Problem A
Ice Cream

Time limit: 1 second
Memory limit: 1024 megabytes

Problem Description

To celebrate the 3rd anniversary of the Ice Cream Professional Company (ICPC), the company has an anniversary sale activity. The activity's name is "Buy X units and Get Y units free". The contents of the activity is as below. First, you will be asking to draw a lottery ticket of X units and a lottery ticket of Y units, and then you can enjoy the discount of buy X units and get Y units free. It means for every X units of ice cream you bought, you will get Y units free ice cream. The price of every unit of the ice cream is 3 dollars, and you want to buy some units of ice cream. Please write a program to calculate the minimum price of your order.

Input Format

The first line contains an integer T, which represents the number of test cases. Each test case contains a single line with 3 integers X, Y, N, which represents the lottery tickets of X, Y, and you want to obtain at least N ice cream. Any two consecutive integers are separated by exactly one space.

Output Format

For each test case outputs an integer, which represents the minimum price of your order.

Technical Specification

- $1 \leq T \leq 15$
- $1 \leq X, Y \leq 1,000$
- $1 \leq N \leq 15,000$

Sample Input 1

```
1
1 1 3
```

Sample Output 1

```
6
```

Sample Input 2

```
2
4 7 22
4 8 22
```

Sample Output 2

```
24
24
```

Almost blank page

Problem B
Maximum Sub-Reverse Matching

Time limit: 3 seconds
Memory limit: 1024 megabytes

Problem Description

Consider two strings of the same length, for example, s_1 = `"happycoding"` and s_2 = `"onedocument"`. Comparison s_1 and s_2 character by character, they get 2 matches, such as

```
happycoding
onedocument
----------
00000100010
```

Suppose that s_2' is obtained by reversing the substring from the 4^{th} character to the 8^{th} character of s_2, i.e.,
s_2' = `"onemucodent"`. Then, s_1 and s_2' will get 4 matches, such as

```
happycoding
onemucodent
----------
00000111010
```

Given a pair of two strings of the same length, say s_1 and s_2, your task is to reverse a substring of s_2 so that the resulting number of characters matched by position can be maximized.

Input Format

The first line contains an integer T, which represents the number of test cases. Each test case below contains three lines. For each test case, the first line is an integer $n \leq 1,000$, which represents the length of the next two lines of strings, and the next two lines are two lowercase strings of length n, called s_1 and s_2 respectively.

Output Format

Each test case outputs four integers $m_{1,2}, m_{1,2*}, a, b$, separated by a space. The first integer $m_{1,2}$ is the number of characters that s_1 and s_2 match by position. The second integer $m_{1,2*}$ is the maximum possible number of matches by matching s_1 and the substring reversed version of s_2. $a \leq b$ are the indexes of s_2, starting from 1. By reversing the substring indexed from a to b in s_2, the maximum number of matches will be obtained. If there are multiple solutions, output the one with the smallest value of $b - a$. If multiple solutions are still obtained, the one with the minimum value of a is output.

Technical Specification

- $1 \leq T \leq 50$.
- $1 \leq n \leq 1,000$
- All string characters are lowercase letters.

Sample Input 1

```
5
11
happycoding
onedocument
8
abbbcccd
xcbcbybd
3
cat
dog
4
abcd
xyza
3
dog
pop
```

Sample Output 1

```
2 4 4 8
2 5 2 7
0 0 1 1
0 1 1 4
1 1 1 1
```

Hint

- dynamic programming.
- Let $dp(i, j)$ represent the number of matches obtained by matching s_1 with a renewed version of s_2, which is obtained by reversing its substring from the i^{th} character to the j^{th}. Denote a_{1i} and a_{2i} as the i^{th} character of s_1 and s_2, respectively. Then, we have

$$dp(i,j) = \begin{cases} mat(s_1, s_2) & i \geq j \\ dp(i+1, j-1) + mat(a_{1i}a_{1j}, a_{2j}a_{2i}) - mat(a_{1i}a_{1j}, a_{2i}a_{2j}) & 1 \leq i < j \leq n \end{cases}$$

where $mat(s, t)$ represents the number of characters matched by position in the strings s and t. The solution appears somewhere in the table.

Problem C
Community Service

Time limit: 3 seconds
Memory limit: 1024 megabytes

Problem Description

ICPC (International-Competition Partner Community) is an emerging community. The people who live there are very young, competitive and willing to serve the community. The residents of ICPC live on a straight-line street. One can use integers to locate or address specific points on the line. The point addresses in the line are numbered from 0 to $n-1$, where $1 \leq n \leq 1,000,000$.

Many young people have moved to ICPC one after another. Everyone who newly moves into the community must provide a one-time community service at certain time. According to the place of residence, each newcomer can choose a set of interesting points covered by an integer range, say $[a, b] (0 \leq a \leq b < n)$ to provide such a one-time service. When the community center requests a service on a set of points covered by an integer range, say $[c, d] (0 \leq c \leq d < n)$, the newest newcomer whose interesting-point set overlays with that set, i.e., has a non-empty intersection, should provide the service. Once the newcomer has provided the service, there is no need to provide service any more.

In this problem, you are given the number of points, say n, on ICPC street, and a sequence of events that occur consecutively over time. There are two types of events, called newcomer events and service-request events, defined below:

Newcomer event

1 *name a b*

where *name* is a string (without space) representing the newcomer's name, a and b are integers such that $0 \leq a \leq b < n$. Any points between a and b (included) are interesting points of this newcomer to provide the one-time service.

Service-request events

2 *c d*

where c and d are integers such that $0 \leq c \leq d < n$. This event represents that ICPC community center issues a service request in the area which includes all points between c and d (included).

Write a program to process the events described above. When receiving a service-request event, output a single line which reports the name of newest newcomer who provides the service. If it is unavailable (because residents who meet the preference condition all have provided services before, or no one interest to provide services in the designated area), output a line of string `">_<"` (without quotation mark).

Figure 1 gives an schematic explanation for the sample input/output. On different time ticks, display the corresponding events that occurred. The shallow-gray bar with name attached

shows the newcomer's interesting area, and the dark-gray bar shows the service request area.

Input Format

The first line includes two integer $n(1 \leq n \leq 1,000,000)$ and $1 \leq e \leq 200,000$, where n denotes the number of points that the community line has, and $e$denotes the number of events to be processed in sequel. The next e lines describes events that occur in sequence over time. The format of the event description is specified in the problem statement.

Output Format

For each service event, output a line describing the name of the resident who provided the corresponding service. If not available, output a line of string `">_<"` (without quotation mark). Please note that different residents may have the same name, but this in fact has nothing to do with problem solving.

Technical Specification

- $1 \leq n \leq 1,000,000$
- $1 \leq e \leq 200,000$
- $0 \leq a \leq b < n$
- $0 \leq c \leq d < n$
- All newcomers' names are composed of alphabets without any spaces, and each length does not exceed 16.

