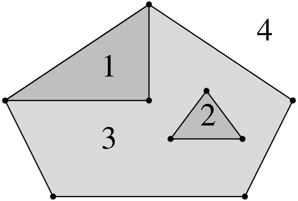
Planar Subdivision (chapter 2)

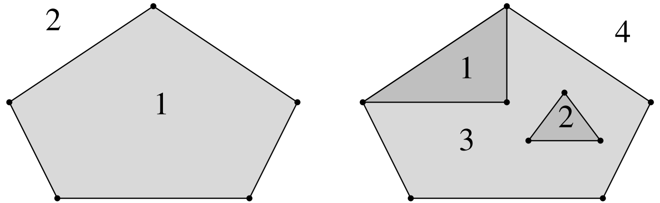
Elisha Sacks

Planar Subdivision



- Division of the plane into open regions called faces.
- The region boundary elements are line segments called edges.
- The edge endpoints are called vertices.
- The edges form outer and inner boundary loops.
- The unbounded face has no outer boundary.
- The others have one outer and zero or more inner boundaries.
- Example: face 4 is unbounded; face 3 has one inner boundary.

Euler Formula



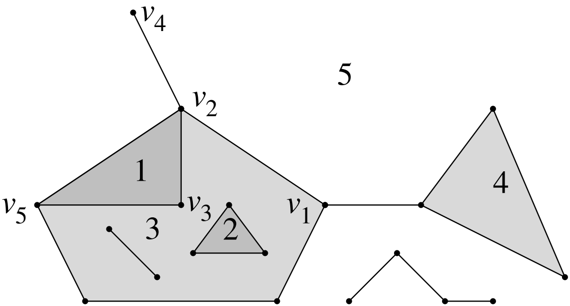
$v = 5, e = 5, f = 2, c = 1$ $\quad$ $v = 9, e = 10, f = 4, c = 2$

- Notation: v vertices, e edges, f faces, c components.
- Euler formula: $v - e + f = 1 + c$.
- This formula is used in many computational geometry proofs.
- It generalizes to polyhedrons and far beyond.

Boundary Representation

- Subdivisions are represented with a boundary representation.
- A vertex stores its coordinates and incident edges.
- An edge ab stores its tail vertex a, twin edge ba, the next edge on its loop, and the face that it bounds.
- A face stores one edge from each of its boundary loops.
- A face is to the left of its boundary edges when they are traversed from tail to head.
- The boundary representation is also called a doubly connected edge list or a winged edge structure.

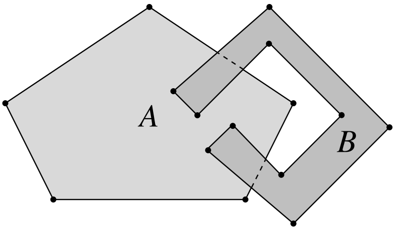
Example



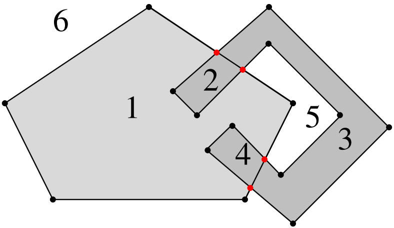
$v = 19, e = 19, f = 5, c = 4$

- The next of edge v_1v_2 is v_2v_3 and both bound face 3.
- The next of edge v_5v_2 is v_2v_4 and both bound face 5.
- The next of edge v_2v_4 is v_4v_2 and both bound face 5.
- This type of edge is called dangling.

Overlay



two polygons



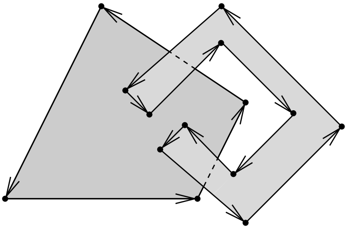
overlay with 6 faces

- Two polygons define a joint subdivision called an overlay.
- Boolean operations yield sets of faces.
  - $A \cup B = \{1, 2, 3, 4\}$
  - $A \cap B = \{2, 4\}$
  - $A - B = \{1\}$
  - $B - A = \{3\}$
- The overlay of two or more subdivisions is defined analogously.

