(d) Find all primitive roots modulo 17.

(e) Prove that p has exactly $\phi(p-1)$ primitive roots.

[M] **5.206** Let p be a prime, g a primitive root modulo p, and $\gcd(a,p)=1$. Argue that there exists an integer x such that $g^x \equiv a \pmod{p}$. This x is called the *index* or *discrete logarithm* of a with respect to g. Prove that x is unique modulo $\phi(p)=p-1$. If we restrict x to an integer value between 0 and $p-2$, we denote $x=\operatorname{ind}_g a$. Let $x=\operatorname{ind}_g a$, $y=\operatorname{ind}_g b$, and $e\in\mathbb{Z}$. Prove that $\operatorname{ind}_g ab \equiv x+y \pmod{p-1}$ and $\operatorname{ind}_g a^e \equiv ex \pmod{p-1}$.

(**Remark:** The computation of $\operatorname{ind}_g a$, given p,g,a, is called the *discrete logarithm problem* (*DLP*). The DLP is a computationally difficult problem for which no polynomial-time algorithms are known. It is widely believed that solving the DLP is roughly as difficult as solving the IFP. Several cryptosystems are based on this apparent intractability of the DLP.)

[H] **5.207** Let p be a prime and g,g' two primitive roots modulo p. Assume that there exists an algorithm to compute discrete logarithms to the base g in polynomial time. Design an algorithm for computing discrete logarithms to the base g' in polynomial time.

Geometric algorithms

Unless otherwise specified, you may assume that the geometric objects in the following exercises are in general position.

[M] **5.208** **(a)** Deduce Formulas (23.1.1) and (23.1.2).

(b) Prove the claim following Equation (23.1.3) regarding the oriented line $\overline{P_1P_2}$ and the point P.

5.209 Write an algorithm that, given two oriented segments L_1 and L_2, computes the angle from L_1 to L_2 in $O(1)$ time. Your algorithm should return a value between 0 (inclusive) and 2π (exclusive).

5.210 Write an algorithm that, given the three vertices of a triangle, computes the area of the triangle in $O(1)$ time.

[M] **5.211** **(a)** Prove that an arbitrary intersection of convex regions in $\mathbb{R}^2$ is again convex.

(b) Let $S\subseteq\mathbb{R}^2$ be a convex region. What can you say about the convexity of the complementary region $\mathbb{R}^2\setminus S$?

(c) Demonstrate, by an example, that the union of two convex regions need not be convex.

(d) Supply an example where the union of two disjoint non-empty convex regions is again convex.

5.212 You are given a convex polygon with $n\geqslant 3$ edges.

(a) Devise an algorithm that computes the perimeter of the polygon in $O(n)$ time.

(b) Devise an algorithm that computes the area of the polygon in $O(n)$ time.

(c) Devise an algorithm that outputs a point inside (but not on) the polygon in $O(1)$ time.

[H] **(d)** Suppose that a point P on the polygon is given (assume that P is not a vertex of the polygon). Propose an $O(\log n)$-time algorithm that determines the exact edge on which P lies.

5.213 A circle C in the two-dimensional plane is specified by its center $Z=(c,d)$ and its radius r. Let $C_i=(Z_i,r_i)$ with $Z_i=(c_i,d_i)$ be two circles for $i=1,2$. In this exercise, we arrive at an algorithm to compute the area common to the two circles. For simplicity, assume that $c_1\neq c_2$ and $d_1\neq d_2$.

(a) Determine how we can identify whether the circles are disjoint, or overlap, or one of the circles is contained in the other (see Figure 109).

(b) Let $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$ be two points on a circle $C=(Z,r)$ with $Z=(c,d)$. The chord P_1P_2 dissects the circle into two segments. Consider the arc from P_1 to P_2 in counterclockwise

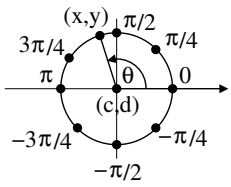
direction. This arc along with the chord P_1P_2 is one of the segments mentioned above. Propose an algorithm to compute the area of this segment.

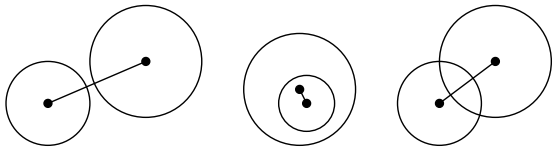
(c) Let C_1, C_2 be overlapping circles. Propose an algorithm to compute the two points of intersection P_1, P_2 of the given circles.

