CSCI 4560/6560 Computational Geometry

https://www.cs.rpi.edu/~cutler/classes/computationalgeometry/S22/

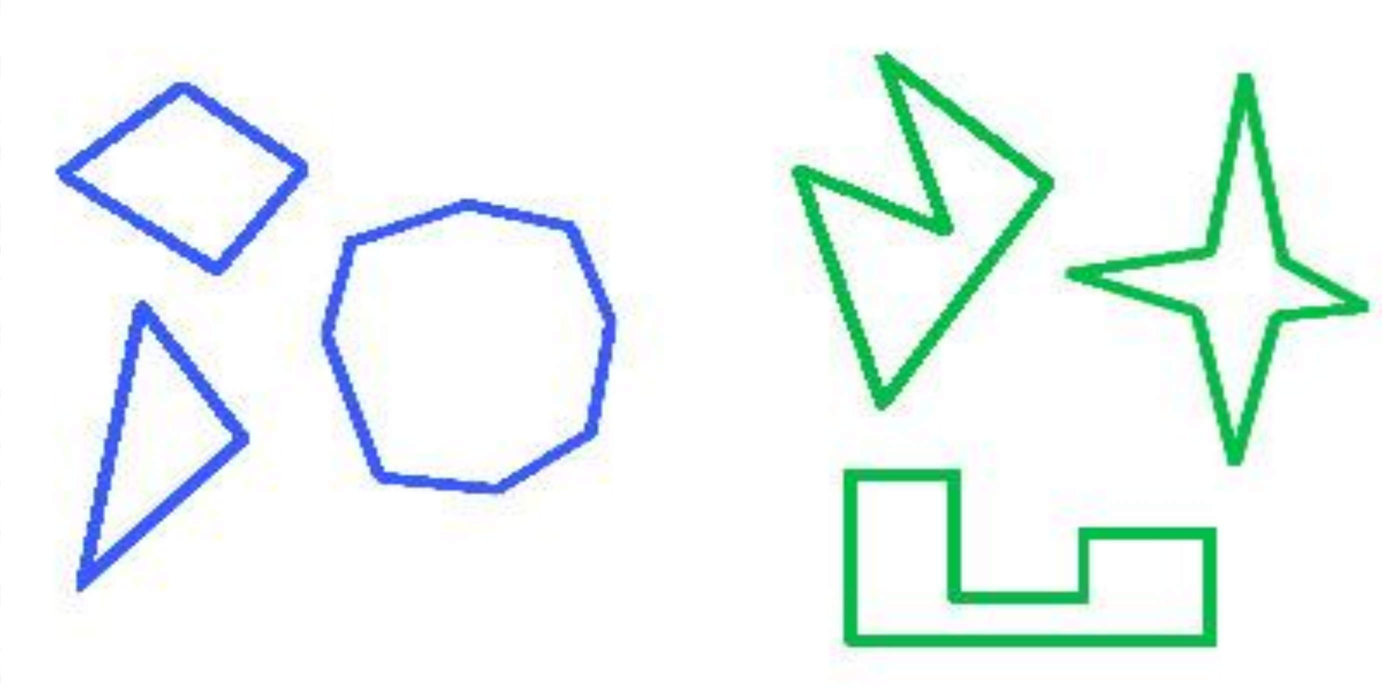
Lecture 1: Introduction & Convex Hulls

Outline for Today

- 2D Planar Convex Hulls
  - Definitions
  - A few different algorithms to construct
  - Discussion of accuracy & robustness
  - Analysis of running time
- Applications of Computational Geometry
- Introductions
- Website & Syllabus
- Homework 1: Convex Hulls

Convex: Shape has no inward corners or curving faces.

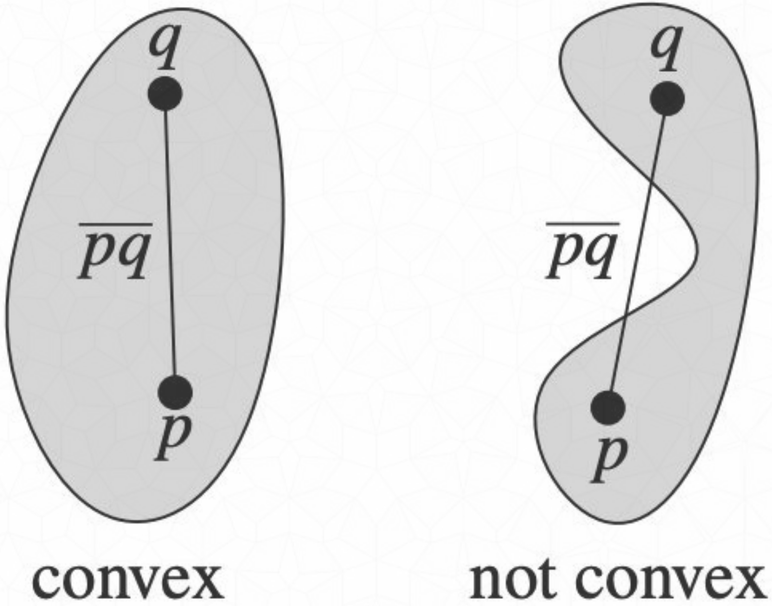
Concave: Has inward corner(s) or inward curving face(s).



http://img.sparknotes.com/figures/B/b333d91dce2882b2db48b8ad670cd15a/convexconcave.gif

Convex vs. Non-Convex

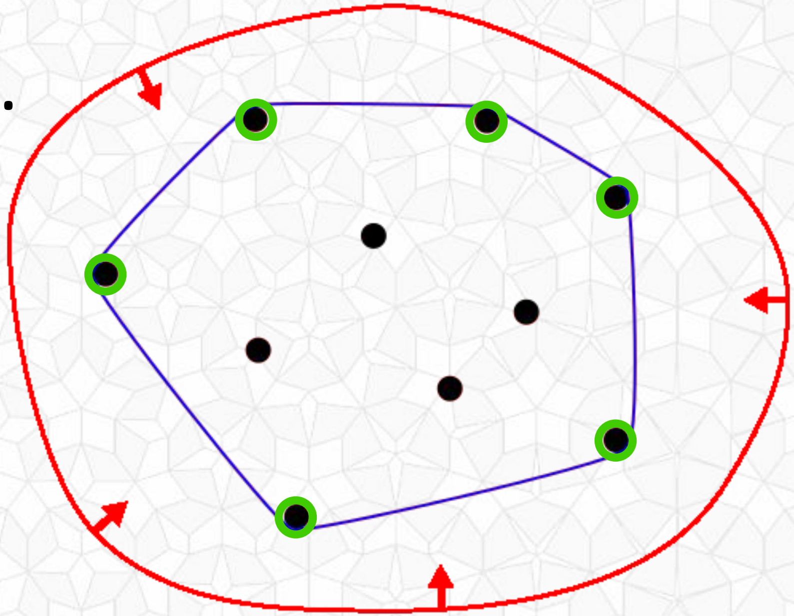
A subset S of the plane is called convex if and only if for any pair of points $p,q \in S$ the line segment $\overline{pq}$ is completely contained in S.



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 1

Convex Hull: The smallest convex shape that contains all of the input points / elements.

