Graph Solver README

Version 0.1-alpha

I. Introduction:

This readme file will be used to instruct in ways of using Graph Solver to solve graph theory problems. The list of graphs will be updated as I add new graph problems. Right now the two graph problems this application can solve is Edmonds-Karp “Network-Flow” problem, and Dijkstra “Shortest-Path” problem. Using the JUNG 2 Java Universal Network Graph framework to represent the different graphs and show some interative graphics. Wanted to implement a Prim's UndirectedGraph solution however needed to spend time implementing a minimum spanning tree.

This graph solver application uses an ANTLR4 Adaptive LL(*) island grammar to parser an XML file and then using the ANTLR4 Listener interface setup a graph and run an identified algorithm against the given graph and then give its solution. It will perform each iterations of the identified algorithm in steps which will be reported to the end-user. The idea is to be an educational tool for use in teaching and understanding graph theory problems.

II. The XML Input File:

I decided to use XML as the language to input a graph because of its relative simplicity and easy of use. With the use of XML it allows the user to be able to focus on how the problem is solved and less on how to describe the problem to be solved. What follows is the template for each graph problem that is supported with the current version of the application. The graph_type value is case sensative and only supports either “edmonds_karp”, or “dijkstra” in all lower case.

A. Edmonds-Karp “Network-Flow” Problem.

The idea of Ford-Fulkerson problems is to have a network of edges that connect together to form a vertex. Each edge will have a Source, Sink (ie. Destination), and Capacity of that specific edge. The vertices will be listed out before-hand.

Edmonds-Karp Graph Input File Template:

```
<workspace>
    <graph_type= ”edmonds_karp”>
        <vertices>
            <vertex id= ”Vertex name” />
            ..….more vertices here.....
            <vertex id= “Vertex name“ />
        </vertices>

        <edges> .. ..more edges here...
            <edge>
                <source id= ”source vertex name”/>
                <sink id= ”destination vertex name”/>
                <capacity value= ”integer value”/>
            </edge>
        </edges>
    </graph>
    <s_node id=”Start Vertex Name” />
    <t_node id= ”Sink Vertex Name” />
</workspace>
```

B. **Dijkstra's *“Shortest Path” Problem.***

The Dijkstra's shortest path problem will first create the vertices required. Then all the edge objects required each to have a source “start” and a sink “destination” vertices. The weight can be only a postitive. There can be any number of edges and any number of vertices required to describe the graph for the algorithm to come up with a solution. The s_node is the identified start vertex that is identified along with the t_node the sink vertex which will be used as the shortest paths start and destination.

Dijkstra Graph Input File Template:

```
<workspace>
    <graph_type=”dijkstra”>
        <vertices>
            <vertex id= ”Vertex name” />
            ..….more vertices here.....
            <vertex id= “Vertex name“ />
        </vertices>

        <edges> .. ..more edges here...
            <edge>
                <source id= ”source vertex name”/>
                <sink id= ”destination vertex name”/>
                <capacity value= ”integer value”/>
            </edge>
        </edges>
        <s_node id=”Start Vertex Name” />
        <t_node id= ”Sink Vertex Name” />
    </graph>
</workspace>
```

III. Requirements and Restrictions.

All requirements for the two different graphs are as such. Graph_type I have already alluded to, however it is case sensative word. The tag names are shown in the templates and the assignments are all reserved words... The reserved words list are as follows; “workspace”, “graph”, “vertices”, “edge”, “source”, “sink”, “s_node”, “t_node”, “weight”, “value”, “id”. The edge tag will not accept any form of <edge id= “xx”> … </edge> format as I have made the edges go off of the source and sink attributes. Neither will the workspace tag have a value attribute of the tag. The workspace is there for defining the start of the file and the end of the file. The “id=” are used for referencing vertex objects, the “value=” are for integer values only. No decimal values (ie. Floats, or Doubles) will be accepted. The XML Declaration tag can be used if you are requiring to specify it but will be ignored for use in this application. Therefore, all *<? ... ?>* and *<?xml … ?>* entries will be ignored; also, along with any whitespace (ie. Tabs, newlines, spacing, etc..). Also very important you have to sourrond all names and values for “id=” or “value=” with quotations “ “.

****Note:** If you want to run the GUI make sure that you run this with the ability to see the graphics representation of the graph else you won't be able to see the graph. Commands are: **Left-Click** move graph, **Shift-Left_Click** drag graph around, **Mouse wheel** zoom/scale in and out.

To compile code:

cd <to root directory of project>

```
javac -cp .:lib/jung-graph-impl-2.0.1.jar:lib/jung-visualization-2.0.1.jar:lib/collections-generic-4.0.1.jar:/usr/local/lib/antlr-4.0-complete.jar test/Driver.java graphAlgorithms/*.java parser/*.java
```

To run sample edmonds-karp input graph:

```
java -cp .:lib/jung-graph-impl-2.0.1.jar:lib/jung-visualization-2.0.1.jar:lib/collections-generic-4.0.1.jar:/usr/local/lib/antlr-4.0-complete.jar test.Driver test/edmondKarp_graph.xml
```

To run samle dijkstra's input graph:

```
java -cp .:lib/jung-graph-impl-2.0.1.jar:lib/jung-visualization-2.0.1.jar:lib/collections-generic-4.0.1.jar:/usr/local/lib/antlr-4.0-complete.jar test.Driver test/dijkstra_graph.xml
```

****Note:** This is of course if you keep all the code and test files in the sub-directories of the tar file. If they are moved around adjust the target directories accordingly. External jars are in lib/, graph code is in graphAlgorithms/, antlr4 code and grammar is in parser/, driver file and test input files and test report in test/.