Algorithms

1 Outline

→ How to find the best algorithmic solutions to problems.

I. How to design algorithms

• basic repertoire of algorithms
  – sorting (1st year), string algorithms (CS 240)
  – domain specific algs. covered in other courses
    e.g., graph algs., linear programming (C&O); numerical
    algs. (AM); algebraic algs. in computer algebra

• general paradigms: divide and conquer, greedy
  dynamic programming, reductions

II. How to analyze algorithms → How good is this alg.? 

• time, space, goodness of approximation 

• O-notation, worst/average case 

• models of computations 

III. Lower bounds → Do we have the best alg.? 

• basic lower bounds 

• NP-completeness and undecidability
2 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 the "shape" of a set of points – better container then a minimal bounding box.

Equivalently (and more useful for thinking about alg.) the convex hull is a polygon whose sides are formed by lines \( \ell \) that go through (at least) 2 points and have no points to one side of \( \ell \).

A. Straightforward algorithm

\[
\text{for all pairs of points } r, s \ \ \ \ \ \ \ \ O(n^2)
\]

find line through \( r, s \)

if all other points lie on or to one side of \( \ell \) \( \hspace{1cm} \otimes \hspace{1cm} \)
then \( \ell \) forms part of convex hull

Time for \( n \) points? \( O(n^3) \)

Note: this is high-level pseudo-code

\[
(x_r-x_1)(y_s-y_1) - (y_r-y_1)(x_s-x_1) > 0
\]

e.g. How do we test \( \otimes \)? Compute the equation of line \( \ell \)?

Better (avoids division by 0, overflow) test sign of cross-product. Only uses +, −, ×, <.

Can we do better? Yes — several possibilities.
B. Jarvis’ march

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

How can we find $\ell'$? 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 $O(nh)$, where $h$ is the number of 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. Recursively find convex hull on each side. Combine by finding upper \& lower bridge.

[details: initial e = edge from max x on left to min x on right. “walk e up” to get upper bridge, down to get lower bridge]

\( O(n) \) to find median, upper and lower bridge

Get recurrence relation

\[ T(n) = 2T(n/2) + O(n) \]

Like recurrence for merge sort, so \( T(n) = O(n \log n) \).
Can we do better?
In some sense, NO.
If we could find convex hull faster, we could sort faster.

![Diagram of convex hull giving points in sorted order](image)

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 $O(nh)$. Which is better? Neither — hence Chan’s alg.]