CS314: Algorithms
Midterm 1 review
In class, March 15

Problems

1. **Traveling the Trails** (Taken from first solved exercise in Ch. 4 of text, page 183)

   Suppose your friends have decided to go camping. They want to hike as much as possible during the day, but (since they've watched *The Blair Witch Project* too often) they don't want to do any hiking after dark. On the map, they've identified a large number of good stopping points for camping, and they're proposing the following system. Every time they come to a good stopping point, they'll determine whether they can make it to the next one before nightfall. If they can make it, they keep hiking; otherwise, they stop. They claim that this system must be good, since it minimizes the number of stops they need to make.

   Being a good algorithms student, you immediately recognize a greedy algorithm and become skeptical. Does this really minimize the number of stopping points?

   To formalize the problem, we'll model the trail as a line segment of length L, and assume your friends can hike d miles per day. We'll assume potential stopping points are labeled $x_1, x_2, \ldots, x_n$ along the segment, starting with 0 at the beginning of the segment. We'll also assume (quite generously) that your friends are correct in their estimation of whether they can make it to the next good stopping point before dark. Finally, we'll say a set of the stopping points is valid if the distance between each adjacent pair is at most d, the first is at most d from 0, and the last is at most d from the end of the segment.

   We now state the question formally: is your friends greedy algorithm optimal, in the sense that it finds a valid set that is as small as possible?

2. **Edge in an MST** (Taken from third solved exercise in Ch. 4 of text, page 188)

   Suppose you are given a connected graph G, with edge costs that you may assume are all distinct. G has n vertices and m edges. A particular edge e of G is specified. Give an algorithm with running time $O(m + n)$ that decides whether e is contained in a minimum spanning tree of G.

3. **Maximizing Fun**

   A company is planning a party for its employees. The employees in the company are organized into a strict hierarchy, that is, a tree with the company president at the root. The organizers of the party have assigned a real number to each employee measuring how fun the employee is. In order to keep things social, there is one restriction on the guest list: an employee cannot attend the party if their immediate supervisor is present. On the other hand, the president of the company must attend the party, even though she has a negative fun rating; it is her company, after all. Give an algorithm that makes a guest list for the party that maximizes the sum of the fun ratings of the guests.

4. **Adding an edge**

Let $G = (V, E)$ be an undirected graph with costs $c_e \geq 0$ on the edges $e \in E$. Assume you are given a minimum-cost spanning tree T in G. Now assume that a new edge is added to G, connecting two nodes u and v with cost c.

(a) Give an algorithm to test if T remains the minimum cost spanning tree with the new edge added to G. Can you make your algorithm run in $O(m)$ time? How about $O(n)$ time?

(b) Suppose T is no longer the MST. Given a linear time $O(m+n)$ algorithm to update the tree T to be the new MST.

5. **Moving on a Checkerboard**

Suppose that you are given an $n \times n$ checkerboard and a checker. You must move the checker from the bottom edge of the board to the top edge of the board according to the following rule. At each step you may move the checker to one of three squares:

1) the square immediately above
2) the square that is one up and one to the left (but only if the checker is not already in the leftmost column)
3) the square that is one up and one to the right (but only if the checker is not already in the rightmost column)

Each time you move from square x to square y, you receive $p(x, y)$ dollars. You are given a list of the values $p(x, y)$ for each pair (x, y) for which a move from x to y is legal. Do not assume that $p(x, y)$ is positive.

Give an algorithm that figures out the set of moves that will move the checker from somewhere along the bottom edge to somewhere along the top edge while gathering as many dollars as possible. You algorithm is free to pick any square along the bottom edge as a starting point and any square along the top edge as a destination in order to maximize the number of dollars gathered along the way. What is the running time of your algorithm?