**Отчет**

по лабораторной работе «1628. Белые полосы»

по дисциплине «Алгоритмы и структуры данных»

**Авторы**:

Полит Алексей Денисович

**Факультет**: СУиР

**Группа**: R3235

**Преподаватель**: Тропченко Андрей Александрович


Санкт-Петербург, 2021г

1. Задача

   У каждого неудачника в жизни бывают не только чёрные, но и белые полосы. Марсианин Вась-Вась отмечает в календаре, представляющем собой таблицу $m \times n$, те дни, когда ему ужасно не повезло. Если Вась-Васю не повезло в $j$-й день $i$-й недели, то он закрашивает ячейку таблицы $(i, j)$ в чёрный цвет. Все незакрашенные ячейки в таблице имеют белый цвет.

   Будем называть отрезками жизни прямоугольники размером $1 \times l$ либо $l \times 1$. Белыми полосами Вась-Вась считает все максимальные по включению белые отрезки таблицы. А сможете ли Вы определить, сколько всего белых полос было в жизни Вась-Вася?

2. Исходные данные

   Первая строка содержит целые числа $m$, $n$, $k$ — размеры календаря и количество неудачных дней в жизни Вась-Вася ($1 \le m, n \le 30000$; $0 \le k \le 60000$). В следующих $k$ строках перечислены неудачные дни в виде пар ($xi$, $yi$), где $xi$ — номер недели, к которой относится неудачный день, а $yi$ — номер дня в этой неделе ($1 \le xi \le m$; $1 \le yi \le n$). Описание каждого неудачного дня встречается только один раз.

3. Код программы

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

bool comp1(std::pair<int, int> a, std::pair<int, int> b) {
    if (a.first != b.first) {
        return a.first < b.first;
    } else {
        return a.second < b.second;
    }
}

bool comp2(std::pair<int, int> a, std::pair<int, int> b) {
    if (a.second != b.second) {
        return a.second < b.second;
    } else {
        return a.first < b.first;
    }
}

int main() {
    int m, n, k, res = 0;
```

```
std::cin >> m >> n >> k;

std::vector<std::pair<int, int>> point;
std::vector<std::pair<int, int>> sq;

for (int i = 0; i < k; ++i) {
    std::pair<int, int> p;
    std::cin >> p.first >> p.second;
    p.first--;
    p.second--;
    point.push_back(p);
}

for (int i = 0; i < m; ++i) {
    std::pair<int, int> p = {i, -1};
    std::pair<int, int> p2 = {i, n};
    point.push_back(p2);
    point.push_back(p);
}

for (int i = 0; i < n; ++i) {
    std::pair<int, int> p = {-1, i};
    std::pair<int, int> p2 = {m, i};
    point.push_back(p2);
    point.push_back(p);
}

sort(point.begin(), point.end(), comp1);

for (int i = 0; i < point.size() - 1; ++i) {
    if (point[i].first == point[i + 1].first) {
        if (point[i + 1].second - point[i].second - 1 > 1) {
            res++;
        } else if (point[i + 1].second - point[i].second - 1 == 1) {
            std::pair<int, int> p(point[i].first, point[i].second + 1);
            sq.push_back(p);
        }
    }
}

sort(point.begin(), point.end(), comp2);
sort(sq.begin(), sq.end(), comp2);

for (int i = 0; i < point.size() - 1; ++i) {
    if (point[i].second == point[i + 1].second) {
        if (point[i + 1].first - point[i].first - 1 > 1) {
            res++;
        } else if (point[i + 1].first - point[i].first - 1 == 1) {
            std::pair<int, int> p(point[i].first + 1, point[i].second);
```

```
                sq.push_back(p);
            }
        }
    }

    sort(sq.begin(), sq.end(), comp1);

    if (sq.size() > 1) {
        int i = 0;
        while (i < sq.size() - 1) {
            if (sq[i] == sq[i + 1]) {
                res++;
                i++;
            }
            i++;
        }
    }

    std::cout << "\n\n" << res;

    return 0;
}
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 9350505 | 15:02:27<br>3 май 2021 | Aleksey | 1628. Белые полосы | G++ 9.2 x64 | Accepted | | 0.203 | 3 428 КБ |

4. Описание алгоритма

Отсортируем все точки с левого верхнего угла по правый нижний: сначала строки, потом столбцы. Каждую полосу, длиной больше 1 добавим в результат, каждую полосу, длиной 1 сохраним на будущее в вектор sq. Аналогично отсортируем все точки снова, только теперь сначала столбцы, потом строки. Теперь у нас есть количество вертикальных и горизонтальных полос. Осталось проверить только полосы 1x1. Если полоса 1x1 встречается в векторе sq 2 раза, значит - квадрат 1x1 максимален и по вертикали и по горизонтали => его следует добавить в результат