Laboratory work # 6

Student: HU Riqian
Student ID: 20321114
Timus Name: hduads2022_20321114
Mail: jhlxhrq@163.com

Problem # 1628. White Streaks

Screenshot from Timus:

9860919	13:44:23 6 May 2022	hduads2022_20321114	1628. White Streaks	Java 1.8	Accepted		0.203	13 228 KB

Explanation of algorithm:

1. I sort the input points first by row order, then find out how much white space is spanned between each point.

2. If there are at least two spaces in between, this constitutes one streak.

3. Do the same as above again, for the points sorted by column major order, and count how many singular points we've counted twice, add this to the answer, and we have the final answer.

Computational complexity of algorithm:

$$O(N^2)$$

Source code:

```java
import java.io.*;
import java.util.ArrayList;
import java.util.Collections;

public class WhiteStreaks {
    public static void main(String[] args) throws IOException {
        new WhiteStreaks().run();
    }

    StreamTokenizer in;
    PrintWriter out;
    static int M;
    static int N;
```

```
    static int K;

    ArrayList<Integer>[] xList = new ArrayList[60009];
    ArrayList<Integer>[] yList = new ArrayList[60009];

    int nextInt() throws IOException {
        in.nextToken();
        return (int) in.nval;
    }

    void run() throws IOException {
        in = new StreamTokenizer(new BufferedReader(new
InputStreamReader(System.in)));
        out = new PrintWriter(System.out);
        prepareData();
        solve();
        out.flush();
    }

    private void prepareData() throws IOException {
        M = nextInt();
        for (int i = 1; i <= M; i++) {
            xList[i] = new ArrayList<>();
        }

        N = nextInt();
        for (int i = 1; i <= N; i++) {
            yList[i] = new ArrayList<>();
        }
    }

    boolean judge(int x, int y) {
        return judgeArray(x, y, xList);
    }

    static boolean judgeArray(int x, int y, ArrayList<Integer>[] xList) {
        int left = 0, right = xList[x].size() - 1, mid, ans = 0;
        while (left <= right) {
            mid = (left + right) >> 1;
            if (xList[x].get(mid) <= y) {
                ans = mid;
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return xList[x].get(ans) == y && xList[x].get(ans + 1) == (y + 2);
    }

    void solve() throws IOException {
        K = nextInt();
        int a, b;
        for (int i = 0; i < K; i++) {
            a = nextInt();
            b = nextInt();
            xList[a].add(b);
            yList[b].add(a);
        }

        for (int i = 1; i <= M; i++) {
            xList[i].add(0);
            xList[i].add(N + 1);
            Collections.sort(xList[i]);
        }
```

```
        for (int i = 1; i <= N; i++) {
            yList[i].add(0);
            yList[i].add(M + 1);
            Collections.sort(yList[i]);
        }

        int sum = 0;
        for (int i = 1; i <= M; i++) {
            for (int j = 1; j < xList[i].size(); j++) {
                if (xList[i].get(j) - xList[i].get(j - 1) > 2) {
                    sum++;
                }
            }
        }

        for (int i = 1; i <= N; i++) {
            for (int j = 1; j < yList[i].size(); j++) {
                if (yList[i].get(j) - yList[i].get(j - 1) > 2) {
                    sum++;
                } else if (yList[i].get(j) - yList[i].get(j - 1) == 2) {
                    if (judge(yList[i].get(j) - 1, i - 1)) {
                        sum++;
                    }
                }
            }
        }
        out.println(sum);
    }
}
```