一、编程题

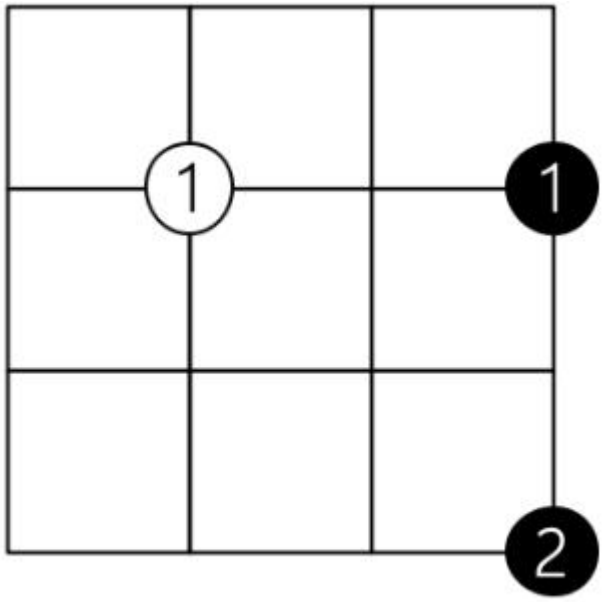
ACM：围棋的气

题目描述：

围棋棋盘由纵横各 19 条线垂直相交组成，棋盘上一共 19x19=361 个交点，对弈双方一方执白棋，一方执黑棋，落子时只能将棋子置于交点上。

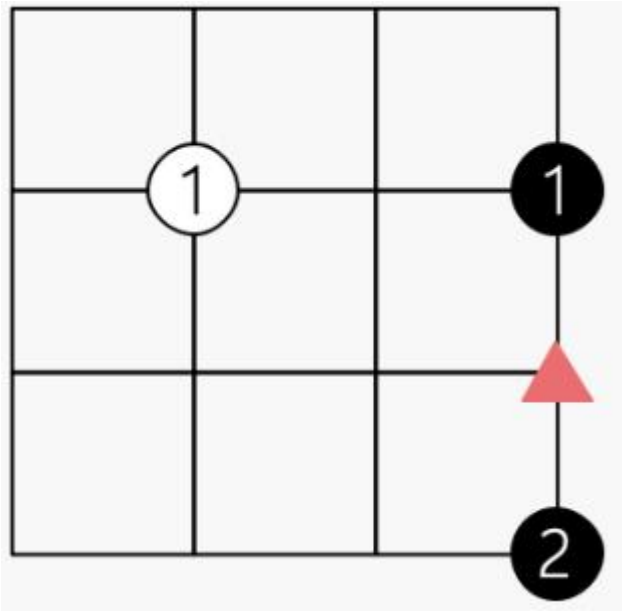
“气”是围棋中很重要的一个概念，某个棋子有几口气，是指其上下左右方向四个相邻的交叉点中，有几个交叉点没有棋子，由此可知：

1、在棋盘的边缘上的棋子最多有 3 口气（黑 1），在棋盘角点的棋子最多有 2 口气（黑 2），其它情况最多有 4 口气（白 1）



白1四口气，黑1三口气，黑2两口气

2、所有同色棋子的气之和叫作该色棋子的气，需要注意的是，同色棋子重合的气点，对于该颜色棋子来说，只能计算一次气，比如下图中，黑棋一共 4 口气，而不是 5 口气，因为黑 1 和黑 2 中间红色三角标出的气是两个黑棋共有的，对于黑棋整体来说只能算一个气。



