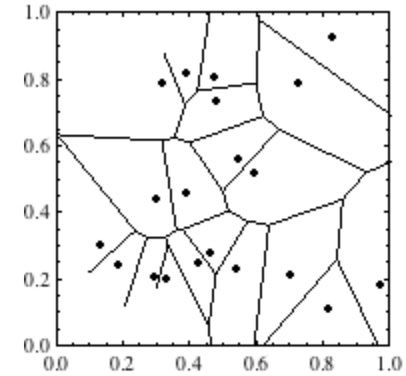
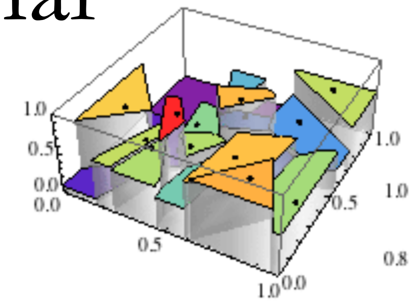
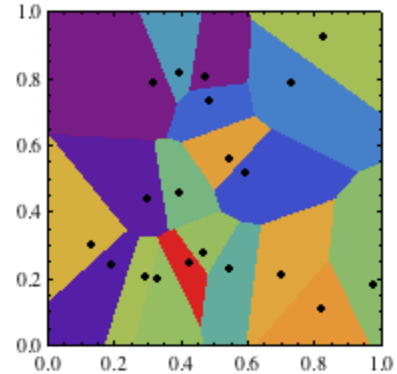
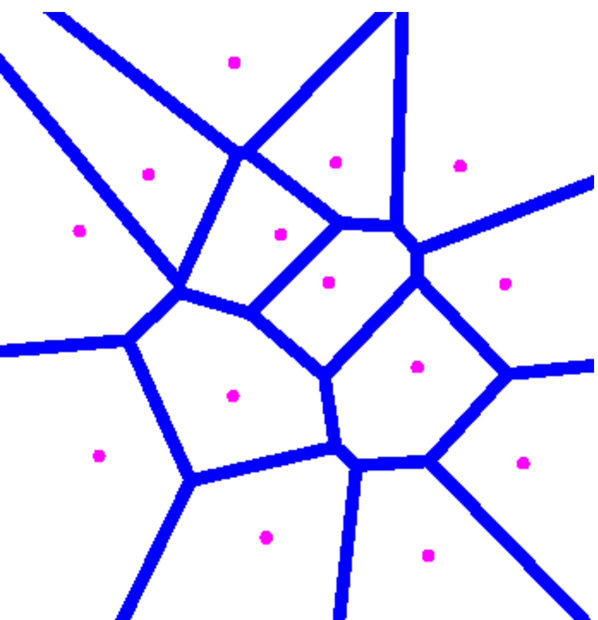
Flavor of Computational Geometry





Voronoi Diagrams





Shireen Y. Elhabian

Aly A. Farag

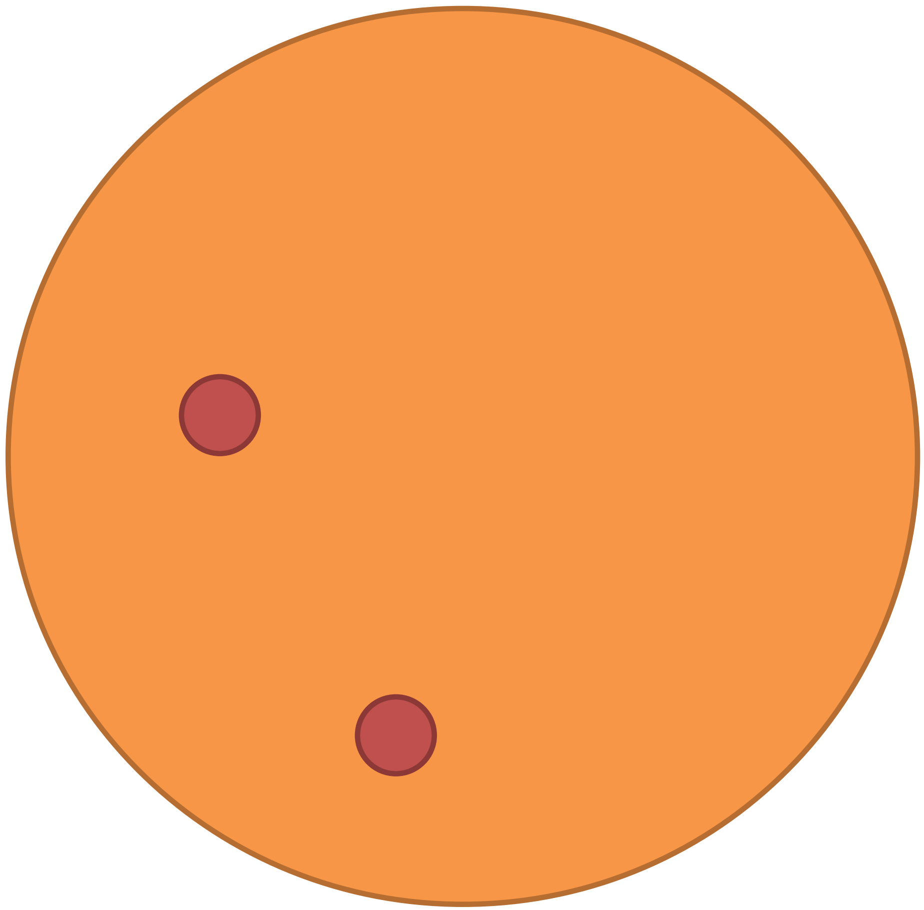
University of Louisville

March 2010

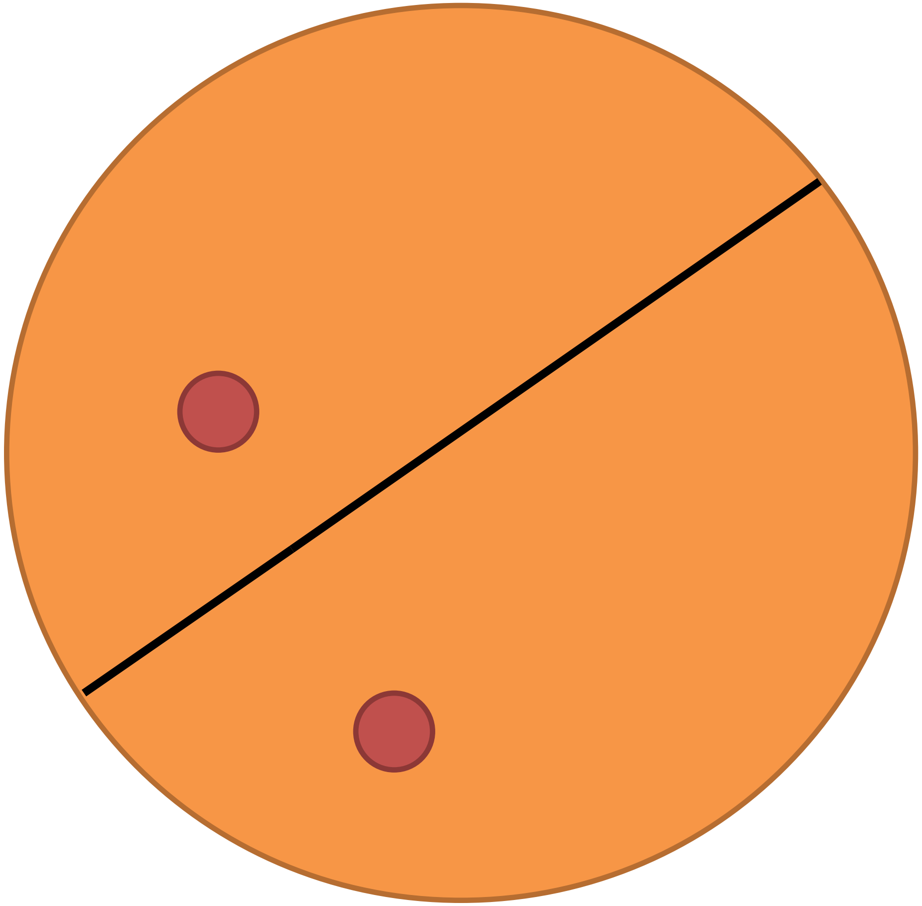

ClipArtOf.com

Pepperoni Sparse Pizzas
Olive Sparse Pizzas

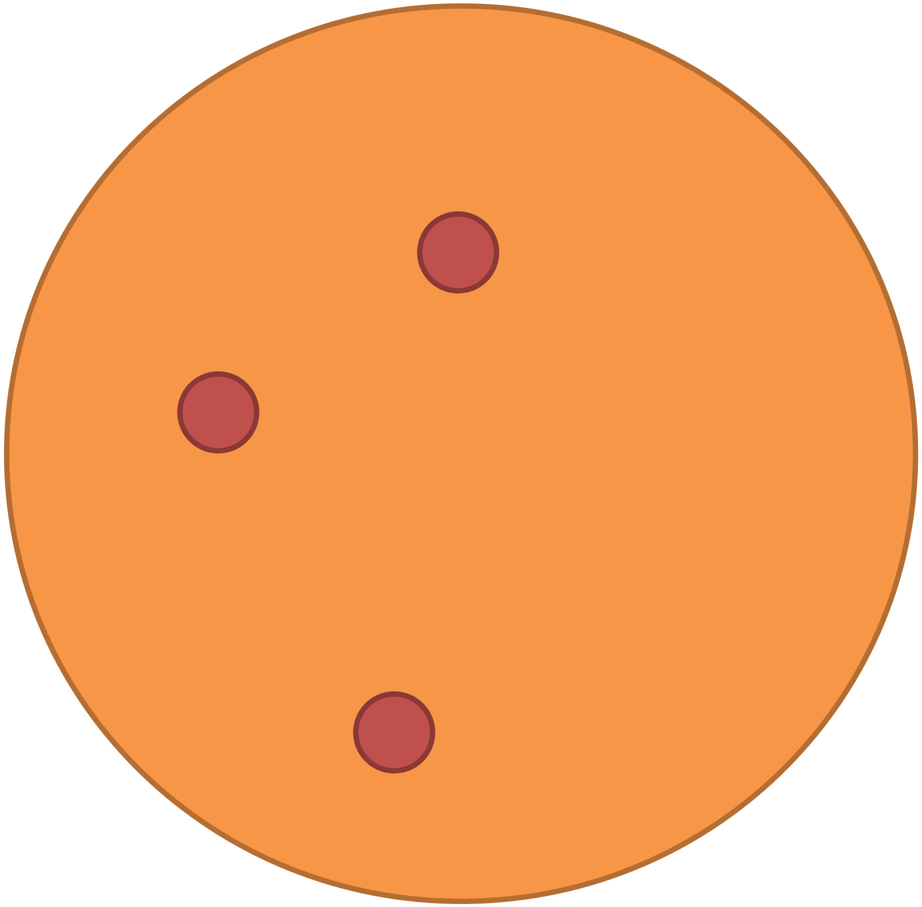
Just Two Pepperonis



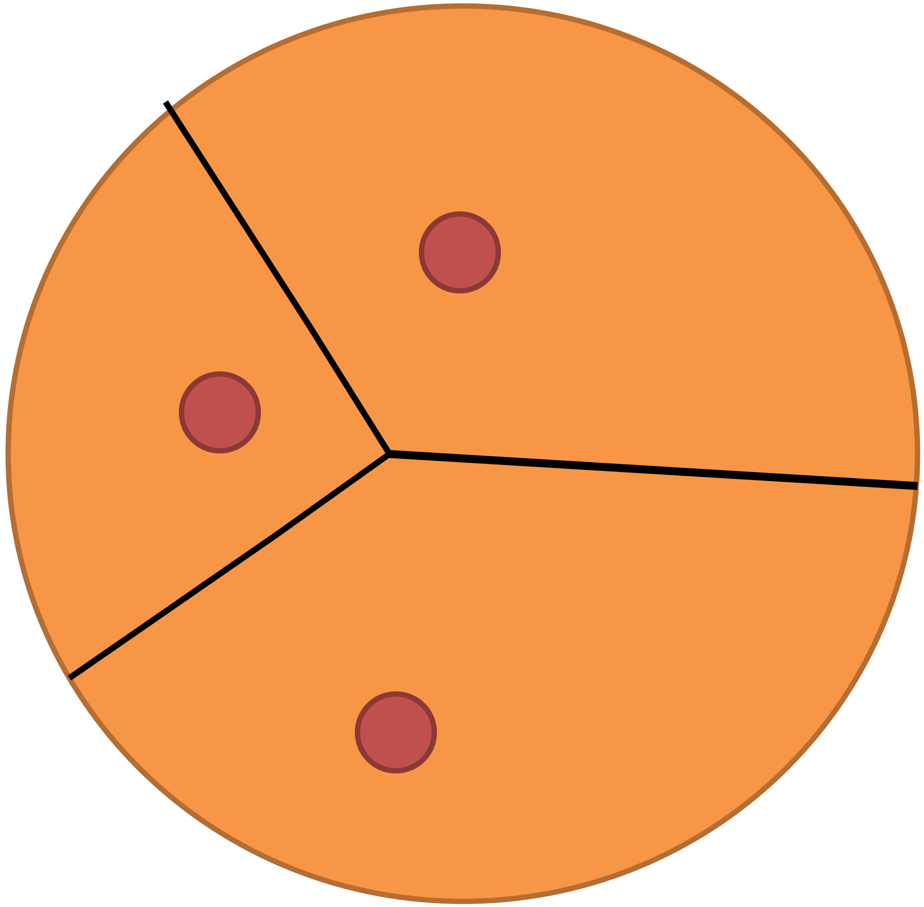
A person gets the part of the pizza closest to their pepperoni.



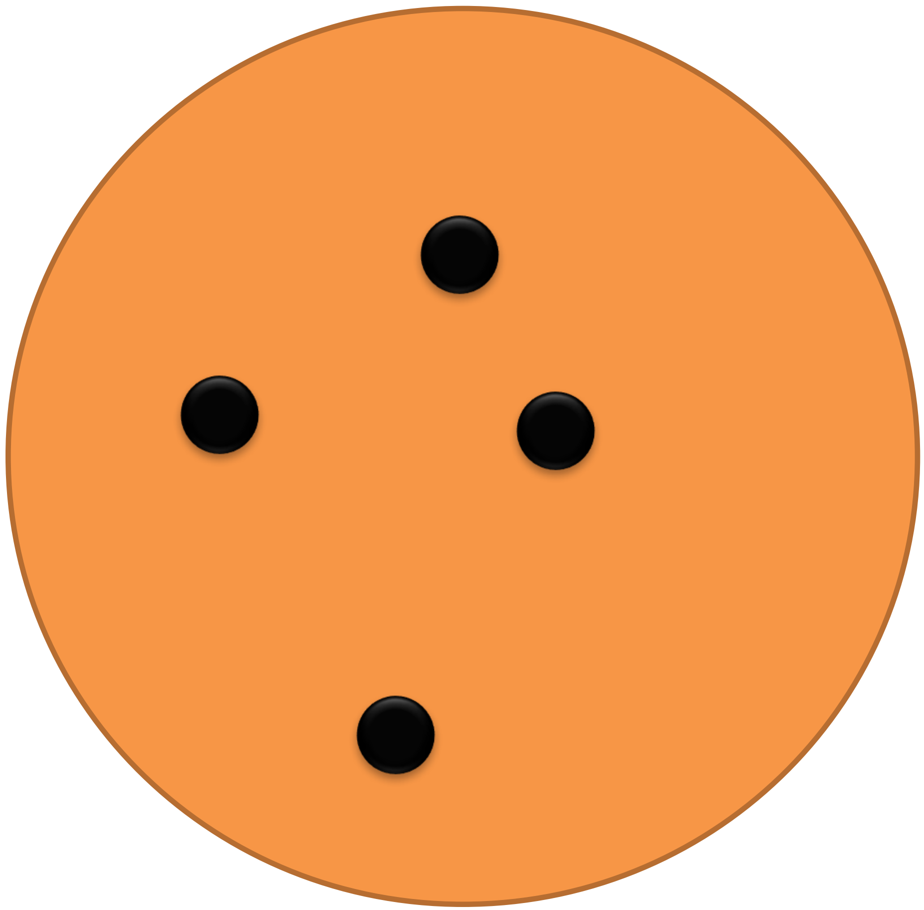
Pizza with 3 Pepperonis



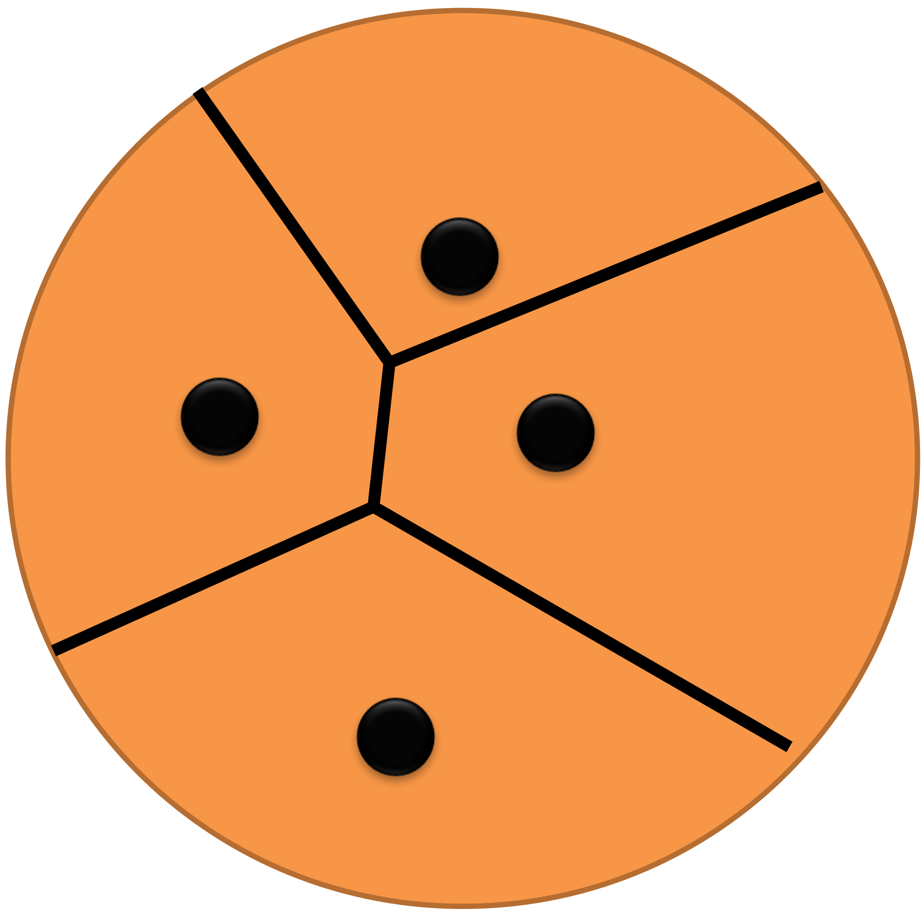
Each person gets the part of the pizza closest to their pepperoni.



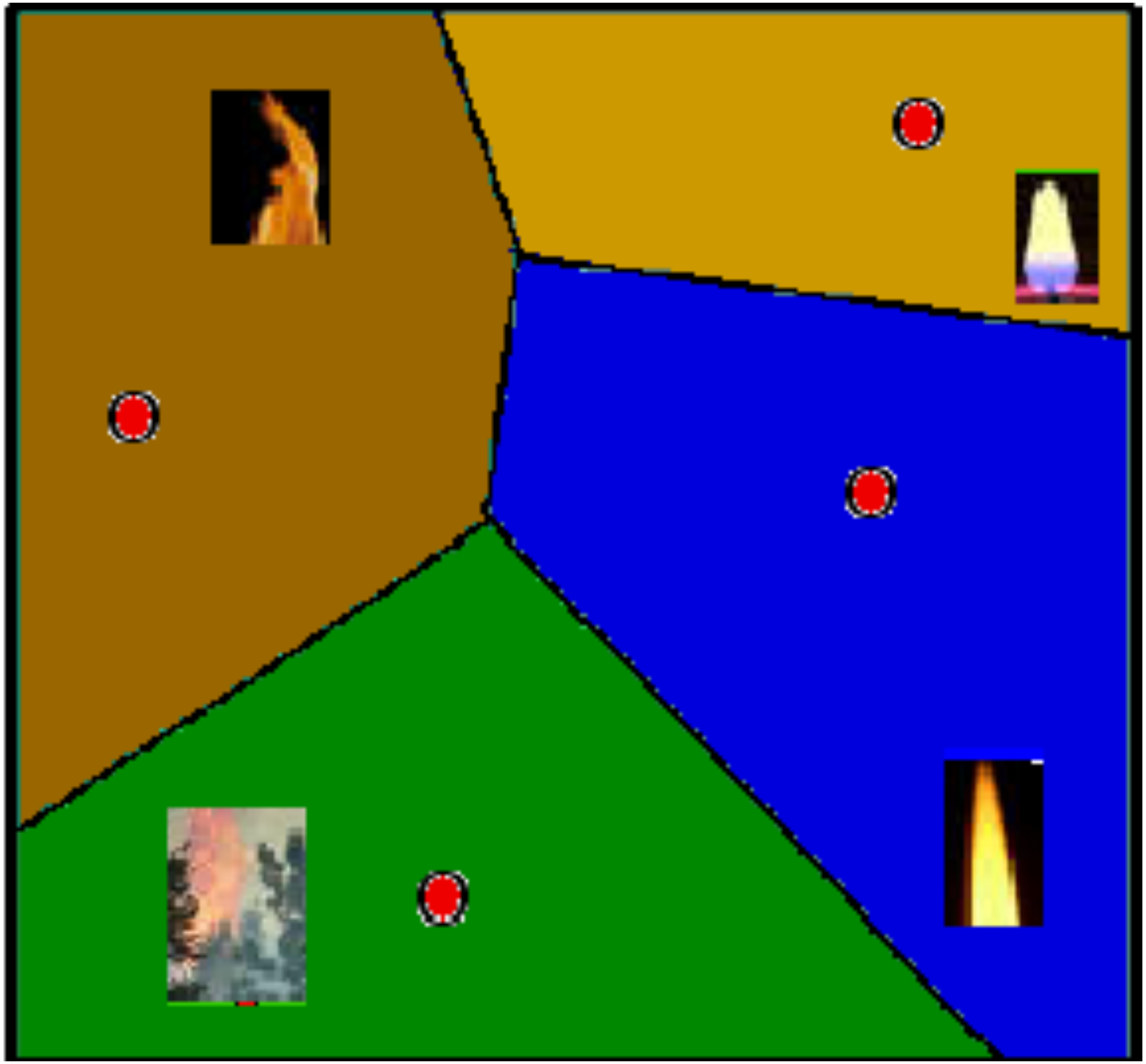
Four olives



Each person gets the part of the pizza closest to their olive.

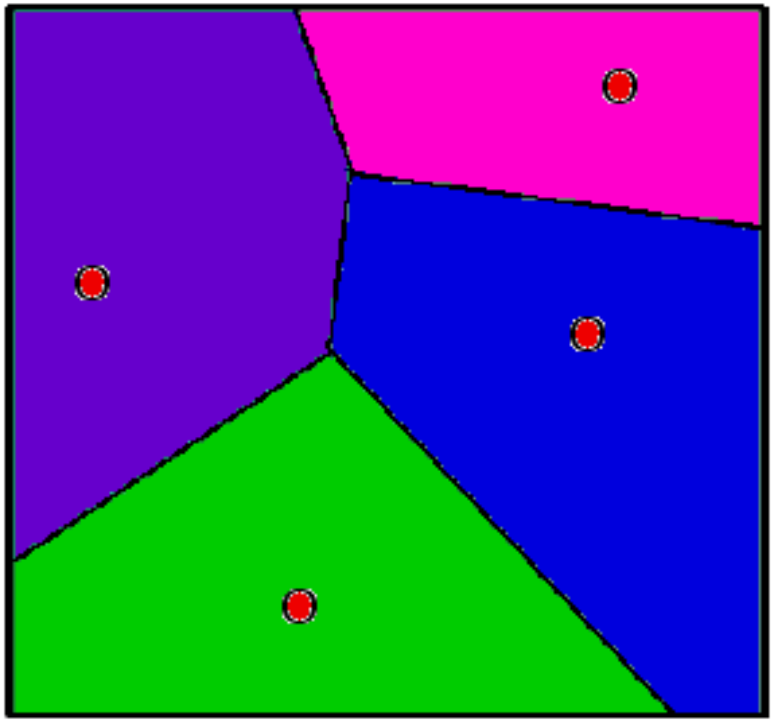


The State Forest has four fire towers. We want to partition the State Forest into four regions so that the individuals in the fire towers are searching for fires in the region nearest to their own tower.



