Computational geometry

Inge Li Gørtz

CLRS Chapter 33.0, 33.1, 33.3.

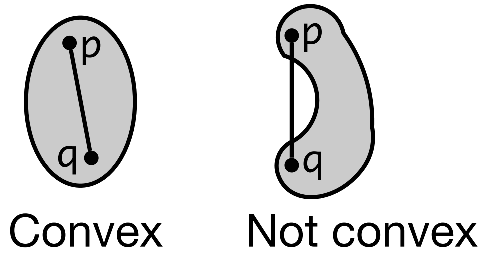
Computational Geometry

- Geometric problems (this course Euclidean plane).
- Does a set of line segments intersect, dividing plane into regions, find closest point, motion planning (robotics), overlaying of maps, lightening of scenes/computing shadows (computer graphics).
- This course:
    - Convex hull

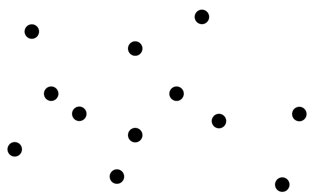
Convex Hull

Convex Hull

- Convex hull. Given set of points Q, the convex hull CH(Q), is the smallest polygon containing Q.
  - Polygon. Region of plane bounded by a cycle of line segments (edges). Points where edges meet are called the vertices of the polygon.
  - Convex. For any two points p, q inside the polygon, the line segment $\overline{pq}$ is completely inside the polygon.

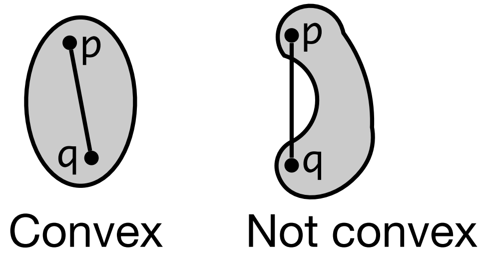


  - Smallest. Any convex proper subset of the convex hull excludes at least one point in Q.
  - Example.

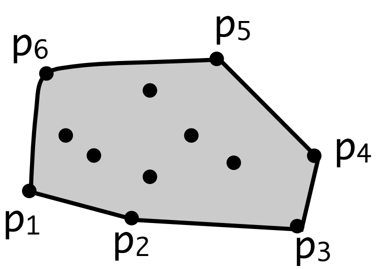


Convex Hull

- Convex hull. Given set of points Q, the convex hull CH(Q), is the smallest convex polygon containing Q.
  - Polygon. Region of plane bounded by a cycle of line segments (edges). Points where edges meet are called the vertices of the polygon.
  - Convex. For any two points p, q inside the polygon, the line segment $\overline{pq}$ is completely inside the polygon.



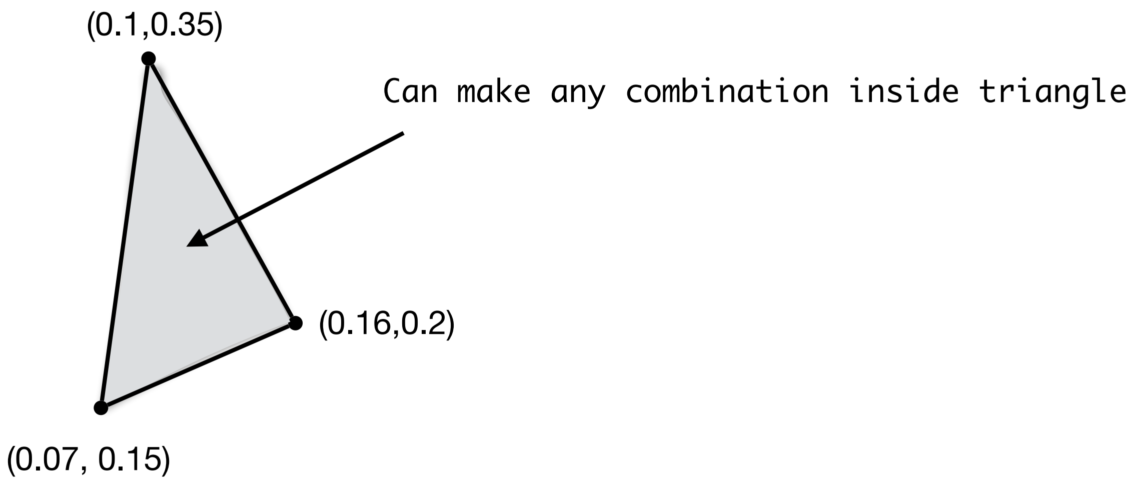
  - Smallest. Any convex proper subset of the convex hull excludes at least one point in Q.
  - Example.



