Chapter 10

Line Arrangements

During the course of this lecture we encountered several situations where it was convenient to assume that a point set is "in general position". In the plane, general position usually amounts to no three points being collinear and/or no four of them being cocircular. This raises an algorithmic question: How can we test for n given points whether or not three of them are collinear? Obviously, we can test all triples in $O(n^3)$ time. Can we do better? Yes, we can! Using a detour through the so-called dual plane, we will see that this problem can be solved in $O(n^2)$ time. However, the exact algorithmic complexity of this innocent-looking problem is not known. In fact, to determine this complexity is one of the major open problems in theoretical computer science.

We will get back to the complexity theoretic problems and ramifications at the end of this chapter. But first let us discuss how to obtain a quadratic time algorithm to test whether n given points in the plane are in general position. This algorithm is a nice application of the projective duality transform, as defined below. Such transformations are very useful because they allow us to gain a new perspective on a problem by formulating it in a different but equivalent form. Sometimes such a *dual* form of the problem is easier to work with and—given that it is equivalent to the original *primal* form—any solution to the dual problem can be translated back into a solution to the primal problem.

So what is this duality transform about? Recall the concept of a hyperplane in $\mathbb{R}^d$: the set of solutions to an equation of the form $\sum_{i=1}^{d} h_i x_i = h_{d+1}$, 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 using d coordinates/parameters. It is thus tempting to match these parameters to each other and so create a mapping between points and hyperplanes. In $\mathbb{R}^2$, hyperplanes are lines and the standard *projective duality* transform maps a point $p = (p_x, p_y)$ to the non-vertical line $p^* : y = p_x x - p_y$ and a non-vertical line $g : y = mx + b$ to the point $g^* = (m, -b)$.

Proposition 10.1. *The standard projective duality transform is*

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

Exercise 10.2. *Prove Proposition 10.1.*

Exercise 10.3. *Describe the image of the following point sets under this mapping*

a) a halfplane

b) $k \geqslant 3$ *collinear points*

c) a line segment

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

Another way to think of duality is in terms of the parabola $\mathcal{P} : y = \frac{1}{2}x^2$. For a point p on $\mathcal{P}$, the dual line p^* is the tangent to $\mathcal{P}$ at p. For a point p not on $\mathcal{P}$, consider the vertical projection p' of p onto $\mathcal{P}$: the slopes of p^* and p'^* are the same, just p^* is shifted by the difference in y-coordinates.

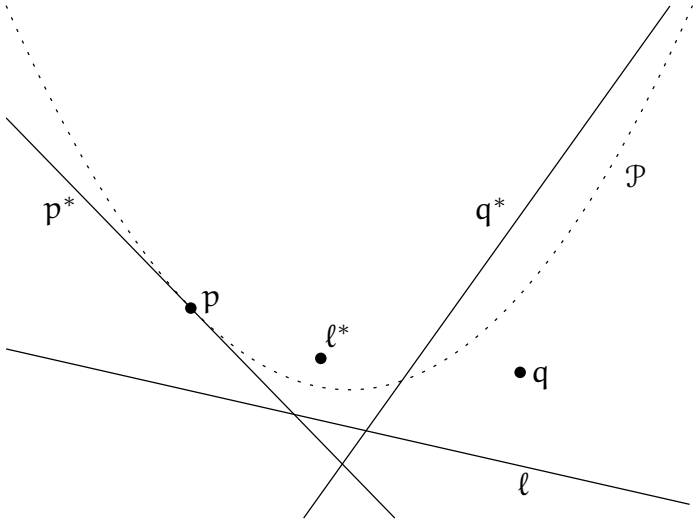


Figure 10.1: *Point* $\leftrightarrow$ *line duality with respect to the parabola* $\mathcal{P} : y = \frac{1}{2}x^2$.

The question 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 point. This question in turn we will answer with the help of *line arrangements*, as defined below.

10.1 Arrangements

The subdivision of the plane induced by a finite set L of lines is called the **arrangement** $\mathcal{A}(L)$. We may imagine the creation of this subdivision as a recursive process, defined by the given set L of lines. As a first step, remove all lines (considered as point sets) from the plane $\mathbb{R}^2$. What remains of $\mathbb{R}^2$ are a number of open connected components (possibly only one), which we call the (2-dimensional) **cells** of the subdivision. In the next step, from every line in L remove all the remaining lines (considered as point sets).

In this way every line is split into a number of open connected components (possibly only one), which collectively form the (1-dimensional cells or) **edges** of the subdivision. What remains of the lines are the (0-dimensional cells or) **vertices** of the subdivision, which are intersection points of lines from L.

Observe that all cells of the subdivision are intersections of halfplanes and thus convex. A line arrangement is **simple** if no two lines are parallel and no three lines meet in a point. Although lines are unbounded, we can regard a line arrangement a bounded object by (conceptually) putting a sufficiently large box around that contains all vertices. Such a box can be constructed in $O(n \log n)$ time for n lines.

Exercise 10.4. *How?*

Moreover, we can view a line arrangement as a planar graph by adding an additional vertex at "infinity", that is incident to all rays which leave this bounding box. For algorithmic purposes, we will mostly think of an arrangement as being represented by a doubly connected edge list (DCEL), cf. Section 2.2.1.

Theorem 10.5. *A simple arrangement $\mathcal{A}(L)$ of n lines in $\mathbb{R}^2$ has $\binom{n}{2}$ vertices, n^2 edges, and $\binom{n}{2} + n + 1$ faces/cells.*

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

The number of edges we count using induction on n. For $n = 1$ we have $1^2 = 1$ edge. By adding one line to an arrangement of $n-1$ lines we split $n-1$ existing edges into two and introduce n new edges along the newly inserted line. Thus, there are in total $(n-1)^2 + 2n - 1 = n^2 - 2n + 1 + 2n - 1 = n^2$ edges.

The number f of faces can now be obtained from Euler's formula $v - e + f = 2$, where v and e denote the number of vertices and edges, respectively. However, in order to apply Euler's formula we need to consider $\mathcal{A}(L)$ as a planar graph and take the symbolic "infinite" vertex into account. Therefore,

$$f = 2 - \left(\binom{n}{2} + 1\right) + n^2 = 1 + \frac{1}{2}(2n^2 - n(n-1)) = 1 + \frac{1}{2}(n^2 + n) = 1 + \binom{n}{2} + n.$$

□

The **complexity** of an arrangement is simply the total number of vertices, edges, and faces (in general, cells of any dimension).

Exercise 10.6. *Consider a set of lines in the plane with no three intersecting in a common point. Form a graph G whose vertices are the intersection points of the lines and such that two vertices are adjacent if and only if they appear consecutively along one of the lines. Prove that $\chi(G) \leqslant 3$, where $\chi(G)$ denotes the chromatic number of the graph G. In other words, show how to color the vertices of G using at most three colors such that no two adjacent vertices have the same color.*

10.2 Construction

As the complexity of a line arrangement is quadratic, there is no need to look for a subquadratic algorithm to construct it. We will simply construct it incrementally, inserting the lines one by one. Let $\ell_1, \ldots, \ell_n$ be the order of insertion.

At Step i of the construction, locate ℓ_i in the leftmost cell of $\mathcal{A}(\{\ell_1, \ldots, \ell_{i-1}\})$ it intersects. (The halfedges leaving the infinite vertex are ordered by slope.) This takes $O(i)$ time. Then traverse the boundary of the face F found until the halfedge h is found where ℓ_i leaves F (see Figure 10.2 for illustration). Insert a new vertex at this point, splitting F and h and continue in the same way with the face on the other side of h.

