Algorithms and Data Structures 2020: First Assignment

October 8, 2020

1 Instructions

Below are two problem descriptions. The first one (Evacuating People) is easier than the second (Group Testing for COVID19). 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 3.[1] 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. See a specific problem's subsection on test results for more (important) information.
- 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`.

[1] If you want to do the group testing assignment, you can contact Timo Maarse about other languages, but not for the evacuation problem.

3 Evacuating People

It is 2050 and due to climate change sea water levels have risen significantly. A dangerous hurricane is approaching the Netherlands and if the dikes break the whole western part of the country will be flooded. We need to evacuate the people quickly! There are several roads connecting the endangered cities with the eastern part of the country. The goal is to evacuate everybody from the endangered cities in the west to a few designated cities in the east. We need to evacuate everybody as fast as possible, and your task is to find out what is the maximum number of people that can be evacuated each hour given the capacities of all the roads.

3.1 Input

The first line of the input contains integers $n \leq 30\,000$ and $m \leq 100\,000$ — the number of cities and the number of roads respectively. The second line of the input contains integers e and d, denoting the number of endangered cities in the west and the number of designated cities in the east (so $e + d \leq n$; note that we may also have cities that are neither endangered nor designated). The third line lists the e endangered places, using 0-based indices of the corresponding cities. Similarly, the fourth line lists the d designated cities. Each of the next m lines contains three integers u, v and c describing a particular road — start of the road, end of the road and the number of people that can be transported through this road in this direction in one hour (roads are one-way as we obviously cannot violate traffic rules, but there might be both roads from u to v and from v to u).

3.2 Output

You should return (via `stdout`) the maximum number of people that can be evacuated per hour. (Note that you might not evacuate people from a specific city at all while still evacuating the maximum number of people per hour.)

3.3 Examples

Sample input 1:

```
4 4
1 1
0
3
0 1 400
0 2 600
1 3 500
2 3 700
```

Sample output 1:

```
1000
```

Sample input 2:

```
7 8
2 2
0 4
6 3
0 4 100
4 1 600
4 3 500
5 4 300
5 6 100
5 2 500
1 2 300
2 3 750
```

Sample output 2:

```
800
```

3.4 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! If you pass all test cases on the test server, you can rest assured you will get a good number of points for the test results, but this does not guarantee the maximum number of points for the test results.[2]

If you encounter any issues, ask for help early!

[2] However, you can assume this if your solution works for all valid inputs and finishes well within the time limit.

4 Group Testing for COVID19

If we want to determine for a group of people which members (if any) have a certain disease, and we are in a situation where the likelihood that an individual has the disease is low but the cost of testing is high, we may consider to organize testing as illustrated in Figure 1. Here we see a scenario in which 16 persons need to be tested. Rather than

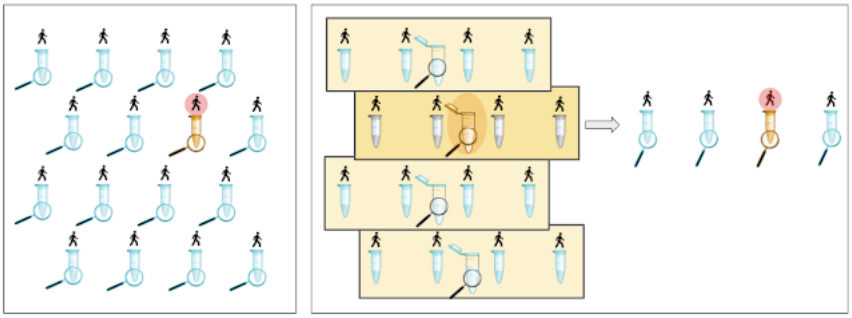

Figure 1: Group testing

testing the samples collected from each person individually, we can combine half of the samples of four persons together and test this combined sample for the disease. If this test is negative then we may conclude that none of these four persons has the disease. In case the test is positive, we use the remaining half of the samples to test all four people individually. Figure 1 illustrates a scenario in which 1 person has the disease: we only need 8 tests to figure out who this person is, rather than 16 tests that are required when we test every person individually. A variant of this problem was first studied by Robert Dorfman in 1943, when he wanted to minimize the expected number of tests to weed out all syphilitic men in a given pool of soldiers. His work started the new field of *group testing*, which has many and diverse applications. (In the A&D exam from 2018 — available via the course page in Brightspace — we considered the problem of finding defects in a collection of software packages using a divide & conquer strategy.)

As you probably know, test capacity for COVID19 is currently a major problem facing society and thus many test laboratories resort to group testing techniques. On the course pages (see Content → Practicum1), we have added some links to recent articles from the news and scientific literature that discuss strategies for group testing for COVID19. None of these strategies, however, takes into account any additional information about close contacts between persons that are tested. In this assignment, while abstracting away from many of the difficulties that arise in real COVID19 testing (outcomes of tests may be unreliable, maximum group size,..), we try to use this information to improve the efficiency of group testing.

