1257：Knight Moves

题目描述

输入n代表有个n×n的棋盘，输入开始位置的坐标和结束位置的坐标，问一个骑士朝棋盘的八个方向走马字步，从开始坐标到结束坐标可以经过多少步。

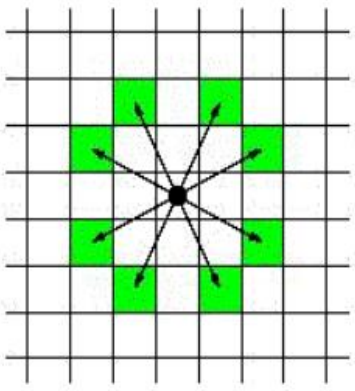

Figure 1: Possible knight moves on the board.

输入格式

首先输入一个n,表示测试样例的个数。

每个测试样例有三行。

第一行是棋盘的大小L(4≤L≤300)；

第二行和第三行分别表示马的起始位置和目标位置(0..L-1)。

输出格式

马移动的最小步数，起始位置和目标位置相同时输出0。

输入样例

```
3
8
0 0
7 0
100
0 0
30 50
10
1 1
1 1
```

输出样例

5
28
0

解析

最短路径，八方向广搜。

编码

```
#include <bits/stdc++.h>

using namespace std;

int const MaxNum = 301;

//节点结构体
struct Node {
    int x;    //坐标
    int y;
    int step;   //移动步数

    //当前节点的基本信息
    Node(int nx, int ny, int stepNum) {
        x = nx;
        y = ny;
        step = stepNum;
    }

    Node() = default;
};

int R;  //棋盘的宽度
bool Vis[MaxNum][MaxNum];  //判断是否走过
//马的八方向
int gapX[8] = {-2, -1, 1, 2, -2, -1, 1, 2,};
int gapY[8] = {-1, -2, -2, -1, 1, 2, 2, 1};
Node NodeQueue[MaxNum * MaxNum];


//检测是否可以通过
bool CanPass(int newX, int newY) {
    //保证没有越界
    if (newX >= 0 && newY >= 0 && newX < R && newY < R) {
        //没有被访问过
        if (!Vis[newX][newY]) {
            return true;
```

```
        }
    }
    return false;
}


//广度搜索
void Bfs(Node start, Node end) {
    //特判断起点和终点相等
    if (start.x == end.x && start.y == end.y) {
        cout << 0 << endl;
        return;
    }
    //初始化
    memset(Vis, false, sizeof(Vis));
    memset(NodeQueue, 0, sizeof(NodeQueue));
    //设置头尾
    int head = 0;
    int tail = 0;
    //将首点加入队列
    NodeQueue[head] = start;
    //设置该节点已经被访问
    Vis[start.x][start.y] = true;
    //标记当前节点已经走过
    while (head <= tail) {
        //取出一个节点
        Node room = NodeQueue[head];
        head++;
        //向四个方向移动;
        for (int k = 0; k < 8; ++k) {
            int newX = room.x + gapX[k];
            int newY = room.y + gapY[k];
            //找到目标;
            if (newX == end.x && newY == end.y) {
                //输出一共走的步数，不需要记录最后一个点
                cout << room.step << endl;
                return;
            }
            //下个节点可以被访问
            if (CanPass(newX, newY)) {
                //设置该节点已经被访问
                Vis[newX][newY] = true;
                tail++;
                //将数据加入新的队列
                NodeQueue[tail] = Node(newX, newY, room.step + 1);
            }
        }
    }
    cout << "oop!" << endl;
```

```
}

//读取输入的数据
void ReadInfo() {
    Node start = Node(-1, -1, 1);
    Node end = Node(-1, -1, 1);
    int startX, startY;
    cin >> startX >> startY;
    int endX, endY;
    cin >> endX >> endY;

    start = Node(startX, startY, 1);
    end = Node(endX, endY, 1);

    Bfs(start, end);
}

int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; ++i) {
        cin >> R;
        ReadInfo();
    }

    return 0;
}
```