procedure SORT(i, j):
if \( i = j \) then return \( x_i \)
else
    begin
        \( m \leftarrow (i + j - 1) / 2; \)
        return MERGE(SORT(i, m), SORT(m + 1, j))
    end

Fig. 2.14. Mergesort.

ancing the size of the subproblems has paid off handsomely. A similar analy-
ysis shows that the total time, not only comparisons, spent in procedure SORT
is \( O(n \log n) \).

2.8 DYNAMIC PROGRAMMING

Recursive techniques are useful if a problem can be divided into subproblems
with reasonable effort and the sum of the sizes of the subproblems can be
kept small. Recall from Theorem 2.1 that if the sum of the sizes of the sub-
problems is \( an \), for some constant \( a > 1 \), the recursive algorithm is likely to
be polynomial in time complexity. However, if the obvious division of a prob-
lem of size \( n \) results in \( n \) problems of size \( n - 1 \), then a recursive algorithm is
likely to have exponential growth. In this case a tabular technique called
dynamic programming often results in a more efficient algorithm.

In essence, dynamic programming calculates the solution to all subprob-
lems. The computation proceeds from the small subproblems to the larger
subproblems, storing the answers in a table. The advantage of the method
lies in the fact that once a subproblem is solved, the answer is stored and
never recalculated. The technique is easily understood from a simple example.

Consider the evaluation of the product of \( n \) matrices

\[
M = M_1 \times M_2 \times \cdots \times M_n,
\]

where each \( M_i \) is a matrix with \( r_{i-1} \) rows and \( r_i \) columns. The order in which
the matrices are multiplied together can have a significant effect on the total
number of operations required to evaluate \( M \), no matter what matrix multi-
plication algorithm is used.

Example 2.7. Assume that the multiplication of a \( p \times q \) matrix by a \( q \times r \)
matrix requires \( pqr \) operations, as it does in the "usual" algorithm, and con-
sider the product

\[
M = \begin{bmatrix} M_1 & \times & M_2 & \times & M_3 & \times & M_4 \end{bmatrix}^{(2.8)}
   \begin{bmatrix} 10 \times 20 & [20 \times 50] & [50 \times 1] & [1 \times 100] \end{bmatrix}
\]