CHAPTER 2

Planar Graphs

Es nimmt mich Wunder, dass diese allgemeinen proprietates in der Stereometrie noch von Niemand, so viel mir bekannt, sind angemerkt worden; doch viel mehr aber, dass die fürnehmsten davon als theor. 6 und theor. 11 so schwer zu beweisen sind, denn ich kann dieselben noch nicht so beweisen, dass ich damit zufrieden bin.

[It seems miraculous to me that these general results in solid geometry have not been noticed by anyone yet, as far as I know; but even more, that results as fundamental as Theorem 6 and Theorem 11 are so difficult to prove, because I cannot prove them in a way that makes me happy.]

— Leonhard Euler, letter to Christian Goldbach, November 14, 1750.
(Theorem 6 is the angle-defect formula; Theorem 11 is Euler's formula.)

Mirabilis. . . est haec inter angulos solidos et triangula lateralia conspiratio;
mihi certe eius notitia, vel sola, remunerari videbatur laborem in haec impensum.

[This harmony between solid angles and triangular sides is astonishing;
certainly for me, just observing it seems to repay the effort spent on them.]

— Albrecht Ludwig Friedrich Meister, "Commentatio de solidis geometricis. . . " (1785)

2.1 Graphs

Basic Definitions

A graph is an abstract combinatorial structure that models pairwise relationships. Graphs are traditionally defined as pairs (V, E), where V is an arbitrary finite set of so-called *vertices*, and E is a set of unordered pairs of vertices, called *edges*. While admirably terse, this definition is both unnecessarily restrictive and inconsistent with the usual

abstract graph
V
vertex!of a graph
D
darts!of a graph
rev
head
reversal
head
tail
endpoint
leave a vertex
enter a vertex
u to v
edge!of a graph
E
e+
e-
orientation
incident
neighbor
uv
loop!in a graph
parallel edges
simple graph
non-simple graph
degree of a vertex
deg G v
deg v
isolated
topological graph
GT
quotient space
To avoid excessive formality

data-structure representation of graphs. Instead, we formally define an ***abstract graph*** to be a quadruple $G := (V, D, rev, head)$, where

- V is a non-empty set of abstract objects called ***vertices***;
- D is a set of abstract objects called ***darts***;
- rev is a permutation of D such that $rev(rev(d)) = d \neq rev(d)$ for every dart $d \in D$;
- $head$ is a function from the darts D to the vertices V.

For any dart d, we call the dart $rev(d)$ the ***reversal*** of d, and we call the vertex $head(d)$ the ***head*** of d. The ***tail*** of a dart is the head of its reversal: $tail(d) := head(rev(d))$. The head and tail of a dart are its ***endpoints***. Intuitively, a dart is a directed path from its tail to its head; in keeping with this intuition, we say that a dart d ***leaves*** its tail and ***enters*** its head. We often write $u \to v$ to denote an dart with tail u and head v, even (at the risk of confusing the reader) when there is more than one such dart.

For any dart $d \in D$, the unordered pair $\{d, rev(d)\}$ is called an ***edge*** of the graph. We often write E to denote the set of edges of a graph, and we write e^+ and e^- to denote the constituent darts of an edge e. The endpoints of an edge $e = \{e^+, e^-\}$ are the endpoints (equivalently, just the tails) of its constituent darts. Intuitively, each dart is an ***orientation*** of some edge from one of its endpoints to the other. A vertex v and an edge e are ***incident*** if v is an endpoint of e; two vertices are ***neighbors*** if they are endpoints of the same edge. We often write uv to denote an edge with endpoints u and v, even (at the risk of confusing the reader) when there is more than one such edge.

A ***loop*** is an edge e with only one endpoint, that is, $head(e^+) = tail(e^+)$. Two edges are ***parallel*** if they have the same endpoints. A graph is ***simple*** if it has no loops or parallel edges and ***non-simple*** otherwise. (Non-simple graphs are sometimes called *generalized graphs* or *multigraphs*.) A simple graph is more tersely defined as a pair of sets (V, E), where each element of E *is* a set of two elements of V; our more complex definition is necessary because we do not want to assume *a priori* that all graphs are simple.

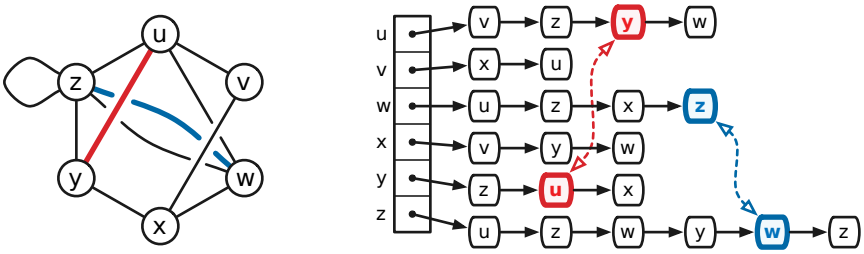
The ***degree*** of a vertex v, denoted $\mathbf{deg}_G(v)$ (or just $\mathbf{deg}(v)$ if the graph G is clear from context), is the number of darts whose head is v, or equivalently, the number of incident edges plus the number of incident loops. A vertex is ***isolated*** if it is not incident to any edge.

It is often convenient to regard graphs as continuous topological spaces, in which vertices are points and edges are interior-disjoint paths between their endpoints, rather than strictly combinatorial objects. More formally, let $V^\top$ be a set of distinct points $v^\top$, one for each vertex $v \in V$, and let $E^\top$ be a set of disjoint closed real intervals $e^\top$, one for each edge $e \in E$. The ***topological graph*** $G^\top$ is the quotient space $(V^\top \sqcup E^\top)/\sim$, where for each edge e, we have $a \sim head(e^+)^\top$ and $b \sim tail(e^+)^\top$, where $e^\top = [a, b]$. To avoid excessive formality, however, we rarely distinguish between an abstract graph G and the corresponding topological graph $G^\top$.

Data Structures

incidence list
adjacency list!see incidence list
linked list
balanced binary search tree
hash table
deleting an edge
G without e
G without v
contracting an edge
G mod e

In implementations of graph algorithms, graphs are normally represented using a data structure called an ***incidence list***; for simple graphs, the same data structure is more commonly known as an ***adjacency list***. A standard incidence list is an array of linked lists, indexed by the vertices, where each record in each linked list corresponds to one of the darts entering the corresponding vertex.[1] The record for each dart d contains the index of $tail(d)$, a pointer to the record for $rev(d)$, and a constant amount of other algorithm-dependent auxiliary data. Auxiliary data associated with the vertices is stored in the main array; auxiliary data associated with the *edges*, if any, is stored in the dart records. Storing a graph with n vertices and m edges in an incidence list requires $O(n+m)$ space altogether.



An incidence list representation of a graph, with the dart records for two edges emphasized. For clarity, most reversal pointers are omitted.

If a graph is stored in an incidence list, we can insert a new edge in $O(1)$ time, delete an edge in $O(1)$ time (given a pointer to one of its darts), and visit all the edges incident to any vertex v in $O(1)$ time per edge, or $O(\deg(v))$ time altogether. There are several standard operations that incidence lists do *not* support on $O(1)$ time, the most glaring of which is testing whether two vertices are neighbors. Surprisingly, however, most efficient graph algorithms do not require this operation. For those few that do, we can store the darts entering each vertex in a more efficient data structure, such as a balanced binary search tree or a hash table, instead of a linked list.

Deletion, Contraction, Subgraphs, and Minors

Let G be a graph with n vertices and m edges. ***Deleting*** an edge e from G yields a smaller graph $\boldsymbol{G \setminus e}$ with n vertices and $m-1$ edges. More generally, deleting any subset of edges $F \subseteq E$ yields a smaller graph $G \setminus F$. We also write $\boldsymbol{G \setminus v}$ to denote the graph obtained from G by deleting a vertex v and all its incident edges.

If e is not a loop, then ***contracting*** e merges the endpoints of e into a single vertex and destroys the edge, yielding a smaller graph $\boldsymbol{G / e}$ with $n-1$ vertices and $m-1$ edges. Contracting a loop is simply forbidden. More generally, contracting any subset $F \subseteq E$ of non-loop edges yields a smaller graph G / F.

subgraph
subgraph!proper
minor of a graph
minor!proper
vertex-disjoint
edge-disjoint
walk
tail!of an walk
head!of an walk
closed walk
open walk
length of a walk
simple walk
path!in a graph
cycle!in a graph
even subgraph
connected!graph
component!of a graph
acyclic graph
forest
tree
cut!in a graph
crossing a cut
boundary!of a cut
boundary S
edge cut
bond
bridge
spanning tree
exercise for the reader

A ***subgraph*** of a graph G is another graph obtained from G by deleting edges and vertices; a ***proper subgraph*** of G is any subgraph other than G itself. Similarly, a ***minor*** of G is any graph obtained from a subgraph of G by contracting edges; a ***proper minor*** of G is any minor other than G itself. Two or more subgraphs of G are ***vertex-disjoint*** if they have no vertices in common, and ***edge-disjoint*** if they have no edges in common.

Walks, Paths, Cycles, Cuts, Bonds, and Trees

A ***walk*** in a graph G is an alternating sequence $\omega = \langle v_0, d_1, v_1, d_2, \ldots, d_k, v_k \rangle$ of vertices and darts of G, where for every index i, we have $v_{i-1} = \mathit{tail}(d_i)$ and $v_i = \mathit{head}(d_i)$. The initial vertex v_0 and the final vertex v_k are respectively the ***tail*** and ***head*** of the walk; informally, we say that ω is a walk from v_0 to v_k. A walk is ***closed*** if it has at least one dart and its first and last vertices coincide; otherwise, the walk is ***open***. The ***length*** of a walk is the number of darts. If the graph G has no loops, we can safely regard any walk as an alternating sequence of vertices and *edges*; on the other hand, when a walk traverses a loop, the choice of dart specifies the direction of traversal. At the risk of ambiguity, we sometimes use the more mnemonic notation $v_0 \to v_1 \to \cdots \to v_k$ to describe a walk.

A walk is ***simple*** if all its vertices are distinct, except possibly the first and last. A ***path*** is a simple open walk; and a ***cycle*** is a simple closed walk. For example, a loop is a cycle of length 1, and a single vertex is a path (but *not* a cycle!) of length 0. We can safely regard paths and cycles as subgraphs rather than walks.

An ***even subgraph*** of a graph G is a subgraph in which every vertex has positive, even degree. Every even subgraph is the union of one or more edge-disjoint cycles [25].

A graph is ***connected*** if it contains a path from any vertex to any other. A ***component*** of a graph is a maximally connected subgraph. A graph is ***acyclic*** if the graph does not contain a cycle; acyclic graphs are also called ***forests***. Finally, a ***tree*** is any graph that is both connected and acyclic; thus, any forest is the disjoint union of trees.

A ***cut*** G is a partition of the vertices V of G into two non-empty subsets S and $V \setminus S$. An edge ***crosses*** the cut $(S, V \setminus S)$ if it has one endpoint in S and the other in $V \setminus S$. The set of all edges that cross the cut is called ***boundary*** of the cut and denoted ∂S. An ***edge cut*** is the boundary of some cut. An edge cut C is called a ***bond*** if no proper subset of C is also a edge cut; thus, a graph is connected if and only if it has a non-empty bond. Every non-empty edge cut is the union of one or more edge-disjoint bonds. For any connected graph G, an edge cut C is a bond if and only if the subgraph $G \setminus C$ has exactly two components. A ***bridge*** is an edge cut consisting of a single edge.

Spanning Trees

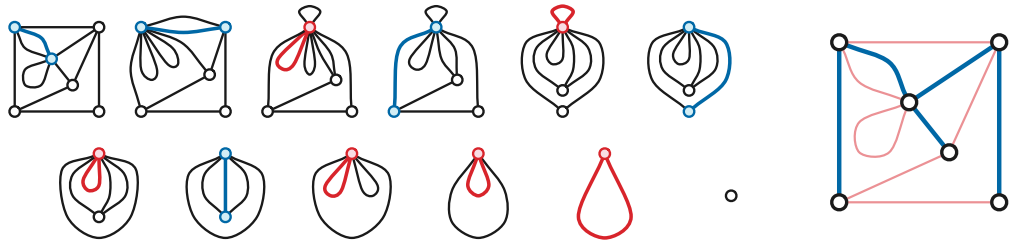
A ***spanning tree*** of G is a connected, acyclic subgraph of G that includes every vertex of G. We leave the following lemma as an exercise for the reader.

Lemma 2.1. *Let G be a connected graph, and let e be an edge of G.*

depth-first search
breadth-first search

(a) *If e is a loop, then every spanning tree of G excludes e.*
(b) *If e is not a loop, then for any spanning tree U of G/e, the subgraph $U \cup e$ is a spanning tree of G.*
(c) *If e is a bridge, then every spanning tree of G includes e.*
(d) *if e is not a bridge, then every spanning tree of $G \setminus e$ is also a spanning tree of G.*

This lemma immediately suggests the following general strategy to compute a spanning tree of any connected graph: For each edge e, either contract e or delete e. Loops must be deleted and bridges must be contracted; otherwise, the decision to contract or delete is arbitrary. Lemma 2.1 inductively implies that the set of contracted edges is a spanning tree of G, regardless of the order that edges are visited, or which non-loop non-bridge edges are deleted or contracted.



Computing a spanning tree of a graph.

