Robert Sedgewick | Kevin Wayne

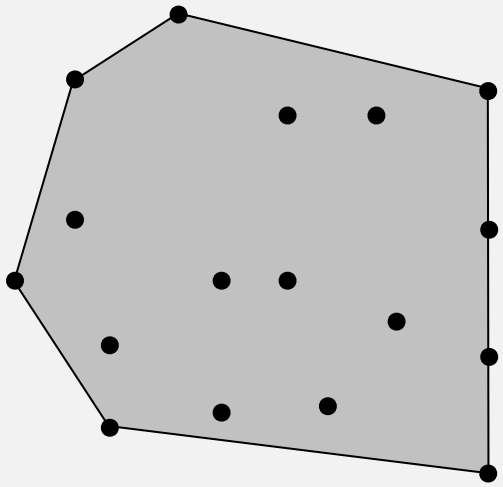
http://algs4.cs.princeton.edu

2.1 Elementary Sorts

- *rules of the game*
- *selection sort*
- *insertion sort*
- *shellsort*
- *shuffling*
- *convex hull*

Convex hull

The convex hull of a set of N points is the smallest perimeter fence enclosing the points.

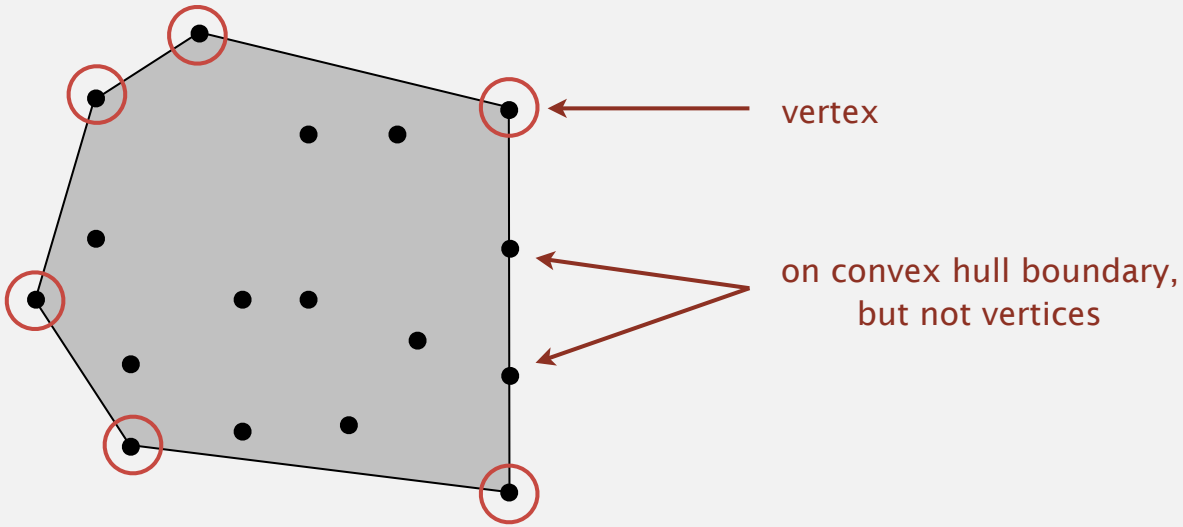


Equivalent definitions.

- Smallest convex set containing all the points.
- Smallest area convex polygon enclosing the points.
- Convex polygon enclosing the points, whose vertices are points in set.

Convex hull

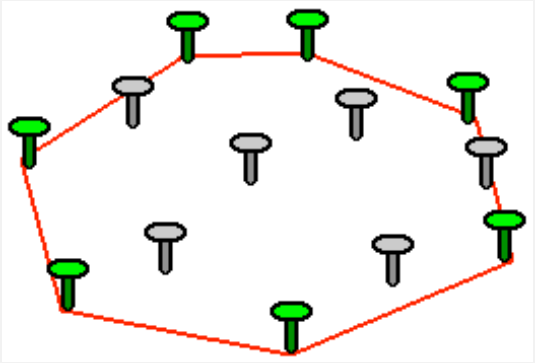
The convex hull of a set of N points is the smallest perimeter fence enclosing the points.



