Homework Policies

• 3-credit students should submit solutions for at least three problems in each homework set. (Only your $3k$ highest problem scores count, where $k$ is the actual number of assignments.)

• 4-credit students should submit solutions for at least four problems in each homework set. (Only your $4k$ highest problem scores count, where $k$ is the actual number of assignments.)

• You are welcome to use any resource at your disposal to help you solve homework problems. You do not need to cite this semester’s course materials (linked from the schedule page), or sources for material typically covered in prerequisite classes (173, 225, and 374). Otherwise, please cite every source that you use.

• Teams of up to three students can submit joint homework solutions. You are welcome to work on the homework in larger groups, but each team must write up solutions in their own words, giving proper credit to any other students (or groups) that helped them. Please list all team members at the top of the first page of each submitted solution.

• Submit your solutions on Gradescope, following the link on the course web page, as separate PDF files for each numbered problem. Exactly one member of each homework team should upload the team’s solution and identify the other team members.

• Whenever a homework or exam problem asks for an algorithm, your solution should include the following:
  – If necessary, a concise specification of the problem your algorithm actually solves. (This could be more general than the actual homework/exam problem.)
  – If necessary, a concise description of the input and output representations.
  – A clear description of the actual algorithm, preferably in pseudocode.
  – A brief justification of your algorithm’s correctness.
  – A brief analysis of the algorithm’s running time.

• Unless the problem explicitly states otherwise, you can implicitly assume that inputs are always in general position. Similarly, if a problem asks for a certain running time, a randomized algorithm with that expected running time is fine.

• Homework and exam scores will depend not only on correctness and speed, but also on clarity and style. Your solutions should be complete and mathematically precise, but at the same time, easy to understand and concise. For most numbered homework problems, a complete solution should fit on a single typeset page. Don’t submit your first draft!
1. Let $P$ be a set of points in the plane. A point $p \in P$ is Pareto-optimal if no other point in $P$ is both above and to the right of $p$. The Pareto-optimal points can be connected by horizontal and vertical lines into the staircase of $P$, with a Pareto-optimal point at the top right corner of every step.

\[
\text{The staircase of a set of points in the plane.}
\]

Now suppose you are given a set $P$ of $n$ points in the plane, with distinct $x$- and $y$-coordinates.

(a) Describe an algorithm to compute the the staircase of $P$ in $O(n \log h)$ time, where $h$ is the number of Pareto-optimal points in $P$. (For partial credit, describe an algorithm that runs in $O(nh)$ time.)

(b) Now suppose the points in $P$ are sorted by their $x$-coordinates. Describe an algorithm to compute the the staircase of $P$ in $O(n)$ time.

2. Suppose we are given a simple polygon $P$, and we want to compute its convex hull. Of course we can ignore the order of the polygon vertices and compute their convex hull in $O(n \log n)$ time, but in fact, this problem can be solved in only $O(n)$ time.

\[
\text{(a) Prove that the three-penny algorithm from Graham's Scan — greedily remove reflex vertices — does not necessarily compute the convex hull of $P$. To avoid trivial counterexamples, assume $P$ is given as an array of vertices in counterclockwise order.}
\]

\[
\text{(b) Describe a simple $O(n)$-time algorithm to test whether the result of running the three-penny algorithm on $P$ is actually convex.}
\]

\[
\text{(c) So why does Graham's scan actually work?}
\]

\[
\text{*(d) Extra credit: Describe an $O(n)$-time algorithm to compute the convex hull of an arbitrary simple polygon. [Hint: This is not straightforward; several published algorithms are incorrect!]}\]
3. Suppose you are given two arrays $X[1..n]$ and $Y[1..n]$ of real numbers, each of which could be positive, negative, or zero. Describe an algorithm to find array indices $i$ and $j$ that maximize the expression $X[i] \cdot Y[j] + X[j] \cdot Y[i]$. The obvious brute-force algorithm runs in $O(n^2)$ time; for full credit, your algorithm should run in $O(n \log n)$ time.

4. Suppose you are given a non-empty set $S$ of disjoint line segments in the plane, along with a single point $p$ that is disjoint from $S$. We can classify the segments in $S$ as follows:

- A segment $s \in S$ is **fully visible** from $p$ if for every point $q \in s$, the line segment $pq$ does not intersect any other segment in $S$.
- A segment $s \in S$ is **partially visible** from $p$ if there is at least one point $q \in s$, such that the line segment $pq$ does not intersect any other segment in $S$.
- Finally, a segment $s \in S$ is **invisible** from $p$ if it is not even partially visible.

![Classifying segments as fully visible (blue), partially visible (red), and invisible (gray) from a point.](image)

(a) Prove or disprove the following claim: At least one segment $s \in S$ must be fully visible from $p$.

(b) Describe an efficient algorithm that labels each segment in $S$ as either fully visible, partially but not fully visible, or invisible from $p$.

5. Two polygons in the plane are **congruent** if and only if one can be transformed into the other by some combination of translation, rotation, and reflection (but no scaling or other transformations).

(a) Let $L$ denote the orthogonal polygon shown below; numbers indicate edge lengths. Describe a partition of $L$ into two congruent simple polygons.

![Orthogonal polygon with edge lengths](image)
(b) For any positive real numbers $a$, $b$, $c$, let $L(a, b, c)$ denote the orthogonal $L$-shaped polygon shown below. Give a complete description of the set of all triples $(a, b, c) \in \mathbb{R}^3_+$ such that the polygon $L(a, b, c)$ be partitioned into two congruent simple polygons using the same pattern that you used to solve part (a).\(^1\) Sketch a proof that your description is correct.

\[ a+c \]
\[ a+b \]
\[ a \]
\[ c \]
\[ b \]

(c) Describe and analyze an algorithm to determine, given three positive integers $a$, $b$, and $c$, whether the polygon $L(a, b, c)$ be partitioned into two simple congruent polygons using the same pattern that you used to solve part (a). Let $N$ denote the number of input bits; analyze your algorithm in terms of $N$.

(d) Can you still describe an algorithm that solves this problem when the inputs $a$, $b$, and $c$ are arbitrary positive real numbers? Why or why not? If so, what is the running time of your algorithm as a function of the input size? [Hint: Careful!]

---

\(^1\)A complete analysis of all ways to partition an arbitrary simple polygon into two congruent pieces requires tedious case analysis. For this particular family of polygons, most of the cases turn out to be impossible or trivial, but enumerating the cases is still tedious. I want you to focus just on one case.