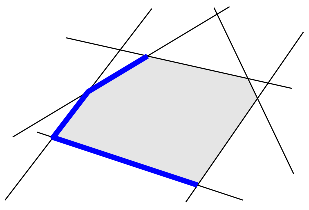
For each cell of $Z_{\mathcal{A}(L)}(\ell)$ split its boundary at its topmost vertex and at its bottommost vertex and orient all edges from bottom to top, horizontal edges from left to right. Those edges that have the cell to their right are called *left-bounding* for the cell and those edges that have the cell to their left are called right-bounding. For instance, for the cell depicted to the right all left-bounding edges are shown blue and bold.



We will show that there are at most $3n$ left-bounding edges in $Z_{\mathcal{A}(L)}(\ell)$ by induction on n. By symmetry, the same bound holds also for the number of right-bounding edges in $Z_{\mathcal{A}(L)}(\ell)$.

For $n = 1$, there is at most one (exactly one, unless ℓ is parallel to and lies above the only line in L) left-bounding edge in $Z_{\mathcal{A}(L)}(\ell)$ and $1 \leqslant 3n = 3$. Assume the statement is true for $n - 1$.

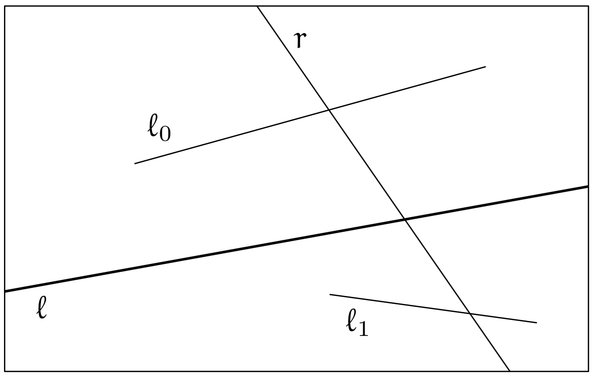


Figure 9.3: *At most three new left-bounding edges are created by adding* r *to* $\mathcal{A}(L\backslash\{r\})$.

If no line from L intersects ℓ, then all lines in $L \cup \{\ell\}$ are horizontal and there is at most $1 < 3n$ left-bounding edge in $Z_{\mathcal{A}(L)}(\ell)$. Else consider the rightmost line r from L intersecting ℓ and the arrangement $\mathcal{A}(L \setminus \{r\})$. By the induction hypothesis there are at most $3n - 3$ left-bounding edges in $Z_{\mathcal{A}(L\setminus\{r\})}(\ell)$. Adding r back adds at most three new left-bounding edges: At most two edges (call them ℓ_0 and ℓ_1) of the rightmost cell of $Z_{\mathcal{A}(L\setminus\{r\})}(\ell)$ are intersected by r and thereby split in two. Both of these two edges may be left-bounding and thereby increase the number of left-bounding edges by at most two. In any case, r itself contributes exactly one more left-bounding edge to that cell. The line r cannot contribute a left-bounding edge to any cell other than the rightmost: to the left of r, the edges induced by r form right-bounding edges only and to the right of r all other cells touched by r (if any) are shielded away from ℓ by one of ℓ_0 or ℓ_1. Therefore, the total number of left-bounding edges in $Z_{\mathcal{A}(L)}(\ell)$ is bounded from above by $3 + 3n - 3 = 3n$. □

Corollary 9.8 *The arrangement of* n *lines in* $\mathbb{R}^2$ *can be constructed in optimal* $O(n^2)$ *time and space.*

Proof. Use the incremental construction described above. In Step i, for $1 \leqslant i \leqslant n$, we do a linear search among $i-1$ elements to find the starting face and then traverse (part of) the zone of the line ℓ_i in the arrangement $\mathcal{A}(\{\ell_1, \ldots, \ell_{i-1}\})$. By Theorem 9.7 the complexity of this zone and hence the time complexity of Step i altogether is $O(i)$. Overall we obtain $\sum_{i=1}^{n} ci = O(n^2)$ time (and space), for some constant $c > 0$, which is optimal by Theorem 9.5. □

The corresponding bounds for hyperplane arrangements in $\mathbb{R}^d$ are $\Theta(n^d)$ for the complexity of a simple arrangement and $O(n^{d-1})$ for the complexity of a zone of a hyperplane.

