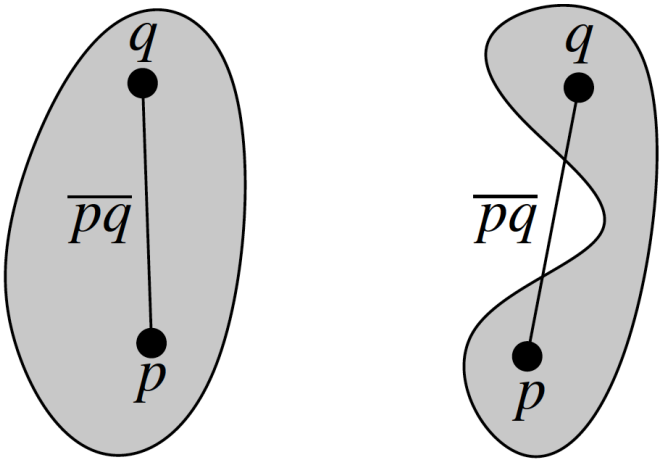
Convex Hulls

Michael T. Goodrich

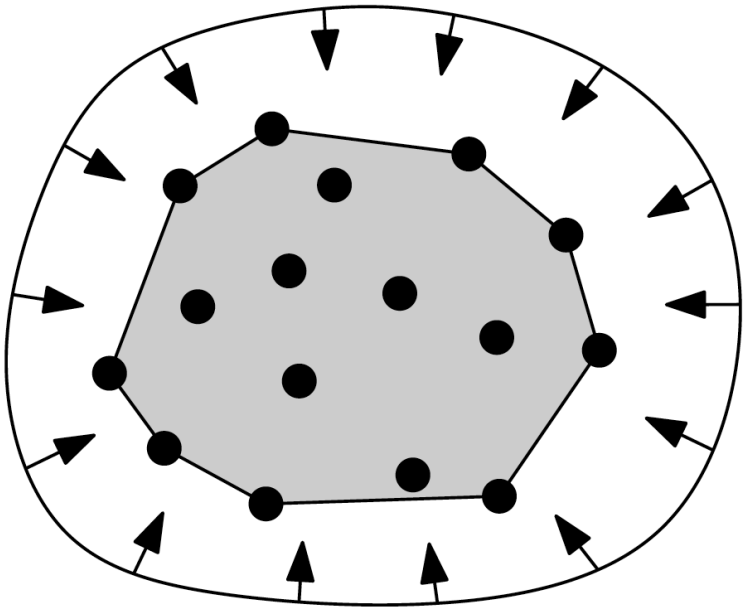
Review: Convexity



convex not convex

Convex hull

- Smallest convex set containing all *n* points



Convex hull

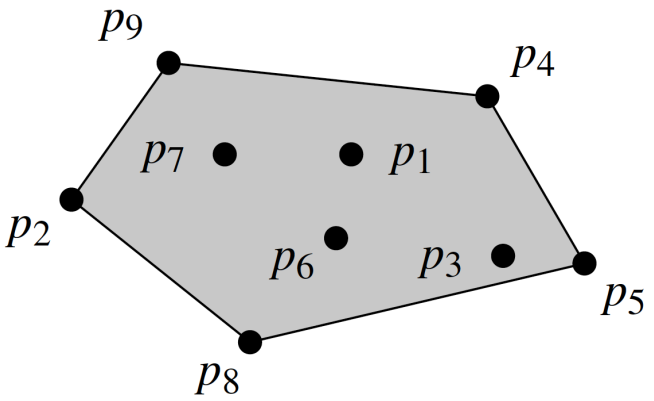
- Smallest convex set containing all *n* points

input = set of points:

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

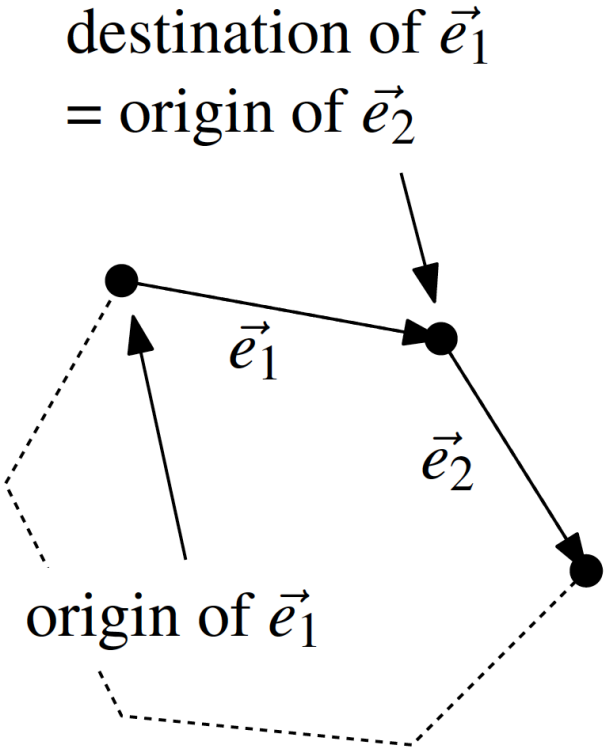
output = representation of the convex hull:

