CS633 Lecture 09
Voronoi Diagram

Jyh-Ming Lien

Department of Computer Science
George Mason University

Based on Allen Miu's lecture notes

Independently Rediscovered Many Times

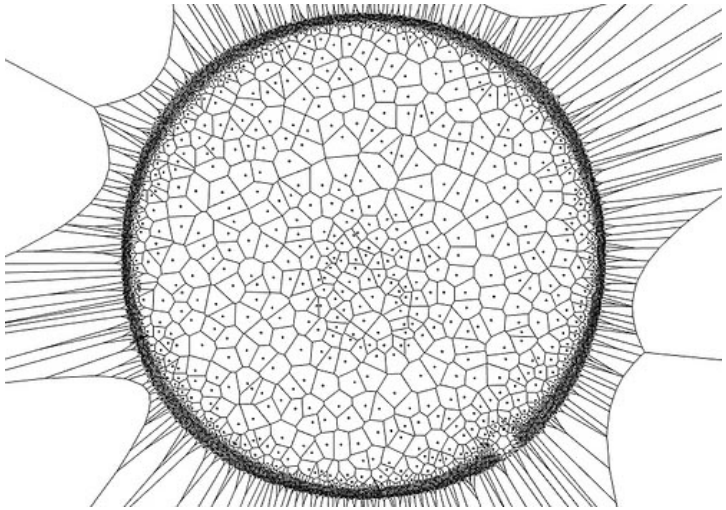
It is a fundamental concept

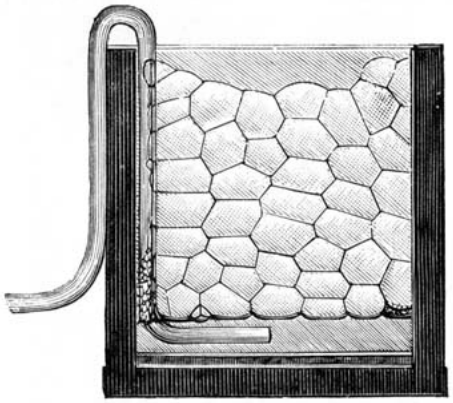
Descartes	Astronomy	1644	“Heavens”
Dirichlet	Math	1850	Dirichlet tesselation
Voronoi	Math	1908	Voronoi diagram
Boldyrev	Geology	1909	area of influen polygons
Thiessen	Meteorology	1911	Theissen polygons
Niggli	Crystallography	1927	domains of action
Wigner & Seitz	Physics	1933	Wigner-Seitz regions
Frank & Casper	Physics	1958	atom domains
Brown	Ecology	1965	areas potentially available
Mead	Ecology	1966	plant polygons
Hoofd et al.	Anatomy	1985	capillary domains
Icke	Astronomy	1987	Voronoi diagram

Kenneth E. Hoff III, Tim Culver, John Keyser, Ming Lin, and Dinesh Manocha, 99

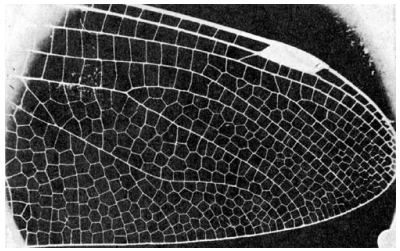
Fun Stuff

- Paul Chew's Jave applet
  - http://www.cs.cornell.edu/Info/People/chew/Delaunay.html
- Simon Barber's flash
  - http://www.quasimondo.com/archives/voronoi1.html
- FLIGHT404's Blog
  - http://www.flight404.com/blog/?p=82
- Scott Snibbe's Blog
  - http://www.snibbe.com/scott/bf/index.htm





Soap Bubbles



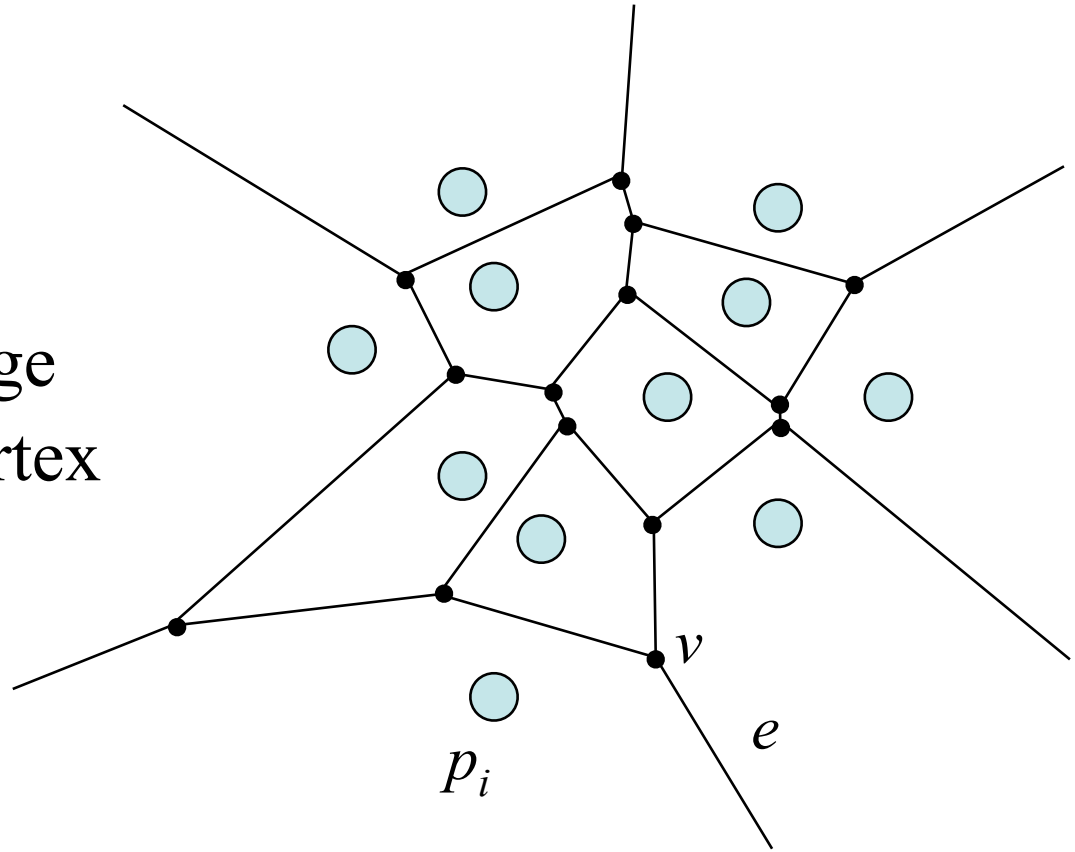
dragonfly's wing

Post Office: What is the area of service?

p_i : site

e : Voronoi edge

v : Voronoi vertex

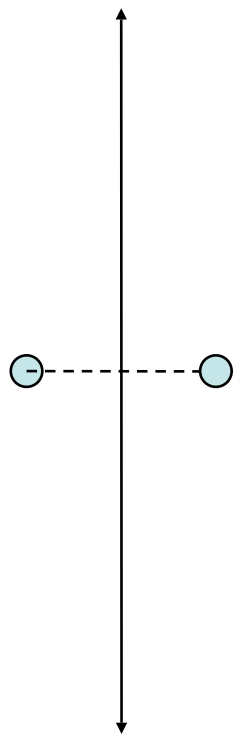


Definition of Voronoi Diagram

- Let P be a set of n distinct points (sites) in the plane.
- The Voronoi diagram of P is the subdivision of the plane into n cells, one for each site.
- A point q lies in the cell corresponding to a site $p_i \in P$ *iff* Euclidean_Distance(q, p_i) < Euclidean_distance(q, p_j), for each $p_i \in P$, $j \neq i$.

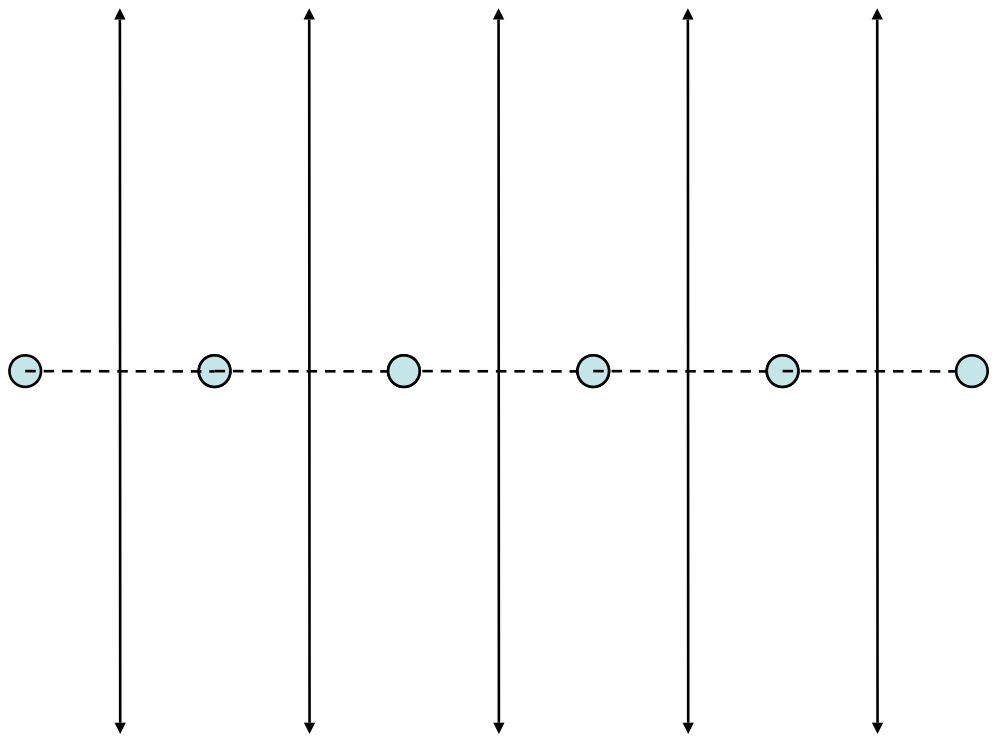
1 site

Two sites



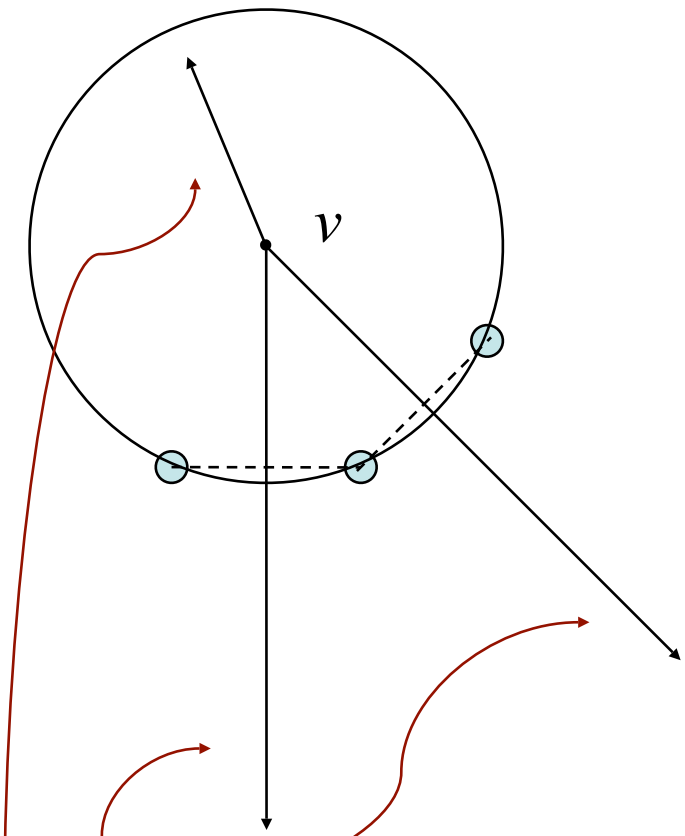
Voronoi Diagram is a line that extends infinitely in both directions, and the two half planes on either side.

Collinear sites



Non-collinear sites

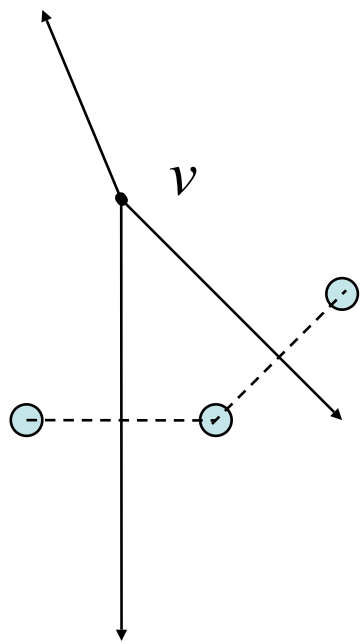
A vertex has degree ≥ 3



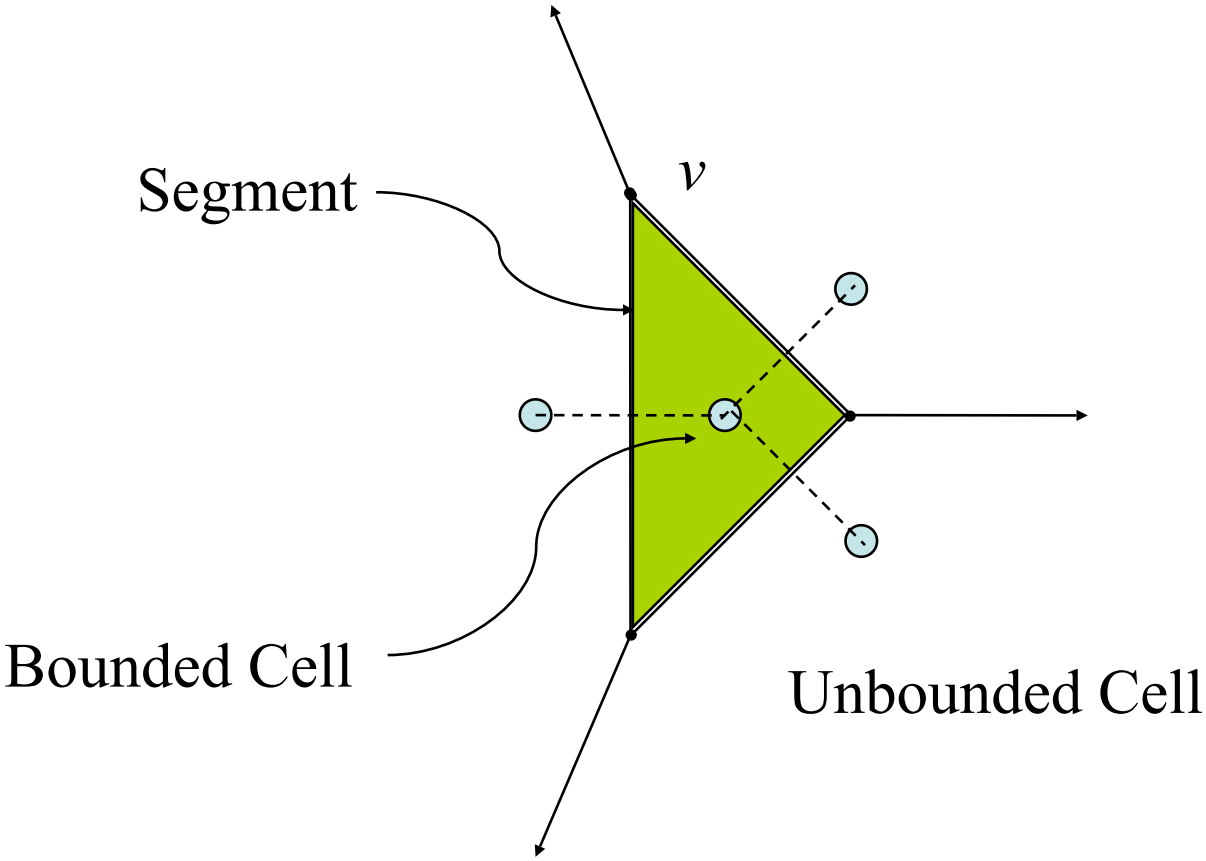
Half lines

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

Voronoi Cells and Segments



Voronoi Cells and Segments

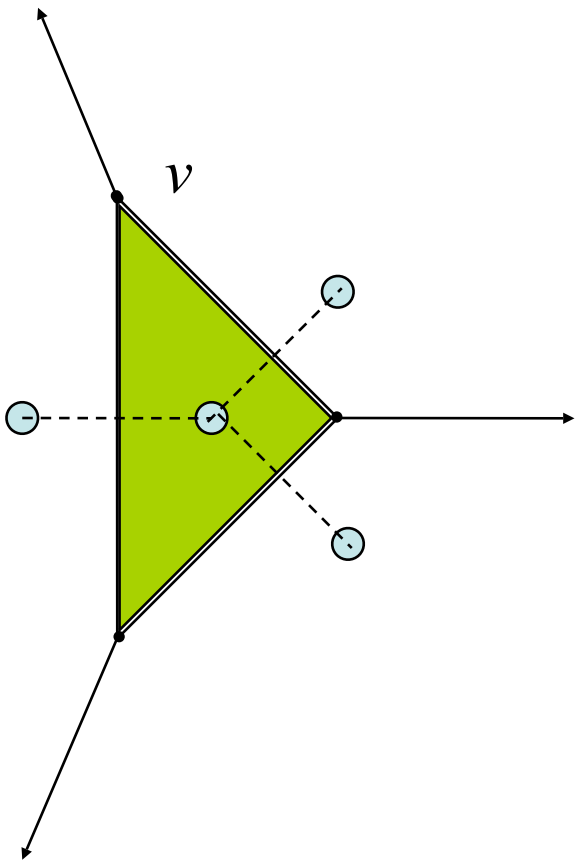


Who wants to be a Millionaire?

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

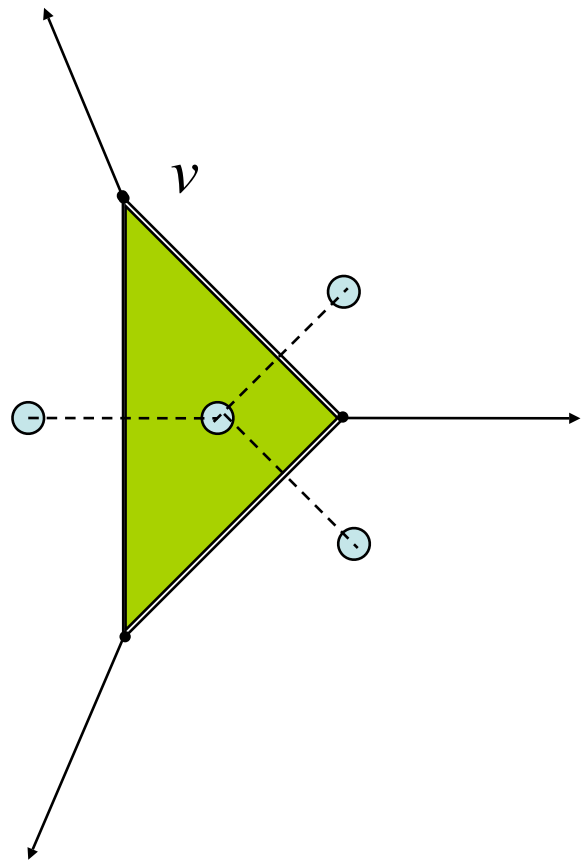


Who wants to be a Millionaire?

