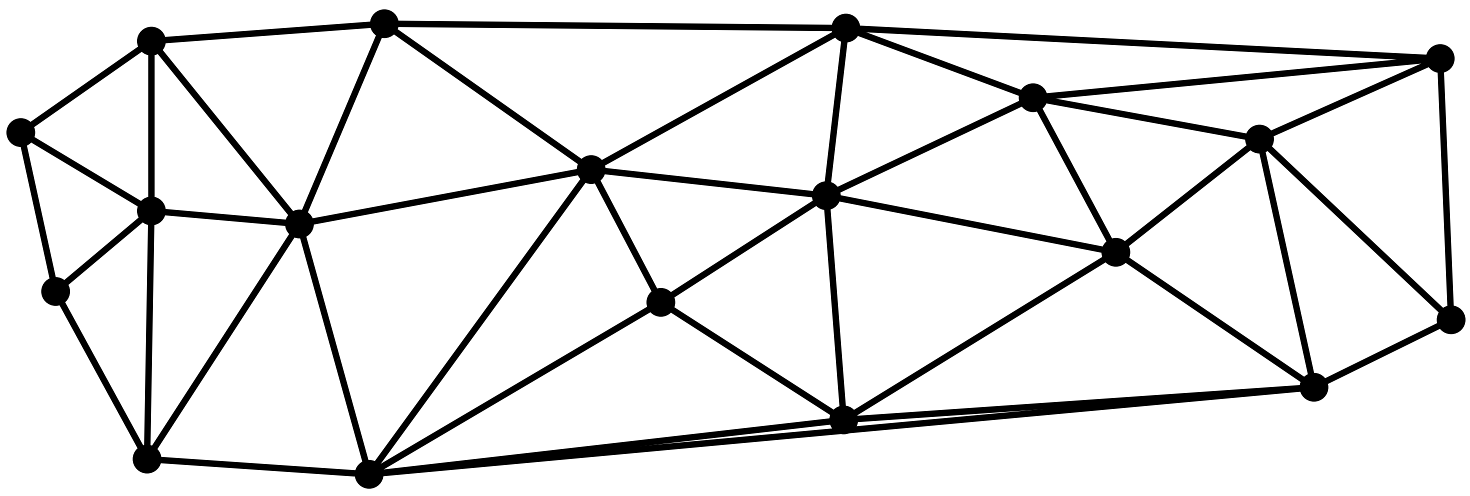
Triangulations

Birgit Vogtenhuber

Slides partially by Alexander Pilz

Overview

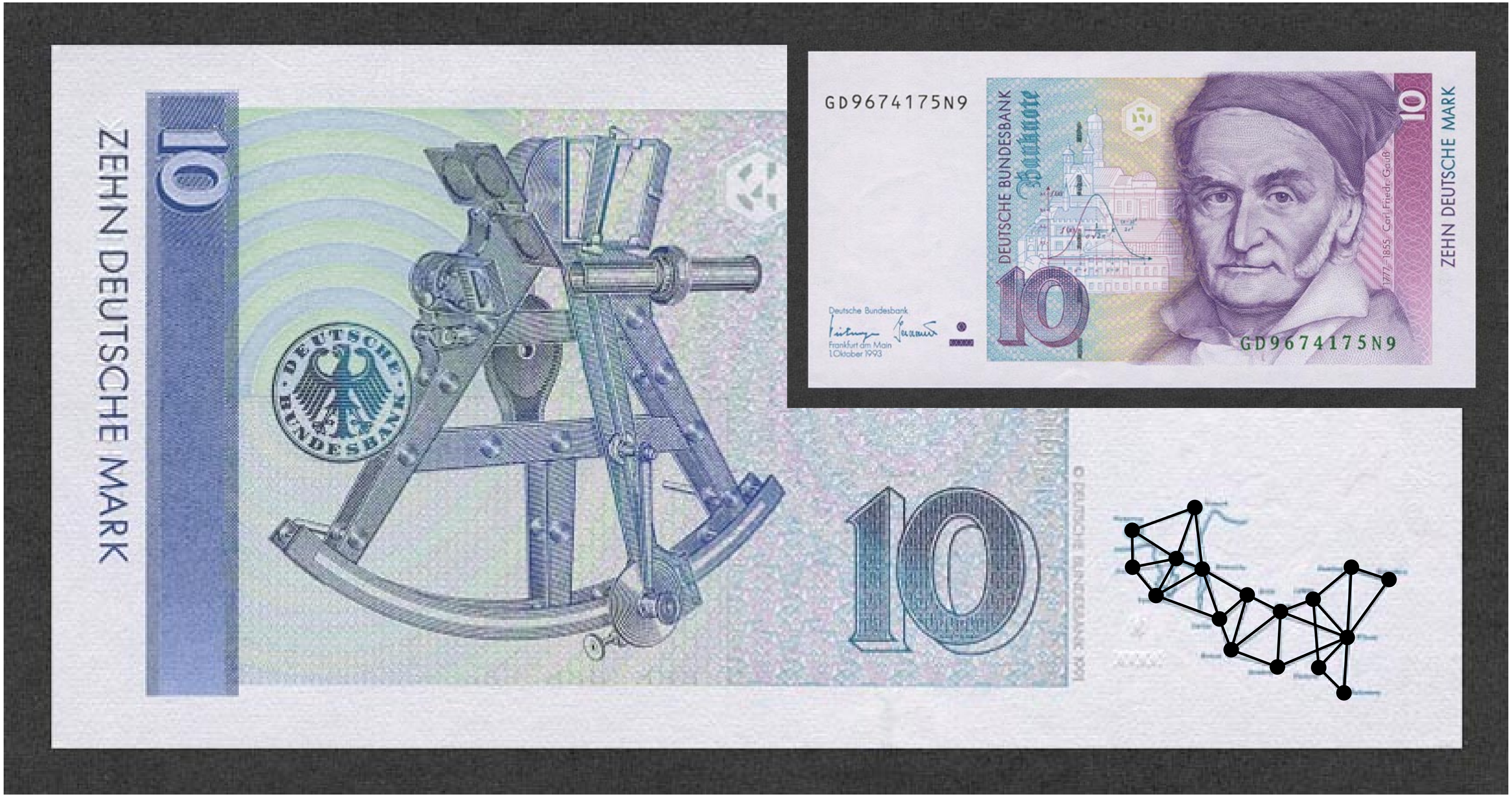
- Description of an important geometric data structure
- Combinatorial properties
- Overview of applications
- Construction
- Local transformations



A Natural Data Structure

- Triangles are the simplest piece of a surface
- First uses of triangular networks in cartography

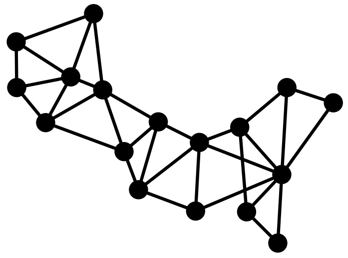
A Natural Data Structure



A Natural Data Structure

- Triangles are the simplest piece of a surface
- First uses of triangular networks in cartography

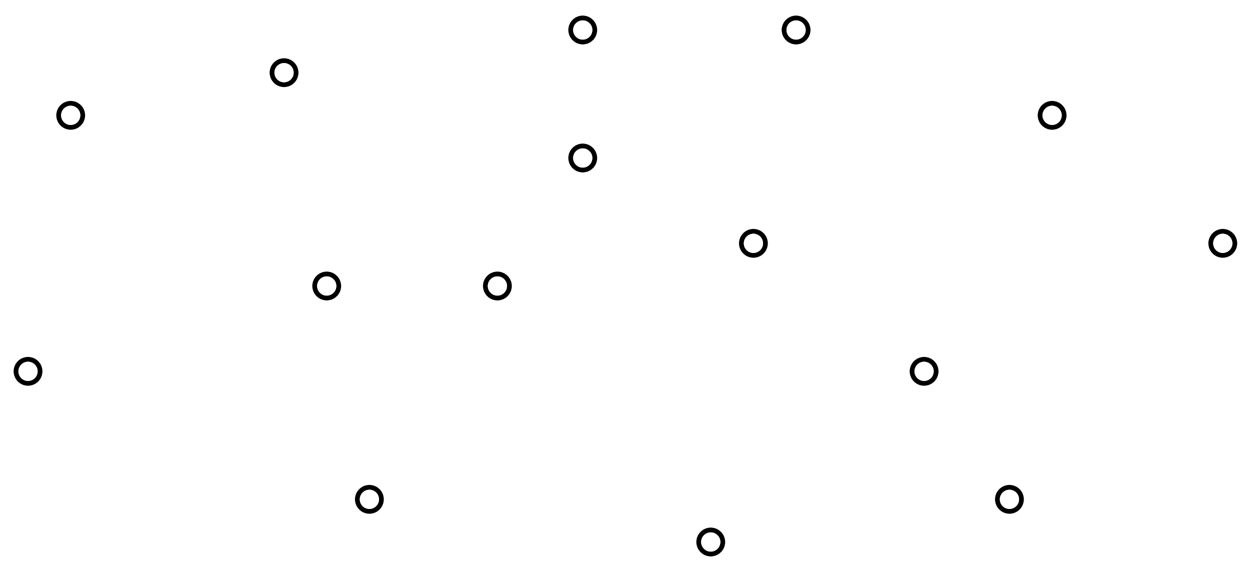
- Allows modelling surfaces
- Important and widely used in Computer Graphics
- Most straightforward way to partition the plane
- Computational Geometry addresses triangulations
  - of point sets
  - of polygons
  - and many other related structures.



Triangulation of a Point Set

Triangulation of a point set S in the plane:

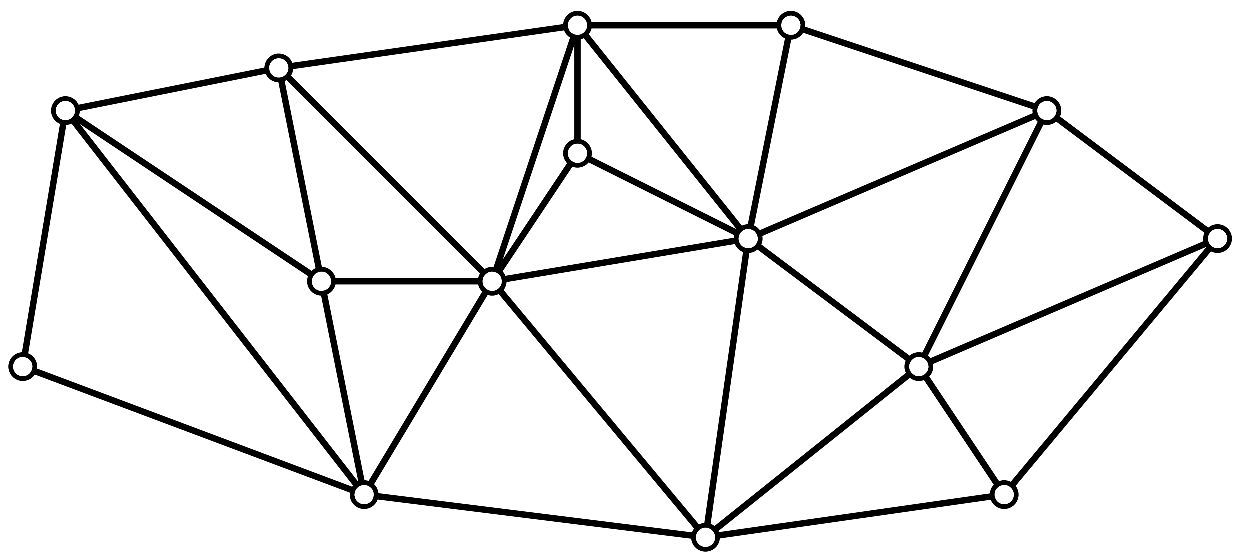
- Partition of the convex hull of S into interior-disjoint triangles whose edges are segments spanned by S.
- No point of S lies inside a segment or a triangle.



Triangulation of a Point Set

Triangulation of a point set S in the plane:

- Partition of the convex hull of S into interior-disjoint triangles whose edges are segments spanned by S.
- No point of S lies inside a segment or a triangle.

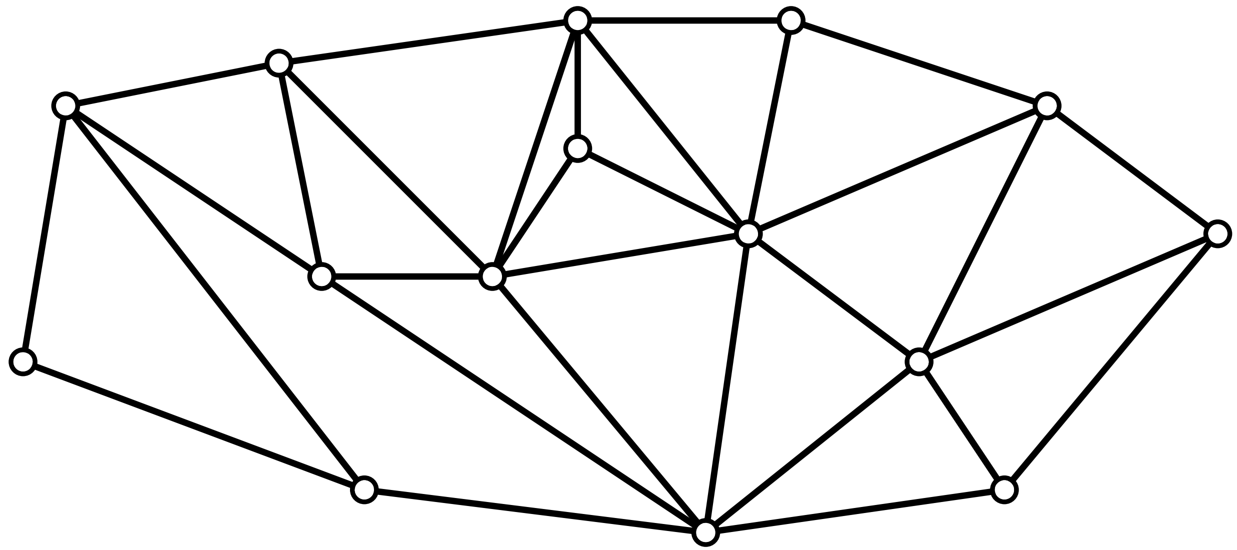


A triangulation of S.

Triangulation of a Point Set

Triangulation of a point set S in the plane:

- Partition of the convex hull of S into interior-disjoint triangles whose edges are segments spanned by S.
- No point of S lies inside a segment or a triangle.

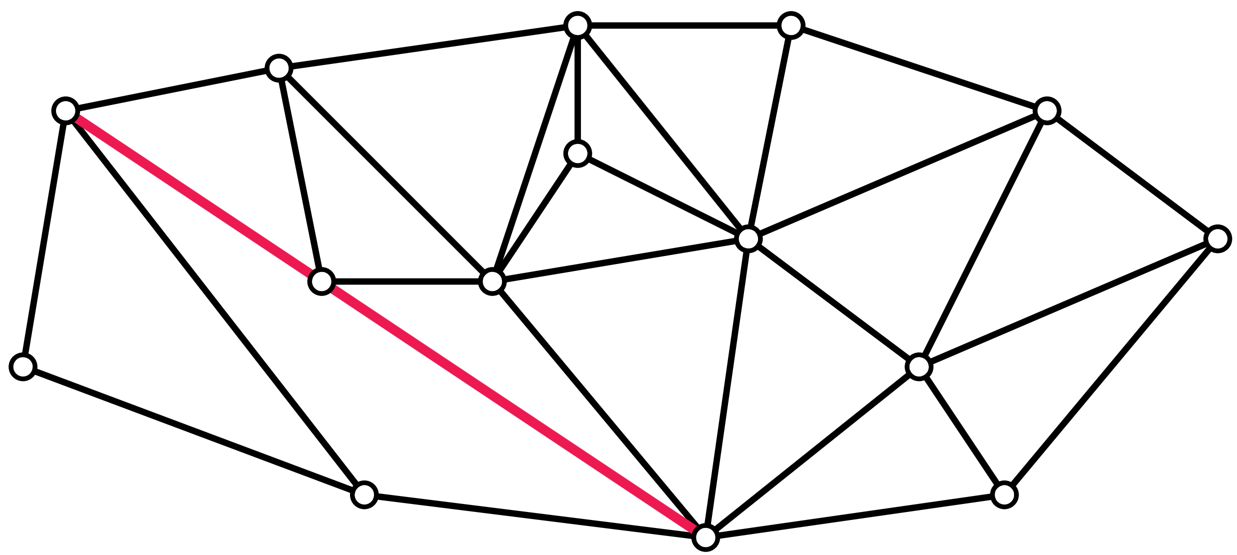


Not a triangulation of S. **Why?**

Triangulation of a Point Set

Triangulation of a point set S in the plane:

- Partition of the convex hull of S into interior-disjoint triangles whose edges are segments spanned by S.
- No point of S lies inside a segment or a triangle.

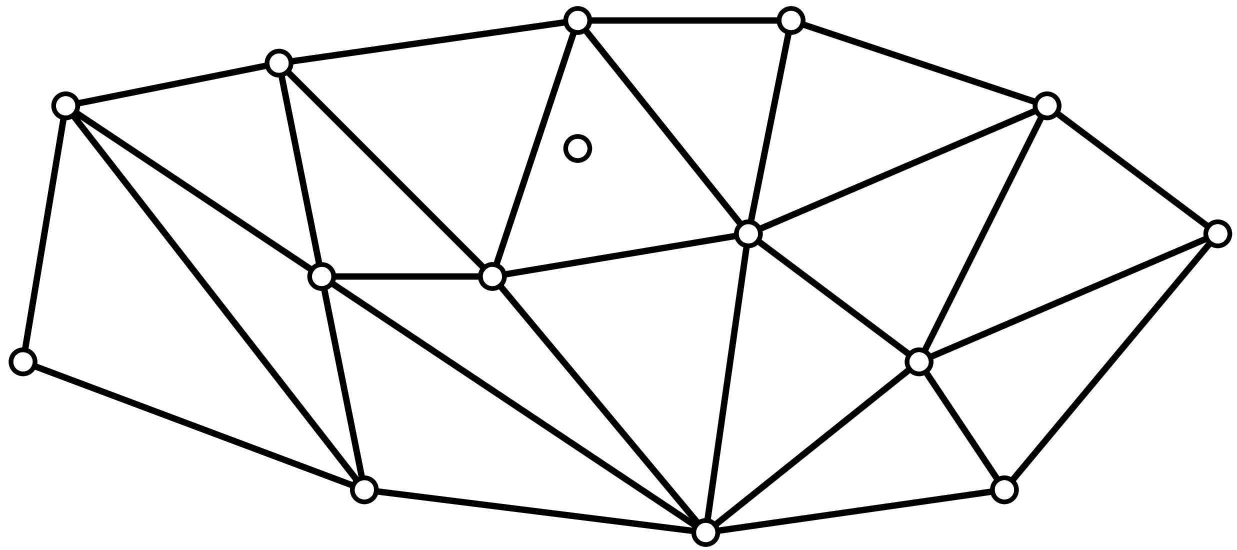


Not a triangulation of S. **Why?**

Triangulation of a Point Set

Triangulation of a point set S in the plane:

- Partition of the convex hull of S into interior-disjoint triangles whose edges are segments spanned by S.
- No point of S lies inside a segment or a triangle.

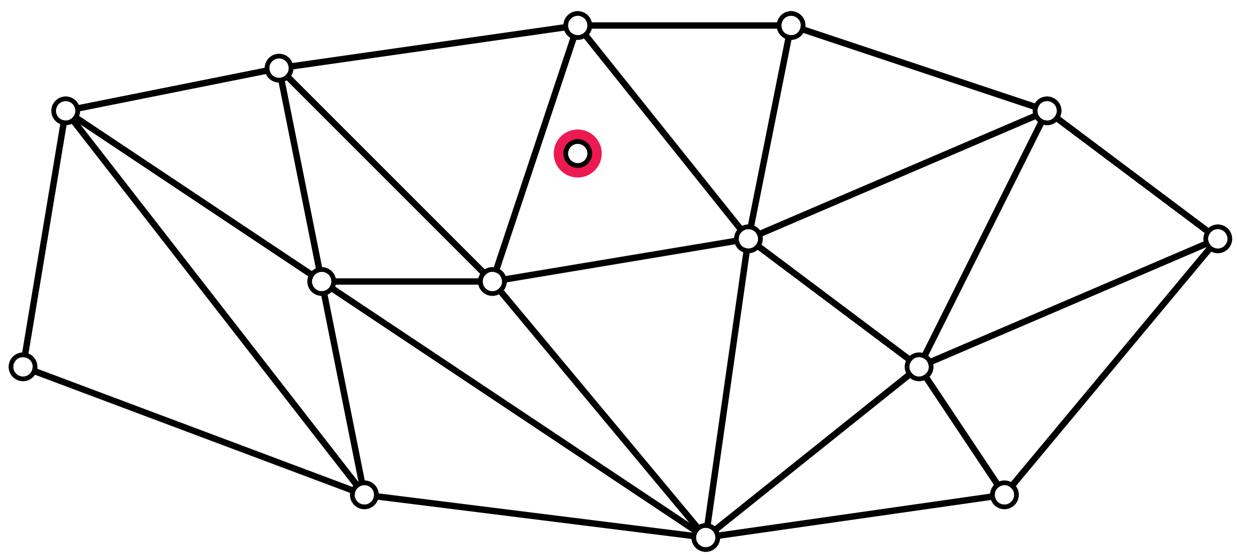


Not a triangulation of S. **Why?**

Triangulation of a Point Set

Triangulation of a point set S in the plane:

- Partition of the convex hull of S into interior-disjoint triangles whose edges are segments spanned by S.
- No point of S lies inside a segment or a triangle.

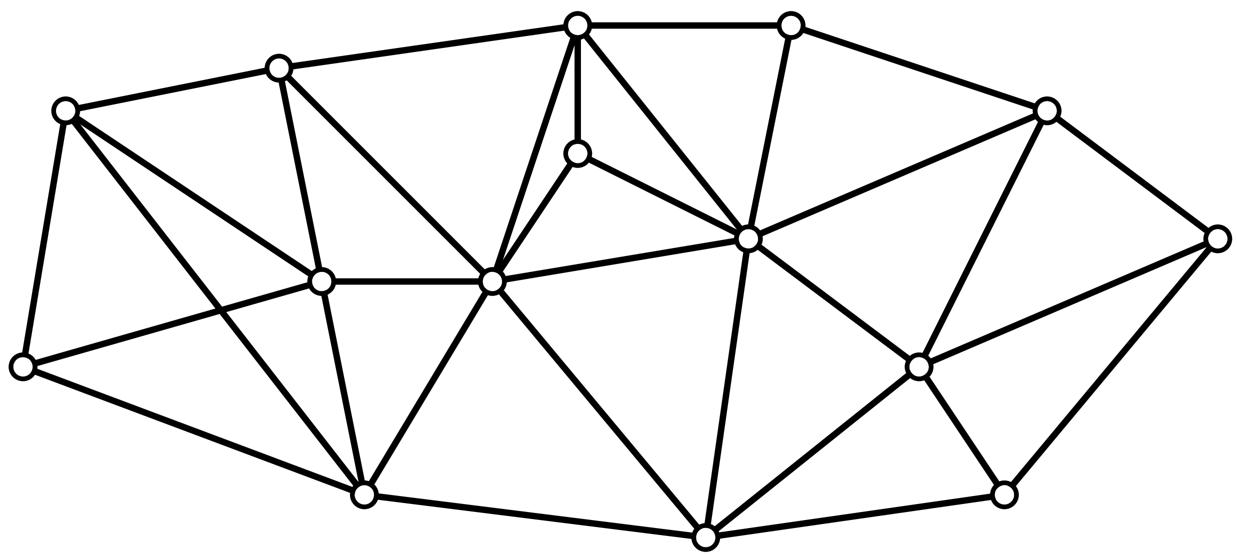


Not a triangulation of S. **Why?**

Triangulation of a Point Set

Triangulation of a point set S in the plane:

- Partition of the convex hull of S into interior-disjoint triangles whose edges are segments spanned by S.
- No point of S lies inside a segment or a triangle.

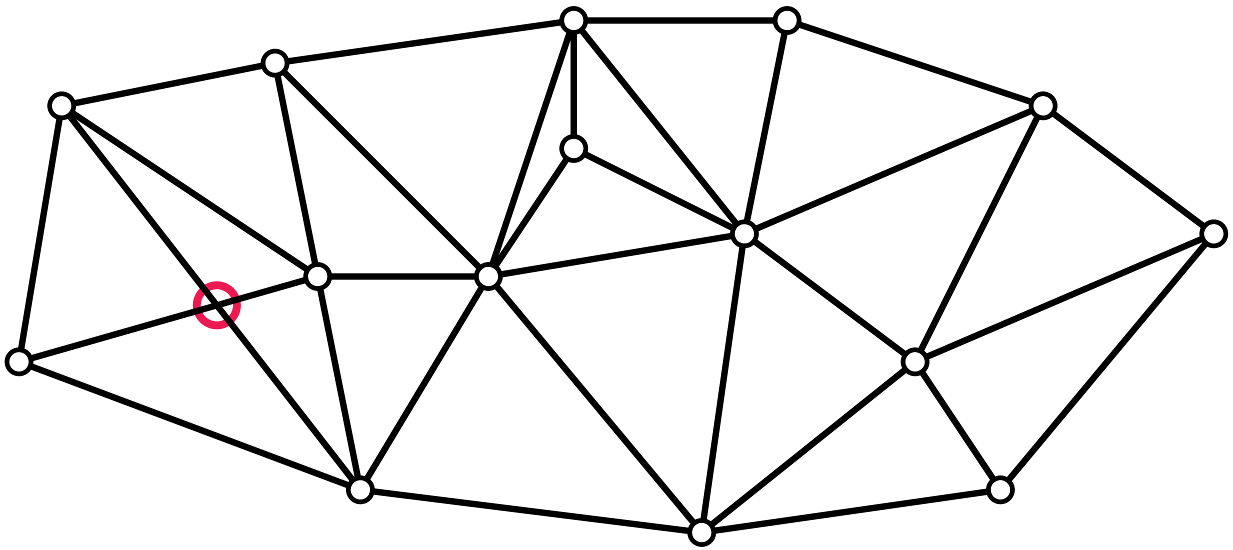


Not a triangulation of S. **Why?**

Triangulation of a Point Set

Triangulation of a point set S in the plane:

- Partition of the convex hull of S into interior-disjoint triangles whose edges are segments spanned by S.
- No point of S lies inside a segment or a triangle.

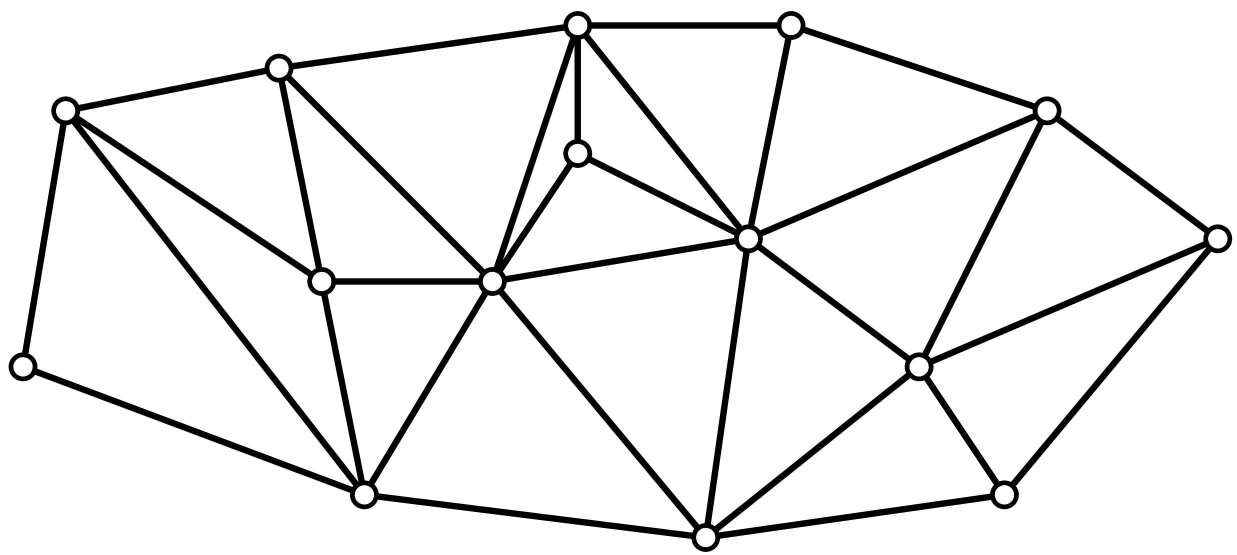


Not a triangulation of S. **Why?**

Triangulation of a Point Set

Triangulation of a point set S in the plane:

- Partition of the convex hull of S into interior-disjoint triangles whose edges are segments spanned by S.
- No point of S lies inside a segment or a triangle.

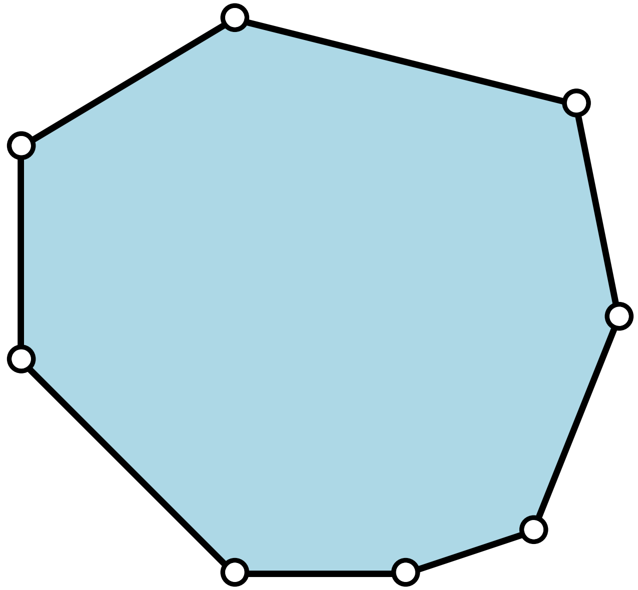


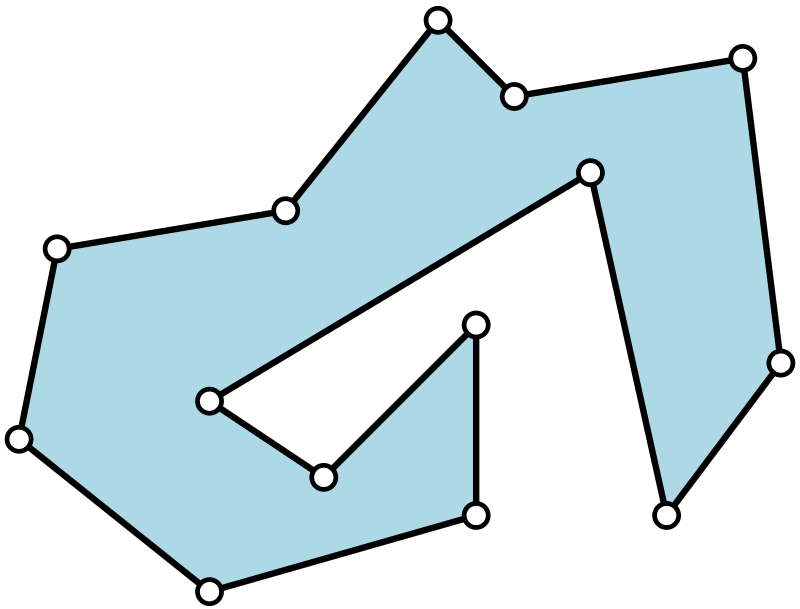
Alternative definition:
Maximal plane straight-line graph on S.