  - Output. Vertices of convex hull in counterclockwise order: $\langle p_1, p_2, p_3, p_4, p_5, p_6 \rangle$.

Application of Convex Hull

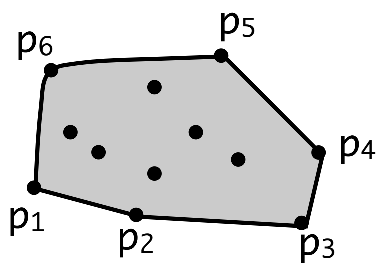
- Output from oil wells: mixture of several different components and proportions may vary between different sources. Can be mixed to obtain specific mixture. Say only interested in 2 of the components A and B. Want 12% A and 30% B. If we have 3 mixtures:
- M1 (10% A, 35%B) and M2 (16% A, 20% B) and M3 (7% A, 15% B).
- Mix M1 and M2 in ratio: 2:1.
- Cannot get 13% A and 22% B from M1 and M2.
- Mix M1, M2 and M3 in ratio 1:3:1.
- Represent mixtures by point in plane: p1=(0.1,0.35), p2=(0.16,0.2), p3 = (0.07, 0.15):



- n base mixtures: can make any combination in convex hull.

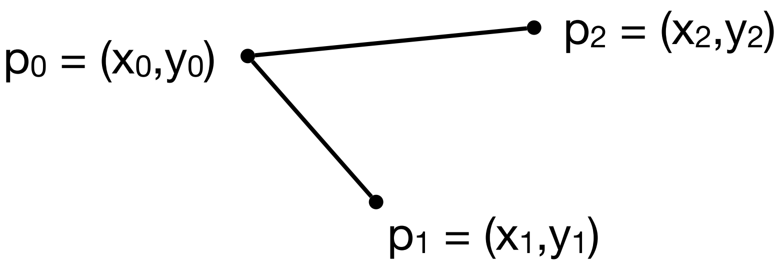
Convex Hull

- 3 equivalent definitions of convex hull: Given set of points Q, the convex hull CH(Q) is
  - Def 1. The *smallest* convex polygon *containing* Q.
  - Def 2. The *largest* convex polygon, whose *vertices all are points in* Q.
  - Def 3. The convex polygon *containing Q and whose vertices all are points in* Q.
- Assumption (we will get rid of this later). No three points lie on a common line.



Convex hull: Easy cases

- $|Q| = 1$. Return Q.
- $|Q| = 2$. Return Q.
- $|Q| = 3$. All 3 points are in CH(Q). Check if in counterclockwise order.
  - Assume p_0 is furthest to the left.



  - Consider line segments $\overline{p_0p_1}$ and $\overline{p_0p_2}$.

Counterclockwise $\Leftrightarrow$ slope of $\overline{p_0p_1}$ is less than slope of $\overline{p_0p_2}$.

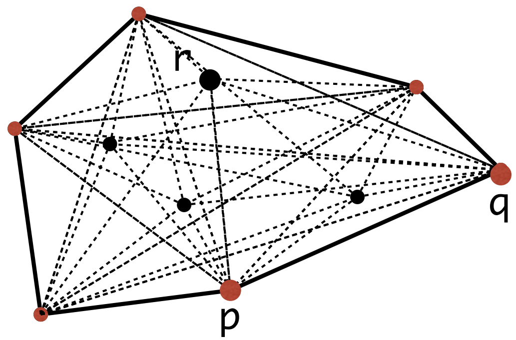
$\Leftrightarrow$ $(y_1-y_0)/(x_1-x_0) < (y_2-y_0)/(x_2-x_0)$

$\Leftrightarrow$ $(y_1-y_0)(x_2-x_0) < (y_2-y_0)(x_1-x_0)$.

