French-Australian Regional Informatics Olympiad

Friday 13th March, 2015

Duration: 4 hours

3 questions

All questions should be attempted

Problem 1
Pam-Can Retires

Input File: *standard input*
Output File: *standard output*

Time and Memory Limits: 1 second, 256 MB

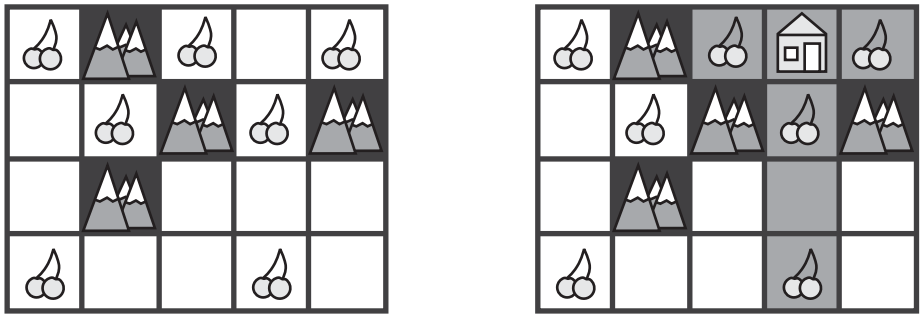
After spending the last thirty-five years roaming maze after maze and fleeing from ghosts and goblins you – the once world famous Pam-Can – have decided to retire somewhere in the blissful estate known only as *Level 257*.

Level 257 is a grid of cells with R rows and C columns. The rows are numbered 1 to R from top to bottom and columns are numbered 1 to C from left to right. Every cell is either blocked off by uninhabitable mountains (a *block cell*), contains fruit (a *fruit cell*) or is completely empty.

You resolve to pick precisely one of these cells as your *retirement cell*. This cell should either be a fruit cell or completely empty – it cannot be a block cell. From it, you may be able to see other grid cells. Specifically, a cell is *visible* from your retirement cell if and only if:

- The cell is either in the same row **or** in the same column as your retirement cell; **and**
- There are no block cells on the straight line between the cell and your retirement cell.

Note that your retirement cell is always visible from itself. The diagram below shows a possible layout for *Level 257* (on the left) and all visible cells (shaded) if you choose your retirement cell to be at row 1, column 4 on the grid (on the right).



As you ponder your future you ask yourself: "What is the maximum number of fruit cells that could be visible from my retirement cell?" You shudder at the thought of never seeing your beloved fruit again so you decide to write a program to answer this question for you.

Input

- The first line of input will contain one line with four space-separated integers R C B F, representing the number of rows and columns in the grid, respectively, and the number of block cells and fruit cells in the grid, respectively.
- The following B lines of input will each contain two space-separated integers describing the row and column of a block cell.
- The following F lines of input will each contain two space-separated integers describing the row and column of a fruit cell.

Every cell will be described **at most once** in the input. Every cell that is not described in the input is completely empty.

Output

Your program must output a single integer: the maximum number of fruit cells that are visible from a valid retirement cell. You are guaranteed that there is at least one valid retirement cell in every testcase.

Sample Input

```
4 5 4 7
2 3
3 2
2 5
1 2
2 2
1 1
1 3
4 4
1 5
2 4
4 1
```

Sample Output

```
4
```

Explanation

If you retire to row 1, column 4, you will be able to see the fruit at coordinates $(1, 3), (1, 5), (2, 4), (4, 4)$ for a total of 4 fruit. Note that you will not be able to see the fruit at location $(1, 1)$ as it is blocked by location $(1, 2)$. No other square can see more fruit, therefore the answer is 4.

Subtasks & Constraints

For all subtasks, $1 \leq R, C \leq 100\,000$ and $0 \leq B + F \leq 100\,000$. Additionally, all cells have a row number between 1 and R (inclusive) and a column number between 1 and C (inclusive).

- For Subtask 1 (10 points), $1 \leq R, C \leq 100$.
- For Subtask 2 (50 points), $1 \leq R, C \leq 1\,000$.
- For Subtask 3 (15 points), $B = 0$. That is, there are no block cells.
- For Subtask 4 (25 points), no further constraints apply.

