ALGORITHMS

Admin stuff – from web page https://www.student.cs.uwaterloo.ca/~cs341/

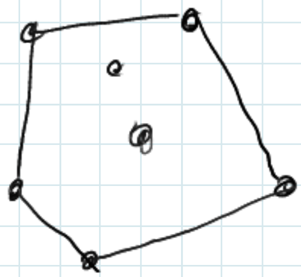
<u>Outline</u> How to find the <u>best</u> algorithmic solutions to problems

I. How to Design Algorithms
- general paradigms – greedy, divide and conquer, dynamic programming, reductions
- basic repertoire of algorithms
  - sorting (1st year), string algorithms (CS240)
  - domain specific algs. covered in other courses e.g. graph algorithms, linear prog. (C&O); 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.

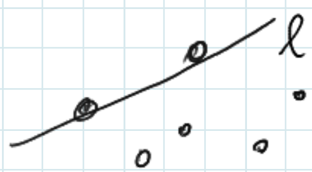
Case Study Convex Hull



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.

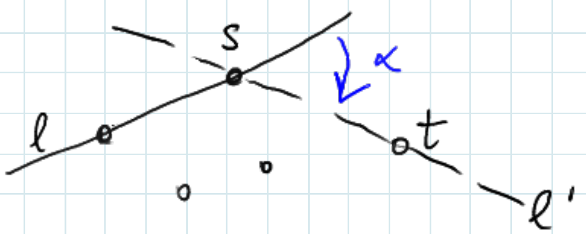
for all pairs of points r, s
    find line through r, s
    if all other points lie on or to one side of ℓ
    then ℓ forms part of convex hull.

Time for n points: $O(n^3)$

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 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 α.

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.]

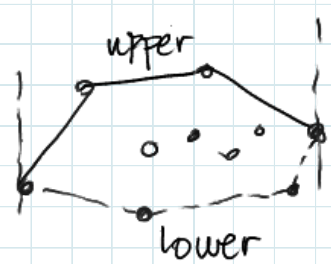
Can we do better? Yes.

C. Reduction.

Repeatedly finding the min. should remind you of sorting.

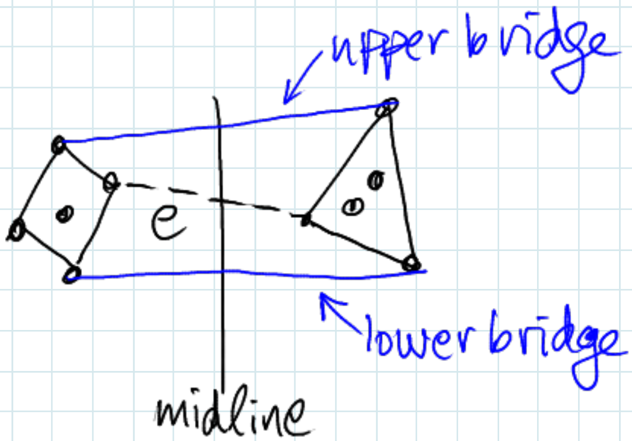
Sort points by x-coordinate. Then you can find convex hull with $O(n)$ further work.

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
Recursively find convex hull on each side
Combine by finding upper & lower bridge

[details: e = edge from max x on left to min x on right.
"Walk up" to get upper bridge
...]

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

Get recurrence relation

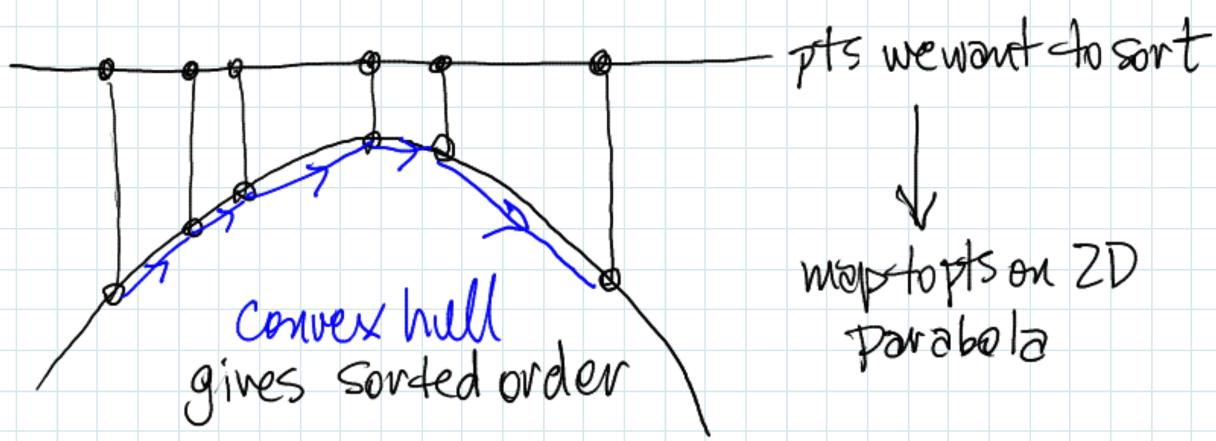
$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

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



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

<u>Challenge</u> 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.]

Analyzing Algorithms

Definitions An algorithm is a finite answer to an infinite question.

Problem – specification of infinite set of inputs
– specification of corresponding outputs

[Note: can be difficult in practice to distinguish infinite from large finite, e.g. chess – finite, but large enough to be very hard & interesting]

Algorithm – well defined computational procedure to go from any input to corresponding output.
For our purposes – described in pseudo code.

Analyze an Algorithm – measure time and space used by the algorithm as a function of input size
– measured not by running the program, but by using an abstract model of computing.

Models of Computation

- specify the elementary computations out of which algorithms are built.
- specify measure of time, space, input size.

Bottom line: model should reflect (but simplify) reality.