# CMSC 754 - Computational Geometry
## Lecture 10: Voronoi Diagrams

**Metric Spaces:** Distances modeled as **metric space** $(X, f)$: $f: X \times X \to \mathbb{R}^{\geq 0}$, s.t. for all $p, q, r \in X$:

- **Symmetry**: $f(p,q) = f(q,p)$
- **Positivity**: $f(p,q) \geq 0$ and $f(p,q) = 0$ iff $p = q$
- **Triangle Inequality**: $f(p,q) \leq f(p,r) + f(r,q)$

**Euclidean Distance:** for $p, q \in \mathbb{R}^d$:

$$\| p - q \| = \left[ \sum_i (p_i - q_i)^2 \right]^{1/2}$$

**Voronoi Diagram:**

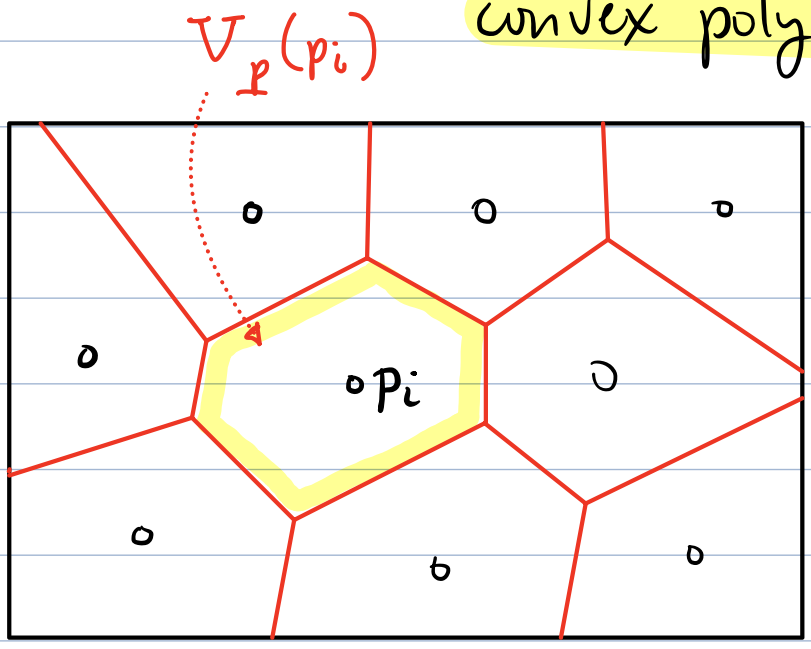
A fundamental structure for metric spaces.

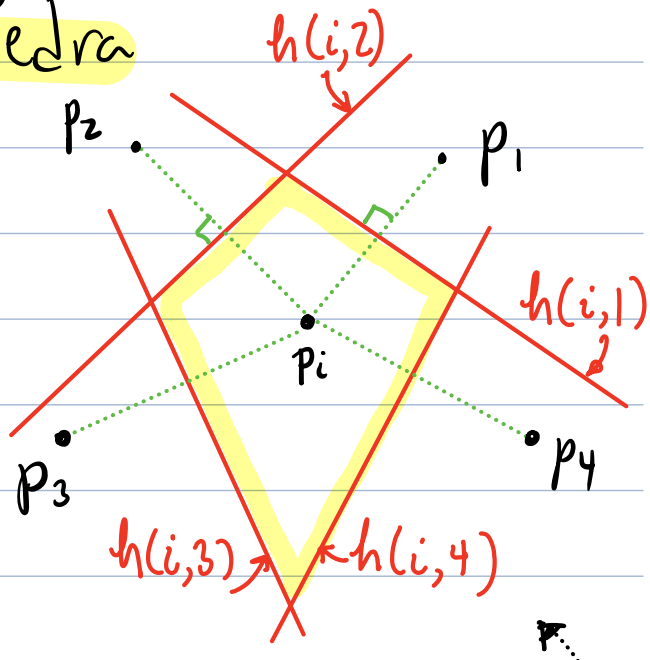
Given a point set $P = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^d$ called **sites**, we want to subdivide space based on each site's "**region of influence**"

