***Grid Game***

C371\_Coding\_March2023

**Topic**: Searching Algorithms

**Difficulty Level:** Hard

**Question / Problem Statement**:

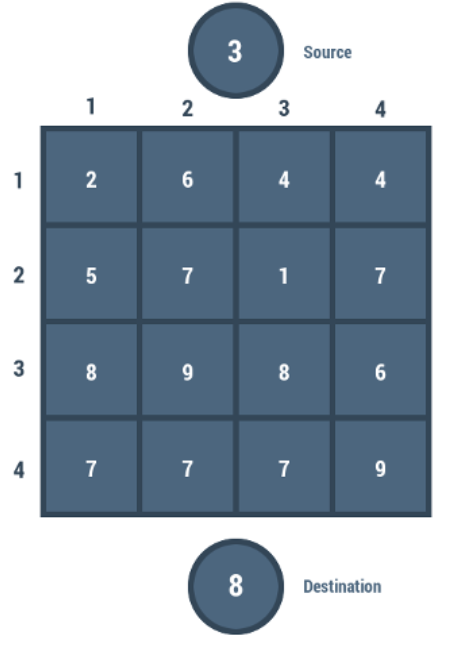
Eleanor is playing a Tiles game. It consists of a Source node and Destination node and a tiles

grid of size **N \* N**.

Grid lies in between Source and Destination (See the image below).

All of them have some value associated with them denoted by **S** (Source value), **D**

(Destination value) and **grid[i][j]** (tile value) where 1 <= **i** <= **N**, and 1 <= **j** <= **N**.



Eleanor is at the Source node and she has to reach the Destination node to finish the game. Rules of the game are:

* She can move from Source to any tile of row1, if the value of that tile is 1 greater than Source. In other words, she can move from source to **grid[1][j]** , if **grid[1][j]** = 1+**S** where 1 ≤ j ≤ **N**.
* She can move from row*i* to any tile of row*i*+1, if the value of the tile at row*i*+1 is 1 greater than the current tile's value. In other words, she can move from **grid[i][j1]** to **grid[i+1][j2],** if **grid[i+1][j2**] = 1+**grid[i][j1]** where 1≤ j1, j2 ≤ **N**.
* She can move from the last row to the Destination node, if the value of the Destination node is 1 greater than the current tile's value. In other words, she can move from **grid[N][j]** to the Destination node, if **D** = 1+**grid[N][j]** where 1 ≤ j ≤  **N**.

Write a program to find the number of ways Eleanor can move from Source to Destination node. Since the answer can be very large, output it modulo 1000000007.

**Note**

There are several queries with separate values of **S** and **D** on the same grid.

**Function Description**

In the provided code snippet, implement the provided **machineAndWorkers(...)** method using the variables to find the number of ways Eleanor can move from Source to Destination node. You can write your code in the space below the phrase **“WRITE YOUR LOGIC HERE”**.

There will be multiple test cases running so the Input and Output should match exactly as provided.  
The base Output variable **result** is set to a default value of **-404** which can be modified. Additionally, you can add or remove these output variables.

**Input Format**

First line contains the size of grid **N**.

Next N lines each contain **N** space separated integers **grid[i][j]**.

Followed by integer **Q**, number of queries.

Each of the next **Q** lines contains two space separated integers, **S** and **D**.

**Sample Input**

4

2 6 4 4

5 7 1 7

8 9 8 6

7 7 7 9

2

3 8

5 10

**Constraints**

1 ≤ **N** ≤ 1000.

1 ≤ **grid[i][j]** ≤ 10^9.

1 ≤ **Q** ≤ 1000.

1 ≤ **S, D** ≤ 10^9.

Sum of **(Q\*N)** over all the test cases in each file will not exceed 10^6.

**Output Format**

For each query, print in a new line number of ways Eleanor can go from **S** to **D**.

