Advanced Computer Contest Preparation

Lecture 20

POLYGONS

Terms

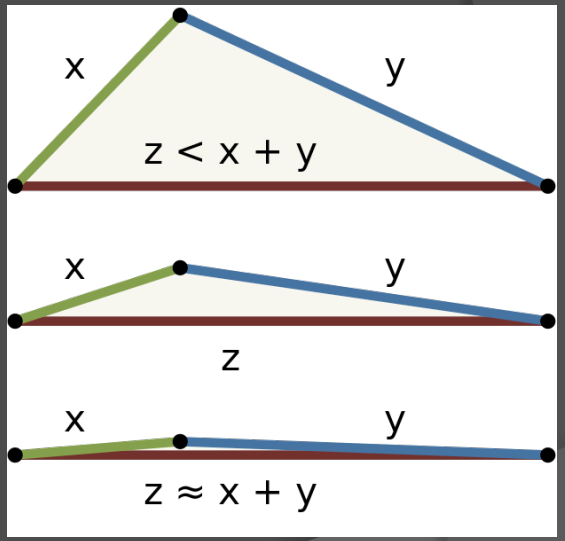
- Polygon
  - A figure on a plane bounded by a finite sequence of line segments, forming a loop
- n-gon
  - A polygon with n sides

Terms

- Simple
  - Line segments do not intersect with each other (except adjacent segments)
- Convex
  - No interior angle exceeds $180°$
  - If two points inside of a polygon are chosen, the line segment between them lies entirely within the polygon
  - All convex polygons are simple polygons

Properties

- Triangle inequality
  - True for any triangle
  - The length of one side does not exceed the sum of the other two
    - For a triangle with lengths x, y, and z, where z is the longest side
    - $z \leq x+y$
  - Equality holds for longest side when triangle is a straight line

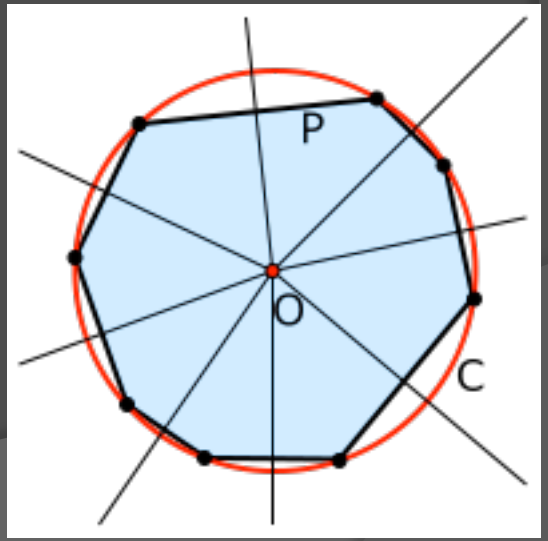


Properties

- The largest side of any polygon is less than or equal to the sum of the other sides
  - Extension of triangle inequality
  - Equality holds when polygon is a straight line

Properties

- If we are given N line segments and we want to form an N-gon with maximal area, the vertices of the polygon should be concyclic
  - All vertices lie on a common circle



