Algorithms
Computer Science 140 & Mathematics 168
Instructor: B. Thom
Spring 2004
Homework 12b
Due on 4/27/04 (beginning of class)

1. **[10 Points] Professor Lai and NP!**

   (a) Professor I. Lai from P.I.T. claims that if P_1 and P_2 are any two problems in NP such that $P_1 \leq_p P_2$, then $P_2 \leq_p P_1$. Prove that this would imply that $P = NP$.

   (b) Professor Lai is studying an interesting problem called the Polygon Intersection Testing problem (PIT). He claims to have proved that PIT is NP-complete by showing that PIT is in NP and that PIT $\leq_p$ VC (VC is the Vertex Cover problem). Is this a valid approach for proving that PIT is NP-complete? Explain.

2. **[10 Points] More on ILP!**

   In class we proved that Integer Linear Programming (ILP) is NP-complete. Our reduction was from 3SAT. In this problem you will prove that ILP is NP-complete using another reduction. You do not need to show that ILP is in NP (we already argued that in class). You need only give a reduction, prove its correctness (remember, this is an "if and only if" proof), and explain why the reduction can be done in polynomial time.

   (a) Show that ILP is NP-complete using a reduction from Vertex Cover (VC).

3. **[25 Points] The Kernel!**

   Consider a directed graph $G = (V, E)$. A *kernel* for graph G is a subset K of the vertices such that:

   (a) Every vertex in V is either in the kernel K or has an incoming edge from some vertex in the kernel. (Formally, "$\forall v \in V$, $v \in K$ or $\exists u \in K$ s.t. $(u, v) \in E$.")

   (b) There do not exist two vertices in the kernel K with a directed edge between them. (Formally, "$\forall u, v \in K$, $(u, v) \notin E$ and $(v, u) \notin E$.")

   Observe that the graph in Figure 1 has a kernel. (A set containing just one of the two vertices is a kernel.)

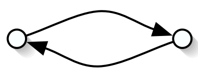

Figure 1: This directed cycle has a kernel.

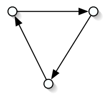

Figure 2: This directed cycle has no kernel.

On the other hand, the graph in Figure 2 has no kernel. (Take a look at the definition of the kernel to make sure you see why.)

The **Kernel Problem** is the following: Given a directed graph $G = (V, E)$, does G have a kernel? Prove that the Kernel Problem is NP-complete. (Hint: Perform a reduction from 3SAT and use the graphs in Figures 1 and 2 as the gadgets.)

4. **[25 Points] Snow Plows Revisited.**

   Burrsburgh, Minnesota can be represented by a directed graph where vertices represent points in the city and directed edges represent one-way roads between pairs of points. Some roads are maintained by the city and are denoted *type 1* roads. Other roads are maintained by the county and are denoted *type 2* roads. Every directed edge has a non-negative integer weight associated with it which indicates the time it takes for a snowplow to traverse that road. The city owns exactly one snow plow which is parked at a shed (a particular point in the city). The objective of the optimization problem is to traverse all the type 1 roads in the minimum total amount of time. A road may be traversed more than once and type 2 roads may be traversed if its useful (but type 2 roads are not required to be traversed). The vertex at which the snowplow ends up is of no consequence.

   The Burrsburgh Decision Problem (BDP) is the following: Given a directed graph in which each edge is labeled either type 1 or type 2 and each edge has a non-negative integer weight, and given a designated start vertex for the snowplow, and given a positive integer k, does there exist a path for the snowplow to follow such that every type 1 edge is traversed at least once and the total cost of the solution is at most k?

   Prove that this problem is NP Complete.

5. **[25 Points Extra Credit] 2SAT (Due on Wednesday at 7pm)!**

   You might recall from CS 81 that the *resolution* algorithm solves 2SAT. In this problem, you'll show that 2SAT can also be solved efficiently using graph algorithms. (Amazing but true: Those ever-versatile graph algorithms raise their ugly heads again!)