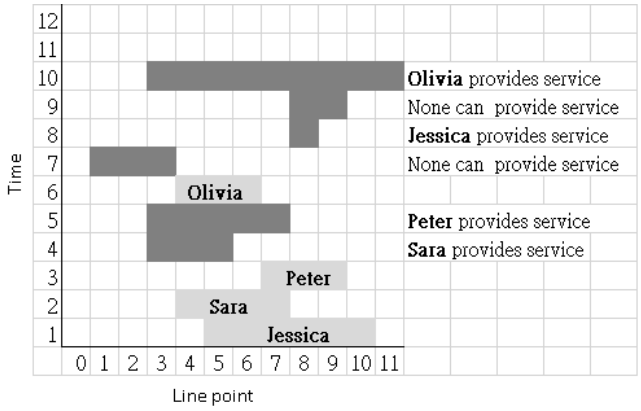


Figure 1: Schematic explanation of sample input and output

Sample Input 1

```
12 10
1 Jessica 5 10
1 Sara 4 7
1 Peter 7 9
2 3 5
2 3 7
1 Olivia 4 6
2 1 3
2 8 8
2 8 9
2 3 11
```

Sample Output 1

```
Sara
Peter
>_<
Jessica
>_<
Olivia
```

Hint

The segment tree is a useful tool for performing range queries, such as range sum and range min/max. This problem is in fact a certain type of range-max query problem. Traditionally, using a segment tree to solve the range-max query problem, newly arrived range data will only overwrite the old. In such a circumstance, a simple lazy propagation mechanism is sufficient to handle dynamic updates of range data. During the range data update, the data is pushed down layer by layer to reach the canonical covering nodes of that range. After reaching a canonical covering node, the node will store the data and then will marks itself as lazy to save downward propagation time. Although lazy, the parent can still push up to maintain the true maximum value of the underlying range. In order to ensure that each range query receives the correct answer, a node that receives a query interval being not canonically covered by it will initiate a pushdown process to propagate lazy data downwards so that the downward queries can be summarized correctly. This ensures that the true maximum value is reported from downward queries.

We will use the segment tree to solve this problem, but the life time of the range data is quite different from the above scenario. As newcomers (with higher event indexes) arrive, their range data, i.e., the event index, will temporarily cover some of the older ones. When all the data above it is removed, the old data will reappear. A naive approach to achieve this effect is to place a stack on the node representing each point, i.e., leaves of the segment tree. This method is only applicable when each independent data is limited in a small range and the global data is not widely distributed.

Instead of placing traditional stacks at leaf nodes, we place a so the called semi-stack on each tree node for storing the event indexes for the canonical covering intervals of Type-1 events. The semi-stack behaves just like a normal stack. Its main feature is that any element in the stack can be deleted on demand[1]. In addition, each tree node also contains two other pieces of

[1]The semi-stack can be implemented using the doubly link list

data, i.e., `lazydown` and `sumup`, as shown in the figure below.

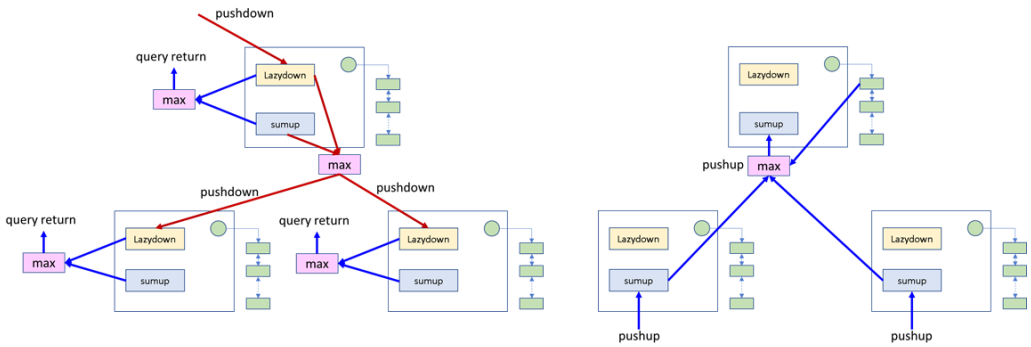


Figure 2: Schematic explanation of main items of segment tree to solve the problem. The left figure shows the behaviors of the `pushdown()` and `query()` methods. The right figure shows the behavior of the `pushup()` method

The `lazydown` variable is used for query. When a tree node receives a query interval of Type-2 event which is not canonically covered by it, the node must compare the event index in its `lazydown` variable with the event index stored on top of its semi-stack, and then push the maximum value down to the `lazydown` variables of its two children. This lets children know the most recent Type-1 event index covered by their parents. The data stored in the `sumup` variable is the summary of their two children and its own knowledge, i.e., the most recent Type-1 event index it has received. Its value is simply the maximum value stored in the `sumup` variables of its two children and the value stored on the top of its semi-stack.

The main methods of the segment tree used to solve the problem are depicted below:

- `update()`: Store Type-1 event index to the tops of semi-stacks of tree nodes which can canonically cover the event interval. Another important point in this method is that the top position of the semi-stack must be stored in the associated list of the type1 event so that the stack node can be removed when service done.
- `pushup()`: Do summary task. It stores the most recent Type-1 event index to `sumup` variable of the tree node. The value represents the most recent Type-1 event index covered by it and its underlying children nodes, as depicted in the right part of the above figure. Besides, for housekeeping, this method will also reset the `lazydown` variables of its two children to 0. Like the classical segment tree, the method will be called at the end of`update()`.
- `pushdown()`: Called only by `query()` method, as depicted in the left part of the above figure. It propagates the most recent Type-1 event index covered by it to its children at lower levels.
- `query()`: Like the classic range segment tree, the information can be covered by a tree node will stop propagating downward. Hence, the method should call `pushdown()` method when needed. The return value of this method is depicted in the above figure.

- `redoPushup()`: After a successful query[2], the main procedure must actively remove all the semi-stack nodes associated with the event from all semi-stacks. This will makes `sumup` variables of some tree node becomes invalid. Therefore, after a successful query, re-pushup is needed so as to renew `sumup` variables. This can be done by along the event update path to perform pushup. Therefore, except that no new event index is pushed onto semi-stack, `redoPushup()` does almost the same thing as `update()`.

Suppose that we have N points globally and M events. Then, the tree height will be $\log N$ and, hence, all of the methods listed above have time complexity $O(\log N)$. The time complexity of this algorithm is $O(M \log N)$.

[2]None zero event index is returned. Event index starts from 1. Index 0 represents NULL event.

Almost blank page

Problem D
Largest Remainder

Time limit: 1 second
Memory limit: 1024 megabytes

Problem Description

Given a list of D digits and an integer K, we consider all different ways to permute these digits into a D-digit decimal number. The target of this problem is to find a permutation X, such that when X is divided by K, the remainder (between 0 and $K-1$) is the largest among all other permutations. If there are more than one possible permutation, output the largest permutation.