p_4, p_5, p_8, p_2, p_9

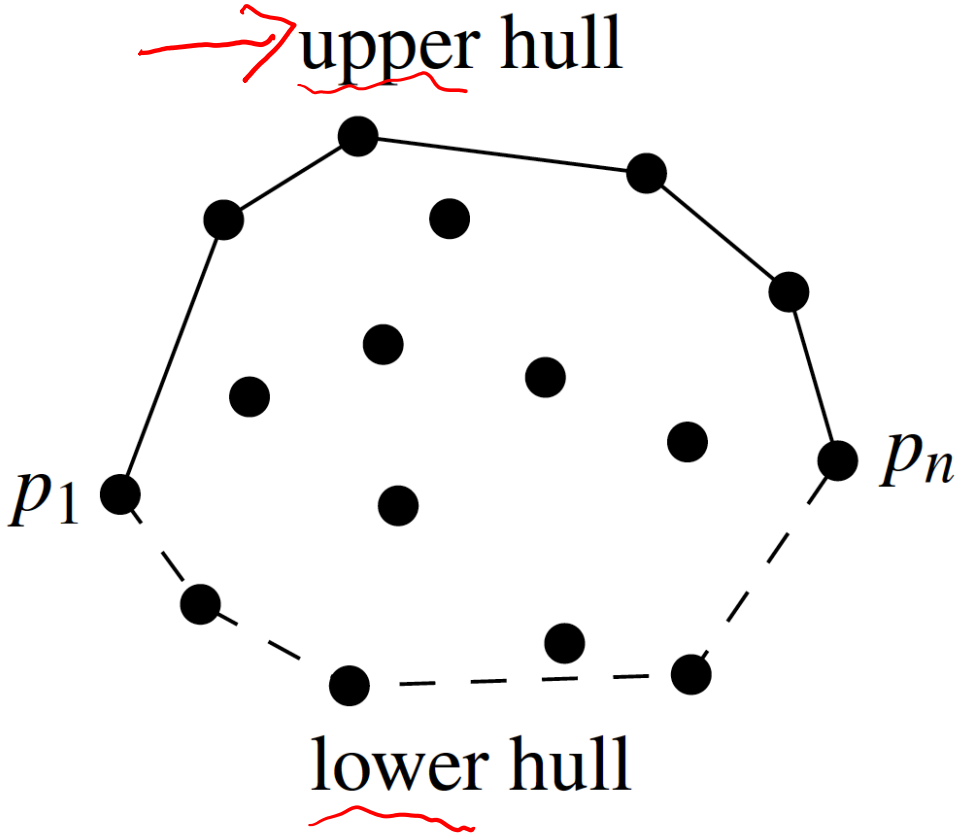


Orientation Test

- right turn or left turn (or straight line)