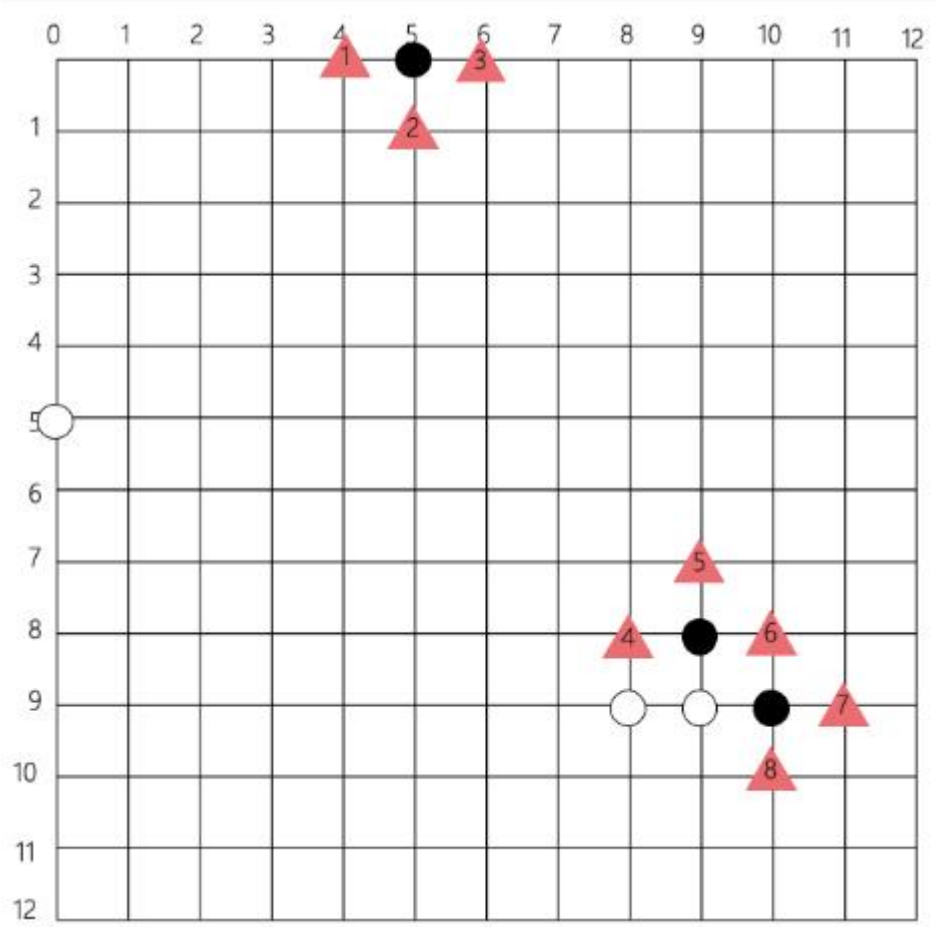
3、本题目只计算气，对于眼也按气计算，如果您不清楚“眼”的概念，可忽略，按照前面描述的规则计算即可。
现在，请根据输入的黑棋和白棋的坐标位置，计算黑棋和白起一共各有多少气？

输入描述：输入包括两行数据，如：
0 5 8 9 9 10
5 0 9 9 9 8
1、每行数据以空格分隔，数据个数是2的整数倍，每两个数是一组，代表棋子在棋盘上的坐标；
2、坐标的原点在棋盘左上角点，第一个值是行号，范围从0到18；第二个值是列号，范围从0到18。
3、举例说明：第一行数据表示三个坐标（0，5）、（8，9）、（9，10）
4、第一行表示黑棋的坐标，第二行表示白棋的坐标。
5、题目保证输入两行数据，无空行且每行按前文要求是偶数个，每个坐标不会超出棋盘范围。

