COSC 6114

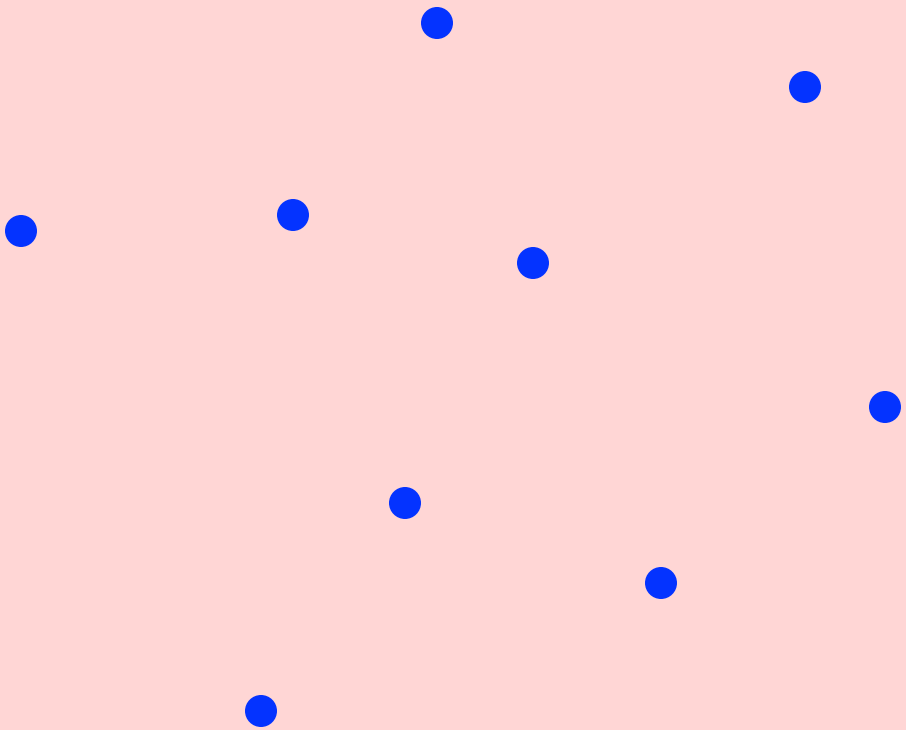
Prof. Andy Mirzaian

Voronoi Diagrams
&
Delaunay Triangulations

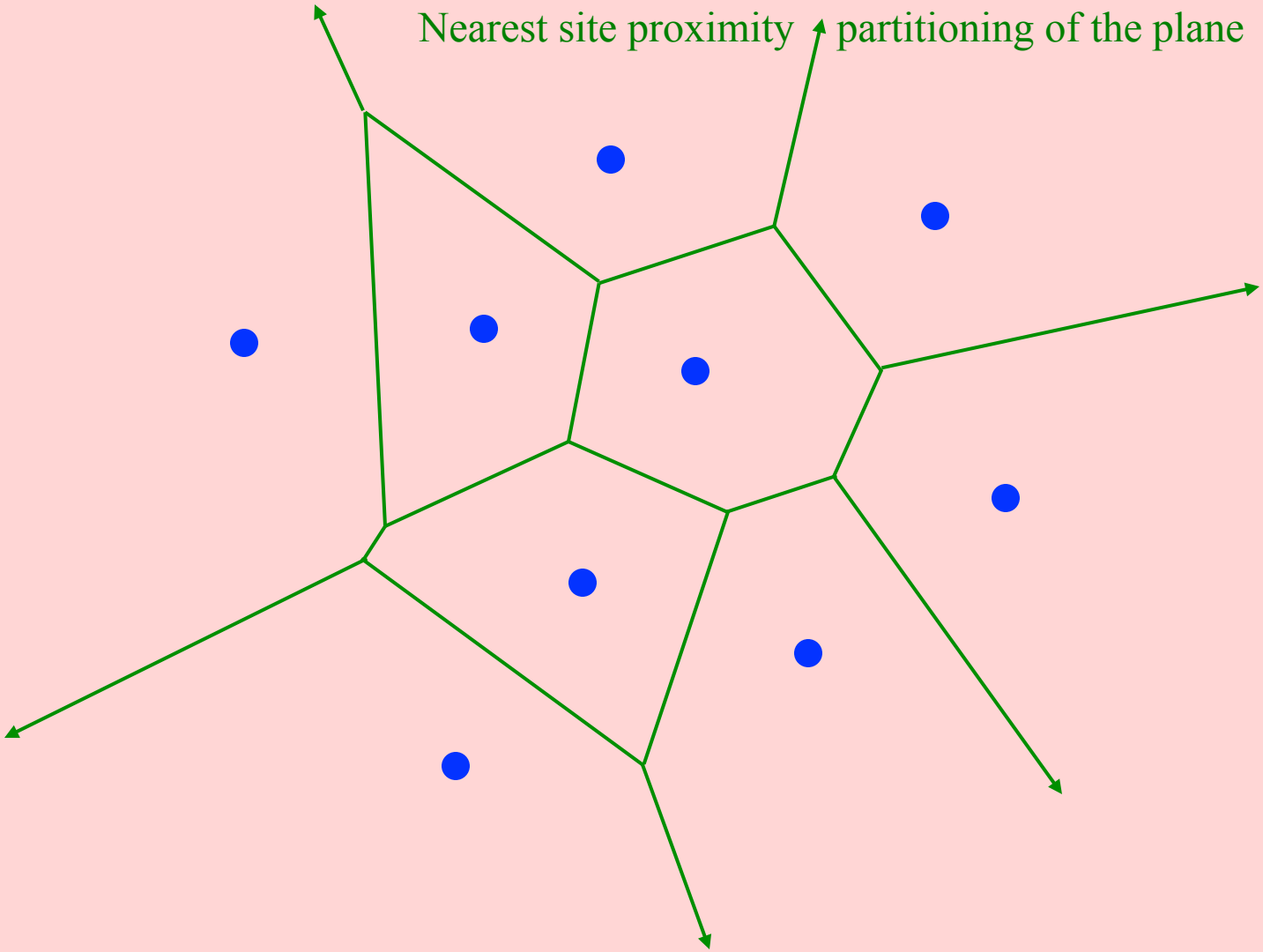
Introduction

Voronoi Diagram & Delaunay Triangulation

P = { p_1, p_2, … , p_n} a set of n points in the plane.



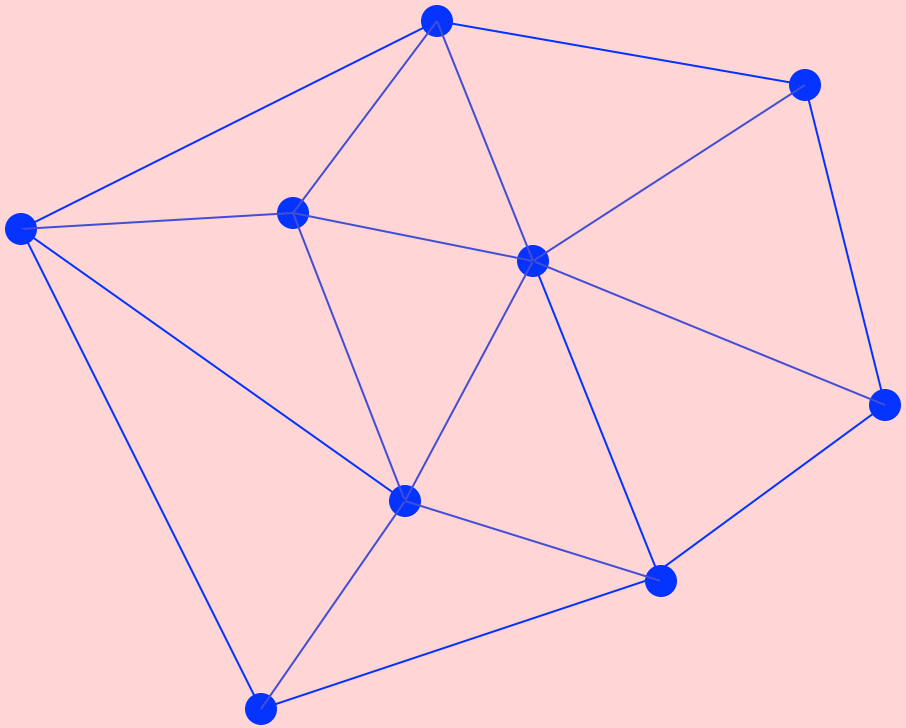
Voronoi Diagram & Delaunay Triangulation



Voronoi(P): # regions = n, # edges ≤ 3n-6, # vertices ≤ 2n-5.

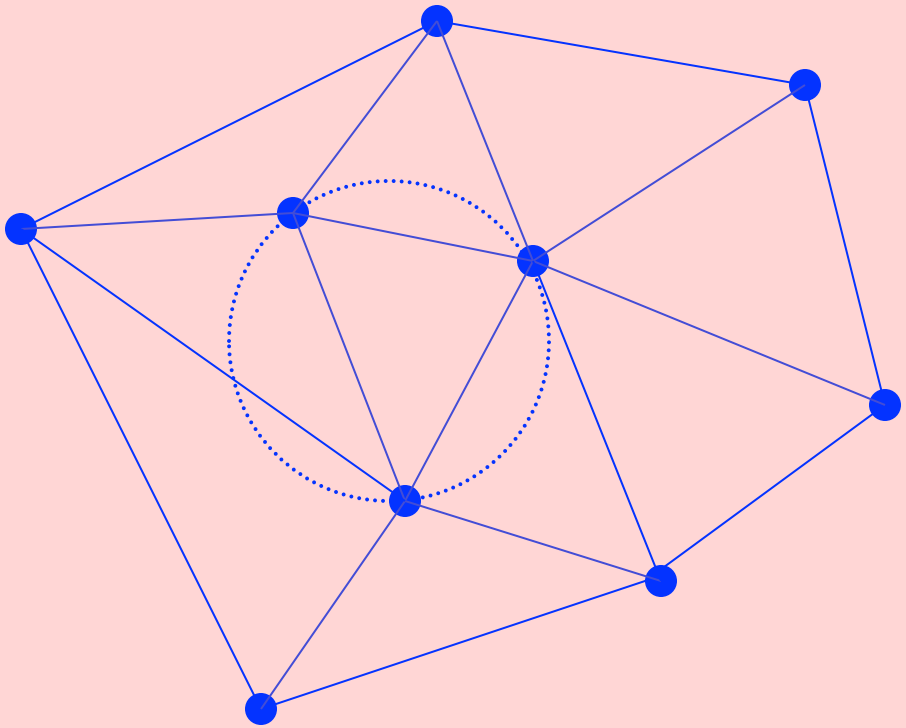
Voronoi Diagram & Delaunay Triangulation

Delaunay Triangulation = Dual of the Voronoi Diagram.



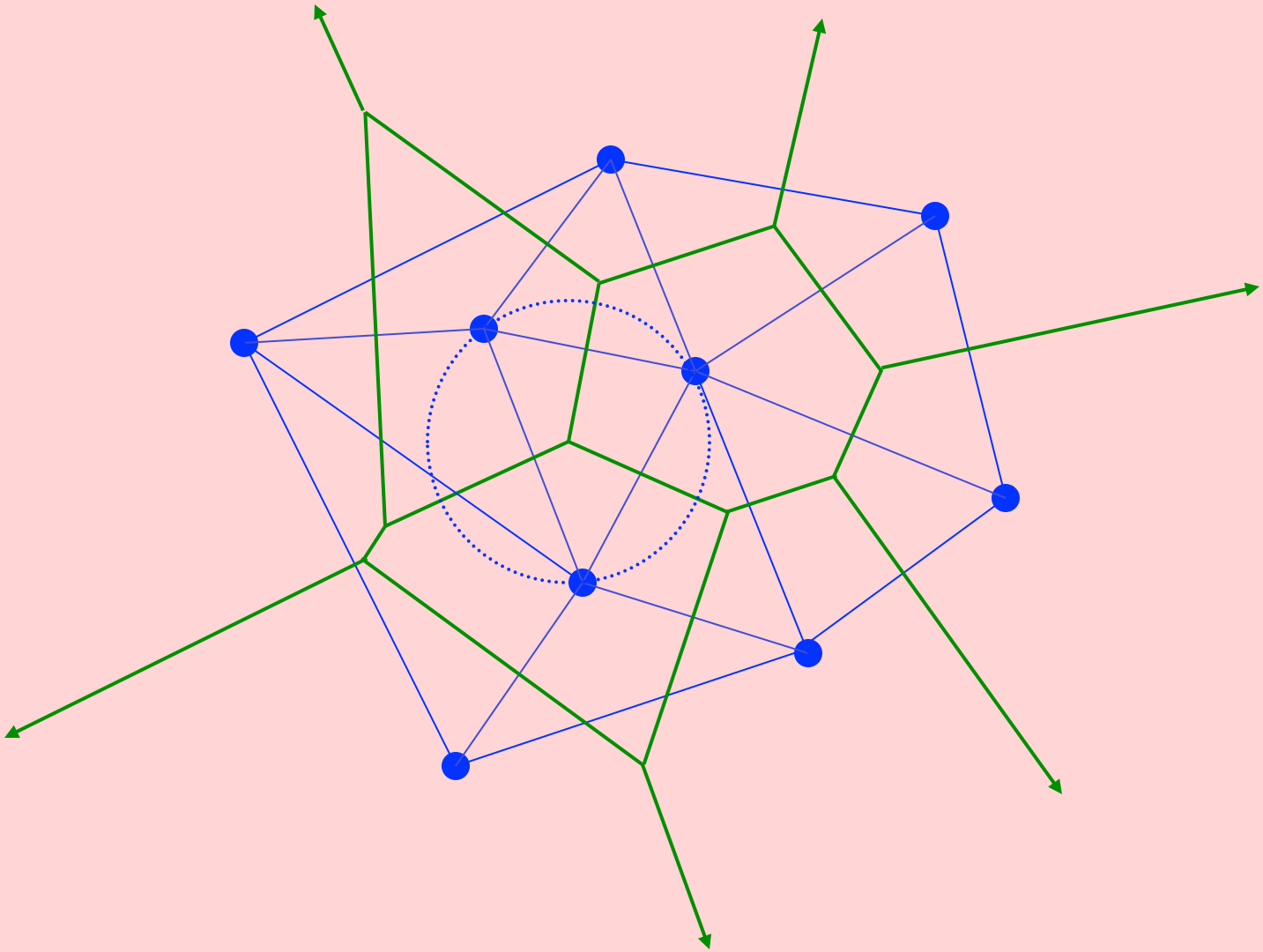
DT(P): # vertices = n, # edges ≤ 3n-6, # triangles ≤ 2n-5.

Voronoi Diagram & Delaunay Triangulation



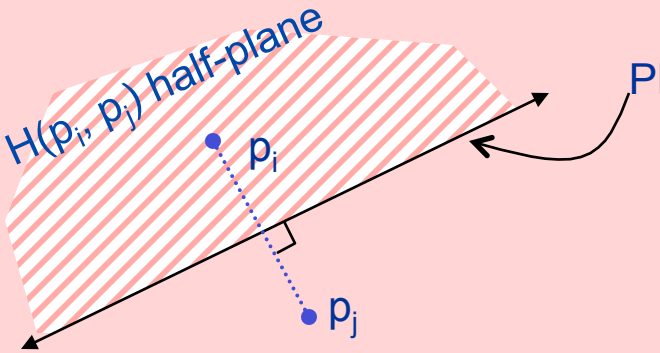
Delaunay triangles have the “empty circle” property.

Voronoi Diagram & Delaunay Triangulation

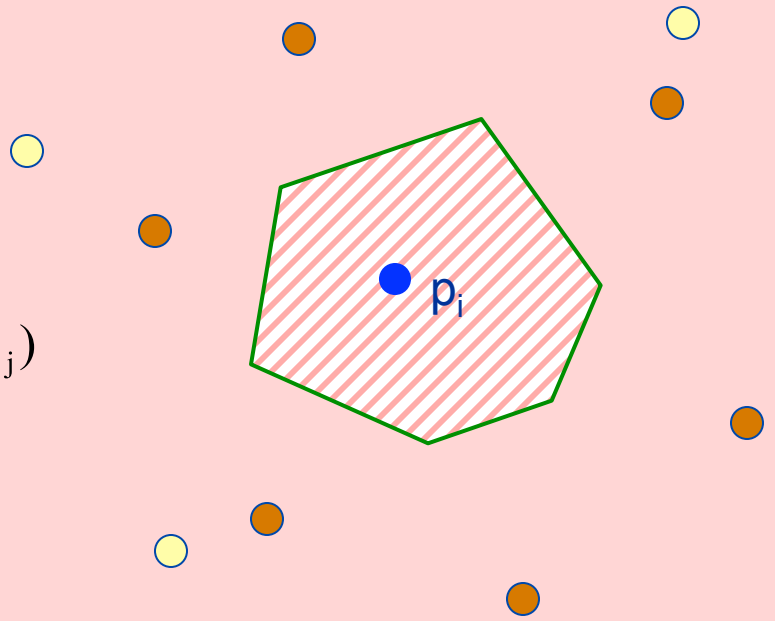


Voronoi Diagram

$P = \{ p_1, p_2, \ldots , p_n\}$ a set of n points in the plane.
Assume: no 3 points collinear, no 4 points cocircular.



$PB(p_i, p_j)$ perpendicular bisector of $\overline{p_i p_j}$.



Voronoi Region of p_i: $V(p_i) = \bigcap_{\substack{j=1 \\ j \neq i}}^{n} H(p_i, p_j)$

Voronoi Diagram of P: $VD(P) = \bigcup_{i=1}^{n} \{V(p_i)\}$

Voronoi Diagram Properties

- Each Voronoi region $V(p_i)$ is a convex polygon (possibly unbounded).
- $V(p_i)$ is unbounded $\Leftrightarrow$ p_i is on the boundary of CH(P).
- Consider a Voronoi vertex $v = V(p_i) \cap V(p_j) \cap V(p_k)$.
  Let C(v) = the circle centered at v passing through p_i, p_j, p_k.
- C(v) is circumcircle of Delaunay Triangle (p_i, p_j, p_k).
- C(v) is an empty circle, i.e., its interior contains no other sites of P.
- p_j = a nearest neighbor of p_i $\Rightarrow$ $V(p_i) \cap V(p_j)$ is a Voronoi edge
  $\Rightarrow$ (p_i, p_j) is a Delaunay edge.
- more later …

Delaunay Triangulation Properties

- DT(P) is straight-line dual of VD(P).
- DT(P) is a triangulation of P, i.e., each bounded face is a triangle (if P is in general position).
- (p_i, p_j) is a Delaunay edge $\Leftrightarrow \exists$ an empty circle passing through p_i and p_j.
- Each triangular face of DT(P) is dual of a Voronoi vertex of VD(P).
- Each edge of DT(P) corresponds to an edge of VD(P).
- Each node of DT(P), a site, corresponds to a Voronoi region of VD(P).
- Boundary of DT(P) is CH(P).
- Interior of each triangle in DT(P) is empty, i.e., contains no point of P.
- more later …

References:

- [M. de Berge et al ’00] chapters 7, 9, 13
- [Preparata-Shamos’85] chapters 5, 6
- [O’Rourke’98] chapter 5
- [Edelsbrunner’87] chapter 13
- AAW
- Lecture Notes 16, 17, 18, 19

ALGORITHMS

A brute-force VD Algorithm

P = { p_1, p_2, … , p_n} a set of n points in the plane.
Assume: no 3 points collinear, no 4 points cocircular.

Voronoi Region of p_i: $V(p_i) = \bigcap_{\substack{j=1 \\ j \neq i}}^{n} H(p_i, p_j)$

intersection of n-1 half-planes

Voronoi Diagram of P: $VD(P) = \bigcup_{i=1}^{n} \{V(p_i)\}$

- Voronoi region of each site can be computed in O(n log n) time.
- There are n such Voronoi regions to compute.
- Total time $O(n^2 \log n)$.

Divide-&-Conquer Algorithm

- **M. I. Shamos, D. Hoey [1975],**
  "Closest Point Problems," FOCS, 208-215.

- D.T. Lee [1978], "Proximity and reachability in the plane,"
  Tech Report No, 831, Coordinated Sci. Lab., Univ. of Illinois at Urbana.

- D.T. Lee [1980], *"Two dimensional Voronoi Diagram in the L_p metric,"*
  JACM 27, 604-618.

The first O(n log n) time algorithm to construct the Voronoi Diagram of n point sites in the plane.

ALGORITHM Construct Voronoi Diagram (P)

INPUT: P = { p_1, p_2, … , p_n} sorted on x-axis.

OUTPUT: CH(P) and DCEL of VD(P).

O(1) 1. [BASIS]: **if** $n \le 1$ **then return** the obvious answer.

2. [DIVIDE]: Let $m \leftarrow \lfloor n/2 \rfloor$

O(n) Split P on the median x-coordinate into

L = { p_1, … , p_m} & R = { p_{m+1}, … , p_n}.

3. [RECUR]:

T(n/2) (a) Recursively compute CH(L) and VD(L).

T(n/2) (b) Recursively compute CH(R) and VD(R).

4. [MERGE]:

(a) Compute Upper & Lower Bridges of CH(L) and CH(R) & obtain CH(P).

(b) Compute the y-monotone dividing chain C between VD(L) & VD(R).

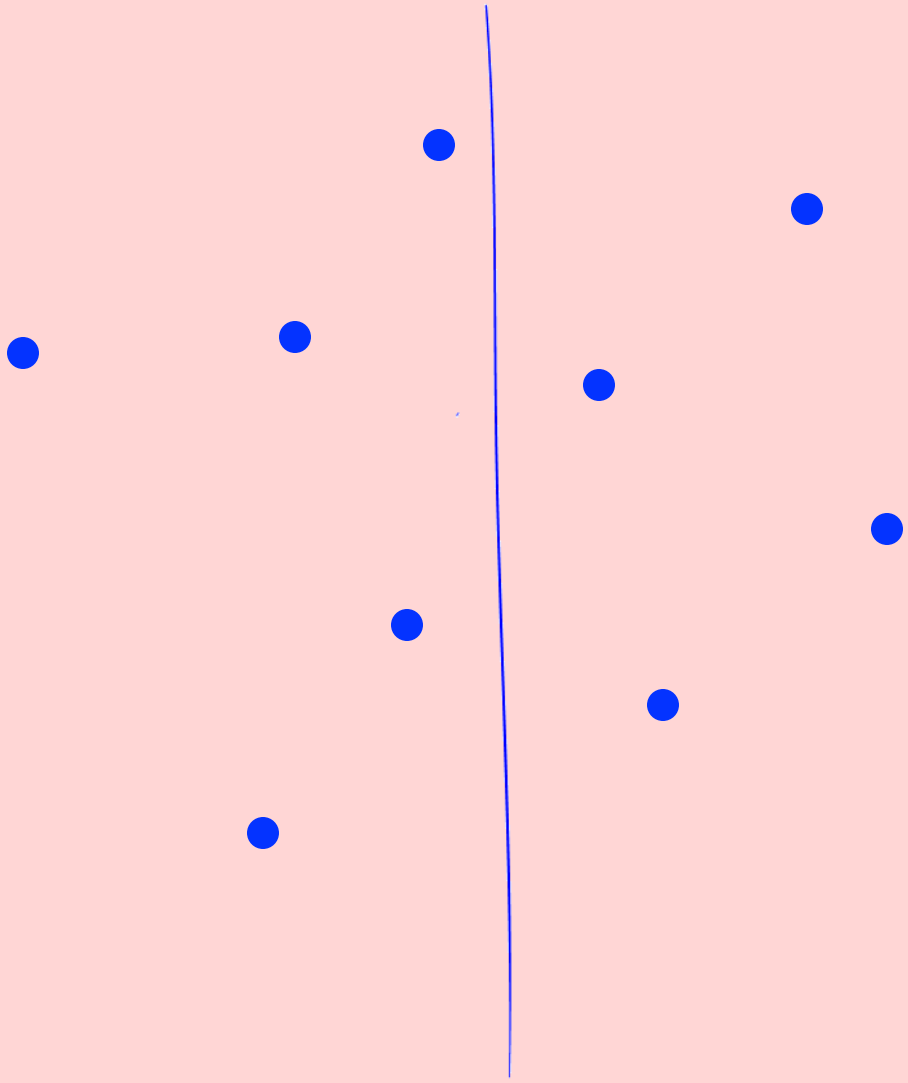
O(n) (c) VD(P) ← [C] ∪ [VD(L) to the left of C] ∪ [VD(R) to the right of C].