A ***Voronoi diagram***
represents
the region of influence
around each of a
given set of sites.

Terminology



- Voronoi site - red dots
- Voronoi diagram - edges of polygonal regions
- Voronoi point - corners of polygonal regions

Georgy Voronoy

- 1868-1908
- Ukraine-Poland
- Theory of numbers:
  - Continued fractions
  - Roots of an irreducible cubic equation
- Winner of the Bunyakovsky Prize from the St. Petersburg Academy of Sciences.
- He defined the Voronoi diagram

The Post Office Problem

Problem Statement

- Suppose we want to open a new branch for a supermarket chain at a certain location.
- To predict whether the new branch will be profitable, we must estimate the number of customers it will attract.
- Thus we need to model the behavior of our potential customers: how do people decide where to do their shopping?
- A similar question arises in social geography, when studying the economic activities in a country: what is the trading area of certain cities?

Post Office: What is the area of service?

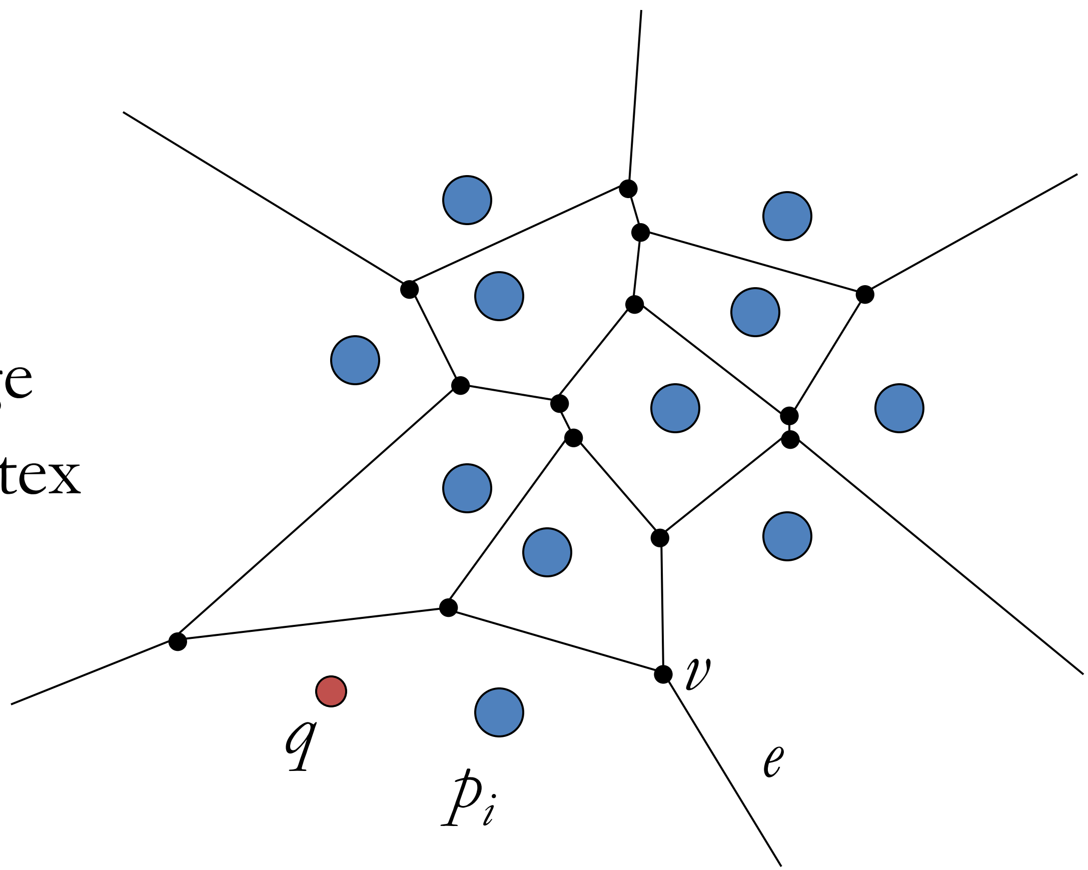
Definition 26: Post Office Problem: *We have a set of central places—called sites—that provide certain goods or services, and we want to know for each site where the people live who obtain their goods or services from that site. (In computational geometry the sites are traditionally viewed as post offices where customers want to post their letters)*

p_i : site points

q : free point

e : Voronoi edge

v : Voronoi vertex



Assumptions

- The price of a particular good or service is the same at every site;
- The cost of acquiring the good or service is equal to the price plus the cost of transportation to the site;
- The cost of transportation to a site equals the Euclidean distance to the site times a fixed price per unit distance;
- Consumers try to minimize the cost of acquiring the good or service.

Geometric Interpretation

- The assumptions in the model induce a subdivision of the total area under consideration into regions, the trading areas of the sites, such that the people who live in the same region all go to the same site.

- Our assumptions imply that people simply get their goods at the nearest site, a fairly realistic situation.

Definition 27: *The* **trading area** *for a given site consists of all those points for which that site is closer than any other site. The model where every point is assigned to the nearest site is called the* **Voronoi assignment model**. *The subdivision induced by this model is called the* **Voronoi diagram** *of the set of sites.*

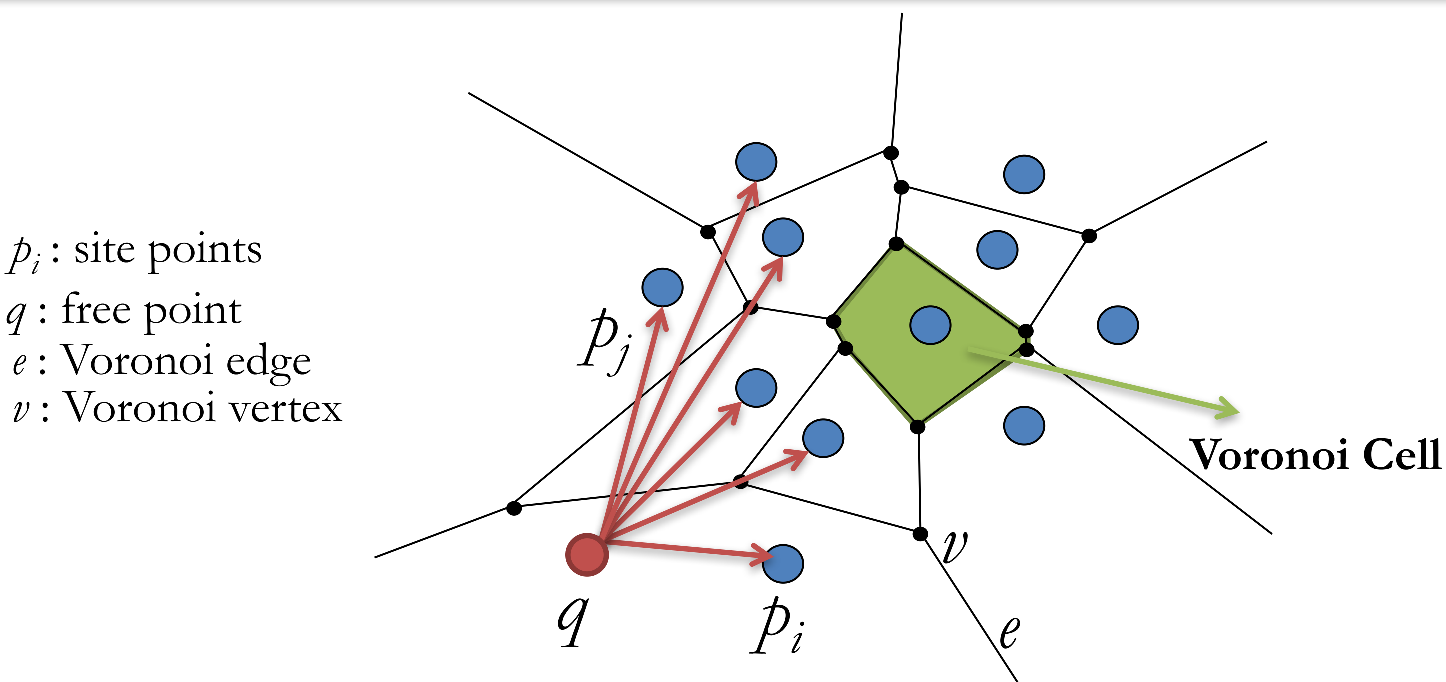
Definitions and Basic Properties

Voronoi Diagram

Definition 28: *Denote the* **Euclidean distance** *between two points* p *and* q *by* $dist(p,q)$ *where in* $\mathbb{R}^2$, *we have:*

$$dist(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Definition 29: *Let* $P = \{p_1, p_2, \ldots, p_n\}$ *be a set of* n *distinct points in the plane; these points are the sites. We define the* **Voronoi diagram** *of* P *as the subdivision of the plane into* n *cells, one for each site in* P, *with the property that a point* q *lies in the cell corresponding to a site* p_i *if and only if* $dist(q,p_i) < dist(q,p_j)$ *for each* $p_j \in P$ *with* $j \neq i$. *We denote the Voronoi diagram of* P *by* $Vor(P)$. *The cell of* $Vor(P)$ *that corresponds to a site* p_i *is denoted* $\mathcal{V}(p_i)$; *we call it the* **Voronoi cell** *of* p_i *which is the trading area of site* p_i

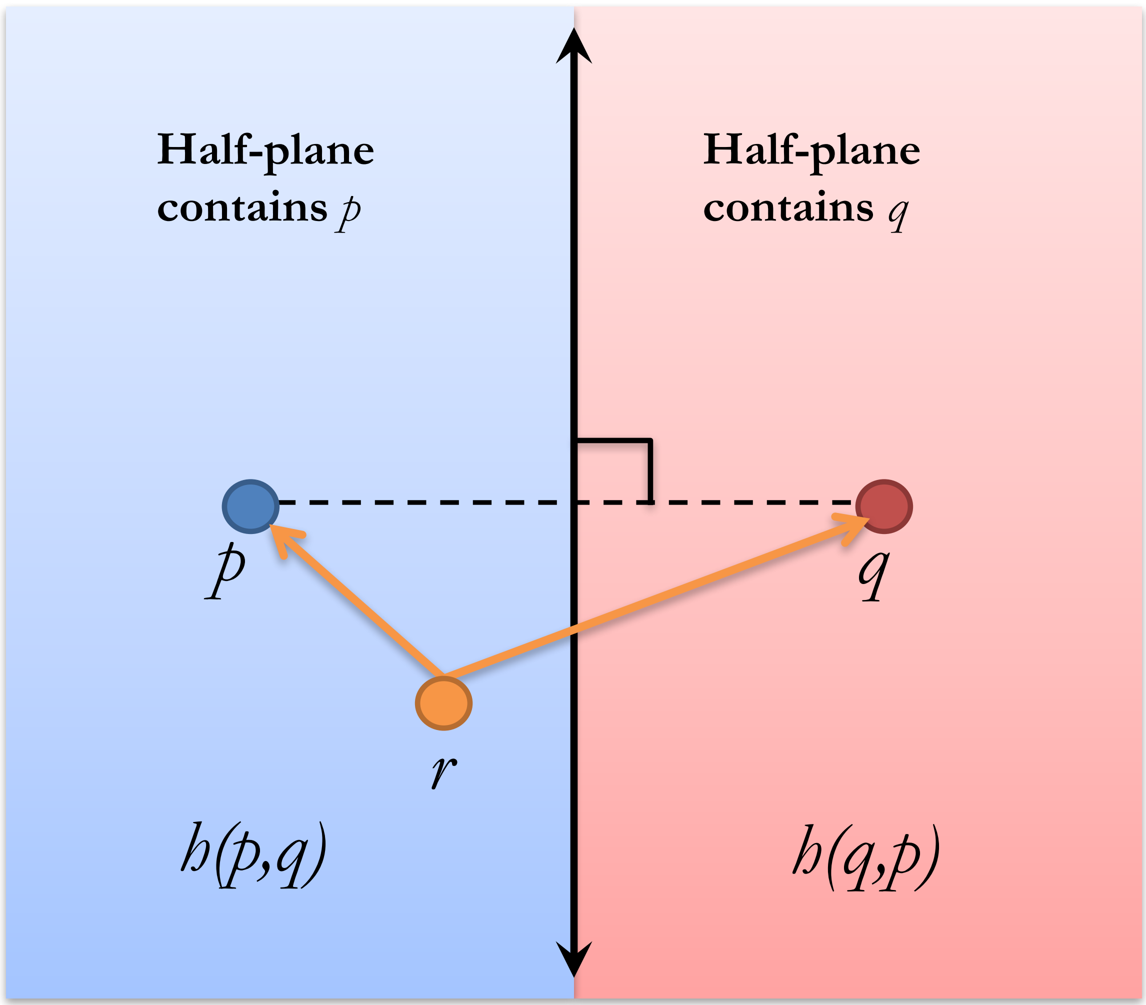


Structure of Voronoi Cell

We now take a closer look at the Voronoi diagram. First we study the structure of a single Voronoi cell.

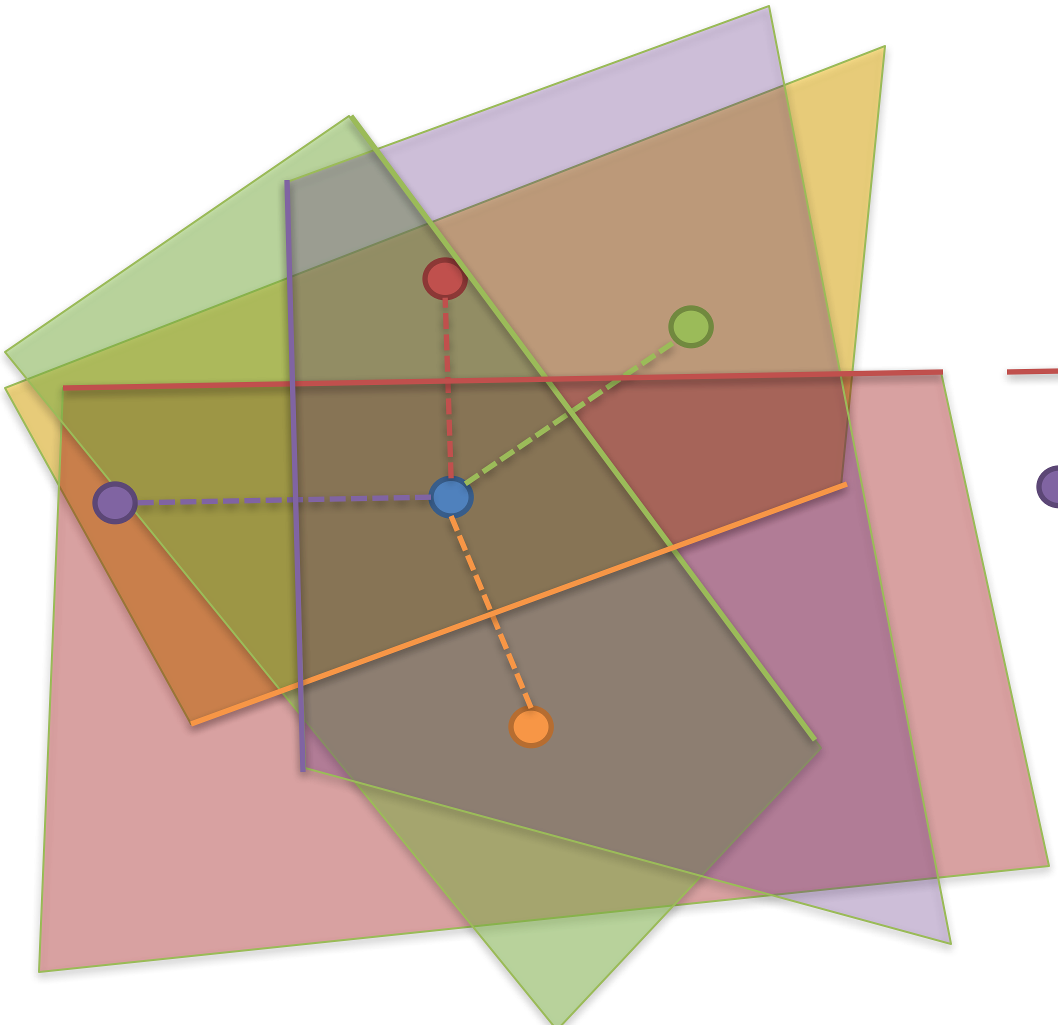
Definition 30: *For two points* p *and* q *in the plane we define the* **bisector** *of* p *and* q *as the perpendicular bisector of the line segment* pq*. This bisector splits the plane into two half-planes. We denote the open half-plane that contains* p *by* $h(p,q)$ *and the open half-plane that contains* q *by* $h(q,p)$.

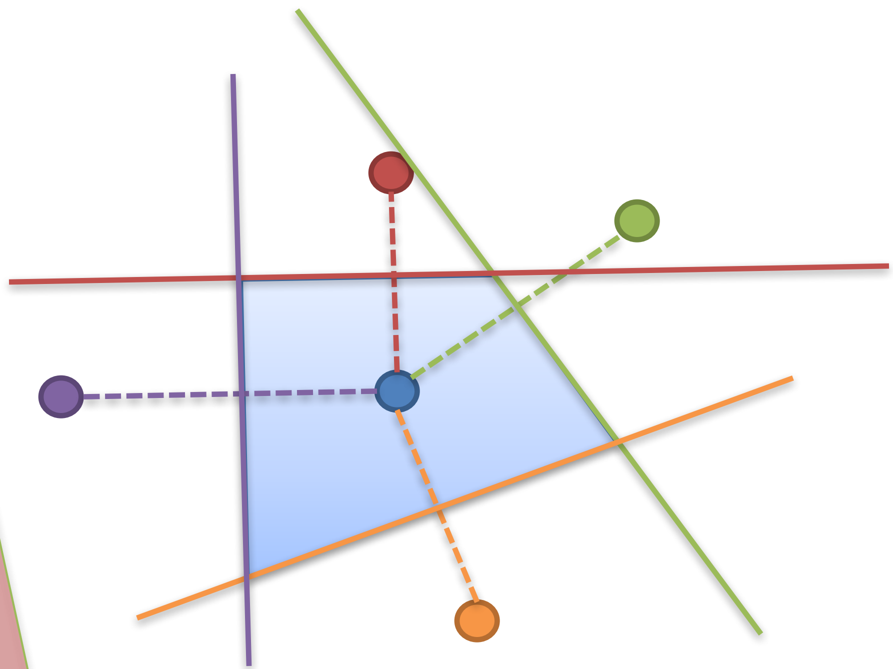
Notice that $r \in h(p,q)$ if and only if $dist(r,p) < dist(r,q)$.



$$\boxed{\textbf{Observation 1: } \mathcal{V}(p_i) = \bigcap_{\substack{1 \le j \le n \\ j \ne i}} h(p_i, p_j)}$$

Thus $\mathcal{V}(p_i)$ is the intersection of $n - 1$ half-planes and, hence, a (possibly unbounded) open convex polygonal region bounded by at most $n - 1$ vertices and at most $n - 1$ edges.





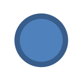
Structure of Voronoi Diagram

What does the complete Voronoi diagram look like?

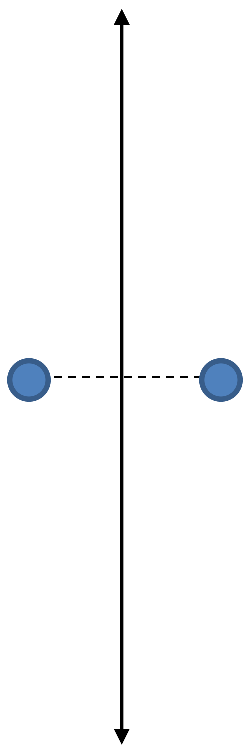
- We just saw that each cell of the diagram is the intersection of a number of half-planes.
- So the Voronoi diagram is a planar subdivision whose edges are straight.
- Some edges are line segments (having starting and ending points) and others are half-lines (only having starting point).
- Unless all sites are collinear there will be no edges that are full lines.

Theorem 8: *Let P be a set of n point sites in the plane. If all the sites are collinear then $Vor(P)$ consists of $n-1$ parallel lines. Otherwise, $Vor(P)$ is connected and its edges are either segments or half-lines.*

Voronoi Diagram Example: 1 site

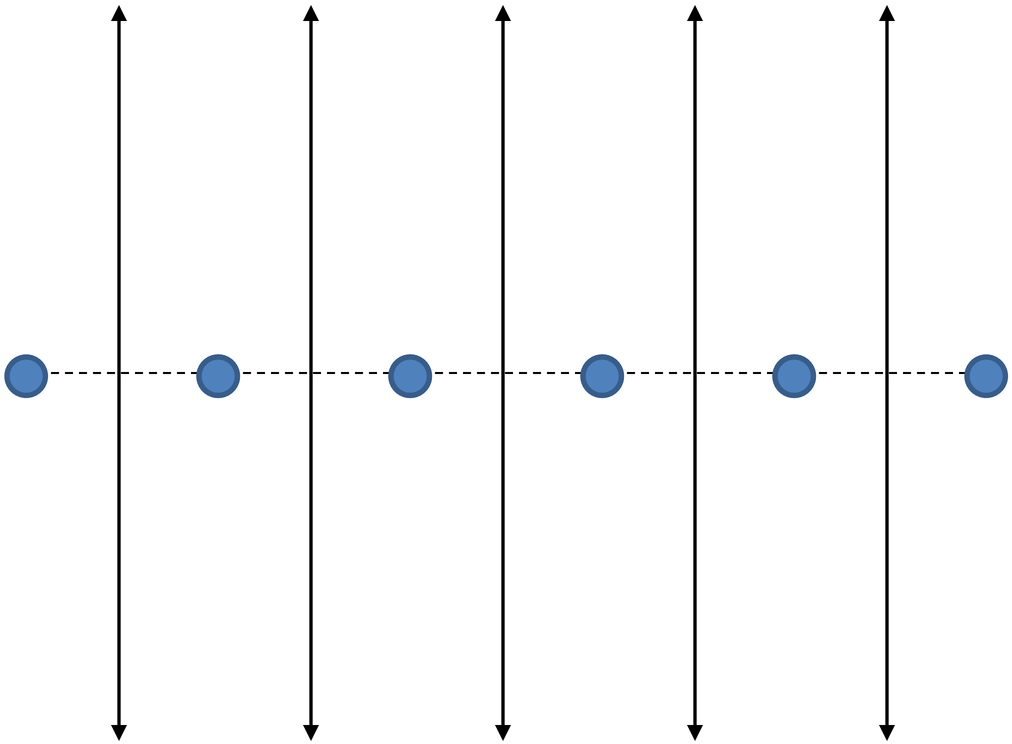


Two sites form a perpendicular bisector



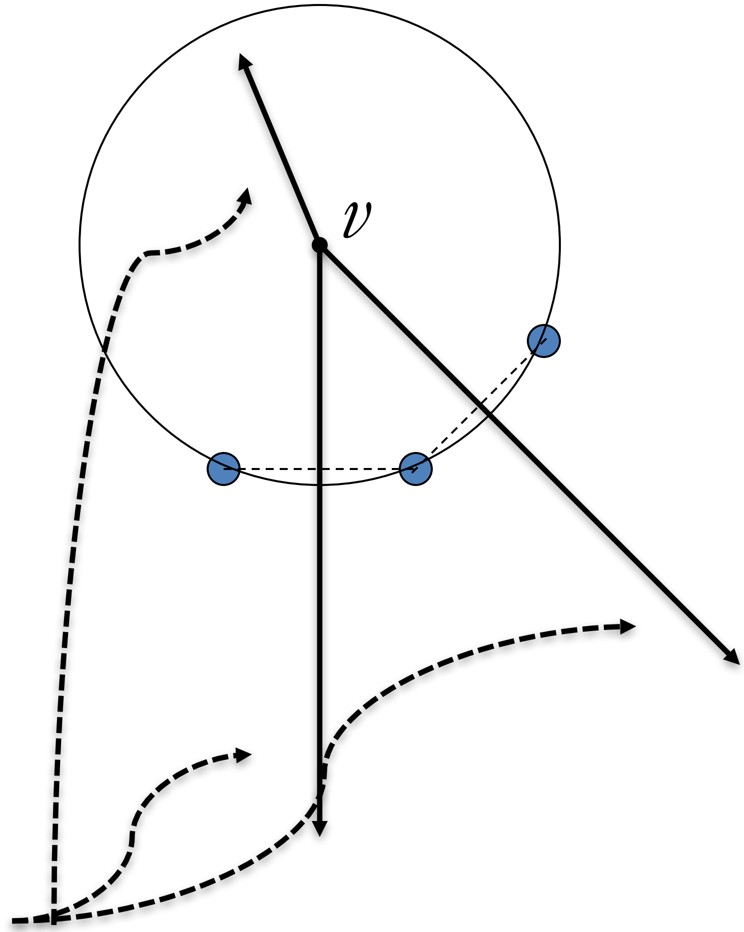
Voronoi Diagram is a line that extends infinitely in both directions, and the two half planes on either side.

