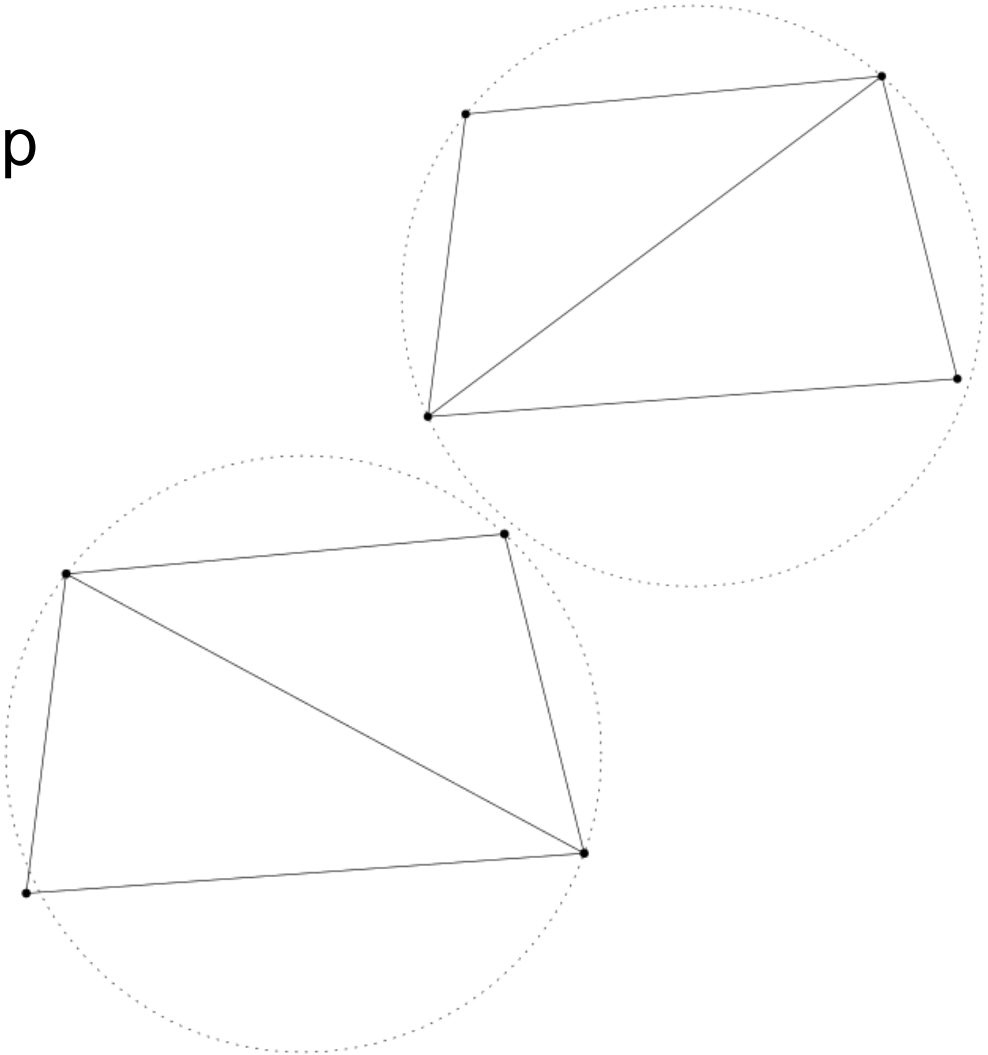
Delaunay Triangulation of Points on a 2D Plane