(d) Propose an algorithm to compute the area A of intersection of C_1 and C_2.

(e) How can you handle the special cases having $c_1 = c_2$ and/or $d_1 = d_2$?

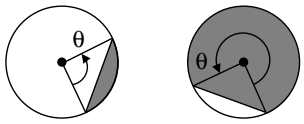
Figure 109: Intersection of two circles





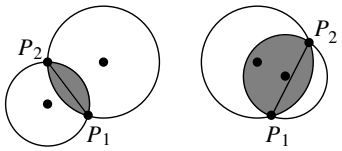
(a) Disjoint (b) Inside (c) Overlapping

Relative position of two circles



(a) $\theta < \pi$ (b) $\theta > \pi$

Area of segment



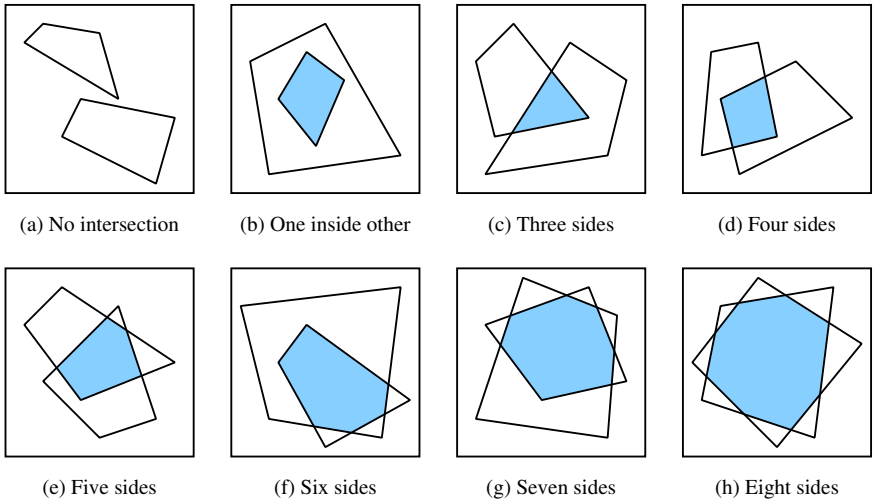
(a) Small arcs (b) Large arc

Intersection of two circles

[H^2] **5.214** Propose an algorithm to compute the area A of intersection of three circles $C_i = (Z_i, r_i)$ with $Z_i = (c_i, d_i)$ for $i = 1, 2, 3$. Assume that no two circles have the same x- or y-coordinate.

5.215 You are given two plane *convex* quadrilaterals (also called quadrangles and tetragons) R_1 and R_2. Each quadrilateral is specified by the x- and y-coordinates of its four corners in the counterclockwise direction starting from any corner. Assume that the quadrilaterals are in general position, that is, the sides of the given quadrilaterals are not parallel to one another, or to the x-axis or the y-axis. Moreover, there are no cases of touch, that is, if two sides of the two quadrilaterals intersect, they do so somewhere in their interiors (not at one or more corners). Propose an efficient (constant-time) algorithm to compute the intersection polygon of the two given quadrilaterals. Figure 110 describes the different possible cases of intersection.

Figure 110: Intersection of two plane convex quadrilaterals



(a) No intersection (b) One inside other (c) Three sides (d) Four sides

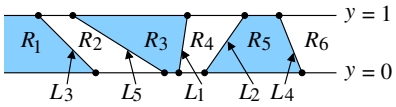
(e) Five sides (f) Six sides (g) Seven sides (h) Eight sides

5.216 Generalize Exercise 5.215 to compute the intersection of two plane convex n-gons (or of an n_1-gon with an n_2-gon with $n = n_1 + n_2$). What is the running time of your algorithm?

5.217 [*Steepest pair*] You are given n points in the plane. Design an $O(n \log n)$-time algorithm that finds a pair (P, Q) of the given points for which the line joining P and Q has the highest (absolute) slope among all pairs of the given points.