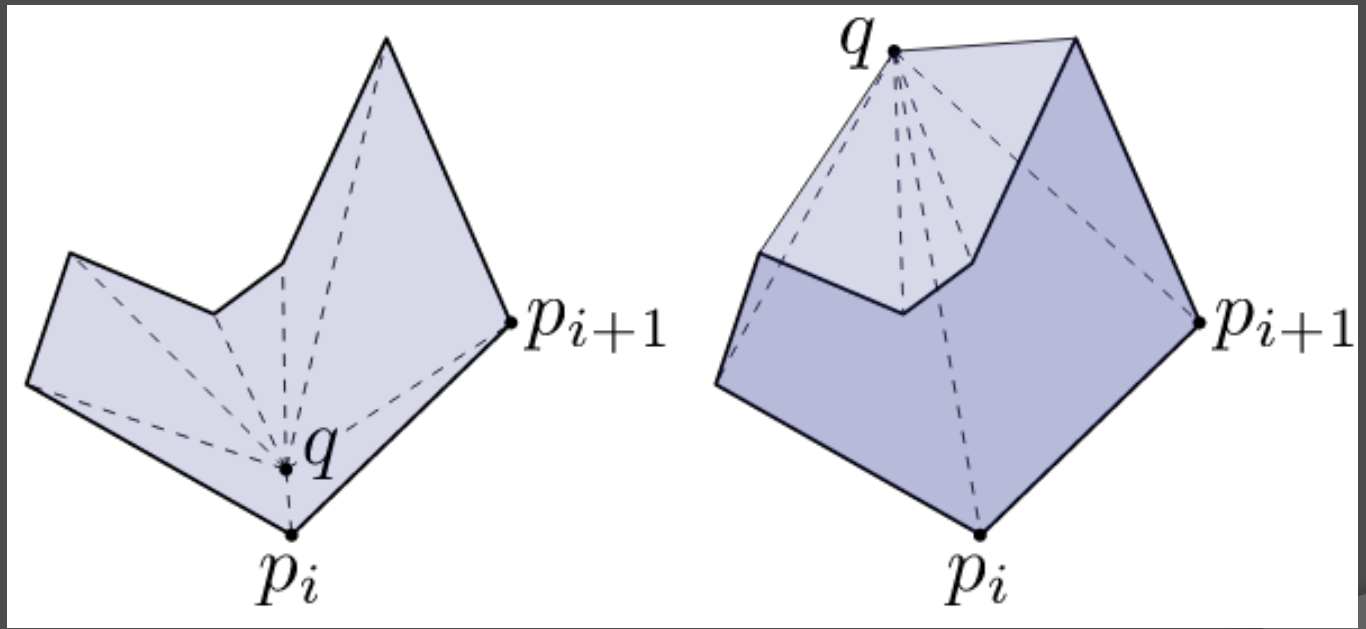
Area of Polygon

- Given an n-gon, what is its area?

Area – Cross Product Method

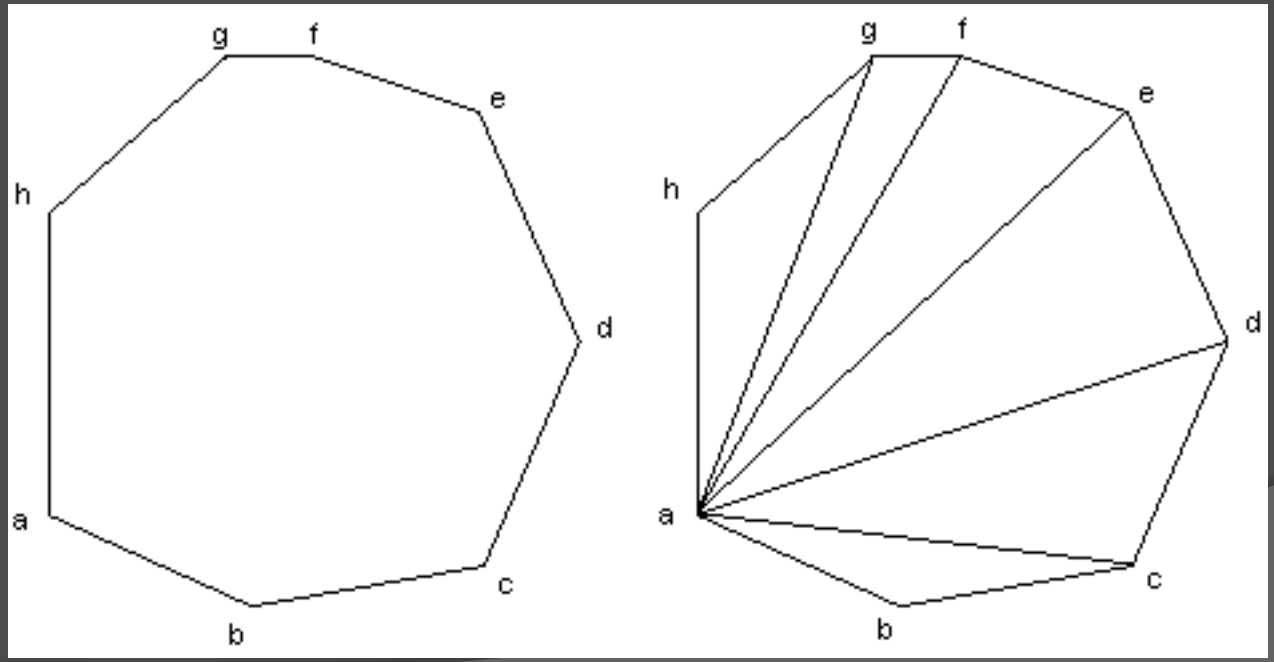
- Pick an arbitrary point
- Get the sum of areas of the triangle between the chosen point and all pairs of two adjacent vertices
  - Use cross product!
- Positive and negative areas of cross product will cancel out unnecessary area
  - e.g. area outside of the polygon
- Area is supposed to always be positive, so take absolute value as the last step