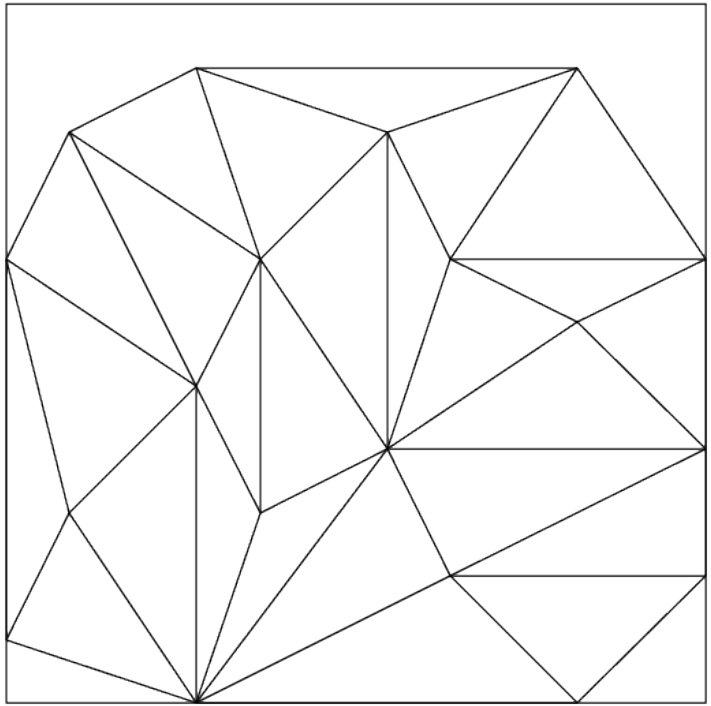
Sai Tanmay Reddy Chakkera

Method: Lawson-Edge Swap

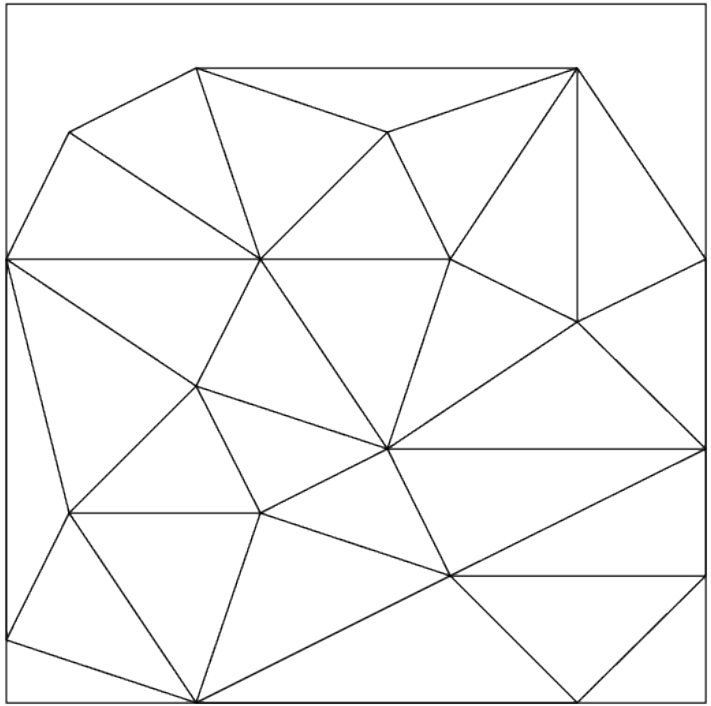
- Swap illegal edges with legal edges. Legality tested with an incircle test.
- Lifting to parabola argument helps us determine running time to be $O(n^2)$.



