

# Project - Railway network DV1490

## 1 Task description

In this project, you will work with the management of new railway networks. The railway networks will be represented by directional graphs where the nodes are different stations and all links have a cost of some kind. An example of this can be seen in Figure 1.

You must implement functionality to determine the cheapest way to lay rails while all the planned stations are interconnected. That is, you must create a Minimum Spanning Tree (MST), either using Kruskal's or Prim's algorithm. The weights in the graph's links represent the cost of adding space between the nodes.

For the project, there is no given API to be followed, but you are free to make your own design regarding how you go about solving the problem, and how your solution is implemented. You may use the standard packages available in Java but no third-party packages or ready-made solutions. Is a representation of the graph and the solution must be self-implemented but may use functionality from standard packages internally. There are also a number of included files to use as input as described in the next section. Your program should produce a response file that follows a certain format. The file formats are described in section 1.1.

In addition to your implementation, a report must also be written in which you address, among other things, how you have solved the problem and which algorithms and data structures you have used. For a more detailed description of the report, see section 2.

Your program is expected to run through the command prompt and receive a parameter. The incoming parameter contains a path to a valid input file, for example

`"C:/Users/JZN/Documents/TrainGraph.txt"`.

An example of how the execution of your program can be performed is

```
java .\StudentSubmission "C :/Users/JZN/Documents/T rainGraph.txt00 "Good Town–Great City00
```

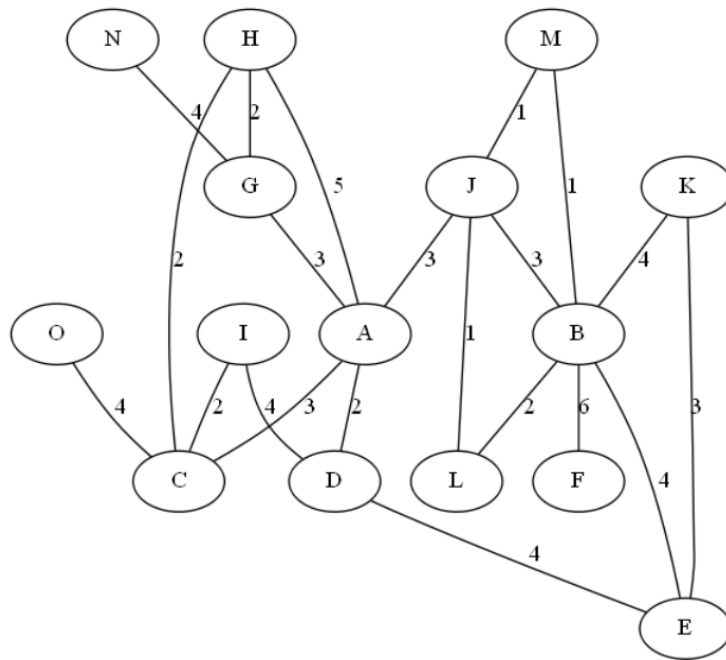


Figure 1: Example of an unoriented graph containing nodes and anchors with weights

Where StudentSubmission is your program, the first quoted text is a path to an existing and valid file, and the second quoted text is instructions that are only relevant if the cheapest route has been implemented. You can test this in Eclipse by going to run, run configurations, arguments and set the desired ones.

## 1.1 File format

Your program should load the data contained in the file, process it, and then print a response to a text file named *Answer.txt*. The files to be used as input are structured as follows

- A newline-separated list of all nodes in the graph ends with an empty line
- A newline-separated list with all links on the form "A B weight" where A and B are nodes that were in the first list, and weight is the weight of the link. The three different parts are separated by tabs.

Below you will find an example of what an input file might look like.

Alpha  
Beta  
Gamma  
Delta  
Epsilon

Alpha            Epsilon            3

Alpha	Beta	2
Beta	Gamma	3
Gamma	Delta	5
Delta	Epsilon	3

The response file should be formatted in the same way as the input data, but it should now only contain the links that should remain after your algorithm has run. Below you will find an example of what the answer file should look like.

V6  
V2  
V7  
V1  
V3  
V4  
V5

V6	V7	1
V2	V1	2
V7	V4	4
V7	V5	6
V1	V4	1
V3	V4	2

## 2. Report

In addition to your implementation, the project also contains a report to be written. Your report must be well structured and written in either Swedish or English. Feel free to take a look at the following link for good information regarding how you should proceed. The report should be written in 12pt Times New Roman, be between 4-8 pages, and contain the following sections.

- Description of the problem
  - The problem should be broken down into sub-problems where appropriate
  - It should be clear what needs to be done to solve the problem
- Description and justification of approach
  - Algorithms used / implemented must be described, referenced, and justified why they were chosen in comparison with other possible alternatives.
  - Data structures used / implemented must be described, referenced, and justified why they were chosen in comparison with other possible alternatives.
  - The procedure, how your program processes the data entered through the file and produces an end result, should be described in its entirety
- Analysis of the implementation
  - How do the algorithms and data structures you used relate to in your implementation against what is expected of them? Is these faster / slower than expected? Explain why

in that case as well as justify it. Note, an actual analysis is required where you prove a time complexity either empirically, via a series of runs of is a solution, or theoretically, m.h.a. a thorough analysis of is code.

- Reflections on the work
  - How much time have you spent on the project and how is the time distributed?
  - What preparations were made and how did these affect the work?
  - Did the algorithms and / or data shortcuts implemented perform as you expected? If not, why did not they?
  - Which choices (algorithms, data structures, etc) had you made the same / different
  - if you were currently starting on the project?

### 3. Project implementation

It is allowed to carry out the project either alone or in groups of two people. All code you use must be developed by the group. if you are alone, you must create an MST m.h.a. either Kruskal's or Prim's algorithm. If you work in pairs, you must produce two separate projects where you implement an MST m.h.a. Kruskal's algorithm and the other m.h.a. Prim's algorithm, ie two separate classes, one for each algorithm. Students who work in pairs must also in the report, under the section 'Analysis of implementation', also include a concluding part that compares and evaluates the two algorithms that have been implemented.

### 4. Submission and examination

- Code and report must be submitted.
  - A cleaned project must be submitted along with detailed compilation instructions in .zip file format.
    - This means that only the source code and for the compilation necessary files must be submitted.
    - No binaries or execution files should be sent with.
    - If you work in a group, you send in both your projects in one and the same zip file, but in a separate folder inside.
  - This project should be able to compile and execute without any errors to be assessed.
  - The report must be submitted in .pdf format named as follows: *kurskod studentakronym projektrapport.pdf* and *kurskod studentakronym1 studentakronym2 projektrapport.pdf* if you work in a group.
    - Exempel: *DV1490 jonf13 projektrapport.pdf* or *DV1490 jonf13 chne11 projektrapport.pdf*
    - *The report must be submitted together with the code but separated from the zip file.*