CS226: Computational Geometry
Homework 1

Feel free to discuss these questions with your classmates or friends, but please write up all of the solutions in your own words. Include a list of people you got ideas from, and also a list of Web sites, books, papers, etc. that you used in solving each problem.
Always explain your reasoning using complete sentences, punctuation, and paragraphs. Unless your handwriting is very, very readable, please type. Using LaTeX would be best, and also good practice. Each part of each problem should be answered in at most a page; longer answers are not better than short, clear answers.
If you cannot completely solve a problem but you can say something relevant, write it down.

1. The *farthest-point Voronoi diagram* of a set of n input point samples P in $\mathbb{R}^d$ is a decomposition of space into regions, in which all the points in each region share the same *farthest* sample point in P. Use the lifting to the paraboloid transformation to show that:
   a) all of the cells of the farthest-point Voronoi diagram are convex polytopes, and
   b) it is possible to construct the farthest-point Voronoi diagram of P in time $O(n^{\lceil d/2 \rceil})$.

2. This problem is motivated by examples in machine learning. We are given, as input, a set of n data points in $\mathbb{R}^d$. Each point is labeled either red or blue. We cannot see the labels; our job is to predict them. This would be impossible without additional information, of course. The additional information we get is that we are told that the red points can be separated from the blue points by a hyperplane, and we are shown the labels of a random sample of r points, where r is a constant; we can assume r grows with the dimension, for instance $r = O(d^2)$.
   a) Describe how to construct a hyperplane separating the red sample points from the blue sample points. Give an upper bound on the complexity of this problem as a function of r and d.
   b) If we use this hyperplane to predict the colors of the remaining $n - r$ points, give an upper bound on the expected error of our predictions.

3. Agglomerative clustering is a class of algorithms in which we are given as input a set of points in d-dimensional Euclidean space, and some kind of graph connecting them, with weights on the edge lengths. Initially each input vertex is in its own cluster. At each step, we connect two clusters (if they are not connected already) using the smallest-weight edge in the given graph.

   a) Show that the complete graph, the edges of the Delaunay triangulation, and the minimum spanning tree all produce the same clustering. Write down a proof for each step of your argument, in your own words; do not simply cite theorems you have found or been told, but do reference your sources. (under two pages)
   b) In the Gabriel graph, v_i, v_j are connected by an edge if the ball whose diameter is the segment v_i, v_j contains no other vertices. Does this produce the same clustering as

the graphs above?
c) Give as efficient an algorithm as you can to computer the Gabriel graph from the Delauany triangulation in dimension d.

4. Let's consider the problem of determining whether a query point lies in a complicated polygon. Let's assume it is a *simple* polygon, that is, its boundary consists of a single cycle of edges, with no self-intersections. We assume the polygon is given as a list of n vertices, in counter-clockwise order on the boundary. One possible data structure we could use to answer point-in-polygon queries is a BSP tree.
a) Argue that it might still require $O(n)$ time, in the worst case, to determine whether a point is in the polygon, if the BSP tree is an auto-partition.

   So let us consider, instead, using vertical lines through each of the vertices as potential splitting lines, and using "free splits" as described in Chapter 12.
b) Prove that the expected time to check whether a point is in the polygon using this BSP tree as a search tree should be $O(lgn)$, when the vertical splitting lines are used in random order.
c) Prove the best bound you can on the expected size of this BSP tree.