Collinear sites form a series of parallel lines



Non-collinear sites form Voronoi half lines that meet at a vertex

A vertex has degree ≥ 3



A Voronoi vertex is the center of an *empty* circle touching 3 or more sites.

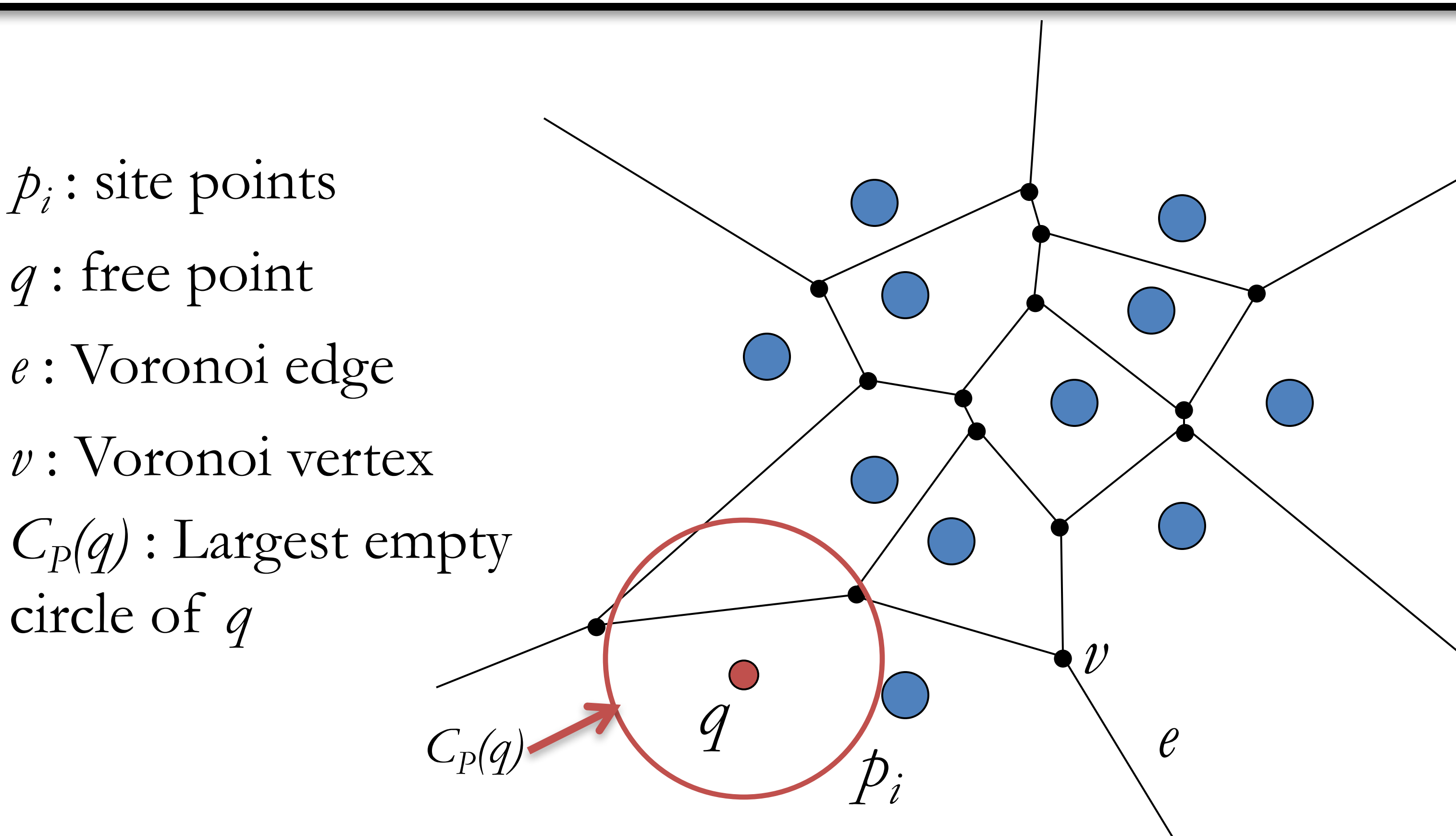
Theorem 9: *For* $n \geq 3$, *the number of vertices in the Voronoi diagram of a set of* n *point sites in the plane is at most* $2n - 5$ *and the number of edges is at most* $3n - 6$.

Voronoi Edges and Vertices

What does the complete Voronoi diagram look like?

We know that the edges are parts of bisectors of pairs of sites and that the vertices are intersection points between these bisectors. There is a quadratic number of bisectors, whereas the complexity of the $Vor(P)$ is only linear. Hence, not all bisectors define edges of $Vor(P)$ and not all intersections are vertices of $Vor(P)$. To characterize which bisectors and intersections define features of the Voronoi diagram we make the following definition.

> **Definition 31:** *For a point* q *we define the* **largest empty circle** *of* q *with respect to* P*, denoted by* $C_P(q)$*, as the largest circle with* q *as its center that does not contain any site of* P *in* q *its interior.*



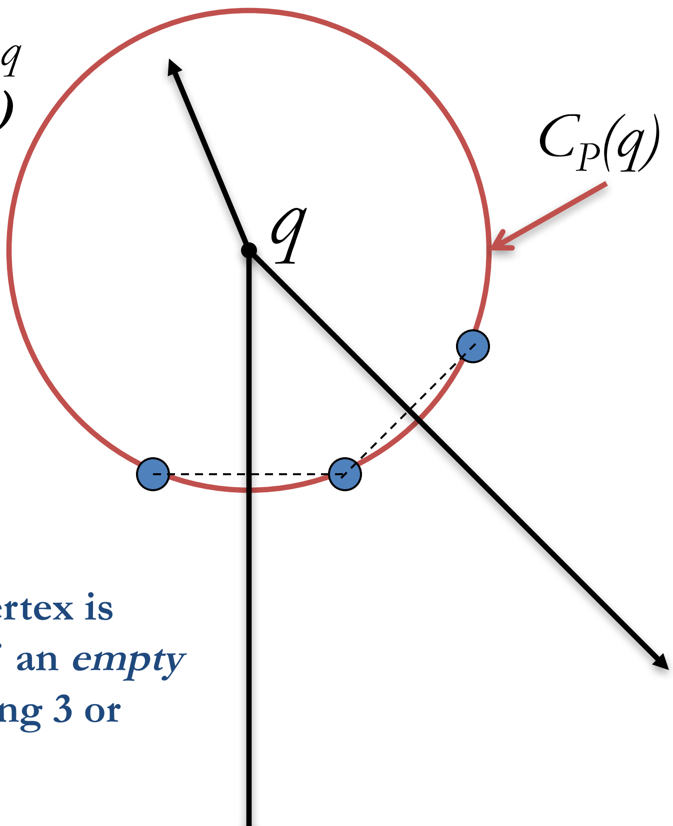
The following theorem characterizes the vertices and edges of the Voronoi diagram.

Theorem 10: For the Voronoi diagram $Vor(P)$ of a set of points P the following holds:

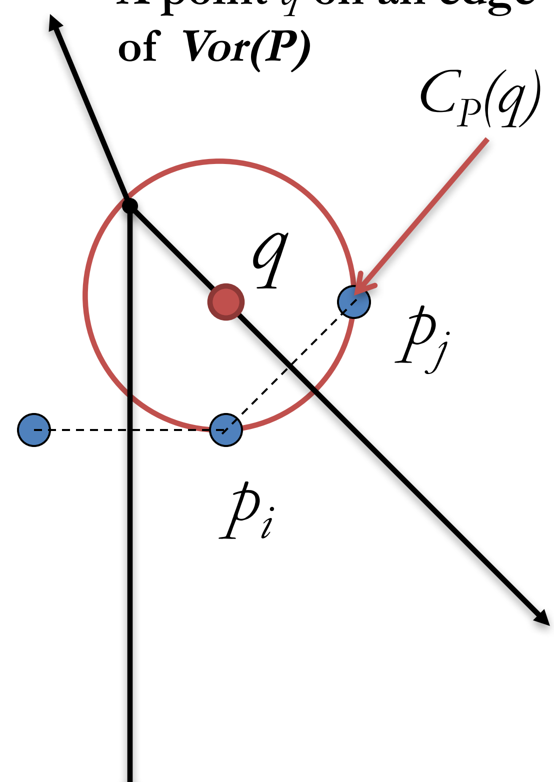
(i) A point q is a vertex of $Vor(P)$ if and only if its largest empty circle $C_P(q)$ contains three or more sites on its boundary.

(ii) The bisector between sites p_i and p_j defines an edge of $Vor(P)$ if and only if there is a point q on the bisector such that $C_P(q)$ contains both p_i and p_j on its boundary but no other site.

A vertex *q* of *Vor(P)*

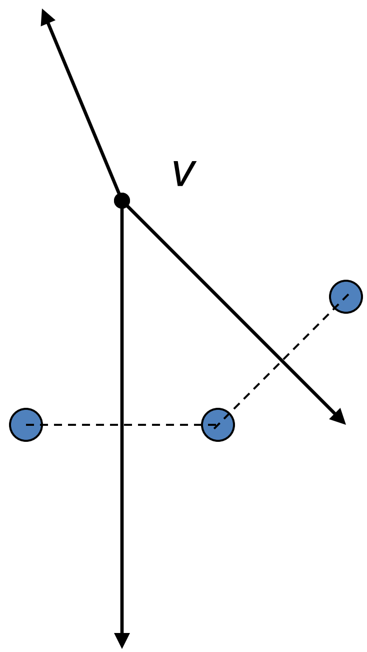


A point *q* on an edge of *Vor(P)*

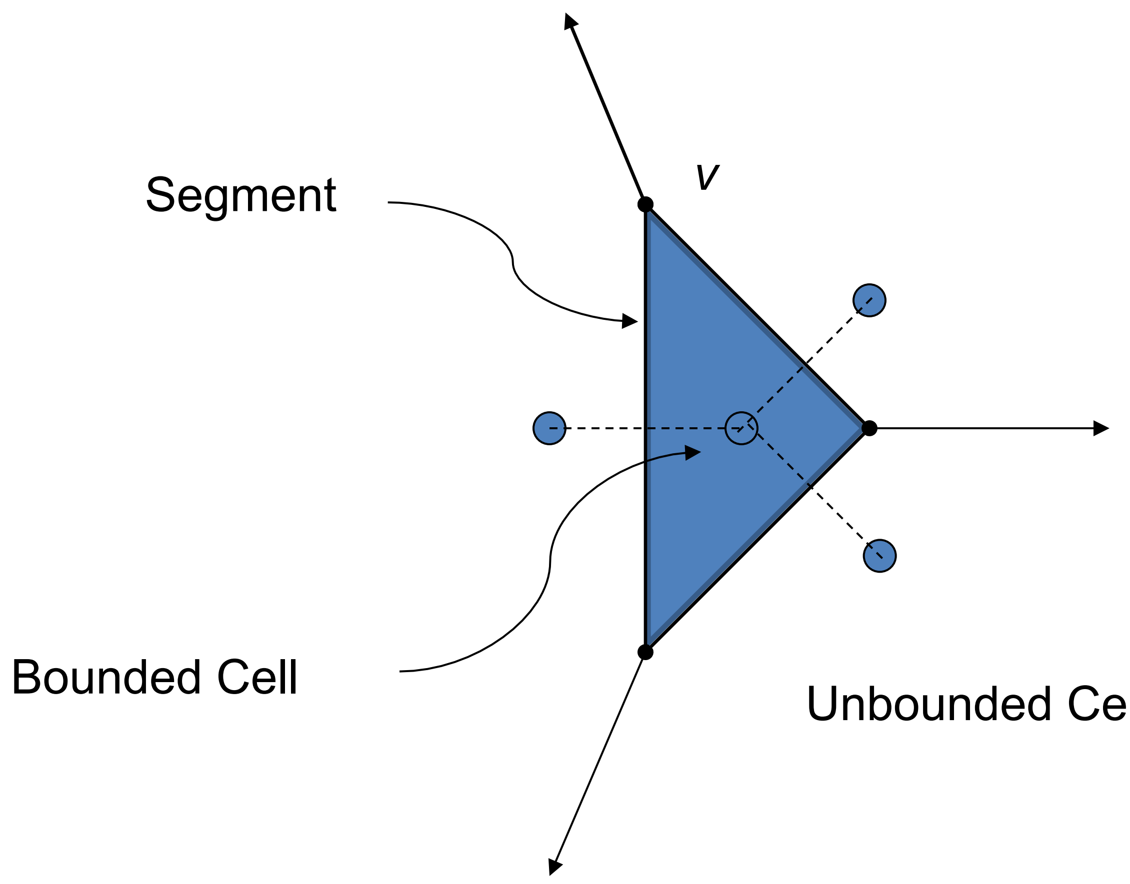


A Voronoi vertex is the center of an *empty* circle touching 3 or more sites.

Voronoi Cells and Segments



Voronoi Cells and Segments

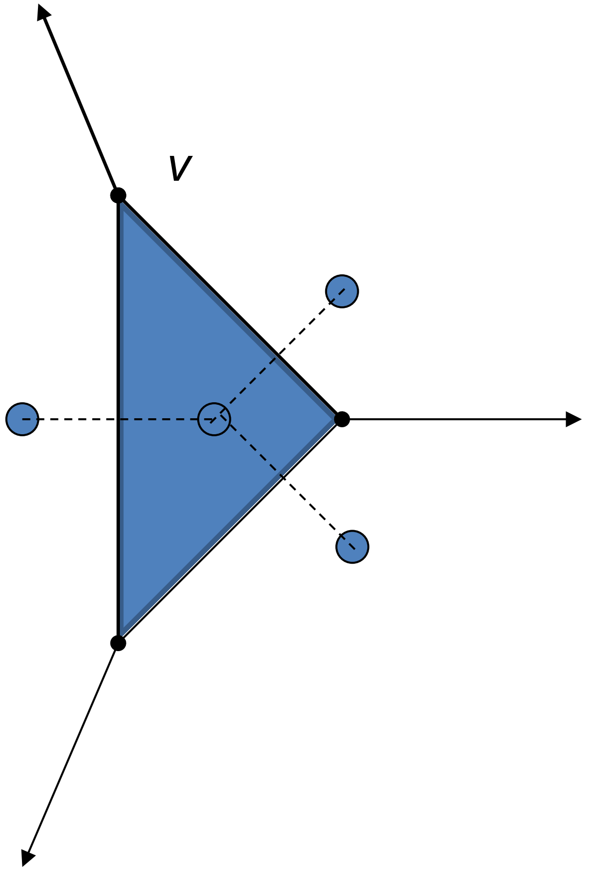


Who wants to be a Millionaire?

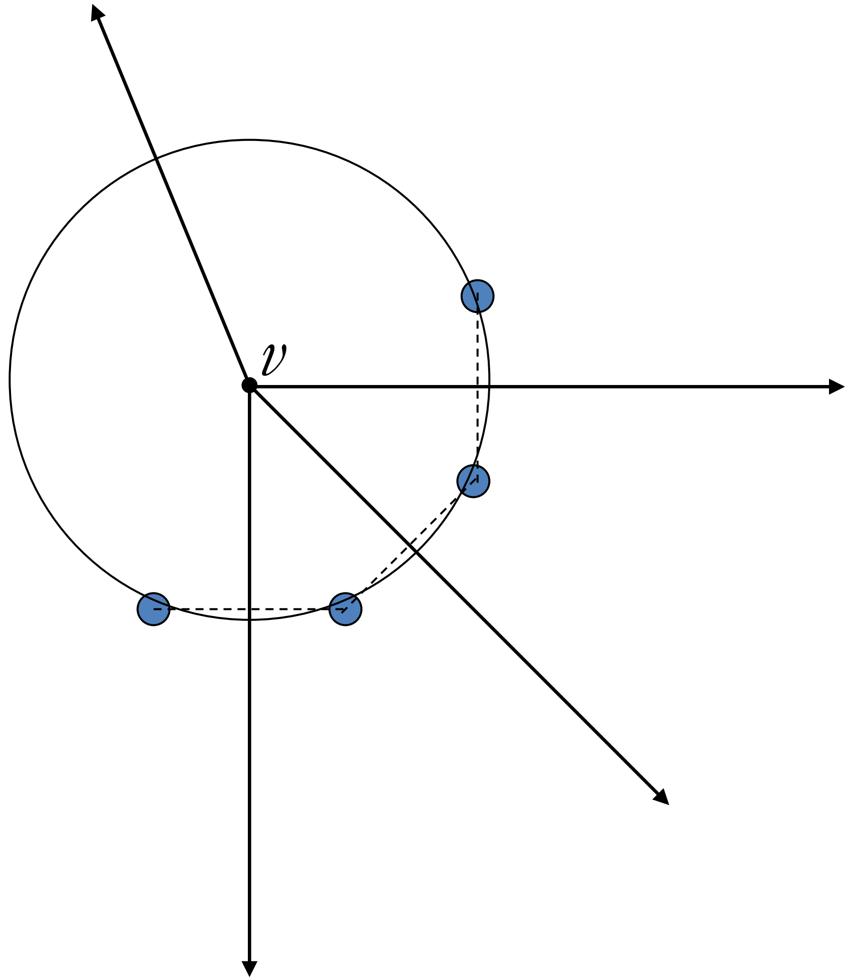
Which of the following is true for
2-D Voronoi diagrams?

Four or more non-collinear sites are…

1. sufficient to create a bounded cell
2. necessary to create a bounded cell
3. 1 and 2
4. none of above



Degenerate Case: no bounded cells ☹



Summary of Voronoi Properties

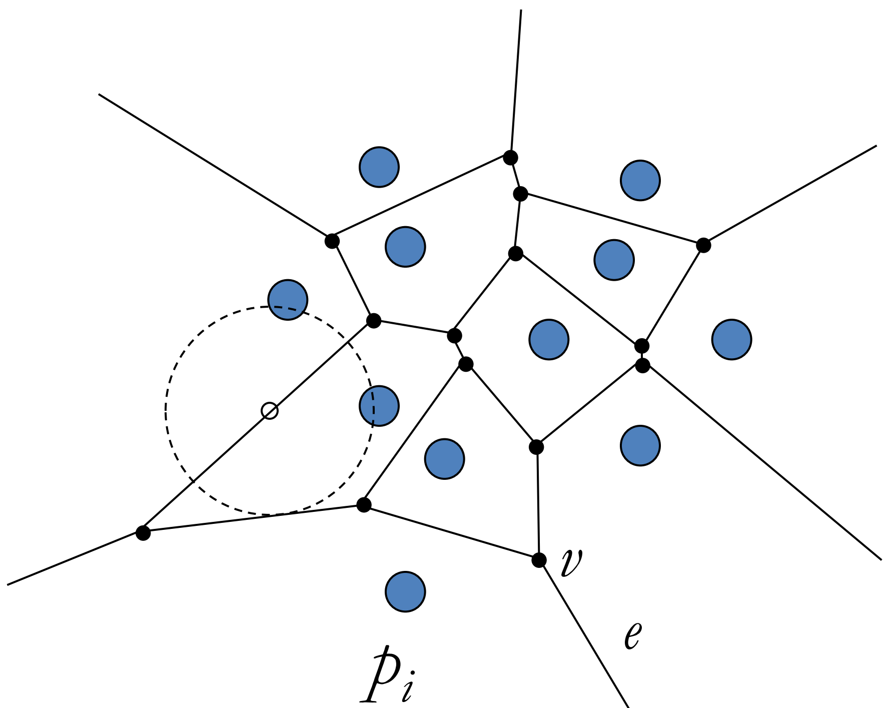
A point q lies on a Voronoi edge between sites p_i and p_j if and only if the largest empty circle centered at q touches only p_i and p_j

– A Voronoi edge is a subset of locus of points equidistant from p_i and p_j

p_i : site points

e : Voronoi edge

v : Voronoi vertex



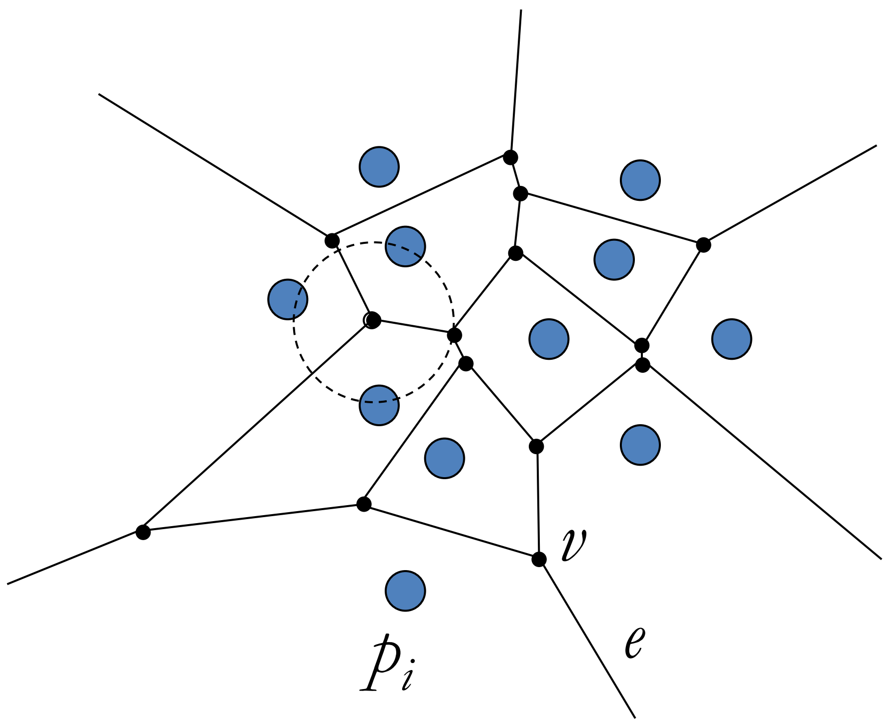
A point q is a vertex if and only if the largest empty circle centered at q touches at least 3 sites

- A Voronoi vertex is an intersection of 3 more segments, each equidistant from a pair of sites

p_i : site points

e : Voronoi edge

v : Voronoi vertex



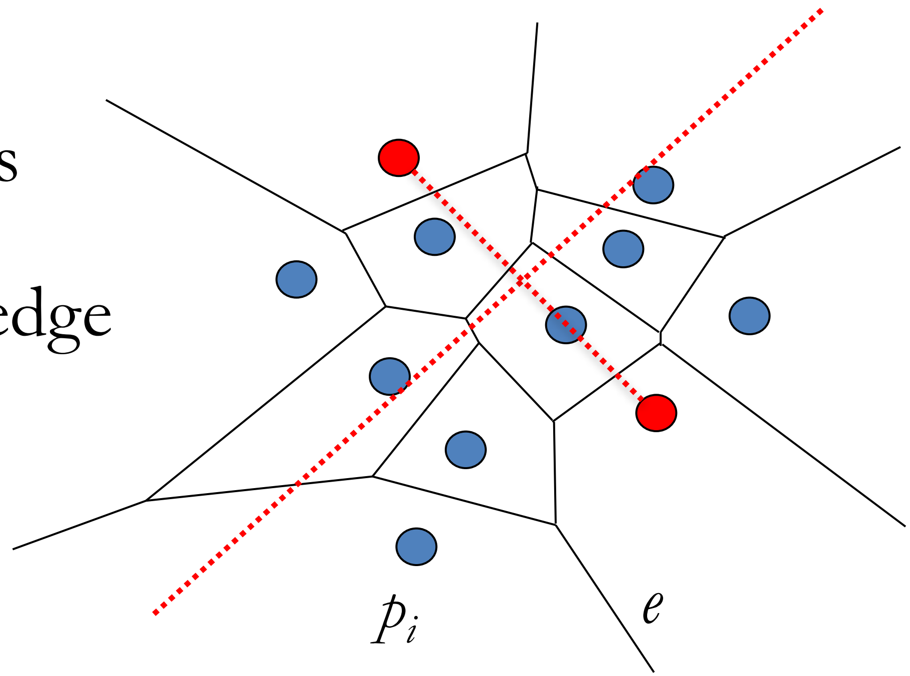
Complexity of Voronoi Diagrams

Voronoi diagrams have linear complexity $\{|v|, |e| = O(n)\}$

Intuition: Not all bisectors are Voronoi edges!

p_i : site points

e : Voronoi edge

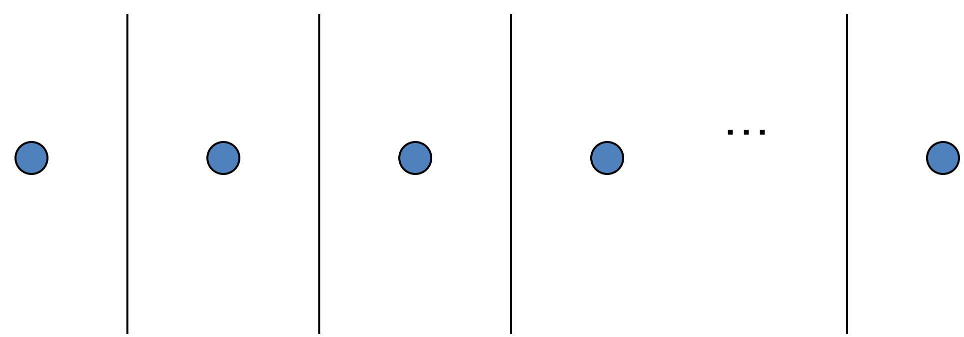


Recall …

> **Theorem 9:** *For* $n \geq 3$, *the number of vertices in the Voronoi diagram of a set of* n *point sites in the plane is at most* $2n - 5$ *and the number of edges is at most* $3n - 6$.

