Problem Set 3

This problem set is officially due at 11:59pm on Monday, March 31, 2014. However, in order to give you some extra time to finish up after spring break, you may submit it up to 11:59pm on Wednesday, April 2, 2014 with no penalty. This extension will not impact the due date for Problem Set 4, which will remain unchanged.

- As always, faster algorithms are better than slower ones. For example, when asked to design an efficient algorithm, a $\Theta(n \log n)$ algorithm is preferred to one with running time $\Theta(n^2)$.

- Both exercises and problems should be solved, but only the problems should be turned in. Exercises are intended to help you master the course material. Even though you should not turn in the exercise solutions, you are responsible for material covered by the exercises and the best way to master that material is to have solved the problems yourself.

- Mark the top of each sheet with (1) your name, (2) the name of your recitation instructor, and the time your recitation section meets, (3) the problem number, (4) the names of any people you worked with on the problem, or “Collaborators: none” if you solved the problem completely alone.

- Each problem must be turned in as a separate PDF file to Stellar.

Exercise 3-1. CLRS Exercise 15.4-6 on page 397.

Exercise 3-2. CLRS Exercise 25.2-7 on page 700.

Exercise 3-3. CLRS Exercise 33.4-1 on page 1043.


Problem 3-1. Flash Mob!

You’ve accepted a summer internship at a startup that’s making a new flash mob app. Here’s how it works: Given $n$ people in the plane, the app will periodically convene a flash mob of size $k$, where $k$ is some constant. The objective is to find a subset of $k$ of the $n$ people such that the sum of the pairwise distances between those $k$ people is as small as possible. In other words, given a set of $n$ points, we wish to find a subset $P$ of exactly $k$ points that minimizes the sum $\sum_{p,q \in P} d(p, q)$, where $d(\cdot, \cdot)$ denotes the Euclidean distance. Describe and analyze a $O(n \log n)$ algorithm for doing this and prove that it is correct.
Problem 3-2. Rectilinear Ghostbusters

First the gratuitous backstory: The Pasadena Institute of Ghostbusting (PIG) is the premier center for training Ghostbusters. In the training program, ghosts and ghostbusters are arranged in the plane and ghostbusters can only shoot horizontally or vertically. Streams will inevitably cross during training and the instructors wish to determine the number of crossings in order to score their cadets.

So, here’s the problem: We’re given a collection of \( n \) axis-parallel (horizontal or vertical) segments in the plane. For simplicity, no two vertical segments overlap, and no two horizontal segments overlap. Describe an algorithm that returns the number of intersections between segments in the collection, explain why it is correct, and analyze its running time and space. For full credit, your algorithm should run in time \( O(n \log n) \) and be space efficient as well. (Notice that the number of intersections may be \( \Theta(n^2) \), but counting the number of intersections is different from listing them!)

Problem 3-3. Wheel Deliver!

Two East Campus students, we’ll call them Adam and Betsy, have started a new unicycle-based meal delivery business called “Wheel Deliver” that works like this: Residents of Cambridge call between 5 and 5:30 PM and order food for delivery. Adam and Betsy go to McCormick Dining Hall, pick up the food, and deliver the food on their unicycles.

Let \( n \) denote the number of customers that call on a given evening and let \( P = (p_1, \ldots, p_n) \) denote the list of locations of these customers (points in the plane), sorted in the order in which the customers called: \( p_i \) is the location of the first caller to place an order and \( p_n \) is the location of the last caller to place an order. Let \( p_0 \) denote the location of McCormick, where Adam and Betsy begin their delivery. Let \( d(p_i, p_j) \) denote the distance between \( p_i \) and \( p_j \).

Adam and Betsy wish to split up the list of locations \( P \) (not necessarily evenly) so that each delivery is made by exactly one of them. In addition, “Wheel Deliver” has a fairness rule that goes like this: If customers at \( p_i \) and \( p_j, i < j \), are assigned to the same unicycle rider then the meal must be delivered to the customer at \( p_i \) before the delivery to the customer at \( p_j \), even if that makes the trip longer. Note that if \( p_i \) and \( p_j, i < j \), are assigned to different riders then we don’t care which of those two get their food first. Subject to these constraints, the objective is to minimize the total distance travelled by the two unicycle riders.

Describe and analyze an efficient algorithm that takes \( P = (p_1, p_2, \ldots, p_n) \) as input and generates the delivery points for Adam and for Betsy. Prove that your algorithm is correct.

Problem 3-4. The Millisoft Party Problem!

You’ve decided to accept a job as senior algorithm designer at Millisoft. One afternoon, Gill Bates comes to you with the following problem. “I’m throwing a company party,” Gill says excitedly, “And I need your help! As you know, Millisoft has a hierarchical structure. You can think of it as a tree. The president, that’s me, is at the root of the tree.” You take a sip of your luke-warm diet coke (which Millisoft provides for free) and listen as Gill continues. “Below the root are supervisors, below them are managers, below them are team leaders, etc., etc., until you get down to the leaves -
the summer interns. The tree is not necessarily binary; some non-leaf nodes may have one “child”, others two, and others even more. Anyhow, to make the party fun, I thought it best that we don’t invite an employee along with their immediate boss (their parent in the tree). Also, I’ve personally assigned every employee a positive number called their coefficient of fun. My objective is to invite employees so as to maximize the total sum of the coefficients of fun of all invited guests, while not inviting an employee with his or her immediate boss.”

Describe an efficient algorithm for this problem, prove its correctness, and analyze its running time.

Problem 3-5. Hurts Car Rental

Hurts Car Rental has designed a new generation of alternative fuel vehicles. The new vehicles use a special fuel comprising a proprietary mixture of the “Algorithms” textbook, chocolate fudge Pop Tarts, Spam, and Mountain Dew. Due to this rather unusual fuel requirement, there are only certain cities in the country where the vehicles can be refueled. Thus, to get from the start city to the destination city, the driver must plan a route that ensures that the car can be refueled along the way.

The on-board computer on such a vehicle contains a weighted directed graph in which the vertices represent cities, the directed edges represent one-way roads, and the weights on the edges represent the (positive integer) lengths of the roads. The graph, of course, can have cycles!

The computer knows which select cities have filling stations. To use the navigation system, the user will enter the starting city, the destination city, and the range of the vehicle on a full tank. (You may assume that the starting city and the destination cities, being Hurts rental cities, both have filling stations - and that the car starts with a full tank of fuel.) The computer will respond with the shortest route from the starting city to the destination city that ensures that the vehicle doesn’t run out of fuel or determines that no such route exists.

Describe an efficient algorithm for determining such a path, analyze its running time, and prove its correctness. If you use existing algorithms that we’ve seen in 6.046 or 6.006, you may use their running times without re-deriving them and you may assume that those algorithms are correct.