CSOR W4231: Midterm 2 Practice Problems (Preliminary Draft) Fall 2019

These problems are ungraded, and are intended as a study aid. Solutions will also be posted on canvas. They are of similar flavor to the problems that will appear on the midterm. The midterm is closed book, closed notes; you are allowed to have 1 page of notes (double-sided). It will cover up material up to and including lecture 14. There are 5 problems.

Problem: Miners. Assume we have n miners and that each miner is assigned to one of m gold mines. Each mine will produce a different profit depending on the number of miners assigned to it. Let the profit of mine i with j miners assigned to it be $p_{i,j}$.

a) Describe an algorithm that produces an assignment of the miners to the m mines that produces the most profit. (Hint: consider a dynamic programming solution which uses the quantity $P[i, j]$, denoting the profit of the optimal assignment of the first j miners to the first i mines. Make sure to state what is your recurrence.)

b) Can we improve the runtime if we are guaranteed that the *law of diminishing returns* holds? In particular, assume that for each mine, each additional miner assigned to it can't result in a larger increase in profits than the previous miner? (As an example, if we have $p_{1,1} = 4$, $p_{1,2} = 6$, and $p_{1,3} = 7$, then the law of diminishing returns holds since the first miner adds a profit of 4 while the second and third add profits of 2 and 1 respectively). Design a greedy algorithm for this problem.

Problem: Summing numbers. Suppose you are given n positive integers $\{a_1, \ldots a_n\}$, and a positive integer bound B. The goal is to take a subset of the n integers whose sum is as close as possible to B without exceeding it. Consider the greedy strategy that first chooses the largest integer $a_i \leq B$, then subtracts a_i from the bound B and repeats with the remaining integers and the new bound $B' = B - a_i$. Either prove this strategy is optimal or give a counterexample where it performs sub-optimally.

Problem: Navigating in Manhattan. Although the streets in Manhattan are grid-like[1], it may still be tough to navigate driving because of the one-way streets. You are to design an algorithm to help your taxi driver get from an (avenue, street) intersection to another (avenue, street) intersection, given a map of the Manhattan. The map of the Manhattan is given as a graph where the nodes are such intersections, and the (directed) edges connect two adjacent intersections (ie, where the first or the second coordinate differs by exactly 1).

Design an algorithm for finding the shortest path from a start intersection to the end intersection in time $O(n + m)$, where n is the number of intersections and m is the number of edges, in the following two cases:

[1] Ok, unless we venture too much south, but we'll assume we don't.

a) Assume that it takes 1 minute to traverse any given edge (adjacent intersections).

b) Recognizing that avenue blocks are longer, assume that all edges have an associated time-to-traverse number, which is either 1 or 2, given as input as well.

Bonus: suppose the time-to-traverse numbers are either 1 or 2.3.

Problem: Planning a Party. You want to plan a company party. The company has a hierarchical binary-tree structure; that is, the supervisor relation forms a binary tree rooted at the president. The personnel office has ranked each employee with a conviviality rating, which is a real number. In order to make the party fun for all attendees, the presiden does not want both an employee and his or her immediate supervisor to attend.

You are given the tree that describes the structure of the corporation. You know for each node, the left-child, right-child and parent Each node of the tree also holds the name of an employee and that employee's conviviality ranking.

Give a dynamic programming recurrence that find the set of employees of maximum total conviviality that you can invite. Be sure to explain what your variables represent.

Explain what the running time of a bottom-up implementation of the algorithm would be. You do not have to give pseudocode, or a proof of optimal substructure.

Problem: Lazy Min Heap. The LAZY-MIN-HEAP data structure supports INSERT and DELETE-MIN operations. Let L be a linked list and H be a normal heap. The operation INSERT(x) is implemented by inserting x into the linked list x. The operations DELETE-MIN is implemented by 1) Inserting all the items in L into the heap H. 2) Setting the linked list L to be empty. 3) Performing delete-min on H.

a) What are the worst case running times of INSERT and DELETE-MIN in the Lazy-min-heap.

b) Prove that in the Lazy-min-heap, INSERT and DELETE-MIN each have an amortized running time of $O(\log n)$.