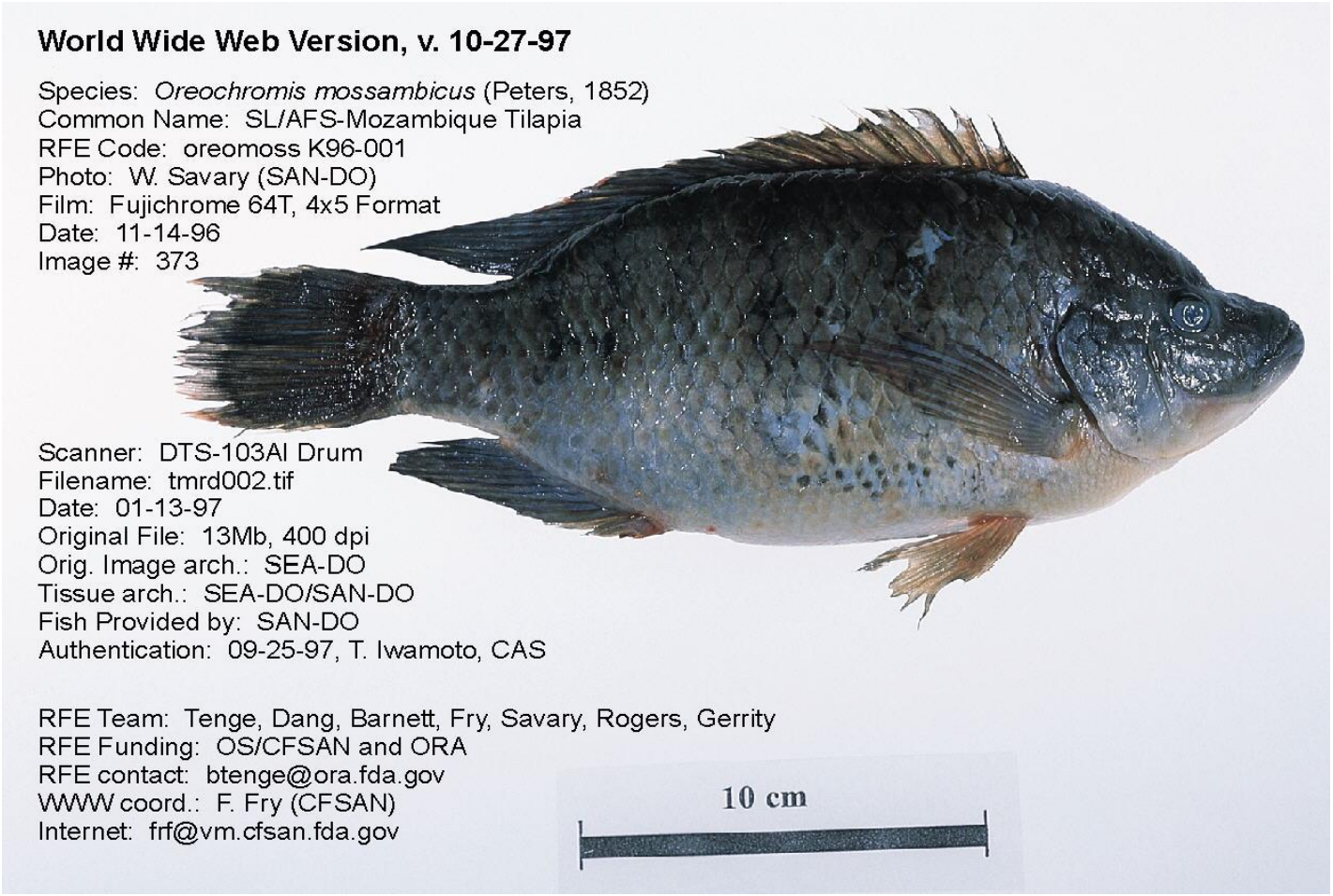
Basic Techniques in Computer Graphics

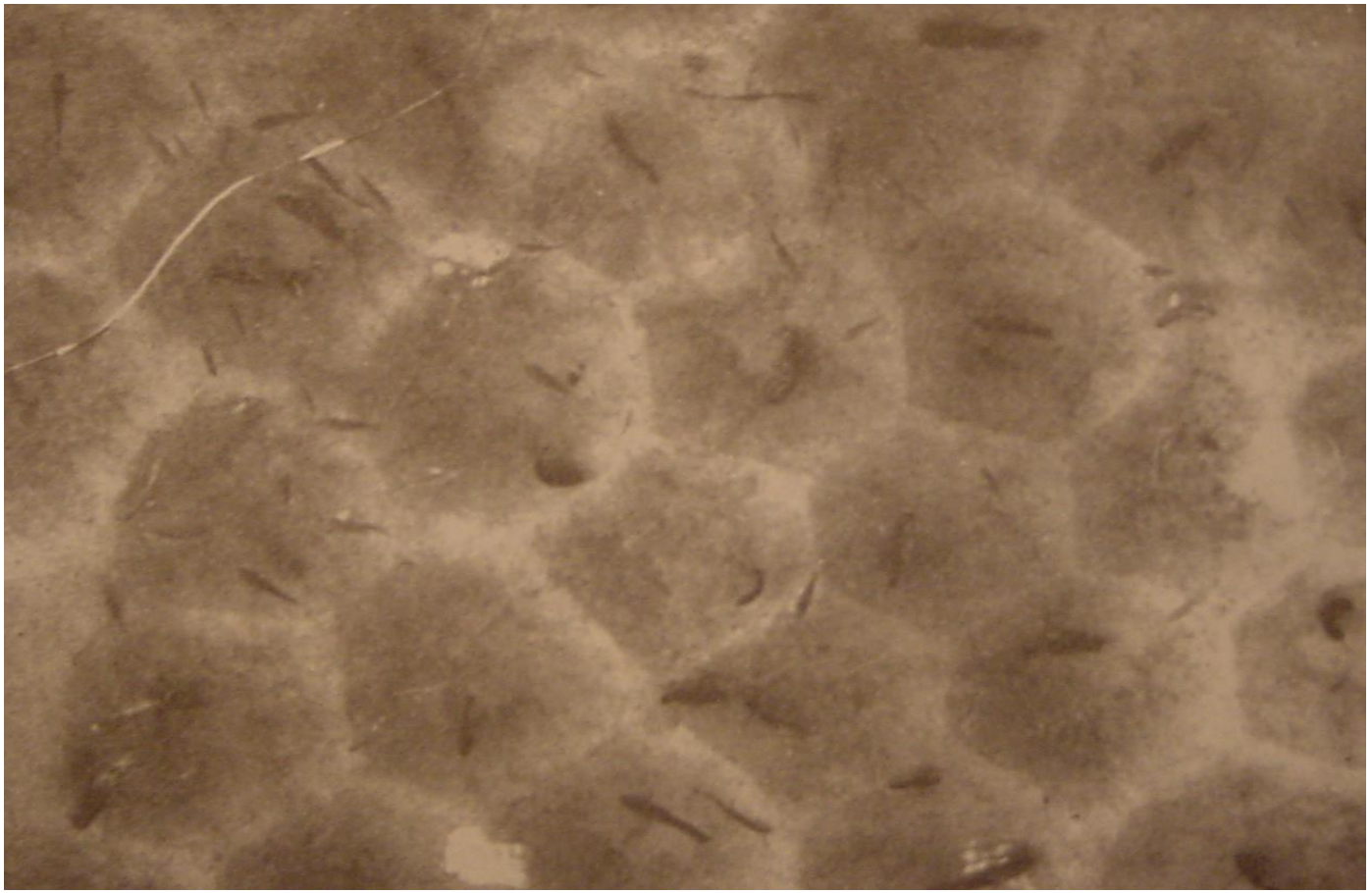
Winter 2017 / 2018

Visual Computing Institute
RWTH AACHEN UNIVERSITY

Neighborhood



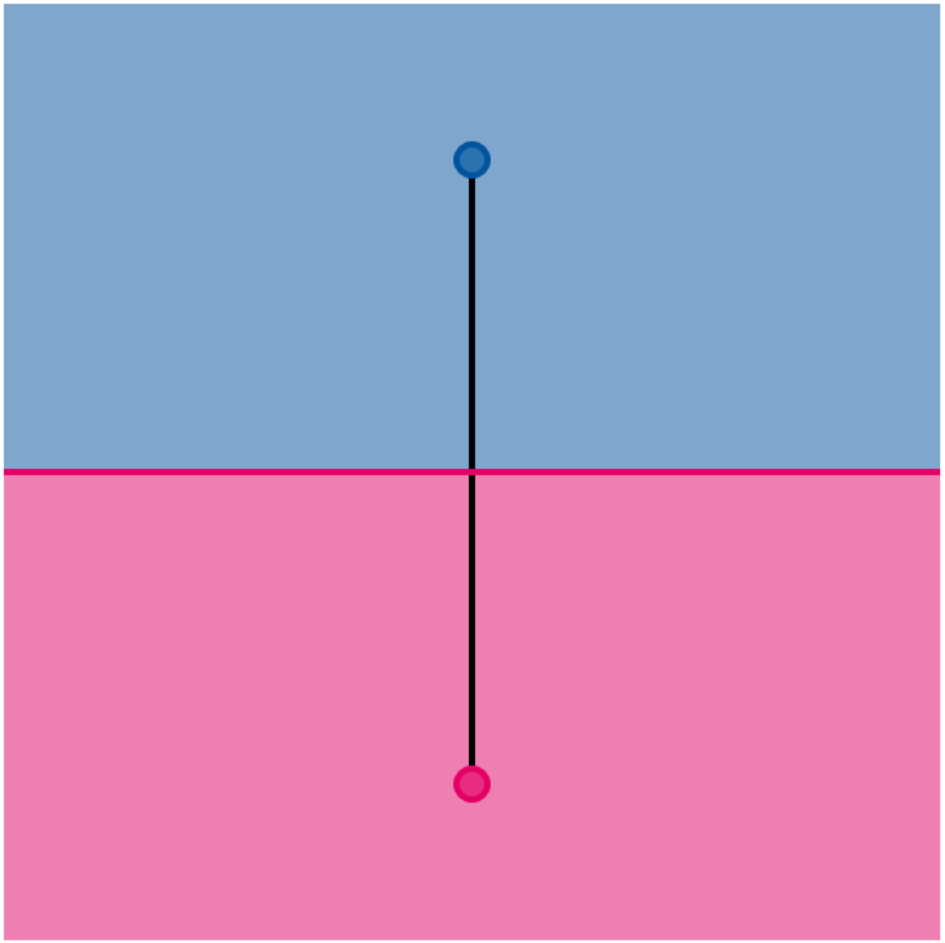
Neighborhood



Nests of Tilapia Mossambica

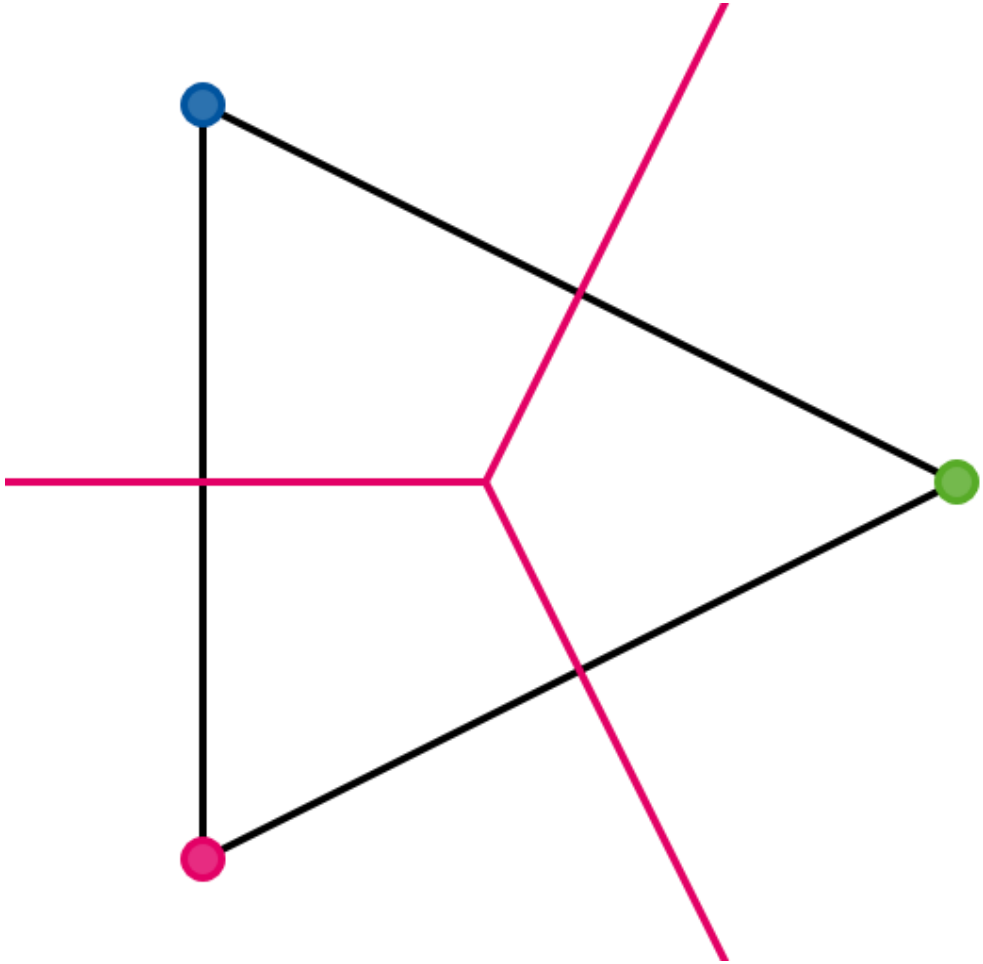
Voronoi Regions

Voronoi Regions

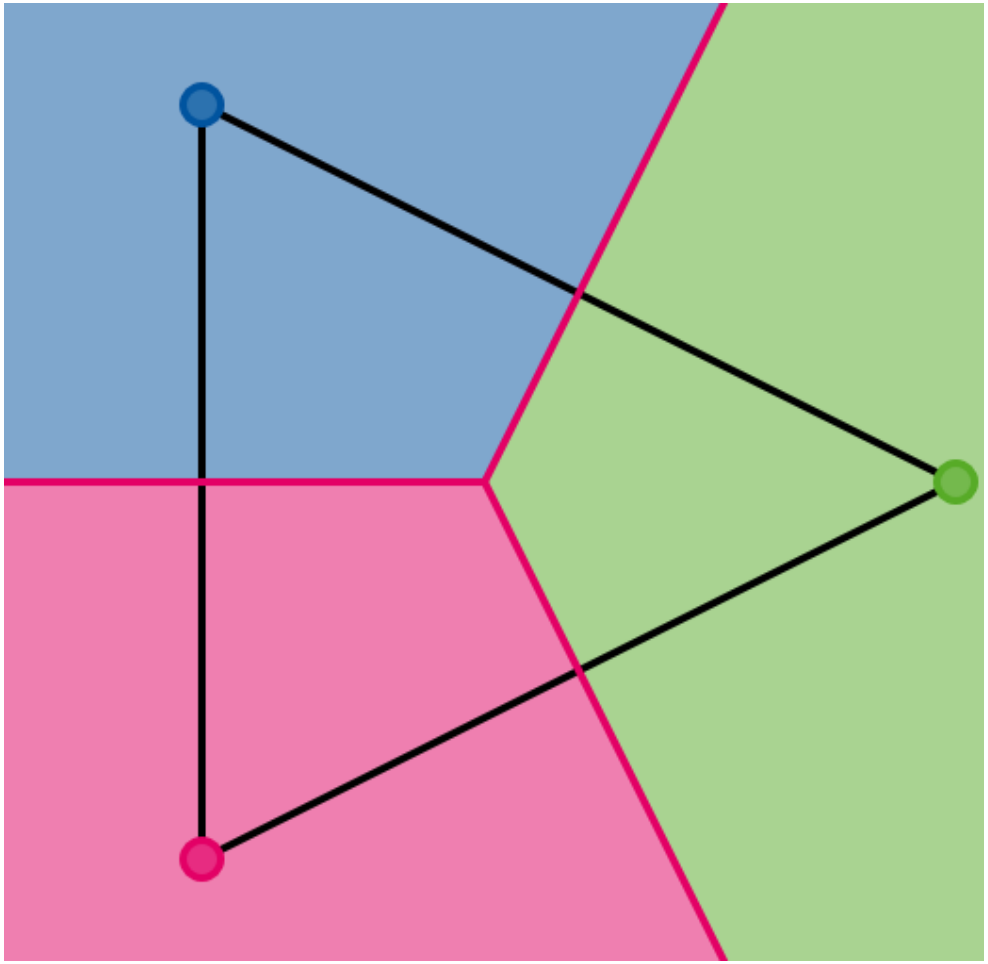


Voronoi Regions

Voronoi Regions

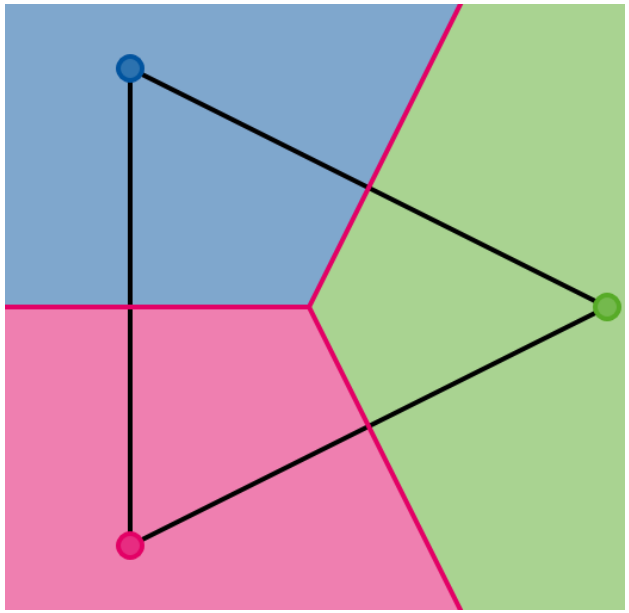


Voronoi Regions



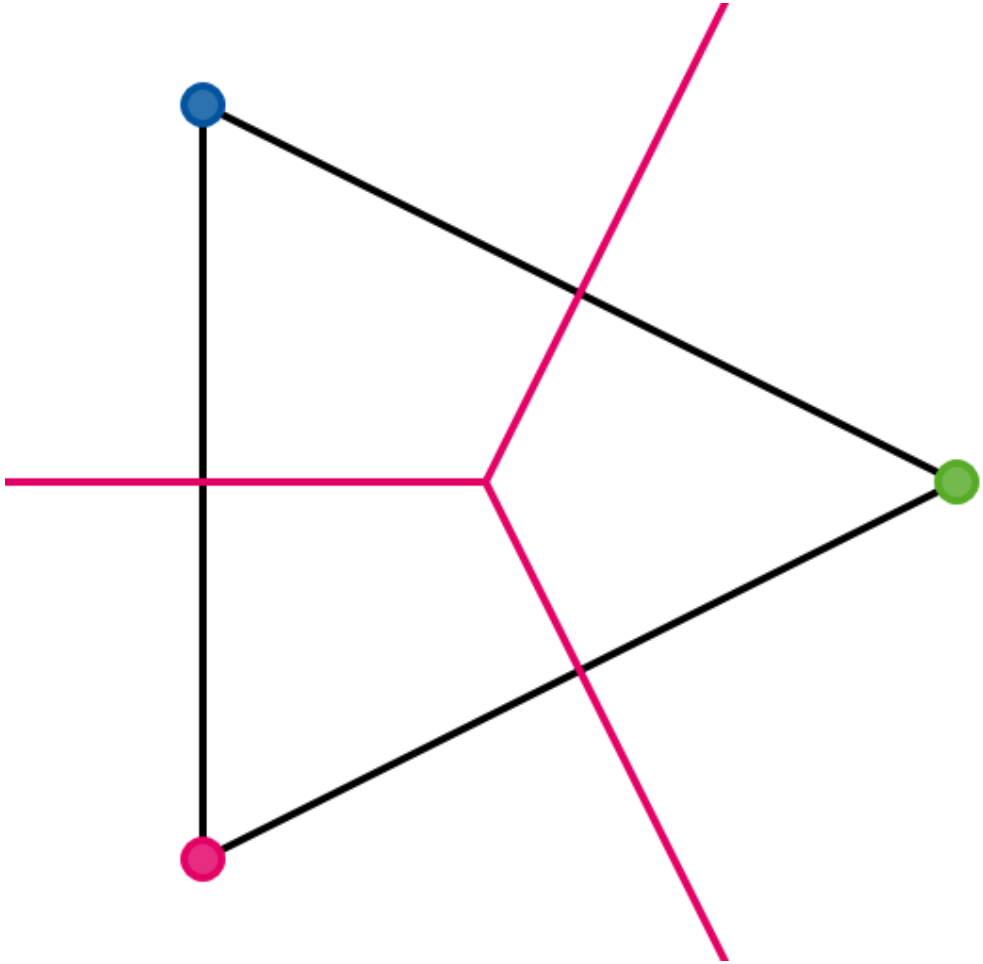
Voronoi Regions

- Given a point cloud $P = p_1, \dots, p_n$
- The **Voronoi-Region** for a point p_i is region with contains all points which have a shorter distance to p_i than to any other vertex p_j
- $V(p_i) = \{q \mid \forall j \neq i \colon \|q - p_i\| < \|q - p_j\|\}$
- Regions are convex