Convex hull output. Sequence of vertices in counterclockwise order.

Convex hull: mechanical algorithm

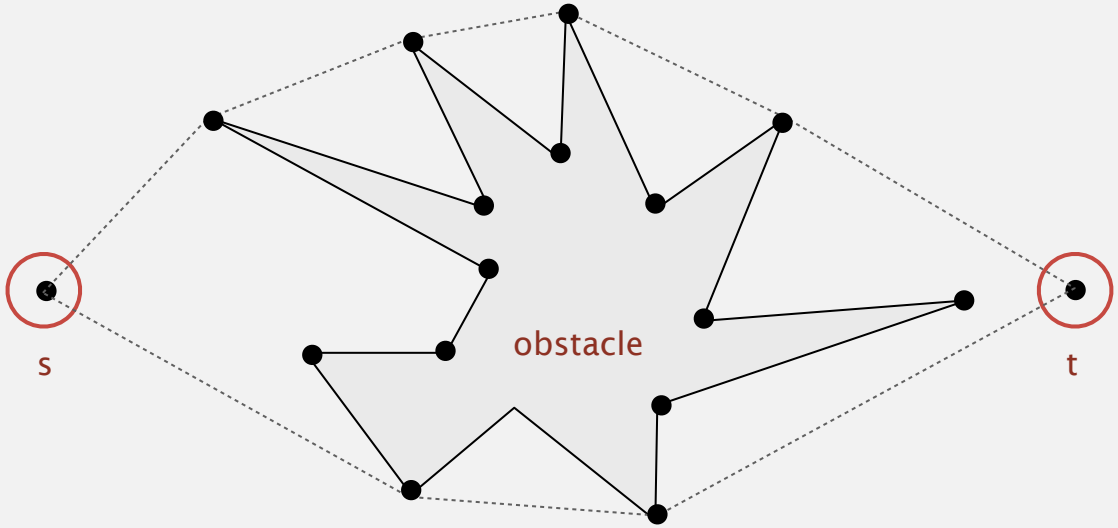
Mechanical algorithm. Hammer nails perpendicular to plane; stretch elastic rubber band around points.



http://www.idlcoyote.com/math_tips/convexhull.html

Convex hull application: motion planning

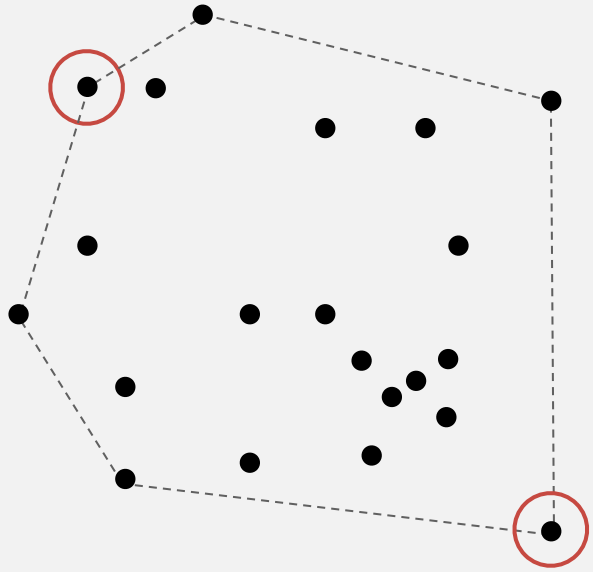
Robot motion planning. Find shortest path in the plane from s to t that avoids a polygonal obstacle.



Fact. Shortest path is either straight line from s to t or it is one of two polygonal chains of convex hull.

Convex hull application: farthest pair

Farthest pair problem. Given N points in the plane, find a pair of points with the largest Euclidean distance between them.

