Information Processing Letters 20 (1985) 13–17
North-Holland

2 January 1985

SELECTION IN X + Y AND MATRICES WITH SORTED ROWS AND COLUMNS *

A. MIRZAIAN and E. ARJOMANDI

Department of Computer Science, York University, Downsview, Ontario M3J 1P3, Canada

Communicated by L. Boasson
Received February 1984
Revised June 1984

Let A be an $n \times n$ matrix of reals with sorted rows and columns and k an integer, $1 \leqslant k \leqslant n^2$. We present an $O(n)$ time

Let A be an $n \times n$ matrix of reals with sorted rows and columns and k an integer, $1 \leqslant k \leqslant n^2$. We present an O(n) time algorithm for selecting the kth smallest element of A. If X and Y are sorted n-vectors of reals, then the Cartesian sum $X+Y$ is such a matrix as A. One application of selection in $X+Y$ can be found in statistics. The algorithm presented here is based on a new divide-and-conquer technique, which can be applied to similar order related problems as well. Due to the fact that the algorithm has a relatively small constant time factor, this result is of practical as well as theoretical interest.

Keywords: Matrix, sorting, divide-and-conquer

1. Introduction

In this paper we consider the selection problem in matrices with sorted rows and columns. Selection in $X+Y$, where X and Y are sorted vectors, is a special case of this problem.

Let $X=(x_1, x_2, \ldots, x_n)$ and $Y=(y_1, y_2, \ldots, y_n)$ be two vectors of real numbers. The Cartesian sum $X+Y$ is the $n \times n$ matrix with ijth entry $x_i + y_j$. If X and Y are sorted, then $X+Y$ is a matrix with sorted rows and columns. Selection and other re-

O(n log n) algorithm for selecting the kth smallest element in $X+Y$. Both algorithms in [10] and [8] sort the vectors X and Y before the algorithms may proceed. Despite the time required to sort X and Y, both these algorithms still require O(n log n) time. Frederickson and Johnson [2] consider selection in matrices with sorted columns. Their algorithm for selecting the kth largest element of $X+Y$, $1 \leqslant k \leqslant \frac{1}{2}n^2$, runs in $O(\max\{n, n\log(k/n)\})$ time. They also give an O(n) time algorithm for selection in matrices with sorted rows and columns [3].

1. Introduction

In this paper we consider the selection problem in matrices with sorted rows and columns. Selection in $X+Y$, where X and Y are sorted vectors, is a special case of this problem.

Let $X=(x_1, x_2, \ldots, x_n)$ and $Y=(y_1, y_2, \ldots, y_n)$ be two vectors of real numbers. The Cartesian sum $X+Y$ is the $n \times n$ matrix with ij*th* entry $x_i + y_j$. If X and Y are sorted, then $X+Y$ is a matrix with sorted rows and columns. Selection and other related problems in $X+Y$ have received considerable attention, due to their application in statistics and operations research [2,3,4,5,6,7,8,10]. $X+Y$ order related problems arise in some VLSI layout problems as well [9].

Jefferson, Shamos and Tarjan [10] present an $O(n \log n)$ time algorithm for selecting the median of $X+Y$. Johnson and Mizoguchi [8] give an $O(n \log n)$ algorithm for selecting the k*th* smallest element in $X+Y$. Both algorithms in [10] and [8] sort the vectors X and Y before the algorithms may proceed. Despite the time required to sort X and Y, both these algorithms still require $O(n \log n)$ time. Frederickson and Johnson [2] consider selection in matrices with sorted columns. Their algorithm for selecting the k*th* largest element of $X+Y$, $1 \leqslant k \leqslant \frac{1}{2}n^2$, runs in $O(\max\{n, n\log(k/n)\})$ time. They also give an $O(n)$ time algorithm for selection in matrices with sorted rows and columns [3].

Let A be an $n \times n$ matrix of real numbers with sorted rows and columns and let k be an integer, $1 \leqslant k \leqslant n^2$. We present an $O(n)$ time algorithm to select the k*th* smallest element of A. The algorithm presented in this paper applies an elegant divide-and-conquer technique. This method may be applied to similar order related problems. For instance, we have used this technique to obtain a linear time algorithm for the optimum offset prob-

problems as well [9].

Jefferson, Shamos and Tarjan [10] present an O(n log n) time algorithm for selecting the median of X + Y. Johnson and Mizoguchi [8] give an

* This work was supported by Natural Sciences and Engineering Research Council of Canada (NSERC) Grants A5516 and A4304.

presented in this paper applies an elegant divide-and-conquer technique. This method may be applied to similar order related problems. For instance, we have used this technique to obtain a linear time algorithm for the optimum offset problem of channel routing in VLSI [9]. Although Frederickson and Johnson's algorithm [3] has a similar time bound, the algorithm presented in this paper is simpler. Also the technique used in our algorithm is of practical as well as theoretical interest.