In practice, most algorithms that compute spanning trees do not actually contract or delete edges; rather, they simply label the edges as belonging to the spanning tree or not. In this context, Lemma 2.1 can be rewritten as follows:

Lemma 2.2. *Let G be a connected graph.*
(a) *Every spanning tree of G excludes at least one edge from every cycle in G.*
(b) *For every edge e of every cycle of G, there is a spanning tree of G that excludes e.*
(c) *Every spanning tree of G includes at least one edge from every bond in G.*
(d) *For every edge e of every bond of G, there is a spanning tree of G that includes e.*

Corollary 2.3. *If the edges of a connected graph G are arbitrarily colored red or blue, so that each cycle in G has at least one red edge and each bond in G has at least one blue edge, then the subgraph of blue edges is a spanning tree of G.*

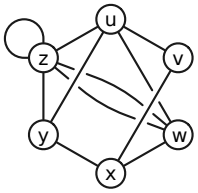
Given a connected graph with n vertices and m edges, we can compute a spanning tree in $O(n+m)$ time using either depth-first search or breadth-first search; both algorithms can be seen as variants of the red-blue coloring algorithm, where the order in which edges are colored is determined on the fly. Similarly, given a disconnected graph, we can compute a spanning tree for each component in $O(n+m)$ time; this is the most efficient method to determine the number of connected components in a graph.

planar embedding
planar graph
plane graph
stereographic projection
face!of a planar embedding
outer face

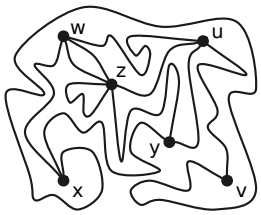
2.2 Planar Graphs

Embeddings

A ***planar embedding*** of a graph G is a continuous injective function from the topological graph $G^\top$ to the plane. More explicitly, a planar embedding maps the vertices of G to distinct points in the plane and maps the edges of G to simple paths in the plane between the images of their endpoints, such that the paths do not intersect except at common endpoints. A ***planar graph*** is an abstract graph that has at least one planar embedding. Somewhat confusingly, a planar embedding of a planar graph is also called a ***plane graph***.

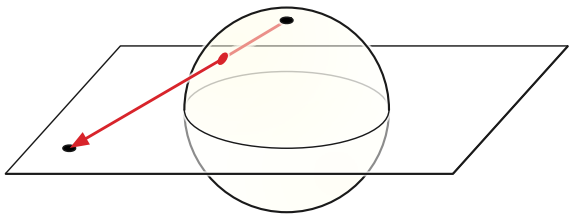


A planar graph G.



A planar embedding of G.

For many proofs, it is actually more natural to consider graph embeddings on the sphere $S^2 = \{(x,y,z) \mid x^2+y^2+z^2 = 1\}$ instead of the plane. Consider the standard ***stereographic projection*** map $st\colon S^2 \setminus (0,0,1) \to \mathbb{R}^2$, where $st(x,y,z) := \left(\frac{x}{1-z}, \frac{y}{1-z}\right)$. The projection $st(p)$ of any point $p \in S^2 \setminus (0,0,1)$ is the intersection of the line through p and the "north pole" $(0,0,1)$ with the xy-plane. Given any spherical embedding, if we rotate the sphere so that the embedding avoids $(0,0,1)$, stereographic projection gives us a planar embedding; conversely, given any planar embedding, inverse stereographic projection immediately gives us a spherical embedding. Thus, a graph is planar if and only if it has an embedding on the sphere.



Stereographic projection.

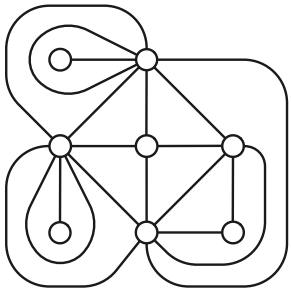
The components of the complement of the image of a planar or spherical embedding are called the ***faces*** of the embedding. The Jordan curve theorem implies that every face of a spherical embedding of a *connected* graph is homeomorphic to an open disk. A planar embedding of a connected graph has a single unbounded ***outer face***, which is

homeomorphic to the complement of a closed disk. For disconnected graphs, some faces are homeomorphic to an open disk with a finite number of open disks removed.

The faces on either side of an edge of a planar embedding are called the ***shores*** of that edge. For any dart d, the face just to the left of the image of d in the embedding is called the ***left shore*** of d, denoted $\mathbf{left}(d)$; symmetrically, the face just to the right is the ***right shore*** of d, denoted $\mathbf{right}(d)$. The same face may be both the left shore and right shore of a dart. We say that an edge e and a face f are ***incident*** if f is one of the shores of e; similarly, an vertex v and a face f are incident if v and f have a common incident edge. The ***degree*** of a face f, denoted $\mathbf{deg}_G(f)$ (or just $\mathbf{deg}(f)$ if G is clear from context), is the number of darts whose right shore is f.

shore of an edge
left shore
left e
right shore
right e
incident
degree of a face
deg G f
deg f
planar map
triangulation!planar map
vertex!of an embedded graph
edge!of an embedded graph
rotation system

Let F be the set of faces of a planar embedding of a connected graph with vertices V and edges E. We refer to the triple (V, E, F) as a ***planar map***. Similarly, a spherical map consists of the vertices, edges, and faces of an embedding of a connected graph onto the sphere. Trapezoidal decompositions and triangulations of polygons are both examples of planar maps. A planar or spherical map is called a ***triangulation*** if every face (including, for planar maps, the outer face) has degree 3. The underlying graph of a triangulation is *not* necessarily simple.



A planar triangulation.

At the risk of further confusing the reader, but following standard practice, we often use the same symbol G to simultaneously denote an abstract planar graph G, the corresponding topological graph $G^\top$, the image of a planar or spherical embedding of G (which, by definition, is homeomorphic to $G^\top$), and the resulting planar map (V, E, F). In particular, a ***vertex*** of an embedding of G is the point associated with a vertex of G, and an ***edge*** of the embedding is the path associated with an edge of G.

Rotation Systems

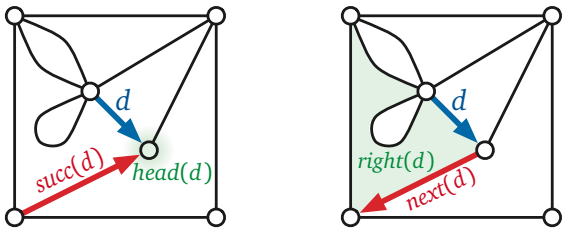
As usual in topology, we are not really interested in particular embeddings, but rather equivalence classes of embeddings, where two embeddings are equivalent if one can be continuously deformed into the other. Fortunately, every equivalence class of embeddings has a concrete combinatorial representation, called a ***rotation system***.

permutation
orbit
cyclic permutation
successor!in a rotation system
succ
counterclockwise
dual successor
succ star
clockwise
planar rotation system

Recall that a ***permutation*** of a finite set X is a bijection $\pi : X \rightarrowtail X$. For any permutation π and any element $x \in X$, let $\pi^0(x) := x$ and $\pi^k(x) := \pi(\pi^{k-1}(x))$ for any integer $k > 0$. The ***orbit*** of an element x is the set $\{\pi^k(x) \mid k \in \mathbb{N}\} = \{x, \pi(x), \pi^2(x), \ldots\}$. The restriction of π to any of its orbits is a ***cyclic*** permutation; the infinite sequence $x, \pi(x), \pi^2(x), \ldots$ repeatedly cycles through the elements of the orbit of x. Thus, the orbits of any two elements of X are either identical or disjoint.

The rotation system for a graph embedding is a permutation of its darts, called the ***successor*** permutation. The successor ***succ(d)*** of any dart d is the next dart entering $head(d)$ in counterclockwise order after d.[2]

The faces of any connected graph embedding are also encoded in its rotation system. Recall that *rev* is the reversal permutation of the darts of a graph. For any dart d, the ***dual successor*** ***next(d)*** $:= rev(succ(d))$ is the next dart after d in clockwise order around the boundary of $right(d)$.



The successor and dual successor of a dart in an embedded planar graph.

For disconnected graphs, it is impossible to determine from a rotation system which boundary components belong to the same face of an embedding, or even whether the components of the graph are embedded on the same sphere or on different spheres; this information must be recorded separately. (Many authors prefer to *define* the faces of any embedding to be the orbits of the permutation *next*, even when the graph is disconnected; see, for example, Klein and Mozes [52]. This is equivalent to declaring that each component of a disconnected graph is embedded on its own surface.) On the other hand, since almost all graph algorithms consider each component of the input graph independently, this extra information is rarely actually used. Thus, from now on ***we implicitly assume that all embedded graphs are connected***, unless explicitly stated otherwise. With this assumption in place, the orbits of *next* correspond precisely to the faces of the embedding.

We call a rotation system ***planar*** if it describes a planar (or spherical) embedding. Every rotation system corresponds to a embedding of a graph on some orientable surface, but not necessarily the sphere or the plane. In the next section, we describe how to construct a planar embedding that is consistent with a given planar rotation system.

The Schoenflies theorem (and Alexander's isotopy theorem) implies that every planar embedding is homeomorphic (and therefore isotopic) to a piecewise-linear planar embedding. Thus, every planar/spherical embedding (even if the edges are pathological space-filling monsters) has a well-defined rotation system.

piecewise-linear embedding
straight-line embedding
equivalent graph embeddings

Rotation systems trace their origins to Hamilton's Icosian Calculus [40,41,42], which can be seen as a rotation system for the regular dodecahedron, and to early research by Kirkman [48,49,50] and Cayley [14] on enumerating convex polyhedra. A more complete account of the history of rotation systems appears in Chapter **??**.

2.3 Straight-Line Embeddings

So far we have not assumed that planar embeddings are in any way well-behaved; edges may be embedded as arbitrarily pathological paths. It is not hard to prove using a compactness argument that any planar graph has a ***piecewise-linear*** planar embedding, meaning a planar embedding in which every edge is embedded as a simple *polygonal* path.[3] (We will see a similar compactness argument in the next chapter.)

However, with the Jordan curve theorem in hand, we can actually prove a stronger result, namely that every *simple* planar graph has a ***straight-line embedding***, meaning an embedding in which every edge is embedded as a single line segment. This result was first proved (indirectly) by Steinitz [83], and later independently rediscovered by Wagner [90], Fáry [34], Stein [84], and Stojaković [85]. We say that two planar graph embeddings are ***equivalent*** if they have the same rotation system.

Theorem 2.4. *Every planar embedding of a simple planar graph G is equivalent to a straight-line embedding of G.*

Proof: We roughly follow an argument of de Fraysseix, Pach, and Pollack [32,33].

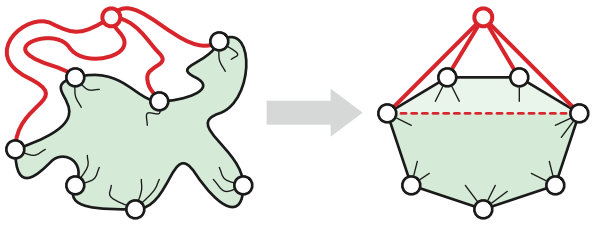
Fix a simple plane graph G with at least four vertices (since otherwise the theorem is trivial). If any face of G has degree greater than 3, it must contain a path between two non-adjacent vertices; adding this path as a new embedded edge yields a planar embedding of a larger simple plane graph. Thus, without loss of generality, we can assume that G is a simple triangulation.

Now we actually prove the following even still more strongerer result. Let H be a simple plane graph in which every *bounded* face has degree 3 and whose outer face is bounded by a simple cycle of length $h \geq 3$. Suppose the outer face of H has vertices $v_1, v_2, \ldots, v_h$ in counterclockwise order. Let P be an *arbitrary* convex polygon with h vertices $p_1, p_2, \ldots, p_h$ in counterclockwise order. We claim that there is a straight-line embedding of H, equivalent to the given embedding, such that P is the boundary of the outer face, and each vertex v_i is mapped to the corresponding point p_i.

If H is a single triangle, the claim is trivial, so assume otherwise. There are two other cases to consider.

Case 1. Suppose the only neighbors of v_h on the outer face are v_1 and v_{h-1}. In this case, we recursively compute an embedding of the subgraph $H' = H \setminus v_h$ and then embed the edges incident to v_h as line segments.

Specifically, let $w_1, w_2, \ldots, w_d$ be the neighbors of v_h, indexed in *clockwise* order around v_h so that $w_1 = v_{h-1}$ and $w_d = v_1$. The vertices $w_2, \ldots, w_{d-1}$ all lie in the



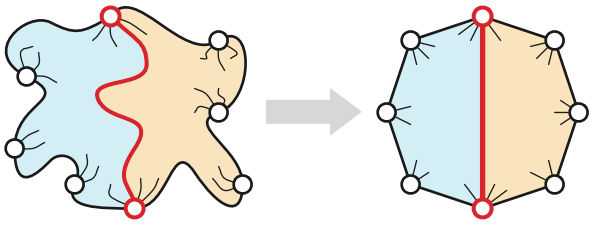
Case 1: Only two neighbors on the outer face; remove the vertex and recurse.

complement of the outer face, and H contains the edge $w_i w_{i+1}$ for every index i. It follows that the outer face of the subgraph $H' = H \setminus v_h$ is bounded by a simple cycle with vertices $v_1, v_2, \ldots, v_{h-1} = w_1, w_2, \ldots, w_d = v_1$. Every bounded face of H' is also a bounded face of H and thus has degree 3.

Now let α be a convex arc from p_1 to p_{h-1} inside the triangle $p_1 p_h p_{h-1}$. (For example, π could be a circular arc tangent to $p_1 p_h$ at p_1 and tangent to $p_h p_{h-1}$ at p_{h-1}.) Place d evenly spaced points $q_1, q_2, q_3, \ldots, q_d$ along α, with $q_1 = p_{h-1}$ and $q_d = p_1$. Finally, let P' be the convex polygon obtained by replacing the edges $p_{h-1}p_h$ and $p_h p_1$ with the polygonal chain $q_1 q_2 \ldots q_d$.

The inductive hypothesis implies that there is a straight-line embedding of H' whose outer face is P', that maps each vertex v_i (with $i \neq h$) to the corresponding point p_i and each vertex w_i to the corresponding point q_i. Mapping the vertex v_h to the point p_h and mapping each edge $v_h w_i$ to the line segment $p_h q_i$ gives us the required embedding of H.

