COMPSCI 330: Design and Analysis of Algorithms April 12, 2016

Convex Hull

Lecturer: Kyle Fox *Scribe: Fred Zhang*

1 Overview

In this lecture we discussed two-dimensional convex hull problem and gave two algorithms for computing convex hull.

2 Convex Hull

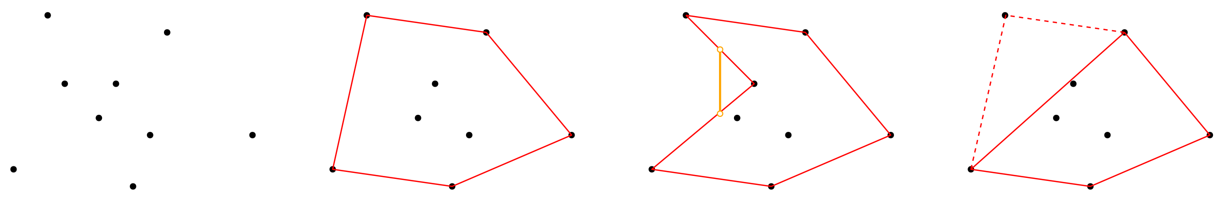

Figure 1: (i). A point set P in $\mathbb{R}^2$. (ii). Convex hull of P. (iii). A violation of convexity, indicated by the orange segment. (iv). A proper subset of $\mathcal{CH}(P)$ cannot possibly contain P.

Definition 1. *Let $P \subseteq \mathbb{R}^2$ be a set of n points. Then the* convex hull *of P is the smallest convex polygon that contains all points in P, denoted by $\mathcal{CH}(P)$.*

Here, we make two more comments on the definition; also see figures above.

- Convex: For any points p, q inside a convex polygon, the line segment $\overline{pq}$ lies in the polygon.
- Smallest: Any convex proper subset of $\mathcal{CH}(P)$ cannot contain P. Informally, that means you cannot cut off any part of the convex hull.

3 Formal Problem

Given a set of n points, we want an algorithm to output its convex hull. Now let's try to formalize the problem a little bit.

The input of our algorithm would be the Cartesian coordinates of the points in P as two arrays, $X[1 \cdots n]$ and $Y[1 \cdots n]$.

Observation 1. *Any vertex of $\mathcal{CH}(P)$ is a point in P.*

Our algorithm should output a circular doubly-linked list of the vertices of $\mathcal{CH}(P)$. By the Observation 1, it consists only of points in P. Concretely, it can be represented as two arrays, $next[1 \cdots n]$ and $prev[1 \cdots n]$. If point i is a vertex of the convex hull, then $next[i]$ is its next vertex in counter-clockwise order; similarly define $prev[i]$.

4 Algorithms

4.1 Preliminaries

We assume the *general position* of input points, which means that

(i) no three points are collinear, and that

(ii) no two points share a horizontal or vertical line.

First we start by looking at some simple cases.

1. If P contains one point, then $\mathcal{CH}(P)$ is simply the point itself.
2. If P contains two points, then $\mathcal{CH}(P)$ is the two points plus the edge that connects them two.
3. Things get trickier when $|P| = 3$. We know that the three edges that connect the input points forms $\mathcal{CH}(P)$. However, it is required that they be output in counter-clockwise order.

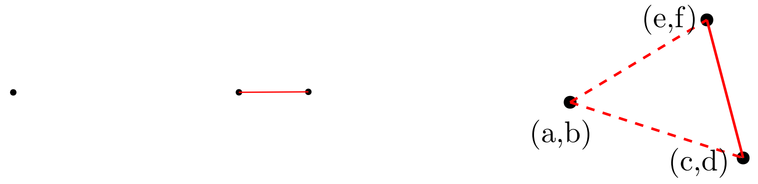

Figure 2: (i). 1-point case. (ii). 2-point case. (iii). 3-point case

Without loss of generality, assume that (a, b) is the leftmost point. Then

$$\begin{aligned}(a,b),(c,d),(e,f) \text{ is in counter-clockwise order} &\iff \text{slope at } \overline{(a,b)(c,d)} \leq \text{ slope at } \overline{(a,b)(e,f)} \\ &\iff \frac{d-b}{c-a} \leq \frac{f-b}{e-a} \\ &\iff (d-b)(e-a) \leq (f-b)(c-a)\end{aligned}$$

This gives use a way to test if three (ordered) points are in counter-clockwise order.