For instance, suppose that we have $D = 3$ digits, and they are respectively $1, 2, 3$.

1. If $K = 1$, then we see that every permutation will give a remainder 0 when divided by K, and 321 is thus the desired answer, as it is the largest permutation among all.
2. If $K = 10$, then both 123 and 213 will give a remainder 3 when divided by K, and this is the largest possible remainder in this case. Consequently, the desired output is 213 since it is a larger permutation.
3. If $K = 100$, then the largest remainder we can get is 32, when the permutation is 132.

Input Format

The first line of the input contains a positive integer D followed by a positive integer K. Then, the next line contains D digits, each digit d has value $1 \leq d \leq 9$ and is separated from the next one by a space.

Output Format

Output the largest permutation that gives the largest remainder in a single line.

Technical Specification

- $1 \leq D \leq 16$
- $1 \leq K \leq 200$
- $1 \leq d \leq 9$

Sample Input 1

```
3 1
1 2 3
```

Sample Output 1

```
321
```

Sample Input 2

```
3 10
1 2 3
```

Sample Output 2

```
213
```

Sample Input 3

```
3 100
1 2 3
```

Sample Output 3

```
132
```

Hint

- Construct a boolean table $T[0..2^D - 1][0..K - 1]$ such that for each subset S of the D digits, and each possible remainder r modulo K,

  $$T[S][r] = 1$$

  if and only if there exists an arrangement of the digits of S that gives a decimal number, whose remainder is r when divided by K.

- Each entry of the table T can be computed by DP, using $O(D)$ time. Precisely, we count how many digits are there in S, and for each digit of S, we choose it as the first digit, and check if the remaining digits in S can be arranged as a desired number with a suitable remainder, so that the resulting number has remainder r when divided by K.

- Once T is constructed, we can, for each possible remainder r, find out the largest arrangement $X(r)$ whose remainder is r when divided by K.

  The desired value is the maximum of them, that is, $\max_r\{X(r)\}$.

- Time complexity: T can be constructed in $O(2^D \times K \times D)$ time. Each $X(r)$ can be computed in $O(D^2)$ time, so that the overall checking is $O(K \times D^2)$ time. When $D \leq 16$ and $K \leq 200$, a time limit of 1 second should be sufficient.

Problem E
Composition with Large Red Plane, Yellow, Black, Gray, and Blue

Time limit: 1 second
Memory limit: 1024 megabytes

Problem Description

Mona is a mosaic designer. Recently many customers ask her to build rectangle frames with indicated size to embed photos with instructed layouts. The frames contain borders with 1 pixel thick, and 1 pixel gaps between embedded photos. These photos could be resized with preserved width-height ratios. And they must have integer widths and heights after being resized.

And she needs your help to give her a program to check if each customer's request could be fulfilled.

Input Format

The first line will be 2 positive integers W and H indicating the width and height of the requested frame.

The following lines will describe the layout. The description composed by H-blocks, V-blocks and photos, each occupies a rectangle region.

An H-block describes a region separated horizontally into one or more subregions. The first line of the H-block description will be a positive integer S, which means this region will be split into S subregions. Each subregion will be a V-block or a photo. And the followings are descriptions for subregions.

A V-block describes a region separated vertically into one or more subregions. The first line of the V-block description will be a positive integer S, which means this region will be split into S subregions. Each subregion will be an H-blocks or a photo. And the followings are descriptions for subregions.

A photo description describes a region which occupied by a photo. The first line of the description is a 0 to make this description distinguishable from descriptions of H-blocks and V-blocks. The second line has 2 positive integers w and h indicating the width and the height of the given photo.

The whole layout is an H-block or a photo, distinguished by the first number.

Output Format

If the request could be fulfilled. Print a preview of the frame.

The preview should have H lines and W columns in each line. For each character, print a blank

space ' ' for embedded photos, and print an asterisk '*' for borders or photo gaps.

If the request couldn't be fulfilled. Print a line with message 'Impossible'.

Technical Specification

- $1 \leq W, H \leq 1000$, the width and the height of the frame.
- $1 \leq w, h \leq 1000$, the width and the height of each photo.
- $N \leq 2000$, the total number of H-blocks, V-blocks and photos.

Sample Input 1

```
11 7
2
2
0
3 1
0
2 2
0
1 1
```

Sample Output 1

```
***********
*    *      *
*****       *
*    *      *
*    *      *
*    *      *
***********
```

Sample Input 2

```
11 7
2
2
0
3 1
0
2 2
0
1 2
```

Sample Output 2

```
Impossible
```

Sample Input 3

```
34 19
2
2
3
0
7 1
0
11 1
0
9 1
2
3
0
3 3
0
3 5
0
3 5
2
2
0
15 7
2
0
9 3
2
0
4 3
0
4 3
2
2
0
7 3
0
7 3
3
2
0
7 1
```

```
0
9 1
2
2
0
7 1
0
7 1
0
9 3
0
17 1
2
0
2 11
0
2 5
```

Sample Output 3

```
**********************************
*         *              *          *   *
**********************************   *
*    *                   *          *   *
*    *                   *          *   *
*    *                   *          *   *
*****                    ***********   *
*    *                   *     *    *   *
*    *                   *     *    *   *
*    *                   *     *    *   *
*    ****************************   *
*    *         *         *          *   *
*****         ************************
*    *         *         *          *   *
*    ******************             *   *
*    *         *         *          *   *
*    *         ********************   *
*    *         *                    *   *
**********************************
```

Hint

- Build a linear equation.
  - $2N$ unknowns for widths and heights of H-blocks, V-blocks, and photos.
  - For each photo, one equation for the ratio of the width and height.
  - For each H-block with n subregions.
    - One equation for the width and the sum of widths of all the subregions and gaps.
    - n equations for the height equal to the heights of all the subregions.
  - For each V-block with n subregions.
    - One equation for the height and the sum of heights of all the subregions and gaps.
    - n equations for the width equal to the widths of all the subregions.
  - There are totally $2N$ equations.
- Solve the linear equation by any method (including Gram-Schmidt QR decompositon or Householder QR decomposition).
- Check if all the values to unknowns are integers.
- Check if the height of the outer most H-block or photo is consistent with the height of the frame.

