Chapter 10

Arrangements

During this course we encountered several situations where it was convenient to assume a point set "in general position". In the plane it usually means no three points are collinear and/or no four points are cocircular. This raises an algorithmic question: How do we test for n given points whether or not three of them are collinear?

The exact computational complexity of this innocent-looking problem is a major open problem in theoretical computer science. Obviously we could test all triples in $O(n^3)$ time, but can we do better? Yes, indeed! Via the so-called *projective duality transform*, we are able to solve the problem in $O(n^2)$ time, although we do not know if a better algorithm exists. Nevertheless, this transformation is interesting as it offers an equivalent *dual* perspective to a problem. Sometimes the dual form is easier to work with, and its solution can be efficiently translated back into a solution to the original or *primal* form.

So what is this transformation about? Recall that a hyperplane in $\mathbb{R}^d$ is the set of solutions $x \in \mathbb{R}^d$ to a linear equation $\sum_{i=1}^{d} h_i x_i = h_{d+1}$, where at least one of $h_1, \ldots, h_d$ is nonzero. If $h_d = 1$, we call the hyperplane *non-vertical*. Now observe that points and non-vertical hyperplanes in $\mathbb{R}^d$ can both be described by d real numbers. It is thus tempting to map them to each other. In $\mathbb{R}^2$, hyperplanes are lines and the standard **projective duality transform** maps a point $p = (a, b)$ to the non-vertical line $p^* : y = ax - b$, and a non-vertical line $\ell : y = ax + b$ to the point $\ell^* := (a, -b)$.

Proposition 10.1. *The standard projective duality transform is*

- *incidence preserving:* $p \in \ell \iff \ell^* \in p^*$ *and*
- *order preserving:* p *is above* $\ell \iff \ell^*$ *is above* p^*.

Exercise 10.2. *Prove Proposition 10.1.*

Exercise 10.3. *For each of the following point sets, what image do we get after applying the duality transform pointwise?*

(a) $k \geqslant 3$ *collinear points;*

(b) a line segment;

(c) a halfplane;

(d) the boundary points of the upper convex hull of a finite point set.

One can also visualize duality in terms of the parabola $\mathcal{P} : y = \frac{1}{2}x^2$. Let $p = (a, b)$ be any point. If it is on $\mathcal{P}$, then its dual line p^* is the tangent to $\mathcal{P}$ at p. Otherwise we consider its vertical projection $p' := (a, \frac{1}{2}a^2)$ onto $\mathcal{P}$. As we argued, $(p')^*$ is the tangent to $\mathcal{P}$ at p'. Note that the slopes of p^* and $(p')^*$ are the same, and p^* is just $(p')^*$ shifted vertically by $\frac{1}{2}a^2 - b$.

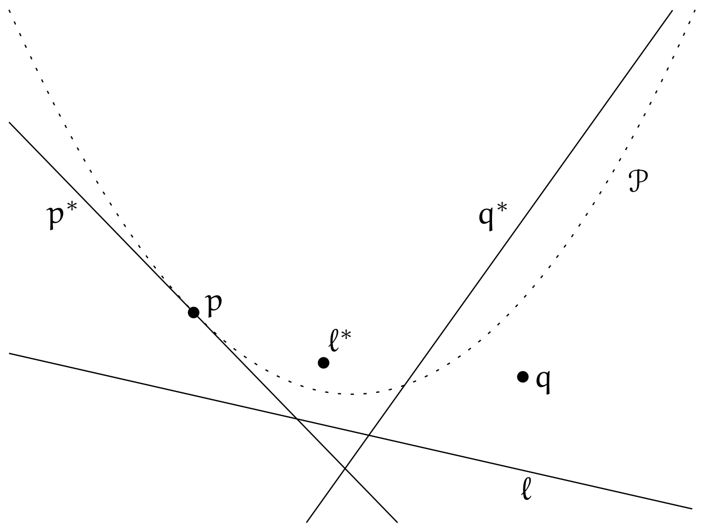


Figure 10.1: *Point $\leftrightarrow$ line duality through the lens of the parabola $\mathcal{P} : y = \frac{1}{2}x^2$.*

The problem of whether or not three points in the primal plane are collinear transforms to whether or not three lines in the dual plane meet in a common point. We will solve the latter problem with the help of *line arrangements*, as defined below.

10.1 Line Arrangements

The subdivision of the plane induced by a finite set L of lines is called the **line arrangement** $\mathcal{A}(L)$. We may imagine its creation as follows. First, from the plane $\mathbb{R}^2$ we subtract all lines in L (considered as point sets), thus breaking $\mathbb{R}^2$ into one or more open connected regions. Note that every region is an intersection of open halfplanes and hence convex. We call these regions the (2-dimensional) **cells** of the arrangement. Next, from each line $\ell \in L$ we subtract all the other lines, thus splitting ℓ into one or more open connected components. These collectively form the 1-dimensional cells or **edges**. What remains are the intersection points of lines from L. They are called the 0-dimensional cells or **vertices**.

The notion naturally generalizes to curve arrangements in $\mathbb{R}^2$ and hyperplane arrangements in $\mathbb{R}^d$. Without further specification, the word "arrangement" refers to line arrangements in this chapter. The **complexity** of an arrangement is just the total number of vertices, edges and cells (in general, the total number of cells of any dimension).

A line arrangement is called **simple** if no two lines are parallel and no three lines meet at a common point.

Theorem 10.4. *Every simple arrangement of* n *lines has* $\binom{n}{2}$ *vertices,* n^2 *edges, and* $\binom{n}{2}+n+1$ *cells.*

Proof. Since every pair of lines intersect and all the intersection points are distinct, there are $\binom{n}{2}$ vertices.

We count the number of edges by induction on n. For $n = 1$ we have $1^2 = 1$ edge. By adding a new line to an arrangement of $n-1$ lines, we split $n-1$ existing edges into two and also introduce n edges along the new line. So inductively there are $(n-1)^2 + (n-1) + n = n^2$ edges in total.

The number f of cells can be obtained from Euler's formula. For this we need to treat the arrangement as a planar graph $G = (V, E)$ by adding a vertex at "infinity" which absorbs all unbounded edges. Then the faces of G one-to-one correspond to the cells of the arrangement, with $|V| = \binom{n}{2} + 1$ and $|E| = n^2$. By Euler's formula we have $|V| - |E| + f = 2$. It follows that

$$f = 2 - |V| + |E| = 1 - \binom{n}{2} + n^2 = 1 + \binom{n}{2} + n. \qquad \square$$

So the complexity of a simple arrangement is $\Theta(n^2)$. Tweaking the above proof, it is easy to see that the complexity of *any* line arrangement is $O(n^2)$.

Exercise 10.5. *Consider a set of lines in* $\mathbb{R}^2$ *with no three meeting at a common point. Form a plane graph* G *whose vertices are the intersection points of the lines. Two vertices are adjacent if and only if they appear consecutively along some line. Prove that* G *is 3-colorable. That is, we can paint the vertices using at most three colors so that adjacent vertices receive different colors.*

10.2 Constructing Line Arrangements

How do we store a line arrangement in computers? Although some cells and edges are unbounded, we can effectively bound the arrangement by a sufficiently large box that cages all vertices. Such a box can be constructed in $O(n \log n)$ time for n lines.

Exercise 10.6. *How?*

Moreover, as we have seen in the previous proof, we can view the arrangement as a planar graph by adding a symbolic vertex that absorbs all (unbounded) edges leaving the box. For algorithmic purposes, we usually represent the graph by a doubly connected edge list (DCEL), cf. Section 2.2.1.

How do we construct an arrangement algorithmically? We would be satisified with any $O(n^2)$ algorithm, as the worst case complexity of line arrangements is quadratic already. A natural approach is incremental construction: just insert the lines one by one in some arbitrary order $\ell_1, \ldots, \ell_n$.

At Step i, suppose that the edges leaving the left side of the bounding box were ordered by slope. Hence we can locate in $O(i)$ time the leftmost cell F in $\mathcal{A}\{\ell_1, \ldots, \ell_{i-1}\}$ that ℓ_i intersects. Then we traverse the boundary of F counterclockwise, until we intersect ℓ_i again when walking on some halfedge h (see Figure 10.2 for illustration). Insert a new vertex at this intersection point, split F and h accordingly, and continue in the same way with the cell on the twin side of h.

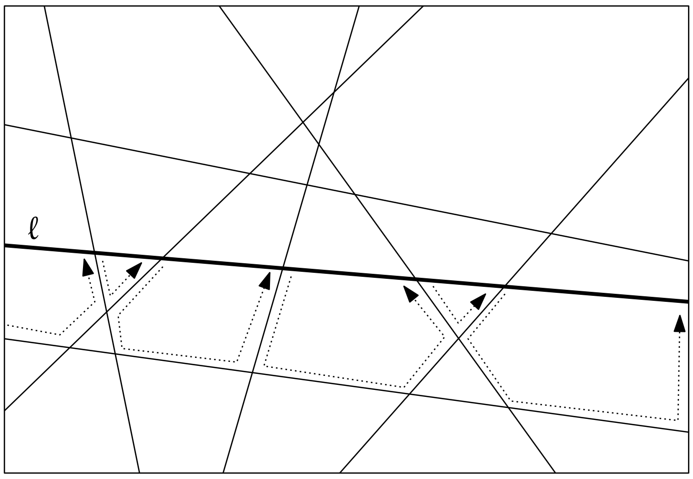


Figure 10.2: *Incremental construction: Insertion of a line ℓ. (Only part of the arrangement is shown in order to increase readability.)*

The insertion and splits are both constant time operations. But what is the time needed for the traversal? The worst case complexity of $\mathcal{A}\{\ell_1, \ldots, \ell_{i-1}\}$ is $\Omega(i^2)$, but not all cells and edges are relevant to our traversal. In fact, the relevant zone has linear complexity only, as we will show next.

10.3 Zone Theorem

For an arrangement $\mathcal{A}(L)$ and an arbitrary line ℓ (not necessarily from L), the **zone** $Z_{\mathcal{A}(L)}(\ell)$ is the set of cells from $\mathcal{A}(L)$ whose closure intersects ℓ.

Theorem 10.7. *Given an arrangement $\mathcal{A}(L)$ of n lines and a line ℓ, there are at most $10n$ edges in all cells of $Z_{\mathcal{A}(L)}(\ell)$.*

Proof. Fix the line ℓ and assume with loss of generality that it is horizontal (otherwise rotate the plane accordingly). For each cell, orient its horizontal edges from left to right, and the other edges from bottom to top. An oriented edge is *left-bounding* if the cell is to its right; otherwise it is *right-bounding*. Figure 10.3 gives an example.

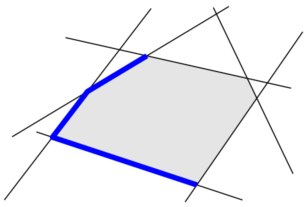


Figure 10.3: *Left-bounding edges (blue and bold) of a cell.*

We will show that there are at most $5n$ left-bounding edges in all cells of $Z_{\mathcal{A}(L)}(\ell)$ by induction on $n = |L|$. By symmetry, the same also holds for the number of right-bounding edges, thus the theorem follows.

For $n = 1$, there is at most $1 < 5n$ left-bounding edge in $Z_{\mathcal{A}(L)}(\ell)$. (The number could be zero because the only line in L might be horizontal and above ℓ.)

