JOURNAL OF ALGORITHMS 4, 324–331 (1983)

# Finding the Convex Hull of a Simple Polygon

RONALD L. GRAHAM

*Bell Laboratories, Murray Hill, New Jersey 07974*

AND

F. FRANCES YAO

*Xerox Palo Alto Research Center, Palo Alto, California 94304*

Received October 31, 1981; revised June 18, 1982

It is well known that the convex hull of a set of $n$ points in the plane can be found by an algorithm having worst-case complexity $O(n \log n)$. A short linear-time algorithm for finding the convex hull when the points form the (ordered) vertices of a simple (i.e., non-self-intersecting) polygon is given.

## 1. INTRODUCTION

The problem of finding the *convex hull* of a planar set of points $P$, that is, finding the smallest convex region enclosing $P$, arises frequently in computer graphics. For example, to fit $P$ into a square or a circle, it is necessary and sufficient that $H(P)$, the convex hull of $P$, fits; and since it is usually the case that $H(P)$ has many fewer points than $P$ has, it is a simpler object to manipulate. It is also the case that many fast graphics algorithms on polygons require that the input polygon be convex, thus making it a useful preprocessing step sometimes to first transform a general polygon into its convex hull. A number of algorithms exist for finding the convex hull of a set of points (e.g., [1, 2, 6]), with worst-case complexity $O(n \log n)$ for $|P| = n$. It is also known that $\Omega(n \log n)$ is a lower bound just for determining $H(P)$—that is, not necessarily rendering $H(P)$ in, say, clockwise order [7]. This lower bound is proved for a decision tree model with quadratic tests, which accommodates all the known convex hull algorithms.

An interesting case of the convex hull problem that occurs frequently in practice is when the points of $P$ form the vertices of a *simple polygon* (i.e., a polygon without self-intersections). Several authors have tried to find a fast algorithm for this problem. Sklansky [5] proposed an $O(n)$ algorithm, which Bykat [1] later showed does not always work. It has also been noted that the algorithm that Shamos [4] suggested can sometimes fail. McCallum and Avis [3] published an $O(n)$ algorithm which, being quite complicated and utilizing two stacks, entails rather intricate case analysis for the proof of its validity. In this paper we give a simple linear-time algorithm for this problem and a proof of its validity.

## 2. Preliminaries

Let $P$ be a simple polygon in the plane with $n$ vertices. Without loss of generality, assume that $P = \langle v_1, v_2, \ldots, v_n \rangle$ is given by a linked list of its vertices as they are encountered in a clockwise traversal of the boundary, where each vertex $v_i$ is represented by its $X$ and $Y$ coordinates. (The orientation of $P$ can be easily tested and, if necessary, reversed.) We will assume henceforth that all polygons are clockwise oriented. The convex polygon whose vertices are the set of extreme points of $P$ is called the convex hull of $P$, denoted by $H(P)$. For a polygon $P$, we will use $P[v_i, v_j]$ to denote the path in $P$ from $v_i$ to $v_j$ (following the orientation of $P$). For two paths $p = P[v_i, v_j]$ and $q = P[v_j, v_k]$, their concatenation $p[v_i, v_k]$ is written as $p \circ q$. Given two points $x$ and $y$, $L[x, y]$ refers to the (directed) line segment from $x$ to $y$. We say that $z$ lies *to the right* (*left*) of $L[x, y]$ if $z$ is in the right (left) half-plane defined by the extension of $L[x, y]$. More generally, for a simple path $p = L[x_1, x_2] \circ L[x_2, x_3] \circ \cdots \circ L[x_{k-1}, x_k]$, we say that $z$ lies *to the right* (*left*) of $p$ if $z$ is in the region to the right (left) of the extended path $p$ (where $L[x_1, x_2]$ and $L[x_{k-1}, x_k]$ are stretched to infinity in the appropriate directions). This is illustrated in Fig. 1. For an ordered triple of three points $(x, y, z)$, we write $(x, y, z) = 1, 0$, or $-1$, depending on whether $z$ is to the right of, collinear with, or to the left of $L[x, y]$, respectively.

Let the vertices of the convex hull $H(P)$ be $\langle r_1, \ldots, r_h \rangle$. It is easy to see that $\langle r_1, \ldots, r_h \rangle$ must satisfy the following conditions (see Fig. 2).

(A1) $\langle r_1, \ldots, r_h \rangle$ forms a convex polygon.

