Problem 3257: Maximum Value Sum by Placing Three Rooks II

Problem Information

Difficulty: Hard
Acceptance Rate: 26.64%
Paid Only: No
Tags: Array, Dynamic Programming, Matrix, Enumeration

Problem Description

You are given a `m x n` 2D array `board` representing a chessboard, where `board[i][j]` represents the **value** of the cell `(i, j)`.

Rooks in the **same** row or column **attack** each other. You need to place _three_ rooks on the chessboard such that the rooks **do not** **attack** each other.

Return the **maximum** sum of the cell **values** on which the rooks are placed.

Example 1:

Input: board = [[-3,1,1,1],[-3,1,-3,1],[-3,2,1,1]]

Output: 4

Explanation:

![](https://assets.leetcode.com/uploads/2024/08/08/rooks2.png)

We can place the rooks in the cells `(0, 2)`, `(1, 3)`, and `(2, 1)` for a sum of `1 + 1 + 2 = 4`.

Example 2:

Input: board = [[1,2,3],[4,5,6],[7,8,9]]

Output: 15

Explanation:

We can place the rooks in the cells `(0, 0)`, `(1, 1)`, and `(2, 2)` for a sum of `1 + 5 + 9 = 15`.

Example 3:

Input: board = [[1,1,1],[1,1,1],[1,1,1]]

Output: 3

Explanation:

We can place the rooks in the cells `(0, 2)`, `(1, 1)`, and `(2, 0)` for a sum of `1 + 1 + 1 = 3`.

Constraints:

* `3 <= m == board.length <= 500` * `3 <= n == board[i].length <= 500` * `-109 <= board[i][j] <= 109`

Code Snippets

C++:

```
class Solution {
public:
long long maximumValueSum(vector<vector<int>>& board) {

}
};
```

Java:

```
class Solution {
public long maximumValueSum(int[][] board) {

}
}
```

Python3:

```
class Solution:
def maximumValueSum(self, board: List[List[int]]) -> int:
```