AlgoExpert   Quad Layout   C++   12px   Sublime   Monokai   00:00:00

Prompt   Scratchpad   Our Solution(s)   Video Explanation   Run Code

Solution 1   Solution 2   Solution 3

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
#include <unordered_map>
using namespace std;

struct Point {
  int x;
  int y;
};

string UP = "up";
string RIGHT = "right";
string DOWN = "down";

unordered_map<string, unordered_map<int, vector<Point>>>
getCoordsTable(vector<Point> coords);
int getRectangleCount(
    vector<Point> coords,
    unordered_map<string, unordered_map<int, vector<Point>>> coordsTable);
int clockwiseCountRectangles(
    Point coord1,
    unordered_map<string, unordered_map<int, vector<Point>>> coordsTable,
    string direction, int lowerLeftY);

// O(n^2) time | O(n) space - where n is the number of coordinates
int rectangleMania(vector<Point> coords) {
  unordered_map<string, unordered_map<int, vector<Point>>> coordsTable =
      getCoordsTable(coords);
  return getRectangleCount(coords, coordsTable);
}

unordered_map<string, unordered_map<int, vector<Point>>>
getCoordsTable(vector<Point> coords) {
  unordered_map<string, unordered_map<int, vector<Point>>> coordsTable;
  coordsTable.insert({"x", unordered_map<int, vector<Point>>{}});
  coordsTable.insert({"y", unordered_map<int, vector<Point>>{}});
  for (Point coord : coords) {
    if (coordsTable["x"].find(coord.x) == coordsTable["x"].end()) {
      coordsTable["x"].insert({coord.x, vector<Point>{}});
    }
    if (coordsTable["y"].find(coord.y) == coordsTable["y"].end()) {
      coordsTable["y"].insert({coord.y, vector<Point>{}});
    }
    coordsTable["x"][coord.x].push_back(coord);
    coordsTable["y"][coord.y].push_back(coord);
  }
  return coordsTable;
}

int getRectangleCount(
    vector<Point> coords,
    unordered_map<string, unordered_map<int, vector<Point>>> coordsTable) {
  int rectangleCount = 0;
  for (Point coord : coords) {
    int lowerLeftY = coord.y;
    rectangleCount +=
        clockwiseCountRectangles(coord, coordsTable, UP, lowerLeftY);
  }
  return rectangleCount;
}

int clockwiseCountRectangles(
    Point coord1,
    unordered_map<string, unordered_map<int, vector<Point>>> coordsTable,
    string direction, int lowerLeftY) {
  if (direction == DOWN) {
    vector<Point> relevantCoords = coordsTable["x"][coord1.x];
    for (Point coord2 : relevantCoords) {
      int lowerRightY = coord2.y;
      if (lowerRightY == lowerLeftY)
        return 1;
    }
    return 0;
  } else {
    int rectangleCount = 0;
    if (direction == UP) {
      vector<Point> relevantCoords = coordsTable["x"][coord1.x];
      for (Point coord2 : relevantCoords) {
        bool isAbove = coord2.y > coord1.y;
        if (isAbove)
          rectangleCount +=
              clockwiseCountRectangles(coord2, coordsTable, RIGHT, lowerLeftY);
      }
    } else if (direction == RIGHT) {
      vector<Point> relevantCoords = coordsTable["y"][coord1.y];
      for (Point coord2 : relevantCoords) {
        bool isRight = coord2.x > coord1.x;
        if (isRight)
          rectangleCount +=
              clockwiseCountRectangles(coord2, coordsTable, DOWN, lowerLeftY);
      }
    }
    return rectangleCount;
  }
}
```