Voronoi Diagrams and Delaunay Triangulation

slides by Andy Mirzaian

(a subset of the original slides are used here)

Voronoi Diagram & Delaunay Triangualtion

Algorithms

- *Divide-&-Conquer*
- *Plane Sweep*
- *Lifting into d+1 dimensions*
- *Edge-Flip*
- *Randomized Incremental Construction*

Applications

- *Proximity space partitioning and the post office problem*
- *Height Interpolation*
- *Euclidean: Minimum Spanning Tree, Traveling Salesman Problem,*
- *Minimum Weight Triangulation, Relative Neighborhood Graph, Gabriel Graph.*

Extensions

- *Higher Order Voronoi Diagrams*
- *Generalized metrics - Robot Motion Planning*

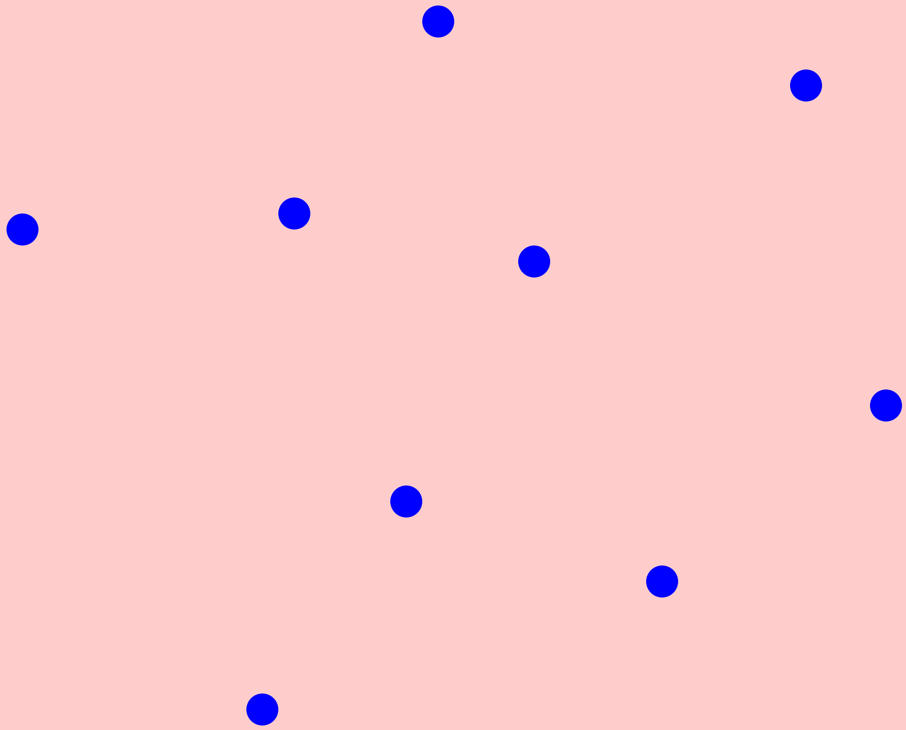
References:

- [M. de Berge et al] chapters 7, 9, 13
- [Preparata-Shamos'85] chapters 5, 6
- [O'Rourke'98] chapter 5
- [Edelsbrunner'87] chapter 13

Introduction

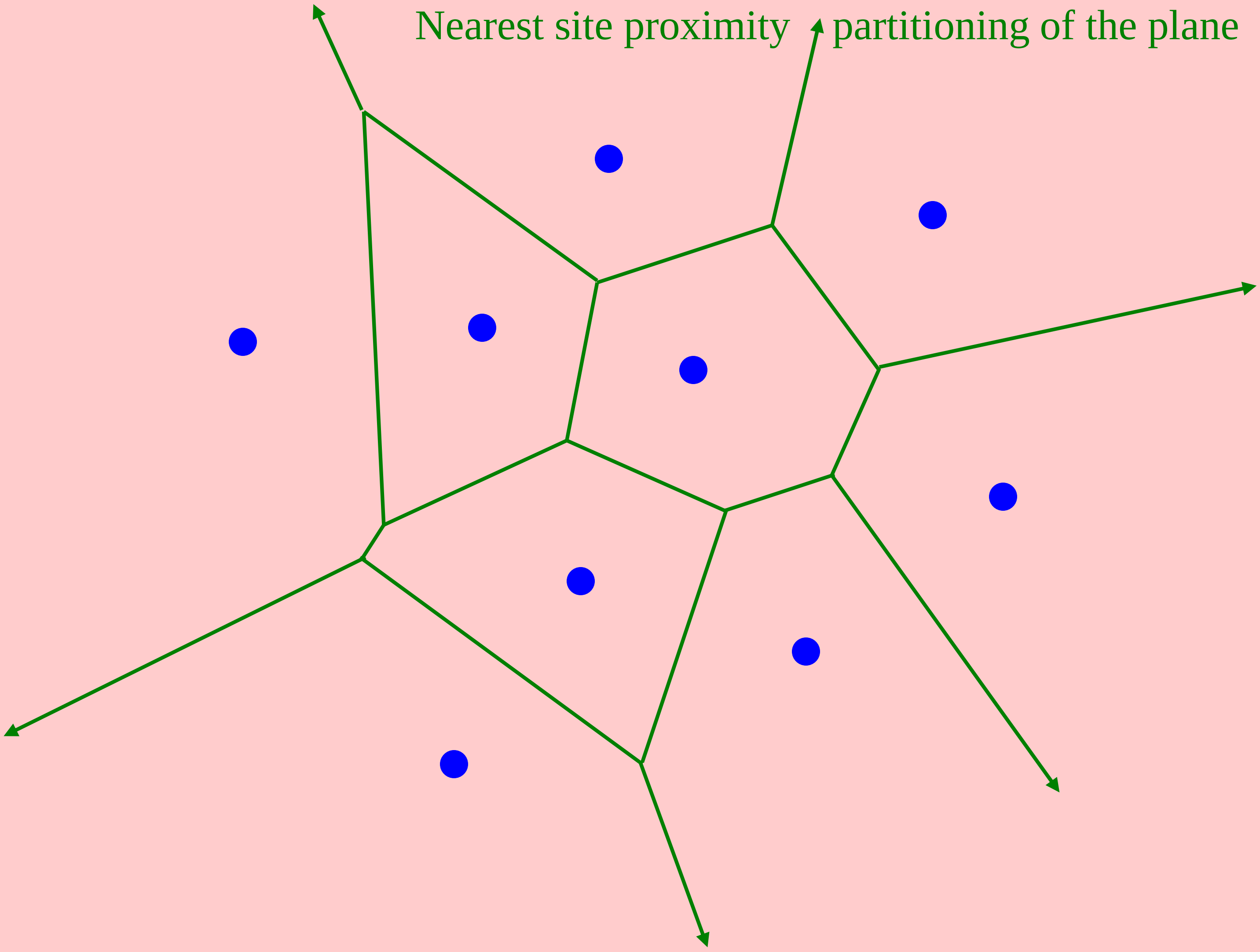
Voronoi Diagram & Delaunay Triangulation

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



Voronoi Diagram & Delaunay Triangulation

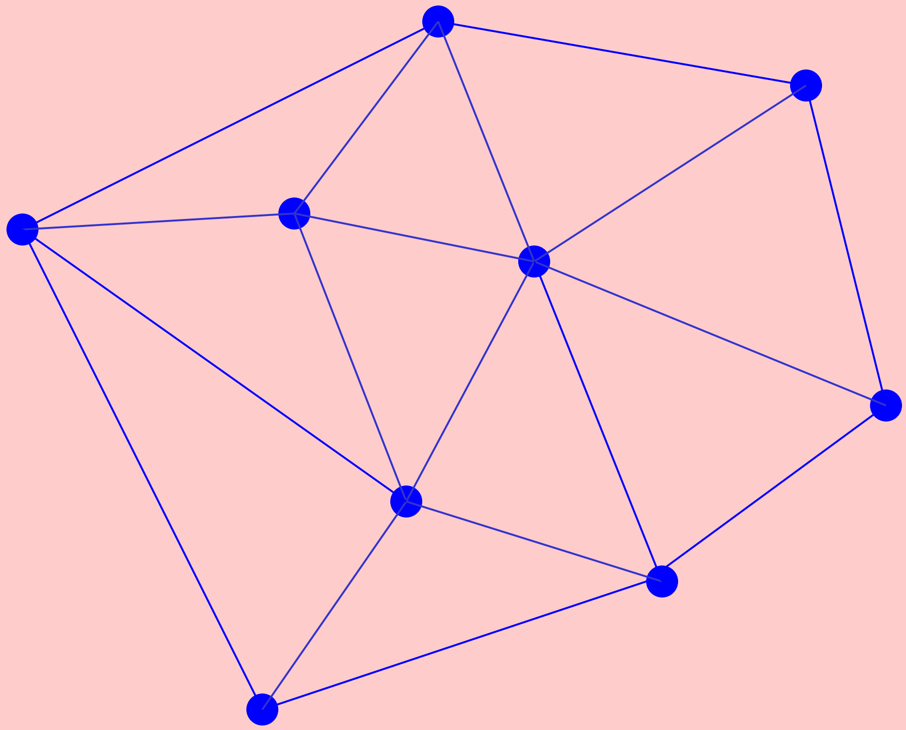
Nearest site proximity partitioning of the plane



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

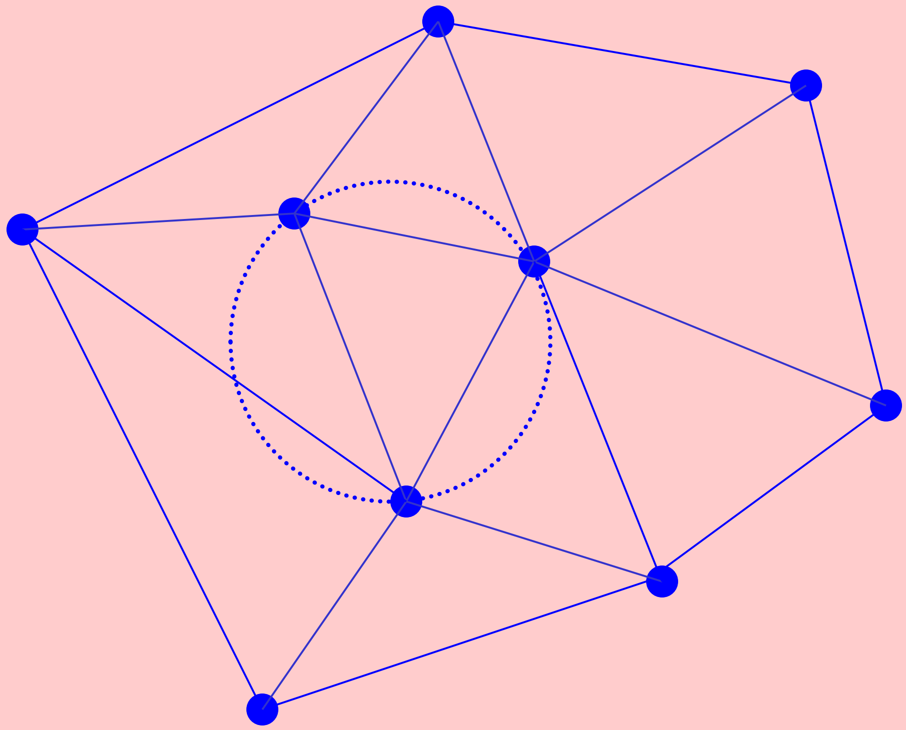
Voronoi Diagram & Delaunay Triangulation

Delaunay Triangulation = Dual of the Voronoi Diagram.



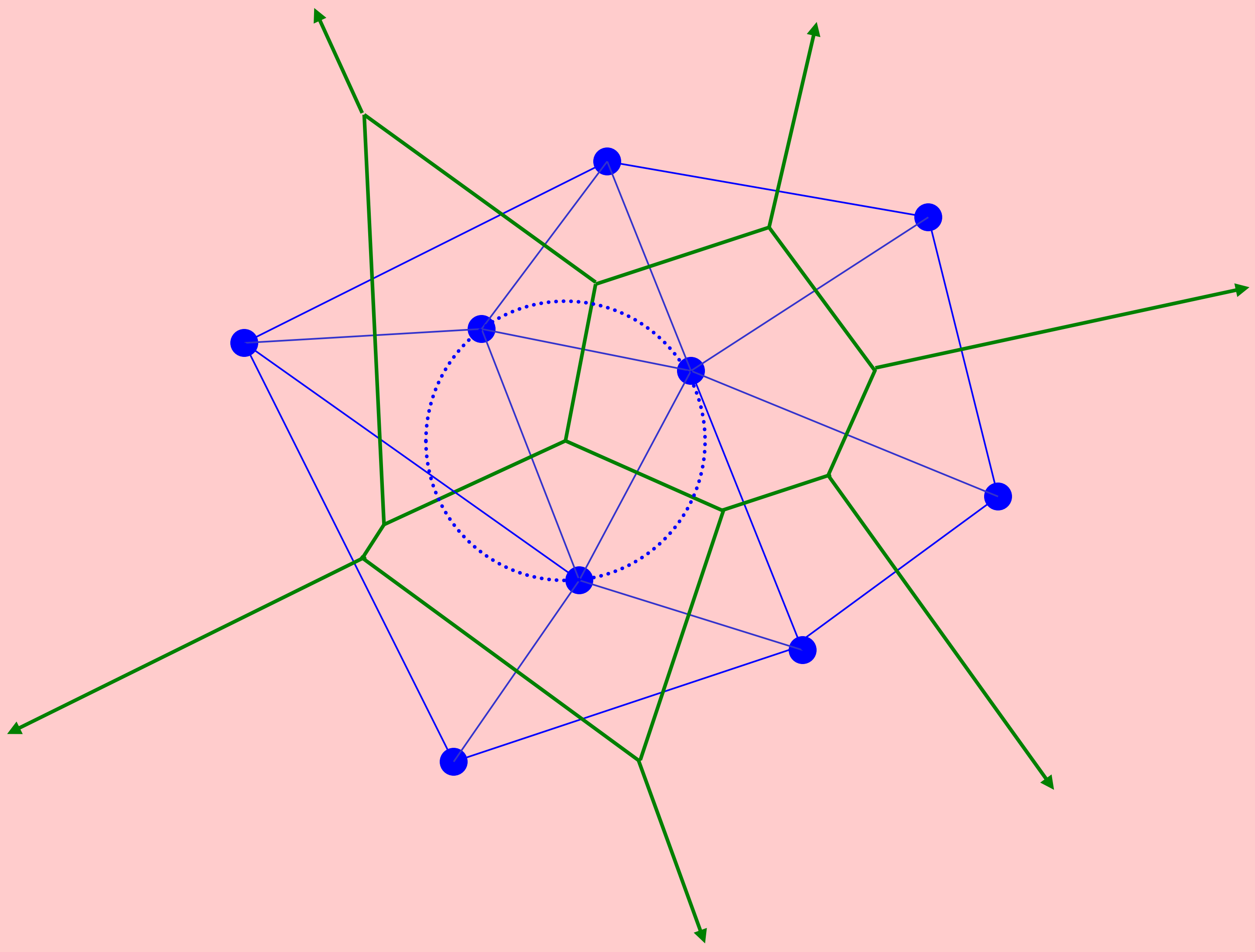
DT(P): # vertices = n, # edges ≤ 3n-6, # triangles ≤ 2n-5.

Voronoi Diagram & Delaunay Triangulation



Delaunay triangles have the "empty circle" property.

