Geometry: Combinatorics & Algorithms
Lecture Notes HS 2014

Bernd Gärtner <gaertner@inf.ethz.ch>
Michael Hoffmann <hoffmann@inf.ethz.ch>
Emo Welzl <emo@inf.ethz.ch>

Monday 12th January, 2015

Preface

These lecture notes are designed to accompany a course on "Geometry: Combinatorics & Algorithms" that we teach at the Department of Computer Science, ETH Zürich, for the first time in the winter term 2014. The course is a synthesis of topics from computational geometry, combinatorial geometry, and graph drawing that are centered around triangulations, that is, geometric representations of maximal planar graphs. The selection of topics has been done according to three criteria.

Importance. What are the most essential concepts and techniques that we want our students to know? (for instance, if they plan to write a thesis in the area)

Overlap. What is covered and to which extent in other courses of our curriculum?

Coherence. How closely related is something to the focal topic of triangulations and how well does it fit with the other topics selected?

Our focus is on low-dimensional Euclidean space (mostly 2D), although we sometimes discuss possible extensions and/or remarkable differences when going to higher dimensions. At the end of each chapter there is a list of questions that we expect our students to be able to answer in the oral exam.

In the current setting, the course runs over 14 weeks, with two hours of lecture and two hours of exercises each week. In addition, there are three sets of graded homeworks which students have to hand in spread over the course. The target audience are third-year Bachelor or Master students of Mathematics or Computer Science.

Most parts of these notes have been used within earlier courses of a similar type. Hence they have gone through a number of iterations of proof-reading. But experience tells that there are always a few mistakes that escape detection. So in case you notice some problem, please let us know, regardless of whether it is a minor typo or punctuation error, a glitch in formulation, or a hole in an argument. This way the issue can be fixed for the next edition and future readers profit from your findings.

We thank Kateřina Böhmová, Tobias Christ, Anna Gundert, Gabriel Nivasch, Júlia Pap, Marek Sulovský, May Szedlák, and Hemant Tyagi for pointing out errors in preceding versions.

Bernd Gärtner, Michael Hoffmann, and Emo Welzl
Department of Computer Science, ETH Zürich

Universitätstrasse 6, CH-8092 Zürich, Switzerland
E-mail address: {gaertner,hoffmann,emo}@inf.ethz.ch

Contents

Chapter 1

Fundamentals

1.1 Models of Computation

When designing algorithms, one has to agree on a model of computation according to which these algorithms can be executed. There are various such models, but when it comes to geometry some are more convenient to work with than others. Even using very elementary geometric operations—such as taking the center of a circle defined by three points or computing the length of a given circular arc—the realms of rational and even algebraic numbers are quickly left behind. Representing the resulting real numbers/coordinates would be a rather painful task in, for instance, a Turing machine type model of computation.

Therefore, other models of computation are more prominent in the area of geometric algorithms and data structures. In this course we will be mostly concerned with two models: the *Real RAM* and the *algebraic computation/decision tree* model. The former is rather convenient when designing algorithms, because it sort of abstracts from the aforementioned representation issues by simply *assuming* that it can be done. The latter model typically appears in the context of lower bounds, that is, proofs that certain problems cannot be solved more efficiently than some function depending on the problem size (and possibly some other parameters).

So let us see what these models are in more detail.

Real RAM Model. A memory cell stores a real number (that is what the "Real" stands for)[1]. Any single arithmetic operation (addition, subtraction, multiplication, division, and k-th root, for small constant k) or comparison can be computed in constant time.[2] This is a quite powerful (and somewhat unrealistic) model of computation, as a single real number in principle can encode an arbitrary amount of information. Therefore we

[1] RAM stands for random access machine, meaning that every memory cell can be accessed in constant time. Not like, say, a list where one always has to start from the first element.

[2] In addition, sometimes also logarithms, other analytic functions, indirect addressing (integral), or floor and ceiling are used. As adding some of these operations makes the model more powerful, it is usually specified and emphasized explicitly when an algorithm uses them.

have to ensure that we do not abuse the power of this model. For instance, we may want to restrict the numbers that are manipulated by any single arithmetic operation to be bounded by some fixed polynomial in the numbers that appear in the input.

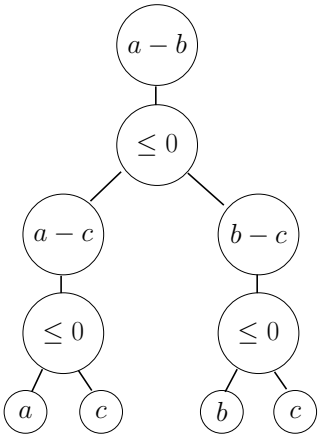
On the positive side, the real RAM model allows to abstract from the lowlands of numeric and algebraic computation and to concentrate on the algorithmic core from a combinatorial point of view.

But there are also downsides to using such a powerful model. In particular, it may be a challenge to efficiently implement a geometric algorithm designed for the real RAM on an actual computer. With bounded memory there is no way to represent general real numbers explicitly, and operations using a symbolic representation can hardly be considered constant time.

When interested in lower bounds, it is convenient to use a model of computation that encompasses and represents explicitly all possible execution paths of an algorithm. This is what the following model is about.

Algebraic Computation Trees (Ben-Or [1]). A computation is regarded as a binary tree.

- The leaves contain the (possible) results of the computation.
- Every node v with one child has an operation of the form $+, -, *, /, \sqrt{}, \ldots$ associated to it. The operands of this operation are constant input values, or among the ancestors of v in the tree.
- Every node v with two children has associated to it a branching of the form > 0, $\geqslant 0$, or $= 0$. The branch is with respect to the result of v's parent node. If the expression yields `true`, the computation continues with the left child of v; otherwise, it continues with the right child of v.



The term *decision tree* is used if all of the final results (leaves) are either `true` or `false`. If every branch is based on a linear function in the input values, we face a *linear decision tree*. Analogously one can define, say, quadratic decision trees.

The complexity of a computation or decision tree is the maximum number of vertices along any root-to-leaf path. It is well known that $\Omega(n \log n)$ comparisons are required to sort n numbers. But also for some problems that appear easier than sorting at first glance, the same lower bound holds. Consider, for instance, the following problem.

Element Uniqueness

Input: $\{x_1, \ldots, x_n\} \subset \mathbb{R}$, $n \in \mathbb{N}$.

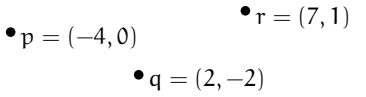
Output: Is $x_i = x_j$, for some $i, j \in \{1, \ldots, n\}$ with $i \neq j$?

Ben-Or [1] has shown that any algebraic decision tree to solve Element Uniqueness for n elements has complexity $\Omega(n \log n)$.

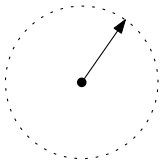
1.2 Basic Geometric Objects

We will mostly be concerned with the d-dimensional Euclidean space $\mathbb{R}^d$, for small $d \in \mathbb{N}$; typically, $d = 2$ or $d = 3$. The basic objects of interest in $\mathbb{R}^d$ are the following.

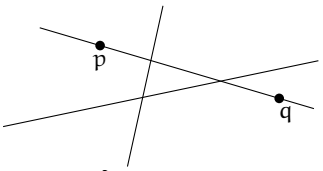
Points. A point p, typically described by its d Cartesian coordinates $p = (x_1, \ldots, x_d)$.



Directions. A vector $v \in \mathbb{S}^{d-1}$ (the $(d-1)$-dimensional unit sphere), typically described by its d Cartesian coordinates $v = (x_1, \ldots, x_d)$, with $\|v\| = \sqrt{\sum_{i=1}^{d} {x_i}^2} = 1$.

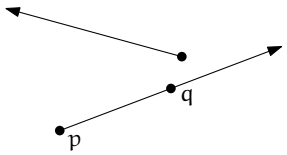


Lines. A line is a one-dimensional affine subspace. It can be described by two distinct points p and q as the set of all points r that satisfy $r = p + \lambda(q - p)$, for some $\lambda \in \mathbb{R}$.

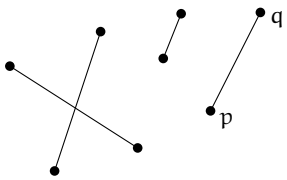


While any pair of distinct points defines a unique line, a line in $\mathbb{R}^2$ contains infinitely many points and so it may happen that a collection of three or more points lie on a line. Such a collection of points is termed *collinear*[3].

Rays. If we remove a single point from a line and take the closure of one of the connected components, then we obtain a ray. It can be described by two distinct points p and q as the set of all points r that satisfy $r = p + \lambda(q - p)$, for some $\lambda \geqslant 0$. The *orientation* of a ray is the direction $(q - p)/\|q - p\|$.



Line segment. A line segment is a compact connected subset of a line. It can be described by two points p and q as the set of all points r that satisfy $r = p + \lambda(q - p)$, for some $\lambda \in [0, 1]$. We will denote the line segment through p and q by $\overline{pq}$. Depending on the context we may allow or disallow *degenerate* line segments consisting of a single point only ($p = q$ in the above equation).



Hyperplanes. A hyperplane $\mathcal{H}$ is a $(d-1)$-dimensional affine subspace. It can be described algebraically by $d + 1$ coefficients $\lambda_1, \ldots, \lambda_{d+1} \in \mathbb{R}$, where $\|(\lambda_1, \ldots, \lambda_{d+1})\| = 1$, as the set of all points $(x_1, \ldots, x_d)$ that satisfy the linear equation $\mathcal{H} : \sum_{i=1}^{d} \lambda_i x_i = \lambda_{d+1}$.

[3] Not *colinear*, which refers to a notion in the theory of coalgebras.

If the above equation is converted into an inequality, we obtain the algebraic description of a *halfspace* (in $\mathbb{R}^2$: halfplane).

Spheres and balls. A sphere is the set of all points that are equidistant to a fixed point. It can be described by a point c (center) and a number $\rho \in \mathbb{R}$ (radius) as the set of all points p that satisfy $\|p - c\| = \rho$. The *ball* of radius ρ around p consists of all points p that satisfy $\|p - c\| \leqslant \rho$.

1.3 Graphs

In this section we review some basic definitions and properties of graphs. For more details and proofs, refer to any standard textbook on graph theory [2, 3, 5].

An (undirected) graph $G = (V, E)$ is defined on a set V of *vertices*. Unless explicitly stated otherwise, V is always finite. Vertices are associated to each other through *edges* which are collected in the set $E \subseteq \binom{V}{2}$. The two vertices defining an edge are *adjacent* to each other and *incident* to the edge.

For a vertex $v \in V$, denote by $N_G(v)$ the *neighborhood* of v in G, that is, the set of vertices from G that are adjacent to v. Similarly, for a set $W \subset V$ of vertices define $N_G(W) := \bigcup_{w \in W} N_G(w)$. The *degree* $\deg_G(v)$ of a vertex $v \in V$ is the size of its neighborhood, that is, the number of edges from E incident to v. The subscript is often omitted when it is clear which graph it refers to.

Lemma 1.1 (Handshaking Lemma) *In any graph* $G = (V, E)$ *we have* $\sum_{v \in V} \deg(v) = 2|E|$.

Two graphs $G = (V, E)$ and $H = (U, W)$ are *isomorphic* if there is a bijection $\phi : V \to U$ such that $\{u, v\} \in E \iff \{\phi(u), \phi(v)\} \in W$. Such a bijection ϕ is called an *isomorphism* between G and H. The structure of isomorphic graphs is identical and often we do not distinguish between them when looking at them as graphs.

For a graph G denote by $V(G)$ the set of vertices and by $E(G)$ the set of edges. A graph $H = (U, F)$ is a *subgraph* of G if $U \subseteq V$ and $F \subseteq E$. In case that $U = V$ the graph H is a *spanning* subgraph of G. For a set $W \subseteq V$ of vertices denote by $G[W]$ the *induced subgraph* of W in G, that is, the graph $(W, E \cap \binom{W}{2})$. For $F \subseteq E$ let $G \setminus F := (V, E \setminus F)$. Similarly, for $W \subseteq V$ let $G \setminus W := G[V \setminus W]$. In particular, for a vertex or edge $x \in V \cup E$ we write $G \setminus x$ for $G \setminus \{x\}$. The *union* of two graphs $G = (V, E)$ and $H = (W, F)$ is the graph $G \cup H := (V \cup W, E \cup F)$.

For an edge $e = \{u, v\} \in E$ the graph G/e is obtained from $G \setminus \{u, v\}$ by adding a new vertex w with $N_{G/e}(w) := (N_G(u) \cup N_G(v)) \setminus \{u, v\}$. This process is called *contraction* of e in G. Similarly, for a set $F \subseteq E$ of edges the graph G/F is obtained from G by contracting all edges from F.

Graph traversals. A *walk* in G is a sequence $W = (v_1, \ldots, v_k)$, $k \in \mathbb{N}$, of vertices such that v_i and v_{i+1} are adjacent in G, for all $1 \leqslant i < k$. The vertices v_1 and v_k are referred

to as the walk's *endpoints*, the other vertices are called *interior*. A walk with endpoints v_1 and v_k is sometimes referred to as a *walk between* v_1 and v_k. For a walk W denote by $V(W)$ its set of vertices and by $E(W)$ its set of edges (pairs of vertices adjacent along W). We say that W *visits* the vertices and edges in $V(W) \cup E(W)$. A walk for which both endpoints coincide, that is, $v_1 = v_k$, is called *closed*. Otherwise the walk is *open*.

If a walk uses each edge of G at most once, it is a *trail*. A closed walk that visits each edge and each vertex at least once is called a *tour* of G. An *Euler tour* is both a trail and a tour of G, that is, it visits each edge of G exactly once. A graph that contains an Euler tour is termed *Eulerian*.

If the vertices $v_1, \ldots, v_k$ of a closed walk W are pairwise distinct except for $v_1 = v_k$, then W is a *cycle* of size $k-1$. If the vertices $v_1, \ldots, v_k$ of a walk W are pairwise distinct, then W is a *path* of size k. A *Hamilton cycle (path)* is a cycle (path) that visits every vertex of G. A graph that contains a Hamilton cycle is *Hamiltonian*.

Two trails are *edge-disjoint* if they do not share any edge. Two paths are called (internally) *vertex-disjoint* if they do not share any vertices (except for possibly common endpoints). For two vertices $s, t \in V$ any path with endpoints s and t is called an (s,t)-*path* or a *path between* s and t.

Connectivity. Define an equivalence relation "$\sim$" on V by setting $a \sim b$ if and only if there is a path between a and b in G. The equivalence classes with respect to "$\sim$" are called *components* of G and their number is denoted by $\omega(G)$. A graph G is *connected* if $\omega(G) = 1$ and *disconnected*, otherwise.

A set $C \subset V$ of vertices in a connected graph $G = (V, E)$ is a *cut-set* of G if $G \setminus C$ is disconnected. A graph is *k-connected*, for a positive integer k, if $|V| \geqslant k+1$ and there is no cut-set of size less than k. Similarly a graph $G = (V, E)$ is *k-edge-connected*, if $G \setminus F$ is connected, for any set $F \subseteq E$ of less than k edges. Connectivity and cut-sets are related via the following well-known theorem.

Theorem 1.2 (Menger [4]) *For any two non-adjacent vertices u, v of a graph $G = (V, E)$, the size if a minimum cut that disconnects u and v is the same as the maximum number of pairwise internally vertex-disjoint paths between u and v.*

Specific families of graphs. A graph with a maximum number of edges, that is, $(V, \binom{V}{2})$, is called a *clique*. Up to isomorphism there is only one clique on n vertices; it is referred to as the *complete graph* K_n, $n \in \mathbb{N}$. At the other extreme, the *empty graph* $\overline{K_n}$ consists of n isolated vertices that are not connected by any edge. A set of U of vertices in a graph G is *independent* if $G[U]$ is an empty graph. A graph whose vertex set can be partitioned into at most two independent sets is *bipartite*. An equivalent characterization states that a graph is bipartite if and only if it does not contain any odd cycle. The bipartite graphs with a maximum number of edges (unique up to isomorphism) are the *complete bipartite graphs* $K_{m,n}$, for $m, n \in \mathbb{N}$. They consist of two disjoint independent sets of size m and n, respectively, and all mn edges in between.

A *forest* is a graph that is *acyclic*, that is, it does not contain any cycle. A connected forest is called *tree* and its *leaves* are the vertices of degree one. Every connected graph contains a spanning subgraph which is a tree, a so called *spanning tree*. Beyond the definition given above, there are several equivalent characterizations of trees.

Theorem 1.3 *The following statements for a graph* G *are equivalent.*

(1) G *is a tree (i.e., it is connected and acyclic).*

(2) G *is a connected graph with* n *vertices and* $n-1$ *edges.*

(3) G *is an acyclic graph with* n *vertices and* $n-1$ *edges.*

(4) Any two vertices in G *are connected by a unique path.*

(5) G *is minimally (edge-)connected, that is,* G *is connected but removal of any single edge yields a disconnected graph.*

(6) G *is maximally acyclic, that is,* G *is acyclic but adding any single edge creates a cycle.*

Directed graphs. In a directed graph or, short, *digraph* $D = (V, E)$ the set E consists of ordered pairs of vertices, that is, $E \subseteq V^2$. The elements of E are referred to as *arcs*. An arc $(u, v) \in E$ is said to be directed from its *source* u to its *target* v. For $(u, v) \in E$ we also say "there is an arc from u to v in D". Usually, we consider *loop-free* graphs, that is, arcs of the type (v, v), for some $v \in V$, are not allowed.

The *in-degree* $\deg^-_D(v) := |\{(u, v) \mid (u, v) \in E\}|$ of a vertex $v \in V$ is the number of *incoming* arcs at v. Similarly, the *out-degree* $\deg^+_D(v) := |\{(v, u) \mid (v, u) \in E\}|$ of a vertex $v \in V$ is the number of *outgoing* arcs at v. Again the subscript is often omitted when the graph under consideration is clear from the context.

From any undirected graph G one can obtain a digraph on the same vertex set by specifying a direction for each edge of G. Each of these $2^{|E(G)|}$ different digraphs is called an *orientation* of G. Similarly every digraph $D = (V, E)$ has an *underlying* undirected graph $G = (V, \{\{u, v\} \mid (u, v) \in E \text{ or } (v, u) \in E\})$. Hence most of the terminology for undirected graphs carries over to digraphs.

A *directed walk* in a digraph D is a sequence $W = (v_1, \ldots, v_k)$, for some $k \in \mathbb{N}$, of vertices such that there is an arc from v_i to v_{i+1} in D, for all $1 \leqslant i < k$. In the same way we define *directed trails*, *directed paths*, *directed cycles*, and *directed tours*.

References

[1] Michael Ben-Or, Lower bounds for algebraic computation trees. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pp. 80–86, 1983, URL `http://dx.doi.org/10.1145/800061.808735`.

[2] John Adrian Bondy and U. S. R. Murty, *Graph Theory*, vol. 244 of *Graduate texts in Mathematics*. Springer-Verlag, New York, 2008, URL http://dx.doi.org/10.1007/978-1-84628-970-5.

[3] Reinhard Diestel, *Graph Theory*. Springer-Verlag, Heidelberg, 4th edn., 2010.

[4] Karl Menger, Zur allgemeinen Kurventheorie. *Fund. Math.*, **10**, 1, (1927), 96—-115, URL http://matwbn.icm.edu.pl/ksiazki/fm/fm10/fm1012.pdf.

[5] Douglas B. West, *An Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ, 2nd edn., 2001.

Chapter 2

Plane Embeddings

In this chapter we investigate properties of plane embeddings and under which conditions they hold.

2.1 Embeddings and planarity

A *curve* is a set $C \subset \mathbb{R}^2$ that is of the form $\{\gamma(t) \mid 0 \leqslant t \leqslant 1\}$, where $\gamma : [0,1] \to \mathbb{R}^2$ is a continuous function. The function γ is called a *parameterization* of C. The points $\gamma(0)$ and $\gamma(1)$ are the *endpoints* of the curve. For a *closed* curve, we have $\gamma(0) = \gamma(1)$. A curve is *simple*, if it admits a parameterization γ that is injective on $[0,1]$. For a closed simple curve we allow as an exception that $\gamma(0) = \gamma(1)$. The following famous theorem describes an important property of the plane. A proof can, for instance, be found in the book of Mohar and Thomassen [15].

Theorem 2.1 (Jordan) *Any simple closed curve C partitions the plane into exactly two regions (connected open sets), each bounded by C.*

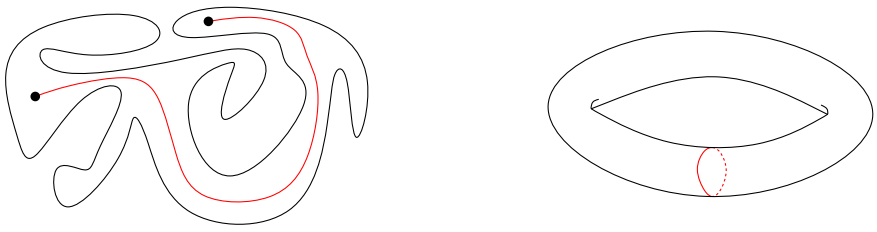


Figure 2.1: *A Jordan curve and two points in one of its faces (left); a simple closed curve that does not disconnect the torus (right).*

Observe that, for instance, on the torus there are closed curves that do not disconnect the surface (and so the theorem does not hold there).

An *embedding* or *drawing* of a (multi-)graph $G = (V, E)$ into the plane is a function $f : V \cup E \to \mathbb{R}^2$ that assigns

- a point $f(v)$ to every vertex $v \in V$ and
- a simple curve $f(\{u, v\})$ with endpoints $f(u)$ and $f(v)$ to every edge $\{u, v\} \in E$,

such that f is injective on V and $f(\{u, v\}) \cap f(V) = \{f(u), f(v)\}$, for every edge $\{u, v\} \in E$. A common point $f(e) \cap f(e')$ between two curves that represent edges $e \neq e' \in E$ is called a *crossing*, unless it is a common endpoint of e and e'. In many cases it is convenient to demand that no three edges share a crossing.

Planar vs. plane. A multigraph is *planar* if it admits an embedding without crossings into the plane. Such an embedding is also called a *plane* or *crossing-free* embedding. A planar graph together with a particular plane embedding is called a *plane* graph. Note the distinction between "planar" and "plane": the former indicates the possibility of an embedding, whereas the latter refers to a concrete embedding (Figure 2.2).

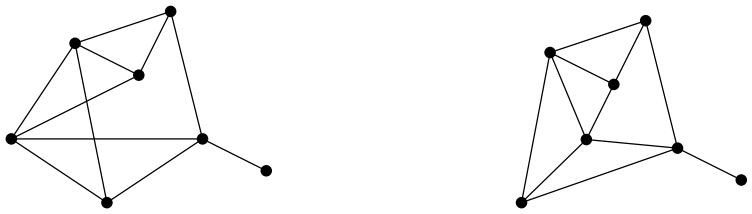


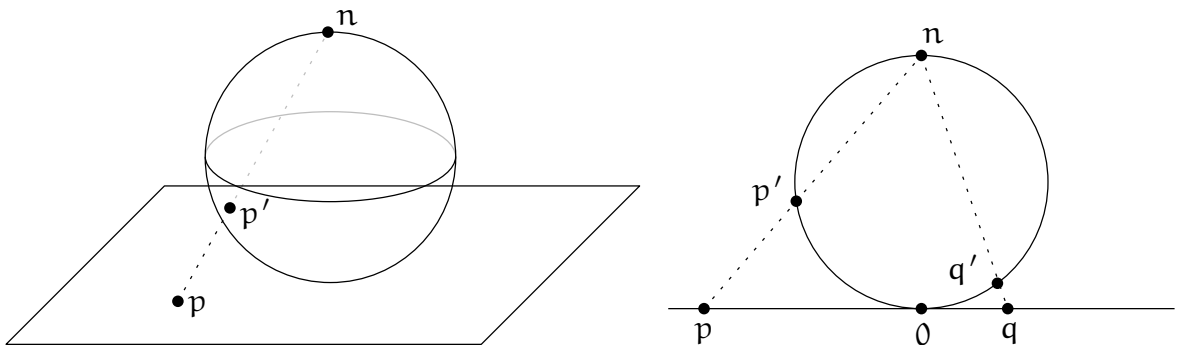
Figure 2.2: *A planar graph (left) and a plane drawing of it (right).*

A *geometric graph* is a graph together with an embedding, in which all edges are realized as straight-line segments. Note that such an embedding is completely defined by the mapping for the vertices. A plane geometric graph is also called a *plane straight-line graph* (PSLG). In contrast, a plane graph in which the edges may form arbitrary simple curves is called a *topological plane graph*.

The *faces* of a plane multigraph are the maximally connected regions of the plane that do not contain any point used by the embedding (as the image of a vertex or an edge). Each embedding of a finite multigraph has exactly one *unbounded face*, also called *outer* or *infinite* face. Using stereographic projection, it is not hard to show that the role of the unbounded face is not as special at it may seem at first glance.

Theorem 2.2 *If a graph G has a plane embedding in which some face is bounded by the cycle $(v_1, \ldots, v_k)$, then G also has a plane embedding in which the unbounded face is bounded by the cycle $(v_1, \ldots, v_k)$.*

Proof. (Sketch) Take a plane embedding Γ of G and map it to the sphere using *stereographic projection*: Imagine $\mathbb{R}^2$ being the x/y-plane in $\mathbb{R}^3$ and place a unit sphere S such that its south pole touches the origin. We obtain a bijective continuous mapping between $\mathbb{R}^2$ and $S \setminus \{n\}$, where n is the north pole of S, as follows: A point $p \in \mathbb{R}^2$ is mapped to the point p' that is the intersection of the line through p and n with S, see Figure 2.3.



(a) Three-dimensional view. (b) Cross-section view.

Figure 2.3: *Stereographic projection.*

Consider the resulting embedding Γ' of G on S: The infinite face of Γ corresponds to the face of Γ' that contains the north pole n of S. Now rotate the embedding Γ' on S such that the desired face contains n. Mapping back to the plane using stereographic projection results in an embedding in which the desired face is the outer face. □

Exercise 2.3 *Consider a graph* G *with the plane embedding depicted in Figure 2.4. Give a plane embedding of* G *in which the cycle* $1, 2, 3$ *bounds the outer face.*

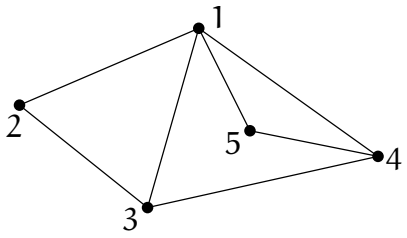


Figure 2.4: *Plane embedding of* G.

Duality. Every plane graph G has a *dual* G^*, whose vertices are the faces of G and two are connected by an edge in G^*, if and only if they have a common edge in G. In general, G^* is a multigraph (may contain loops and multiple edges) and it depends on the embedding. That is, an abstract planar graph G may have several non-isomorphic duals. If G is a connected plane graph, then $(G^*)^* = G$. We will show later in Section 2.3 that the dual of a 3-connected planar is unique (up to isomorphism).

The Euler Formula and its ramifications. One of the most important tools for planar graphs (and more generally, graphs embedded on a surface) is the Euler–Poincaré Formula.

Theorem 2.4 (Euler's Formula) *For every connected plane graph with* n *vertices,* e *edges, and* f *faces, we have* $n - e + f = 2$.

Figure 2.5: *Two plane drawings and their duals for the same planar graph.*

In particular, this shows that for any planar graph the number of faces is the same in every plane embedding. Therefore, the number of faces is actually a parameter of an abstract planar graph. It also follows (stated below as a corollary) that planar graphs are *sparse*, that is, they have a linear number of edges (and faces) only. So the asymptotic complexity of a planar graph is already determined by its number of vertices.

Corollary 2.5 *A simple planar graph on $n \geqslant 3$ vertices has at most $3n-6$ edges and at most $2n-4$ faces.*

Proof. Consider a simple planar graph G on $n \geqslant 3$ vertices. Without loss of generality we may assume that G is connected. (If not, add edges between components of G until the graph is connected. The number of faces remains unchanged and the number of edges only increases.) Consider a plane drawing of G and denote by E the set of edges and by F the set of faces of G. Let

$$X = \{(e, f) \in E \times F \mid e \text{ bounds } f\}$$

denote the set of incident edge-face pairs. We count X in two different ways.

First note that each edge bounds at most two faces and so $|X| \leqslant 2 \cdot |E|$.

Second note that in a simple connected planar graph on three or more vertices every face is bounded by at least three vertices. Therefore $|X| \geqslant 3 \cdot |F|$.

Using Euler's Formular we conclude that

$$\begin{aligned} 4 &= 2n - 2|E| + 2|F| \leqslant 2n - 3|F| + 2|F| = 2n - |F| \text{ and} \\ 6 &= 3n - 3|E| + 3|F| \leqslant 3n - 3|E| + 2|E| = 3n - |E| , \end{aligned}$$

which yields the claimed bounds. □

It also follows that the degree of a "typical" vertex in a planar graph is a small constant. There exist several variations of this statement, a few more of which we will encounter during this course.

Corollary 2.6 *The average vertex degree in a simple planar graph is less than six.*

Exercise 2.7 *Prove Corollary 2.6.*

Exercise 2.8 *Show that neither K_5 (the complete graph on five vertices) nor $K_{3,3}$ (the complete bipartite graph where both classes have three vertices) is planar.*

Characterizing planarity. The classical theorems of Kuratowski and Wagner provide a characterization of planar graphs in terms of forbidden sub-structures. A *subdivision* of a graph $G = (V, E)$ is a graph that is obtained from G by replacing each edge with a path.

Theorem 2.9 (Kuratowski [13, 19]) *A graph is planar if and only if it does not contain a subdivision of* $K_{3,3}$ *or* K_5.

A *minor* of a graph $G = (V, E)$ is a graph that is obtained from G using zero or more edge contractions, edge deletions, and/or vertex deletions.

Theorem 2.10 (Wagner [22]) *A graph is planar if and only if it does not contain* $K_{3,3}$ *or* K_5 *as a minor.*

In some sense, Wagner's Theorem is a special instance[1] of a much more general theorem.

Theorem 2.11 (Graph Minor Theorem, Robertson/Seymour [17]) *Every minor-closed family of graphs can be described in terms of a finite set of forbidden minors.*

Being *minor-closed* means that for every graph from the family also all of its minors belong to the family. For instance, the family of planar graphs is minor-closed because planarity is preserved under removal of edges and vertices and under edge contractions. The Graph Minor Theorem is a celebrated result that was established by Robertson and Seymour in a series of twenty papers, see also the survey by Lovász [14]. They also described an $O(n^3)$ algorithm (with horrendous constants, though) to decide whether a graph on n vertices contains a fixed (constant-size) minor. Later, Kawarabayashi et al. [11] showed that this problem can be solved in $O(n^2)$ time. As a consequence, every minor-closed property can be decided in polynomial time.

Unfortunately, the result is non-constructive in the sense that in general we do not know how to obtain the set of forbidden minors for a given family/property. For instance, for the family of toroidal graphs (graphs that can be embedded without crossings on the torus) more than 16'000 forbidden minors are known, and we do not know how many there are in total. So while we know that there exists a cubic time algorithm to test membership for minor-closed families, we have no idea what such an algorithm looks like in general.

Graph families other than planar graphs for which the forbidden minors are known include forests (K_3) and outerplanar graphs ($K_{2,3}$ and K_4). A graph is *outerplanar* if it admits a plane drawing such that all vertices appear on the outer face (Figure 2.6).

Exercise 2.12 *(a) Give an example of a 6-connected planar graph or argue that no such graph exists.*

[1] Strictly speaking, it is more than just a special instance because it also specifies the forbidden minors explicitly.

Figure 2.6: *An outerplanar graph (left) and a plane drawing of it in which all vertices are incident to the outer face (right).*

(b) Give an example of a 5-connected planar graph or argue that no such graph exists.

(c) Give an example of a 3-connected outerplanar graph or argue that no such graph exists.

Planarity testing. For planar graphs we do not have to contend ourselves with a cubic-time algorithm, as there are several approaches to solve the problem in linear time. In fact, there is quite a number of papers that describe different linear time algorithms, all of which—from a very high-level point of view—can be regarded as an annotated depth-first-search. The first such algorithm was described by Hopcroft and Tarjan [10], while the current state-of-the-art [25] is probably among the "path searching" method by Boyer and Myrwold [3] and the "LR-partition" method by de Fraysseix et al [7]. Although the overall idea in all these approaches is easy to convey, there are many technical details, which make a in-depth discussion rather painful to go through.

2.2 Graph representations

There are two standard representations for an abstract graph $G = (V, E)$ on $n = |V|$ vertices. For the *adjacency matrix* representation we consider the vertices to be ordered as $V = \{v_1, \ldots, v_n\}$. The adjacency matrix of an undirected graph is a symmetric $n \times n$-matrix $A = (a_{ij})_{1 \leqslant i,j \leqslant n}$ where $a_{ij} = a_{ji} = 1$, if $\{i, j\} \in E$, and $a_{ij} = a_{ji} = 0$, otherwise. Storing such a matrix explicitly requires $\Omega(n^2)$ space, and allows to test in constant time whether or not two given vertices are adjacent.

In an *adjacency list* representation, we store for each vertex a list of its neighbors in G. This requires only $O(n+|E|)$ storage, which is better than for the adjacency matrix in case that $|E| = o(n^2)$. On the other hand, the adjacency test for two given vertices is not a constant-time operation, because it requires a search in one of the lists. Depending on the representation of these lists, such a search takes $O(d)$ time (unsorted list) or $O(\log d)$ time (sorted random-access representation, such as a balanced search tree), where d is the minimum degree of the two vertices.

Both representations have their merits. The choice of which one to use (if any) typically depends on what one wants to do with the graph. When dealing with embedded graphs, however, additional information concerning the embedding is needed beyond

the pure incidence structure of the graph. The next section discusses a standard data structure to represent embedded graphs.

2.2.1 The Doubly-Connected Edge List

The *doubly-connected edge list* (DCEL) is a data structure to represent a plane graph in such a way that it is easy to traverse and to manipulate. In order to avoid unnecessary complications, let us discuss only connected graphs here that contain at least two vertices. It is not hard to extend the data structure to cover all plane graphs. For simplicity we also assume that we deal with a straight-line embedding and so the geometry of edges is defined by the mapping of their endpoints already. For more general embeddings, the geometric description of edges has to be stored in addition.

The main building block of a DCEL is a list of *halfedges*. Every actual edge is represented by two halfedges going in opposite direction, and these are called *twins*, see Figure 2.7. Along the boundary of each face, halfedges are oriented counterclockwise.

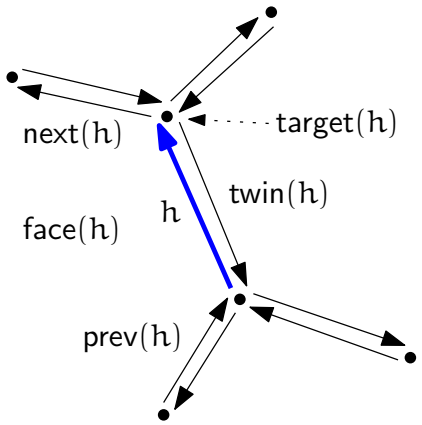


Figure 2.7: *A halfedge in a DCEL.*

A DCEL stores a list of halfedges, a list of vertices, and a list of faces. These lists are unordered but interconnected by various pointers. A vertex v stores a pointer $\mathsf{halfedge}(v)$ to an arbitrary halfedge originating from v. Every vertex also knows its coordinates, that is, the point $\mathsf{point}(v)$ it is mapped to in the represented embedding. A face f stores a pointer $\mathsf{halfedge}(f)$ to an arbitrary halfedge within the face. A halfedge h stores *five* pointers:

- a pointer $\mathsf{target}(h)$ to its target vertex,
- a pointer $\mathsf{face}(h)$ to the incident face,
- a pointer $\mathsf{twin}(h)$ to its twin halfedge,
- a pointer $\mathsf{next}(h)$ to the halfedge following h along the boundary of $\mathsf{face}(h)$, and

- a pointer $\mathsf{prev}(h)$ to the halfedge preceding h along the boundary of $\mathsf{face}(h)$.

A constant amount of information is stored for every vertex, (half-)edge, and face of the graph. Therefore the whole DCEL needs storage proportional to $|V| + |E| + |F|$, which is $O(n)$ for a plane graph with n vertices by Corollary 2.5.

This information is sufficient for most tasks. For example, traversing all edges around a face f can be done as follows:

$s \leftarrow \mathsf{halfedge}(f)$
$h \leftarrow s$
do
    something with h
    $h \leftarrow \mathsf{next}(h)$
while $h \neq s$

Exercise 2.13 *Give pseudocode to traverse all edges incident to a given vertex v of a DCEL.*

Exercise 2.14 *Why is the previous halfedge* $\mathsf{prev}(\cdot)$ *stored explicitly and the source vertex of a halfedge is not?*

2.2.2 Manipulating a DCEL

In many applications, plane graphs appear not just as static objects but rather they evolve over the course of an algorithm. Therefore the data structure used to represent the graph must allow for efficient update operations to change it.

First of all, we need to be able to generate new vertices, edges, and faces, to be added to the corresponding list within the DCEL and—symmetrically—the ability to delete an existing entity. Then it should be easy to add a new vertex v to the graph within some face f. As we maintain a connected graph, we better link the new vertex to somewhere, say, to an existing vertex u. For such a connection to be possible, we require that the open line segment uv lies completely in f.

