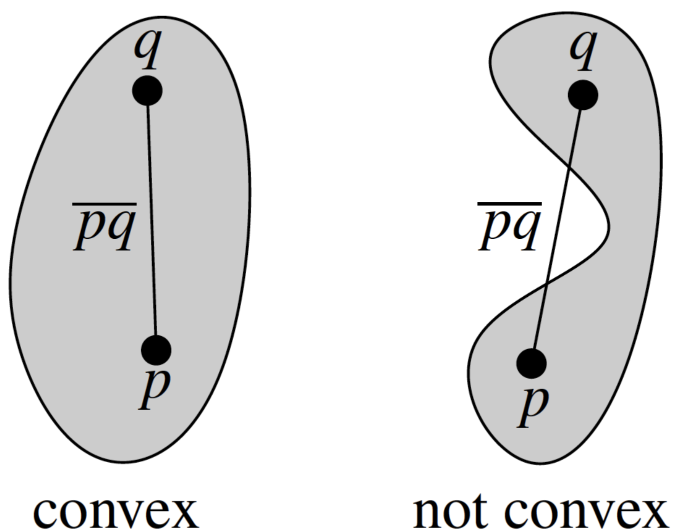
Convex Hulls

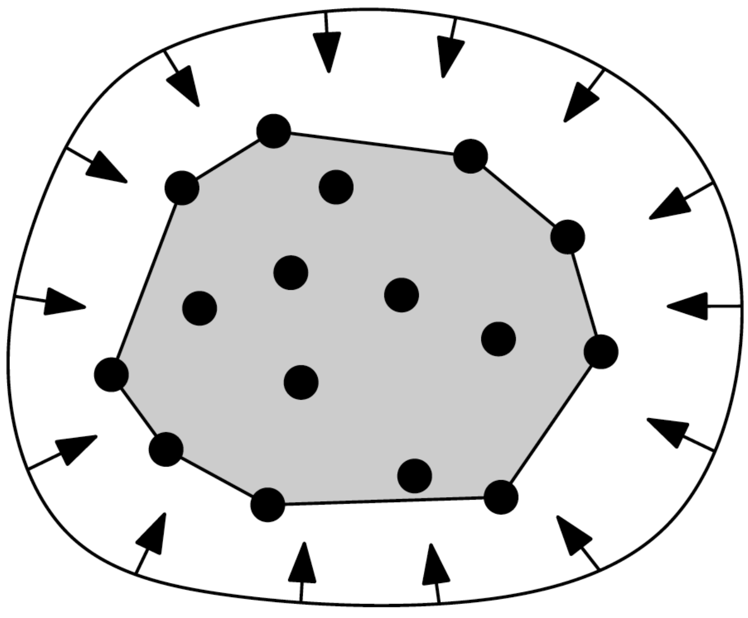
Michael T. Goodrich

Review: Convexity



Convex hull

- Smallest convex set containing all *n* points



Convex hull

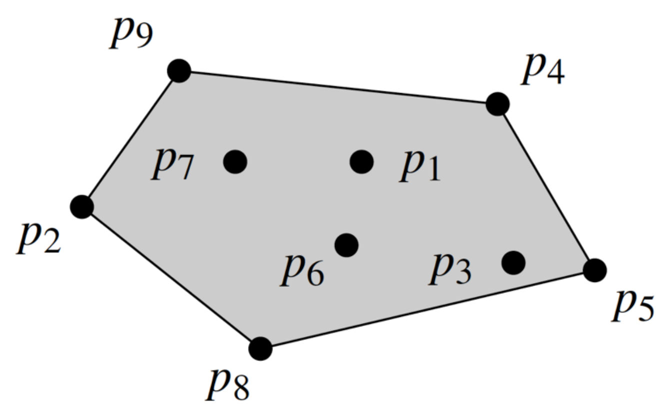
- Smallest convex set containing all *n* points

input = set of points:

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

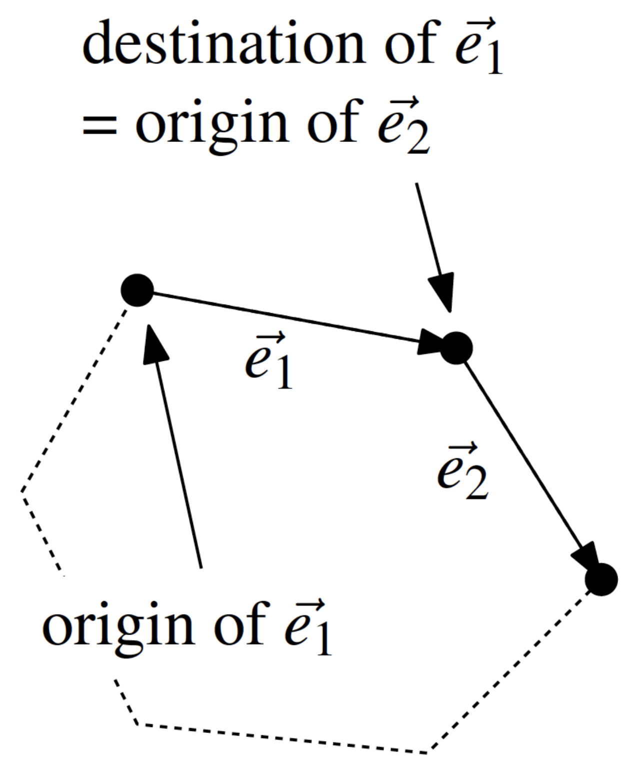
output = representation of the convex hull:

p_4, p_5, p_8, p_2, p_9

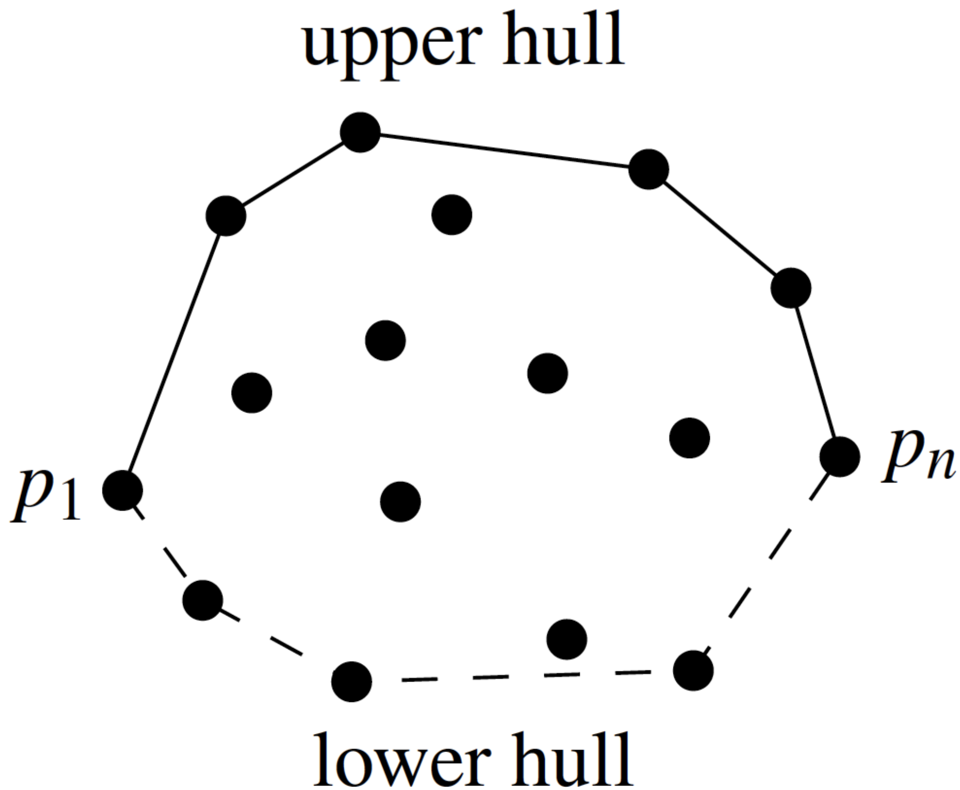


Orientation Test

- right turn or left turn (or straight line)

