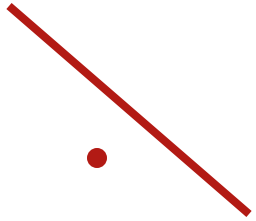
Hulls & triangulations

Georgia Tech College of Computing
Interactive & Intelligent Computing

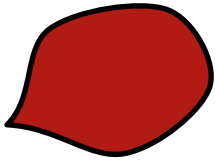
- Convexity
- Convex hull
- Delaunay triangulation
- Voronoi diagram

Attributes of subset of the plane

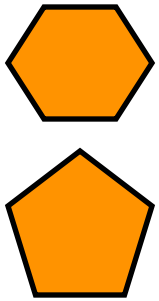
In what follows, A and B are subsets of the plane:



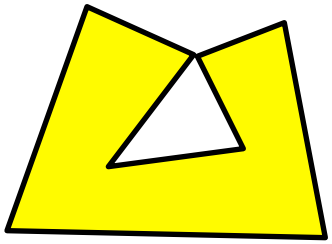
Lower-dimensional



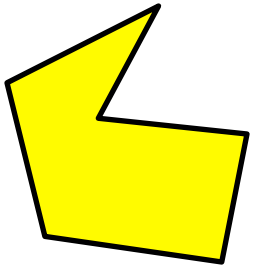
Not a **polygon**



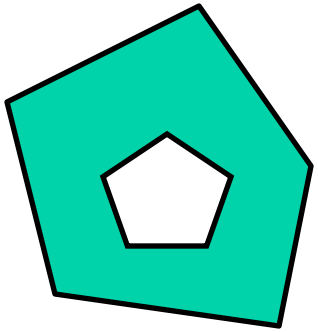
Disconnected (multiple components)



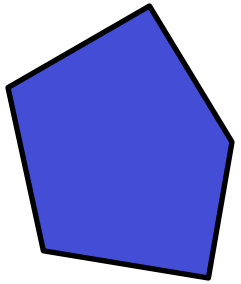
Non-manifold polygon



Simply connected polygon (non-convex)



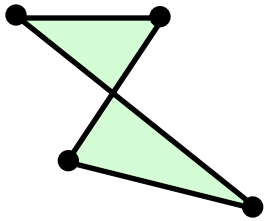
Connected but with one or more **holes**



Convex polygon

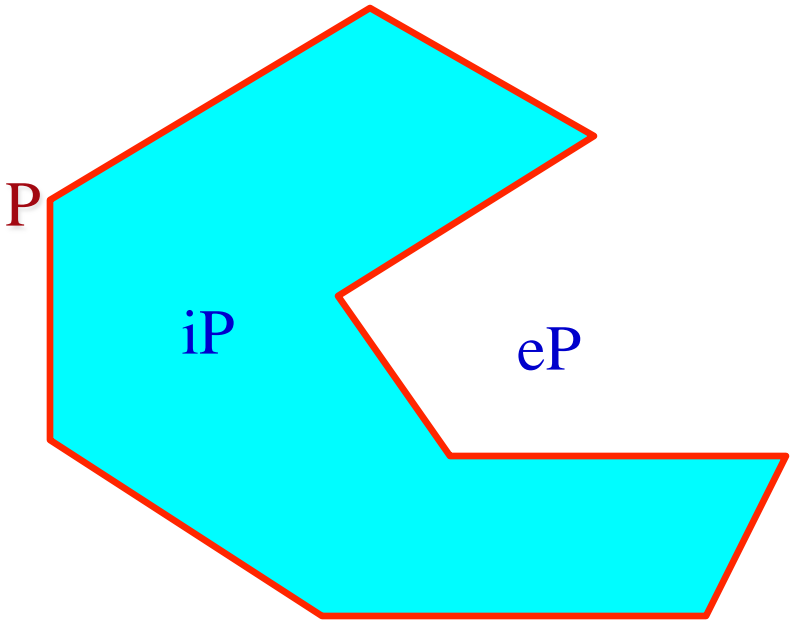
Polyloop, boundary, interior, exterior

When A is a simply connected polygon**.**
Its boundary**, denoted δA, is a polyloop P.**



VALIDITY ASSUMPTION**: We assume that the vertices and edge-interiors of P are pair-wise** disjoint

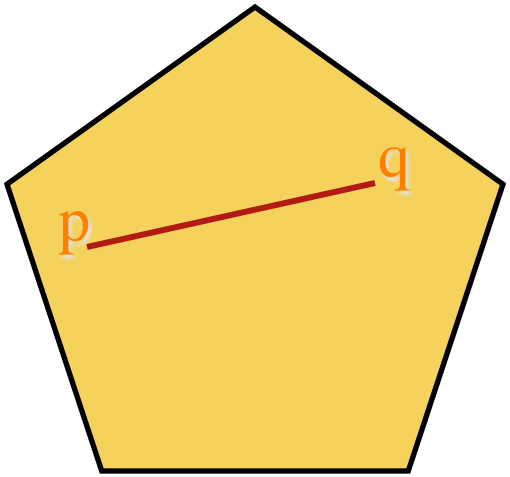
P divides space into 3 sets:
the boundary: P
the interior: iP,
the exterior: eP

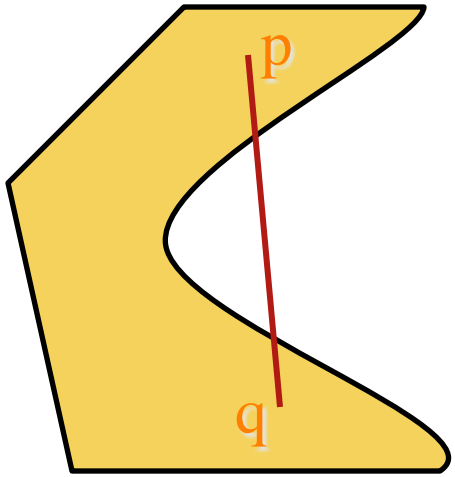


When is a planar set A **convex**?

seg(p,q) is the line segment from p to q

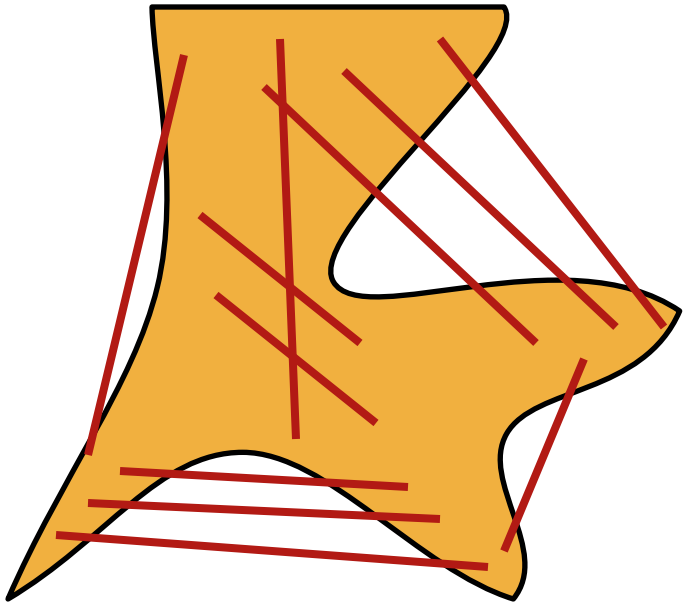
A is convex $\leftrightarrow$ ($\forall p \in A \; \forall q \in A \rightarrow seg(p,q) \subset A$)



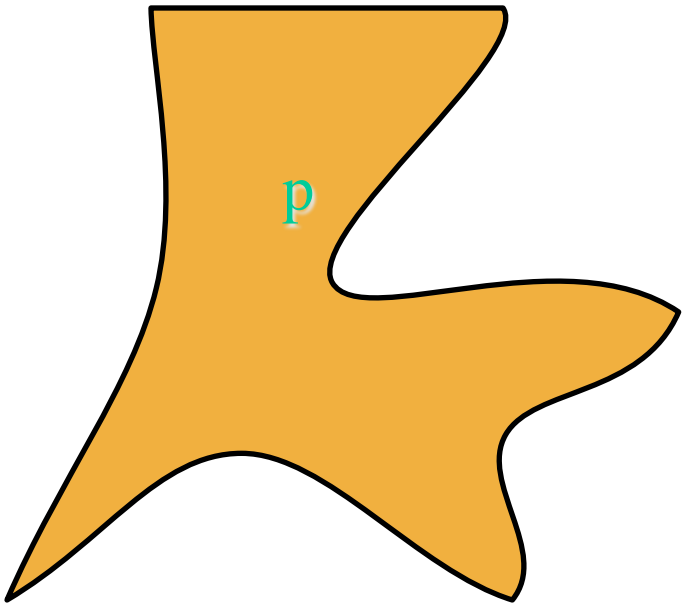


What is the **Stringing** of A?

The "**Stringing**" S(A) of A is the union of all segments seg(p,q) with p∈A and q∈A



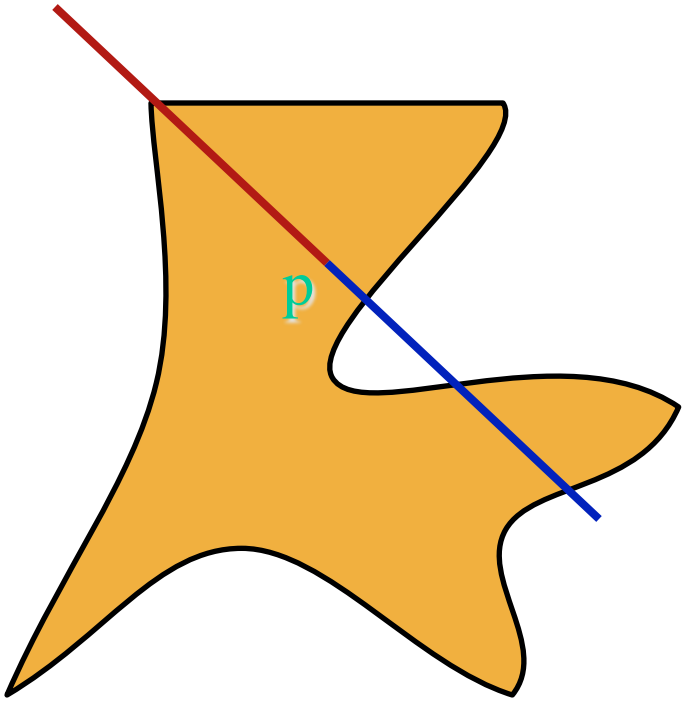
Prove that A⊂S(bA)



Prove that A⊂S(bA)

∀p∈A, p∈seg(p,p)⊂S(A)

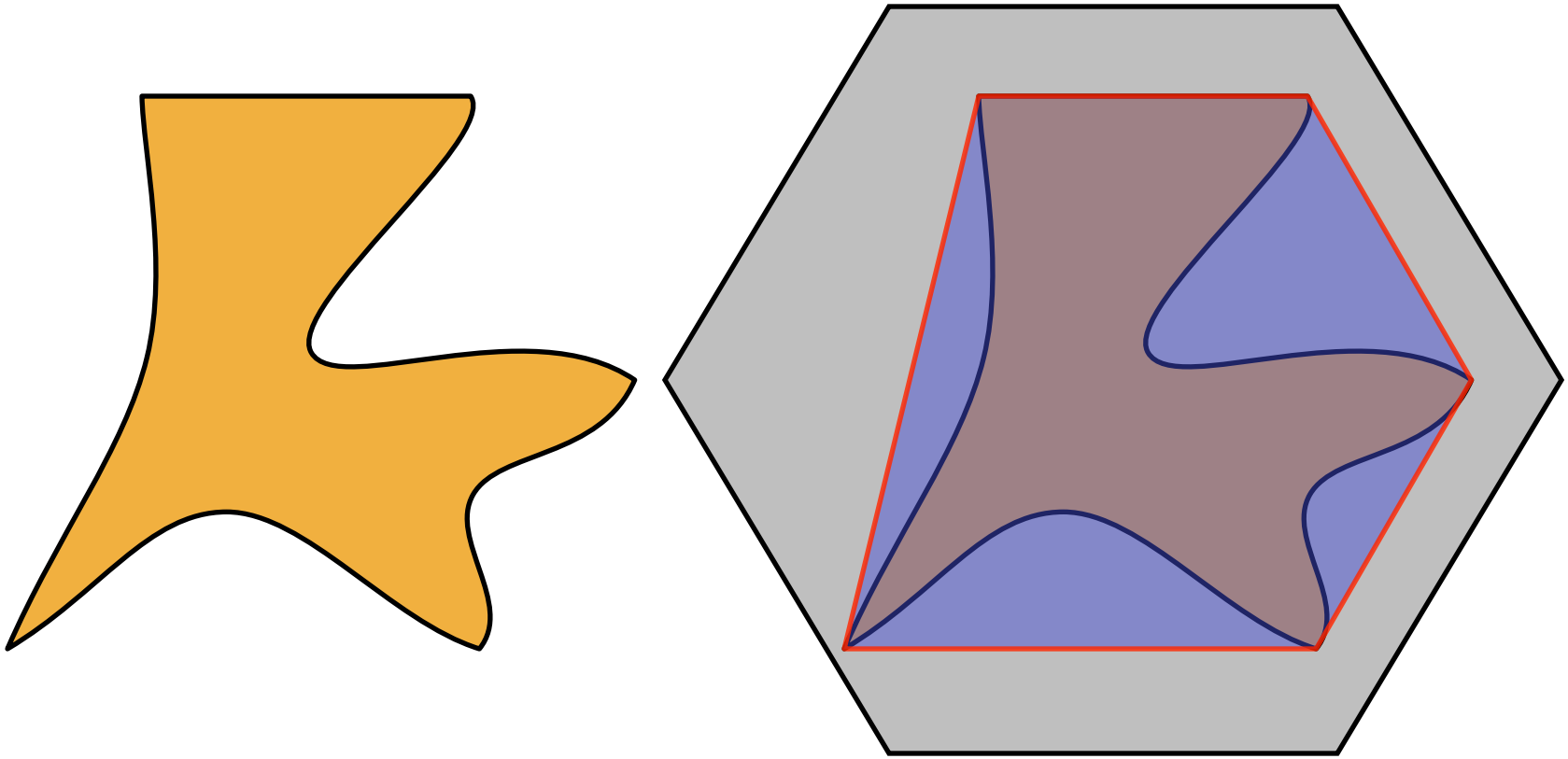
In fact, A⊂S(δA), since opposite rays from each point p of the interior of ∂ δA must each exit A at a point of bA



What is the **convex hull** of a set A?