(d) **return** CH(P) & VD(P).

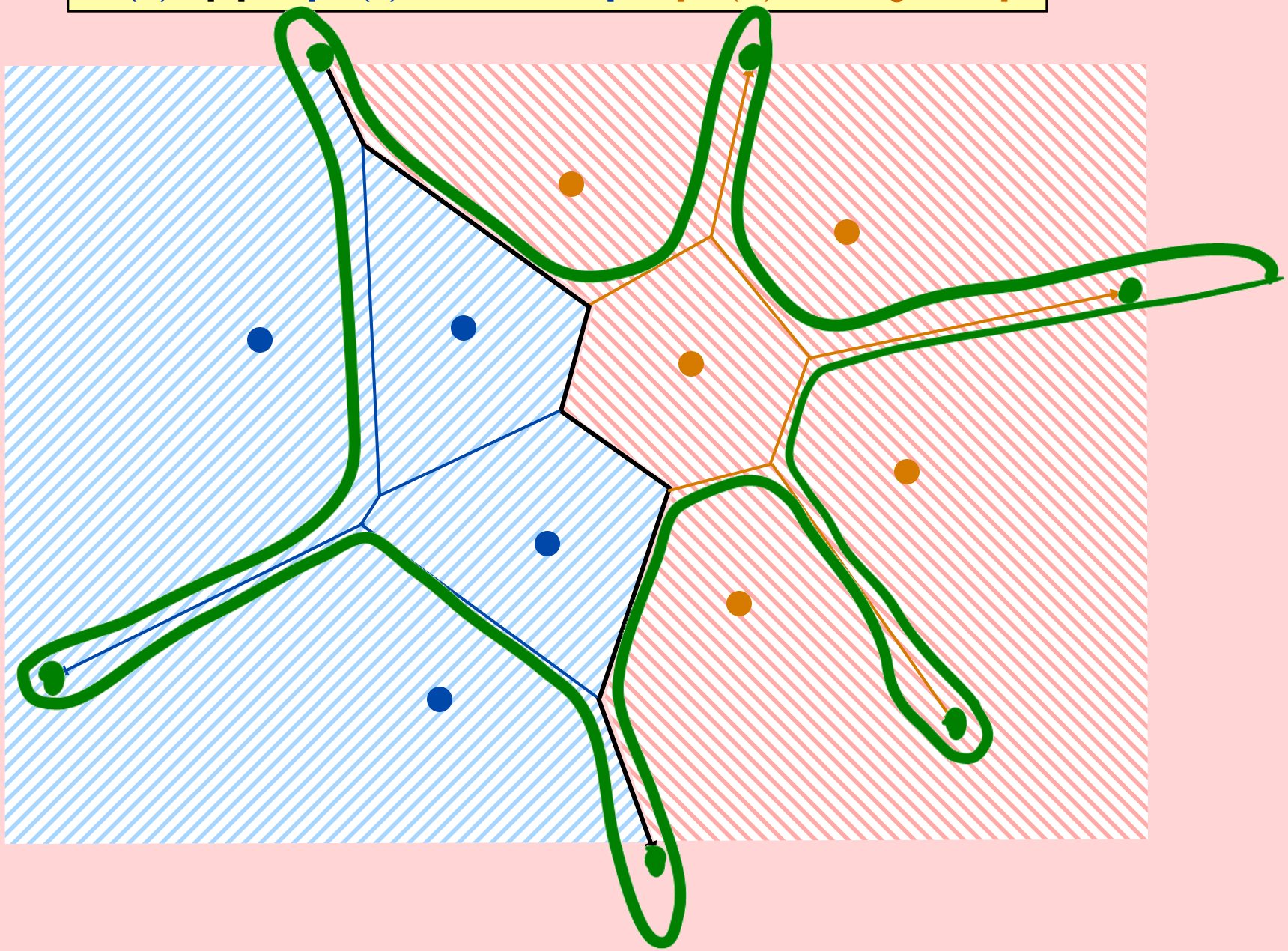
END.

$T(n) = 2\,T(n/2) + O(n) = O(n \log n)$.

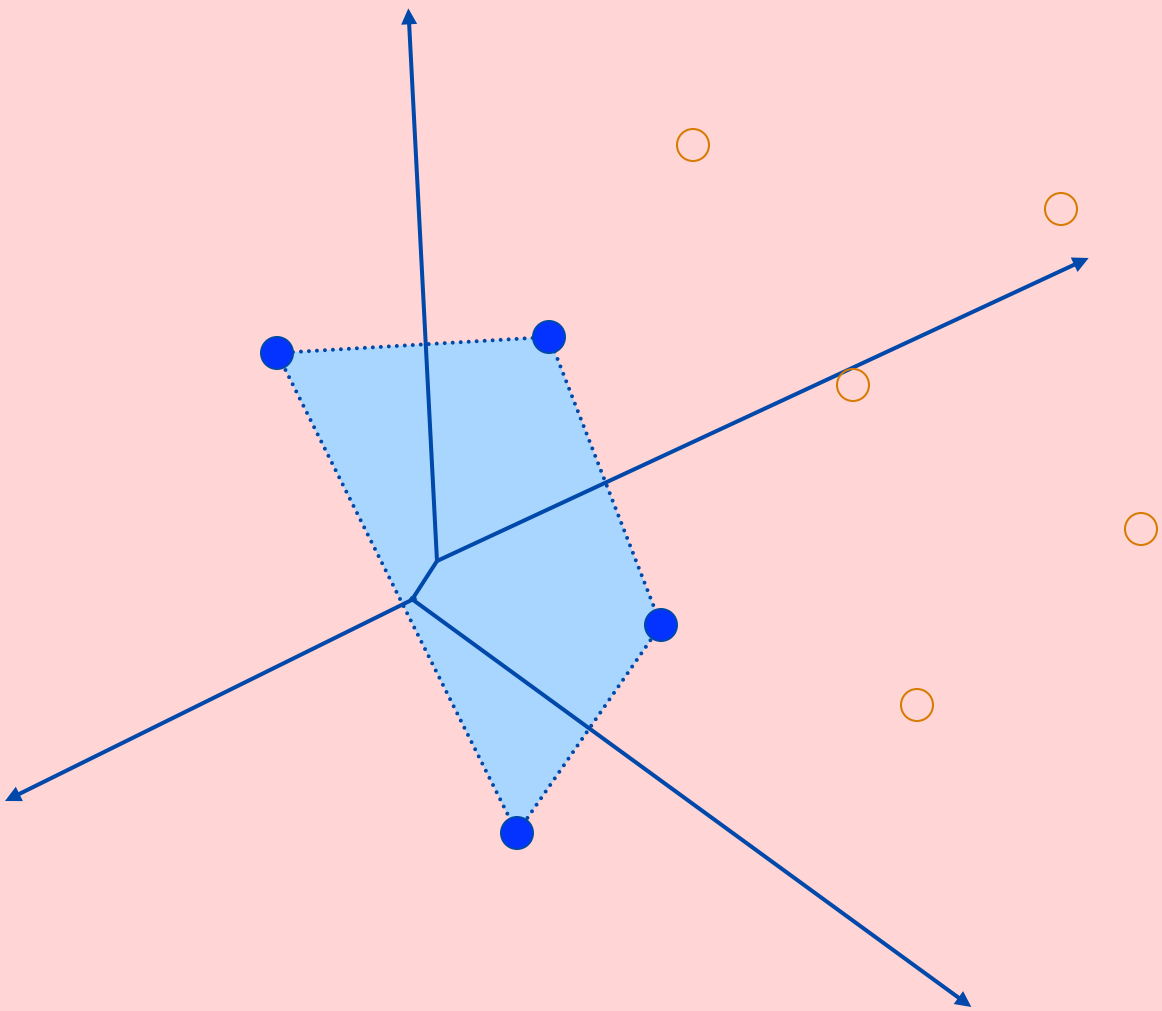
$P = \{ p_1, p_2, \ldots , p_n\}$ a set of n points in the plane.



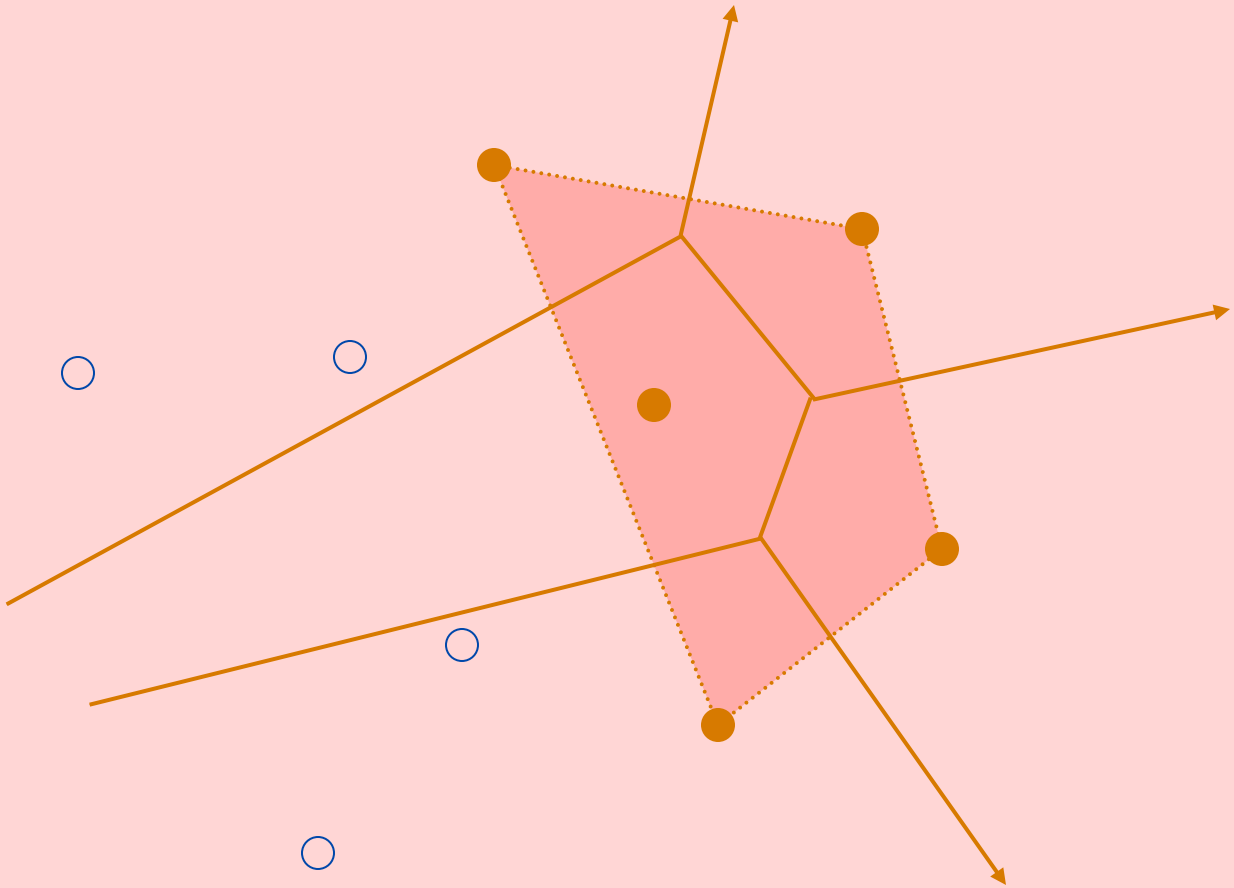
VD(P) = [C] ∪ [VD(L) to the left of C] ∪ [VD(R) to the right of C] .



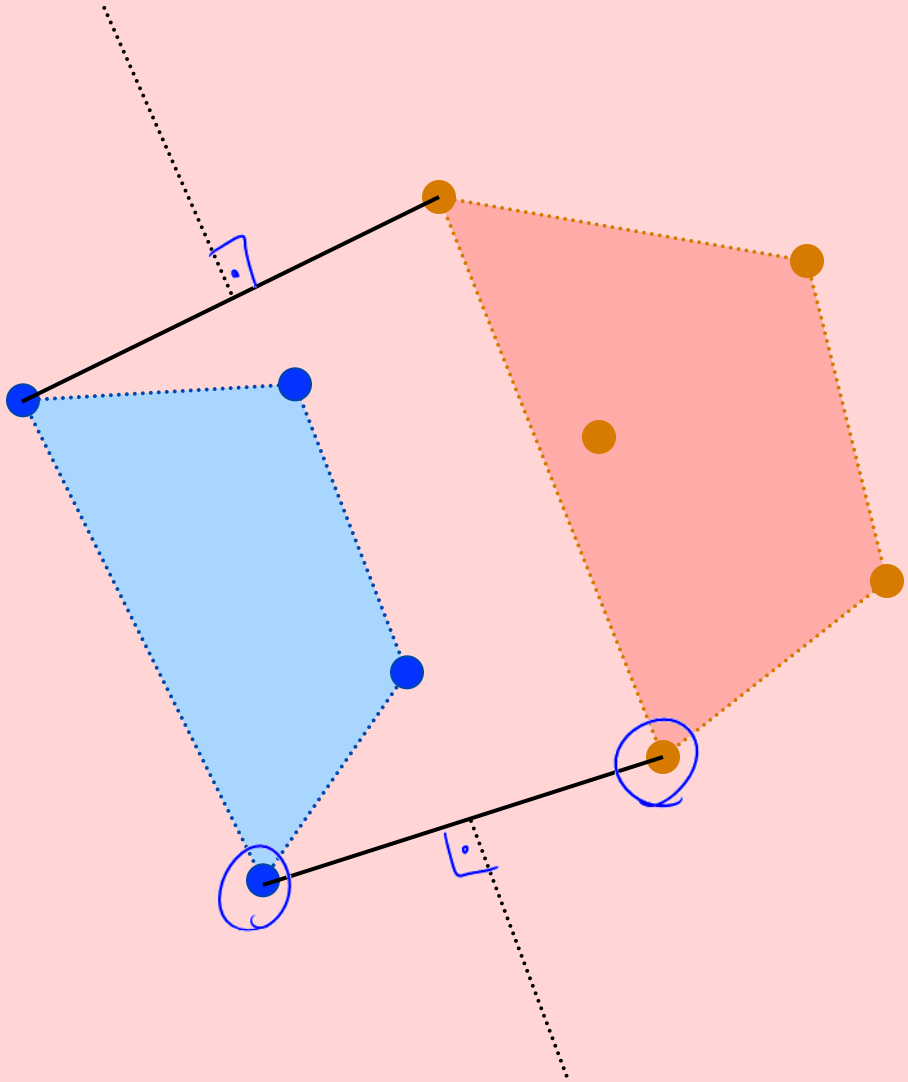
VD(L) and CH(L)

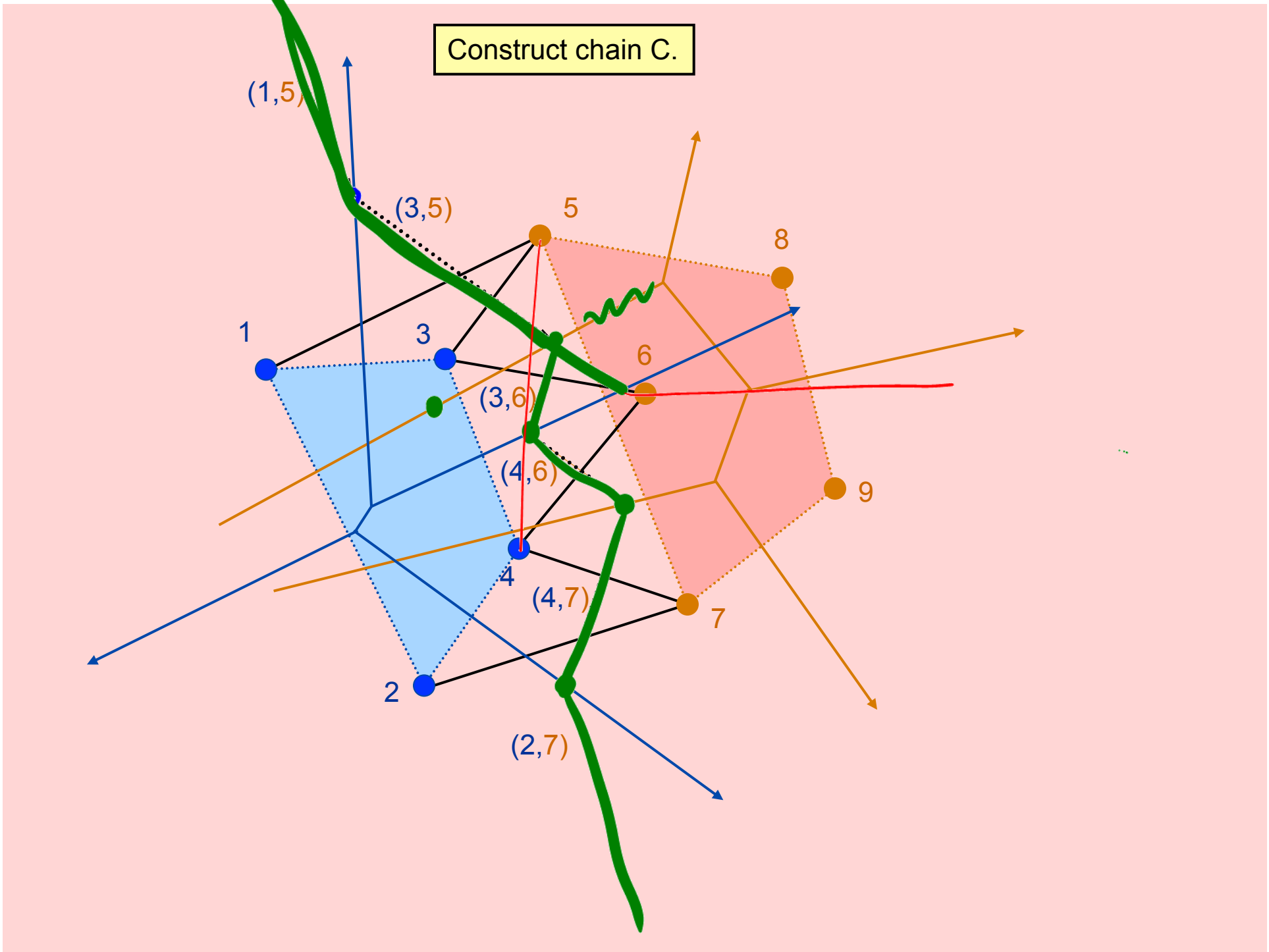


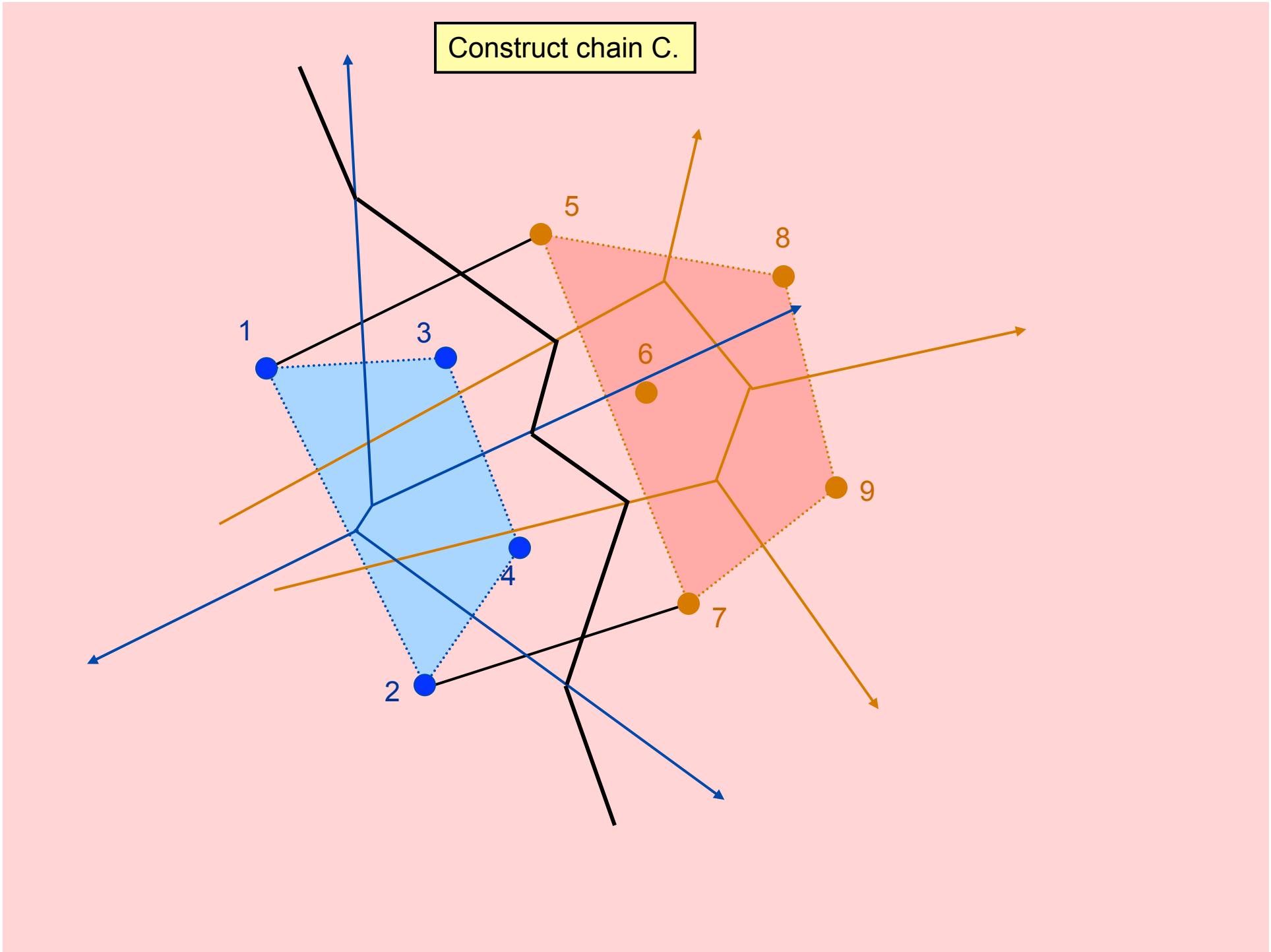
VD(R) and CH(R)