Now assume the statement is true for $n - 1$. If ℓ does not intersect with L, then all lines are horizontal and there is at most $1 < 5n$ left-bounding edge in $Z_{\mathcal{A}(L)}(\ell)$. Else let p be the rightmost point where ℓ intersects with L. We distinguish two cases:

Case 1: there is a unique line $r \in L$ **through** p. Consider $Z_{\mathcal{A}(L\setminus\{r\})}(\ell)$, namely the cells of the arrangement $\mathcal{A}(L \setminus \{r\})$ that touch ℓ. They have at most $5n-5$ left-bounding edges by the induction hypothesis. Now we add r back to the arrangement and see what happens. Let R be the rightmost cell in $Z_{\mathcal{A}(L\setminus\{r\})}(\ell)$. The line r intersects ∂R in at most two points and thus splits at most two edges of R (call them ℓ_0 and ℓ_1), both of which may be left-bounding. Further, the new edge $r \cap R$ is left-bounding for the rightmost cell R' of $Z_{\mathcal{A}(L)}(\ell)$. So locally the number of left-bounding edges increases by at most three.

Note that r does not contribute left-bounding edges to any cell of $Z_{\mathcal{A}(L)}(\ell)$ other than R': To any cell of $Z_{\mathcal{A}(L)}(\ell)$ that lies to the left of r, the line r can contribute right-bounding edges only; and any cell other than R' that lies to the right of r is shielded away from ℓ by ℓ_0 or ℓ_1, that is, it is not a cell in $Z_{\mathcal{A}(L)}(\ell)$. Therefore, the total number of left-bounding edges in $Z_{\mathcal{A}(L)}(\ell)$ is at most $(5n-5)+3 < 5n$.

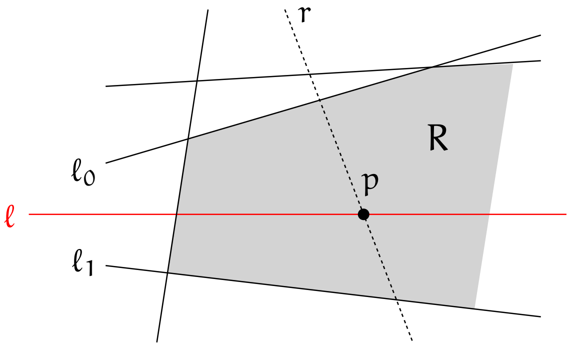


Figure 10.4: *At most three new left-bounding edges are created by adding* r *to* $\mathcal{A}(L\setminus\{r\})$.

Case 2: there are multiple lines through p. We add these lines to the arrangement of the remaining lines one by one. Adding the first line r creates at most three left-bounding edges by Case 1. From then on, adding another line r' that goes through p would create at most five left-bounding edges. We can argue as in Case 1: The line r' splits at most two left-bounding edges of R, and it also splits the left-bounding edge r of R'. Finally, the line r' itself provides two left-bounding edges (joined at the point p). Thus, the number of left-bounding edges increases by at most five for each additional edge. Hence the claim follows. □

Corollary 10.8. *The arrangement of* n *lines in* $\mathbb{R}^2$ *can be constructed in optimal* $\Theta(n^2)$ *time and space.*

Proof. Use the incremental construction described earlier. At Step $i = 1, \ldots, n$, we do a linear search among $i-1$ leftmost edges to find the starting cell 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 10.7 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 10.4. □

Generally in $\mathbb{R}^d$, a simple hyperplane arrangement has complexity $\Theta(n^d)$, and a zone of a hyperplane has complexity $O(n^{d-1})$.

Exercise 10.9. *For an arrangement* $\mathcal{A}$ *of a set of* n *lines in* $\mathbb{R}^2$*, let*

$$\mathcal{F} := \bigcup_{C \text{ is a bounded cell of } \mathcal{A}} \overline{C}$$

be 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* $O(n)$*.*

10.4 General Position and Minimum Triangle

The real beauty and power of line arrangements manifest through the projective duality. It is often convenient to assume that no two points in the primal plane have the same x-coordinate, so that no line through two points in the primal is vertical (and hence becomes an infinite point in the dual). This degeneracy can be discovered by sorting the points according to x-coordinate, and resolved by rotating the whole plane by a sufficiently small angle ε. To select ε, we can iterate over all non-vertical lines through two points, compute its (absolute) angle to verticality, and then choose ε to be strictly less than all these angles. The procedure clearly runs in $O(n^2)$ time.

Therefore, the following two 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 there three collinear points? (Dual: do three of the n dual lines meet at a common point?)

Minimum area triangle. Given a set $P \subset \mathbb{R}^2$ of n points, what is the minimum area triangle spanned by three distinct points from P? This can be viewed as a quantified version of general position test: the minimum area is zero if and only if there are three collinear points.

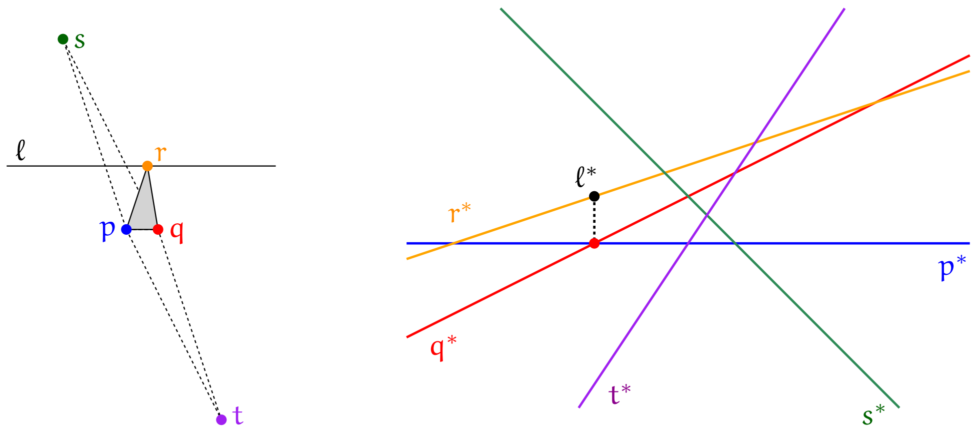
Let us make the problem easier by fixing two distinct points $p, q \in P$ and ask for a minimum area triangle pqr, where $r \in P \setminus \{p, q\}$. With pq fixed, the area of pqr is proportional to the distance between r and the line pq. Thus we want to find

a closest line ℓ parallel to pq and passing through some point $r \in P \setminus \{p, q\}$. $(\star)$

Consider the set $P^* := \{p^* : p \in P\}$ of dual lines and their arrangement $\mathcal{A}$. In $\mathcal{A}$ the statement $(\star)$ translates to

a closest point ℓ^* with the same x-coordinate as the vertex $p^* \cap q^*$ and lying on some line $r^* \in P^*$.

See Figure 10.5 for illustration. In other words, for the vertex $p^* \cap q^*$ of $\mathcal{A}$ we want to find a line $r^* \in P^*$ closest to it vertically—above or below. Of course, in the end we want this information not only for one particular vertex (which provides the minimum area triangle for fixed p, q) but for *all* vertices of $\mathcal{A}$, that is, for all possible pairs of fixed points $\{p, q\} \in \binom{P}{2}$.



(a) primal (b) dual

Figure 10.5: *Minimum area triangle spanned by two fixed points* p, q.

Luckily, such information can be maintained over the incremental construction of $\mathcal{A}$. When inserting a new line ℓ, it may become the vertically closest line from vertices of the already computed partial arrangement. However, only vertices in the zone of ℓ may be affected. The zone is traversed anyway, so in the same pass we can update the information for vertices immediately and vertically above or below ℓ, at no extra cost asymptotically.

By the end of the incremental construction, each vertex $p^* \cap q^*$ has recorded a vertically closest line, for $\{p, q\} \in \binom{P}{2}$. These correspond to minimum area triangles for every fixed p, q. The smallest such candidate can be found by simply comparing their areas, which takes $O(n^2)$ time.

Exercise 10.10. *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: The set* 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.*

Exercise 10.11. *Let* F *be a family of vertical line segments such that for each three of them, there exists a line that intersects all three. Show that there exists a line which intersects all line segments in* F.

10.5 Constructing Rotation Systems

Here is an application of line arrangements with a different flavor. Recall the notion of a combinatorial embedding from Chapter 2. It is specified by the circular order of boundary edges of each face. Equivalently, we may represent it by the circular order of edges around each vertex. This latter view is called a rotation system.

In a similar way we can also condense the geometry of a finite point set $P \subset \mathbb{R}^2$ combinatorially. For a point $q \in P$ let $c_P(q)$ denote the circular sequence of points from $P \setminus \{q\}$ around q (that is, in the order as they would be encountered by a ray sweeping around q). The **rotation system** of P is nothing but $\{c_P(q) : q \in P\}$.[1]

Given a set P of n points, it is trivial to construct its rotation system in $O(n^2 \log n)$ time, by sorting each of the n lists independently. But in fact we can do it optimally in $O(n^2)$ time by duality transform.

Consider a directed line sweeping counterclockwise around a point $q \in P$ in the primal plane, initially vertically downward. The slope increases from $-\infty$ to ∞ until the line becomes vertically upward. This finishes the right half of the circular sweep. The left half is similar: the slope changes from $-\infty$ to ∞, until the line completes a full circle.

In the dual plane, this corresponds to traversing the line q^* from $-\infty$ to ∞ twice. (The sweeping line ℓ always goes through the point q in the primal, so ℓ^* always sits on q^* in the dual, and its x-coordinate reflects the slope of ℓ.) Hence we may retrieve the circular sequence $c_P(q)$ by two passes in the dual arrangement: In the first (respectively the second) traversal of q^* we record the sequence of intersections with p^* for all $p \in P$

[1]You may also think of it as the "combinatorial embedding" of the complete geometric graph on P. But as these graphs are not planar for $|P| \geqslant 5$, they do not have the notion of faces.

to the right (respectively left) of q. The circular sequence is exactly the concatenation of the two. Clearly, the traversals along all lines in the dual arrangement can be done in $O(n^2)$ time.

10.6 Segment Endpoint Visibility Graphs

In this section we present a more complex application of duality and line arrangements, in the context of motion planning. Here a fundamental problem is to find a short(est) path between two given positions in some domain, subject to certain constraints. As an example, suppose we are given two points $p, q \in \mathbb{R}^2$ and a set $S \subset \mathbb{R}^2$ that models obstacles. What is the shortest path between p and q that avoids S?

Observation 10.12. *Let S be a finite set of polygonal obstacles. The shortest path between two points that avoids S, if it exists, is a polygonal path whose vertices (except the two ends) are obstacle vertices.*

A simplest type of obstacles conceivable is a line segment. In general they might separate the two query points, say when they form a closed curve that surrounds one of the points. However, if we require the obstacles to be pairwise disjoint line segments then there is always a free path between any query points. Hence by the above observation we may restrict our attention to straight line edges connecting obstacle vertices—in our case, segment endpoints.

