Socializing

Tigress Tuli studies in Tiger School, where all the tigers are very talkative and friendly. There are **N** tigers in the school (including Tuli), numbered from 1 to **N**.

Initially none of the **N** tigers knew each other, that is, they were not friends. One day, the **N** tigers arranged themselves in a line, such that, the Kth tiger from the left was the tiger numbered K (for all K such that 1 <= K <= **N**). Of course, the tigers started talking and tigers standing next to each other quickly became friends. That is, tiger number i and (i + 1) became friends, and vice versa. (for all i, 1 <= i <= **N** - 1)

The next day, Tuli was given the task of arranging the **N** tigers in a line. She wanted to arrange them in such a way so that at least **newFriends** pairs of tigers stand next to each other, who are not already friends.

She has already arranged **M** tigers at the beginning of the line, and they are the tigers numbered **T**[1], **T**[2], ... **T**[M] from left to right.

Tuli wants to know the number of ways she can arrange the remaining (**N** - **M**) tigers. Since this number can be very large, output it modulo 1000,000,007.

Constraints
N will be between 1 and 100, inclusive.
newFriends will be between 0 and (**N** - 1), inclusive.
M will be between 0 and (**N -** 1), inclusive.
T[1], T[2], ... T[M] will be distinct and contain numbers between 1 and **N**, inclusive.

Input Format
The first line of input will contain two space-separated integers **N** and **newFriends**.
The second line contains a single integer **M**.
The third line contains **M** space-separated integers T[1], T[2], ..., T[M]. Note that if **M** is 0, then this line would be blank.

Output Format
Output a single integer, the number of ways Tuli can arrange the remaining tigers as described in the statement, modulo 1000,000,007.

Notes
There might be no valid ways to arrange the remaining tigers. In that case, the correct output is

0.
The output must be an integer between 0 and 1000,000,006, inclusive.

Examples
1.
Sample Input
4 3
0

Sample Output
2

Explanation Since, **newFriends** = **N** - 1, all three pairs of adjacent tigers must not be old friends. That is, tigers i and (i + 1) should not be adjacent. There are two ways of arranging the tigers.
2 4 1 3
3 1 4 2

2.
Sample Input
4 3
2
3 1

Sample Output
1

Explanation Only one arrangement, starts with 3 1, viz. (3 1 4 2).

3.
Sample Input
5 3
2
1 3

Sample Output
3

Explanation This time, at most one pair of friends is allowed. The three permutations are:
1 3 4 2 5

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

4.

Sample Input

```
10 5
3
1 2 3
```

Sample Output

```
3649
```

Explanation That’s a lot of permutations.

5.

Sample Input

```
30 20
5
1 3 6 10 30
```

Sample Output

```
570753391
```

Explanation Don’t forget to count module 1000,000,007.

Crossings

Elephant Elly likes numbers arranged in circles. She has arranged the numbers 1, 2, …, **N** in a circle in clockwise order. That is, i and i+1 are adjacent for i between 1 and **N**-1, inclusive, and 1 and **N** are adjacent. Since Elly prefers even numbers, **N** is even.

Elephant Elly likes drawing straight lines. After seeing the numbers arranged in such a nice circle, she couldn't resist drawing straight lines joining pairs of numbers.

Elephant Elly drew exactly **N**/2 straight lines with her trunk, each joining a pair of numbers. Moreover, every number from 1 to **N**, inclusive, was the endpoint of exactly one line. A crossing is an (unordered) pair of lines which intersect. Since Elephant Elly doesn't like messy sets of lines, she made sure there are at most **maxCrossings** crossings.

Elly wants to know the number of ways she can draw lines given the above conditions. Since this number might be very large, output it modulo 1000,000,007.

Constraints
N will be between 2 and 100, inclusive.
N will be even.
maxCrossings will be between 1 and 1225, inclusive.

Input Format
The first and only line of input will contain two space-separated integers **N** and **maxCrossings**.