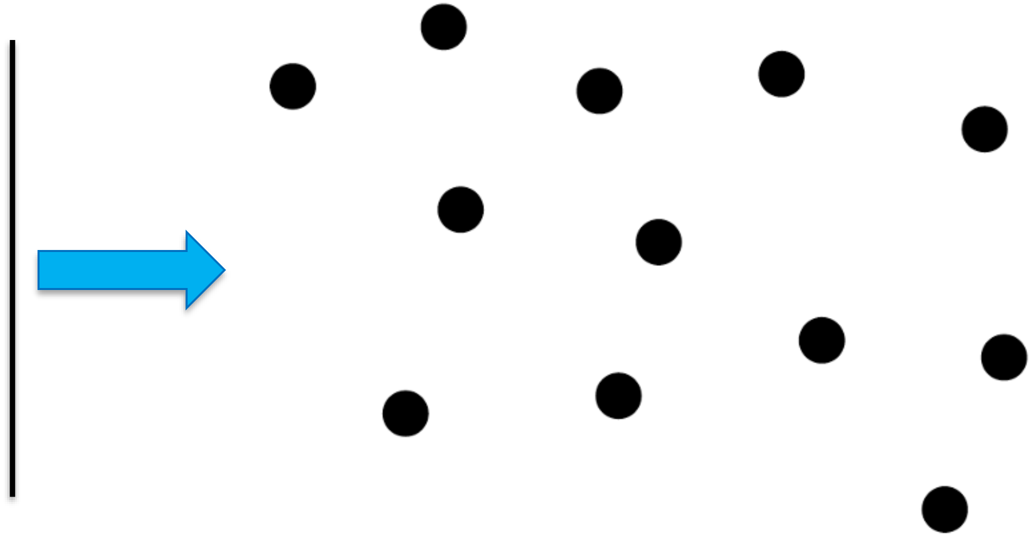


A Better Convex Hull Algorithm



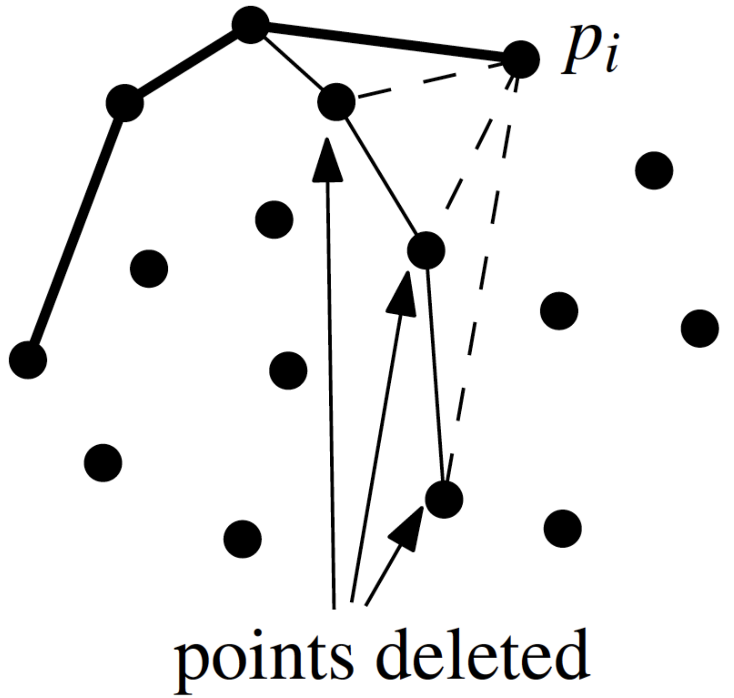
Plane-Sweep Technique

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



Graham Scan Algorithm

- Each point determines an event



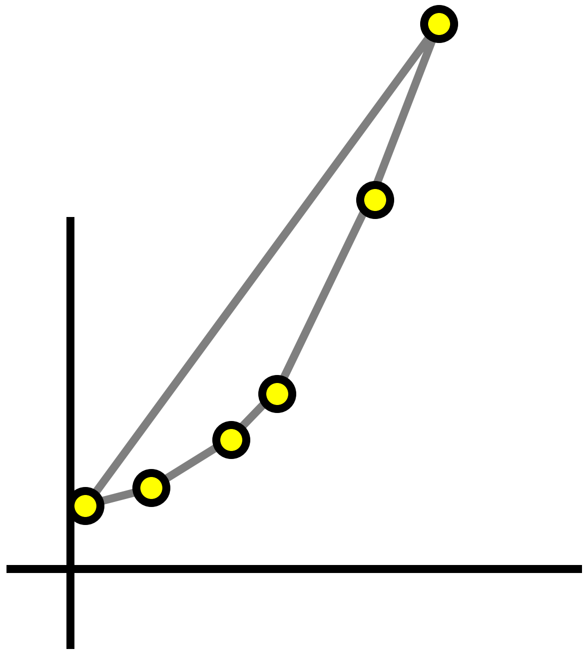
Graham Scan Upper Hull Algorithm

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.

