CMSC 754 - Computational Geometry
Lecture 10: Voronoi Diagrams

Metric: A function that returns the **distance** between points

Symmetry: $dist(p,q) = dist(q,p)$

Positivity: $dist(p,q) \geq 0$ + $dist(p,q) = 0$ iff $p = q$

Triangle Inequality:

$$dist(p,q) \leq dist(p,r) + dist(r,q)$$

Examples:

Euclidean: $\|p-q\| = \left(\sum_{i=1}^{d} (p_i - q_i)^2\right)^{1/2}$

Minkowski L_k: $\|p-q\|_k = \left(\sum_{i=1}^{d} |p_i - q_i|^k\right)^{1/k}$

Edit (Levenshtein) Distance: strings (comp. biology)

Frechét/Hausdorf Distance: for curves/shapes

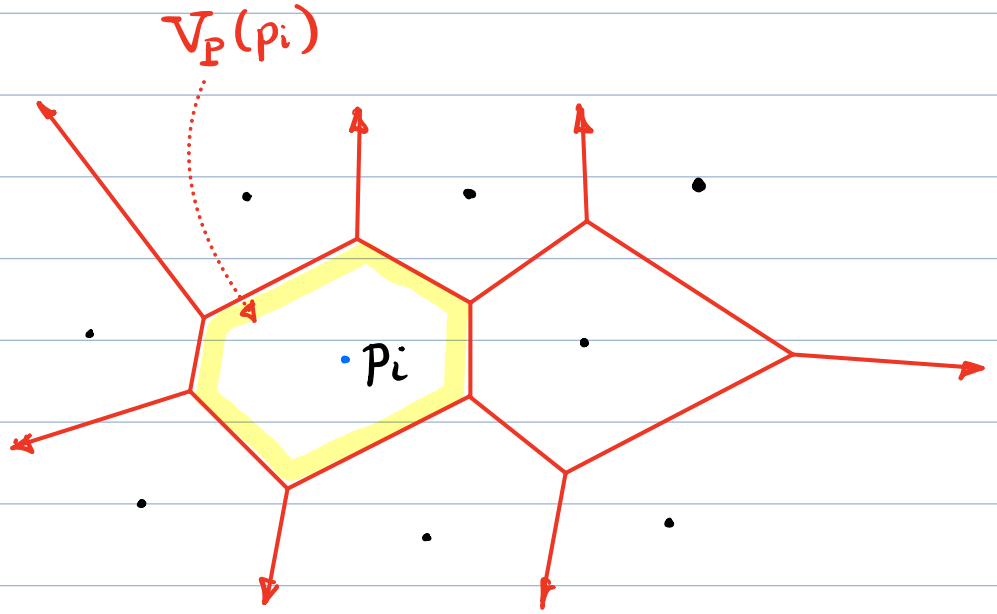
Voronoi Diagram: Given a **point set** $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$ called **sites**, subdivide space into **regions** based on **which site is closest** ("regions of influence")

Formally - For $p_i \in P$, define the **Voronoi cell**:

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

Observe:

- Voronoi cells have disjoint interiors
- Cover $\mathbb{R}^d$
- Voronoi cells are the faces of the Voronoi diagram
- Each cell is a (possibly unbounded) convex polytope



