Introduction to Algorithms	**Problem Set 10**
CS 4820, Spring 2014	**Due 11:59pm Monday, May 5**

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. *(15 points in total)* Recall that in the Knapsack Problem, one is given a set of items numbered $1, 2, \ldots, n$, such that the i-th item has value $v_i \geq 0$ and size $s_i \geq 0$. Given a total size budget B, the problem is to choose a subset $S \subseteq \{1, 2, \ldots, n\}$ so as to maximize the combined value, $\sum_{i \in S} v_i$, subject to the size constraint $\sum_{i \in S} s_i \leq B$.

1.a. *(5 points)* Consider the following *greedy algorithm*, GA.

1. Eliminate items whose size s_i is greater than B.
2. For each remaining i, compute the *value density* $\rho_i = v_i / s_i$.
3. Sort the remaining items in order of decreasing ρ_i.
4. Choose the longest initial segment of this sorted list that does not violate the size constraint.

Also consider the following *even more greedy algorithm*, EMGA.

1. Eliminate items whose size s_i is greater than B.
2. Sort the remaining items in order of decreasing v_i.
3. Choose the longest initial segment of this sorted list that does not violate the size constraint.

For each of these two algorithms, give a counterexample to demonstrate that its approximation ratio is not bounded above by any constant C. (Use different counterexamples for the two algorithms.)

1.b. *(5 points)* Now consider the following algorithm: run GA and EMGA, look at the two solutions they produce, and pick the one with higher total value. Prove that this is a 2-approximation algorithm for the Knapsack Problem, i.e. it selects a set whose value is at least half of the value of the optimal set.

1.c. *(5 points)* Use part (b) to show that the the dynamic-programming based approximation algorithm for Knapsack presented in class can be modified so that it achieves an approximation ratio of at most $1 + \varepsilon$ with running time $O(n^2/\varepsilon)$. In your solution, it is not necessary to repeat the proof of correctness of the algorithm presented in class.

2. *(10 points in total)*

2.a. *(5 points)* Consider the special case of unweighted set cover in which every element belongs to at most k sets. Design a polynomial-time $O(k)$-approximation algorithm for this problem.

2.b. *(5 points)* Consider the same specialization of weighted set cover and design a polynomial-time $O(k)$-approximation algorithm for it. (If you are confident in your solution of this part, you can simply write, "Special case of 2b" as your solution to 2a. In that case we'll simply double your score for this part of the problem so that it counts for 10 points instead of 5.)

3. *(10 points in total)* You are asked to create a schedule for the upcoming season of a sports league. Based on records from the previous year, it was decided which pairs of teams should play against each other in the upcoming season. Games are played on Sundays and each team can play at most one game on the same Sunday. The goal is to schedule all games in such a way that the number of weeks is minimized.

This scheduling problem is equivalent to the following problem on graphs (called EDGE COLORING): You are given a graph G with vertices $t_1, \ldots, t_n$ corresponding to the teams. The edges of the graph correspond to the set of games that are to be scheduled in the upcoming season. An *edge coloring* of G assigns to each edge (t_i, t_j) in the graph a "color" w_k such that no vertex is incident to more than one edge of the same color. (This constraint ensured that all games (t_i, t_j) with the same color w_k can be held in the same week.) Your goal is to find an edge coloring of G that uses as few colors as possible.

Give an efficient algorithm to compute an edge coloring of G that uses at most $2d - 1$ different colors, where d is the maximum degree in the graph. Show that the approximation ratio of this algorithm is at most 2.