305. Number of Islands II

Hard 1238 34 Add to List Share

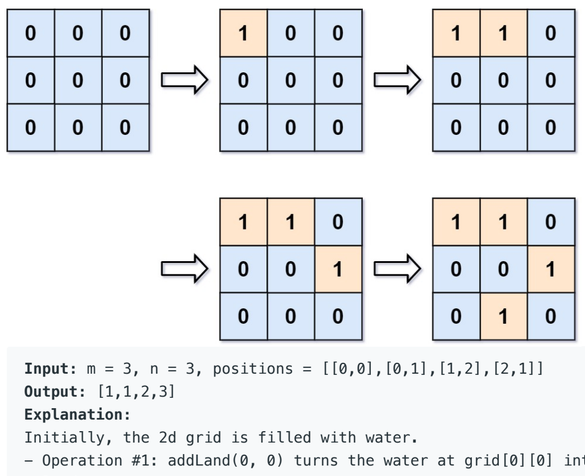
You are given an empty 2D binary grid `grid` of size `m x n`. The grid represents a map where `0`'s represent water and `1`'s represent land. Initially, all the cells of `grid` are water cells (i.e., all the cells are `0`'s).

We may perform an add land operation which turns the water at position into a land. You are given an array `positions` where `positions[i] = [r_i, c_i]` is the position `(r_i, c_i)` at which we should operate the `i^th` operation.

Return *an array of integers* `answer` *where* `answer[i]` *is the number of islands after turning the cell* `(r_i, c_i)` *into a land.*

An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:




```
Input: m = 3, n = 3, positions = [[0,0],[0,1],[1,2],[2,1]]
Output: [1,1,2,3]
Explanation:
Initially, the 2d grid is filled with water.
- Operation #1: addLand(0, 0) turns the water at grid[0][0] into a land. We have 1 island.
- Operation #2: addLand(0, 1) turns the water at grid[0][1] into a land. We still have 1 island.
- Operation #3: addLand(1, 2) turns the water at grid[1][2] into a land. We have 2 islands.
- Operation #4: addLand(2, 1) turns the water at grid[2][1] into a land. We have 3 islands.
```

Example 2:

```
Input: m = 1, n = 1, positions = [[0,0]]
Output: [1]
```

Constraints:

- `1 <= m, n, positions.length <= 10^4`
- `1 <= m * n <= 10^4`
- `positions[i].length == 2`
- `0 <= r_i < m`
- `0 <= c_i < n`

Follow up: Could you solve it in time complexity `O(k log(mn))`, where `k == positions.length`?

Accepted 101,941 | Submissions 259,229

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Google | 6 Uber | 3 Apple | 2

Related Topics

Array Union Find

Similar Questions

Number of Islands	Medium
Process Restricted Friend Requests	Hard

Java Autocomplete

```java
class Solution {
  public List<Integer> numIslands2(int m, int n, int[][] positions) {
    List<Integer> ans = new ArrayList<>();
    HashMap<Integer, Integer> land2id = new HashMap<Integer, Integer>();
    int num_islands = 0;
    int island_id = 0;

    for (int[] pos : positions) {
      int r = pos[0], c = pos[1];
      Set<Integer> overlap = new HashSet<Integer>();

      if (r - 1 >= 0 && land2id.containsKey((r-1) * n + c)) {
        overlap.add(land2id.get((r-1) * n + c));
      }
      if (r + 1 < m && land2id.containsKey((r+1) * n + c)) {
        overlap.add(land2id.get((r+1) * n + c));
      }
      if (c - 1 >= 0 && land2id.containsKey(r * n + c - 1)) {
        overlap.add(land2id.get(r * n + c - 1));
      }
      if (c + 1 < n && land2id.containsKey(r * n + c + 1)) {
        overlap.add(land2id.get(r * n + c + 1));
      }

      if (overlap.isEmpty()) {
        ++num_islands;
        land2id.put(r * n + c, island_id++);
      } else if (overlap.size() == 1) {
        land2id.put(r * n + c, overlap.iterator().next());
      } else {
        int root_id = overlap.iterator().next();
        for (Map.Entry<Integer, Integer> entry : land2id.entrySet()) {
          int k = entry.getKey();
          int id = entry.getValue();
          if (overlap.contains(id)) {
            land2id.put(k, root_id);
          }
        }
        land2id.put(r * n + c, root_id);
        num_islands -= (overlap.size() - 1);
      }
      ans.add(num_islands);
    }

    return ans;
  }
}
```