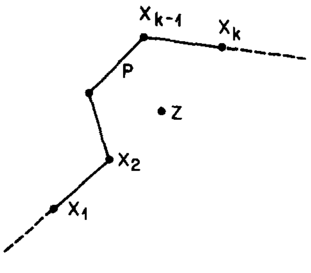


FIG. 1. $z$ lies to the right of path $p$.

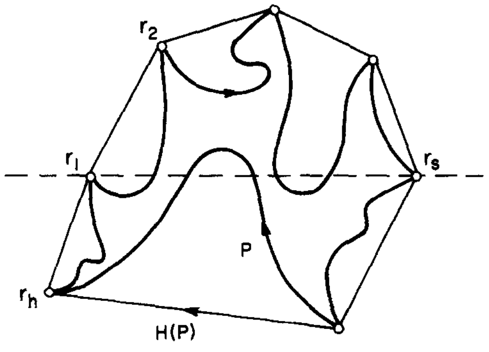


FIG. 2. A simple polygon $P$ and its convex hull $H(P)$.

(A2) Each vertex of $P[r_{i-1}, r_i]$ lies either on or to the right of $L[r_{i-1}, r_i]$ for $2 \leqslant i \leqslant h + 1$.

We call any path $P[r_{i-1}, r_i]$ that satisfies (A2) a *pocket* of $P$, and call $L[r_{i-1}, r_i]$ the *top* of the pocket. The *closure* of the pocket $P[r_{i-1}, r_i]$, denoted by $U[r_{i-1}, r_i]$, is the finite, closed area bounded by the polygon $L[r_{i-1}, r_i] \circ P[r_i, r_{i-1}]$. Lemma 2.1 below implies that (A1) and (A2) are also sufficient conditions for $\langle r_1, \ldots, r_h \rangle$ to be the convex hull. The task of finding $H(P)$ can thus be described as that of identifying a sequence of pockets $P[r_1, r_2], P[r_1, r_3], \ldots, P[r_h, r_1]$ such that the concatenation of their tops $L[r_1, r_2] \circ L[r_2, r_3] \circ \cdots \circ L[r_h, r_1]$ forms a convex polygon.

LEMMA 2.1. *Let $P[r_{i-1}, r_i]$ be a pocket of $P$. No vertices of $P$ that lie in $U[r_{i-1}, r_i]$, with the possible exception of $r_{i-1}$ and $r_i$, can belong to $H(P)$.*

*Proof.* Let $x$ be a vertex of $P$, other than $r_{i-1}$ and $r_i$, that lies in $U[r_{i-1}, r_i]$. If $x$ lies either in the interior of $U[r_{i-1}, r_i]$, or on $L[r_{i-1}, r_i]$, then $x$ obviously cannot be an extreme point. Assume then that $x$ is on the path $P[r_{i-1}, r_i]$. By the Jordan Curve Theorem, the line connecting $r_i$ and $x$ must intersect the boundary of $P$ at some point $y \neq x$ such that $x$ is interior to the line segment $L[r_i, y]$. Therefore $x$ is not an extreme point of $P$. □

Let $P[r_{i-1}, r_i]$ be a pocket of $P$. We define the *emergence vertex* of $P[r_{i-1}, r_i]$ to be the first successor $x^*$ of $r_i$ in $P$ that lies outside of $U[r_{i-1}, r_i]$. The following lemma shows that every pocket has an emergence vertex and gives a way for computing it.

LEMMA 2.2. *Let $P[r_{i-1}, r_i]$ be a pocket of $P$, and let $y$ and $x$ be the vertices that appear immediately before and after $r_i$, respectively, in $P$. Then, the emergence vertex of $P[r_{i-1}, r_i]$ is $x$ unless $(r_{i-1}, r_i, x) \geqslant 0$ and $(y, r_i, x) < 0$; in the latter case the emergence vertex is the first successor $x^*$ of $x$ satisfying $(r_{i-1}, r_i, x^*) < 0$.*

*Proof.* The first case follows from the definition of a pocket and the fact that $P$ is simple. To prove the second case, note that some successor of $x$ must lie outside of $U[r_{i-1}, r_i]$, because otherwise $P$ would have a bounded exterior (contained in $U[r_{i-1}, r_i]$). The first such successor $x^*$ must be connected to the vertex immediately preceding it by an edge that crosses the top of the pocket. In other words, $x^*$ is the first successor of $x$ satisfying $(r_{i-1}, r_i, x^*) < 0$. □