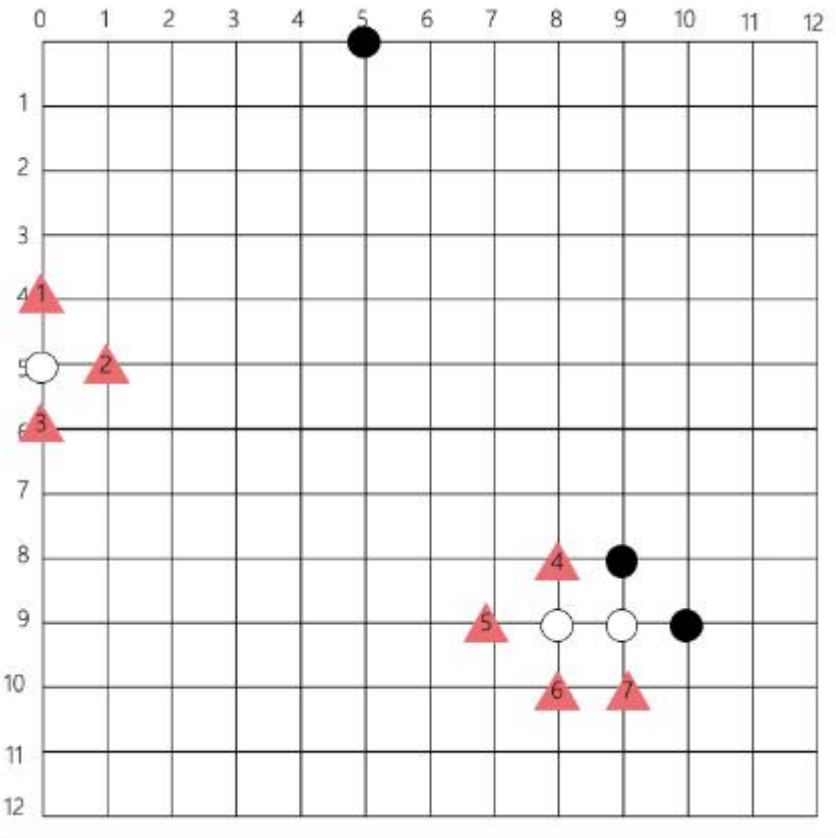
输出描述：8 7
两个数字以空格分隔，第一个数代表黑棋的气数，第二个数代表白棋的气数。

补充说明：

示例1
输入：0 5 8 9 9 10
5 0 9 9 9 8
输出：8 7
说明：如果将输入数据放到棋盘上，数数黑棋一共8口气：



数数白棋一共 7 口气：



代码：

```
import java.util.Arrays;
import java.util.HashSet;
import java.util.Scanner;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Integer[] black = Arrays.stream(sc.nextLine().split("\\s+")).map(Integer::parseInt).toArray(Integer[]::new);
        Integer[] white = Arrays.stream(sc.nextLine().split("\\s+")).map(Integer::parseInt).toArray(Integer[]::new);
        HashSet<String> blackPos = new HashSet<>();
        for (int i = 0; i < black.length; i += 2) {
            blackPos.add(black[i] + "-" + black[i + 1]);
        }
        HashSet<String> whitePos = new HashSet<>();
        for (int i = 0; i < white.length; i += 2) {
            whitePos.add(white[i] + "-" + white[i + 1]);
        }

        int[][] chess = new int[19][19];
```

```
        int blackRes = 0;
        int whiteRes = 0;


        int[][] step = new int[][]{{0, 1}, {1, 0}, {0, -1}, {-1, 0}};

        HashSet<String> blackChecked = new HashSet<>();
        for (int i = 0; i < black.length; i += 2) {
            int x = black[i];
            int y = black[i + 1];
            for (int[] s : step) {
                int newx = x + s[0];
                int newy = y + s[1];
                if (valid(newx, newy)) {
                    if (!blackPos.contains(newx + "-" + newy) && !whitePos.contains(newx +
"-" + newy)) {
                        if (!blackChecked.contains(newx + "-" + newy)) {
                            blackRes++;
                            blackChecked.add(newx + "-" + newy);
                        }
                    }
                }
            }
        }

        HashSet<String> whiteChecked = new HashSet<>();
        for (int i = 0; i < white.length; i += 2) {
            int x = white[i];
            int y = white[i + 1];
            for (int[] s : step) {
                int newx = x + s[0];
                int newy = y + s[1];
                if (valid(newx, newy)) {
                    if (!blackPos.contains(newx + "-" + newy) && !whitePos.contains(newx +
"-" + newy)) {
                        if (!whiteChecked.contains(newx + "-" + newy)) {
                            whiteRes++;
                            whiteChecked.add(newx + "-" + newy);
                        }
                    }
                }
            }
        }
```

```
            System.out.println(blackRes + " " + whiteRes);
        }

        public static boolean valid(int x, int y) {
            return x >= 0 && x <= 18 && y >= 0 && y <= 18;
        }
    }
```