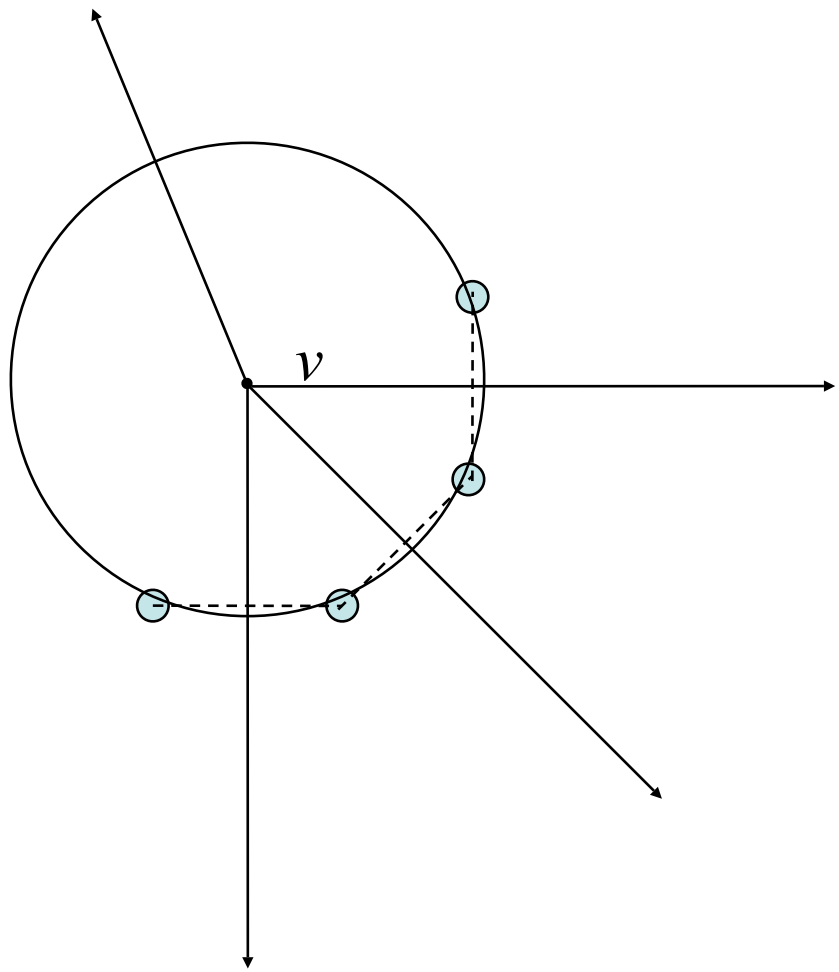
Which of the following is true for
2-D Voronoi diagrams?

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!

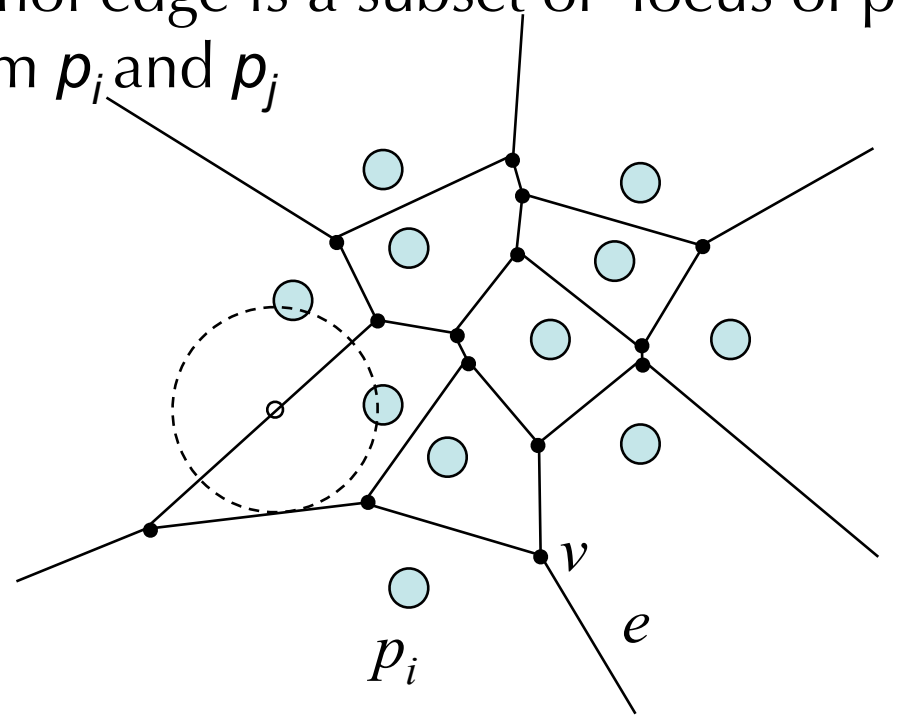


Summary of Voronoi Properties

A point q lies on a Voronoi edge between sites p_i and p_j *iff* 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

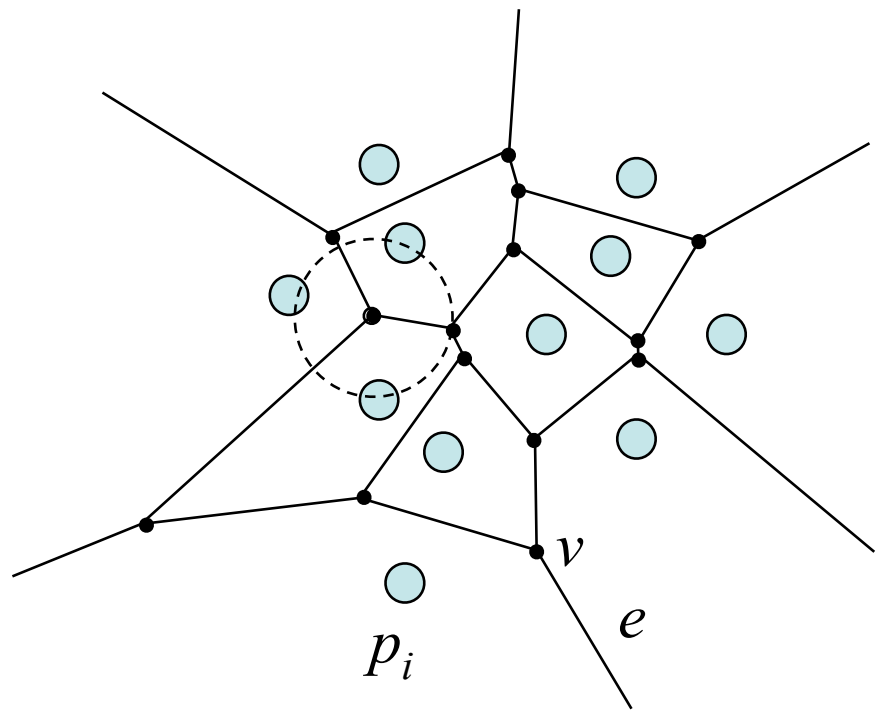


Summary of Voronoi Properties

A point q is a vertex *iff* 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



Outline

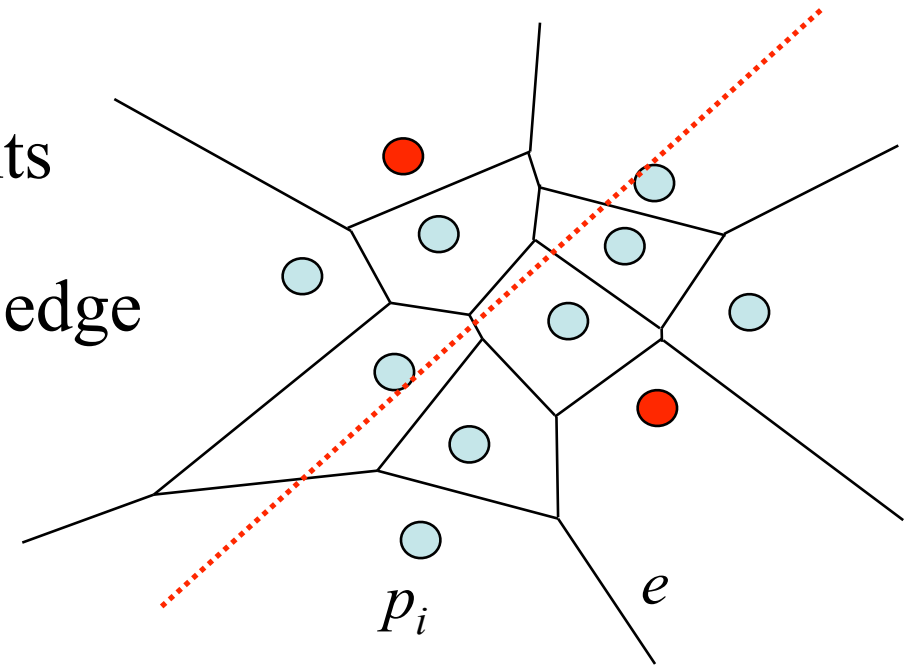
- Definitions and Examples
- Properties of Voronoi diagrams
- **Complexity of Voronoi diagrams**
- Constructing Voronoi diagrams
  - Intuitions
  - Data Structures
  - Algorithm
- Running Time Analysis
- Duality and degenerate cases

Linear complexity {$|v|$, $|e|$ = O(n)}

Intuition: Not all bisectors are Voronoi edges!

p_i : site points

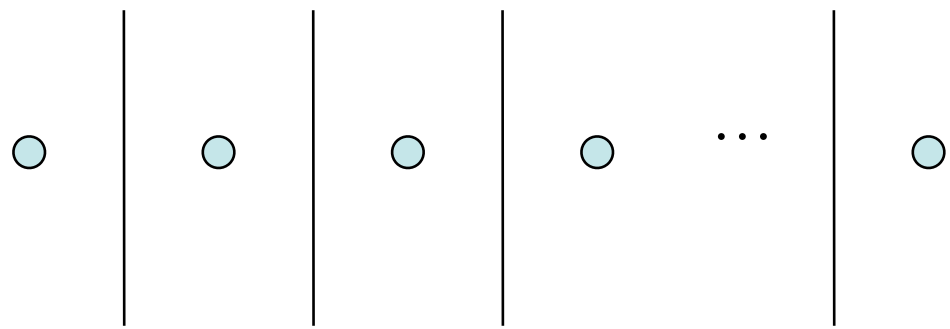
e : Voronoi edge



Linear complexity {$|v|$, $|e| = O(n)$}

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

Proof: (Easy Case)



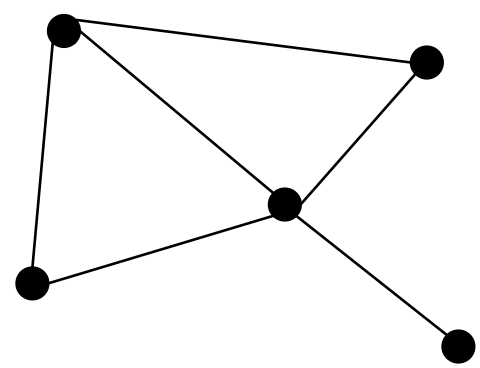
Collinear sites: $|v| = 0$, $|e| = n - 1$

Linear complexity {$|v|$, $|e|$ = O(n)}

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

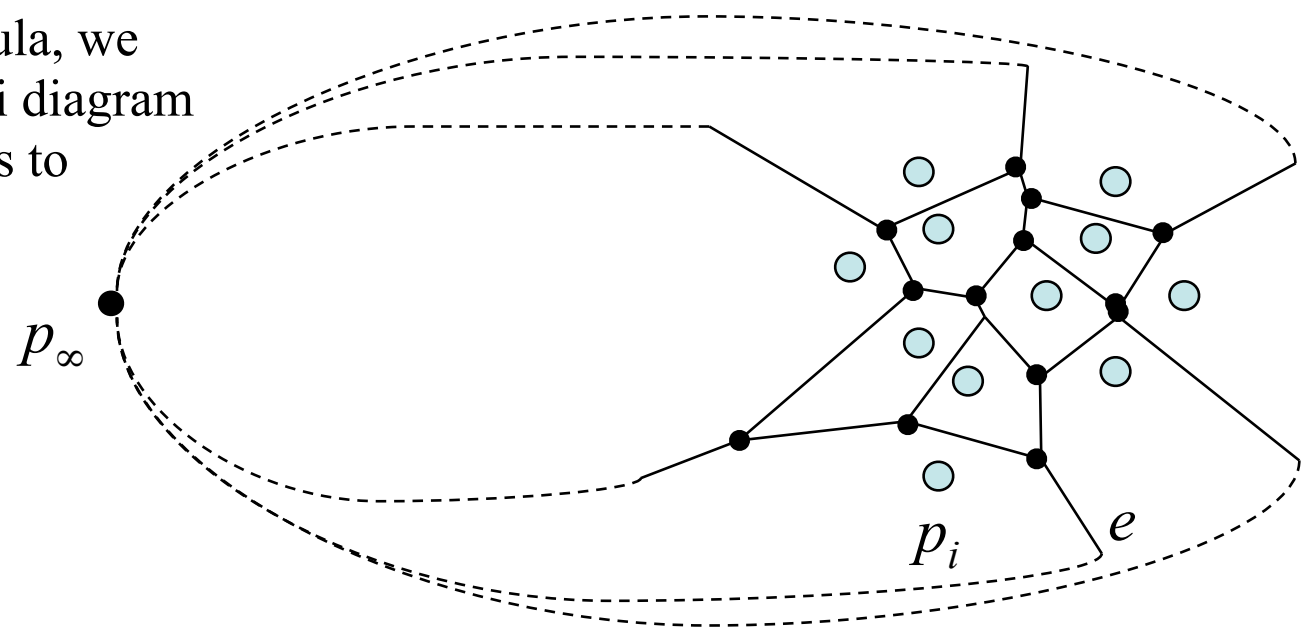
Linear complexity $\{|v|, |e| = O(n)\}$

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.



Linear complexity {|*v*|, |*e*| = O(*n*)}

Moreover,

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

and

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

so

$$\boxed{\begin{aligned} &2 \cdot |e| \geq 3(|v|+1) \\ &(|v|+1) - |e| + n = 2 \end{aligned}}$$

together with

we get, for $n \geq 3$

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

Outline

- Definitions and Examples
- Properties of Voronoi diagrams
- Complexity of Voronoi diagrams
- **Constructing Voronoi diagrams**
  - **Intuitions**
  - **Data Structures**
  - **Algorithm**
- Running Time Analysis
- Duality and degenerate cases

Constructing Voronoi Diagrams

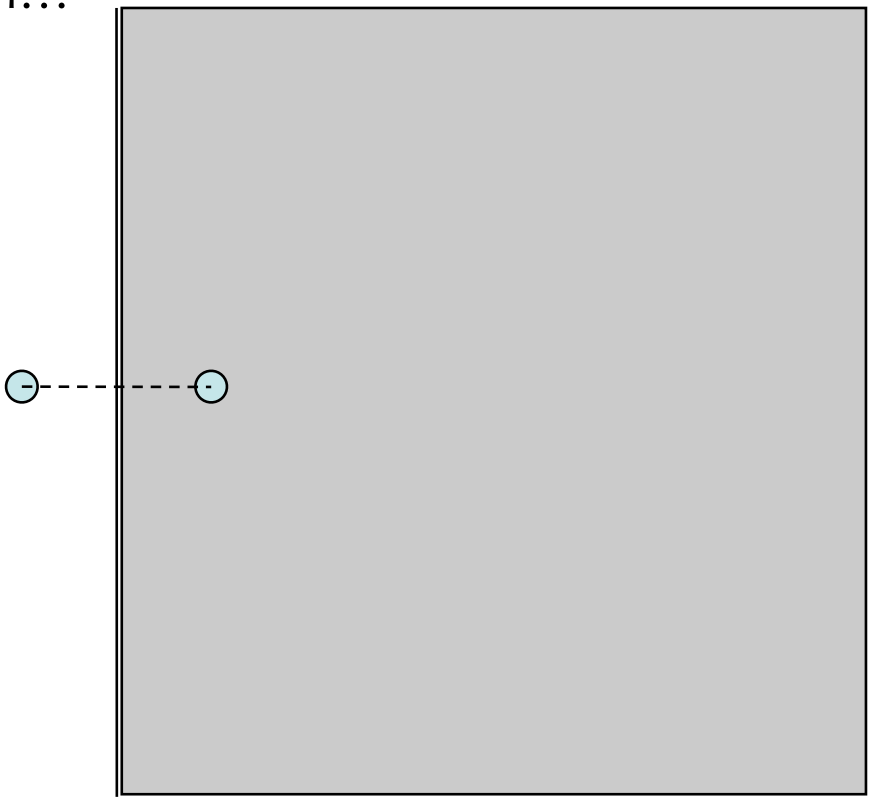
Brute force algorithm

- a half plane intersection…

Constructing Voronoi Diagrams

Brute force algorithm

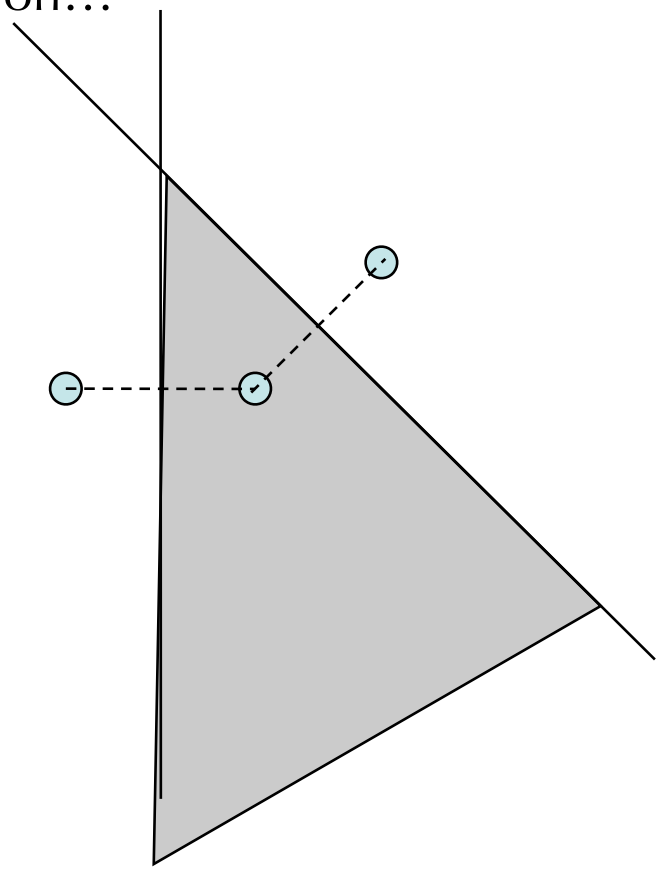
– a half plane intersection…



Constructing Voronoi Diagrams

Brute force algorithm

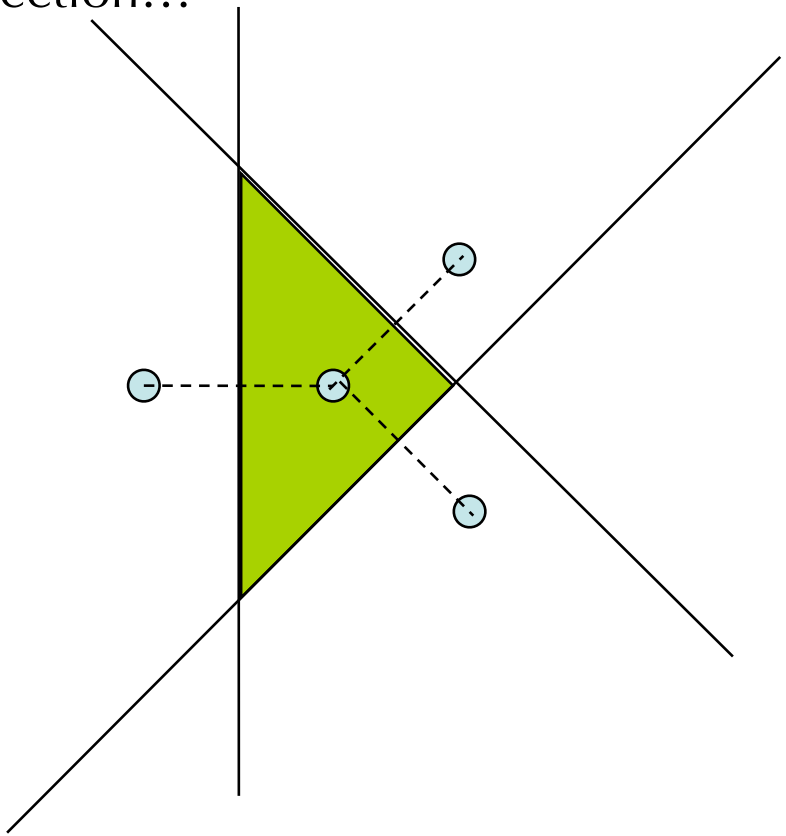
– a half plane intersection…



Constructing Voronoi Diagrams

Brute force algorithm

– a half plane intersection…



Repeat for each site

Running Time:
O($n^2 \log n$)

