Computational Geometry

References:

Algorithms in C (2nd edition), Chapters 24-25

Algorithm Design and Analysis (3rd edition)

Programming Challenges ()

http://www.cs.princeton.edu/introalgsds/71primitives

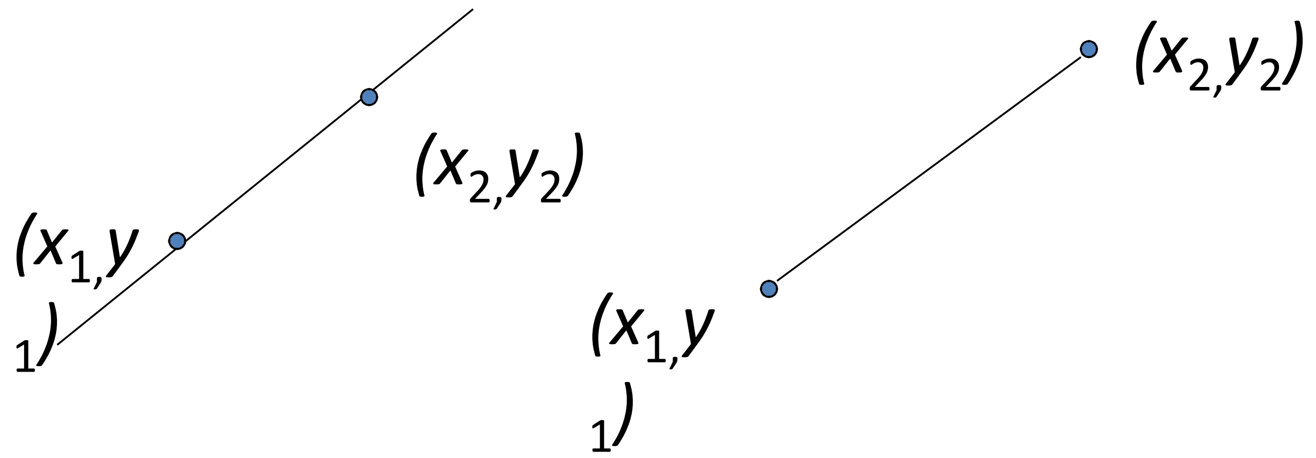
http://www.cs.princeton.edu/introalgsds/72hull

Computational Geometry

- Applications
  - Computer graphics
  - Robotics
  - VLSI design
  - Computer-aided design
  - Statistics
- Input: a description of a set of geometric objects
  - a set of points
  - a set of line segments
- Output: a response to a query about the objects
  - Whether any of the lines intersects
- We focus on two-dimensional problems
  - Computational geometry algorithms in the plane