Area – Cross Product Method



Area – Cross Product Method

- Example: choose a single vertex of a convex polygon to be the arbitrarily chosen point



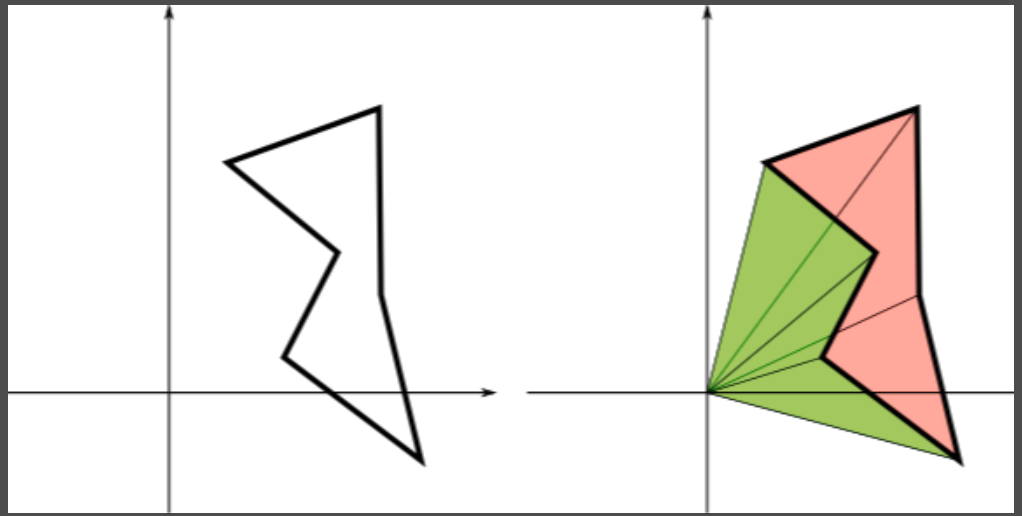
Area – Shoelace Formula

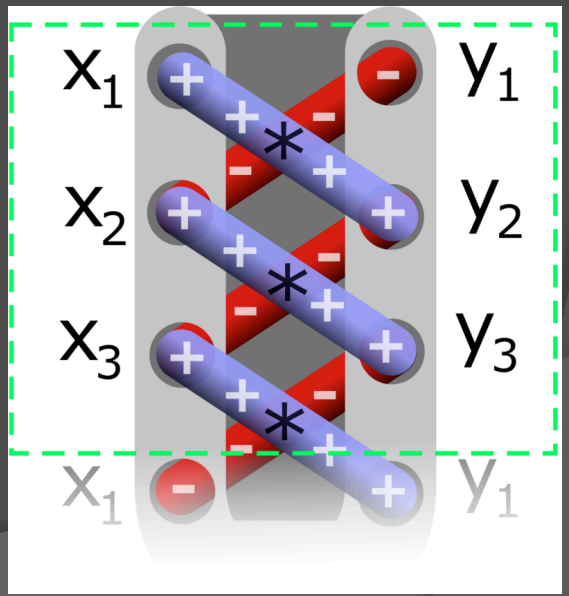
- Use cross product method, but the chosen point is the origin
- Vector of each vertex is the vertex's coordinates
- Cross product of 2 2-D vectors is $x_1y_2-x_2y_1$
- When applying for every pair of adjacent vertices, we get the shoelace formula

Area – Shoelace Formula

$$\mathbf{A} = \frac{1}{2}\left|\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right|$$

