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

Assignment 4: Due Friday August 9, Noon

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.

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 $165 \to 236 \to 263 \to 373$. After the crash, the prerequisite $373 \to 236$ was added, thus creating a cycle from $236 \to 263 \to 373 \to 236$. 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.

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.

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.

*T-rex humans
†[n] means modulo n