МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра прикладной математики, механики, управления и программного обеспечения**

**О Т Ч Е Т**

по лабораторной работе №7 по дисциплине “Структуры и алгоритмы компьютерной обработки данных”

|  |
| --- |
| Выполнил студент гр. Б8204  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. А. Федоренко |
| Проверил к.т.н. доцент  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.Н. Остроухова |

г. Владивосток

2018

**Неформальная постановка задачи**

Одна из площадей Архангельска замощена прямоугольными плитками размера 1 х k. Если ввести систему координат, так что левый нижний угол одной из плиток будет иметь координаты (0,0), то левые нижние углы плиток будут иметь координаты (i\*k + j, j) для всех целых i и j.

На площади было решено установить памятник известному архангельскому писателю и художнику Степану Писахову. Для установки памятника необходимо удалить все плитки, полностью или частично попадающие под основание. Основание памятника имеет форму многоугольника с целочисленными координатами вершин, все стороны которого параллельны осям координат. Известно, что любая прямая, пересекающая основание памятника и параллельная одной из осей координат, в пересечении с основанием образует один отрезок.

Для установки памятника необходимо выбрать место на площадке таким образом, чтобы количество удалённых плиток было минимальным. При выборе места основание разрешается только передвигать параллельно осям координат.

Требуется написать программу, вычисляющую минимальное количество плиток, которые придётся удалить.

**Формат входных данных**

Первая строка входных данных содержит два числа n и k — количество вершин в основании памятника и размер плитки.

Каждая из последующих n строк содержит два целых числа , - координаты i-ой вершины основания. Координаты перечислены в порядке обхода против часовой стрелки.

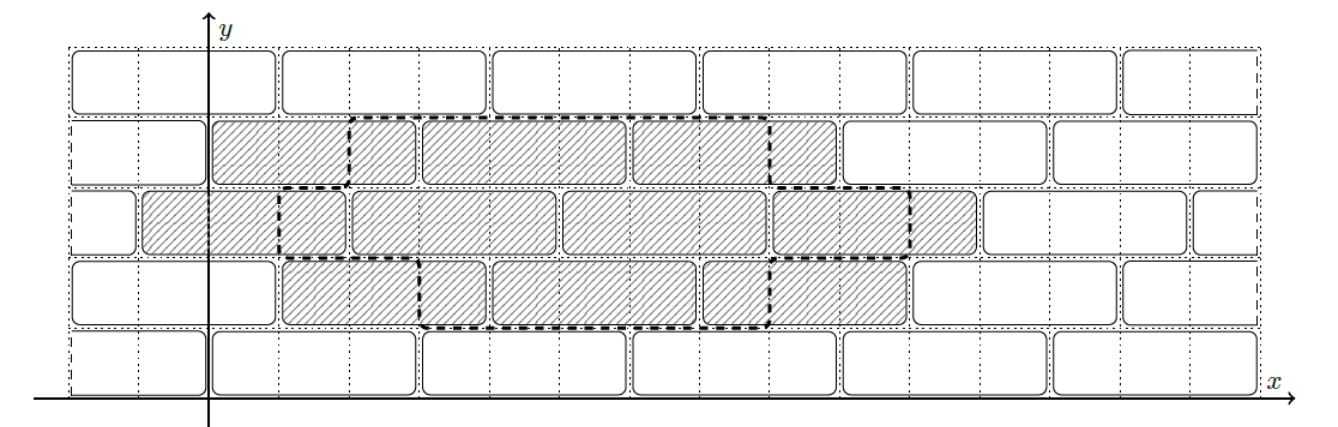
**Формат выходных данных**

Единственная строка выходных данных должна содержать минимально возможное количество плиток, которые необходимо удалить для размещения памятника на площади.

**Пример**

|  |  |
| --- | --- |
| monument.in | monument.out |
| 12 3  2 3  1 3  1 2  3 2  3 1  8 1  8 2  10 2  10 3  8 3  ­8 4  2 4 | 7 |

**Решение задачи**

- Заметим, что сдвиг на один по y-координате эквивалентен сдвигу на один по x-координате

- Будем перебирать сдвиги только по одной из координат

В данном примере n = 12 k = 3

y

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1 2 3 4 5 6 7 8 9 10 x

- Для каждой y-координаты посчитаем LEFTy — x-координату самой левой точки многоугольника с такой y-координатой

- Аналогично посчитаем RIGHTy

Для y = 1: LEFTy = 3 RIGHTy = 8

Для y = 2: LEFTy = 1 RIGHTy = 10

Для y = 2: LEFTy = 2 RIGHTy = 8

- Для каждой y-координаты посчитаем количество плиток, которые точно будут накрыты , с такой y-координатой:

Для y = 1: 3 плитки

Для y = 2: 4 плитки

Для y = 2: 3 плитки

В итоге накрыты 10 плиток

- Осталось сколько плиток освободятся при сдвиге основания памятника по оси x

- Когда x-координата левой границы прямоугольника станет кратна k, то пересечение слева исчезнет

- Аналогично, когда x-координата правой границы прямоугольника сравнима с единицей по модулю k, то появляется новое пересечение

При сдвиге на 0: ничего не поменяется

y

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1 2 3 4 5 6 7 8 9 10 x

При сдвиге на 1: исчезнет 3 пересечения

y

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1 2 3 4 5 6 7 8 9 10 11 x

При сдвиге на 2: исчезнет 1 пересечение

y

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1 2 3 4 5 6 7 8 9 10 11 12 x

При сдвиге на 3: основание вернётся в исходное положение

y

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1 2 3 4 5 6 7 8 9 10 11 12 13 x

- Теперь от общего количества накрытых плиток надо отнять максимальное из чисел пересечений, которые исчезнут при сдвиге основания памятника. Это и будет ответ

Ответ: 7

**Код программы**

#include <vector>

using namespace std;

int main() {

#ifndef LOCAL

freopen("monument.in", "r", **stdin**);

freopen("monument.out", "w", **stdout**);

#endif

int n, k;

scanf("%d%d", &n, &k);

vector <int> x(n), y(n);

int maxY = 0;

int minY = (int)1e9;

for (int i = 0; i < n; i++) {

scanf("%d%d", &x[i], &y[i]);

if (y[i] > maxY) {

maxY = y[i];

}

if (y[i] < minY) {

minY = y[i];

}

}

vector <int> minX(maxY + 1, (int)1e9);

vector <int> maxX(maxY + 1, -1);

for (int i = 0; i < n; i++) {

int ni = (i + 1) % n;

int y1 = y[i];

int y2 = y[ni];

if (y1 > y2) {

*// swap(y1, y2);*

int yyy = y1;

y1 = y2;

y2 = yyy;

}

for (int yy = y1; yy < y2; yy++) {

if (minX[yy] > x[ni]) {

minX[yy] = x[ni];

}

if (maxX[yy] < x[ni]) {

maxX[yy] = x[ni];

}

}

}

vector <int> ans(maxY + 1, 0);

long long carry = 0;

for (int yy = minY; yy < maxY; yy++) {

int from = minX[yy] + k - (yy % k);

int to = maxX[yy] + k - (yy % k);

ans[k - (from % k)]--;

ans[k - ((to + k - 1) % k)]++;

carry += max(0, ((to + k - 1) / k) - (from / k));

}

long long answer = ans[0];

for (int i = 1; i < k; i++) {

ans[i] += ans[i - 1];

if (ans[i] < answer) {

answer = ans[i];

}

}

printf("%lld\n", carry + answer);

return 0;

}