Security Clearance

The office you work in, Super Top Secret Stuff LLC, is implementing a new security system. The office contains many rooms with doors linking them together. However, every employee is only allowed to enter certain rooms. To provide security, each door has been outfitted with a badge reader that will only allow authorized employees through the door. For simplicity, your firm decided that each door would be coded with a range of valid badge numbers (e.g., 25-29 would allow badge numbers 25,26,27,28, and 29 through). Complicating matters, the lock on each side of the door is different, so it might be the case that employees can enter a room, and then not leave (of course, this is an unintended consequence).

A significant problem for the company is this: Given a new employee, what badge number should they get so that they can actually get from the front door to their work station. Write a program that, given the badge numbers permitting passage between each room, returns a list of badge numbers that will get someone from a given start room to a given destination room.

Input

The first line of input will contain integers N, L, and B, denoting the number of rooms, of locks, and of badge numbers, respectively. $2 \leq N \leq 1000$, $1 \leq L \leq 5000$, $1 \leq B \leq 10^9$

The next line of input will contain two integers, S and D, $1 \leq S \leq N$, $1 \leq D \leq N$, $S \neq D$, denoting the starting and destination rooms that we are interested in.

This is followed by L lines, each describing a lock as four integers: a, b, x, y indicating that a lock permits passage from room a to room b (but not from b to a) for badges numbered from x to y, inclusive. It is guaranteed that $1 \leq a, b \leq N$, $a \neq b$, $1 \leq x \leq B$, $1 \leq y \leq B$, $x \leq y$, and no (a, b) pair will occur more than once, although both (a, b) and (b, a) may occur within separate lines.

Output

Output a single line showing the number of badge numbers that allow passage from room S to room D. You **do not** need to print the actual badge numbers themselves, just how many unique valid badge numbers exist that work.

Sample Input

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

Sample Output

```
5
```

Hints

In order to solve this problem, you are going to have to be clever about a couple things. Here are some hints to get you started / to consider when working on this problem:

- For any given badge number, you can do a graph search (BFS, DFS, etc.) to figure out if that one badge number can make it from the start room to the end room. Depending on how you approach the problem, a method that checks if one badge is feasible might be useful.
- The above is true, but consider that the number of badges B is quite large ($B \leq 10^9$). Therefore, you cannot try every badge (that is way too many) and you cannot even store that many items in memory. So, regardless of your appraoch, you need **SOME** way to reduce that value. Consider this: because $B \leq 10^9$ is so large, but the number of locks $L \leq 5000$ is much smaller, you would expect to have LARGE stretches of badges that all act the same (because there are only 10000 total badge endpoints at most). So you'll need some way to handle large groups of badges all at once. For example, if the smallest lock entry badge is at $85,000$, there is not need to test badges 1 through $84,999$ because they can't possible make it through any locks. If you can test LARGE batches of locks all at once like this, you can get your runtime way down.