9 Planar Graphs

9.1 Abstract graph vocabulary

A graph is an abstract combinatorial structure that models pairwise relationships. You probably have a good idea what a graph is already. Nevertheless, to avoid subtle but pervasive differences in terminology, notation, and basic assumptions, I will carefully define everything from scratch.

Instead, we formally define an *abstract graph* to be a quadruple $G := (V, D, \mathsf{rev}, \mathsf{head})$, 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 the darts D such that $\mathsf{rev}(\mathsf{rev}(d)) = d \neq \mathsf{rev}(d)$ for every dart $d \in D$;
- head is a function from the darts D to the vertices V.

Darts are also called *half-edges* or *arcs* or *brins* (French for "strands").

For any dart d, we call the dart $\mathsf{rev}(d)$ the *reversal* of d, and we call the vertex $\mathsf{head}(d)$ the *head* of d. The *tail* of a dart is the head of its reversal: $\mathsf{tail}(d) := \mathsf{head}(\mathsf{rev}(d))$. 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. I often write $u \to v$ to denote a 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 $\{d, \mathsf{rev}(d)\}$ is called an *edge* of the graph. We often write E to denote the set of edges of a graph. The constituent darts of an edge e are arbitrarily denoted e^+ and 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, $\mathsf{head}(e^+) = \mathsf{tail}(e^+)$. 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.[1] Non-simple graphs are sometimes called "generalized graphs" or "multigraphs"; I will just call them "graphs".

Let me repeat this louder for the kids in the back: ***Graphs are not necessarily simple.***

The degree of a vertex v, denoted $\deg_G(v)$ (or just $\deg(v)$ if the graph G is clear from context), is the number of darts whose tail 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.

9.2 Data structures

Here is an equivalent definition that might be clearer to computer scientists: **A graph is whatever can be stored in a standard graph data structure.**

[1] Simple graphs are traditionally defined as pairs (V, E), where V is an arbitrary non-empty set of *vertices*, and E is a *set* of unordered pairs of vertices called *edges*. While admirably terse, this definition only works for *simple* graphs, and we really don't want to *define* graphs to be simple.

The canonical textbook graph data structure is the *incidence list*. (For simple graphs, the same data structure is more commonly called an *adjacency list*.) The n vertices of the graph are represented by distinct integers between 1 and n. A standard incidence list consists of a vector/array indexed by the vertices; each array entry in the array points to a linked list of the darts leaving the corresponding vertex. (The order of darts in these linked lists is unimportant; we use linked lists only because they support certain operations quickly.) The record for each dart d contains the index of $\mathsf{head}(d)$, a pointer to the record for $\mathsf{rev}(d)$. Storing a graph with n vertices and m edges in an incidence list requires $O(n+m)$ space.

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 a few 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, and for those few that do, we can use hash tables instead of linked lists.

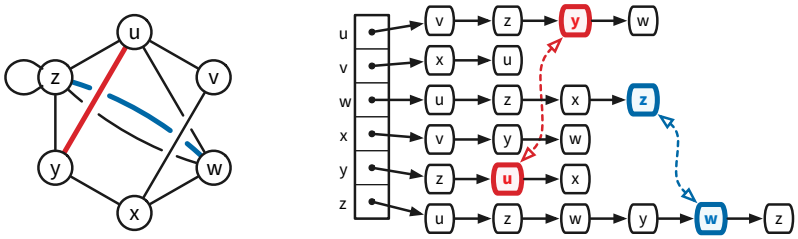

Figure 1: An incidence list, with the dart records for two edges emphasized. For clarity, most reversal pointers are omitted.

More generally, "array" and "linked list" can be replaced by any suitable data structures that allow random access and fast iteration, respectively. A particularly simple and efficient implementation keeps *all* data in standard arrays. For a graph with n vertices and m edges, we represent vertices by integers between 0 and $n-1$, edges by integers from 0 to $m-1$, and darts by integers between 0 and $2m-1$. Each edge e is composed of darts $e^- = 2e$ and $e^+ = 2e+1$; thus, the reversal of any dart d is obtained by flipping its least significant bit: $d \oplus 1$. The actual data structure consists of three arrays:

- $\mathsf{first}[0..n-1]$, where $\mathsf{first}[v]$ is any dart leaving vertex v.
- $\mathsf{head}[0..2m-1]$, where $\mathsf{head}[d]$ is the head of dart d.
- $\mathsf{next}[0..2m-1]$, where $\mathsf{next}[d]$ is the successor of d in the list of darts leaving $\mathsf{tail}(d)$.

It will prove convenient to treat the list of darts leaving each vertex as a *circular* list; then next stores a permutation of the darts, each of whose cycles is the set of darts leaving a vertex. We may also want to store a predecessor array $\mathsf{prev}[0..2m-1]$ that stores the inverse of this permutation. We do not need a separate tail array, because $\mathsf{tail}(d) = \mathsf{head}[d \oplus 1]$.