5.218 [*Flattest pair*] You are given n points in the plane. Design an $O(n \log n)$-time algorithm that finds a pair (P, Q) of the given points for which the line joining P and Q has the smallest (absolute) slope among all pairs of the given points.

5.219 You are given n straight line segments $L_1, L_2, \ldots, L_n$ each connecting a point on $y = 0$ to a point on $y = 1$. It is given that these lines are non-intersecting with one another. These segments partition the strip between $y = 0$ and $y = 1$ into $n + 1$ regions $R_1, R_2, \ldots, R_{n+1}$. The following figure illustrates a case of $n = 5$ segments. Each region is specified by the left and the right bounding segments. The first and the last regions are unbounded on one side, denoted by $-$. The six regions (shaded alternately) in the following figure are $R_1 = (-, L_3)$, $R_2 = (L_3, L_5)$, $R_3 = (L_5, L_1)$, $R_4 = (L_1, L_2)$, $R_5 = (L_2, L_4)$ and $R_6 = (L_4, -)$.



(a) Design an $O(n \log n)$-time algorithm to output the regions $R_1, R_2, \ldots, R_{n+1}$ from left to right.

(b) You are given a point P in the strip between $y = 0$ and $y = 1$, but not on any of the input segments L_i. Describe an $O(\log n)$-time algorithm to identify the region R_k to which P belongs.

5.220 We often need to compute the convex hull (smallest enclosing convex polygon) of general geometric objects.

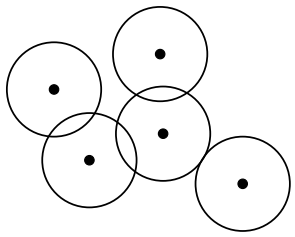
(a) Design an $O(n \log n)$-time algorithm to compute the convex hull of n triangles in the plane.

(b) Design an $O(n \log n)$-time algorithm to compute the convex hull of n quadrilaterals (not necessarily convex) in the plane.

(c) What is the smallest convex polygon enclosing a circle?

5.221 This exercise deals with the smallest convex *region* enclosing a set of circles of the same radius.

(a) The following figure shows five circles (of the same radius) in the plane. Draw the (boundary of the) smallest convex region enclosing these circles. Also indicate the convex hull of the centers of the five circles.



(b) Prove that the smallest convex region eenclosing the given circles consists of line segments and arcs of the given circles.

5.222 Let $P_1, P_2, \ldots, P_n$ be n points in the plane not necessarily sorted with respect to their x (or y) coordinates. Consider the recursive algorithm of Section 23.2.3. Prescribe an algorithm for merging C_1 and C_2 in $O(n)$ time. Deduce that this recursive algorithm runs in $O(n \log n)$ time. (**Remark:** This exercise demonstrates that the points $P_1, \ldots, P_n$ need not be sorted before the divide-and-conquer construction.)

5.223 Prove that any algorithm for computing the convex hull of n points in the plane must take a running time of $\Omega(n \log n)$ in the worst case. Thus, Graham's scan and Preparata and Hong's divide-and-conquer algorithm are both optimal. (**Hint:** Reduce the problem of sorting n numbers to the problem of finding the convex hull of n points in $\mathbb{R}^2$. For the reduction, you may use a suitable parabola.)

5.224 The four corners of a rectangle and some n points inside the rectangle are given. But it is not specified which four of the given $n+4$ points are the corners of the rectangle. Suggest an $O(n)$-time algorithm that identifies these four corners.

5.225 You are given a set of n points in the plane. Devise an $O(n)$-time algorithm that computes the rectangle of minimum area enclosing the given points and having sides parallel to the x and y axes.

[H] **5.226** This is similar to the previous exercise but with the restriction on the edges of the enclosing rectangle removed, that is, given a set S of n points in the plane, our task is to compute the minimum-area enclosing rectangle having any orientation.

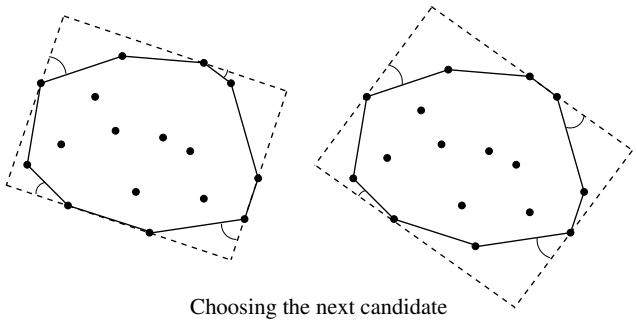
(a) Argue that it is sufficient to compute the minimum-area rectangle enclosing the convex hull $\mathrm{CH}(S)$ of the given points.

