Project 2

The subject of this project is to create an implementation of the Nagamochi-Ibaraki algorithm (see in the lecture notes) for finding a minimum cut in an undirected graph, and experiment with it.

Tasks:

1. Explain how your implementation of the algorithm works. Provide pseudo code for the description, with sufficient comments to make it 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.

3. Run the program on randomly generated examples. Let the number of nodes be fixed at $n = 20$, while the number m of edges will vary between 40 and 400, increasing in steps of 5. Once a value of m is selected, the program creates a graph with $n = 20$ nodes and m edges. The actual edges are selected randomly among all possible ones, with parallel edges allowed, but self-loops are excluded.

4. Experiment with your random graph examples to find an experimental connection between the average degree of the graph and its edge-connectivity $\lambda(G)$. (If the graph happens to be disconnected, then take $\lambda(G) = 0$.) How does the connectivity depend on the average degree? (Note that the average degree in a graph characterizes the density of the graph, as it is expressible as $d = 2m/n$, where m, n are the number of edges and nodes, respectively.) Show the found relationship graphically in a diagram, exhibiting $\lambda(G)$ as a function of the average degree d, while keeping $n = 20$ fixed.

5. Let us call an edge *critical*, if its deletion decreases the connectivity. That is, an edge e is critical, if $\lambda(G-e) < \lambda(G)$, where $G-e$ denotes the graph G with edge e deleted. Let $C(G)$ denote the number of critical edges in graph G. Using the random graphs generated above, show an

experimental relationship between the number of critical edges and the average degree d of the graph. That is, show $C(G)$ graphically as the function of d, while keeping $n = 20$ fixed.

6. Give a short explanation why the functions found in items 4 and 5 look the way they look. In other words, try to argue that they indeed show the behavior that can be intuitively expected, so your program is likely to work correctly. Note that it is part of your task to find out what behavior can be expected.

Important note: If there is anything that is not specified in this project description, that automatically means it is left to your choice.

Submission guidelines

Describe everything, including algorithms, program, results and figures neatly and clearly in a study. Include everything in a *single document* that can be read as a report and submit it through eLearning. (Do not send it via e-mail!) Do not include executable code, but include the source code in an appendix to the study. Your submission will be read as a report, but usually we do not run the program. On the other hand, if either there are signs of non-original work or if you want to dispute the received score, then you will be asked to demonstrate your program on a computer, show and explain its details.

Notes:

- The work must be fully individual and original. Any form of cheating is a serious violation of University policies and can lead to serious consequences.
- 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", who either wants to buy such a software, or perhaps wants to hire somebody to carry out similar tasks. How convincing would your presentation look for such a customer?