Problem: Design and implement an efficient (i.e. using a minimal number of tests) group testing algorithm for a setting with an undirected graph $G = (V, E)$, where nodes represent persons, and edges represent that persons have been in close contact (living

in the same house, close friends,..) Figure 2 shows an example graph. Most graphs your program will receive (but certainly not all) will follow a relatively realistic structure containing small cliques[3] (but possibly of size 1).

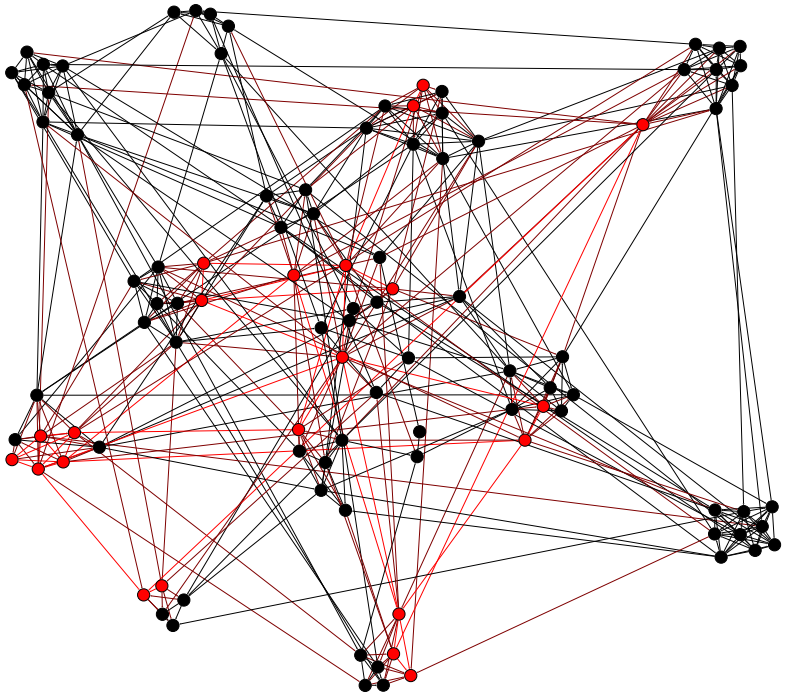

Figure 2: Spreading COVID19 in a small social network

[3]A clique is a subset of vertices such that every pair of vertices is adjacent.

4.1 Input and output

The program communicates interactively via standard in- and output. All indices are 0-based. Your program should end each line of output with a newline and flush the output immediately. It will *first receive the number of problems* on a single line, and then, for each problem:

1. The program receives as input the social graph:
   (a) A line with the number of nodes n
   (b) A line with the number of edges
   (c) A line with the number of initially infected people
   (d) A line with the chance p that people infect a close contact with $0 < p < 1$ and formatted as `0.` followed by decimals[4]
   (e) A line with a lower bound l and upper bound u on the number of total infections, with $0 < l \leq i \leq u \leq 0.7n$, with i the number of infections (just like in the real world, you already have an indication of how many people are infectious)
   (f) For each edge, a line with the two indices of the connected nodes
2. Now, your program should continuously output either:
   (a) A line starting with the word `test` and a whitespace-separated list of nodes to test in a single group. In response the program will get as input either `true` in case someone has tested positive, or `false` otherwise, both on a single line.
   (b) A line starting with the word `answer` and a whitespace-separated list of nodes that are precisely the nodes infected with COVID19. If the answer is correct, the program receives as input `success`, and `failure` otherwise, both on a single line. In any case, the program should move on to the next problem (and therefore receives a new graph as input) or exit if there are no more problems.

[4] The set of infected people is determined by the following process with discrete steps: we start with the (randomly chosen) set of initially infected people and then, in each step, infect close contacts of people infected in the previous step with chance p, until no new infections have occurred. (This is essentially the process modelled by the Independent Cascade Model using fixed edge weights.)

4.2 Example

Sample input and output (input given to the program is black, output given by the program is red):

```
2
3
2
1
0.3
1 2
0 1
0 2
test 0 1
true
answer 0
failure
6
7
1
0.5
1 4
0 1
0 2
1 2
1 3
2 4
3 4
3 5
test 0 1 2
true
test 3 4 5
false
test 0
true
test 1
true
test 2
false
answer 0 1
success
```

4.3 Test results

We will be running your program on a single input with many problems of varying sizes. You will get a very reasonable time limit for all test cases combined (in the order of minutes, but you will not get any points if you exceed it).

Your score on the test results will be determined by the quality of your solution (i.e. the number of tests required) and the ranking of your solution's quality compared to other students' solutions. If you hand in a trivial solution that does not exploit the additional information about the social network (for instance, test every person and return all positives, or the simple scheme of Figure 1), you can expect a low total grade.

There will be a custom test server (different from the one for the evacuation problem that is usually used) with a scoreboard[5] and instructions on how to use it. Please use it to test your solution! If your code does not compile or does not read and write via `stdin` and `stdout` correctly, you will get zero points for the test results.

If you encounter any issues, ask for help early!

[5]Note that the scoreboard is only for motivation, your final results will be determined on a secret, fixed set of graphs and infections.