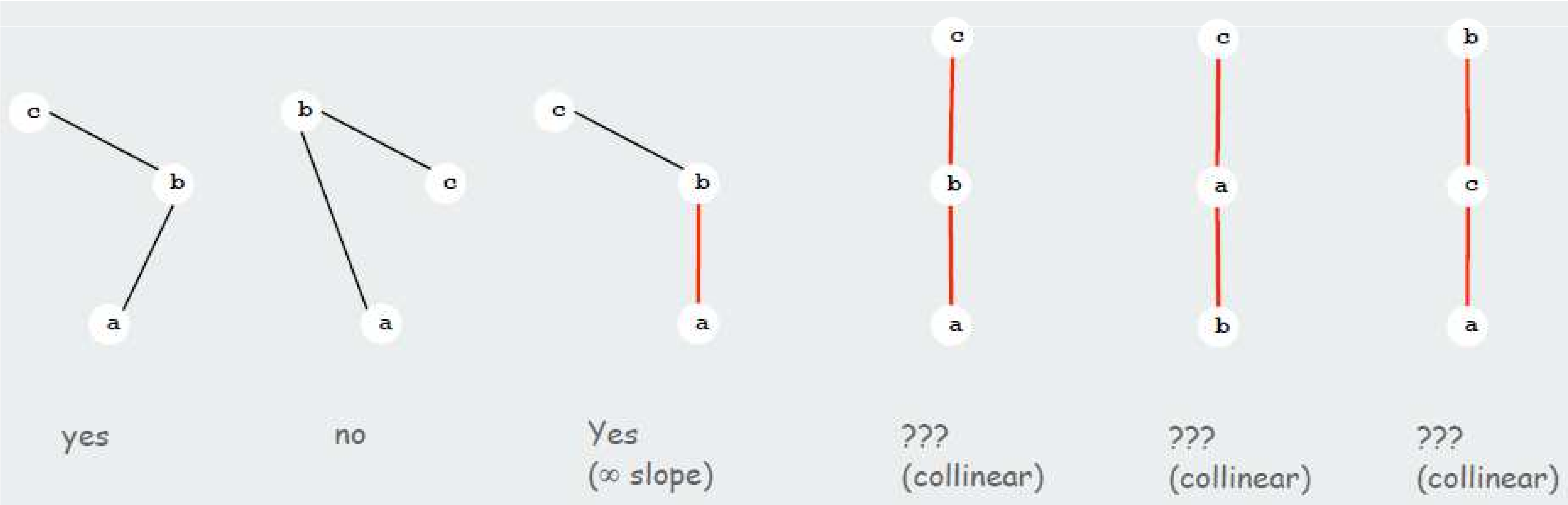
Computational Geometry

- Intersection of line segment
- Basic geometric objects
  - Point : (x, y)
  - Line : $\{(x_1,y_1) , (x_2,y_2) \}$
  - Line segment : size of line is given



Counterclockwise

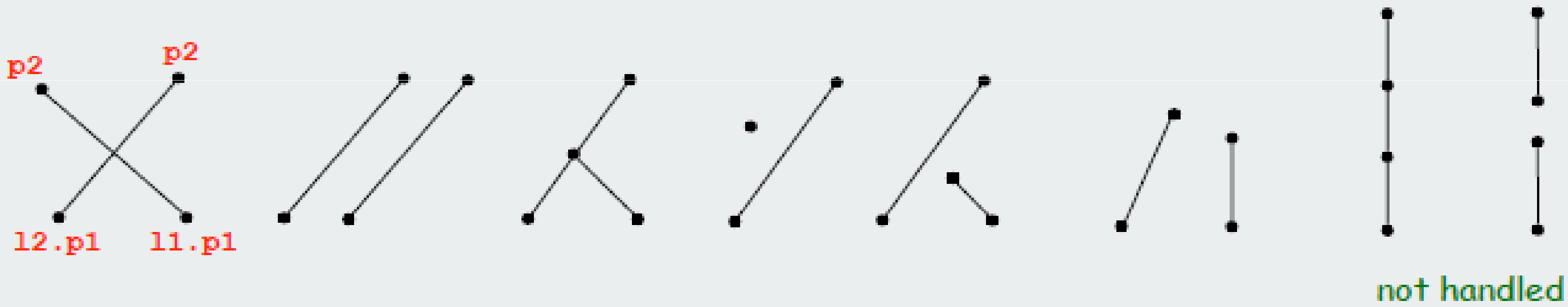
- Given three point a, b, and c, is a-b-c a counterclockwise turn?
- Idea: compare slopes.



Segment intersect

Intersect: Given two line segments, do they intersect?

- Idea 1: find intersection point using algebra and check.
- Idea 2: check if the endpoints of one line segment are on different "sides" of the other line segment.
- 4 ccw computations.




```
public static boolean intersect(Line l1, Line l2)
{
   int test1, test2;
   test1 = Point.ccw(l1.p1, l1.p2, l2.p1)
           * Point.ccw(l1.p1, l1.p2, l2.p2);
   test2 = Point.ccw(l2.p1, l2.p2, l1.p1)
           * Point.ccw(l2.p1, l2.p2, l1.p2);
   return (test1 <= 0) && (test2 <= 0);
}
```

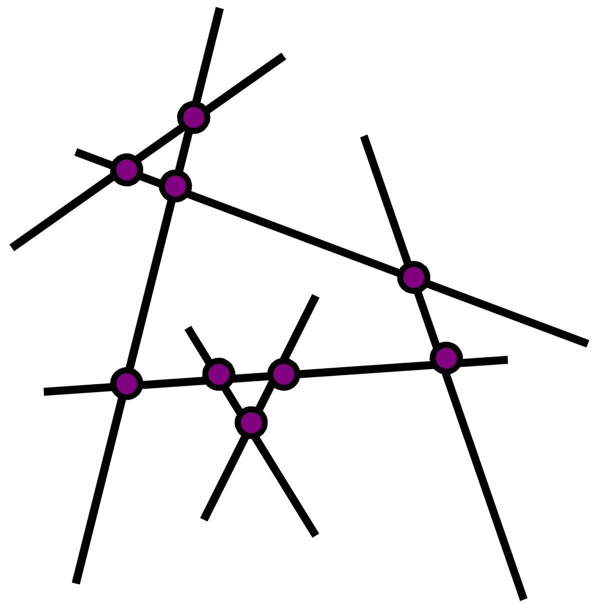
```
SEGMENTS-INTERSECT(p1, p2, p3, p4)
d1 ← DIRECTION(p3, p4, p1)
d2 ← DIRECTION(p3, p4, p2)
d3 ← DIRECTION(p1, p2, p3)
d4 ← DIRECTION(p1, p2, p4)
if ((d1 > 0 and d2 < 0) or (d1 < 0 and d2 > 0)) and
        ((d3 > 0 and d4 < 0) or (d3 < 0 and d4 > 0))
   then return TRUE
elseif d1 = 0 and ON-SEGMENT(p3, p4, p1)
   then return TRUE
elseif d2 = 0 and ON-SEGMENT(p3, p4, p2)
   then return TRUE
elseif d3 = 0 and ON-SEGMENT(p1, p2, p3)
   then return TRUE
elseif d4 = 0 and ON-SEGMENT(p1, p2, p4)
   then return TRUE
else return FALSE
```

```
DIRECTION(p_i, p_j, p_k)
    return (p_k − p_i) × (p_j − p_i)
```

```
ON-SEGMENT(p_i, p_j, p_k)
    if min(x_i, x_j) ≤ x_k ≤ max(x_i, x_j) and min(y_i, y_j) ≤ y_k ≤ max(y_i, y_j)
        then return TRUE
        else return FALSE
```

