Project 1

The theme of this project is to implement the basic network design model that is presented in the lecture note entitled "An Application to Network Design", and experiment with it.

Specific Tasks:

1. Create software that is capable of doing the following:

   - As input, it receives the number of nodes (N), the traffic demand values (b_{ij}) between pairs of nodes, and the unit cost values for the potential links (a_{ij}).
   - As output, the program generates a network topology, with capacities assigned to the links, according to the studied model, **using the shortest path based fast solution method** (see at the end of the cited lecture note). The program also computes the total cost of the designed network.

   Important notes:

   - Any programming language and operating system can be used, it is your choice.
   - For the shortest path algorithm you may download and utilize any existing software module from the Internet. If you use this opportunity, then include in your documentation a precise reference that tells where the module comes from.

2. Clearly explain how your program works. It is helpful to use flowcharts for visualizing the explanation.

3. Run your program on randomly generated examples, as explained below.

   - Let the number of nodes be $N = 36$ in each example.
   - For each example, generate the a_{ij}, b_{ij} values according to the rules described below. In these rules k is a parameter that will change in the experiments.

- For generating the b_{ij} values, pick independent random integers from the range $[0, 1, 2, 3]$.
- For generating the a_{ij} values, do the following. For any given i, pick k random indices $j_1, j_2, \ldots, j_k$, all different from each other and from i. Then set

$$a_{ij_1} = a_{ij_2} = \ldots = a_{ij_k} = 1,$$

and set $a_{ij} = 299$, whenever $j \neq j_1, \ldots, j_k$. Carry out this independently for every i.
Remark: The effect of this is that for every node i there will be k low cost links going out of the node, the others will have large cost. The shortest path algorithm will try to avoid the high cost links, so it effectively means that we limit the number of links that go out of the node, thus limiting the density of the network topology.

- Run your program with $k = 3, 4, 5 \ldots, 15$. For each run generate new random a_{ij}, b_{ij} parameters independently.

4. Show graphically in diagrams the following:

   - How does the total cost of the network depends on k?
   - How does the density of the obtained network depends on k? Here the density is defined as the number of directed edges that are assigned nonzero capacity, divided by the total possible number of directed edges, which is $N(N-1)$.
   - Show some of the obtained network topologies graphically. You do not have to draw all of them (it would be too many), just select a few characteristic examples, to demonstrate what happens in case of low, medium and high k values.

5. Provide a brief (1-2 paragraph) verbal justification that explains why the obtained diagrams look the way they do. In other words, try to convince a reader that what your diagrams show is indeed the "right" behavior, that is, your program that carries out the network design is likely correct.

Notes:

1. Your project document should include a section that is often referred to in a software package as "ReadMe file." The ReadMe file (or section) provides instructions on how to run the program. Even though in the default case we do not plan to actually run the program, we should be able to do it, if we wanted. For example, if something does not seem right, and we want to double check whether the program indeed works correctly. Therefore, instructions should be included on how to run the program, and it has to be sufficiently detailed, so that one can actually run it, if necessary.

2. If there is anything that 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.

Submit the document through eLearning. (Do not send it via e-mail!) Do not include executable code, but include the source code that you wrote, in an appendix to the study. If the executable code is needed, we will ask for it.

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. Also note that while there were *similar* projects in earlier semesters, they are *not exactly the same*. The minor differences between this and earlier similar projects make it easy to detect if a submission is copied from an earlier one.

- 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 document presentation look for a customer?

Evaluation

The evaluation will focus on how well each of the tasks have been carried out. Even though the submission will not be run, only read as a document, but 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 score received and want to improve it. In this case the demonstration should show that your work is actually better than the received score.
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.

Note:

The work should be fully individual and original. Any form of cheating is a serious violation of University policies and can lead to very serious consequences.