

Solution 1Solution 2Solution 3

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 #include <vector>
4 #include <unordered_map>
5 #include <algorithm>
6 using namespace std;
7
8 struct Point {
9     int x;
10    int y;
11
12    bool operator==(Point point2) { return x == point2.x && y == point2.y; }
13 };
14
15 string UP = "up";
16 string RIGHT = "right";
17 string DOWN = "down";
18 string LEFT = "left";
19
20 unordered_map<string, unordered_map<string, vector<Point>>>
21 getCoordsTable(vector<Point> coords);
22 string getCoordDirection(Point coord1, Point coord2);
23 int getRectangleCount(
24     vector<Point> coords,
25     unordered_map<string, unordered_map<string, vector<Point>>> coordsTable);
26 int clockwiseCountRectangles(
27     Point coord,
28     unordered_map<string, unordered_map<string, vector<Point>>> coordsTable,
29     string direction, Point origin);
30 string getNextClockwiseDirection(string direction);
31 string coordToString(Point coord);
32
33 // O(n^2) time | O(n^2) space - where n is the number of coordinates
34 int rectangleMania(vector<Point> coords) {
35     unordered_map<string, unordered_map<string, vector<Point>>> coordsTable =
36         getCoordsTable(coords);
37     return getRectangleCount(coords, coordsTable);
38 }
39
40 unordered_map<string, unordered_map<string, vector<Point>>>
41 getCoordsTable(vector<Point> coords) {
42     unordered_map<string, unordered_map<string, vector<Point>>> coordsTable;
43     for (Point coord1 : coords) {
44         unordered_map<string, vector<Point>> coord1Directions({
45             {UP, vector<Point>{}},
46             {RIGHT, vector<Point>{}},
47             {DOWN, vector<Point>{}},
48             {LEFT, vector<Point>{}},
49         });
50         for (Point coord2 : coords) {
51             string coord2Direction = getCoordDirection(coord1, coord2);
52             if (coord1Directions.find(coord2Direction) != coord1Directions.end()) {
53                 coord1Directions[coord2Direction].push_back(coord2);
54             }
55         }
56         string coord1String = coordToString(coord1);
57         coordsTable.insert({coord1String, coord1Directions});
58     }
59     return coordsTable;
60 }
61
62 string getCoordDirection(Point coord1, Point coord2) {
63     if (coord2.y == coord1.y) {
64         if (coord2.x > coord1.x) {
65             return RIGHT;
66         } else if (coord2.x < coord1.x) {
67             return LEFT;
68         }
69     } else if (coord2.x == coord1.x) {
70         if (coord2.y > coord1.y) {
71             return UP;
72         } else if (coord2.y < coord1.y) {
73             return DOWN;
74         }
75     }
76     return "";
77 }
78
79 int getRectangleCount(
80     vector<Point> coords,
81     unordered_map<string, unordered_map<string, vector<Point>>> coordsTable) {
82     int rectangleCount = 0;
83     for (Point coord : coords) {
84         rectangleCount += clockwiseCountRectangles(coord, coordsTable, UP, coord);
85     }
86     return rectangleCount;
87 }
88
89 int clockwiseCountRectangles(
90     Point coord,
91     unordered_map<string, unordered_map<string, vector<Point>>> coordsTable,
92     string direction, Point origin) {
93     string coordString = coordToString(coord);
94     if (direction == LEFT) {
95         bool rectangleFound = find(coordsTable[coordString][LEFT].begin(),
96                                   coordsTable[coordString][LEFT].end(),
97                                   origin) != coordsTable[coordString][LEFT].end();
98         return rectangleFound ? 1 : 0;
99     } else {
100         int rectangleCount = 0;
101         string nextDirection = getNextClockwiseDirection(direction);
102         for (Point nextCoord : coordsTable[coordString][direction]) {
103             rectangleCount += clockwiseCountRectangles(nextCoord, coordsTable,
104                                                         nextDirection, origin);
105         }
106         return rectangleCount;
107     }
108 }
109
110 string getNextClockwiseDirection(string direction) {
111     if (direction == UP)
112         return RIGHT;
113     if (direction == RIGHT)
114         return DOWN;
115     if (direction == DOWN)
```

```
116     return LEFT;
117     return "";
118 }
119
120 string coordToString(Point coord) {
121     return to_string(coord.x) + "-" + to_string(coord.y);
122 }
123
```