Exercise 9.9 *For an arrangement $\mathcal{A}$ of a set of n lines in $\mathbb{R}^2$, let $\mathcal{F} := \bigcup_{C \text{ is cell of } \mathcal{A}} \overline{C}$ denote the union of the closure of all bounded cells. Show that the complexity (number of vertices and edges of the arrangement lying on the boundary) of $\mathcal{F}$ is $O(n)$.*

9.4 The Power of Duality

The real beauty and power of line arrangements becomes apparent in context of projective point $\leftrightarrow$ line duality. It is often convenient to assume that no two points in the primal have the same x-coordinate so that no line defined by any two points is vertical (and hence becomes an infinite point in the dual). This degeneracy can be tested for by sorting according to x-coordinate (in $O(n \log n)$ time) and resolved by rotating the whole plane by some sufficiently small angle. In order to select the rotation angle it is enough to determine the line of maximum absolute slope that passes through two points. Then we can take, say, half of the angle between such a line and the vertical direction. As the line of maximum slope through any given point can be found in linear time, the overall maximum can be obtained in $O(n^2)$ time.

The following problems can be solved in $O(n^2)$ time and space by constructing the dual arrangement.

General position test. Given n points in $\mathbb{R}^2$, are any three of them collinear? (Dual: do three lines meet in a point?)

Minimum area triangle. Given n points in $\mathbb{R}^2$, what is the minimum area triangle spanned by any three of them? For any vertex ℓ^* of the dual arrangement (primal: line ℓ through two points p and q) find the closest point vertically above/below ℓ^* through which an input line passes (primal: closest line below/above and parallel to ℓ that passes through an input point). In this way one can find $O(n^2)$ candidate triangles by constructing the arrangement of the n dual lines. For instance, maintain over the incremental construction for each vertex a vertically closest line. The number of vertices to be updated during insertion of a line ℓ corresponds to the complexity of the zone of ℓ in the arrangement constructed so far. Therefore maintaining this information comes at no extra cost asymptotically.

The smallest among those candidates can be determined by a straightforward minimum selection (comparing the area of the corresponding triangles). Observe that vertical distance is not what determines the area of the corresponding triangle but orthogonal distance. However, the points that minimize these measures for any fixed line are the same...

Exercise 9.10 *A set* P *of* n *points in the plane is said to be in* ε-general position *for* $\varepsilon > 0$ *if no three points of the form*

$$p + (x_1, y_1), q + (x_2, y_2), r + (x_3, y_3)$$

are collinear, where $p, q, r \in P$ *and* $|x_i|, |y_i| < \varepsilon$*, for* $i \in \{1, 2, 3\}$*. In words:* P *remains in general position under changing point coordinates by less than* ε *each.*

Give an algorithm with runtime $O(n^2)$ *for checking whether a given point set* P *is in* ε*-general position.*

9.5 Rotation Systems—Sorting all Angular Sequences

Recall the notion of a combinatorial embedding from Chapter 2. It is specified by the circular order of edges along the boundary of each face or—equivalently, dually—around each vertex. In a similar way we can also give a combinatorial description of the geometry of a finite point set $P \subset \mathbb{R}^2$ using its **rotation system**. This is nothing else but a combinatorial embedding of the complete geometric (straight line) graph on P, specified by the circular order of edges around vertices.[1]

For a given set P of n points, it is trivial to construct the corresponding rotation system in $O(n^2 \log n)$ time, by sorting each of the n lists of neighbors independently. The following theorem describes a more efficient, in fact optimal, algorithm.