Triangulation of a Polygon

Triangulation of a polygon P with vertex set S:

- Partition of P into interior-disjoint triangles whose edges are segments spanned S.
- No point of S lies inside a segment (or a triangle).

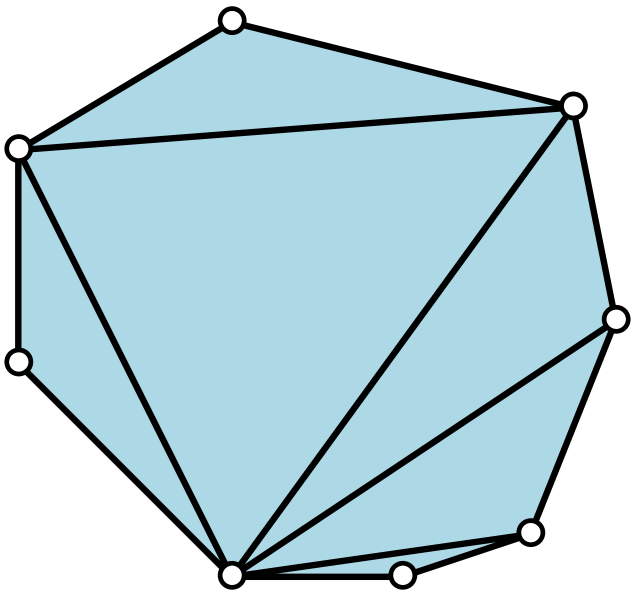


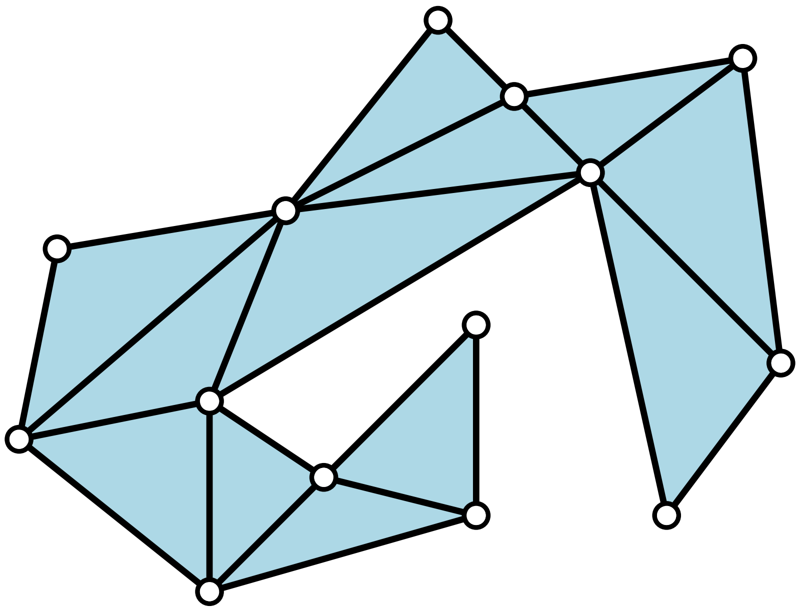


Triangulation of a Polygon

Triangulation of a polygon P with vertex set S:

- Partition of P into interior-disjoint triangles whose edges are segments spanned S.
- No point of S lies inside a segment (or a triangle).

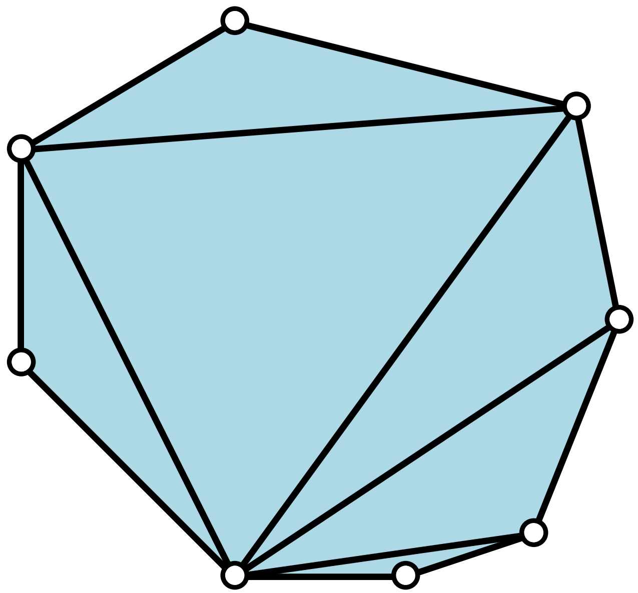


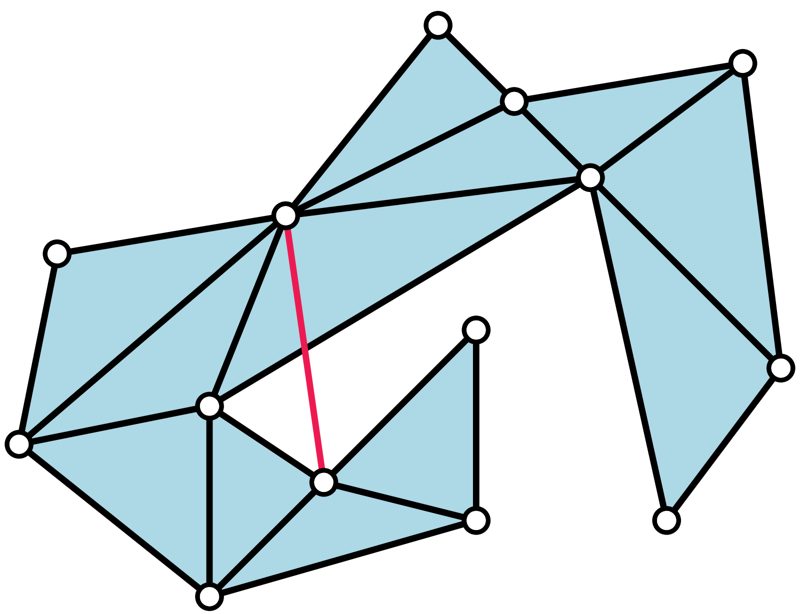


Triangulation of a Polygon

Triangulation of a polygon P with vertex set S:

- Partition of P into interior-disjoint triangles whose edges are segments spanned S.
- No point of S lies inside a segment (or a triangle).



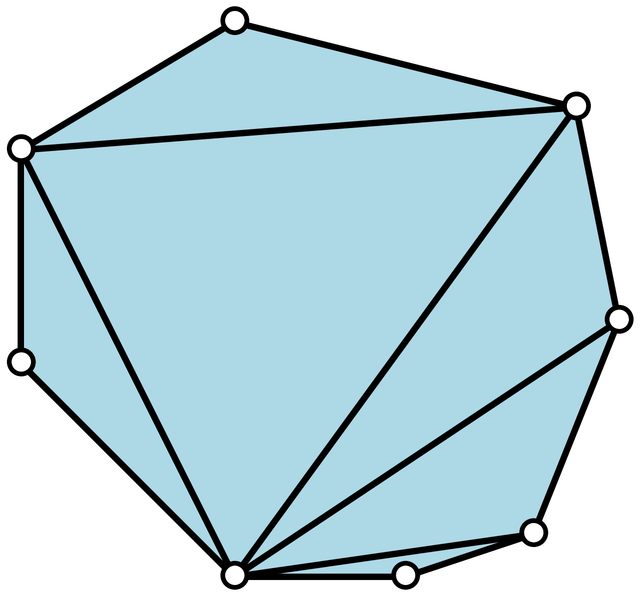


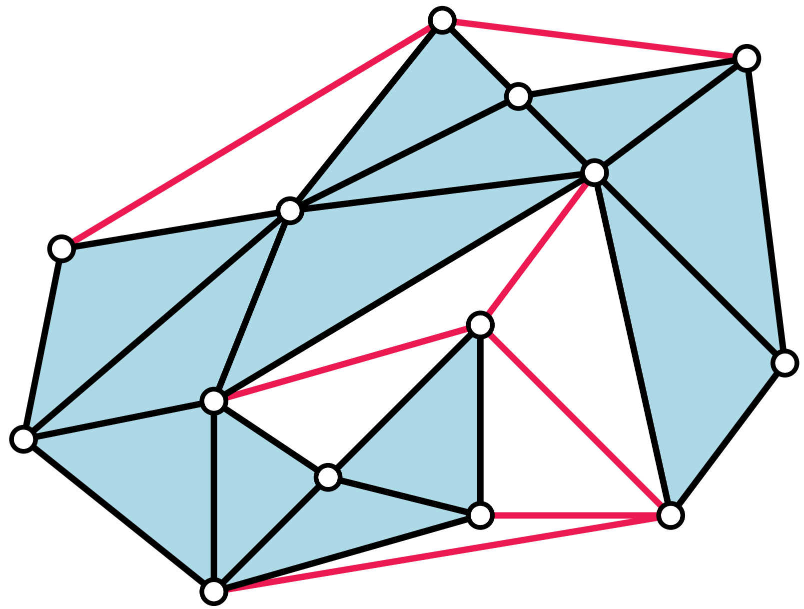
Not a triangulation of P. **Why?**

Triangulation of a Polygon

Triangulation of a polygon P with vertex set S:

- Partition of P into interior-disjoint triangles whose edges are segments spanned S.
- No point of S lies inside a segment (or a triangle).



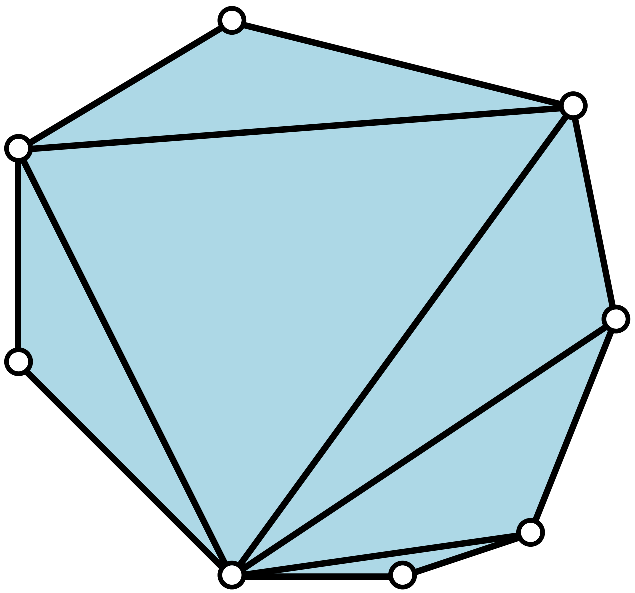


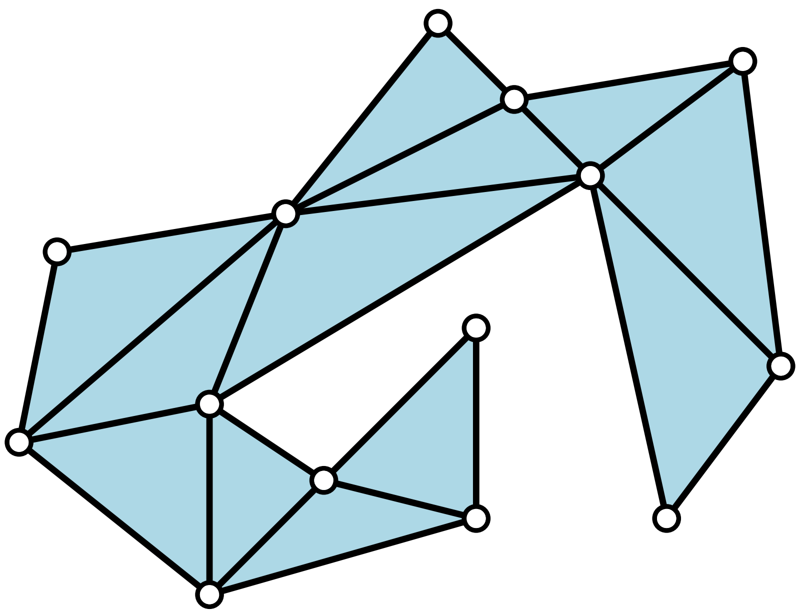
Not a triangulation of P. **Why?**

Triangulation of a Polygon

Triangulation of a polygon P with vertex set S:

- Partition of P into interior-disjoint triangles whose edges are segments spanned S.
- No point of S lies inside a segment (or a triangle).

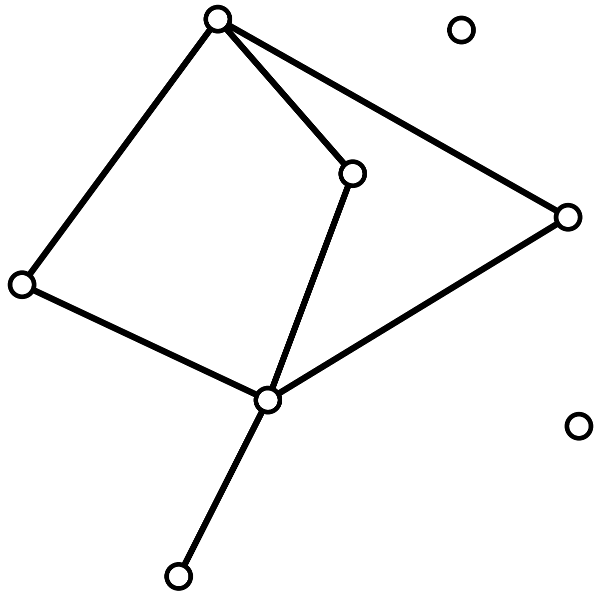




Alternative definition:
Maximal plane straight-line graph on S inside P.

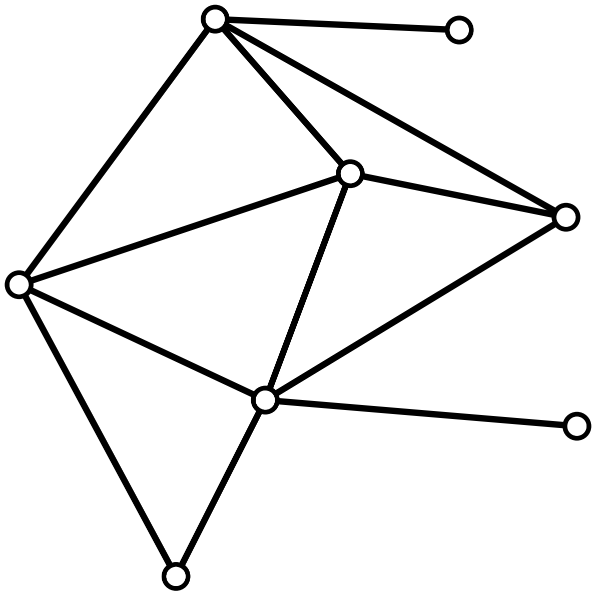
Maximal Plane Straight-Line Graphs

- Given a plane straight-line graph
- We add edges as long as we can.



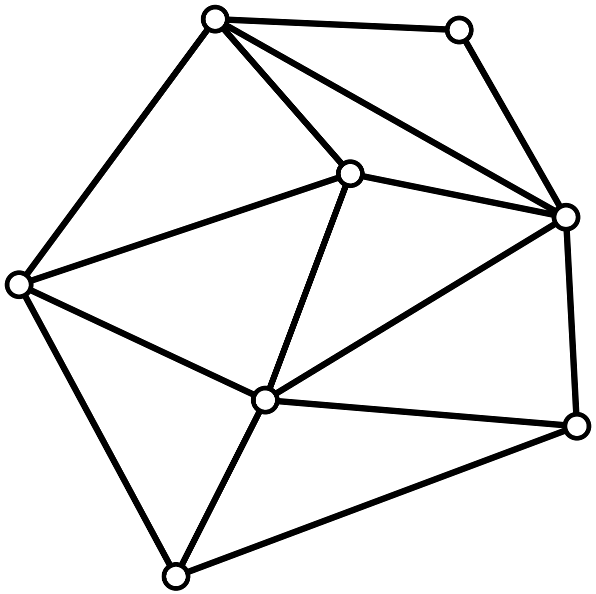
Maximal Plane Straight-Line Graphs

- Given a plane straight-line graph
- We add edges as long as we can.



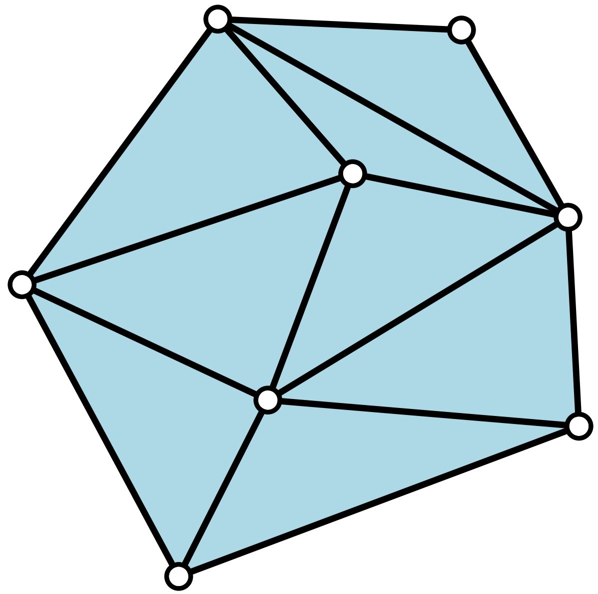
Maximal Plane Straight-Line Graphs

- Given a plane straight-line graph
- We add edges as long as we can.



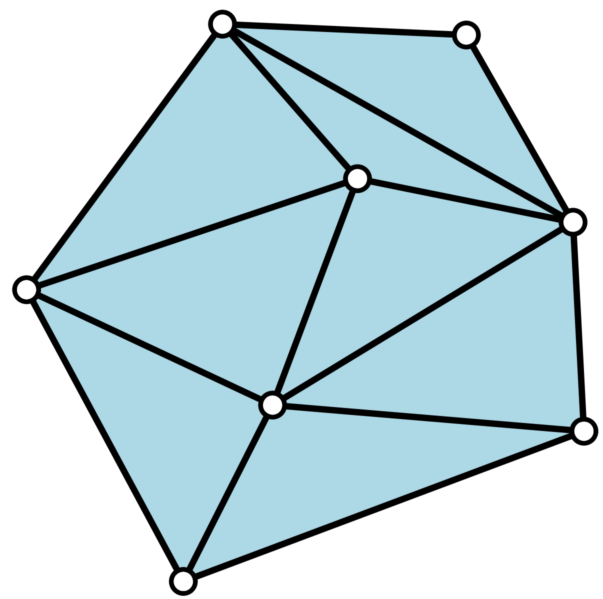
Maximal Plane Straight-Line Graphs

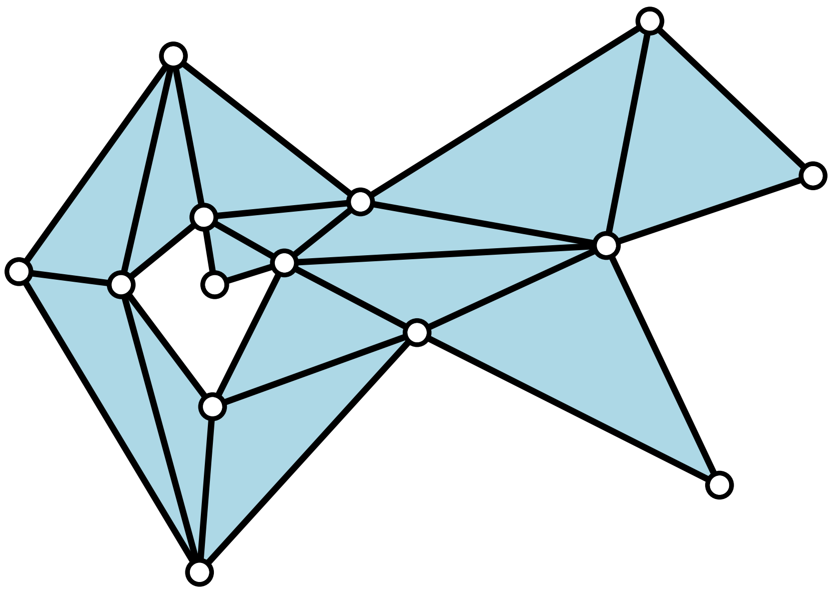
- Given a plane straight-line graph
- We add edges as long as we can.
- The result is a triangulation. **Why?**



Maximal Plane Straight-Line Graphs

- Given a plane straight-line graph
- We add edges as long as we can.
- The result is a triangulation.
- The same can be done for polygons / polygonal regions.



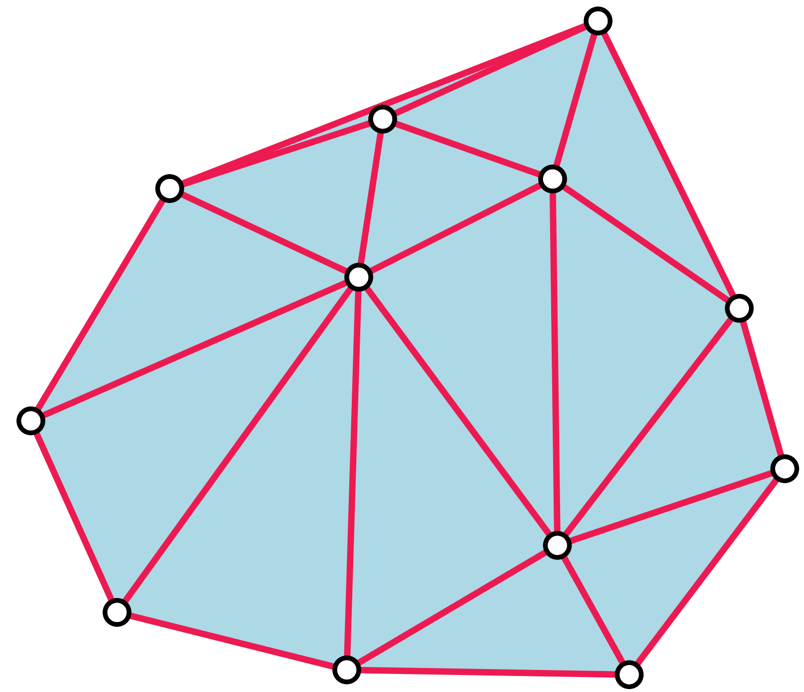


Elements of a Triangulation

A triangulation consists of vertices, edges, and triangles.

Question: How many edges, and triangles can a triangulation of a pointset S with n points have?

Different numbers possible for the same point set?



n=12, 25 edges, 14 triangles

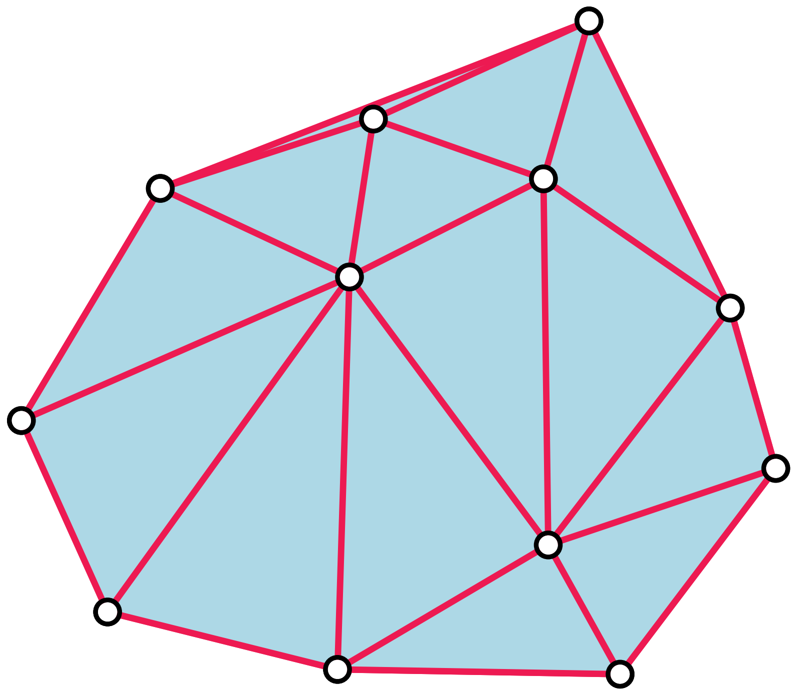
Elements of a Triangulation

A triangulation consists of vertices, edges, and triangles.

Question: How many edges, and triangles can a triangulation of a pointset S with n points have?

$\Rightarrow$ If S has h extreme points:

- $e = 3n - 3 - h$ edges
- $t = 2n - 2 - h$ triangles



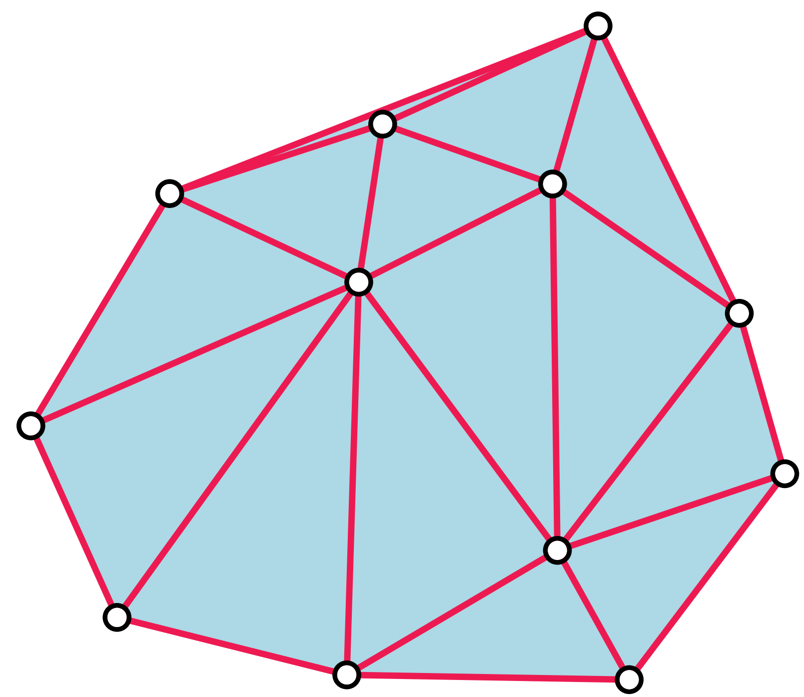
n=12, 25 edges, 14 triangles

Elements of a Triangulation

A triangulation consists of vertices, edges, and triangles.

Question: How many edges, and triangles can a triangulation of a pointset S with n points have?

- Every triangulation of S has the **same** number of edges and triangles
- The number of edges and triangles depends on the number of **extreme points** of S (vertices of the convex hull of S)



n=12, 25 edges, 14 triangles

Elements of a Triangulation

A triangulation consists of vertices, edges, and triangles.

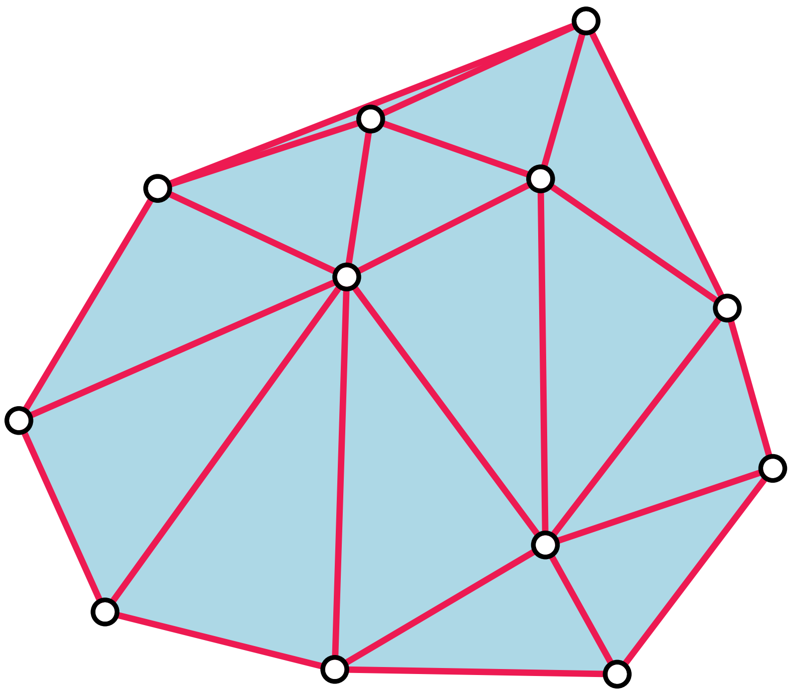
Question: How many edges, and triangles can a triangulation of a pointset S with n points have?

$\Rightarrow$ If S has h extreme points:

- $e = 3n - 3 - h$ edges
- $t = 2n - 2 - h$ triangles

Question: What about polygons with n vertices?

- $e = 2n - 3$ edges
- $t = n - 2$ triangles



Note: The number of edges and triangles is **linear** in the number of vertices.

Some Applications

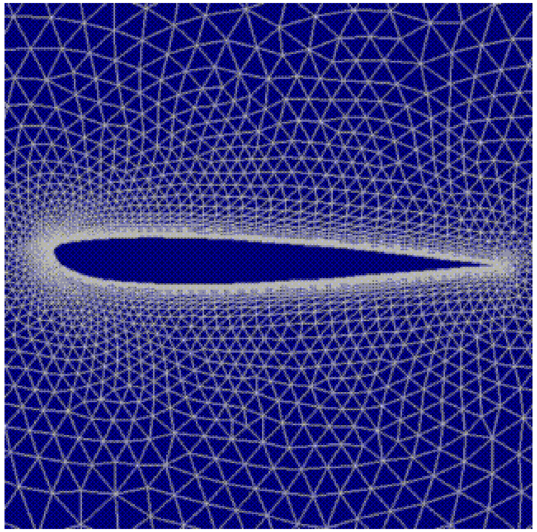
What can triangulations be used for?