(b) Prove that (at least) one side of the smallest area rectangle is collinear with an edge of $\mathrm{CH}(S)$.

(c) Derive a straightforward $O(n^2)$-time algorithm that computes the minimum-area enclosing rectangle using the result of the last part.

(d) In order to reduce the complexity to $O(n \log n)$, use a modification of the above algorithm. The idea is to check the different candidate rectangles efficiently. We plan to investigate the candidates

following a particular sequence in such a way that the next candidate is generated from the current candidate in $O(1)$ time. The following figure illustrates this step.



Choosing the next candidate

In short, start with the enclosing rectangle of the previous exercise. Then, determine the smallest amount of clockwise rotation that lets one side of $CH(S)$ overlap with one edge of the enclosing rectangle. (Notice that this rotation may be zero.) Subsequently, at each step of the iteration determine the four angles as illustrated in the figure above, and rotate the rectangle clockwise by the minimum of these angles. The iteration stops after the total amount of rotation exceeds 90^0.

Fill out the details of this algorithm. Prove that this algorithm works correctly and also that it may be implemented so as to take $O(n)$ running time (excluding the time to compute the convex hull).

[H] **5.227** Suppose that you are given n points in the plane and your task is to compute the minimum-perimeter rectangle of any orientation, that encloses all the given points.

(a) Demonstrate, by an example, that the minimum-perimeter enclosing rectangle need not be the same as the minimum-area enclosing rectangle.

(b) Derive an $O(n \log n)$-time algorithm to determine the minimum-perimeter enclosing rectangle. (**Hint:** Use an algorithm similar to that for finding the minimum-area enclosing rectangle.)

[H^2] **5.228** Let S be a set of $n \geqslant 2$ points in the plane. The *diameter* of S is defined as

$$\text{diam}(S) = \max_{\substack{P,Q \in S \\ P \neq Q}} d(P,Q)\,,$$

where $d(\cdot\,,\cdot)$ is the Euclidean distance. In this exercise, we develop an $O(n \log n)$-time algorithm for computing $\text{diam}(S)$.

(a) Let $\text{diam}(S) = d(P,Q)$ for some $P, Q \in S$. Prove that P and Q are vertices of the convex hull $CH(S)$. In view of this result, we may assume, without loss of generality, that S itself equals the vertex set of $CH(S)$.

(b) Propose a straightforward $O(n^2)$-time algorithm to compute $\text{diam}(S)$.

(c) A sequence is called *unimodal* if it consists of a (possibly empty) increasing subsequence followed by a (possibly empty) decreasing subsequence. Let P be a vertex of $CH(S)$, and let $P, Q_1, Q_2, \ldots, Q_{n-1}$ be the clockwise listing of the vertices of $CH(S)$. Prove that the sequence $d(P,Q_1), d(P,Q_2), \ldots, d(P,Q_{n-1})$ need not be unimodal.

(d) By $d(Q,L)$, let us denote the (perpendicular) distance of a point Q from a line L. Now, let L be the line passing through P and Q_1. By convexity, it follows that the sequence $d(Q_2,L), d(Q_3,L), \ldots, d(Q_{n-1,L})$ is unimodal. Rename $P = Q_0$. Choose some $i \in \{0,1,2,\ldots,n-1\}$, and let L_i denote the line through Q_i and Q_{i+1} (where $i+1$ is considered modulo n). Let Q_k be a point of maximum

distance from L_i. Build a collection C of points (Q_i, Q_k) and (Q_{i+1}, Q_k) for all choices of i and for all k corresponding to the given i. Prove that the $\mathrm{diam}(S)$ is achieved by a pair in C.

(e) If the points of S are in general position, what is the maximum possible size of C?

(f) Establish that $\mathrm{diam}(S)$ can be computed in $O(n \log n)$ time.

(g) Design an algorithm that, given $\mathrm{CH}(S)$, computes $\mathrm{diam}(S)$ in $O(n)$ time.