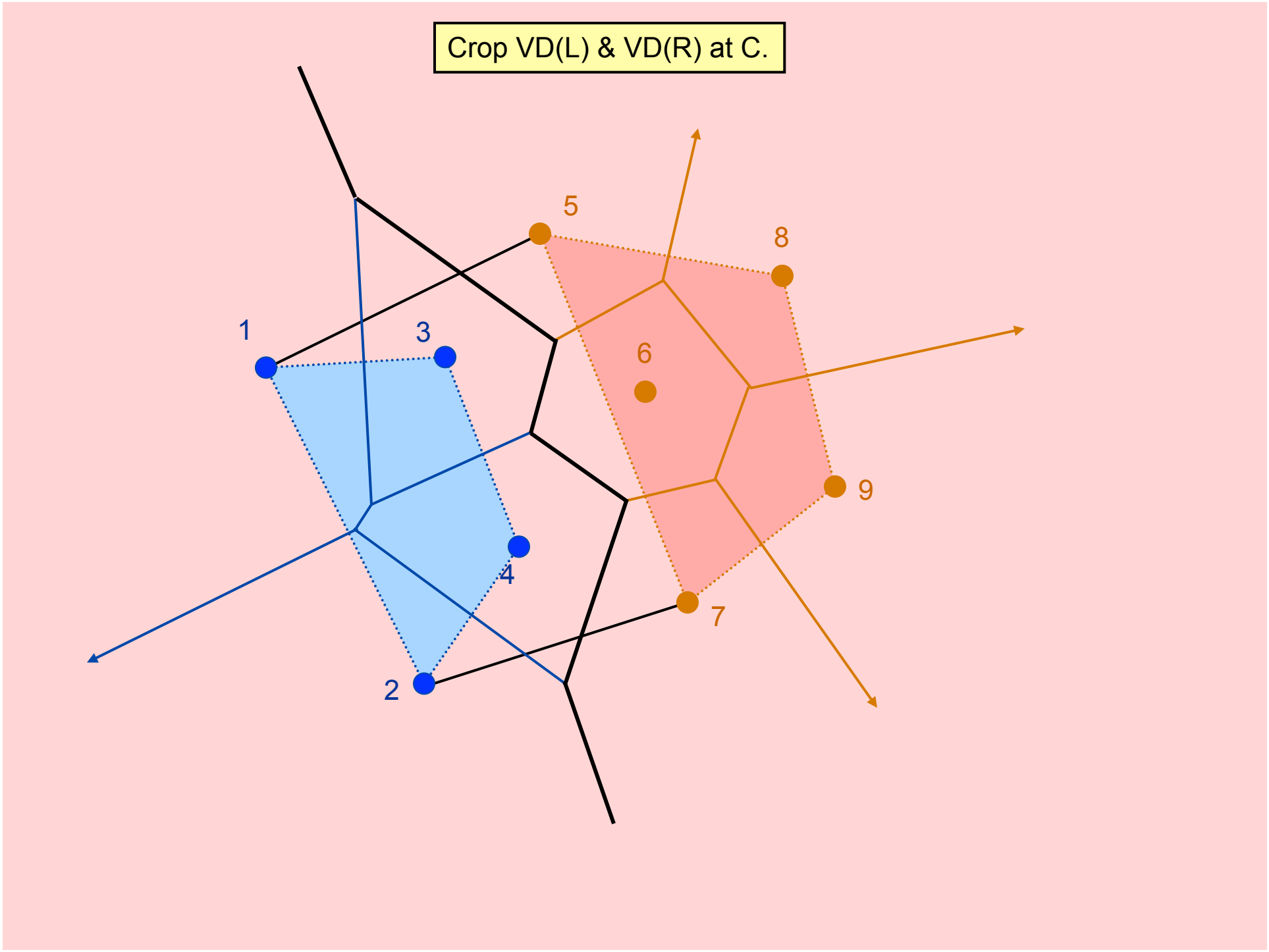


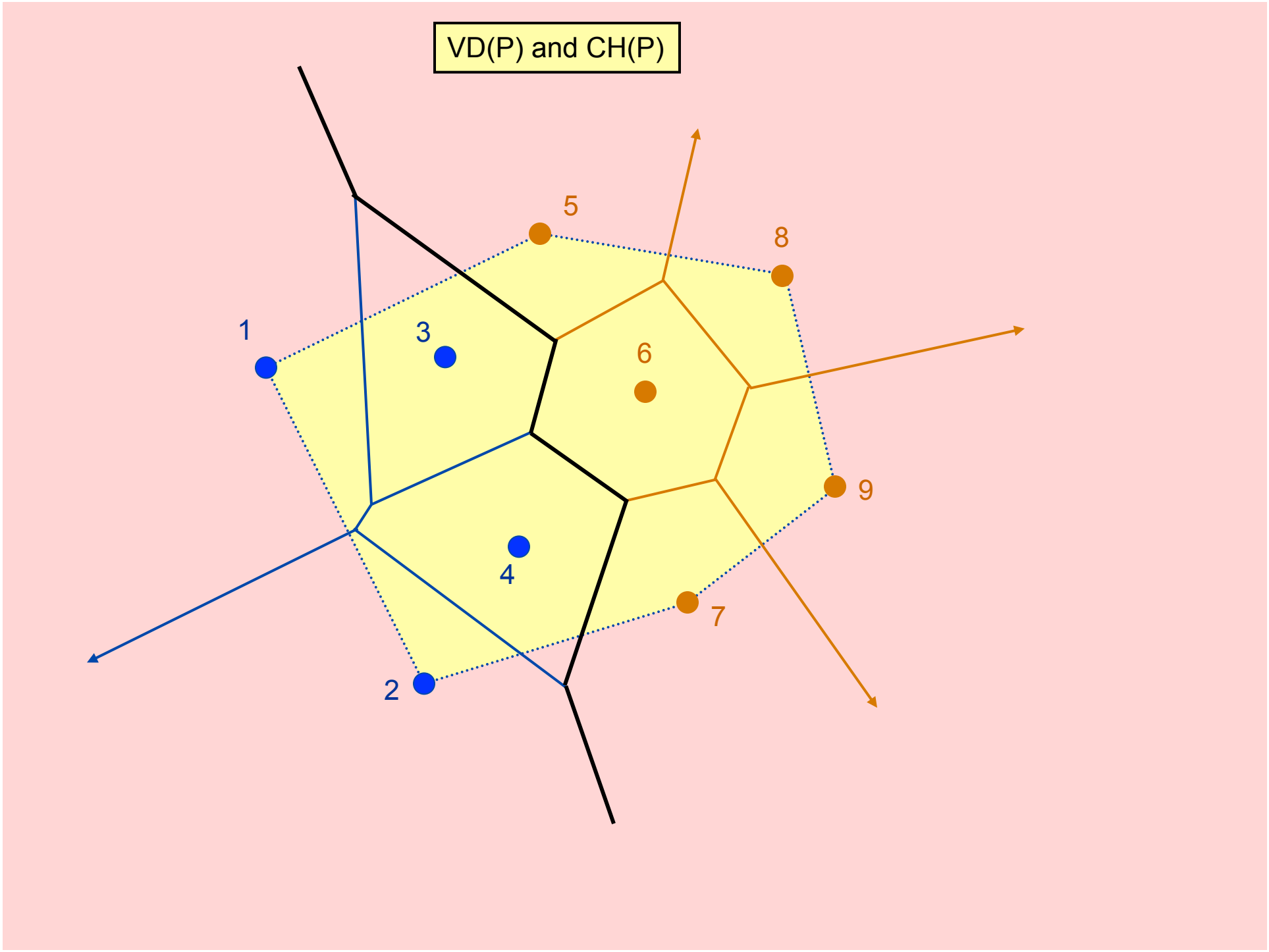
Upper & Lower bridges between CH(L) and CH(R) & two end-rays of chain C.




Construct chain C.
(1,5)
(3,5)
5
8
1
3
6
(3,6)
(4,6)
9
4
(4,7)
7
2
(2,7)


Construct chain C.
1
2
3
4
5
6
7
8
9


Crop VD(L) & VD(R) at C.
1
2
3
4
5
6
7
8
9


VD(P) and CH(P)
1
2
3
4
5
6
7
8
9

Fortune's Algorithm

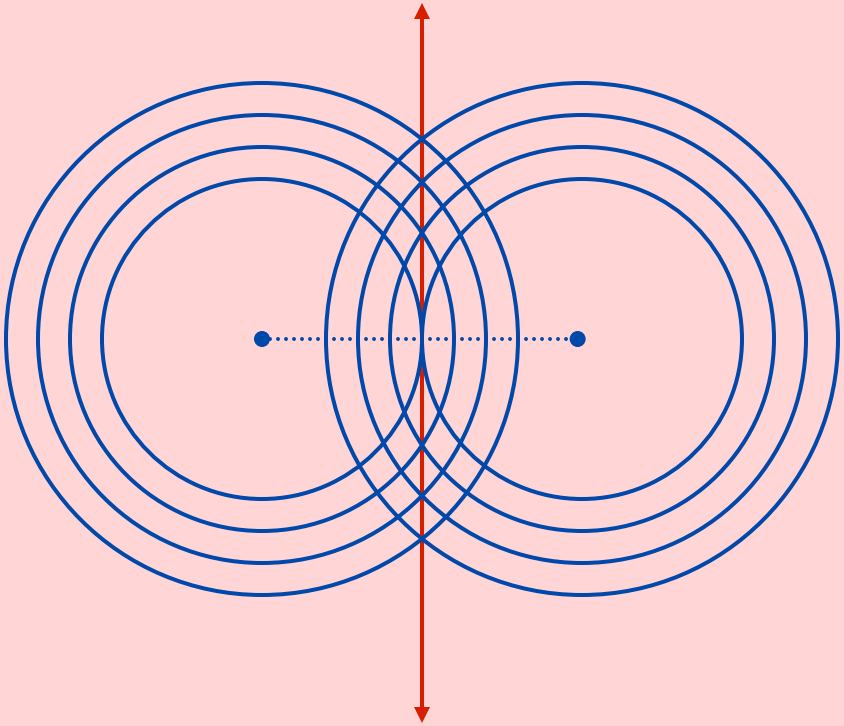
• **Steve Fortune [1987], "A Sweepline algorithm for Voronoi Diagrams," Algorithmica, 153-174.**

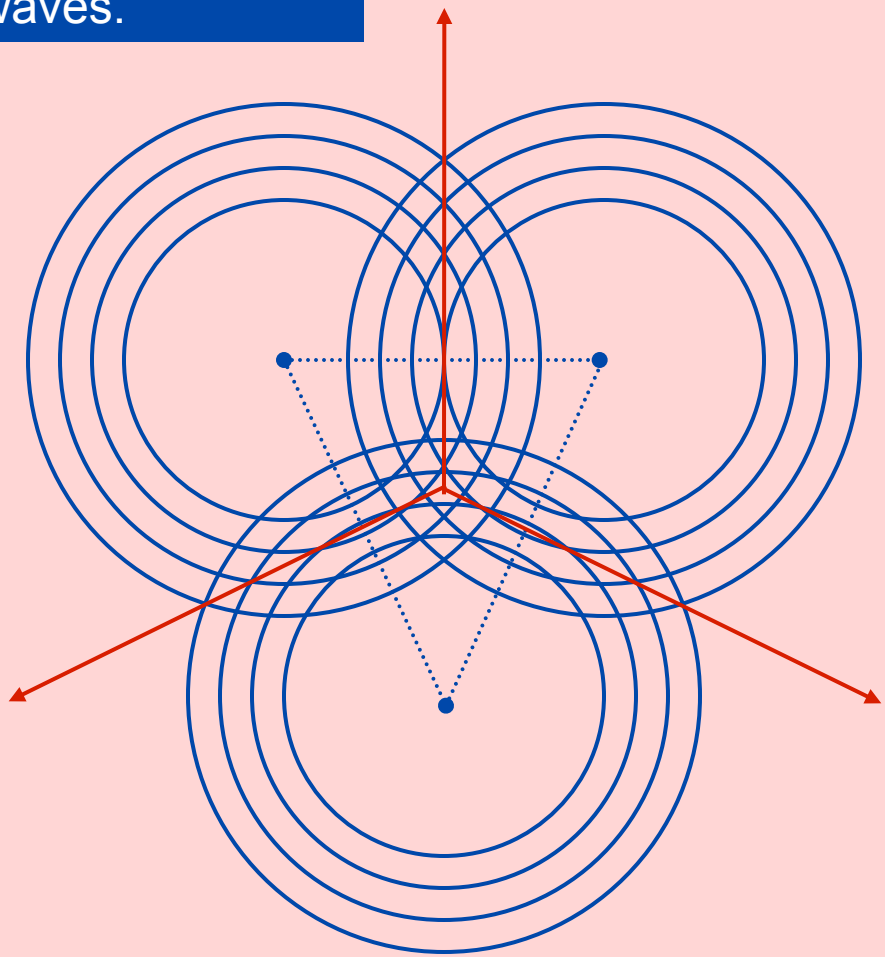
• Guibas, Stolfi [1987],
"Ruler, Compass and computer: The design and analysis of geometric algorithms,"
Proc. of the NATO Advanced Science Institute, series F, vol. 40: Theoretical Foundations of Computer Graphics and CAD, 111-165.

- O(n log n) time algorithm by plane-sweep.
- See AAW animation.
- Generalization: VD of line-segments and circles.

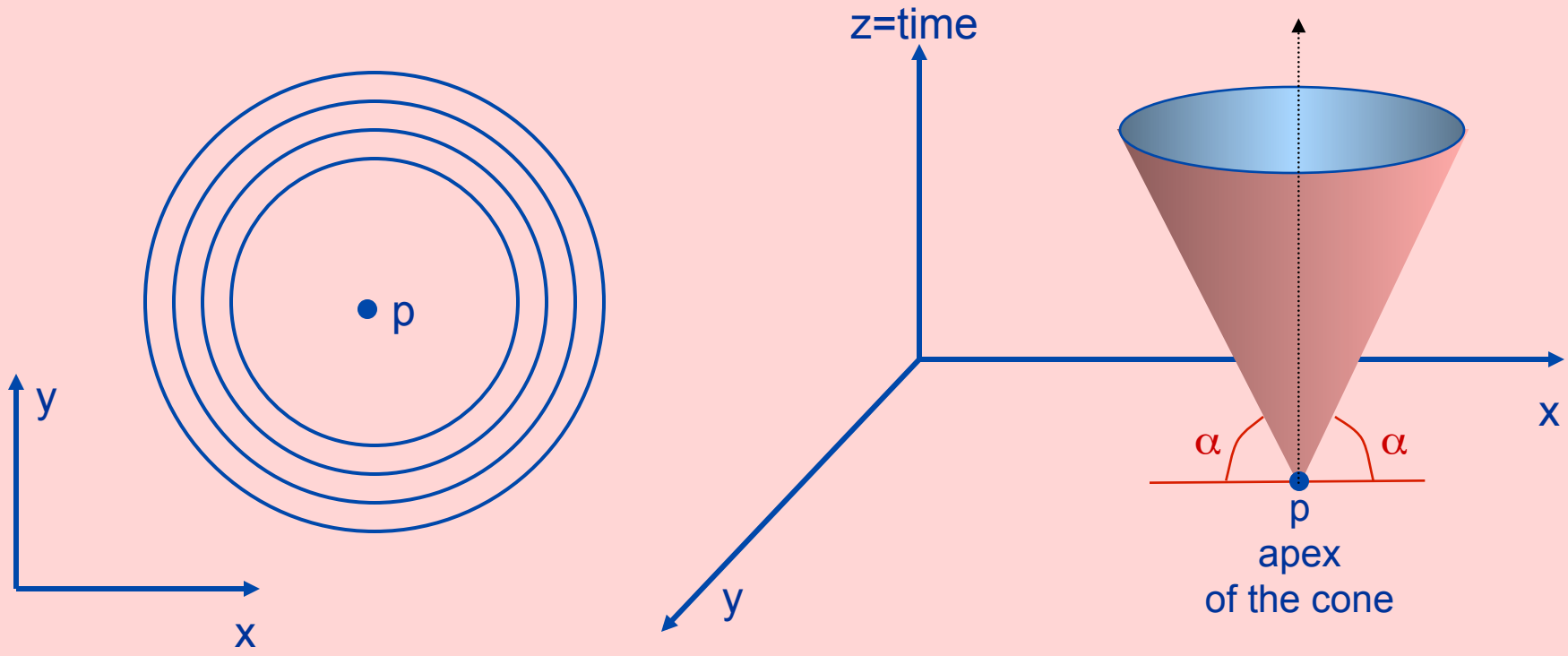
The Waive Propagation View

- Simultaneously drop pebbles on calm lake at n sites.
- Watch the intersection of expanding waves.



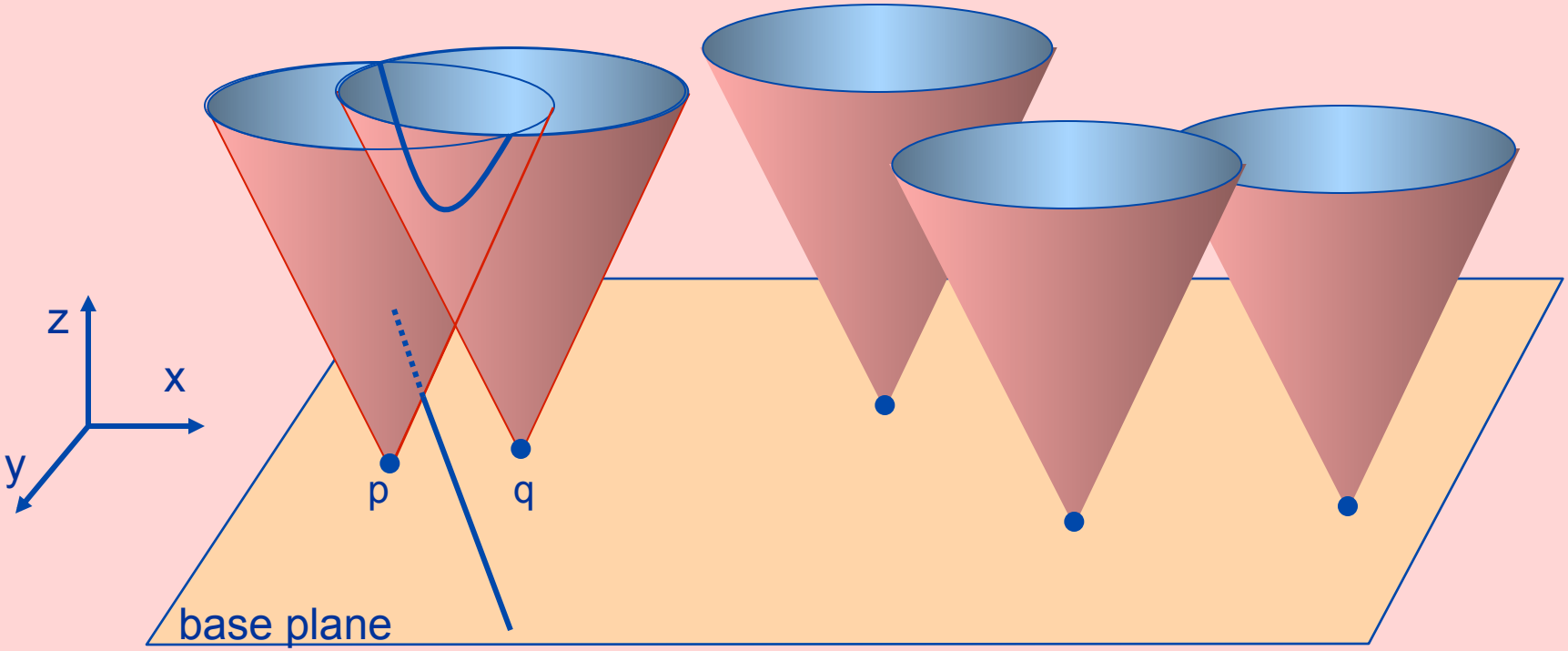


Time as 3rd dimension



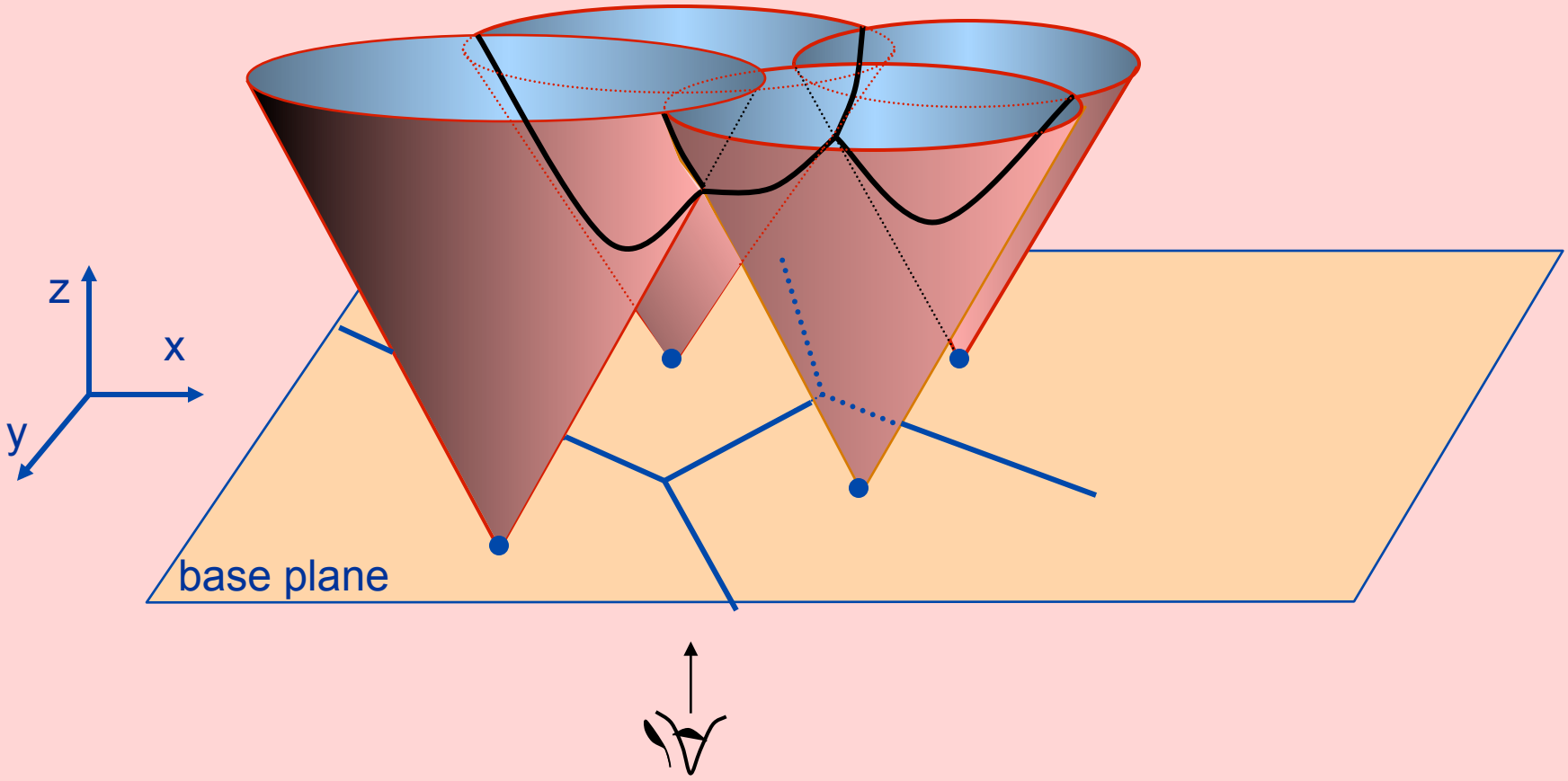
All sites have identical opaque cones.

