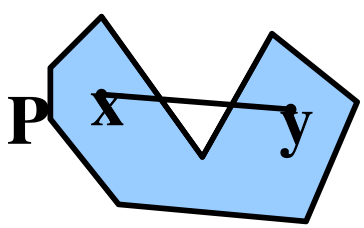
Convex Hull

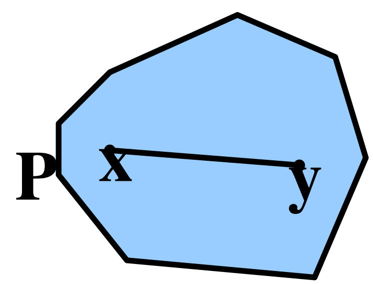
Jehn-Ruey Jiang
National Central University, Jhongli City, 32001, Taiwan

Convex vs. Concave

- A polygon P is convex if for every pair of points x and y in P, the line xy is also in P; otherwise, it is called concave.



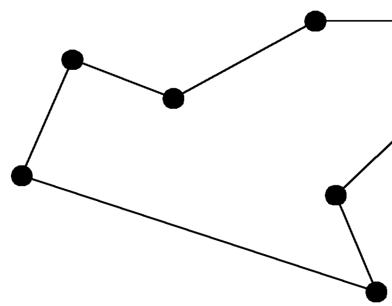
concave



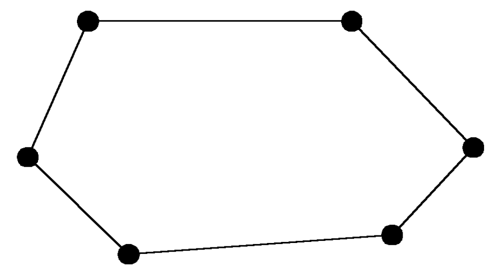
convex

The convex hull problem

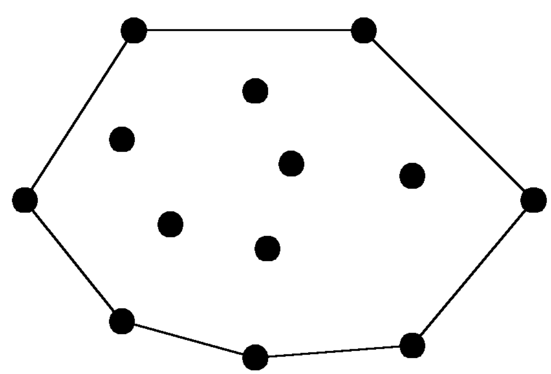
concave polygon:



convex polygon:



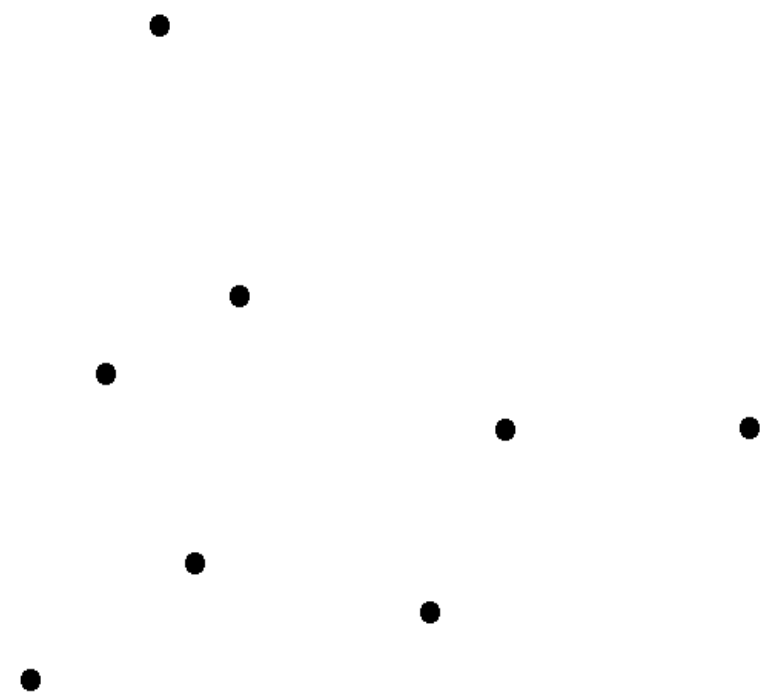
- The convex hull of a set of planar points is the smallest convex polygon containing all of the points.



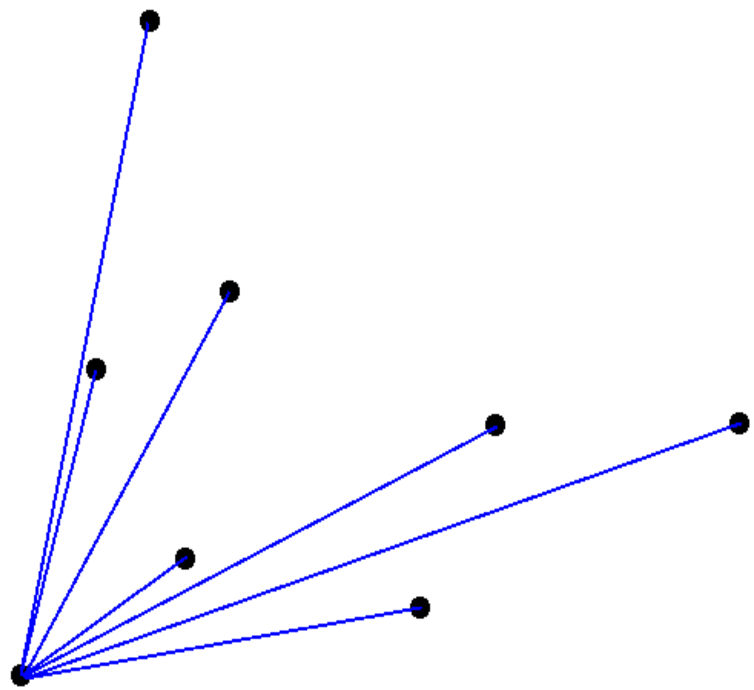
Graham's Scan

- Start at point guaranteed to be on the hull. (the point with the minimum y value)
- Sort remaining points by polar angles of vertices relative to the first point.
- Go through sorted points, keeping vertices of points that have left turns and dropping points that have right turns.