**Sample Output**

6

4

**Explanation**

Query 1:

Consider the test case as shown in the image:

From source, there are 2 ways to move to row1 (tiles with value 4)  
From row1, there is only 1 way to move to row2 (tile with value 5)  
From row2, again there is only one way to move to row3 (tile with value 6)  
From row3, there are 3 ways to move to row4 (tiles with value 7)  
And then there's 1 way to move to your destination.

So total 2 \* 1 \* 1 \* 3 \* 1 = 6 ways. They are:

1st way: S → grid[1][3] → grid[2][1] → grid[3][3] → grid[4][1] → D  
2nd way: S → grid[1][3] → grid[2][1] → grid[3][3] → grid[4][2] → D  
3rd way: S → grid[1][3] → grid[2][1] → grid[3][3] → grid[4][3] → D  
4th way: S → grid[1][4] → grid[2][1] → grid[3][3] → grid[4][1] → D  
5th way: S → grid[1][4] → grid[2][1] → grid[3][3] → grid[4][2] → D  
6th way: S → grid[1][4] → grid[2][1] → grid[3][3] → grid[4][3] → D

Query 2:

1st way: S → grid[1][2] → grid[2][2] → grid[3][1] → grid[4][4] → D  
2nd way: S → grid[1][2] → grid[2][4] → grid[3][1] → grid[4][4] → D  
3rd way: S → grid[1][2] → grid[2][2] → grid[3][3] → grid[4][4] → D  
4th way: S → grid[1][2] → grid[2][4] → grid[3][3] → grid[4][4] → D

So, total 4 ways.

**Solution Steps**

1. The following is an outline of the solution for a single query:

Start from S, go to the first row and count the number of integers with value *S+1* (say C1), we can move to any of these C1 numbers from S. The total number of moves possible is C1 since for each of these integers there is only one way to move from S.

Consider row 2, count the number of integers with value *S+2*, (one more than *S+1*, what we are standing on at row 1). Say there are C2 such numbers and we can move to any of those. The total number of ways to reach till row 2 are C1 x C2.

The same can be extended for other rows below these ones. If the number of integers with value *S+i* in row *i* is Ci, the total number of ways to reach the last row are C1 x ... x CN. For each of the positions you may be standing at row *N*, there is only one way to go to D, if D = S+N+1, otherwise there is no way.

2. Note that it may be possible that there are no ways inside the matrix to reach from row *i* to row *i+1* or from S to row *1*.

With the above template, we can do for each query to compute the answer for that particular query.

3. The problem now is to solve the following;  
*Subproblem P1*: In a given row, find how many integers are there of a value *v*.

Solution 1 (doesn’t work for all the cases): Iterate over all the elements of the row and count the number of elements that match the value you are looking for. This is linear in the size of the row.

Since, this operation will be done for each row Q (the number of queries) number of times, we need a faster way to do this. The above will exceed the time limit given for the problem.

Observation: The input matrix is not changing for each query. This helps us in coming up with ideas to pre-compute or manipulate the rows to compute *P1* more efficiently.

Solution 2: Sort each row and for each query of *P1*, do a binary search to find how many numbers are there of a given value.

Don’t forget to take modulo while multiplying.

**Running Solution in C++** :

#include <bits/stdc++.h>

#define MOD 1000000007

using namespace std;

int main(){

int n;

scanf(" %d",&n);

vector <long long> grid[n+1];

long long tmp;

for(int i=1;i<=n;i++)

{

for(int j=1;j<=n;j++)

{

scanf(" %lld",&tmp);

grid[i].push\_back(tmp);

}

//we sort each row, it will help us to find number of occurences of any number in logn

//we can also use a map and store frequency count, it will have the same complexity

sort(grid[i].begin(),grid[i].end());

}

int q;

scanf(" %d",&q);

while(q--)

{

long long start, dest;

scanf(" %lld %lld",&start,&dest);

if((dest-start) != (n+1))