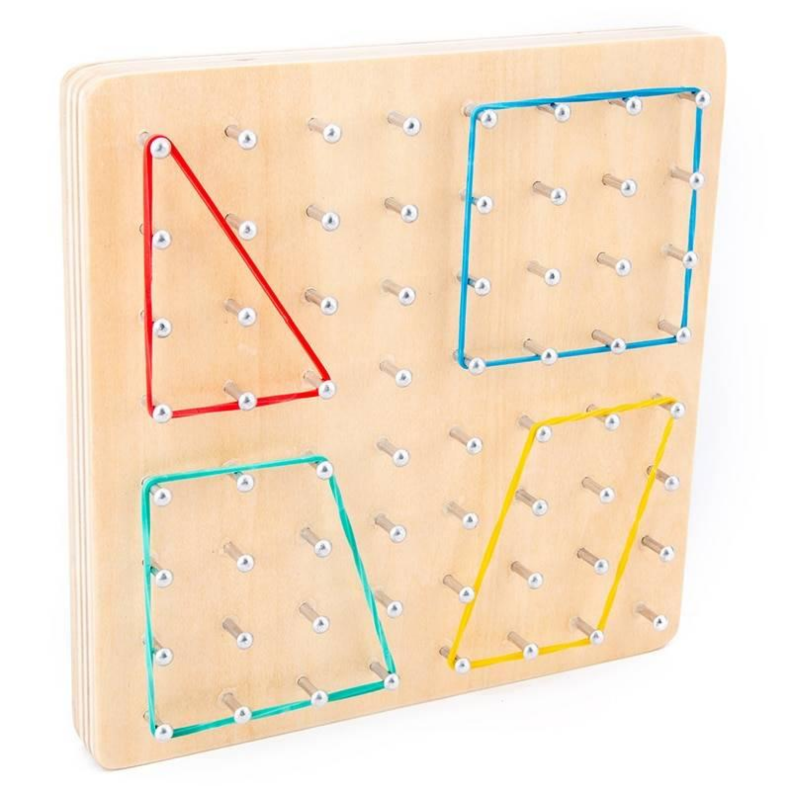
- In 2D, put a nail in the board at each point location. Stretch a rubber band over / around the outside of these nails.
- The final position of the rubber band is the convex hull.
- The nails / points touching the rubber band are the extreme points.



http://en.wikipedia.org/wiki/File:ConvexHull.svg

Convex Hull: The smallest convex shape that contains all of the input points / elements.

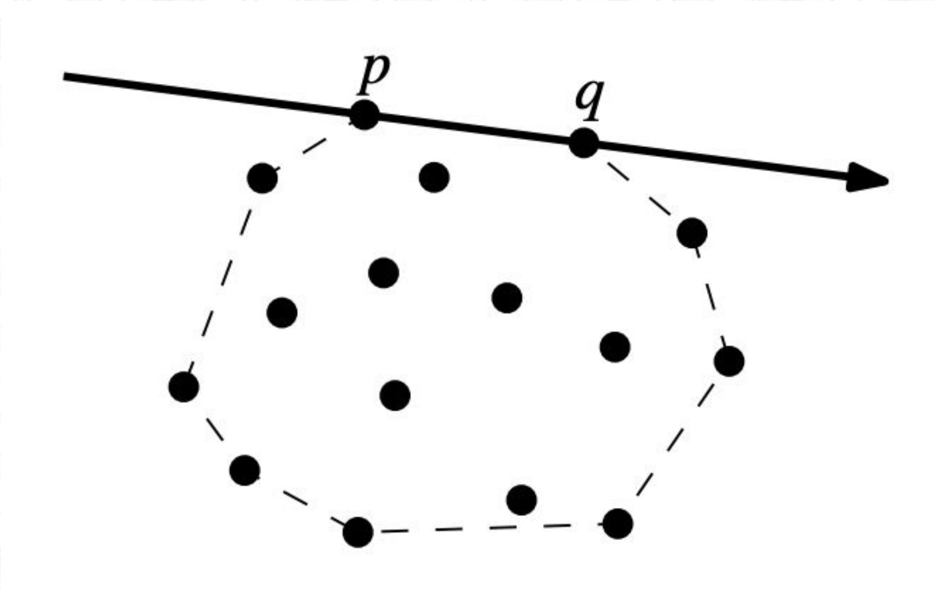
- In 2D, put a nail in the board at each point location. Stretch a rubber band over / around the outside of these nails.
- The final position of the rubber band is the convex hull.
- The nails / points touching the rubber band are the extreme points.



https://themontessoriclub.com/montessori-peg-board-the-montessori-club/

Naive Algorithm

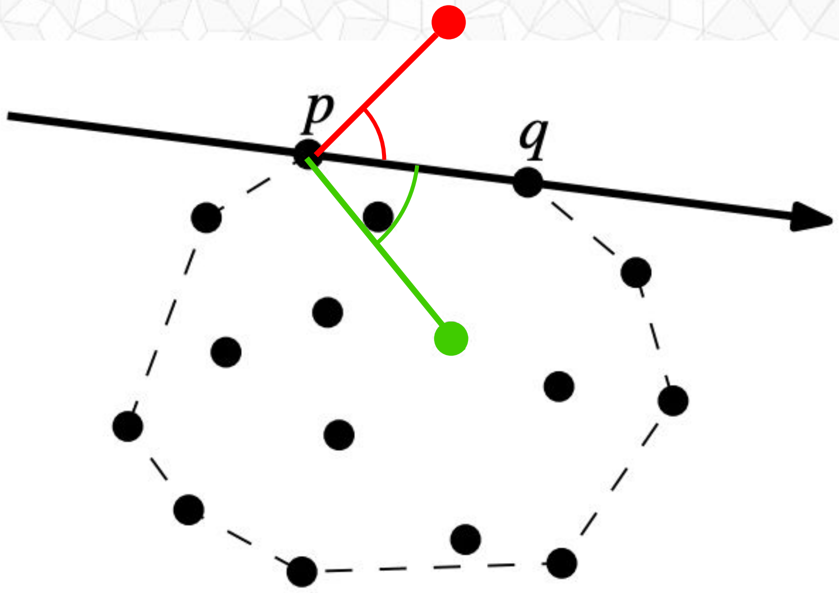
- Step 1: Find all directed line segments $\overline{pq}$ that are on the convex hull.
    - A line segment is on the convex hull if when looking down the line segment from *p* to *q*, there are no points to the left of that line.
- Step 2: Organize those line segments in clockwise order.
- Step 3: Output the starting point of each line segments
    - This will be all of the extreme points of the convex hull.