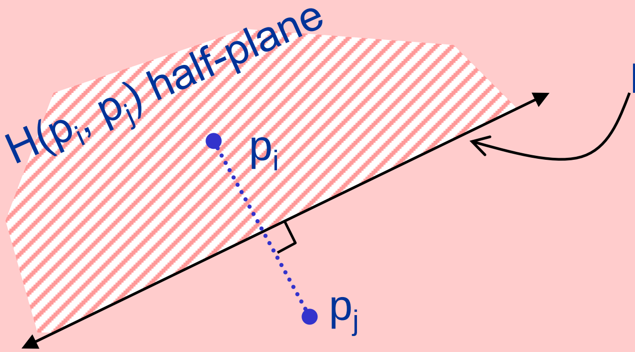
Voronoi Diagram & Delaunay Triangulation



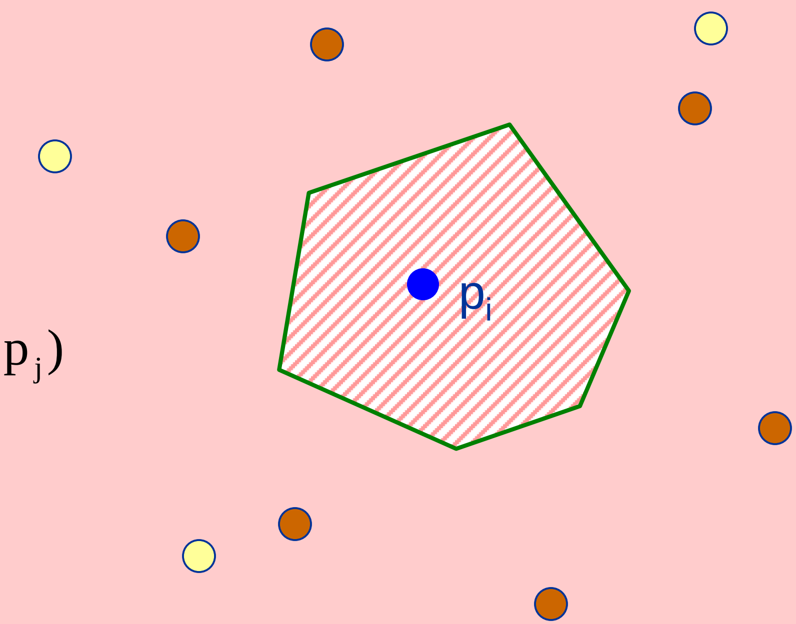
Voronoi Diagram

$P = \{ p_1, p_2, \ldots, p_n\}$ 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.

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.

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: $P = \{ p_1, p_2, \ldots , p_n\}$ 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

$L = \{ p_1, \ldots , p_m\}$ & $R = \{ p_{m+1}, \ldots , p_n\}$.

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).

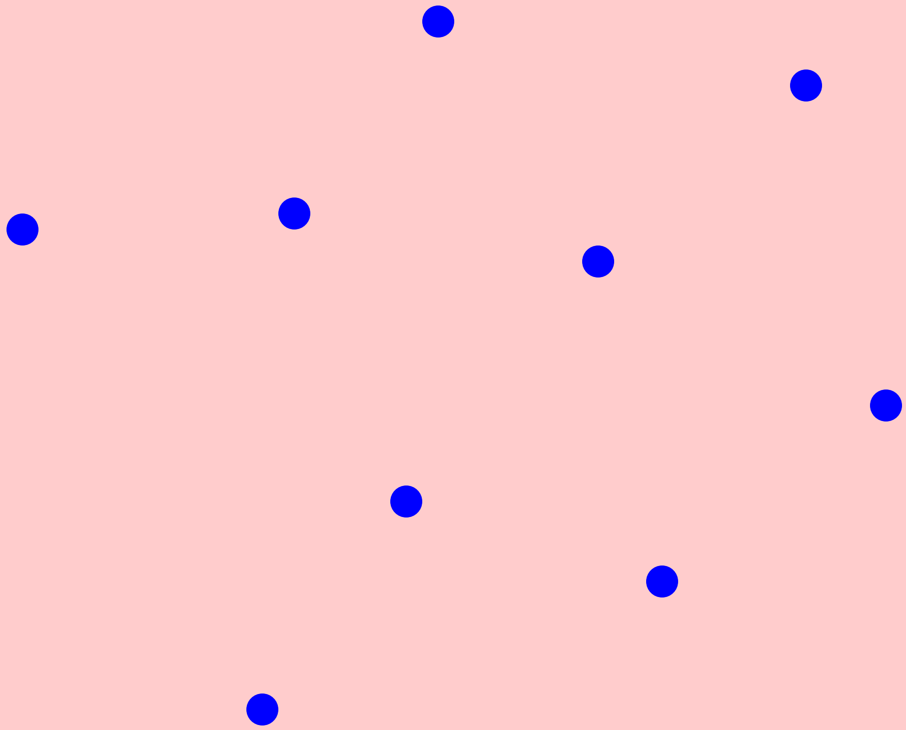
O(n) (c) $VD(P) \leftarrow [C] \cup [VD(L)$ to the left of $C] \cup [VD(R)$ to the right of $C]$.

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

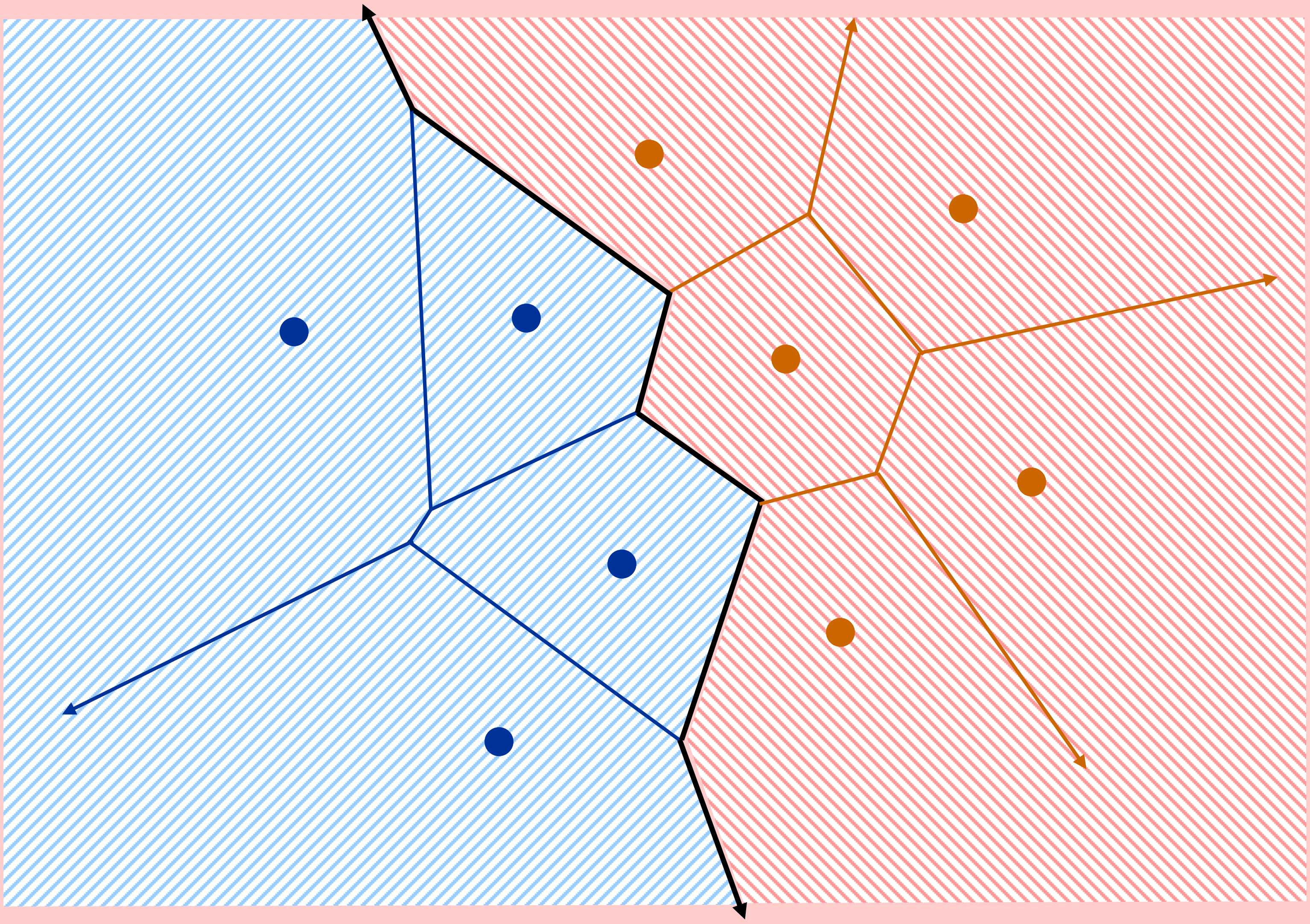
END.

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

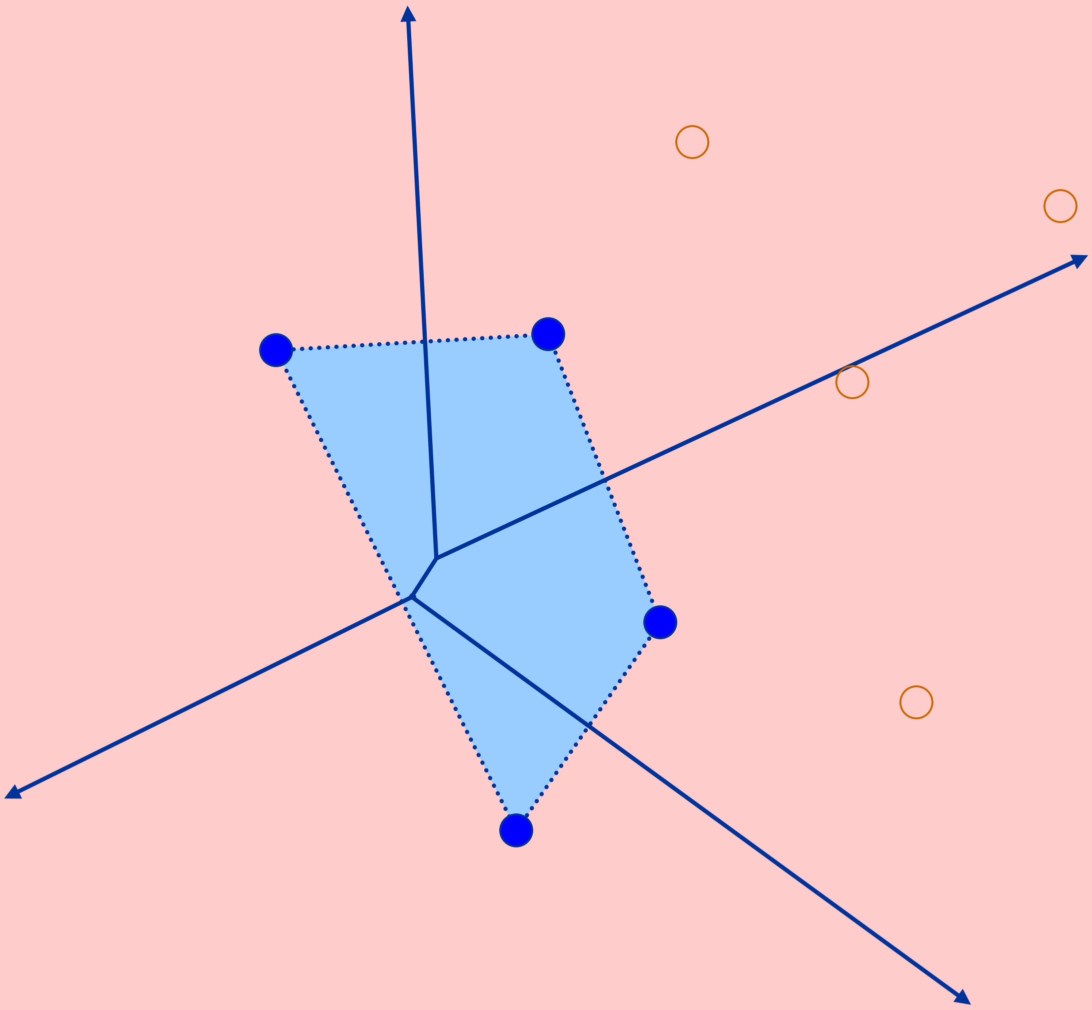
P = { p_1, p_2, … , p_n} a set of n points in the plane.



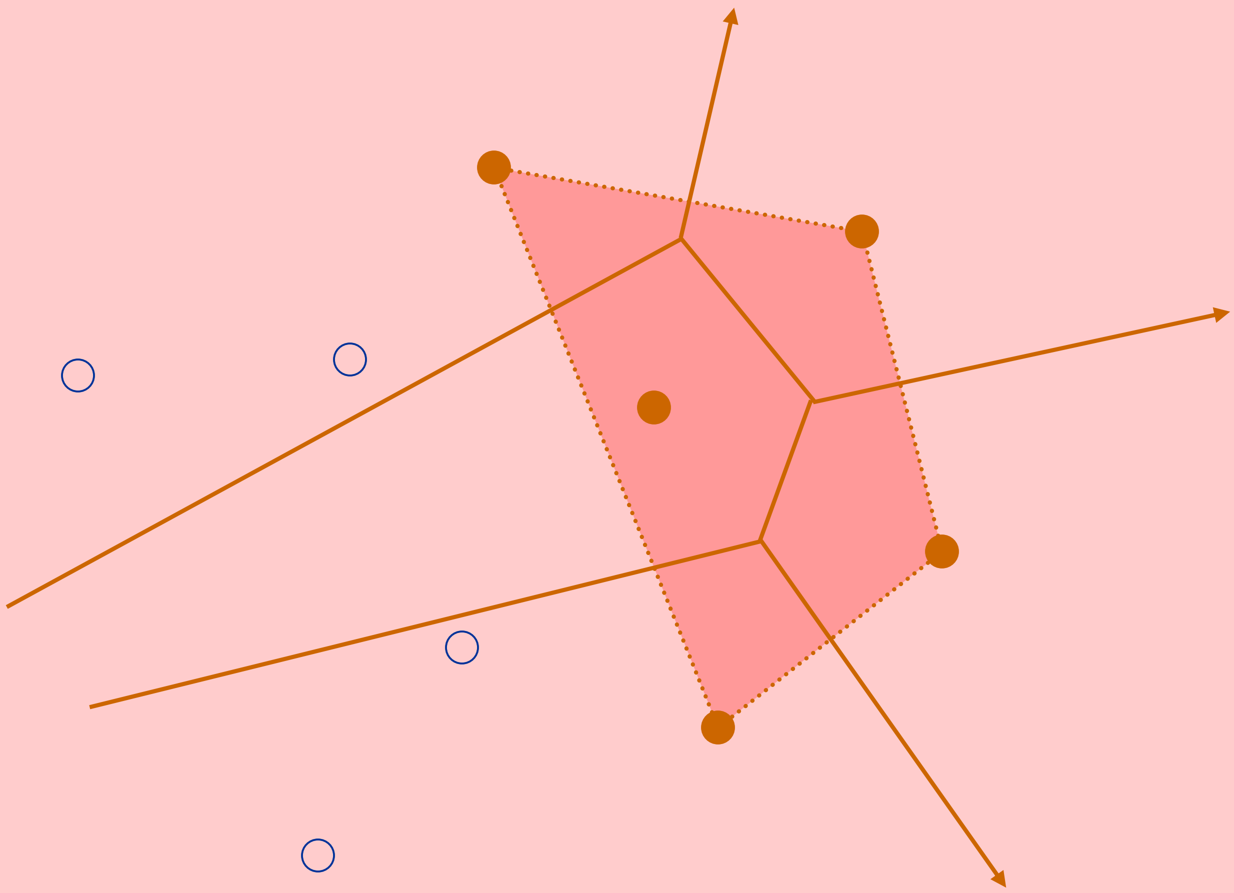
VD(P) = [C] ∪ [VD(L) to the left of C] ∪ [VD(R) to the right of C] .