Triangulations in Graphics/GIS

- The height of a terrain is measured at certain points.
- The points in a triangulation are elevated.
- An approximation of the terrain is obtained.
- Surfaces in 3D are modeled using meshes.
- Every element is a triangle
  - allows fast processing by graphics hardware
  - no checks necessary

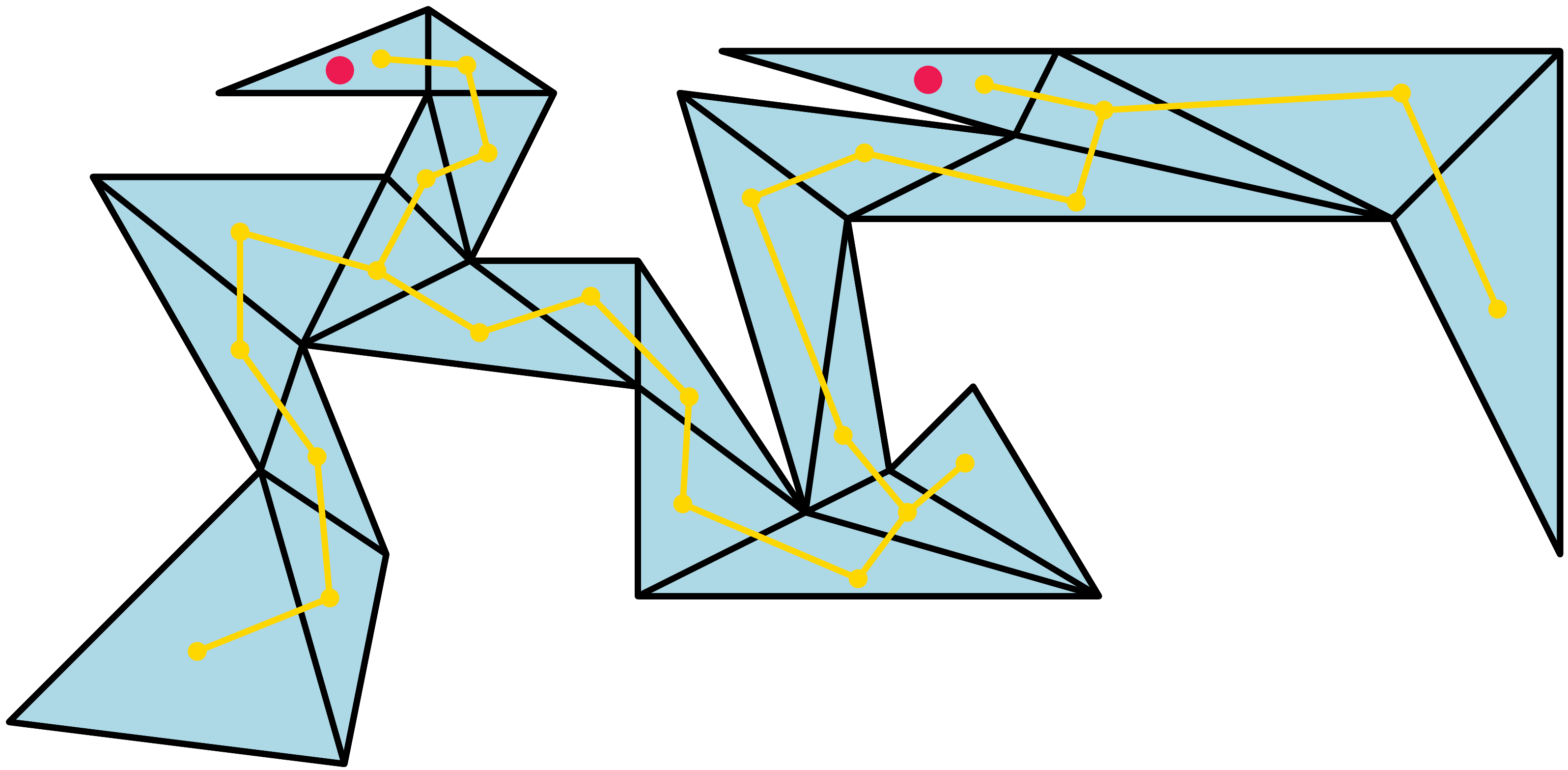
Triangulations in FEM

- Technique to numerically solve partial differential equations or integral equations.
- Approximated by a triangular mesh.
- Finite number of elements (usually triangles) give piecewise linear function.
- Numerically stable approximations need good meshes.
- Used, e.g., to analyze heat flow or forces in materials, or simulating fluid flows.



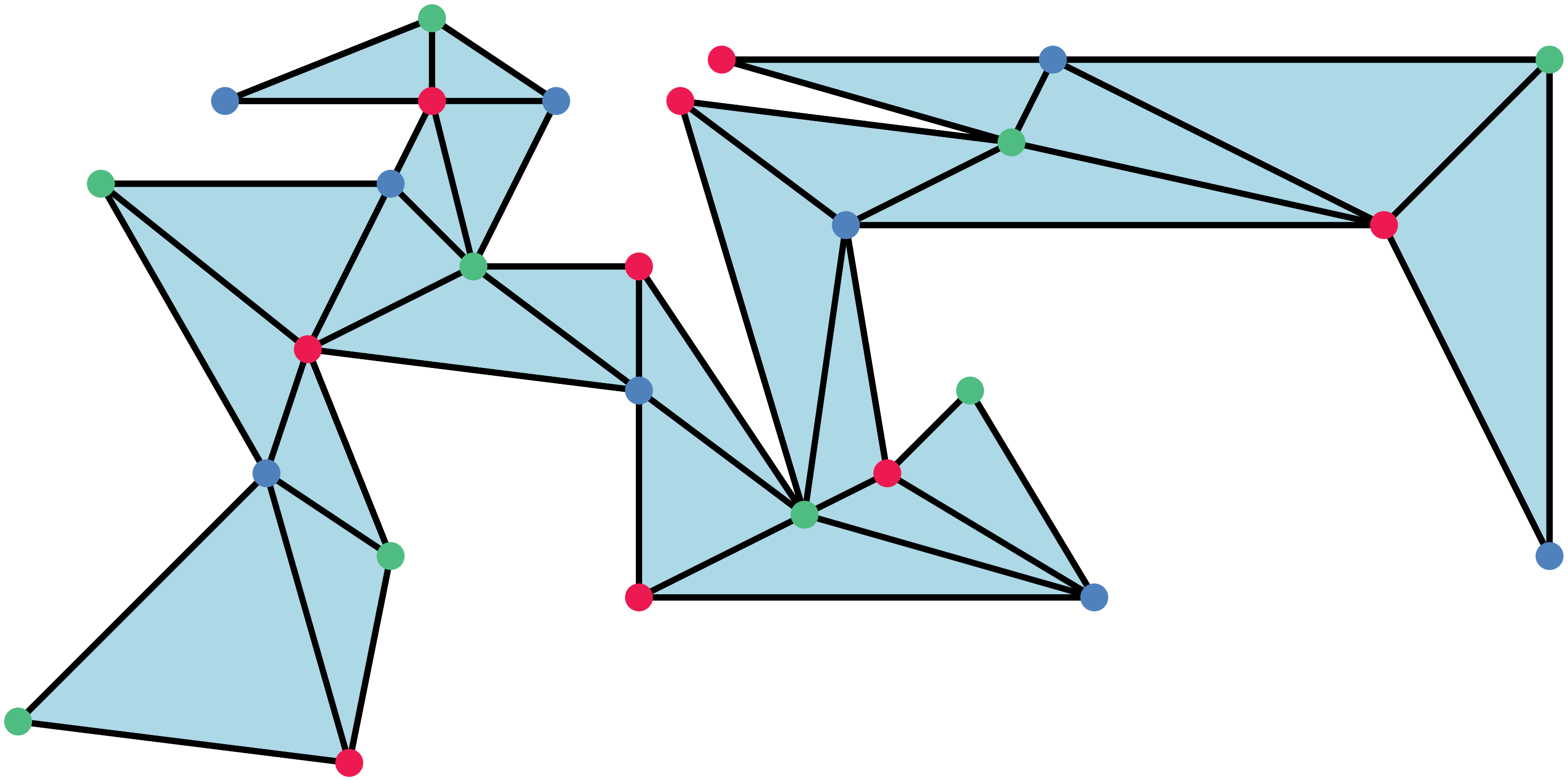
Triangulations in Algorithms

- Example: Shortest Path inside a polygon.
- Also used in proving time bounds.
- Example: Guarding



Triangulations in Combinatorics

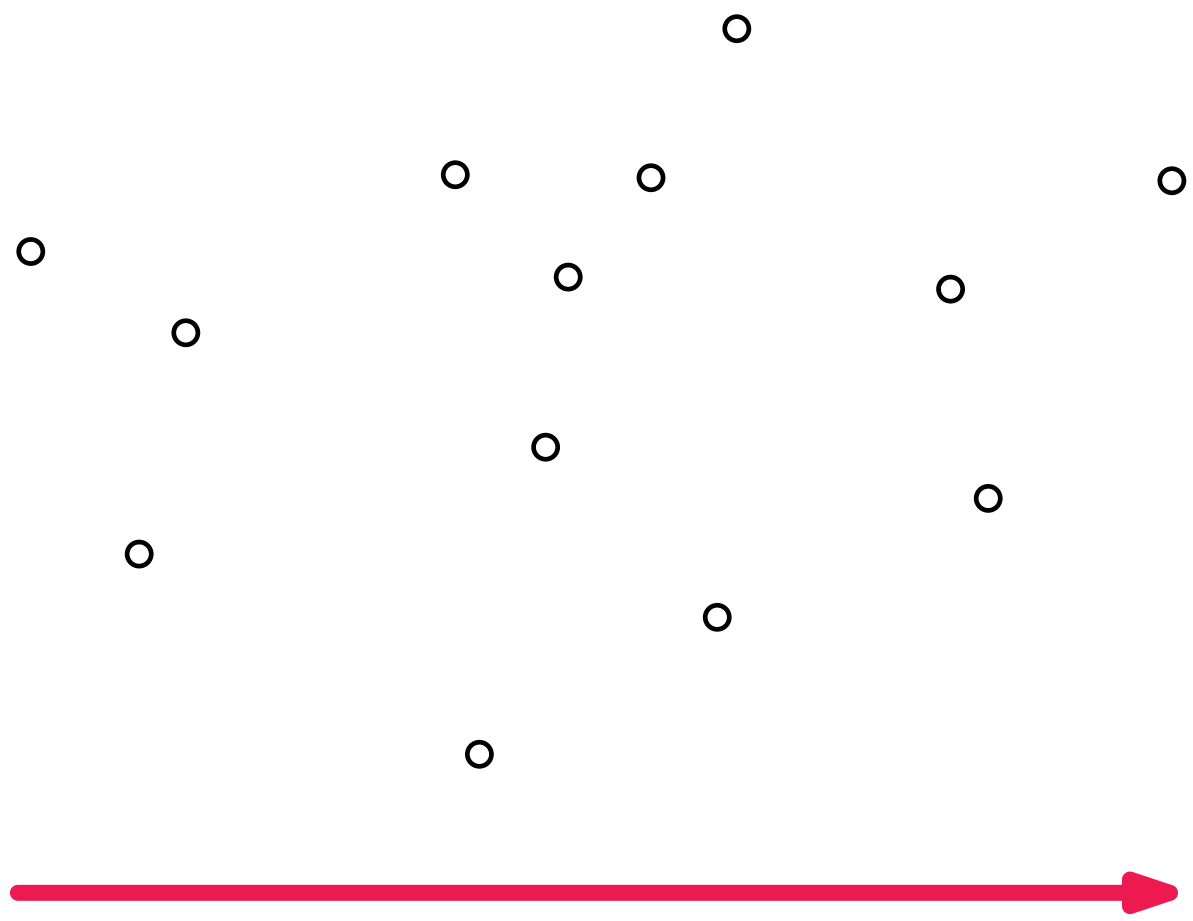
Example: Guarding



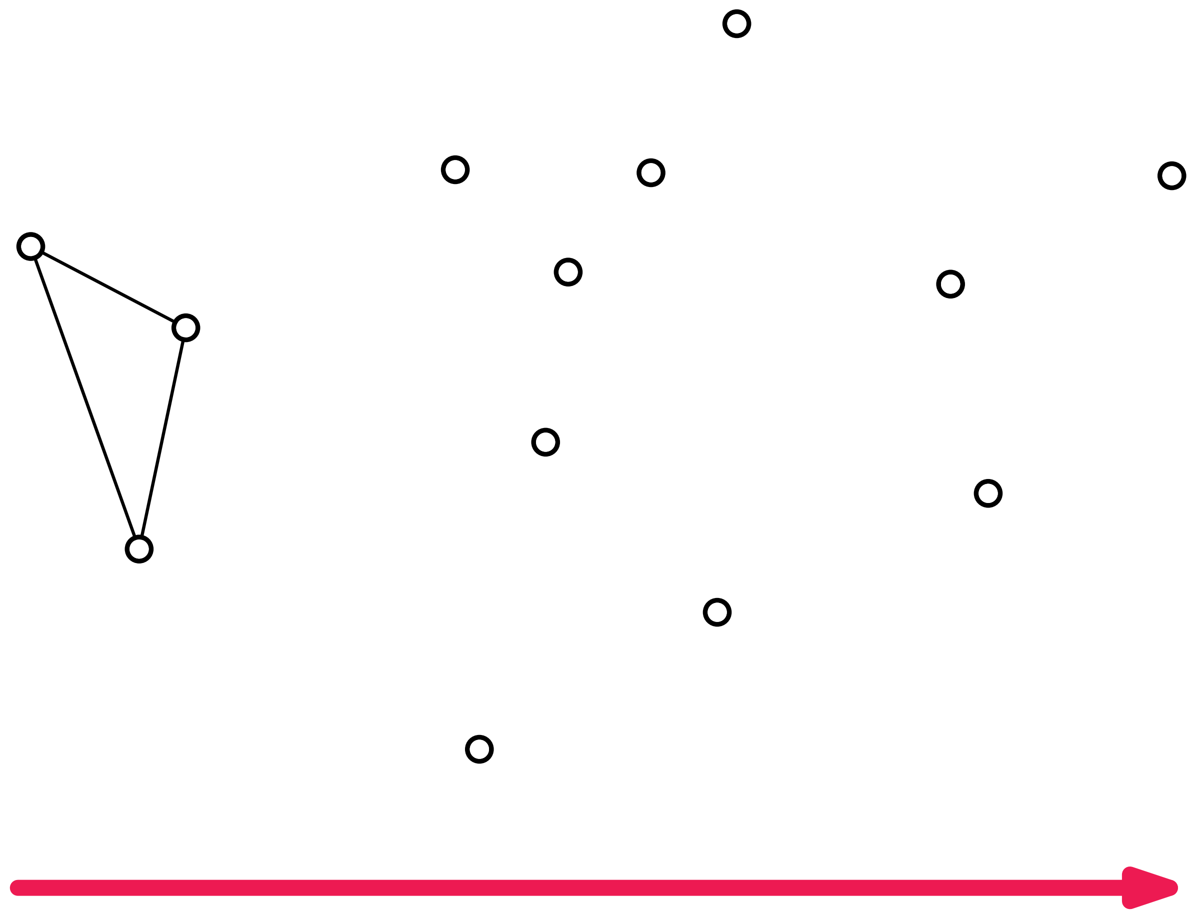
Construction of a Triangulation

How efficiently can we construct a triangulation?

