List of Problems

Spring 2006

As part of the course you have to write a paper about one of the problems on the list below. The papers are supposed to be written in pairs. Each pair has to select three preferred problems from the list by **April 6**, I will then assign each pair a topic by **April 7**, if possible from their preferred problems. You are supposed to be able to solve each problem with the knowledge gained in this course. However, discussions with your fellow students or with me are highly encouraged, while researching on the internet is highly discouraged.

Requirements: Your paper should be roughly 5.000 words in length, which is equivalent to about 10 pages of text. It has to be written in English, using an appropriate and consistent layout, with sufficient and suitable illustrations of good quality (where applicable). Carefully phrase definitions, lemmas, and theorems and make sure that you cite all sources appropriately. Use a spell checker and if you are not sure that your knowledge of English grammar is sufficient, get a more qualified friend to help you—badly written English will influence your grade negatively.

Time line

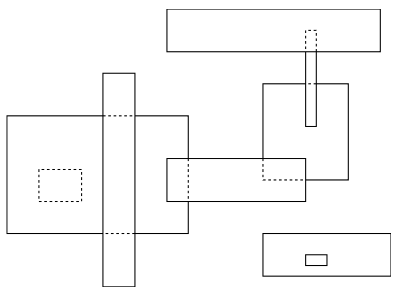
April 6. Email me your list of 3 preferred problems, together with your name and student number and the name and student number of your partner. I will post the list of assigned problems on the course web page on **April 7**.

May 15-19. Make an appointment with me to talk about your progress and to address any questions you might have.

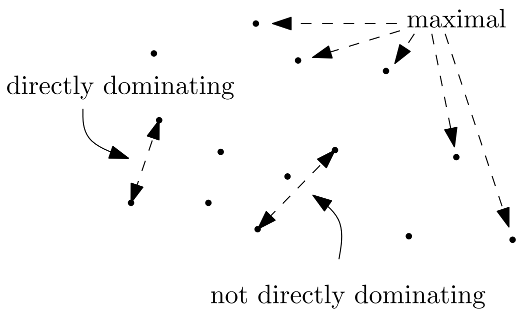
June 23. The final paper is due. Place a printout of the paper (which clearly states your names and student numbers) in my mailbox opposite room HG 7.22.

1 Hidden-line elimination

Let R be a set of n rectangles in $\mathbb{R}^3$ which lie parallel to the xy-plane (but on different heights) and whose edges are parallel to the x-axis or the y-axis. Imagine now that we are looking down onto the rectangles: some of the edges we see completely, some we do not see at all, and from some edges we see only one or more parts. Give an algorithm that decides for each edge e which part of e is visible. Your algorithm should have a running time in $O(n \log n + I \log n)$ where I is the number of intersections between the edges of the rectangles in a projection onto the xy-plane.



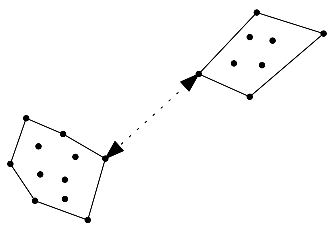
2 Maximal and dominating points



Let S be a set of n points in the plane. A point $p = (p_x, p_y)$ is *dominating* a point $q = (q_x, q_y)$ if $p_x \geq q_x$ and $p_y \geq q_y$. A point in S is *maximal* if it is not dominated by any other point in S. Describe an $O(n \log n)$ algorithm that computes all maximal points in S.

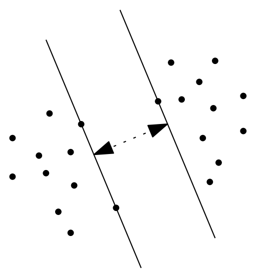
A point $p \in S$ is *directly* dominating a point $q \in S$ if there is no point r such that p dominates r and r dominates q. Give an algorithm that computes all pairs $(p, q) \in S \times S$ such that p directly dominates q. Your algorithm should have a running time in $O(n \log n + k)$ where k is the number of reported pairs.

