Professor Diane Souvaine
Scribe: M. McPike

Lectures 1, 2, 3: Convex Hull Algorithms

1 Definitions

- **Convex:** $S \subseteq \mathbb{R}$ is convex if for any points $p, q \in S$ the line segment between p and q is contained in S.
  - A line intersects a convex polygon at 0, 1, or 2 vertices.
- **Convex Hull** of $S \subseteq \mathbb{R}$: $CH(S)$ is the smallest convex set containing S.
  Points guaranteed to be on $CH(S)$: points with min/max x/y coordinate, point furthest from centroid
- The boundary of a polygon is defined by vertices listed in counterclockwise order (main text uses clockwise). Line segments connect consecutive vertices. Any two line segments intersect either at a vertex or not at all.
- A polygon is **non-simple** if two nonconsecutive edges share a vertex, e.g. a bow-tie. Otherwise, it is simple.
- A polygon is **monotone** with direction m if every line with slope $\frac{-1}{m}$ intersects P at 0, 1, or 2 points.
- A polygon is **star-shaped** if there exists a point z such that for any point p in the polygon, the line segment between p and z lies in P. The collection of all such points z is called the kernel.
- **Left hand turn:** $< ABC$ is a left hand turn if C is in the left half plane bound by the line through A and B. In other words, A, B, C appear in counterclockwise order on the boundary of $\triangle ABC$.
- L is a **supporting line** of polygon P if at least one point of P is on L and the interior of P is entirely in one half plane defined by L

2 Test for convexity

Assume $p_0, p_1, \cdots, p_n = p_0$ is a circularly linked lists that defines a polygon (in counterclockwise order). P is convex $\implies$ every turn $< p_i p_{i+1} p_{i+2}$ is a left hand turn.

$< pqr$ is a left hand turn $\iff$ the determinant below is positive:

$$\begin{vmatrix} x_p & y_p & 1 \\ x_q & y_q & 1 \\ x_r & y_r & 1 \end{vmatrix}$$

3 Convex Hull Algorithms

3.1 Lower bound for convex hull algorithms

We have to look at every point in S to find $CH(S)$, so $\Omega(n)$ is a lower bound. We can use a reduction argument to find a tighter lower bound. We will describe a comparison based sorting algorithm that uses a convex hull algorithm.

- Input: unsorted list $x_1, \cdots, x_n$
- Output: sorted list
- For each x_i, let $p_i = (x_i, x_i^2)$ be a point in $S \subset \mathbb{R}$ - $\Theta(n)$
- Use a convex hull algorithm to find $CH(S)$ - $T_{CH}(n)$

- Find the point p in S with minimum x-coordinate - $\Theta(n)$
- Read the points that define $CH(S)$, starting with p.
- We described a sorting algorithm that takes $\Theta(n) + T_{CH}$.
- Lower bound for comparison based sorting is $\Omega(n \log n)$, so this is also a lower bound for convex hull.

3.2 Slow Convex Hull

- Page 3 of main text
- Steps
  - For every pair (p_i, p_j) of distinct points in S, and for every $p_k \in S \setminus \{p_i, p_j\}$,sue the "left hand turn test" to determine which side of line $\overleftrightarrow{p_i p_j}$ p_k lies on.
  - If every p_k lies on the right side of $\overleftrightarrow{p_i p_j}$, then add $\overrightarrow{p_i p_j}$ to the edge set of $CH(S)$.
  - Order the edges
- $O(n^3)$

3.3 Jarvis March (Gift Wrapping)

3.3.1 Time

$T(n) = O(nh)$ where h is the number of points that define $CH(S)$. In the worst case, $h = n$.

3.3.2 Steps

- Identify a point p_0 that is in $CH(S)$, e.g. the point in S with minimum x-coordinate.
- To find the next point p_1 in $CH(S)$: For all $q \in S$, find the slope of $\overleftrightarrow{p_0 q}$. Let p_1 be the point with the smallest (most negative) such slope.
- In general, p_i is the point such that $< p_{i-1} p_i r$ is a left hand turn for all $r \in S$.
- To find p_i: Choose the first two points $r_1, r_2 \in S \setminus \{p_0, \cdots, p_{i-1}\}$. If $< p_{i-1} r_1 r_2$ is a left hand turn, replace r_2 with r_3 and repeat with $< p_{i-1} r_1 r_3$. Else, replace r_1 with r_2 and repeat with $< p_{i-1} r_2 r_3$.

3.4 Graham Scan (Incremental Algo)

- Page 6 of text
- We build the upper hull and lower hull separately. We incrementally add points to the hulls left to right.
- Sort points in order of increasing x-coordinate: $p_1, \cdots p_n$. Note that p_1 and p_n are both in the convex hull. This takes $O(n \log(n))$.
- To build the lower hull:
  - Push p_1 and p_2 to a stack. If $< p_1 p_2 p_3$ is a left hand turn, push p_3. Else, pop p_2 then push p_3. Now we are at p_4.
  - When we get to p_i: If the top two stack points and p_i make a left hand turn, push p_i. Else, pop the stack then push p_i.
  - Stop after adding p_n (the right most point) to the stack. Now the stack contains the lower convex hull points in order.
- Each point p_i gets pushed once and gets popped at most once. Building the convex hull after pre-sorting is $\theta(n)$.
- Time: $O(n \log n) + \theta(n) = O(n \log n)$

3.4.1 Divide and Conquer Algorithm

We recursively find the convex hull of S. First, we presort the points in S by x-coordinate, which takes $O(n \log n)$. This makes it easier to "bridge" two convex hulls.

Base case: Pair the points in S, e.g. the first two in sorted order, etc.

Inductive step: All points in S_1 are to the left of all points in S_2. We have $CH(S_1)$ and $CH(S_2)$. To build $CH(S_1 \cup S_2)$, we need to add two edges that bridge $CH(S_1)$ and $CH(S_2)$ and we need to throw out some edges in $CH(S_1)$ and $CH(S_2)$. A bridge between $CH(S_1)$ and $CH(S_2)$ is a line that supports both polygons. To build the lower bridge, we can use a ladder ($O(n)$) or binary search ($O(\log n)$).

Time Complexity: $T(n) = 2T(n/2) + f(n)$ where $f(n)$ is $O(n)$ or $O(\log n)$ depending on how we build the bridge. $T(n) = \Theta(n \log n)$ or $T(n) = \Theta(n)$, respectively.

Note: This algorithm is continued in the next lecture notes and on page 8 of the "User Guide" co-authored by Diane.

3.4.2 Quick Hull

This algorithm is used a lot because it is $\Theta(n)$ in the best case. In the worst case, it is $\Theta(n^2)$. We iteratively find points in $CH(S)$ as follows:

- Find the points $p_{min}, p_{max} \in S$ with minimum and maximum x-coordinate. Add these points to $CH(S)$.
- Calculate the slope of the line through p_{min}, p_{max}
- Find the highest and lowest points in S relative to that slope (e.g. slide the line through p_{min}, p_{max} up and down without changing its slope)
- Add the points r_{min}, r_{max} found above to $CH(S)$
- Iterate with the line through r_{min}, p_{min} and the line through r_{min}, p_{max}. Do the same for r_{max}.