Convex Hull

COMP 215 Lecture 5
Computational Geometry

• The area of CS concerned with solving geometric problems.

• Examples:
  − Finding intersections between line segments.
  − Finding closest pairs of points.
  − Finding the convex hull. (More on this in a second.)

• Uses in:
  − Graphics.
  − Robotics.
  − VLSI design.
  − etc.
The convex hull of a set $Q$ of points is the smallest convex polygon $P$ for which each point $Q$ is either on the boundary of $P$ or in its interior. (Introduction to Algorithms, Cormen et. al. 2001)

The problem: For an arbitrary set of points $Q$, find the corresponding $P$. 
Line Segments Properties

- First question:
  - Given two directed line segments: \( \overrightarrow{p_0 p_1} \) and \( \overrightarrow{p_0 p_2} \), is \( \overrightarrow{p_0 p_1} \) clockwise from \( \overrightarrow{p_0 p_2} \)?
Cross Product

- 2d cross product: \( p_1 \times p_2 = x_1 y_2 - x_2 y_1 \)

- When this is positive \( p_1 \) is clockwise from \( p_2 \).
- When this is negative \( p_2 \) is clockwise from \( p_1 \).
Solution to Clockwise Problem

• The original question:
  – Given two directed line segments: $\overrightarrow{p_0p_1}$ and $\overrightarrow{p_0p_2}$, is $\overrightarrow{p_0p_1}$ clockwise from $\overrightarrow{p_0p_2}$?

• The solution: move $p_1$ and $p_2$ to use $p_0$ as the origin, and calculate cross product:

$$ (p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0) $$

• If this is positive then $\overrightarrow{p_0p_1}$ is clockwise from $\overrightarrow{p_0p_2}$. 
Clockwise Turns

• Next question: do two consecutive line segments $\overrightarrow{p_0p_1}$ and $\overrightarrow{p_1p_2}$ make a clockwise, or counterclockwise turn at $p_1$?

• This is almost the same as the previous question:

\[
(p_2 - p_0) \times (p_1 - p_0) = (x_2 - x_0)(y_1 - y_0) - (x_1 - x_0)(y_2 - y_0)
\]

• Positive is a clockwise turn, negative is counterclockwise.
Back To Convex Hull

• Any ideas for a good algorithm?
Candidate Algorithm

• First, sort all points by their x coordinate.
  - (Theta(n lg n) time)

• Then divide and conquer:
  - Find the convex hull of the left half of points.
  - Find the convex hull of the right half of points.
  - Merge the two hulls into one. (this is the tricky step.)
Convex Hull Pseudocode

//input: the number of points n, and
//an array of points S, sorted by x coord.
//output: the convex hull of the points in S.

point[] findHullDC(int n, point S[]) {
    if (n > 1) {
        int h = floor(n/2);
        m = n-h;
        point LH[], RH[]; //left and right hulls
        LH = findHullDC(h, S[1..h]);
        RH = findHullDC(m, S[h+1..n]);
        return mergeHulls(LH.size(), RH.size(),
                           LH, RH);
    } else {
        return S;
    }
}
Merging Hulls

• Big picture:
  - first find the lines that are upper tangent, and lower tangent to the two hulls (the two red lines)
  
  - Then remove the points that are cut off.
Finding Tangent Lines

- Start with the rightmost point of the left hull, and the leftmost point of the right hull:
  - While the line is not upper tangent to both left and right:
    - While the line is not upper tangent to the left, move to the next point (counter-clockwise).
    - While the line is not upper tangent to the right, move to the next point (clockwise).
Checking Tangentness

- How can we tell if a line is upper tangent to the left hull?
- The pair of line segments $\overline{p_r p_l}$, and $\overline{p_l p_{l-ccw}}$ should make a CCW turn at $p_l$.
- The same goes for $\overline{p_r p_l}$ and $\overline{p_l p_{l-cw}}$. 
The Tricky Bits

- Hulls need to be maintained in order (CW or CCW).
- Needs to be stored in a data structure that allows wrapped forward and backward iteration.
  - Circularly linked list.
  - Array with clever indexing.
- Several ways to handle base cases:
  - Special code to create hulls of size 1, 2, and 3?
  - Clever merging that can merge a hull of size 2 with a hull of size 1? (or 1 and 1, or 3 and 2, etc.)
Analysis & PP

- Let's talk about running time.
- Then let's talk about the programming project.