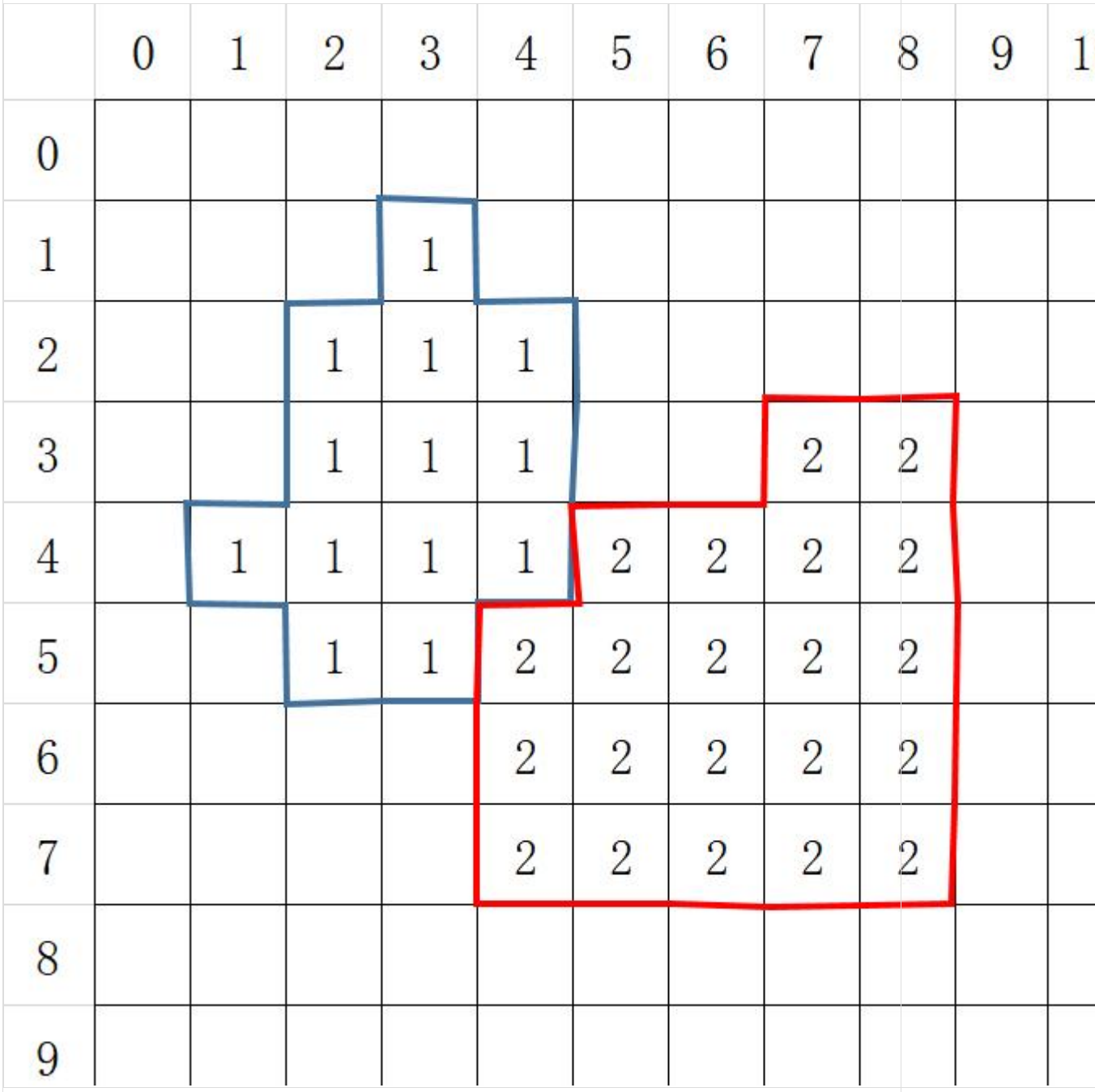
题目描述：

有一个 64*64 的矩阵，每个元素的默认值为 0，现在向里面填充数字，相同的数字组成一个实心图形，如下图所示是矩阵的局部（空白表示填充 0）：



	0	1	2	3	4	5	6	7	8	9	1
0											
1				1							
2			1	1	1						
3			1	1	1			2	2		
4		1	1	1	1	2	2	2	2		
5			1	1	2	2	2	2	2		
6					2	2	2	2	2		
7					2	2	2	2	2		
8											
9											

