they must be connected. Unfortunately, this is not true, say if the graph G has 4 nodes y, v, w, s and all edges with the one edge (v, w) missing, than among y's neighbors, $\{v, w, s\}$ is an almost clique, but is not a clique.