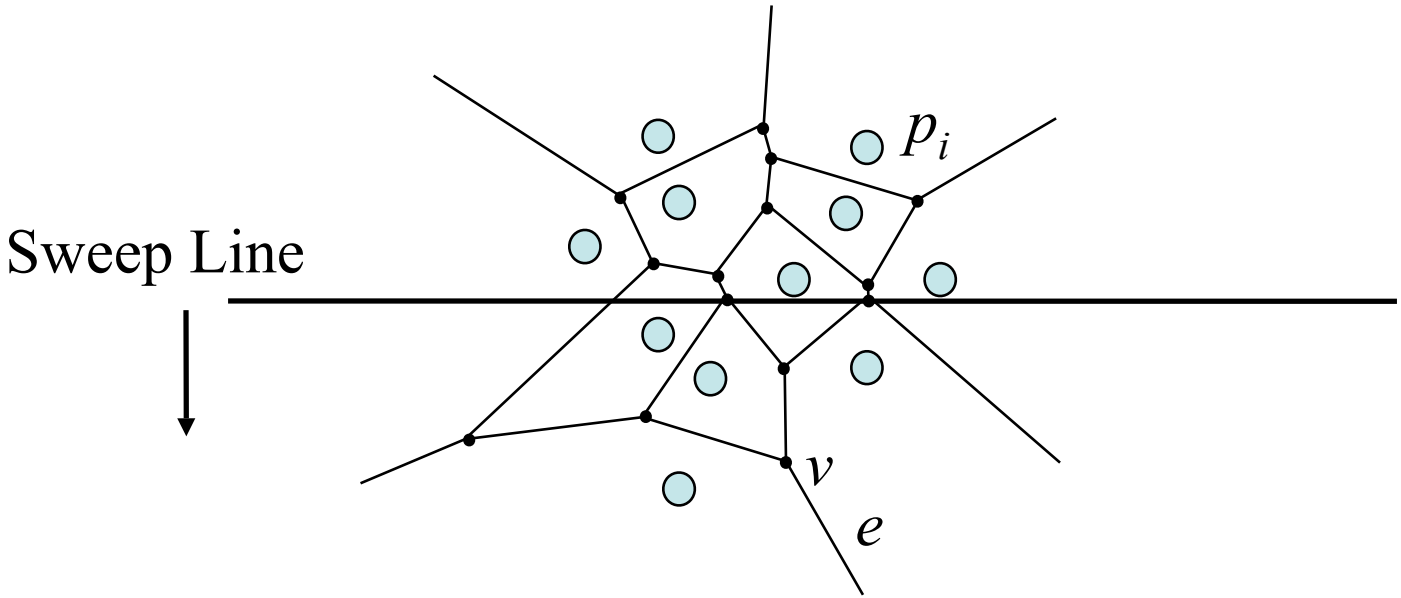
Constructing Voronoi Diagrams

- We should be able to do better
  - the linear complexity of Voronoi diagram
- Fortune's Algorithm '87
  - Sweep line algorithm
    - Voronoi diagram constructed as horizontal line sweeps the set of sites from top to botton
    - Maintains portion of diagram which cannot change due to sites below sweep line, keeping track of incremental changes for eac site (and Voronoi vertex) it "sweeps"

Steve Fortune
Bell lab

Constructing Voronoi Diagrams

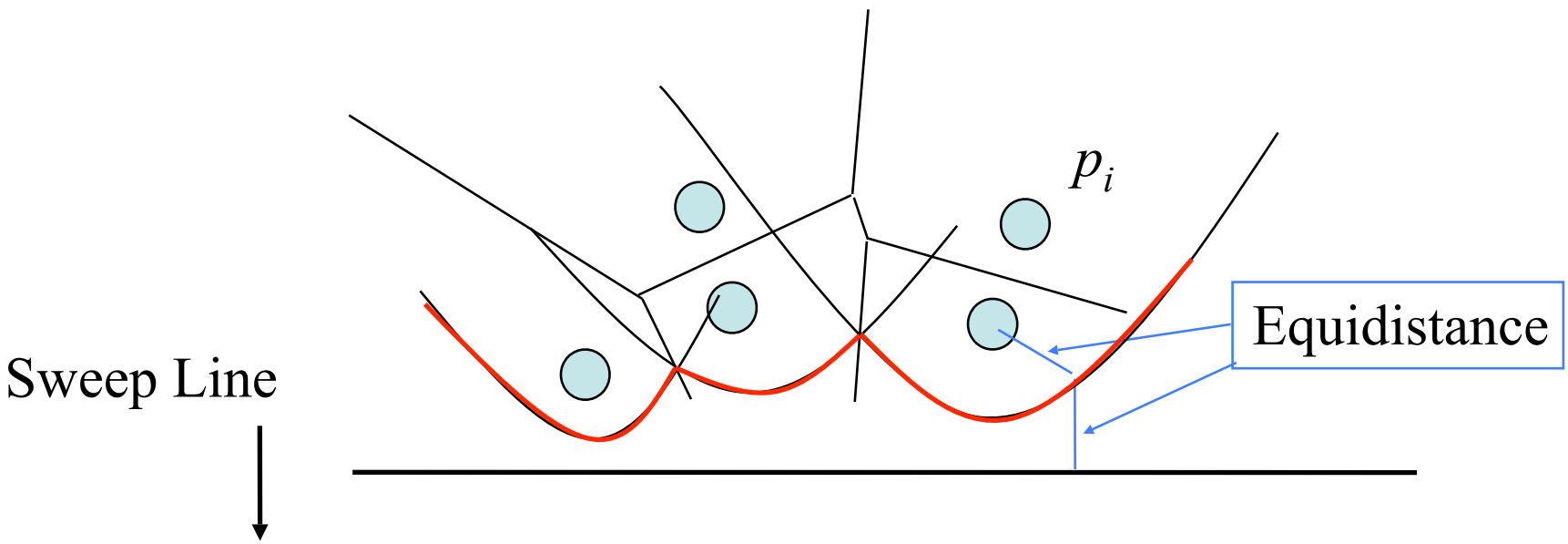
What is the invariant we are looking for?



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).

Constructing Voronoi Diagrams

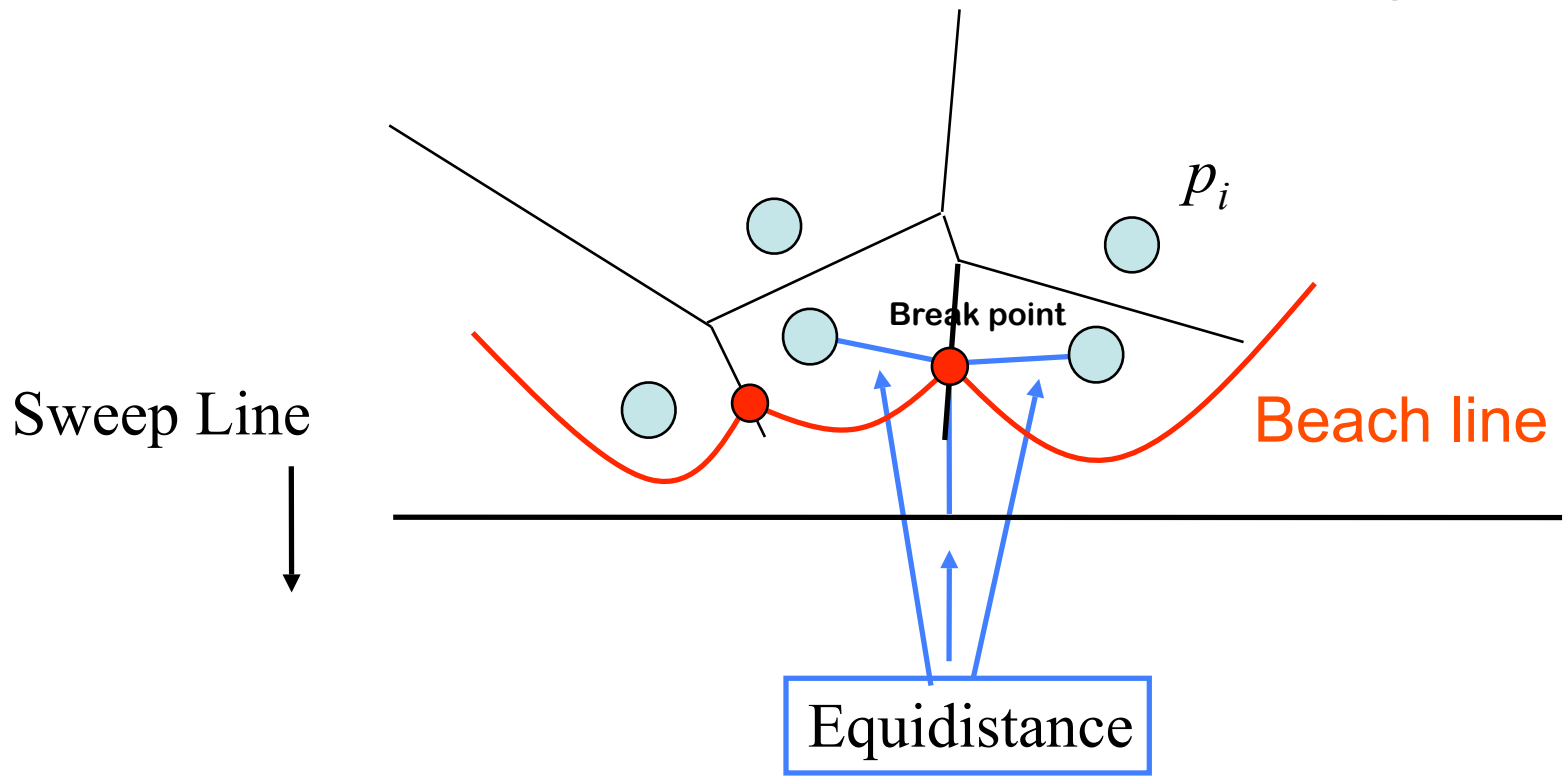
Which points are closer to a site above the sweep line



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

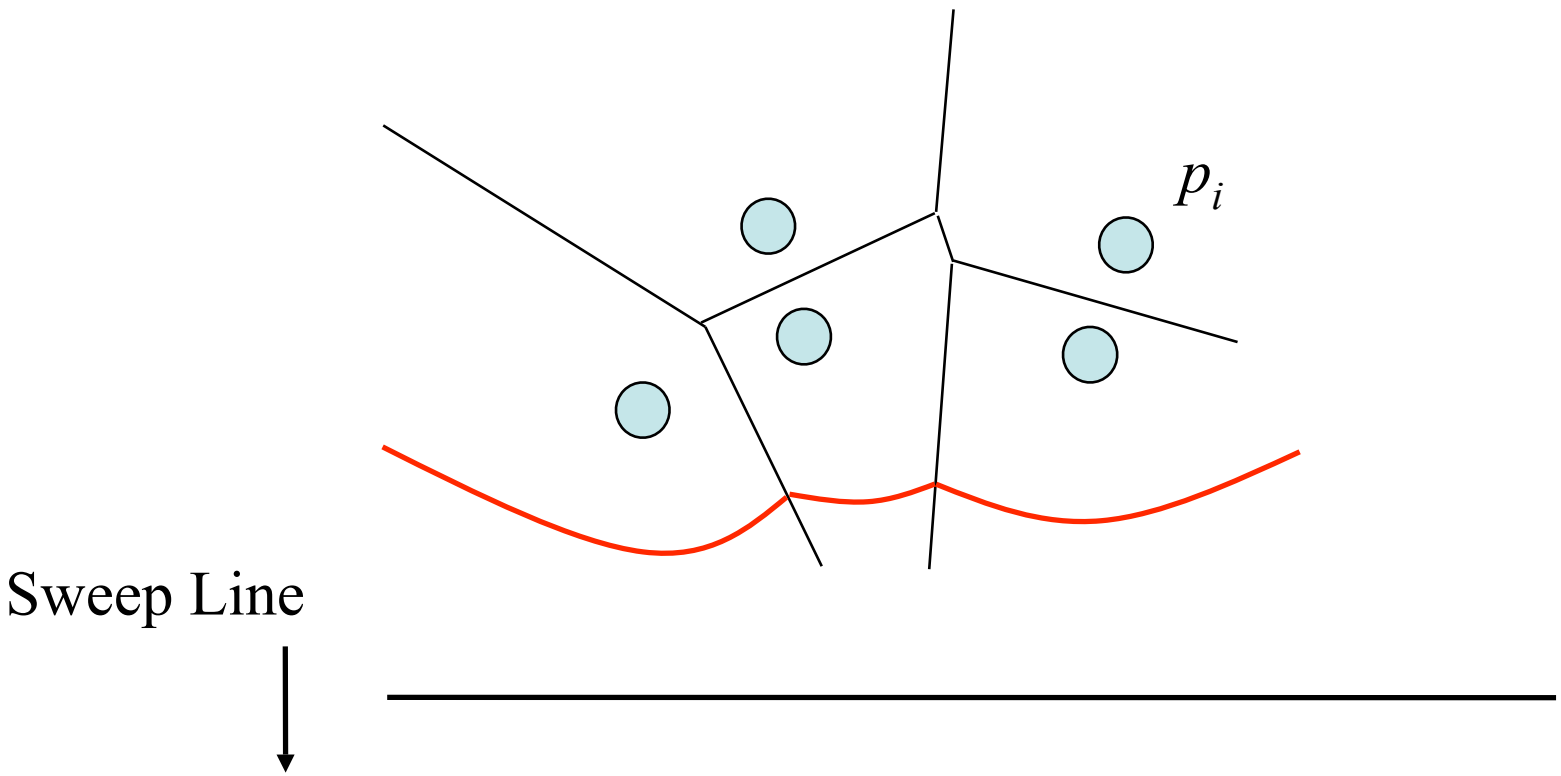
Constructing Voronoi Diagrams

Break points trace out Voronoi edges.



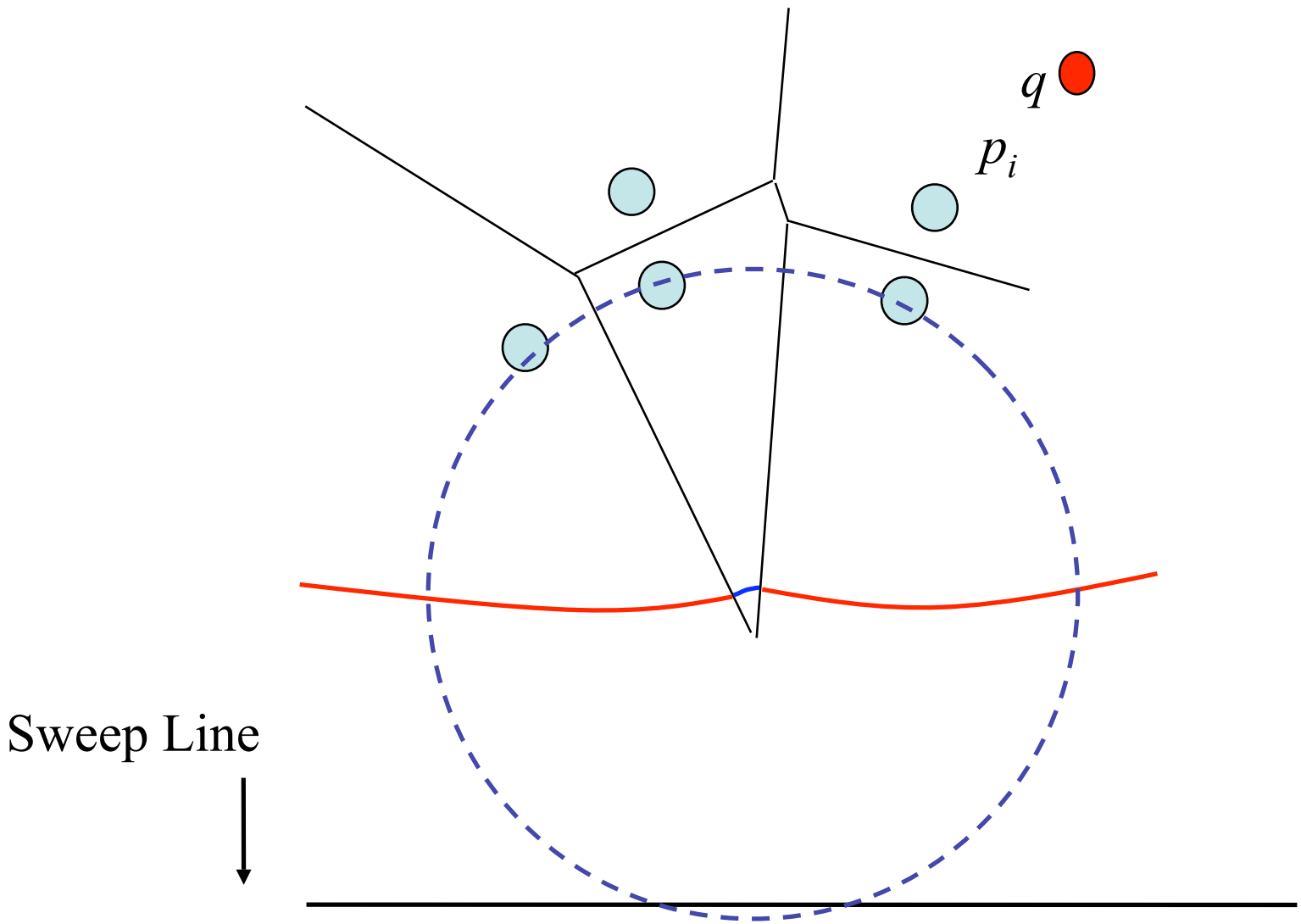
Constructing Voronoi Diagrams

Arcs flatten out as sweep line moves down.



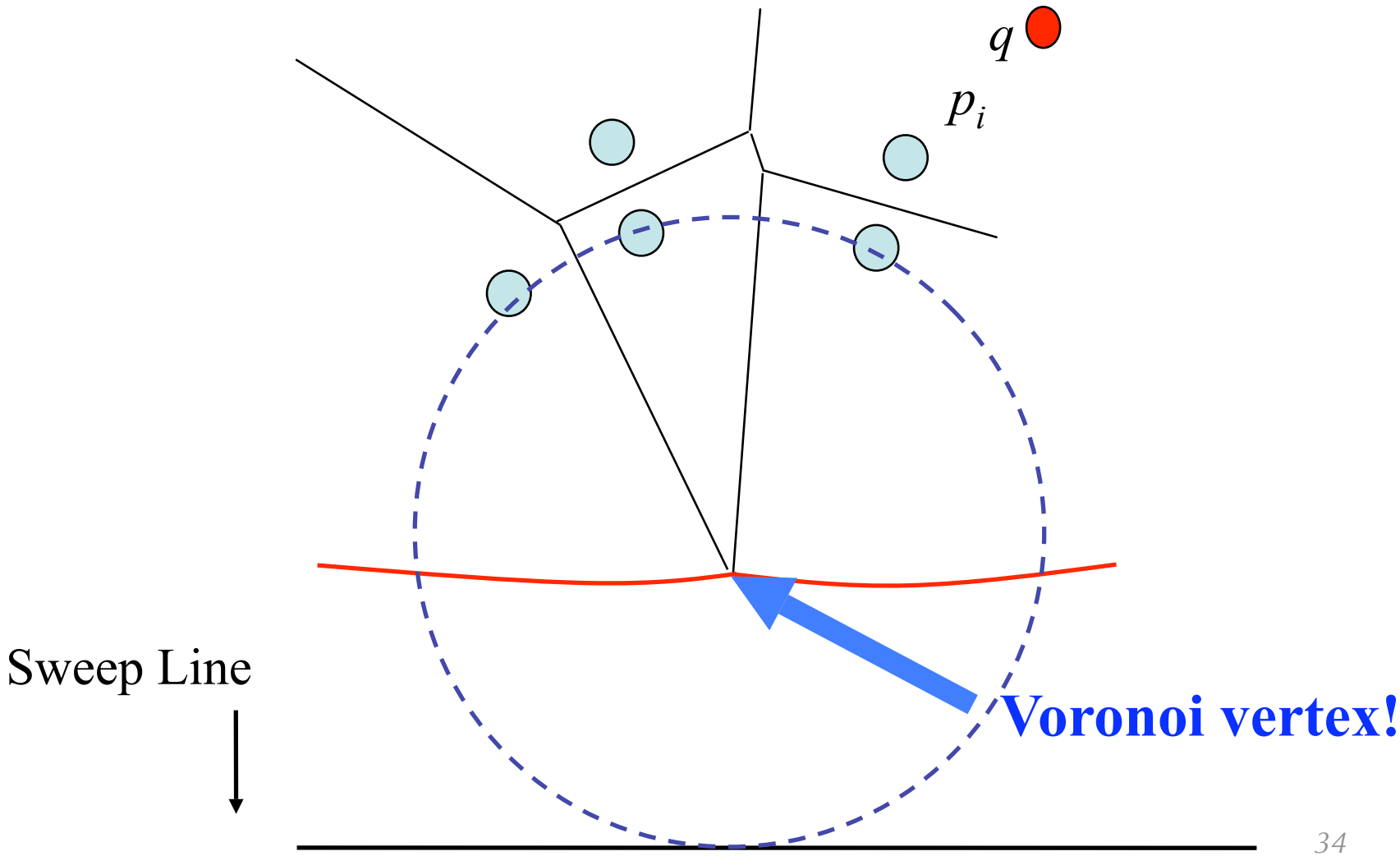
Constructing Voronoi Diagrams

Eventually, the middle arc disappears.