For algorithms that make frequent changes to the graph (adding and/or deleting vertices and/or edges), we should use dynamic hash tables instead of raw arrays. Finally, we can easily store algorithm-dependent auxiliary data such as edge weights, distances, or flow capacities in separate arrays (or hash tables) indexed by vertices, edges, or darts, as appropriate.

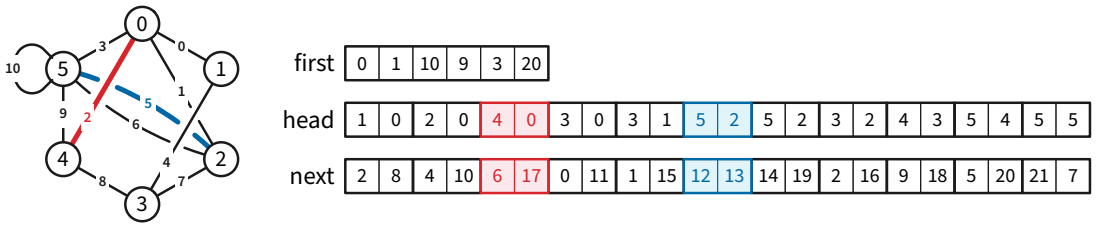

Figure 2: An incidence array representation of the same graph as Figure 1.

9.3 Planar graphs and planar maps

A *planar embedding* of a graph G represents its vertices as distinct points in the plane (typically *drawn* as small circles) and its edges as simple interior-disjoint paths between their endpoints. Equivalently, a planar embedding of G is a continuous injective function from the topological graph $G^\top$ to the plane. A graph is *planar* it it has at least one planar embedding. Somewhat confusingly, the image of a planar embedding of a planar graph is also called a *plane graph*.

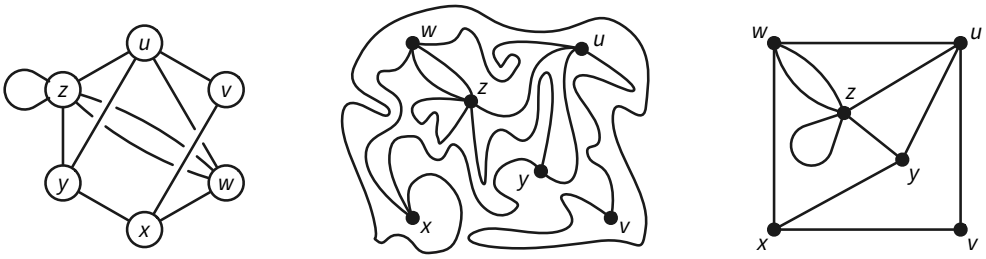

Figure 3: A planar graph G and two planar embeddings of G.

The components of the complement of the image of a planar embedding are called the *faces* of the embedding. Assuming the embedded graph is connected, the Jordan curve theorem implies that every bounded face is homeomorphic to an open disk, and the unique unbounded face is homeomorphic to the complement of a closed disk. For disconnected graphs, at least one face is homeomorphic to an open disk with a finite number of closed 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 the *left shore* of d, denoted $\mathsf{left}(d)$; symmetrically, the face just to the right is the *right shore* $\mathsf{right}(d)$. The left and right shores of a dart can be the same face. 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 $\deg_G(f)$ (or just $\deg(f)$ if G is clear from context), is the number of darts whose right shore is f.

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*. Trapezoidal decompositions and triangulations of polygons are both examples of planar maps. A planar map is called a *triangulation* if every face, including the outer face, has degree 3. The underlying graph (V, E) of a planar triangulation is *not* necessarily simple. (For readers familiar with topology, a triangulation is *not* necessarily a simplicial complex, but rather what Hatcher calls a Δ-complex.)

It is somewhat confusing standard practice to use the same symbol G (and the sme word "graph") to simultaneously denote an abstract planar graph G, the corresponding topological graph

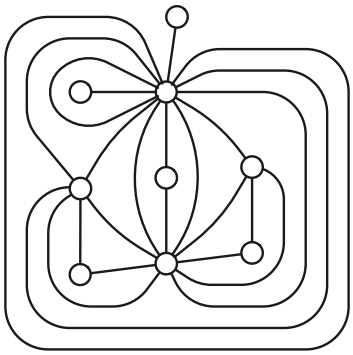

Figure 4: A nonsimple planar triangulation.

$G^\top$, and the image of a planar embedding of G (which by definition is homeomorphic to $G^\top$), and even the resulting planar map (V, E, F) induced by that embedding. I will *try* to carefully distinguish between these various objects when the distinction matters, but there is a serious tension here between formality and clarity, so I am very likely to slip occasionally.

