Mathias Hiron
Szymon Acedański
Task idea and formulation

Szymon Acedański, Łukasz Kowalik
Marcin Stefaniak
Analysis

Available memory: 32 MB, Maximum running time: 0.5 s.

Garden

Byteman owns the most beautiful garden in Bytetown. He planted n roses in his garden. Summer has come and the flowers have grown big and beautiful. Byteman has realized that he is not able to take care of all the roses on his own. He has decided to employ two gardeners to help him. He wants to select two rectangular areas, so that each of the gardeners will take care of the roses inside one area. The areas should be disjoint and each should contain exactly k roses.

Byteman wants to make a fence surrounding the rectangular areas, but he is short of money, so he wants to use as little fence as possible. Your task is to help Byteman select the two rectangular areas.

The garden forms a rectangle l meters long and w meters wide. It is divided into $l \cdot w$ squares of size 1 meter $\times$ 1 meter each. We fix a coordinate system with axes parallel to the sides of the garden. All squares have integer coordinates (x,y) satisfying $1 \leqslant x \leqslant l$, $1 \leqslant y \leqslant w$. Each square may contain any number of roses.

The rectangular areas, which must be selected, should have their sides parallel to the sides of the garden and the squares in their corners should have integer coordinates. For $1 \leqslant l_1 \leqslant l_2 \leqslant l$ and $1 \leqslant w_1 \leqslant w_2 \leqslant w$, a rectangular area with corners (l_1,w_1), (l_1,w_2), (l_2,w_1) and (l_2,w_2):

- *contains all the squares with coordinates (x,y) satisfying $l_1 \leqslant x \leqslant l_2$ and $w_1 \leqslant y \leqslant w_2$, and*
- *has perimeter $2 \cdot (l_2 - l_1 + 1) + 2 \cdot (w_2 - w_1 + 1)$.*

The two rectangular areas must be disjoint, that is they cannot contain a common square. Even if they have a common side, or part of it, they must be surrounded by separate fences.

Task

Write a program, that:

- *reads from the standard input the dimensions of the garden, the number of roses in the garden, the number of roses that should be in each of the rectangular areas, and the positions of the roses,*
- *finds the corners of two such rectangular areas with minimum sum of perimeters that satisfy the given conditions,*
- *writes to the standard output the minimum sum of perimeters of two non-overlapping rectangular areas, each containing exactly the given number of roses (or a single word* `NO`*, if no such pair of areas exists).*

Input

The first line of standard input contains two integers: l and w ($1 \leqslant l,w \leqslant 250$) separated by a single space — the length and the width of the garden. The second line contains two integers: n and k ($2 \leqslant n \leqslant 5000$, $1 \leqslant k \leqslant n/2$) separated by a single space — the number of roses in the garden and the number of roses that should be in each of the rectangular areas. The following n lines contain the coordinates of the roses, one rose per line. The $(i+2)$-nd line contains two integers l_i, w_i ($1 \leqslant l_i \leqslant l$, $1 \leqslant w_i \leqslant w$) separated by a single space — the coordinates of the square containing the i-th rose. Two or more roses can occur in the same square.

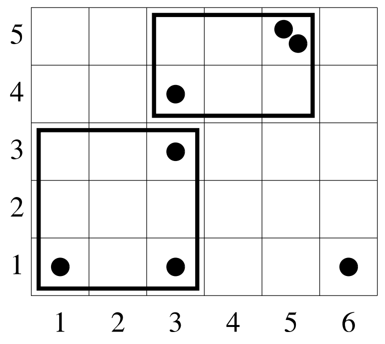
In 50% of test cases, the dimensions of the garden will satisfy $l,w \leqslant 40$.

Output

The standard output should contain only one line with exactly one integer — the minimum sum of perimeters of two non-overlapping rectangular areas, each containing exactly k roses, or a single word `NO`*, if no such pair of areas exists.*

Example

For the input data:
```
6 5
7 3
3 4
3 3
6 1
1 1
5 5
5 5
3 1
```
the correct result is:
```
22
```



Solution

Let us call a rectangular region containing exactly k roses a *k-rectangle*. Let us also denote by $r_{x,y}$ the number of roses in the square (x,y). The problem is to find two disjoint k-rectangles with minimal sum of perimeters.

