Computational Geometry

Tutorial #2 — Convex Hulls & Polygon Operations

Peter Kramer

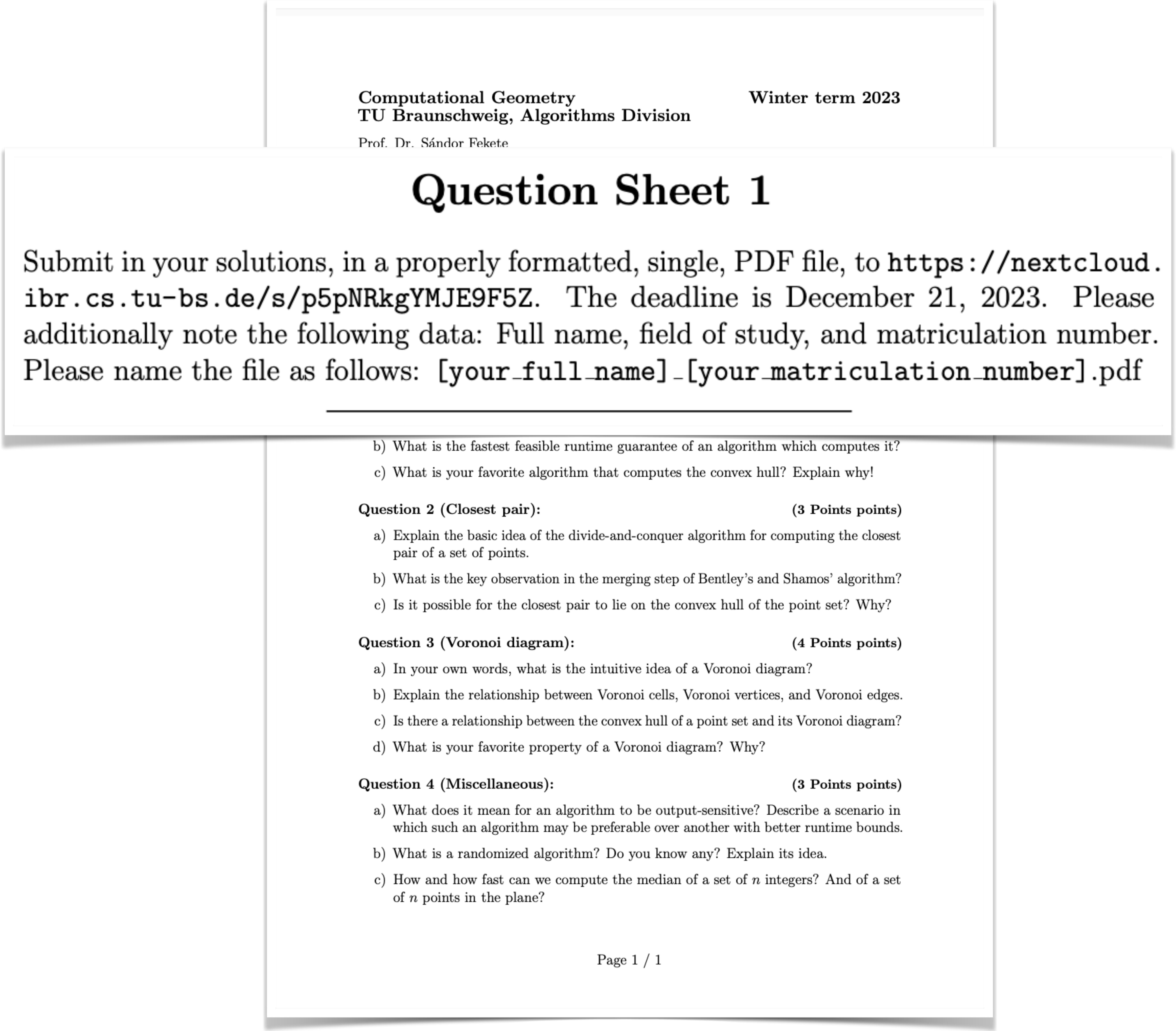
November 23, 2023

Organisation

Organisation

Question Sheet #1

- Covers **Chapters I - III**
- **Accessible on Course Website**
- Due on **Dec. 21st (in 4 weeks)**
  - Digital (properly formatted!)
  - Sketches where appropriate
- Second sheet in January



Computational Geometry | Winter term 2023
TU Braunschweig, Algorithms Division
Prof. Dr. Sándor Fekete

Question Sheet 1

Submit in your solutions, in a properly formatted, single, PDF file, to `https://nextcloud.ibr.cs.tu-bs.de/s/p5pNRkgYMJE9F5Z`. The deadline is December 21, 2023. Please additionally note the following data: Full name, field of study, and matriculation number. Please name the file as follows: `[your_full_name]_[your_matriculation_number]`.pdf

b) What is the fastest feasible runtime guarantee of an algorithm which computes it?

c) What is your favorite algorithm that computes the convex hull? Explain why!