Time as 3rd dimension



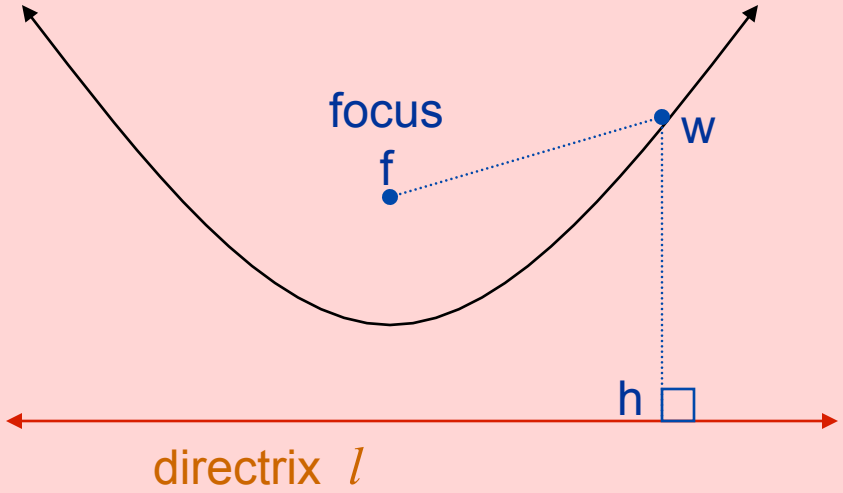
- All sites have identical opaque cones.
- cone(p) ∩ cone(q) = vertical hyperbola h(p,q).
- Vertical projection of h(p,q) on the xy base plane is PB(p,q).

Time as 3rd dimension



Visible intersection of the cones viewed upward from z = - ∞ is VD(P).

Conic Sections: Focus-Directrix

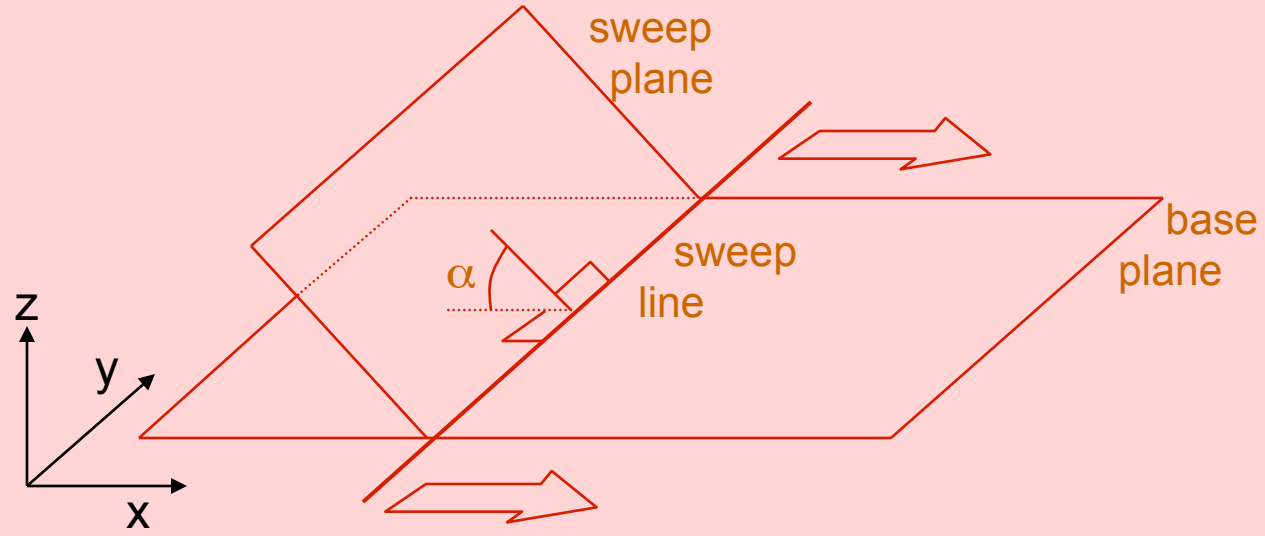


Eccentricity constant:

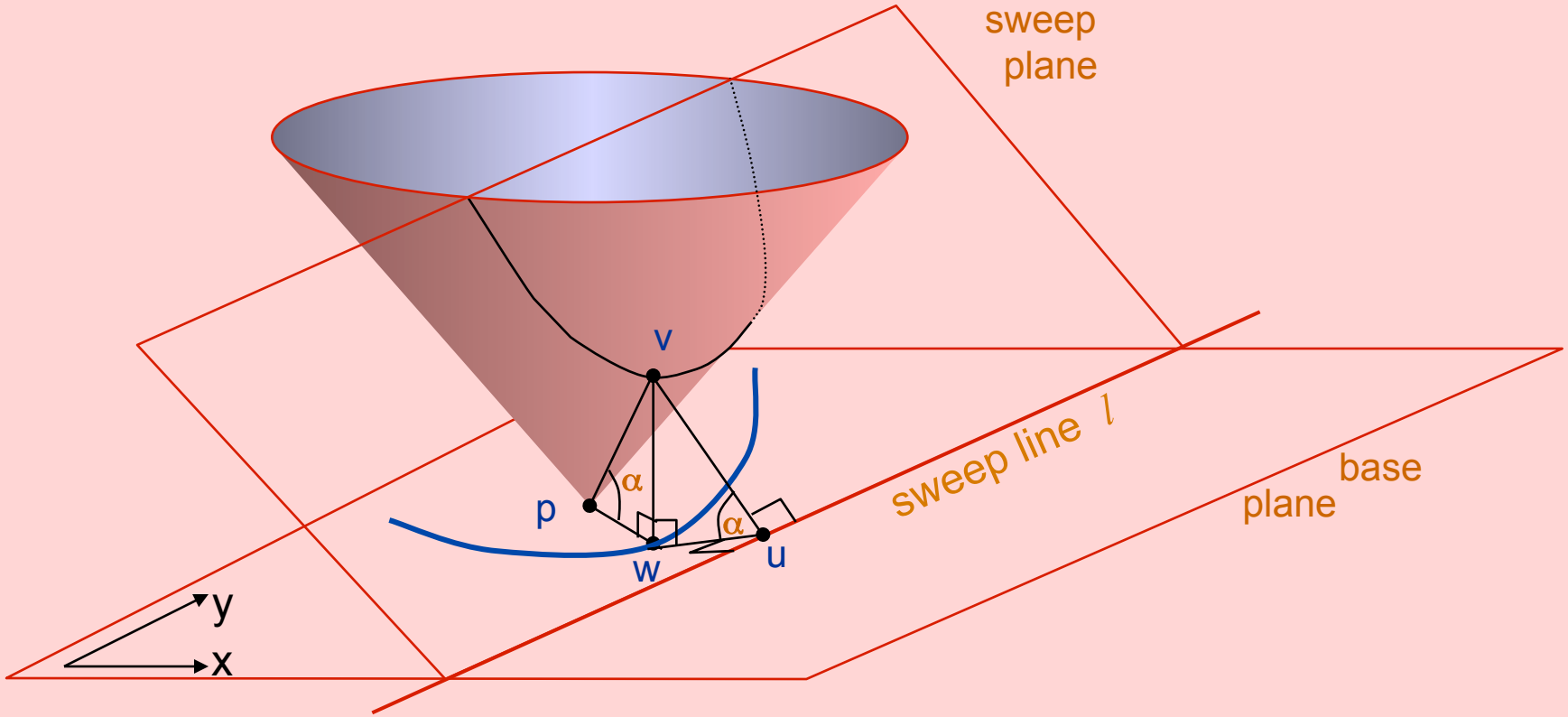
$$e = \frac{\overline{fw}}{\overline{hw}}$$

0 = e	point (focus)
0 < e < 1	ellipse
e = 1	parabola
e > 1	hyperbola

Sweep Plane & Sweep Line

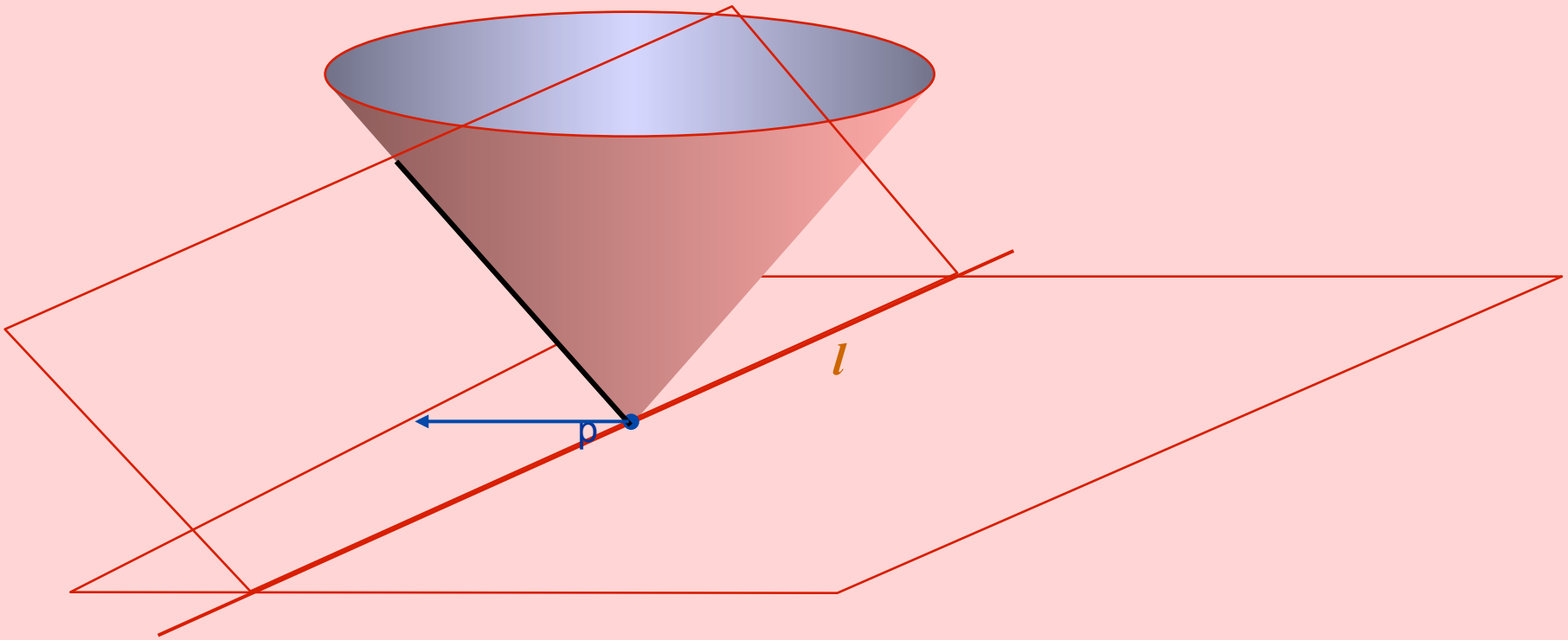


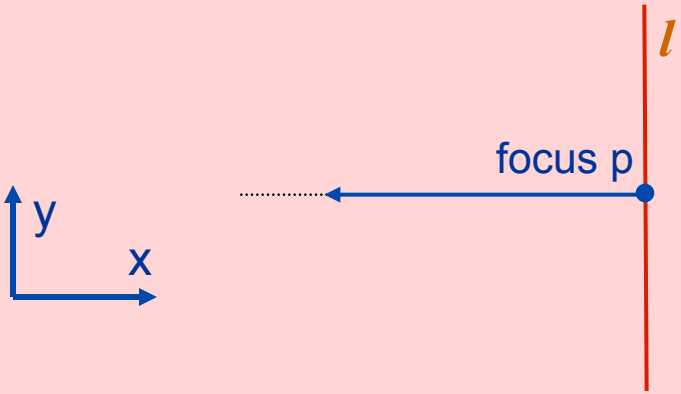
Sweep Plane & Cone Intersection



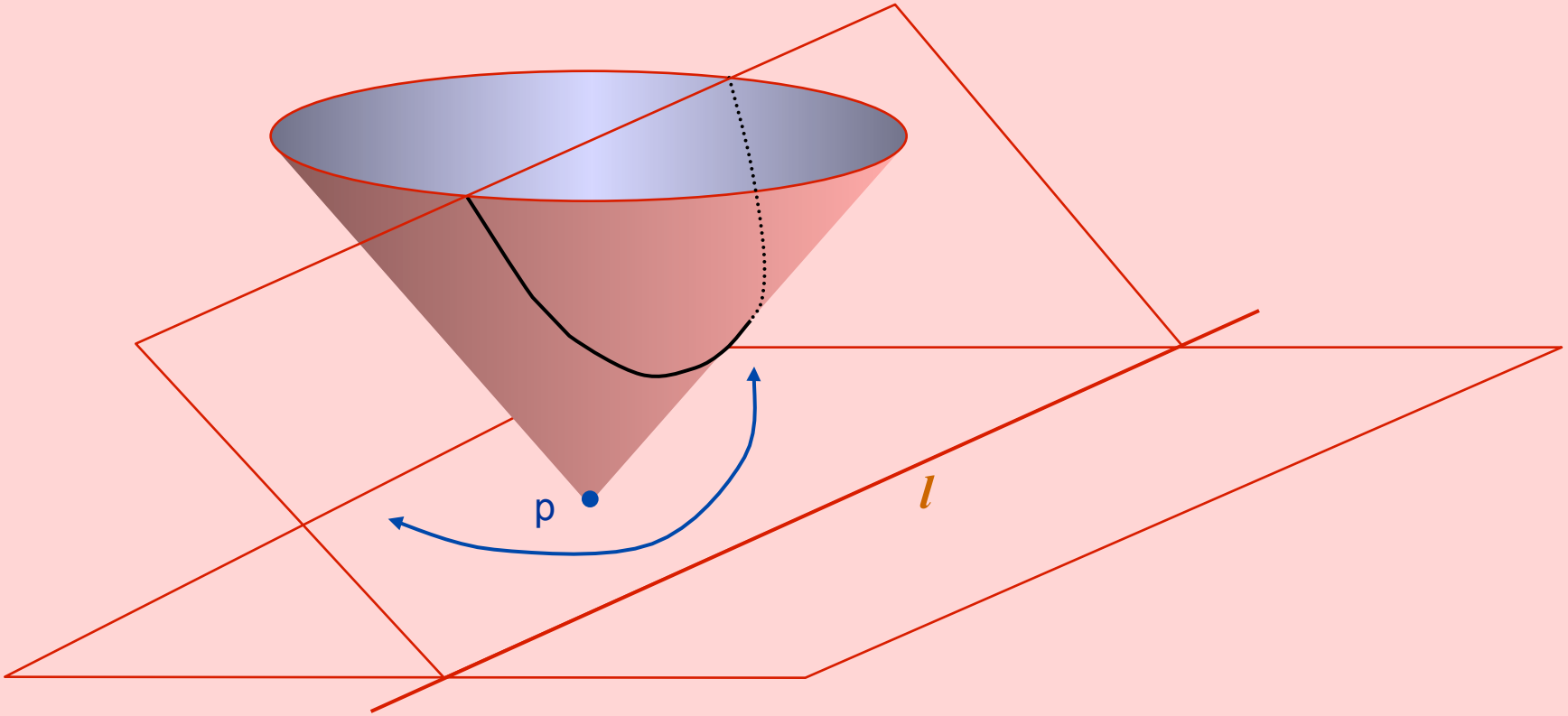
Vertical projection of intersection of cone(p) & the sweep plane on the base plane is a **parabola** with focus p and directrix l.

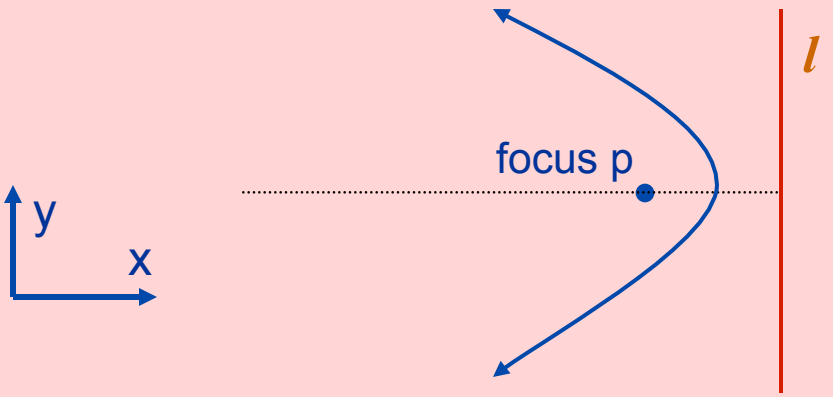
Parabolic Evolution



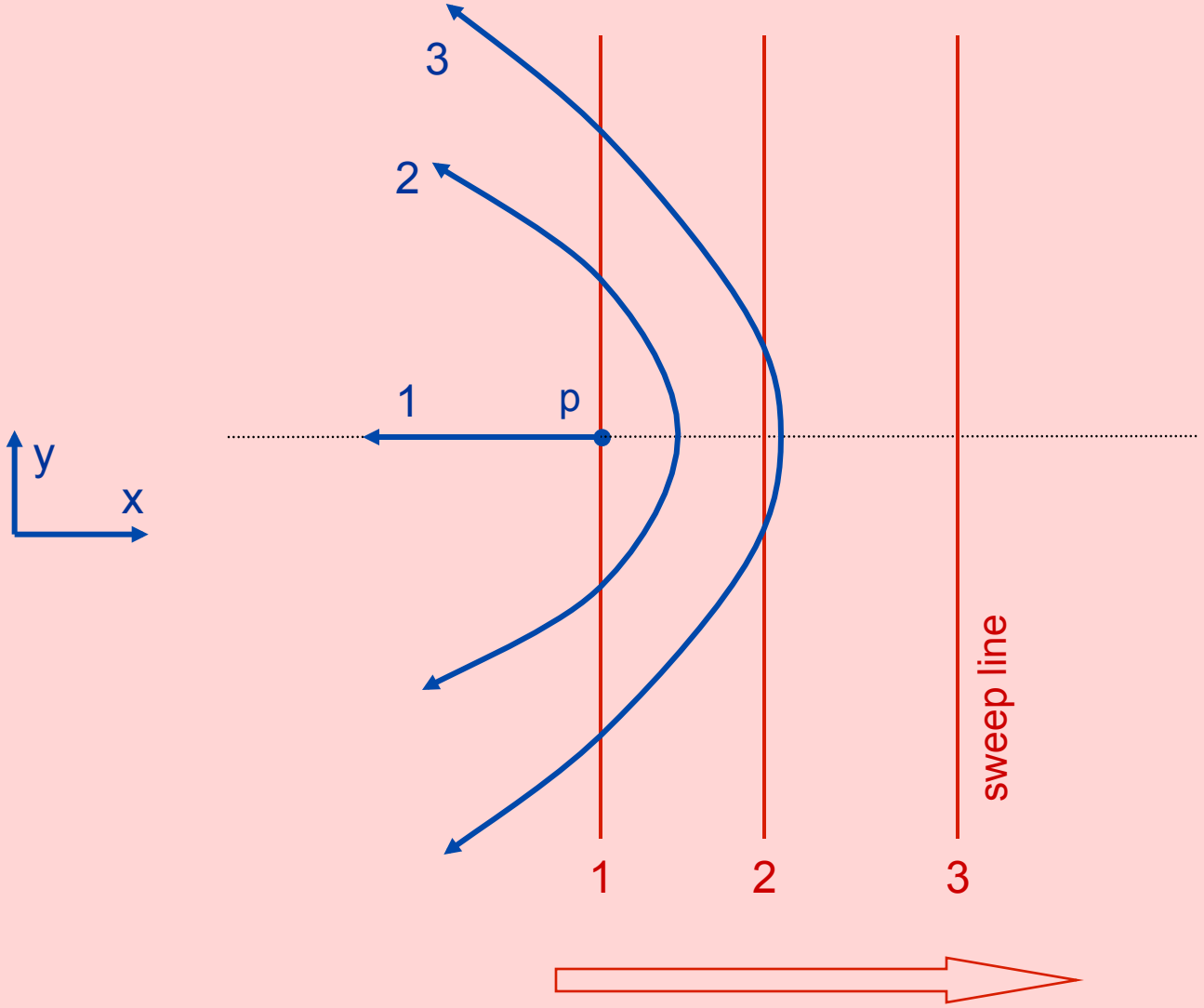


Parabolic Evolution





Time snapshots of moving parabola associated with site p

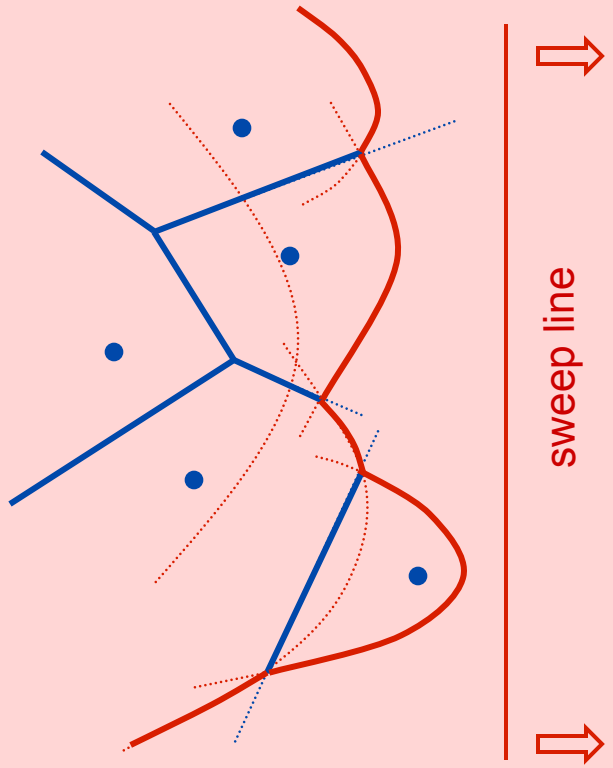


The parabolic front

- Sweep plane opaque. So we don't see future events.
- Any part of a parabola inside another one is **invisible**, since a point (x,y) is inside a parabola iff at that point the cone of the parabola is below the sweep plane.
- **Parabolic Front** = visible portions of parabola; those that are on the boundary of the union of the cones past the sweep.
- Parabolic Front is a **y-monotone** piecewise-parabolic chain. (Any horizontal line intersects the Front in exactly one point.)

- Each **parabolic arc** of the Front is in some Voronoi region.

- Each **"break"** between 2 consecutive parabolic arcs lies on a Voronoi edge.



Evolution of the parabolic front

- The breakpoints of the parabolic front trace out every Voronoi edge as the sweep line moves from $x = -\infty$ to $x = +\infty$.

- Every point of every Voronoi edge is a breakpoint of the parabolic front at some time during the sweep.

Proof:

(a) Fig 1: Event w:
C_u is an empty circle.

(b) Fig 2: At event w point u must be a breakpoint of the par. front.
Otherwise:
Some parabola Z covers u at v
⇒
Focus of Z is on C_v and C_v is inside C_u
⇒
Focus of Z is inside C_u
⇒
C_u is not an empty circle
⇒
a contradiction.

Fig 1.

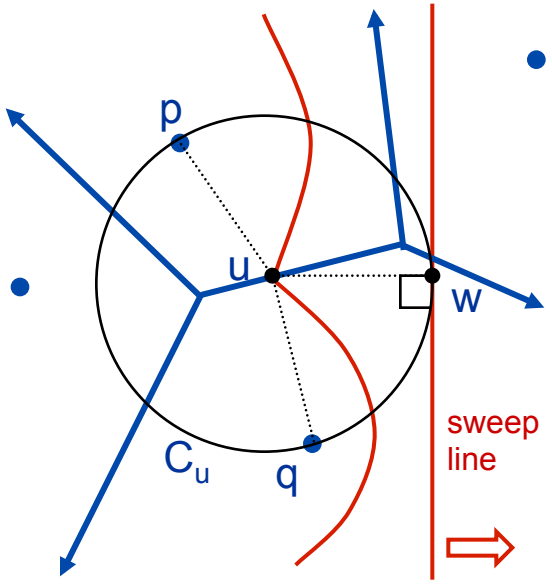
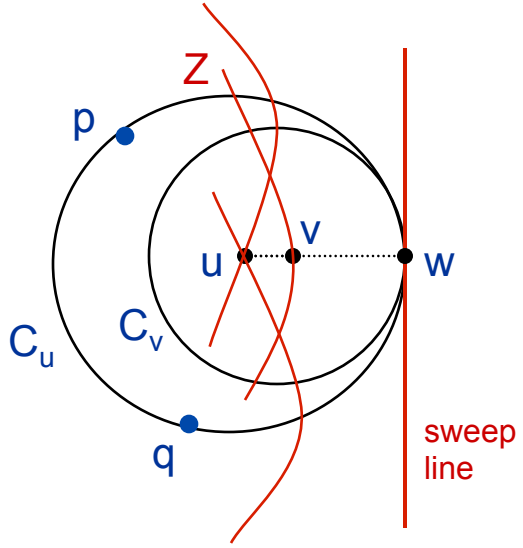


Fig 2.

