317. Shortest Distance from All Buildings

Hard 705 42 Add to List Share

You want to build a house on an *empty* land which reaches all buildings in the shortest amount of distance. You can only move up, down, left and right. You are given a 2D grid of values **0**, **1** or **2**, where:

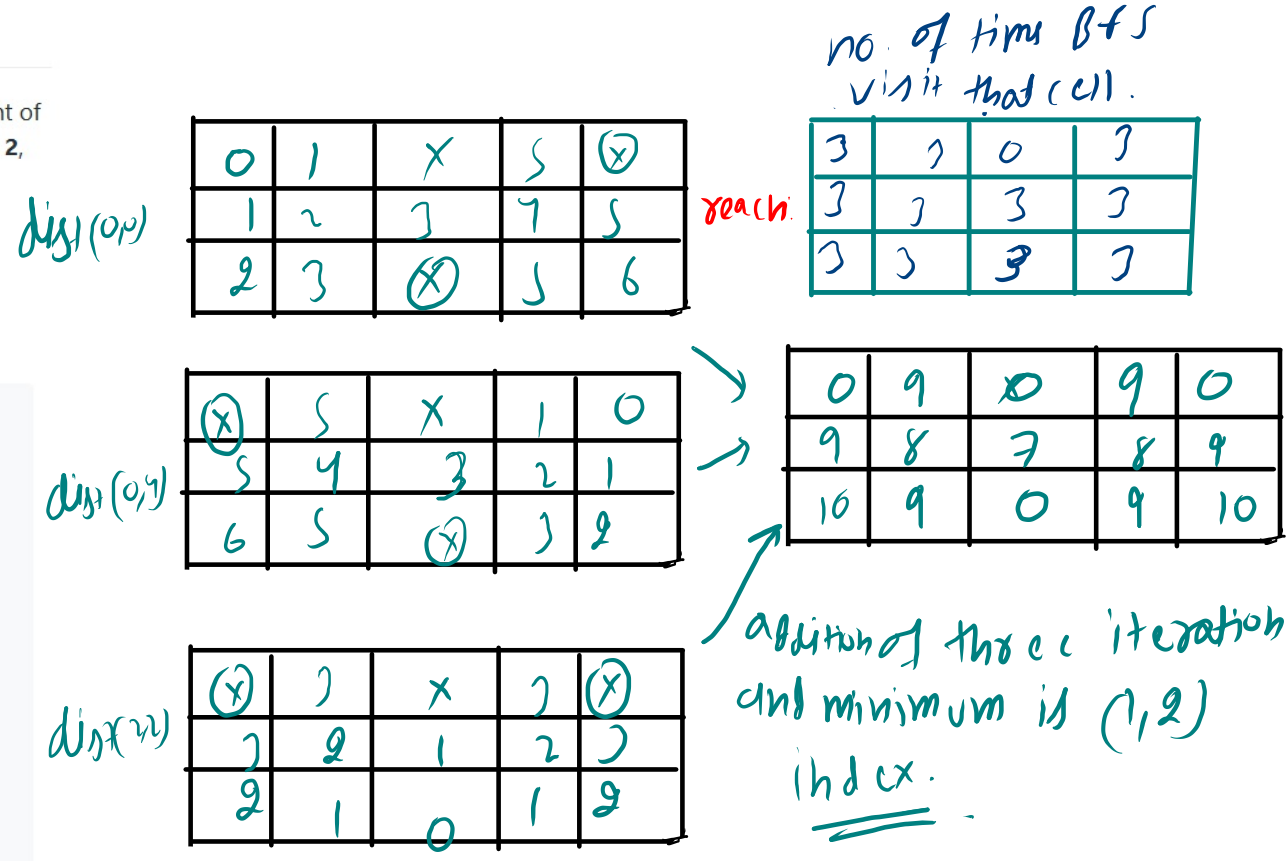
- Each **0** marks an empty land which you can pass by freely.
- Each **1** marks a building which you cannot pass through.
- Each **2** marks an obstacle which you cannot pass through.

Input: [[1,0,2,0,1],[0,0,0,0,0],[0,0,1,0,0]]

```
  0   1   2   3   4
0 1 - 0 - 2 - 0 - 1
  |   |   |   |   |
1 0 - 0 - 0 - 0 - 0
  |   |   |   |   |
2 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.



ek baar mein sirf ek 1 active rahega baki 1 inactive rahenge and 2 is a blockage.

a.) isme aap sabse phele 1 (0,0) se BFS chlao or distance matrix ko fill karo level se.

b.) phir same cheez (0,4) wale 1 se karo and then same cheez (2,2) wale 1 se.

c.) distance matrix mai bs add krna hai or reach matrix pe visit ka +1 krna hai and last mai distance matrix mai sabse minimum cell ka pata lagana hai jo haar building ko reach krta ho.