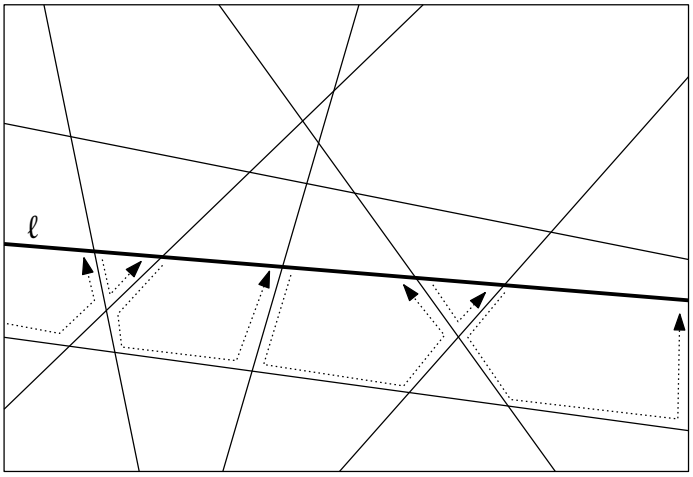


Figure 10.2: *Incremental construction: Insertion of a line ℓ. (Only part of the arrangement is shown in order to increase readability.)*

The insertion of a new vertex involves splitting two halfedges and thus is a constant time operation. But what is the time needed for the traversal? The complexity of $\mathcal{A}(\{\ell_1, \ldots, \ell_{i-1}\})$ is $\Theta(i^2)$, but we will see that the region traversed by a single line has linear complexity only.

10.3 Zone Theorem

For a line ℓ and an arrangement $\mathcal{A}(L)$, the **zone** $Z_{\mathcal{A}(L)}(\ell)$ of ℓ in $\mathcal{A}(L)$ 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 in $\mathbb{R}^2$ and a line ℓ (not necessarily from L), the total number of edges in all cells of the zone $Z_{\mathcal{A}(L)}(\ell)$ is at most $10n$.*

Proof. Without loss of generality suppose that ℓ is horizontal (rotate the plane accordingly). 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 in Figure 10.3, all left-bounding edges are shown blue and bold.

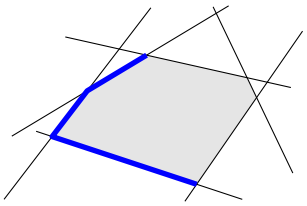


Figure 10.3: *Left-bounding edges (blue and bold) of a cell.*

We will show that there are at most $5n$ left-bounding edges in $Z_{\mathcal{A}(L)}(\ell)$ 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 $1 \leqslant 5n = 5$. Assume the statement is true for $n - 1$.

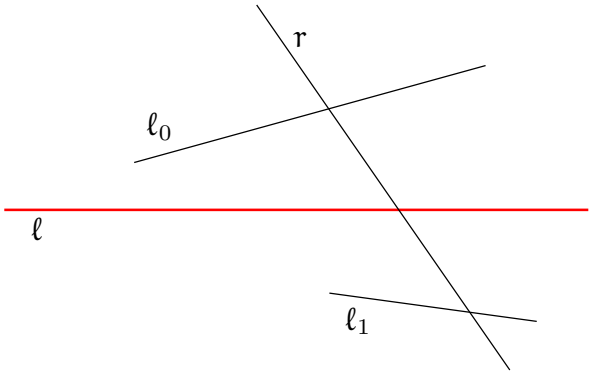


Figure 10.4: *At most three new left-bounding edges are created by adding* r *to* $\mathcal{A}(L\backslash\{r\})$.

If no line from L intersects ℓ, then all lines in $L \cup \{\ell\}$ are horizontal and there is at most $1 < 5n$ left-bounding edge in $Z_{\mathcal{A}(L)}(\ell)$. Else assume first that there is a single rightmost line r from L intersecting ℓ and the arrangement $\mathcal{A}(L \setminus \{r\})$. By the induction hypothesis there are at most $5n - 5$ 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 $3 + 5n - 5 < 5n$.

If there are several rightmost lines that intersect ℓ in the same point, we consider these lines in an arbitrary order. Using the same line of arguments as in the previous case, it can be observed that we add at most five left-bounding edges when adding a line r' after having added a line r, where both r and r' pass through the rightmost intersection point on ℓ. Apart from r, the line r' intersects at most two left-bounding edges ℓ_0 and ℓ_1 of cells in the zone of ℓ. There are two new left-bounding segments on r', and at most one additional on r. Hence, the number of left-bounding edges in this case is at most $5 + 5n - 5 = 5n$. □

Corollary 10.8. *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 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 10.5. □

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

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

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 $O(n)$.

10.4 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.

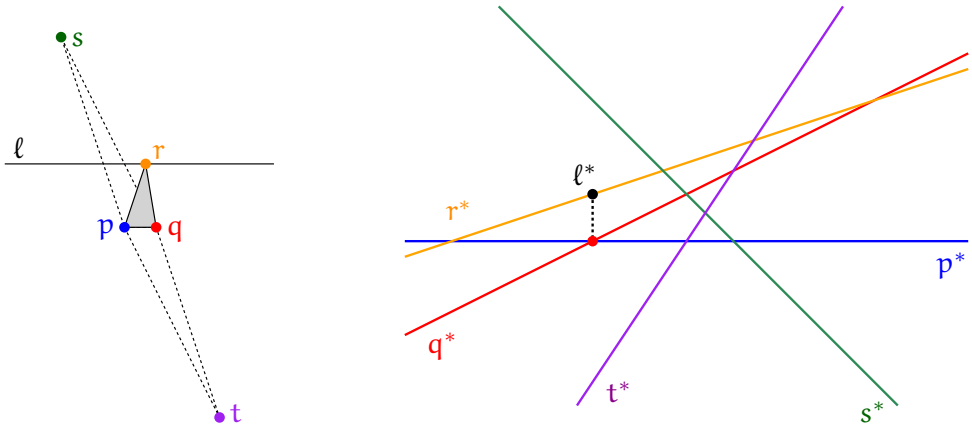
The following problems can be solved in $O(n^2)$ time and space by constructing the dual arrangement.

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

Minimum area triangle. Given a set $P \subset \mathbb{R}^2$ of n points, what is the minimum area triangle spanned by any three (pairwise distinct) points of P? 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 determined by the distance between r and the line pq. Thus, we want to find a point $r \in P \setminus \{p, q\}$ of minimum distance to pq. Equivalently, we want to find

a closest line ℓ parallel to pq so that ℓ passes 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 $p^* \cap q^*$ of $\mathcal{A}$ that lies on some line $r^* \in P^*$." See Figure 10.5 for illustration.



(a) primal **(b)** dual

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

In other words, for the vertex $v = p^* \cap q^*$ of $\mathcal{A}$ we want to know what is a first line from P^* that is hit by a vertical ray—upward or downward—emanating from v. Of course, in the end we want this information not only for such a single 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 vertices $p, q \in \binom{P}{2}$. Luckily, all this information can easily be maintained over the incremental construction of $\mathcal{A}$. When inserting a line ℓ, this new line may become the first line hit by some vertical rays from vertices of the already computed

partial arrangement. However, only vertices in the zone of ℓ may be affected. This zone is traversed, anyway, during the insertion of ℓ. So, during the traversal we can also check possibly update the information for vertices that lie vertically above or below a new edge of the arrangement, with no extra cost asymptotically.

In this way obtain $O(n^2)$ candidate triangles by constructing the arrangement of the n dual lines in $O(n^2)$ time. The smallest among those candidates can be determined by a straightforward minimum selection (comparing the area of the corresponding triangles).

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: 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.

10.5 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 10.11. *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

[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.

remaining points according to the slope of the connecting line; to construct the circular 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. □

10.6 Segment Endpoint Visibility Graphs

A fundamental problem in motion planning 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 $S \subset \mathbb{R}^2$ of obstacles. What is the shortest path between p and q that avoids S?