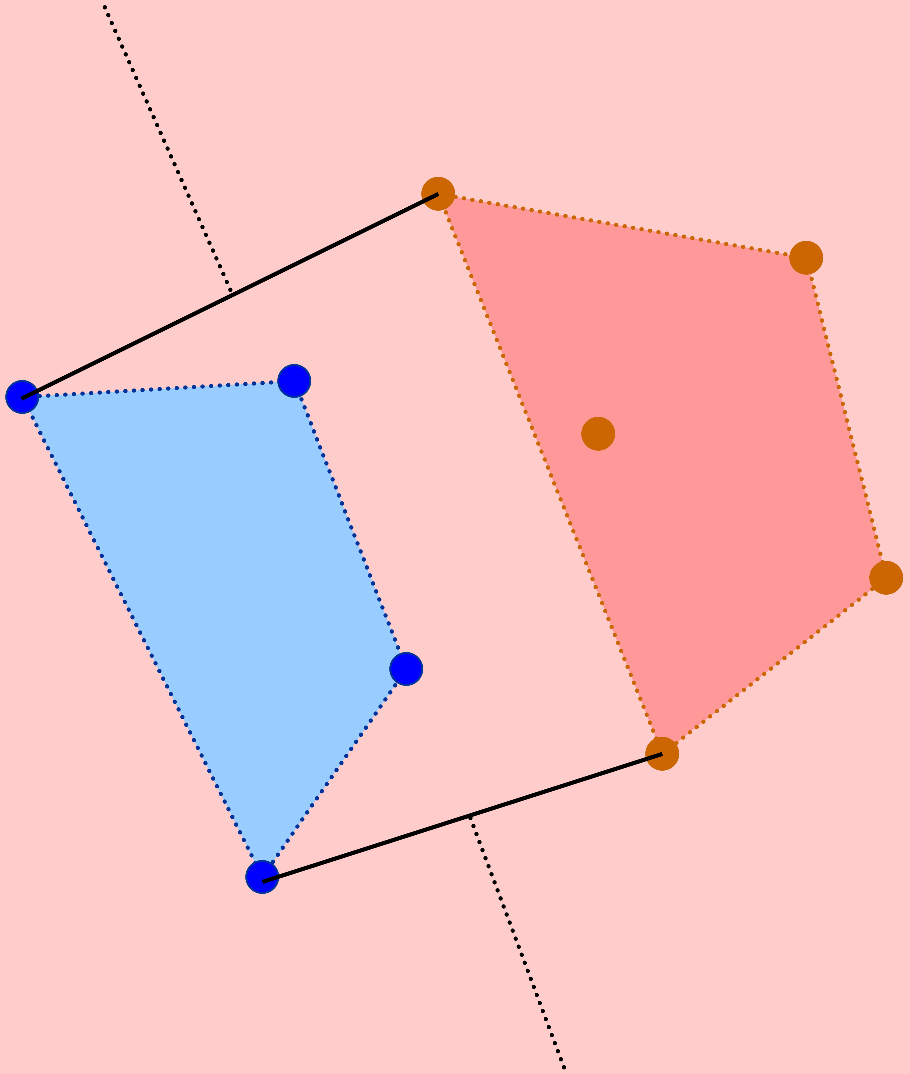
VD(L) and CH(L)



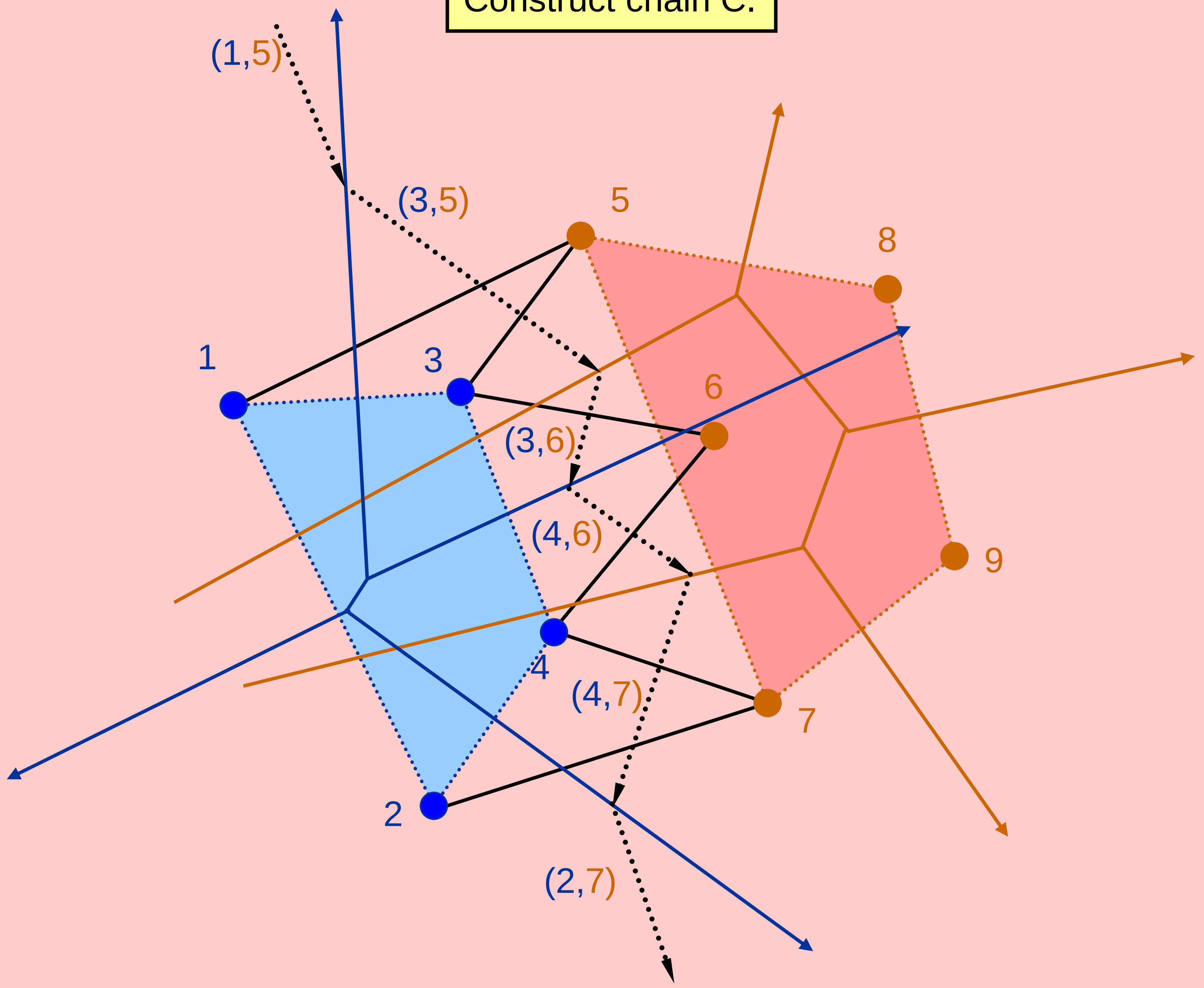
VD(R) and CH(R)



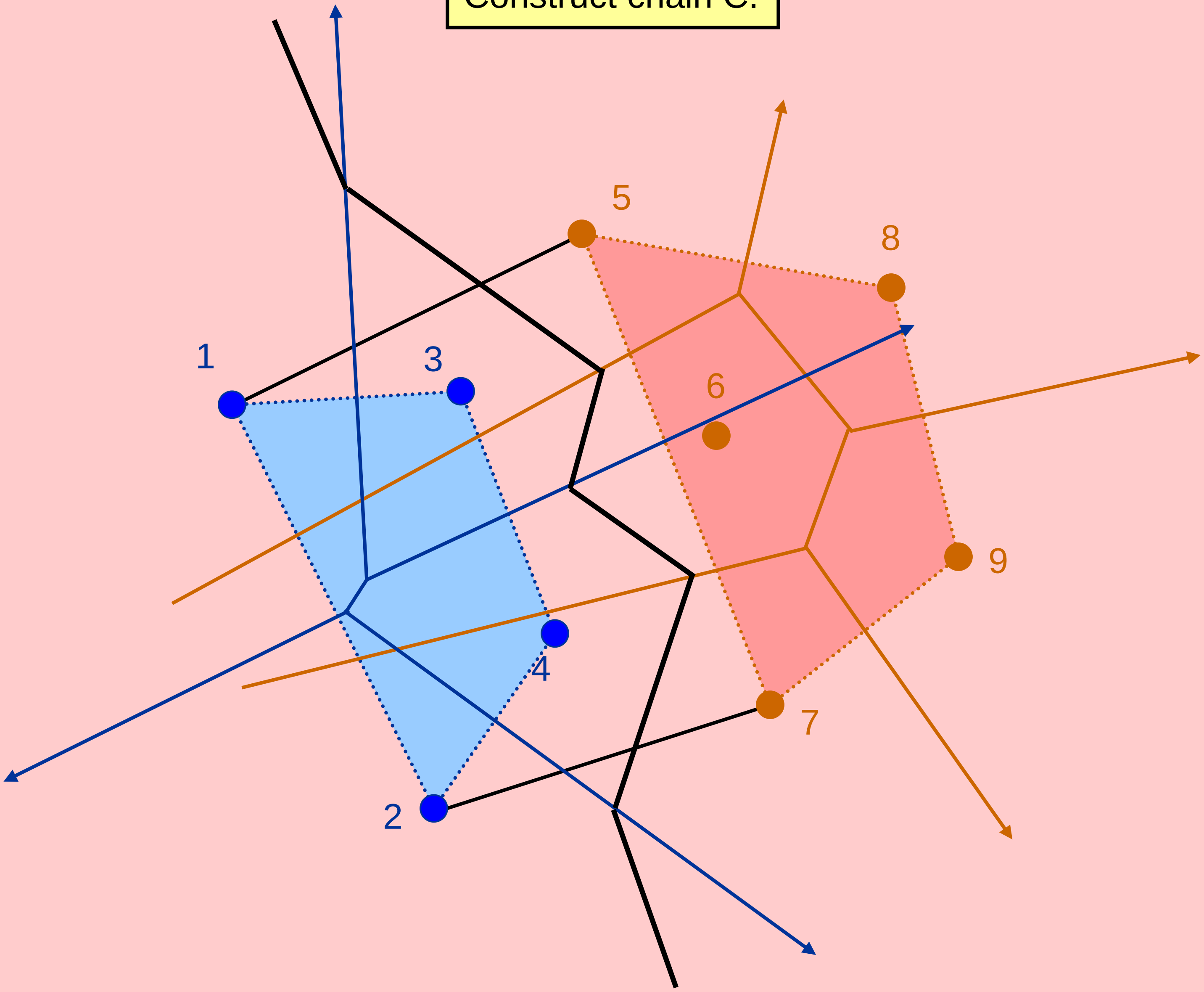
Upper & Lower bridges between CH(L) and CH(R) & two end-rays of chain C.



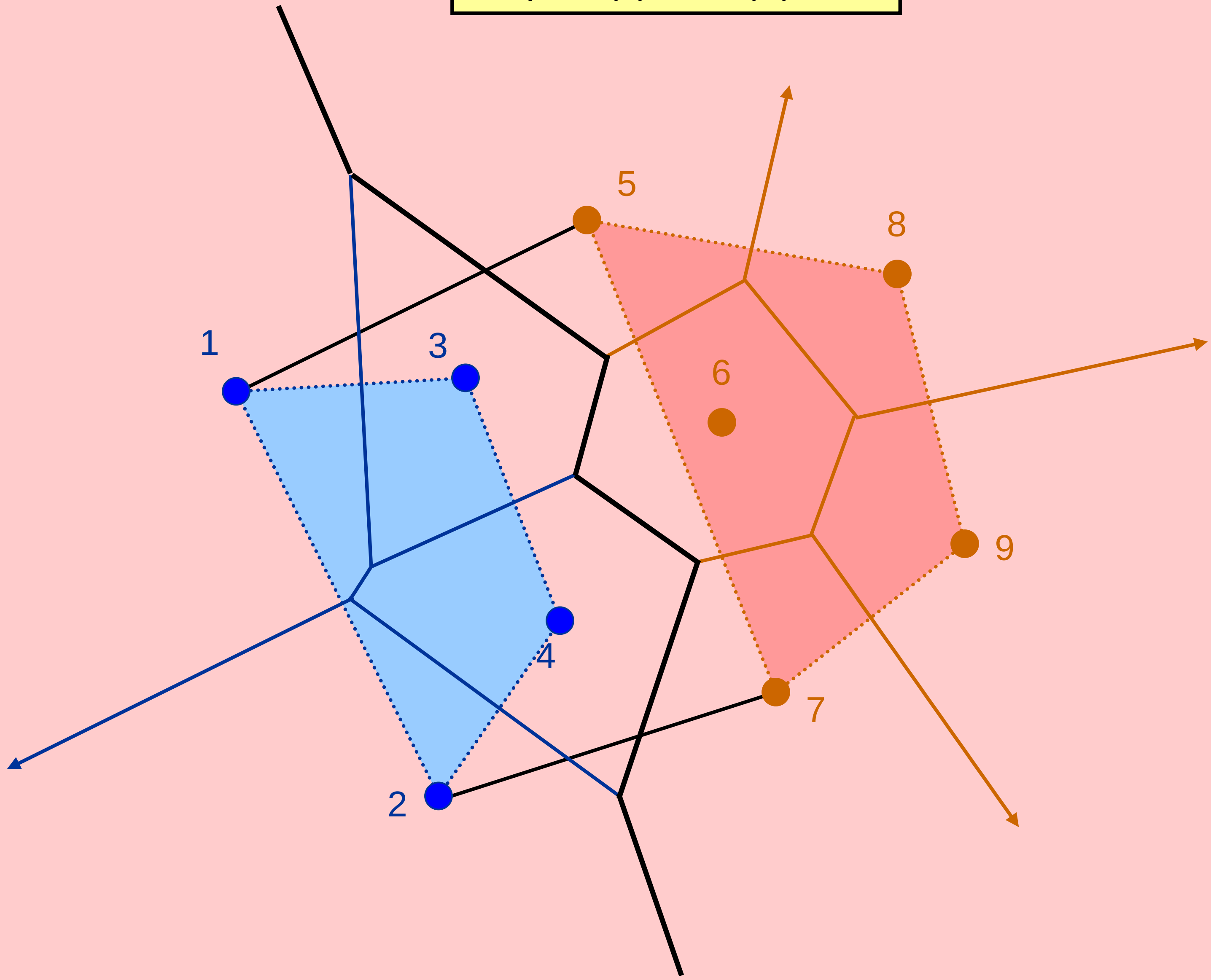
Construct chain C.



