Two-Dimensional Range Minimum Queries*

Amihood Amir[1], Johannes Fischer[2], and Moshe Lewenstein[1]

[1] Computer Science Department,
Bar Ilan University,
Ramat Gan 52900, Israel
{moshe,amir}@cs.biu.ac.il
[2] Ludwig-Maximilians-Universität München,
Institut für Informatik,
Amalienstr. 17, D-80333 München
Johannes.Fischer@bio.ifi.lmu.de

Abstract. We consider the two-dimensional Range Minimum Query problem: for a static $(m \times n)$-matrix of size $N = mn$ which may be preprocessed, answer on-line queries of the form "where is the position of a minimum element in an axis-parallel rectangle?". Unlike the one-dimensional version of this problem which can be solved in provably optimal time and space, the higher-dimensional case has received much less attention. The only result we are aware of is due to Gabow, Bentley and Tarjan [1], who solve the problem in $O(N \log N)$ preprocessing time and space and $O(\log N)$ query time. We present a class of algorithms which can solve the 2-dimensional RMQ-problem with $O(kN)$ additional space, $O(N \log^{[k+1]} N)$ preprocessing time and $O(1)$ query time for any $k > 1$, where $\log^{[k+1]}$ denotes the iterated application of $k+1$ logarithms. The solution converges towards an algorithm with $O(N \log^* N)$ preprocessing time and space and $O(1)$ query time. All these algorithms are significant improvements over the previous results: query time is optimal, preprocessing time is quasi-linear in the input size, and space is linear. While this paper is of theoretical nature, we believe that our algorithms will turn out to have applications in different fields of computer science, e.g., in computational biology.

1 Introduction

One of the most basic problems in computer science is finding the minimum (or maximum) of a list of numbers. An elegant and interesting dynamic version of this problem is the *Range Minimum Query (RMQ) problem*. The popular one dimensional version of this problem is defined as follows:

INPUT: An array $A[1..n]$ of natural numbers.
We seek to preprocess the array A in a manner that yields efficient solutions to the following queries:

* The second author was partially funded by the German Research Foundation (DFG, Bioinformatics Initiative).

B. Ma and K. Zhang (Eds.): CPM 2007, LNCS 4580, pp. 286–294, 2007.

QUERY: Given $1 \leq i \leq j \leq n$, output an index k, $i \leq k \leq j$, such that $A[k] \leq A[\ell]$, $i \leq \ell \leq j$.

Clearly, with $O(n^2)$ preprocessing time, one can answer such queries in constant time. Answering queries in time $O(j-i)$ needs no preprocessing. The surprising news is that linear time preprocessing can still yield constant time query solutions. The trek to this result was long. Harel and Tarjan [2] showed how to solve the *Lowest Common Ancestor (LCA)* problem with a linear time preprocessing and constant time queries. The LCA problem has as its input a tree. The query gives two nodes in the tree and requests the lowest common ancestor of the two nodes. It turns out that constructing a Cartesian tree of the array A and seeking the LCA of two indices, gives the minimum in the range between them [1].

The Harel-Tarjan algorithm was simplified by Schieber and Vishkin [3] and then by Berkman et al. [4] who presented optimal work parallel algorithms for the LCA problem. The parallelism mechanism was eliminated and a simple serial algorithm was presented by Bender and Farach-Colton [5]. In all above papers, there was an interplay between the LCA and the RMQ problems. Fischer and Heun [6] presented the first algorithm for the RMQ problem with linear preprocessing time, optimal $2n + o(n)$ bits of additional space, and constant query time that makes no use of the LCA algorithm. In fact, LCA can then be solved by doing RMQ on the array of levels of the tree's inorder tree traversal. This last result gave another beautiful motivation to the naturally elegant RMQ problem.

The problem of finding the minimum number in a given range is by no means restricted to one dimension. In this paper, we investigate the two-dimensional case. Consider an $(m \times n)$-matrix of $N = mn$ numbers. One may be interested in preprocessing it so that queries seeking the minimum in a given rectangle can be answered efficiently. Gabow, Bentley and Tarjan [1] solve the problem in $O(N \log N)$ preprocessing time and space and $O(\log N)$ query time.

We present a class of algorithms which can solve the 2-dimensional RMQ-problem with $O(kN)$ additional space, $O(N \log^{[k+1]} N)$ preprocessing time and $O(1)$ query time for any $k > 1$. The solution converges towards an algorithm with $O(N \log^* N)$ preprocessing time and space and $O(1)$ query time.

2 Preliminaries

Let us first give some general definitions. By $\log n$ we mean the binary logarithm of n, and $\log^{[k]} n$ denotes the k-th iterated logarithm of n, i.e. $\log^{[k]} n = \log \log \ldots \log n$, where there are k log's. Further, $\log^* n$ is the usual *iterated logarithm* of n, i.e., $\log^* n = \min\{k : \log^{[k]} n \leq 1\}$. For natural numbers $l \leq r$, the notation $[l : r]$ stands for the set $\{l, l+1, \ldots, r\}$.

Now let us formally define the problem which is the issue of this paper. We are given a 2-dimensional array $A[0 : m-1][0 : n-1]$ of size $m \times n$. We wish to preprocess A such that queries asking for the position of the minimal element in an axis-parallel rectangle (denoted by $\text{RMQ}(y_1, x_1, y_2, x_2)$ for *range*