3625. Count Number of Trapezoids II

Solved

Hard | Topics | Companies | Hint

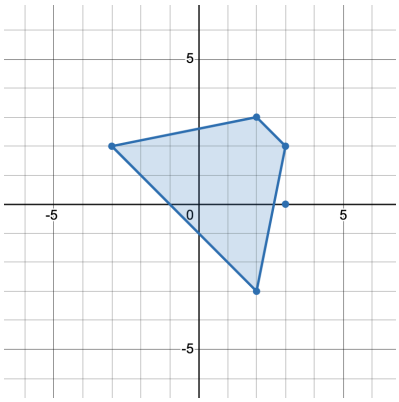
You are given a 2D integer array `points` where `points[i] = [x_i, y_i]` represents the coordinates of the `i^th` point on the Cartesian plane.

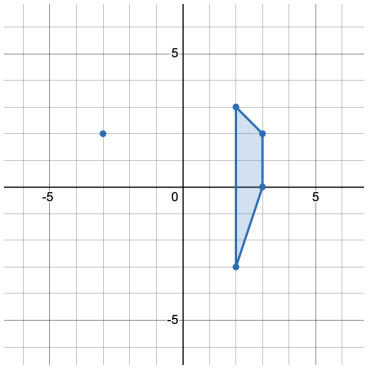
Return *the number of unique trapezoids* that can be formed by choosing any four distinct points from `points`.

A **trapezoid** is a convex quadrilateral with **at least one pair** of parallel sides. Two lines are parallel if and only if they have the same slope.

Example 1:

> **Input:** `points = [[-3,2],[3,0],[2,3],[3,2],[2,-3]]`
>
> **Output:** 2
>
> **Explanation:**

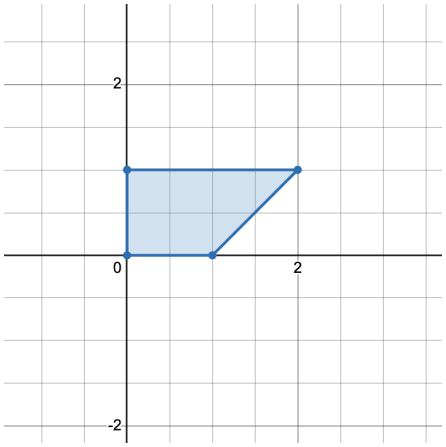




> There are two distinct ways to pick four points that form a trapezoid:
>
> - The points `[-3,2], [2,3], [3,2], [2,-3]` form one trapezoid.
> - The points `[2,3], [3,2], [3,0], [2,-3]` form another trapezoid.

Example 2:

> **Input:** `points = [[0,0],[1,0],[0,1],[2,1]]`
>
> **Output:** 1
>
> **Explanation:**



> There is only one trapezoid which can be formed.

Constraints:

- `4 <= points.length <= 500`
- `-1000 <= xi, yi <= 1000`
- All points are pairwise distinct.

Python:

class Solution:

```
    def countTrapezoids(self, A: List[List[int]]) -> int:
        slopes = Counter()
        lines = Counter()
        mids = Counter()
        midlines = Counter()

        for (x1, y1), (x2, y2) in combinations(A, 2):
            dx, dy = x2 - x1, y2 - y1
            g = gcd(dx, dy)
            dx, dy = dx // g, dy // g
            if dx < 0 or (dx == 0 and dy < 0):
                dx, dy = -dx, -dy

            inter = dx * y1 - dy * x1
            slopes[dx, dy] += 1
            lines[dx, dy, inter] += 1
            mids[x1 + x2, y1 + y2] += 1
            midlines[x1 + x2, y1 + y2, dx, dy, inter] += 1

        ans = sum(comb(v, 2) for v in slopes.values())
        ans -= sum(comb(v, 2) for v in lines.values())
        ans -= sum(comb(v, 2) for v in mids.values())
        ans += sum(comb(v, 2) for v in midlines.values())
        return ans
```

