Lëtzebuerger Informatiksolympiad 2021

Finals 27.4.2021

Task Descriptions

INSTRUCTIONS

- All the programs must be realized in the form of console applications.
- Under the input is meant either the direct entry of data from the keyboard or the redirection from a text file in console mode. Under output is meant either the direct display of data to the screen or the redirection to a text file in console mode (see remarks on the site *www.infosolympiad.lu* under the heading *Les questionnaires*).
- The formats of the data as well as of the results shown in the execution examples must absolutely be respected.
- The allowed programming language is exclusively C/C++.
- For testing respectively for submitting a program, the source file (*.c/*.CPP) must be uploaded to the automated online judge CMS (Contest Management System), accessible via the homepage *www.infosolympiad.lu* or directly via the URL *http://158.64.46.20*. Please use your personal login (username & password) to access your account on the CMS. The filename of the source file should be the same than the task name). Please refer to the CMS for technical details on how to test and submit a program.
- Please refer to the CMS for technical details like time limits and memory limits.

TASK 1 CIPHER 50 POINTS

Description

You have discovered an old archive with encrypted documents. An encrypted document contains ***N*** numbers and you suspect that it has been encrypted in the following way.

If ***K*** is the length of the longest strictly increasing subsequence (not necessarily continuous) of the ***N*** numbers of the decrypted document then each number in the document is **XOR**ed with ***K***.

To verify your suspicion you want to find all possible ciphers ***K*** given an encrypted document with ***N*** numbers.

Task

Compute all ciphers ***K*** and output them in increasing order.

Constraints

$1 \leq N \leq 5000$.

$0 \leq n_i < 2^{32}$.

Input and output of the program.

Input

The first line contains the integer ***N***.

The ***N*** next lines contain each the respective number n_i.

Output

All possible ciphers ***K***. Each line contains one cipher ***K*** in increasing order.

Execution example 1

Input file	Result
3 10 7 15	2

There are three cases:

K=1, then the potential decrypted document is {10 **XOR** 1, 7 **XOR** 1, 15 **XOR** 1} = {11, 6, 14} whose longest strictly increasing subsequence is 2, e.g., 11 → 14. As 2 ≠ 1, we can reject this candidate.

K=2, then the potential decrypted document is {10 **XOR** 2, 7 **XOR** 2, 15 **XOR** 2} = {8, 5, 13} whose longest strictly increasing subsequence is 2, e.g., 12 → 13.

K=3, then the potential decrypted document is {10 **XOR** 3, 7 **XOR** 3, 15 **XOR** 3} = {12, 4, 16} whose longest strictly increasing subsequence is 2, e.g., 9 → 16. As 2 ≠ 3, we can reject this candidate.

Execution example 2

Input file	Result
3 3 2 1	2 3

There are three cases:

K=1, then the potential decrypted document is {3 **XOR** 1, 2 **XOR** 1, 1 **XOR** 1} = {2, 3, 0} whose longest strictly increasing subsequence is 2, e.g., 2 → 3. As 2 ≠ 1, we can reject this candidate.

K=2, then the potential decrypted document is {3 **XOR** 2, 2 **XOR** 2, 1 **XOR** 2} = {1, 0, 3} whose longest strictly increasing subsequence is 2, e.g., 0 → 3.

K=3, then the potential decrypted document is {3 **XOR** 3, 2 **XOR** 3, 1 **XOR** 3} = {0, 1, 2} whose longest strictly increasing subsequence is 3, e.g., 0 → 1 → 2

Subtasks

Subtask	Points	Description
1	10	$N \leq 20$
2	20	$N \leq 100$
3	20	No further constraints

Note

The **XOR** operation is represented in C/C++ by ^ and it is self inverse, i.e.,

let **a** and **b** be two numbers, if **a^b = c** then **c^b = a**.

This means that we have

encrypted_text = clear_text ^ LIS(clear_text)

and

clear_text = encrypted_text ^ LIS(clear_text),

where **LIS(...)** is the longest strictly increasing subsequence and the operation is performed element wise..

More examples

Input file	Result
5 1 1 2 2 3	2

Input file	Result
20 793 794 795 772 773 774 775 768 769 770 771 780 781 782 783 776 777 778 779 820	3 20

TASK 2 ROADTRIP 50 POINTS

Description

Alice and Bob are currently planning a roadtrip, which they would like to undertake with an eletric rental car. However due to the sparsity of charging stations in Lioland they need to take into account their layout while planning the trip, as well as the size of the battery of their car. For simplicity's sake they at the moment only consider the problem of getting from one charging station to another.