Intersection of line segments

Is there a pair of line segments intersecting each other?



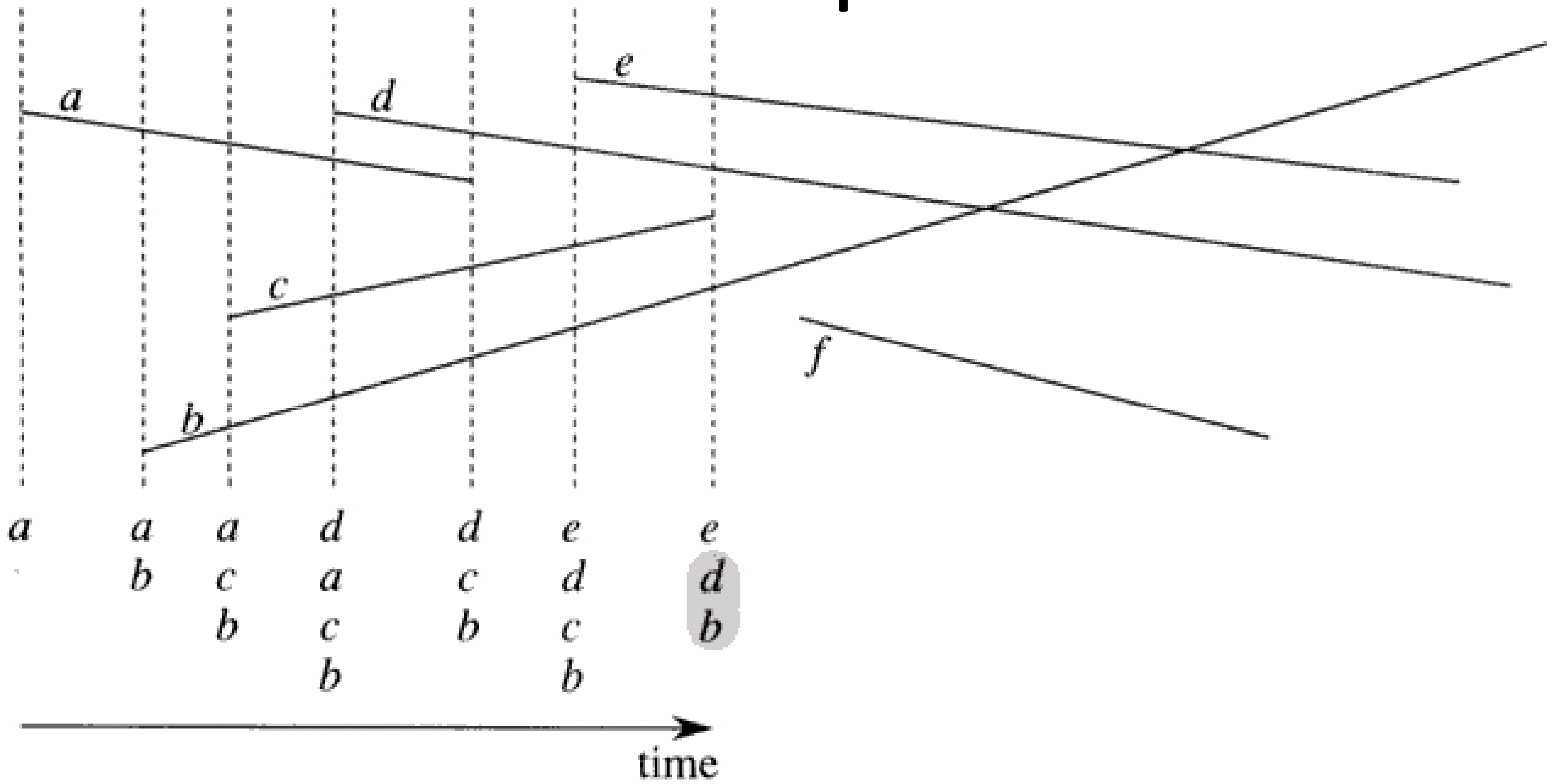
- Naive algorithm: Check each two segments for intersection. Complexity: O(n^2).

