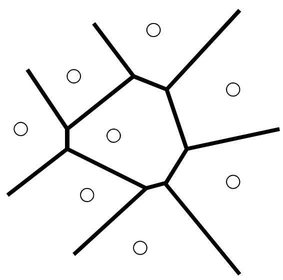
Computational Geometry

Chapter 7

Voronoi Diagrams

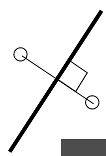
Definition

- **Input:** A set *S* of *n* point locations (*sites*) in the plane.
- **Output:** Vor(*S*), a planar subdivision into cells. Each cell contains all the points for which a certain site is the closest.



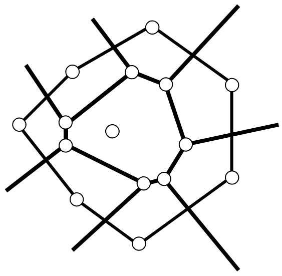
Application: Nearest-neighbor queries (by point location in the diagram).

We use the regular Euclidean distance function. Edges of the diagram are portions of *bisectors*. The bisector of two points is a line.



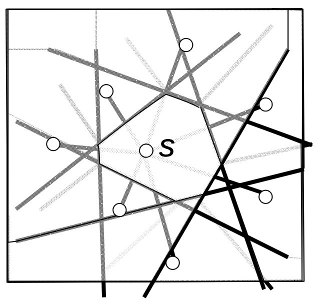
Simple Properties

- General position assumption: No four sites are cocircular.
- The Voronoi diagram is a planar graph, whose vertices are equidistant from three sites, and edges equidistant from two sites.
- Question: What would happen if more than three sites were cocircular?
- Observation: The convex hull of the sites is of those who have an unbounded cell in the diagram. (Prove!)



Naive Construction

- For a site *s*, construct the bisectors between *s* and all the other sites.
- The Voronoi cell of *s* is the intersection of all the half-planes defined by the bisectors.
- Time complexity:
  $O(n \log n)$ for *each* cell (why?);
  $O(n^2 \log n)$ in total.

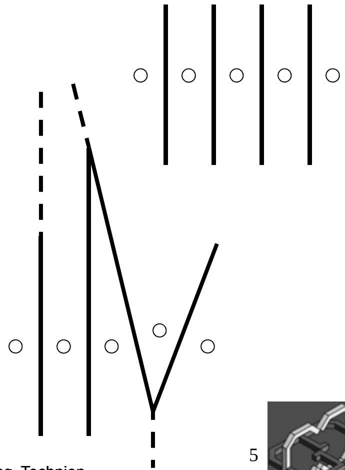


- **Corollary:** Each cell in the Voronoi diagram is a convex polygon, possibly unbounded.

Graph Property

Theorem:

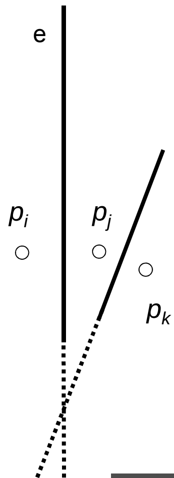
1. If all the sites are collinear, then the Voronoi diagram consists of n-1 parallel lines only.
2. Otherwise, the diagram is a **connected** planar graph, in which all the edges are line segments or rays (half lines).



Graph Property (Cont.)

Proof:

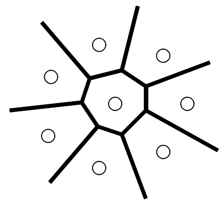
1. Trivial.
2. Assume, on the contrary, that there is a straight line e, the bisector of p_i and p_j, in the diagram. Since not all the points are collinear, there exists a point p_k that is not on p_ip_j. Thus, the bisector of p_k and p_j is not parallel to e. Obviously, the portion of e that is closer to p_k than to p_j cannot be in the diagram on the boundary of p_j's cell, which is a contradiction.



Question: Why connected?

Complexity

- One cell can have the maximum possible complexity n-1, but not all the cells can have it simultaneously.



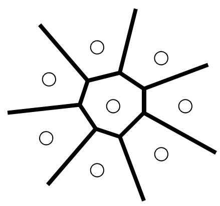
- Theorem:

  |S|=n. Let V,E,F be the numbers of vertices, edges, and faces, respectively, of Vor(S). Then,
  - $V \le 2n-5$ (equality in the worst case);
  - $E \le 3n-6$ (equality in the worst case); and
  - $F = n$.

Complexity Analysis

Proof:

- We complete the graph to a planar graph by connecting all rays to a "vertex at infinity". Thus, V is increased by one, while E and F are unaltered.
- $F = n$. (Why?)
- By Euler's formula: $(V+1)-E+n = 2$.
- Let X be the number of edge-vertex coincidences (that is, the sum of ranks).



