Construction of Voronoi Diagrams

Outline:

I. Circle event

II. Data structures

III. Handling of circle events

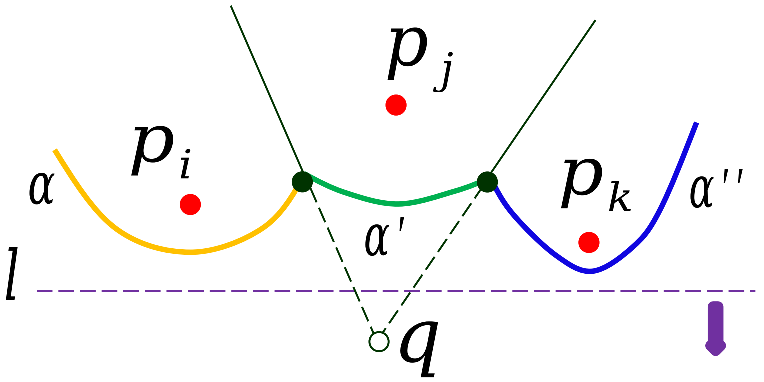
IV. Algorithm and time complexity

V. Handling of degeneracies

VI. Applications

I. Circle Event

An existing arc shrinks to a point and disappears afterward.



(a) Before

Event also referred to as .

Circle Event Summary

At a circle event:

- An existing arc drops out.
- Two growing Voronoi edges merge at a vertex.

Lemma An existing arc disappears from the beach line only through a circle event.

II. Data Structures

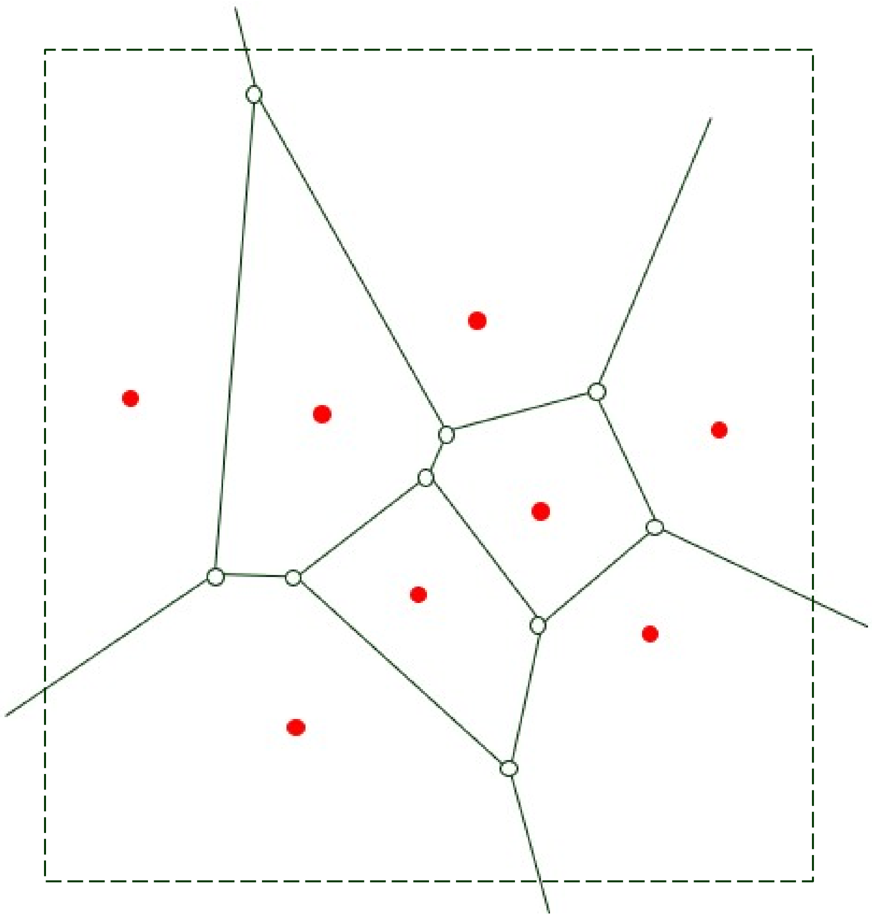
a) A data structure to store part of Vor computed so far.

b) Standard data structures for sweeping.

- event queue
- sweep line status

Voronoi Diagram Storage

As a planar subdivision (DCEL)