Of course, two halfedges are to be added connecting u and v. But where exactly? Given that from a vertex and from a face only some arbitrary halfedge is directly accessible, it turns out convenient to use a halfedge in the interface. Let h denote the halfedge incident to f for which $\mathsf{target}(h) = u$. Our operation then becomes (see also Figure 2.8)

$\mathsf{add\text{-}vertex\text{-}at}(v, h)$
Precondition: the open line segment $\overline{\mathsf{point}(v)\mathsf{point}(u)}$, where $u := \mathsf{target}(h)$, lies completely in $f := \mathsf{face}(h)$.
Postcondition: a new vertex v has been inserted into f, connected by an edge to u.

and it can be realized by manipulating a constant number of pointers as follows.

(a) before (b) after

Figure 2.8: *Add a new vertex connected to an existing vertex* u.

add-vertex-at(v, h) {
  $h_1 \leftarrow$ a new halfedge
  $h_2 \leftarrow$ a new halfedge
  halfedge$(v) \leftarrow h_2$
  twin$(h_1) \leftarrow h_2$
  twin$(h_2) \leftarrow h_1$
  target$(h_1) \leftarrow v$
  target$(h_2) \leftarrow u$
  face$(h_1) \leftarrow f$
  face$(h_2) \leftarrow f$
  next$(h_1) \leftarrow h_2$
  next$(h_2) \leftarrow$ next(h)
  prev$(h_1) \leftarrow h$
  prev$(h_2) \leftarrow h_1$
  next$(h) \leftarrow h_1$
  prev(next$(h_2)) \leftarrow h_2$
}

Similarly, it should be possible to add an edge between two existing vertices u and v, provided the open line segment uv lies completely within a face f of the graph, see Figure 2.9. Since such an edge insertion splits f into two faces, the operation is called *split-face*. Again we use the halfedge h that is incident to f and for which target$(h) = u$. Our operation becomes then

split-face(h, v)
Precondition: v is incident to $f :=$ face(h) but not adjacent to $u :=$ target(h).
  The open line segment $\overline{\text{point}(v)\text{point}(u)}$ lies completely in f.
Postcondition: f has been split by a new edge uv.

The implementation is slightly more complicated compared to add-vertex-at above, because the face f is destroyed and so we have to update the face information of all incident halfedges. In particular, this is not a constant time operation, but its time complexity is proportional to the size of f.

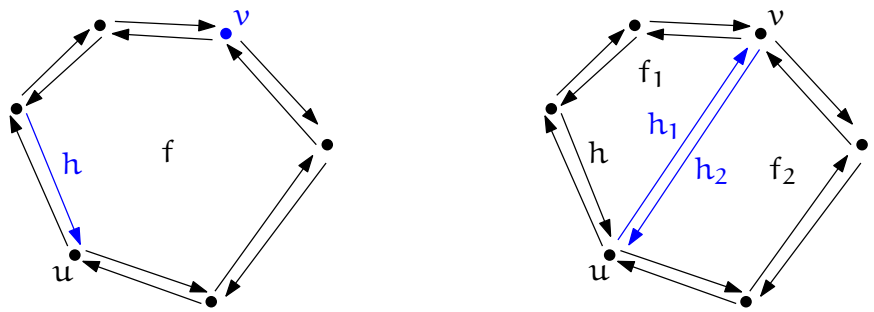


Figure 2.9: *Split a face by an edge* uv.

```
split-face(h, v) {
    f1 ← a new face
    f2 ← a new face
    h1 ← a new halfedge
    h2 ← a new halfedge
    halfedge(f1) ← h1
    halfedge(f2) ← h2
    twin(h1) ← h2
    twin(h2) ← h1
    target(h1) ← v
    target(h2) ← u
    next(h2) ← next(h)
    prev(next(h2)) ← h2
    prev(h1) ← h
    next(h) ← h1
    i ← h2
    loop
        face(i) ← f2
        if target(i) = v break the loop
        i ← next(i)
    endloop
    next(h1) ← next(i)
    prev(next(h1)) ← h1
    next(i) ← h2
    prev(h2) ← i
    i ← h1
    do
        face(i) ← f1
        i ← next(i)
    until target(i) = u
```

delete the face f
}

In a similar fashion one can realize the inverse operation join-face(h) that removes the edge (represented by the halfedge) h, thereby joining the faces face(h) and face(twin(h)).

It is easy to see that every connected plane graph on at least two vertices can be constructed using the operations add-vertex-at and split-face, starting from an embedding of K_2 (two vertices connected by an edge).

Exercise 2.15 *Give pseudocode for the operation join-face*(h). *Also specify preconditions, if needed.*

Exercise 2.16 *Give pseudocode for the operation split-edge*(h), *that splits the edge (represented by the halfedge)* h *into two by a new vertex* w, *see Figure 2.10.*

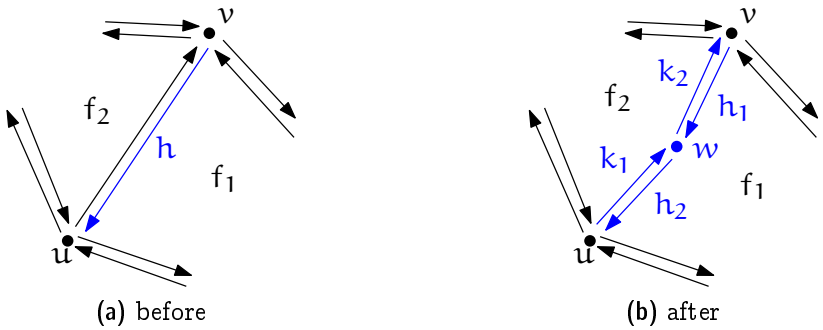


Figure 2.10: *Split an edge by a new vertex.*

2.2.3 Graphs with unbounded edges

In some cases it is convenient to consider plane graphs, in which some edges are not mapped to a line segment but to an unbounded curve, such as a ray. This setting is not really much different from the one we studied before, except that one vertex is placed "at infinity". One way to think of it is in terms of *stereographic projection* (see the proof of Theorem 2.2). The further away a point in $\mathbb{R}^2$ is from the origin, the closer its image on the sphere S gets to the north pole n of S. But there is no way to reach n except in the limit. Therefore, we can imagine drawing the graph on S instead of in $\mathbb{R}^2$ and putting the "infinite vertex" at n.

All this is just for the sake of a proper geometric interpretation. As far as a DCEL representation of such a graph is concerned, there is no need to consider spheres or, in fact, anything beyond what we have discussed before. The only difference to the case with all finite edges is that there is this special infinite vertex, which does not have any point/coordinates associated to it. But other than that, the infinite vertex is treated in exactly the same way as the finite vertices: it has in– and outgoing halfedges along which the unbounded faces can be traversed (Figure 2.11).

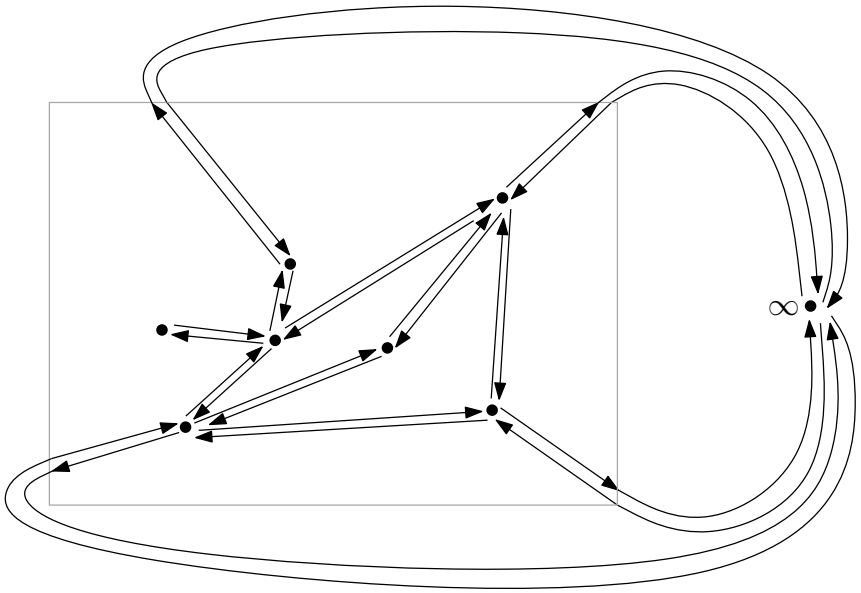


Figure 2.11: *A DCEL with unbounded edges. Usually, we will not show the infinite vertex and draw all edges as straight-line segments. This yields a geometric drawing, like the one within the gray box.*

Remarks. It is actually not so easy to point exactly to where the DCEL data structure originates from. Often Muller and Preparata [16] are credited, but while they use the term DCEL, the data structure they describe is different from what we discussed above and from what people usually consider a DCEL nowadays. Overall, there are a large number of variants of this data structure, which appear under the names *winged edge* data structure [1], *halfedge* data structure [23], or *quad-edge* data structure [9]. Kettner [12] provides a comparison of all these and some additional references.

2.2.4 Combinatorial embeddings

The DCEL data structure discussed in the previous section provides a fully fleshed-out representation of what is called a *combinatorial embedding*. From a mathematical point of view this can be regarded an equivalence relation on embeddings: Two embeddings are equivalent if their face boundaries—regarded as circular sequences of edges (or vertices) in counterclockwise order—are the same (as sets) up to a global change of orientation (reversing the order of all sequences simultaneously). For instance, the faces of the plane graphs shown in Figure 2.12a are (described as a list of vertices)

(a) : $\{(1,2,3),(1,3,6,4,5,4),(1,4,6,3,2)\}$,
(b) : $\{(1,2,3,6,4,5,4),(1,3,2),(1,4,6,3)\}$, and
(c) : $\{(1,4,5,4,6,3),(1,3,2),(1,2,3,6,4)\}$.

Note that a vertex can appear several times along the boundary of a face (if it is a cut-vertex). Clearly (b) is not equivalent to (a) nor (c), because it is the only graph that contains a face bounded by seven vertices. However, (a) and (c) turn out to be equivalent: after reverting orientations f_1 takes the role of h_2, f_2 takes the role of h_1, and f_3 takes the role of h_3.

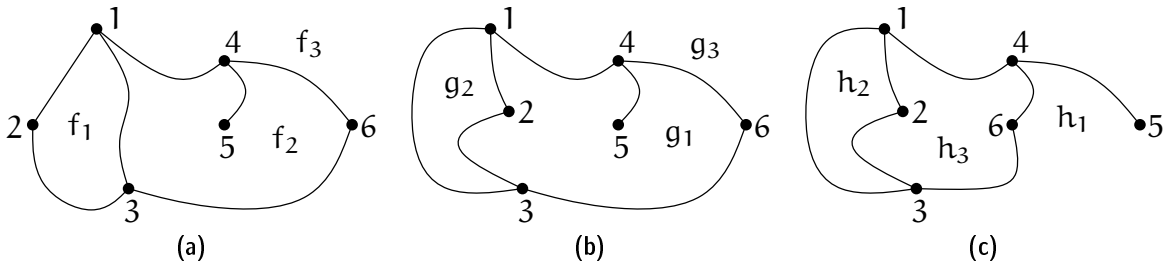


Figure 2.12: *Equivalent embeddings?*

In a dual interpretation one can just as well define equivalence in terms of the cyclic order of neighbors around all vertices. In this form, a compact way to describe a combinatorial embedding is as a so-called *rotation system* that consists of a permutation π and an involution ρ, both of which are defined on the set of halfedges (in this context often called *darts* or *flags*) of the embedding. The orbits of π correspond to the vertices, as they iterate over the incident halfedges. The involution ρ maps each halfedge to its twin.

Many people prefer this dual view, because one does not have to discuss the issue of vertices or edges that appear several times on the boundary of a face. The following lemma shows that such an issue does not arise when dealing with biconnected graphs.

Lemma 2.17 *In a biconnected plane graph every face is bounded by a cycle.*

We leave the proof as an exercise. Intuitively the statement is probably clear. But we believe it is instructive to think about how to make a formal argument. An easy consequence is the following corollary, whose proof we also leave as an exercise.

Corollary 2.18 *In a 3-connected plane graph the neighbors of a vertex lie on a cycle.*

Note that the statement does not read "form a cycle" but rather "lie on a cycle".

Exercise 2.19 *Prove Lemma 2.17 and Corollary 2.18.*

2.3 Unique embeddings

We have seen in Lemma 2.17 that all faces in biconnected plane graphs are bounded by cycles. Conversely one might wonder which cycles of a planar graph G bound a face in *some* plane embedding of G. Such a cycle is called a *facial cycle* (Figure 2.13).

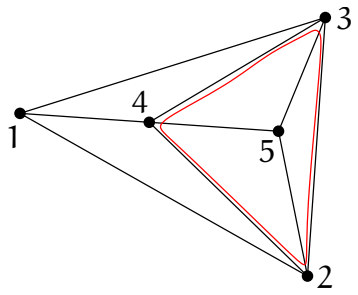


Figure 2.13: *The cycle* $(1, 2, 3)$ *is facial and we can show that* $(2, 3, 4)$ *is not.*

In fact, we will look at a slightly different class of cycles, namely those that bound a face in *every* plane embedding of G. The lemma below provides a complete characterization of those cycles. In order to state it, let us introduce a bit more terminology. A *chord* of a cycle C in a graph G is an edge that connects two vertices of C but is not an edge of C. A cycle C in a graph G is an *induced cycle*, if $C = G[V(C)]$, that is, C does not have any chord in G.

Lemma 2.20 *Let* C *be a cycle in a planar graph* G *such that* $G \neq C$ *and* G *is not* C *plus a single chord of* C*. Then* C *bounds a face in every plane embedding of* G *if and only if* C *is an induced cycle and it is not separating (i.e.,* $G \setminus C$ *is connected).*

Proof. "$\Leftarrow$": Consider any plane embedding Γ of G. As $G \setminus C$ is connected, by the Jordan Curve Theorem it is contained either in the interior of C or in the exterior of C in Γ. In either case, the other component of the plane is bounded by C, because there are no edges among the vertices of C.

"$\Rightarrow$": Using contraposition, suppose that C is not induced or $G \setminus C$ is disconnected. We have to show that there exists a plane embedding of G in which C does not bound a face.

If C is not induced, then there is a chord c of C in G. As $G \neq C \cup c$, either G has a vertex v that is not in C or G contains another chord $d \neq c$ of C. In either case, consider any plane embedding Γ of G in which C bounds a face. (If such an embedding does not exist, there is nothing to show.) We can modify Γ by drawing the chord c in the face bounded by C to obtain an embedding Γ' of G in which C does not bound a face: one of the two regions bounded by C according to the Jordan Curve Theorem contains c and the other contains either the vertex v or the other chord d.

If $G \setminus C$ contains two components A and B, then consider a plane embedding Γ of G. If C is not a face in Γ, there is nothing to show. Hence suppose that C is a face of Γ (Figure 2.14a). From Γ we obtain induced plane embeddings Γ_A of $G \setminus B = A \cup C$ and Γ_B of $G \setminus A = B \cup C$. Using Theorem 2.2 we may suppose that C bounds the outer face in Γ_A and it does not bound the outer face in Γ_B. Then we can glue both embeddings at C, that is, extend Γ_B to an embedding of G by adding Γ_A within the face bounded by C (Figure 2.14b). The resulting embedding is a plane drawing of G in which C does not bound a face.

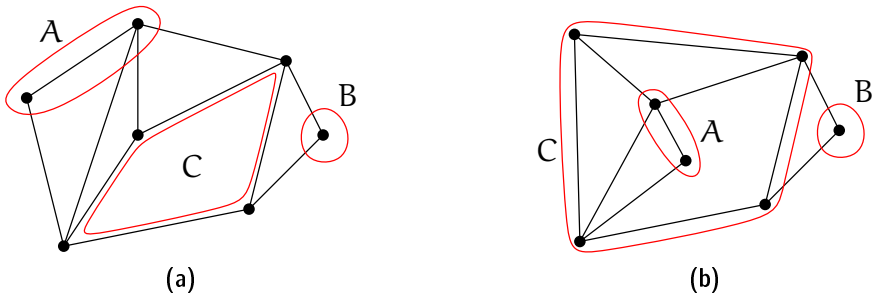


Figure 2.14: *Construct a plane embedding of* G *in which* C *does not bound a face.*

Finally, consider the case that $G \setminus C = \emptyset$ (which is not a connected graph according to our definition). As we considered above the case that C is not an induced cycle, the only remaining case is $G = C$, which is excluded explicitly. □

For both special cases for G that are excluded in Lemma 2.20 it is easy to see that all cycles in G bound a face in every plane embedding. This completes the characterization. Also observe that in these special cases G is not 3-connected.

Corollary 2.21 *A cycle* C *of a* 3*-connected planar graph* G *bounds a face in every plane embedding of* G *if and only if* C *is an induced cycle and it is not separating.* □

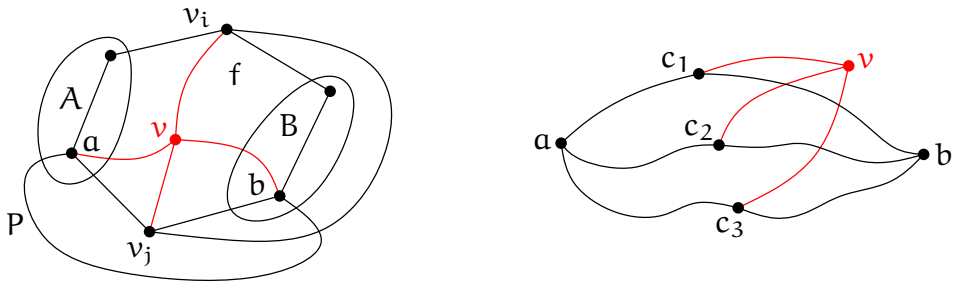
The following theorem tells us that for a wide range of graphs we have little choice as far as a plane embedding is concerned, at least from a combinatorial point of view. Geometrically, there is still a lot of freedom, though.

Theorem 2.22 (Whitney [24]) *A* 3*-connected planar graph has a unique combinatorial plane embedding (up to equivalence).*

Proof. Let G be a 3-connected planar graph and suppose there exist two embeddings Φ_1 and Φ_2 of G that are not equivalent. That is, there is a cycle $C = (v_1, \ldots, v_k)$, $k \geqslant 3$, in G that bounds a face in, say, Φ_1 but C does not bound a face in Φ_2. By Corollary 2.21 such a cycle has a chord or it is separating. We consider both options.

Case 1: C has a chord $\{v_i, v_j\}$, with $j \geqslant i + 2$. Denote $A = \{v_x \mid i < x < j\}$ and $B = \{v_x \mid x < i \vee j < x\}$ and observe that both A and B are non-empty (because $\{v_i, v_j\}$ is a chord and so v_i and v_j are not adjacent in C). Given that G is 3-connected, there is at least one path P from A to B that does not use either of v_i or v_j. Let a denote the last vertex of P that is in A, and let b denote the first vertex of B that is in b. As C bounds a face f in Φ_1, we can add a new vertex v inside the face bounded by C and connect v by four pairwise internally disjoint curves to each of v_i, v_j, a, and b. The result is a plane graph $G' \supset G$ that contains a subdivision of K_5 with branch vertices v, v_i, v_j, a, and b. By Kuratowski's Theorem (Theorem 2.9) this contradicts the planarity of G′.

Case 2: C is separating and, therefore, $G \setminus C$ contains two distinct components A and B. (We have $G \neq C$ because G is 3-connected.) Consider now the embedding Φ_1



(a) Case 1. (b) Case 2.

Figure 2.15: *Illustration of the two cases in Theorem 2.22.*

in which C bounds a face, without loss of generality (Theorem 2.2) a bounded face f. Hence both A and B are embedded in the exterior of f.

Choose vertices $a \in A$ and $b \in B$ arbitrarily. As G is 3-connected, by Menger's Theorem (Theorem 1.2), there are at least three pairwise internally vertex-disjoint paths from a to b. Fix three such paths $\alpha_1, \alpha_2, \alpha_3$ and denote by c_i the first point of α_i that is on C, for $1 \leqslant i \leqslant 3$. Note that c_1, c_2, c_3 are well defined, because C separates A and B, and they are pairwise distinct. Therefore, $\{a, b\}$ and $\{c_1, c_2, c_3\}$ are branch vertices of a $K_{2,3}$ subdivision in G. We can add a new vertex v inside the face bounded by C and connect v by three pairwise internally disjoint curves to each of c_1, c_2, and c_3. The result is a plane graph $G' \supset G$ that contains a $K_{3,3}$ subdivision. By Kuratowski's Theorem (Theorem 2.9) this contradicts the planarity of G'.

In both cases we arrived at a contradiction and so there does not exist such a cycle C. Thus Φ_1 and Φ_2 are equivalent. □

Whitney's Theorem does not provide a characterization of unique embeddability, because there are both biconnected graphs that have a unique plane embedding (such as cycles) and biconnected graphs that admit several non-equivalent plane embeddings (for instance, a triangulated pentagon).

2.4 Triangulating a plane graph

A large and important class of 3-connected graphs is formed by the maximal planar graphs. A graph is *maximal planar* if no edge can be added so that the resulting graph is still planar.

Lemma 2.23 *A maximal planar graph on $n \geqslant 3$ vertices is biconnected.*

Proof. Consider a maximal planar graph $G = (V, E)$. If G is not biconnected, then it has a cut-vertex v. Take a plane drawing Γ of G. As $G \setminus v$ is disconnected, removal of v also splits $N_G(v)$ into at least two components. Therefore, there are two vertices $a, b \in N_G(v)$ that are adjacent in the circular order of vertices around v in Γ and are in

different components of $G \setminus v$. In particular, $\{a, b\} \notin E$ and we can add this edge to G (routing it very close to the path (a, v, b) in Γ) without violating planarity. This is in contradiction to G being maximal planar and so G is biconnected. □

Lemma 2.24 *In a maximal planar graph on $n \geqslant 3$ vertices, all faces are topological triangles, that is, each is bounded by exactly three edges.*

Proof. Consider a maximal planar graph $G = (V, E)$ and a plane drawing Γ of G. By Lemma 2.23 we know that G is biconnected and so by Lemma 2.17 every face of Γ is bounded by a cycle. Suppose that there is a face f in Γ that is bounded by a cycle $v_0, \ldots, v_{k-1}$ of $k \geqslant 4$ vertices. We claim that at least one of the edges $\{v_0, v_2\}$ or $\{v_1, v_3\}$ is not present in G.

Suppose to the contrary that $\{\{v_0, v_2\}, \{v_1, v_3\}\} \subseteq E$. Then we can add a new vertex v' in the interior of f and connect v' inside f to all of v_0, v_1, v_2, v_3 by an edge (curve) without introducing a crossing. In other words, given that G is planar, also the graph $G' = (V \cup \{v'\}, E \cup \{\{v_i, v'\} \mid i \in \{0, 1, 2, 3\}\})$ is planar. However, v_0, v_1, v_2, v_3, v' are branch vertices of a K_5 subdivision in G': v' is connected to all other vertices within f, along the boundary ∂f of f each vertex v_i is connected to both $v_{(i-1) \bmod 4}$ and $v_{(i+1) \bmod 4}$ and the missing two connections are provided by the edges $\{v_0, v_2\}$ and $\{v_1, v_3\}$ (Figure 2.16a). By Kuratowski's Theorem this is in contradiction to G' being planar. Therefore, one of the edges $\{v_0, v_2\}$ or $\{v_1, v_3\}$ is not present in G, as claimed.

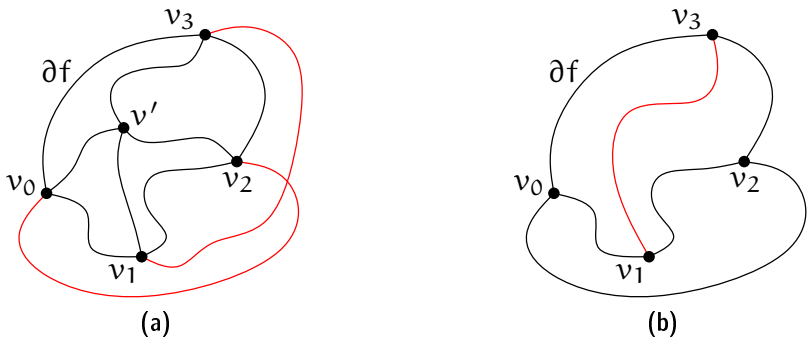


Figure 2.16: *Every face of a maximal planar graph is a topological triangle.*

So suppose without loss of generality that $\{v_1, v_3\} \notin E$. But then we can add this edge (curve) within f to Γ without introducing a crossing (Figure 2.16b). It follows that the edge $\{v_1, v_3\}$ can be added to G without sacrificing planarity, which is in contradiction to G being maximal planar. Therefore, there is no such face f bounded by four or more vertices. □

The proof of Lemma 2.24 also contains the idea for an algorithm to *topologically triangulate* a plane graph.

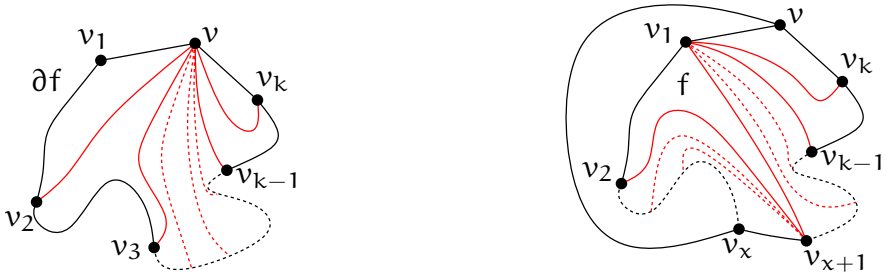
Theorem 2.25 *For a given connected plane graph $G = (V, E)$ on n vertices one can compute in $O(n)$ time and space a maximal plane graph $G' = (V, E')$ with $E \subseteq E'$.*

Proof. Suppose, for instance, that G is represented as a DCEL[2], from which one can easily extract the face boundaries. If some vertex v appears several times along the boundary of a single face, it is a cut-vertex. We fix this by adding an edge between the two neighbors of all but the first occurrence of v. This can easily be done in linear time by maintaining a counter for each vertex on the face boundary. The total number of edges and vertices along the boundary of all faces is proportional to the number of edges in G, which by Corollary 2.5 is linear. Hence we may suppose that all faces of G are bounded by a cycle.

For each face f that is bounded by more than three vertices, select a vertex v_f on its boundary and store with each vertex all faces that select it. Then process each vertex v as follows: First mark all neighbors of v in G. Then process all faces that selected v. For each such face f with $v_f = v$ iterate over the boundary $\partial f = (v, v_1, \ldots, v_k)$, where $k \geqslant 3$, of f to test whether there is any marked vertex other than the two neighbors v_1 and v_k of v along ∂f.

If there is no such vertex, we can safely triangulate f using a star from v, that is, by adding the edges $\{v, v_i\}$, for $i \in \{2, \ldots, k-1\}$ (Figure 2.17a).

Otherwise, let v_x be the first marked vertex in the sequence $v_2, \ldots, v_{k-1}$. The edge $\{v, v_x\}$ that is embedded as a curve in the exterior of f prevents any vertex from $v_1, \ldots, v_{x-1}$ from being connected by an edge in G to any vertex from $v_{x+1}, \ldots, v_k$. (This is exactly the argument that we made in the proof of Lemma 2.24 above for the edges $\{v_0, v_2\}$ and $\{v_1, v_3\}$, see Figure 2.16a.) In particular, we can safely triangulate f using a bi-star from v_1 and v_{x+1}, that is, by adding the edges $\{v_1, v_i\}$, for $i \in \{x+1, \ldots, k\}$, and $\{v_j, v_{x+1}\}$, for $j \in \{2, \ldots, x-1\}$ (Figure 2.17b).



(a) Case 1: v does not have any neighbor on ∂f other than v_1 and v_k.

(b) Case 2: v has a neighbor v_x on ∂f other than v_1 and v_k.

Figure 2.17: *Topologically triangulating a plane graph.*

Finally, conclude the processing of v by removing all marks on its neighbors.

Regarding the runtime bound, note that every face is traversed a constant number of times. In this way, each edge is touched a constant number of times, which by Corollary 2.5 uses linear time overall. Similarly, the vertex marking is done at most twice

[2]If you wonder how the—possibly complicated—curves that correspond to edges are represented: they do not need to be, because here we need a representation of the combinatorial embedding only.

(mark und unmark) per vertex. Therefore, the overall time needed can be bounded by $\sum_{v \in V} \deg_G(v) = 2|E| = O(n)$ by the Handshaking Lemma and Corollary 2.5. □

Theorem 2.26 *A maximal planar graph on $n \geqslant 4$ vertices is 3-connected.*

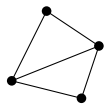
Exercise 2.27 *Prove Theorem 2.26.*

Using any of the standard planarity testing algorithms we can obtain a combinatorial embedding of a planar graph in linear time. Together with Theorem 2.25 this yields the following

Corollary 2.28 *For a given planar graph $G = (V, E)$ on n vertices one can compute in $O(n)$ time and space a maximal planar graph $G' = (V, E')$ with $E \subseteq E'$.* □

The results discussed in this section can serve as a tool to fix the combinatorial embedding for a given graph G: augment G using Theorem 2.25 to a maximal planar graph G′, whose combinatorial embedding is unique by Theorem 2.22.

Being maximal planar is a property of an abstract graph. In contrast, a geometric graph to which no straight-line edge can be added without introducing a crossing is called a *triangulation*. Not every triangulation is maximal planar, as the example depicted to the right shows.



It is also possible to triangulate a geometric graph in linear time. But this problem is much more involved. Triangulating a single face of a geometric graph amounts to what is called "triangulating a simple polygon". This can be done in near-linear[3] time using standard techniques, and in linear time using Chazelle's famous algorithm, whose description spans a fourty pages paper [4].

Exercise 2.29 *We discussed the DCEL structure to represent plane graphs in Section 2.2.1. An alternative way to represent an embedding of a maximal planar graph is the following: For each triangle, store references to its three vertices and to its three neighboring triangles. Compare both approaches. Discuss different scenarios where you would prefer one over the other. In particular, analyze the space requirements of both.*

Connectivity serves as an important indicator for properties of planar graphs. Another example is the following famous theorem of Tutte that provides a sufficient condition for Hamiltonicity. Its proof is beyond the scope of our lecture.

Theorem 2.30 (Tutte [20]) *Every 4-connected planar graph is Hamiltonian.*

[3] $O(n \log n)$ or—using more elaborate tools—$O(n \log^* n)$ time

2.5 Compact straight-line drawings

As a next step we consider plane embeddings in the geometric setting, where every edge is drawn as a straight-line segment. A classical theorem of Wagner and Fáry states that this is not a restriction in terms of plane embeddability.

Theorem 2.31 (Fáry [6], Wagner [21]) *Every planar graph has a plane straight-line embedding (that is, it is isomorphic to a plane straight-line graph).*

Although this theorem has a nice inductive proof, we will not prove it here. Instead we will prove a stronger statement that implies Theorem 2.31.

A very nice property of straight-line embeddings is that they are easy to represent: We need to store points/coordinates for the vertices only. From an algorithmic and complexity point of view the space needed by such a representation is important, because it appears in the input and output size of algorithms that work on embedded graphs. While the Fáry-Wagner Theorem guarantees the existence of a plane straight-line embedding for every planar graph, it does not provide bounds on the size of the coordinates used in the representation. But the following strengthening provides such bounds, by describing an algorithm that embeds (without crossings) a given planar graph on a linear size integer grid.

Theorem 2.32 (de Fraysseix, Pach, Pollack [8]) *Every planar graph on $n \geqslant 3$ vertices has a plane straight-line drawing on the $(2n-3) \times (n-1)$ integer grid.*

Canonical orderings. The key concept behind the algorithm is the notion of a canonical ordering, which is a vertex order that allows to construct a plane drawing in a natural (hence canonical) way. Reading it backwards one may think of a shelling or peeling order that destructs the graph vertex by vertex from the outside. A canonical ordering also provides a succinct representation for the combinatorial embedding.

Definition 2.33 *A plane graph is* **internally triangulated** *if it is biconnected and every bounded face is a (topological) triangle. Let G be an internally triangulated plane graph and $C_\circ(G)$ its outer cycle. A permutation $\pi = (v_1, v_2, \ldots, v_n)$ of $V(G)$ is a* **canonical ordering** *for G, if*

(1) G_k is internally triangulated, for all $k \in \{3, \ldots, n\}$;

(2) v_1v_2 is on the outer cycle $C_\circ(G_k)$ of G_k, for all $k \in \{3, \ldots, n\}$;

(3) v_{k+1} is located in the outer face of G_k and its neighbors appear consecutively along $C_\circ(G_k)$, for all $k \in \{3, \ldots, n-1\}$;

where G_k is the subgraph of G induced by $v_1, \ldots, v_k$.

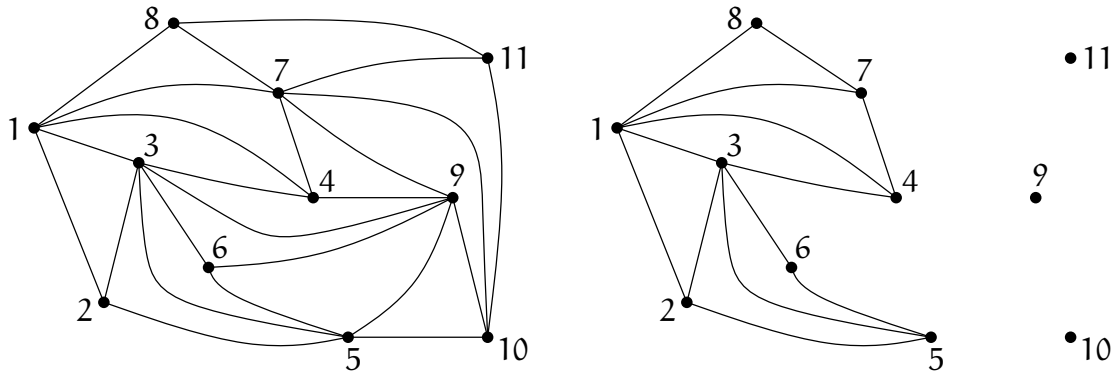
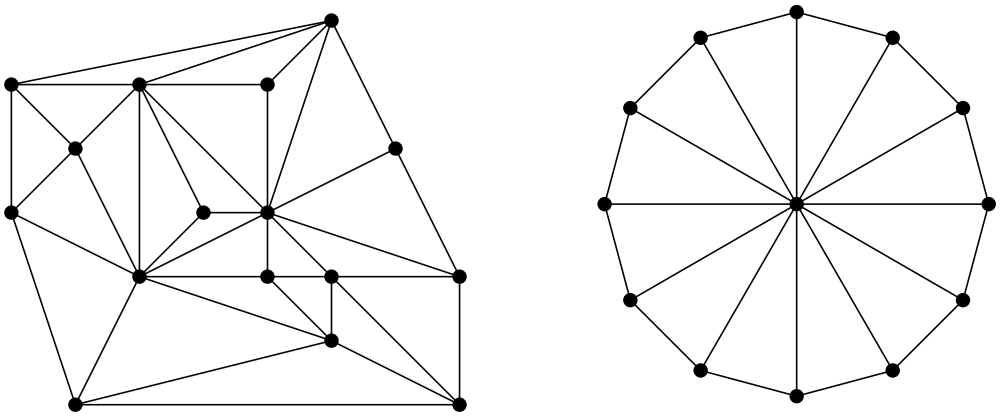


Figure 2.18: *An internally triangulated plane graph and a canonical order for it.*

Figure 2.18 shows an example. Note that there are permutations that do not correspond to a canonical order: for instance, when choosing the vertex 4 as the eighth vertex instead of 8 in Figure 2.18b the graph G_8 is not biconnected (1 is a cut-vertex).

Exercise 2.34 *(a) Compute a canonical ordering for the following internally triangulated plane graphs:*



(b) Give a family of internally triangulated plane graphs G_n *on* $n = 2k$ *vertices with at least* $(n/2)!$ *canonical orderings.*

Exercise 2.35 *(a) Describe a plane graph* G *with* n *vertices that can be embedded (while preserving the outer face) on a grid of size* $(2n/3 - 1) \times (2n/3 - 1)$ *but not on a smaller grid.*

(b) Can you draw G *on a smaller grid if you are allowed to change the embedding?*

Theorem 2.36 *For every internally triangulated plane graph* G *and every edge* $\{v_1, v_2\}$ *on its outer face, there exists a canonical ordering for* G *that starts with* v_1, v_2*. Moreover, such an ordering can be computed in linear time.*

Proof. Induction on n, the number of vertices. For a triangle, any order suffices and so the statement holds. Hence consider an internally triangulated plane graph $G = (V, E)$ on $n \geqslant 4$ vertices. We claim that it is enough to select a vertex $v_n \notin \{v_1, v_2\}$ on $C_\circ(G)$ that is not incident to a chord of $C_\circ(G)$.

First observe that G is plane and $v_n \in C_\circ(G)$ and so all neighbors of v_n in G must appear on the outer face of $G_{n-1} = G \setminus \{v_n\}$. Consider the circular sequence of neighbors around v_n in G and break it into a linear sequence $u_1, \ldots, u_m$, for some $m \geqslant 2$, that starts and ends with the neighbors of v_n in $C_\circ(G)$. As G is internally triangulated, each of the bounded faces spanned by v_n, u_i, u_{i+1}, for $i \in \{i, \ldots, m-1\}$, is a triangle and hence $\{u_i, u_{i+1}\} \in E$. This implies *(3)* for $k = n$. Properties *(1)* and *(2)* hold trivially (by assumption) in that case. In order to complete the ordering inductively we need to show that G_{n-1} is also internally triangulated.