Problem 2
The Nightlight

Input File: *standard input*
Output File: *standard output*

Time and Memory Limits: 2 seconds, 512 MB

"Every night I roam the city blocks of Gotham, silently ridding my path of crime, bringing light to a frightened city. Up until now I have been undetected, but there are powerful people who will stop at nothing to extinguish *The Nightlight*."

On Gotham's R rows by C columns grid of city-blocks, *The Nightlight* starts in the top-left block and has to finish at the bottom-right block. To retain the element of surprise *The Nightlight* only moves to adjacent squares either down or to the right. Finding such a path would be easy but each block has a colour - and she can only move to a block of the same colour as the one she is standing on. Hope is not lost though, she can use her network of spies at any time to change the colour of her block or an adjacent block so that she can move between them - but this takes time - time that Gotham doesn't have.

Your task is to find the minimum total number of times the colour of a block has to be changed in order for *The Nightlight* to walk to the bottom-right block.

Input

- The first line of input will contain one line with three space-separated integers R C K, respectively representing the number of rows and columns in the grid, and the number of different colours of blocks.
- The following R lines of input will each contain C space-separated integers representing the colour of the blocks in the current row from left to right. Each of these C integers will be between 1 and K inclusive.

The first of the R lines represents the 'top' line, and the final line represents the 'bottom', similarly the first integer in each of the R lines is the 'left', and the last is the 'right'.

Output

Your program must output a single integer: the minimum total number of times the colour of a block has to be changed in order for *The Nightlight* to be able to pass from the top-left block to the bottom-right block.

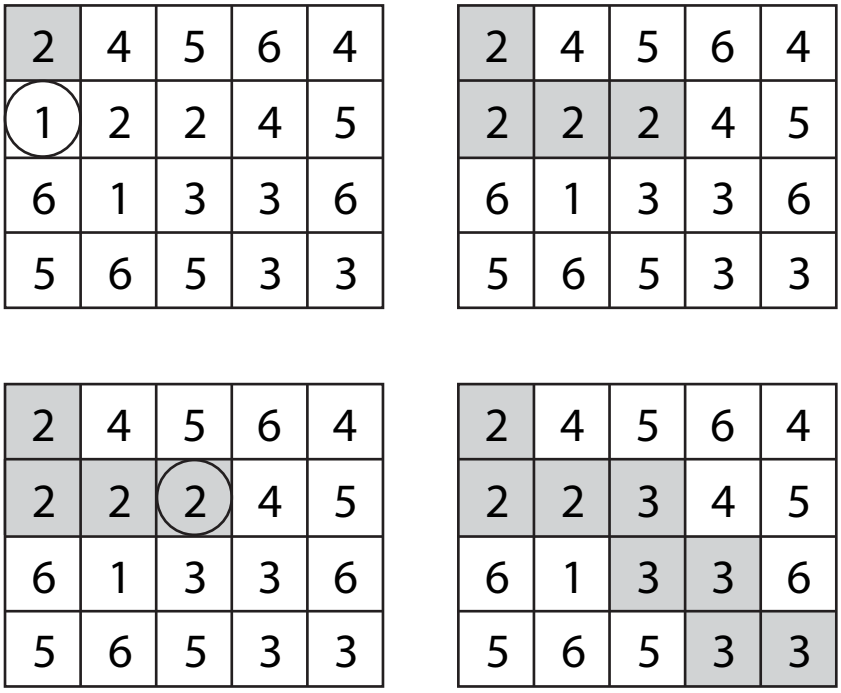
Sample Input

```
4 5 6
2 4 5 6 4
1 2 2 4 5
6 1 3 3 6
5 6 5 3 3
```

Sample Output

```
2
```

Explanation



Before moving off the top-left block *The Nightlight* can change the colour of the block directly beneath her from 1 to 2. This gives her safe passage to that block, and then to the right twice. Then *The Nightlight* can change the colour of the block she is standing on to 3. This allows *The Nightlight* to walk to the bottom-right block. This path involved changing the colour of squares twice, so 2 is the answer.

Subtasks & Constraints

For all subtasks, $1 \leq R \leq 1\,000$, $1 \leq C \leq 100\,000$ and $1 \leq K \leq 1\,000\,000$.