Counterclockwise $\Leftrightarrow$ $(y_1-y_0)(x_2-x_0) < (y_2-y_0)(x_1-x_0)$

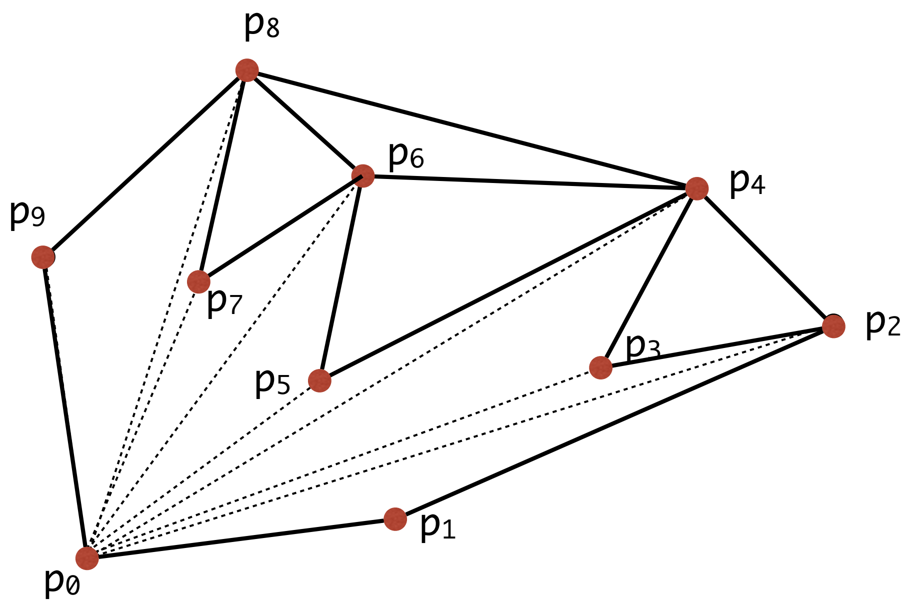
Jarvis's march

- Start with adding lowest point p_0 to CH(Q).
- Next point after p:
  - point appearing to be furthest to the right to someone standing at p and looking at the other points (smallest if sorted in counterclockwise order).
  - If q is the point following p then for any other point r in Q p,q,r are in counterclockwise order.
  - Can find next vertex by performing n-1 counterclockwise tests.
  - Time:
    - Θ(1) for each counterclockwise test.
    - n tests for each vertex in the convex hull
    - Θ(nh)
    - *Output sensitive*



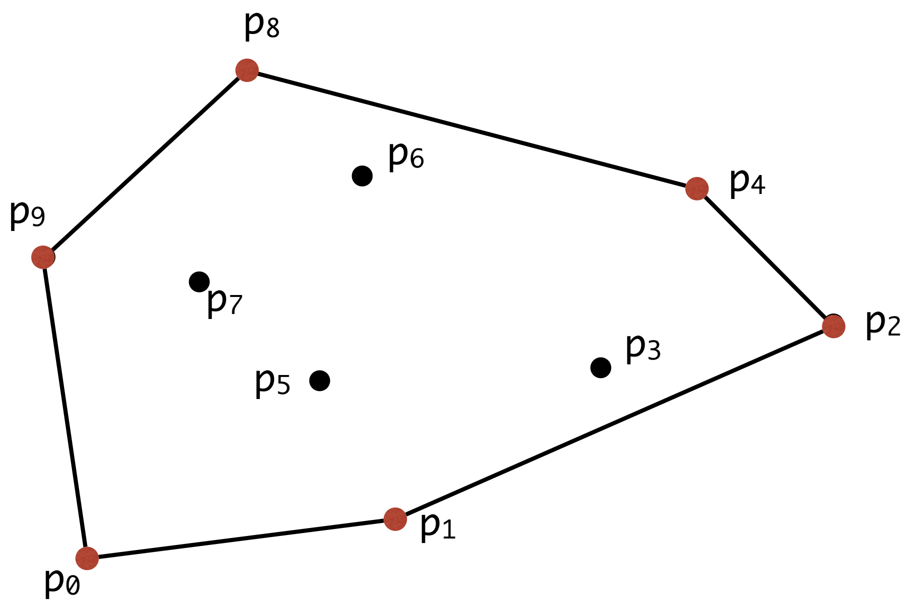
Graham's scan

- Graham's scan.
  - Pick lowest point p_0 as starting point
  - Sort remaining points in counterclockwise order around p_0.
  - Use linear time scan to build hull:
    - Push p_0, p_1 and p_2 onto the stack.
    - Next point p:
      - If adding p gives a left turn push p onto stack
      - If adding p gives a right turn pop top element from stack and check again. Continue checking until we get a left turn or only 2 vertices left on stack.



