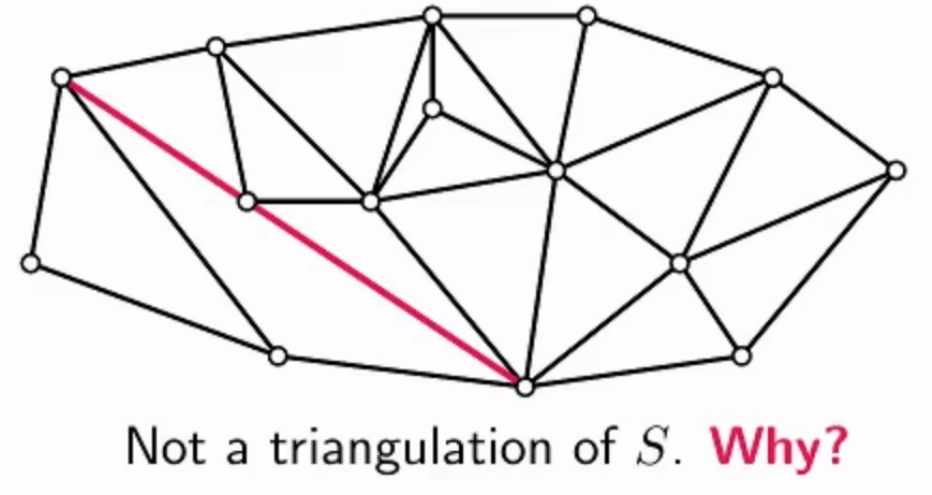
Triangel

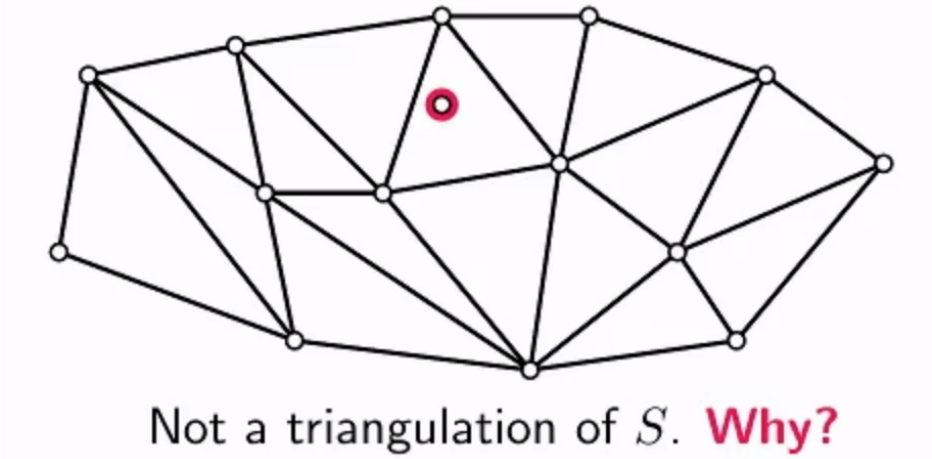
- simplest piece of surface
- allow modelling surfaces
- important for Computer Graphics

Triangulation of Point Set S

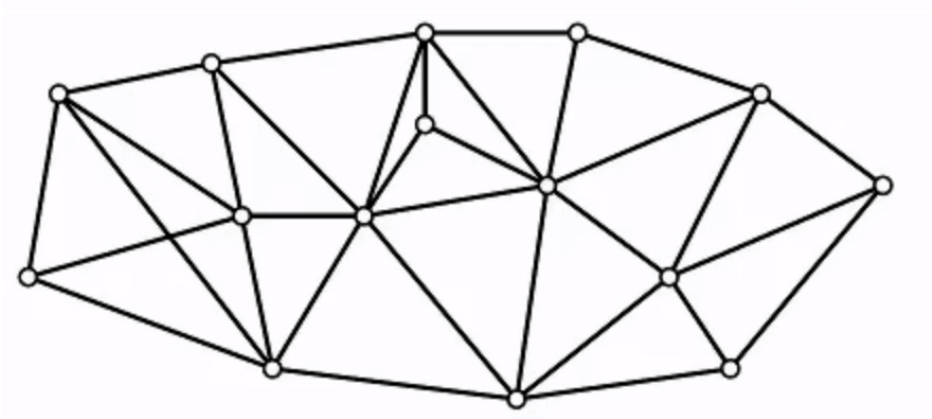
- [[Convex Hull]] partitioned into interior-disjoint triangels
- no point of S inside a segment or triangle



