Convex Hulls

Given a set of points \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\), the **convex hull** is the smallest convex polygon containing all the points.

**Output**: \(A_2 A_8 A_9 A_6 A_{12} A_5\)

**Gift-Wrapping Algorithm**

1. Start with the topmost point, let it be \(A\).
2. Compute the angles from \(A\) to every other point and choose the point with the "leftmost" angle (as compared with the \(x\)-axis at first and later with the last added line segment).
3. Let the new point be \(A\) and repeat step 2 until come back to the starting point.

**In the worst case** \(O(n^2)\)

**Or more precisely** \(O(kn)\) where \(k\) is the points on the hull
Convex Hulls

**Gift-wrapping algorithm** running in time $O(n^2)$, or, more precisely, $O(nk)$ where $k$ is the number of vertices on the hull.

An $O(n \log n)$ algorithm?

\[ \Omega(n \log n) \] steps via reduction from sorting

(i.e. if can do CH faster than $O(n \log n)$, then can sort faster than $O(n \log n)$)

(Contradiction)
Convex Hulls

The **Graham Scan** algorithm

1) sort the points by their angle with respect to a selected point inside the convex hull (e.g., the average of the points)

2) starting at the rightmost/topmost point (definitely on the hull), go linearly through the points in the sorted order:
   - connect the last point on the hull with the new point → if angle with respect to the last edge on the hull is convex, keep the point in the hull;
   - if not convex:
     - remove the last point on the hull, keep doing this while the angle is not convex

**Running time?**

\[ O(n \log n) \rightarrow O(n \log n) \text{ for sorting, } O(n) \text{ for scan} \]
Convex Hulls

A divide-and-conquer algorithm?

Idea:
- Initially, sort the points by their x-axis
  (done only once, not in the recursive procedure)

Recursively:
1) split points into the $\frac{n}{2}$ on the left
   and the $\frac{n}{2}$ on the right
2) recursively find convex hulls for
   the left points and the right points
3) merge the two hulls
   - how? (think about how to
     do this in linear time)

Running time?

Recurrence: $T(n) \leq 2T(\frac{n}{2}) + cn$

Hence: $O(n\log n)$ for sorting
$O(n\log n)$ for $T(n)$

Overall $O(n\log n)$