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 1

Naive Algorithm

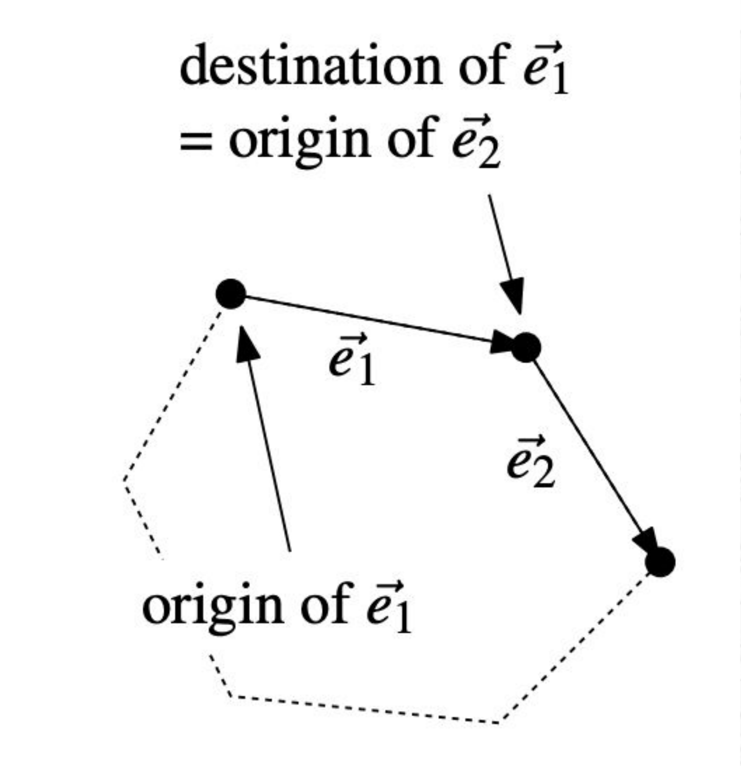
- Step 1: Find all directed line segments $\overline{pq}$ that are on the convex hull.
  - A line segment is on the convex hull if when looking down the line segment from p to q, there are no points to the left of that line.
- Step 2: Organize those line segments in clockwise order.
- Step 3: Output the starting point of each line segments
  - This will be all of the extreme points of the convex hull.



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 1

Naive Algorithm

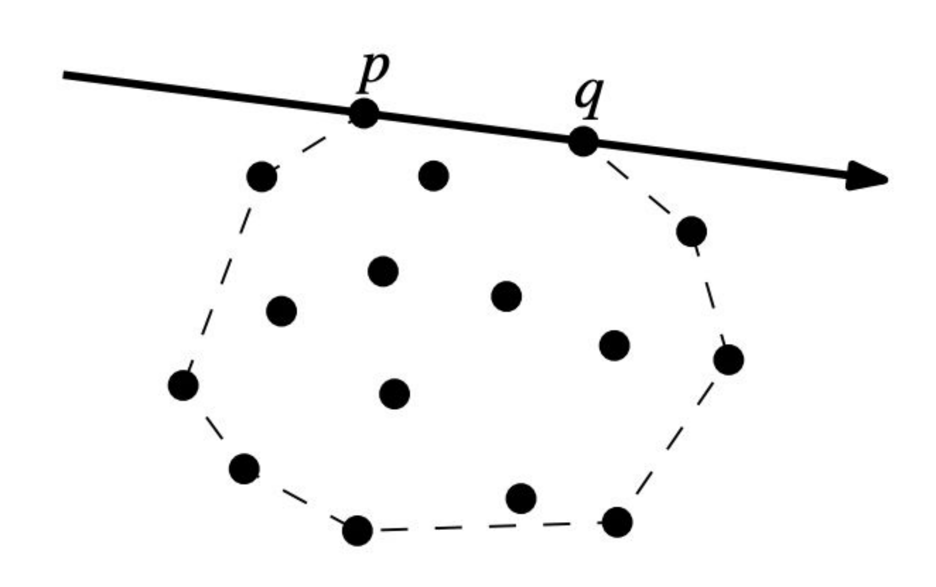
- Step 1: Find all directed line segments $\overline{pq}$ that are on the convex hull.
  - A line segment is on the convex hull if when looking down the line segment from *p* to *q*, there are no points to the left of that line.
- Step 2: Organize those line segments in clockwise order.
- Step 3: Output the starting point of each line segments
  - This will be all of the extreme points of the convex hull.



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 1

Cost of the Naive Algorithm?

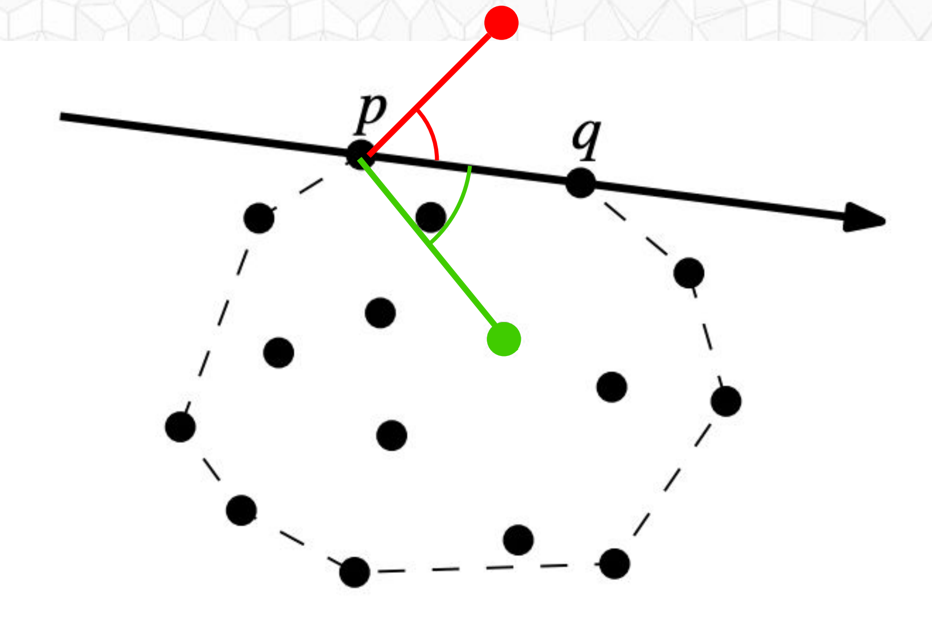
- Let *n* be # of input points, and *h* be the number of extreme points on convex hull.
- Step 1: Find edges
- Step 2: Order edges
- Step 3: Output edges



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 1

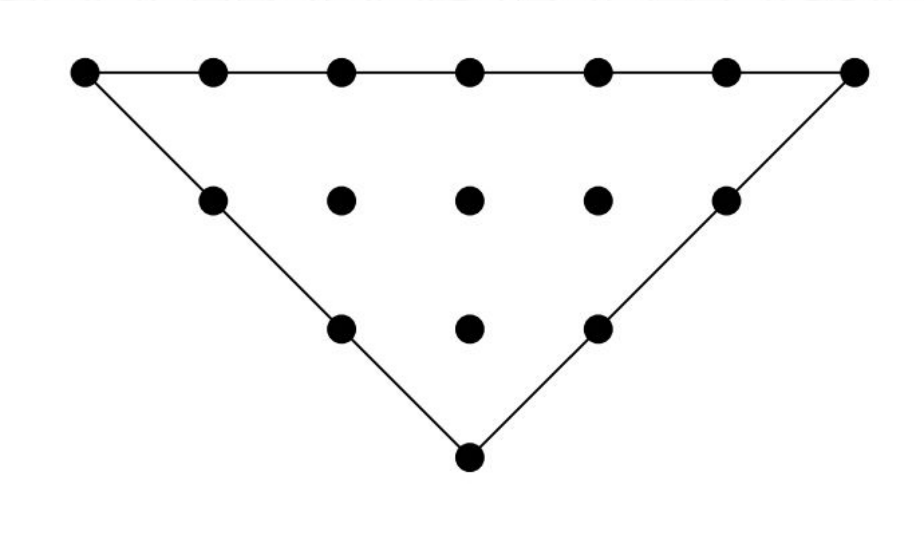
Cost of the Naive Algorithm?