Plane Sweep – Algorithm

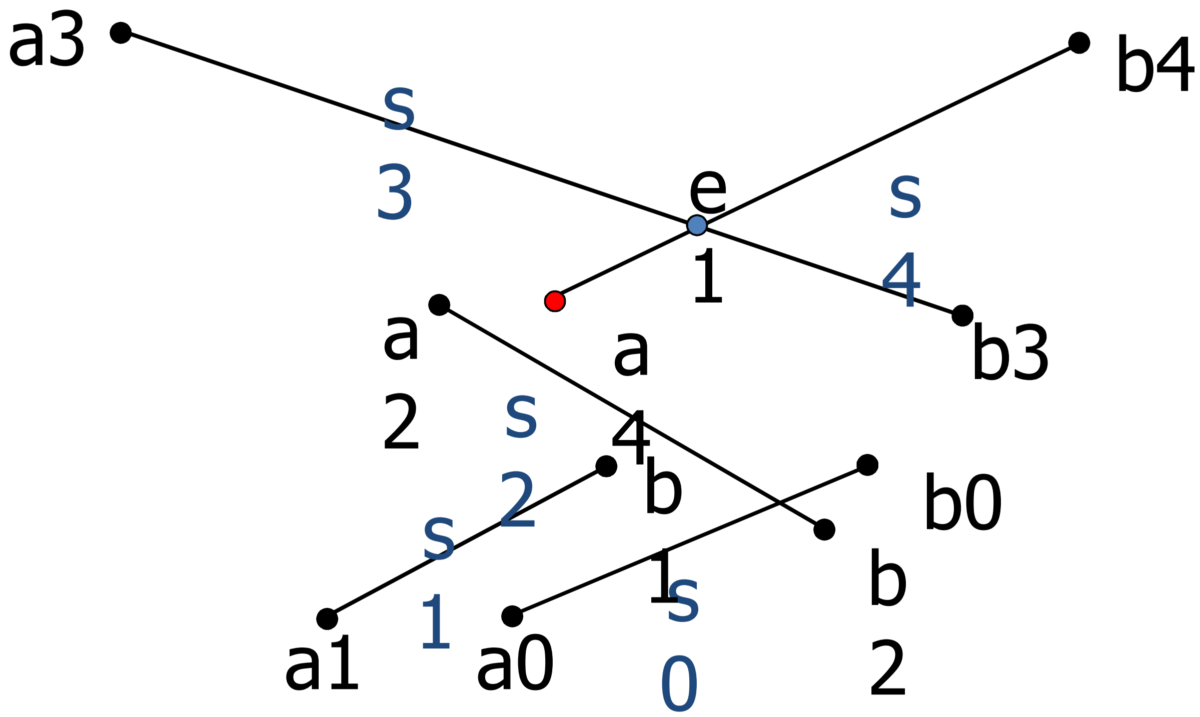
- Problem: Given *n* segments in the plane, compute all their intersections.
- Assume:
  - No line segment is vertical.
  - No two segments are collinear.
  - No three segments intersect at a common point.
- Event is any end point or intersection point.
- Sweep the plane using a vertical line.
- Maintain two data structures:
  - Event priority queue – sorted by *x* coordinate.
  - Sweep line status – Stores segments currently intersected by sweep line, sorted by *y* coordinate.

```
ANY-SEGMENTS-INTERSECT(S)
T ← Ø
sort the endpoints of the segments in S from left to right,
        breaking ties by putting left endpoints before right endpoints
        and breaking further ties by putting points with lower
        y-coordinates first
for each point p in the sorted list of endpoints
    do if p is the left endpoint of a segment s
          then INSERT(T, s)
               if (ABOVE(T, s) exists and intersects s)
                     or (BELOW(T, s) exists and intersects s)
                 then return TRUE
       if p is the right endpoint of a segment s
          then if both ABOVE(T, s) and BELOW(T, s) exist
                     and ABOVE(T, s) intersects BELOW(T, s)
                 then return TRUE
               DELETE(T, s)
return FALSE
```