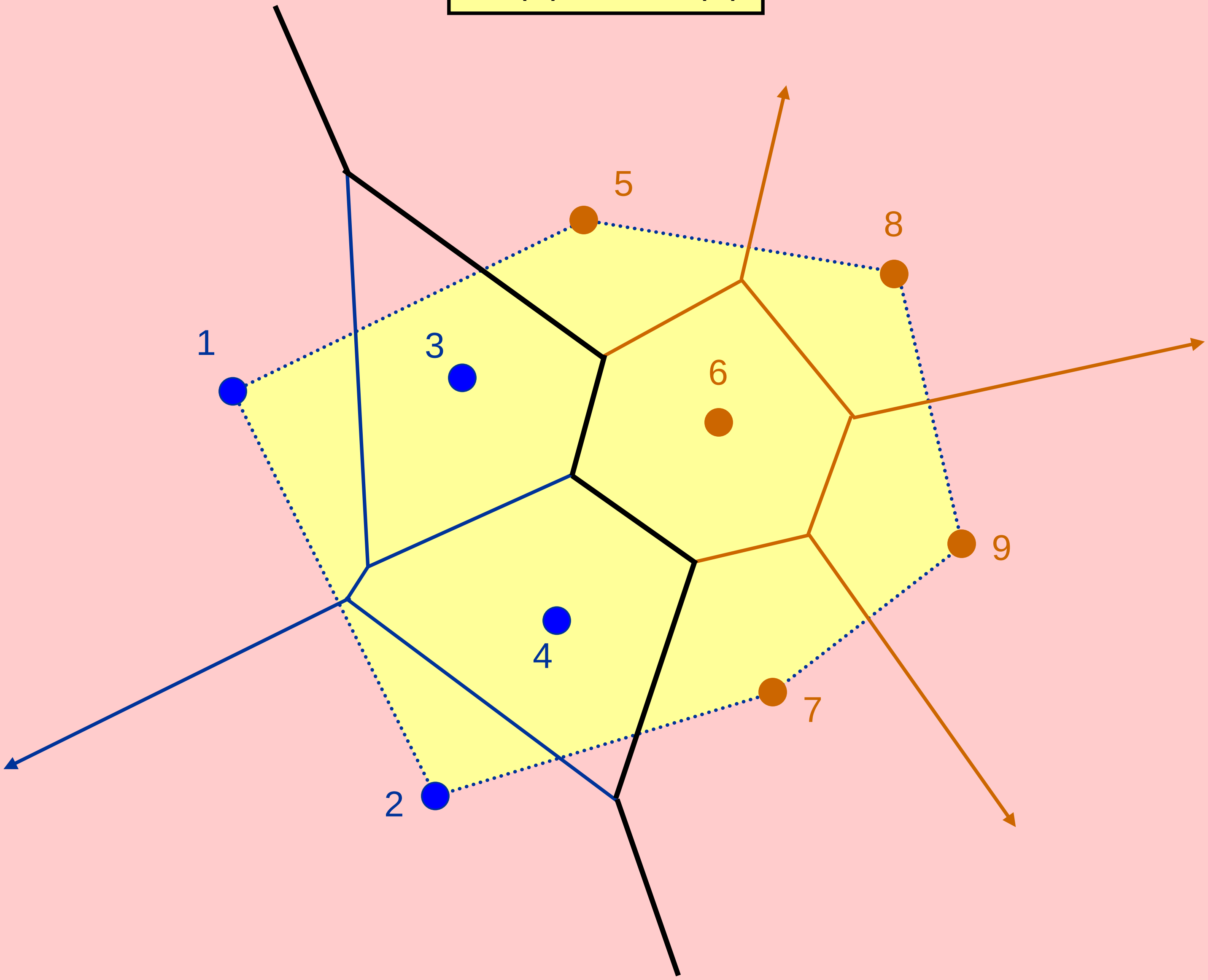
Construct chain C.



Crop VD(L) & VD(R) at C.



VD(P) and CH(P)



Fortune's Algorithm

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

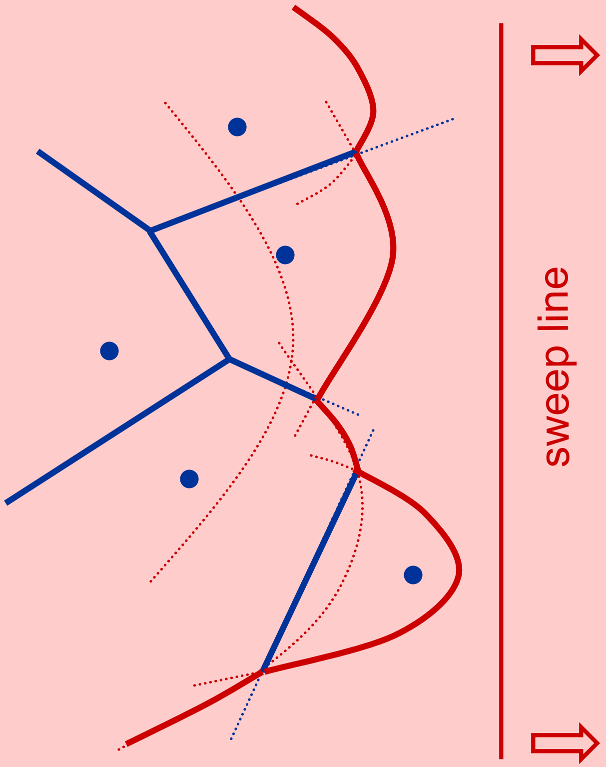
- Guibas, Stolfi [1987],
"Ruler, Compass and computer: The design and analysis of geometric algorithms,"
Proc. of the NATO Advanced Science Institute, series F, vol. 40:
Theoretical Foundations of Computer Graphics and CAD, 111-165.

- O(n log n) time algorithm by plane-sweep.
- See AAW animation.
  - http://www.cse.yorku.ca/~aaw/GregoryFine/applet.html
- Generalization: VD of line-segments and circles.

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 = - ∞ to x = + ∞ .

- 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.

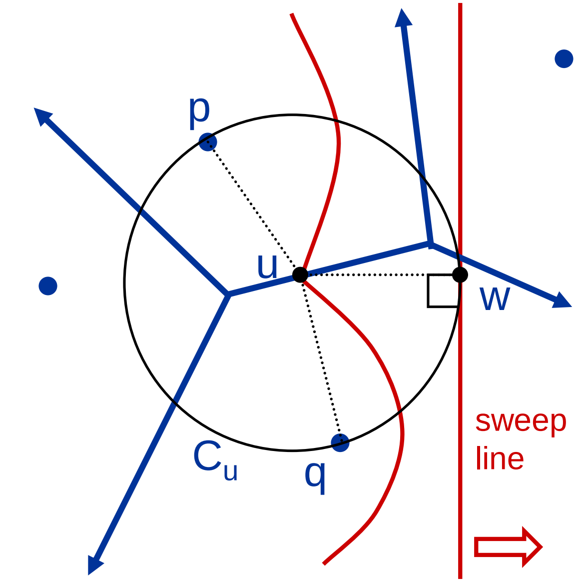
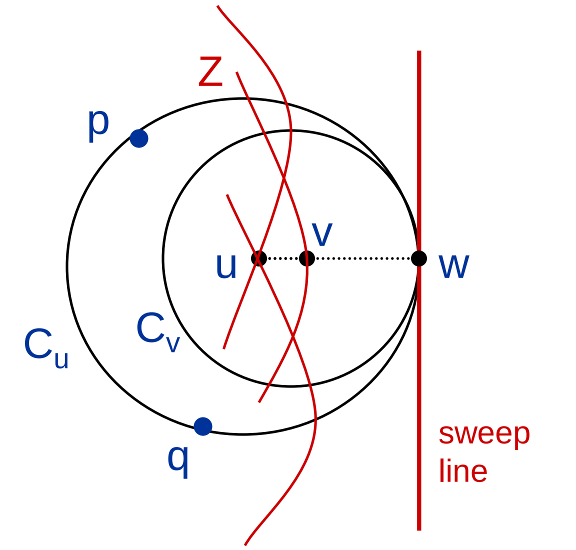


Fig 2.

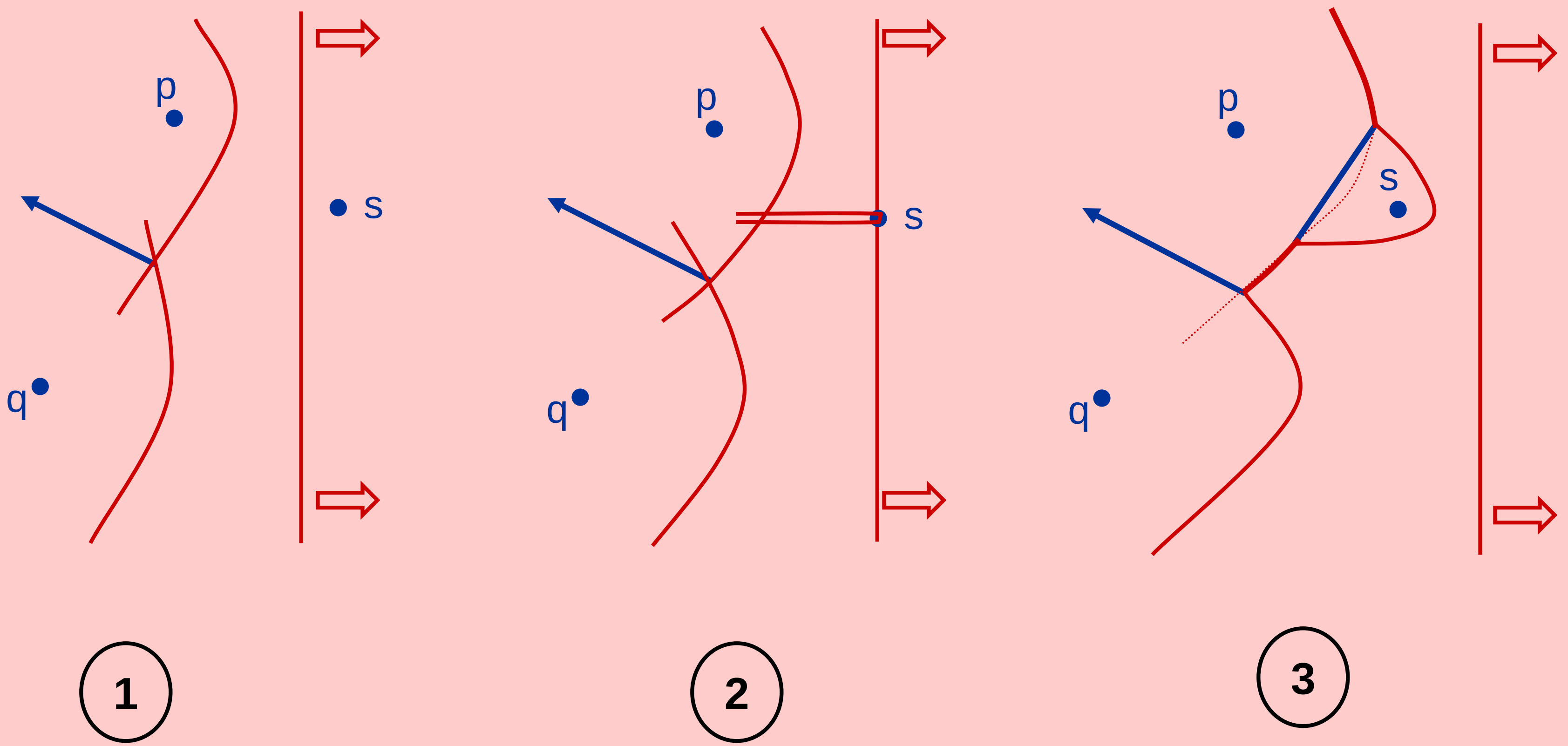


The Discrete Events

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

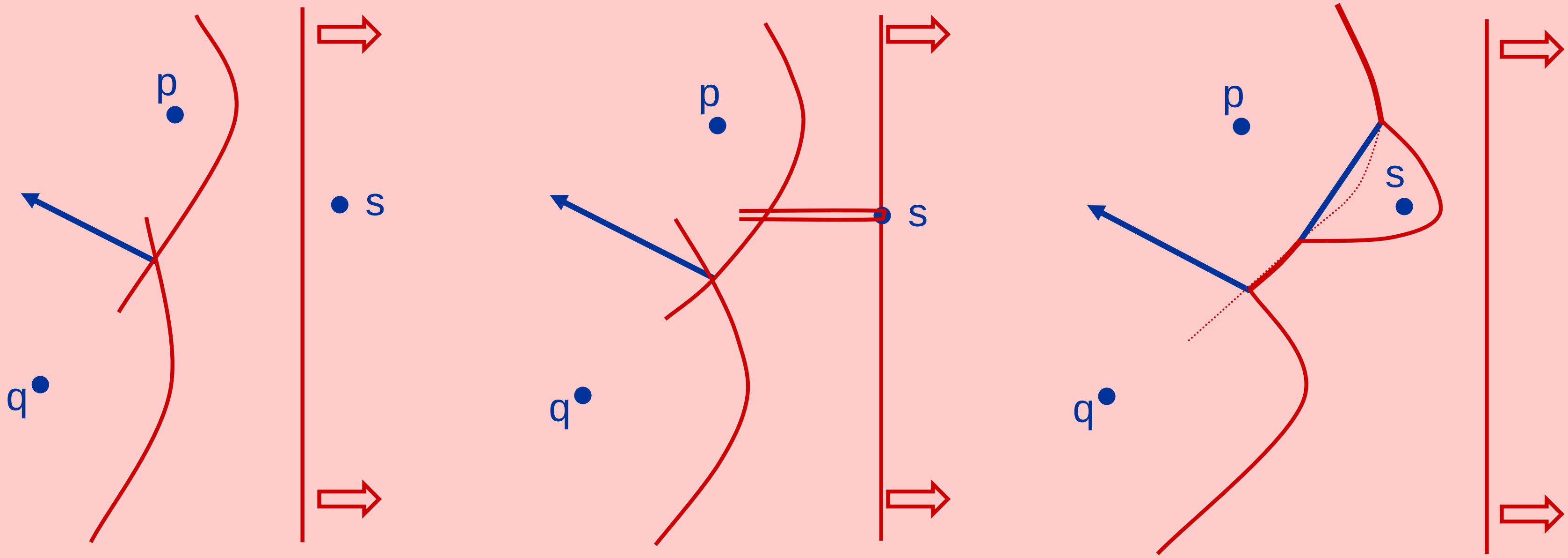
SITE EVENT

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.

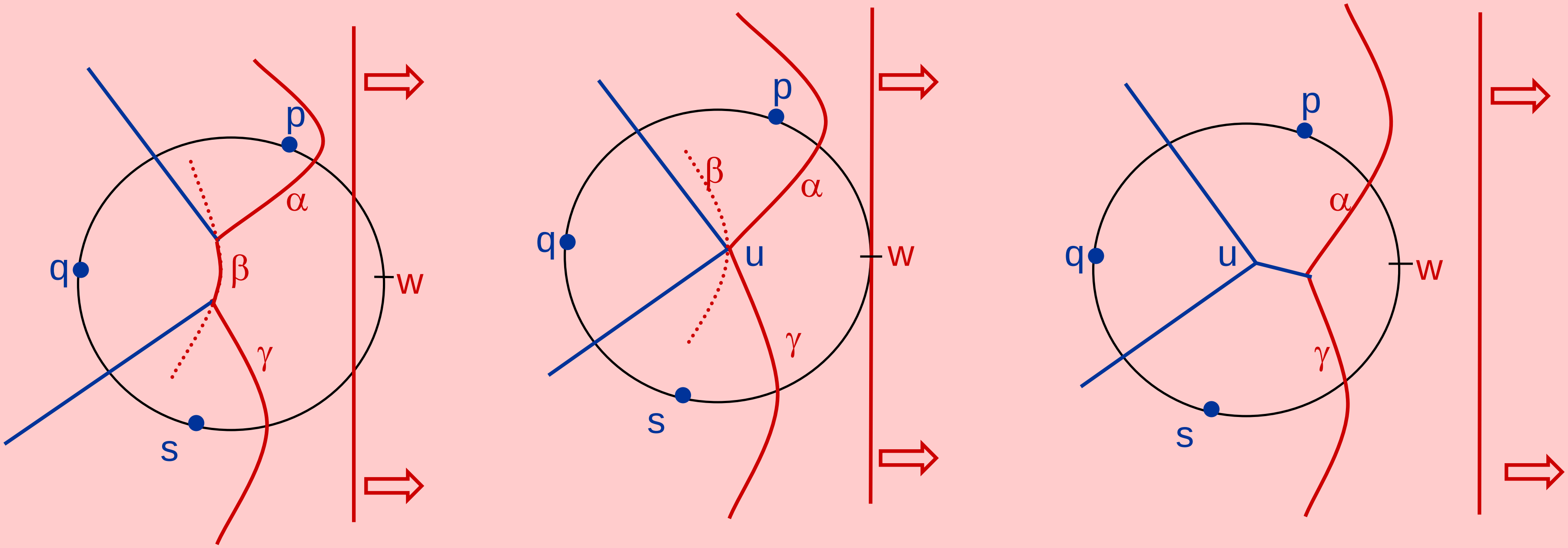


A parabola cannot appear on the front by breaking through from behind.