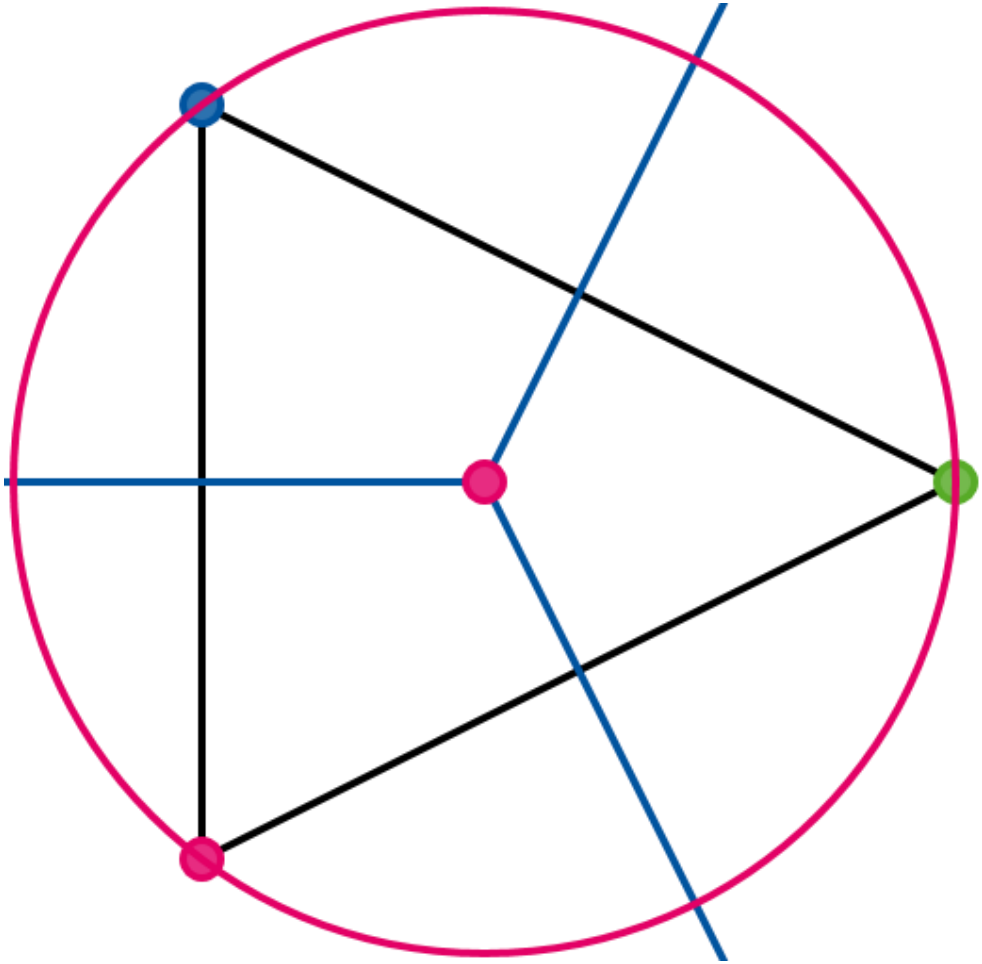


Each color is one **Voronoi-Region**

Voronoi Regions

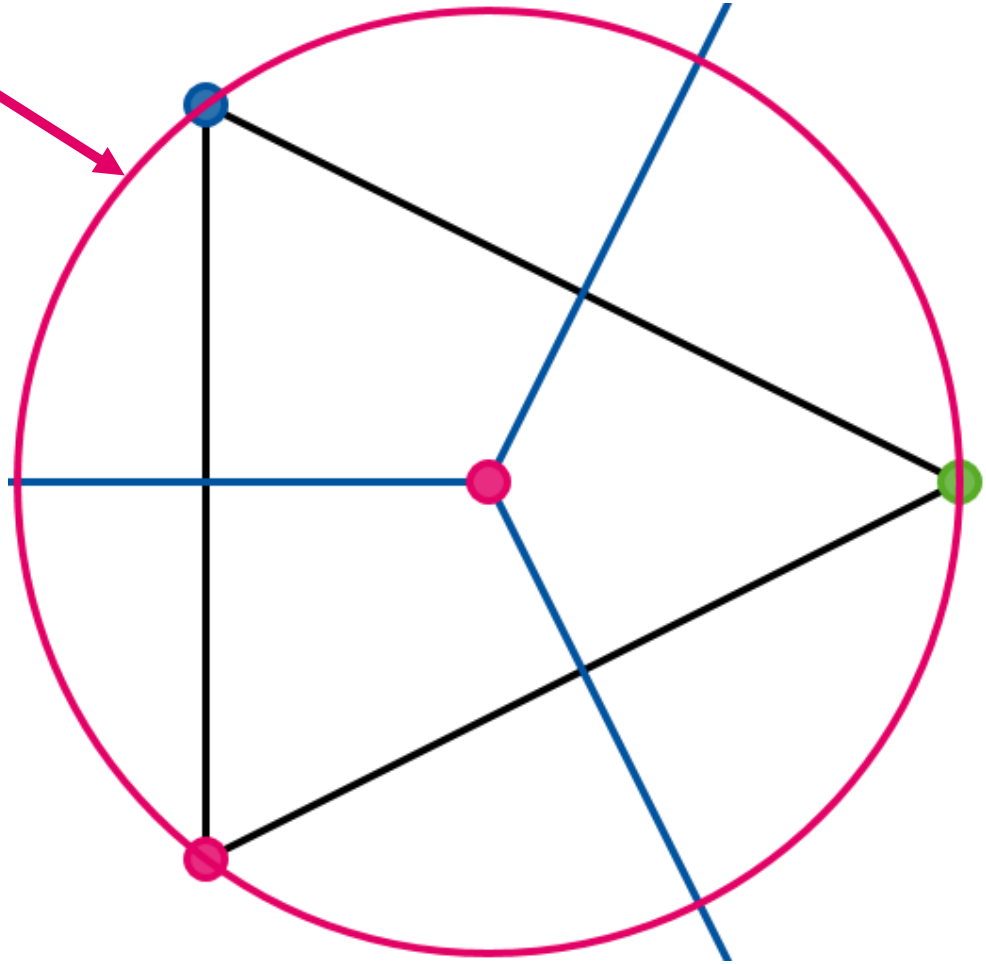


Voronoi Regions

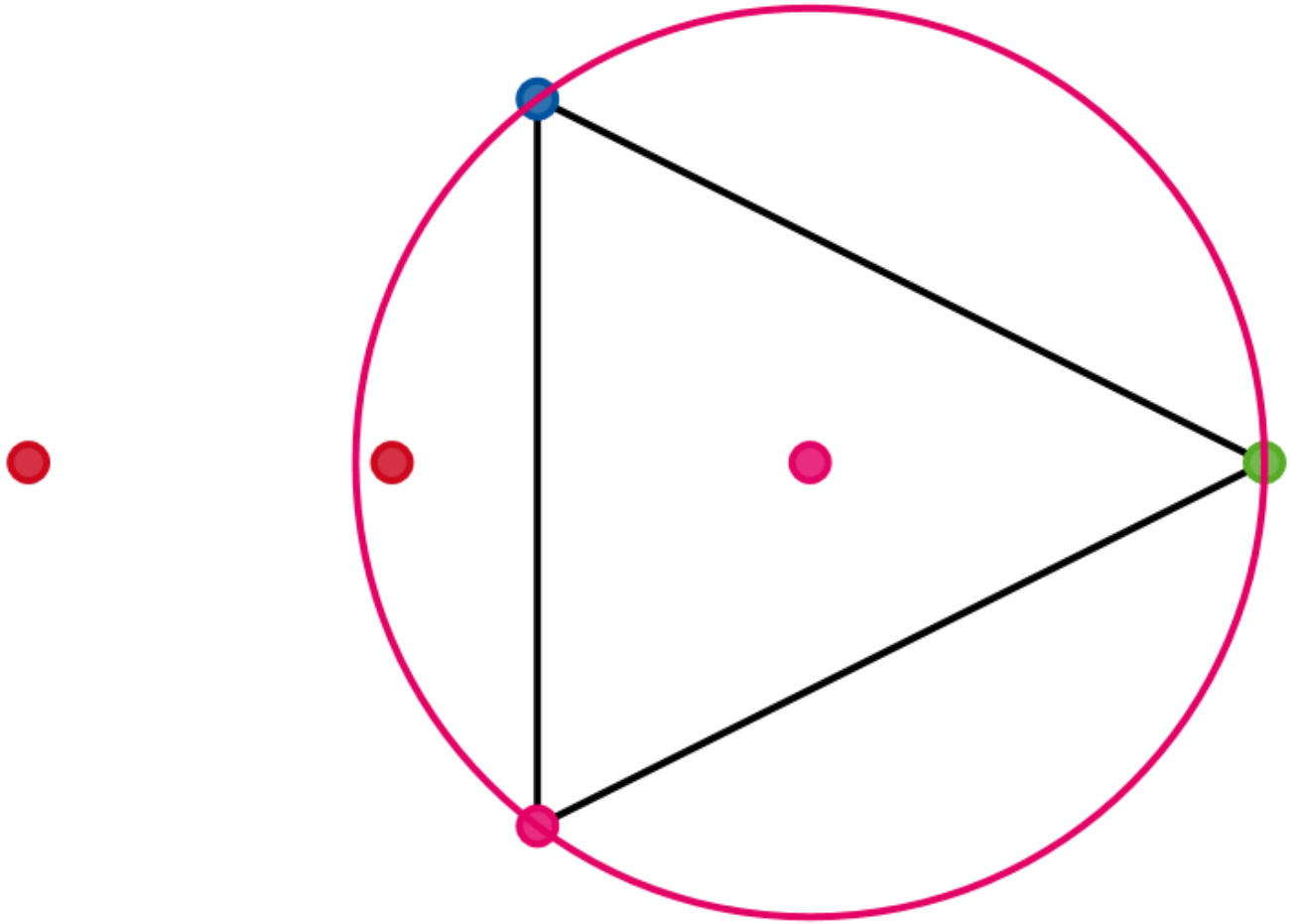


Voronoi Regions

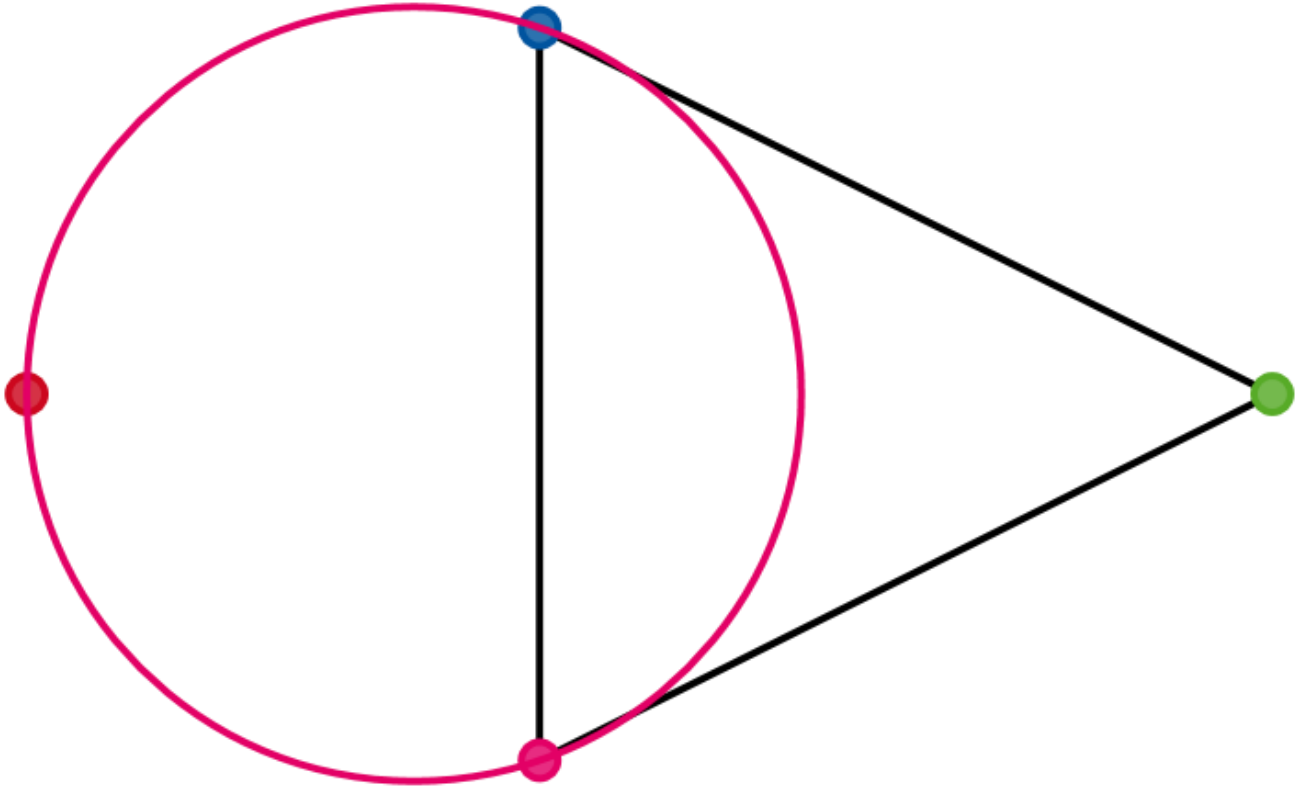
Circumcircle



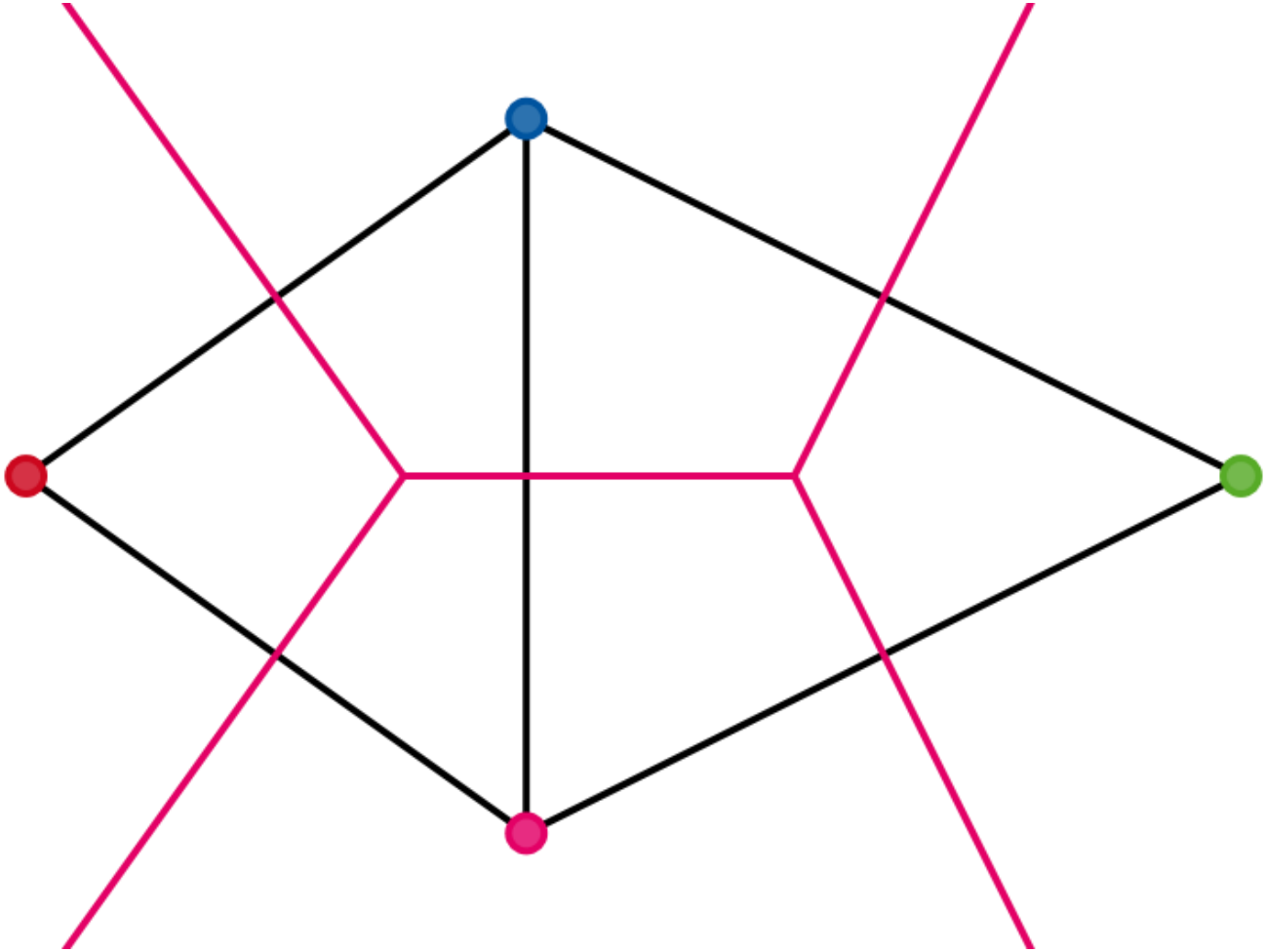
Voronoi Regions



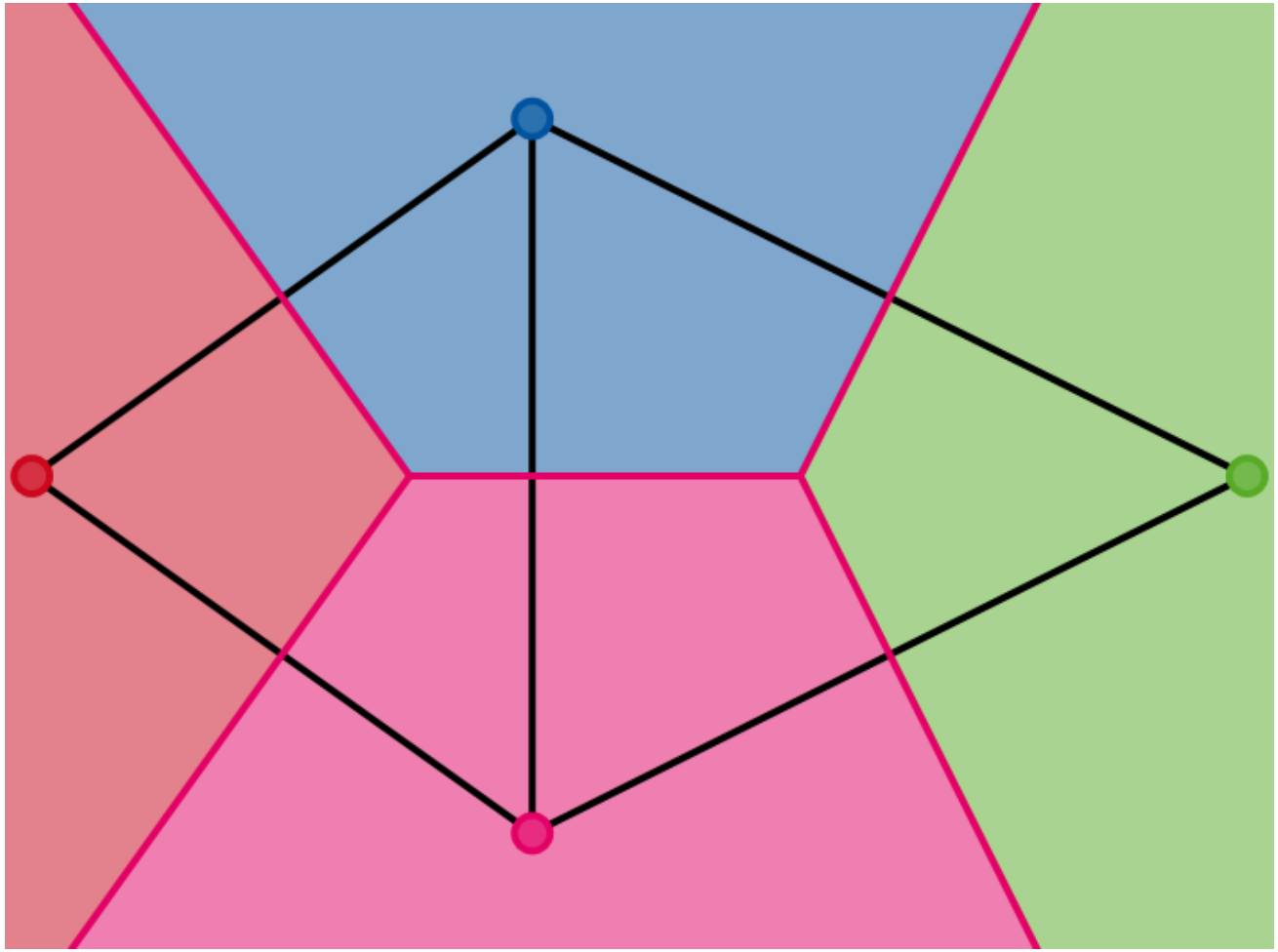
Voronoi Regions



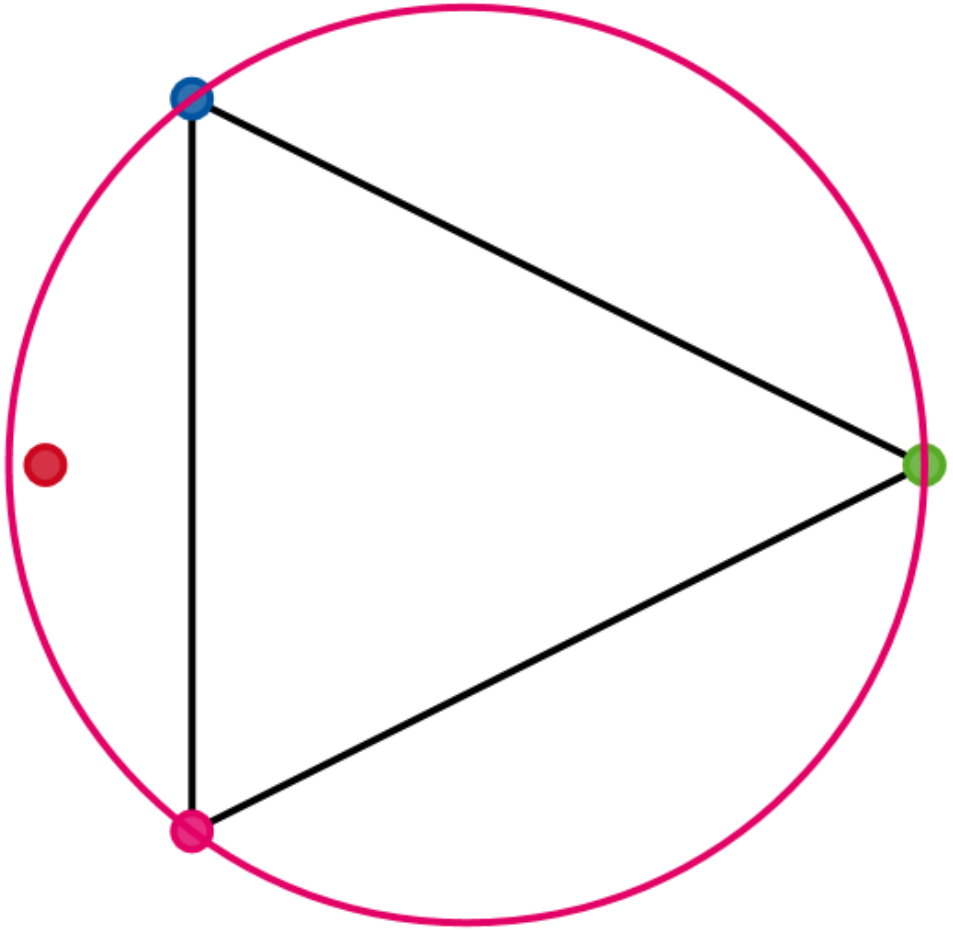
Voronoi Regions



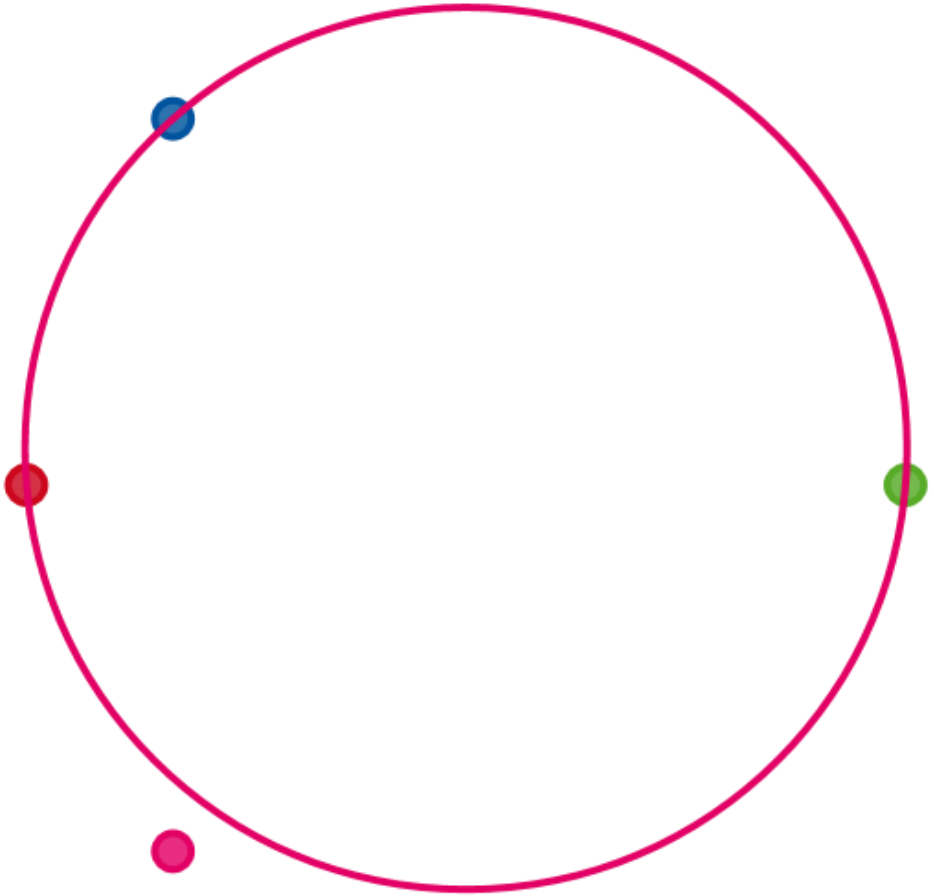
Voronoi Regions



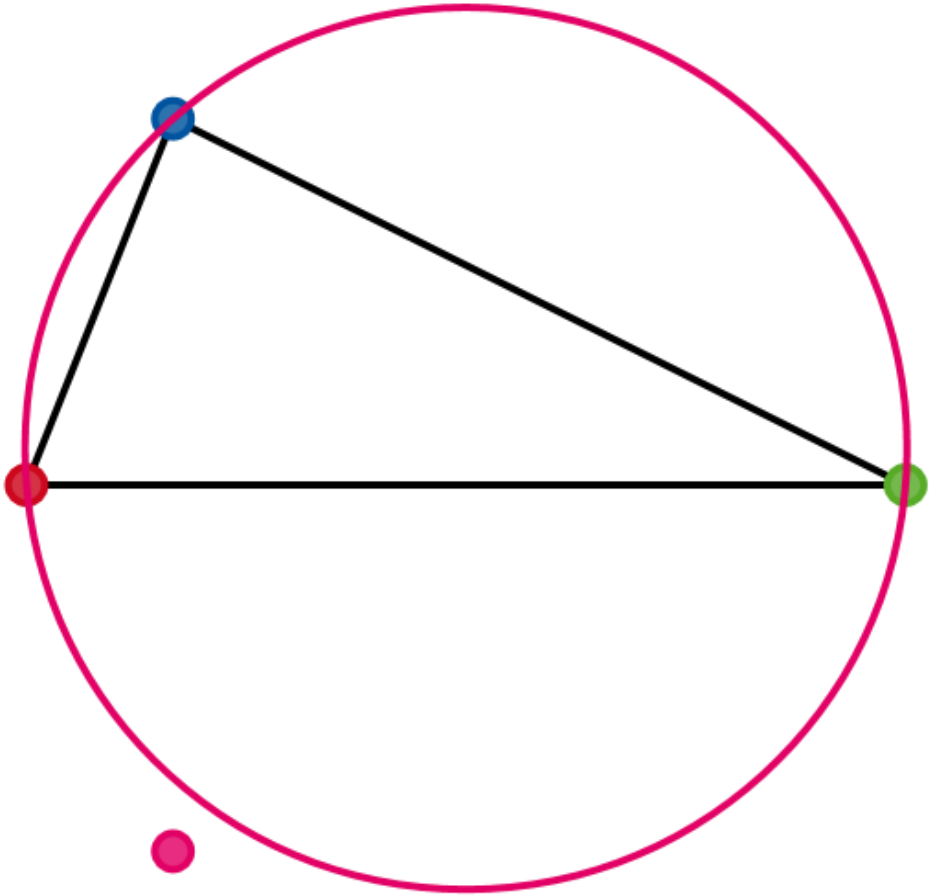
Voronoi Regions



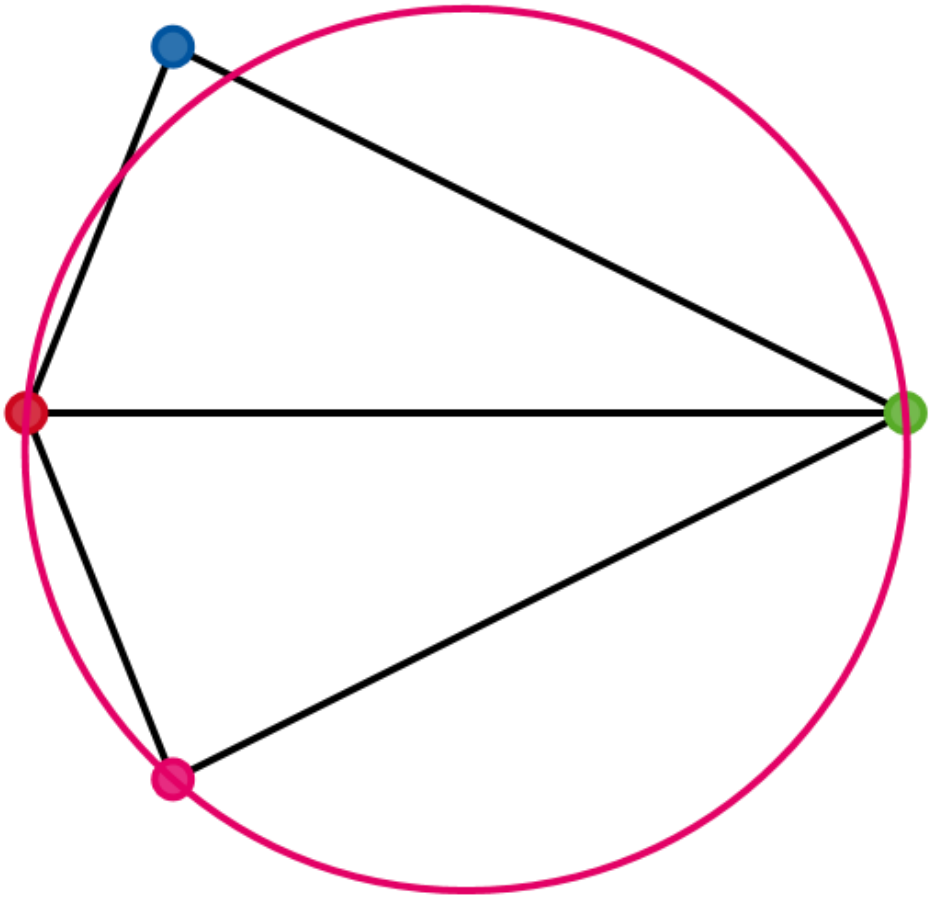
Voronoi Regions



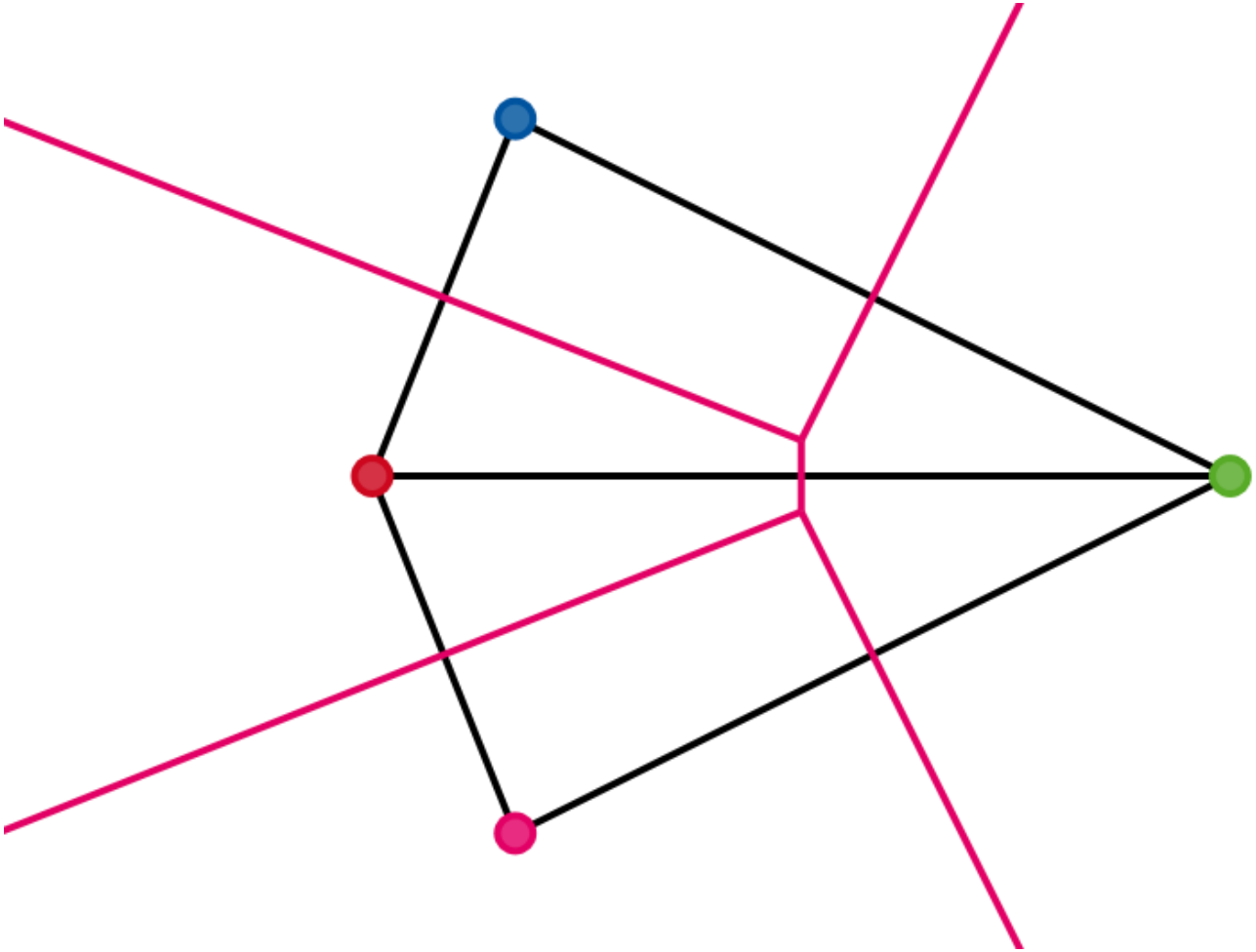
Voronoi Regions



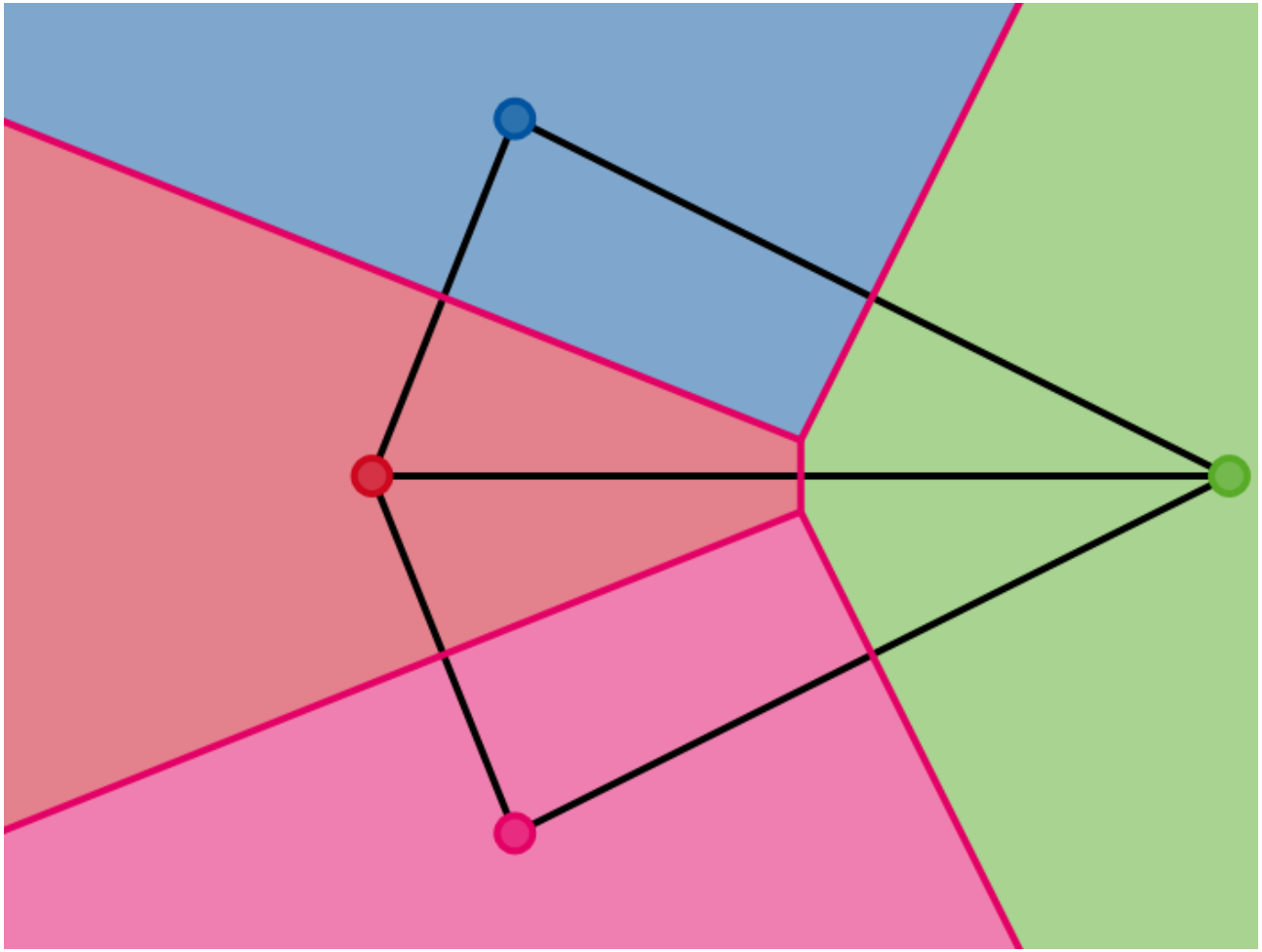
Voronoi Regions



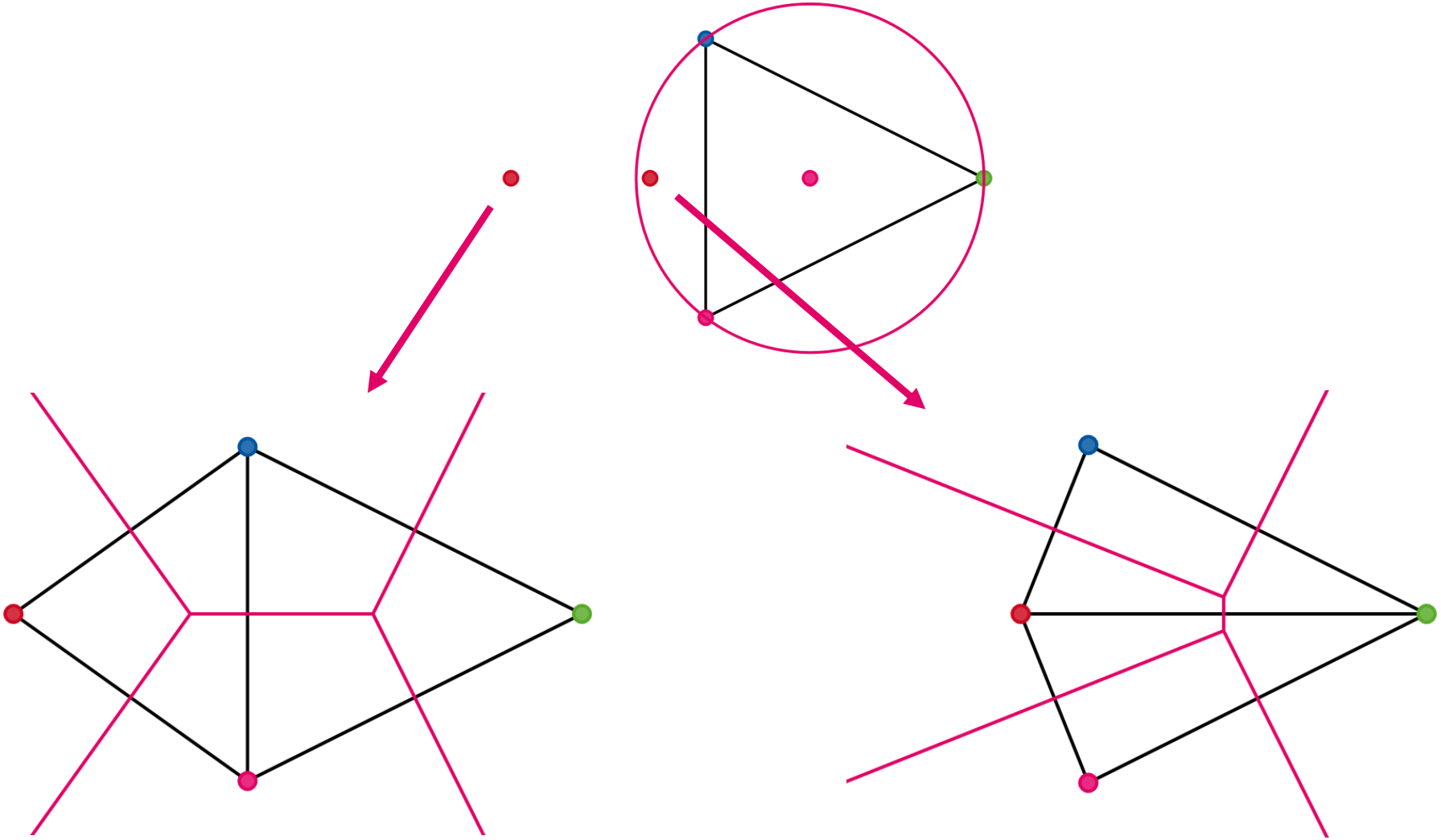
Voronoi Regions



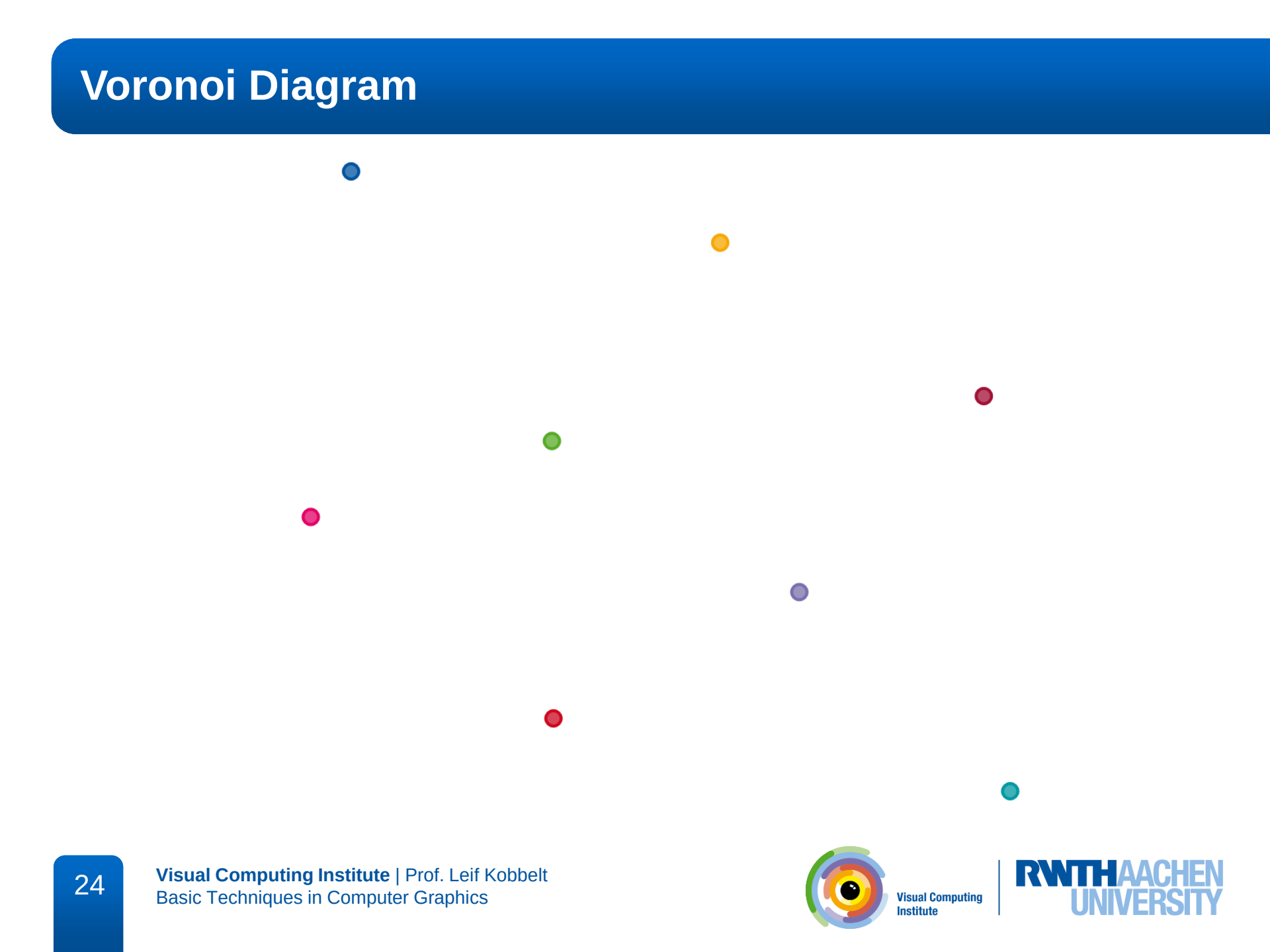
Voronoi Regions



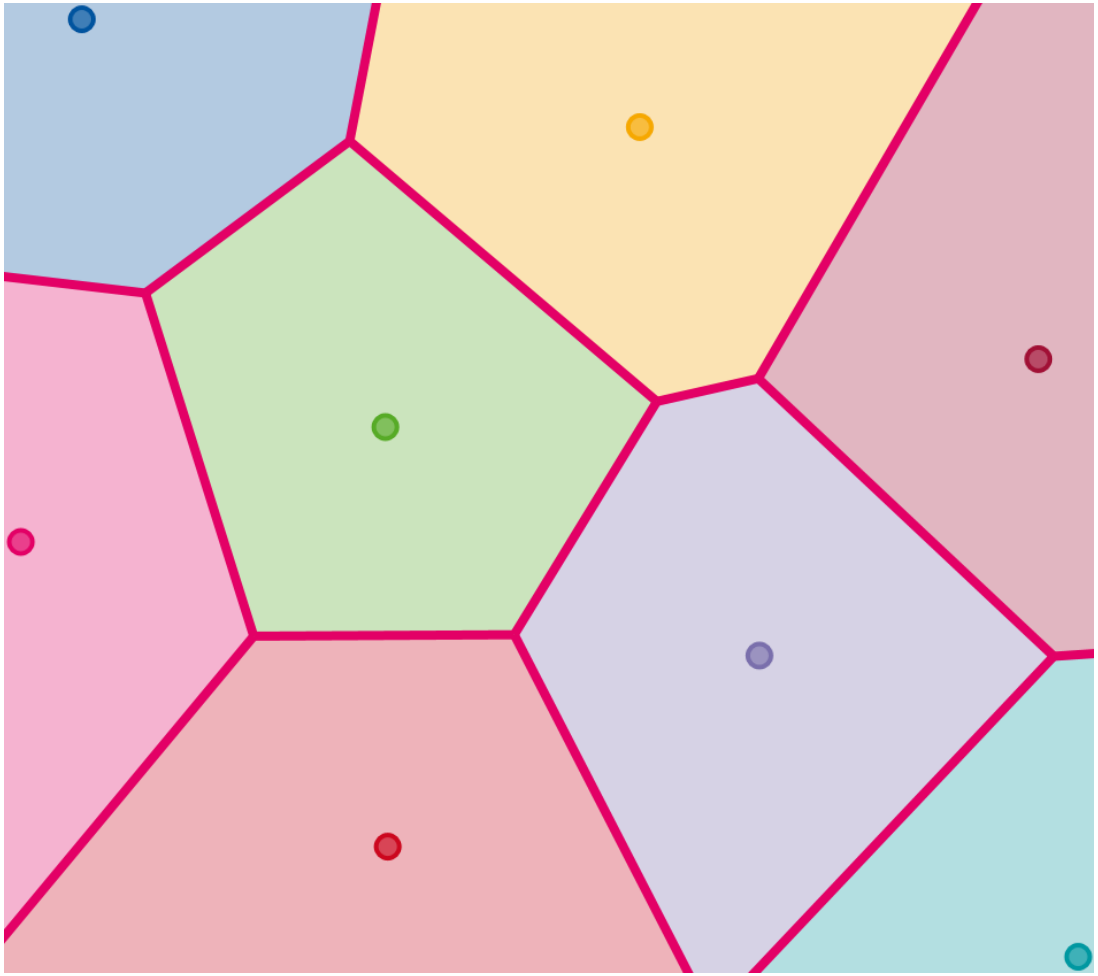
Voronoi Regions



Voronoi Diagram

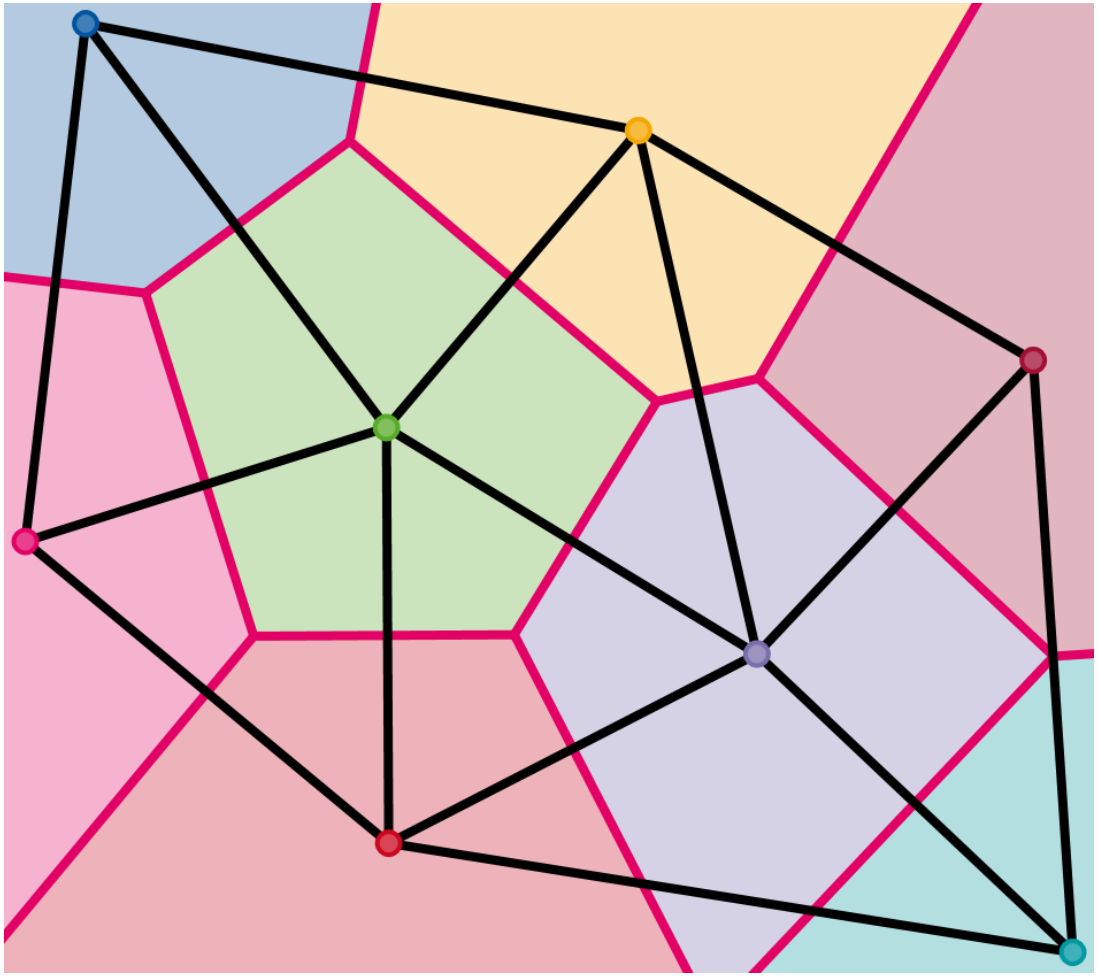


Voronoi Diagram

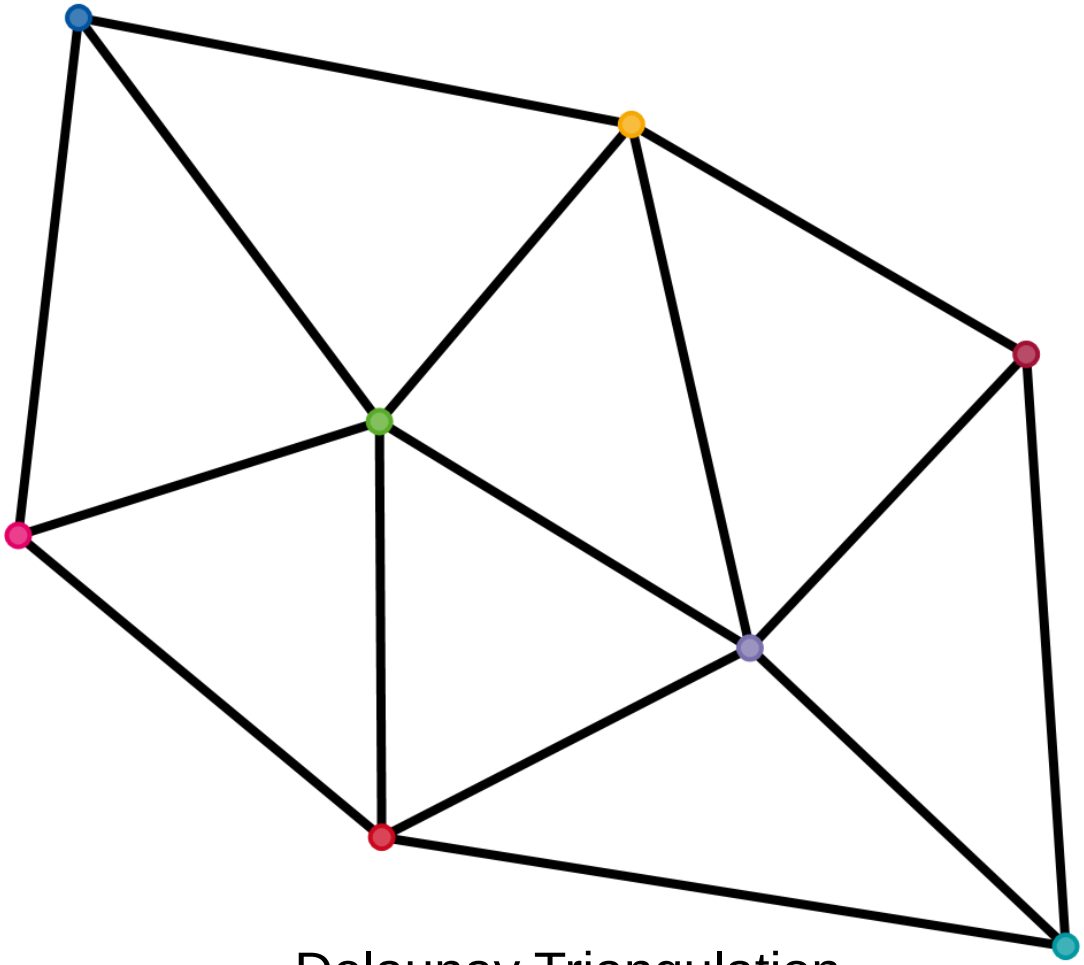


Voronoi Diagram

Voronoi Diagram and Delaunay Triangulation

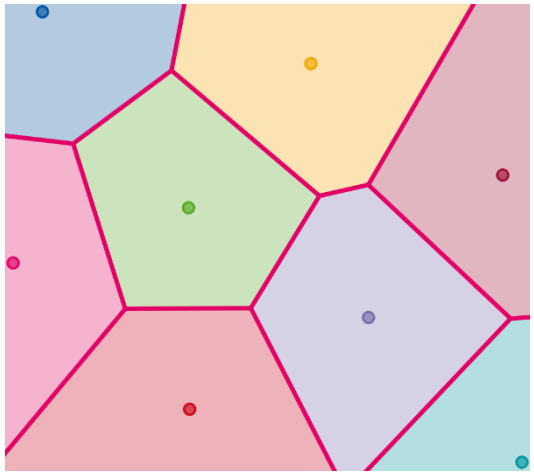


Delaunay Triangulation



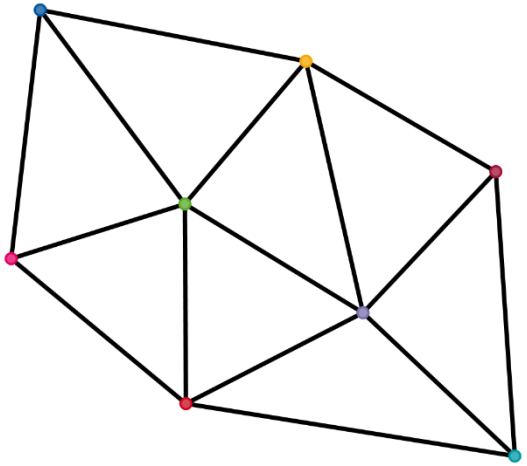
Delaunay Triangulation

Duality



Voronoi Diagram

DUAL



Delaunay Triangulation

Vertices ⟷ Faces

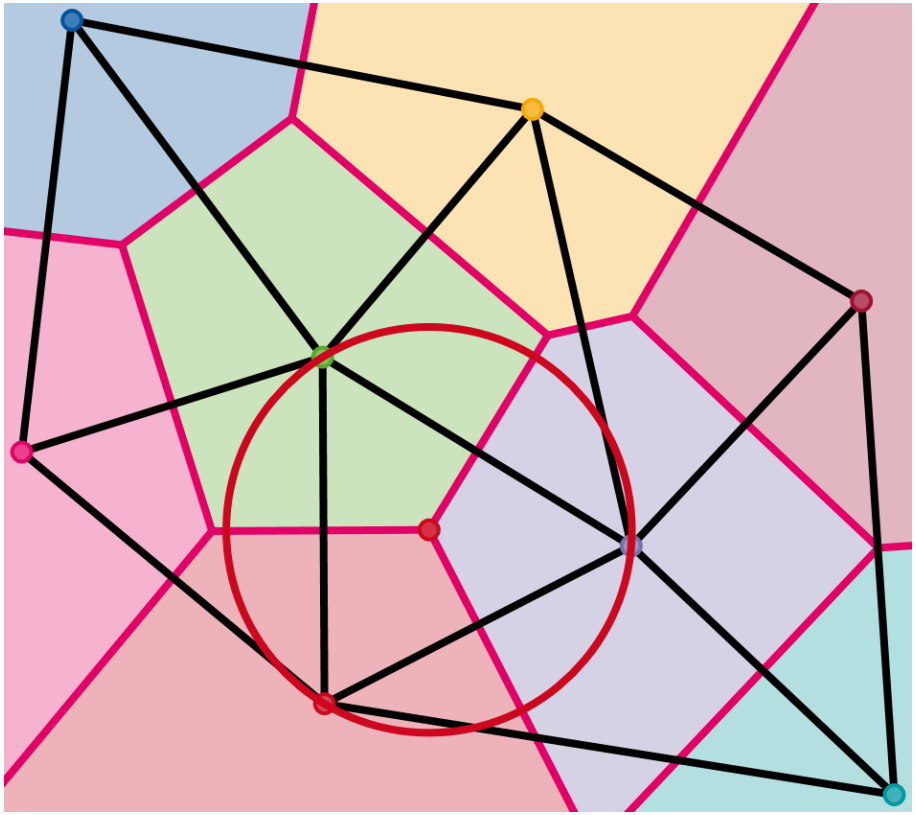
Faces ⟷ Vertices

Delaunay Property

Delaunay Property

For each face:

- Circumcircle contains no other point



Delaunay Triangulation

Properties

- Delaunay Triangulation 

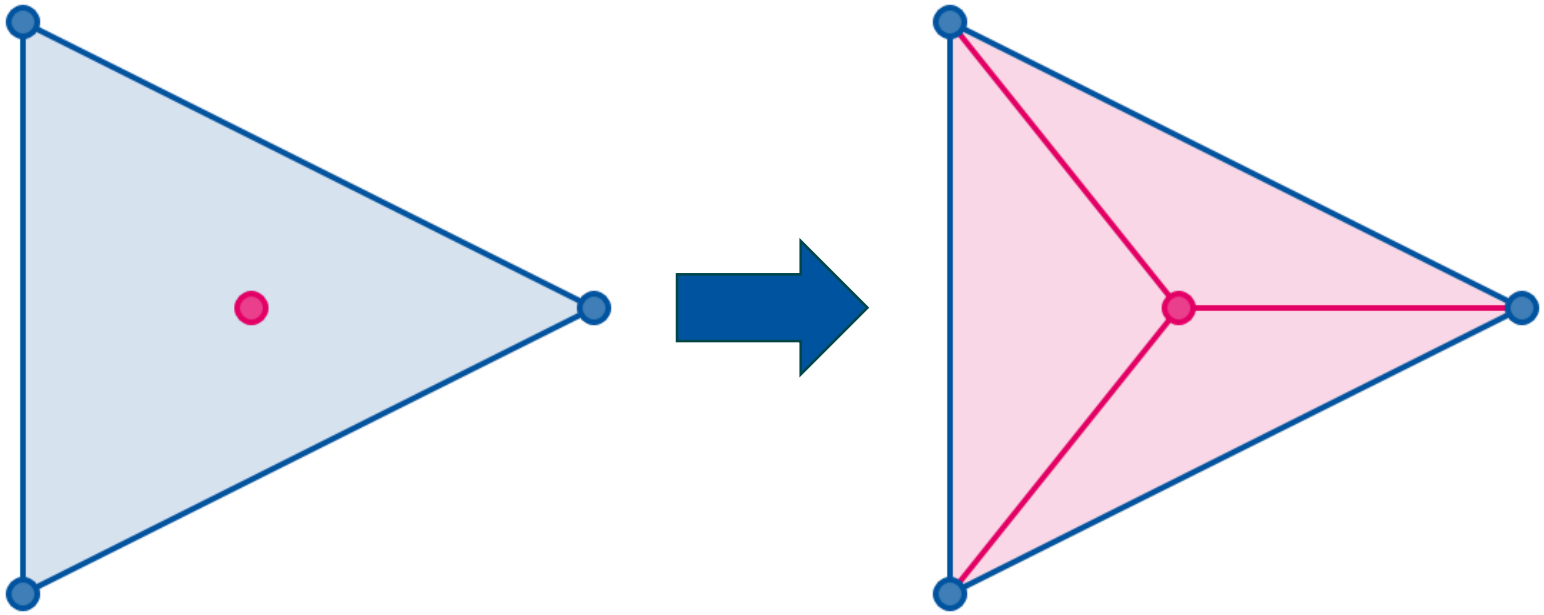
Voronoi Diagram
- Voronoi regions are convex
- The circumcircle of each triangle does not contain any other point p_j
- The Delaunay Triangulation is **uniquely** defined
- **The minimal inside angle of all triangles is maximized**

Delaunay Triangulation: Incremental Algorithm

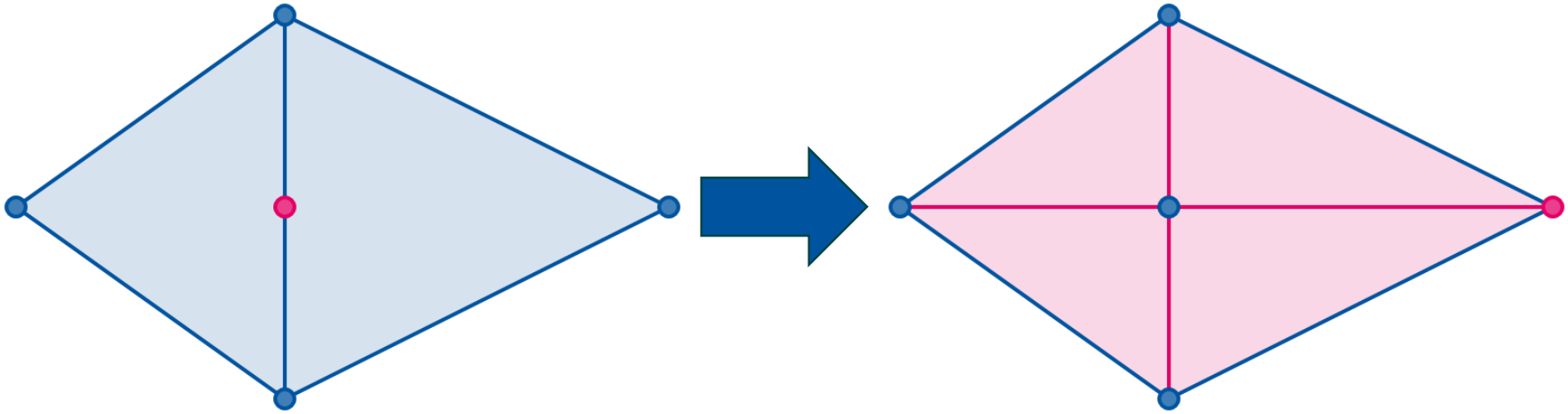
Algorithm for creation of Delaunay Triangulations

- Can handle any number of points
- Add points one after each other
- Restore Delaunay Property after each operation
- Operations: 1-3 Split, 2-4 Split, Edge Flipping

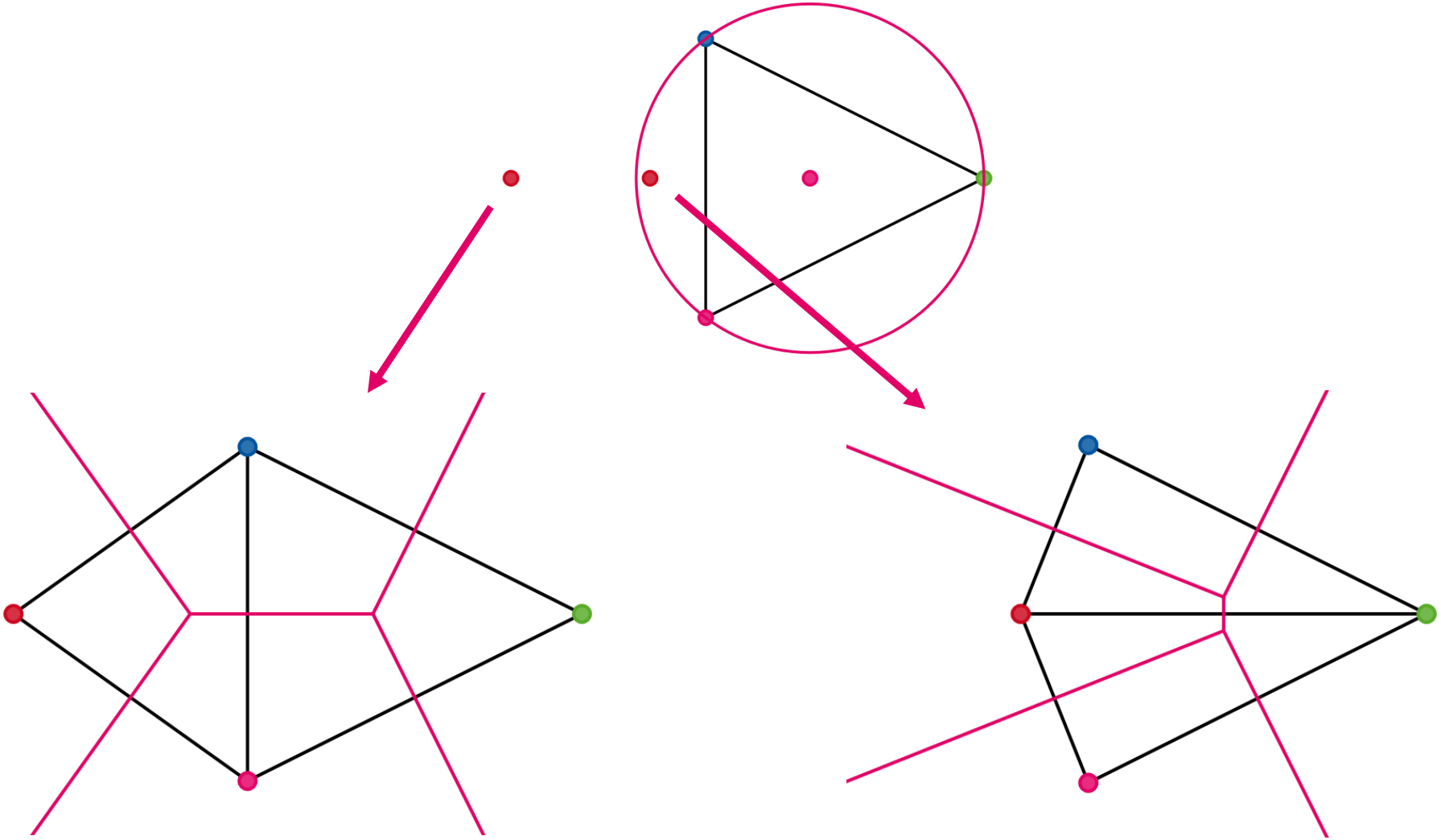
Operation: 1-3 Split



Operation: 2-4 Split



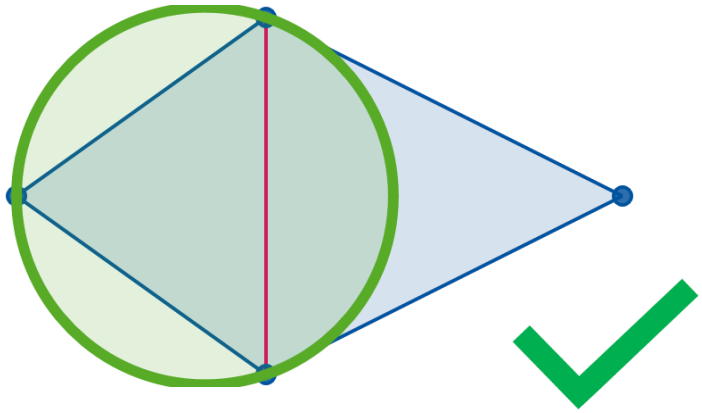
Operation: Edge Flip

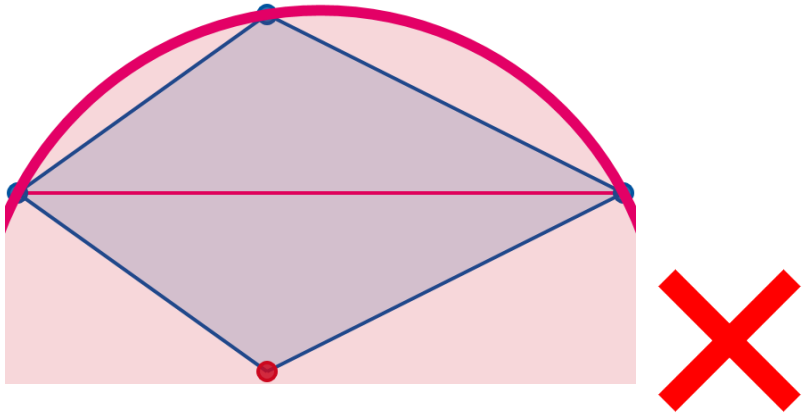


Operation: Edge Flip

Edge Flipping

- Operation on a triangle mesh
- Change topology
- In one configuration the circumcircle is empty *(Delaunay Property)*





Delaunay Triangulation: Incremental Algorithm

Input: Set of points $p_1, \dots, p_n$

Output: Delaunay triangulation of input points

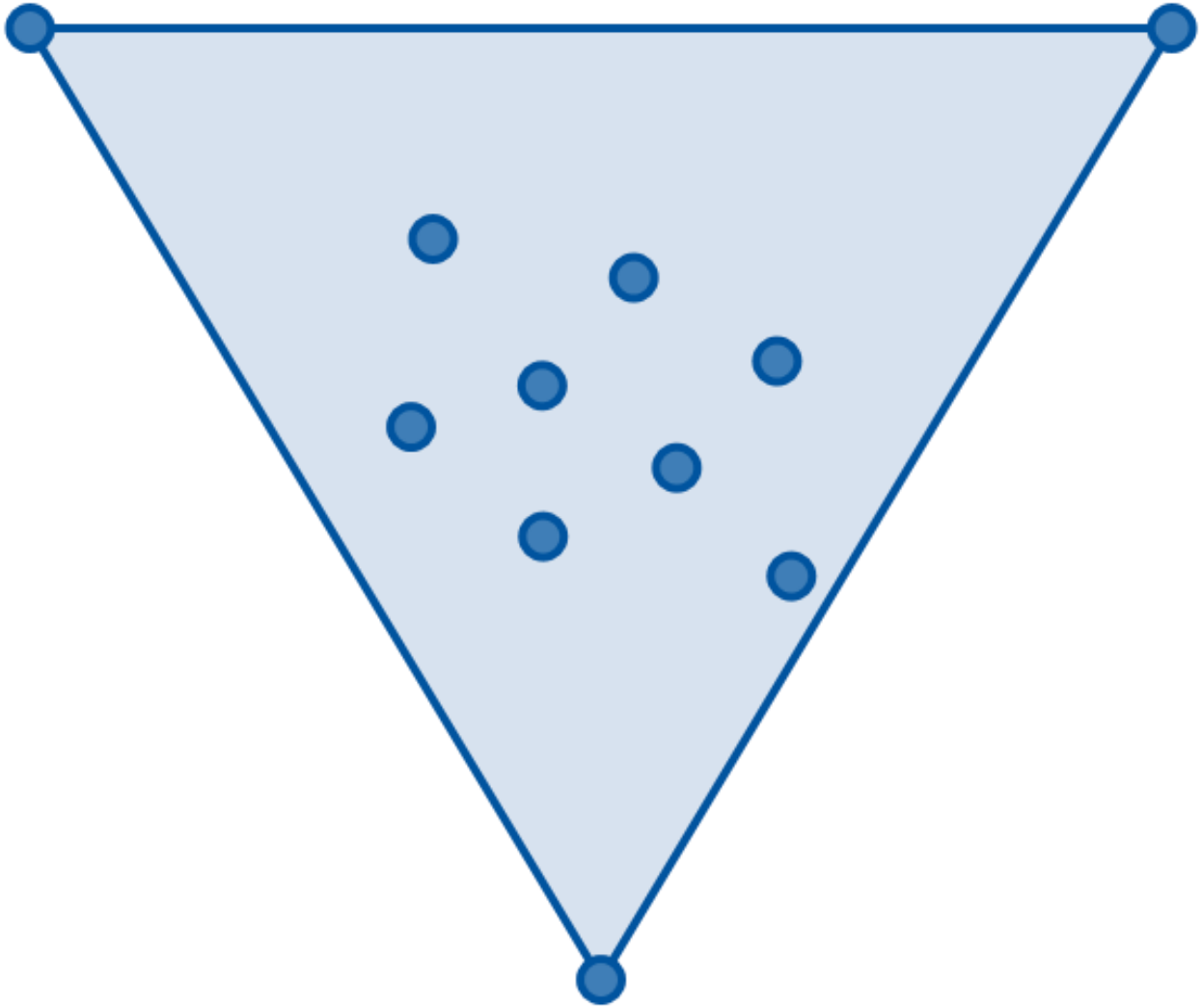
Algorithm:

1. Add three points q_1, q_2, q_3, such that the triangle $[q_1, q_2, q_3]$ contains all points $p_1, \dots, p_n$

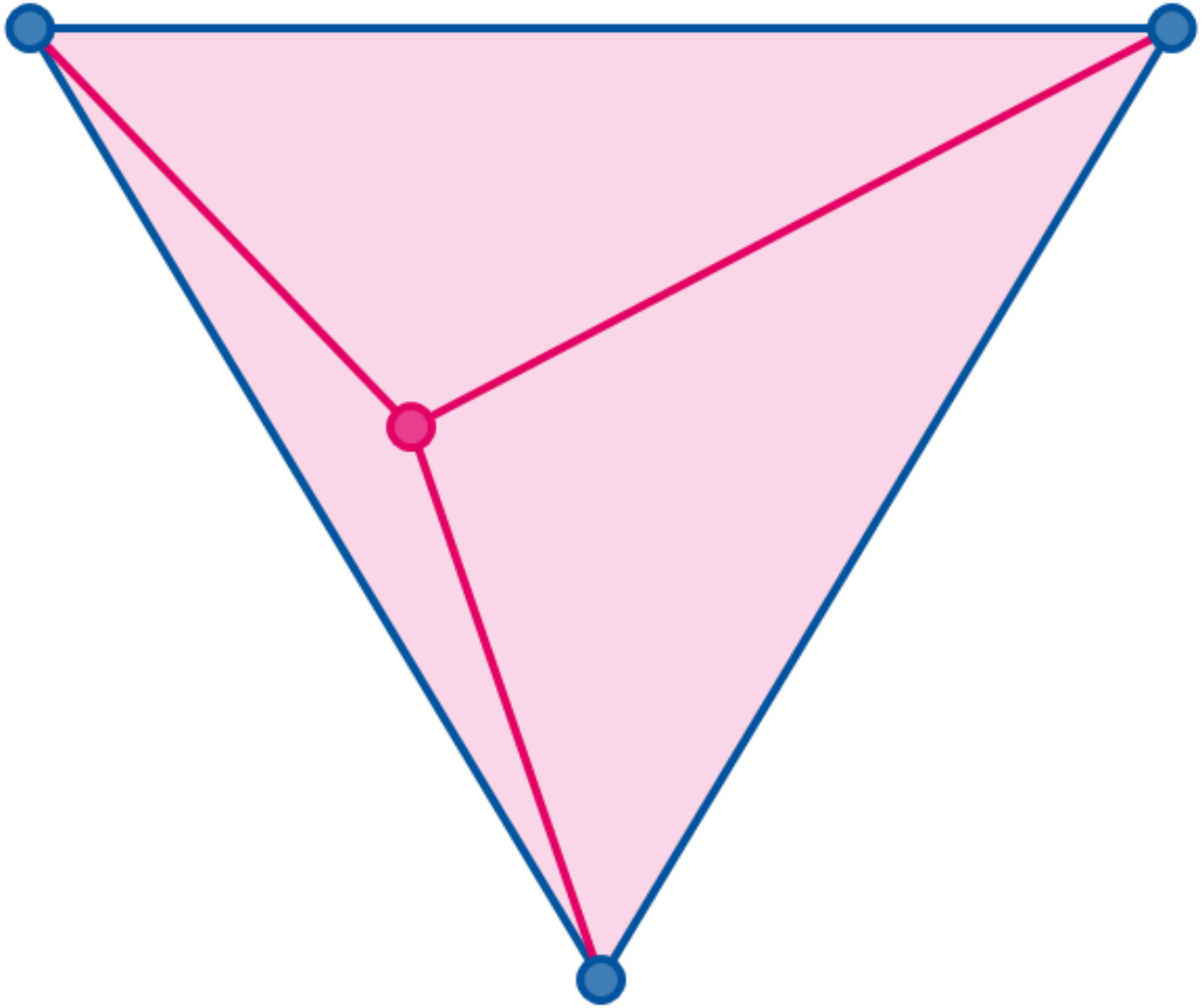
2. For each point $p_i \in p_1, \dots, p_n$
   a) Find the triangle which contains p_i
   b) Split the triangle (or edge) at position p_i
   c) Restore Delaunay property using edge flips

3. Remove the extra points q_1, q_2, q_3 and the adjacent edges

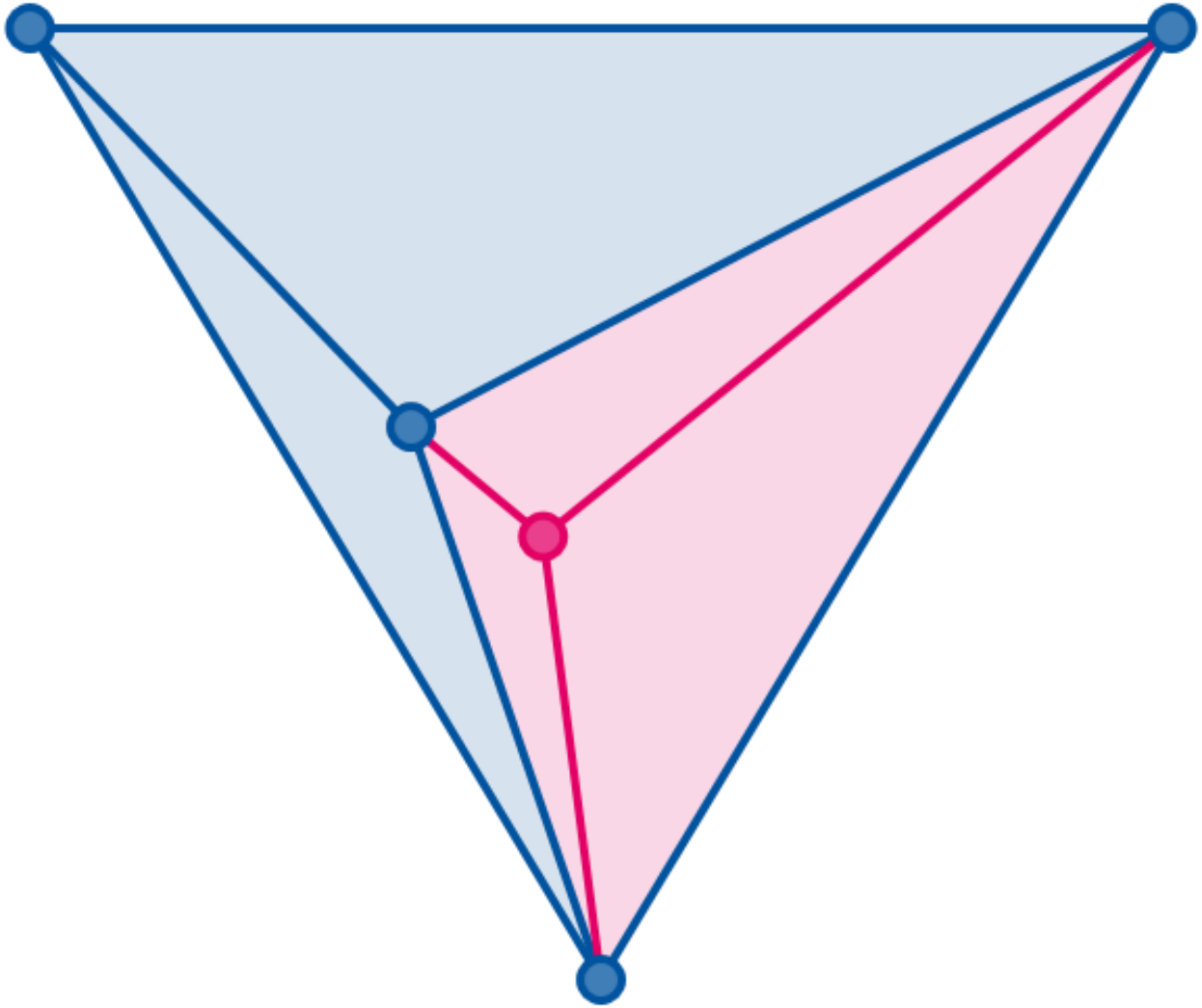
Delaunay Triangulation: Incremental Algorithm



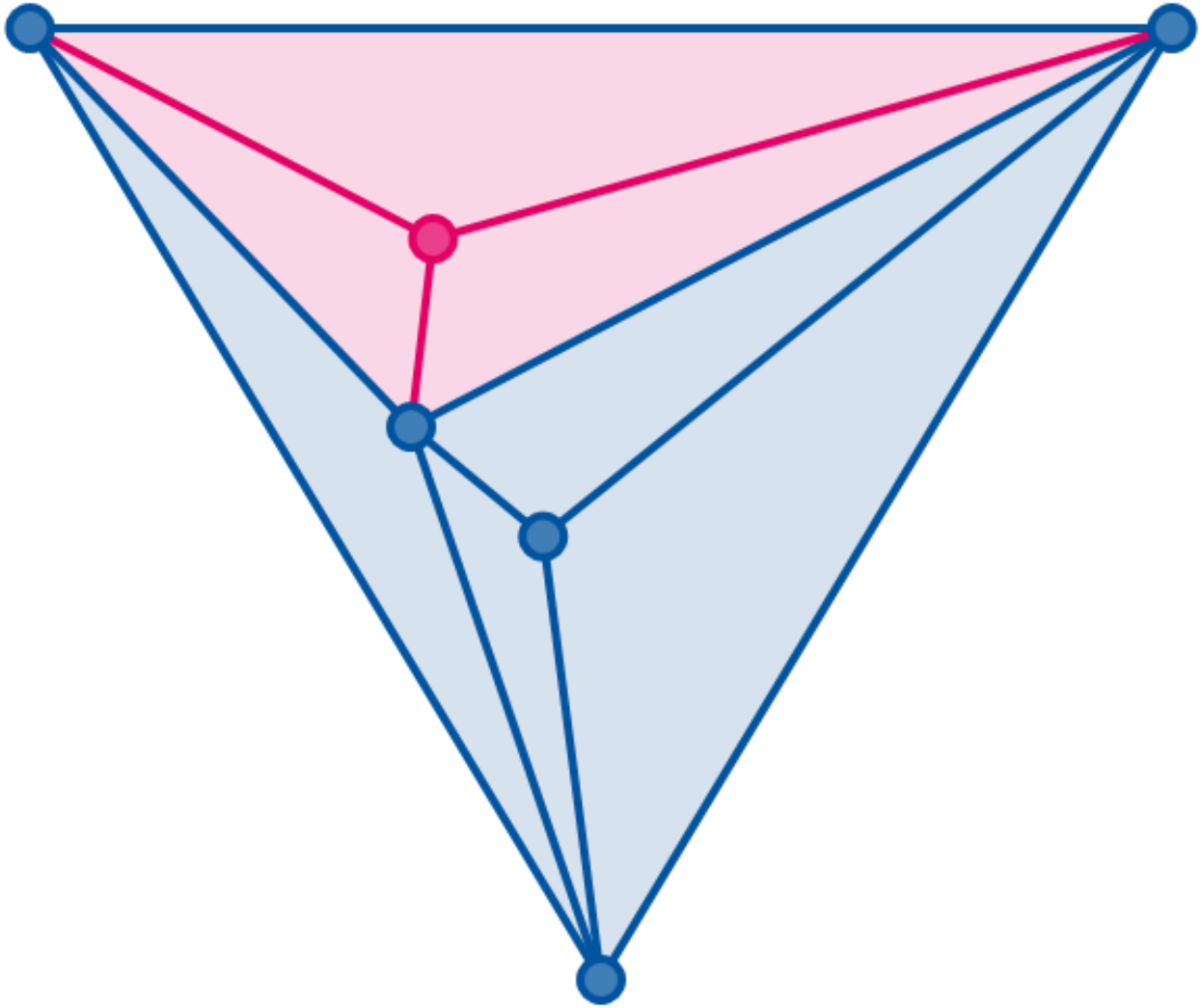
Delaunay Triangulation: Incremental Algorithm



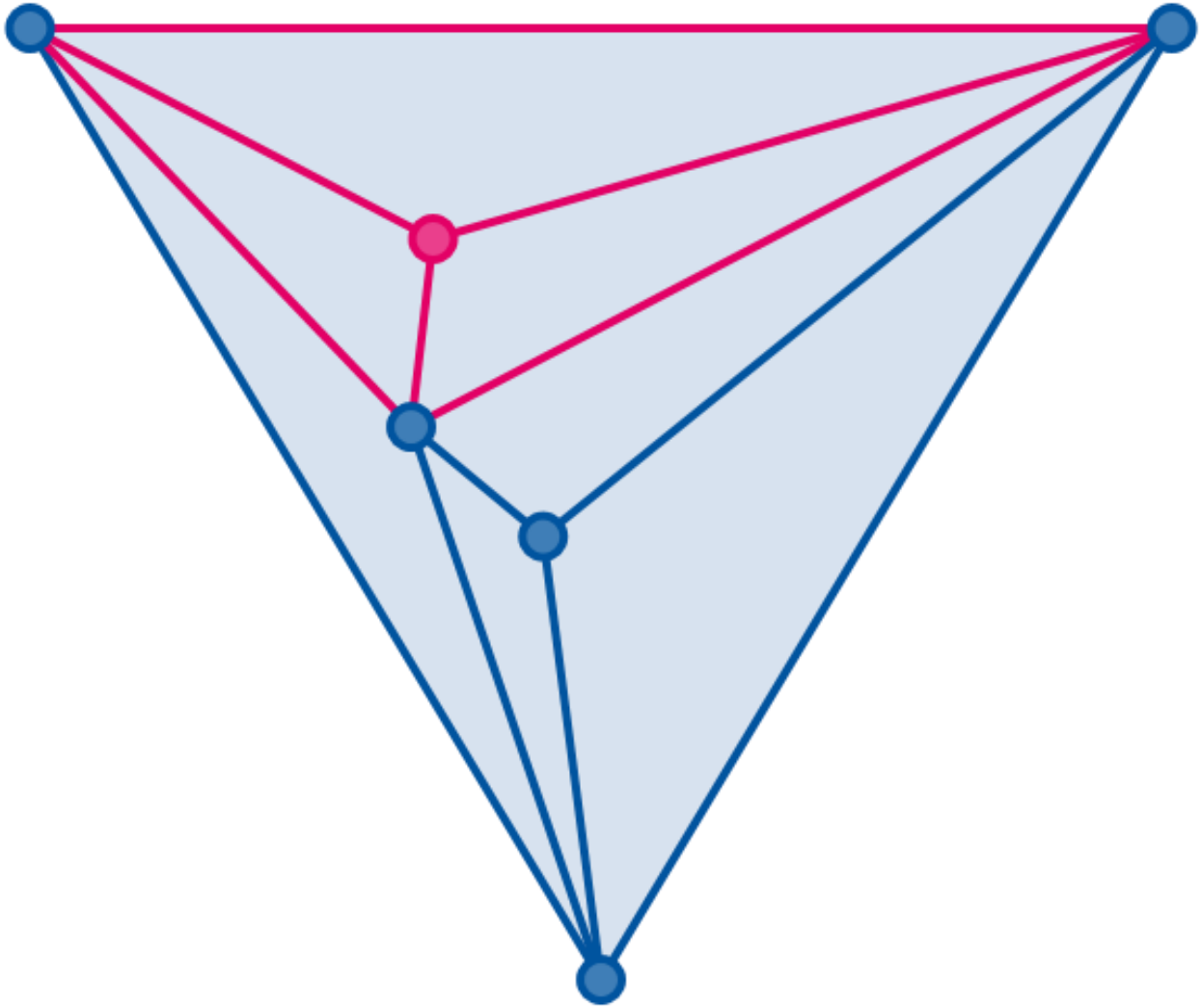
Delaunay Triangulation: Incremental Algorithm



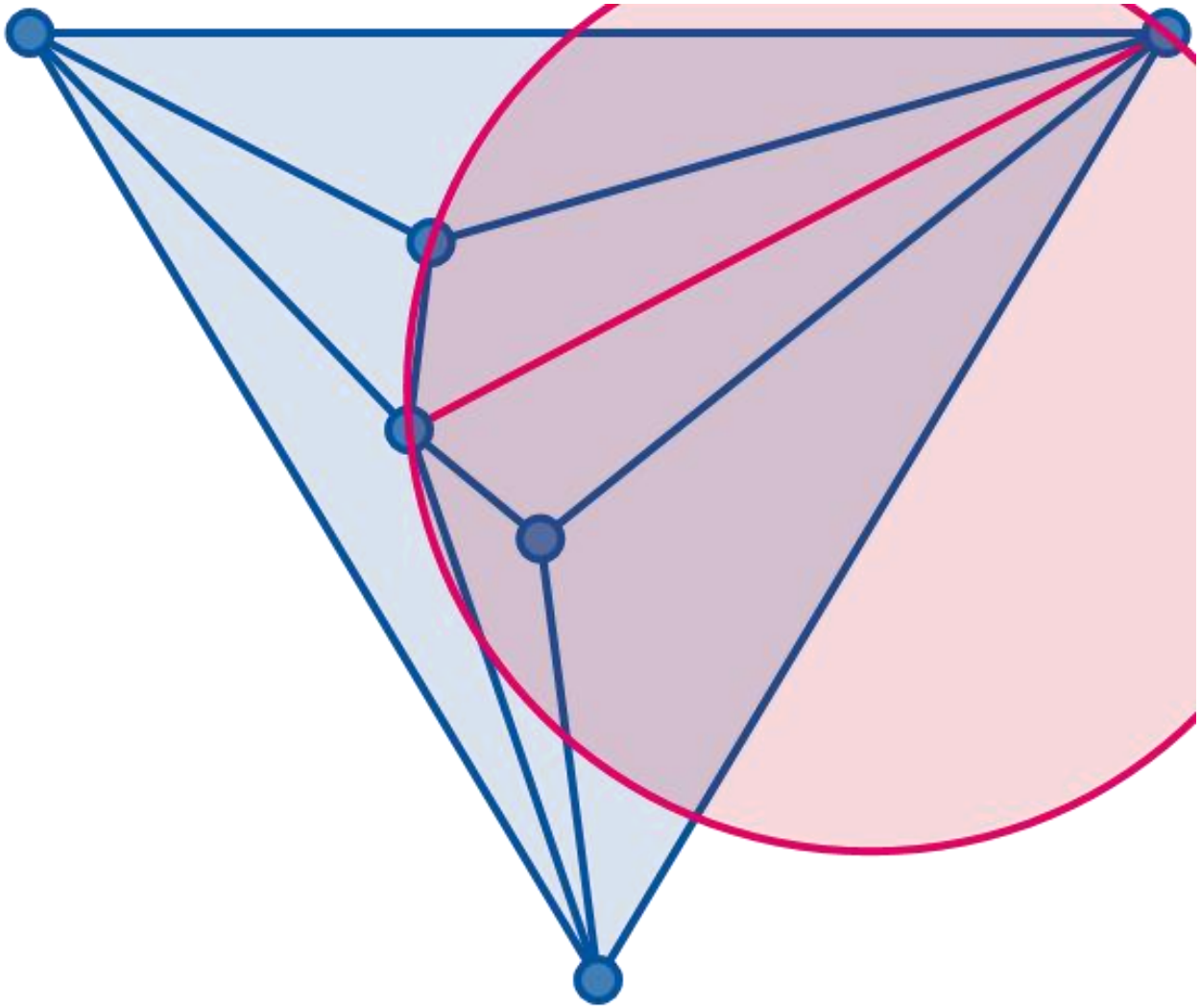
Delaunay Triangulation: Incremental Algorithm



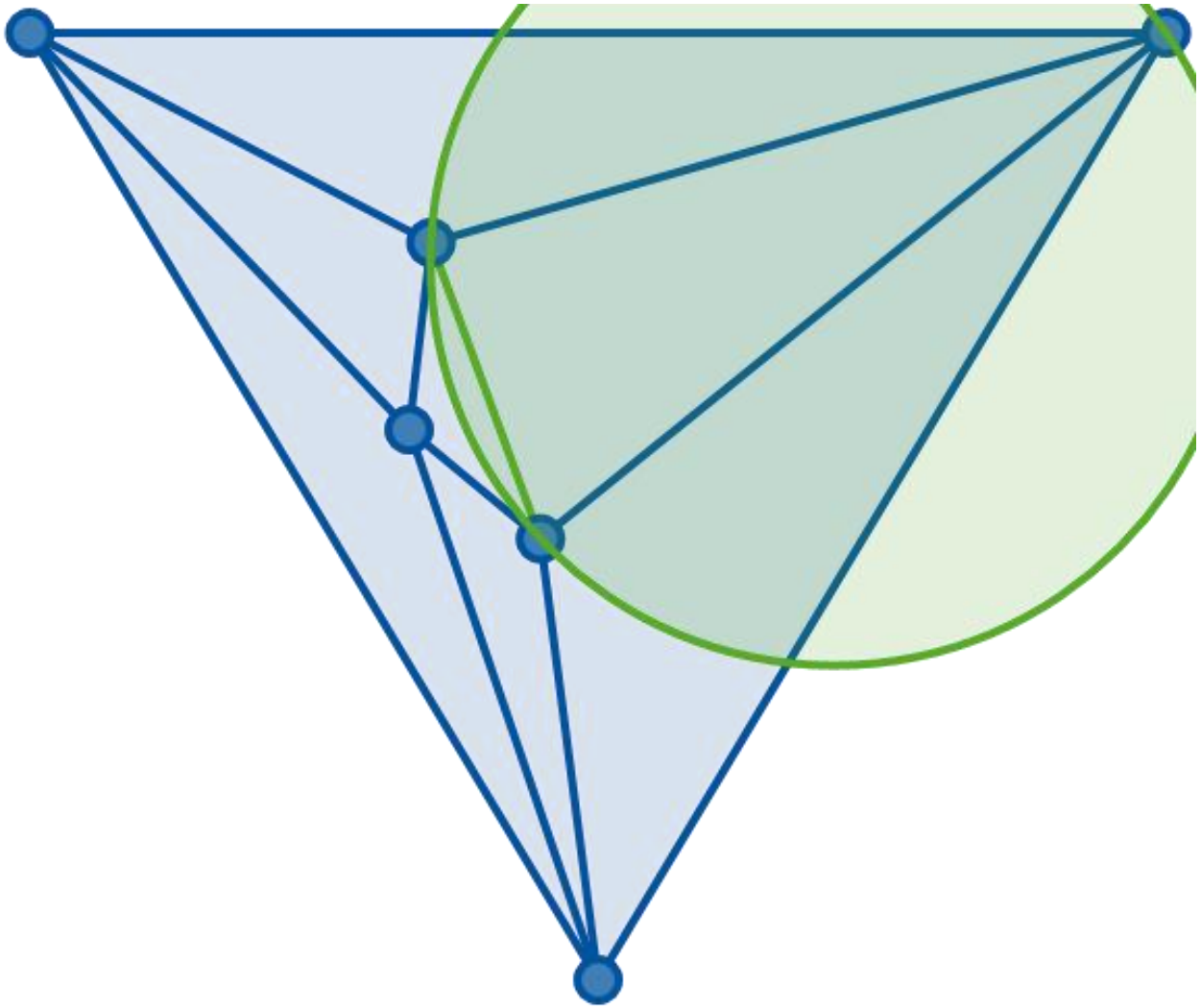
Delaunay Triangulation: Incremental Algorithm