Suppose two vertices of $H(P)$ are given—say $v_1$ and $v_m$. Then the chord $L[v_1, v_m]$ divides $H(P)$ into two convex polygons $H(P)[v_1, v_m]$ and $H(P)[v_m, v_1]$, consisting of those extreme vertices of $P$ that lie, respectively, to the left and to the right of $L[v_1, v_m]$ (in addition to vertices $v_1$ and $v_m$). We will call $H(P)[v_1, v_m]$ the *left hull* of $P[v_1, v_m]$. (Thus $H(P)[v_m, v_1]$ is the left hull of $P[v_m, v_1]$.) The following characterization of the left hull $H(P)[v_1, v_m]$ is an immediate consequence of (A1), (A2), and Lemma 2.1. Note that the characterization is dependent only on the path $P[v_1, v_m]$.

LEMMA 2.3. *Let $\langle r_1, \ldots, r_s \rangle$ be a subsequence of $P[v_1, v_m]$ where $v_1$ and $v_m$ are two extreme points of $P$. Then $H(P)[v_1, v_m] = \langle r_1, \ldots, r_s \rangle$ if and only if*

(B1) $\langle r_1, \ldots, r_s \rangle$ forms a convex polygon with $r_1 = v_1$ and $r_s = v_m$.

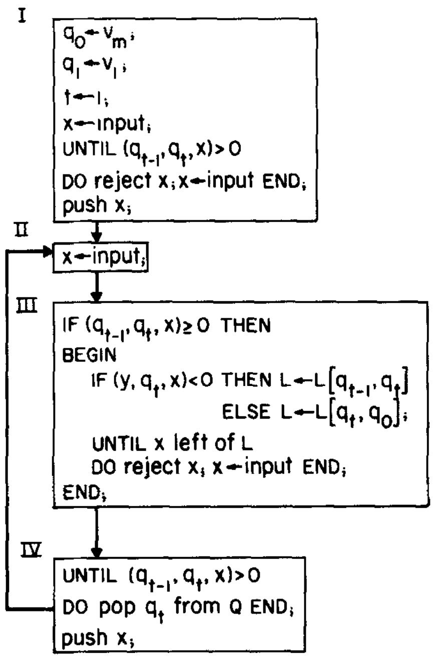
(B2) $P[r_{i-1}, r_i]$ forms a pocket for $2 \leqslant i \leqslant s$.

The problem of finding $H(P)$ can thus be solved by finding the left hulls of $P[v_1, v_m]$ and $P[v_m, v_1]$ and concatenating them. We will describe an algorithm for finding the left hull of $P[v_1, v_m]$ in the next section. In practice, a convenient choice for $v_1$ and $v_m$ could be, say, two points of $P$ with the minimum and the maximum $X$-coordinate, respectively. Without loss of generality, we now assume that in $P = \langle v_1, v_2, \ldots, v_m, v_{m+1}, \ldots, v_n \rangle$, vertices $v_1$ and $v_m$ have been so chosen.

## 3. THE ALGORITHM

Algorithm LeftHull is given by the diagram in Fig. 3. The algorithm considers the vertices of $P[v_1, v_m]$ in the order $\langle v_m, v_1, v_2, \ldots, v_{m-1} \rangle$. The main data structure used is a stack $Q = \langle q_0, q_1, \ldots, q_t \rangle$, where $q_0$ denotes the bottom of the stack, and variable $t$ points to the stack top. Variable $x$ refers to the input vertex under consideration, and $y$ refers to the input vertex immediately preceding $q_t$. "Pushing $x$" means executing $[t \leftarrow t + 1;$ $q_t \leftarrow x$; update $y]$, "popping $q_t$" means setting $[t \leftarrow t - 1]$, and no code is executed for "rejecting $x$." The algorithm halts when the input is exhausted.

Intuitively, the algorithm works as follows. Box I performs the initialization, and picks the first vertex to the right of $L[q_0, q_1]$ to be $q_2$. Box III



```
I
    $q_0 \leftarrow v_m$;
    $q_1 \leftarrow v_1$;
    $t \leftarrow 1$;
    $x \leftarrow$ input;
    UNTIL $(q_{t-1}, q_t, x) > 0$
    DO reject $x$; $x \leftarrow$ input END;
    push $x$;
II
    $x \leftarrow$ input;
III
    IF $(q_{t-1}, q_t, x) \geq 0$ THEN
    BEGIN
        IF $(y, q_t, x) < 0$ THEN $L \leftarrow L[q_{t-1}, q_t]$
                            ELSE $L \leftarrow L[q_t, q_0]$;
        UNTIL $x$ left of $L$
        DO reject $x$; $x \leftarrow$ input END;
    END;
IV
    UNTIL $(q_{t-1}, q_t, x) > 0$
    DO pop $q_t$ from Q END;
    push $x$;
```

FIG. 3. Algorithm LeftHull. Input: $P[v_1, v_m] = \langle v_m, v_1, v_2, \ldots, v_{m-1} \rangle$. Output: $H(P)[v_1, v_m] = \langle q_0, \ldots, q_t \rangle$.