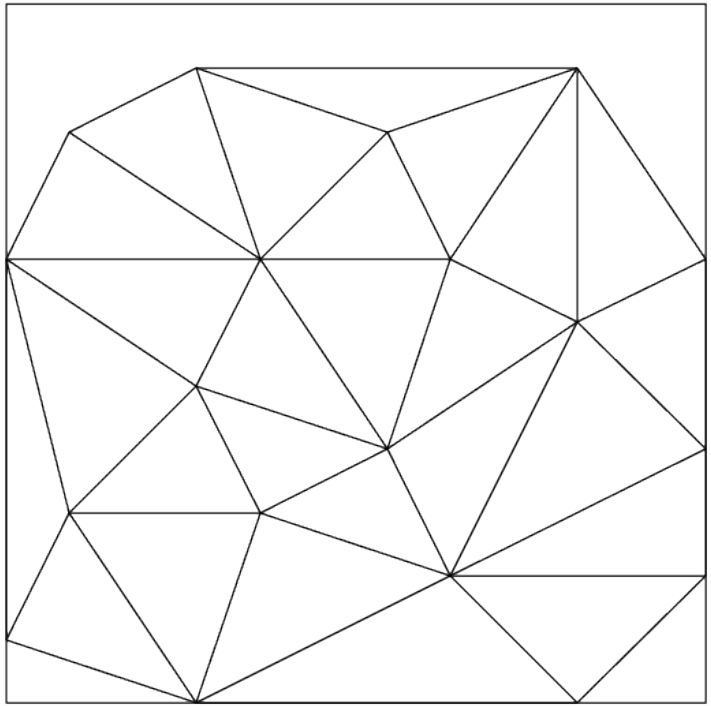
Visualization



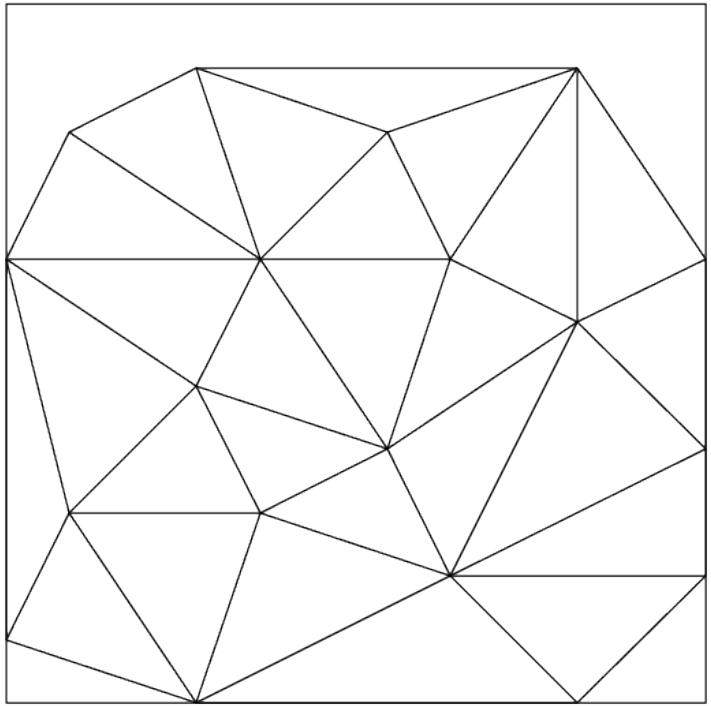
Visualization



Visualization

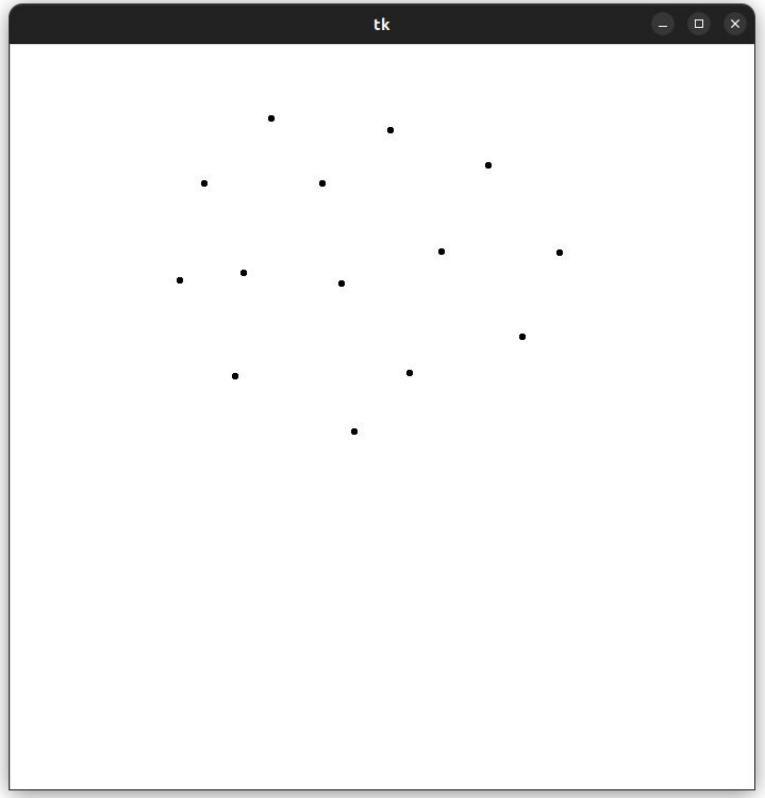


Visualization



