ALGORITHMS

Outline: How to find the best algorithmic solutions to problems

I. How to Design Algorithms
   - general paradigms — greedy, divide and conquer, dynamic programming, reductions
   - basic repertoire of algorithms
     - sorting (1st year), string algorithms (CS 240)
     - domain specific algs. covered in other courses
       e.g. graph algorithms, linear prog. (C80);
       numerical algs.; algebraic algs in symbolic computing.

II. How to analyze algorithms — How good is this alg.? 
   - time, space, goodness of approximation
   - $O$ notation, worst/avg. case
   - models of computation

III. Lower bounds — Do we have the best alg.? 
   - models of computation
   - basic lower bounds
   - NP-completeness and undecidability.
Case Study

Convex Hull

Given n points in the plane, find their convex hull — the smallest convex set containing the points. (Like putting a rubber band around nails sticking out.)

Why? Convex hull gives "shape" of a set of points — better container than min. bounding box.

Equivalently (and better for alg.) the convex hull is a polygon whose sides are formed by lines l that go through [at least] 2 points and have no points to one side of l.

A. Straightforward Algorithm.

for all pairs of points r, s
find line through r, s
if all other points lie on or to one side of l
then l forms part of convex hull.

Time for n points: O(n^3)

Note: this is high-level pseudocode

e.g. How do we test? Compute equation of line l?
Better (avoids division by 0, overflow)
test sign of cross-product. Only uses +, -, x, <.

Can we do better? Yes — several possibilities.
B. Jarvis' March

Observe that once we have found one line $l$, there is a natural "next" line $l'$. Rotate $l$ through $s$ until it hits $\ell'$, the next point $t$.

How can we find $l'$? Look at all lines through $s$ and another point, and find the "extreme" one in the sense of minimizing angle $\alpha$.

Finding extreme is like finding min. element of a set — $O(n)$
Whole alg. is $O(n^2)$.

This alg. is good to use when the convex hull has few points. It actually takes time $O(n \cdot h)$, $h = \#$ convex hull points.

Can we do better? Yes.

C. Reduction:

Repeatedly finding the min. should remind you of sorting.

Sort points by x-coordinate. Then you can find convex hull with $O(n)$ further work.

Exercise. Hint: Find upper and lower convex hull separately.

A reduction uses an alg. you know (sorting) to solve a new problem.
D. Use divide and conquer

Divide in half by vertical line.
Recursively find convex hull on each side.
Combine by finding upper & lower bridge.

Initial "walk e up" to get upper bridge,
down to get lower bridge.

$O(n)$ to find median, upper & lower bridge.

Get recurrence relation

$T(n) = 2T(\frac{n}{2}) + O(n)$

Like recurrence for merge sort, so $T(n) = O(n \log n)$

Note: this algorithm extends to 3D but the others do not. (e.g. it is not clear how to sort points in 3D)
Can we do better?
In some sense, no.

If we could find convex hull faster, we could sort faster.

Convex hull gives sorted order

This is not rigorous — what is the model of computation?

Challenge: Look up Timothy Chan’s “output sensitive convex hull alg.” $O(n \log h)$

(Note: we saw $O(n \log n)$ and $O(n \cdot h)$, which is better?

Neither — hence Timothy’s alg.)