–



–

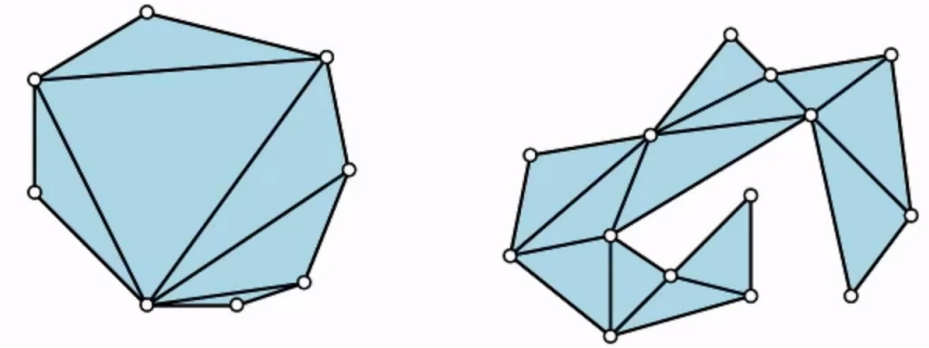


–

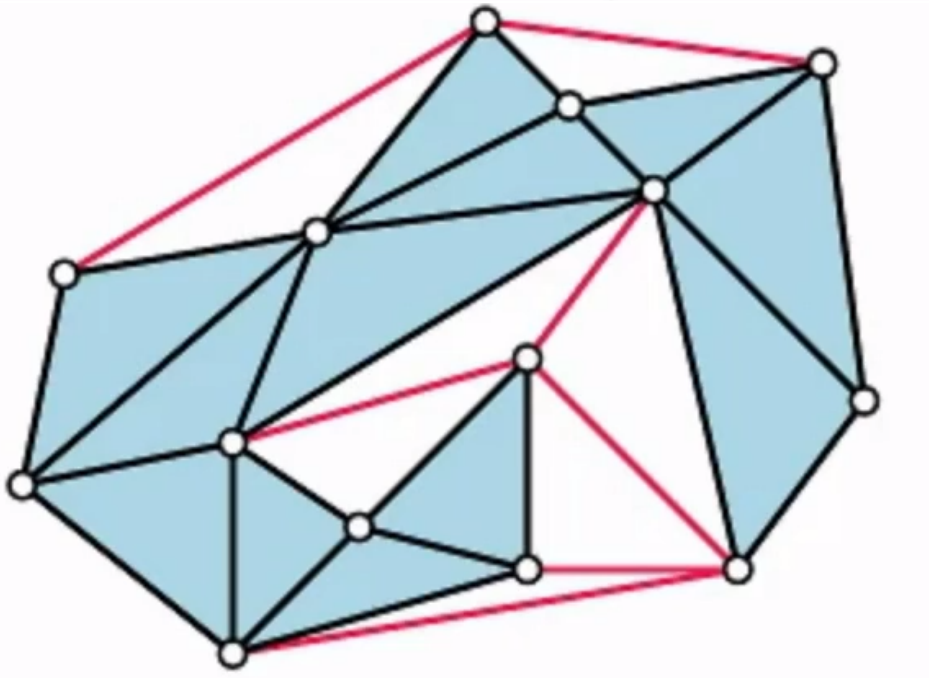
Triangulation of Polygon P

- polygon P with vertex set S partitioned into interior-disjoint triangles
- no point of S inside a segment (or triangle)