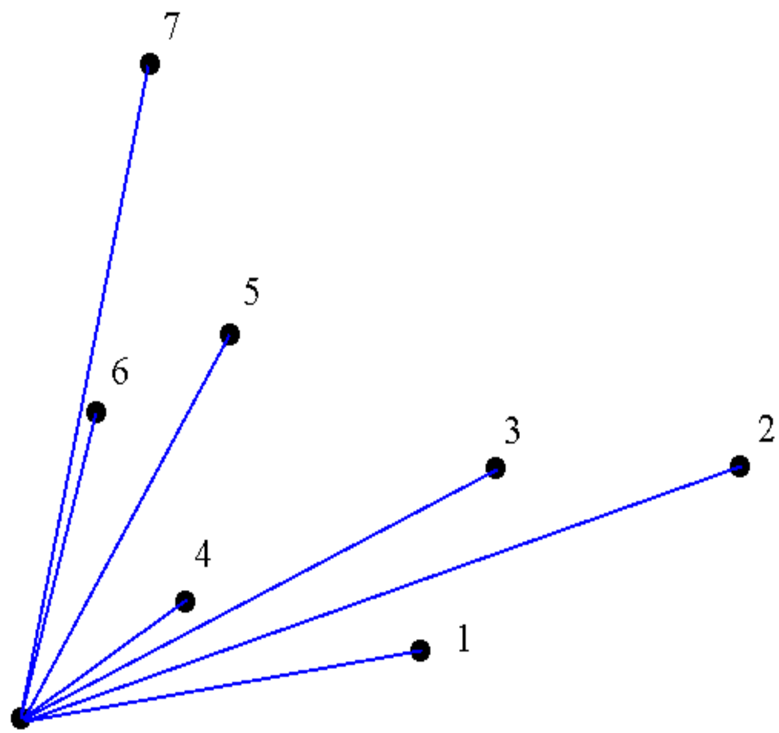
Graham's Scan



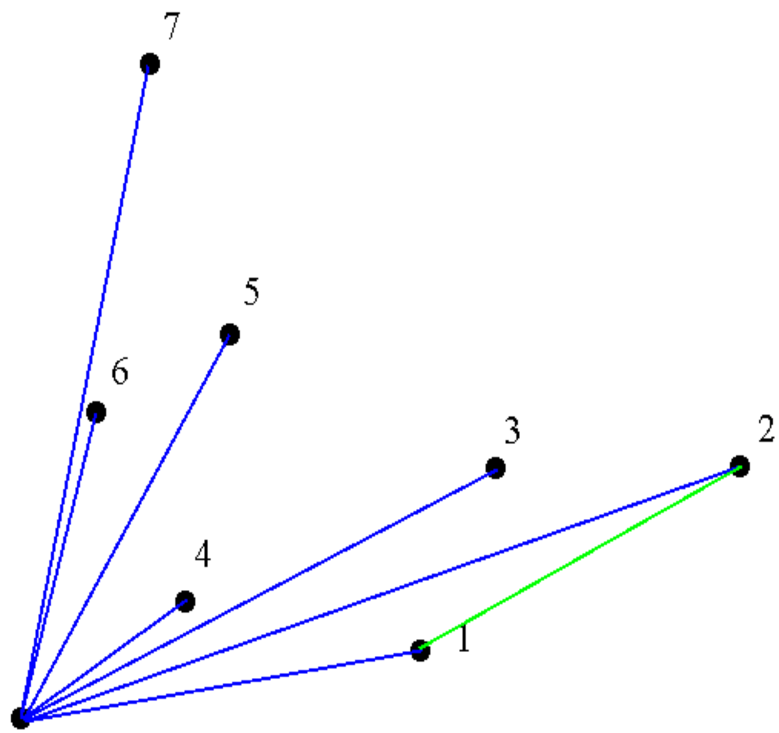
Graham’s Scan



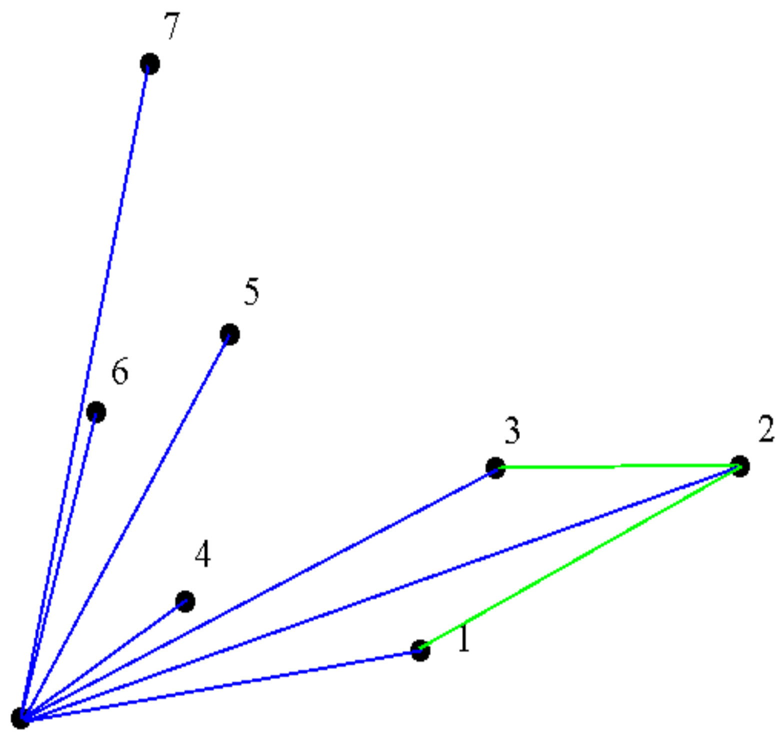
Graham's Scan



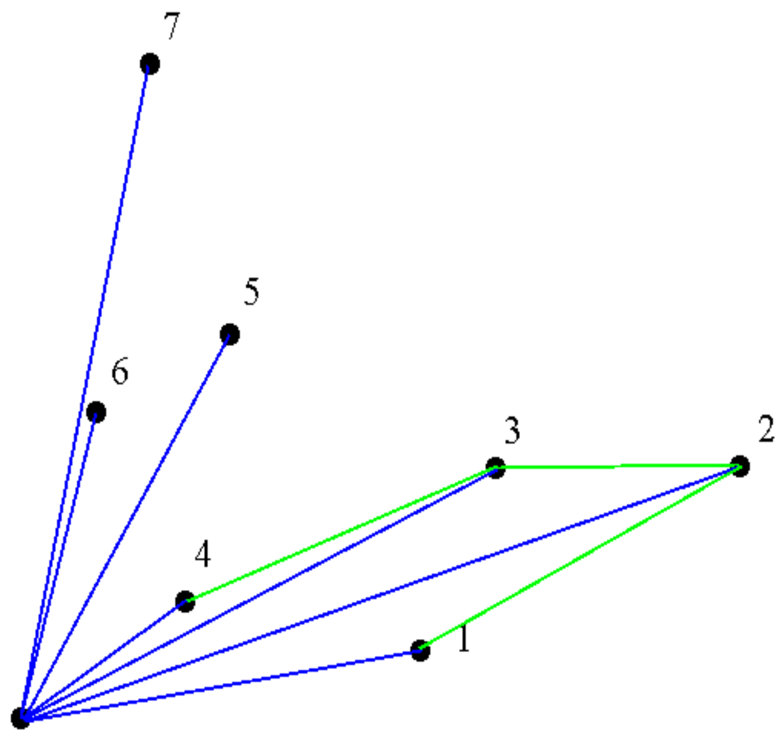
Graham’s Scan



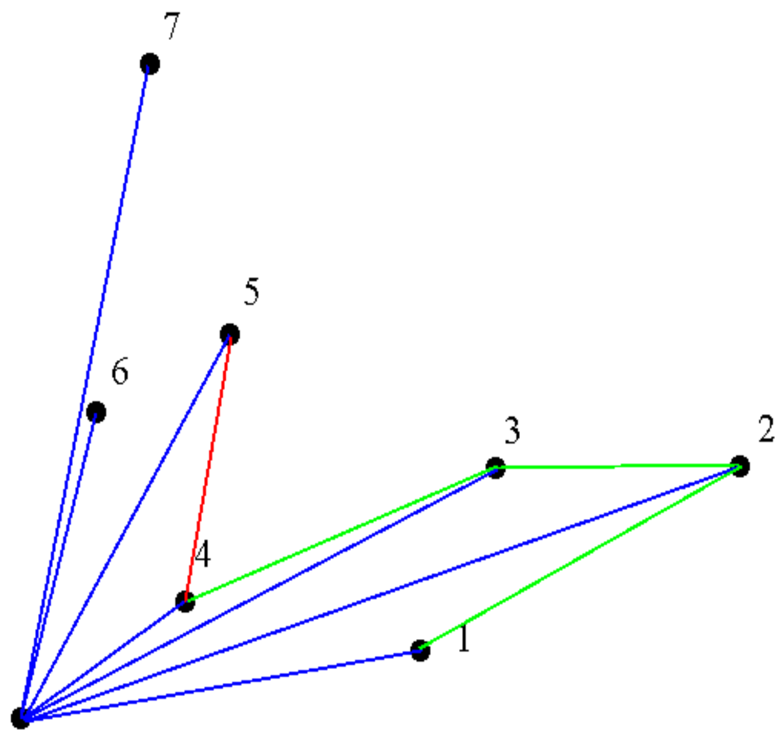