Claim: For $n \geq 3$, $|v| \leq 2n - 5$ and $|e| \leq 3n - 6$

Proof: (Easy Case)



Collinear sites → $|v| = 0$, $|e| = n - 1$

Recall …

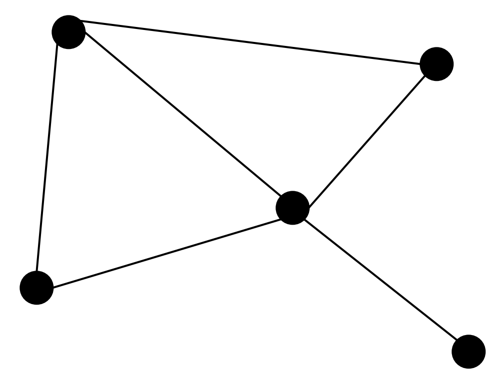
> **Theorem 7:** *Let V, E, and F be the number of vertices, edges, and faces respectively of a polyhedron, then what is now known as* **Euler's formula** *is:*
>
> $$V - E + F = 2$$

Claim: For $n \geq 3$, $|v| \leq 2n - 5$ and $|e| \leq 3n - 6$

Proof: (General Case)

- Euler's Formula: for connected, planar graphs, $|v| - |e| + f = 2$

Where:



$|v|$ is the number of vertices

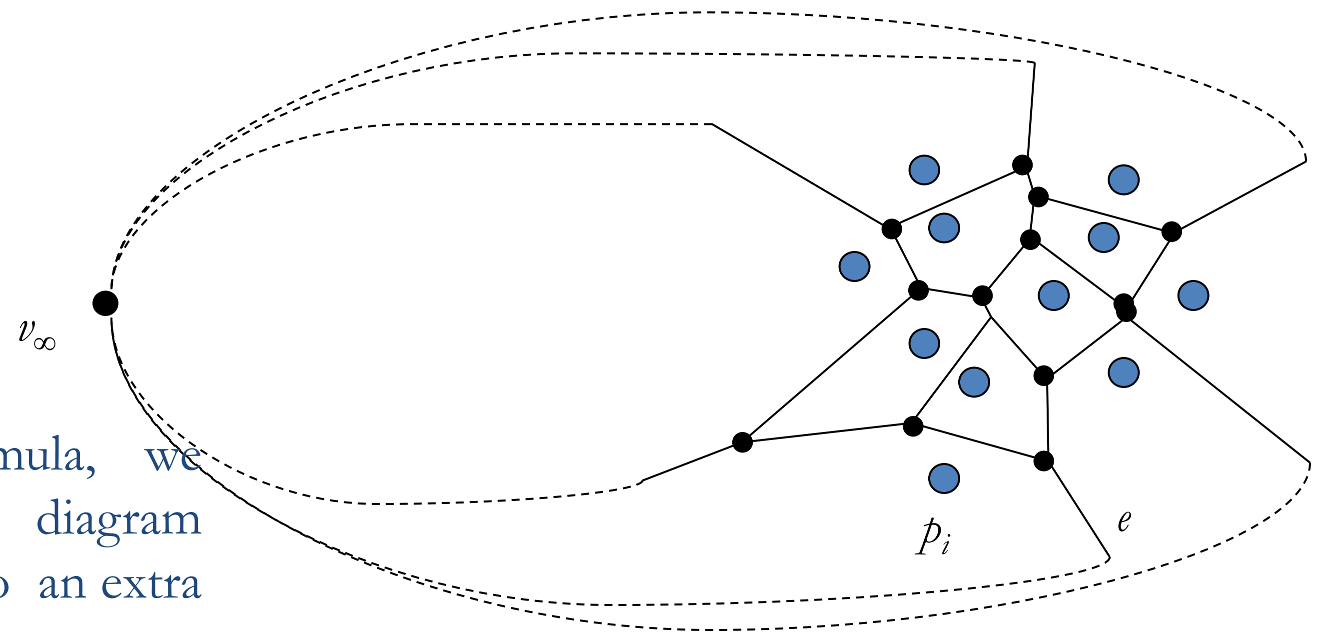
$|e|$ is the number of edges

f is the number of faces

Claim: For $n \geq 3$, $|v| \leq 2n - 5$ and $|e| \leq 3n - 6$

Proof: (General Case)

- For Voronoi graphs, $f = n \rightarrow (|v| + 1) - |e| + n = 2$



To apply Euler's Formula, we "planarize" the Voronoi diagram by connecting half lines to an extra vertex.

Moreover,

$$\sum_{v \in Vor(P)} \deg(v) = 2 \cdot |e|$$

and

$$\forall v \in Vor(P), \quad \deg(v) \geq 3$$

so

$$2 \cdot |e| \geq 3(|v|+1)$$

$$(|v|+1) - |e| + n = 2$$

together with

$$|v| \leq 2n-5$$
$$|e| \leq 3n-6$$

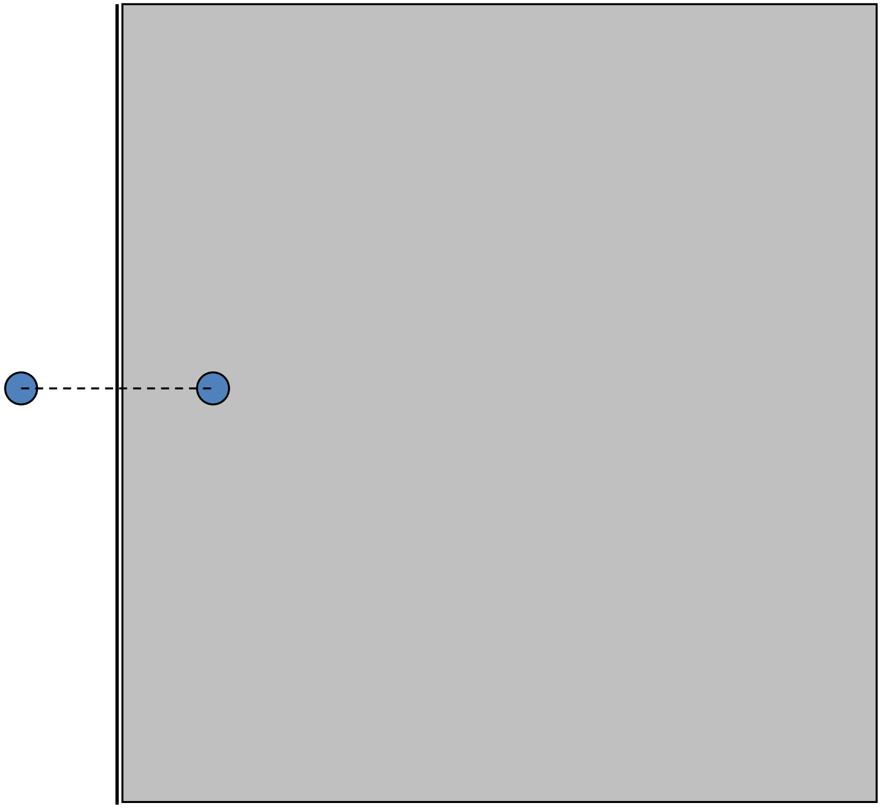
we get, for $n \geq 3$

Constructing Voronoi Diagrams

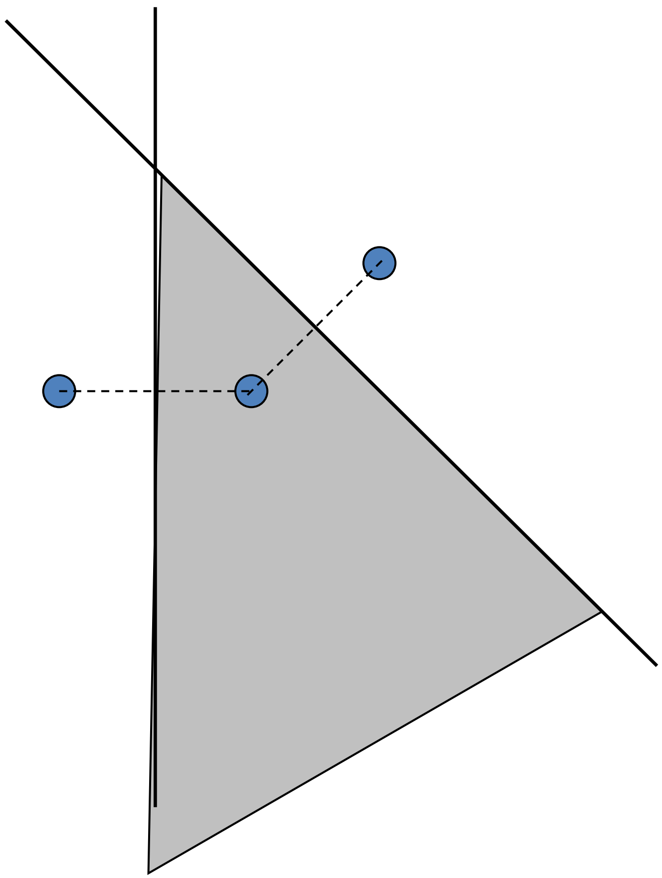
Intuitions

Given a half plane intersection algorithm…

Given a half plane intersection algorithm…



Given a half plane intersection algorithm…

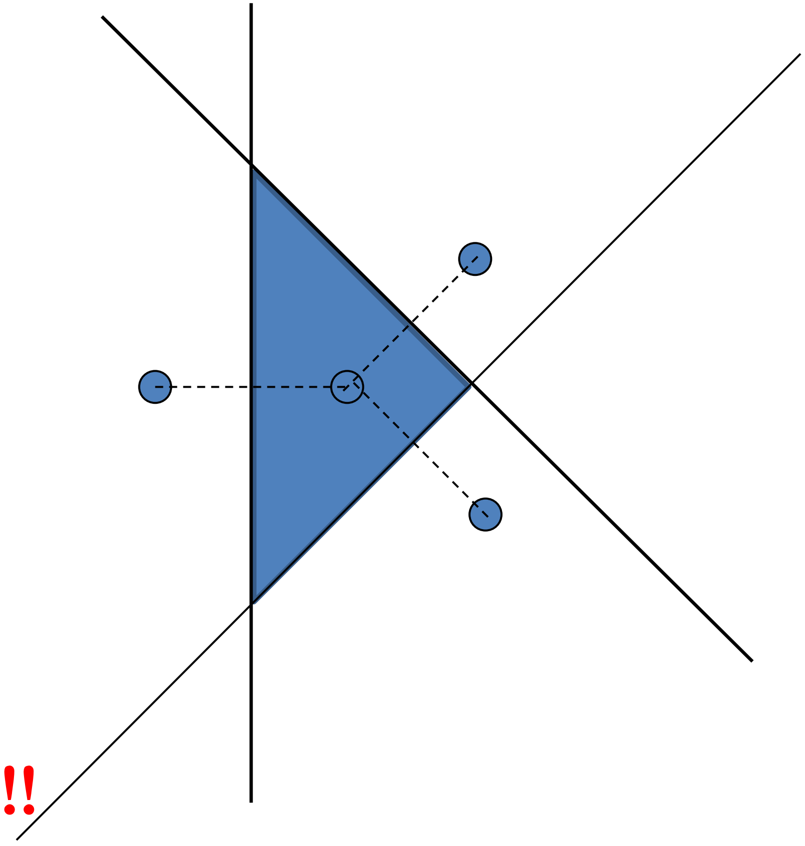


Given a half plane intersection algorithm…

Repeat for each site

Running Time:
O(n^2 log n)

Can we do better ?!!!

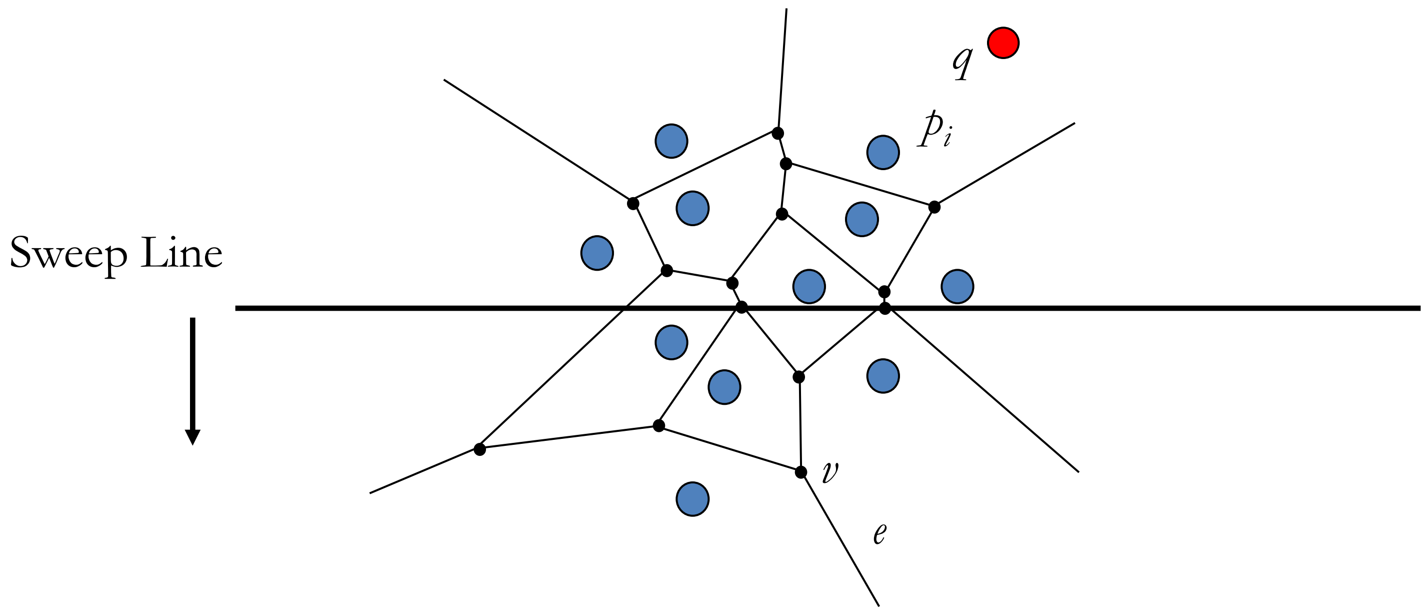


Constructing Voronoi Diagrams

Fortune's Algorithm

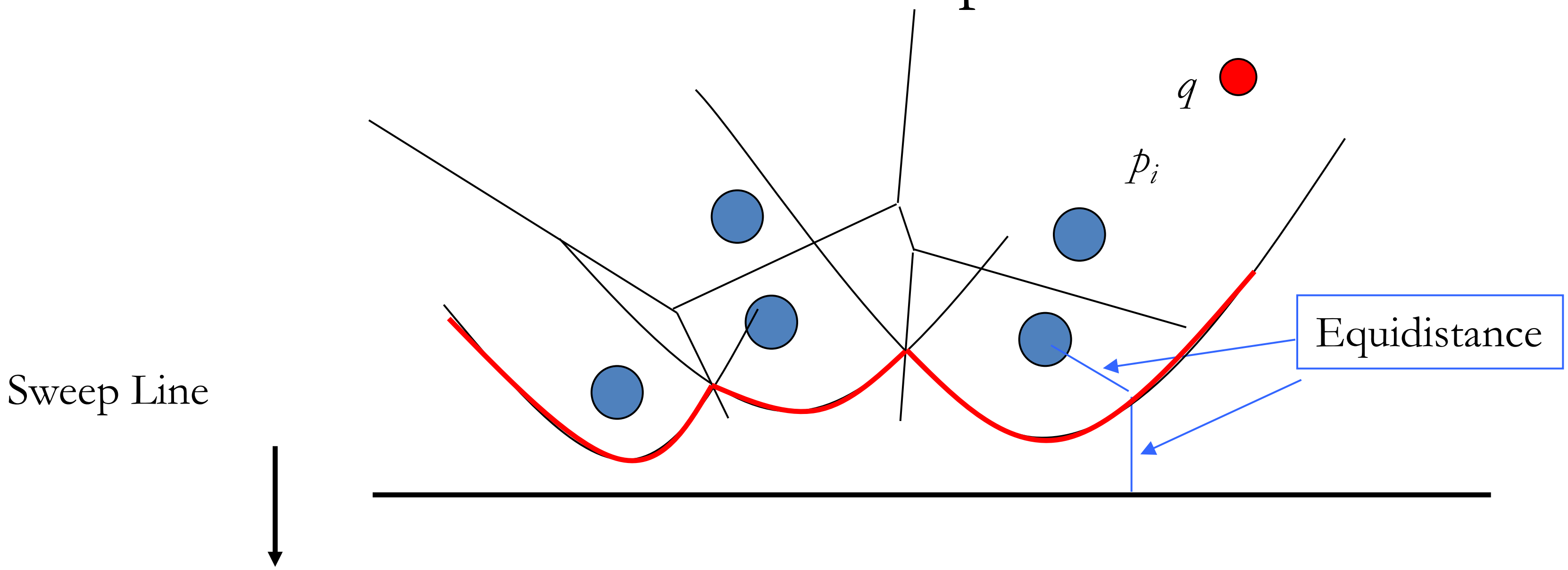
- Half plane intersection O($n^2 \log n$)
- Fortune's Algorithm
  - Sweep line algorithm
    - Voronoi diagram constructed as horizontal line sweeps the set of sites from top to bottom
    - Incremental construction → maintains portion of diagram which cannot change due to sites below sweep line, keeping track of incremental changes for each site (and Voronoi vertex) it "sweeps"

What is the invariant we are looking for?



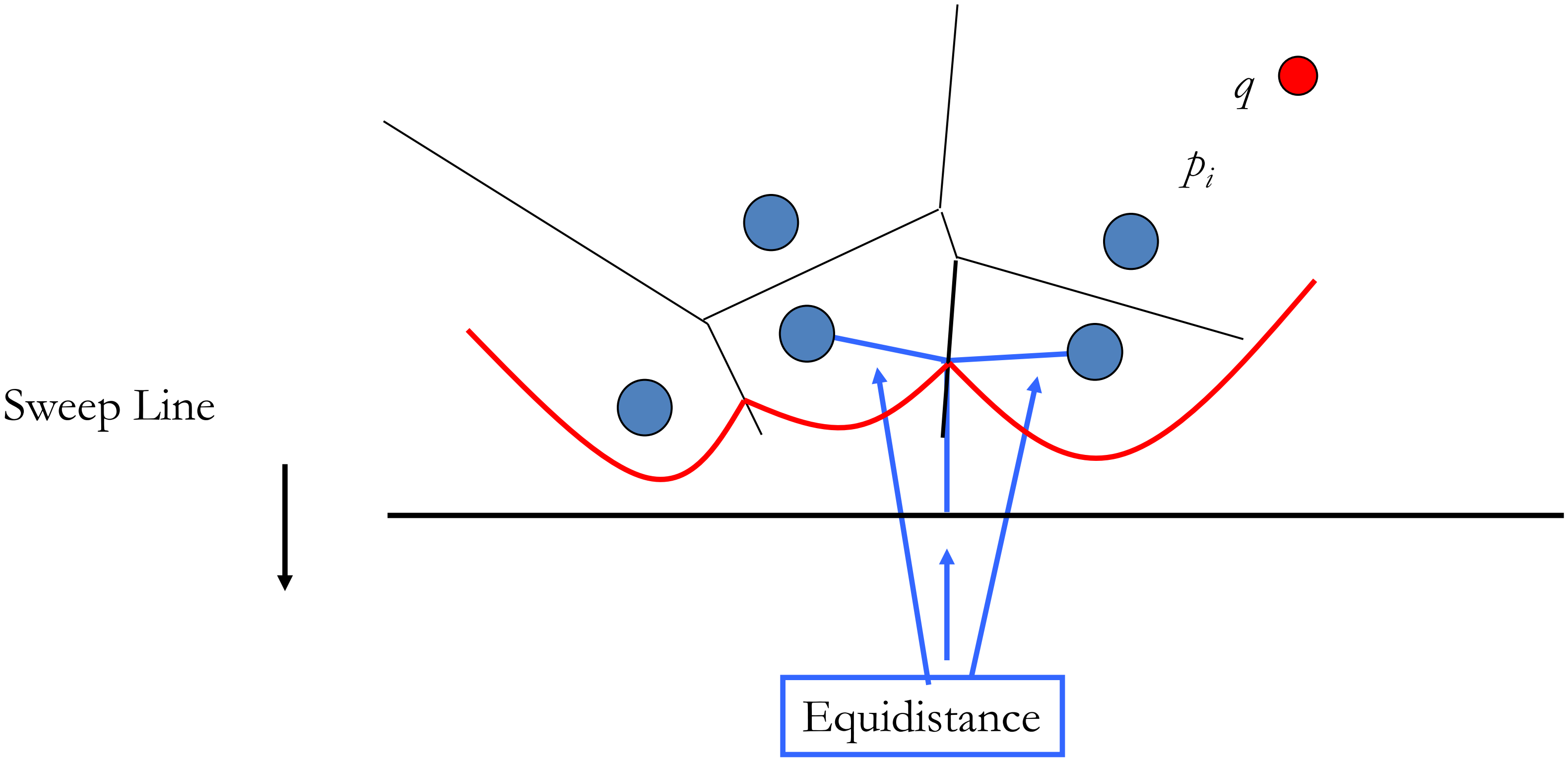
Maintain a representation of the locus of points q that are closer to some site p_i *above* the sweep line than to the line itself (and thus to any site below the line).

Which points are closer to a site above the sweep line than to the sweep line itself?



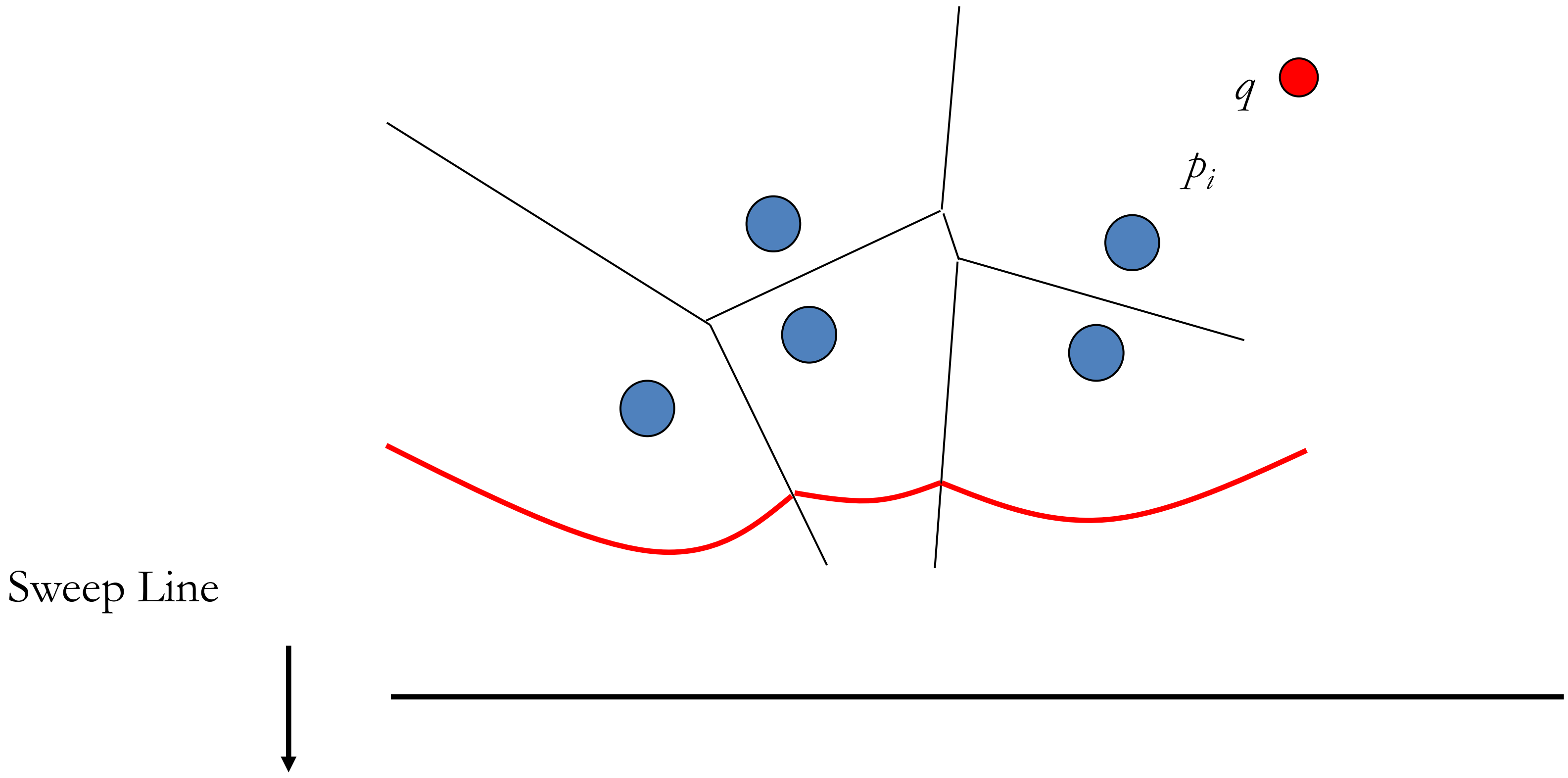
The set of parabolic arcs form a beach-line that bounds the locus of all such points

Break points trace out Voronoi edges.

