CS 498 TC ✧ Computational Geometry ✧ Spring 2022

Final Exam

Due Wednesday, May 11, 2022 at 4:30pm

✧ Important instructions — Please read carefully! ✧

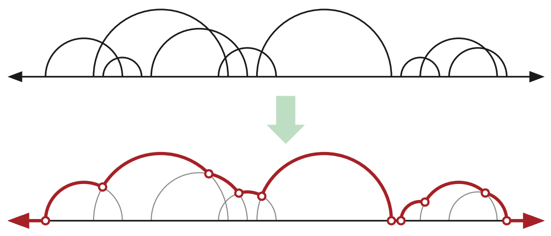
- **Don't panic!**
- **You have at most 180 minutes (that is, 3 hours) to complete this exam and upload your solutions.** The clock started when you opened the Gradescope assignment. All solutions must be uploaded before 4:30pm (Central Daylight Time) on Wednesday, May 11.
- The exam is designed to be completed on paper in 90 minutes. There are four problems.
- **This is an open-book-but-closed-everything-else exam.** You are welcome to use any materials linked directly from the course web site (textbook, notes, lecture videos/scribbles, homework solutions) and anything you wrote yourself before starting the exam. All other resources, including materials from other classes and/or previous semesters, other textbooks, other internet resources, and (most importantly) other people, are not permitted.
- If you use a standard algorithm or data structure from this class, or from any prerequisite class (such as CS 374, CS 361, CS 225, CS 173, or CS 124) as a black box, ***please do not*** give a detailed description; just name the algorithm or data structure. (For example: "balanced binary search tree" or "Gaussian elimination" or "Chan's algorithm" or "Euler's formula" or "the plane-sweep algorithm to count segment intersections".)
- Similarly, if you use a minor variant of an algorithm from this class or a prerequisite class, please describe only the necessary modifications.
- You can implicitly assume general position without comment.
- **I cannot answer questions about the exam while the exam is in progress.** If you need to make additional assumptions to solve an exam problem, please state those assumptions clearly in your solution. Do not discuss the exam with anyone until after Tuesday at 8pm.
- The exam is not proctored. I am trusting you to take the exam by yourself, with no help from anyone, within the declared time limits. The exam is first and foremost a mechanism to give you honest feedback on your mastery of the course material; please treat it as such. All academic integrity policies are still in place.
- Thanks for a great semester!

---

1. Inspired by your sketches of the Manhattan skyline, the residents of Elon Musk's rapidly growing Martian settlement ask you develop an algorithm to sketch *their* skyline. The Musketeers, as they are obviously called,[1] live in geodesic domes of various sizes.

   We are given an array $D[1..n]$ of upper semicircles ("domes") centered on the x-axis. Each dome is represented by its center x-coordinate $D[i].x$ and its radius $D[i].r$. The *skyline* of D is the boundary of the set of points that are above the x-axis and outside every dome in D.

   Describe and analyze an efficient algorithm to compute the skyline of D. Assume you can compute the exact point where two domes intersect, or correctly determine that they do not intersect, in constant time.



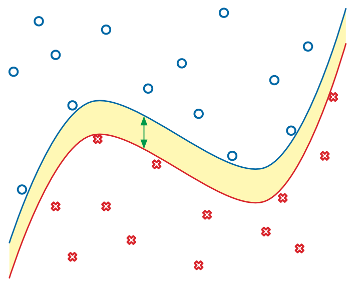
2. Recall that a cubic polynomial is any function of the form $f(x) = ax^2 + bx^2 + cx + d$, where a, b, c, d are real numbers. A *cubic slab* is the region of the plane bounded between two cubic polynomials that differ only in their constant coefficient:

$$\left\{(x,y) \,\middle|\, ax^3 + bx^2 + cx + d^- \le y \le ax^3 + bx^2 + cx + d^+\right\}$$

   The *width* of a cubic slab is the vertical distance $d^+ - d^-$ between its two bounding curves.

   (a) Describe and analyze an algorithm to find the *minimum*-width cubic slab that contains a given set of points in the plane.

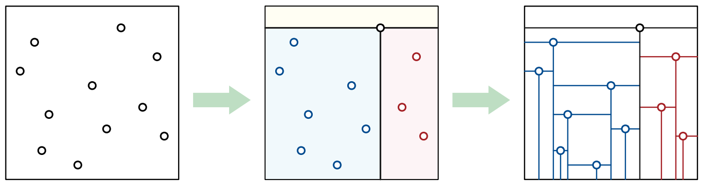
   (b) Now suppose we are given a set R of red points and a set B of blue points in the plane. Describe and analyze an algorithm that either computes the *maximum*-width cubic slab that separates the red points from the blue points, or correctly reports that there is no such slab.



[1] Elon's insistence on "Musken Raiders" has largely been ignored, except by Disney's lawyers.

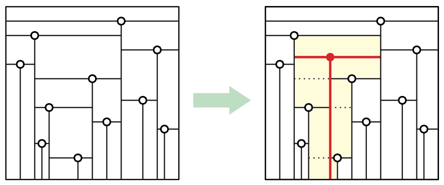
3. Let P be a set of n points with distinct x- and y-coordinates in the interior of some bounding rectangle R. The *Cartesian subdivision* of P is defined recursively as follows.

   - If $P = \varnothing$, the subdivision is just the rectangle R.
   - Otherwise, let p be the highest point in P.
     - First split R into two smaller rectangles R^+ and R^- using a horizontal segment through p.
     - Then split the lower rectangle R^- into two smaller rectangles $R_<$ and $R_>$ using a vertical segment downward from p.
     - Finally, recursively construct Cartesian subdivisions of the points inside the two lower rectangles $R_<$ and $R_>$.


Recursively defining the Cartesian subdivision.

   (a) What is the *exact* number of vertices, edges, and faces in a Cartesian subdivision of n points?

   (b) Prove that if n points are inserted *in random order* into an initially empty Cartesian subdivision, then the expected number of structural changes for each insertion is $O(1)$.


Inserting a point into a Cartesian subdivision. Modified faces of the subdivision are highlighted.

4. Let P be a set of n points in the plane. The *nearest neighbor graph* of P is a directed graph, which contains an edge from one point p to another point q if and only if q is p's nearest neighbor in P.

   (a) Prove that for every directed edge $p{\rightarrow}q$ in the nearest-neighbor graph of P, the corresponding segment pq is an edge of the Delaunay triangulation of P.

   (b) Describe and analyze an algorithm to compute the nearest neighbor graph of a given set of points. *[Hint: Use part (a).]*