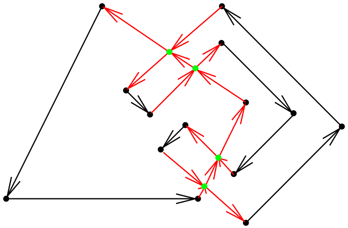
Overlay Algorithm

1. Copy the input edges into a new subdivision.
2. Compute the next fields of the new edges.
3. Split the edges at their intersection points.
4. Construct the faces.

Computing the New Edges



2



3

2. Compute the next fields of the new edges.

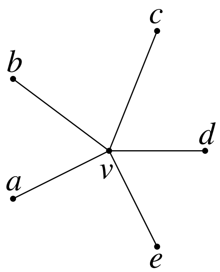
   Arrows indicate edges with interiors on left.

3. Split the edges at their intersection points.

   Red arrows indicate sub-edges of these edges.

   Green dots are new vertices.

Computing the Next Fields



- What if the input subdivisions share a vertex v?
- The output v can be incident on any number of edges.
- Sort the edges clockwise around v.
- The next of an incoming edge is the following outgoing edge.
- Example: the next of av is vb.

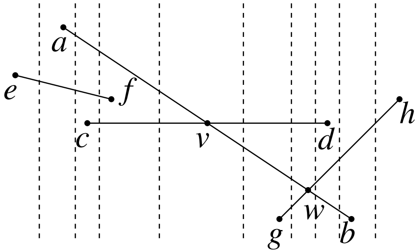
Intersection Point Computation

- Input: n edges.
- Output: m intersection points.
- Worst case: $m = O(n^2)$, so running time is $O(n^2)$.
- Brute force algorithm: test every pair of edges.
- Sweep algorithm: test pairs of edges that see each other.
- Output sensitive: $O((n+m)\log n)$.
- Complicated optimal algorithm: $O(n \log n + m)$.

Sweep Algorithm Input

- We assume that the input is not degenerate.
- There are three types of degenerate input.
  - A vertex lies on an edge.
  - Three or more edges intersect at a common point.
  - Two collinear edges overlap.
- A fourth type is specific to the sweep algorithm.
  - An edge is parallel to the sweep line.
- Degeneracy can be prevented by input perturbation.
- The textbook handles the first two types of degeneracy.
- The algorithm and the analysis are more tedious.

Sweep Algorithm

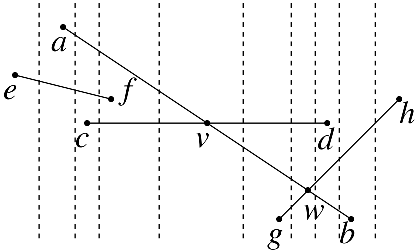


- Sweep a vertical line through the edges.
- Track the vertical order of the edges that intersect the sweep.
  1. (ef) from e.
  2. (ef, ab) from a.
  3. (cd, ef, ab) from c.
  4. (cd, ab) from f.
  5. (ab, cd) from v.
- Check incident edges for intersection.
  - Check ef and ab at a; no intersection.
  - Check cd and ab at f; compute v.
  - Check gh and ab at v; compute w.