Constructing Voronoi Diagrams

We have detected a circle that is empty (contains no sites)



Beach Line properties

- **voronoi edge = break point trajectory**
  - 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 = collision of break points = disappeared parabolic curve**
  - Decimation of an old arc identifies new vertex

Break

- 10 Minutes

Demo

- **A visual implementation of Fortune's Voronoi algorithm**
  - *by Allan Odgaard & Benny Kjær Nielsen*
  - Source code is available
  - http://www.diku.dk/hjemmesider/studerende/duff/Fortune/

"It is notoriously difficult to obtain a practical implementation of an abstractly described geometric algorithm"

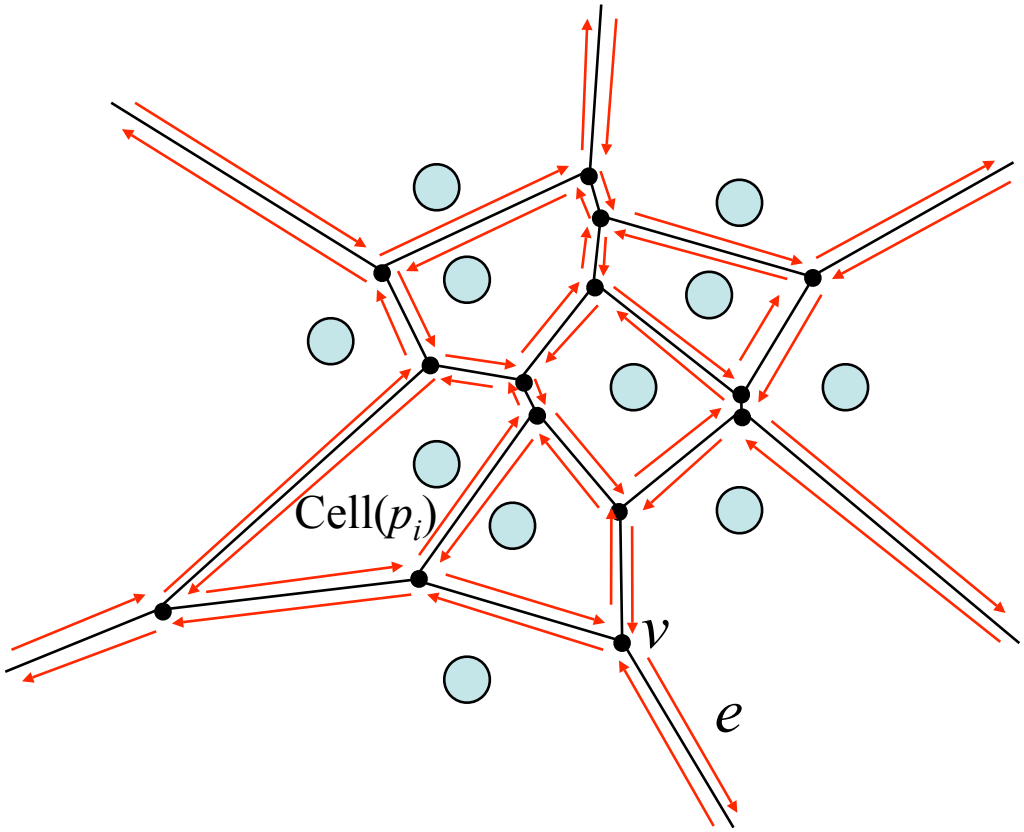
– *Steven Fortune*

Data Structures

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

Doubly Linked List (*D*)

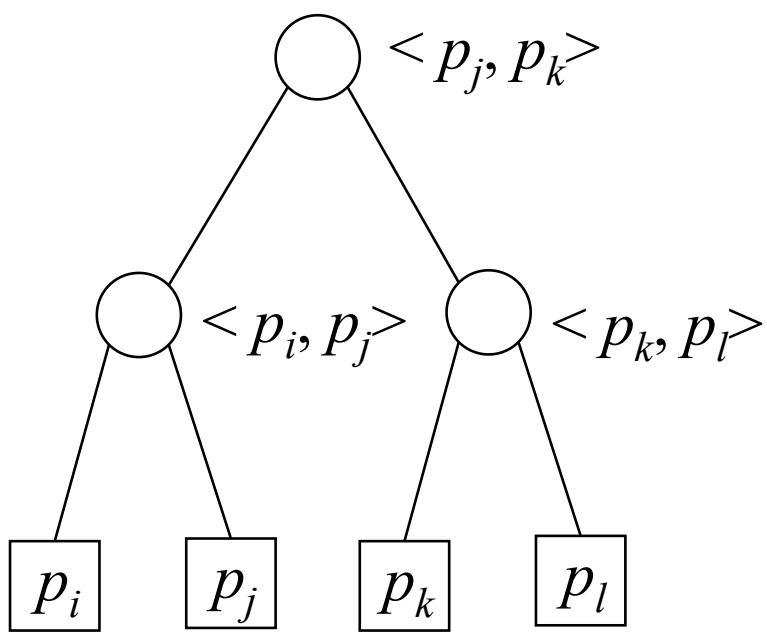
- Divide segments into uni-directional half-edges
- A chain of counter-clockwise half-edges forms a cell
- Define a half-edge's "twin" to be its opposite half-edge of the same segment

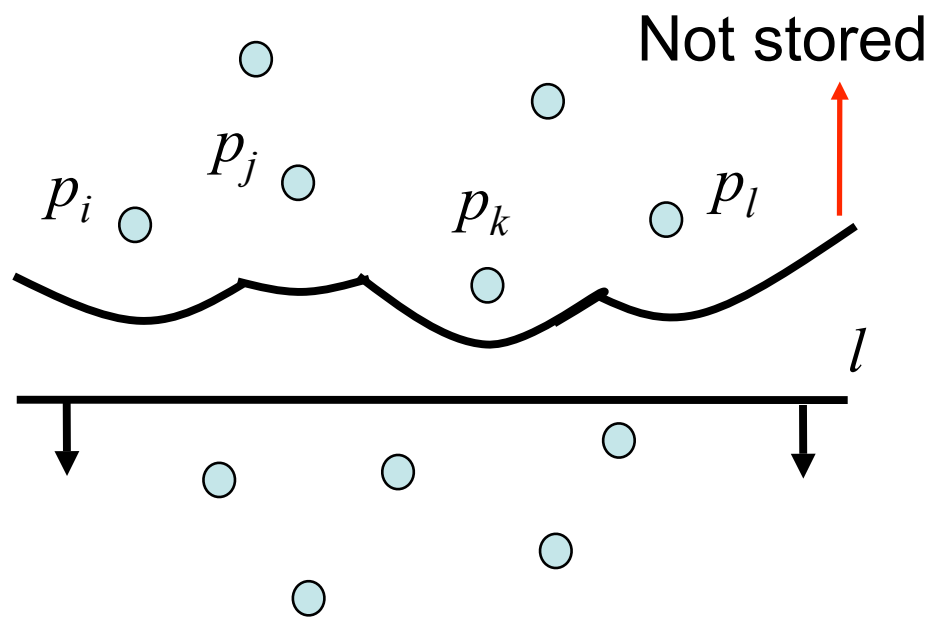


Balanced Binary Tree (*T*)

- **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



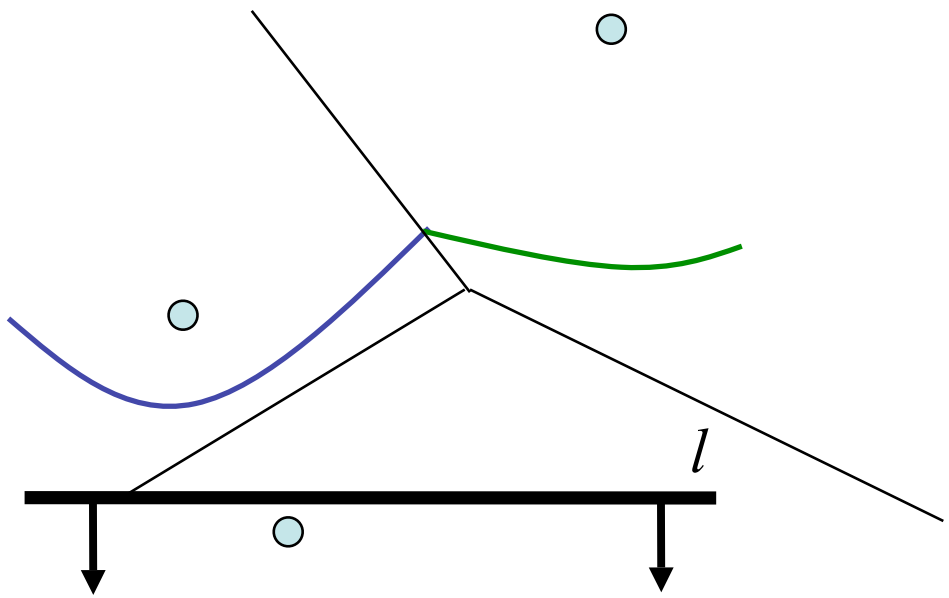


Event Queue (*Q*)

- 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
  - Site Events (when the sweep line encounters a new site point)
  - 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

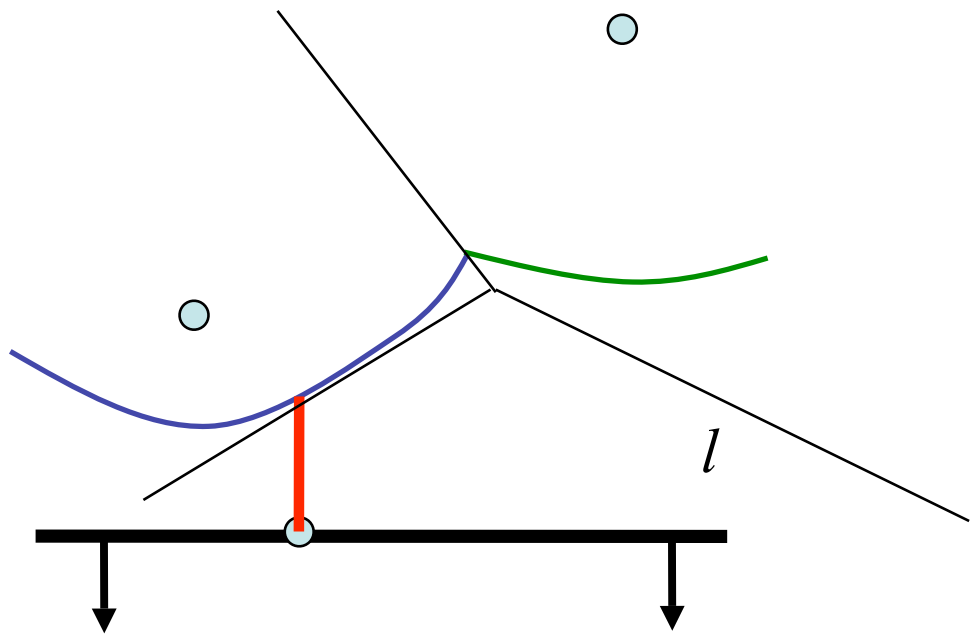
Site Event

A new arc appears when a new site appears.



Site Event

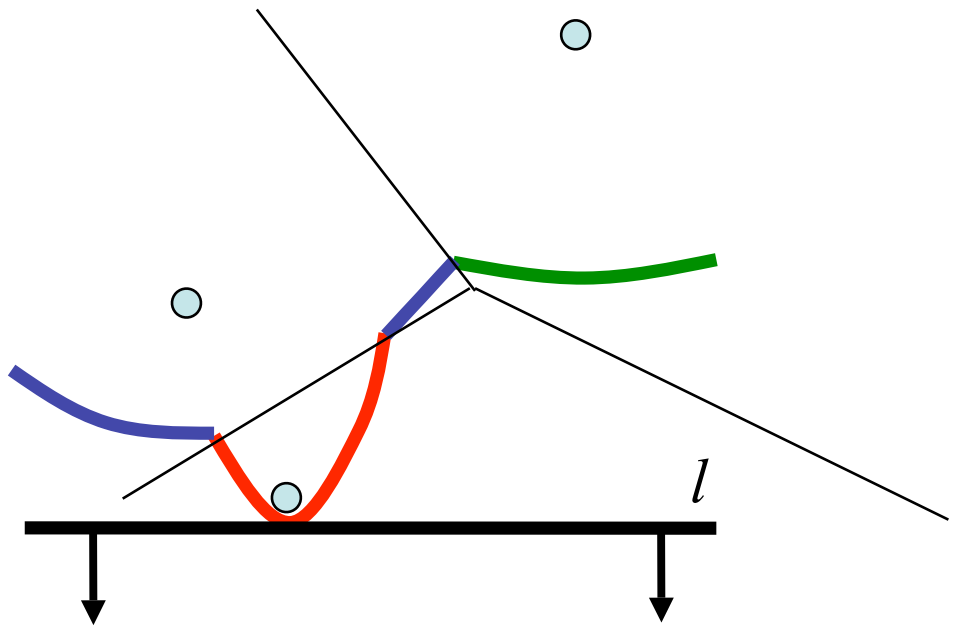
A new arc appears when a new site appears.



Site Event

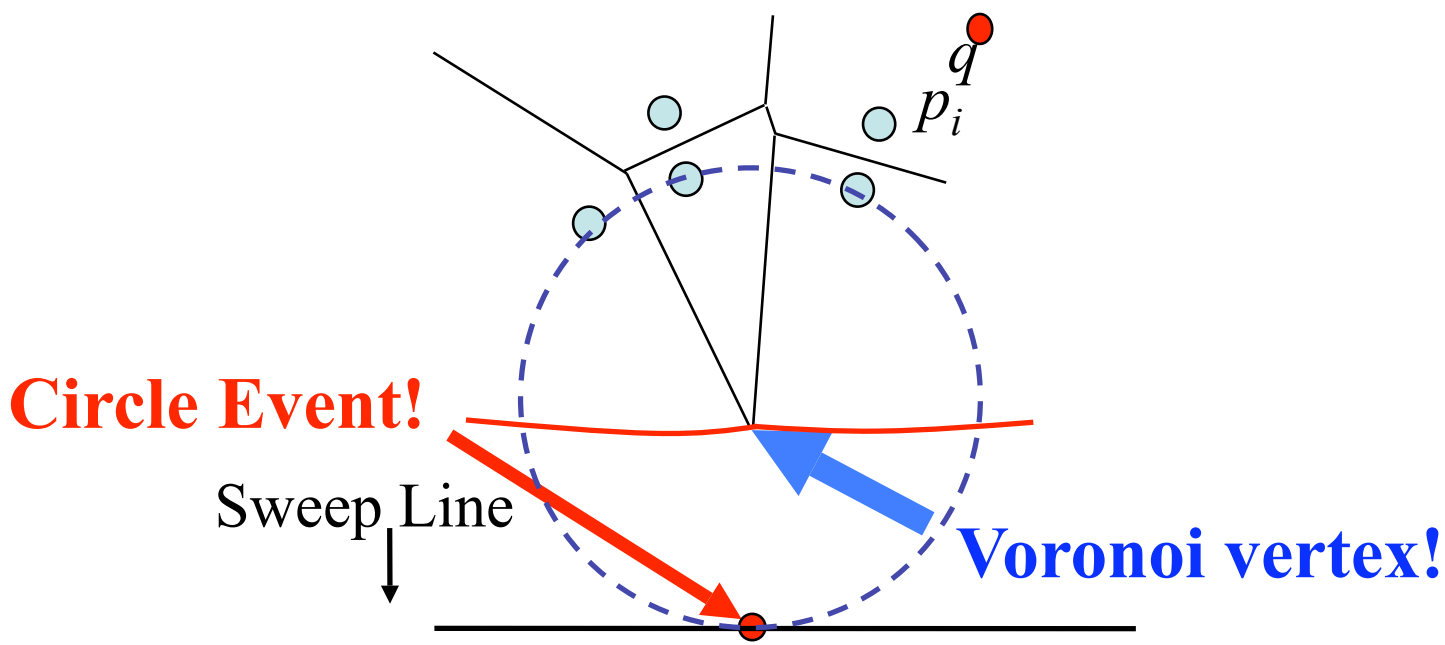
Original arc above the new site is broken into two

→ 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

Algorithm

1. Initialize
   - Event queue Q $\Leftarrow$ all site events
   - Binary search tree T $\Leftarrow \varnothing$
   - Doubly linked list D $\Leftarrow \varnothing$

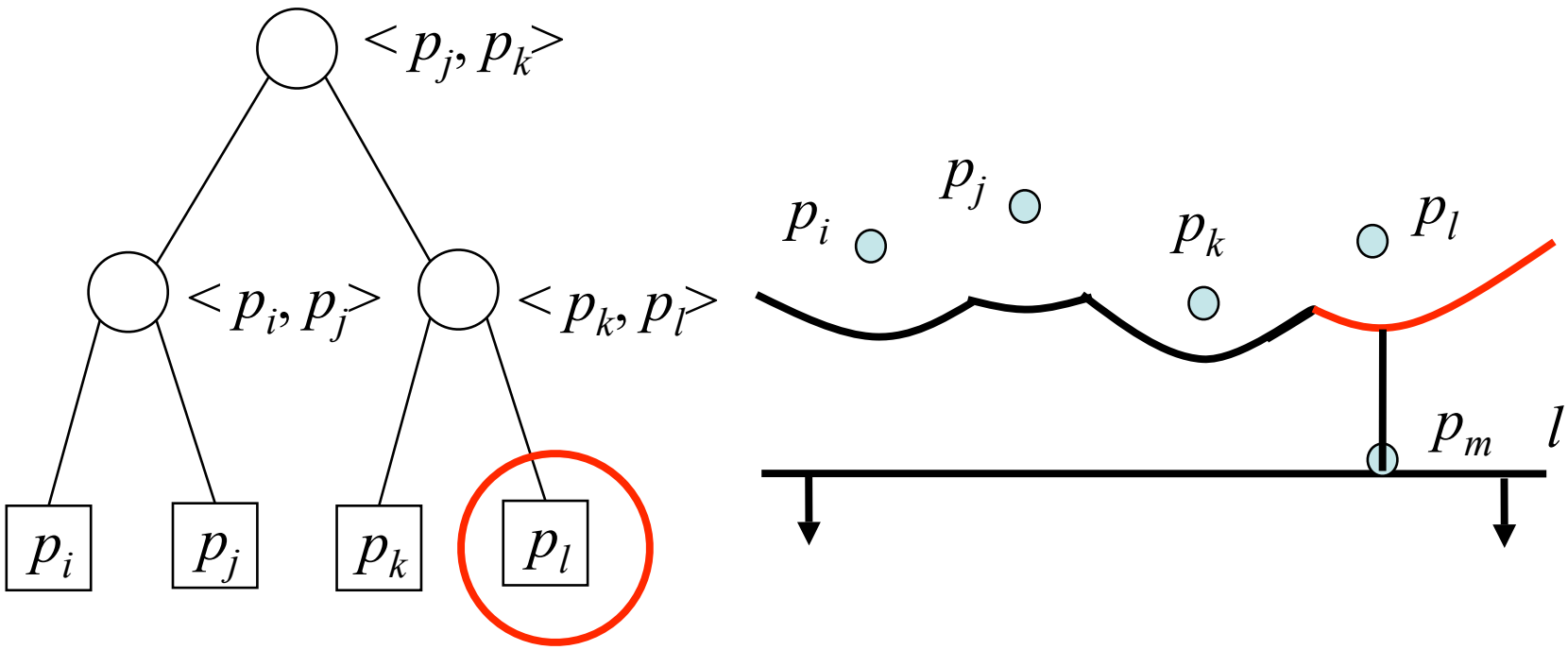
2. While Q not $\varnothing$,
   - Remove event (e) from Q with largest y-coordinate
     - HandleEvent(e, T, D)