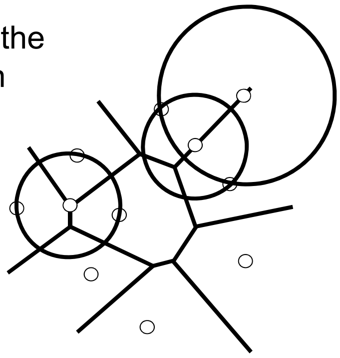
- Then:
  - X = 2E
  - X $\ge 3(V+1) \;\Rightarrow\; 3(V+1) \le 2E$

    $\Rightarrow\; E \ge 3V/2+3/2,\;\; V \le 2E/3-1$
  - $V = E-n+1 \ge 3V/2+3/2-n+1 = 3V/2-n+5/2 \;\Rightarrow\; V \le 2n-5$
  - $E = V+n-1 \le 2E/3-1+n-1 = 2E/3+n-2 \;\Rightarrow\; E \le 3n-6$
- Justify why equalities in the worst case.

Main Properties

Theorem:

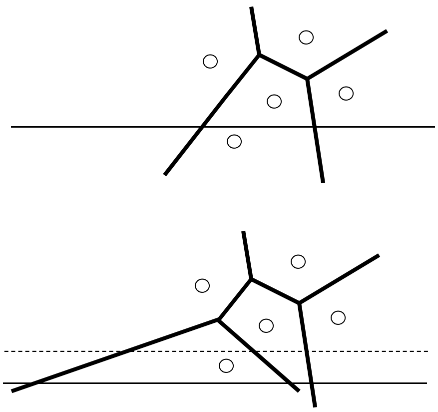
1. A vertex of a Voronoi diagram is the center of an empty circle passing through three (or more) sites.
2. Every point on an edge of the diagram is the center of an empty circle passing through two sites.



Proof: By definition.

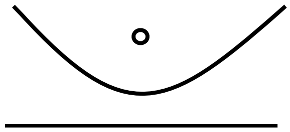
A Plane-Sweep (Fortune's) Algorithm

- Main difficulty: If we maintain the diagram of all points seen in the sweep so far, this can change as new points are swept.

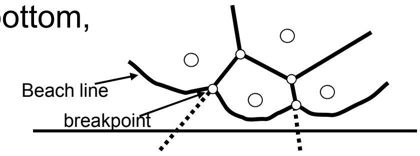


The "Beach Line"

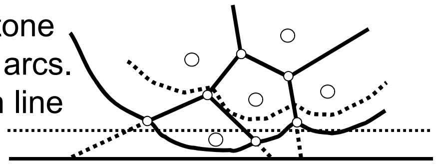
- The bisector of a point and a line is a *parabola*. The *beach line* (front) is the lower envelope of all the parabolas already seen.



- Sweep the plane from top to bottom, maintaining the invariant: The Voronoi diagram is correct **above** the beach line.



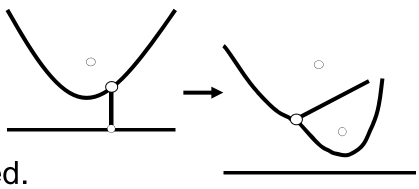
- The beach line is an *x*-monotone curve consisting of parabolic arcs. The *breakpoints* of the beach line lie on the Voronoi edges of the final diagram.



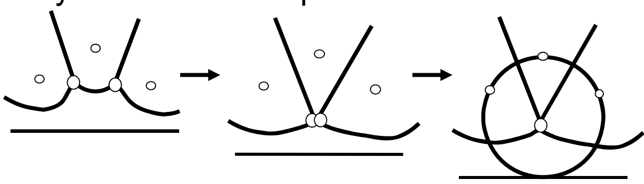
Events

Possible events:

- **Site** event: The sweep line meets a site. A vertical edge (skinny parabola) connects the point and the front arc above it. These events are predetermined.



- **Vertex** event: An existing arc of the front line shrinks to a point and disappears, creating a Voronoi vertex. Generated by **consecutive** triples of sites. These events are detected on-line.



Site-Event Properties

- A new edge appears in the diagram in a site event.
- A site event creates a new arc on the front (beach) line, and its endpoints slide along edges of the Voronoi diagram.
- As long as the line is swept downward, the front moves down too, but the break points may move up!

Vertex-Event Properties

- **Theorem**:

  Let p_i, p_j, and p_k be three sites causing a vertex event q when the swept line is in position ℓ.

  1. The sites p_i, p_j, and p_k are distinct.
  2. The sites p_i, p_j, and p_k are cocircular, defining a circle centered at q and tangent to ℓ.
  3. An arc disappears from the front only in a vertex event of three consecutive arcs of the front line.