As G_{n-1} is a subgraph of G, which is internally triangulated, it suffices to show that G_{n-1} is biconnected. The outer cycle $C_\circ(G_{n-1})$ of G_{n-1} is obtained from $C_\circ(G)$ by removing v_n and replacing it with the (possibly empty) sequence $u_2, \ldots, u_{m-1}$. As v_n is not incident to a chord of $C_\circ(G)$ (and so neither of $u_2, \ldots, u_{m-1}$ appeared along $C_\circ(G)$ already), the resulting sequence forms a cycle, indeed. Add a new vertex v in the outer face of G_{n-1} and connect v to every vertex of $C_\circ(G_{n-1})$ to obtain a maximal planar graph $H \supset G_{n-1}$. By Theorem 2.26 H is 3-connected and so G_{n-1} is biconnected, as desired. This also completes the proof of the initial claim.

It remains to show that we can always find such a vertex $v_n \notin \{v_1, v_2\}$ on $C_\circ(G)$ that is not incident to a chord of $C_\circ(G)$. If $C_\circ(G)$ does not have any chord, this is obvious, because every cycle has at least three vertices, one of which is neither v_1 nor v_2. So suppose that $C_\circ(G)$ has a chord c. The endpoints of c split $C_\circ(G)$ into two paths, one of which does not have v_1 nor v_2 as an internal vertex. Among all possible chords of $C_\circ(G)$ select c such that this path has minimal length. (It has always at least two edges, because there is always at least one vertex "behind" a chord.) Then by definition of c this path is an induced path in G and none of its (at least one) interior vertices is incident to a chord of $C_\circ(G)$, because such a chord would cross c. So we can select v_n from these vertices. By the way the path is selected with respect to c, this procedure does not select v_1 nor v_2.

Regarding the runtime bound, we maintain the following information for each vertex v: whether it has been chosen already, whether it is on the outer face of the current graph, and the number of incident chords with respect to the current outer cycle. Given a combinatorial embedding of G, it is straighforward to initialize this information in linear time. (Every edge is considered at most twice, once for each endpoint on the outer face.)

When removing a vertex, there are two cases: Either v_n has two neighbors u_1 and u_2 only (Figure 2.19a), in which case the edge u_1u_2 ceases to be a chord. Thus, the chord

count for u_1 and u_2 has to be decremented by one. Otherwise, there are $m \geqslant 3$ neighbors $u_1, \ldots, n_m$ (Figure 2.19b) and (1) all vertices $u_2, \ldots, u_{m-1}$ are new on the outer cycle, and (2) every edge incident to u_i, for $i \in \{2, \ldots, k-1\}$, and some other vertex on the outer cycle other than u_{i-1} or u_{i+1} is a new chord (and the corresponding counters at the endpoints have to by incremented by one).

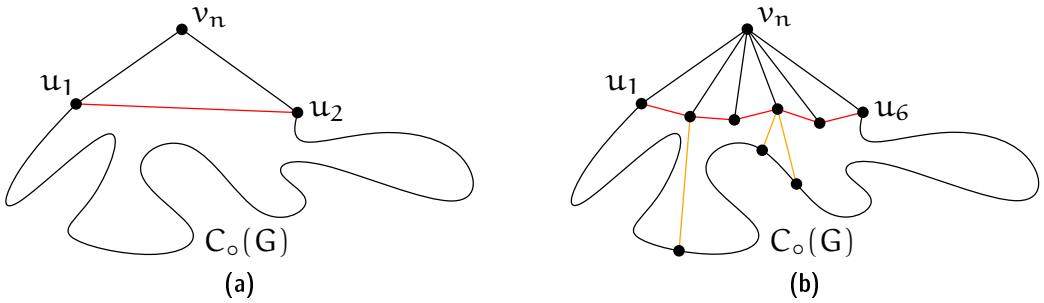


Figure 2.19: *Processing a vertex when computing a canonical ordering.*

During the course of the algorithm every vertex appears once as a new vertex on the outer face. At this point all incident edges are examined. Overall, every edge is inspected at most twice—once for each endpoint—which takes linear time by Corollary 2.5. □

Using one of the linear time planarity testing algorithms, we can obtain a combinatorial embedding for a given maximal planar graph G. As every maximal plane graph is internally triangulated, we can then use Theorem 2.36 to provide us with a canonical ordering for G, in overall linear time.

Corollary 2.37 *Every maximal planar graph admits a canonical ordering. Moreover, such an ordering can be computed in linear time.* □

As simple as they may appear, canonical orderings are a powerful and versatile tool to work with plane graphs. As an example, consider the following partitioning theorem.

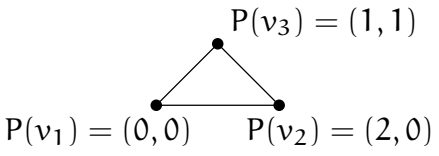
Theorem 2.38 (Schnyder [18]) *For every maximal planar graph* G *on at least three vertices and every face* f *of* G*, the multigraph obtained from* G *by doubling the (three) edges of* f *can be partitioned into three spanning trees.*

Exercise 2.39 *Prove Theorem 2.38.* Hint: *Take a canonical ordering and build one tree by taking for every vertex* v_k *the edge to its first neighbor on the outer cycle* $C_o(G_{k-1})$.

Of a similar flavor is the following open problem, for which only partial answers for specific types of point sets are known [2].

Problem 2.40 (In memoriam Ferran Hurtado (1951–2014))
Can every complete geometric graph on $n = 2k$ vertices (the complete straight line graph on a set of n points in general position) be partitioned into k plane spanning trees?

The shift-algorithm. Let $(v_1, \ldots, v_n)$ be a canonical ordering. The general plan is to construct an embedding by inserting vertices in this order, starting from the triangle $P(v_1) = (0,0)$, $P(v_3) = (1,1)$, $P(v_2) = (2,0)$. At each step, some vertices will be shifted to the right, making room for the edges to the freshly inserted vertex. For each vertex v_i already embedded, maintain a set $L(v_i)$ of vertices that move rigidly together with v_i. Initially, $L(v_i) = \{v_i\}$, for $1 \leqslant i \leqslant 3$.



Ensure that the following invariants hold after Step k (that is, after v_k has been inserted):

(i) $P(v_1) = (0,0)$, $P(v_2) = (2k-4, 0)$;

(ii) The x-coordinates of the points on $C_\circ(G_k) = (w_1, \ldots, w_t)$ are strictly increasing (in this order)[4];

(iii) each edge of $C_\circ(G_k)$ is drawn as a straight-line segment with slope ± 1.

Clearly these invariants hold for G_3, embedded as described above. Invariant (i) implies that after Step n we have $P(v_2) = (2n-4, 0)$, while (iii) implies that the Manhattan distance[5] between any two points on $C_\circ(G_k)$ is even.

Idea: put v_{k+1} at $\mu(w_p, w_q)$, where $w_p, \ldots, w_q$ are its neighbors on $C_\circ(G_k)$ (recall that they appear consecutively along $C_\circ(G_k)$ by definition of a canonical ordering), where

$$\mu((x_p, y_p), (x_q, y_q)) = \frac{1}{2}(x_p - y_p + x_q + y_q, -x_p + y_p + x_q + y_q)$$

is the point of intersection between the line $\ell_1 : y = x - x_p + y_p$ of slope 1 through $w_p = (x_p, y_p)$ and the line $\ell_2 : y = x_q - x + y_q$ of slope -1 through $w_q = (x_q, y_q)$.

Proposition 2.41 *If the Manhattan distance between w_p and w_q is even, then $\mu(w_p, w_q)$ is on the integer grid.*

Proof. By Invariant (ii) we know that $x_p < x_q$. Suppose without loss of generality that $y_p \leqslant y_q$. The Manhattan distance d of w_p and w_q is $x_q - x_p + y_q - y_p$, which by assumption is an even number. Adding the even number $2x_p$ to d yields the even number $x_q + x_p + y_q - y_p$, half of which is the x-coordinate of $\mu((x_p, y_p), (x_q, y_q))$. Adding the

[4] The notation is a bit sloppy here because both t and the w_i in general depend on k. So in principle we should write w_i^k instead of w_i. But as the k would just make a constant appearance throughout, we omit it to avoid index clutter.

[5] The *Manhattan distance* of two points (x_1, y_1) and (x_2, y_2) is $|x_2 - x_1| + |y_2 - y_1|$.

even number $2y_p$ to d yields the even number $x_q - x_p + y_q + y_p$, half of which is the y-coordinate of $\mu((x_p, y_p), (x_q, y_q))$. □

After Step n we have $P(v_n) = (n-2, n-2)$, because v_n is a neighbor of both v_1 and v_2. However, $P(v_{k+1})$ may not "see" all of $w_p, \ldots, w_q$, in case that the slope of $w_p w_{p+1}$ is 1 and/or the slope of $w_{q-1} w_q$ is -1 (Figure 2.20).

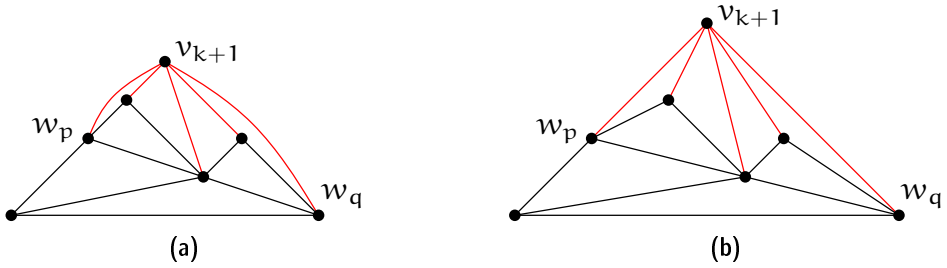


Figure 2.20: *(a) The new vertex v_{k+1} is adjacent to all of $w_p, \ldots, w_q$. If we place v_{k+1} at $\mu(w_p, w_q)$, then some edges may overlap, in case that w_{p+1} lies on the line of slope 1 through w_p or w_{q-1} lies on the line of slope -1 through w_q; (b) shifting $w_{p+1}, \ldots, w_{q-1}$ by one and $w_q, \ldots, w_t$ by two units to the right solves the problem.*

In order to resolve these problems we shift some points around so that after the shift w_{p+1} does not lie on the line of slope 1 through w_p and w_{q-1} does not lie on the line of slope -1 through w_q. The process of inserting v_{k+1} then looks as follows.

1. Shift $\bigcup_{i=p+1}^{q-1} L(w_i)$ to the right by one unit.
2. Shift $\bigcup_{i=q}^{t} L(w_i)$ to the right by two units.
3. $P(v_{k+1}) := \mu(w_p, w_q)$.
4. $L(v_{k+1}) := \{v_k\} \cup \bigcup_{i=p+1}^{q-1} L(w_i)$.

Observe that the Manhattan distance between w_p and w_q remains even, because the shift increases their x-difference by two and leaves the y-coordinates unchanged. Therefore by Proposition 2.41 the vertex v_{k+1} is embedded on the integer grid.

The slopes of the edges $w_p w_{p+1}$ and $w_{q-1} w_q$ (might be just a single edge, in case that $p+1 = q$) become < 1 in absolute value, whereas the slopes of all other edges along the outer cycle remain ± 1. As all edges from v_{k+1} to $w_{p+1}, \ldots, w_{q-1}$ have slope > 1 in absolute value, and the edges $v_{k+1} w_p$ and $v_{k+1} w_q$ have slope ± 1, each edge $v_{k+1} w_i$, for $i \in \{p, \ldots, q\}$ intersects the outer cycle in exactly one point, which is w_i. In other words, adding all edges from v_{k+1} to its neighbors in G_k as straight-line segments results in a plane drawing.

Next we argue that the invariants (i)–(iii) are maintained. For (i) note that we start shifting with w_{p+1} only so that even in case that v_1 is a neighbor of v_{k+1}, it is never

shifted. On the other hand, v_2 is always shifted by two, because we shift every vertex starting from (and including) w_q. Clearly both the shifts and the insertion of v_{k+1} maintain the strict order along the outer cycle, and so (ii) continues to hold. Finally, regarding (iii) note that the edges $w_p w_{p+1}$ and $w_{q-1} w_q$ (possibly this is just a single edge) are the only edges on the outer cycle whose slope is changed by the shift. But these edges do not appear on $C_\circ(G_{k+1})$ anymore. The two edges $v_{k+1} w_p$ and $v_{k+1} w_q$ incident to the new vertex v_{k+1} that appear on $C_\circ(G_{k+1})$ have slope 1 and -1, respectively. So all of (i)–(iii) are invariants of the algorithm, indeed.

So far we have argued about the shift with respect to vertices on the outer cycle of G_k only. To complete the proof of Theorem 2.32 it remains to show that the drawing remains plane under shifts also in its interior part.

Lemma 2.42 *Let G_k, $3 \leqslant k \leqslant n$, be straight-line grid embedded as described, $C_\circ(G_k) = (w_1, \ldots, w_t)$, and let $\delta_1 \leqslant \ldots \leqslant \delta_t$ be non-negative integers. If for each i, we shift $L(w_i)$ by δ_i to the right, then the resulting straight-line drawing is plane.*

Proof. Induction on k. For G_3 this is obvious. Let $v_k = w_\ell$, for some $1 < \ell < t$. Construct a delta sequence Δ for G_{k-1} as follows. If v_k has only two neighbors in G_k, then $C_\circ(G_{k-1}) = (w_1, \ldots, w_{\ell-1}, w_{\ell+1}, \ldots, w_t)$ and we set $\Delta = \delta_1, \ldots, \delta_{\ell-1}, \delta_{\ell+1}, \ldots, \delta_t$. Otherwise, $C_\circ(G_{k-1}) = (w_1, \ldots, w_{\ell-1} = u_1, \ldots, u_m = w_{\ell+1}, \ldots, w_t)$, where $u_1, \ldots, u_m$ are the $m \geqslant 3$ neighbors of v_k in G_k. In this case we set

$$\Delta = \delta_1, \ldots, \delta_{\ell-1}, \underbrace{\delta_\ell, \ldots, \delta_\ell}_{m \text{ times}}, \delta_{\ell+1}, \ldots, \delta_t \,.$$

Clearly, Δ is monotonely increasing and by the inductive assumption a correspondingly shifted drawing of G_{k-1} is plane. When adding v_k and its incident edges back, the drawing remains plane: All vertices $u_2, \ldots, u_{m-1}$ (possibly none) move rigidly with (by exactly the same amount as) v_k by construction. Stretching the edges of the chain $w_{\ell-1}, w_\ell, w_{\ell+1}$ by moving $w_{\ell-1}$ to the left and/or $w_{\ell+1}$ to the right cannot create any crossings. □

Linear time. *(This part was not covered in the lecture.)* The challenge in implementing the shift algorithm efficiently lies in the eponymous shift operations, which modify the x-coordinates of potentially many vertices. In fact, it is not hard to see that a naive implementation—which keeps track of all coordinates explicitly—may use quadratic time. De Fraysseix et al. described an implmentation of the shift algorithm that uses $O(n \log n)$ time. Then Chrobak and Payne [5] observed how to improve the runtime to linear, using the following ideas.

Recall that $P(v_{k+1}) = (x_{k+1}, y_{k+1})$, where

$$\begin{aligned} x_{k+1} &= \frac{1}{2}(x_p - y_p + x_q + y_q) \text{ and} \\ y_{k+1} &= \frac{1}{2}(-x_p + y_p + x_q + y_q) = \frac{1}{2}((x_q - x_p) + y_p + y_q)\,. \end{aligned} \tag{2.43}$$

Thus,

$$x_{k+1} - x_p = \frac{1}{2}((x_q - x_p) + y_q - y_p). \tag{2.44}$$

$\Rightarrow$ We need the y-coordinates of w_p and w_q together with the relative x-position (offset) of w_p and w_q only in to determine the y-coordinate of v_{k+1} and its offset to w_p.

Maintain the outer cycle as a rooted binary tree T, with root v_1. For each node v of T, the *left child* is the first vertex covered by insertion of v (if any), that is, w_{p+1} in the terminology from above (if $p+1 \neq q$), whereas the *right child* of v is the next node along the outer cycle (if any; either along the current outer cycle or along the one at the point where both points were covered together). See Figure 2.21 for an example.

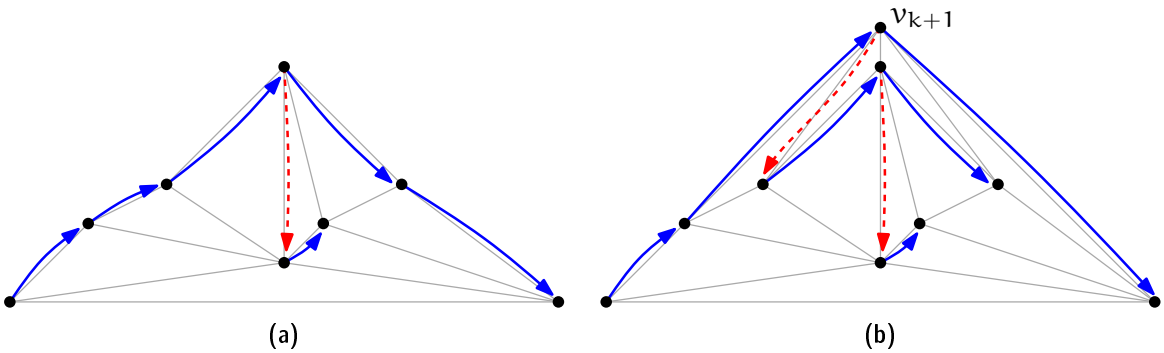


Figure 2.21: *Maintaining the binary tree representation when inserting a new vertex v_{k+1}. Red (dashed) arrows point to left children, blue (solid) arrows point to right children.*

At each node v of T we also store its x-offset $dx(v)$ with respect to the parent node. For the root v_1 of the tree set $dx(v_1) = 0$. In this way, a whole subtree (and, thus, a whole set $L(\cdot)$) can be shifted by changing a single offset entry at its root.

Initially, $dx(v_1) = 0$, $dx(v_2) = dx(v_3) = 1$, $y(v_1) = y(v_2) = 0$, $y(v_3) = 1$, $\text{left}(v_1) = \text{left}(v_2) = \text{left}(v_3) = 0$, $\text{right}(v_1) = v_3$, $\text{right}(v_2) = 0$, and $\text{right}(v_3) = v_2$.

Inserting a vertex v_{k+1} works as follows. As before, let $w_1, \ldots, w_t$ denote the vertices on the outer cycle $C_\circ(G_k)$ and $w_p, \ldots, w_q$ be the neighbors of v_{k+1}.

1. Increment $dx(w_{p+1})$ and $dx(w_q)$ by one. *(This implements the shift.)*
2. Compute $\Delta_{pq} = \sum_{i=p+1}^{q} dx(w_i)$. *(This is the total offset between w_p and w_q.)*
3. Set $dx(v_k) \leftarrow \frac{1}{2}(\Delta_{pq} + y(w_q) - y(w_p))$ and $y(v_k) \leftarrow \frac{1}{2}(\Delta_{pq} + y(w_q) + y(w_p))$. *(This is exactly what we derived in (2.43) and (2.44).)*
4. Set $\text{right}(w_p) \leftarrow v_k$ and $\text{right}(v_k) \leftarrow w_q$. *(Update the current outer cycle.)*
5. If $p+1 = q$, then set $\text{left}(v_k) \leftarrow 0$; else set $\text{left}(v_k) \leftarrow w_{p+1}$ and $\text{right}(w_{q-1}) \leftarrow 0$. *(Update the part that is covered by insertion of v_{k+1}.)*

6. Set $\mathrm{dx}(w_q) \leftarrow \Delta_{pq} - \mathrm{dx}(v_k)$ and—unless $p+1 = q$—set $\mathrm{dx}(w_{p+1}) \leftarrow \mathrm{dx}(w_{p+1}) - \mathrm{dx}(v_k)$. *(Update the offsets according to the changes in the previous two steps.)*

Observe that the only step that possibly cannot be executed in constant time is Step 2. But all vertices but the last vertex w_q for which we sum the offsets are covered by the insertion of v_{k+1}. As every vertex can be covered at most once, the overall complexity of this step during the algorithm is linear. Therefore, this first phase of the algorithm can be completed in linear time.

In a second phase, the final x-coordinates can be computed from the offsets by a single recursive pre-order traversal of the tree. The (pseudo–)code given below is to be called with the root vertex v_1 and an offset of zero. Clearly this yields a linear time algorithm overall.

```
compute_coordinate(Vertex v, Offset d) {
  if (v == 0) return;
  x(v) = dx(v) + d;
  compute_coordinate(left(v), x(v));
  compute_coordinate(right(v), x(v));
}
```

Remarks. From a geometric complexity point of view, Theorem 2.32 provides very good news for planar graphs in a similar way that the Euler Formula does from a combinatorial complexity point of view. Euler's Formular tells us that we can obtain a combinatorial representation (for instance, as a DCEL) of any plane graph using $O(n)$ space, where n is the number of vertices.

Now the shift algorithm tells us that for any planar graph we can even find a geometric plane (straight-line) representation using $O(n)$ space. In addition to the combinatorial information, we only have to store $2n$ numbers from the range $\{0, 1, \ldots, 2n-4\}$.

When we make such claims regarding space complexity we implicitly assume the so-called *word RAM model*. In this model each address in memory contains a *word* of b bits, which means that it can be used to represent any integer from $\{0, \ldots, 2^b - 1\}$. One also assumes that b is sufficiently large, for instance, in our case $b \geqslant \log n$.

There are also different models such as the *bit complexity model*, where one is charged for every bit used to store information. In our case that would already incur an additional factor of $\log n$ for the combinatorial representation: for instance, for each halfedge we store its endpoint, which is an index from $\{1, \ldots, n\}$.

Questions

1. *What is an embedding? What is a planar/plane graph?* Give the definitions and explain the difference between planar and plane.

2. *How many edges can a planar graph have? What is the average vertex degree in a planar graph?* Explain Euler's formula and derive your answers from it.

3. *How can plane graphs be represented on a computer?* Explain the DCEL data structure and how to work with it.

4. *How can a given plane graph be (topologically) triangulated efficiently?* Explain what it is, including the difference between topological and geometric triangulation. Give a linear time algorithm, for instance, as in Theorem 2.25.

5. *What is a combinatorial embedding? When are two combinatorial embeddings equivalent? Which graphs have a unique combinatorial embedding?* Give the definitions, explain and prove Whitney's Theorem.

6. *What is a canonical ordering and which graphs admit such an ordering? For a given graph, how can one find a canonical ordering efficiently?* Give the definition. State and prove Theorem 2.36.

7. *Which graphs admit a plane embedding using straight line edges? Can one bound the size of the coordinates in such a representation?* State and prove Theorem 2.32.

References

[1] Bruce G. Baumgart, A polyhedron representation for computer vision. In *Proc. AFIPS Natl. Comput. Conf.*, vol. 44, pp. 589–596, AFIPS Press, Alrington, Va., 1975, URL http://dx.doi.org/10.1145/1499949.1500071.

[2] Prosenjit Bose, Ferran Hurtado, Eduardo Rivera-Campo, and David R. Wood, Partitions of complete geometric graphs into plane trees. *Comput. Geom. Theory Appl.*, **34**, 2, (2006), 116–125, URL http://dx.doi.org/10.1016/j.comgeo.2005.08.006.

[3] John M. Boyer and Wendy J. Myrvold, On the cutting edge: simplified O(n) planarity by edge addition. *J. Graph Algorithms Appl.*, **8**, 3, (2004), 241–273, URL http://jgaa.info/accepted/2004/BoyerMyrvold2004.8.3.pdf.

[4] Bernard Chazelle, Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, **6**, 5, (1991), 485–524, URL http://dx.doi.org/10.1007/BF02574703.

[5] Marek Chrobak and Thomas H. Payne, A linear-time algorithm for drawing planar graphs. *Inform. Process. Lett.*, **54**, (1995), 241–246, URL http://dx.doi.org/10.1016/0020-0190(95)00020-D.

[6] István Fáry, On straight lines representation of planar graphs. *Acta Sci. Math. Szeged*, **11**, (1948), 229–233.

[7] Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl, Trémaux Trees and Planarity. *Internat. J. Found. Comput. Sci.*, **17**, 5, (2006), 1017—-1030, URL http://dx.doi.org/10.1142/S0129054106004248.

[8] Hubert de Fraysseix, János Pach, and Richard Pollack, How to Draw a Planar Graph on a Grid. *Combinatorica*, **10**, 1, (1990), 41–51, URL http://dx.doi.org/10.1007/BF02122694.

[9] Leonidas J. Guibas and Jorge Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, **4**, 2, (1985), 74–123, URL http://dx.doi.org/10.1145/282918.282923.

[10] John Hopcroft and Robert E. Tarjan, Efficient planarity testing. *J. ACM*, **21**, 4, (1974), 549–568, URL http://dx.doi.org/10.1145/321850.321852.

[11] Ken ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed, The disjoint paths problem in quadratic time. *J. Combin. Theory Ser. B*, **102**, 2, (2012), 424–435, URL http://dx.doi.org/10.1016/j.jctb.2011.07.004.

[12] Lutz Kettner, *Software design in computational geometry and contour-edge based polyhedron visualization*. Ph.D. thesis, ETH Zürich, Zürich, Switzerland, 1999, URL http://dx.doi.org/10.3929/ethz-a-003861002.

[13] Kazimierz Kuratowski, Sur le problème des courbes gauches en topologie. *Fund. Math.*, **15**, 1, (1930), 271–283, URL http://matwbn.icm.edu.pl/ksiazki/fm/fm15/fm15126.pdf.

[14] László Lovász, Graph Minor Theory. *Bull. Amer. Math. Soc.*, **43**, 1, (2005), 75–86, URL http://dx.doi.org/10.1090/S0273-0979-05-01088-8.

[15] Bojan Mohar and Carsten Thomassen, *Graphs on surfaces*. Johns Hopkins University Press, Baltimore, 2001.

[16] David E. Muller and Franco P. Preparata, Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, **7**, (1978), 217–236, URL http://dx.doi.org/10.1016/0304-3975(78)90051-8.

[17] Neil Robertson and Paul Seymour, Graph Minors. XX. Wagner's Conjecture. *J. Combin. Theory Ser. B*, **92**, 2, (2004), 325–357, URL http://dx.doi.org/10.1016/j.jctb.2004.08.001.

[18] Walter Schnyder, Planar Graphs and Poset Dimension. *Order*, **5**, (1989), 323–343, URL http://dx.doi.org/10.1007/BF00353652.

[19] Carsten Thomassen, Kuratowski's Theorem. *J. Graph Theory*, **5**, 3, (1981), 225–241, URL http://dx.doi.org/10.1002/jgt.3190050304.

[20] William T. Tutte, A theorem on planar graphs. *Trans. Amer. Math. Soc.*, **82**, 1, (1956), 99–116, URL http://dx.doi.org/10.1090/S0002-9947-1956-0081471-8.

[21] Klaus Wagner, Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, **46**, (1936), 26–32, URL http://eudml.org/doc/146109.

[22] Klaus Wagner, Über eine Eigenschaft der ebenen Komplexe. *Math. Ann.*, **114**, 1, (1937), 570–590, URL http://dx.doi.org/10.1007/BF01594196.

[23] Kevin Weiler, Edge-based data structures for solid modeling in a curved surface environment. *IEEE Comput. Graph. Appl.*, **5**, 1, (1985), 21–40, URL http://dx.doi.org/10.1109/MCG.1985.276271.

[24] Hassler Whitney, Congruent Graphs and the Connectivity of Graphs. *Amer. J. Math.*, **54**, 1, (1932), 150–168, URL http://www.jstor.org/stable/2371086.

[25] Wikipedia, Planarity testing — Wikipedia, The Free Encyclopedia. 2013, URL http://en.wikipedia.org/wiki/Planarity_testing, [Online; accessed 19-Feb-2013].

Chapter 3

Polygons

Although we can think of a line $\ell \subset \mathbb{R}^2$ as an infinite point set that consists of all points in $\mathbb{R}^2$ that are on ℓ, there still exists a finite description for ℓ. Such a description is, for instance, provided by the three coefficients $a, b, c \in \mathbb{R}$ of an equation of the form $ax + by = c$, with $(a, b) \neq (0, 0)$. Actually this holds true for all of the fundamental geometric objects that were mentioned in the previous section: Each of them has constant *description complexity* (or, informally, just *size*), that is, it can be described by a constant[1] number of parameters.

In this course we will typically deal with objects that are not of constant size. Often these are formed by merely aggregating constant-size objects, for instance, points to form a finite set of points. But sometimes we also demand additional structure that goes beyond aggregation only. Probably the most fundamental geometric objects of this type are what we call *polygons*. You probably learned this term in school, but what *is* a polygon precisely? Consider the examples shown in Figure 3.1. Are all of these polygons? If not, where would you draw the line?

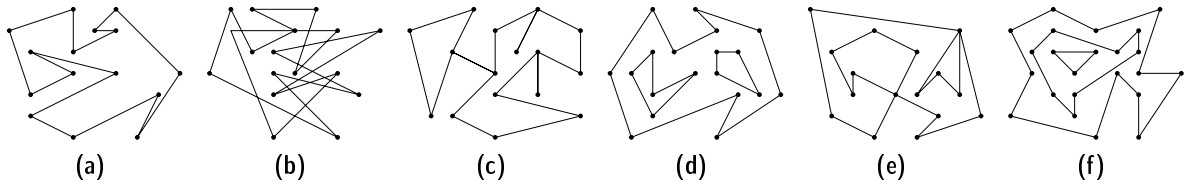


Figure 3.1: *What is a polygon?*

3.1 Classes of Polygons

Obviously, there is not *the* right answer to such a question and certainly there are different types of polygons. Often the term polygon is used somewhat sloppily in place

[1] Unless specified differently, we will always assume that the dimension is (a small) constant. In a high-dimensional space $\mathbb{R}^d$, one has to account for a description complexity of $\Theta(d)$.

of what we call a *simple polygon*, defined below.

Definition 3.1 *A* **simple polygon** *is a compact region $P \subset \mathbb{R}^2$ that is bounded by a simple closed curve $\gamma : [0,1] \to \mathbb{R}^2$ that consists of a finite number of line segments. A* **curve** *is a continuous map $\gamma : [0,1] \to \mathbb{R}^2$. A curve γ is* **closed**, *if $\gamma(0) = \gamma(1)$ and it is* **simple** *if it is injective on $[0,1)$, that is, the curve does not intersect itself.*

Out of the examples shown above only Polygon 3.1a is simple. For each of the remaining polygons it is impossible to combine the bounding segments into a simple closed curve.

The term *compact* for subsets of $\mathbb{R}^d$ means bounded and closed. A subset of $P \subset \mathbb{R}^d$ is *bounded*, if it is contained in the ball of radius r around the origin, for some finite $r > 0$. Being closed means that the boundary is considered to be part of the polygon. In order to formally define these terms, let us briefly review a few basic notions from topology.

The standard topology of $\mathbb{R}^d$ is defined in terms of the Euclidean metric. A point $p \in \mathbb{R}^d$ is *interior* to a set $P \subseteq \mathbb{R}^d$, if there exists an ε-ball $B_\varepsilon(p) = \{x \in \mathbb{R}^d : \|x-p\| < \varepsilon\}$ around p, for some $\varepsilon > 0$, that is completely contained in P. A set is *open*, if all of its points are interior; and it is *closed*, if its complement is open.

Exercise 3.2 *Determine for each of the following sets whether they are open or closed in $\mathbb{R}^2$. a) $B_1(0)$ b) $\{(1,0)\}$ c) $\mathbb{R}^2$ d) $\mathbb{R}^2 \setminus \mathbb{Z}^2$ e) $\mathbb{R}^2 \setminus \mathbb{Q}^2$ f) $\{(x,y) : x \in \mathbb{R}, y \geqslant 0\}$*

Exercise 3.3 *Show that the union of countably many open sets in $\mathbb{R}^d$ is open. Show that the union of a finite number of closed sets in $\mathbb{R}^d$ is closed. (These are two of the axioms that define a topology. So the statements are needed to assert that the metric topology is a topology, indeed.) What follows for intersections of open and closed sets? Finally, show that the union of countably many closed sets in $\mathbb{R}^d$ is not necessarily closed.*

The *boundary* ∂P of a set $P \subset \mathbb{R}^d$ consists of all points that are neither interior to P nor to its complement $\mathbb{R}^d \setminus P$. By definition, for every $p \in \partial P$ every ball $B_\varepsilon(p)$ contains both points from P and from $\mathbb{R}^d \setminus P$. Sometimes one wants to consider a set $P \subset \mathbb{R}^d$ open although it is not. In that case one can resort to the *interior* P° of P that is formed by the subset of points interior to P. Similarly, the *closure* $\overline{P}$ of P is defined by $\overline{P} = P \cup \partial P$.

Lower-dimensional objects, such as line segments in $\mathbb{R}^2$ or triangles in $\mathbb{R}^3$, do not possess any interior point (because the ε-balls needed around any such point are full-dimensional). Whenever we want to talk about the interior of a lower-dimensional object, we use the qualifier *relative* and consider it relative to the smallest affine subspace that contains the object.

For instance, the smallest affine subspace that contains a line segment is a line and so the relative interior of a line segment in $\mathbb{R}^2$ consists of all points except the endpoints, just like for an interval in $\mathbb{R}^1$. Similarly, for a triangle in $\mathbb{R}^3$ the smallest affine subspace that contains it is a plane. Hence its relative interior is just the interior of the triangle, considered as a two-dimensional object.

Exercise 3.4 *Show that for any $P \subset \mathbb{R}^d$ the interior P° is open. (Why is there something to show to begin with?) Show that for any $P \subset \mathbb{R}^d$ the closure $\overline{P}$ is closed.*

When describing a simple polygon P it is sufficient to describe only its boundary ∂P. As ∂P by definition is a simple closed curve γ that consists of finitely many line segments, we can efficiently describe it as a sequence $p_1, \ldots, p_n$ of points, such that γ is formed by the line segments $\overline{p_1p_2}, \overline{p_2p_3}, \ldots, \overline{p_{n-1}p_n}, \overline{p_np_1}$. These points are referred to as the *vertices* of the polygon, and the segments connecting them are referred as the *edges* of the polygon. The *set of vertices* of a polygon P is denoted by $V(P)$, and the *set of edges* of P is denoted by $E(P)$.

Knowing the boundary, it is easy to tell apart the (bounded) interior from the (unbounded) exterior. This is asserted even for much more general curves by the well-known Jordan-Curve Theorem.

Theorem 3.5 (Jordan 1887) *Any simple closed curve $\gamma : [0,1] \to \mathbb{R}^2$ divides the plane into exactly two connected components whose common boundary is formed by γ.*

In full generality, the proof of the deceptively obvious claim is surprisingly difficult. We will not prove it here, the interested reader can find a proof, for instance, in the book of Mohar and Thomassen [11]. There exist different generalizations of the theorem and there also has been some debate about to which degree the original proof of Jordan is actually correct. For simple polygons the situation is easier, though. The essential idea can be worked out algorithmically, which we leave as an exercise.

Exercise 3.6 *Describe an algorithm to decide whether a point lies inside or outside of a simple polygon. More precisely, given a simple polygon $P \subset \mathbb{R}^2$ as a list of its vertices $(v_1, v_2, \ldots, v_n)$ in counterclockwise order and a query point $q \in \mathbb{R}^2$, decide whether q is inside P, on the boundary of P, or outside. The runtime of your algorithm should be $O(n)$.*

There are good reasons to ask for the boundary of a polygon to form a simple curve: For instance, in the example depicted in Figure 3.1b there are several regions for which it is completely unclear whether they should belong to the interior or to the exterior of the polygon. A similar problem arises for the interior regions in Figure 3.1f. But there are more general classes of polygons that some of the remaining examples fall into. We will discuss only one such class here. It comprises polygons like the one from Figure 3.1d.

Definition 3.7 *A region $P \subset \mathbb{R}^2$ is a* **simple polygon with holes** *if it can be described as $P = F \setminus \bigcup_{H \in \mathcal{H}} H^\circ$, where $\mathcal{H}$ is a finite collection of pairwise disjoint simple polygons (called holes) and F is a simple polygon for which $F^\circ \supset \bigcup_{H \in \mathcal{H}} H$.*

The way this definition heavily depends on the notion of simple polygons makes it straightforward to derive a similar trichotomy as the Jordan Curve Theorem provides for simple polygons, that is, every point in the plane is either inside, or on the boundary, or outside of P (exactly one of these three).

3.2 Polygon Triangulation

From a topological point of view, a simple polygon is nothing but a disk and so it is a very elementary object. But geometrically a simple polygon can be—as if mocking the label we attached to it—a pretty complicated shape, see Figure 3.2 for an example. While there is an easy and compact one-dimensional representation in terms of the boundary, as a sequence of vertices/points, it is often desirable to work with a more structured representation of the whole two-dimensional shape.

