RESTART

Unfortunately, while you thought you were all set to graduate in 7 semesters (and already rented an office for your startup!), you suddenly realize that you never actually satisfied your CS theory requirement. Being forced to stay in school another semester, you decide to take CS124.

This situation has some advantages - since without the skills from 124 you'd never be able to make your startup's server fast enough - but the workload and an upcoming quiz means work on the startup has halted.

One practice question goes as follows: You are given a **connected, undirected, and weighted** graph on $N$ vertices with $M$ black edges. You are also given $K$ triples $(a_i, b_i, c_i)$, which mean you can add a white edge between $a_i$ and $b_i$ with length $c_i$.

You want to ensure the graph remains connected, but minimize the total length of the edges ("**total-edge-length**"). To do so, you can **remove any (even all) of the black edges, and then add any legal white edges**.

You soon realize that since one option is to just leave the original graph with all black edges (it was connected initially), the optimal solution will have total-edge-length no more than the original. What is the most length you can save - the **largest difference between the original total-edge-length and a total-edge-length you can achieve** after removing some black edges and adding some white ones?

## Input

The first line contains three space-separated integers, $N$, $M$, and $K$.

The next $M$ lines contain the black edges. The $i+1$'st ($1 \leq i \leq M$) contains three space-separated integers $p_i, q_i, r_i$, indicating a black edge between $p_i$ and $q_i$ of length $r_i$. It is guaranteed that $1 \leq p_i, q_i \leq N, p_i \neq q_i, 1 \leq r_i \leq 10^5$.

The next $K$ lines contain the legal white edges in the same way. The $j+M+1$'st ($1 \leq j \leq K$) contains three space-separated integers $a_j, b_j, c_j$, indicating a legal white edge between $a_j$ and $b_j$ of length $c_j$. It is guaranteed that $1 \leq a_j, b_j \leq N, a_j \neq b_j, 1 \leq c_j \leq 10^5$.

## Output

A single number - the maximum amount you can reduce the total-edge-length by removing some black edges and adding some white ones, while ensuring the final graph is still connected.

## Constraints

In all test cases, $1 \leq N \leq 10^5, 1 \leq M, K \leq 1.5 \cdot 10^5$. Beyond the sample input, the tests are divided into batches with additional constraints. Time limits below are for C/C++; Ocaml gets 2x, Java 3x, and Python 10x.

- 12 points worth satisfy $1 \leq N \leq 1000$. TL: 100ms.
- 20 points worth satisfy $1 \leq N \leq 10000$. TL: 400ms.
- 68 points worth satisfy no further constraints. TL: 1000ms.

## Sample explanation

The original graph has total-edge-length $9$. If you remove the black edges between $(2, 3)$ and $(1, 3)$, and add a white edge between $(1, 3)$, then you get a connected graph with total-edge-length $4$. Thus, the answer is $9 - 4 = 5$.

View submissions (https://cs124.seas.harvard.edu/problem/RESTART/code-submission)

Test cases

| Input | Output | Points | Timeout |
|---|---|---|---|
| 3 3 3<br>1 2 3<br>2 3 4<br>1 3 2 | 5 | 0 | 300 ms |
| *Hidden* | *Hidden* | 3 | 150 ms |
| *Hidden* | *Hidden* | 3 | 150 ms |
| *Hidden* | *Hidden* | 3 | 150 ms |
| *Hidden* | *Hidden* | 3 | 150 ms |
| *Hidden* | *Hidden* | 5 | 500 ms |
| *Hidden* | *Hidden* | 5 | 500 ms |
| *Hidden* | *Hidden* | 5 | 500 ms |
| *Hidden* | *Hidden* | 5 | 500 ms |
| *Hidden* | *Hidden* | 17 | 1500 ms |
| *Hidden* | *Hidden* | 17 | 1500 ms |
| *Hidden* | *Hidden* | 17 | 1500 ms |
| *Hidden* | *Hidden* | 17 | 1500 ms |

Download (https://cs124.seas.harvard.edu/problem/RESTART/test-cases)

Inspired by the "Ultra Cool Programming Contest Control Centre" by Sonny Chan.
Modified for CS 124 by Neal Wu (https://github.com/nealwu), with design help from Martin Camacho.
Further refined by Nikhil Benesch (https://github.com/benesch).