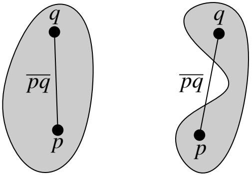
Convex Hull (chapter 1)

Elisha Sacks

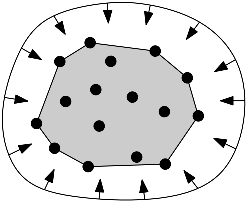
Convexity



convex not convex

- A point set S is convex if for all points p and q in S the line segment $\overline{pq}$ is in S.
- The convex hull of S is its smallest convex superset.
- Hence, the convex hull is the intersection of all the convex sets that contain S.

Finite Point Sets



- The convex hull of a finite point set S is the smallest polygon that contains every point in S.
- Intuition: shrink wrap the polygon.

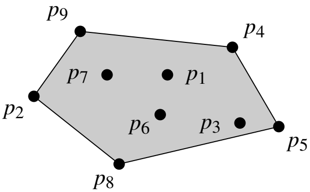
Problem Statement

input = set of points:

$p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9$

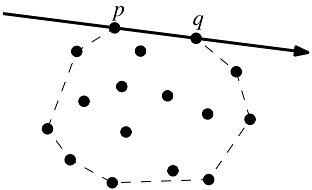
output = representation of the convex hull:

p_4, p_5, p_8, p_2, p_9



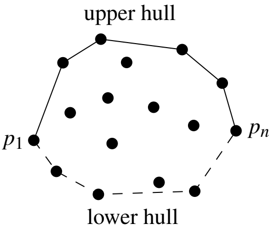
- The output points are in clockwise order.
- Clockwise order is more convenient for this example.
- Counterclockwise order is the norm.
- The rest of the course uses counterclockwise order.

Hull Edge



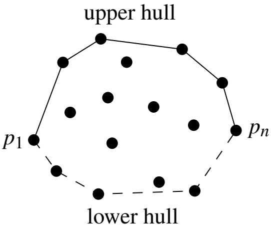
pq is a hull edge if every other point lies on the same side of its supporting line.

Upper Hull



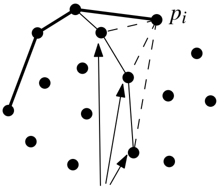
- The upper hull is the edges whose supporting lines are above all the other points.
- It consists of a polygonal curve from the leftmost point p_1 to the rightmost point p_n.
- The lower hull is the edges whose supporting lines are below all the other points.
- It consists of a polygonal curve from the rightmost point to the leftmost point.
- Algorithm: construct the two hulls then append them.

Hull Predicate



- Let a and b be points with $a_x < b_x$.
- ab is an upper hull edge if every point c is below ab.
- Equivalently $\mathrm{LT}(a, b, c) < 0$.
- ab is a lower hull edge if every point c is above ab.
- Equivalently $\mathrm{LT}(a, b, c) > 0$.

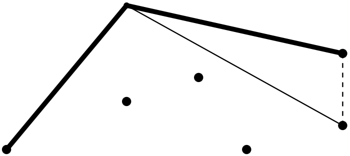
Generic Upper Hull Algorithm

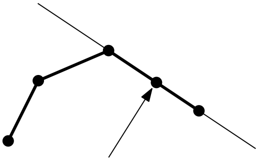


Points have distinct x coordinates and no three are collinear.

1. Sort the points in increasing x order: $p_1, \ldots, p_n$.
2. Initialize an empty hull $h = ()$.
3. For $i = 1$ to n
   3.1 Append p_i to h.
   3.2 While h contains $m \geq 3$ points and $\mathrm{LT}(h_{m-2}, h_{m-1}, h_m) > 0$
      3.2.1 Set h_{m-1} to h_m.
      3.2.2 Remove the last element of h.

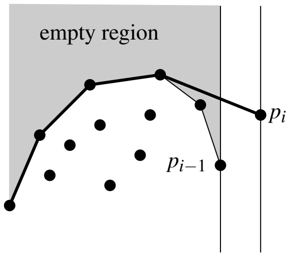
Handling Degenerate Cases





- Degeneracy 1: points with equal x coordinates.
- Handling: break ties by y order (lexicographic order).
- The higher point is on the upper hull.
- Degeneracy 2: collinear points.
- Handling: treat as left turn (replace $>$ with $\geq$).
- The interior points are not on the hull.
- What happens when both degeneracies occur?

Correctness



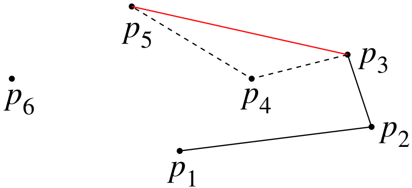
Inductive correctness proof for the upper hull algorithm.

- Correctness is trivial for $i = 2$ points, so consider $i > 2$.
- The update creates a curve h_i from p_1 to p_i with right turns.
- Let p_j with $j < i$ be a point that is not an h_i vertex.
- p_j is in the x range of h_{i-1} because p_{i-1} is an h_{i-1} vertex and the points are in x order.
- p_j is below or on h_{i-1} by inductive hypothesis.
- p_j is below or on h_i because removing left turns increases y.

Complexity

- Sorting the points takes $O(n \log n)$ time.
- Each point is removed at most once from h.
- Hence, the time spent on updating the hull is $O(n)$.
- Thus, the running time is $O(n \log n)$.
- The space complexity is $O(n)$.
- These bounds are optimal.

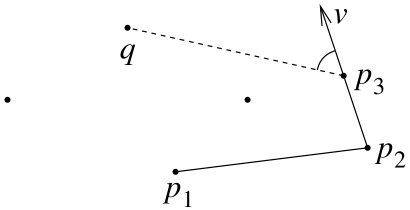
Improved Version



Construct the entire hull with one subroutine.

1. Set p_1 to the point with the smallest y coordinate.
2. Sort the other points counterclockwise around p_1.
3. Construct the hull as before, but keeping left turns.

Gift Wrapping Algorithm



1. Initialize the hull to $h = (p_1)$ with p_1 the point with the smallest y coordinate.
2. Initialize v to $(1, 0)$.
3. Repeat
   3.1 Let $h = (p_1, \ldots, p_i)$.
   3.2 Find the point q that minimizes the angle $\angle(v, q - p_i)$.
   3.3 Append q to h.
   3.4 Set v to $q - p_i$.
   3.5 If $p_1 = q$ return h.

Analysis

- All n points can be on the hull.
- Adding a point to the hull takes $O(n)$ time.
- Hence, the running time is $O(n^2)$.
- The running time is also $O(nh)$ with h the size of the hull.
- Gift wrapping is faster than the $O(n \log n)$ algorithms when most of the points are in the interior of the hull.
- An algorithm whose running time depends on the output size is called output sensitive.