Aliens

Bizim uydumuz uzaklarda yeni bir gelişmiş bir alien medeniyeti olan gezegen buldu. Biz şimdididen gezegenin kare şeklindeki bir kısmının düşük çözünürlüklü fotografını aldık. Fotograf zekice bir hayat yaşadıklarının birçok işaretini taşıyor. Uzmanlarımız fotografta n tane ilgi çeken nokta belirlemişler. Şimdi biz o n tane noktanın hepsini içine alan yüksek çözünürlüklü fotografını istiyoruz.

Uydu o düşük çözünürlüklü fotoyu m x m (grid) kareye böldü. Hem satır hem sütunlar yukarıdan ve soldan başlayarak 0-da $m-1$-e kadarsırasıyla numaralandırılmış. Biz i satır ve j sütun için (i, j) kullanacağız. Bütün ilgi çeken noktalar bu m x m cellerinin içinde kalıyor. Her cellde bu noktalardan rastgele sayıda var.

Uydumuz gridin tam *main* diagonal-ı üzerinden geçen bir yörüngede (orbitte) sabit duruyor. Main diagonal (i, i), $0 \leq i \leq m-1$ celleri üzerinden geçen diagonaldir. Uydu aşağıdaki şartları sağlayan bütün alanların yüksek çözünürlüklü fotosunu alabiliyor:

- alanın şekli kare şeklinde olmalı,
- karenin bir diagonali gridin main diagonalinin tamamen içinde olmalı,
- grid-in her cell-i fotosu çekilmiş alanın ya tamamen içinde ya da tamamen dışında olmalı.
  Uydu en fazla k high-resolution photo çekebiliyor.

Uydu fotoyu çekince, o fotosu çekilmiş her cell-in high-resolution photosunu bize gönderecek (ilgi çeken nokta içerip içermediğine bakmaksızın). Her fotosu çekilmiş cell sadece *bir* kere gönderilecek, fotosu birkaç kere çekilmiş olsa dahi.

Onun için, fotosu çekilecek en fazla k kare alan seçmeliyiz, şu şartlarla:

- en az bir ilgi çeken nokta içeren her cell en az bir kere fotosu çekilmeli, ve
- en az bir kere fotosu çekilecek cell sayısı minimuma çekilmeli.
  Yapmanız gereken bu sayıyı bulmak.

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 $0 \leq i \leq$ `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 photo 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])
```

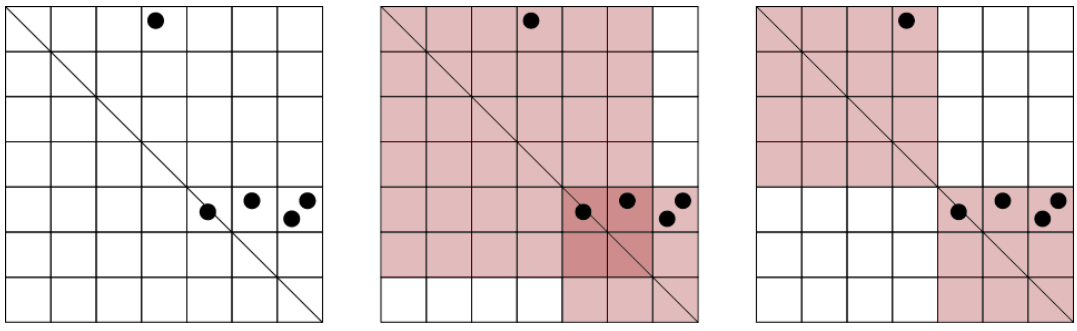
In this example we have a 7×7 grid with 5 points of interest. The points of interest are located in four different cells: $(0, 3)$, $(4, 4)$, $(4, 5)$ and $(4, 6)$. You may take at most 2 high-resolution photos.

One way to capture all five points of interest is to make two photos: one with the cells $(0, 0)$ and $(5, 5)$ in the opposite corners, and the other with cells $(4, 4)$ and $(6, 6)$ in the opposite corners. If we take these two photos, the satellite will transmit the photos of 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.

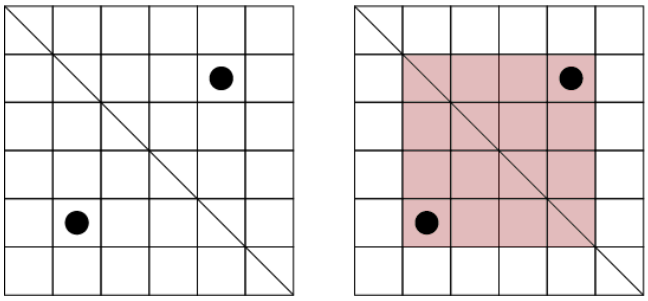
Both ways of taking the photos are depicted below. The figure on the right shows the optimal solution.



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 optimal solution (depicted below) captures the photo of 16 cells.



Subtasks

For all subtasks, $1 \leq k \leq n$.

1. (4 points) $1 \leq n \leq 50$, $1 \leq m \leq 100$, $k = n$.
2. (12 points) $1 \leq n \leq 500$, $1 \leq m \leq 1000$, for all i such that $0 \leq i \leq n-1$, $r_i = c_i$.
3. (9 points) $1 \leq n \leq 500$, $1 \leq m \leq 1000$,
4. (16 points) $1 \leq n \leq 2000$, $1 \leq m \leq 1\,000\,000$,
5. (12 points) $1 \leq n \leq 20\,000$, $1 \leq k \leq 100$, $1 \leq m \leq 1\,000\,000$,
6. (7 points) $1 \leq n \leq 20\,000$, $1 \leq k \leq 1000$, $1 \leq m \leq 1\,000\,000$,
7. (40 points) $1 \leq n \leq 200\,000$, $1 \leq m \leq 1\,000\,000$.

Sample grader

The sample grader reads the input in the following format:

- line 1: integers n, m and k,
- line 2 + i ($0 \leq i \leq n-1$): integers r_i and c_i.