Definition 10.13. *Consider a set S of n disjoint line segments in $\mathbb{R}^2$. The* **segment endpoint visibility graph** *$\mathcal{V}(S)$ is a geometric graph defined on the segment endpoints. Two segment endpoints p and q are adjacent if they see each other; that is, if*

- *the line segment $\overline{pq}$ is in S, or*
- *$\overline{pq} \cap s \subseteq \{p, q\}$ for all $s \in S$.*

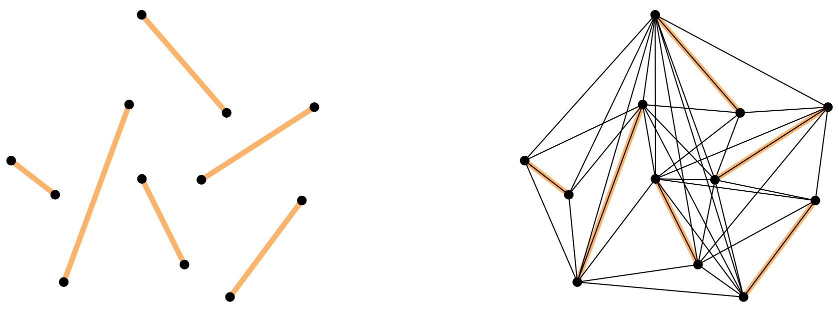


Figure 10.6: *A set of disjoint line segments and their endpoint visibility graph.*

If all segments are on the convex hull boundary, then the visibility graph is complete. If the segments form parallel chords of a convex polygon, then the visibility graph consists of copies of K_4 glued together side by side, and the number of edges is linear only.

On another note, these graphs also arise in the context of the following question: Given a set of disjoint line segments, can we link them at their endpoints via additional segments to form a closed Jordan curve? This is not always possible: Just consider three parallel chords of a convex polygon (Figure 10.7a). However, if we do not insist that all segments appear in the curve, but allow them to be diagonals or epigonals of the curve, then it is always possible [11, 12]. To rephrase: the segment endpoint visibility graph of disjoint line segments is Hamiltonian (unless all segments are collinear). It is actually essential to allow epigonals and not only diagonals [9, 20] (Figure 10.7b).

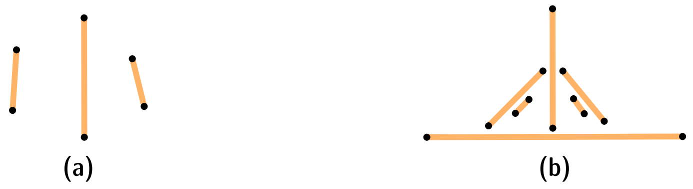


Figure 10.7: *Sets of disjoint line segments that do not admit certain polygons.*

Back to motion planning, our problem essentially reduces to finding a shortest path in the visibility graph $\mathcal{V}(S)$. But first of all, we need to construct the graph. Doing it in brute force takes $O(n^3)$ time where n denotes the number of segments in S. (Take all pairs of endpoints and check all other segments for obstruction.)

Theorem 10.14 (Welzl [21]). *The segment endpoint visibility graph of n disjoint line segments can be constructed in worst case optimal $O(n^2)$ time.*

Proof. Let P be the set of endpoints of S. As before we assume general position, so that no three points in P are collinear and no two have the same x-coordinate. (One can handle such degeneracy explicitly.)

Conceptually we perform a *rotational sweep*. That is, we rotate a direction vector v, initially pointing vertically downwards, in a counterclockwise fashion until it points vertically upwards. While rotating, we maintain for each point $p \in P$ the segment $s(p)$ that it "sees" in direction v (if any). Figure 10.8 shows an example. If v is parallel to some segment $\overline{pq}$, say pointing in the direction of $\overrightarrow{pq}$, then we assign $s(p) := \overline{pq}$.

During the sweep, we can output the edges of the visibility graph on the fly: If currently q is an endpoint of the segment $s(p)$, and v is in the direction of $\overrightarrow{pq}$, then we output the edge $\{p, q\}$.

Why does it work? Every output edge $\{p, q\}$ must be in $\mathcal{V}(S)$ because p sees $s(p) \ni q$ by definition. Conversely, let $\{p, q\}$ be an edge in $\mathcal{V}(S)$ where p is to the left of q, say. Assume that q is an endpoint of segment s. When v sweeps over the direction of $\overrightarrow{pq}$, we must have $s(p) = s$ and the output condition is met.

To perform the actual sweep, we first observe that changes of visible segments can only occur discretely. Indeed, for any point $p \in P$, the segment $s(p)$ can only change when p sees some point $q \in P$ exactly in direction v. Hence, we only need to consider directions $\overrightarrow{pq}$ with $p, q \in P$.

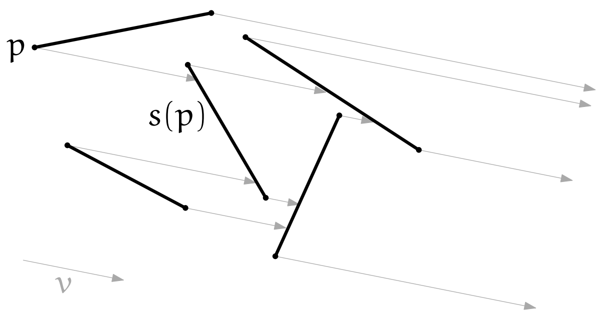


Figure 10.8: *Visible segments along a rotating direction v. The arrow from an endpoint p indicates the segment $s(p)$ it sees; if the arrow extends to infinity, then no segment is visible from p.*

Hence let us call a pair $(p, q) \in \binom{P}{2}$, q to the right of p, an *event*. Its *slope* is the slope of the line through p and q. Then the sweep can be implemented as follows: process the events in order of increasing slope (by general position, all slopes are distinct); whenever we get to process an event (p, q), there are four cases (Figure 10.9):

(1) p and q belong to the same input segment $\implies$ output the edge $\{p, q\}$.

(2) q is obscured from p by $s(p)$ $\implies$ no change.

(3) q is an endpoint of $s(p)$ $\implies$ output $\{p, q\}$ and update $s(p)$ to $s(q)$.

(4) q is an endpoint of a segment s that now obscures $s(p)$ $\implies$ output $\{p, q\}$ and update $s(p)$ to s.

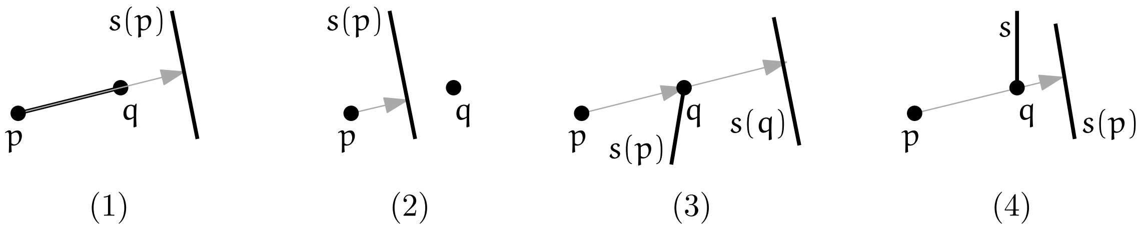


Figure 10.9: *Processing an event during the rotational sweep. Arrows indicate the segment $s(p)$ just before the event.*

What is the runtime of this rotational sweep? We have $O(n^2)$ events, and sorting them in increasing slope takes $O(n^2 \log n)$ time. After this, the actual sweep takes $O(n^2)$ time, as each event is processed in constant time.

To get rid of the $O(\log n)$ factor—we promised an $O(n^2)$ algorithm—we replace the sweep by a *topological* sweep, based on the observation that we do not strictly need to proceed by increasing slope. We just need property (a) below in order to correctly handle cases (1)(2)(4), while property (b) takes care of case (3).

(a) For each fixed $p \in P$, the events (p, q) are processed in order of increasing slope.

(b) When processing event (p, q), we have already processed the events (q, r) of smaller slope but not any event (q, r) of larger slope.

Any order of events that satisfies properties (a) and (b) will work. We can easily come by such an order when we interpret these properties in the arrangement $\mathcal{A}(P^*)$, where $P^* := \{p^* : p \in P\}$ is the projective dual of P. Recall that the slope of a line through two points $p, q \in P$ corresponds to the x-coordinate of the intersection of the dual lines p^*, q^*. Moreover, q is to the right of p if and only if q^* has larger slope than p^*. Hence, events in the dual are pairs of lines (p^*, q^*) where q^* has larger slope than p^*. The x-coordinate of an event is defined as the x-coordinate of the arrangement vertex $p^* \cap q^*$. Then, properties (a) and (b) translate to

(a*) For each fixed $p^* \in P^*$, the events (p^*, q^*) are processed in increasing x-coodinate.

(b*) When processing event (p^*, q^*), we have already processed the events (q^*, r^*) of smaller x-coordinate but not any event (q^*, r^*) of larger x-coodinate.

As the dual events correspond to arrangement vertices, properties (a*) and (b*) essentially say that if vertex u is to the left of vertex v *on the same line*, then u should appear before v. This notion coincides with the *topological order* on the *directed* arrangement graph with all edges directed from left to right. Clearly this directed graph is acyclic, so a topological order exists and can be computed, for instance, via (reversed) post order DFS in time linear in the size of the graph, which in our case is $O(n^2)$. □

Although the topological sweep is easy, the reader may ask whether the real sweep is also possible in $O(n^2)$ time. In other words: given a set of n points, is it possible to sort the $\binom{n}{2}$ lines defined by the points according to slope in $O(n^2)$ time? Standard lower bounds for sorting do not apply, since the $\binom{n}{2}$ slopes are highly interdependent. The answer is unknown, and according to Exercise 10.15, the problem is at least as hard as another, rather prominent, open problem.

Exercise 10.15. *The* $X+Y$ *sorting problem is the following: given two sets* X *and* Y *of* n *distinct numbers each, sort the set* $X + Y = \{x + y : x \in X,\ y \in Y\}$. *It is an open problem whether this can be done with* $o(n^2 \log n)$ *comparisons.*

Prove that $X + Y$ *sorting reduces (in* $O(n)$ *time) to the problem of sorting the* $\binom{2n}{2}$ *lines* pq *by slope, for all pairs* $\{p, q\}$ *from a set of* $2n$ *points.*

10.7 3-Sum

We have seen a quadratic time algorithm to test whether a point set is in general position (no three points on a common line). But as we mentioned, the exact computational complexity of this problem is unsettled. In fact, it is closely related to an abstract problem called **3-Sum**, in the sense that resolving the complexity of one of the problems also resolves the complexity of the other. However, doing so is one of the major open problems in theoretical computer science.

