2503. Maximum Number of Points From Grid Queries

Hard | Topics | Companies | Hint

You are given an `m x n` integer matrix `grid` and an array `queries` of size `k`.

Find an array `answer` of size `k` such that for each integer `queries[i]` you start in the **top left** cell of the matrix and repeat the following process:

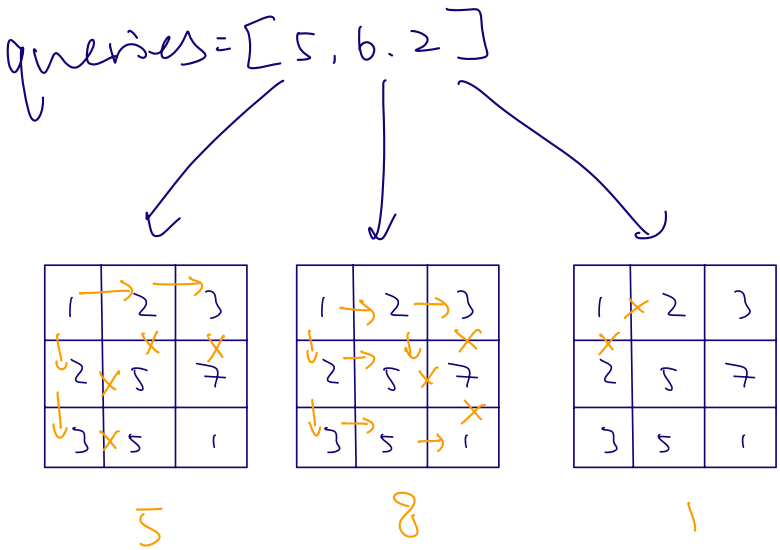
- If `queries[i]` is **strictly** greater than the value of the current cell that you are in, then you get one point if it is your first time visiting this cell, and you can move to any **adjacent** cell in all `4` directions: up, down, left, and right.
- Otherwise, you do not get any points, and you end this process.

After the process, `answer[i]` is the **maximum** number of points you can get. **Note** that for each query you are allowed to visit the same cell **multiple** times.

Return *the resulting array* `answer`.

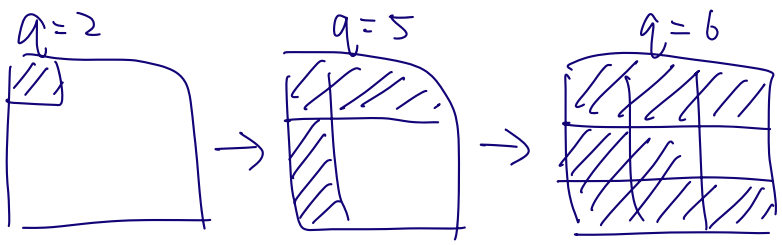
Example 1:

> **Input:** grid = [[1,2,3],[2,5,7],[3,5,1]], queries = [5,6,2]
> **Output:** [5,8,1]
> **Explanation:** The diagrams above show which cells we visit to get points for each query.



→ optimize, sort queries first.

it's like find the boundary of the grid.



→ min_heap

```python
class Solution:
    def maxPoints(self, grid: List[List[int]], queries: List[int]) -> List[int]:
        ROWS, COLS = len(grid), len(grid[0])

        q = [(n, i) for i, n in enumerate(queries)]
        q.sort()  by n.

        visited = set([(0,0)])   not set((0,0)) !
        mh = [(grid[0][0], 0, 0)]
        res = [0] * len(q)
        points = 0

        for qn, qi in q:
            while mh and mh[0][0] < qn:
                val, r, c = heapq.heappop(mh)
                points += 1
                neighbours = [[r-1, c], [r+1, c], [r, c-1], [r, c+1]]

                for nr, nc in neighbours:
                    if (0 <= nr < ROWS and 0 <= nc < COLS and (nr, nc) not in visited):
                        heapq.heappush(mh, (grid[nr][nc], nr, nc))
                        visited.add((nr, nc))

            res[qi] = points

        return res
```