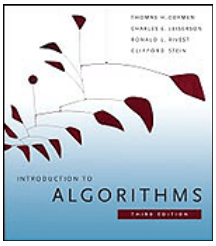

Chapter 33 - Computational Geometry
Introduction to Algorithms, Third Edition
by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein
The MIT Press © 2009 *Citation*

Previous Next

33.3 Finding the Convex Hull

The ***convex hull*** of a set Q of points, denoted by CH(Q), is the smallest convex polygon p for which each point in Q is either on the boundary of p or in its interior. (See Exercise 33.1-5 for a precise definition of a convex polygon.) We implicitly assume that all points in the set Q are unique and that Q contains at least three points which are not colinear. Intuitively, we can think of each point in Q as being a nail sticking out from a board. The convex hull is then the shape formed by a tight rubber band that surrounds all the nails. Figure 33.6 shows a set of points and its convex hull.

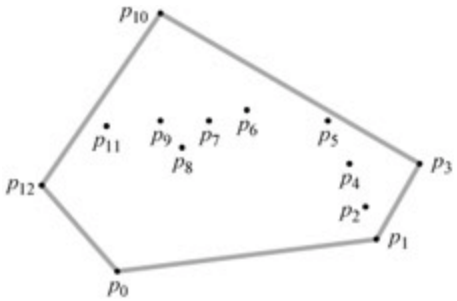

Figure 33.6: A set of points $Q = \{p_0, p_1, \ldots, p_2\}$ with its convex hull CH(Q) in gray.

In this section, we shall present two algorithms that compute the convex hull of a set of n points. Both algorithms output the vertices of the convex hull in counterclockwise order. The first, known as Graham's scan, runs in $O(n \lg n)$ time. The second, called Jarvis's march, runs in $O(nh)$ time, where h is the number of vertices of the convex hull. As Figure 33.6 illustrates, every vertex of CH(Q) is a point in Q. Both algorithms exploit this property, deciding which vertices in Q to keep as vertices of the convex hull and which vertices in Q to reject.

We can compute convex hulls in $O(n \lg n)$ time by any one of several methods. Both Graham's scan and Jarvis's march use a technique called "rotational sweep," processing vertices in the order of the polar angles they form with a reference vertex. Other methods include the following:

- In the ***incremental method***, we first sort the points from left to right, yielding a sequence $\langle p_1, p_2, \ldots, p_n \rangle$. At the ith stage, we update the convex hull of the i - 1 leftmost points, CH($\{p_1, p_2, \ldots, p_{i-1}\}$), according to the ith point from the left, thus forming CH($\{p_1, p_2, \ldots, p_i\}$). Exercise 33.3-6 asks you how to implement this method to take a total of $O(n \lg n)$ time.
- In the ***divide-and-conquer method***, we divide the set of n points in $\Theta(n)$ time into two subsets, one containing the leftmost $\lceil n/2 \rceil$ points and one containing the rightmost $\lceil n/2 \rceil$ points, recursively compute the convex hulls of the subsets, and then, by means of a clever method, combine the hulls in $O(n)$ time. The running time is described by the familiar recurrence $T(n) = 2T(n/2) + O(n)$, and so the divide-and-conquer method runs in $O(n \lg n)$ time.
- The ***prune-and-search method*** is similar to the worst-case linear-time median algorithm of Section 9.3. With this method, we find the upper portion (or "upper chain") of the convex hull by repeatedly throwing out a constant fraction of the remaining points until only the upper chain of the convex hull remains. We then do the same for the lower chain. This method is asymptotically the fastest: if the convex hull contains h vertices, it runs in only $O(n \lg h)$ time.

Computing the convex hull of a set of points is an interesting problem in its own right. Moreover, algorithms for some other computational-geometry problems start by computing a convex hull. Consider, for example, the two-dimensional ***farthest-pair problem***: we are given a set of n points in the plane and wish to find the two points whose distance from

each other is maximum. As Exercise 33.3-3 asks you to prove, these two points must be vertices of the convex hull. Although we won't prove it here, we can find the farthest pair of vertices of an n-vertex convex polygon in $O(n)$ time. Thus, by computing the convex hull of the n input points in $O(n \lg n)$ time and then finding the farthest pair of the resulting convex-polygon vertices, we can find the farthest pair of points in any set of n points in $O(n \lg n)$ time.

Graham's Scan

Graham's scan solves the convex-hull problem by maintaining a stack S of candidate points. It pushes each point of the input set Q onto the stack one time, and it eventually pops from the stack each point that is not a vertex of CH(Q). When the algorithm terminates, stack S contains exactly the vertices of CH(Q), in counterclockwise order of their appearance on the boundary.

The procedure GRAHAM-SCAN takes as input a set Q of points, where $|Q| \geq 3$. It calls the functions TOP(S), which returns the point on top of stack S without changing S, and NEXT-TO-TOP(S), which returns the point one entry below the top of stack S without changing S. As we shall prove in a moment, the stack S returned by GRAHAM-SCAN contains, from bottom to top, exactly the vertices of CH(Q) in counterclockwise order.

```
GRAHAM-SCAN(Q)
 1 let p_0 be the point in Q with the minimum y-coordinate,
       or the leftmost such point in case of a tie
 2 let ⟨p_1, p_2, …, p_m⟩ be the remaining points in Q,
       sorted by polar angle in counterclockwise order around p_0
       (if more than one point has the same angle, remove all but
       the one that is farthest from p_0)
 3 if m < 2
 4     return "convex hull is empty"
 5 else let S be an empty stack
 6     PUSH(p_0, S)
 7     PUSH(p_1, S)
 8     PUSH(p_2, S)
 9     for i = 3 to m
10         while the angle formed by points NEXT-TO-TOP(S), TOP(S),
               and p_i makes a nonleft turn
11             POP(S)
12         PUSH(p_i, S)
13     return S
```

Figure 33.7 illustrates the progress of GRAHAM-SCAN. Line 1 chooses point p_0 as the point with the lowest y-coordinate, picking the leftmost such point in case of a tie. Since there is no point in Q that is below p_0 and any other points with the same y-coordinate are to its right, p_0 must be a vertex of CH(Q). Line 2 sorts the remaining points of Q by polar angle relative to p_0, using the same method—comparing cross products—as in Exercise 33.1-3. If two or more points have the same polar angle relative to p_0, all but the farthest such point are convex combinations of p_0 and the farthest point, and so we remove them entirely from consideration. We let m denote the number of points other than p_0 that remain. The polar angle, measured in radians, of each point in Q relative to p_0 is in the half-open interval $[0, \pi]$. Since the points are sorted according to polar angles, they are sorted in counterclockwise order relative to p_0. We designate this sorted sequence of points by $\langle p_1, p_2, \ldots, p_m \rangle$. Note that points p_1 and p_m are vertices of CH(Q) (see Exercise 33.3-1). Figure 33.7(a) shows the points of Figure 33.6 sequentially numbered in order of increasing polar angle relative to p_0.