Task 5: `Mushroom Ring`

In the wild, mushrooms can often be spotted growing in rings, arcs and other shapes. Though they appear naturally, mushroom rings in particular have long been the subject of myths and folklore. It was said that they grew from the land when scorched by a dragon's tail. Another myth states that witches danced around them at night. More recently, legends told of geniuses who flocked to mushroom rings, stood in the centre, and conjured up six competitive programming problems about collecting mushrooms.

Following his travels on the infinite field, Stuart the Snail now lives in Snail Village which consists of a ring of n giant mushrooms! The mushrooms are numbered from 1 to n. Next to each mushroom, there are $n-1$ signs pointing to all other mushrooms, giving a total of $n(n-1)$ signs.

m consecutive ranges of numbers are written on some signs. On the sign which is placed next to mushroom $u[i]$ and points towards mushroom $v[i]$, **all numbers from** $a[i]$ **to** $b[i]$ **inclusive** are written ($1 \le a[i] \le b[i] \le n, 1 \le i \le m$). Signs satisfy the two *clarity rules* below:

1. A sign cannot contain the number of the mushroom it is next to. Formally, $u[i] < a[i]$ or $b[i] < u[i]$.
2. Two ranges on signs next to the same mushroom cannot both contain the same number. Formally, if $i \neq j$ and $u[i] = u[j]$, then $b[i] < a[j]$ or $b[j] < a[i]$.

Note that there is no constraint on $v[i]$.

Snails are able to follow the instructions on the signs perfectly. Suppose a snail is currently at mushroom c and wants to get to mushroom d in the end. If $c = d$, she is done. Otherwise, she looks at the signs next to mushroom c and finds the sign that contains the number d. She follows that sign to the next mushroom $v[i]$ and repeats the process.

Unfortunately, the signs themselves may not be perfect. If a snail is not able to find the destination mushroom d on any sign, she is stuck and cannot proceed. On the other hand, signs may bring a snail on an infinite cycle without ever passing by mushroom d.

Define the **utility** of the signs as the number of ordered pairs (s, d), such that a snail starting at mushroom s can reach mushroom d by following signs.

Stuart would like to improve the signs to maximise the utility. Due to a lack of resources, at most k edits to signs can be made. There are two types of edits: add a new number to a sign, or remove an existing number. After making edits, the two *clarity rules* must remain satisfied. It is not necessary for the signs to contain consecutive ranges of numbers after the edits.

Help Stuart find the maximum possible utility after making edits optimally.

Input Format

Your program must read from standard input.

The first line of input contains three space-separated integers n, m, and k, describing the number of mushrooms, the number of ranges, and the maximum number of allowed edits.

The next m lines of input each contain four space-separated integers. The i-th of these lines contains $u[i], v[i], a[i]$, and $b[i]$, meaning that initially, the i-th consecutive range of numbers from $a[i]$ to $b[i]$ is written on the sign at mushroom $u[i]$ pointing towards mushroom $v[i]$.

Output Format

Your program must print to standard output.

Output one integer, the maximum possible utility after freely making at most k edits.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $2 \leq n \leq 150\,000$
- $1 \leq m \leq 300\,000$
- $0 \leq k \leq 10^{12}$
- $1 \leq u[i], v[i] \leq n$ and $u[i] \neq v[i]$ for all $1 \leq i \leq m$
- $1 \leq a[i] \leq b[i] \leq n$ for all $1 \leq i \leq m$
- $u[i] < a[i]$ or $b[i] < u[i]$ for all $1 \leq i \leq m$
- If $i \neq j$ and $u[i] = u[j]$, then $b[i] < a[j]$ or $b[j] < a[i]$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Score	Additional Constraints
0	0	Sample test cases
1	6	$n \leq 200, m \leq 400, k = 0$
2	6	$n \leq 1500, m \leq 3000, k = 0$
3	22	$n \leq 1500, m \leq 3000, k \leq 10$
4	11	$n \leq 1500, m \leq 3000, k \leq 1000$
5	7	$n \leq 1500, m \leq 3000$
6	20	$n \leq 30\,000, m \leq 60\,000, k = 0$
7	15	$n \leq 30\,000, m \leq 60\,000$
8	13	No additional constraints

The rest of this page is intentionally left blank. Sample test cases start on the next page.

Sample Test Case 1

This test case is valid for all subtasks.

