BAT算法面试题(九)--三维形体投影面积

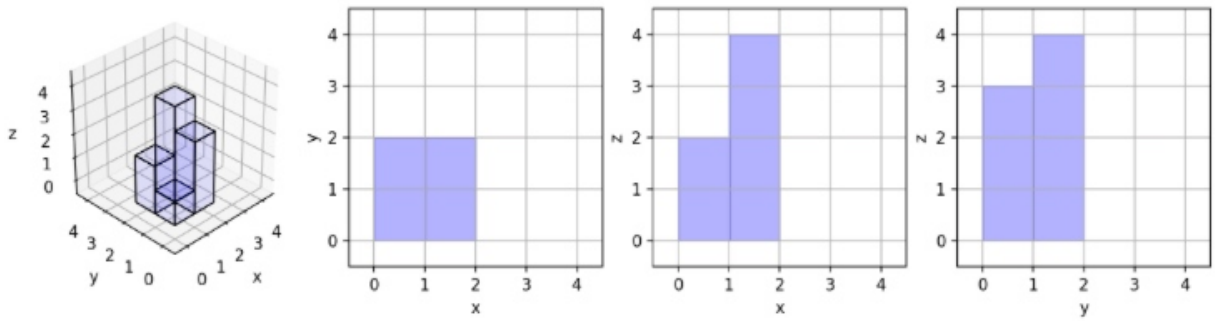
> **文章出自 Hello Code 开发者学习平台 CC老师**
> **获取更新文章/视频 关注公众号:**
> **HelloCode开发者学习平台**

一.题目

在 `N * N` 的网格中，我们放置了一些与 `x，y，z` 三轴对齐的 `1 * 1 * 1` 立方体。每个值 `v = grid[i][j]` 表示 v 个正方体叠放在单元格 (i, j) 上。现在，我们查看这些立方体在 `xy、yz 和 zx` 平面上的投影。

投影就像影子，将三维形体映射到一个二维平面上。在这里，从顶部、前面和侧面看立方体时，我们会看到“影子”。返回所有三个投影的总面积。

- **例子**
  - **输入**: [[1,2],[3,4]]
  - **输出**: 17
  - **解释**: 这里的形状在3个轴对齐平面上的3个投影("阴影部分")



- **提示**:
  - `1<= grid.length = grid[0].length <=50`
  - `0 <= grid[i][j] <=50`

二.解决方案

算法思路

- 从顶部看,由该形状生成的阴影将是网格中非零值的数目
- 从侧面看,由该形状生成的阴影将是网格中每一行的最大值
- 从前面看,由该形状生成的阴影将是网格中每一列中的最大值.

例如 `[[1,2][3,4]]`

- 从顶部的阴影将为4,因为网格中有4个非零值
- 侧面阴影为2+4,因为第一行的最大值为2,第二行的最大值为4
- 前面阴影为3+4,因为第一列的最大值是3,第二列的最大值为4;

三.代码实现

C++ Code

```
class Solution {
public:
    int projectionArea(vector<vector<int>>& grid) {
        int N = grid.size();
        int ans = 0;

        for (int i = 0; i < N;  ++i) {
```

```
            int bestRow = 0;  // 最大行 grid[i][j]
            int bestCol = 0;  // 最大列 grid[j][i]
            for (int j = 0; j < N; ++j) {
                if (grid[i][j] > 0) ans++;  // 顶部阴影
                bestRow = max(bestRow, grid[i][j]);
                bestCol = max(bestCol, grid[j][i]);
            }
            ans += bestRow + bestCol;
        }

        return ans;
    }
};
```

Java Code

```
class Solution {
    public int projectionArea(int[][] grid) {
        int N = grid.length;
        int ans = 0;

        for (int i = 0; i < N;  ++i) {
            int bestRow = 0;
            int bestCol = 0;
            for (int j = 0; j < N; ++j) {
                if (grid[i][j] > 0) ans++;
                bestRow = Math.max(bestRow, grid[i][j]);
                bestCol = Math.max(bestCol, grid[j][i]);
            }
            ans += bestRow + bestCol;
        }

        return ans;
    }
}
```

Python Code

```
lass Solution:
    def projectionArea(self, grid):
        N = len(grid)
        ans = 0

        for i in xrange(N):
            best_row = 0
            best_col = 0
            for j in xrange(N):
                if grid[i][j]: ans += 1
                best_row = max(best_row, grid[i][j])
                best_col = max(best_col, grid[j][i])

            ans += best_row + best_col

        return ans
```

四.复杂度分析

- **时间复杂度**: `O(N^2)` ,其中N是grid的长度
- **空间复杂度**: `O(1)`

五.学习建议

- 结合案例图例分析题目
- 结合代码分析计算逻辑