CSC 373 - Algorithm Design, Analysis, and Complexity **Summer 2014**

Assignment 4: Due Friday August 9, Noon - Solutions

Please follow the instructions provided on the course website to submit your assignment. You may submit the assignments in pairs. Also, if you use **any** sources (textbooks, online notes, friends) please cite them for your own safety.
You can use those data-structures and algorithms discussed in CSC263 (e.g. merge-sort, heaps, etc.) and in the lectures by stating their name. You do not need to provide any explanation or pseudo-code for their implementation. You can also use their running time without proving them: for example, if you are using the merge-sort in your algorithm you can simply state that merge-sorts running time is $\mathcal{O}(n \log n)$.

Every time you are asked to design an efficient algorithm, you should provide both a short high level explanation of how your algorithms works in plain English, and the pseudo-code of your algorithm similar to what we've seen in class. State the running time of your algorithm with a brief argument supporting your claim. You must prove that your algorithm finds an optimal solution!

The Mute Prison [2.5]

An international prison manager, trying to keep swearing off his institution, wants to place inmates into cells so that no two inmates speak the same language. Given a table T of n inmates and m languages where $T[i, j] = 1$ if inmate i speaks language j and 0 otherwise; the manager wants to know whether there exists a subset S of the inmates where $|S| \geq k$ and no two inmates in S speak the same language. Let's call this problem The Mute Prison problem.

1. Show that The Mute Prison problem is NP-complete.

Solution:
The problem is in NP since we can exhibit a set of k inmates, and in polynomial time check whether any two of them speak the same language.

To show that the problem is NP-hard, we give a reduction from INDEPENDENT SET. Given an instance of INDEPENDENT SET $G(V, E), k$, we construct an inmate for every $v \in V$, and a language for every edge $e \in E$. We then build an array that says that inmate v speaks language e if edge e is incident to vertex v. The Mute Prison problem asks whether this array has a subset S of size at least k such that no two prisoners in S speak the same language.

We claim that this holds if and only if G has an independent set of size at least k. If such a subset S exists where $|S| \geq k$, then the corresponding set of nodes in G has the property that no two are incident to the same edge - so it is an independent set of size at least k. Conversely, if there is an independent set of size at least k in G, then the corresponding set of inmates has the property that no two speak the same language.

The Nonsense Prerequisites [2.5]

University of Chaos computer system crashed. While recovering it, you noticed that the prerequisite data was corrupted. Some prerequisites were added and shouldn't have been. For instance, the original prerequisite

path to get to CSC373 was to take 165 → 236 → 263 → 373. After the crash, the prerequisite 373 → 236 was added, thus creating a cycle from 236 → 263 → 373 → 236. At University of Chaos, we don't really care that much if prerequisites make sense, but rather just want to avoid having cycles. Your job is to break these cycles. In the example above, you could do so by removing one the following prerequisites: $236 \to 263, 263 \to 373$ or $373 \to 236$.

More formally, we define The Nonsense Prerequisites problem as follows: Suppose you are given a list n courses, you can construct a directed graph $G(V, E)$ where every vertex is a course, and the edge (i, j) directed $i \to j$ means course i is a prerequisite for course j. You are also given a positive integer k, and you want to return 1 if and only if there exists a subset $E' \subseteq E$ of edges whose removal makes G acyclic and $|E'| \leq k$.

1. Show that The Nonsense Prerequisites problem is in NP and give a reduction from VERTEX COVER to show that it is NP-hard.

Solution:

This problem is known as the **Feedback Arc Set Problem** (FAS), formally defined as follows:
Input: A directed graph $G(V, E)$ and a positive integer k.
Problem: Return 1 if and only there exists a subset $E' \subseteq E$ of edges such that $|E'| \leq k$ and the removal of E' makes G acyclic.

Recall topological sort takes linear time in the size of the graph, i.e. $\mathcal{O}(|V| + |E|)$ time. Given a subset of edges, we can thus check in linear time if the size of the subset E' is at most k, and if the $G(V, E \backslash E')$ is acyclic. Therefore FAS is in NP.

To show that FAS is NP-hard, we give a reduction from VERTEX COVER. Let $G(V, E), k$ be an instance of VERTEX COVER, which we transform to an instance of FAS $G'(V', E'), k'$ in linear time as follows:
For every vertex $v \in V$, we add two corresponding vertices $v_{in}, v_{out} \in V'$ that we connect by a directed edge $(v_{in} \to v_{out})$, and for every edge $(u, v) \in E$, we add the directed edges $(u_{out} \to v_{in})$ and $(v_{out} \to u_{in})$* to E'. Finally we set $k' = k$; and claim that G has a vertex cover of size at most k iff G' has a FAS of size at most k'.