{

printf("0\n");

}

else

{

long long res = 1;

long long look\_up = start+1;

for(int i=1;i<=n;i++)

{

long long count;

vector<long long>::iterator low, up;

low = lower\_bound(grid[i].begin(), grid[i].end(), look\_up);

up = upper\_bound(grid[i].begin(), grid[i].end(), look\_up);

count = (up-grid[i].begin()) - (low-grid[i].begin());

res \*=count;

res %= MOD;

if(count == 0)

{

break;

}

else

{

look\_up++;

}

}

if(dest == look\_up)

{

res \*= 1;

}

else

{

res \*= 0;

}

printf("%lld\n",res);

}

}

return 0;

}

Input:

2

5 5

6 6

5

4 7

5 6

5 7

4 6

1 10

Output:

4

0

0

0

0

**Test Cases [ Qty: 12 ]**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case No** | **Input** | **Output** | **Score** |
| 1 | 4  2 6 4 4  5 7 1 7  8 9 8 6  7 7 7 9  2  3 8  5 10 | 6  4 | 0 |
| 2 | 2  5 5  6 6  5  4 7  5 6  5 7  4 6  1 10 | 4  0  0  0  0 | 0 |
| 3 | 2  2 3  3 4  5  1 5  2 5  3 5  2 4  1 4 | 0  1  0  0  1 | 1 |
| 4 | 1  4  3  3 4  3 3  4 4 | 0  0  0 | 1 |
| 5 | 1  999999999  3  999999998 1000000000  1000000000 999999998  999999998 999999999 | 1  0  0 | 1 |
| 6 | 4  2 4 3 3  5 4 4 7  6 9 5 5  7 7 7 6  3  2 7  1 6  3 8 | 8  0  3 | 1 |
| 7 | 50  349 350 349 679 586 184 586 977 507 349 586 769 184 355 232 549 838 507 349 349 130 232 977 432 124 586 583 432 355 586 432 355 249 838 273 586 232 586 977 124 838 273 432 349 838 273 977 355 232 130  130 249 679 249 184 432 944 130 432 184 273 977 432 769 769 977 432 769 184 130 355 273 769 479 977 583 249 249 679 583 184 124 698 549 838 679 273 249 944 583 507 355 124 838 507 583 249 349 507 432  679 130 349 944 977 679 977 249 273 698 769 679 583 769 349 977 769 479 838 769 549 586 124 184 273 349 355 769 507 583 977 698 507 349 349 184 349 184 679 679 349 583 232 184 977 944 432 586 698 507  944 249 769 124 349 583 432 507 232 273 130 679 507 586 769 249 769 698 549 232 249 355 977 249 944 549 355 549 124 679 479 130 977 583 679 586 232 977 769 507 698 977 232 769 944 349 355 349 507 838  769 507 355 232 273 355 349 698 124 124 232 944 355 698 549 124 232 944 698 586 130 479 698 583 549 432 349 977 130 273 549 273 355 679 507 698 232 769 479 349 679 586 232 273 977 698 349 232 977 698  586 432 355 432 184 698 583 479 838 944 130 586 586 249 273 977 679 479 977 249 273 232 273 583 944 586 355 249 232 249 184 355 583 124 944 838 769 769 507 769 549 838 507 586 124 273 232 184 232 507  432 249 679 124 944 432 944 124 349 698 507 130 507 249 130 679 507 232 130 355 124 698 838 586 184 184 679 583 273 586 977 249 232 184 507 273 549 507 432 273 124 232 679 698 507 838 586 130 249 583  977 698 124 838 679 130 349 944 679 698 583 273 944 355 432 479 130 944 273 507 130 432 507 184 769 944 698 944 838 583 977 549 944 479 583 130 507 432 698 432 977 583 679 130 232 184 838 130 249 583  349 349 249 944 130 130 184 838 349 679 679 479 769 549 349 232 432 479 432 549 944 479 679 273 232 698 679 249 583 944 355 349 698 249 273 586 355 583 479 944 130 184 355 432 124 273 679 249 769 432  273 479 