Lec 1 Case Study: Convex Hull

Thanks to Anna Lubiw and other previous CS 341 instructors.

Problem: Given n points in the plane, find their *convex hull*: the smallest convex set containing the points.

Why?

- Gives a better "shape" of a set of points rather than simply using a minimal bounding box.
- Shortest perimeter fence
- Robot motion planning

Algorithm 1: Brute Force

Equivalent Definition: 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.

Alg 1: For all pairs of points r, s, define a line ℓ through r and s. If all other points lie on only one side of ℓ then ℓ is part of the convex hull.

Runtime Alg 1:

- $\Theta(1)$ to check which side of ℓ a point is on.
- $O(n)$ points to check - if all on the same side.
- n^2 pairs of points (possible lines to check).

Total: $O(n^3)$

Algorithm 2: Jarvis March

Alg 2: Once we have a first line ℓ through points r and s, there is a natural next line ℓ':

- Rotate ℓ at point s until it hits another point, call it t.

Finding ℓ':

- Compute all lines through s and another point.
- From all such lines, find the one that minimizes angle α with ℓ.

Runtime Alg 2:

- Finding a first line can be done in $O(n)$.
- Next line: consider $O(n)$ lines, find min α from a set of size $O(n)$.
- Convex hull may be composed of $O(n)$ points (lines).

Total: $O(n) + O(n^2)$ which is $O(n^2)$

Note: Let h be the number of points on the convex hull.
Then, runtime is $O(hn)$. If only $O(1)$ points on hull then runtime is $O(n)$.

Algorithm 3: Reduction Approach

Reduction: Solve a new problem (convex hull) by using an algorithm you already know (sorting).

Alg 3: Sort points by x-coordinate.

- Traverse points (from leftmost) to find the "top" edges of convex hull. If next point makes a concave angle, skip it - may have to do some backtracking.
- Repeat this process to find the "bottom" edges of convex hull.

Runtime Alg 3:

- Sort points: $O(n \log n)$.
- Traverse points twice to find top/bottom of convex hull $O(n)$ - can argue each time we backtrack a point is removed from consideration.

Total: $O(n \log n)$

Note: If we can improve sorting, we can improve this runtime.

Algorithm 4: Divide and Conquer Approach

Divide: Divide points in half (left and right).
Find the convex hull on each side. Base case?

Conquer: Combine by finding upper and lower bridges between the two convex hulls.

- Initial egde e: edge from max x on left to min x on right.
- "walk e up" to find upper bridge.
  "walk e down" to find lower bridge.

Runtime Alg 4:

- $O(n)$ to find median (divide points), upper and lower bridges.
- Recurrence Relation: $T(n) = 2T(n/2) + O(n)$

Total: $O(n \log n)$

Can we do better?

Alg 1: $O(n^3)$
Alg 2: $O(n^2)$ with special case $O(hn)$
Alg 3 and Alg 4: $O(n \log n)$

Which is better: $O(hn)$ or $O(n \log n)$?

If we can find the convex hull faster, then we can sort faster!

A Sorting Algorithm: Given n unsorted numbers, $x_1, x_2, \ldots, x_n$.

- Create 2D points $(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)$.
  Note, the points now form a 2D parabola which is a convex shape.
- Find convex hull. Traversing it gives original numbers in sorted order.

Runtime: $O(n)$ + time to find Convex Hull
Recall: If comparison based, sorting is $\Omega(n \log n)$

Timothy Chan (1996): "Output sensitive convex hull" in $O(n \log h)$.