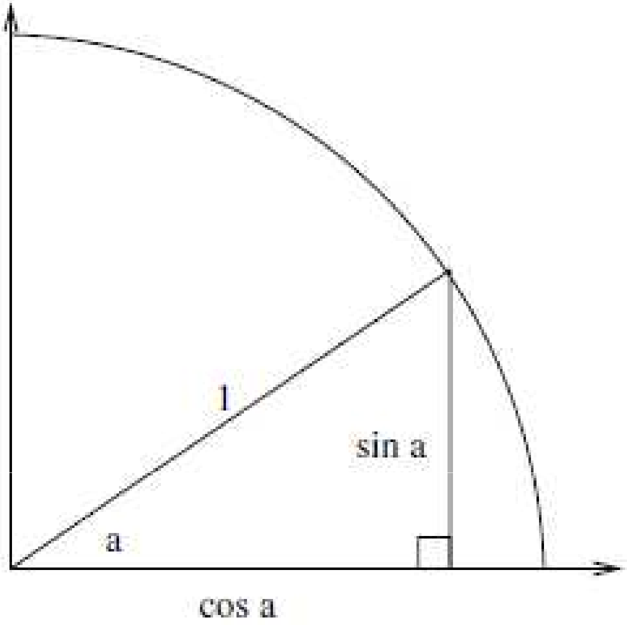
Example

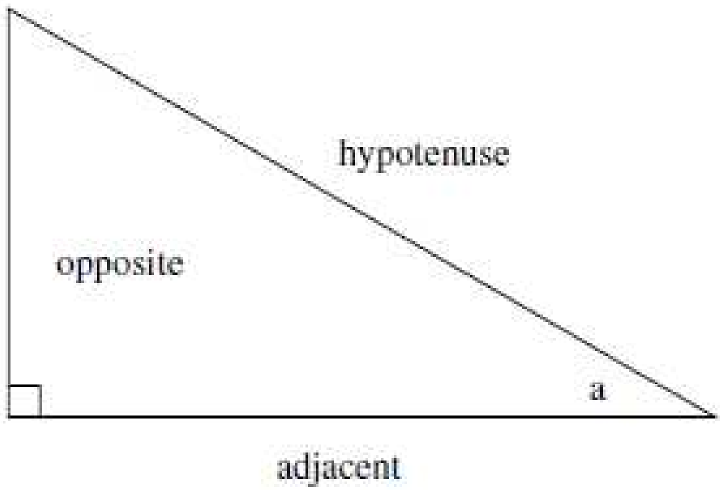


Example



Triangles and Trigonometry

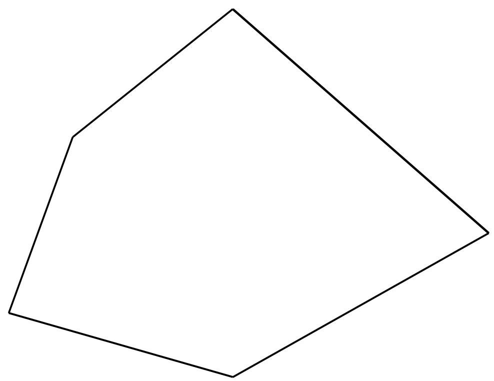




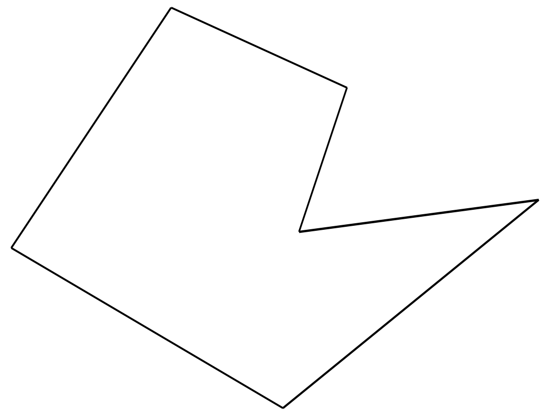
$$\cos(a) = \frac{|\text{adjacent}|}{|\text{hypotenuse}|}, \quad \sin(a) = \frac{|\text{opposite}|}{|\text{hypotenuse}|}, \quad \tan(a) = \frac{|\text{opposite}|}{|\text{adjacent}|}$$

Basic geometric objects(contd)

