Implementing a Weighted Graph and Prim's Algorithm

Homework #7

Logan Miles

1. Objectives

The objective of this project was to implement a weighted graph using adjacency lists, as well as Prim's algorithm to find the minimum search tree of this graph. The nodes of the weighted graph had to be read from a text file, then added into an undirected graph via a linked list of connected nodes to each node. The text files that were given in this assignment were "tinyDG," "mediumDG.txt," "largeDG.txt," and "XtralargerDG.txt." Each listed the number of vertices in line one and the number of edges in line two. Each line after the first two contains two whole numbers and one decimal number with spaces in between each of them. The two whole numbers represent the connected nodes of an edge, and the decimal number represents the weight of the edge. Several lines also contain an inconsistent number of spaces.

2. Program Design

To implement the required functionality for this assignment, three classes were created: a HW7 class containing the driver code, a WeightedGraph class containing the functions necessary to construct the graph, print the adjacency list, and implement Prim's algorithm, and lastly the Edge class which contains the constructor for each edge. The following functions are contained within these three classes:

WeightedGraph()

This constructor initializes a WeightedGraph object when called. The vertices attribute is assigned the number of nodes that the constructor is passed as a parameter. The edges is initialized as 0 because the number of edges is iterated forward every time the addEdge() function is called. A new array list object that contains array lists is also initialized to contain array lists that represent the adjacency list of each node. A for loop is used to add a new array list for each node.

addEdge()

This function is passed two integers, node1 and node2, as well as one float, weight, that are read from the text file in main. It then calls the get() function on adjList to get the current adjacency list for that the first node. A new Edge object is created using the parameters passed into it from main() and added to this adjacency list.

printAdjList()

This function is responsible for representing the graph as an adjacency list for each node. The function first prints a statement containing the number of edges and vertices. Next, it iterates through the list of adjacency lists for each node by iterating from 0 to the number of vertices in the graph, which corresponds to the number of lists contained within the adjacency list. A nested for loop then iterates through each neighbor contained in the current adjacency list. The function then appends the neighbor to the string builder object. The getWeight() function is also called on each edge within the nested for loop to determine their weight, which is also appended to the string builder object. Once the nested for loop exits, a newline is appended to the string. After the outer for loop exits, the function prints the string.

findMST()

The findMST() function implements Prim's algorithm to find the minimum search tree (MST) of the constructed graph. First a new WeightedGraph object, mst, is initialized, as well as a Boolean array used to determine which vertices have been visited, and a PriorityQueue object that will prioritize the edges with the lowest weights. The MST is started from the first node in the graph at index 0. The node is marked as visited using the Boolean array and then a for loop iterates through all edges that are adjacent to the first node, adding them to the priority queue. A while loop continues until the maximum number of edges in an MST have been added to the graph (number of vertices – 1) and the priority queue is empty. While the loop continues, the edge with the lowest weight is removed from the quest and the attributes are retrieved using the getters in the Edge class. If one of the nodes connected by the edge is marked as visited and the other is not, an edge is added between them using the retrieved attributes in the new mst object and the unvisited node is marked as visited, as it is now included in the MST. A for loop then iterates through the adjacent edges, checking if the endpoints of those edges are visited, adding them to the priority queue if not.

Edge()

This constructor initializes the Edge objects when called. The Edge objects contain three attributes: the first node to be connected (u), the second node to be connected (v), and the weight of the edge (weight). There are also a series of getters for these attributes contained within the Edge class. This class also implements the Comparable interface with a function that overrides compareTo() that allows the float values of the weights of the edges to be compared in Prim's algorithm.

main()

This driver function is responsible for reading the text file and calling the functions necessary to build the weighted graph, find the minimum search tree, and printing the results. First the file name is initialized as a string, the number of lines as a long, the weighted graph as a WeightedGraph object, as well as a BufferedReader object and a line counter as an integer to properly iterate through the text file. Inside a while loop that continues if there is a next line for the buffered reader to read, a line counter variable is iterated forward to keep track of what line the reader is currently on, then a series of if statements determine the next operation. If the current line is the first line, this means that the reader has read the number of vertices, so an integer is parsed from the line and the setVertices() function is called to correctly set the graph's number of vertices. If the reader has read the second line, then the function simply continues, because this line contains the number of edges, and the addEdge() function is responsible for setting the number of edges in the graph. If the current line being read is not the first or second line, then it must represent an edge. So, the line is formatted to remove extra spaces using a regex and split at the space between each number into an array of strings. The integers and floats are parsed out of these strings and assigned to the first node, second node, and weight of the represented edge. The addEdge() function is called on these parameters to add the edge to the weighted graph. This continues until all lines of the text file have been read and all edges have been added to the graph. After the while loop exits and the reader is closed, the printAdjList() function is called to print the adjacency list of the newly created weighted graph. After this, the findMST() function is called to find the minimum search tree of the graph using Prim's algorithm. The system time is recorded in nanoseconds before and after the execution of

this function, and the difference is calculated and printed in different units of time for testing purposes. The adjacency list of the MST is also printed using printAdjList().

