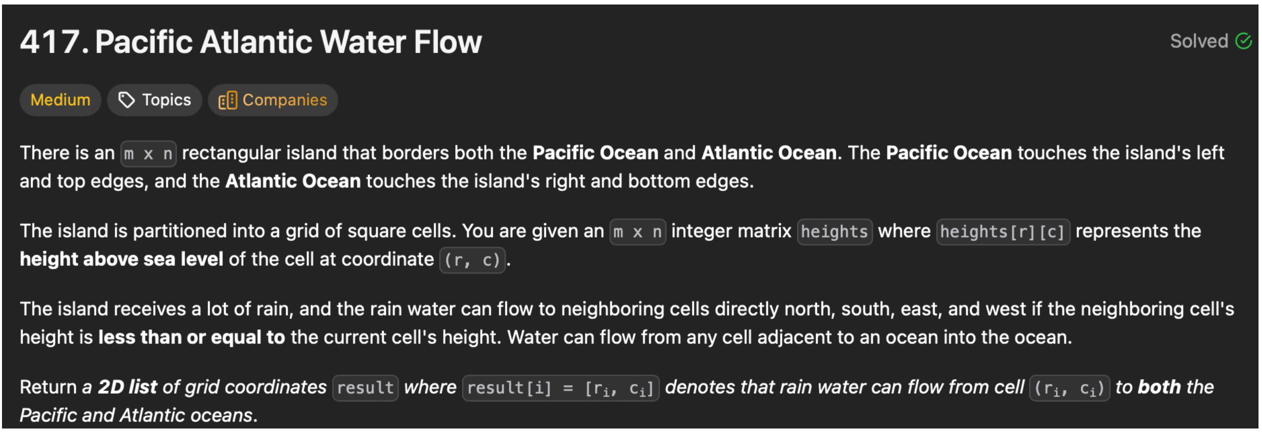


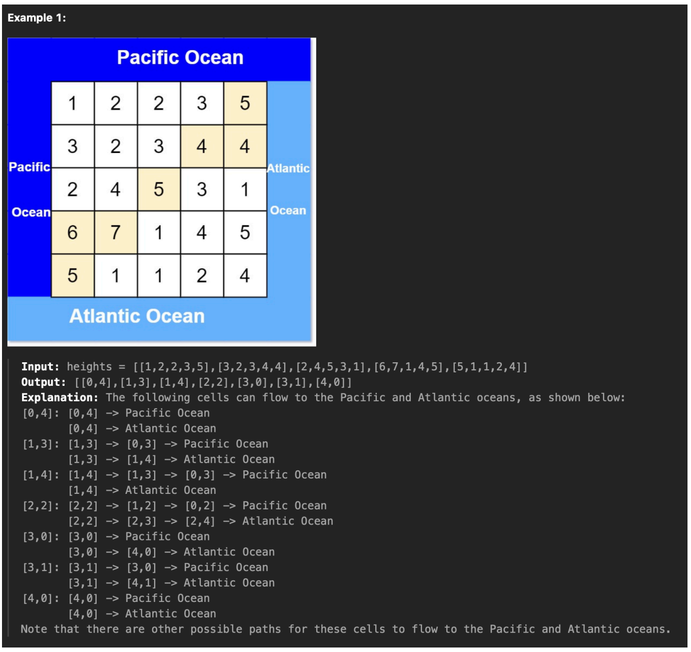
417. Pacific Atlantic Water Flow

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.

