COT5520/CIS4930: COMPUTATIONAL GEOMETRY

Homework # 1

Due date: Feb 2, 2026, Monday

Your solutions should be concise, but complete, and typed (or handwritten clearly). Feel free to consult textbooks, journal and conference papers and also each other, but write the solutions yourself and cite your sources. Answer **only five** of the following six questions. Each problem is worth 20 pts.

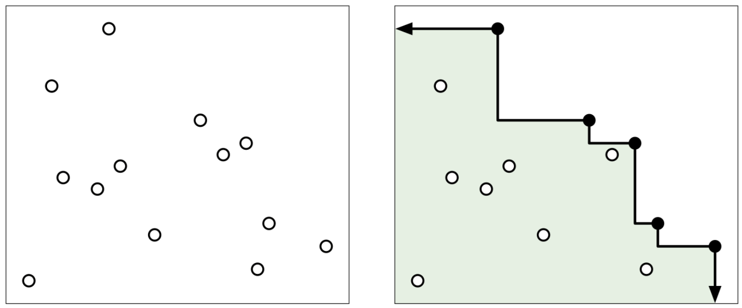
1. Implement at least three of the convex hull algorithms discussed in class using a programming language of your choice. Test your algorithms on self-generated random data sets. Summarize your observations and experiments (ease of implementation, computation times, etc.). You may use available libraries.

2. The region between two parallel lines is called a *slab*. The *width* of a point set P, denoted by $w(P)$ is the width of the smallest slab that contains all the points in P. First, prove that $w(P) = w(\mathit{ConvexHull}(P))$. For a point set P, given the $\mathit{ConvexHull}(P)$ in counterclockwise ordered representation, describe how to determine $w(P)$ in linear time.

3. A *simple polygon* is a region enclosed by a single chain of edges that does not intersect itself. Recall Graham's scan algorithm discussed in class. First, we compute a star-shaped simple polygon of the point set. Then we use a walking scheme with three markers (top three elements of the stack) to compute the convex hull of the star-shaped simple polygon. (i) Describe a simple polygon (which is not star-shaped) for which this second stage fails to produce the convex hull. (ii) Describe an algorithm to construct the convex hull of any simple polygon in linear time. [Hint: You should be able to do the part (i) by yourself. However, part (ii) of the question is not that easy. There were false attempts published in the literature. You might want to do a literature review to have a good understanding of the issues here.]

4. Recall Chan's algorithm described in class. Given a set P of n points we first partition P into n/h subsets each of size h, where h is the size of $\mathit{ConvexHull}(P)$. Then, we compute the convex hull of each subset in $O(h \log h)$ time. Finally we use a Jarvis-type wrapping scheme which employs $O(\log h)$ time queries on one dimensional balanced binary search trees for each subset. The complexity of this wrapping step is $O(n \log h)$. Describe an alternative wrapping scheme that takes only $O(n)$ time. [Hint: You might want to use the notion of amortized complexity analysis.]

5. Let P be a set of points in the plane. A point p in P is *extremal* if no other point in P is both above and to the right of p. The extremal points can be connected by horizontal and vertical lines into the *staircase* of P, with an extremal point at the top right corner of every step (see the figure).



   (a) Describe a divide-and-conquer algorithm to compute the staircase of a given set of n points in the plane in $O(n \log n)$ time.

   (b) Describe an algorithm to compute the staircase of a given set of n points in the plane, sorted in left to right order, in $O(n)$ time.

   (c) Describe an algorithm to compute the staircase of a given set of n points in the plane in $O(nh)$ time, where h is the number of extremal points.

   (d) Describe an algorithm to compute the staircase of a given set of n points in the plane in $O(n \log h)$ time, where h is the number of extremal points.

6. The *staircase layers* of a point set are defined by repeatedly computing the staircase (as described in the previous question) and removing the extremal points from the set, until the set becomes empty.

   (a) Describe and analyze an algorithm to compute the staircase layers of a given set of n points in $O(n \log n)$ time.

   (b) An increasing subsequence of a point set P is a sequence of points in P such that each point is above and to the right of its predecessor in the sequence. Describe and analyze an algorithm to compute the *longest increasing subsequence* of a given set of n points in the plane in $O(n \log n)$ time. [Hint: It is easy to design an algorithm based on part (a), however make sure to prove its correctness.]