- What is the running time?
- What if the points are already sorted and we can skip step 1?

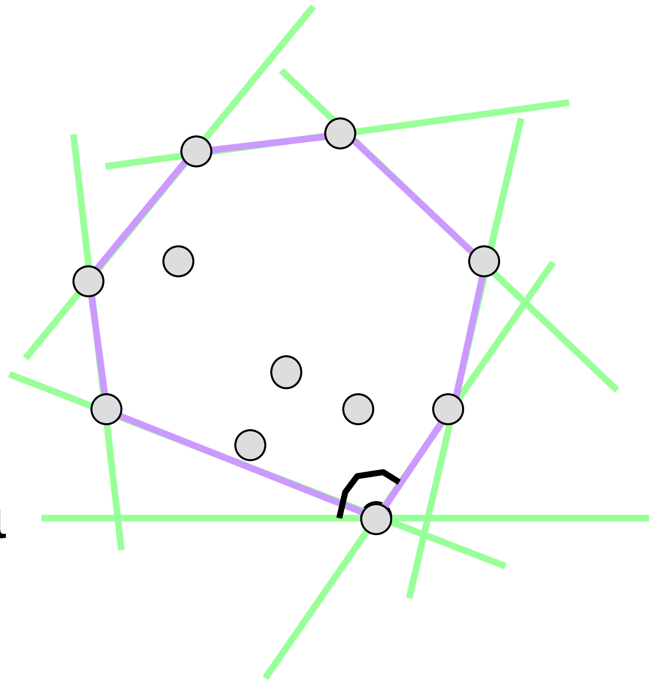
Lower Bound for Convex Hull

- 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 $\Omega(n \log n)$



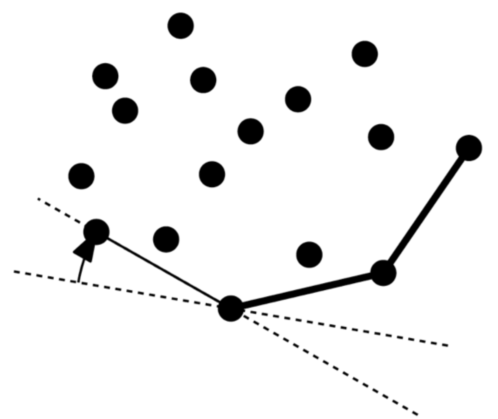
Convex Hull – Gift Wrapping

- **Jarvis March** 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.



Jarvis March 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.

Divide-and-Conquer Convex Hull

if $n \leq 1$ **then**
    return $U = S$.
Divide step: Divide S into S_1 and S_2 of size at most $\lceil n/2 \rceil$ each, such that the points of S_1 have smaller x-coordinates than those in S_2.
Conquer step:
Recursively call ClassicalUpperHull(S_1, U_1).
Recursively call ClassicalUpperHull(S_2, U_2).
Combine step:
Find a ***bridge*** upper tangent edge, $e = (v, w)$, such that $v \in S_1$ and $w \in S_2$ no point of S is above the line $\overline{vw}$.
Remove all points from U_1 (resp., U_2) below e and concatenate the list of remaining points of U_1 with e and the remaining points of U_2, returning this as U.