Case 2. Now suppose v_h is adjacent to some vertex v_j on the outer face besides v_1 and v_{h-1}.[4] In this case, we split H into two subgraphs along the edge $v_h v_j$, split the polygon P into two smaller polygons along the diagonal $p_h p_j$, and recursively embed each subgraph of H into the corresponding fragment of P.



Case 2: A distant neighbor on the outer face; split along the diagonal and recurse.

Specifically, let $H^\sharp$ be the subgraph of H obtained by deleting every vertex outside the simple cycle $v_h \to v_1 \to v_2 \to \cdots \to v_j \to v_h$ in the given embedding. (The outside of this cycle is well-defined by the Jordan Curve Theorem.) Similarly, let $H^\flat$ be the subgraph of H obtained by deleting every vertex outside the cycle $v_h \to v_j \to v_{j-1} \to \cdots \to v_{h-1} \to v_h$. Both $H^\flat$ and $H^\sharp$ satisfy the conditions of our claim.

The line segment $p_h p_j$ partitions the polygon P into two smaller convex polygons $P^\flat$ and $P^\sharp$. The induction hypothesis implies that $H^\flat$ and $H^\sharp$ have straight-line embeddings,

equivalent to their given embeddings, with respective outer faces bounded by $P^\flat$ and $P^\sharp$, mapping each vertex v_i to the corresponding point p_i. In particular, both embeddings map the edge $v_h v_j$ to the line segment $p_h p_j$. Combining the straight-line embeddings of $H^\flat$ and $H^\sharp$ gives us the required straight-line embedding of H. □

exercise for the reader
normal labeling

The following corollaries are now immediate; we leave the omitted details as an exercise for the reader.

Corollary 2.5. *Every planar embedding of a planar graph is equivalent to a piecewise-linear embedding.*

Corollary 2.6. *(a) Every component of a planar graph is planar.*
(b) Every subgraph of a planar graph is planar.
(c) Every minor of a planar graph is planar.

Corollary 2.7. *Given a planar rotation system for a planar graph G with n vertices and m edges, we can compute an equivalent piecewise-linear embedding (or an equivalent straight-line embedding if G is simple) in $O(n+m)$ time.*

2.4 Schnyder Woods

In 1989, Walter Schnyder [78, 79] discovered a significant refinement of the previous proof, which implies that nay n-vertex planar graph has a straight-line embedding whose vertices lie on an $n \times n$ integer grid. As in the proof of Theorem 2.4, we consider only simple planar *triangulations*.

Let G be a simple planar triangulation. A vertex or edge of G is *internal* if it is not on the boundary of the outer face. A ***normal labeling*** of G assigns colors to the corners of each bounded face to satisfy two conditions.

- The corners of each face are colored red, green, and blue in counterclockwise order.
- The corners around each internal vertex are colored red, then green, then blue in counterclockwise order.

See Figure 2.1.

A normal labeling can be constructed in linear time as follows. Color the outer vertices red, green, and blue in counterclockwise order. If G has a single bounded face, its three corners inherit the colors of the incident vertices. Otherwise, choose an arbitrary edge from a boundary vertex u to an interior vertex v. There are two cases to consider.

- Suppose v has exactly two common neighbors with u. The contracted graph G/uv has two pairs of parallel edges; deleting one edge in each pair yields a smaller triangulation H. Recursively compute a normal labeling for H, and transfer the corner colors back to G. Finally, there is only one way to consistently color the

Schnyder wood
Schnyder wood

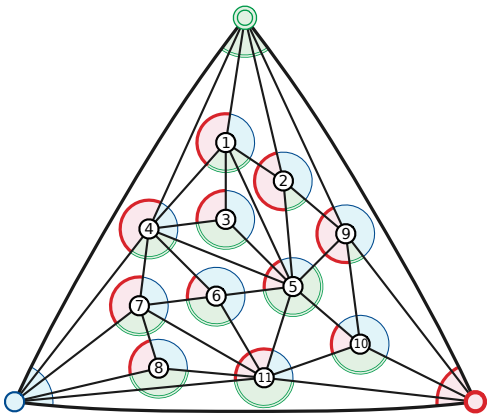


Figure 2.1. A normal labeling of a planar triangulation.

faces of G on either side of uv; specifically, the corners incident to the boundary vertex u inherit u's color.

- If v has at least three common neighbors with u, there must be a triangle uvw containing at least one vertex in its interior. In this case, we recursively compute normal labelings of the subgraphs of G inside uvw and outside uvw. In both recursive subproblems the boundary vertex u retains its originally assigned color, so that the resulting labelings are compatible.

In fact, the second case is unnecessary. By expanding the recursion from the second case, we see that the entire construction is carried out by a sequence of contractions; equivalently, an easy induction argument implies that at least one edge from each boundary vertex can be contracted immediately. The normal labeling in Figure 2.1 was computed by contacting the interior vertices toward the top (green) vertex in the order indicated by the vertex labels.

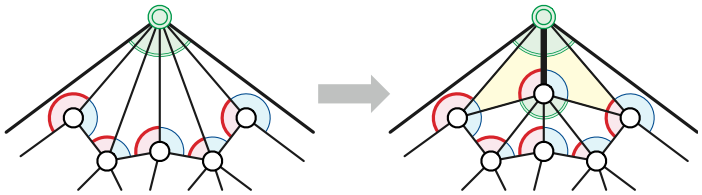


Figure 2.2. Recursively computing a normal labeling.

Schnyder defined a ***realizer*** as a coloring and orientation of the internal edges of the triangulation satisfying certain conditions; this structure is now commonly known as a ***Schnyder wood***. Every internal edge in G is incident to corners with all three colors, with one color appearing twice at one endpoint. To define the realizer, Schnyder orients each internal edge toward the endpoint with the same color twice, and assigns the

repeated color to the edge. The resulting coloring and orientation has the following useful properties:

- Each boundary vertex has only incoming internal edges, all with the same color as the vertex itself.
- Every internal vertex has exactly one outgoing edge of each color.
- At every internal vertex, incident edges appear in counterclockwise order as follows: one outgoing red, all incoming blue, one outgoing green, all incoming red, one outgoing blue, all incoming green.

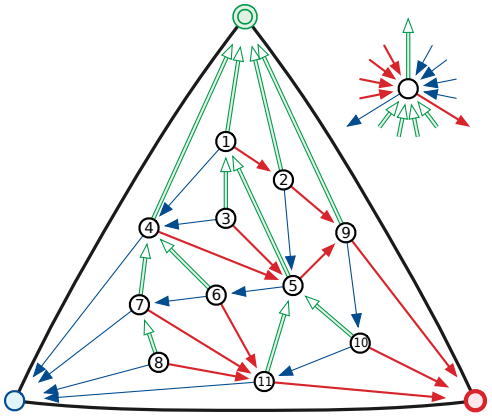


Figure 2.3. The Schnyder wood constructed from the normal labeling in Figure 2.1.

We can also construct Schnyder woods directly using a sequence of edge contractions, just as we constructed normal labelings. Expanding an edge introduces one vertex and three edges; we orient the new edges away from the new vertex, and assign them the only colors consistent with the properties listed above.

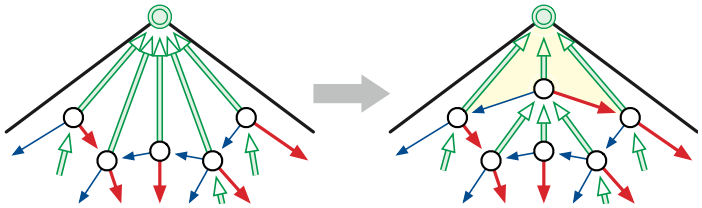


Figure 2.4. Recursively computing a Schnyder wood.

Lemma 2.8. *In any Schnyder wood, the edges of each color induce a spanning tree of the internal vertices, rooted at the boundary vertex with that color.*

Proof: Without loss of generality ⟨⟨**really?**⟩⟩, assume the Schnyder wood is constructed by repeatedly contracting to the green boundary vertex. Each interior vertex has exactly

one outgoing edge of each color, either leading to another interior vertex or the boundary vertex of that color.

Number the interior vertices by the order in which they are contracted to the green boundary vertex, as shown in the figures. Label the green boundary vertex 0 and the other two boundary vertices ∞. Call an edge $v \to w$ *increasing* if the label of w is larger than the label of w and *decreasing* otherwise. Every green edge is decreasing, and every red and blue edge is increasing. □

Now we assign integer coordinates to every interior vertex v as follows. There is a unique path of red edges, a unique path of green edges, and a unique path of blue edges rom v to the boundary. These three paths partition the triangulation into three regions, which we color red, green, and blue as shown in Figure 2.5. For example, the green region is bounded by the red and blue paths from v to the boundary. Each region contains its clockwise bounding path—for example, the green region includes the blue path, including the blue boundary vertex—but not v itself.

For purposes of defining these regions when v is a boundary vertex, we orient the boundary edges clockwise and color each edge according to its head. Thus, if r, g, b are the red, green, and blue boundary vertices, the green region of g contains every vertex except g and r, the red region of g contains only r, and the blue region of g is empty.

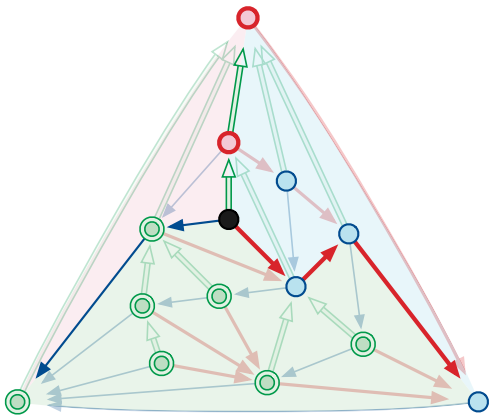


Figure 2.5. Schnyder regions for an interior vertex with coordinates $(2, 7, 4)$.

For any interior vertex, let $r(v)$, $g(v)$, $b(v)$ respectively denote the number of vertices in the red, green, and blue region of v. By definition, we have $r(v) + g(v) + b(v) = n - 1$ for every vertex v, where n is the total number of vertices in G.

Lemma 2.9. *(a) For every red edge $v \to w$, we have $r(v) < r(w)$ and $g(v) \geq g(w)$ and $b(v) > b(w)$.*

(b) For every green edge $v \to w$, we have $r(v) > r(w)$ and $g(v) < g(w)$ and $b(v) \geq b(w)$.

(c) For every blue edge $v \to w$, we have $r(v) \geq r(w)$ and $g(v) > g(w)$ and $b(v) < b(w)$.

(d) For every triangle uvw whose corners at u, v, and w are respectively colored red, green, and blue by the normal labeling, we have $r(u) \geq r(v) > r(w)$ and $g(v) \geq g(w) > g(u)$ and $b(w) \geq b(u) > b(v)$.

Proof: Color each vertex according to the region that contains it, and color the reference vertex that defines the regions black. When we move the reference vertex from v to w along a green edge $v \to w$, v changes from black to green, w changes from red to black, every green vertex remains green, and no vertex changes from red to blue or vice versa. (Some red or blue vertices might become green, and the set of blue vertices might remain unchanged.) Part (b) now follows immediately; parts (a) and (c) follow from symmetric arguments.

Finally, part (d) follows by simple case analysis. (Up to symmetry, there are only two cases to consider: either the edges of uvw have distinct colors, or two edges have the same color. □

Theorem 2.10. *Any planar embedding of a simple planar graph with n vertices is equivalent to a straight-line embedding with vertices on the $(n-1) \times (n-1)$ integer grid.*

Proof: Assign each vertex v the integer coordinates $(g(v), b(v))$. Let uvw be an arbitrary triangle whose corners at u, v, and w are respectively colored red, green, and blue by the normal labeling. The orientation of the new embedding of uvw is given by the sign of the determinant

$$\begin{vmatrix} 1 & g(u) & b(u) \\ 1 & g(v) & b(v) \\ 1 & g(w) & b(w) \end{vmatrix} = \big(g(v)-g(u)\big)\big(b(w)-b(u)\big) - \big(g(w)-g(u)\big)\big(b(v)-b(u)\big).$$

Lemma 2.9 immediately implies that this expression is positive, which implies that the triangle is oriented counterclockwise. Because every triangle is oriented correctly, no two triangles in the embedding can overlap, and therefore no pair of edges can intersect. (The signed area of the outer triangle is equal to the sum of the signed areas of the interior triangles; if two triangles overlapped, the sum of the *unsigned* areas of the interior triangles would be strictly larger than the unsigned area of the outer triangle [75]) All vertex coordinates are integers between 0 and $n-2$. □

Instead of projecting directly into the plane, we can use the triples $(r(v), g(v), b(v))$ directly as vertex coordinates, to compute an embedding in the plane $x + y + z = n - 1$. These coordinates embed the vertices in an equilateral triangular grid instead of the standard integer grid, but we can map back to the standard square grid with a simple linear transformation.

3-connected
convex embedding

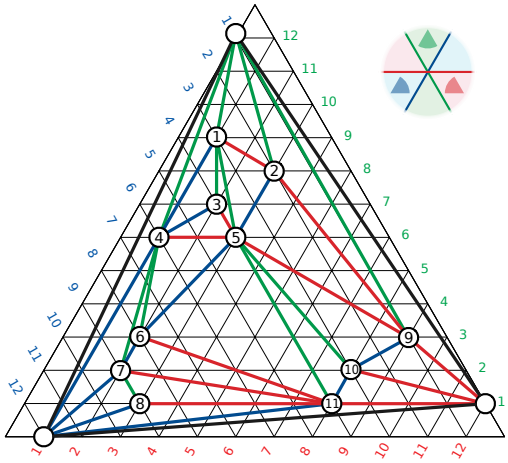


Figure 2.6. Schnyder's barycentric grid embedding of the example graph.

