The Convex Hull

Definition 1
1. A set $C$ of points is convex if for all $p, q \in C$ the whole line segment $pq$ is contained in $C$.
2. The convex hull of a set $Q$ of points is the smallest convex set $C$ that contains $Q$.

Observation 2
The convex hull of a finite set $Q$ of points is a convex polygon whose vertices (corner points) are elements of $Q$.

The Convex Hull Problem
Input: A finite set $Q$ of points in the plane
Output: The vertices of the convex hull of $Q$ in counterclockwise order.

Example:
Output of a convex-hull algorithm: $a, b, c, g, j$

Polar Angles
The polar angle of a point $q$ with respect to a point $p$ is the (as usual anti-clockwise) angle between a horizontal line and the line through $p$ and $q$.

Lemma 3
There is an algorithm that, given points $p_0, p_1, \ldots, p_n$, sorts $p_1, \ldots, p_n$ by non-decreasing polar angle with respect to $p_0$ in $O(n \lg n)$ time.
Graham's Scan

IDEA

- Let \( p_0 \) be a "bottom-most" point in the set. Start walking around the points in the order of increasing polar angles.
- As long as you turn left, keep on walking.
- If you have to turn right to reach the next point, discard the current point and step back to the previous point. Repeat this until you can turn left to the next point.
- The points that remain are the vertices of the convex hull.

Turning Left (reminder)

Problem

Given \( p, q, r \) in the plane, if we walk from \( p \rightarrow q \rightarrow r \), do we make a left, a right, or no turn at \( q \)?

Solution

\[
\begin{align*}
(q - p) \times (r - p) = 0: & \text{ collinear segments — no turn.} \\
(q - p) \times (r - p) < 0: & \text{ right turn at } q. \\
(q - p) \times (r - p) > 0: & \text{ left turn at } q.
\end{align*}
\]

Implementation

Algorithm Graham-Scan(\( Q \))
1. Let \( p_0 \) be the point in \( Q \) with minimum \( y \) coordinate.
2. Sort \( Q \setminus \{p_0\} \) "lexicographically" in terms of (primary key) non-decreasing polar angle with respect to \( p_0 \) and (secondary key) distance from \( p_0 \). For angles with more than one point, delete all corresponding points except the one farthest from \( p_0 \).
3. if \( m \leq 2 \) then return \( \langle p_0, \ldots, p_m \rangle \)
4. else {
5. Initialise stack \( S \)
6. \( S.\text{push}(p_0) \)
7. \( S.\text{push}(p_1) \)
8. \( S.\text{push}(p_2) \)
9. for \( i \leftarrow 3 \) to \( m \) do
10. while the angle formed by the topmost two elements of \( S \) and \( p_i \) does not make a left turn do
11. \( S.\text{pop} \)
12. \( S.\text{push}(p_i) \)
13. return \( S \)
14. }

Example (BOARD)
Analysis of Running time

Let \( n = |Q| \), then \( m \leq n \).

- Lines 3–8, 13 require time \( \Theta(1) \).
- Line 1 requires time \( \Theta(n) \) in the worst case.
- Line 2 requires time \( \Theta(n \lg n) \).
- The outside (for) loop in lines 9–12 is iterated \( m - 2 \) times. Thus, disregarding the time needed by the inner while loop, the loop requires time \( \Theta(m) = O(n) \).
- The inner loop in lines 10–11 is executed at most once for each element, because every element enters the stack at most once and thus can only be popped once. Thus overall the inner loop requires time \( O(n) \).

Thus the overall worst-case running time is

\[ \Theta(n \lg n). \]

Proof of Correctness

(I) First we consider the effect of executing lines 1 and 2 to get the (possibly smaller) set of points \( P = p_0, p_1, \ldots, p_m \).

CLAIM (I): The convex hull of \( Q \) is equal to the convex hull of \( P \).

Proof of CLAIM (I): We only discard a point \( q \in Q \) if it has the same polar angle \( \text{wrt} p_0 \) as some point \( p_i \in P \), \text{AND} \( q \) is closer to \( p_0 \) than this \( p_i \). When \( q \) satisfies these 2 conditions, then \( q \) lies on \( \overline{p_0p_i} \). The convex hull of \( P \) by definition must contain \( \overline{p_0p_i} \) for every \( p_i \), so the convex hull of \( P \) must contain \( q \). Applying this inductively (on the entire set of points removed) we find that the convex hull of \( P \) equals that of \( Q \).

(II) Next we must prove that lines 3-14 compute the convex hull of \( p_0, p_1, \ldots, p_m \).

If \( m \leq 2 \) then the alg returns all \( m + 1 \) (1, 2, or 3) points (line 3). Correct.

Else \( m > 2 \) and the algorithm executes lines 5.-13.

For any \( 2 \leq i \leq m \), define \( C_i \) to be the convex hull of \( p_0, \ldots, p_i \).

After executing lines 5.-8., the points on stack \( S \) are the vertices of \( C_2 \) (clockwise).

We now prove that this situation holds for \( C_i \) after we execute the for loop with \( i \).