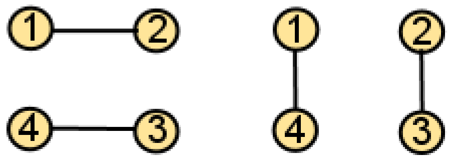
Output Format
Output a single integer, the number of ways Elly can draw a set of lines as described in the problem statement, modulo 1000,000,007.

Notes
Two sets of lines **A** and **B** are different if there is a pair of numbers joined by a line in **A**, but not joined by any line in **B**.

Examples
1)
Sample Input
4 0

Sample Output

2

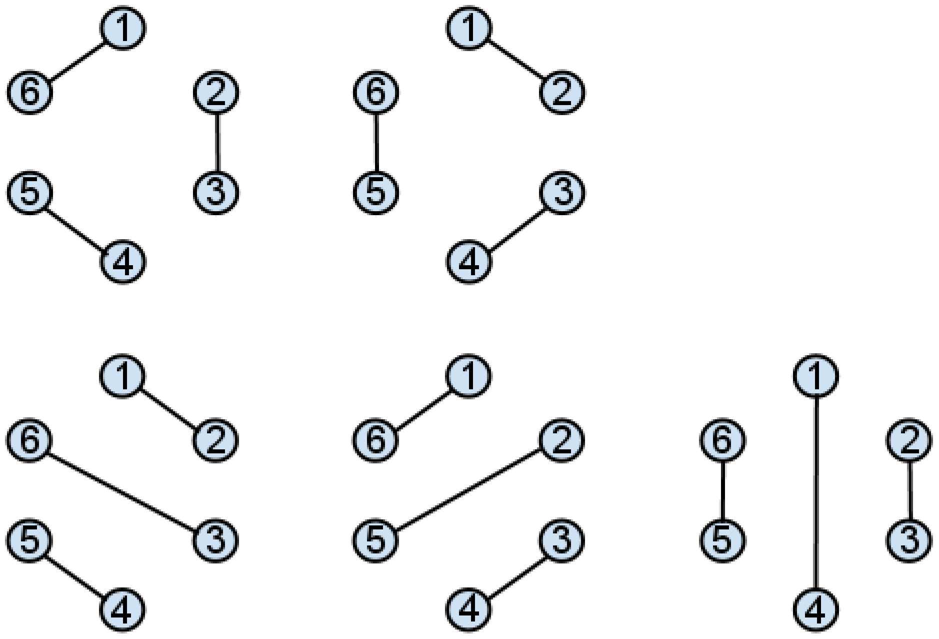


Explanation The two ways are:

2)
Sample Input
6 0

Sample Output
5

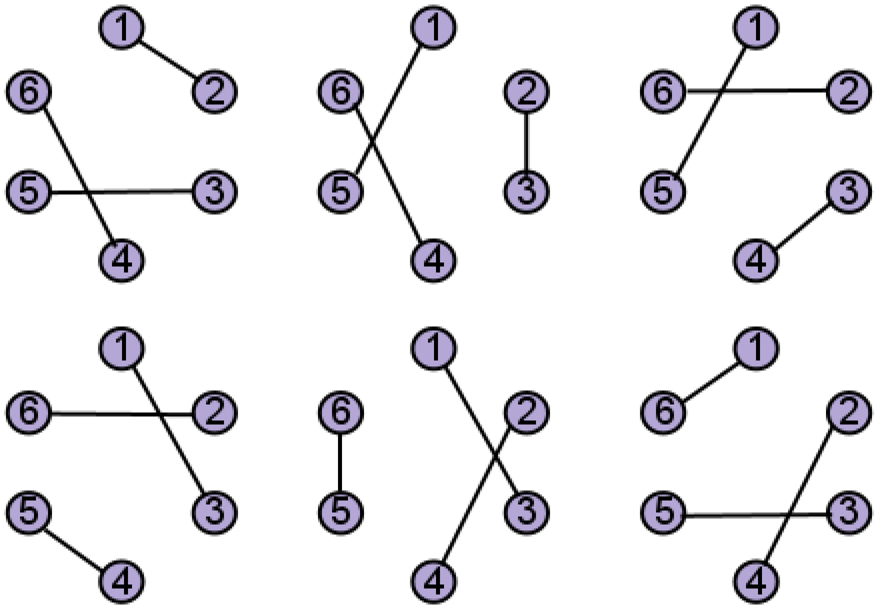
Explanation There are five ways to draw three pairs of lines without intersecting each other.