– Polygon



Convex

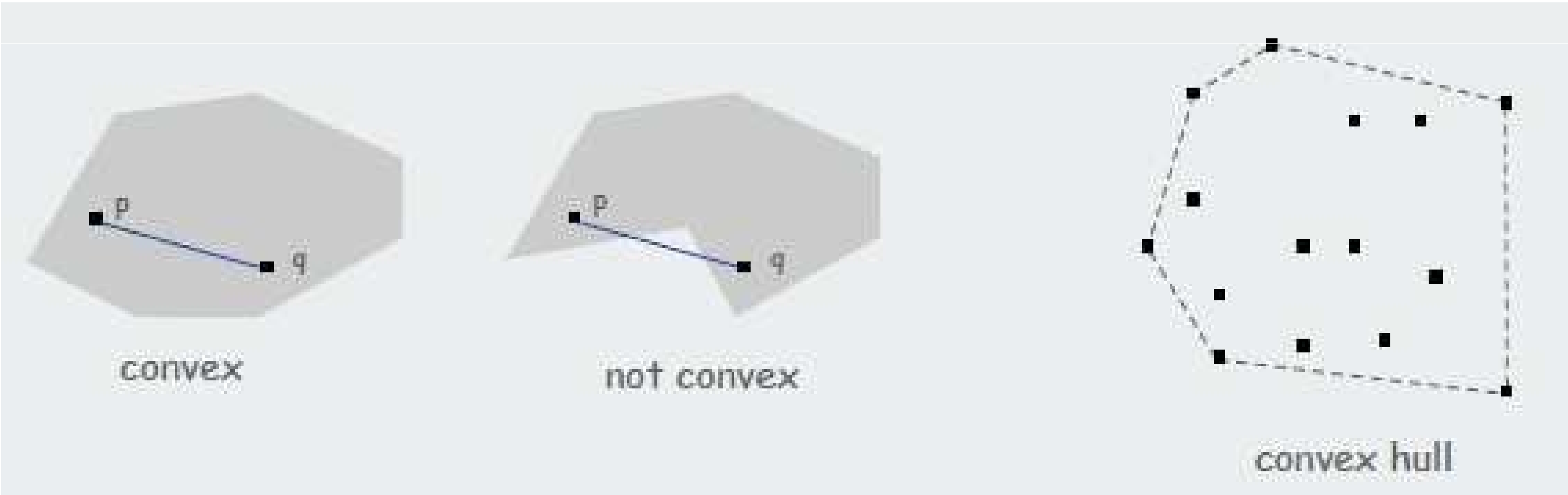


Non-convex

– Convex : Each indegree less than 180

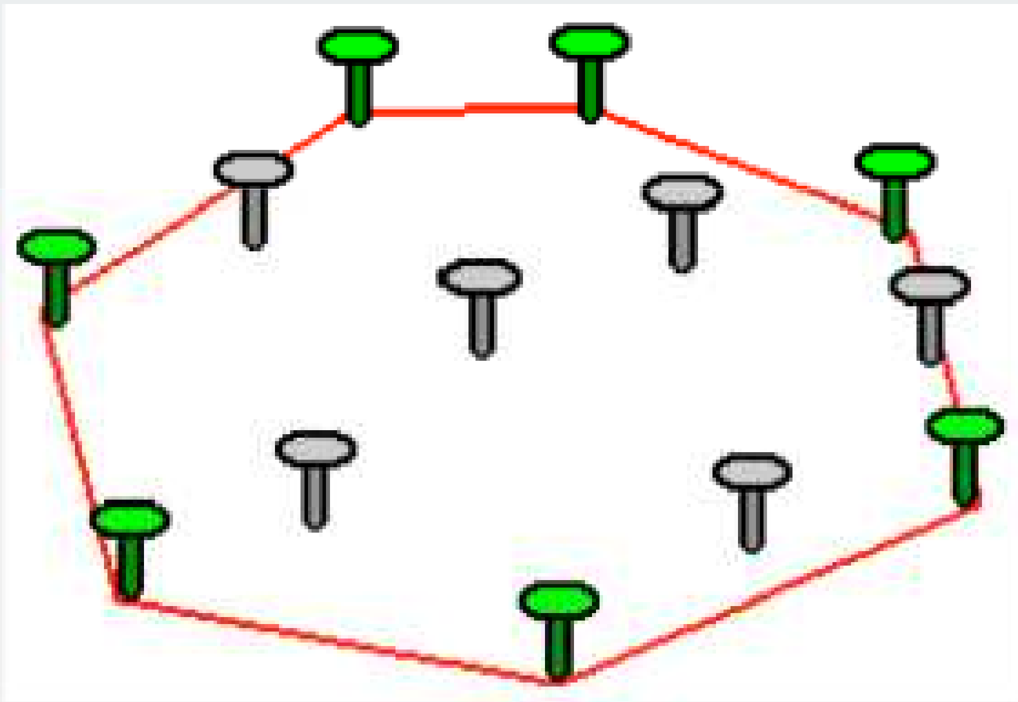
Convex Hull

- A set of points is convex if for any two points p and q in the set, the line segment pq is completely in the set.
- Convex hull = Smallest convex set containing all the points.



