Artifact Acquisition

Objective
Give additional practice with Disjoint Sets.
Give practice with non-trivial offline queries.

Story
You work your way through a seemingly endless series of twisty halls to come across a large casm with pedestals upon which sits some immaculate artifacts. The pedestals are connected by narrow paths each connecting exactly two pedestals. The paths themselves are covered with some slick foliage. Each path can be traversed in both directions.

You watch several archeologists dropping in from the ceiling on to certain pedestals, making their way barely to the artifacts only to fall off when trying to return to their original spot.

Through analysis you have found a weight limit that each pedestal can support in terms of artifact weight. You are now going to help your fellow archeologists determine if they can recover certain artifacts.

Problem
Given the descriptions of the connections of the pedestals and their weight limit, and the description of which pedestal an archeologist will travel between and the weight of the artifact, determine which of ones will be able to cross safely. You can assume that an archeologist's weight will be negligible.

Input
Input will begin with a line containing 2 positive integers, n and m ($1 \le n, m \le 100{,}000$) representing the number of pedestals and the number of narrow paths, respectively. The following m lines will each contain the description of a narrow connecting path. The path description will contain 3 positive integers, u, v, and w, ($1 \le u, v \le n$; $1 \le w \le 1{,}000{,}000{,}000$), representing the identifying numbers of the two connecting pedestals and the maximum supported artifact weight. Following this will be a line containing the integer q ($1 \le q \le 100{,}000$), representing the number of archeologists. The following q lines will each contain 3 integers, s, e, and w ($1 \le s, e \le n$; $1 \le w \le 1{,}000{,}000{,}000$), representing that an archeologist starts at location s, picks up the artifact with weight w at location e, and moves back to location s.

Output
Output should contain q lines each containing either the word "Yes" if an archeologist can escape with the artifact, or a "No" otherwise.

Sample Input	**Sample Output**

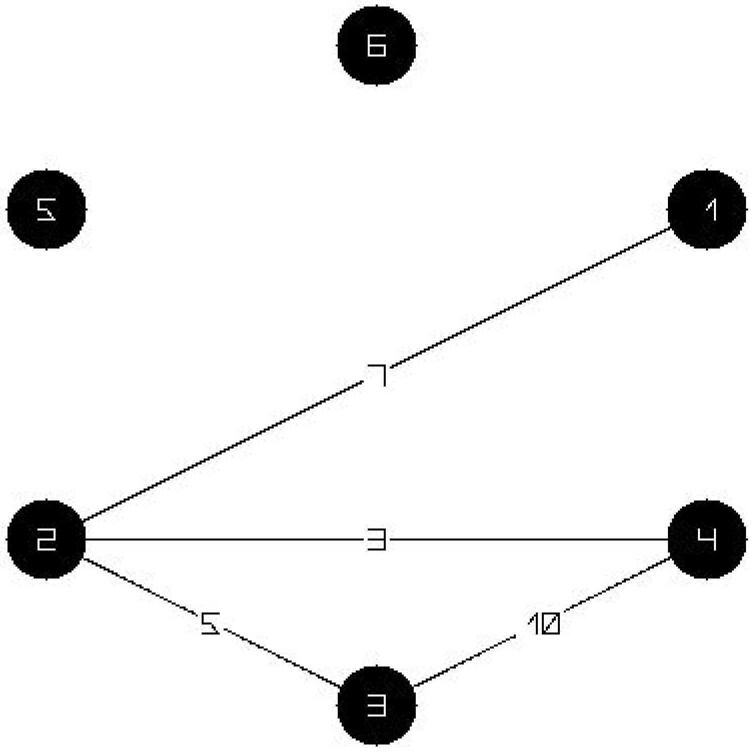
6 4
1 2 7
2 3 5
2 4 3
3 4 10
5
4 2 4
1 4 6
3 4 8
6 1 1
1 3 5
	Yes
	No
	Yes
	No
	Yes

3 5
1 2 10
1 2 3
2 3 2
2 3 5
3 1 1
4
1 2 1
1 3 4
2 3 6
3 2 11
	Yes
	Yes
	No
	No

Explanation

Case 1

The paths look like the following,



The first person (`4 2 4`) is going from pedestal 2 to 4 with a weight of 4. They cannot go directly from 2 to 4, but they can go from 2 to 3 and from 3 to 4. The first person can make the trip. The output is **<u>Yes</u>**.