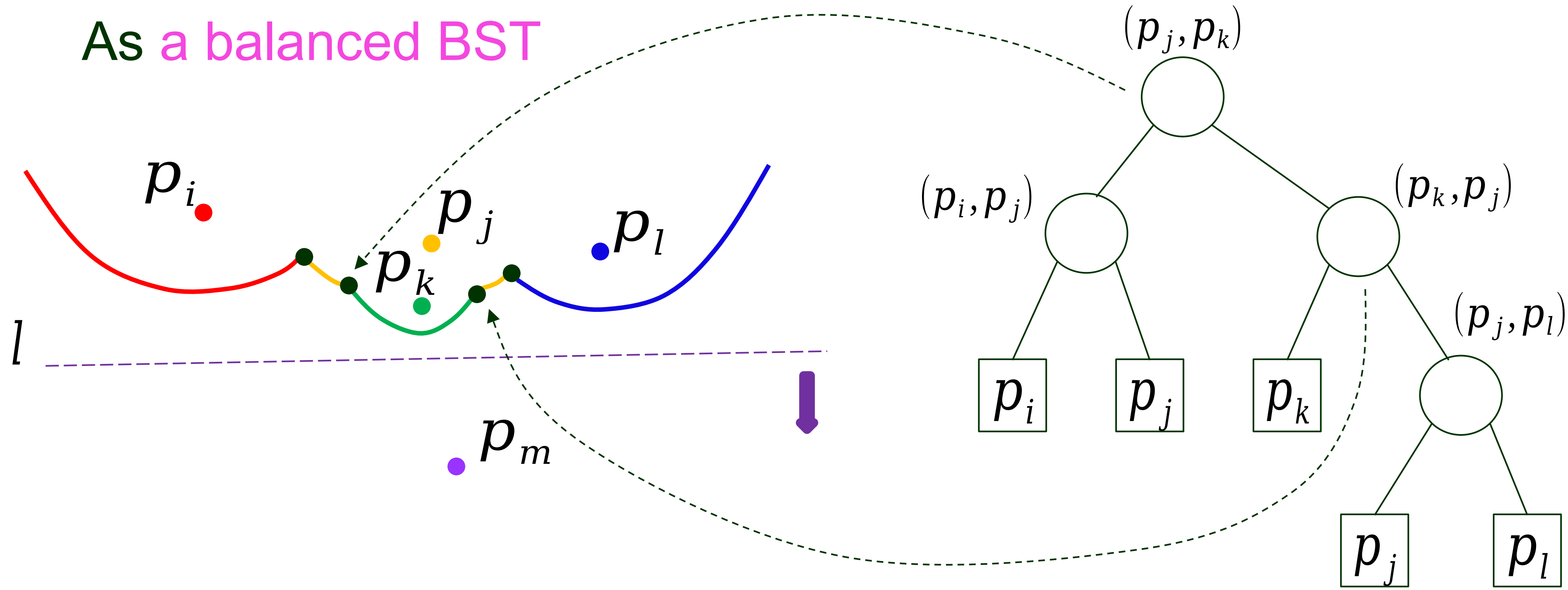


Bounding box

- added after the computation
- large enough to contain all vertices

Beach Line Storage

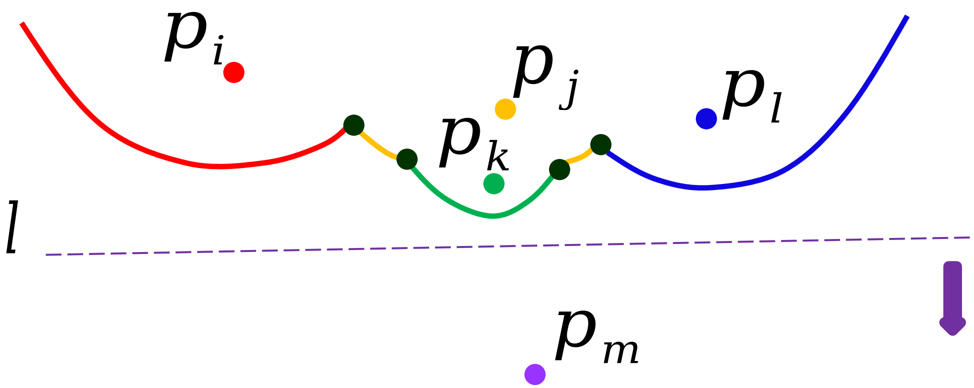
As a balanced BST

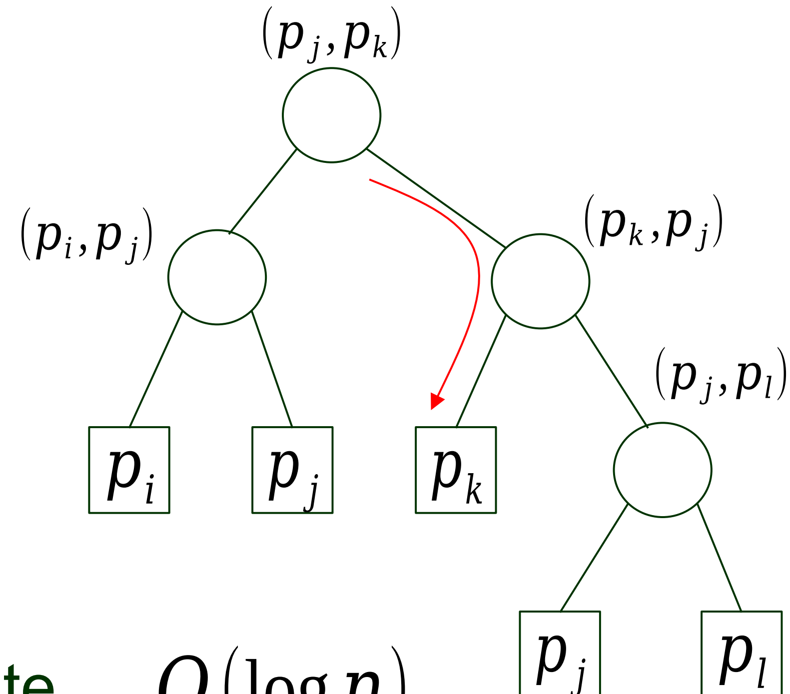


- *Leaves*: arcs of the beach line
  - ordered from left to right
  - labeled with their defining sites
- *Internal nodes*: break points

  : joining a left arc defined by
  and a right arc defined by

Arc Above a New Site





How to find the arc above a new site $O(\log n)$

- Compare its -coordinate with that of the breakpoint stored at an internal node.
- The -coordinate of can be computed in time .

