Search a 2D Matrix

Key observation:

Because:

- each row is sorted
- the first of a row > last of previous row

the whole m×n matrix behaves like one sorted 1D array of length m·n in row-major order.

So you can do a single binary search in $O(\log(m \cdot n))$ time.

How to index it:

Treat an index $k \in [0, m \cdot n - 1]$ as pointing to:

- row $r = k$ // (integer division)
- col $c = k \% n$

Then matrix[r][c] is the "k-th" element in that virtual 1D array.

Algorithm (steps):

1. Set lo = 0, hi = m·n − 1
2. While lo ≤ hi
   - mid = lo + (hi − lo) / 2
   - Map mid → (r, c) as above.
   - If matrix[r][c] == target: return true.
   - If matrix[r][c] < target: move right → lo = mid + 1.
   - Else: move left → hi = mid − 1
3. If the loop ends, return false.

Complexity:

- Time: $O(\log(m \cdot n))$
- Space: $O(1)$