3)
Sample Input
6 1

Sample Output
11

Explanation Apart from the above five ways with no crossings there are six with exactly one crossing.



4)
Sample Input
16 28

Sample Output
2027025

Explanation The maximum number of crossings there can be is (8 choose 2) = 28. So we need to count all possible sets of lines. There are 16! / (8! * 2^8) = 2027025 such sets.

5)
Sample Input
100 1225

Sample Output
196932377

Explanation 100! / (50! * 2^50) = 196932377 modulo 1000,000,007.

Travelling

Panda Poki likes to travel a lot. This year, he wants to visit a faraway country. There are **N** countries in panda-world and they are numbered from 1 to **N**.

Panda Poki is currently in the country 1, and he wants to go to country **N**. There are **nFlights** bidirectional flights between pairs of countries. The ith such flight goes from **A**[i] to **B**[i] (and from **B**[i] to **A**[i]) and takes **time**[i] units of time.

Unfortunately, Panda Poki is not allowed to have a legitimate passport since he is a panda. He can only travel to a country if he holds a passport which allow him to visit that country. He is going to procure fake passports which will allow him to travel to various countries.

In country i, Poki can buy only the ith kind of fake passport available. This passport would allow Poki to visit a subset of **nCountries**[i] countries, they are given in **passport**[i][j] (1 <= j <= **nCountries**[i])

Poki must visit the country i to buy the ith kind of fake passport, at which point he has to throw away the passport he was previously holding. Note that he might not choose to buy the ith passport, even if he visits country i.

Poki can afford to buy at most **K** passports. He wants to find out the minimum time required to travel from 1 to **N**. If it is not possible to reach country **N**, output -1.

Constraints
N will be between 1 and 500, inclusive.
maxPassports will be between 1 and **N**, inclusive.
nFlights will be between 1 and **N** * (**N** - 1) / 2;
A[i] and **B**[i] will be different, and between 1 and **N**, inclusive. (1 <= i <= M)
time[i] will be between 1 and 10,000, inclusive. (1 <= i <= M)
nCountries[i] will be between 1 and **N**, inclusive.
passport[i][1], passport[i][2], ..., passport[i][nCountries[i]] will contain distinct elements from 1 to **N**, and will contain i.

Input Format
The first line of input will contain three space-separated integers **N** and **nFlights** and **maxPassports**.
Lines 2 to (**nFlights** + 1) describe the flights, the (i + 1)th line contains three integers **A**[i], **B**[i] and **time**[i].

The next **N** lines describe the passport available at the different countries. The ith line among these contain an integer **nCountries**[i], followed by the list of integers **passport**[i][1], **passport**[i][2], ..., passport[i][nCountries[i]].

Output Format
Output a single integer, the minimum time required to travel from country 1 to country **N**.

Notes
Poki must buy a passport at city

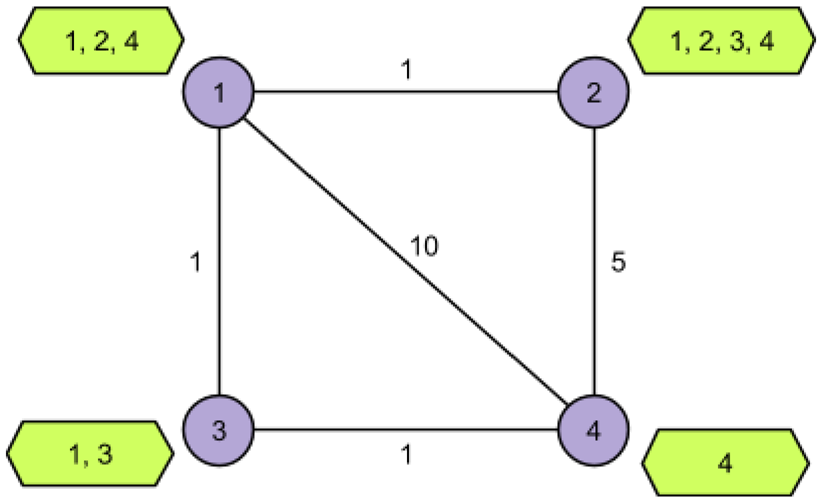
Examples
1.
Sample Input (two extra blank lines for clarity)

```
4 5 2

1 2 1
1 3 1
1 4 10
2 4 5
3 4 1

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

Sample Output

