题目描述：

存在一个 m*n 的二维数组，其成员取值范围为 0,1,2。其中值为 1 的元素具备同化特性，每经过 1S，将上下左右值为 0 的元素同化为 1。而值为 2 的元素，免疫同化。将数组所有成员随机初始化为 0 或 2，再将矩阵的[0,0]元素修改成 1，在经过足够长的时间后，求矩阵中有多少个元素是 0 或 2（即 0 和 2 数量之和）。

输入描述：

输入的前两个数字是矩阵大小。后面的数字是矩阵内容。

输出描述：

返回矩阵中非 1 的元素个数

补充说明：

m 和 n 不会超过 30(含 30)。

示例 1

输入：

```
4 4
0 0 0 0
0 2 2 2
0 2 0 0
0 2 0 0
```

输出：

```
9
```

说明：

输入数字前两个数字是矩阵大小。后面的是数字是矩阵内容。
这个矩阵的内容如下：

{

0,0,0,0

0,2,2,2

0,2,0,0

0,2,0,0

}

起始位置 (0,0)被修改为 1 后，最终只能同化矩阵为：

{

1,1,1,1

1,2,2,2

1,2,0,0

1,2,0,0

}

所以矩阵中非 1 的元素个数为 9。

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int H, W;
    cin >> H >> W;
    if (H<=0 || W<=0) {
        cout << 0;
        return 0;
    }
    int tensor[31][31] = {-1};
    for (int h = 0; h < H; h++) {
        for (int w = 0; w < W; w++) {
            cin >> tensor[h][w];
        }
    }
    if (tensor[0][0] == 0) tensor[0][0] = 1;
    vector<int> to_search_offsets;
    int num_1 = 0;
    for (int h=0; h<H; h++) {
        for (int w=0; w<W; w++) {
            if (tensor[h][w] == 1) {
                int offset=h*W+w;
                to_search_offsets.push_back(offset);
```

```
                num_1++;
            }
        }
    }

    int arround[4][2] = {{0, -1}, {0, 1}, {-1, 0}, {1, 0}};
    while (!to_search_offsets.empty()) {
        int offset = to_search_offsets.back();
        to_search_offsets.pop_back();
        int h = offset / W;
        int w = offset % W;
        for (int i = 0; i < 4; i++) {
            int new_h = h + arround[i][0];
            int new_w = w + arround[i][1];
            if (new_h < 0 || new_h >= H || new_w < 0 || new_w >= W) {
                continue;
            }
            if (tensor[new_h][new_w] == 0) {
                num_1++;
                tensor[new_h][new_w] = 1;
                int new_offset = new_h * W + new_w;
                to_search_offsets.push_back(new_offset);
            }
        }
    }
    cout << H*W-num_1;
}
```