Project 2

The theme of the project is to study experimentally how the network reliability depends on the individual link reliabilities, in the specific situation described below.

Network topology: A complete undirected graph on $n = 5$ nodes. Complete means that every node is connected with every other one (parallel edges and self-loops are excluded in this graph). Therefore, this graph has $m = 10$ edges, representing the links of the network.

Components that may fail: The links of the network may fail, the nodes are always up. The reliability of the links is computed as follows. Index the links with the numbers 1,...,10 (in arbitrary order). Take a parameter p, $0 \leq p \leq 1$, the same for every link. The parameter p will take different values in the experiments. For link i, $(i = 1, 2, \ldots, 10)$, let its reliability be p_i, which is given by the following formula

$$p_i = p^{\lceil d_i/3 \rceil},$$

where d_i is the i^{th} digit in your 10-digit student ID, assuming the ID consists of the digits $d_1, d_2, \ldots, d_{10}$. The $\lceil ... \rceil$ sign means upper integer part, which rounds up any non-integer number to the nearest larger integer. For example, if your ID is 0123456789, then link 5 has reliability $p_5 = p^{\lceil d_5/3 \rceil} = p^{\lceil 4/3 \rceil} = p^2$, and link 1 has reliability $p_1 = p^{\lceil d_1/3 \rceil} = p^0 = 1$.

Reliability configuration: The system is considered operational, if the network is connected.

Specific tasks:

1. Create an algorithm to compute the network reliability in the above described situation, using the method of exhaustive enumeration (see in the Lecture Notes). Note that the high level description given in the notes is not enough, since you also have to specify how you actually want to find the details, such as how to generate the possible states, how to assign an up/down system condition to each, how to convert it into a reliability value, etc.

***IMPORTANT:* Finding algorithmic solutions for these details is part of the task!**

Describe how your algorithm works. First briefly explain informally the ideas. Then provide pseudo code for the description of the algorithm, with sufficient comments to make them readable and understandable by a human.

2. Write a computer program that implements the algorithm. You may use any programming language under any operating system, this is entirely of your choice. Make sure, however, that your program is well structured to support finding potential errors (debugging), checking correctness or trying out algorithm changes. Explain how your program supports these goals. Include a section that tells how to run your program (this is usually called ReadMe file).

3. Run the program for different values of p. Let the parameter p run over the $[0.05, 1]$ interval, in steps of 0.05. Show graphically in a diagram how the obtained network reliability values depend on p.

4. Now fix the p parameter at $p = 0.9$, and do the following experiment. Among the $2^{10} = 1024$ possible combinations of component states pick k of the combinations randomly, and flip the corresponding system condition. That is, if the system was up, change it to down, if it was down, change it to up. This models the situation that there are some random errors in the computation. Show in a diagram, how the reliability of the system changes due to this alteration. With this we want to investigate how the result depends on random errors. Specifically, show in a diagram how the change depends on k, in the range $k = 0, 1, 2, 3, \ldots, 20$. During this experiment keep the value of the parameter p fixed at $p = 0.9$. To reduce the effect of randomness, run several experiments for each value of k, and average them out,

5. Provide a short (1-2 paragraph) explanation why the obtained diagrams look the way they look. In other words, try to argue that they exhibit a reasonable behavior that one could intuitively expect, so the program is likely to work correctly.

Note: If something is not specified in this project description, that means it is left to your choice.

Submission guidelines

Describe everything, including algorithms, program, sources, results and figures neatly and clearly in a study. Include everything in a *single document* that can be read as a report. It should have a professional appearance, scanned handwriting is not acceptable! The preferred file type is pdf. Do not submit executable code, but include the source code as an Appendix in the document. The project report will be read as a document and not run as a program (but there are two exceptions, see them under Evaluation).

Submit the document through eLearning. Do not send it via e-mail!

Notes:

- The work should be fully individual and original. Any form of cheating is a serious violation of University policies and can lead to serious consequences. Note that some of the data depend on the student ID, which is unique for everybody, so no two projects can generate the exact same results, even if the same random number generator is used.
- It may be helpful to think about the whole project presentation that your task is not only to solve a technical problem, but you also have to "sell" the results. Try to look at your work from the viewpoint of a potential customer, to whom you want to sell such a software product. How convincing would your presentation look for a customer?

Evaluation

The evaluation will focus on how well each of the specific tasks have been carried out. Even though the submission will not be run, only read as a document, there are two exceptions. You will be asked to demonstrate on a computer how your program actually runs, if any of the following cases occur:

1. You do not agree with the grade and want to improve it. In this case the demonstration should show that your work is actually better than the received grade.

2. There is suspicion that the work is not original or not individually done or the results were not produced by your own correctly running program. In this case a demonstration is required to clarify the situation and to receive any score.