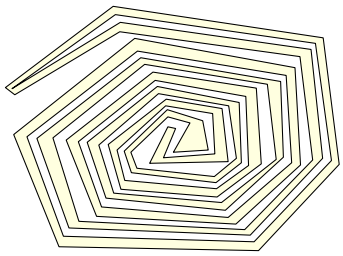


Figure 3.2: *A simple (?) polygon.*

For instance, it is not straightforward to compute the area of a general simple polygon. In order to do so, one usually describes the polygon in terms of simpler geometric objects, for which computing the area is easy. Good candidates for such shapes are triangles, rectangles, and trapezoids. Indeed, it is not hard to show that every simple polygon admits a "nice" partition into triangles, which we call a triangulation.

Definition 3.8 *A* **triangulation** *of a simple polygon* P *is a collection* $\mathcal{T}$ *of triangles, such that*

(1) $P = \bigcup_{T \in \mathcal{T}} T$*;*

(2) $V(P) = \bigcup_{T \in \mathcal{T}} V(T)$*; and*

(3) for every distinct pair $T, U \in \mathcal{T}$*, the intersection* $T \cap U$ *is either a common vertex, or a common edge, or empty.*

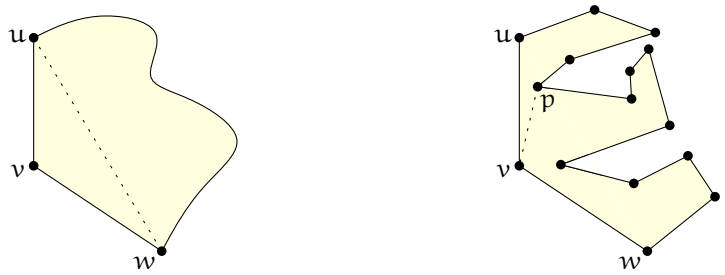
Exercise 3.9 *Show that each condition in Definition 3.8 is necessary in the following sense: Give an example of a non-triangulation that would form a triangulation if the condition was omitted. Is the definition equivalent if (3) is replaced by* $T^\circ \cap U^\circ = \emptyset$*, for every distinct pair* $T, U \in \mathcal{T}$*?*

If we are given a triangulation of a simple polygon P it is easy to compute the area of P by simply summing up the area of all triangles from $\mathcal{T}$. Triangulations are an incredibly useful tool in planar geometry, and one reason for their importance is that every simple polygon admits one.

Theorem 3.10 *Every simple polygon has a triangulation.*

Proof. Let P be a simple polygon on n vertices. We prove the statement by induction on n. For $n = 3$ we face a triangle P that is a triangulation by itself. For $n > 3$ consider the lexicographically smallest vertex v of P, that is, among all vertices of P with a smallest x-coordinate the one with smallest y-coordinate. Denote the neighbors of v (next vertices) along ∂P by u and w. Consider the line segment $\overline{uw}$. We distinguish two cases.

Case 1: except for its endpoints u and w, the segment $\overline{uw}$ lies completely in P°. Then $\overline{uw}$ splits P into two smaller polygons, the triangle uvw and a simple polygon P' on $n-1$ vertices (Figure 3.3a). By the inductive hypothesis, P' has a triangulation that together with T yields a triangulation of P.



(a) Case 1. (b) Case 2.

Figure 3.3: *Cases in the proof of Theorem 3.10.*

Case 2: the relative interior of $\overline{uw}$ does not lie completely in P° (Figure 3.3b). By choice of v, the polygon P is contained in the closed halfplane to the right of the vertical line through v. Therefore, as the segments $\overline{uv}$ and $\overline{vw}$ are part of a simple closed curve defining ∂P, every point sufficiently close to v and between the rays vu and vw must be in P°.

On the other hand, since $\overline{uw} \not\subset P^\circ$, there is some point from ∂P in the interior of the triangle $T = uvw$ (by the choice of v the points u, v, w are not collinear and so T is a triangle, indeed) or on the line segment $\overline{uw}$. In particular, as ∂P is composed of line segments, there is a vertex of P in T° or on $\overline{uw}$ (otherwise, a line segment would have to intersect the line segment $\overline{uw}$ twice, which is impossible). Let p denote a leftmost such vertex. Then the open line segment $\overline{vp}$ is contained in T° and, thus, it splits P into two polygons P_1 and P_2 on less than n vertices each (in one of them, u does not appear as a vertex, whereas w does not appear as a vertex in the other). By the inductive hypothesis, both P_1 and P_2 have triangulations and their union yields a triangulation of P. □

The configuration from Case 1 above is called an *ear*: Three consecutive vertices u, v, w of a simple polygon P such that the relative interior of $\overline{uw}$ lies in P°. In fact, we could have skipped the analysis for Case 2 by referring to the following theorem.

Theorem 3.11 (Meisters [9, 10]) *Every simple polygon that is not a triangle has two non-overlapping ears, that is, two ears* A *and* B *such that* $A^\circ \cap B^\circ = \emptyset$.

But knowing Theorem 3.10 we can obtain Theorem 3.11 as a direct consequence of the following

Theorem 3.12 *Every triangulation of a simple polygon on $n \geqslant 4$ vertices contains at least two (triangles that are) ears.*

Exercise 3.13 *Prove Theorem 3.12.*

Exercise 3.14 *Let P be a simple polygon with vertices $v_1, v_2, \ldots, v_n$ (in counterclockwise order), where v_i has coordinates (x_i, y_i). Show that the area of P is*

$$\frac{1}{2}\sum_{i=1}^{n} x_{i+1}y_i - x_i y_{i+1},$$

where $(x_{n+1}, y_{n+1}) = (x_1, y_1)$.

The number of edges and triangles in a triangulation of a simple polygon are completely determined by the number of vertices, as the following simple lemma shows.

Lemma 3.15 *Every triangulation of a simple polygon on $n \geqslant 3$ vertices consists of $n-2$ triangles and $2n-3$ edges.*

Proof. Proof by induction on n. The statement is true for $n = 3$. For $n > 3$ consider a simple polygon P on n vertices and an arbitrary triangulation T of P. Any edge uv in T that is not an edge of P (and there must be such an edge because P is not a triangle) partitions P into two polygons P_1 and P_2 with n_1 and n_2 vertices, respectively. Since $n_1, n_2 < n$ we conclude by the inductive hypothesis that T partitions P_1 into $n_1 - 2$ triangles and P_2 into $n_2 - 2$ triangles, using $2n_1 - 3$ and $2n_2 - 3$ edges, respectively.

All vertices of P appear in exactly one of P_1 or P_2, except for u and v, which appear in both. Therefore $n_1+n_2 = n+2$ and so the number of triangles in T is $(n_1-2)+(n_2-2) = (n_1+n_2)-4 = n+2-4 = n-2$. Similarly, all edges of T appear in exactly one of P_1 or P_2, except for the edge uv, which appears in both. Therefore the number of edges in T is $(2n_1-3)+(2n_2-3)-1 = 2(n_1+n_2)-7 = 2(n+2)-7 = 2n-3$. □

The universal presence of triangulations is something particular about the plane: The natural generalization of Theorem 3.10 to dimension three and higher does not hold. What is this generalization, anyway?

Tetrahedralizations in $\mathbb{R}^3$. A simple polygon is a planar object that is a topological disk that is locally bounded by patches of lines. The corresponding term in $\mathbb{R}^3$ is a *polyhedron*, and although we will not formally define it here yet, a literal translation of the previous sentence yields an object that topologically is a ball and is locally bounded by patches of planes. A triangle in $\mathbb{R}^2$ corresponds to a tetrahedron in $\mathbb{R}^3$ and a *tetrahedralization* is a nice partition into tetrahedra, where "nice" means that the union of the tetrahedra covers the object, the vertices of the tetrahedra are vertices of the polyhedron, and any

two distinct tetrahedra intersect in either a common triangular face, or a common edge, or a common vertex, or not at all.[2]

Unfortunately, there are polyhedra in $\mathbb{R}^3$ that do not admit a tetrahedralization. The following construction is due to Schönhardt [12]. It is based on a triangular prism, that is, two congruent triangles placed in parallel planes where the corresponding sides of both triangles are connected by a rectangle (Figure 3.4a). Then one triangle is twisted/rotated slightly within its plane. As a consequence, the rectangular faces are not plane anymore, but they obtain an inward dent along their diagonal in direction of the rotation (Figure 3.4b). The other (former) diagonals of the rectangular faces—labeled ab', bc', and

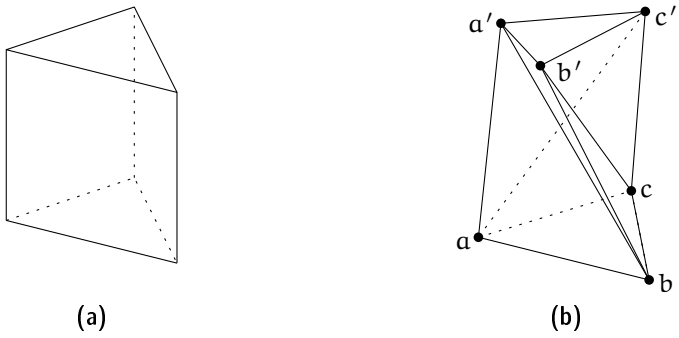


Figure 3.4: *The Schönhardt polyhedron cannot be subdivided into tetrahedra without adding new vertices.*

ca' in Figure 3.4b—are now epigonals, that is, they lie in the exterior of the polyhedron. Since these epigonals are the only edges between vertices that are not part of the polyhedron, there is no way to add edges to form a tetrahedron for a subdivision. Clearly the polyhedron is not a tetrahedron by itself, and so we conclude that it does not admit a subdivision into tetrahedra without adding new vertices. If adding new vertices—so-called Steiner vertices—is allowed, then there is no problem to construct a tetrahedralization, and this holds true in general.

Algorithms. Knowing that a triangulation exists is nice, but it is much better to know that it can also be constructed efficiently.

Exercise 3.16 *Convert Theorem 3.10 into an $O(n^2)$ time algorithm to construct a triangulation for a given simple polygon on n vertices.*

The runtime achieved by the straightforward application of Theorem 3.10 is not optimal. We will revisit this question at several times during this course and discuss improved algorithms for the problem of triangulating a simple polygon.

[2]These "nice" subdivisions can be defined in an abstract combinatorial setting, where they are called *simplicial complices*.

The best (in terms of worst-case runtime) algorithm known due to Chazelle [4] computes a triangulation in linear time. But this algorithm is very complicated and we will not discuss it here. There is also a somewhat simpler randomized algorithm to compute a triangulation in expected linear time [2], which we will not discuss in detail, either. Instead you will later see a much simpler algorithm with a pretty-close-to linear runtime bound. The question of whether there exists a simple (which is not really a well-defined term, of course, except that Chazelle's Algorithm does not qualify) deterministic linear time algorithm to triangulate a simple polygon remains open [7].

Polygons with holes. It is interesting to note that the complexity of the problem changes to $\Theta(n \log n)$, if the polygon may contain holes [3]. This means that there is an algorithm to construct a triangulation for a given simple polygon with holes on a total of n vertices (counting both the vertices on the outer boundary and those of holes) in $O(n \log n)$ time. But there is also a lower bound of $\Omega(n \log n)$ operations that holds in all models of computation in which there exists the corresponding lower bound for comparison-based sorting. This difference in complexity is a very common pattern: There are many problems that are (sometimes much) harder for simple polygons with holes than for simple polygons. So maybe the term "simple" has some justification, after all...

Genaral triangle covers. What if we drop the "niceness" conditions required for triangulations and just want to describe a given simple polygon as a union of triangles? It turns out this is a rather drastic change and, for instance, it is unlikely that we can efficiently find an optimal/minimal description of this type: Christ has shown [5] that it is NP-hard to decide whether for a simple polygon P on n vertices and a positive integer k, there exists a set of at most k triangles whose union is P. In fact, the problem is not even known to be in NP, because it is not clear whether the coordinates of solutions can always be encoded compactly.

3.3 The Art Gallery Problem

In 1973 Victor Klee posed the following question: "How many guards are necessary, and how many are sufficient to patrol the paintings and works of art in an art gallery with n walls?" From a geometric point of view, we may think of an "art gallery with n walls" as a simple polygon bounded by n edges, that is, a simple polygon P with n vertices. And a guard can be modeled as a point where we imagine the guard to stand and observe everything that is in sight. In sight, finally, refers to the walls of the gallery (edges of the polygon) that are opaque and, thus, prevent a guard to see what is behind. In other words, a guard (point) g can watch over every point $p \in P$, for which the line segment $\overline{gp}$ lies completely in P°, see Figure 3.5.

It is not hard to see that $\lfloor n/3 \rfloor$ guards are necessary in general.

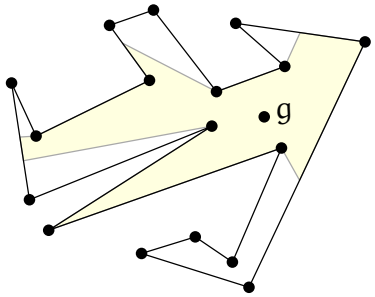


Figure 3.5: *The region that a guard* g *can observe.*

Exercise 3.17 *Describe a family* $(P_n)_{n\geqslant 3}$ *of simple polygons such that* P_n *has* n *vertices and at least* $\lfloor n/3 \rfloor$ *guards are needed to guard it.*

What is more surprising: $\lfloor n/3 \rfloor$ guards are always sufficient as well. Chvátal [6] was the first to prove that, but then Fisk [8] gave a much simpler proof using—you may have guessed it—triangulations. Fisk's proof was considered so beautiful that it was included into "Proofs from THE BOOK" [1], a collection inspired by Paul Erdős' belief in "a place where God keeps aesthetically perfect proofs". The proof is based on the following lemma.

Lemma 3.18 *Every triangulation of a simple polygon is 3-colorable. That is, each vertex can be assigned one of three colors in such a way that adjacent vertices receive different colors.*

Proof. Induction on n. For $n = 3$ the statement is obvious. For $n > 3$, by Theorem 3.12 the triangulation contains an ear uvw. Cutting off the ear creates a triangulation of a polygon on $n - 1$ vertices, which by the inductive hypothesis admits a 3-coloring. Now whichever two colors the vertices u and w receive in this coloring, there remains a third color to be used for v. □

Theorem 3.19 (Fisk [8]) *Every simple polygon on* n *vertices can be guarded using at most* $\lfloor n/3 \rfloor$ *guards.*

Proof. Consider a triangulation of the polygon and a 3-coloring of the vertices as ensured by Lemma 3.18. Take the smallest color class, which clearly consists of at most $\lfloor n/3 \rfloor$ vertices, and put a guard at each vertex. As every point of the polygon is contained in at least one triangle and every triangle has exactly one vertex in the guarding set, the whole polygon is guarded. □

Questions

8. *What is a simple polygon/a simple polygon with holes* Explain the definitions and provide some examples of members and non-members of the respective classes.

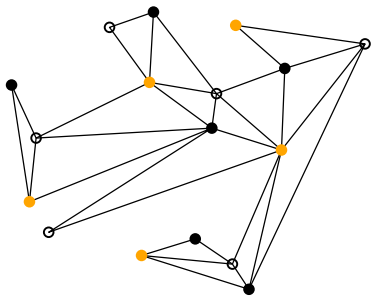


Figure 3.6: *A triangulation of a simple polygon on* 17 *vertices and a 3-coloring of it. The vertices shown solid orange form the smallest color class and guard the polygon using* $\lfloor 17/3 \rfloor = 5$ *guards.*

For a given polygon you should be able to tell which of these classes it belongs to or does not belong to and argue why this is the case.

9. *What is a closed/open/bounded set in* $\mathbb{R}^d$*? What is the interior/closure of a point set?* Explain the definitions and provide some illustrative examples. For a given set you should be able to argue which of the properties mentioned it possesses.

10. *What is a triangulation of a simple polygon? Does it always exist?* Explain the definition and provide some illustrative examples. Present the proof of Theorem 3.10 in detail.

11. *How about higher dimensional generalizations? Can every polyhedron in* $\mathbb{R}^3$ *be nicely subdivided into tetrahedra?* Explain Schönhardt's construction.

12. *How many points are needed to guard a simple polygon?* Present the proofs of Theorem 3.12, Lemma 3.18, and Theorem 3.19 in detail.

References

[1] Martin Aigner and Günter M. Ziegler, *Proofs from THE BOOK*. Springer-Verlag, Berlin, 3rd edn., 2003.

[2] Nancy M. Amato, Michael T. Goodrich, and Edgar A. Ramos, A randomized algorithm for triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, **26**, 2, (2001), 245–265, URL `http://dx.doi.org/10.1007/s00454-001-0027-x`.

[3] Takao Asano, Tetsuo Asano, and Ron Y. Pinter, Polygon triangulation: efficiency and minimality. *J. Algorithms*, **7**, 2, (1986), 221–231, URL `http://dx.doi.org/10.1016/0196-6774(86)90005-2`.

[4] Bernard Chazelle, Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, **6**, 5, (1991), 485–524, URL `http://dx.doi.org/10.1007/BF02574703`.

[5] Tobias Christ, Beyond triangulation: covering polygons with triangles. In *Proc. 12th Algorithms and Data Struct. Sympos.*, vol. 6844 of *Lecture Notes Comput. Sci.*, pp. 231–242, Springer-Verlag, 2011, URL http://dx.doi.org/10.1007/978-3-642-22300-6_20.

[6] Václav Chvátal, A combinatorial theorem in plane geometry. *J. Combin. Theory Ser. B*, **18**, 1, (1975), 39–41, URL http://dx.doi.org/10.1016/0095-8956(75)90061-1.

[7] Erik D. Demaine, Joseph S. B. Mitchell, and Joseph O'Rourke, The Open Problems Project, Problem #10. http://cs.smith.edu/~orourke/TOPP/P10.html.

[8] Steve Fisk, A short proof of Chvátal's watchman theorem. *J. Combin. Theory Ser. B*, **24**, 3, (1978), 374, URL http://dx.doi.org/10.1016/0095-8956(78)90059-X.

[9] Gary H. Meisters, Polygons have ears. *Amer. Math. Monthly*, **82**, 6, (1975), 648–651, URL http://www.jstor.org/stable/2319703.

[10] Gary H. Meisters, Principal vertices, exposed points, and ears. *Amer. Math. Monthly*, **87**, 4, (1980), 284–285, URL http://www.jstor.org/stable/2321563.

[11] Bojan Mohar and Carsten Thomassen, *Graphs on surfaces*. Johns Hopkins University Press, Baltimore, 2001.

[12] Erich Schönhardt, Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Math. Ann.*, **98**, (1928), 309–312, URL http://dx.doi.org/10.1007/BF01451597.

Chapter 4

Convex Hull

There exists an incredible variety of point sets and polygons. Among them, some have certain properties that make them "nicer" than others in some respect. For instance, look at the two polygons shown below.

(a) A convex polygon. (b) A non-convex polygon.

Figure 4.1: *Examples of polygons: Which do you like better?*

As it is hard to argue about aesthetics, let us take a more algorithmic stance. When designing algorithms, the polygon shown on the left appears much easier to deal with than the visually and geometrically more complex polygon shown on the right. One particular property that makes the left polygon nice is that one can walk between any two vertices along a straight line without ever leaving the polygon. In fact, this statement holds true not only for vertices but for any two points within the polygon. A polygon or, more generally, a set with this property is called *convex*.

Definition 4.1 *A set* $P \subseteq \mathbb{R}^d$ *is* **convex** *if* $\overline{pq} \subseteq P$*, for any* $p, q \in P$.

An alternative, equivalent way to phrase convexity would be to demand that for every line $\ell \subset \mathbb{R}^d$ the intersection $\ell \cap P$ be connected. The polygon shown in Figure 4.1b is not convex because there are some pairs of points for which the connecting line segment is not completely contained within the polygon. An immediate consequence of the definition is the following

Observation 4.2 *For any family* $(P_i)_{i\in I}$ *of convex sets, the intersection* $\bigcap_{i\in I} P_i$ *is convex.*

Indeed there are many problems that are comparatively easy to solve for convex sets but very hard in general. We will encounter some particular instances of this phenomenon later in the course. However, not all polygons are convex and a discrete set of points is never convex, unless it consists of at most one point only. In such a case it is useful to make a given set P convex, that is, approximate P with or, rather, encompass P within a convex set $H \supseteq P$. Ideally, H differs from P as little as possible, that is, we want H to be a smallest convex set enclosing P.

At this point let us step back for a second and ask ourselves whether this wish makes sense at all: Does such a set H (always) exist? Fortunately, we are on the safe side because the whole space $\mathbb{R}^d$ is certainly convex. It is less obvious, but we will see below that H is actually unique. Therefore it is legitimate to refer to H as **the** smallest convex set enclosing P or—shortly—the *convex hull* of P.

4.1 Convexity

In this section we will derive an algebraic characterization of convexity. Such a characterization allows to investigate convexity using the machinery from linear algebra.

Consider $P \subset \mathbb{R}^d$. From linear algebra courses you should know that the *linear hull*

$$\operatorname{lin}(P) := \left\{ q \;\middle|\; q = \sum \lambda_i p_i \wedge \forall i : p_i \in P, \lambda_i \in \mathbb{R} \right\}$$

is the set of all *linear combinations* of P (smallest linear subspace containing P). For instance, if $P = \{p\} \subset \mathbb{R}^2 \setminus \{0\}$ then $\operatorname{lin}(P)$ is the line through p and the origin.

Similarly, the *affine hull*

$$\operatorname{aff}(P) := \left\{ q \;\middle|\; q = \sum \lambda_i p_i \wedge \sum \lambda_i = 1 \wedge \forall i : p_i \in P, \lambda_i \in \mathbb{R} \right\}$$

is the set of all *affine combinations* of P (smallest affine subspace containing P). For instance, if $P = \{p, q\} \subset \mathbb{R}^2$ and $p \neq q$ then $\operatorname{aff}(P)$ is the line through p and q.

It turns out that convexity can be described in a very similar way algebraically, which leads to the notion of *convex combinations*.

Proposition 4.3 *A set* $P \subseteq \mathbb{R}^d$ *is convex if and only if* $\sum_{i=1}^n \lambda_i p_i \in P$, *for all* $n \in \mathbb{N}$, $p_1, \ldots, p_n \in P$, *and* $\lambda_1, \ldots, \lambda_n \geqslant 0$ *with* $\sum_{i=1}^n \lambda_i = 1$.

Proof. "$\Leftarrow$": obvious with $n = 2$.

"$\Rightarrow$": Induction on n. For $n = 1$ the statement is trivial. For $n \geqslant 2$, let $p_i \in P$ and $\lambda_i \geqslant 0$, for $1 \leqslant i \leqslant n$, and assume $\sum_{i=1}^n \lambda_i = 1$. We may suppose that $\lambda_i > 0$, for all i. (Simply omit those points whose coefficient is zero.) We need to show that $\sum_{i=1}^n \lambda_i p_i \in P$.

Define $\lambda = \sum_{i=1}^{n-1} \lambda_i$ and for $1 \leqslant i \leqslant n-1$ set $\mu_i = \lambda_i/\lambda$. Observe that $\mu_i \geqslant 0$ and $\sum_{i=1}^{n-1} \mu_i = 1$. By the inductive hypothesis, $q := \sum_{i=1}^{n-1} \mu_i p_i \in P$, and thus by convexity of P also $\lambda q + (1-\lambda)p_n \in P$. We conclude by noting that $\lambda q + (1-\lambda)p_n = \lambda \sum_{i=1}^{n-1} \mu_i p_i + \lambda_n p_n = \sum_{i=1}^{n} \lambda_i p_i$. □

Definition 4.4 *The* **convex hull** $\mathrm{conv}(P)$ *of a set* $P \subseteq \mathbb{R}^d$ *is the intersection of all convex supersets of* P.

At first glance this definition is a bit scary: There may be a whole lot of supersets for any given P and it not clear that taking the intersection of all of them yields something sensible to work with. However, by Observation 4.2 we know that the resulting set is convex, at least. The missing bit is provided by the following proposition, which characterizes the convex hull in terms of exactly those convex combinations that appeared in Proposition 4.3 already.

Proposition 4.5 *For any* $P \subseteq \mathbb{R}^d$ *we have*

$$\mathrm{conv}(P) = \left\{ \sum_{i=1}^{n} \lambda_i p_i \;\middle|\; n \in \mathbb{N} \wedge \sum_{i=1}^{n} \lambda_i = 1 \wedge \forall i \in \{1,\dots,n\} : \lambda_i \geqslant 0 \wedge p_i \in P \right\}.$$

The elements of the set on the right hand side are referred to as *convex combinations* of P.

Proof. "$\supseteq$": Consider a convex set $C \supseteq P$. By Proposition 4.3 (only-if direction) the right hand side is contained in C. As C was arbitrary, the claim follows.

"$\subseteq$": Denote the set on the right hand side by R. Clearly $R \supseteq P$. We show that R forms a convex set. Let $p = \sum_{i=1}^{n} \lambda_i p_i$ and $q = \sum_{i=1}^{n} \mu_i p_i$ be two convex combinations. (We may suppose that both p and q are expressed over the same p_i by possibly adding some terms with a coefficient of zero.)

Then for $\lambda \in [0,1]$ we have $\lambda p + (1-\lambda)q = \sum_{i=1}^{n} (\lambda\lambda_i + (1-\lambda)\mu_i)p_i \in R$, as $\underbrace{\lambda\lambda_i}_{\geqslant 0} + \underbrace{(1-\lambda)}_{\geqslant 0}\underbrace{\mu_i}_{\geqslant 0} \geqslant 0$, for all $1 \leqslant i \leqslant n$, and $\sum_{i=1}^{n} (\lambda\lambda_i + (1-\lambda)\mu_i) = \lambda + (1-\lambda) = 1$. □

In linear algebra the notion of a basis in a vector space plays a fundamental role. In a similar way we want to describe convex sets using as few entities as possible, which leads to the notion of extremal points, as defined below.

Definition 4.6 *The convex hull of a finite point set* $P \subset \mathbb{R}^d$ *forms a* **convex polytope**. *Each* $p \in P$ *for which* $p \notin \mathrm{conv}(P \setminus \{p\})$ *is called a* **vertex** *of* $\mathrm{conv}(P)$. *A vertex of* $\mathrm{conv}(P)$ *is also called an* **extremal point** *of* P. *A convex polytope in* $\mathbb{R}^2$ *is called a* **convex polygon**.

Essentially, the following proposition shows that the term vertex above is well defined.

Proposition 4.7 *A convex polytope in* $\mathbb{R}^d$ *is the convex hull of its vertices.*

Proof. Let $P = \{p_1, \ldots, p_n\}$, $n \in \mathbb{N}$, such that without loss of generality $p_1, \ldots, p_k$ are the vertices of $\mathcal{P} := \text{conv}(P)$. We prove by induction on n that $\text{conv}(p_1, \ldots, p_n) \subseteq \text{conv}(p_1, \ldots, p_k)$. For $n = k$ the statement is trivial.

For $n > k$, p_n is not a vertex of $\mathcal{P}$ and hence p_n can be expressed as a convex combination $p_n = \sum_{i=1}^{n-1} \lambda_i p_i$. Thus for any $x \in \mathcal{P}$ we can write $x = \sum_{i=1}^{n} \mu_i p_i = \sum_{i=1}^{n-1} \mu_i p_i + \mu_n \sum_{i=1}^{n-1} \lambda_i p_i = \sum_{i=1}^{n-1} (\mu_i + \mu_n \lambda_i) p_i$. As $\sum_{i=1}^{n-1} (\mu_i + \mu_n \lambda_i) = 1$, we conclude inductively that $x \in \text{conv}(p_1, \ldots, p_{n-1}) \subseteq \text{conv}(p_1, \ldots, p_k)$. □

4.2 Classical Theorems for Convex Sets

Next we will discuss a few fundamental theorems about convex sets in $\mathbb{R}^d$. The proofs typically use the algebraic characterization of convexity and then employ some techniques from linear algebra.

Theorem 4.8 (Radon [8]) *Any set $P \subset \mathbb{R}^d$ of $d+2$ points can be partitioned into two disjoint subsets P_1 and P_2 such that $\text{conv}(P_1) \cap \text{conv}(P_2) \neq \emptyset$.*

Proof. Let $P = \{p_1, \ldots, p_{d+2}\}$. No more than $d+1$ points can be affinely independent in $\mathbb{R}^d$. Hence suppose without loss of generality that p_{d+2} can be expressed as an affine combination of $p_1, \ldots, p_{d+1}$, that is, there exist $\lambda_1, \ldots, \lambda_{d+1} \in \mathbb{R}$ with $\sum_{i=1}^{d+1} \lambda_i = 1$ and $\sum_{i=1}^{d+1} \lambda_i p_i = p_{d+2}$. Let P_1 be the set of all points p_i for which λ_i is positive and let $P_2 = P \setminus P_1$. Then setting $\lambda_{d+2} = -1$ we can write $\sum_{p_i \in P_1} \lambda_i p_i = \sum_{p_i \in P_2} -\lambda_i p_i$, where all coefficients on both sides are non-negative. Renormalizing by $\mu_i = \lambda_i / \mu$ and $\nu_i = \lambda_i / \nu$, where $\mu = \sum_{p_i \in P_1} \lambda_i$ and $\nu = -\sum_{p_i \in P_2} \lambda_i$, yields convex combinations $\sum_{p_i \in P_1} \mu_i p_i = \sum_{p_i \in P_2} \nu_i p_i$ that describe a common point of $\text{conv}(P_1)$ and $\text{conv}(P_2)$. □

Theorem 4.9 (Helly) *Consider a collection $\mathcal{C} = \{C_1, \ldots, C_n\}$ of $n \geqslant d+1$ convex subsets of $\mathbb{R}^d$, such that any $d+1$ pairwise distinct sets from $\mathcal{C}$ have non-empty intersection. Then also the intersection $\bigcap_{i=1}^{n} C_i$ of all sets from $\mathcal{C}$ is non-empty.*

Proof. Induction on n. The base case $n = d+1$ holds by assumption. Hence suppose that $n \geqslant d+2$. Consider the sets $D_i = \bigcap_{j \neq i} C_j$, for $i \in \{1, \ldots, n\}$. As D_i is an intersection of $n-1$ sets from $\mathcal{C}$, by the inductive hypothesis we know that $D_i \neq \emptyset$. Therefore we can find some point $p_i \in D_i$, for each $i \in \{1, \ldots, n\}$. Now by Theorem 4.8 the set $P = \{p_1, \ldots, p_n\}$ can be partitioned into two disjoint subsets P_1 and P_2 such that $\text{conv}(P_1) \cap \text{conv}(P_2) \neq \emptyset$. We claim that any point $p \in \text{conv}(P_1) \cap \text{conv}(P_2)$ also lies in $\bigcap_{i=1}^{n} C_i$, which completes the proof.

Consider some C_i, for $i \in \{1, \ldots, n\}$. By construction $D_j \subseteq C_i$, for $j \neq i$. Thus p_i is the only point from P that may not be in C_i. As p_i is part of only one of P_1 or P_2, say, of P_1, we have $P_2 \subseteq C_i$. The convexity of C_i implies $\text{conv}(P_2) \subseteq C_i$ and, therefore, $p \in C_i$. □

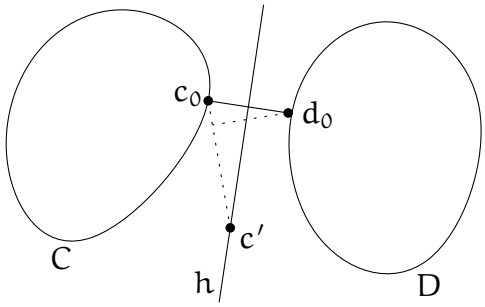
Theorem 4.10 (Carathéodory [3]) *For any* $P \subset \mathbb{R}^d$ *and* $q \in \mathrm{conv}(P)$ *there exist* $k \leqslant d+1$ *points* $p_1, \ldots, p_k \in P$ *such that* $q \in \mathrm{conv}(p_1, \ldots, p_k)$.

Exercise 4.11 *Prove Theorem 4.10.*

Theorem 4.12 (Separation Theorem) *Any two compact convex sets* $C, D \subset \mathbb{R}^d$ *with* $C \cap D = \emptyset$ *can be separated strictly by a hyperplane, that is, there exists a hyperplane* h *such that* C *and* D *lie in the opposite open halfspaces bounded by* h.

Proof. Consider the distance function $\delta : C \times D \to \mathbb{R}$ with $(c, d) \mapsto \|c - d\|$. Since $C \times D$ is compact and δ is continuous and strictly bounded from below by 0, the function δ attains its minimum at some point $(c_0, d_0) \in C \times D$ with $\delta(c_0, d_0) > 0$. Let h be the hyperplane perpendicular to the line segment $\overline{c_0 d_0}$ and passing through the midpoint of c_0 and d_0.

If there was a point, say, c' in $C \cap h$, then by convexity of C the whole line segment $\overline{c_o c'}$ lies in C and some point along this segment is closer to d_0 than is c_0, in contradiction to the choice of c_0. The figure shown to the right depicts the situation in $\mathbb{R}^2$. If, say, C has points on both sides of h, then by convexity of C it has also a point on h, but we just saw that there is no such point. Therefore, C and D must lie in different open halfspaces bounded by h. □



The statement above is wrong for arbitrary (not necessarily compact) convex sets. However, if the separation is not required to be strict (the hyperplane may intersect the sets), then such a separation always exists, with the proof being a bit more involved (cf. [7], but also check the errata on Matoušek's webpage).

Exercise 4.13 *Show that the Separation Theorem does not hold in general, if not both of the sets are convex.*

Exercise 4.14 *Prove or disprove:*

(a) The convex hull of a compact subset of $\mathbb{R}^d$ *is compact.*

(b) The convex hull of a closed subset of $\mathbb{R}^d$ *is closed.*

Altogether we obtain various equivalent definitions for the convex hull, summarized in the following theorem.

Theorem 4.15 *For a compact set* $P \subset \mathbb{R}^d$ *we can characterize* $\mathrm{conv}(P)$ *equivalently as one of*

(a) the smallest (w. r. t. set inclusion) convex subset of $\mathbb{R}^d$ *that contains* P*;*

(b) the set of all convex combinations of points from P*;*

(c) the set of all convex combinations formed by $d+1$ *or fewer points from* P*;*

(d) the intersection of all convex supersets of P*;*

(e) the intersection of all closed halfspaces containing P*.*

Exercise 4.16 *Prove Theorem 4.15.*

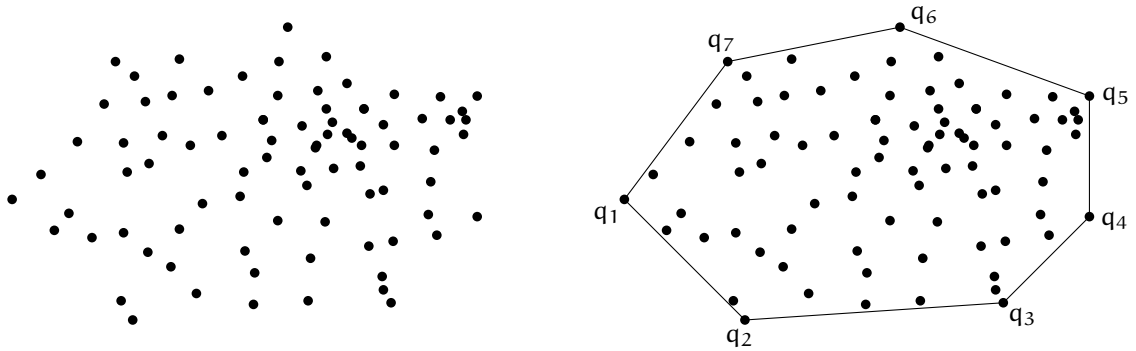
4.3 Planar Convex Hull

Although we know by now what is the convex hull of point set, it is not yet clear how to construct it algorithmically. As a first step, we have to find a suitable representation for convex hulls. In this section we focus on the problem in $\mathbb{R}^2$, where the convex hull of a finite point set forms a convex polygon. A convex polygon is easy to represent, for instance, as a sequence of its vertices in counterclockwise orientation. In higher dimensions finding a suitable representation for convex polytopes is a much more delicate task.

Problem 4.17 (Convex hull)

Input: $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, $n \in \mathbb{N}$.

Output: Sequence $(q_1, \ldots, q_h)$, $1 \leqslant h \leqslant n$, of the vertices of $\mathrm{conv}(P)$ (ordered counterclockwise).



(a) Input. (b) Output.

Figure 4.2: *Convex Hull of a set of points in* $\mathbb{R}^2$*.*

Another possible algorithmic formulation of the problem is to ignore the structure of the convex hull and just consider it as a point set.

Problem 4.18 (Extremal points)

Input: $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, $n \in \mathbb{N}$.

Output: Set $Q \subseteq P$ of the vertices of $\text{conv}(P)$.

Degeneracies. A couple of further clarifications regarding the above problem definitions are in order.

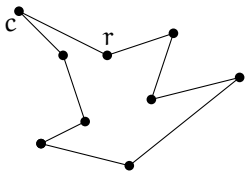
First of all, for efficiency reasons an input is usually specified as a sequence of points. Do we insist that this sequence forms a set or are duplications of points allowed?

What if three points are collinear? Are all of them considered extremal? According to our definition from above, they are not and that is what we will stick to. But note that there may be cases where one wants to include all such points, nevertheless.

By the Separation Theorem, every extremal point p can be separated from the convex hull of the remaining points by a halfplane. If we take such a halfplane and translate its defining line such that it passes through p, then all points from P other than p should lie in the resulting open halfplane. In $\mathbb{R}^2$ it turns out convenient to work with the following "directed" reformulation.