Proof of Correctness (\( m > 2 \)) cont’d

CLAIM (II): Let \( i \) be such that \( 2 \leq i \leq m \). Then after the \( i \)'-execution of the for loop (lines 9-12), the points on \( S \) are the vertices of \( C_i \) in clockwise order.

Proof of CLAIM (II): Our proof is by induction.

Base case (\( i=2 \)): In this case there is no \( i \)-iteration of the loop. However, the stack holds \( p_0, p_1, p_2 \) (lines 6.-8.), which form the convex hull of \{\( p_0, p_1, p_2 \)\}.

Induction hypothesis (IH): Assume CLAIM (II) holds for some \( i, 2 \leq i < m \).

Induction step: We will show CLAIM (II) also holds for \( i + 1 \).

- Since the polar angle of \( p_{i+1} \) is strictly greater than the polar angle of \( p_i \), therefore \( p_0p_ip_{i+1} \) forms a triangle that is not contained in \( C_i \).

Proof of Correctness (\( m > 2 \)) cont’d

CLAIM (III): Note \( p_{i+1} \) is NOT contained in \( C_i \) and thus is definitely a vertex of \( C_{i+1} \).

Proof of CLAIM (III): By (IH) any \( q \) “popped” so far is in the convex hull formed by the points currently on stack \( S \). \( \Rightarrow \ldots \) the convex hull \( C_{i+1} \) is contained in the convex hull of \( p_{i+1} \) and the points on \( S \).

- Left: First suppose the “next-to-top” point \( p \) on \( S \), followed by the “top” point \( p_i \), followed by \( p_{i+1} \) creates a “left turn”:

\[
\begin{array}{c}
\text{Then the triangle } p_0p_ip_{i+1} \text{ does NOT contain all of triangle } \overline{p_0p_ip_{i+1}p} \\
\Rightarrow p_i \text{ must be on the Convex Hull } C_{i+1}. \\
\Rightarrow \text{Using convexity of the points on } S, p_0 \rightarrow \overline{p} \rightarrow p_{i+1} \text{ is a left turn for all points } \overline{p} \text{ on } S \\
\Rightarrow \text{all such } \overline{p} \text{ must be on the Convex Hull } C_{i+1}. \\
\Rightarrow \text{hence the decision to “push” } p_{i+1} \text{ and leave all items of } S \text{ there, correctly constructs } C_{i+1}. \Rightarrow \text{CLAIM (II) Left proven.}
\end{array}
\]
Proof of Correctness \((m > 2)\) cont’d

- **Right:** Otherwise suppose the “next-to-top” point \(p\) on \(S\), followed by the “top” point \(p_i\), followed by \(p_{i+1}\), creates a “right turn”:

  ![Diagram of right turn]

  Then the triangle \(p_0pp_{i+1}\) does contain all of triangle \(p_0p_ip_{i+1}\).
  \(\Rightarrow\) \(C_{i+1}\) does not need to include the point \(p_i\).
  \(\Rightarrow\) decision to “pop” \(p_i\) (top item on \(S\)) on line 11 is correct.

We can apply this iteratively by considering the “turn direction” of the top two items on the stack, \(p^*, p\) say (taking the roles of \(p, p_i\)), followed by \(p_{i+1}\), “popping” until there is a left turn.

Once we find a left turn slide 12 applies, and we push \(p_{i+1}\) onto \(S\) on line 12, to complete \(C_{i+1}\). \(\Rightarrow\) CLAIM (ii) right proven.

ADS: lect 17 – slide 13 – Friday, 18th Nov, 2014

Wrapping up...

- We have proven the inductive step for CLAIM (ii).
- Hence CLAIM (ii) holds after the consideration of every point \(p_3, \ldots, p_m\).
- and in particular for \(i = m\):
- \(\Rightarrow\) after the \(m\)-execution (the final execution) of the for, the points on the stack \(S\) are the vertices of \(C_m\) in clockwise order.

The vertices \(C_m\) are the vertices of the original set of points \(Q\) (by CLAIM (i)).

Hence Graham’s scan computes the Convex Hull of its input correctly.

ADS: lect 17 – slide 14 – Friday, 18th Nov, 2014

Optimality

- The best-known algorithm for finding the convex hull has a running time of \(O(n \lg h)\), where \(h\) is the number of vertices of the convex hull.

- It can be shown (based on fairly natural assumptions) that every algorithm for finding the convex hull has a worst-case running time of \(\Omega(n \lg n)\).

The proof of this lower bound is due to the fact that we can implement real-number sorting using Convex Hull.

ADS: lect 17 – slide 15 – Friday, 18th Nov, 2014

Reading Assignment

- Section 33.3 of [CLRS].

Problems

1. Exercises 33.3-3 and 33.3-5 of [CLRS].
2. Show how to sort a collection of \(n\) points by polar angle (wrt some lowest point \(p_0\)) in \(O(n \lg(n))\) time, without using division or trigonometry.
3. Prove that the problem of finding the Convex Hull of \(n\) points has a lower bound of \(\Omega(n \lg n)\). For this, think about using a reduction from sorting to Convex Hull (that is, think about how to use a Convex Hull algorithm to sort a list of numbers).

ADS: lect 17 – slide 16 – Friday, 18th Nov, 2014