Implementation

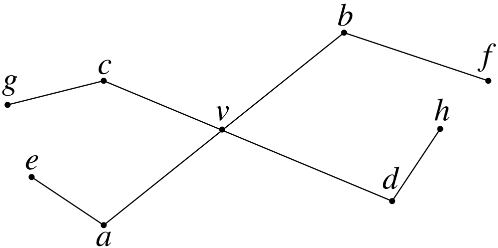
- Sweep list: a balanced tree of edges in vertical order.
- Initialize an empty sweep list.
- Events: start endpoint, end endpoint, intersection point.
- Two edges intersect at an intersection point.
- Initialize a priority queue with the edge endpoints.
- Queue order is x-order with ends before starts for equal x.
- Process the next event until the queue is empty.
  - Add and remove edges or swap two edges.
  - Check the newly adjacent pairs of edges for intersection.
  - Add the intersection points to the queue.
- Two edges can become incident many times, but only one intersection event should be created.

Computing the Vertical Order



- The vertical order is computed at the left endpoint a of an edge ab with respect to an edge ef with $e_x \leq a_x < f_x$.
- disjoint edges: ab is above ef if aef is a left turn.
- shared tail: vd is above vb if dvb is a left turn.
- The order flips when edges swap.

Edge Split



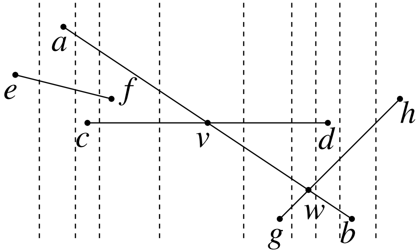
Edges *ab* and *cd* intersect at the point *v*.

A *v* vertex with four incident edges is created.

The twin fields of *ab*, *ba*, *cd*, and *dc* are set to these edges yielding edges *av*, *bv*, *cv*, *dv*, and their twins.

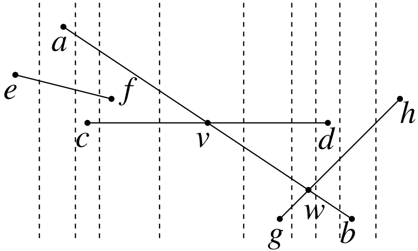
The next fields of these eight edges are set, e.g. the next of *av* is *vc* and the next of *vc* is *cg*.

Sweep Update



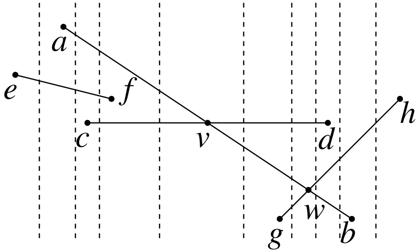
- The edges that are split are in the event queue.
- The edge split invalidates them.
- Example: *ab* becomes *av*, which does not intersect *gh*.
- How is this handled?

Sweep Update



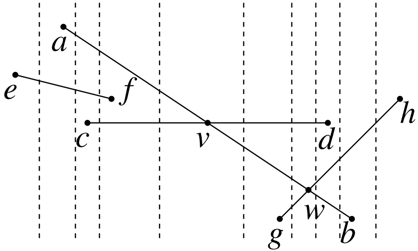
- The edges that are split are in the event queue.
- The edge split invalidates them.
- Example: ab becomes av, which does not intersect gh.
- How is this handled?
- Option 1: update the event queue.

Sweep Update



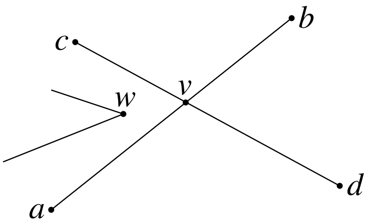
- The edges that are split are in the event queue.
- The edge split invalidates them.
- Example: ab becomes av, which does not intersect gh.
- How is this handled?
- Option 1: update the event queue.
- Option 2: record the split points and split after the sweep.

Sweep Update



- The edges that are split are in the event queue.
- The edge split invalidates them.
- Example: ab becomes av, which does not intersect gh.
- How is this handled?
- Option 1: update the event queue.
- Option 2: record the split points and split after the sweep.
- Option 3: store the twins in the event queue, e.g. ba and dc.

Correctness



The sweep algorithm finds all the edge intersection points.