Question 2 (Closest pair): (3 Points points)

a) Explain the basic idea of the divide-and-conquer algorithm for computing the closest pair of a set of points.

b) What is the key observation in the merging step of Bentley's and Shamos' algorithm?

c) Is it possible for the closest pair to lie on the convex hull of the point set? Why?

Question 3 (Voronoi diagram): (4 Points points)

a) In your own words, what is the intuitive idea of a Voronoi diagram?

b) Explain the relationship between Voronoi cells, Voronoi vertices, and Voronoi edges.

c) Is there a relationship between the convex hull of a point set and its Voronoi diagram?

d) What is your favorite property of a Voronoi diagram? Why?

Question 4 (Miscellaneous): (3 Points points)

a) What does it mean for an algorithm to be output-sensitive? Describe a scenario in which such an algorithm may be preferable over another with better runtime bounds.

b) What is a randomized algorithm? Do you know any? Explain its idea.

c) How and how fast can we compute the median of a set of n integers? And of a set of n points in the plane?

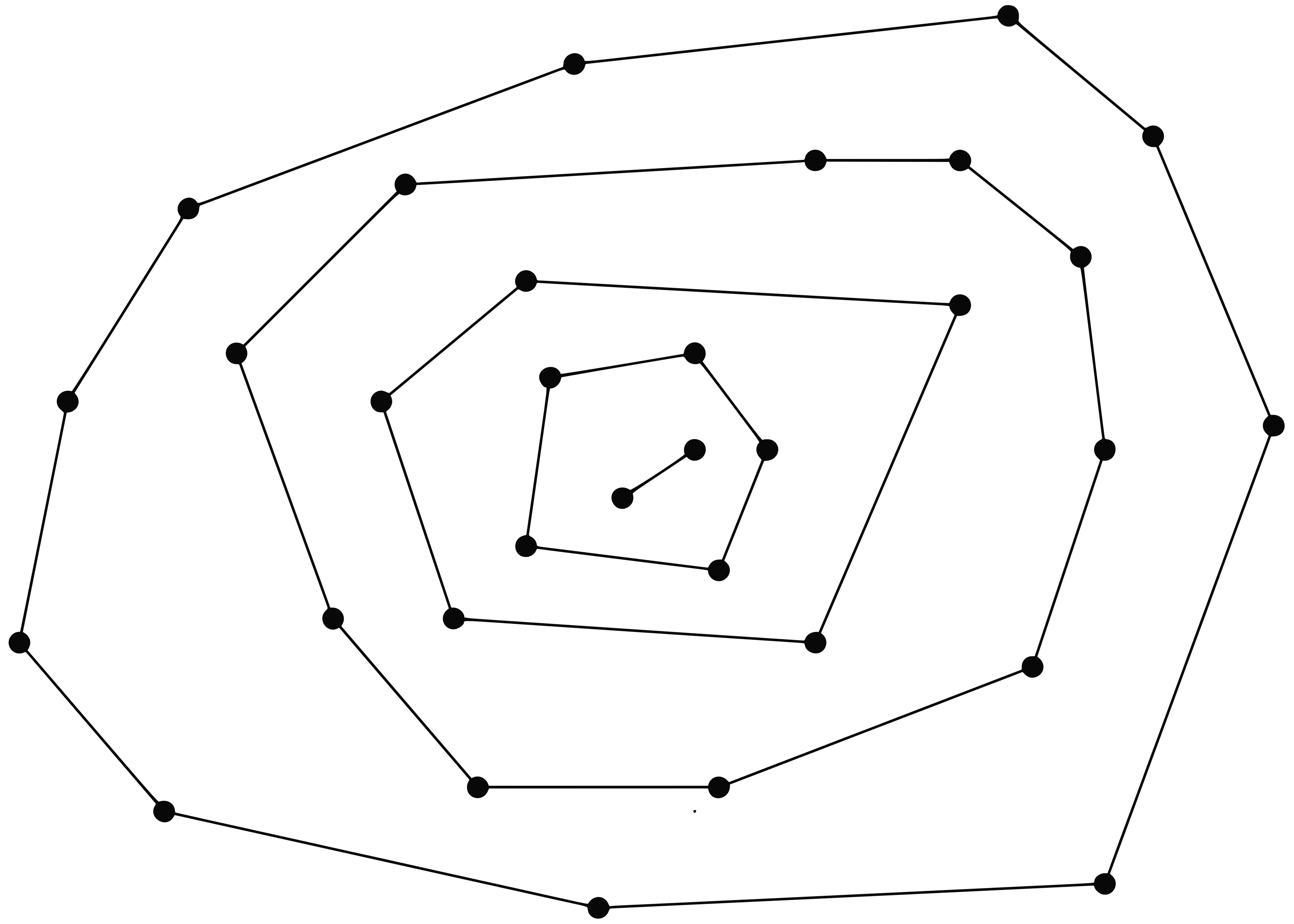
Page 1 / 1

Convex Layers & Polygons

Convex Layers of Point Sets

"Onion Decomposition"

The **convex layers** of a point set $\mathscr{P}$ are a decomposition based on repeated deletion of the convex hull vertices of $\mathscr{P}$, until there are no points left.



How (quickly) can we compute this?

Applications: Outlier Detection, Central Tendency (Probabilistic Analysis), …

Convex Layers of Point Sets

Chazelle, 1985

This is possible in $\mathcal{O}(n \log n)$ time.

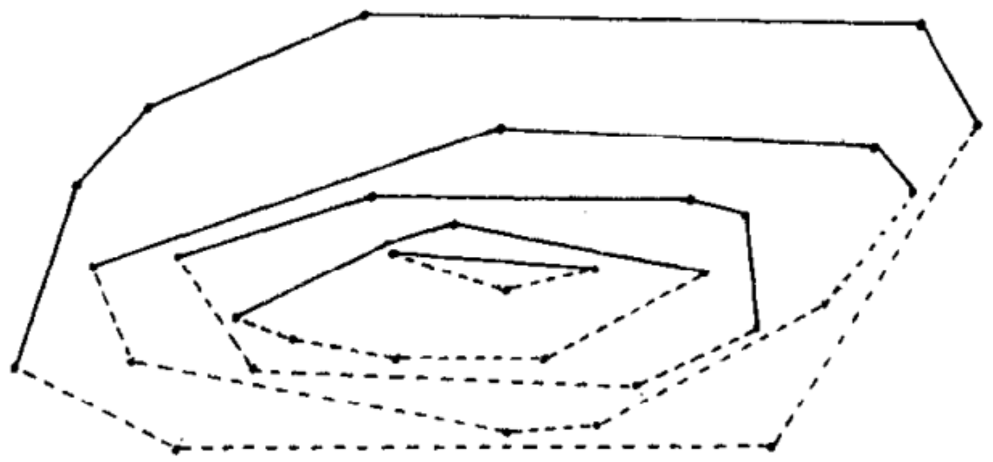


Fig. 2. Upper and lower chains.