Handling Site Events

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

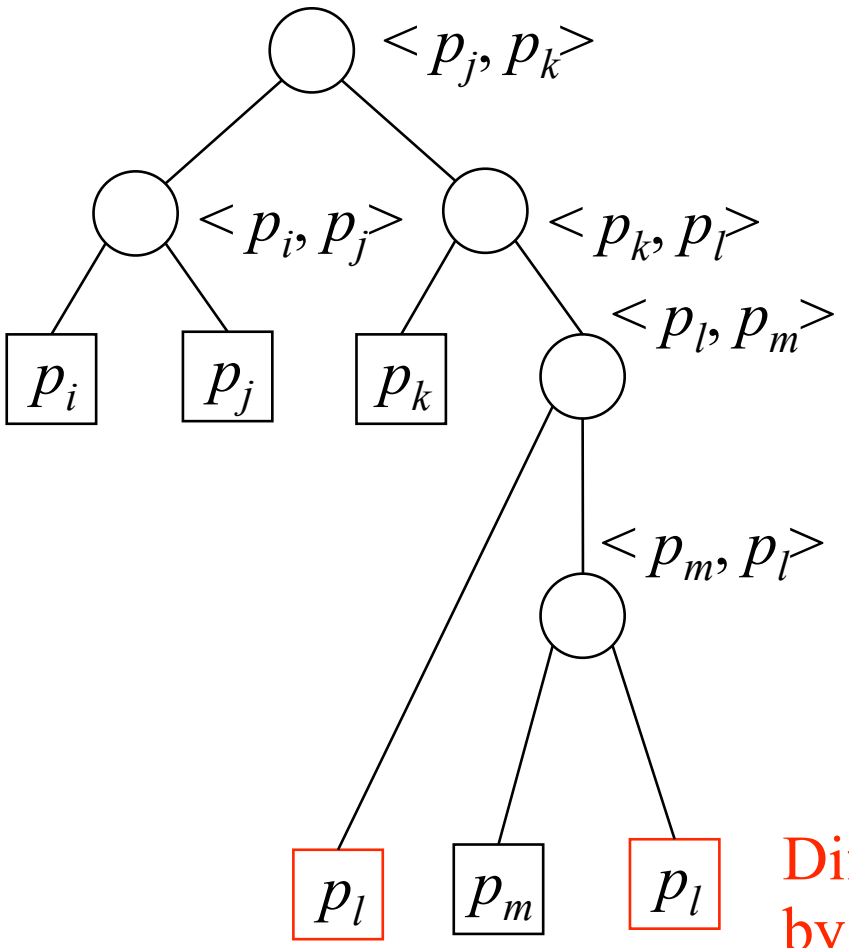
Locate the existing arc that is above the new site

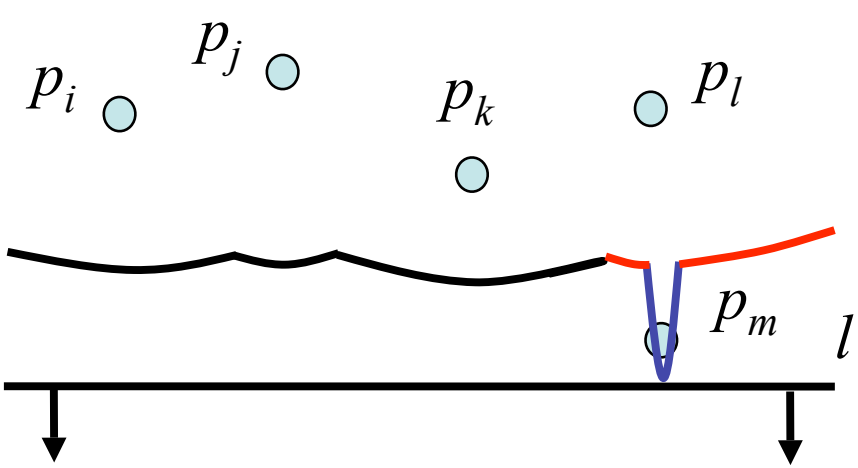
- The x coordinate of the new site is used for the binary search
- The x coordinate of each breakpoint along the root to leaf path is computed on the fly



Break the Arc

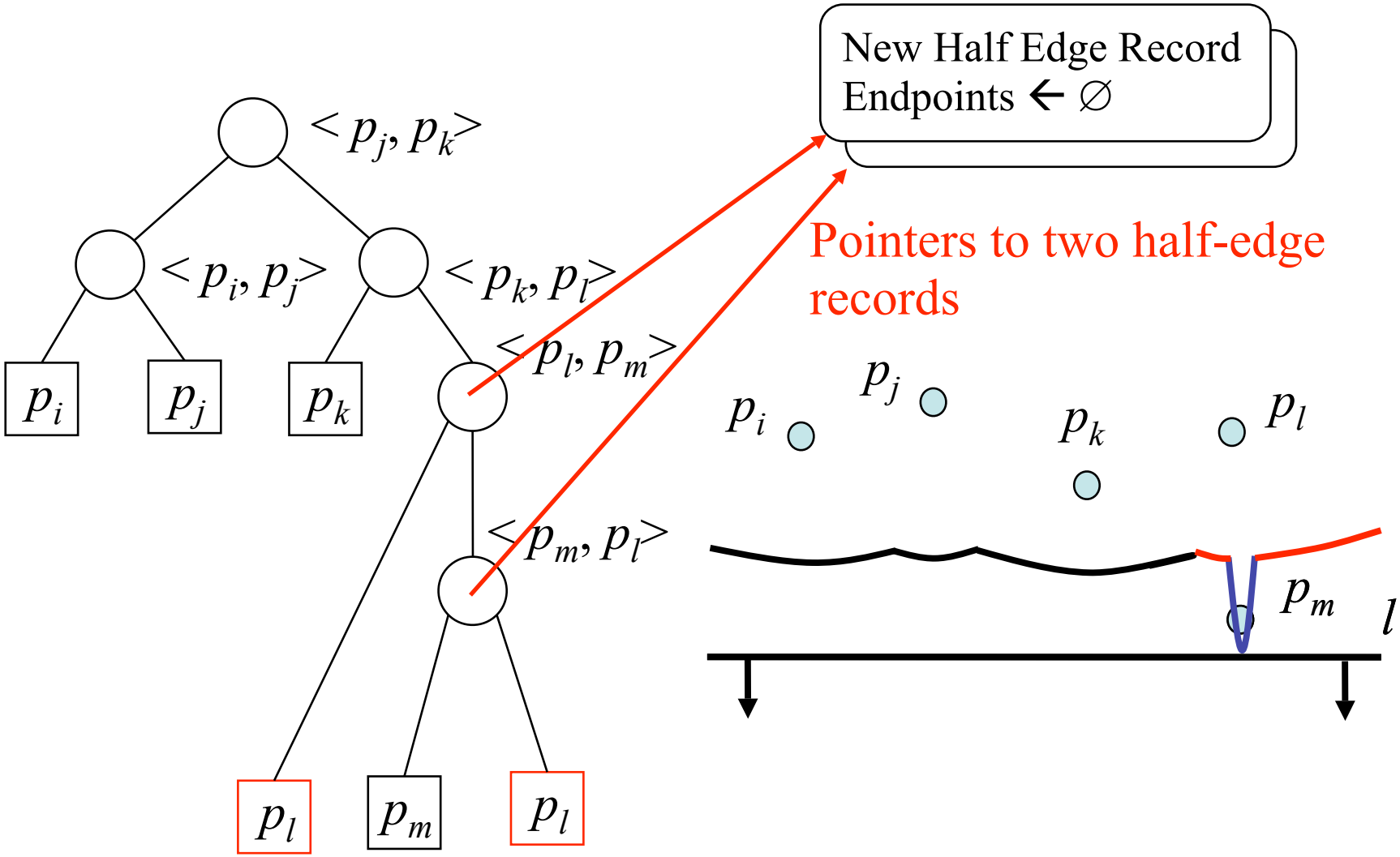
Corresponding leaf replaced by a new sub-tree





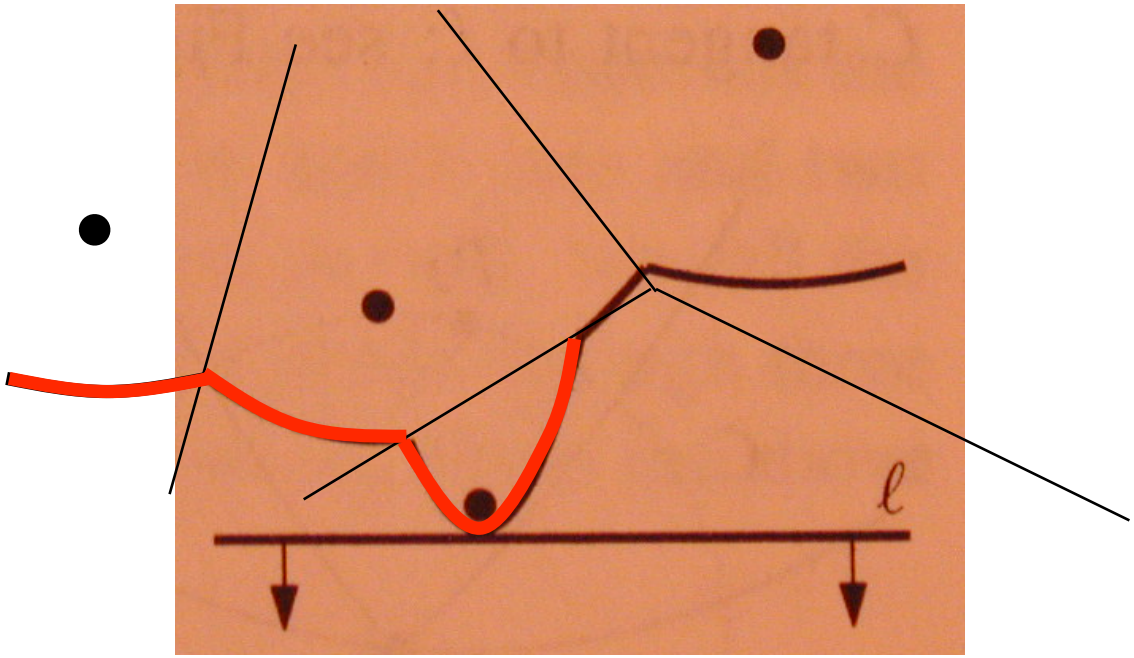
Different arcs can be identified by the same site!

Add a new edge record in the doubly linked list



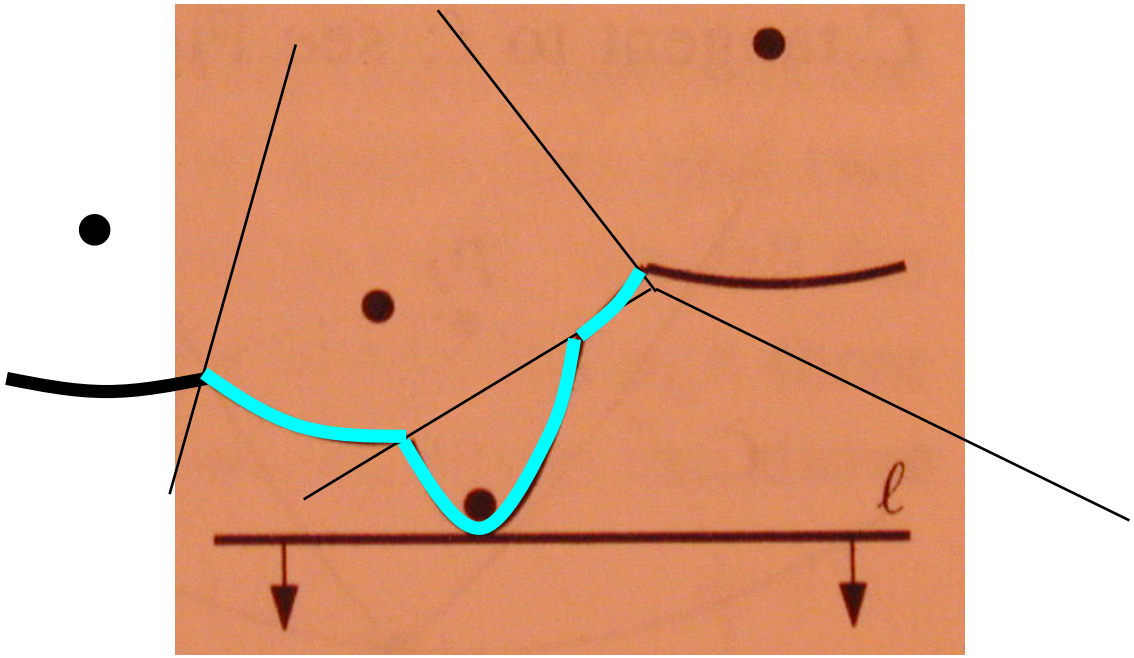
Checking for Potential Circle Events

- Scan for 3 consecutive arcs and determine if breakpoints converge
  - Triples with new arc in the middle do not have break points that converge



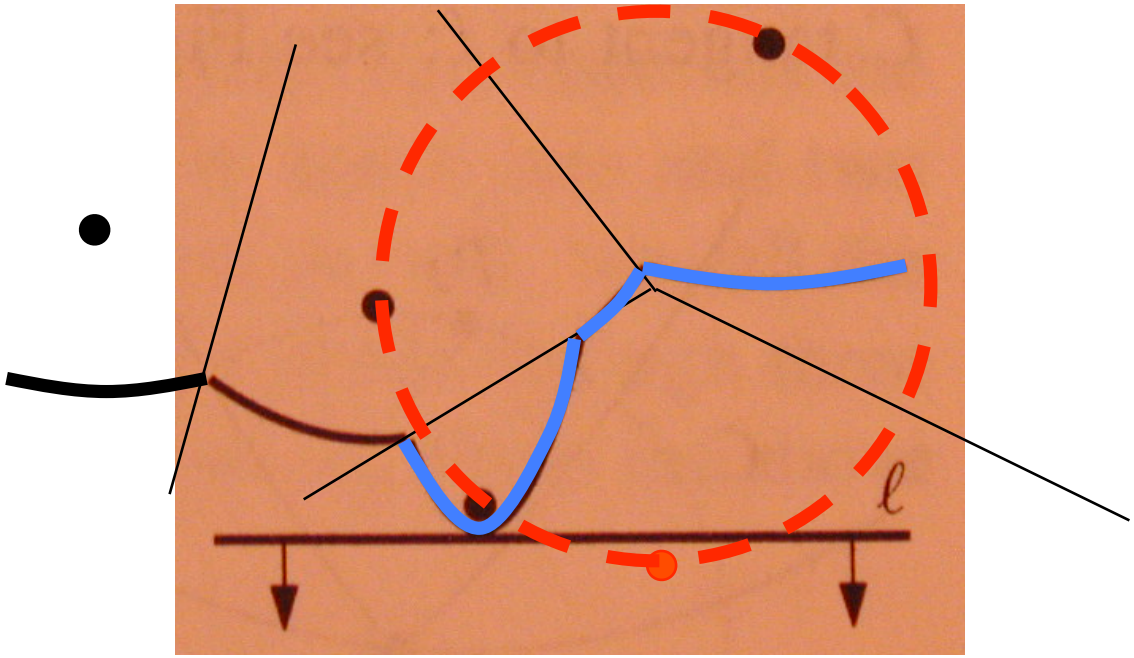
Checking for Potential Circle Events

- Scan for 3 consecutive arcs and determine if breakpoints converge
  - Triples with new arc in the middle do not have break points that converge



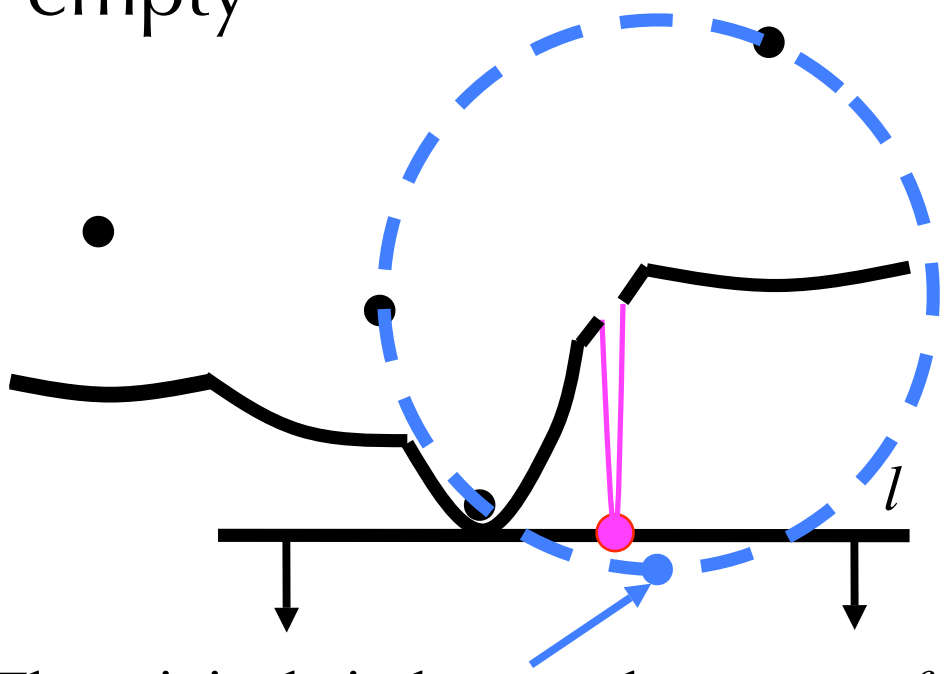
Checking for Potential Circle Events

- Scan for 3 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. **Update T**:
   - Locate the leaf representing the existing arc that is above the new site
     - Delete the potential circle event in the event queue
   - Break the arc by replacing the leaf node with a sub tree representing the new arc and break points