Proof

Suppose edges ab and cd intersect at v and every intersection point u with $u_x < v_x$ is found.

The sweep order is correct after the last event w with $w_x < v_x$.

Edges ab and cd are adjacent in the sweep after the w update because they are adjacent at v_x and no events intervene.

The intersection point v is found at w if not earlier.

Complexity

- There are $2n$ endpoint events and m intersection point events.
- Processing an event takes $O(\log n)$ time.
  - $O(\log n)$ to update the queue.
  - $O(\log n)$ to update the sweep.
  - $O(1)$ to check at most two newly adjacent pairs.
- The running time is $O((n + m) \log n)$.

Face Construction

1. Form the edge loops.
2. Classify each loop as an outer or an inner boundary.
3. Each outer boundary defines a bounded face.
4. Assign the inner boundaries to their faces.

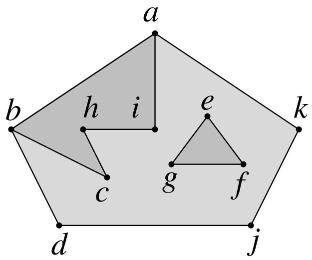
Edge Loops

An edge loop is represented by one of its edges.

Algorithm

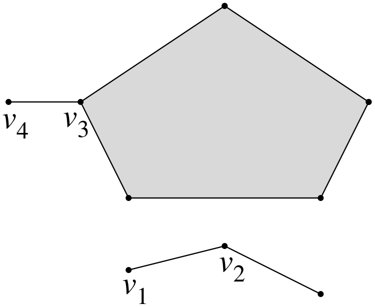
1. mark the edges as not traversed
2. visit every edge e
3. if e is not traversed
   3.1 add e to the output
   3.2 mark the edges in the e loop as traversed

Loop Classification



- A loop is as an outer or an inner boundary.
- It is outer if a left turn occurs at its leftmost vertex.
- Examples
  - *abchi* is outer because of *b*.
  - *aihcbdjk* is outer because of *b*.
  - *fge* is inner because of *g*.
- Another vertex can give the wrong answer, e.g. *chi*.
- The vertex can be extremal in any direction.

Dangling Edges



- Loops with dangling leftmost edges are inner.
- The left turn is degenerate (an identity).

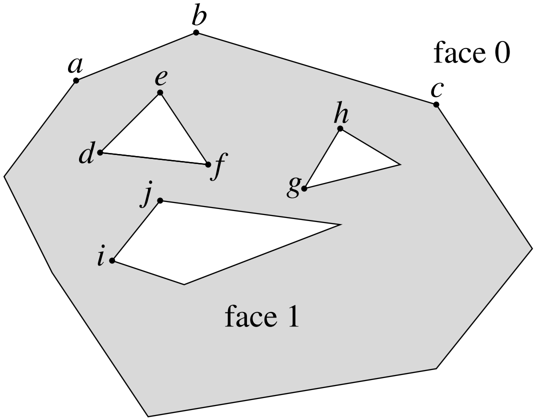
Face Construction Algorithm

1. Construct an unbounded face with a null outer boundary.
2. Construct a face f for each outer loop representative e.
   2.1 Set the outer boundary of f to e.
   2.2 Set the face of each loop edge to f.
3. For each inner loop representative e
   3.1 Find the face f that contains e.
   3.2 Add e to the inner boundary list of f.
   3.3 Set the face of each loop edge to f.

Face Finding

- The sweep algorithm records the edge above each vertex.
- Visit the inner boundaries in decreasing highest vertex order.
- A boundary belongs to the face of the edge above its vertex.
- The face pointer of that edge has already been set.

Example



1. Assign the *ab* boundary to face 0 because no edge is above.
2. Assign the *de* boundary to face 1 because *ba* is above.
3. Assign the *gh* boundary to face 1 because *cb* is above.
4. Assign the *ij* boundary to face 1 because *fd* is above.