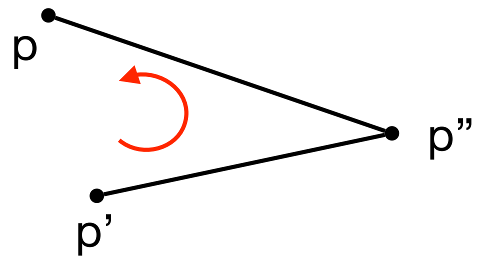
Graham's scan

- Graham's scan
  - Pick lowest point p_0 as starting point
  - Sort remaining points in counterclockwise order around p_0.
  - Use linear time scan to build hull.
    - Push p_0, p_1 and p_2 onto the stack.
    - Next point p: Let p' and p" be the two top elements of the stack
      - If adding p gives a left turn push p onto stack
      - If adding p gives a right turn pop top element from stack and check again. Continue checking until we get a left turn or only 2 vertices left on stack.

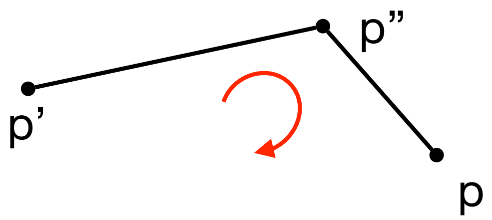


Left turns and right turns

- Next point p: Let p' and p" be the two top elements of the stack
  - Check if adding p gives a left turn or a right turn.
  - If adding p gives a left turn then p', p", p are in counterclockwise order:

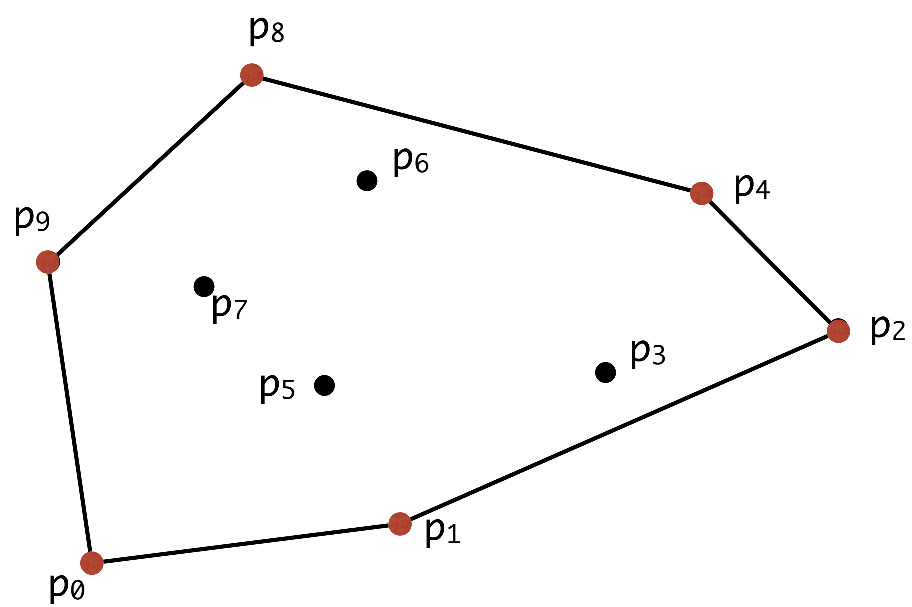


  - If adding p gives a right turn then p', p", p are in clockwise order:



Graham's scan

- Graham's scan
  - Pick lowest point p_0 as starting point
  - Sort remaining points in counterclockwise order around p_0.
  - Use linear time scan to build hull.
    - Push p_0, p_1 and p_2 onto the stack.
    - Next point p: Let p' and p" be the two top elements of the stack
      - If p', p", p are in counterclockwise order: push p onto stack
      - If p', p", p are in clockwise order: pop top element from stack and check again. Continue until we get a left turn or only 2 vertices left on stack.



