**Algorithms with Java: Exam 28-06-2020**

This document defines the exam for ["Algorithms – Fundamentals (Java)" course @ Software University](https://softuni.bg/trainings/2991/algorithms-fundamentals-with-java-may-2020). Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/Contests/2484/Algorithms-Fundamentals-with-Java-Exam-28-June-2020).

# 3. Black Mesa

*Black Mesa undertakes research in various fields from standard scientific research to radiation, rocketry, theoretical physics, lasers, experimental propulsion, hydraulics, robotics, hydroelectricity, genetics, zoology, applied mathematics, and a very wide spectrum of research into chemistry and all manner of physics research. The research is mostly conducted in the Black Mesa Research Facility and includes both pure research and applied science.*

*"Working to make a better tomorrow for all mankind." ― The Black Mesa motto*

You have been accepted to work as a part of the high end research laser team I guess congratulations are in order… or have been in order.

You see working with high energy lasers to condense the space time continuum sometimes has some strange side effects. And as your gut feelings suggest there has been quiet small malfunction which resulted in unexpected and quiet interesting incident.

The space time around the laser research facility has gone wild things are happening in multiple time dimensions, the problem seems to be multiple wormholes opened after the last experiment which caused many alternative universes to cross at those points.

Your task is very simple, since this is not the first incident the security protocol is in place.

You know one thing you can go **from** **a** **previous** **version** **to** **the** **corresponding** **next** **version** in straight timeline, think about it as a persistent model of the multiverse. There is one thing that you don’t know and you can't find it anywhere in the protocol from which **version** you should **retrieve** **all** **the** **steps** to the **latest** **version** before the incident. However you are lucky one of your fellow scientists has **remembered** the initial state and the latest state, or at least he claims he did, do you trust him?

You need to use those values in order to compute the new input settings for the lasers so you can reverse the process and fix the multiverse.

First you will be given – **N an integer** –the total **count of version**, then you will be given – **M an integer –** the **count** of **transitions** from one version to the other.

After that you have to read the persistent state model of the multiverse as **integers** from the console represented in the following format:

* **{prevVersion} {nextVersion}**

And on the last **two** **lines** your colleague will tell you the **start** **version** and the **target** **version** you want to reach.

From now on the first thing left is to print the **shortest sequence** of **versions** from the start to the target on a single line separated by a single space.

On the second line you have to print **all** **the** **versions** you **cannot** reach from the **start** **version** on a single line separated by a single space in **increasing order of the version**.

## Input

The input will come from the console:

* On the first line the number of versions **N** single integer
* On the second line the number of connections between the versions **M**
* On each **M** line the data describing the versions in a straight timeline:

**{prevVersion} {nextVersion}**

* On the next two lines two integer the **start** version and the **target** version you need to go to

## Output

* On the first output line print the versions you pass through **from** the **start** to the **target** version.
* On the second line print the versions you **cannot** go to from the **start** version in **increasing** **order**. If there **are** **none** (**all** **versions** are **reachable** from the **start** one) **print** **nothing**.

## Constraints

* All input lines will be **valid integers you do not need to check that.**
* The range of the integers will be in the range **[1…1000]**
* The **versions** **number** will be **numbers** from **one** **increasing** for each **next** version.
* There will **always** be a **path** from the **start** **version** to the **target** **one**, **however** there may **not** **always** be versions **unreachable** from the **start** one

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 6  6  1 2  2 3  3 4  4 5  6 5  6 4  1  5 | 1 2 3 4 5  6 |
| 11  11  5 11  1 4  5 10  7 8  8 2  2 3  3 4  4 1  6 2  9 10  11 9  6  1 | 6 2 3 4 1  5 7 8 9 10 11 |

# *“Study hard what interests you the most in the most undisciplined, irreverent and original manner possible.” ― Richard Feynmann*