Algorithms
Computer Science 140 & Mathematics 168
Instructor: B. Thom
Fall 2004
Homework 14b
Due on December 7, Tuesday, beginning of class

1. **[20 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.)

   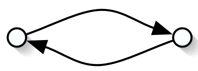

   Figure 1: This directed cycle has a kernel.

   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.)

   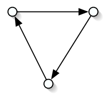

   Figure 2: This directed cycle has no kernel.

   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.)

2. **[25 points] Network Reliability is NPC!**

   The Network Reliability Problem (NRP) is a famous problem arising in the area of network design. First, a few definitions. We will represent a network as an undirected graph with n vertices $v_1, \ldots, v_n$. We say that two paths in the network are *disjoint* if they have *no vertices* in common except for the endpoints. For example, there can be two disjoint paths from vertex v_{42} to v_{100}, but v_{42} and v_{100} can be the only vertices that these paths have in common (since these vertices are the endpoints of the paths). In this definition, we're assuming that each pair of vertices considered is unique, i.e. $\forall\ v_i, v_j, i \neq j$ (in other words, the number of disjoint paths between a vertex and itself is zero). In the interest of reliability, it is desirable to have multiple disjoint paths between pairs of nodes in the network.

The Network Reliability Problem (NRP) is defined as follows: Given is an undirected graph with n vertices $v_1, \ldots, v_n$, an $n \times n$ symmetric matrix R of positive integers, and a positive integer b. The question is whether there exists a subset S containing exactly b edges in G such that for *each pair* of vertices v_i and v_j there exist at least r_{ij} *vertex disjoint* paths from v_i to v_j such that all paths are constructed using *only* edges from set S?

One could imagine NRP being very useful when designing a specific network configuration. For example, if you had n nodes you wanted to connect together, you could begin by assuming you were going to link each node to every other (i.e. build a complete graph, $$$!), and then, given some desired minimal number of disjoint paths between node pairs, ask "How few physical links can be used to still achieve a desired disjointness?"

Some advice:

(a) The fact that this problem is in NP is less obvious than any others we've seen so far. Write this one up carefully (a novel is not required; a short paragraph will probably do the trick). You might wish to use the following fact: answering if there exists foo vertex-disjoint paths in a undirected graph between s and t is answerable in polynomial time using a Network-flow based approach (we discussed how this might be done in class).

(b) You will find it very useful to first prove (so you can then use) the following lemma: For any graph with E edges, the sum of all its vertices' degrees is exactly $2 \cdot |E|$.

In this problem, you're to do the following:

(a) Prove that NRP is NP-complete using a reduction from a problem that we've shown in class to be NP-complete.

(b) As mentioned earlier, you can answer if there exists foo vertex-disjoint paths between a particular s and t in polynomial time. So what fundamental aspect of the NRP problem makes the difference, allowing NPC problem instances to exist? Explain briefly.

3. **2SAT in polynomial time! [45 Points]**

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 property blah, blah, blah." Fill in the "blah, blah, blah" with mathematically precise language.

(i) In the next item, you will prove your conjecture. But first, prove the following Lemma by inducting on path length: for any vertices i, j in a graph that has been constructed as outlined above, if there exists a path from i to j in the graph then there also exists a path from $\neg j$ to $\neg i$.

(j) Now, prove your conjecture. This is the main part of this problem (it's also worth most of the points!) and will take several paragraphs to prove. Note that, while the conjecture is if and only if we are still in the land of proving faithfulness—that for some arbitrary 2SAT, if we were to reduce it to such a graph, the answer to the graph having property "blah, blah, blah" is faithful to answering the 2SAT question.

   Break your proof up as follows:

   i. Demonstrate that 2SAT is "yes" implies that "graph with property blah,blah,blah" is yes.

   ii. Dmenostrate that "graph with property blah,blah,blah" is "yes" implies that 2SAT is "yes." Begin by constructing a valuation for all the variables by using the graph for which "blah blah blah" is true. Be careful and precise in describing this construction, as it will make your faithfulness argument easier to formulate and parse. Next, argue that the valuation is both valid and that the 2SAT formula will evaluate to true. If you are not using the Lemma when proving this, you are probably not proving everything that you should.

(k) 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: It better be polynomial)?

(l) Is 2SAT in NP? Explain briefly. Is it NP-Complete? Explain briefly.