**Def:** Voronoi cell for site $p_i$

$$V_P(p_i) = \{ q \in \mathbb{R}^d \mid \|p_i - q\| < \|p_j - q\|, \forall j \neq i \}$$

**Obs:** - Voronoi cells are disjoint
- For Euclidean dist, Voronoi cells are (possibly unbounded) convex polyhedra




Let $h(i,j) = \{ q \mid \|p_i - q\| < \|p_j - q\| \}$

$h(i,j)$ - halfspace bounded by perpendicular bisector between $p_i$ & $p_j$

$\mathrm{Vor}(p_i) = \bigcap_{j \neq i} h(i,j)$ - intersection of halfspaces $\Rightarrow$ polytope

**Def:** Vor(P) is the subdivision (cell complex) induced by P's voronoi cells.

- Vor(P) covers $\mathbb{R}^d$
- Has n cells (faces of dim d)
- Polyhedral subdivision (for Euclidean dist)
- Combinatorial complexity:
  $\mathbb{R}^2$: $O(n)$ edges + vertices
  $\mathbb{R}^d$: $O(n^{\lceil d/2 \rceil})$ size [Closely related to convex polytopes in $\mathbb{R}^{d+1}$]

**Many applications:**

**Nearest neighbor search:**

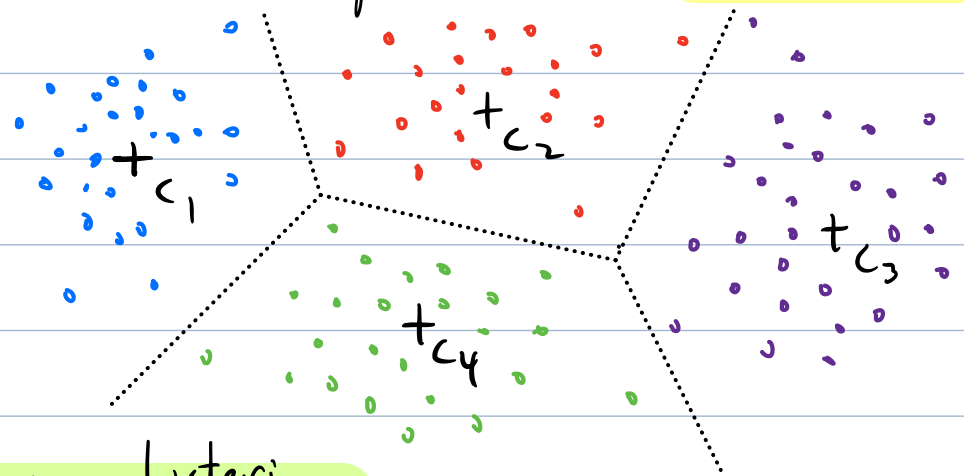
Preprocess a set of sites $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$ s.t. given any query point $q \in \mathbb{R}^d$ can find q's nearest site

How? - Compute Vor(P)
- Build a point-location data structure for Vor(P)

[Optimal in $\mathbb{R}^2$. Not as good in $\mathbb{R}^d$.]

## Point-based Clustering:

- Given set T of training points, group them into k clusters
- Clusters are defined by k cluster centers $\{c_1, \ldots, c_k\}$
- Cluster membership based on closest center




- k-means clustering

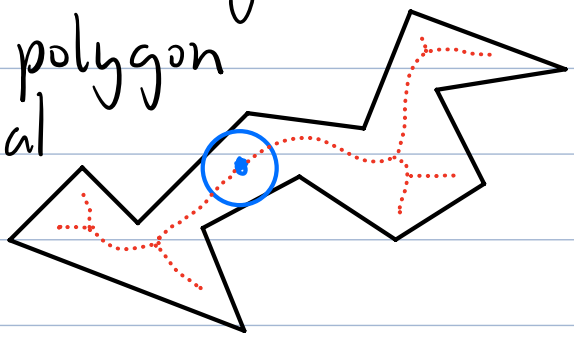
## Variations:

- Other metrics: $L_1$-Vor diagram (Manhattan distance)
- Weighted pts:
  - Multiplicative: $dist(q, p_i) = \alpha_i \| p_i - q \|$
  - Additive: $dist(q, p_i) = \| p_i - q \| + \omega_i$

- $k^{th}$ Nearest:
  - $Vor_k(P)$ = subdivide based on $k^{th}$ closest

$Vor_n(P)$ = farthest point Vor. diag

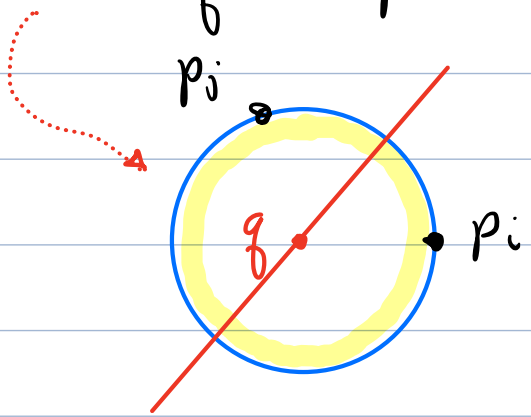
- Other shapes:
  - Voronoi diagram of line segments
  - Medial axis of polygon
    centers of maximal disks

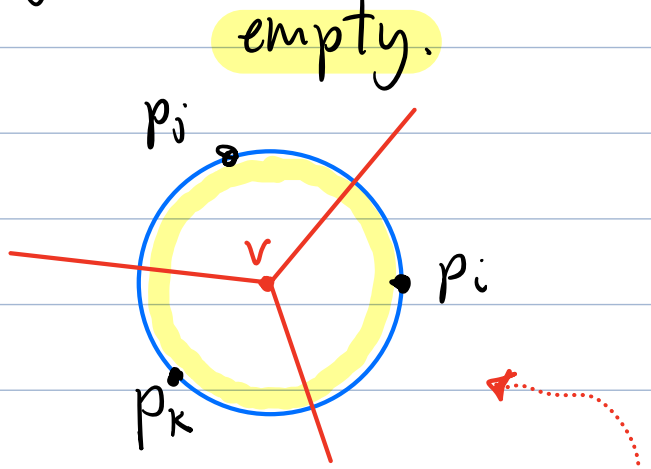


## Properties of the Voronoi Diagram:

### Empty-circle Property:

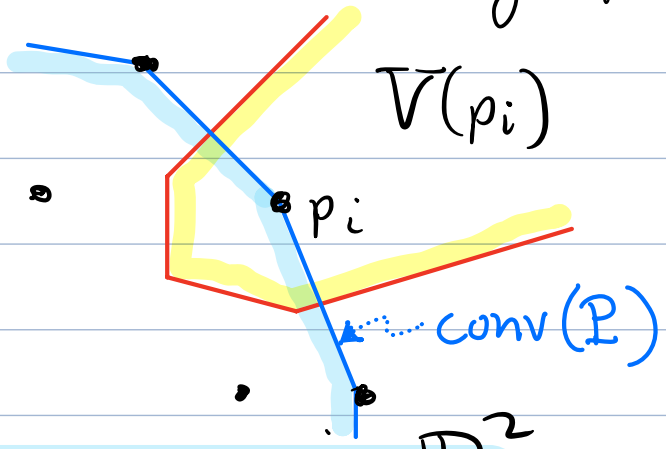
A pt $q$ is on an edge of the Vor. diag iff there is a circle centered at $q$ that passes through 2 sites & is otherwise empty.




**Circumcircle Property:** A pt $v$ is a vertex of the diagram iff it is the center of a circle passing through 3 sites & is otherwise empty.