Problem F
What a Colorful Wall

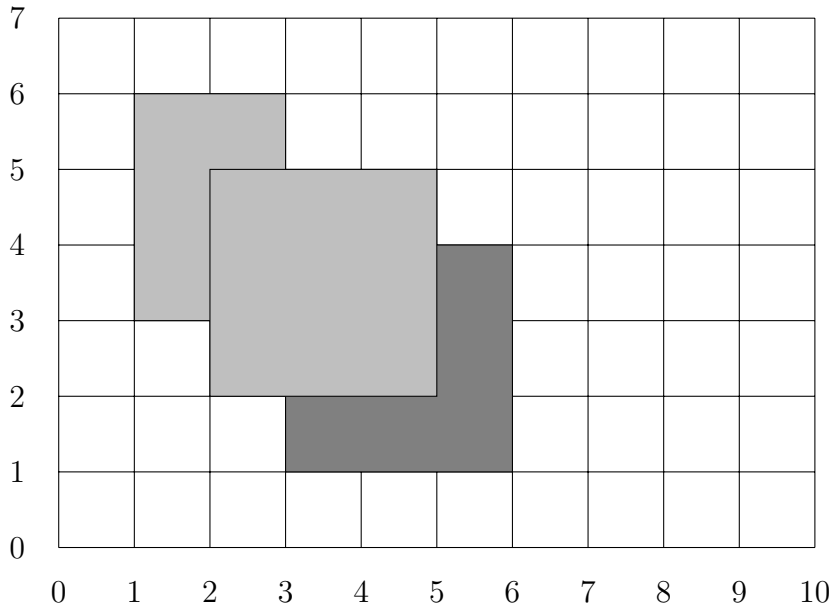
Time limit: 3 seconds
Memory limit: 1024 megabytes

Problem Description

In a lovely park, where you can see trees of green and red roses under skies of blue and clouds of white, there is a wall public for everybody to put up posters. The color of a poster is indicated by a number. If a new poster to put up overlaps with older ones, the overlapping area will be covered by the new one. Given n posters with their coordinates and color, please help to put up them on the wall in the given order and figure out the number of distinct colors that can be seen on the resultant wall. At the beginning, the wall is blank. All the posters are rectangles and cannot be rotated.

For example, the figure shows three posters being put up on the wall in the following order.

1. A light gray poster with a left-top and a right-bottom coordinates of (1, 6) and (3, 3), respectively. The color is numbered 1.
2. A dark gray poster with a left-top and a right-bottom coordinates of (3, 4) and (6, 1), respectively. The color is numbered 2.
3. A light gray poster with a left-top and a right-bottom coordinates of (2, 5) and (5, 2), respectively. The color is numbered 1.



On the resultant wall, two colors, i.e. light gray and dark gray, are visible.

Input Format

Each test case begins with a line showing an integer n denoting the number of posters. Each of the following n lines specifies a poster by five integers, x_1, y_1, x_2, y_2, and c, where (x_1, y_1) denotes the coordinate of the left-top corner of the poster, (x_2, y_2) denotes the coordinate of the right-bottom corner of the poster, and c is the color of the poster. Please help to put up these n posters on the wall in their given order.

Output Format

For each case, print a single integer that is the number of distinct colors visible on the resultant wall.

Technical Specification

- $1 \leq n \leq 4000$
- $0 \leq x_1 < x_2 < 2^{28}$, $0 \leq y_2 < y_1 < 2^{28}$, $1 \leq c \leq n$, for every poster

Sample Input 1

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

Sample Output 1

```
2
```

Sample Input 2

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

Sample Output 2

```
2
```

Sample Input 3

```
3
0 1 2 0 1
0 1 1 0 2
2 1 3 0 2
```

Sample Output 3

```
2
```

Hint

The number of distinct colors visible on the resultant wall can be obtained by figuring out the total area of each color visible on the resultant wall. The brute-force $O(n^3)$ algorithm that fills the $2n \times 2n$ grids is not fast enough for the testdata. The following two-level sweep line algorithm in $O(n^2 \log n)$ is one of acceptable solutions.

1. Assign the z-index of each poster by its given order.
2. Sort x-coordinate of all the left and the right boundaries of posters as $X = \{x_1, x_2, ..., x_{2n}\}$

$(O(n \log n))$.

3. Sort y-coordinate of all the top and the bottom boundaries of posters as $Y = \{y_1, y_2, ..., y_{2n}\}$ $(O(n \log n))$..
4. For every successive interval between x_i and x_{i+1} in X, create a binary search tree T and scan all elements in Y in order with the following consideration:
   - If y_j is the bottom boundary of a poster p_k, and p_k overlaps with the interval x_i and x_{i+1}, then add p_k to T by using its z-index as the key.
   - If y_j is the top boundary of a poster p_k, then remove p_k from T.
   - For every successive interval between y_j and y_{j+1}, the color of the top poster in T will be visible on the resultant wall.

T can be replaced with a binary heap that supports the inner node deletion.

Almost blank page

Problem G
The Treasure of the Sierra Jade

Time limit: 1 second

Memory limit: 1024 megabytes

Problem Description

Congratulations! You are so lucky to find the treasure of the Sierra Jade. Now, your last mission is to transport the treasure down the mountain to the river, where a boat will pick you up. The heavy treasure is shipped on a cart, and you cannot push the cart uphill. Your danger comes from the patrol of the Sierra Jade. The patrol consists of a group of guards, and each of them has a patrol route to traverse the mountain every turn. Your mission fails if you meet any guard at the same location. Given the map of the Sierra Jade and the patrol routes, please figure out the minimum number of turns to transport the treasures to any part of the river.

Input Format

The first line of the input is an integer N denoting the number of test cases. Each case begins with a line of two integers n and m specifying the width and the height of the map of the Sierra Jade. The altitude of each grid in the map is given in the following m lines for a total of $n \times m$ integers. That is, the xth number in the yth line denotes the altitude of the grid $(x-1, y-1)$. A grid with the altitude of 0 is a part of the river. The river may not be connected, and reaching any grid with the altitude of 0 is considered success. Then, your initial location, the grid where the treasure is found, is given in two integers x_s and y_s. Of course, the grid (x_s, y_s) is not a part of the river. The last part of the input is for the guards. An integer g denotes the number of guards. Each ith line in the g lines specifies the patrol route of the ith guard. The first number k_i indicates the number of locations in the route, and the following k_i grids are given in $2k_i$ integers as $x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2}, ..., x_{i,k_i}, y_{i,k_i}$. The guard traverses these grids every turn in order, and then returns to the first grid to start again. As the guards known for their agility, the patrol route may consist of non-adjacent grids. Fortunately, the guards do not patrol the river and will never be located on the grid (x_s, y_s) because of their respect for the treasure.

At the beginning, all guards are at their own first location, and you are at your initial location as well. You and the patrol take turns when moving. In each turn, you move first. You can stay at the same grid or push the cart to one of the four adjacent grids if the grid with an altitude less than or equal to your current grid. The rest of the turn is for the guards to move to the next location in their patrol route. Your mission fails if you are caught by the guard in two situations:

1. You move to a grid where at least one guard is located.
2. You move to or stay at a grid that at least one guard will immediately move to in the same turn.

