AlgraviS
Software Requirements Specification

Anica Hoppe and Sonja Türpitz

July 18, 2018

The software "AlgraviS" shall read (undirected) graphs from text files, shall allow manual editing of a graph and shall stepwise visualize the algorithms Depth-First Search (DFS), Find Connected Components, Maximal Matching and BUILD. Furthermore, it shall be easily extendable by more algorithms as Breadth-First Search (BFS), Topological Sort, Minimum Spanningtree, etc. by using not only undirected but also directed graphs and showing edge weights in addition.

1 User Requirements

In the following, it is described how to use the software "AlgraviS". Keep in mind, the visualization gets unclear for more than 50 vertices and for vertex names greater than 99. Note, this is a beta-version.

1.1 Algorithm

By clicking on 'Algorithm' a menu shall appear from which you can choose the algorithms 'Depth-First Search', 'Find Connected Components', 'Maximal Matching' or 'BUILD'. Keep in mind, the visualization of some algorithms might take some minutes (depending on the graph size).

The first three algorithms work on a graph that you should have produced before (see 'Edit').

For 'BUILD' you have to choose an input text file, first. This file has to be of the following format:

```
# of leaves
# of triples
each line one number as name of a leaf, e.g. leaves '1', '5' and '78':
1
5
78
each line one triple ((x,y),z) as 'number of x space number of y space number of z', e.g.:
1 78 5
```

By clicking on 'visualize algorithm' a stepwise visualization of the algorithm shall start. In this process the green color means, that an edge/vertex is considered, red means that it is chosen. For 'BUILD' a second window shall appear in which the Aho-graph shall be visualized.

By clicking on 'Show result' the result of the algorithm shall be presented.

1.2 Edit

By clicking on 'Edit' a menu for editing a graph shall appear. You can choose between adding, deleting and moving. For adding/deleting a vertex you have to click once in the window, for an edge you have to click on two vertices. Furthermore, you can add an edge by specifying its end vertices in the appearing window. For moving a vertex you have to click on a vertex and hold it while moving. You can also delete a whole graph.

1.3 File

By clicking on ’File’ you can choose a text file of a graph that shall be drawn. The text file has to be of the following format:

```
’d’ or ’u’ for directed or undirected graph
# of vertices
# of edges
each line one number as name of a vertex, e.g. vertices ’1’, ’5’ and ’78’:
1
5
78
each line one edge as either ’start vertex name space end vertex name’ or ’start vertex name
space end vertex name space edge weight’, e.g.:
1 5 6
78 1
```

2 System Requirements

The software ”AlgraviS” shall be implemented in Java in the way following.

2.1 GUI

The GUI shall have a menu bar with menus for

1. running algorithms
2. editing the graph
3. dealing with text files.

It should also have menus for

4. showing or hiding edge weights
5. showing a help page.

The menu ’Algorithm’ shall have menu items for the algorithms

1. DFS
2. Find Connected Components
3. Maximal Matching
4. BUILD

and should also have menu items for

5. BFS
6. Topological Sort
7. Minimum Spanning Tree.

The menu 'Edit' shall have menu items for

1. adding
   - i. a vertex
   - ii. an edge by drawing (clicking on two vertices)
   - iii. an edge by specifying
2. deleting
   - i. a vertex
   - ii. an edge
   - iii. the complete graph
3. moving vertices.

It should also have a menu item for

4. undoing the last step.

The menu 'File' shall have menu items for

1. opening an existing file for loading a graph.

And should have menu items for

2. opening a new file creating a new text file for a graph
3. saving a displayed graph as a text file
4. showing the text file for the displayed graph

In the following, the menu items are explained in detail.

Menu item **Depth-First Search**	
Function	Runs the Depth-First Search for a start vertex that has to be specified by the user.
Description	After pressing this menu item a window shall appear, that asks for a start vertex for the DFS. If the entered vertex is contained in the graph, DFS shall be executed for this vertex. Then the button "visualize algorithm" can be pressed to see the visualization of the DFS. Otherwise, the status bar shall say that the input was wrong.
Inputs	A graph and a start vertex.
Source	The input graph is the currently shown graph and the start vertex has to be specified by the user.
Outputs	A list of operations that are made by the algorithm and used for the visualization.

Menu item **Find connected components**	
Function	Runs the algorithm Find Connected Components.
Description	After pressing this menu item the algorithm for finding connected components shall be executed. Then the button "visualize algorithm" can be pressed to see the visualization of the algorithm.
Inputs	A graph.
Source	The input graph is the currently shown graph.
Outputs	A list of operations that are made by the algorithm and used for the visualization.

Menu item **Maximal Matching**	
Function	Runs the algorithm Maximal Matching.
Description	After pressing this menu item the algorithm for finding a maximal matching shall be executed. Then the buttons ”visualize algorithm” and ”show result” can be pressed to see the visualization of the algorithm or just the result.
Inputs	A graph.
Source	The input graph is the currently shown graph.
Outputs	A list of operations that are made by the algorithm and used for the visualization.

Menu item **BUILD**	
Function	Runs the BUILD algorithm.
Description	After pressing this menu item a window shall open that asks if the currently shown graph should really be deleted and the BUILD algorithm started. When clicking ’yes’ a text file containing a leafset and tripleset has to be chosen for loading it and running the BUILD algorithm. If a wrong file was chosen or the tripleset in the file is inconsistent, a respective status message shall be given. Otherwise, the algorithm shall run for these sets. Then the buttons ”visualize algorithm” and ”show result” can be pressed to see the visualization of the algorithm or just the result.
Inputs	A leafset and a tripleset.
Source	Read from an input file that has to be chosen by the user.
Outputs	A list of operations that are made by the algorithm and used for the visualization.

