CS368: Geometric Algorithms Handout #3
Design and Analysis
Stanford University Tuesday, 13 April 1999

Homework #1: Arrangements, zones, straight and topological sweep [70 points]
Due Date: Tuesday, 27 April 1999

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.

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 library work, 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.

Each problem set will consist of three parts, to accommodate the different needs of the students in the theory and applied tracks of the course. The common theory problems are to be worked out by everyone. The additional theory problems should be done by those in the theory track. The programming problem(s) should be implemented by those in the applied track.

- **The Common 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. [10 points]

We saw in class that, in an arrangement $\mathcal{A}$ of n lines in the plane, the horizon or zone of another line ℓ has combinatorial complexity $O(n)$. Given as input only the n lines of the arrangement and ℓ (say by their 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. [10 points]

Show that the topological sweep that computes the arrangement of n lines in the plane can be carried through within the same time and space bounds even if the search for an intersection when propagating a line into a bay of the upper horizon tree is done

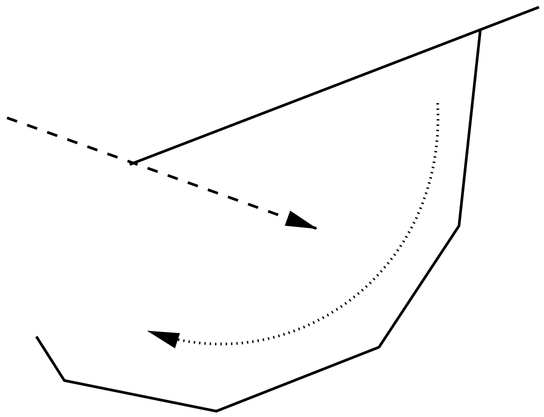

Figure 1: Searching for an intersection clockwise

by traversing the bay *clockwise* instead, starting from the other (shortened) edge of the elementary step. See figure 1.

- **The Additional Theory Problems**

Problem 4. [20 points]

Suppose that we modify the straight-line sweep method for computing a line arrangement so that, when processing an event, the future events corresponding to intersections for the two newly created adjacencies are added to the priority queue, but the events corresponding to the two 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. 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. Use this fact to argue that maximum size of the priority queue now is at most $O(n \log n)$. Give an example showing that this bound is in fact attainable. (Partial credit will be given for any subquadratic upper bound.)

Problem 5. [5 points]

Show that the topological sweep that computes the arrangement of n lines in the plane can be carried through within the same time and space bounds, even when the topological line is required to proceed *vertically* through each region and can only move horizontally by following arrangement edges. In other words, the topological line should consist entirely of vertical segments crossing faces and portions of arrangement edges. This variant can be used to compute the threads needed for a triangulation of the arrangement.

Problem 6. [15 points]

Show how to implement the topological sweep we discussed in class using only a *single* horizon tree, say the upper horizon tree. (*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 7. [**40 points**]

The goal of this problem is to get you familiar with implementing geometric computations and with some of the algorithms packages we are going to be using throughout the course.

Implement your algorithm for solving Problem 2 (or another one, if you think it is preferable in practice) in C++, using the libraries specified below. More specifically, the input of the program is a list $\mathcal{L}$ of n lines $\ell_1, \ell_2, \cdots, \ell_n$ and a query line ℓ which is specified interactively by the user. The output is the zone of ℓ in the arrangement $\mathcal{A}(\mathcal{L})$, i.e. the collection of all the convex faces of $\mathcal{A}(\mathcal{L})$ crossed by ℓ. The input lines can be specified interactively, or their coefficients may be read from a file. The input lines, the query line, and the zone of the query line are to be displayed in a graphics window.

The Programming Environment

Throughout this course, we will use existing C++ libraries to facilitate geometric programming. The libraries are LEDA (*Library for Efficient Data Structures and Algorithms*) and CGAL (*Computational Geometry Algorithms Library*) developed in Europe.

As implied in their names, LEDA includes an implementation of the most common and useful data structures and discrete algorithms, and CGAL includes implementations of primitive geometric objects and related algorithms. While LEDA itself also implements some basic geometric primitives, CGAL is targeted for geometric objects and is much richer than LEDA in terms of geometric computation. In this course, we suggest that you use CGAL to handle geometric objects and LEDA to handle combinatorial structures (and certain number types). You are welcome to use STL (Standard Template Library) for simple data structures, as STL is much lighter than LEDA. More information about these packages can be found at:

`http://www.mpi-sb.mpg.de/LEDA/index.html`
`http://www.cs.uu.nl/CGAL`
`http://www.sgi.com/Technology/STL`

The manuals for LEDA and CGAL are kept locally at:

`http://graphics.stanford.edu/courses/cs368/LEDA`
`http://graphics.stanford.edu/courses/cs368/CGAL`

For this first project, you will only need to read the introduction and relevant sections about straight lines and the graphical interface.

Getting Started

For this project, you are expected to use C++ on Sun SPARC workstations. Since all the libraries to be used are template libraries, you may want to review the use of template classes in C++ in the case you are not familiar with templates. Both LEDA and CGAL have been installed at the class directory at Sun SPARC workstations on the Leland system. A helpful way to learn these libraries is to run their demo programs, which are located at `/usr/class/cs368/LEDA/demo` and `/usr/class/cs368/CGAL/demo`. We also provide an example file for you to start with. You can get that by copying all the files under the directory `/usr/class/cs368/example/start`. This provides a simple example about how to use CGAL and LEDA.

Deliverables

To get full credit for this problem, you need to handle in a two-to-three page write-up of your algorithm and its implementation. In the write-up, you should provide a complexity analysis of your algorithm and a justification for the key implementation decisions you made. The submission information will be given later.

In your program you should allow for at least two ways to input the line set $\mathcal{L}$. One is from a data file, where the format is the number of lines followed by the (a, b, c) coefficients for each line (the line is defined by $ax + by + c = 0$); the other is interactive input directly in the graphics window. Please refer to the sample program for the input formats. The output should also be produced in two ways. One is to paint the zone in the window; the other is to list the convex faces in the zone in the ordering they appear along the query line. For each face, you need to list the lines in the order they appear on the boundary of the face.

Points will be given based on the correctness, efficiency, and clarity of your implementation.