The following figure illustrates the first case in the sample input as an example. Your location is marked by the letter `Y`, and the guard is marked by the letter `G`. In the first turn, your only choice is to stay at (0, 0) since both adjacent grids are infeasible to move to. That is, the altitude of (1, 0) is greater than that of (0, 0), and (0, 1) is occupied by the only guard. In the second turn, you move to (0, 1). In the third turn, you reach (0, 2), where is a part of the river, with the success of the mission.

Y	2
G	2
0	0

$t = 0$

Y	2
1	G
0	0

$t = 1$

1	2
Y	G
0	0

$t = 2$

1	2
G	2
Y	0

$t = 3$

Output Format

The output contains N lines for the N test cases. For each case, output a single line containing an integer that is the minimum number of turns for you to transport the treasure downhill to any location of the river. Output -1 if it is impossible to ship the treasure to the river.

Technical Specification

- $1 \leq N \leq 10$
- $0 < n, m \leq 40$
- $0 \leq g \leq 10$
- $0 < k_i \leq 8$
- All x-coordinates are integers in the range of 0 and $n - 1$, all y-coordinates are integers in the range of 0 and $m - 1$, and all altitudes are integers in the range of 0 and 2^{20}.

Sample Input 1

```
5
2 3
1 2
1 2
0 0
0 0
1
3 0 1 1 1 1 1
2 3
1 2
1 2
0 0
0 0
1
2 0 1 1 1
2 3
1 2
1 2
0 0
1 0
0
2 3
1 2
1 2
0 0
1 0
2
1 0 1
1 1 1
3 3
1 2 1
1 1 1
0 2 0
1 0
1
3 0 1 1 1 2 1
```

Sample Output 1

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

Hint

This problem can be solved with breath first search. The number of all possible states is bounded by $n \times m \times L$, where L is the least common multiple of $\{k_1, k_2, k_3, ..., k_g\}$. In the case of this problem, the number of states are up to $40 \times 40 \times 840 = 1,344,000$ only.

Problem H
A Big Project

Time limit: 1 second
Memory limit: 1024 megabytes

Problem Description

The ICPC company is running a big project. The project manager, Peter, divides this big project into n subprojects and asks his colleagues to join. Peter finds $2n$ colleagues who are interested in the project, and he wants to have all of them in. For each subproject, Peter decides to have a team of two people. Some of the colleagues have never collaborated with each other before, and Peter would like to encourage collaborations through this project. However, r of the n subprojects are very essential so Peter expects that the two team members of an essential subproject have experience on collaboration.

Given "a list of collaborations", i.e. a list of pairs of the $2n$ colleagues who have collaborated before, please find the number k minimizing $|k-r|$ such that the $2n$ colleagues can be partitioned into n 2-sets with exactly k of which appeared in the list of collaborations. You also have to give the n teams.

Input Format

For each test case, the first line contains three integers n, m, and r, where n is the number of subprojects, m is the number of pairs in the list of collaborations, and r is the number of essential subprojects. The $2n$ colleagues who join the project are numbered from 1 to $2n$. Each of the following m lines contains two integers x and y, indicating that x and y have collaborated before. Consecutive integers in a line are separated by a space.

Output Format

For each test case, the output consists of $n+1$ lines. The first line is the integer k. Each of the following n lines contains two integers x and y. In exactly k of the n lines, either "x y" or "y x" has to be in the list of collaborations. For the remaining $n-k$ lines, neither "x y" nor "y x" appears in the list of collaborations.

Technical Specification

- $0 \le r \le n \le 450$
- $0 \le m \le \binom{2n}{2}$

Sample Input 1

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

Sample Output 1

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

Hint

This is the problem of find *exact perfect matching in a complete graph*.

Given a complete graph G on $2n$ vertices, with edges colored red and blue, and an integer r, the problem asks for a perfect matching consisting of exactly r red edges (denoted by r-pm in following). [3]

Let $r_{\max}$ and $r_{\min}$ be the largest and smallest possible numbers of red edges in a perfect matching of G. If $r \geq r_{\max}$ (respectively, $r \leq r_{\min}$), the problem is to find a maximum (respectively, minimum) cardinality matching in the graph consisting of only red edges.

Assume that $r_{\min} < r < r_{\max}$. Let G_R and G_B be the spanning subgraphs consisting of red and blue edges, respectively. The graph G belongs to one of the following types. (We keep the types disjoint by excluding the former ones.)

(i) Every component of G_R and G_B is complete or complete bipartite.

(ii) Every component of G_R is complete or complete bipartite, and has an even number of vertices.

(iii) Every component of G_B is complete or complete bipartite, and has an even number of vertices.

(iv) G belongs to neither (i), (ii), nor (iii).

The graph G has an r-pm iff one of the following holds:

- G belongs to (i) and $r \equiv r_{\max} \equiv r_{\min} (\mathrm{mod} 2)$,
- G belongs to (ii) and $r \neq n-1$,
- G belongs to (iii) and $r \neq 1$, or
- G belongs to (iv).

If the requested perfect matching exists, an algorithm for computing the matching works as follows.

[3] R. Gurjar, A. Korwar, J. Messner, T. Thierauf: Exact perfect matching in complete graphs. ACM Transactions on Algorithms 9 (2017), Article 8.

First, construct a *pseudo perfect matching* P containing exactly r red edges. A pseudo perfect matching is a set of n edges with at most two of them sharing a common vertex (the bad vertex). In addition, the two edges incident with the bad vertex have different colors. There is one vertex, called the exposed vertex, in the graph, which is not incident with the edges in P.

If there is no bad/exposed vertices, then P is the requested matching. Otherwise, we find a $(1,3)$-cycle or a $(3,1)$-cycle[4]. Then we modify P by "rematching" the vertices in the set V' consisting of

- the bad vertex, the exposed vertex, and the two matched neighbors of the bad vertex;
- vertices of a $(1,3)$-cycle (or a $(3,1)$-cycle), and the matched neighbors (in P) of these vertices.

Assume that the induced subgraph $G[V']$ has r' red edges from P. Computing an r'-pm of G' in a brute-forced way can be done in reasonable time because there are at most 14 vertices in G'[5].

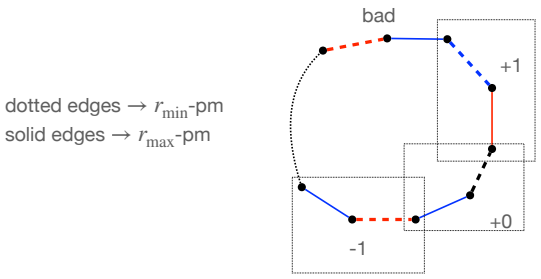
[4]For an (r,b)-cycle, we mean a cycle with exactly r red edges and b blue edges.