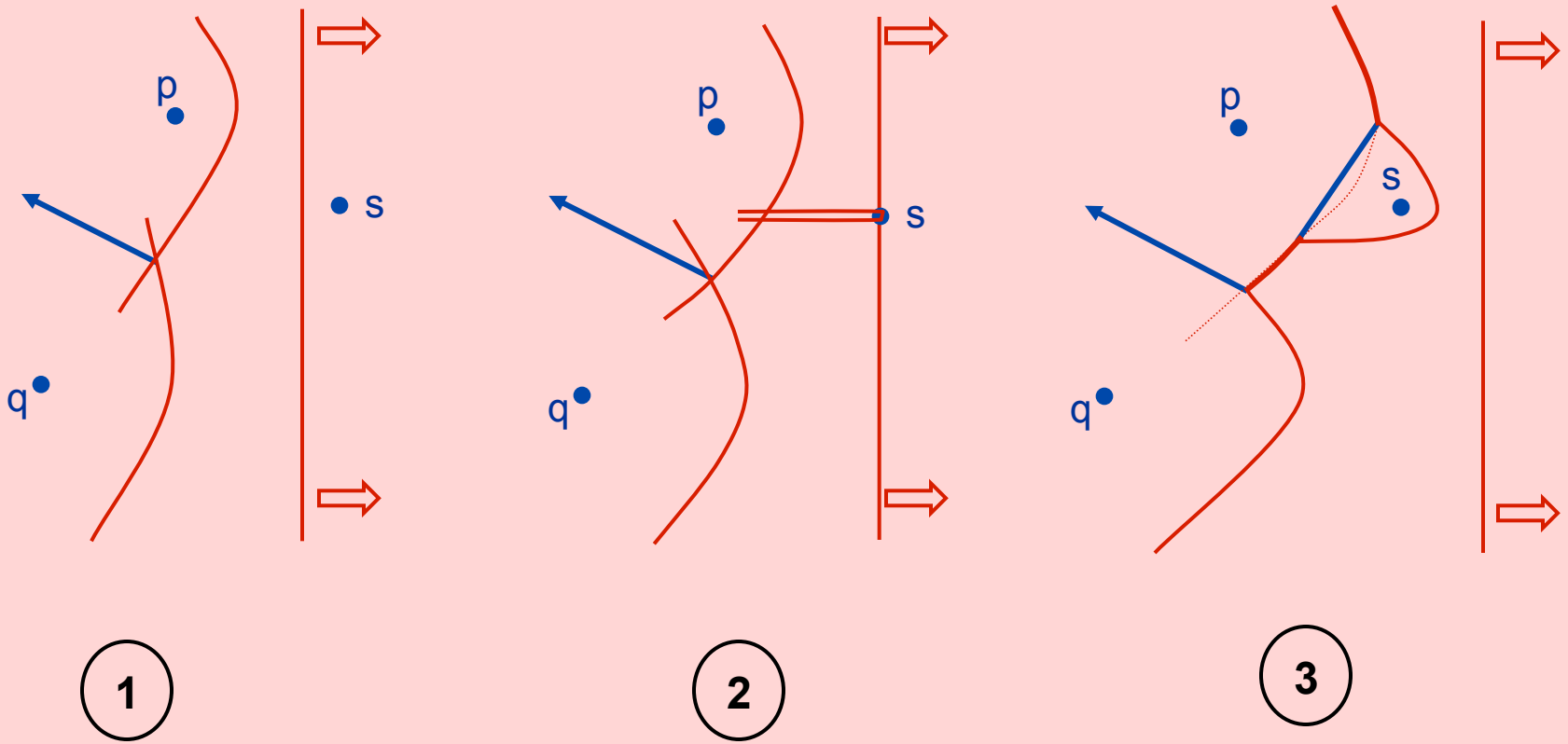


The Discrete Events

- **SITE EVENT:** Insert into the Parabolic Front.
- **CIRCLE EVENT:** Delete from the Parabolic Front.

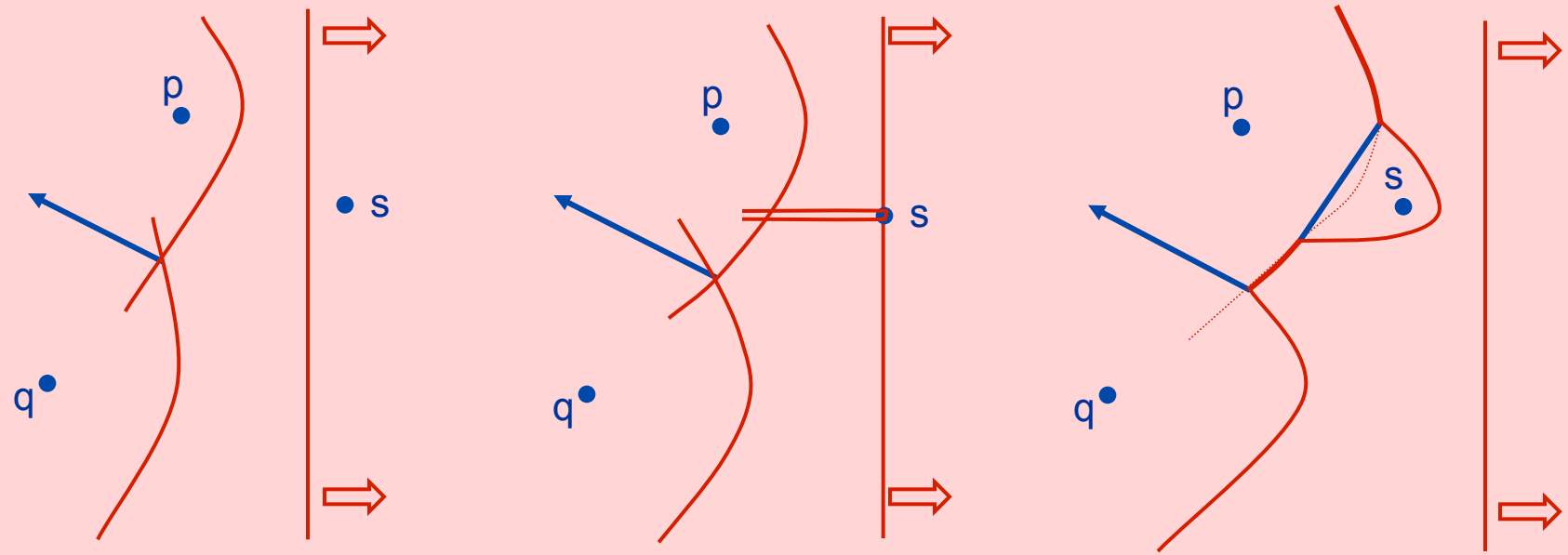
SITE EVENT

A new parabolic arc is inserted into the front when sweep line hits a new site.



SITE EVENT

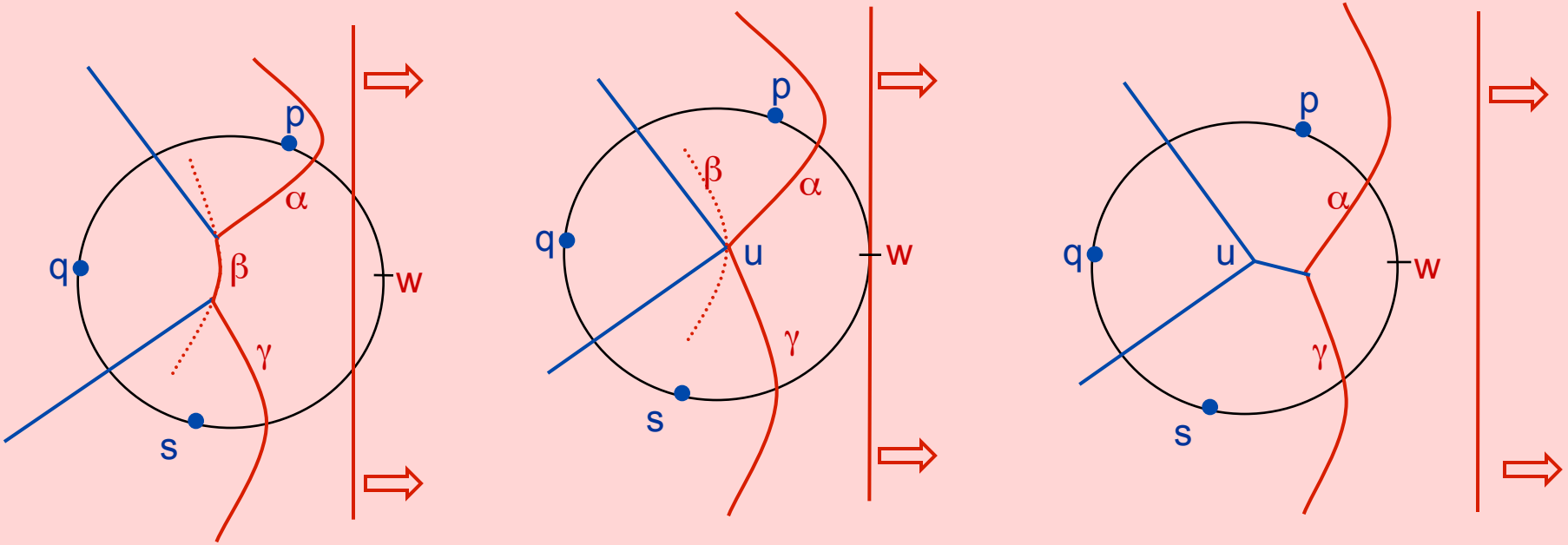
A new parabolic arc is inserted into the front when sweep line hits a new site.



A parabola cannot appear on the front by breaking through from behind.
The following are impossible:

CIRCLE EVENT

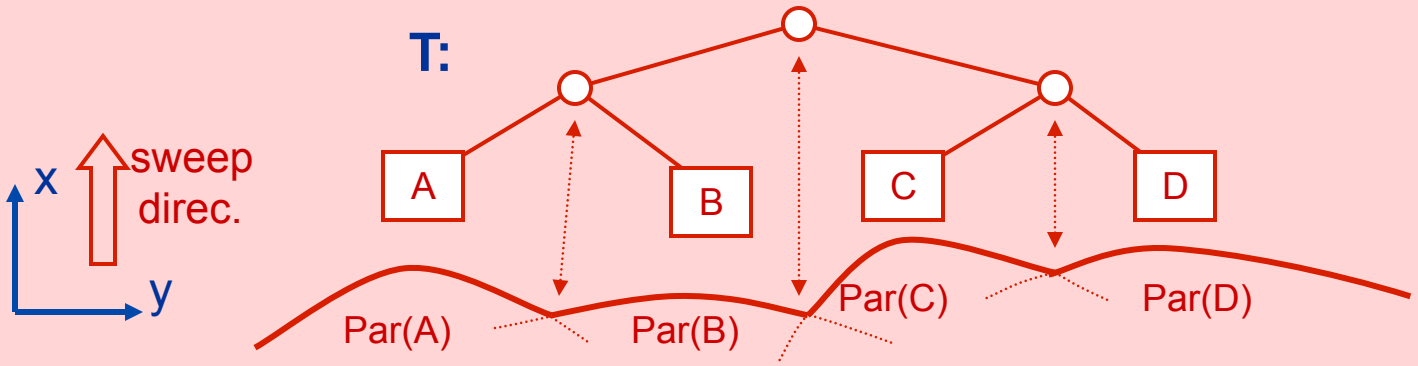
- Circle event w causes parabolic arc β to disappear.
- α and γ cannot belong to the same parabola.



DATA STRUCTURES (T & Q)

T: [SWEEP STATUS: a balanced search tree]
maintains a description of the current parabolic front.

Leaves: arcs of the parabolic front in y-monotone order.
Internal nodes: the break points.



Operations:
(a) insert/delete an arc.
(b) locate an arc intersecting a given horizontal line (for site event).
(c) locate the arcs immediately above/below a given arc (for circle event).

We also hang from this the part of the Voronoi Diagram swept so far.
- Each leaf points to the corresponding site.
- Each internal node points to the corresponding Voronoi edge.

DATA STRUCTURES (T & Q)

Q: [SWEEP SCHEDULE: a priority queue] schedule of future events:

- all future site-events &
- some circle-events, i.e.,
  - those corresponding to 3 consecutive arcs of the current parabolic front as represented by T.
  - The others will be discovered & added to the sweep schedule before the sweep lines advances past them.
  - Conversely, not every 3 consecutive arcs of the current front specify a circle-event. Some arcs may drop out too early.

Event Processing & Scheduling

Event-driven simulation loop:

At each iteration remove the next event (with min x-coordinate) from Q & simulate the effect of the sweep-line advancing past that event point.

Event Processing & Scheduling

Event-driven simulation loop:

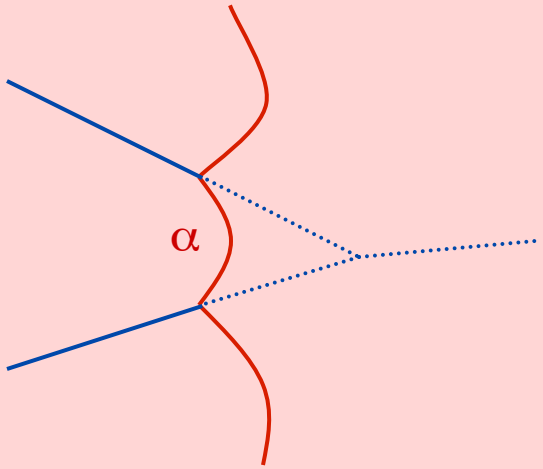
At each iteration remove the next event (with min x-coordinate) from Q & simulate the effect of the sweep-line advancing past that event point.

death(α) : pointing to a circle-event in Q as the meeting point of the Voronoi edges. (If the edges are diverging, then death(α) = nil.)

Remove circle-event death(α) if:

(a) α is split in two by a site-event, or

(b) whenever one of the two arcs adjacent to α is deleted by a circle-event.



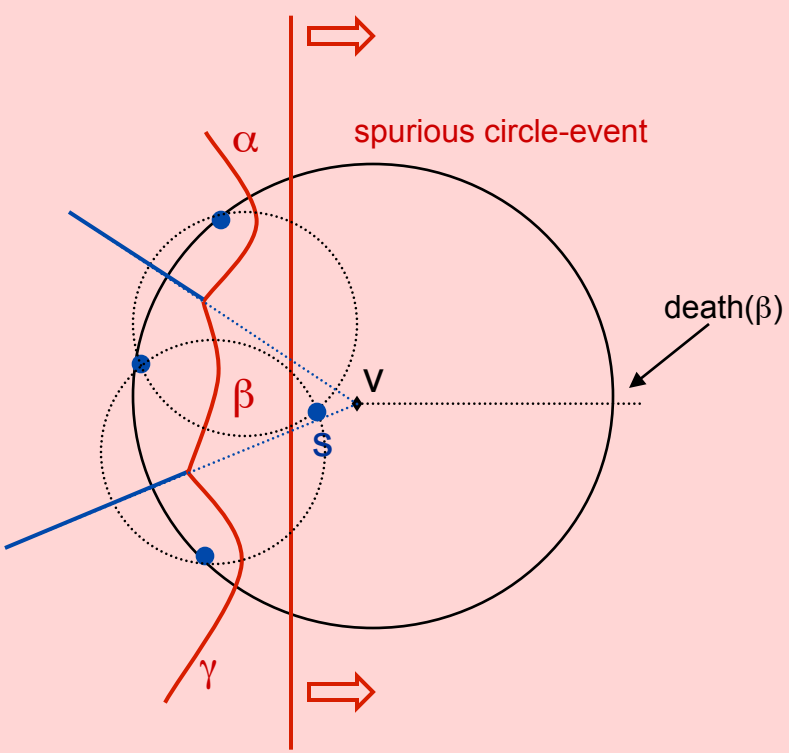
Event Processing & Scheduling

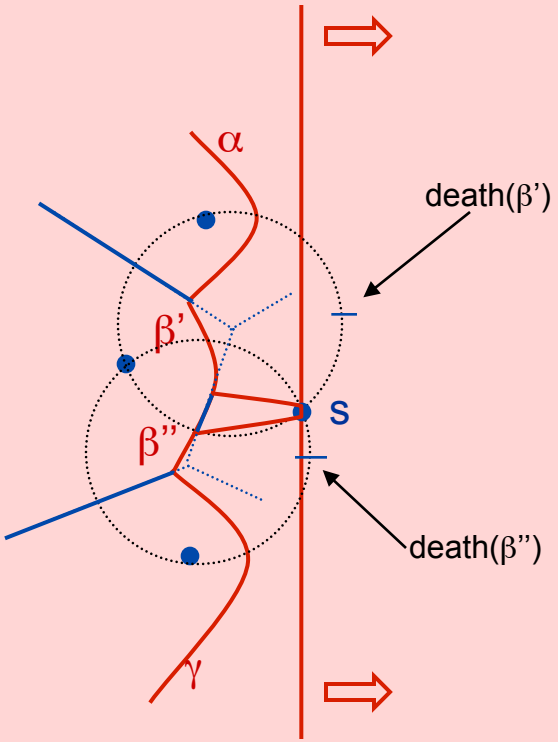
Event-driven simulation loop:

At each iteration remove the next event (with min x-coordinate) from Q & simulate the effect of the sweep-line advancing past that event point.

A circle-event update:

each parabolic arc β (leaf of T) points to the earliest circle-event, death(β), in Q that would cause deletion of β at the corresponding Voronoi vertex.





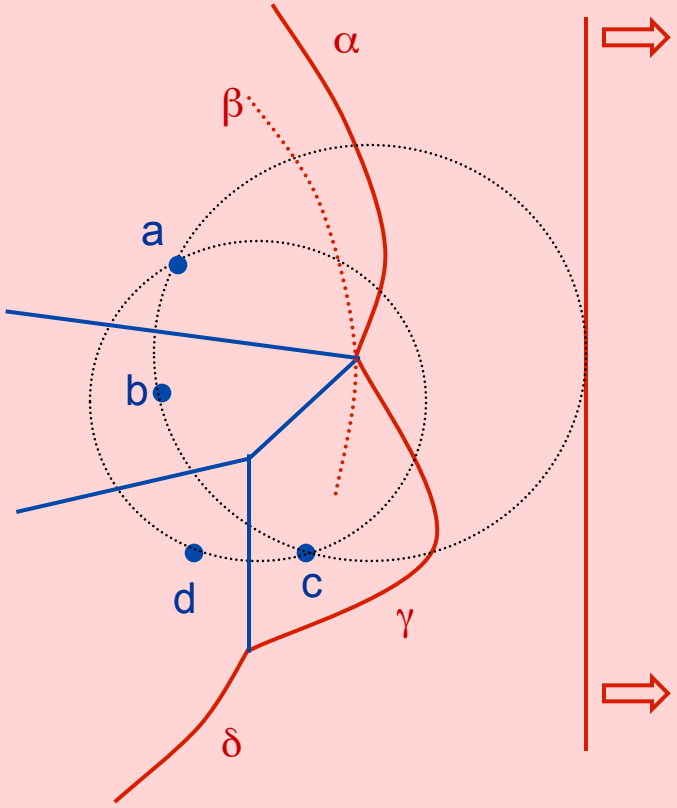
Event Processing & Scheduling

Event-driven simulation loop:

At each iteration remove the next event (with min x-coordinate) from Q & simulate the effect of the sweep-line advancing past that event point.

(α,γ,δ) do not define a circle-event:

(a,c,d) is not a circle-event now, it is past the current sweep position.



ANALYSIS

|T| = O(n) : the front always has O(n) parabolic arcs, since splits occur at most n times by site events.

Also by Davenport-Schinzel:

… α … β … α … β … is impossible.

[At most 2n-1 parabolic arcs in T.]

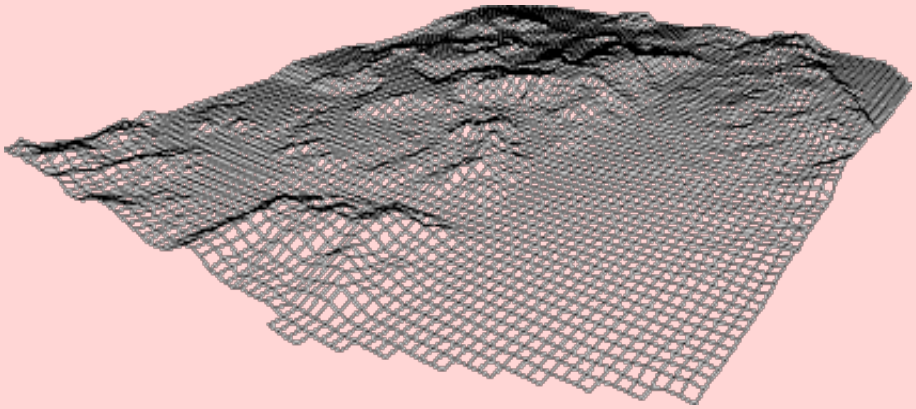
|Q| = O(n) : there are at most n site-events and O(n) triples of consecutive arcs on the parabolic front to define circle-events.

Total # events = O(n), Time per event processing = O(log n).

THEOREM: Fortune's algorithm computes Voronoi Diagram of n sites in the plane using optimal O(n log n) time and O(n) space.

Delaunay Triangulation

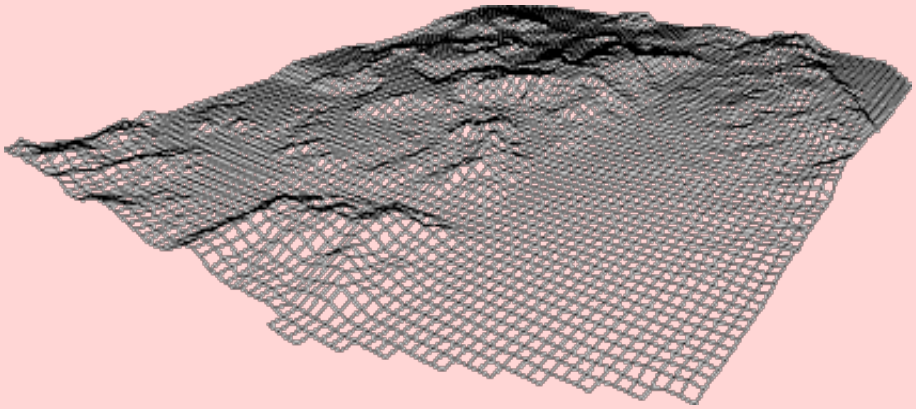
Terrain Height Interpolation



A perspective view of a terrain.

A topographical map of a terrain.