Event Queue

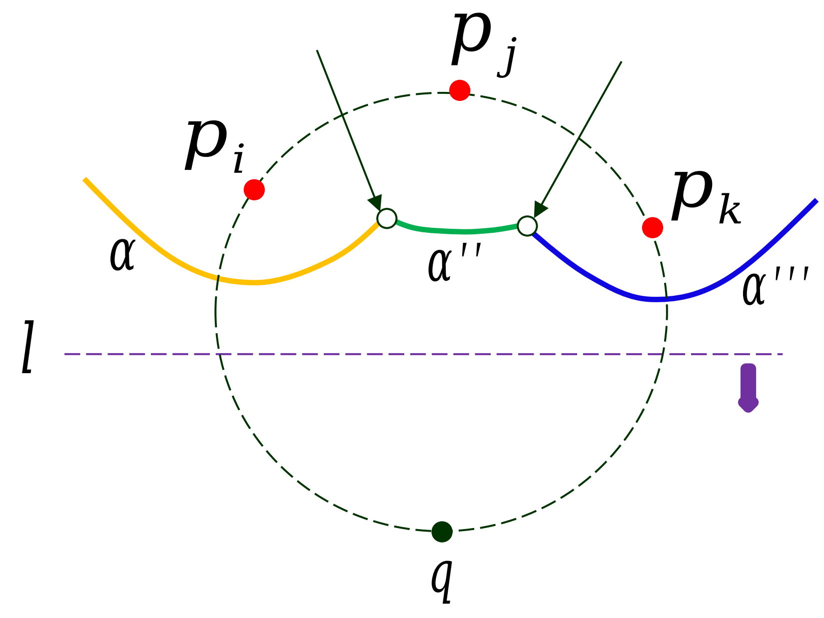
As a priority queue in -coordinate

- For a site event, store the site.