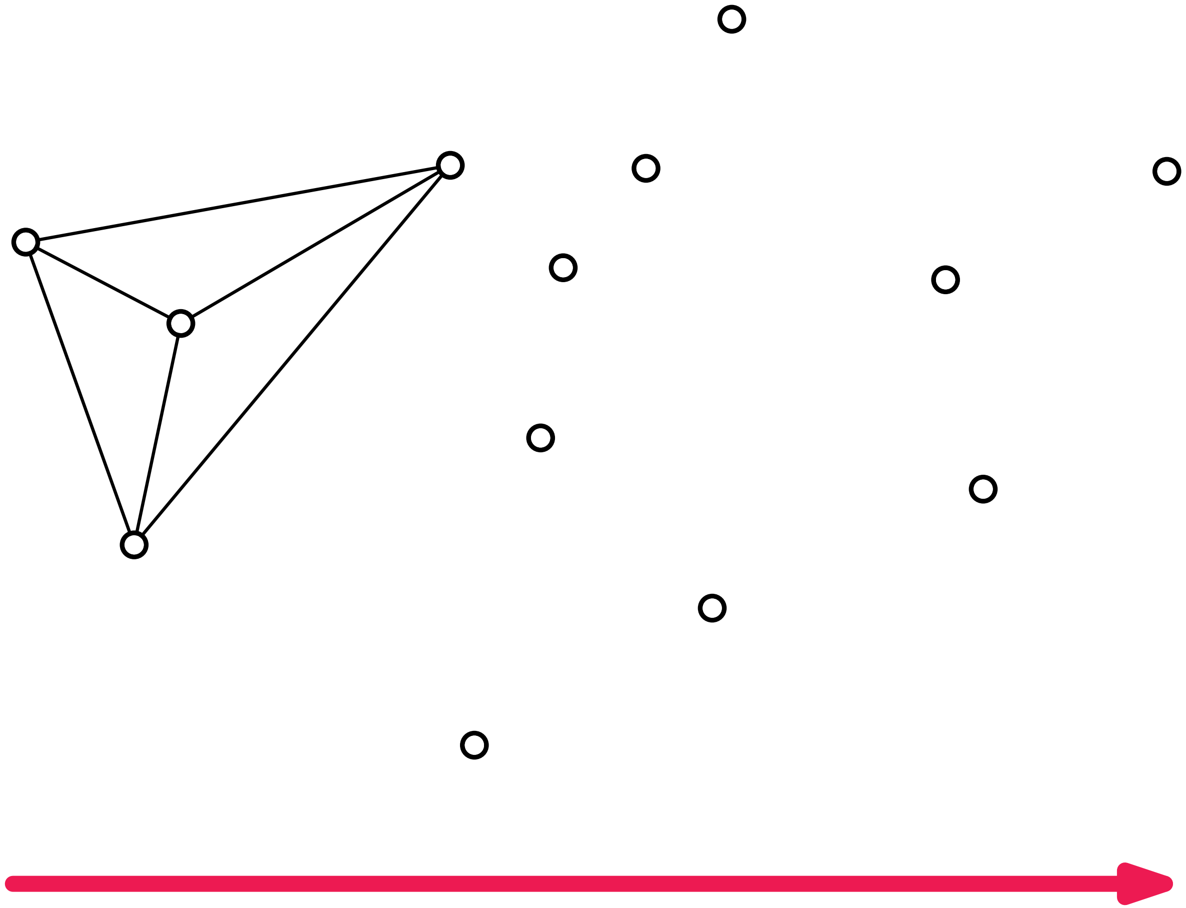
A Canonical Triangulation



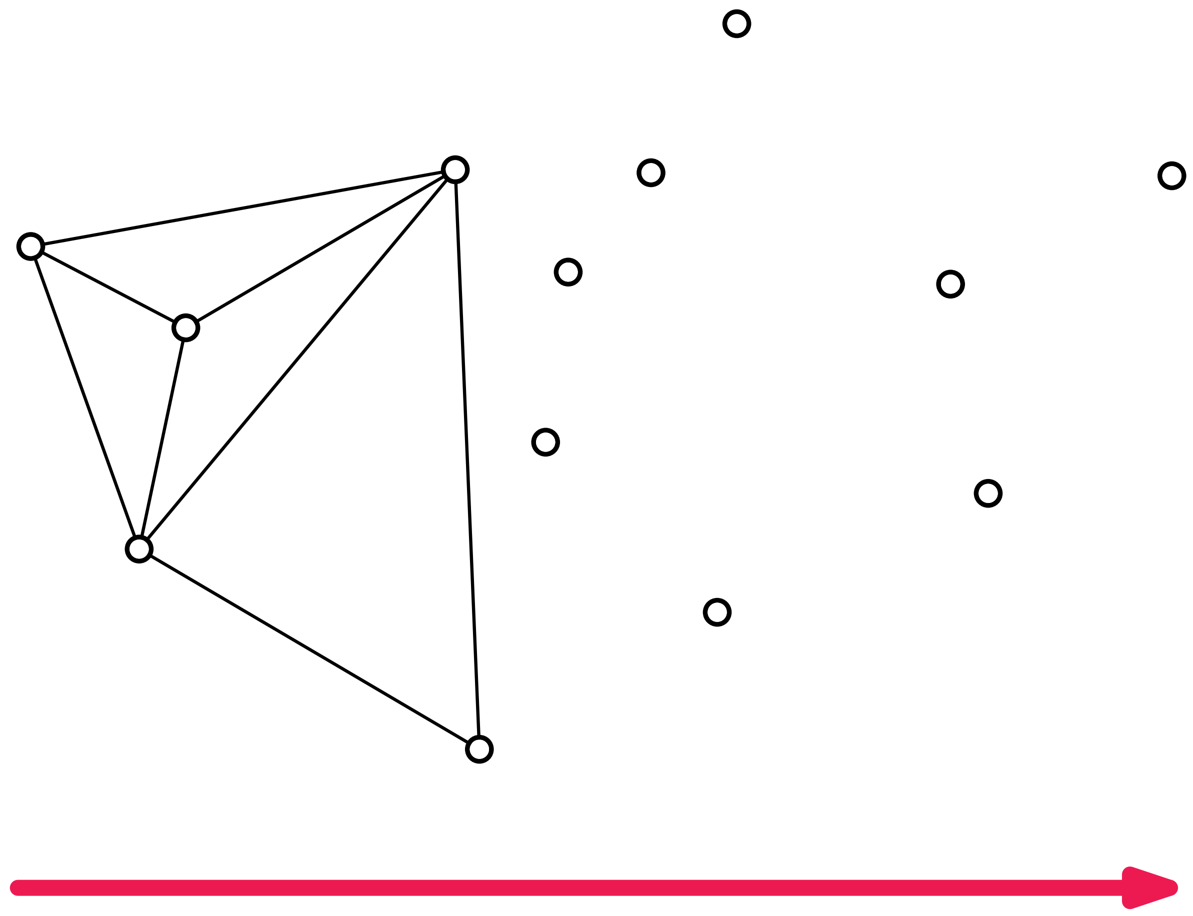
A Canonical Triangulation



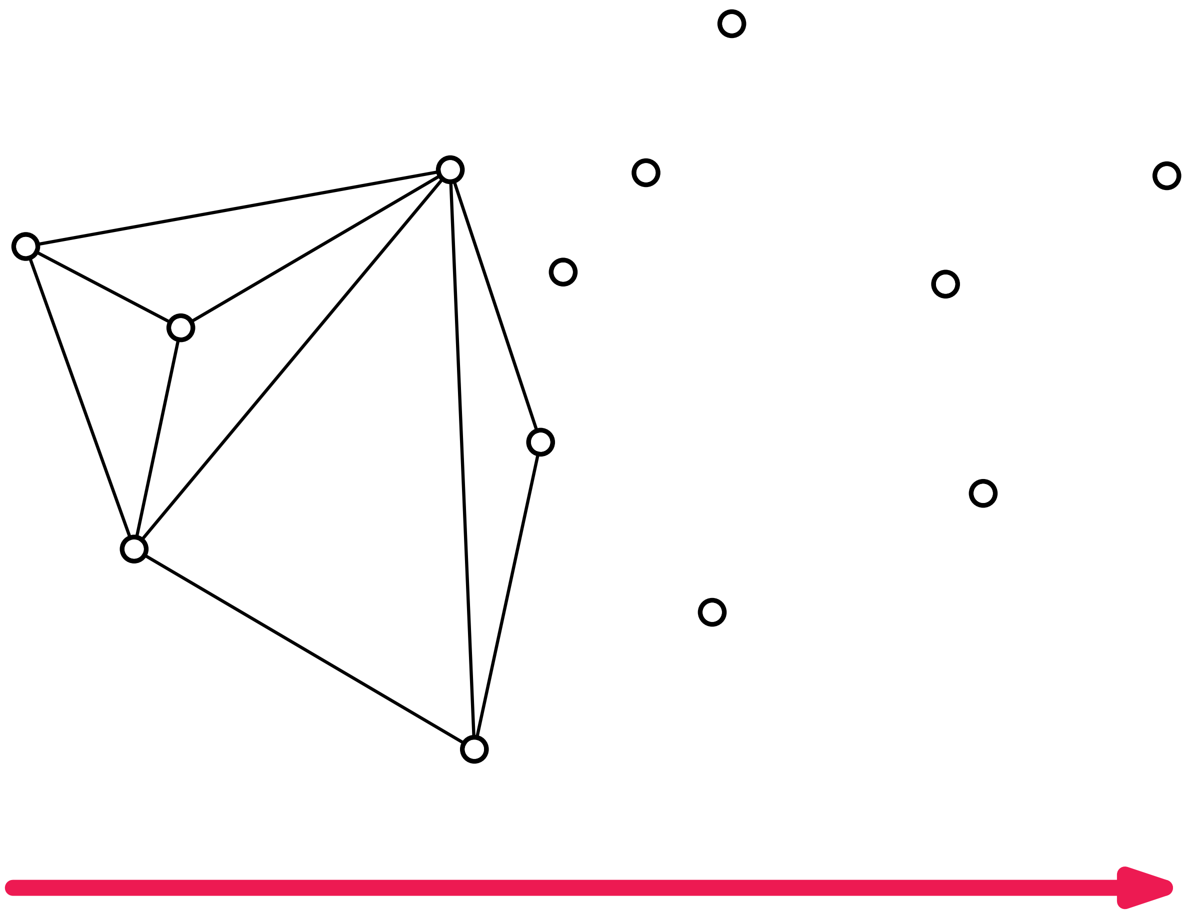
A Canonical Triangulation



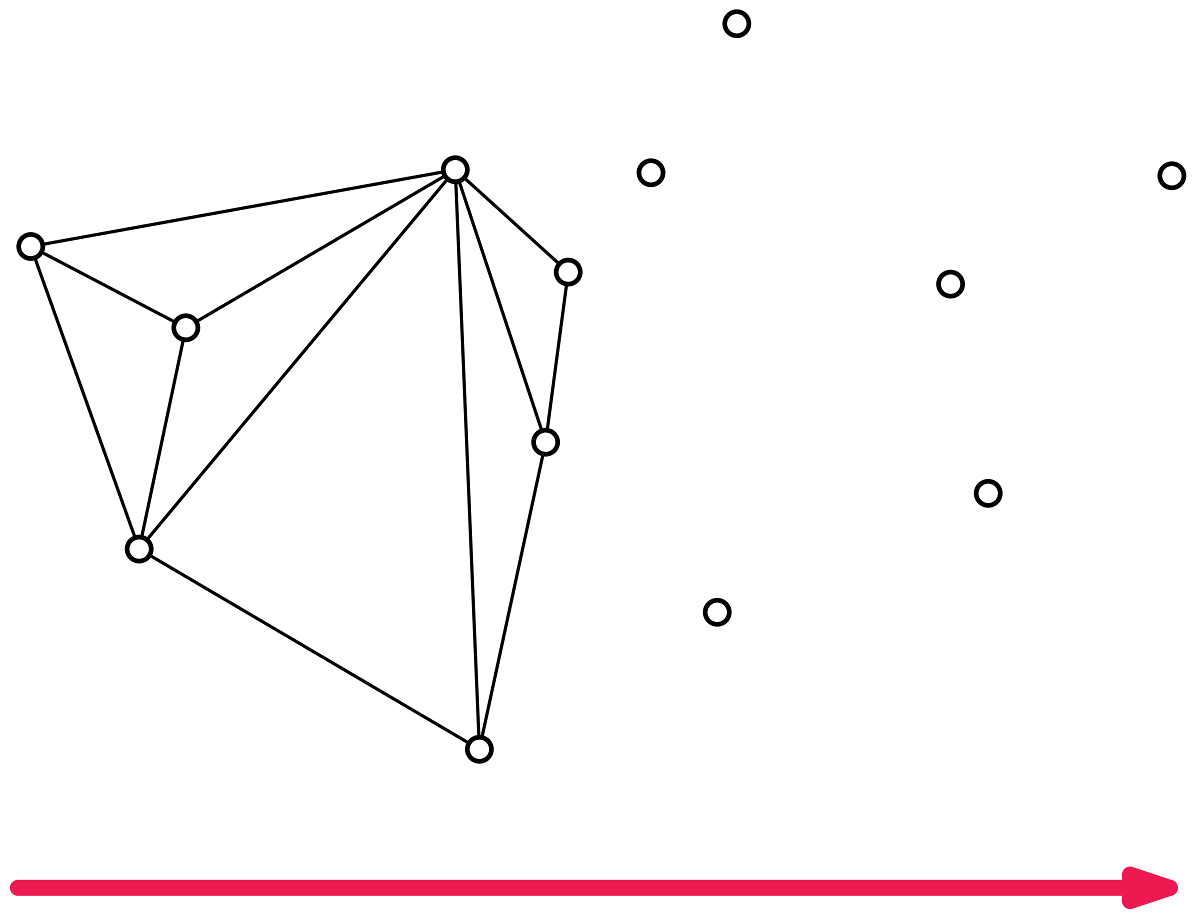
A Canonical Triangulation



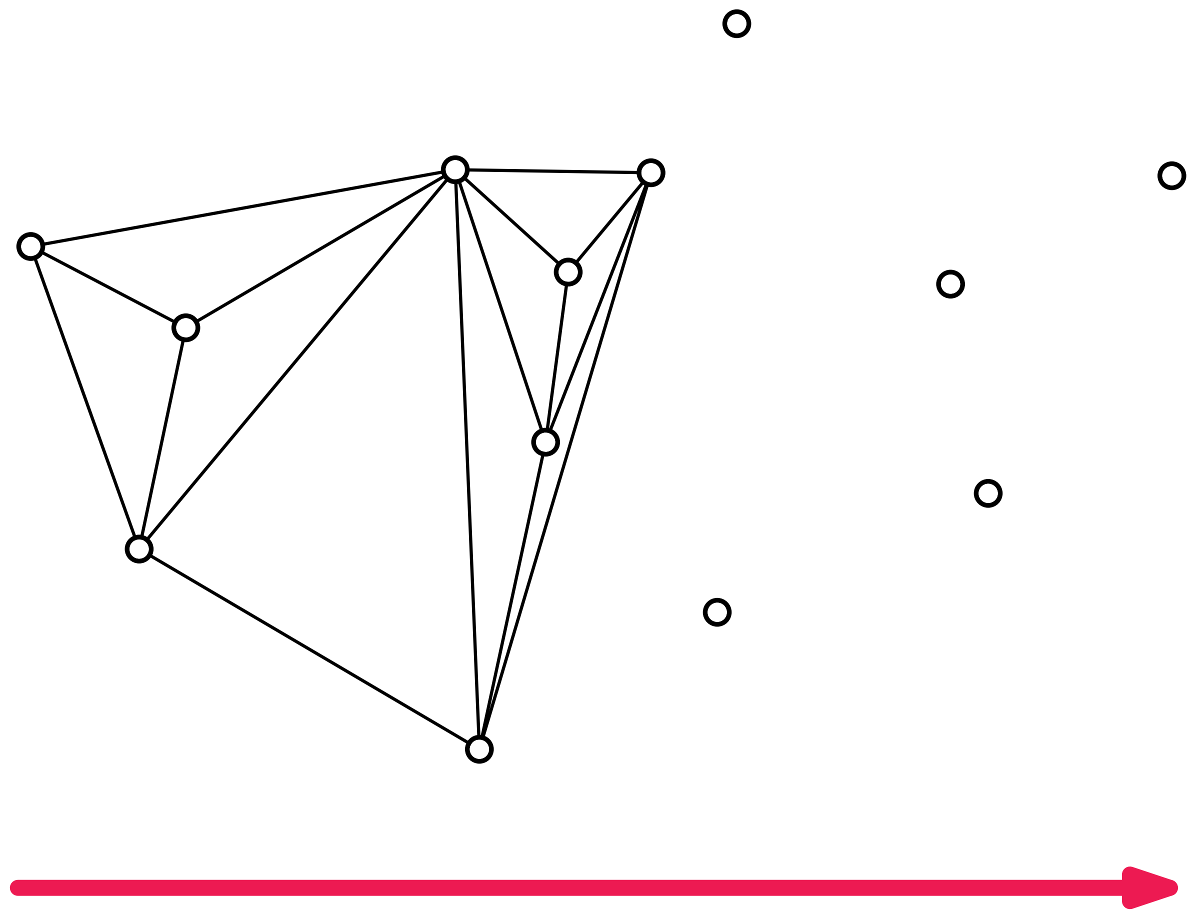
A Canonical Triangulation



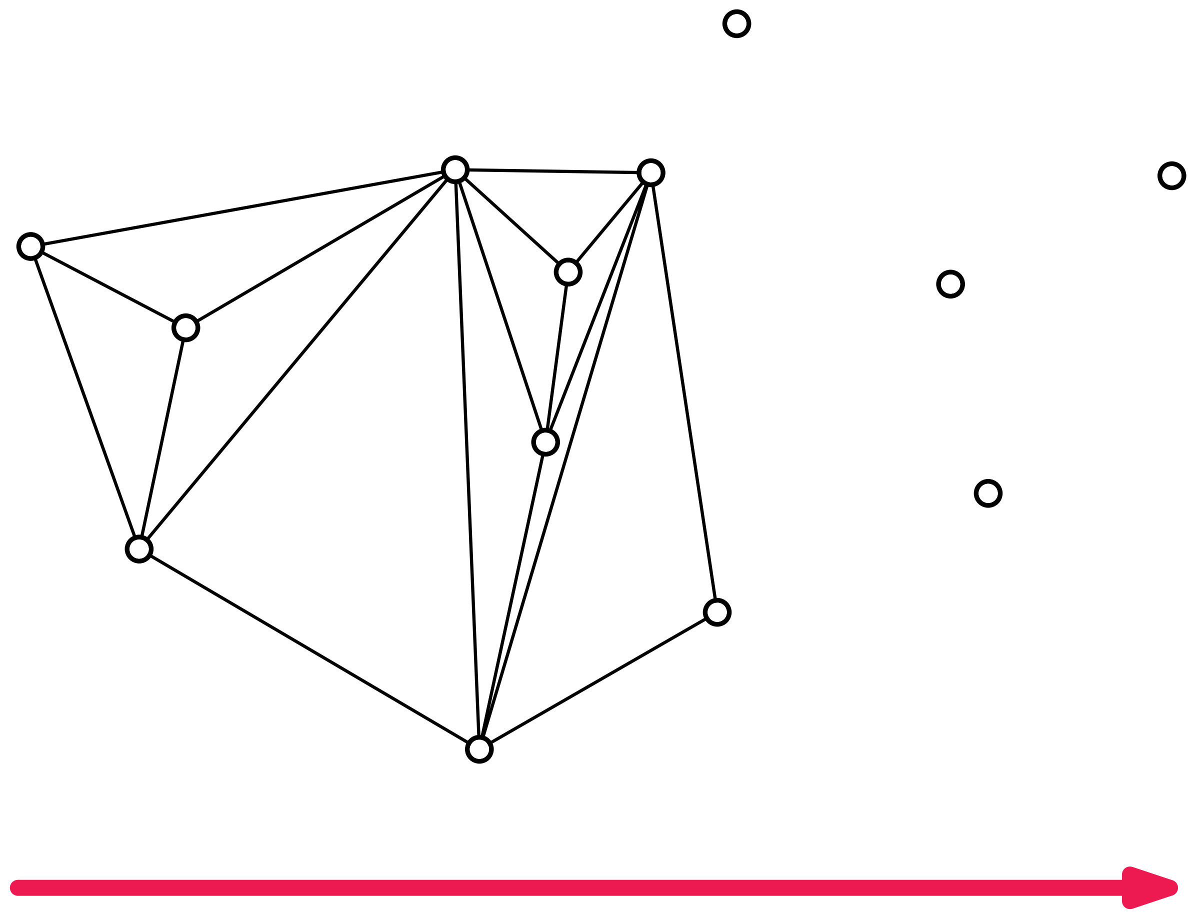
A Canonical Triangulation



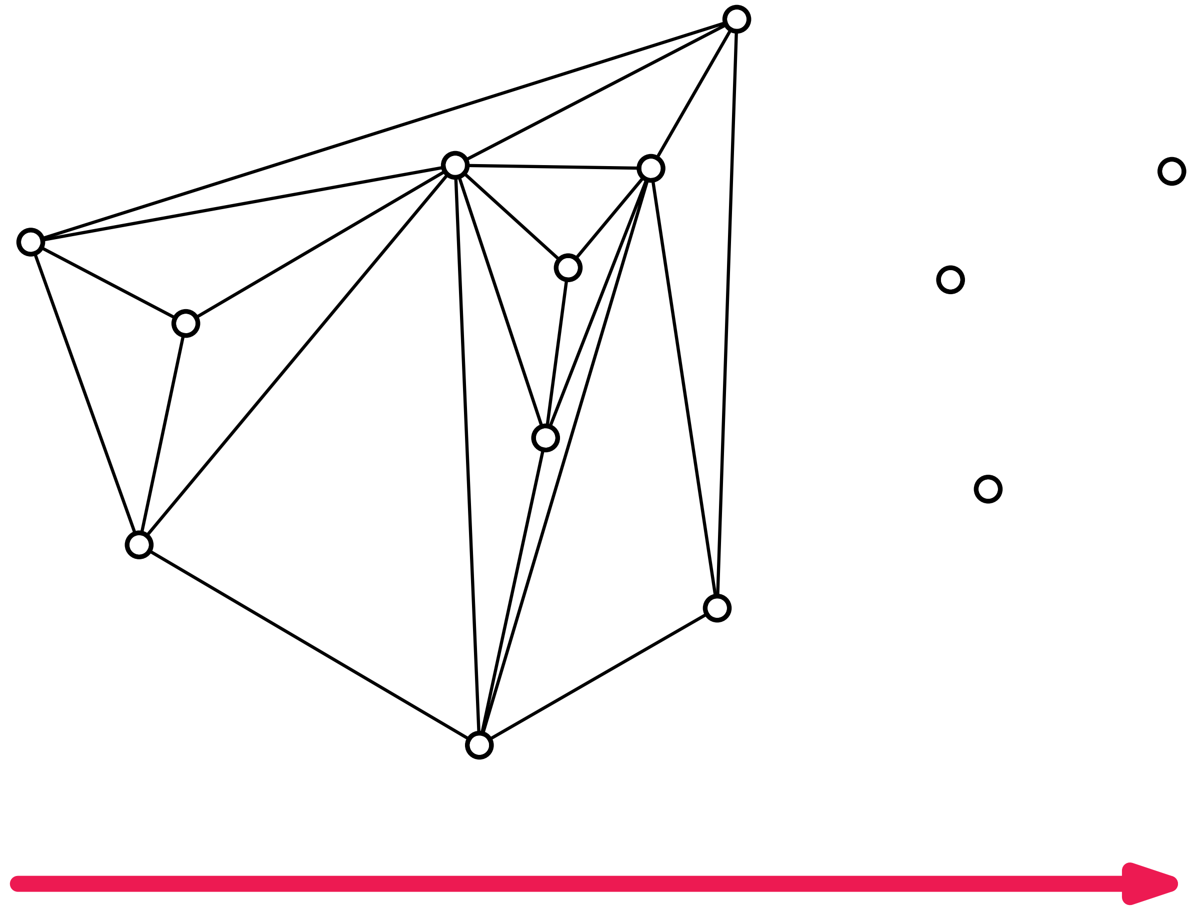
A Canonical Triangulation



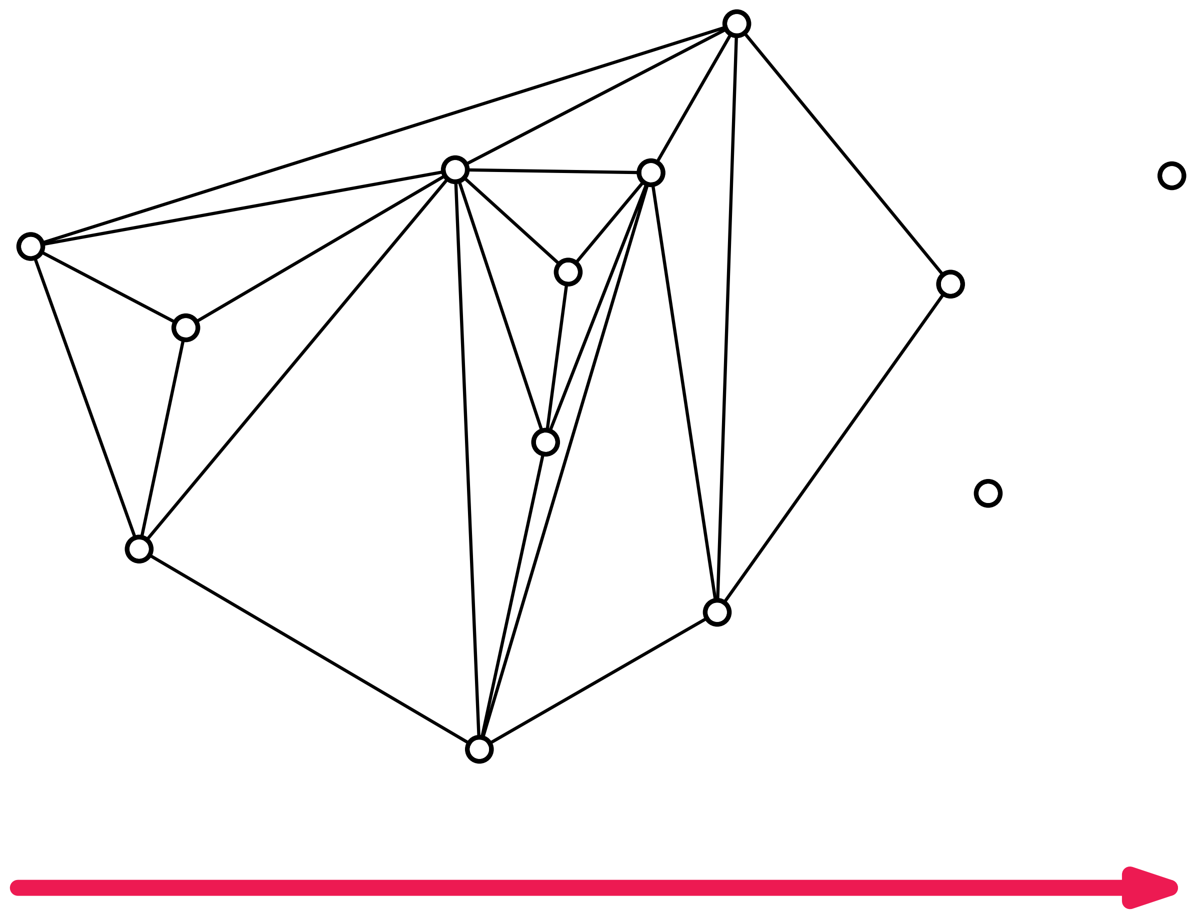
A Canonical Triangulation



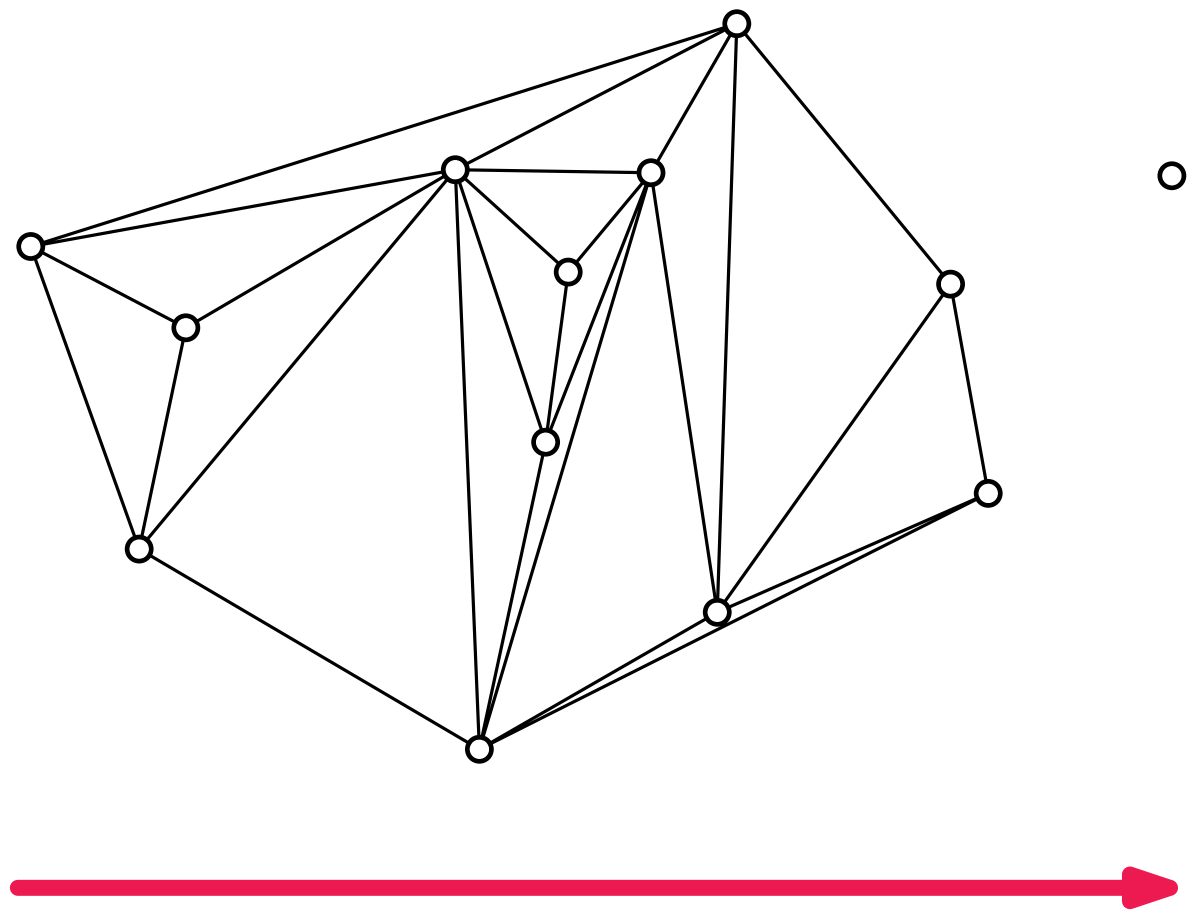
A Canonical Triangulation



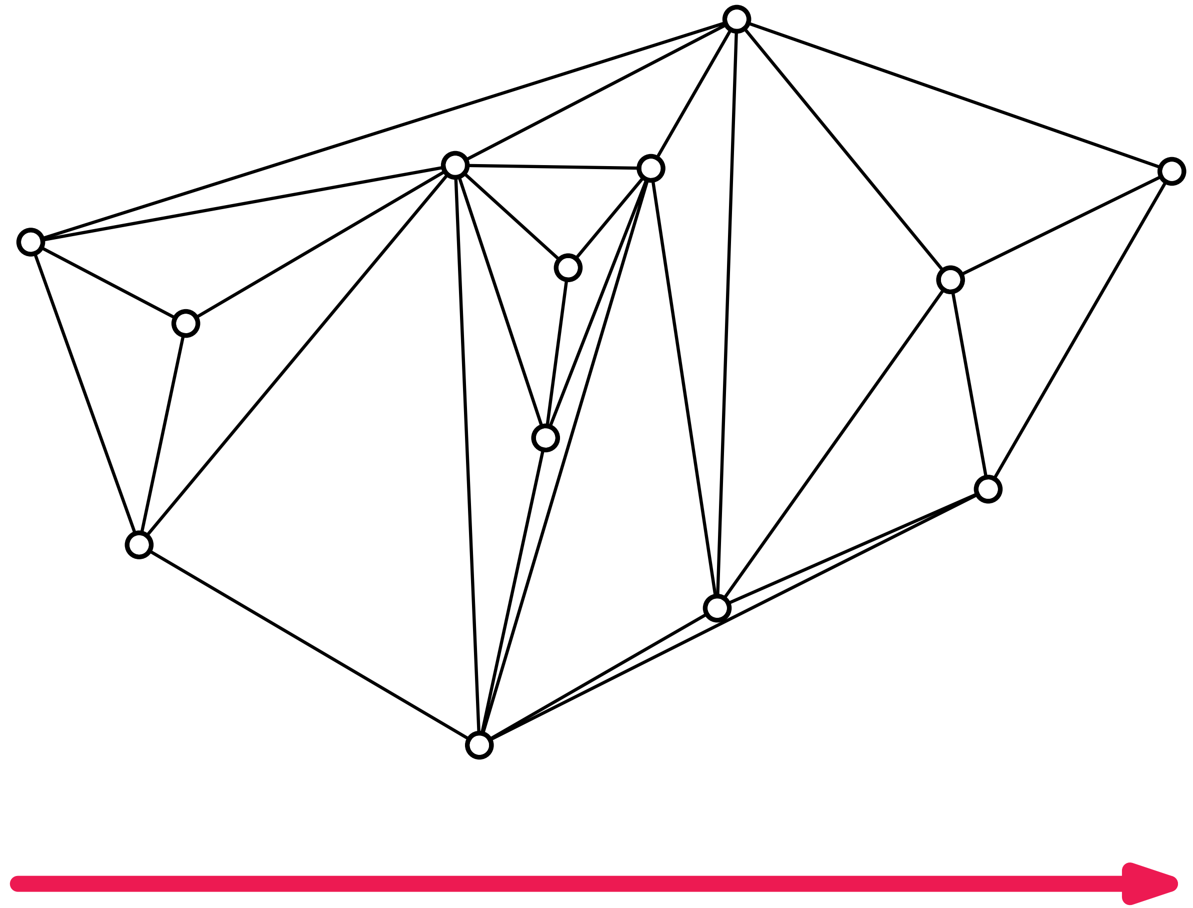
A Canonical Triangulation



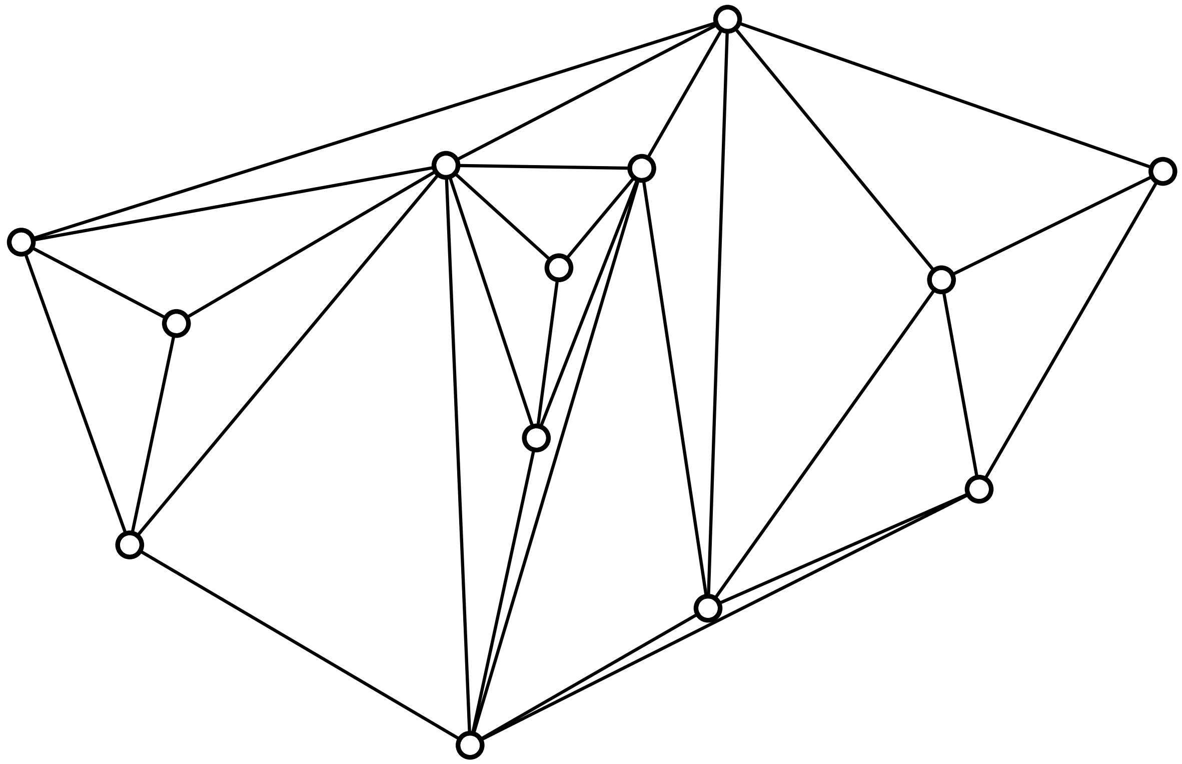
A Canonical Triangulation



A Canonical Triangulation



A Canonical Triangulation



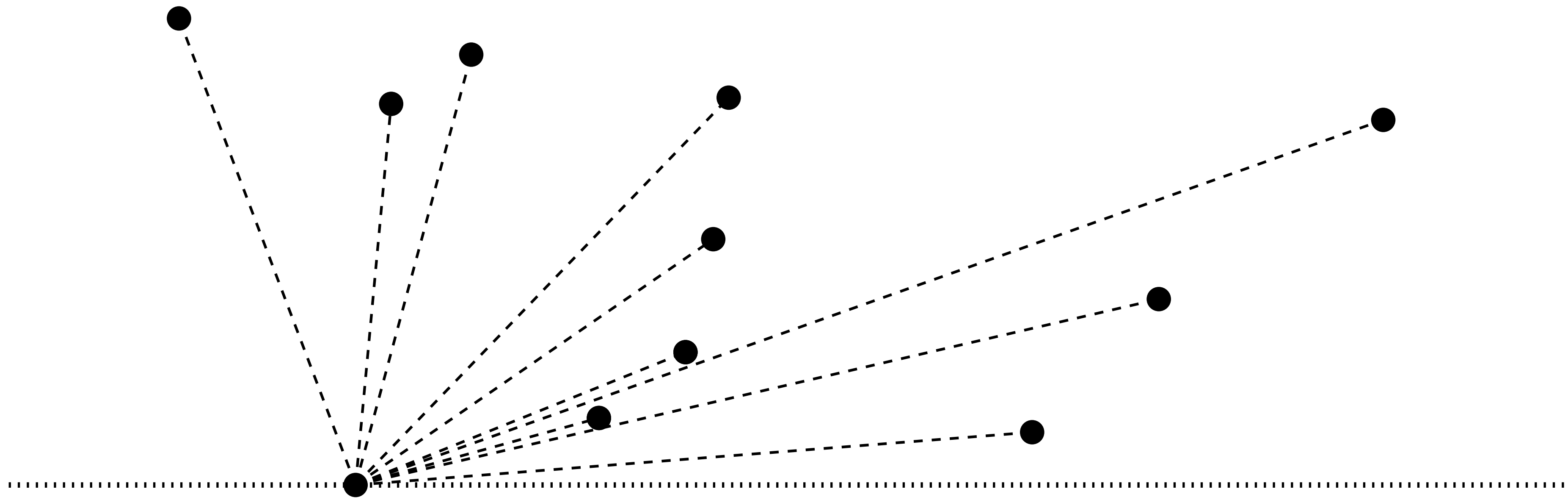
Questions:

Details? Correctness? Time & Space?

Other possible constructions?

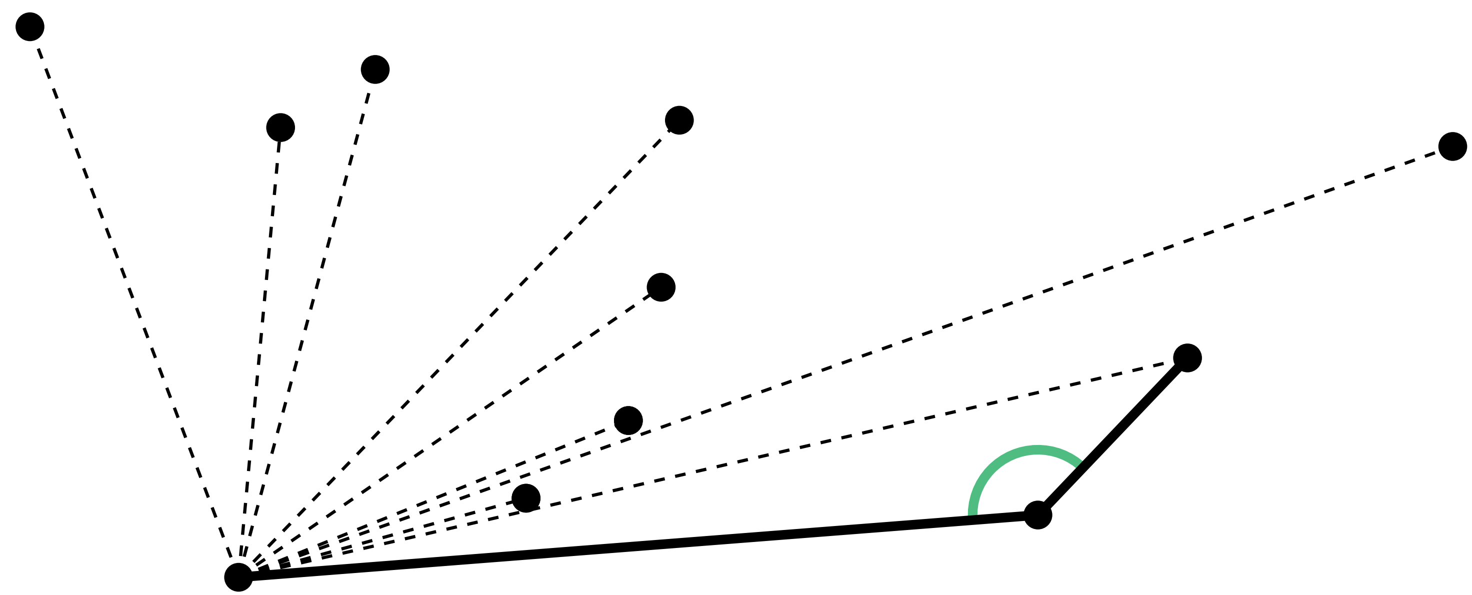
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



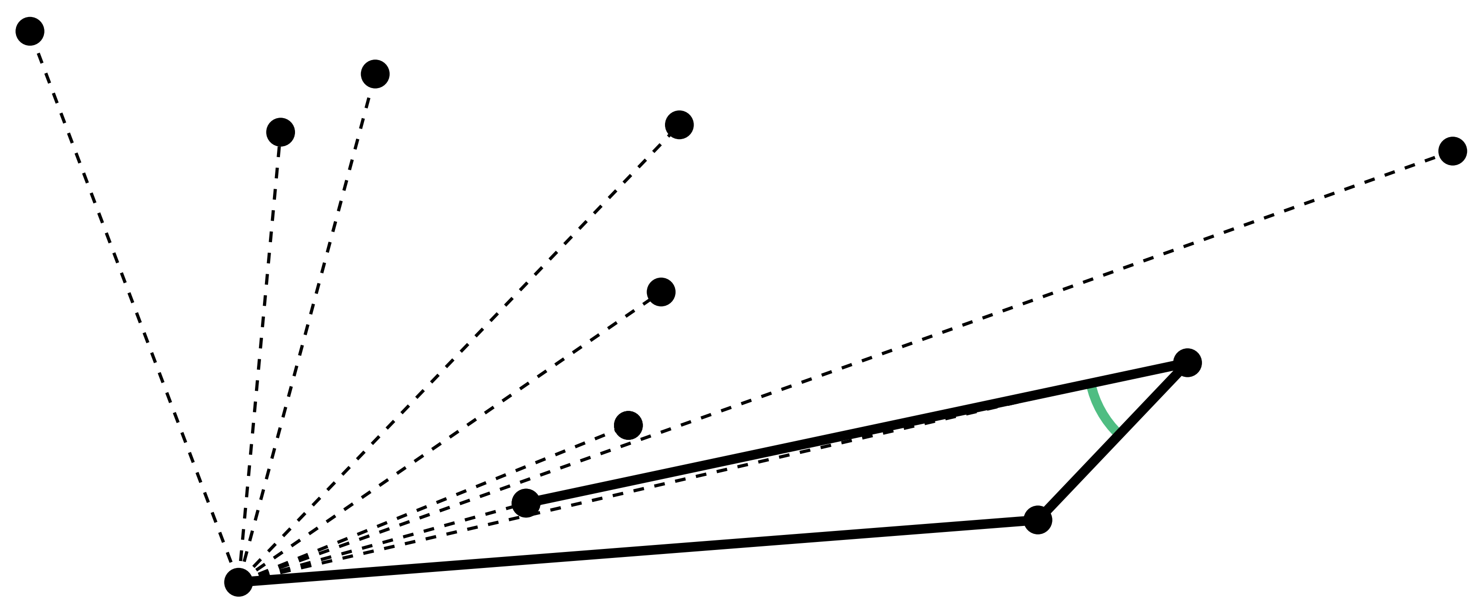
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