Terrain Height Interpolation

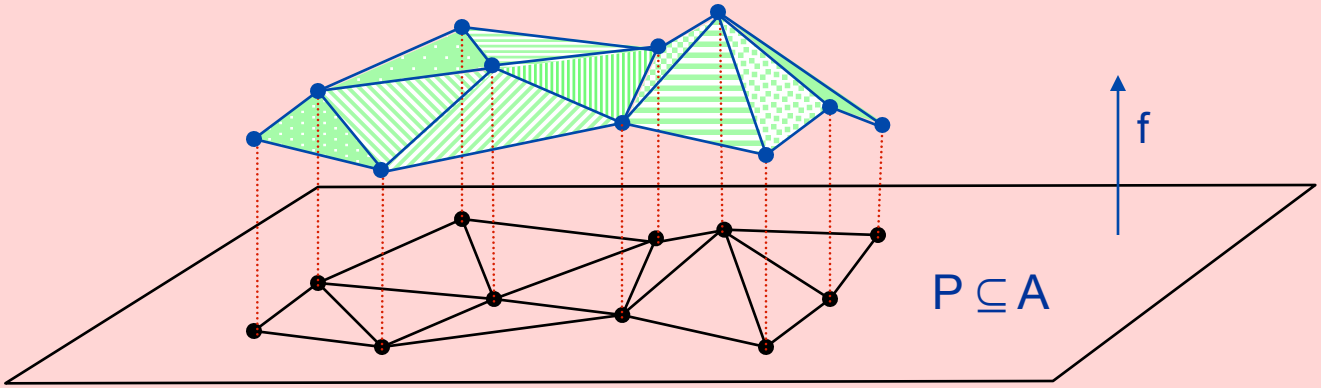


A perspective view of a terrain.

A topographical map of a terrain.

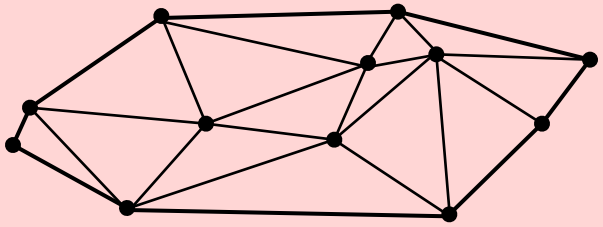
Terrain: A 2D surface in 3D such that each vertical line intersects it in at most one point.
$f : A \subseteq \Re^2 \longrightarrow \Re$. f(p) = height of point p in the domain A of the terrain.

Method: Take a finite sample set $P \subseteq A$. Compute f(P), and interpolate on A.



Triangulations of Planar Point Sets

$P = \{p_1, p_2, \ldots, p_n\} \subseteq \Re^2$.
A triangulation of P is a maximal planar straight-line subdivision with vertex set P.



THEOREM: Let P be a set of n points, not all collinear, in the plane.
Suppose h points of P are on its convex-hull boundary.
Then any triangulation of P has 3n-h-3 edges and 2n-h-2 triangles.

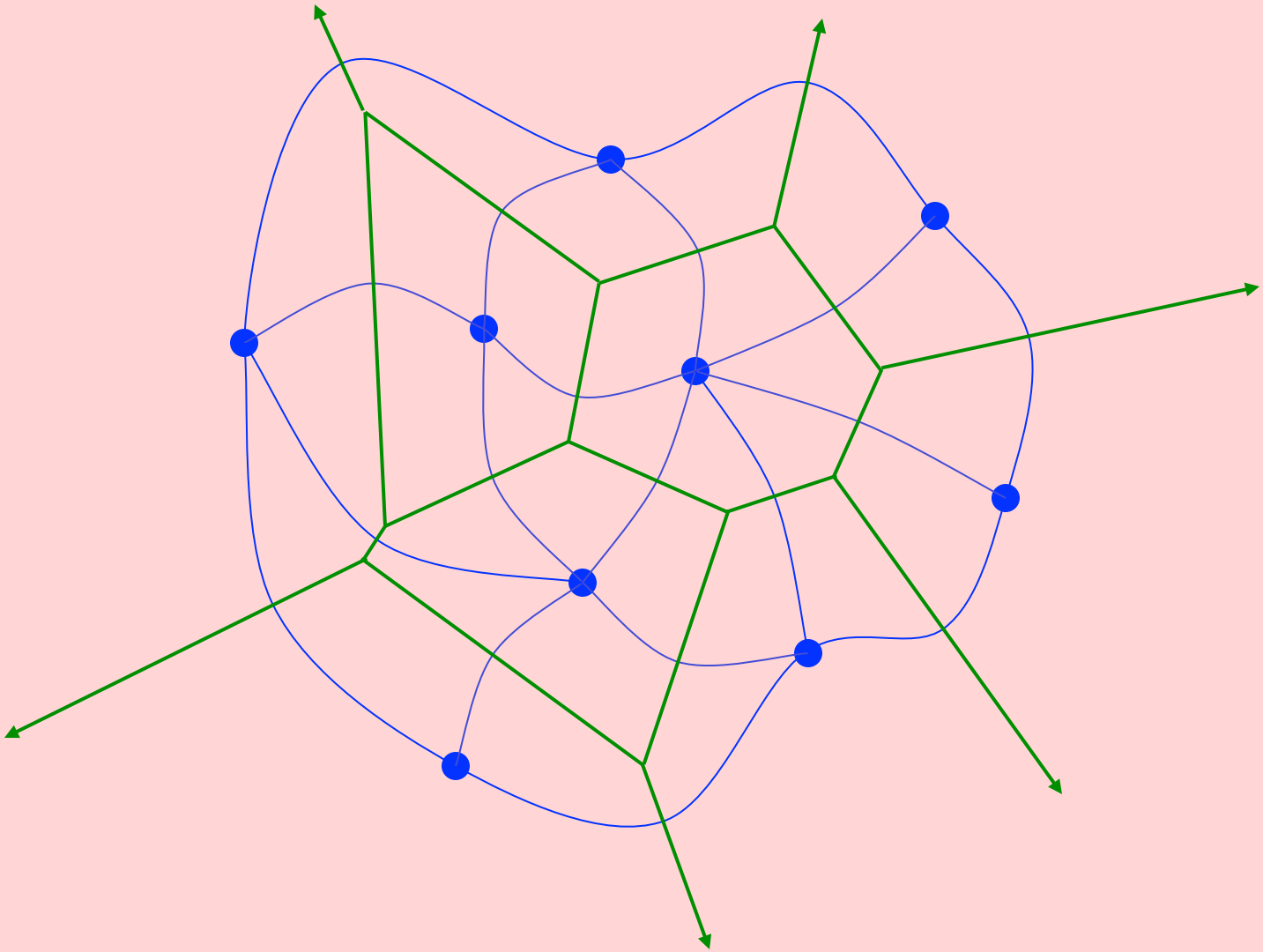
Proof: m = # triangles

$3m + h = 2E$ (each triangle has 3 edges; each edge incident to 2 faces)

Euler: $n - E + (m+1) = 2$

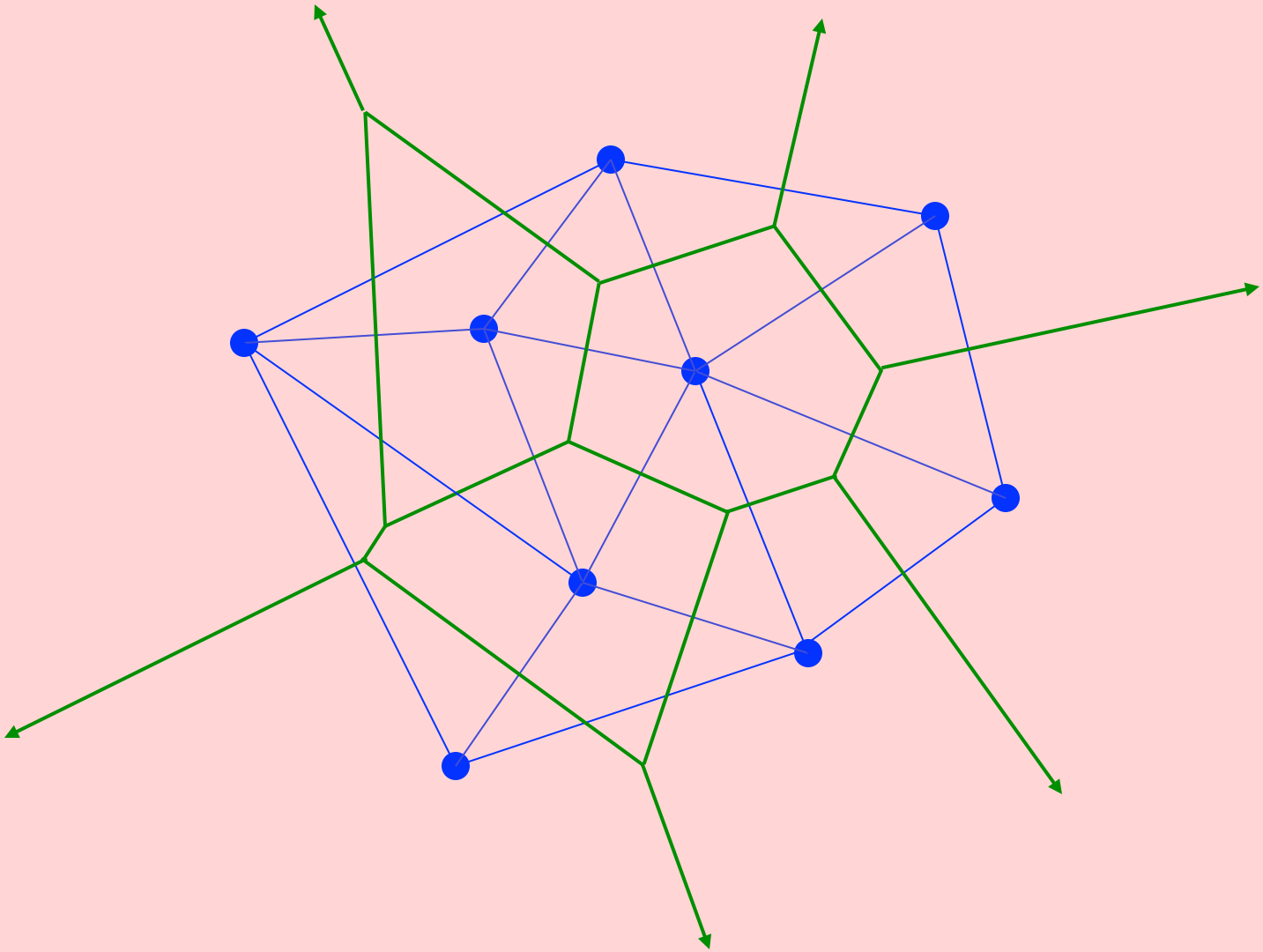
$\therefore m = 2n - h - 2, \quad E = 3n - h - 3.$

Delaunay Graph: Dual of Voronoi Diagram



Delaunay Graph DG(P) as dual of Voronoi Diagram VD(P).

Delaunay Graph: Dual of Voronoi Diagram



Delaunay Graph DG(P) as strainght-line dual of Voronoi Diagram VD(P).

Delaunay Graph is a Triangulation

Alternative Definition of Delaunay Graph:

- A triangle $\Delta(p_i , p_j , p_k)$ is a Delaunay triangle iff the circumscribing circle $C(p_i , p_j , p_k)$ is empty.
- Line segment (p_i, p_j) is a Delaunay edge iff there is an empty circle passing through p_i and p_j, and no other point in P.

THEOREM: Delaunay Graph of P is

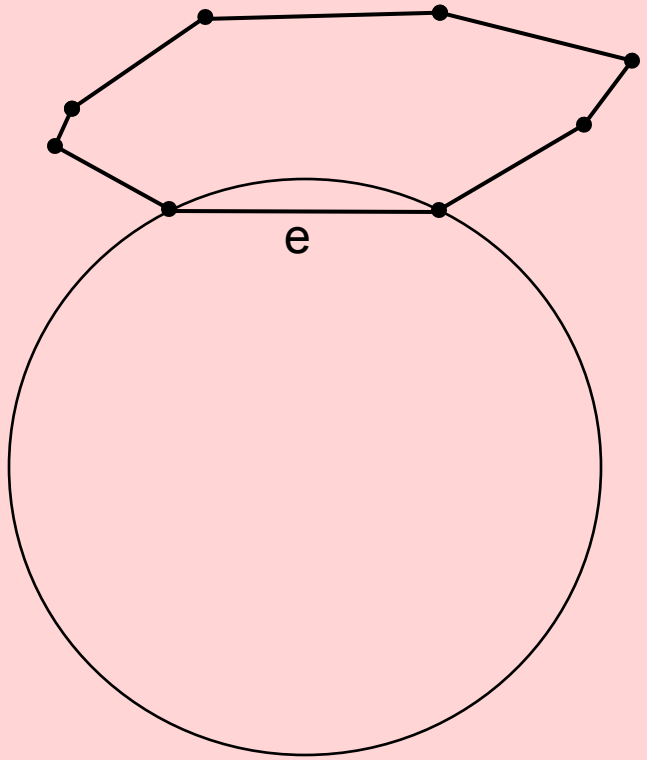
- a straight-line plane graph, &
- a triangulation of P.

Proof: Follows from the following Lemmas.

Delaunay Graph is a Triangulation

LEMMA 1: Every edge of CH(P) is a Delaunay edge.

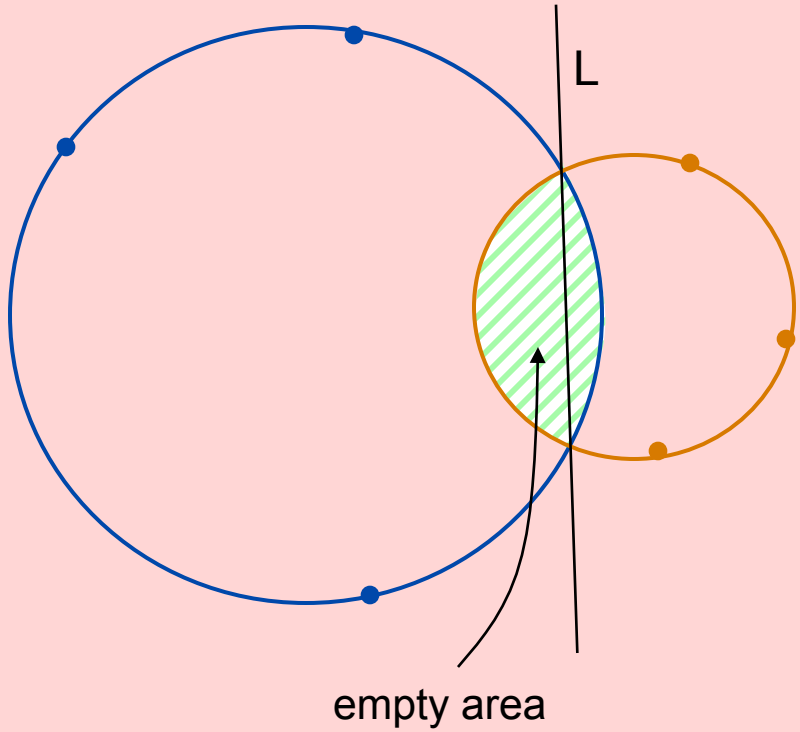
Proof: Consider a sufficiently large circle that passes through the 2 ends of CH edge e, and whose center is separated from CH(P) by the line aff(e).



Delaunay Graph is a Triangulation

LEMMA 2: No two Delaunay triangles overlap.

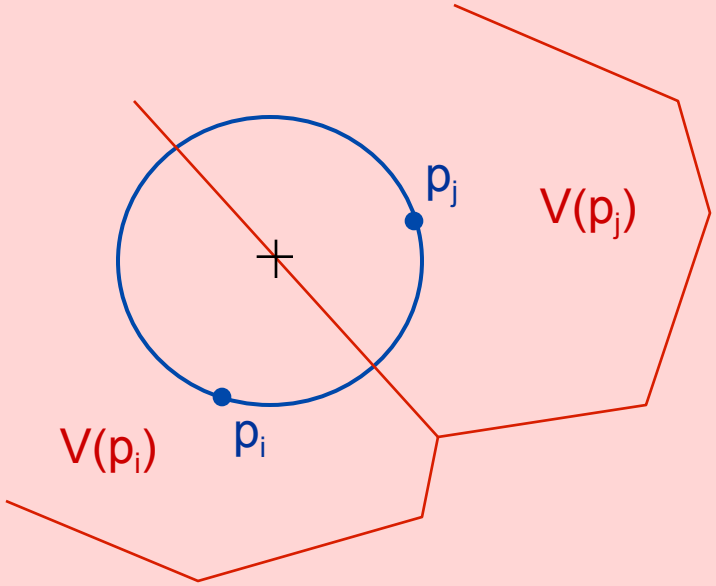
Proof: Consider circumscribing circles of two such triangles.
Line L separates the two triangles.



Delaunay Graph is a Triangulation

LEMMA 3: p_i & p_j are Voronoi neighbors $\Rightarrow$ (p_i, p_j) is a Delaunay edge.

Proof: Consider the circle that passes through p_i & p_j and whose center is in the relative interior of the common Voronoi edge between $V(p_i)$ & $V(p_j)$.



Delaunay Graph is a Triangulation

LEMMA 4: If p_j and p_k are two (rotationally) successive Voronoi neighbors of p_i & $\angle p_j p_i p_k < 180°$, then $\Delta(p_i, p_j, p_k)$ is a Delaunay triangle.

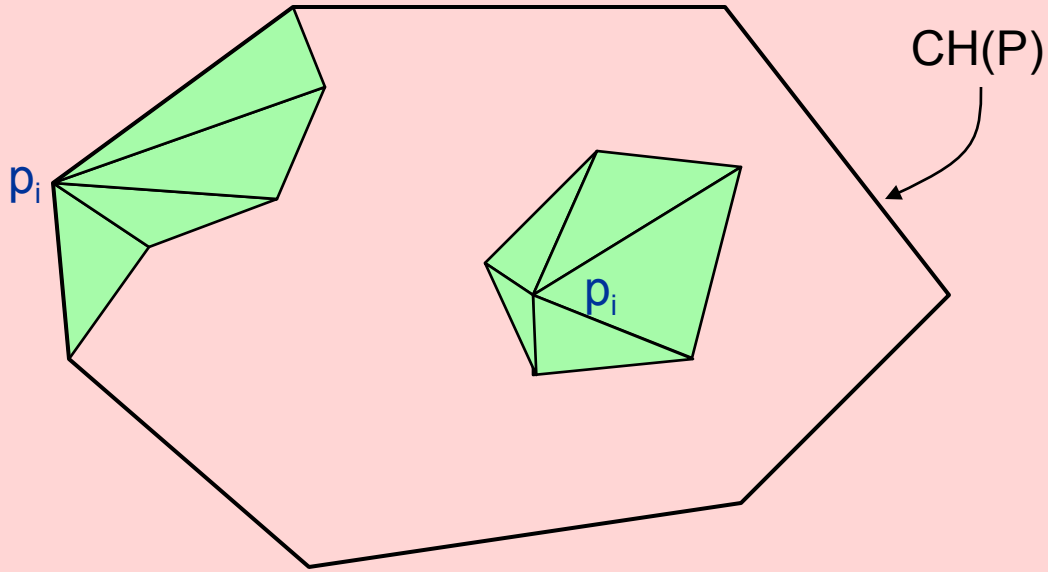
Proof: p_j & p_k must also be Voronoi neighbors.
Now apply Lemma 3 to (p_i, p_j), (p_i, p_k), (p_j, p_k).

Delaunay Graph is a Triangulation

LEMMA 4: If p_j and p_k are two (rotationally) successive Voronoi neighbors of p_i & $\angle p_j p_i p_k < 180°$, then $\Delta(p_i, p_j, p_k)$ is a Delaunay triangle.

Proof: p_j & p_k must also be Voronoi neighbors.
Now apply Lemma 3 to (p_i, p_j), (p_i, p_k), (p_j, p_k).

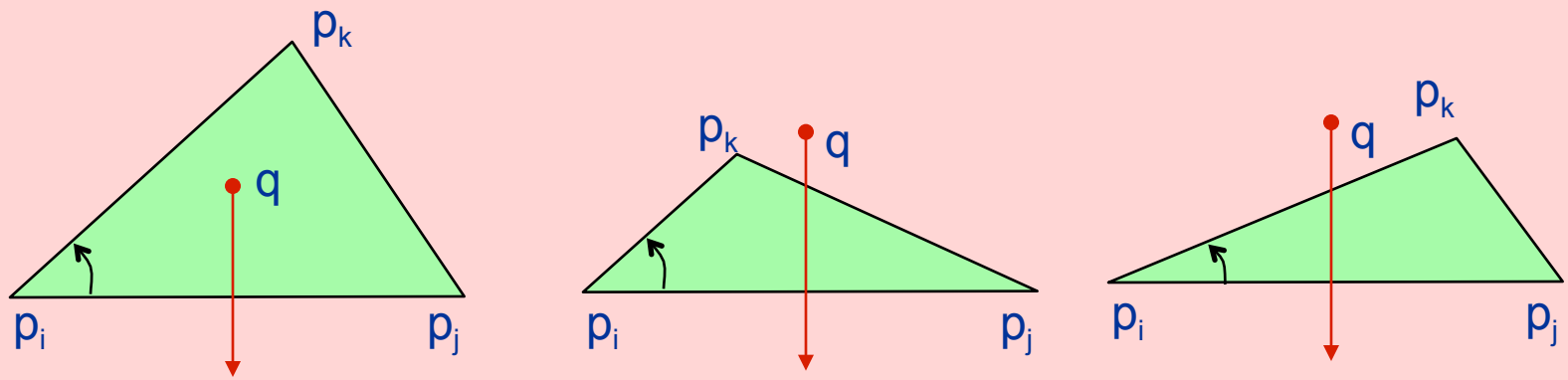
COROLLARY 5: For each $p_i \in P$, the Delaunay triangles incident to p_i completely cover a small open neighborhood of p_i inside CH(P).



Delaunay Graph is a Triangulation

LEMMA 6: Every point inside CH(P) is covered by some Delaunay triangle in DG(P).

Proof: Let q be an arbitrary point in CH(P). Let (p_i , p_j) be the Delaunay edge immediately below q. ((p_i , p_j) exists because all convex-hull edges are Delaunay by Lemma 1.) From Corollary 5 let $\Delta(p_i,p_j,p_k)$ be the next Delaunay triangle incident to p_i as in the Figure below. Then, either $q \in \Delta(p_i,p_j,p_k)$, or the choice of (p_i , p_j) is contradicted.



The THEOREM follows from Lemmas 2-6. We now use DT(P) to denote the Delaunay triangulation of P.

Angles in Delaunay Triangulation

DEFINITION:
$\mathcal{T}$ = an arbitrary triangulation (with m triangles) of point set P.
$\alpha_1, \alpha_2, \ldots, \alpha_{3m}$ = the angles of triangles in $\mathcal{T}$, sorted in increasing order.
$A(\mathcal{T}) = (\alpha_1, \alpha_2, \ldots, \alpha_{3m})$ is called the angle-vector of $\mathcal{T}$.

THEOREM: DT(P) is the **unique** triangulation of P that lexicographically maximizes $A(\mathcal{T})$.

Proof: Later.

COROLLARY: DT(P) maximizes the smallest angle.

Useful for terrain approximation by triangulation & linear interpolation.
Small angles (long skinny triangles) cause large approximation errors.