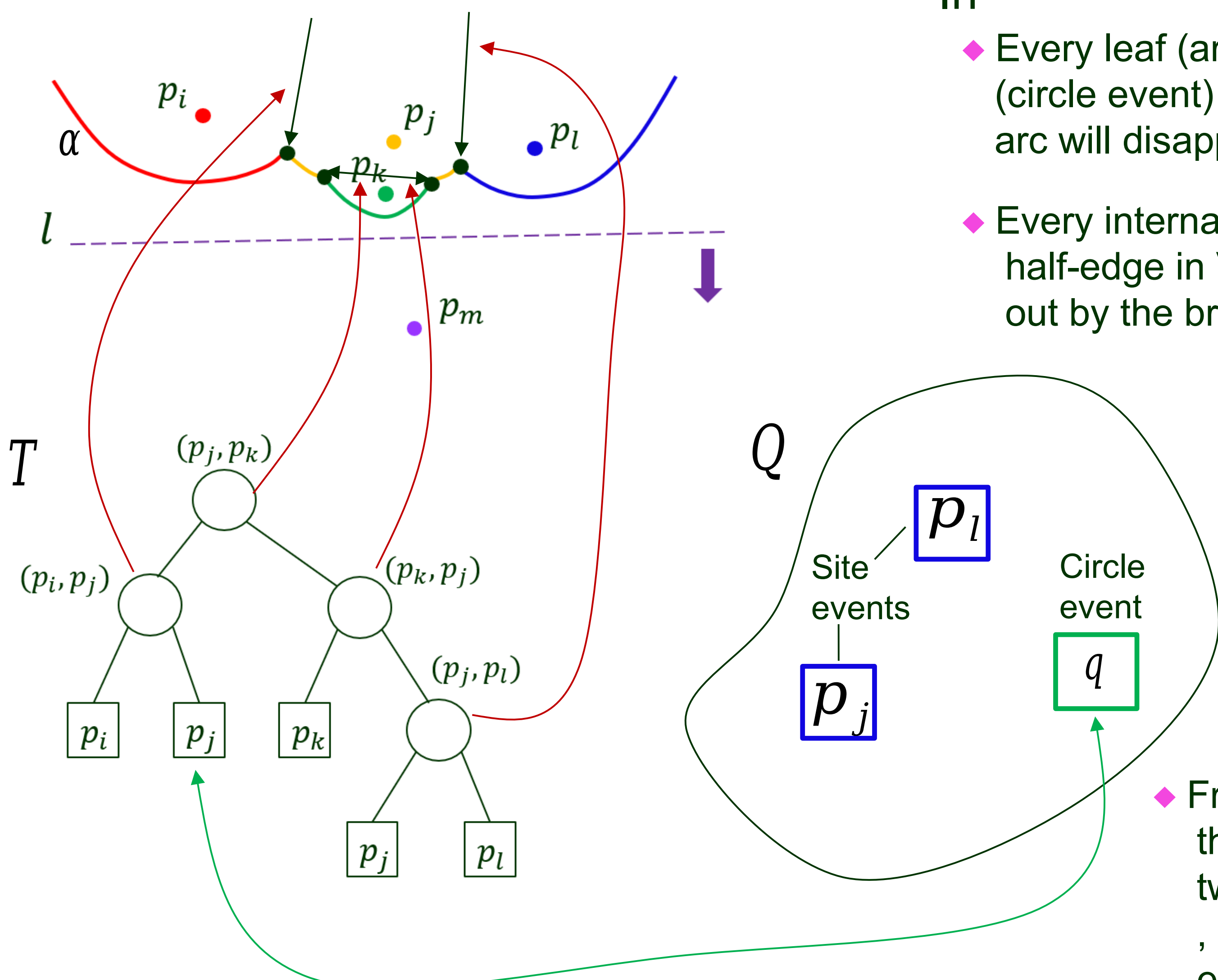
- For a circle event, store

  - the lowest point () of the circle

  - a pointer to the leaf representing the arc to disappear



Cross Referencing



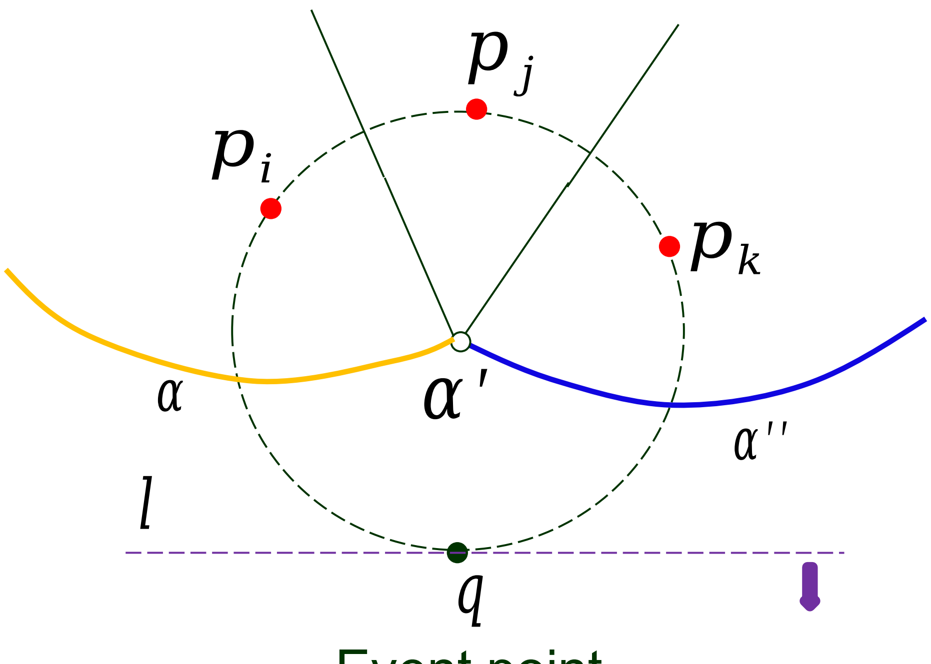
In

- Every leaf (arc) points to a node (circle event) in , in which the arc will disappear.
- Every internal node points to a half-edge in Vor being traced out by the break point.

In

- Every circle event points to the leaf in representing the arc to disappear.
- From the site defining the arc we can find its two adjacent sites and , retrieving the circle event .

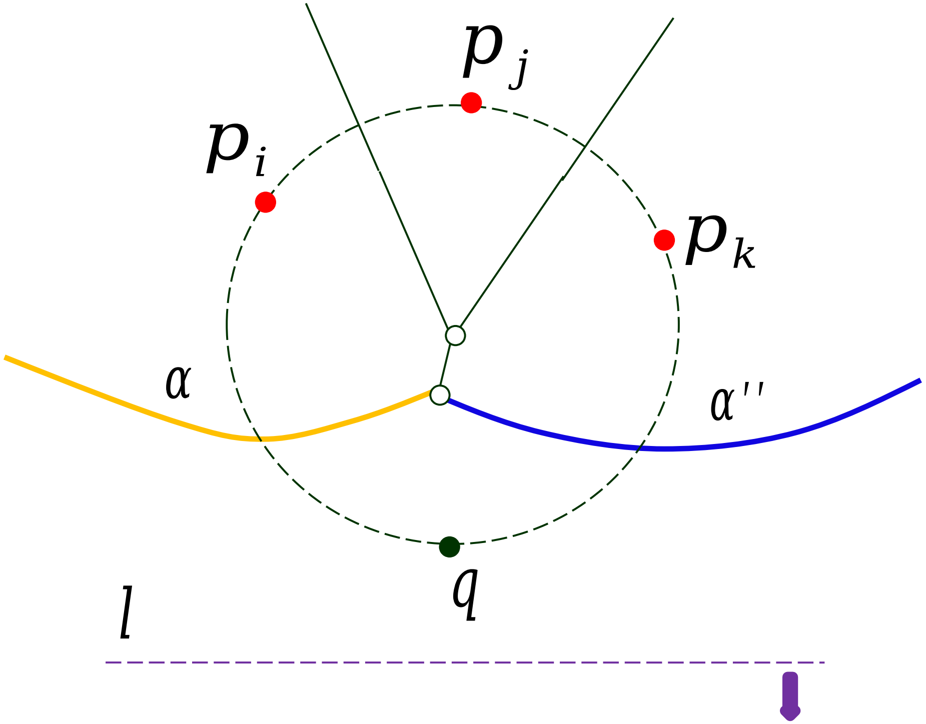
III. Detection of a Circle Event



is to disappear
when is on .

Handled Event

Every three consecutive arcs on the beach line introduce a circle event.

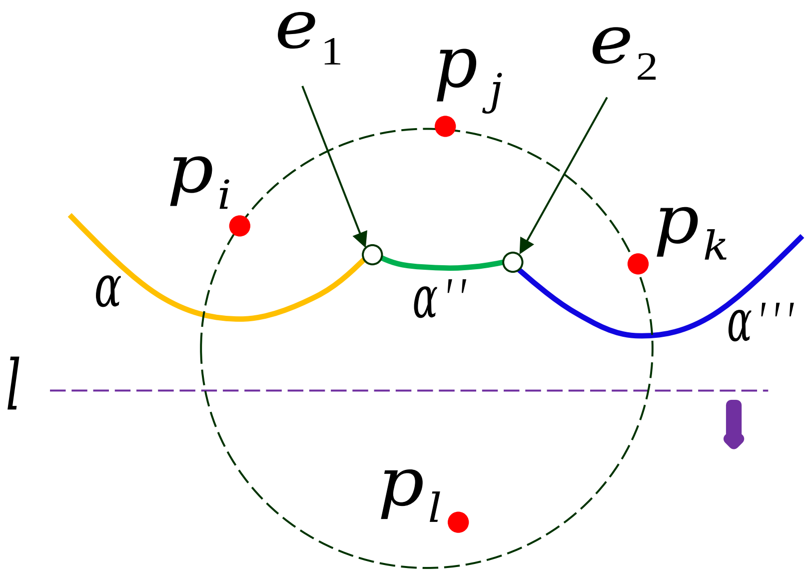


♣ Circle completely above .