   Recall that 2SAT is just like 3SAT, except that every clause in a 2SAT instance contains the disjunction of exactly two variables. (*Note:* We use the notation $\neg x$ to denote the negation of variable x.) Although 3SAT is NP-complete, you will demonstrate in this problem that 2SAT can be solved efficiently. In particular, you'll derive a polynomial-time reduction from 2SAT to a graph problem whose Yes/No answers are faithful to the original problem. Moreover, you'll develop an algorithm for answering this question in the reduced-to-graph domain that runs in polynomial-time.

(a) Consider the following instance of 2SAT:

$$(x_1|x_2)\&(x_1|\neg x_2)\&(\neg x_1|x_2)$$

Show that this instance is satisfiable by giving a satisfying assignment for the variables.

(b) Now consider the following instance of 2SAT:

$$(x_1|x_2)\&(x_1|\neg x_2)\&(\neg x_1|x_2)\&(\neg x_1|\neg x_2)$$

Explain briefly why this instance of 2SAT is not satisfiable.

(c) Given an instance of 2SAT, let's construct a corresponding directed graph as follows: For each variable x_i that appears in the 2SAT instance, construct a *pair* of vertices, one labeled x_i and the other labeled $\neg x_i$. For every clause of the form $(a|b)$ in the 2SAT instance (where a and b are variables which may or may not be negated), place a directed edge from vertex $\neg a$ to vertex b and also a directed edge from vertex $\neg b$ to vertex a. For example, if we had a clause $(x_1|\neg x_3)$ then $a = x_1$ and $b = \neg x_3$. Therefore, we would place a directed edge from vertex $\neg x_1$ to vertex $\neg x_3$ as well as a directed edge from vertex x_3 to vertex x_1. In this example, you should interpret the edge from vertex $\neg x_1$ to vertex $\neg x_3$ to mean "if $\neg x_1$ is true (that is, x_1 is false) then $\neg x_3$ must be true." Similarly, the edge from vertex x_3 to the vertex x_1 is interpreted as "if x_3 is true then x_1 must be true."

Construct this directed graph for the 2SAT instance in part (a). This graph should contain 4 vertices 6 edges. Notice that there is no path from vertex x_1 to vertex $\neg x_1$.

(d) Notice that in the graph you constructed there is a path from vertex $\neg x_1$ to vertex x_1. Clearly explain why *the existence of this path* implies that a satisfying assignment for this instance of 2SAT cannot have $\neg x_1$ be **true** (that is, x_1 cannot be **false**). Be very clear and precise about your reasoning here.

(e) Notice that there does *not* exist a path in this graph from vertex x_1 to vertex $\neg x_1$. Notice also that in your satisfying assignment for this instance, x_1 was set to **true**. Clearly explain why this graph tells us that x_1 should be assigned **true** if the instance has any hope of being satisfied.

(f) What does the graph tell you about the value that should be assigned to variable x_2? Explain.

(g) Now, construct the graph for the 2SAT problem in part (b). What property does this graph have that forces you to conclude that the 2SAT instance is not satisfiable? Be precise.

(h) Next, write down a conjecture that starts as follows: "A 2SAT instance is satisfiable if and only if the corresponding directed graph has the property that blah, blah, blah." Fill in the "blah, blah, blah" with mathematically precise language.

(i) Now, prove your conjecture. This is the main part of this problem (it's also worth most of the points!) and will take a few paragraphs to prove. Note that this is an "if and only if" proof, which means that there are two things to show. (Essentially, what you're proving here is that this 2SAT to graph reduction is faithful.)

In this proof, the fact that two edges were introduced for each clause will be absolutely critical. You'll need to use this fact!

(j) Now, describe an algorithm to determine whether or not a 2SAT instance is satisfiable. What is the running time of your algorithm? (Big fat hint: you want this to be polynomial).

(k) Is 2SAT in NP? Why (or why not)? Is it NP-Complete? Why (or why not)?