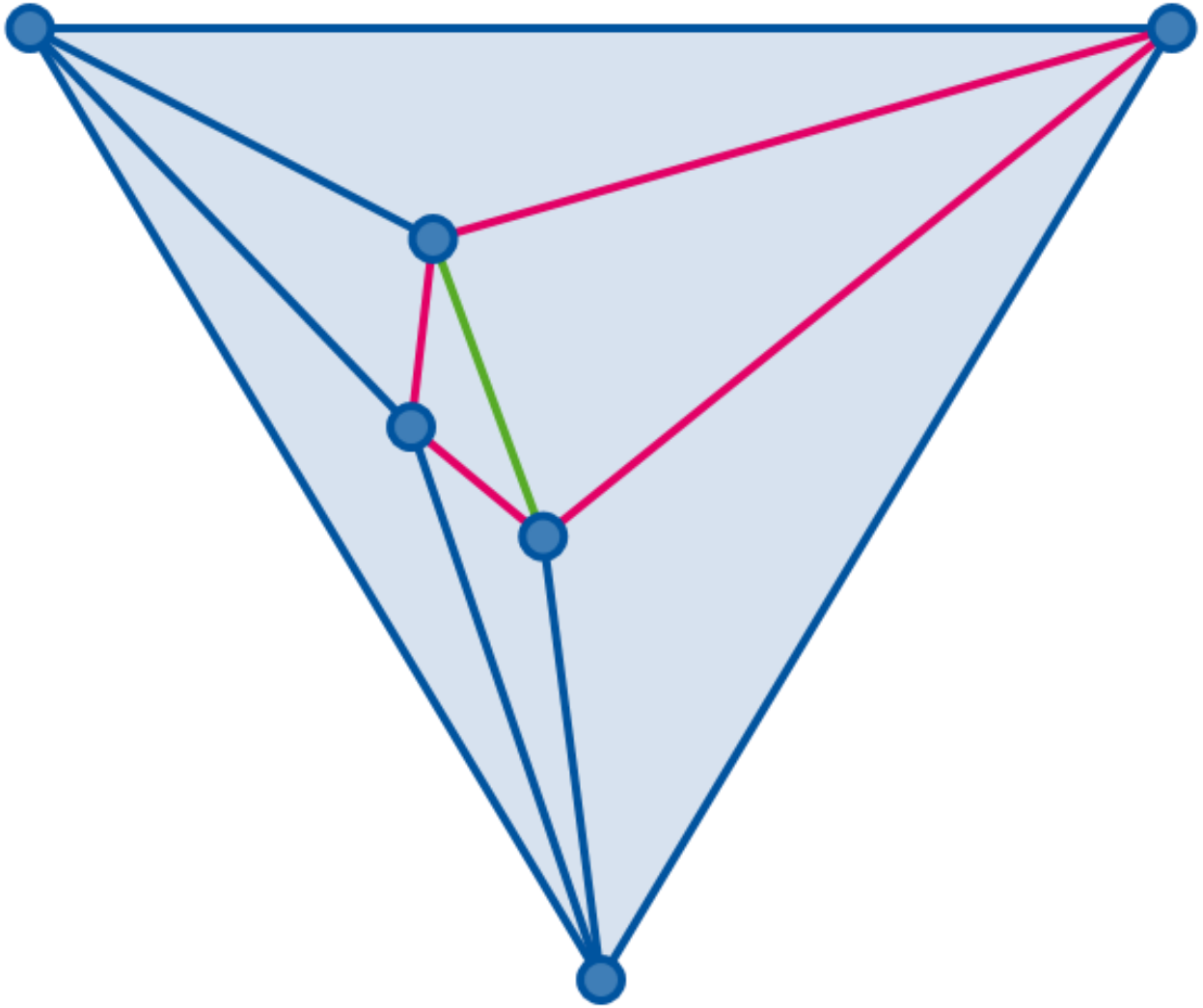
Delaunay Triangulation: Incremental Algorithm



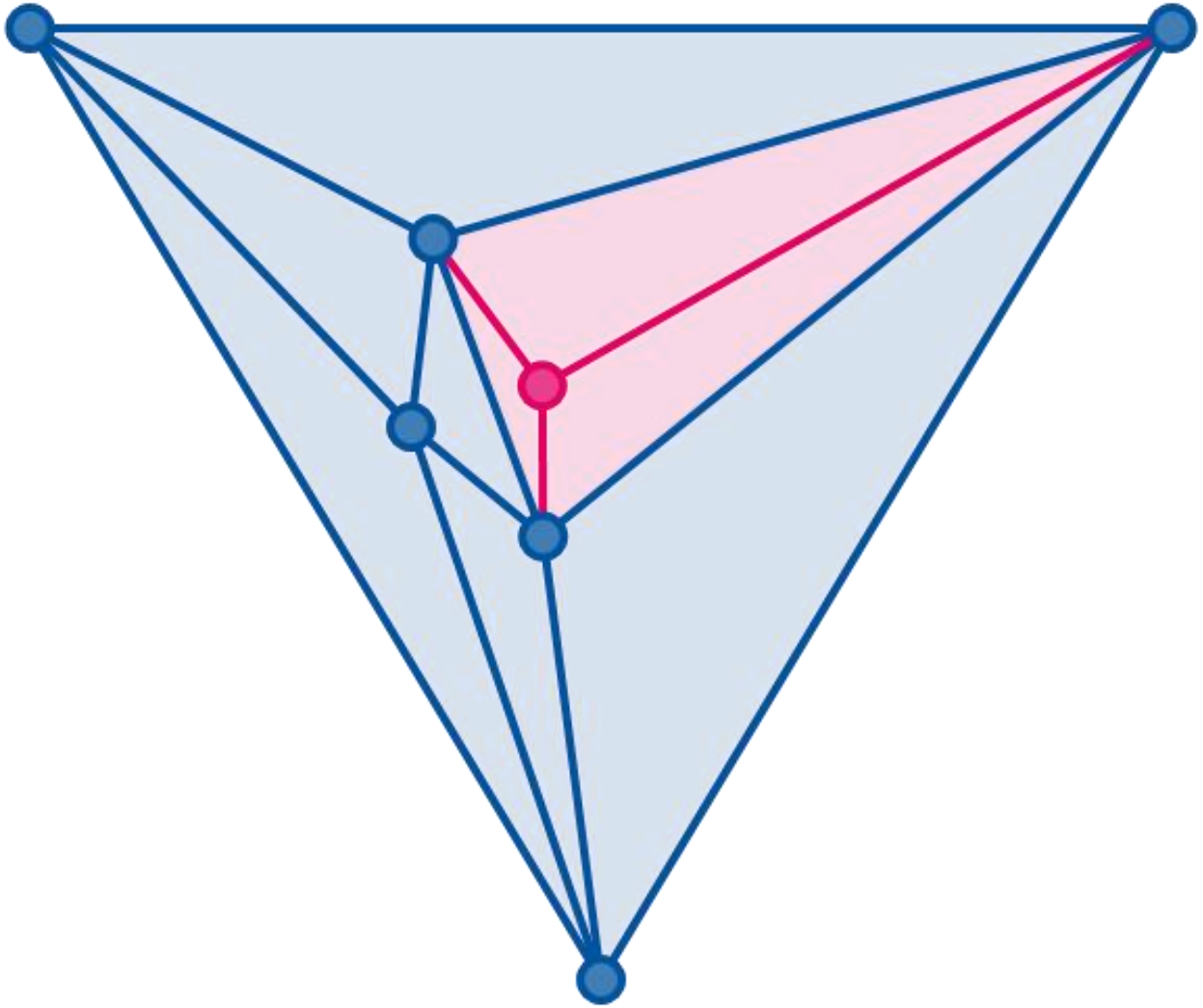
Delaunay Triangulation: Incremental Algorithm



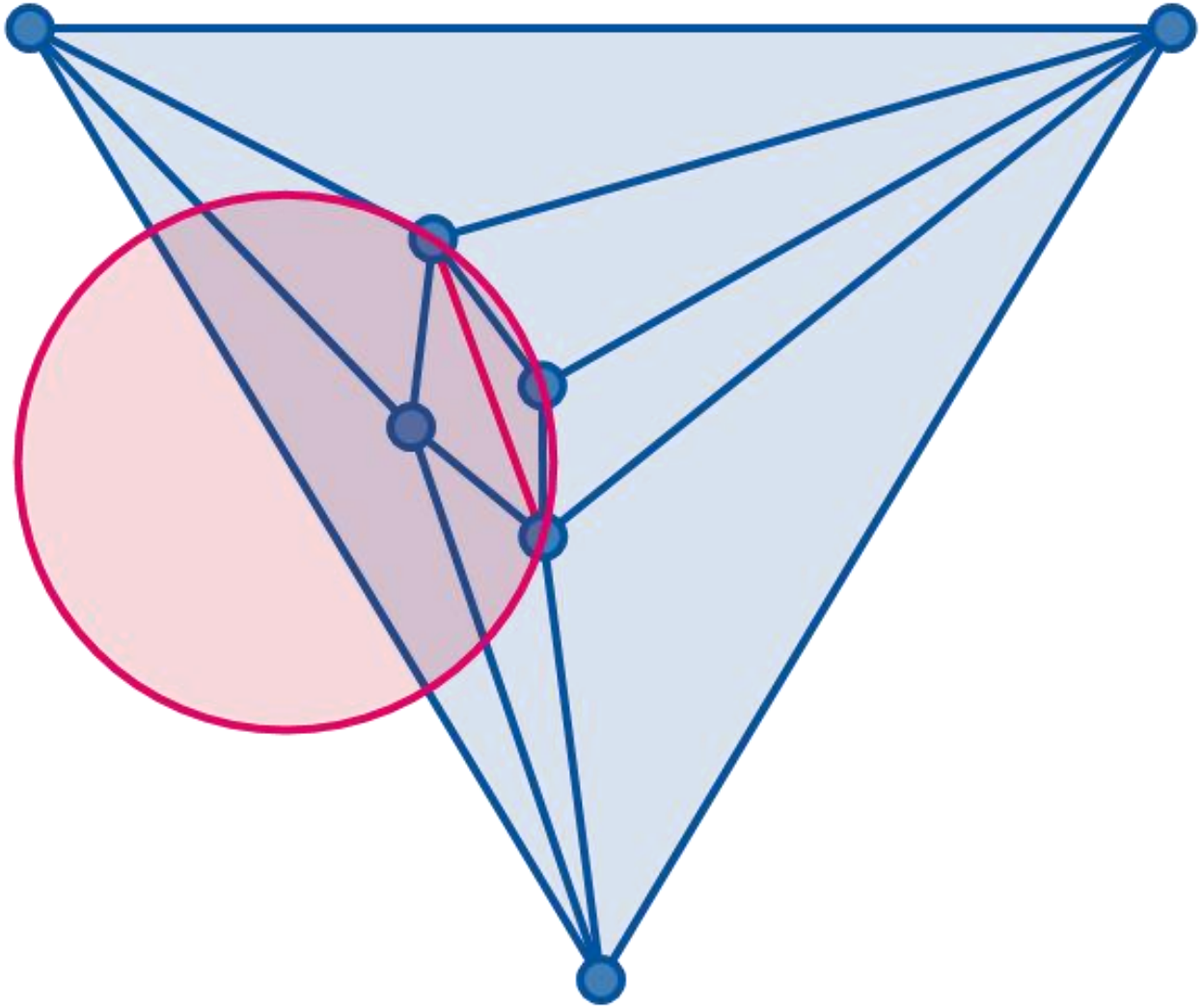
Delaunay Triangulation: Incremental Algorithm



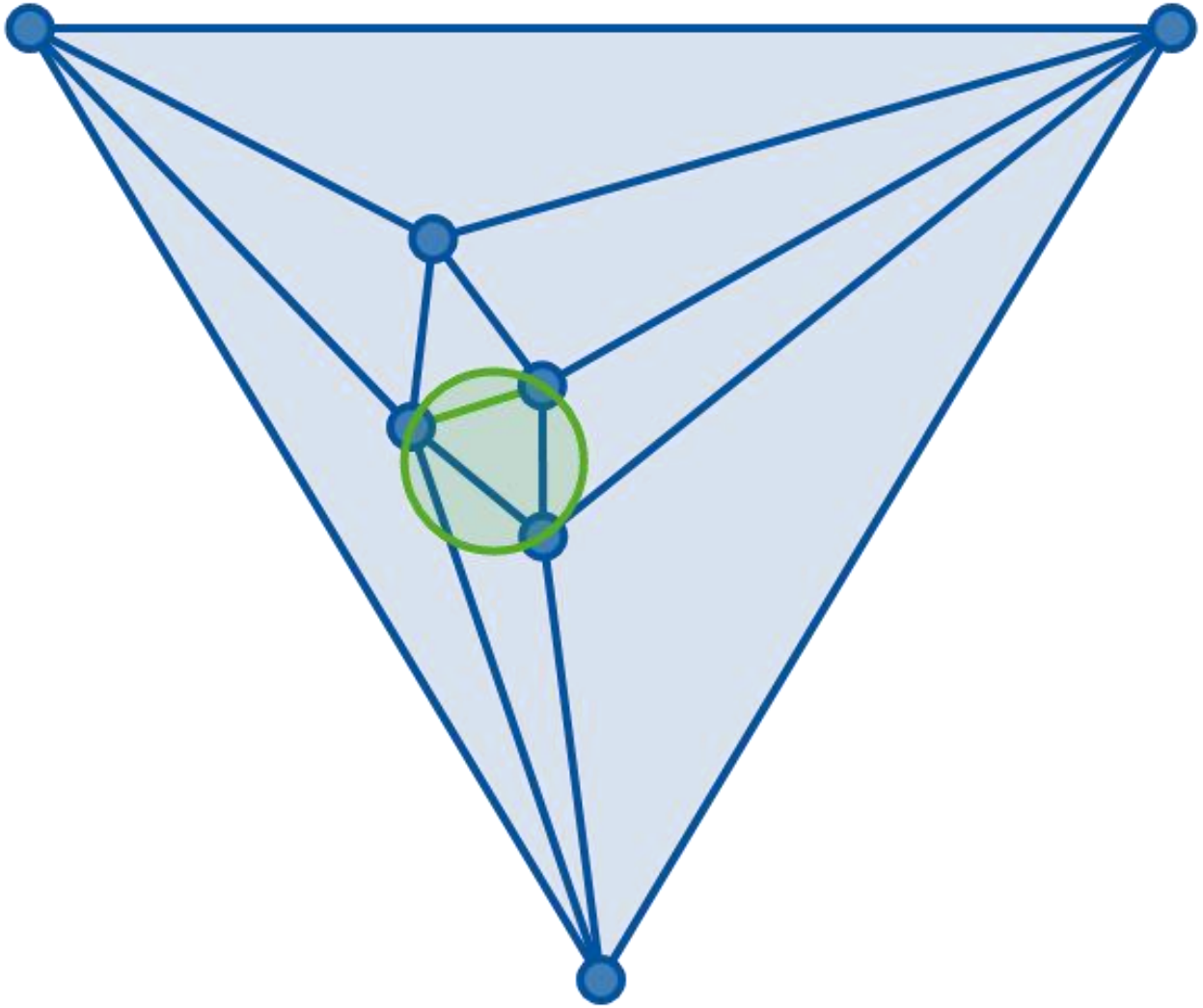
Delaunay Triangulation: Incremental Algorithm



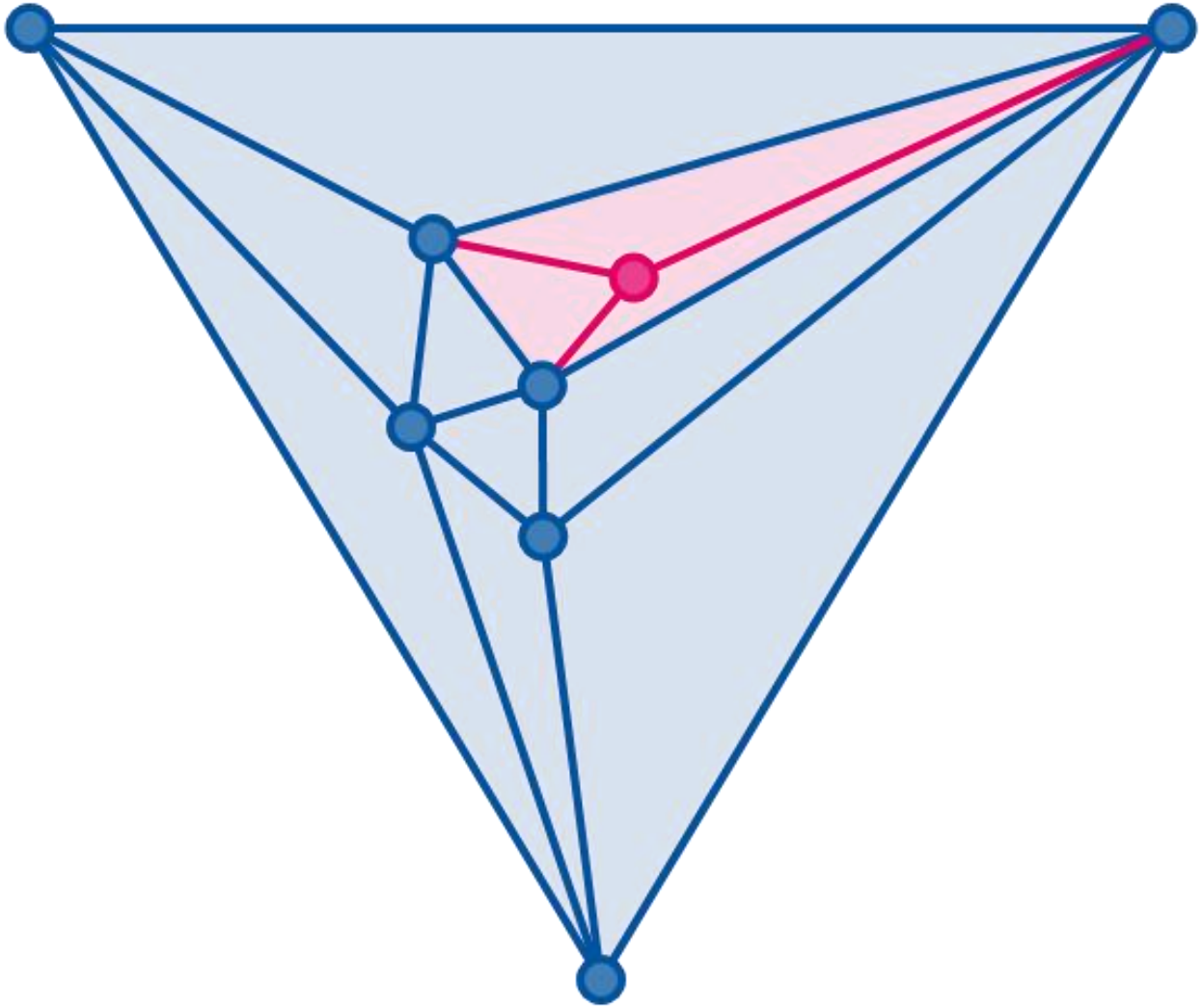
Delaunay Triangulation: Incremental Algorithm



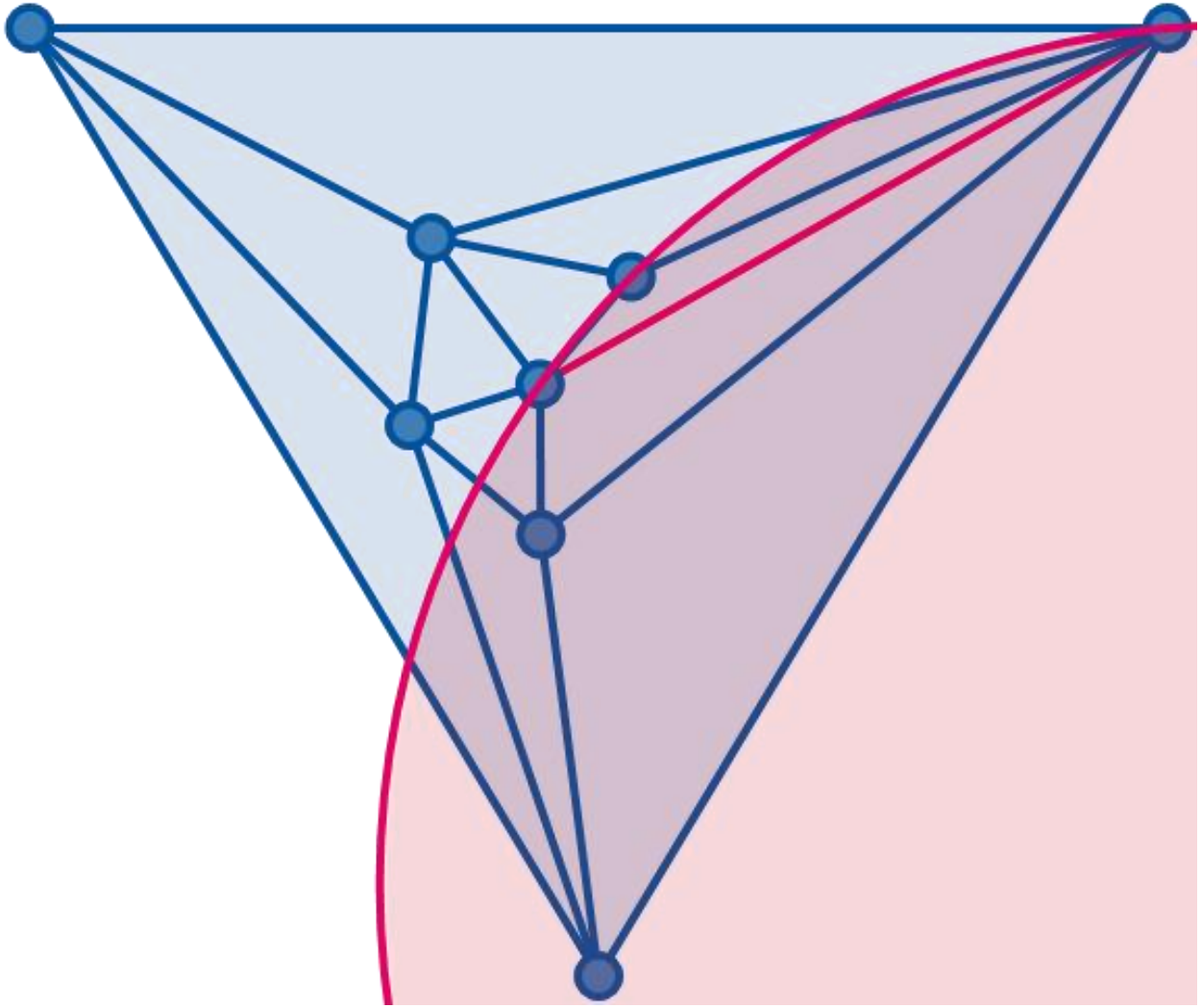
Delaunay Triangulation: Incremental Algorithm



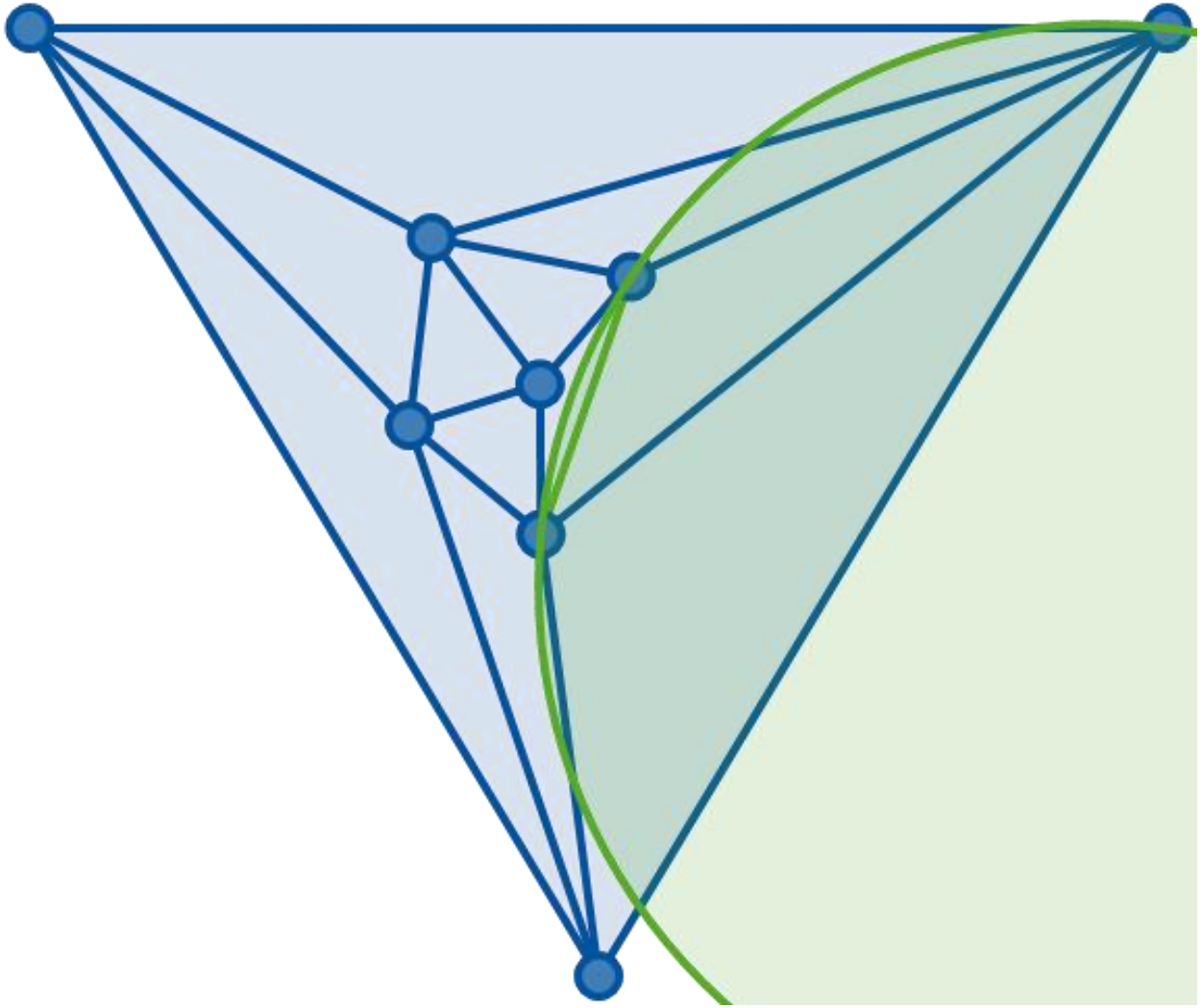
Delaunay Triangulation: Incremental Algorithm



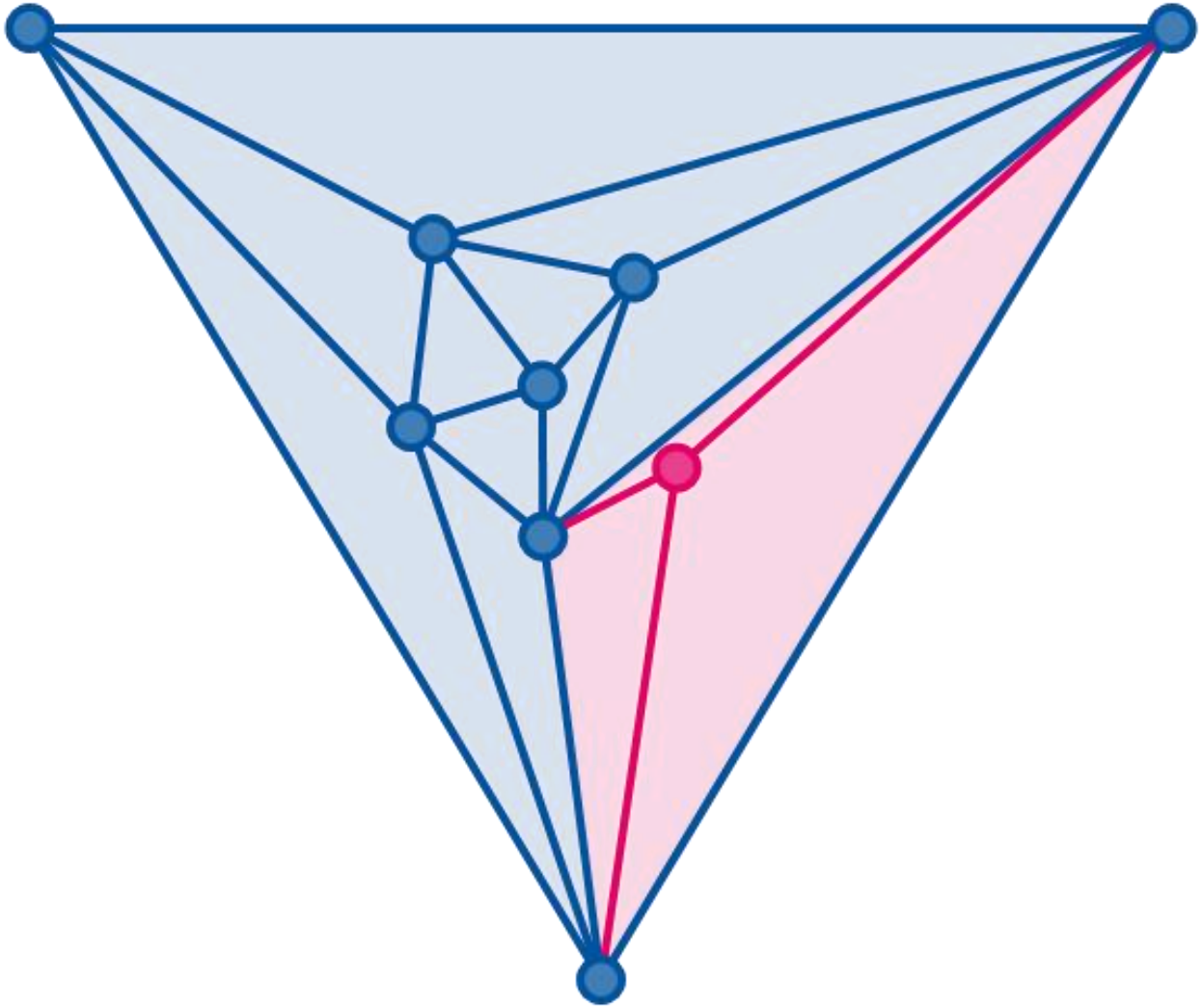
Delaunay Triangulation: Incremental Algorithm



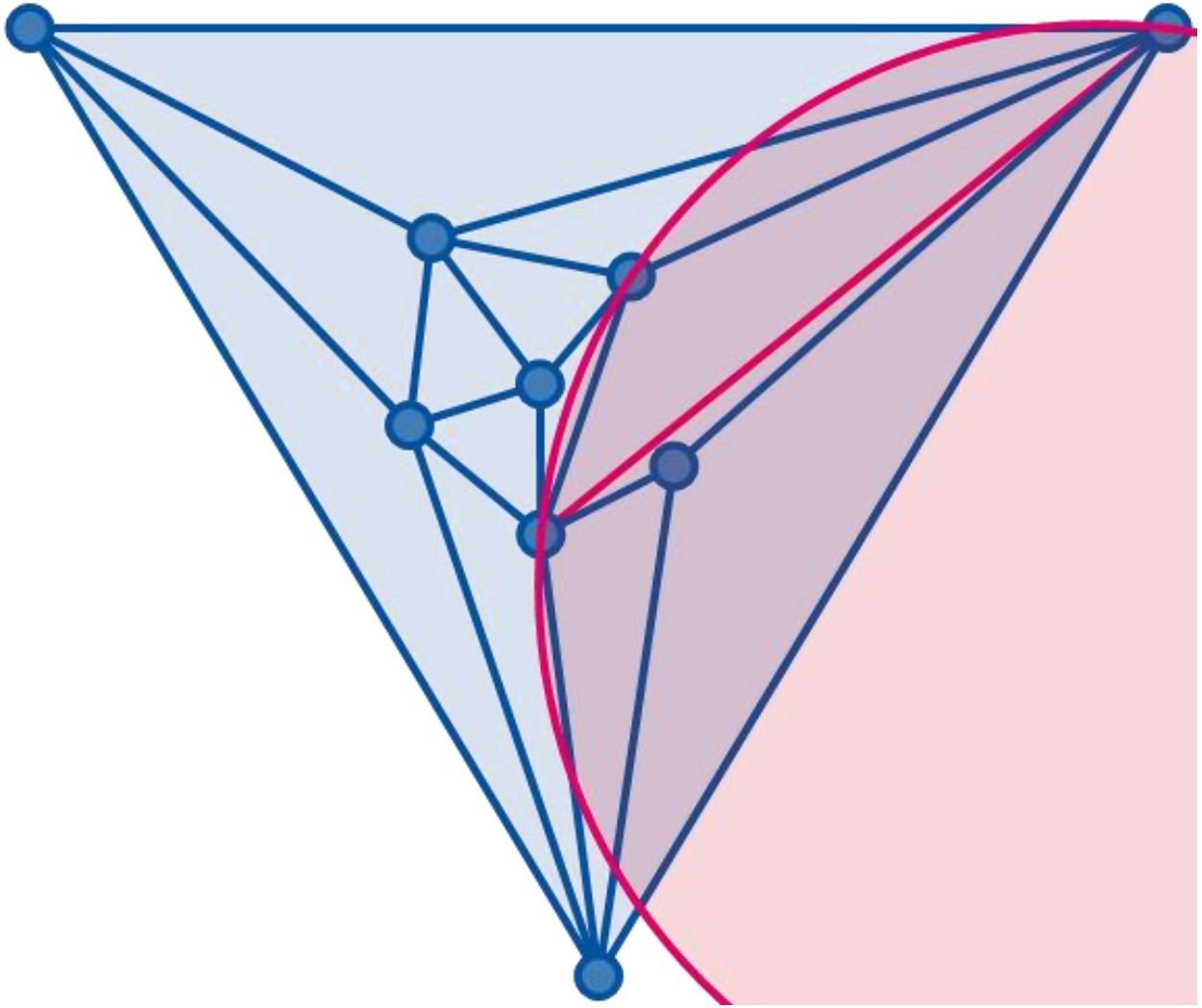
Delaunay Triangulation: Incremental Algorithm



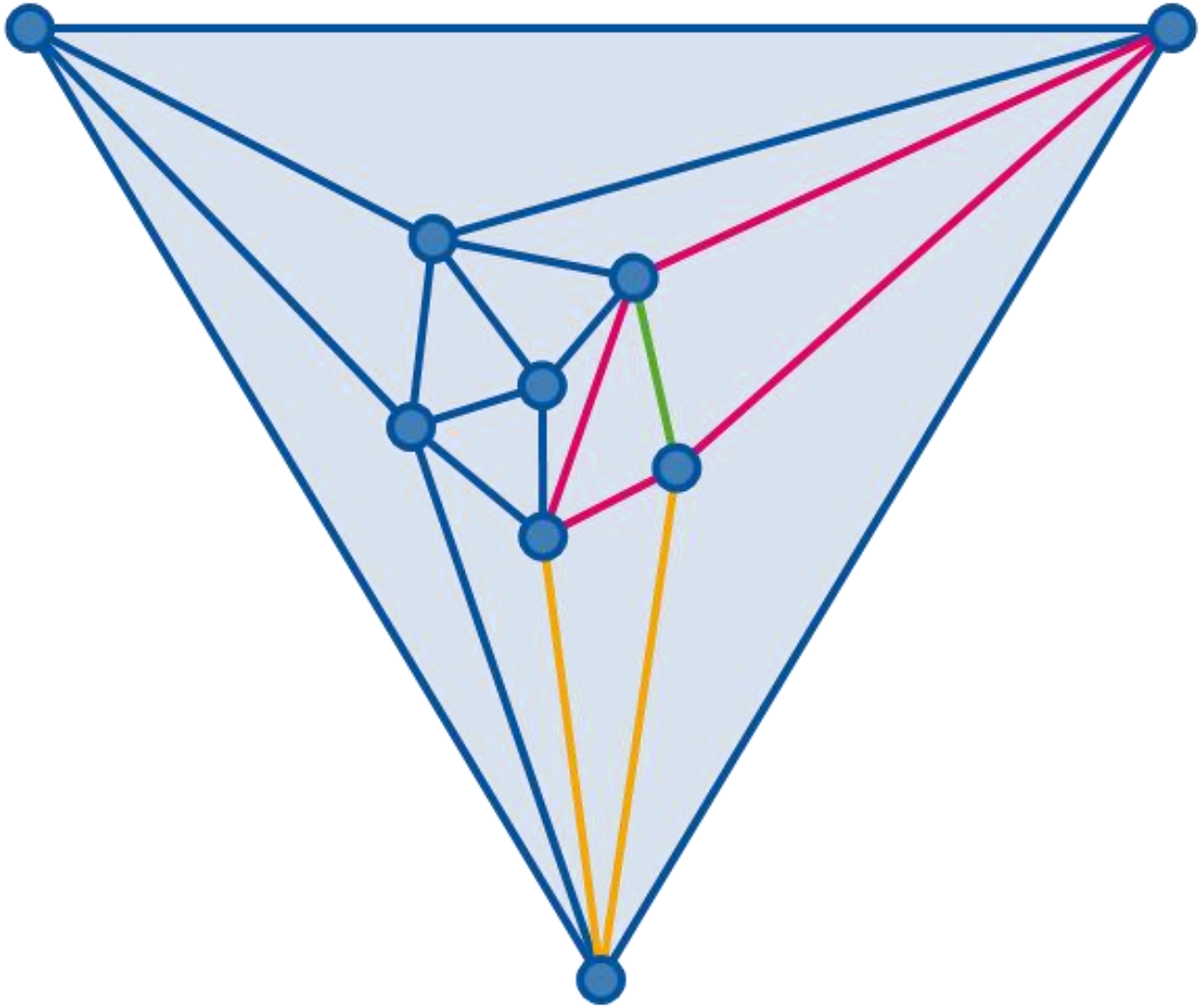
Delaunay Triangulation: Incremental Algorithm



