

A NOTE ON GRAHAM'S CONVEX HULL ALGORITHM

David GRIES

Computer Science Department, Cornell University, 405 Upson Hall, Ithaca, NY 14853-7501, U.S.A.

Ivan STOJMENOVIĆ

Mathematics Institute, University of Novi Sad, Novi Sad, Yugoslavia

Communicated by David Gries

Received 4 April 1986

Revised 6 November 1986

Keywords: Convex hull, algorithm, computational geometry, programming methodology

1. Introduction

Graham [6] proposed an $O(n \log n)$ algorithm for determining the convex hull of a finite set of points in the plane. Variations of this method have been described in [1–4,9–15]. However, five of these references, though in refereed journals and texts, contain errors in this rather simple algorithm. Why should this be? The main culprit, we believe, is the method used for developing and describing the algorithm and its proof—typically directly in terms of circular doubly-linked lists. In this note, we present an alternative method, which, we believe, allows for a more rigorous but simple and understandable treatment. This continues work in [5,7,8] on formulating methods for dealing with linked lists and other similar data structures.

2. Graham's algorithm

The first step of Graham's algorithm for determining the convex hull of a set S of $n \geq 3$ points is to construct a sequence $P(0..n-1)$ of the points in polar coordinates ordered about an interior point in terms of increasing angle (see Fig. 1). This step takes $O(n \log n)$ time, because of the need to sort, while the rest of the algorithm—the

part we are interested in—takes $O(n)$ time. Assume the first step has been done.

In sequence P , call a point *reflex* if the interior angle made by it and its adjacent points is $\geq \pi$ (note that point P_0 is adjacent to P_{n-1}). For example, in Fig. 1, P_0 is the only reflex point.

A reflex point of P does not belong to the convex hull of S . Conversely, if no point of P is reflex, then P is the convex hull of S . Graham's algorithm examines the points of P in counterclockwise order and deletes those that are reflex; upon termination, only nonreflex points remain, so P is the convex hull of S . Determining when the counterclockwise examination of points can stop seems to be the major difficulty, because deleting a reflex point can change its neighbors from non-reflex to reflex. For example, deleting reflex point

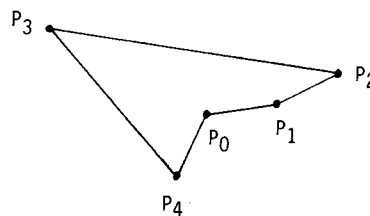


Fig. 1. The list of points S around the origin. $\sphericalangle P_4 P_0 P_1 > 180^\circ$, $\sphericalangle P_0 P_1 P_2 < 180^\circ$, and $\sphericalangle P_4 P_1 P_2 > 180^\circ$.

P_0 in Fig. 1 changes P_1 from nonreflex to reflex. Koplowitz and Jouppi [9] have the wrong stopping criterion; Preparata, Lee and Shamos [10,11] have a correct stopping criterion but code it incorrectly in their algorithms, which are written completely in terms of circular doubly-linked lists.

3. Our description of the algorithm

We first develop an algorithm that operates on a high-level structure called a *ring*, instead of the traditional circular doubly-linked list. We introduce some notation. When dealing with sequences of elements, capital letters denote sequences and lower-case letters denote elements of sequences. The symbol Φ denotes the empty sequence. Catenation of sequences and elements is denoted by juxtaposition, e.g., $X p Y$ denotes the sequence

consisting of the elements of X followed by element p followed by the elements of Y .

For X a sequence, the notation $[X]$ denotes a *ring* consisting of the elements of X . A ring is like a sequence, except that rotating the elements one (or more) position(s) in either direction does not change the ring, while it does change the sequence. We indicate this by giving the following (additional) axiom for rings:

For any element x and sequence X ,
 $[x X] = [X x]$.

Using the concept of a ring, we give algorithm (1) below for Graham's scan; invariant P of the loop is given in (0). The algorithm uses the construct "**with identifier satisfying predicate do statement**", which calls for executing the *statement* with *identifier* having a value that satisfies the *predicate*.

-
- (0) P : Ring $[X Y]$ contains (only) points of S , in order of increasing angle;
 All points of the convex hull of S are in $[X Y]$;
 The elements of X are not reflex.
- (1) $X, Y := \Phi$, "elements of S , in order of increasing angle";
do $Y \neq \Phi \rightarrow$
 with p, \bar{Y} **satisfying** $Y = p \bar{Y}$ **do**
 if $\neg \text{reflex}.p \rightarrow X, Y := X p, \bar{Y}$
 \square $\text{reflex}.p \rightarrow$
 if $\bar{Y} \neq \Phi \wedge X \neq \Phi$
 \rightarrow **with** \bar{X}, r **satisfying** $X = \bar{X} r$ **do**
 $\{[X Y] = [\bar{X} r p \bar{Y}]; \text{delete } p; r \text{ may then be reflex}\}$
 $X, Y := \bar{X}, r \bar{Y}$
 \square $X = \Phi \rightarrow \{[X Y] = [p \bar{Y}]; \text{delete } p\}$
 $Y := \bar{Y}$
 \square $\bar{Y} = \Phi \rightarrow$ **with** q, \bar{X}, r **satisfying** $X = q \bar{X} r$ **do**
 $\{[X Y] = [q \bar{X} r p]; \text{delete } p; r, q \text{ may then be reflex}\}$
 $X, Y := \bar{X}, r q$
 fi
fi
od
-

Algorithm (1), written in terms of $[X Y]$, is easily seen to be correct: loop invariant P is initially true; its truth is maintained by each itera-

tion; since each iteration deletes a node or reduces the length of Y , termination is assured; and, upon termination, P together with $Y = \Phi$ implies that X

is the convex hull of set S. Some of the arguments concerning correctness rest on the fact that [X Y] contains at least three elements.