Observation 10.12. *The shortest path (if it exists) between two points that does not cross a finite set of finite polygonal obstacles is a polygonal path whose interior vertices are obstacle vertices.*

One of the simplest type of obstacle conceivable is a line segment. In general the plane may be disconnected with respect to the obstacles, for instance, if they form a closed curve. However, if we restrict the obstacles to pairwise disjoint line segments then there is always a free path between any two given points. Apart from start and goal position, by the above observation we may restrict our attention concerning shortest paths to straight line edges connecting obstacle vertices, in this 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 straight line graph defined on the segment endpoints. Two segment endpoints p and q are connected by an edge in $\mathcal{V}(S)$ if and only if*

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

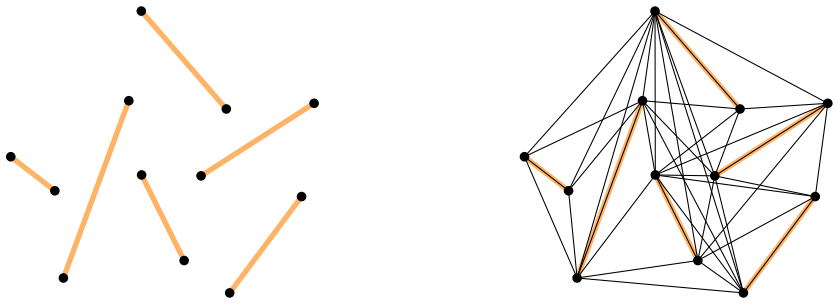


Figure 10.6: *A set of disjoint line segments and their endpoint visibility graph.*

If all segments are on the convex hull, the visibility graph is complete. If they form parallel chords of a convex polygon, the visibility graph consists of copies of K_4, glued together along opposite edges and the total number of edges is linear only.

These graphs also appear in the context of the following question: Given a set of disjoint line segments, is it possible to connect them to form (the boundary of) a simple polygon? It is easy to see that this is not possible in general: Just take three parallel chords of a convex polygon (Figure 10.7a). However, if we do not insist that the segments appear on the boundary, but allow them to be diagonals or epigonals, then it is always possible [11, 12]. In other words, 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 allow certain polygons.*

Constructing $\mathcal{V}(S)$ for a given set S of disjoint segments in a brute force way takes $O(n^3)$ time. (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, that is, no three points in P are collinear and no two have the same x-coordinate. It is no problem to handle such degeneracies explicitly.

Conceptually, we perform a *rotational sweep*. This means, 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 points exactly in direction of a segment $\overline{pq}$, then $s(p)$ is this segment.

During the sweep, the visibility graph can be computed as well, by simply outputting all pairs of the form $\{p, q\}, q \neq p$, where q is an endpoint of $s(p)$ at some stage of the sweep.

Why does this work? We only output edges of the visibility graph, by definition of the $s(p)$'s; moreover, each edge of the visibility graph has a left endpoint p, and its right endpoint q is at some stage of the sweep an endoint of $s(p)$.

To perform the actual sweep, we first observe that changes of visible segments can only occur at a discrete set of directions. Indeed, for any point $p \in P$, the segment $s(p)$ can only change when another point q is exactly in direction v, as seen from p. Hence, we only need to consider directions $q - p$ with $p, q \in P$, and we go through them in increasing order of slope (of the line through p, q), from $-\infty$ to $+\infty$.

Let us call a pair (p, q), q to the right of p, an *event*. The slope of (p, q) is the slope of the line through p and q. Then the sweep can be implemented as follows: process the

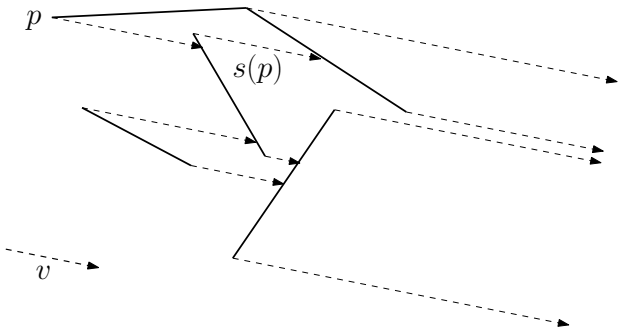


Figure 10.8: *Maintaining visible segments along a rotating direction v. Arrows pointing "far away" indicate that no segment is visible.*

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; see Figure 10.9.

1. p and q belong to the same input segment $\rightarrow$ output the edge pq, no change otherwise.
2. q is obscured from p by $s(p) \rightarrow$ no change.
3. q is endpoint of $s(p) \rightarrow$ output pq and update $s(p)$ to $s(q)$.
4. q is endpoint of a segment t that now obscures $s(p) \rightarrow$ output pq and update $s(p)$ to t.

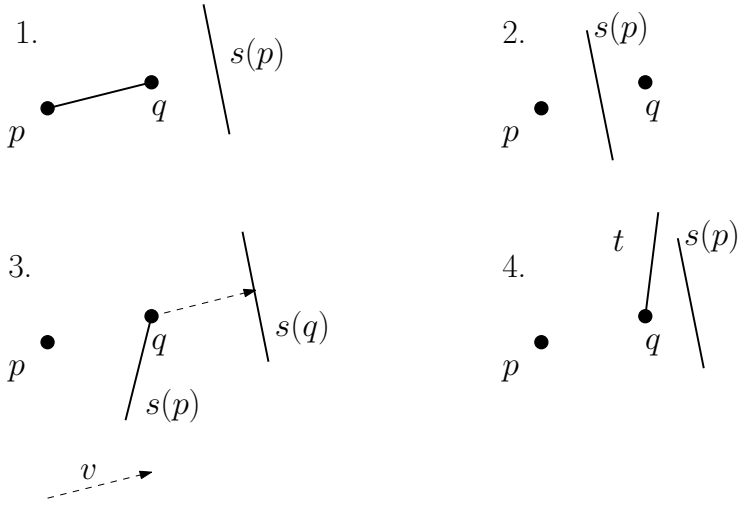


Figure 10.9: *Processing an event during the rotational sweep.*

What is the runtime of this rotational sweep? We have $O(n^2)$ events, and in order to process them in increasing order of slope, we need to sort them which 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. In order to correctly handle cases 1,2,4, we just need property (a) below, while property (b) takes care of case 3.

(a) For each $p \in P$, 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^* is the projective dual of P, the set of lines dual to the segment endpoints. Recall that the slope of a line through two points p, q corresponds to the x-coordinate of the intersection point of the dual lines p^* and q^*. Moeover, 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 the x-coordinate of the arrangement vertex $p^* \cap q^*$.

Then, properties (a) and (b) translate as follows.

(a^*) For $p^* \in P^*$, the events (p^*, q^*) are processed in order of 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 each arrangment vertex corresponds to an event, we satisfy properties (a^*) and (b^*) after *topologically* sorting the vertices. By this we mean that if vertex a is left of vertex b *on the same line*, then a should appear before b in the topological order. To obtain such an order, we perform a topological sort of the *directed* arrangement graph that has all edges directed from left to right. Clearly, this graph is acyclic (does not contain a directed cycle), so a topological sort exists. Such a topological sort can be obtained, for instance, via (reversed) post order DFS in time linear in the size of the graph (number of vertices and edges), which in our case here 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 P of n points, is it possible to sort the $\binom{n}{2}$ line segments $\overline{pq}$ by 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 (https://topp.openproblem.net/p41).*

Prove that $X + Y$ *sorting reduces (in* $O(n)$ *time) to the problem of sorting the* $\binom{2n}{2}$ *line segments* $\overline{pq}$ *by slope, for all pairs* $\{p, q\}$ *from a set of* $2n$ *points. Here, the slope of* $\overline{pq}$ *is defined as* ∞ *if* p *and* q *have the same* x*-coordinate; otherwise, the slope is* a*, where* $y = ax + b$ *is the unique non-vertical line containing* p *and* q*.*

10.7 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 $(s_1, \ldots, s_n)$ 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, ..., n {
    j = i, k = n.
    While k ⩾ 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 $x \in \{i, \ldots, j-1\}$, and $s_i + s_j + s_x > 0$, for all $x \in \{k+1, \ldots, n\}$.

Interestingly, until very recently this was the best 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 give a 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 (where a query amounts to ask for the sign of the sum of at most six input numbers with coefficients in $\{-1, 1\}$). 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. In other words, the results on the decision tree complexity of 3-Sum demonstrate that the (supposed) hardness of 3-Sum does not originate from the complexity of the decision tree.

[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 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[3]—it is known that 3-Sum requires quadratic time [6].

3-Sum hardness 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* 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* $o(n^{2-\varepsilon})$ *additional time, for some* $\varepsilon > 0$.

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 10.17. *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, $P = \{(a, a^3) \mid a \in S\}$.