3 Clustering



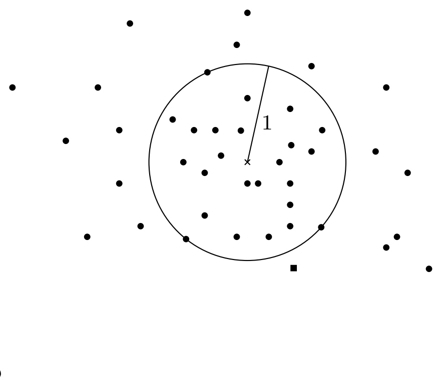
Develop a linear time (or faster . . .) algorithm that determines the distance between two convex polygons. Use this algorithm to give an $O(n^3)$ algorithm that solves the following clustering problem: Split a set S of n points in the plane into two such that the distance of the convex hulls of the two parts is maximized.

4 The widest corridor



Let S be a set of points in the plane. A *corridor* in S is a strip—the region between two parallel lines—such that at least one point of S lies on each side of the strip. The width of a corridor is the distance between the two lines that define the strip. Give an $O(n^2)$ algorithm that computes the widest corridor in S.

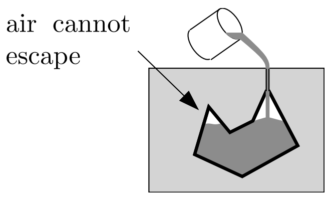
5 The most densely populated unit circle



Imagine that we would like to build an ice cream parlour close to as many high schools as possible. We assume that students are willing to walk a certain distance to buy ice cream. Therefore, given the location of all n high schools as points in the plane, we would like to place a unit circle—a circle with radius one—in such a way that it contains the maximal number of points. Give an algorithm that does this in $O(n^2)$.

6 Filling molds

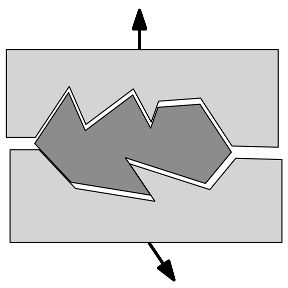
Let $\mathcal{P}$ be a simple planar polygon with n vertices. We would like to produce $\mathcal{P}$ by pouring liquid metal into a mold. After the metal hardens we should have an exact copy of $\mathcal{P}$. (This problem does indeed concern the production of 2-dimensional objects.) We model the mold as a rectangle with a hole in the shape of $\mathcal{P}$. We then create a vertical tunnel—through which we will pour the liquid metal—that ends at the highest point of $\mathcal{P}$. Unfortunately we might encounter a serious problem during the production of $\mathcal{P}$: when we are pouring the liquid metal into the mold we might create pockets of air that cannot escape.



Give an $O(n \log n)$ algorithm that decides if there is an orientation of $\mathcal{P}$ where we can be sure not to create any such pockets of air (if there is such an orientation then your algorithm should report it). Furthermore, sketch a possible generalization of your algorithm to 3-dimensional polyhedra.

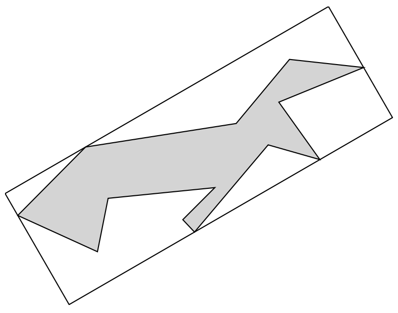
7 Removing molds

Again—as previously discussed in Section 4 of the book and Problem 6—we would like to create an object by pouring liquid metal into a mold and letting it harden. Here, however, we are considering molds that consist of two pieces. Give an algorithm that decides for a given simple polygon $\mathcal{P}$ if there is a two-piece mold such that both parts of the mold can be removed after the object has been created. Distinguish between the case where the parts have to be removed in opposite directions and the case where the parts can be removed in arbitrary directions. For both cases give an $O(n \log n)$ algorithm.