130 124 944 249 583 130 583 698 130 507 586 355 479 124 232 977 507 184 977 432 698 232 769 549 479 184 944 944 838 549 586 549 124 432 977 232 273 679 977 507 698 273 698 479 355 549 349 479  349 838 349 679 586 184 586 977 507 349 586 769 184 355 232 549 838 507 349 349 130 232 977 432 124 586 583 432 355 586 432 355 249 838 273 586 232 586 977 124 838 273 432 349 838 273 977 355 232 130  130 249 679 249 184 432 944 130 432 184 273 977 432 769 769 977 432 769 184 130 355 273 769 479 977 583 249 249 679 583 184 124 698 549 838 679 273 249 944 583 507 355 124 838 507 583 249 349 507 432  679 130 349 944 977 679 977 249 273 698 769 679 583 769 349 977 769 479 838 769 549 586 124 184 273 349 355 769 507 583 977 698 507 349 349 184 349 184 679 679 349 583 232 184 977 944 432 586 698 507  944 249 769 124 349 583 432 507 232 273 130 679 507 586 769 249 769 698 549 232 249 355 977 249 944 549 355 549 124 679 479 130 977 583 679 586 232 977 769 507 698 977 232 769 944 349 355 349 507 838  769 507 355 232 273 355 349 698 124 124 232 944 355 698 549 124 232 944 698 586 130 479 698 583 549 432 349 977 130 273 549 273 355 679 507 698 232 769 479 349 679 586 232 273 977 698 349 232 977 698  586 432 355 432 184 698 583 479 838 944 130 586 586 249 273 977 679 479 977 249 273 232 273 583 944 586 355 249 232 249 184 355 583 124 944 838 769 769 507 769 549 838 507 586 124 273 232 184 232 507  432 249 679 124 944 432 944 124 349 698 507 130 507 249 130 679 507 232 130 355 124 698 838 586 184 184 679 583 273 586 977 249 232 184 507 273 549 507 432 273 124 232 679 698 507 838 586 130 249 583  977 698 124 838 679 130 349 944 679 698 583 273 944 355 432 479 130 944 273 507 130 432 507 184 769 944 698 944 838 583 977 549 944 479 583 130 507 432 698 432 977 583 679 130 232 184 838 130 249 583  349 349 249 944 130 130 184 838 349 679 679 479 769 549 349 232 432 479 432 549 944 479 679 273 232 698 679 249 583 944 355 349 698 249 273 586 355 583 479 944 130 184 355 432 124 273 679 249 769 432  273 479 130 124 944 249 583 130 583 698 130 507 586 355 479 124 232 977 507 184 977 432 698 232 769 549 479 184 944 944 838 549 586 549 124 432 977 232 273 679 977 507 698 273 698 479 355 549 349 479  349 838 349 679 586 184 586 977 507 349 586 769 184 355 232 549 838 507 349 349 130 232 977 432 124 586 583 432 355 586 432 355 249 838 273 586 232 586 977 124 838 273 432 349 838 273 977 355 232 130  130 249 679 249 184 432 944 130 432 184 273 977 432 769 769 977 432 769 184 130 355 273 769 479 977 583 249 249 679 583 184 124 698 549 838 679 273 249 944 583 507 355 124 838 507 583 249 349 507 432  679 130 349 944 977 679 977 249 273 698 769 679 583 769 349 977 769 479 838 769 549 586 124 184 273 349 355 769 507 583 977 698 507 349 349 184 349 184 679 679 349 583 232 184 977 944 432 586 698 507  944 249 769 124 349 583 432 507 232 273 130 679 507 586 769 249 769 698 549 232 249 355 977 249 944 549 355 549 124 679 479 130 977 583 679 586 232 977 769 507 698 977 232 769 944 349 355 349 507 