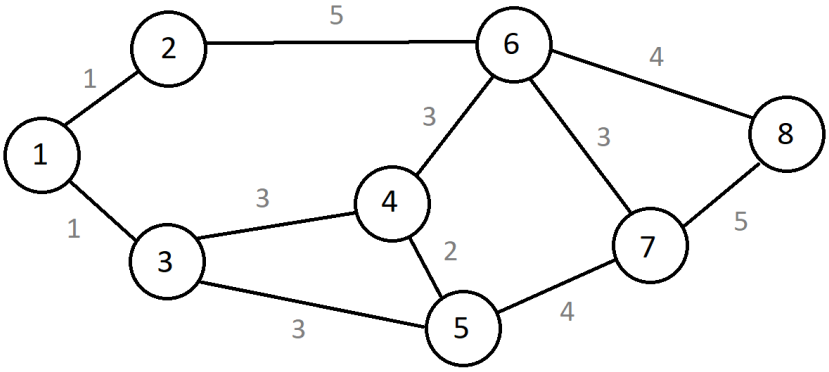
Suppose there are N stations labeled from 1 to N, which are connected by M roads. The j^{th} road has length l_j joins the stations with numbers a_j and b_j. The unit of length is chosen such that every car at their local car rental shop requires one unit of charge for every unit of length travelled. Now, a trip between the stations s and t with a car that has a capacity c is possible if they do not run out of charge inbetween stations. They may recharge at every station they encounter, as electricity is cheap in Lioland.

Since cars at the rental service become more expensive the bigger their battery is, Alice and Bob would like to know what the smallest possible capacity is with which they can succesfully drive between certain stations.

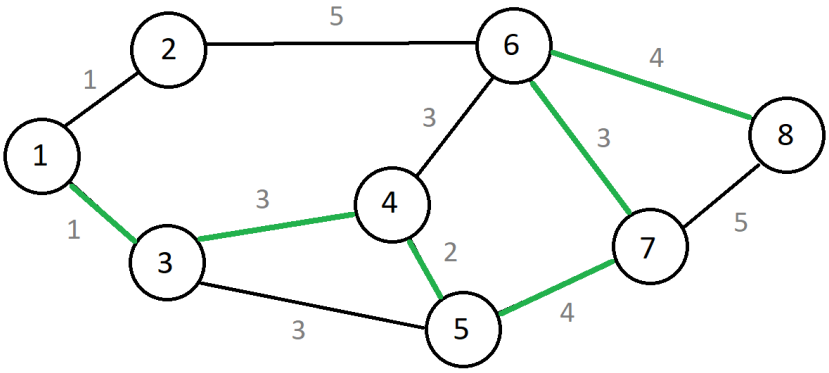
As it is quite tedious to do compute this by hand for the Q pairs of stations they are interested in, the two travelers have asked you to write a program that does it for them.

Example

Let's consider the following layout of N = **8** charging stations, which are joined by M = **11** roads.

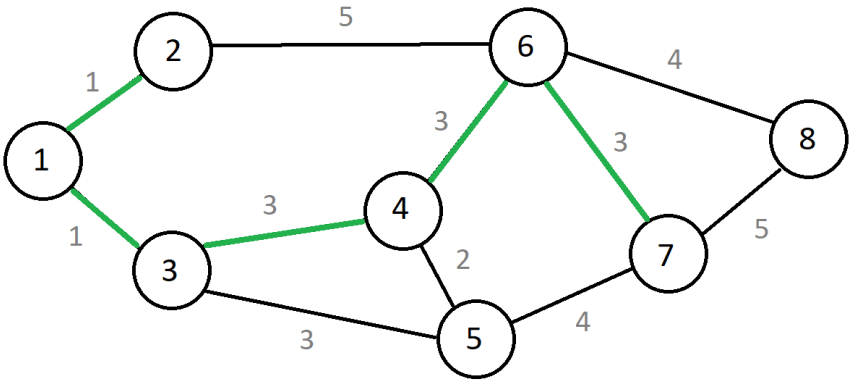


We would for instance like to know what the minimum battery capacity of a car needs to be if it should be able to travel between stations s = **1** and t = **8**. It turns out that the car needs to have a battery which can hold at least **4** units of charge, as exemplified by the following path:



Notice in particular that the path does not need to be minimal, it is simply required that edge road should length **4** or less. You can check that no path with edges of length no more than **3** join station **1** to station **8**. Thus the answer is indeed **4**.

If we were to ask the same question about stations **s** = **2** and **t** = **7**, the answer is **3**, as the roads with length **4** can be avoided in this case.



Task

Given the charging station layout, answer ***Q*** questions of the form "What is the minimum battery capacity needed to drive from station ***s*** to station ***t*** without running out of charge inbetween two gas stations?".

Input and output

Input data

The first line contains three integers, ***N***, ***M***, and ***Q***, separated by a space.

The $(1+j)^{th}$ line contains the data describing the j^{th} road, namely a_j, b_j and l_j, separated by a space, for $1 \le j \le M$.

The $(1+M+k)^{th}$ line contains the data describing the k^{th} request, namely s_k and t_k.

Output data

The program produces ***Q*** lines of output. The k^{th} line should contain a single integer, namely the minimum battery capacity of a car able to drive between stations s_k and t_k.

Execution example

Input	Output
8 11 2 1 2 1 1 3 1 2 6 5 3 4 3 3 5 3 4 5 2 4 6 3 5 7 4 6 7 3 6 8 4 7 8 5 1 8 2 7	4 3

Constraints

$1 \le N \le 10000$.

$1 \le M \le \min(100000, N(N-1)/2)$, and the power station network is connected, i.e. it is possible to travel between any to stations with a battery of size N.

$1 \le Q \le 1000$.

$1 \le a_j, b_j, s_k, t_k \le N$.

$1 \le l_j \le 10000$.

Distribution of points

Subtask	Points	Description
1	10	$N \le 300$
2	5	$M = N - 1$
3	15	$Q = 1$
4	20	No further constraints