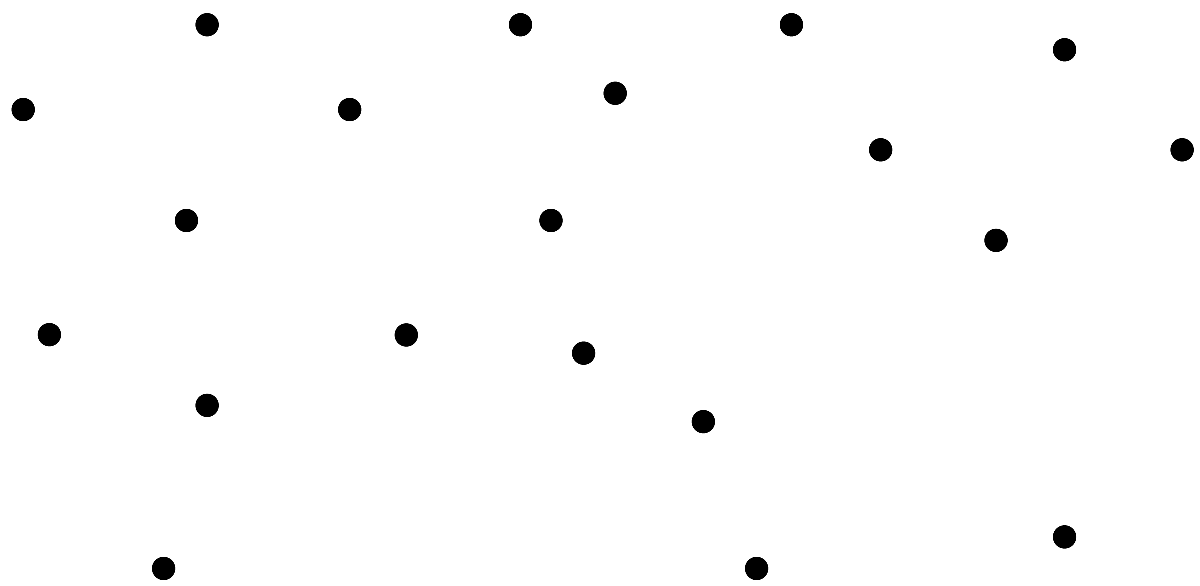
Graham’s scan

- Graham’s scan
  - Pick lowest point p_0 as starting point
  - Sort remaining points in counterclockwise order around p_0.
  - Use linear time scan to build hull.
    - Push p_0, p_1 and p_2 onto the stack.
    - Next point p: Let p’ and p” be the two top elements of the stack
      - If p’, p”, p are in counterclockwise order: push p onto stack
      - If p’, p”, p are in clockwise order: pop top element from stack and check again. Continue until we get a left turn or only 3 vertices left on stack.

- Analysis.
  - Sorting $\Theta(n \log n)$
  - Counterclockwise check: $\Theta(1)$
  - Each check is due to a push or a pop.
  - Each point pushed once and popped at most once.
  - n pops, O(n) pushes, O(n) counterclockwise checks. All constant time each.
  - Time $\Theta(n)$ for scan.
  - Total time $\Theta(n \log n)$

