This homework is due Wednesday, November 7, at the start of class. The questions are drawn from the material in class, and in Chapter 16 of the text, on greedy algorithms.

The homework is worth a total of 100 points. If a point breakdown is not given for each part of a problem, each part has equal weight.

For questions that ask you to design a greedy algorithm, prove that your algorithm is correct using a greedy augmentation 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 so it contains the augmentation and then argue that the transformation does not worsen the solution value. Finally, prove correctness of your algorithm in a theorem that makes use of the lemma.

Remember to (a) put your name at the top of every page, (b) start each problem on a new page, (c) write on just one side of a page and do not use scrap paper, (d) put your answers in the correct order, and (e) 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 demonstrate a counterexample: 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)** (30 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 $\frac{1}{n} \sum_{1 \leq i \leq n} c_i$. 

(a) (30 points) Design an efficient greedy algorithm that, given the task durations \( t_1, t_2, \ldots, t_n \), 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) (Room scheduling) (30 points) Suppose you have \( n \) classes that you want to schedule in rooms. Each class has a fixed time interval at which it is offered, and classes whose times overlap cannot be scheduled in the same room. There are enough rooms to schedule all the classes.

Design a greedy algorithm to find an assignment of classes to rooms that minimizes the total number of rooms used, in \( O(n \log n) \) time. Prove that your algorithm finds an optimal assignment using a lemma of the required form.

Note that Problem (3)(b) is a bonus question, and is not required.