Note that there is no worrying over a good ‘stopping condition’, something that most of the papers have done. The stopping condition is simply $Y = \Phi$, since that together with P implies that X is the convex hull.

It is a fairly straightforward matter to translate this algorithm into one that uses a circular doubly-linked list. However, rather than translate this particular algorithm, we first look at a modification that reduces the execution time in some cases; this modification will then be translated into an algorithm that uses a circular doubly-linked list.

4. An improvement

Several modifications of Graham’s algorithm have been proposed, all having to do with the

following, which is discussed in terms of the notation of algorithm (1). If the first point in X is guaranteed to be on the convex hull, then it is never reflex and need never be placed in Y. Sklansky [12] used this idea, but his change was incorrect, as discussed in [14]. He amended it [13], but the amendment was again incorrect [15]. Preparata, Lee and Shamos [10,11] used the rightmost smallest-ordinate point as the first point in X, but their algorithm is incorrectly coded. Anderson [3] proposed using the bottom-most point as the first point in X. Andrew [4] and Akl and Toussaint [1,2] also made improvements.

Let us modify our algorithm to use a point on the convex hull as the first element of X. We strengthen the invariant with an additional conjunct:

P1: X is not empty and the first element of X is on the convex hull.

With this change, we write the algorithm as follows (#X denotes the length of X):

```
(2)  X := "any point on the convex hull";
      Y := "sequence consisting of the rest of S, in suitable order";
      {invariant: P ∧ P1}
      do Y ≠ Φ →
        with p,  $\bar{Y}$  satisfying  $Y = p \bar{Y}$  do
          if ¬ reflex.p → X, Y := X p,  $\bar{Y}$ 
          □ reflex.p →
            if #X = 1 → {delete p}
                       Y :=  $\bar{Y}$ 
            □ #X > 1 → {delete p; last point of X may be reflex}
                       with  $\bar{X}$ , r satisfying  $X = \bar{X} r$  do X, Y :=  $\bar{X}$ , r  $\bar{Y}$ 
          fi
        fi
      od
```

5. Translating the algorithm from ring to linked list

We now turn to an implementation of ring [X Y] as a circular doubly-linked list. Each element (indexed by i, say) of the list has fields succ.i and pred.i that contain the indices of the succes-

or and predecessor of i, respectively—as well as other fields that describe the position of the corresponding point in the plane, etc. (see Fig. 2). We assume that operation delete(h) deletes the element indexed by h from the list, changing all necessary pred and succ fields accordingly.

Suppose that S contains n points and that the convex hull contains m points. Algorithm (4) tests $n - 1$ points at least once for reflexivity and, for each of the $n - m$ points deleted, it may test its predecessor again. Hence, it tests at most $2n - m - 1$ points. The minimum number of points tested is $n - 1$.

References

- [1] S.G. Akl and G.T. Toussaint, A fast convex hull algorithm, *Inform. Process. Lett.* 7 (1978) 219–222.
- [2] S.G. Akl and G.T. Toussaint, Efficient convex hull algorithms for pattern recognition applications, *Proc. 4th Internat. Joint Conf. on Pattern Recognition*, Kyoto, Japan (1978) 483–487.
- [3] K.R. Anderson, A reevaluation of an efficient algorithm for determining the convex hull of a finite planar set, *Inform. Process. Lett.* 7 (1978) 53–55.
- [4] A.M. Andrew, Another efficient algorithm for convex hulls in two dimensions, *Inform. Process. Lett.* 9 (1979) 216–219.
- [5] W.H.J. Feijen, A.J.M. Van Gasteren and D. Gries, In-situ inversion of a cyclic permutation, *Inform. Process. Lett.* 24 (1) (1987) 11–14.
- [6] R.L. Graham, An efficient algorithm for determining the convex hull of a finite planar set, *Inform. Process. Lett.* 1 (1972) 132–133.
- [7] D. Gries and J.F. Prins, McLaren's masterpiece, *Tech. Rept. TR86-729*, Computer Science Dept., Cornell Univ., January 1986.
- [8] D. Gries and J.-Y. Xue, Generating a random cyclic permutation, *Tech. Rept. TR86-786*, Computer Science Dept., Cornell Univ., September 1986.
- [9] J. Koplowitz and D. Jouppi, A more efficient convex hull algorithm, *Inform. Process. Lett.* 7 (1978) 56–57.
- [10] D.T. Lee and F.P. Preparata, Computational geometry—A survey, *IEEE Trans. Comput.* C-33 (12) (1984) 1072–1101.
- [11] F.P. Preparata and M.I. Shamos, *Computational Geometry—An Introduction* (Springer, Berlin/New York, 1985).
- [12] J. Sklansky, Measuring concavity on a rectangular mosaic, *IEEE Trans. Comput.* C-21 (12) (1972) 1355–1364.
- [13] J. Sklansky, Finding the convex hull of a simple polygon, *Pattern Recognition* 15 (1) (1982) 79–83.
- [14] G.T. Toussaint and D. Avis, On a convex hull algorithm for polygons and its application to triangulation problems, *Pattern Recognition Lett.* 15 (1) (1982) 23–29.
- [15] G.T. Toussaint and H. El Gindy, A counterexample to an algorithm for computing monotone hulls of simple polygons, *Pattern Recognition Lett.* 1 (1983) 219–222.