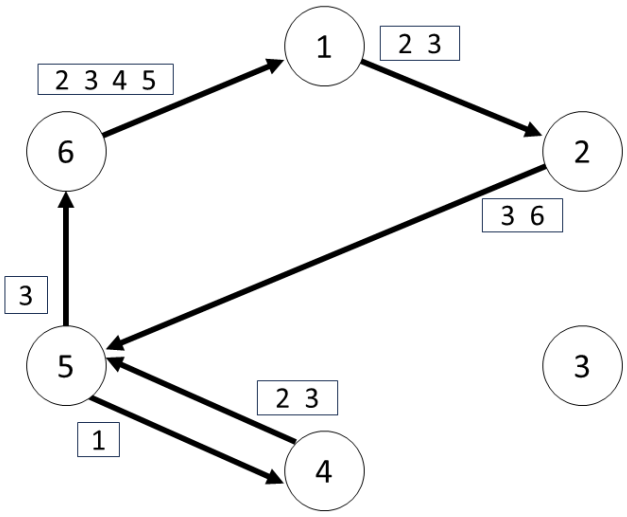
Input	Output
6 7 0 1 2 2 3 2 5 3 3 2 5 6 6 4 5 2 3 5 4 1 1 5 6 3 3 6 1 2 5	8

Sample Test Case 1 Explanation

The diagram below shows the mushrooms and non-empty signs in the village.

Suppose a snail wants to move from mushroom 6 to mushroom 2. He finds the sign containing the number 2, which points to mushroom 1, so he moves there. Since he is not at mushroom 2 yet, he finds the sign containing the number 2 again. It points to mushroom 2, so he moves there. Thus, he can successfully reach mushroom 2 by following signs.

The ordered pairs (s, d) are $(1, 1)$, $(2, 2)$, $(3, 3)$, $(4, 4)$, $(5, 5)$, $(6, 6)$, $(1, 2)$, and $(6, 2)$. The utility is 8. Since $k = 0$, no edits can be made, hence the answer is 8.



Sample Test Case 2

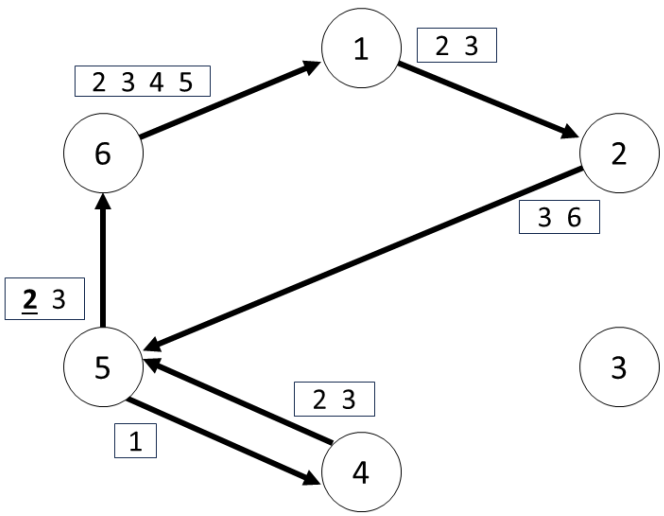
This test case is valid for subtasks 3, 4, 5, 7, and 8.

Input	Output
6 7 1 1 2 2 3 2 5 3 3 2 5 6 6 4 5 2 3 5 4 1 1 5 6 3 3 6 1 2 5	10

Sample Test Case 2 Explanation

The signs are initially the same as Sample Test Case 1. We can make $k = 1$ edit.

One optimal solution is to add the number 2 to the sign at mushroom 5 pointing to mushroom 6. In addition to the 8 ordered pairs (s, d) in Sample Test Case 1, we have $(4, 2)$ and $(5, 2)$. The utility is 10.



Sample Test Case 3

This test case is valid for subtasks 3, 4, 5, 7, and 8.

Input	Output
6 7 2 1 2 2 3 2 5 3 3 2 5 6 6 4 5 2 3 5 4 1 1 5 6 3 3 6 1 2 5	13

Sample Test Case 3 Explanation

The signs are initially the same as Sample Test Case 1. We can make $k = 2$ edits.

One optimal solution is to delete the number 3 from the sign at mushroom 6 pointing to mushroom 1, and add the number 3 to the sign at mushroom 6 pointing to mushroom 3. This allows a snail to move to mushroom 3 using signs, starting from any mushroom. The utility is 13.