CS 341 - lecture 1 Intro

Algorithm ~800 AD Abu Ja'far Muhammad ibn Musa Al-Khwarizmi
(CS 135)

Basics - similar to CS 240

- Focus on correctness, efficiency, ...
- Analysis
  - Runtime of an Alg as a function of a measure of input size n
  - machine independent
    - models of computation
  - Time, space
  - Asymptotic Notations: big-O, Θ, etc
  - Worst-case, Average case, etc

Pseudo code - try to be realistic about "elementary operations"

Algorithms - general paradigms: divide & conquer greedy, dynamic programming, reductions

Domain Specific: graph algs, linear prog => CEO, numerical => AM, etc

Lower Bounds → Do we have the best alg?

- basic lower bounds
- NP-completeness and undecidability

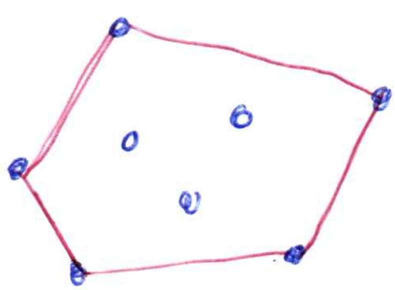
Case Study: Convex Hull

- shortest perimeter fence
- robot motion planning, etc

Problem: Given n points in the plane,
Find their convex hull:
the smallest convex set containing the points

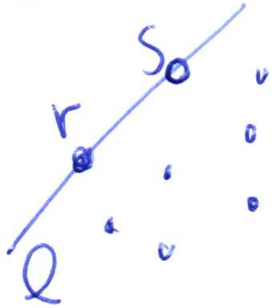
- think: put rubber band around points

eg



Why? gives "shape" of a set of points - better than a bounding box

Equivalently: the convex hull is a polygon
1
whose sides are formed by lines ℓ that go through at least 2 points and <u>have no points to one side of ℓ</u>.



Alg 1: Find all lines that satisfy 1

- "all lines" - each pair of points r, s forms a line ℓ

$O(n^2)$ pairs

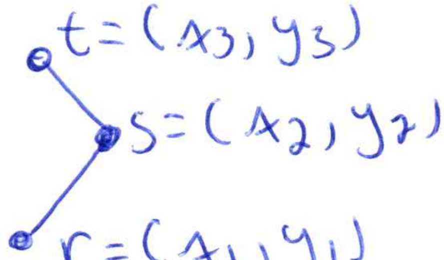
- if all other points lie on one side of ℓ then ℓ is part of convex hull.

$O(n)$ check points

$O(n^3)$

Test if points all on one side of ℓ (or on)

- various methods



Look at sign of determinant

$$\begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix}$$

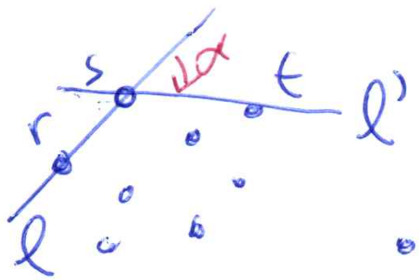
cross product

$$= [(x_2 - x_1) * (y_3 - y_1)] - [(y_2 - y_1) * (x_3 - x_1)]$$

- if $= 0$ then on line (check > 0)?
- if all one side, will have same sign

Alg 2: Jarvis March

Once we have 1 line ℓ, there is a natural next line ℓ'.



Rotate ℓ at s until it hits next point t.

- points are not stored in any particular order

Finding ℓ':
- compute all lines through s and another point — $O(n)$ such lines
- Find one that minimizes α — $O(n)$ find min α

$O(n^2)$

* only need to iterate #edges on convex hull times.