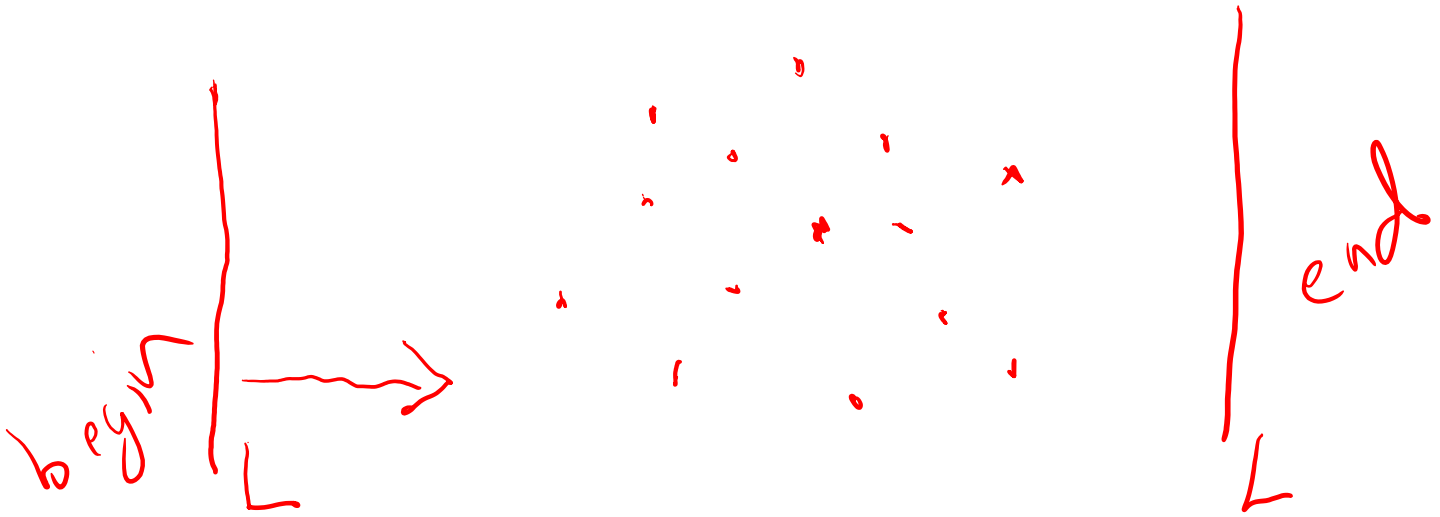


A Better Convex Hull Algorithm



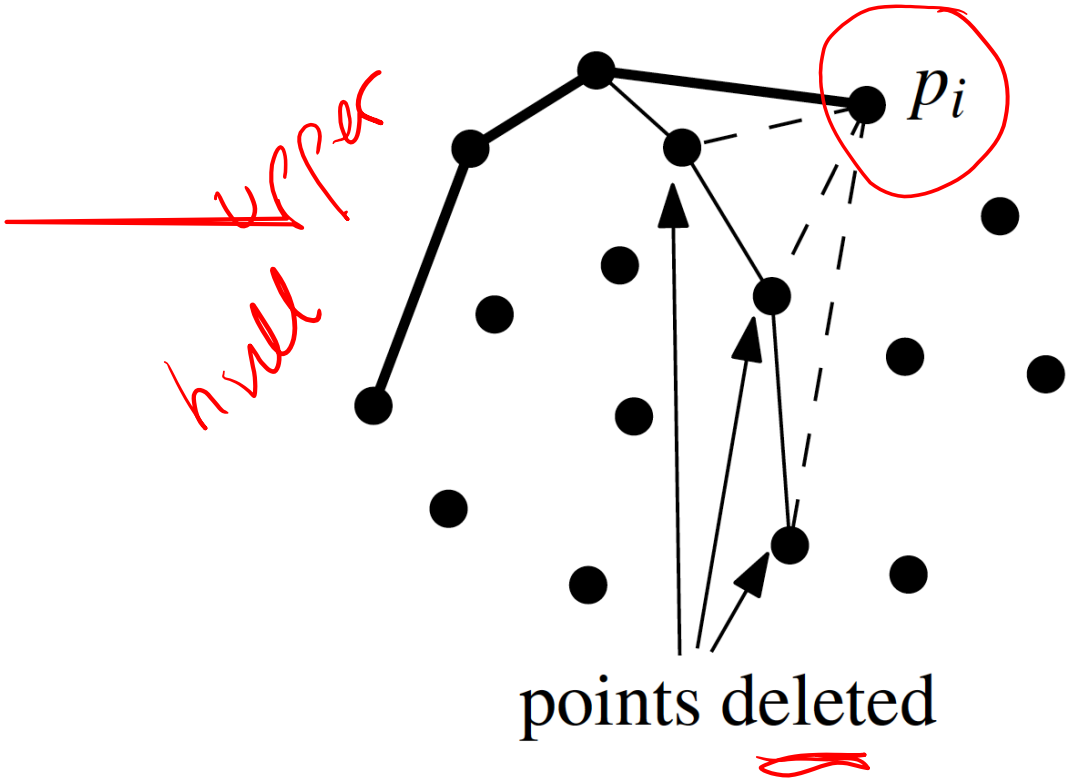
Plane-Sweep Technique

- We “sweep” the plane with a vertical line
- Stop at **event points**
- Maintain a partial solution for the swept-over area



Events

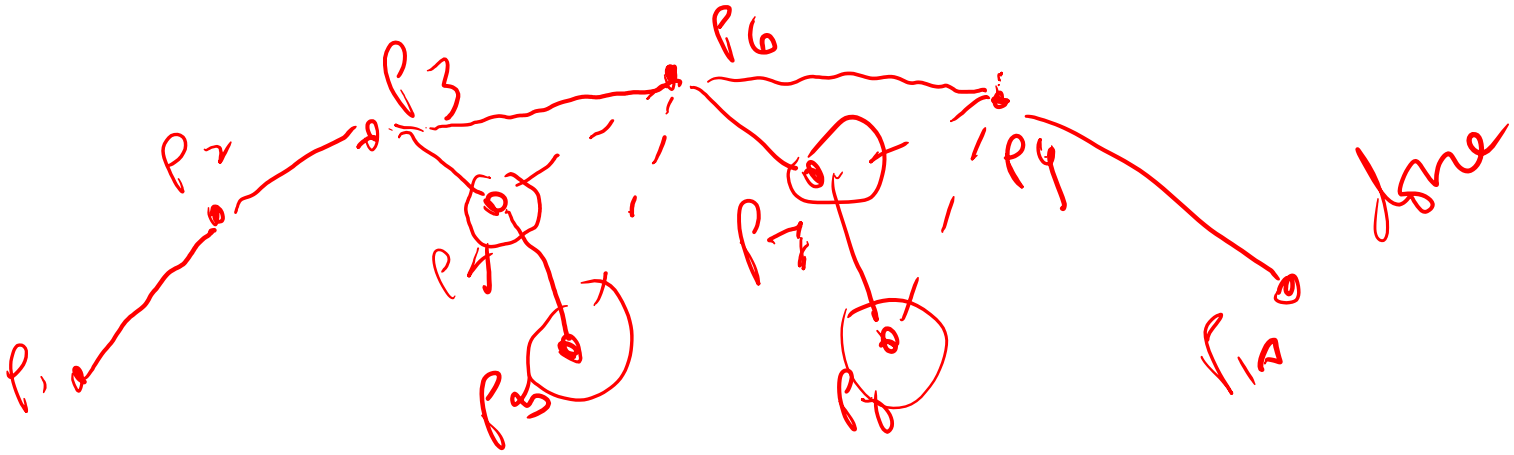
- Each point determines an event



Upper Hull Algorithm

Algorithm CONVEXHULL(P)
Input. A set P of points in the plane.
Output. A list containing the vertices of $\mathcal{CH}(P)$ in clockwise order.