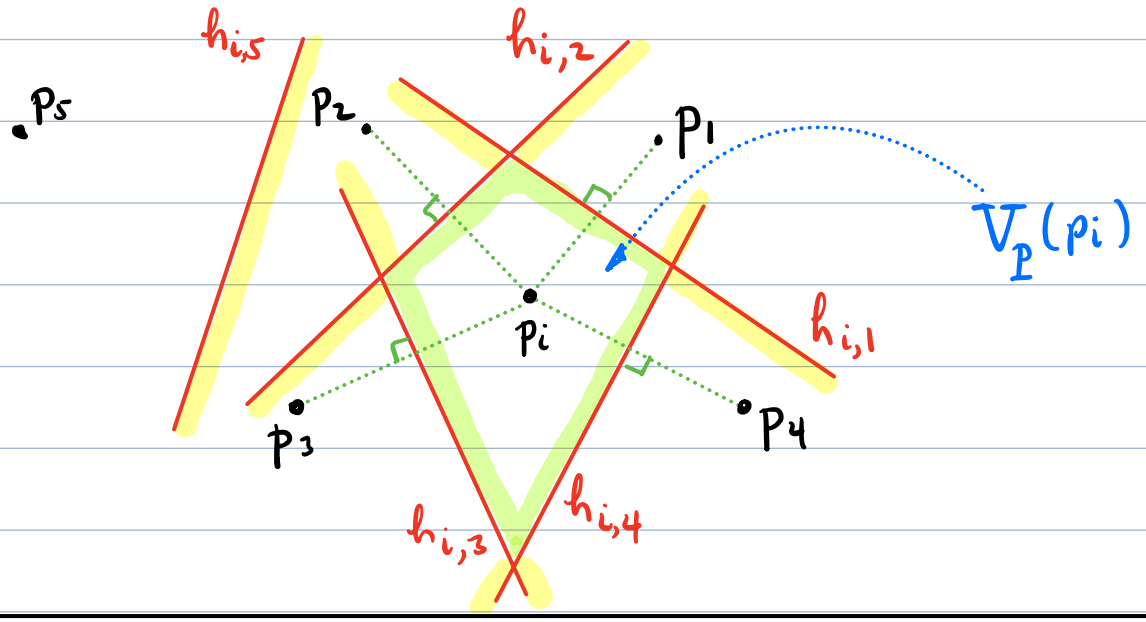
Why convex?

Def: $h_{i,j} = \{ q \in \mathbb{R}^d \mid \|p_i - q\| \leq \|p_j - q\| \}$

In Euclidean space, $h_{i,j}$ is a halfspace
(bounded by perpendicular bisector $p_i \leftrightarrow p_j$)

Clearly:

$V_P(p_i) = \bigcap_{j \neq i} h_{i,j}$ = intersection of n-1 halfspaces $\Rightarrow$ convex polytope



Lemma: Given a n-element point set $P \subseteq \mathbb{R}^2$,

$Vor(P)$ has:

(i) n faces

(ii) $2n-5 = O(n)$ vertices

(iii) $3n-6 = O(n)$ edges

Proof: (Optional)

- Let h = number of unbounded edges
  - imagine joining them to a vertex at infinity
  - $h \geq 3$
- Let v = number of regular vertices. e = no. edges
- Assuming general position, each regular vertex has degree 3 $\Rightarrow$ $3v + h = 2e$

  $\Rightarrow v = (2e-h)/3$

  $\Rightarrow e = (3v+h)/2$

Degree of vertex at ∞

- No. of faces f = no. of sites n $\Rightarrow$ (i)
- By Euler's formula: (including vertex at ∞)

$$(1+v) - e + f = 2$$

$$\Rightarrow v - e + n = 1$$

$$\Rightarrow e - v = n - 1 \quad (1)$$

- Since $h \geq 3$, $v = (2e-h)/3 \leq (2/3)\cdot e - 1$ (2)

$$e = (3v+h)/2 \geq 3(v+1)/2 \quad (3)$$

- (1)+(2) $\Rightarrow e - ((\frac{2}{3})e - 1) \leq n - 1$

$\Rightarrow$ $e \leq 3n - 6$ edges $\Rightarrow$ (iii)

- (1)+(3) $\Rightarrow 3(v+1)/2 - v \leq n - 1$

$\Rightarrow$ $v \leq 2n - 5$ vertices $\Rightarrow$ (ii)

□

Complexity grows rapidly with dimension:

> **Lemma:** In $\mathbb{R}^d$, worst-case combinatorial complexity of Voronoi diagram of n points is $O(n^{\lceil d/2 \rceil})$

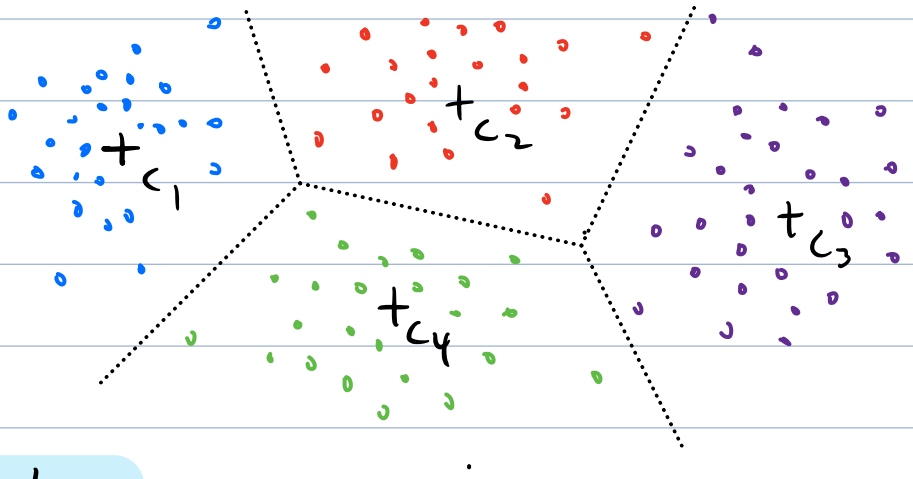
Applications:

Nearest-neighbor search in $\mathbb{R}^2$ - Given n point sites P, build data structure so that for any $q \in \mathbb{R}^2$, can find closest site in P to q.

Solution: ① Build $Vor(P)$ } $O(n)$ space

② Build point location structure } $O(\log n)$ time

Clustering: Given point set P + integer k, partion P into k clusters.

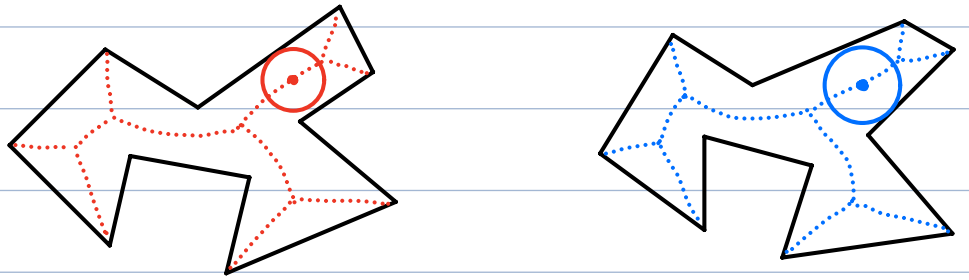


Solution:

- Identify k sites as cluster centers (e.g. using k-means or k-medians)
- Assign points to clusters based on the Voronoi cells.

Shape similarity:

- Use Voronoi diagram to reduce a shape to its skeleton
- Match shapes based on similarity of skeletons.

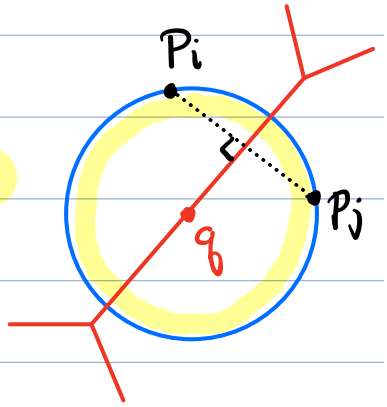


Variations:

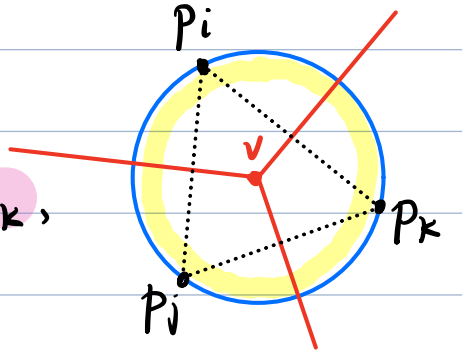
- Other metrics (e.g. L_1, L_∞)
- Weighted points: p_i has weight w_i
  - Multiplicative: $dist(q, p_i) = w_i \cdot \|p_i - q\|$
  - Additive: $dist(q, p_i) = \|p_i - q\| + w_i$
- k^{th} Nearest (special case: Farthest-Point Voronoi - subdivide based on which site is farthest)

Properties of Voronoi Diagrams:

- Empty-circle property:
  A point q is on an edge of the diagram iff there is a circle centered at q, that passes through 2 sites + is otherwise empty (of sites)

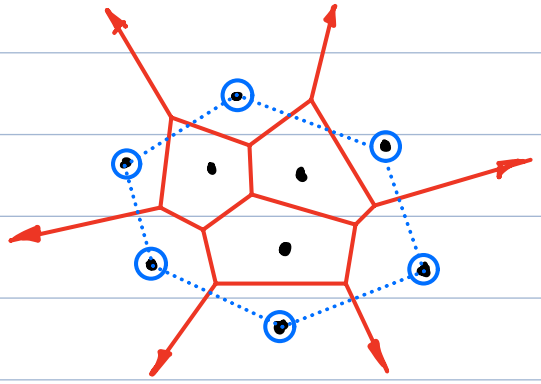


- Empty-circumcircle property:
  A point v is a Voronoi vertex iff there are 3 sites p_i, p_j, p_k s.t. v is the circumcenter of $\Delta p_i p_j p_k$, and the circumcircle is otherwise empty.