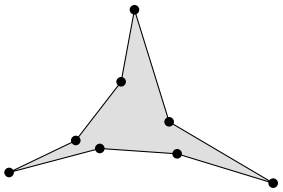
Proposition 4.19 *A point $p \in P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ is* extremal *for P $\iff$ there is a directed line g through p such that $P \setminus \{p\}$ is to the left of g.*

The *interior angle* at a vertex v of a polygon P is the angle between the two edges of P incident to v whose corresponding angular domain lies in P°. If this angle is smaller than π, the vertex is called *convex*; if the angle is larger than π, the vertex is called *reflex*. For instance, the vertex c in the polygon depicted to the right is a convex vertex, whereas the vertex labeled r is a reflex vertex.



Exercise 4.20

A set $S \subset \mathbb{R}^2$ is star-shaped *if there exists a point $c \in S$, such that for every point $p \in S$ the line segment $\overline{cp}$ is contained in S. A simple polygon with exactly three convex vertices is called a pseudotriangle (see the example shown on the right).*



In the following we consider subsets of $\mathbb{R}^2$. Prove or disprove:

a) Every convex vertex of a simple polygon lies on its convex hull.

b) Every star-shaped set is convex.

c) Every convex set is star-shaped.

d) The intersection of two convex sets is convex.

e) The union of two convex sets is convex.

f) The intersection of two star-shaped sets is star-shaped.

g) The intersection of a convex set with a star-shaped set is star-shaped.

h) Every triangle is a pseudotriangle.

i) Every pseudotriangle is star-shaped.

4.4 Trivial algorithms

One can compute the extremal points using Carathéodory's Theorem as follows: Test for every point $p \in P$ whether there are $q, r, s \in P \setminus \{p\}$ such that p is inside the triangle with vertices q, r, and s. Runtime $O(n^4)$.

Another option, inspired by the Separation Theorem: test for every pair $(p, q) \in P^2$ whether all points from $P \setminus \{p, q\}$ are to the left of the directed line through p and q (or on the line segment $\overline{pq}$). Runtime $O(n^3)$.

Exercise 4.21 *Let $P = (p_0, \ldots, p_{n-1})$ be a sequence of n points in $\mathbb{R}^2$. Someone claims that you can check by means of the following algorithm whether or not P describes the boundary of a convex polygon in counterclockwise order:*

```
bool is_convex(p_0, ..., p_{n-1}) {
    for i = 0, ..., n-1:
        if (p_i, p_{(i+1) mod n}, p_{(i+2) mod n}) form a rightturn:
            return false;
    return true;
}
```

Disprove the claim and describe a correct algorithm to solve the problem.

Exercise 4.22 *Let $P \subset \mathbb{R}^2$ be a convex polygon, given as an array `p[0]...p[n-1]` of its n vertices in counterclockwise order.*

a) Describe an $O(\log(n))$ time algorithm to determine whether a point q lies inside, outside or on the boundary of P.

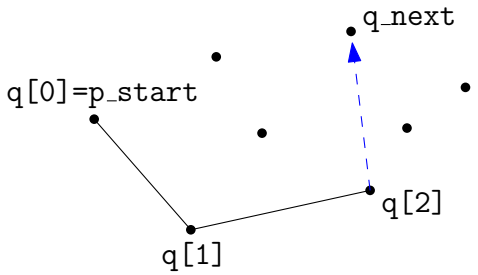
b) Describe an $O(\log(n))$ time algorithm to find a (right) tangent to P from a query point q located outside P. That is, find a vertex $p[i]$, such that P is contained in the closed halfplane to the left of the oriented line $qp[i]$.

4.5 Jarvis' Wrap

We are now ready to describe a first simple algorithm to construct the convex hull. It works as follows:

> Find a point p_1 that is a vertex of $\mathrm{conv}(P)$ (e.g., the one with smallest x-coordinate). "Wrap" P starting from p_1, i.e., always find the next vertex of $\mathrm{conv}(P)$ as the one that is rightmost with respect to the direction given by the previous two vertices.

Besides comparing x-coordinates, the only geometric primitive needed is an *orientation* test: Denote by $\texttt{rightturn}(p, q, r)$, for three points $p, q, r \in \mathbb{R}^2$, the predicate that is true if and only if r is (strictly) to the right of the oriented line pq.



Code for Jarvis' Wrap.

p[0..N) contains a sequence of N points.
p_start point with smallest x-coordinate.
q_next some *other* point in p[0..N).

```
int h = 0;
Point_2 q_now = p_start;
do {
  q[h] = q_now;
  h = h + 1;

  for (int i = 0; i < N; i = i + 1)
    if (rightturn_2(q_now, q_next, p[i]))
      q_next = p[i];

  q_now = q_next;
  q_next = p_start;
} while (q_now != p_start);
```

q[0,h) describes a convex polygon bounding the convex hull of p[0..N).

Analysis. For every output point the above algorithm spends n rightturn tests, which is $\Rightarrow O(nh)$ in total.

Theorem 4.23 [6] *Jarvis' Wrap computes the convex hull of n points in $\mathbb{R}^2$ using $O(nh)$ rightturn tests, where h is the number of hull vertices.*

In the worst case we have $h = n$, that is, $O(n^2)$ rightturn tests. Jarvis' Wrap has a remarkable property that is called *output sensitivity*: the runtime depends not only on the size of the input but also on the size of the output. For a huge point set it constructs the convex hull in optimal linear time, if the convex hull consists of a constant number of vertices only. Unfortunately the worst case performance of Jarvis' Wrap is suboptimal, as we will see soon.

Degeneracies. The algorithm may have to cope with various degeneracies.

- Several points have smallest x-coordinate $\Rightarrow$ lexicographic order:

$$(p_x, p_y) < (q_x, q_y) \iff p_x < q_x \vee p_x = q_x \wedge p_y < q_y \ .$$

- Three or more points collinear $\Rightarrow$ choose the point that is farthest among those that are rightmost.

Predicates. Besides the lexicographic comparison mentioned above, the Jarvis' Wrap (and most other 2D convex hull algorithms for that matter) need one more geometric predicate: the rightturn or—more generally—orientation test. The computation amounts to evaluating a polynomial of degree two, see the exercise below. We therefore say that the orientation test has *algebraic degree* two. In contrast, the lexicographic comparison has degree one only. The algebraic degree not only has a direct impact on the efficiency of a geometric algorithm (lower degree $\leftrightarrow$ less multiplications), but also an indirect one because high degree predicates may create large intermediate results, which may lead to overflows and are much more costly to compute with exactly.

Exercise 4.24 *Prove that for three points $(p_x, p_y), (q_x, q_y), (r_x, r_y) \in \mathbb{R}^2$, the sign of the determinant*

$$\begin{vmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{vmatrix}$$

determines if r lies to the right, to the left or on the directed line through p and q.

Exercise 4.25 *The InCircle predicate is: Given three points $p, q, r \in \mathbb{R}^2$ that define a circle C and a fourth point s, is s located inside C or not? The goal of this exercise is to derive an algebraic formulation of the incircle predicate in form of a determinant, similar to the formulation of the orientation test given above in*

Exercise 4.24. To this end we employ the so-called parabolic lifting map, which will also play a prominent role in the next chapter of the course.

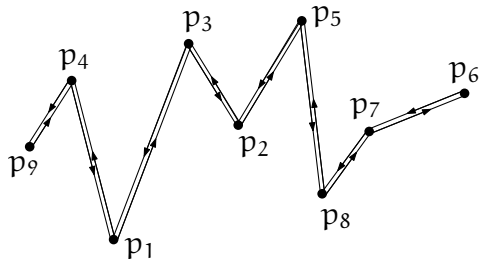
The parabolic lifting map $\ell : \mathbb{R}^2 \to \mathbb{R}^3$ *is defined for a point* $p = (x, y) \in \mathbb{R}^2$ *by* $\ell(p) = (x, y, x^2 + y^2)$. *For a circle* $C \subseteq \mathbb{R}^2$ *of positive radius, show that the "lifted circle"* $\ell(C) = \{\ell(p) \mid p \in C\}$ *is contained in a unique plane* $h_C \subseteq \mathbb{R}^3$. *Moreover, show that a point* $p \in \mathbb{R}^2$ *is strictly inside (outside, respectively) of* C *if and only if the lifted point* $\ell(p)$ *is strictly below (above, respectively)* h_C.

Use these insights to formulate the InCircle predicate for given points (p_x, p_y), (q_x, q_y), (r_x, r_y), $(s_x, s_y) \in \mathbb{R}^2$ *as a determinant.*

4.6 Graham Scan (Successive Local Repair)

There exist many algorithms that exhibit a better worst-case runtime than Jarvis' Wrap. Here we discuss only one of them: a particularly elegant and easy-to-implement variant of the so-called *Graham Scan* [5]. This algorithm is referred to as *Successive Local Repair* because it starts with some polygon enclosing all points and then step-by-step repairs the deficiencies of this polygon, by removing non-convex vertices. It goes as follows:

Sort points lexicographically and remove duplicates: $(p_1, \ldots, p_n)$.



$p_9\ p_4\ p_1\ p_3\ p_2\ p_5\ p_8\ p_7\ p_6\ p_7\ p_8\ p_5\ p_2\ p_3\ p_1\ p_4\ p_9$

As long as there is a (consecutive) triple (p, q, r) such that r is to the right of or on the directed line $\overrightarrow{pq}$, remove q from the sequence.

Code for Graham Scan.

`p[0..N)` lexicographically sorted sequence of pairwise distinct points, $N \geqslant 2$.

```
q[0] = p[0];
int h = 0;
// Lower convex hull (left to right):
for (int i = 1; i < N; i = i + 1) {
  while (h>0 && !leftturn_2(q[h-1], q[h], p[i]))
    h = h - 1;
  h = h + 1;
```

```
    q[h] = p[i];
  }

  // Upper convex hull (right to left):
  for (int i = N-2; i >= 0; i = i - 1) {
    while (!leftturn_2(q[h-1], q[h], p[i]))
      h = h - 1;
    h = h + 1;
    q[h] = p[i];
  }
```

`q[0,h)` describes a convex polygon bounding the convex hull of `p[0..N)`.

Analysis.

Theorem 4.26 *The convex hull of a set $P \subset \mathbb{R}^2$ of n points can be computed using $O(n \log n)$ geometric operations.*

Proof.

1. Sorting and removal of duplicate points: $O(n \log n)$.
2. At the beginning we have a sequence of $2n - 1$ points; at the end the sequence consists of h points. Observe that for every positive orientation test, one point is discarded from the sequence for good. Therefore, we have exactly $2n - h - 1$ such shortcuts/positive orientation tests. In addition there are at most $2n - 2$ negative tests (#iterations of the outer `for` loops). Altogether we have at most $4n - h - 3$ orientation tests.

In total the algorithm uses $O(n \log n)$ geometric operations. Note that the number of orientation tests is linear only, but $O(n \log n)$ lexicographic comparisons are needed. □

4.7 Lower Bound

It is not hard to see that the runtime of Graham Scan is asymptotically optimal in the worst-case.

Theorem 4.27 *$\Omega(n \log n)$ geometric operations are needed to construct the convex hull of n points in $\mathbb{R}^2$ (in the algebraic computation tree model).*

Proof. Reduction from sorting (for which it is known that $\Omega(n \log n)$ comparisons are needed in the algebraic computation tree model). Given n real numbers $x_1, \ldots, x_n$, construct a set $P = \{p_i \mid 1 \leqslant i \leqslant n\}$ of n points in $\mathbb{R}^2$ by setting $p_i = (x_i, x_i^2)$. This construction can be regarded as embedding the numbers into $\mathbb{R}^2$ along the x-axis and

then projecting the resulting points vertically onto the unit parabola. The order in which the points appear along the lower convex hull of P corresponds to the sorted order of the x_i. Therefore, if we could construct the convex hull in $o(n \log n)$ time, we could also sort in $o(n \log n)$ time. □

Clearly this reduction does not work for the Extremal Points problem. But using a reduction from Element Uniqueness (see Section 1.1) instead, one can show that $\Omega(n \log n)$ is also a lower bound for the number of operations needed to compute the set of extremal points only. This was first shown by Avis [1] for linear computation trees, then by Yao [9] for quadratic computation trees, and finally by Ben-Or [2] for general algebraic computation trees.

4.8 Chan's Algorithm

Given matching upper and lower bounds we may be tempted to consider the algorithmic complexity of the planar convex hull problem settled. However, this is not really the case: Recall that the lower bound is a worst case bound. For instance, the Jarvis' Wrap runs in $O(nh)$ time an thus beats the $\Omega(n \log n)$ bound in case that $h = o(\log n)$. The question remains whether one can achieve both output dependence and optimal worst case performance at the same time. Indeed, Chan [4] presented an algorithm to achieve this runtime by cleverly combining the "best of" Jarvis' Wrap and Graham Scan. Let us look at this algorithm in detail. The algorithm consists of two steps that are executed one after another.

Divide. *Input:* a set $P \subset \mathbb{R}^2$ of n points and a number $H \in \{1, \ldots, n\}$.

1. Divide P into $k = \lceil n/H \rceil$ sets $P_1, \ldots, P_k$ with $|P_i| \leqslant H$.

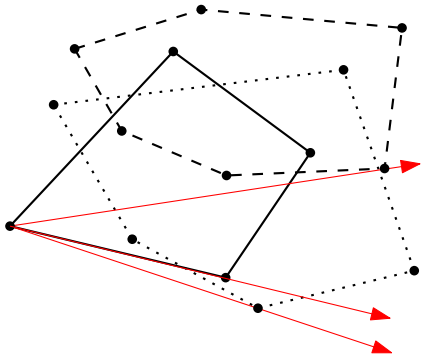
2. Construct $\text{conv}(P_i)$ for all i, $1 \leqslant i \leqslant k$.

Analysis. Step 1 takes $O(n)$ time. Step 2 can be handled using Graham Scan in $O(H \log H)$ time for any single P_i, that is, $O(n \log H)$ time in total.

Conquer. *Output:* the vertices of $\text{conv}(P)$ in counterclockwise order, if $\text{conv}(P)$ has less than H vertices; otherwise, the message that $\text{conv}(P)$ has at least H vertices.

1. Find the lexicographically smallest point in $\text{conv}(P_i)$ for all i, $1 \leqslant i \leqslant k$.

2. Starting from the lexicographically smallest point of P find the first H points of $\text{conv}(P)$ oriented counterclockwise (simultaneous Jarvis' Wrap on the sequences $\text{conv}(P_i)$).



Determine in every wrap step the point q_i of tangency from the current point of $\mathrm{conv}(P)$ to $\mathrm{conv}(P_i)$, for all $1 \leqslant i \leqslant k$. We have seen in Exercise 4.22 how to compute q_i in $O(\log|\mathrm{conv}(P_i)|) = O(\log H)$ time. Among the k candidates $q_1, \ldots, q_k$ we find the next vertex of $\mathrm{conv}(P)$ in $O(k)$ time.

Analysis. Step 1 takes $O(n)$ time. Step 2 consists of at most H wrap steps. Each wrap step needs $O(k \log H + k) = O(k \log H)$ time, which amounts to $O(Hk \log H) = O(n \log H)$ time for Step 2 in total.

Remark. Using a more clever search strategy instead of many tangency searches one can handle the conquer phase in $O(n)$ time, see Exercise 4.28 below. However, this is irrelevant as far as the asymptotic runtime is concerned, given that already the divide step takes $O(n \log H)$ time.

Exercise 4.28 *Consider k convex polygons $P_1, \ldots P_k$, for some constant $k \in \mathbb{N}$, where each polygon is given as a list of its vertices in counterclockwise orientation. Show how to construct the convex hull of $P_1 \cup \ldots \cup P_k$ in $O(n)$ time, where $n = \sum_{i=1}^{k} n_i$ and n_i is the number of vertices of P_i, for $1 \leqslant i \leqslant k$.*

Searching for h. While the runtime bound for $H = h$ is exactly what we were heading for, it looks like in order to actually run the algorithm we would have to know h, which—in general—we do not. Fortunately we can circumvent this problem rather easily, by applying what is called a *doubly exponential search*. It works as follows.

Call the algorithm from above iteratively with parameter $H = \min\{2^{2^t}, n\}$, for $t = 0, \ldots$, until the conquer step finds all extremal points of P (i.e., the wrap returns to its starting point).

Analysis: Let 2^{2^s} be the last parameter for which the algorithm is called. Since the previous call with $H = 2^{2^{s-1}}$ did not find all extremal points, we know that $2^{2^{s-1}} < h$, that is, $2^{s-1} < \log h$, where h is the number of extremal points of P. The total runtime is therefore at most

$$\sum_{i=0}^{s} cn \log 2^{2^i} = cn \sum_{i=0}^{s} 2^i = cn(2^{s+1} - 1) < 4cn \log h = O(n \log h),$$

for some constant $c \in \mathbb{R}$. In summary, we obtain the following theorem.

Theorem 4.29 *The convex hull of a set $P \subset \mathbb{R}^2$ of n points can be computed using $O(n \log h)$ geometric operations, where h is the number of convex hull vertices.*

Questions

13. *How is convexity defined? What is the convex hull of a set in $\mathbb{R}^d$?* Give at least three possible definitions.

14. *What does it mean to compute the convex hull of a set of points in* $\mathbb{R}^2$*?* Discuss input and expected output and possible degeneracies.

15. *How can the convex hull of a set of* n *points in* $\mathbb{R}^2$ *be computed efficiently?* Describe and analyze (incl. proofs) Jarvis' Wrap, Successive Local Repair, and Chan's Algorithm.

16. *Is there a linear time algorithm to compute the convex hull of* n *points in* $\mathbb{R}^2$*?* Prove the lower bound and define/explain the model in which it holds.

17. *Which geometric primitive operations are used to compute the convex hull of* n *points in* $\mathbb{R}^2$*?* Explain the two predicates and how to compute them.

Remarks. The sections on Jarvis' Wrap and Graham Scan are based on material that Emo Welzl prepared for a course on "Geometric Computing" in 2000.

References

[1] David Avis, Comments on a lower bound for convex hull determination. *Inform. Process. Lett.*, **11**, 3, (1980), 126, URL http://dx.doi.org/10.1016/0020-0190(80)90125-8.

[2] Michael Ben-Or, Lower bounds for algebraic computation trees. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pp. 80–86, 1983, URL http://dx.doi.org/10.1145/800061.808735.

[3] Constantin Carathéodory, Über den Variabilitätsbereich der Fourierschen Konstanten von positiven harmonischen Funktionen. *Rendiconto del Circolo Matematico di Palermo*, **32**, (1911), 193–217, URL http://dx.doi.org/10.1007/BF03014795.

[4] Timothy M. Chan, Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete Comput. Geom.*, **16**, 4, (1996), 361–368, URL http://dx.doi.org/10.1007/BF02712873.

[5] Ronald L. Graham, An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.*, **1**, 4, (1972), 132–133, URL http://dx.doi.org/10.1016/0020-0190(72)90045-2.

[6] Ray A. Jarvis, On the identification of the convex hull of a finite set of points in the plane. *Inform. Process. Lett.*, **2**, 1, (1973), 18–21, URL http://dx.doi.org/10.1016/0020-0190(73)90020-3.

[7] Jiří Matoušek, *Lectures on discrete geometry*. Springer-Verlag, New York, NY, 2002.

[8] Johann Radon, Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten. *Math. Annalen*, **83**, 1–2, (1921), 113–115, URL http://dx.doi.org/10.1007/BF01464231.

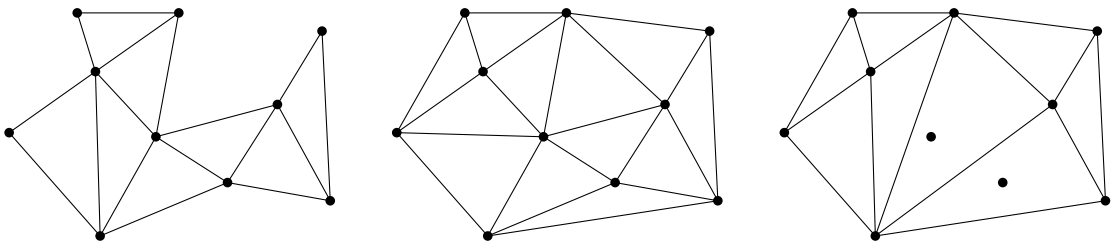
[9] Andrew C. Yao, A lower bound to finding convex hulls. *J. ACM*, **28**, 4, (1981), 780–787, URL http://dx.doi.org/10.1145/322276.322289.

Chapter 5

Delaunay Triangulations

In Chapter 3 we have discussed triangulations of simple polygons. A triangulation nicely partitions a polygon into triangles, which allows, for instance, to easily compute the area or a guarding of the polygon. Another typical application scenario is to use a triangulation T for interpolation: Suppose a function f is defined on the vertices of the polygon P, and we want to extend it "reasonably" and continuously to P°. Then for a point $p \in P^\circ$ find a triangle t of T that contains p. As p can be written as a convex combination $\sum_{i=1}^{3} \lambda_i v_i$ of the vertices v_1, v_2, v_3 of t, we just use the same coefficients to obtain an interpolation $f(p) := \sum_{i=1}^{3} \lambda_i f(v_i)$ of the function values.

If triangulations are a useful tool when working with polygons, they might also turn out useful to deal with other geometric objects, for instance, point sets. But what could be a triangulation of a point set? Polygons have a clearly defined interior, which naturally lends itself to be covered by smaller polygons such as triangles. A point set does not have an interior, except ... Here the notion of convex hull comes handy, because it allows us to treat a point set as a convex polygon. Actually, not really a convex polygon, because points in the interior of the convex hull should not be ignored completely. But one way to think of a point set is as a convex polygon—its convex hull—possibly with some holes—which are points—in its interior. A triangulation should then partition the convex hull while respecting the points in the interior, as shown in the example in Figure 5.1b.



(a) Simple polygon triangulation. (b) Point set triangulation. (c) Not a triangulation.

Figure 5.1: *Examples of (non-)triangulations.*

In contrast, the example depicted in Figure 5.1c nicely subdivides the convex hull

but should not be regarded a triangulation: Two points in the interior are not respected but simply swallowed by a large triangle.
This interpretation directly leads to the following adaption of Definition 3.8.

Definition 5.1 *A* **triangulation** *of a finite point set* $P \subset \mathbb{R}^2$ *is a collection* $\mathcal{T}$ *of triangles, such that*

(1) $\text{conv}(P) = \bigcup_{T \in \mathcal{T}} T$*;*

(2) $P = \bigcup_{T \in \mathcal{T}} V(T)$*; and*

(3) for every distinct pair $T, U \in \mathcal{T}$*, the intersection* $T \cap U$ *is either a common vertex, or a common edge, or empty.*

Just as for polygons, triangulations are universally available for point sets, meaning that (almost) every point set admits at least one.

Proposition 5.2 *Every set* $P \subseteq \mathbb{R}^2$ *of* $n \geqslant 3$ *points has a triangulation, unless all points in* P *are collinear.*

Proof. In order to construct a triangulation for P, consider the lexicographically sorted sequence $p_1, \ldots, p_n$ of points in P. Let m be minimal such that $p_1, \ldots, p_m$ are not collinear. We triangulate $p_1, \ldots, p_m$ by connecting p_m to all of $p_1, \ldots, p_{m-1}$ (which *are* on a common line), see Figure 5.2a.

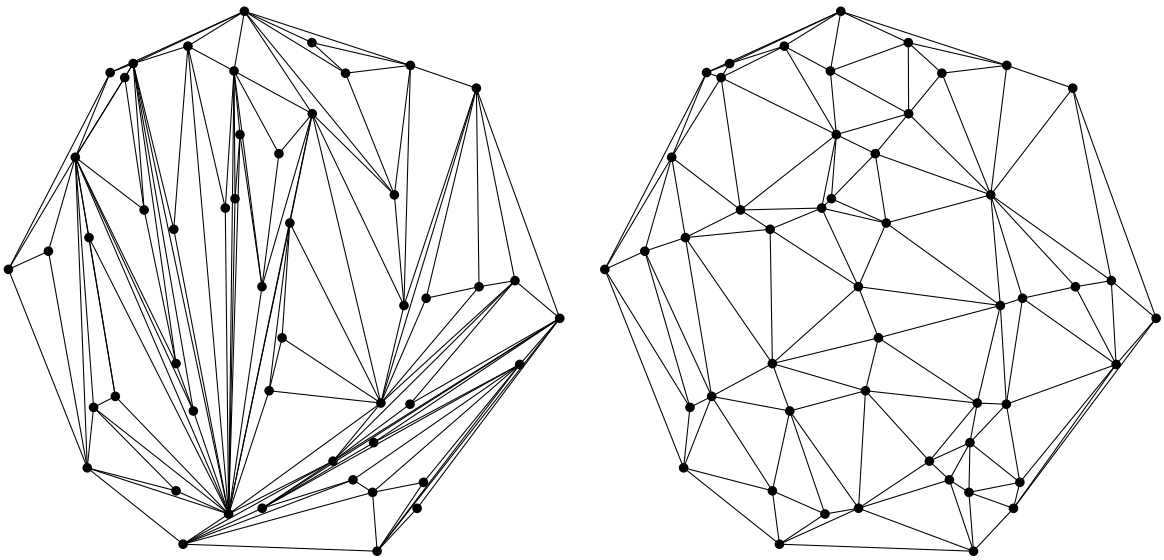
(a) Getting started. (b) Adding a point.

Figure 5.2: *Constructing the scan triangulation of* P*.*

Then we add $p_{m+1}, \ldots, p_n$. When adding p_i, for $i > m$, we connect p_i with all vertices of $C_{i-1} := \text{conv}(\{p_1, \ldots, p_{i-1}\})$ that it "sees", that is, every vertex v of C_{i-1} for which $\overline{p_i v} \cap C_{i-1} = \{v\}$. In particular, among these vertices are the two points of tangency from p_i to C_{i-1}, which shows that we always add triangles (Figure 5.2b) whose union after each step covers C_i. □

The triangulation that is constructed in Proposition 5.2 is called a *scan triangulation*. Such a triangulation (Figure 5.3a (left) shows a larger example) is usually "ugly", though, since it tends to have many long and skinny triangles. This is not just an aesthetic deficit. Having long and skinny triangles means that the vertices of a triangle tend to be spread out far from each other. You can probably imagine that such a behavior is undesirable,

for instance, in the context of interpolation. In contrast, the *Delaunay triangulation* of the same point set (Figure 5.3b) looks much nicer, and we will discuss in the next section how to get this triangulation.



(a) Scan triangulation. **(b)** Delaunay triangulation.

Figure 5.3: *Two triangulations of the same set of* 50 *points.*

Exercise 5.3 *Describe an* $O(n \log n)$ *time algorithm to construct a scan triangulation for a set of* n *points in* $\mathbb{R}^2$*.*

On another note, if you look closely into the SLR-algorithm to compute planar convex hull that was discussed in Chapter 4, then you will realize that we also could have used this algorithm in the proof of Proposition 5.2. Whenever a point is discarded during SLR, a triangle is added to the polygon that eventually becomes the convex hull.

In view of the preceding chapter, we may regard a triangulation as a plane graph: the vertices are the points in P and there is an edge between two points $p \neq q$, if and only if there is a triangle with vertices p and q. Therefore we can use Euler's formula to determine the number of edges in a triangulation.

Lemma 5.4 *Any triangulation of a set* $P \subset \mathbb{R}^2$ *of* n *points has exactly* $3n-h-3$ *edges, where* h *is the number of points from* P *on* $\partial\mathrm{conv}(P)$*.*

Proof. Consider a triangulation T of P and denote by E the set of edges and by F the set of faces of T. We count the number of edge-face incidences in two ways. Denote $\mathcal{I} = \{(e, f) \in E \times F : e \subset \partial f\}$.

On the one hand, every edge is incident to exactly two faces and therefore $|\mathcal{I}| = 2|E|$. On the other hand, every bounded face of T is a triangle and the unbounded face has h edges on its boundary. Therefore, $|\mathcal{I}| = 3(|F| - 1) + h$.

Together we obtain $3|F| = 2|E| - h + 3$. Using Euler's formula $(3n - 3|E| + 3|F| = 6)$ we conclude that $3n - |E| - h + 3 = 6$ and so $|E| = 3n - h - 3$. □

In graph theory, the term "triangulation" is sometimes used as a synonym for "maximal planar". But geometric triangulations are different, they are maximal planar in the sense that no straight-line edge can be added without sacrificing planarity.

Corollary 5.5 *A triangulation of a set $P \subset \mathbb{R}^2$ of n points is maximal planar, if and only if* $\text{conv}(P)$ *is a triangle.*

Proof. Combine Corollary 2.5 and Lemma 5.4. □

Exercise 5.6 *Find for every $n \geqslant 3$ a simple polygon P with n vertices such that P has exactly one triangulation. P should be in general position, meaning that no three vertices are collinear.*

Exercise 5.7 *Show that every set of $n \geqslant 5$ points in general position (no three points are collinear) has at least two different triangulations.*
Hint: Show first that every set of five points in general position contains a convex 4-hole, that is, a subset of four points that span a convex quadrilateral that does not contain the fifth point.

5.1 The Empty Circle Property

We will now move on to study the ominous and supposedly nice Delaunay triangulations mentioned above. They are defined in terms of an empty circumcircle property for triangles. The *circumcircle* of a triangle is the unique circle passing through the three vertices of the triangle, see Figure 5.4.

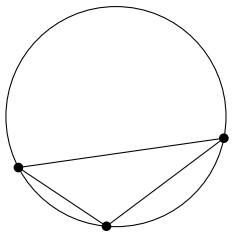


Figure 5.4: *Circumcircle of a triangle.*

Definition 5.8 *A triangulation of a finite point set $P \subset \mathbb{R}^2$ is called a* **Delaunay triangulation**, *if the circumcircle of every triangle is empty, that is, there is no point from P in its interior.*

Consider the example depicted in Figure 5.5. It shows a Delaunay triangulation of a set of six points: The circumcircles of all five triangles are empty (we also say that the triangles satisfy the empty circle property). The dashed circle is not empty, but that is fine, since it is not a circumcircle of any triangle.

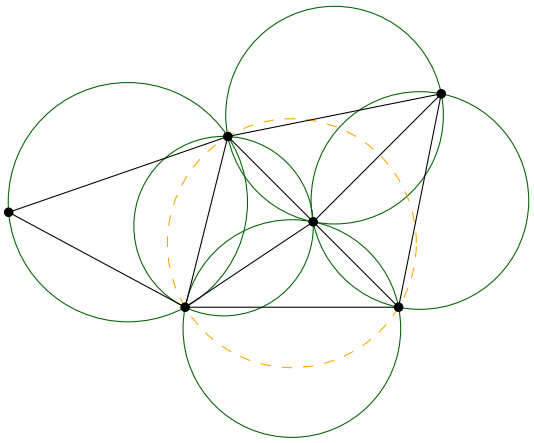
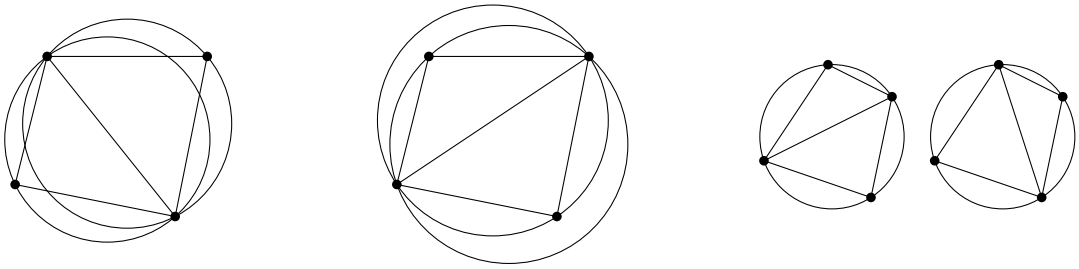


Figure 5.5: *All triangles satisfy the empty circle property.*

It is instructive to look at the case of four points in convex position. Obviously, there are two possible triangulations, but in general, only one of them will be Delaunay, see Figure 5.6a and 5.6b. If the four points are on a common circle, though, this circle is empty; at the same time it is the circumcircle of *all* possible triangles; therefore, both triangulations of the point set are Delaunay, see Figure 5.6c.



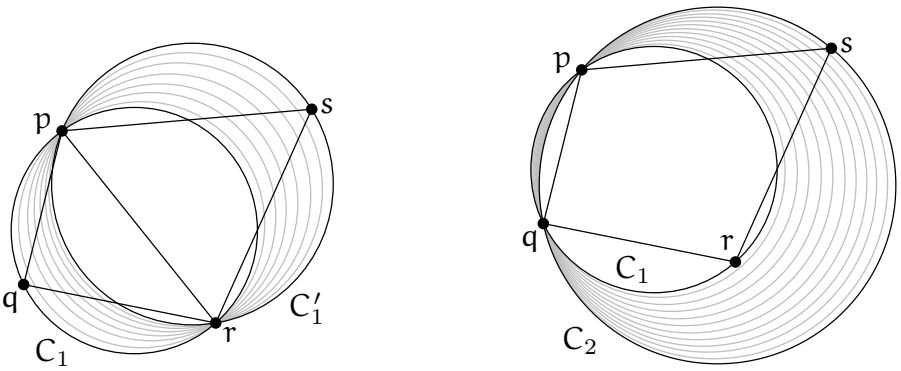
(a) Delaunay triangulation. **(b)** Non-Delaunay triangulation. **(c)** Two Delaunay triangulations.

Figure 5.6: *Triangulations of four points in convex position.*

Proposition 5.9 *Given a set* $P \subset \mathbb{R}^2$ *of four points that are in convex position but not cocircular. Then* P *has exactly one Delaunay triangulation.*

Proof. Consider a convex polygon $P = pqrs$. There are two triangulation of P: a triangulation $\mathcal{T}_1$ using the edge pr and a triangulation $\mathcal{T}_2$ using the edge qs.

Consider the family $\mathcal{C}_1$ of circles through pr, which contains the circumcircles $C_1 = pqr$ and $C_1' = rsp$ of the triangles in $\mathcal{T}_1$. By assumption s is not on C_1. If s is outside of C_1, then q is outside of C_1': Consider the process of continuously moving from C_1 to C_1' in $\mathcal{C}_1$ (Figure 5.7a); the point q is "left behind" immediately when going beyond C_1 and only the final circle C_1' "grabs" the point s.



(a) Going from C_1 to C_1' in $\mathcal{C}_1$. **(b)** Going from C_1 to C_2 in $\mathcal{C}_2$.

Figure 5.7: *Circumcircles and containment for triangulations of four points.*

Similarly, consider the family $\mathcal{C}_2$ of circles through pq, which contains the circumcircles $C_1 = pqr$ and $C_2 = spq$, the latter belonging to a triangle in $\mathcal{T}_2$. As s is outside of C_1, it follows that r is inside C_2: Consider the process of continuously moving from C_1 to C_2 in $\mathcal{C}_2$ (Figure 5.7b); the point r is on C_1 and remains within the circle all the way up to C_2. This shows that $\mathcal{T}_1$ is Delaunay, whereas $\mathcal{T}_2$ is not.

The case that s is located inside C_1 is symmetric: just cyclically shift the roles of $pqrs$ to $qrsp$. □

5.2 The Lawson Flip algorithm

It is not clear yet that every point set actually has a Delaunay triangulation (given that not all points are on a common line). In this and the next two sections, we will prove that this is the case. The proof is algorithmic. Here is the *Lawson flip algorithm* for a set P of n points.

1. Compute some triangulation of P (for example, the scan triangulation).
2. While there exists a subtriangulation of four points in convex position that is not Delaunay (like in Figure 5.6b), replace this subtriangulation by the other triangulation of the four points (Figure 5.6a).

We call the replacement operation in the second step a *(Lawson) flip*.

Theorem 5.10 *Let $P \subseteq \mathbb{R}^2$ be a set of n points, equipped with some triangulation $\mathcal{T}$. The Lawson flip algorithm terminates after at most $\binom{n}{2} = O(n^2)$ flips, and the resulting triangulation $\mathcal{D}$ is a Delaunay triangulation of P.*

We will prove Theorem 5.10 in two steps: First we show that the program described above always terminates and, therefore, is an algorithm, indeed (Section 5.3). Then we show that the algorithm does what it claims to do, namely the result is a Delaunay triangulation (Section 5.4).

5.3 Termination of the Lawson Flip Algorithm: The Lifting Map

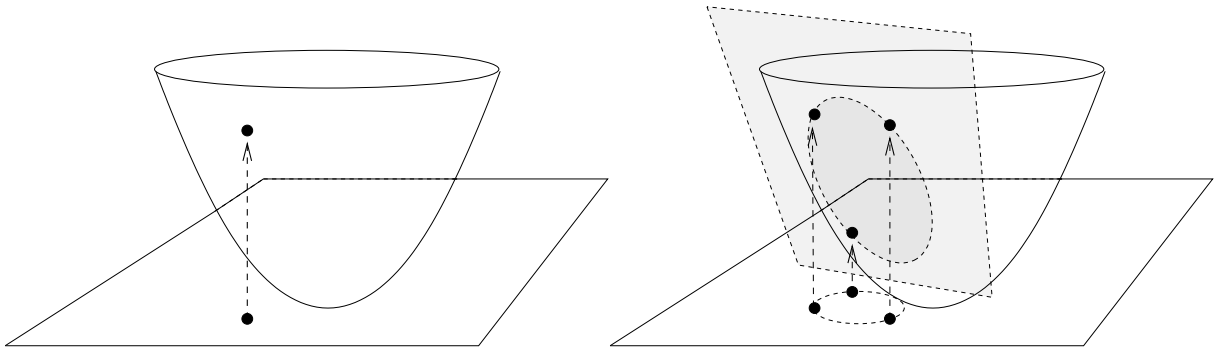
In order to prove Theorem 5.10, we invoke the (parabolic) *lifting map*. This is the following: given a point $p = (x, y) \in \mathbb{R}^2$, its *lifting* $\ell(p)$ is the point

$$\ell(p) = (x, y, x^2 + y^2) \in \mathbb{R}^3.$$

Geometrically, ℓ "lifts" the point vertically up until it lies on the *unit paraboloid*

$$\{(x, y, z) \mid z = x^2 + y^2\} \subseteq \mathbb{R}^3,$$

see Figure 5.8a.