The 3-Sum problem is the following: Given a set S of n integers, is there a triple[2] of elements from S that sum up to zero? Obviously, by testing all triples this can be solved in $O(n^3)$ time. If we pick the triples to be tested more cleverly, we obtain an $O(n^2)$ algorithm. To this end, we sort the elements from S in increasing order and obtain the sequence $s_1 < \ldots < s_n$. This takes $O(n \log n)$ time. Then we test the triples as follows.

```
For $i = 1, \ldots, n$ {
    $j = i$, $k = n$.
    While $k \geqslant j$ {
        If $s_i + s_j + s_k = 0$ then output the triple $s_i, s_j, s_k$ and stop.
        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 the following invariant at the start of every iteration of the inner loop: $s_i + s_x + s_k < 0$ for all $x \in \{i, \ldots, j-1\}$, and $s_i + s_j + s_x > 0$ for all $x \in \{k+1, \ldots, n\}$.

Interestingly, until recently this was the fastest algorithm known for 3-Sum. But at FOCS 2014, Grønlund and Pettie [8] presented a deterministic algorithm that solves 3-Sum in $O(n^2 (\log \log n / \log n)^{2/3})$ time.

They also gave an upper bound of $O(n^{3/2}\sqrt{\log n})$ on the decision tree complexity of 3-Sum, which since then has been further improved in a series of papers. The latest improvement is due to Kane, Lovett, and Moran [13] who showed that $O(n \log^2 n)$ linear queries suffice (each query amounts to asking for the sign of the weighted sum of six numbers in S, with coefficients in $\{-1, 0, 1\}$). In this decision tree model, only queries that involve the input numbers are counted, all other computation, for instance, using these query results to analyze the parameter space are for free. So the results demonstrate that the (supposed) hardness of 3-Sum does not originate from the complexity of the decision tree.

The big open question remains whether an $O(n^{2-\varepsilon})$ algorithm can be achieved. Only in some very restricted models of computation—such as the 3-linear decision tree model where a decision can only be based on the sign of a linear expression in 3 input variables—it is known that 3-Sum requires quadratic time [6].

There is a whole class of problems that are equivalent to 3-Sum up to sub-quadratic time reductions [7]; such problems are referred to as **3-Sum-hard**.

Definition 10.16. *A problem X is* **3-Sum-hard** *if every 3-Sum instance of size n reduces to solving a constant number of X instances of size $O(n)$, within $O(n^{2-\varepsilon})$ reduction time for some constant $\varepsilon > 0$.*

[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.

Exercise 10.17. *Show that the following variation of 3-Sum—call it 3-Sum°—is 3-Sum-hard: Given a set S of n integers, are there three (distinct) elements of S that sum up to zero?*

As another example, consider the problem **Geometry Base**: Given n points on the horizontal lines $y = 0$, $y = 1$, and $y = 2$, is there a non-horizontal line that goes through at least three points?

Given any 3-Sum instance $S = \{s_1, \ldots, s_n\}$, we can reduce it to a Geometry Base instance P of size $3n$: For each s_i we create three points $(2s_i, 0)$, $(-s_i, 1)$ and $(2s_i, 2)$ in P. Now, if there is a non-horizontal line through three points in P, then each level contributes one, and so the three collinear points have form $p = (2s_i, 0)$, $q = (-s_j, 1)$ and $r = (2s_k, 2)$. The inverse slopes of lines pq and qr must be equal, hence $\frac{-s_j-2s_i}{1-0} = \frac{2s_k+s_j}{2-1}$, or $s_i + s_j + s_k = 0$. Conversely, if there is a triple $s_i + s_j + s_k = 0$, then by a reverse argument we see that the points $(2s_i, 0), (-s_j, 1), (2s_k, 2) \in P$ are collinear.

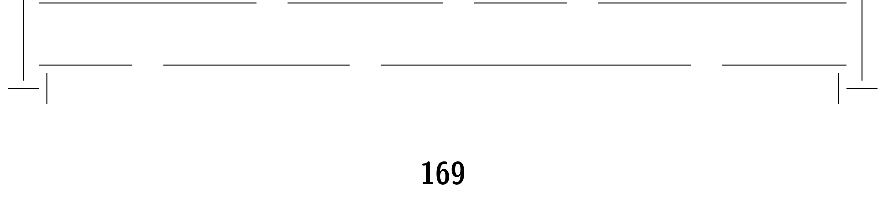
A very similar problem is **General Position** that we studied earlier. For a 3-Sum° instance S, we create a General Position instance P by lifting the numbers onto the curve $y = x^3$, that is $P := \{(a, a^3) \mid a \in S\}$. Three distinct points $(a, a^3), (b, b^3), (c, c^3) \in P$ are collinear if and only if the slopes of the lines through each pair are equal; that is

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

Hence P has a solution if and only if S has a solution. Note that the "if" part needs a, b, c to be distinct, and this is why we reduce from 3-Sum°.

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

In **Segment Separation**, we are given a set of n line segments and have to decide whether there exists a line that does not intersect the segments but separates them into two non-empty subsets. To show that this problem is 3-Sum-hard, we can reduce from Geometry Base, where we emulate the points along the three levels $y = 0$, $y = 1$, and $y = 2$ by punching sufficiently small "holes". The resulting segments form the Segment Separation instance. Horizontal splits can be prevented by constant size gadgets next to the left and right ends:



Constructing such an instance requires sorting the points along each of the three levels, which can be done in $O(n \log n) = O(n^{2-\varepsilon})$ time. It remains to specify how "sufficiently small" are those holes. As all input numbers are integers, it is not hard to show that punching a hole of range $\pm 1/4$ around each point is small enough.

In **Segment Visibility**, we are given a set S of n horizontal line segments and two specific $s_1, s_2 \in S$. The question is: Are there two points, $p_1 \in s_1$ and $p_2 \in s_2$ that can see each other, that is, $\text{relint}(\overline{p_1 p_2})$ does not intersect any segment from S? The reduction from Geometry Base is basically the same as for Segment Separation, just put s_1 above and s_2 below the three levels.

In **Motion Planning**, we are given a robot (modeled as a line segment), some obstacles (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 obstacles? To show that Motion Planning is 3-Sum-hard, employ the reduction from Geometry Base as above. The holes on the three punched lines form the doorways from "the top room" to "the bottom room". Each room is surrounded by walls on the outer sides, so a robot in the top room can only reach the bottom room through the doorways. By specifying a long enough robot, the only way that it can pass is via three collinear holes.

Exercise 10.18. *The 3-Sum' problem asks: 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 3-Sum' and 3-Sum are equivalent; more precisely, that they are reducible to each other in subquadratic time.*

10.8 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 in $\mathbb{R}^2$, how do we find a line that bisects both sets? This means that each side of the line contains half of the points from R and half of the points from D. To solve this problem, we will make use of the concept of *levels* in arrangements.

Definition 10.19. *Consider an arrangement $\mathcal{A}(L)$ of 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.*

Another way to look at the k-level is to consider the lines as real functions; then the

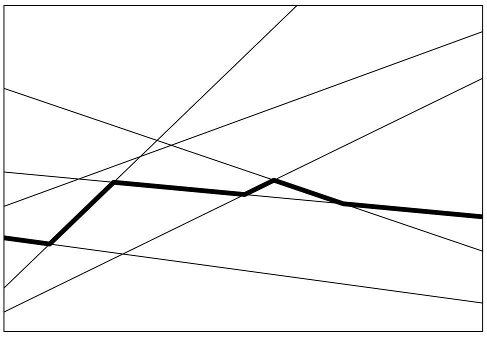


Figure 10.10: *The 3-level of an arrangement.*

lower/upper envelope is the pointwise minimum/maximum of those functions, and the k-level is defined by taking pointwise the k^{th}-smallest function value.

Theorem 10.20. *Let $R, D \subset \mathbb{R}^2$ be finite sets of points. Then there exists a line that bisects both R and D. That is, in either open halfplane bounded by ℓ, there are no more than $|R|/2$ points from R and no more than $|D|/2$ points from D.*

Proof. Without loss of generality suppose that both $|R|$ and $|D|$ are odd. (If, say, $|R|$ is even, simply remove an arbitrary point from R. Any bisector for the resulting set is also a bisector for R.) We may also suppose that no two points from $R \cup D$ have the same x-coordinate; otherwise we rotate the plane infinitesimally.

Let R^* and D^* denote the set of lines dual to the points from R and D, respectively. Consider the arrangement $\mathcal{A}(R^*)$. Every point on the median level of $\mathcal{A}(R^*)$ corresponds to a primal line that bisects R. As $|R^*| = |R|$ is odd, both the leftmost and the rightmost segments of the median level are contributed by the same line $\ell_r \in R^*$: the one with median slope. Similarly there is a line $\ell_d \in D^*$ that contributes to the leftmost and rightmost segments of the median level in $\mathcal{A}(D^*)$.

Since no two points from $R \cup D$ have the same x-coordinate, no two lines from $R^* \cup D^*$ have the same slope, and thus ℓ_r and ℓ_d intersect. Consequently, being piecewise linear continuous functions, the median level of $\mathcal{A}(R^*)$ and the median level of $\mathcal{A}(D^*)$ also intersect (see Figure 10.11 for an example). Any point that lies on this intersection corresponds to a primal line that bisects both point sets simultaneously. $\square$

How can the thieves use Theorem 10.20? If they are smart, they drape the necklace along some convex curve, say a circle. Then by Theorem 10.20 there exists a line that simultaneously bisects the set of diamonds and the set of rubies. As any line intersects the circle at most twice, they need to cut the necklace at only two points.

You can also think of the two point sets as a discrete distribution of a ham sandwich that is to be cut fairly, that is, in such a way that both parts have the same amount of ham and the same amount of bread. That is where the name "ham sandwich cut" comes from. The theorem generalizes both to higher dimension and to more general types of measures (here we study the discrete setting only where we simply count points). These

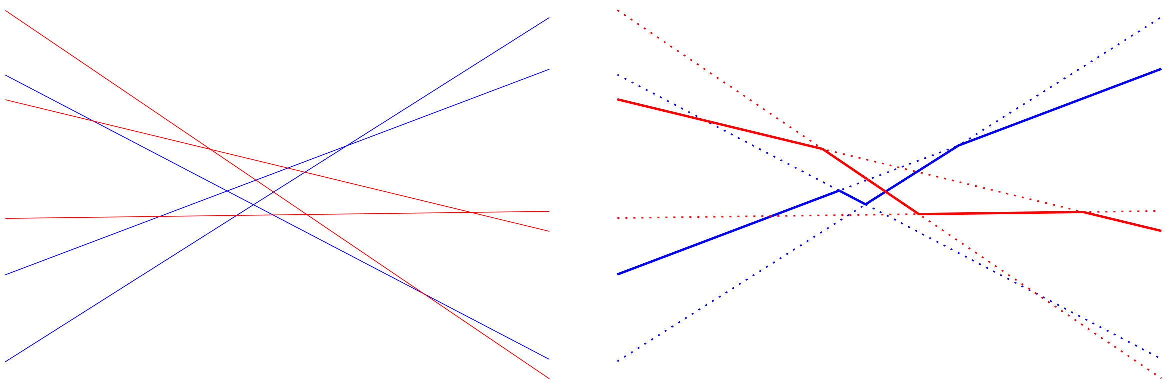


Figure 10.11: *An arrangement of 3 red lines and an arrangement of 3 blue lines, drawn in one picture. Their median levels are marked bold on the right.*

