Data Structures and Algorithm Design
CSCI 340
Spring 2019
Programming Project 4
Closest Points

Due Date: Tuesday, April 9

Points: 40

Objective:
The student will write a program to efficiently find the closest two points in a set of points.

Background:
Consider the problem of finding the closest pair of points in a set Q of $n \geq 2$ points. "Closest" refers to the usual Euclidean distance: the distance between points p1 = (x1, y1) and p2 = (x2, y2) is sqrt ($(x1-x2)^2 + (y1-y2)^2$). For our purposes, we will assume that all x values and all y values are unique. This simplifies the problem slightly, and means that we will never have a distance = 0.

This problem has applications in, for example, traffic-control systems. A system for controlling air or sea traffic might need to know which the two closest vehicles are in order to detect potential collisions.

A brute force algorithm simply looks at all the possible pairs of points. Since there are $\Theta(n^2)$ pairs, the brute algorithm is $\Theta(n^2)$. Here we describe a divide and conquer algorithm that is $\Theta(n \lg n)$.

The Divide and Conquer Algorithm:
The algorithm works on two arrays (or lists) of points. X is the set of points sorted in increasing x order. Y is the same set of points sorted in increasing y order. Presort the points into these arrays and then make sure that any manipulations keep the points in the appropriate order.

Divide
If X contains 3 or fewer elements, simply calculate the distance between all pairs and return the pair that has the smallest distance. You may save a bit of time on the combine step by also returning the distance between them.

Otherwise,

Divide the X array into two halves, X_L will contain the smaller $\lceil n/2 \rceil$ points based on their x values.

X_R will contain the larger $\lfloor n/2 \rfloor$ points. Note that this division is trivial since no two points share the same x value. We can think of this as drawing a vertical line, *l*, down through a graph of our points such that half the points are on each side.

Divide the Y array into two halves, where Y_L contains the same points as X_L and Y_R contains the same points as X_R. This takes a little bit more work, but simply walk through the Y array and copy the points to either Y_L or Y_R. They should still be in monotonically increasing order based on y values.

Conquer
Make two recursive calls to your algorithm, one passing in X_L and Y_L. The other passing in X_R and Y_R.

Combine
Let *dist* be the minimum of the distances from the two recursive calls. The closest pair of points to return from this level is either the pair that gave us *dist* or a pair with one point on each side of *l*. We still need to determine if any of these latter pairs give us a distance less than *dist*.

Checking pairs that cross *l* is the tricky part of the problem to do efficiently, since it seems like we might still have to check a lot of pairs. There are two keys to limiting the number of pairs. First, realize that any point further than *dist* from *l* can be pruned. Take the Y array and copy it into a new array Y_{prime}, throwing out any points more than *dist* from *l*. This pruning might or might not help us significantly, since all the points might be clustered close to *l*.

The next step is the magical one and is left without proof here. (Your instructor will informally try to convince of its correctness in class). For each point p[i] in Y_{prime} we only need to check its distance to the next 5 points in Y_{prime}. If none of them gives us a distance less than *dist*, points further along Y_{prime} will not. This limits the distance calculations in this step to be linear in the size of Y_{prime}.

Return the pair of points that gives us the minimum distance.

The Assignment:
Write a Java program that finds the closest pair of points in a set Q of points.
The x and y values are doubles.
The points should be read in from a file whose name is specified by the user.
The file contains one X-Y pair per line. The values are separated by white space.
The file is not sorted.
Your instructor has provided some sample files on the lab machines in /home/student/Classes/Cs340/PointFiles.

Your program should report the two closest points and the distance between them.

What to Turn in:
Submit a copy of your source code. Name your class ClosestPoints and the file ClosestPoints.java. Only an electronic submission is needed for this assignment.

Grading Considerations:
The program should produce correct and nicely formatted output.
The program should run efficiently. Programs that take too long, even if the run correctly, will be graded down significantly.
Good software engineering is expected. Use lots of comments, lots of methods, appropriate variable names, etc.