CS 498 TC, Fall 2024
Timothy Chan

Homework 3 (due Oct 30 Wed 5pm)

Instructions: See previous homework.

1. [*35 pts*]

   (a) [*15 pts*] Given a set H of n halfplanes in 2D, we want to find the largest-perimeter axis-aligned rectangle inside the intersection of H. ("Axis-aligned" means that the sides of the rectangle are parallel to the x- and y-axes.) Show that this problem can be solved in linear time.

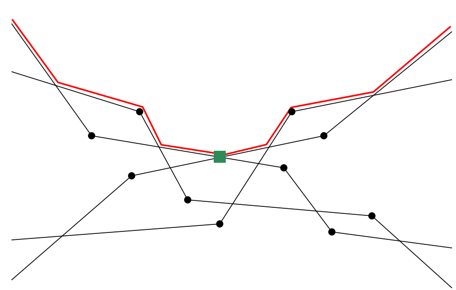
   Note: you may use a known linear-time algorithm for linear programming in any constant dimension.

   (b) [*20 pts*] Given a set H of n halfplanes in 2D, we want to find the largest-perimeter *parallelogram* inside the intersection of H. (The parallelogram need not axis-aligned, and angles need not be $\pi/2$.) Show that this problem can be solved in cubic time.

   Note: you may use a known $O(n^{\lfloor d/2 \rfloor})$-time algorithm for computing the intersection of n halfspaces in $\mathbb{R}^d$. You may also assume that such an algorithm can decompose the intersection into $O(n^{\lfloor d/2 \rfloor})$ cells each with $O(1)$ vertices.

2. [*65 pts*] An *increasing* chain of size k is a (not necessarily convex) polygonal curve $p_0p_1p_2 \cdots p_k$ such that $0 = p_0.x < p_1.x < p_2.x < \cdots < p_k.x = 1$ and $p_0.y < p_1.y < p_2.y < \cdots < p_k.y$. A *decreasing* chain is a polygonal curve $q_0q_1q_2 \cdots q_k$ such that $0 = q_0.x < q_1.x < q_2.x < \cdots < q_k.x = 1$ and $q_0.y > q_1.y > q_2.y > \cdots > q_k.y$. Given n increasing chains and decreasing chains of size k, we want to find the lowest point on their upper envelope.

   (See the figure below; the upper envelope is shown in red, and its lowest point is shown in green. Note that two increasing chains may intersect a large number of times, but an increasing and a decreasing chain may intersect at most once. You may assume that the intersection between an increasing and a decreasing chain can be computed in $O(\log k)$ time by binary search.)



(a) [*40 pts*] Give a deterministic algorithm with worst-case running time $O(n \log^2 k \log n)$ or better, by modifying Megiddo/Dyer's LP algorithm.

(Hint: consider the middle x-value of each curve, and take the median x_m of these middle x-values. How can we decide whether the solution is to the left or right of x_m? How many iterations are needed to reduce the size of *every* curve by a half?)

(b) [*25 pts*] Give a randomized algorithm with expected running time $O(n \log k)$, by modifying Seidel's randomized LP algorithm.