[HM] **5.229** Let S be a set of n points in the plane and $C = \mathrm{CH}(S)$. A *line of support* for C is a straight line that touches C and that has all vertices of C on or to one side of it. Two vertices of C constitute an *antipodal pair*, if there exists a pair of parallel lines of support through these vertices. Prove that $\mathrm{diam}(S)$ is the maximum of the distances between points of antipodal pairs of C. (Notice that the algorithm of the last exercise essentially finds all antipodal pairs.)

5.230 A point P in the two-dimensional plane is said to *dominate* another point Q if their x- and y-coordinates satisfy the conditions: $x(P) \geqslant x(Q)$ and $y(P) \geqslant y(Q)$. A *maximal point* in a collection C of n points $P_1, P_2, \ldots, P_n$ is a point P_i which is not dominated by any other point P_j in the collection.

(a) Demonstrate by an example that a collection may have multiple maximal points.

(b) Prove or disprove: Every maximal point of C must lie on the convex hull of C.

(c) Propose an $O(n \log n)$-time algorithm to compute *all* the maximal points in a given collection C of n points. For simplicity, assume that the points in C do not have equal x- or y-coordinates.

5.231 Let C be as in Exercise 5.230. All the maximal points in C constitute a subcollection C_1 called the *first maximal layer* of C. All the maximal points in $C \setminus C_1$ constitute another subcollection C_2 called the *second maximal layer* of C. In general, if $C_1, C_2, \ldots, C_k$ are the first k maximal layers of C, then the $(k+1)$*-st maximal layer* of C is the set C_{k+1} of all maximal points in $C \setminus \bigcup_{i=1}^{k} C_i$. Prove that no algorithm can compute the maximal layers of a collection C of n points in $o(n \log n)$ time in the worst case.

[H] **5.232** Propose a worst-case $O(n \log n)$-time algorithm to compute all the maximal layers in a collection C of n points in the plane.

5.233 Consider the line-sweep algorithm of Section 23.3.1 used to compute the intersection points of a set of n line segments. Demonstrate by an example that an intersection point event may be inserted to and deleted from the event queue $\Theta(n)$ times.

5.234 Consider the line-sweep algorithm for computing line-segment intersections. Suppose that some of the input segments are allowed to be vertical (that is, parallel to the y-axis). You may, however, assume that no two given vertical segments are collinear.

(a) Define a new type of event "Vertical Segment" to deal with this situation. Describe how you can efficiently handle this event. You are required to maintain the running time of the original algorithm.

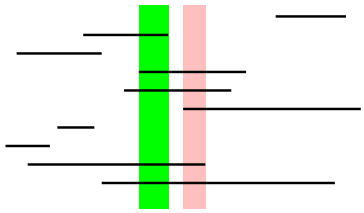
(b) Prove that you achieve a running time of $O((n+h) \log n)$ in presence of "Vertical Segment" events. You may assume that a height-balanced binary search tree on k nodes can be so implemented that the predecessor or successor of a node in the tree can be located in $O(\log k)$ time.

5.235 You are given a set of n line segments in the plane, each of which is either horizontal or vertical (that is, parallel to either the x- or the y-axis). Assume that no two of the given segments are collinear. Propose a line sweep algorithm that determines all intersection points of the given segments. Your algorithm should run in $O((n+h) \log n)$ time, where h is the number of intersection points.

5.236 An *interval* $[a,b]$ refers to the set of real numbers x satisfying $a \leqslant x \leqslant b$. Two intervals I_1 and I_2 are called *overlapping* if $I_1 \cap I_2 \neq \emptyset$.

You are given n intervals $I_1 = [a_1,b_1]$, $I_2 = [a_2,b_2]$, ..., $I_n = [a_n,b_n]$, each specified by its two endpoints. Your task is to find all the pairs (I_i,I_j) of overlapping intervals. Describe an $O(n\log n + h)$-time algorithm to solve this problem, where h is the number of overlapping pairs. You must supply an argument corroborating that your program achieves this running time. You may assume that the endpoints of the input intervals are in general position (no repetitions).

5.237 Let $I_i = [a_i,b_i]$, $i = 1,2,\ldots,n$, be a given set of n closed real intervals as in Exercise 5.236. Your task is to locate a region of maximum overlap of the given intervals. The following figure illustrates this problem for ten intervals. Here, the maximum overlap size is five, and is achieved twice.