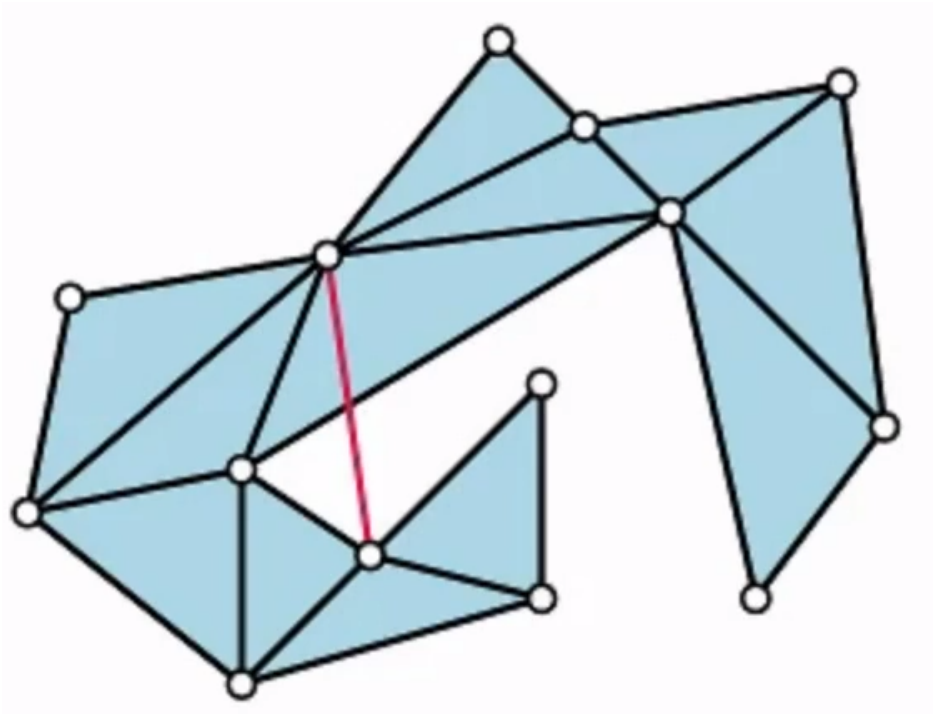
–



–

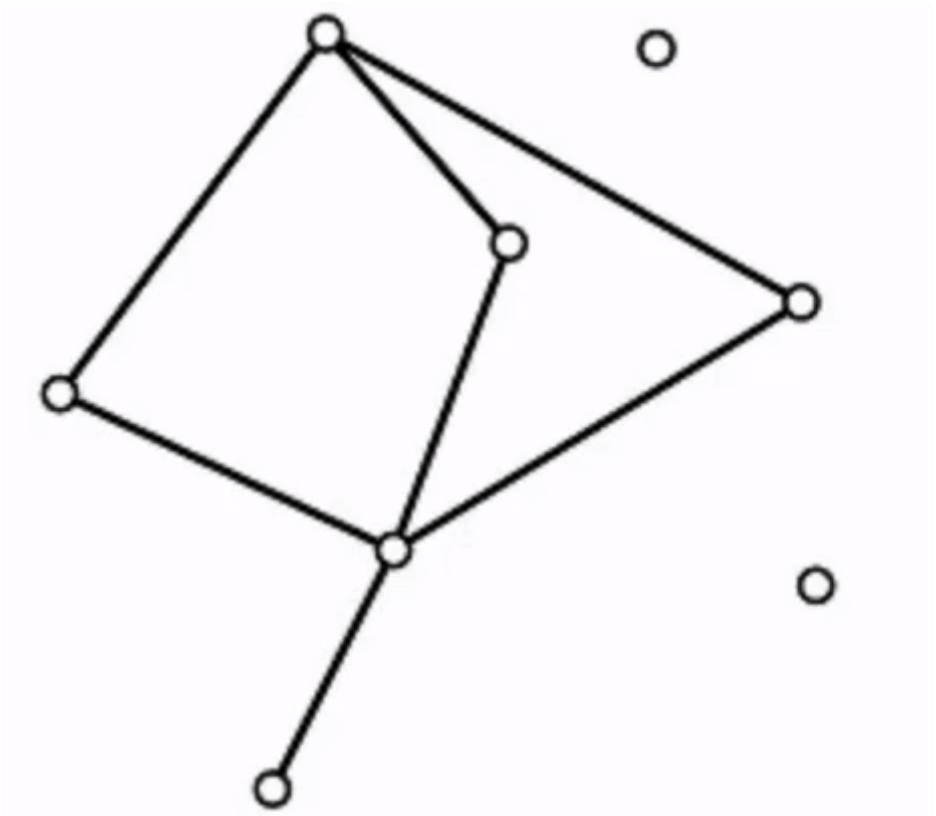
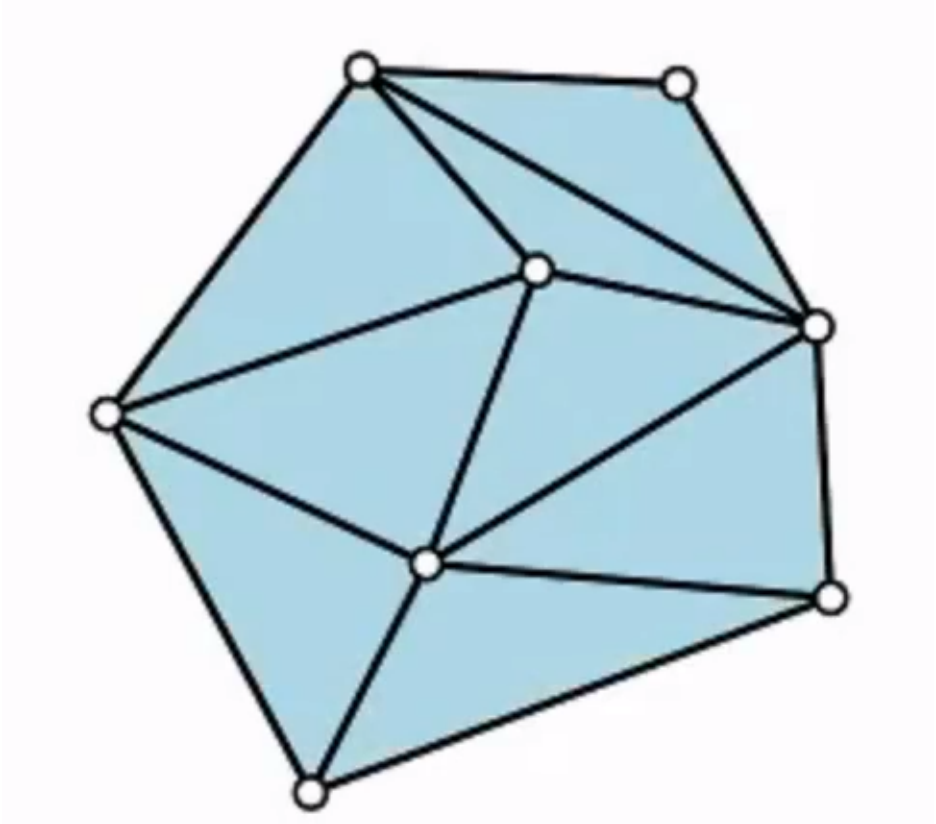


