Problem B: The Jumping Knight of Square County

Filename: *bsmall, blarge*
Time Limit: 8 seconds

Square County can be modeled of as an n by n grid of squares. Each individual square has a level of elevation, so visually, traveling in Square County involves jumping between rectangular blocks with square bases. (Each block has dimensions 1 x 1 x h, where h is its elevation.)

Instead of police officers, real live Knights typically save the day when any problems occur in Square County. As everyone knows, Knights positioned on grids can jump in an L-shape only. Thus, a knight at the square at (r, c) can jump to (r-2, c-1), (r-2, c+1), (r-1, c-2), (r-1, c+2), (r+1, c-2), (r+1, c+2), (r+2, c-1) or (r+2, c+1). The only other restrictions are that the Knights are NOT allowed to leave the county (go outside the bounds of the grid) and that each Knight has a maximum limit for how far they can jump up or down in elevation on a single jump. This maximum limit is the same for all Knights.

Of course, when there is a problem, we want a Knight to travel to where the problem has occurred as soon as possible.

The Problem

Write a program to help figure out the shortest number of jumps any Knight has to take to arrive at a problem location on the grid, and which Knight can arrive in that shortest number of jumps. Knights can occupy any square on the grid except other squares that have knights.

The Input

The first line of input will consist of a single positive integer, c ($c \le 15$), representing the number of input cases to process. The first line of each input case will contain three positive integers, n ($5 \le n \le 1000$), representing that the grid for the input case has dimensions n by n, k ($1 \le k \le 100$), representing the number of knights for the input case, and m ($1 \le m \le 10{,}000$), representing the maximum height all knights can jump up or down. The following n lines will contain n space separated integers each, representing the elevation of each square on the corresponding row on the grid. Each of these integers is guaranteed to be in between 0 and 10,000, inclusive. The following k lines of input describe the initial position of each of the k knights. The i^{th} ($1 \le i \le k$) of these lines will contain two space separated integers r_i and c_i ($1 \le r_i$, $c_i \le n$), representing the initial location, row followed by column, of the i^{th} knight. The last line of input will contain two space separated integers r_p and c_p, representing the row and column respectively, of where the problem has occurred. (Note: At most two cases will have $n = 1000$, for the rest $n \le 100$.)

Partial Credit Input (40%)

Here are the additional restrictions for the partial credit cases:
$c \le 10$
$k = 1$
$5 \le n \le 15$

All other bounds will be the same as previously described.

The Output

If no knight can travel to the location of the problem, output -1 on a line by itself for the input case. Otherwise, output two integers, *j*, the fewest number of jumps necessary for any knight to travel to the location of the problem, and *id*, the minimum numbered knight who can travel to the location of the problem in *j* jumps. Thus, if more than one knight can make it to the location of the problem in the fewest number of jumps, you must report the knight with the smallest identification number (described earliest in the input).

Sample Input

```
3
5 1 5
3 100 22 100 12
27 100 7 12 100
100 8 100 17 19
100 22 15 100 100
100 100 100 24 100
1 1
5 4
5 3 10
3 100 22 100 12
27 100 7 12 100
100 8 100 17 19
100 22 15 100 100
100 100 100 24 100
1 1
1 5
1 3
5 4
5 1 98
1 100 100 100 100
100 100 100 100 100
100 100 100 100 100
100 100 100 100 100
100 100 100 100 100
1 1
5 5
```

Sample Output

```
5 1
3 2
-1
```

Note: The desired path in the first case is (1, 1) → (3, 2) → (2, 4) → (4, 3) → (3, 5) → (5, 4). The values in these cells are 3, 8, 12, 15, 19 and 24, respectively. In the second case, there are paths from two different knights of length 3: (1, 3) → (3, 4) → (4, 2) → (5, 3) and (1, 5) → (3, 4) → (4, 2) → (5, 3). Of these, we prefer the knight at (1, 5) because its ID number is 2, and that is lower than the other knight, which has ID number 3. Also note that the second case in the sample input isn't valid for the small input.