Propose an efficient algorithm to solve this problem.

5.238 You are given n chords in a circle. Your task is to find all the points of intersection of the chords. Propose an $O((n+h)\log n)$-time ray-sweep algorithm to solve this problem, where h is the number of intersection points. Assume that the chords are in general position, that is, no two of them share an endpoint, and no three of them are concurrent.

5.239 You are given n chords in a circle. Each chord may be viewed as an opaque piece of string. Your task is to determine which chords are visible (fully or partially) from the center. Propose an $O((n+h)\log n)$-time ray-sweep algorithm for solving this problem, where h is the total number of intersections of the given chords. Assume that the chords are in general position, that is, no two of them share an endpoint, and no three of them are concurrent.

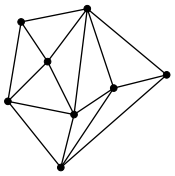
[HM] **5.240** Let S be a set of sites (in general position), and $\mathrm{Vor}(S)$ the Voronoi diagram of S.

(a) Let the Voronoi cells of $P,Q,R \in S$ meet at a point C. Clearly, C is equidistant form P, Q and R. Prove that the circle having its center at C and passing through P, Q and R does not contain any point of S in its interior. Conversely, if C is the center of a circle containing three points P,Q,R of S but no other points of S (on or) inside it, prove that C is a vertex in the Voronoi diagram $\mathrm{Vor}(S)$.

(b) Let $P,Q \in S$. Prove that there exists a circle passing through P and Q and containing no other point of S on or inside it if and only if the Voronoi cells of P and Q share an edge.

(c) Let $P,Q \in S$ be such that the Voronoi cells of P and Q share an edge. Prove that this edge is semi-infinite if and only if PQ is an edge of the convex hull $\mathrm{CH}(S)$. In particular, the Voronoi cell of a site $P \in S$ is infinite if and only if P is a vertex of $\mathrm{CH}(S)$.

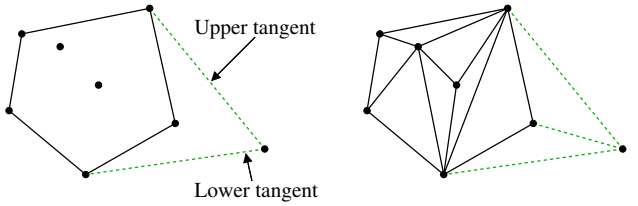
[M] **5.241** Let S be a set of points in the plane. A *triangulation* of S is a way of joining the points of S by line segments in such a way that every face inside the convex hull of S is a triangle. An example is provided in the following figure.



Plane triangulation

Assume that the points of S are in general position. Let $\mathrm{CH}(S)$ contain h points, and let the number of points of S that lie in the interior of $\mathrm{CH}(S)$ be k. Finally, let $n = h + k$. Prove that every triangulation of S contains exactly $h+2k-2 = 2n-h-2$ (internal) faces and exactly $2h+3k-3 = 3n-h-3$ edges (including the edges on the hull).

5.242 This exercise deals with an algorithm for computing a triangulation of a set S of $n \geqslant 3$ points. Assume that the points $P_1, P_2, \ldots, P_n$ are provided in the sorted order of their x-coordinates. We iteratively compute the convex hull $C_i = \mathrm{CH}(P_1, \ldots, P_i)$. Initially, C_3 is computed by joining the vertices P_1, P_2, P_3 so as to form a triangle. Now, suppose that C_{i-1} has been computed. The point P_i lies to the right of C_i. We compute the lower and upper tangents of C_i passing through P_i, and discard the vertices of C_i that lie strictly between these two tangents. Part (a) of the following figure illustrates this construction.



(a) Incremental construction of a convex hull

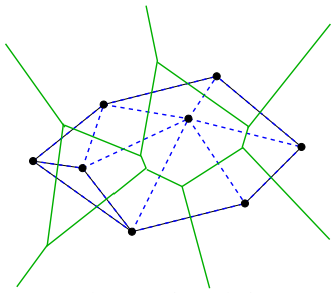
(b) Computation of triangulation

(a) Establish the details of the above strategy together with the underlying data structures so that $C_n = \mathrm{CH}(S)$ can be computed in $\mathrm{O}(n)$ time.

