CS 7301.003.20F Lecture 3—August 24, 2020

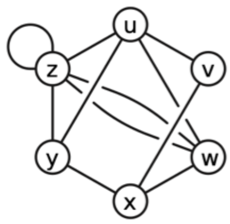
Main topics are #introduction, #administrivia, #planar_graphs.

A Different Way to Think About Graphs

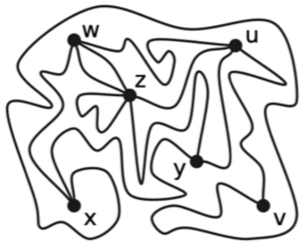
- Now that we've gotten some basics out of the way regarding planar curves, we're going to turn to graphs and their embeddings.
- Traditionally, graphs are defined as pairs (V, E) where V is an arbitrary finite set of vertices, and the edges E are pairs of vertices. Today, I want to give a different definition that ends up being incredibly useful when we work with embedded graph algorithms. I like to think of this as a more edge-centric definition than what you normally see.
- Erickson defines an *abstract graph* (which I'll probably just call a graph from here on) as 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 ≠ rev(d) for every dart d. In other words, rev is an involution.
  - head is a function from darts D to vertices V
- We call rev(d) the *reversal* of dart d and head(d) the *head* (naturally).
- The *tail* of dart d is tail(d) := head(rev(d)). The head and tail are *endpoints* of d. The intuition here is that a dart is a directed path that *leaves* its tail and *enters* its head. I'll often write u → v as a dart from u to v.
- The unordered pair {d, rev(d)} is called an *edge* of the graph, and we use E to denote the edges. I'll often use uv for an edge with endpoints u and v. We'll use e^+ and e^- to denote constituent darts of edge e. You can think of both darts as *orientations* of the edge.
- It's sometimes useful to think of abstract graphs as topological spaces. Erickson defines the *topological graph* G^T as a set V^T of distinct points, one per vertex, and E^T as a set of disjoint closed real intervals, one per edge. You then (formally) take the quotient space (V^T cup E^T) / ~ where the endpoints of the edge intervals are identified with the head and tail of each edge. But it's probably easier to just imagine drawing the graph in 3D without the edges intersecting, and that's your topological graph.
- From here on, I'll assume you're familiar with other basic graph definitions and things like representing graphs with adjacency lists. I'll point out when the abstract graph representation affects these things in surprising ways. Erickson describes all the gritty details if you want a refresher.

Planar Graphs

- A *planar embedding* of a graph G is a continuous injective function from the topological graph G^T to the plane. In other words, vertices of G map to points, and edges map to interiorly disjoint simple paths between their vertices' endpoints.
- A *planar graph* is an abstract graph with at least one planar embedding. The embedding itself is often called a *plane graph* for some reason.

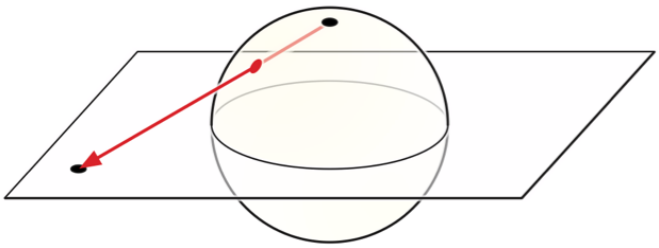


A planar graph G.



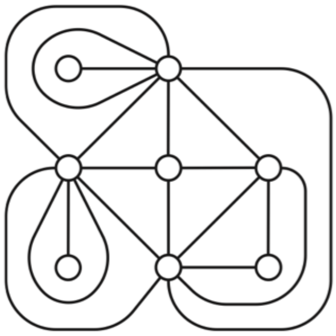
A planar embedding of G.

- One thing that's not so obvious is that planar graphs are exactly those with a continuous injective function to the sphere $S^2 = \{(x, y, z) : x^2 + y^2 + z^2 = 1\}$. The *stereographic projection* map $st : S^2 \setminus (0, 0, 1) \rightarrow R^2$ is defined as $st(x, y, z) := (x / (1 - z), y / (1 - z))$. The projection st(p) maps p to the intersection of the line through p and (0, 0, 1) with the xy-plane. Any embedding on the sphere avoiding (0, 0, 1) maps to a planar embedding, and this embedding is invertable.



Stereographic projection.