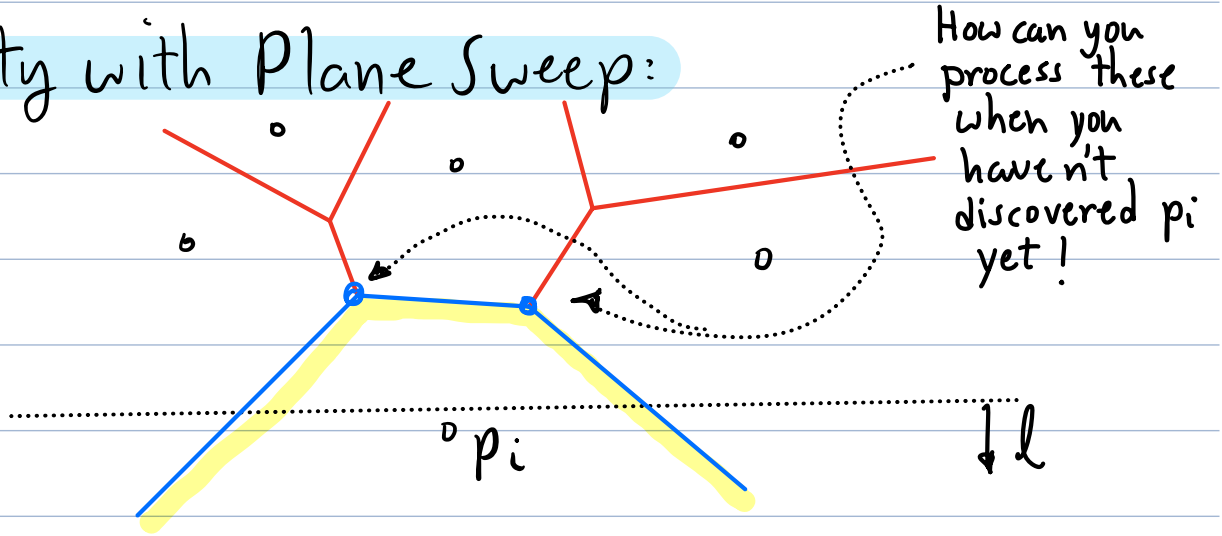
**Hull Property:** A site $p_i$ has an **unbounded Voronoi cell** iff $p_i$ is on boundary of **convex hull** of $\mathbb{P}$.




## Constructing Voronoi Diagrams in $\mathbb{R}^2$

- **Incremental** – add a site; update (best if randomized)
- **Divide + Conquer** - $O(n \log n)$
- **Plane Sweep** (this lecture)
  - Fortune's Algorithm - $O(n \log n)$

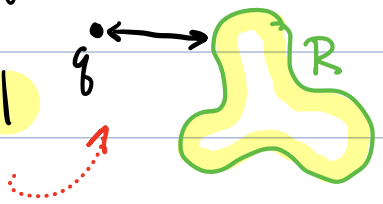
## Difficulty with Plane Sweep:



**Clever twist:** We'll maintain two sweeping structures: **sweep line** + **beach line**

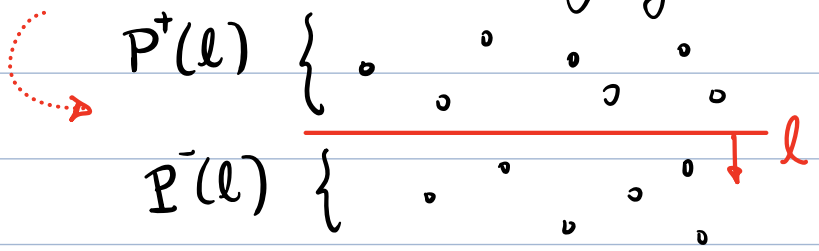
Def: Given a set of pts $R$ and pt $q$, define