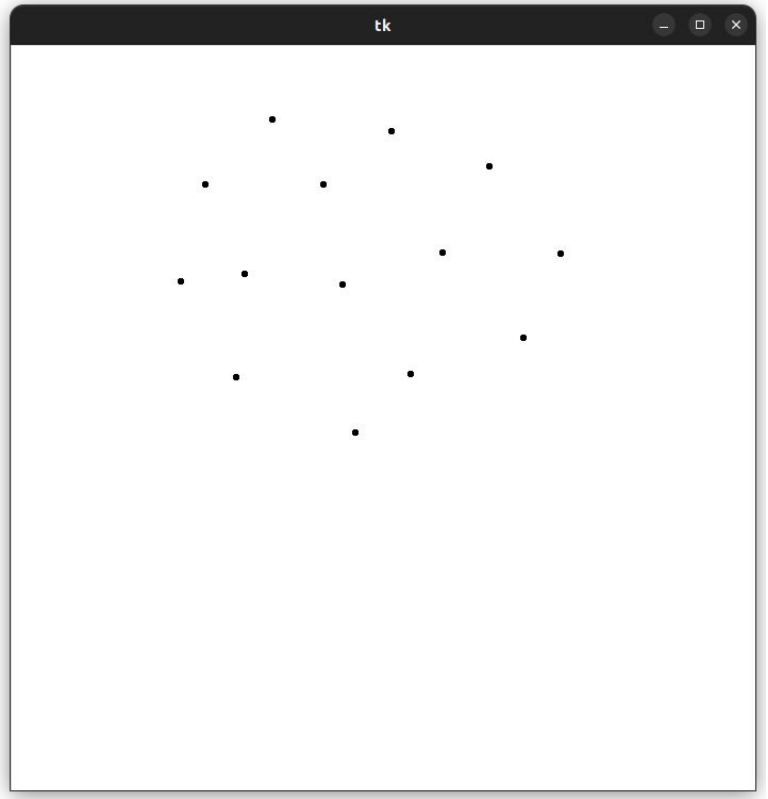
Delaunay Triangulation on Random Points?

- Lawson edge swap algorithm begins by assuming the presence of a random triangulation.
- How to get to that from the given set of points?