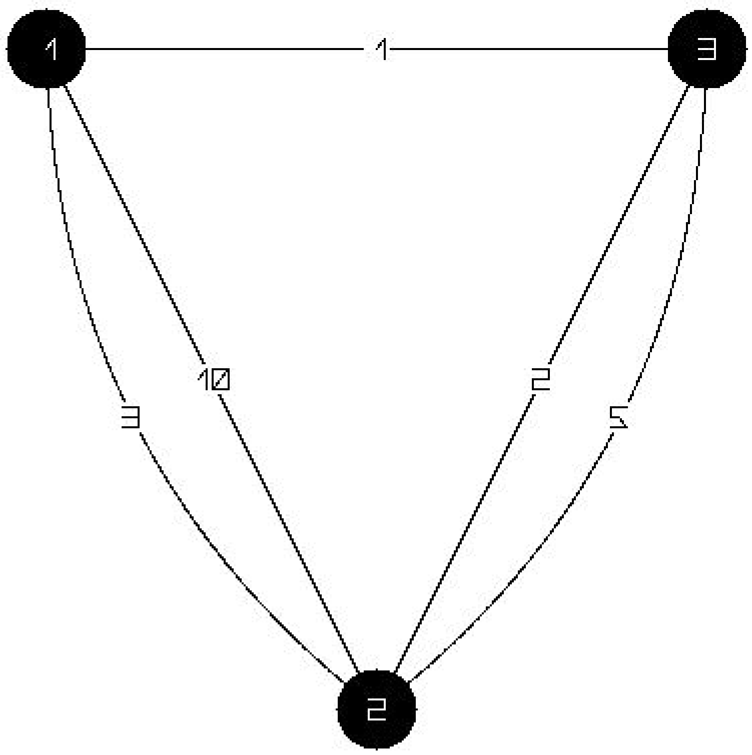
The second person (`1 4 6`) is going from pedestal 4 to 1 with an artifact weight of 6. They cannot go directly from 4 to 1. From 4 they could only reach 3. Unfortunately, the artifact is too heavy to be carried to any other node. The output is **<u>No</u>**.

The third person (`3 4 8`) is going from pedestal 4 to 3 with an artifact weight of 8. They CAN go directly from 4 to 3. The output is **<u>Yes</u>**.

The fourth person (`6 1 1`) is going from pedestal 1 to 6 with an artifact weight of 1. There is no way to reach the 1-st pedestal from the 6-th, so the archeologist would not be able to even attempt a dangerous crossing. The output is **<u>No</u>**.

The last person (`1 3 5`) is going from pedestal 3 to 1 with an artifact weight of 5. The archeologist can barely make it to 2, and from 2 they can reach 1 no problem. The output is **<u>Yes</u>**.

Case 2
The paths look like the following,



There is a path from 1 to 2 that allows for 10 units of weight, so the first person (carrying 1 unit) will make it.

There is a path from 1 to 3 that allows for 5 units of weight, so the second person (carrying 4 units) will make it.

There is a path from 2 to 3 that only allows at most 5 units of weight, so the third person (carrying 6 units) will fall.

There is a path from 3 to 2 that allows for at most 5 units of weight so the fourth person (carrying 11 units) will fall.

Sorting events
For this sample my events would be sorted in the following manner

- Archeologist 4, weight 11, start 3, end 2
- Path 1, weight 10, start 1, end 2
- Archeologist 3, weight 6, start 2, end 3
- Path 4, weight 5, start 2, end 3
- Archeologist 2, weight 4, start 1, end 3
- Path 2, weight 3, start 1, end 2
- Path 3, weight 2, start 2, end 3
- Path 5, weight 1, start 1, end 3
- Archeologist 1, weight 1, start 1, end 2

