# 3531. Count Covered Buildings

Solved

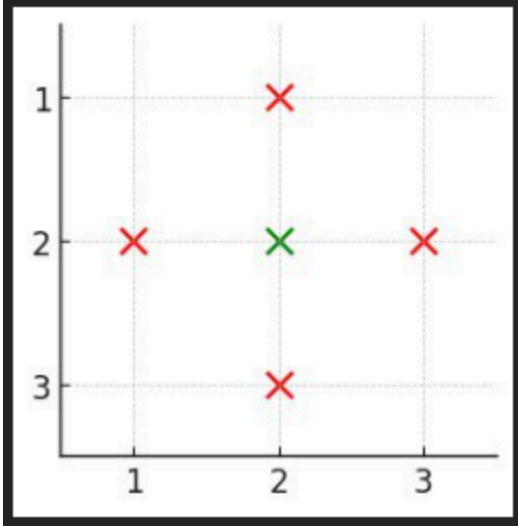
Medium | Topics | Companies | Hint

You are given a positive integer `n`, representing an `n x n` city. You are also given a 2D grid `buildings`, where `buildings[i] = [x, y]` denotes a **unique** building located at coordinates `[x, y]`.

A building is **covered** if there is at least one building in all **four** directions: left, right, above, and below.

Return the number of **covered** buildings.

**Example 1:**



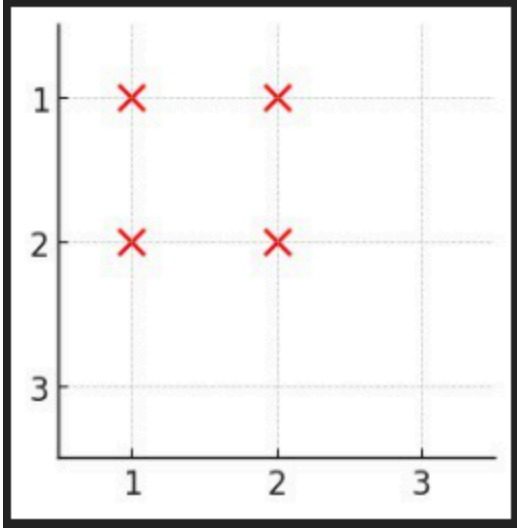

**Input:** `n = 3, buildings = [[1,2],[2,2],[3,2],[2,1],[2,3]]`

**Output:** `1`

**Explanation:**

- Only building `[2,2]` is covered as it has at least one building:
  - above (`[1,2]`)
  - below (`[3,2]`)
  - left (`[2,1]`)
  - right (`[2,3]`)
- Thus, the count of covered buildings is 1.

**Example 2:**



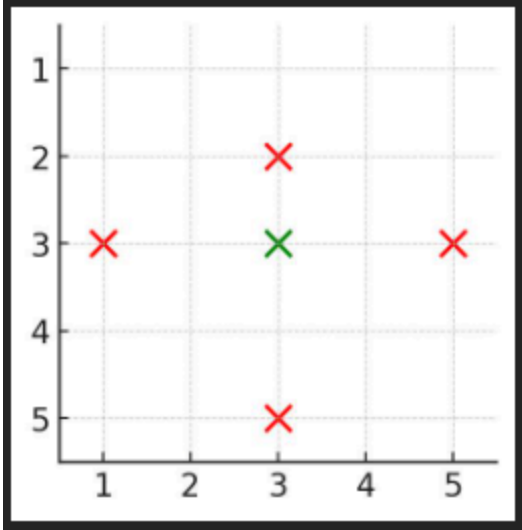

**Input:** `n = 3, buildings = [[1,1],[1,2],[2,1],[2,2]]`

**Output:** `0`

**Explanation:**

- No building has at least one building in all four directions.

