Documentation in Pacific Atlantic Water Flow

1. Problem Statement

You are given an m x n integer matrix heights where heights[r][c] represents the height above sea level of the cell at coordinate (r, c). The Pacific Ocean touches the island's left and top edges, and the Atlantic Ocean touches the island's right and bottom edges.

Rainwater can flow to neighboring cells (North, South, East, and West) if the neighboring cell's height is less than or equal to the current cell's height. Water can also flow from any ocean-adjacent cell directly into the ocean.

Return a list of coordinates (r, c) where rainwater can flow to both the Pacific and Atlantic oceans.

Example 1:

Input: heights = [

[1,2,2,3,5],
[3,2,3,4,4],
[2,4,5,3,1],
[6,7,1,4,5],
[5,1,1,2,4]
]

Output: [[0,4],[1,3],[1,4],[2,2],[3,0],[3,1],[4,0]]

Example 2:

Input: heights = [[1]]

Output: [[0,0]]

2. Intuition

- The problem can be seen as a graph traversal problem where each cell represents a node.
- Water can only flow downhill or at the same level, meaning a cell can flow into another if its height is greater than or equal to the next cell.
- Instead of checking each cell individually, we start from ocean-adjacent cells and explore reachable areas using Depth-First Search (DFS).

3. Key Observations

- Water must be able to flow to both the Pacific and Atlantic oceans.
- A cell is valid if it can reach both oceans via DFS traversal.
- DFS can be used to mark the visited cells that can reach each ocean.
- Intersection of Pacific-reachable and Atlantic-reachable cells gives the final answer.

4. Approach

a. Initialize sets for tracking reachable cells:

- ***pacific:*** Stores cells that can reach the Pacific Ocean.
- ***atlantic:*** Stores cells that can reach the Atlantic Ocean.

b. Define a DFS function:

- Takes a row, column, a visited set, and the previous height.
- ***Stops if:***
  - The cell is out of bounds.
  - The cell is already visited.
  - The cell's height is less than the previous height (violating water flow condition).
- Otherwise, marks the cell as visited and explores all four possible directions.

c. Run DFS from all ocean-adjacent cells:

- ***Pacific Ocean:*** Start DFS from top row and left column.
- ***Atlantic Ocean:*** Start DFS from bottom row and right column.

d. Find the intersection of Pacific and Atlantic reachable cells:

- Convert to a list of coordinates and return the result.

5. Edge Cases

- ***Smallest Grid (1x1):*** Directly connected to both oceans.
- ***Increasing Heights:*** Water can flow directly from any cell.
- ***Decreasing Heights:*** Water may not be able to flow to both oceans.
- ***Large Grids (200x200):*** Algorithm should efficiently handle the worst case.

6. Complexity Analysis

Time Complexity:

- ***O(m × n):*** Each cell is visited at most twice (once for each ocean).

Space Complexity:

- ***O(m × n):*** Additional space for the visited sets and DFS recursion stack.

7. Alternative Approaches

a. Breadth-First Search (BFS)

- Instead of DFS, we can use BFS starting from ocean-adjacent cells.
- BFS processes all nodes layer by layer, ensuring optimal traversal.
- Pros: Avoids deep recursion issues.
- Cons: Uses additional memory for queue storage.

8. Test Cases

Test Case 1

```
heights = [
  [1,2,2,3,5],
  [3,2,3,4,4],
  [2,4,5,3,1],
  [6,7,1,4,5],
```

```
    [5,1,1,2,4]
]
sol = Solution()
print(sol.pacificAtlantic(heights))  # Expected: [[0,4],[1,3],[1,4],[2,2],[3,0],[3,1],[4,0]]

# Test Case 2
heights = [[1]]
print(sol.pacificAtlantic(heights))  # Expected: [[0,0]]
```

9. Final Thoughts

- This DFS-based solution effectively finds all cells that can reach both oceans.
- The algorithm is optimized for large grids (200x200) while maintaining clarity.
- BFS is an alternative approach but requires more memory.
- Understanding graph traversal techniques is crucial for solving similar problems.