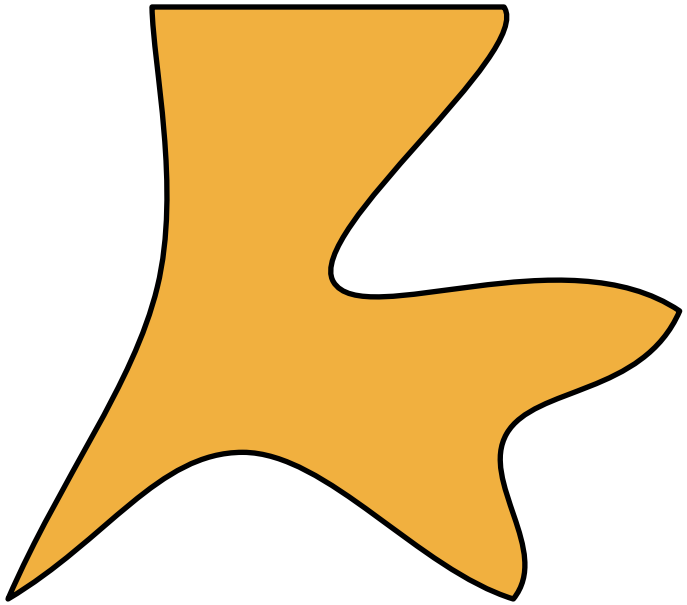
The convex hull H(A) is the intersection of all convex sets that contain A.

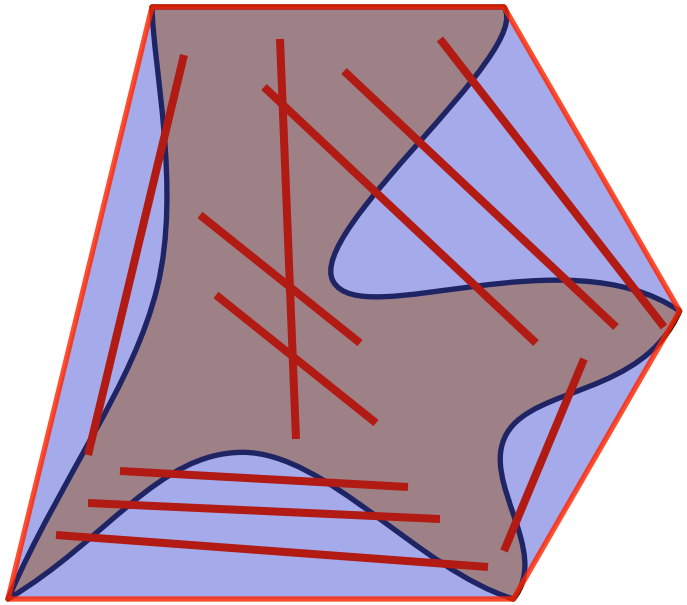
(It is the smallest convex set containing A)



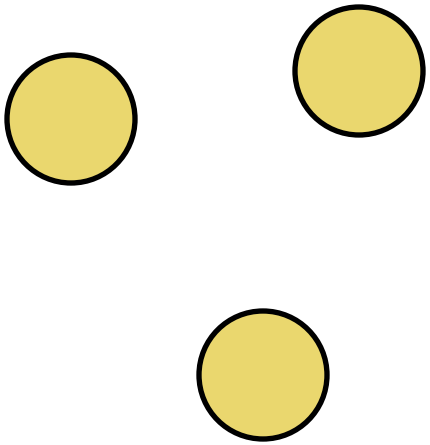
Conjecture: S(A)=H(A)

Prove, disprove…

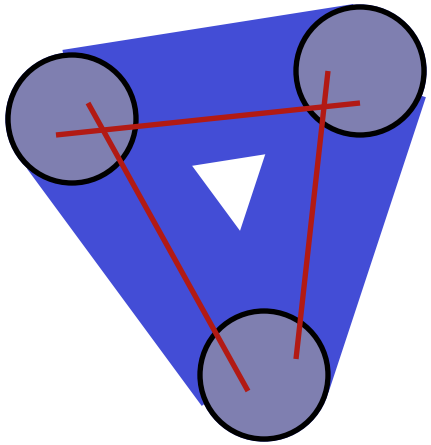




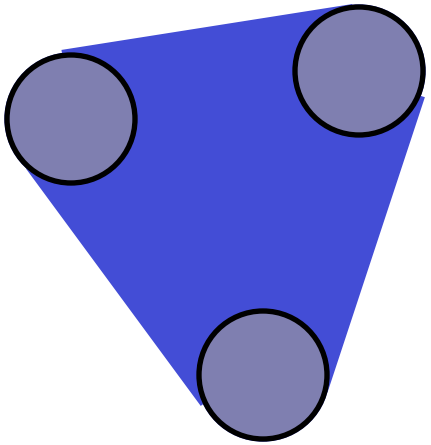
Counter example where S(A)≠H(A)



S



$\cup$ segment(p,q) for all p∈S and q∈S



H(S)

Other properties/conjectures?

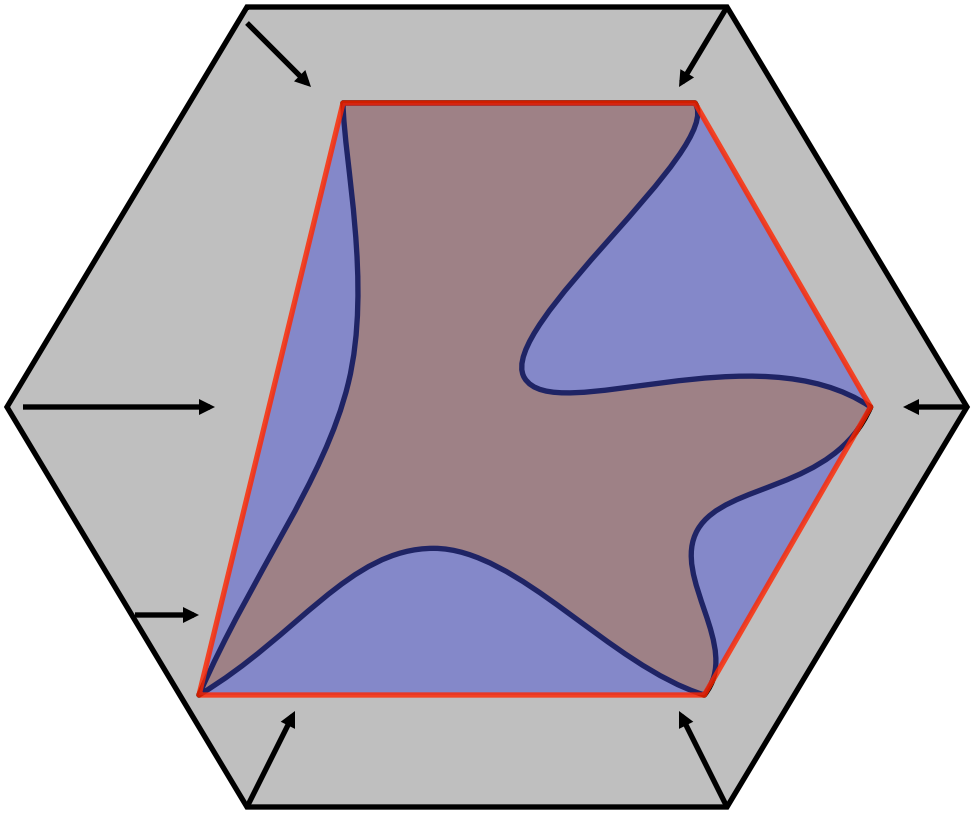
Prove/disprove:

- $S(A) \subset H(A)$
- A convex $\rightarrow S(A) = H(A)$
- $H(A)$ = union of all triangles with their 3 vertices in A

H(A) is a **Tightening**?

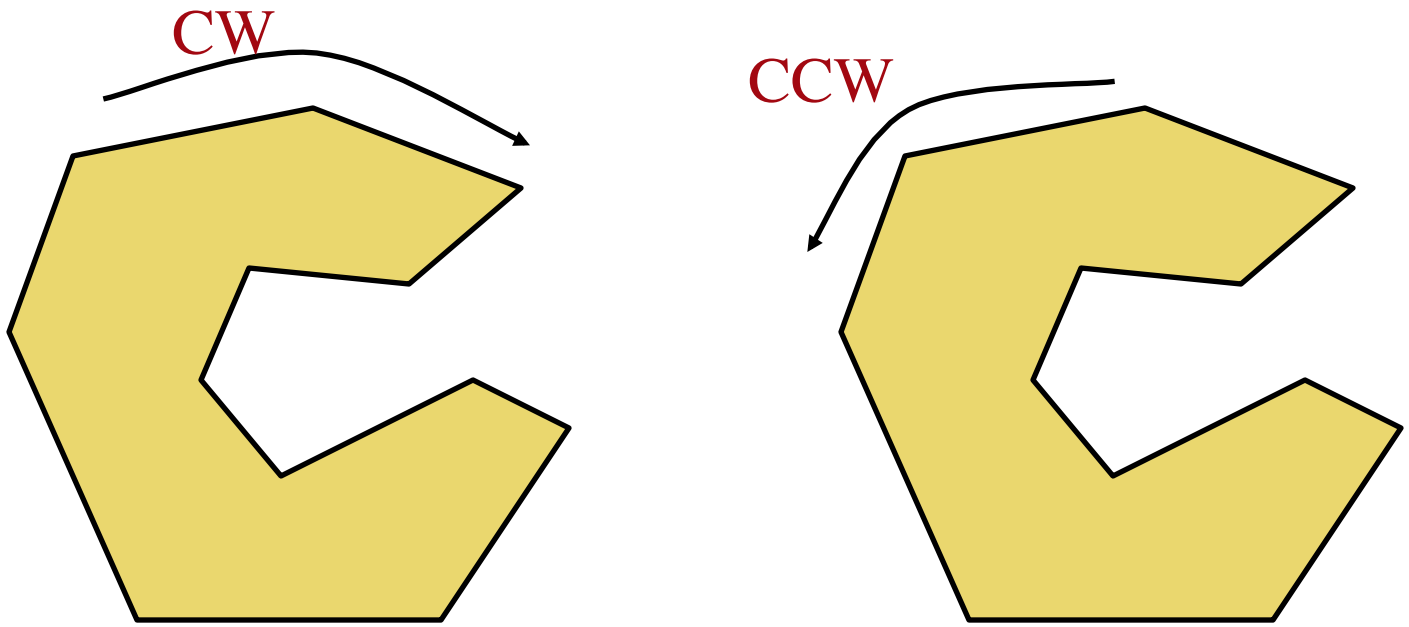
H(A) is the tightening of a polyloop P that contains A in its interior

Tightening shrinks P without penetrating in A



What is the **orientation** of a polyloop

P is oriented clockwise (CW) or counterclockwise (CCW)

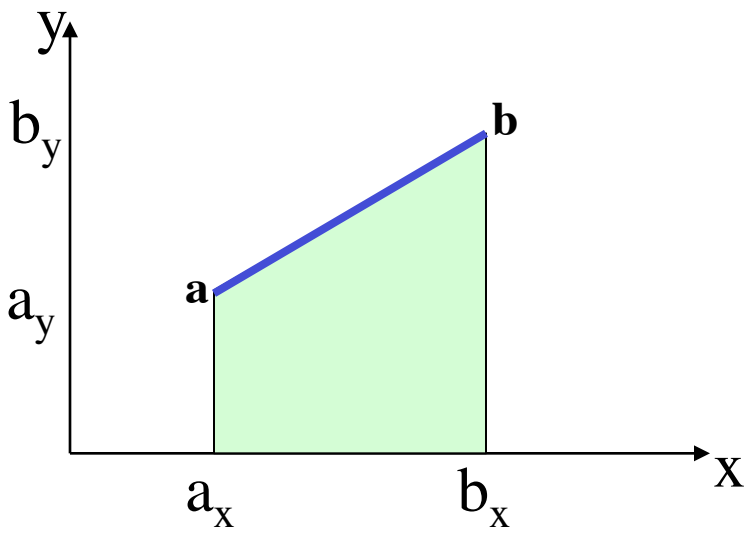


Often orientation is implicitly encoded in the order in which the vertices are stored.