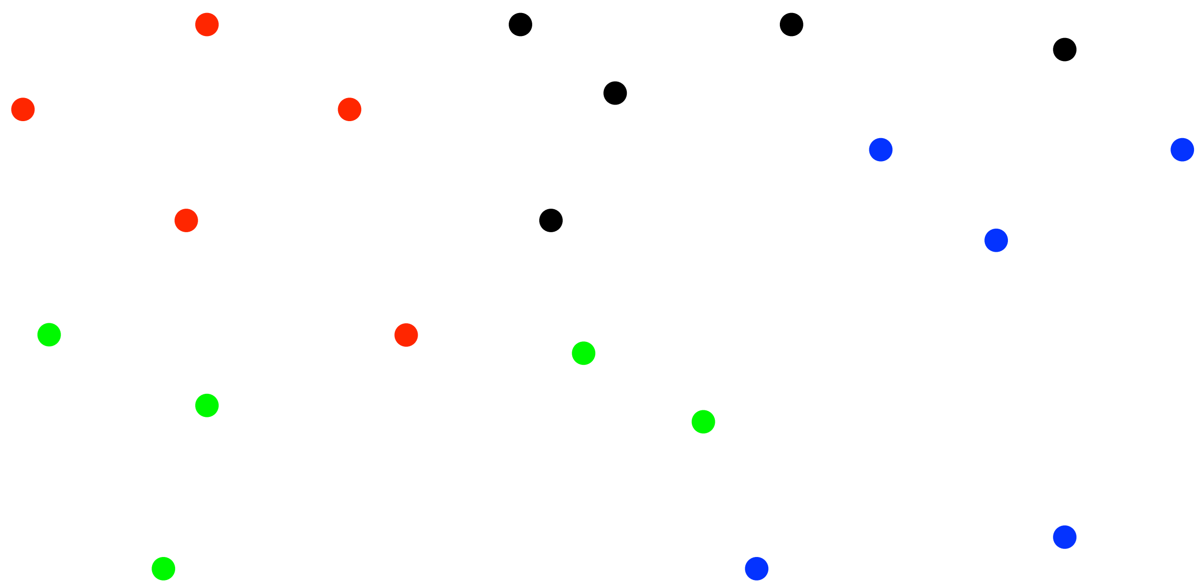
Chan’s algorithm (shattering)

- Guess h.
- Shatter the input into arbitrary n/h subsets.



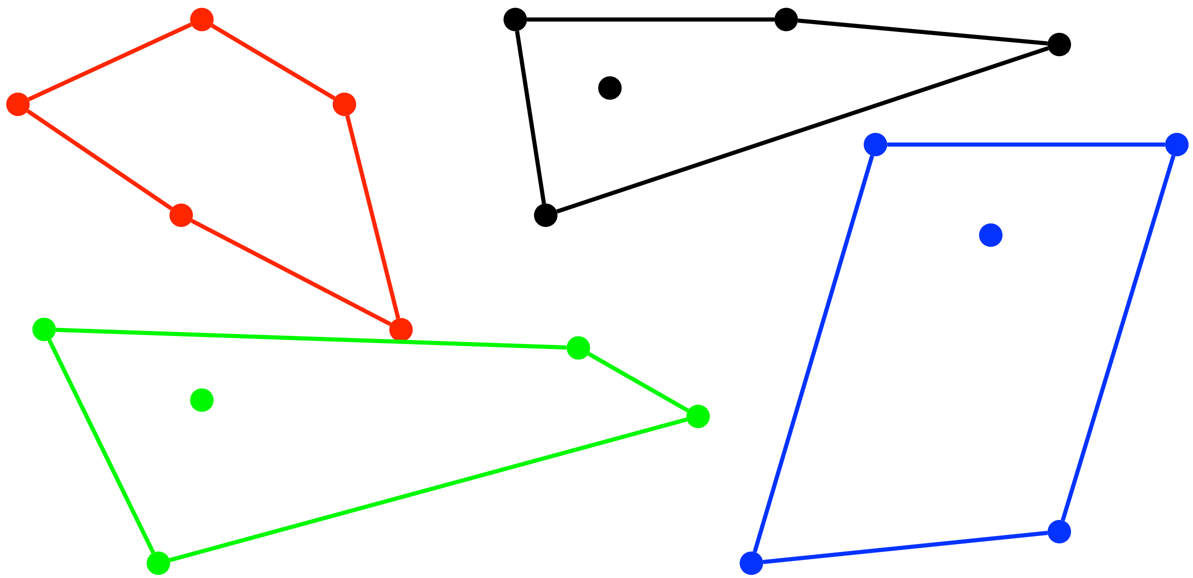
Chan's algorithm (shattering)

- Guess h.
- Shatter the input into arbitrary n/h subsets.



Chan's algorithm (shattering)

- Guess h.
- Shatter the input into arbitrary n/h subsets.
- Compute the convex hull of each subset using Graham's scan.
  - Time: O((n/h) h log h) = O(n log h).



Chan’s algorithm (shattering)

- Guess h.
- Shatter the input into arbitrary n/h subsets.
- Compute the convex hull of each subset using Graham’s scan.
    - Time: O((n/h) h log h) = O(n log h).
- Use idea from Jarvis’ march (wrapping) around the n/h subhulls.
    - Successor can be found in O(h log h) time.
    - Time for second part: O((n/h) h log h) = O(n log h).