- Let *n* be # of input points, and *h* be the number of extreme points on convex hull.
- Step 1: Find edges
    - For *n* points
    - *n*(n-1)* directed segments to consider
    - For each, check all other *n* points to see if any lie to the left.
    - $O(n^3)$
- Step 2: Order edges
    - For each edge $\overline{pq}$, finding the next edge (that starts with *q*) takes n time.
    - $O(h^2)$
- Step 3: Output edges
    - $O(h)$



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 1

Besides the expensive running time,
what are the problems with Naive Algorithm?



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 1

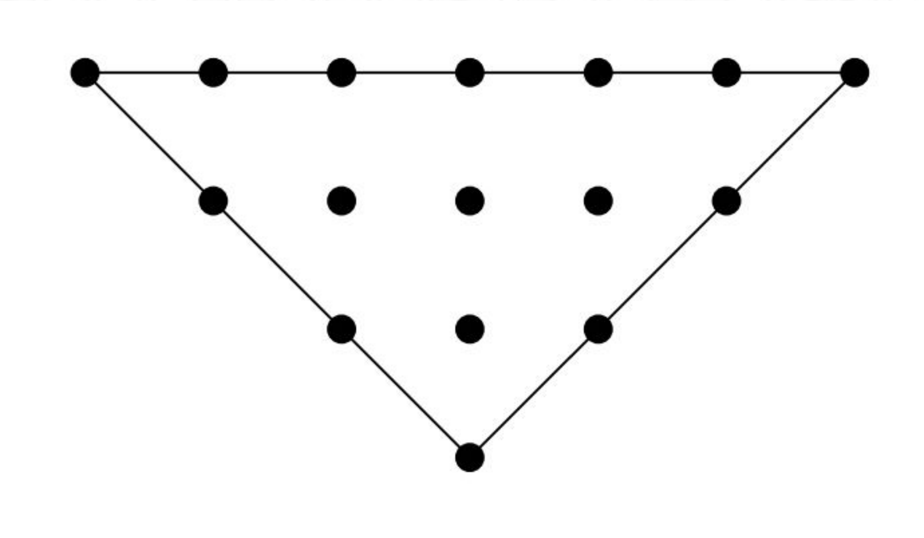
Besides the expensive running time, what are the problems with Naive Algorithm?

- Is it well defined?

  Do we agree on what is the right answer in all cases?

- Might we have problems with numerical precision?

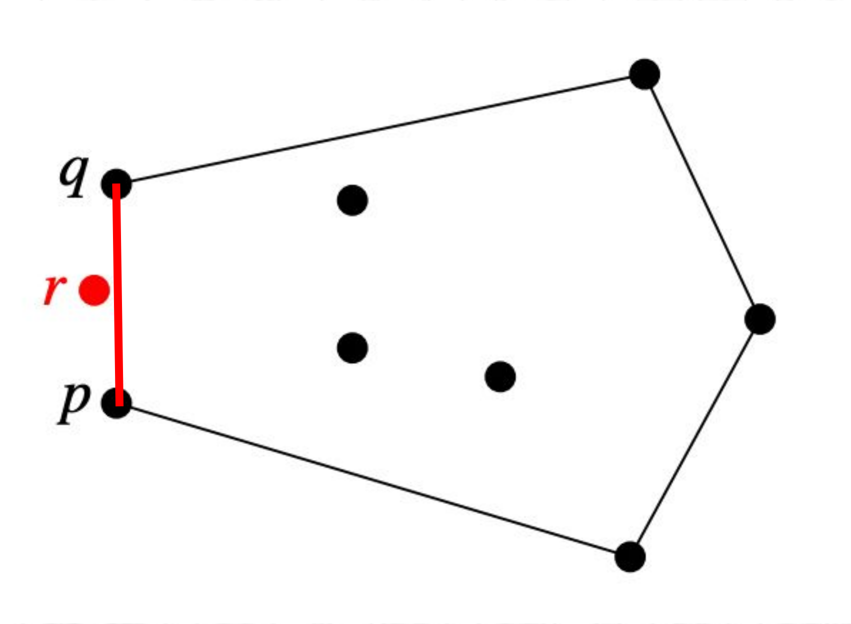
  Floating point rounding errors?

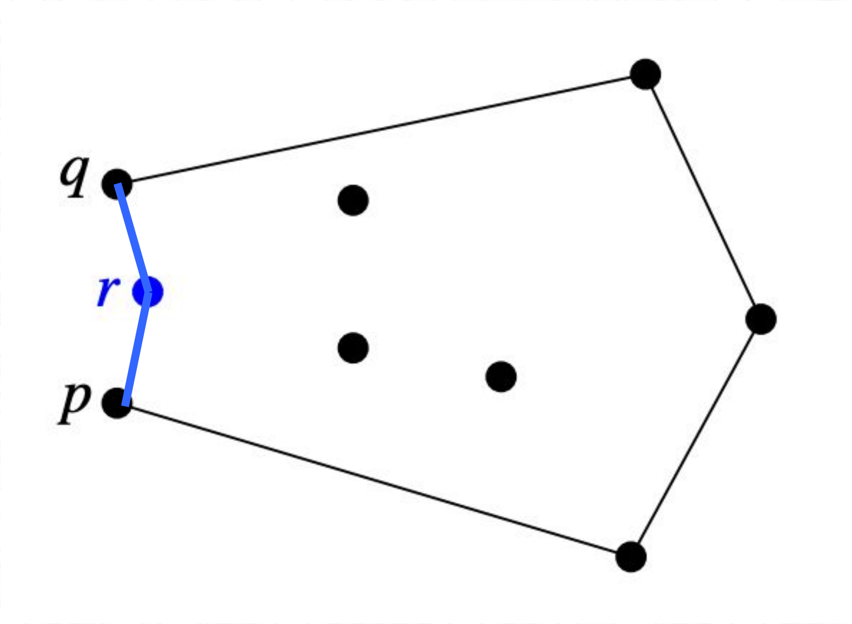


Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 1

Floating point rounding errors

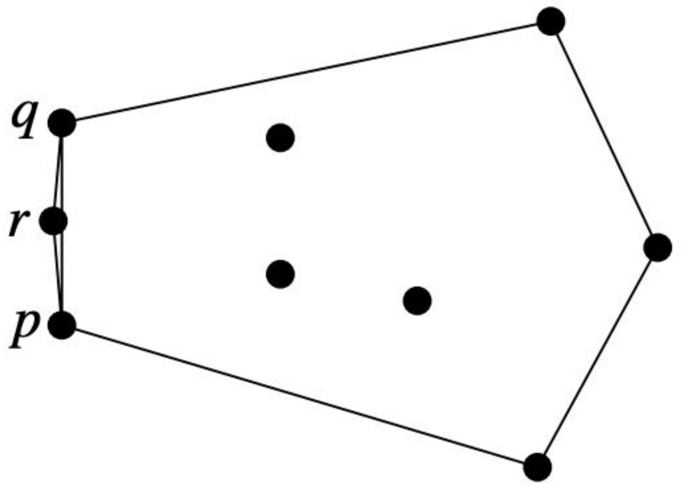
- May cause a point to be missed that *should be* on the boundary
- May cause a point to be included that *should not be* on the boundary