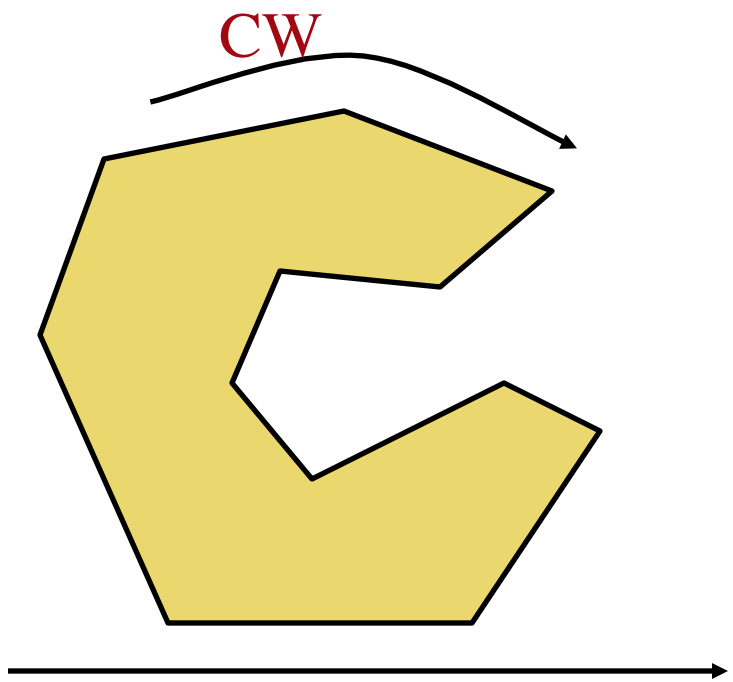
How to compute the **orientation**?

Compute the signed area aP of iP as the sum of signed areas of trapezes between each oriented edge and its orthogonal projection (shadow) on the x-axis.

If aP>0, the P is CW. Else, reveres the orientation (vertex order).

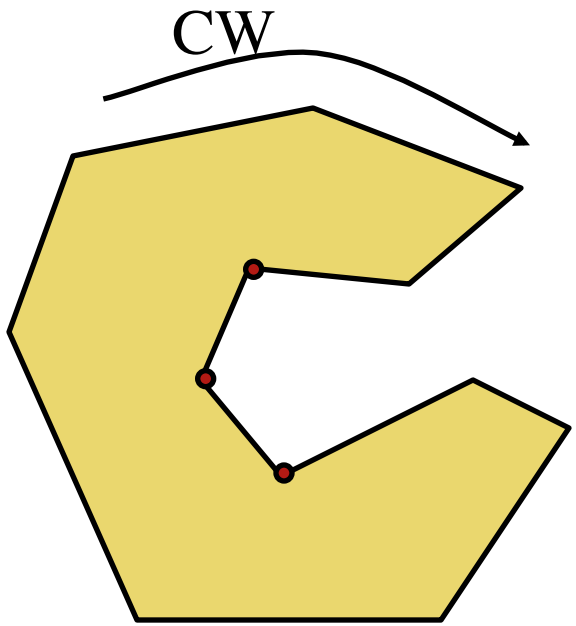


$$\text{area}(\mathbf{a},\mathbf{b}) := (a_y + b_y)(b_x - a_x)/2$$



When is a vertex **concave/convex**?

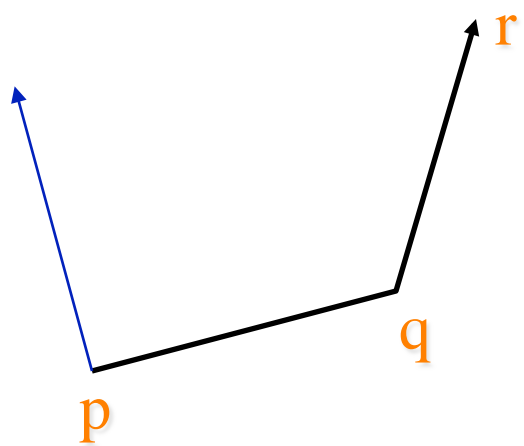
Vertex q a CW polyloop is concave when P makes a left turn at q.
If is convex otherwise.



When is a vertex a **left turn?**

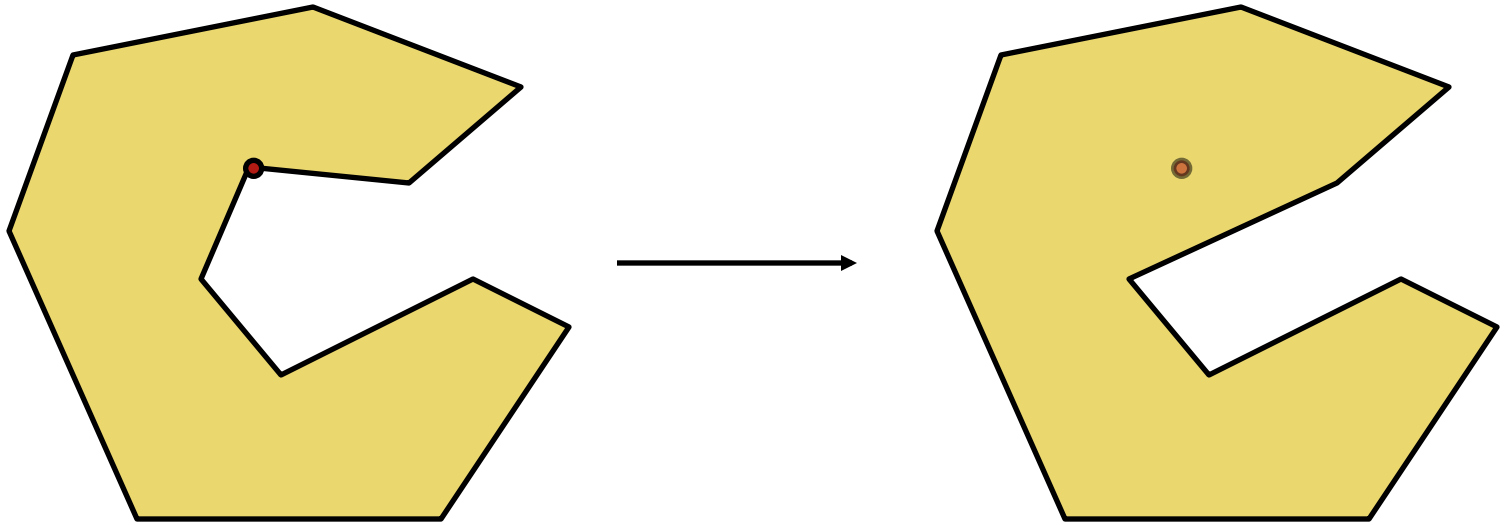
Consider a sequence of 3 vertices (p, q, r) in a polyloop P.

Vertex q is a left turn, written leftTurn(p,q,r), when



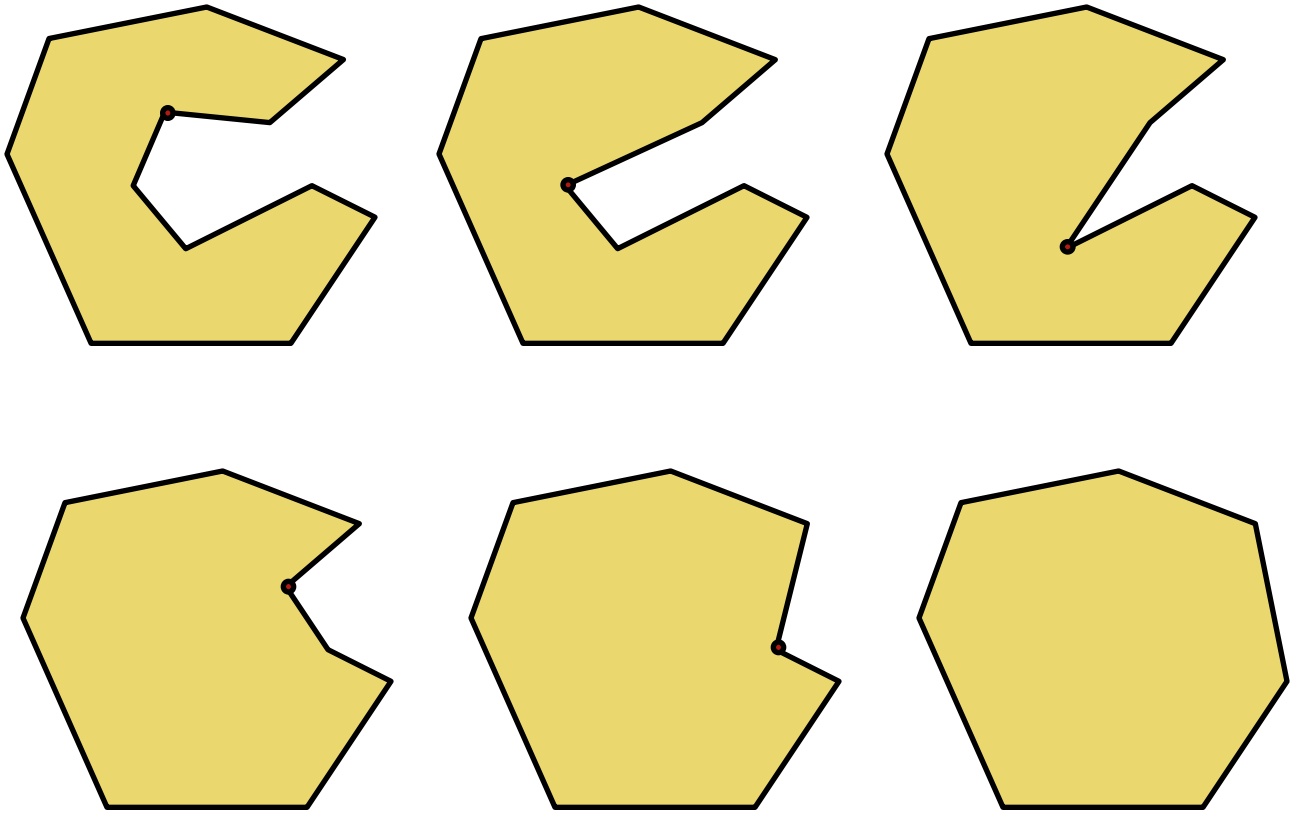
What is the **decimation** of a vertex?

Delete **the vertex from P**



Concave Vertex Decimation: CVD(P)

The CVD algorithm keeps finding and decimating the left-turn vertices of a CW polyloop P until none are left

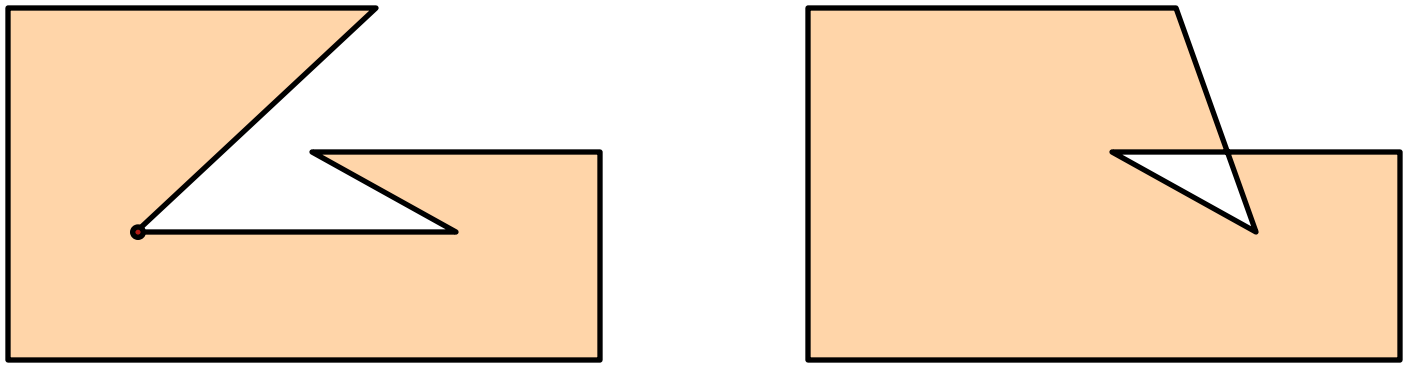


Conjecture: CVD(P) computes H(P)

Prove/disprove….

Counterexample

CVD may lead to self-intersecting polyloops

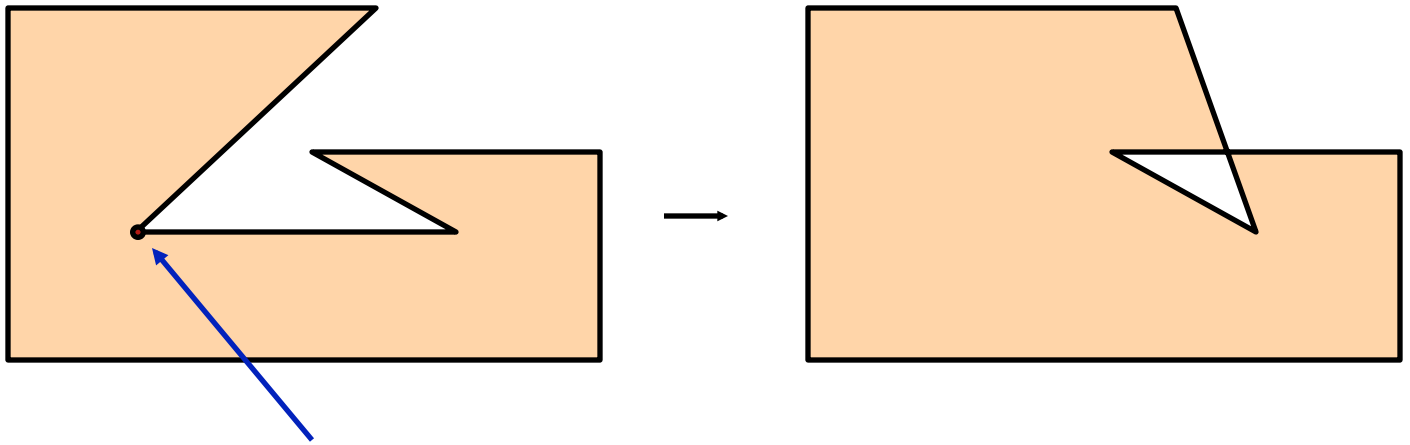


Now, all vertices are right turns, but we do not have H(P). In fact the polyloop is invalid (self-intersects).

