Solution

This algorithm will use a divide-and-conquer approach to determine if an ordered $N = n$ x n matrix M contains the integer x. We will begin by finding the element y in the "$middle$" of M. We then compare x to said element y, and if x is equal to y then we return `true`. Now, imagine M being divided into four square submatrices. If x is greater than y we can ignore the top left submatrix and run our algorithm on all of the three remaining submatrices. On the other hand, if x happens to be less than y, then we ignore the bottom right submatrix and run our algorithm on the remaining three submatrices. If at any point x equals the "$middle$" of a submatrix, then we return `true`; however, if we keep dividing the matrix and running our algorithm until the matrices are single cells and none of these cells are equal to x then we return `false`.

Time Complexity: Finding the "$middle$" element of a matrix takes constant time, comparing x to the "$middle$" element of a matrix also takes constant time, we run our algorithm again on three submatrices of size at most $\frac{N}{4}$. We can form the following recurrence relation: $T(N) \leq cN^0 + 3T(\frac{N}{4})$. Using the Master Theorem we determine that the runtime for this algorithm is $O(N^{\log_4 3})$.

Proof. Base Case: If our matrix is of size 1 x 1, then we simply compare x to the element of said matrix. This element happens to be the middle, so we return `true` if x equals this element or `false` if they do not equal.

Inductive Hypothesis: Assume our algorithm determines whether an ordered matrix of size up to $K = k$ x k contains the element x where $K \geq 1$.

Inductive Hypothesis: We must show that our divide-and-conquer approach accurately determines if x is an element of an ordered $K + 1 = k + 1$ x $k + 1$ matrix M^*. We can easily compare x to the "$middle$" element y of M^*. If x equals y then our algorithm returns `true` and it works. Now, if x does not equal y we take the divide-and-conquer approach. We can partition M^* into four square submatrices occupying each quadrant of M^* (Note that these submatrices are also ordered). The reason we do not run our algorithm on all four submatrices is because all elements of the upper left submatrix are less than y, while all elements of the lower right submatrix are greater than y. Thus, we only run our algorithm on the three submatrices where x is certain to be found. Notice that these submatrices are of size less than $K + 1$, and according to our inductive hypothesis, our algorithm will determine whether or not x is in these ordered matrices. We have shown that our algorithm will always determine whether an element x is in an ordered square matrix. □