$$\text{dist}(q, R) = \min_{p \in R} \|p - q\|$$



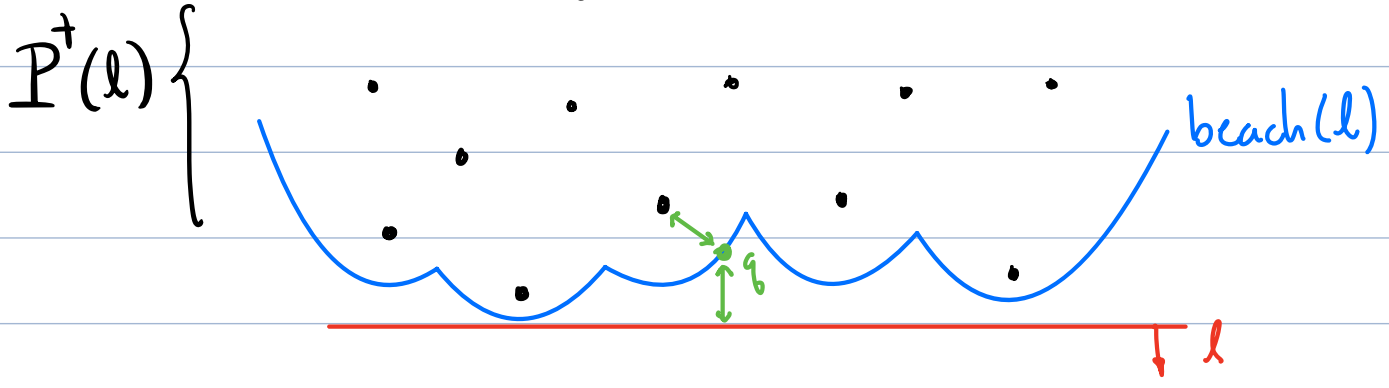

Given a sweep line $\ell$ (horizontal + moving down) define

$P^+(\ell)$ to be sites lying above $\ell$



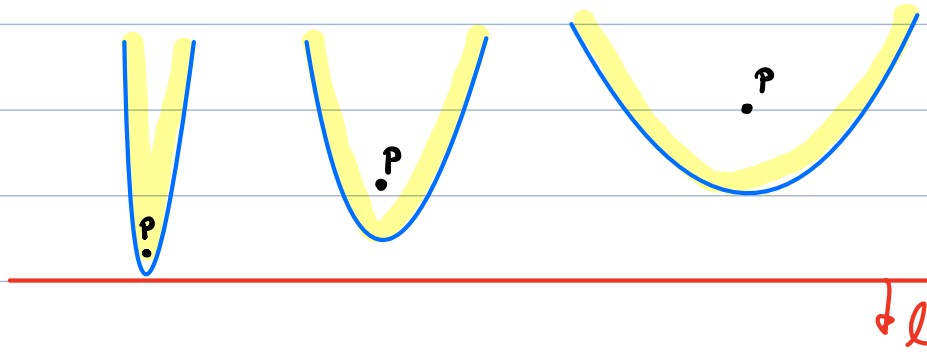

Given sweep line $\ell$, define the **beach line** to be set of pts $q \in \mathbb{R}^2$ that are **equidistant from $P^+(\ell)$ and $\ell$**

$$\text{beach}(\ell) = \{ q \in \mathbb{R}^2 \mid \text{dist}(q, P^+(\ell)) = \text{dist}(q, \ell) \}$$