2.5 Further Extensions

Steinitz [83] proved a much more subtle and difficult generalization of the straight-line embedding theorem. A graph is ***3-connected*** if it remains connected after deleting any two vertices.

Theorem 2.11 (Steinitz [83]). *A simple planar graph is the graph of a 3-dimensional convex polytope if and only if it is 3-connected.*

In fact, Steinitz's theorem implies that every 3-connected simple planar graph has a ***convex*** embedding, meaning an embedding in which every bounded face is convex and the complement of the outer face is convex. This corollary was further strengthened, and given a more direct proof, by Tutte [87, 88].

Theorem 2.12 (Tutte [87, 88]). *Any planar embedding of a simple 3-connected planar graph is equivalent to a convex embedding, in which every vertex not on the outer face lies at the center of mass of its neighbors. Moreover, the outer face can be chosen to be the complement of any convex polygon with the correct number of vertices.*

Koebe-Andreev-Thurston

2.6 Duality

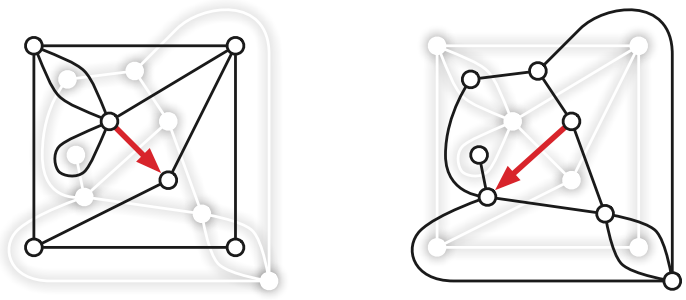
Definition

Fundamentally, a rotation system is just a pair $(succ, rev)$ of permutations of a finite set of abstract objects, such that every orbit of rev has exactly two elements. The objects

being permuted can be interpreted as the darts of an embedded graph G whose vertices are the orbits of *succ*, whose edges are the orbits of *rev*, and whose faces are the orbits of $succ \circ rev$.

dual!rotation system
G star
dual graph
counterclockwise
clockwise
one can verify mechanically

The ***dual*** of a rotation system $(succ, rev)$ is the rotation system $(rev \circ succ, rev)$. The embedded graph G^* determined by this dual rotation system is called the ***dual graph*** of G. That is, the vertices of G^* are orbits of $rev \circ succ$; the edges of G^* are the orbits of *rev*; and the faces of G^* are the orbits of *succ*. Thus, each vertex v, edge e, dart d, or face f of the original graph G corresponds to—or more evocatively, "is dual to"—a distinct face v^*, edge e^*, dart d^*, or vertex f^* of the dual graph G^*, respectively. The endpoints of any primal edge e are dual to the shores of the corresponding dual edge e^*, and vice versa. Specifically, for any dart d, we have $tail(d^*) = left(d)$ and $head(d^*) = right(d)$, and symmetrically, $left(d^*) = tail(d)$ and $right(d^*) = head(d)$.



A planar embedded graph and its dual. One dart and its dual are emphasized.

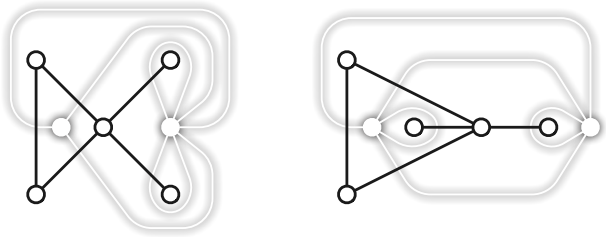
Attentive readers may have noticed that the rotation system of a graph encodes the *counterclockwise* order of darts leaving each vertex, while the dual rotation system encodes the *clockwise* order of darts around each face. Formally, to avoid this inconsistency, the graphs G and G^* use opposite orientations of the plane or sphere to distinguish left from right, and clockwise from counterclockwise. Intuitively, G and G^* are drawn on opposite sides of the plane or sphere.

We can also define the dual graph G^* directly in terms of a topological embedding of G, as follows. Choose an arbitrary point f^* in the interior of each face f of G. Let F^* denote the collection of all such points. For any edge e of G, choose a path e^* between the chosen points in the shores of e, such that e^* intersects e once transversely and does not intersect any other edge of G. Let E^* denote the collection of all such paths. Then the dual graph G^* is the topological graph with vertices F^* and edges E^*. By construction, G^* is embedded in the sphere; one can verify mechanically that each face of G^* contains exactly one vertex of G.

We must emphasize that duality is a correspondence between *embedded* graphs, not between abstract graphs. An abstract planar graph can have many non-isomorphic planar embeddings, each of which defines a different abstract dual graph. Moreover, the dual of a simple embedded graph is *not* necessary simple; any vertex of degree 2 in G

involution

gives rise to parallel edges in G^*, and any bridge in G is dual to a loop in G^*. This is why we don't want graphs to be simple by definition!



Two embeddings of a simple planar graph, with non-simple, non-isomorphic dual graphs.

Duality is an involution; the dual of G^* is the original graph G. This observation is a trivial consequence of the combinatorial definition (**Proof:** $rev \circ rev \circ succ = succ$ □), but with some care, it can also be proved directly from the topological formulation.

When the graph G is clear from context, we abuse notation by writing H^* to denote the subgraph of G^* containing the edges dual to the edges of a subgraph H of G.

Correspondences

The correspondence between primal and dual edges easily extends to larger structures within any connected planar graph G. For example, as we already observed, an edge e is a bridge in G if and only if the dual edge e^* is a loop in G^*. The following tables summarize some of these correspondences; we develop these further in the next several lemmas.

primal G	**dual G^***
vertex v	face v^*
dart d	dart d^*
edge e	edge e^*
face f	vertex f^*
$tail(d)$	$left(d^*)$
$head(d)$	$right(d^*)$
$succ$	$rev \circ succ$
clockwise	counterclockwise

primal G	**dual G^***
empty loop	spur
loop	bridge
cycle	bond
even subgraph	edge cut
spanning tree	complement of spanning tree
$G \setminus e$	G^* / e^*
G / e	$G^* \setminus e^*$
minor $G \setminus X / Y$	minor $G^* \setminus Y^* / X^*$

Correspondences between features of primal and dual planar maps

Lemma 2.13 (Contraction ⇋ deletion). *Fix a connected plane graph G. For any edge e of G that is not a loop, we have $(G / e)^* = G^* \setminus e^*$. Symmetrically, for any edge e of G that is not a bridge, we have $(G \setminus e)^* = G^* / e^*$.*

Proof: Let *succ* denote the rotation system of G, and let $next = succ \circ rev$ denote its dual rotation system. Pick an arbitrary edge e of G. There are two cases to consider.

First, suppose e is not a loop. Then $G \,/\, e$ is a connected plane graph that contains every dart in G except e^+ and e^-. Let $succ \,/\, e$ and $next \,/\, e$ denote the induced primal and dual rotation systems of $G \,/\, e$. Then for any dart d of $G \,/\, e$, we have

$$(succ \,/\, e)(d) = \begin{cases} succ(e^-) & \text{if } succ(d) = e^+, \\ succ(e^+) & \text{if } succ(d) = e^-, \\ succ(e) & \text{otherwise,} \end{cases}$$

and

$$(next \,/\, e)(d) = \begin{cases} next(e^+) & \text{if } next(d) = e^+, \\ next(e^-) & \text{if } next(d) = e^-, \\ next(e) & \text{otherwise.} \end{cases}$$

On the other hand, suppose e is not a bridge. Then $G \setminus e$ is a connected plane graph that contains every dart in G except e^+ and e^-. Let $succ \setminus e$ and $next \setminus e$ denote the induced primal and dual rotation systems of $G \setminus e$. Then for any dart d in $G \setminus e$, we have

$$(succ \,/\, e)(d) = \begin{cases} succ(e^+) & \text{if } succ(d) = e^+, \\ succ(e^-) & \text{if } succ(d) = e^-, \\ succ(e) & \text{otherwise,} \end{cases}$$

and

$$(next \setminus e)(d) = \begin{cases} next(e^-) & \text{if } next(d) = e^+, \\ next(e^+) & \text{if } next(d) = e^-, \\ next(e) & \text{otherwise.} \end{cases}$$

These two cases are obviously symmetric. By definition, the dual rotation system *next* is the rotation system of G^*. □

Lemma 2.14 (Even subgraph ⇋ edge cut). *Fix a connected plane graph G. A subgraph H is an even subgraph of G if and only if H^* is an edge cut in G^*.*

Proof: Let H be an even subgraph of G. Let $C_1, C_2, \ldots, C_k$ be edge-disjoint cycles in G whose union is H. Color each vertex of G^* *black* if it lies in the interior of an odd number of cycles C_i, and *white* otherwise. Any path in G^* from a white vertex to a black vertex must cross some edge in H, and therefore must contain some dual edge in H^*. We conclude that H^* is a cut in G^*.

On the other hand, let H^* be an edge cut in G^*. Let S^* be a subset of vertices of G^* such that $H^* = \partial S^*$. Color a face of G *black* if its dual vertex lies in S^* and *white* otherwise. The primal subgraph H contains precisely the edges of G with one white shore and one black shore. Every vertex in G is clearly incident to an even number of such edges. We conclude that H is an even subgraph of G^*. □

algebraic dual

Corollary 2.15 (Cycle ⇋ bond). *Fix a connected plane graph G. A subgraph H is a cycle in G if and only if H^* is a bond in G^*.*

Proof: A cycle is a minimal even subgraph, and a bond is a minimal edge cut. □

Corollary 2.15 was first proved by Whitney [92, 93]. Whitney actually proved the converse of this result as well; yielding the following result. An ***algebraic dual*** of an abstract graph G is another abstract graph $G^\star$ with the same set of edges, such that a subset of edges defines a cycle in G if and only if the same subset defines a bond in $G^\star$.

Theorem 2.16 (Whitney [92, 93]). *A connected abstract graph is planar if and only if it has an algebraic dual.*

Corollary 2.17 (Spanning tree ⇋ spanning cotree). *Fix a connected plane graph G. A subgraph T is a spanning tree of G if and only if $G^* \setminus T^*$ is a spanning tree of G^*.*

Proof: Let T be an arbitrary spanning tree of G, and let $C^* = G^* \setminus T^*$ be the complementary dual subgraph of T. Lemma 2.2 implies that every cycle of G excludes at least one edge in T, and every bond of G contains at least one edge in T. Thus, Corollary 2.15 implies that every bond of G^* contains at least one edge in C^*, and every cycle of G^* excludes at least one edge in C^*. We conclude from Lemma 2.2 that C^* is a spanning tree of G^*. □

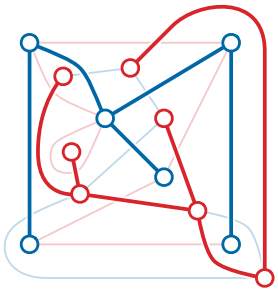


Figure 2.7. Interdigitating spanning trees of a planar map and its dual.

Self-Dual Data Structures

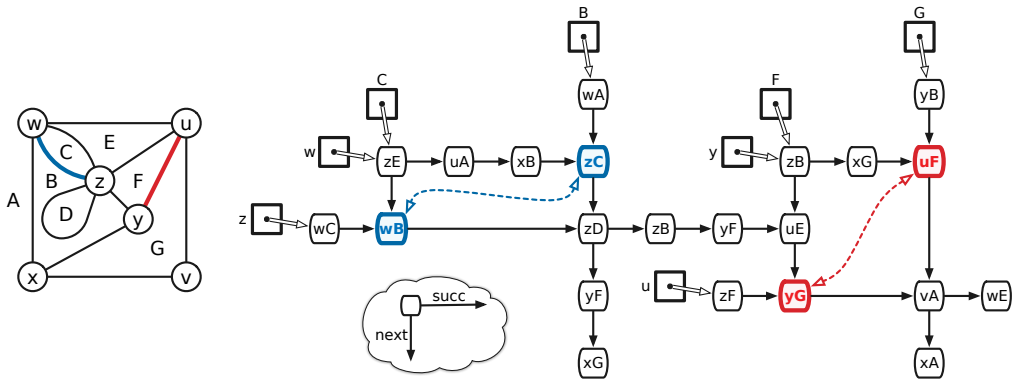
Recall that the incidence list data structure for any graph G stores a permutation of the darts entering each vertex of G, in the form of a linked list; thus, any incidence list actually represents a (not necessarily planar) rotation system for G. If we add a predecessor pointer to $succ^{-1}(d)$ to the record of every dart d, to complement the existing pointers to $succ(d)$ and $rev(d)$, the resulting data structure allows us to quickly navigate either the embedding of G or the induced dual embedding G^*. To emphasize

the importance of the order of the linked lists, we refer to this data structure as a ***sorted incidence list***.

sorted incidence list
self-dual incidence list
half-edge

Although the dual embedding G^* is implicitly encoded in the sorted incidence list of G, it is often more convenient to store the dual embedding explicitly. A ***self-dual incidence list*** of G is essentially an overlay of the sorted incidence lists of G and G^*. This data structure consists of two arrays, one indexed by vertices of G and the other by faces of G, and a set of dart records. The record for each dart d stores the index of the vertex $head(d)$, the index of the face $left(d)$, and pointers to five darts $rev(d)$, $succ(d)$, $succ^{-1}(d)$, $next(d)$, and $next^{-1}(d)$. For each vertex v, the corresponding entry in the vertex array points to an arbitrary dart d with $head(d) = v$; symmetrically, for each face f, the corresponding entry in the face array points to an arbitrary dart d with $right(d) = f$. Any self-dual incidence list of G is also a self-dual incidence list list of G^*, in which the *interpretations* of the *succ* and *next* pointers, the *head* and *left* pointers, and the vertex and face arrays are swapped; no actual modification of the data structure is required.