```cpp
int shortestDistance(vector<vector<int>> &grid)
{
    if (grid.size() == 0 || grid[0].size() == 0)
        return -1;

    n = grid.size();
    m = grid[0].size();
    distance.resize(n, vector<int>(m, 0));
    reach.resize(n, vector<int>(m, 0));

    int NoofOnces = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            if (grid[i][j] == 1)
            {
                vis.clear();
                vis.resize(n, vector<int>(m, 0));
                vis[i][j] = 1;
                shortestDistance_(i * m + j, grid);
                NoofOnces++;
            }
    int min_ = 1e8;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            if (distance[i][j] != 0 && reach[i][j] == NoofOnces)
            {
                min_ = min(min_, distance[i][j]);
            }
    return min_ != 1e8 ? min_ : -1;
}
```

only for 1 and one at a time each time we call BFS, should clear old visited array.

count of ones's, jiska use hum ye pata lagnane ke liye krenge ki kaunse cell sarre building ko reach krte hai.

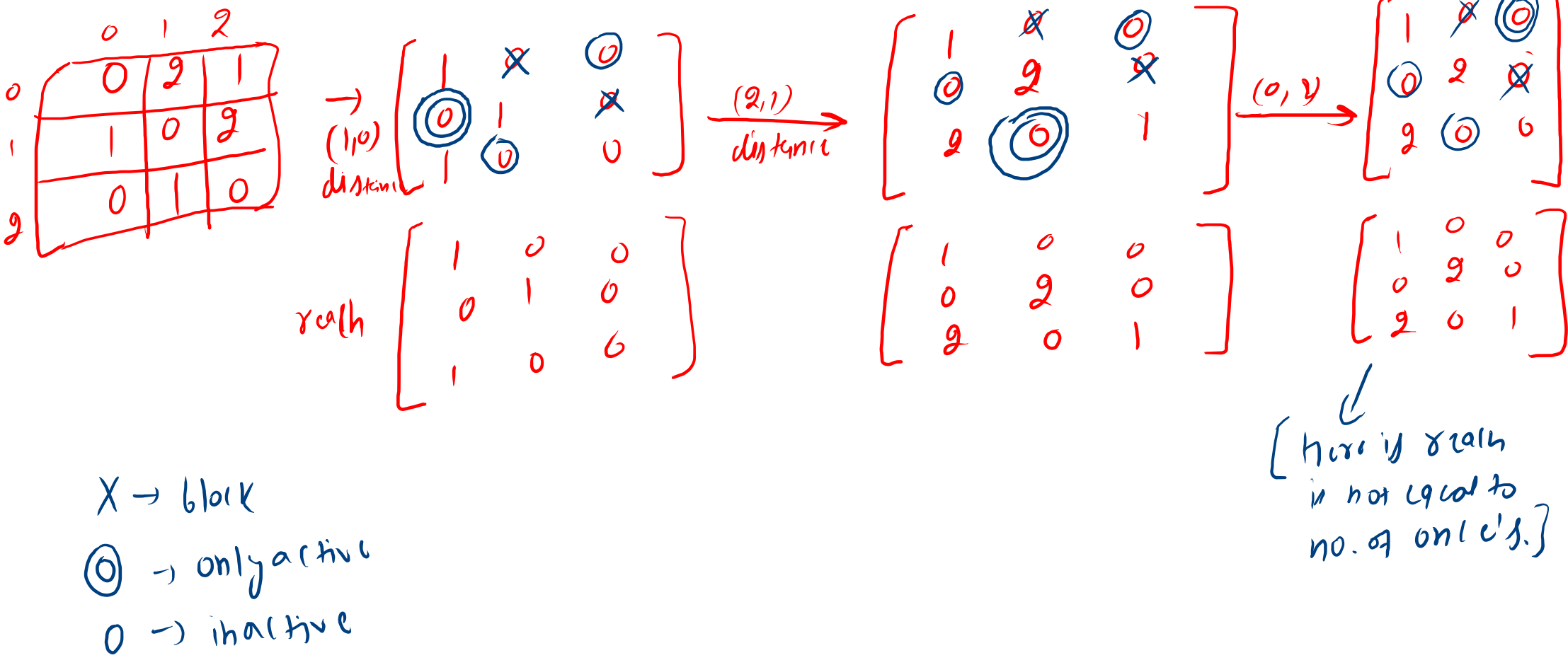
```cpp
vector<vector<int>> distance, reach, vis;
int n, m;

void shortestDistance_(int src, vector<vector<int>> &grid)
{
    int dirA[4][2] = {{-1, 0}, {0, -1}, {0, 1}, {1, 0}};
    queue<int> que;
    que.push(src);

    int level = 1;
    while (que.size() != 0)
    {
        int size = que.size();
        while (size-- > 0)
        {
            int rvtx = que.front();
            que.pop();

            int r = rvtx / m;
            int c = rvtx % m;
            reach[r][c]++;

            for (int d = 0; d < 4; d++)
            {
                int x = r + dirA[d][0];
                int y = c + dirA[d][1];

                if (x >= 0 && x < n && y >= 0 && y < m && grid[x][y] == 0 && vis[x][y] == 0)
                {
                    que.push((x * m + y));
                    distance[x][y] += level;
                    vis[x][y] = 1;
                }
            }
        }
        level++;
    }
}
```

grid mai sirf empty cell pe jana hai and visited mark krna hai. or level +1. because first building apna level dalega and second building apna level dalega iska matalb empty cell ka distance first building se + second building se. similiar sabhi building se

1. distance[i][j] !=0 means empty cell because BFS mai hum log kaabhi building ke pass ya block pe nahi jate hai to ye sarre cells 0 ke 0 rhenge or bakki empty cells fill hojynege upcoming level se and final mein har ek empty cell jiski reach NoofOnces ke equal hogi vo sabhi sabhi building se apna distance store krke betha hoga as a sum.

2. reach[i][j] means inn empty cells pe kitni building recah krti hai. jispe sarre building reach krenge vo cells ek valid cell hoga or inka minimum distance humra answer bnega.

```
[[0,2,1],[1,0,2],[0,1,0]]
```


0 1 2
0 2 1
1 0 2
0 1 0
(1,0) distinct
(2,1) distinct
(0,1)
graph
[Here if graph is not equal to no. of one's.]
X → block
◎ → only active
0 → inactive