French-Australian Regional Informatics Olympiad

Friday 15th March, 2013

Duration: 4 hours

3 questions

All questions should be attempted

Problem 1
Algo-fu

Time and Memory Limits: 3 seconds, 128 MB

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

You are on your way to the algo-fu temple, hidden in the mountains, where you plan to become a master of algo-fu. To receive the teachings of the monks, you first need to prove that your heart is pure.

You are given a map that represents the mountains as a rectanglular grid of squares, each having a height. You may move from your current square to any adjacent square on the map (east, west, north or south), as long as the new square's height is lower or equal to the purity value of your heart.

As you arrive into the top-left corner of the grid (which is row 1, column 1, and is guaranteed to have a height of 0) your heart is already tainted by the impurity of modern civilization, thus has a purity value of 0. Some of the squares may offer you an opportunity to perform a task which will add a certain amount of purity to your heart, allowing you to reach new heights. Your goal is to minimize the number of tasks that you will have to perform to reach the temple.

Input

- The first line contains three integers: R, C and T: the number of rows and columns of the map, and the number of tasks available.
- Each of the next R lines contains C integers: the heights of the corresponding squares on the map.
- Each of the next T lines contains three integers: the row and column of the task on the map, and the purity value gained when performing the task. There can be at most one task in each square, and a given task can only be performed once.
- The last line contains two integers: the row and column of the temple on the map.

Output

Output should consist of one line containing one integer: the minimum number of tasks that you need to perform in order to move to the square of the temple. If it is impossible to reach the temple, output `-1`.

Sample Input

```
5 4 4
0 0 5 3
1 2 3 4
0 1 6 7
2 3 9 5
1 3 8 3
1 2 1
3 1 1
3 2 2
5 2 4
4 4
```

Sample Output

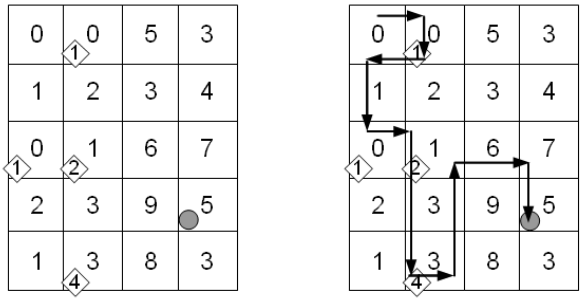
```
3
```

Explanation

The map consists of 5 rows and 4 columns, with heights ranging from 0 to 9.

There are 4 tasks available, all in the first two columns of the map, as shown on the left side of the illustration below. The temple is in the last column, on the 4th row. It can be reached by performing the 3 tasks of the second column, following the path shown on the right side of the illustration.

The first task adds 1 to the purity value of your heart. It allows you to reach the second task of colum 2, which adds 2 to your purity, for a total of 3. You can then reach the third task of column 2, which adds 4 to your purity, for a total of 7, enough for you to reach the temple.



Subtasks

For all subtasks, $R \geq 1$, $C \geq 1$, $T \geq 0$, all heights are at least 0, task purities are at least 1, and all task and temple positions have a row between 1 and R inclusive, and a column between 1 and C inclusive.

- For Subtask 1 (25 pts), $R, C \leq 100$, heights ≤ 100, $T \leq 1\,000$, task purities ≤ 100.
- For Subtask 2 (25 pts), $R, C \leq 1\,000$, heights ≤ 1000, $T \leq 1\,000$, task purities ≤ 100.
- For Subtask 3 (50 pts), $R, C \leq 1\,000$, heights $\leq 100\,000$, $T \leq 100\,000$, task purities $\leq 1\,000$.

Problem 2
Super Maria

Time and Memory Limits: 1 second, 128 MB

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

Maria is an electrician who lives in the Fungus Republic, where every few years disaster strikes and Maria always ends up being the one who must restore order. This year her evil sister Waria has kidnapped her Prince Nectarine, so she is once again on a long and confusing quest to save him.

Maria is called Super Maria by her friends because she posseses three super-powers:

- The amazing ability to run to the left.
- The amazing ability to run to the right.
- The ordinary ability to instantly transport herself **from any position** to a *teleport receiver*. After Maria teleports to a receiver, it is depleted of energy and can not be used again. Hence, Maria can teleport to each receiver at most once.