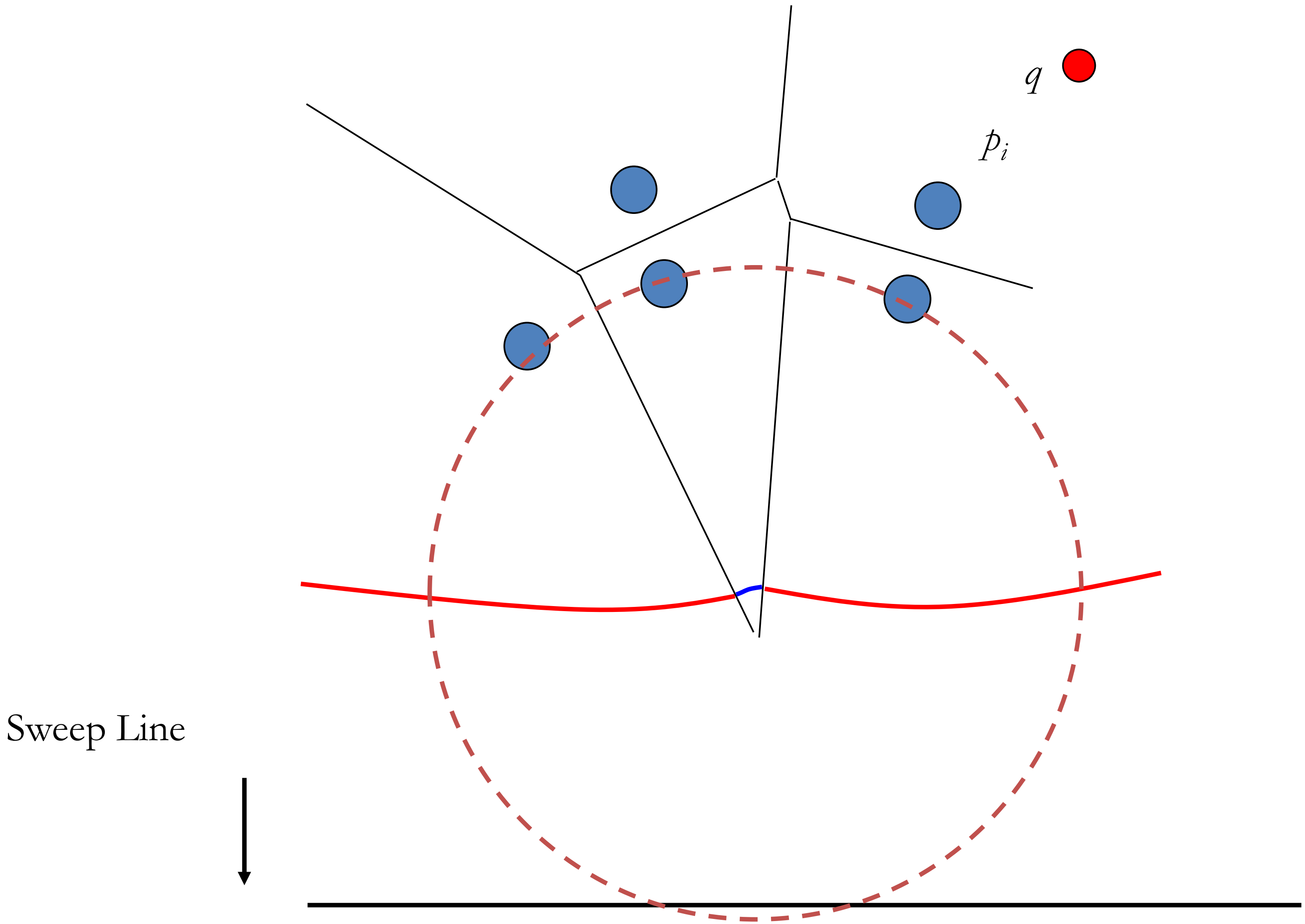


Break points do not trace out edges continuously in the actual algorithm. The sweep line stops at discrete event points as will be shown later.

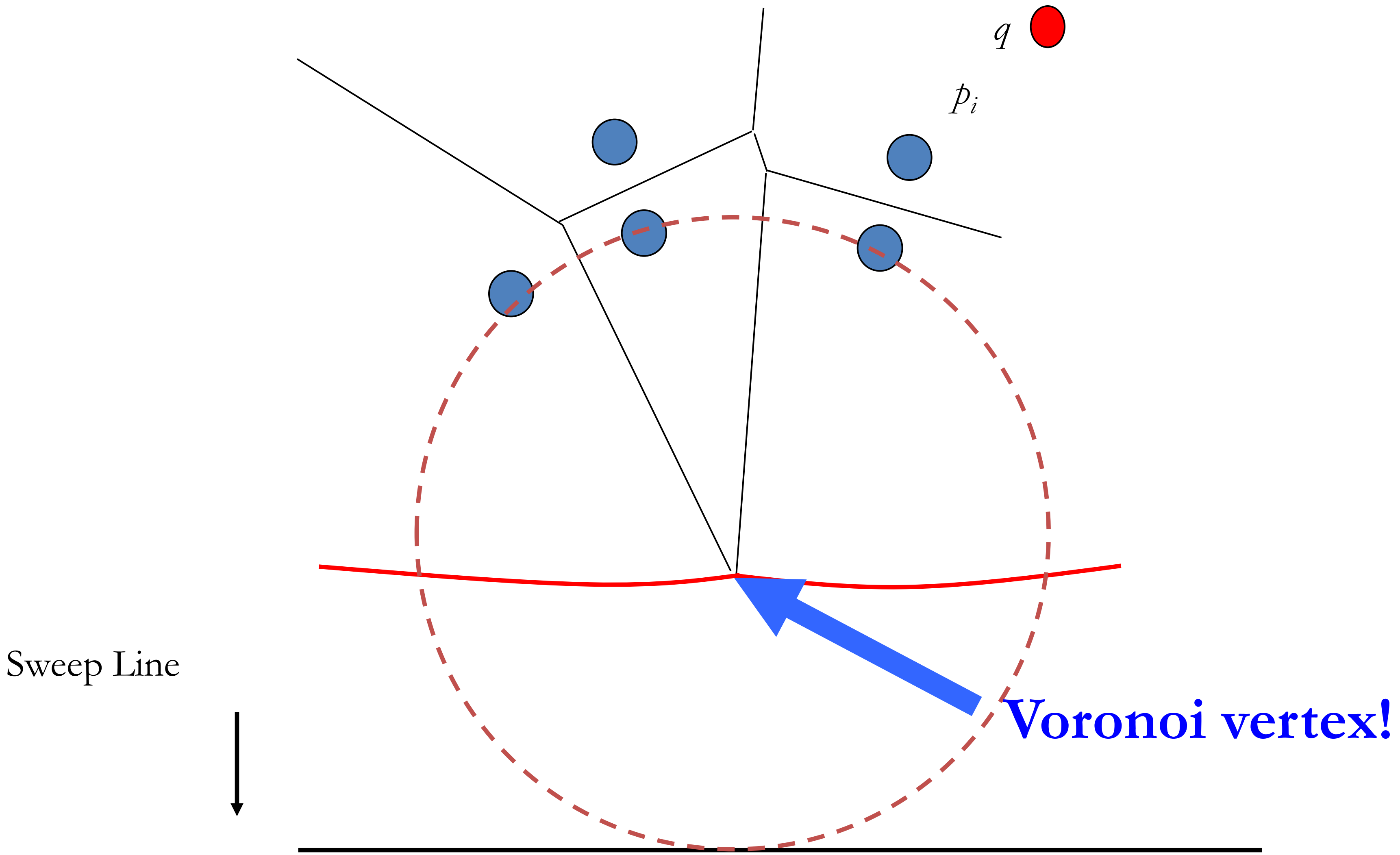
Arcs flatten out as sweep line moves down.



Eventually, the middle arc disappears.



We have detected a circle that is empty (contains no sites) and touches 3 or more sites.



Beach Line properties

- Voronoi edges are traced by the break points as the sweep line moves down.
  - Emergence of a new break point(s) (from formation of a new arc or a fusion of two existing break points) identifies a new edge
- Voronoi vertices are identified when two break points meet (fuse).
  - Decimation of an old arc identifies new vertex

Constructing Voronoi Diagrams

Data Structures

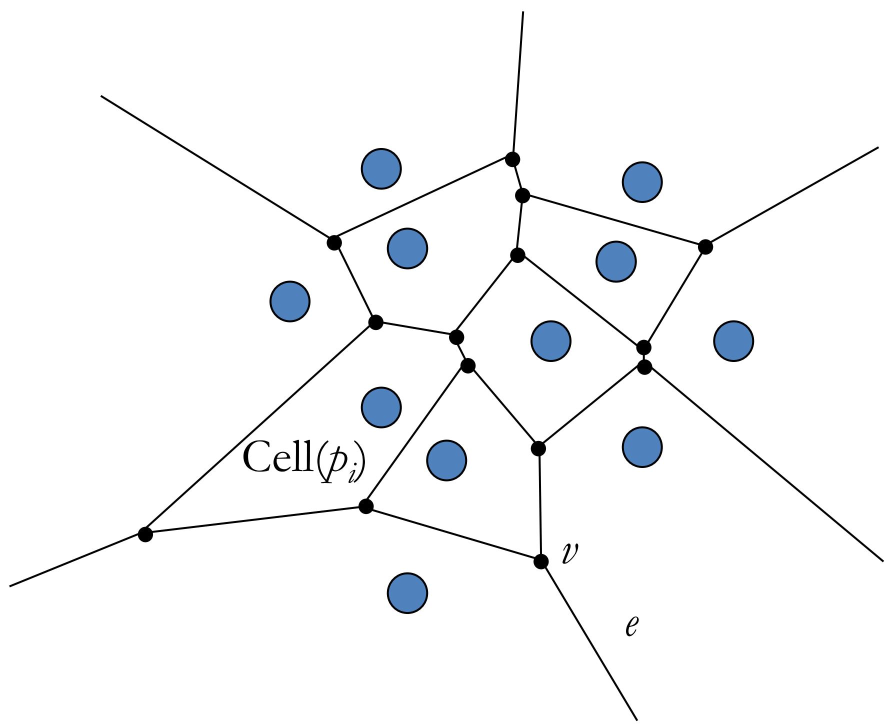
Data Structures

- Current state of the Voronoi diagram
  - Doubly linked list of half-edge, vertex, cell records
- Current state of the beach line
  - Keep track of break points
  - Keep track of arcs currently on beach line
- Current state of the sweep line
  - Priority event queue sorted on decreasing y-coordinate

Discrete sweep steps, rather than a continuous sweep

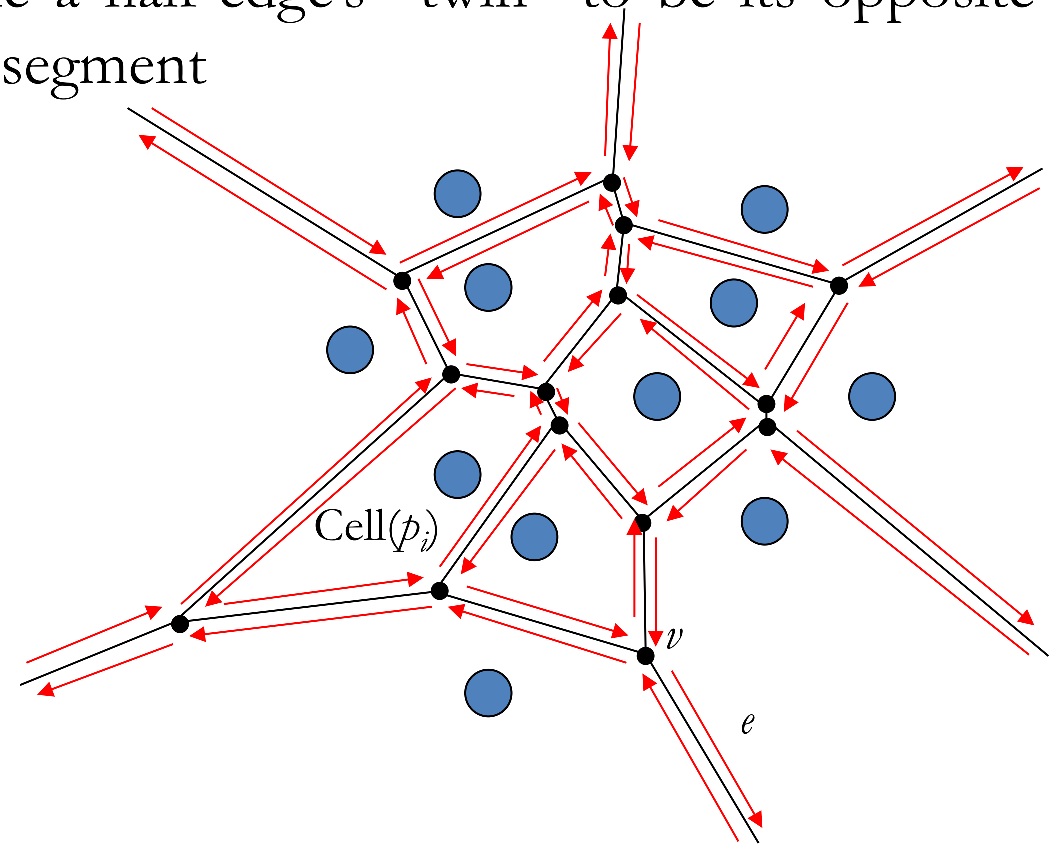
Doubly Linked List (D)

- Goal: a simple data structure that allows an algorithm to traverse a Voronoi diagram's segments, cells and vertices



Doubly Linked List (D)

- Divide segments into uni-directional half-edges
- A chain of counter-clockwise half-edges forms a cell
- Define a half-edge's "twin" to be its opposite half-edge of the same segment

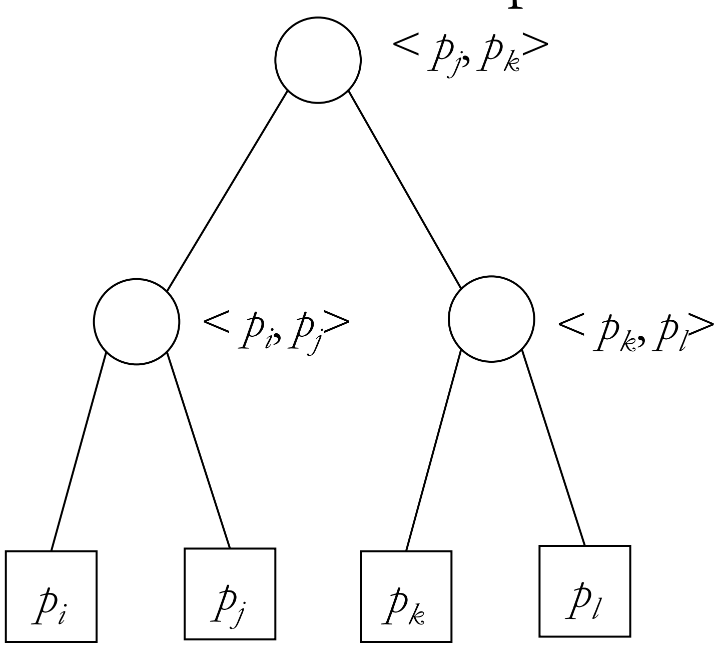


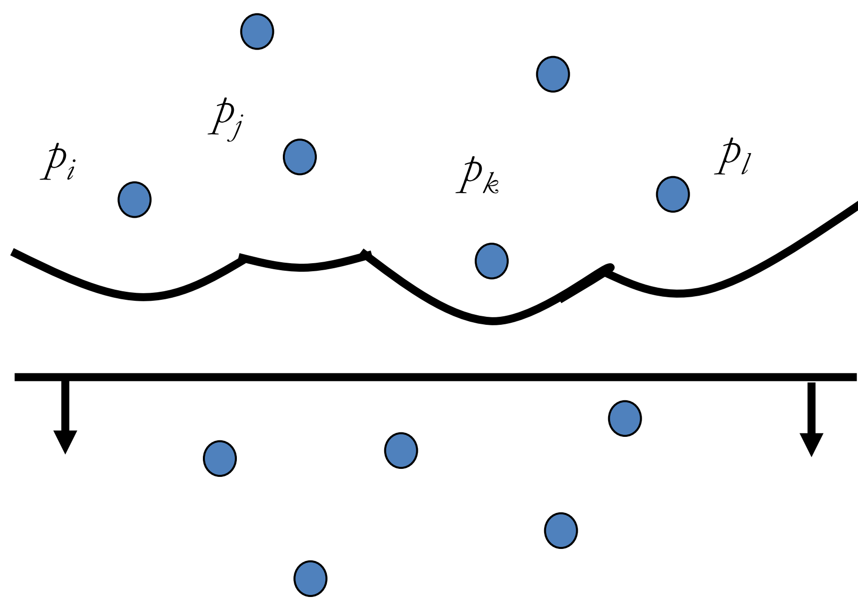
Doubly Linked List (*D*)

- Cell Table
  - Cell(p_i) : pointer to any incident half-edge
- Vertex Table
  - v_i: list of pointers to all incident half-edges
- Doubly Linked-List of half-edges; each has:
  - Pointer to Cell Table entry
  - Pointers to start/end vertices of half-edge
  - Pointers to previous/next half-edges in the CCW chain
  - Pointer to twin half-edge

Balanced Binary Tree (*T*)

- Internal nodes represent break points between two arcs
  - Also contains a pointer to the *D* record of the edge being traced
- Leaf nodes represent arcs, each arc is in turn represented by the site that generated it
  - Also contains a pointer to a potential circle event



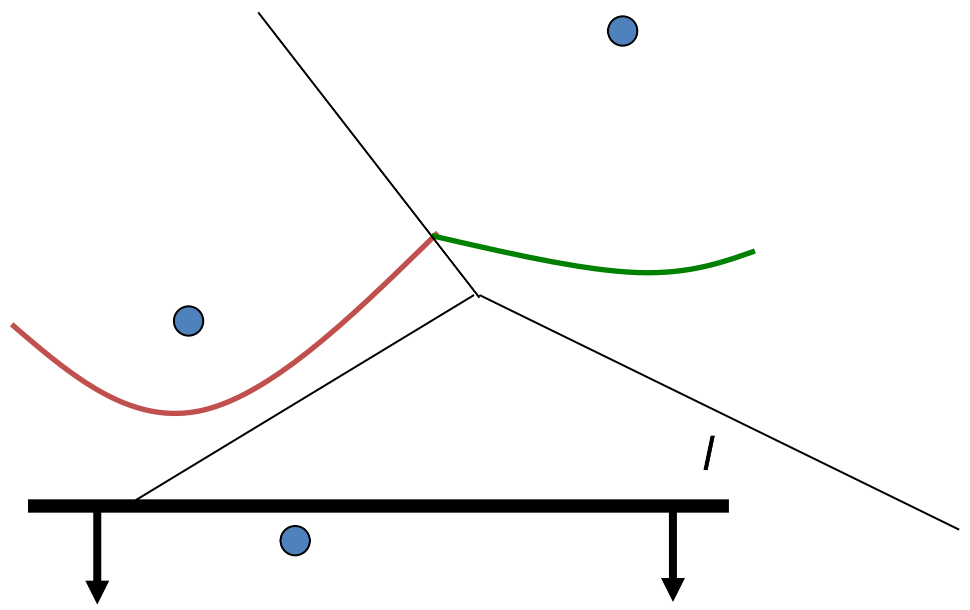


Event Queue (*Q*)

- An event is an interesting point encountered by the sweep line as it sweeps from top to bottom
  - Sweep line makes discrete stops, rather than a continuous sweep
- Consists of Site Events (when the sweep line encounters a new site point) and Circle Events (when the sweep line encounters the *bottom* of an empty circle touching 3 or more sites).
- Events are prioritized based on y-coordinate

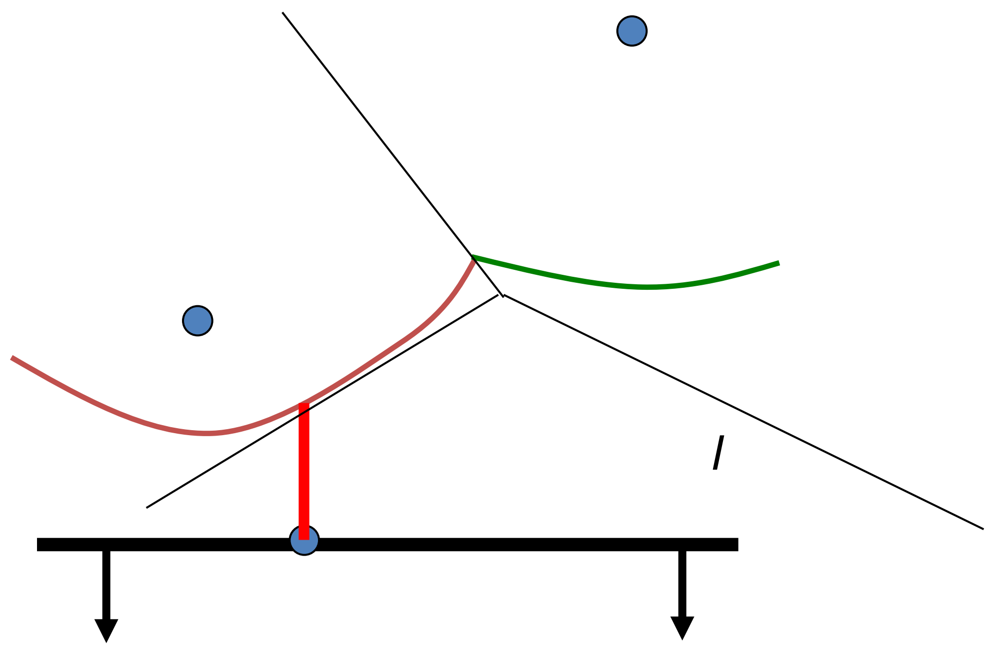
Site Event

A new arc appears when a new site appears.