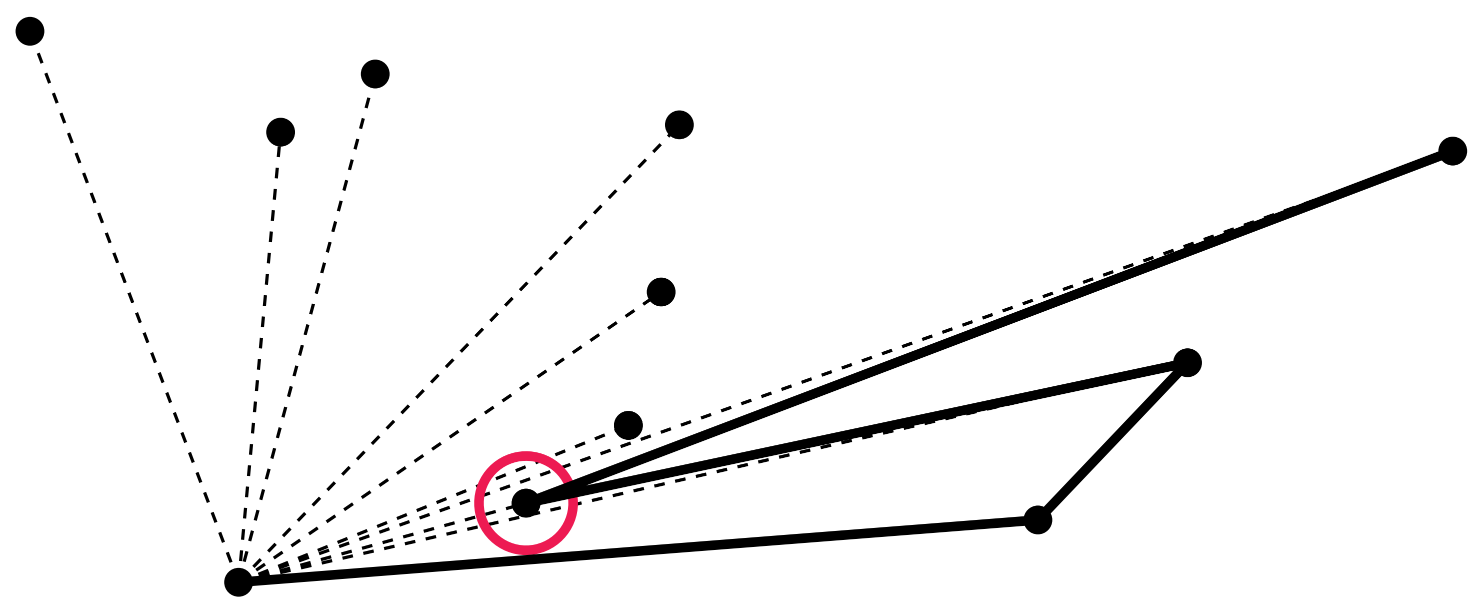
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



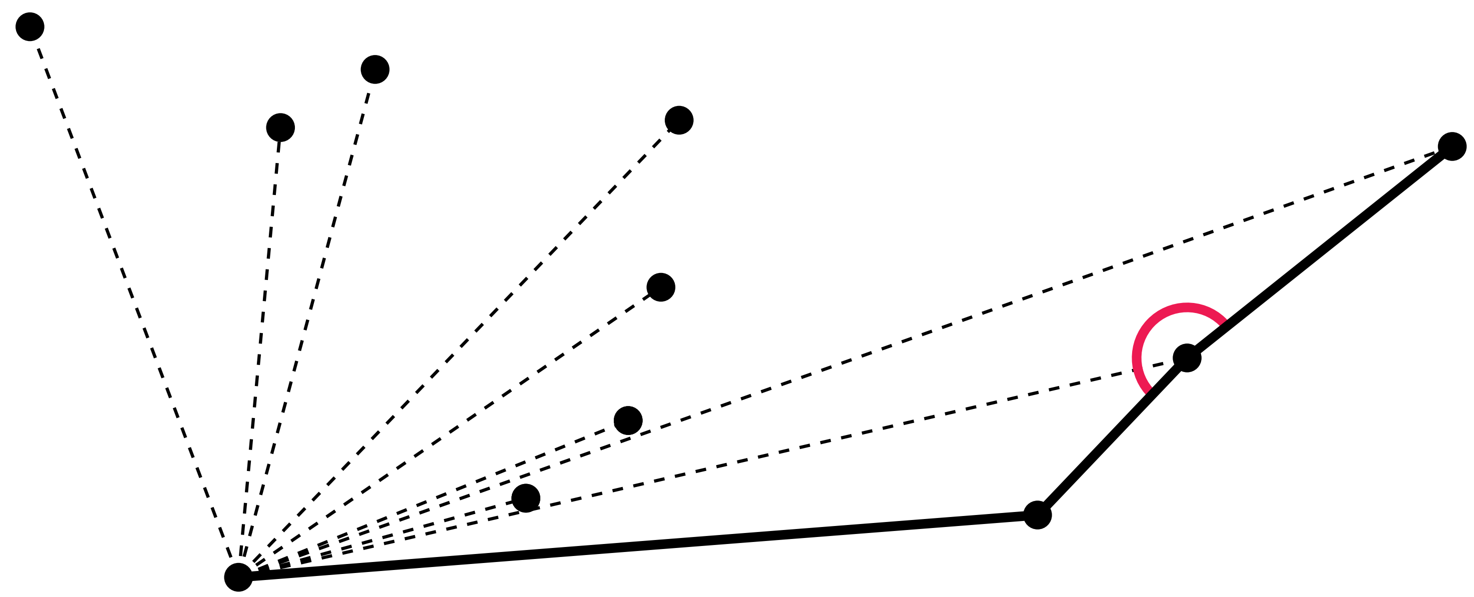
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



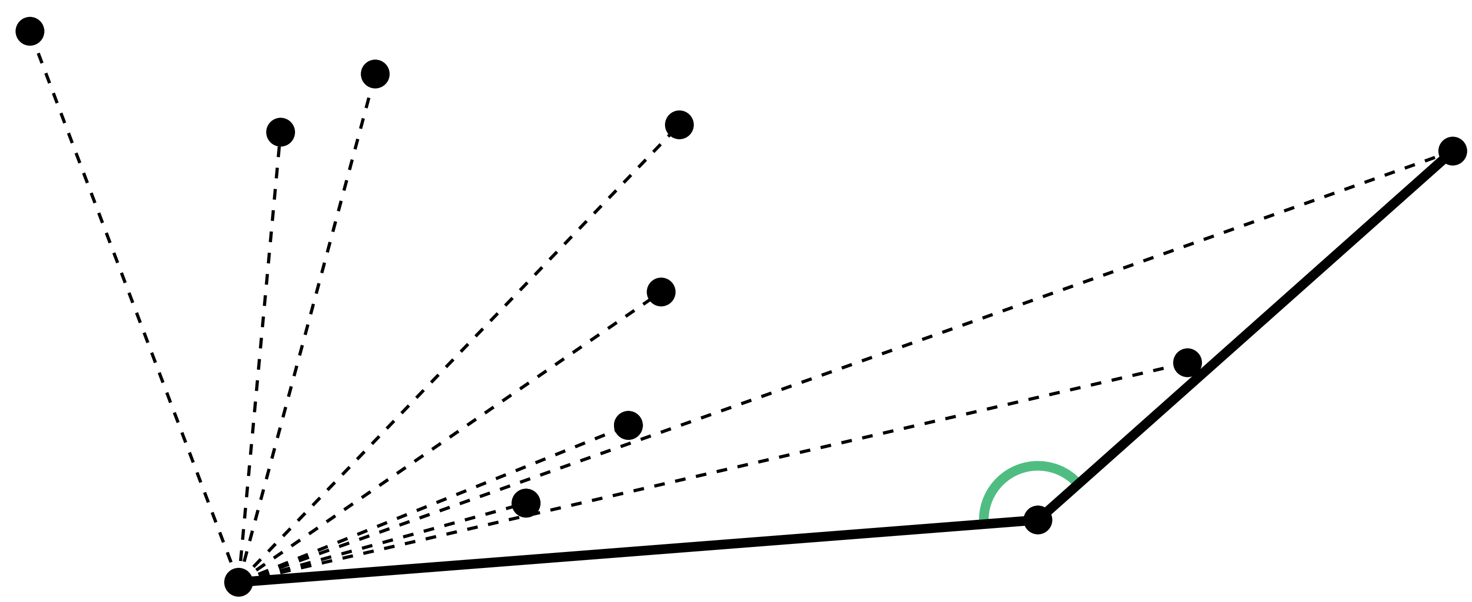
Recall: Graham Scan

- Create hull by “Successive Local Repair”
- sequence sorted around extreme point: remove points not making a left turn



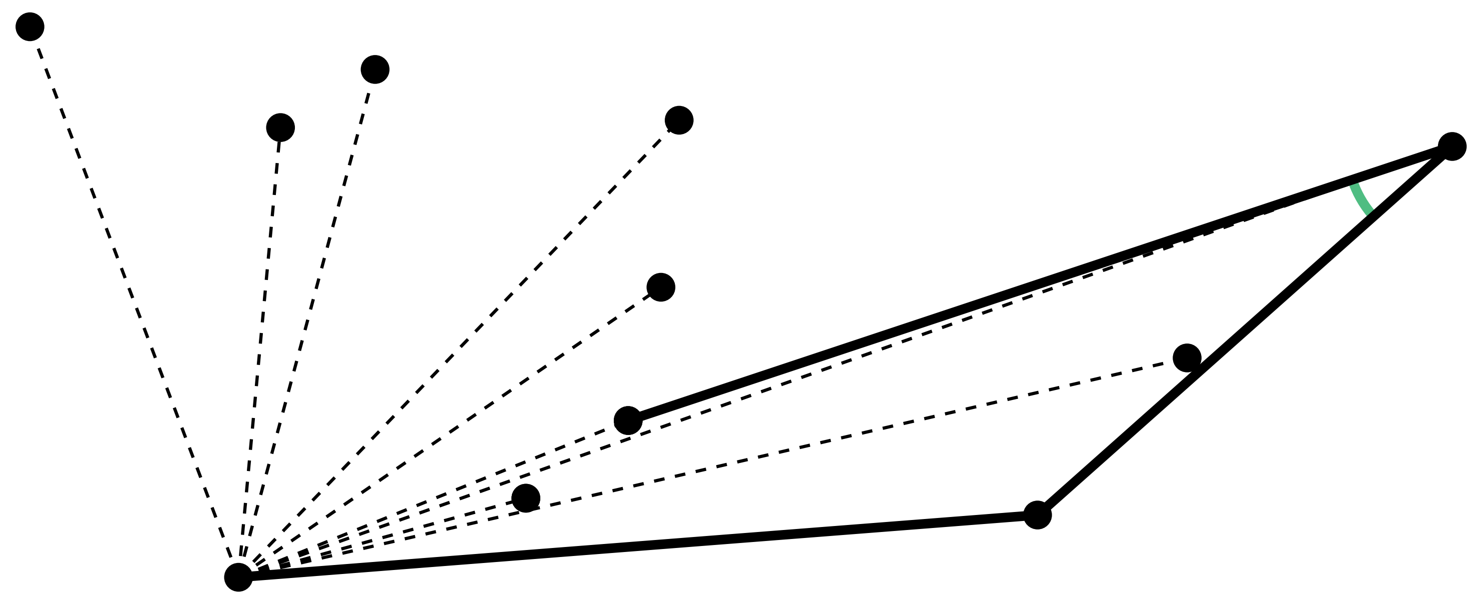
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



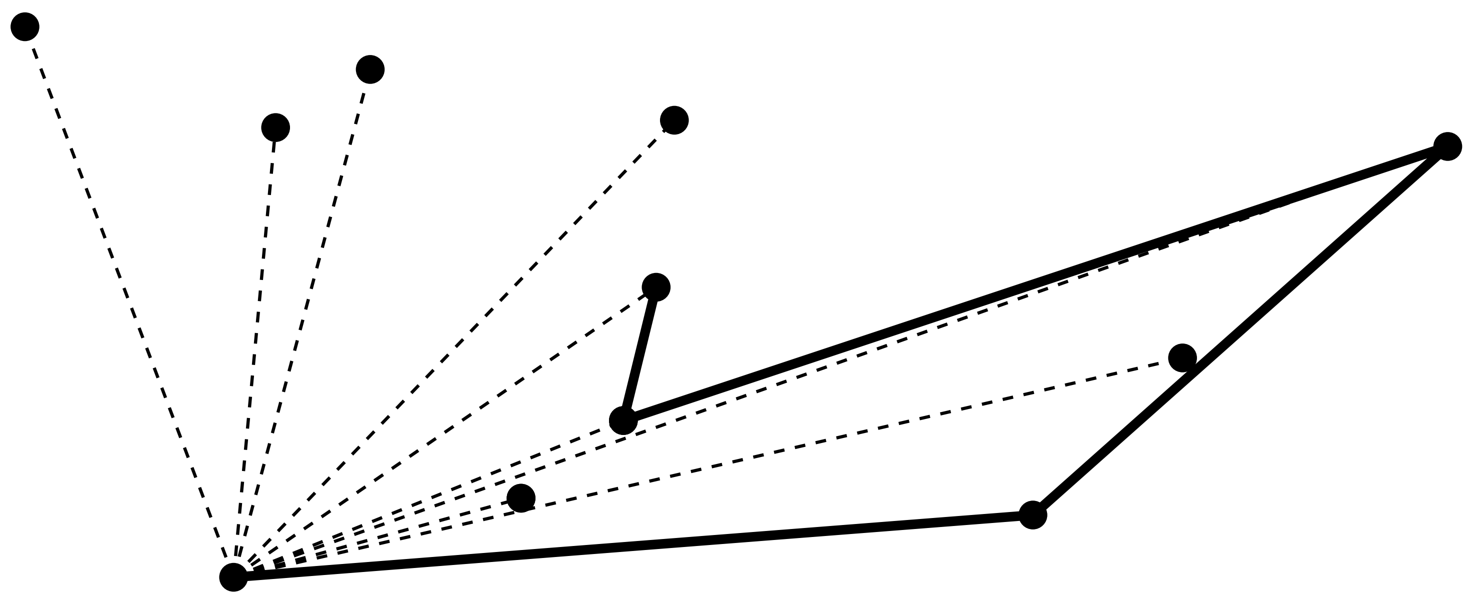
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



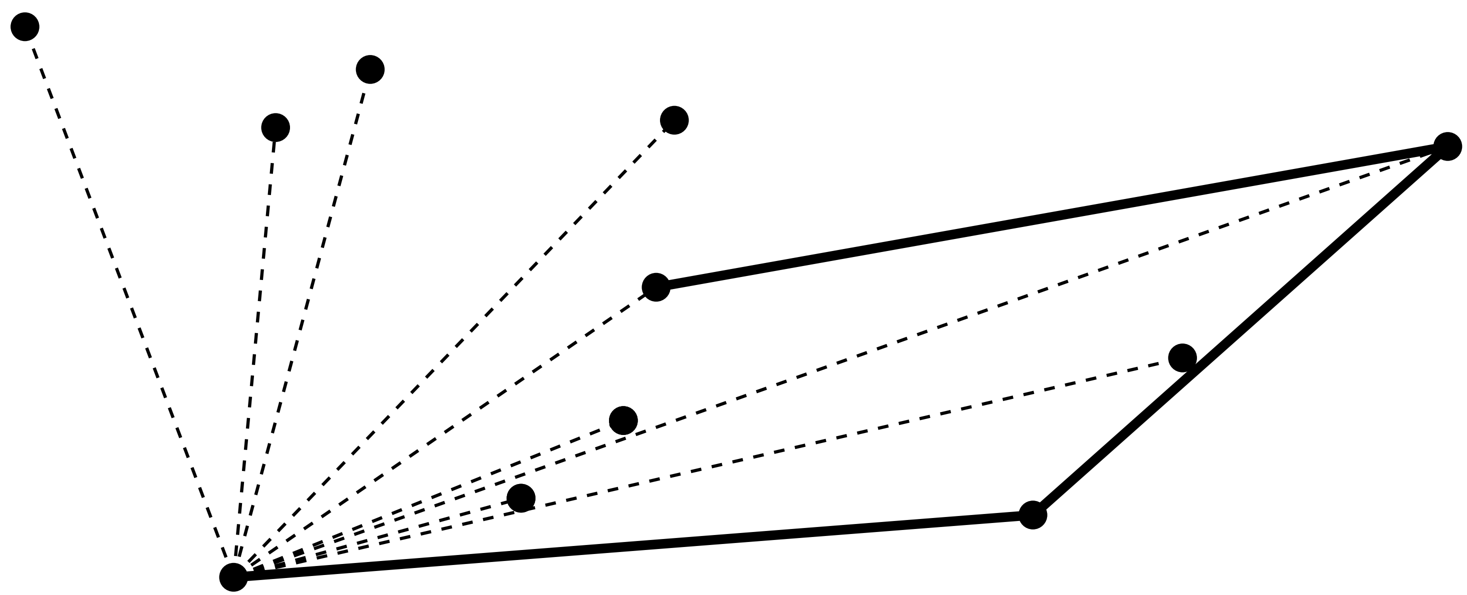
Recall: Graham Scan

- Create hull by “Successive Local Repair”
- sequence sorted around extreme point: remove points not making a left turn



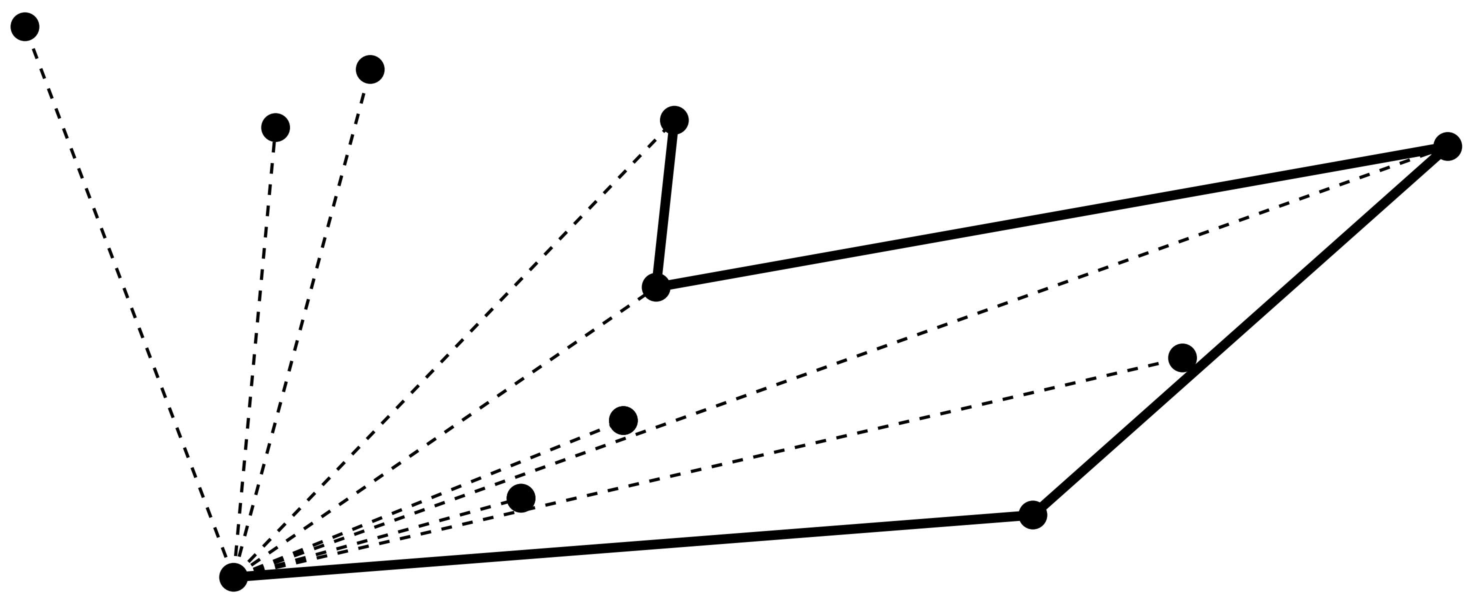
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



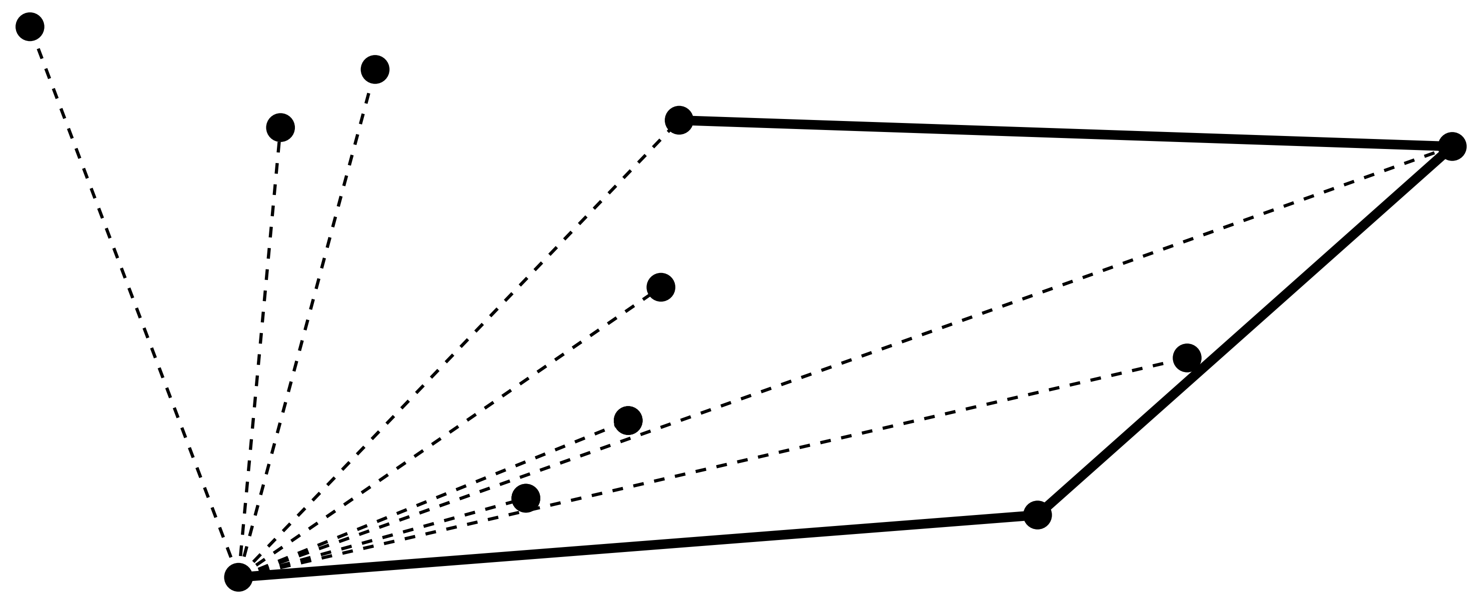
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



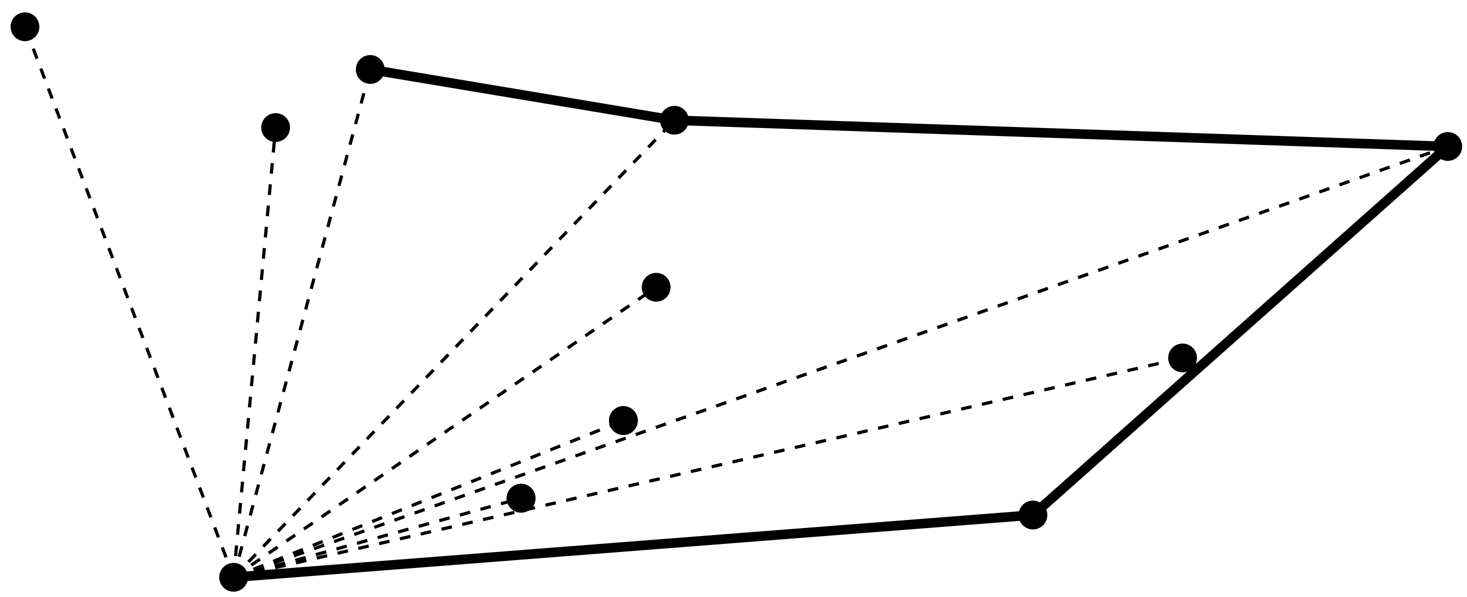
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



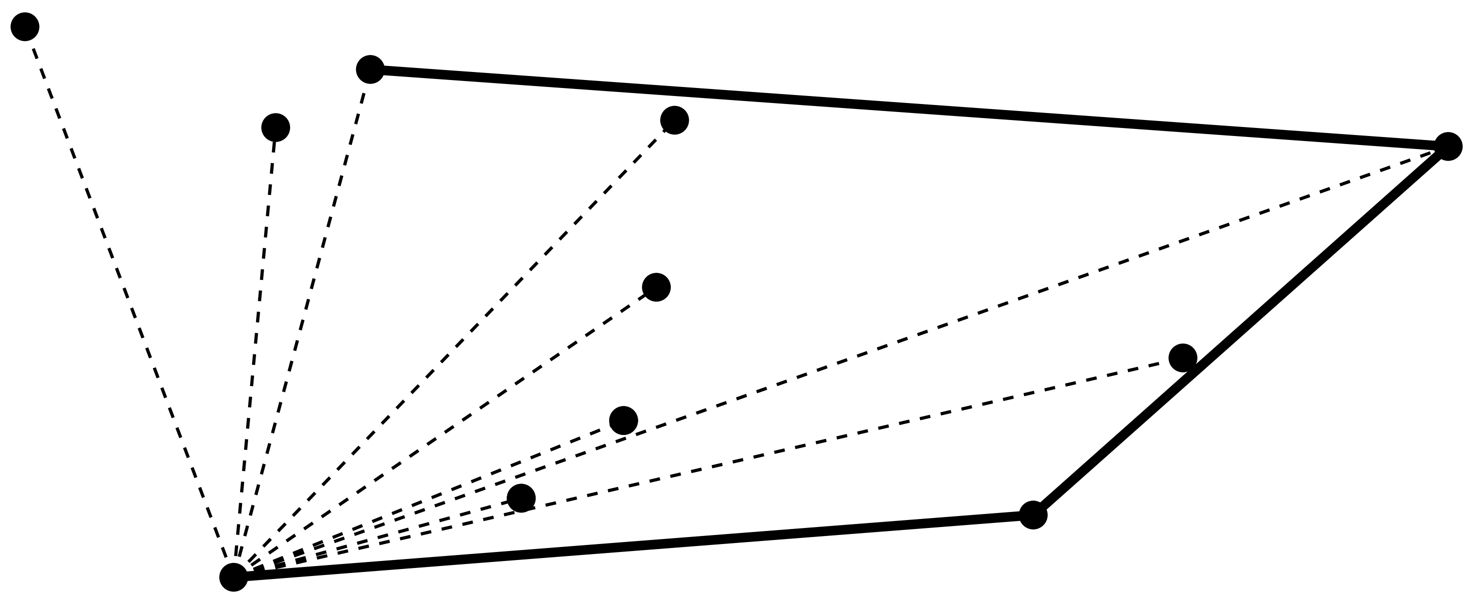
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



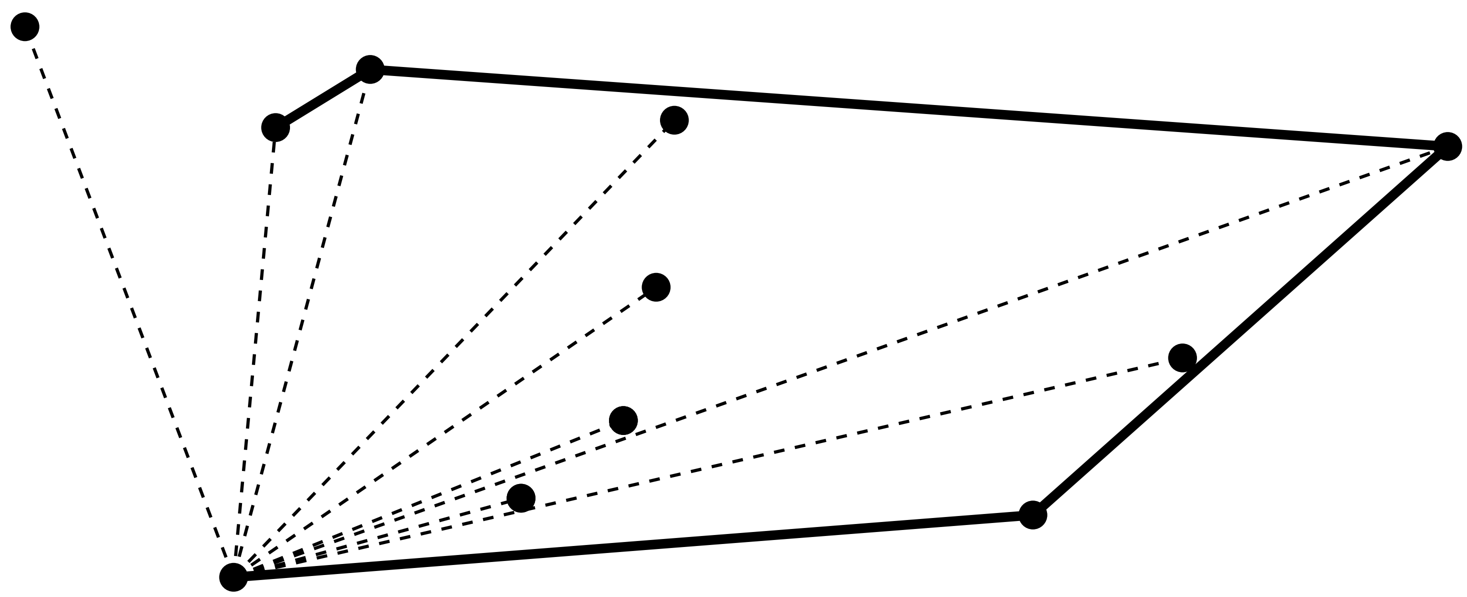
Recall: Graham Scan

- Create hull by “Successive Local Repair”
- sequence sorted around extreme point: remove points not making a left turn



