定义一个二维数组N*M（其中2<=N<=10;2<=M<=10），如5 × 5数组下所示：

int maze[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,
};

它表示一个迷宫，其中的1表示墙壁，0表示可以走的路，只能横着走或竖着走，不能斜着走，要求编程序找出从左上角到右下角的最短路线。入口点为[0,0],既第一空格是可以走的路。

本题含有多组数据。

输入描述:

> 输入两个整数，分别表示二位数组的行数，列数。再输入相应的数组，其中的1表示墙壁，0表示可以走的路。数据保证有唯一解,不考虑有多解的情况，即迷宫只有一条通道。

输出描述:

> 左上角到右下角的最短路径，格式如样例所示。

示例1

输入

```
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

```
(0,0)
(1,0)
(2,0)
(2,1)
(2,2)
(2,3)
(2,4)
(3,4)
(4,4)
```

```
#include<iostream>
#include<vector>
using namespace std;

struct coord{
	int x;
	int y;
};

int maze[10][10];
vector<coord> path_temp;
vector<coord> path_best;
int n,m;


void travel(int i, int j)
{
	maze[i][j] = 1;//表示当前节点已走，不可再走
	path_temp.push_back({ i, j });//将当前节点加入到路径中

	if (i == n - 1 && j == m - 1) //判断是否到达终点
		if (path_best.empty() || path_temp.size() < path_best.size())
			path_best = path_temp;

	if (i - 1 >= 0 && maze[i - 1][j] == 0)//探索向上走是否可行
		travel(i - 1, j);
	if (i + 1 < n && maze[i + 1][j] == 0)//探索向下走是否可行
		travel(i + 1, j);
	if (j - 1 >= 0 && maze[i][j - 1] == 0)//探索向左走是否可行
		travel(i, j - 1);
	if (j + 1 < m && maze[i][j+1] == 0)//探索向右走是否可行
		travel(i, j + 1);
	maze[i][j] = 0;			//恢复现场，设为未走
	path_temp.pop_back();
}

int main()
{

	while(cin>>n>>m)
	{
		path_temp.clear();
		path_best.clear();
		for(int i=0;i<n;i++)
```

```
        {
            for(int j=0;j<m;j++)
            {
                cin>>maze[i][j];
            }
        }
        travel(0,0);//从头开始查找
        for(int i=0;i<path_best.size();i++)
        {
            cout<<"("<<path_best[i].x<<","<<path_best[i].y<<")"<<endl;
        }
    }
    return 0;
}
```