⇩

Event () has been dealt with.

False Alarm



♣ Circle contains some other site.

⇩

Event should not be handled.

The two Voronoi edges and being traced out will not meet at a vertex.

Lemma Every Voronoi vertex is detected at a circle event.

Structural Changes of the Beach Line

The beach line *changes its topological structure* at every event.

Triples of adjacent arcs are introduced or destroyed.

Triple

Delete
Add

Circle Event Insertion

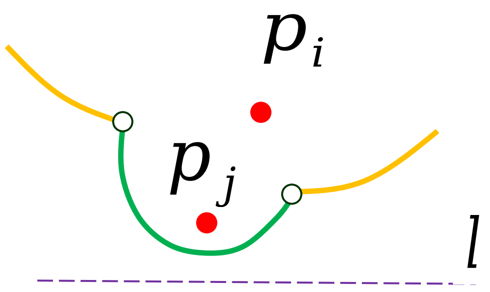
Circle event will happen only if the corresponding triple stays until the sweep line *reaches the lowest point* of the circle.

- Insert a new triple of consecutive arcs as a circle event if all three conditions below hold:

  a) the arcs are defined by 3 sites instead of 2;

  b) the triple is not in the event queue

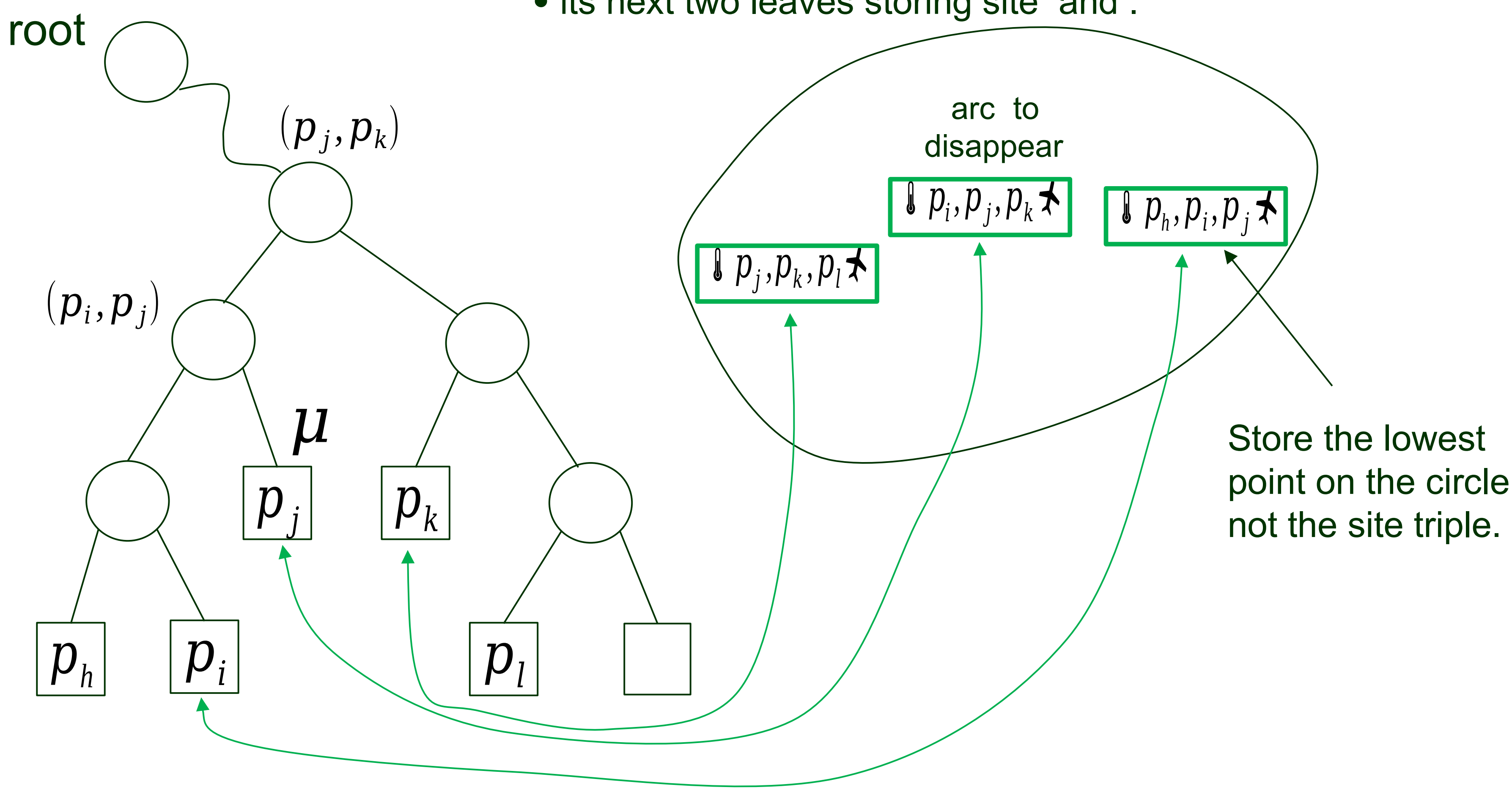
  c) the circle through the three defining sites intersects the sweep line.