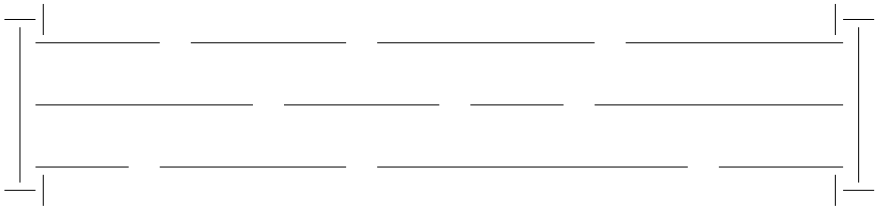
Suppose three of the points, say, (a, a^3), (b, b^3), and (c, c^3) are collinear. This is the case if and only if the slopes of the lines through each pair of them are equal. (Observe that a, b, and c are pairwise distinct.)

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

[3] where a decision depends on the sign of a linear expression in 3 input variables

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 $\overline{p_1 p_2}$ 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 10.18. *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.*

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, 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 10.19. *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.*

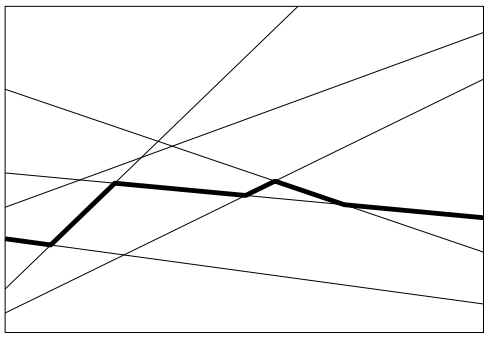


Figure 10.10: *The 3-level of an arrangement.*

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.

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 defined 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, 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 $\mathcal{A}(R^*)$. The median level of $\mathcal{A}(R^*)$ defines the bisecting lines

for R. As $|R| = |R^*|$ is odd, both the leftmost and the rightmost segment of this level are defined by the same line ℓ_r from R^*, the one with median slope. Similarly there is a corresponding line ℓ_d 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 $\mathcal{A}(D^*)$ intersect (see Figure 10.11 for an example). Any point that lies on both median levels corresponds to a primal line that bisects both point sets simultaneously. □

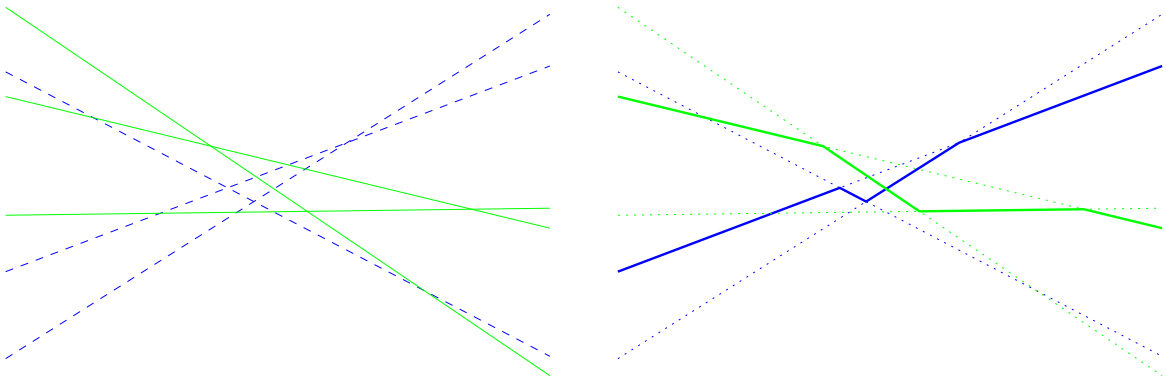


Figure 10.11: *An arrangement of 3 green lines (solid) and 3 blue lines (dashed) and their median levels (marked bold on the right hand side).*

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, the necklace is cut at most twice.

However, knowing about the existence of such a line certainly is not good enough. It is easy to turn the proof given above into an $O(n^2)$ algorithm to construct a line that simultaneously bisects both sets. But we can do better...

10.9 Constructing Ham Sandwich Cuts in the Plane

The algorithm outlined below is not only interesting in itself but also because it illustrates one of the fundamental general paradigms for designing optimization algorithms: *prune & search*. The basic idea behind prune & search is to search the space of possible solutions by at each step excluding some part of this space from further consideration. For instance, if at each step a constant fraction of all possible solutions can be discarded and a single step is linear in the number of solutions to be considered, then for the runtime we obtain a recursion of the form

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

that is, a linear time algorithm overall. Another well-known example of prune & search is binary search: every step takes constant time and about half of the possible solutions can be discarded, resulting in a logarithmic runtime overall.

Theorem 10.21 (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. That is, in either open halfplane defined by ℓ there are no more than $|R|/2$ points from R and no more than $|D|/2$ points from D.*

Proof. We describe a recursive algorithm find$(L_1, k_1, L_2, k_2, (x_1, x_2))$, for sets L_1, L_2 of lines in $\mathbb{R}^2$, non-negative integers k_1 and k_2, and a real interval (x_1, x_2), to find an intersection between the k_1-level of $\mathcal{A}(L_1)$ and the k_2-level of $\mathcal{A}(L_2)$, under the following assumption that is called *odd-intersection property*: 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, x_2) and they do not intersect at $x \in \{x_1, x_2\}$. Note that the odd-intersection property is equivalent to saying that the level that is above the other at $x = x_1$ is below the other at $x = x_2$. In the end, we are interested in find$(R^*, (|R| + 1)/2, D^*, (|D| + 1)/2, (-\infty, \infty))$. As argued in the proof of Theorem 10.20, for these arguments the odd-intersection property holds.

First let $L = L_1 \cup L_2$ and find a line μ with median slope in L. Denote by $L_<$ and $L_>$ the lines from L with slope less than and greater than μ, respectively. Using an infinitesimal rotation of the plane if necessary, we may assume without loss of generality that no two points in $R \cup D$ have the same x-coordinate and thus no two lines in L have the same slope. Pair the lines in $L_<$ with those in $L_>$ arbitrarily to obtain an almost perfect matching in the complete bipartite graph on $L_< \cup L_>$. Denote by I the $\lfloor (|L_<| + |L_>|)/2 \rfloor$ points of intersection generated by the pairs chosen, and let j be a point from I with median x-coordinate.

Determine the intersection (j, y_1) of the k_1-level of L_1 with the vertical line $x = j$ and the intersection (j, y_2) of the k_2-level of L_2 with the vertical line $x = j$. If both levels intersect at $x = j$, return the intersection and exit. Otherwise, if $j \in (x_1, x_2)$, then exactly one of the intervals (x_1, j) or (j, x_2) has the odd-intersection property, say[4], (x_1, j). In other words, we can from now on restrict our focus to the halfplane $x \leqslant j$. The case $j \notin (x_1, x_2)$ is no different, except that we simply keep the original interval (x_1, x_2).

In the following it is our goal to discard a constant fraction of the lines in L from further consideration. To this end, let $I_>$ denote the set of points from I with x-coordinate greater than j, and let μ' be a line parallel to μ such that about half of the points from $I_>$ are above μ' (and thus the other about half of points from $I_>$ are below μ'). We consider the four quadrants formed by the two lines $x = j$ and μ'. By assumption the odd-intersection property (for the k_1-level of L_1 and the k_2-level of L_2) holds for the (union of the) left two quadrants. Therefore the odd-intersection property holds for exactly one of the left two quadrants; we call this the *interesting* quadrant. Suppose furthermore that the upper left quadrant Q_2 is interesting. We will later argue how to algorithmically determine the interesting quadrant (see Figure 10.12 for an example).

[4]The other case is completely symmetric and thus will not be discussed here.