Code Screenshots:

```java
You, 1 second ago | 3 authors (lcmiles and others)
import java.util.ArrayList;      lmiles1511, 2 days ago • Added functionality to build weighted graph and …
import java.util.List;
import java.util.PriorityQueue;

You, 1 second ago | 3 authors (lcmiles and others)
public class WeightedGraph {

    private int vertices; //number of vertices in the graph
    private int edges; //number
    private List<List<Edge>> adjList;

    /*
    Description: This is the constructor for the graph object
    Parameters:
    int nodes - The number of nodes the graph will contain
    Returns: Nothing
    Sources:
    https://chat.openai.com/share/8b4c2e60-b4e3-4b2a-909b-4a3300ec4287
    https://www.youtube.com/watch?v=X1LdtRW88c0
    https://stackoverflow.com/questions/44831436/java-implementing-weighted-graph
    */
    public WeightedGraph(int nodes) {
    this.vertices = nodes; //initializes the number of vertices as the number of nodes
    this.edges = 0; //initializes the number of edges to 0
    adjList = new ArrayList<>(); //initilizes ArrayList for adjacency lists for each node
    for (int i = 0; i < nodes; i++) { //for each node within the graph a new adjacency list is created
        adjList.add(new ArrayList<>()); //adds the list for each node to adjList
    }
    }

    /*
    Description: This function adds undirected edges to the graph represented as a 1 in the adjacency matrix both ways
    Parameters:
    int node1 - A node read from the text file to be connected to node2 via an edge
    int node2 - A node read from the text file to be connected to node1 via an edge
    Returns: Nothing
    Sources:
    https://chat.openai.com/share/8b4c2e60-b4e3-4b2a-909b-4a3300ec4287
    https://www.youtube.com/watch?v=X1LdtRW88c0
    https://stackoverflow.com/questions/44831436/java-implementing-weighted-graph
    */
    public void addEdge(int u, int v, float weight) {
    adjList.get(u).add(new Edge(u,v, weight)); //gets the adjacency list for node1 and adds a new node object to that list
    edges++; //counts edges
    }

    public List<Edge> getAdjacentEdges(int u) { //getter for the adjList of a certain node
    return adjList.get(u);
    }

    public int getVertices() { //getter for vertices
    return this.vertices;
    }

    public void setVertices(int vertices) { //setter for vertices
        this.vertices = vertices;
    }

```

Figure 1: WeightedGraph.java

```java

    /*
    Description: This function iterates through the list of adjacency lists for each node alond with the weight of each ed
    Parameters: None
    Returns: Nothing
    Sources:
    https://www.youtube.com/watch?v=X1LdtRW88c0
    */
    public void printAdjList() {
        StringBuilder string = new StringBuilder(); //initialize StringBuilder object
        string.append("The graph contains "+ edges + " edges and " + vertices + " vertices. \n");
        for (int i = 0; i < vertices; i++) { //iterates through the list of adjacency lists using using a for loop stoppir
            string.append("Adjacency list for node " + (i + 1) + ": ");
            for (Edge neighbor : adjList.get(i)) { //for each neighbor contained in the nested adjacency list
                string.append(neighbor.getV() + " ");
                string.append("(Weight: " + neighbor.getWeight() + ") ");
            }
            string.append(str:"\n");
        }
        System.out.println(string);
    }
    /*
    Description: This function implements Prim's aglorithm, finding the MST of the graph by repeatedly selecting the edge
    Parameters: None
    Returns:
    WeightedGaph mst - A WeightedGraph object that represents the found MST
    Sources:
    https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/
    https://chat.openai.com/share/c6ed91be-1c2b-4dbf-b7a9-749de512e504
    */
    public WeightedGraph findMST() {
        WeightedGraph mst = new WeightedGraph(vertices); //create a new graph for the MST
        boolean[] visited = new boolean[vertices]; //initialize visited flag
        PriorityQueue<Edge> priorityQueue = new PriorityQueue<>(); //initialize priority queue

        visited[0] = true; //set first vertex to visited because it will always be visited
        for (Edge edge : adjList.get(index:0)) { //iterate through edges adjacent to first vertex
            priorityQueue.offer(edge); //add each to the priority queue; the priority queue prioritizes edges with the low
        }

        while (mst.edges < vertices - 1 && !priorityQueue.isEmpty()) { //continue until the the maximum number of edges ir
            Edge minEdge = priorityQueue.poll(); //remove the edge with the lowest weight from the queue
            int u = minEdge.getU(); //get the edges first connnected node        lcmiles, yesterday • Implemented prim's a
            int v = minEdge.getV(); //get the edges second connected node
            float weight = minEdge.getWeight(); //get the edges weight

            if (visited[u] && !visited[v]) { //if one of the nodes connected by the edge is marked as visited and the othe
                mst.addEdge(u, v, weight); //add an edge between them in the MST
                visited[v] = true; //mark the v vertex visited because it is now included in the MST

                for (Edge neighbor : adjList.get(v)) { //iterate through the edges adjacent to the vertex v
                    if (!visited[neighbor.getV()]) { //if the other endpoint of the edge is not visited
                        priorityQueue.offer(neighbor); //add it to the priority queue to consider it in the next iteratior
                    }
                }
            }
        }
        return mst;
    }

}
```

