CMPS 3120 Computational Geometry – Spring 13

3/19/13

# 4. Homework

Due **Tuesday 4/2/13** before class

1. **Worst-Case DT Runtime (5 points)**
   Give an example that shows that the worst-case runtime of the randomized algorithm to compute the Delaunay triangulation of a set of $n$ points in the plane is $\Omega(n^2)$. *(Hint: It might help to play with one of the Delaunay triangulation programs.)*

2. **Voronoi (10 points)**
   We saw in class that the Voronoi diagram of a set of points in $\mathbb{R}^2$ is the projection of the upper envelope of the dual lifted set of planes in $\mathbb{R}^3$. What does the projection of the *lower* envelope correspond to? Similarly, what does the projection of the *upper* convex hull of the points lifted to $\mathbb{R}^3$ correspond to?

   Answer these questions by researching on the internet; as usual, cite the source you were using and give an explanation in your own words.

3. **Convex Hull of Intersections (10 points)**
   Let $\mathcal{L}$ be a set of $n$ lines in the plane, no two of which are parallel. Let $S$ be the set of all $O(n^2)$ intersection points between any two lines in $\mathcal{L}$.

   (a) Give an $O(n \log n)$ time algorithm to compute an axis-parallel rectangle that contains $S$.

   (b) [**Optional; for extra credit**] Give an $O(n \log n)$ time algorithm that computes $CH(S)$.

   *(Hint: Your algorithms cannot compute all points in $S$ explicitly. Sort all lines by slope, and prove that it is enough to consider only a certain subset of intersection points.)*

4. **Linear Separator (10 points)**
   Let $R = \{r_1, \ldots, r_m\}$ be set of $m$ red points, and let $B = \{b_1, \ldots, b_n\}$ be a set of $n$ blue points in the plane. A line $l$ is called a **linear separator** if all points of $R$ lie on one side of $l$ and all points of $B$ lie on the other side. (You may assume appropriate general position, and may disregard points that lie exactly on the line.)

   Use point-line duality to develop an algorithm for this problem which runs in expected linear time. *(Hint: Linear Programming.)*