Algorithms and Data Structures 2019: First Assignment

October 8, 2019

1 Instructions

Below are two problem descriptions. The first one (Raspberry Pi supercluster) is easier than the second (Garbage Collection). You only have to make one, but the maximal grade for the first one is 8. You are allowed to submit a solution in either Java, C++, C or Python. You are only allowed to use the Standard Library corresponding to your selected language. You will find a set of examples to test your solution on Brightspace. Besides handing in code, we would like to receive a report – **of at least 2 and at most 10 pages** – in which you explain your algorithm and analyse its correctness and runtime complexity.

The deadline for sending in your solution is November 12, 15:30, Nijmegen time. You can submit your solutions via Brightspace. You are allowed to work in groups of two persons. Only one team member has to submit a solution; the names of both team members must be mentioned in the report.

2 Grading

Grades will be determined as follows. You may earn up to 100 points for your solution:

- 20 points for the explanation of your algorithm.
- 10 points for the correctness analysis.
- 10 points for the complexity analysis.
- 50 points for the test results. We will be running several tests and you will get points for every correct answer within the time limit. If your code does not compile or does not read and write via `stdin` and `stdout`, you will get zero points on the test cases. So please test your code on the test server that will be available! Examples of reading/writing input/output for the first problem can be found on Brightspace along with this assignment.
- 10 points for the quality of the code.

The grade for the first problem equals the total number of points divided by 12.5, and the grade for the second problem is the total number of points divided by 10. If you hand in solutions for both problems then your grade will be the maximum of the grades for the two submissions.

If you have questions, do not hesitate to contact Timo Maarse, `t.maarse@student.ru.nl`.

3 Rasberry Pi supercluster

John has a big pile of Raspberry Pi's and he already connected several of them. For his next application he wants to build a network that connects all his Pi's.

Now, cables are quite cheap and he can transfer data along them with a high data rate, so length of the cables is not an issue. Connectors, on the other hand, are not so cheap, so John wants to connect all Pi's with a minimal number of cables (of course, the current connections already follow this rule). Also the Pi's need some time to process communication, so John wants a path from one Pi to another to pass through as few other Pi's as possible. In fact, John wants to minimize the maximum number of Pi's on any path in the network. He asks you - an expert in Algorithms and Data Structures - to write some software that will help him solving this problem. More specifically, he asks you to design and implement an algorithm which, for a given network, computes the maximum number of Pi's on any path in the network.

3.1 Input

John gives you (via `stdin`) the following data:

- On the first line, you get two integers: the number $1 \leq p \leq 10^6$ of Pis and the number $0 \leq l \leq p-1$ of connections.
- Then l lines with each two integers a and b between 0 and $p-1$ (inclusive), denoting the endpoints of a connection.

3.2 Output

You should return (via `stdout`) the maximum number of Pis on any simple path (that is, a path on which each Pi occurs at most once), excluding the end points (so for the longest path $X \to Y \to Z$, the output would be 1).

3.3 Examples

Sample input 1:

```
6 4
2 0
1 0
5 3
4 3
```

Sample output 1:

```
2
```

Sample input 2:

```
12 9
0 1
0 2
3 4
4 5
3 6
3 7
8 9
9 10
9 11
```

Sample output 2:

```
3
```

4 Garbage Collection

The municipal administration of Nijmegen wants to keep the city clean and neat. In order to do this, they want to place garbage bins on as many street intersections as possible. This means the garbage collectors have to make more stops on their daily routes, something they are not happy about. After discussion between the city and the garbage collectors a compromise has been reached: the city will place as many garbage bins as possible, but if two intersections are adjacent to each other at most one of them will have a garbage bin.

Both parties are happy with this agreement, and the city has already ordered the bins, but placing them in the city turns out to be a much harder problem than they anticipated. They have asked you - an expert in Algorithms and Data Structures - to find out if it is possible to place all the bins in the city.

4.1 Input

The local government of Nijmegen gives you (via `stdin`) the following data:

- The first line contains integers n, m and k, $(1 \leq n \leq 200, 1 \leq m \leq 100, 1 \leq k \leq 20)$, which denote the number of streets in Nijmegen, the number of intersections in Nijmegen and the number of bins which are already ordered. It is given that on every intersection a maximum of 4 streets meet.
- Then n lines, each containing two integers a and b, indices of intersections that are connected by a street (and thus adjacent). Intersections are numbered 1 to m. Moreover, a and b are always distinct (so an intersection is never adjacent to itself) and no two streets can connect the same two intersections.

4.2 Output

Your output (via `stdout`) should be a string: "possible" if the city can place k bins, "impossible" if it can not.

4.3 Examples

Sample input 1:

```
8 7 4
1 2
1 4
2 3
2 5
4 5
5 6
5 7
6 7
```

Sample output 1:

```
impossible
```

Sample input 2:

```
9 8 3
1 2
1 4
2 3
2 5
4 5
5 6
5 7
6 8
7 8
```

Sample output 2:

```
possible
```