Can you fix the CVD algorithm?

Preserve validity!

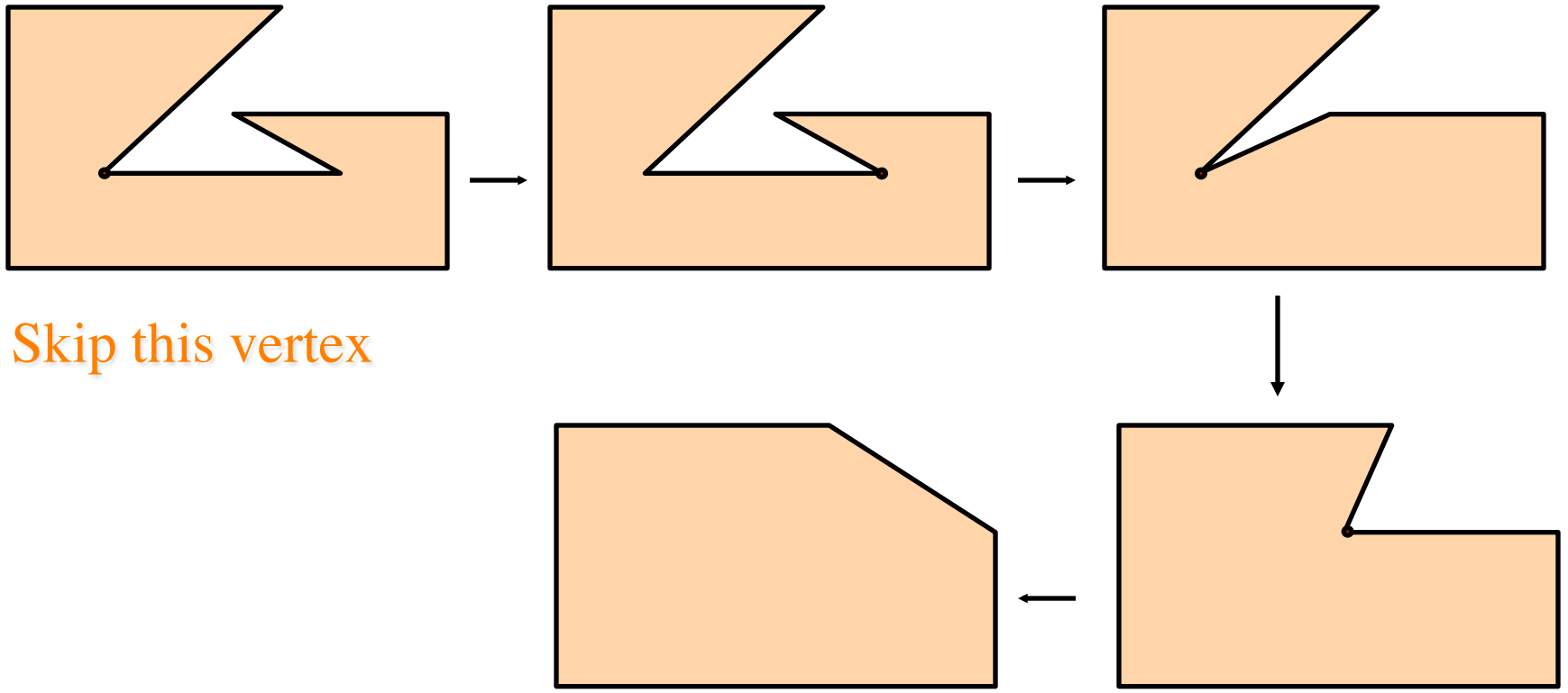
- Only do decimations of concave vertices when they preserve validity of the polyloop



This vertex should not be decimated

Example of the fixed the CVD algorithm

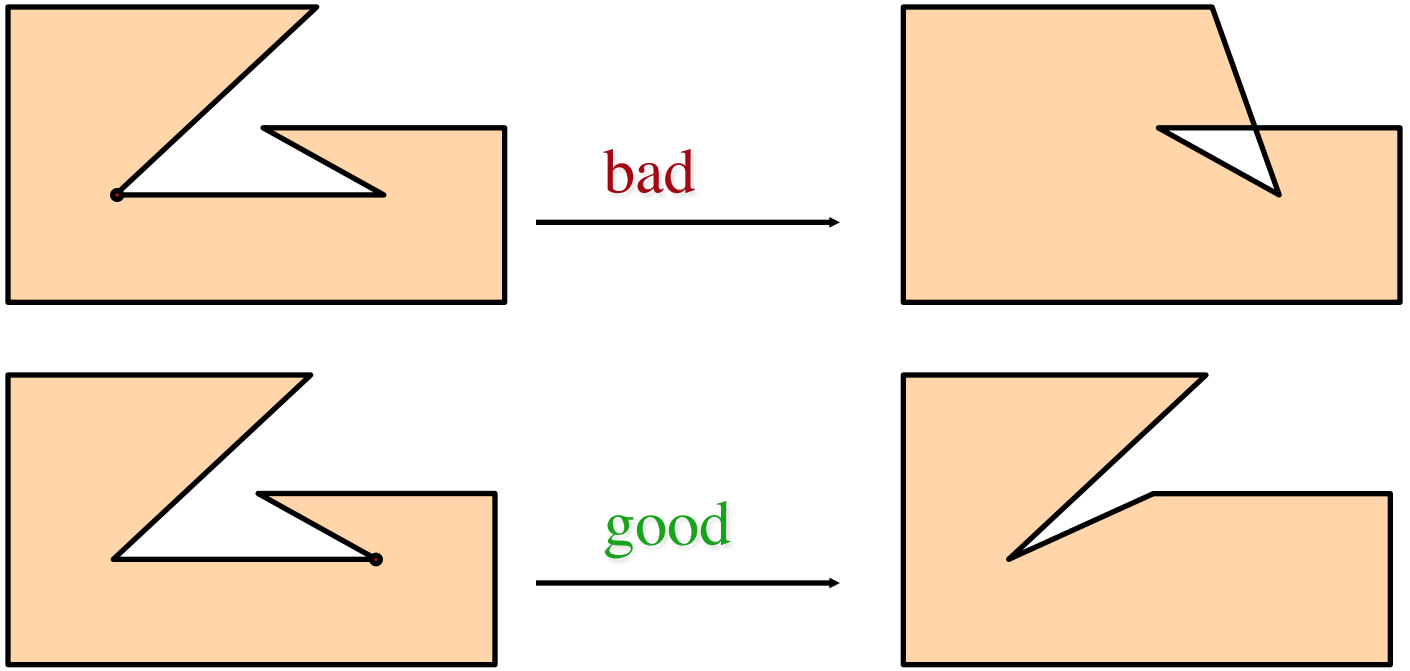
Only decimate concave vertices when the **validity** of the polyloop is preserved



Skip this vertex

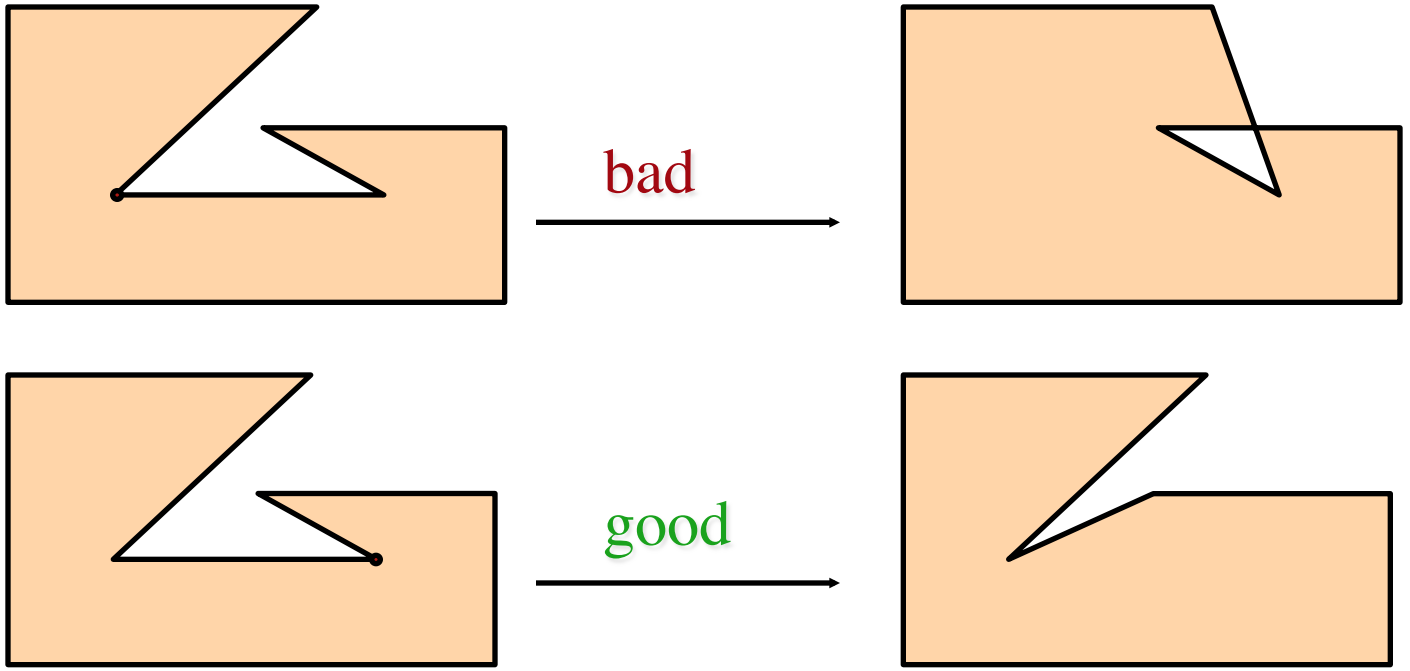
How to test the validity of a decimation?

What is different between these two decimations?



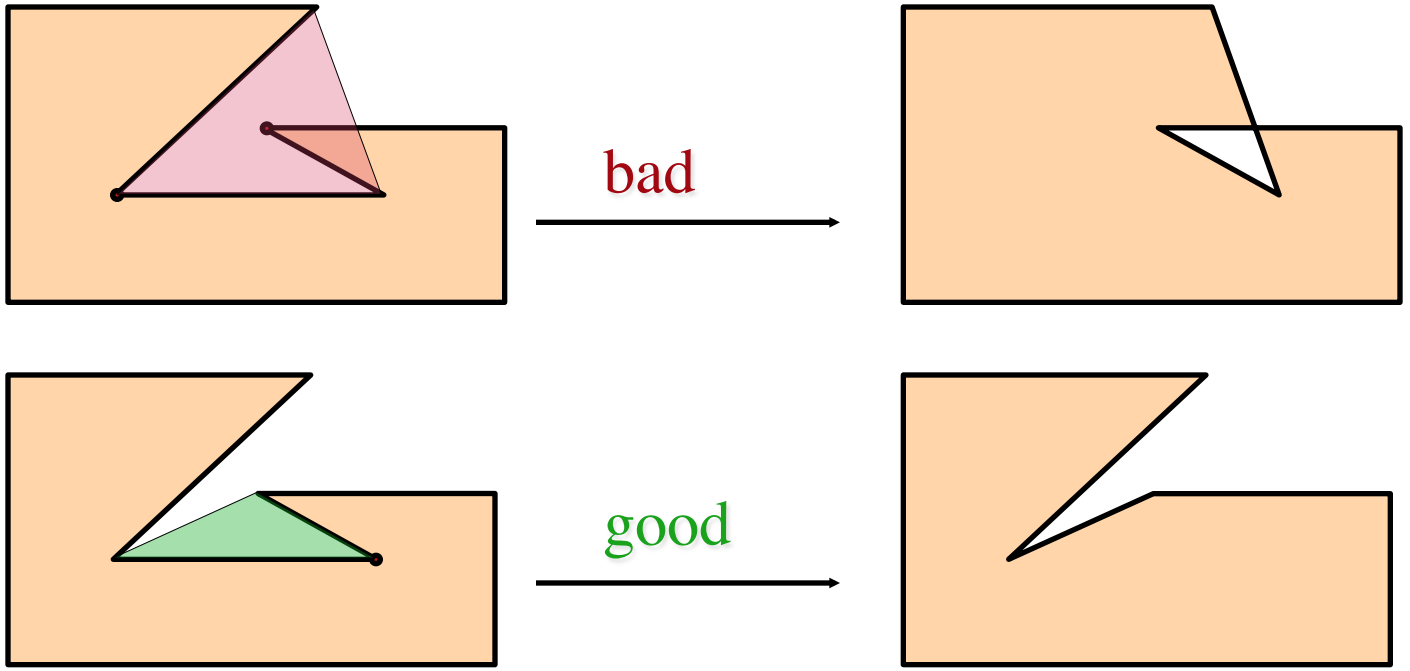
How to test the validity of a decimation?

What is different between these two decimations?