The following are impossible:

CIRCLE EVENT

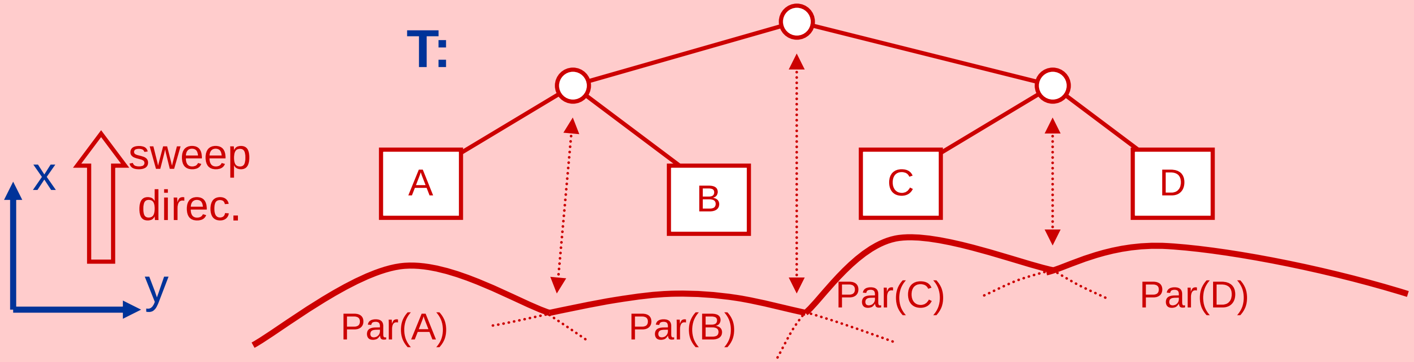
- Circle event w causes parabolic arc β to disappear.
- α and γ cannot belong to the same parabola.



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:

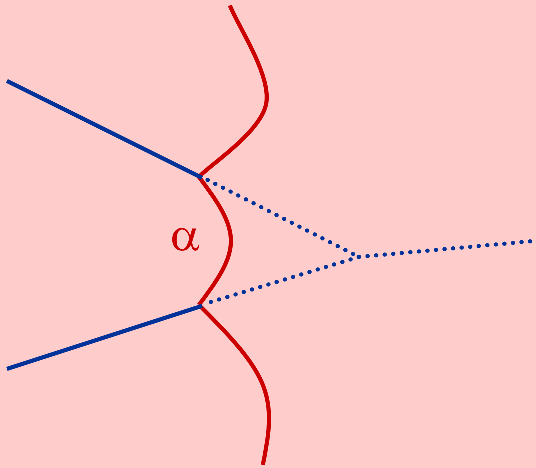
At each iteration remove the next event (with min x-coordinate) from Q & simulate the effect of the sweep-line advancing past that event point.

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.



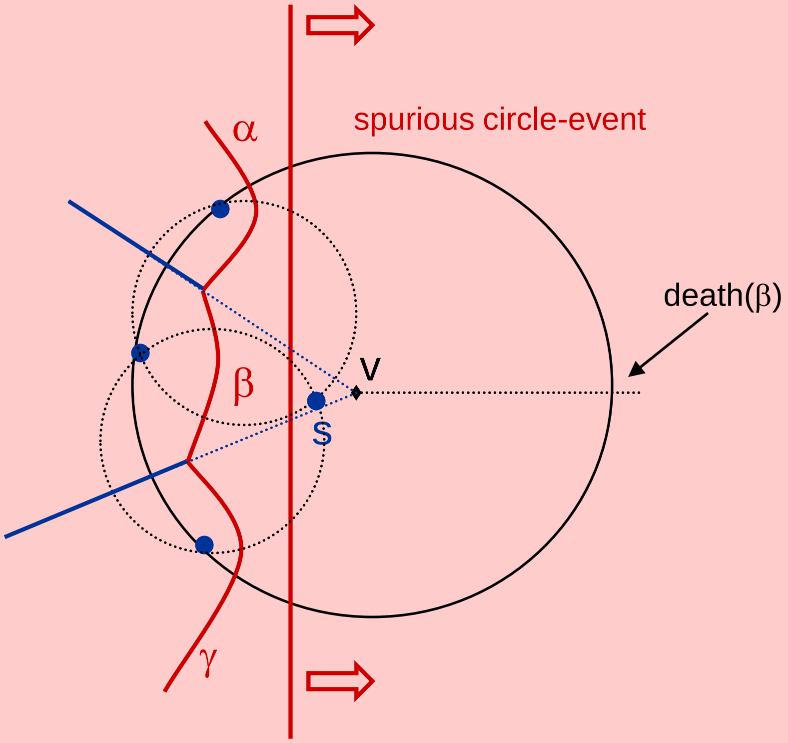
Event Processing & Scheduling

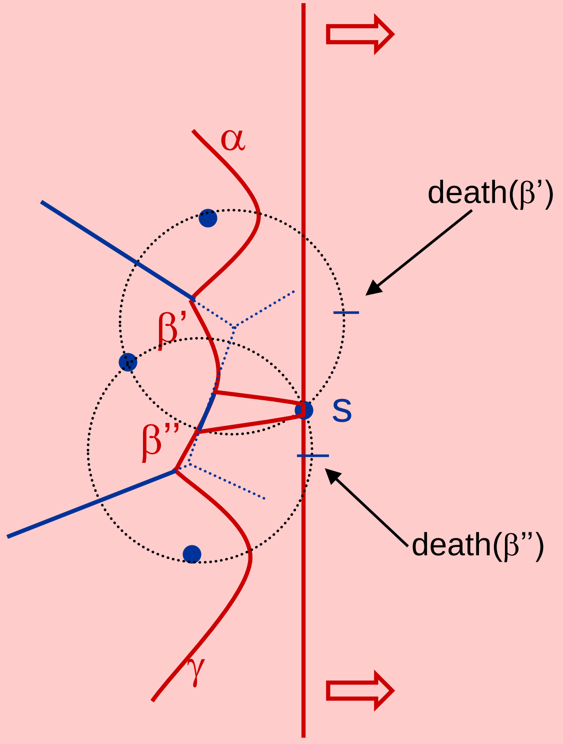
Event-driven simulation loop:

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.





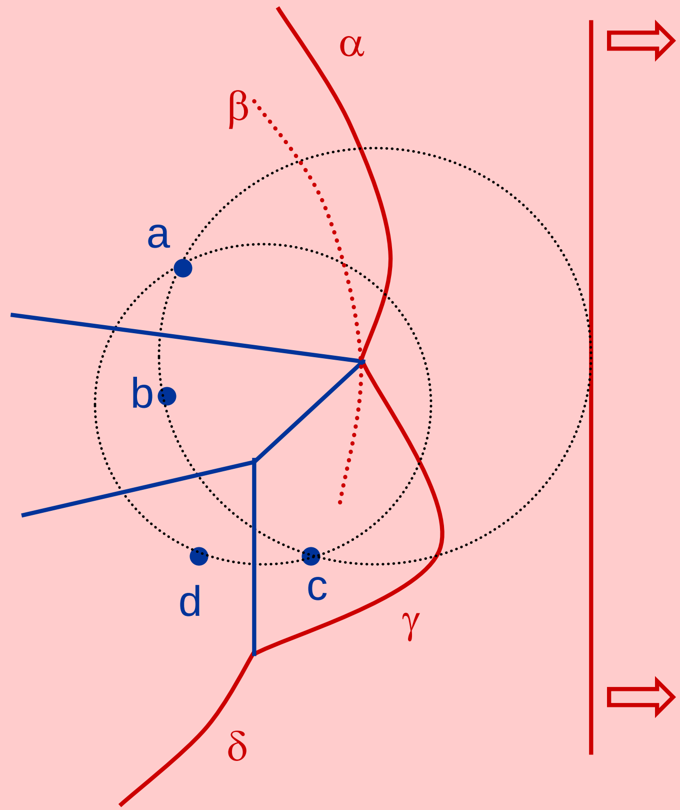
Event Processing & Scheduling

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.]

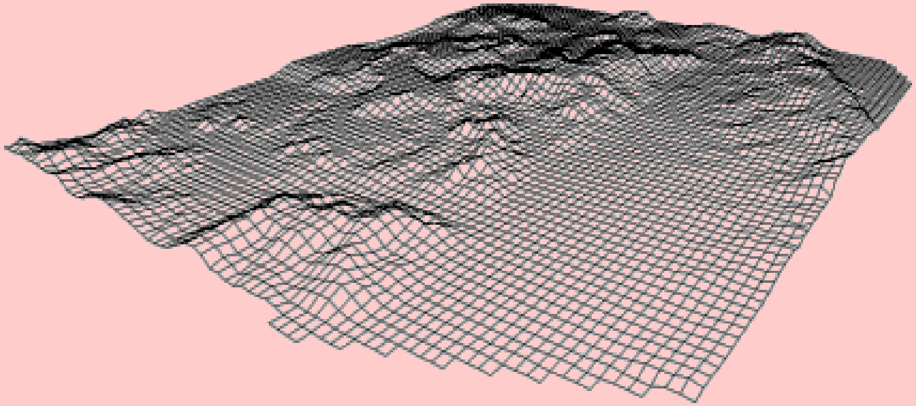
|Q| = O(n) : there are at most n site-events and O(n) triples of consecutive arcs on the parabolic front to define circle-events.

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

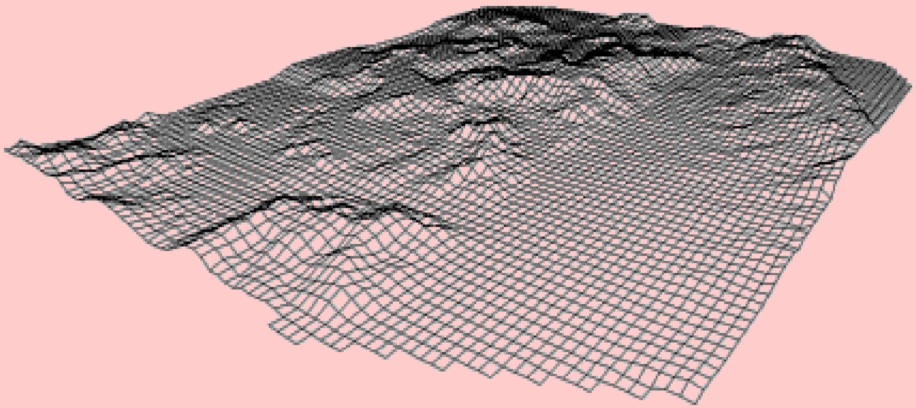
Terrain Height Interpolation



A perspective view of a terrain.

A topographical map of a terrain.

Terrain Height Interpolation



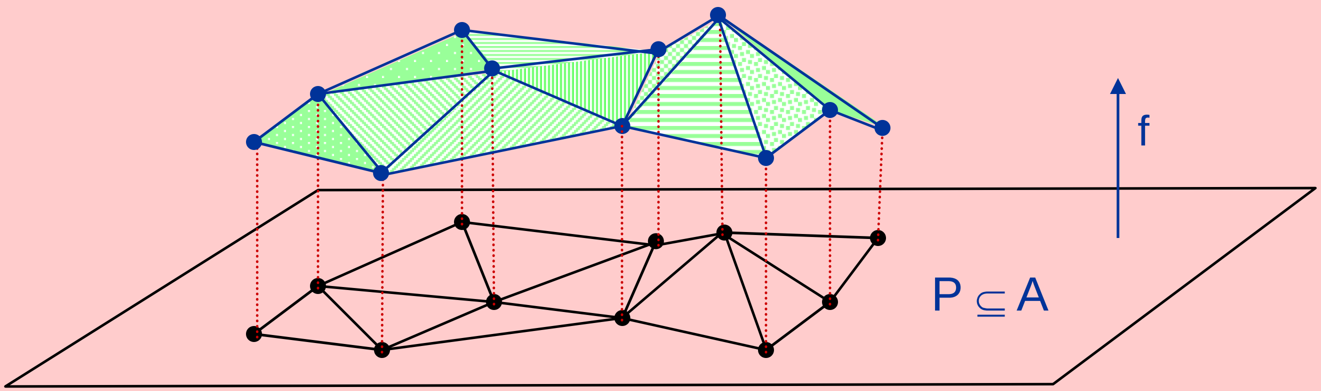
A perspective view of a terrain.

A topographical map of a terrain.

Terrain: A 2D surface in 3D such that each vertical line intersects it in at most one point.

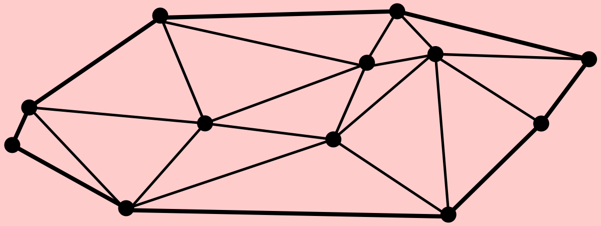
$f : A \subseteq \Re^2 \longrightarrow \Re$. f(p) = height of point p in the domain A of the terrain.

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.

