In-Place Algorithms for Computing (Layers of) Maxima

Henrik Blunck · Jan Vahrenhold

Received: 8 February 2007 / Accepted: 11 April 2008 / Published online: 2 May 2008
© Springer Science+Business Media, LLC 2008

Abstract We describe space-efficient algorithms for solving problems related to finding maxima among points in two and three dimensions. Our algorithms run in optimal $O(n \log n)$ time and occupy only constant extra space in addition to the space needed for representing the input.

Keywords In-place algorithms · Pareto-optimal points · Computational geometry

1 Introduction

Space-efficient solutions for fundamental algorithmic problems such as merging, sorting, or partitioning have been studied over a long period of time; see [12, 13, 15, 18, 25, 28]. The advent of small-scale, handheld computing devices and an increasing interest in utilizing fast but limited-size memory, e.g., caches, recently led to a renaissance of space-efficient computing with a focus on processing geometric data. Brönnimann et al. [6] were the first to consider space-efficient geometric algorithms and showed how to optimally compute 2d-convex hulls using constant extra space. Subsequently, a number of space-efficient geometric algorithms, e.g., for computing
3d-convex hulls and its relatives, as well as for solving intersection and proximity problems, have been presented [1, 2, 8, 9, 27].

In this paper, we consider the fundamental geometric problems of computing the maxima of point sets in two and three dimensions and of computing the layers of maxima in two dimensions. Given two points \( p \) and \( q \), the point \( p \) is said to dominate the point \( q \) iff the coordinates of \( p \) are larger than the coordinates of \( q \) in all dimensions. A point \( p \) is said to be a maximal point (or: a maximum) of \( P \) iff it is not dominated by any other point in \( P \). The union \( \text{MAX}(P) \) of all points in \( P \) that are maximal is called the set of maxima of \( P \). This notion can be extended in a natural way to compute layers of maxima [5]. After \( \text{MAX}(P) \) has been identified, the computation is repeated for \( P := P \setminus \text{MAX}(P) \), i.e., the next layer of maxima is computed. This process is iterated until \( P \) becomes empty.

Related Work The problem of finding maxima of a set of \( n \) points has a variety of applications in statistics, economics, and operations research (as noted by Preparata and Shamos [23]), and thus was among the first problems studied in Computational Geometry: In two and three dimensions, the best known algorithm which has been developed by Kung, Luccio, and Preparata [17] identifies the set of maxima in \( O(n \log n) \) time which is optimal since the problem exhibits a sorting lower bound [17, 23]. For constant dimensionality \( d \geq 4 \), their divide-and-conquer approach yields an algorithm with \( O(n \log^d n) \) running time [3, 17], and Matoušek [20] gave an \( O(n^{2.688}) \) algorithm for the case \( d = n \). The problem has also been studied for dynamically changing point sets in two dimensions [16] and under assumptions about the distribution of the input points in higher dimensions [4, 14]. Buchsbaum and Goodrich [5] presented an \( O(n \log n) \) algorithm for computing the layers of maxima for point sets in three dimensions. Their approach is based on the plane-sweeping paradigm and relies on dynamic fractional cascading to maintain a point-location structure for dynamically changing two-dimensional layers of maxima. Using a dynamic convex hull algorithm, an \( O(n \log^2 n) \) algorithm by Overmars and van Leeuwen [22] computes the set of convex layers, i.e., the decomposition of a two-dimensional point set into nested convex polygons instead of a decomposition into sets of maxima. This approach was later refined by Chazelle [10] to obtain optimal \( O(n \log n) \) running time.

The maxima problem has been actively investigated in the database community following Börzsönyi, Kossmann, and Stocker’s [7] definition of the SQL “skyline” operator. Börzsönyi et al. [7] noted that such an operator producing the set of maxima is needed in queries that, e.g., ask for hotels that are both close to the beach and have low room rates.\(^1\) Following their definition, a number of results have been presented that use spatial index structures to produce the “skyline”, i.e., the set of maxima, practically efficient and/or in a progressive way, that is outputting results while the algorithm is running [19, 24, 26]. It remains open, though, to prove non-trivial upper bounds for the complexity of these approaches.

\(^1\)Technically speaking, this example query needs an operator that returns the set of minima. To unify the presentation, we do not distinguish between these (and similar) algorithmically equivalent variants of the same problem.