For some unexplained reason, Maria's quest requires her to collect a sequence of gold coins which are conveniently lying on a flat straight line (a *platform*). There is also at least one teleport receiver on this platform. Maria needs to **first teleport to one of the receivers** as she is not initially on the platform, then use her three super-powers to move around and collect all the coins. Maria doesn't like running around on a platform collecting things like some silly plumber, so she has asked you to write a program to calculate the minimum total distance she must run.

Maria doesn't care which receiver she starts from, or where she ends up on the platform after collecting all the coins. No two coins will be at the same position and no two receivers will be at the same position (however a coin and a receiver may share a position).

Input

- The first line of input will contain two integers C and R: the number of coins and the number of receivers on the platform.
- The second line of input will contain C integers, which are in ascending order. The ith integer is the distance (in metres) of the ith coin from the left edge of the platform.
- The third line of input will contain R integers, which are in ascending order. The ith integer is the distance (in metres) of the ith receiver from the left edge of the platform.

Output

Output should consist of one line containing one integer: the minimum distance Maria must run in order to collect all the coins.

Sample Input

```
4 2
5 10 18 41
14 32
```

Sample Output

```
26
```

Explanation

Maria can do the following:

1. Teleport to the receiver at 32m
2. Run right to reach the coin at 41m
3. Teleport to the receiver at 14m
4. Run right to the coin at 18m
5. Run left to the coins at 10m and then 5m

She will now have collected all the coins and ran a total distance of $9 + 4 + 13 = 26$ metres.

Subtasks

For all subtasks, $C \geq 1$, $R \geq 1$ and all receiver and coin positions are integers between 0 and L (the length of the platform) inclusive.

- For Subtask 1 (15 pts), $C \leq 100$, $R \leq 2$, $L \leq 1\,000\,000$.
- For Subtask 2 (25 pts), $C \leq 100$, $R \leq 100$, $L \leq 1\,000\,000$.
- For Subtask 3 (25 pts), $C \leq 1000$, $R \leq 1000$, $L \leq 1\,000\,000\,000$.
- For Subtask 4 (35 pts), $C \leq 100\,000$, $R \leq 100\,000$, $L \leq 1\,000\,000\,000$.

Problem 3
Torusia

Time and Memory Limits: 1 second, 128 MB

Scientists Alison and Bill were on an important scientific expedition for science to Torusia, a newly discovered donut-shaped planet (yes, the cosmologists are still scratching their heads). During some standard experiments, the two scientists were separated and freak solar winds destroyed their communications equipment. They now have no idea how to find each other, however they each have a machine designed for scientific purposes which they can use to re-unite.

Alison and Bill both view Torusia as a 4096 x 4096 grid, with (0, 0) in the **top-left** corner. The grid "wraps around" at the edges so that horizontal lines are in fact horizontal circles and vertical lines are vertical circles. Hence, the point east of (4095, 34) is (0, 34) and the point north of (17, 0) is (17, 4095).

Alison perceives her own position as (0, 0) and Bill perceives his own position as (0, 0) however their absolute positions are different and hence their co-ordinate systems are not aligned. Note however that they have the same orientation (direction of north, south, east, west) so if Alison is x_A metres east and y_A metres south of Bill, then a point Alison refers to as (x, y) would be the point on Bill's grid labelled $((x + x_A) \mod 4096, (y + y_A) \mod 4096)$.

Alison has a machine that performs one function:

- **mark(x, y)**, creates an electromagnetic marker x metres east and y metres south of her position (the cell (x, y) on her grid). Calling **mark(x, y)** on a cell that already contains a marker doesn't create another marker. Alison must ensure that $0 \leq x, y \leq 4095$.

Bill has a machine that can perform two functions:

- **numRow(y)** finds the number of electromagnetic markers that lie on the row y metres south of him. Bill must ensure that $0 \leq y \leq 4095$.
- **numColumn(x)** finds the number of electromagnetic markers that lie on the column x metres east of him. Bill must ensure that $0 \leq x \leq 4095$.