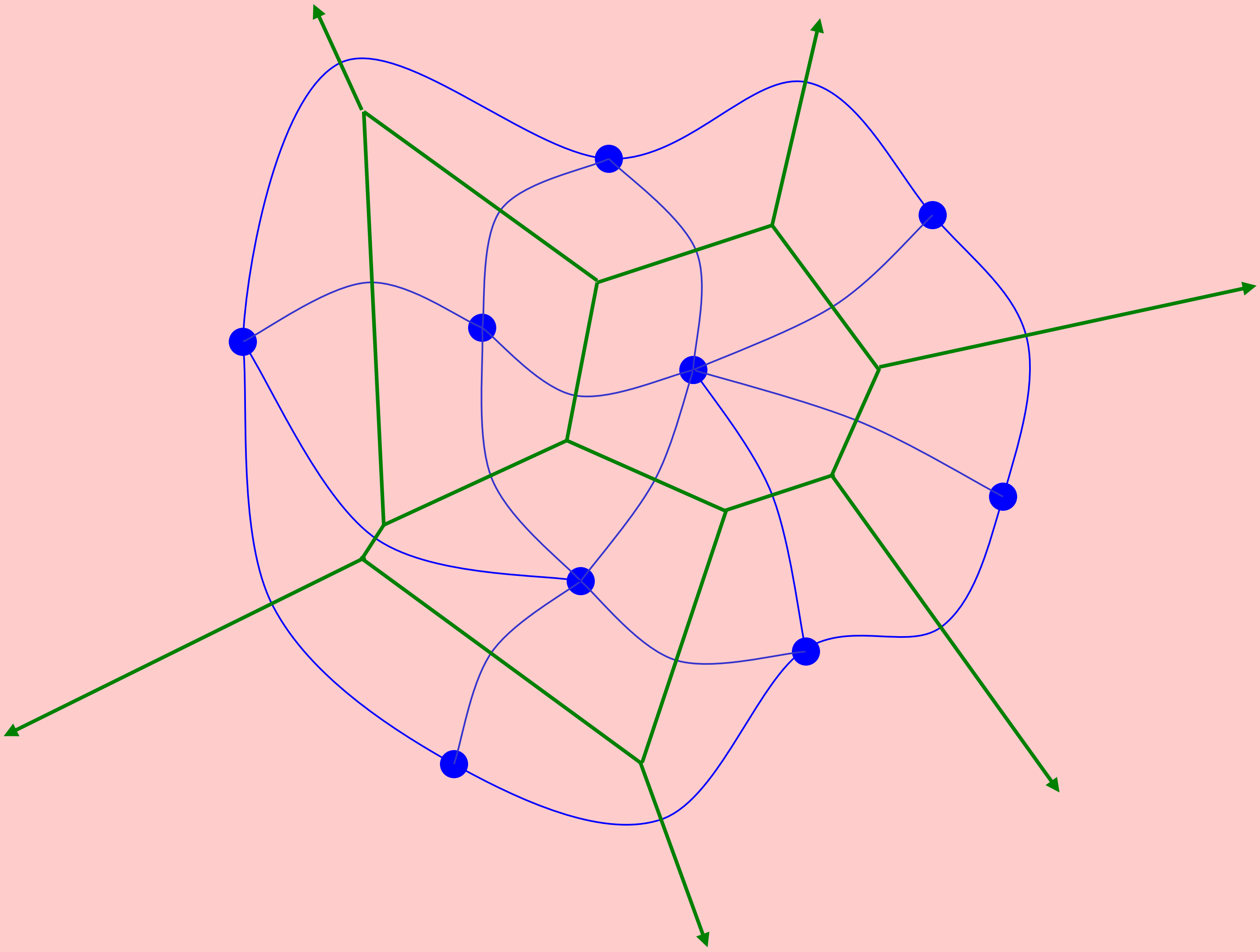
Proof: m = # triangles

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

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

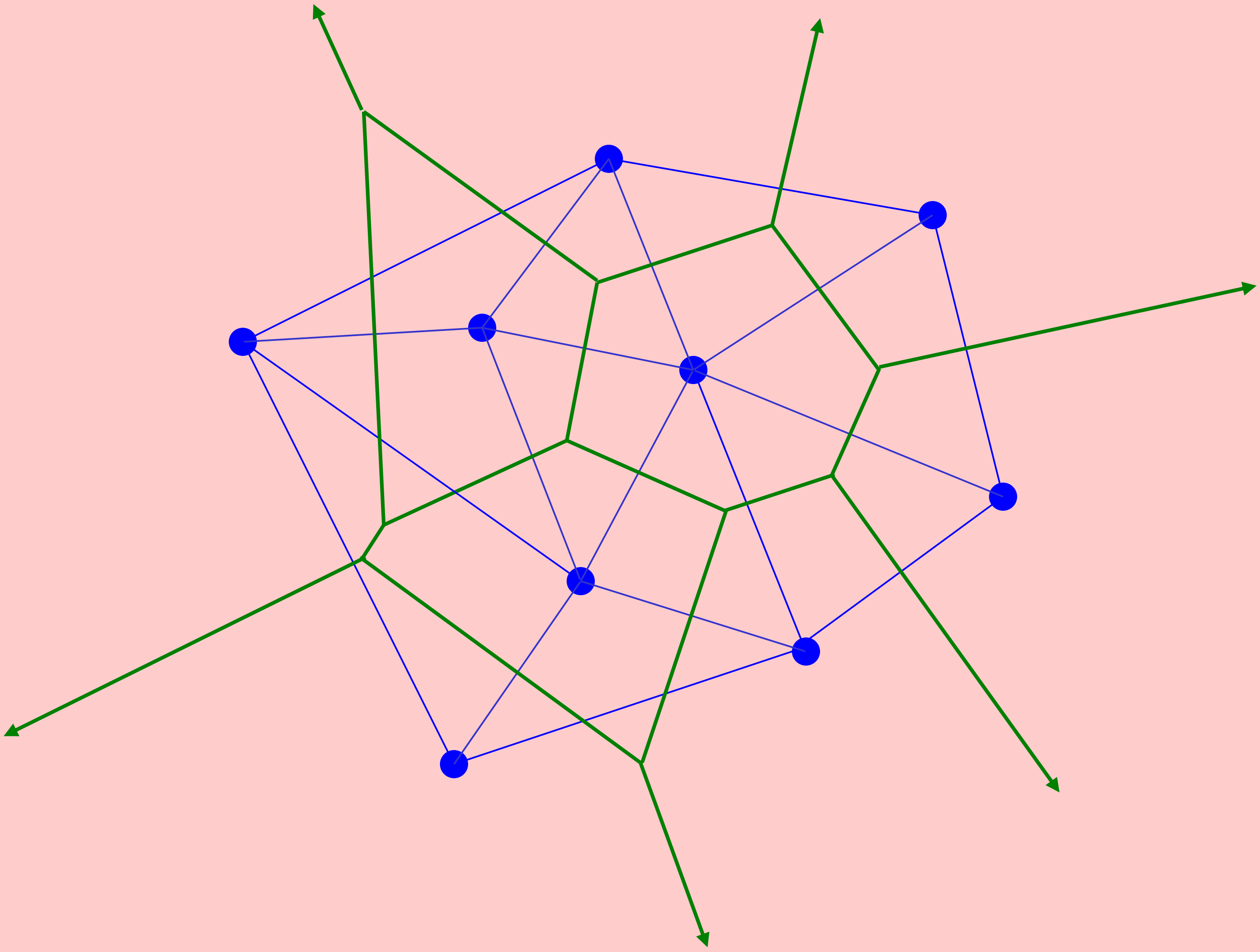
$\therefore\ m = 2n - h - 2, \quad E = 3n - h - 3.$

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 straineght-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

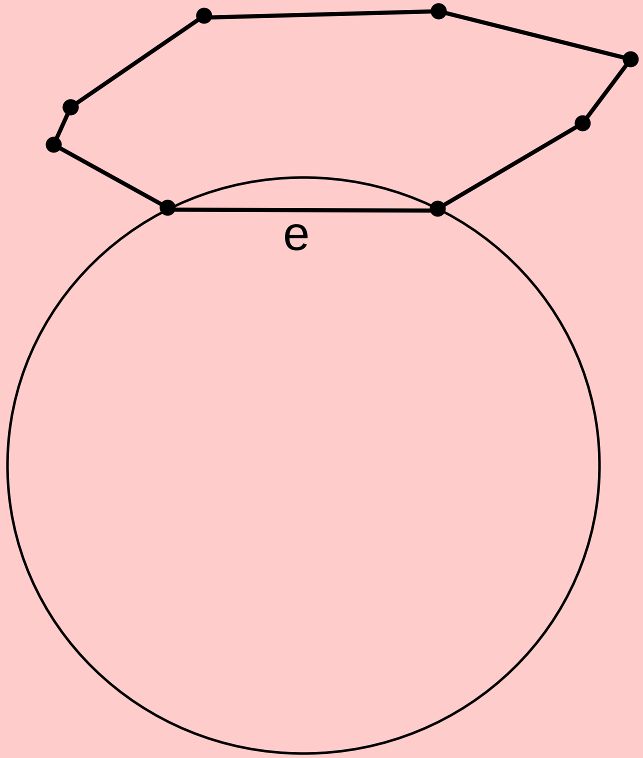
- a straight-line plane graph, &
- a triangulation of P.

Proof: Follows from the following Lemmas.

Delaunay Graph is a Triangulation

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

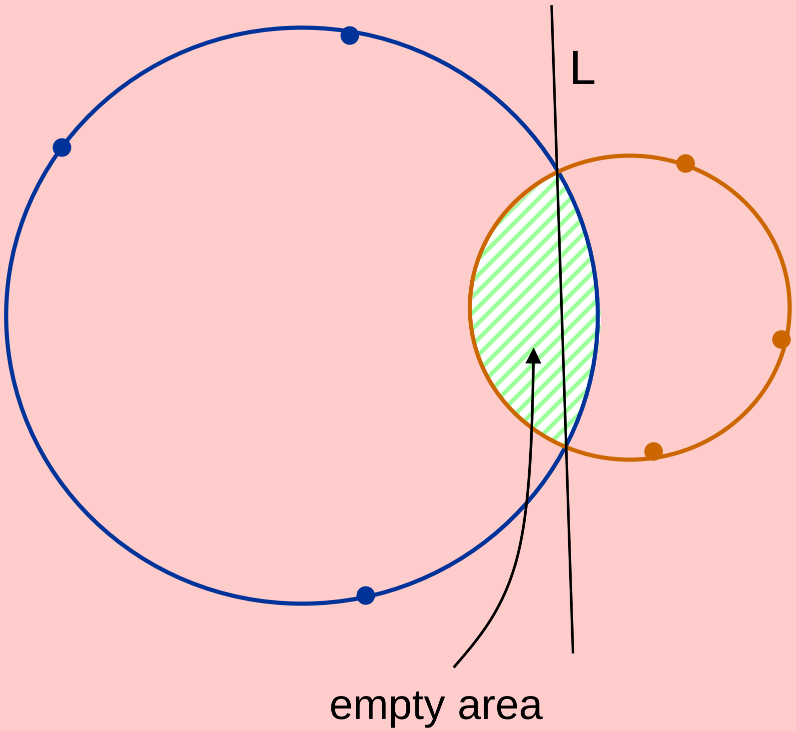
Proof: Consider a sufficiently large circle that passes through the 2 ends of CH edge e, and whose center is separated from CH(P) by the line aff(e).



Delaunay Graph is a Triangulation

LEMMA 2: No two Delaunay triangles overlap.

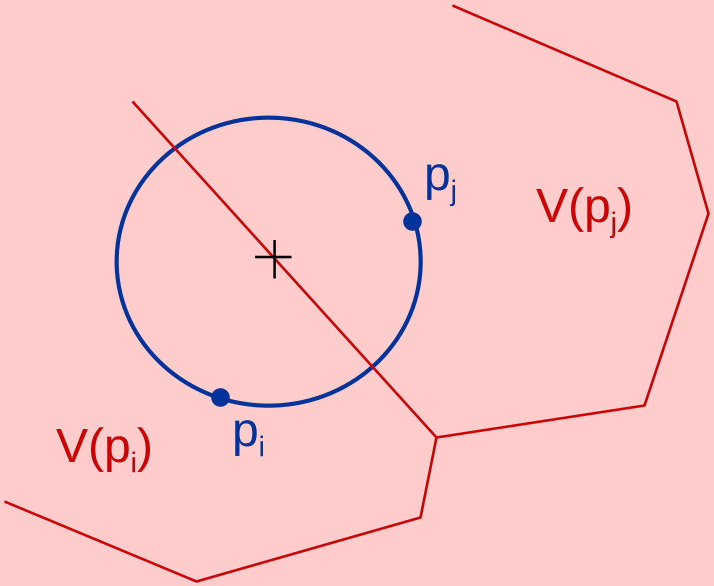
Proof: Consider circumscribing circles of two such triangles.
Line L separates the two triangles.



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.

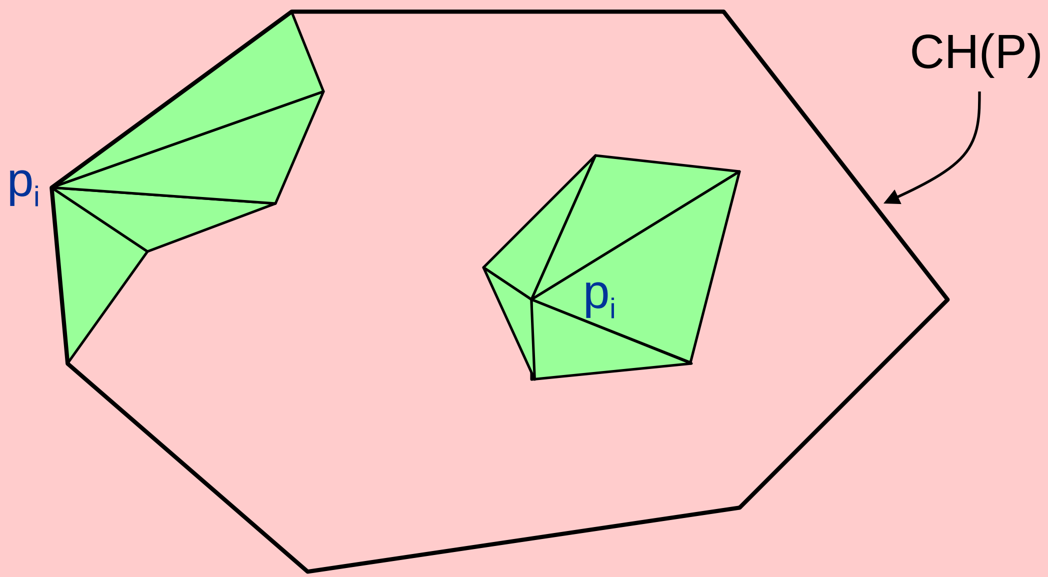
Proof: p_j & p_k must also be Voronoi neighbors.
Now apply Lemma 3 to (p_i , p_j), (p_i , p_k), (p_j , p_k).

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).

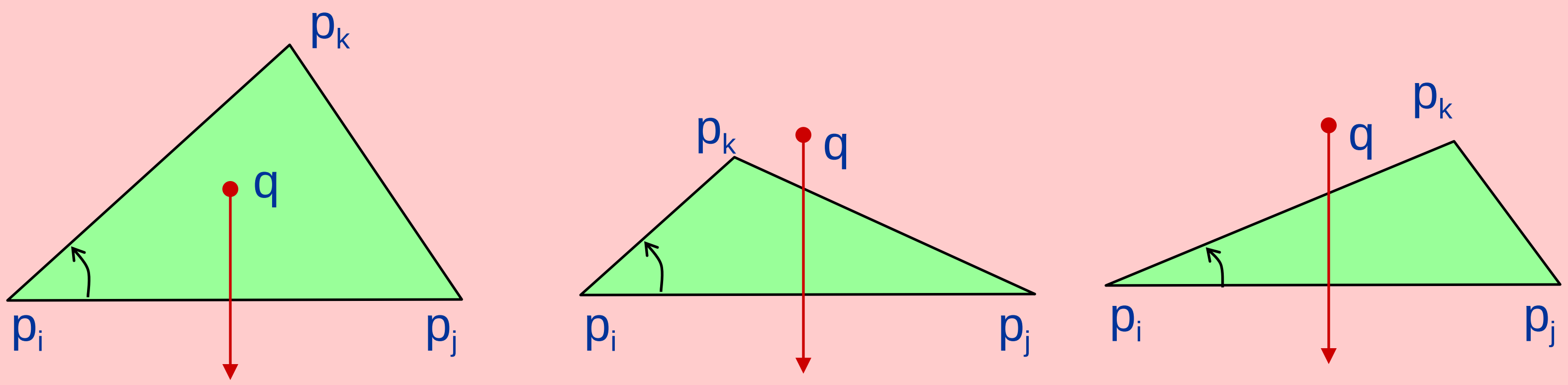
COROLLARY 5: For each $p_i \in P$, the Delaunay triangles incident to p_i completely cover a small open neighborhood of p_i inside CH(P).



