📓 Diagonal Traverse - Full Documentation

1. Problem Statement

You are given an m x n matrix mat. Your task is to return all elements of the matrix in a diagonal order.

◆ **Examples:**

Input: mat = [[1,2,3],[4,5,6],[7,8,9]]
Output: [1,2,4,7,5,3,6,8,9]

Input: mat = [[1,2],[3,4]]
Output: [1,2,3,4]

2. Intuition

Matrix diagonals alternate directions:

- One diagonal goes up-right
- The next goes down-left

If we simulate this zig-zag pattern by:

- Tracking current position (row, col)
- Changing direction when hitting borders

…then we can traverse the matrix diagonally as required.

3. Key Observations

- The direction of movement switches when we hit the edges (top, bottom, left, right).
- There are m * n elements, and each must be visited exactly once.
- A pattern of diagonal traversal emerges if tracked carefully:

- ↗ (up-right): row--, col++
- ↙ (down-left): row++, col--

4. Approach

- Start at (0, 0) with direction up-right.
- Loop for m * n times.
- At each step:
  - Add current element to result.
  - Move in the current direction.
  - If a boundary is hit, switch direction and adjust position.
- Alternate between directions:
  - If direction is up-right:
    - If at the last column, move to the next row.
    - If at the first row, move to the next column.
  - If direction is down-left:
    - If at the last row, move to the next column.
    - If at the first column, move to the next row.

5. Edge Cases

- Empty matrix: Return an empty list.
- Single row or column: Traverse normally.
- Non-square matrices: Ensure direction switches still work properly at boundaries.

6. Complexity Analysis

- □ Time Complexity:
  - O(m * n) — Every element is visited once.
- □ Space Complexity:
  - O(1) extra space (excluding the output list).

7. **Alternative Approaches**

- Diagonal Hash Map (Grouping by row+col):
  - Group elements by i + j sum.
  - Reverse alternate diagonals.
  - More intuitive but uses extra space (O(m+n)).
- Flatten with math (Less readable):
  - Pre-calculate start points for diagonals.
  - Use loops with indices — not practical in interviews due to complexity.

8. **Test Cases**

Test Case 1: Square matrix

mat = [[1,2,3],[4,5,6],[7,8,9]]

Output: [1,2,4,7,5,3,6,8,9]

Test Case 2: Rectangular matrix (more columns)

mat = [[1,2,3,4],[5,6,7,8]]

Output: [1,2,5,6,3,4,7,8]

Test Case 3: Rectangular matrix (more rows)

mat = [[1,2],[3,4],[5,6]]

Output: [1,2,3,5,4,6]

Test Case 4: Single row

mat = [[1,2,3,4]]

Output: [1,2,3,4]

Test Case 5: Single column

mat = [[1],[2],[3]]

Output: [1,2,3]

Test Case 6: Empty matrix

mat = []

Output: []

9. Final Thoughts

- The key to solving this problem is identifying the zig-zag diagonal movement.
- Carefully managing direction switching and boundary checks ensures correct traversal.
- This solution is efficient and interview-friendly due to O(1) space and O(m*n) time complexity.