Aliens

Onze satelliet heeft zojuist buitenaards leven ontdekt op een afgelegen planeet. We hebben al een foto in een lage resolutie van een vierkant gebied van de planeet kunnen maken. De foto toont sporen van intelligent leven. Onze experts hebben n interessante plekken kunnen ontdekken op de foto. De plekken zijn genummerd van 0 tot en met $n-1$. Nu willen we hoge-resolutie foto's maken van die n plekken.

In de camera van de satelliet is het gebied van de lage-resolutie foto verdeeld in een rooster van m bij m vierkante eenheidscellen. Zowel de rijen als de kolommen van het rooster zijn genummerd van 0 tot en met $m-1$ (van boven naar beneden en van links naar rechts). We gebruiken (s,t) om de cel van rij s en kolom t aan te geven. De plek met nummer i bevindt zich in cel (r_i, c_i). In elke cel kan een willekeurig aantal interessante plekken zijn.

Onze satelliet zit in een stabiele baan die steeds over de *hoofddiagonaal* van het rooster beweegt. De hoofddiagonaal is het lijnstuk dat van de linkerbovenhoek naar de rechteronderhoek gaat. De satelliet kan hoge-resolutie foto's maken van elk gebied dat voldoet aan de volgende voorwaarden:

- het is een vierkant gebied,
- Twee tegenoverliggende punten van het vierkant liggen beiden op de hoofddiagonaal van het rooster
- elke cel van het rooster zit óf geheel binnen óf geheel buiten het gefotografeerde gebied.

De satelliet kan maximaal k hoge-resolutie foto's maken.

Zodra de satelliet klaar is met het maken van de foto's, zal het de hoge-resolutie foto's van iedere gefotografeerde cel doorsturen naar de aarde (ongeacht of de cel interessante plekken bevat of niet). De data van elke gefotografeerde cel zal slechts *eenmaal* doorgestuurd worden, zelfs als de cel meerdere keren is gefotografeerd.

Je moet maximaal k vierkante gebieden kiezen om te fotograferen, zodanig dat:

- elke cel die een interessante plek bevat minstens eenmaal gefotografeerd wordt, en
- het aantal cellen dat minstens een keer gefotografeerd wordt minimaal is.

Het is jouw taak om het kleinst mogelijke aantal gefotografeerde cellen te vinden.

Implementatie details

Je moet de volgende functie (methode) implementeren:

- `int64 take_photos(int n, int m, int k, int[] r, int[] c)`
  - `n`: het aantal interessante plekken,

- `m`: het aantal rijen (en kolommen) in het rooster,
- `k`: het maximale aantal foto's dat de satelliet mag maken,
- `r` en `c`: twee arrays met lengte n die de coördinaten bevatten van de cellen met interessante plekken. Voor $0 \leq i \leq \texttt{n} - 1$, geldt dat de i'de interessante plek ligt in cel (`r[i]`, `c[i]`),
- de functie moet het kleinst mogelijke totale aantal cellen terug geven dat minstens eenmaal gefotografeerd wordt (waarbij dat de foto alle interessante plekken omvat).

Gelieve de voorziene template bestanden te gebruiken voor de implementatiedetails in je programmeertaal.

Voorbeelden

Voorbeeld 1

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

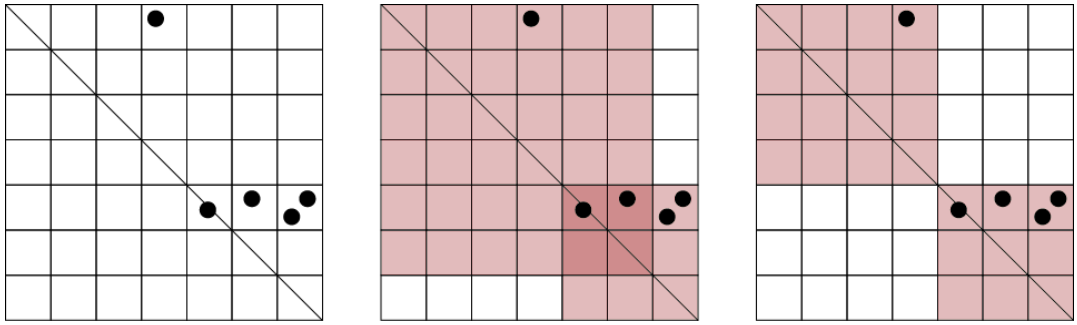
In dit voorbeeld hebben we een 7×7 rooster met 5 interessante plekken. Deze plekken bevinden zich in vier verschillende cellen: $(0, 3)$, $(4, 4)$, $(4, 5)$ en $(4, 6)$. Je mag maximaal 2 hoge-resolutie foto's maken.