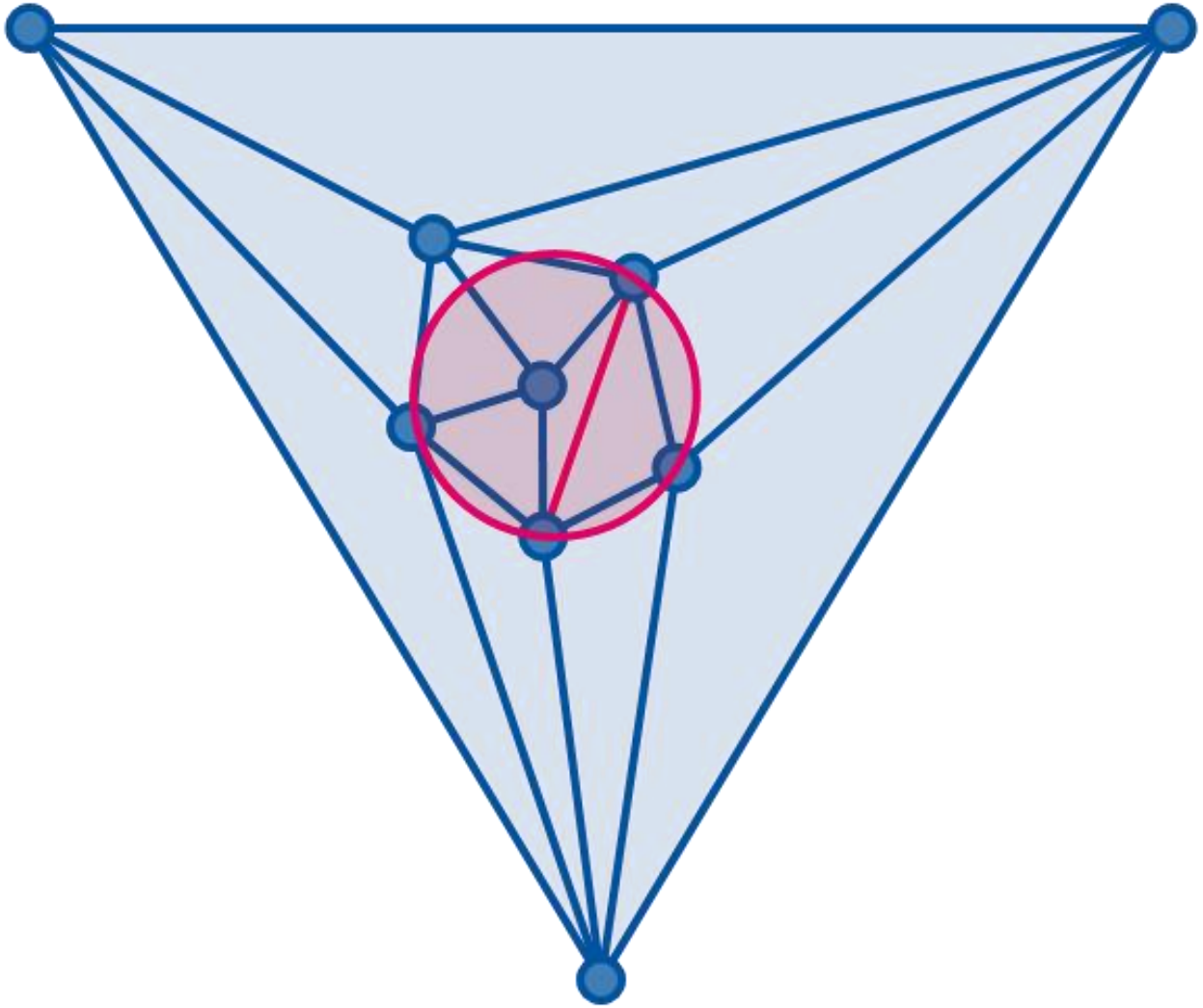
Delaunay Triangulation: Incremental Algorithm



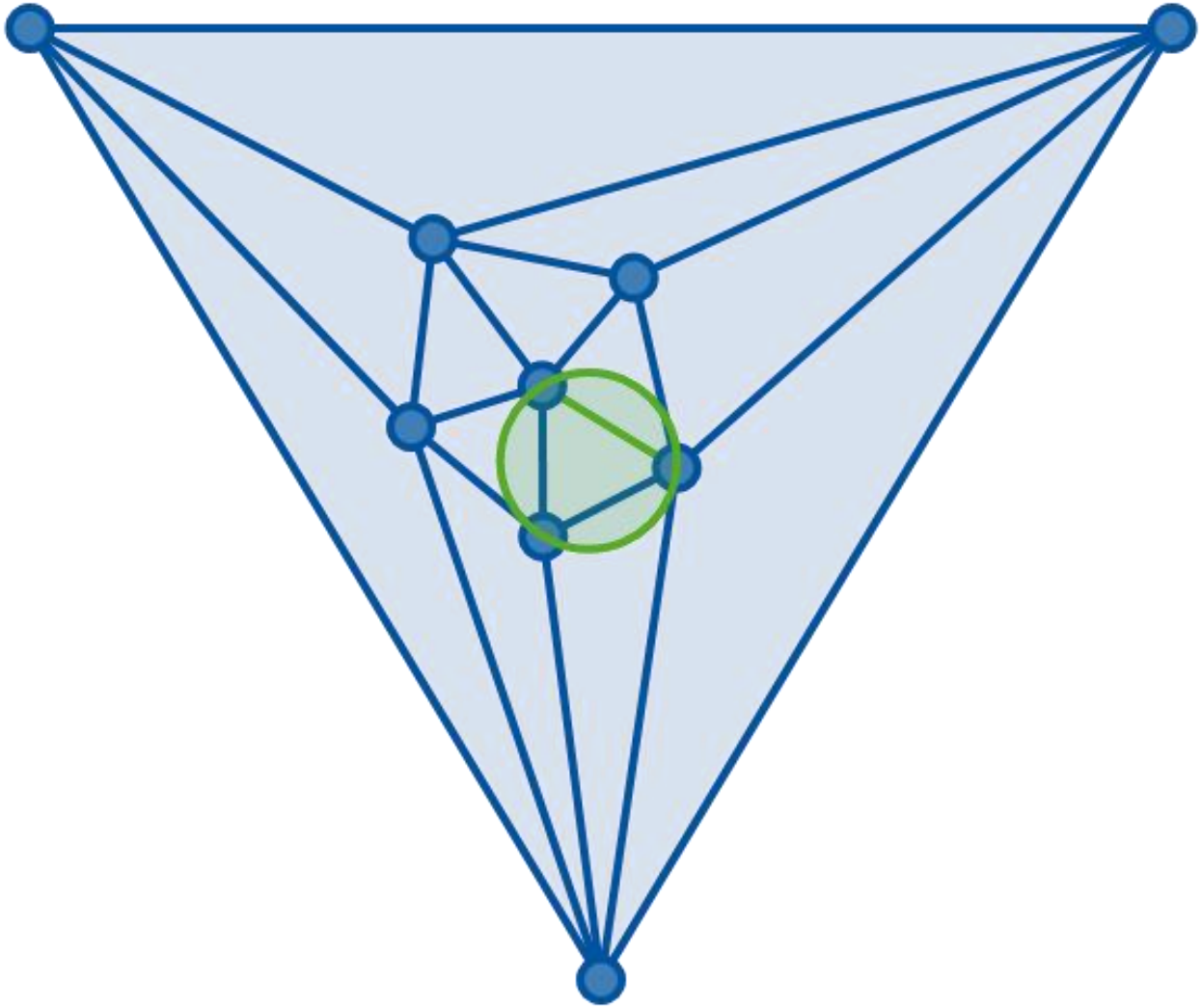
Delaunay Triangulation: Incremental Algorithm



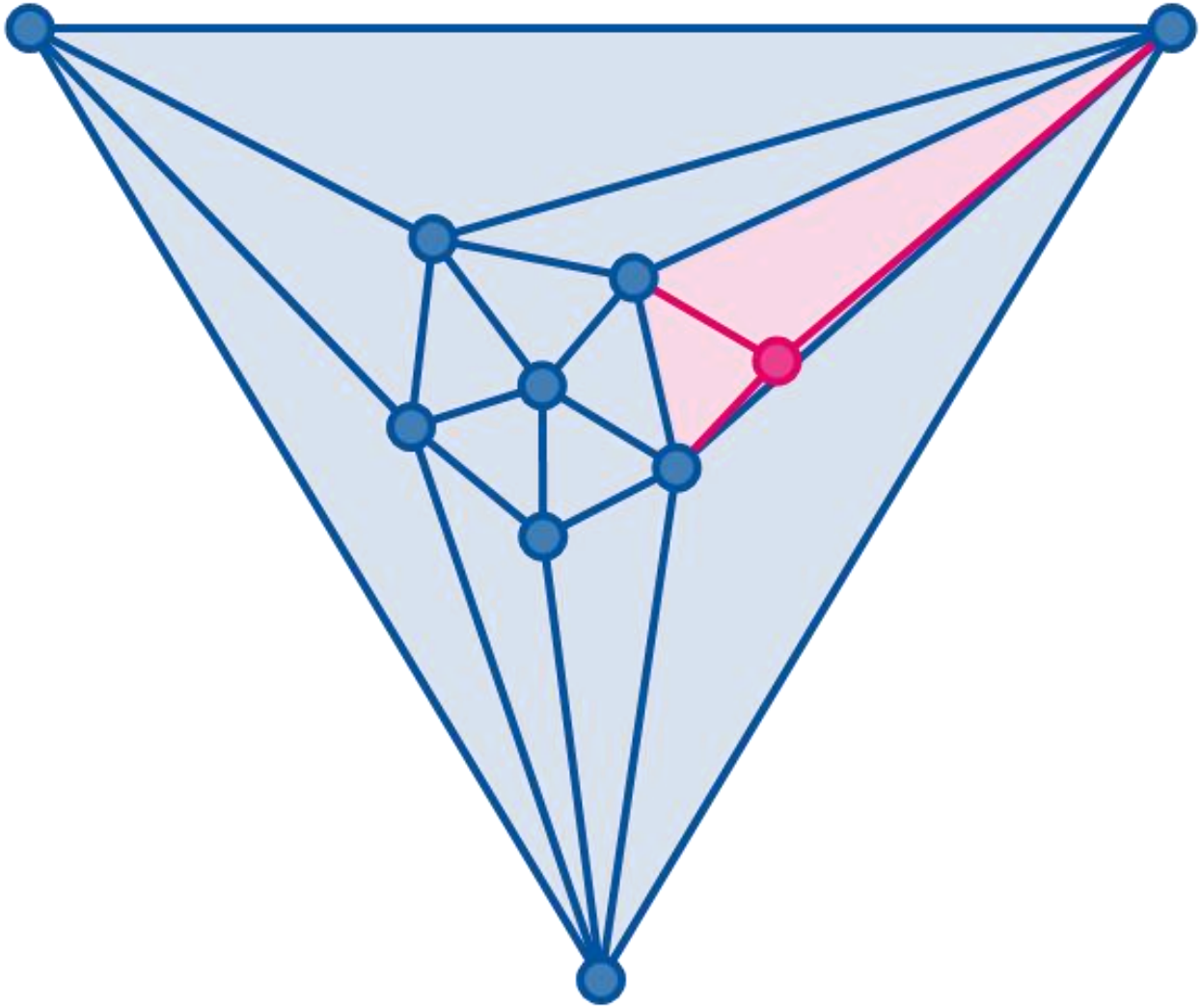
Delaunay Triangulation: Incremental Algorithm



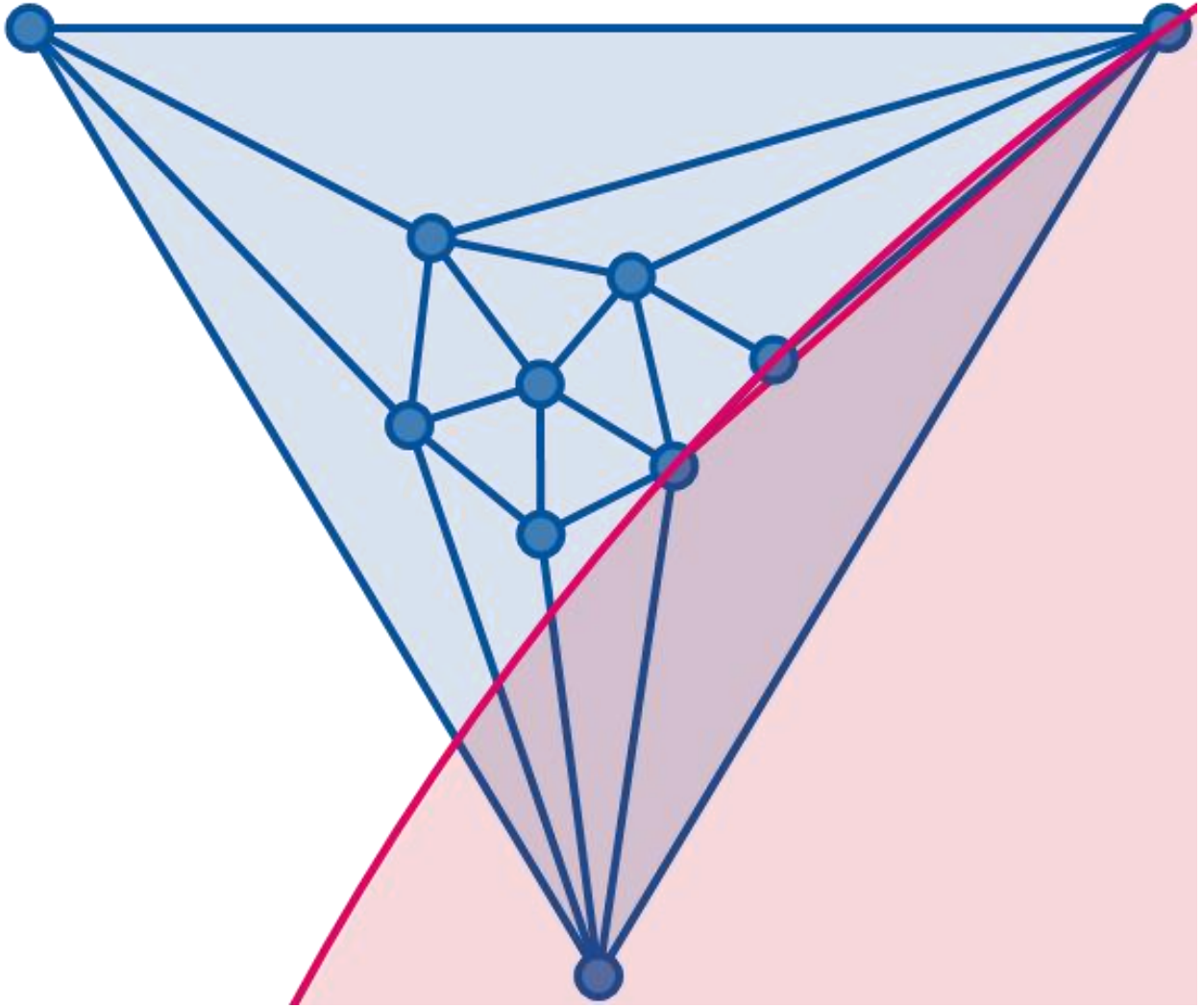
Delaunay Triangulation: Incremental Algorithm



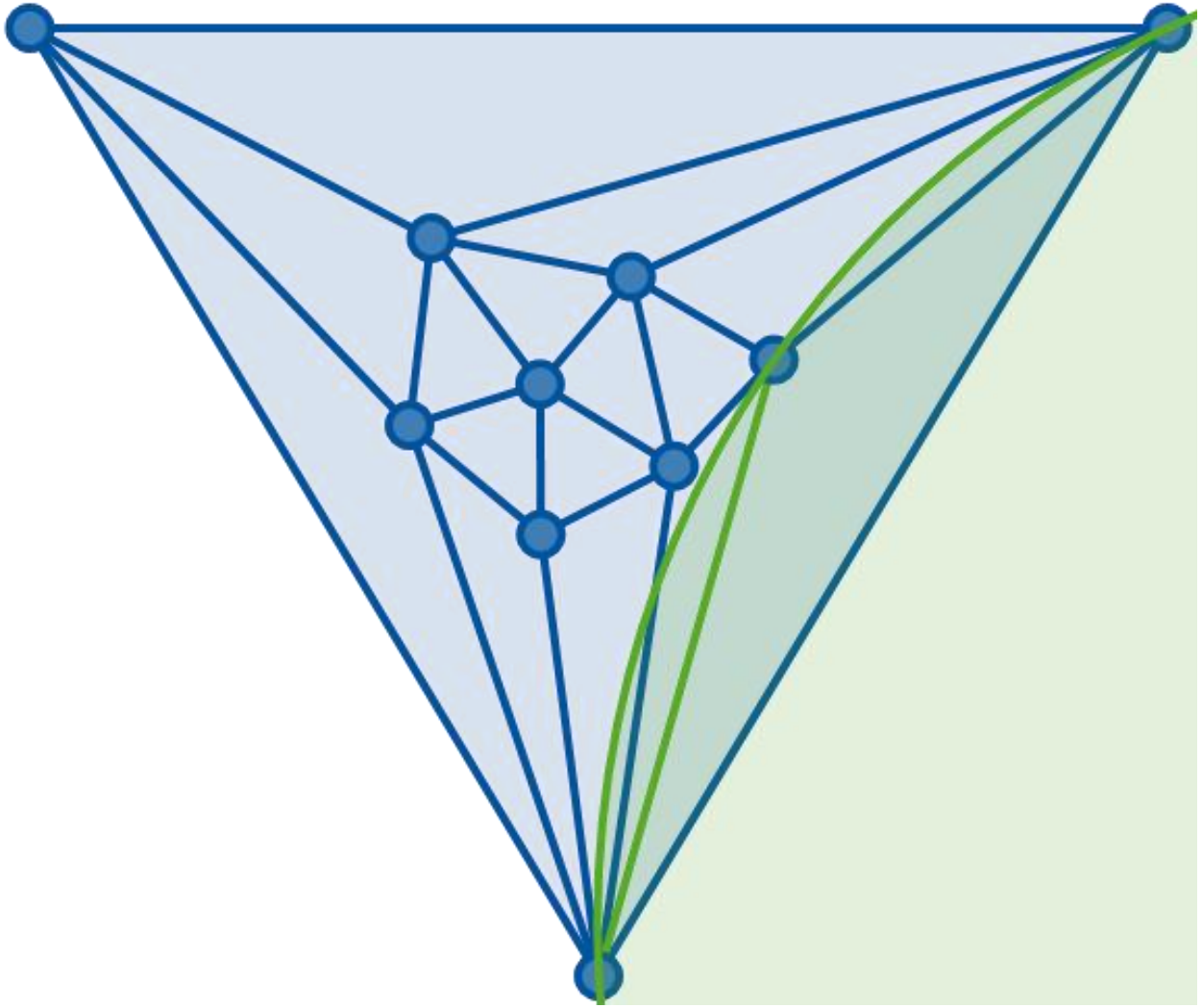
Delaunay Triangulation: Incremental Algorithm



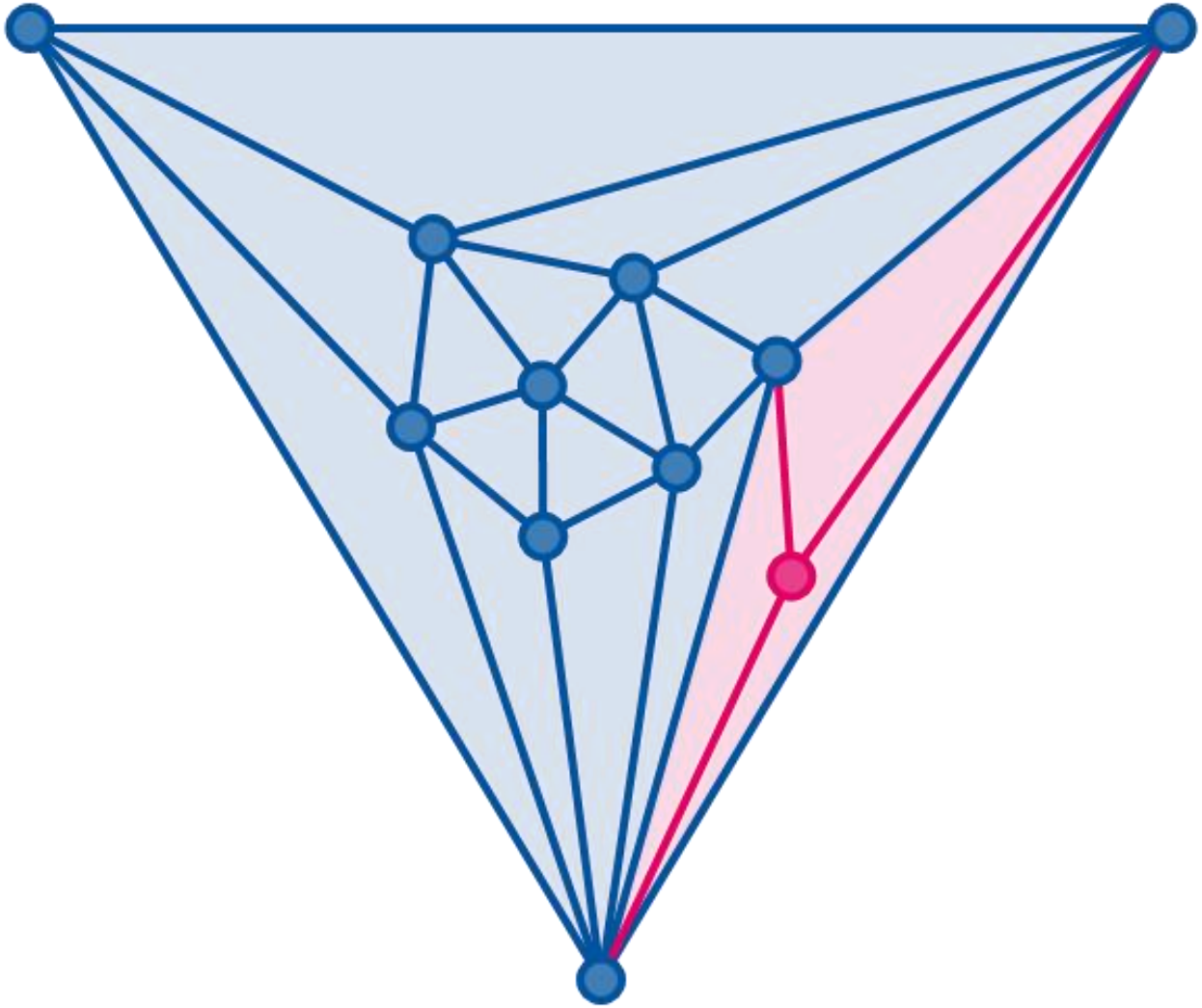
Delaunay Triangulation: Incremental Algorithm



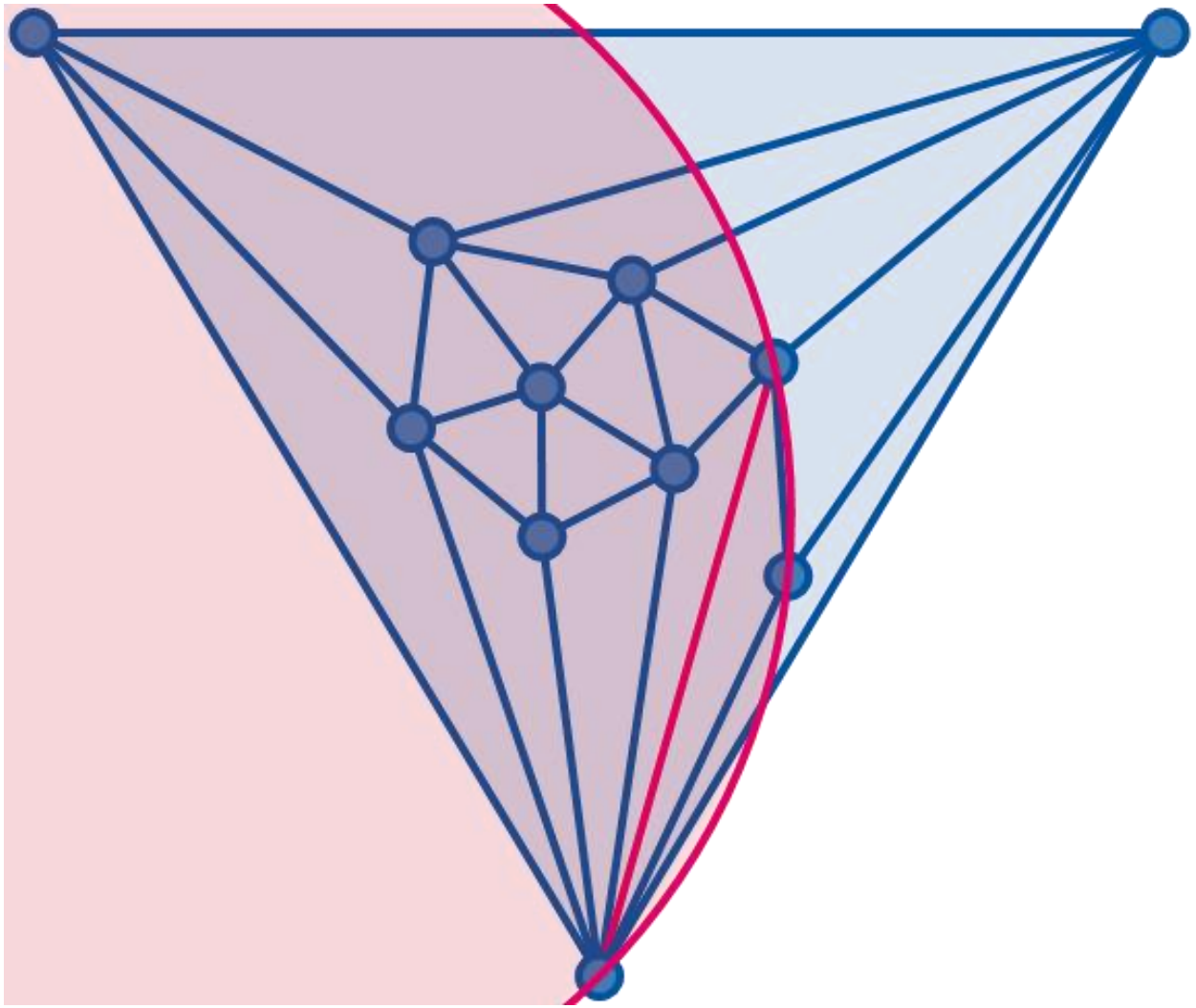
Delaunay Triangulation: Incremental Algorithm



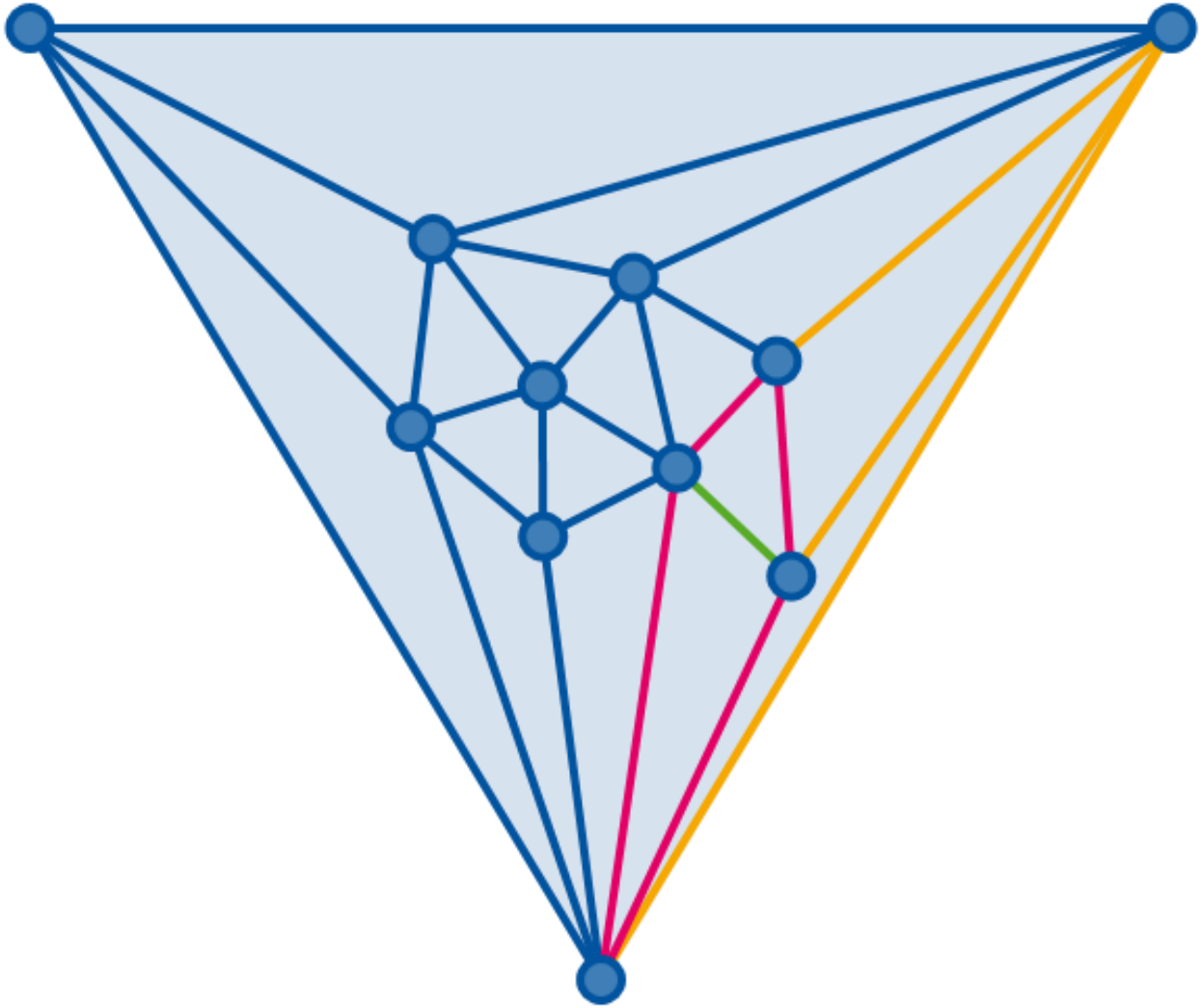
Delaunay Triangulation: Incremental Algorithm



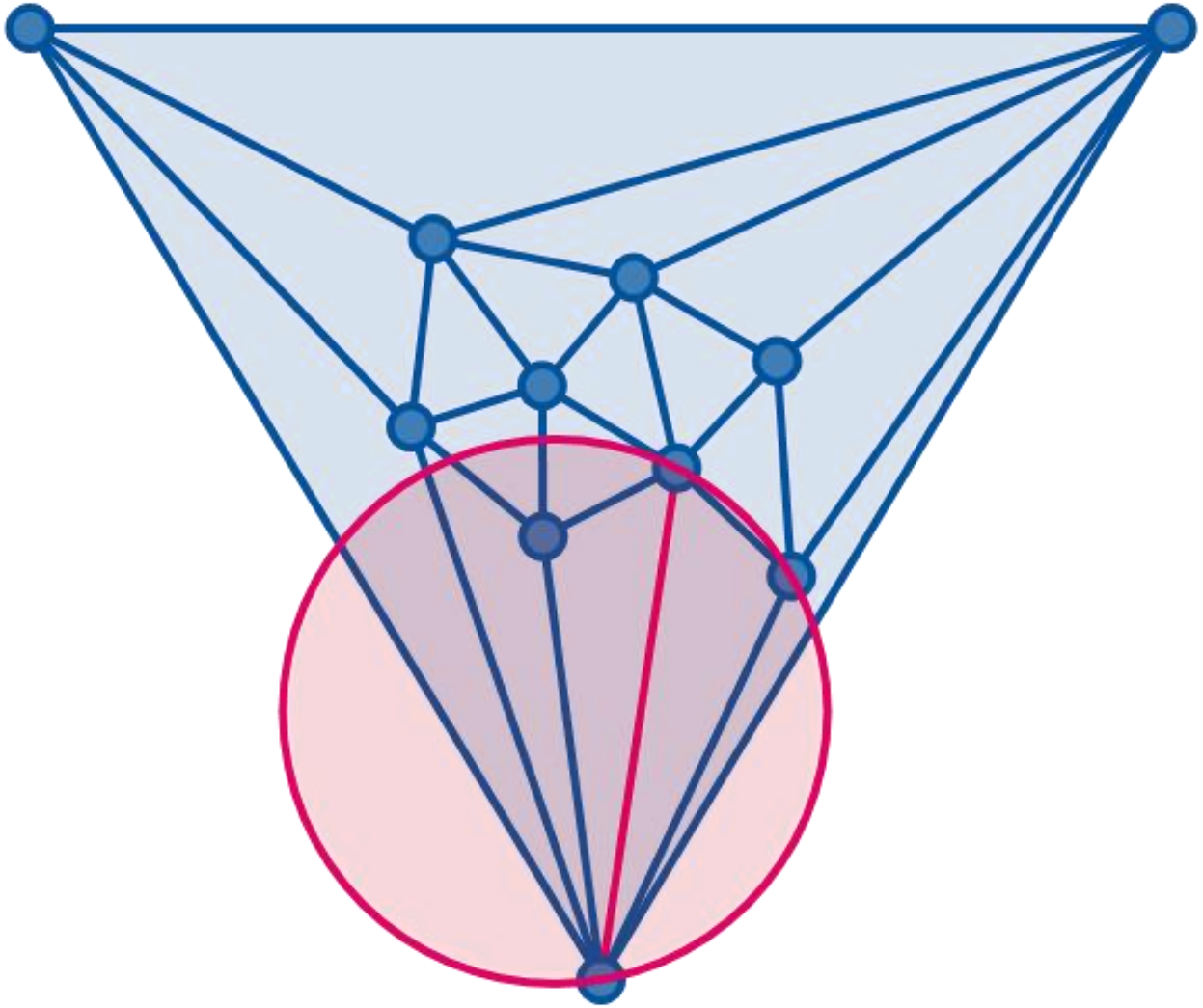
Delaunay Triangulation: Incremental Algorithm



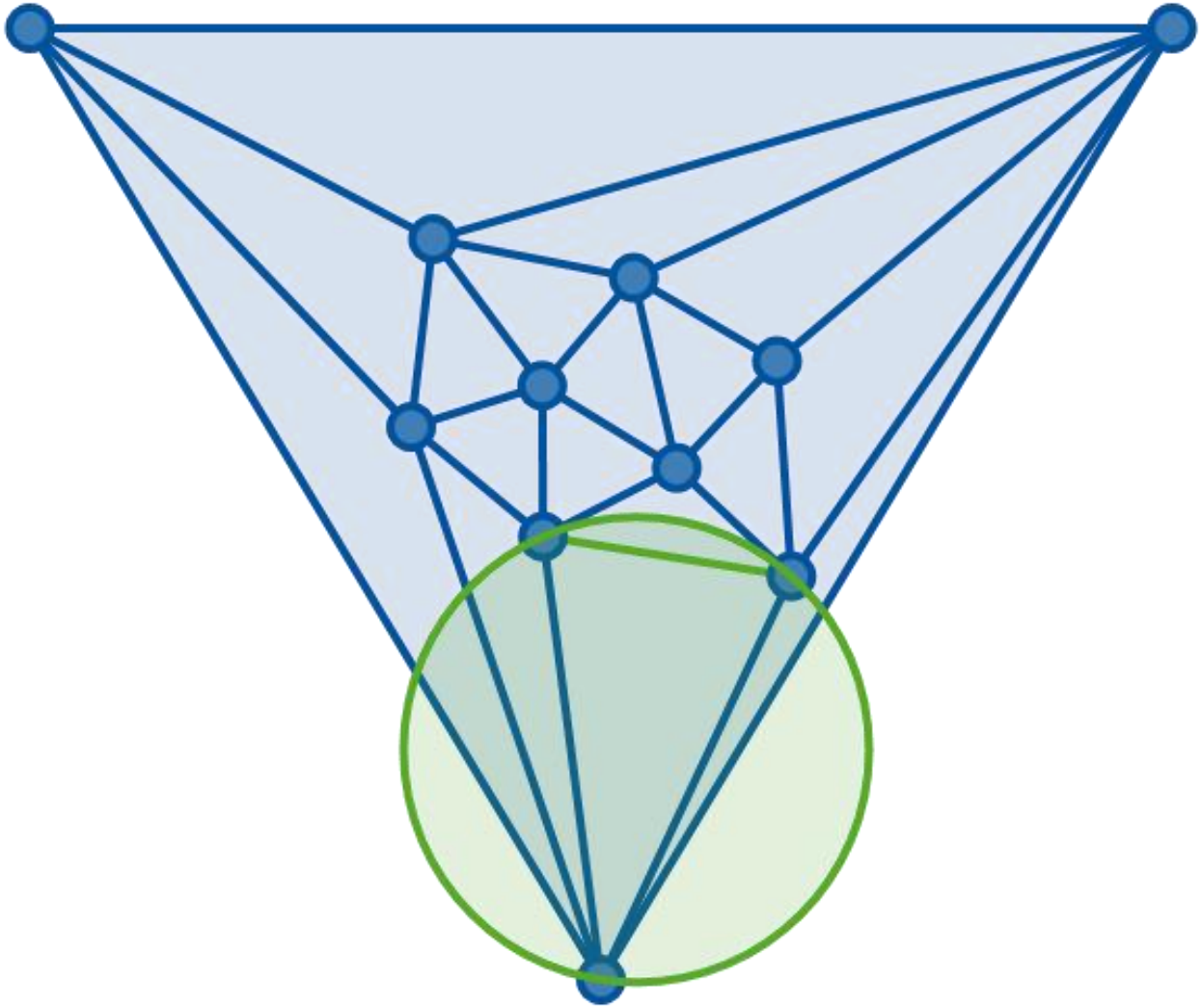
Delaunay Triangulation: Incremental Algorithm



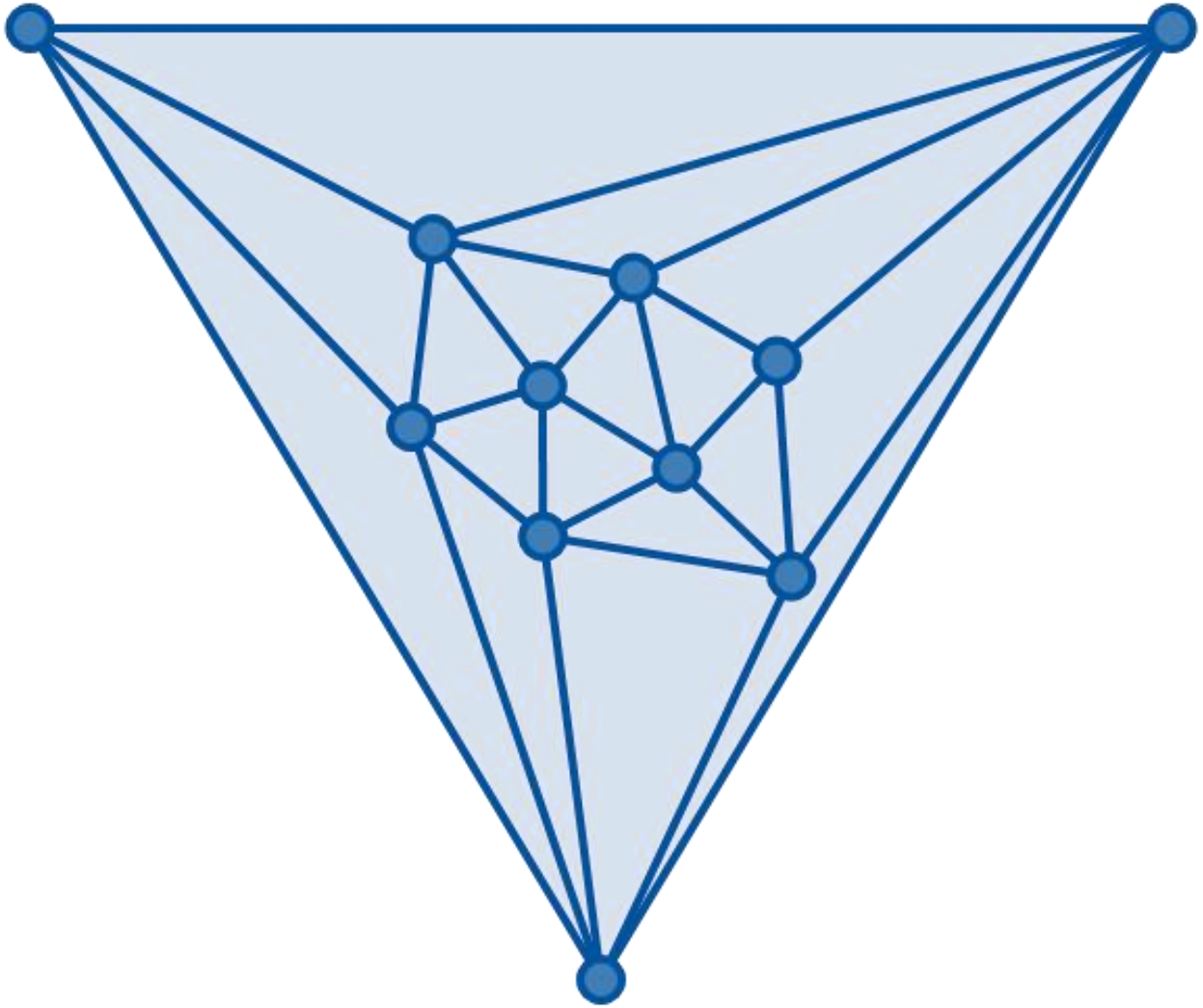
Delaunay Triangulation: Incremental Algorithm