generalizations can be proven using the *Borsuk-Ulam Theorem*, which states that any continuous map from S^d to $\mathbb{R}^d$ must map some pair of antipodal points to the same point. For a proof of both theorems and many applications see Matoušek's book [17].

Theorem 10.21. *Let $P_1, \ldots, P_d \subset \mathbb{R}^d$ be finite sets of points. Then there exists a hyperplane h that simultaneously bisects all of $P_1, \ldots, P_d$. That is, in either open halfspace defined by h there are no more than $|P_i|/2$ points from P_i, for every $i \in \{1, \ldots, d\}$.*

This implies that the thieves can fairly distribute a necklace consisting of d types of gems using at most d cuts.

Exercise 10.22. *Prove or disprove the following statement: Given three finite sets A, B, C of points in the plane, there is always a circle or a line that bisects A, B and C simultaneously (that is, no more than half of the points of each set are inside or outside the circle or on either side of the line, respectively).*

Knowing about the existence of a ham sandwich cut certainly is not good enough. It is not hard to turn the proof given above into an $O(n^2)$ algorithm, but we can do better...

10.9 Constructing Ham Sandwich Cuts in the Plane

The algorithm outlined below is interesting not only in itself, but also because it illustrates a fundamental paradigm for designing optimization algorithms: *prune & search*. The basic idea behind prune & search is to exclude part of the search space from further consideration ("prune") at each step, and then search in the remaining space. A well-known example is binary search: every step takes constant time and discards about half of the possible solutions, resulting in a logarithmic runtime overall. As another example, if at each step some $1/d$ fraction of all potential solutions is discarded, and the step costs linear time cn in the number n of (remaining) solutions, then the runtime T satisfies

$$T(n) \leqslant cn + T\left(\left(1-\frac{1}{d}\right)n\right) < cn\sum_{i=0}^{\infty}\left(1-\frac{1}{d}\right)^{i} = cdn,$$

which gives a linear time performance.

Theorem 10.23 (Edelsbrunner and Waupotitsch [5]). *Let $R, D \subset \mathbb{R}^2$ be finite sets of points with $n = |R| + |D|$. Then in $O(n \log n)$ time one can find a line ℓ that simultaneously bisects R and D.*

Proof. We design a recursive algorithm $\mathrm{Find}(L_1, k_1;\ L_2, k_2;\ (x_1, x_2))$ that, for sets of lines L_1, L_2, non-negative integers k_1, k_2 and an open interval (x_1, x_2), finds an intersection between the k_1-level of $\mathcal{A}(L_1)$ and the k_2-level of $\mathcal{A}(L_2)$. It operates under two premises:

Distinct slopes: different lines in $L_1 \cup L_2$ have different slopes.

Odd-intersection: the two levels of interest should intersect an odd number of times in (x_1, x_2) and do not intersect at x_1 or x_2. Equivalently, the level above the other at x_1 should run below at x_2.

In the end, we are interested in $\mathrm{Find}\left(R^*, \frac{|R|+1}{2};\ D^*, \frac{|D|+1}{2};\ (-\infty, \infty)\right)$. Rotating the plane if necessary, we may assume without loss of generality that points in $R \cup D$ have distinct x-coordinates and thus lines in $R^* \cup D^*$ have distinct slopes. Also, as shown in the proof of Theorem 10.20, the odd-intersection property is satisfied.

Now we describe the algorithm. Let $L := L_1 \cup L_2$ and find the line $\mu \in L$ of median slope. Partition $L =: L_< \cup \{\mu\} \cup L_>$ depending on whether a line has slope less than or greater than the median. Pair the lines in $L_<$ with those in $L_>$ arbitrarily (with one line unpaired if $|L|$ is even). Denote by I the $\lfloor\frac{|L|-1}{2}\rfloor$ intersection points generated by the pairs. and let j be the median x-coordinate of these points.

Find the point on the k_1-level of $\mathcal{A}(L_1)$ at j, and the point on the k_2-level of $\mathcal{A}(L_2)$ at j. If the points coincide then we simply return it. Otherwise, if $j \in (x_1, x_2)$ then exactly one of the open intervals (x_1, j) or (j, x_2) satisfies the odd-intersection property, and we restrict our attention to it. If $j \notin (x_1, x_2)$, then we keep the original interval (x_1, x_2). In either case, we may assume by symmetry that the interval of interest is to the left of j.

In the following it is our goal to discard a constant fraction of the lines in L from future consideration. To this end, let $I_>$ denote the set of points from I with x-coordinate greater than j. Let μ' be a line parallel to μ that bisects $I_>$. The two lines $x = j$ and μ' subdivides the plane into four quadrants. By definition, each quadrant contains about a quarter of points from I; that is, about $|L|/8$ points.

As we assumed, the interval of interest is (x_1, t) for some $t \leqslant j$. In particular, exactly one of the left two quadrants Q is *interesting* in the sense that $((x_1, t) \times \mathbb{R}) \cap Q$ contains an odd number of intersections between the two levels. We will later argue how to algorithmically determine the interesting quadrant. For now, suppose that the upper

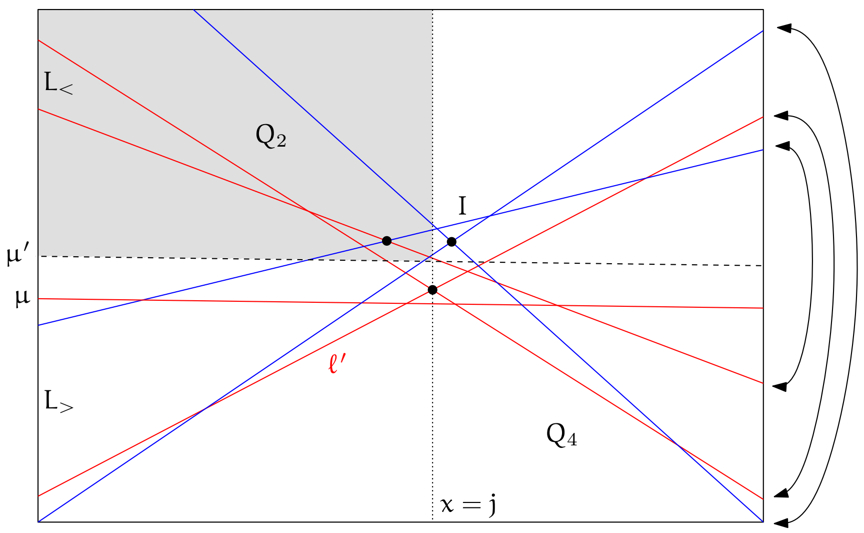


Figure 10.12: *An example with a set* L_1 *of* 4 *red lines and a set* L_2 *of* 3 *blue lines. Suppose that* $k_1 = 3$ *and* $k_2 = 2$*. Then the interesting quadrant is the top-left one (shaded) and the red line* ℓ' *(the line with the smallest slope in* L_1*) would be discarded because it does not intersect the interesting quadrant.*

left quadrant Q_2 is interesting (Figure 10.12). Consider its opposite (i.e. the lower right) quadrant Q_4. Any line in $L_>$ that passes through Q_4 is completely below the interesting quadrant Q_2, so all such lines can be safely discarded from further consideration. How many are they? Recall that every point in I is generated by paired lines, one from $L_>$ and the other from $L_<$. So there are at least $|I \cap Q_4| \approx |L|/8$ such lines.

After we discard these lines, we want to resume our search recursively. For every discarded line from L_1 (resp. L_2), we decrease the parameter k_1 (resp. k_2) by one. In the other case where Q_3 is interesting and we discard lines passing through Q_1, the parameters k_1 and k_2 stay the same. Denote the remaining sets of lines by L'_1 and L'_2, and the adjusted parameters by k'_1 and k'_2.

We want to apply the algorithm recursively to compute an intersection between the k'_1-level of $\mathcal{A}(L'_1)$ and the k'_2-level of $\mathcal{A}(L'_2)$. However, discarding lines changes the arrangement and its levels. As a result, it is not clear that the odd-intersection property holds for the k'_1-level of $\mathcal{A}(L'_1)$ and the k'_2-level of $\mathcal{A}(L'_2)$ on the interval (x_1, t). Note that we do know that these levels intersect in the interesting quadrant, and this intersection persists because none of the involved lines is removed. However, it is conceivable that the removal of lines changes the parity of intersections in the non-interesting quadrant of the interval (x_1, t). Luckily, this issue can be resolved as a part of the algorithm to determine the interesting quadrant, which we will discuss next. More specifically, we will show how to determine an interval $(x'_1, x'_2) \subseteq (x_1, x_2)$ on which the odd-intersection property holds for the k'_1-level of $\mathcal{A}(L'_1)$ and the k'_2-level of $\mathcal{A}(L'_2)$.

So let us argue how to determine the interesting quadrant, that is, how to test whether the k_1-level of $\mathcal{A}(L_1)$ and the k_2-level of $\mathcal{A}(L_2)$ intersect an odd number of times in $((x_1, t) \times \mathbb{R}) \cap H^+$, where H^+ is the open halfplane above μ'. For this it is enough to sweep the line μ' from left to right in the arrangement $\mathcal{A}(L)$ (conceptually) while keeping track of the relative ordering of the two levels and μ'. Initially at $x = x_1$, we know which level is above the other. Then we inspect all intersections between μ and L from left to right. If the current intersection point is on one of the two levels, then the relative ordering might change. So we check whether it is still consistent with that initial ordering. For instance, if the initial ordering is "the k_1-level above the k_2-level above μ'" and the k_1-level intersects μ' before the k_2-level does, then we can deduce that there must be an intersection of the two levels above μ'. As the relative position of the two levels is reversed at $x = x_2$, at some point an inconsistency, that is, the presence of an intersection will be detected and we will be able to tell whether it is above or below μ'. (There could be many more intersections between the two levels, but finding just one intersection is good enough.) Along with this above/below information we also obtain a suitable interval (x'_1, x'_2) for which the odd-intersection property holds because the levels of interest do not change in that interval.

The sweep amounts to computing all intersections of μ' with L, sorting them by x-coordinate, and keeping track of the number of lines from L_1 (resp. L_2) below μ'. At every point of intersection, these counters can be adjusted and the inconsistency can be tested in constant time. Overall it takes $O(|L| \log |L|)$ time. This step dominates the whole algorithm, noting that all other operations are based on rank-i element selection, which can be done in linear time [4]. Altogether, we obtain as a recursion for the runtime

$$T(n) \leqslant cn \log n + T(7n/8).$$

Solving it gives $T(n) = O(n \log n)$. □

