CS 373: Combinatorial Algorithms, Fall 2000
Homework 1 (due November 16, 2000 at midnight)

Name:		
Net ID:	Alias:	U ¾ 1

Name:		
Net ID:	Alias:	U ¾ 1

Name:		
Net ID:	Alias:	U ¾ 1

Starting with Homework 1, homeworks may be done in teams of up to three people. Each team turns in just one solution, and every member of a team gets the same grade. Since 1-unit graduate students are required to solve problems that are worth extra credit for other students, **1-unit grad students may not be on the same team as 3/4-unit grad students or undergraduates.**

Neatly print your name(s), NetID(s), and the alias(es) you used for Homework 0 in the boxes above. Please also tell us whether you are an undergraduate, 3/4-unit grad student, or 1-unit grad student by circling U, ¾, or 1, respectively. Staple this sheet to the top of your homework.

Required Problems

1. Give an $O(n^2 \log n)$ algorithm to determine whether any three points of a set of n points are collinear. Assume two dimensions and *exact* arithmetic.

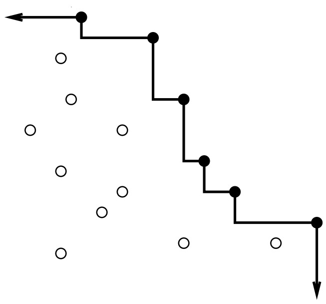
2. We are given an array of n bits, and we want to determine if it contains two consecutive 1 bits. Obviously, we can check every bit, but is this always necessary?

   (a) (4 pts) Show that when $n \bmod 3 = 0$ or 2, we must examine every bit in the array. that is, give an adversary strategy that forces any algorithm to examine every bit when $n = 2, 3, 5, 6, 8, 9, \ldots$.

   (b) (4 pts) Show that when $n = 3k+1$, we only have to examine $n-1$ bits. That is, describe an algorithm that finds two consecutive 1s or correctly reports that there are none after examining at most $n-1$ bits, when $n = 1, 4, 7, 10, \ldots$.

   (c) (2 pts) How many n-bit strings are there with two consecutive ones? For which n is this number even or odd?

3. You are given a set of points in the plane. A point is *maximal* if there is no other point both both above and to the right. The subset of maximal points of points then forms a *staircase*.



The staircase of a set of points. Maximal points are black.

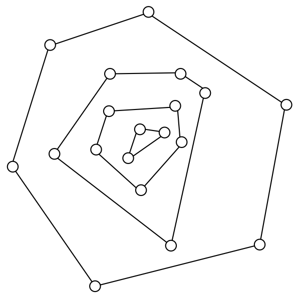
   (a) (0 pts) Prove that maximal points are *not* necessarily on the convex hull.

   (b) (6 pts) Give an $O(n \log n)$ algorithm to find the maximal points.

   (c) (4 pts) Assume that points are chosen uniformly at random within a rectangle. What is the average number of maximal points? Justify. Hint: you will be able to give an exact answer rather than just asymptotics. You have seen the same analysis before.

4. Given a set Q of points in the plane, define the *convex layers* of Q inductively as follows: The first convex layer of Q is just the convex hull of Q. For all $i > 1$, the ith convex layer is the convex hull of Q after the vertices of the first $i - 1$ layers have been removed.

   Give an $O(n^2)$-time algorithm to find all convex layers of a given set of n points.



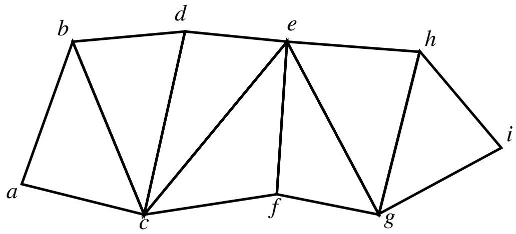
A set of points with four convex layers.

5. Prove that finding the second smallest of n elements takes $n + \lceil \lg n \rceil - 2$ comparisons in the worst case. Prove for both upper and lower bounds. Hint: find the (first) smallest using an elimination tournament.

6. *[This problem is required only for graduate students taking CS 373 for a full unit; anyone else can submit a solution for extra credit.]*

   Almost all computer graphics systems, at some level, represent objects as collections of triangles. In order to minimize storage space and rendering time, many systems allow objects to be stored as a set of *triangle strips*. A triangle strip is a sequence of vertices $\langle v_1, v_2, \ldots, v_k \rangle$, where each contiguous triple of vertices v_i, v_{i+1}, v_{i+2} represents a triangle. As the rendering system reads the sequence of vertices and draws the triangles, it keeps the two most recent vertices in a cache.