- Convex-Hull Property:

  A site p_i is on boundary of $conv(P)$ iff $V_P(p_i)$ is unbounded.



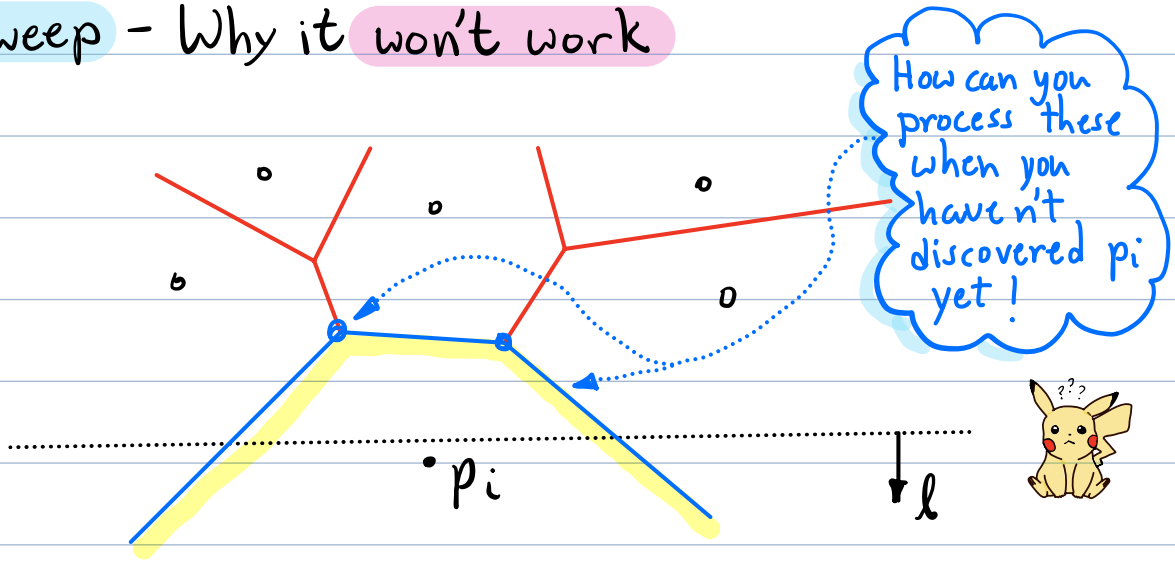
Constructing Voronoi Diagrams in $\mathbb{R}^2$

- Divide + Conquer
  - Partition points about a vertical line
  - Recursively construct left/right diagrams
  - Merge them - $O(n)$ time
  - $O(n \log n)$ total
- (Randomized) Incremental
  - Insert site, locate its cell, update
  - More efficient in dual (Delaunay triang.)
  - $O(n \log n)$ expected

Discussed in future lecture

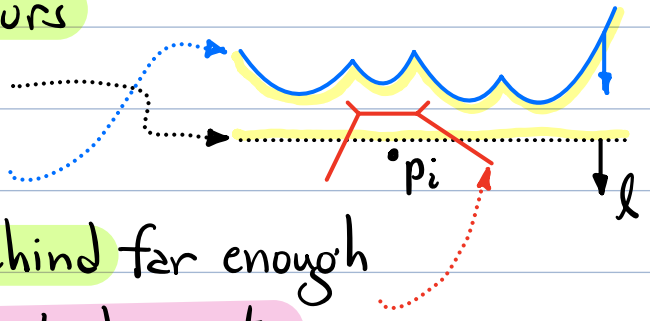
- Plane sweep
  - This lecture - Fortune's Algorithm
  - $O(n \log n)$

Plane Sweep - Why it won't work



Clever Trick -

- Sweep two contours
  - sweep line
  - beach line
- Beach line lags behind far enough to avoid unanticipated events



