A. LRU cache

Description

In our lab, we learned a variation of the LRU algorithm that is used in database buffer pool. Now, Let's implement it.

Building your own data structure to solve the problem is recommended. But feel free to use LinkedList, LinkedHashMap or other containers if you want to.

Input format

The first line contains two integers N and M, donating the capacity of young sublist and old sublist, respectively.

The second line contains one integer K.

In following K lines, every line may have three type of instructions:

- **`put key value`**: update the value of the key if the key exists. Otherwise, add the key-value pair to the cache.
- **`get key`**: print the value of the key if the key exists, otherwise print **`-1`**.
- **`print`**: print the cache contents in order from newest to oldest, in form of **`key:value`**. If the cache is not full, ignore the empty places.

Specifically, there are six moves of changing cache:

- get a key in the old list: put the pair in head of the young list.
- get a key in the young list: put the pair in head of the young list.
- get a key in the neither: do nothing.
- put a pair in the old list: put the pair in head of the young list.
- put a pair in the young list: put the pair in head of the young list.
- put a new pair: if the young list is full, put the pair in head of the old list; if not, put the pair in head of the young list

Output format

For each **`get key`** and **`print`** instruction, print one line according to the description above.

Samples

Sample input

```
2 2
10
put 1 1
put 2 2
get 1
put 3 3
put 4 4
print
put 5 5
get 3
put 5 4
print
```

Sample output

```
1
1:1 2:2 4:4 3:3
-1
5:4 1:1 2:2 4:4
```

Limitations & Hints

$1 \le M, N \le 10^4$

$1 \le K \le 10^5$, where the number of **`print`** is less than 50.

$1 \le key, value \le 10^9$

Reference: https://dev.mysql.com/doc/refman/5.7/en/innodb-buffer-pool.html. This link just give a real world example of the LRU algorithm, which isn't exactly same as the problem.

B. Dream

Description

One day FluffyBunny finds herself trapped in a dream. In her dream, there are N rooms and M one-way corridors. Each corridor is defined as (x_i, y_i, z_i), which means there is a corridor from x_i to y_i, which takes z_i seconds.

However, naughty Satori is casting spells at the same time so that the time required to move between rooms is constantly changing. Specifically, every time after FluffyBunny moves on one corridor, the time spent on **all** corridors changes from z_i to $f(z_i)$, in which $f(x) = \frac{1+x}{1-x} \bmod p, \quad x \in (1, p-1)$.

It is guaranteed that p is a prime number and $f(z_i)$ is defined in any time.

FluffyBunny, of course, wants to escape from the dream. She notices that the exit is room N and she is currently at room 1. Please tell her the minimum time required to get to room N.

Input format

The first line contains three integers N, M, p.

The next M lines describe the one-way corridors. Each line contains three integer x_i, y_i, z_i, denoting that there is a corridor from room x_i to y_i,which takes z_i seconds.

Output format

One integer,denoting the shortest time.

Samples

Sample Input

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

Sample Output

```
4
```

Limitations & Hints

$1 \leq N, M \leq 10^5, 5 \leq p \leq 10^9$. It is guaranteed that p is a prime number.

$1 \leq x_i, y_i \leq N, 1 < z_i < p-1$. It is guaranteed that there is no self loop.

Hint 1: Calculating $f = \frac{a}{b} \bmod p$, which is equivalent to find an integer $b^{-1}(1 \leq b^{-1} \leq p-1)$ to meet $b \times b^{-1} = 1 \bmod p$, can be translated into calculating $f = a \times b^{-1}$.

Hint 2: According to Fermat's Little Theorem, $a^{p-1} \equiv 1 (\mod p)$, where p is a prime number.

Hint 3: To calculate a^n in $O(\log n)$ time, please refer to https://oi-wiki.org/math/quick-pow/