CS6050/7050 Computational Geometry

Spring Semester, 2018

Assignment 1: Convex Hulls

Due Day: 8:00 p.m. Wednesday, January 31, 2018 (at the beginning of the class)

1. **(15 points)** Recall that a subset S of the plane is a *convex set* if for any two points p and q in S, the line segment $\overline{pq}$ is also in S.

   Prove that the intersection of any two convex sets is still convex. This implies that the intersection of any finite number of convex sets is convex as well.

2. **(20 points)** Let P be a set of n points in the plane. Let p_0 be the leftmost point of P, i.e., the point with the smallest x-coordinate. Assume all other points of P have been sorted by polar angle in clockwise order around p_0, and the sorted list is given in the input. You may assume the sorted list is $p_1, p_2, \ldots, p_{n-1}$ (e.g., see Fig. 1).

   With p_0 and the above given sorted list, design an $O(n)$ time algorithm to compute the convex hull of P.

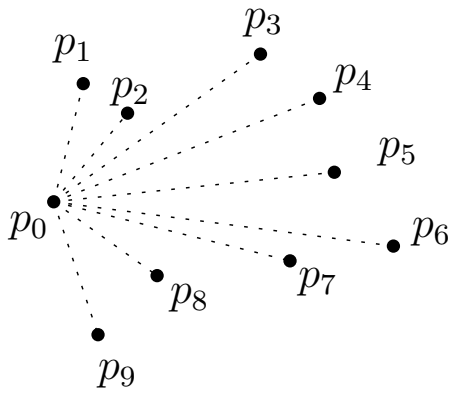

Figure 1: An example for Problem 2

3. **(15 points)** Let P be a set of points in the plane. Prove that the pair of points in P farthest from each other must be two vertices of the convex hull of P.

4. **(20 points)** Given k convex polygons, each represented in a standard way (e.g., by a list storing all vertices in the clockwise order), the total number of vertices of all these k convex polygons is n. Design an $O(n \log k)$ (not $O(n \log n)$) time algorithm to compute the convex hull of all these k convex polygons.

   Note: An $O(n \log k)$ time algorithm would be better than an $O(n \log n)$ time one when k is sufficiently small. For example, if $k = \log n$, then $n \log k = n \log \log n$, which is absolutely smaller than $n \log n$ (i.e., if you know the small-o notation, we have $n \log \log n = o(n \log n)$).
   (continue on the next page)

Note: I would like to emphasize the following, which applies to the algorithm design questions in all assignments of this course.

1. **Algorithm Description** You are required to clearly describe the main idea of your algorithm.
2. **Pseudocode** Although it is not required, I usually find pseudocode very helpful to explain the algorithm. So you are strongly encouraged to provide pseudocode for your algorithm as well. (The reason I want to see the algorithm description instead of only the code or pseudocode is that it would be difficult to understand another person's code without any explanation.)
3. **Correctness** If you feel that the correctness of your algorithm is not that obvious, please explain why your algorithm is correct.
4. **Time Analysis** Please make sure that you analyze the running time of your algorithm.

Total Points: 70