After running an algorithm the following buttons shall appear (if necessary).

Button **visualize algorithm**	
Function	Visualizes the just run algorithm.
Description	After pressing this button a stepwise visualization of the just run algorithm shall start. In this process the green color means, that an edge/vertex is considered, red means that it is chosen. For ’BUILD’ a second window shall appear in which the Aho-graph shall be visualized.
Inputs	A list of operations made by an algorithm.
Source	The list is the output of an executed algorithm.
Output	The visualization of the algorithm.

Button **show result**	
Function	Shows the result of the just run algorithm.
Description	After pressing this button the result of a just run algorithm shall be shown.
Inputs	A list of operations made by an algorithm or the resulting graph of the algorithm.
Source	The list is the output of an executed algorithm.
Output	The result of the algorithm.

Menu item **Add vertex**	
Function	Vertices can be added by clicking in the DrawPanel.
Description	After pressing this menu item and then clicking in the DrawPanel/window, a new vertex shall appear and be added to the graph.
Inputs	A position for the new vertex.
Source	Coordinates from clicking the mouse.
Output	A drawn vertex with default name.

Menu item **Add edge (draw)**	
Function	An edge can be added by clicking on two vertices.
Description	After pressing this menu item and then clicking on two vertices, an edge between these vertices shall appear and be added to the graph.
Inputs	Start and end vertex for the new edge.
Source	Coordinates from clicking the mouse.
Output	A drawn edge with default weight.

Menu item **Add edge (specify)**	
Function	An edge can be added by specifying the start and end vertex.
Description	After pressing this menu item a window shall open. There you have to enter two vertex names to add an edge between them. If at least one entered vertex name is not contained in the graph, nothing shall happen.
Inputs	Start and end vertex for the new edge.
Source	Vertex names as specified in the new window.
Output	A drawn edge with default weight.

Menu item **Delete vertex**	
Function	Vertices can be deleted by clicking on it in the DrawPanel.
Description	After pressing this menu item and then clicking on a vertex in the DrawPanel/window, this vertex and all incident edges shall disappear and be deleted from the graph.
Inputs	A vertex to be deleted.
Source	Coordinates from clicking the mouse.
Output	Background where the vertex was before.

Menu item **Delete edge**	
Function	An edge can be deleted by clicking on two vertices.
Description	After pressing this menu item and then clicking on two vertices, an edge between these vertices shall disappear, if existing, and be deleted from the graph.
Inputs	Start and end vertex for the edge to be deleted.
Source	Coordinates from clicking the mouse.
Output	Background where the edge was before.

Menu item **Delete graph**	
Function	A graph can be can be deleted completely.
Description	After pressing this menu item a window shall open that asks if the currently shown graph should really be deleted. When clicking 'yes' an empty DrawPanel/window shall be visible.
Inputs	A graph.
Source	The input graph is the currently shown graph.
Output	Pure background.

Menu item **Move**	
Function	A vertex can be moved by holding it.
Description	After pressing this menu item and then clicking on a vertex in the DrawPanel/window and holding it, this vertex shall move with the mouse as long as the mouse is not released.
Inputs	Old and new position of a vertex.
Source	Coordinates from the mouse.
Output	Background where the vertex was before and the vertex drawn on its new position.

Menu item **Open file**	
Function	A text file can be chosen to draw the graph specified in it.
Description	After pressing this menu item a text file containing a graph can be chosen for loading it. If a wrong file was chosen or the format of the file was wrong, a respective status message shall be given. Otherwise, the specified graph shall be drawn.
Inputs	A graph.
Source	Read from an input file that has to be chosen by the user.
Output	A drawn graph.

A status message shall be given always.

2.2 Graph

For gaphs the vertices shall be specified as integer and the edges shall be specified by their start and end vertex. An undirected graph shall save all its vertices as keys in a hashmap and the respective incident edges as elements, whereby the edges are organized in a treemap (or red-black-tree) where the endpoints of an edge are the keys and the weight of this edge is the respective value. A directed graph shall have a second hashmap constructed in the same way to distinguish between opposite directions of the edges. They both shall save the highest vertex name at all to avoid creating a vertex with an already taken name/number. Both graphs shall have getter for these hashmaps and the highest vertex name. A graph shall have the general methods 'isEmpty' and 'typeOfGraph'. Moreover, it shall have the following methods concerning vertices and edges, respective.

vertices	**edges**
add with(out) specified name	add with(out) specified weight
contains	contains
delete	delete
get list of all vertices	
is adjacent to	
	contains edges at all
	get/set edge weight

2.3 Algorithms

An algorithm shall be stepwise visualizable. Therefore, the method 'execute' shall return a list of operations. Each time, an algorithm influences or uses a graph this action shall be saved as an operation. The operations shall save whether an edge or a vertex was changed, which kind of action it was, and which vertices/vertex the target of that action was. Additionally, an algorithm shall have a method for getting the result as a graph: 'getResult'.

2.4 Visualization

For visualizing a graph a class DrawPanel shall be implemented. This class shall contain a list of coordinates of the vertices of the graph. The coordinates shall be created randomly for vertices that are not added by drawing. The DrawPanel shall also have methods for drawing a complete graph with specified vertex and edge color, drawing an edge with specified color and drawing a vertex with specified color. For drawing vertices the class 'Circle' is used, which can draw a filled circle in a specified color with the vertex name as string displayed in the middle of the circle. It shall also contain a method that changes the graph that shall be drawn and a second one that also draws it directly after changing it.

2.5 Test

Everything has to be tested and documented. For more information see 'AlgraviSDocumentationOfTests.pdf'.