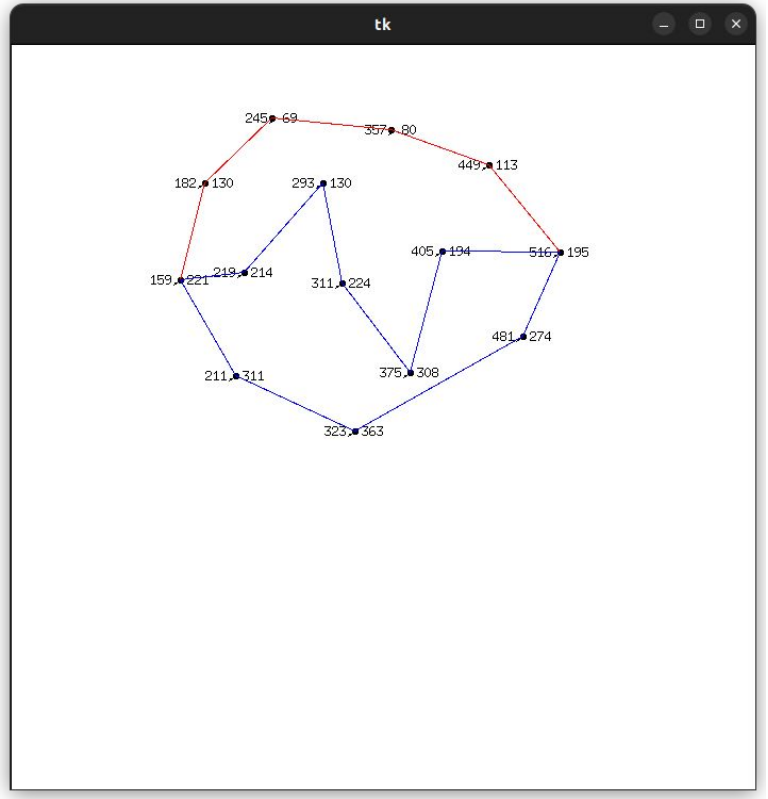
Conditions to random triangulation

- The delaunay triangulation of any given set of points will include its convex hull.
- But the edge-swap algorithm doesn’t address edges at the boundary (adjacent to the unbounded face).
- Convex hull must be a part of the random triangulation



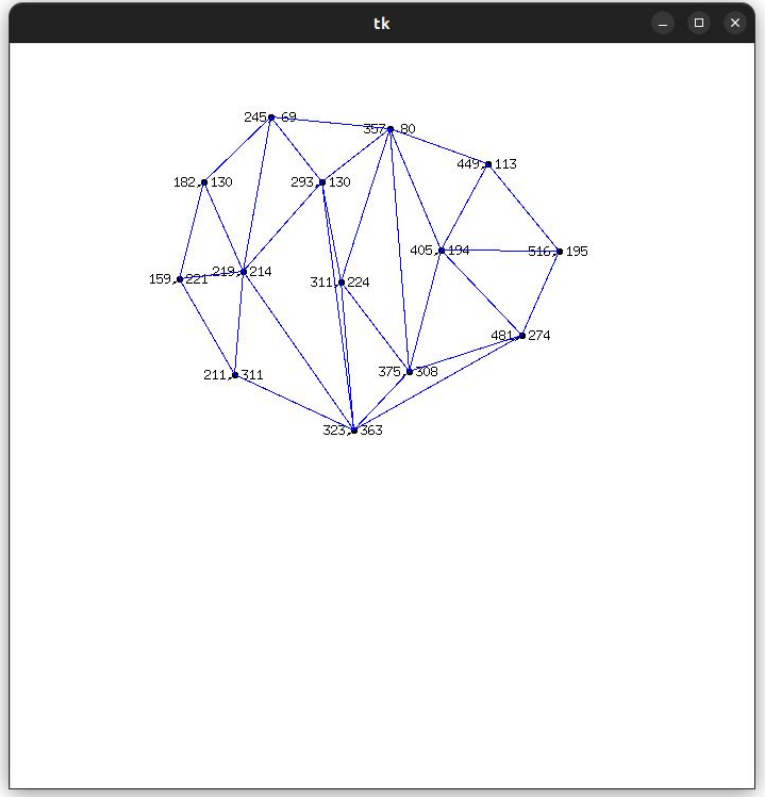
What about internal points?

- After finding the convex hull, the points inside still remain without edges.
- Sort these points by their x-coordinates. And join consecutive points in this order. And join this chain with the leftmost and rightmost points of the convex-hull boundary.



What next?

- We have 2 x-monotone polygons. Use the O(n) algorithm to triangulate it.
- The idea is to triangulate everything to the left (or right) of the current vertex and forget about the already triangulated region.
- Use a stack to maintain the list of vertices that have been seen, but haven’t been triangulated (didn’t add diagonals to it).
- Further details in BCKO.



EVERYTHING, NOT EVERYWHERE, ALL AT ONCE

- Find convex hull of points
- Partition convex polygon into two monotone polygons
- Triangulate each of the 2 monotone polygons
- Merge along the common chain
- Lawson-edge swap