1. Sort the points by x-coordinate, resulting in a sequence $p_1, \ldots, p_n$.
2. Put the points p_1 and p_2 in a list $\mathcal{L}_{\text{upper}}$, with p_1 as the first point.
3. **for** $i \leftarrow 3$ **to** n
4. **do** Append p_i to $\mathcal{L}_{\text{upper}}$.
5. **while** $\mathcal{L}_{\text{upper}}$ contains more than two points **and** the last three points in $\mathcal{L}_{\text{upper}}$ do not make a right turn
6. **do** Delete the middle of the last three points from $\mathcal{L}_{\text{upper}}$.
7. Put the points p_n and p_{n-1} in a list $\mathcal{L}_{\text{lower}}$, with p_n as the first point.



Upper Hull Algorithm

Algorithm CONVEXHULL(P)
Input. A set P of points in the plane.
Output. A list containing the vertices of $\mathcal{CH}(P)$ in clockwise order.

1. Sort the points by x-coordinate, resulting in a sequence $p_1, \ldots, p_n$.
2. Put the points p_1 and p_2 in a list $\mathcal{L}_{\text{upper}}$, with p_1 as the first point.
3. **for** $i \leftarrow 3$ **to** n
4. **do** Append p_i to $\mathcal{L}_{\text{upper}}$.
5. **while** $\mathcal{L}_{\text{upper}}$ contains more than two points **and** the last three points in $\mathcal{L}_{\text{upper}}$ do not make a right turn
6. **do** Delete the middle of the last three points from $\mathcal{L}_{\text{upper}}$.
7. Put the points p_n and p_{n-1} in a list $\mathcal{L}_{\text{lower}}$, with p_n as the first point.

O(n log n)

O(n)

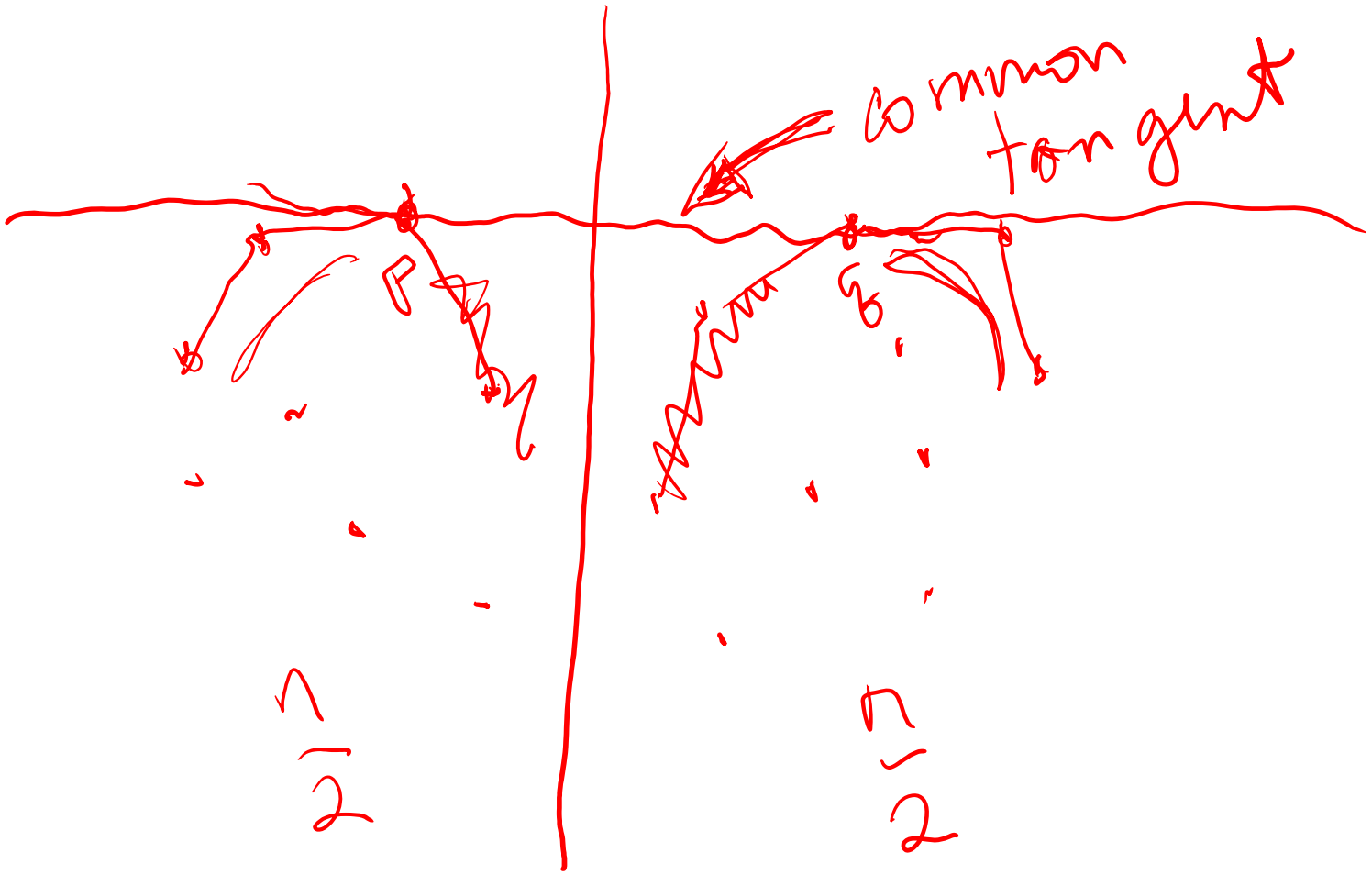
Each iteration — add a point
— delete a point
] O(n)

