一、编程题

ACM：欢乐的周末

题目描述：小华和小为是很要好的朋友，他们约定周末一起吃饭。通过手机交流，他们在地图上选择了多个聚餐地点（由于自然地形等原因，部分聚餐地点不可达），求小华和小为都能到达的聚餐地点有多少个？

输入描述：第一行输入m和n，m代表地图的长度，n代表地图的宽度。
第二行开始具体输入地图信息，地图信息包含：
0 为通畅的道路
1 为障碍物（且仅1为障碍物）
2 为小华或者小为，地图中必定有且仅有2个（非障碍物）
3 为被选中的聚餐地点（非障碍物）

输出描述：可以被两方都到达的聚餐地点数量，行末无空格。

补充说明：地图的长宽为m和n，其中：
4 <= m <= 100
4 <= n <= 100

聚餐的地点数量为 k，则
1< k <= 100

示例1
输入：4 4
2 1 0 3
0 1 2 1
0 3 0 0
0 0 0 0
输出：2
说明：第一行输入地图的长宽为3和4。
第二行开始为具体的地图，其中：3代表小华和小明选择的聚餐地点；2代表小华或者小明（确保有2个）；0代表可以通行的位置；1代表不可以通行的位置。

此时两者能都能到达的聚餐位置有2处

示例2
输入：4 4
2 1 2 3
0 1 0 0
0 1 0 0
0 1 0 0
输出：0
说明：第一行输入地图的长宽为4和4。
第二行开始为具体的地图，其中：3代表小华和小明选择的聚餐地点；2代表小华或者小明（确保有2个）；0代表可以通行的位置；1代表不可以通行的位置。

由于图中小华和小为之间有个阻隔，此时，没有两人都能到达的聚餐地址，故而返回0

代码;

```
import sys

fangxiangs = [(-1,0), (1, 0), (0,1), (0, -1)]

class unionfindset:
    def __init__(self, n):
        self.set = [x for x in range(n)]
        self.num = n

    def union(self, x, y):
        x_ = self.find(x)
```

```
            y_ = self.find(y)

            if x_ != y_:
                self.set[y_] = self.set[x_]
                self.num -= 1

        def find(self, x):
            if x != self.set[x]:
                self.set[x] = self.find(self.set[x])
                return self.set[x]
            else:
                return self.set[x]

    while 1:
        line = sys.stdin.readline().strip()
        if line == '':
            break
        n, m = list(map(int, line.split()))

        arr = []
        for _ in range(n):
            line = sys.stdin.readline().strip()
            arr.append(list(map(int, line.split())))

        huawei = []
        targets = []
        ufs = unionfindset(n*m)

        for i in range(n):
            for j in range(m):
                if arr[i][j] == 2:
                    huawei.append([i, j])
                elif arr[i][j] == 3:
                    targets.append([i, j])

                if arr[i][j] != 1:
                    for fangxiang in fangxiangs:
                        newx = i + fangxiang[0]
                        newy = j + fangxiang[1]
                        if 0 <= newx < n and 0 <= newy < m and arr[newx][newy] != 1:
                            ufs.union(i*m+j, newx*m+newy)

        count = 0
        k = ufs.find(huawei[0][0]*m+huawei[0][1])
```

```
if ufs.find(huawei[1][0]*m+huawei[1][1]) == k:
        for t in targets:
                if ufs.find(t[0]*m+t[1]) == k:
                        count += 1

print(count)
```