```
4
```

Explanation
Poki must buy a passport at country 1. This allows him to visit countries 1, 2 and 4.
In the given graph, one possible route is to go from 1 to 4, directly. This would take 10 units time. The route 1 -> 2 -> 4 would be shorter, it takes only 6 units time. Poki cannot take the much shorter route 1 -> 3 -> 4 since he isn't allowed to enter 3 using country 1's passport. However, he can take 4 units time with the following route: 1 (buy passport 1) -> 2 (buy passport 2) -> 1 -> 3 -> 4.



2.

Sample Input

```
4 5 1
1 2 1
1 3 1
1 4 10
2 4 5
3 4 1
3 1 2 4
4 1 2 3 4
2 1 3
1 4
```

Sample Output

```
6
```

Explanation

It is the same situation as above, except Poki is allowed to buy 1 passport instead of 2. If he is allowed to buy only one passport, then Poki cannot take the shorter route. Poki takes the route 1 -> 2 -> 4.

Intervals

Rabbit Ravi studies in Rabbit School, where all rabbits have large watches and are very punctual. There are **N** rabbits in the school, numbered from 1 to **N**. Every day the ith rabbit needs to use a study room precisely from time **S**[i] to **E**[i]. The rabbits are numbered such that for 1 <= i <= **N** - 1, **S**[i] <= **S**[i + 1].

Ravi has been assigned the duty of assigning the rabbits to study rooms. There are **N** study rooms numbered from 1 to **N**. Two rabbits cannot use the same study room simultaneously. If a rabbit uses the room till time T, the next rabbit must start using it in some time greater than or equal to T + 1.

Ravi would like to assign rabbits to rooms such that the total number of rooms used is minimized. If there are many possible such assignments, he wants to choose the lexicographically smallest assignment.

More precisely, if the ith rabbit is assigned room R[i]. Then, first Ravi would minimize the total number of distinct rooms in the sequence (R[1], R[2], R[3], ..., R[N]). Next, among all such sequences Ravi would choose the lexicographically smallest sequence.

Constraints
N will be between 1 and 100,000, inclusive.
S[i] and **E**[i] will be between 1 and 1000,000,000, inclusive. (1 <= i <= N)
S[i] <= **E**[i] (1 <= i <= N)
S[i] <= **S**[i + 1] (1 <= i < N)

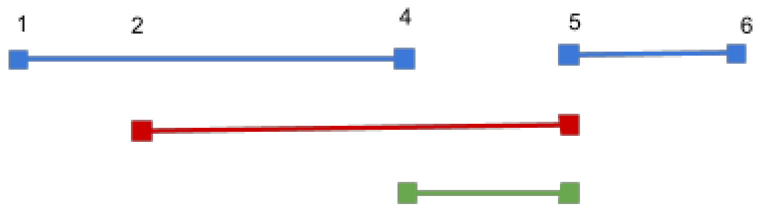
Input Format
The first line of input will contain integer **N**.
The next **N** lines will contain two space-separated integers each. The ith line will contain **S[**i] and **E**[i], in that order.

Output Format
The first line of output should contain a single integer, the total number of rooms used in the assignment.
The next line should describe the lexicographically smallest assignment. It should contain **N** space-separated integers, the ith integer should be the room number the ith rabbit is assigned to.

Notes
Two intervals [a, b] and [c, d] intersect if min(b, d) >= max(a, c). In particular, intervals [1, 2] and [2, 3] intersect.
A sequence P is lexicographically smaller than Q if they differ, and at the first position they differ, (say at position i), P[i] < Q[i].

Examples
1.
Sample Input
4
1 4
2 5
4 5
5 6

Sample Output
3
1 2 3 1

Explanation


2.
Sample Input
10
1 7
2 4
2 6
2 9
2 9
4 6
5 5
6 8
8 9