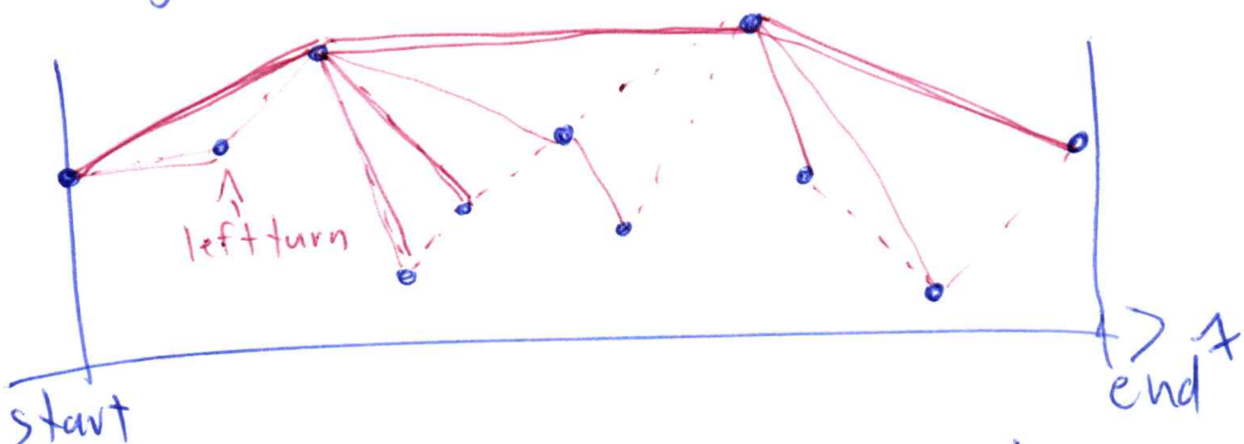
- Let h be # {edges / points} on C.H.

$\Rightarrow O(h \cdot n)$ · good if $O(1)$ points on C.H.

Alg 3 Reduction

- Solve a new problem (convex hull) by using an alg you already know (sorting)

- Sort points by x-coord
- Start at 1st point (leftmost)
- Traverse points in order to find edges of the convex hull



- going around a convex hull should only make right turns
- if turn left, go back, remove old edge
  make new one

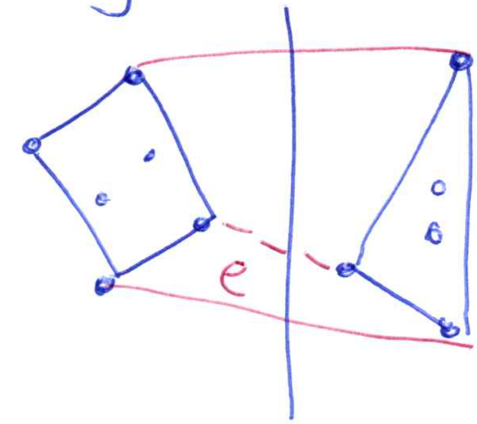
Similarly find lower convex hull.

O(nlogn) sorting + O(n) extra work: traverse points, etc

Runtime: O(n) + O(time to sort)

- if we can sort faster, this also improves

Alg 4 Divide and Conquer



Divide points in half.

Find convex hull on each side.

Conquer: combine by finding upper & lower bridge

Initial e = edge from max x on left to min x on right

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

O(n) to find median, upper & lower bridges

Recurrence Relation: $T(n) = 2T(n/2) + O(n)$

similar to mergesort $\Rightarrow O(n \log n)$

Timothy Chan: "Output Sensitive Convex Hull Alg"

- $O(n \log h)$

Alg 2: $O(nh)$ which is better?

Alg 3, 4: $O(n \log n)$

Can we do better?

If we can find convex hull faster, then we can sort faster.

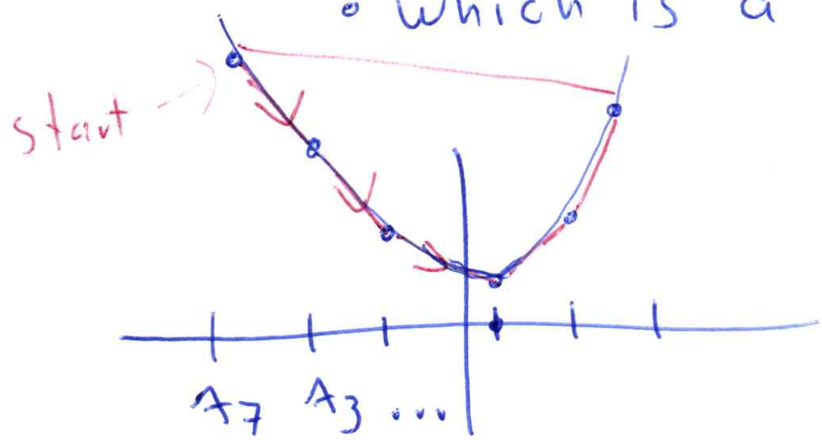
Sorting Alg:

1 Dim

Given n "points": $x_1, x_2, \ldots, x_n$

Map each x_i to (2D) (x_i, x_i^2)

- the 2D points now form a parabola
  - which is a convex shape.



- Find convex hull
- Traversing C.H. gives the points in sorted order by x

Runtime: $O(n) + O(\text{time to find C.H.})$

- if comparison based, $\Omega(n \log n)$
  - lower bound on sorting comparison based