CS 498 TC ✧ Computational Geometry ✧ Spring 2022

Midterm

Due Tuesday, March 1, 2022 at 8pm

✧ Important instructions — Please read carefully! ✧

- **Don't panic!**
- **You have at most 150 minutes (that is, 2½ hours) to complete this exam and upload your solutions.** The clock started when you opened the Gradescope assignment. All solutions must be uploaded before 8pm (Central Standard Time) on Tuesday, March 1.
- 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. You are welcome to use randomized algorithms, but none of the problems in this exam require them.
- **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.
- Soon may the Wellerman come, to bring us sugar and tea and rum. One day, when the tonguin' is done, we'll take our leave and go.

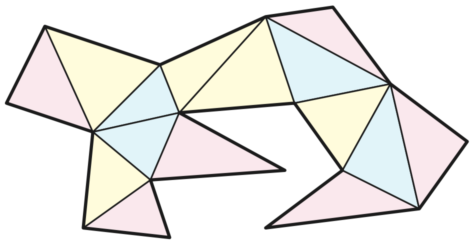
---

1. Let P be an arbitrary simple polygon with $n \geq 4$ vertices, and let T be an arbitrary frugal triangulation of P. Two triangles in T are *adjacent* if they share an edge. The *degree* of a triangle $\triangle$ is the number of other triangles in T that adjacent to $\triangle$. Finally, for any integer d, let t_d denote the number of triangles with degree d.

   (a) Prove that $t_1 + t_2 + t_3 = n - 2$.

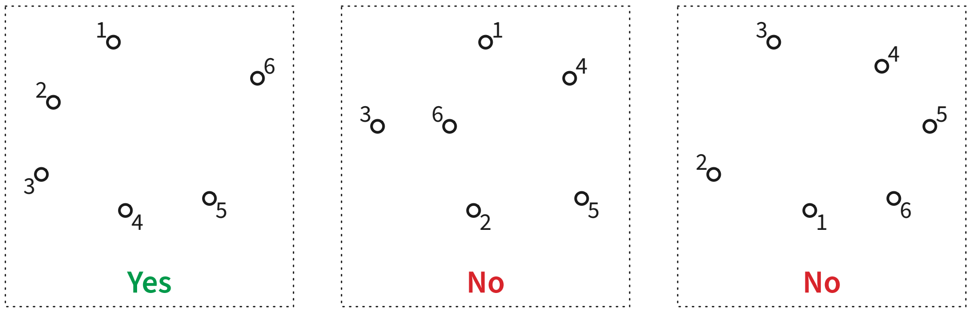
   (b) Prove that $t_1 - t_3 = 2$.

   [Hint: You can use one of these results to prove the other. Consider the dual tree.]


A polygon triangulation with $(t_1, t_2, t_3) = (6, 5, 4)$ and $n = 17$.

2. As an instructor in a computational geometry course, you have asked your students to implement one of the textbook algorithms to compute two-dimensional convex hulls. Now you need to write an auto-grader to check whether their code is correct. As a first pass, you want to confirm that their code outputs convex polygons in the expected format.

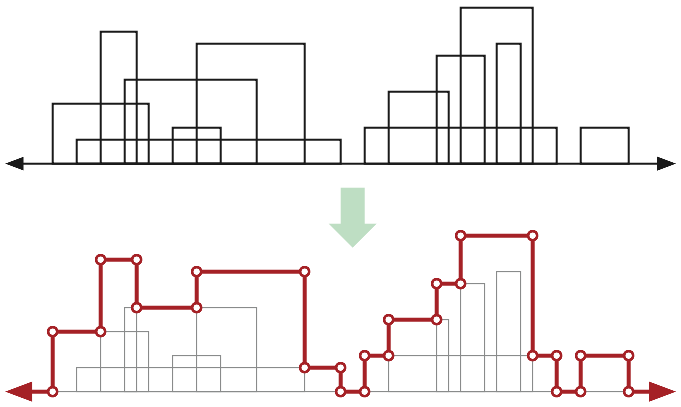
   Suppose you are given an array $H[1..n]$ of points, each with coordinates $H[i].x$ and $H[i].y$. Describe and analyze an algorithm to determine whether H is the counterclockwise sequence of vertices of a convex polygon. For full credit, your algorithm should run in linear time.



3. Suppose we want to sketch the Manhattan skyline (minus the interesting bits like the Empire State and Chrysler buildings).

   We are given an array $R[1..n]$ of rectangles (“buildings”) with their bottom edges on the x-axis. Each rectangle is represented by its left x-coordinate $R[i].l$, its right x-coordinate $R[i].r$, and its height $R[i].h$. The *skyline* of R is the boundary of the set of points that are above the x-axis and outside every rectangle in R.

   Describe and analyze an efficient algorithm to compute the skyline of R.



4. Suppose you are given two convex polygons P and Q, represented as usual by arrays of vertices in counterclockwise order. Describe an algorithm to determine whether P lies entirely inside Q. For full credit, your algorithm should run in linear time.