Site Event

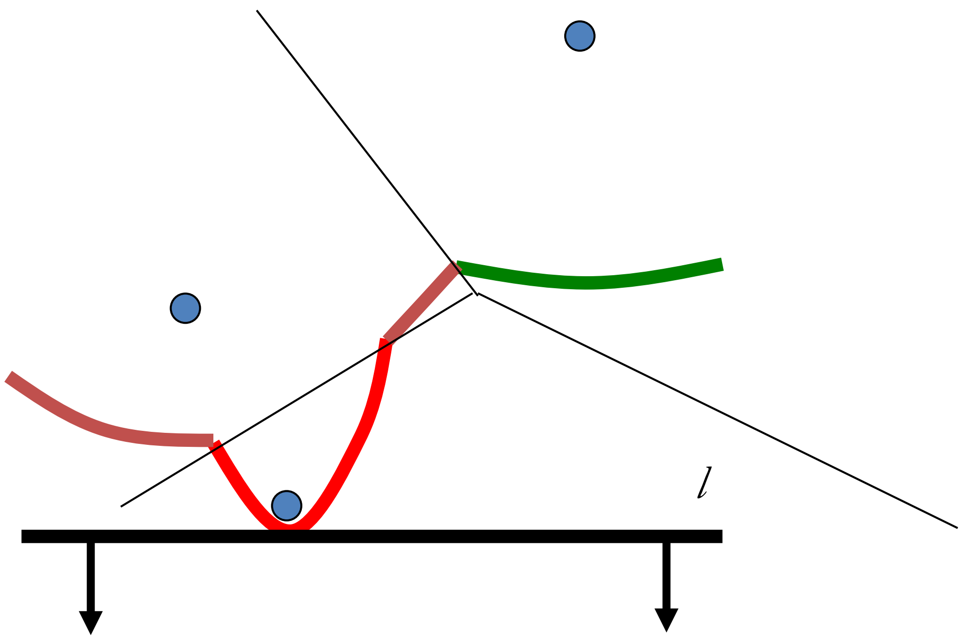
A new arc appears when a new site appears.



Site Event

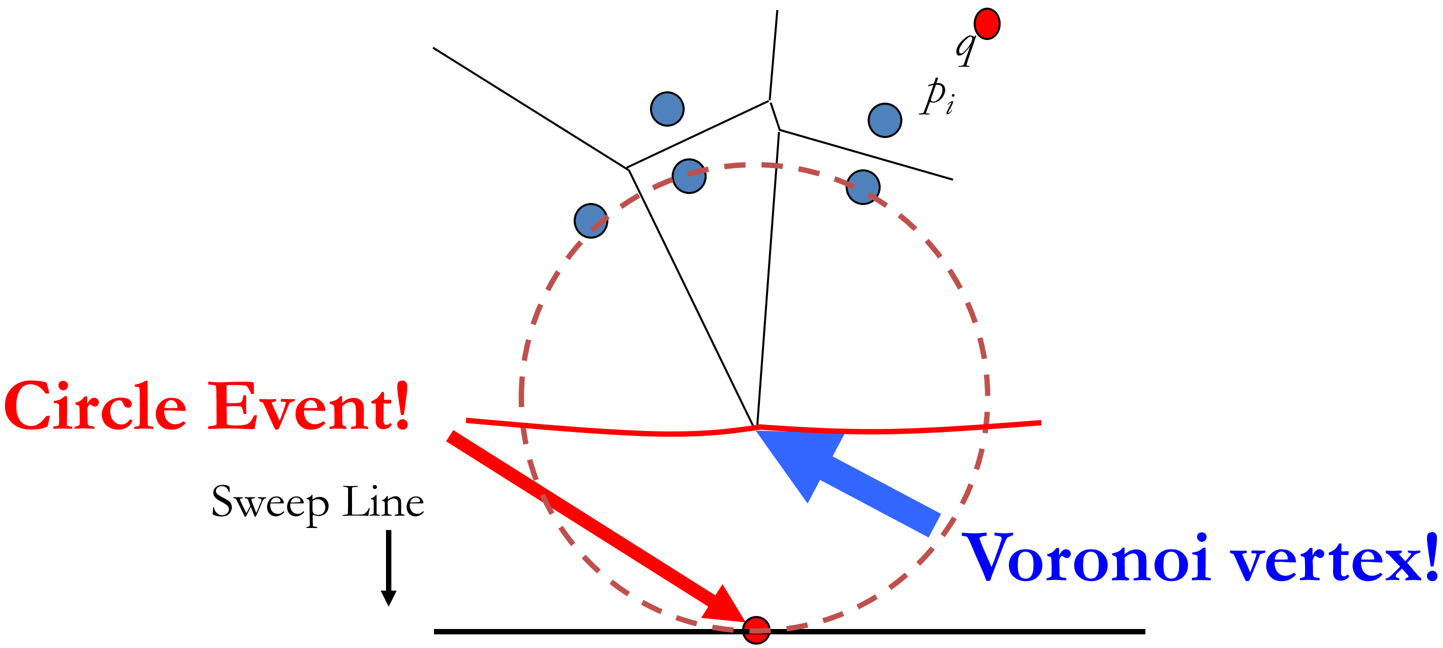
Original arc above the new site is broken into two

→ Number of arcs on beach line is O(n)



Circle Event

An arc disappears whenever an empty circle touches three or more sites and is tangent to the sweep line.



Sweep line helps determine that the circle is indeed empty.

Event Queue Summary

- Site Events are
  - given as input
  - represented by the xy-coordinate of the site point
- Circle Events are
  - computed on the fly (intersection of the two bisectors in between the three sites)
  - represented by the xy-coordinate of the lowest point of an empty circle touching three or more sites
  - "anticipated", these newly generated events may be false and need to be removed later
- Event Queue prioritizes events based on their y-coordinates

Summarizing Data Structures

- Current state of the Voronoi diagram
  - Doubly linked list of half-edge, vertex, cell records
- Current state of the beach line
  - Keep track of break points
    - Inner nodes of binary search tree; represented by a tuple
  - Keep track of arcs currently on beach line
    - Leaf nodes of binary search tree; represented by a site that generated the arc
- Current state of the sweep line
  - Priority event queue sorted on decreasing y-coordinate

Constructing Voronoi Diagrams

Algorithm

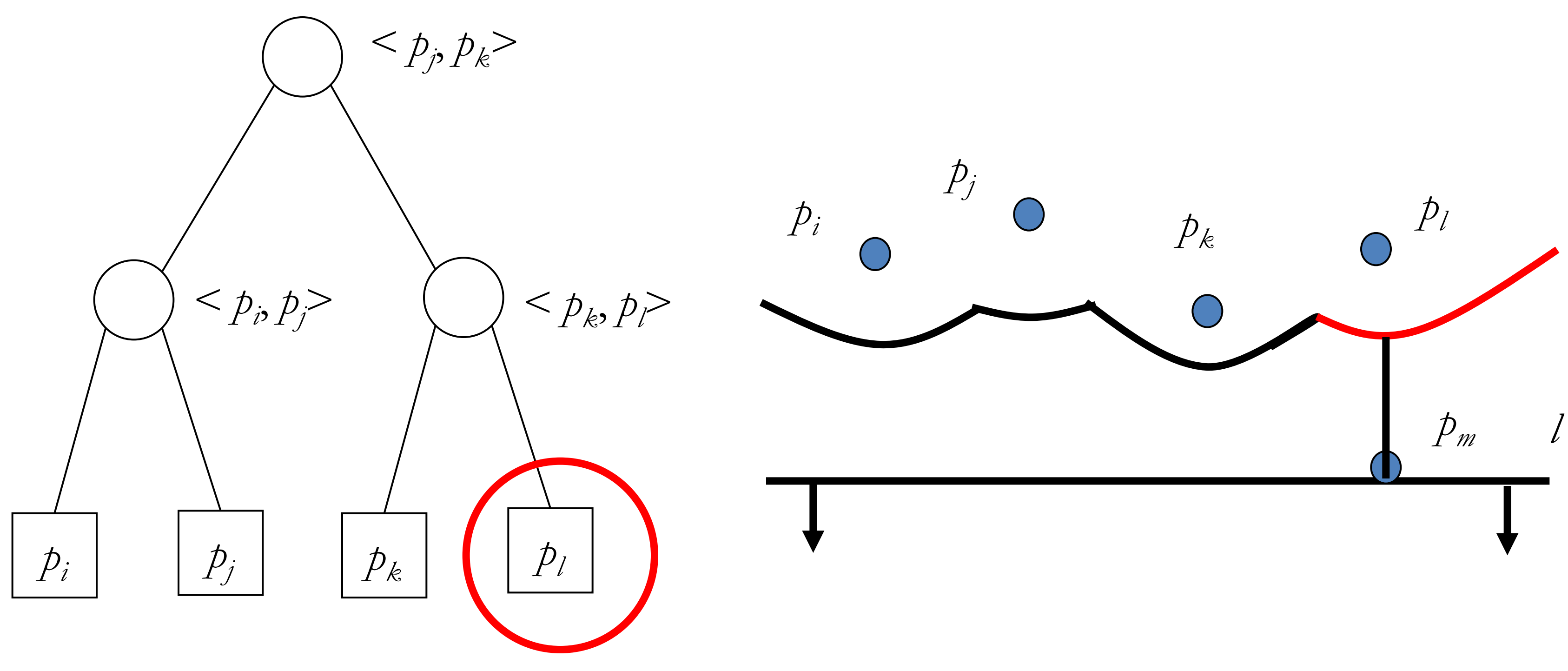
1. Initialize
    - Event queue Q ← all site events
    - Binary search tree T ← ∅
    - Doubly linked list D ← ∅
2. While Q not ∅,
    - Remove event (e) from Q with largest y-coordinate
        - HandleEvent(e, T, D)

Handling Site Events

1. Locate the existing arc (if any) that is above the new site
2. Break the arc by replacing the leaf node with a sub tree representing the new arc and its break points
3. Add two half-edge records in the doubly linked list
4. Check for potential circle event(s), add them to event queue if they exist

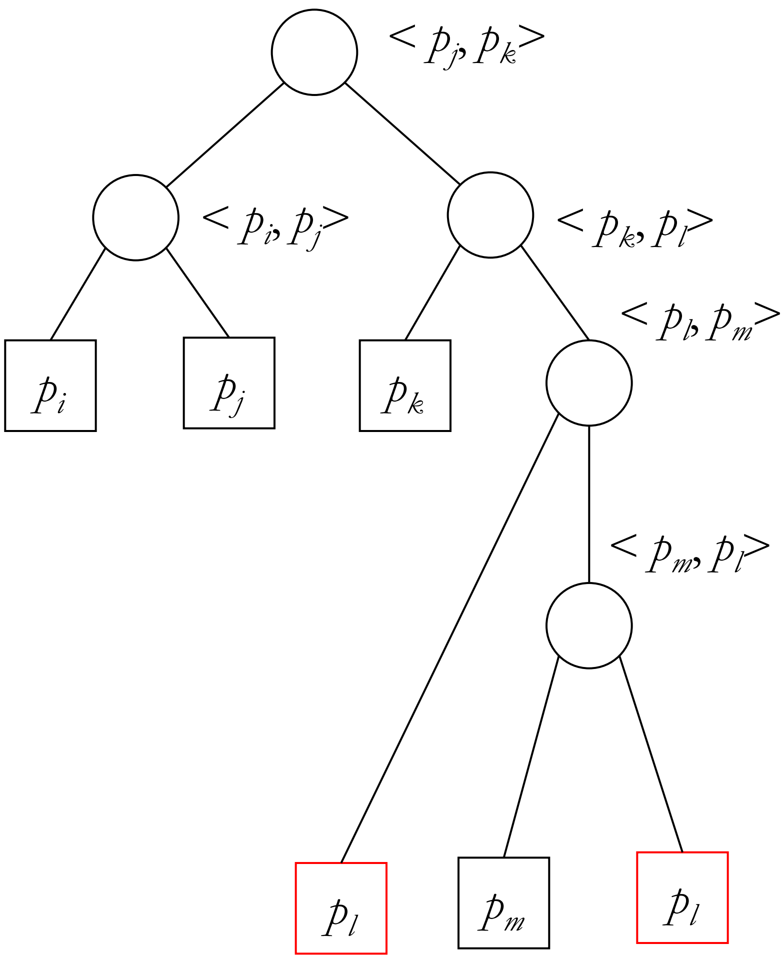
Locate the existing arc that is above the new site

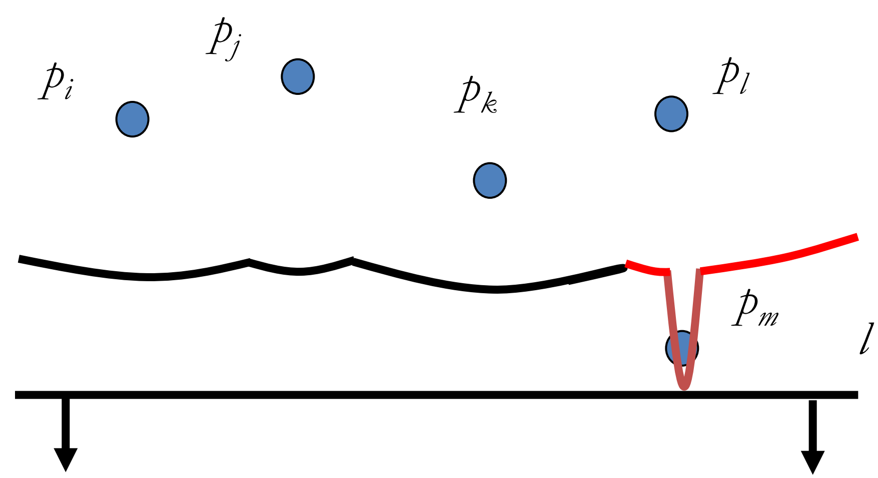
- The x coordinate of the new site is used for the binary search
- The x coordinate of each breakpoint along the root to leaf path is computed on the fly



Break the Arc

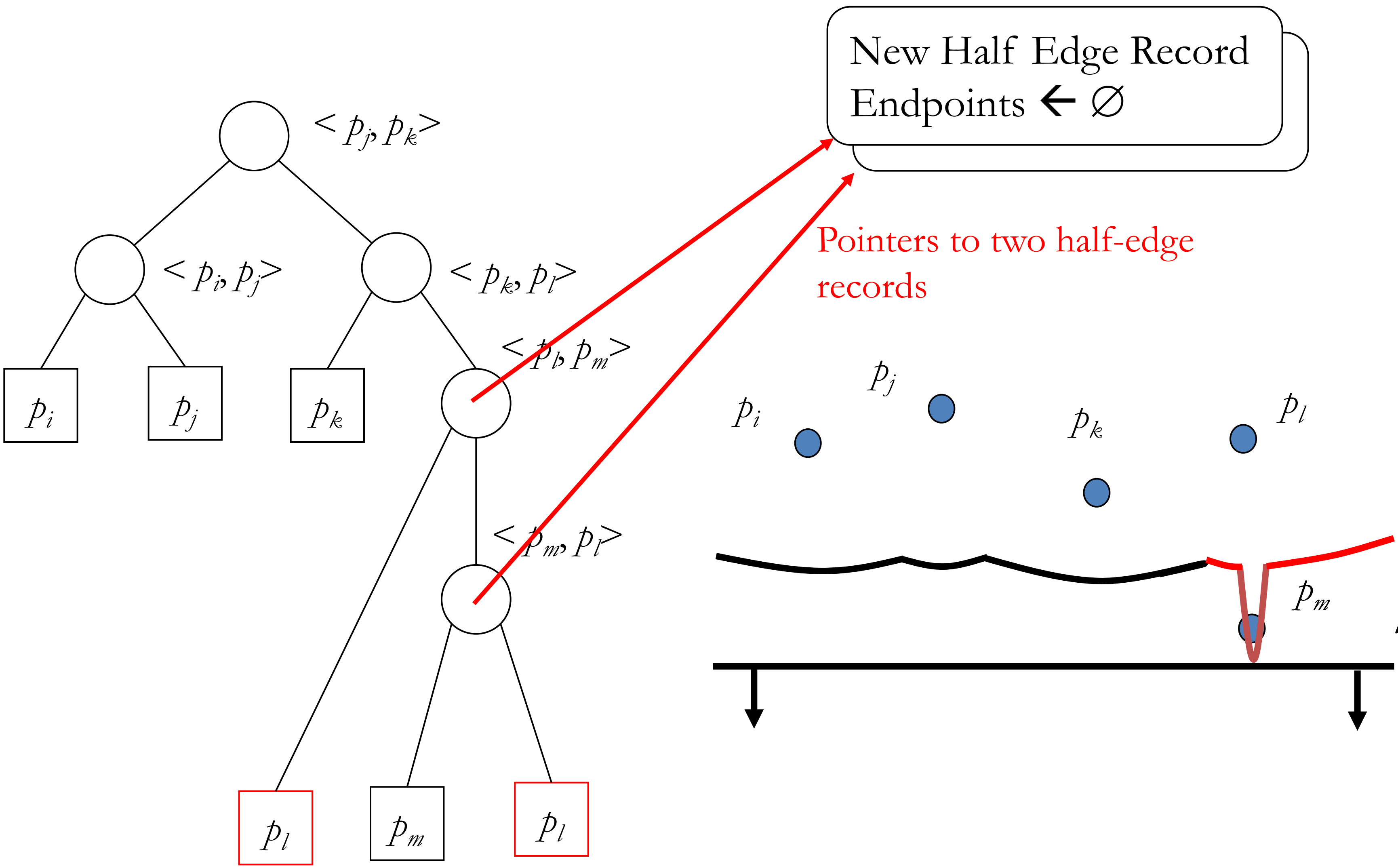
Corresponding leaf replaced by a new sub-tree





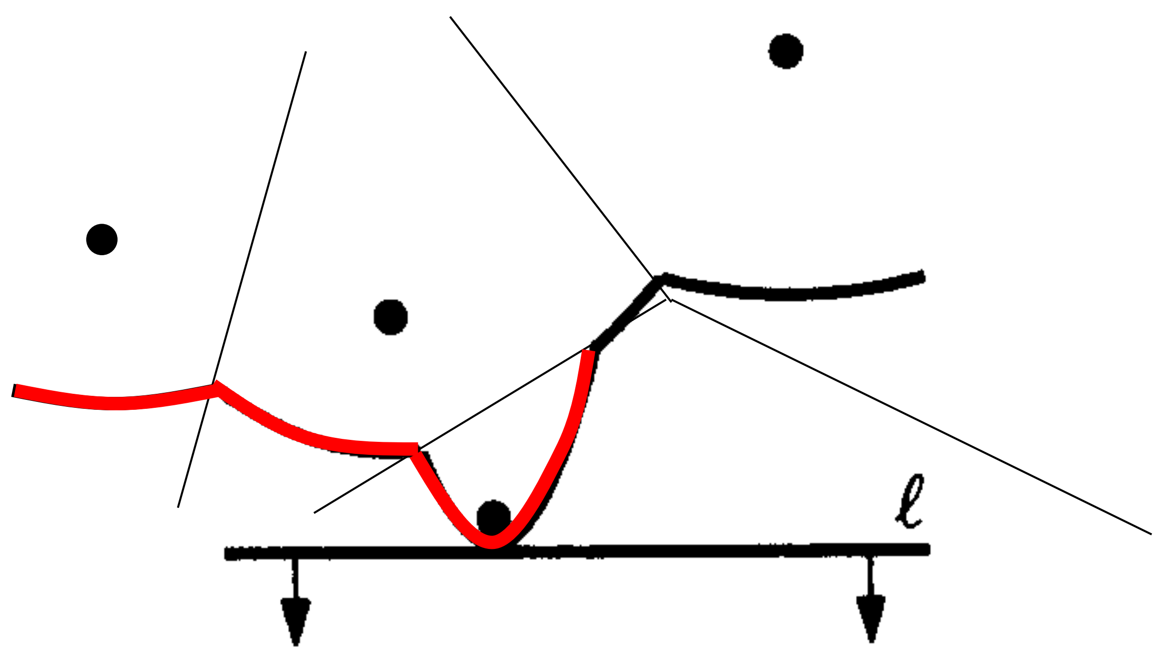
Different arcs can be identified
by the same site!

Add a new edge record in the doubly linked list



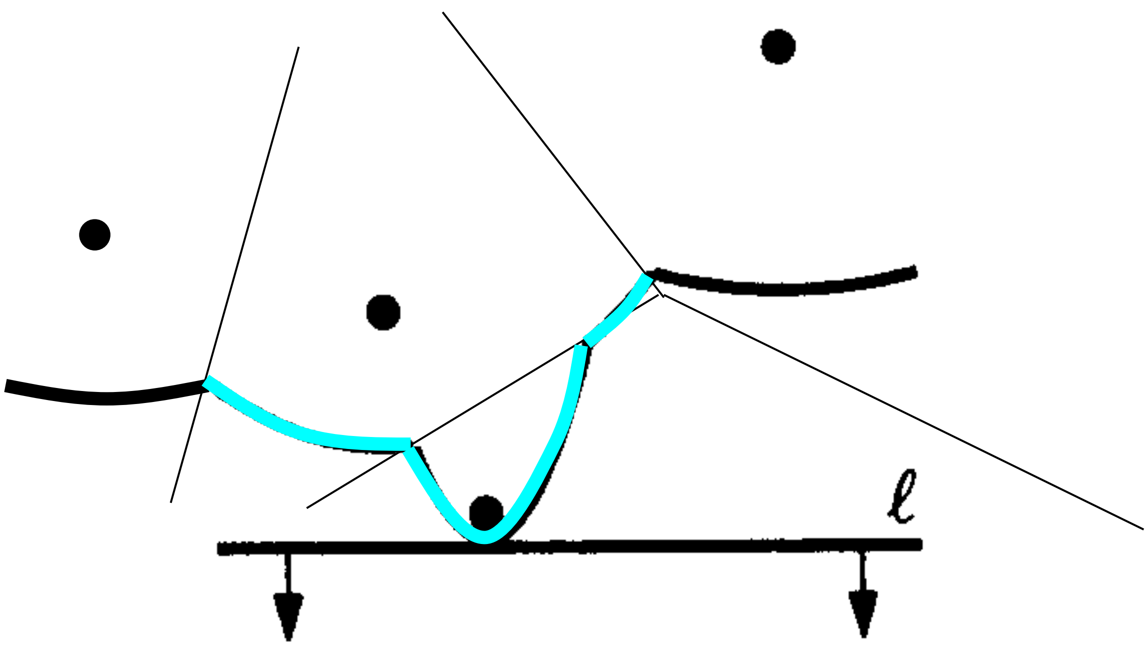
Checking for Potential Circle Events

- Scan for triple of consecutive arcs and determine if breakpoints converge
  - Triples with new arc in the middle do not have break points that converge



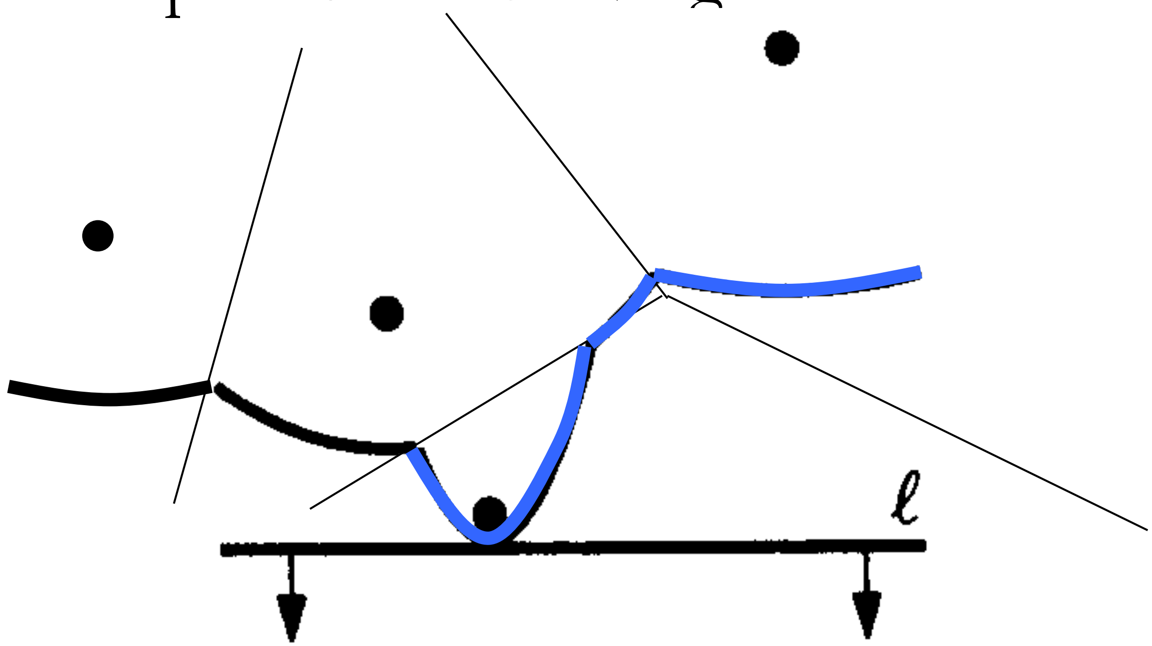
Checking for Potential Circle Events

- Scan for triple of consecutive arcs and determine if breakpoints converge
  - Triples with new arc in the middle do not have break points that converge