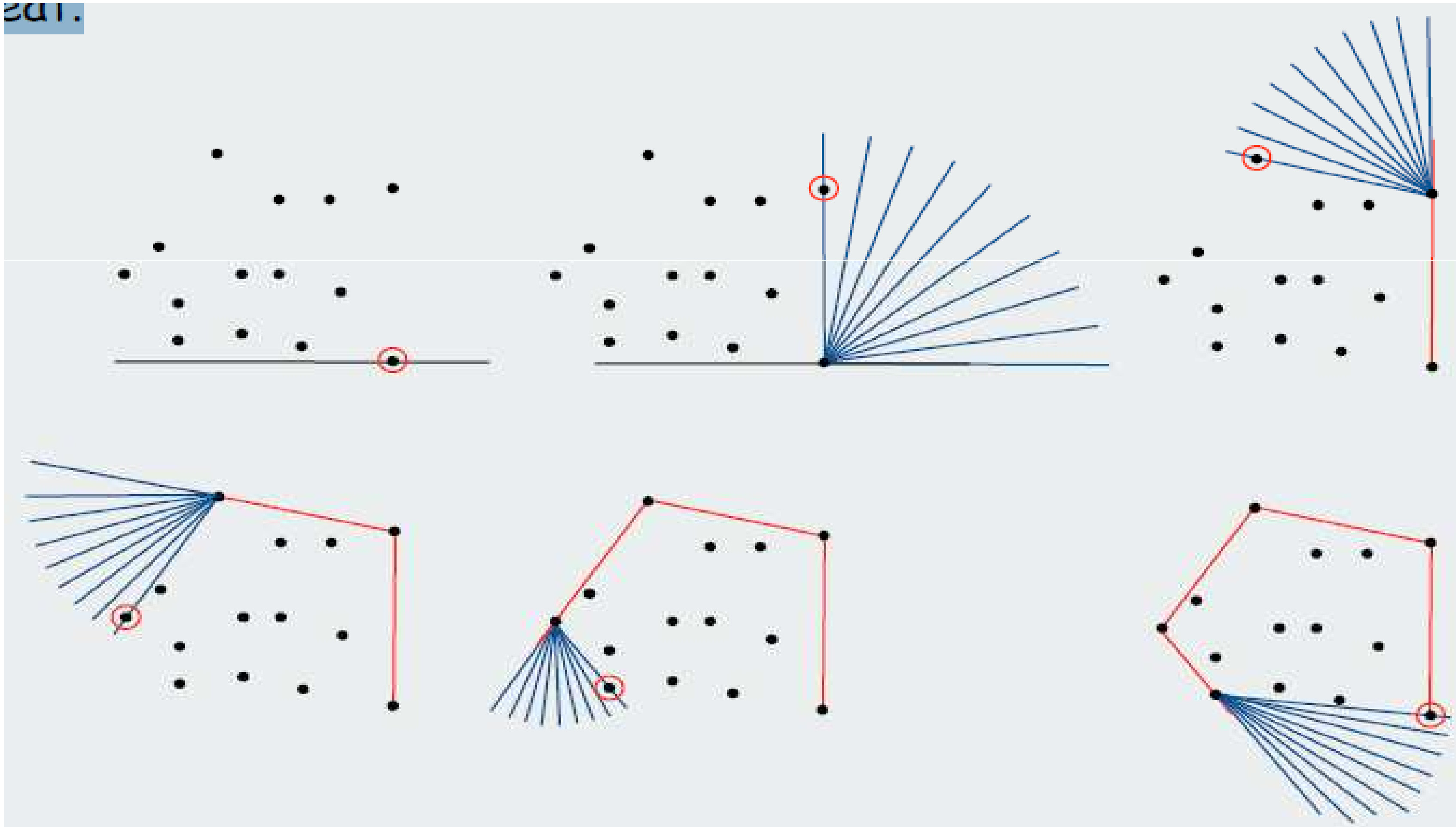
Mechanical algorithm

- Hammer nails perpendicular to plane; stretch elastic rubber band around points



http://www.dfanning.com/math_tips/convexhull_1.gif

- Start with point with smallest y-coordinate.
- Rotate sweep line around current point in ccw direction.
- First point hit is on the hull. Repeat.