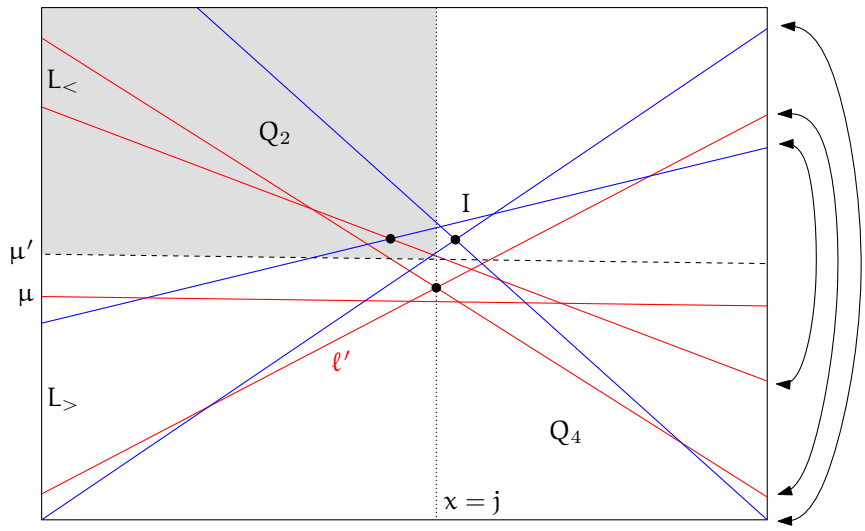


Figure 10.12: *An example with a set L_1 of 4 red lines and a set L_2 of 3 blue lines. Suppose that $k_1 = 3$ and $k_2 = 2$. Then the interesting quadrant is the top-left one (shaded) and the red line ℓ' (the line with a smallest slope in L_1) would be discarded because it does not intersect the interesting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

Then by definition of j and μ' about a quarter of the points from I are contained in the opposite, that is, the lower right quadrant Q_4. Any point in Q_4 is the point of intersection of two lines from L, exactly one of which has slope larger than μ'. As no line with slope larger than μ' that passes through Q_4 can intersect Q_2, any such line can be discarded from further consideration. In this case, the lines discarded pass completely below the interesting quadrant Q_2. For every line discarded in this way from L_1 or L_2, the parameter k_1 or k_2, respectively, has to be decreased by one. In the symmetric case where the lines discarded pass above the interesting quadrant, the parameters k_1 and k_2 stay the same. In any case, about a 1/8-fraction of all lines in L is discarded. Denote the resulting sets of lines (after discarding) by L_1' and L_2', and let k_1' and k_2' denote the correspondingly adjusted levels.

We want to apply the algorithm recursively to compute an intersection between the k_1'-level of L_1' and the k_2'-level of 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 L_1' and the k_2'-level of L_2' on the interval (x_1, j), or even on the original interval (x_1, x_2). 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 under consideration. Luckily, this issue can easily 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 a subinterval $(x_1', x_2') \subseteq (x_1, x_2)$ on which the odd-intersection property holds for the k_1'-level of L_1'

and the k_2'-level of L_2'.

So let us argue how to determine the interesting quadrant, that is, how to test whether the k_1-level of L_1 and the k_2-level of L_2 intersect an odd number of times in $S_{(x_1,j)} \cap H_\mu^+$, where $S_{(x_1,j)}$ is the vertical strip $(x_1, j) \times \mathbb{R}$ and H_μ^+ is the open halfplane above μ'. For this it is enough to trace μ' through the arrangement $\mathcal{A}(L)$ while keeping track of the position of the two levels of interest. Initially, at $x = x_1$ we know which level is above the other. At every intersection of one of the two levels with μ', we can check whether the ordering is still consistent with that initial ordering. For instance, if both were above μ' initially and the level that was above the other intersects μ' first, 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 trace of μ' in $\mathcal{A}(L)$ can be computed by a sweep along μ', which amounts to computing all intersections of μ' with the lines from L and sorting them by x-coordinate. During the sweep we keep track of the number of lines from L_1 below μ' and the number of lines from L_2 below μ'. At every point of intersection, these counters can be adjusted and any intersection with one of the two levels of interest is detected. Therefore computing the trace 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

$$T(n) \leqslant cn \log n + T(7n/8) = O(n \log n). \qquad \square$$

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 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 10.22. *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.

In the plane, a ham sandwich cut can 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[5] 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 10.23. *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.

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 $0 < \alpha < 1$.

Exercise 10.24. *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).*

10.10 Davenport-Schinzel Sequences

The complexity of a simple arrangement of n lines in $\mathbb{R}^2$ is $\Theta(n^2)$ and so every algorithm that uses such an arrangement explicitly needs $\Omega(n^2)$ time. However, there are many scenarios in which we do not need the whole arrangement but only some part of it. For instance, to construct a ham sandwich cut for two sets of points in $\mathbb{R}^2$ one needs the median levels of the two corresponding line arrangements only. As mentioned in the previous section, the relevant information about these levels can actually be obtained in linear time. Similarly, in a motion planning problem where the lines are considered as obstacles we are only interested in the cell of the arrangement we are located in. There is no way to ever reach any other cell, anyway.

This chapter is concerned with analyzing the complexity—that is, the number of vertices and edges—of a single cell in an arrangement of n curves in $\mathbb{R}^2$. In case of a line arrangement this is mildly interesting only: Every cell is convex and any line can appear at most once along the cell boundary. On the other hand, it is easy to construct an example in which there is a cell C such that every line appears on the boundary ∂C.

But when we consider arrangements of 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 $\mathbb{R}^2$?

[5] 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.

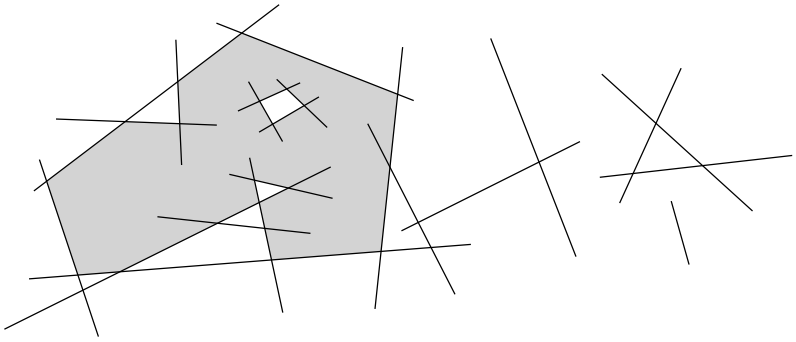


Figure 10.13: *A single cell in an arrangement of line segments.*

You will find out the correct answer soon, although we will not prove it here. But my guess would be that it is rather unlikely that your guess is correct, unless, of course, you knew the answer already. :-)

For a start we will focus on one particular cell of any arrangement that is very easy to describe: the lower envelope or, intuitively, everything that can be seen vertically from below. To analyze the complexity of lower envelopes we use a combinatorial description using strings with forbidden subsequences, so-called Davenport-Schinzel sequences. These sequences are of independent interest, as they appear in a number of combinatorial problems [2] and in the analysis of data structures [19]. The techniques used apply not only to lower envelopes but also to arbitrary cells of arrangements.

Definition 10.25. *An* (n, s)-**Davenport-Schinzel sequence**, *for* $n, s \in \mathbb{N}$, *is a sequence over an alphabet* A *of size* n *in which*

- *no two consecutive characters are the same and*
- *there is no alternating subsequence of the form* $\ldots a \ldots b \ldots a \ldots b \ldots$ *of* $s+2$ *characters, for any* $a, b \in A$.

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

Exercise 10.31 asks you to prove that $\lambda_s(n)$ is indeed finite. As an example, $abcbacb$ is a $(3,4)$-DS sequence but not a $(3,3)$-DS sequence because it contains the subsequence $bcbcb$.

Proposition 10.26. $\lambda_s(m) + \lambda_s(n) \leqslant \lambda_s(m+n)$.

Proof. On the left hand side, we consider two Davenport-Schinzel sequences, one over an alphabet A of size m and another over an alphabet B of size n. We may suppose that $A \cap B = \emptyset$ (for each character $x \in A \cap B$ introduce a new character x' and replace all occurrences of x in the second sequence by x'). Concatenating both sequences yields a Davenport-Schinzel sequence over the alphabet $A \cup B$ of size $m+n$. □

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 and bounded) interval $I \subset \mathbb{R}$. The *lower envelope* $\mathcal{L}_{\mathcal{F}}$ of $\mathcal{F}$ is defined as the pointwise minimum of the functions f_i, for $1 \leqslant i \leqslant n$, over I. Formally, for $x \in I$, $\mathcal{L}_{\mathcal{F}}(x) := \min\{f_i(x) : 1 \leqslant i \leqslant n\}$.