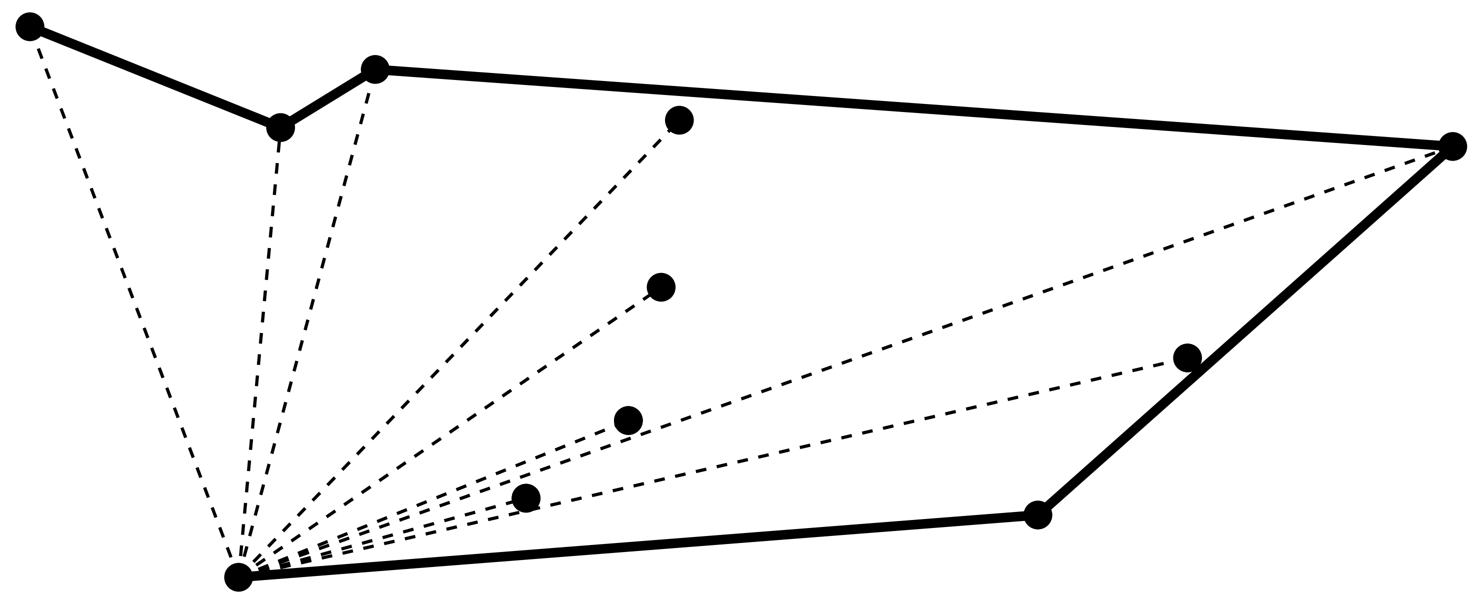
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



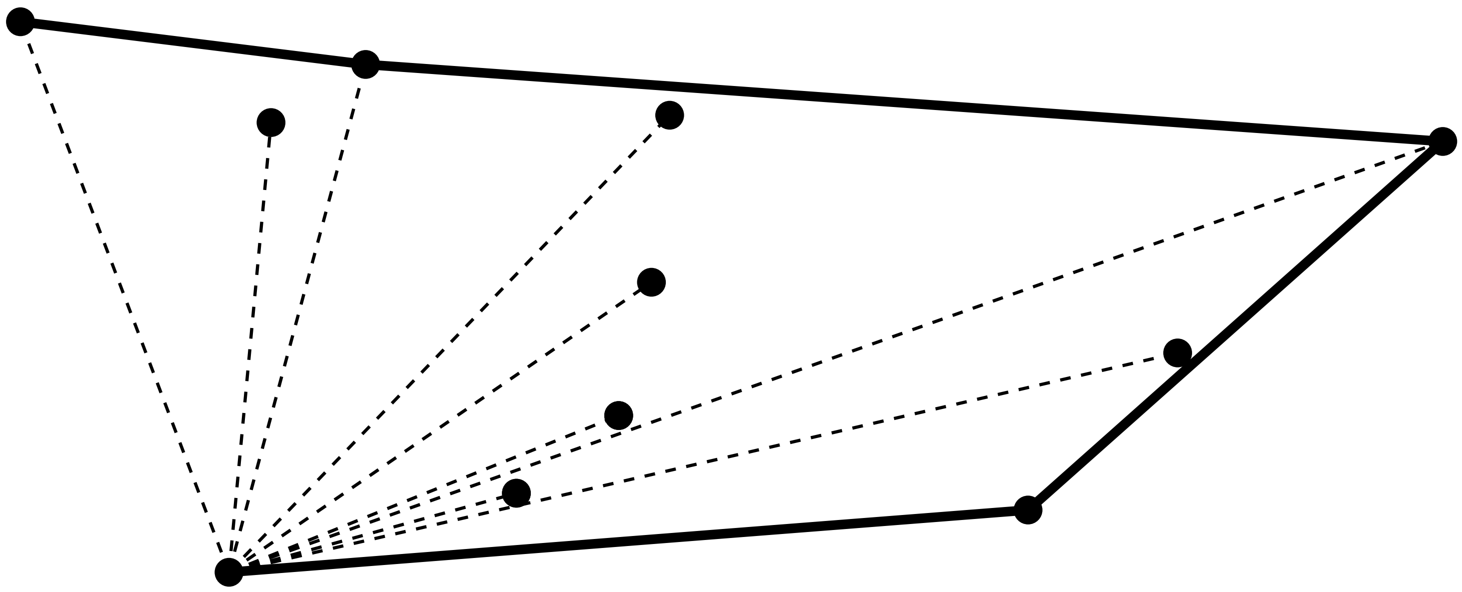
Recall: Graham Scan

- Create hull by “Successive Local Repair”
- sequence sorted around extreme point: remove points not making a left turn



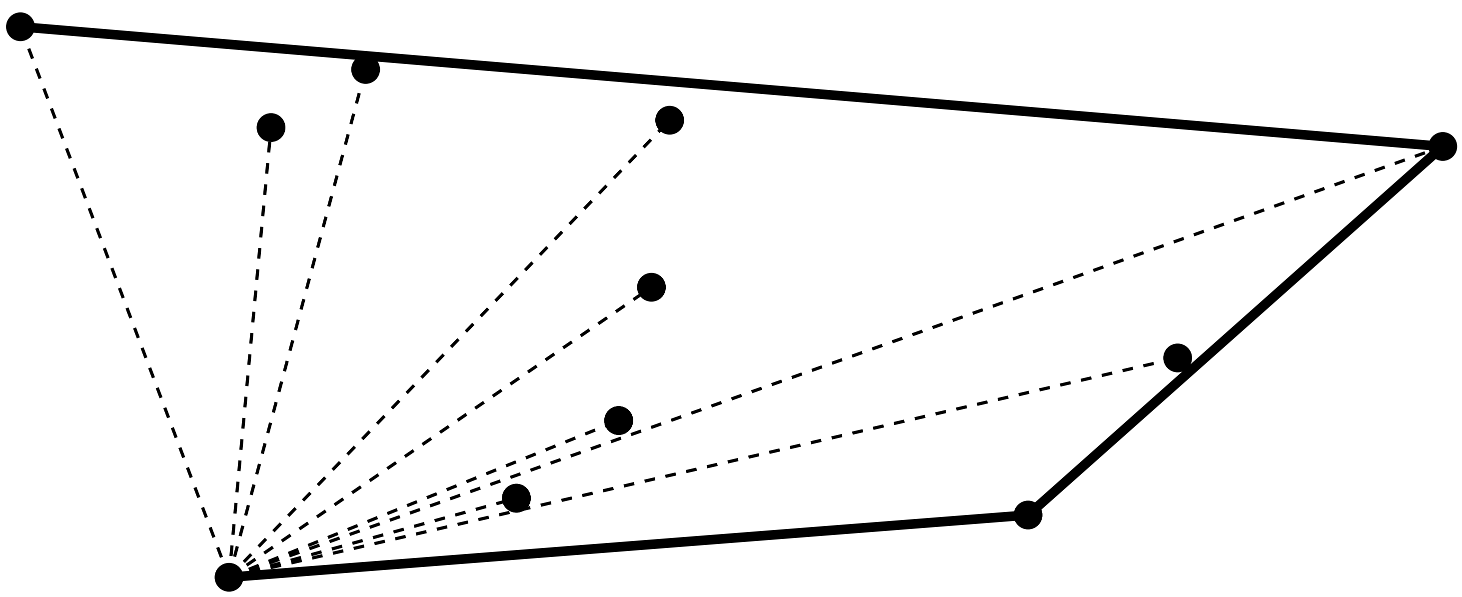
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



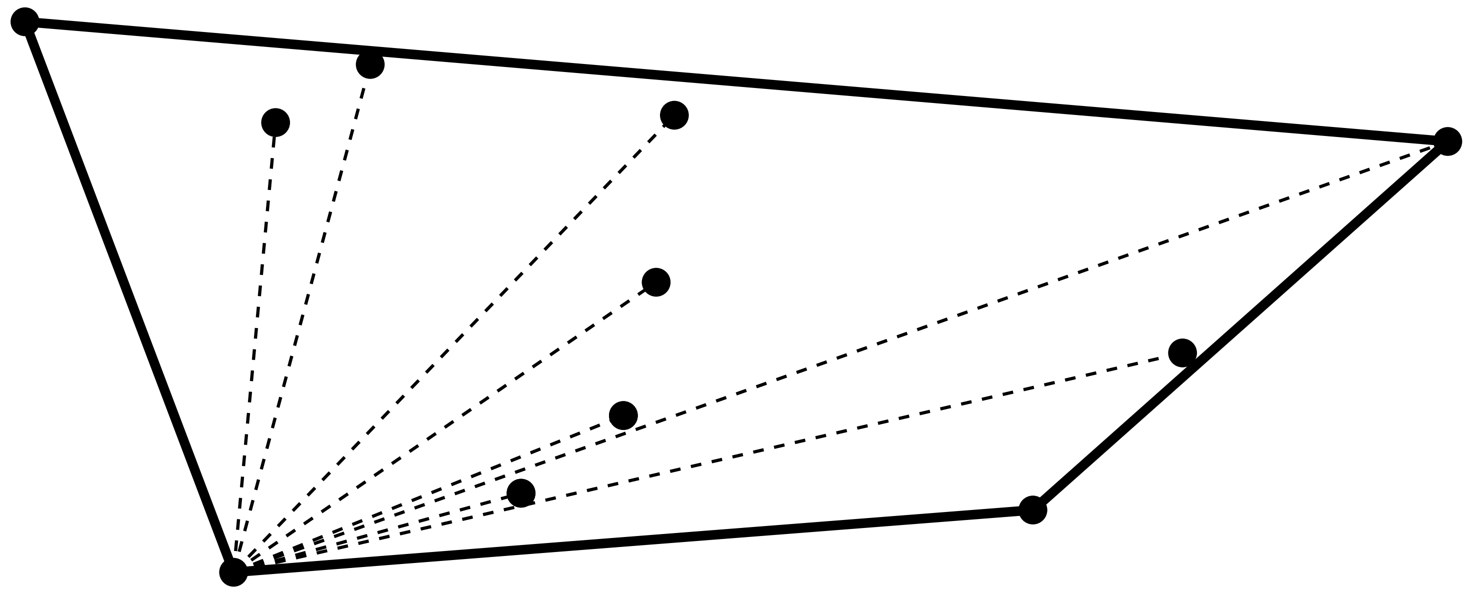
Recall: Graham Scan

- Create hull by "Successive Local Repair"
- sequence sorted around extreme point: remove points not making a left turn



Recall: Graham Scan

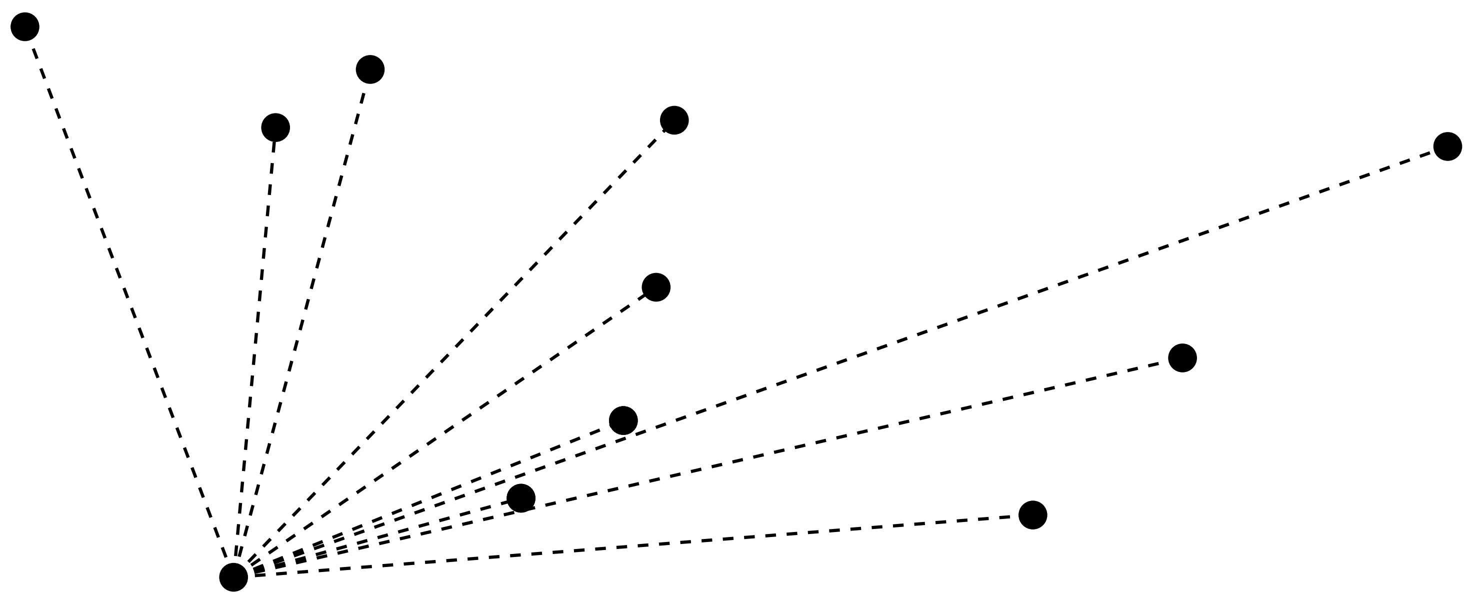
- Create hull by “Successive Local Repair”
- sequence sorted around extreme point: remove points not making a left turn



Recall: Graham Scan

Extend to also build a triangulation:

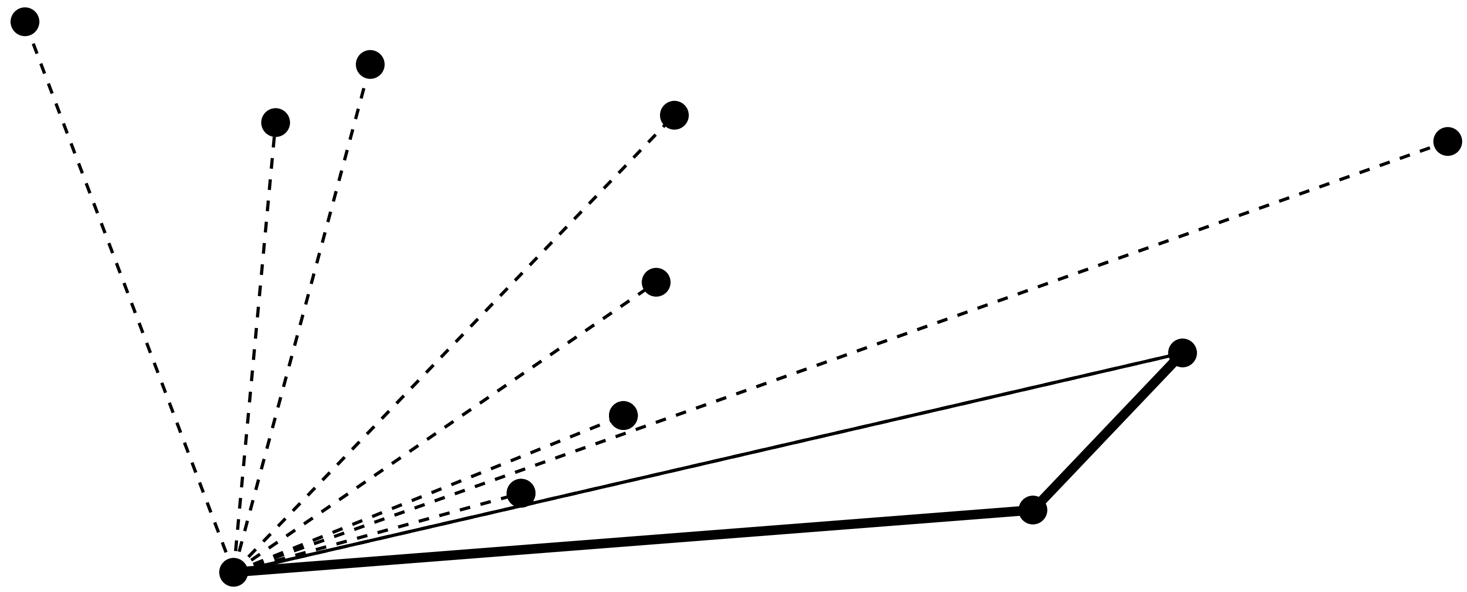
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

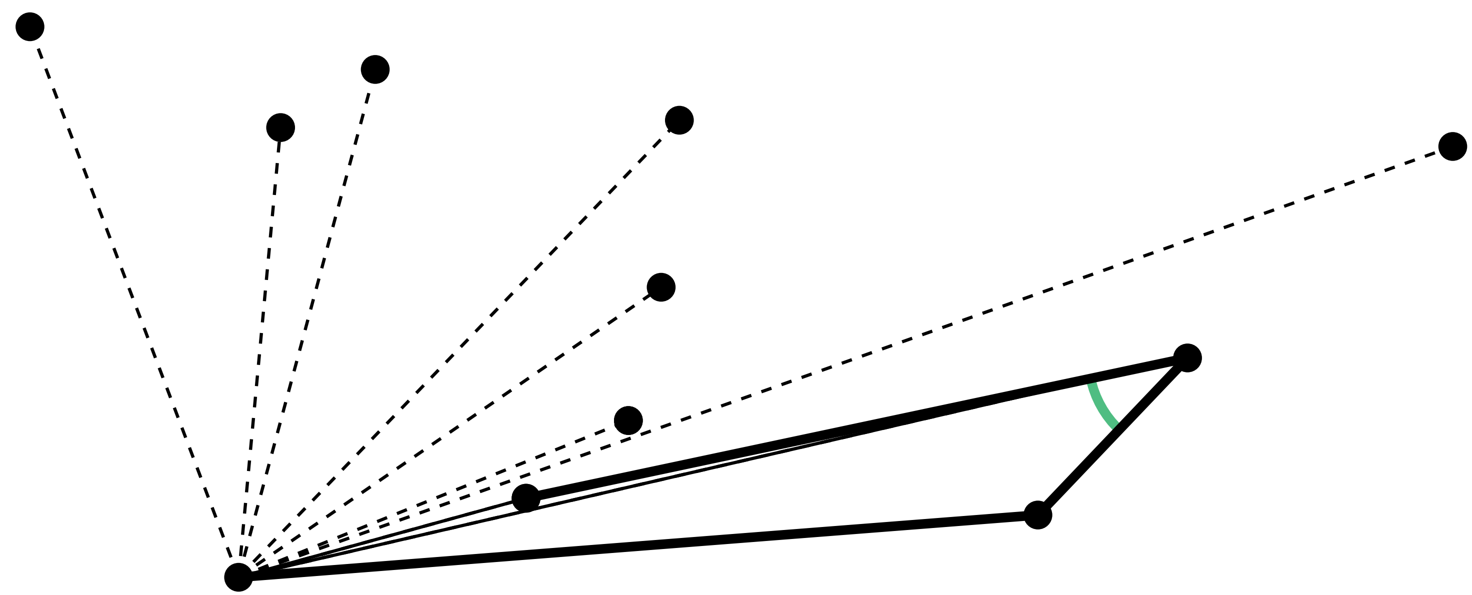
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

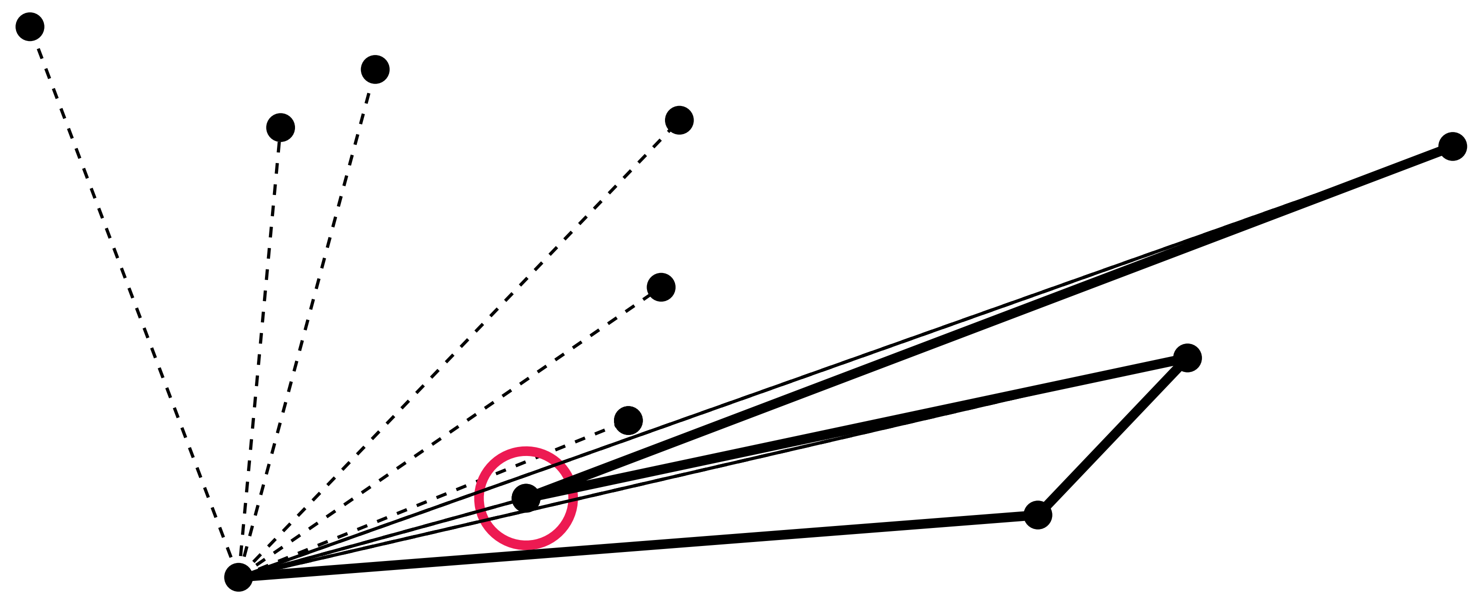
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

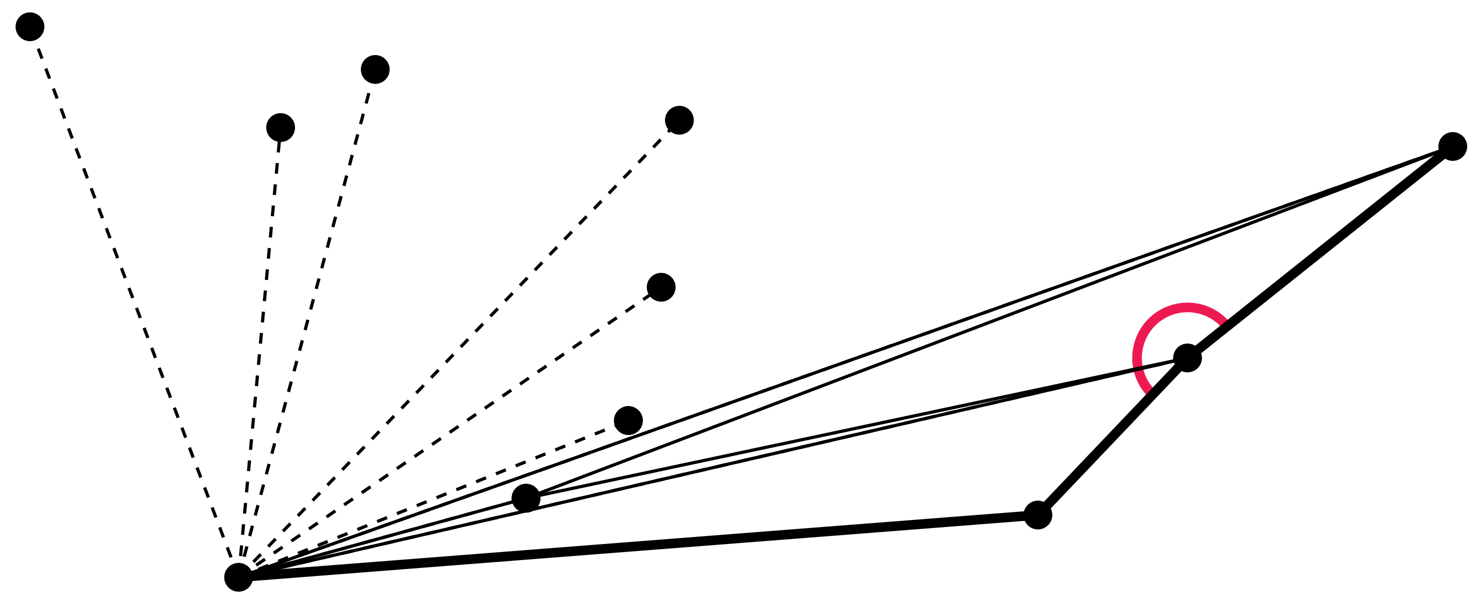
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

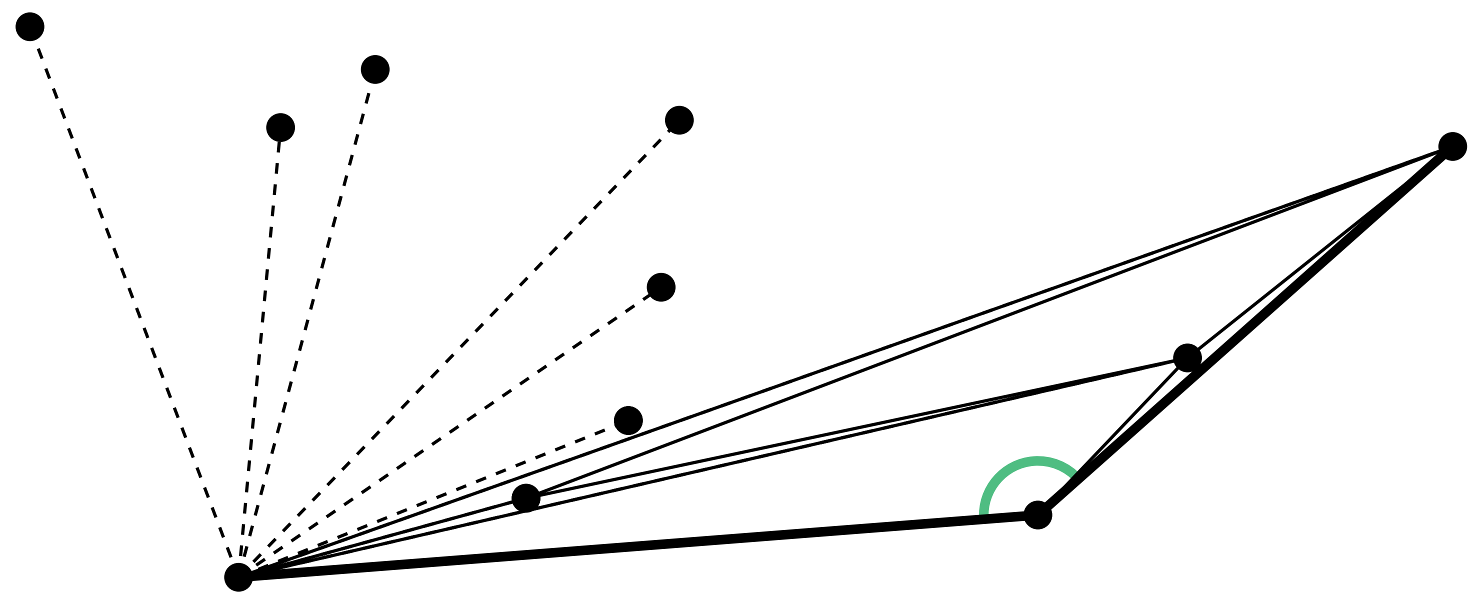
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