## Beach-line Structure:

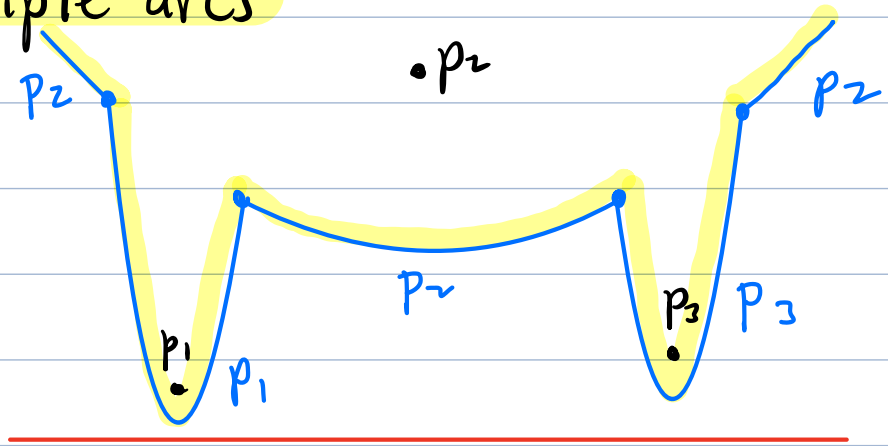
The points equidistant to a site $p$ + line $\ell$ form a **parabola**




(wider as $p$ is higher)

The beach line is the **lower envelope** of these parabolas for all sites in $P^+(\ell)$

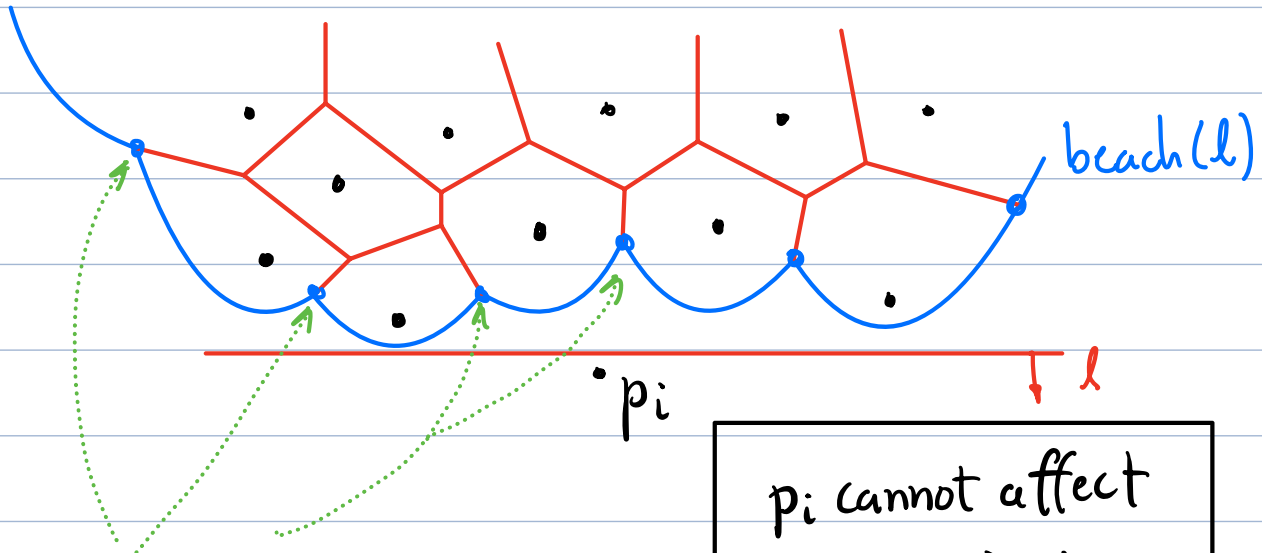
- Beach line is **$x$-monotone**
- A single site may contribute **0, 1, or multiple arcs**




- **Total complexity** is $\mathcal{O}(|P^+(\ell)|) = \mathcal{O}(n)$

[Proof: Exercise]