Notation:

- Given point q + set S:

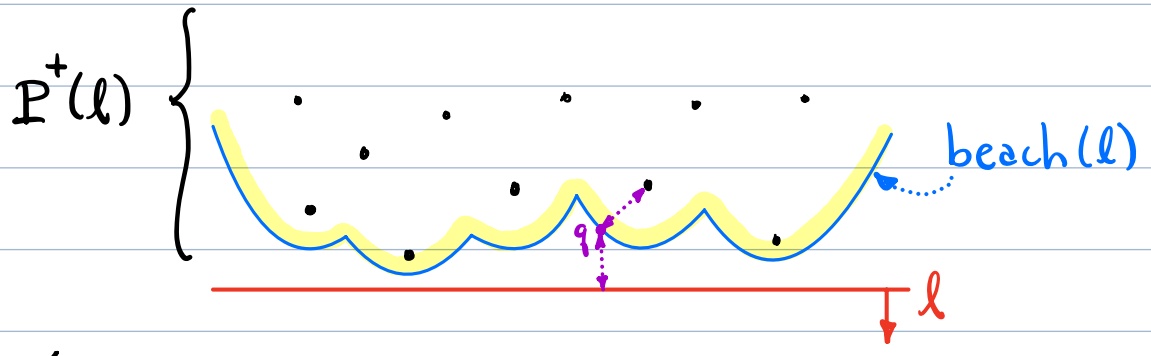
$$dist(q, S) = \min_{p \in S} \|p - q\|$$

- Given sites P + line ℓ:

$P^+(\ell)$ - subset of P on or above ℓ

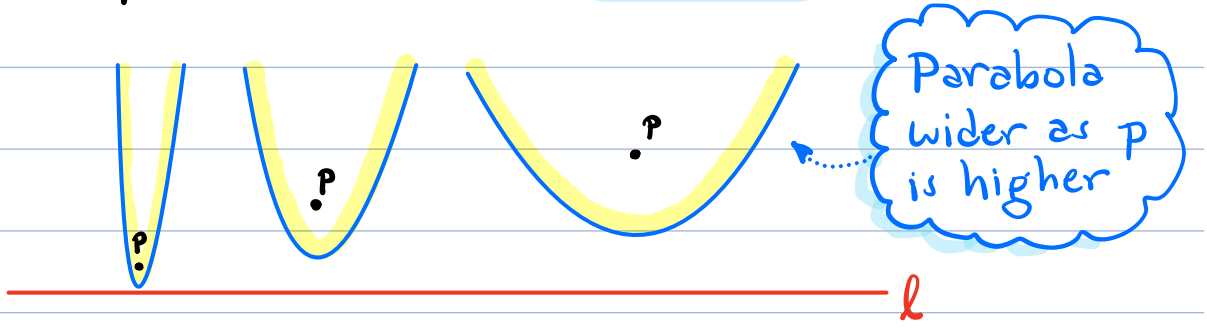
- Given sites P + horizontal line ℓ, the beach line is the set of points equidistant to $P^+(\ell)$ + ℓ:

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

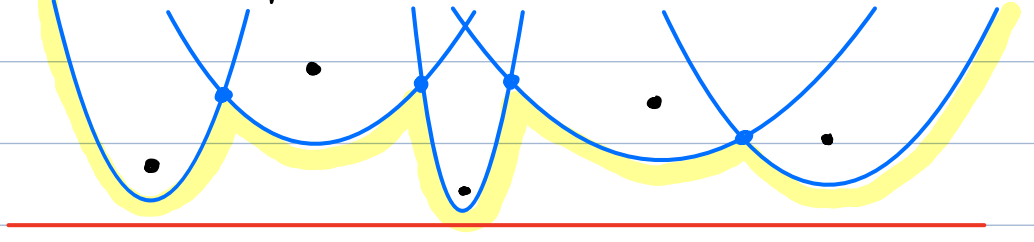


Recall (from basic geometry)

- The set of points equidistant to a point + line is a parabola.