4.2 March algorithm

Observation 2. *The leftmost point of P has to be a vertex of $\mathcal{CH}(P)$.*

This observation gives us a starting point of our algorithm. We can simply find the leftmost point, *i.e.*, the point with the smallest x coordinate, and somehow wrap around the point set.

Observation 3. *Let P be a point on $\mathcal{CH}(P)$. Then $\overline{pq}$ is an edge of $\mathcal{CH}(P)$ if and only if $\forall r \in P \setminus \{p, q\}$, r is on the left of $\overline{pq}$.*

Informally, it claims that for any edge of a convex hull, the entire convex hull lies to the left of the edge. This leads us to an algorithm that looks for the smallest slope at each iteration and finishes when it goes back to the leftmost starting point.

```
MARCH($X[1 \cdots n]$, $Y[1 \cdots n]$)
    $l \leftarrow 1$
    for $j \leftarrow 2$ to $n$ do
        if $X[j] < X[l]$ then
            $l \leftarrow j$
    end
    $p \leftarrow l$
    repeat
        if $p \neq 1$ then
            $q \leftarrow 1$
        else
            $q \leftarrow 2$
        for $i \leftarrow 2$ to $n$, skipping $p$ and $q$ do
            if $ccw(p, i, q)$ then
                $q \leftarrow i$
        end
        $next[p] \leftarrow q$, $prev[q] \leftarrow p$
        $p \leftarrow q$
    until $p = l$
    return $next[1 \cdots n]$, $prev[1 \cdots n]$
```

Algorithm 1: March algorithm for convex hull, where $ccw(\cdot)$ is the procedure testing if three points are in counter-clockwise order; see Section 4.1.

Let h be the number of edges of $\mathcal{CH}(P)$. The outer loop takes h iterations, since each time it adds a new vertex. The inner *for* loop runs in $O(n)$ time. Hence, the March algorithm has $O(nh)$ run time.

In a certain sense, the algorithm is analogous to selection sort, where you look for the minimum value ahead of you in each iteration and insert it in right position. Yet, we know that selection sort is not optimal. The next algorithm finds the next vertex faster.

4.3 Scanning algorithm

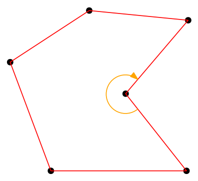

Figure 3: Note that the non-convex polygon contains a right turn.

Lemma 4. *A polygon with vertices on the points of P is the convex hull if and only if it contains P and has no clockwise turn.*

Intuitively, the lemma is saying that if you walk through the edges of a convex hull counter-clockwise, you may never turn right. In other words, right turn violates convexity; see the figure above.

```
1  SCAN($X[1 \cdots n]$, $Y[1 \cdots n]$)
2      $l \leftarrow 1$
3      for $j \leftarrow 2$ to $n$ do
4          if $X[j] < X[l]$ then
5              $l \leftarrow j$
6      end
7      Swap two points, 1 and $l$
8      Sort points $2 \cdots n$ using $ccw(l, \_, \_)$ as comparison criterion.
9      $prev[1] \leftarrow n$
10     for $i \leftarrow 1$ to $n$ do
11         $next[i-1] \leftarrow i$, $prev[i] \leftarrow i-1$
12     end
13     $p \leftarrow 1, q \leftarrow 2, r \leftarrow 3$.
14     repeat
15         if $ccw(p, q, r)$ then
16             $p \leftarrow q, q \leftarrow r, r \leftarrow r+1$
17         else
18             $next[q] \leftarrow 0$, $prev[q] \leftarrow 0$
19             $next[p] \leftarrow r$, $prev[r] \leftarrow p$
20             $q \leftarrow p$, $p \leftarrow prev[p]$
21     until $r = l$
22     return $next[1 \cdots n]$, $prev[1 \cdots n]$
```

Algorithm 2: Scanning algorithm for convex hull.

Observe that the *if* block in the main loop may run at most $O(n)$ times, and same for the *else* block. Hence, the runtime of algorithm is determined by the sorting step in line 8, which is $O(n \log n)$. For a visualization of the scanning algorithm, see [Vis].

5 Summary

We defined the convex hull problem, and presented two algorithms for computing convex hull in 2D; one runs in $O(nh)$ time and the other in $O(n \log n)$ time.

References

[Vis] VisuAlgo - Computational Geometry. http://visualgo.net/geometry.html.