Staircase Search in a Sorted Matrix

Problem: Given a matrix where each row is sorted left to right and each column is sorted top to bottom, determine if a target exists in the matrix.

Example Matrix:

```
[ [1, 4, 7, 11],
  [2, 5, 8, 12],
  [3, 6, 9, 16],
  [10,13,14,17] ]
Target = 6
```

Approach: Staircase Search
Start from the top-right corner. If the current value is greater than target, move left. If it is smaller, move down. This eliminates one row or column at each step.

Time Complexity: O(rows + cols)
Space Complexity: O(1)

Dry Run:
Start at (0,3) = 11 → 11 > 6 → move left
(0,2) = 7 → 7 > 6 → move left
(0,1) = 4 → 4 < 6 → move down
(1,1) = 5 → 5 < 6 → move down
(2,1) = 6 → FOUND

Java Code:

```
public boolean searchMatrix(int[][] matrix, int target) {
  int rows = matrix.length;
  int cols = matrix[0].length;
  int r = 0, c = cols - 1;
  while (r < rows && c >= 0) {
    if (matrix[r][c] == target) return true;
    else if (matrix[r][c] > target) c--;
    else r++;
  }
  return false;
}
```

Python Code:

```
def searchMatrix(matrix, target):
  rows, cols = len(matrix), len(matrix[0])
  r, c = 0, cols - 1
  while r < rows and c >= 0:
    if matrix[r][c] == target:
      return True
    elif matrix[r][c] > target:
      c -= 1
    else:
      r += 1
  return False
```

C++ Code:

```
bool searchMatrix(vector>& matrix, int target) {
  int rows = matrix.size();
  int cols = matrix[0].size();
  int r = 0, c = cols - 1;
  while (r < rows && c >= 0) {
    if (matrix[r][c] == target) return true;
    else if (matrix[r][c] > target) c--;
    else r++;
  }
  return false;
}
```

Interview Tip: Staircase search works only when both rows and columns are sorted. It is commonly used in LeetCode-240.