**Example 3:**

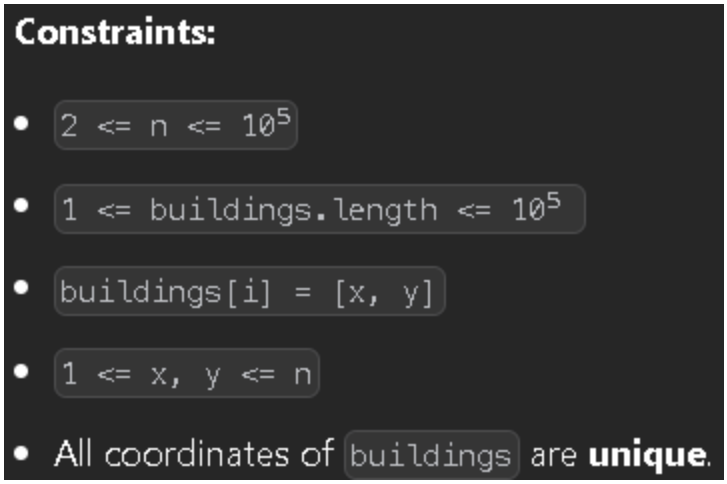



**Input:** `n = 5, buildings = [[1,3],[3,2],[3,3],[3,5],[5,3]]`

**Output:** 1

**Explanation:**

- Only building `[3,3]` is covered as it has at least one building:
  - above (`[1,3]`)
  - below (`[5,3]`)
  - left (`[3,2]`)
  - right (`[3,5]`)
- Thus, the count of covered buildings is 1.




# Python:

```
class Solution:
    def countCoveredBuildings(self, n: int, buildings: List[List[int]]) -> int:
        cm = [n + 1] * (n + 1)   # col_min_y: smallest y in column x
        cM = [0]     * (n + 1)   # col_max_y: largest  y in column x
        rm = [n + 1] * (n + 1)   # row_min_x: smallest x in row    y
        rM = [0]     * (n + 1)   # row_max_x: largest  x in row    y

        # First pass: compute extremes
        for x, y in buildings:
            cm[x] = min(cm[x], y)
            cM[x] = max(cM[x], y)
            rm[y] = min(rm[y], x)
            rM[y] = max(rM[y], x)

        # Second pass: count covered
        covered = 0
        for x, y in buildings:
            if cm[x] < y < cM[x] and rm[y] < x < rM[y]:
                covered += 1

        return covered
```

# JavaScript:

```
/**
 * @param {number} n
 * @param {number[][]} buildings
 * @return {number}
 */
var countCoveredBuildings = function(n, buildings) {
    const rows = new Map();
```

```
    const cols = new Map();

    // Store all buildings row-wise and column-wise
    for (const [x, y] of buildings) {
        if (!rows.has(x)) rows.set(x, []);
        if (!cols.has(y)) cols.set(y, []);
        rows.get(x).push(y);
        cols.get(y).push(x);
    }

    // Sort each row and column list
    for (const y_list of rows.values()) {
        y_list.sort((a, b) => a - b);
    }
    for (const x_list of cols.values()) {
        x_list.sort((a, b) => a - b);
    }

    let count = 0;

    // Check each building
    for (const [x, y] of buildings) {
        const row = rows.get(x);
        const col = cols.get(y);

        // Find the position of the current building in row and column using binary search
        const idx_row = lowerBound(row, y);
        const idx_col = lowerBound(col, x);

        const left = idx_row > 0;
        const right = idx_row < row.length - 1;
        const above = idx_col > 0;
        const below = idx_col < col.length - 1;

        // If there are buildings on all four sides, increment the counter
        if (left && right && above && below) {
            count++;
        }
    }

    return count;
};

// Helper function for lower_bound (binary search)
```

```
function lowerBound(arr, target) {
    let left = 0, right = arr.length;
    while (left < right) {
        const mid = Math.floor((left + right) / 2);
        if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid;
        }
    }
    return left;
}
```

# Java:

```
class Solution {
    public int countCoveredBuildings(int n, int[][] buildings) {
        Map<Integer, int[]> yRangeGivenX = new HashMap<>();
        Map<Integer, int[]> xRangeGivenY = new HashMap<>();

        for (int[] b : buildings) {
            int x = b[0], y = b[1];
            yRangeGivenX.putIfAbsent(x, new int[]{Integer.MAX_VALUE, Integer.MIN_VALUE});
            yRangeGivenX.get(x)[0] = Math.min(yRangeGivenX.get(x)[0], y);
            yRangeGivenX.get(x)[1] = Math.max(yRangeGivenX.get(x)[1], y);

            xRangeGivenY.putIfAbsent(y, new int[]{Integer.MAX_VALUE, Integer.MIN_VALUE});
            xRangeGivenY.get(y)[0] = Math.min(xRangeGivenY.get(y)[0], x);
            xRangeGivenY.get(y)[1] = Math.max(xRangeGivenY.get(y)[1], x);
        }

        int count = 0;
        for (int[] b : buildings) {
            int x = b[0], y = b[1];
            if (xRangeGivenY.get(y)[0] < x && x < xRangeGivenY.get(y)[1] &&
                yRangeGivenX.get(x)[0] < y && y < yRangeGivenX.get(x)[1]) {
                count++;
            }
        }

        return count;
    }
}
```