  Proof:

  Exercise.

- A new edge may appear in the diagram in a vertex event.

Event Summary

- **Site** event:
  An arc appears in the front line, and its endpoints are sliding along an edge of the Voronoi diagram.

- **Vertex** (circle) event:
  An arc disappears from the front line, and the degenerating point is a vertex of the Voronoi diagram.

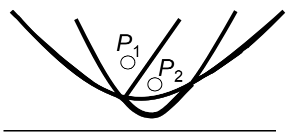
Center for Graphics and Geometric Computing, Technion

15

Complexity of the Front Line

- Theorem: The complexity of the beach line is O(n).

- Proof:
  - The first site generates one parabola.
  - Arcs may disappear from the front line, but this only helps.
  - Each other site splits one parabolic arc into three arcs.
  - Total: 1+ (n-1)(3-1) = 2n-1 = O(n)
  - $\Omega(n)$ in the worst case.



Beach line:
$P_1 \Rightarrow P_1, P_2, P_1$

Center for Graphics and Geometric Computing, Technion

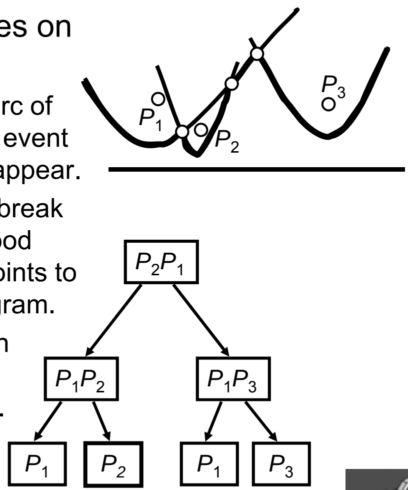
16

Diagram Data Structure

- Doubly-connected edge list (DCEL), allowing half-lines (rays).
- At the termination of the algorithm we will surround the “action area” by a bounding box and trim all rays.

Beach Line Data Structure

- A balanced binary tree of sites on the front line:
  - A leaf represents a parabolic arc of the front. It points to the circle event in which this arc will (may) disappear.
  - An internal node represents a break point in the front (a neighborhood relation of two front arcs). It points to the respective edge in the diagram.
  - A site may appear in more than one leaf (e.g., P_1 in the figure), but there is no asymptotic loss.
  - The tree supports search, insertion, and deletion in logarithmic time.

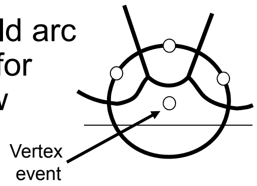


Event Queue

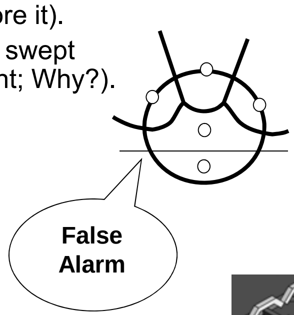
- Ordered according to *y* coordinates.
- Contains events of two types:
  - **Site** event: Key: *y* coordinate of the site.
    Keeps the point (or its ID).
  - **Vertex** event: Key: *y* coordinate of the **lowest** point of the circle (the event's timestamp).
    Keeps the front arc that will **disappear**.
- Note: Events are added **and may be deleted.**

Creating a Vertex Event

- Each time a new arc appears (or an old arc disappears) on the beach line, check for possible vertex events among the new consecutive triples of arcs.



- The circle must:
  - Intersect the sweep line (otherwise ignore it).
  - Contain no other points lying above the swept line. (Otherwise it would not be an event; Why?).



- **Note:** A vertex event may be a **false alarm**, in case the circle contains points below the sweep line. It will be deleted later in the processing of a site event.

Handling Events

- Site event:
  - Create an edge in the Voronoi diagram.
  - Split a front arc vertically above the site.
  - Delete from the queue circle events one of whose three generators have been eliminated.
  - Add (if needed) new vertex events to the queue.

- Vertex event:
  - Create a vertex in the Voronoi diagram.
  - If needed, create an edge in the Voronoi diagram.
  - Delete the respective arc from the front.
  - Delete (if needed) vertex events from the queue.
  - Add new vertex events to the queue.

Complexity Analysis

- Initialization: O(n log n).
- Number of events: O(n).
  Each event requires O(log n) time.
- **Note:** A constant number of false-alarm vertex events are created and deleted only when another real event occurs (why?), so their number is also O(n).
- Total time: O(n log n). Question: Why is it optimal?

- Space: Both the front line structure and the Voronoi diagram consume linear space – $\Theta(n)$.