Hints

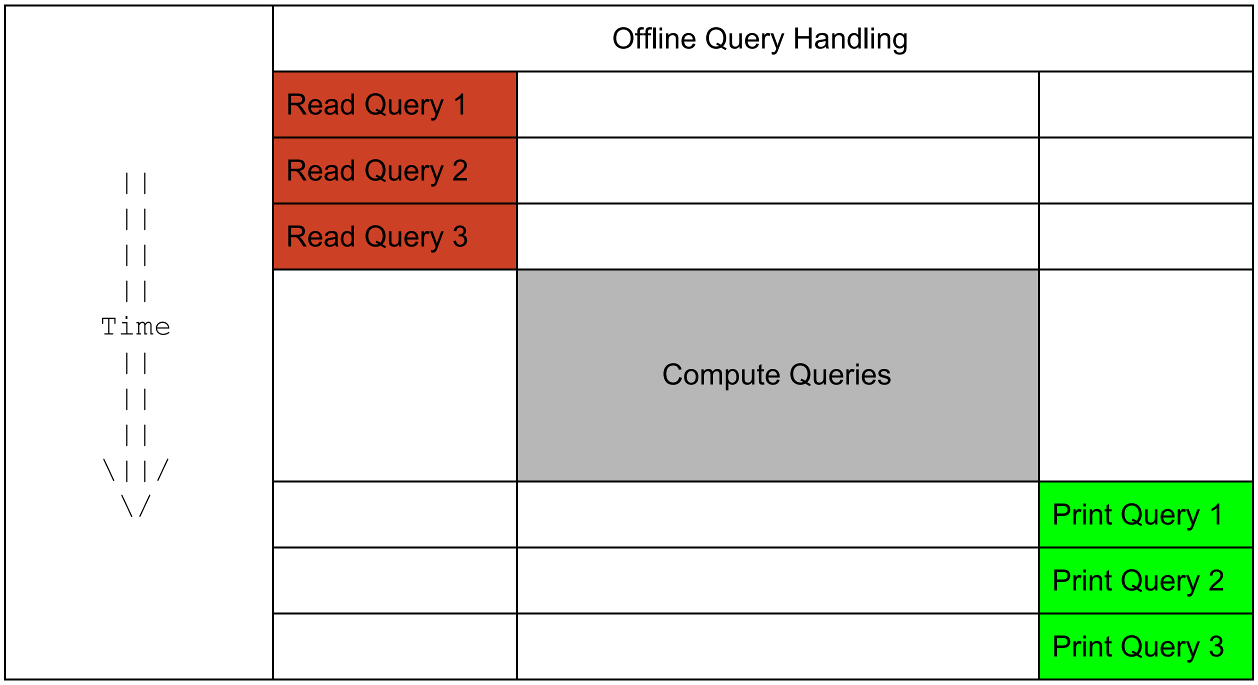
Disjoint Sets: If you know how much weight you are carrying you can merge all pedestals together that share a path that can support our weight. To check if two pedestals have a valid path simply check if two pedestals are in the same Disjoint Set.

Offline Vs Online Queries: Offline query handling is an important technique that will be required for full credit on this assignment. Online query handling is a straightforward approach to tackling a problem.

Online handling involves reading in the query computing the answer and printing it prior to reading in the next query. See table below for an idea

	Online Query Handling		
	Read Query 1		
		Compute Query 1	
| | | |			Print Query 1
| | | |	Read Query 2		
Time | | | | | |		Compute Query 2	
\\\| |/ \\/			Print Query 2
	Read Query 3		
		Compute Query 3	
			Print Query 3

<u>Offline</u> query handling means that you can read in all the input and compute the queries in the order most convenient.



In this case if we change the order of the queries we can compute the answer very quickly, and print the answers quickly. Offline query handling is useful, if some queries can be solved while building up the answers for another (as is the case in this assignment). Offline sometimes might not be possible (e.g. situations in which the answer of one query can change the state of the problem).

Sort Events by Weight: I recommend creating an event object that details what is happening. Either you want to modify the disjoint set that is building up the answer or you want to pose a query.

```
public static class Event {
    int type; // 0 a query and 1 a path connecting pedestals
    int start, end, w;
    int queryIndex; // stores which query this corresponds to
}
```

Create a comparable interface to sort the events and process all the events in their weight order

Sweeping Through by Weight: Imagine if the queries were given in an order sorted from most weight carried to least weight carried, then a solution could add pedestal connectors (our edges) that are usable by all people following some given query (since their weight is decreasing). We would never have to worry about removing a connector! So if we arrange all the queries in order of weight and solve them, we could quickly determine the answer for any given person.

The main idea is to always keep moving forward through an array (either of paths or queries). Going back and redoing effort will increase the runtime.

FAQ

Q: Can the paths be reused by archeologists?
A: Yes, the paths never break.

Q: If an archeologist falls off some path, can another archeologist use it?
A: Yes, the paths never break.

Q: Why is the end index of the query listed as the first point in the description?
A: Because the archeologist only picks up the artifact at the end pedestal.

Q: Can an archeologist use another archeologist rope (exit point)?
A: NO. They are selfish jerks!

Grading Criteria

- Good comments, whitespace, and variable names
    - 15 points
- No extra input output (e.g. input prompts, “Please enter the number of friends:”)
    - 10 points
- Uses an Event class to store BOTH an add path event (based on the max carrying weight) AND a query reachability event (based on the weight of the archeologist).
    - 10 points
- Reads in the FULL input before printing ANYTHING.
    - 10 points
- Merges (and checks) groups of pedestals using the Disjoint Set data structure.
    - 5 points
- Programs will be tested on <u>10 cases</u>
    - 5 points <u>each</u>

<u>Failure to read from standard input will result in -10 points.</u>

No points will be awarded to programs that do not compile using javac.

Sometime a requested technique will be given, and solutions without the requested technique will have their maximum points total reduced. For this problem use a disjoint set to represent connected pedestals. ***<u>Without this programs will earn at most 50 points!</u>***

Any case that causes a program to return a non-zero return code will be treated as wrong. Additionally, any case that takes longer than the maximum allowed time (the max of {5 times my solutions time, 10 seconds}) will also be treated as wrong.

<u>No partial credit will be awarded for an incorrect case.</u>