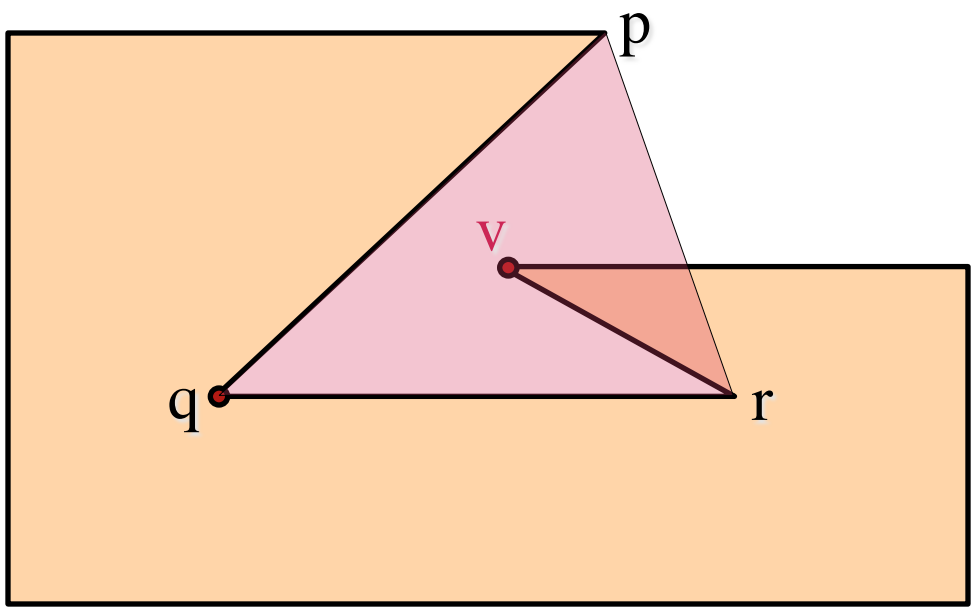
Other vertex in swept triangle!

What is different between these two decimations?



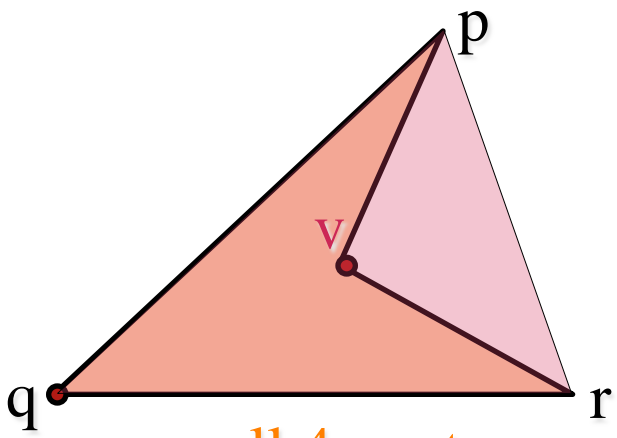
How to implement point-in-triangle?

- How to test whether point v is in triangle (p,q,r)?

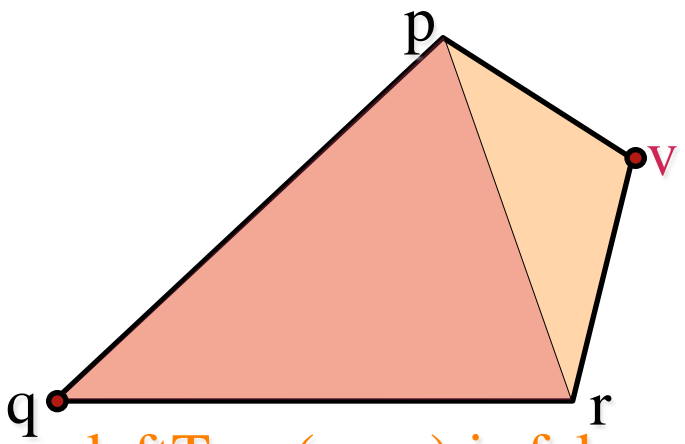


Use the left-turn test!

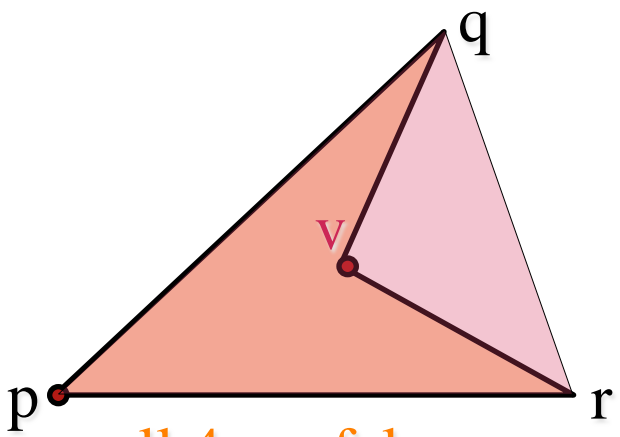
If leftTurn(p,q,r), leftTurn(p,q,v), leftTurn(q,r,v), leftTurn(r,p,v) are all equal, then v is in the triangle.



all 4 are true

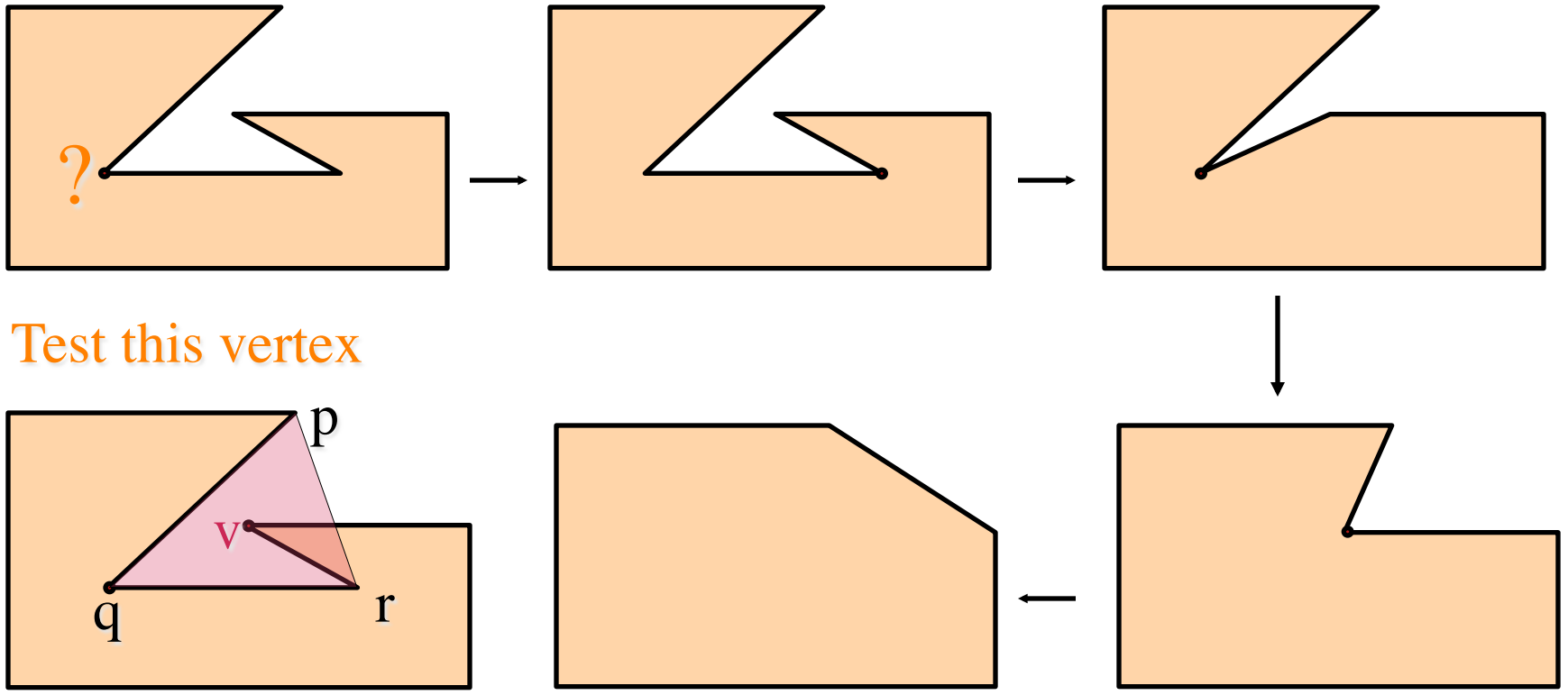


leftTurn(r,p,v) is false

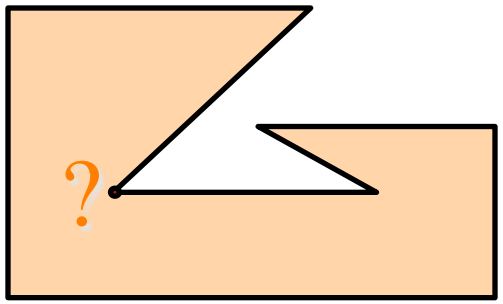


all 4 are false

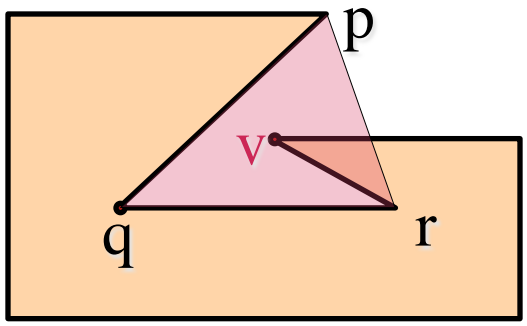
What is the complexity of the fixed CVD?



What is the cost of testing a vertex?

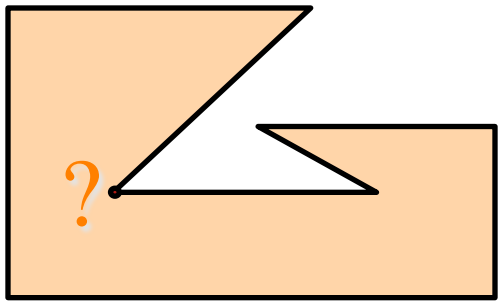


Test this vertex

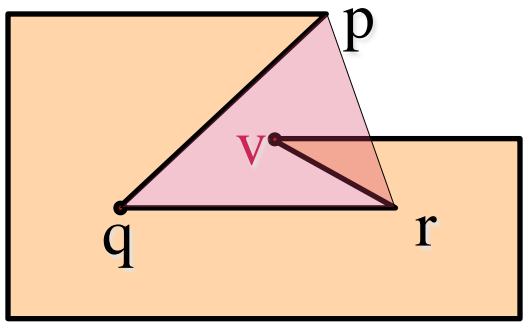


What is the cost of testing a vertex?

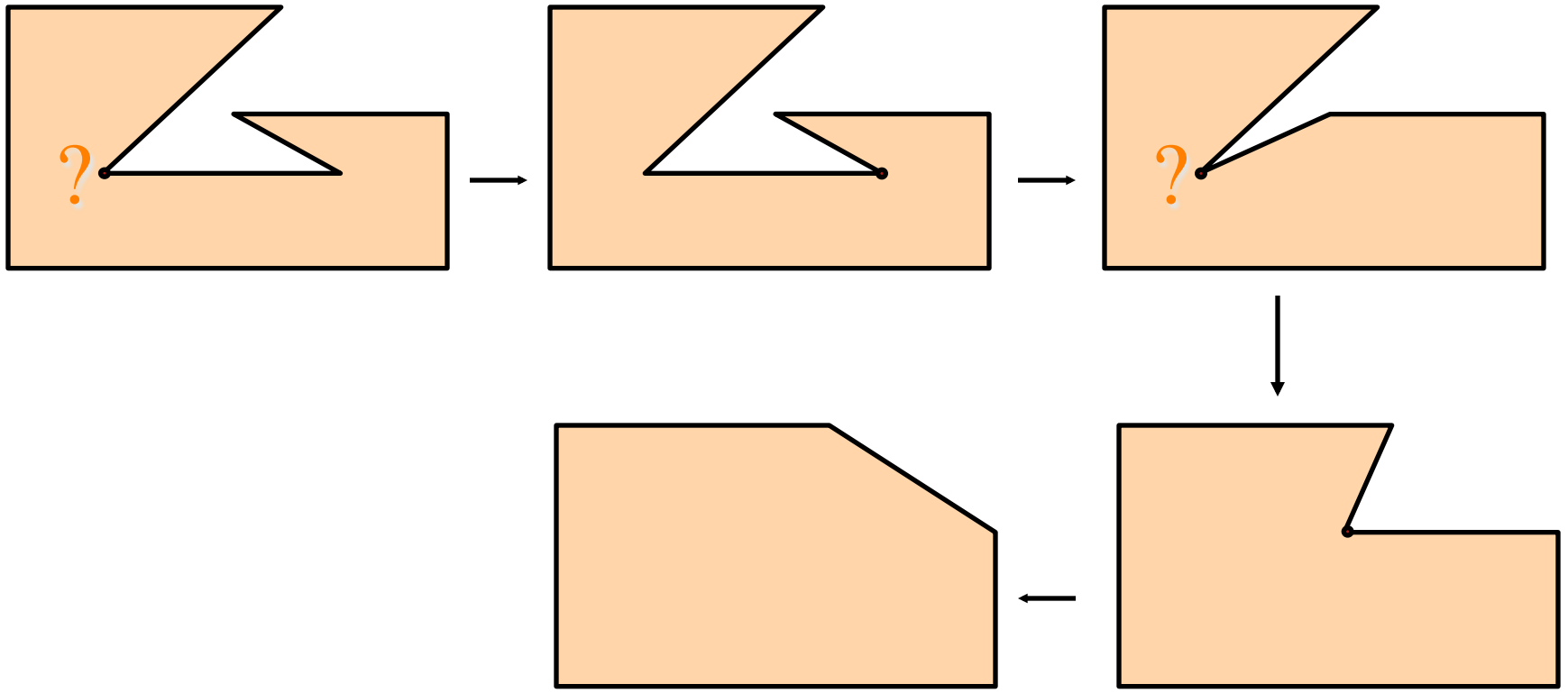
- Go through all the remaining vertices and perform a constant cost test: O(n)



Test this vertex

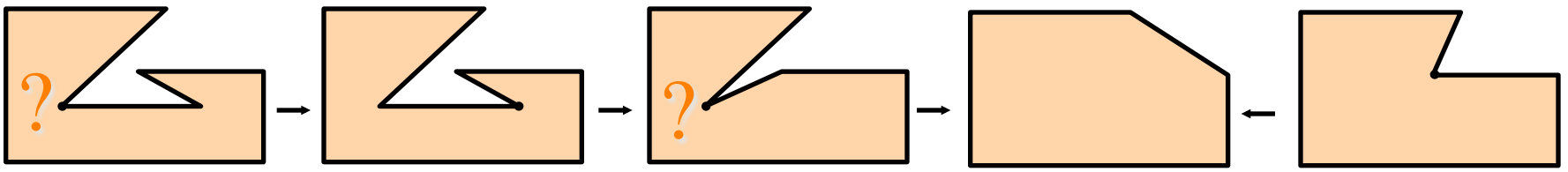


How many times is a vertex tested?



