Travel in HackerLand

by muratekici

Problem Submissions Leaderboard Discussions Editorial

HackerLand is a country with n beautiful cities and m undirected roads. Like every other beautiful country, HackerLand has traffic jams.

Each road has a *crowd value*. The crowd value of a path is defined as the maximum crowd value for all roads in the path. For example, if the crowd values for all roads are $[1, 4, 5, 6, 3]$, then the crowd value for the path will be $max(1, 4, 5, 6, 3) = 6$.

Each city i has a type value, T_i, denoting the type of buildings in the city.

David just started his vacation in HackerLand. He wants to travel from city x to city y. He also wants to see at least k different types of buildings along the path from x to y. When choosing a path from city x to city y that has at least k different types of buildings along the path, David always selects the one with the *minimum* crowd value.

You will be given q queries. Each query takes the form of 3 space-separated integers, x_i, y_i, and k_i, denoting the respective values for starting city, destination city, and minimum number of unique buildings that David wants to see along the way. For each query, you must print the minimum crowd value for a path between x_i and y_i that has at least k_i different buildings along the route. If there is no such path, print `-1`.

Note: A path may contain cycles (i.e., the same roads or cities may be traveled more than once).

Input Format

The first line contains 3 space-separated integers denoting the respective values for n (the number of cities), m (the number of roads), and q (the number of queries).

The second line contains n space-separated integers describing the respective building type for each city in array T (where the i-th value is T_i and $1 \leq i \leq n$).

Each of the m subsequent lines defines a road in the form of 3 space-separated integers, x_i, y_i, and u_i, defining an undirected road with crowd value u_i that connects cities x_i and y_i.

Each of the q subsequent lines defines a query in the form of 3 space-separated integers, x_j, y_j, and k_j (where $0 \leq j < q$), respectively.

Constraints

- $1 \leq n, m, q \leq 10^5$
- $1 \leq T_i, u_i \leq 10^9$
- $1 \leq x_j, y_j, k_j \leq n$
- Each road connect 2 distinct cities, meaning no road starts and ends in the same city.

Output Format

For each query, print its answer on a new line.

Sample Input

```
7 6 1
1 1 4 5 1 3 2
1 2 3
```

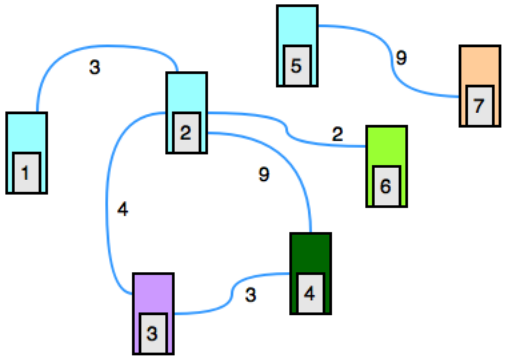
```
2 6 2
2 3 4
3 4 3
2 4 9
5 7 9
1 2 4
```

Sample Output

```
4
```

Explanation

The diagram below depicts the country given as *Sample Input*. Different values of T_i are shown in different colors.



The path for the last query (`1 2 4`) will be $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 6 \rightarrow 2$. David sees his first type of building in cities 1 and 2, his second type of building in city 3, his third type of building in city 4, and his fourth type of building in city 6. The crowd values for each road traveled on this path are $[3, 4, 3, 3, 4, 2, 2]$; the maximum of these values is 4. Thus, we print 4 on a new line.

Submissions:102

Max Score:80
Difficulty: Hard

Rate This Challenge:

More

Current Buffer (saved locally, editable) Java 7

```
1  import java.io.*;
2  import java.util.*;
3  import java.text.*;
4  import java.math.*;
5  import java.util.regex.*;
6
7  public class Solution {
8
9      public static void main(String[] args) {
10         /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. */
11     }
12 }
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code