[5]Sometimes we need to include an additional red/blue edge from P into $G[V']$ to avoid the case in which a matching with exactly r' red edges does not exist; i.e. type (ii) with $r' = |V'|/2 - 1$ or type (iii) with $r = 1$

Note:

- G belongs to type (i) iff G_R (resp. G_B) has exactly two components which are complete and G_B (resp. G_R) has exactly one component which is complete bipartite. Moreover, any cycle of even length has an even number of red/blue edges.
- G belongs to type (ii) iff $r_{\max} = n$ and G has no $(3, 1)$-cycle.
- G belongs to type (iii) iff $r_{\min} = 0$ and G has no $(1, 3)$-cycle.
- For $r_{\min} < r < r_{\max}$, if an r-pm does not exist, there is always an $(r+1)$- or an $(r-1)$-pm. In these cases we can have the requested matching from the pseudo perfect matching, by arbitrarily matching the bad and exposed vertices with the two neighbors of the bad vertex.

Constructing a pseudo perfect matching. A pseudo perfect matching can be constructed as follows. First, take the symmetric difference of an $r_{\min}$-pm and an $r_{\max}$-pm. The resulting edges induce a set of disjoint cycles of even length, say $C_1, C_2, \ldots, C_k$. (Edges on each cycle "alternates" between the $r_{\min}$-pm and the $r_{\max}$-pm. Then modify the $r_{\min}$-pm by iteratively swapping the cycle-edges until a cycle C_i, by swapping whose edges results in a r^+-pm with $r^+ \geq r$. If $r^+ = r$, the we have the r-pm. Otherwise, find a vertex (the bad vertex) that is incident to both a red and a blue edges. Starting from the bad vertex, we swap the edges, a pair at a time, along the cycle until exactly r edges included. Since swapping a pair can increase/decrease the number red edges by at most 1 and swapping all the pairs results in a r^+-pm, the requested pseudo perfect matching is guaranteed to exist. See the following figure for an illustration.



Problem I
Seesaw

Time limit: 1 second
Memory limit: 1024 megabytes

Problem Description

There is a seesaw in a playground. On each side of the seesaw, there are S seats, where the seats are located at distance exactly $1, 2, \ldots, S$ units from the center. When a seat at distance d units from the center is occupied by someone with weight w, a torque of magnitude $w \times d$ will appear on the corresponding side of the seesaw. The seesaw is balanced when the total torque on both sides have the same magnitude.

Initially, all the seats on the seesaw are occupied with kids with positive integral weights. Furthermore, on the left side of the seesaw, all kids have odd weights, while on the right side of the seesaw, all kids have even weights. Unfortunately, the seesaw is not balanced. Someone suggests to exchange the kids around, so as to see if the seesaw can be balanced. However, kids with weight more than 2 are unwilling to move, so that they will never change their seats. Your task is to determine if it is possible to make the seesaw balanced, and if so, what the minimum number of exchanges is.

For example, suppose $S = 3$, the kids on the left side all have weight 1, and the kids on the right side all have weight 2. Then, it is possible to make 1 exchange, by exchanging the kids at distance 3 on both sides, so that the total torque of the left side becomes $1 \times 1 + 1 \times 2 + 2 \times 3 = 9$, and the total torque of the right side becomes $2 \times 1 + 2 \times 2 + 1 \times 3 = 9$, and the seesaw is balanced.

For another example, suppose $S = 3$, the kids on the left side all have weight 1, and the kids on the right side of the seesaw have weights $4, 2, 2$, at distance $1, 2, 3$ units, respectively. Then, it is possible to make 2 exchanges, first by exchanging the kids at distance 2 on both sides, and then by exchanging the kid at distance 1 on the left with the kid at distance 3 on the right, so that the total torque on the left side becomes $2 \times 1 + 2 \times 2 + 1 \times 3 = 9$, and the total torque of the right side becomes $4 \times 1 + 1 \times 2 + 1 \times 3 = 9$, and the seesaw is balanced. In contrast, we see that it is impossible to balance the seesaw with just 1 exchange, so the minimum number of exchanges in this example is 2.

Input Format

The first line of the input is a positive integer T denoting the number of test cases. Each test case begins with a positive integer S, denoting the number of seats on each side of the seesaw. Then, the next line contains S odd integers, which are the weights of the kids on the left side of the seesaw, listed from the one with distance 1 unit to the one with distance S units from the center, respectively. After that will be a line containing S even integers, which are the weights of the kids on the right side of the seesaw, listed from the one with distance 1 unit to the one

with distance S units from the center, respectively.

Output Format

For each of the test cases, if the seesaw can be balanced, output the minimum number of exchanges needed; otherwise, output -1. Each output is on a single line.

Technical Specification

- $1 \leq T \leq 5$
- $1 \leq S \leq 100$
- $1 \leq w \leq 50000$; w is an integer

Sample Input 1

```
4
3
1 1 3
2 2 2
3
1 1 1
2 2 2
3
1 1 1
4 2 2
3
3 3 3
2 2 2
```

Sample Output 1

```
0
1
2
-1
```

Hint

- We first simplify the problem by deducting 1 from the weight of each kid. Now, the problem is reduced to finding, if it is possible to move kids with weight 1 on the right side, to the empty positions (weight 0) on the left side, so that the total torque on both sides can be balanced.

  And, if it is possible, what is the minimum number of kids that are moved.

- Let Δ be the torque difference exerted by weights more than 1 (after deducting 1) on the two sides of the seesaw.

  Let N denote the number of weight 1 on the right side of the seesaw.

  This problem is equivalent to a modified knapsack problem, where we find if it is possible to select K positions with weight 0, and $N - K$ positions with weight 1, so that

  $$\Delta + \text{ sum of } K \text{ positions with weight } 0 \;=\; \text{sum of } N - K \text{ positions with weight } 1.$$

If multiple possible solutions exist, we report the smallest feasible K.

- The possible sum of any $K = 1, 2, ..., S$ numbers of a set can be computed in a total of $O(S^4)$ time, using DP. Precisely, we use a boolean table B such that

  $B[X][K] = 1$

  if it is possible to generate a sum of value X using exactly K numbers from the set. The table B is constructed by adding one number into consideration at a time.

  Since the size of B is $\binom{S}{2} \times S$, and processing each number accesses each entry at most once, total time is $O(S^4)$. Note that using bitwise operations, this step could be sped up by a constant factor depending on the computer word size.

  When $S = 100$, and the number of test cases is $T = 5$, a time limit of 1 second should be sufficient.

Almost blank page

Problem J
Transportation Network

Time limit: 3 seconds
Memory limit: 1024 megabytes

Problem Description

A retail company owns many convenience stores and a central warehouse in a city. Commodities are transported from the central warehouse to convenience stores. In addition, convenience stores provide a service that customers can send packages from a convenience store to another. A logistics team of this company aim at redesigning their transportation network in the city. The *original transportation network* can be modeled by a simple, undirected, and complete graph $G = (V, E, w)$, in which $V = \{0, 1, 2, \ldots, n-1\}$ is the vertex set, $E = \{\{x, y\} \mid x, y \in V \text{ and } x \neq y\}$ is the edge set, and $w : E \to \{1, 2\}$ is a weight function. Vertex 0 represents the central warehouse and vertices $1, 2, \ldots, n-1$ represent $n-1$ convenience stores. All the edges in E represent the transportation links between all the pairs of vertices in V. The weight of each edge represents the cost to establish the corresponding transportation link.