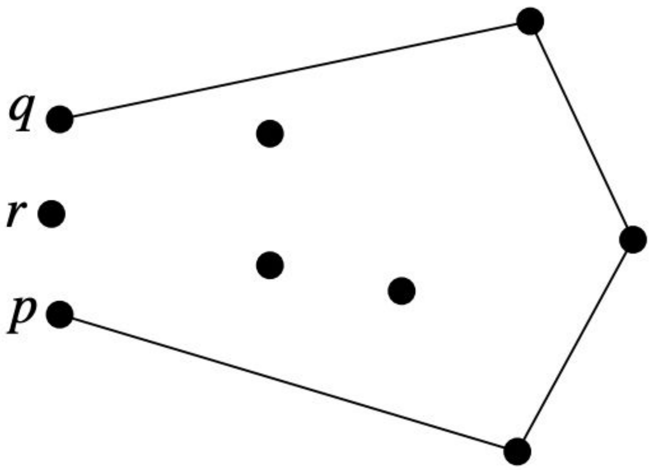




Or worse…

- Judgements about being left vs. right side may be inconsistent
- This can cause duplicates or gaps in the boundary

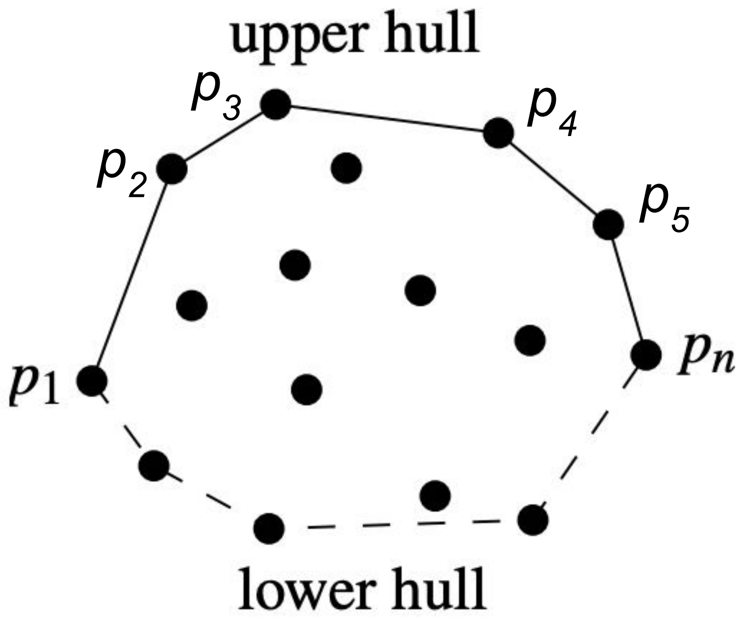




Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 1

Let's try again…

- We will construct the upper hull (and then similarly, the lower hull)
- Maintain a list of the points $p_1, p_2, .. p_i$ that form the current upper hull

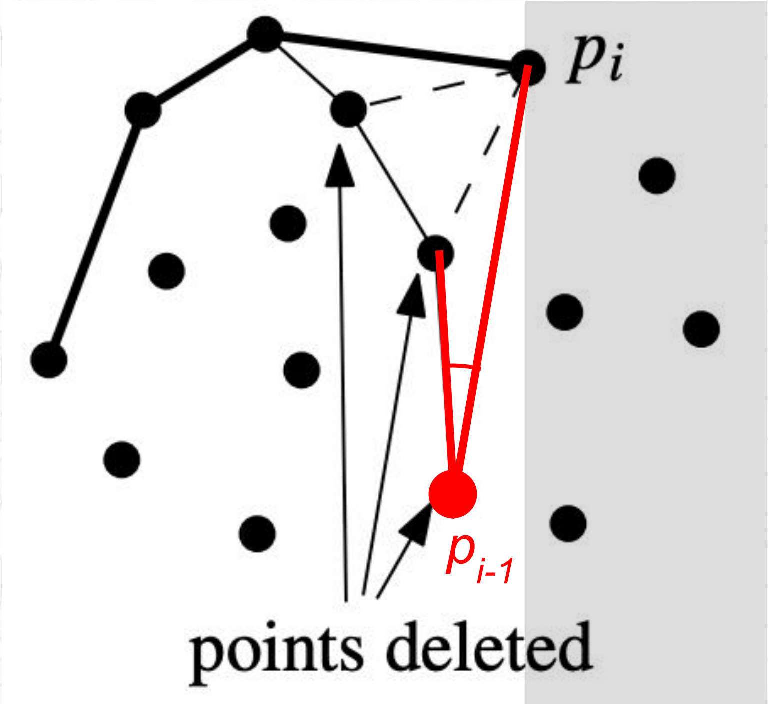


Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 1

Let’s try again… Construct the Upper Hull

- Step 1: Sort the input points by x coordinate. The leftmost point must be on the upper hull.

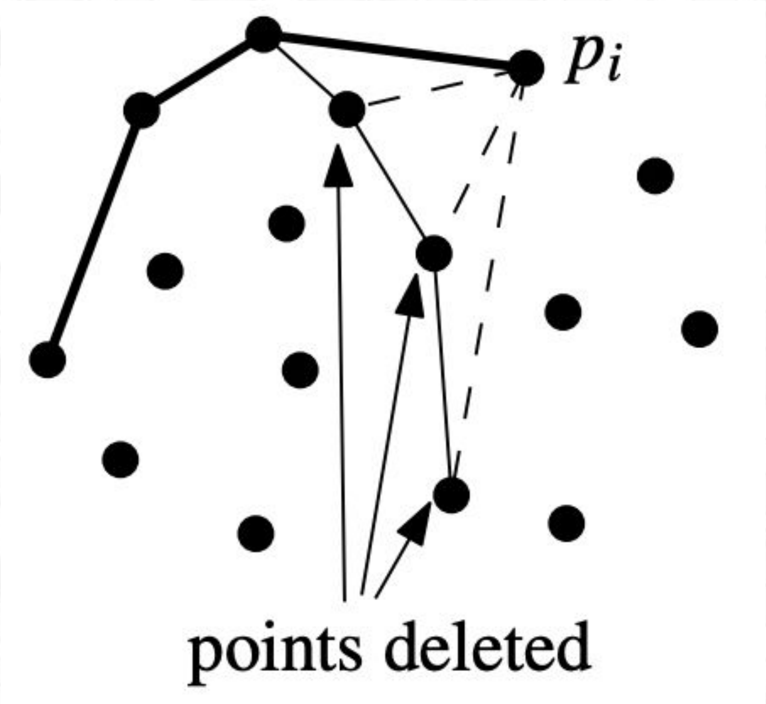
- Step 2: Walk through the points from left to right. Add p_i to the upper hull.
- Step 3: For each added point… if the angle $p_{i-2}\, p_{i-1}\, p_i$ is a left bend, remove p_{i-1} (& check previous point too)



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 1

Analysis of Constructing the Upper Hull?

