LeetCode 750

https://leetcode.com/problems/number-of-corner-rectangles/description/

Yifeng Zeng

Description

750. Number Of Corner Rectangles

Given a grid where each entry is only 0 or 1, find the number of corner rectangles.

A corner rectangle is 4 distinct 1s on the grid that form an axis-aligned rectangle. Note that only the corners need to have the value 1. Also, all four 1s used must be distinct.

Example 1:

```
Input: grid =
[[1, 0, 0, 1, 0],
 [0, 0, 1, 0, 1],
 [0, 0, 0, 1, 0],
 [1, 0, 1, 0, 1]]
Output: 1
```

Explanation: There is only one corner rectangle, with corners grid[1][2], grid[1][4], grid[3][2], grid[3][4].

Example 2:

```
Input: grid =
[[1, 1, 1],
 [1, 1, 1],
 [1, 1, 1]]
Output: 9
```

Explanation: There are four 2x2 rectangles, four 2x3 and 3x2 rectangles, and one 3x3 rectangle.

Example 3:

```
Input: grid =
[[1, 1, 1, 1]]
Output: 0
```

Explanation: Rectangles must have four distinct corners.

Note:

- The number of rows and columns of grid will each be in the range [1, 200].
- Each grid[i][j] will be either 0 or 1.
- The number of 1s in the grid will be at most 6000.

Idea Report

We can do a brute force to count all the possible rectangles the time complexity would be O(n^4) because we enumerate all four corners. How do we speed up? We know that the top two corners need to be in the same row, the same as bottom two corners. And we need to make sure the left two corners and right two corners are in the same column. So we can enumerate each two columns pair by pair, and find in these two columns, how many pair of rows has '1'. So by enumerate each two columns pair, we make sure the left two coners and right two corners are in the same column, and use O(n^2) time. In these two columns, we find how many rows have both '1's, this make sure the top two corners and bottom two coners are in the same row, which make it a rectangle, this take O(n) time to check, so the algorithm takes O(n^3) time.

Code:

```
class Solution {
    // AC
    public int countCornerRectangles(int[][] grid) {
        int sum = 0;
        for (int i = 0; i < grid[0].length - 1; i++) {
            for (int j = i + 1; j < grid[0].length; j++) {
                int onePairs = getPairs(grid, i, j);
                sum += combination(onePairs);
            }
        }

        return sum;
    }

    private int getPairs(int[][] grid, int i, int j) {
        int count = 0;
        for (int r = 0; r < grid.length; r++) {
            if (grid[r][i] == 1 && grid[r][j] == 1) {
                count++;
            }
        }
        return count;
    }

    private int combination(int n) {
        if (n < 2) {
            return 0;
        }
        return n * (n - 1) / 2;
    }
}
```

Summary

- Fix the outter loop of the brute force to speed up.