Homework #4

You do not need to turn in these problems. The goal is to be ready for the in-class quiz that will cover the same or similar problems, and to prepare you for the exams.

Problem 1: Discipline of Children

Your job is to arrange n rambunctious children in a straight line, facing front. You are given a list of m statements of the form "i hates j". If i hates j, then you do not want to put i somewhere behind j because then i is capable of throwing something at j. Give an algorithm that orders the line, or says that such an order does not exist, in $O(m+n)$ time.

Problem 2: Course room assignment

You have n lectures that need to be allocated rooms. Each lecture i has its own start time s_i and finish time f_i. Your goal is to assign lectures to rooms using as few rooms as possible. Of course, if two lectures overlap in time, they cannot share a room.

(a) Consider the following greedy algorithm skeleton:

```
Sort lectures in some way (to be determined)
d := 0   ← number of allocated classrooms

for j = 1 to n {
   if (lecture j is compatible with some classroom k)
      schedule lecture j in classroom k
   else
      allocate a new classroom d + 1
      schedule lecture j in classroom d + 1
      d := d + 1
}
```

You remember that you learned about interval scheduling in class, and for that problem you needed to sort by finish times. Show by counter-example that sorting by finish times doesn't work here.

(b) Prove that sorting by start time always gives the optimal solution.

(c) Show that your algorithm is $O(n \log n)$.

Problem 3: Dijkstra's algorithm with negative weights

Edge weights in weighted graphs don't have to represent lengths. Imagine you are trying to model business scenarios and edge weights represent costs and benefits of different paths through the "space of business choices". In this case it makes sense to have weights that are both positive (spend money) and negative (earn money). A "shortest path" represents a set of business choices with lowest total cost. (Assume that there is no cycle of negative total weight. If there were a cycle of negative total weight, then going around this cycle arbitrarily many times, you could get the total weight of a path to be as negative as you like, and then shortest paths are not well defined.)

Unfortunately, Dijkstra's algorithm doesn't work if some weights can be negative. Explain why Dijkstra's algorithm could give incorrect shortest paths if some weights are negative. Where does the proof of correctness we did in class fail? (Later in the class, we'll learn about a shortest path algorithm that *can* handle negative weights.)

Problem 4: Phone Base Stations

Consider a long, quiet country road with houses scattered sparsely along it. (We can picture the road as a long line segment, with an eastern endpoint and a western endpoint.) Further, let's suppose that despite the bucolic setting, the residents of all these houses are avid cell phone users. You want to place cell phone base stations at certain points along the road, so that every house is within 4 miles of one of the base stations. Give a greedy efficient algorithm that achieves this goal, using as few base stations as possible. Prove that the greedy choice that your algorithm makes is the optimal choice.

Problem 5: Fractional Knapsack Problem

A thief is given the choice of n objects to steal, but only has one knapsack with a capacity of taking M weight. Each object i has weight w_i, and profit p_i.

(a) First, suppose that the objects are divisible (e.g., the thief is in a cheese shop and the items are rolls of cheese that can be cut). For each object i, if a fraction x_i, $0 \leq x_i \leq 1$ (taking $x_i = 1$ would be taking the entire object) is placed in the knapsack, then the profit earned is $p_i x_i$. Come up with an efficient greedy algorithm for maximizing the profit of the thief. Prove the correctness and running time.

(b) Now, suppose the items cannot be taken fractionally. In other words, the thief can either take an entire item or leave it behind. Suppose we also know the following: the order of these items when sorted by increasing weight is the same as their order when sorted by decreasing value. Give a greedy algorithm to find an optimal solution to this variant of the knapsack problem. Prove the correctness and running time.