Graham's Scan



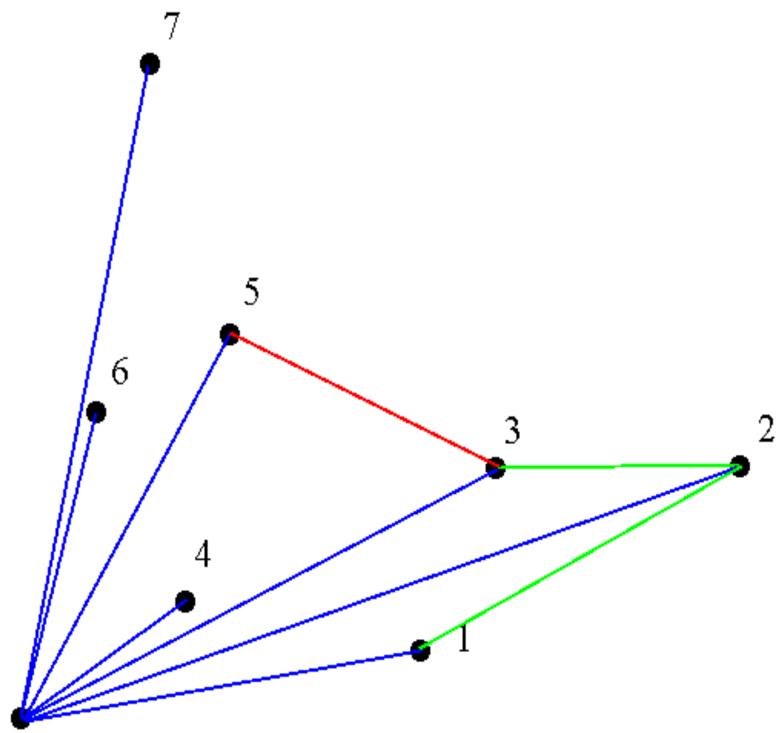
Graham’s Scan



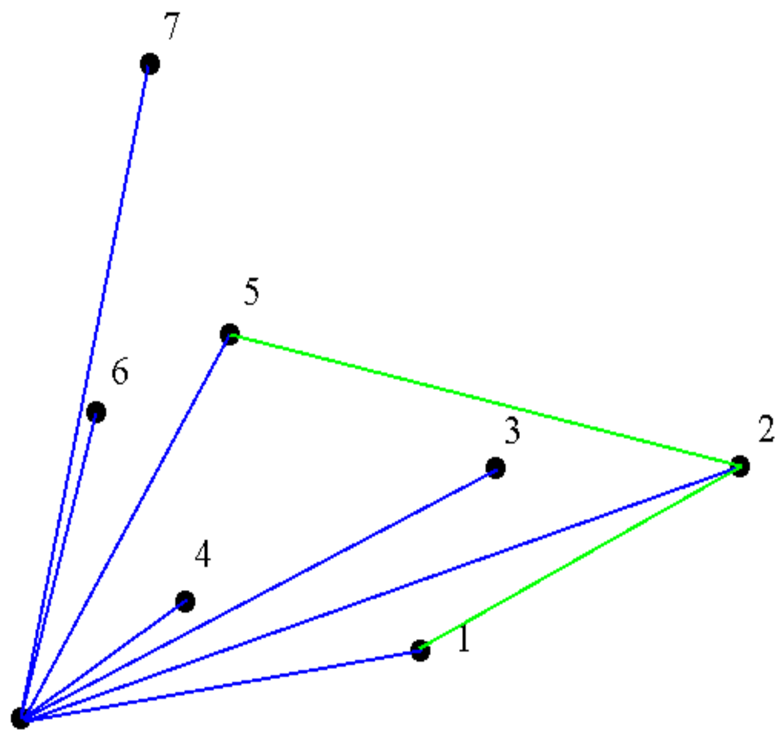
Graham’s Scan



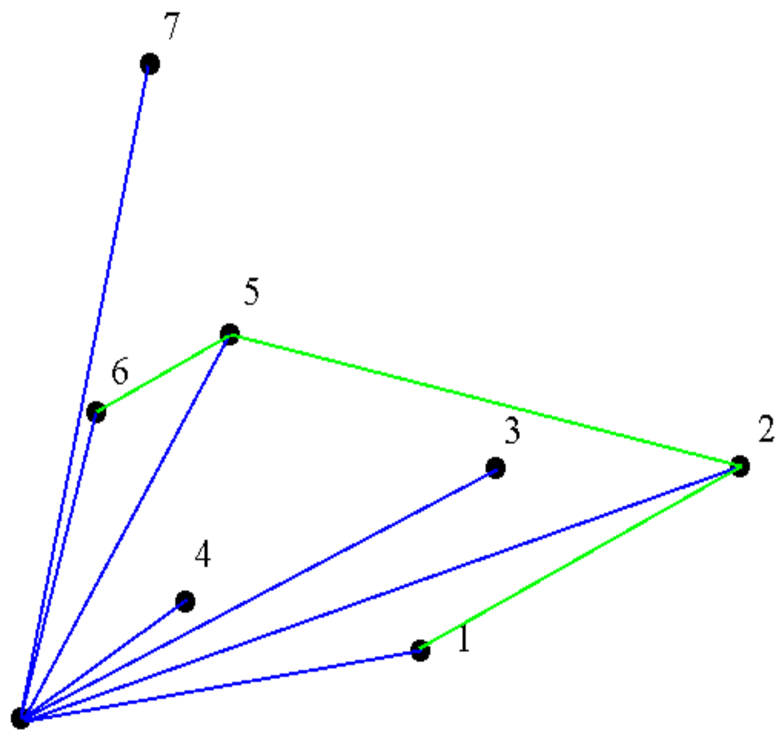
Graham’s Scan



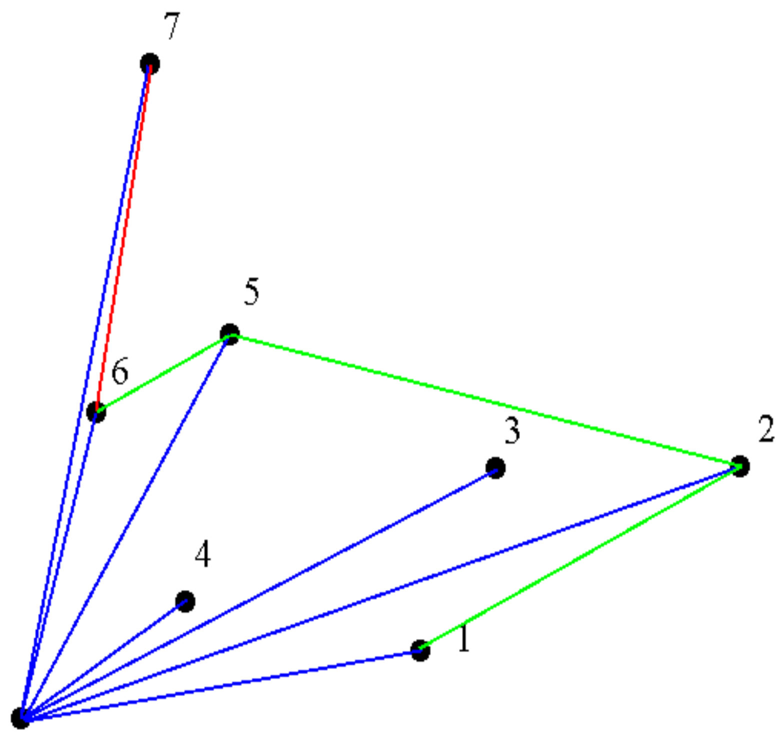
Graham's Scan