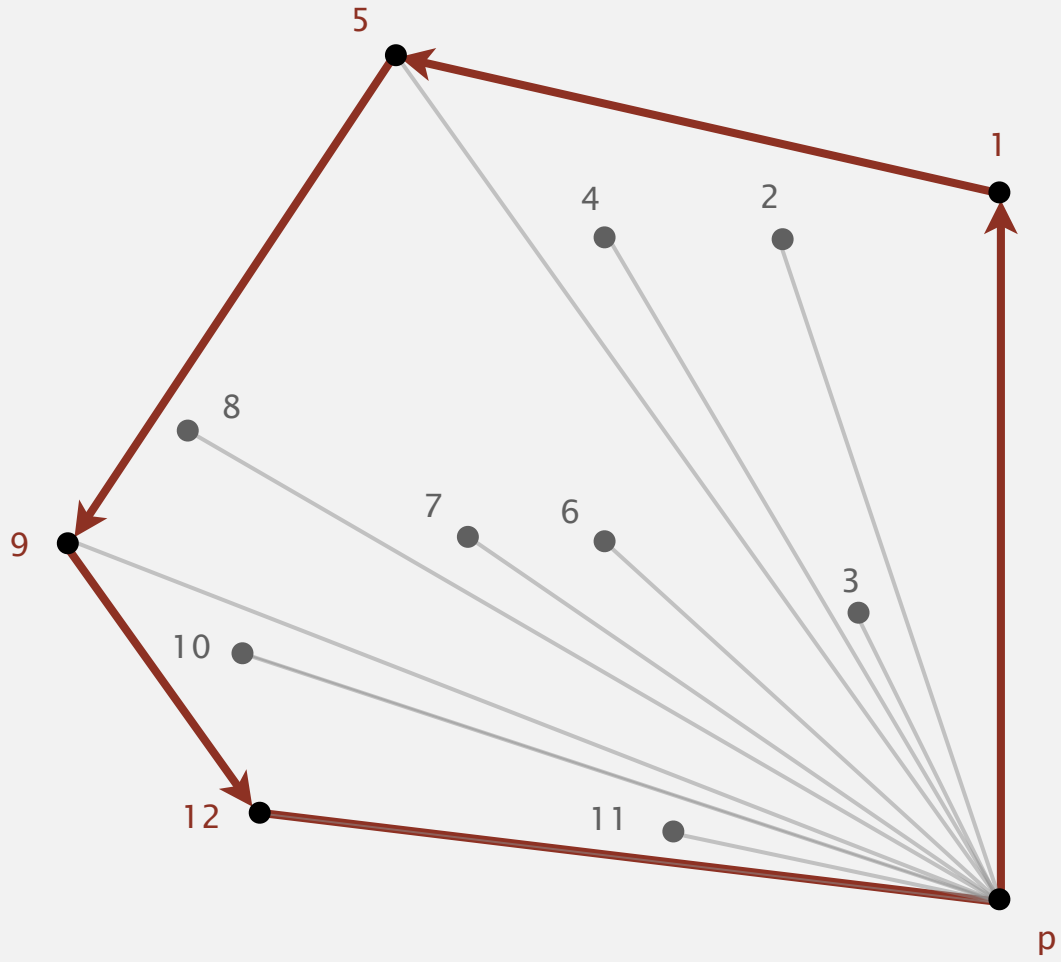


Fact. Farthest pair of points are extreme points on convex hull.