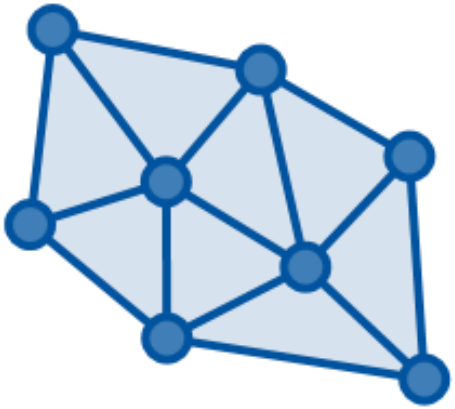
Delaunay Triangulation: Incremental Algorithm



Delaunay Triangulation: Incremental Algorithm



Delaunay Triangulation: Incremental Algorithm



Delaunay Triangulation: Incremental Algorithm

Complexity:

1. For each point $p_i \in p_1, \dots, p_n$ $\boldsymbol{O(n)}$
   a) Find the triangle which contains p_i $\boldsymbol{O(n)}$
   b) Split the triangle (or edge) at position p_i $\boldsymbol{O(1)}$
   c) Restore Delaunay property using edge flips $\boldsymbol{O(k)}$

- Maximal node degree *k* is limited and can be assumed to be constant
- Complexity: $\boldsymbol{O(n^2)}$

Delaunay Triangulation: Incremental Algorithm

Complexity (with better triangle search):

1. For each point $p_i \in p_1, \dots, p_n$ $\boldsymbol{O(n)}$
   a) Find the triangle which contains p_i $\boldsymbol{O(n)} \rightarrow \boldsymbol{O(\log n)}$
   b) Split the triangle (or edge) at position p_i $\boldsymbol{O(1)}$
   c) Restore Delaunay property using edge flips $\boldsymbol{O(k)}$

- Maximal node degree *k* is limited and can be assumed to be constant
- Complexity: $\boldsymbol{O(n \log n)}$

Vornoi Diagram from Delaunay Triangulation

Input: Delaunay Triangulation

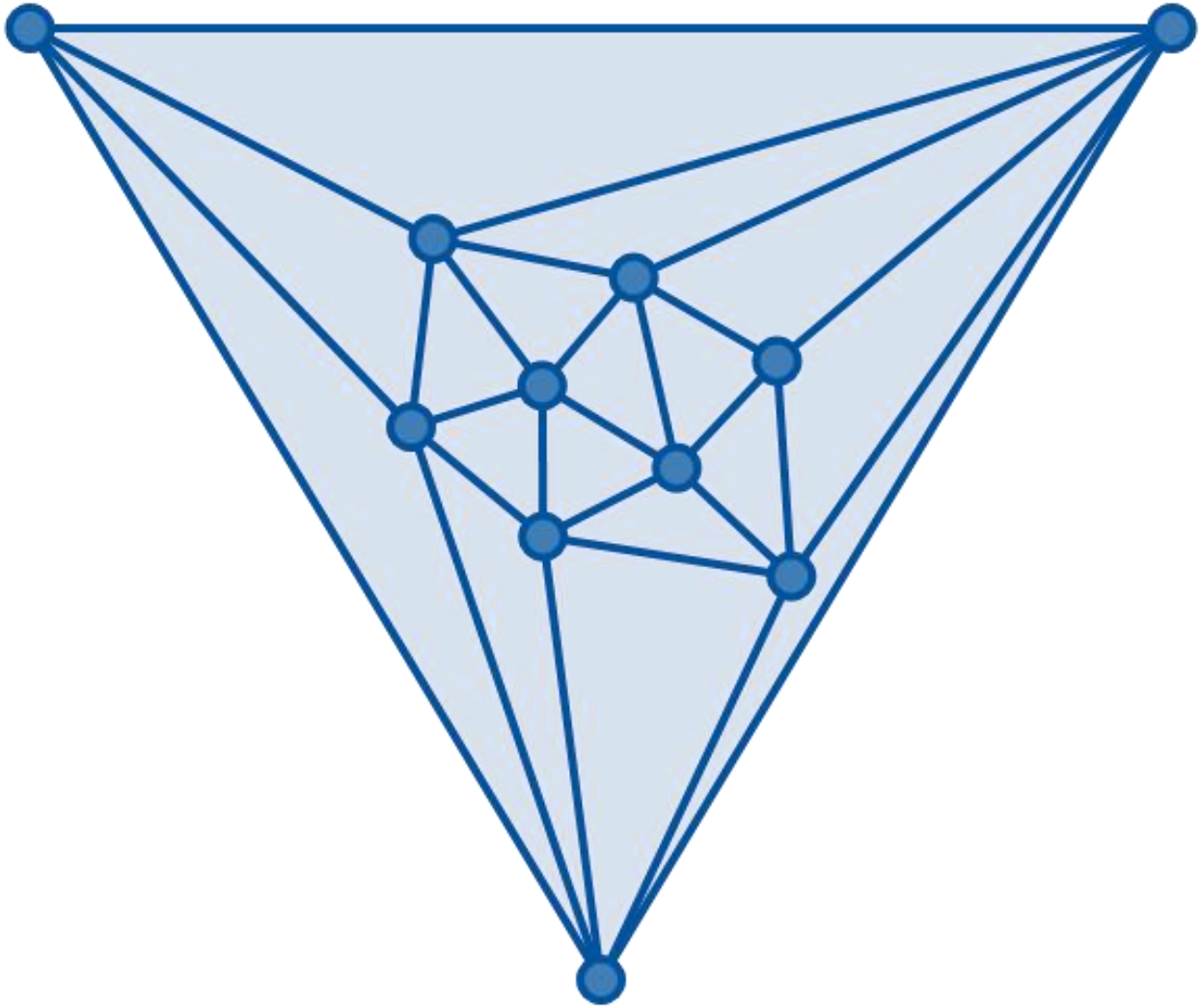
Output: Voronoi Diagram

Algorithm:

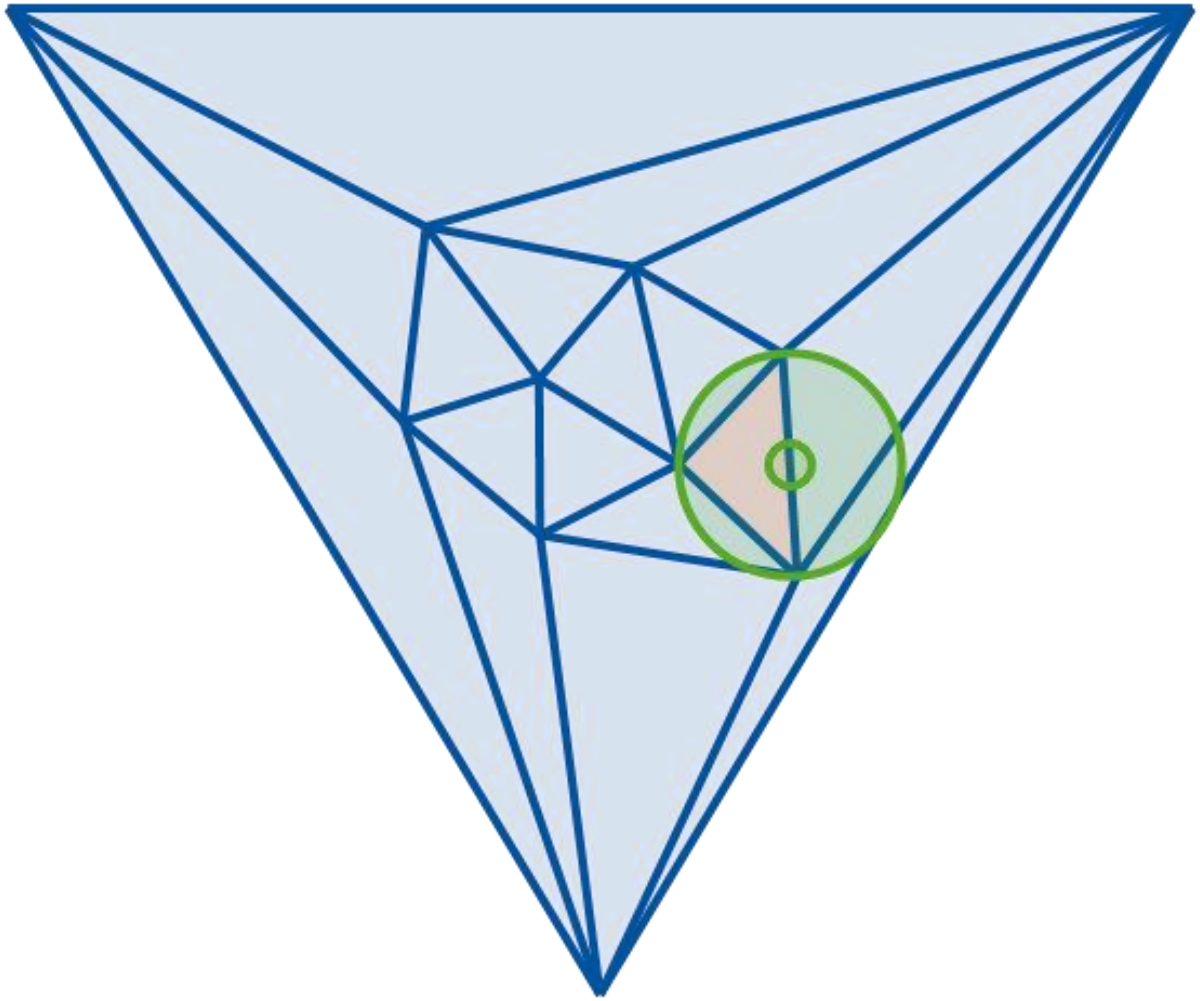
1. For each **face** in the Delaunay Triangulation
   a) Calculate **Circumcircle**
   b) Generate **vertex** in the Vornoi Diagram at the center of the circle

2. For each **vertex** in the Delaunay Triangulation
   a) Create a **face** in the Vornoi diagram
   b) Use the Voronoi vertices which belong to adjacent Delaunay faces

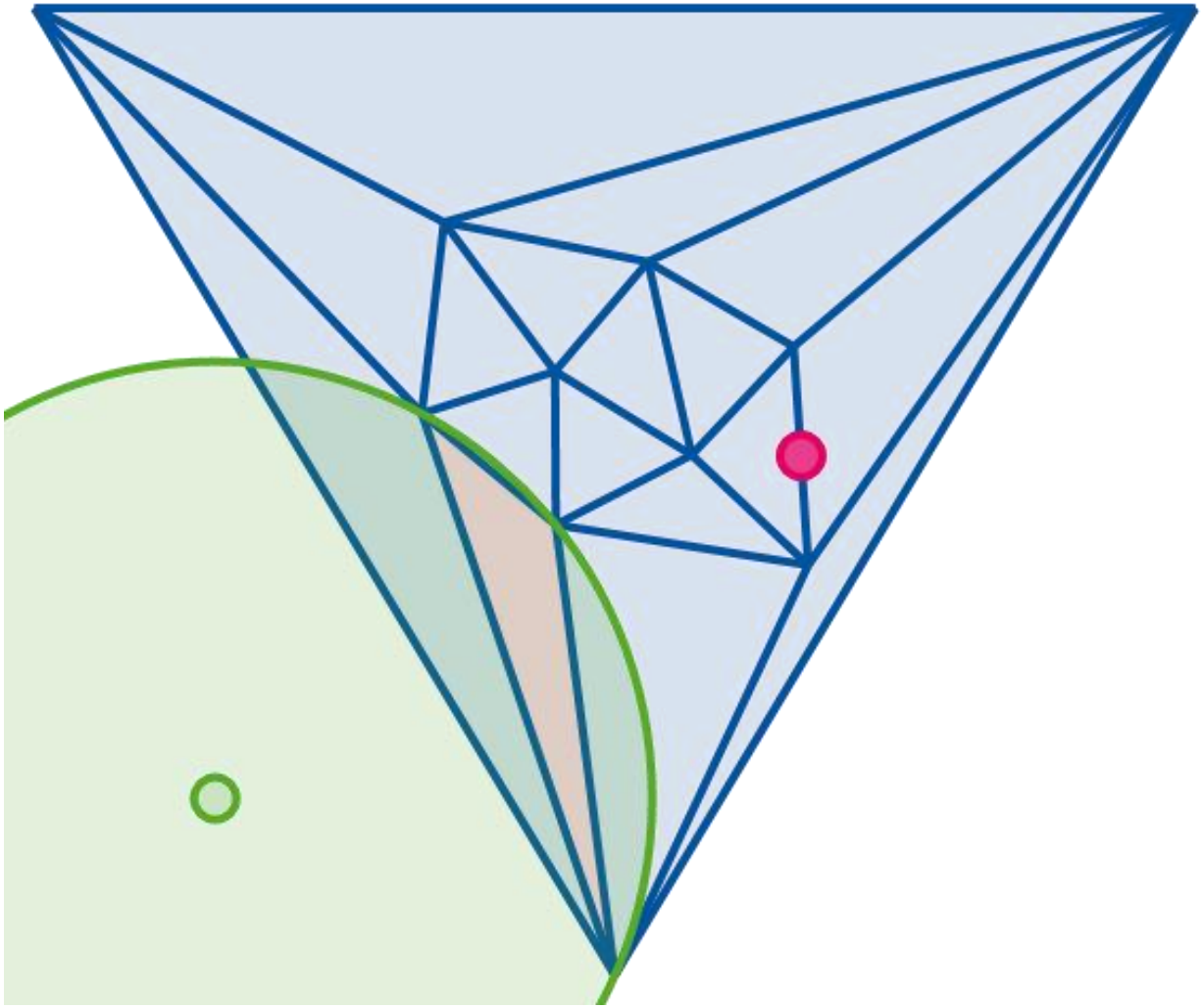
Vornoi Diagram from Delaunay Triangulation



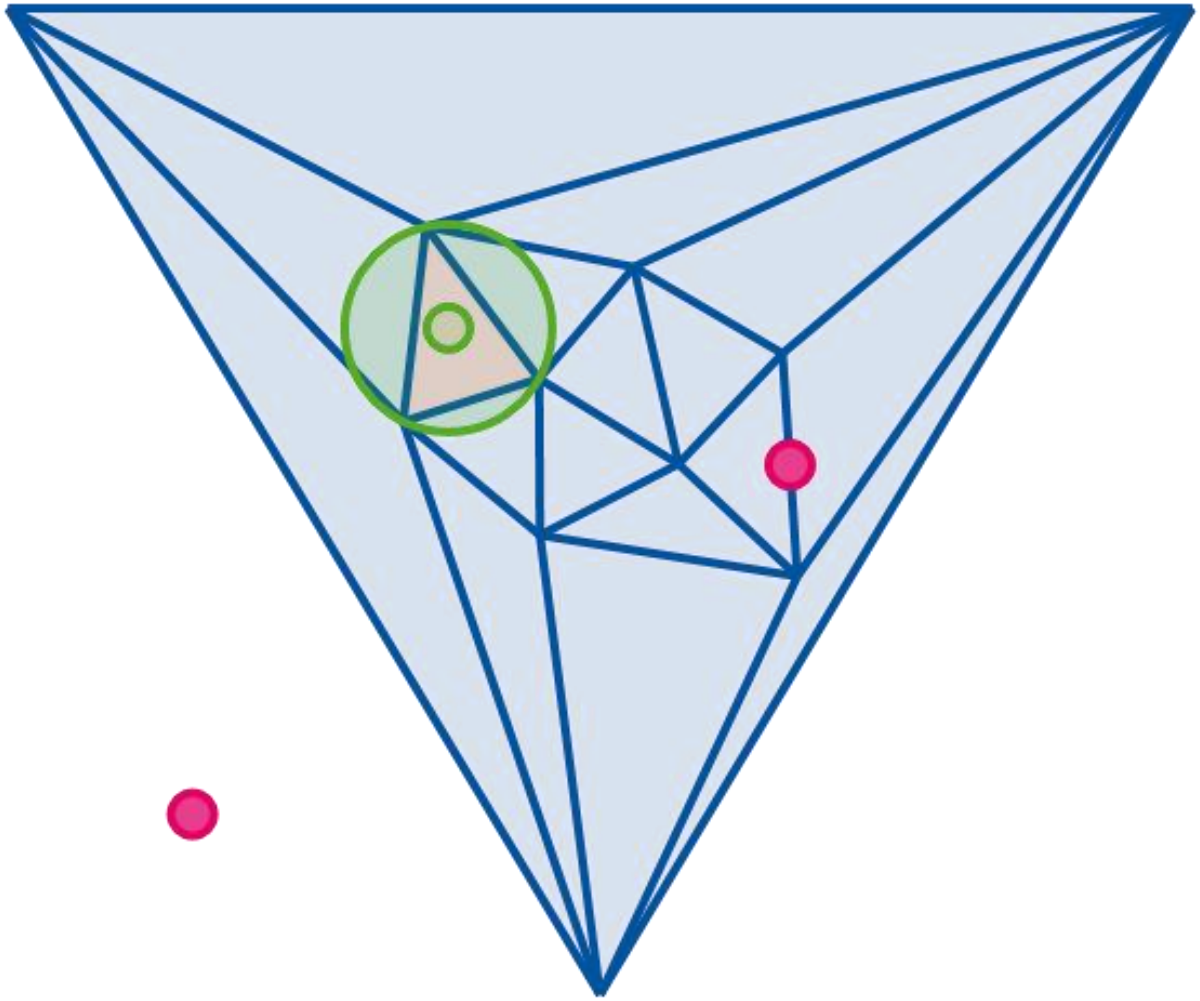
Vornoi Diagram from Delaunay Triangulation



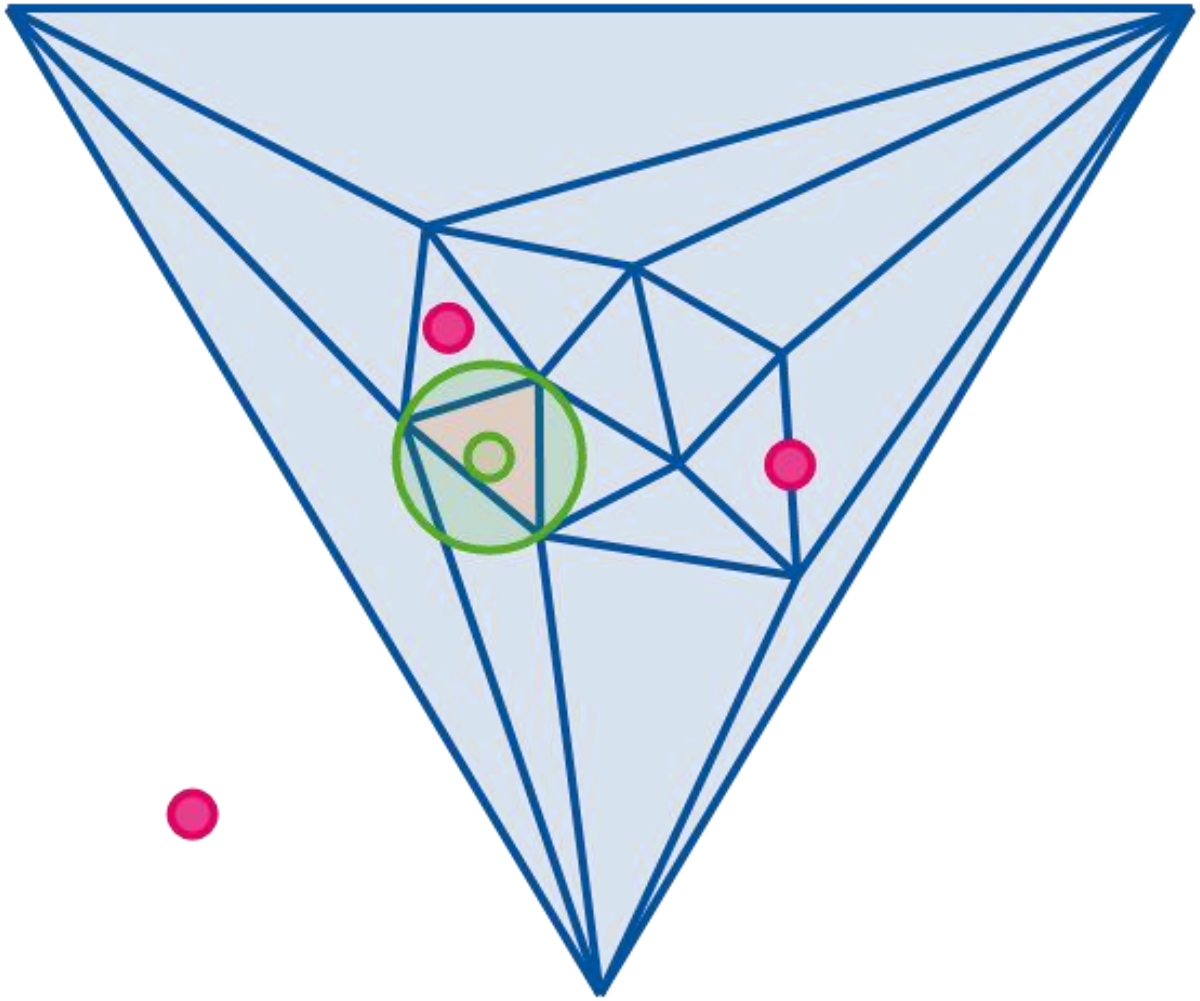
Vornoi Diagram from Delaunay Triangulation



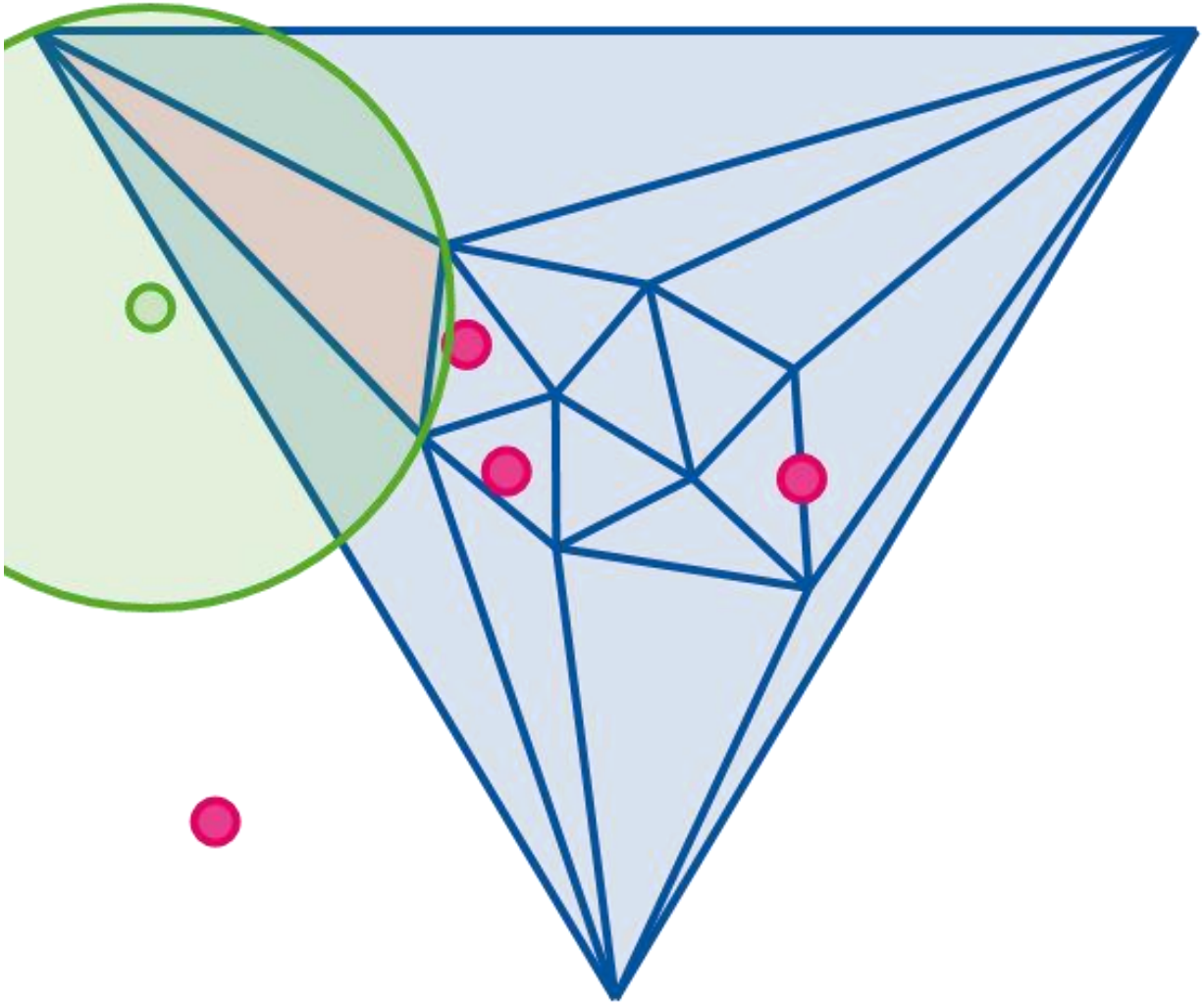
Vornoi Diagram from Delaunay Triangulation



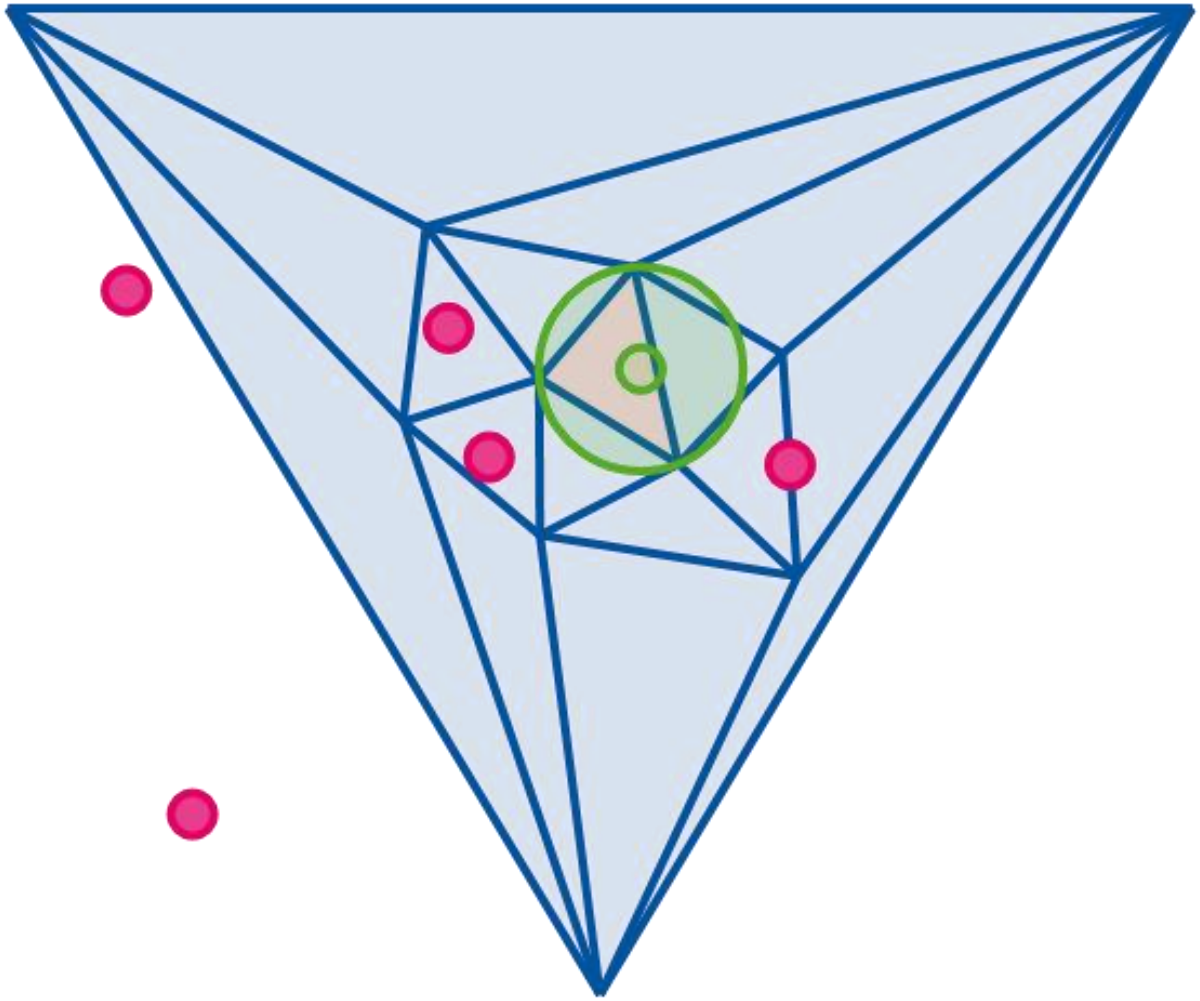
Vornoi Diagram from Delaunay Triangulation



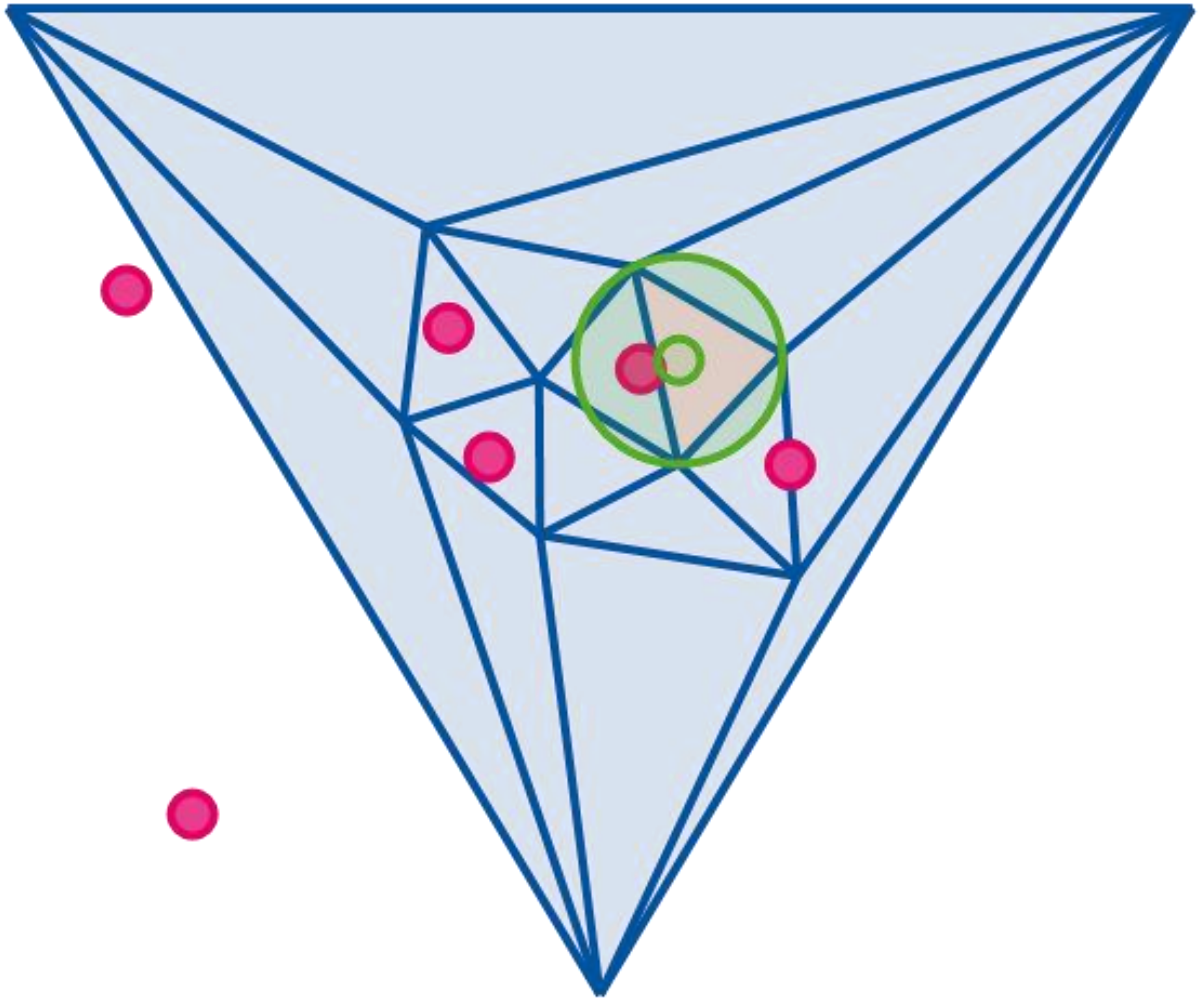
Vornoi Diagram from Delaunay Triangulation



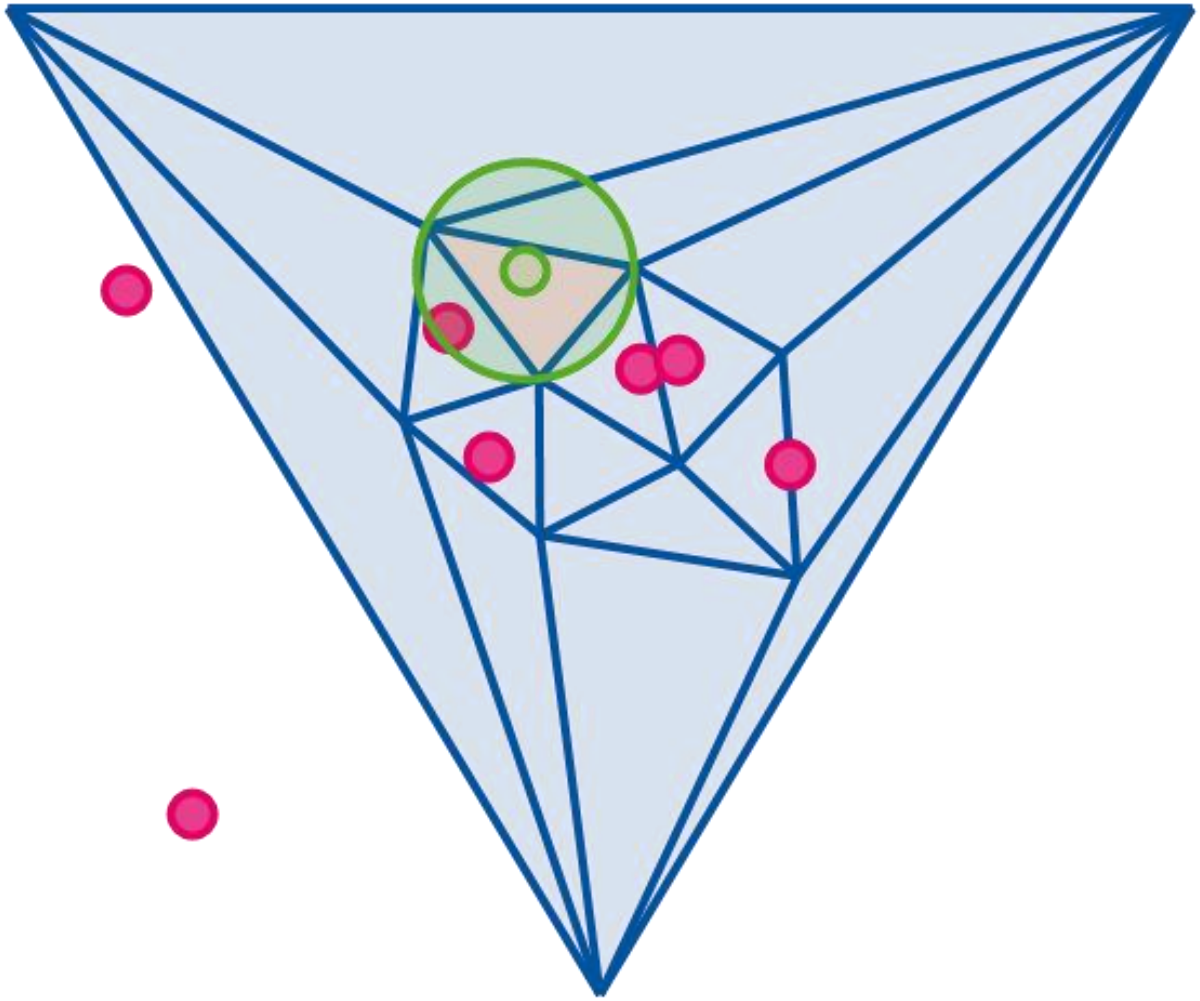
Vornoi Diagram from Delaunay Triangulation



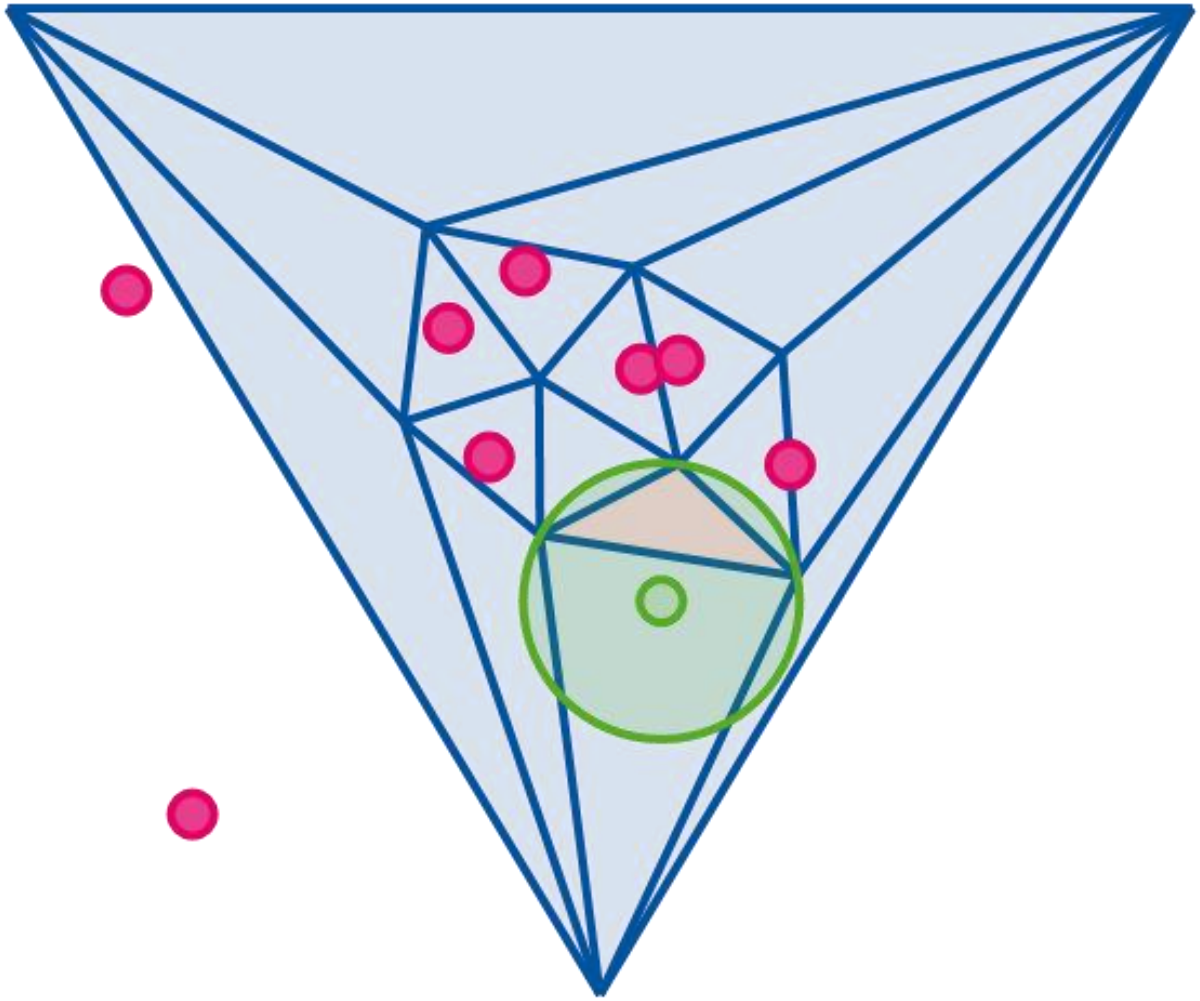
Vornoi Diagram from Delaunay Triangulation