Double check figure for head-tail switch



A portion of a self-dual incidence list for a plane graph.
The dart records of two edges are emphasized.
The horizontal and vertical lists of darts are actually circular doubly-linked lists.

Sorted and self-dual incidence lists are two examples of a wide family of so-called ***half-edge*** data structures. Several variants appear in the literature, some omitting the vertex and face arrays, some storing the tails of darts instead of heads, some combining both dart records for each edge into a single edge record, some sorting only by the dual rotation system *next* instead of *succ*, some storing darts in arrays instead of linked lists, some storing vertices and faces in linked lists instead of arrays, and so on. These low-level design choices have no impact on the design and theoretical analysis of the algorithms that use the resulting data structures, and so we will largely ignore them. From a practical standpoint, however, these variants reflect significant tradeoffs between computation speed, compactness, ease of implementation, and ease of use.

winged-edge data structure
doubly-connected edge list
half-edge
split-edge
vertex-edge
face-edge
doubly-connected face list
doubly-connected edge list
blade
graph-encoded maps
gem
corner structure
quad-edge
generalized map
cell-tuple

One of the earliest examples of a half-edge data structure is the *winged-edge data structure* proposed by Baumgart in 1975 [3, 35], which keeps a single record for each edge, with pointers to both endpoints, both shores, and the four neighboring edges. In 1977, Preparata and Hong described an algorithm for computing three-dimensional convex hulls, represented by sorted incidence lists [72]; only a year later, Muller and Preparata describe this data structure as "one of the most commonly used representations for a planar graph" [66]. Other examples include an unnamed data structure of Danaraj and Klee [19], Muller and Preparata's *doubly-connected edge list* [66], Eastman's *half-edge* and *split-edge* structures [23, 67], Weiler's *vertex-edge* and *face-edge* structures [91], Chen's *doubly-connected face list* [16], and the *doubly-connected edge list* described by de Berg *et al.* [4] (which is slightly different from Muller and Preparata's structure of the same name).

Another family of data structures represent each edge by *four* records rather than two. Each record is associated with a ***blade*** [8]: an edge with a direction, specifying which endpoint is the tail, and an *independent* orientation, specifying which shore is on the left. Examples of blade-based data structures include the *graph-encoded maps* or *gem* representation proposed by Lins [59], Eastman's *corner structure* [23], Guibas and Stolfi's popular *quad-edge* data structure [39], Lienhardt's *generalized maps* [57, 58], and Brisson's *cell-tuple* structure [7]. Blade-based data structures offer some additional flexibility that half-edge data structures do not, which we will revisit in Chapter **??**.

2.7 Euler's Formula

Arguably the earliest fundamental result in combinatorial topology is a simple formula first published by Leonhard Euler. We provide three short proofs, one directly inductive, one relying on tree-cotree decompositions, and one relying on straight-line embeddings.

Euler's Formula for Planar Graphs. *For any connected plane graph G with n vertices, m edges, and f faces, we have $n - m + f = 2$.*

Proof (induction): If G has no edges, it has one vertex and one face. Otherwise, let e be any edge in G; there are two overlapping cases to consider.

- If e is not a loop, then contracting e yields a connected plane graph G / e with $n-1$ vertices, $m-1$ edges, and f faces. The induction hypothesis implies that $(n-1) - (m-1) + f = 2$.
- If e is not a bridge, then deleting e yields a connected plane graph $G \setminus e$ with n vertices, $m-1$ edges, and $f-1$ faces. The induction hypothesis implies that $n - (m-1) + (f-1) = 2$.

In all cases, we conclude that $n - m + f = 2$. □

Proof (von Staudt [81]): Let T be a spanning tree of G. Because T has n vertices, it also has $n-1$ edges. The complementary dual subgraph $C^* = (G \setminus T)^*$ is a spanning

tree of G^*. Because C^* has f vertices, it also has $f-1$ edges. Every edge in G is either an edge of T or the dual of an edge in C^*, but not both. Thus, $m = (n-1)+(f-1)$. □

exercises for the reader
maximally planar

Proof (l'Huiller [55, 56]): It suffices to prove the theorem for simple triangulations. If G is not a simple graph, we can make it simple by splitting each edge into a path of three edges, by introducing two new vertices; the resulting simple graph has $n+2m$ vertices, $3m$ edges, and f faces. Similarly, if any face of G has degree greater than 3, it must contain a path between two non-adjacent vertices; adding this path to the embedding yields a new plane graph with n vertices, $m+1$ edges, and $f+1$ faces. Finally, Theorem 2.4 implies that we only need to consider straight-line embeddings.

So let G be a simple straight-line plane triangulation with n vertices, m edges, and f faces (including the outer face). We compute the sum of the interior angles around the vertices of G in two different ways. First, each face is a triangle, so the sum of all angles is πf. Second, the angles around each interior vertex sum to 2π, and the angles around the three boundary vertices also sum to 2π (specifically, π from the bounded faces plus π from the inverted outer face). Thus, $\pi f = 2\pi(n-3)+2\pi$, which implies that $f = 2n-4$. Finally, because every face is a triangle, we have $2m = 3f = 6n-12$. We conclude that $n-m+f = n-(3n-6)+(2n-4) = 2$. □

Proof (via Schnyder woods): As in the previous proof, it suffices to prove the theorem for simple triangulations. Consider any Schnyder wood of an n-vertex simple triangulation G with n vertices, m edges, and f faces (including the outer face). The edges of each color define a rooted tree with $n-2$ vertices, and therefore $n-3$ edges, and every interior edges belongs to exactly one such tree. It immediately follows that $m = 3(n-3)+3 = 3n-6$. Finally, because every face is a triangle, we have $2m = 3f = 6n-12$. We conclude that $n-m+f = n-(3n-6)+(2n-4) = 2$. □

Euler's formula has several straightforward but useful consequences, whose proofs we leave as exercises for the reader. A simple planar graph G is ***maximally planar*** if inserting a new edge between any two distinct non-adjacent vertices makes the graph non-planar.

Corollary 2.18. *Every simple planar graph G with $n \geq 3$ vertices has at most $3n-6$ edges and at most $2n-4$ faces, with equality if some embedding of G is a triangulation.*

Corollary 2.19. *A simple planar graph G is maximally planar if and only if **every** planar embedding of G is a triangulation.*

Corollary 2.20. *Every simple planar graph has a vertex with degree less than 6.*

Some Muddled History

Like the Jordan curve theorem, the early history of Euler's formula is complex and confusing. Literally dozens of proofs were published in the 19th century alone; the

shelling

precise conditions of the formula are themselves a subject of intense debate[5]; most of the early proofs were either incorrect or incomplete; even when correct, many early authors overstated the generality of their results; and finally, a complete *formal* proof requires the Jordan curve theorem! Here I briefly sketch only a few important early landmarks, using modern terminology and notation. Brückner [9] provides a much more thorough (but still incomplete) survey of the literature up to about 1900; see also the more recent survey by Eppstein [24].

Early statements of Euler's formula did not consider arbitrary planar graphs, but only *convex polyhedra*. The first incarnation of the formula appears an unpublished manuscript of Descartes [21,22], written more than a century before Euler's rediscovery. Descartes observed without proof that the sum of all the plane angles in any convex polyhedron is $2\pi(n-2)$; starting from this observation, Descartes proved that $f = 2n-4$ if every face is a triangle and that the number of plane angles is always $2f + 2n - 4$.[6]

Euler first described both his formula $n + f = m + 2$ and the angle-sum formula $\Sigma = 2\pi(n-2)$ in a letter to Goldbach in 1750 [26]. Twelve days later, he presented both formulas to the St. Petersburg Academy of Sciences [28]; he did not prove his formula, but he did derive the angle-sum formula from it. Two years later, Euler proposed an inductive proof [27]. Specifically, Euler proposed a method for removing an arbitrary vertex and its incident faces, and then patching the resulting hole with new triangles, to obtain a new polyhedron with $n-1$ vertices, $m-3$ edges, and $f-2$ faces; the polyhedral formula immediately follows by induction. However, Euler's vertex-deletion algorithm sometimes yields a non-convex "polyhedron" with disconnected interior, which eventually makes further progress impossible; consequently, Euler's proof is incorrect. In hindsight, Euler's argument is easy to fix; to retriangulate the hole, it suffices to compute the convex hull of the undeleted vertices, and then arbitrarily triangulate any non-triangular faces.

Karsten included Euler's formula in his influential series of textbooks on mathematics and physics [47], after learning of the formula directly from Euler [77]. Karsten offered the following inductive proof. Decompose the polyhedron into pyramids, by joining each facet to a common interior point. By a direct counting argument, each individual pyramid satisfies Euler's formula. Now delete the pyramids one at a time until only one pyramid remains, and consider the number of vertices and facets of the remaining solid. Deleting a pyramid whose base has b edges and that shares c contiguous triangular faces with the remaining undeleted pyramids yields a polyhedron with $n-b+c+1$ vertices, $m-2b+3c$ edges, and $f+2c-b-1$ faces. Unfortunately, Karsten's argument is incomplete, because it requires that each deleted pyramid share a *connected* set of faces with the remaining solid; carelessly removing pyramids can easily violate this invariant. In more modern terminology, Karsten assumes that any plane graph has a *shelling*. Nearly identical proofs, with the same flaw, were later independently proposed by Meister [65] (for polyhedra with triangular facets) and L'Huillier [55,56] (in addition to the angle-based proof given above).[7]

In hindsight, Karsten's proof is easy to fix; a suitable deletion order can be de-

rived from any spanning cotree; the straight-line embedding algorithms described in Sections 2.3 and 2.4 also yield shelling orders.

fungus

The first complete proof of Euler's polyhedral formula was given by Legendre [54]. Legendre projects the vertices and edges of the polyhedron onto the unit sphere from an arbitrary interior point, and then applies the already well-known fact that a spherical triangle with interior angles α, β, and γ has area $\alpha+\beta+\gamma-\pi$. Suppose the original polyhedron has n vertices and f facets, all triangles. The angles at each vertex of the resulting spherical triangulation sum to exactly 2π; thus, the total area of all f spherical triangles is $2\pi n-\pi f$. We immediately conclude that $f=2n-4$, because the surface area of the unit sphere is 4π. The proof for more general polyhedra follows by triangulating the faces. Essentially the same proof was later given by Hirsch [43].

Cauchy [13] is usually cited as the author of the first purely *combinatorial* proof of Euler's formula. However, all three of Cauchy's proofs closely follow the inductive shelling strategy published by Karsten almost 50 years earlier, and suffer from precisely the same flaw. In short, Cauchy's proof is neither Cauchy's nor a proof—it is a fungus![8] In two of his proofs (one with triangular faces, the other for arbitrary faces), Cauchy removes a single face of the polyhedron, projects the remaining faces into the plane, and then recursively removes faces from the resulting planar map. Cauchy's third proof recursively removes tetrahedra from a decomposition of the polyhedron, obtained by joining a vertex to all non-incident faces. Cauchy's planar arguments were slightly simplified, but not repaired, by Grunert [37]. The first *correct* combinatorial proof appears to be von Staudt's tree-cotree proof [81]. Again, in hindsight, von Staudt's proof can be interpreted as a formalization of the Karsten-Meister-L'Huillier-Cauchy-Grunert inductive shelling argument; indeed, Brückner [9] incorrectly attributes the tree-cotree proof to Cauchy.

The first proofs that explicitly consider arbitrary plane graphs are due to Cayley [15] and Listing [60]. In fact, both Cayley and Listing allowed their graphs to include isolated closed "edges" with no vertices, and Listing considered much more general "acyclodic spatial complexes" constructed by gluing disks to cycles in graphs. Cayley's argument is a prototype for our first inductive proof; he observed that the quantity $n-m+f$ does not change when one inserts a new vertex in the interior of an edge or inserts a new edge in the interior of a face. Listing repeats (and further generalizes) Cauchy's proof, using a global counting argument instead of induction, but again assuming without proof the existence of a shelling. Both of these proofs implicitly assume the Jordan curve theorem, and therefore are technically incomplete (shelling issues aside).

Jordan [45] neatly sidestepped the Jordan curve theorem by *defining* a surface map (or "polyhedron") to be "Eulerian" if every simple cycle separates its surface into two components. Jordan's proof that all "Eulerian" polyhedra satisfy Euler's formula can be applied without modification to arbitrary planar maps. Jordan also avoided the shelling issue by using a divide-and-conquer strategy, similar to the second case of our proof of Theorem 2.4, instead of removing faces one by one.

1

minimum spanning tree
n
m

2.8 Minimum Spanning Trees

In many applications of graphs, there is a numerical *weight* associated with each edge of the graph. For example, the graph might model a road network, where each vertex represents a city, each edge represents a road, and the weight of an edge is the length of the corresponding road. Or the graph might represent an digital image, where each vertex is a pixel, edges join adjacent pixels, and the weight of each edge reflects the similarity between the corresponding pixels.

A common task in many graph algorithms is finding a ***minimum spanning tree*** of an edge-weighted graph G; this is a spanning tree of G whose total weight is no bigger than total weight of any other spanning tree of G. Minimum spanning trees were originally studied as a model of efficient communication networks [5, 6], but have since proved useful in many other contexts, including more complex network design and optimization problems, clustering, image processing, preconditioning linear systems, and computing approximate solutions of several NP-hard problems. Graham and Hell [36] give a detailed early history of the minimum spanning tree problem, tracing several classical algorithms to their (multiple) original sources. The more recent survey Mareš [63] gives a thorough overview of the state of the art. Finding minimum spanning trees is relatively straightforward in *arbitrary* graphs, but Euler's formula implies even simpler and faster algorithms for planar graphs.

Following standard practice, we report running times of graph algorithms as functions of two variables n and m, which respectively denote the number of vertices and the number of edges of the input graph. If the input graph is simple and planar, Euler's formula implies $m = O(n)$; thus, we report running times for simple planar graphs only as a function of n. On the other hand, if the input graph is connected, then $m \geq n - 1$.

