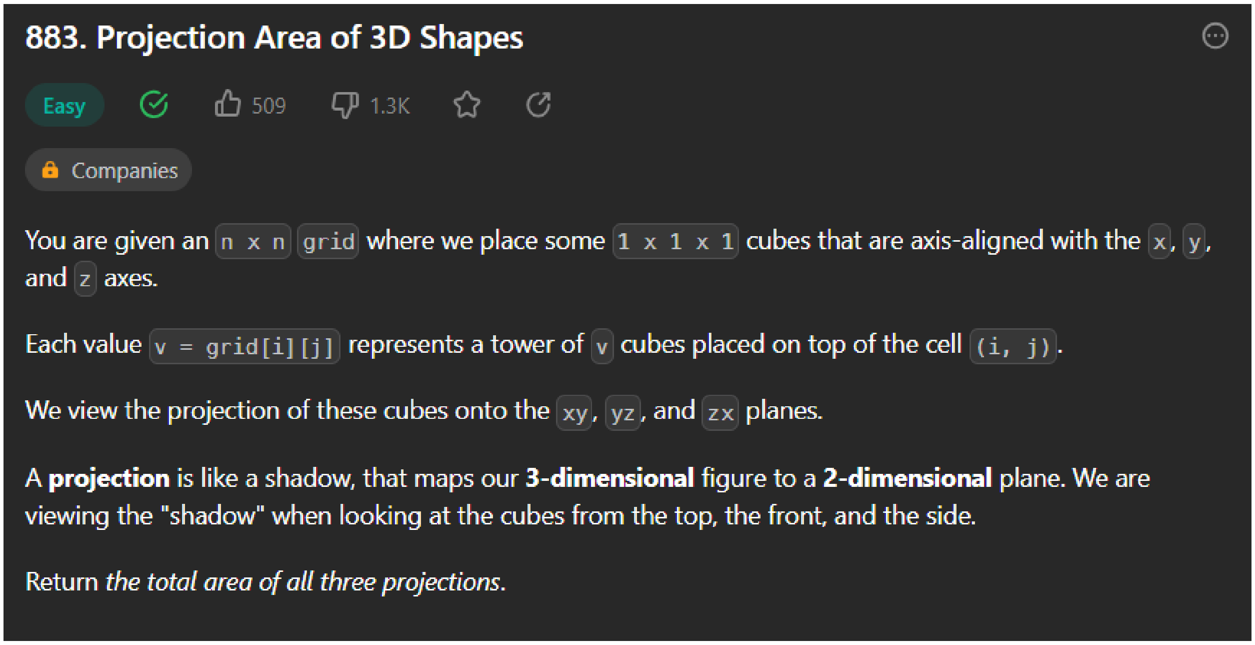
GEOMETRIC ALGORITHMS 3D

Lê Chí Cường – 21520012 | Đoàn Nguyễn Trần Hoàn - 21520239

KHTN2021 Group 13

LEETCODE HOMEWORK



883. Projection Area of 3D Shapes

You are given an `n x n` `grid` where we place some `1 x 1 x 1` cubes that are axis-aligned with the `x`, `y`, and `z` axes.

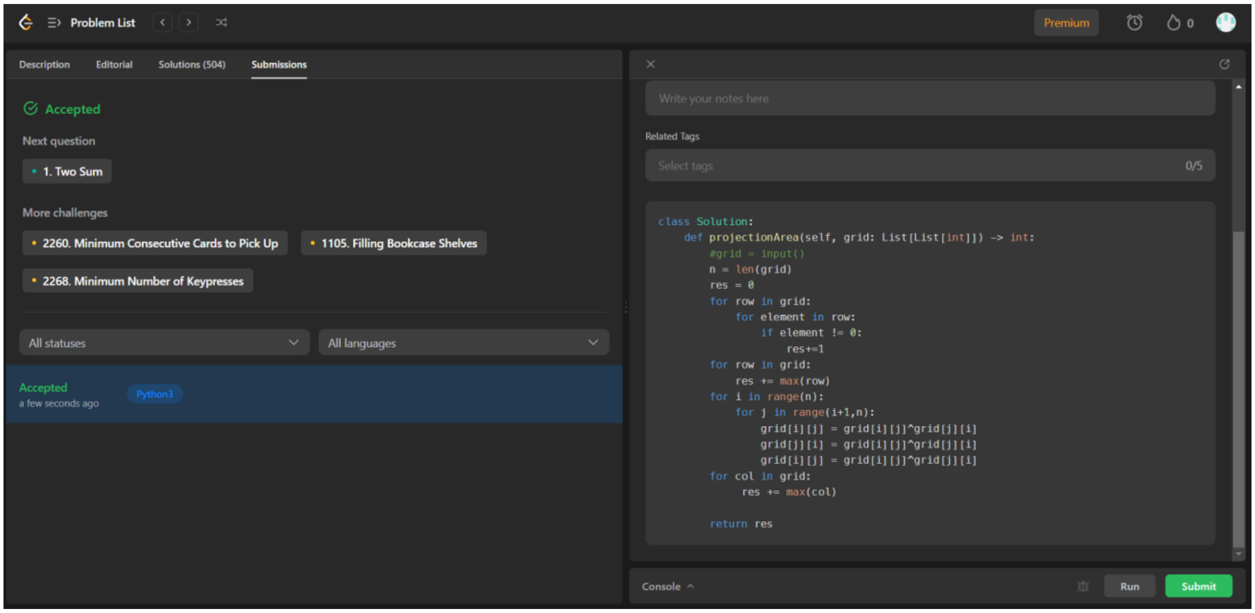
Each value `v = grid[i][j]` represents a tower of `v` cubes placed on top of the cell `(i, j)`.

We view the projection of these cubes onto the `xy`, `yz`, and `zx` planes.

A **projection** is like a shadow, that maps our **3-dimensional** figure to a **2-dimensional** plane. We are viewing the "shadow" when looking at the cubes from the top, the front, and the side.

Return *the total area of all three projections*.

Solution:



Accepted

Next question

- 1. Two Sum

More challenges

- 2260. Minimum Consecutive Cards to Pick Up
- 1105. Filling Bookcase Shelves
- 2268. Minimum Number of Keypresses

Accepted
a few seconds ago

```
class Solution:
    def projectionArea(self, grid: List[List[int]]) -> int:
        #grid = input()
        n = len(grid)
        res = 0
        for row in grid:
            for element in row:
                if element != 0:
                    res+=1
        for row in grid:
            res += max(row)
        for i in range(n):
            for j in range(i+1,n):
                grid[i][j] = grid[i][j]^grid[j][i]
                grid[j][i] = grid[i][j]^grid[j][i]
                grid[i][j] = grid[i][j]^grid[j][i]
        for col in grid:
             res += max(col)

        return res
```

Explain:

- To calculate the area of the projection from the top, we just need to count the number of cell (i, j) that have cubes on it. (Line 6 – 9 in the code)

- To calculate the area of the projection from the front, we need to to calculate the sum of cubes on the cells which have the most cubes on it of each row (only choose one if ties). (Line 10 – 11 in the code)
- To calculate the area of the projection from the side, we need to to calculate the sum of cubes on the cells which have the most cubes on it of each column (only choose one if ties). For easily implementing in Python, I swapped corresponding rows and columns of the *grid* array (Line 12 – 16 in the code), then I just need to use *max* function in Python to get the maximum value of each row (Line 17 – 18 in the code)