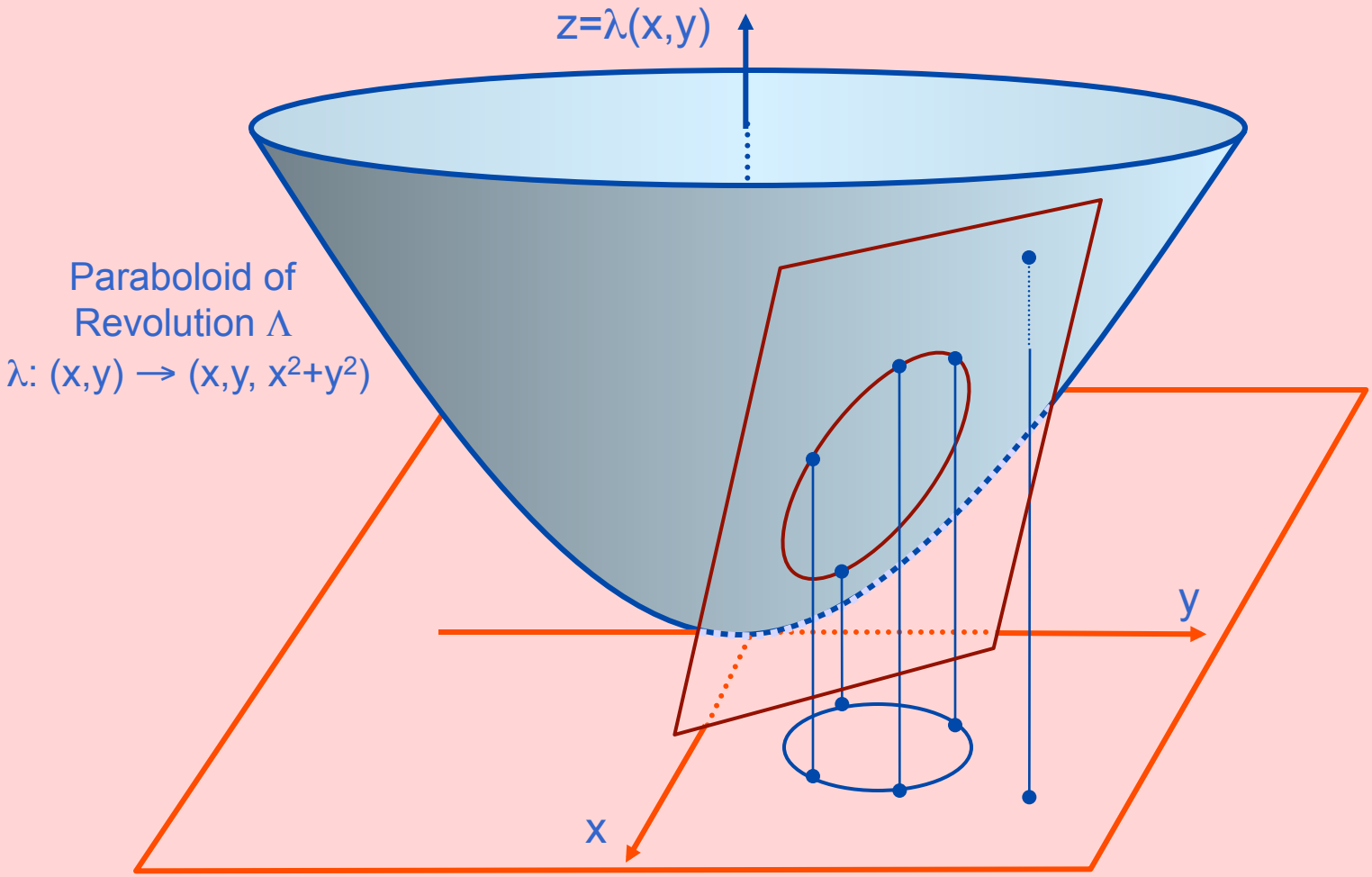
DT & VD via CH

- K.Q. Brown [1979], "Voronoi diagrams from convex hulls," IPL 223-228.
- K.Q. Brown [1980], "Geometric transforms for fast geometric algorithms," PhD. Thesis, CMU-CS-80-101.
- Guibas, Stolfi [1987], "Ruler, Compass and computer: The design and analysis of geometric algorithms," *Proc. of the NATO Advanced Science Institute, series F, vol. 40: Theoretical Foundations of Computer Graphics and CAD*, 111-165.
- Guibas, Stolfi [1985], "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams," ACM Trans. Graphics 4(2), 74-123.
- [Edelsbrunner'87] pp: 302-306.
- Aurenhammer [1987], "Power diagrams: properties, algorithms, and applications," SIAM J. Computing 16, 78-96.

DT in $\Re^d$ $\propto$ CH in $\Re^{d+1}$

Lifting Transform λ : point $(x_1, x_2, \ldots, x_d) \mapsto$ point $(x_1, x_2, \ldots, x_d, x_{d+1})$

where $x_{d+1} = x_1^2 + x_2^2 + \ldots + x_d^2$



DT in $\Re^2$ $\propto$ CH in $\Re^3$

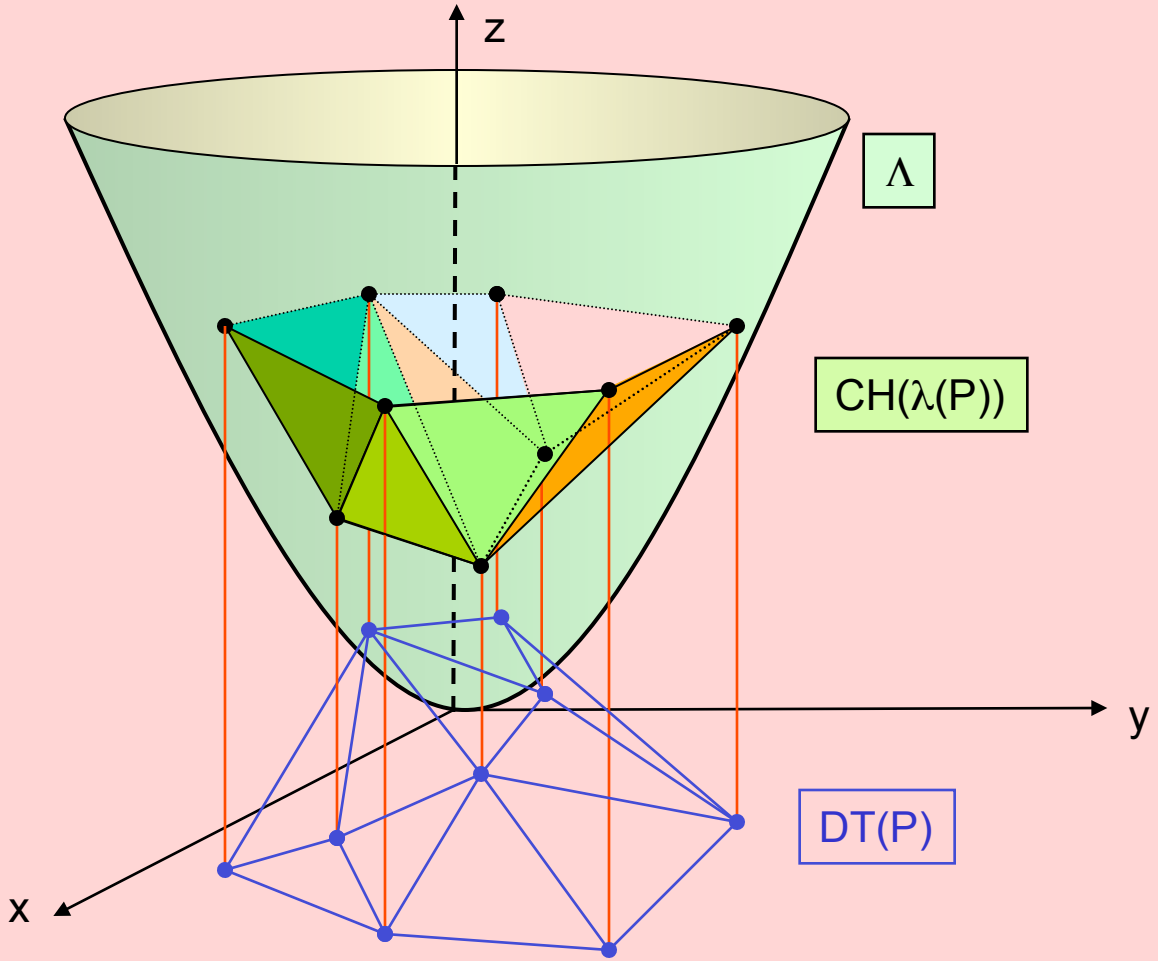
SUMMARY:

Consider a plane Π in $\Re^3$ and the paraboloid of revolution Λ.

(1) Projection of $\Pi \cap \Lambda$ down to $\Re^2$ is a circle C.
(2) Every point of Λ below Π projects down to interior of C.
(3) Every point of Λ above Π projects down to exterior of C.

2D ("Nearest-Point" and "Farthest-Point")
Delaunay Triangulation algorithm via 3D-convex-hull in
O(n log n) time.

DT in $\Re^d$ $\propto$ CH in $\Re^{d+1}$



Generalizations & Applications

The Post Office Problem

PROBLEM: Preprocess a given set P of n points in the plane for:
Nearest Neighbor Query: Given a query point q, determine which point in P is nearest to q.

Shamos [1976]: Slab Method:

Query Time:	$O(\log n)$
Preprocessing Time:	$O(n^2)$
Space:	$O(n^2)$

Kirkpatrick [1983]: Triangulation refinement method for planar point location:

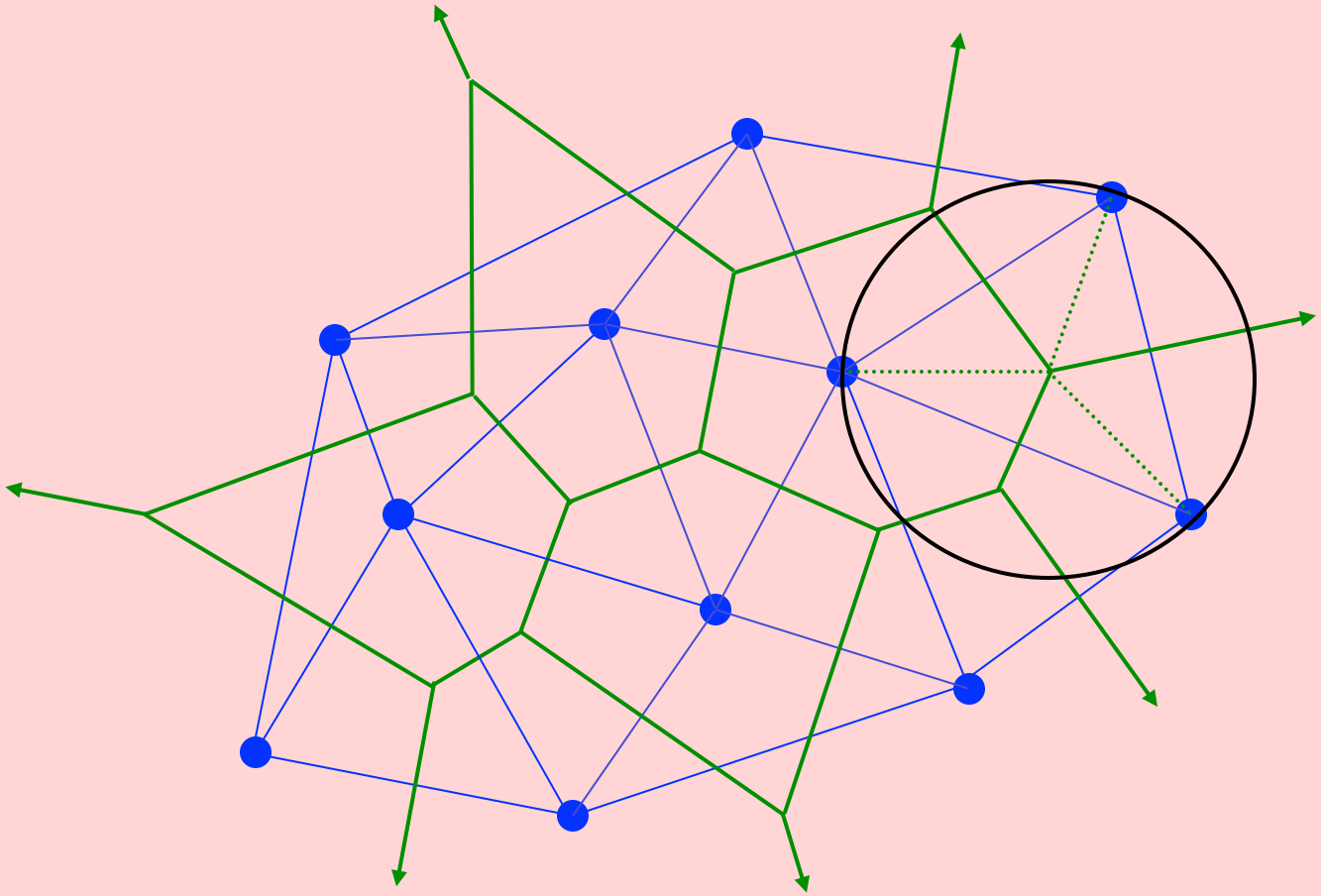
Query Time:	$O(\log n)$
Preprocessing Time:	$O(n \log n)$
Space:	$O(n)$

Construct Voronoi Diagram. Each Voronoi region is convex, hence monotone. Triangulate the Voronoi regions in O(n) time. Then apply Kirkpatrick's method.

Andoni, Indyk [2006] "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," FOCS'06.

Largest Empty Circle Problem

PROBLEM: Determine the largest empty circle with center in CH(P).



O(n) Candidate centers. All can be found in O(n) time (after VD(P) is given):

(1) Voronoi vertex inside CH(P),

(2) Intersection of a Voronoi edge and an edge of CH(P).

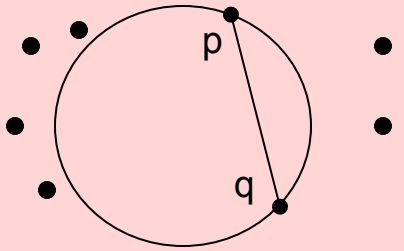
Subgraphs of Delaunay Triangulation

- **Gabriel Graph**
- **Relative Neighborhood Graph**
- **Euclidean Minimum Spanning Tree**
- **Nearest Neighbor Graph**

NNG ⊆ EMST ⊆ RNG ⊆ GG ⊆ DT

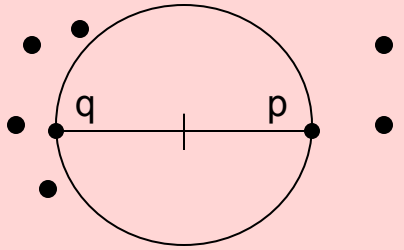
Delaunay Triangulation:

(p,q) is a DT edge ⇔ ∃ empty circle through p and q.



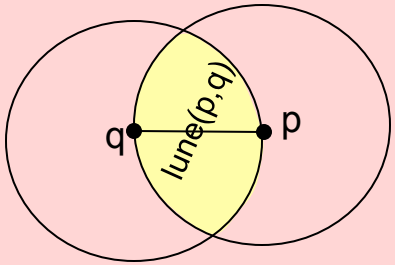
Gabriel Graph: (p,q) is a GG edge ⇔

∃ empty circle with diameter (p,q),
(i.e., (p,q) intersects its dual Voronoi edge).



Relative Neighborhood Graph: (p,q) is an RNG edge ⇔
∀r∈P-{p,q}: (p,q) is **NOT** the longest edge of triangle (p,q,r)
(i.e., d(p,q) ≤ max{d(p,r), d(q,r)})

(i.e., lune(p,q) is empty).

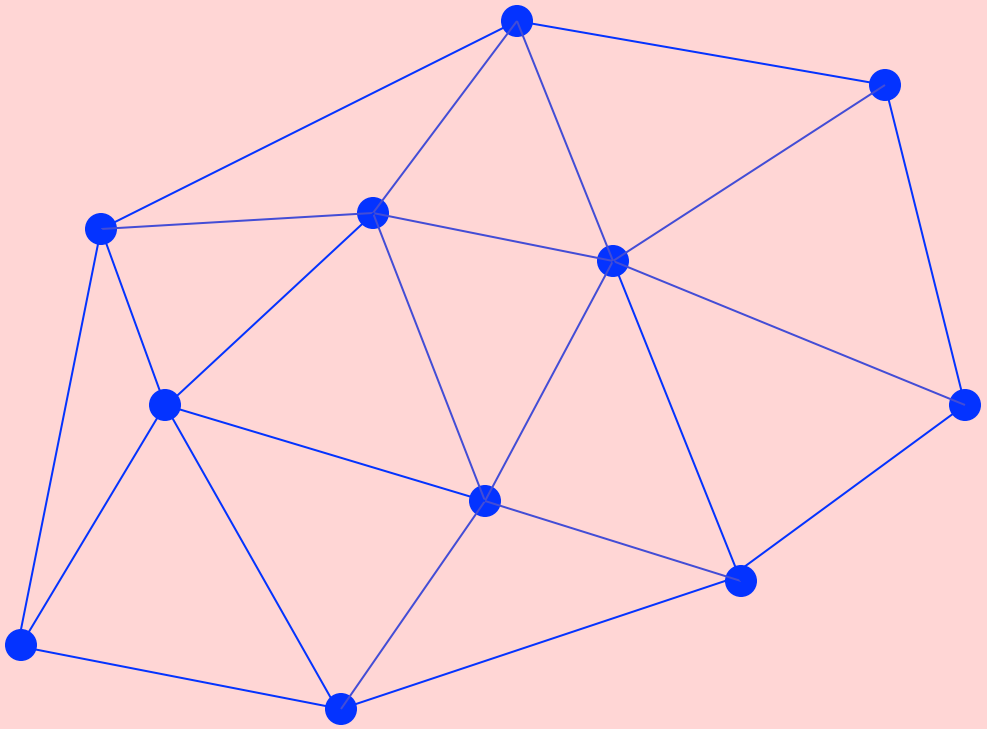


Euclidean Minimum Spanning Tree: (p,q) is in EMST ⇔
∀cycles: (p,q) is **NOT** the longest edge of the cycle.

Nearest Neighbor Graph: (p,q) is a directed edge in NNG ⇔
∀r∈P-{p,q}: d(p,q) ≤ d(p,r).

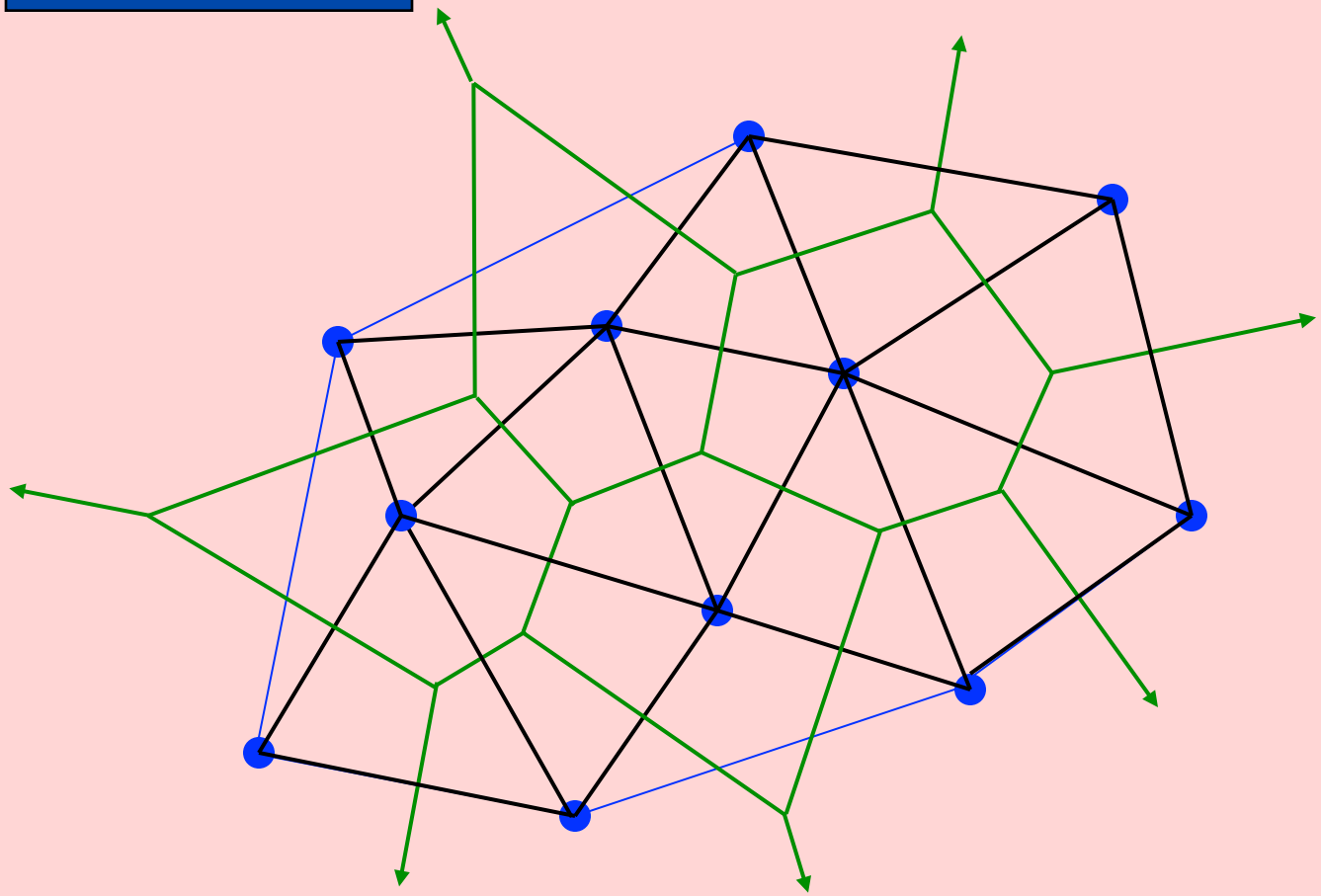
$$NNG \subseteq EMST \subseteq RNG \subseteq GG \subseteq DT$$

Delaunay Triangulation



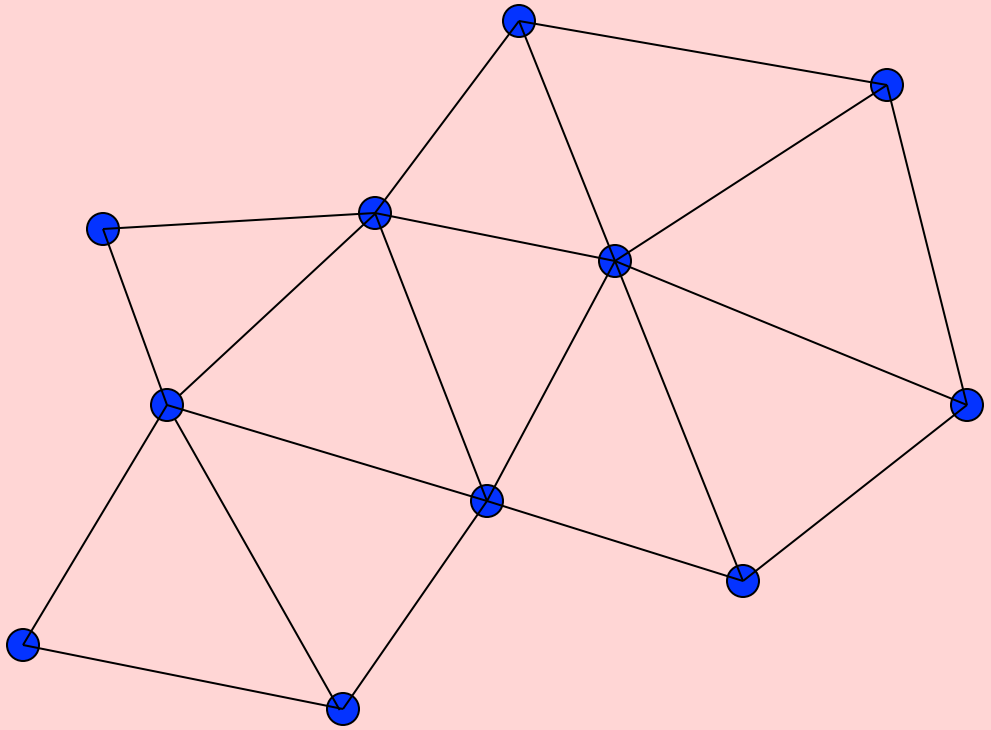
$$NNG \subseteq EMST \subseteq RNG \subseteq GG \subseteq DT$$