To simplify exposition, we implicitly assume that all edge weights are distinct; this assumption implies that the minimum spanning tree is unique. Our assumption can be enforced if necessary by the following simple tie-breaking rule. Arbitrarily index the edges from 1 to m; if two edges have the same weight, proceed as though the edge with smaller index has smaller weight.

Tarjan's Red-Blue Meta-Algorithm

Tarjan [86] observed that several classical minimum spanning tree algorithms can be described by a single general strategy. Color each edge of G blue if it is the lightest edge in some bond, or red if it is the heaviest edge in some cycle.

Lemma 2.21 (Tarjan's "blue rule"). *Let e be any blue edge in any edge-weighted graph G, and let T be the minimum spanning tree of G / e. Then $T \cup e$ is the minimum spanning tree of G.*

Proof: Let G be any edge-weighted graph, let B be an arbitrary bond in G, and let e be the lightest edge in B. Let T be any spanning tree of G that *excludes* the blue edge e. The

spanning tree T contains a unique path between the endpoints of e; at least one edge e' in this path must lie in the bond B. The subgraph $T' = (T \cup e) \setminus e'$ is a spanning tree of G with smaller total weight than T, so T cannot be the minimum spanning tree of G.

We conclude that e is an edge in the minimum spanning tree; the lemma now follows from Lemma 2.2. □

We leave the symmetric proof of Tarjan's red rule as an exercise for the reader.

Lemma 2.22 (Tarjan's "red rule"). *Let e be any red edge in any edge-weighted graph G. The minimum spanning tree of G / e is also the minimum spanning tree of G.*

Corollary 2.23. *In any edge-weighted graph G, every edge is either red or blue, but not both, and the subgraph of blue edges is the minimum spanning tree of G.*

With these rules in place, Tarjan's general strategy is simple: If the input graph G has no edges, there is nothing to do; otherwise, either contract an arbitrary blue edge or delete an arbitrary red edge, and then recurse. The correctness of this strategy follows inductively from Tarjan's blue and red rules, regardless of which rule is applied at each recursive call, or to which bond or cycle the rule is applied. For the sake of efficiency, we may prefer to contract several blue edges at once, or delete several red edges at once, rather than one at a time.

Flattening

Call an edge of G ***redundant*** if it is a loop, or if it is *not* the lightest edge between its endpoints. Every redundant edge is the heaviest edge in a cycle of length 1 or 2; thus, redundant edges are in fact red. Thus, we can safely delete all redundant edges from a graph without changing its minimum spanning tree; we call this process ***flattening*** the graph.

Lemma 2.24. *Any edge-weighted graph can be flattened in $O(n+m)$ time.*

Proof: As usual, we assume that the graph is stored in an incidence list, with the weight of each edge stored in both of the dart records for that edge. To simplify the description of the algorithm, we treat each individual linked list as a *stack*, accessible only through the following operations:

- EMPTY?(S): Return TRUE if the stack is empty and FALSE otherwise.
- PUSH(S, d): Push a new dart d onto stack S
- POP(S): Remove and return the newest dart in stack S

Each of these operations can be performed in $O(1)$ time if the stack is represented as a standard linked list. We also assume that each dart stores both its head and its tail.

Our FLATTEN algorithm performs two passes over the edges. The first pass *transposes* the adjacency list data structure, transforming the lists of darts entering each vertex into

exercise for the reader
redundant edge
flattening a graph
stack

lists of darts leaving each vertex; the first pass also removes loops from the graph. An important side-effect of our transposition algorithm is that parallel darts are clustered together consecutively. The second pass transposes the graph again and removes parallel edges. Pseudocode for our FLATTEN algorithm appears below. The algorithm spends $O(1)$ time for each vertex and edge, so its overall running time is $O(n+m)$. □

```
FLATTEN(G):
  ⟨⟨Transpose and remove loops⟩⟩
  for i ← 1 to n
      while ¬EMPTY?(G[i])
          d ← POP(G[i])
          if head(d) ≠ tail(d)
              PUSH(H[tail(d)], d)
          else
              discard d

      ⟨⟨Transpose and remove parallel edges⟩⟩
      for i ← 1 to n
          d ← POP(H[i])
          while ¬EMPTY?(H[i])
              d' ← POP(H[i])
              if head(d) ≠ head(d')
                  PUSH(G[head(d)], d)
                  d ← d'
              else
                  weight(d) ← min{weight(d), weight(d')}
                  discard d'
          PUSH(G[head(d)], d)

      return G
```

Flattening a graph in $O(m+n)$ time.

Borůvka's Algorithm

The earliest algorithm to compute minimum spanning trees of arbitrary graphs was described by the Czech mathematician Otakar Borůvka in his 1926 PhD thesis [5, 6, 68], and later independently rediscovered several times [18, 30, 80]. Borůvka's algorithm has the following simple description: Simultaneously contract the lightest edge incident to every vertex, flatten the contracted graph, and recurse on the resulting minor. The recursion halts when the graph has no edges. The set of edges incident to a vertex is a bond, so the lightest such edge is blue. Thus, Borůvka's algorithm is an instance of Tarjan's general red-blue strategy, and therefore correctly computes the minimum spanning tree.

We can find the lightest edge incident to each vertex in $O(m)$ time by a brute-force traversal of the incidence list of G. We compute the contracted graph G / L in $O(m)$ time as follows. First, we compute a label $\ell(v)$ for every vertex v, such that any two vertices

```
BORŮVKA(G):
  if G has no edges
      return ∅
  L ← ∅
  for each vertex v of G
      add the lightest edge incident to v to L
  return L ∪ BORŮVKA(FLATTEN(G / L))
```

Borůvka's minimum spanning tree algorithm.

have the same label if and only if they lie in the same component of the subgraph L. These labels can be computed in $O(m)$ time by a depth- or breadth-first search of the subgraph L; the set of labels is also the vertex set of G / L. Then we copy each dart of G into a new incidence list for G / L, using the endpoint labels as vertices; that is, each dart $u \to v$ in G becomes a dart $\ell(u) \to \ell(v)$ in G / L with the same weight. In particular, each edge in L becomes a loop, which is deleted when the graph is flattened. Finally, FLATTENing G / L requires $O(m)$ time. Thus, ignoring the cost of the recursive call, Borůvka's algorithm runs in $O(m)$ time. Each round of contraction reduces the number of vertices by at least a factor of 2, so the algorithm ends after $O(\log n)$ recursive calls. Thus, for arbitrary graphs, the entire algorithm runs in $O(m \log n)$ time.

However, Cheriton and Tarjan [17] observed that Borůvka's algorithm actually runs in $O(m)$ time when the input graph is planar. After the first contraction round, Corollary 2.6(c) implies that the contracted graph G / L is planar, so Euler's formula implies that FLATTEN(G / L) has $O(n)$ edges. Thus, the second contraction round requires only $O(n)$ time. Moreover, because each contraction round removes at least half the vertices, the running times of successive rounds decrease geometrically. Thus, the total time for *all* rounds after the first is only $O(n)$.

Theorem 2.25. *Borůvka's algorithm computes the minimum spanning tree of any connected edge-weighted planar graph in $O(m)$ time.*

Mareš's Algorithm

The following "local" variant of Borůvka's algorithm, proposed by Mareš [62], avoids the difficulty of computing the contraction G / L by contracting edges one at a time and only contracting edges incident to low-degree vertices. Unlike Borůvka's original algorithm, Mareš's algorithm does not work for arbitrary graphs.

Any edge vw can be contracted in time $O(\deg(v))$ by moving all the darts from v's linked list into w's, and then marking v as deleted in the top-level array. Thus, each contraction in MAREŠ requires only $O(1)$ time, and because we cannot contract more than $n - 1$ edges, the entire while loop runs in $O(n)$ time. The following lemma implies that if the input graph is simple, then the while loop reduces the number of vertices by a constant factor.

```
MAREŠ(G):
  if G has no edges
      return ∅
  L ← ∅
  while G has at least two vertices, one of which has degree at most 6
      v ← any vertex of G with degree at most 6
      e ← lightest non-loop edge incident to v
      G ← G / e
      Add e to L
  return L ∪ MAREŠ(FLATTEN(G))
```

Mareš's minimum spanning tree algorithm for planar graphs.

Lemma 2.26. *Any simple planar graph with n vertices has at least $n/4$ vertices of degree at most 6.*

Proof: Without loss of generality, let G be a triangulation. For each vertex v, the number $\delta(v) := \deg(v) - 3$ is non-negative. Euler's formula implies that $\sum_v \delta(v) \le 3n - 12$. Thus, there are at most $3n/4 - 3$ vertices v such that $\delta(v) \ge 4$. □

Suppose the input graph G is simple. At the start of the algorithm, G has at least $n/4$ vertices with degree at most 6, and each contraction reduces the number of such vertices by at most 2. Thus, the while loop iterates at least $n/8$ times, leaving a graph with at most $7n/8$ vertices. We conclude that the running times for successive rounds decrease geometrically, and thus the total running time of Mareš's algorithm is $O(n)$.

Again, if G is simple but non-planar, the first round requires $O(m)$ time, and the rest of the algorithm takes only $O(n)$ time.

Algorithms for Planar Maps

Both Borůvka's algorithm and Mareš's incremental variant compute the minimum spanning tree of any simple *abstract* planar graph in $O(n)$ time; neither algorithm requires an embedding. However, if we are lucky enough to be given a planar embedding, there are even simpler linear-time algorithms.

The first algorithm is a further simplification of Mareš's incremental algorithm: repeatedly find a vertex with degree at most 5, contract the lightest edge leaving that vertex, and flatten the resulting graph. The key insight is that if the graph G is simple, we can flatten the contracted graph G / e in only $O(1)$ time. G / e has at most two pairs of parallel edges. Specifically, if $\mathit{right}(e^+)$ is a triangle, then the edges carrying $\mathit{next}(e^+)$ and $\mathit{next}^{-1}(e^+)$ are parallel in G / e; symmetrically, if $\mathit{right}(e^-)$ is a triangle, then the edges carrying $\mathit{next}(e^-)$ and $\mathit{next}^{-1}(e^-)$ are parallel in G / e. If the graph G is simple, these are the only possible parallel edges in G / e. Each parallel pair appears consecutively in the rotation system of G / e and thus are accessible in $O(1)$ time from the records associated with the contracted edge e.

```
MarešEmbedded(G):
  T ← ∅
  G ← Flatten(G)
  while G has edges
      v ← any vertex of G with degree at most 5
      e ← lightest edge incident to v
      G ← Flatten(G / e)
      Add e to T
  return T
```

A minimum spanning tree algorithm for embedded planar graphs.

Arguably an even easier approach is to choose a *random* vertex at each iteration. Euler's formula implies that the expected degree of a random vertex in a simple planar graph is less than 6, so the total expected running time of the resulting algorithm is still $O(n)$.

The second algorithm, proposed by Matsui [64], eliminates flattening altogether by exploiting duality. Corollary 2.17 implies that if T is the minimum spanning tree of a plane graph G, then the complementary dual subgraph $G^* \setminus T^*$ is the *maximum* spanning tree of G^*. Euler's formula implies that every planar map, simple or not, contains either a vertex degree at most 3 or a face of degree at most 3. Thus, we immediately obtain a simple "self-dual" algorithm: at each iteration, either contract the lightest edge incident to a low-degree vertex (unless that edge is a loop), or delete the heaviest edge incident to a low-degree face (unless that edge is a bridge).

```
MatsuiEmbedded(G):
  T ← ∅
  while G has more than one vertex or more than one face
      either
          v ← any vertex of G with degree at most 3
          if v is incident to a loop e
              G ← G \ e
          else
              e ← lightest edge incident to v
              G ← G / e
              add e to T
      or
          f ← any face with degree at most 3
          if f is incident to a bridge e
              G ← G / e
              add e to T
          else
              e ← heaviest edge incident to f
              G ← G \ e
  return T
```

A self-dual minimum spanning tree algorithm for embedded planar graphs.

bipartite
coloring of a graph
k-coloring of a graph
Four Color Theorem

2.9 Exercises

1. Collected "exercises for the reader":
   a) Prove Lemma 2.1.
   b) Prove Corollary 2.5.
   c) Prove Corollary 2.6.
   d) Prove Corollary 2.7.
   e) Prove Corollary 2.18.
   f) Prove Corollary 2.19.
   g) Prove Corollary 2.20.
   h) Prove Lemma 2.22.
2. Prove that every planar map has either a vertex with degree at most 3 or a face with degree at most 3.
3. A graph G is ***bipartite*** if its vertices can be partitioned into disjoint subsets L and R, such that every edge has one endpoint in L and one endpoint in R. Prove that every simple bipartite planar graph has at most $2n-4$ edges.
4. A ***coloring*** of a graph G is an assignment of colors (elements of some finite set) to the vertices of G such that the endpoints of each edge have distinct colors. A ***k-coloring*** is a coloring that uses at most k distinct colors. The infamous Four Color Theorem states that every planar graph has a 4-coloring.
   a) Prove that every loopless planar graph has a 6-coloring.
   b) Prove that every loopless planar graph has a 5-coloring.
5. A three-dimensional convex polyhedron is *regular* if all faces have the same degree and all vertices have the same degree. To rule out degenerate cases like line segments and two-sided regular polygons, we insist that all vertex and face degrees are at least 3. Prove that the regular tetrahedron, the cube, the regular octahedron, the regular dodecahedron, and the regular icosahedron are the only regular convex polyhedra.
6. Kirkpatrick's planar point-location data structure [51]:
   a) Prove that every simple planar graph with n vertices has an independent subset of at least $n/12$ vertices, each with degree less than 12.
   b) Describe an algorithm to compute such an independent set in $O(n)$ time.
   c) Let G be a simple straight-line plane triangulation with n vertices. Describe a method to preprocess G in $O(n)$ time, into a data structure of size $O(n)$, so that given an arbitrary point q in the plane, we can determine in $O(\log n)$ time which triangle of G (if any) contains q.
7. Let G be an arbitrary embedded planar graph, let T be an arbitrary spanning tree of G, and let e be an arbitrary edge of T. Color the vertices in one component of

$T \setminus e$ red and the vertices in the other component blue. Prove that any face of G is incident to either zero or two edges that have one red endpoint and one blue endpoint.

