CSC 545 **Homework 3 (version 1)** **Spring 2015**

This homework is due Thursday, March 26, at the start of class. The questions are drawn from the material in the lectures and Chapter 16 of the text on *greedy algorithms*.

The homework is worth a total of 100 points. When questions with several parts do not specify the points for each part, each part has equal weight.

For questions that ask you to design a *greedy* algorithm, prove that your algorithm is correct making use of a lemma of the following form:

> **Lemma** *If a partial solution P is contained in an optimal solution, then the greedy augmentation of P is still contained in an optimal solution.*

Prove the lemma using an exchange-style argument, where you transform the optimal solution to contain the augmentation and argue that this transformation does not worsen the solution value.

Remember to write on just one side of a page, do not use scrap paper, put your answers in the correct order, and staple your pages together. If you can't solve a problem, state this, and write only what you know to be correct. Neatness and conciseness count.

(1) **(Counterexamples to greedy procedures)** (15 points) Prove that the following greedy procedures for the Activity Selection Problem are *not* correct. Each procedure considers the activities in a particular order, and selects an activity if it is compatible with those already chosen.

   (a) The activities are considered in order of increasing *duration*.
   (b) The activities are considered in order of increasing *start-time*.
   (c) The activities are considered in order of increasing *number of overlaps* with the remaining compatible activities. (This is a dynamically-determined order.)

   (Note: To prove an optimization procedure is not correct, it suffices to construct an instance of the problem that has a feasible solution that is better than the one the procedure outputs.)

(2) **(Trip refueling)** (25 points) Suppose you want to travel from city A to city B by car, following a fixed route. Your car can travel m miles on a full tank of gas, and you have a map of the n gas stations on the route between A and B that gives the number of miles between each station.

   Design a greedy algorithm to find a way to get from A to B without running out of gas that minimizes the total number of refueling stops, in $O(n)$ time. Prove that your algorithm finds an optimal sequence of stops.

(3) **(Minimizing average completion-time)** (25 points) Suppose you are given a collection of n tasks that need to be scheduled. With each task, you are given its duration. Specifically, task i takes t_i units of time to execute, and can be started at any time. At any moment, only one task can be scheduled.

   The problem is to determine how to schedule the tasks so as to minimize their *average completion-time*. More precisely, if c_i is the time at which task i completes in a particular schedule, the average completion-time for the schedule is $\frac{1}{n}\sum_{1\le i\le n} c_i$.

   (a) (25 points) Design an efficient *greedy* algorithm that, given the task durations t_1, t_2, ..., t_n, finds a schedule that minimizes the average completion-time, assuming that once a task is started it must be run to completion.

Analyze the running time of your algorithm, and prove that your algorithm is correct using a lemma of the required form.

(b) **(bonus)** (10 points) Suppose with each task we also have a *release time* r_i, and that a task may not be started before its release time. Furthermore, tasks may be *preempted*, in that a scheduled task can be interrupted and later resumed, and this can happen repeatedly.

Design an algorithm that finds a schedule that minimizes the average completion-time in this new situation. Analyze its running time and prove that it is correct.

(4) **(Least change)** (35 points) Suppose that for a given amount of cents n, you want to choose coins whose total value adds up to n, where the coins have denominations that come from a fixed assortment of values. The problem is to choose coins that add up to n, while minimizing the number of coins used. In other words, you want to make n cents in change, using the *fewest* possible number of coins. You may assume that there are as many coins as you wish of each denomination.

(a) (20 points) Design a greedy algorithm that makes change for the US system of coins, in other words, where the coin denominations are quarters, dimes, nickels, and pennies. Prove that your algorithm finds an optimal solution with the fewest number of coins, using a lemma of the required form.

(b) (5 points) Prove that your greedy algorithm from Part (a) does *not* make optimal change for all systems of coins. More precisely, demonstrate a system of coin denominations that does contain the penny, and an explicit value of n, such that your greedy algorithm from Part (a) does not use the fewest number of coins.

(c) (10 points) Suppose that the system of coins has denominations $\{a^0, a^1, \ldots, a^k\}$ for integers $a > 1$ and $k \geq 1$. Prove that your greedy algorithm makes optimal change for this system.

(d) **(bonus)** (10 points) Using dynamic programming, design an algorithm that makes optimal change for any system of k coins that contains the penny, and which runs in $O(kn)$ time.

(Note: Since by Part (b), a greedy change-making algorithm will in general not be optimal for *arbitrary* coin denominations, your proof of correctness for Part (a) will have to make use of *special* properties of the US coin denominations.)

Note that Problems (3)(b) and (4)(d) are bonus questions. They are not required, and their points are not added to regular points.