zxi.mytechroad.com

花花酱 LeetCode 1001. Grid Illumination – Huahua's Tech Road

zxi

3-4 minutes

On a `N x N` grid of cells, each cell `(x, y)` with `0 <= x < N` and `0 <= y < N` has a lamp.

Initially, some number of lamps are on. `lamps[i]` tells us the location of the `i`-th lamp that is on. Each lamp that is on illuminates every square on its x-axis, y-axis, and both diagonals (similar to a Queen in chess).

For the i-th query `queries[i] = (x, y)`, the answer to the query is 1 if the cell (x, y) is illuminated, else 0.

After each query `(x, y)` [in the order given by `queries`], we turn off any lamps that are at cell `(x, y)` or are adjacent 8-directionally (ie., share a corner or edge with cell `(x, y)`.)

Return an array of answers. Each value `answer[i]` should be equal to the answer of the `i`-th query `queries[i]`.

Example 1:

Input: N = 5, lamps = [[0,0],[4,4]], queries = [[1,1],[1,0]]
Output: [1,0]
Explanation:

Before performing the first query we have both lamps [0,0] and [4,4] on.
The grid representing which cells are lit looks like this, where [0,0] is the top left corner, and [4,4] is the bottom right corner:
1 1 1 1 1
1 1 0 0 1
1 0 1 0 1
1 0 0 1 1
1 1 1 1 1
Then the query at [1, 1] returns 1 because the cell is lit. After this query, the lamp at [0, 0] turns off, and the grid now looks like this:
1 0 0 0 1
0 1 0 0 1
0 0 1 0 1
0 0 0 1 1
1 1 1 1 1
Before performing the second query we have only the lamp [4,4] on. Now the query at [1,0] returns 0, because the cell is no longer lit.

Note:

1. `1 <= N <= 10^9`
2. `0 <= lamps.length <= 20000`
3. `0 <= queries.length <= 20000`
4. `lamps[i].length == queries[i].length == 2`

Solution: HashTable

use lx, ly, lp, lq to track the # of lamps that covers each row, col, diagonal, antidiagonal

Time complexity: O(|L| + |Q|)
Space complexity: O(|L|)

- C++
- C++ v2

C++

```
// Author: Huahua, running time: 460 ms, 82.2 MB
class Solution {
public:
  vector<int> gridIllumination(int N, vector<vector<int>>& lamps, vector<vector<int>>& queries) {
    unordered_set<long> s;
    unordered_map<int, int> lx, ly, lp, lq;
    for (const auto& lamp : lamps) {
      const int x = lamp[0];
      const int y = lamp[1];
      s.insert(static_cast<long>(x) << 32 | y);
      ++lx[x];
      ++ly[y];
      ++lp[x + y];
      ++lq[x - y];
    }
    vector<int> ans;
    for (const auto& query : queries) {
      const int x = query[0];
      const int y = query[1];
      if (lx.count(x) || ly.count(y) || lp.count(x + y) || lq.count(x - y)) {
        ans.push_back(1);
        for (int tx = x - 1; tx <= x + 1; ++tx)
          for (int ty = y - 1; ty <= y + 1; ++ty) {
            if (tx < 0 || tx >= N || ty < 0 || ty >= N) continue;
            const long key = static_cast<long>(tx) << 32 | ty;
            if (!s.count(key)) continue;
            s.erase(key);
            if (--lx[tx] == 0) lx.erase(tx);
            if (--ly[ty] == 0) ly.erase(ty);
            if (--lp[tx + ty] == 0) lp.erase(tx + ty);
            if (--lq[tx - ty] == 0) lq.erase(tx - ty);
          }
      } else {
        ans.push_back(0);
      }
    }
    return ans;
```

```
  }
};
```

C++ v2