2. **Update D**:
   - Add a new edge record in the doubly linked list

3. **Update Q**:
   - 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

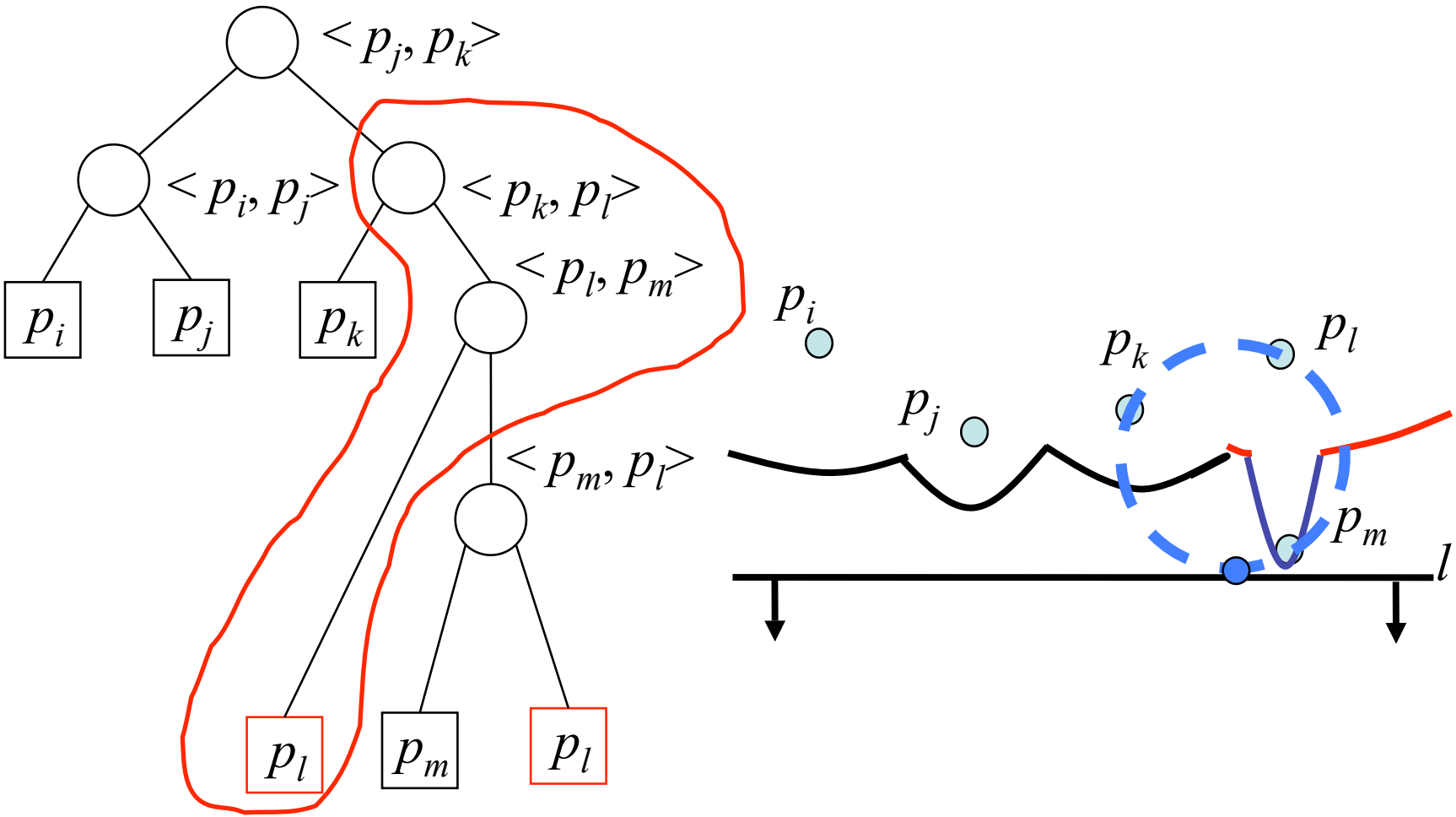
Handling Circle Events

1. **Update T**:
   - Delete from T the leaf node of the disappearing arc and its associated circle events in the event queue

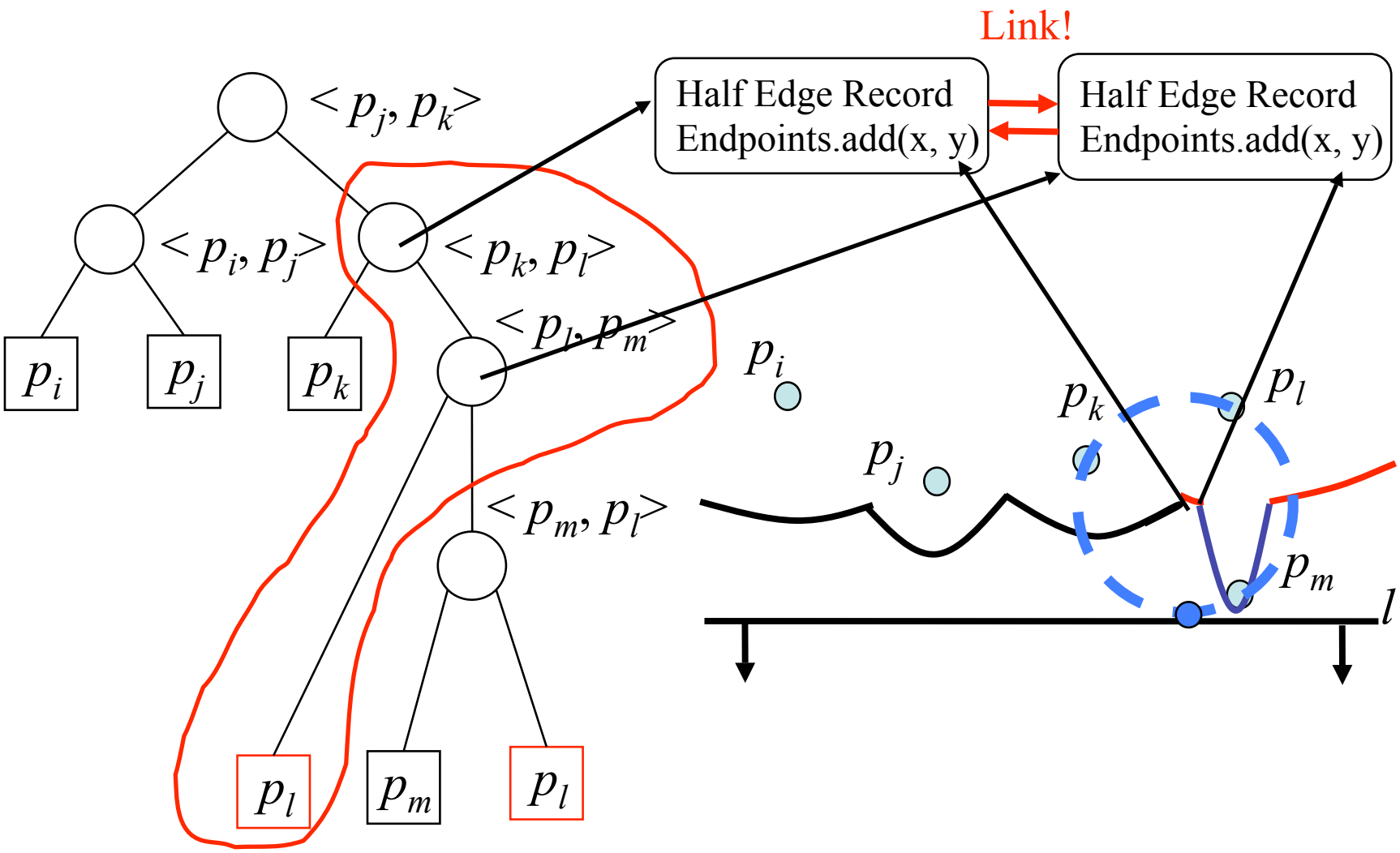
2. **Update D**:
   - Add vertex to corresponding edge record in doubly linked list
   - Create new edge record in doubly linked list

3. **Update Q**:
   - Check the new triplets formed by the former neighboring arcs for potential circle events

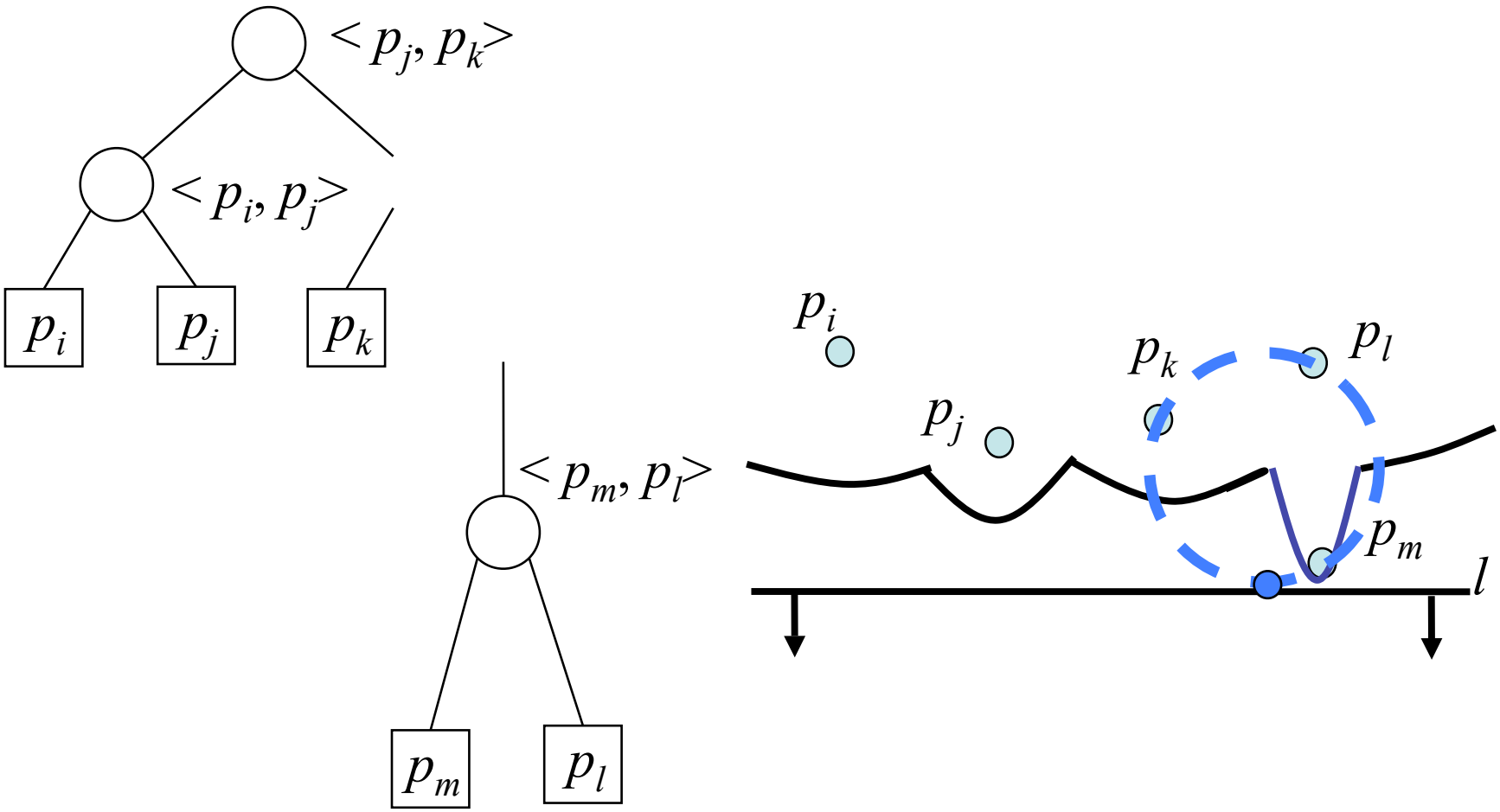
A Circle Event



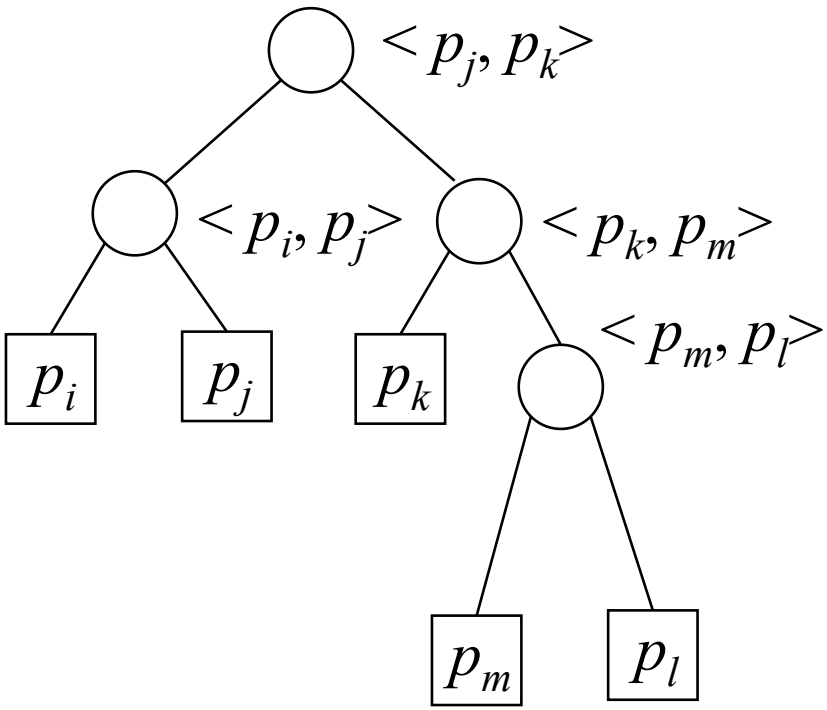
Add vertex to corresponding edge record

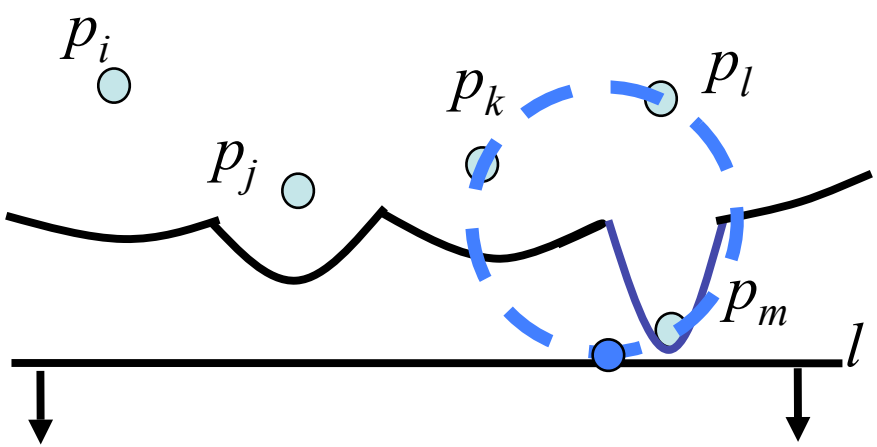


Deleting disappearing arc

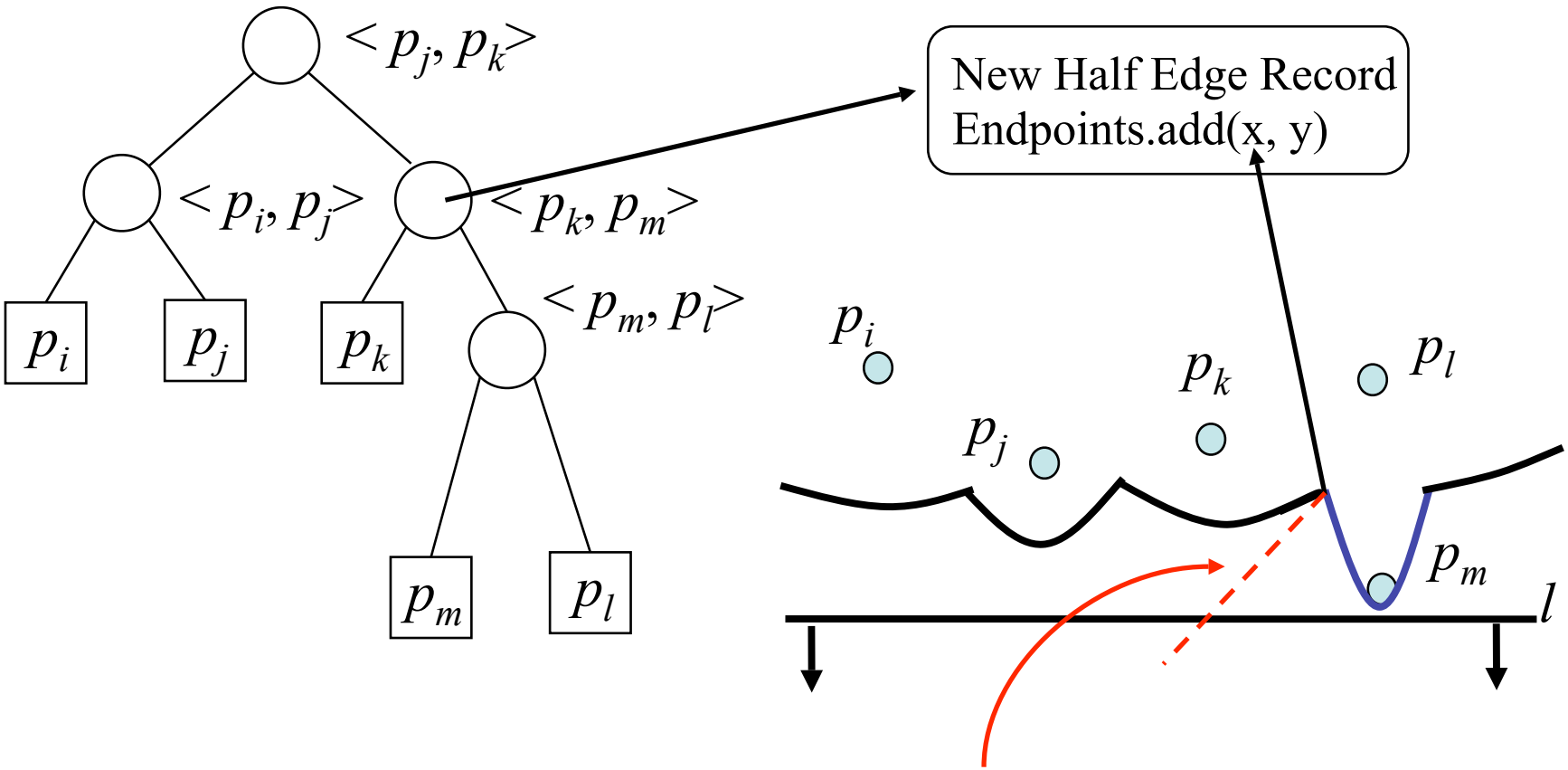


Deleting disappearing arc



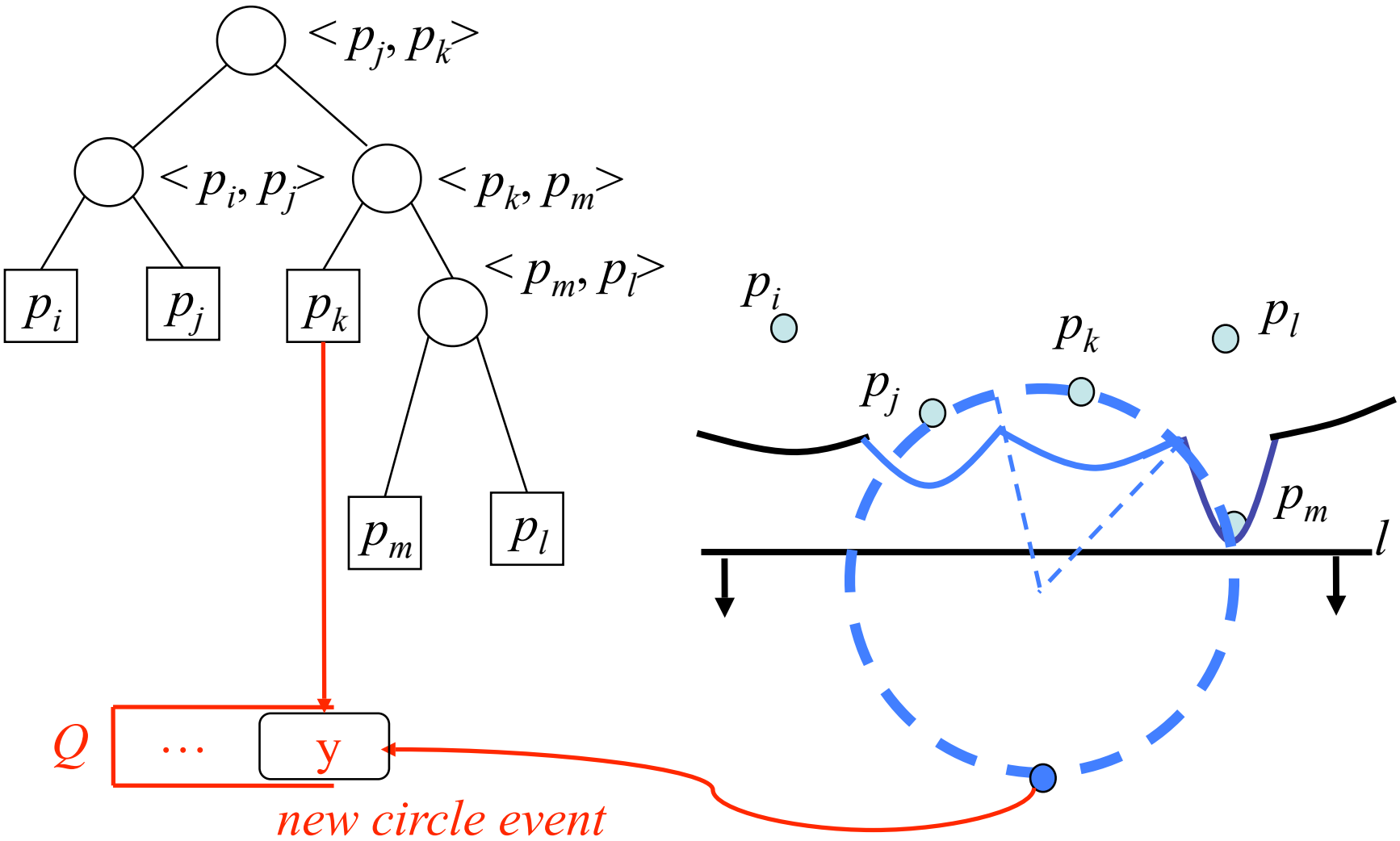


Create new edge record



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

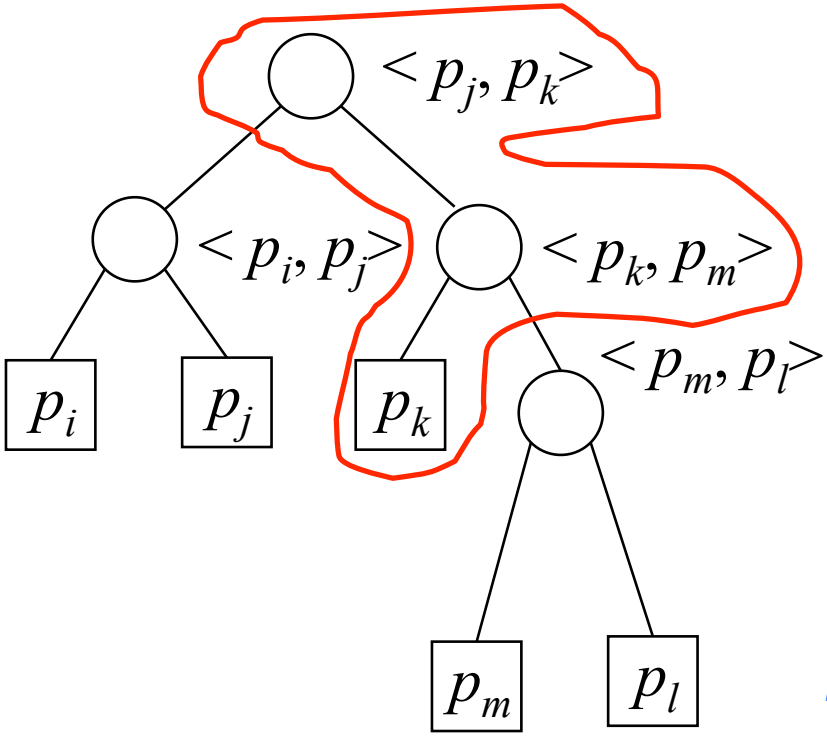
Check the new triplets for potential circle events

