Algorithmica (1987) 2: 209-233

Algorithmica

Linear-Time Algorithms for Visibility and Shortest Path Problems Inside Triangulated Simple Polygons

Leonidas Guibas,[1,2] John Hershberger,[1] Daniel Leven,[3] Micha Sharir,[3,4] and Robert E. Tarjan[5]

Abstract. Given a triangulation of a simple polygon P, we present linear-time algorithms for solving a collection of problems concerning shortest paths and visibility within P. These problems include calculation of the collection of all shortest paths inside P from a given source vertex s to all the other vertices of P, calculation of the subpolygon of P consisting of points that are visible from a given segment within P, preprocessing P for fast "ray shooting" queries, and several related problems.

Key Words. Triangulation, Simple polygon, Visibility, Shortest paths, Ray shooting, Computational geometry.

1. Introduction. Recently, Tarjan and Van Wyk [30] developed an algorithm for triangulating simple polygons that runs in time $O(n \log \log n)$, thereby improving the previous $O(n \log n)$ algorithm of [11] and making significant progress on a major open problem in computational geometry. Even though this result falls short of the goal of achieving a linear time bound, it shows that triangulating simple polygons is a simpler problem than sorting, and raises the hope that linear time triangulation might be possible. This result thus renews interest in linear-time algorithms on already-triangulated polygons, a considerable number of which have been recently developed (for a list of these see, e.g., [10] and [30]). (Such algorithms should, of course, be contrasted with linear-time algorithms on "raw" simple polygons, such as calculation of the convex hull of such a polygon P [12], [22], calculation of the subpolygon of P visible from a given point [8], [19], and others.) Problems known to be solvable in linear time, given a triangulation of the polygon P, include calculation of the shortest path inside P between two specified points [21], preprocessing P to support logarithmic-time point location queries [18], [5], and stationing guards in simple art galleries [9].

[1] Computer Science Department, Stanford University, Stanford, CA 94305, USA.
[2] DEC/SRC, 130 Lytton Avenue, Palo Alto, CA 94301, U.S.A.
[3] School of Mathematical Sciences, Tel-Aviv University, Tel Aviv 69978, Israel.
[4] Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA.
Work on this paper by this author has been supported by Office of Naval Research Grant N00014-82-K-0381, National Science Foundation Grant No. NSF-DCR-83-20085, and by grants from the Digital Equipment Corporation, the IBM Corporation, and from the U.S.-Israel Binational Science Foundation.
[5] Department of Computer Science, Princeton University, Princeton, NJ 08544, USA, and AT&T Bell Laboratories, Murray Hill, NJ 07974, USA.
Work on this paper by this author has been supported by National Science Foundation Grant DCR-86-05962.

Received May 14, 1986; revised December 9, 1986. Communicated by Bernard Chazelle.

In this paper we continue the search for linear-time postprocessing algorithms on triangulated simple polygons. We present several such algorithms, which solve solve the following problems, given a triangulated simple polygon P with n sides:

1. Given a fixed source point x inside P, calculate the shortest paths inside P from x to all vertices of P. (Our algorithm even provides a linear-time processing of P into a data structure from which the length of the shortest path inside P from x to any desired target point y can be found in time $O(\log n)$; the path itself can be found in time $O(\log n+k)$, where k is the number of segments along the path.)
2. Given a fixed edge e of P, calculate the subpolygon $\mathrm{Vis}(P, e)$ consisting of all points in P visible from (some point on) e.
3. Given a fixed edge e of P, preprocess P so that, given any query ray r emanating from e into P, the first point on the boundary of P hit by r can be found in $O(\log n)$ time.
4. Given a fixed edge e of P, preprocess P so that, given any point x inside P, the subsegment of e visible from x can be computed in $O(\log n)$ time.
5. Preprocess P so that, given any point x inside P and any direction u, the first point $\mathrm{hit}(x, u)$ on the boundary of P hit by the ray in direction u from x can be computed in $O(\log n)$ time. (To solve this problem, we use the techniques described in [4], but show how to construct the data structure they need in linear time.)
6. Calculate a balanced decomposition tree of P by recursively cutting P along diagonals, as in [3].
7. Given a vertex x of P lying on its convex hull, calculate for all other vertices y of P the clockwise and counterclockwise *convex ropes* around P from x to y, when such paths exist. (These are polygonal paths in the exterior of P from x to y that wrap around P, always turning in a clockwise (resp. counterclockwise) direction; see Section 2.1.)

Our results improve previous algorithms for some of these problems (see [3], [4], and [24]). Most of our algorithms are based on the solution to Problem 1 and exploit interesting relationships between visibility and shortest path problems on a simple polygon. Our technique for solving Problem 1 extends the technique of Lee and Preparata [21] for calculating the shortest path inside P between a single pair of points. To obtain an overall linear-time performance, it uses *finger search trees*, a data structure for efficient access into an ordered list when there is locality of reference (see [14], [17], and [30]).

The paper contains four sections. In Section 2 we present our linear-time solution to the shortest path problem (Problem 1), and also obtain a solution to Problem 7 as an easy application. In Section 3 we solve the visibility problems (Problems 2–4). In Section 4 we address Problem 5, and in the Appendix we present our balanced tree decomposition algorithm for Problem 6.

2. Calculating a Shortest Path Tree for a Simple Polygon. Let P be a (triangulated) simple polygon having n vertices, and let s be a given *source vertex* of P. (Our algorithm will also apply, with some minor modifications, to the case in