Introduction to Algorithms	**Problem Set 8**
CS 4820, Spring 2014	**Due 11:59pm Thursday, March 27**

Hand in your solutions electronically using CMS. Each solution should be submitted as a separate file. For multi-part problems, all parts of the solution to that problem should be included in a single 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. The same guideline applies to reductions.

(1) *(10 points)*

Consider the following scheduling problem:

An organization meets every weekend, either on Saturday or Sunday but not both, and they need to schedule their meetings to accomodate a set of speakers, each of whom lists the days when they would be available to make a presentation. A meeting is not limited to one presentation, you can have any number of presentations at a meeting as long as you respect the speakers' availability constraints.

The input for the problem consists a list of n weeks, indexed by numbers 1 to n, and for every speaker two lists that indicate which Saturdays and Sundays the speaker is available. The goal is to decide if there exists a way to schedule meetings during the weekends so that every speaker is available for at least one of the meetings.

Prove that this decision problem is NP-complete.

Solution

First, to prove that this scheduling problem is in NP, we describe a polynomial-time verifier. Given an instance of the problem, and a proposed schedule specifying the dates of the meetings, the verifier tests that (a) in each weekend, the proposed schedule includes either Saturday or Sunday but not both, (b) for each speaker, the proposed schedule includes a date when the speaker is available. Both of these tests can easily be performed in linear time; to test property (b) one first constructs an array that stores one bit for each Saturday or Sunday specifying whether a meeting is taking place on that day, then in linear time one can run through each speaker's lists and look up those dates in the array to check if a meeting is taking place that day.

Next, to prove that the scheduling problem is NP-hard, we reduce from 3SAT. Given an instance of 3SAT with n variables and m clauses we construct an instance of the scheduling problem with n weekends and m speakers. For each clause j containing three literals, the corresponding speaker is available on precisely three days belonging to the corresponding weekends. A literal such as x_i corresponds to the speaker being available on the Saturday of weekend i. A negated literal such as $\bar{x}_j$ corresponds to the speaker being available on the Sunday of weekend j. Generating the set of weekends, the set of speakers, and their lists of available dates takes $O(m+n)$ time, so this is a polynomial-time reduction. To prove the correctness of the reduction, we must show two things.

1. If the 3SAT instance is satisfiable, then there is a schedule of meetings that satisfies all of the problem constraints. Given a satisfying truth assignment of the 3SAT formula, schedule a meeting on Saturday of weekend i if x_i is true, and on Sunday of weekend i if x_i is false. Clearly this schedule contains a meeting each weekend on Saturday or Sunday, but not both. Furthermore, by our assumption that the truth assignment satisfies at least one literal in each clause, and by our construction of the speakers' availability constraints, the schedule contains a meeting on at least one of speaker j's three available dates, for every j.

2. If there is a schedule of meetings that satisfies all of the problem constraints, then there is a satisfying truth assignment of the 3SAT formula. Given a schedule of meetings, assign x_i the value true or false according to whether the meeting in weekend i is on Saturday or Sunday, respectively. (The problem constraints ensure that there is a meeting either Saturday or Sunday, but not both, so the truth assignment of x_i is unambiguous.) By our construction of the availability constraints of speaker j, we know that if the schedule includes a meeting when j is available to speak, then there corresponding truth assignment satisfies at least one of the literals in clause j. Thus, we have constructed a satisfying truth assignment of the 3SAT formula, as desired.

(2) *(10 points)*

Solve Exercise 11 in Chapter 8 of the textbook.

Solution

First, to prove that PLOT FULFILLMENT is in NP, we describe a polynomial-time verifier. Given an instance of the problem, and a proposed path P from s to t, the verifier first checks that P is really an s-t path in G, i.e. that it starts at s, ends at t, and every two consecutive vertices on P are joined by an edge in G; this is easily accomplished in linear time. Then it checks that for every thematic element T_i, at least one element of T_i belongs to P. This can also be accomplished in linear time, by constructing an array of size $|V(G)|$ that stores a single bit for every vertex according to whether it belongs to P, and then running through the elements of each set T_i using the array to check whether they belong to P.

Next, to prove that PLOT FULFILLMENT is NP-hard, we describe a reduction from HAMILTONIAN CYCLE. Given an instance of HAMILTONIAN CYCLE consisting of a directed graph $G' = (V', E')$ with n vertices and m edges, we create an instance of PLOT FULFILLMENT described by a directed graph $G = (V, E)$ constructed as follows. First, let r be an arbitrary vertex of G'. The graph G has the following vertices and edges.

- start node s, end node t
- vertices $x_{u,i}$ for all $u \in V' \setminus \{r\}$, $i \in [n-1]$
- edges $(s, x_{u,1})$ for all edges $(r, u) \in E'$
- edges $(x_{u,i}, x_{v,i+1})$ for all $1 \le i < n-1$ and all edges $(u, v) \in E'$ such that $u \neq r$ and $v \neq r$. $1 \le i < n-1$
- edges $(x_{u,n-1}, t)$ for all edges $(u, r) \in E'$.

The thematic elements are the sets $T_u = \{x_{u,1}, x_{u,2}, \ldots, x_{u,n-1}\}$ for each $u \in V' \setminus \{r\}$. Creating the graph G and listing its thematic elements takes time $O((m+n)n)$, so this is a polynomial-time reduction. To prove the correctness of the reduction, we must show two things.

1. If the graph G' contains a Hamiltonian cycle, then the graph G contains an s-t path that visits every set T_u. Write the vertices of the Hamiltonian cycle as $u_0, u_1, \ldots, u_n$, where $u_0 = u_n = r$. The graph G contains a path $s, x_{u_1,1}, x_{u_2,2}, \ldots, x_{u_{n-1},n-1}, t$. Our assumption that $u_0, \ldots, u_n$ is a Hamiltonian cycle with $u_0 = u_n = r$ implies that the set $\{u_1, \ldots, u_{n-1}$ contains each element of $V' \setminus \{r\}$, which in turn implies that the path we constructed in G visits every set T_u.

2. If the graph G contains an s-t path that visits every set T_u, then G' contains a Hamiltonian cycle. For $i = 1, \ldots, n-1$, let us refer to the vertex set $\{x_{u,i}\}$ as Layer i of the graph G. Define Layer 0 and Layer n to be the sets $\{s\}$ and $\{t\}$, respectively. By construction, every edge in G goes from Layer i to Layer $i+1$, for some $i = 0, \ldots, n-1$. Therefore, every path from s to t must be of the form

$s, x_{u_1,1}, x_{u_2,2}, \ldots, x_{u_{n-1},n-1}, t$ for some sequence $u_1, \ldots, u_{n-1}$. Supposing the path visits each set T_u, there are $n-1$ such sets and only $n-1$ vertices in the path that could belong to those sets, so every vertex $u \in V' \setminus \{r\}$ must appear exactly once in the list $u_1, u_2, \ldots, u_{n-1}$. Now, by the construction of the edge set of G, we know that every consecutive pair of vertices in the sequence $r = u_0, u_1, u_2, \ldots, u_{n-1}, u_n = r$ are joined by an edge of G'. Hence $u_0, \ldots, u_n$ is a Hamiltonian cycle in G', as desired.