CSCI-3901-Assignment3

Name: Pratheep Kumar Manoharan

Banner Id: B00837436

Files: MainClass.java, VertexCluster.java, Edge.java

Objective:

To create graphs and form cluster of vertices using the minimum spanning tree algorithm with the use of data structures.

Assumptions:

1. A vertex name cannot be an empty string.
2. Zero is not a positive integer. Weights cannot be zero.
3. All operations on the graph are done on a simple graph. (as mentioned in assignment question).

Technical Code Flow:

Used the same MainUI.java class file shared in the previous assignment with slight modifications for this assignment.

Edge Class:

1. Each object of the Edge class represents an edge of the graph.

2. Vertex1, vertex2, and Weight are the attributes of the Edge class.

3. The constructor method is used to create the edge object.

4. Edge class implements Comparable interface to sort the edge list.

5. *compareTo method*

   compareTo method is an overridden method from the Comparable interface. It is used to sort the edges. The edge list is sorted with weight as the first priority, vertex1 as second priority and vertex2 as the third priority. Technically it is done as follows:

   a. Compare the weights of the edges and return if the value is not Zero.
   b. If the weights are equal, then compare the vertex1 of the edges. Return the value if it is not Zero.

    c. When both weights and vertex1 of the edges are equal, then compare the vertex2 value of the edges. Return the resulted value.
    d. Duplication check is done before sorting so when the vertex2 value is compared it will not be zero.

6. Edge class has getter methods to return the value from “this” object.

VertexCluster Class:

1. VertexCluster object has all the details of the graph.

2. allEdges, clustNoVertMap, clustNoMaxWeiMap, vertClustNoMap, and vertMaxWeiMap are the key attributes of the VertexCluster class.

3. allEdges is a list which is used to store all the edges of the graph.

4. clustNoVertMap is a map with cluster number as the key and the set of vertices belongs to the cluster as values.

5. clustNoMaxWeiMap is a map with cluster number as the key and the maximum edge weight of the cluster as value.

6. vertClustNoMap is a map with vertex name as key and cluster number of the vertex as value.

7. vertMaxWeiMap is a map with vertex name as key and the maximum weight of the cluster in which the vertex belongs to as the value.

8. *validateString method*

    a. Gets a string value as input.
    b. Basic string validations null check and empty string validations are done on the input.
    c. Returns false if the string is valid.
    d. Returns true if the string is invalid.

9. *printString method*

    a. Gets a string message as input.
    b. Prints the message to the user.

10. *duplicateCheck method*

    a. Gets an edge object as input.
    b. The input object is compared with the existing objects in the edge list.

c. Returns true if the input is a duplicate of an existing edge.
d. Returns false if the input is not a duplicate.

11. *addEdge method*

a. Gets vertex1, vertex2, and weight as the input parameters.
b. Vertex1 and vertex2 are validated with the validateString method.
c. Weight cannot be zero or negative so validated.
d. If any of the validations is failed, then returns false with an error message to the user.
e. If the vertex1 and vertex2 are the same, then return false with an error message to the user as the same vertex forming an edge of the graph is not in scope. The graph is assumed to be a simple graph.
f. After the basic validations on the input, both the vertices passed to this method are compared. While creating an edge, the vertex with smaller value is set as vertex1 and the vertex with bigger value is set as vertex2. The comparison is done using the string compareTo method.
g. Once the edge object is created, the created edge is passed to the duplicateCheck method to check whether it is a duplicate or a new valid edge. If it is a duplicate edge, then return false with an error message to the user.
h. If it is a valid edge, then add to the edge list and return true.

12. *sameClusterCheck method*

a. Gets an edge object as the input.
b. Vertices of the edge are compared with the cluster number vertices map.
c. If the vertices belong to the same cluster, then return true.
d. If the vertices are not part of any cluster, then return false.

13. *clusterVertices method*

a. Gets the user tolerance as the input.
b. To form the cluster of vertices, all edges are looped.
c. The edge object is passed to the sameClusterCheck method to validate whether the vertices of the edge belong to the same cluster.
d. If the vertices belong to the same cluster, then the current edge manipulation is skipped, and the loop will continue with the next edge.
e. Current edge tolerance is calculated by calling the getEdgeTolerance method.

Non clustering scenario:

f. If the edge tolerance is more than the user tolerance, then the no merge logic is executed.
g. Check whether the vertex cluster number map contains the vertices.
h. If the vertices are present in the map, then update the cluster number after comparing the cluster sequence number and the value already in the map. The lowest cluster number is stored in this map.

i. The cluster number of the vertices will differ since this is a no merge scenario.
j. If the vertex cluster number map does not contain the vertex, then the vertex is put into the map with the cluster sequence number as the value.

Clustering scenario:

k. If the edge tolerance is less than the user tolerance, then the merge logic is executed.
l. Check whether the vertex cluster number map contains the vertices.
m. Compare the cluster number of the vertex1, vertex2, and the cluster sequence number.
n. Smallest cluster number is set into the map for the vertices.
o. Check whether the vertex maximum weight map contains the vertices.
p. Compare the maximum weight of the vertex1, vertex2, and the weight of the current edge.
q. The heaviest weight is set into the map for the vertices.
r. To add the current edge into the cluster number vertices map and cluster number maximum weight map formClusterNoVertMap and formClusterNoWeiMap methods are called respectively.
s. If one of the vertices belong to another cluster already, grouping into a single cluster and deleting the smaller cluster are done using the cluster number.
t. After the loop, the output is formed by calling the getReturnValue method.
u. The required output is returned in the expected format.

