Constant-work-space algorithms for geometric problems

Dusan Trtica

November 2023

1 Introduction

The main characteristic of constant-work-space algorithms is they require constantly many cells of storage in addition to their input, which is provided as a read-only array. In this paper, authors show some of the important computational geometry algorithms that use no extra memory sacrificing execution speed. Traditional algorithms typically process the input in order to create efficient data structures which are at a later point utilized to access, for example, adjacent points. In case of constant work space algorithms typically linear time to read input at each step is needed.
Several fundamental geometric algorithms are presented in this paper, including constructing a triangulation for a planar point set; finding the trapezoidal decomposition of a polygonal region; computing the Delaunay triangulation, Voronoi diagram, and Euclidean MST of a planar point set; and finding a geodesic shortest path in a simple polygon.

2 About the authors

- **Tetsuo Asano**, born 1949 in Kyoto Prefecture, Japan is a Japanese computer scientist, the president of the Japan Advanced Institute of Science and Technology. His main research interest is in computational geometry. Asano was a student at Osaka University, earning bachelor's, master's, and doctoral degrees there in 1972, 1974, and 1977. He was on the faculty of Osaka Electro-Communication University from 1977 until 1998, when he joined JAIST. From 2012 to 2014 he was dean of the School of Information Science at JAIST. He became president of JAIST in April 2011
- **Wolfgang Johann Heinrich Mulzer** is a German professor of Computer Science at Freie Universitat, Berlin. Most of his work is in the field of computational geometry, where he explores how additional structure in the inputs can be exploited to find faster algorithms for classical problems, like Delaunay triangulation and convex hull computation.

- **Prof. Dr. Gunter Rote** is an Austrian professor of Computer Science at Freie Universitat, Berlin

3 Motivation

You may pose the question what is the essential value which this approach and work brings to us. With hard-drives surpassing the terabyte mark, programmers can exploit a virtually unlimited amount of work-storage for their programs. While this is true more than ever, on the flip side, this leads to space-inefficient and clumsy programs that use too much storage and become very slow if sufficiently large memory is not at their disposal. Authors strongly believe that space-efficient programs deserve more attention.
More importantly, in some scenarios, it is even impossible to provide a sufficient amount of memory for an extended period of time without significantly elevating the costs of the system. Sensor networks provide an excellent example: with flash memory becoming cheaper, even a large number of expensive sensors can be equipped with huge-volume flash drives. While the data measured by the sensor must be stored onboard for processing and needs to be written, it is preferable to write to the flash drive as little as possible, since this is a slow and expensive operation and reduces the flash drive's lifetime. Hence, we would like to process the data while performing only read operations on the flash drive and using only higher-level memory for writing (e.g., CPU registers).

4 Results

Traditionally, geometry algorithms employ efficient data structures such are KD trees, linked lists, DCEL structures. With these structures at hand, operations for finding the clockwise next edge for a given edge incident to one of its endpoints, finding the triangles or trapezoids incident to a given edge, finding the twin edge for a given edge, etc, typically require constant time. In case of this paper's work, we do not have the space for such a structure at our disposal. Instead, we establish a scheme for executing these operations without referring to any data structures. Usually, each operation needs a single scan over the input data, resulting in linear time per operation, and quadratic time overall.
Our algorithms need more time than those in the standard computational model, but when considering the product of the time and space requirements, we actually often improve over the previous results: while existing algorithms for Delaunay triangulations and Voronoi diagrams need $\mathcal{O}(n \log n)$ time and linear space, resulting in a time-space product of $\mathcal{O}(n^2 \log n)$, we obtain a time-space product of $\mathcal{O}(n^2) * \mathcal{O}(1) = \mathcal{O}(n^2)$

5 Related results

There have been several previous results on log-space algorithms. One of the most important one is the selection algorithms by Munro and Raman [1] which runs in $\mathcal{O}(n^{1+\epsilon})$ using work-space $\mathcal{O}(1/\epsilon)$, for any small constant $\epsilon > 0$. Further, in 2005, Reingold solved a long standing open problem in complexity theory by describing a deterministic log-space algorithm for finding a path between two given vertices in an undirected graph [2]. A similar but more restricted computational model is used by Lenz [3] Chan and Chen [4] present algorithms that have read-only random access to the input and are allowed only sublinear (but super-constant) space. They give a randomized linear-time algorithm for finding the convex hull of sorted points in the plane with $\mathcal{O}(n\delta)$space, for any $\delta > /$. They also show how to perform linear programming in constant dimension, using $\mathcal{O}(n)$ expected time and $\mathcal{O}(n \log n)$ cells of work-space

6 Example: Triangulation of a Point set

Let's describe a constant-work-space algorithm that uses the plane sweep paradigm in order to compute a triangulation of a planar n-point set S. The main idea is to sweep over the point set in a non-decreasing order of x-coordinate, applying the gift-wrapping algorithm for convex hulls in order to add the triangles for the next point. Suppose that we were given a poligon P which is defined by its points $p_1, ...p_n$. First, we sort points in a nondecreasing order. The first triangle is defined by the first three points p_1, p_2, p_3. Starting from p_4 we scan each edge e (in case of p_4 those are the triangle's edges) and whenever an edge is visible from p_4 we add a triangle defined by the edge and point p_4 and move on to the next point p_5 and so on.

References

[1] J. I. Munro and V. Raman. Selection from a read-only memory and sorting with minimum data movement. *Theoret. Comput. Sci.*, 165(2):311–323, 1996.

[2] O. Reingold. Undirected connectivity in log-space.J. ACM, 55(4):Art. 17, 24 pp.,2008.

[3] T. Lenz. Deterministic splitter finding in a stream with constant storage and guarantees. In Proc. 17th Annu. Internat. Sympos. Algorithms Comput. (ISAAC), volume 4288 of Lecture Notes in Computer Science, pages 26–35, Kolkata, India, 2006.Springer-Verlag.

[4] T. M. Chan and E. Y. Chen. Multi-pass geometric algorithms. Discrete Comput.Geom., 37(1):79–102, 2007.

[5] Origianl paper: https://jocg.org/index.php/jocg/article/view/2961/2666