838  769 507 355 232 273 355 349 698 124 124 232 944 355 698 549 124 232 944 698 586 130 479 698 583 549 432 349 977 130 273 549 273 355 679 507 698 232 769 479 349 679 586 232 273 977 698 349 232 977 698  586 432 355 432 184 698 583 479 838 944 130 586 586 249 273 977 679 479 977 249 273 232 273 583 944 586 355 249 232 249 184 355 583 124 944 838 769 769 507 769 549 838 507 586 124 273 232 184 232 507  432 249 679 124 944 432 944 124 349 698 507 130 507 249 130 679 507 232 130 355 124 698 838 586 184 184 679 583 273 586 977 249 232 184 507 273 549 507 432 273 124 232 679 698 507 838 586 130 249 583  977 698 124 838 679 130 349 944 679 698 583 273 944 355 432 479 130 944 273 507 130 432 507 184 769 944 698 944 838 583 977 549 944 479 583 130 507 432 698 432 977 583 679 130 232 184 838 130 249 583  349 349 249 944 130 130 184 838 349 679 679 479 769 549 349 232 432 479 432 549 944 479 679 273 232 698 679 249 583 944 355 349 698 249 273 586 355 583 479 944 130 184 355 432 124 273 679 249 769 432  273 479 130 124 944 249 583 130 583 698 130 507 586 355 479 124 232 977 507 184 977 432 698 232 769 549 479 184 944 944 838 549 586 549 124 432 977 232 273 679 977 507 698 273 698 479 355 549 349 479  349 838 349 679 586 184 586 977 507 349 586 769 184 355 232 549 838 507 349 349 130 232 977 432 124 586 583 432 355 586 432 355 249 838 273 586 232 586 977 124 838 273 432 349 838 273 977 355 232 130  130 249 679 249 184 432 944 130 432 184 273 977 432 769 769 977 432 769 184 130 355 273 769 479 977 583 249 249 679 583 184 124 698 549 838 679 273 249 944 583 507 355 124 838 507 583 249 349 507 432  679 130 349 944 977 679 977 249 273 698 769 679 583 769 349 977 769 479 838 769 549 586 124 184 273 349 355 769 507 583 977 698 507 349 349 184 349 184 679 679 349 583 232 184 977 944 432 586 698 507  944 249 769 124 349 583 432 507 232 273 130 679 507 586 769 249 769 698 549 232 249 355 977 249 944 549 355 549 124 679 479 130 977 583 679 586 232 977 769 507 698 977 232 769 944 349 355 349 507 838  769 507 355 232 273 355 349 698 124 124 232 944 355 698 549 124 232 944 698 586 130 479 698 583 549 432 349 977 130 273 549 273 355 679 507 698 232 769 479 349 679 586 232 273 977 698 349 232 977 698  586 432 355 432 184 698 583 479 838 944 130 586 586 249 273 977 679 479 977 249 273 232 273 583 944 586 355 249 232 249 184 355 583 124 944 838 769 769 507 769 549 838 507 586 124 273 232 184 232 507  432 249 679 124 944 432 944 124 349 698 507 130 507 249 130 679 507 232 130 355 124 698 838 586 184 184 679 583 273 586 977 249 232 184 507 273 549 507 432 273 124 232 679 698 507 838 586 130 249 583  977 698 124 838 679 130 349 944 679 698 583 273 944 355 432 479 130 944 273 507 130 432 507 184 769 944 698 944 838 583 977 549 944 479 583 130 507 432 698 432 977 583 679 130 232 184 838 130 249 583  349 349 249 944 130 130 184 838 349 679 679 479 769 549 349 232 432 479 432 549 944 479 679 273 232 698 679 249 583 944 355 349 698 249 273 586 355 583 479 944 130 184 355 432 124 273 