Problem 317: Shortest Distance from All Buildings

Problem Information

Difficulty: Hard
Acceptance Rate: 44.73%
Paid Only: Yes
Tags: Array, Breadth-First Search, Matrix

Problem Description

You are given an `m x n` grid `grid` of values `0`, `1`, or `2`, where:

* each `0` marks **an empty land** that you can pass by freely, * each `1` marks **a building** that you cannot pass through, and * each `2` marks **an obstacle** that you cannot pass through.

You want to build a house on an empty land that reaches all buildings in the **shortest total travel** distance. You can only move up, down, left, and right.

Return _the**shortest travel distance** for such a house_. If it is not possible to build such a house according to the above rules, return `-1`.

The **total travel distance** is the sum of the distances between the houses of the friends and the meeting point.

Example 1:

![](https://assets.leetcode.com/uploads/2021/03/14/buildings-grid.jpg)

Input: grid = [[1,0,2,0,1],[0,0,0,0,0],[0,0,1,0,0]] **Output:** 7 **Explanation:** Given three buildings at (0,0), (0,4), (2,2), and an obstacle at (0,2). The point (1,2) is an ideal empty land to build a house, as the total travel distance of 3+3+1=7 is minimal. So return 7.

Example 2:

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

Example 3:

Input: grid = [[1]] **Output:** -1

Constraints:

* `m == grid.length` * `n == grid[i].length` * `1 <= m, n <= 50` * `grid[i][j]` is either `0`, `1`, or `2`. * There will be **at least one** building in the `grid`.

Code Snippets

C++:

```
class Solution {
public:
int shortestDistance(vector<vector<int>>& grid) {

}
};
```

Java:

```
class Solution {
public int shortestDistance(int[][] grid) {

}
}
```

Python3:

```
class Solution:
def shortestDistance(self, grid: List[List[int]]) -> int:
```