Proof. ($\implies$) Suppose G has a vertex cover S where $|S| \leq k$. For every $v \in S$, consider removing the edge (v_{in}, v_{out}) in G'. The resulting graph is acyclic. If a cycle in G' enters a vertex in v_{in}, it can only leave through the edge (v_{in}, v_{out}) given our construction. Similarly, a cycle can only enter v_{out} through the edge (v_{in}, v_{out}).

Thus if a cycle in G' uses the edge (u_{out}, v_{in}), it must use the edge (u_{in}, u_{out}) and the edge (v_{in}, v_{out}). At least one of the these two edges would have been removed since at least one of u or v must be in the vertex cover to cover the edge $(u, v) \in E$.

Therefore if G has a vertex cover of size at most k then G' has a feedback arc set of size at most $k' = k$ as well.

($\impliedby$) Conversely, suppose G' has a FAS S' of size at most k'. Without loss of generality, we assume that the only edges removed are of the form (u_{in}, u_{out}) because if some other edge (u_{out}, v_{in}) is removed, then all cycles in which this edge participated would have included the edge (v_{in}, v_{out}) too and we could remove this edge instead.

*We drop the arrow notations for the rest of the proof and assume $(u, v) \in V'$ to be the edge directed from u to v

We claim that the set of vertices $v \in V$ for which the corresponding edge (v_{in}, v_{out}) is removed in E' forms a vertex cover for G. Suppose not, and suppose there is some edge $(u, v) \in E$ not covered. Then in G' the cycle $(u_{in}, u_{out}), (u_{out}, v_{out}), (v_{in}, v_{out}), (v_{out}, u_{in})$ is unbroken by the removal of the edges contradicting the assumption. Thus (G, k) must have a vertex cover of size at most $k = k'$. □

Since FAS is in NP and is NP-hard, it follows that it is NP-complete.

T-rex Christmas [2.5]

Suppose there is a group of n "short armed" people[†] sitting around a table, wanting to exchange their Christmas presents. If person i has a present for person j, person i can send their wrapped package either clockwise or counter clockwise around the table. Suppose the people are numbered from 0 to $n-1$ clockwise around the table. Because these people's arms are short, passing presents around the table is quite tiring, and you, *kind supervisor*, want to minimize the number of *passes* on the table. The number of passes P_i between $(i, i+1[n])$[‡] is the total number of presents that have to go between person i and person $i+1$ modulo n. The total number of passes of the table is $\max_{0 \le i \le n-1} P_i$.

The T-rex Christmas Problem: You are given the cycle from 0 to $n-1$ people and a list $\mathcal{L}$ of pairs (i, j), person i has a present for person j. Your goal is to decide which way to send the presents around the table so as to minimize the number of passes on the table.

1. Give a $\{0, 1\}$-integer programming formulation of the T-rex Christmas Problem.

2. Give an LP relaxation and rounding scheme of your integer program above. Prove that the rounded solution is also a solution to the integer program, and show that the rounding scheme attains a 2-approximation.

Solution:

This problem is known as the **SONNET ring loading problem** and it arises in communication networks quite often. In this setting, the network consists of n nodes numbered 0 to $n-1$ clockwise around the cycle, and the pair (i, j) denotes the call originated at node i and destined to node j. The call can be routed clockwise or counterclockwise around the ring. We first give a $\{0, 1\}$-integer programing formulation to the problem:
For the k^{th} pair (i, j), we create two 0-1 variables x_0^k, x_1^k that correspond to the two possibilities of how to send the gift (or route the call): 0 for clockwise and 1 for counterclockwise. Clearly, we must have $x_0^k + x_1^k = 1$.

Notice that for any link $(i, i+1[n])$, one of the two routings of the k^{th} pair (i, j) will use this link. We capture this, by introducing a variable $\alpha(k, i)$ where $\alpha(k, i) = 0$ if the clockwise routing uses this link and $\alpha(k, i) = 1$ if the counterclockwise routing uses this link.