Theorem 9.11 *Consider a set* P *of* n *points in the plane. For a point* $q \in P$ *let* $c_P(q)$ *denote the circular sequence of points from* $S \setminus \{q\}$ *ordered counterclockwise around* q *(in order as they would be encountered by a ray sweeping around* q*). The rotation system of* P*, consisting of all* $c_P(q)$*, for* $q \in P$*, collectively can be obtained in* $O(n^2)$ *time.*

Proof. Consider the projective dual P^* of P. An angular sweep around a point $q \in P$ in the primal plane corresponds to a traversal of the line q^* from left to right in the dual plane. (A collection of lines through a single point q corresponds to a collection of points on a single line q^* and slope corresponds to x-coordinate.) Clearly, the sequence of intersection points along all lines in P^* can be obtained by constructing the arrangement in $O(n^2)$ time. In the primal plane, any such sequence corresponds to an order of the remaining points according to the slope of the connecting line; to construct the circular

[1]As these graphs are not planar for $|P| \geqslant 5$, we do not have the natural dual notion of faces as in the case of planar graphs.

sequence of points as they are encountered around q, we have to split the sequence obtained from the dual into those points that are to the left of q and those that are to the right of q; concatenating both yields the desired sequence. □

Exercise 9.12 (Eppstein [1]) *Describe an $O(n^2)$ time algorithm that given a set P of n points in the plane finds a subset of five points that form a strictly convex empty pentagon (or reports that there is none if that is the case). Empty means that the convex pentagon may not contain any other points of P.*

Hint: *For each $p \in P$ discard all points to the left of p and consider the polygon $S(p)$ formed by p and the remaining points taken in circular order around p. Explain why it suffices to check for all p whether $S(p)$ has four vertices other than p that form an empty convex quadrilateral. How do you check this in $O(n^2)$ time?*

Remark: *It was shown by Harborth [5] that every set of ten or more points in general position contains a subset of five points that form a strictly convex empty pentagon.*

9.6 3-Sum

The 3-Sum problem is the following: Given a set S of n integers, does there exist a three-tuple[2] of elements from S that sum up to zero? By testing all three-tuples this can obviously be solved in $O(n^3)$ time. If the tuples to be tested are picked a bit more cleverly, we obtain an $O(n^2)$ algorithm.

Let $(s_1, \dots, s_n)$ be the sequence of elements from S in increasing order. This sequence can be obtained by sorting in $O(n \log n)$ time. Then we test the tuples as follows.

```
For $i = 1, \dots, n$ {
    $j = i$, $k = n$.
    While $k \geqslant j$ {
        If $s_i + s_j + s_k = 0$ then exit with triple $s_i, s_j, s_k$.
        If $s_i + s_j + s_k > 0$ then $k = k - 1$ else $j = j + 1$.
    }
}
```

The runtime is clearly quadratic. Regarding the correctness observe that the following is an invariant that holds at the start of every iteration of the inner loop: $s_i + s_x + s_k < 0$, for all $x \in \{i, \dots, j-1\}$, and $s_i + s_j + s_x > 0$, for all $x \in \{k+1, \dots, n\}$.

Interestingly, until very recently this was the best algorithm known for 3-Sum. But at FOCS 2014, Grønlund and Pettie [4] present a deterministic algorithm that solves 3-Sum in $O(n^2(\log\log n / \log n)^{2/3})$ time. They also give a bound of $O(n^{3/2}\sqrt{\log n})$ on

[2]That is, an element of S may be chosen twice or even three times, although the latter makes sense for the number 0 only. :-)

the decision tree complexity of 3-Sum. The big open question remains whether an $O(n^{2-\varepsilon})$ algorithm can be achieved. On the other hand, in some very restricted models of computation—such as the linear decision tree model—3-Sum cannot be solved in sub-quadratic time [2].