**Key:** The portion of Vor(P) **above** the beach line is **safe** from sites lying **below** $\ell$.



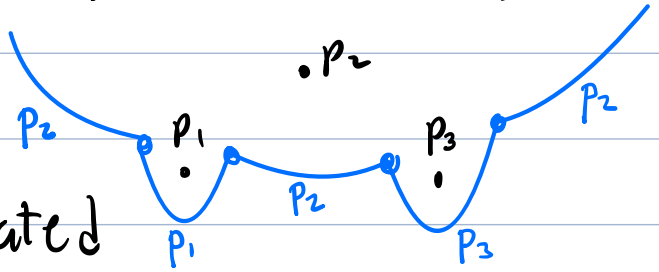

Break points

$p_i$ cannot affect Vor(P) above beach($\ell$)

## Fortune's Algorithm:

**Sweep-line status:**

- **y-coord** of sweep line
- **seq. of sites** (left to right) that contribute arc to beach line (eg. $\langle 2,1,2,3,2 \rangle$)

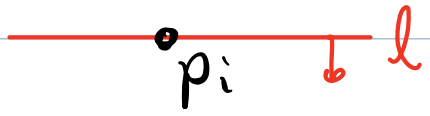



- Parabolic arcs not computed
- Breakpts generated as needed

**Voronoi diagram:** Portion of Voronoi diagram (rep. as DCEL) above beach line is stored/updated.

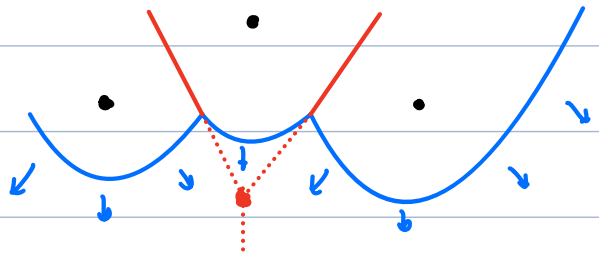
## Events:

**Site event:** Sweep line passes over a site




**Vertex event** (circle event):
- A new Voronoi vertex is discovered

≡ An arc on beach line vanishes

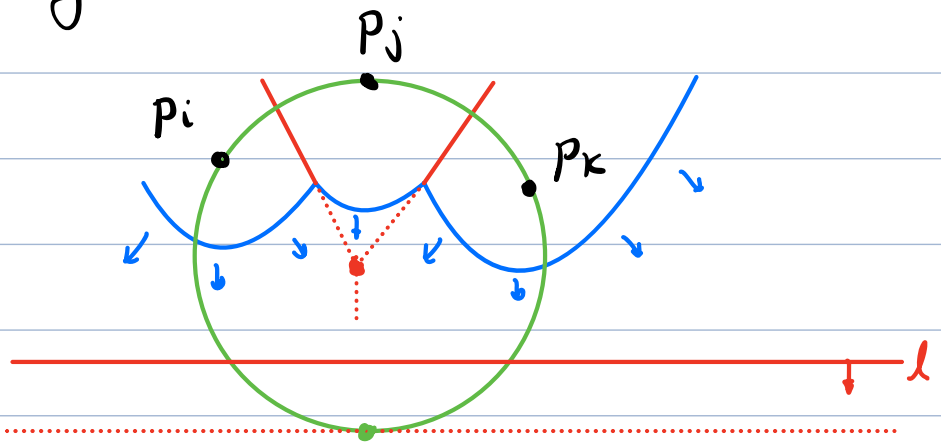


**Priority Queue:** Stores y-coords for sweep line at events.

**Site events:** Easy - just y-coord of site (static)

**Vertex events:** Tricky! (see below)

# Scheduling vertex events:



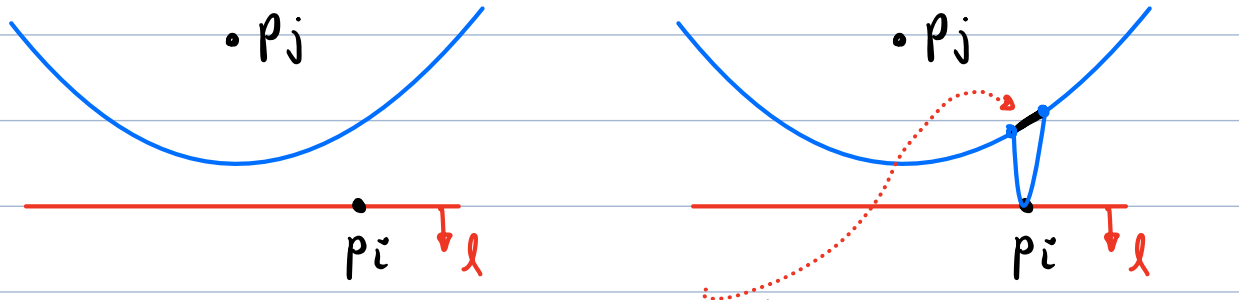

- For each consecutive triple $\langle \ldots p_i, p_j, p_k \ldots \rangle$ on beach line compute lowest y-coord of circumcircle $(p_i, p_j, p_k)$
- Schedule vertex event when sweep line reaches this y-coord.

**Site Event:** for site $p_i$

(1) Find arc of beach line above $p_i$

(2) Split this arc:

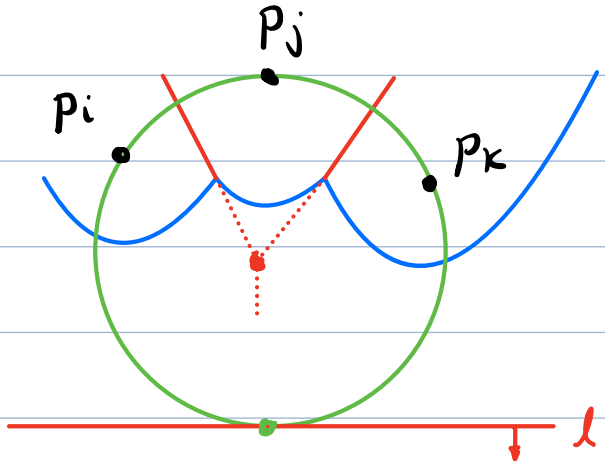
$$\langle \ldots, p_j, \ldots \rangle \Rightarrow \langle \ldots p_j, p_i, p_j, \ldots \rangle$$

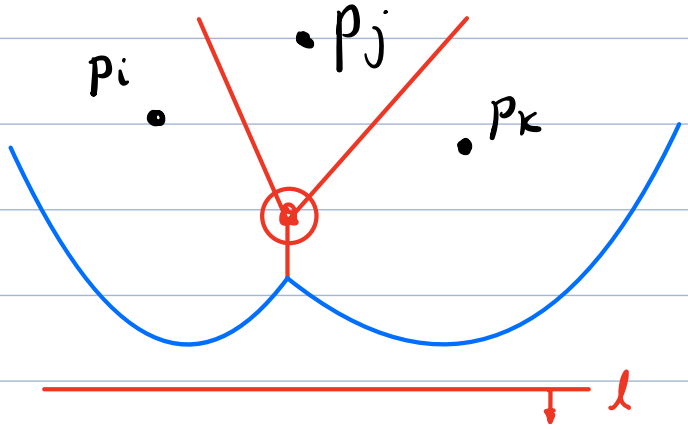



(3) Create a new "dangling" edge (between $p_i$ & $p_j$) add to Voronoi diagram.

(4) Update priority queue vertex events (below)

Vertex Event: for triple $\langle p_i, p_j, p_k \rangle$




(1) Delete $p_j$'s arc from beach line

$$\langle \dots p_i\ p_j\ p_k \dots \rangle \Rightarrow \langle \dots p_i\ p_k \dots \rangle$$

(2) Create new Voronoi vertex joining edges $p_ip_j$ + $p_jp_k$ in diagram

(3) Start new (partial) Voronoi edge for $p_ip_k$

(4) Update priority queue vertex events

Analysis:
- $O(n)$ events
- $O(\log n)$ per event
- $O(n \log n)$ total time