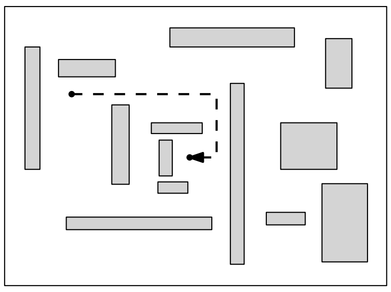
8 Smallest bounding box of a polygon

Let $\mathcal{P}$ be a simple planar polygon with n vertices. A *bounding box* of $\mathcal{P}$ is a rectangle that completely contains $\mathcal{P}$. This rectangle does not have to be axis-aligned but it can have any arbitrary orientation. Give an algorithm that computes a bounding box for $\mathcal{P}$ that has the smallest perimeter among all possible bounding boxes in $O(n \log n)$ time.



Furthermore, also give an algorithm for the 3-dimensional problem where you have to compute the smallest bounding box for a polyhedron with n vertices. Analyze you algorithm. (This algorithm does not need to work in $O(n \log n)$ time.)

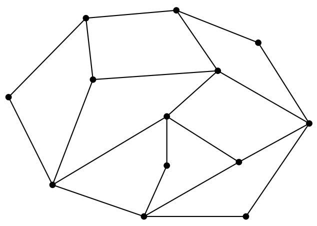
9 Rectilinear link distance



Imagine that we have a robot that moves through a factory. The factory is a rectangle that contains n rectangular obstacles. The edges of all rectangles are parallel to the x- or y-axis. The robot—which we imagine to be simply a point—can not move in arbitrary directions but only parallel to the x- or y-axis. To change its direction the robot has to first slow down, then stop, then turn, and finally accelerate again. Therefore, given starting and final position, we would like to compute a path for the robot that avoids all obstacles and contains as few turns as possible. Give an algorithm that computes such a path in $O(n^2)$ time.

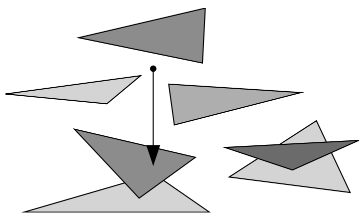
Now imagine that we define the cost of a particular path as the number of turns plus the sum of the Euclidean distances of the line segments that constitute the path. Extend your algorithm such that it finds the cheapest path (still in $O(n^2)$ time).

10 Quadrangulations



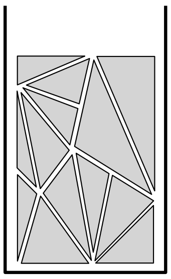
Let S be a set of n points in the plane. A *quadrangulation* of S is a partition of the convex hull of S into quadrilaterals (quadrangles) such that the vertices of the quadrilaterals are precisely the points of S. Give a necessary and sufficient condition that assures that a quadrangulation does always exist and give an $O(n \log n)$ algorithm that computes a quadrangulation of a set of points. Be careful to ensure that your proof and algorithm also work correctly if three or more points lie on a line.

11 Vertical ray shooting



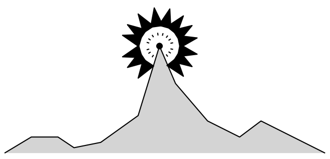
Let $S = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\}$ be a set of polygons which are parallel to the xy-plane but lie on different heights. We would like to store S in a data structure such that we can query with a 3-dimensional point q and quickly report the polygon that lies directly under q (if such a polygon exists). Describe such a data structure for the special case that all polygons are triangles. Your structure should use at most $O(n^2)$ space and support $O(\log^2 n)$ query time. Furthermore, describe a structure which uses $O(n \log^2 n)$ space and supports $O(\log^3 n)$ query time if the polygons are rectangles whose edges are parallel to the the x- or y-axis.