   Some systems allow triangle strips to contain *swaps*: special flags indicating that the order of the two cached vertices should be reversed. For example, the triangle strip $\langle a, b, c, d, \mathsf{swap}, e, f, \mathsf{swap}, g, h, i \rangle$ represents the sequence of triangles (a, b, c), (b, c, d), (d, c, e), (c, e, f), (f, e, g).



   Two triangle strips are *disjoint* if they share no triangles (although they may share vertices). The *length* of a triangle strip is the length of its vertex sequence, including swaps; for example, the example strip above has length 11. A *pure* triangle strip is one with no swaps. The adjacency graph of a triangle strip is a simple path. If the strip is pure, this path alternates between left and right turns.

   Suppose you are given a set S of interior-disjoint triangles whose adjacency graph is a tree. (In other words, S is a triangulation of a simple polygon.) Describe a linear-time algorithm to decompose S into a set of disjoint triangle strips of minimum total length.

Practice Problems

1. Consider the following generic recurrence for convex hull algorithms that divide and conquer:

$$T(n, h) = T(n_1, h_1) + T(n_2, h_2) + O(n)$$

   where $n \geq n_1 + n_2, h = h_1 + h_2$ and $n \geq h$. This means that the time to compute the convex hull is a function of both n, the number of input points, and h, the number of convex hull vertices. The splitting and merging parts of the divide-and-conquer algorithm take $O(n)$ time. When n is a constant, $T(n, h) = O(1)$, but when h is a constant, $T(n, h) = O(n)$. Prove that for both of the following restrictions, the solution to the recurrence is $O(n \log h)$:

   (a) $h_1, h_2 < \frac{3}{4}h$

   (b) $n_1, n_2 < \frac{3}{4}n$

2. Circle Intersection
   Give an $O(n \log n)$ algorithm to test whether any two circles in a set of size n intersect.

3. Basic polygon computations (assume *exact* arithmetic)

   (a) Intersection: Extend the basic algorithm to determine if two line segments intersect by taking care of *all* degenerate cases.

   (b) Simplicity: Give an $O(n \log n)$ algorithm to determine whether an n-vertex polygon is simple.

   (c) Area: Give an algorithm to compute the area of a simple n-polygon (not necessarily convex) in $O(n)$ time.

   (d) Inside: Give an algorithm to determine whether a point is within a simple n-polygon (not necessarily convex) in $O(n)$ time.

4. We are given the set of points one point at a time. After receiving each point, we must compute the convex hull of all those points so far. Give an algorithm to solve this problem in $O(n^2)$ total time. (We could obviously use Graham's scan n times for an $O(n^2 \log n)$-time algorithm). Hint: How do you maintain the convex hull?

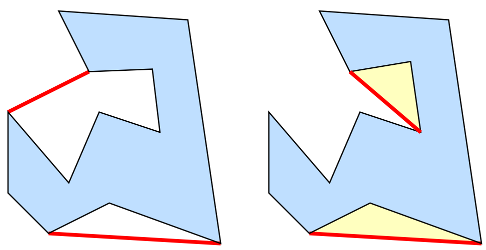
5. *(a) Given an n-polygon and a point outside the polygon, give an algorithm to find a tangent.

   (b) Suppose you have found both tangents. Give an algorithm to remove the points from the polygon that are within the angle formed by the tangents (as segments!) and the opposite side of the polygon.

   (c) Use the above to give an algorithm to compute the convex hull on-line in $O(n \log n)$

6. (a) A pair of polygon vertices defines an *external diagonal* if the line segment between them is completely outside the polygon. Show that every nonconvex polygon has at least one external diagonal.

   (b) Three consective polygon vertices p, q, r form a *mouth* if p and r define an external diagonal. Show that every nonconvex polygon has at least one mouth.



Two external diagonals Two mouths

7. A group of n ghostbusters is battling n ghosts. Each ghostbuster can shoot a single energy beam at a ghost, eradicating it. A stream goes in a straight line and terminates when it hits the ghost. The ghostbusters all fire at the same time and no two energy beams may cross. The positions of the ghosts and ghostbusters are fixed points in the plane.

(a) Prove that for any configuration of ghosts and ghostbusters, there is such a non-crossing matching. (Assume that no three points are collinear.)

(b) Show that there is a line passing through one ghostbuster and one ghost such that the number of ghostbusters on one side of the line equals the number of ghosts on the same side. Give an efficient algorithm to find such a line.

(c) Give an efficient divide and conquer algorithm to pair ghostbusters and ghosts so that no two streams cross.