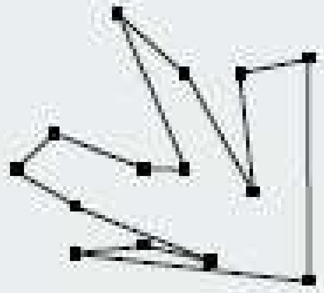


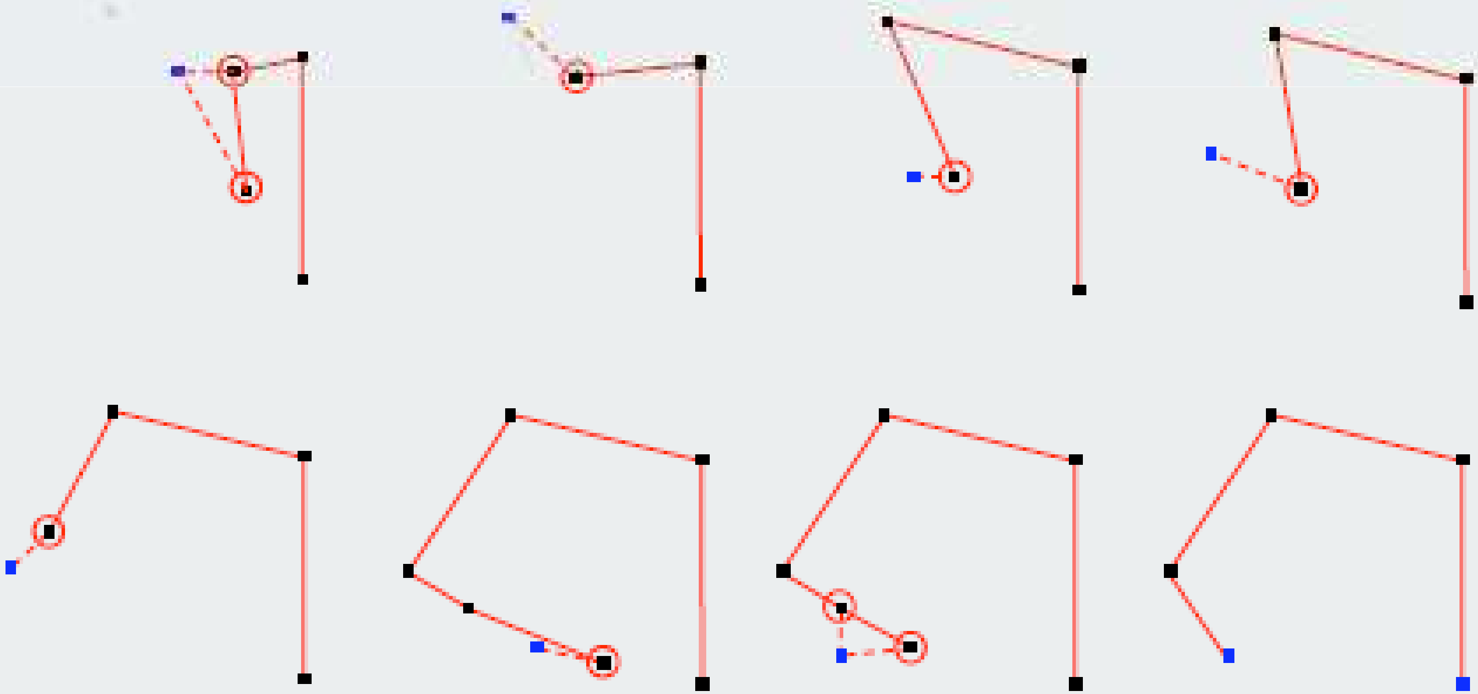
Implementation

- Compute angle between current point and all remaining points.
- Pick smallest angle larger than current angle.
- Parameters
  - N = number of points.
  - h = number of points on the hull.

Graham scan.

- Choose point p with smallest y-coordinate.
- Sort points by polar angle with p to get simple polygon.
- Consider points in order, and discard those that would create a clockwise turn.





GRAHAM-SCAN(Q)

```
let p_0 be the point in Q with the minimum y-coordinate,
        or the leftmost such point in case of a tie
let ⟨p_1, p_2, ..., p_m⟩ be the remaining points in Q,
        sorted by polar angle in counterclockwise order around p_0
        (if more than one point has the same angle, remove all but
        the one that is farthest from p_0)
PUSH(p_0, S)
PUSH(p_1, S)
PUSH(p_2, S)
for i ← 3 to m
    do while the angle formed by points NEXT-TO-TOP(S), TOP(S),
                and p_i makes a nonleft turn
            do POP(S)
        PUSH(p_i, S)
return S
```

Problems

- Skyline (~/1/105.html)
- Intersection (~/1/191.html)
- SCUD Buster (~/1/109.html)
- Moth Eradication (~/2/218.html)
- Lining Up (~/2/270.html)
- Intersecting line (~/3/378.html)
- Polygon (~/6/634.html)
- Convex Hull Finding (~/6/681.html)