CPSC 420+500 Problem Set 2 17 Jan 2017
W. Evans Due: 30 Jan 2017 at 21.00 (9pm)

Submission Instructions

Handin your solutions using `handin`. You can write your solutions by hand and scan the pages or take pictures of them with your phone; or use a word processing package to typeset your solutions. Whatever you do, you should produce **pdf** files **assign2Q1.pdf**, **assign2Q2.pdf**, etc. containing your solutions for Question 1, Question 2, etc.

To handin: Copy your solution files to the directory `~/cs420/a2` in your home directory on a CS undergraduate machine. (You may have to create this directory using `mkdir ~/cs420/a2`.) Then run `handin cs420 a2` from your home directory.

If you do not have a CS department account, you can email your assign1.pdf to me `will@cs.ubc.ca`.

Late submissions are not accepted.

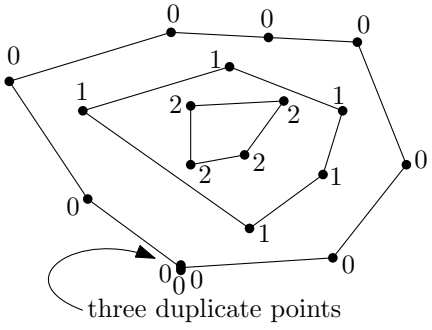
Grading Policy

We will grade a subset of these questions of size at least two. It's a good idea to do all of the questions because (1) you don't know which ones we'll grade and (2) answering these questions is good practice for the exams. We will email feedback to the email address associated with the account you used to handin the assignment.

Questions

Try to answer these on your own but if you work with someone or use an outside source you must acknowledge them in your write-up.

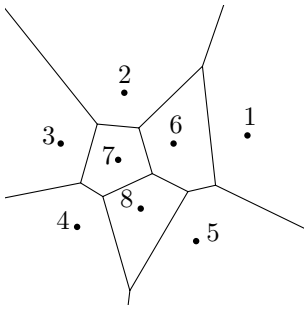
1. The *depth* of a point p in a multi-set of points P in the plane (there might be duplicate points) is 0 if p is on the boundary of the convex hull of P, otherwise the depth of p is one plus the depth of p in the set P without the points on the boundary of the convex hull of P. Point depth is indicated in the figure.

   Prove that any algorithm that calculates the maximum depth of a point in P takes $\Omega(n \log n)$ time, where $n = |P|$, in the algebraic decision tree model of computation.

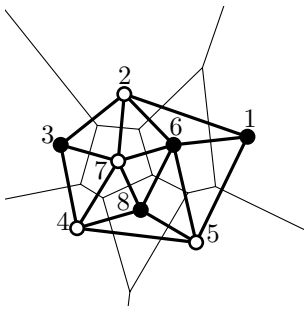
   You may use the fact that solving Element Uniqueness takes $\Omega(n \log n)$ time in the algebraic decision tree model.

2. **Voronoi diagram.** Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of n points in the plane. A Voronoi diagram partitions the plane into Voronoi regions $R_1, R_2, \ldots, R_n$ where R_i is the set of points in the plane whose closest site is s_i, that is, $R_i = \{p \mid d(p, s_i) \leq d(p, s_j) \text{ for all } j\}$. Here's an example:



Prove that every Voronoi region is convex. Remember that a set R is convex if for any two points a and b in R, the line segment connecting a and b is in R; and the intersection of convex sets is convex.

3. **Delaunay Triangulation** The Delaunay triangulation of a set S of points in the plane is a partition of the convex hull of S into triangles whose vertices are the points of S such that the circumcircle of any triangle in the partition contains no points of S in its interior. The following figure shows the Delaunay triangulation, with bold lines, and the Voronoi diagram of a set of points. As you can see, the Delaunay triangulation is closely related to the Voronoi diagram.



Suppose the points are colored white and black. Prove that the closest pair of points of different colors in S is an edge of a triangle in the Delaunay triangulation. (This should be a very short proof. You may use the fact that $\overline{pq}$ for $p, q \in S$ is an edge of a Delaunay triangle for S if and only if there exists a circle through p and q that contains no points in S.)

4. (from Exercise 5.4.5.2 from "Computational Geometry in C" by J. O'Rourke) **One-dimensional Voronoi diagrams.** Let $S = s_1 < s_2 < \cdots < s_n$ be a sequence of real numbers. We think of these as Voronoi sites on the x-axis. A *one-dimensional Voronoi diagram* of S is a sequence of real numbers $(m_1, m_2, \ldots, m_{n-1})$ such that m_i is the midpoint of $s_i s_{i+1}$.

   Suppose you are given a sequence $M = (m_1, m_2, \ldots, m_{n-1})$ of real numbers. Describe an efficient algorithm that finds a sequence of n *sites* for which M is the Voronoi diagram, or reports that there is no such sequence.

5. **Kuba's March** is an algorithm to calculate the convex hull of a set of points P in the plane. Here it is:

   KubasMarch(P)
   1. Let p_1 and p_2 be the points with minimum and maximum y-coordinate.
   2. Let S be an empty stack and Q be an empty queue.

3. Push segment (p_2, p_1) and segment (p_1, p_2) onto S.
4. While S is not empty
5. Pop segment (p, q) from S.
6. Find the point r furthest to the right of the directed line $\overrightarrow{pq}$ through p to q.
   (In case of ties, pick the one that is closest to p.)
7. If no such r lies strictly to the right of $\overrightarrow{pq}$ then enqueue p into Q.
8. else push segment (r, q) and segment (p, r) onto S.
9. Output Q.

(a) Explain why Kuba's March takes $O(nh)$ time where $n = |P|$ and h is the number of points on the convex hull of P.

(b) As in Chan's algorithm, describe how to find the convex hull of $\lceil n/h \rceil$ convex hulls, each of size at most h using Graham's Scan and Kuba's March in $O(n \log h)$ time.

6. Consider the set Z of inputs $x_1, x_2, \ldots, x_n$ (where $x_i \in \mathbb{R}$ for all i) that are NO-inputs of Element Uniqueness (i.e., $x_i = x_j$ for some $i \neq j$). How many connected components does Z have? Explain why.

7. Let $I = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be a set of n intervals on the positive real line: $0 \leq x_i \leq y_i$. Interval (x_i, y_i) is contained in interval (x_j, y_j) if $x_j \leq x_i$ and $y_i \leq y_j$. We are interested in calculating the number of pairs of intervals in I such that one interval is contained in the other.

   (a) Describe an algorithm that solves this problem in $O(n \log n)$ time in the linear decision tree model of computation. Hint: There is a divide and conquer solution when the intervals are viewed as points in the plane. (Please use English and pseudo-code to describe your algorithm. You do *not* need to describe the algorithm using a linear decision tree. In fact, a linear decision tree only works for a particular input size, while your algorithm should work for any input size.)

   (b) Prove that any algorithm (in the linear decision tree model) that solves this problem takes $\Omega(n \log n)$ time. Hint: Use a reduction.