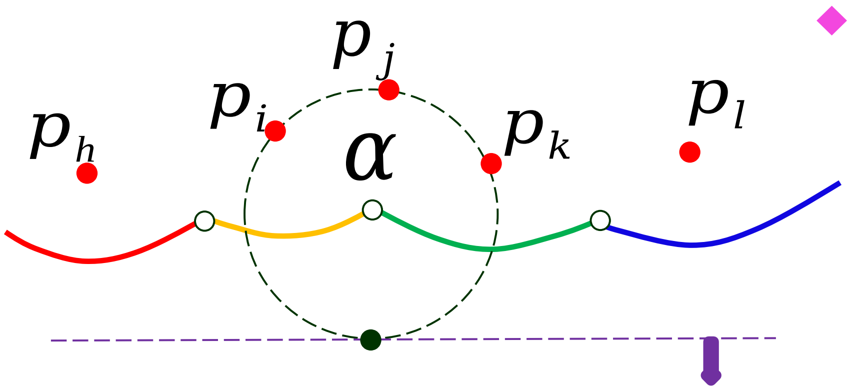
not inserted

Circle Event Deletion

- Takes place when arc disappears or splits. To find all affected circle events, search in for
  - the leaf storing the site defining .
  - its previous two leaves storing sites and .
  - its next two leaves storing site and .



Circle Event Deletion



- In the case arc disappears

  Delete $\langle p_h,p_i,p_j \rangle$

  $\langle p_i,p_j,p_k \rangle$

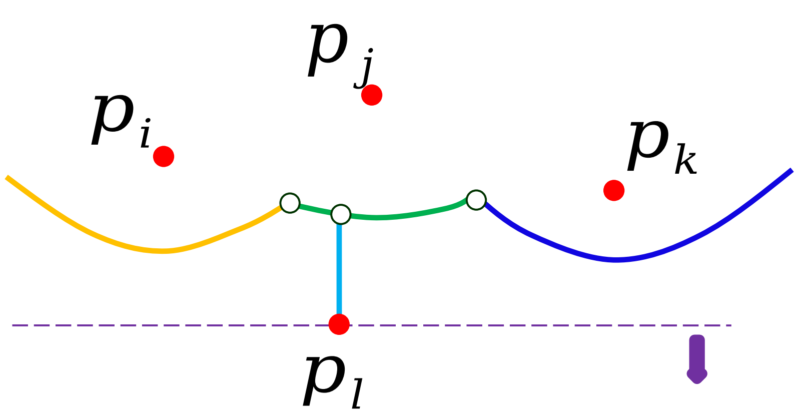
  $\langle p_j,p_k,p_l \rangle$

  May have been removed already

  Circle events involving

  Add if and circle intersecting

  if and circle intersecting



- In the case arc splits due to a site event

  Delete $\langle p_i,p_j,p_k \rangle$

  Add $\langle p_i,p_j,p_l \rangle$

  $\langle p_l,p_j,p_k \rangle$

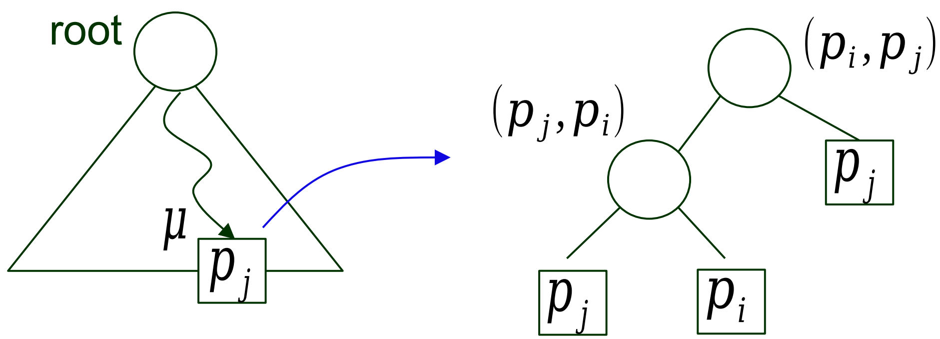
IV. Construction Algorithm

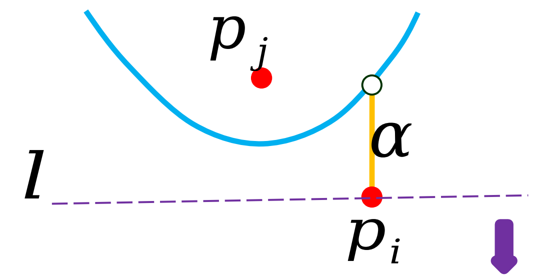
```
VoronoiDiagram()

1. Initialize event queue  with
2. while
3.     do extract event  with the largest -coordinate from .
4.         if  is a site event at
5.             then HandleSiteEvent()
6.             else HandleCircleEvent() //  is the lowest
                                 // point on the corresponding
circle.
7. Compute a bounding box with all Voronoi vertices in its
interior.
8. Update DCEL for half-infinite edges.
9. Traverse DCEL to add cell records and related pointers.
```