Red Herring Principle
homeomorphic
isotopic

Notes

1. (page 3) Readers bothered by the fact that an "incidence list" is not actually a *list* should remember Hirsch's Red Herring Principle [44]. In computer science, as in mathematics, a red herring is neither necessarily red nor necessarily a fish; conversely, a herring that happens to be red is not necessarily a red herring.

> *The ring worm is not ringed, nor is it worm. It is a fungus.*
> *The puff adder is not a puff, nor can it add. It is a snake.*
> *The funny bone is not funny, nor is it a bone. It is a nerve.*
> *The fishstick is not a fish, nor is it a stick. It is a fungus.*
>
> — Matt Groening, "Life in Hell" (1986)

2. (page 8) Because the edges of a planar embedding can be *arbitrary* continuous paths, it is not immediately obvious that every planar embedding has a well-defined rotation system. The existence of such a rotation system follows from the observation that every embedding is isotopic to a piecewise-linear embedding; see note 3.

3. (page 9) Fix an arbitrary planar embedding of an abstract planar graph G. We will locally modify the embedding, first in the neighborhoods of the vertices, and then in the neighborhoods of the edges, so that it becomes piecewise-linear.

First, let ε be the minimum distance between any two vertices of G. For each vertex v, let C_v be a circle of radius $\varepsilon/3$ centered at v. For each dart d, let π_d denote the corresponding path from $\mathit{tail}(d)$ to $\mathit{head}(d)$ in the embedding. For each dart d, let σ_d be a minimal subpath of π_d that starts on $C_{\mathit{tail}(d)}$ and ends on $C_{\mathit{head}(d)}$; we can choose these paths so that $\sigma_{\mathit{rev}(d)}$ is the reversal of σ_d. Finally, let τ_d be the path consisting of the line segment from $\mathit{tail}(d)$ to $\sigma_d(0)$, the subpath σ_d, and the line segment from $\sigma_d(1)$ to $\mathit{head}(d)$. By construction, the new paths τ_d are interior-disjoint, and thus define a new planar embedding of G.

Now let δ be the minimum distance between any two paths σ_d and $\sigma_{d'}$. Because each path σ_d is compact, σ_d can be covered by a finite number of balls of radius $\delta/3$, each centered at a point on σ_d. It follows that there is a finite sequence of points $x_0, x_1, \ldots, x_k$ on σ_d, such that x_0 and x_k are the endpoints of σ_d, and any two neighboring points x_i and x_{i+1} have distance at most $\delta/3$. The polygonal chain with vertices $\mathit{tail}(d), x_0, x_1, \ldots, x_k, \mathit{head}(d)$ may not be simple, but after removing a finite number of subloops, we obtain a simple polygonal chain ϖ_d from $\mathit{tail}(d)$ to $\mathit{head}(d)$. By construction, the new piecewise-linear paths ϖ_d are interior-disjoint, and thus define a piecewise-linear embedding of G.

A more careful argument using the Jordan-Schoenflies theorem implies that that any planar embedding of G is homeomorphic to a piecewise-linear embedding of G. That is, for any embedding $\phi : G^\top \hookrightarrow \mathbb{R}^2$, there is a homeomorphism $h : \mathbb{R}^2 \hookrightarrow\!\!\!\rightarrow \mathbb{R}^2$ such that $h \circ \phi$ is piecewise-linear. More strongly, a theorem of Alexander ⟨⟨cite⟩⟩ implies that any planar embedding of G is isotopic to a piecewise-linear embedding of G—there is

a continuous deformation of the entire plane that transforms one embedding into the other. The latter result implies that *every* planar embedding has a unique well-defined rotation system, even if the edges of the embedding are space-filling monsters.

4. (page 10) This case is actually redundant. A simple recursive argument, similar to the proof that every polygon triangulation has two ears, implies that at least two vertices on the outer face each have exactly two neighbors on the outer face.

5. (page 24) I will not add to the massive sea of ink that has already been spilled disputing the proper definition of "polyhedron"—and therefore the correct statement of Euler's formula for *all* polyhedra—except to point to the detailed discussions by Lakatos [53] and Grünbaum [38].

6. (page 24) Descartes' unpublished manuscript *Progymnasmata de solidorum elementis* [*Exercises in the Elements of Solids*] was most likely written around 1630 [22]. After Descartes' death in Sweden in 1650, his possessions were shipped to his friend Claude Clerselier in Paris; upon arrival, a box of manuscripts, including the *Progymnasmata*, fell into the Seine and was not recovered for three days. After carefully drying them, Clerselier made Descartes' manuscripts available to other scholars. Gottfried Leibniz transcribed several of these manuscripts, including the *Progymnasmata*, during a trip to Paris in 1676, most likely in an effort to collect evidence against recent charges by English mathematicians that his results were merely elaborations of Descartes' ideas. (Isaac Newton charged Leibniz of plagiarizing his calculus later that same year.) Descartes' original manuscript was then lost forever. Leibniz's hand-written copy vanished into his archives for almost two centuries; its existence was unknown until 1859, when it was discovered in an uncatalogued pile of Leibniz's papers by Louis Alexandre Foucher de Careil. Foucher de Careil published Leibniz's transcription [31], but his re-transcription introduced several significant errors, rendering it essentially useless. An accurate transcription of the *Progymnasmata* finally appeared in 1908, thanks to the combined efforts of several Cartesian and Leibnizian scholars [2]. The remarkable story is told in more detail by Federico [29], Richeson [74], and (with some creative embellishment) Aczel [1].

It is a matter of considerable dispute whether Descartes actually stated Euler's formula, and therefore deserves to share credit with Euler, or only came close, and therefore does not. The result that Descartes actually proves is the following [21, p. 269]:

> Ponam semper pro numero angulorum solidorum α & pro numero facirum φ....
> Numerus verorum angulorum planorum est $2\varphi - 2\alpha - 4$.
> [I always write α for the number of solid angles and φ for the number of faces....
> The total number of plane angles is $2\varphi - 2\alpha - 4$.]

In my opinion, the difference between Descartes' and Euler's formulas is one of notational emphasis, not content. As both Descartes [21, p. 268] and Euler [28, Proposition I] observe, the number of plane angles is exactly twice the number of edges. Had Descartes published his result, I believe even Euler (who exhibited surprise that the formula was not already known) would have called it "Descartes' formula".

Lakatos [53], Malkevich [61], and several others argue that Descartes should *not* share credit with Euler, because he did not identify edges as useful components of polyhedra in their own right. But this assertion is actually false. In the second half of the *Progymnasmata*, Descartes derives closed-form polynomial expressions for figurate numbers based on the Platonic and Archimedean solids; Descartes describes these solids by listing the number and shapes of the faces, the number of solid angles, and the number of *edges* in each [21, pp. 271–275].

Moreover, the phrase "Euler's formula" is commonly used to describe much more general results that Euler never even suggested, including L'Huillier's formula for polyhedra of higher genus [55, 56], Shläfli's formula for higher-dimensional polytopes [76], and Poincaré's formula relating face counts and Betti numbers of polyhedral complexes [69, 70]. In light of this sloppy generosity toward Euler, refusing to share credit with Descartes because of a minor notational difference seems remarkably inconsistent.

7. (page 24) Meister cites both Euler [27, 28] and Karsten [47] in a footnote, in which he claims ignorance of their prior work before Kästner brought it to Meister's attention shortly before publication. Both Karsten and Meister reproduce Euler's argument that any polyhedron with n vertices has between $\lceil n/2 \rceil + 2$ and $2n - 4$ facets. Meister also proved the by construction that for any integers n and f such that $n \geq 4$ and $\lceil n/2 \rceil + 2 \leq f \leq 2n - 4$, a convex polyhedron with n vertices and f facets actually exists; this result is usually attributed to Steinitz [82], who independently proved it 120 years later.

Both Karsten and Meister appear to be almost completely forgotten. One of the few places where Meister's results are cited is Brückner's 1900 treatise on polygons and polyhedra [9], which includes an *extremely* detailed historical survey. Brückner correctly observed that Meister's work "... seems to have received little attention, because its results are later presented as new by others." ("Die Abhandlung scheint aber wenig beachtet worden zu sein, denn die in ihr niedergelegten Resultate werden verschiedentlich später von anderen als neu vorgetragen.")

In strict accordance with Stigler's Law of Eponymy, recursively applied, Brückner's otherwise exhaustive survey does not mention Karsten at all. In fact, the *only* citation of Karsten's work on Euler's formula I have seen is the footnote in Meister's paper. Karsten is slightly better known for his geometric interpretation of logarithms of negative and complex numbers [46], although even this interpretation is usually attributed to De Morgan [20]. Cajori [11, 12] describes the obscurity of Karsten's discovery as follows: "It looks very much as if the transactions of academies had been in some cases the safest places for the concealment of scientific articles from the scientific public."

8. (page 25) The lacuna in Cauchy's proof argument is often ignored even by modern mathematicians, *even in the context of discussing formal proofs of Euler's formula*. For example, Lakatos mentions the issue in passing [53, pp. 10–13] but clearly considers it less worthy of discussion than Cauchy's sloppy definition of "polyhedron". Lakatos never

justifies Cauchy's claim that a shelling order exists for any convex polyhedron; instead, he cites a more recent proof by Reichart [73], written in answer to an "exercise" posed by van der Waerden [89]. (In fact, Reichart proves a much stronger result: The triangles in any *infinite* simply-connected surface triangulation can be indexed $\triangle_1, \triangle_2, \ldots$ so that for all $n \geq 1$, the union $\bigcup_{i=1}^{n-1} \triangle_i$ is homeomorphic to a disk.)

More recently, in a discussion of persistent errors in mathematical proofs, Bundy *et al.* [10] correctly observe that Cauchy's argument does not apply to all polyhedra (whatever that means), but like Lakatos, they do not notice that his argument is incomplete even for convex polyhedra. Categorical imprecision is apparently more philosophically interesting than simple omission.

I think the most annoying thing about secondary sources
is not that they contain errors or off-the-wall interpretations,
but that they NEVER include the specific information one is looking for.

— Craig Smoryński, *History of Mathematics: A Supplement* (2008)

Bibliography

[1] Amir D. Aczel. *Descartes' Secret Notebook*. Broadway Books, 2005. (35)

[2] Charles Adam. De solidorum elementis: Avertissement. *Ouvres de Descartes*, vol. X, 257–263, 1908. Reprinted by Vrin, Paris, 1996. (35)

[3] Bruce G. Baumgart. A polyhedron representation for computer vision. *Proc. AFIPS Natl. Comput. Conf.*, vol. 44, 589–596, 1975. AFIPS Press, Alrington, Va. (22)

[4] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*, 3rd edition. Springer-Verlag, 2008. (22)

[5] Otakar Borůvka. O jistém problému minimálním [About a certain minimal problem]. *Práce Moravské Přírodovědecké Společnosti v Brně III* 3:37–58, 1926. English translation in [68]. (26, 28)

[6] Otakar Borůvka. Příspěvek k řešení otázky ekonomické stavby elektrovodních sítí [Contribution to the solution of a problem of economical construction of electrical networks]. *Elektronický Obzor* 15:153–154, 1926. English translation in [68]. (26, 28)

[7] Erik Brisson. Representing geometric structures in d dimensions: Topology and order. *Discrete Comput. Geom.* 9:387–426, 1993. (22)

[8] Robin P. Bryant and David Singerman. Foundations of the theory of maps on surfaces with boundary. *Quart. J. Math.* 36(1):17–41, 1985. "Received 16 August 1982". (22)

[9] Max Brückner. *Vielecke und Vielflache: Theorie und Geschichte*. Teubner, 1900. (24, 25, 36)

[10] Alan Bundy, Mateja Jamnik, and Andrew Fugard. What is a proof? *Phil. Trans. Royal Soc. A* 363(1835):2377–2391, 2005. (37)

[11] Florian Cajori. Historical note on the graphic representation of imaginaries before the time of wessel. *Amer. Math. Monthly* 19(10/11):167–171, 1912. ⟨http://www.jstor.org/stable/2971879⟩. (36)

[12] Florian Cajori. History of the exponential and logarithmic concepts. *Amer. Math. Monthly* 20(4):107–117, 1913. ⟨http://www.jstor.org/stable/2972960⟩. (36)

[13] Augustin L. Cauchy. Recherches sur les polyèdres. *J. École Polytechnique* 9(16):68–86, 1813. Presented February 1811. *Œuvres Complètes*, IIe Sèrie 1:7–25, 1905. (25)

[14] Arthur Cayley. On the analytic problem of the polyedra. *Phil. Trans. Royal Soc. London* 147:183–185, 1857. Incorporated as an extended footnote in [50]. (9)

[15] Arthur Cayley. On the partitions of a close. *Phil. Mag. (Ser. 5)* 21:424–428, 1861. (25)

[16] Jianer Chen. Algorithmic graph embeddings. *Theoret. Comput. Sci.* 161(2):247–266, 1997. (22)

[17] David R. Cheriton and Robert Endre Tarjan. Finding minimum spanning trees. *SIAM J. Comput.* 5:724–742, 1976. (29)

[18] Gustave Choquet. Etude de certains résaux de routes. *Compt. Rend. Acad. Sci., Paris* 206:310–313, 1938. (28)

[19] Gopal Danaraj and Victor Klee. A representation of 2-dimensional pseudomanifolds and its use in the design of a linear-time shelling algorithm. *Algorithmic Aspects of Combinatorics*, 53–63, 1978. Ann. Discrete Math. 2, North-Holland. (22)