* O(n) time
+ sorting : O(n log n)

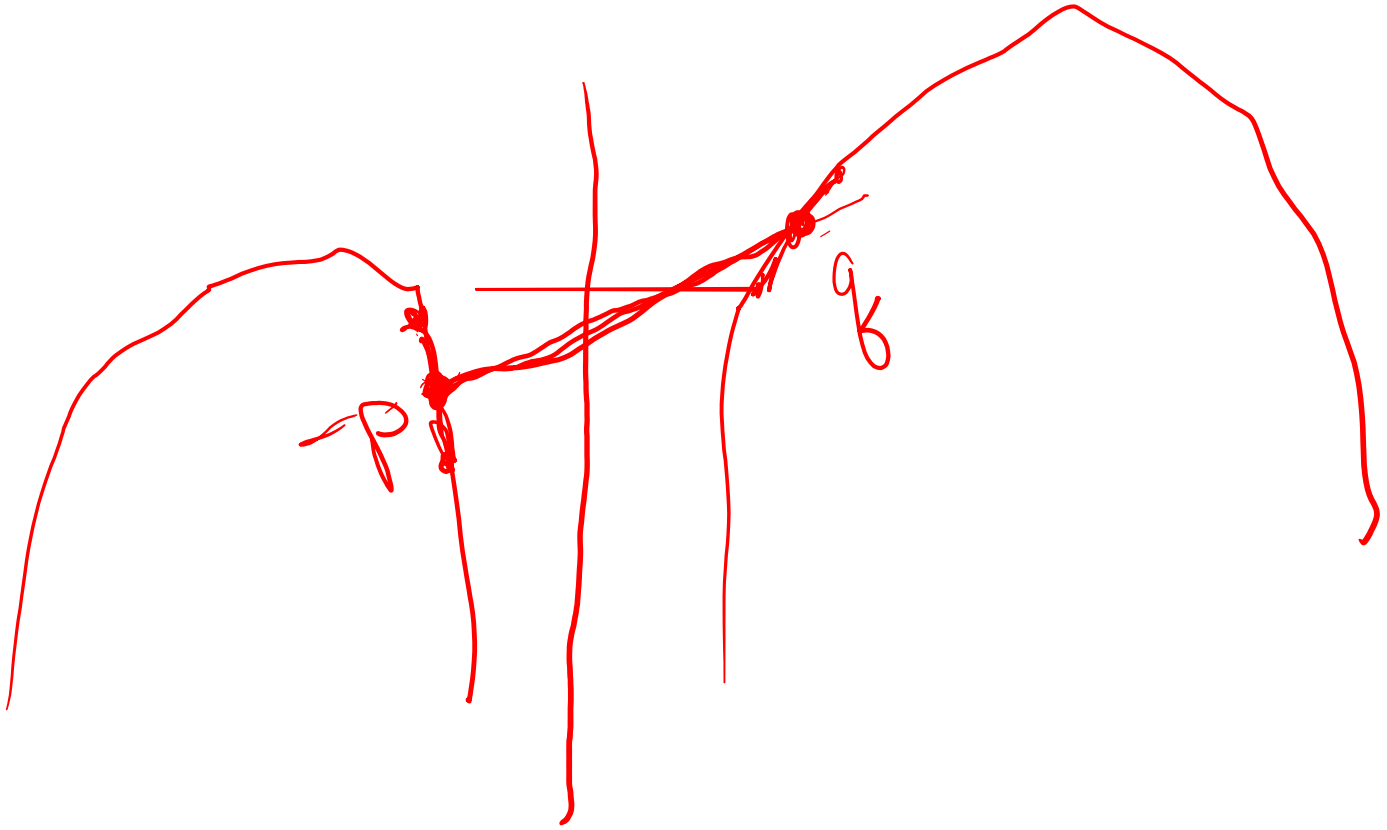
Divide-and-Conquer

- If the problem is smaller than a constant, solve it
- Else:
  - Divide the problem into two or more subproblems
  - Solve each subproblem recursively
  - Merge the two solutions together

