International Olympiad in Informatics 2016
12-19th August 2016
Kazan, Russia
day2_3

`aliens`
Country: ISC

Aliens

Our satellite has just discovered an alien civilization on a remote planet. We have already obtained a low-resolution photo of a square area of the planet. The photo shows many signs of intelligent life. Our experts have identified n points of interest in the photo. The points are numbered from 0 to $n-1$. We now want to take high-resolution photos that contain all of those n points.

Internally, the satellite has divided the area of the low-resolution photo into an m by m grid of unit square cells. Both rows and columns of the grid are consecutively numbered from 0 to $m-1$ (from the top and left, respectively). We use (s, t) to denote the cell in row s and column t. The point number i is located in the cell (r_i, c_i). Each cell may contain an arbitrary number of these points.

Our satellite is on a stable orbit that passes directly over the *main* diagonal of the grid. The main diagonal is the line segment that connects the top left and the bottom right corner of the grid. The satellite can take a high-resolution photo of any area that satisfies the following constraints:
  - the shape of the area is a square,
  - two opposite corners of the square both lie on the main diagonal of the grid,
  - each cell of the grid is either completely inside or completely outside the photographed area.

The satellite is able to take at most k high-resolution photos.

Once the satellite is done taking photos, it will transmit the high-resolution photo of each photographed cell to our home base (regardless of whether that cell contains some points of interest). The data for each photographed cell will only be transmitted *once*, even if the cell was photographed several times.

Thus, we have to choose at most k square areas that will be photographed, assuring that:
  - each cell containing at least one point of interest is photographed at least once, and
  - the number of cells that are photographed at least once is minimized.

Your task is to find the smallest possible total number of photographed cells.

Implementation details

You should implement the following function (method):
  - `int64 take_photos(int n, int m, int k, int[] r, int[] c)`
    - `n`: the number of points of interest,
    - `m`: the number of rows (and also columns) in the grid,

- ○ k: the maximum number of photos the satellite can take,
- ○ r and c: two arrays of length n describing the coordinates of the grid cells that contain points of interest. For $0 \le i \le n-1$, the i-th point of interest is located in the cell (r[i], c[i]),
- ○ the function should return the smallest possible total number of cells that are photographed at least once (given that the photos must cover all points of interest).

Please use the provided template files for details of implementation in your programming language.

Examples

Example 1

```
take_photos(5, 7, 2, [0, 4, 4, 4, 4], [3, 4, 6, 5, 6])
```

In this example we have a 7×7 grid with 5 points of interest. The points of interest are located in four different cells: $(0,3)$, $(4,4)$, $(4,5)$ and $(4,6)$. You may take at most 2 high-resolution photos.

One way to capture all five points of interest is to make two photos: a photo of the 6×6 square containing the cells $(0,0)$ and $(5,5)$, and a photo of the 3×3 square containing the cells $(4,4)$ and $(6,6)$. If the satellite takes these two photos, it will transmit the data about 41 cells. This amount is not optimal.

The optimal solution uses one photo to capture the 4×4 square containing cells $(0,0)$ and $(3,3)$ and another photo to capture the 3×3 square containing cells $(4,4)$ and $(6,6)$. This results in only 25 photographed cells, which is optimal, so take_photos should return 25.

Note that it is sufficient to photograph the cell $(4,6)$ once, even though it contains two points of interest.

This example is shown in the figures below. The leftmost figure shows the grid that corresponds to this example. The middle figure shows the suboptimal solution in which 41 cells were photographed. The rightmost figure shows the optimal solution.
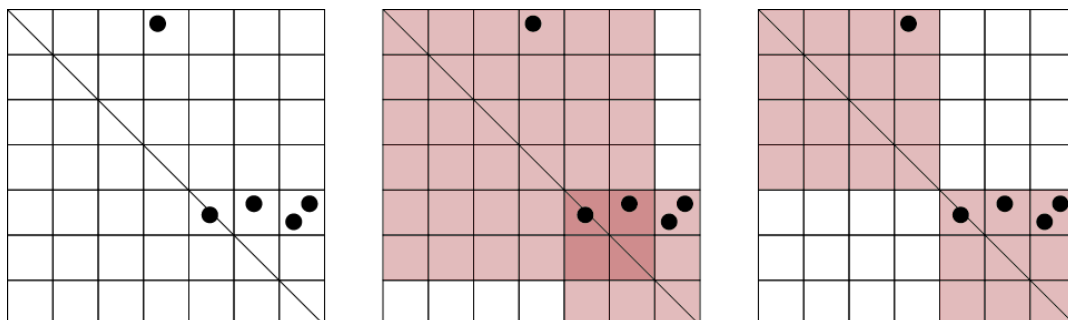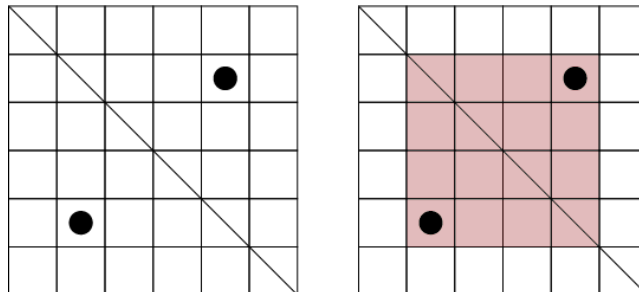

Example 2

```
take_photos(2, 6, 2, [1, 4], [4, 1])
```

Here we have 2 points of interest located symmetrically: in the cells $(1, 4)$ and $(4, 1)$. Any valid photo that contains one of them contains the other one as well. Therefore, it is sufficient to use a single photo.

The figures below show this example and its optimal solution. In this solution the satellite captures a single photo of 16 cells.

Subtasks

For all subtasks, $1 \le k \le n$.
1. (4 points) $1 \le n \le 50$, $1 \le m \le 100$, $k = n$,
2. (12 points) $1 \le n \le 500$, $1 \le m \le 1000$, for all i such that $0 \le i \le n - 1$, $r_i = c_i$,
3. (9 points) $1 \le n \le 500$, $1 \le m \le 1000$,
4. (16 points) $1 \le n \le 4000$, $1 \le m \le 1\,000\,000$,
5. (19 points) $1 \le n \le 50\,000$, $1 \le k \le 100$, $1 \le m \le 1\,000\,000$,
6. (40 points) $1 \le n \le 100\,000$, $1 \le m \le 1\,000\,000$.

Sample grader

The sample grader reads the input in the following format:
- line 1: integers n, m and k,
- line $2 + i$ ($0 \le i \le n - 1$): integers r_i and c_i.