BFS（1）

给定一个 $n \times m$ 的二维整数数组，用来表示一个迷宫，数组中只包含 0 或 1，其中 0 表示可以走的路，1 表示不可通过的墙壁。

最初，有一个人位于左上角 $(1,1)$ 处，已知该人每次可以向上、下、左、右任意一个方向移动一个位置。

请问，该人从左上角移动至右下角 (n,m) 处，至少需要移动多少次。

数据保证 $(1,1)$ 处和 (n,m) 处的数字为 0，且一定至少存在一条通路。

输入格式

第一行包含两个整数 n 和 m。

接下来 n 行，每行包含 m 个整数（0 或 1），表示完整的二维数组迷宫。

输出格式

输出一个整数，表示从左上角移动至右下角的最少移动次数。

数据范围

$1 \le n,m \le 100$

输入样例：

```
5 5
0 1 0 0 0
0 1 0 1 0
0 0 0 0 0
0 1 1 1 0
0 0 0 1 0
```

输出样例：

```
8
```

y总代码：

```
#include <cstring>
#include <iostream>
#include <algorithm>
#include <queue>

using namespace std;

typedef pair<int, int> PII;

const int N = 110;

int n, m;
int g[N][N], d[N][N];

int bfs()
{
    queue<PII> q;

    memset(d, -1, sizeof d);
```

```
    d[0][0] = 0;
    q.push({0, 0});

    int dx[4] = {-1, 0, 1, 0}, dy[4] = {0, 1, 0, -1};

    while (q.size())
    {
        auto t = q.front();
        q.pop();

        for (int i = 0; i < 4; i ++ )
        {
            int x = t.first + dx[i], y = t.second + dy[i];

            if (x >= 0 && x < n && y >= 0 && y < m && g[x][y] == 0 && d[x][y] ==
-1)
            {
                d[x][y] = d[t.first][t.second] + 1;
                q.push({x, y});
            }
        }
    }

    return d[n - 1][m - 1];
}

int main()
{
    cin >> n >> m;
    for (int i = 0; i < n; i ++ )
        for (int j = 0; j < m; j ++ )
            cin >> g[i][j];

    cout << bfs() << endl;

    return 0;
}
```


理解：

```
#include<iostream>
#include<algorithm>
#include<cstring>

using namespace std;

const int N=110;


int g[N][N];     //存储地图
int d[N][N];     //标记搜索到的点的距离
int n,m;         //地图的长宽
```

```
struct elem{
    int x;
    int y;
};

struct elem q[N*N]; //队列用于BFS操作
int bfs()
{
    memset(d,-1,sizeof d);  //初始化d

    int hh=0,tt=0;
    d[0][0]=0;  //左上角第一个点开始搜索

    q[hh]={0,0};    //将第一个点入队

    int dx[4]={-1,0,1,0};   //方向盘用于对个方向进行尝试
    int dy[4]={0,-1,0,1};

    while(hh<=tt)   //
    {
        struct elem j=q[hh++];  //获取队头元素 然后将其出队

        for(int i=0;i<4;i++)
        {
            int x=j.x+dx[i];
            int y=j.y+dy[i];
            if(x>=0&&y>=0&&x<n&&y<m&&g[x][y]==0&&d[x][y]==-1)     //点未出界 该点可
走 且未走过
            {
                d[x][y]=d[j.x][j.y]+1;
                q[++tt]={x,y};
            }
        }
    }

    return    d[n-1][m-1];
}
int main()
{
    cin>>n>>m;
    for(int i=0;i<n;i++)
        for(int j=0;j<m;j++)
            cin>>g[i][j];

    cout<<bfs()<<endl;
    return 0;

}
```