Title: Searching in a 2D Sorted Matrix using Binary Search.

Introduction: Searching for an element in a sorted data structure is a fundamental problem in computer science. Binary search is one of the most efficient techniques for searching in a sorted array, reducing the time complexity to logarithmic order.

problem Statement:

Given an MxN enter matrix where;

1. Each row is sorted in ascending order.
2. The first element of each row is greater than the last element of the previous row.

The task is to determine whether a given target value exists within the matrix.

For example, Consider the following matrix:

[[1, 3, 5, 7],
[10, 11, 16, 20],
[23, 30, 34, 60]],

If the target value is 3, the function should return true. If the target value is 13, it should return false.

Brute Force Approach :- The most straight forward approach is to iterate through each element in the matrix and check if the matches the target value. This approach has a time complexity of O(M * N), which is inefficient for large matrices.

Improved Approach : Row-wise Binary Search

Since each row is sorted, we can perform a binary search on each row separately. The steps are as follows:

1. Iterate over each row.
2. Apply binary search on the row.
3. If the target is found, return true; Otherwise, move to the next row.

- The time complexity of this approach is $O(M * \log N)$, as we perform binary search ($O(\log N)$) on each of the M rows.

Binary Search Algorithm:

- Initialize low = 0 and high = M * N - 1.
- while low <= high :
  - Compute mid = (low + high)/2.
  - Retrieve matrix [row][col] using the mapping above.
  - If matrix [row][col] matches the target, return true.
  - If matrix [row][col] is less than the target, adjust low = mid + 1.
  - If matrix [row][col] is greater than the target, adjust high = mid - 1.
- → If no match is found, return false.

Conclusion:

The problem of searching for an element in a 2D sorted matrix can be solved in different ways:

1. Brute Force : $O(M * N)$
2. Row-wise Binary Search : $O(M * \log N)$
3. Optimized Binary Search (Treating as 1D Array) : $O(\log (M * N))$

Among these, the optimized Binary Search approach is the most efficient, leveraging the sorted structure of the matrix to achieve logarithmic time complexity.