Information Processing Letters 116 (2016) 361–366

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Linear space adaptive data structures for planar range reporting

Ananda Swarup Das [a,*], Prosenjit Gupta [b,*]

[a] IBM India Research Labs, New Delhi, India
[b] NIIT University, Neemrana, Rajasthan, India

ARTICLE INFO

Article history:
Received 15 August 2014
Received in revised form 22 December 2015
Accepted 5 January 2016
Available online 13 January 2016
Communicated by R. Uehara

Keywords:
Range aggregate queries
Range minima queries
Linear space data structures
Layers of maxima
Algorithms
Analysis of algorithms
Computational geometry

ABSTRACT

Let S be a set of n points on an $n \times n$ integer grid. The maximal layer of S is a set of points in S that are not dominated by any other point in S. Considering Q as an axes-parallel query rectangle, we design an adaptive space efficient data structure using layers of maxima (iterative maximal layers) for reporting the points in $Q \cap S$. Our data structure needs linear space and can be queried in time $O(\log^{\varepsilon} n + A \log^{\varepsilon} n + k)$. Here A is the number of layers of maxima with points in the query rectangle, k is the size of the output and ε is a small arbitrary constant in the range of $(0, 1)$. Also, $A \leq k$. In the worst case, the query time of our data structure is $O(k \log^{\varepsilon} n + k)$. Our model of computation is the word RAM with size of each word being $\Theta(\log n)$.

1. Introduction

In this work, we study the problem of orthogonal range searching for planar points which is a frequently encountered query in geographical information systems (GIS). Applications in GIS tools need to preprocess huge volume of data into space efficient data structures that can be queried efficiently to retrieve data. In this work, we present a linear space adaptive data structure for the problem. The notion of adaptive data structures is not new and instances of such data structures can be found in [1,5]. As stated in [5], intuitively, an adaptive algorithm is the one which enforces its running time to be small for easier instances of the problem while allowing the run time to be large for difficult instances. Thus, an adaptive algorithm focuses on the special instances (say sortedness of the data for example) and performs better for the special cases. For a concrete example, consider the problem of computing the maximal layer for a planar point set. Given a point set, the maximal layer is the subset of the points that are not dominated by any other point in the set. Assuming $p = (p_x, p_y)$ and $q = (q_x, q_y)$ to be two points in $\mathbb{R}^2$ with distinct coordinates, the point p dominates q if $q_x < p_x$ and $q_y < p_y$. It is known that the maximal layer for a point set can be computed in $O(n \log n)$ time. However, if the points are already sorted in non-increasing order of their x-coordinates, then one can traverse the point set from left to right. The first point (the one with the maximum x-coordinate) is sure to be a maximal point. The next point (the one with the second most maximum x-coordinate) will also be a maximal point if its y-coordinate is greater than the y-coordinate of the last discovered maximal point. Thus, if the data is already sorted, the maximal layer for the data set can be found in $O(n)$ time. The previous and the only known adaptive data structure for the problem of orthogonal planar range searching [1] solves the problem in pointer machine model. Our model of computation is the word RAM with size of each word equal to $\Theta(\log n)$. To the best of

* Corresponding authors.
E-mail addresses: anandaswarup@gmail.com (A.S. Das), prosenjit_gupta@acm.org (P. Gupta).

http://dx.doi.org/10.1016/j.ipl.2016.01.001
0020-0190/

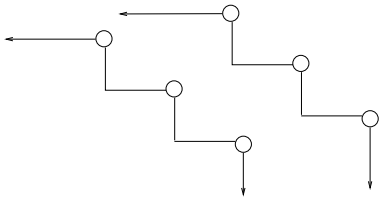

Fig. 1. The layers of maximas for a planar point set. The segments with the arrow heads are semi-infinite segments.

our knowledge, this is the first adaptive data structure for the problem in the word RAM model.