- Let *n* be # of input points
- Step 1: Sort
- Step 2: Add each point
- Step 3: Remove points
- Overall:

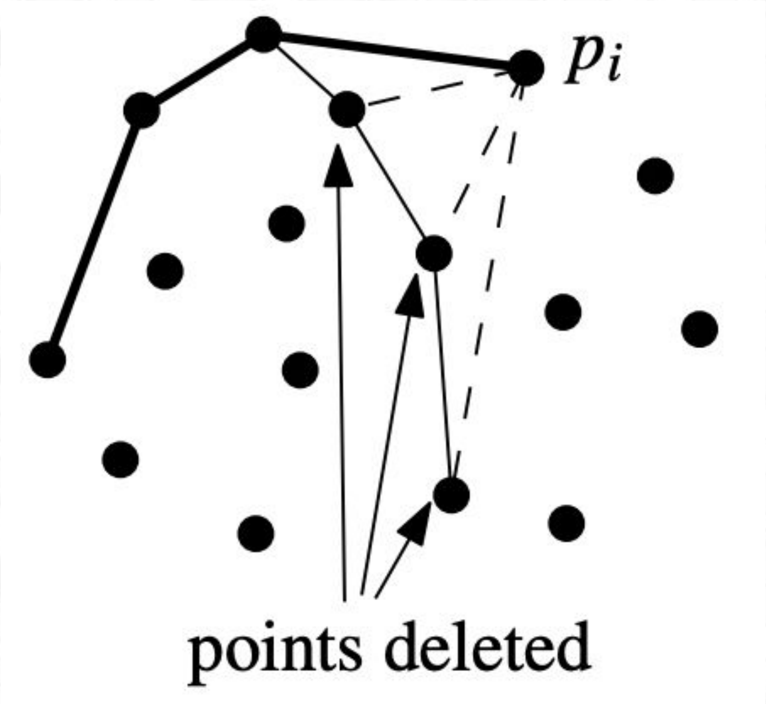


Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 1

Analysis of Constructing the Upper Hull?

- Let *n* be # of input points
- Step 1: Sort
  - *O(n log n)*
- Step 2: Add each point
  - *O(n)* total
- Step 3: Remove points
  - *O(n)* max total cost

- Overall:
  - *O(n log n)*

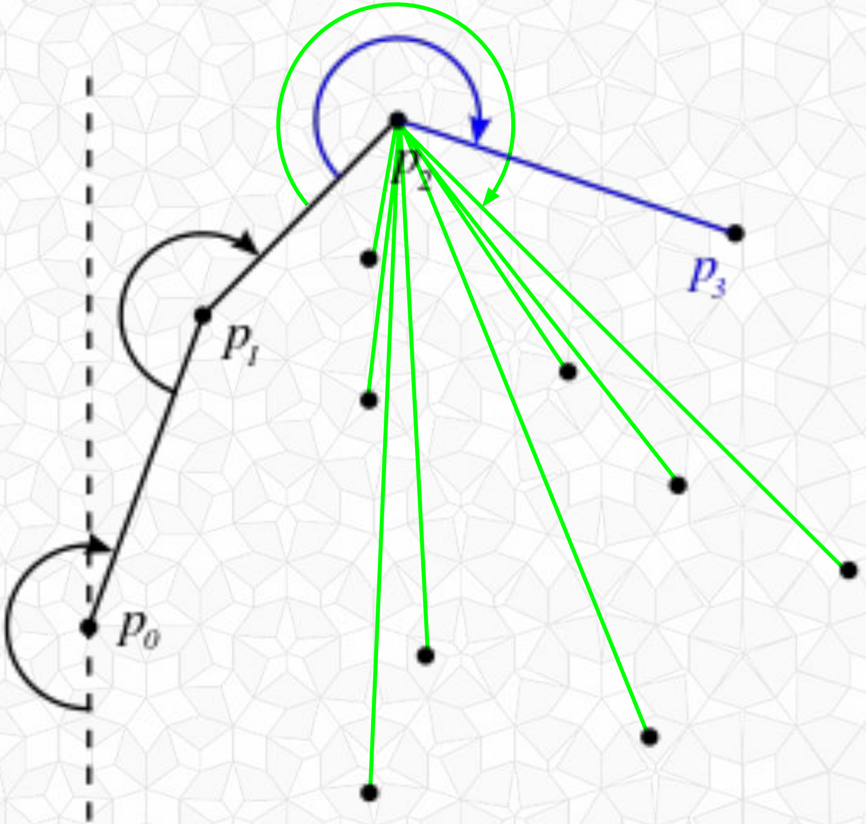


Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 1

Can we do better? “Gift Wrapping” Algorithm

- Step 1: Find p_0
  The point with the smallest x coordinate.

- Step 2: “Walk around” the point set in the clockwise direction.
  - At each point e.g., p_2, find the next point, p_3 on the hull.
  - Check all other points…
  - Find the smallest outer angle between lines $\overline{p_1 p_2}$ & $\overline{p_2 p_3}$

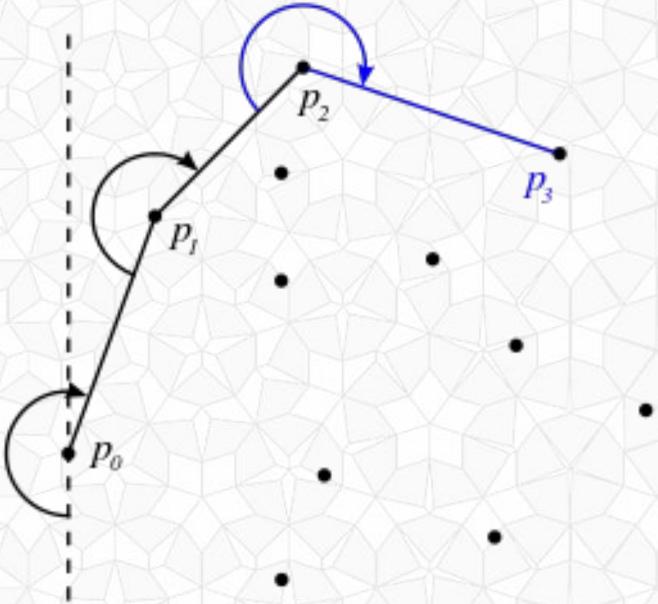


Gift Wrapping Algorithm Analysis

- Let *n* be # of input points, and
  h be the number of extreme points on convex hull.

- Step 1: Find p_0

- Step 2: Find each next point on the hull



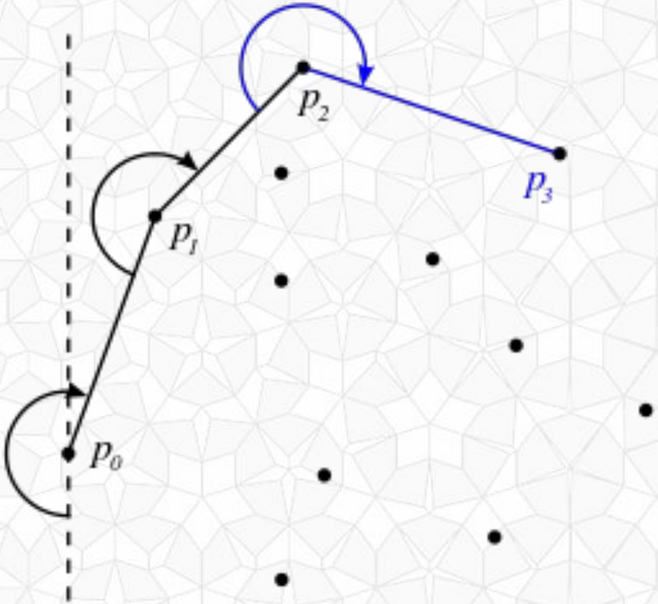
Gift Wrapping Algorithm Analysis

- Let n be # of input points, and
  h be the number of extreme points on convex hull.