How many times is a vertex tested?

- Test each vertex once and put concave vertices in a queue.
- The h vertices on H(P) never become concave. They will not be tested again.
- The initial set of concave vertices will be tested at least once.
- At least one of them will be decimated.
- You only need to retest the vertex preceding a decimation.
- So the number of in-triangle tests is < 2n–h… O(n)

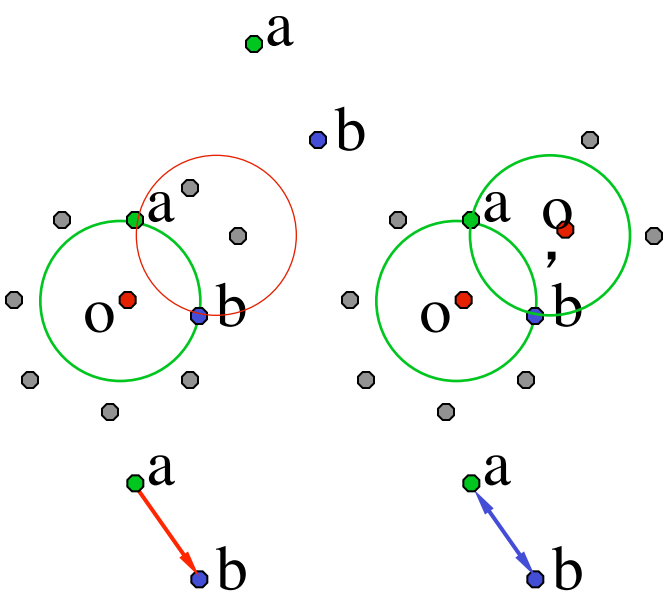


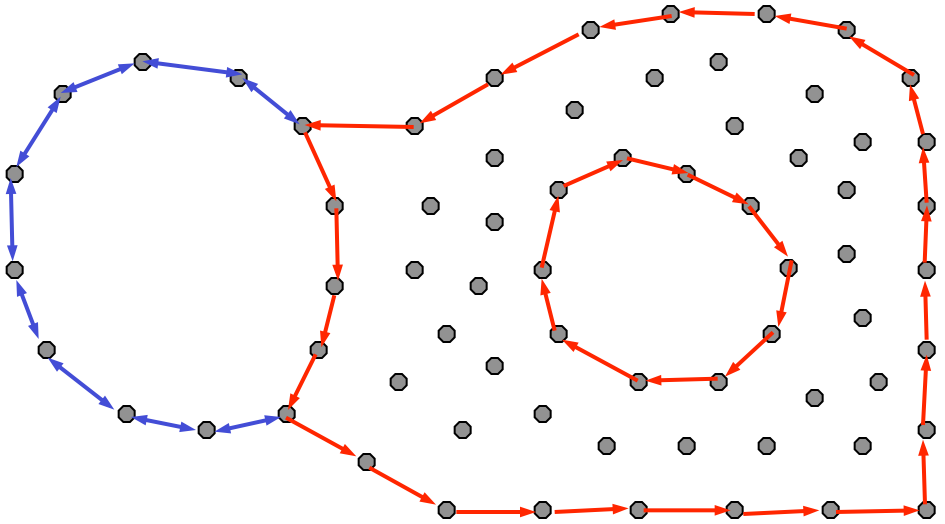
What is the total complexity?

- O(n) tests
- Each O(n)

Analysis of point-clouds in 2D

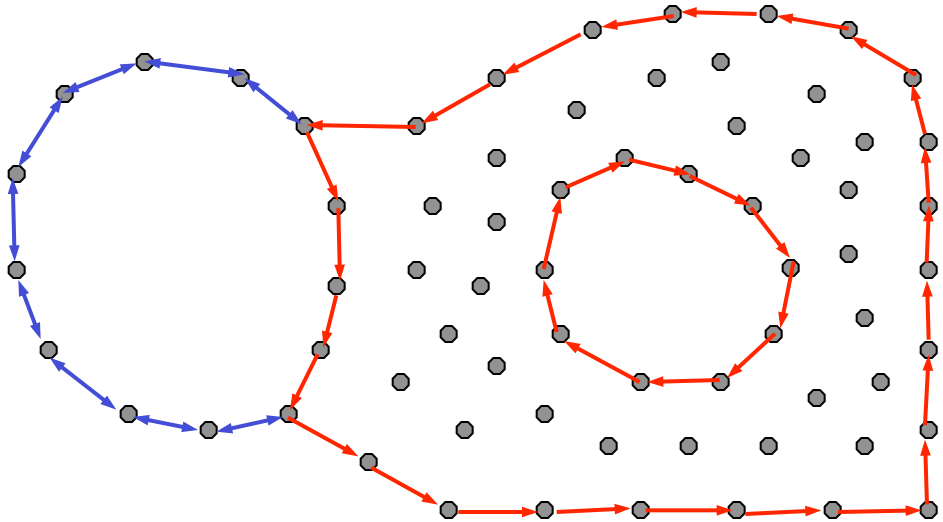
- Pick a radius r (from statistics of average distance to nearest point)
- For each ordered pair of points a and b such that ||ab||<2r, find the positions o of the center of a circle of radius r through a and b such that b is to the right of a as seen from o.
- If no other point lies in the circle circ(o,r), then create the oriented edge (a,b)





Interpreting the loops in 2D

- Chains of pairs of edges with opposite orientations (blue) form dangling polygonal curves that are adjacent to the exterior (empty space) on both sides.
- Chains of (red) edges that do not have an opposite edge form loops. They are adjacent to the exterior on their right and to interior on their left.
- Each components of the interior is bounded by one or more loops.
  - It may have holes



Delaunay & Voronoi

Georgia Tech College of Computing
Interactive & Intelligent Computing

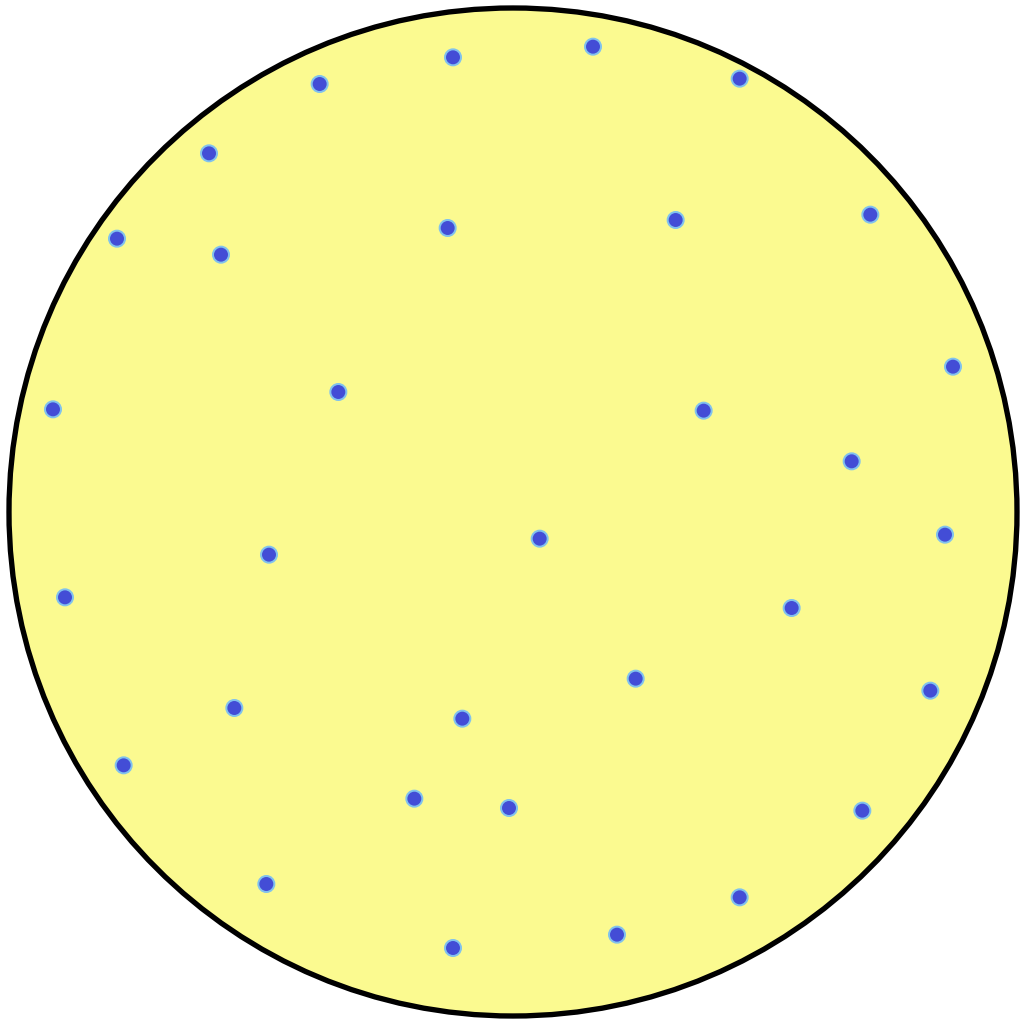
- How to find furthest place from a set of point sites?
- How to compute the circumcenter of 3 points?
- What is a planar triangulation of a set of sites?
- What is a Delaunay triangulation of a set of sites?
- How to compute a Delaunay triangulation?
- What is the asymptotic complexity of computing a Delaunay triangulation?
- What is the largest number of edges and triangles in a Delaunay triangulation of n sites?
- How does having a Delaunay triangulation reduces the cost of finding the closest pair?
- What practical problems may be solved by computing a Delaunay triangulation?

Delaunay & Voronoi

- What is the Voronoi region of a site?
- What properties do Voronoi regions have?
- What are the Voronoi points of a set of sites?
- What practical problems may be solved by computing a Voronoi diagram of a set of sites?
- What is the correspondence (duality) between Voronoi and Delaunay structures?
- Explain how to update the Voronoi diagram when a new site is inserted.
- What are the natural coordinates of a site?

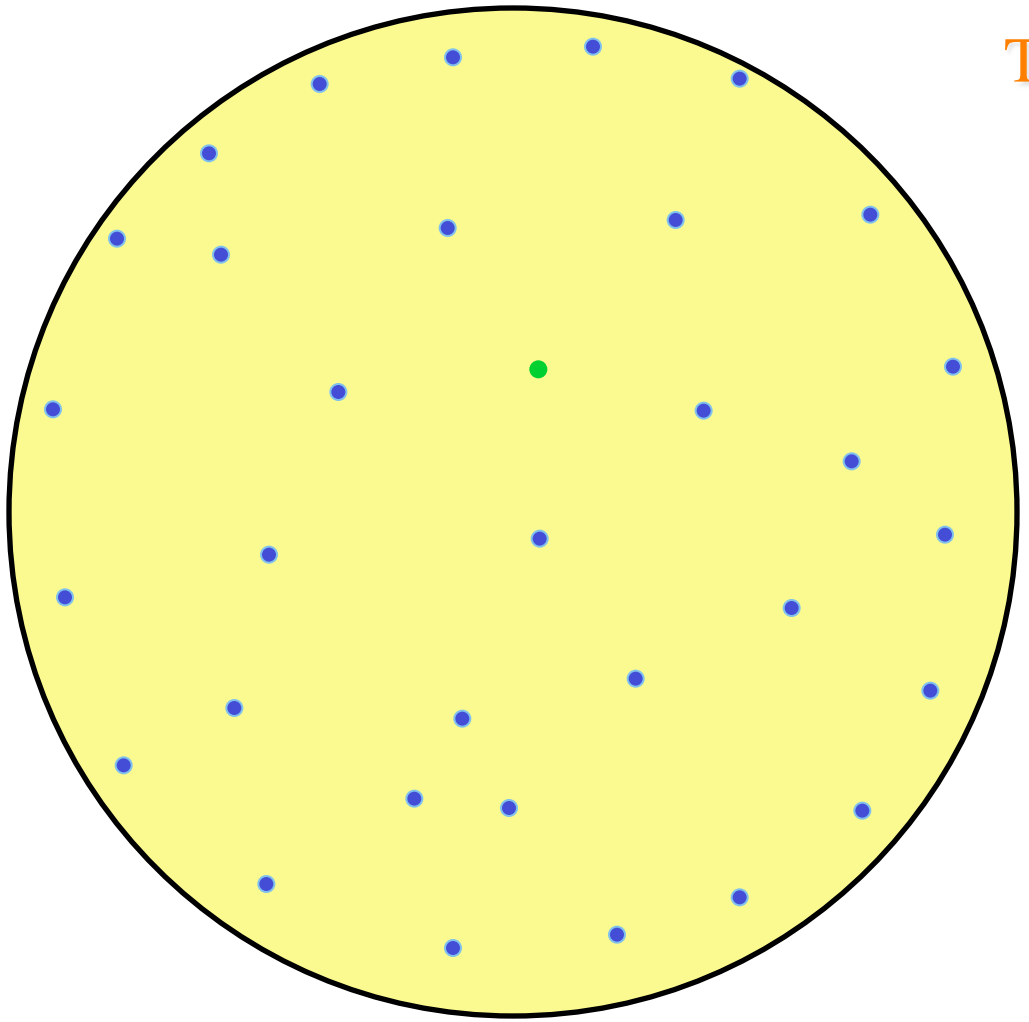
Find the place furthest from nuclear plants