```
10 20
```

Sample Output

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

Boxes

Beaver Bindu loves building stuff. For her next project, she wants to build a stack of boxes. Before she tackles real-life boxes, she wants to get some practice.

Bindu has cut out **N** boxes (or rectangles) from a sheet of paper. The ith box has width **W**[i] and height **H**[i]. She can rotate each box so that either the width or the height is on the top: let us call this length the base size of the box. A box X can be put on top of another box Y if the base size of X is strictly smaller than Y. Note that either X or Y or both might be rotated, that is, their base sizes might be their width or height.

Bindu will select a subset S out of the **N** boxes, reorder and rotate them in any way she wants. (That is the base size of a box can be either its width, or its height.) And then she will arrange the boxes in S in a valid stack.

Bindu wants to know what is the largest stack she can create. That is, she wants to maximize the number of boxes in a stack. Please find this number.

Constraints
N will be between 1 and 5,000, inclusive.
W[i] and **H**[i] will be between 1 and 5,000, inclusive. (1 <= i <= N)

Input Format
The first line of input will contain integer **N**.
The next **N** lines will contain two space-separated integers each. The (i + 1)th line will contain **W**[i] and **H**[i], in that order.

Output Format
Output a single integer, the maximum number of boxes Bindu can build a stack with.

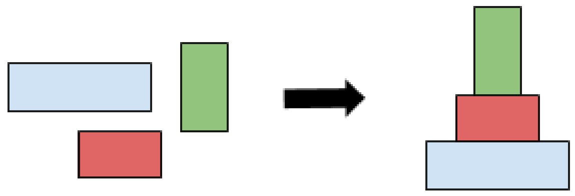
Notes
Bindu cannot use the same box twice in a stack.
Some boxes may not be used in an optimal stack.
Two different boxes may have the same width and height.

Examples
1.
Sample Input
3

1 3
2 1
1 2

Sample Output
3

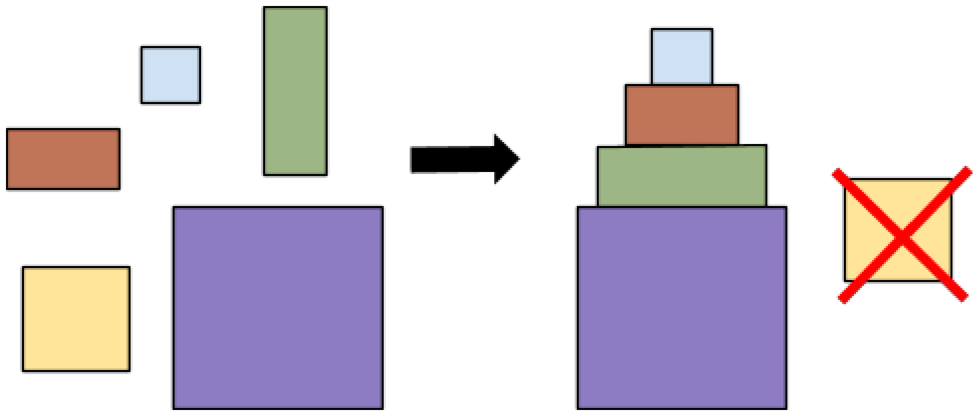
Explanation Bindu has 3 boxes: of sizes 1 x 3, 2 x 1 and 1 x 2. She uses all three boxes for a stack with bases 1, 2 and 3.



2.
Sample Input
5
1 1
1 2
3 1
4 4
2 2

Sample Output
4

Explanation In this case Bindu cannot use all the boxes for a stack. In one of the optimum solutions, she rotates the 3rd box and leaves out the 5th box.



3.

Sample Input

```
10
4 6
7 3
7 4
5 5
4 5
1 6
3 1
8 7
9 2
4 7
```

Sample Output

```
8
```