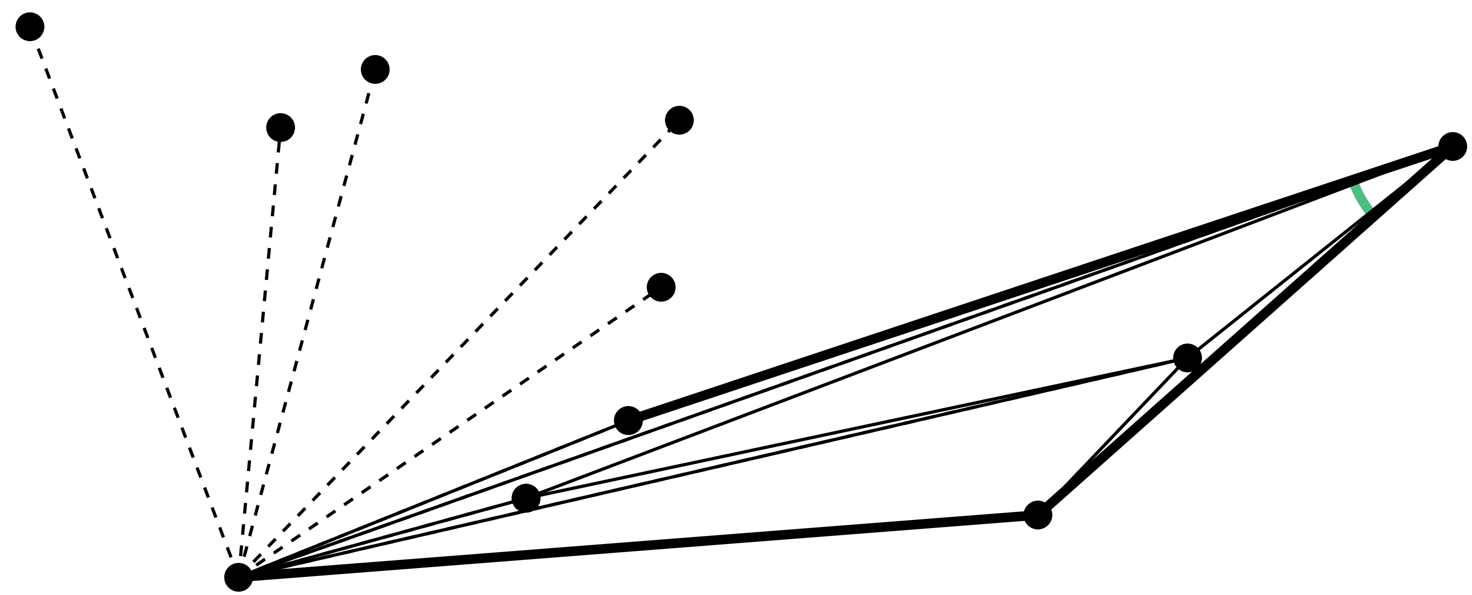
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

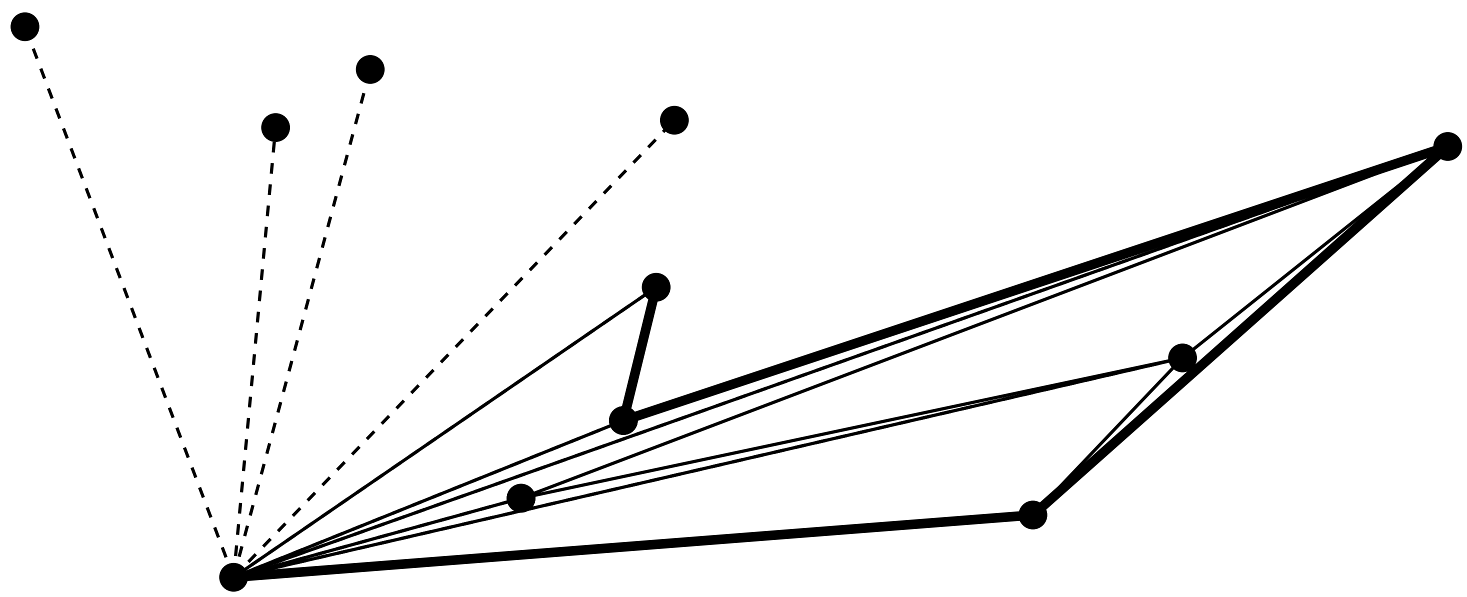
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

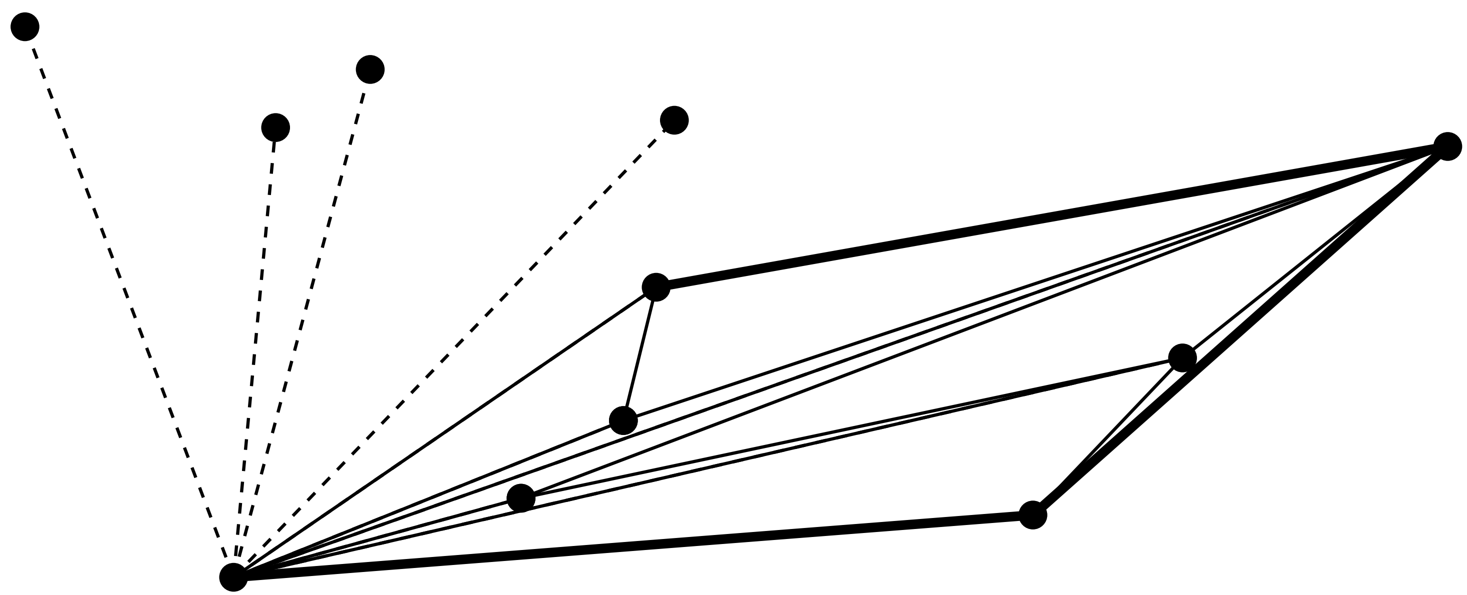
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

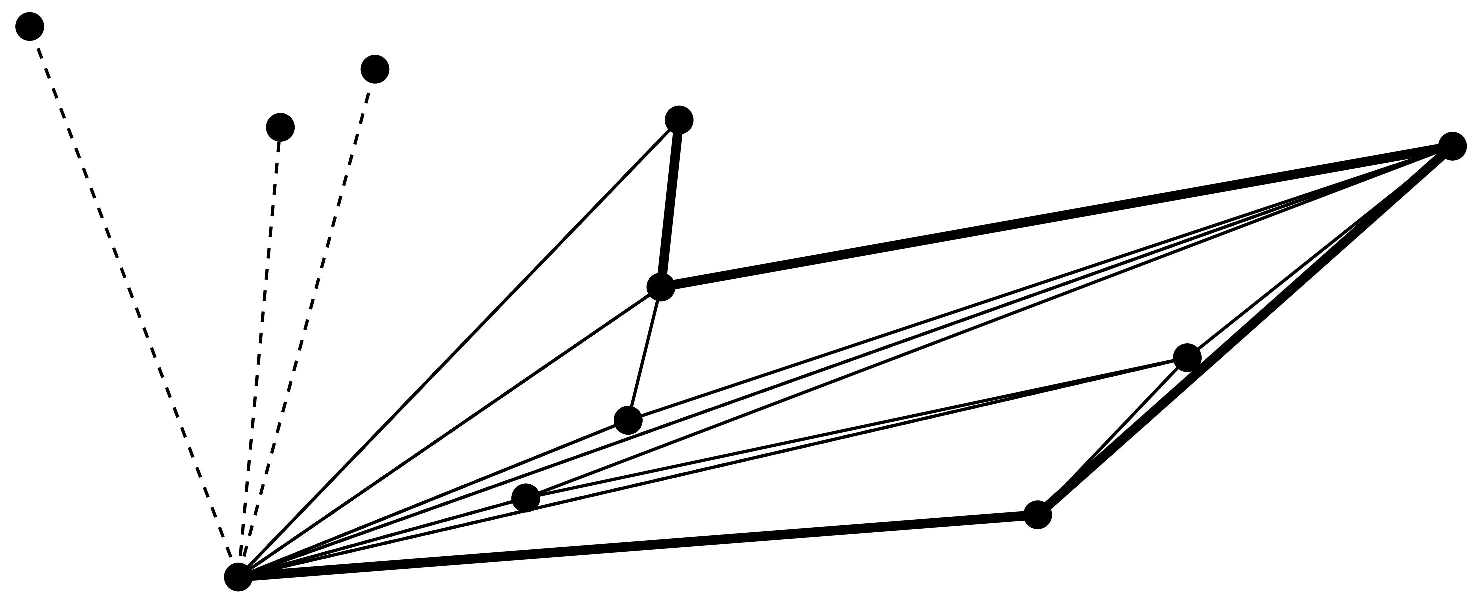
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

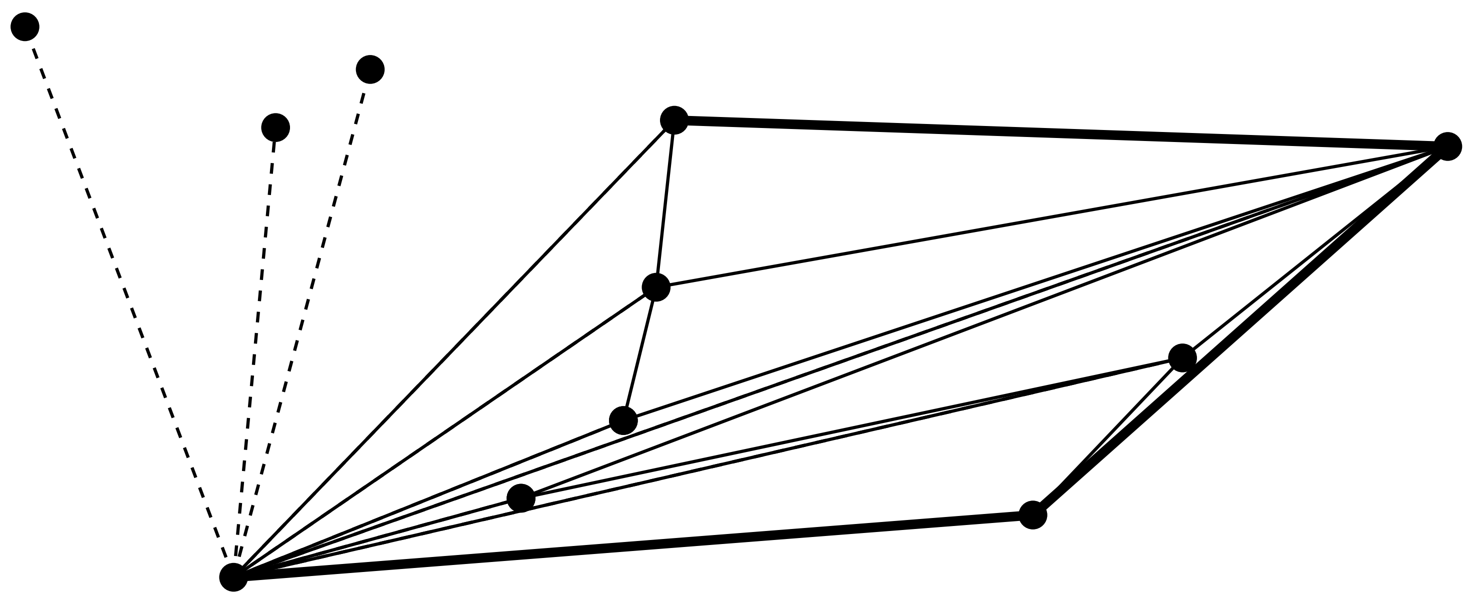
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

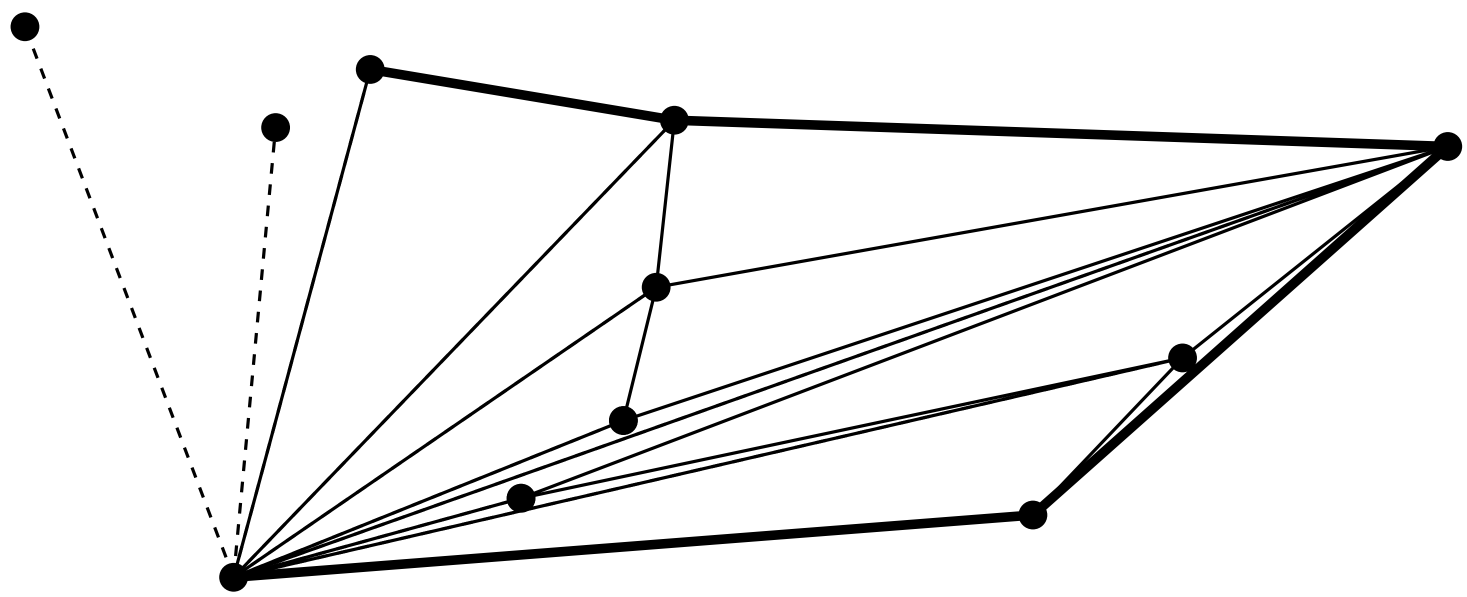
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

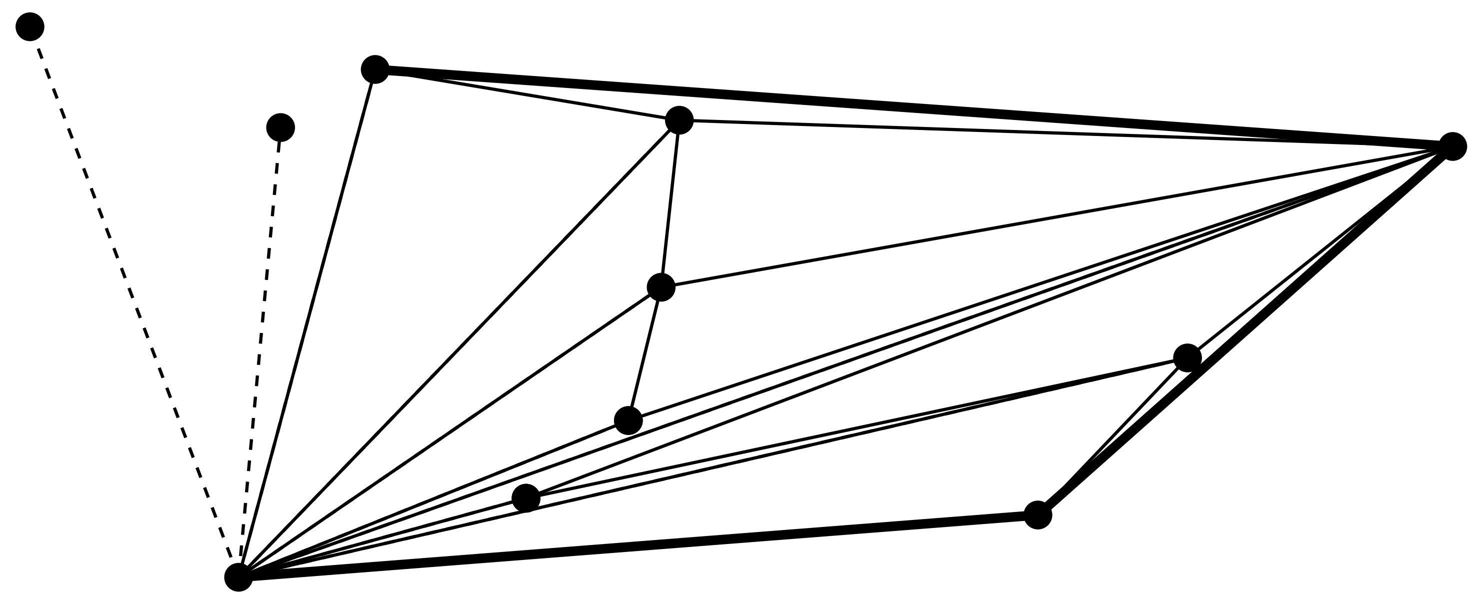
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

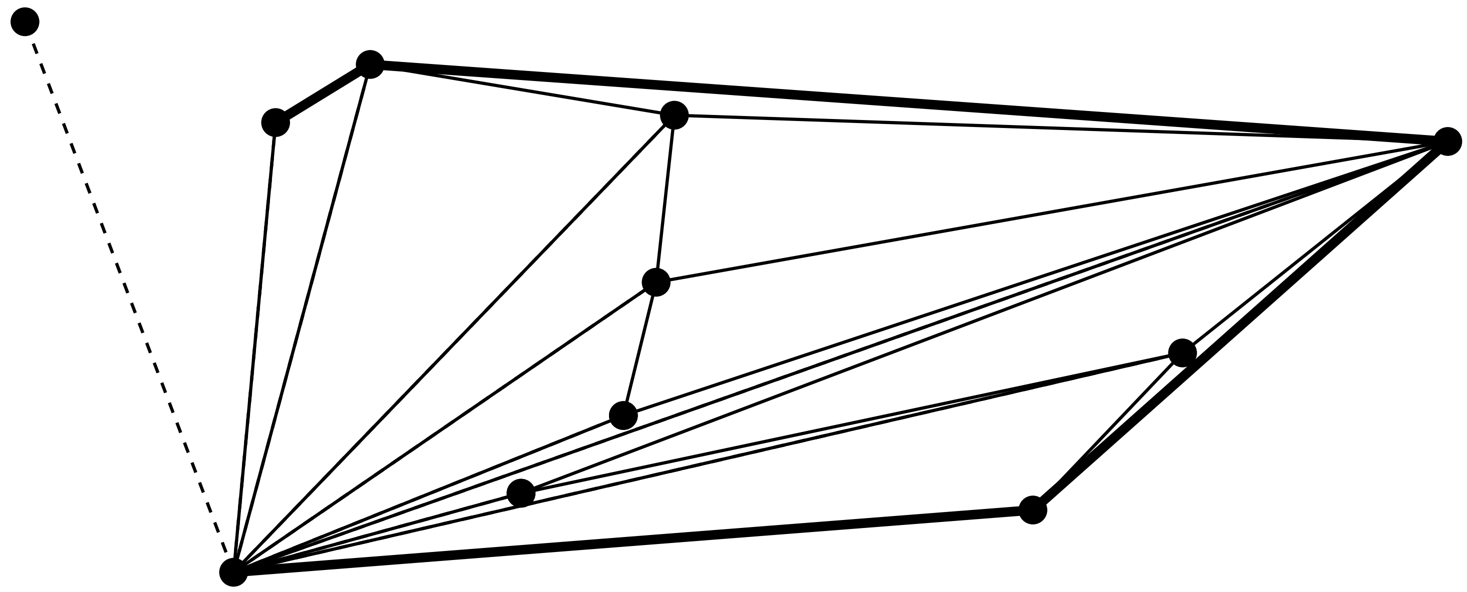
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

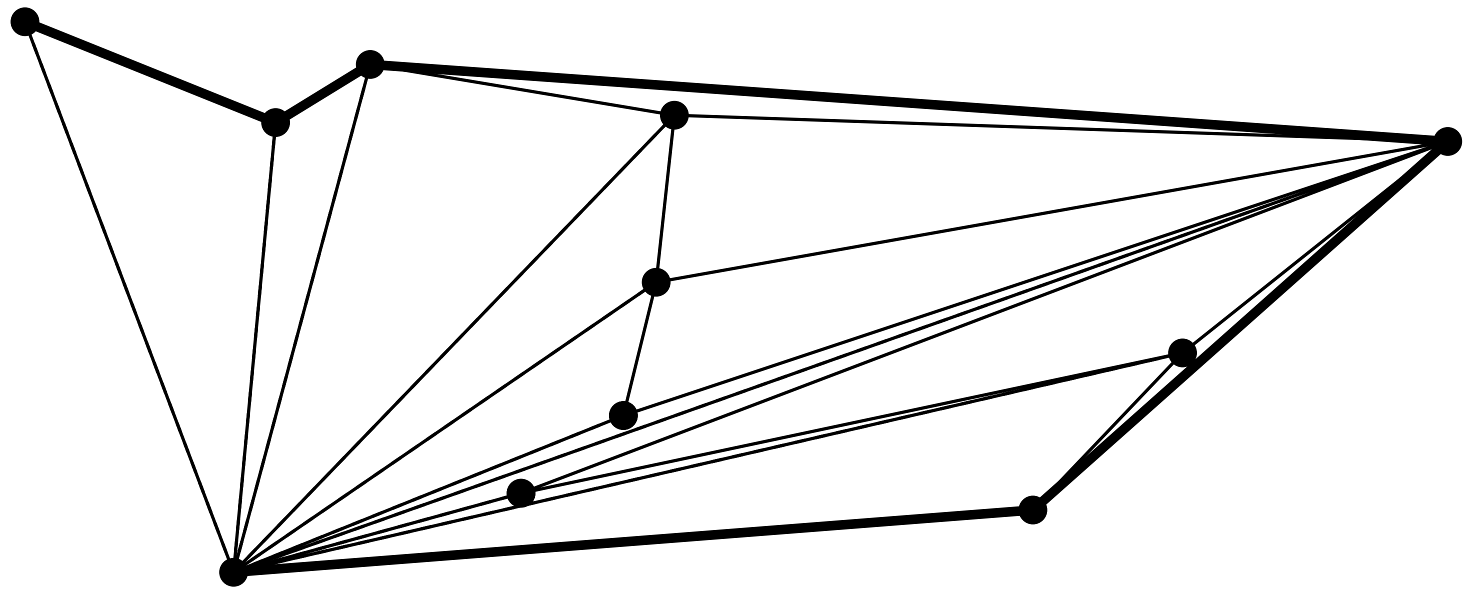
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

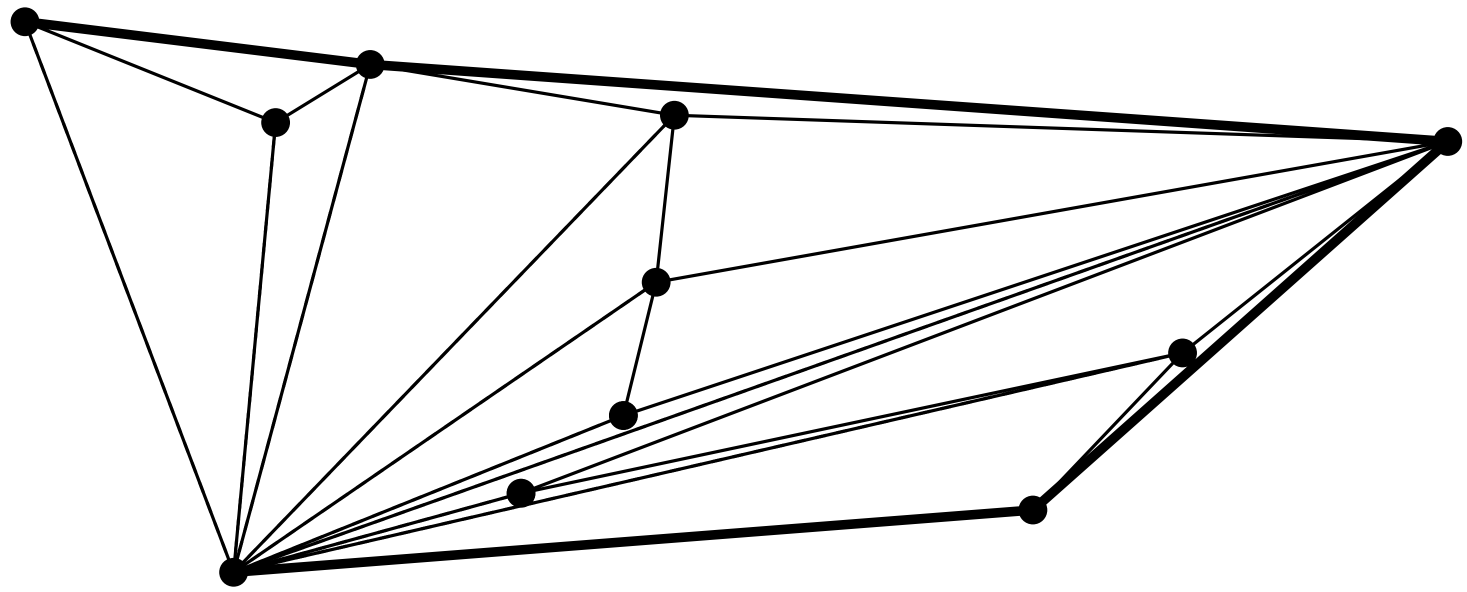
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

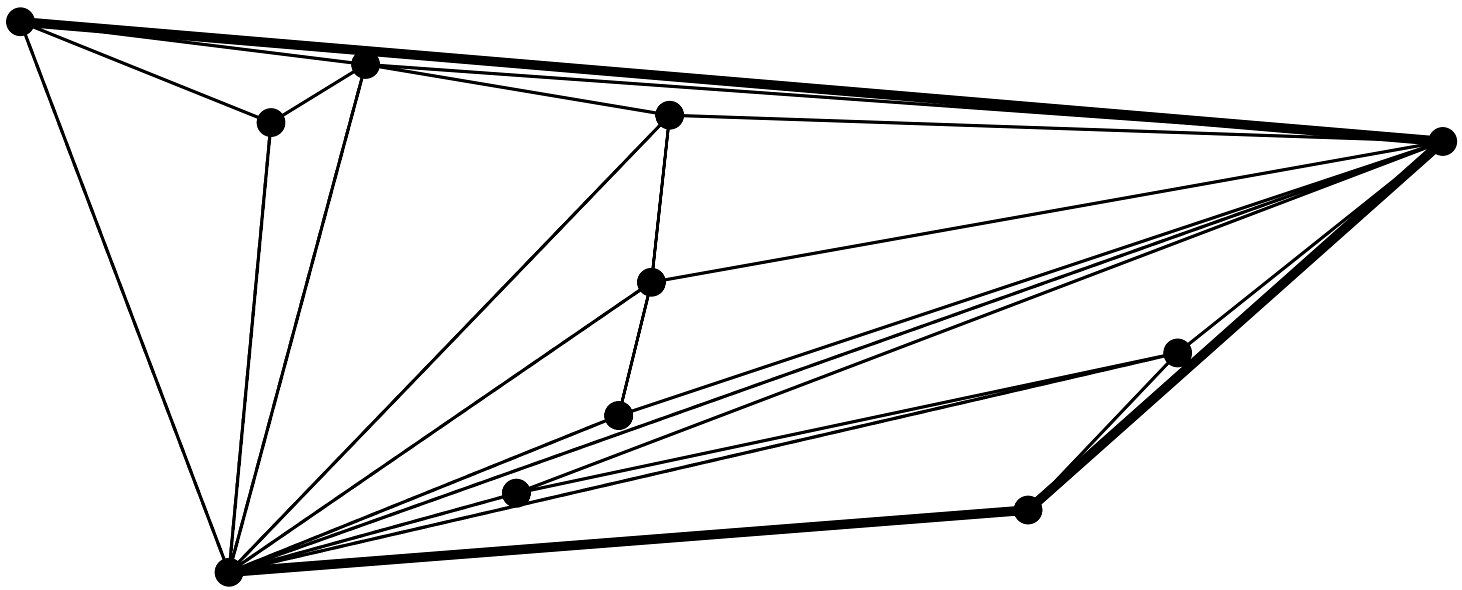
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

