Information Processing Letters 31 (1989) 17–20
North-Holland

12 April 1989

A LINEAR TIME ALGORITHM FOR FINDING ALL FARTHEST NEIGHBORS IN A CONVEX POLYGON

Alok AGGARWAL *

IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A.

Dina KRAVETS * *

Laboratory for Computer Science, MIT, Cambridge, MA 02139, U.S.A.

Communicated by E.C.R. Hehner
Received 18 August 1988
Revised 12 November 1988

Aggarwal et al. [A. Aggarwal, M.M. Klawe, S. Moran, P. Shor and R. Wilber, Geometric applications of a matrix-searching algorithm, *Algorithmica* **2** (1987) 209–233] showed how to compute in O(n) time *one* farthest neighbor for every vertex of a convex n-gon. In this note we extend this result to obtain a linear time algorithm for finding *all* farthest neighbors for every vertex of a convex polygon. Our algorithm yields a linear time solution to the symmetric all-farthest neighbors problem for simple polygons, thereby settling an open question raised by Toussaint in 1983 [G.T. Toussaint, The symmetric all-farthest neighbor problem, *Comp. Math. Appl.* **9** (6) (1983) 747–753.]

Keywords: Convex polygon, farthest neighbors

1. Introduction

A two-dimensional matrix $M = \{m_{i,j}\}$ is called *monotone* if the maximum value in the ith row lies below or to the right of the maximum value in the $(i-1)$st row. If a row has several maxima then we will take the leftmost one. A matrix M is called *totally monotone* if every 2×2 submatrix (i.e., every 2×2 minor) is monotone (See Fig. 1).

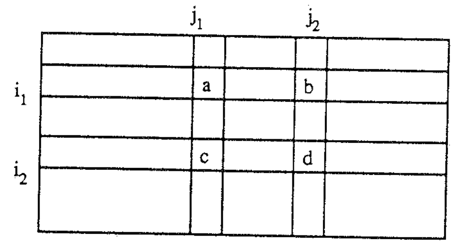

Fig. 1. Every 2×2 minor, given by a, b, c and d, of a totally monotone matrix is monotone, i.e. it is not possible that $a < b$ and $c > d$.

Although the question of finding the row maxima in a two-dimensional totally monotone matrix may seem rather odd at first glance, [1] and [2] show that a wide variety of problems can be reduced to one or more instances of this problem. For example, the totally monotone property arises in problems that deal with polygons whose vertices obey the quadrangle inequality. Suppose a convex polygon has n vertices numbered 0, 1,..., $n-1$.

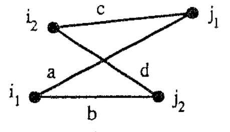

Fig. 2.

* This work was done while the author was visited MIT in the spring of 1988.

* * Research supported in part by Air Force Contract OSR-86-0076, DARPA Contract N00014-80-C-0622, and Army Contract DAAL-03-86-K-0171.

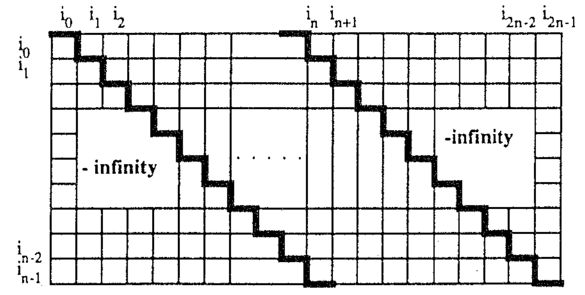

Fig. 3.

Let the distance between two vertices, i and j, be denoted by $d(i, j)$. If we take any 4 distinct vertices, i_1, i_2, j_1, and j_2, such that $0 \leqslant i_1 \leqslant i_2 \leqslant j_1 \leqslant j_2 \leqslant n-1$, and form a quadrangle $i_1 i_2 j_1 j_2$, then by the quadrangle inequality, the sum of the diagonals $d(i_1, j_1) + d(i_2, j_2)$ is *strictly greater* than the sum of the sides $d(i_1, j_2) + d(i_2, j_1)$ (see Fig. 2). We define a matrix $M = \{m_{i,j}\}$ corresponding to this convex polygon as follows (see Fig. 3): $\forall i$, $m_{i,j} = d(i, j \bmod n)$ for $i < j < i+n$, and $m_{i,j} = -\infty$ for all other j.