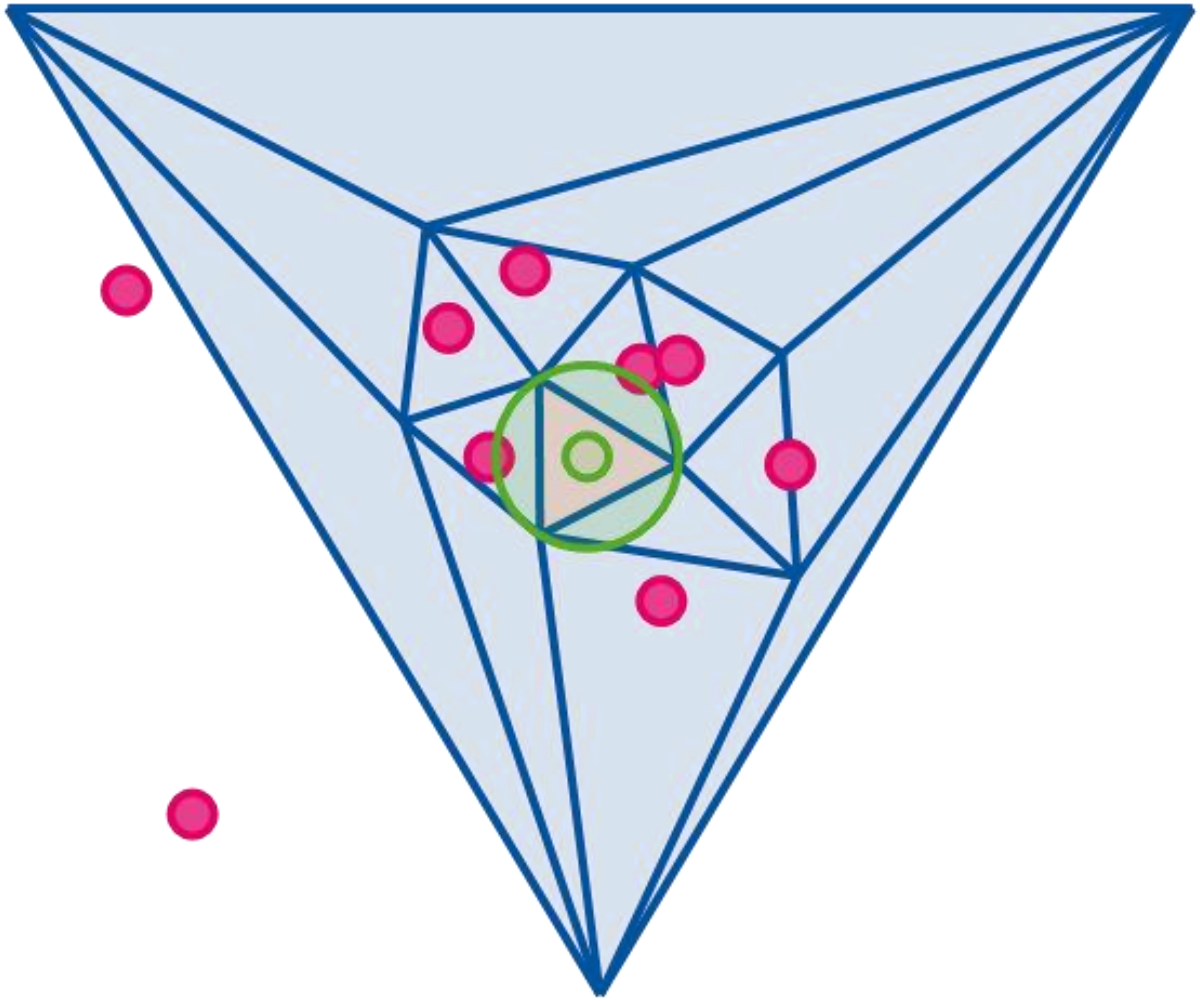
Vornoi Diagram from Delaunay Triangulation



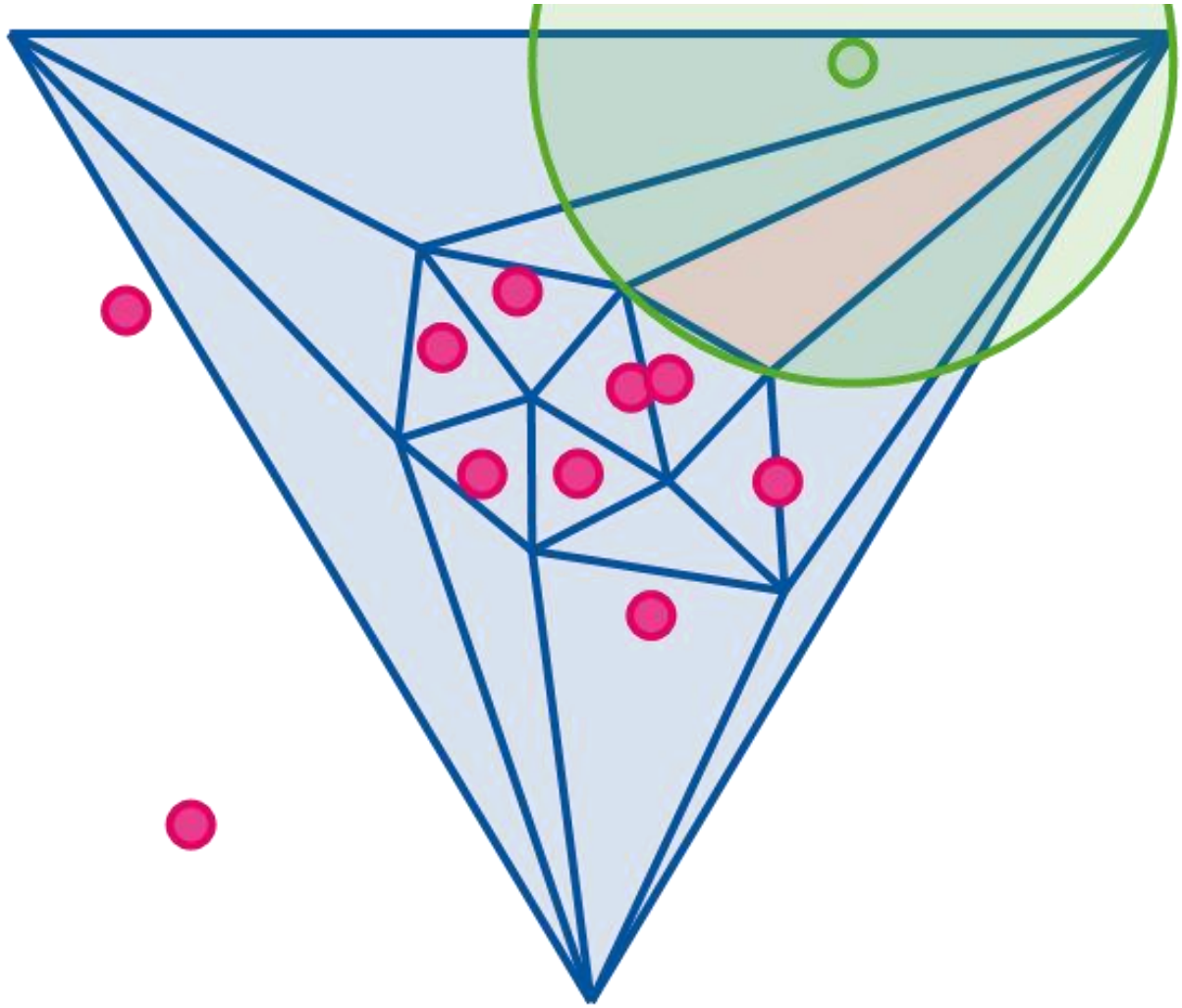
Vornoi Diagram from Delaunay Triangulation



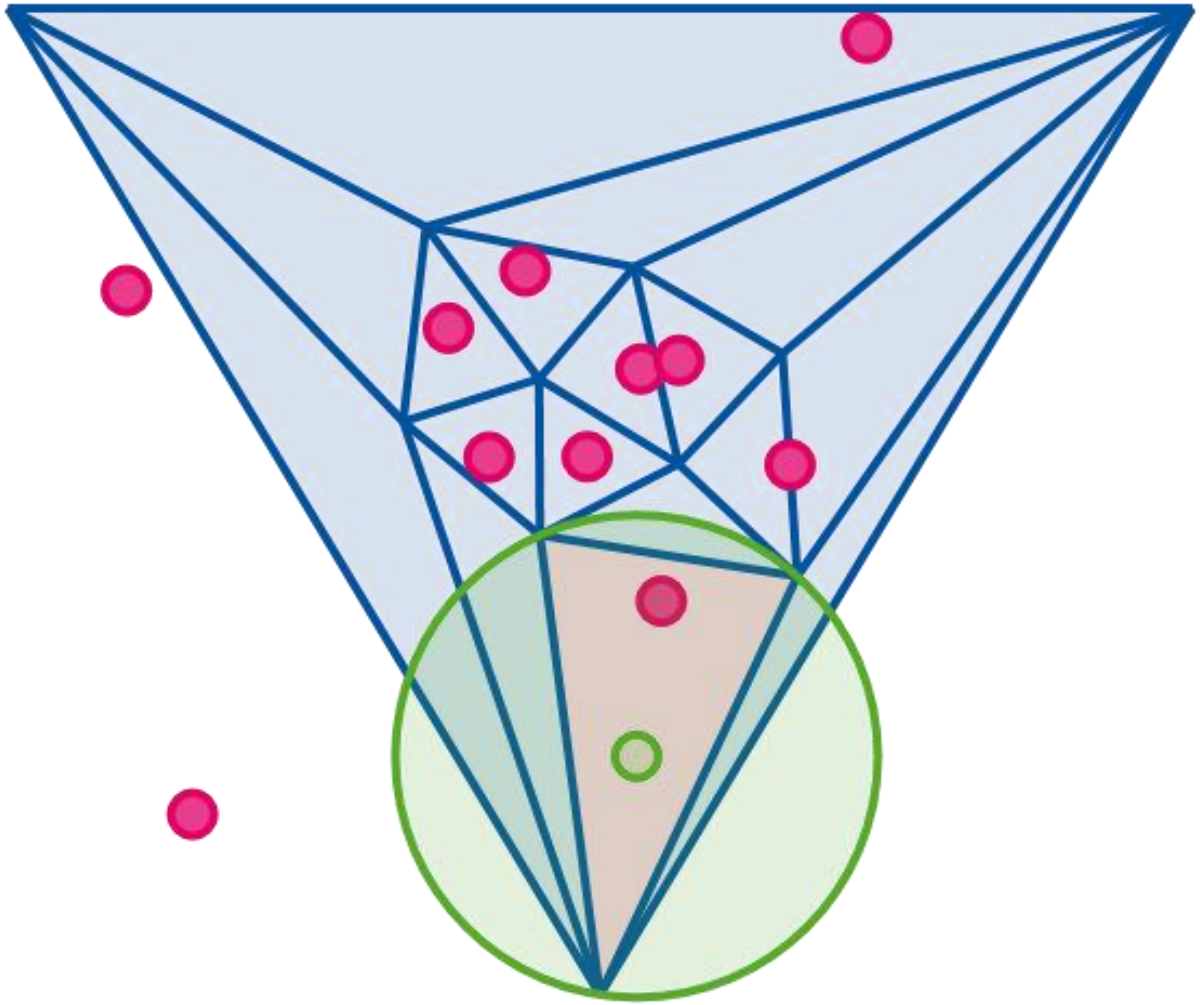
Vornoi Diagram from Delaunay Triangulation



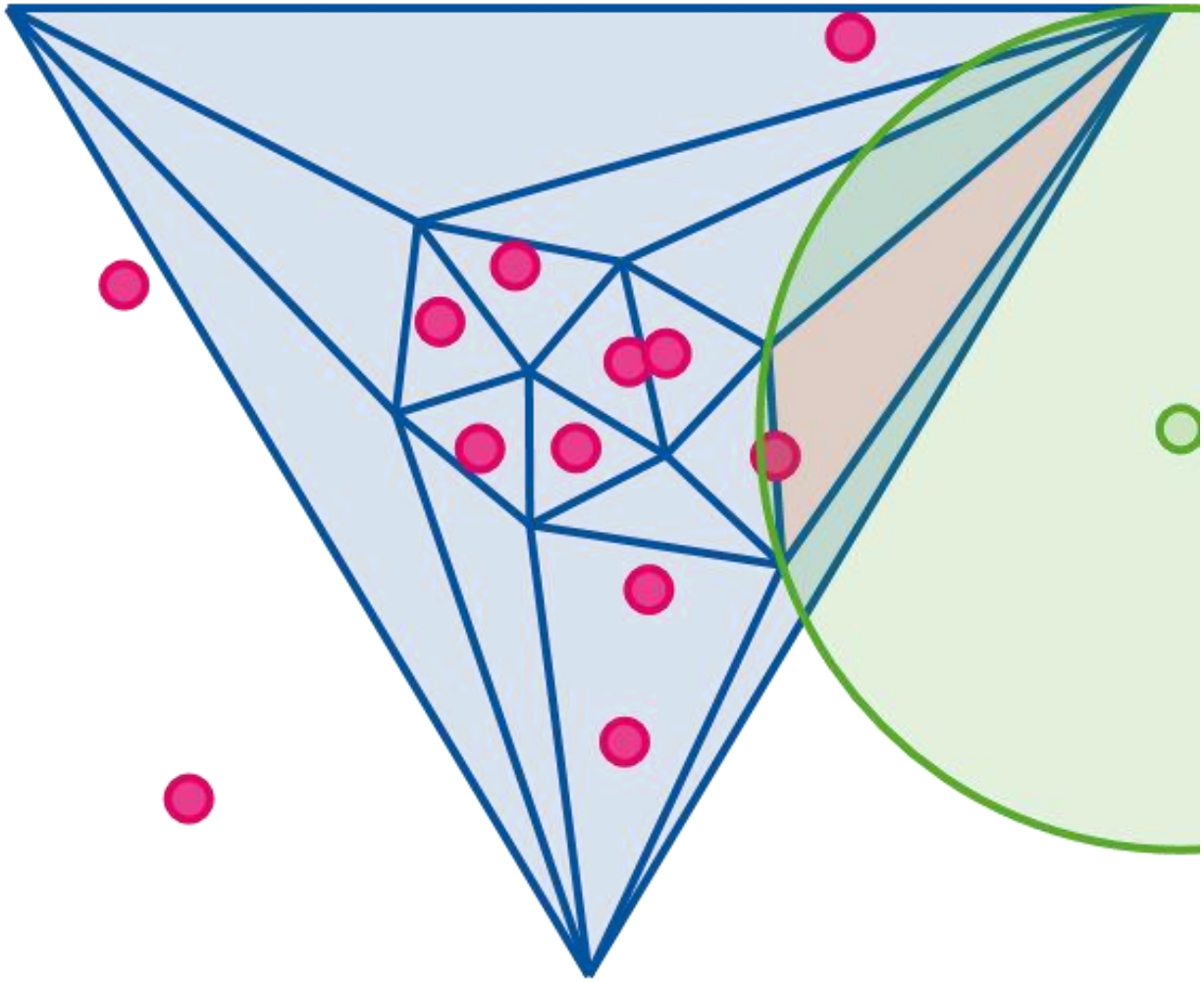
Vornoi Diagram from Delaunay Triangulation



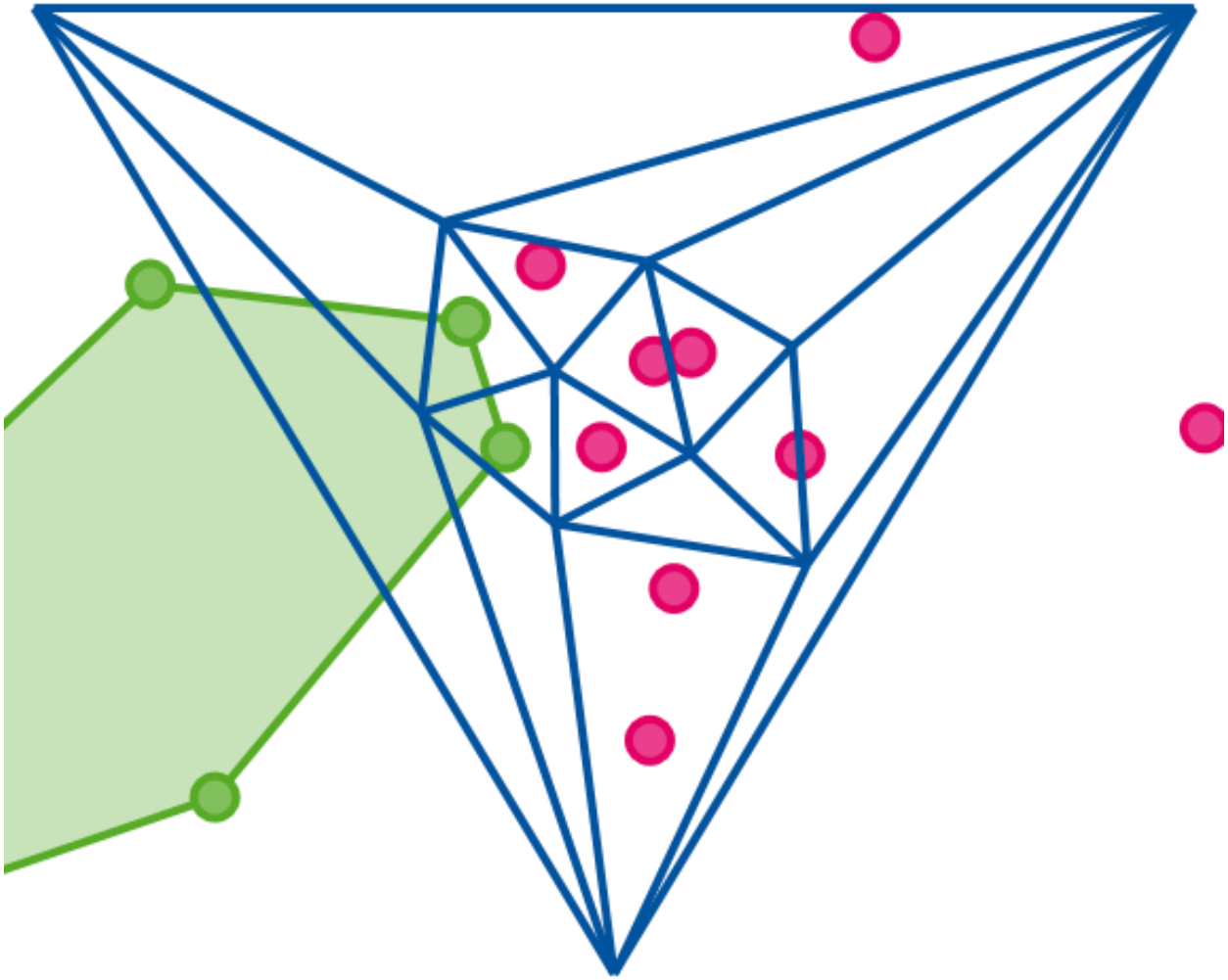
Vornoi Diagram from Delaunay Triangulation



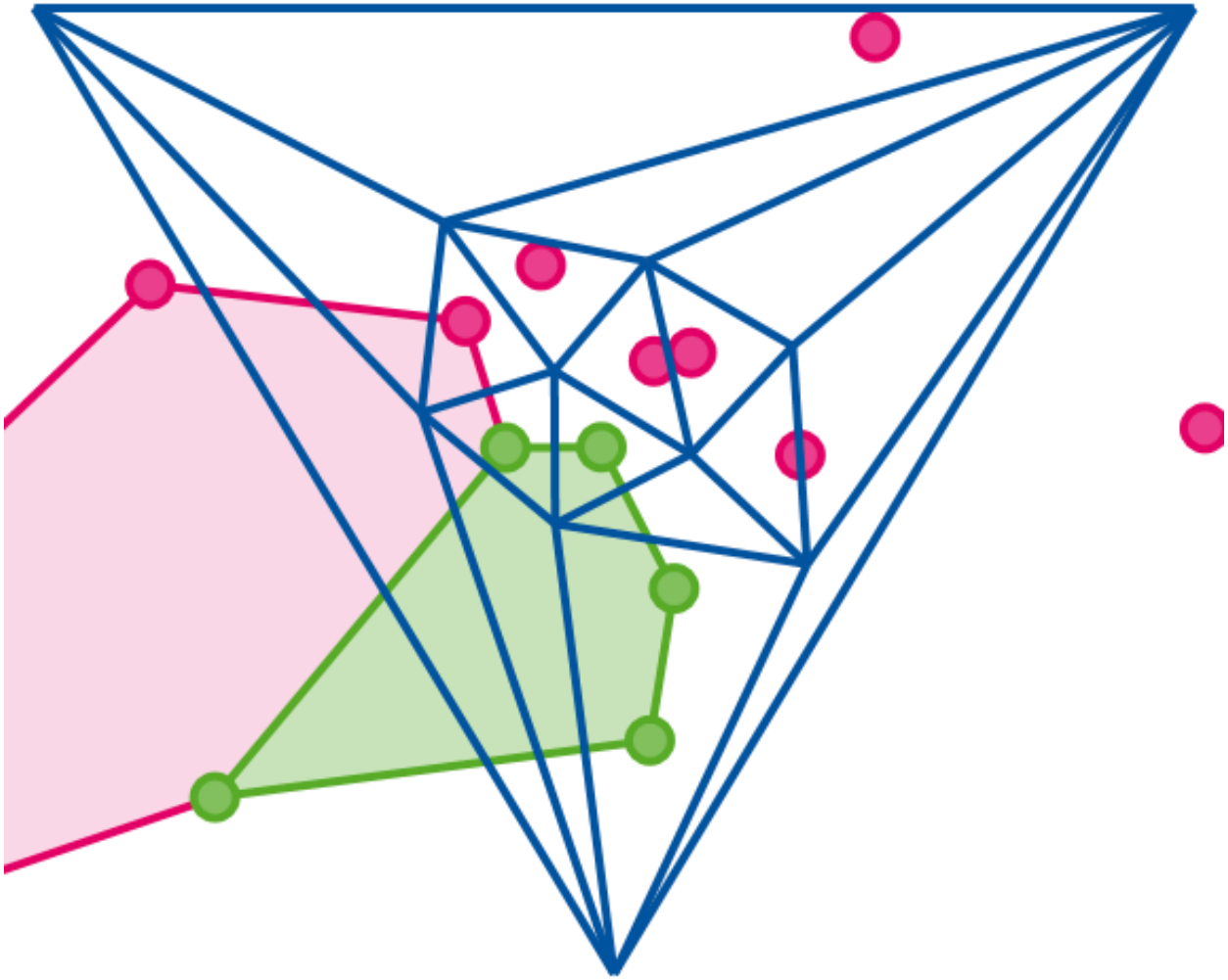
Vornoi Diagram from Delaunay Triangulation



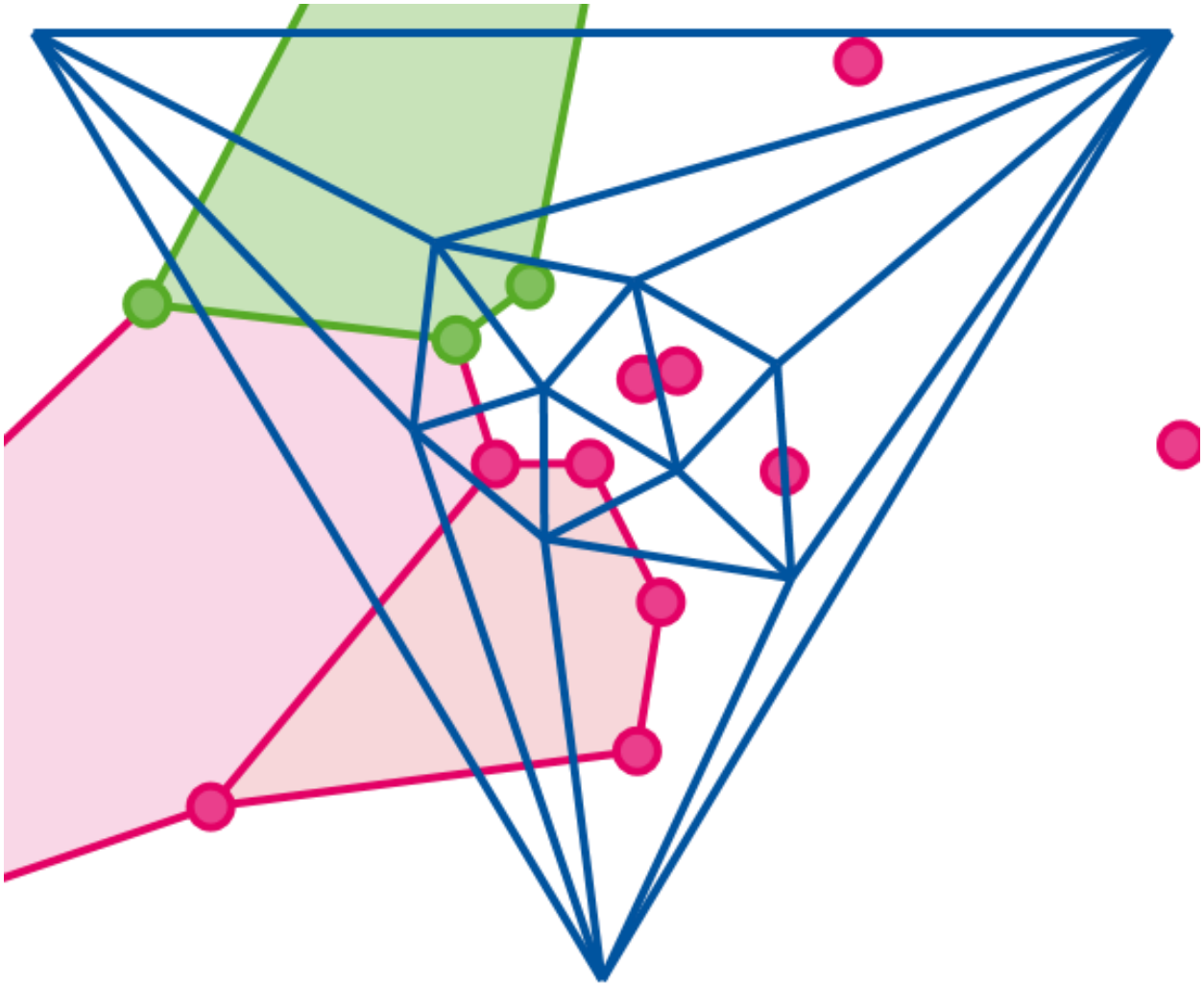
Vornoi Diagram from Delaunay Triangulation



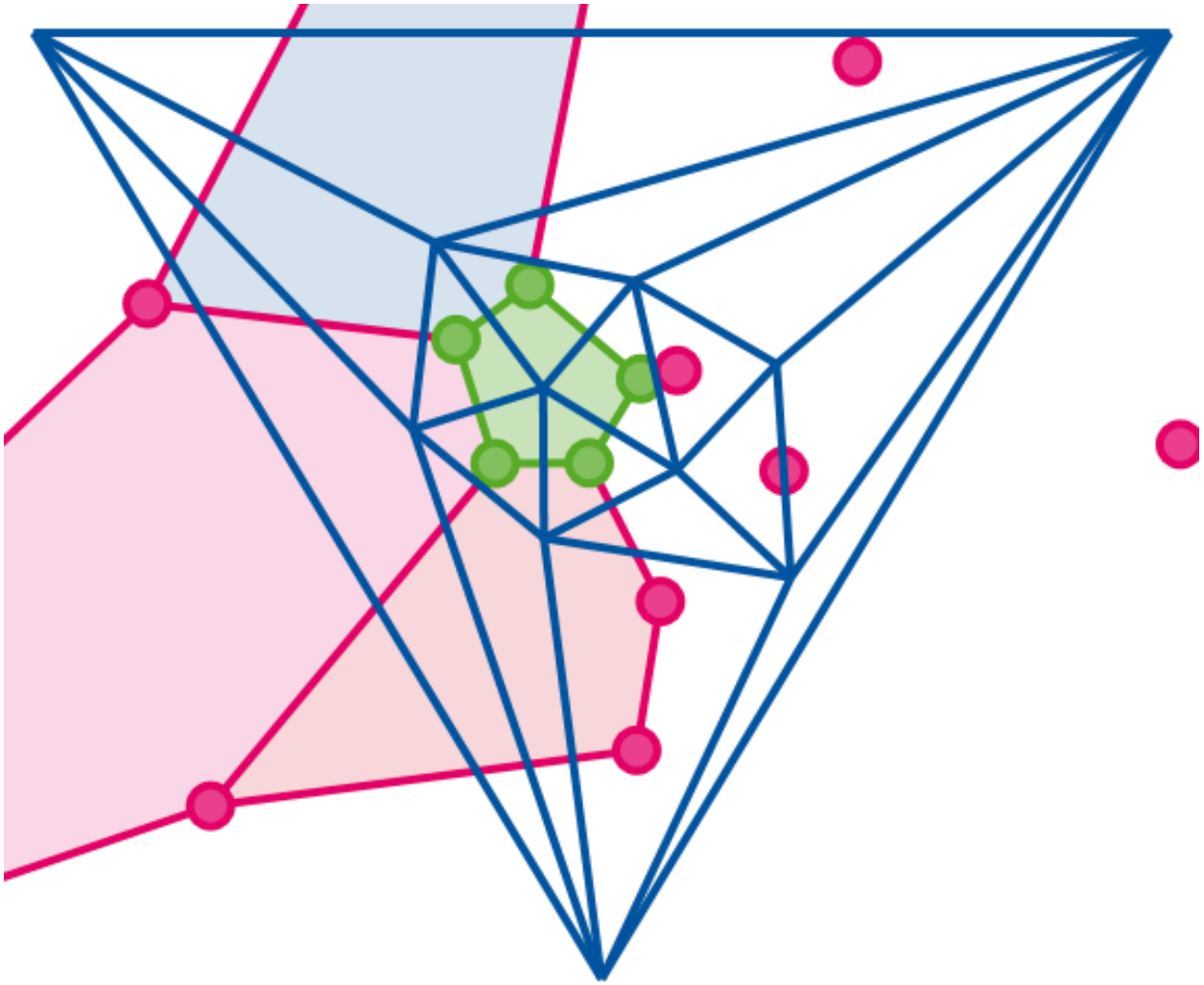
Vornoi Diagram from Delaunay Triangulation



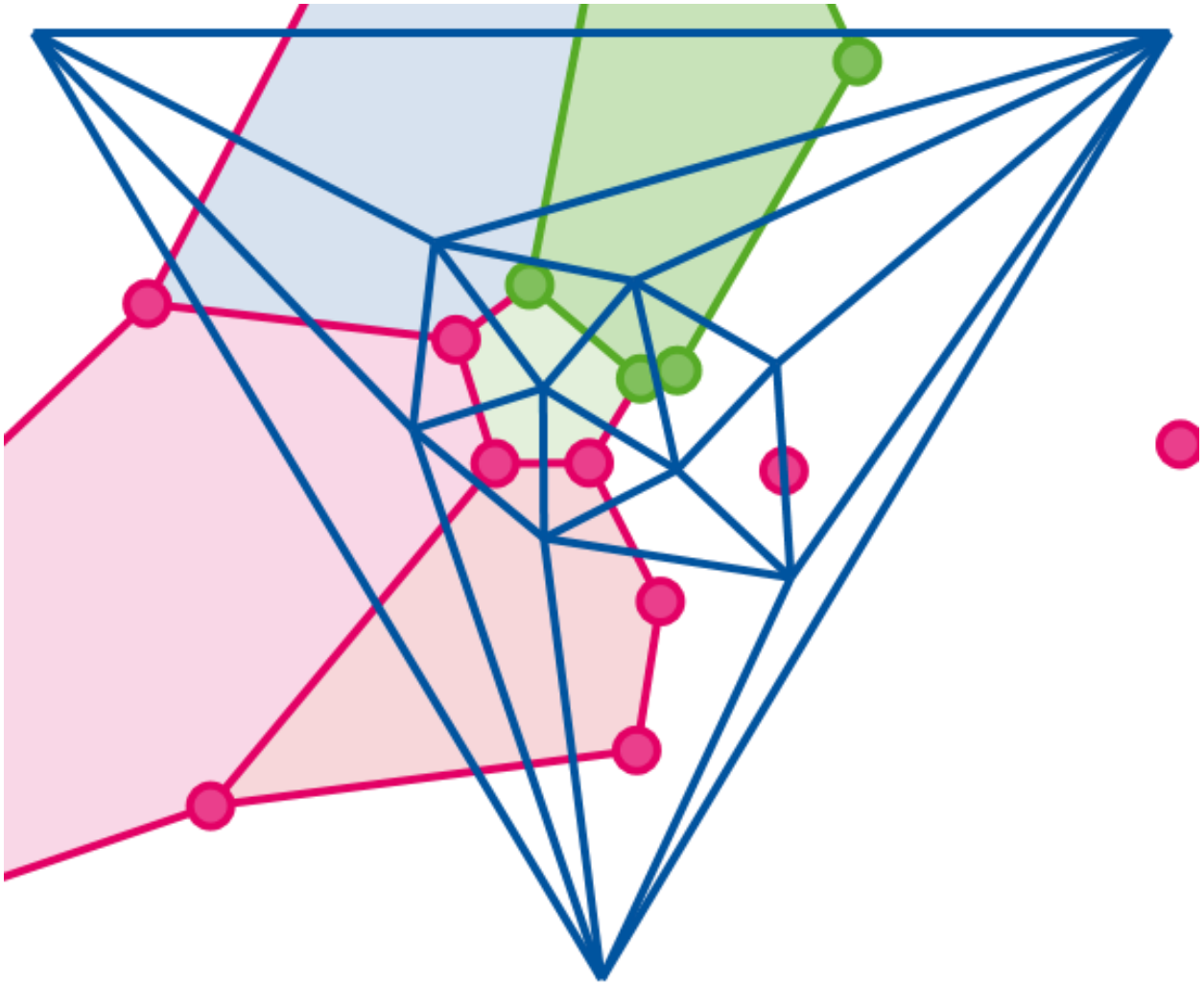
Vornoi Diagram from Delaunay Triangulation



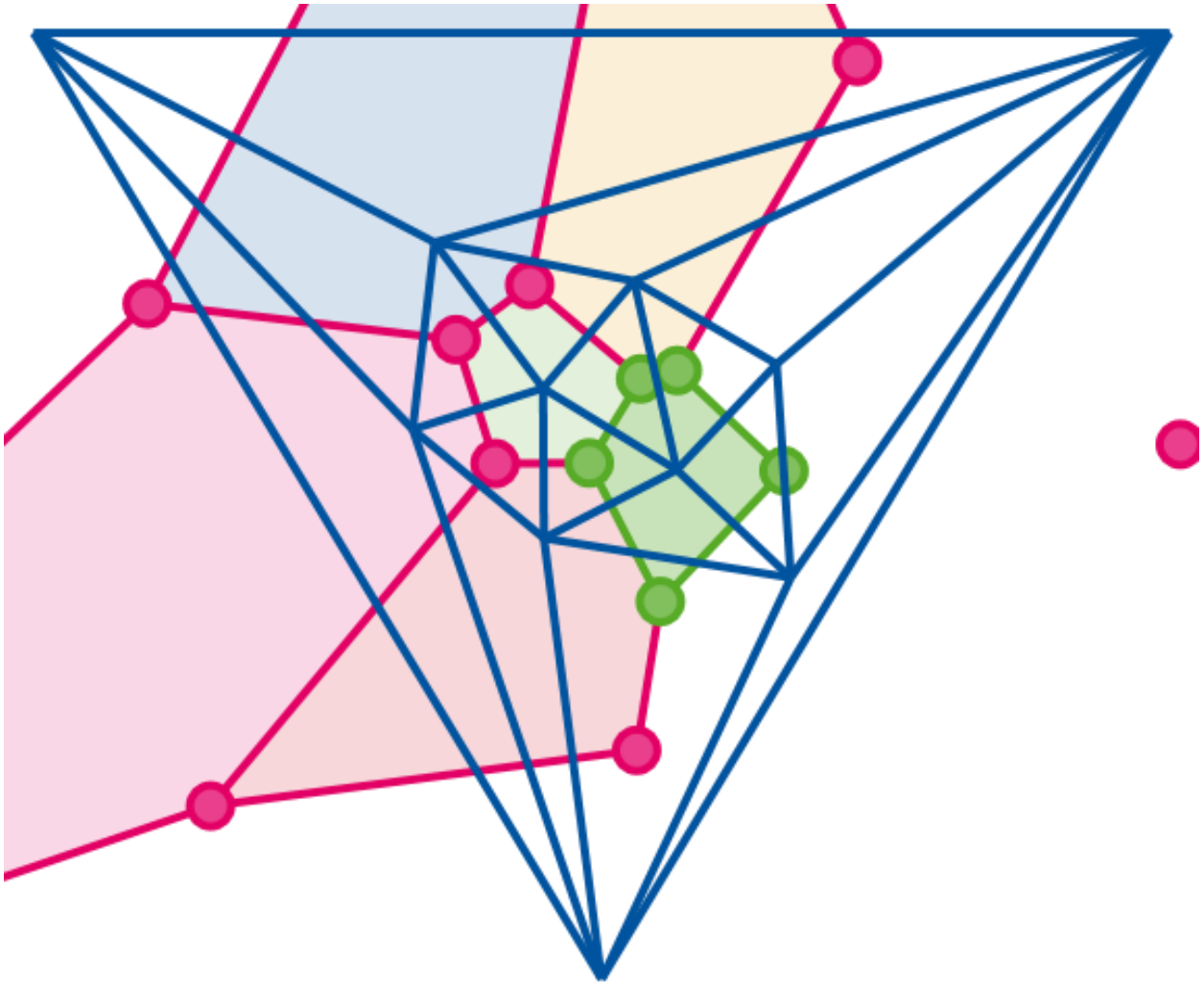
Vornoi Diagram from Delaunay Triangulation



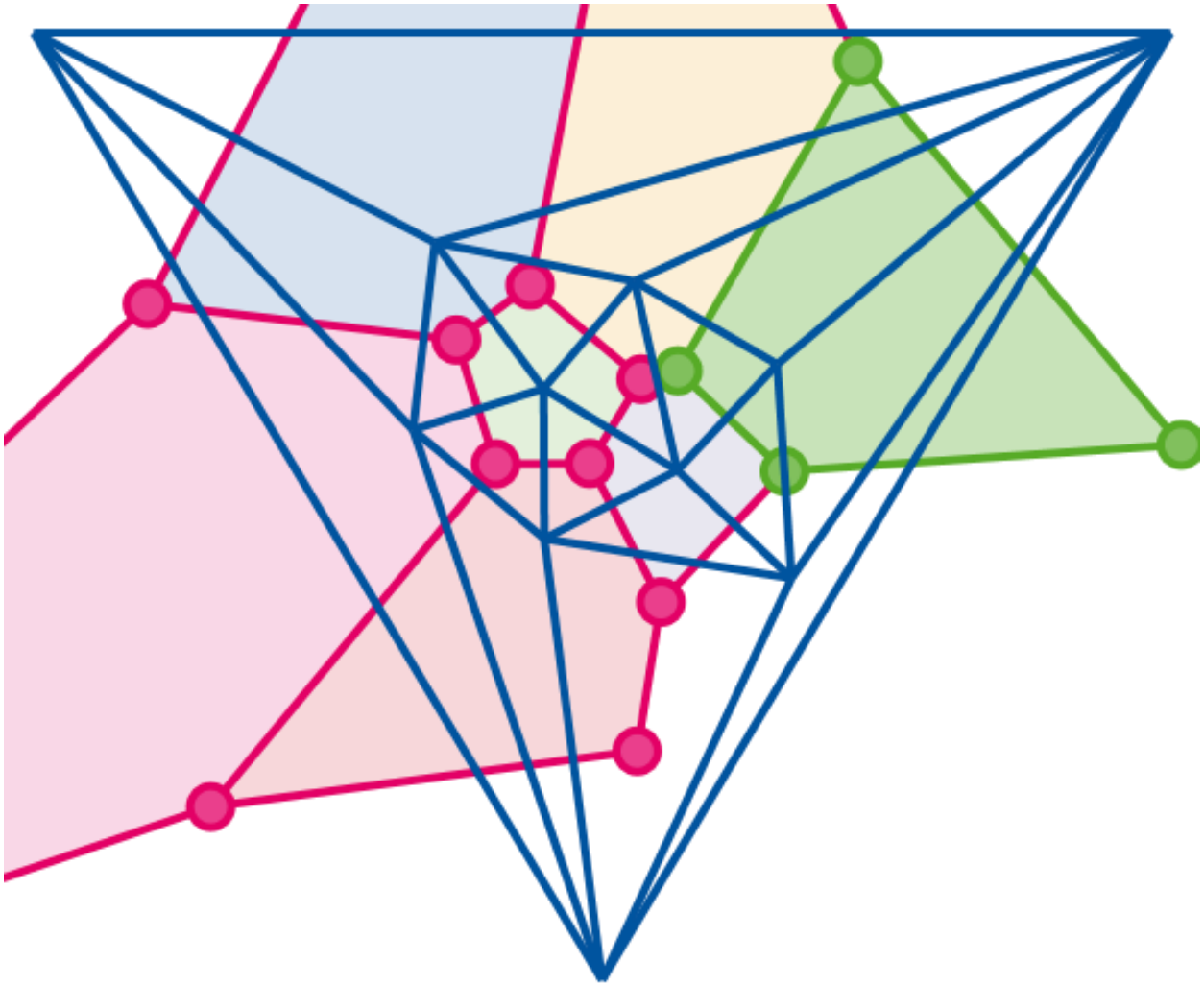
Vornoi Diagram from Delaunay Triangulation