For ease of management, $V - \{0\}$ is partitioned into two disjoint sets, denoted by S and U, in which S contains the convenience stores located at the main streets, and U contains the convenience stores located at some alleys. That is, except for vertex 0, every vertex in V belongs to either set S or set U. Moreover, the weights of edges in G are defined as follows.

- For vertex 0
  - $w(\{0, s\}) = 1$ if $s \in S$.
  - $w(\{0, u\}) = 2$ if $u \in U$.
- For $s \in S$
  - $w(\{s, s'\}) = 2$ if $s' \in S$ and $s \neq s'$.
  - $w(\{s, u\}) = 1$ if $u \in U$.
- For $u \in U$
  - $w(\{u, u'\}) = 2$ if $u' \in U$ and $u \neq u'$.

A *rooted tree* is a tree with one specified vertex r chosen as the root. For each vertex v, there is a unique path $P(v)$ from v to the root r. The *parent* of v is its neighbor on $P(v)$; its *childern* are its other neighbors. For a non-root vertex v in a rooted tree T, the *depth* of v is the number of edges in the path from v to the root. The *depth of a rooted tree* T is the maximum depth among all the vertices in T. To achive the goal of efficient transportation, the logistics team members devote their effort to connect the central warehouse and all the convenience stores to form a special *depth-2 spanning tree* T such that the following coditions hold:

1. The root of T is vertex 0.
2. The depth of T is at most 2.
3. T contains all the vertices in the original transportation network.
4. The root (i.e., vertex 0) is adjacent to exact p vertices called *hubs*, and each of the remaining vertices is adjacent to exactly one hub.

For any two vertices u and v in a rooted tree T, let $w_T(u, v)$ denote the *weight between u and v in T* (i.e., the sum of the weight of edges in the path between u and v in T). Define $W(T) = \sum_{u \in V} \sum_{v \in V} w_T(u, v)$ to be the *routing cost* of T. The logistics team aims to construct a transportation network T with the minimum of routing cost. For convenience, we call such a depth-2 spanning tree with the minimum routing cost as *critical transportation network*. Figure 3 illustrates a toy example of the problem.

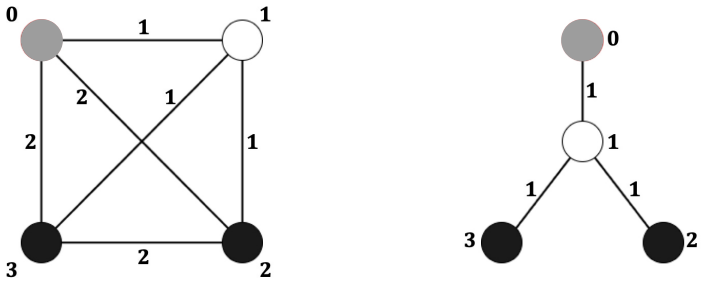


Figure 3: An example of the problem with $n = 4$ and $p = 1$. The original transportation network G is shown on the left hand side, where gray vertex represents vertex 0, white vertex belongs to S, and black vertices belong to U. The critical transportation network is shown on the right hand side.

Given the original transportation network G and a positive integer p, your task is to write a computer program for computing the minimum routing cost of the critical transportation network.

Input Format

The first line of the input file contains an integer L ($L \leq 16$) that indicates the number of test cases as follows. For each test case, the first line contains three integers (separated by whitespaces) representing n, $|S|$, and p. Then it is immediately followed by a single line, which contains $|S|$ integers (separated by whitespaces) that represent the vertices belonging to the set S, where $s \in S$ and $1 \leq s \leq n - 1$. It guarantees the sum of $|S|$ in all testcases are at most $300,000$.

Output Format

The output contains one line for each test case. Each line contains one positive integer representing the routing cost of the critical transportation network.

Technical Specification

- $3 \leq n \leq 1,000,000$ for each test case.
- $|S| + |U| \geq p$.
- $1 \leq |S| \leq p \leq n - 1$.

Sample Input 1

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

Sample Output 1

```
18
68
```

Hint

Let T^* denote the new transportation network. To construct T^*, we can follow the steps below.

Step 1: Let all vertices in S be hubs.

Step 2: Choose arbitrary vertices in U to be the rest of hubs if $p > |S|$.

Step 3: Connect all the hubs to vertex 0.

Step 4: Connect all the remaining vertices to an arbitrary vertex in S.

We can calculate the routing cost of T^* as follows.

$$\begin{aligned} C(T^*) &= 2 \cdot 1 \cdot (n-1) \cdot 1 \cdot (n-p-1) + 2 \cdot 1 \cdot (n-1) \cdot 1 \cdot (|S|-1) \\ &\quad + 2 \cdot 1 \cdot (n-1) \cdot 2 \cdot (p-|S|) + 2 \cdot (n-p) \cdot p \cdot 1 \\ &= 2 \cdot (n-1) \cdot (|U| + p - 1) + 2 \cdot (n-p) \cdot p \end{aligned}$$

Almost blank page

Problem K
Insertion Array

Time limit: 3 seconds
Memory limit: 1024 megabytes

Problem Description

You are given two strings a and b. Let len_a be the length of a and len_b be the length of b. There are $len_b + 1$ ways to insert the string a into the string b. Let $S[i]$ be the string generated by inserting a into the position in front of the i-th chacter(0-based) of b. And let $S[len_b]$ is just generated by place a after b. We call these strings as "inserted strings". For example, when $a =$ "bb", $b =$ "abc", there are four insrted strings. They are $S[0] =$ "bbabc", $S[1] = S[2] =$ "abbbc", $S[3] =$ "abcbb".

The "insertion array" Ins of b with respect to a is a 0-based array of size $len_b + 1$, which contains a permutation of integers from 0 to len_b. Ins describes the lexicographical order of "inserted strings". More formally, the array Ins satisfies following property:

$$i < j \Leftrightarrow [S[Ins[i]] < S[Ins[j]] \vee (S[Ins[i]] = S[Ins[j]] \wedge Ins[i] > Ins[j])$$

In above formula, strings are compared by lexicographical order.

Now your task is calculating the "insertion array" Ins of b with respect to a. In order to avoid large output file, just output the value of $\sum_{i=0}^{len_b}(Ins[i] \times 1234567^i)$ modulo $(10^9 + 7)$.

Input Format

The first line of the input is an integer T denoting the number of test cases. Each test case is in two lines, the first line contains a non-empty string a and the second line contains the non-empty string b.