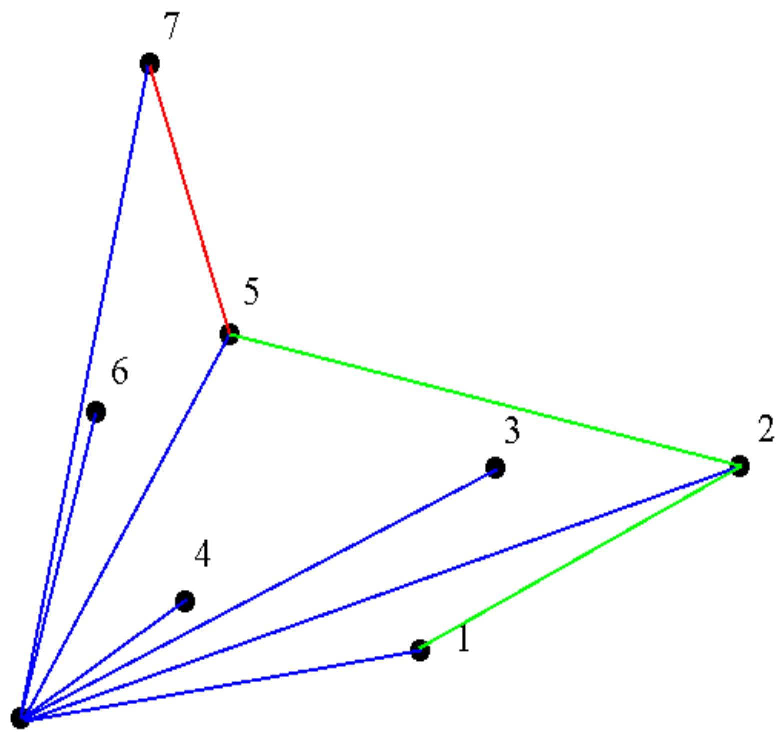
Graham’s Scan



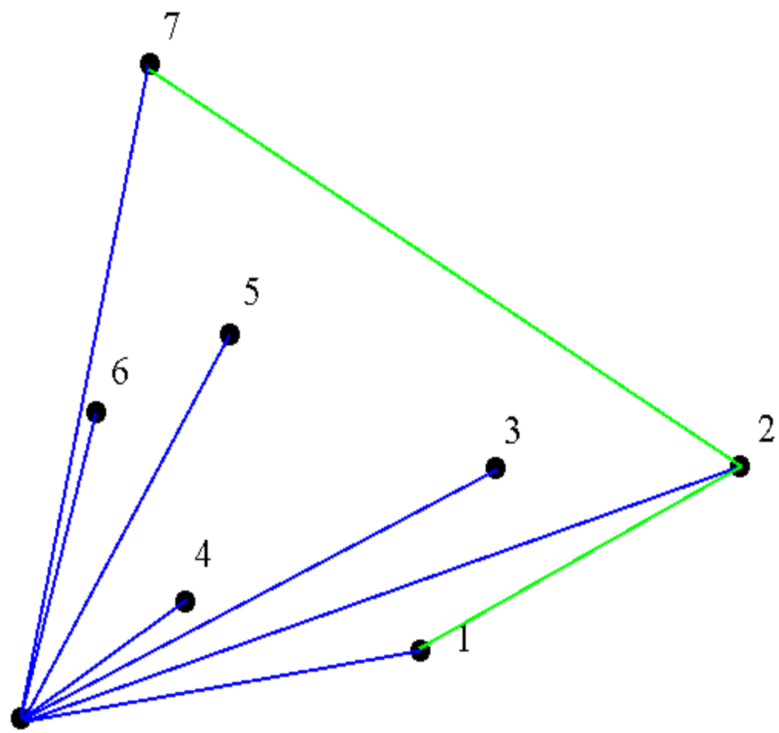
Graham's Scan



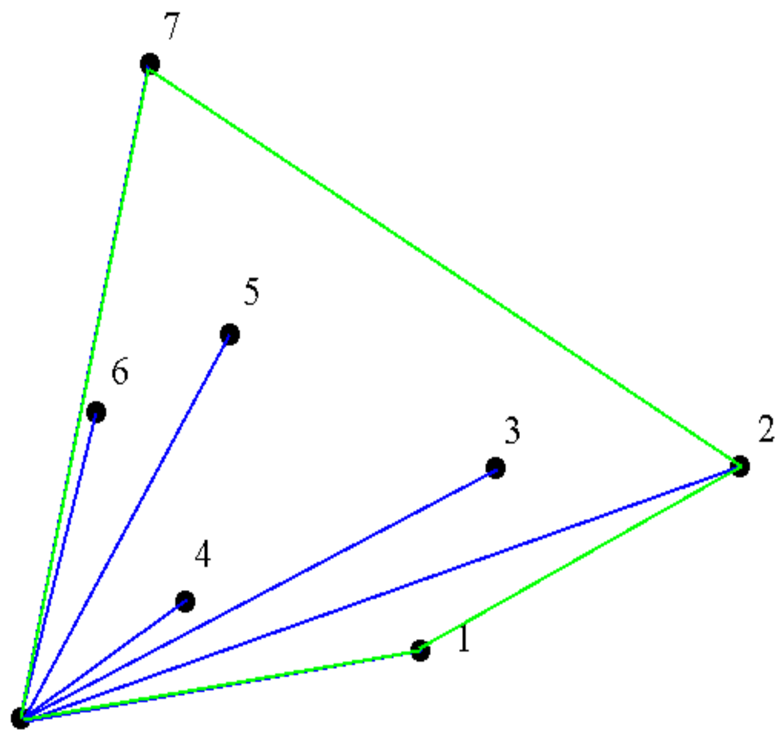
Graham’s Scan



Graham's Scan



Graham's Scan



Graham’s Runtime

- Graham’s scan is O(n log n) due to initial sort of angles.

■A more detailed algorithm

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 $\langle p_1, p_2, \ldots, p_m \rangle$ 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 $i \leftarrow 3$ **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