–



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 Triangulation

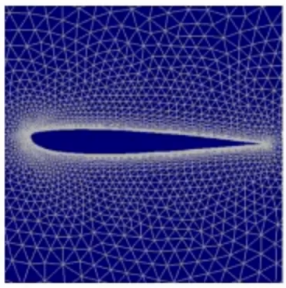
- A triangulation consists of vertices, edges, and triangles.

- number of edges and triangels depend on number of extreme points h of S
  - $\Rightarrow$ 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

Applications

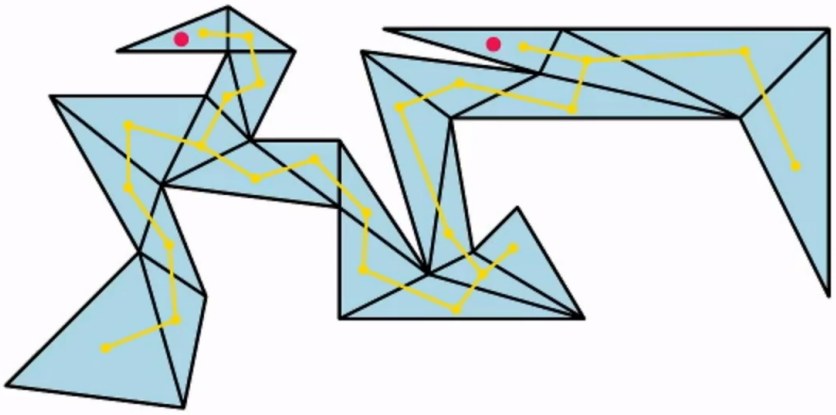
- Graphics/GIS
  - The height of a terrain is measured at certain points.
  - The points in a triangulation are elevated.
  - An approximation of the terrain is obtained.
  - Surfaces in 3D are modeled using meshes.
  - Every element is a triangle
    - allows fast processing by graphics hardware
  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