In the plane, a ham sandwich cut can actually be found in linear time using a sophisticated prune and search algorithm by Lo, Matoušek and Steiger [16]. But in higher dimension, the algorithmic problem gets harder. In fact, already for $\mathbb{R}^3$ the complexity of finding a ham sandwich cut is wide open: The best algorithm known, from the same paper by Lo et al. [16], has runtime $O(n^{3/2} \log^2 n / \log^* n)$ and no non-trivial lower bound is known. If the dimension d is not fixed, it is both NP-hard and W[1]-hard[3] in d to decide the following question [15]: Given $d \in \mathbb{N}$, finite point sets $P_1, \ldots, P_d \subset \mathbb{R}^d$, and a point $p \in \bigcup_{i=1}^{d} P_i$, is there a ham sandwich cut through p?

Exercise 10.24. *The goal of this exercise is to develop a data structure for halfspace range counting.*

(a) *Given a set $P \subset \mathbb{R}^2$ of n points in general position, show that it is possible to partition this set by two lines such that each region contains at most $\lceil \frac{n}{4} \rceil$ points.*

[3] Essentially this means that it is unlikely to be solvable in time $O(f(d)p(n))$, for an arbitrary function f and a polynomial p.

(b) *Design a data structure of size* $O(n)$ *that can be constructed in time* $O(n \log n)$ *and allows you, for any halfspace* h*, to output the number of points* $|P \cap h|$ *of* P *contained in this halfspace* h *in time* $O(n^{\alpha})$*, for some* $0 < \alpha < 1$.

10.10 Davenport-Schinzel Sequences

The complexity of a simple arrangement of n lines in $\mathbb{R}^2$ is $\Theta(n^2)$, so every algorithm that uses the whole arrangement explicitly needs $\Omega(n^2)$ time. However, there are many scenarios where we only need a local part of the arrangement. For instance, if we only care about the cells touching a specific line, then the zone theorem ensures a linear complexity overall. As another example, to construct a ham sandwich cut for two sets in $\mathbb{R}^2$ we only need the median levels of their dual line arrangements. As mentioned in the previous section, the relevant information can actually be obtained in linear time.

Hence it can be rewarding to study the "local" complexity of arrangements. Here we pursue one such direction: to analyze the complexity—the number of vertices and edges—of a single cell in an arrangement of n *simple curves* in $\mathbb{R}^2$.

If the curves are lines then this is of little interest: Every line can appear at most once on the boundary of each cell; and it is possible that all lines appear on the boundary of one cell simultaneously.

But if the curves are line segments rather than lines, the situation changes in a surprising way. Certainly a single segment can appear several times along the boundary of a cell, see the example in Figure 10.13. Make a guess: What is the maximal complexity of a cell in an arrangement of n line segments in $\mathbb{R}^2$? You will find out the correct answer soon, although we only prove part of it. But my guess would be that it is rather unlikely that your guess is correct, unless, of course, you knew the answer already. :-)

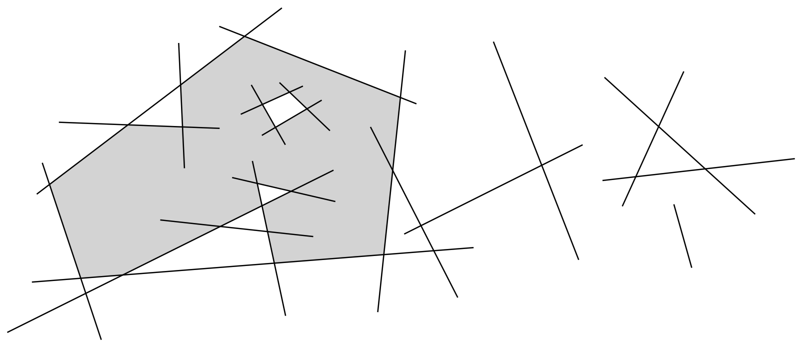


Figure 10.13: *A single cell in an arrangement of line segments.*

To start our program, we will focus on one particular cell in the curve arrangement: its lower envelope, or intuitively everything that can be seen vertically from below. To analyze the lower envelopes we will describe them combinatorially using sequences with forbidden substructures. These sequences, named after Davenport and Schinzel, are of independent interest as they appear in a number of combinatorial problems [2] and in the analyses of data structures [19]. The techniques apply not only to lower envelopes but also to arbitrary cells of curve arrangements.

Definition 10.25. *Let $n, s \in \mathbb{N}$. An (n, s)-***Davenport-Schinzel sequence** *is a sequence over an alphabet of n symbols, such that*

- *no adjacent slots have the same symbol, and*
- *no subsequence of length $s+2$ alternates between two different symbols.*[4]

Let $\lambda_s(n)$ be the length of a longest (n, s)-Davenport-Schinzel sequence.

So, for instance, an $(n, 2)$-Davenport-Schinzel sequence forbids alternations of the form $\ldots a \ldots b \ldots a \ldots b \ldots$. As another example, consider the sequence abcbacb. It is a $(3, 4)$-Davenport-Schinzel sequence but not a $(3, 3)$-Davenport-Schinzel sequence, because it contains a subsequence bcbcb of length 5 that alternates between b and c.

Note that the Davenport-Schinzel property is invariant of the alphabet in the following sense. If a sequence $\sigma_1, \ldots, \sigma_\ell$ over alphabet A is (n, s)-Davenport-Schinzel and $\varphi : A \to B$ is a bijection, then the sequence $\varphi(\sigma_1), \ldots, \varphi(\sigma_\ell)$ over alphabet B is again (n, s)-Davenport-Schinzel. In other words, we can change the alphabet whenever it is convenient.

Exercise 10.26. *Show that $\lambda_1(n) = n$ and $\lambda_2(n) = 2n - 1$.*

Exercise 10.27. *Prove that $\lambda_s(n)$ is finite for all $n, s \in \mathbb{N}$. Can you give explicit upper and lower bounds?*

Proposition 10.28. $\lambda_s(m) + \lambda_s(n) \leqslant \lambda_s(m+n)$.

Proof. Consider any (m, s)-Davenport-Schinzel sequence of length ℓ and (n, s)-Davenport-Schinzel sequence of length ℓ'. Assume without loss of generality that their alphabets are disjoint, so concatenating them yields a sequence of length $\ell + \ell'$ over an alphabet of $m+n$ symbols. It is straightforward to check that the sequence is $(m+n, s)$-Davenport-Schinzel. □

Let us now see how Davenport-Schinzel sequences are connected to lower envelopes. Consider a set $\mathcal{F} = \{f_1, \ldots, f_n\}$ of real-valued continuous functions that are defined on a common closed interval $I \subset \mathbb{R}$. The *lower envelope* $\mathcal{L}_\mathcal{F}$ of $\mathcal{F}$ is defined as the pointwise minimum of the functions. Formally, $\mathcal{L}_\mathcal{F}(x) := \min\{f_j(x) : 1 \leqslant j \leqslant n\}$ for $x \in I$.

Suppose that the graphs of any two functions in $\mathcal{F}$ intersect at finitely many points. Then each function can appear on the lower envelope $\mathcal{L}_\mathcal{F}$ only finitely many times. Imagine scanning the lower envelope from left to right, then the indices of functions that appear on it form the *lower envelope sequence* $\phi(\mathcal{F})$; see Figure 10.14 for an illustration.

Each intersection between the graphs of f_j and f_k can lead to at most one alternation between j and k in $\phi(\mathcal{F})$. Hence we have the following:

[4]Note that a subsequence need not be contiguous. For example, geo is a subsequence of congestion.

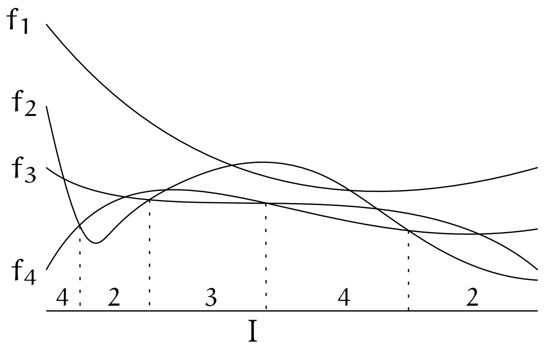


Figure 10.14: *The lower envelope sequence $(4, 2, 3, 4, 2)$ of a set of functions.*

Observation 10.29. *If the graphs of any two functions in $\mathcal{F}$ intersect at most s times, then $\phi(\mathcal{F})$ is an (n, s)-Davenport-Schinzel sequence.*

But in order to deal with line segments (understood as linear functions over interval domains), the above machinery needs slight extension because the segments may have different domains. So let us allow each function in $\mathcal{F}$ having an individual closed interval as its domain. The lower envelope $\mathcal{L}_\mathcal{F}$, now defined over the real line, is given by $\mathcal{L}_\mathcal{F}(x) := \inf\{f_j(x) : 1 \leqslant j \leqslant n$ and x is in the domain of $f_j\}$. In the case where no f_j is defined at x, we have $\mathcal{L}_\mathcal{F}(x) = \infty$.

Again assuming that the graphs of any two functions intersect at finitely many points, the lower envelope sequence $\phi(\mathcal{F})$ records the indices of the functions that appear on $\mathcal{L}_\mathcal{F}$ as we scan from left to right. By convention we use index 0 for intervals where $\mathcal{L}_\mathcal{F}(x) = \infty$; see Figure 10.15 for an illustration.

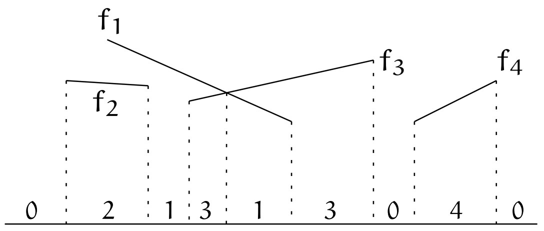


Figure 10.15: *The lower envelope sequence of a set of segments.*

In Figure 10.15 we already see that two segments f_j, f_k, despite crossing only once, may lead to a lower envelope subsequence $jkjk$ of length 4 (instead of 2 as it would be if they were defined over a common interval). Generally we have:

Proposition 10.30. *Let $\mathcal{F}$ be a collection of n real-valued continuous functions, each defined on some closed interval, and the graphs of any two functions intersect at most s times. Then $\phi(\mathcal{F})$ is an $(n + 1, s + 2)$-Davenport-Schinzel sequence.*

Proof. We first show that there is no subsequence of length $s+4$ that alternates between two symbols $j, k \in \{1, \ldots, n\}$. For this, we consider the at most s points at which f_j, f_k are equal, plus the at most 4 domain endpoints of f_j and f_k. These together subdivide the real line into at most $s+5$ intervals. Within each of the at most $s+3$ inner intervals, f_j is either always above or always below f_k by continuity; only the lower one may appear on $\mathcal{L}_\mathcal{F}$ (possibly zero or multiple times). Hence in $\phi(\mathcal{F})$, the symbols j, k cannot alternate within this interval. It follows that any j-k alternating subsequence in $\phi(\mathcal{F})$ has length most $s+3$.

Finally, we observe that $\phi(\mathcal{F})$ cannot contain subsequence $j0j$ for $j \in \{1, \ldots, n\}$, since the domain of f_j is an interval. Hence in particular there is no subsequence $0j0j$ or $j0j0$; that is, no subsequence of length 4 that alternates between symbols 0 and $j \in \{1, \ldots, n\}$. The statement then follows. □

So let us now focus on obtaining upper bounds for $\lambda_s(n)$. You have seen linear bounds for $s = 1, 2$ in Exercise 10.26. But the situation for $s = 3$ is more complicated.

Lemma 10.31. $\lambda_3(n) \leqslant 2n(1 + \log n)$.

Proof. For $n = 1$ we have $\lambda_3(1) = 1 \leqslant 2$. For $n > 1$ consider any $(n, 3)$-Davenport-Schinzel sequence σ of length $\lambda_3(n)$. Let a be a symbol that appears least frequently in σ; so it appears at most $\frac{1}{n}\lambda_3(n)$ times. Delete all appearances of a from σ to obtain a sequence σ' of length at least $(1-\frac{1}{n})\lambda_3(n)$ over $n-1$ symbols. But σ' is not necessarily a Davenport-Schinzel sequence because it might contain consecutive bb for some symbol b; this happens when $\sigma = \ldots bab \ldots$.