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($\mathcal{T}$) = (α_1 , α_2 , … , α_{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.

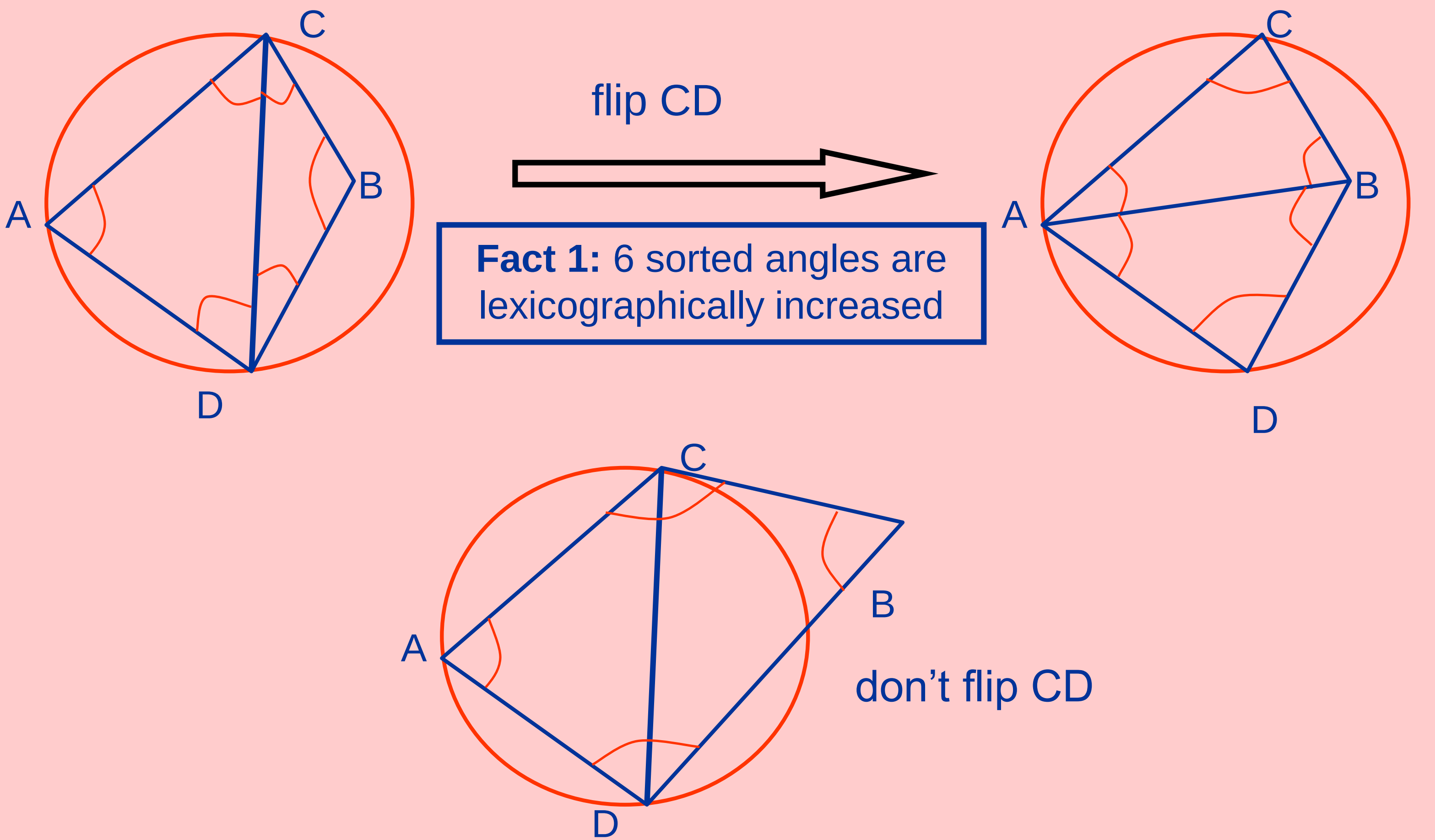
A simple $O(n^2)$ time DT Algorithm

Step 1: Let T be an arbitrary triangulation of $P \subseteq \Re^2$.

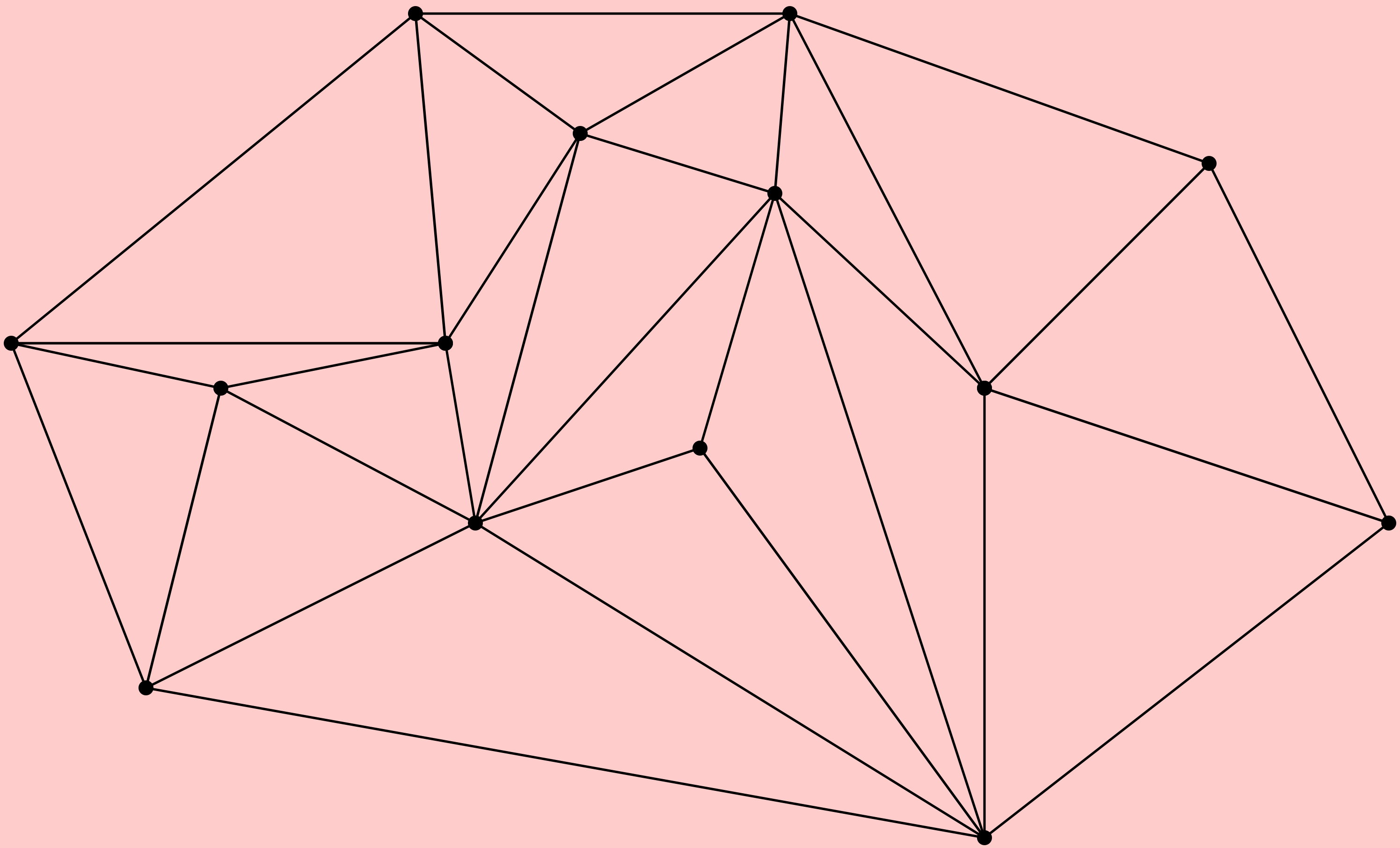
[e.g., use sweep in O(n log n) time]

Step 2: **while** T has a quadrangle of the form below with $\angle A + \angle B > 180°$

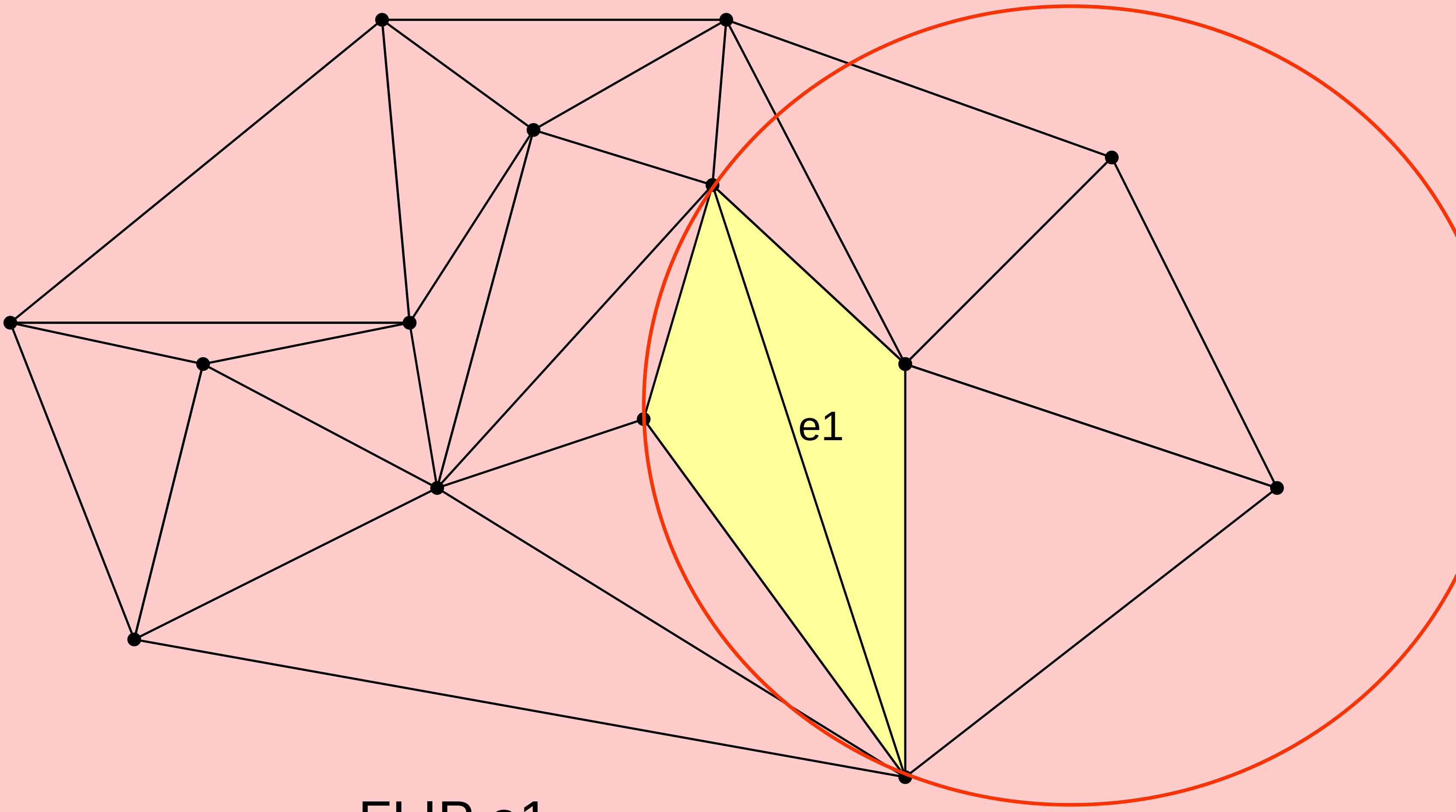
do flip diagonal CD (i.e., replace it with diagonal AB). [$O(n^2)$ iterations]