Output Format

For each test case, output an integer between 0 and $1,000,000,006$ inclusive, representing the value of $\sum_{i=0}^{len_b}(Ins[i] \times 1234567^i)$ modulo $(10^9 + 7)$.

Technical Specification

- $1 \leq T \leq 4 \times 10^4$
- the sum of len_a won't exceed 2×10^6
- the sum of len_b won't exceed 2×10^6
- a and b are consisting of only lower case Latin letters

Sample Input 1

```
2
bb
abc
abaa
abab
```

Sample Output 1

```
468235032
667196656
```

Hint

We can put all indices in an array and sort them using built-in sorting library with custom compared function. Before calling sorting function, we can precompute some information about string a and b such as Z-Algorithm. After that, we can easily (athor think it's easy) compare two inserted strings in $O(1)$ time complexity. Overall time complexity is $O(log(len_b) \cdot len_b + len_a)$.

Problem L
Leadfoot

Time limit: 1 second
Memory limit: 1024 megabytes

Problem Description

Car parking can be one of the toughest challenges when acquiring a driver's license for many people. The International Car Parking Competition (ICPC) is held every year for car parking specialists around the world to join. The rules of ICPC are as follows:

- People who participate in ICPC are called Drivers.
- All Drivers keep track of their own number of Wins and Losses.
- All Drivers start with 0 Wins and 0 Losses initially.
- Two Drivers with the same number of Wins and Losses compete in a match.
- After a match, one Driver has the number of Wins increased by one while the other has the number of Losses increased by one.
- A Driver stops competing others when reaching n Wins or m Losses.
- Drivers finish with 0 Wins and m Losses are rewarded Leadfoot badges.

As the ICPC organizer, it is important for you to make sure all Drivers finish their tournament. That is, you have to invite a specific number of Drivers so that everyone either ends with n Wins or m Losses after multiple matches. To minimize the cost, you want to invite as few Drivers as possible. Given n and m, determine how many Leadfoot badges you have to prepare.

Input Format

The first line contains a single integer k indicating the number of competitions you are holding. Each of the following lines contains two space separated integers n_i and m_i indicating the termination conditions for each competition.

Output Format

For each competition output a single integer, the number of Leadfoot badges you have to prepare modulo 1000000007.

Technical Specification

- $1 \leq k \leq 10^5$
- $1 \leq n_i, m_i \leq 10^5$

Sample Input 1

```
5
1 1
2 2
4 6
6 4
12 3
```

Sample Output 1

```
1
1
4
16
1024
```

Hint

We can observe that the total number of drivers needed is 2^p. Let $\nu_2\left(\binom{n}{m}\right)$ denote the 2-adic valuation of a binomial coefficient. Then, p is the max of the two following equations.

$$\max_{0\le i\le l} w+i-\nu_2\left(\binom{w-1+i}{i}\right) \tag{1}$$

$$\max_{0\le i\le w} l+i-\nu_2\left(\binom{l-1+i}{i}\right) \tag{2}$$

With p, we can then calculate the answer by 2^{p-l}. We can apply Kummer's theorem to the above equations, so that they become

$$\max_{l-20\le i\le l} w+i-\nu_2\left(\binom{w-1+i}{i}\right) \tag{3}$$

$$\max_{w-20\le i\le w} l+i-\nu_2\left(\binom{l-1+i}{i}\right) \tag{4}$$

Problem M
Escaping the Foggy Forest

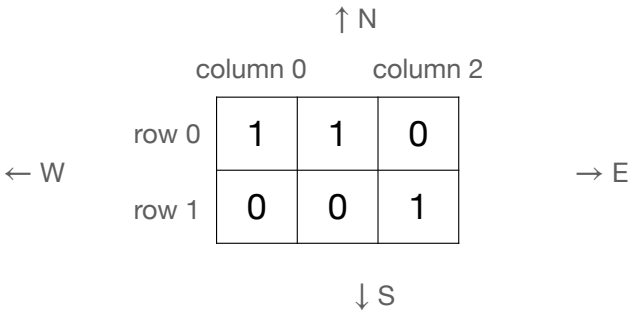
Time limit: 1 second
Memory limit: 1024 megabytes

Problem Description

Nobi is walking in a foggy forest. He tries to locate himself using a map. The forest is divided using vertical and horizontal lines into $m \times n$ regions so the map is represented by an m by n matrix, with rows indexed from 0 to $m-1$ and columns indexed from 0 to $n-1$. Each entry of the matrix corresponds to a region of the forest, where the entry with row index r and column index c is denoted by $M(r, c)$. Each entry is either 1 or 0. A 1 indicates that there are very tall trees in the region, and a 0 indicates that the region contains only bushes. It is known that the outside of the forest is surrounded by bushes, which means that regions not specified in the map are treated as 0s.

Nobi wants to know which region he might be in. Because the fog is heavy, Nobi can only observe **the region he stays in**, **the region in the front**, and **the region on the right side**. However, he **does not know which direction** he faces to. Please help Nobi to find the possible regions he is in, assuming that there are only four directions, "N", "E", "S", and "W".

For example, below is a 2 by 3 matrix.



Nobi observes that there are tall trees in the regions he stays in and in front of him (corresponding to matrix entries of 1), and the region on the right side of him has only bushes (corresponding to a matrix entry of 0). It is possible that Nobi is in the region corresponding to entry $M(0, 0)$ because $M(0, 0) = 1$, and when he faces to the direction "E", the entry in front of him, $M(0, 1)$, is 1; the entry on the right side of him, $M(1, 0)$, is 0. The other possible region is $M(0, 1)$ because $M(0, 1) = 1$ and when he faces to "W", the entries in the front is $M(0, 0) = 1$; the entry on the right side is outside the map, which is treated as 0.

Input Format

The first line of a test case contains two integers m and n, which are the numbers of rows and columns of the matrix, respectively. The rows of the matrix are indexed from 0 to $m-1$, and columns are indexed from 0 to $n-1$. The following m lines are the content of the map. Each of

the m lines contains n integers. The last line contains three integers s, f, and r. The number s, f, and r indicate what Nobi observes in the regions he stays in, in front of him, and on the right side of him, respectively.

Output Format

Each line of the output contains two integers, which are the possible row and column indices of the region Nobi is in, respectively. It is guaranteed that Nobi **is in the forest**. If there are more than one possible regions, output the regions with smaller row index first; for regions with the same row index, output the one with smaller column index first.

Technical Specification

- $1 \leq m, n \leq 100$
- Each entry is either 0 or 1.
- $s, f, r \in \{0, 1\}$

Sample Input 1

```
2 3
1 1 0
0 0 1
1 1 0
```

Sample Output 1

```
0 0
0 1
```

Hint

- For each entry of the matrix, test for each of the four directions whether the front and the right entries match f and r.