Each minute, Alison and Bill can use their respective machines to perform one function. Alison's machine is a bit faster to start up than Bill's, so each minute you can assume that her marker is placed *before* Bill performs his query.

You are able to send Alison and Bill a program to help Alison place markers and Bill make measurements so that Bill can determine Alison's position as fast as possible.

Library

Your code file will interact with functions provided in the downloadable source file `science.h`. You must implement the following two functions:

- `void alison();` which can make calls to `mark(x, y)`. Note that each time you call `mark`, the current time is increased by one minute. You can call `mark` at most 10 000 times before exiting your function, otherwise the program will terminate execution.
- `void bill();` which can make calls to `numRow(y)` and `numColumn(x)`. Note that each time either of these functions are called, the current time is increased by one minute. Bill must also finally call `found(x, y)` indicating a belief that Alison is x metres east and y metres south of Bill. Calling `found` will terminate execution.

Your code should not have a "main" function, this will be supplied by `science.h`, in addition to the following:

`void mark(int x, int y);` which can only be called (directly or indirectly) by Alison.
`int numRow(int y);` which can only be called (directly or indirectly) by Bill.
`int numColumn(int x);` which can only be called (directly or indirectly) by Bill.
`void found(int x, int y);` which can only be called (directly or indirectly) by Bill.

You may assume that all these functions run in constant time.

Compilation

You must add `#include "science.h"` to the beginning of your code, and make sure the file `science.h` is in the same folder as your code.

Experimentation

Whilst Alison and Bill are in theory simultaneously performing their actions, we are able to simulate this by first running the program "as Alison", which creates a log file of the cell she marks each minute, then running the program "as Bill". The executable created will take one line of standard input that determines which scientist it runs.

- You can enter one line containing the one character '`A`', which will run Alison's code and output to a file `mark_log`. Pro-tip: this file can be useful for debugging!
- You can enter the character '`B`' followed by two space-separated integers x_a and y_a, representing Alison's position relative to Bill. This will run Bill's code and output the result. You must run Alison before running Bill, as Bill will require the file `mark_log` to be present.

You may wish to temporarily change the value of the constant `SIZE` at the top of `science.h`, to test your program on smaller grids.

Sample Session

The following sample session is based on this code:

```
#include "science.h"

void alison() {
    mark(1, 100);
    mark(2000, 100);
}

void bill() {
    int a = numRow(120);
    int b = numColumn(904);
    int c = numRow(120);
    found(3000, 20);
}
```

First, the program is run and the following input is provided to standard input (the screen):

```
A
```

The program will run `alison()`, which makes these function calls:

Function Call	Explanation
`mark(1, 100)`	In the first minute, Alison marks the cell 1m east and 100m south of her.
`mark(2000, 100)`	In the second minute, Alison marks the cell 2000m east and 100m south of her.

The program exits successfully and creates a file `mark_log`. We now run the program again and provide the following input, which places Alison 3000m east and 20m south of Bill.

```
B 3000 20
```

The program will run `bill()`, which makes these function calls:

Function Call	Explanation
`numRow(120)`	returns 1, as in the first minute Alison marked a cell 100m south of her, and she is 20m south of Bill, hence there is one marked cell on the row 120m south of Bill.
`numColumn(904)`	returns 1, as in the second minute Alison, 3000m east of Bill, marked a cell 2000m east, which is the cell only 904m east of Bill since the grid is 4096m wide.
`numRow(120)`	returns 2, as there are now two markers on the row 120m south of Bill
`found(3000, 20)`	Through some amazingly lucky guesswork, Bill correctly finds Alison's position after only 3 minutes.

Scoring

Your program will be given a score based on the time it takes Bill to find Alison. Specifically, if `found` is called with the correct parameters after M minutes:

$$\text{score} = \begin{cases} 100, & \text{if } M \leq 144 \\ \frac{13000}{M} + 10, & \text{if } 144 < M \leq 10\,000 \\ 0, & \text{if } 10\,000 < M \end{cases}$$

Here is a table showing ten example scores and corresponding success time required:

Score	11	20	30	40	50	60	70	80	90	100
M	10 000	1 300	650	433	325	260	216	185	162	144

There are no subtasks for this problem. Your total score will be the minimum score your program received over all test cases.