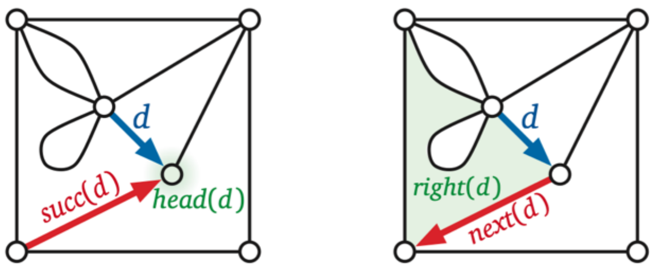
- The components of the complement of the image of the embedding are called the *faces* of the embedding. If the graph is connected, then every face is homeomorphic to an open disk. There's a single *outer face* for a planar embedding.
- We'll call faces on either side of an edge the *shores* of that edge.
- For a dart d, the face just left of d's embedding is the *left shore* of d denoted, left(d). There's also a right shore, right(d). Sometimes, left(d) = right(d).
- The triple (V, E, F) of vertices, edges, and faces of a connected graph embedding is called a *planar* or *spherical map*. It's a *triangulation* if every face (including the outer one) has degree 3. A triangulation need not be simple.



A planar triangulation.

Rotation Systems

- So now let's put these concepts of abstract and planar graphs together. For most embedded graph algorithms, we really don't care about the exact embedding, but instead about its combinatorial properties. We describe these succinctly using a *rotation system*.
- So first, recall a *permutation* of a finite set X is a bijection from X to itself. For a permutation pi, let pi^0(x) : = x and p^k(x) = pi(pi^{k-1}(x)); i.e., apply pi to x k times.
- The *orbit* of an element x is the set {pi^k(x) | k in N} = {x, pi(x), pi^2(x), …}. The restriction of pi to any of its orbits is a *cyclic* permutation; the sequence x, pi(x), pi^2(x), … repeatedly cycles through the elements in the orbit of x. The orbits of any two elements of X are either disjoint or identical.
- The rotation system for a graph embedding is a permutation of its darts called the *successor* permutation. You can think of the successor succ(d) of dart d as the next dart entering head(d) in counterclockwise order after d.
- Something neat is that the rotation system already encodes the faces. 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).



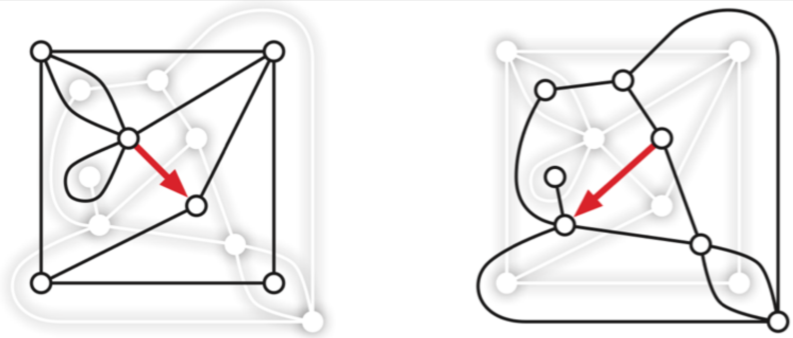
The successor and dual successor of a dart in an embedded planar graph.

- Something a little odd is that if your graph is disconnected, then you cannot determine from a rotation system which boundary components belong to the same face of an embedding. I don't really mind, this though. Most algorithms work with each component of the graph independently anyway. And some authors may even define faces based purely on the rotation system, so each connected component essentially has its own independent set of faces. I like thinking of things this way myself, because some processes become much easier to describe if you just lose yourself in the abstract rotation system and see where it takes you. I hope to help you understand why by the end of the lecture.

- Rotation systems are perfectly well defined even for graphs that aren't planar! Instead they may correspond to a graph embedding on some orientable surface other than the sphere or plane. We'll talk more about these other surfaces in a few weeks.