Divide-and-Conquer Analysis

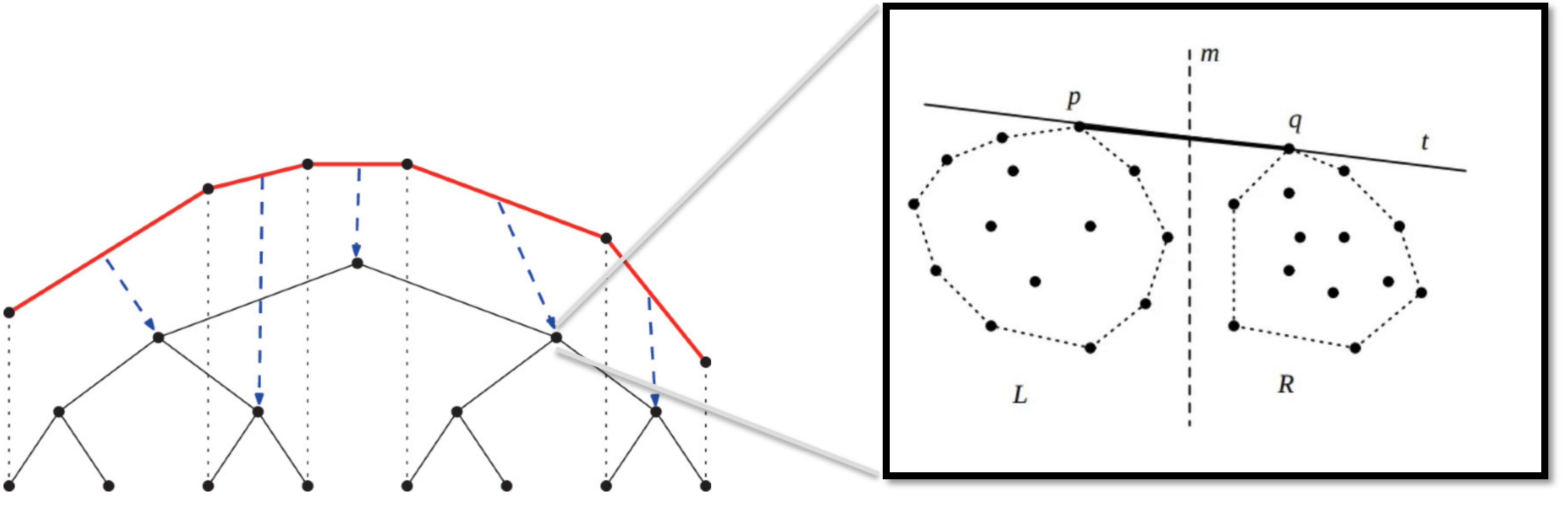
- Assume we can find the bridge edge in O(n) time (more on this later):

- $T(n) = 2T(n/2) + n$

- Implies $T(n)$ is $O(n \log n)$

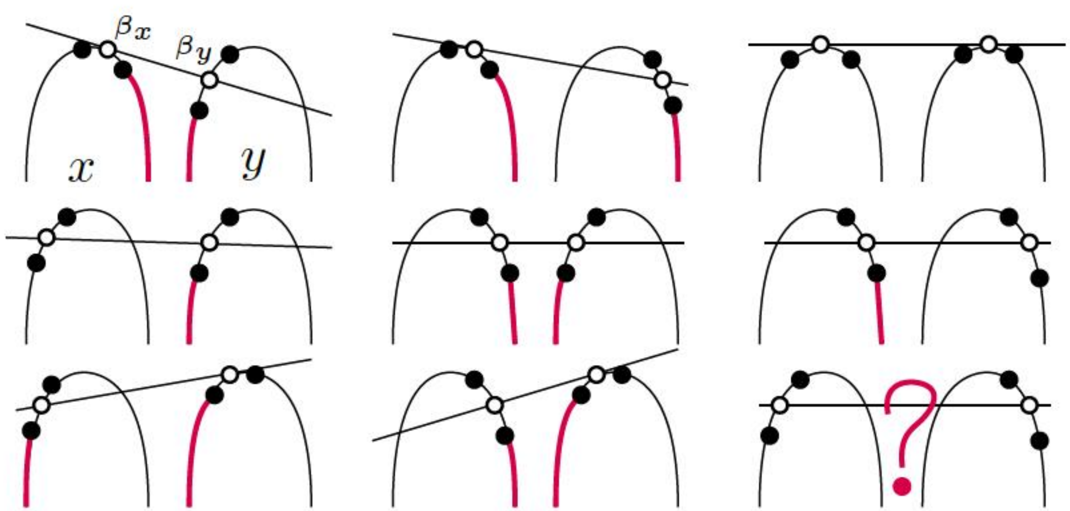
Dynamic Upper Hulls

- Insert or delete points
- Maintain a tree of the recursive calls made in the divide-and-conquer algorithm



Double Binary Search Upper Hull

- Compute the bridge edge in O(log n) time given two upper hulls:



Dynamic Convex Hulls

- Insert/delete in $O(\log^2 n)$ time
- Upper hull queries in $O(\log n)$ time
  - split with vertical line
  - compute 2 hulls recursively => $O(\lg n)$ levels
  - find bridges -- $O(\lg n)$
  - cut+merge hull trees -- $O(\lg n)$

    => $t_u = O(\lg^2 n)$

  - examine bridges
  - recurse left or right

    => $t_q = O(\lg n)$