Claim 1. *Matrix M is a totally monotone matrix.*

Proof. Take any 2×2 minor (see Fig. 1). If none of the entries in the minor are $-\infty$ and $i_1 < i_2 < j_1 < j_2$, then it is not possible that $a < b$ and $c > d$ because of the quadrangle inequality. $i_1 < i_2$ and $j_1 < j_2$ have to hold simply from the way we have drawn the minor. If $i_2 \geqslant j_1$, then $c = -\infty$. Similarly, if any other inequality among $i_1 < i_2 < j_1 < j_2$ does not hold, then at least one of the entries in the minor is $-\infty$, which is the case we consider next.

If $b = -\infty$ or $c = -\infty$, then either $a \not< b$ or $c \not> d$. If $a = -\infty$, then by the way we have defined M, either $b = -\infty$ or $c = -\infty$ and the previous case applies. Similarly, if $d = -\infty$, then either $c = -\infty$ or $b = -\infty$. Thus, M is monotone. □

Aggarwal et al. [1] showed how to find the leftmost maximum in each row of an $n \times m$ matrix M in $O(n+m)$ time on a sequential RAM. Note that the algorithm of [1] does not explicitly create the entire matrix M (that would take $O(nm)$ time); rather, it only computes $O(n+m)$ entries of M. Thus, using [1]'s algorithm, we can find the *leftmost farthest neighbor* of every vertex on a convex n-gon in $O(n)$ time. In this note we show how to find *all farthest neighbors* of every vertex of a convex n-gon in $O(n)$ time. The note is organized as follows. In Section 2 we describe an algorithm to find all farthest neighbors of a convex polygon. In Section 3 we prove that the algorithm takes $O(n)$ time. In Section 4 we describe briefly how our algorithm can be used to find all symmetric farthest neighbors of a simple polygon, and in Section 5 we give some open problems.

2. Algorithm to find all farthest neighbors of a convex polygon

Let M be a totally monotone matrix corresponding to a convex polygon, i.e., $\forall i$, $m_{i,j} = d(i, j \bmod n)$ for $i < j < i+n$, and $m_{i,j} = -\infty$ for all other j.

Algorithm Compute-all-farthest-neighbors(M)

Step 1. We use the algorithm of [1] to find the leftmost maximum in each row of M. Let l_i be the column in which the leftmost maximum in row i appears.

Step 2. Consider a different totally monotone matrix M' such that $\forall i$, $m'_{ij} = d((n-1-i) \bmod n, (2n-1-j) \bmod n)$ for $i < j < i+n$ and $m'_{i,j} = -\infty$ for all other j. Note that the matrix M' is simply M rotated upside down, or equivalently, M flipped horizontally and vertically (see Fig. 4). Such a rotation preserves the totally monotone property: it is not possible to have $d < c$ and $b > a$ (which is the same

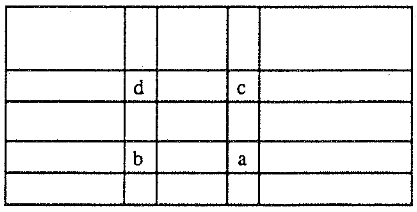

Fig. 4. The matrix from Fig. 1 flipped horizontally and vertically.

as saying that it is not possible to have $a < b$ and $c > d$). Again, we use the algorithm of [1] to find the leftmost maximum in each row of M'. But, the way we have defined M', the leftmost maximum in row i of M' is the rightmost maximum in row $n-1-i$ of M. Therefore, we now have the rightmost maximum in each row of M. Let r_i be the column in which the rightmost maximum in row i appears.

Step 3. For each row i, check all the entries between m_{i,l_i} for other maxima in that row.

3. Time complexity of the algorithm

Steps 1 and 2 take time $O(n + 2n) = O(n)$ [1]. Clearly, all the row maxima have to be between the leftmost and the rightmost maxima in that row. Moreover, $m_{i,l_i} = m_{i,r_i}$. So, all that remains to be proven is that Step 3 of the algorithm takes $O(n)$ time.

From the quadrangle inequality we know that $a + d > b + c$ (see Fig. 2). From this we derive the following observations:

$a \leqslant b$ implies $c < d$,

$c \geqslant d$ implies $a > b$.