- For Subtask 1 (20 points), $1 \leq K \leq 10$ and $C \leq 1\,000$.
- For Subtask 2 (30 points), $R = 1$.
- For Subtask 3 (30 points), $1 \leq K \leq 1\,000$ and $C \leq 1\,000$.
- For Subtask 4 (20 points), $C \leq 1\,000$.

Problem 3
Frequent Flyer

Input File: *standard input*
Output File: *standard output*

Time and Memory Limits: 1 second, 256 MB

Congratulations! You've just become an official card-carrying member of Flagship Air Rewards.

Here at Flagship Air we serve N different cities (numbered from 1 to N) helping you to travel to wherever your heart leads you. We operate a **single flight** in **one direction** between **every pair** of cities. Specifically, between every pair of distinct cities i and j, we either operate a flight from i to j, or, from j to i; but not both. To thank you for your loyalty, every time you fly with us we will credit points to your account depending on which flight you take. See the *Input* section below for details.

To help you kickstart your account, we highly recommend you to relax on a "Flagship Trip". A "Flagship Trip":

- Starts in a city of your choosing;
- Ends in a different city;
- Uses exactly $N - 1$ Flagship Air flights to move between cities; and
- Visits each of the N cities **precisely once**.

As we understand that you may be far too lazy to plan out your own "Flagship Trip" by hand, we encourage you to write a program that does this for you. Additionally, your program should attempt to maximise the total number of points that you will accrue over the entire course of the trip.

Please note that your program **does not** have to find the optimal "Flagship Trip" in every case; see the *Scoring* section below for details.

We look forward to flying with you soon!

Input

- The first line of input will contain one integer N, representing the total number of cities that Flagship Air flies between.
- The following N lines of the input will each contain N integers separated by spaces. The jth integer on the ith of these lines contains `-1` if there is no Flagship Air flight from i to j and the number of points gained by taking this flight, otherwise.

Between every distinct pair of cities, the direction of the flight between them is determined uniformly at random (there is a 50% chance that the flight will go in either direction). Additionally, for every flight the number of points available on it is a uniformly random integer between 0 and 1 000 000, inclusive.

Output

Your program must output N lines each containing exactly one integer. These lines describe a valid "Flagship Trip". The kth of these lines should contain a single integer: the number of the kth city visited on the trip. Each integer from 1 to N should appear precisely once in the output.

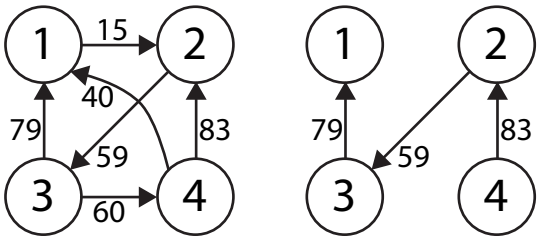
Sample Input

```
4
-1 15 -1 -1
-1 -1 59 -1
79 -1 -1 60
40 83 -1 -1
```

Sample Output

```
4
2
3
1
```

Explanation



By examination of all possible "Flagship Trips", we find that we collect the most points by going from 4 to 2, 2 to 3, and finally 3 to 1. This trip scores $83 + 59 + 79 = 221$ for its flights. Please note that this sample input was *not* generated according to the random procedure described in the *Input* section above and thus **will not** be used for grading.

Subtasks & Constraints

For all subtasks, $2 \leq N \leq 100$.

- For Subtask 1 (20 points), $2 \leq N \leq 10$.
- For Subtask 2 (80 points), no additional constraints.

Scoring

For each test case, if your program does not output a valid trip, it will score 0%. Otherwise, your score will be determined from the number of points gained by following your program's trip and the number of points gained by following the judges' trip. Specifically, if your trip gains x points and the judges' trip gains y points then your score (as a percentage) is calculated as:

$$\text{score} = \min(\left\lfloor 20 + 75e^{15\left(\frac{x-y}{y}\right)} \right\rfloor, 100)$$

Where the mathematical constant $e \approx 2.71828$. Note that for each test case:

- A valid trip always scores at least 20%; and
- The judges' trip scores 95%.