The solution that we propose here decomposes the point set into maximal layers (which we discuss shortly) and can be queried in time $O((1+A)\log^{\varepsilon} n + k)$ where A is the number of layers of maxima with points in the query rectangle, k is the size of the output and ε is a small arbitrary constant in the range of $(0, 1)$. Also, $A \leq k$. Thus, if $A = \frac{k}{\log^{\varepsilon} n}$, our algorithm has a query time of $O(\log^{\varepsilon} n + k)$. The proposed data structure needs linear space. In the worst case that is when $A = k$, our query algorithm has a run time of $O(k \log^{\varepsilon} n + k)$. Thus, our proposed data structure has a worst case performance guarantee which is at par with the performance of the best known linear space data structure of [4] for the problem. In this context, it should be mentioned that the problem of beating $O(k \log^{\varepsilon} n)$ is a long standing open problem. Though our result is not an improvement in the worst case sense, it is interesting as it shows that it is possible to push the penalty of $O(\log^{\varepsilon} n)$ to a potentially smaller term which in this case is the number of layers of maxima with at least one output point. It should be noted that a similar idea of decomposing the point set into untangled chains can be found in [1].

1.1. Layers of maxima

Layers of maxima as defined in [3] is the iterative lists of maximal points discovered as follows: Find the maximal points for the set S. Denote the set of these maximal points as S_1. Remove these maximal points from S and then find the maximal points in the remaining set. Denote the new set of maximal points as S_2. Iterate until the set S is empty. The *iteration index* i at which a point p becomes a maximum point is defined to be its layer. See Fig. 1. It should be noted that for each layer, the points can be arranged in increasing order of their x-coordinates and can be joined by using alternate horizontal and vertical segments such that the layers are disjoint as shown in Fig. 1. We will use the *iteration indices* to distinctly denote the layers.

1.2. Our contributions

We assume our model of computation to be the unit-cost RAM with word size of $\log n$ bits. The primary result of this work is a linear space solution to the fundamental problem of reporting points in a query rectangle. The problem is defined as follows.

Table 1
In the last row, the term A denotes the number of maximal layers with at least one point in the query rectangle Q. The results stated in the last row is adaptive in nature. All the results are in word-RAM model.

Query time	Source
$O(\frac{\log n}{\log\log n} + k\log^{\varepsilon} n)$	[6]
$O(\log^{\varepsilon} n + k\log^{\varepsilon} n)$	[7]
$O(\log^{\varepsilon} n + k\log^{\varepsilon} n)$	[4]
$O(\log^{\varepsilon} n + A\log^{\varepsilon} n + k)$	**this work**

Problem 1. Given a static data set S of n points on an $n \times n$ integer grid, preprocess S into a linear space data structure such that given an axes-parallel query rectangle Q, the points in $S \cap Q$ can be reported efficiently.

We denote by A' the number of layers of maxima intersecting the query rectangle while A is used to denote the number of layers of maxima with at least one point in the query rectangle. k denotes the output size. $A \leq k$. By Q, we denote the orthogonal query rectangle and n denotes the size of the data set. Throughout this work, ε is an arbitrarily small constant. The constants of proportionality in the big-Oh notation increase as ε decreases. For the above problem, we present the following results.

Theorem 1. *There exists a linear space data structure for a static data set of n points on an $n \times n$ grid such that given an axes-parallel query rectangle Q, we can report in time $O(\log^{\varepsilon} n + A\log^{\varepsilon} n + k)$, the points in $S \cap Q$.*

Though the above result is not an improvement in the worst case sense, it is interesting as it shows that it is possible to push the penalty of $O(\log^{\varepsilon} n)$ to a potentially smaller term which in this case is the number of layers of maxima with at least one output point.

Table 1 summarizes the efficiency of different linear space data structures known for the problem in the word RAM model.

2. Range reporting using layers of maxima

A brief sketch of our solution is as follows. The points of the set S are stored in layers of maxima. We first find a point p which is certain to be inside the query rectangle Q, unless the result is an empty set. Suppose that p belongs to layer S_i. We then traverse the layer of maxima S_i to which the point p belongs and report points from S_i that also fall inside Q. Using the layer S_i as a reference, we then check if layers S_j for $j > i$ and layers S_j for $j < i$ have points inside Q. This is done by considering the layers $S_{i+1}, S_{i+2}, \cdots$ in that order for the former and the layers $S_{i-1}, S_{i-2}, \cdots$, in that order for the latter. If we come across a layer S_j such that S_j intersects Q but does not have any points inside Q, then we have two possibilities. Either there are no further points inside Q, or there are layers numbered higher (or lower) than j which contain points inside Q. In the latter case, we again find a point which is certain to be in Q and identify the layer to which this point belongs. We keep on repeating the above steps until we find a layer which does not intersect Q.