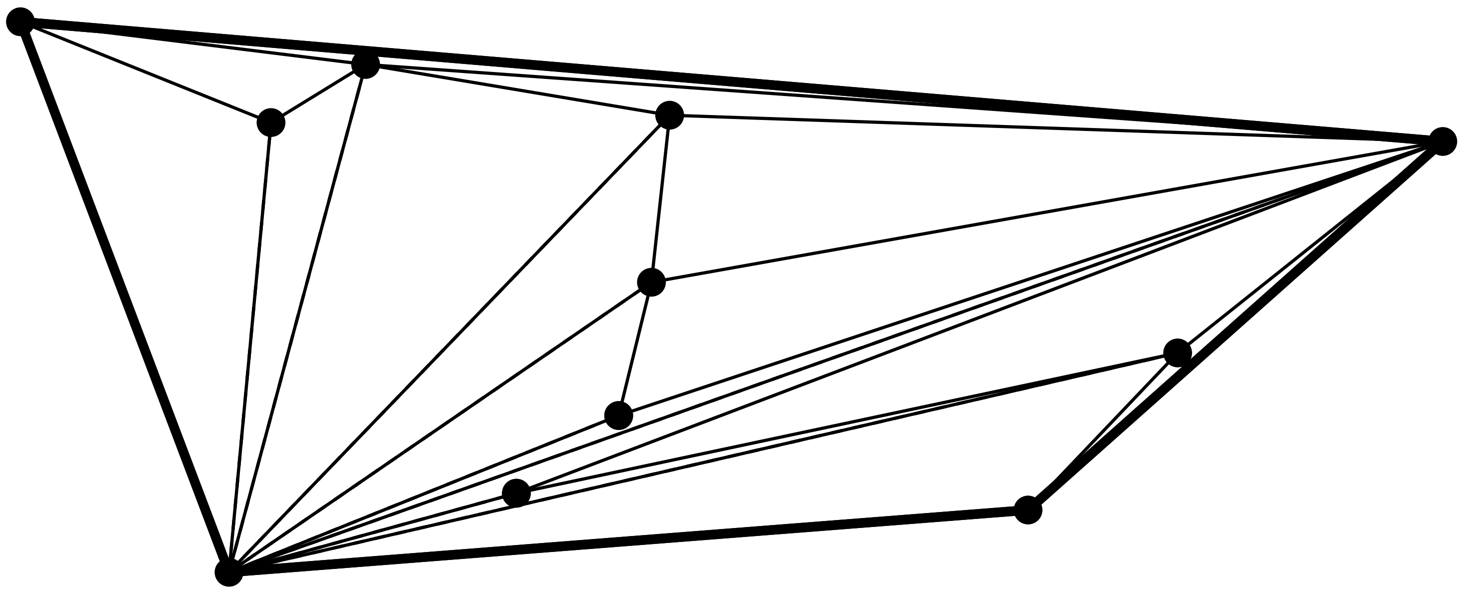
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

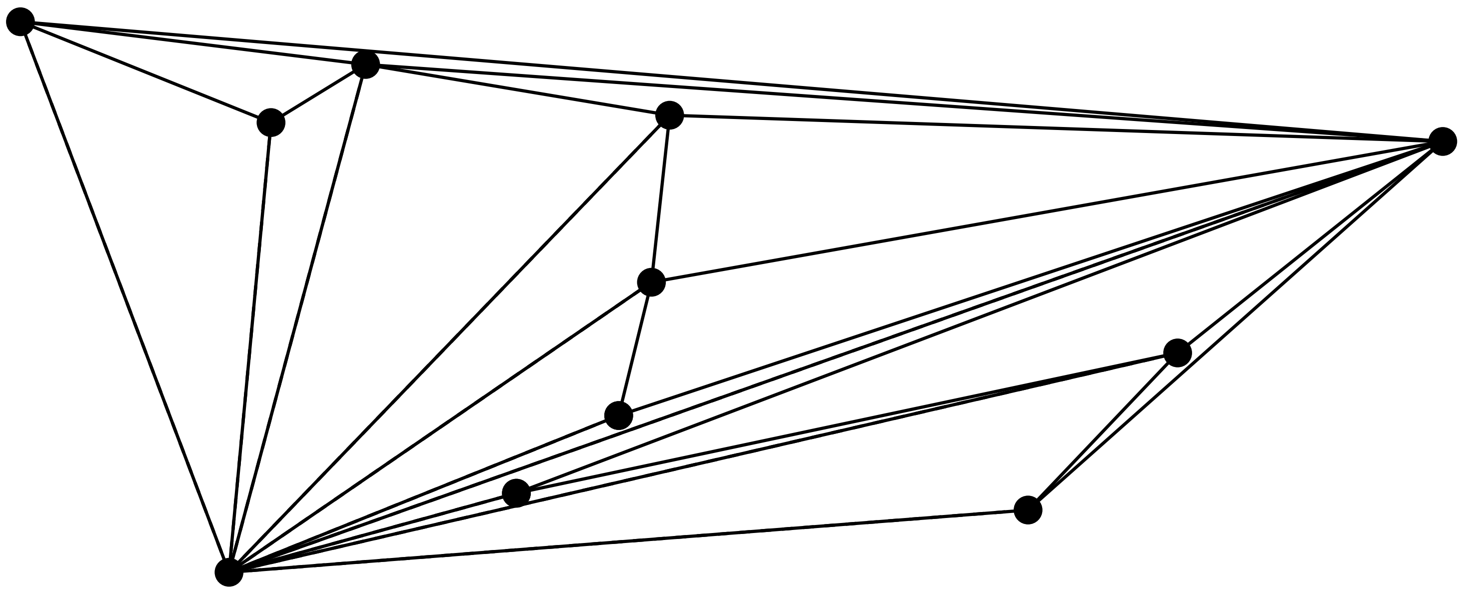
- add edges to anchor point and to last convex hull vertex
- don't remove edges



Recall: Graham Scan

Extend to also build a triangulation:

- add edges to anchor point and to last convex hull vertex
- don't remove edges



Triangulation with Graham Scan

Input:

- Array `p[1..N]` of points ($N \geq 3$)

Output:

- Array `q` with convex hull vertices (in order)
- Stack `t` with triangulation edges

Preparation:

- Place the point with smallest y-coordinate into `p[1]`
- Sort all other points counterclockwise around `p[1]`: `p[i]` is larger than `p[j]` if `p[i]` is left of the directed line from `p[1]` to `p[j]`
- `p[1]` and `p[2]` are the first two convex hull vertices: Add them in this order to `q`, add `(p[1],p[2])` to `t`.

Triangulation with Graham Scan

Process the remaining points from `p[3]` to `p[N]`.

Processing point `p[i]`:

- Add `(p[1],p[i])` and `(p[i-1],p[i])` to `t`
- While from the last edge of the convex hull there is no left turn to `p[i]`:
  - remove the last point from q.
  - add an edge from `p[i]`, to the last vertex in q to `t`.
- Add `p[i]` to q.

End:

- After `p[N]` has been processed, the convex hull vertices are stored in order in q and the triangulation edges are stored in `t`.

Triangulation with Graham Scan

```
for (i = 2 to N)
  if (p[i].y < p[1].y)   swap(p[1], p[i])
sort p[2..N] counterclockwise around p[1]
q[1] = p[1], q[2] = p[2], h = 2
t.push((p[1],p[2]))
for (i = 3 to N)
  t.push((p[1],p[i]))
  t.push((p[i-1],p[i]))   /* p[i-1] == q[h] */
  while (h>1 and not leftturn(q[h-1],q[h],p[i]))
     h = h - 1
     t.push((p[i],q[h]))
  h = h + 1
  q[h]= p[i]
```

Triangulation with Graham Scan

Running time:

- Preparation in $O(n \log n)$ time due to sorting.
- Buliding the triangulation: $\Theta(n)$ time. Why?

→ in total $O(n \log n)$ time.

Memory requirement:

- $O(n)$ in addition to input. Why?

Triangulation with Graham Scan

Correctness:

- After processing `p[3]`, we have a triangle.
- Assume that before the round for `p[i]`, $i \geq 4$, `t` contains all edges of a triangulation for `p[1..i-1]`
- In the round for `p[i]`, we add edges between `p[i]` and all points of the convex hull of `p[1..i-1]` that `p[i]` "sees" $\Rightarrow$ "fan" of triangles from `p[i]` to extreme points, no edge crosses the convex hull of `p[1..i-1]`.

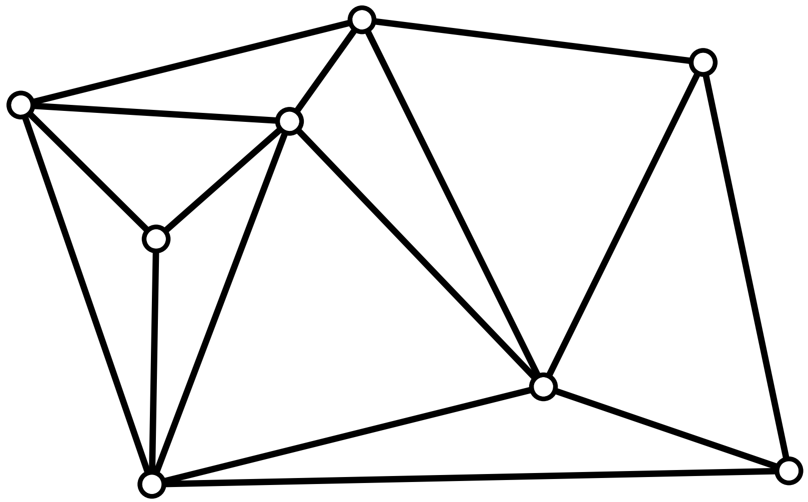
$\Rightarrow$ After the round for `p[i]`, `t` contains all edges of a triangulation for `p[1..i]`.

$\Rightarrow$ After the round for `p[N]`, `t` contains all edges of a triangulation for `p[1..N]`.

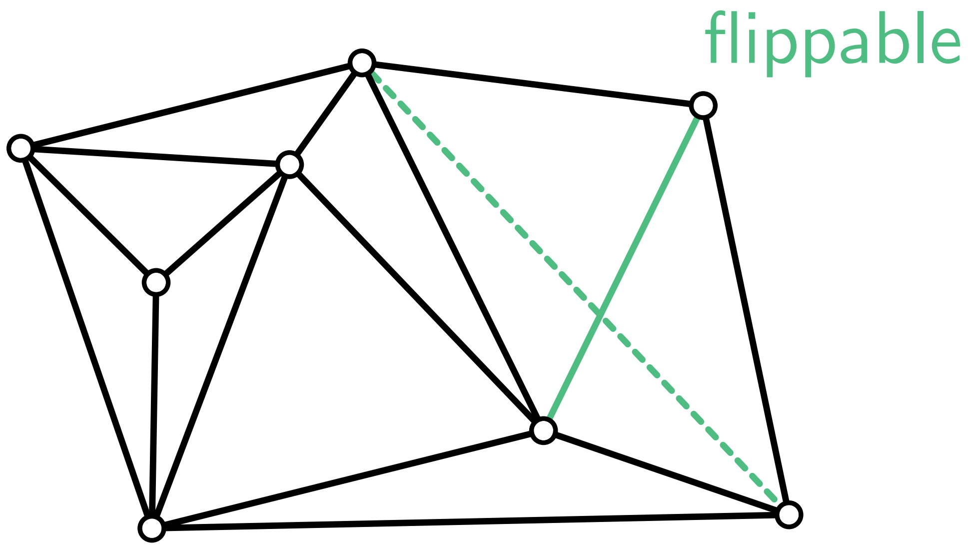
Local Transformation

Can we locally change a triangulation to get a different one?

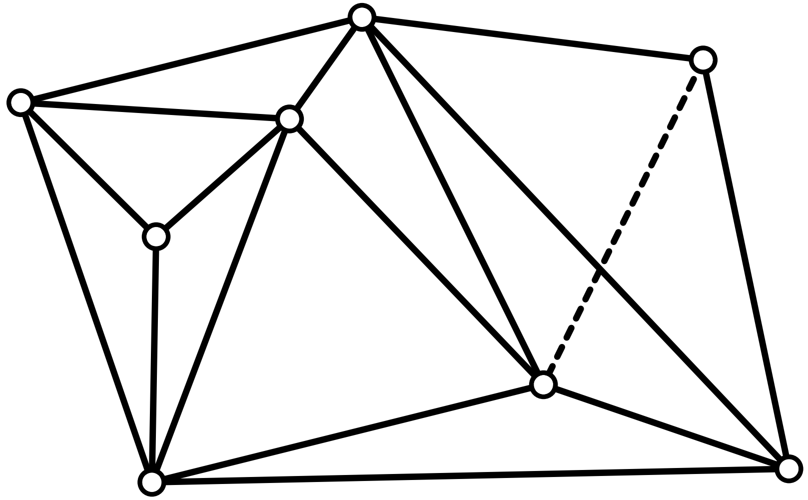
Edge Flips



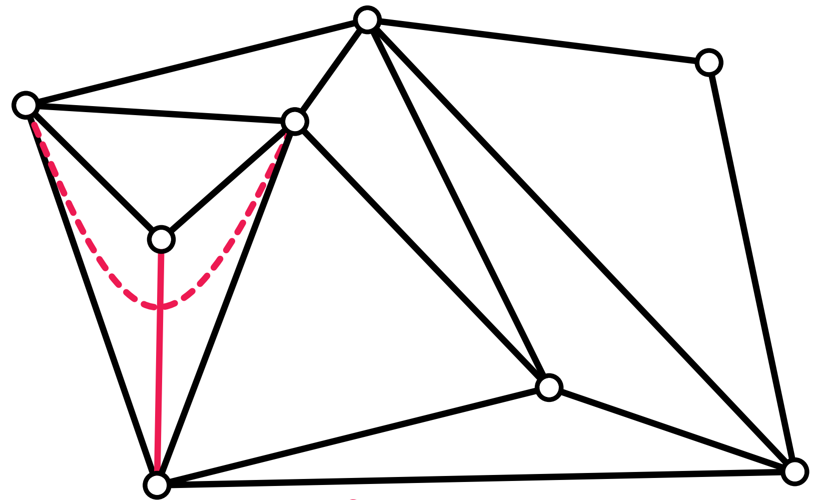
Edge Flips



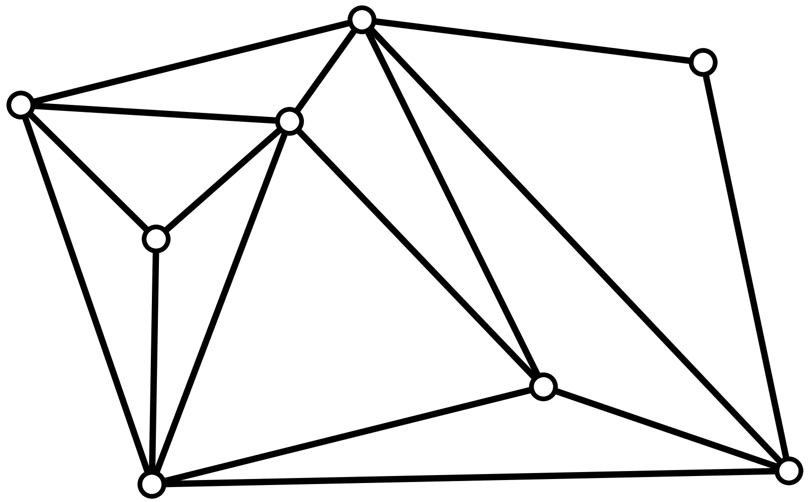
Edge Flips



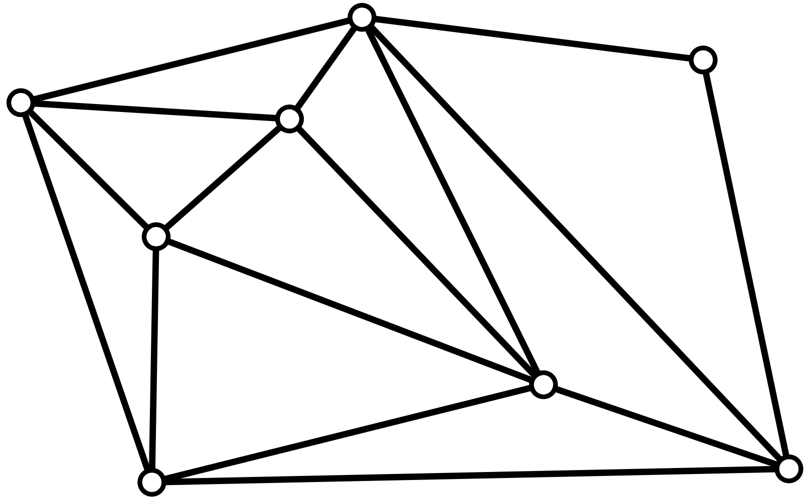
Edge Flips



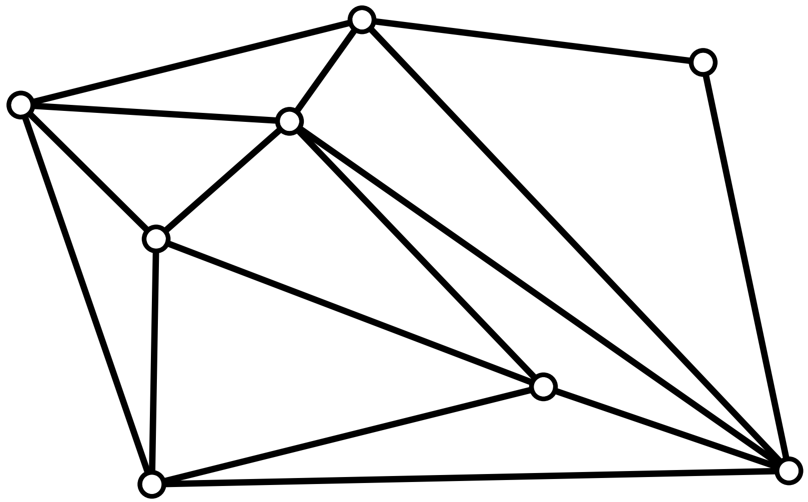
Edge Flips



Edge Flips

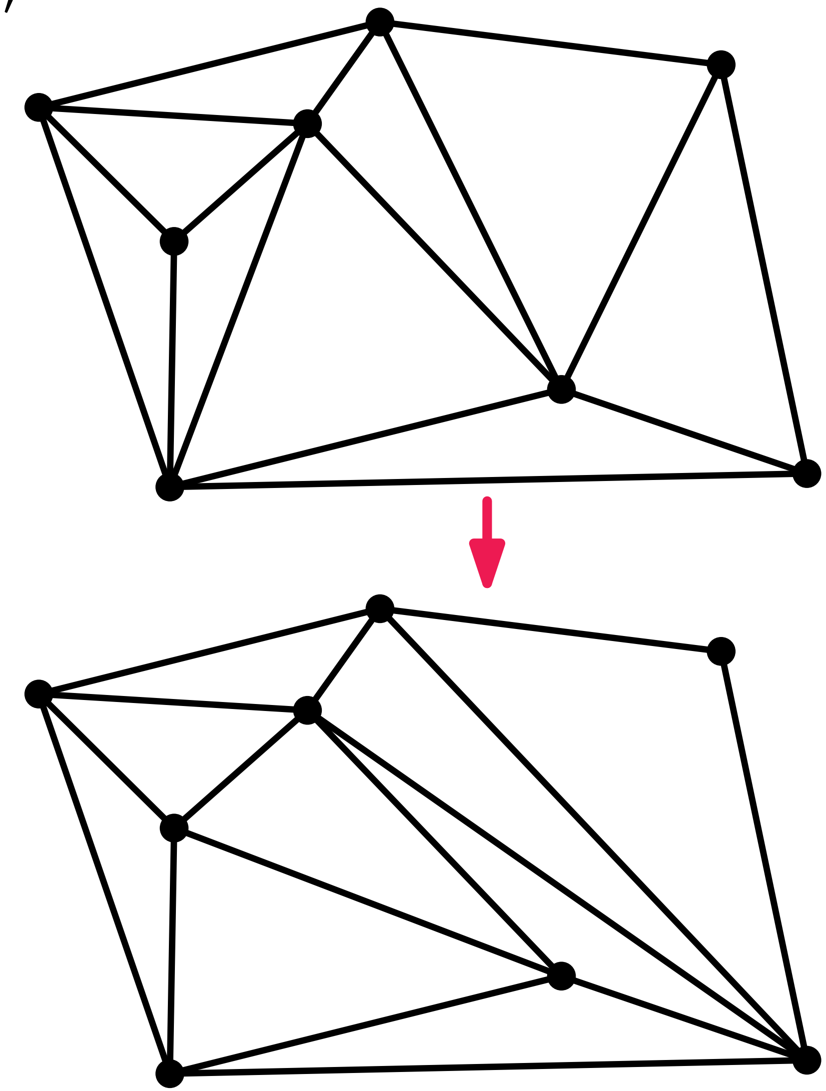


Edge Flips



The Flip Distance Problem

- **Given:** two triangulations T, T' of a point set.
- **Goal:** transform T into T' by subsequently *flipping* one edge at a time.
- Any triangulation can be transformed into any other by $O(n^2)$ flips (tight).
- **Question:** what is the minimum number of flips needed, the *flip distance*?
- Determining flip distance is NP-complete.

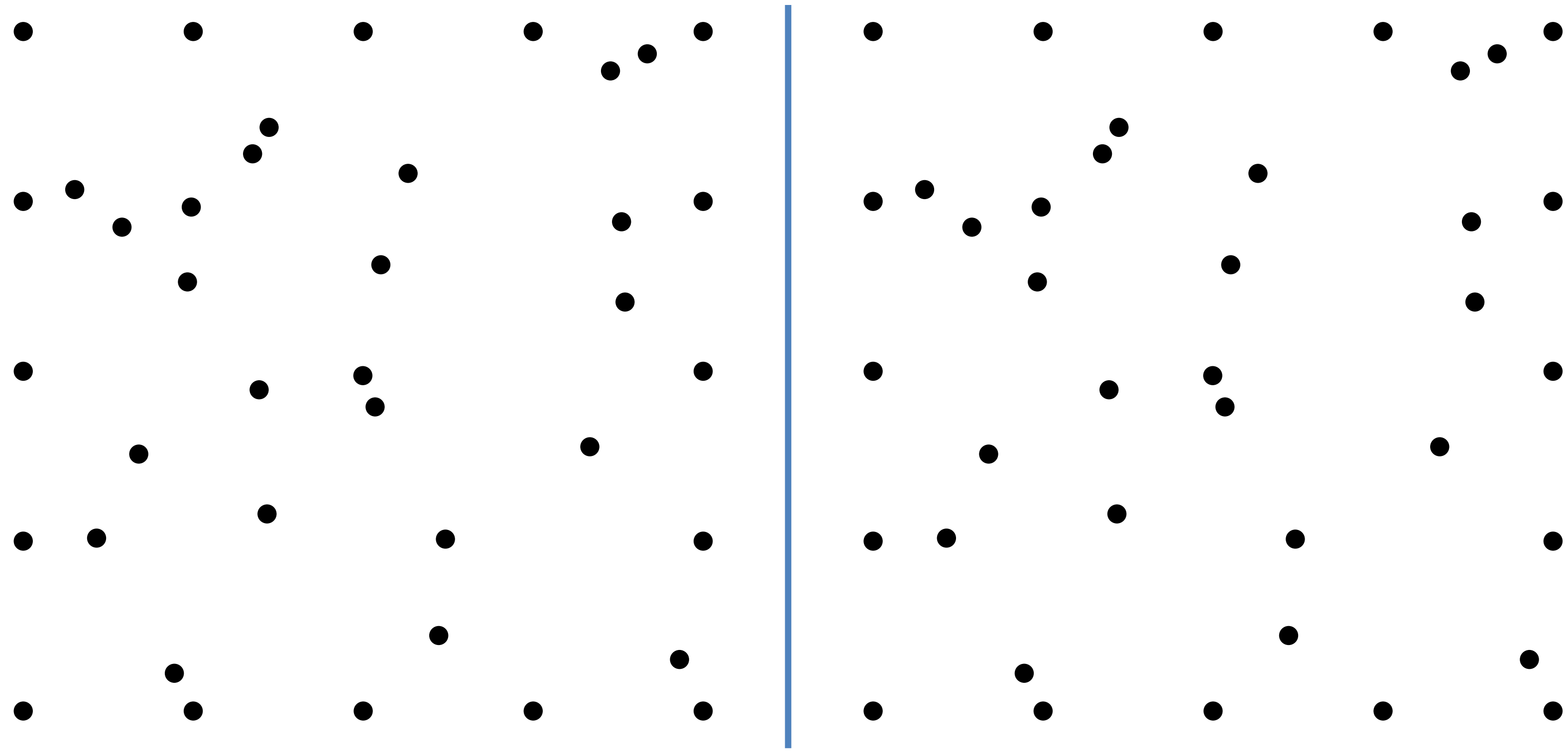


Conclusion

- Triangulations are a paramount data structure in Computational Geometry.
- A triangulation for an n-point set can be constructed in $O(n \log n)$ time and $\Theta(n)$ space.
- Every triangulation can be obtained from any other triangulation of the same point set by flips.
- Useful in practice and theory: Delaunay triangulation

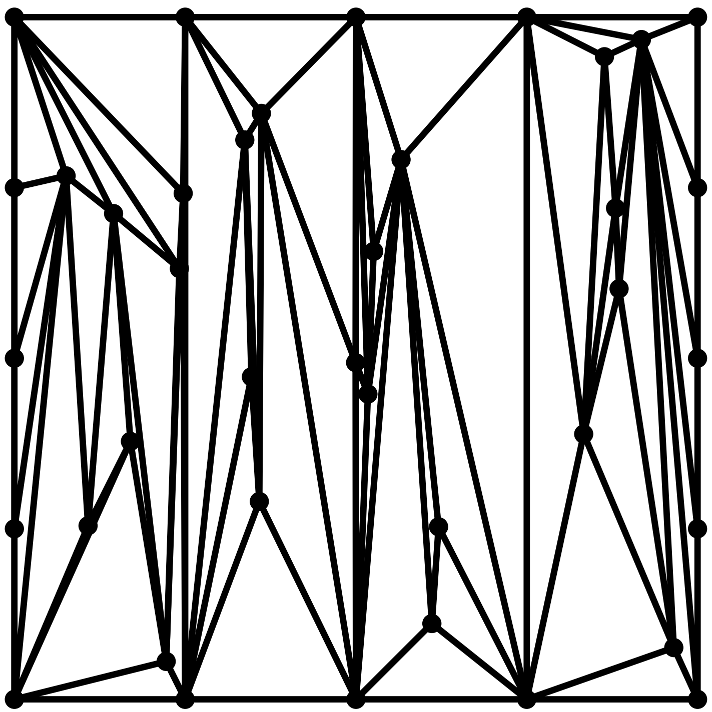
Conclusion

Two copies of the same point set ...

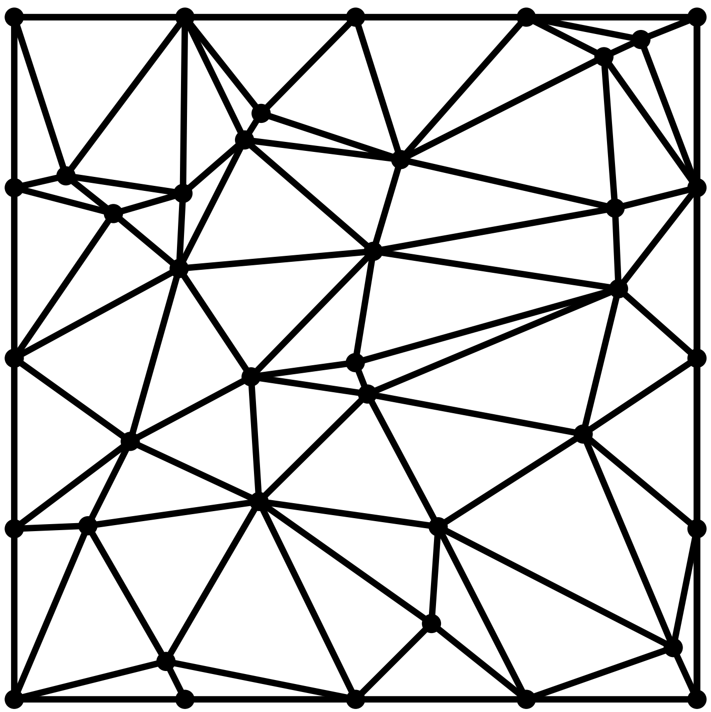


Conclusion

Two copies of the same point set ...



Canonical triangulation

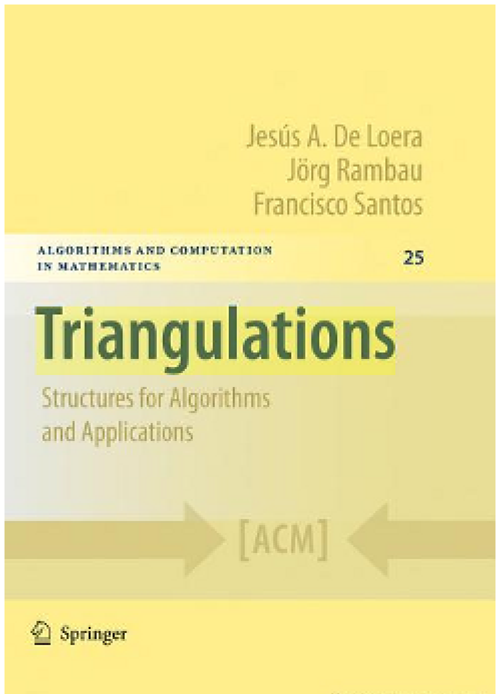


Delaunay triangulation

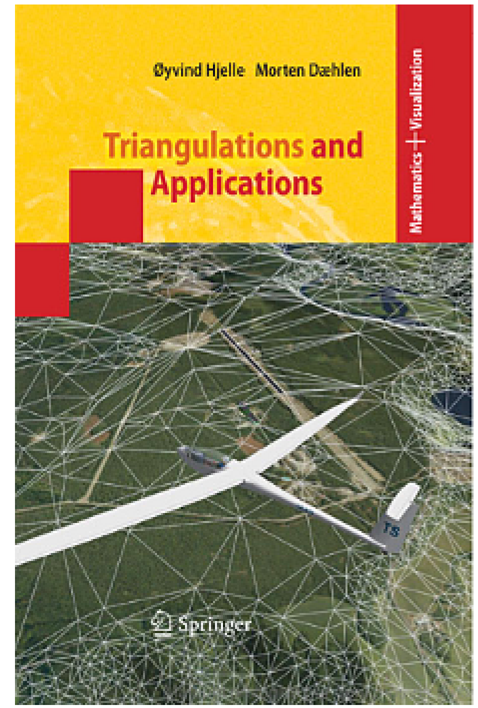
Conclusion

- Triangulations are a paramount data structure in Computational Geometry.
- A triangulation for an n-point set can be constructed in $O(n \log n)$ time and $\Theta(n)$ space.
- Every triangulation can be obtained from any other triangulation of the same point set by flips.
- Useful in practice and theory: Delaunay triangulation
  - obtainable by simple flip rules
  - optimizes several criteria
  - dual to the Voronoi diagram
- Flips also used in heuristics for optimization.

Sources / Further Reading



De Loera, Rambau, Santos: Triangulations (2010)



Hjelle, Dæhlen: Triangulations and Applications (2006)