Problem 4 - Dungeons of Sacred Abyss (100 pts + 20 bonus pts)

Problem Description

Dungeons of Sacred Abyss is a brand-new game developed by the software company *Nextgen Technology Universal Company of Software* (NTUCS) in order to fill the video game market after **Dragon's Song: Awakening** has failed due to power inflation. This new game is designed to give players a unique immersive experience while discovering treasures through traversing the magnificent dungeons and the beauty of the sacred abyss.

In each game, the map consists of N dungeons and the player is initially located at dungeon 0. After the game starts, M pathways between dungeons u_i, v_i where $u_i < v_i$ with length ℓ_i, will appear one by one. In the regular version of the game, all pathways will be given before the player starts traversing, but in the bonus version of the game, some pathways might appear during the game. Interestingly, for each dungeon j except dungeon 0, at most one pathway with $v_i = j$ ever exists. That is, each dungeon has a unique "upstream" dungeon. The pathways are bi-directional and will always be accessed by moving downstream first before going back upstream. After the player goes upstream, the pathway will fall apart and disappear. At any timestamp, it is guaranteed that dungeon 0 can be reached from the player's current dungeon by repeatedly going to the upstream dungeon.

During the player's traversal, ze might find some valuable treasure in the current dungeon. Since the dungeons are buried deep underground, the game includes a transportation system to ship the treasures upstream. The system operates under the constraint that each dungeon can store at most one treasure. The constraint is maintained as follows. The system checks if the newly-found treasure can be stored in the current dungeon. If so (i.e. the dungeon is empty), the treasure is stored. Otherwise, the system places the newly-found treasure in the current dungeon, and push the originally-stored treasure to the upstream dungeon. The process (of checking whether the upstream dungeon can store the treasure) is repeated until a empty dungeon is reached, or until dungeon 0 is reached. If an empty and non-zero dungeon is reached, the last-pushed treasure is stored. Dungeon 0 is a special dungeon that cannot be used to store any treasure. Thus, if dungeon 0 is reached, the treasure will be escorted out of the system immediately.

The transportation system operates at a cost. In particular, transporting the treasure through an upstream pathway decreases its value by the length of the pathway. The value may even drop to a negative number. The transportation system will not make the pathway disappear though—only the upstream action of the player can make the pathway disappear.

The game will give player Q instructions on what to do and you, as the player, should follow them exactly. There are six kinds of instructions that may be given to you.

1. **downstream**: go to a downstream dungeon that appeared first (when the map is constructed, or in the order of any **construct** instructions below). Print out the dungeon number. If no such dungeon exists, print `-1`.

2. **upstream**: go to the unique upstream dungeon. Print out the dungeon number. If no such dungeon exists (i.e. you are in dungeon 0), print `-1`. Note that going upstream makes the pathway disappear.

3. **plan**: if a *hypothetical* treasure of value t_i appears in the current dungeon, it will gradually be pushed upstream step by step when future treasures appear. During its upstream journey, the treasure will gradually lose its value. Print out the *smallest* dungeon number where the treasure can still maintain a non-negative value along the upstream journey. In addition, given that this is an instruction to **plan**, no actual treasure gets stored in the dungeon.

4. **guess**: assume that a *hypothetical* treasure of value 0 has been transported to the current dungeon step by step along some upstream journey, without considering whether any dungeon stores a treasure or not, print out the maximum original value of the treasure. Only the pathways that appeared (when the map is constructed, or with any previous **construct** instructions) and have not disappeared (with the **upstream** instruction) need to be considered. Note that the original value of the treasure can also be 0, representing a possible treasure that appears in the current dungeon. In addition, given that this is an instruction to **guess**, no actual treasure gets stored in the dungeon.

5. **discover**: A treasure with value p_i is discovered in the current dungeon—the transportation system will work on storing it.

6. **construct** (the bonus instruction): a new pathway of length ℓ_i from the current dungeon to dungeon v_i appears.