9.4 Rotation systems

As usual in topology, we are not really interested in *particular* embeddings of maps, but rather *isotopy* classes of embeddings or maps. Two embeddings of the same graph G are isotopic if one can be continuously deformed to the other through embeddings. Fortunately, every equivalence class of embeddings has a concrete combinatorial representation, called a *rotation system*.

Recall that a *permutation* of a finite set X is a bijection $\pi : X \to X$. 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* of an embedding of G is a permutation succ of the darts of G, called the *successor permutation*. The successor $\mathsf{succ}(d)$ of any dart d is the successor of d in the *clockwise* sequence of darts entering $\mathsf{head}(d)$.[2]

In other words, a rotation system is (almost) an incidence list where the order of darts in each list actually matters! The only annoying discrepancy is that rotation systems order darts *into* each vertex, while incidence lists order darts *out* of each vertex, but we can easily translate between these two standards using the identity $\mathsf{next}(d) = \mathsf{rev}(\mathsf{succ}(\mathsf{rev}(d)))$.

The faces of any connected graph embedding are also implicitly 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* $\mathsf{succ}^*(d) := \mathsf{rev}(\mathsf{succ}(d))$ is the next dart after d in *counterclockwise* order around the boundary of $\mathsf{left}(d)$.

Thus, we can define a *combinatorial map* as a triple $\Sigma = (D, \mathsf{rev}, \mathsf{succ})$, where D is a finite set of

[2]Because the edges of a planar embedding can be *arbitrary* paths, it is not immediately obvious that this cyclic order is well-defined. In fact, the existence of a consistent order follows from careful application of the Jordan-Schönflies theorem.

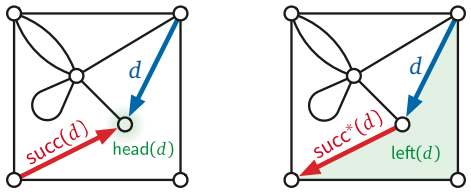

Figure 5: The successor and dual successor of a dart in a planar map.

darts, rev is a fixed-point-free involution of D, and succ is a permutation of D. Then the *vertices* of the combinatorial map are the orbits of succ, the *edges* are the orbits of rev, and the *faces* are the orbits of rev(succ). The vertices and edges define the *1-skeleton* or *underlying graph* of Σ. Again, it is somewhat confusing standard practice to use the same symbol Σ (and the same word "map") to denote both a topological planar map (V, E, F) and the corresponding combinatorial map $(D, \mathsf{rev}, \mathsf{succ})$.

9.5 Duality

Recall that a combinatorial map is a triple $\Sigma = (D, \mathsf{rev}, \mathsf{succ})$, where D is a set of darts, rev is an involution of D, and succ is a permutation of D. For any such triple, the triple $\Sigma^* = (D, \mathsf{rev}, \mathsf{rev} \circ \mathsf{succ})$ is also a well-defined combinatorial map, called the *dual map* of Σ.

- The *vertices* of Σ^* are the orbits of $\mathsf{rev} \circ \mathsf{succ}$, which are also the faces of Σ.
- The *edges* of Σ^* are the orbits of rev, which are also the edges of Σ.
- The *faces* of Σ^* are the orbits of $\mathsf{rev} \circ \mathsf{rev} \circ \mathsf{succ} = \mathsf{succ}$, which are also the vertices of Σ!

In other words, each vertex v, edge e, dart d, or face f of the original map Σ corresponds to—or more evocatively, "is dual to"—or more formally, ***IS***—a face v^*, edge e^*, dart d^*, or vertex f^* of the dual map Σ^*, 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 $\mathsf{head}(d^*) = \mathsf{left}(d)^*$ and $\mathsf{tail}(d^*) = \mathsf{right}(d)^*$. Intuitively, the dual of a dart is obtained by rotating it a quarter-turn counterclockwise.

Duality is trivially an involution: $(\Sigma^*)^* = \Sigma$, because $\mathsf{rev} \circ \mathsf{rev} \circ \mathsf{succ} = \mathsf{succ}$. It immediately follows that for any dart d, we have $\mathsf{left}(d^*) = \mathsf{head}(d)^*$ and $\mathsf{right}(d^*) = \mathsf{tail}(d)^*$.

We can also directly define the dual of a *topological* map Σ as follows. Choose an arbitrary point f^* in the interior of each face f of Σ. Let F^* denote the collection of all such points. For each edge e of Σ, choose a simple 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 Σ. Let E^* denote the collection of all such paths. These paths partition the plane into regions V^*, each of which contains a unique vertex.[3] The dual map Σ^* is the decomposition of the plane into vertices F^*, edges E^*, and faces V^*.