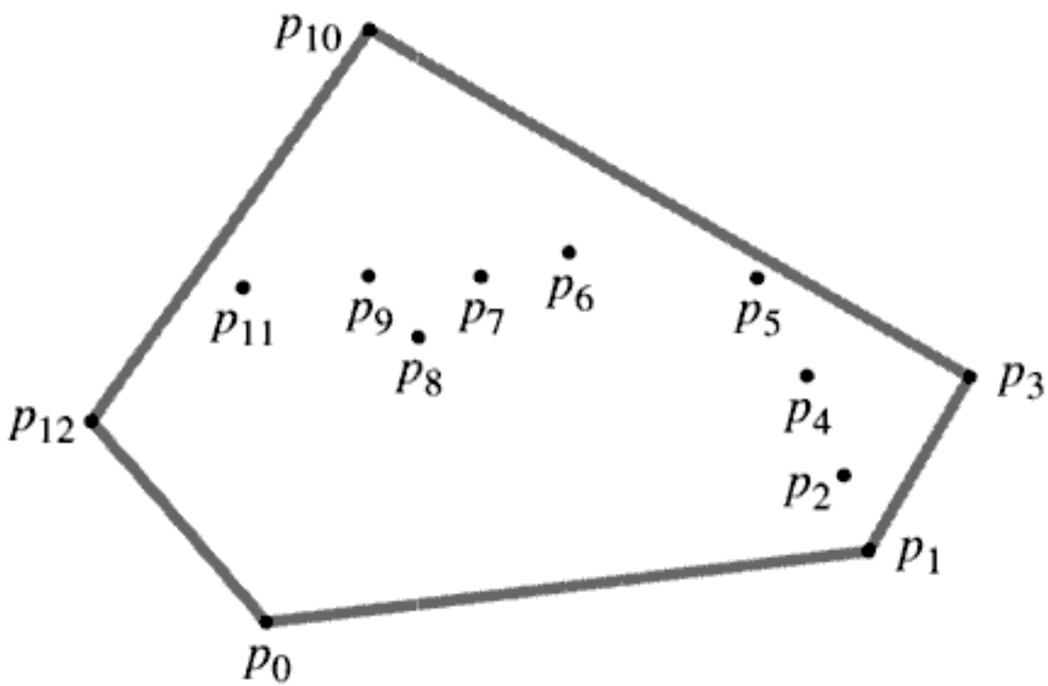
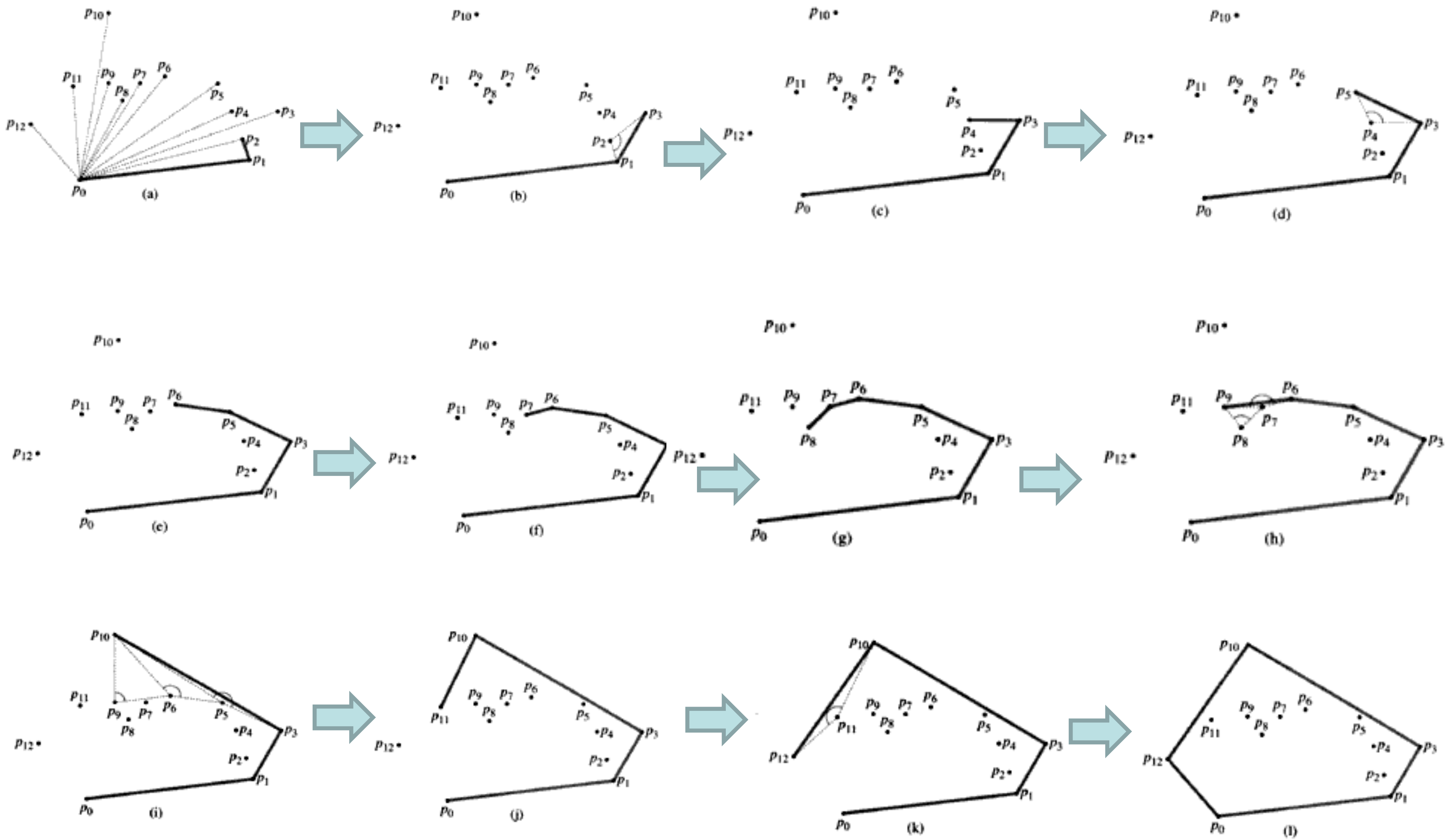


Figure 33.6 A set of points $Q = \{p_0, p_1, \ldots, p_{12}\}$ with its convex hull $\text{CH}(Q)$ in gray.



The execution of GRAHAM-SCAN on the set Q The current convex hull contained in stack S is shown in gray at each step. **(a)** The sequence $\langle p_1, p_2, \ldots, p_{12} \rangle$ of points numbered in order of increasing polar angle relative to p_0, and the initial stack S containing p_0, p_1, and p_2. **(b)–(k)** Stack S after each iteration of the **for** loop of lines 6–9. Dashed lines show nonleft turns, which cause points to be popped from the stack. In part (h), for example, the right turn at angle $\angle p_7 p_8 p_9$ causes p_8 to be popped, and then the right turn at angle $\angle p_6 p_7 p_9$ causes p_7 to be popped. **(l)** The convex hull returned by the procedure.

Convex Hull by Divide-and-Conquer

- First, sort all points by their x coordinate.
  - ($O(n \log n)$ time)
- Then divide and conquer:
  - Find the convex hull of the left half of points.
  - Find the convex hull of the right half of points.
  - Merge the two hulls into one.

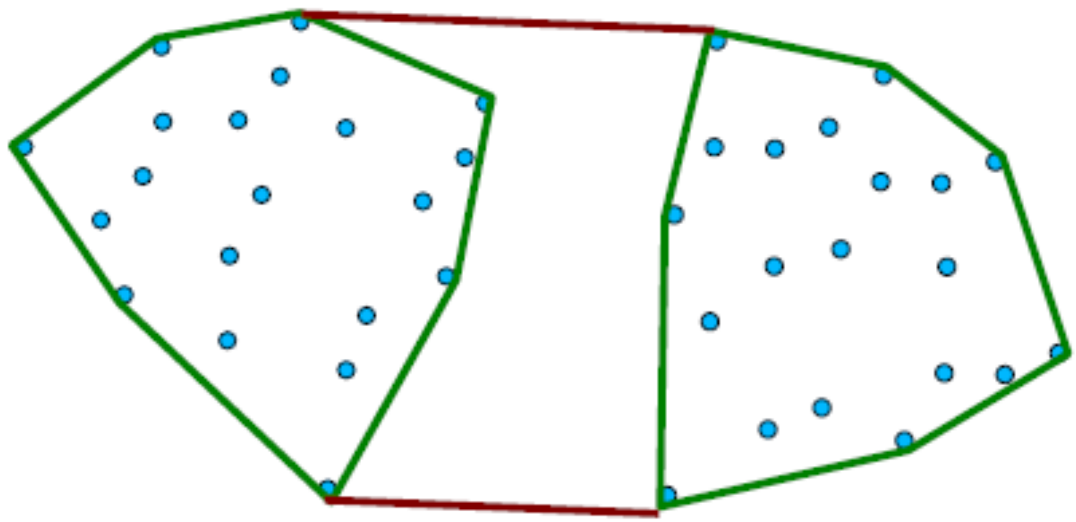
Convex Hull Pseudocode

```
//input: the number of points n, and
//an array of points S, sorted by x coord.
//output: the convex hull of the points in S.

point[] findHullDC(int n, point S[]) {
    if (n > 5) {
        int h = floor(n/2);
        m = n-h;
        point LH[], RH[]; //left and right hulls
        LH = findHullDC(h, S[1..h]);
        RH = findHullDC(m, S[h+1..n]);
        return mergeHulls(LH.size(), RH.size(),
                          LH, HR);
    } else {
        return Hull of S by exhaustive search;
    }
}
```