12 Construction sequences



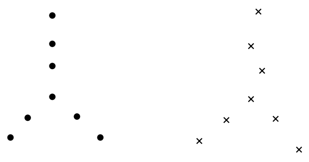
Imagine that we would like to manufacture a product that consists of several parts. To simplify the problem we will consider only the planar case where all parts are triangles. We assemble the product by vertically translating each part into its proper position. This implies that we first have to move the bottommost parts into position, then the next higher ones, etc. Prove that there is always a construction sequence—if the parts are triangles—and give an $O(n \log n)$ algorithm that computes such a sequence.

13 Visibility in terrains



Assume that we are given a polyhedral terrain, that is, a triangulation of a planar point set where each point has been assigned a height value (see Section 9 in the book). Describe an algorithm that decides in $O(n)$ time if there is a point on the terrain that sees all other points. You can assume that the boundary of the terrain has height zero (sea level) and that the remainder of the terrain has positive height. Extend your algorithm such that it can compute the position of the shortest watchtower that can see the complete terrain. The extended algorithm should run in $O(n^2)$ time.

14 Matching point sets

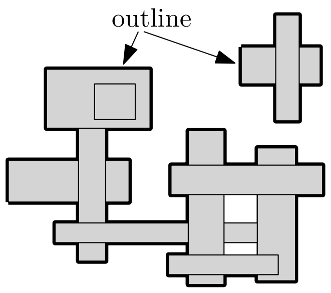


Let $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_n\}$ be two sets of n points in the plane. We would like to determine how closely P matches Q. For this we assume that there is a correspondence between the points in P and Q: p_1 belongs to q_1, p_2 belongs to q_2, etc. To decide if P matches Q we will try to translate P such that p_i lies close to q_i for each $1 \leq i \leq n$. If such a translation is possible, then P matches Q.

We can formalize this idea as follows: For a given point p and a translation vector τ let $p(\tau) := p + \tau$. Define $P(\tau) := \{p(\tau) : p \in P\}$, that is, we obtain $P(\tau)$ by translating P by τ. Define the distance between two points p and q as $\text{dist}(p, q) := |p_x - q_x| + |p_y - q_y|$. (This distance function is called the L_1-distance.) We now can define the distance between $P(\tau)$ and Q as the maximal distance between a pair of points p_i, q_i, that is, $\text{dist}(P(\tau), Q) := \max_{1 \leq i \leq n} \text{dist}(p_i, q_i)$. Describe an algorithm that decides for a given $\varepsilon \geq 0$ if there is a translation τ such that $\text{dist}(P(\tau), Q) \leq \varepsilon$.

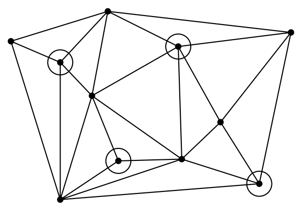
Furthermore, consider the case where we allow up to k points from P to be outliers, that is, these points can be further than ε away from their corresponding points in Q. Give an $O(n^2)$ algorithm that determines if P matches Q under this assumption.

15 Outline of a set of rectangles



Let R be a set of axis-parallel rectangles in the plane. The complement of the union of the rectangles $\bigcup R$ can consist of many regions. One region is infinite, the other regions are holes in the union. The *outline* of $\bigcup R$ is the boundary between $\bigcup R$ and the infinite region of the complement of $\bigcup R$. Prove that the outline has $O(n)$ complexity and give an $O(n \log n)$ algorithm that computes it.

16 Point location in triangulations



Let $\mathcal{T}$ be a triangulation of a set of n points. An *independent set* with respect to $\mathcal{T}$ is a subset of the points such that no two points in this subset are connected by an edge of the triangulation. Prove that there is constant $\alpha > 0$ such that every triangulation has an independent set of size at least αn that consists of points with degree at most 12. (The degree of a point is the number of incident edges.) Use this fact to built a hierarchical point location structure for a triangulation. Your structure should use at most $O(n)$ space and support $O(\log n)$ query time.