Lehrstuhl für Informatik I
Effiziente Algorithmen und
wissensbasierte Systeme
Universität Würzburg

Würzburg, November 12, 2014

Prof. Dr. Alexander Wolff
Dipl.-Inform. Philipp Kindermann

Homework Assignment #6
Computational Geometry (Winter Term 2014/15)

Exercise 1

Give an example of a set of n line segments with an order on them that makes the algorithm TRAPEZOIDALMAP, without the randomization (line 2), create a search structure of size $\Theta(n^2)$ and query time $\Theta(n)$. **[6 points]**

Exercise 2

Describe a randomized, incremental algorithm that calculates the intersection of a set H of n half-planes in $O(n \log n)$ time. The algorithm first chooses a random permutation $\langle h_1, \ldots, h_n \rangle$ of H. In every step $i = 1, \ldots, n$, it calculates the intersection $I_i := h_1 \cap \cdots \cap h_i$ of the first i half-planes in the permutation. Use the following witness structure to find out where to insert a half-plane: For every half-plane h_j with $j > i$, save a witness. This is a corner of I_i that does not lie in h_j. For the sake of simplicity, assume that $h_1 \cap h_2 \cap h_3$ is bounded.

Hint: First, consider the running time that is required to create and delete the vertices of I_i ($i = 1, \ldots, n$). Then define a random variable X_i for the number of witnesses that change in step i. Use backwards analysis to determine the total running time $\sum_i X_i$ to manage the witness structure. **[14 points]**

Exercise 3

Recall Exercise 3 from Homework Assignment #5.

> Given a set P of n points and a number $\varepsilon > 0$, we want to compute all pairs of points whose L_∞-distance is at most ε, that is, we want to find each pair p, q in P whose x-coordinates differ by at most ε and whose y-coordinates differ by at most ε.
>
> Device a sweep-line algorithm to solve this problem in $O(k + n \log n)$ time, where k is the size of the output.

Implement the sweep-line algorithm in Java. Use the classes provided in WueCampus.

- `Point.java` and `Pair.java` are the objects used for in- and output of the algorithm. You can add new attributes and Getters / Setters to these classes as you like.
- `IpeDraw.java` is a class provided to visualize the output of the algorithm.
- Put your implementation into the file `ClosePairs.java`. The function `Iterable<Pair> closePairs (Iterable<Point> points, double epsilon)` returns each `Pair` of `Point`s whose x- and y-coordinates differ by at most `epsilon`.
- `Test.java` generates a random set of points in the plane and uses your algorithm to find the close pairs. It visualizes your output using ipe (`http://ipe7.sourceforge.net/`).

You are not allowed to use external libraries. Your code, that is, your implementation of the class `ClosePairs` and any newly created classes, should not exceed 100 lines, with at most one command **or** closing bracket per line. Upload all classes within one zip-File to WueCampus.

Hint: Create two comparators for the points: One that orders them by x-, and one that orders them by y-coordinate. Use the class `TreeSet` as implementation of a binary tree. The functions `add`, `contains`, `higher`, `lower`, `remove`, and `subSet` are guaranteed to run in time logarithmic in the number of elements currently in the `TreeSet`.

[10 extrapoints]

This assignment is due at the beginning of the next lecture, that is, on November 19 at 10:15. Solutions will be discussed in the tutorial on Friday, November 21, 14:00–15:30 in room SE I.