Practice Problems for Final Exam

The final exam will be on **Tuesday, December 10th from 5:30-8pm**. The exam will be closed-book and closed-notes, but you will be allowed three cheat-sheets (front and back). It will be a cumulative exam, with an emphasis on the material covered since the last midterm exam.

Disclaimer: These are practice problems. They do not necessarily reflect the actual length, difficulty, or coverage for the exam. You should also be sure to look over your homework problems as well as the in-class practice problems. You should also review the practice problems and exam questions from midterms 1 and 2.

Problem 0. You should expect at least one problem in which you will be asked to work an example of one of the algorithms we have presented in class or some NP-complete reduction we covered.

Problem 1. Short Answer Questions

a. Technically advanced aliens come to Earth and show us that some known NP-hard problem cannot be solved faster than $O(n^{100})$ time. Does this resolve the question of whether P = NP? (Explain briefly.)

b. **True or False:** If a graph G has a vertex cover of size k, then it has a dominating set of size k or smaller.

c. Suppose that $A \leq_p B$, the reduction runs in $O(n^2)$ time, and B can be solved in $O(n^4)$ time. What can we infer about the time needed to solve A? (Explain briefly.)

d. Suppose that $A \leq_p B$ and there is a factor-2 approximation to problem B, then which of the following necessarily follows:

   i. There is a factor-2 approximation for A.

   ii. There is a constant factor approximation for A, but the factor might not be 2.

   iii. We cannot infer anything about our ability to approximate A.

Problem 2. Show that the following problem is NP-complete.

Balanced 3-coloring (B3C): Given a graph $G = (V, E)$, where $|V|$ is a multiple of 3, can G be 3-colored such that the sizes of the 3 color groups are all equal to $|V|/3$. That is, can we assign an integer from $\{1, 2, 3\}$ to each vertex of G such that no two adjacent vertices have the same color, and such that all the colors are used equally often.

Hint: Reduction from the standard 3-coloring problem (3COL).

Problem 3. In ancient times, King Arthur had a large round table around which all the knights would sit. Unfortunately, some knights hate each other, cannot be seated next to each other. There are n knights altogether, $\langle v_1, v_2, ..., v_n \rangle$, and the king has given you a list of pairs of the form $\langle v_i, v_j \rangle$, which indicates that knights v_i and v_j hate each other. You are asked to write a program to determine if it is possible to seat the knights about the table, called the angry knight seating problem (AKS). Prove that AKS is NP-complete.

Problem 4. The set cover optimization problem is: Given a pair (X, S), where X is a finite set and a $S = \{s_1, s_2, ..., s_n\}$ is a collection of subsets of X, find a minimum sized collection of these sets C whose union equals X. Consider a special version of the set-cover problem in which each element of X occurs in at most three sets of S. Present an approximation algorithm for this special version of the set cover problem with a ratio bound of 3. Briefly derive the ratio bound of your algorithm.

Problem 5. Recall the following problem, called the Interval Scheduling Problem. We are given a set $S = \{1, 2, ..., n\}$ of n activity requests, each of which has a given start and finish time, $[s_i, f_i]$. The objective is to compute the maximum number of activities whose corresponding intervals do not overlap. In class we presented an optimal algorithm greedy algorithm. We will consider some alternatives here.

a. **Earliest Activity First (EAF):** Schedule the activity with the earliest start time. Remove all activities that overlap it. Repeat until no more activities remain. Give an example to show that, not only is EAF not optimal, but it may be arbitrarily bad, in the sense that its approximation ratio may be arbitrarily high.

b. **Shortest Activity First (SAF):** Schedule the activity with the smallest duration $(f_i - s_i)$. Remove all activities that overlap it. Repeat until no more activities remain. Give an example to show that SAF is not optimal.

c. Prove that SAF has an approximation ratio of 2, that is, it schedules at least half as many activities as the optimal algorithm.