We claim that there are at most two places where such situation may arise. In fact, the only two possible places are around the first and last appearances of a. Indeed, if any intermediate appearance of a had resulted in consecutive bb after deletion, then $\sigma = \ldots a \ldots bab \ldots a \ldots$. So σ would contain the subsequence $ababa$, in contradiction to it being an $(n, 3)$-Davenport-Schinzel sequence.

Given the claim, one can repair σ' by deleting at most two characters from it. This produces an $(n-1, 3)$-Davenport-Schinzel sequence of length at least $(1-\frac{1}{n})\lambda_3(n) - 2$, which by definition cannot exceed $\lambda_3(n-1)$. So via rearranging we have the recursion

$$\frac{\lambda_3(n)}{n} \leqslant \frac{\lambda_3(n-1)}{n-1} + \frac{2}{n-1}.$$

Denoting $\overline{\lambda}_3(n) := \frac{\lambda_3(n)}{n}$, this means

$$\overline{\lambda}_3(n) \leqslant \overline{\lambda}_3(n-1) + \frac{2}{n-1} \leqslant \cdots \leqslant \overline{\lambda}_3(1) + \sum_{i=2}^{n} \frac{2}{i-1} = 1 + 2H_{n-1}$$

where H_{n-1} is the $(n-1)$-st harmonic number. Together with $2H_{n-1} < 1 + 2\log n$ we obtain $\lambda_3(n) \leqslant 2n(1 + \log n)$. □

Bounds for higher-order Davenport-Schinzel sequences. As we have seen, $\lambda_1(n)$ (no aba) and $\lambda_2(n)$ (no $abab$) are both linear in n. It turns out that for $s \geqslant 3$, $\lambda_s(n)$ is slightly superlinear in n (with s fixed). The bounds are known almost exactly, and they involve the inverse Ackermann function $\alpha(n)$, which grows extremely slowly.

To define the inverse Ackermann function, we first define a hierarchy of functions $\alpha_1(n), \alpha_2(n), \alpha_3(n), \ldots$ where $\alpha_k(n)$ grows much more slowly than $\alpha_{k-1}(n)$ for every fixed k.

We let $\alpha_1(n) = \lceil n/2 \rceil$. Then, for each $k \geqslant 2$, we define $\alpha_k(n)$ as the number of repeating applications of α_{k-1} on n until the value becomes no larger than 1. In other words, $\alpha_k(n)$ is defined recursively by:

$$\alpha_k(n) = \begin{cases} 0, & \text{if } n \leqslant 1; \\ 1 + \alpha_k(\alpha_{k-1}(n)), & \text{otherwise.} \end{cases}$$

Thus $\alpha_2(n) = \lceil \log_2 n \rceil$, and $\alpha_3(n) = \log^* n$.

Now fix n and consider the sequence $\alpha_1(n), \alpha_2(n), \alpha_3(n), \ldots$. The sequence decreases rapidly until it drops below 3. We define the inverse Ackermann function $\alpha(n)$ by

$$\alpha(n) = \min\{k : \alpha_k(n) \leqslant 3\}.$$

Exercise 10.32.

(a) Show that for every fixed $k \geqslant 2$ we have $\alpha_k(n) = o(\alpha_{k-1}(n))$; in fact, for every fixed k and t we have $\alpha_k(n) = o(\alpha_{k-1}(\alpha_{k-1}(\cdots\alpha_{k-1}(n)\cdots)))$, with t applications of α_{k-1}.

(b) Show that for every fixed k we have $\alpha(n) = o(\alpha_k(n))$.

Coming back to the bounds for Davenport-Schinzel sequences, for $\lambda_3(n)$ it is known that $\lambda_3(n) = \Theta(n\alpha(n))$ [10]. In fact we even know $\lambda_3(n) = 2n\alpha(n) \pm O(n\sqrt{\alpha(n)})$ [14, 18]. For $\lambda_4(n)$ we have $\lambda_4(n) = \Theta(n \cdot 2^{\alpha(n)})$ [3].

For higher-order sequences the known upper and lower bounds are almost tight, and they are of the form $\lambda_s(n) = n \cdot 2^{\text{poly}(\alpha(n))}$, where the degree of the polynomial in the exponent is roughly $s/2$ [3, 18].

Realizing Davenport-Schinzel sequences as lower envelopes. There exists a construction of a set of n segments in the plane whose lower-envelope sequence has length $\Omega(n\alpha(n))$. (In fact, the lower-envelope sequence has length $n\alpha(n) - O(n)$, with a leading coefficient of 1; it is an open problem to get a leading coefficient of 2, or prove that this is impossible.)

It is an open problem to construct a set of n parabolic arcs in the plane whose lower-envelope sequence has length $\Omega(n \cdot 2^{\alpha(n)})$.

Exercise 10.33. *Show that every (n, s)-Davenport-Schinzel sequence can be realized as the lower envelope of n continuous functions from $\mathbb{R}$ to $\mathbb{R}$, every pair of which intersect at most s times.*

Exercise 10.34. *Show that every $(n, 2)$-Davenport-Schinzel sequence can be realized as the lower envelope of n parabolas.*

Generalizations of Davenport-Schinzel sequences. Generalized Davenport-Schinzel sequences have also been studied, for instance, where arbitrary subsequences (not necessarily an alternating pattern) are forbidden. For a pattern π and $n \in \mathbb{N}$ define $\mathrm{Ex}(\pi, n)$ to be the maximum length of a sequence over $\{1, \ldots, n\}$ that does not contain a subsequence of the form π. For example, $\mathrm{Ex}(ababa, n) = \lambda_3(n)$. If π contains two meta-symbols only, say a and b, then $\mathrm{Ex}(\pi, n)$ is super-linear if and only if π contains $ababa$ as a subsequence [1]. This highlights that the alternating forbidden pattern is of particular interest.

10.11 Constructing Lower Envelopes

Theorem 10.35. *Let $\mathcal{F}$ be a collection of n real-valued continuous functions defined on a common closed interval. Assume that any two functions intersect at most s times, and that each intersection point can be constructed in constant time. Then the lower envelope $\mathcal{L}_{\mathcal{F}}$ can be constructed in $O(\lambda_s(n) \log n)$ time.*

Proof. Divide and conquer. For simplicity, assume that n is a power of two. Split $\mathcal{F}$ into two subsets $\mathcal{F}_1$ and $\mathcal{F}_2$ of $n/2$ functions each. Construct their lower envelopes $\mathcal{L}_{\mathcal{F}_1}$ and $\mathcal{L}_{\mathcal{F}_2}$ recursively. After they return, we merge them by a scan from left to right. At coordinate x, the defining function of $\mathcal{L}_{\mathcal{F}}$ can change only when (1) the defining function of $\mathcal{L}_{\mathcal{F}_1}$ changes; (2) the defining function of $\mathcal{L}_{\mathcal{F}_2}$ changes; or (3) $\mathcal{L}_{\mathcal{F}_1}$ and $\mathcal{L}_{\mathcal{F}_2}$ intersect. So the scan concerns with discrete events only.

There are at most $\lambda_s(n/2)$ events of type (1); and the same holds for type (2). (Note, however, that they do not necessarily guarantee a change for $\mathcal{L}_{\mathcal{F}}$.) Events of type (3) are different: every occurrence does contribute to a change for $\mathcal{L}_{\mathcal{F}}$, thus we can upper bound the number of occurrences by the complexity of $\mathcal{L}_{\mathcal{F}}$, namely $\lambda_s(n)$. So the total number of events is at most $2\lambda_s(n/2) + \lambda_s(n) \leqslant 2\lambda_s(n)$ where we used Proposition 10.28.

Now that the scan costs time linear in $\lambda_s(n)$, we obtain a recursion for the runtime $T(n)$ of the algorithm: $T(n) \leqslant 2T(n/2) + c\lambda_s(n)$ for some constant c. Therefore, $T(n) \leqslant c\sum_{i=1}^{\log n} 2^i \lambda_s(n/2^i) \leqslant c\sum_{i=1}^{\log n} \lambda_s(n) = O(\lambda_s(n) \log n)$ where the second inequality is by Proposition 10.28. □

10.12 Complexity of a Single Cell

With all tools gathered, we can now go back to the initial question: What is the complexity of a single cell in an arrangement of n simple curves?

Theorem 10.36. *Let $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ be a collection of simple curves in $\mathbb{R}^2$ such that each pair intersects at most s times. Then the combinatorial complexity of any single cell in the arrangement $\mathcal{A}(\Gamma)$ is $O(\lambda_{s+2}(n))$.*

Proof. Consider a cell f of $\mathcal{A}(\Gamma)$. In general, ∂f might consist of multiple connected components. But as every γ_i can appear in at most one component, we may deal with each component separately and sum up their complexity by Proposition 10.28. Hence it is no loss of generality to assume that ∂f is connected.

Replace each γ_i by two opposite directed curves γ_i^+ and γ_i^- that together form a closed curve that is infinitesimally close to γ_i. Denote by S the circular order that these directed curves appear along a clockwise traversal of ∂f. See Figure 10.16 for an illustration.

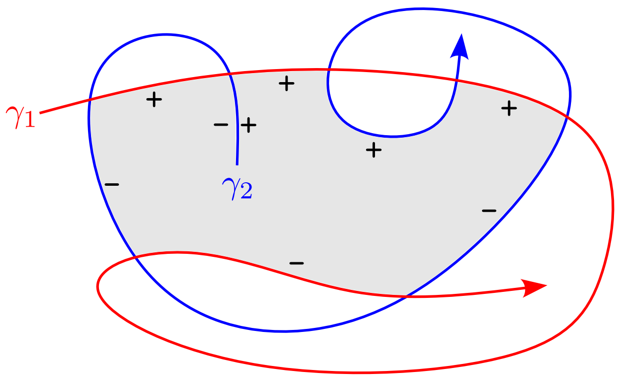
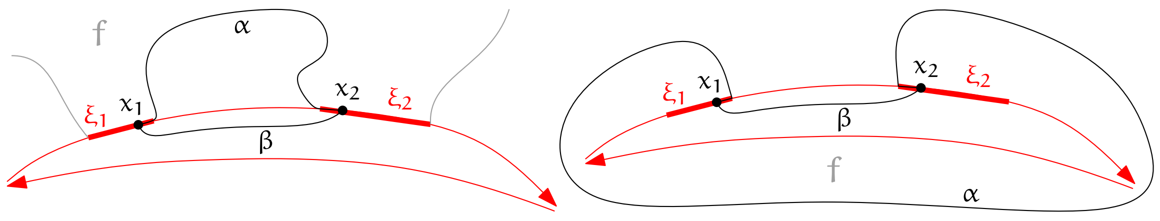


Figure 10.16: *An arrangement of two simple curves γ_1, γ_2. The arrows indicate the positive curves γ_1^+, γ_2^+, and their reversals are the negative curves γ_1^-, γ_2^-. Traversing the boundary of the shaded cell clockwise from the top-left corner, we encounter $S = (\gamma_1^+, \gamma_2^-, \gamma_2^+, \gamma_1^+, \gamma_2^+, \gamma_1^+, \gamma_2^-, \gamma_1^-, \gamma_2^-)$.*

Consistency Lemma. Fix $i \in \{1, \ldots, n\}$, and let ξ be one of the directed curves γ_i^+ or γ_i^-. The order of portions of ξ that appear in S is consistent with their order along ξ. Hence we can break the circular order S into a linear sequence $S(\xi)$ that, as we scan it from left to right, the portions of ξ appear in their order along ξ.

Consider two portions ξ_1 and ξ_2 of ξ that appear consecutively in S (that is, there is no other portion of ξ in between). Choose points $x_1 \in \xi_1$ and $x_2 \in \xi_2$ and connect them via two curves: a curve α following ∂f clockwise, and a curve β sandwiched between γ_i^+ and γ_i^-. Then α and β do not intersect internally and they are both contained in $\mathbb{R}^2 \setminus f^\circ$. In other words, $\alpha \cup \beta$ forms a closed Jordan curve and f lies either in its interior or in its exterior (Figure 10.17). In either case, the part of ξ between ξ_1 and ξ_2 is shielded away from f by $\alpha \cup \beta$ and, therefore, no point from this part can appear anywhere along ∂f. In other words, the boundary portions ξ_1 and ξ_2 are also consecutive along ξ, which proves the lemma.