–

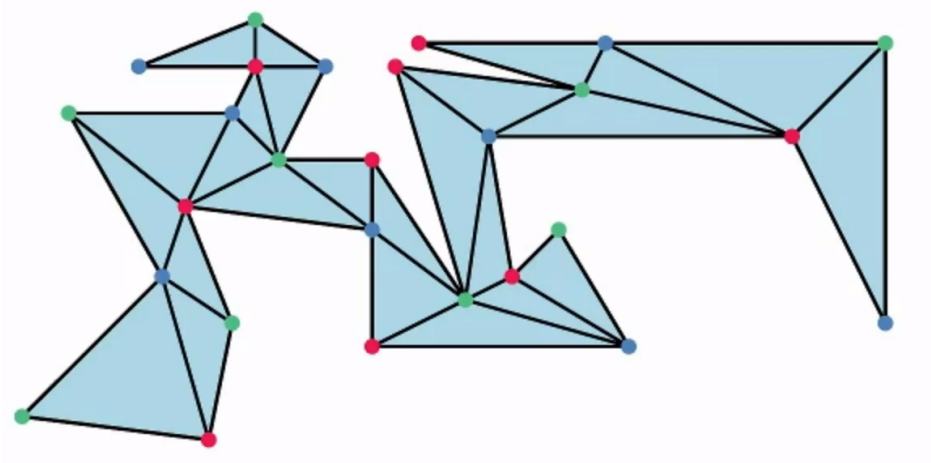
- geometric algorithms
  - Example: Shortest Path inside a polygon.
  - Also used in proving time bounds.
  - Example: Guarding



–

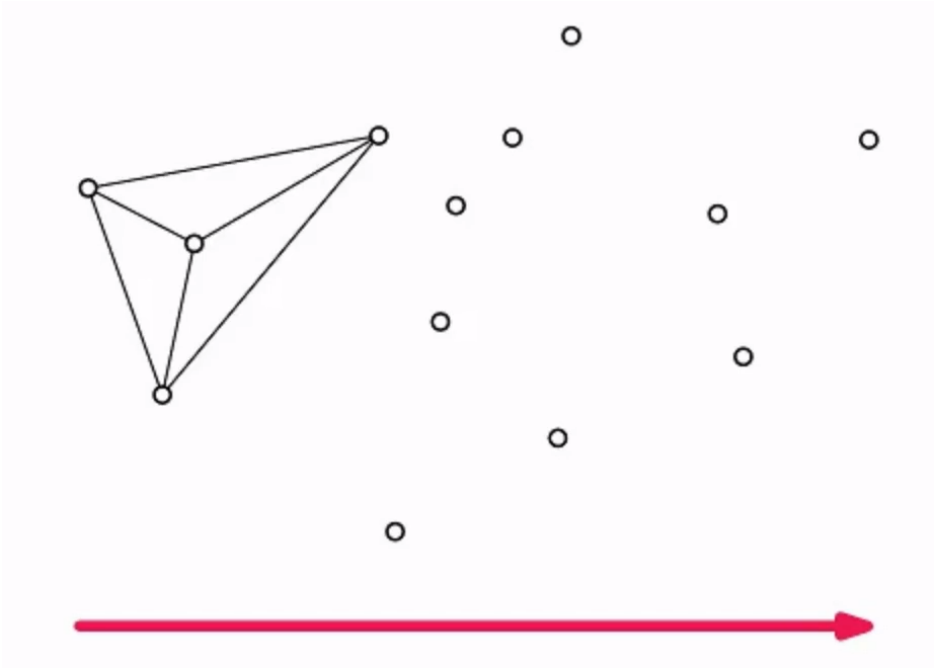
- [[Kombinatorik]]
  - neighbours have different colors