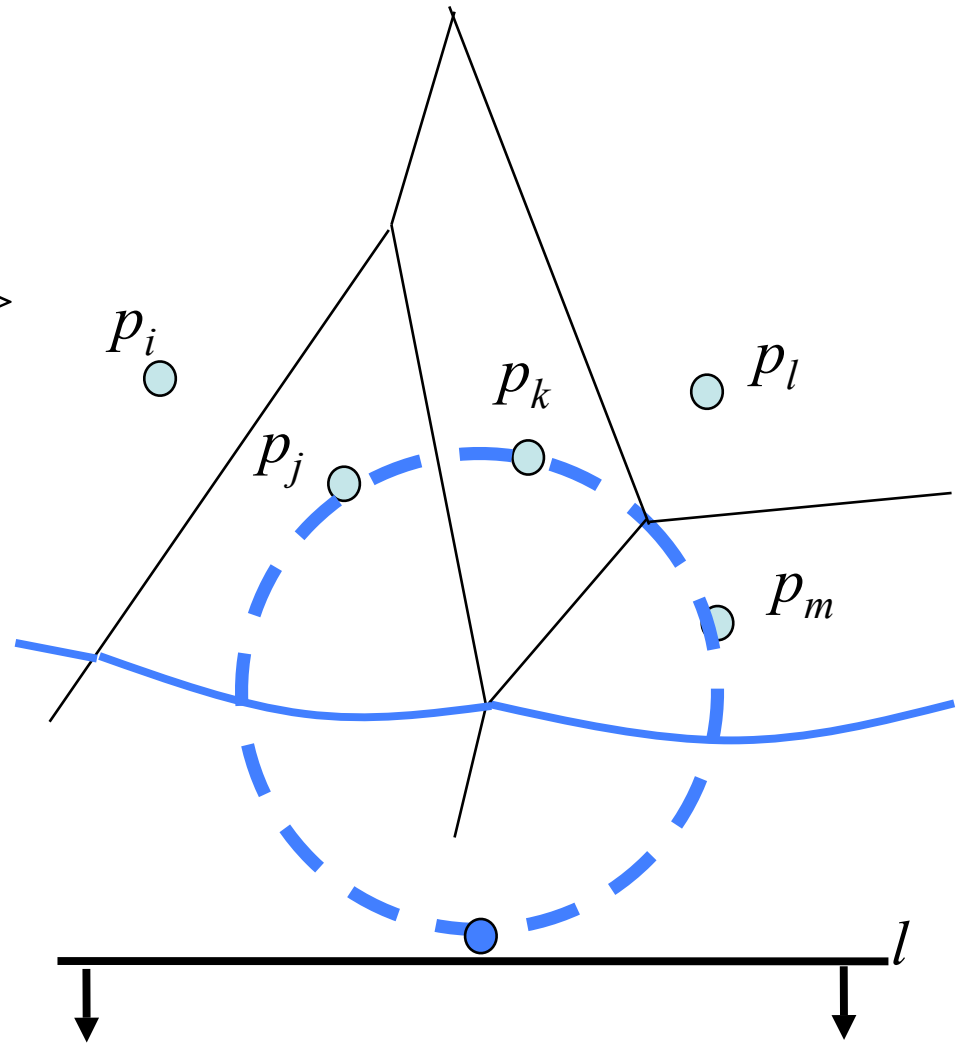


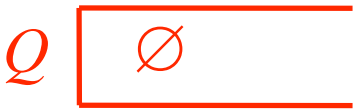
Minor Detail

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

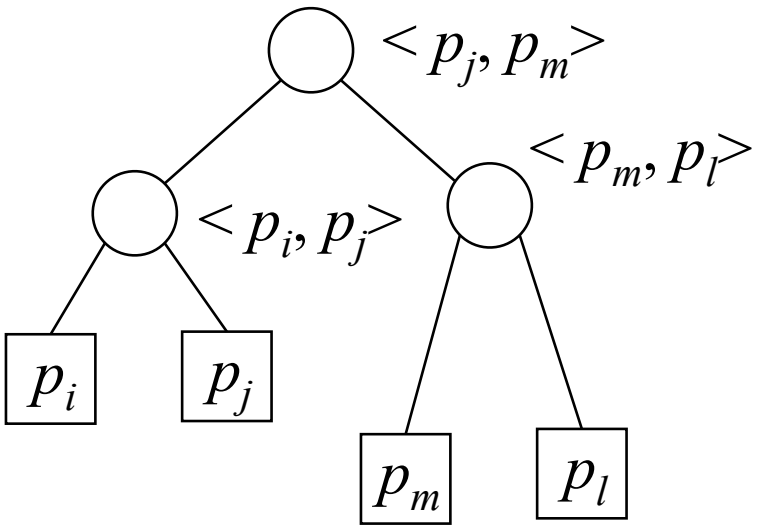
Algorithm Termination

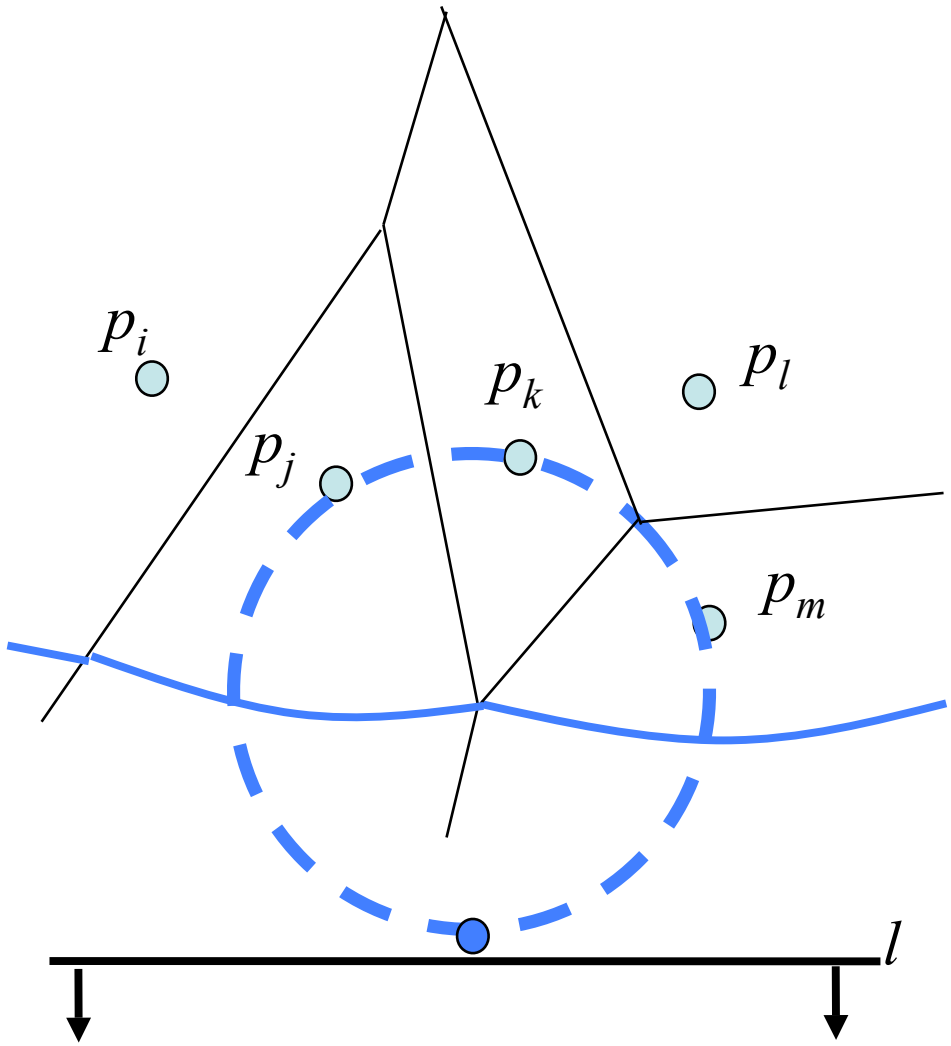


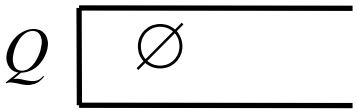




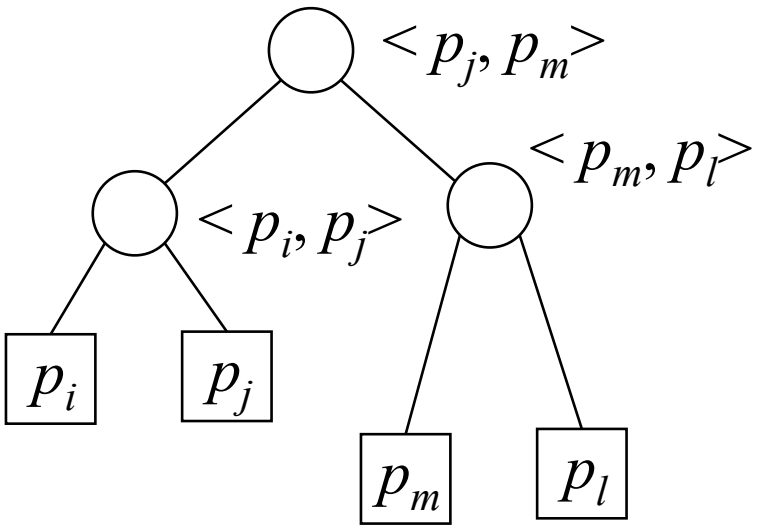
Algorithm Termination

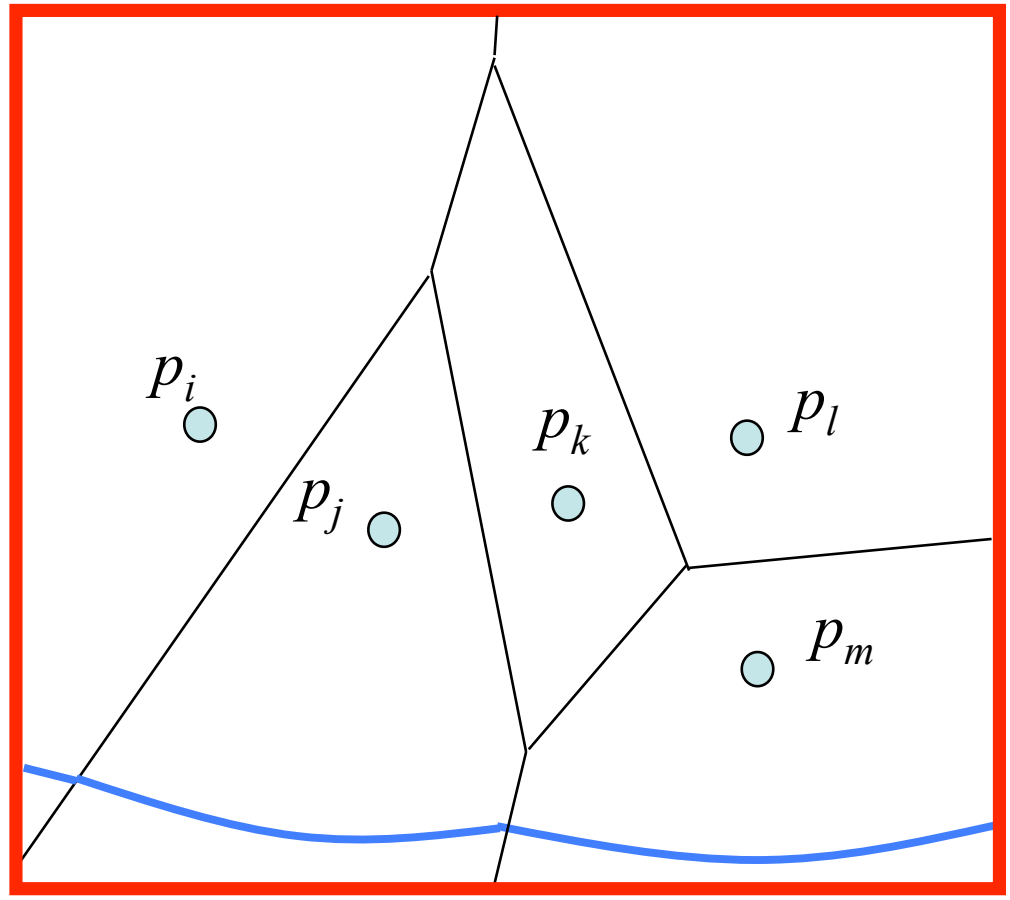




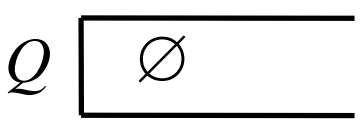


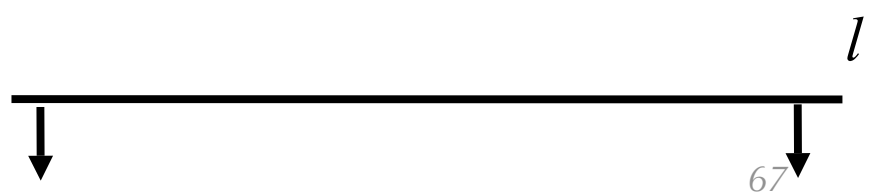
Algorithm Termination





Terminate half-lines with a bounding box!





Outline

- Definitions and Examples
- Properties of Voronoi diagrams
- Complexity of Voronoi diagrams
- Constructing Voronoi diagrams
  - Intuitions
  - Data Structures
  - Algorithm
- **Running Time Analysis**
- Demo
- Duality and degenerate cases

Handling Site Events

	Running Time
1. Locate the leaf representing the existing arc that is above the new site – Delete the potential circle event in the event queue	$O(\log n)$
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	$O(1)$

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	$O(1)$

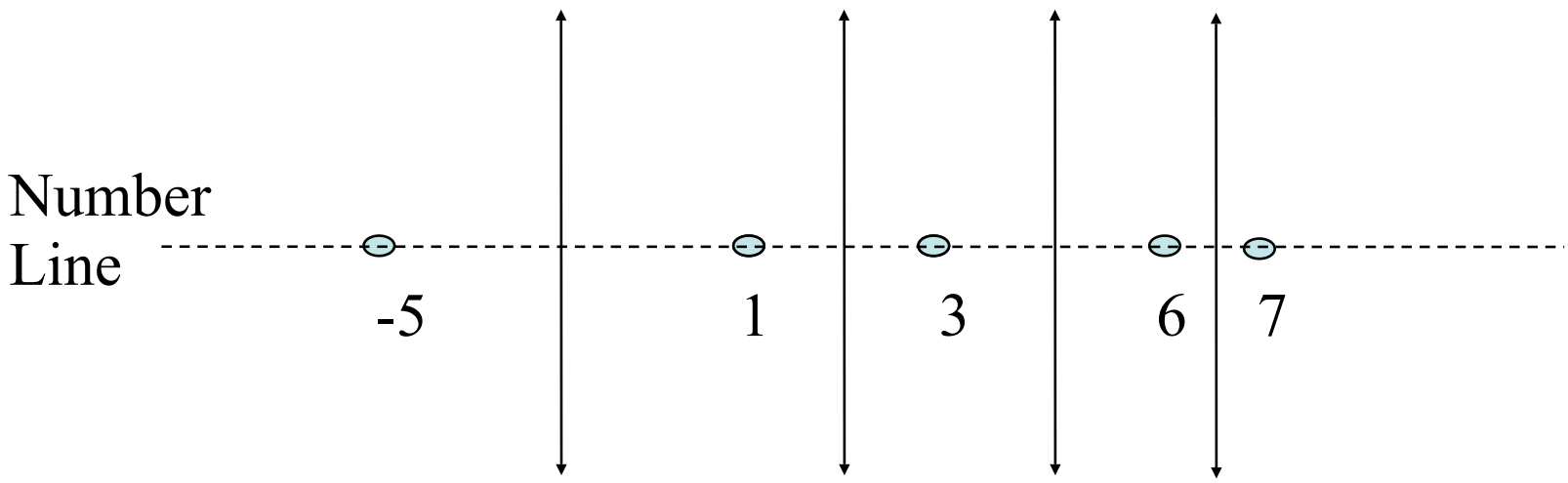
Total Running Time

- 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!