Let us break the circular order S into a linear sequence $S' = (s_1, \ldots, s_t)$ arbitrarily. Now we traverse a directed curve ξ, paying attention to its portions that appear in ∂f. By the Consistency Lemma, these portions correspond to $(s_{i_1}, \ldots, s_{i_\ell}, s_{i_{\ell+1}}, \ldots, s_{i_r})$, where $1 \leqslant i_{\ell+1} < \cdots < i_r < i_1 < \cdots < i_\ell \leqslant t$. This "wrap-around" issue is caused by our breaking the circular S into a linear S'. As a precaution, we replace all occurrences of ξ



(a) f lies in the exterior of $\alpha \cup \beta$. **(b)** f lies in the interior of $\alpha \cup \beta$.

Figure 10.17: *Two cases in the proof of Consistency Lemma.*

in S' after index i_ℓ by a brand new symbol ξ'. Doing so for all directed curves results in a sequence S^* over $4n$ symbols. With this trick, any subsequence $\xi\xi\ldots\xi$ (as well as $\xi'\xi'\ldots\xi'$) of S^* will agree with the traversal order of directed curve ξ.

Claim. S^* is a $(4n, s+2)$-Davenport-Schinzel sequence.

Clearly no two adjacent symbols in S^* are the same. Suppose S^* contains a subsequence σ of length $s+4$ that alternates between ξ and η. For any occurrence of ξ in σ, choose a point from the corresponding part of ∂f. This gives a sequence $x_1, \ldots, x_{\lceil (s+4)/2 \rceil}$ of points on ∂f. Connect them in this order by a curve $C(\xi)$ sandwiched between ξ and its counterpart. (This is possible thanks to our precaution.) Similarly we may choose points $y_1, \ldots, y_{\lfloor (s+4)/2 \rfloor}$ on ∂f that correspond to the occurrences of η in σ, and connect them in this order by a curve $C(\eta)$ sandwiched between η and its counterpart.

Now consider any quadruple $x_i, y_i, x_{i+1}, y_{i+1}$. (The case $y_i, x_i, y_{i+1}, x_{i+1}$ is symmetric.) They must be appearing in this order along ∂f. In addition, the pair $x_i x_{i+1}$ is connected by part of $C(\xi)$, and similarly the pair $y_i y_{i+1}$ is connected by part of $C(\eta)$; both curves, except for their endpoints, lie in the exterior of f. Finally, we can place a point u at the interior of f, and connect it to x_i, y_i, x_{i+1} and y_{i+1} via internally disjoint curves inside the interior of f (Figure 10.18). By construction, no two of these curves cross, except possibly for the curves $x_i x_{i+1}$ and $y_i y_{i+1}$ in the exterior of f. In fact, these two curves must intersect, because otherwise we are facing a plane embedding of K_5 which does not exist.

In other words, any quadruple of consecutive elements from the subsequence σ induces an intersection between $C(\xi)$ and $C(\eta)$. Clearly these intersections are distinct for all quadruples, which altogether provide $s+4-3 = s+1$ intersections between $C(\xi)$ and $C(\eta)$, in contradiction to the fact that these curves can intersect at most s times. □

Corollary 10.37. *The combinatorial complexity of a single cell in an arrangement of n line segments in* $\mathbb{R}^2$ *is* $O(\lambda_3(n)) = O(n\alpha(n))$.

When counting the number of Davenport-Schinzel sequences of a certain type we want to count *essentially distinct* sequences only. Therefore we call two sequences over a common alphabet A *equivalent* if and only if one can be obtained from the other by a permutation of A. Then two sequences are *distinct* if and only if they are not

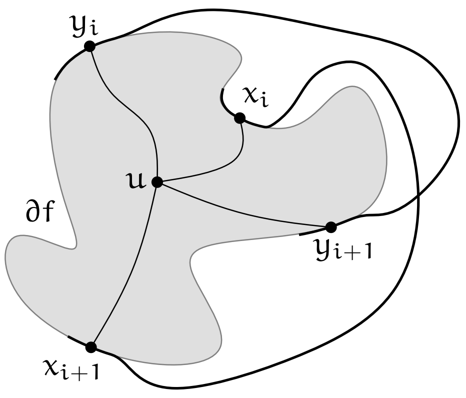


Figure 10.18: *Every quadruple x_i, y_i, x_{i+1}, y_{i+1} generates an intersection between the curves ξ and η.*

equivalent. A typical way to select a representative from each equivalence class is to order the alphabet and demand that the first appearance of a symbol in the sequence follows that order. For example, ordering $A = \{a, b, c\}$ alphabetically demands that the first occurrence of a precedes the first occurrence of b, which in turn precedes the first occurrence of c.

Exercise 10.38. *Let P be a convex polygon with $n+1$ vertices. Find a bijection between the triangulations of P and the set of pairwise distinct $(n, 2)$-Davenport-Schinzel sequences of maximum length ($2n - 1$). It follows that the number of distinct maximum $(n, 2)$-Davenport-Schinzel sequences is exactly $C_{n-1} = \frac{1}{n}\binom{2n-2}{n-1}$, which is the $(n-1)$-st Catalan number.*

Questions

53. *How can one construct an arrangement of lines in $\mathbb{R}^2$?* Describe the incremental algorithm and prove that its time complexity is quadratic in the number of lines (incl. statement and proof of the Zone Theorem).

54. *How can one test whether there are three collinear points in a set of n given points in $\mathbb{R}^2$?* Describe an $O(n^2)$ time algorithm.

55. *How can one compute the minimum area triangle spanned by three out of n given points in $\mathbb{R}^2$?* Describe an $O(n^2)$ time algorithm.

56. *What is a ham sandwich cut? Does it always exist? How to compute it?* State and prove the theorem about the existence of a ham sandwich cut in $\mathbb{R}^2$ and describe an $O(n^2)$ algorithm to compute it.

57. *What is the endpoint visibility graph for a set of disjoint line segments in the plane and how can it be constructed?* Give the definition and explain the

relation to shortest paths. Describe the $O(n^2)$ algorithm by Welzl, including a proof of Theorem 10.14.

58. *Is there a subquadratic algorithm for General Position?* Explain the term 3-Sum hard and its implications and give the reduction from 3-Sum to General Position.

59. *Which problems are known to be 3-Sum-hard?* List at least three problems (other than 3-Sum) and briefly sketch the corresponding reductions.

60. **(This topic was not covered in HS23 and therefore the question will not be asked in the exam.)** *What is an (n, s) Davenport-Schinzel sequence and how does it relate to the lower envelope of real-valued continuous functions?* Give the precise definitions and some examples. Explain in particular how to apply the machinery to line segments.

61. **(This topic was not covered in HS23 and therefore the question will not be asked in the exam.)** *What is the value of $\lambda_s(n)$, for $1 \leqslant s \leqslant 3$?* Derive the precise value for $s \in \{1, 2\}$ and prove an $O(n \log n)$ upper bound for $\lambda_3(n)$.

62. **(This topic was not covered in HS23 and therefore the question will not be asked in the exam.)** *What is the combinatorial complexity of the lower envelope of a set of n lines/parabolas/line segments?*

63. **(This topic was not covered in HS23 and therefore the question will not be asked in the exam.)** *What is the combinatorial complexity of a single cell in an arrangement of n line segments?* State the result and sketch the proof (Theorem 10.36).

References

[1] Radek Adamec, Martin Klazar, and Pavel Valtr, Generalized Davenport-Schinzel sequences with linear upper bound. *Discrete Math.*, 108, (1992), 219–229.

[2] Pankaj K. Agarwal and Micha Sharir, *Davenport-Schinzel sequences and their geometric applications*, Cambridge University Press, New York, NY, 1995.

[3] Pankaj K. Agarwal, Micha Sharir, and Peter W. Shor, Sharp upper and lower bounds on the length of general Davenport-Schinzel sequences. *J. Combin. Theory Ser. A*, 52/2, (1989), 228–274.

[4] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan, Time bounds for selection. *J. Comput. Syst. Sci.*, 7/4, (1973), 448–461.

[5] Herbert Edelsbrunner and Roman Waupotitsch, Computing a ham-sandwich cut in two dimensions. *J. Symbolic Comput.*, 2, (1986), 171–178.

[6] Jeff Erickson, Lower bounds for linear satisfiability problems. *Chicago J. Theoret. Comput. Sci.*, 1999/8.

[7] Anka Gajentaan and Mark H. Overmars, On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5, (1995), 165–185.

[8] Allan Grønlund and Seth Pettie, Threesomes, degenerates, and love triangles. *J. ACM*, 65/4, (2018), 22:1–22:25.

[9] Branko Grünbaum, Hamiltonian polygons and polyhedra. *Geombinatorics*, 3/3, (1994), 83–89.

[10] Sergiu Hart and Micha Sharir, Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6, (1986), 151–177.

[11] Michael Hoffmann, *On the existence of paths and cycles*. Ph.D. thesis, ETH Zürich, 2005.

[12] Michael Hoffmann and Csaba D. Tóth, Segment endpoint visibility graphs are Hamiltonian. *Comput. Geom. Theory Appl.*, 26/1, (2003), 47–68.

[13] Daniel M. Kane, Shachar Lovett, and Shay Moran, Near-optimal linear decision trees for k-SUM and related problems. *J. ACM*, 66/3, (2019), 16:1–16:18.

[14] Martin Klazar, On the maximum lengths of Davenport-Schinzel sequences. In R. Graham et al., ed., *Contemporary Trends in Discrete Mathematics*, vol. 49 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 169–178, Amer. Math. Soc., Providence, RI, 1999.

[15] Christian Knauer, Hans Raj Tiwary, and Daniel Werner, On the computational complexity of ham-sandwich cuts, Helly sets, and related problems. In *Proc. 28th Sympos. Theoret. Aspects Comput. Sci.*, vol. 9 of *LIPIcs*, pp. 649–660, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011.

[16] Chi-Yuan Lo, Jiří Matoušek, and William L. Steiger, Algorithms for ham-sandwich cuts. *Discrete Comput. Geom.*, 11, (1994), 433–452.

[17] Jiří Matoušek, *Using the Borsuk–Ulam theorem*, Springer, Berlin, 2003.

[18] Gabriel Nivasch, Improved bounds and new techniques for Davenport-Schinzel sequences and their generalizations. *J. ACM*, 57/3, (2010), Article No. 17.

[19] Seth Pettie, Splay trees, Davenport-Schinzel sequences, and the deque conjecture. In *Proc. 19th ACM-SIAM Sympos. Discrete Algorithms*, pp. 1115–1124, 2008.

[20] Masatsugu Urabe and Mamoru Watanabe, On a counterexample to a conjecture of Mirzaian. *Comput. Geom. Theory Appl.*, 2/1, (1992), 51–53.

[21] Emo Welzl, Constructing the visibility graph for n line segments in $O(n^2)$ time. *Inform. Process. Lett.*, 20, (1985), 167–171.