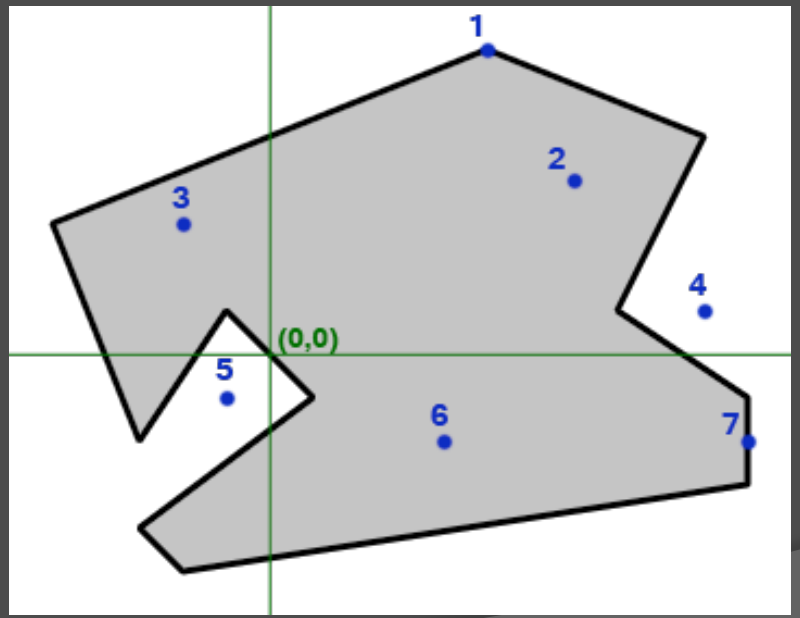
$$= \frac{1}{2}\left|x_1 y_2 + x_2 y_3 + \cdots + x_{n-1} y_n + x_n y_1 - x_2 y_1 - x_3 y_2 - \cdots - x_n y_{n-1} - x_1 y_n\right|$$





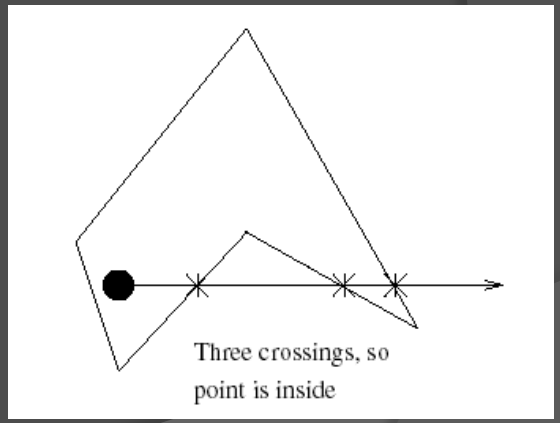
Point in Polygon

- Given a simple polygon and a point, is the point inside of the polygon?



Point in Polygon

- Cast a ray from the given point in any direction
- Ray from point inside of simple polygon should intersect with the polygon's sides an odd number of times
- Implementation:
  - Form vector between point and another point very far away (e.g. $(10^9, 10^9)$)
  - Perform line intersection algorithm with each side of polygon and the vector

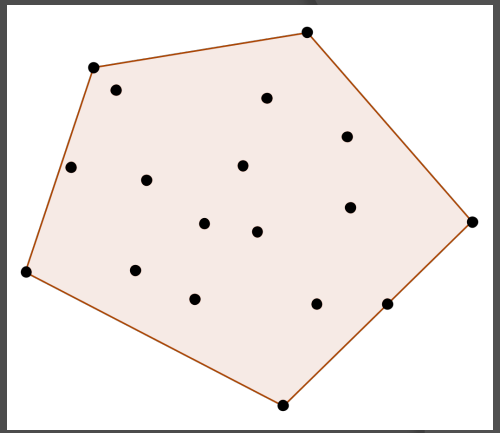


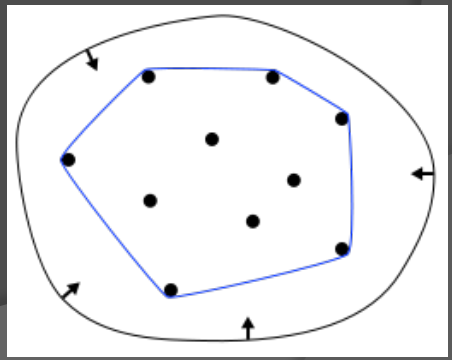
Point in Polygon

- Be wary if:
  - Point is on the edge of the polygon
    - Manually check if point is on an edge
  - Ray/vector intersects polygon at a vertex
    - Rare situation: try another vector if this occurs