3-Sum hardness There is a whole class of problems that are equivalent to 3-Sum up to sub-quadratic time reductions [3]; such problems are referred to as **3-Sum-hard**.

Definition 9.13 *A problem* P *is* **3-Sum-hard** *if and only if every instance of 3-Sum of size* n *can be solved using a constant number of instances of* P*—each of* $O(n)$ *size—and* $o(n^2)$ *additional time.*[3]

For instance, it is not hard to show that the following variation of 3-Sum—let us denote it by 3-Sum°—is 3-Sum hard: Given a set S of n integers, does there exist a three-element subset of S whose elements sum up to zero?

Exercise 9.14 *Show that 3-Sum° is 3-Sum hard.*

As another example, consider the Problem **GeomBase**: Given n points on the three horizontal lines $y = 0$, $y = 1$, and $y = 2$, is there a non-horizontal line that contains at least three of them?

3-Sum can be reduced to GeomBase as follows. For an instance $S = \{s_1, \ldots, s_n\}$ of 3-Sum, create an instance P of GeomBase in which for each s_i there are three points in P: $(s_i, 0)$, $(-s_i/2, 1)$, and $(s_i, 2)$. If there are any three collinear points in P, there must be one from each of the lines $y = 0$, $y = 1$, and $y = 2$. So suppose that $p = (s_i, 0)$, $q = (-s_j/2, 1)$, and $r = (s_k, 2)$ are collinear. The inverse slope of the line through p and q is $\frac{-s_j/2-s_i}{1-0} = -s_j/2 - s_i$ and the inverse slope of the line through q and r is $\frac{s_k+s_j/2}{2-1} = s_k + s_j/2$. The three points are collinear if and only if the two slopes are equal, that is, $-s_j/2 - s_i = s_k + s_j/2 \iff s_i + s_j + s_k = 0$.

A very similar problem is **General Position**, in which one is given n arbitrary points and has to decide whether any three are collinear. For an instance S of 3-Sum°, create an instance P of General Position by projecting the numbers s_i onto the curve $y = x^3$, that is, $P = \{(a, a^3) \mid a \in S\}$.

Suppose three of the points, say, (a, a^3), (b, b^3), and (c, c^3) are collinear. This is the case if and only if the slopes of the lines through each pair of them are equal. (Observe

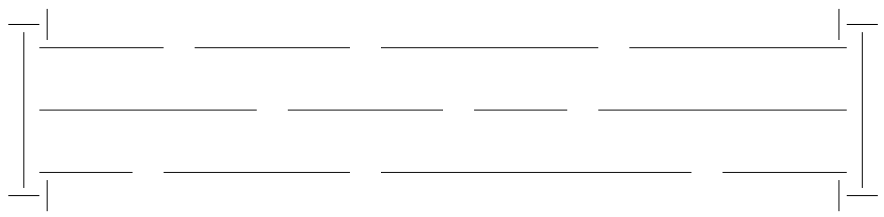
[3]In light of the recent results of Grønlund and Pettie one should probably write $O(n^{2-\varepsilon})$ here. Anyway, the reductions discussed here will be either linear or $O(n \log n)$ time.

that a, b, and c are pairwise distinct.)

$$\begin{aligned}(b^3-a^3)/(b-a) &= (c^3-b^3)/(c-b) \iff\\ b^2+a^2+ab &= c^2+b^2+bc \iff\\ b &= (c^2-a^2)/(a-c) \iff\\ b &= -(a+c) \iff\\ a+b+c &= 0\,.\end{aligned}$$

Minimum Area Triangle is a strict generalization of General Position and, therefore, also 3-Sum-hard.