(a) The lifting map.

(b) Points on/inside/outside a circle are lifted to points on/below/above a plane.

Figure 5.8: *The lifting map: circles map to planes.*

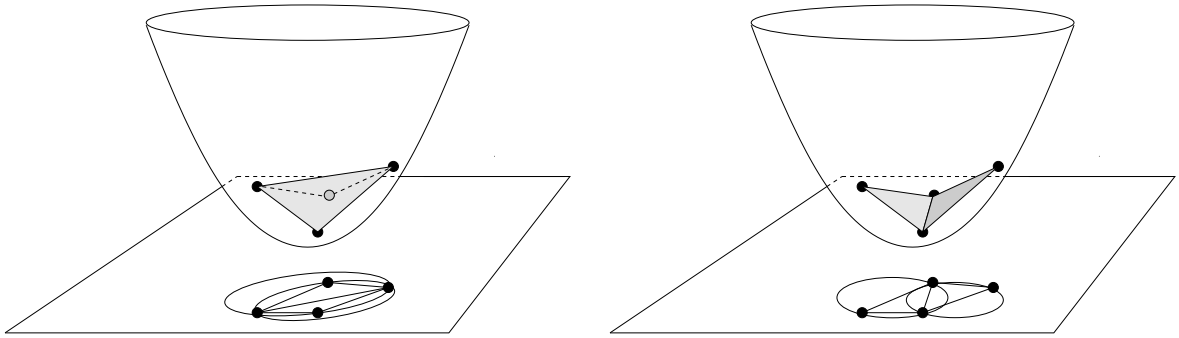
Recall the following important property of the lifting map that we proved in Exercise 4.25. It is illustrated in Figure 5.8b.

Lemma 5.11 *Let $C \subseteq \mathbb{R}^2$ be a circle of positive radius. The "lifted circle" $\ell(C) = \{\ell(p) \mid p \in C\}$ is contained in a unique plane $h_C \subseteq \mathbb{R}^3$. Moreover, a point $p \in \mathbb{R}^2$ is strictly inside (outside, respectively) of C if and only if the lifted point $\ell(p)$ is strictly below (above, respectively) h_C.*

Using the lifting map, we can now prove Theorem 5.10. Let us fix the point set P for this and the next section. First, we need to argue that the algorithm indeed terminates (if you think about it a little, this is not obvious). So let us interpret a flip operation in the lifted picture. The flip involves four points in convex position in $\mathbb{R}^2$, and their lifted images form a tetrahedron in $\mathbb{R}^3$ (think about why this tetrahedron cannot be "flat").

The tetrahedron is made up of four triangles; when you look at it from the top, you see two of the triangles, and when you look from the bottom, you see the other two. In fact, what you see from the top and the bottom are the lifted images of the two possible triangulations of the four-point set in $\mathbb{R}^2$ that is involved in the flip.

Here is the crucial fact that follows from Lemma 5.11: The two top triangles come from the non-Delaunay triangulation before the flip, see Figure 5.9a. The reason is that both top triangles have the respective fourth point below them, meaning that in $\mathbb{R}^2$, the circumcircles of these triangles contain the respective fourth point—the empty circle property is violated. In contrast, the bottom two triangles come from the Delaunay triangulation of the four points: they both have the respective fourth point above them, meaning that in $\mathbb{R}^2$, the circumcircles of the triangles do not contain the respective fourth point, see Figure 5.9b.



(a) Before the flip: the top two triangles of the tetrahedron and the corresponding non-Delaunay triangulation in the plane.

(b) After the flip: the bottom two triangles of the tetrahedron and the corresponding Delaunay triangulation in the plane.

Figure 5.9: *Lawson flip: the height of the surface of lifted triangles decreases.*

In the lifted picture, a Lawson flip can therefore be interpreted as an operation that replaces the top two triangles of a tetrahedron by the bottom two ones. If we consider the lifted image of the current triangulation, we therefore have a surface in $\mathbb{R}^3$ whose pointwise height can only decrease through Lawson flips. In particular, once an edge has been flipped, this edge will be strictly above the resulting surface and can therefore never be flipped a second time. Since n points can span at most $\binom{n}{2}$ edges, the bound on the number of flips follows.

5.4 Correctness of the Lawson Flip Algorithm

It remains to show that the triangulation of P that we get upon termination of the Lawson flip algorithm is indeed a Delaunay triangulation. Here is a first observation telling us that the triangulation is "locally Delaunay".

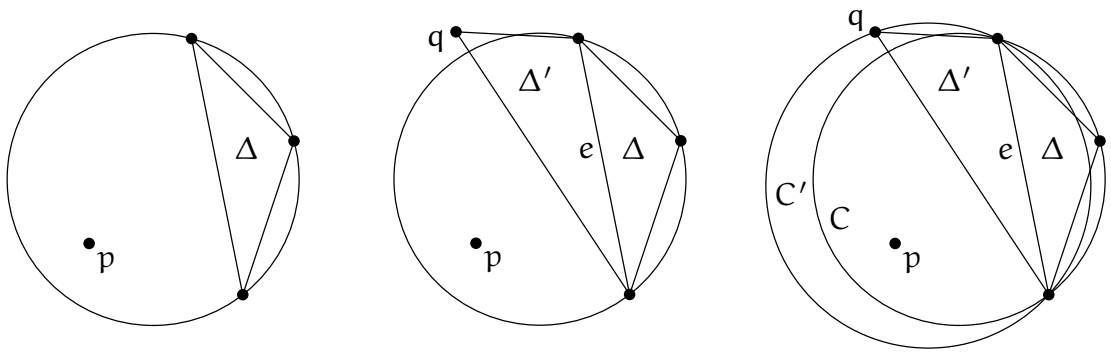
Observation 5.12 *Let Δ, Δ' be two adjacent triangles in the triangulation $\mathcal{D}$ that results from the Lawson flip algorithm. Then the circumcircle of Δ does not have any vertex of Δ' in its interior, and vice versa.*

If the two triangles together form a convex quadrilateral, this follows from the fact that the Lawson flip algorithm did not flip the common edge of Δ and Δ'. If the four vertices are not in convex position, this is basic geometry: given a triangle Δ, its circumcircle C can only contains points of $C \setminus \Delta$ that form a convex quadrilateral with the vertices of Δ.

Now we show that the triangulation is also "globally Delaunay".

Proposition 5.13 *The triangulation $\mathcal{D}$ that results from the Lawson flip algorithm is a Delaunay triangulation.*

Proof. Suppose for contradiction that there is some triangle $\Delta \in \mathcal{D}$ and some point $p \in P$ strictly inside the circumcircle C of Δ. Among all such pairs (Δ, p), we choose one for which we the distance of p to Δ is minimal. Note that this distance is positive since $\mathcal{D}$ is a triangulation of P. The situation is as depicted in Figure 5.10a.



(a) A point p inside the circumcircle C of a triangle Δ.

(b) The edge e of Δ closest to p and the second triangle Δ' incident to e.

(c) The circumcircle C' of Δ' also contains p, and p is closer to Δ' than to Δ.

Figure 5.10: *Correctness of the Lawson flip algorithm.*

Now consider the edge e of Δ that is facing p. There must be another triangle Δ' in $\mathcal{D}$ that is incident to the edge e. By the local Delaunay property of $\mathcal{D}$, the third vertex q of Δ' is on or outside of C, see Figure 5.10b. But then the circumcircle C' of Δ' contains the whole portion of C on p's side of e, hence it also contains p; moreover, p is closer to

Δ' than to Δ (Figure 5.10c). But this is a contradiction to our choice of Δ and p. Hence there was no (Δ, p), and $\mathcal{D}$ is a Delaunay triangulation. □

Exercise 5.14 *The Euclidean minimum spanning tree (EMST) of a finite point set $P \subset \mathbb{R}^2$ is a spanning tree for which the sum of the edge lengths is minimum (among all spanning trees of P). Show:*

a) Every EMST of P is a plane graph.

b) Every EMST of P contains a closest pair, i.e., an edge between two points $p, q \in P$ that have minimum distance to each other among all point pairs in $\binom{P}{2}$.

c) Every Delaunay Triangulation of P contains an EMST of P.

5.5 The Delaunay Graph

Despite the fact that a point set may have more than one Delaunay triangulation, there are certain edges that are present in every Delaunay triangulation, for instance, the edges of the convex hull.

Definition 5.15 *The* **Delaunay graph** *of $P \subseteq \mathbb{R}^2$ consists of all line segments $\overline{pq}$, for $p, q \in P$, that are contained in every Delaunay triangulation of P.*

The following characterizes the edges of the Delaunay graph.

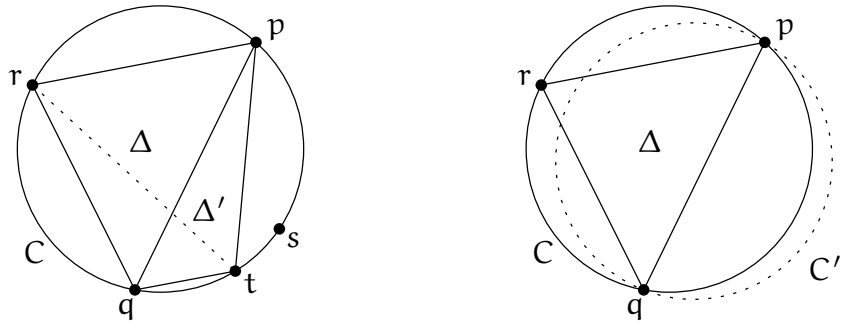
Lemma 5.16 *The segment $\overline{pq}$, for $p, q \in P$, is in the Delaunay graph of P if and only if there exists a circle through p and q that has p and q on its boundary and all other points of P are strictly outside.*

Proof. "⇒": Let pq be an edge in the Delaunay graph of P, and let $\mathcal{D}$ be a Delaunay triangulation of P. Then there exists a triangle $\Delta = pqr$ in $\mathcal{D}$, whose circumcircle C does not contain any point from P in its interior.

If there is a point s on ∂C such that $\overline{rs}$ intersects $\overline{pq}$, then let $\Delta' = pqt$ denote the other ($\neq \Delta$) triangle in $\mathcal{D}$ that is incident to pq (Figure 5.11a). Flipping the edge pq to rt yields another Delaunay triangulation of P that does not contain the edge pq, in contradiction to pq being an edge in the Delaunay graph of P. Therefore, there is no such point s.

Otherwise we can slightly change the circle C by moving away from r while keeping p and q on the circle. As P is a finite point set, we can do such a modification without catching another point from P with the circle. In this way we obtain a circle C' through p and q such that all other points from P are strictly outside C' (Figure 5.12b).

"⇐": Let $\mathcal{D}$ be a Delaunay triangulation of P. If $\overline{pq}$ is not an edge of $\mathcal{D}$, there must be another edge of $\mathcal{D}$ that crosses $\overline{pq}$ (otherwise, we could add $\overline{pq}$ to $\mathcal{D}$ and still have



(a) Another point $s \in \partial C$. (b) Moving C away from s.

Figure 5.11: *Characterization of edges in the Delaunay graph (I).*

a plane graph, a contradiction to $\mathcal{D}$ being a triangulation of P). Let rs denote the first edge of $\mathcal{D}$ that intersects the directed line segment $\overline{pq}$.

Consider the triangle Δ of $\mathcal{D}$ that is incident to rs on the side that faces p (given that $\overline{rs}$ intersects $\overline{pq}$ this is a well defined direction). By the choice of rs neither of the other two edges of Δ intersects $\overline{pq}$, and $p \notin \Delta^\circ$ because Δ is part of a triangulation of P. The only remaining option is that p is a vertex of $\Delta = prs$. As Δ is part of a Delaunay triangulation, its circumcircle C_Δ is empty (i.e., ${C_\Delta}^\circ \cap P = \emptyset$).

Consider now a circle C through p and q, which exists by assumption. Fixing p and q, expand C towards r to eventually obtain the circle C′ through p, q, and r (Figure 5.12a). Recall that r and s are on different sides of the line through p and q. Therefore, s lies strictly outside of C′. Next fix p and r and expand C′ towards s to eventually obtain the circle C_Δ through p, r, and s (Figure 5.12b). Recall that s and q are on the same side of the line through p and r. Therefore, $q \in C_\Delta$, which is in contradiction to C_Δ being empty. It follows that there is no Delaunay triangulation of P that does not contain the edge pq. □

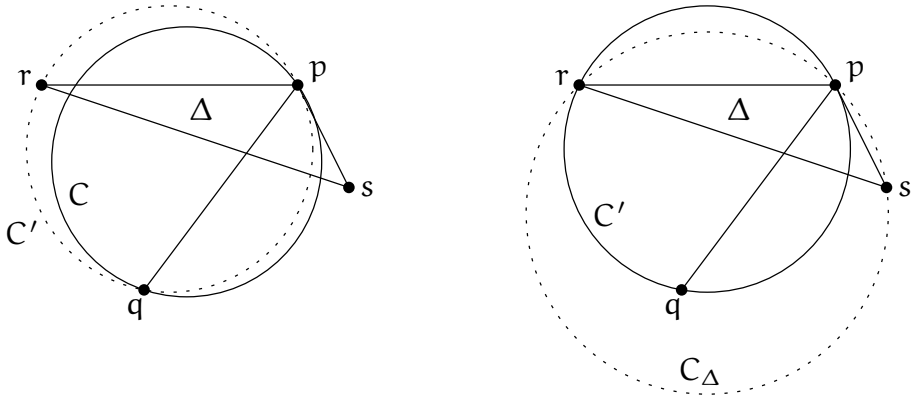
The Delaunay graph is useful to prove uniqueness of the Delaunay triangulation in case of general position.

Corollary 5.17 *Let $P \subset \mathbb{R}^2$ be a finite set of points in general position, that is, no four points of P are cocircular. Then P has a unique Delaunay triangulation.* □

5.6 Every Delaunay Triangulation Maximizes the Smallest Angle

Why are we actually interested in Delaunay triangulations? After all, having empty circumcircles is not a goal in itself. But it turns out that Delaunay triangulations satisfy a number of interesting properties. Here we show just one of them.

Recall that when we compared a scan triangulation with a Delaunay triangulation of the same point set in Figure 5.3, we claimed that the scan triangulation is "ugly" because it contains many long and skinny triangles. The triangles of the Delaunay triangulation,



(a) Expanding C towards r. (b) Expanding C′ towards s.

Figure 5.12: *Characterization of edges in the Delaunay graph (II).*

at least in this example, look much nicer, that is, much closer to an equilateral triangle. One way to quantify this "niceness" is to look at the angles that appear in a triangulation: If all angles are large, then all triangles are reasonably close to an equilateral triangle. Indeed, we will show that Delaunay triangulations maximize the smallest angle among all triangulations of a given point set. Note that this does not imply that there are no long and skinny triangles in a Delaunay triangulation. But if there is a long and skinny triangle in a Delaunay triangulation, then there is an at least as long and skinny triangle in *every* triangulation of the point set.

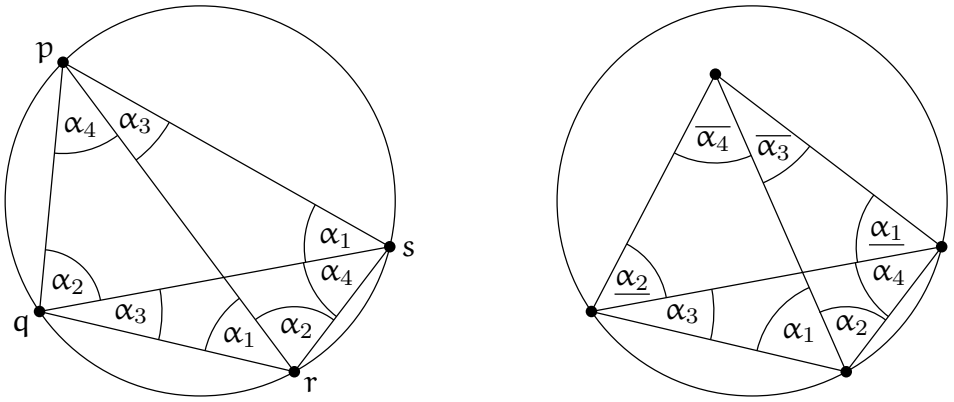
Given a triangulation $\mathcal{T}$ of P, consider the sorted sequence $A(\mathcal{T}) = (\alpha_1, \alpha_2, \ldots, \alpha_{3m})$ of interior angles, where m is the number of triangles (we have already remarked earlier that m is a function of P only and does not depend on $\mathcal{T}$). Being sorted means that $\alpha_1 \leqslant \alpha_2 \leqslant \cdots \leqslant \alpha_{3m}$. Let $\mathcal{T}, \mathcal{T}'$ be two triangulations of P. We say that $A(\mathcal{T}) < A(\mathcal{T}')$ if there exists some i for which $\alpha_i < \alpha'_i$ and $\alpha_j = \alpha'_j$, for all $j < i$. (This is nothing but the lexicographic order on these sequences.)

Theorem 5.18 *Let $P \subseteq \mathbb{R}^2$ be a finite set of points in general position (not all collinear and no four cocircular). Let $\mathcal{D}^*$ be the unique Delaunay triangulation of P, and let $\mathcal{T}$ be any triangulation of P. Then $A(\mathcal{T}) \leqslant A(\mathcal{D}^*)$.*

In particular, $\mathcal{D}^*$ maximizes the smallest angle among all triangulations of P.

Proof. We know that $\mathcal{T}$ can be transformed into $\mathcal{D}^*$ through the Lawson flip algorithm, and we are done if we can show that each such flip lexicographically increases the sorted angle sequence. A flip replaces six interior angles by six other interior angles, and we will actually show that the smallest of the six angles *strictly* increases under the flip. This implies that the whole angle sequence increases lexicographically.

Let us first look at the situation of four cocircular points, see Figure 5.13a. In this situation, the *inscribed angle theorem* (a generalization of Thales' Theorem, stated below as Theorem 5.19) tells us that the eight depicted angles come in four equal pairs.



(a) Four cocircular points and the induced eight angles.

(b) The situation before a flip.

Figure 5.13: *Angle-optimality of Delaunay triangulations.*

For instance, the angles labeled α_1 at s and r are angles on the same side of the chord pq of the circle.

In Figure 5.13b, we have the situation in which we perform a Lawson flip (replacing the solid with the dashed diagonal). By the symbol $\underline{\alpha}$ ($\overline{\alpha}$, respectively) we denote an angle strictly smaller (larger, respectively) than α. Here are the six angles before the flip:

$$\alpha_1 + \alpha_2, \quad \alpha_3, \quad \alpha_4, \quad \underline{\alpha_1}, \quad \underline{\alpha_2}, \quad \overline{\alpha_3} + \overline{\alpha_4}.$$

After the flip, we have

$$\alpha_1, \quad \alpha_2, \quad \overline{\alpha_3}, \quad \overline{\alpha_4}, \quad \underline{\alpha_1} + \alpha_4, \quad \underline{\alpha_2} + \alpha_3.$$

Now, for *every* angle after the flip there is at least one smaller angle before the flip:

$$\begin{aligned}
\alpha_1 &> \underline{\alpha_1},\\
\alpha_2 &> \underline{\alpha_2},\\
\overline{\alpha_3} &> \alpha_3,\\
\overline{\alpha_4} &> \alpha_4,\\
\underline{\alpha_1} + \alpha_4 &> \alpha_4,\\
\underline{\alpha_2} + \alpha_3 &> \alpha_3.
\end{aligned}$$

It follows that the smallest angle strictly increases. □

Theorem 5.19 (Inscribed Angle Theorem) *Let C be a circle with center c and positive radius and $p, q \in C$. Then the angle $\angle prq \bmod \pi = \frac{1}{2}\angle pcq$ is the same, for all $r \in C$.*

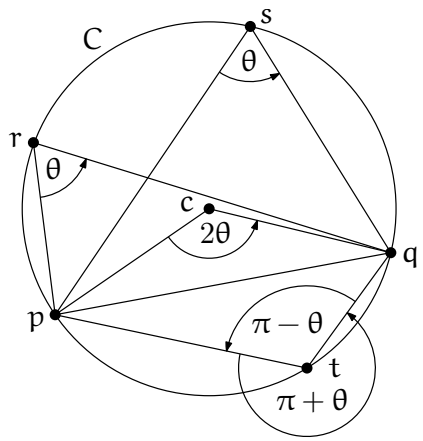
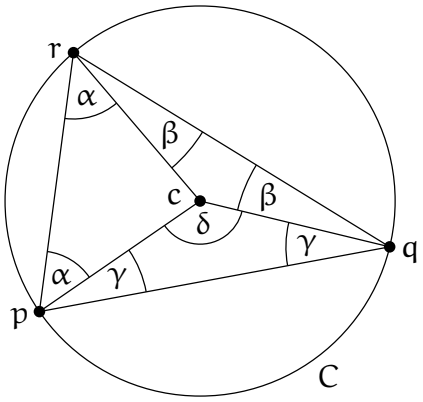


Figure 5.14: *The Inscribed Angle Theorem with* $\theta := \angle prq$.

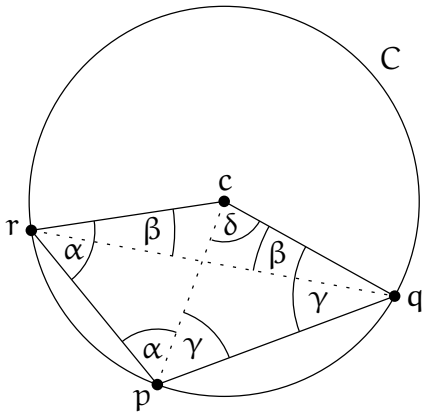
Proof. Without loss of generality we may assume that c is located to the left of or on the oriented line pq.

Consider first the case that the triangle $\Delta = pqr$ contains c. Then Δ can be partitioned into three triangles: pcr, qcr, and cpq. All three triangles are isosceles, because two sides of each form the radius of C. Denote $\alpha = \angle prc$, $\beta = \angle crq$, $\gamma = \angle cpq$, and $\delta = \angle pcq$ (see the figure shown to the right). The angles we are interested in are $\theta = \angle prq = \alpha + \beta$ and δ, for which we have to show that $\delta = 2\theta$.

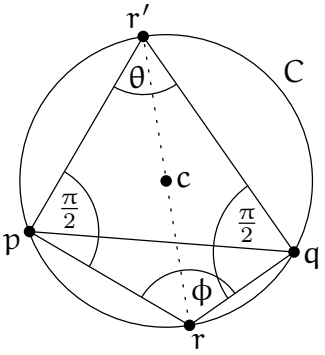


Indeed, the angle sum in Δ is $\pi = 2(\alpha + \beta + \gamma)$ and the angle sum in the triangle cpq is $\pi = \delta + 2\gamma$. Combining both yields $\delta = 2(\alpha + \beta) = 2\theta$.

Next suppose that $pqcr$ are in convex position and r is to the left of or on the oriented line pq. Without loss of generality let r be to the left of or on the oriented line qc. (The case that r lies to the right of or on the oriented line pc is symmetric.) Define α, β, γ, δ as above and observe that $\theta = \alpha - \beta$. Again have to show that $\delta = 2\theta$.



The angle sum in the triangle cpq is $\pi = \delta + 2\gamma$ and the angle sum in the triangle rpq is $\pi = (\alpha - \beta) + \alpha + \gamma + (\gamma - \beta) = 2(\alpha + \gamma - \beta)$. Combining both yields $\delta = \pi - 2\gamma = 2(\alpha - \beta) = 2\theta$.



It remains to consider the case that r is to the right of the oriented line pq.

Consider the point r′ that is antipodal to r on C, and the quadrilateral $Q = prqr'$. We are interested in the angle ϕ of Q at r. By Thales' Theorem the inner angles of Q at p and q are both $\pi/2$. Hence the angle sum of Q is $2\pi = \theta + \phi + 2\pi/2$ and so $\phi = \pi - \theta$.

□

What happens in the case where the Delaunay triangulation is not unique? The following still holds.

Theorem 5.20 *Let $P \subseteq \mathbb{R}^2$ be a finite set of points, not all on a line. Every Delaunay triangulation $\mathcal{D}$ of P maximizes the smallest angle among all triangulations $\mathcal{T}$ of P.*

Proof. Let $\mathcal{D}$ be some Delaunay triangulation of P. We infinitesimally perturb the points in P such that no four are on a common circle anymore. Then the Delaunay triangulation becomes unique (Corollary 5.17). Starting from $\mathcal{D}$, we keep applying Lawson flips until we reach the unique Delaunay triangulation $\mathcal{D}^*$ of the perturbed point set. Now we examine this sequence of flips on the original *unperturbed* point set. All these flips must involve four cocircular points (only in the cocircular case, an infinitesimal perturbation can change "good" edges into "bad" edges that still need to be flipped). But as Figure 5.13 (a) easily implies, such a "degenerate" flip does not change the smallest of the six involved angles. It follows that $\mathcal{D}$ and $\mathcal{D}^*$ have the same smallest angle, and since $\mathcal{D}^*$ maximizes the smallest angle among all triangulations $\mathcal{T}$ (Theorem 5.18), so does $\mathcal{D}$. □

5.7 Constrained Triangulations

Sometimes one would like to have a Delaunay triangulation, but certain edges are already prescribed, for example, a Delaunay triangulation of a simple polygon. Of course, one cannot expect to be able to get a proper Delaunay triangulation where all triangles satisfy the empty circle property. But it is possible to obtain some triangulation that comes as close as possible to a proper Delaunay triangulation, given that we are forced to include the edges in E. Such a triangulation is called a *constrained Delaunay triangulation*, a formal definition of which follows.

Let $P \subseteq \mathbb{R}^2$ be a finite point set and $G = (P, E)$ a geometric graph with vertex set P (we consider the edges $e \in E$ as line segments). A triangulation $\mathcal{T}$ of P *respects* G if it contains all segments $e \in E$. A triangulation $\mathcal{T}$ of P that respects G is said to be a *constrained Delaunay triangulation* of P with respect to G if the following holds for every triangle Δ of $\mathcal{T}$:

The circumcircle of Δ contains only points $q \in P$ in its interior that are not visible from the interior of Δ. A point $q \in P$ is *visible* from the interior of Δ if there exists a point p in the interior of Δ such that the line segment $\overline{pq}$ does not intersect any segment $e \in E$. We can thus imagine the line segments of E as "blocking the view".

For illustration, consider the simple polygon and its constrained Delaunay triangulation shown in Figure 5.15. The circumcircle of the shaded triangle Δ contains a whole other triangle in its interior. But these points cannot be seen from Δ°, because all possible connecting line segments intersect the blocking polygon edge e of Δ.

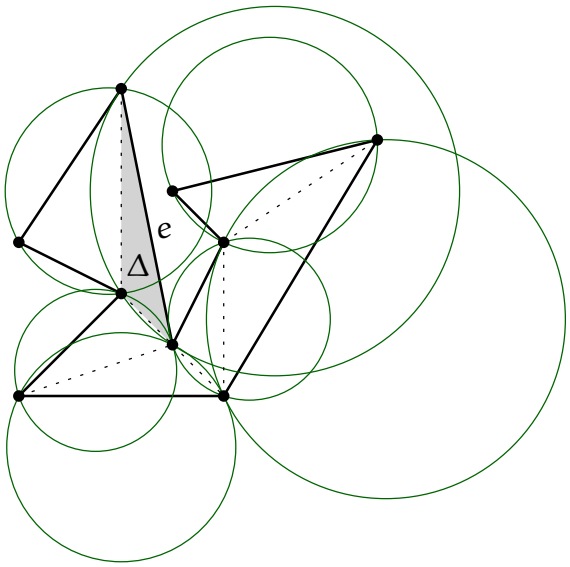


Figure 5.15: *Constrained Delaunay triangulation of a simple polygon.*

Theorem 5.21 *For every finite point set P and every plane graph $G = (P, E)$, there exists a constrained Delaunay triangulation of P with respect to G.*

Exercise 5.22 *Prove Theorem 5.21. Also describe a polynomial algorithm to construct such a triangulation.*

Questions

18. *What is a triangulation?* Provide the definition and prove a basic property: every triangulation with the same set of vertices and the same outer face has the same number of triangles.

19. *What is a triangulation of a point set?* Give a precise definition.

20. *Does every point set (not all points on a common line) have a triangulation?* You may, for example, argue with the scan triangulation.

21. *What is a Delaunay triangulation of a set of points*? Give a precise definition.

22. *What is the Delaunay graph of a point set?* Give a precise definition and a characterization.

23. *How can you prove that every set of points (not all on a common line) has a Delaunay triangulation?* You can for example sketch the Lawson flip algorithm and the Lifting Map, and use these to show the existence.

24. *When is the Delaunay triangulation of a point set unique?* Show that general position is a sufficient condition. Is it also necessary?

25. *What can you say about the "quality" of a Delaunay triangulation?* Prove that every Delaunay triangulation maximizes the smallest interior angle in the triangulation, among the set of all triangulations of the same point set.

Chapter 6

Delaunay Triangulation: Incremental Construction

In the last lecture, we have learned about the Lawson flip algorithm that computes a Delaunay triangulation of a given n-point set $P \subseteq \mathbb{R}^2$ with $O(n^2)$ Lawson flips. One can actually implement this algorithm to run in $O(n^2)$ time, and there are point sets where it may take $\Omega(n^2)$ flips.

In this lecture, we will discuss a different algorithm. The final goal is to show that this algorithm can be implemented to run in $O(n \log n)$ time; this lecture, however, is concerned only with the correctness of the algorithm. Throughout the lecture we assume that P is in general position (no 3 points on a line, no 4 points on a common circle), so that the Delaunay triangulation is unique (Corollary 5.17). There are techniques to deal with non-general position, but we don't discuss them here.

6.1 Incremental construction

The idea is to build the Delaunay triangulation of P by inserting one point after another. We always maintain the Delaunay triangulation of the point set R inserted so far, and when the next point s comes along, we simply update the triangulation to the Delaunay triangulation of $R \cup \{s\}$. Let $\mathcal{DT}(R)$ denote the Delaunay triangulation of $R \subseteq P$.

To avoid special cases, we enhance the point set P with three artificial points "far out". The convex hull of the resulting point set is a triangle; later, we can simply remove the extra points and their incident edges to obtain $\mathcal{DT}(P)$. The incremental algorithm starts off with the Delaunay triangulation of the three artificial points which consists of one big triangle enclosing all other points. (In our figures, we suppress the far-away points, since they are merely a technicality.)

Now assume that we have already built $\mathcal{DT}(R)$, and we next insert $s \in P \setminus R$. Here is the outline of the update step.

1. Find the triangle $\Delta = \Delta(p, q, r)$ of $\mathcal{DT}(R)$ that contains s, and replace it with the three triangles resulting from connecting s with all three vertices p, q, r; see Figure

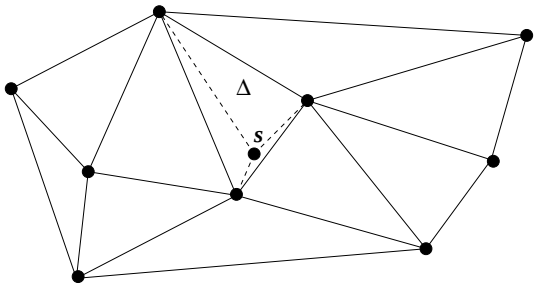


Figure 6.1: *Inserting s into $\mathcal{DT}(R)$: Step 1*

6.1. We now have a triangulation $\mathcal{T}$ of $R \cup \{s\}$.

2. Perform Lawson flips on $\mathcal{T}$ until $\mathcal{DT}(R \cup \{s\})$ is obtained; see Figure 6.2

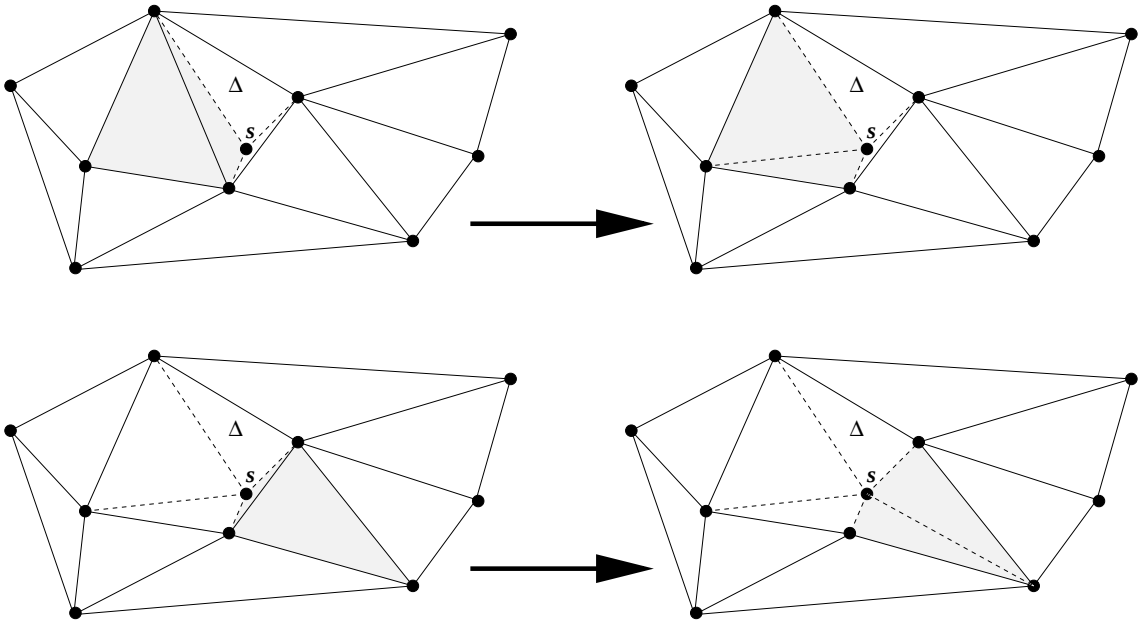


Figure 6.2: *Inserting s into $\mathcal{DT}(R)$: Step 2*

How to organize the Lawson flips. The Lawson flips can be organized quite systematically, since we always know the candidates for "bad" edges that may still have to be flipped. Initially (after step 1), only the three edges of Δ can be bad, since these are the only edges for which an incident triangle has changed (by inserting s in Step 1). Each of the three new edges is good, since the 4 vertices of its two incident triangles are not in convex position.

Now we have the following invariant (part (a) certainly holds in the first flip):

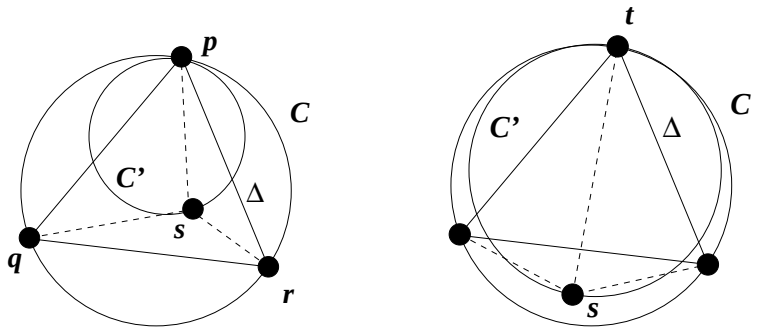
(a) In every flip, the convex quadrilateral Q in which the flip happens has exactly two edges incident to s, and the flip generates a new edge incident to s.

(b) Only the two edges of Q that are *not* incident to s can become bad through the flip.

We will prove part (b) in the next lemma. The invariant then follows since (b) entails (a) in the next flip. This means that we can maintain a queue of potentially bad edges that we process in turn. A good edge will simply be removed from the queue, and a bad edge will be flipped and replaced according to (b) with two new edges in the queue. In this way, we never flip edges incident to s; the next lemma proves that this is correct and at the same time establishes part (b) of the invariant.

Lemma 6.1 *Every edge incident to s that is created during the update is an edge of the Delaunay graph of* $R \cup \{s\}$ *and thus an edge that will be in* $\mathcal{DT}(R \cup \{s\})$. *It easily follows that edges incident to s will never become bad during the update step.*[1]

Proof. Let us consider one of the first three new edges, $\overline{sp}$, say. Since the triangle Δ has a circumcircle C strictly containing only s (Δ is in $\mathcal{DT}(R)$), we can shrink that circumcircle to a circle C' through s and p with no interior points, see Figure 6.3 (a). This proves that $\overline{sp}$ is in the Delaunay graph. If $\overline{st}$ is an edge created by a flip, a similar argument works. The flip destroys exactly one triangle Δ of $\mathcal{DT}(R)$. Its circumcircle C contains s only, and shrinking it yields an empty circle C' through s and t. Thus, $\overline{st}$ is in the Delaunay graph also in this case. □



(a) New edge $\overline{sp}$ incident to s created in Step 1

(b) New edge $\overline{st}$ incident to s created in Step 2

Figure 6.3: *Newly created edges incident to s are in the Delaunay graph*

[1] If such an edge was bad, it could be flipped, but then it would be "gone forever" according to the lifting map interpretation from the previous lecture.