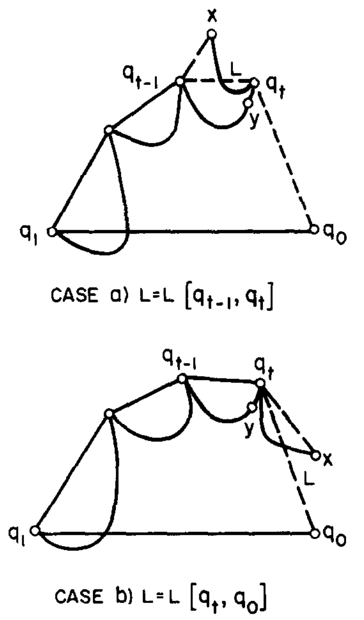


FIG. 4. The result of executing Box III.

finds the first vertex $x$ that emerges from the interior of the present convex polygon $Q = \langle q_0, \ldots, q_t \rangle$; the two possible cases are illustrated in Fig. 4. Finally Box IV updates $Q$ and restores its convexity.

## 4. Correctness of Algorithm LeftHull

We will prove the following theorem in this section.

Main Theorem. *Algorithm LeftHull finds the left hull of* $P[v_1, v_m]$ *correctly. The convex hull of a simple polygon can therefore be found in linear space and linear time.*

We prove the correctness of Algorithm LeftHull by establishing two lemmas. Lemma 4.1 and Lemma 4.2 together show that the output $Q = \langle q_0, \ldots, q_t \rangle$ of the algorithm satisfies the characterization (B1) and (B2) for the left hull $H(P)[v_1, v_m]$.

Lemma 4.1. *The following induction hypotheses are true each time Box* II *is entered* (*Fig.* 5):

(H1) $Q = \langle q_0, \ldots, q_t \rangle$ forms a convex polygon where $q_0 = v_m$, $q_1 = v_1$ and $t \geqslant 2$.

(H2) $P[q_{i-1}, q_i]$ forms a pocket for $2 \leqslant i \leqslant t$.

We first observe the following consequence of induction hypotheses (H1) and (H2); the proof is straightforward and will be omitted. We adopt the convention that $q_{t+1}$ means $q_0$.

*Fact. Induction hypotheses* (H1) *and* (H2) *together imply that* $P[q_i, q_t]$ *lies to the right of* $L[q_{i-1}, q_i] \circ L[q_i, q_{i+1}]$ *for any* $2 \leqslant i \leqslant t$.

*Proof of Lemma* 4.1. When Box II is entered for the first time, we have $Q = \langle q_0, q_1, q_2 \rangle$ where $q_2$ is the first vertex to the right of $L[q_0, q_1]$. Thus (H1) and (H2) are satisfied initially. We will establish the inductive step first

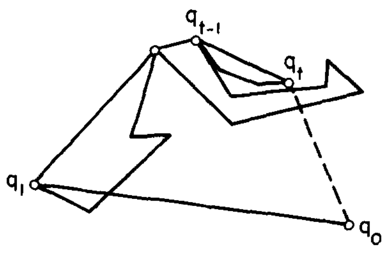


Fig. 5. The induction hypotheses.