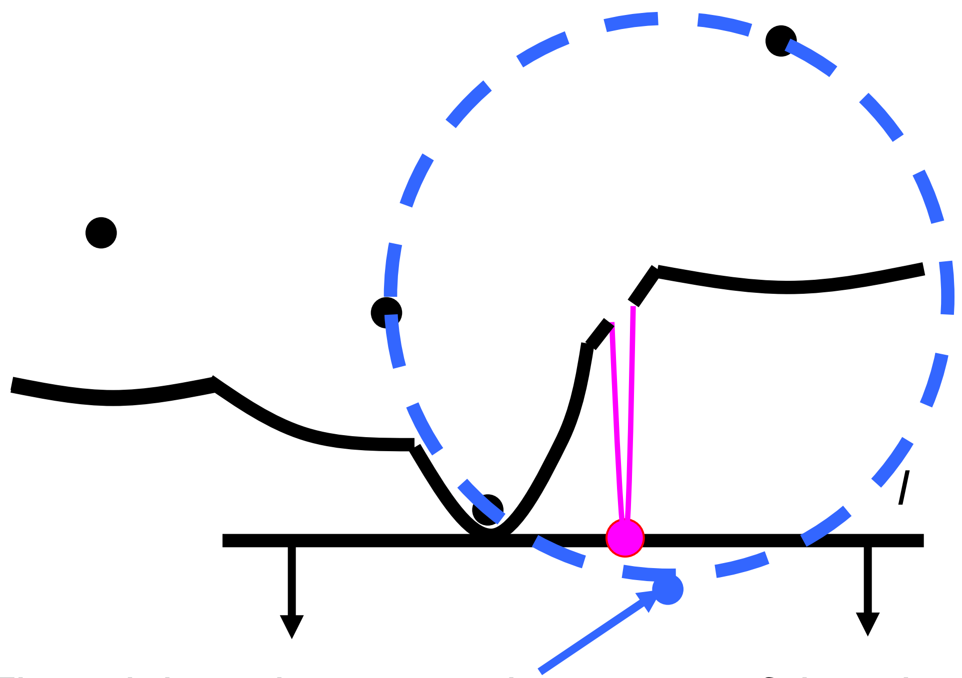
Checking for Potential Circle Events

- Scan for triple of consecutive arcs and determine if breakpoints converge
  - Triples with new arc in the middle do not have break points that converge



Converging break points may not always yield a circle event

- Appearance of a new site before the circle event makes the potential circle non-empty



(The original circle event becomes a *false alarm*)

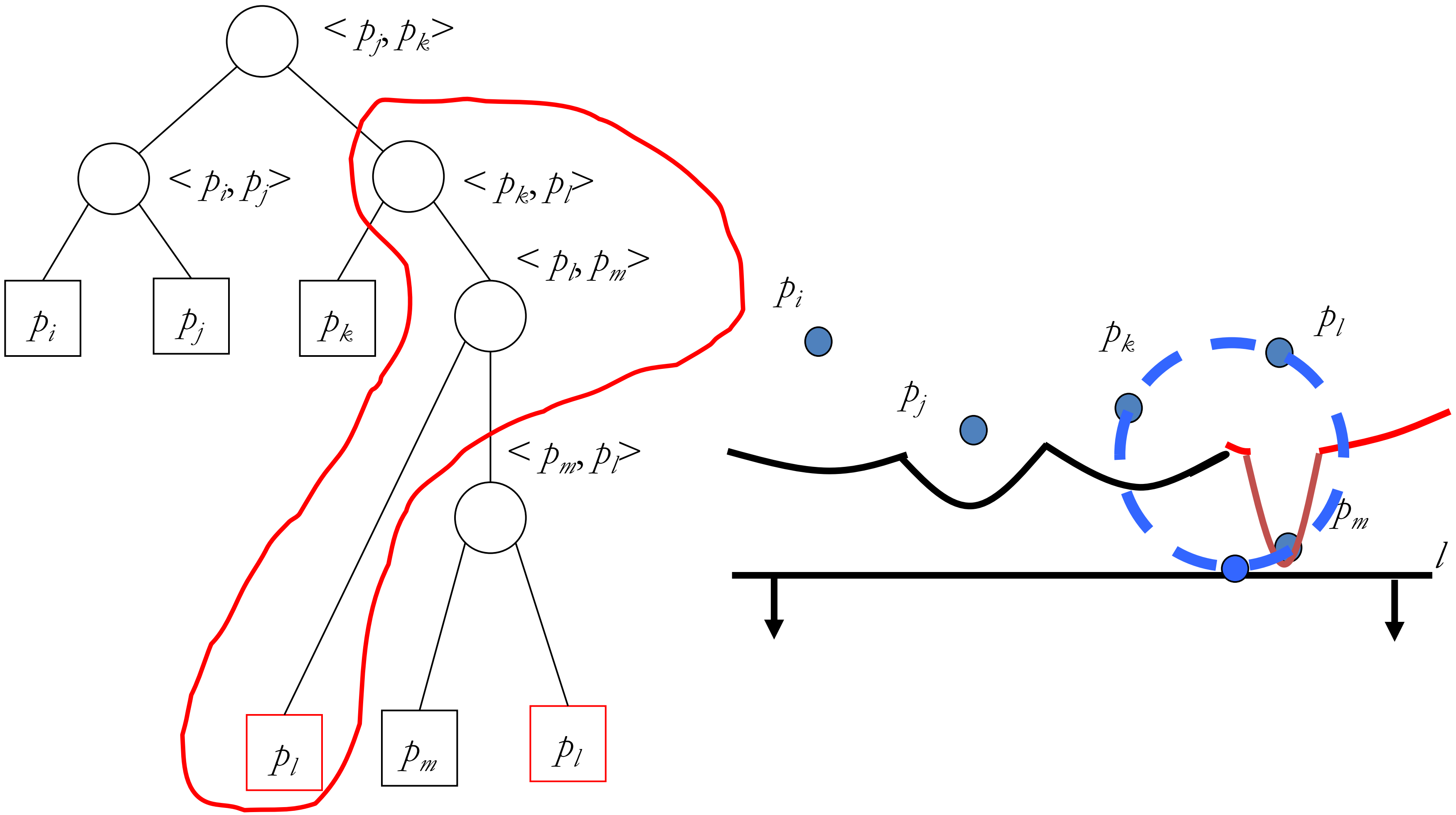
Handling Site Events

1. Locate the leaf representing the existing arc that is above the new site
   - Delete the potential circle event in the event queue
2. Break the arc by replacing the leaf node with a sub tree representing the new arc and break points
3. Add a new edge record in the doubly linked list
4. Check for potential circle event(s), add them to queue if they exist
   - Store in the corresponding leaf of T a pointer to the new circle event in the queue

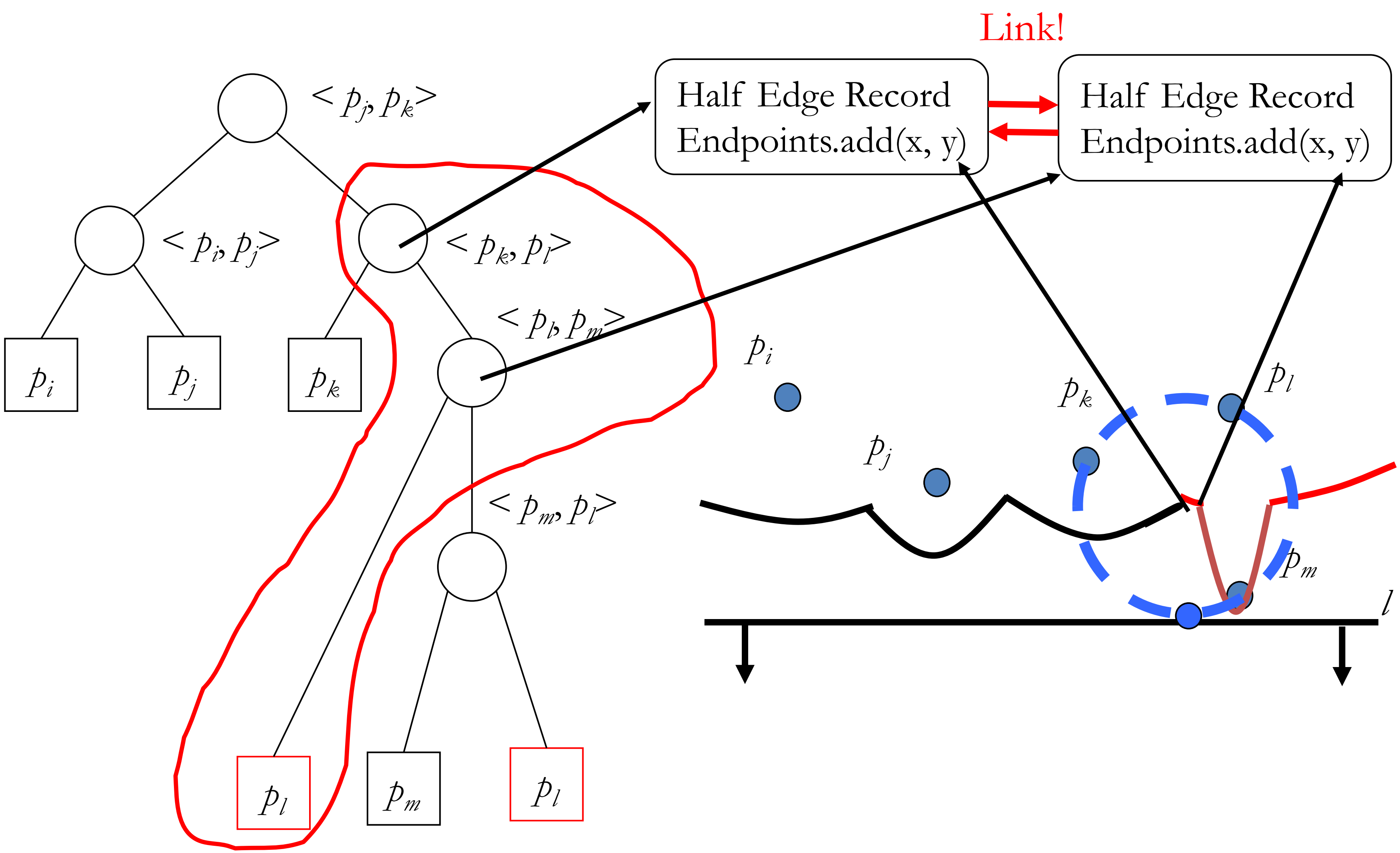
Handling Circle Events

1. Add vertex to corresponding edge record in doubly linked list
2. Delete from T the leaf node of the disappearing arc and its associated circle events in the event queue
3. Create new edge record in doubly linked list
4. Check the new triplets formed by the former neighboring arcs for potential circle events

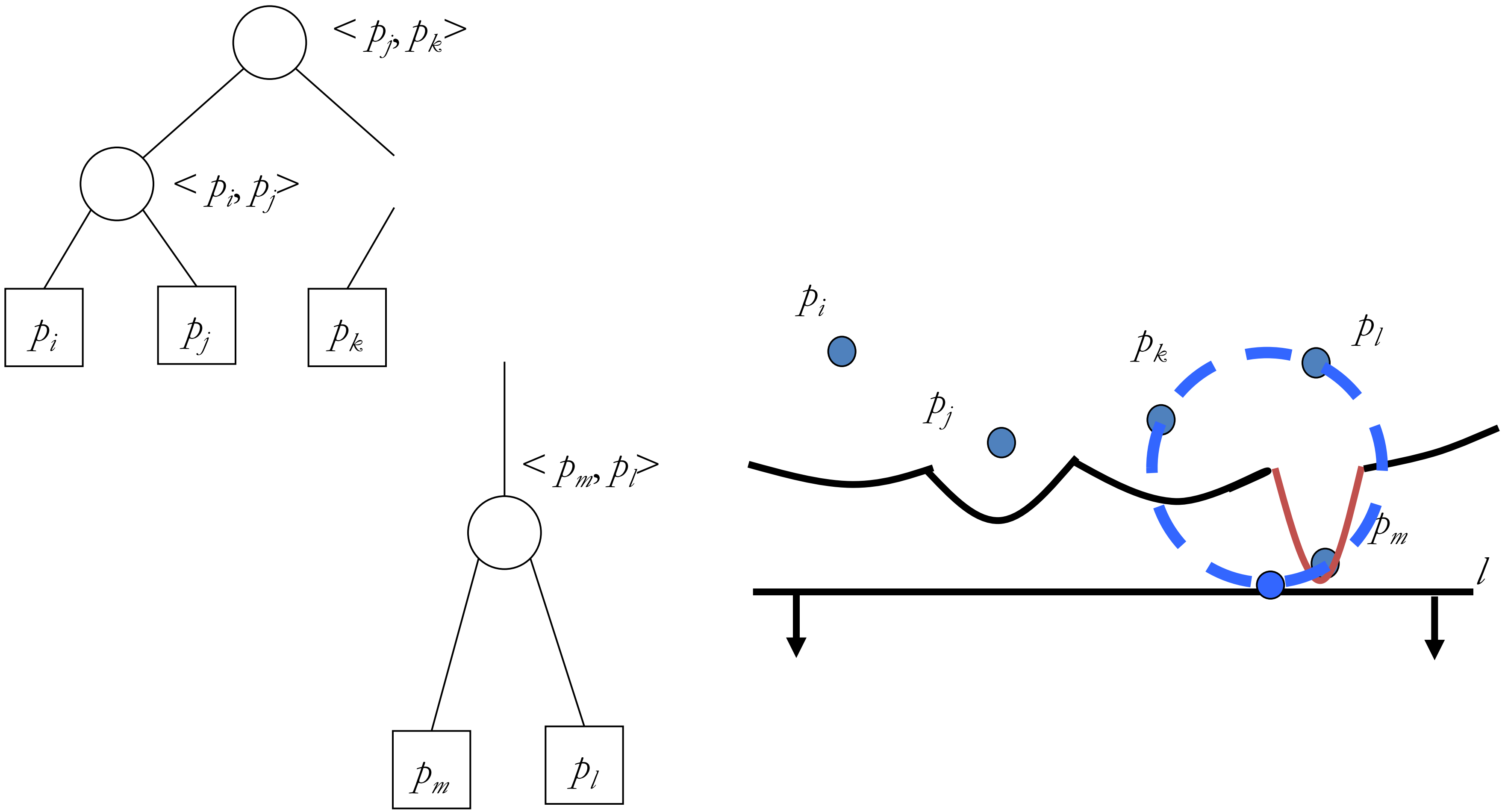
A Circle Event



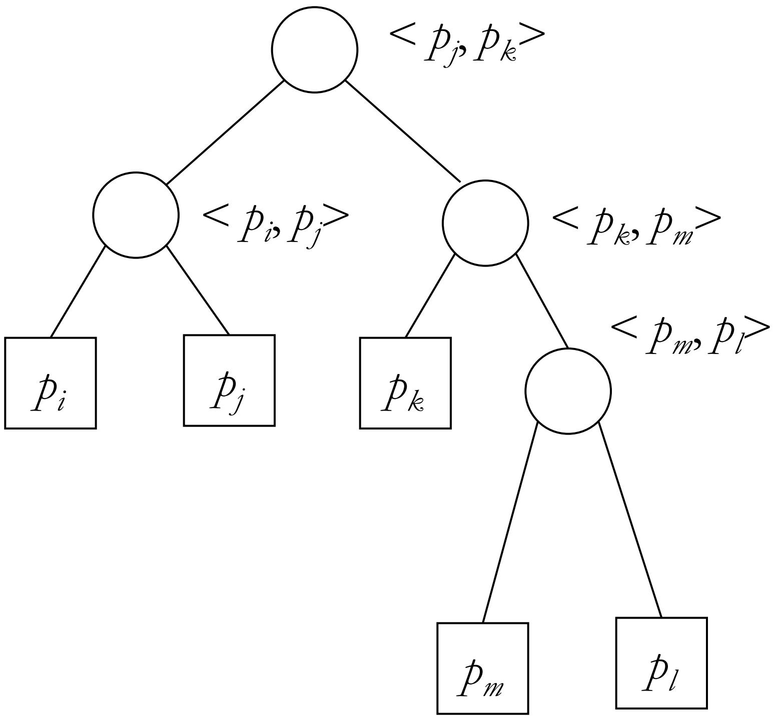
Add vertex to corresponding edge record

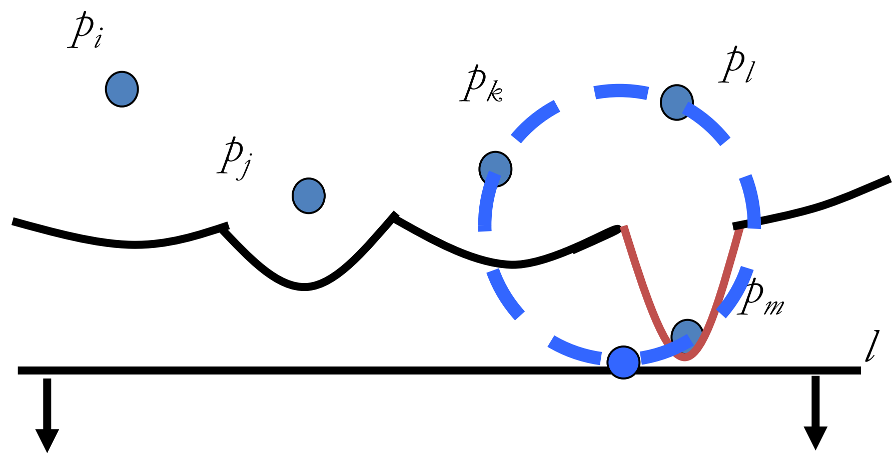


Deleting disappearing arc

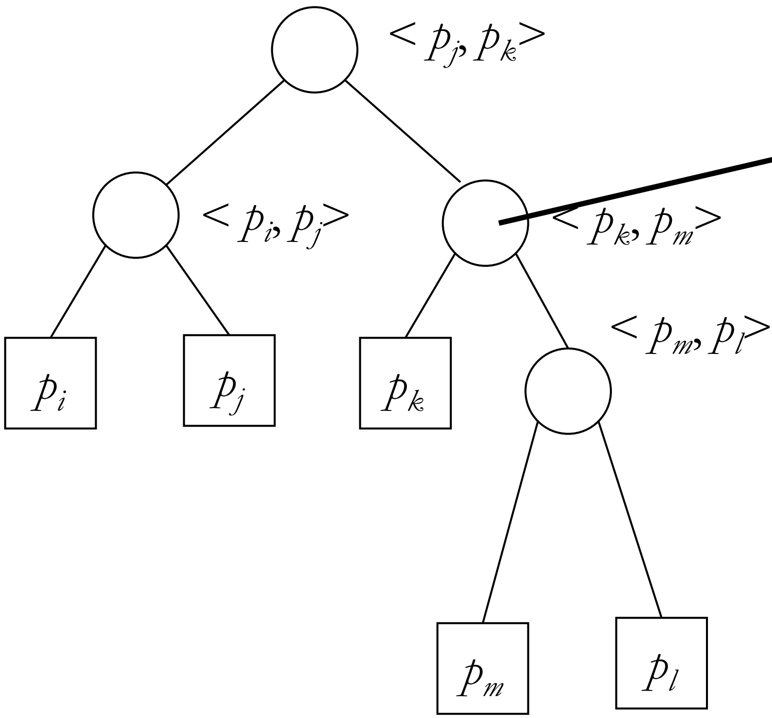


Deleting disappearing arc

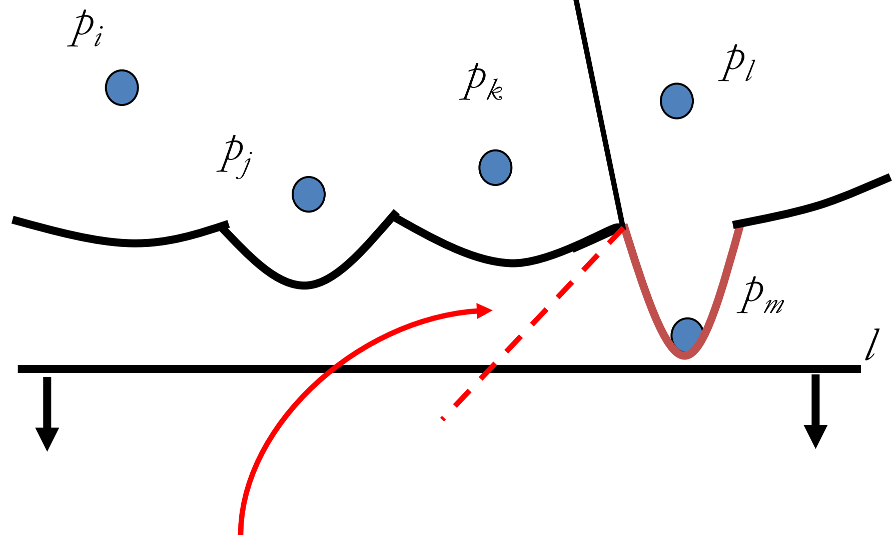




Create new edge record

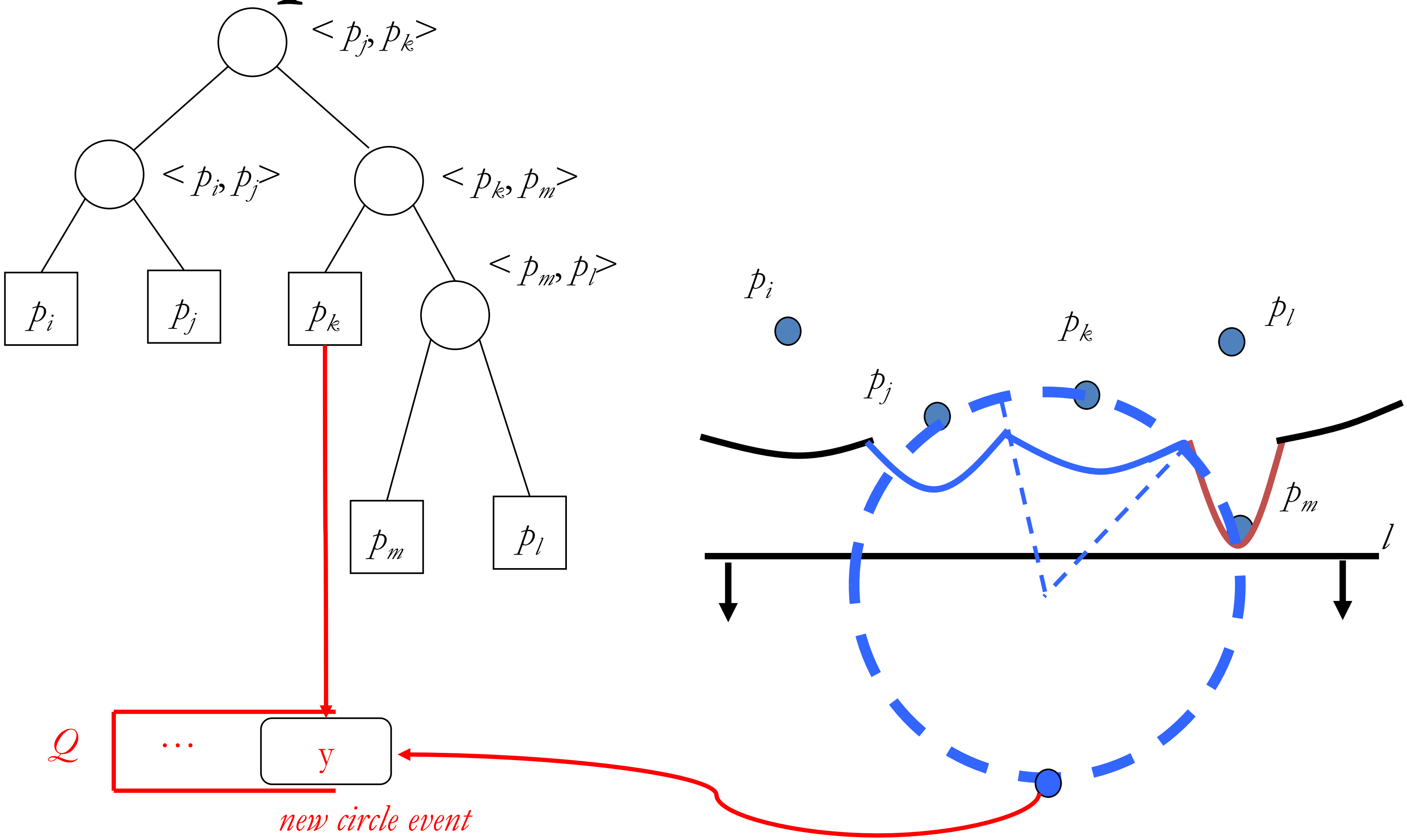


New Half Edge Record
Endpoints.add(x, y)



