Pacific Atlantic Water Flow

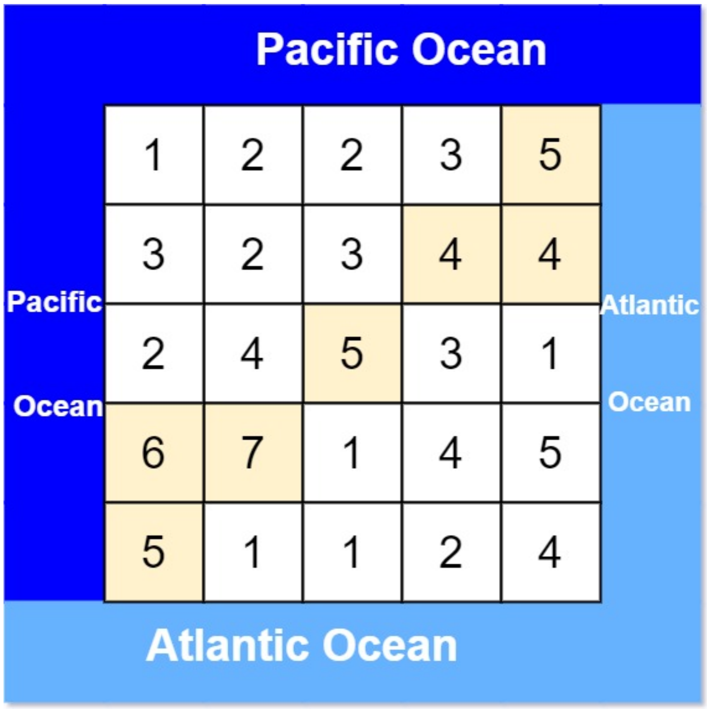
Difficulty	Medium
Category	Graph
Question	https://leetcode.com/problems/pacific-atlantic-water-flow/
Solution	https://youtu.be/s-VkcjHqkGI
Status	Done

Question

> There is an `m x n` rectangular island that borders both the **Pacific Ocean** and **Atlantic Ocean**. The **Pacific Ocean** touches the island's left and top edges, and the **Atlantic Ocean** touches the island's right and bottom edges.
>
> The island is partitioned into a grid of square cells. 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 island receives a lot of rain, and the rain water can flow to neighboring cells directly north, south, east, and west if the neighboring cell's height is **less than or equal to** the current cell's height. Water can flow from any cell adjacent to an ocean into the ocean.
>
> Return *a **2D list** of grid coordinates* `result` *where* `result[i] = [ri, ci]` *denotes that rain water can flow from cell* `(ri, ci)` *to **both** the Pacific and Atlantic oceans*.

Example

Example 1:



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

```
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]]
Explanation: The following cells can flow to the Pacific and Atlantic oceans, as shown below:
[0,4]: [0,4] -> Pacific Ocean
       [0,4] -> Atlantic Ocean
[1,3]: [1,3] -> [0,3] -> Pacific Ocean
       [1,3] -> [1,4] -> Atlantic Ocean
[1,4]: [1,4] -> [1,3] -> [0,3] -> Pacific Ocean
       [1,4] -> Atlantic Ocean
[2,2]: [2,2] -> [1,2] -> [0,2] -> Pacific Ocean
       [2,2] -> [2,3] -> [2,4] -> Atlantic Ocean
[3,0]: [3,0] -> Pacific Ocean
       [3,0] -> [4,0] -> Atlantic Ocean
[3,1]: [3,1] -> [3,0] -> Pacific Ocean
       [3,1] -> [4,1] -> Atlantic Ocean
[4,0]: [4,0] -> Pacific Ocean
       [4,0] -> Atlantic Ocean
Note that there are other possible paths for these cells to flow to the Pacific and Atlantic oceans.
```

Example 2:

```
Input: heights = [[1]]
Output: [[0,0]]
Explanation: The water can flow from the only cell to the Pacific and Atlantic oceans.
```

Idea

> 💡 DFS on each cell along Pacific and Atlanta. On each cell makes sure that each boundary cell is taken care of, instead of just starting from the corners. Record which cells can reach Pacific or Atlanta and find the intersection of them

Solution

```
class Solution:
    def pacificAtlantic(self, heights: List[List[int]]) -> List[List[int]]:
        if not heights:
            return []

        rows, cols = len(heights), len(heights[0])

        # Define two sets to store cells that can flow to the Pacific and Atlantic oceans.
        pacific_reachable = set()
        atlantic_reachable = set()

        # Define a DFS function to mark reachable cells.
        def dfs(r, c, ocean_reachable):
            # Mark the current cell as reachable.
            ocean_reachable.add((r, c))

            # Define the four possible directions: up, down, left, and right.
            directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]

            for dr, dc in directions:
                new_r, new_c = r + dr, c + dc

                # Check if the new cell is within the bounds of the grid.
                if 0 <= new_r < rows and 0 <= new_c < cols:
                    # Check if the new cell is not already marked as reachable and
                    # if its height is greater than or equal to the current cell's height.
                    if (new_r, new_c) not in ocean_reachable and heights[new_r][new_c] >= heights[r][c]:
                        # Recursively explore the new cell.
                        dfs(new_r, new_c, ocean_reachable)

        # Perform DFS from the left and top edges (Pacific Ocean).
        for r in range(rows):
            dfs(r, 0, pacific_reachable)
        for c in range(cols):
            dfs(0, c, pacific_reachable)

        # Perform DFS from the right and bottom edges (Atlantic Ocean).
        for r in range(rows):
            dfs(r, cols - 1, atlantic_reachable)
        for c in range(cols):
            dfs(rows - 1, c, atlantic_reachable)

        # Find the intersection of cells that can flow to both oceans.
        result = list(pacific_reachable.intersection(atlantic_reachable))
        return result
```