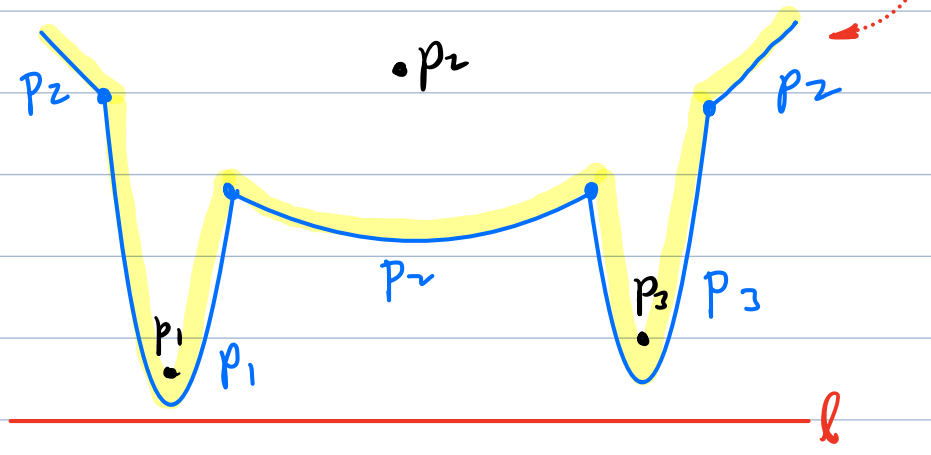
$\Rightarrow$ $\text{beach}_P(\ell)$ is the lower envelope of m parabolas, where $m = |P^+(\ell)|$



Claims:

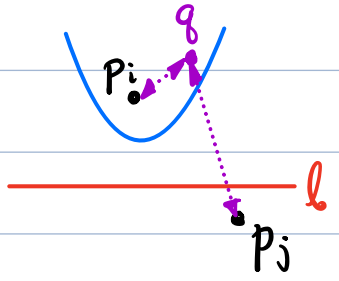
- Beach line is x-monotone (easy)
- Beach line consists of $\leq 2m-1$ "arcs" where $m = |P^+(\ell)|$

(Nontrivial, since a single site can can contribute up to m arcs)



Obs:

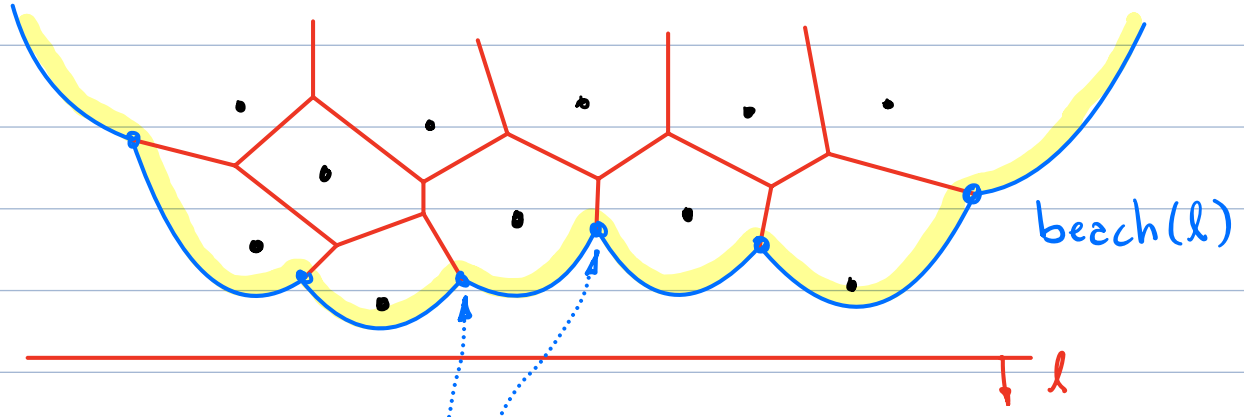
Any point q above the beach line is closer to some site above beach line than any site below ℓ.



$$\|p_i - q\| < \|p_j - q\|$$

$\Rightarrow$ The portion of $\text{Vor}(P)$ above the beach line is unaffected by any site below ℓ

