Programming Project 4 – MST CS146

Graph Algorithms Minimum Spanning Tree

Important: Do in groups of 2 students; each group has to turn in one project.

In this project you are asked to implement one known and one new graph algorithm solving the Minimum Spanning Tree (MST). First, create an undirected graph from a file (see examples), and then use Prim's algorithm and a new algorithm to solve MST.

The new algorithm for finding minimum spanning trees is based upon the following property:

"Pick any cycle in the graph, and let e be the heaviest edge in that cycle. Then there is a minimum spanning tree that does not contain e."

Here is the pseudocode of the new MST algorithm. Given an undirected graph G = (V,E) (in adjacency list representation) with edge weights:

```
sort the edges in descending order according to their weights
for each edge e in E:
     if e is part of a cycle of G (use DFS):
          G = G - e (remove e from G)
return G
```

A linear time algorithm that checks whether there is a cycle containing a specific edge e= (u, v) is to start a DFS from u. If e is a back-edge in the DFS then e participates in a cycle. Use the Adjacency list representation.

Programming Standards:

- Your header comment must describe what your program does.
- You must include a comment explaining the purpose of every variable or named constant you use in your program.
- You must use meaningful identifier names that suggest the meaning or purpose of the constant, variable, method, etc.
- Precede every major block of your code with a comment explaining its purpose. You don't have to describe how it works unless you do something tricky.
- You must use indentation and blank lines to make control structures more readable.
- Use the provided weighted graph examples, in WeightedGraphExamples.zip.
- Provide running times and the value of the MSTs.

Final Project Deliverables:

Once the project is completed, the following is expected of you:

1. Your main grade will be based on (a) how well your tests cover your own

code, (b) how well your code does on your tests (create for all non-trivial methods)

2. Use sjsu.<lastname>.cs146.project4 as your package, and Test classes should be your main java file, along with your JUnit java tests.
3. Zip up the directory with your entire project (source code and report) with your LastNameFirstName.zip. Turn the zip file by uploading it to canvas.
4. All projects need to compile. If your program does not compile you will receive 0 points on this project.
5. Do not use any fancy libraries. We should be able to compile it under standard installs. Include a readme file on how to you compile the project.
6. You should be prepared to answer detailed questions on the system design and implementation if asked. We will also examine your code to check for code quality, code documentation, etc.
7. Prepare a complete project report, which contains details about your project, such as main data structures, main components of the algorithm, design of the user-interface for input/output, experimental results, e.g. charts of running time versus input size, Junit etc.

.