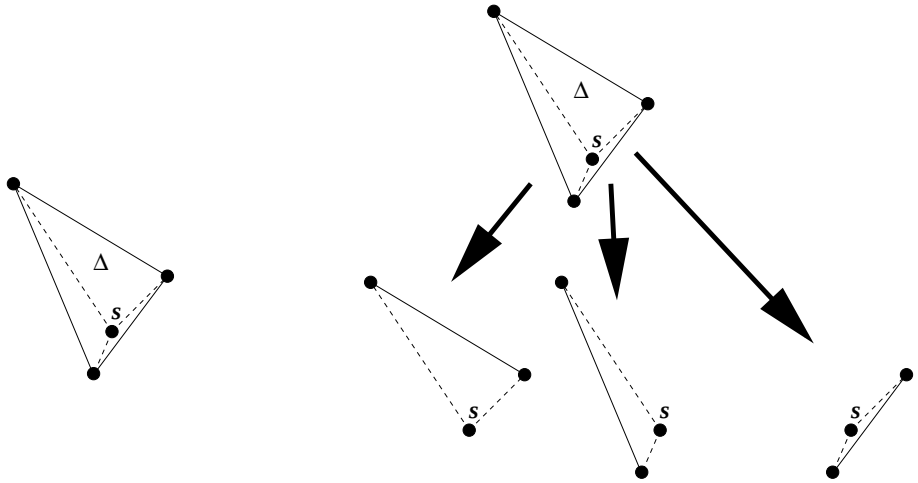


Figure 6.4: *The history graph: one triangle gets replaced by three triangles*

6.2 The History Graph

What can we say about the performance of the incremental construction? Not much yet. First of all, we did not specify how we find the triangle Δ of $\mathcal{DT}(R)$ that contains the point s to be inserted. Doing this in the obvious way (checking all triangles) is not good, since already the find steps would then amount to $O(n^2)$ work throughout the whole algorithm. Here is a smarter method, based on the *history graph*.

Definition 6.2 *Given $R \subseteq P$ (regarded as a sequence that reflects the insertion order), the history graph of R is a directed acyclic graph whose vertices are all triangles that have ever been created during the incremental construction of $\mathcal{DT}(R)$. There is a directed edge from Δ to Δ' whenever Δ has been destroyed during an insertion step, Δ' has been created during the same insertion step, and Δ overlaps with Δ' in its interior.*

It follows that the history graph contains triangles of outdegrees 3, 2 and 0. The ones of outdegree 0 are clearly the triangles of $\mathcal{DT}(R)$.

The triangles of outdegree 3 are the ones that have been destroyed during Step 1 of an insertion. For each such triangle Δ, its three outneighbors are the three new triangles that have replaced it, see Figure 6.4.

The triangles of outdegree 2 are the ones that have been destroyed during Step 2 of an insertion. For each such triangle Δ, its two outneighbors are the two new triangles created during the flip that has destroyed Δ, see Figure 6.5.

The history graph can be built during the incremental construction at asymptotically no extra cost; but it may need extra space since it keeps all triangles ever created. Given the history graph, we can search for the triangle Δ of $\mathcal{DT}(R)$ that contains s, as follows. We start from the big triangle spanned by the three far-away points; this one certainly

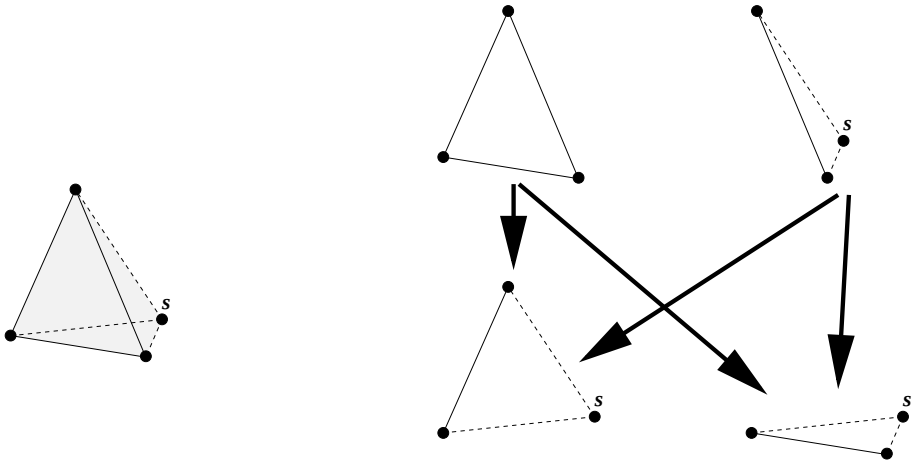


Figure 6.5: *The history graph: two triangles get replaced by two triangles*

contains s. Then we follow a directed path in the history graph. If the current triangle still has outneighbors, we find the unique outneighbor containing s and continue the search with this neighbor. If the current triangle has no outneighbors anymore, it is in $\mathcal{DT}(R)$ and contains s—we are done.

Types of triangles in the history graph. After each insertion of a point s, several triangles are created and added to the history graph. It is important to note that these triangles come in two types: Some of them are valid Delaunay triangles of $R \cup \{s\}$, and they survive to the next stage of the incremental construction. Other triangles are immediately destroyed by subsequent Lawson flips, because they are not Delaunay triangles of $R \cup \{s\}$. These "ephemeral" triangles will give us some headache (though not much) in the algorithm's analysis in the next chapter.

Note that, whenever a Lawson flip is performed, of the two triangles destroyed one of them is always a "valid" triangle from a previous iteration, and the other one is an "ephemeral" triangle that was created at this iteration. The ephemeral triangle is always the one that has s, the newly inserted point, as a vertex.

6.3 The structural change

Concerning the actual update (Steps 1 and 2), we can make the following

Observation 6.3 *Given $\mathcal{DT}(R)$ and the triangle Δ of $\mathcal{DT}(R)$ that contains s, we can build $\mathcal{DT}(R \cup \{s\})$ in time proportional to the degree of s in $\mathcal{DT}(R \cup \{s\})$, which is the number of triangles of $\mathcal{DT}(R \cup \{s\})$ containing s.*

Indeed, since every flip generates exactly one new triangle incident to s, the number of flips is the degree of s minus three. Step 1 of the update takes constant time, and

since also every flip can be implemented in constant time, the observation follows.

In the next lecture, we will show that a clever insertion order guarantees that the search paths traversed in the history graph are short, and that the structural change (the number of new triangles) is small. This will then give us the $O(n \log n)$ algorithm.

Exercise 6.4 *For a sequence of n pairwise distinct numbers $y_1, \ldots, y_n$ consider the sequence of pairs $(\min\{y_1, \ldots, y_i\}, \max\{y_1, \ldots, y_i\})_{i=0,1,\ldots,n}$ ($\min \emptyset := +\infty, \max \emptyset := -\infty$). How often do these pairs change in expectation if the sequence is permuted randomly, each permutation appearing with the same probability? Determine the expected value.*

Questions

26. *Describe the algorithm for the incremental construction of $\mathcal{DT}(P)$*: how do we find the triangle containing the point s to be inserted into $\mathcal{DT}(R)$? How do we transform $\mathcal{DT}(R)$ into $\mathcal{DT}(R \cup \{s\})$? How many steps does the latter transformation take, in terms of $\mathcal{DT}(R \cup \{s\})$?

27. *What are the two types of triangles that the history graph contains?*

Chapter 7

The Configuration Space Framework

In Section 6.1, we have discussed the incremental construction of the Delaunay triangulation of a finite point set. In this lecture, we want to analyze the runtime of this algorithm if the insertion order is chosen *uniformly at random* among all insertion orders. We will do the analysis not directly for the problem of constructing the Delaunay triangulation but in a somewhat more abstract framework, with the goal of reusing the analysis for other problems.

Throughout this lecture, we again assume general position: no three points on a line, no four on a circle.

7.1 The Delaunay triangulation — an abstract view

The incremental construction constructs and destroys triangles. In this section, we want to take a closer look at these triangles, and we want to understand exactly when a triangle is "there".

Lemma 7.1 *Given three points* $p, q, r \in R$*, the triangle* $\Delta(p, q, r)$ *with vertices* p, q, r *is a triangle of* $\mathcal{DT}(R)$ *if and only if the circumcircle of* $\Delta(p, q, r)$ *is empty of points from* R*.*

Proof. The "only if" direction follows from the definition of a Delaunay triangulation (Definition 5.8). The "if" direction is a consequence of general position and Lemma 5.16: if the circumcircle C of $\Delta(p, q, r)$ is empty of points from R, then all the three edges $\overline{pq}, \overline{qr}, \overline{pr}$ are easily seen to be in the Delaunay graph of R. C being empty also implies that the triangle $\Delta(p, q, r)$ is empty, and hence it forms a triangle of $\mathcal{DT}(R)$. □

Next we develop a somewhat more abstract view of $\mathcal{DT}(R)$.

Definition 7.2

(i) For all $p, q, r \in P$*, the triangle* $\Delta = \Delta(p, q, r)$ *is called a* configuration*. The points* p, q *and* r *are called the* defining elements *of* Δ*.*

(ii) *A configuration* Δ *is* in conflict *with a point* $s \in P$ *if* s *is strictly inside the circumcircle of* Δ. *In this case, the pair* (Δ, s) *is called a* conflict.

(iii) *A configuration* Δ *is called* active *w.r.t.* $R \subseteq P$ *if (a) the defining elements of* Δ *are in* R, *and (b) if* Δ *is not in conflict with any element of* R.

According to this definition and Lemma 7.1, $\mathcal{DT}(R)$ consists of exactly the configurations that are active w.r.t. R. Moreover, if we consider $\mathcal{DT}(R)$ and $\mathcal{DT}(R \cup \{s\})$ as sets of configurations, we can exactly say how these two sets differ.

There are the configurations in $\mathcal{DT}(R)$ that are not in conflict with s. These configurations are still in $\mathcal{DT}(R \cup \{s\})$. The configurations of $\mathcal{DT}(R)$ that *are* in conflict with s will be removed when going from R to $R \cup \{s\}$. Finally, $\mathcal{DT}(R \cup \{s\})$ contains some new configurations, all of which must have s in their defining set. According to Lemma 7.1, it cannot happen that we get a new configuration without s in its defining set, as such a configuration would have been present in $\mathcal{DT}(R)$ already.

7.2 Configuration Spaces

Here is the abstract framework that generalizes the previous configuration view of the Delaunay triangulation.

Definition 7.3 *Let* X *(the ground set) and* Π *(the set of configurations) be finite sets. Furthermore, let*

$$D : \Pi \to 2^X$$

be a function that assigns to every configuration Δ *a set of* defining elements $D(\Delta)$. *We assume that only a constant number of configurations have the same defining elements. Let*

$$K : \Pi \to 2^X$$

be a function that assigns to every configuration Δ *a set of elements* in conflict *with* Δ *(the "killer" elements). We stipulate that* $D(\Delta) \cap K(\Delta) = \emptyset$ *for all* $\Delta \in \Pi$.

Then the quadruple $\mathcal{S} = (X, \Pi, D, K)$ *is called a* configuration space. *The number*

$$d = d(\mathcal{S}) := \max_{\Delta \in \Pi} |D(\Delta)|$$

is called the dimension *of* $\mathcal{S}$.

Given $R \subseteq X$, *a configuration* Δ *is called* active *w.r.t.* R *if*

$$D(\Delta) \subseteq R \quad \textit{and} \quad K(\Delta) \cap R = \emptyset,$$

i.e. if all defining elements are in R *but no element of* R *is in conflict with* Δ. *The set of active configurations w.r.t.* R *is denoted by* $\mathcal{T}_{\mathcal{S}}(R)$, *where we drop the subscript if the configuration space is clear from the context.*

In case of the Delaunay triangulation, we set $X = P$ (the input point set). Π consists of all triangles $\Delta = \Delta(p, q, r)$ spanned by three points $p, q, r \in X \cup \{a, b, c\}$, where a, b, c are the three artificial far-away points. We set $D(\Delta) := \{p, q, r\} \cap X$. The set $K(\Delta)$ consists of all points strictly inside the circumcircle of Δ. The resulting configuration space has dimension 3, and the technical condition that only a constant number of configurations share the defining set is satisfied as well. In fact, every set of three points defines a *unique* configuration (triangle) in this case. A set of two points or one point defines three triangles (we have to add one or two artificial points which can be done in three ways). The empty set defines one triangle, the initial triangle consisting of just the three artificial points.

Furthermore, in the setting of the Delaunay triangulation, a configuration is active w.r.t. R if it is in $\mathcal{DT}(R \cup \{a, b, c\})$, i.e. we have $\mathcal{T}(R) = \mathcal{DT}(R \cup \{a, b, c\})$.

7.3 Expected structural change

Let us fix a configuration space $\mathcal{S} = (X, \Pi, D, K)$ for the remainder of this lecture. We can also interpret the incremental construction in $\mathcal{S}$. Given $R \subseteq X$ and $s \in X \setminus R$, we want to update $\mathcal{T}(R)$ to $\mathcal{T}(R \cup \{s\})$. What is the number of new configurations that arise during this step? For the case of Delaunay triangulations, this is the relevant question when we want to bound the number of Lawson flips during one update step, since this number is exactly the number of new configurations minus three.

Here is the general picture.

Definition 7.4 *For $Q \subseteq X$ and $s \in Q$, $\deg(s, Q)$ is defined as the number of configurations of $\mathcal{T}(Q)$ that have s in their defining set.*

With this, we can say that the number of new configurations in going from $\mathcal{T}(R)$ to $\mathcal{T}(R \cup \{s\})$ is precisely $\deg(s, R \cup \{s\})$, since the new configurations are by definition exactly the ones that have s in their defining set.

Now the random insertion order comes in for the first time: what is

$$E(\deg(s, R \cup \{s\})),$$

averaged over all insertion orders? In such a random insertion order, R is a random r-element subset of X (when we are about to insert the $(r+1)$-st element), and s is a random element of $X \setminus R$. Let $\mathcal{T}_r$ be the "random variable" for the set of active configurations after r insertion steps.

It seems hard to average over all R, but there is a trick: we make a movie of the randomized incremental construction, and then we watch the movie backwards. What we see is elements of X being deleted one after another, again in random order. This is due to the fact that the reverse of a random order is also random. At the point where the $(r+1)$-st element is being deleted, it is going to be a *random* element s of the currently

present $(r+1)$-element subset Q. For fixed Q, the expected degree of s is simply the average degree of an element in Q which is

$$\frac{1}{r+1}\sum_{s\in Q}\deg(s,Q) \leqslant \frac{d}{r+1}|\mathcal{T}(Q)|,$$

since the sum counts every configuration of $\mathcal{T}(Q)$ at most d times. Since Q is a random $(r+1)$-element subset, we get

$$E(\deg(s,R\cup\{s\})) \leqslant \frac{d}{r+1}t_{r+1},$$

where t_{r+1} is defined as the expected number of active configurations w.r.t. a random $(r+1)$-element set.

Here is a more formal derivation that does not use the backwards movie view. It exploits the bijection

$$(R,s) \mapsto (\underbrace{R\cup\{s\}}_{Q}, s)$$

between pairs (R,s) with $|R|=r$ and $s\notin R$ and pairs (Q,s) with $|Q|=r+1$ and $s\in Q$. Let $n=|X|$.

$$\begin{aligned}
E(\deg(s,R\cup\{s\})) &= \frac{1}{\binom{n}{r}}\sum_{R\subseteq X,|R|=r}\frac{1}{n-r}\sum_{s\in X\setminus R}\deg(s,R\cup\{s\})\\
&= \frac{1}{\binom{n}{r}}\sum_{Q\subseteq X,|Q|=r+1}\frac{1}{n-r}\sum_{s\in Q}\deg(s,Q)\\
&= \frac{1}{\binom{n}{r+1}}\sum_{Q\subseteq X,|Q|=r+1}\frac{\binom{n}{r+1}}{\binom{n}{r}}\frac{1}{n-r}\sum_{s\in Q}\deg(s,Q)\\
&= \frac{1}{\binom{n}{r+1}}\sum_{Q\subseteq X,|Q|=r+1}\frac{1}{r+1}\sum_{s\in Q}\deg(s,Q)\\
&\leqslant \frac{1}{\binom{n}{r+1}}\sum_{Q\subseteq X,|Q|=r+1}\frac{d}{r+1}|\mathcal{T}(Q)|\\
&= \frac{d}{r+1}t_{r+1}.
\end{aligned}$$

Thus, the expected number of new configurations in going from $\mathcal{T}_r$ to $\mathcal{T}_{r+1}$ is bounded by

$$\frac{d}{r+1}t_{r+1},$$

where t_{r+1} is the expected size of $\mathcal{T}_{r+1}$.

What do we get for Delaunay triangulations? We have $d = 3$ and $t_{r+1} \leqslant 2(r+4) - 4$ (the maximum number of triangles in a triangulation of $r+4$ points). Hence,

$$E(\deg(s, R \cup \{s\})) \leqslant \frac{6r+12}{r+1} \approx 6.$$

This means that on average, ≈ 3 Lawson flips are done to update $\mathcal{DT}_r$ (the Delaunay triangulation after r insertion steps) to $\mathcal{DT}_{r+1}$. Over the whole algorithm, the expected update cost is thus $O(n)$.

7.4 Bounding location costs by conflict counting

Before we can even update $\mathcal{DT}_r$ to $\mathcal{DT}_{r+1}$ during the incremental construction of the Delaunay triangulation, we need to locate the new point s in $\mathcal{DT}_r$, meaning that we need to find the triangle that contains s. We have done this with the history graph: During the insertion of s we "visit" a sequence of triangles from the history graph, each of which contains s and was created at some previous iteration $k < r$.

However, some of these visited triangles are "ephemeral" triangles (recall the discussion at the end of Section 6.2), and they present a problem to the generic analysis we want to perform. Therefore, we will do a charging scheme, so that all triangles charged are valid Delaunay triangles.

The charging scheme is as follows: If the visited triangle Δ is a valid Delaunay triangle (from some previous iteration), then we simply charge the visit of Δ during the insertion of s to the triangle-point pair (Δ, s).

If, on the other hand, Δ is an "ephemeral" triangle, then Δ was destroyed, together with some neighbor Δ', by a Lawson flip into another pair Δ'', Δ'''. Note that this neighbor Δ' was a valid triangle. Thus, in this case we charge the visit of Δ during the insertion of s to the pair (Δ', s). Observe that s is contained in the circumcircle of Δ', so s is in conflict with Δ'.

This way, we have charged each visit to a triangle in the history graph to a triangle-point pair of the form (Δ, s), such that Δ is in conflict with s. Furthermore, it is easy to see that no such pair gets charged more than once.

We define the notion of a *conflict* in general:

Definition 7.5 *A* conflict *is a configuration-element pair* (Δ, s) *where* $\Delta \in \mathcal{T}_r$ *for some* r *and* $s \in K(\Delta)$.

Thus, the running time of the Delaunay algorithm is proportional to the number of conflicts. We now proceed to derive a bound on the expected number of conflicts in the generic configuration-space framework.

7.5 Expected number of conflicts

Since every configuration involved in a conflict has been created in some step r (we include step 0), the total number of conflicts is

$$\sum_{r=0}^{n} \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)|,$$

where $\mathcal{T}_{-1} := \emptyset$. $\mathcal{T}_0$ consists of constantly many configurations only (namely those where the set of defining elements is the empty set), each of which is in conflict with at most all elements; moreover, no conflict is created in step n. Hence,

$$\sum_{r=0}^{n} \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)| = O(n) + \sum_{r=1}^{n-1} \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)|,$$

and we will bound the latter quantity. Let

$$K(r) := \sum_{\Delta \in \mathcal{T}_r \setminus \mathcal{T}_{r-1}} |K(\Delta)|, \quad r = 1, \ldots, n-1.$$

and $k(r) := E(K(r))$ the expected number of conflicts created in step r.

Bounding $k(r)$. We know that $\mathcal{T}_r$ arises from a random r-element set R. Fixing R, the backwards movie view tells us that $\mathcal{T}_{r-1}$ arises from $\mathcal{T}_r$ by deleting a random element s of R. Thus,

$$\begin{aligned} k(r) &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{1}{r} \sum_{s \in R} \sum_{\Delta \in \mathcal{T}(R) \setminus \mathcal{T}(R \setminus \{s\})} |K(\Delta)| \\ &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{1}{r} \sum_{s \in R} \sum_{\Delta \in \mathcal{T}(R), s \in D(\Delta)} |K(\Delta)| \\ &\leqslant \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{\Delta \in \mathcal{T}(R)} |K(\Delta)|, \end{aligned}$$

since in the sum over $s \in R$, every configuration is counted at most d times. Since we can rewrite

$$\sum_{\Delta \in \mathcal{T}(R)} |K(\Delta)| = \sum_{y \in X \setminus R} |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}|,$$

we thus have

$$k(r) \leqslant \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}|.$$

To estimate this further, here is a simple but crucial

Lemma 7.6 *The configurations in $\mathcal{T}(R)$ that are not in conflict with y are the configurations in $\mathcal{T}(R \cup \{y\})$ that do not have y in their defining set; in formulas:*

$$|\mathcal{T}(R)| - |\{\Delta \in \mathcal{T}(R) : y \in K(\Delta)\}| = |\mathcal{T}(R \cup \{y\})| - \deg(y, R \cup \{y\}).$$

The proof is a direct consequence of the definitions: every configuration in $\mathcal{T}(R)$ not in conflict with y is by definition still present in $\mathcal{T}(R \cup \{y\})$ and still does not have y in its defining set. And a configuration in $\mathcal{T}(R \cup \{y\})$ with y not in its defining set is by definition already present in $\mathcal{T}(R)$ and already there not in conflict with y.

The lemma implies that

$$k(r) \leqslant k_1(r) - k_2(r) + k_3(r),$$

where

$$k_1(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\mathcal{T}(R)|,$$

$$k_2(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\mathcal{T}(R \cup \{y\})|,$$

$$k_3(r) = \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} \deg(y, R \cup \{y\}).$$

Estimating $k_1(r)$. This is really simple.

$$\begin{aligned} k_1(r) &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r} \sum_{y \in X \setminus R} |\mathcal{T}(R)| \\ &= \frac{1}{\binom{n}{r}} \sum_{R \subseteq X, |R|=r} \frac{d}{r}(n-r)|\mathcal{T}(R)| \\ &= \frac{d}{r}(n-r)t_r. \end{aligned}$$

Estimating $k_2(r)$. For this, we need to employ our earlier $(R, y) \mapsto (R \cup \{y\}, y)$ bijection again.

$$\begin{aligned}
k_2(r) &= \frac{1}{\binom{n}{r}} \sum_{R\subseteq X, |R|=r} \frac{d}{r} \sum_{y\in X\setminus R} |\mathcal{T}(R\cup\{y\})| \\
&= \frac{1}{\binom{n}{r}} \sum_{Q\subseteq X, |Q|=r+1} \frac{d}{r} \sum_{y\in Q} |\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q\subseteq X, |Q|=r+1} \frac{\binom{n}{r+1}}{\binom{n}{r}} \frac{d}{r}(r+1)|\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q\subseteq X, |Q|=r+1} \frac{d}{r}(n-r)|\mathcal{T}(Q)| \\
&= \frac{d}{r}(n-r)t_{r+1} \\
&= \frac{d}{r+1}(n-(r+1))t_{r+1} + \frac{dn}{r(r+1)}t_{r+1} \\
&= k_1(r+1) + \frac{dn}{r(r+1)}t_{r+1}.
\end{aligned}$$

Estimating $k_3(r)$. This is similar to $k_2(r)$ and in addition uses a fact that we have employed before: $\sum_{y\in Q} \deg(y, Q) \leqslant d|\mathcal{T}(Q)|$.

$$\begin{aligned}
k_3(r) &= \frac{1}{\binom{n}{r}} \sum_{R\subseteq X, |R|=r} \frac{d}{r} \sum_{y\in X\setminus R} \deg(y, R\cup\{y\}) \\
&= \frac{1}{\binom{n}{r}} \sum_{Q\subseteq X, |Q|=r+1} \frac{d}{r} \sum_{y\in Q} \deg(y, Q) \\
&\leqslant \frac{1}{\binom{n}{r}} \sum_{Q\subseteq X, |Q|=r+1} \frac{d^2}{r}|\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q\subseteq X, |Q|=r+1} \frac{\binom{n}{r+1}}{\binom{n}{r}} \frac{d^2}{r}|\mathcal{T}(Q)| \\
&= \frac{1}{\binom{n}{r+1}} \sum_{Q\subseteq X, |Q|=r+1} \frac{n-r}{r+1} \cdot \frac{d^2}{r}|\mathcal{T}(Q)| \\
&= \frac{d^2}{r(r+1)}(n-r)t_{r+1} \\
&= \frac{d^2 n}{r(r+1)}t_{r+1} - \frac{d^2}{r+1}t_{r+1}.
\end{aligned}$$

Summing up. Let us recapitulate: the overall expected number of conflicts is $O(n)$ plus

$$\sum_{r=1}^{n-1} k(r) = \sum_{r=1}^{n-1} (k_1(r) - k_2(r) + k_3(r)).$$

Using our previous estimates, $k_1(2), \ldots, k_1(n-1)$ are canceled by the first terms of $k_2(1), \ldots, k_2(n-2)$. The second term of $k_2(r)$ can be combined with the first term of $k_3(r)$, so that we get

$$\begin{aligned}
\sum_{r=1}^{n-1} (k_1(r) - k_2(r) + k_3(r)) &\leqslant k_1(1) - \underbrace{k_1(n)}_{=0} + n \sum_{r=1}^{n-1} \frac{d(d-1)}{r(r+1)} t_{r+1} - \sum_{r=1}^{n-1} \frac{d^2}{r+1} t_{r+1} \\
&\leqslant d(n-1)t_1 + d(d-1)n \sum_{r=1}^{n-1} \frac{t_{r+1}}{r(r+1)} \\
&= O\left(d^2 n \sum_{r=1}^{n} \frac{t_r}{r^2} \right).
\end{aligned}$$

The Delaunay case. We have argued that the expected number of conflicts asymptotically bounds the expected total location cost over all insertion steps. The previous equation tells us that this cost is proportional to $O(n)$ plus

$$O\left(9n \sum_{r=1}^{n} \frac{2(r+3)-4}{r^2} \right) = O\left(n \sum_{r=1}^{n} \frac{1}{r} \right) = O(n \log n).$$

Here,

$$\sum_{r=1}^{n} \frac{1}{r} =: H_n$$

is the n-th Harmonic Number which is known to be approximately $\ln n$.

By going through the abstract framework of configuration spaces, we have thus analyzed the randomized incremental construction of the Delaunay triangulation of n points. According to Section 7.3, the expected update cost itself is only $O(n)$. The steps dominating the runtime are the location steps via the history graph. According to Section 7.5, all history graph searches (whose number is proportional to the number of conflicts) can be performed in expected time $O(n \log n)$, and this then also bounds the space requirements of the algorithm.

Exercise 7.7 *Design and analyze a sorting algorithm based on randomized incremental construction in configuration spaces. The input is a set S of numbers, and the output should be the sorted sequence (in increasing order).*

a) *Define an appropriate configuration space for the problem! In particular, the set of active configurations w.r.t. S should represent the desired sorted sequence.*

b) *Provide an efficient implementation of the incremental construction algorithm. "Efficient" means that the runtime of the algorithm is asymptotically dominated by the number of conflicts.*

c) *What is the expected number of conflicts (and thus the asymptotic runtime of your sorting algorithm) for a set S of n numbers?*

Questions

28. *What is a configuration space?* Give a precise definition! What is an active configuration?

29. *How do we get a configuration space from the problem of computing the Delaunay triangulation of a finite point set?*

30. *How many new active configurations do we get on average when inserting the r-th element?* Provide an answer for configuration spaces in general, and for the special case of the Delaunay triangulation.

31. *What is a conflict?* Provide an answer for configuration spaces in general, and for the special case of the Delaunay triangulation.

32. *Explain why counting the expected number of conflicts asymptotically bounds the cost for the history searches during the randomized incremental construction of the Delaunay triangulation!*

Chapter 8

Voronoi Diagrams

8.1 Post Office Problem

Suppose there are n post offices $p_1, \ldots p_n$ in a city. Someone who is located at a position q within the city would like to know which post office is closest to him.[1] Modeling the city as a planar region, we think of $p_1, \ldots p_n$ and q as points in the plane. Denote the set of post offices by $P = \{p_1, \ldots p_n\}$.

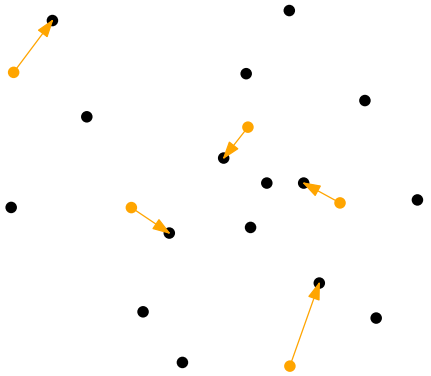


Figure 8.1: *Closest post offices for various query points.*

While the locations of post offices are known and do not change so frequently, we do not know in advance for which—possibly many—query locations the closest post office is to be found. Therefore, our long term goal is to come up with a data structure on top of P that allows to answer any possible query efficiently. The basic idea is to apply a so-called *locus approach*: we partition the query space into regions on which is the answer is the same. In our case, this amounts to partition the plane into regions such that for all points within a region the same point from P is closest (among all points from P).

[1] Another—possibly historically more accurate—way to think of the problem: You want to send a letter to a person living at q. For this you need to know the corresponding zip code, which is the code of the post office closest to q.

As a warmup, consider the problem for two post offices $p_i, p_j \in P$. For which query locations is the answer p_i rather than p_j? This region is bounded by the bisector of p_i and p_j, that is, the set of points which have the same distance to both points.

Proposition 8.1 *For any two distinct points in $\mathbb{R}^d$ the bisector is a hyperplane, that is, in $\mathbb{R}^2$ it is a line.*

Proof. Let $p = (p_1, \ldots, p_d)$ and $q = (q_1, \ldots, q_d)$ be two points in $\mathbb{R}^d$. The bisector of p and q consists of those points $x = (x_1, \ldots, x_d)$ for which

$$\|p - x\| = \|q - x\| \iff \|p - x\|^2 = \|q - x\|^2 \iff \|p\|^2 - \|q\|^2 = 2(p - q)^\top x \,.$$

As p and q are distinct, this is the equation of a hyperplane. □

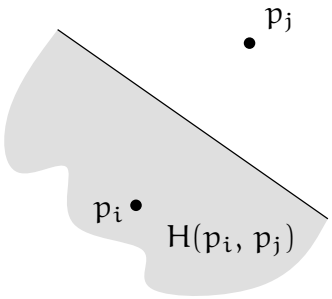


Figure 8.2: *The bisector of two points.*

Denote by $H(p_i, p_j)$ the closed halfspace bounded by the bisector of p_i and p_j that contains p_i. In $\mathbb{R}^2$, the region $H(p_i, p_j)$ is a halfplane; see Figure 8.2.

Exercise 8.2

a) What is the bisector of a line ℓ and a point $p \in \mathbb{R}^2 \setminus \ell$, that is, the set of all points $x \in \mathbb{R}^2$ with $\|x - p\| = \|x - \ell\|$ $(= \min_{q \in \ell} \|x - q\|)$?

b) For two points $p \neq q \in \mathbb{R}^2$, what is the region that contains all points whose distance to p is exactly twice their distance to q?

8.2 Voronoi Diagram

In the following we work with a set $P = \{p_1, \ldots, p_n\}$ of points in $\mathbb{R}^2$.

Definition 8.3 (Voronoi cell) *For $p_i \in P$ denote the* **Voronoi cell** $V_P(i)$ *of p_i by*

$$V_P(i) := \left\{ q \in \mathbb{R}^2 \mid \|q - p_i\| \leqslant \|q - p\| \text{ for all } p \in P \right\}.$$

Proposition 8.4

$$V_P(i) = \bigcap_{j \neq i} H(p_i, p_j) \ .$$

Proof. For $j \neq i$ we have $\|q - p_i\| \leqslant \|q - p_j\| \iff q \in H(p_i, p_j)$. □

Corollary 8.5 *$V_P(i)$ is non-empty and convex.*

Proof. According to Proposition 8.4, the region $V_P(i)$ is the intersection of a finite number of halfplanes and hence convex. As $p_i \in V_P(i)$, we have $V_P(i) \neq \emptyset$. □

Observe that every point of the plane lies in some Voronoi cell but no point lies in the interior of two Voronoi cells. Therefore these cells form a *subdivision* of the plane (a partition[2] into interior-disjoint simple polygons). See Figure 8.3 for an example.

Definition 8.6 (Voronoi Diagram) *The* **Voronoi Diagram** $VD(P)$ *of a set* $P = \{p_1, \ldots, p_n\}$ *of points in* $\mathbb{R}^2$ *is the subdivision of the plane induced by the Voronoi cells* $V_P(i)$*, for* $i = 1, \ldots, n$*. Denote by* $VV(P)$ *the set of vertices, by* $VE(P)$ *the set of edges, and by* $VR(P)$ *the set of regions (faces) of* $VD(P)$*.*

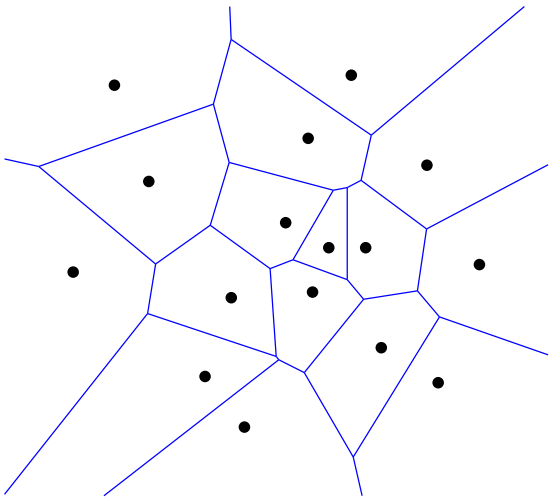


Figure 8.3: *Example: The Voronoi diagram of a point set.*

Lemma 8.7 *For every vertex* $v \in VV(P)$ *the following statements hold.*

a) v *is the common intersection of at least three edges from* $VE(P)$*;*

b) v *is incident to at least three regions from* $VR(P)$*;*

[2]Strictly speaking, to obtain a partition, we treat the shared boundaries of the polygons as separate entities.

c) v *is the center of a circle* $C(v)$ *through at least three points from* P *such that*

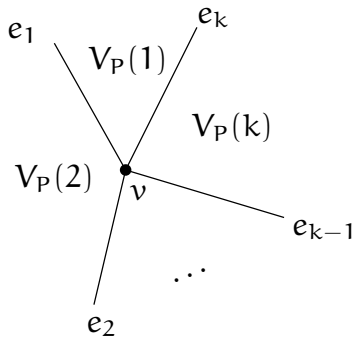
d) $C(v)^\circ \cap P = \emptyset$.

Proof. Consider a vertex $v \in VV(P)$. As all Voronoi cells are convex, $k \geqslant 3$ of them must be incident to v. This proves Part a) and b).

Without loss of generality let these cells be $V_P(i)$, for $1 \leqslant i \leqslant k$. Denote by e_i, $1 \leqslant i \leqslant k$, the edge incident to v that bounds $V_P(i)$ and $V_P((i \bmod k) + 1)$.

For any $i = 1, \ldots, k$ we have $v \in e_i \Rightarrow \|v - p_i\| = \|v - p_{(i \bmod k)+1}\|$. In other words, p_1, p_2, $\ldots$, p_k are cocircular, which proves Part c).

Part d): Suppose there exists a point $p_\ell \in C(v)^\circ$. Then the vertex v is closer to p_ℓ than it is to any of $p_1, \ldots, p_k$, in contradiction to the fact that v is contained in all of $V_P(1), \ldots, V_P(k)$. □



Corollary 8.8 *If* P *is in general position (no four points from* P *are cocircular), then for every vertex* $v \in VV(P)$ *the following statements hold.*

a) v *is the common intersection of exactly three edges from* $VE(P)$*;*

b) v *is incident to exactly three regions from* $VR(P)$*;*

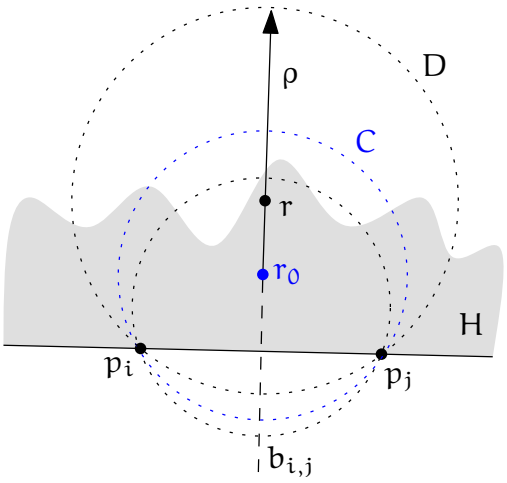
c) v *is the center of a circle* $C(v)$ *through exactly three points from* P *such that*

d) $C(v)^\circ \cap P = \emptyset$. □

Lemma 8.9 *There is an unbounded Voronoi edge bounding* $V_P(i)$ *and* $V_P(j) \iff \overline{p_i p_j} \cap P = \{p_i, p_j\}$ *and* $\overline{p_i p_j} \subseteq \partial\mathrm{conv}(P)$*, where the latter denotes the boundary of the convex hull of* P.