Find the point in the disk that is the furthest from all blue dots



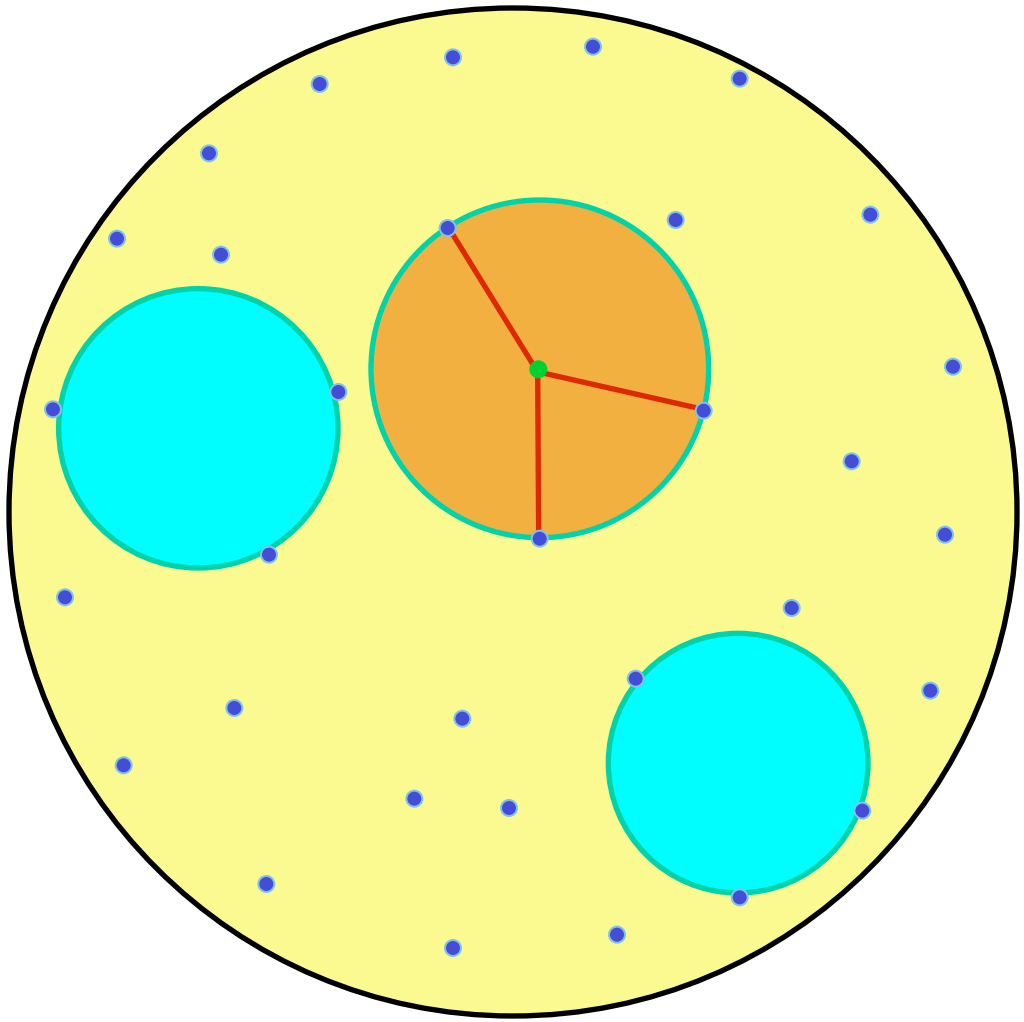
The best place is …

The green **dot**. Find an algorithm for computing it.



Teams of 2.

Center of largest disk that fits between points



Algorithm for best place to live

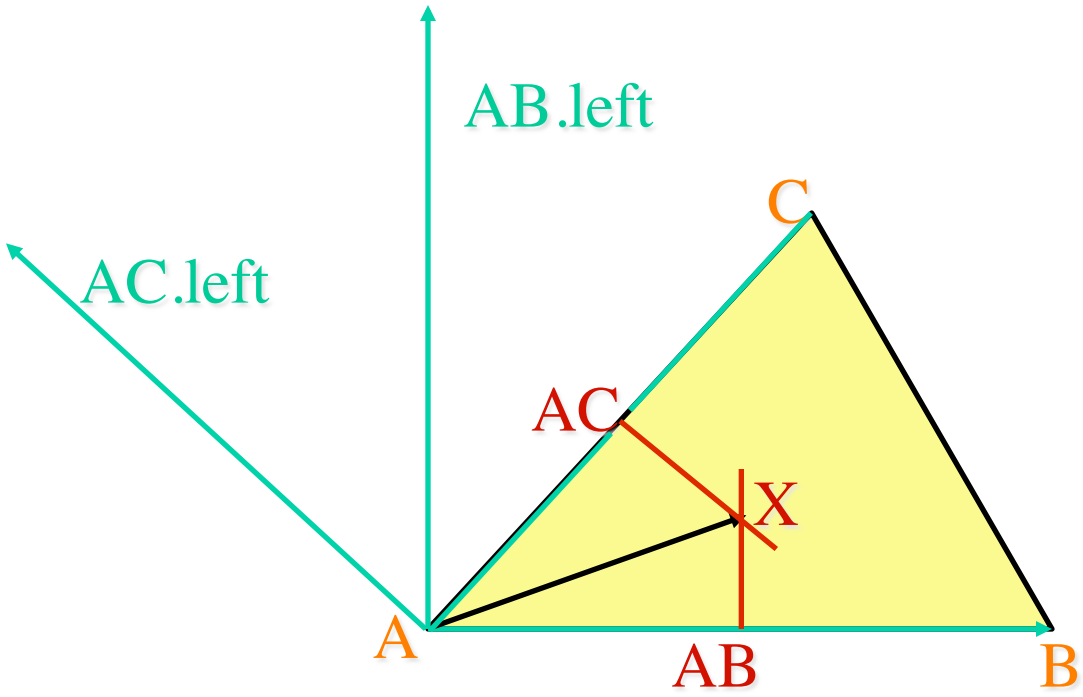
```
max=0;
foreach triplets of sites A, B, C {
    (O,r) = circle circumscribing triangle (A,B,C)
    found = false;
    foreach other vertex D {if (||OD||<r) {found=true;};};
    if (!found) {if (r>max) {bestO=O; max=r;};
    }
return (O);
```

Complexity?

Circumcenter

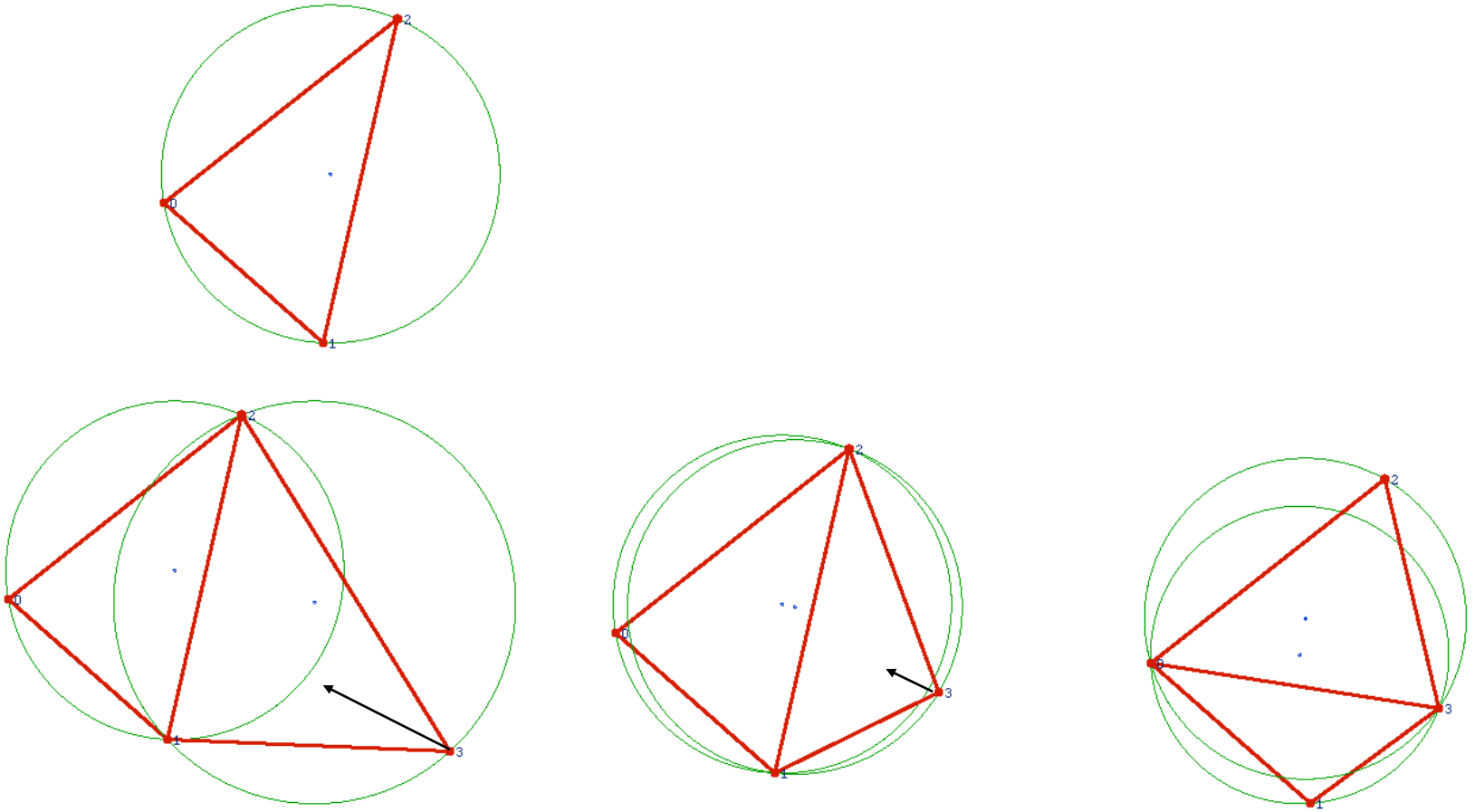
```
pt centerCC (pt A, pt B, pt C) {    // circumcenter to triangle (A,B,C)
  vec AB =  A.vecTo(B);
  float ab2 = dot(AB,AB);
  vec AC =  A.vecTo(C); AC.left();
  float ac2 = dot(AC,AC);
  float d = 2*dot(AB,AC);
  AB.left();
  AB.back(); AB.mul(ac2);
  AC.mul(ab2);
  AB.add(AC);
  AB.div(d);
  pt X =  A.makeCopy();
  X.addVec(AB);
  return(X);
  };
```

$$\begin{cases} 2AB \bullet AX = AB \bullet AB \\ 2AC \bullet AX = AC \bullet AC \end{cases}$$

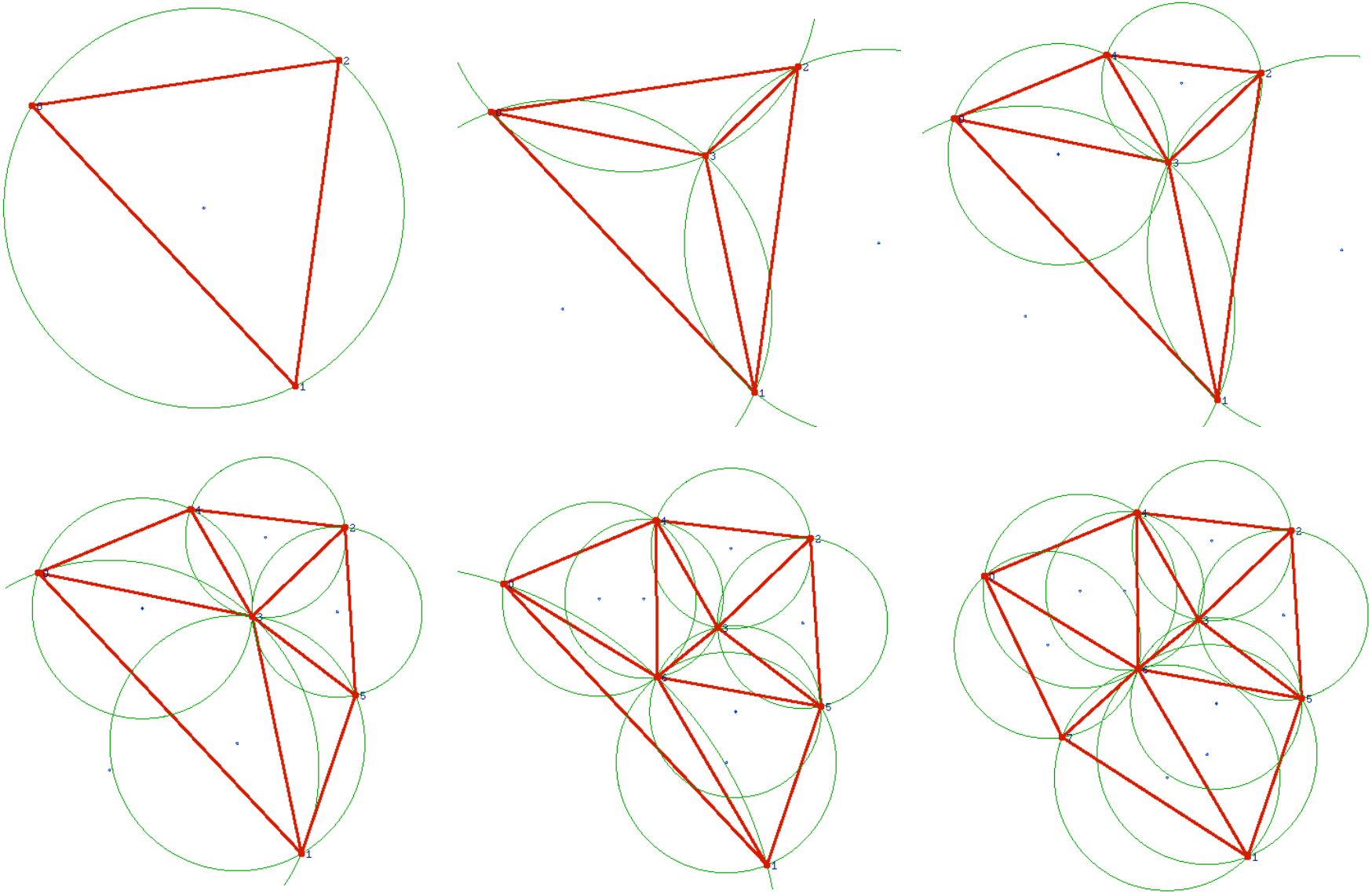


Delaunay triangles

3 sites (vertices) form a **Delaunay triangle** if their circumscribing circle does not contain any other site.