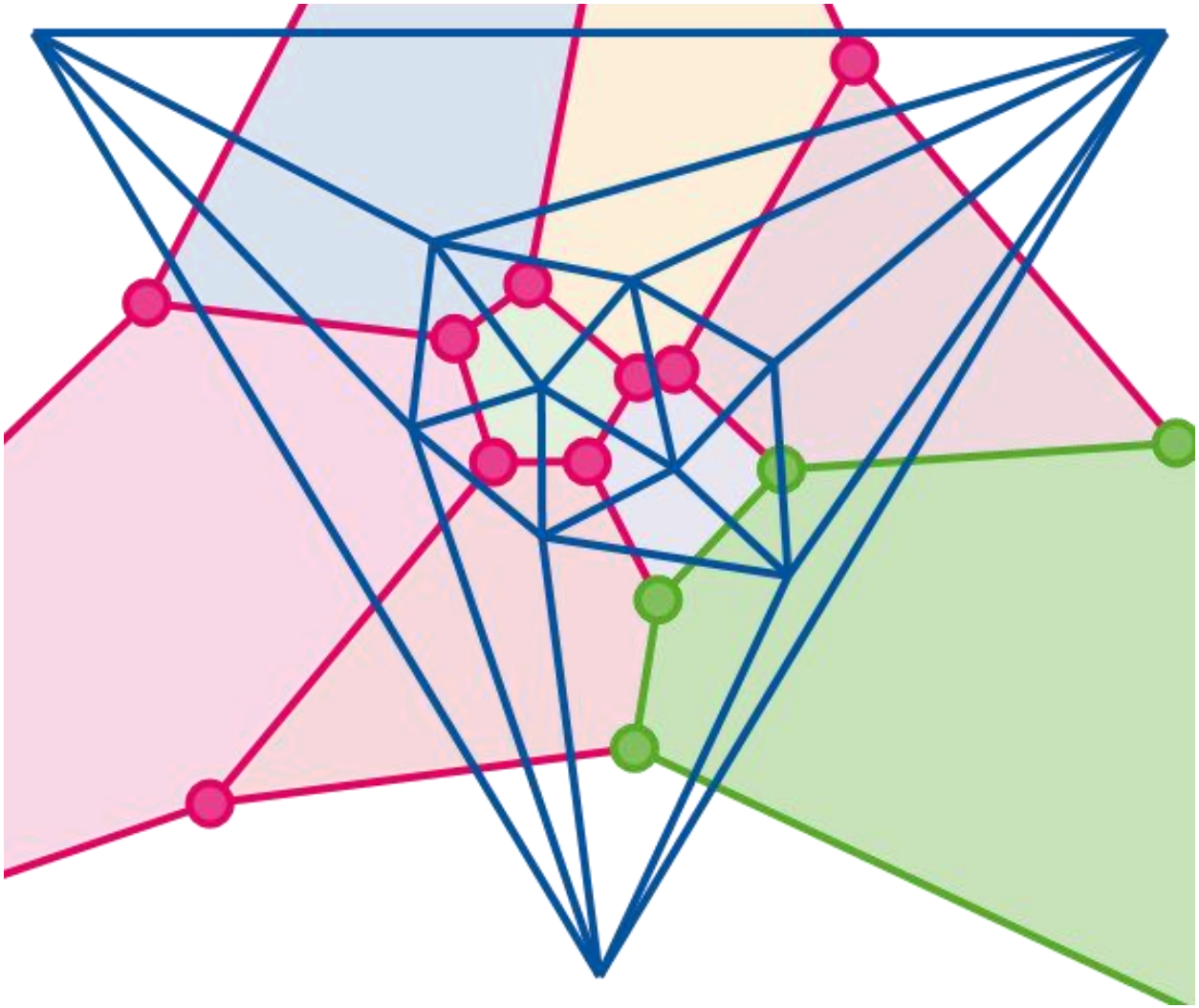
Vornoi Diagram from Delaunay Triangulation



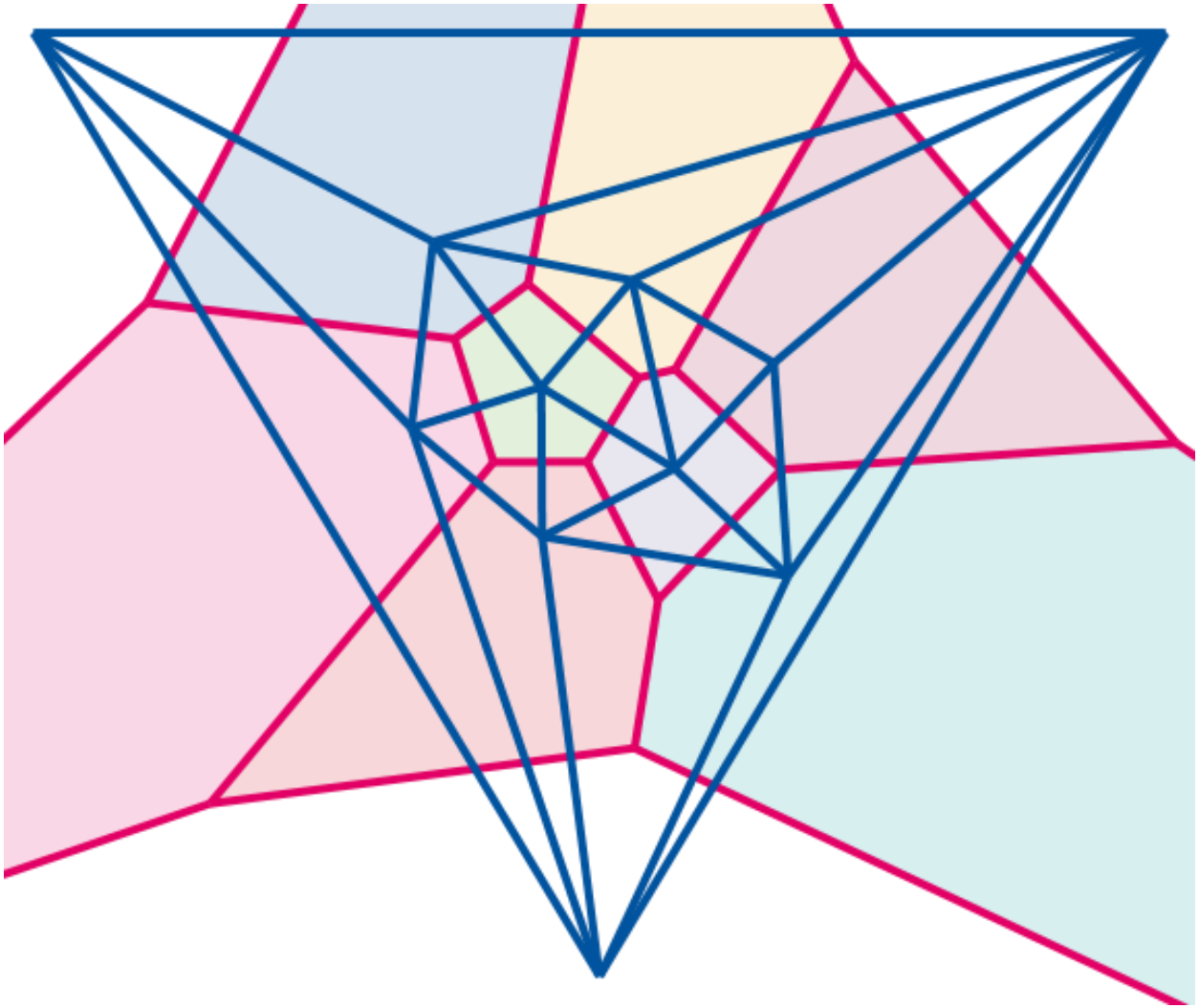
Vornoi Diagram from Delaunay Triangulation



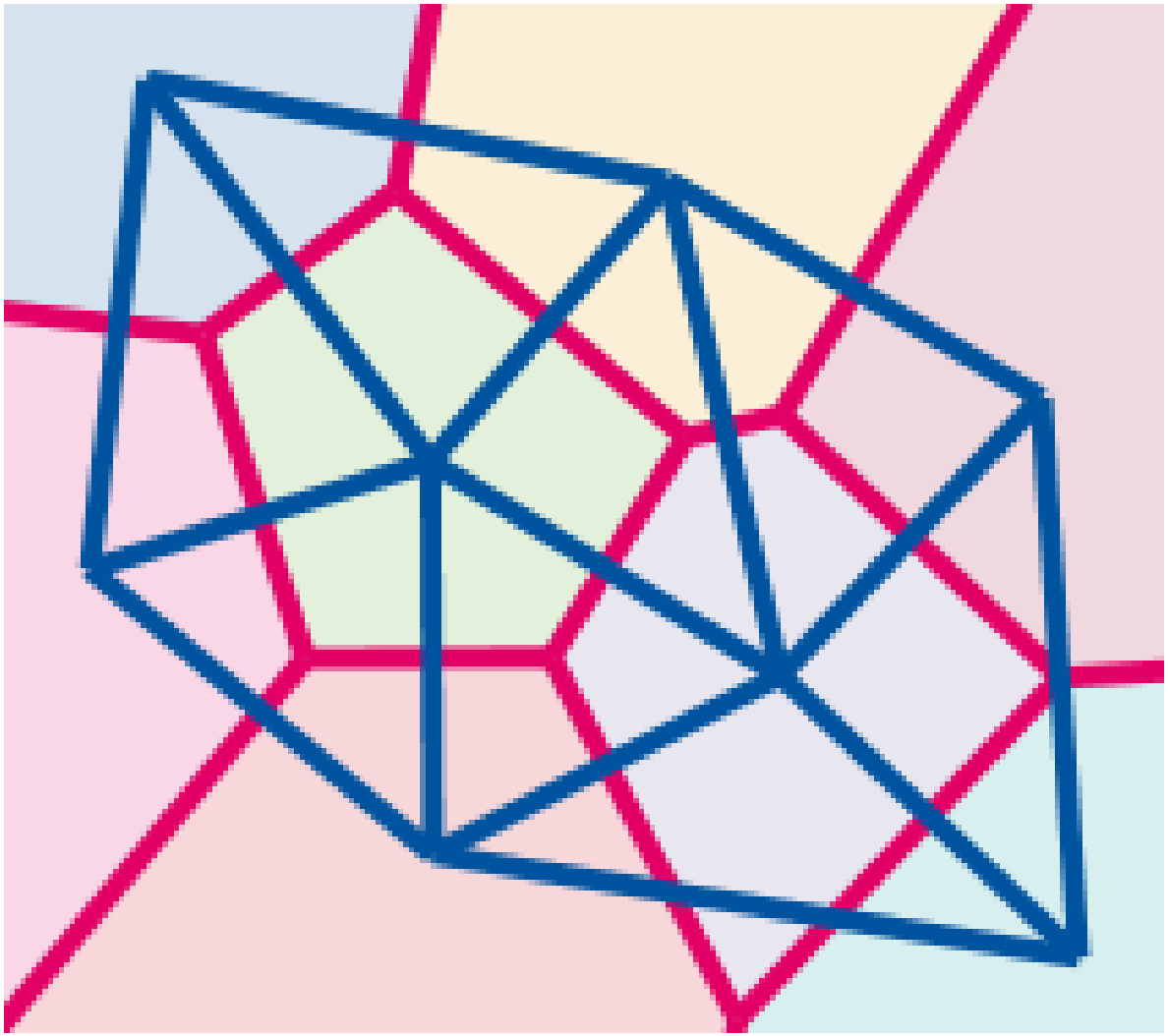
Vornoi Diagram from Delaunay Triangulation



Vornoi Diagram from Delaunay Triangulation



Vornoi Diagram from Delaunay Triangulation



Termination of Incremental Algorithm

Delaunay Triangulation: Incremental Algorithm

- Fix Delaunay Property using Edge Flips
- Can the algorithm infinitely Flip Edges

Termination of Incremental Algorithm

Delaunay Triangulation: Incremental Algorithm

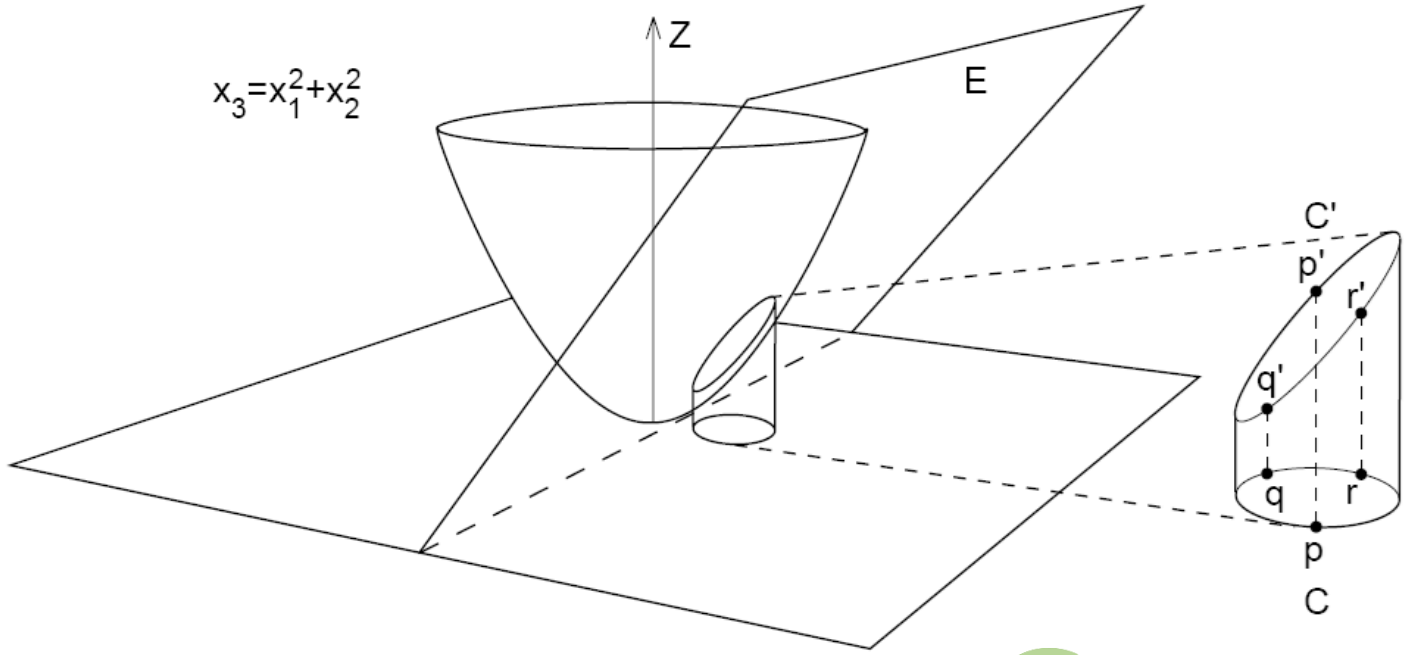
- Fix Delaunay Property using Edge Flips
- Can the algorithm infinitely Flip Edges
  - **NO.** The number of required flips is finite
  - The algorithm is always terminating

Termination of Incremental Algorithm

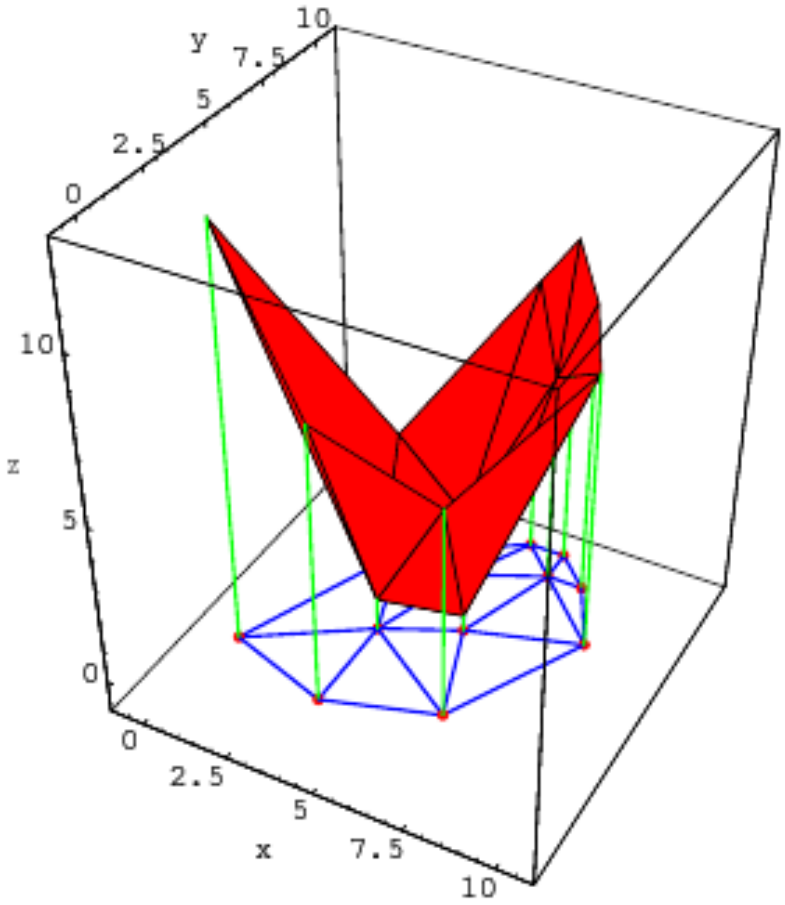
Lifting to 3-space

- Paraboloid $F: (x, y) \rightarrow (x, y, x^2 + y^2)$
- Circle: $(x - x_0)^2 + (y - y_0)^2 - r^2 = x^2 + y^2 - 2xx_0 - 2yy_0 + x_0^2 + y_0^2 - r^2 = 0$
- Plane: $z + ax + by + c = 0$

- Lifting of a circle is planar:



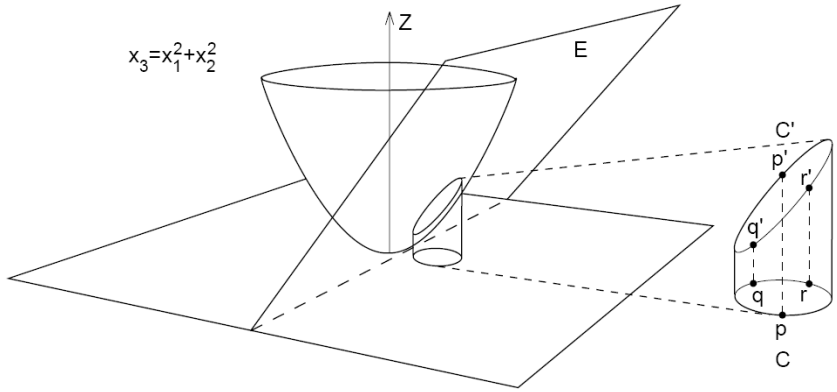
Lifting to 3-space: Example



Termination of Incremental Algorithm

Paraboloid is **convex**

1. Points within the circle lift to points below the plane
2. Points outside the circle lift to points above the plane

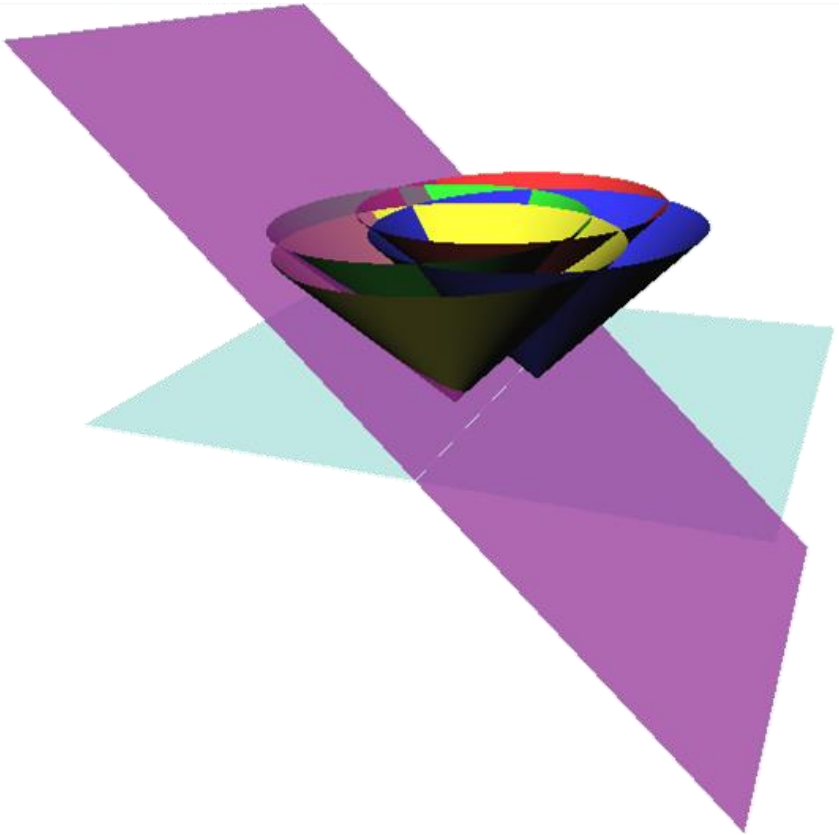


Theorem: The projection of the lower convex hull of the lifted points equals the Delaunay triangulation.

- Works for Delaunay triangulations in arbitrary dimensions!
- The convex hull can be computed in $\boldsymbol{O(n \log n)}$

Efficient Voronoi Computation: GPU

- 2D base points
- Represent distance function by **cone**
- Cone intersections = Vornoi edges
- Render using parallel projection with z-buffer



Sweep-Line algorithm (Fortune's algorithm)

Input: A set of points $p_1, \dots, p_n$

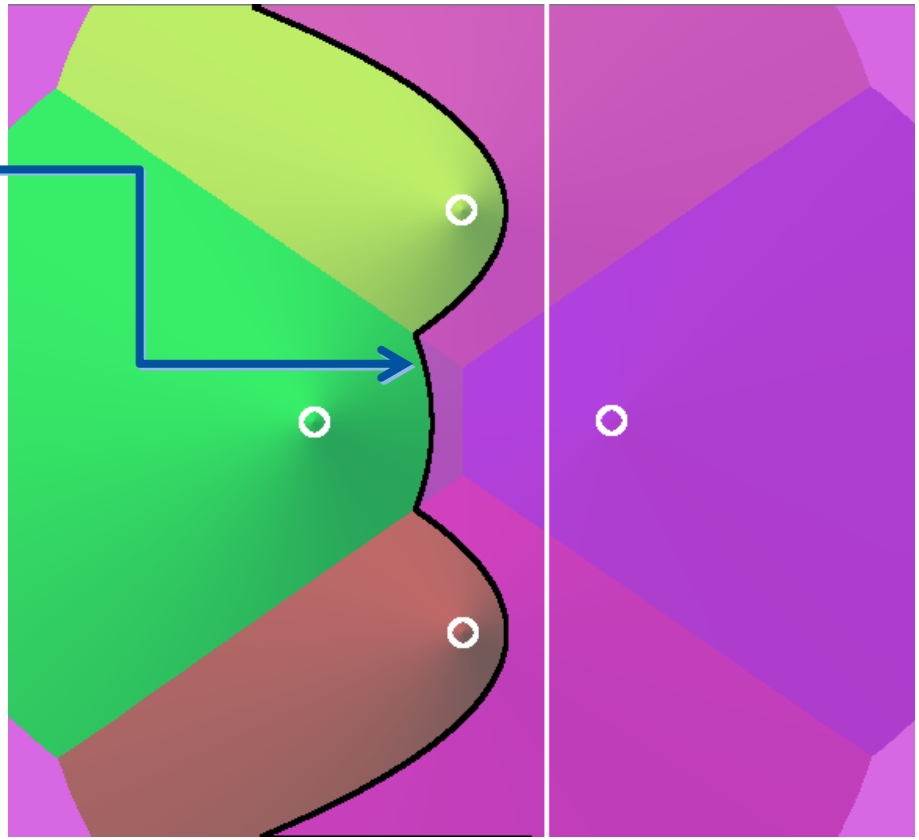
Output: Voronoi diagram of the points $p_1, \dots, p_n$

Algorithm:

1. Sort points by x-coordinate
2. Sweep along x-axis
3. Process evens
   - Point events
   - Circle events

Sweep-Line algorithm (Fortune's algorithm)

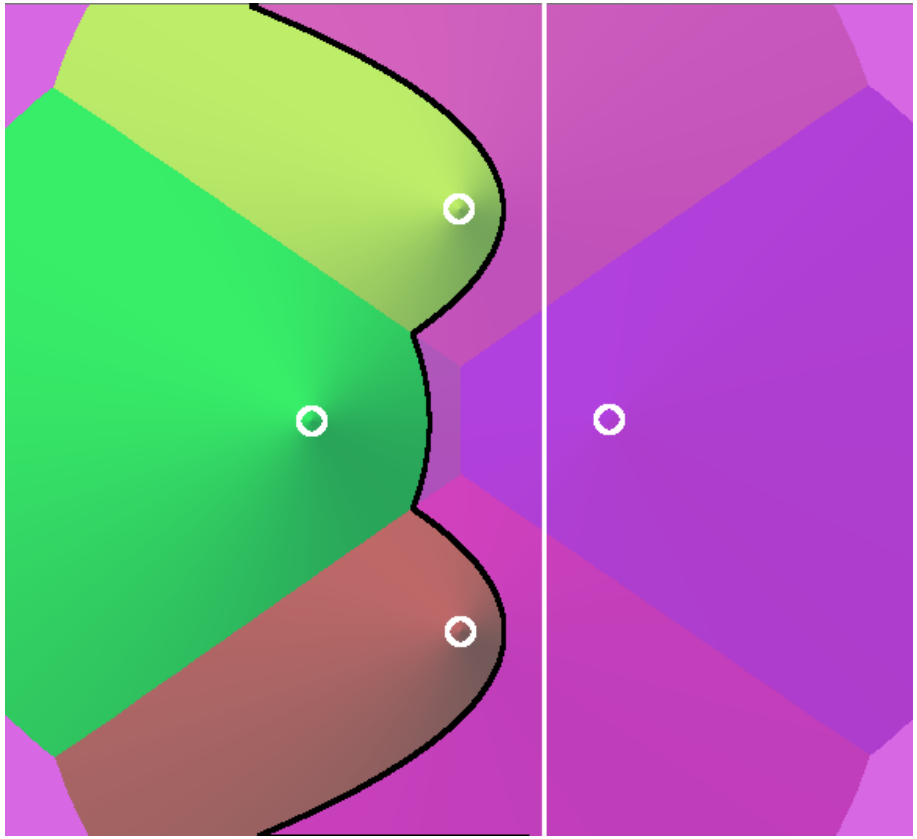
Beach Line

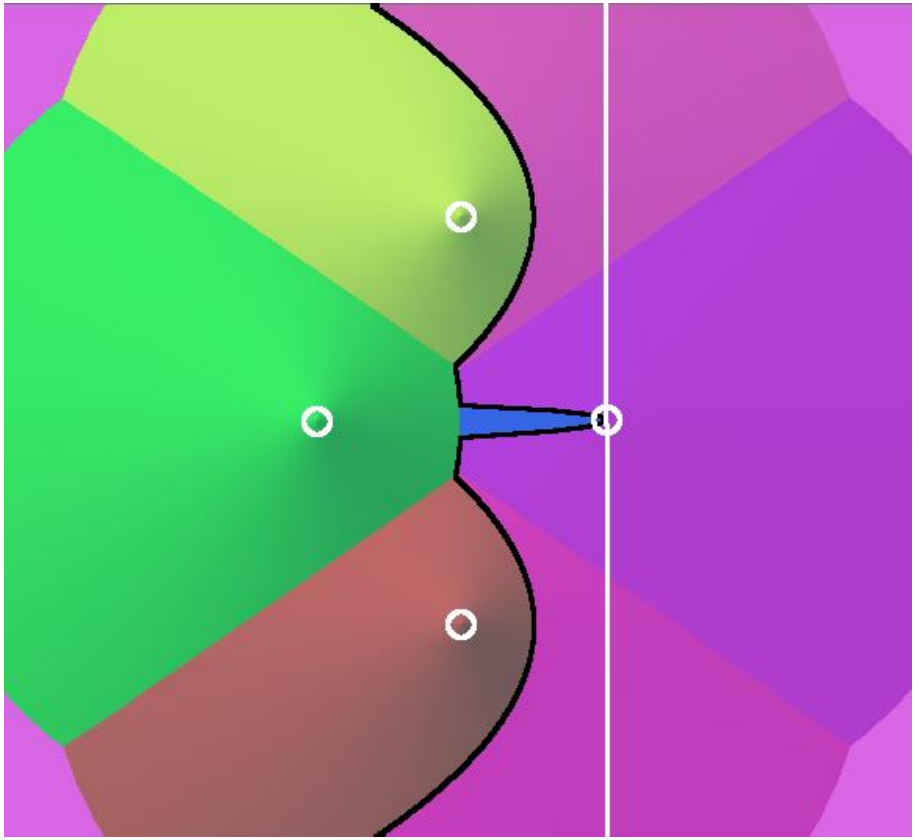


- First intersection between cones and sweep-plane
- Distance to site equals distance to sweepline

Sweep-Line algorithm (Fortune's algorithm)

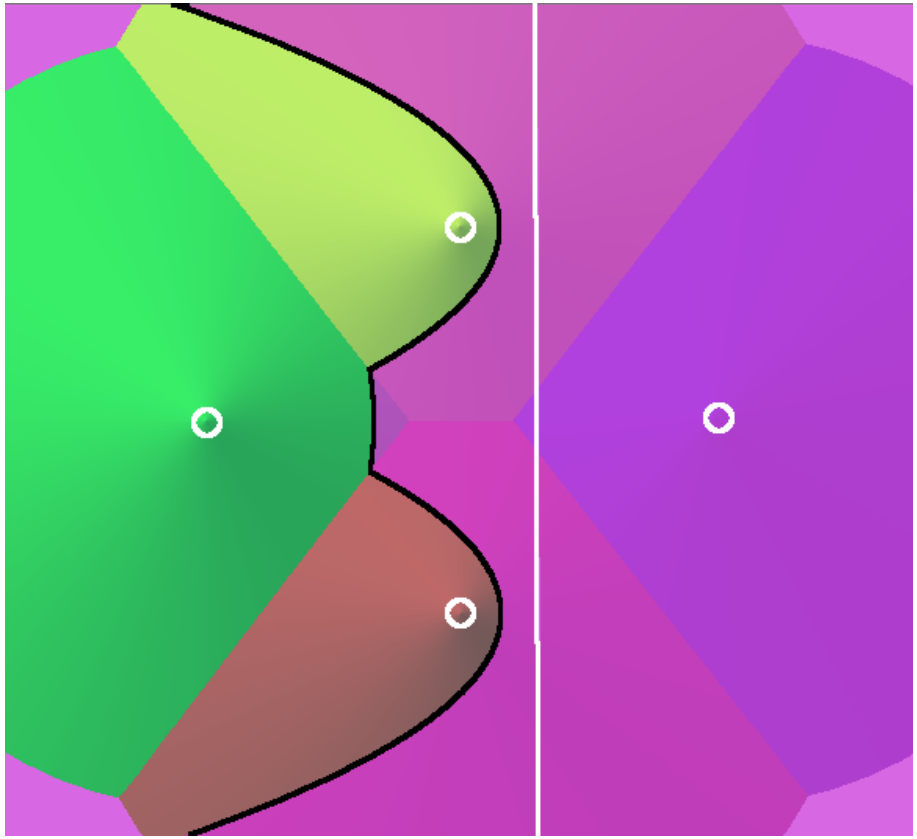
Point Event

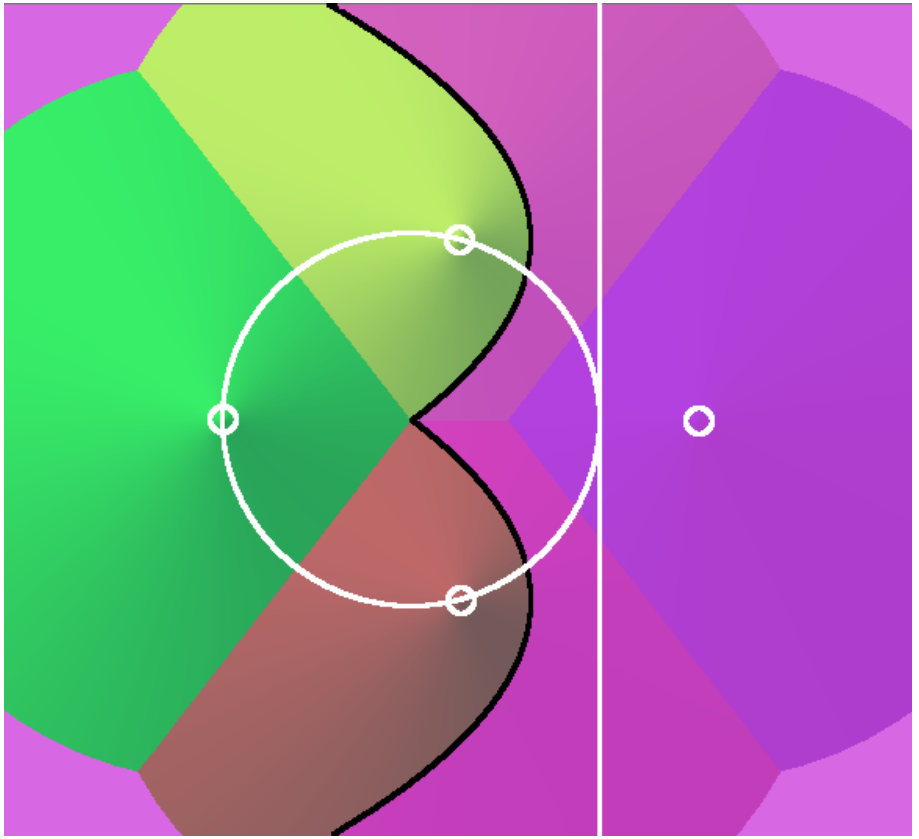




Sweep-Line algorithm (Fortune's algorithm)

Circle Event





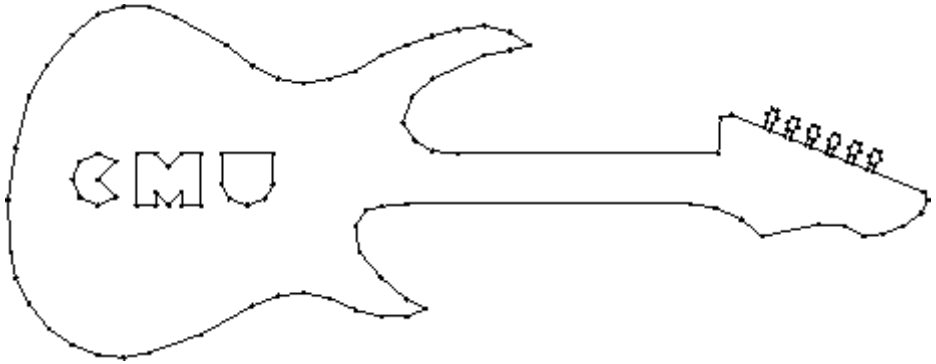
Sweep-Line algorithm (Fortune's algorithm)

Complexity:

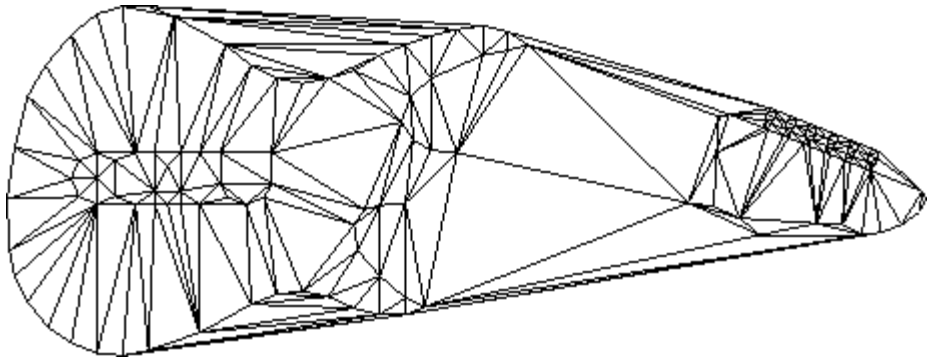
1. Sort points by x-coordinate $O(n \log n)$
2. Sweep along x-axis $O(n)$
3. Process evens $O(\log n)$
   - Point events
   - Circle events

Complexity: $\boldsymbol{O(n \log n)}$

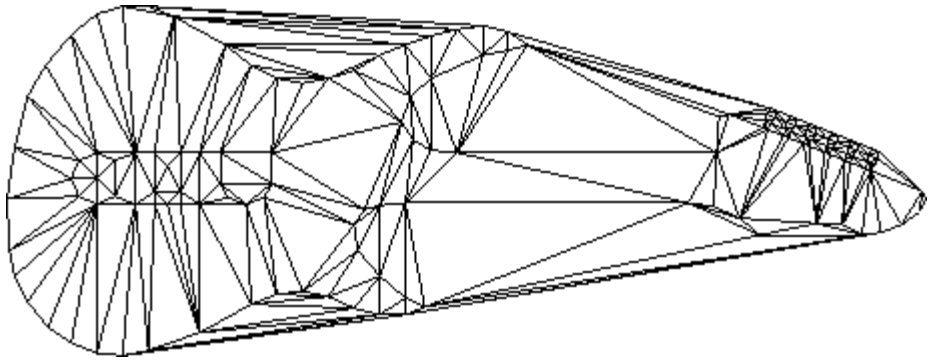
Delaunay Refinement



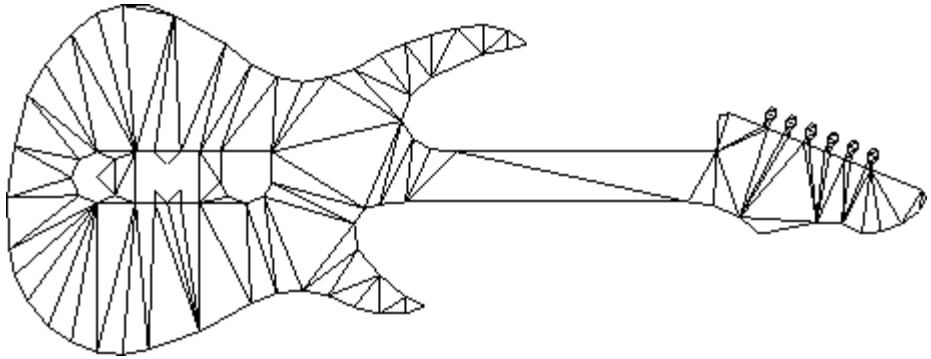
Delaunay Refinement



Delaunay Refinement



Delaunay Refinement



Delaunay Refinement