Suppose that the graphs of any two functions f_i, f_j, for $1 \leqslant i < j \leqslant n$, intersect in finitely many points. Then the function that defines $\mathcal{L}_{\mathcal{F}}$ at x can change only finitely many times as x moves along I from left to right. The indices of the defining functions that we encounter from left to right form the *lower envelope sequence* $\phi(\mathcal{F}) = (\phi_1, \ldots, \phi_\ell)$; see Figure 10.14 for an illustration.

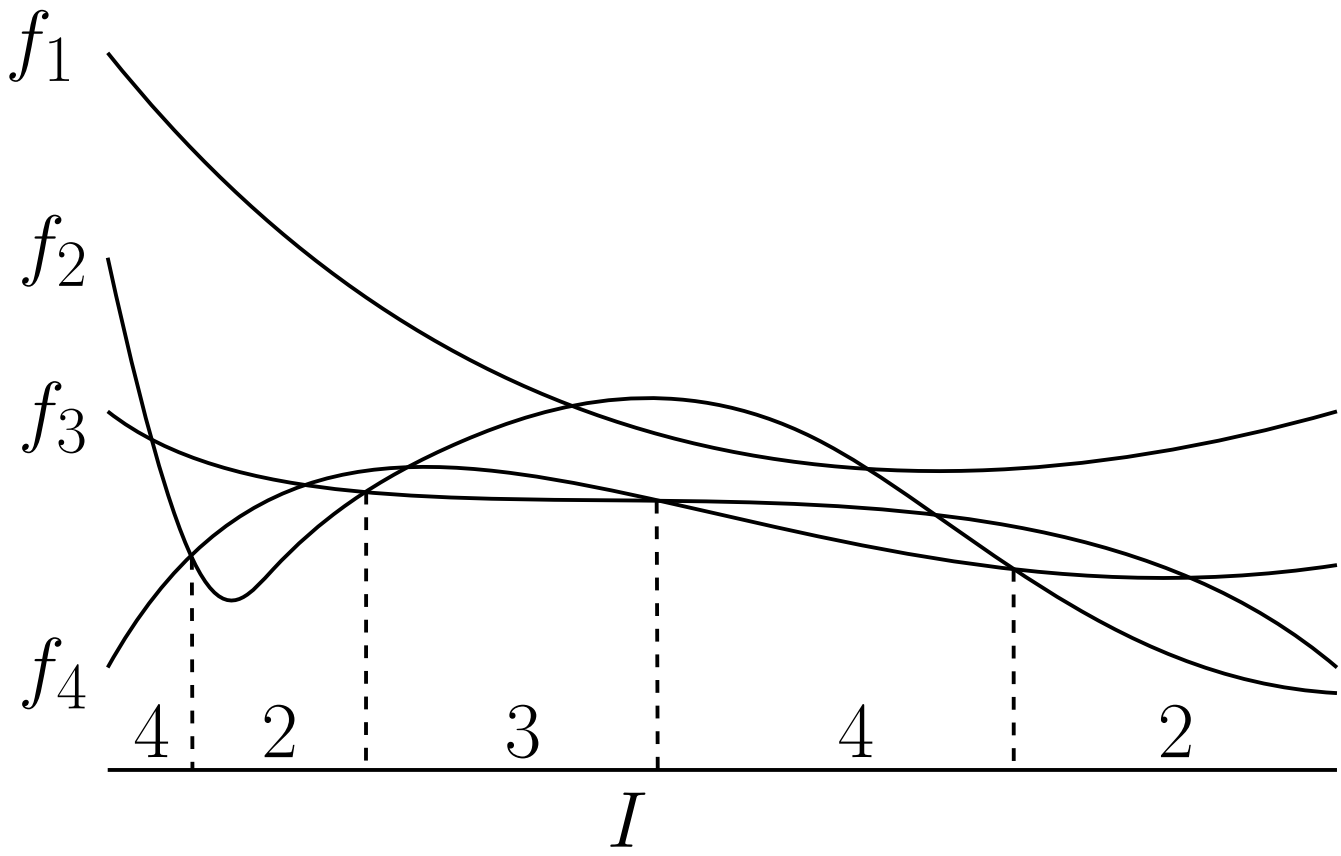


Figure 10.14: *The lower envelope sequence of a set of continuous functions.*

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

Observation 10.27. *If the graphs of any two functions intersect at most s times, then $\phi(\mathcal{F})$ is an (n, s)-Davenport-Schinzel sequence.*

In the case of line segments (linear functions over intervals) the above machinery is not applicable because a set of line segments is in general not defined on a common

interval. Let us first adapt our definitions to the more general case where each function in $\mathcal{F}$ has an individual (closed and bounded) interval as its domain, but again assuming that the graphs of any two functions f_i, f_j, for $1 \leqslant i < j \leqslant n$, intersect in at most finitely many points.

The lower envelope $\mathcal{L}_{\mathcal{F}}$ (now defined over the real line) is then the function f given by $f(x) = \min\{f_i(x) : 1 \leqslant i \leqslant n \text{ and } f_i \text{ is defined at } x\}$. In the case where no f_i is defined at x, we have $f(x) = \infty$. The lower envelope sequence $\phi(\mathcal{F})$ again records the indices of the functions that define $\mathcal{L}_{\mathcal{F}}$ as we encounter them from left to right, where we use index 0 for intervals where $\mathcal{L}_{\mathcal{F}}(x) = \infty$; see Figure 10.15 for an illustration in the case of line segments.

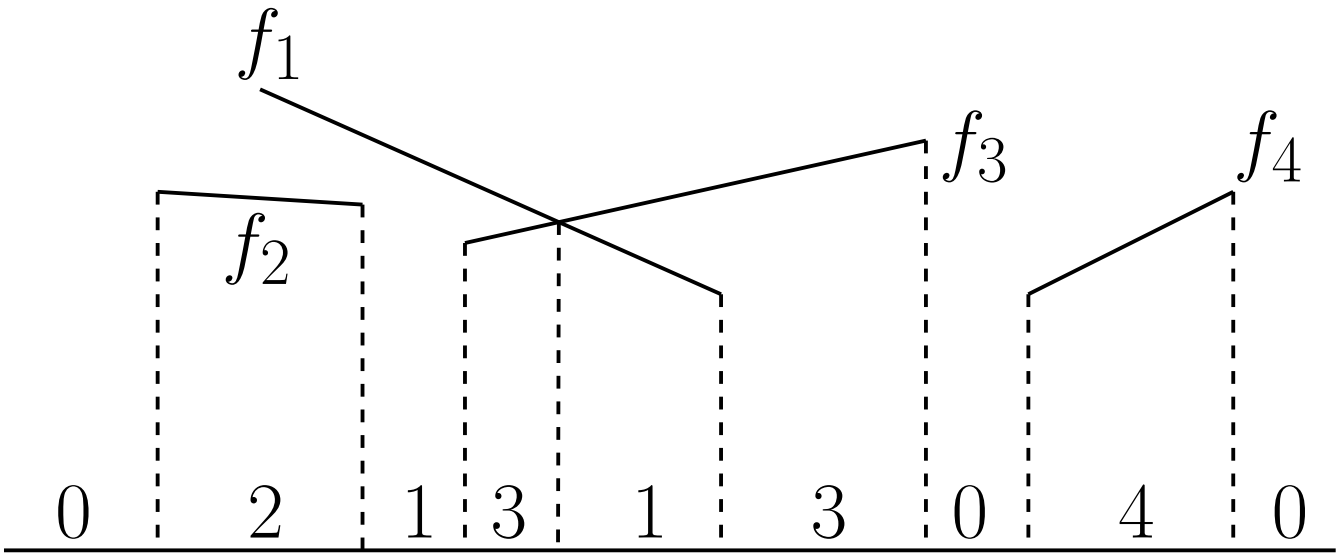


Figure 10.15: *The lower envelope sequence of a set of segments.*

In Figure 10.15 we already see that two segments f_i, f_j, despite crossing only once, may lead to a lower envelope subsequence $\ldots i \ldots j \ldots i \ldots j \ldots$ of 4 characters (and not 2 as in the case where the segments would be defined over a common interval). Here is the general result.