Convex hull: geometric properties

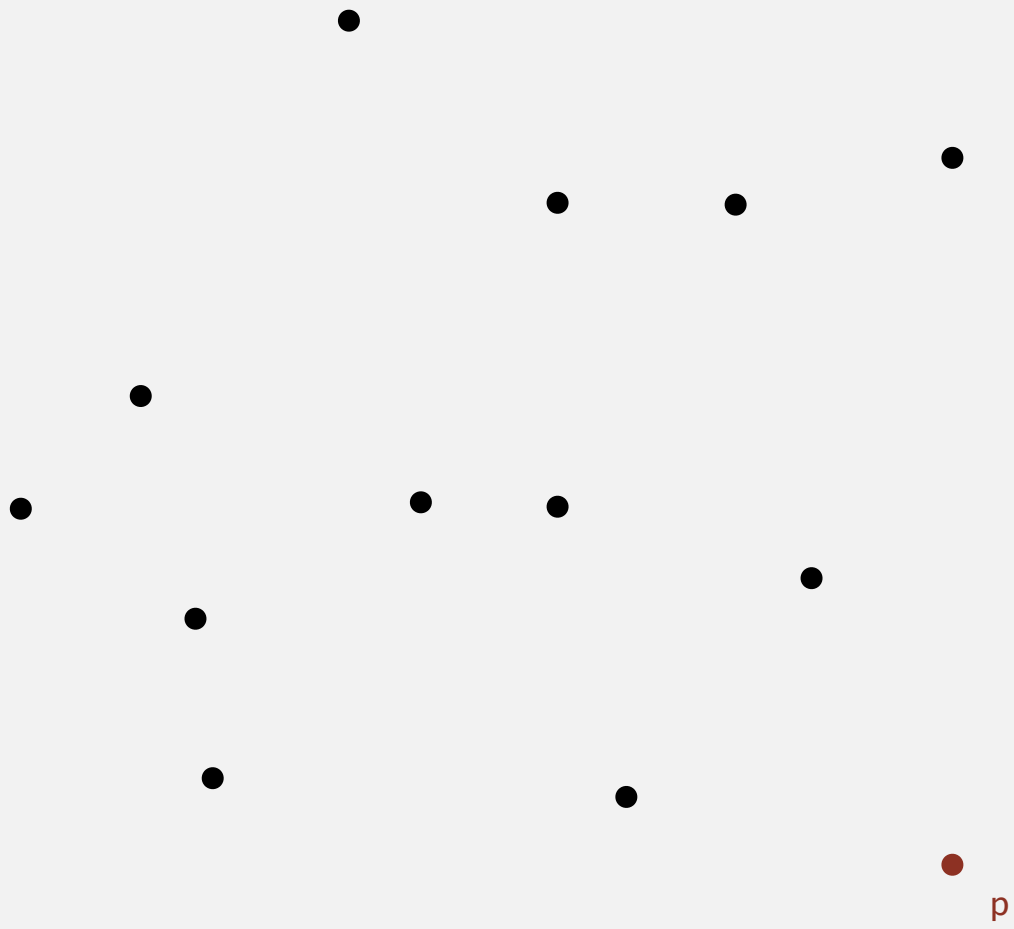
Fact. Can traverse the convex hull by making only counterclockwise turns.

Fact. The vertices of convex hull appear in increasing order of polar angle with respect to point p with lowest y-coordinate.