Proof.
Denote by $b_{i,j}$ the bisector of p_i and p_j, and let $\mathcal{D}$ denote the family of disks centered at some point on $b_{i,j}$ and passing through p_i (and p_j). There is an unbounded Voronoi edge bounding $V_P(i)$ and $V_P(j) \iff$ there is a ray $\rho \subset b_{i,j}$ such that $\|r - p_k\| > \|r - p_i\|$ $(= \|r - p_j\|)$, for every $r \in \rho$ and every $p_k \in P$ with $k \notin \{i, j\}$. Equivalently, there is a ray $\rho \subset b_{i,j}$ such that for every point $r \in \rho$ the disk $C \in \mathcal{D}$ centered at r does not contain any point from P in its interior.

The latter statement implies that the open halfplane H, whose bounding line passes through



p_i and p_j and such that H contains the infinite
part of ρ, contains no point from P in its interior.
Therefore, $\overline{p_ip_j}$ appears on $\partial\mathrm{conv}(P)$ and $\overline{p_ip_j}$ does not contain any $p_k \in P$, for $k \neq i, j$.

Conversely, suppose that $\overline{p_ip_j}$ appears on $\partial\mathrm{conv}(P)$ and $\overline{p_ip_j} \cap P = \{p_i, p_j\}$. Then some halfplane H whose bounding line passes through p_i and p_j contains no point from P in its interior. In particular, the existence of H together with $\overline{p_ip_j} \cap P = \{p_i, p_j\}$ implies that there is some disk $C \in \mathcal{D}$ such that $C \cap P = \{p_i, p_j\}$. Denote by r_0 the center of C and let ρ denote the ray starting from r_0 along $b_{i,j}$ such that the infinite part of ρ is contained in H. Consider any disk $D \in \mathcal{D}$ centered at a point $r \in \rho$ and observe that $D \setminus H \subseteq C \setminus H$. As neither H nor C contain any point from P in their respective interior, neither does D. This holds for every D, and we have seen above that this statement is equivalent to the existence of an unbounded Voronoi edge bounding $V_P(i)$ and $V_P(j)$. □

8.3 Duality

A *straight-line dual* of a plane graph G is a graph G' defined as follows: Choose a point for each face of G and connect any two such points by a straight edge, if the corresponding faces share an edge of G. Observe that this notion depends on the embedding; that is why the straight-line dual is defined for a plane graph rather than for an abstract graph. In general, G' may have edge crossings, which may also depend on the choice of representative points within the faces. However, for Voronoi diagrams is a particularly natural choice of representative points such that G' is plane: the points from P.

Theorem 8.10 (Delaunay [2]) *The straight-line dual of* $\mathrm{VD}(P)$ *for a set* $P \subset \mathbb{R}^2$ *of* $n \geqslant 3$ *points in general position (no three points from* P *are collinear and no four points from* P *are cocircular) is a triangulation: the unique Delaunay triangulation of* P.

Proof. By Lemma 8.9, the convex hull edges appear in the straight-line dual T of $\mathrm{VD}(P)$ and they correspond exactly to the unbounded edges of $\mathrm{VD}(P)$. All remaining edges of $\mathrm{VD}(P)$ are bounded, that is, both endpoints are Voronoi vertices. Consider some $v \in \mathrm{VV}(P)$. According to Corollary 8.8(b), v is incident to exactly three Voronoi regions, which, therefore, form a triangle $\triangle(v)$ in T. By Corollary 8.8(d), the circumcircle of $\triangle(v)$ does not contain any point from P in its interior. Hence $\triangle(v)$ appears in the (unique by Corollary 5.17) Delaunay triangulation of P.

Conversely, for any triangle $p_ip_jp_k$ in the Delaunay triangulation of P, by the empty circle property the circumcenter c of $p_ip_jp_k$ has p_i, p_j, and p_k as its closest points from P. Therefore, $c \in \mathrm{VV}(P)$ and—as above—the triangle $p_ip_jp_k$ appears in T. □

It is not hard to generalize Theorem 8.10 to general point sets. In this case, a Voronoi vertex of degree k is mapped to a convex polygon with k cocircular vertices. Any triangulation of such a polygon yields a Delaunay triangulation of the point set.

Corollary 8.11 $|\mathrm{VE}(P)| \leqslant 3n - 6$ *and* $|\mathrm{VV}(P)| \leqslant 2n - 5$.

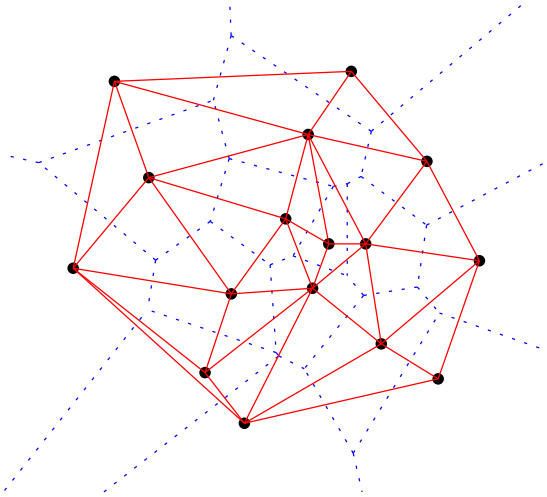


Figure 8.4: *The Voronoi diagram of a point set and its dual Delaunay triangulation.*

Proof. Every edge in $\mathrm{VE}(P)$ corresponds to an edge in the dual Delaunay triangulation. The latter is a plane graph on n vertices and thus has at most $3n-6$ edges and at most $2n-4$ faces by Corollary 2.5. Only the bounded faces correspond to a vertex in $\mathrm{VD}(P)$. □

Corollary 8.12 *For a set $P \subset \mathbb{R}^2$ of n points, the Voronoi diagram of P can be constructed in expected $O(n \log n)$ time and $O(n)$ space.*

Proof. We have seen that a Delaunay triangulation T for P can be obtained using randomized incremental construction in the given time and space bounds. As T is a plane graph, its number of vertices, edges, and faces all are linear in n. Therefore, the straight-line dual of T—which by Theorem 8.10 is the desired Voronoi diagram—can be computed in $O(n)$ additional time and space. □

Exercise 8.13 *Consider the Delaunay triangulation T for a set $P \subset \mathbb{R}^2$ of $n \geqslant 3$ points in general position. Prove or disprove:*

a) Every edge of T intersects its dual Voronoi edge.

b) Every vertex of $\mathrm{VD}(P)$ is contained in its dual Delaunay triangle.

8.4 Lifting Map

Recall the lifting map that we used in Section 5.3 to prove that the Lawson Flip Algorithm terminates. Denote by $\mathcal{U} : z = x^2 + y^2$ the unit paraboloid in $\mathbb{R}^3$. The lifting map $\ell : \mathbb{R}^2 \to \mathcal{U}$ with $\ell : p = (p_x, p_y) \mapsto (p_x, p_y, {p_x}^2 + {p_y}^2)$ is the projection of the x/y-plane onto $\mathcal{U}$ in direction of the z-axis.

For $p \in \mathbb{R}^2$ let H_p denote the plane of tangency to $\mathcal{U}$ in $\ell(p)$. Denote by $h_p : \mathbb{R}^2 \to H_p$ the projection of the x/y-plane onto H_p in direction of the z-axis (see Figure 8.5).

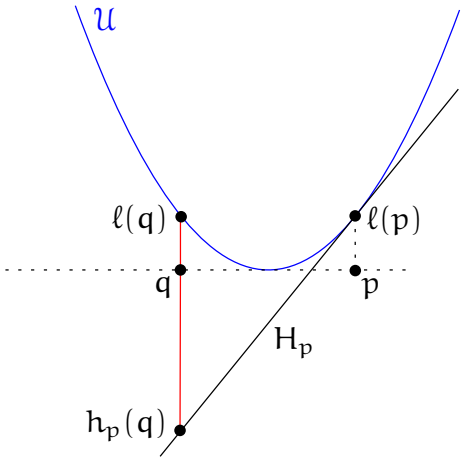


Figure 8.5: *Lifting map interpretation of the Voronoi diagram in a two-dimensional projection.*

Lemma 8.14 $\|\ell(q) - h_p(q)\| = \|p - q\|^2$, *for any points* $p, q \in \mathbb{R}^2$.

Exercise 8.15 *Prove Lemma 8.14.* Hint: *First determine the equation of the tangent plane* H_p *to* $\mathcal{U}$ *in* $\ell(p)$.

Theorem 8.16 *For* $p = (p_x, p_y) \in \mathbb{R}^2$ *denote by* H_p *the plane of tangency to the unit paraboloid* $\mathcal{U} = \{(x, y, z) : z = x^2 + y^2\} \subset \mathbb{R}^3$ *in* $\ell(p) = (p_x, p_y, {p_x}^2 + {p_y}^2)$. *Let* $\mathcal{H}(P) := \bigcap_{p \in P} H_p^+$ *the intersection of all halfspaces above the planes* H_p, *for* $p \in P$. *Then the vertical projection of* $\partial\mathcal{H}(P)$ *onto the* x/y*-plane forms the Voronoi Diagram of* P *(the faces of* $\partial\mathcal{H}(P)$ *correspond to Voronoi regions, the edges to Voronoi edges, and the vertices to Voronoi vertices).*

Proof. For any point $q \in \mathbb{R}^2$, the vertical line through q intersects every plane H_p, $p \in P$. By Lemma 8.14 the topmost plane intersected belongs to the point from P that is closest to q. □

8.5 Point location in a Voronoi Diagram

One last bit is still missing in order to solve the post office problem optimally.

Theorem 8.17 *Given a triangulation* T *for a set* $P \subset \mathbb{R}^2$ *of* n *points, one can build in* $O(n)$ *time an* $O(n)$ *size data structure that allows for any query point* $q \in \mathrm{conv}(P)$ *to find in* $O(\log n)$ *time a triangle from* T *containing* q.

The data structure we will employ is known as *Kirkpatrick's hierarchy*. But before discussing it in detail, let us put things together in terms of the post office problem.

Corollary 8.18 (Nearest Neighbor Search) *Given a set $P \subset \mathbb{R}^2$ of n points, one can build in expected $O(n \log n)$ time an $O(n)$ size data structure that allows for any query point $q \in \text{conv}(P)$ to find in $O(\log n)$ time a nearest neighbor of q among the points from P.*

Proof. First construct the Voronoi Diagram V of P in expected $O(n \log n)$ time. It has exactly n convex faces. Every unbounded face can be cut by the convex hull boundary into a bounded and an unbounded part. As we are concerned with query points within $\text{conv}(P)$ only, we can restrict our attention to the bounded parts.[3] Any convex polygon can easily be triangulated in time linear in its number of edges (= number of vertices). As V has at most $3n - 6$ edges and every edge appears in exactly two faces, V can be triangulated in $O(n)$ time overall. Label each of the resulting triangles with the point from p, whose Voronoi region contains it, and apply the data structure from Theorem 8.17. □

8.6 Kirkpatrick's Hierarchy

We will now the develop the data structure for point location in a triangulation, as described in Theorem 8.17. For simplicity we assume that the triangulation T we work with is a maximal planar graph, that is, the outer face is a triangle as well. This can easily be achieved by an initial normalization step that puts a huge triangle T_h around T and triangulates the region in between T_h and T (in linear time—how?).

The main idea for the data structure is to construct a hierarchy $T_0, \ldots, T_h$ of triangulations, such that

- $T_0 = T$,
- the vertices of T_i are a subset of the vertices of T_{i-1}, for $i = 1, \ldots, h$, and
- T_h is a single triangle only.

Search. For a query point x we can find a triangle from T that contains x as follows.

Search$(x \in \mathbb{R}^2)$

1. For $i = h, h-1, \ldots, 0$: Find a triangle t_i from T_i that contains x.
2. return t_0.

This search is efficient under the following conditions.

[3] We even know how to decide in $O(\log n)$ time whether or not a given point lies within $\text{conv}(P)$, see Exercise 4.22.

(C1) Every triangle from T_i intersects only few ($\leqslant c$) triangles from T_{i-1}. (These will then be connected via the data structure.)

(C2) h is small ($\leqslant d \log n$).

Proposition 8.19 *The search procedure described above needs* $\leqslant 3cd \log n = O(\log n)$ *orientation tests.*

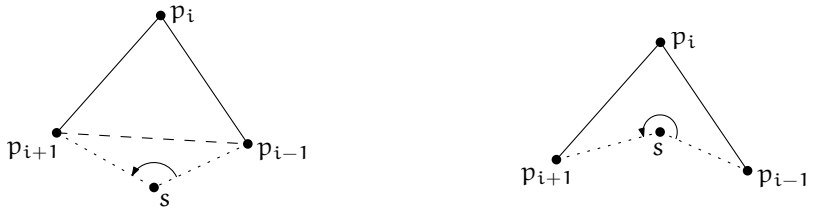
Proof. For every T_i, $0 \leqslant i < h$, at most c triangles are tested as to whether or not they contain x. Using three orientation tests one can determine whether or not a triangle contains a given point. □

Thinning. Removing a vertex v and all its incident edges from a triangulation creates a non-triangulated hole that forms a star-shaped polygon since all points are visible from v (the star-point). Here we remove vertices of constant degree only and therefore these polygons are of constant size. But even if they were not, it is not hard to triangulate a star-shaped polygon in linear time.

Lemma 8.20 *A star-shaped polygon, given as a sequence of* $n \geqslant 3$ *vertices and a star-point, can be triangulated in* $O(n)$ *time.*

Proof. For $n = 3$ there is nothing to do. For $n > 3$, consider a star-shaped polygon $P = (p_0, \ldots, p_{n-1})$ and a star-point s of P. Consider some convex vertex p_i of P, that is, $\angle p_{i+1} p_i p_{i-1} \leqslant \pi$, all indices taken $\bmod n$. (Every simple polygon on $n \geqslant 3$ vertices has at least three convex vertices, on the convex hull.)

As P is star-shaped, the quadrilateral $Q = p_{i-1} p_i p_{i+1} s$ is completely contained in P. Therefore, if $\angle p_{i-1} s p_{i+1} \leqslant \pi$ and hence Q is convex (Figure 8.6a), then we can add $p_{i-1} p_{i+1}$ as a diagonal. In this way one triangle is cut off from P, leaving a star-shaped polygon P' (with respect to s) on $n-1$ vertices. The polygon P' can then be triangulated recursively. If, on the other hand, $\angle p_{i-1} s p_{i+1} > \pi$ (Figure 8.6b), we cannot safely add



(a) $\angle p_{i-1} s p_{i+1} \leqslant \pi$. (b) $\angle p_{i-1} s p_{i+1} > \pi$.

Figure 8.6: *The quadrilateral* $p_{i-1} p_i p_{i+1} s$ *is contained in* P.

the edge $p_{i-1} p_{i+1}$ because it might intersect other edges of P or even lie outside of P. But we claim that in this case there exists another convex vertex p_j of P, for which $\angle p_{j-1} s p_{j+1} \leqslant \pi$ and therefore we can add the edge $p_{j-1} p_{j+1}$ instead.

In fact, it is enough to choose p_j to be some convex vertex of P that is not a neighbor of p_i: As $\sum_{k=0}^{n-1} \angle p_{k-1}sp_k = 2\pi$ and $\angle p_{i-1}sp_i + \angle p_isp_{i+1} = \angle p_{i-1}sp_{i+1} > \pi$, we have $\angle p_{j-1}sp_j + \angle p_jsp_{j+1} = \angle p_{j-1}sp_{j+1} < \pi$.

It remains to show that such a vertex p_j exists. P has at least three convex vertices. One of them is p_i. If the only other two convex vertices are p_{i-1} and p_{i+1}, then we make the whole argument with p_{i-1} instead of p_i and find $j = i + 1$. Note that p_{i-1} and p_{i+1} are not neighbors because P is not a triangle.

As for the linear time bound, we simply scan the sequence of vertices as in Graham's Scan. For every triple $p_{i-1}p_ip_{i+1}$ of successive vertices it is tested (in constant time) whether $p_{i-1}p_ip_{i+1}s$ forms a convex quadrilateral. If so, then the edge $p_{i-1}p_{i+1}$ is added, effectively removing p_i from further consideration. Therefore, p_i can be charged for the (potential, if there are enough vertices left) additional test of the new triple $p_xp_{i-1}p_{i+1}$ formed with the predecessor p_x of p_{i-1} in the current sequence. As shown for Graham's Scan in Theorem 4.26, this results in a linear time algorithm overall.[4] □

As a side remark, the *kernel* of a simple polygon, that is, the (possibly empty) set of all star-points, can be constructed in linear time as well using linear programming.

Our working plan is to obtain T_i from T_{i-1} by removing several *independent* (pairwise non-adjacent) vertices and re-triangulating. These vertices should

a) have small degree (otherwise the degree within the hierarchy gets too large, that is, we need to test too many triangles on the next level) and

b) be many (otherwise the height h of the hierarchy gets too large).

The following lemma asserts the existence of a sufficiently large set of independent small-degree vertices in every triangulation.

Lemma 8.21 *In every triangulation of n points in $\mathbb{R}^2$ there exists an independent set of at least $\lceil n/18 \rceil$ vertices of maximum degree 8. Moreover, such a set can be found in $O(n)$ time.*

Proof. Let $T = (V, E)$ denote the graph of the triangulation, which we consider as an abstract graph in the following. We may suppose that T is maximal planar, that is, the outer face is a triangle. (Otherwise combinatorially triangulate it arbitrarily. An independent set in the resulting graph is also independent in T.) For $n = 3$ the statement is true. Let $n \geqslant 4$.

By the Euler formula we have $|E| = 3n - 6$, that is,

$$\sum_{v \in V} \deg_T(v) = 2|E| = 6n - 12 < 6n.$$

Let $W \subseteq V$ denote the set of vertices of degree at most 8. Claim: $|W| > n/2$. Suppose $|W| \leqslant n/2$. By Theorem 2.26 we know that T is 3-connected and so every vertex has

[4] Recall that the $O(n \log n)$ time bound for Graham's Scan was caused by the initial sorting only.

degree at least three. Therefore

$$\begin{aligned}\sum_{v\in V}\deg_T(v) &= \sum_{v\in W}\deg_T(v)+\sum_{v\in V\setminus W}\deg_T(v)\geqslant 3|W|+9|V\setminus W|\\ &= 3|W|+9(n-|W|)=9n-6|W|\geqslant 9n-3n=6n,\end{aligned}$$

in contradiction to the above.

Construct an independent set U in T as follows (greedily): As long as $W\neq\emptyset$, add an arbitrary vertex $v\in W$ to U and remove v and all its neighbors from W.

Obviously U is independent and all vertices in U have degree at most 8. At each selection step at most 9 vertices are removed from W. Therefore $|U|\geqslant\lceil(n/2)/9\rceil=\lceil n/18\rceil$. □

Proof. (of Theorem 8.17)
Construct the hierarchy $T_0,\dots T_h$ with $T_0=T$ as follows. Obtain T_i from T_{i-1} by removing an independent set U as in Lemma 8.21 and re-triangulating the resulting holes. By Lemma 8.20 and Lemma 8.21 every step is linear in the number $|T_i|$ of vertices in T_i. The total cost for building the data structure is thus

$$\sum_{i=0}^{h}\alpha|T_i|\leqslant\sum_{i=0}^{h}\alpha n(17/18)^i<\alpha n\sum_{i=0}^{\infty}(17/18)^i=18\alpha n\in O(n),$$

for some constant α. Similarly the space consumption is linear.

The number of levels amounts to $h=\log_{18/17}n<12.2\log n$. Thus by Proposition 8.19 the search needs at most $3\cdot 8\cdot\log_{18/17}n<292\log n$ orientation tests. □

Improvements. As the name suggests, the hierarchical approach discussed above is due to David Kirkpatrick [5]. The constant 292 that appears in the search time is somewhat large. There has been a whole line of research trying to improve it using different techniques.

- Sarnak and Tarjan [6]: $4\log n$.
- Edelsbrunner, Guibas, and Stolfi [3]: $3\log n$.
- Goodrich, Orletsky, and Ramaiyer [4]: $2\log n$.
- Adamy and Seidel [1]: $1\log n+2\sqrt{\log n}+O(\sqrt[4]{\log n})$.

Exercise 8.22 *Let $\{p_1,p_2,\dots,p_n\}$ be a set of points in the plane, which we call* obstacles*. Imagine there is a disk of radius r centered at the origin which can be moved around the obstacles but is not allowed to intersect them (touching the boundary is ok). Is it possible to move the disk out of these obstacles? See the example depicted in Figure 8.7 below.*

More formally, the question is whether there is a (continuous) path $\gamma : [0,1] \longrightarrow \mathbb{R}^2$ *with* $\gamma(0) = (0,0)$ *and* $\|\gamma(1)\| \geqslant \max\{\|p_1\|, \ldots, \|p_n\|\}$, *such that at any time* $t \in [0,1]$ *and* $\|\gamma(t) - p_i\| \geqslant r$, *for any* $1 \leqslant i \leqslant n$. *Describe an algorithm to decide this question and to construct such a path—if one exists—given arbitrary points* $\{p_1, p_2, \ldots, p_n\}$ *and a radius* $r > 0$. *Argue why your algorithm is correct and analyze its running time.*

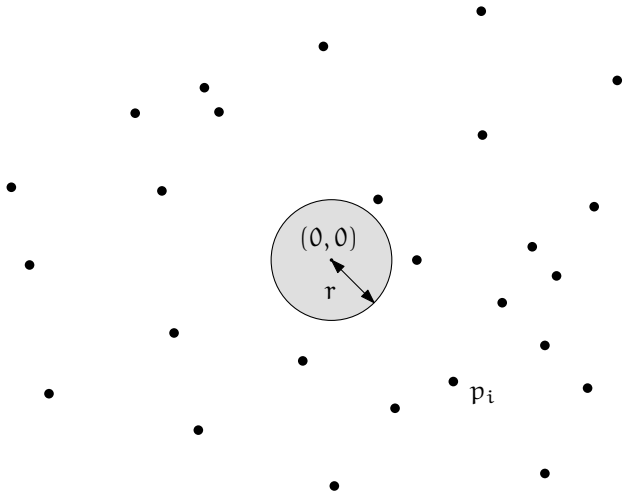


Figure 8.7: *Motion planning: Illustration for Exercise 8.22.*

Exercise 8.23 *This exercise is about an application from* Computational Biology*: You are given a set of disks* $P = \{a_1, .., a_n\}$ *in* $\mathbb{R}^2$, *all with the same radius* $r_a > 0$. *Each of these disks represents an atom of a protein. A water molecule is represented by a disc with radius* $r_w > r_a$. *A water molecule cannot intersect the interior of any protein atom, but it can be tangent to one. We say that an atom* $a_i \in P$ *is* accessible *if there exists a placement of a water molecule such that it is tangent to* a_i *and does not intersect the interior of any other atom in* P. *Given* P, *find an* $O(n \log n)$ *time algorithm which determines all atoms of* P *that are inaccessible.*

Exercise 8.24 *Let* $P \subset \mathbb{R}^2$ *be a set of* n *points. Describe a data structure to find in* $O(\log n)$ *time a point in* P *that is furthest from a given query point* q *among all points in* P.

Exercise 8.25 *Show that the bounds given in Theorem 8.17 are optimal in the algebraic computation tree model.*

Proof. There are $2n - 4$ possible answers (triangles). Therefore, any computation tree solving this problem has $\Omega(n)$ leaves and so its height is $\Omega(\log n)$. □

Questions

33. *What is the Voronoi diagram of a set of points in $\mathbb{R}^2$?* Give a precise definition and explain/prove the basic properties: convexity of cells, why is it a subdivision of the plane?, Lemma 8.7, Lemma 8.9.

34. *What is the correspondence between the Voronoi diagram and the Delaunay triangulation for a set of points in $\mathbb{R}^2$?* Prove duality (Theorem 8.10) and explain where general position is needed.

35. *How to construct the Voronoi diagram of a set of points in $\mathbb{R}^2$?* Describe an $O(n \log n)$ time algorithm, for instance, via Delaunay triangulation.

36. *How can the Voronoi diagram be interpreted in context of the lifting map?* Describe the transformation and prove its properties to obtain a formulation of the Voronoi diagram as an intersection of halfspaces one dimension higher.

37. *What is the Post-Office Problem and how can it be solved optimally?* Describe the problem and a solution using linear space, $O(n \log n)$ preprocessing, and $O(\log n)$ query time.

38. *How does Kirkpatrick's hierarchical data structure for planar point location work exactly?* Describe how to build it and how the search works, and prove the runtime bounds. In particular, you should be able to state and prove Lemma 8.21 and Theorem 8.17.

References

[1] Udo Adamy and Raimund Seidel, On the exaxt worst case query complexity of planar point location. *J. Algorithms*, **37**, (2000), 189–217, URL `http://dx.doi.org/10.1006/jagm.2000.1101`.

[2] Boris Delaunay, Sur la sphère vide. A la memoire de Georges Voronoi. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, **7**, (1934), 793–800.

[3] Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi, Optimal point location in a monotone subdivision. *SIAM J. Comput.*, **15**, 2, (1986), 317–340, URL `http://dx.doi.org/10.1137/0215023`.

[4] Michael T. Goodrich, Mark W. Orletsky, and Kumar Ramaiyer, Methods for achieving fast query times in point location data structures. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pp. 757–766, 1997, URL `http://doi.acm.org/10.1145/314161.314438`.

[5] David G. Kirkpatrick, Optimal search in planar subdivisions. *SIAM J. Comput.*, **12**, 1, (1983), 28–35, URL `http://dx.doi.org/10.1137/0212002`.

[6] Neil Sarnak and Robert E. Tarjan, Planar point location using persistent search trees. *Commun. ACM*, **29**, 7, (1986), 669–679, URL http://dx.doi.org/10.1145/6138.6151.

Chapter 9

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? Observe that points and hyperplanes in $\mathbb{R}^d$ are very similar objects, given that both can 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 line $p^* : y = p_x x - p_y$ and a non-vertical line $g : y = mx + b$ to the point $g^* = (m, -b)$.

Proposition 9.1 *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 9.2 *Prove Proposition 9.1.*

Exercise 9.3 *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.

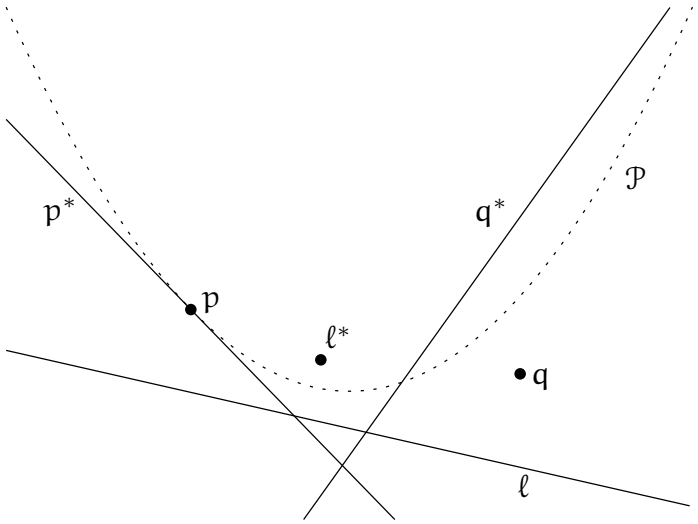


Figure 9.1: *Point* $\leftrightarrow$ *line duality with respect to the parabola* $\mathcal{P} : y = \frac{1}{2}x^2$.

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.

9.1 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 9.4 *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 9.5 *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 9.6 *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.*

9.2 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 9.2 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.

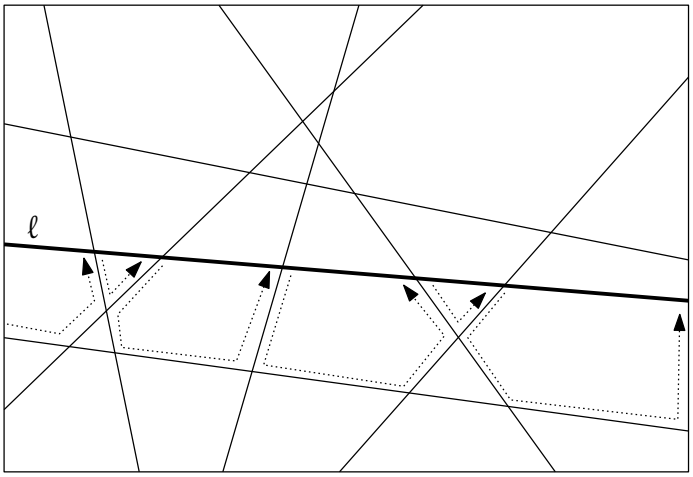


Figure 9.2: *Incremental construction: Insertion of a line ℓ. (Only part of the arrangement is shown in order to increase readability.)*

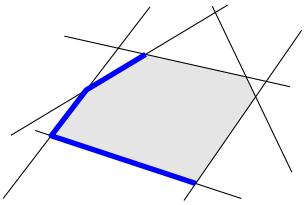
The insertion of a new vertex involves splitting two halfedges and thus is a constant time operation. But what is the time needed for the traversal? The complexity of $\mathcal{A}(\{\ell_1, \ldots, \ell_{i-1}\})$ is $\Theta(i^2)$, but we will see that the region traversed by a single line has linear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

9.3 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 9.7 *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 $6n$.*

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 to the right all left-bounding edges are shown blue and bold.

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

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

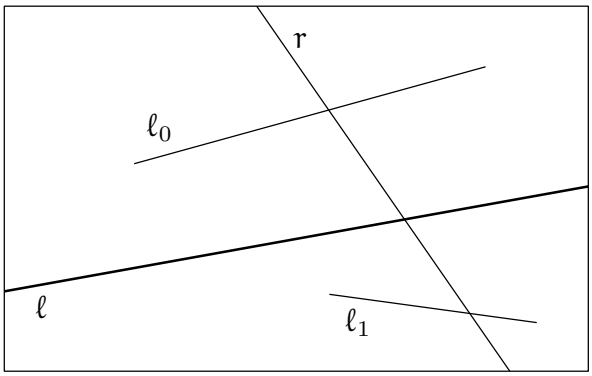


Figure 9.3: *At most three new left-bounding edges are created by adding* r *to* $\mathcal{A}(L\setminus\{r\})$.

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

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

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

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

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

9.4 The Power of Duality

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

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

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

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

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

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

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

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

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

9.5 Rotation Systems—Sorting all Angular Sequences

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

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

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

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

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

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

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

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

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

9.6 3-Sum

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

Let $(s_1, \ldots, s_n)$ 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 [4] present a deterministic algorithm that solves 3-Sum in $O(n^2(\log\log n/\log n)^{2/3})$ time. They also give a bound of $O(n^{3/2}\sqrt{\log n})$ on

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

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

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

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

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

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

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

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

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

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

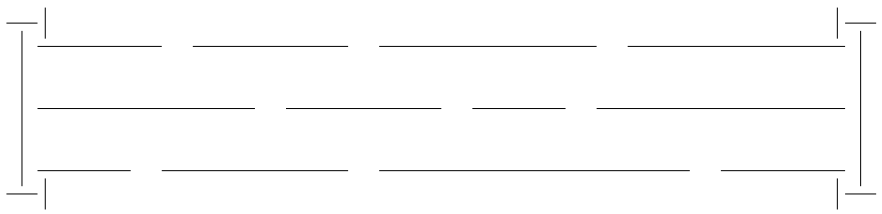
[3] In light of the recent results of Grønlund and Pettie one should probably write $O(n^{2-\varepsilon})$ here. Anyway, the reductions discussed here will be either linear or $O(n \log n)$ time.

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

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

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

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



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

In **Segment Visibility**, we are given a set S of n horizontal line segments and two segments $s_1, s_2 \in S$. The question is: Are there two points, $p_1 \in s_1$ and $p_2 \in s_2$ which can see each other, that is, the open line segment $\overline{p_1 p_2}$ 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 9.15 *The 3-Sum' problem is defined as follows: given three sets S_1, S_2, S_3 of n integers each, are there $a_1 \in S_1$, $a_2 \in S_2$, $a_3 \in S_3$ such that $a_1 + a_2 + a_3 = 0$? Prove that the 3-Sum' problem and the 3-Sum problem as defined in the lecture ($S_1 = S_2 = S_3$) are equivalent, more precisely, that they are reducible to each other in subquadratic time.*

9.7 Ham Sandwich Theorem

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

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

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

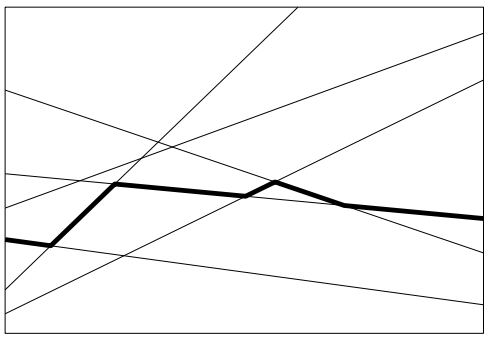


Figure 9.4: *The 3-level of an arrangement.*

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

Theorem 9.17 *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 9.5 for an example). Any point that lies on both median levels corresponds to a primal line that bisects both point sets simultaneously. □

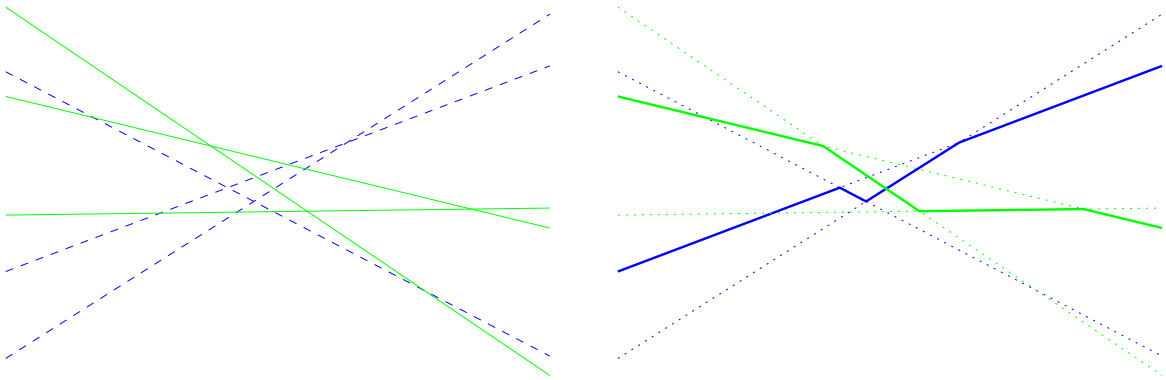


Figure 9.5: *An arrangement of 3 green lines (solid) and 3 blue lines (dashed) and their median levels (marked bold on the right hand side).*

How can the thieves use Theorem 9.17? If they are smart, they drape the necklace along some convex curve, say, a circle. Then by Theorem 9.17 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. It is easy to turn the proof given above into an $O(n^2)$ algorithm to construct a line that simultaneously bisects both sets.

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 see, for instance, Matoušek's book [8].

Theorem 9.18 *Let $P_1, \dots, P_d \subset \mathbb{R}^d$ be finite sets of points. Then there exists a hyperplane H that simultaneously bisects all of $P_1, \dots, P_d$. That is, in either open halfspace defined by H there are no more than $|P_i|/2$ points from P_i, for every $i \in \{1, \dots, d\}$.*

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 [7]. 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. [7], 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[4] in d to decide the following question [6]: Given $d \in \mathbb{N}$, finite point sets $P_1, \dots, P_d \subset \mathbb{R}^d$, and a point $p \in \bigcup_{i=1}^d P_i$, is there a ham sandwich cut through p?

Exercise 9.19 *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 $O(n)$, which 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 9.20 *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).*

Questions

39. *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).

[4] 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.

40. *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.

41. *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.

42. *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.

43. *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.

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

References

[1] David Eppstein, Happy endings for flip graphs. *J. Comput. Geom.*, **1**, 1, (2010), 3–28, URL http://jocg.org/index.php/jocg/article/view/21.

[2] Jeff Erickson, Lower bounds for linear satisfiability problems. *Chicago J. Theoret. Comput. Sci.*, **1999**, 8, URL http://cjtcs.cs.uchicago.edu/articles/1999/8/contents.html.

[3] 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, URL http://dx.doi.org/10.1016/0925-7721(95)00022-2.

[4] Allan Grønlund and Seth Pettie, Threesomes, Degenerates, and Love Triangles. In *Proc. 55th Annu. IEEE Sympos. Found. Comput. Sci.*, p. to appear, 2014, URL http://arxiv.org/abs/1404.0799.

[5] Heiko Harborth, Konvexe Fünfecke in ebenen Punktmengen. *Elem. Math.*, **33**, (1979), 116–118, URL http://dx.doi.org/10.5169/seals-32945.

[6] 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, URL http://dx.doi.org/10.4230/LIPIcs.STACS.2011.649.

[7] Chi-Yuan Lo, Jiří Matoušek, and William L. Steiger, Algorithms for ham-sandwich cuts. *Discrete Comput. Geom.*, **11**, (1994), 433–452, URL http://dx.doi.org/10.1007/BF02574017.

[8] Jiří Matoušek, *Using the Borsuk–Ulam theorem*. Springer-Verlag, Berlin, 2003, URL http://dx.doi.org/10.1007/978-3-540-76649-0.