Site Event Handling

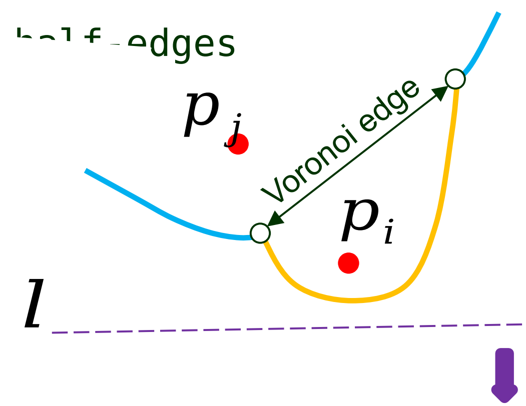
`HandleSiteEvent()`

```
1.  arc vertically above  in the plane // search in to find defined by
2. { circle events involving  }
3. Replace the leaf  with a subtree below.
```





half-edges

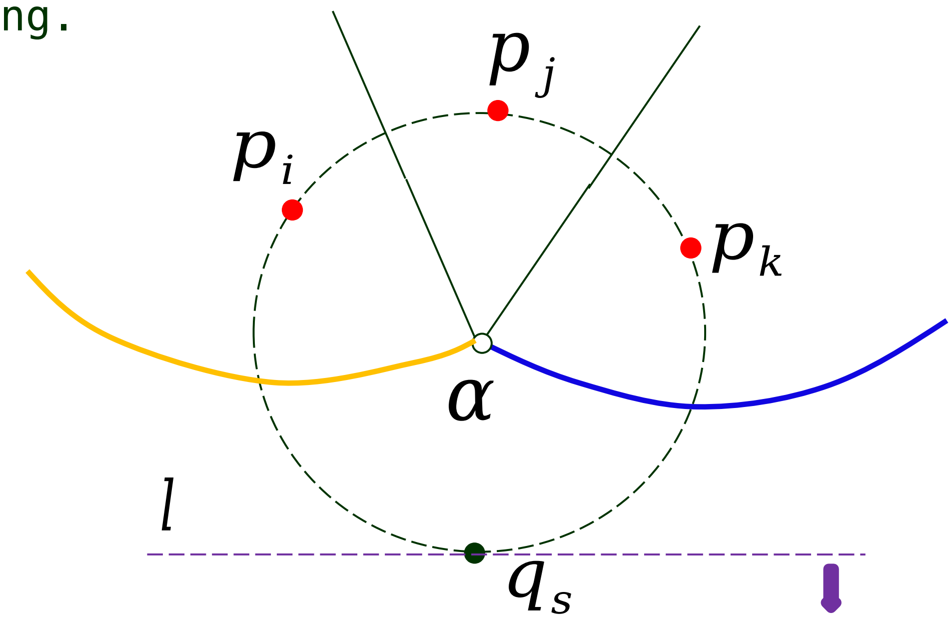


,.

Circle Event Handling

HandleCircleEvent()

1. arc vertically above to disappear // search in
2. { circle events involving }
3. Delete leaf representing in .
4. Update the tuples (representing the breakpoints) at the internal node
5. Add the center of the circle as a vertex record to DCEL.
6. Create two half-edge records (between cells and).
7. Check new triples that arise: and , inserting
   each as a circle event into if the two breakpoints in the
   triple are converging.



Time Complexity

Theorem The algorithms runs in time using storage.

Proof (sketch)

- Insertions and deletions in and take each.
- Operations on DCEL take each.
- Each event requires operations thus time to process.
- site events
- #circle events proportional to #vertices

- false alarms simply deleted

- vertices handled

□

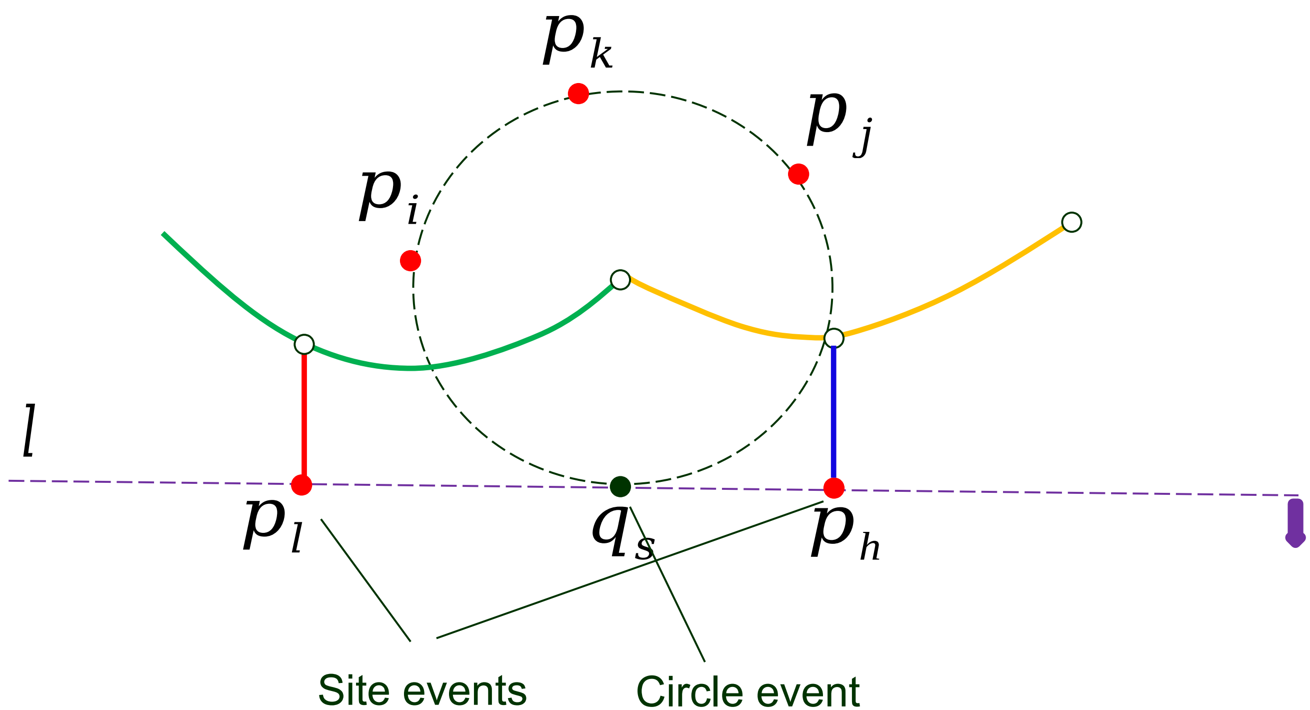
V. Degeneracy (1)

Two or more event sites with the same -coordinate.