Merging Hulls

- Big picture:
  - first find the lines that are upper tangent, and lower tangent to the two hulls (the two red lines)

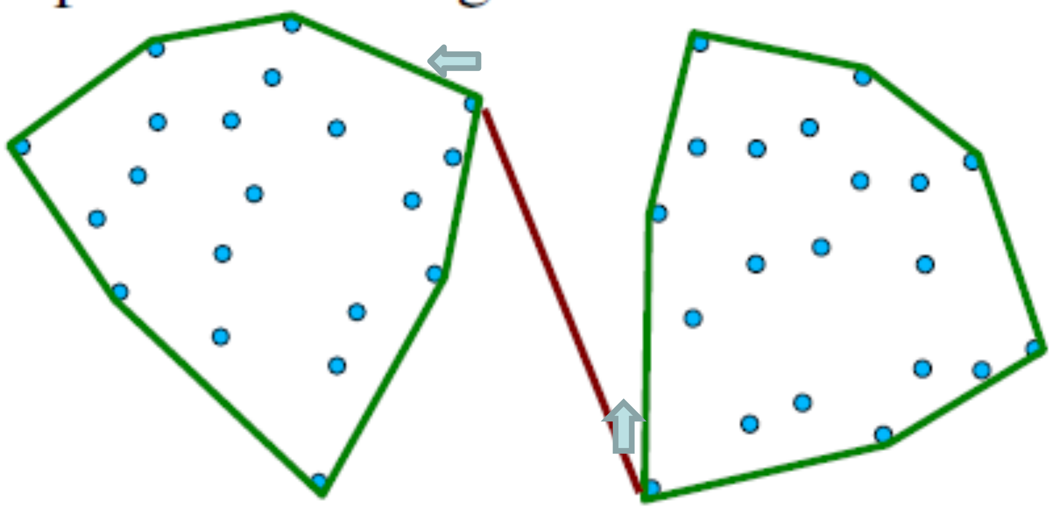


  - Then remove the points that are cut off.

Upper

Finding Tangent Lines

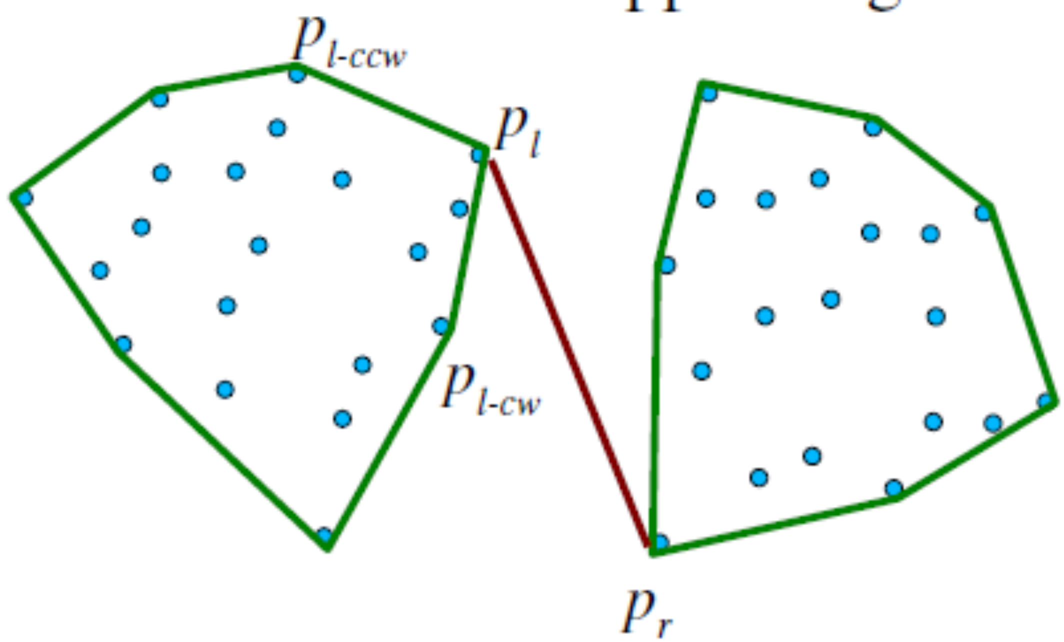
- Start with the rightmost point of the left hull, and the leftmost point of the right hull:



- While the line is not upper tangent to both left and right:
  - While the line is not upper tangent to the left, move to the next point (counter-clockwise).
  - While the line is not upper tangent to the right, move to the next point (clockwise).

Checking Tangentness

- How can we tell if a line is upper tangent to the left hull?