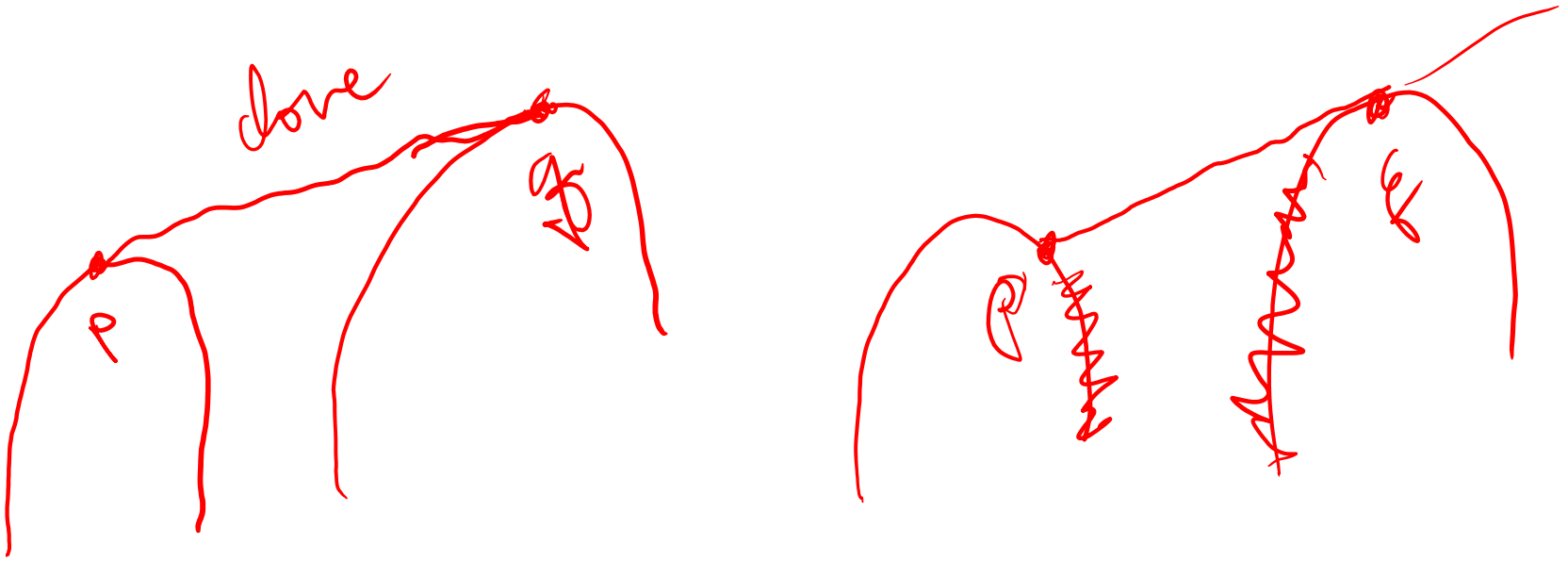
Divide-and-Conquer Convex Hull



Divide-and-Conquer Convex Hull

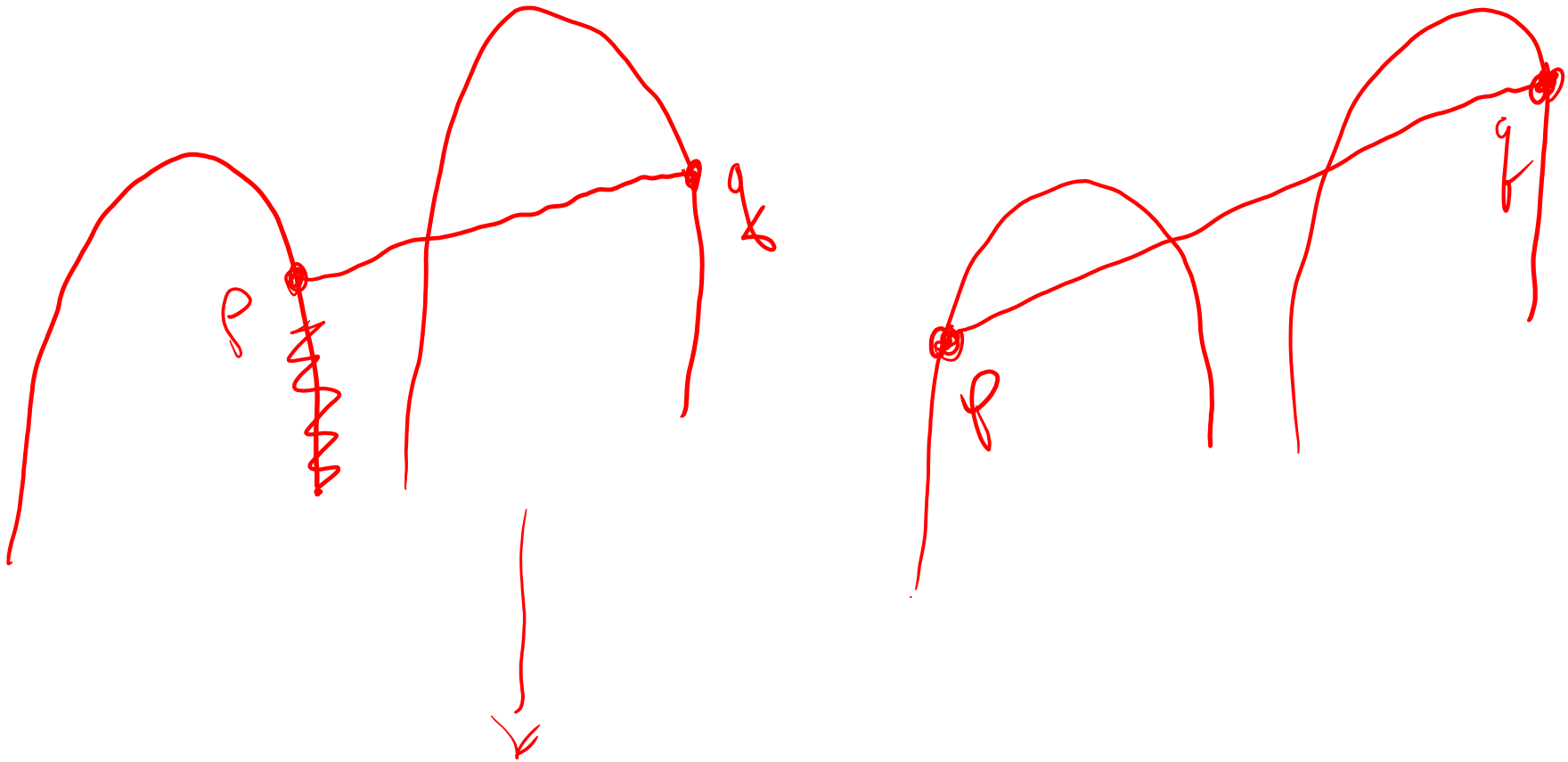


Divide-and-Conquer Convex Hull

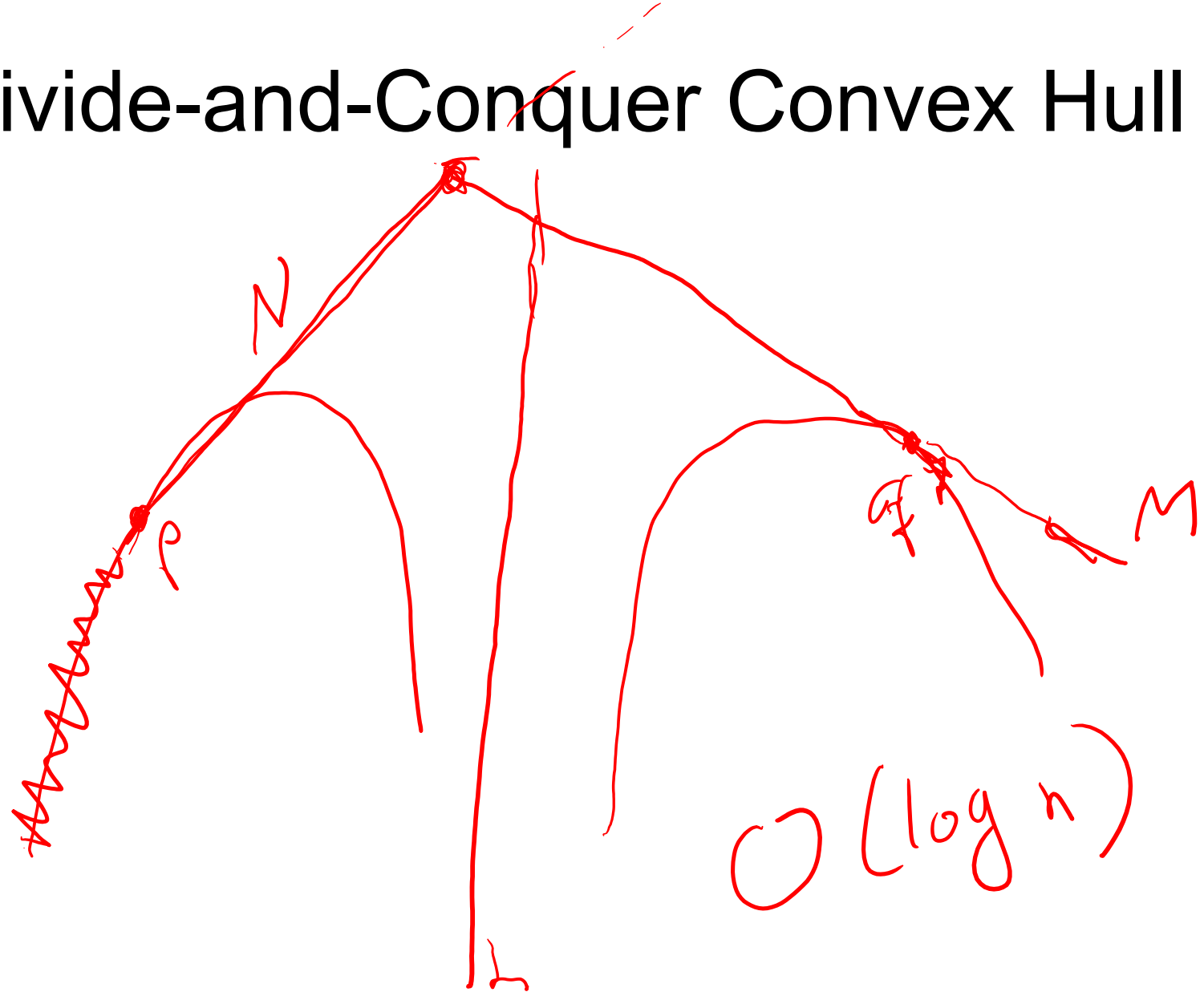


Divide-and-Conquer Convex Hull

Divide-and-Conquer Convex Hull



Divide-and-Conquer Convex Hull

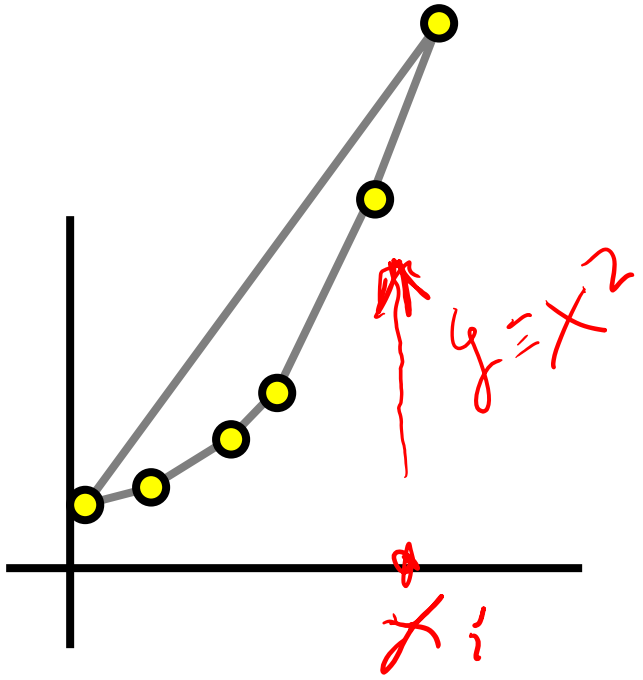


Divide-and-Conquer Convex Hull

Divide-and-Conquer Convex Hull

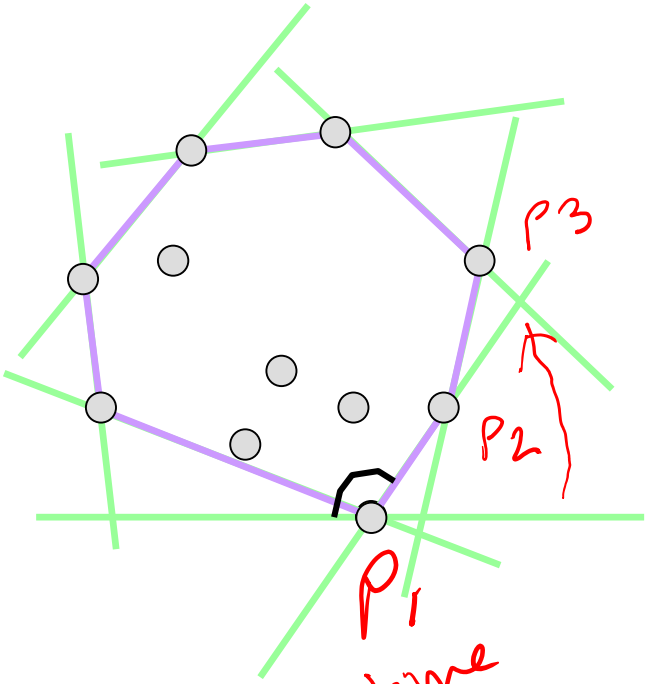
Lower Bound for Convex Hull

- A reduction from sorting to convex hull is:
  - Given n real values x_i, generate n 2D points on the graph of a convex function, e.g. (x_i, x_i^2).