数字 1 组成了蓝色边框的实心图形，数字 2 组成了红色边框的实心图形。

单元格的边长规定为 1 个单位，请根据输入，计算每个非 0 值填充出来的实心图形的周长。

输入描述：

2

1 1 3 2 2 2 3 2 4 3 2 3 3 3 4 4 1 4 2 4 3 4 4 5 2 5 3

2 3 7 3 8 4 5 4 6 4 7 4 8 5 4 5 5 5 6 5 7 5 8 6 4 6 5 6 6 6 7 6 8 7 4 7

5 7 6 7 7 7 8

输入数据说明如下：

1、第一行输入 N，表示一共有 N 个图形，N>0 且 N<64*64；

2、矩阵左上角单元格坐标记做（0,0），第一个数字表示行号，第二个数字表示列号；

3、接下来是 N 行，每行第一个数字是矩阵单元格填充的数字，后续每两个一组，表示填充该数字的单元格的坐标；

4、答题者无需考虑数据格式非法的场景，题目用例不考察数据格式；

5、题目用例保证同一个填充值只会有一行输入数据。

输出描述：

18 20

1、一共输出 N 个数值，每个数值表示某一输入行表示图形的周长；

2、输出顺序需和输入的各行顺序保持一致，即第 1 个数是输入的第 1 个图形的周长，第 2 个数是输入的第 2 个图形的周长，以此类推。

示例 1

输入：

2

```
1 1 3 2 2 2 3 2 4 3 2 3 3 3 4 4 1 4 2 4 3 4 4 5 2 5 3
2 3 7 3 8 4 5 4 6 4 7 4 8 5 4 5 5 5 6 5 7 5 8 6 4 6 5 6 6 6 7 6 8 7
4 7 5 7 6 7 7 7 8
```

输出：

```
18 20
```

说明：

本样例中，经过观察和计算，1 组成的图形的周长为 18 个单位，2 组成的图形的周长为 20 个单位。

```
import java.util.Objects;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int count = Integer.parseInt(sc.nextLine());

        String[] lineList = new String[count];

        for (int i = 0; i < count; i++) {
            lineList[i] = sc.nextLine();
        }

        int[] res = new int[count];

        for (int i = 0; i < lineList.length; i++) {
            int[][] area = new int[64][64];
            String[] strArr = lineList[i].split(" ");
            int[] intArr = new int[strArr.length];
            for (int numIndex = 0; numIndex < intArr.length; numIndex++) {
                intArr[numIndex] = Integer.parseInt(strArr[numIndex]);
            }

            int num = intArr[0];

            for (int j = 0; j < (intArr.length - 1) / 2; j++) {
                int index = 2 * j;
                area[intArr[index + 1]][intArr[index + 2]] = num;
            }

            int l = 0;
```

```
                for (int j = 0; j < (intArr.length - 1) / 2; j++) {
                    int index = 2 * j;
                    l += length(num, area, intArr[index + 1], intArr[index + 2]);
                }
                res[i] = l;
            }
            StringBuilder sb = new StringBuilder();
            for (int re : res) {
                if (sb.length() > 0) {
                    sb.append(" ");
                }
                sb.append(re);
            }

            System.out.println(sb.toString());
        }

        private static int length(int num, int[][] area, int row, int column) {
            int res = 0;
            if (row == 0 || !Objects.equals(num, area[row - 1][column])) {
                res++;
            }

            if (row == area.length - 1 || !Objects.equals(num, area[row + 1][column])) {
                res++;
            }

            if (column == 0 || !Objects.equals(num, area[row][column - 1])) {
                res++;
            }

            if (column == area.length - 1 || !Objects.equals(num, area[row][column + 1])) {
                res++;
            }
            return res;
        }
    }
```