JavaScript:

```
/**
 * @param {number[][]} points
 * @return {number}
 */
var countTrapezoids = function(points) {

    const t = new Map();
    const v = new Map();
    const n = points.length;

    function add(map, key, des) {
        if (!map.has(key)) map.set(key, new Map());
        const inner = map.get(key);
        inner.set(des, (inner.get(des) || 0) + 1);
    }

    for (let i = 0; i < n; i++) {
        let [x1, y1] = points[i];
```

```
        for (let j = i + 1; j < n; j++) {

            let [x2, y2] = points[j];
            let dx = x2 - x1;
            let dy = y2 - y1;

            if (dx < 0 || (dx === 0 && dy < 0)) {
                dx = -dx;
                dy = -dy;
            }

            let g = gcd(dx, Math.abs(dy));
            let sx = dx / g;
            let sy = dy / g;

            let des = sx * y1 - sy * x1;

            let key1 = (sx << 12) | (sy + 2000);
            let key2 = (dx << 12) | (dy + 2000);

            add(t, key1, des);
            add(v, key2, des);
        }
    }

    return count(t) - Math.floor(count(v) / 2);
};

function gcd(a, b) {
    a = Math.abs(a);
    b = Math.abs(b);
    while (b !== 0) {
        let tmp = a % b;
        a = b;
        b = tmp;
    }
    return a;
}

function count(map) {
    let ans = 0;

    for (let inner of map.values()) {
        let total = 0;
```

```
        for (let val of inner.values()) total += val;

        let rem = total;
        for (let val of inner.values()) {
            rem -= val;
            ans += val * rem;
        }
    }
    return ans;
}
```

Java:

```
class Solution {

    public int gcd(int a, int b) {
        while(b != 0) {
            int tmp = a % b;
            a = b;
            b = tmp;
        }
        return a;
    }

    public String hash(int a, int b) {
        int g = gcd(Math.abs(a), Math.abs(b));
        if(g == 0) {
            return "0/0";
        }

        int num = a / g;
        int den = b / g;
        return (num * den < 0 && den != 0 ? "-" : "") + Math.abs(num) + "/" + Math.abs(den);
    }

    public int countTrapezoids(int[][] points) {
        // 1. Trapezium contains two pairs of opposite sides
        // 2. Atleast one pair should be parallel
        // 3. A Trapezium will be counted twice if it has two parallel pairs
        // 4. So parallelogram will be counted twice.
        // 5. So, Result = Trapeziums - Parallelograms
        // 6. Intercept = y1 - m * x1 = (y1 * (x2 - x1) - (y2 - y1) * x) / (x2 - x1)


        int trapeziums_parallelograms = 0;
```

```
        HashMap<String, Integer> parallel_lines = new HashMap<>();
        HashMap<String, Integer> collinear_lines = new HashMap<>();

        int n = points.length;

        for(int i = 0; i < n; i++) {
            int[] p2 = points[i];
            for(int j = 0; j < i; j++) {
                int[] p1 = points[j];

                String slope = p1[0] != p2[0] ? hash(p2[1] - p1[1], p2[0] - p1[0]) : "infinity";
                String intercept = p1[0] != p2[0] ? hash(p1[1] * (p2[0] - p1[0]) - (p2[1] - p1[1]) * p1[0],
p2[0] - p1[0]) : p1[0] + "";
                String h = slope + "," + intercept;

                int seen_parallel_lines = parallel_lines.getOrDefault(slope, 0);
                int seen_collinear_lines = collinear_lines.getOrDefault(h, 0);

                trapeziums_parallelograms += seen_parallel_lines - seen_collinear_lines;

                parallel_lines.put(slope, parallel_lines.getOrDefault(slope, 0) + 1);
                collinear_lines.put(h, collinear_lines.getOrDefault(h, 0) + 1);
            }
        }

        int parallelograms = 0;
        HashMap<String, Integer> parallel_lines_dist = new HashMap<>();
        HashMap<String, Integer> collinear_lines_dist = new HashMap<>();

        for(int i = 0; i < n; i++) {
            int[] p2 = points[i];
            for(int j = 0; j < i; j++) {
                int[] p1 = points[j];

                String slope = p1[0] != p2[0] ? hash(p2[1] - p1[1], p2[0] - p1[0]) : "infinity";
                String intercept = p1[0] != p2[0] ? hash(p1[1] * (p2[0] - p1[0]) - (p2[1] - p1[1]) * p1[0],
p2[0] - p1[0]) : p1[0] + "";
                int dist = (p1[0] - p2[0]) * (p1[0] - p2[0]) + (p1[1] - p2[1]) * (p1[1] - p2[1]);

                String h1 = slope + "," + dist;
                String h2 = slope + "," + intercept + "," + dist;

                int seen_parallel_lines_dist = parallel_lines_dist.getOrDefault(h1, 0);
                int seen_collinear_lines_dist = collinear_lines_dist.getOrDefault(h2, 0);
```

```
                parallelograms += seen_parallel_lines_dist - seen_collinear_lines_dist;

                parallel_lines_dist.put(h1, parallel_lines_dist.getOrDefault(h1, 0) + 1);
                collinear_lines_dist.put(h2, collinear_lines_dist.getOrDefault(h2, 0) + 1);

            }
        }

        // Again in above loop each parallelogram will be read twice

        return trapeziums_parallelograms - parallelograms / 2;
    }
}
```