Convex Hull

- Given N points in 2-D space, what is the smallest convex polygon that contains all points?
- Convex hull will always be formed using some of the given points
  - Think of a rubber band stretched around some pegs





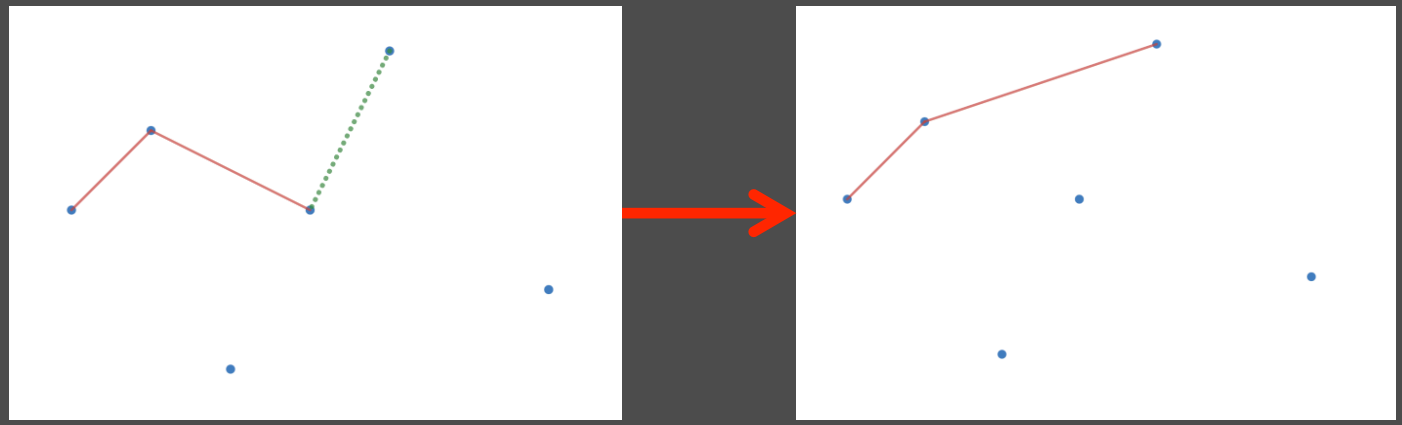
Andrew's Monotone Chain Convex Hull Algorithm

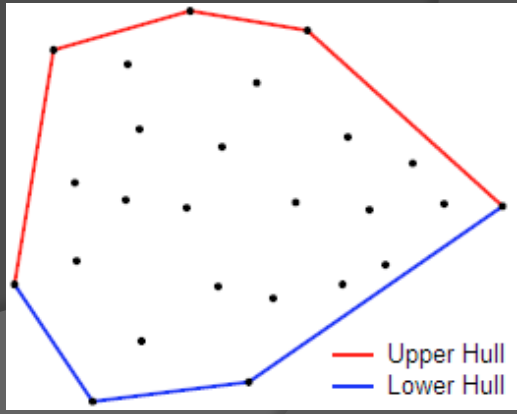
- $O(N \log N)$ algorithm that computes the convex hull
- First, sort points lexicographically
  - Sort by increasing x, then by y

Monotone Chain Convex Hull

- Construct upper hull first
  - Maintain a stack of points, specifying those that will be a part of the convex hull
  - Iterate through points in order
  - Let $\mathbf{a}$ be the vector from the 2nd topmost point of the stack, pointing toward the top point of the stack
  - Let $\mathbf{b}$ be the vector from the topmost point of the stack pointing toward the current point
  - If $\mathbf{b}$ is a counter clockwise turn from $\mathbf{a}$, pop the top point from the stack
  - Repeat until turn is clockwise
- Repeat steps, iterating through points in reverse order, to construct lower hull

Monotone Chain Convex Hull





Pseudocode

```
point pts[];
sort(pts);
stack<point> h1,h2;
for (point p : pts){
       while(h1.size() > 1)
              if (!clockwise(
                     vector(h1.second(),h1.top()),
                     vector(h1.top(),p))
                     h1.pop();
       h1.push(p);
}
reverse(pts);
//repeat for loop with h2 instead
//hull will be in both h1 and h2
```

THANK YOU!