679 249 769 432  273 479 130 124 944 249 583 130 583 698 130 507 586 355 479 124 232 977 507 184 977 432 698 232 769 549 479 184 944 944 838 549 586 549 124 432 977 232 273 679 977 507 698 273 698 479 355 549 349 479  349 838 349 679 586 184 586 977 507 349 586 769 184 355 232 549 838 507 349 349 130 232 977 432 124 586 583 432 355 586 432 355 249 838 273 586 232 586 977 124 838 273 432 349 838 273 977 355 232 130  130 249 679 249 184 432 944 130 432 184 273 977 432 769 769 977 432 769 184 130 355 273 769 479 977 583 249 249 679 583 184 124 698 549 838 679 273 249 944 583 507 355 124 838 507 583 249 349 507 432  679 130 349 944 977 679 977 249 273 698 769 679 583 769 349 977 769 479 838 769 549 586 124 184 273 349 355 769 507 583 977 698 507 349 349 184 349 184 679 679 349 583 232 184 977 944 432 586 698 507  944 249 769 124 349 583 432 507 232 273 130 679 507 586 769 249 769 698 549 232 249 355 977 249 944 549 355 549 124 679 479 130 977 583 679 586 232 977 769 507 698 977 232 769 944 349 355 349 507 838  769 507 355 232 273 355 349 698 124 124 232 944 355 698 549 124 232 944 698 586 130 479 698 583 549 432 349 977 130 273 549 273 355 679 507 698 232 769 479 349 679 586 232 273 977 698 349 232 977 698  586 432 355 432 184 698 583 479 838 944 130 586 586 249 273 977 679 479 977 249 273 232 273 583 944 586 355 249 232 249 184 355 583 124 944 838 769 769 507 769 549 838 507 586 124 273 232 184 232 507  432 249 679 124 944 432 944 124 349 698 507 130 507 249 130 679 507 232 130 355 124 698 838 586 184 184 679 583 273 586 977 249 232 184 507 273 549 507 432 273 124 232 679 698 507 838 586 130 249 583  977 698 124 838 679 130 349 944 679 698 583 273 944 355 432 479 130 944 273 507 130 432 507 184 769 944 698 944 838 583 977 549 944 479 583 130 507 432 698 432 977 583 679 130 232 184 838 130 249 583  349 349 249 944 130 130 184 838 349 679 679 479 769 549 349 232 432 479 432 549 944 479 679 273 232 698 679 249 583 944 355 349 698 249 273 586 355 583 479 944 130 184 355 432 124 273 679 249 769 432  273 479 130 124 944 249 583 130 583 698 130 507 586 355 479 124 232 977 507 184 977 432 698 232 769 549 479 184 944 944 838 549 586 549 124 432 977 232 273 679 977 507 698 273 698 479 355 549 349 479  1  313 234 | 0 | 1 |
| 8 | 2  2 3  3 3  1  1 4 | 2 | 1 |
| 9 | 4  4 6 6 4  5 7 5 7  8 9 8 6  7 8 7 9  2  3 8  5 10 | 8  8 | 1 |
| 10 | 2  1 1  2 1  3  1 2  1 3  2 3 | 0  0  0 | 1 |
| 11 | 1  99999  2  99998 100000  99999 100001 | 1  0 | 1 |
| 12 | 4  2 4 3 3  5 4 4 7  6 9 5 5  7 7 7 6  3  2 7  1 6  3 8 | 8  0  3 | 1 |

Plagiarism found – No

Clarity of the problem statement - Yes

Clarity of the example in the problem statement - Yes

Clarity of sample test cases - Yes

Clarity of test cases (Dual output) – Yes

Clarity of explanations - Yes

Provided Solution running – Yes

EEOC complaint (using abusive words/Indian Names/) - No

Similar Question in System - No

Difficulty Level – Hard

Question w.r.t Searching algorithms concepts- Yes

Final Comment: **Accepted**