In **Segment Splitting/Separation**, we are given a set of n line segments and have to decide whether there exists a line that does not intersect any of the segments but splits them into two non-empty subsets. To show that this problem is 3-Sum-hard, we can use essentially the same reduction as for GeomBase, where we interpret the points along the three lines $y = 0$, $y = 1$, and $y = 2$ as sufficiently small "holes". The parts of the lines that remain after punching these holes form the input segments for the Splitting problem. Horizontal splits can be prevented by putting constant size gadgets somewhere beyond the last holes, see the figure below. The set of input segments for the segment



splitting problem requires sorting the points along each of the three horizontal lines, which can be done in $O(n \log n) = o(n^2)$ time. It remains to specify what "sufficiently small" means for the size of those holes. As all input numbers are integers, it is not hard to show that punching a hole of $(x - 1/4, x + 1/4)$ around each input point x is small enough.

In **Segment Visibility**, we are given a set S of n horizontal line segments and two segments $s_1, s_2 \in S$. The question is: Are there two points, $p_1 \in s_1$ and $p_2 \in s_2$ which can see each other, that is, the open line segment $\overline{p_1p_2}$ does not intersect any segment from S? The reduction from 3-Sum is the same as for Segment Splitting, just put s_1 above and s_2 below the segments along the three lines.

In **Motion Planning**, we are given a robot (line segment), some environment (modeled as a set of disjoint line segments), and a source and a target position. The question is: Can the robot move (by translation and rotation) from the source to the target position, without ever intersecting the "walls" of the environment?

To show that Motion Planning is 3-Sum-hard, employ the reduction for Segment Splitting from above. The three "punched" lines form the doorway between two rooms, each modeled by a constant number of segments that cannot be split, similar to the boundary gadgets above. The source position is in one room, the target position in the other, and to get from source to target the robot has to pass through a sequence of three collinear holes in the door (suppose the doorway is sufficiently small compared to the length of the robot).

Exercise 9.15 *The 3-Sum' problem is defined as follows: given three sets S_1, S_2, S_3 of n integers each, are there $a_1 \in S_1$, $a_2 \in S_2$, $a_3 \in S_3$ such that $a_1 + a_2 + a_3 = 0$? Prove that the 3-Sum' problem and the 3-Sum problem as defined in the lecture ($S_1 = S_2 = S_3$) are equivalent, more precisely, that they are reducible to each other in subquadratic time.*

9.7 Ham Sandwich Theorem

Suppose two thieves have stolen a necklace that contains rubies and diamonds. Now it is time to distribute the prey. Both, of course, should get the same number of rubies and the same number of diamonds. On the other hand, it would be a pity to completely disintegrate the beautiful necklace. Hence they want to use as few cuts as possible to achieve a fair gem distribution.

To phrase the problem in a geometric (and somewhat more general) setting: Given two finite sets R and D of points, construct a line that bisects both sets, that is, in either halfplane defined by the line there are about half of the points from R and about half of the points from D. To solve this problem, we will make use of the concept of levels in arrangements.

Definition 9.16 *Consider an arrangement $\mathcal{A}(L)$ induced by a set L of n non-vertical lines in the plane. We say that a point p is on the* **k-level** *in $\mathcal{A}(L)$ if there are at most $k-1$ lines below and at most $n-k$ lines above p. The 1-level and the n-level are also referred to as* **lower** *and* **upper envelope**, *respectively.*

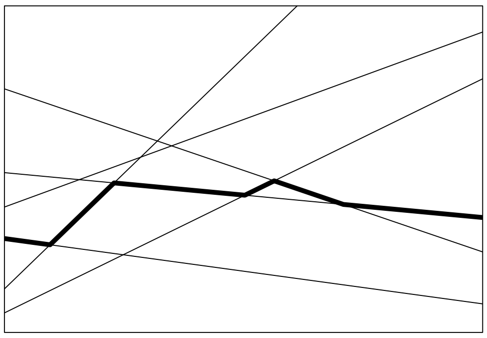


Figure 9.4: *The 3-level of an arrangement.*

Another way to look at the k-level is to consider the lines to be real functions; then the lower envelope is the pointwise minimum of those functions, and the k-level is defined by taking pointwise the k^{th}-smallest function value.