LeetCode
Explore
Problems
Interview New
Contest
Discuss
Store
Description
Solution
Discuss (206)
Submissions
Java
Autocomplete

296. Best Meeting Point

Hard 787 63 Add to List Share

Given an `m x n` binary grid `grid` where each `1` marks the home of one friend, return ***the minimal total travel distance***.

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

The distance is calculated using Manhattan Distance, where `distance(p1, p2) = |p2.x - p1.x| + |p2.y - p1.y|`.

Example 1:

1	0	0	0	1
0	0	0	0	0
0	0	1	0	0

```
Input: grid = [[1,0,0,0,1],[0,0,0,0,0],[0,0,1,0,0]]
Output: 6
Explanation: Given three friends living at (0,0), (0,4), and (2,2).
The point (0,2) is an ideal meeting point, as the total travel distance of 2 + 2 +
2 = 6 is minimal.
So return 6.
```

Example 2:

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

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 200`
- `grid[i][j]` is either `0` or `1`.
- There will be **at least two** friends in the `grid`.

Accepted 52,459 | Submissions 88,699

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Facebook | 5 Expedia | 2

Related Topics

Array Math Sorting Matrix

Similar Questions

Shortest Distance from All Buildings	Hard
Minimum Moves to Equal Array Elements II	Medium

Hide Hint 1

Try to solve it in one dimension first. How can this solution apply to the two dimension case?

```java
public int minTotalDistance(int[][] grid) {
    List<Integer> rows = new ArrayList<>();
    List<Integer> cols = new ArrayList<>();
    for (int row = 0; row < grid.length; row++) {
        for (int col = 0; col < grid[0].length; col++) {
            if (grid[row][col] == 1) {
                rows.add(row);
                cols.add(col);
            }
        }
    }
    int row = rows.get(rows.size() / 2);
    Collections.sort(cols);
    int col = cols.get(cols.size() / 2);
    return minDistance1D(rows, row) + minDistance1D(cols, col);
}

private int minDistance1D(List<Integer> points, int origin) {
    int distance = 0;
    for (int point : points) {
        distance += Math.abs(point - origin);
    }
    return distance;
}
```