PYRAMIDS: A DATA TYPE FOR MATRIX REPRESENTATION IN PASCAL*

N. SOLNTSEFF and D. WOOD

Abstract.
Pyramids, a new data structure for the representation of matrices, are introduced. The motivation for this is Strassen's algorithm for matrix multiplication. The basic operations for pyramids are described in Pascal.

Keywords and Phrases: matrix multiplication, Strassen's algorithm, data structures, Pascal, data encoding.

Introduction.

The notion of "pyramids" has its origin in a challenge of the first author by the second to provide a "straightforward" or "natural" implementation of Strassen's algorithm [8] in Pascal [6]. In Strassen's algorithm for matrix multiplication of two \( m \times m \) matrices (with \( m = 2^k \)), the original matrices are partitioned into four \( (m/2) \times (m/2) \) submatrices. The problem has now been reduced to the multiplication of two \( 2 \times 2 \) matrices whose elements are \( (m/2) \times (m/2) \) matrices. This product can be carried out with seven multiplications and 15 additions (see [1], p. 231). Of course, since these multiplications involve matrices the technique can be applied recursively. An empirical investigation of Strassen's algorithm has been recently reported in [2].

As Strassen's algorithm is recursive, it follows that a straightforward implementation would exploit this. Pascal supports recursive procedures and, hence, there is no problem on this account. However, at each level of recursion each matrix multiplication requires nine submatrices for temporary storage. Therefore, a straightforward (or "naive") approach requires Pascal to support dynamic arrays. On the other hand, dynamic arrays are not a Pascal language feature, although they are supported, for example, in Algol 60 or PL/I. This has been held as a major deficiency of Pascal as pointed out by Haberman (see [5] and the designer's reply in [9]). Thus, at first sight, it appears to be impossible to implement Strassen's algorithm in Pascal, but some further thought revealed a new direction — away from the traditional unstructured view of arrays.

Since matrix partitioning in Strassen's algorithm is performed recursively, the original matrix can be considered as an object with nested structure in which the basic structural unit is of the form...

where the arrows represent pointers to other nodes of the same type or, if partitioning has progressed to "ground level", pointers to the individual matrix elements. A data structure of this form used to represent a matrix will be called a *pyramid*. This is essentially a quaternary or quad tree [4] representation of two-dimensional matrices. A binary-tree representation of such matrices has been suggested and investigated theoretically in [3] and [7]. The two methods of representation are compared in [10].

In the following, we concentrate on the implementation of addition, subtraction, and the usual multiplication operations for integer matrices. However, we do not present an implementation of Strassen's Algorithm as this is now straightforward.

2. Implementation.

Pyramids have been implemented in Pascal 6000* [6] and this section briefly describes the principal features of the implementation. It is assumed, for simplicity, that only square matrices of size $m$, with $m = 2^k$, need to be handled. The extension to general $m \times n$ matrices is straightforward and programs for this case are available from the authors.

Pyramids in Pascal are defined to be of pointer type $PYRAMID = \uparrow PNODE$, where $PNODE$ is defined to be the type

\[
PNODE = \text{record}
\begin{align*}
  \text{BASE: INTEGER;} \quad & \\
  \text{case BOOLEAN of} \quad & \\
  \text{false: (INT: INTEGER);} \quad & \\
  \text{true: (MTX: array [1..2,1..2] of PYRAMID)} \quad & 
\end{align*}
\]

The procedures for pyramid manipulation that have been implemented, namely, *CONSTRUCT*, to construct a pyramid representing a matrix, *STORE*, to store the numeric value of a given matrix element into a terminal node of the pyramid,