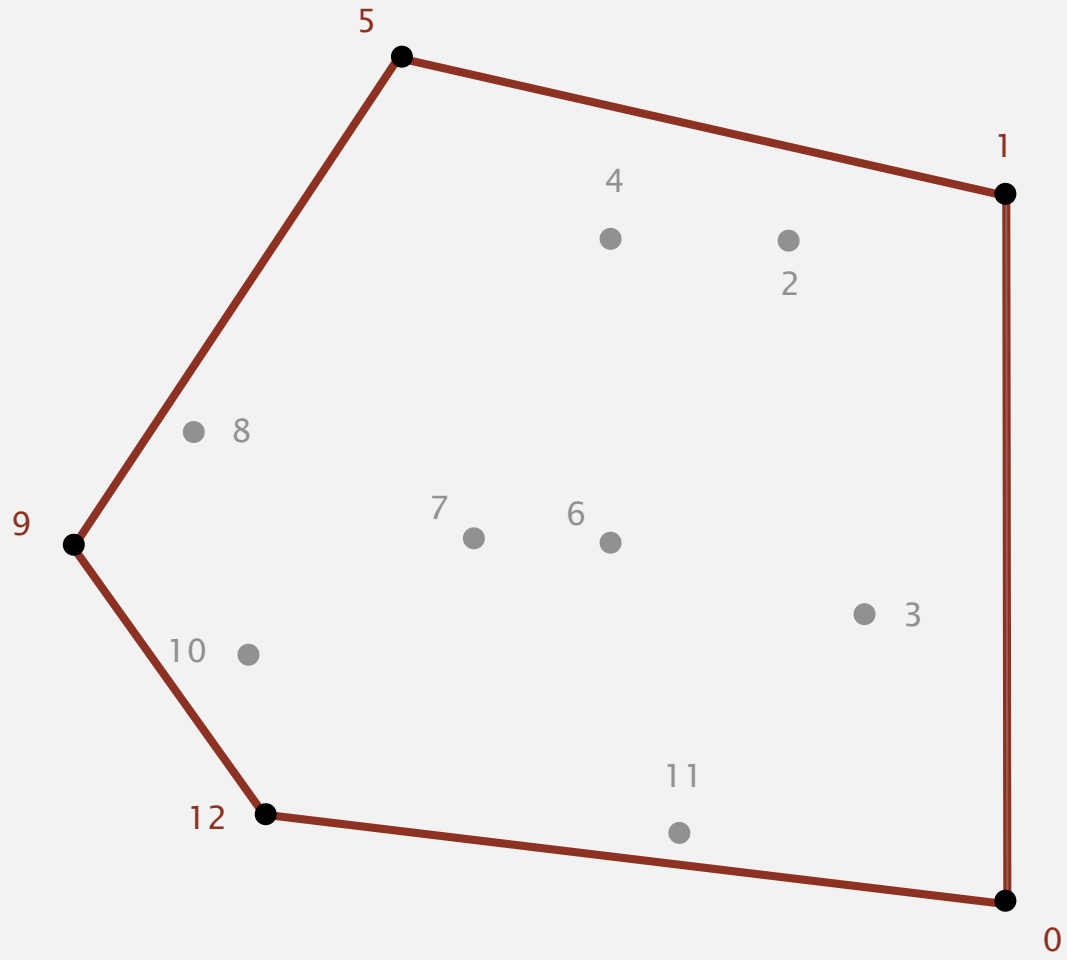
Graham scan demo

- Choose point p with smallest y-coordinate.
- Sort points by polar angle with p.
- In sorted order: Add point, then discard old points until all turns are ccw.



Graham scan demo

- Choose point p with smallest y-coordinate.
- Sort points by polar angle with p.
- In sorted order: Add point, then discard old points until all turns are ccw.



Graham scan: implementation challenges

Q. How to find point p with smallest y-coordinate?

A. Define a total order, comparing by y-coordinate. [next lecture]

Q. How to sort points by polar angle with respect to p ?

A. Define a total order for each point p. [next lecture]

Q. How to determine whether $p_1 \rightarrow p_2 \rightarrow p_3$ is a counterclockwise turn?

A. Computational geometry. [next two slides]

Q. How to sort efficiently?

A. Mergesort sorts in $N \log N$ time. [next lecture]

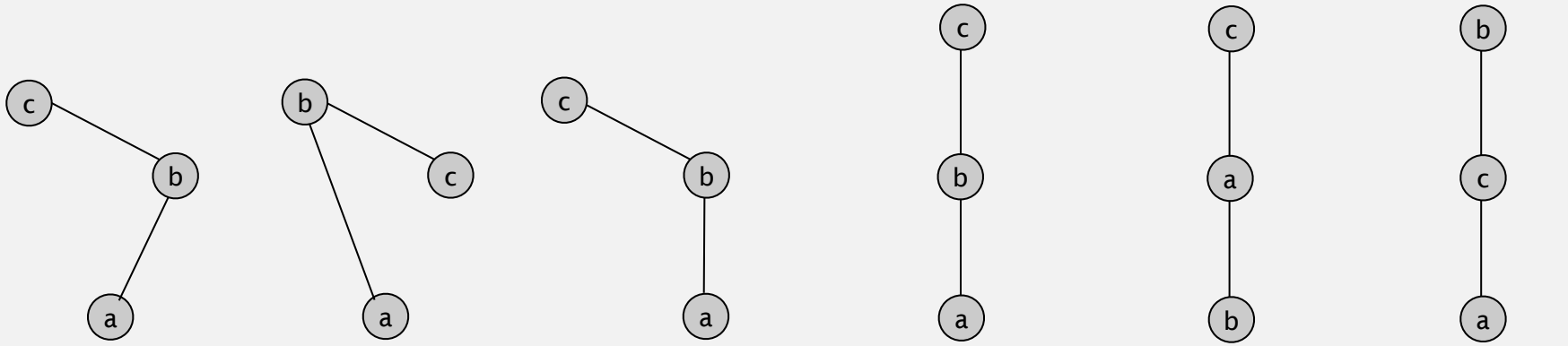
Q. How to handle degeneracies (three or more points on a line)?