Proposition 10.28. *Let $\mathcal{F}$ be a collection of n real-valued continuous functions, each of which is defined on some closed and bounded interval. If the graphs of any two functions from $\mathcal{F}$ intersect in at most s points, then $\phi(\mathcal{F})$ is an $(n+1, s+2)$-Davenport-Schinzel sequence.*

Proof. We first show that there is no alternating subsequence $\ldots i \ldots j \ldots i \ldots j \ldots$ of length $s+4$ if $i, j \geqslant 1$. For this, we consider the at most s points at which $f_i(x) = f_j(x)$ and the at most 4 domain endpoints of f_i and f_j. These together subdivide the real line into at most $s+5$ intervals. Within each of the at most $s+3$ closed inner intervals, one (but only one) of f_i and f_j may define the lower envelope (possibly in several subintervals).

Hence, the longest alternating lower envelope subsequence $\dots i \dots j \dots i \dots j \dots$ has length at most $s+3$.

We finally observe that we cannot have a subsequence $\dots i \dots 0 \dots i$ for $i \geqslant 1$, since the domain of f_i is an interval. Hence, if one of i, j is 0, there are no alternating subsequences $\dots i \dots j \dots i \dots j \dots$ of length 4. The statement follows. □

Next we will give an upper bound on the length of Davenport-Schinzel sequences for small s.

Lemma 10.29. $\lambda_1(n) = n$, $\lambda_2(n) = 2n-1$, *and* $\lambda_3(n) \leqslant 2n(1+\log n)$.

Proof. $\lambda_1(n) = n$ is obvious. $\lambda_2(n) = 2n-1$ is given as an exercise. We prove $\lambda_3(n) \leqslant 2n(1+\log n) = O(n \log n)$.

For $n = 1$ it is $\lambda_3(1) = 1 \leqslant 2$. For $n > 1$ consider any $(n,3)$-DS sequence σ of length $\lambda_3(n)$. Let a be a character that appears least frequently in σ. Clearly a appears at most $\lambda_3(n)/n$ times in σ. Delete all appearances of a from σ to obtain a sequence σ' on $n-1$ symbols. But σ' is not necessarily a DS sequence because there may be consecutive appearances of a character b in σ', in case that $\sigma = \dots bab \dots$.

Claim: There are at most two pairs of consecutive appearances of the same character in σ'. Indeed, such a pair can be created around the first and last appearance of a in σ only. If any intermediate appearance of a creates a pair bb in σ' then $\sigma = \dots a \dots bab \dots a \dots$, in contradiction to σ being an $(n,3)$-DS sequence.

Therefore, one can remove at most two characters from σ' to obtain a $(n-1,3)$-DS sequence $\tilde{\sigma}$. As the length of $\tilde{\sigma}$ is bounded by $\lambda_3(n-1)$, we obtain $\lambda_3(n) \leqslant \lambda_3(n-1) + \lambda_3(n)/n + 2$. Reformulating yields

$$\underbrace{\frac{\lambda_3(n)}{n}}_{=:f(n)} \leqslant \underbrace{\frac{\lambda_3(n-1)}{n-1}}_{=f(n-1)} + \frac{2}{n-1} \leqslant \underbrace{1}_{=f(1)} + 2\sum_{i=1}^{n-1} \frac{1}{i} = 1 + 2H_{n-1}$$

and 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 (taking s fixed). The bounds are known almost exactly, and they involve the inverse Ackermann function $\alpha(n)$, a function that grows extremely slowly.

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

We first let $\alpha_1(n) = \lceil n/2 \rceil$. Then, for each $k \geqslant 2$, we define $\alpha_k(n)$ to be the number of times we must apply α_{k-1}, starting from n, until we get a result not 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 $\alpha_1(n), \alpha_2(n), \alpha_3(n), \ldots$. For every fixed n, this sequence decreases rapidly until it settles at 3. We define $\alpha(n)$ (the inverse Ackermann function) as the function that, given n, returns the smallest k such that $\alpha_k(n)$ is at most 3:

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

We leave as an exercise to show that for every fixed k we have $\alpha_k(n) = o(\alpha_{k-1}(n))$ and $\alpha(n) = o(\alpha_k(n))$.

Coming back to the bounds for Davenport-Schinzel sequences, for $\lambda_3(n)$ (no $ababa$) it is known that $\lambda_3(n) = \Theta(n\alpha(n))$ [10]. In fact it is known that $\lambda_3(n) = 2n\alpha(n) \pm O(n\sqrt{\alpha(n)})$ [14, 18]. For $\lambda_4(n)$ (no $abababa$) 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 DS 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 not possible.)

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

Generalizations of DS sequences. Also generalizations of Davenport-Schinzel sequences have been studied, for instance, where arbitrary subsequences (not necessarily an alternating pattern) are forbidden. For a word σ and $n \in \mathbb{N}$ define $\text{Ex}(\sigma, n)$ to be the maximum length of a word over $A = \{1, \ldots, n\}^*$ that does not contain a subsequence of the form σ. For example, $\text{Ex}(ababa, n) = \lambda_3(n)$. If σ consists of two letters only, say a and b, then $\text{Ex}(\sigma, n)$ is super-linear if and only if σ contains $ababa$ as a subsequence [1]. This highlights that the alternating forbidden pattern is of particular interest.

Exercise 10.30. *Prove that $\lambda_2(n) = 2n - 1$.*

Exercise 10.31. *Prove that $\lambda_s(n)$ is finite for all s and n.*

10.11 Constructing lower envelopes

Theorem 10.32. *Let $\mathcal{F} = \{f_1, \ldots, f_n\}$ be a collection of real-valued continuous functions defined on a common interval $I \subset \mathbb{R}$ such that no two functions from $\mathcal{F}$ intersect in more than s points. Then the lower envelope $\mathcal{L}_\mathcal{F}$ can be constructed in $O(\lambda_s(n) \log n)$ time. (Assuming that an intersection between any two functions can be constructed in constant time.)*

Proof. Divide and conquer. For simplicity, assume that n is a power of two. Split $\mathcal{F}$ into two equal parts $\mathcal{F}_1$ and $\mathcal{F}_2$ and construct $\mathcal{L}_{\mathcal{F}_1}$ and $\mathcal{L}_{\mathcal{F}_2}$ recursively. The resulting envelopes can be merged using line sweep by processing $2\lambda_s(n/2)+\lambda_s(n) \leqslant 2\lambda_s(n)$ events (the inequality $2\lambda_s(n/2) \leqslant \lambda_s(n)$ is by Proposition 10.26). Here the first term accounts for events generated by the vertices of the two envelopes to be merged. The second term accounts for their intersections, each of which generates a vertex of the resulting envelope. Observe that no sorting is required and the sweep line status structure is of constant size. Therefore, the sweep can be done in time linear in the number of events.