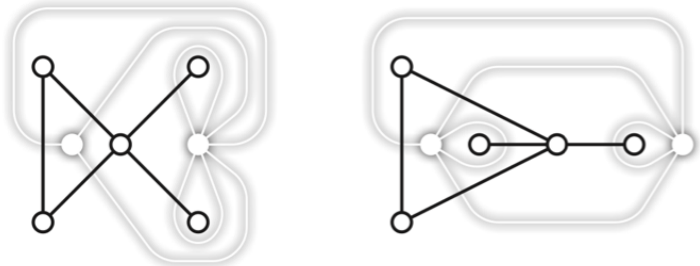
Duality

- Rotation systems involve this pair of permutations (succ, rev) over the darts such that every orbit of rev has exactly two elements. The orbits of succ are the vertices of an embedded graph, its edges are the orbits of rev, and (assuming its connected or by definition) its faces are the orbits of succ circ rev.
- The *dual* of the rotation system (succ, rev) is the rotation system (rev circ succ, rev). The embedded graph G^* determined by the rotation system is called the *dual graph*. So orbits of succ circ rev are the vertices of G^ *, edges are still orbits of rev, and faces of G^ * are orbits of succ.
- And that means each vertex v, edge e, dart d, and face f of graph G corresponds to—or is dual to—a distinct face v^ *, edge e^ *, dart d^ *, and vertex f^ * of dual graph G^ *, respectively.
- We may call original graph G the *primal graph*.
- The endpoints of primal edge e are dual to the shores of dual edge e^* and vice versa. So for any dart d, tail(d^ *) = left(d) and head(d^ *) = right(d). Symmetrically, left(d^ *) = tail(d) and right(d^ *) = head(d).



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

- We could also define G^* by taking the topological embedding of G, placing a point f^* in each face f, and adding an edge path e^* between edge e's shores' dual vertices.
- Duality is only defined by an embedded graph or rotation system. An abstract planar graph could have multiple embeddings with different duals.
- Also, note that the dual of a simple embedded graph may not be simple. Any vertex of degree 2 gives rise to two parallel edges around its dual face. And a bridge is dual to a loop!



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

- As usual when calling something dual, duality is an involution; rev circ rev circ succ = succ.
- Now when I was first learning this stuff, I got very confused at this point. First, the rotation system (succ, rev) describes the counterclockwise order of darts around each vertex, but (rev circ succ, rev) describes the clockwise order of darts around each face/dual vertex.
- Also, it appears that duality turns each dart 90 degrees clockwise; i.e., the dual dart goes left-to-right across the primal dart. Dualizing the dual doesn't cause the dart to flip a full 180 from its original orientation.
- I have two suggestions for helping visualize what's going on. First, you could just accept that the dual system encodes the clockwise order. Accept that dualizing changes the orientation and it all works.
- Alternatively, imagine that instead of drawing all these planar graphs on a whiteboard or the outside of a sphere, you were instead holding a piece of paper and using a marker to do all the drawing. You can see everything from both sides of the paper. Look at the primal graph on one side and the dual graph on the other. Now they both appear oriented in the same direction. This idea of a thin sheet in or through which everything is embedded may be helpful when we discuss non-orientable surfaces later in the semester.

Correspondences

- And now we get to why I really like thinking of the rotation system primarily instead of the embedding.
- Many fundamental structures in a connected planar graph G correspond directly to other structures in the dual graph G^*, and you can verify these claims by working directly with the rotation system.

primal G	**dual G^***	**primal G**	**dual G^***
vertex v	face v^*	empty loop	spur
dart d	dart d^*	loop	bridge
edge e	edge e^*	cycle	bond
face f	vertex f^*	even subgraph	edge cut
$tail(d)$	$left(d^*)$	spanning tree	complement of spanning tree
$head(d)$	$right(d^*)$	$G \setminus e$	G^* / e^*
$succ$	$rev \circ succ$	G / e	$G^* \setminus e^*$
clockwise	counterclockwise	minor $G \setminus X / Y$	minor $G^* \setminus Y^* / X^*$

Correspondences between features of primal and dual planar maps

- Here are some definitions that may not be obvious. Say G has n vertices and m edges.
  - a *spur* is an edge incident to a vertex of degree 1
  - a *bridge* is a single edge whose removal separates a graph into two components
  - an *even subgraph* is a subgraph in which every vertex has positive even degree. It's also the union of one or more edge-disjoint cycles.
  - A *cut* of G is a partition of its vertices into two non-empty subsets. Edges *cross* the cut from one side to the other. Edges crossing a cut form its *boundary*. An *edge cut* is the boundary of some cut.
  - An edge cut C is called a *bond* if no proper subset of C is the boundary of a cut.
  - G \ e denotes deleting edge e from G leaving n vertices and m - 1 edges.
  - G / e denotes *contracting* e, meaning we merge the endpoints of e and destroy e, leaving n - 1 vertices and m - 1 edges. If there are edges parallel with e, they become loops. By convention, we normally don't allow contraction of loops (but we can extend the definition to allow for contracting loops, maybe in the homework?)
  - A *minor* of G is any graph obtained by contracting edges of a subgraph of G. A *proper minor* is any minor that is not G itself.
  - Given a subgraph X of G where every vertex has positive degree, I'll use X^* to denote its dual