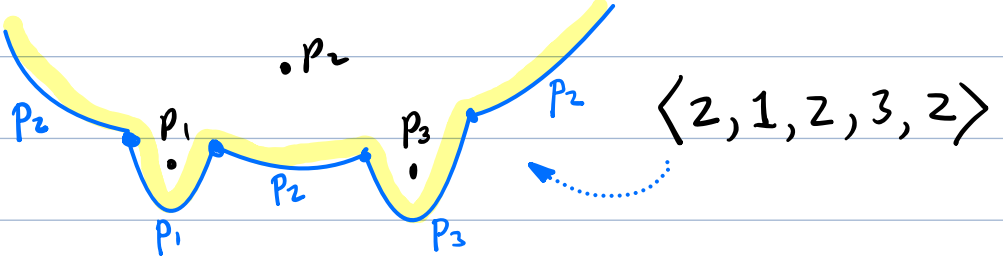
Example:



Endpoints of beachline arcs end called breakpoints
- Incident to Voronoi edges

Fortune's Algorithm - A plane-sweep approach

Sweep-line status:
- y-coord of current sweep line
- ordered dictionary storing indices of sites that contribute to current beach line (sorted left to right)



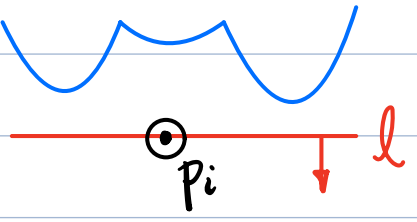
Note: - Parabolic arcs are never computed
- Breakpoints are computed only as needed

We also store the partial Voronoi diagram that lies above the beach line (say, as a DCEL)
Updated as new vertices/edges are discovered

Events:

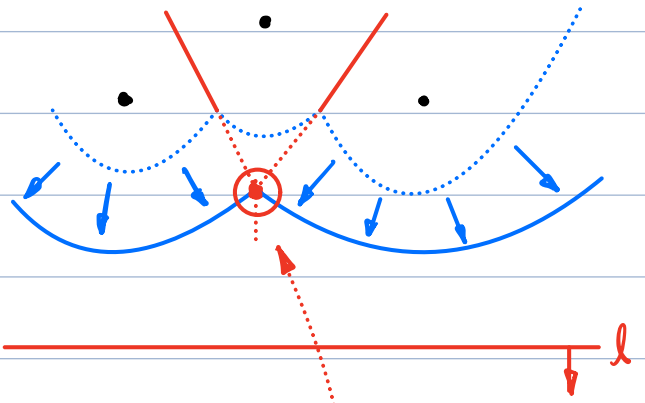
Site event:

- Whenever the sweep line encounters a new site.
(Known in advance)



Vertex event: (book calls this a circle event)

- When a new Voronoi vertex is discovered
≡ arc of beach line vanishes

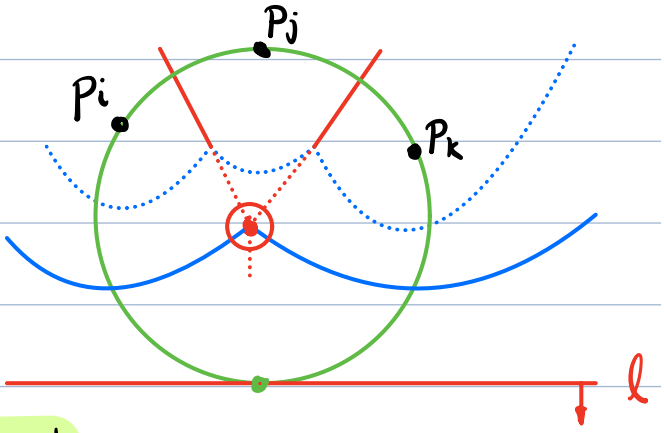


Priority Queue: (Sorted by y-coords)

- Site events → y-coord of site
- Vertex events → Trickier
(The vertex lags behind l)



Scheduling vertex events:

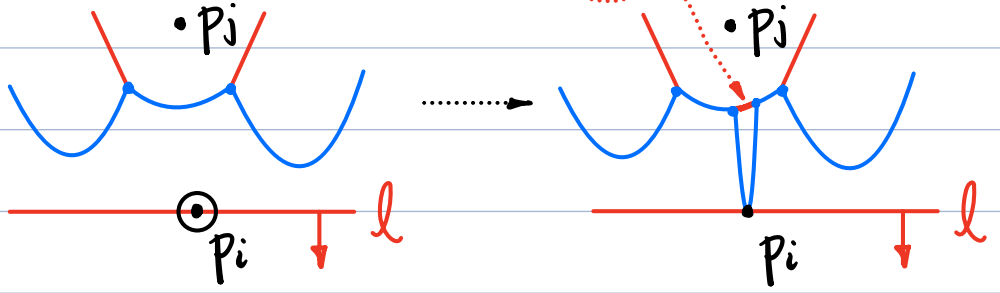


- For each consecutive triple $\langle \ldots i, j, k \ldots \rangle$ on beach line, schedule event at y-coord of horizontal tangent of circumcircle p_i, p_j, p_k (assuming all 3 distinct)