Lemma 1. $r_i \leqslant \min\{2n-1, l_{i+1}\}$.

Proof. Suppose $r_i > \min\{2n-1, l_{i+1}\}$. Let $x = m_{i,r_i} = m_{i,l_i}$, $y = m_{i,l_{i+1}}$, $z = m_{i+1,l_{i+1}}$, and $w = m_{i+1,r_i}$ (see Fig. 5).

Since x is the maximum value for row i, it must be true that $y \leqslant x$. Then by our first observation $z < w$, which contradicts the fact that z is the maximum for row $i+1$. Thus, it must be true that $r_i \leqslant \min\{2n-1, l_{i+1}\}$. $\square$

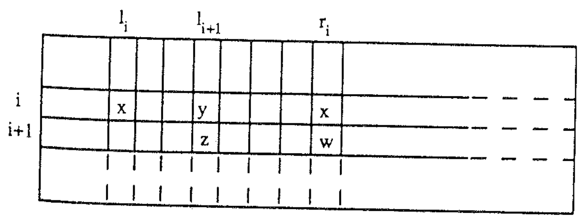

Fig. 5.

Theorem 1. *Step 3 of the Algorithm Compute-all-farthest-neighbors takes* $O(n)$ *time.*

Proof. Let $\mathcal{R}_i$ be the region between m_{i,l_i} and m_{i,r_i}. By Lemma 1, $\forall i, j$, $i \neq j$, $\mathcal{R}_i$ and $\mathcal{R}_j$ cannot overlap horizontally. Thus, each x-coordinate of matrix M will be checked in at most one such region during Step 3. Since the horizontal dimension of M is $2n-1$, we need to check at most $2n-1 = O(n)$ entries of M. $\square$

Before concluding this section, we note that Lemma 1 also shows that if f_i denotes the number of farthest neighbors of the ith vertex, then $\sum_{i=0}^{n-1} f_i \leqslant 2n$. Observe that there are convex polygons for which $\sum_{i=0}^{n-1} f_i = 2n$. For example, put $n-1$ vertices on an arc such that there are vertices at both endpoints of the arc and the size of the arc is $\frac{1}{6}$ of a circle. Let the center of the circle be the nth vertex. The center has $n-1$ farthest neighbors, each endpoint of the arc has 2 farthest neighbors, and each of the other $n-3$ points on the arc has 1 farthest neighbor. The sum is $(n-1) + 2 \times 2 + (n-3) \times 1 = 2n$. In general, since $\sum_{i=0}^{n-1} f_i \leqslant 2n$, there are at most n symmetric farthest neighbor pairs. The polygon described above again achieves the upper bound of n.

4. An application: all symmetric farthest neighbors of a simple polygon

It is well known that the farthest neighbor of any point inside a convex region is one of the vertices on the convex hull. Also, the convex hull of a simple polygon can be found in $O(n)$ time [3]. So, to find all symmetric farthest neighbors of a simple polygon we first find the convex hull of the simple polygon. Then, using the algorithm in Section 2, we find all farthest neighbors of the convex hull, and finally scan the list of vertices of the convex hull to find which ones are symmetric farthest neighbors. Thus, all symmetric farthest neighbors of a simple polygon can be found in $O(n)$ time. This settles a problem raised in [4].

5. Open problems

The best known algorithm for finding all farthest neighbors of a simple n-gon takes $O(n \log(n))$ time. It remains open whether this is an optimal bound. In fact, the best known algorithm for finding all farthest neighbors even of a simple unimodal n-gon requires $O(n \log(n))$ time.

References

[1] A. Aggarwal, M.M. Klawe, S. Moran, P. Shor and R. Wilber, Geometric applications of a matrix-searching algorithm, *Algorithmica* **2** (1987) 209–233.

[2] A. Aggarwal and J. Park, Notes on "Notes on Searching in Multidimensional Monotone Arrays", in: *Proc. 29th IEEE FOCS, 1988*, to appear.

[3] R.L. Graham, F.F. Yao, Finding the convex hull of a simple polygon, *J. Algorithms* **4** (1983) 324–331.

[4] G.T. Toussaint, The symmetric all-farthest neighbor problem, *Comp Math. Appl.* **9** (6) (1983) 747–753.