  - Compute the (ordered) convex hull of the points.
  - The order of the convex hull points is the numerical order of the x_i.
- So CH time is $\Omega(n \log n)$



Convex Hull – Gift Wrapping

- Algorithm:
  - Find a point p_1 on the convex hull (e.g. the lowest point).
  - Rotate counterclockwise a line through p_1 until it touches one of the other points (start from a horizontal orientation).
    - Repeat the last step for the new point.
    - Stop when p_1 is reached again.

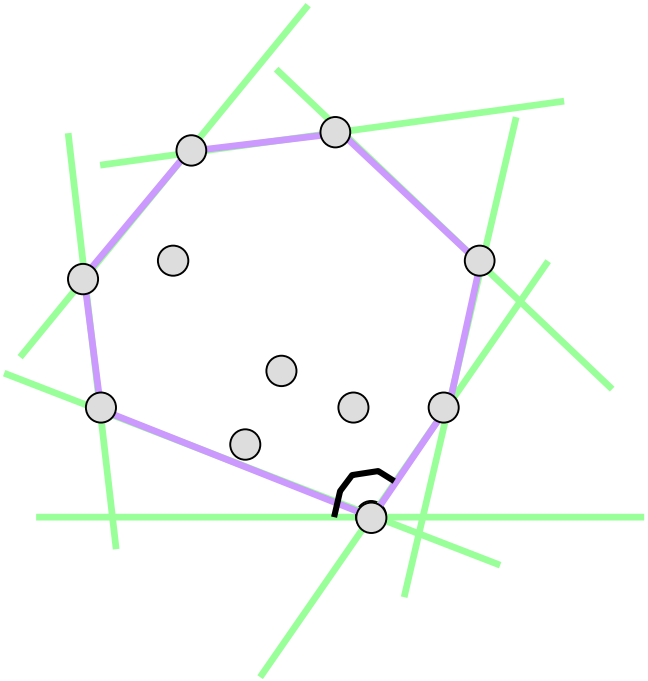


- Time Complexity: O(nh), where n is the input size and h is the output (hull) size.

O(n) time
one-each
gift wrap

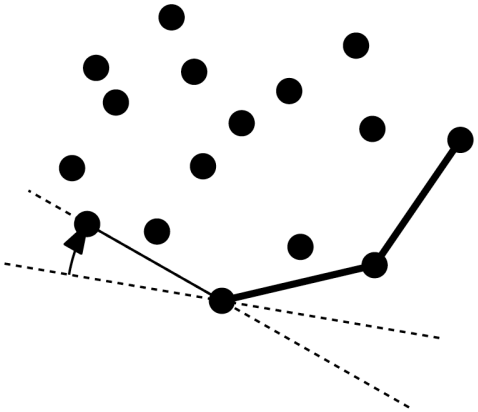
Convex Hull – Gift Wrapping

- Algorithm:
  - Find a point p_1 on the convex hull (e.g. the lowest point).
  - Rotate counterclockwise a line through p_1 until it touches one of the other points (start from a horizontal orientation).
    - Repeat the last step for the new point.
    - Stop when p_1 is reached again.



Gift Wrapping

- Running time is **output sensitive**
  - The time depends on both the size of the input and the size of the output



- Time Complexity: O(*nh*), where *n* is the input size and *h* is the output (hull) size.

$h \simeq \log n$