(b) Suppose that, in addition to computing the tangents and updating the convex hull, we also join the new point P_i with all the points lying between the two tangents (including the points of tangency). This leads to a triangulation of S as illustrated in Part (b) of the above figure. Prove that this additional overhead can be incorporated in $\mathrm{O}(n)$ time.

(Notice that if the points of S are not provided in the sorted order of x-coordinates, we first spend $\mathrm{O}(n \log n)$ time to sort them. Thus, the triangulation of a set of n points in the plane can be computed in $\mathrm{O}(n \log n)$ time.)

[H] **5.243** Let S be a set of n sites in the plane. Join the sites P and Q by a segment if and only if the Voronoi cells of P and Q share an edge (or equivalently, a part of the perpendicular bisector of the segment PQ is an edge of $\mathrm{Vor}(S)$). This construction is illustrated in the following figure.



Delaunay triangulation

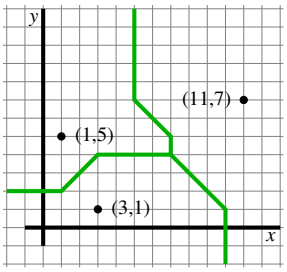
Prove that the above construction yields a triangulation of S. This is called the *Delaunay triangulation* of S. We thus get yet another $O(n \log n)$-time algorithm for triangulating S.

5.244 Let S be a set of n points in the plane. Using the properties of triangulation, prove that the number of vertices in $\mathrm{Vor}(S)$ must be between $n-2$ and $2n-5$, and also that the number of edges in $\mathrm{Vor}(S)$ must be between $2n-3$ and $3n-6$.

[H] **5.245** Let S be a set of n points in the plane. Consider the complete undirected graph G whose vertices are the points of S. For vertices $P, Q \in S$, assign the weight $d(P,Q)$ to the edge PQ. Prove that the minimum spanning tree of G must be a subtree of the Delaunay triangulation of S.

[H^{2}M] **5.246** In this exercise, we investigate the Voronoi diagram of a set S of points in the plane with respect to a non-conventional distance. Imagine that a very well-planned city has an orthogonal mesh of roads with two consecutive roads separated by 100 meters both along x and y directions. The points of S are the sites of grocery stores as usual. An inhabitant of that city will have to travel roads parallel to x and y axes in order to reach a grocery store. In this case, the distance d_1 is a good approximation to the amount of walk necessary between two points.

Recall that for two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, we define $d_1(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2|$. The following figure illustrates the Voronoi diagram of three points with respect to this distance.



(a) Determine the locus of the points that are equidistant from two given points (h_1, k_1) and (h_2, k_2). Conclude that the region consisting of the points closer to one of the given points than the other is not convex in general. (**Hint:** Let μ denote the slope of the segment joining the two given points. Consider the three cases separately: $|\mu| < 1$, $|\mu| > 1$ and $|\mu| = 1$.)

(b) Draw the Voronoi diagram with respect the distance d_1 for the five points: $(1,7)$, $(4,1)$, $(5,5)$, $(7,9)$ and $(10,5)$.

(c) Determine the locus of the points that are equidistant (with respect to the distance d_1) from a given line $y = a$ parallel to the x-axis and from a given point (h,k) with $k > a$.

(d) Suppose that we attempt to compute the Voronoi diagram of S using a line sweep algorithm as described in the text. At a certain instance, the sweep line is positioned along the x-axis (that is, $y = 0$). Suppose that there are only three points of S above the sweep line: $(4,4)$, $(10,7)$ and $(18,8)$. Draw the beach line for this configuration.

(**Remark:** I do not know whether a sweep algorithm has been proposed in the literature for computing Voronoi diagrams under non-conventional notions of distance. A generalization of the divide-and-conquer algorithm described in the text has been proposed by Lee and runs in $O(n \log n)$ time, where n is the number of sites. Voronoi diagrams with respect to the distance d_1 find extensive use in computing circuit layouts for IC chip design.)

[M] **5.247** The d_∞-distance between two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ in the plane is defined as $d_\infty(P_1, P_2) = \max(|x_1 - x_2|, |y_1 - y_2|)$. Draw the Voronoi diagram of the three points $P_1 = (0,0)$, $P_2 = (8,6)$ and $P_3 = (12,-4)$ with respect to the d_∞-distance.