The simplest solution is to consider all possible rectangular regions of the garden, and for each of them to count the number of roses inside. This way we can enumerate all k-rectangles in $O(w^3 \cdot l^3)$ time. There may be up to $O(w^2 \cdot l^2)$ k-rectangles in total. The second step is to consider all the pairs of k-rectangles and choose the one consisting of disjoint regions with minimum sum of perimeters. Such a solution should receive about 50 % points, despite its terrible time complexity of $O(w^4 \cdot l^4)$.

Model solution

Now we will present a number of gradual improvements, which will finally yield us the model solution. Please note, that we can optimize checking if a given rectangular region is a k-rectangle. Let us denote by $R_{x,y}$ the number of roses in a region, with one corner at $(1,1)$ and the opposite one at (x,y). We can precompute all the values of $R_{x,y}$ iteratively in $O(w \cdot l)$ time using the following formula:

$$R_{x,y} = \begin{cases} 0 & \text{if } x = 0 \text{ or } y = 0 \\ R_{x-1,y} + R_{x,y-1} - R_{x-1,y-1} + r_{x,y} & \text{otherwise} \end{cases}$$

Having this, we can express $\mathcal{R}(x_1,y_1,x_2,y_2)$ — the number of roses in a rectangular region with corners (x_1,y_1) and (x_2,y_2) as:

$$\mathcal{R}(x_1,y_1,x_2,y_2) = R_{x_2,y_2} - R_{x_2,y_1-1} - R_{x_1-1,y_2} + R_{x_1-1,y_1-1}$$

This way, $\mathcal{R}(x_1,y_1,x_2,y_2)$ can be evaluated in $O(1)$ time. Using the presented method, we can enumerate all k-rectangles in $O(w^2 \cdot l^2)$ time. Unfortunately, this does not solve the problem of considering all pairs of k-rectangles.

But fortunately, there is another method, which copes with this problem. Please observe, that if we have two disjoint rectangular regions, then there must exist either a horizontal or a vertical line such that one rectangle is above it (or respectively to the left) and the other one is below it (or respectively to the right). Hence, for each horizontal line we can find two k-rectangles with the smallest perimeters, laying on the opposite sides of the line. Similar values are to be found for all vertical lines. When we have done this, we can easily compute the final result in $O(w+l)$ by considering all the possible dividing lines and choosing the result which gives us the optimal sum of perimeters.

Now we will show how to find optimal perimeters for the first case (rectangular regions above the given horizontal line). The three other cases can be solved analogously. Let us denote by A_y the minimal perimeter of k-rectangle laying above the horizontal line with the given y-coordinate, whose bottommost coordinate is greater than or equal y. Let us also denote by a_y the minimal perimeter of the k-rectangle with bottommost coordinate equal y. Please note, that:

$$A_y = \min(a_y, \ldots, a_w)$$

A simple way to calculate a_i's is to initially set them to infinity, and then update them while iterating through all k-rectangles. With this improvement our algorithm works in $O(w^2 \cdot l^2)$ time.

This is not all. Please note, that we do not need to consider *all* k-rectangles. We can limit our considerations to those k-rectangles, which do not contain any other k-rectangles in their interiors. To enumerate all interesting k-rectangles (and maybe some not interesting too), we consider all pairs (y_1,y_2), $1 \leqslant y_1 \leqslant y_2 \leqslant w$. For each such pair, we use a *sliding window*, which is a rectangle having corners at (x_1,y_1) and (x_2,y_2). At the beginning, $x_1 = x_2 = 1$. Then we repeatedly move the sliding window according to the following rules, until $x_2 > l$:

- if there are exactly k roses in the sliding window (i.e. $\mathcal{R}(x_1, y_1, x_2, y_2) = k$), then we have found a new k-rectangle; after updating the four constructed sequences (a_i and the three other analogous sequences), x_1 is incremented by one,
- if $\mathcal{R}(x_1, y_1, x_2, y_2) < k$ then x_2 is incremented by one, to expand the sliding window,
- if $\mathcal{R}(x_1, y_1, x_2, y_2) > k$ then x_1 is incremented by one, to shrink the sliding window,

The above algorithm works in $O(w^2 \cdot l)$ time, and enumerates (among others) all interesting k-rectangles. Of course, we can reduce its running time to $O(w \cdot l \cdot \min(w, l))$ by adapting the direction, in which this method works.

The presented solution, with all the above optimizations, constitutes the model solution.