CS268: Geometric Algorithms Handout # 3
Design and Analysis
Stanford University Monday, 10 October 2016

Homework #1: Arrangements, zones, straight and topological sweeps [100 points]
Due Date: Monday, 24 October 2016

This assignment contains a number of theory problems and a programming problem. You are expected to do all the problems.

Doing problems is a very important part of this course. Although you have two weeks to do this assignment, do not delay starting to work on it — several of these problems are not routine exercises. If you cannot solve the problem fully, please write up whatever you can do and document any partial results you have obtained in the process. We intend to be generous with partial credit when your effort is well documented.

You are encouraged to collaborate in study groups of up to three students on the solution of the homeworks. If you do collaborate on theory problems, you must write up solutions on your own and acknowledge your collaborators by name in the write-up for each problem. If you obtain a solution with outside help (e.g., through literature search, another student not in the class, etc.), acknowledge your source, and write up the solution on your own. For programming problems a single write-up per group is acceptable.

It is very important in this course that every homework be turned in on time. We recognize that occasionally there are circumstances beyond one's control that prevent an assignment from being completed before it is due. You will be allowed two classes of grace during the quarter. This means that you can either hand-in two assignments late by one class, or one assignment late by two classes. Any other assignment handed in late will be penalized by 20% for each class that it is late, unless special arrangements have been made previously with the instructor.

For grading we will use Gradescope in this class ((`https://gradescope.com/`). More details will be posted on Piazza soon.

- **The Theory Problems**

Problem 1. [10 points]

In an arrangement $\mathcal{A}$ of n lines in the plane, a single face can have at most n sides. Prove that any m distinct faces can have at most $n + 4\binom{m}{2}$ sides altogether. [[This bound is best possible if $4\binom{m}{2} \leq n$ and is known as *Canham's Lemma*; it implies, for instance, that any $\sqrt{n}$ faces can have a total of only $O(n)$ sides altogether.]]

Problem 2. [15 points]

We saw in class that, in an arrangement $\mathcal{A}$ of n lines in the plane, the zone of another line ℓ has combinatorial complexity $O(n)$ (zone theorem). Given as input only the n lines of the arrangement and ℓ (say by their line equations), show how to compute all the faces of $\mathcal{A}$ comprising the zone of ℓ, in linear space and $O(n \log n)$ time.

Problem 3. [20 points]

Suppose that we modify the Bentley-Ottmann straight-line sweep method for computing a line arrangement so that, when processing an event, the future events corresponding to intersections for newly created adjacencies between active segments are added to the priority queue, but the events corresponding to the adjacencies just destroyed are *not* removed. This will still give a correct algorithm, but now the priority queue size may increase, as each event adds possibly two new adjacencies but removes only one.

(a) Prove that, given any four lines a, b, x, y in descending slope order, not all three intersections ax, ay, by can be events present in the priority queue at once. (Note that this problem is about infinite lines, *not* line segments.)

(b) Use this fact to argue that maximum size of the priority queue is still at most $O(n \log n)$. (Partial credit will be given for any subquadratic upper bound.)

(c) Give an example of a line arrangement family showing that, in the worst case, the maximum size of the priority queue is $\Omega(n \log n)$ — so the above bound is in fact tight.

Problem 4. [15 points]

Show how to implement the topological sweep we discussed in class using only a *single* horizon tree, say the upper horizon tree, and at most a constant amount of additional memory — while still maintaining the $O(n^2)$ running time. (*Hint:* The crucial step is to discover efficiently a vertex of the tree where an elementary step can be carried out.)

- **The Programming Problem**

Problem 5. [40 points]

The goal of this problem is to start building some familiarity with implementing geometric computations.

Download the zip file for the assignment from the Handouts page on the course web site. This zip file contains starter code in Java and README file that explains how to setup Java programming environment. Feel free to contact the CA if you have any problem starting the homework.

Implement your algorithm for solving the earlier Problem 2 (or another algorithm, if you think it is preferable in practice) using the provided starter code. The input of the program is a list $\mathcal{L}$ of n lines $\ell_1, \ell_2, \cdots, \ell_n$, and the output is the zone of the y-axis in the arrangement $\mathcal{A}(\mathcal{L})$, i.e. the collection of all the convex faces of $\mathcal{A}(\mathcal{L})$ crossed by the y-axis. The input lines are specified in text files that are passed in as an argument to the program. Find TODO in *Manager.java* file and add the algorithm code below. The starter code currently reads the input lines and renders them, along with the y-axis; it also renders an array of faces, currently empty, which you will need to populate.

All input sets of lines will be free of degeneracies, i.e. no parallel lines, no horizontal or vertical lines, no three lines intersecting at the same point (and no two lines with an intersection point on the y-axis). Each test case will have at least one input line. Furthermore, all lines will intersect the y-axis in the visible window, which goes from (-300, -300) to (300, 300); you may take advantage of this fact to choose large finite values to represent infinity.

For up to 5 points extra credit, you may augment your code to handle degenerate input.

Submission instructions: Pack all of your source files into a single zip file and email the file to the CA (mhsung@cs.stanford.edu) with the subject line starting with [cs268-hw1]. Include your own README file with the names of all team members, as well as what degenerate cases (if any) you are able to handle.