Event Processing:

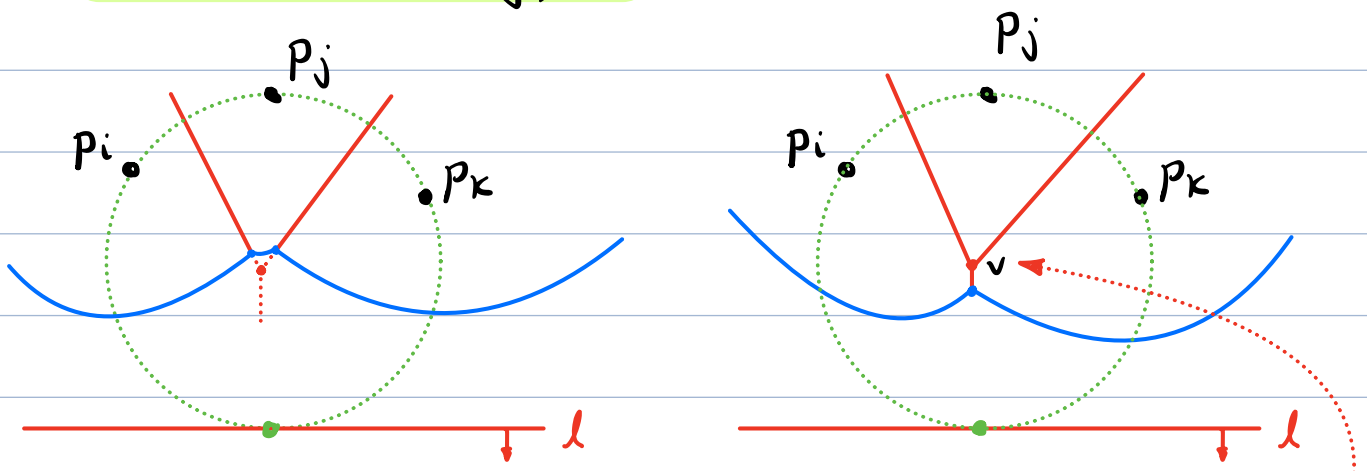
Site event (p_i):

- Search beach line for arc directly above p_i
  (How? Exercise – $O(\log n)$)
- Split this arc at this point
  $\langle \ldots j \ldots \rangle \rightarrow \langle \ldots j, i, j \ldots \rangle$
- Create a new Voronoi edge for $p_i p_j$ bisector



- Schedule vertex events for new triples:
  $\langle \ldots h, j, i, j, k \ldots \rangle \rightarrow$ vertex (h, j, i)
  vertex (i, j, k)

Vertex event (i, j, k):



- Delete middle arc (p_j) from beach line

  $\langle \ldots i, j, k \ldots \rangle \longrightarrow \langle \ldots i, k \ldots \rangle$

- Create new Voronoi vertex v at circumcenter of $p_i p_j p_k$
- Create new Voronoi edge at v for $p_i p_k$ bisector
- Schedule vertex events for new triples

  $\langle \ldots h, i, k, m \ldots \rangle \longrightarrow$ vertex (h, i, k), vertex (i, k, m)

Correctness: (Exercise)

- Check that all invariants are satisfied
- Check that all events are scheduled
- Check that beach line is properly updated

Analysis: $O(n)$ events (1 per site, 1 per Vor. vertex)

- Each involves $O(1)$ ops to dictionary + prior. queue

  $\Rightarrow O(\log n)$ per event

$\Rightarrow O(n \log n)$ total time



Summary:

- Voronoi diagram - Useful structure based on nearest neighbors
- Many variants:
  - Different objects
  - Different metrics
  - Weighted distances
- Computable in $O(n \log n)$ time in $\mathbb{R}^2$ by plane sweep — Fortune's algorithm
- Higher dimensions?
  - Well defined but high complexity $O(n^{\lceil d/2 \rceil})$