- Step 1: Find p_0
  - $O(n)$

- Step 2: Find each next point on the hull
  - h times
  - find the next point = $O(n)$
  - Overall $O(n*h)$

- Is this better?



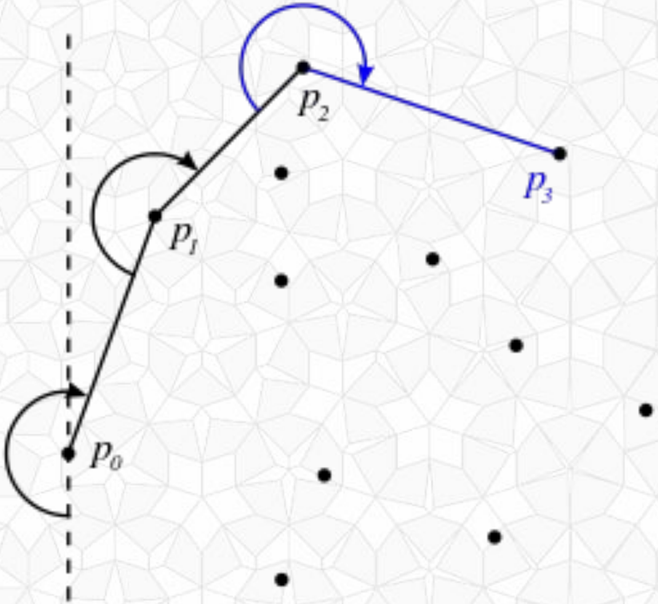
Gift Wrapping Algorithm Analysis

- Let *n* be # of input points, and
  h be the number of extreme points on convex hull.

- Step 1: Find p_0
  - *O(n)*

- Step 2: Find each next point on the hull
  - *h* times
  - find the next point = *O(n)*
  - Overall *O(n*h)*

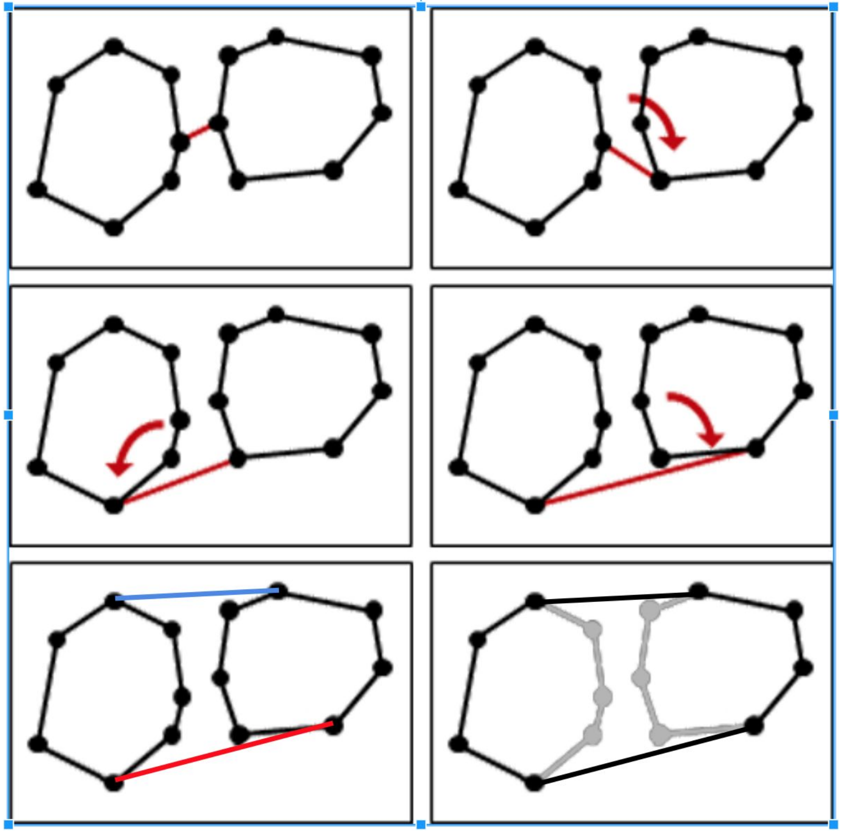
- Is this better?
  - Worst case? *h = n*
    most/all input points are on the convex hull
    $O(n^2)$
  - Best case? *h < log n*
    and then it is better than previous algorithm



Recursive Divide & Conquer Algorithm *(like Merge Sort)*

- Split Step:
  - Sort points by the *x* coordinate
  - Split into 2 equal-sized groups
  - Then recurse…

- Merge Step:
  - Find rightmost point in left hull, and leftmost point in right hull.
  - Walk down to find lower tangent
  - & walk up for upper tangent
  - Discard points in between upper & lower tangents

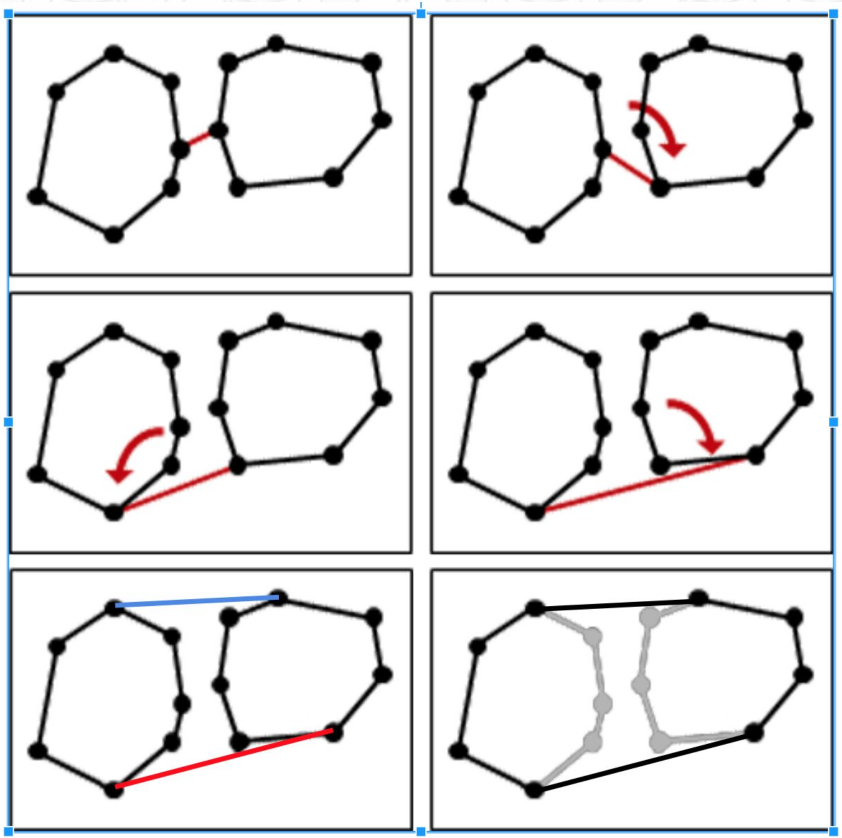


Analysis of Recursive Divide & Conquer Algorithm

- Sort points:

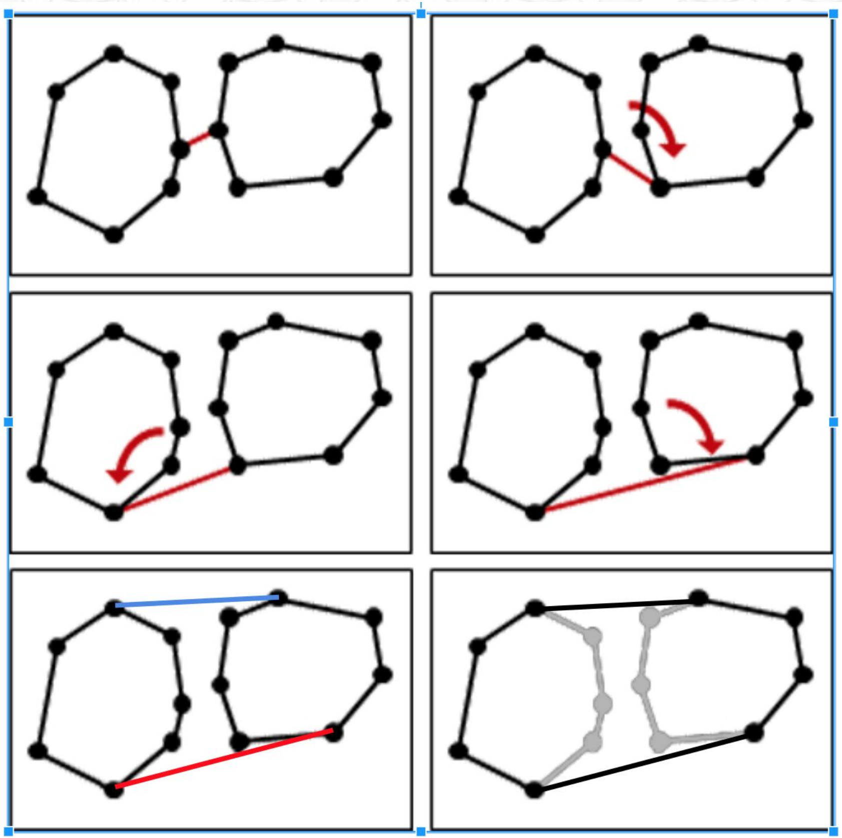
- Split Step:

- Merge Step:



Analysis of Recursive Divide & Conquer Algorithm

- Sort points: *only once*
  - *O(n log n)*
- Split Step:
  - *n* splits
- Merge Step:
  - *n* merges
  - each of the *n* points will be removed at most once
- Overall:
  - *O(n log n)*



Beyond 2D Planar Convex Hulls

- 3D Convex Hulls… & higher dimensions!
- Image Based Visual Hulls (not the same!)

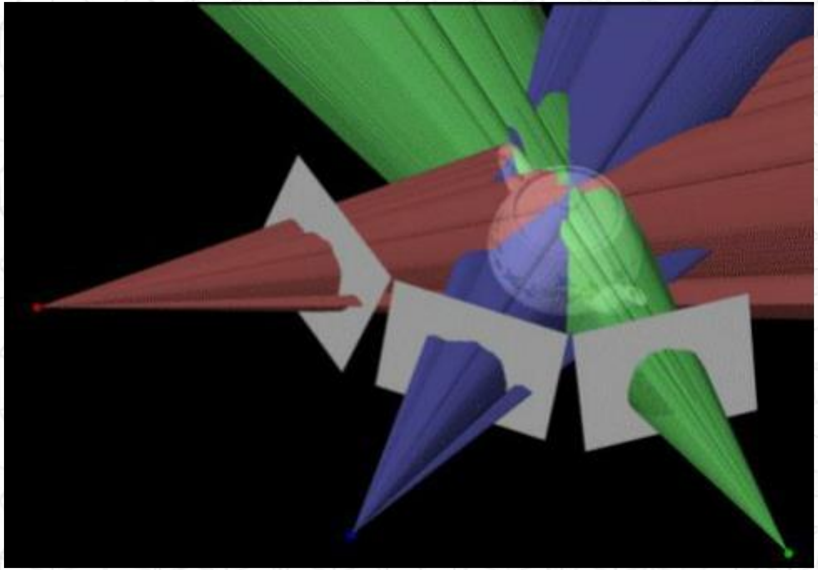
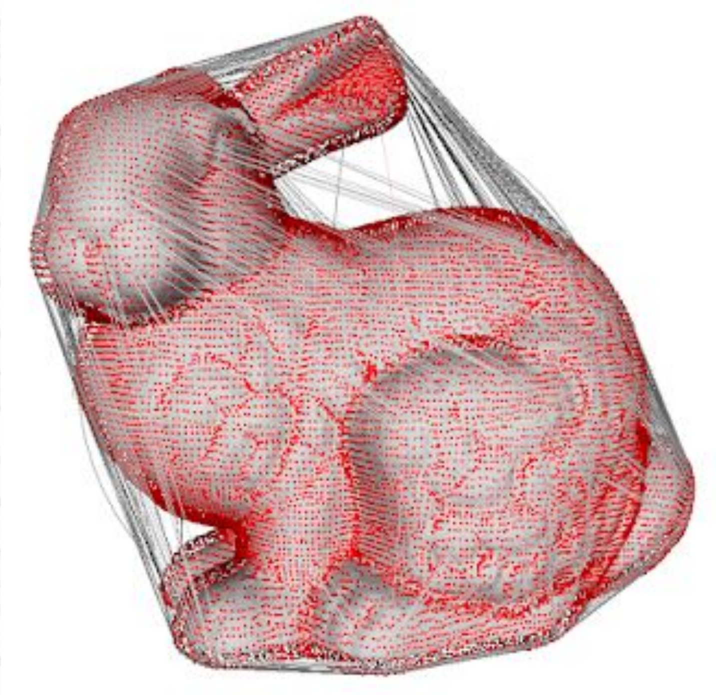


Image-Based Visual Hulls,
Matusik et al, SIGGRAPH 2000



http://diskhkme.blogspot.com/2015/10/convex-hull-algorithm-in-unity-2-3d.html

Outline for Today

- 2D Planar Convex Hulls
  - Definitions
  - A few different algorithms to construct
  - Discussion of accuracy & robustness
  - Analysis of running time
- Applications of Computational Geometry
- Introductions
- Website & Syllabus
- Homework 1: Convex Hulls

Applications for Computational Geometry

- Computer Graphics / Games / Virtual Reality / Computer Vision
  primitive intersections, hidden surface removal, ray tracing, collision detection
- Robotics
  motion planning, kinematics, robot arm placement
- Geographics Information Systems (GIS)
  modeling terrain, river networks, average rainfall, population, map overlays
- CAD/CAM (manufacturing)
  intersection & union of objects, physical simulations, feasibility of assembly
- Other: Molecular Modeling, Optical Character Recognition (OCR), etc.
- General purpose database / data record comparisons
  can be very high dimension! (more than 3D!)

Introductions

- Let's go around the "room" and introduce ourselves
  Share anything you are comfortable sharing

- Name
- Current degree program (department, major, dual major)
- Number of terms you've been at RPI
- *Possible connections to Computational Geometry…*
  - Prior course work
  - Current research
  - Extra-curricular interests
- What you hope to learn this semester

Outline for Today

- 2D Planar Convex Hulls
  - Definitions
  - A few different algorithms to construct
  - Discussion of accuracy & robustness
  - Analysis of running time
- Applications of Computational Geometry
- Introductions
- Website & Syllabus
- Homework 1: Convex Hulls