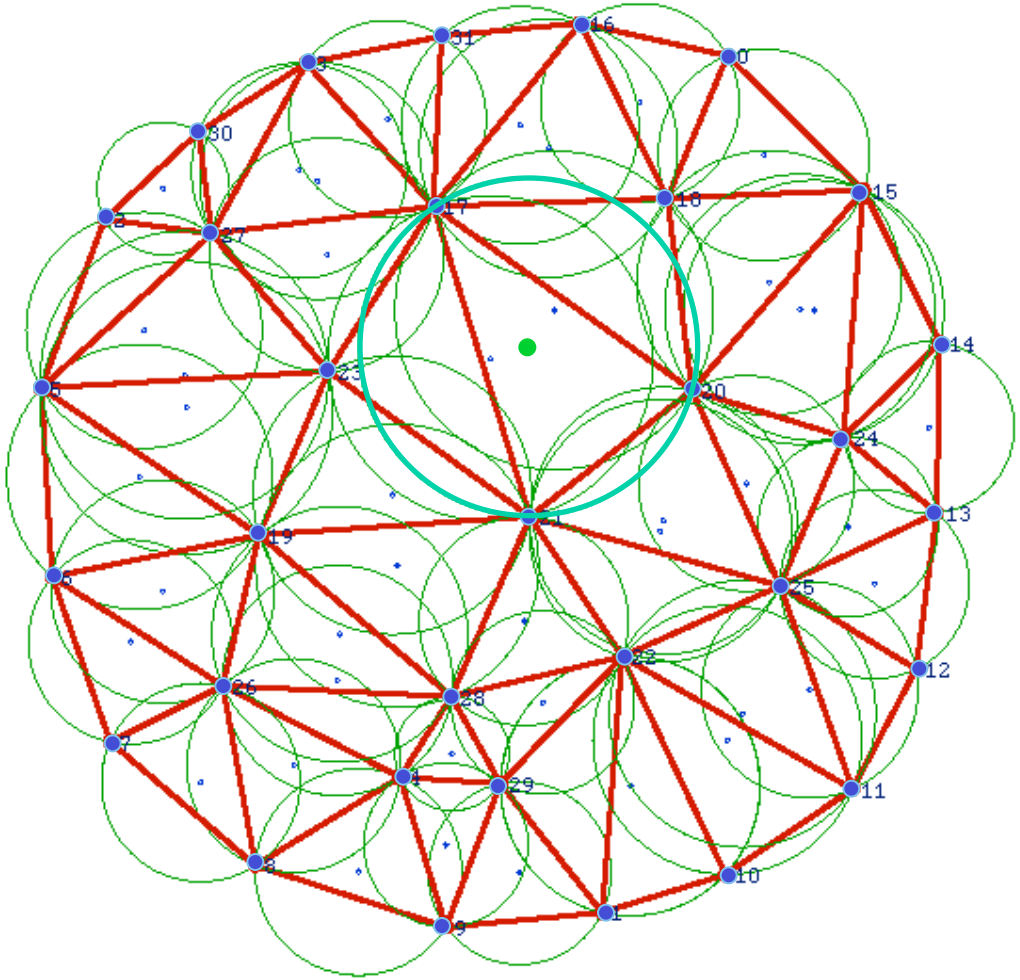


Inserting points one-by-one



The best place is a Delaunay circumcenter

Center of the largest Delaunay circle (stay in convex hull of cites)



Properties of Delaunay triangulations

- If you draw a circle through the vertices of ANY Delaunay triangle, no other sites will be inside that circle.
- It has at most 3n-6 edges and at most 2n-5 triangles.
- Its triangles are fatter than those of any other triangulation.
  - If you write down the list of all angles in the Delaunay triangulation, in increasing order, then do the same thing for any other triangulation of the same set of points, the Delaunay list is guaranteed to be lexicographically smaller.

Closest pair application

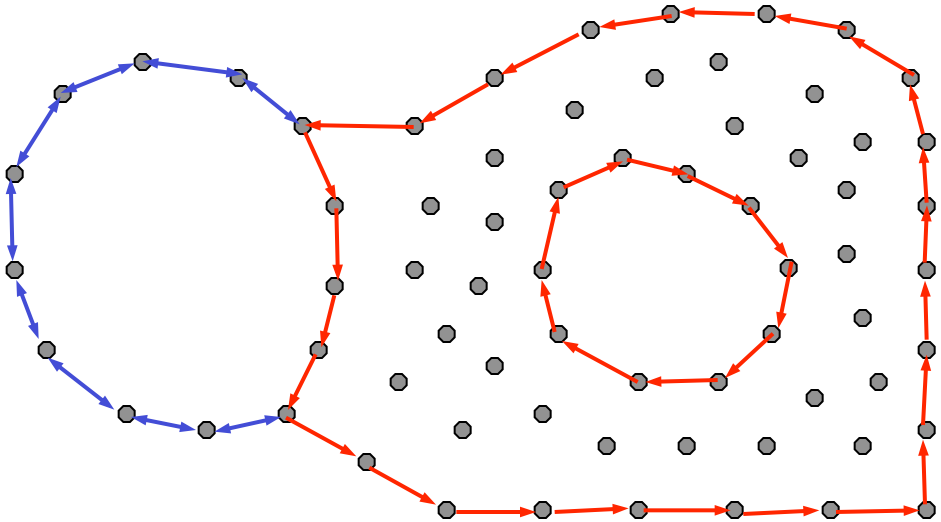
- Each point is connected to its nearest neighbor by an edge in the triangulation.
- It is a planar graph, it has at most 3n-6 edges, where n is the number of sites.
- So, once you have the Dealunay triangulation, if you want to find the closest pair of sites, you only have to look at 3n-6 pairs, instead of all n(n-1)/2 possible pairs.

Applications of Delaunay triangulations

- Find the best place to build your home
- Finite element meshing for analysis: nice meshes
- Triangulate samples for graphics

Alpha shapes

- How to obtain the point cloud interpretation (red polyloops and blue curves) from the Delaunay triangulation?

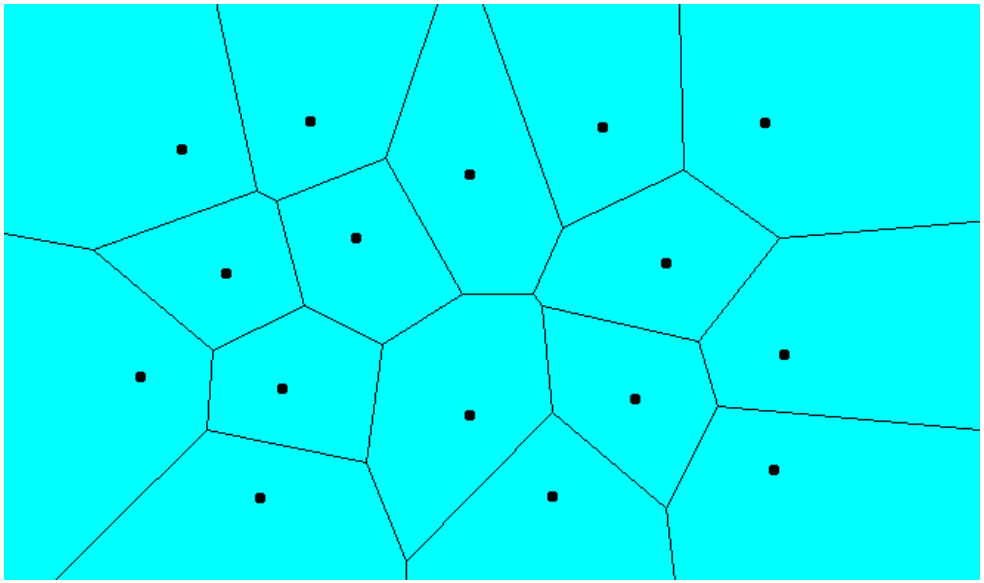


School districts and Voronoi regions

Each school should serve people for which it is the closest school.
Same for the post offices.

Given a set of schools, find the Voronoi region that it should serve.

Voronoi region of a site = points closest to it than to other sites



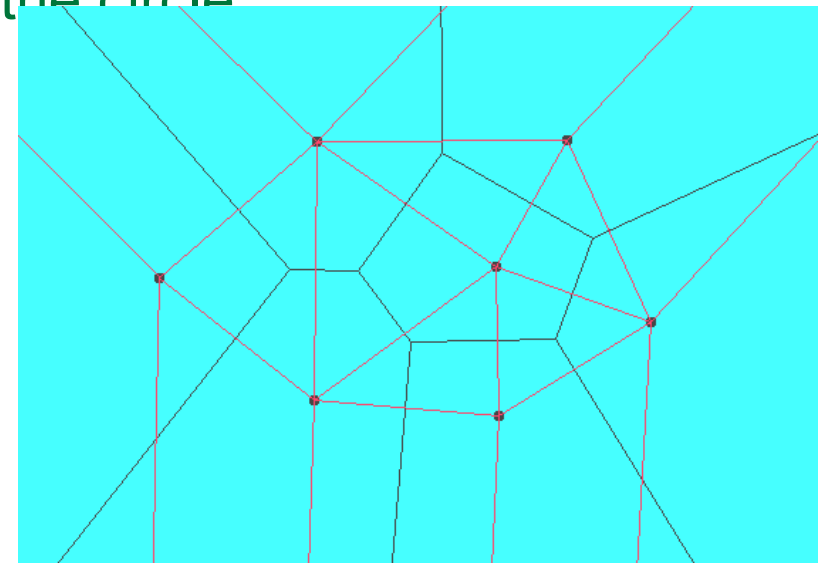
Properties of Voronoi regions

- All of the Voronoi regions are convex polygons.
  - Infinite regions correspond to sites on the convex hull.
- The boundary between adjacent regions is a line segment
  - It is on the bisector of the two sites.
- A Voronoi points is where 3 or more Voronoi regions meet.
  - It is the circumcenter of these 3 sites
  - and there are no other sites in the circle

Voronoi diagram = dual of Delaunay Triangulation

To build the Delaunay triangulation from a Voronoi diagram, draw a line segment between any two sites whose Voronoi regions share an edge.

http://www.cs.cornell.edu/Info/People/chew/Delaunay.html



Insertion algorithms for Voronoi Diagrams

Inserts the points one at a time into the diagram.

Whenever a new point comes in, we need to do three things.

First, we need to figure out which of the existing Voronoi cells contains the new site.

Second, we need to "walk around" the boundary of the new site's Voronoi region, inserting new edges into the diagram.

Finally, we delete all the old edges sticking into the new region.

Divide and conquer alg

Discovered by Shamos and Hoey.

Split the points into two halves, the leftmost n/2 points, which we'll color bLue, and the rightmost n/2 points, which we'll color Red.

Recursively compute the Voronio diagram of the two halves.

Finally, merge the two diagrams by finding the edges that separate the bLue points from the Red points The last step can be done in linear time by the "walking ant" method. An ant starts down at -infinity, walking upward along the path halfway between some blue point and some red point. The ant wants to walk all the way up to +infinity, staying as far away from the points as possible. Whenever the ant gets to a red Voronoi edge, it turns away from the new red point. Whenever it hits a blue edge, it turns away from the new blue point. There are a few surprisingly difficult details left to deal with, like how does the ant know where to start, and how do you know which edge the ant will hit next. (The interested reader is strongly encouraged to consult the standard computational geometry literature for solutions to these details.)

Assigned Reading

- http://www.voronoi.com/applications.htm
- http://www.ics.uci.edu/~eppstein/gina/scot.drysdale.html
- http://www.ics.uci.edu/~eppstein/gina/voronoi.html
- http://www.cs.berkeley.edu/~jrs/mesh/