Een manier om alle vijf interessante plekken te fotograferen met twee foto's is: een foto van een 6×6 vierkant met de cellen $(0, 0)$ en $(5, 5)$ en foto van een vierkant 3×3 met de cellen $(4, 4)$ en $(6, 6)$. Als we deze twee foto's maken, zal de satelliet de data van 41 cellen doorsturen. Deze oplossing is niet optimaal.

De optimale oplossing is om een 4×4 vierkant te fotograferen met de cellen $(0, 0)$ en $(3, 3)$ en een tweede foto van het 3×3 vierkant met de cellen $(4, 4)$ en $(6, 6)$. Dit levert 25 gefotografeerde cellen op en `take_photos` moet dus 25 terug geven.

Merk op dat cel $(4, 6)$ slechts eenmaal gefotografeerd hoeft te worden, ook al bevat het twee interessante plekken.

Dit voorbeeld is te zien in de afbeeldingen hieronder. De linker afbeelding laat het rooster zien dat bij dit voorbeeld hoort. De middelste afbeelding laat de suboptimale oplossing zien waarbij 41 cellen gefotografeerd worden. De rechter figuur laat de optimale oplossing zien.

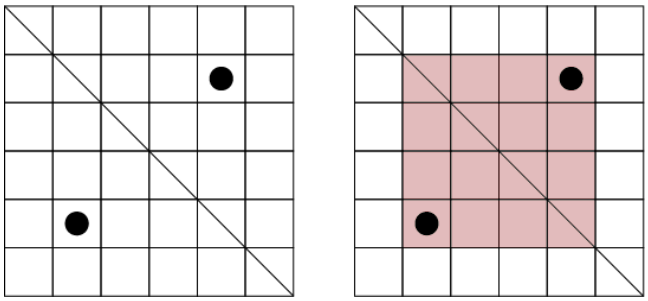


Voorbeeld 2

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

Hier hebben we 2 interessante plekken die symetrisch liggen: in de cellen $(1, 4)$ en $(4, 1)$. Elke foto die één van de twee cellen fotografeert, heeft ook automatisch de andere meegenomen. Daarom is het voldoende om één foto te maken.

De afbeeldingen hieronder laten dit voorbeeld en de bijbehorende optimale oplossing zien. In deze oplossing zijn 16 cellen gefotografeerd met één foto.



Subtaken

Voor alle subtaken, $1 \leq k \leq n$.

1. (4 punten) $1 \leq n \leq 50$, $1 \leq m \leq 100$, $k = n$,
2. (12 punten) $1 \leq n \leq 500$, $1 \leq m \leq 1000$, voor alle i waarvoor geldt $0 \leq i \leq n - 1$, $r_i = c_i$,
3. (9 punten) $1 \leq n \leq 500$, $1 \leq m \leq 1000$,
4. (16 punten) $1 \leq n \leq 4000$, $1 \leq m \leq 1\,000\,000$,
5. (19 punten) $1 \leq n \leq 50\,000$, $1 \leq k \leq 100$, $1 \leq m \leq 1\,000\,000$,
6. (40 punten) $1 \leq n \leq 100\,000$, $1 \leq m \leq 1\,000\,000$.

Voorbeeld grader

De voorbeeld grader leest de input in in het volgende formaat:

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