0020-0190/85/$3.30 © 1985, Elsevier Science Publishers B.V. (North-Holland)

2. Terminology

Let A be an $n \times n$ matrix of reals. The elements of A are not necessarily distinct. We assume rows and columns of A to be indexed $1, 2, \ldots, n$. We call A *ordered* if elements in each row are in nonincreasing order, and elements in each column are in nondecreasing order. Let $\bar{n} = \lceil \frac{1}{2}(n+1) \rceil$. Submatrix $\bar{A}$ of A is an $\bar{n} \times \bar{n}$ matrix and is defined to be the submatrix of A consisting of the odd indexed rows and columns, plus the last row and column of A in case n is even. Let L be a list of reals and a be a real number. We define $rank^+$ and $rank^-$ of a in L as follows:

$$rank^+(L, a) = |\{x \in L \mid x > a\}|, \tag{2.1}$$

$$rank^-(L, a) = |\{x \in L \mid x < a\}|. \tag{2.2}$$

from $A - \bar{A}$. Since the matrix A is ordered, A_L includes all the elements of A that are less than a. Thus

$$|A_L| \geq rank^-(A, a). \tag{3.2}$$

By the construction of A_L from $\bar{A}_L$ we conclude

$$|A_L| \leq 4|\bar{A}_L|. \tag{3.3}$$

From (3.1), (3.2) and (3.3) we have $rank^-(A, a) \leq 4\, rank^-(\bar{A}, a)$. □

4. The selection algorithm

Before describing the details of the selection algorithm we present an O(n) time algorithm to in L as follows:

$$rank^{+}(L,a) = |\{x \in L \mid x > a\}|, \tag{2.1}$$

$$rank^{-}(L,a) = |\{x \in L \mid x < a\}|. \tag{2.2}$$

Suppose $1 \leqslant k \leqslant |L|$. Then a is defined to be the k*th* smallest element of L if and only if $rank^{-}(L, a) \leqslant k-1$ and $rank^{+}(L, a) \leqslant |L| - k$. For simplicity we use the term k*th element of* L to mean k*th smallest element of* L throughout this paper.

3. The main observation

The following theorem is the basis for our selection algorithm.

Theorem 3.1. *Let* A *be an* $n \times n$ *ordered matrix and* $\bar{A}$ *be the submatrix of* A *as defined earlier. Then,*

4. The selection algorithm

Before describing the details of the selection algorithm we present an O(n) time algorithm to compute $rank^{-}$ of a real number a in an $n \times n$ ordered matrix A. $rank^{+}$ may be computed similarly.

The function in Fig. 1 computes $rank^{-}(A, a)$ in O(n) time. Let *pick*(L, k) be a function which takes a list L and an integer k, $1 \leqslant k \leqslant |L|$, and returns the k*th* element of L in O(|L|) time. For such an algorithm, see [1]. Functions *pick*, $rank^{-}$ and $rank^{+}$ are used in our selection algorithm.

The idea behind our selection algorithm is to recursively select two elements a and b, $a \geqslant b$, from $\bar{A}$ so that the following hold:

(1) The k*th* element of A is between a and b.

(2) The number of elements of A which are less than a and greater than b is O(n).

The main result of this paper is that the func-

3. The main observation

The following theorem is the basis for our selection algorithm.

Theorem 3.1. *Let* A *be an* $n \times n$ *ordered matrix and* $\bar{A}$ *be the submatrix of* A *as defined earlier. Then, for any real number* a, *the following inequalities hold*:

(i) $rank^-(A, a) \leqslant 4\, rank^-(\bar{A}, a)$,

(ii) $rank^+(A, a) \leqslant 4\, rank^+(\bar{A}, a)$.

Proof. We only prove (i). Part (ii) may be proved similarly. Let $\bar{A}_L$ consist of the elements of $\bar{A}$ that are less than a. Thus

$$|\bar{A}_L| = rank^-(\bar{A}, a). \tag{3.1}$$

Let A_L be the portion of A that consists of:

(a) $\bar{A}_L$, and

and *rank*$^+$ are used in our selection algorithm.

The idea behind our selection algorithm is to recursively select two elements a and b, $a \geqslant b$, from $\bar{A}$ so that the following hold:

(1) The kth element of A is between a and b.

(2) The number of elements of A which are less than a and greater than b is $O(n)$.

The main result of this paper is that the function *select*(A, k), presented in Fig. 2, computes the kth element of an $n \times n$ ordered matrix A in $O(n)$ time. The function *select* calls the recursive function *biselect*(n, A, k_1, k_2) with $k_1 \geqslant k_2$, which returns (x, y), where x is the k_1th and y is the k_2th element of the $n \times n$ matrix A. Let $\bar{k}_1$ and $\bar{k}_2$ be defined as follows:

$$\bar{k}_1 = \begin{cases} n + 1 + \lceil \frac{1}{4} k_1 \rceil & \text{if n is even,} \\ \lceil \frac{1}{4}(k_1 + 2n + 1) \rceil & \text{if n is odd;} \end{cases} \tag{4.1}$$

$$\bar{k}_2 = \lfloor \tfrac{1}{4}(k_2 + 3) \rfloor. \tag{4.2}$$

Proof. We only prove (i). Part (ii) may be proved similarly. Let $\overline{A}_L$ consist of the elements of $\overline{A}$ that are less than a. Thus

$$|\overline{A}_L| = rank^{-}(\overline{A}, a). \tag{3.1}$$

Let $A_{\underline{L}}$ be the portion of A that consists of:

(a) $\overline{A}_L$, and

(b) for each element $A_{ij} \in \overline{A}_L$, its neighboring elements $A_{i,j-1}$, $A_{i+1,j-1}$ and $A_{i+1,j}$ (if they exist)

element of the $n \times n$ matrix A. Let $\overline{k}_1$ and $\overline{k}_2$ be defined as follows:

$$\overline{k}_1 = \begin{cases} n+1+\lceil \frac{1}{4}k_1 \rceil & \text{if n is even,} \\ \lceil \frac{1}{4}(k_1+2n+1) \rceil & \text{if n is odd;} \end{cases} \tag{4.1}$$

$$\overline{k}_2 = \lfloor \tfrac{1}{4}(k_2+3) \rfloor. \tag{4.2}$$

$\overline{k}_1$ is chosen to be the smallest integer such that

```
function rank⁻(A, a);
  begin
    j=1; x=0;
    for i=1 to n do begin
      while j ≤ n and A_ij ≥ a do j=j+1;
      x=x+n−j+1
    end;
    return x
  end;
```

Fig. 1. A ranking algorithm.

the $\bar{k}_1$th element of $\bar{A}$ is at least as large as the k_1th element of A. $\bar{k}_2$ is chosen to be the largest integer such that the $\bar{k}_2$th element of $\bar{A}$ is no larger than the k_2th element of A. In the algorithm, the phrase i*th of* A is shorthand for i*th element of* A. The first parameter n of the function *biselect* is the dimension of the submatrix which appears as the second parameter of the function (and not necessarily the

5. Proof of correctness

In order to prove the correctness and the claimed running time of the algorithm, we need the following lemma.

Lemma 5.1. *During the course of the algorithm, whenever biselect*(n, A, k_1, k_2) *is called, the following hold*:

(i) $n^2 \geq k_1 \geq k_2 \geq 1$,

(ii) $k_1 - k_2 \leq 4n - 4$.

Proof. We use induction on the number of times *biselect* is called. The function *biselect* is first called from *select* with $k_1 = k_2 = k$ and $n^2 \geq k \geq 1$. Therefore, (i) and (ii) obviously hold in this case. Otherwise, *biselect* is called from line 3. of the algorithm (see Fig. 2). In this case, by the induction hypothesis, we have $n^2 \geq k_1 \geq k_2 \geq 1$ and $k_1 - k_2 \leq$

sion of the submatrix which appears as the second parameter of the function (and not necessarily the dimension of the main matrix). We assume that either the matrix A is present in the memory before the computation begins, or the elements of A can be computed as they are needed. If A is of the form $X+Y$, then only the vectors X and Y need to be present in the memory.

```
function select(A, k);
  begin
    (x, y) := biselect(n, A, k, k);
    return x
  end select;

function biselect(n, A, k1, k2);
    begin
    if n ≤ 2
    then (x, y) := (k1th of A, k2th of A)
    else begin
      (a, b) := biselect(n̄, Ā, k̄1, k̄2);
      ra⁻ := rank⁻(A, a);
```

(see Fig. 2). In this case, by the induction hypothesis, we have $n^2 \geq k_1 \geq k_2 \geq 1$ and $k_1 - k_2 \leq 4n-4$. Furthermore, $n \geq 3$. We consider two cases, depending whether n is even or odd.

Case 1 (n is even): Recall that, in this case, $\bar{n}$, the dimension of $\bar{A}$, is $\frac{1}{2}(n+2)$. Using formulas (4.1) and (4.2) it is easy to show that $\bar{k}_1 \leq \bar{n}^2$, $\bar{k}_2 \geq 1$ and $\bar{k}_1 - \bar{k}_2 \geq 0$. Therefore, (i) holds. Furthermore, $\bar{k}_1 - \bar{k}_2 \leq 2n = 4\bar{n} - 4$.

Case 2 (n is odd): This case is proved similarly.

We conclude that $\bar{n}^2 \geq \bar{k}_1 \geq \bar{k}_2 \geq 1$ and $\bar{k}_1 - \bar{k}_2 \leq 4\bar{n} - 4$ in both cases. This completes the proof. □

Theorem 5.2. *Let* A *be an* $n \times n$ *ordered matrix and* $n^2 \geq k_1 \geq k_2 \geq 1$. *Then* $biselect(n, A, k_1, k_2)$ *returns the* k_1*th and* k_2*th elements of* A.

Proof. Let x^* and y^* be the k_1th and k_2th elements of A, respectively. Notice that $x^* \geq y^*$. We show, by induction on n, that $biselect(n, A, k_1, k_2)$ returns (x^*, y^*).