[†] T-rex humans
[‡] [n] means modulo n

Our formulation is thus the following:

$$\begin{aligned}
&\text{minimize } P \\
&\text{s.t. } x_0^k + x_1^k = 1 \qquad \forall k \in \mathcal{L} \\
&\quad \sum_{k \in \mathcal{L}} x^k_{\alpha(k,i)} \leq P \qquad \forall i \in [0, 1, ..., n-1] \\
&\quad x_0^k \in \{0, 1\} \qquad \forall k \in \mathcal{L} \\
&\quad x_1^k \in \{0, 1\} \qquad \forall k \in \mathcal{L} \\
&\quad P \geq 0
\end{aligned}$$

We relax the integer program to a linear program by changing the integrality constraints to $0 \leq x_0^k \leq 1, 0 \leq x_1^k \leq 1$. Notice that the constraint ≤ 1 can be dropped since it is already covered by the first constraint in the LP ($x_0^k + x_1^k = 1$).

Let $\tilde{P}$ with values $\tilde{x}_0^1, ..., \tilde{x}_0^m, \tilde{x}_1^1, ..., \tilde{x}_1^m$, where m is the total number of presents, be the optimal solution to the relaxed LP. We will construct an integral solution $\hat{x}_0^1, ..., \hat{x}_0^m, \hat{x}_1^1, ..., \hat{x}_1^m$ by performing a rounding scheme as follows: For every k^{th} pair in $\mathcal{L}$, if $\tilde{x}_0^k \geq 1/2$ we round $\hat{x}_0^k = 1, \hat{x}_1^k = 0$. Otherwise we set $\hat{x}_0^k = 0, \hat{x}_1^k = 1$.

We claim that this rounding gives us a $\frac{1}{2}$-approximation. Let OPT denote the optimal IP solution. Notice that for any $i \in 1, 2..., n-1$ we have:

$$\begin{aligned}
\sum_{k \in \mathcal{L}} \hat{x}^k_{\alpha(k,i)} &\leq 2 \sum_{k \in \mathcal{L}} \tilde{x}^k_{\alpha(k,i)} \\
&\leq 2\tilde{P} \\
&\leq 2OPT
\end{aligned}$$

Vertex Cover *again...* [2.5]

Consider this variation of Vertex Cover: Suppose you have a graph $G(V, E)$ with costs on both the vertices and edges: $C_v : V \to \mathbb{R}^+$ and $C_e : E \to \mathbb{R}^+$. Your goal is to construct a vertex cover S so as to minimize the total cost of S, where the cost of S is computed as follows: For every vertex $w \in S$ you pay $C_v(w)$; and for every edge f **not** covered by an element in S, you pay $C_e(f)$. In other words, you are allowed to leave some edges uncovered, but you pay for them.

More formally, given $G(V, E, C_v, C_e)$, you want a set $S \subseteq V$ such that the cost below is minimized:

$$cost(S) = \sum_{w \in S} C_v(w) + \sum_{\substack{(u,z) \in E \\ u,z \notin S}} C_e((u, z))$$

1. Give a $\{0, 1\}$-integer programming formulation of the problem above.
2. Give an LP relaxation and rounding scheme of your integer program above. Prove that the rounded solution is also a solution to the integer program, and show that the rounding scheme attains a 3-approximation.

Solution:
We introduce two new variables x_i for every $i \in V$, and y_{ij} for every $(i, j) \in E$, where $x_i = 1$ iff vertex i is

in the vertex cover and $y_{ij} = 1$ iff edge (i, j) is not covered.

$$\begin{aligned} \text{minimize } & \sum_{i \in V} x_i C_v(i) + \sum_{(i,j) \in E} y_{ij} C_e(i,j) \\ \text{s.t. } & x_i + x_j + y_{ij} \geq 1 \qquad \forall (i,j) \in E \\ & x_i \in \{0,1\} \qquad \forall i \in V \\ & y_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \end{aligned}$$

The rounding scheme for this LP is to set each $\tilde{x}_i$ to 1 if $x^* \geq \frac{1}{3}$ and every $\tilde{y}_{ij}$ to 1 if $y^*_{ij} \geq \frac{1}{3}$, and 0 otherwise for both variables. This rounding gives us a $\frac{1}{3}$ approximation; the rest of the argument is similar to the $\frac{1}{2}$-approximation we did in class.

Notice that setting the rounding at the $\frac{1}{2}$ threshold does not cover the case where for a given edge (i, j), the LP gives us $x^*_i = x^*_j = y^*_{ij} = \frac{1}{3}$. This edge is covered in the optimal LP but wouldn't be cover in our IP, have we used the $\frac{1}{2}$ rounding.