SPRING 2010: **COT 6936** Topics in Algorithms

[Homework 1; Due Feb 11 at start of class]

General submission guidelines and policies: Add the following signed statement. Without this statement, your homework will not be graded.

> I have adhered to the collaboration policy for this class. In other words, everything written down in this submission is my own work. For problems where I received any help, I have cited the source, and/or named the collaborator.

Read the handout on **Homework guidelines and collaboration policy** from your course website before you start on this homework. This is very important.

As mentioned in that handout, the problem labels mean the following: (**Exercise**) – turn in, but will not be graded; (**Regular**) – turn in and will be graded; or (**Extra Credit**) – if turned in, they will be graded but extra credit will be given only if it is completely correct. (**Extra Credit**) scores wil be used **only** if your grade is on the border between two grades.

Problems

0. (**Regular**) Did you follow the instructions above?

1. (**Exercise**) Write down precisely (see Problem 8 below for an example) the *decision* **and** *optimization* versions of the VertexCover problem.

   This homework was inspired by an excellent question asked in class by one of the students. In this homework, we will explore three greedy algorithms for VertexCover, which we will refer to as GreedyOne, GreedyBoth, and GreedyMax. By the end of this homework, you should be able to understand why GreedyBoth is the best greedy approximation algorithm for the VertexCover problem.

   GreedyOne works as follows: *Initialize the vertex cover to the empty set. Repeatedly pick an (arbitrary) uncovered edge and (arbitrarily) choose one of its endpoints to add to the vertex cover.*

2. (**Exercise**) Show that GreedyOne produces a vertex cover.

3. (**Regular**) Show that the performance ratio of GreedyOne cannot be bounded by any constant. (Hint: assume the contrary and construct graphs to contradict. Also, note that whenever the algorithm is allowed to make an arbitrary choice, you may assume that it makes the worst choice.)

   GreedyBoth was discussed in class and works as follows: *Initialize the vertex cover to the empty set. Repeatedly pick an (arbitrary) uncovered edge and choose both of its endpoints to add to the vertex cover.*

4. (**Exercise**) Convince yourself that GreedyBoth produces a vertex cover and that its performance ratio is bounded by 2.

5. (**Regular**) Show that there exists graphs for which the GreedyBoth algorithm cannot achieve a performance ratio better than 2.

   GreedyMax works as follows: *Initialize the vertex cover to the empty set. While there are uncovered edges, repeatedly pick a vertex of maximum degree (if more than one candidate exists, pick one arbitrarily), add it to the vertex cover, and delete all covered edges.*

6. (**Exercise**) Convince yourself that GreedyMax produces a vertex cover and that its performance ratio is bounded by $\log n$. (See slide 16 and 17 from Lec 4).

   Using the steps outlined below, show that there exists graphs for which GreedyMax cannot achieve a performance ratio better than $\log n$.

   Construct a bipartite graph $G(A, B, E)$ (i.e., graph G has vertex bipartitions A and B and the edges in set E only connect vertices in A with vertices in B). Let the size of set B be equal to K. The intention is to make set A as large as possible. Divide the set A into K subsets $A_1, \ldots, A_K$. Set A_i is adjacent to i vertices in B in such a way that no two vertices in A_i share a common neighbor. Let the size of set A_i be $\lfloor K/i \rfloor$.

7. (**Regular**)

   (a) Argue that the number of vertices in the graph, $n = |A| + |B| = K + K \log K = \Theta(K \log K)$. (See Appendix A of Cormen, Leiserson, Rivest and Stein and read up the notes on *Harmonic series*). Thus $\log n = \Theta(\log K)$.

   (b) Show that the size of an optimal vertex cover is K.

   (c) Show that the (only) vertex in A_K has maximum degree in G.

   (d) Show that GreedyMax can pick all vertices in A_K, followed by all vertices in A_{K-1}, and so on, thus picking all vertices in A and no vertices in B.

   (e) Show that the performance ratio of GreedyMax on this graph is no less than $\Omega(\log n)$.

   The decision version of the SetCover problem can be stated as follows:

   Input: Universe, $U = \{x_1, \ldots, x_n\}$. Collection of sets, $\mathcal{S} = \{S_1, \ldots, S_m\}$, such that $S_i \subseteq U, 1 \leq i \leq m$. Integer K.

   Question: Is there a collection of K sets from $\mathcal{S}$ whose union equals U?

   The obvious version of GreedyMax as applied to the maximization version of SetCover is as follows: *repeatedly pick the set from $\mathcal{S}$ containing the largest number of uncovered items from the universe.*

8. (**Exercise**) Argue that using the example in Problem 7 above one can show that GreedyMax cannot have a performance ratio better than $\Omega(\log n)$ for SetCover.

   Given an instance of the SetCover, assume that no item from the universe U is part of more than R sets from the collection $\mathcal{S}$. Now consider algorithm GreedyAll (a simple generalization of GreedyBoth above), which works as follows: *initialize the set cover to the empty collection; repeatedly pick an arbitrary uncovered item from the universe and add (to the set cover) all sets that contain it.*

9. (**Regular**) Show that GreedyAll achieves a performance ratio of R. (Hint: Follow the analysis for GreedyBoth. For SetCover, which is a better approximation algorithm – GreedyAll or GreedyMax?