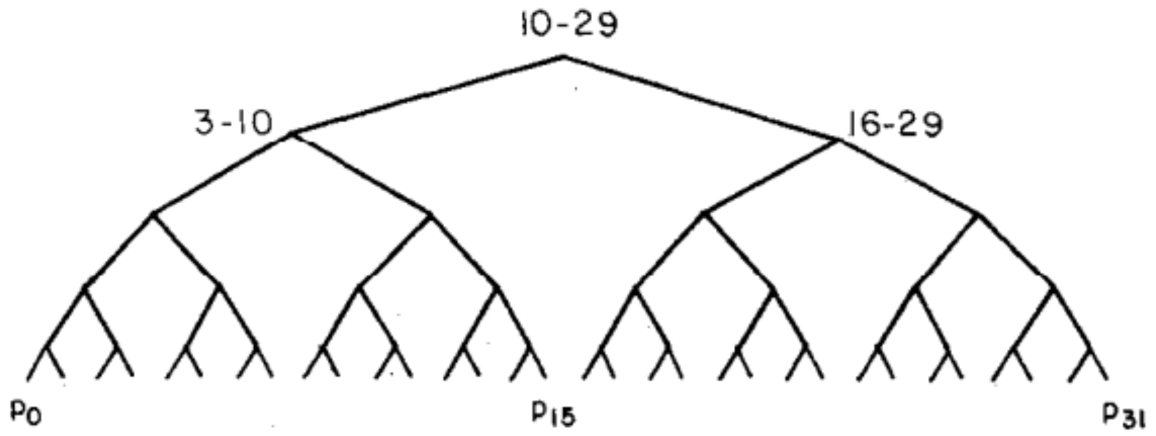
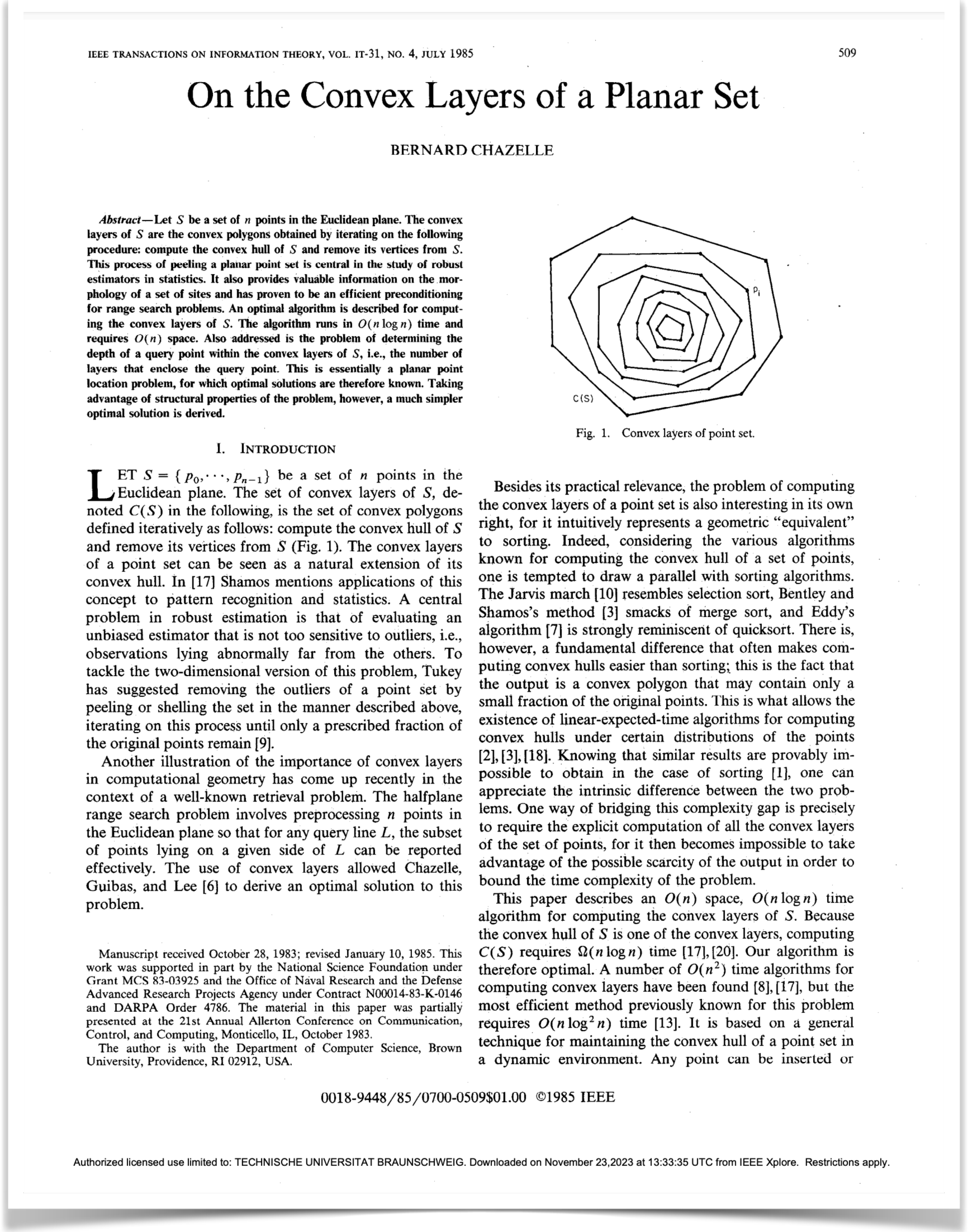


Fig. 3. Hull graph of S.



IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-31, NO. 4, JULY 1985 509

On the Convex Layers of a Planar Set

BERNARD CHAZELLE

Abstract—Let S be a set of n points in the Euclidean plane. The convex layers of S are the convex polygons obtained by iterating on the following procedure: compute the convex hull of S and remove its vertices from S. This process of peeling a planar point set is central in the study of robust estimators in statistics. It also provides valuable information on the morphology of a set of sites and has proven to be an efficient preconditioning for range search problems. An optimal algorithm is described for computing the convex layers of S. The algorithm runs in $O(n \log n)$ time and requires $O(n)$ space. Also addressed is the problem of determining the depth of a query point within the convex layers of S, i.e., the number of layers that enclose the query point. This is essentially a planar point location problem, for which optimal solutions are therefore known. Taking advantage of structural properties of the problem, however, a much simpler optimal solution is derived.

Fig. 1. Convex layers of point set.

I. Introduction

Let $S = \{p_0, \cdots, p_{n-1}\}$ be a set of n points in the Euclidean plane. The set of convex layers of S, denoted $C(S)$ in the following, is the set of convex polygons defined iteratively as follows: compute the convex hull of S and remove its vertices from S (Fig. 1). The convex layers of a point set can be seen as a natural extension of its convex hull. In [17] Shamos mentions applications of this concept to pattern recognition and statistics. A central problem in robust estimation is that of evaluating an unbiased estimator that is not too sensitive to outliers, i.e., observations lying abnormally far from the others. To tackle the two-dimensional version of this problem, Tukey has suggested removing the outliers of a point set by peeling or shelling the set in the manner described above, iterating on this process until only a prescribed fraction of the original points remain [9].

Another illustration of the importance of convex layers in computational geometry has come up recently in the context of a well-known retrieval problem. The halfplane range search problem involves preprocessing n points in the Euclidean plane so that for any query line L, the subset of points lying on a given side of L can be reported effectively. The use of convex layers allowed Chazelle, Guibas, and Lee [6] to derive an optimal solution to this problem.

Manuscript received October 28, 1983; revised January 10, 1985. This work was supported in part by the National Science Foundation under Grant MCS 83-03925 and the Office of Naval Research and the Defense Advanced Research Projects Agency under Contract N00014-83-K-0146 and DARPA Order 4786. The material in this paper was partially presented at the 21st Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, October 1983.
The author is with the Department of Computer Science, Brown University, Providence, RI 02912, USA.