[20] Augustus De Morgan. On the foundations of algebra, No. II. *Trans. Cambridge Phil. Soc.* 7(3):287–300, 1837–42. Read November 29, 1841. (36)

[21] René Descartes. De solidorum elementis: Excerpta ex manuscriptis Cartesii. *Ouvres de Descartes*, vol. X, 265–277, 1908. (24, 35, 36)

[22] René Descartes. *Progymnasmata de solidorum elementis*. Unpublished manuscript, c. 1630. Transcribed by Gottfried Leibniz, 1676. *Ouvres de Descartes* X:265–277, 1908. Corrigenda in *Ouvres de Descartes* XI:690–692, 1909. Reprinted by Vrin, Paris, 1996. (24, 35)

[23] Charles M. Eastman. Introduction to computer aided design. Course Notes, Carnegie-Mellon University, 1982. (22)

[24] David Eppstein. Twenty proofs of Euler's formula: $V - E + F = 2$. *The Geometry Junkyard*, 2013. ⟨http://www.ics.uci.edu/~eppstein/junkyard/euler/⟩. Last accessed September 5, 2017. (24)

[25] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae* 8:128–140, 1741. ⟨http://www.eulerarchive.com/pages/E053.html⟩. Presented to the St. Petersburg Academy on August 26, 1735. *Opera Omnia* I.7:1–10. (4)

[26] Leonhard Euler. Letter to Christian Goldbach, November 14, 1750. ⟨http://eulerarchive.maa.org/correspondence/letters/OO0863.pdf⟩. Published as Lettre

CXXXV [théorèmes de stéréométrie], *Correspondance mathématique et physique de quelques célèbres géomètres du XVIIIème siècle*, 536–539, 1843. To appear in *Opera Omnia* IV-A.4. (24)

[27] Leonhard Euler. Demonstratio nonnullarum insignium proprietatum, quibus solida hedris planis inclusa sunt praedita. *Novi Commentarii academiae scientiarum Petropolitanae* 4:140–160, 1758. ⟨http://www.eulerarchive.com/pages/E231.html⟩. Presented to the St. Petersburg Academy on April 6, 1752. *Opera Omnia* I.26:94–108. (24, 36)

[28] Leonhard Euler. Elementa doctrinae solidorum. *Novi Commentarii academiae scientiarum Petropolitanae* 4:109–140, 1758. ⟨http://www.eulerarchive.com/pages/E230.html⟩. Presented to the Berlin Academy on November 26, 1750. *Opera Omnia* I.26:71–93. (24, 35, 36)

[29] Pasquale Joseph Federico. *Descartes on Polyhedra: A study of the* De Solidorum Elementis. Sources in the History of Mathematics and Physical Sciences 4. Springer-Verlag, New York, NY, 1982. (35)

[30] Kazimierz Florek, Jan Łukaszewicz, H. Perkal, Hugo Steinhaus, and S. Zubrzycki. Sur la liaison et la division des points d'un ensemble fini. *Colloq. Math.* 2:282–285, 1951. (28)

[31] Louis Alexandre Foucher de Careil, editor. *Œuvres inédites de Descartes*. Ladrange, Paris, 1859. (35)

[32] Hubert de Fraysseix, Janos Pach, and Richard Pollack. Small sets supporting Fáry embeddings of planar graphs. *Proc. 20th Annu. ACM Sympos. Theory Comput.*, 426–433, 1988. (9)

[33] Hubert de Fraysseix, Janos Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica* 10(1):41–51, 1990. (9)

[34] István Fáry. On straight lines representation of planar graphs. *Acta Sci. Math. Szeged* 11:229–233, 1948. (9)

[35] Andrew S. Glassner. Maintaining winged-edge data structures. *Graphics Gems II*, chapter IV.6, 191–201, 1991. Academic Press. (22)

[36] Ronald L. Graham and Pavol Hell. On the history of the minimum spanning tree problem. *Ann. History Comput.* 7(1):43–57, 1985. (26)

[37] Johann August Grunert. Einfacher Beweis der von Cauchy und Euler gefundenen Sätze von Figurennetzen und Polyëdern. *J. Reine Angew. Math.* 1827(2):367, 1827. (25)

[38] Branko Grünbaum. Are your polyhedra the same as my polyhedra? *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, 461–488, 2003. Springer. (35)

[39] Leonidas J. Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graphics* 4(2):75–123, 1985. (22)

[40] Heini Halberstam and Richard E. Ingram, editors. *The Mathematical Papers of Sir William Rowan Hamilton—Vol. III, Algebra*. Cunningham Memoir 15. Cambridge Univ. Press, 1967. (9, 41)

[41] William Rowan Hamilton. Letter to John T. Graves (on the Icosian), October 17, 1856. Published in [40, pp. 612–265]. (9)

[42] William Rowan Hamilton. Memorandum respecting a new system of roots of unity. *Phil. Mag. (Ser. 4)* 12:446, 1856. Reprinted in [40, p. 610]. (9)

[43] Meier Hirsch. *Sammlung geometrische Aufgaben*. vol. 2. Heinrich Frölich, Berlin, 1807. (25)

[44] Morris W. Hirsch. *Differential Topology*. Graduate Texts in Mathematics 33. Springer-Verlag, 1976. (34)

[45] Camille Jordan. Recherches sur les polyèdres. *J. Reine Angew. Math.* 66:22–85, 1866. (25)

[46] Wenceslaus Johann Gustav Karsten. Abhandlung von den Logarithmen verneinter Grössen, 1. Abtheilung. *Abh. Churfürstlich-Baierischen Akad. Wiss.* 5:3–110, 1768. (36)

[47] Wenceslaus Johann Gustav Karsten. *Lehrbegrif der gesamten Mathematik*. vol. 2. Anton Ferdinand Röse, Griefswald, 1768. 2nd edition, 1786. (24, 36)

[48] Thomas P. Kirkman. On the representation and enumeration of polyhedra. *Memoirs Lit. Phil. Soc. Manchester* 12:47–70, 1855. Cited in [50]. (9)

[49] Thomas P. Kirkman. On the representation of polyedra. *Phil. Trans. Royal Soc. London* 146:413–418, 1856. (9)

[50] Thomas P. Kirkman. On autopolar polyedra. *Phil. Trans. Royal Soc. London* 147:183–215, 1857. (9, 39, 41)

[51] David G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.* 12(1):28–35, 1983. (32)

[52] Philip N. Klein and Shay Mozes. *Optimization Algorithms for Planar Graphs*. Unpublished textbook draft, 2014. ⟨http://planarity.org/⟩. (8)

[53] Imre Lakatos. *Proofs and Refutations: The Logic of Mathematical Discovery*. Cambridge Univ. Press, 1976. (35, 36)

[54] Adrien-Marie Legendre. *Éléments de géométrie*. F. Didot, Paris, 1794. (25)

[55] Simon Antoine Jean l'Huillier. Démonstration immédiate d'un théorème fondamental d'Euler sur les polyhèdres, et exception dont ce théorème est susceptible. *Mémoires de l'Académie Impériale des Sciences de Saint-Petersbourg* 4:271–301, 1811. (23, 24, 36)

[56] Simon Antoine Jean l'Huillier. Mémoire sur la polyédrométrie contenant une démonstration directe du théorème d'Euler sur les polyédres, et un examen des diverses exceptions auxquelles ce théorème est assujetti. *Annales de Mathématiques Pures et Appliquées [Annales de Gergonne]* 3:169–189, 1813. Summarized by Joseph Diaz Gergonne. (23, 24, 36)

[57] Pascal Lienhardt. Subdivisions of surfaces and genealized maps. *Proc. Eurographics '89*, 439–452, 1989. (22)

[58] Pascal Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Comput. Aided Design* 23(1):59–82, 1991. (22)

[59] Sóstenes Lins. Graph-encoded maps. *J. Comb. Theory Ser. B* 32:171–181, 1982. (22)

[60] Johann Benedict Listing. Der Census räumlicher Complexe, oder Verallgemeinerung des Euler'schen Satzes von den Polyedern. *Abh. König. Ges. Wiss. Göttingen* 10:97–182, 1861. Presented December 1861. (25)

[61] Joseph Malkevitch. The first proof of Euler's formula. *Mitteilungen Math. Sem. Giessen* 165(III):77–82, 1984. Coxeter Festschrift. (36)

[62] Martin Mareš. Two linear time algorithms for MST on minor closed graph classes. *Archivum Mathematicum* 40(3):315–320, 2004. (29)

[63] Martin Mareš. The saga of minimum spanning trees. *Comp. Sci. Rev.* 2:165–221, 2008. (26)

[64] Tomomi Matsui. The minimum spanning tree problem on a planar graph. *Discrete Appl. Math.* 58(1):91–94, 1995. (31)

[65] Albrecht Ludwig Friedrich Meister. Commentatio de solidis geometricis pro cognoscenda eorum indole in certos ordines et versus disponendis. *Commentationes Mathematicae (Soc. Reg. Scient. Gott.)* 7:3–74 + 2 plates, 1784/1785. Presented October 1, 1875. (24)

[66] David E. Muller and Franco P. Preparata. Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.* 7:217–236, 1978. (22)

[67] Martti Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, MD, 1988. (22)

[68] Jaroslav Nešetřil, Eva Milková, and Helena Nešetřilová. Otakar Borůvka on minimum spanning tree problem: Translation of both the 1926 papers, comments, history. *Discrete Math.* 233(1–3):3–36, 2001. (28, 38)

[69] Henri Poincaré. Sur la généralisation d'un théorème d'Euler relatif aux polyèdres. *C. R. Hebd. Séances Acad. Sci. Paris* 117:144–145, 1893. (36)

[70] Henri Poincaré. Complément à l'Analysis Situs. *Rendiconti del Circulo Matematico di Palermo* 13:285–343, 1899. English translation in [71]. (36)

[71] Henri Poincaré. *Papers on Topology:* Analysis Situs *and Its Five Supplements*. History of Mathematics 37. Amer. Math. Soc., 2010. Translated from the French and with an introduction by John Stillwell. (43)

[72] Franco P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM* 20:87–93, 1977. (22)

[73] Hans Reichardt. Lösung der Aufgabe 274. *Jahresbericht Deutschen Math.-Verein.* 51 (2. Abteilung):23–24, 1941. Received July 22, 1939. (37)

[74] Daniel Richeson. *Euler's Gem*. Princeton Univ. Press, 2005. (35)

[75] Alan Saalfeld. Joint triangulations and triangulation maps. *Proc. 3rd Ann. Symp. Comput. Geom.*, 195–204, 1987. (15)

[76] Ludwig Schläfli. *Theorie der Vielfachen Kontinuität*. Zürcher & Furrer, Zürich, 1901. Written 1850–1852. *Gesammelte Mathematische Abhandlungen* 1:167–387, 1950. Written 1850–1852. (36)

[77] Werner Helmut Schmidt. Wenceslaus Johann Gustav Karsten (1732–1787). Von Neubrandenburg nach Halle - Bewerbungen, Beziehungen, Berufungen. Reports on Didactics and History of Mathematics, Report No. 04-02, Dept. Math. Comput. Sci., Martin-Luther-Universität Halle-Wittenberg, 2004. ⟨http://www.mathematik.uni-halle.de/reports/shadows/04-02report.html⟩. (24)

[78] Walter Schnyder. Planar graphs and poset dimension. *Order* 5(4):323–343, 1989. (11)

[79] Walter Schnyder. Embedding planar graphs on the grid. *Proc. 1st Ann. ACM-SIAM Symp. Discrete Algorithms*, 138–148, 1990. (11)

[80] Georges Sollin. Le trace des canalisations. *Programming, Games, and Transportation Networks*, p. 181, 1965. Wiley. (28)

[81] Karl Georg Christian von Staudt. *Geometrie der Lage*. Verlag von Bauer and Rapse (Julius Merz), Nürnberg, 1847. (22, 25)

[82] Ernst Steinitz. Über die Eulerschen Polyederrelationen. *Arch. Math. Phys. (3)* 11:86–88, 1906. (36)

[83] Ernst Steinitz. Polyeder und Raumeinteilungen. *Enzyklopädie der mathematischen Wissenschaften mit Einschluss ihrer Anwendungen* III.AB(12):1–139, 1916. (9, 16)

[84] Sherman K. Stein. Convex maps. *Proc. Amer. Math. Soc.* 2(3):464–466, 1951. (9)

[85] Mirko Stojaković. Über die Konstruktion der ebenen Graphen. *Univ. Beograd. Godišnjak Filozof. Fak. Novom Sadu* 4:375–378, 1959. (9)

[86] Robert Endre Tarjan. *Data Structures and Network Algorithms*. CBMS-NSF Regional Conference Series in Applied Mathematics 44. SIAM, 1983. (26)

[87] William T. Tutte. Convex representations of graphs. *Proc. London Math. Soc., Ser. 3* 10(1):304–320, 1960. (16)

[88] William T. Tutte. How to draw a graph. *Proc. London Math. Soc.* 13(3):743–768, 1963. (16)

[89] Bartel L. van der Waerden. Aufgabe 274. Eine elementare kombinatorisch-topologische Aufgabe, deren Lösung für eine einfache Begründung der Uniformisierungstheorie von großer Bedeutung ist. *Jahresbericht Deutschen Math.-Verein.* 49 (2. Abteilung):1, 1939. (37)

[90] Klaus Wagner. Bemerkungen zum Vierfarbenproblem. *Jahresbericht Deutschen Math.-Verein.* 46:26–32, 1936. (9)

[91] Kevin Weiler. Edge-based data structures for solid modeling in curved-surface environments. *IEEE Comput. Graph. Appl.* 5(1):21–40, 1985. (22)

[92] Hassler Whitney. Non-separable and planar graphs. *Trans. Amer. Math. Soc.* 34:339–362, 1932. (20)

[93] Hassler Whitney. Planar graphs. *Fund. Math.* 21:73–84, 1933. (20)