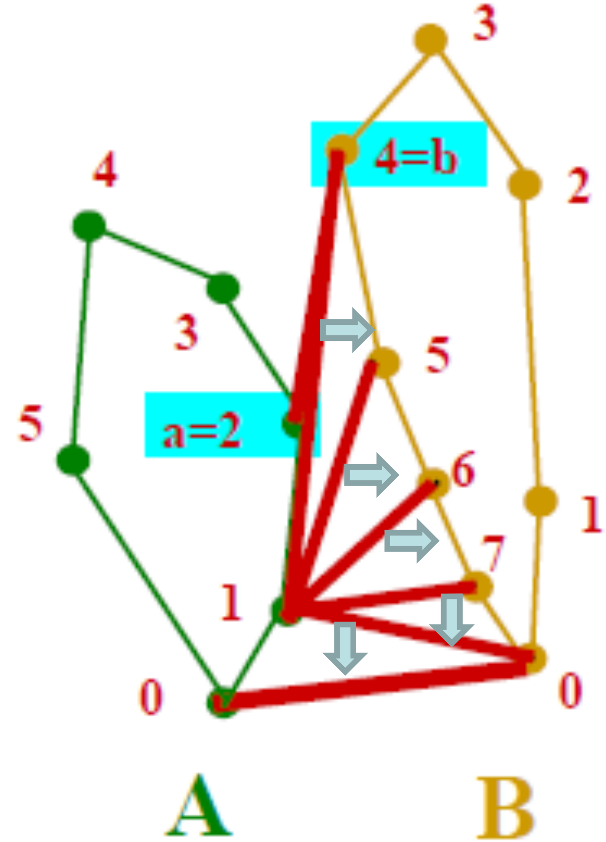


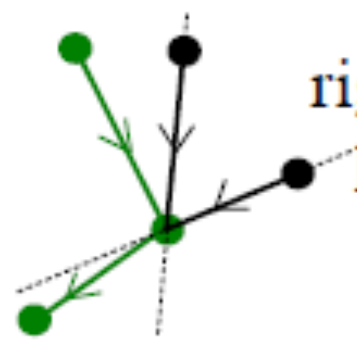
- The pair of line segments $\overline{p_r p_l}$, and $\overline{p_l p_{l\text{-}ccw}}$ should make a CCW turn at p_l.
- The same goes for $\overline{p_r p_l}$ and $\overline{p_l p_{l\text{-}ccw}}$.

Finding the lower tangent in O(*n*) time

```
a = rightmost point of A
b = leftmost point of B
while T=ab not lower tangent to both
      convex hulls of A and B do{
    while T not lower tangent to
     convex hull of A do{
       a=a-1
    }
    while T not lower tangent to
      convex hull of B do{
       b=b+1
     }
}
```

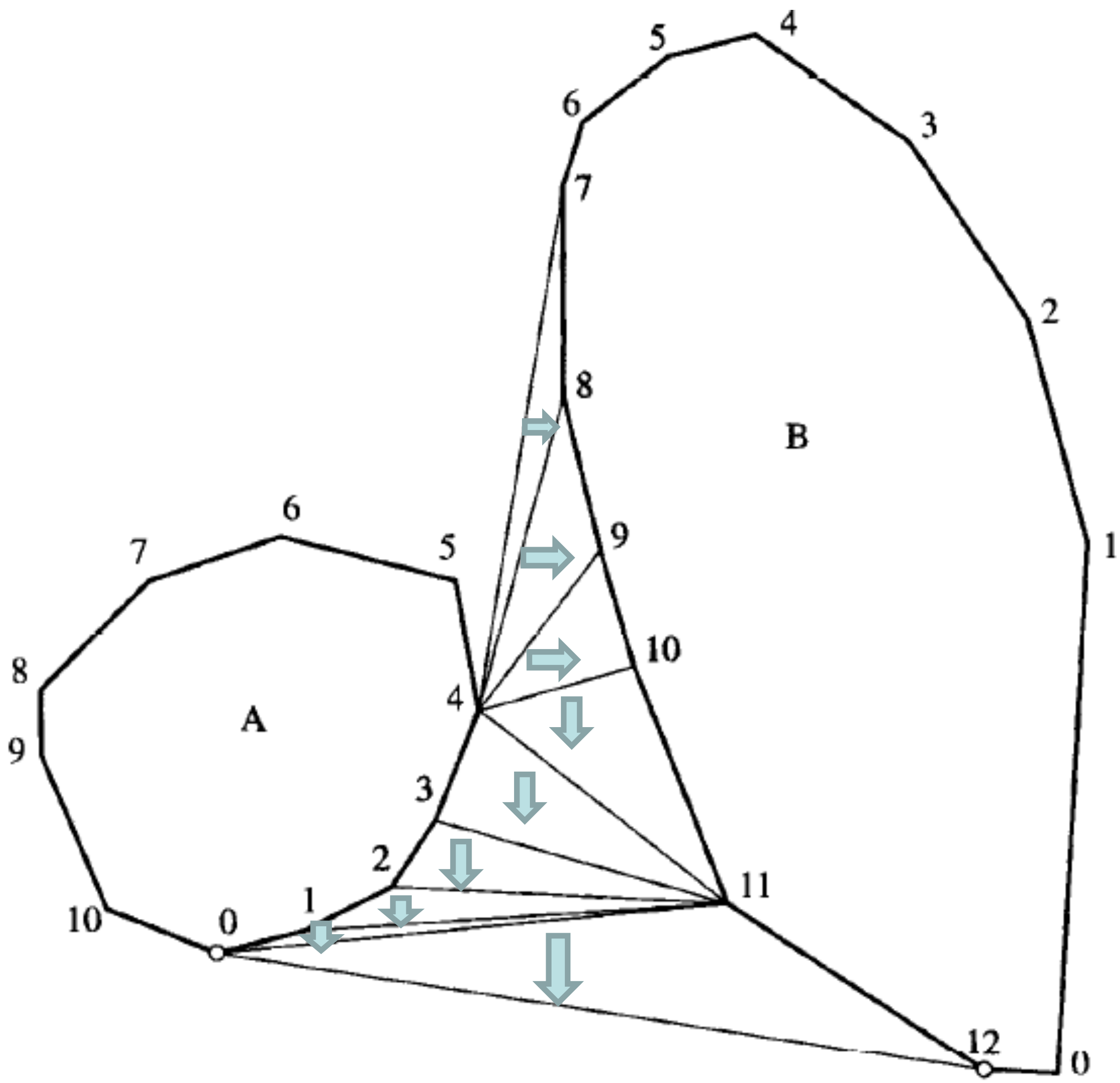
can be checked in constant time





right turn or left turn?

Lower Tangent Example



- •Initially, T=(4, 7) is only a lower tangent for A. The A loop does not execute, but the B loop increments b to 11.
- •But now T=(4, 11) is no longer a lower tangent for A, so the A loop decrements a to 0.
- •T=(0, 11) is not a lower tangent for B, so b is incremented to 12.
- • T=(0, 12) is a lower tangent for both A and B, and T is returned.

Convex Hull: Runtime

• Preprocessing: sort the points by x-coordinate	$O(n \log n)$ just once
• Divide the set of points into two sets **A** and **B**:	$O(1)$
• **A** contains the left $\lfloor n/2 \rfloor$ points,	
• **B** contains the right $\lceil n/2 \rceil$ points	
• Recursively compute the convex hull of **A**	$T(n/2)$
• Recursively compute the convex hull of **B**	$T(n/2)$
• Merge the two convex hulls	$O(n)$

$$T(n) = 2\,T(n/2) + cn$$

$$T(n) = O(n \log n)$$