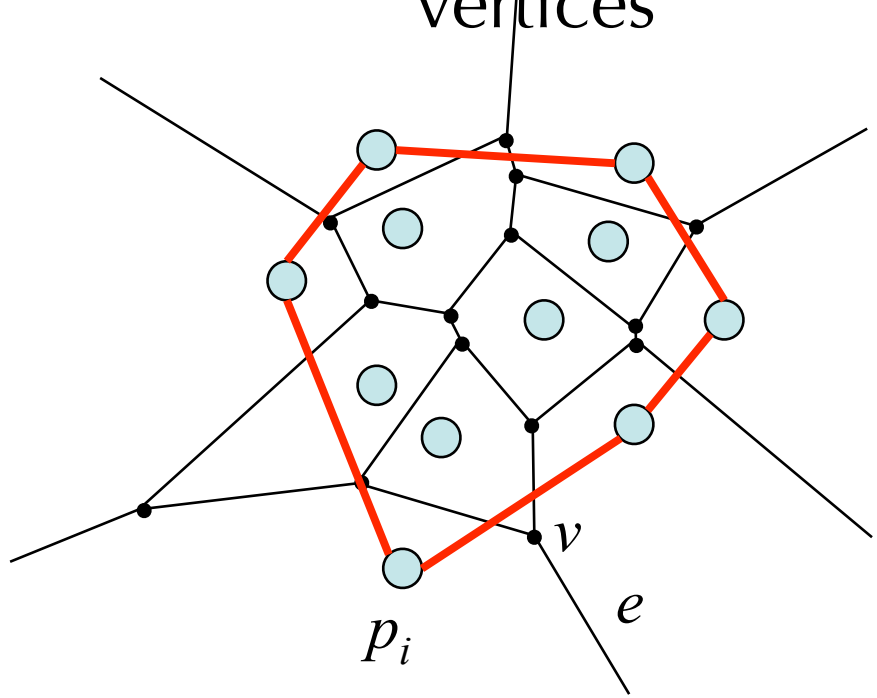


Outline

- Definitions and Examples
- Properties of Voronoi diagrams
- Complexity of Voronoi diagrams
- Constructing Voronoi diagrams
  - Intuitions
  - Data Structures
  - Algorithm
- Running Time Analysis
- **Duality and degenerate cases**

Voronoi Diagram/Convex Hull Duality

Sites sharing a half-infinite edge are convex hull vertices



Degenerate Cases

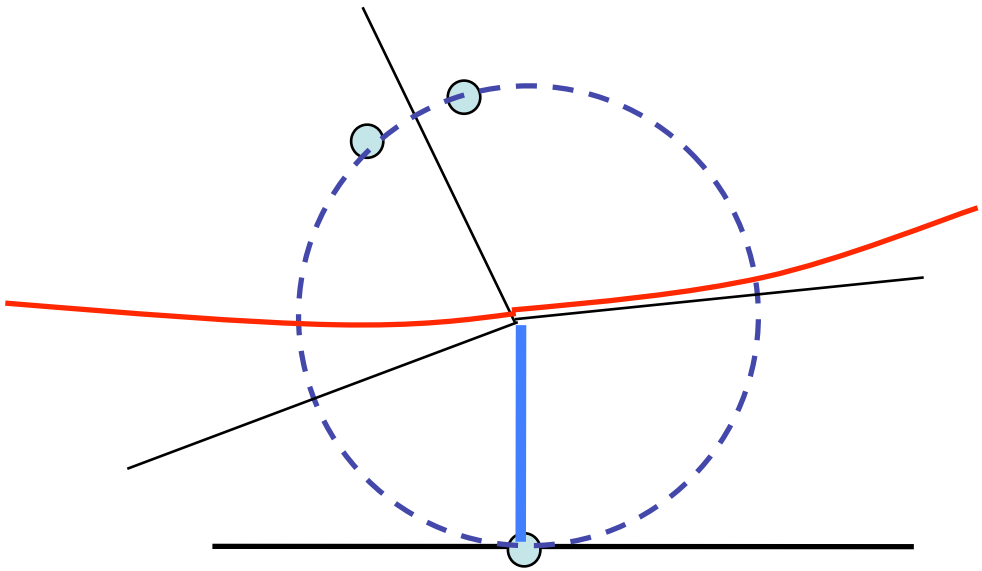
- Events in Q share the same y-coordinate
  - Can additionally sort them using x-coordinate
- Circle event involving more than 3 sites
  - Current algorithm produces multiple degree 3 Voronoi vertices joined by zero-length edges
  - Can be fixed in post processing

Degenerate Cases

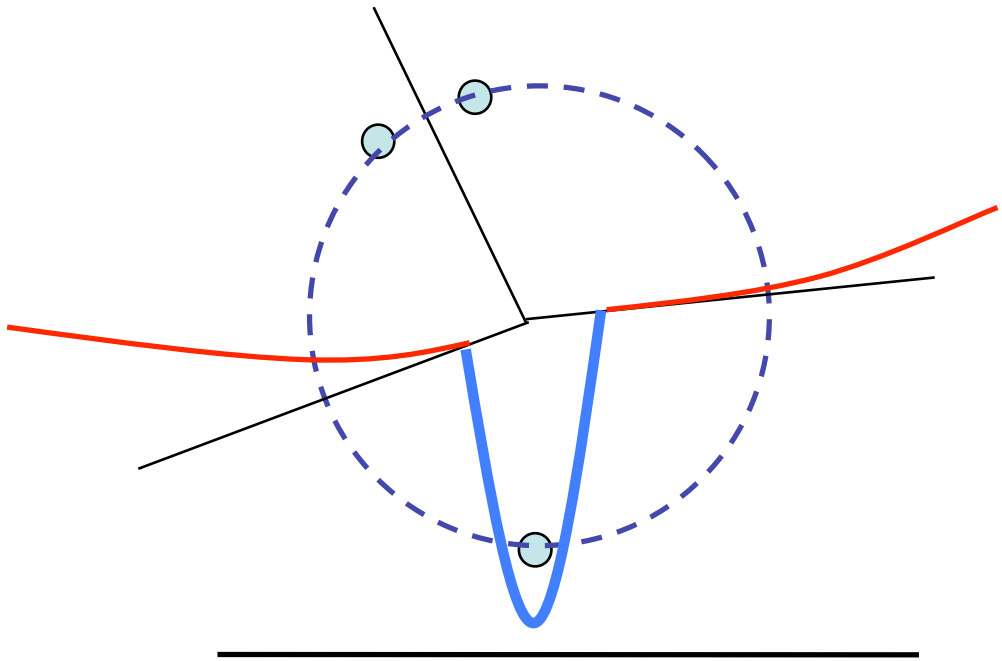
- Site points are collinear (break points neither converge or diverge)
  - Bounding box takes care of this
- One of the sites coincides with the lowest point of the circle event
  - Do nothing

Site coincides with circle event:

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

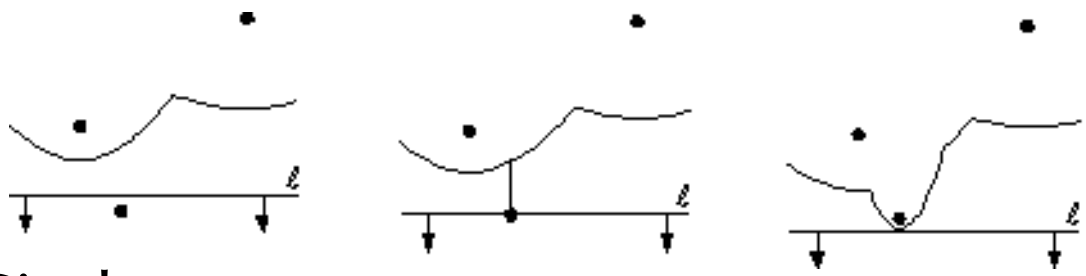


Site coincides with circle event

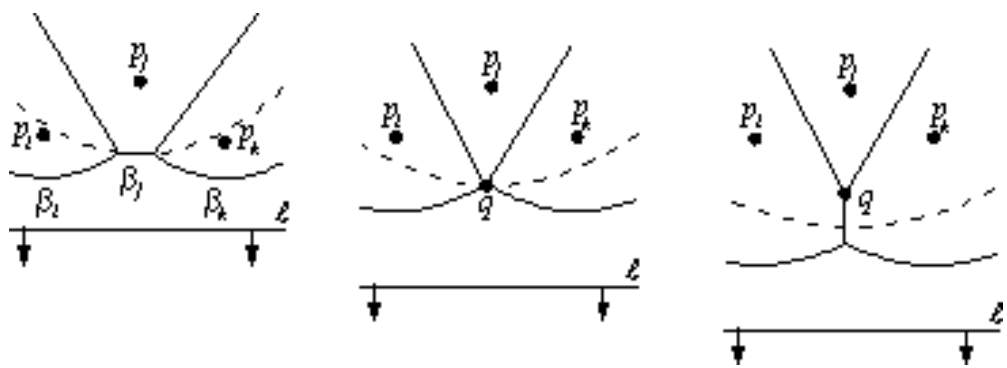


Summary of Fortune's algorithm

- Optimal
- Sweep line algorithm
  - Site events



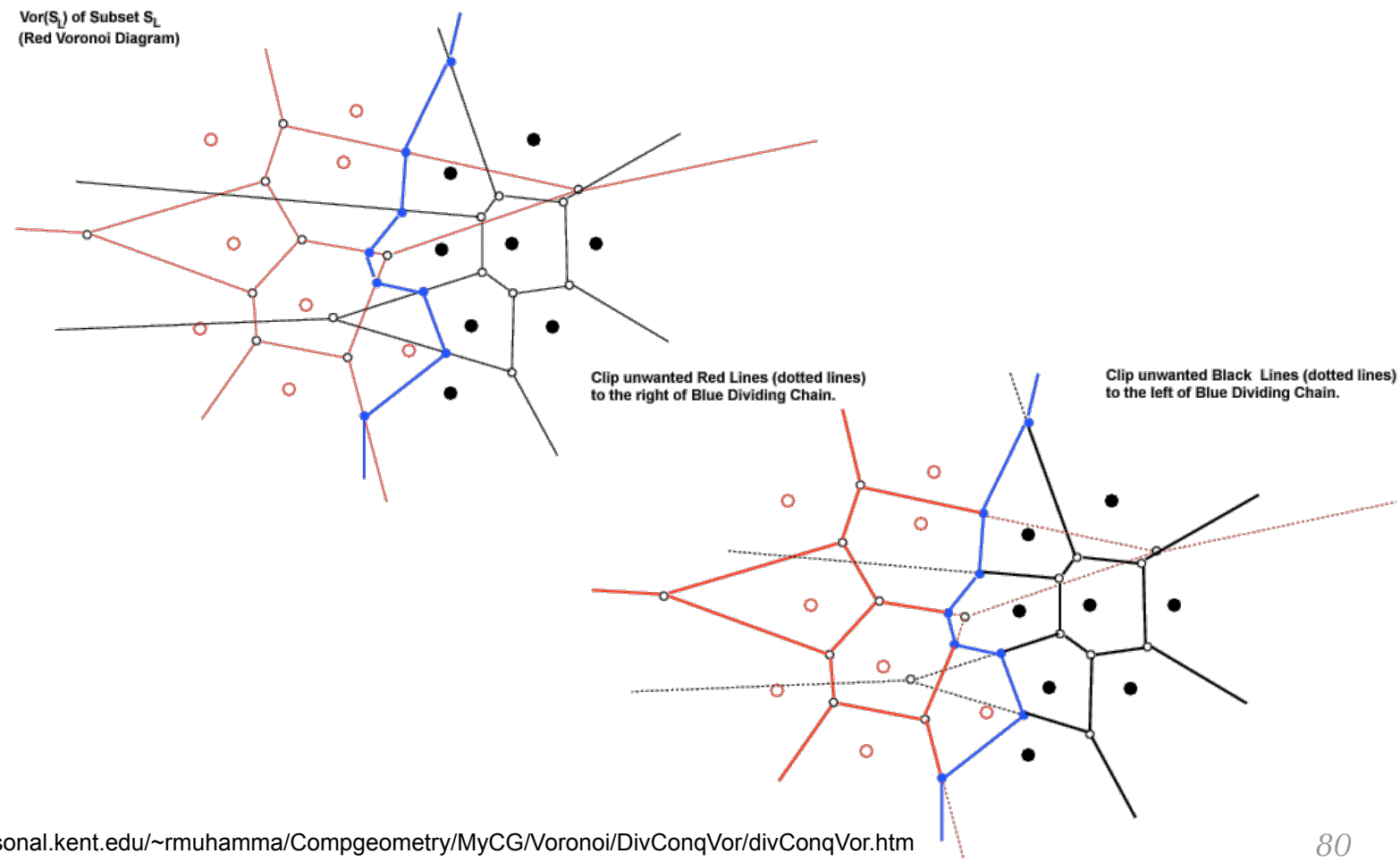
  - Circle events



Images from http://www.cescg.org/CESCG99/RCuk/

Other Ways Computing Voronoi Diagram

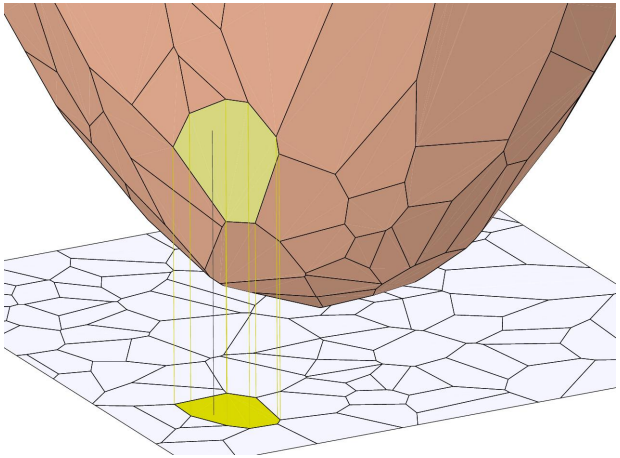
- Divide-and-conquer



http://www.personal.kent.edu/~rmuhamma/Compgeometry/MyCG/Voronoi/DivConqVor/divConqVor.htm

Other Ways Computing Voronoi Diagram

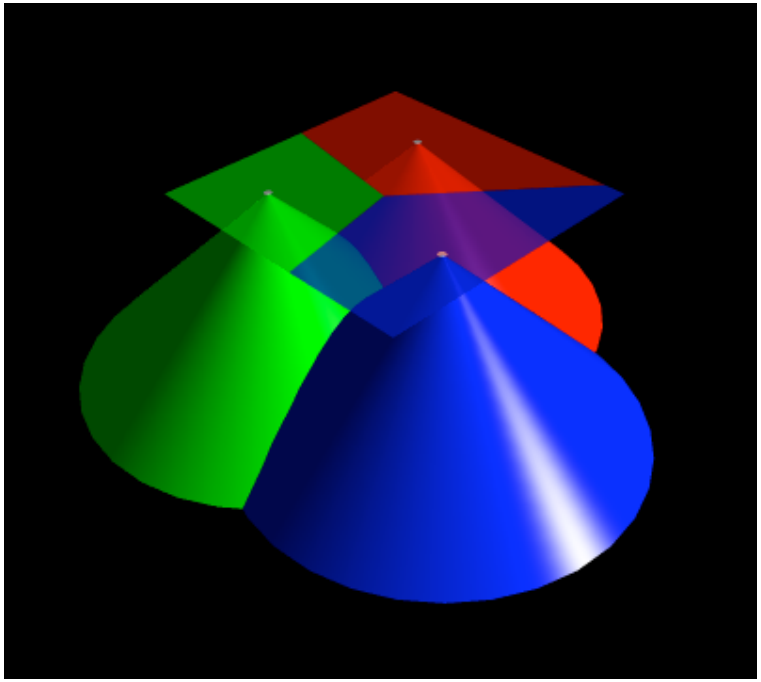
- Lifting: three dimensional convex hull
  - We will learn about this in Chapter 9



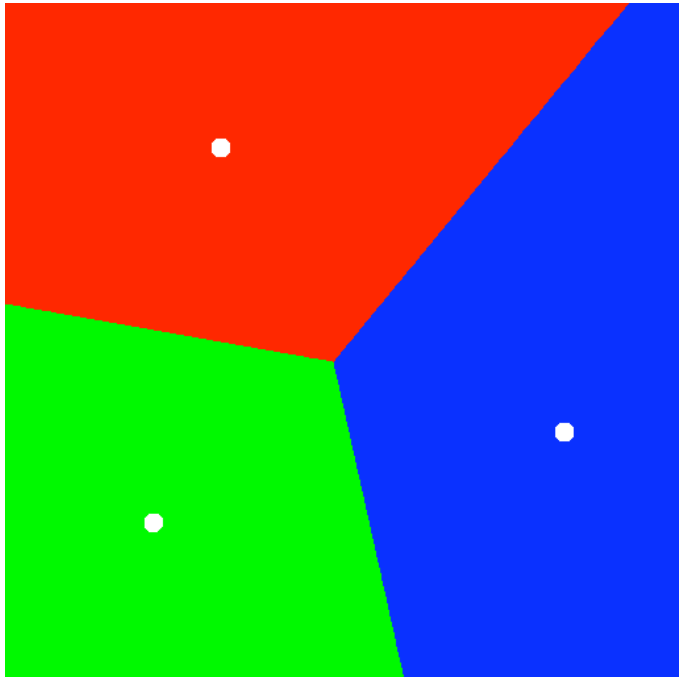
- Incremental
  - We will learn about this in Chapter 9, too

Other Ways Computing Voronoi Diagram

- Using Graphics hardware (GPU)



Perspective, 3/4 view



Parallel, top view

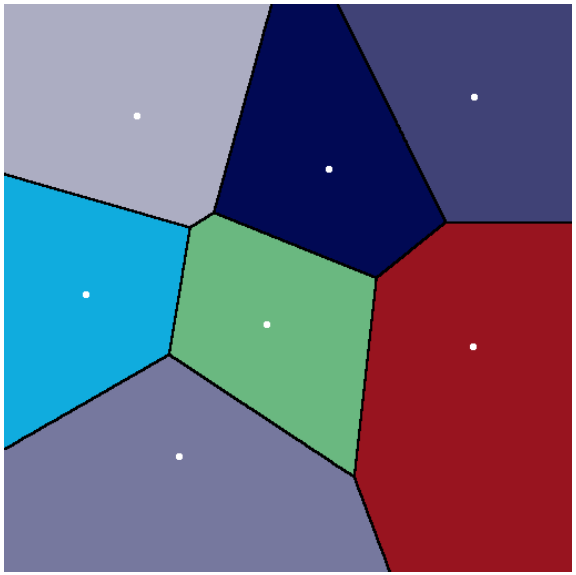
Kenneth E. Hoff III, Tim Culver, John Keyser, Ming Lin, and Dinesh Manocha, 99

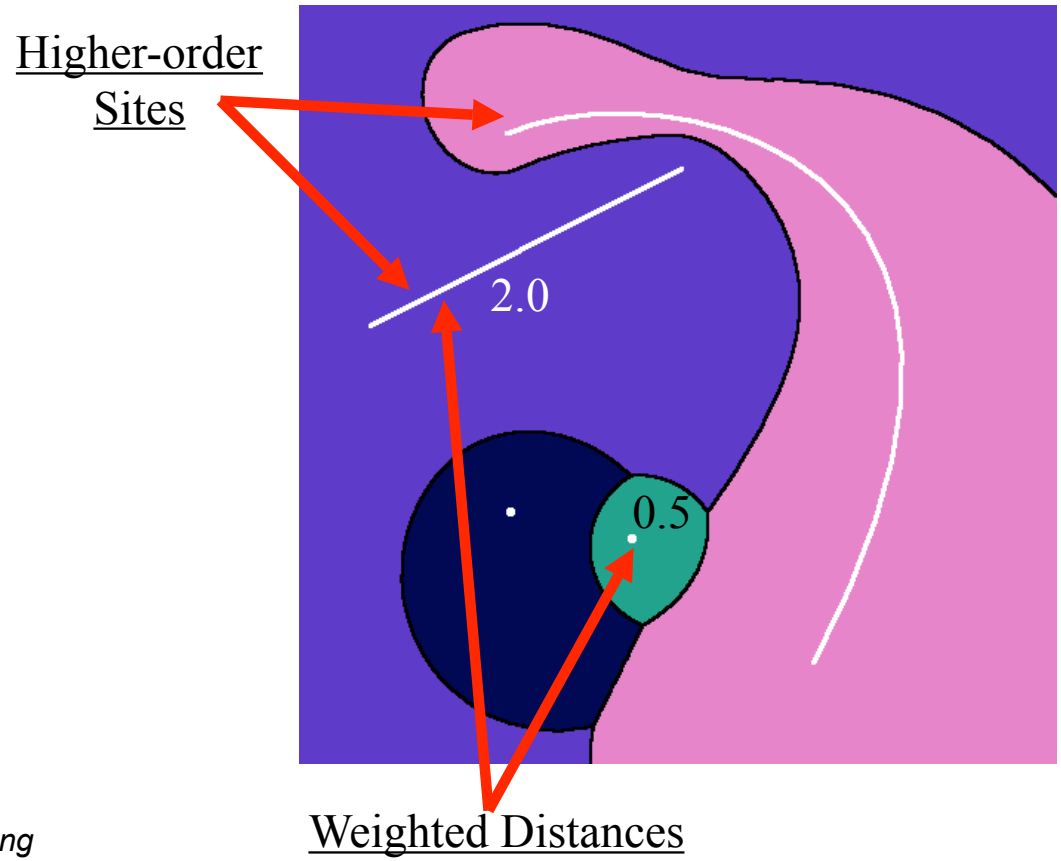
Ordinary

- Point sites
- Nearest Euclidean distance

Generalized

- Higher-order site geometry
- Varying distance metrics





Kenneth E. Hoff III, Tim Culver, John Keyser, Ming Lin, and Dinesh Manocha, 99

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

Homework Assignment

- 7.10, 7.11