1a) different -coordinates

- site and/or circle events.

Handle them in any order.



Degeneracy (1) – cont'd

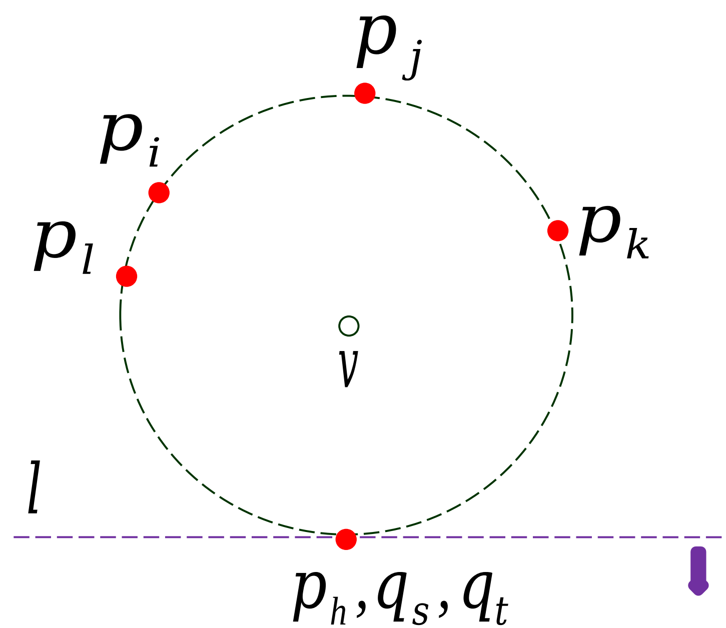
1b) same -coordinate

- site event and circle events, or
- site event and circle event

⇩

a Voronoi vertex of degree

- Handle circle events first.
  - as vertices coinciding
  - each with degree 3 and 0 length in between
- Handle the sole site event last.



Three events coincide:

p_h

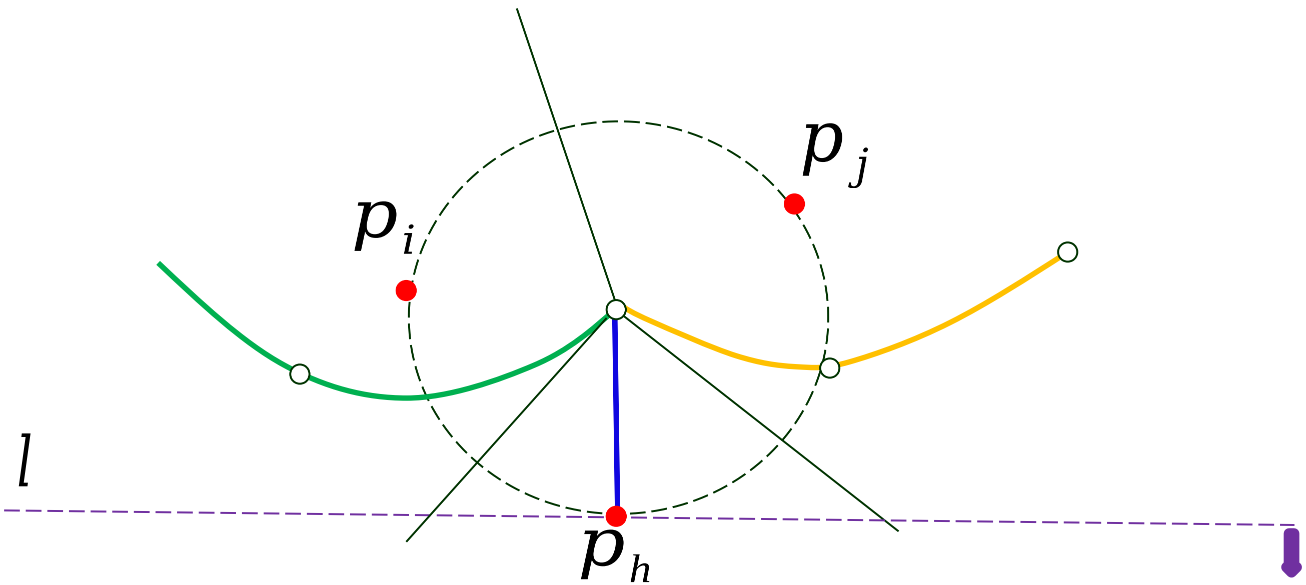
$q_s =$ 🌡 p_l, p_i, p_j ✈

$q_t =$ 🌡 p_i, p_j, p_k ✈

Degeneracy (2)

A new site appears right below a break point on the beach line.

Handle it as a circle event, and
then add a new arc (for).



VI. Applications of Voronoi Diagrams

Given points in the plane, we can use their VD to solve many problems efficiently.

Closest Pair: Find two points that are the closest.

All Nearest Neighbors: Find the nearest neighbor of every point.

Euclidean Minimum Spanning Tree: Construct a tree of minimum total length whose vertices are the given points.

Triangulation: Join the points by nonintersecting straight line segments so that every region internal to the convex hull is a triangle.

Nearest Neighbor Search: With preprocessing allowed, how quickly can a nearest neighbor of a new given query point be found?