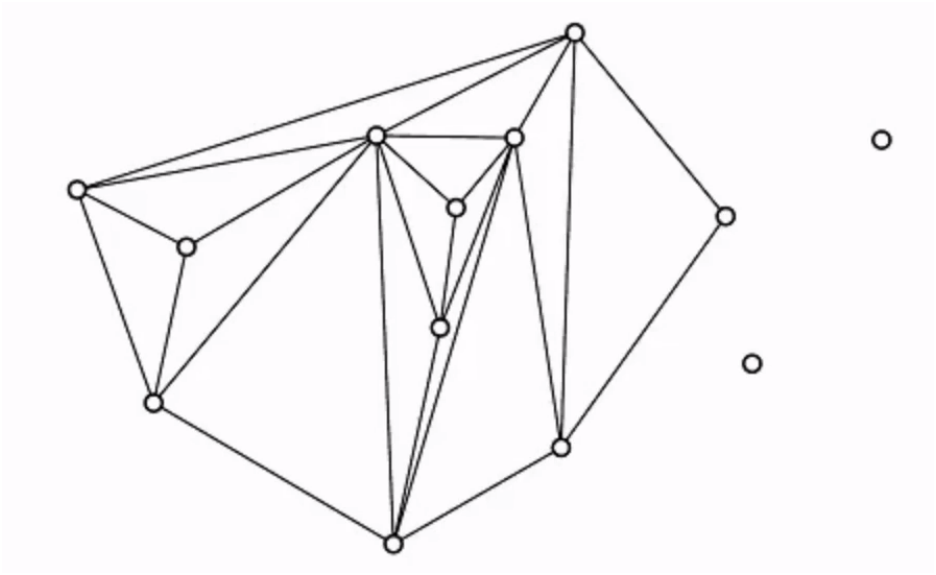
–



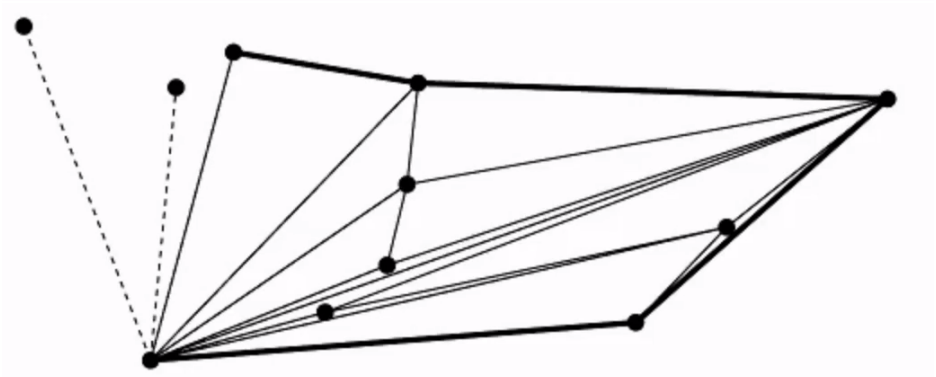
Canonical Triangulation

- combine 3 leftmost points
- add each point sequentially
  - combine this point with each point to the left which it still "sees"



- 

Triangulation with Graham Scan

- similar to creating [[Convex Hull]]
  - add edges to anchor point and to last convex hull vertex
  - don't remove edges
- 
- 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`.

```
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]))
```

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`.

```
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]
```

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

  → in total $O(n \log n)$ time.
- space complexity
  - $O(n)$ in addition to input. Why?
- 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]`.

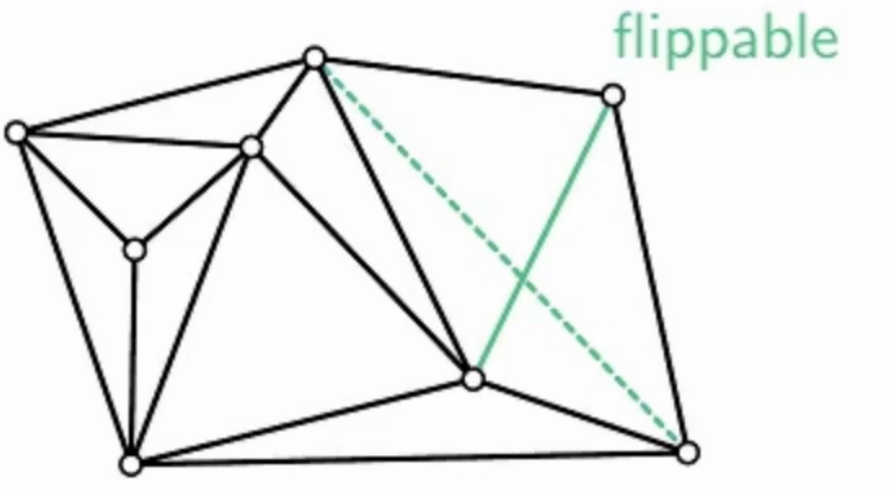
$\Rightarrow$ After the round for `p[N]`, `t` contains all edges of a triangulation for `p[1..N]`.