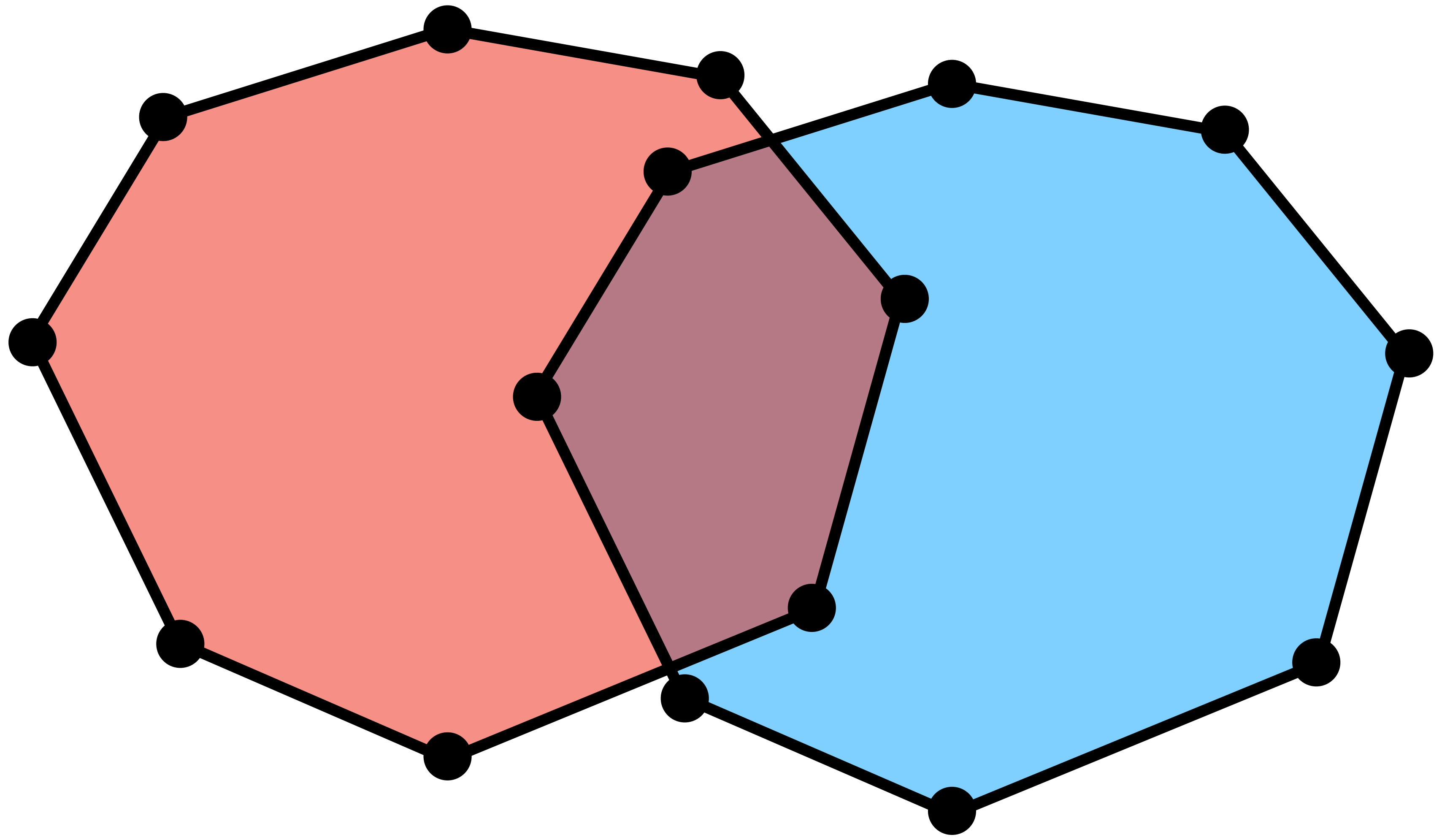
Besides its practical relevance, the problem of computing the convex layers of a point set is also interesting in its own right, for it intuitively represents a geometric "equivalent" to sorting. Indeed, considering the various algorithms known for computing the convex hull of a set of points, one is tempted to draw a parallel with sorting algorithms. The Jarvis march [10] resembles selection sort, Bentley and Shamos's method [3] smacks of merge sort, and Eddy's algorithm [7] is strongly reminiscent of quicksort. There is, however, a fundamental difference that often makes computing convex hulls easier than sorting; this is the fact that the output is a convex polygon that may contain only a small fraction of the original points. This is what allows the existence of linear-expected-time algorithms for computing convex hulls under certain distributions of the points [2], [3], [18]. Knowing that similar results are provably impossible to obtain in the case of sorting [1], one can appreciate the intrinsic difference between the two problems. One way of bridging this complexity gap is precisely to require the explicit computation of all the convex layers of the set of points, for it then becomes impossible to take advantage of the possible scarcity of the output in order to bound the time complexity of the problem.

This paper describes an $O(n)$ space, $O(n \log n)$ time algorithm for computing the convex layers of S. Because the convex hull of S is one of the convex layers, computing $C(S)$ requires $\Omega(n \log n)$ time [17], [20]. Our algorithm is therefore optimal. A number of $O(n^2)$ time algorithms for computing convex layers have been found [8], [17], but the most efficient method previously known for this problem requires $O(n \log^2 n)$ time [13]. It is based on a general technique for maintaining the convex hull of a point set in a dynamic environment. Any point can be inserted or

0018-9448/85/0700-0509$01.00

Applications: Outlier Detection, Central Tendency (Probabilistic Analysis), …

Boolean Operations on Convex Polygons



Given two convex Polygons P and Q, we seek to determine:

$$P \cap Q, P \cup Q, P \setminus Q, (Q \setminus P)$$

Which properties of the resulting polygons can you think of?

Boolean Operations on Convex Polygons

Given two convex Polygons P and Q,
we seek to determine:

$$P \cap Q, P \cup Q, P \backslash Q, (Q \backslash P)$$

Which properties of the resulting polygons can you think of?

Which concepts from the lecture could we use?