14. *getReturnValue method*

a. Cluster number vertices map is validated since the output is formed from that map.
b. TreeSet comparator compare method is used to create a sorting logic. The logic is to sort based on the value of the first vertex of each cluster.
c. Each cluster is looped and added into the TreeSet.
d. All the calculations done on the current tolerance are cleared by calling clearAllMaps method.
e. TreeSet variable is returned with all the clusters in order.

15. *getEdgeTolerance method*

a. Gets the current edge as the input.
b. Check whether the vertices are already in the vertex maximum weight map.
c. If the vertex is already present in the map, then the value from the map is used otherwise ONE is set as the weight of the vertex.
d. The minimum weight of the vertices is stored in the denominator.
e. The current edge weight is divided with the denominator.
f. Resulted value is returned.

16. *formClusterNoVertMap method*

   a. From the vertex cluster number map, a cluster number vertices map is created or if the map is already available, the map will be updated.
   b. In this map, the cluster number is set as key and the set of vertices belong to the cluster number is set as the value.

17. *vertMapLowClustNo method*

   a. The values in the vertex cluster number map and the cluster number vertices map are validated in this method.

18. *formClusterNoWeiMap method*

   a. From the vertex maximum weight map, a cluster number maximum weight map is created or if the map is already available, the map will be updated.
   b. In this map, the cluster number is set as key and the maximum edge weight of the cluster is set as the value.

19. *vertMapMaxWeight method*

   a. The values in the vertex maximum weight map and the cluster number maximum weight map are validated in this method.

20. *clearAllMaps method*

   a. This method is used to clear all the computations done with the current user tolerance.

Test Scenarios:

S. No	**Test scenario**	**expected result**	**combinations to test**
addEdge method			
Input validation test cases			
1	Give vertex1 as a null or empty string and the other two parameters with valid values.	A null or empty string is not a valid input. So, the method will return false.	2
2	Give vertex2 as a null or empty string and the other two parameters with valid values.	A null or empty string is not a valid input. So, the method will return false.	2

3	Give weight as 0 or negative and the other two parameters as valid values.	Zero or negative weight is not a valid weight. So, the method will return false.	2
4	Give vertex1 and vertex2 as a null or empty string and valid weight value.	A null or empty string is not a valid input. So, the method will return false.	4
5	Give vertex1 as a null or empty string, weight as zero or negative and valid vertex2 value.	The null or empty string and zero or negative weight are not valid inputs. So, the method will return false.	4
6	Give vertex2 as a null or empty string, weight as zero or negative and valid vertex1 value.	The null or empty string and zero or negative weight are not valid inputs. So, the method will return false.	4
7	Give all three parameters as invalid null or empty string and zero or negative.	The null or empty string and zero or negative weight are not valid inputs. So, the method will return false.	8
boundary condition test cases			
8	Give an edge and use the characters to represent the edge but in different cases.	The graph should treat them as different edges since they have different cases. So, the method should return true.	1
9	Try to give the same edge again in the same order or different order of vertices.	Duplicate is not allowed. So, the method should return false	2
10	Add an edge with single character vertex names.	The method should return true.	1
11	Add an edge with a large vertices name.	The method should return true.	1
control flow test cases			
12	Add the same weight edges with common vertex to check sorting.	Edges are sorted in the ascending order of weights. If weights are the same, the value of vertex1 is compared. When both weights and vertex1 are the same, then the value of vertex2 is compared and decided on order. So, the method should return true.	1
13	Add an edge with the vertices in descending order. Small vertex should be added as vertex1	While creating an edge, the smaller vertex is set as vertex1 and the bigger one is set as vertex2. Done using the String	1

		compareTo method. So, the method should return true.	
14	Pass the valid parameters in all three parameters.	The method should return true.	1
cluster vertices method			
Control flow test cases			
15	We can give zero, negative or positive value as tolerance.	The method will return the clusters formed. For zero and negative tolerance, each edge will form individual clusters.	3
16	Form cluster, when the same vertex names form edges, but the vertices are in different cases in the graph.	The method will return the clusters formed treating as two different edges.	1
17	Give tolerance value so that all vertices are formed into an individual cluster and no group of vertices.	Each vertex will be a cluster and no group vertices formed a cluster. The method will return the vertices list.	1
18	Give tolerance value so that all vertices are grouped into one cluster.	When the tolerance value is high then all vertices will be combined into a cluster. The method will return the cluster formed.	1
19	Try with a graph in which all vertices are connected.	The method will return the clusters formed.	1
20	Try with a graph in which edges are not connected.	The method will return the clusters formed.	1
21	Give one tolerance value and try to call the clusterVertices method again with different tolerance values.	The second result should not depend on the first result. On each call, the computation should happen correctly.	1
Data flow test cases			
22	Call the addEdge method first and call the clusterVertices method. (Normal order)	Edges will be added, and the cluster will be formed.	1
23	Call the clusterVertices method first. The method will return null since no edges are present in graph to cluster.	Null will be returned.	1