A new edge is traced out by the new break point $< p_k, p_m >$

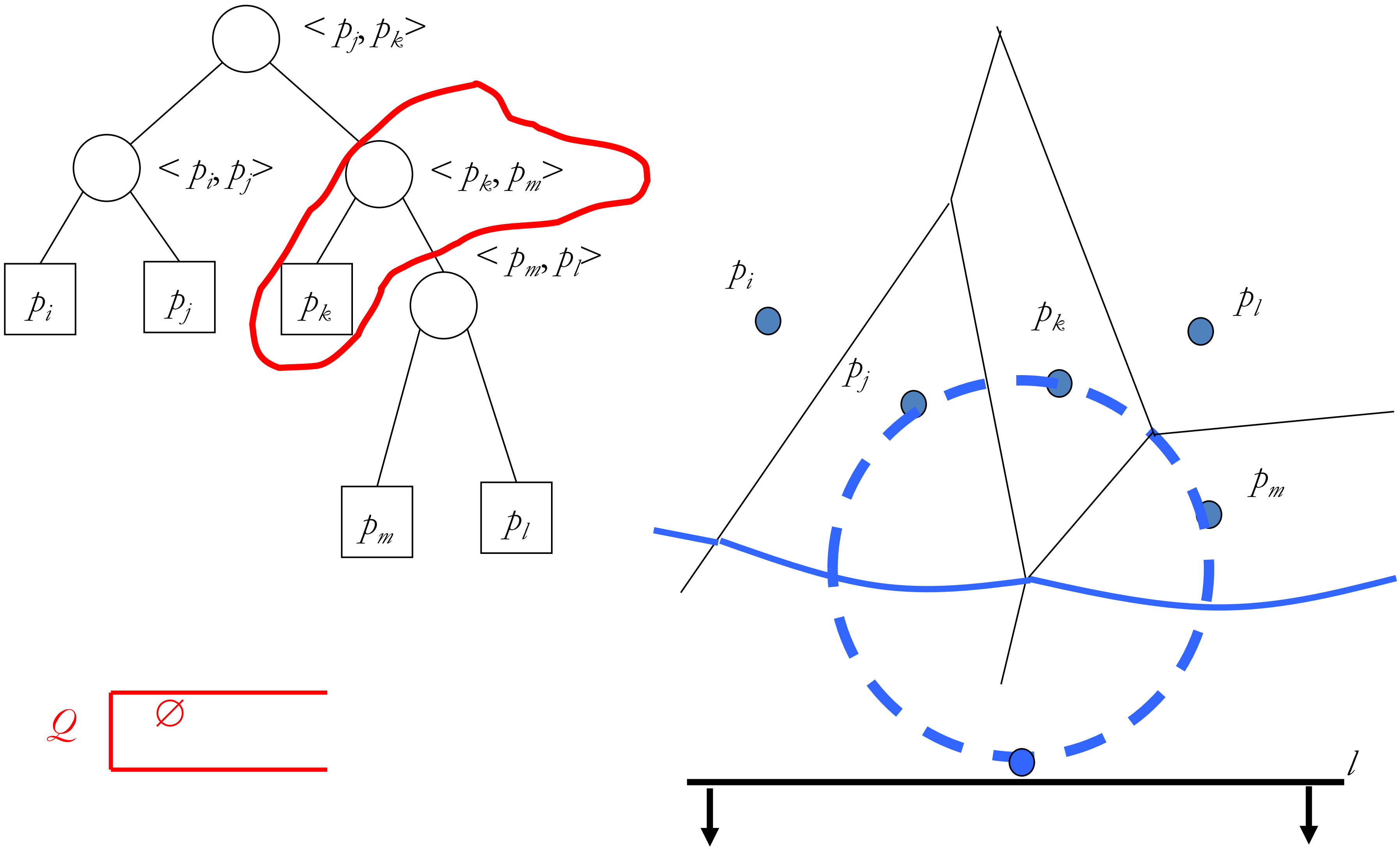
Check the new triplets for potential circle events



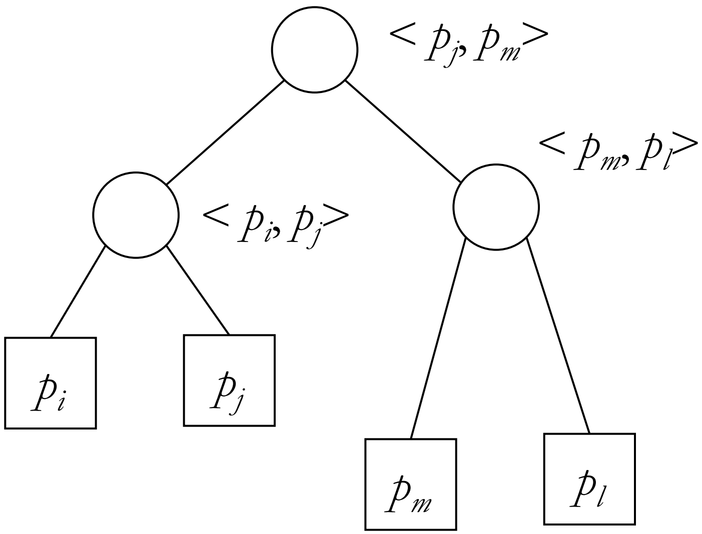
Minor Detail

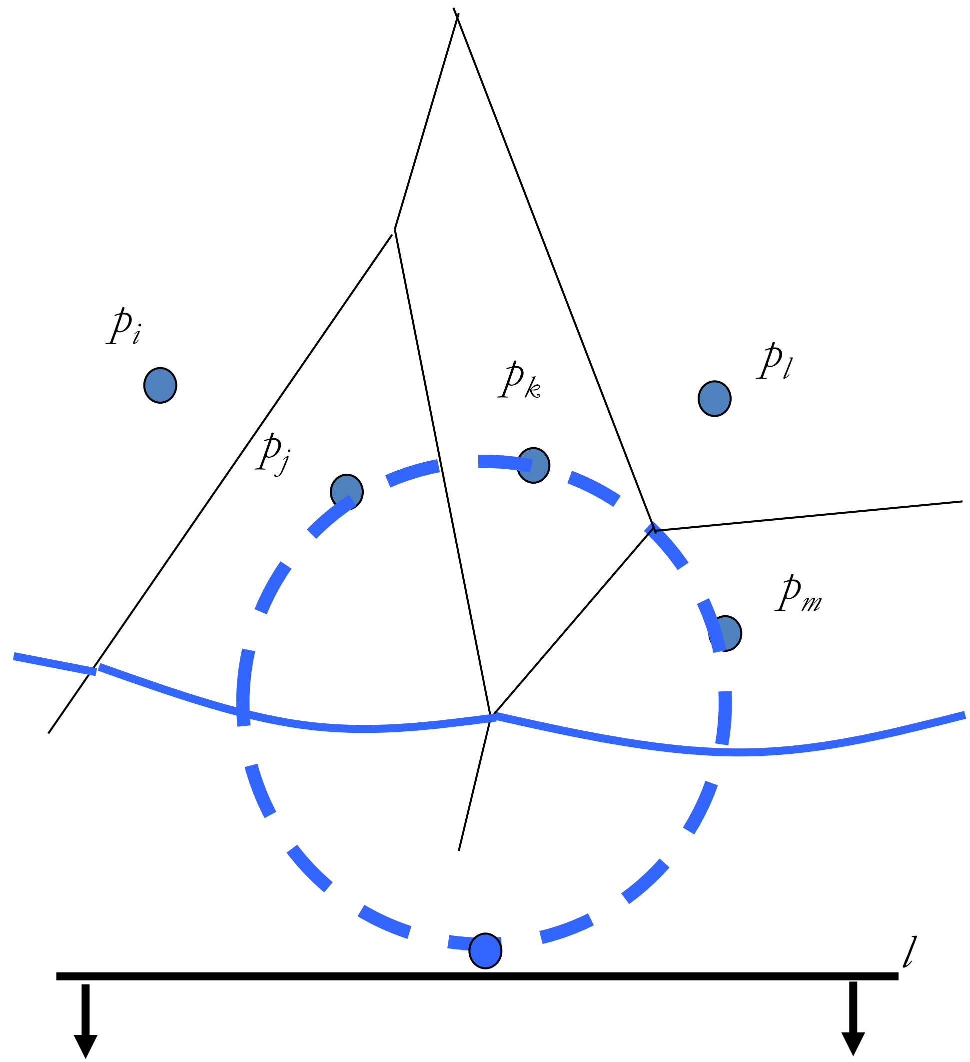
- Algorithm terminates when $Q = \emptyset$, but the beach line and its break points continue to trace the Voronoi edges
  - Terminate these "half-infinite" edges via a bounding box

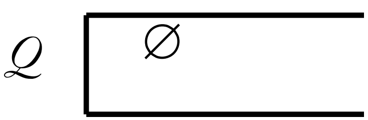
Algorithm Termination



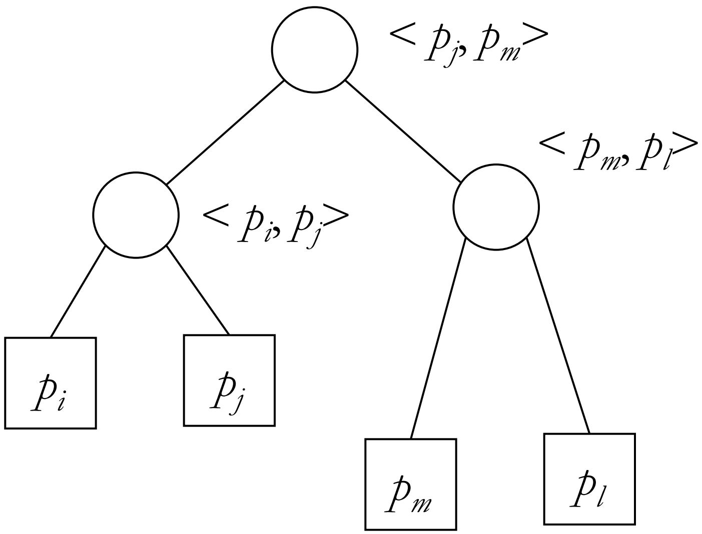
Algorithm Termination

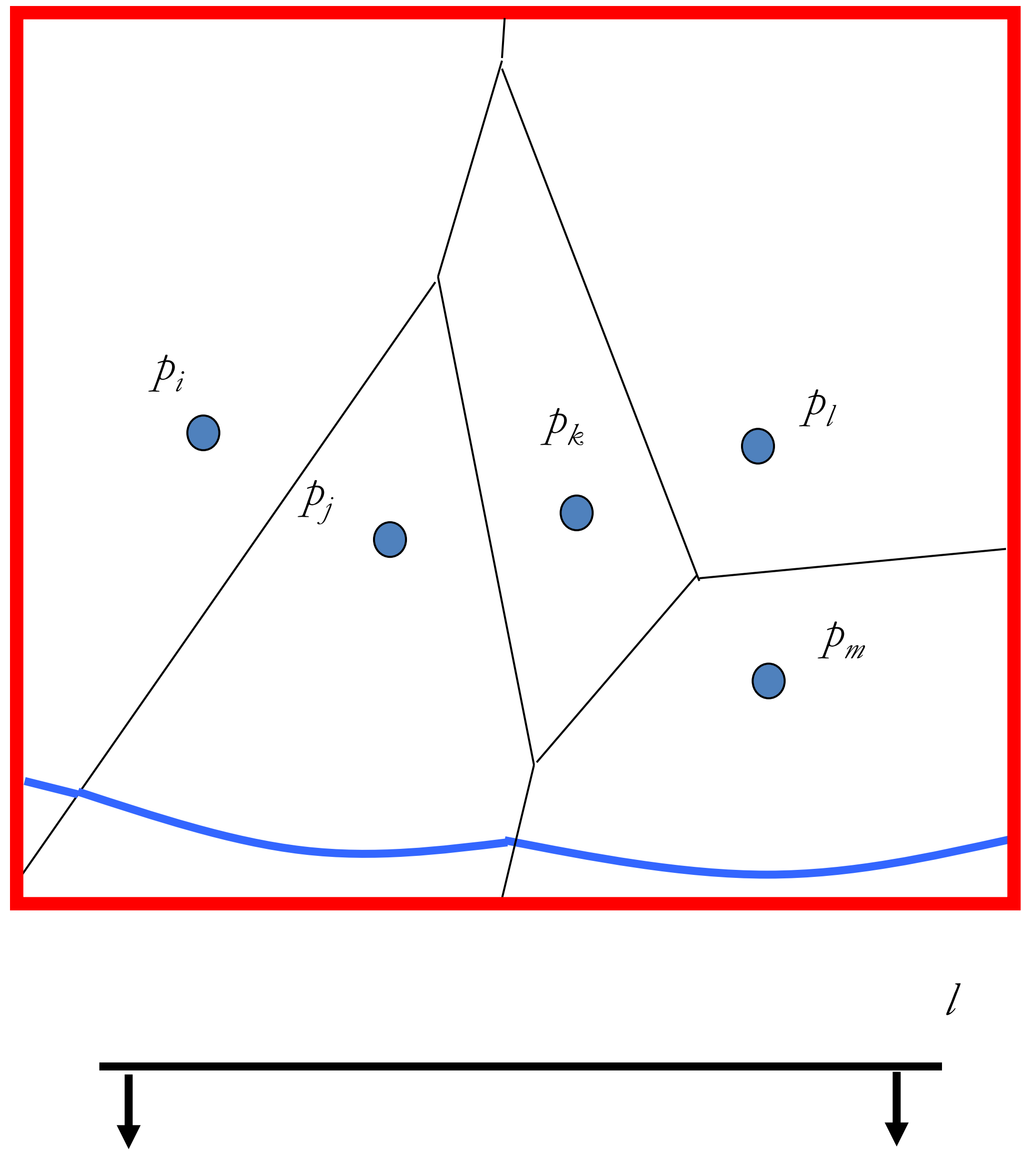




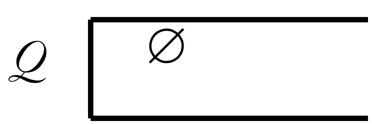


Algorithm Termination





Terminate half-lines with a bounding box!



Constructing Voronoi Diagrams

Running Time Analysis

Handling Site Events

Running Time

1. Locate the leaf representing the existing arc that is above the new site — O(log n)
   - Delete the potential circle event in the event queue — O(1)
2. Break the arc by replacing the leaf node with a sub tree representing the new arc and break points — O(1)
3. Add a new edge record in the link list — O(1)
4. Check for potential circle event(s), add them to queue if they exist
   - Store in the corresponding leaf of T a pointer to the new circle event in the queue

Handling Circle Events

	Running Time
1. Delete from T the leaf node of the disappearing arc and its associated circle events in the event queue	O(log *n*)
2. Add vertex record in doubly link list	O(1)
3. Create new edge record in doubly link list	O(1)
4. Check the new triplets formed by the former neighboring arcs for potential circle events	O(1)

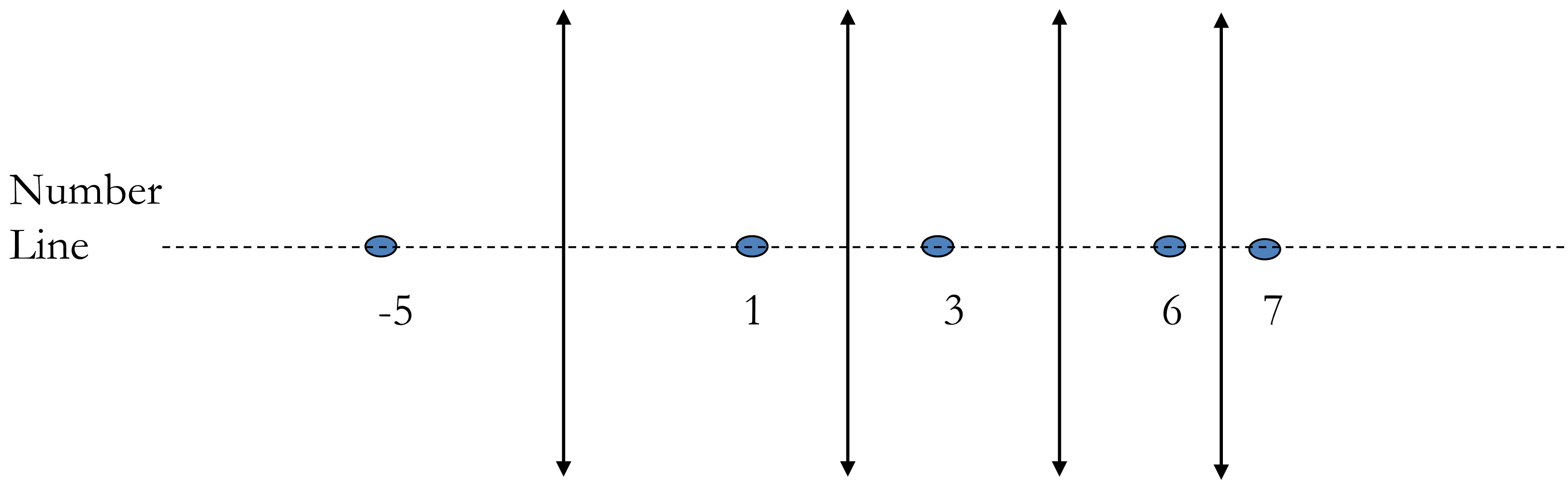
Total Running Time

- Each new site can generate at most two new arcs
  - →beach line can have at most $2n - 1$ arcs
  - →at most $O(n)$ site and circle events in the queue
- Site/Circle Event Handler $O(\log n)$

→ $O(n \log n)$ total running time

Is Fortune's Algorithm Optimal?

- We can sort numbers using any algorithm that constructs a Voronoi diagram!



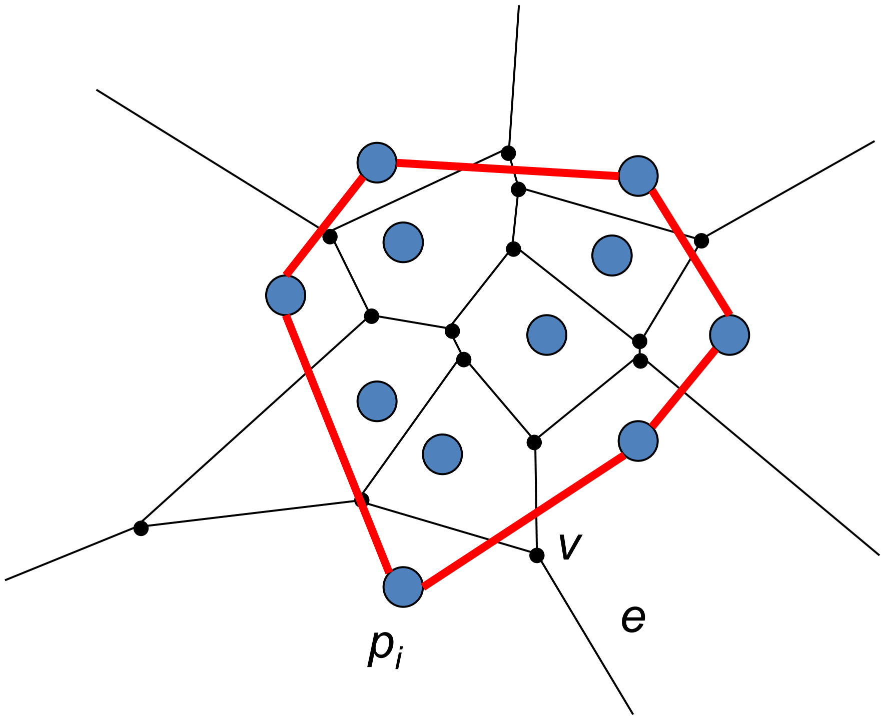
- Map input numbers to a position on the number line. The resulting Voronoi diagram is doubly linked list that forms a chain of unbounded cells in the left-to-right (sorted) order.

Constructing Voronoi Diagrams

Duality and degenerate cases

Voronoi Diagram/Convex Hull Duality

Sites sharing a half-infinite edge are convex hull vertices



Degenerate Cases

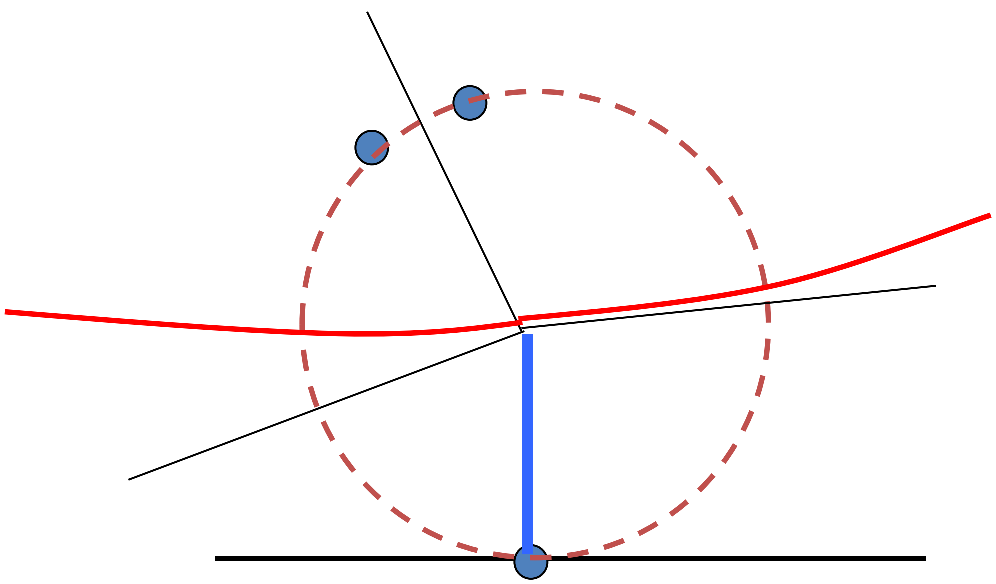
- Events in Q share the same y-coordinate
  - Can additionally sort them using x-coordinate
- Circle event involving more than 3 sites
  - Current algorithm produces multiple degree 3 Voronoi vertices joined by zero-length edges
  - Can be fixed in post processing

Degenerate Cases

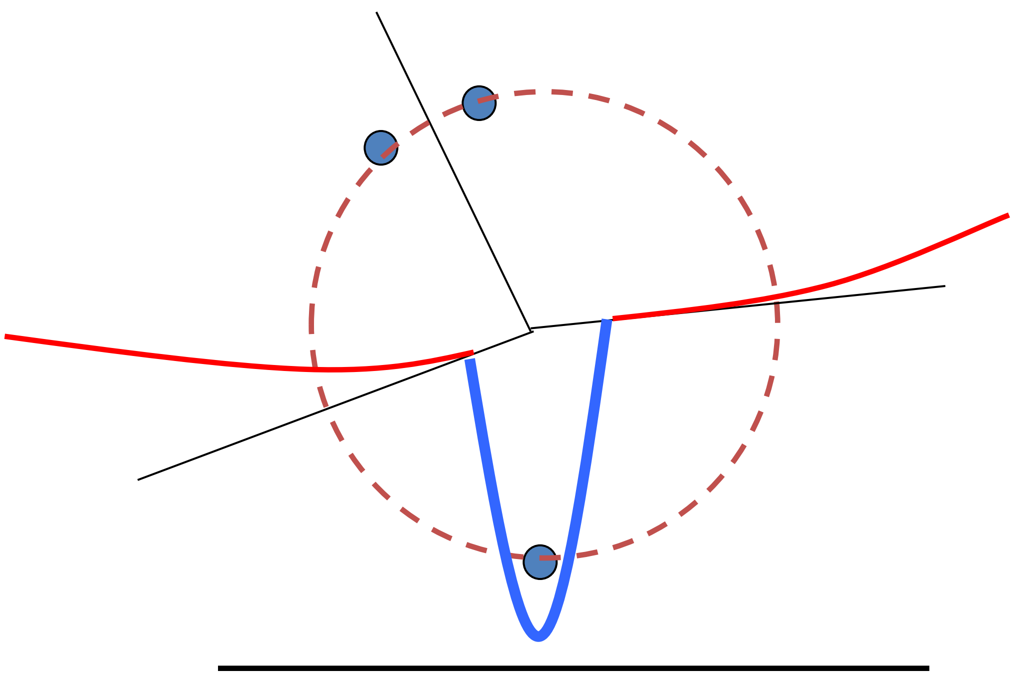
- Site points are collinear (break points neither converge or diverge)
  - Bounding box takes care of this
- One of the sites coincides with the lowest point of the circle event
  - Do nothing

Site coincides with circle event: the same algorithm applies!

1. New site detected
2. Break one of the arcs an infinitesimal distance away from the arc's end point



Site coincides with circle event



Summary

- Voronoi diagram is a useful planar subdivision of a discrete point set
- Voronoi diagrams have linear complexity and can be constructed in $O(n \log n)$ time
- Fortune's algorithm (optimal)