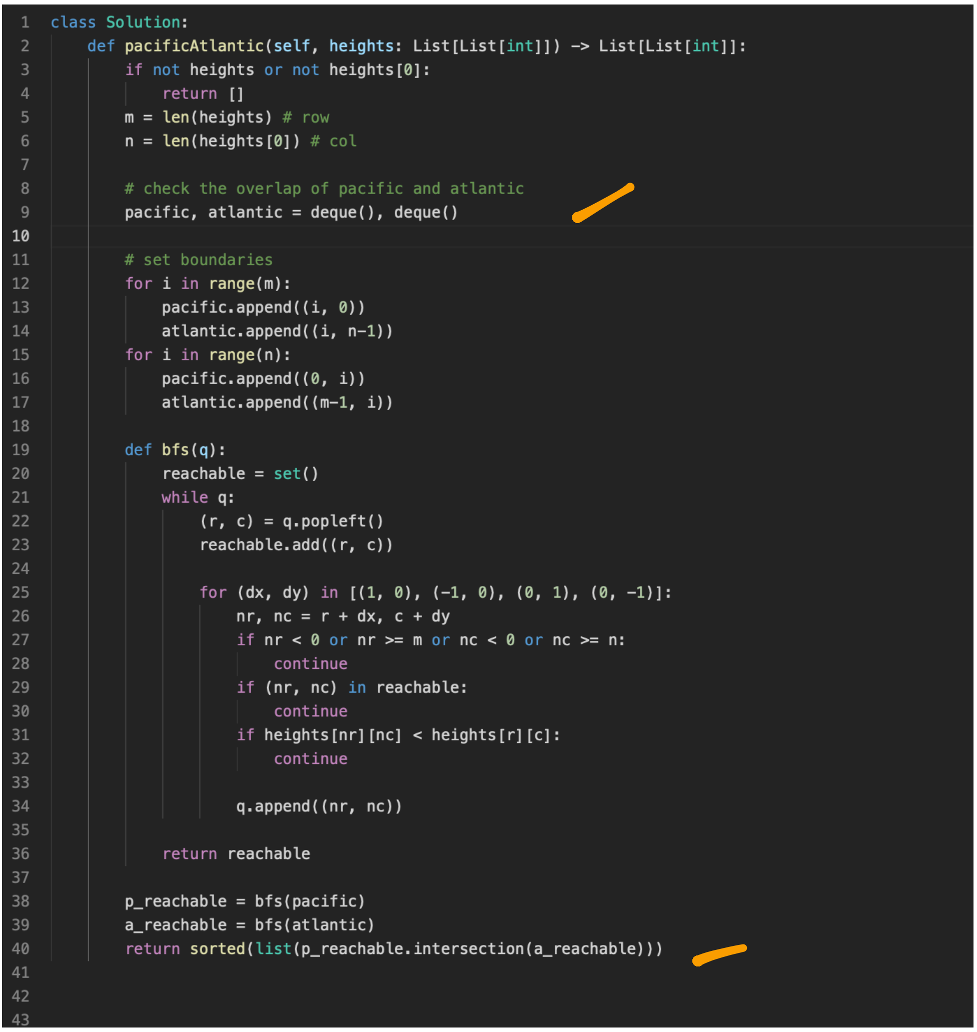
Return *a* ***2D list*** *of grid coordinates* `result` *where* `result[i] = [r_i, c_i]` *denotes that rain water can flow from cell* `(r_i, c_i)` *to* ***both*** *the Pacific and Atlantic oceans*.



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.
```



```python
class Solution:
    def pacificAtlantic(self, heights: List[List[int]]) -> List[List[int]]:
        if not heights or not heights[0]:
            return []
        m = len(heights) # row
        n = len(heights[0]) # col

        # check the overlap of pacific and atlantic
        pacific, atlantic = deque(), deque()

        # set boundaries
        for i in range(m):
            pacific.append((i, 0))
            atlantic.append((i, n-1))
        for i in range(n):
            pacific.append((0, i))
            atlantic.append((m-1, i))

        def bfs(q):
            reachable = set()
            while q:
                (r, c) = q.popleft()
                reachable.add((r, c))

                for (dx, dy) in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
                    nr, nc = r + dx, c + dy
                    if nr < 0 or nr >= m or nc < 0 or nc >= n:
                        continue
                    if (nr, nc) in reachable:
                        continue
                    if heights[nr][nc] < heights[r][c]:
                        continue

                    q.append((nr, nc))

            return reachable

        p_reachable = bfs(pacific)
        a_reachable = bfs(atlantic)
        return sorted(list(p_reachable.intersection(a_reachable)))
```