–

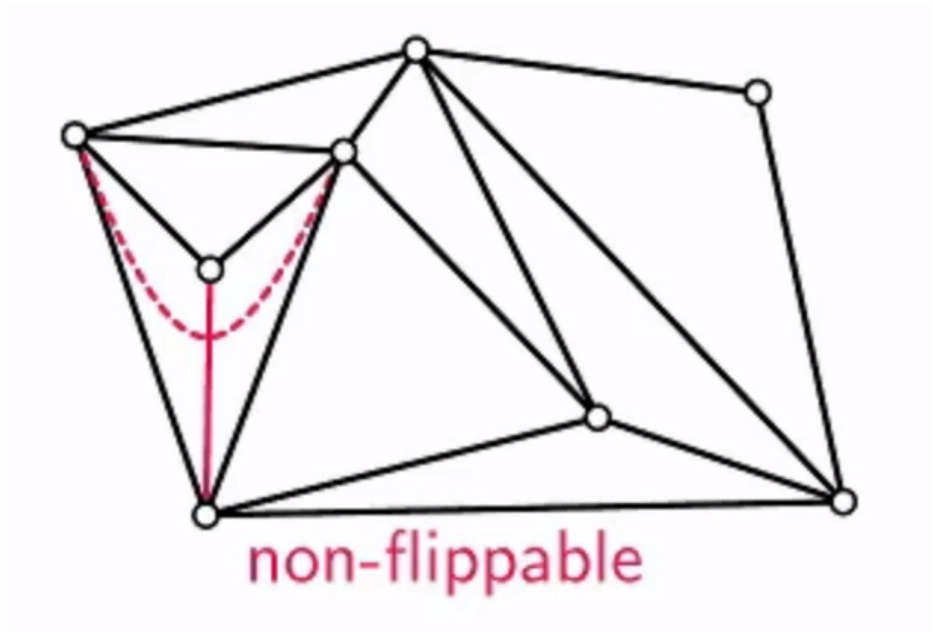
Local Transformation

- rebuild existing triangulation to different triangulation
- edge flips

–



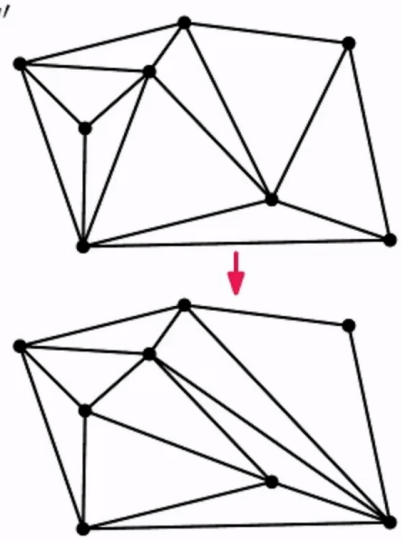
–



- useful for optimization in heuristics

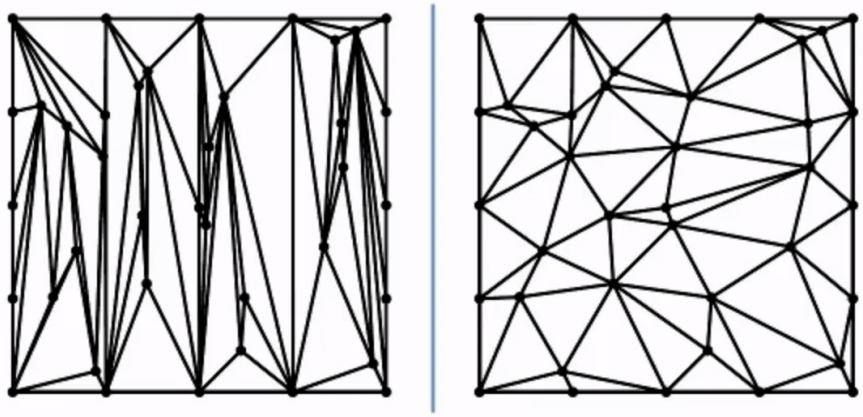
Flip Distance Problem

- **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.



Delaunay Triangulation

- Two copies of the same point set ...



Canonical triangulation Delaunay triangulation

- Useful in practice and theory: Delaunay triangulation
  - obtainable by simple flip rules
  - optimizes several criteria
  - dual to the Voronoi diagram