A. Requires some care, but not hard. [see booksite]

Implementing ccw

CCW. Given three points a, b, and c, is $a \rightarrow b \rightarrow c$ a counterclockwise turn?

is c to the left of the ray a→b



Lesson. Geometric primitives are tricky to implement.

- Dealing with degenerate cases.
- Coping with floating-point precision.

Implementing ccw

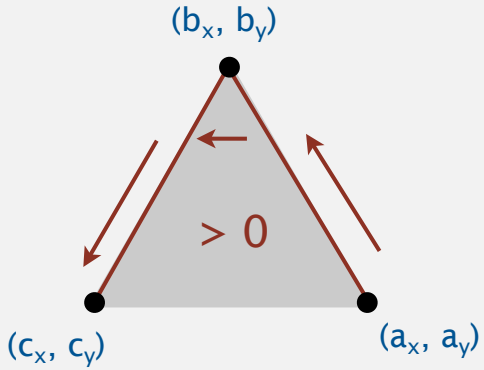
CCW. Given three points a, b, and c, is $a \rightarrow b \rightarrow c$ a counterclockwise turn?

- Determinant of special matrix gives 2x signed area of planar triangle.

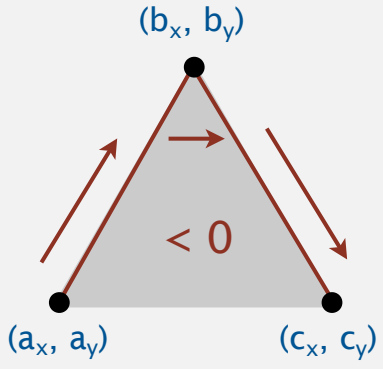
$$2 \times Area(a,b,c) = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} = (b_x - a_x)(c_y - a_y) - (b_y - a_y)(c_x - a_x)$$

$|(b - a) \times (c - a)|$

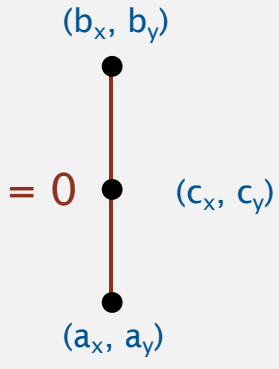
- If signed area > 0, then $a \rightarrow b \rightarrow c$ is counterclockwise.
- If signed area < 0, then $a \rightarrow b \rightarrow c$ is clockwise.
- If signed area $= 0$, then $a \rightarrow b \rightarrow c$ are collinear.



counterclockwise



clockwise



collinear

Immutable point data type

```
public class Point2D
{
   private final double x;
   private final double y;

   public Point2D(double x, double y)
   {
      this.x = x;
      this.y = y;
   }

   ...

   public static int ccw(Point2D a, Point2D b, Point2D c)
   {
      double area2 = (b.x-a.x)*(c.y-a.y) - (b.y-a.y)*(c.x-a.x);
      if      (area2 < 0) return -1;  // clockwise
      else if (area2 > 0) return +1;  // counter-clockwise
      else                return  0;  // collinear
   }
}
```

danger of floating-point roundoff error

Graham scan: implementation

Simplifying assumptions. No three points on a line; at least 3 points.

```
Stack<Point2D> hull = new Stack<Point>();

Arrays.sort(p, Point2D.Y_ORDER);
Arrays.sort(p, p[0].BY_POLAR_ORDER);

hull.push(p[0]);
hull.push(p[1]);

for (int i = 2; i < N; i++) {
   Point2D top = hull.pop();
   while (Point2D.ccw(hull.peek(), top, p[i]) <= 0)
      top = hull.pop();
   hull.push(top);
   hull.push(p[i]);
}
```

p[0] is now point with lowest y-coordinate (can do more efficiently without sorting)

sort by polar angle with respect to p[0]

definitely on hull

discard points that would create clockwise turn

add p[i] to putative hull

Running time. $N \log N$ for sorting and linear for rest.

Pf. $N \log N$ for sorting; each point pushed and popped at most once.