A snapshot of the Algorithm

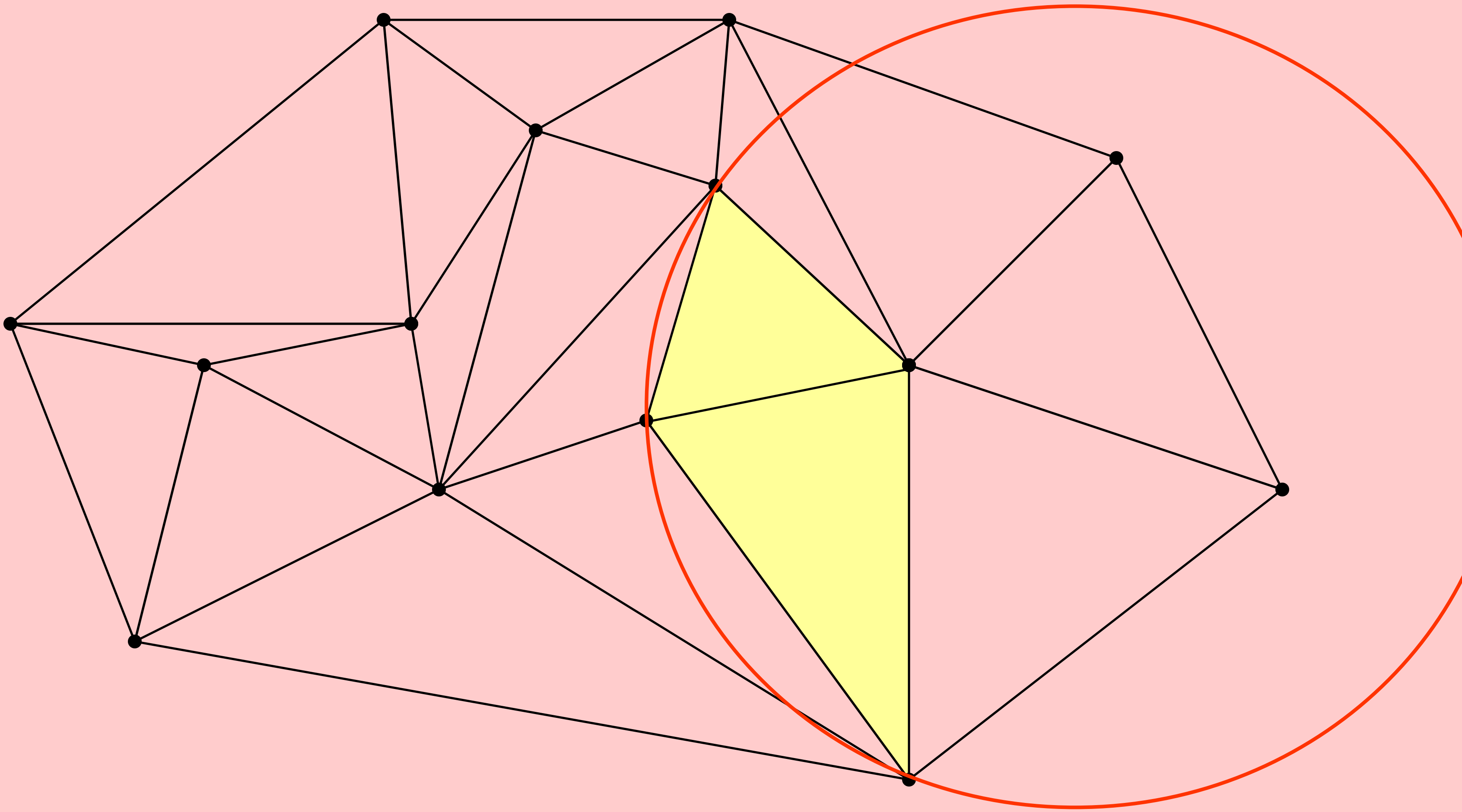


A snapshot of the Algorithm

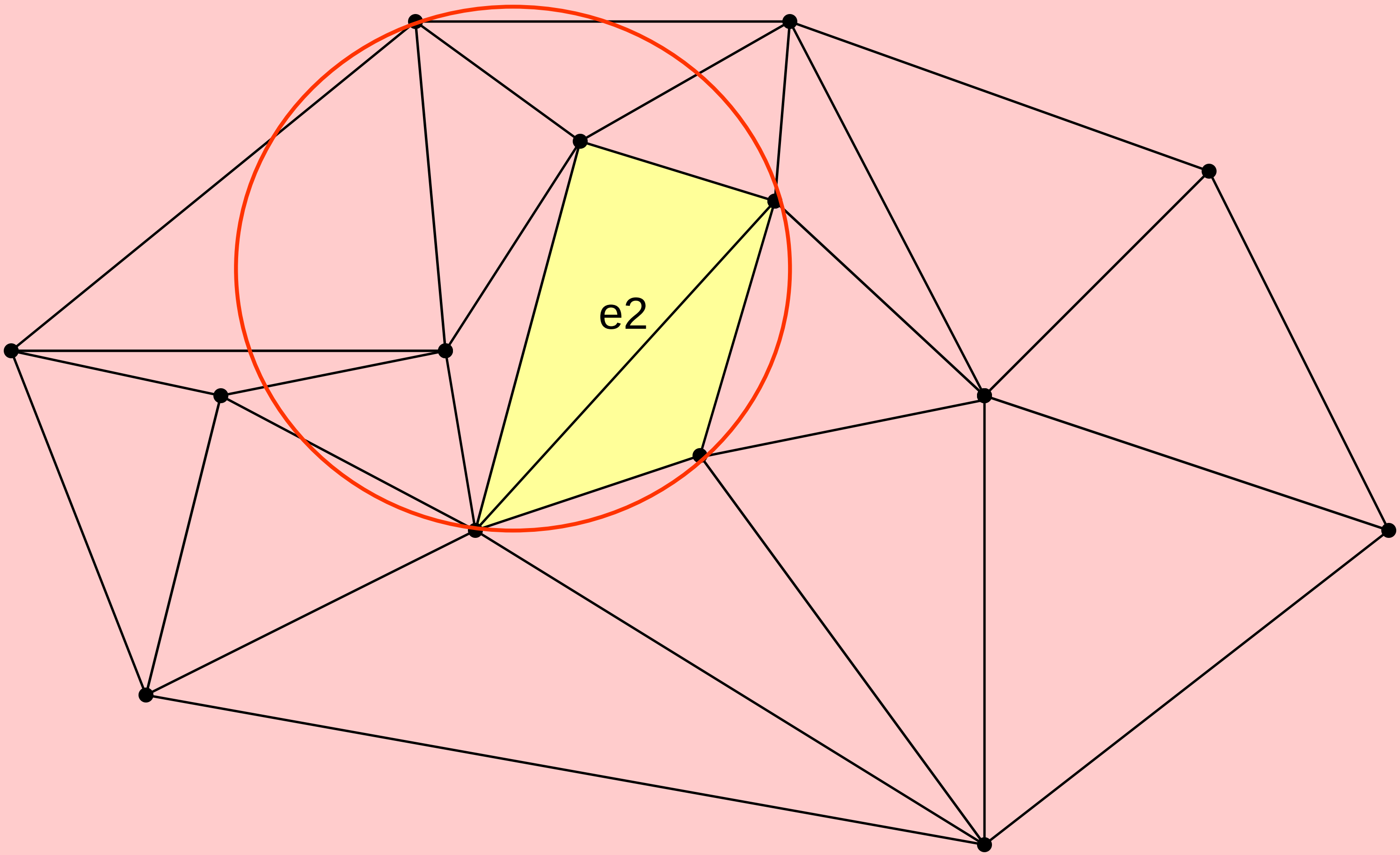


FLIP e1

A snapshot of the Algorithm

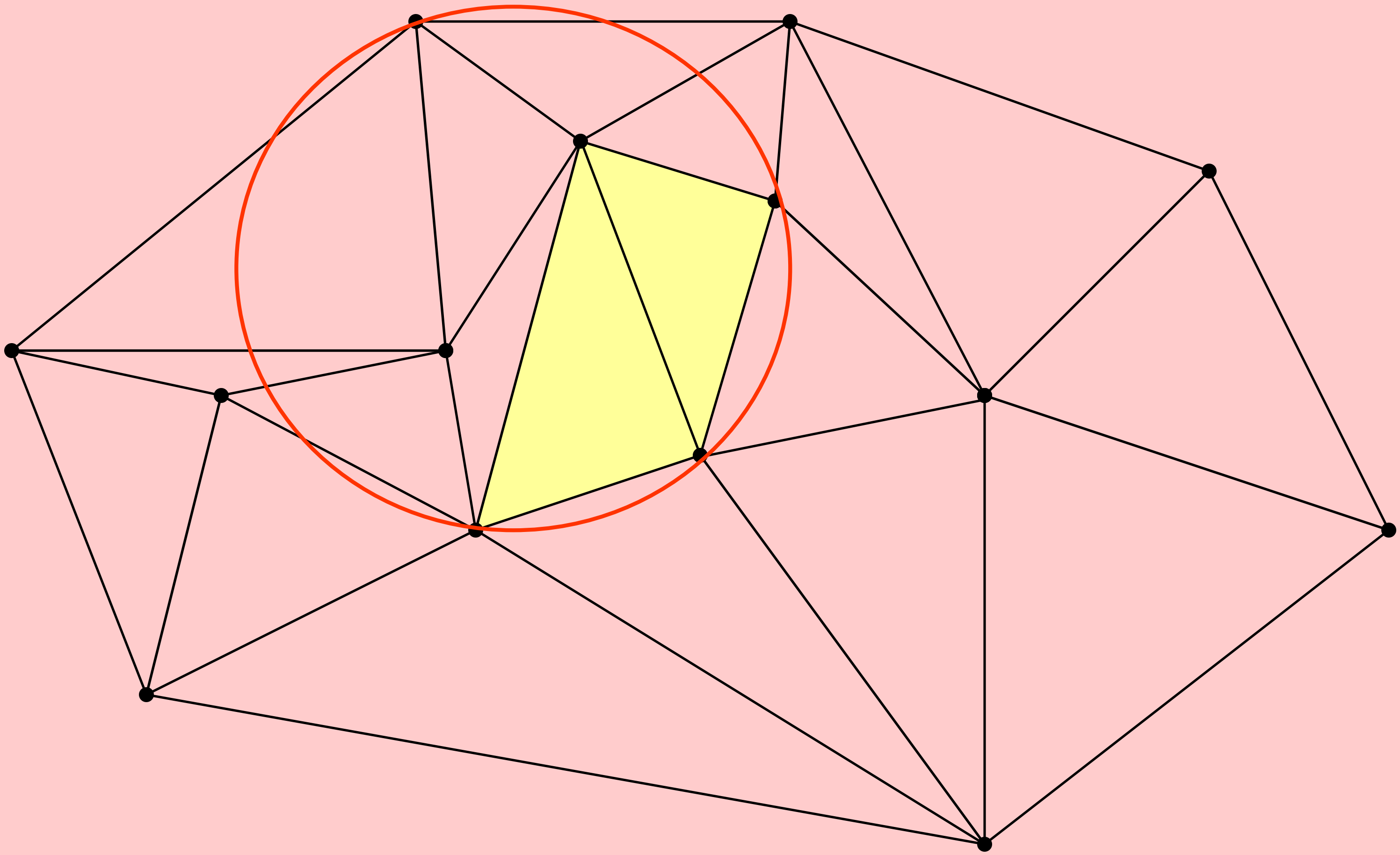


A snapshot of the Algorithm

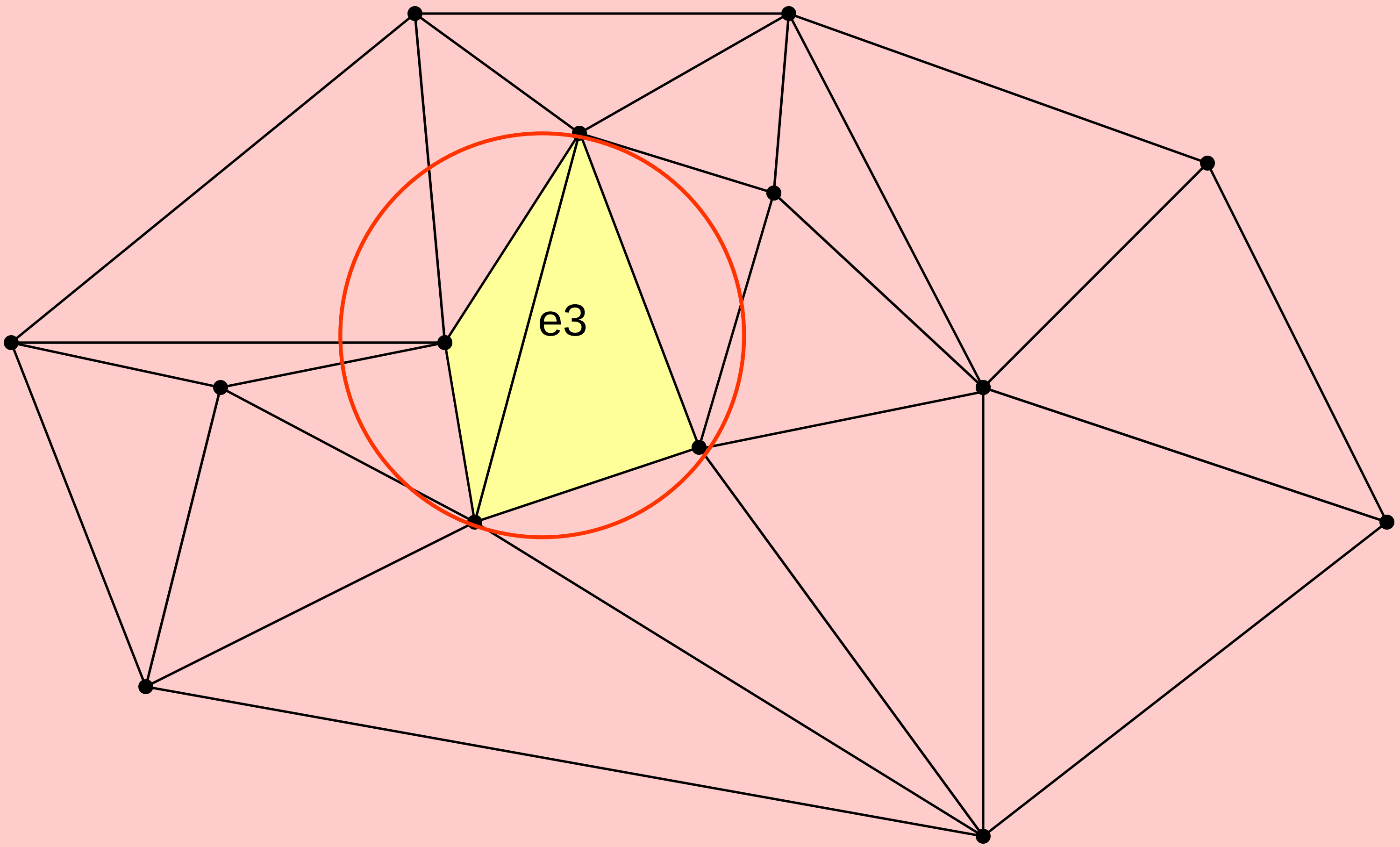


FLIP e2

A snapshot of the Algorithm

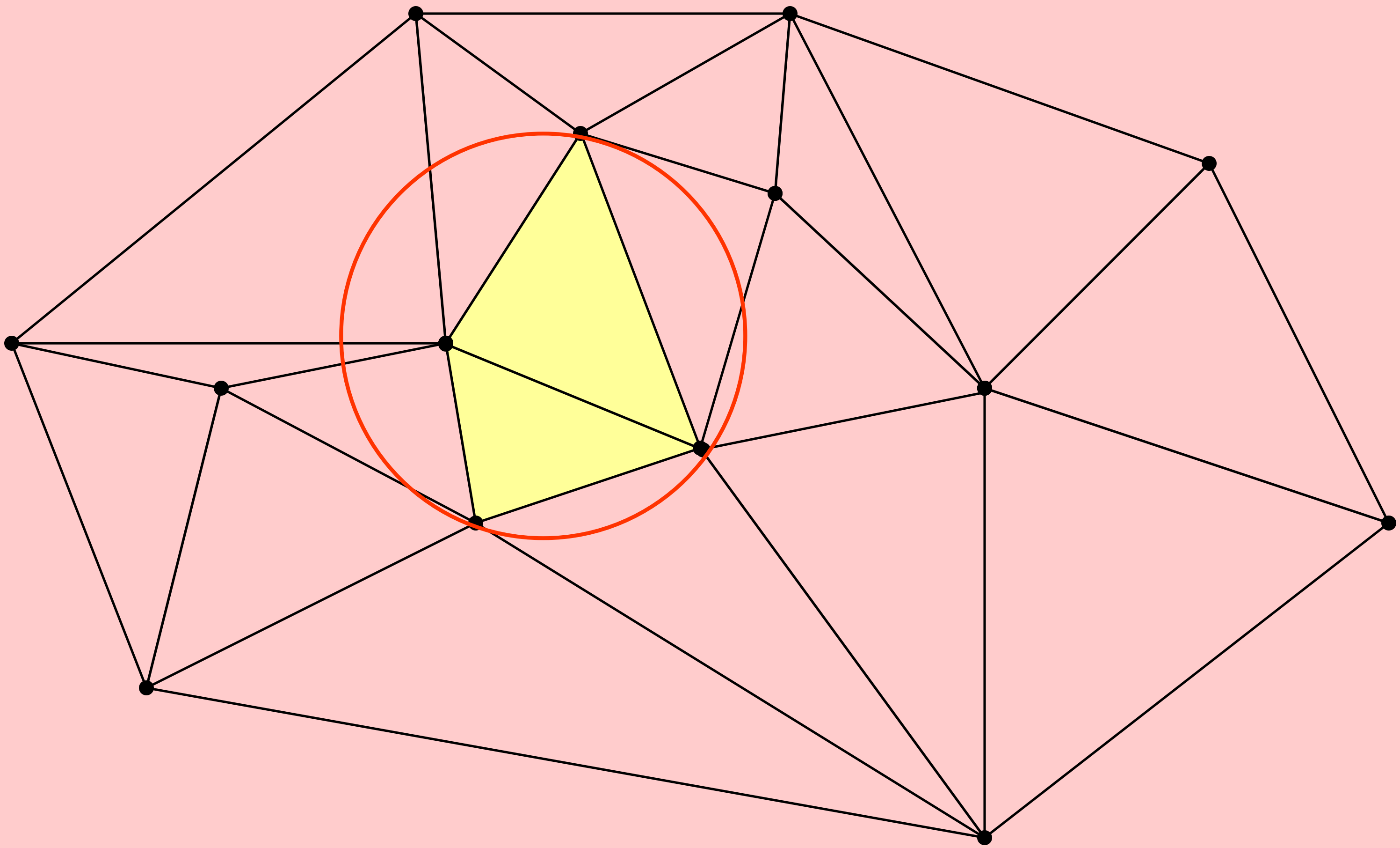


A snapshot of the Algorithm



FLIP e3

A snapshot of the Algorithm



A snapshot of the Algorithm