This yields the following recursion for the runtime $T(n)$ of the algorithm. $T(n) \leqslant 2T(n/2) + c\lambda_s(n)$, for some constant $c \in \mathbb{N}$. Using Proposition 10.26 it follows that $T(n) \leqslant c\sum_{i=1}^{\log n} 2^i\lambda_s(n/2^i) \leqslant c\sum_{i=1}^{\log n} \lambda_s(n) = O(\lambda_s(n)\log n)$. □

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 Davenport-Schinzel sequence of order two can be realized as a lower envelope of n parabolas.*

10.12 Complexity of a single face

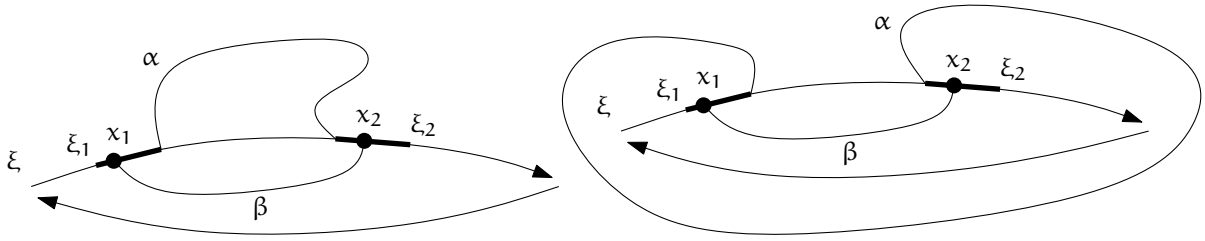
Theorem 10.35. *Let $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ be a collection of Jordan arcs in $\mathbb{R}^2$ such that each pair intersects in at most s points, for some $s \in \mathbb{N}$. Then the combinatorial complexity of any single face in the arrangement $\mathcal{A}(\Gamma)$ is $O(\lambda_{s+2}(n))$.*

Proof. Consider a face f of $\mathcal{A}(\Gamma)$. In general, the boundary of f might consist of several connected components. But as any single curve can appear in at most one component, by Proposition 10.26 we may suppose that the boundary consists of one component only.

Replace each γ_i by two directed arcs γ_i^+ and γ_i^- that together form a closed curve that is infinitesimally close to γ_i. Denote by S the circular sequence of these oriented curves, in their order along the (oriented) boundary ∂f of f.

Consistency Lemma. Let ξ be one of the oriented arcs γ_i^+ or γ_i^-. The order of portions of ξ that appear in S is consistent with their order along ξ. (That is, for each ξ we can break up the circular sequence S into a linear sequence $S(\xi)$ such that the portions of ξ that correspond to appearances of ξ in $S(\xi)$ appear in the same order along ξ.)

Consider two portions ξ_1 and ξ_2 of ξ that appear consecutively in S (that is, there is no other occurrence of ξ in between). Choose points $x_1 \in \xi_1$ and $x_2 \in \xi_2$ and connect them in two ways: first by the arc α following ∂f as in S, and second by an arc β inside the closed curve formed by γ_i^+ or γ_i^-. The curves α and β do not intersect except at their endpoints and they are both contained in the complement of the interior of f. In other words, $\alpha \cup \beta$ forms a closed Jordan curve and f lies either in the interior of this curve or in its exterior. In either case, the part of ξ between ξ_1 and ξ_2 is separated



(a) f lies on the unbounded side of $\alpha \cup \beta$. **(b)** f lies on the bounded side of $\alpha \cup \beta$.

Figure 10.16: *Cases in the Consistency Lemma.*

from f by $\alpha \cup \beta$ and, therefore, no point from this part can appear anywhere along ∂f. In other words, ξ_1 and ξ_2 are also consecutive boundary parts in the order of boundary portions along ξ, which proves the lemma.

Break up S into a linear sequence $S' = (s_1, \ldots, s_t)$ arbitrarily. For each oriented arc ξ, consider the sequence $s(\xi)$ of its portions along ∂f in the order in which they appear along ξ. By the Consistency Lemma, $s(\xi)$ corresponds to a subsequence of S, starting at s_k, for some $1 \leqslant k \leqslant t$. In order to consider $s(\xi)$ as a subsequence of S', break up the symbol for ξ into two symbols ξ and ξ' and replace all occurrences of ξ in S' before s_k by ξ'. Doing so for all oriented arcs results in a sequence S^* on at most $4n$ symbols.

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

Clearly no two adjacent symbols in S^* are the same. Suppose S^* contains an alternating subsequence $\sigma = \ldots\xi\ldots\eta\ldots\xi\ldots\eta\ldots$ of length $s+4$. For any occurrence of ξ in this sequence, 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. These points we can connect in this order by a Jordan arc $C(\xi)$ that stays within the closed curve formed by ξ and its counterpart—except for the points $x_1, \ldots, x_{\lceil (s+4)/2 \rceil}$, which lie on this closed curve. 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 these points in this order by a Jordan arc $C(\eta)$ that stays (except for the points $y_1, \ldots, y_{\lfloor (s+4)/2 \rfloor}$) within the closed curve formed by η and its counterpart.

Now consider any four consecutive elements in σ and the corresponding points x_i, y_i, x_{i+1}, y_{i+1}, which appear in this order—and, thus, can be regarded as connected by an arc—along ∂f. In addition, the points x_i and x_{i+1} are connected by an arc of $C(\xi)$, and similarly y_i and y_{i+1} are connected by an arc of $C(\xi)$. Both arcs, except for their endpoints, lie in the exterior of f. Finally, we can place a point u into the interior of f and connect u by pairwise interior-disjoint arcs to each of x_i, y_i, x_{i+1}, and y_{i+1}, such that the relative interior of these four arcs stays in the interior of f. By construction, no two of these arcs cross (intersect at a point that is not a common endpoint), except possibly for the arcs x_i, x_{i+1} and y_i, y_{i+1} in the exterior of f. In fact, these two arcs must intersect, because otherwise we are facing a plane embedding of K_5, which does not exist (Figure 10.17).

In other words, any quadruple of consecutive elements from the alternating subsequence induces an intersection between the corresponding arcs ξ and η. Clearly these

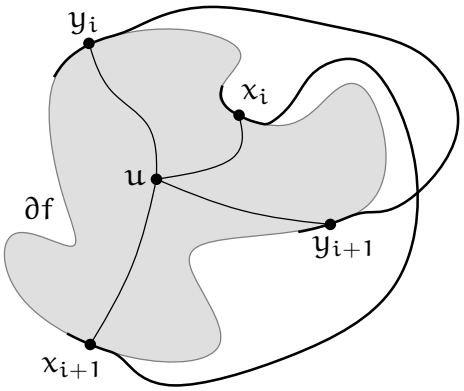


Figure 10.17: *Every quadruple* x_i, y_i, x_{i+1}, y_{i+1} *generates an intersection between the curves* ξ *and* η.

intersection points are pairwise distinct for any pair of distinct quadruples which altogether provides $s+4-3 = s+1$ points of intersection between ξ and η, in contradiction to the assumption that these curves intersect in at most s points. □

Corollary 10.36. *The combinatorial complexity of a single face in an arrangement of* n *line segments in* $\mathbb{R}^2$ *is* $O(\lambda_3(n)) = O(n\alpha(n))$.

Exercise 10.37.

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* j *we have* $\alpha_k(n) = o(\alpha_{k-1}(\alpha_{k-1}(\cdots \alpha_{k-1}(n)\cdots)))$*, with* j *applications of* α_{k-1}.

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

It is a direct consequence of the symmetry in the definition that the property of being a Davenport-Schinzel sequence is invariant under permutations of the alphabet. For instance, $\sigma = bcacba$ is a $(3,3)$-DS sequence over $A = \{a, b, c\}$. Hence the permutation $\pi = (ab)$ induces a $(3,3)$-DS sequence $\pi(\sigma) = acbcab$ and similarly $\pi' = (cba)$ induces another $(3,3)$-DS sequence $\pi'(\sigma) = abcbac$.

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 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

52. *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).

53. *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.

54. *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.

55. *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.

56. (**This topic was not covered in this year's course in HS20 and therefore the following question will not be asked in the exam.**) *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 full proofs of Theorem 10.11 and Theorem 10.14.

57. (**This topic was not covered in this year's course in HS20 and therefore the following question will not be asked in the exam.**) *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.

58. (**This topic was not covered in this year's course in HS20 and therefore the following question will not be asked in the exam.**) *Which problems are known to be 3-Sum-hard?* List at least three problems (other than 3-Sum) and briefly sketch the corresponding reductions.

59. *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.

60. *What is the value of* $\lambda_1(n)$ *and* $\lambda_2(n)$*?*

61. *What is the asymptotic value of* $\lambda_3(n)$*,* $\lambda_4(n)$*, and* $\lambda_s(n)$ *for larger* s*?*

62. *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 this year's course in HS20 and therefore the following question will not be asked in the exam.**) *What is the combinatorial complexity of a single face in an arrangement of* n *line segments?* State the result and sketch the proof (Theorem 10.35).

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-Verlag, 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.