When G is an *embedded* graph, it is extremely common to define the *dual graph* G^* as the 1-skeleton of the dual map of the embedding. This habit is a bit misleading, however; duality is a correspondence between *maps* or *embeddings*, not between abstract graphs. An abstract planar graph can have many non-isomorphic planar embeddings, each of which defines a different "dual graph". Moreover, the dual of a *simple* embedded graph is not necessary simple; any vertex of

[3] You should verify this claim!

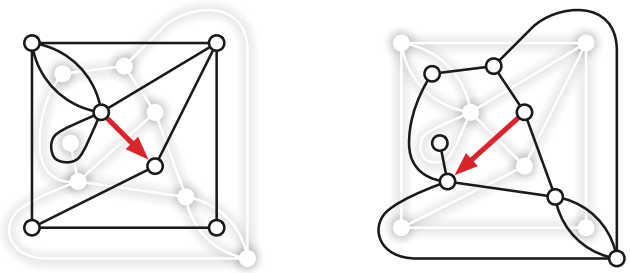

Figure 6: A planar map and its dual; one dart and its dual are emphasized.

degree 2 in G 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!

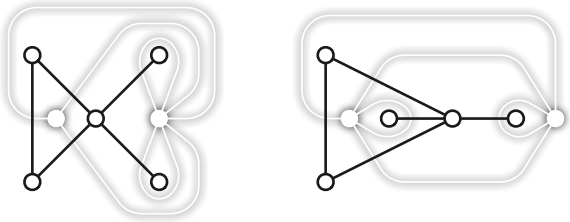

Figure 7: Two planar embeddings of a simple planar graph, with non-simple, non-isomorphic dual graphs.

9.6 Handedness

There are several apparently arbitrary choices to make in the definitions of incidence lists, rotation systems, and duality. Should we store cycles of darts with the same head or the same tail? Should darts be ordered in clockwise or counterclockwise order around vertices? Or around faces? Should $\mathsf{head}(d^*)$ be defined as $\mathsf{left}(d)^*$ or $\mathsf{right}(d)^*$? Different standards are used by different authors, by the same authors in different papers, and sometimes even within the same paper. (Mea culpa!)

Let me attempt to justify, motivate, or at least provide mnemonics for the specific choices I use in these notes. Some of these rules will only make sense later.

- Dualizing a dart should look (and act?) like multiplying a complex number by i: a quarter turn *counterclockwise*. Thus, $\mathsf{head}(d^*) = \mathsf{left}(d)^*$ and $\mathsf{tail}(d^*) = \mathsf{right}(d)^*$.
- Duality is an involution, dammit. Thus, $\mathsf{left}(d^*) = \mathsf{head}(d)^*$ and $\mathsf{right}(d^*) = \mathsf{tail}(d)^*$. It follows that primal and dual planes must have opposite orientations!
- Simple *counterclockwise* cycles have winding number $+1$. So the dual successor function should order darts *counterclockwise* around their *left* shores. Thus, the primal successor function should order darts *clockwise* around their *heads*.
- Derivatives measure how much a function *increases*, so $\delta\omega(d) = \omega(\mathsf{head}(d)) - \omega(\mathsf{tail}(d))$. On the other hand, the directed boundary of a face should be a counterclockwise cycle, so $\partial\alpha(d) = \alpha(\mathsf{left}(d)) - \alpha(\mathsf{right}(d))$. Hey, look, consistency!

This standard creates an annoying discrepancy between the mathematical abstraction of a rotation system and its implementation as an incidence list. Rotation systems order darts around

their heads because that makes the math cleaner; incidence lists order darts around their tails because that better fits our intuition about searching graphs by following directed edges outward. Rather than give up either useful intuition, we'll rely on (and if necessary implement) the identity $\mathsf{next}(d) = \mathsf{rev}(\mathsf{succ}(\mathsf{rev}(d)))$.

9.7 Other derived maps

- Medial $G^{\times}$
- Radial/angle $G^{\diamond}$
- Band/ribbon decomposition $G^{\square}$
- Barycentric subdivision G^{+}
- Tait graph $\gamma^{\bowtie}$ of a curve γ

9.8 Aptly Yadda Yadda

Table 1: A (partial) duality dictionary.

Primal Σ	Dual Σ^*	Primal Σ	Dual Σ^*
vertex v	face v^*	$\mathsf{head}(d)$	$\mathsf{left}(d^*)$
dart d	dart d^*	$\mathsf{tail}(d)$	$\mathsf{right}(d^*)$
edge e	edge e^*	$\mathsf{left}(d)$	$\mathsf{head}(d^*)$
face f	vertex f^*	$\mathsf{right}(d)$	$\mathsf{tail}(d^*)$
succ	$\mathsf{rev} \circ \mathsf{succ}$	clockwise	counterclockwise
rev	rev		