Input

The first line consists of three space-separated integers, N, M, Q. The next M lines each consists of three space-separated integers u_i, v_i, ℓ_i, representing the initial map.

The final Q lines each consists of space-separated integers depending on the instruction given.

- 1, indicating instruction 1 is given.
- 2, indicating instruction 2 is given.
- 3 t_i, indicating instruction 3 is given.
- 4, indicating instruction 4 is given.
- 5 p_i, indicating instruction 5 is given.
- 6 v_i ℓ_i, indicating instruction 6 is given.

Output

- For instructions 1, 2, 3 and 4, print a line as discussed above.
- For instruction 5, if some treasure is escorted out of the system at dungeon 0, print a line as follows.
  - If the value of the treasure has decreased to a negative number when being escorted, print
    `value lost at [dungeon where treasure value first became negative]`
  - If the value of the treasure is still non-negative, print
    `value remaining is [value]`

  Otherwise, nothing needs to be printed out.
- For instruction 6, no printing is needed.

Constraints

- $2 \leq N \leq 10^6$, $0 \leq M \leq N-1$
- $1 \leq Q \leq 10^6$
- $0 \leq u_i < v_i < N$, and all the v_i throughout the M initial pathways and the pathways given by instruction 6 are unique.
- $0 < \ell_i \leq 10^9$
- $0 \leq t_i \leq 10^{18}$
- $0 \leq p_i \leq 10^{18}$
- $0 \leq$ current dungeon index $< v_i < N$
- $M +$ (number of type-6 instructions) $< N$

Subtasks

Subtask 1 (10 pts)

- $M = N - 1$
- Only instructions 1 and 2 are given.

Subtask 2 (15 pts)

- $M = N - 1$
- Only instructions 1, 2 and 3 are given.

Subtask 3 (25 pts)

- $M = N - 1$
- Only instructions 1, 2 and 4 are given.

Subtask 4 (25 pts)

- $M = N - 1$
- Only instructions 1, 2, 3 and 5 are given.

Subtask 5 (25 pts)

- $M = N - 1$
- Only instructions 1, 2, 3, 4 and 5 are given.

Subtask 6 (20 bonus pts)

- No other constraints.

Sample Testcases

Sample Input 1

```
4 3 8
0 1 2
0 2 5
1 3 2
1
3 3
1
1
3 4
3 0
2
2
```

Sample Output 1

```
1
0
3
-1
0
3
1
0
```

Sample Input 2

```
4 3 11
0 2 3
0 1 7
0 3 5
4
1
2
4
1
2
4
1
2
4
1
```

Sample Output 2

```
7
2
0
7
1
0
5
3
0
0
-1
```

Sample Input 3

```
6 5 11
0 1 5
1 2 4
2 3 3
3 4 2
4 5 1
1
5 10
1
5 5
1
1
5 30
5 25
5 50
2
5 15
```

Sample Output 3

```
1
2
3
4
value remaining is 5
3
value lost at 0
```

Sample Input 4 (Bonus)

```
5 2 13
0 1 4
1 4 5
4
6 2 11
4
1
5 10
3 4
5 8
1
5 49
5 0
2
5 1
4
```

Sample Output 4

```
9
11
1
0
value remaining is 6
4
value remaining is 4
1
value remaining is 40
0
```

Hints

- Storing some extra values per dungeon may help you conquer instruction 3 (and instruction 5).
- The maximum possible value in instruction 4 may be related to the maximum possible value of some neighboring dungeons. For instance, for any dungeon with only one downstream dungeon, the answer is trivial, right? :-) Consider extending this idea to other dungeons.
- Fun fact: The TA who proposed this problem suffered from a 6-hour long painful debug session in this problem. Another TA validating this problem also suffered miserably for about 10 hours. *Bugs are everywhere and it is our destiny to keep debugging.* Good luck!