5877. Detect Squares

My Submissions (/contest/weekly-contest-259/problems/detect-squares/submissions/) Back to Contest (/contest/weekly-contest-259/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

You are given a stream of points on the X-Y plane. Design an algorithm that:

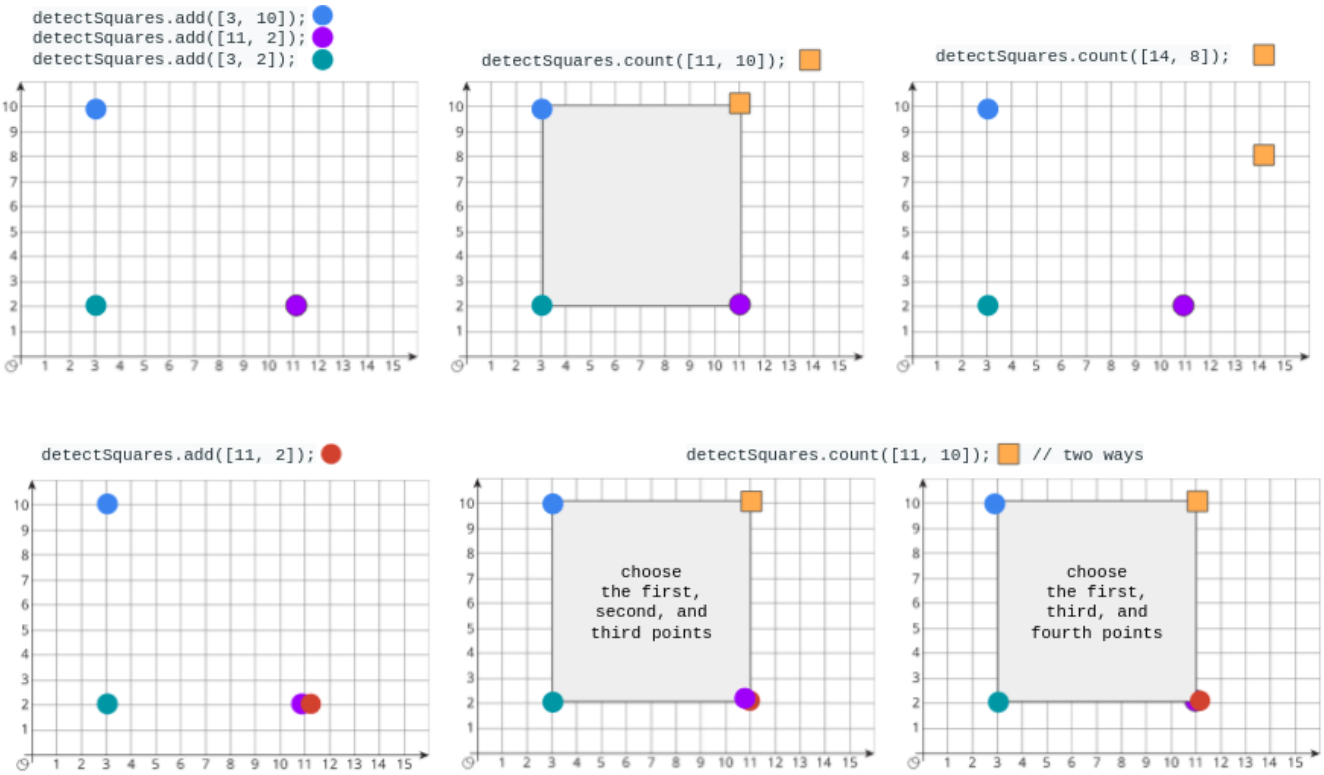
- **Adds** new points from the stream into a data structure. **Duplicate** points are allowed and should be treated as different points.
- Given a query point, **counts** the number of ways to choose three points from the data structure such that the three points and the query point form an **axis-aligned square** with **positive area**.

An **axis-aligned square** is a square whose edges are all the same length and are either parallel or perpendicular to the x-axis and y-axis.

Implement the `DetectSquares` class:

- `DetectSquares()` Initializes the object with an empty data structure.
- `void add(int[] point)` Adds a new point `point = [x, y]` to the data structure.
- `int count(int[] point)` Counts the number of ways to form **axis-aligned squares** with point `point = [x, y]` as described above.

Example 1:



```
Input
["DetectSquares", "add", "add", "add", "count", "count", "add", "count"]
[[], [[3, 10]], [[11, 2]], [[3, 2]], [[11, 10]], [[14, 8]], [[11, 2]], [[11, 10]]]
Output
[null, null, null, null, 1, 0, null, 2]

Explanation
DetectSquares detectSquares = new DetectSquares();
detectSquares.add([3, 10]);
detectSquares.add([11, 2]);
detectSquares.add([3, 2]);
detectSquares.count([11, 10]); // return 1. You can choose:
                               //   - The first, second, and third points
detectSquares.count([14, 8]);  // return 0. The query point cannot form a square with any points in the data st
detectSquares.add([11, 2]);    // Adding duplicate points is allowed.
detectSquares.count([11, 10]); // return 2. You can choose:
                               //   - The first, second, and third points
                               //   - The first, third, and fourth points
```

Constraints:

- `point.length == 2`
- `0 <= x, y <= 1000`
- At most `5000` calls **in total** will be made to `add` and `count`.

JavaScript

```
const mx = Math.max;
const mi = Math.min;
function DetectSquares() {
    let m = new Map();
    let lx = ly = Number.MIN_SAFE_INTEGER;
    let sx = sy = Number.MAX_SAFE_INTEGER;
    return { add, count }
    function add(p) {
        let ke = p[0] + " " + p[1];
        m.set(ke, m.get(ke) + 1 || 1);
        lx = mx(lx, p[0]);
        ly = mx(ly, p[1]);
        sx = mi(sx, p[0]);
        sy = mi(sy, p[1]);
        // pr(m);
    }

    function count(p) {
        if (m.size == 0) return 0;
        let [x, y] = p;
        let res = 0;
        for (let t = 1; x - t >= sx && y - t >= sy; t++) { // downLeft
            let left = (x - t) + " " + y;
            let down = x + " " + (y - t);
            let dia = (x - t) + " " + (y - t);
            // pr(left, down, dia)
            if (m.has(left) && m.has(down) && m.has(dia)) {
                res += m.get(left) * m.get(down) * m.get(dia);
            }
        }
        for (let t = 1; x + t <= lx && y + t <= ly; t++) { // topRight
            let right = (x + t) + " " + y;
            let up = x + " " + (y + t);
            let dia = (x + t) + " " + (y + t);
```

```
            if (m.has(right) && m.has(up) && m.has(dia)) {
                res += m.get(right) * m.get(up) * m.get(dia);
            }
        }
        for (let t = 1; x + t <= lx && y - t >= sy; t++) { // downRight
            let right = (x + t) + " " + y;
            let down = x + " " + (y - t);
            let dia = (x + t) + " " + (y - t);
            if (m.has(right) && m.has(down) && m.has(dia)) {
                res += m.get(right) * m.get(down) * m.get(dia);
            }
        }
        for (let t = 1; x - t >= sx && y + t <= ly; t++) { // topLeft
            let left = (x - t) + " " + y;
            let up = x + " " + (y + t);
            let dia = (x - t) + " " + (y + t);
            if (m.has(left) && m.has(up) && m.has(dia)) {
                res += m.get(left) * m.get(up) * m.get(dia);
            }
        }
        return res;
    }
}
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/557321466/) | More Details > (/submissions/detail/557321466/)

Share your acceptance!

1