for (H1) and then for (H2). When Box IV is entered, vertex $x$ is either to the left of $L[q_{t-1}, q_t]$ or to the left of $L[q_t, q_0]$. In either case, $x$ is external to the convex polygon $Q = \langle q_0, \ldots, q_t \rangle$, and it is easy to check that the execution of Box IV results in a new convex polygon $Q$ satisfying (H1). We next consider induction hypothesis (H2). Assume that the stack pointer $t$ has value $j$ when Box IV is entered, and that $q_j, q_{j-1}, \ldots, q_{i+1}$ (where $1 \leqslant i \leqslant j$) are popped as a result of executing Box IV. Thus $x$ and $q_i$ are the two topmost elements after $Q$ is updated, and we need only show that $P[q_i, x]$ is a pocket. Let $u$ be the input vertex just before $x$. By the Fact stated above, $P[q_i, q_j]$ lies to the right of the path $L[q_{i-1}, q_i] \circ L[q_i, q_{i+1}]$. It then follows from Lemma 2.2 and the way Box III works that $P[q_i, u]$ must lie to the right of the path $L[q_{i-1}, q_i] \circ L[q_i, q_{i+1}]$, and hence also to the right of the path $L[q_{i-1}, q_i] \circ L[q_i, x]$. In particular, this implies that $P[q_i, u]$ lies to the right of $L[q_i, x]$, proving that $P[q_i, x]$ is a pocket. This establishes induction hypothesis (H2). □

LEMMA 4.2. *When the algorithm terminates,* (H2) *is true for* $i = t + 1$; *that is,* $P[q_t, q_0]$ *also forms a pocket.*

*Proof.* If termination occurs in Box II, then this is trivially true since $P[q_t, q_0] = L[q_t, q_0]$. The other possibilities are to terminate while executing the DO loop in either Box I or Box III, where the latter case can happen only if, because of Lemma 2.2, one has entered Box III with $(y, r_i, x) > 0$. For either one of these possibilities, the property "$x$ lies to the right of $L[q_t, q_0]$" will be true for all the input vertices $x$ seen after $q_t$. This shows that $P[q_t, q_0]$ is a pocket. □

Algorithm LeftHull uses linear space and linear time, since each input point can be pushed onto or popped from the stack at most once, if it is not rejected outright. This completes the proof of our main theorem. □

## 5. CONCLUSIONS

We present a simple linear-time algorithm for finding the convex hull of a simple polygon. Note that our algorithm can actually be applied to nonsimple polygons $P$ as follows: First, at each point $p$ of intersection of two (or more) edges of $P$, place a new vertex $v(p)$. Viewed as a graph $G(P)$, all vertices (old and new) have even degrees, so that the graph $G(P)$ is Eulerian. Choose a fixed Eulerian circuit in $G(P)$. It is not difficult to see that each new vertex $v(p)$ of degree $2d(p)$ may now be split into $d(p)$ slightly perturbed vertices $v_i(P)$, $1 \leqslant i \leqslant d(P)$, of degree 2 so that each $v_i(P)$ is in the interior of $H(P)$ and the resulting polygon $P^*$ (= Eulerian circuit) is simple. Therefore $H(P^*) = H(P)$ and our algorithm can be used to compute $H(P^*)$. Observe, however, that if $P$ has $n$ vertices, $P^*$ can have $O(n^2)$ vertices.

In the case that $P$ is a simple path with $n$ vertices, we can join the first and last points of $P$ creating a new edge to form a (possibly nonsimple) polygon $\bar{P}$. By applying the preceding transformation we can then find a simple polygon $\bar{P}^*$ on at most $3n$ vertices with $H(\bar{P}^*) = H(\bar{P}) = H(P)$ in time $O(n)$.

## ACKNOWLEDGMENTS

We are grateful to Joseph O'Rourke and Godfried Toussaint whose careful reading of an earlier draft uncovered an error in our paper. We should also point out that our algorithm is similar in spirit to an unpublished algorithm independently discovered by D. T. Lee.

## REFERENCES

1. A. BYKAT, Convex hull of a finite set of points in two dimensions, *Inform. Process. Lett.* **7** (1978), 296–298.
2. R. L. GRAHAM, An efficient algorithm for determining the convex hull of a planar set, *Inform. Process. Lett.* **1** (1972), 132–133.
3. D. MCCALLUM and D. AVIS, A linear algorithm for finding the convex hull of a simple polygon, *Inform. Process. Lett.* **9** (1979), 201–206.
4. M. SHAMOS, "Problems in Computational Geometry," Doctoral dissertation, Computer Science Department, Yale University, 1978.
5. J. SKLANSKY, Measuring concavity on a rectangular mosaic, *IEEE Trans. Comput.*, **21** (1972), 1355–1364.
6. G. TOUSSAINT, S. AKL, and L. DEVROYE, "Efficient Convex Hull Algorithms for Points in Two and More Dimensions," Technical Report No. 78.5, McGill University, 1978.
7. A. YAO, A lower bound for finding convex hulls, *J. Assoc. Comput. Mach.*, **28** (1981), 780–787.