Gabriel Graph:



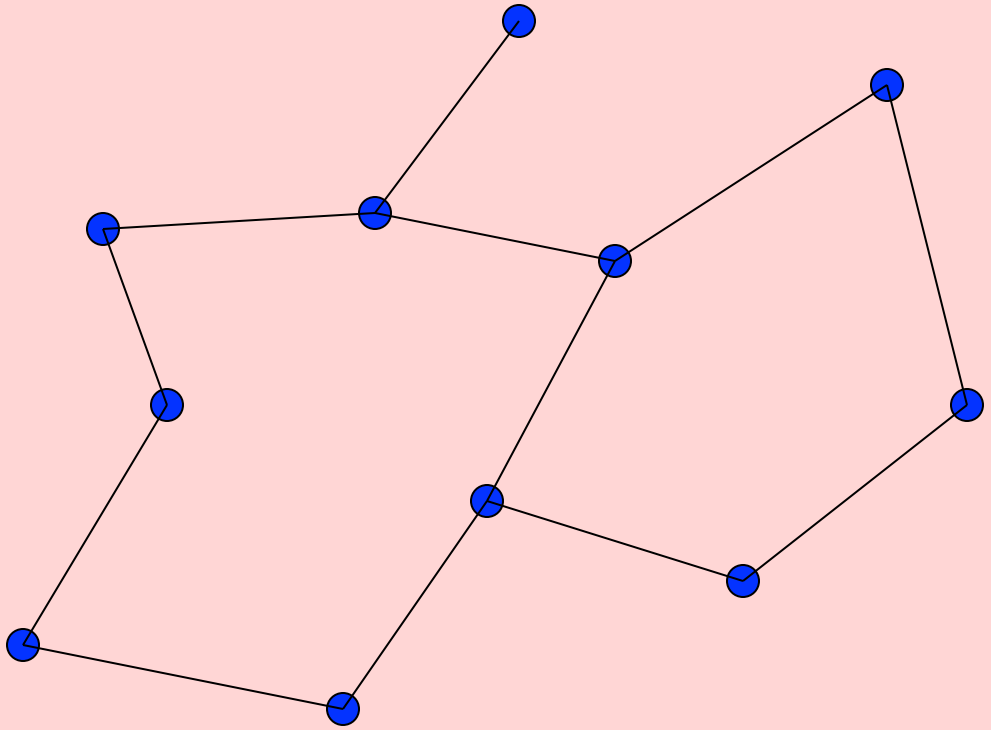
$$NNG \subseteq EMST \subseteq RNG \subseteq GG \subseteq DT$$

Gabriel Graph:



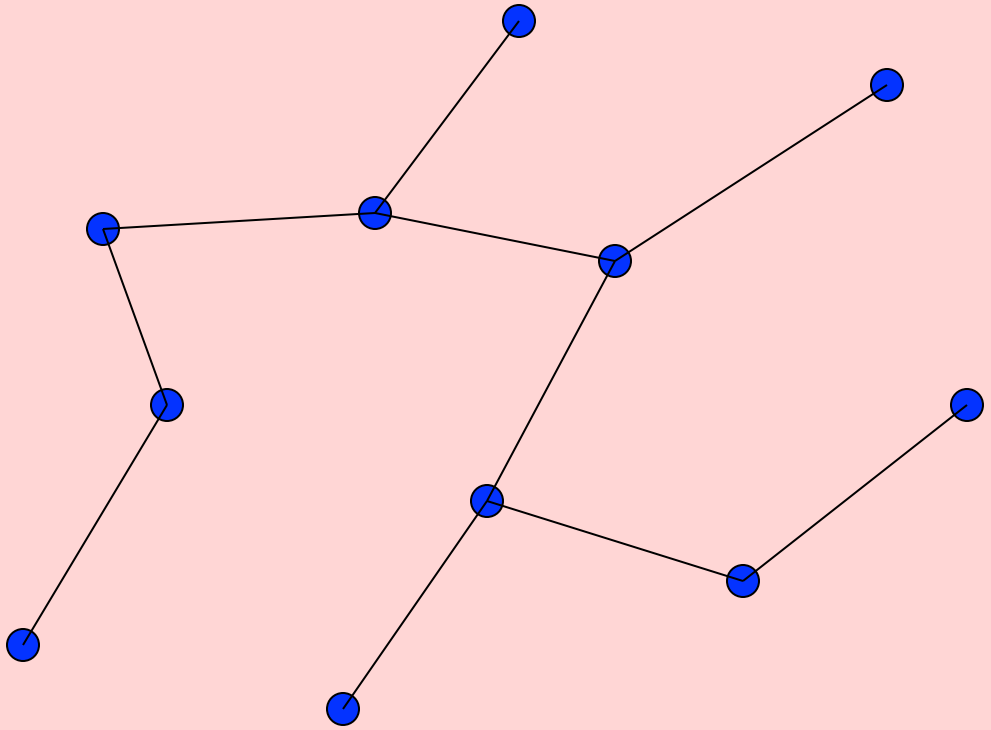
$$NNG \subseteq EMST \subseteq RNG \subseteq GG \subseteq DT$$

Relative Neighborhood Graph:



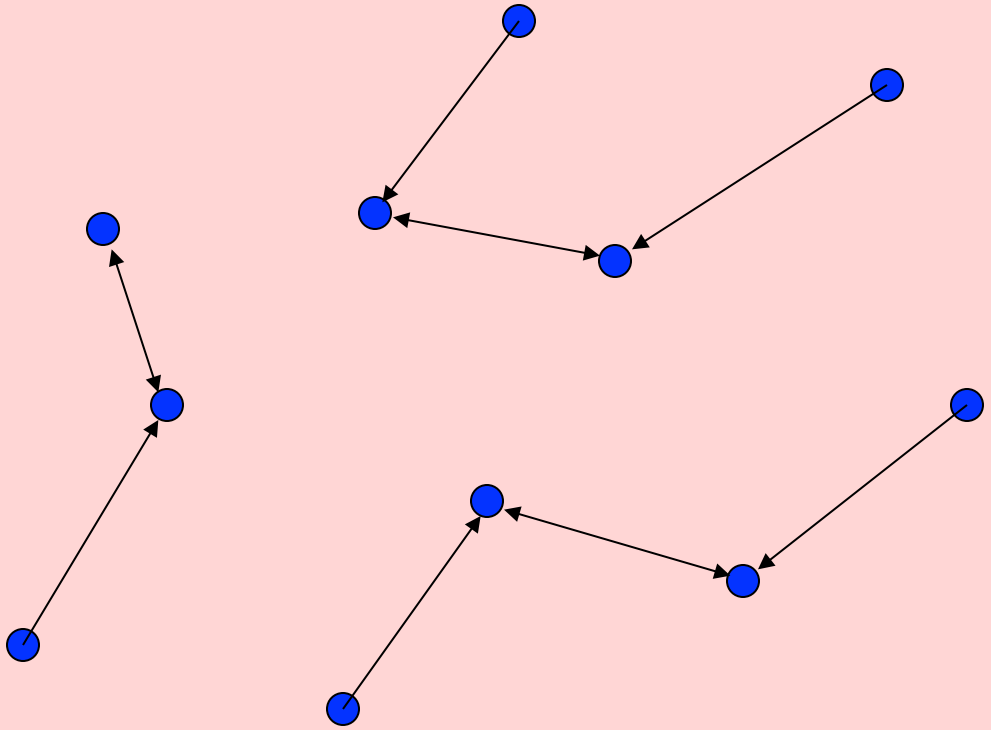
NNG ⊆ EMST ⊆ RNG ⊆ GG ⊆ DT

Euclidean Minimum Spanning Tree:



$NNG \subseteq EMST \subseteq RNG \subseteq GG \subseteq DT$

Nearest Neighbor Graph:



Euclidean Minimum Spanning Tree

- General (m edge, n vertex graph) MST algorithms (See also AAW):

Kruskal or Prim	O(m log n) or O(m + n log n) time.
Yao or Cheriton-Tarjan:	O(m log log n) time
Chazelle:	O(m α(m,n)) time.

- EMST requires Ω(n log n) time in the worst-case.

  [Linear time reduction from the Closest Pair Problem.]

- EMST in O(n log n) time:

  (1) Compute DT in O(n log n) time (# edges in DT $\leq$ 3n –6).

  (2) Apply Prim or Kruskal MST algorithm to DT.

- **Next we will show EMST can be obtained from DT in only O(n) time.**

Euclidean Minimum Spanning Tree

- D. Cheriton, R.E. Tarjan [1976]

  "Finding minimum spanning trees," SIAM J. Comp. 5(4), 724-742.
- Also appears in §6.1 of [Preparata-Shamos'85].

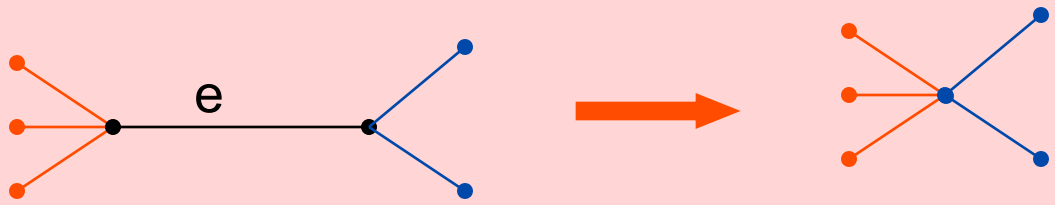
- Cheriton-Tarjan's MST algorithm works on general graphs.

  When applied to a **planar** graph with n vertices and arbitrary edge-weights, it takes only O(n) time.

- The following graph operations preserve planarity:

  (a) vertex or edge removal,

  (b) edge contraction (shrink the edge & identify its two ends):



Cheriton-Tarjan: MST algorithm (overview)

Input: edge-weighted graph G=(V,E)

1. $Q \leftarrow \varnothing$ (* queue of sub-trees *)

   for $v \in V$ **do** enqueue (v, Q) (* n single-node trees in Q *)

2. **while** $|Q| \geq 2$ **do**

   - let T_1 be the tree at the front of Q
   - find edge $(u,v) \in E$ with minimum weight s.t. $u \in T_1$ and $v \notin T_1$
   - let T_2 be the tree (in Q) that contains v
   - $T \leftarrow$ MERGE (T_1 , T_2) by adding edge (u,v)
   - remove T_1 and T_2 from Q
   - add T to the end of Q
   - **CLEAN-UP** after each stage (see next slide)

end

Cheriton-Tarjan

$$\text{stage}(T) = \begin{cases} 0 & \text{if } |T| = 1 \\ 1 + \min\{\text{stage}(T_1), \text{stage}(T_2)\} & \text{if } T = \text{MERGE}(T_1, T_2) \end{cases}$$

Invariants:

(a) stage numbers of trees in Q form a non-decreasing sequence.

(b) stage(T)=j implies T has at least 2^j nodes. So, stage(T) ≤ log |T|.

(c) after completion of stage j (i.e., the first time stage(T) > j, ∀T∈Q) there are ≤ $n/(2^j)$ trees in Q.

CLEAN-UP:

After the completion of each stage do "clean-up", i.e., shrink G to G*, where G* is G with each edge in the same tree contracted, i.e., each tree in Q is contracted to a single node, with only those edges (u,v)∈G* , u∈T, v∈T', That are shortest incident edges between disjoint trees T, T'.

Cheriton-Tarjan: Algorithm

PROCEDURE MST of a Graph G=(V,E)

1. $Q \leftarrow \varnothing$ (* initialize queue *)

   for $v \in V$ **do** stage(v) $\leftarrow$ 0 ; enqueue (v, Q)

   $j \leftarrow 1$

2. **while** $|Q| \geq 2$ **do**

   - let T_1 be the tree at the front of Q
   - **if** stage(T_1) = j **then** **CLEAN-UP**; $j \leftarrow j+1$
   - (u,v) $\leftarrow$ shortest edge, s.t. $u \in T_1$ and $v \notin T_1$
   - let T_2 be the tree (in Q) that contains v
   - $T \leftarrow$ MERGE (T_1 , T_2) by adding edge (u,v)
   - stage(T) $\leftarrow$ 1 + min{ stage(T_1), stage(T_2)}
   - remove T_1 and T_2 from Q
   - add T to the end of Q

end

Cheriton-Tarjan: Analysis

FACTS:

(a) A planar graph with m vertices has O(m) edges.

(b) Shrunken version of a planar graph is also planar

(c) CLEAN-UP of stage j takes $O(n/2^j)$ time.

(d) # stages $\leq \lceil \log n \rceil$

(e) During stage j each of the $< 3n/2^j$ edges of G* are checked at most twice (once from each end). So, stage j takes $O(n/2^j)$ time.

(f) Cheriton-Tarjan's algorithm on planar graphs takes:

$$\text{Total Time} = O\left(\sum_{j=1}^{\lceil \log n \rceil} \frac{6n}{2^j} \right) = O(n)$$

THEOREM: The MST of any weighted connected planar graph with n vertices can be computed in optimal O(n) time.

COROLLARY: Given DT(P) of a set P of n points in the plane, the following can be constructed in O(n) time:

(a) GG(P),

(b) RNG(P),

(c) EMST(P),

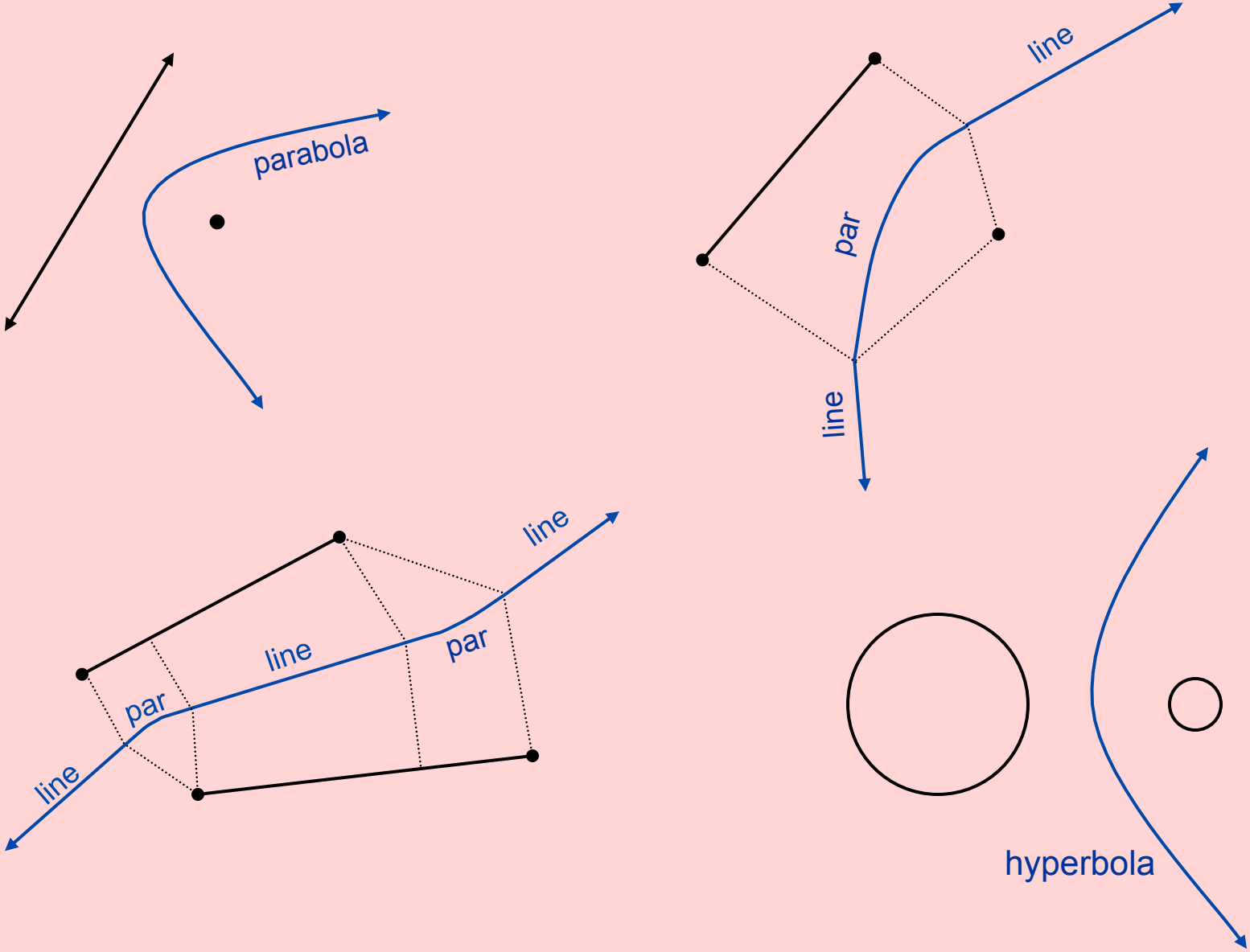
(d) NNG(P).

Proof: (a) & (d): obvious. (c): use Cheriton-Tarjan on DT(P). (b): see Exercise.

Extensions of Voronoi Diagrams

- Voronoi Diagram of line-segments, circles, …
- Medial Axis
- Order k Voronoi Diagram
- Farthest Point Voronoi Diagram (order n-1 VD)
- Weighted Voronoi Diagram & Power Diagrams
- Generalized metric (e.g., L_p metric)

VD of line-segments & circles



Higher Order VD

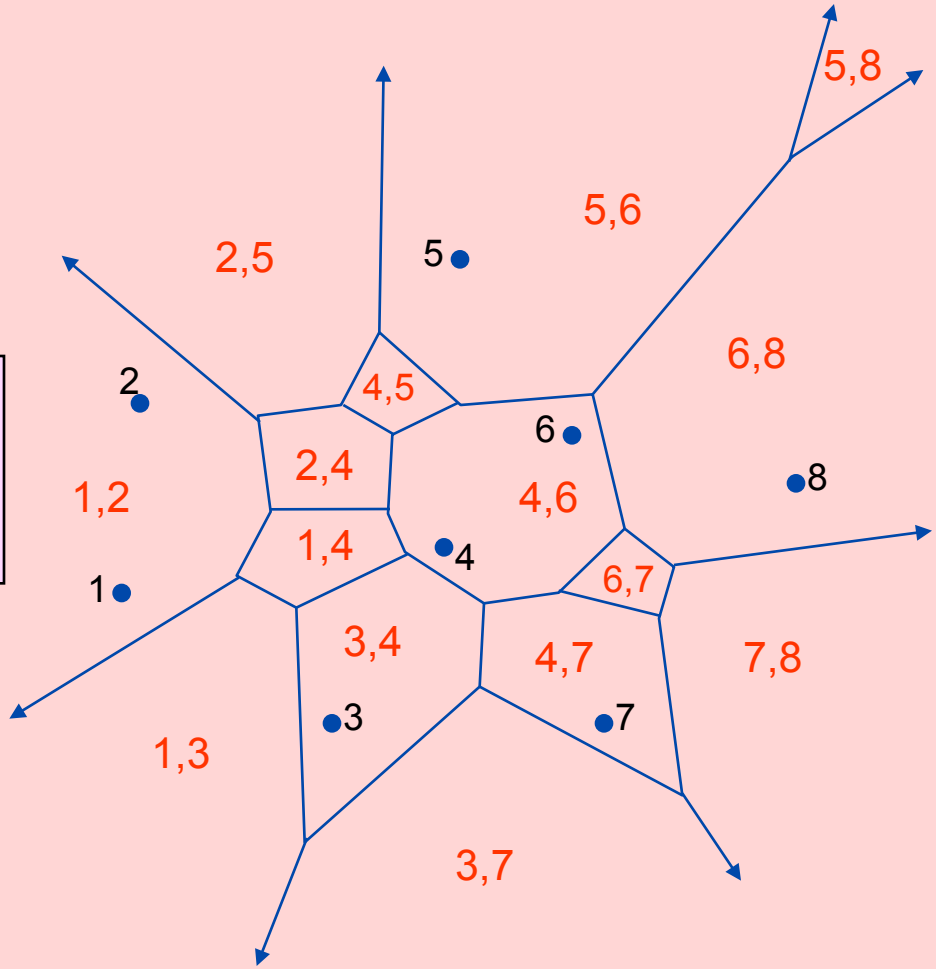
- [PrS85] § 6.3
- [Ede87] § 13.3-13.5
- [ORo98] § 6.6.
- Aurenhammer [1987], "Power diagrams: properties, algorithms, and applications," SIAM J. Computing 16, 78-96.
- D.T. Lee [1982], "On k-Nearest Neighbors Voronoi Diagram in the Plane," IEEE Trans. Computers, C-31, 478-487.

Order k Voronoi Diagram

Given a set of n sites in space, partition the space into regions where any two points belong to the same region iff they have the same set of k nearest sites.

Example: Order 2 Voronoi Diagram

Some Voronoi regions of order 2 are empty, e.g. (5,7).

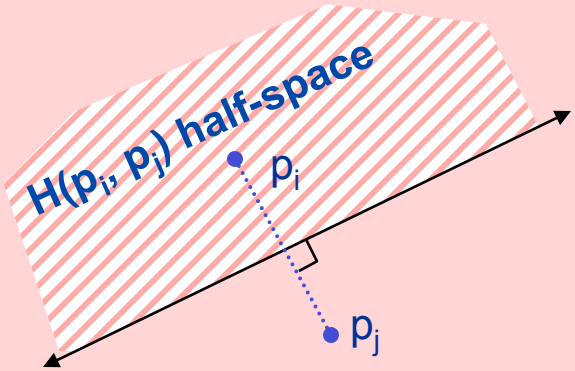


Order k Voronoi Diagram

$$T \subseteq P = \{p_1, p_2, \cdots, p_n\}, \quad |T| = k$$

$$V_k(T) = \bigcap_{\substack{p_i \in T \\ p_j \in P - T}} H(p_i, p_j)$$

Voronoi region of T (a convex polyhedron, possibly empty)



Order k Voronoi Diagram of P: $VD_k(P) = \bigcup_{\substack{T \subseteq P \\ |T| = k}} \{V_k(T)\}$

$\binom{n}{k}$ possible subsets of size k. Most have empty Voronoi regions.

In 2D only O(k(n-k)) of them are non-empty [D.T. Lee'82].

ALGORITHM Order-k VD of $P \subseteq \Re^2$

1. For each point $p \subseteq P$ do
   - lift p onto Λ: $z = x^2 + y^2$, call it $\lambda(p)$
   - $\Pi(p) \leftarrow$ plane tangent to Λ at $\lambda(p)$
2. Construct k-belt of the arrangement of the planes $\Pi(p)$, $p \subseteq P$, in $\Re^3$.
3. Project down this k-belt onto the base plane $\Re^2$.
   This is the k-th order Voronoi Diagram of P.

FACT 1: In $O(n^3)$ time we can compute all k-levels of the arrangement and find the k-th order VD.

Improvement by [D.T. Lee 1982]:

FACT 2: [Preparata-Shamos'85]:
k-th order VD of n points in the plane can be obtained in time $O(\min \{ k^2 , (n-k)^2 \} n \log n)$.

COROLLARY 3: Complexity of the **k nearest neighbors** query problem is:

$O(k + \log n)$	Query Time
$O(k^2 n \log n)$	Preprocessing Time
$O(kn)$	Space.