CSCI 2125 Homework 5

Carefully read the following quote from your text:

> "Adjacency lists are the standard way to represent graphs. Undirected graphs can be similarly represented; each edge (u, v) appears in two lists, so the space usage essentially doubles. A common requirement in graph algorithms is to find all vertices adjacent to some given vertex v, and this can be done, in time proportional to the number of such vertices found, by a simple scan down the appropriate adjacency list.
>
> There are several alternatives for maintaining the adjacency lists. First, observe that the lists themselves can be maintained in any kind of List, namely ArrayLists or LinkedLists. However, for very sparse graphs, when using ArrayLists, the programmer may need to start the ArrayLists with a smaller capacity than the default; otherwise there could be significant wasted space.
>
> Because it is important to be able to quickly obtain the list of adjacent vertices for any vertex, the two basic options are to use a map in which the keys are vertices and the values are adjacency lists, or to maintain each adjacency list as a data member of a Vertex class. The first option is arguably simpler, but the second option can be faster, because it avoids repeated lookups in the map.
>
> In the second scenario, if the vertex is a String (for instance, an airport name, or the name of a street intersection), then a map can be used in which the key is the vertex name and the value is a Vertex and each Vertex object keeps a list of adjacent vertices, and perhaps also the original String name."

I have provided you with an implementation of Vertex, I want you to implement MyGraph using the "second option" referred to in the third quoted paragraph above. The required methods are stubbed out for you and must be implemented by you, the programmer. You may use either a preexisting implementation of Map in the Java API, or your own version.

Write an detailed JUnit tester to put your graph through its paces. Every method should be well tested.

Upload your completed code and JUnit tester to your gitlab repository in a subfolder called "HW4". If you attempt the bonus, in must be in a subfolder called either "BONUS1" or "BONUS2" WITHIN "HW4".

BONUS1 (10 points): Implement the Topological Sorting algorithm discussed in the book, and write a JUnit tester proving that it works for several example graphs.

BONUS2 (40 points): Implement Dijkstra's algorithm using your new MyGraph class, and write a JUnit tester proving that it works for several example graphs.