Figure 2: WeightedGraph.java

```
lcmiles, yesterday | 2 authors (lmiles1511 and others)
class Edge implements Comparable<Edge> {      lcmiles, yesterday • Implemented prim's algorithm
  private int u;
  private int v;
  private float weight;

  /*
    Description: This is the constructor for the edge objects
    Parameters:
    int u - The first node to be connected
    int v - The second node to be connected
    float weight - The weight of the edge
    Returns: Nothing
    Sources:
    https://stackoverflow.com/questions/44831436/java-implementing-weighted-graph
    */
  public Edge(int u, int v, float weight)  {
    this.u = u;
    this.v = v;
    this.weight = weight;
  }

  public int getU() {
    return u;
  }

  public int getV() {
    return v;
  }

  public float getWeight() {
    return weight;
  }

  @Override
    public int compareTo(Edge other) { //override the compareTo class to allow edge weights to be compared
        return Float.compare(this.weight, other.weight);  //compare edges based on their weights
    }
}

```

Figure 3: Edge.java

```java
lcmiles, yesterday | 2 authors (lmiles1511 and others)
import java.io.BufferedReader;      lmiles1511, 2 days ago • Added functionality to build weighted graph and …
import java.io.FileReader;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

lcmiles, yesterday | 2 authors (lmiles1511 and others)
public class HW7 {
    /*
    Description: Executable function that is responsible for reading the text file and calling the functions in the Graph c
    Parameters:
    String[] args - Runtime arguments
    Returns: Nothing
    Sources:
    https://stackoverflow.com/questions/26448352/counting-the-number-of-lines-in-a-text-file-java
    https://www.youtube.com/watch?v=X1LdtRW88c0
    https://chat.openai.com/share/037460fb-c57f-412a-8cd0-43e7559d09d2
    */
    Run | Debug
    public static void main(String[] args) throws Exception {
        String filename = "tinyDG.txt"; //initialize filename string variable
        try {
            long numberOfLines = Files.lines(Paths.get(filename)).count(); //count the number of lines in the text file
            WeightedGraph graph = new WeightedGraph((int)numberOfLines); //initialize graph object with the number of lines
            BufferedReader reader = new BufferedReader(new FileReader(filename)); //initialize reader object
            String line;
            int lineCount = 0; //initialize line count
            while ((line = reader.readLine()) != null) { //iterate through the text file with a while loop
                lineCount++; //iterate the line count variable forward 1 with each loop
                if (lineCount == 1) { //case for first line
                    graph.setVertices(Integer.parseInt(line)); //parse the int value representing the number of vertices an
                }
                else if (lineCount == 2) { //case for second line
                    continue; //skip because addEdge() adds edges
                }
                else {
                    line = line.replaceAll(regex:"\\s+", replacement:" ").trim(); //remove extra spaces around and between
                    String[] parts = line.split(regex:" "); //split each line at the space using a regex
                    int node1 = Integer.parseInt(parts[0]); //parse the int value and assign it to node1
                    int node2 = Integer.parseInt(parts[1]); //parse the int value and assign it to node2
                    float weight = Float.parseFloat(parts[2]); //parse the float value and assign it to weight
                    graph.addEdge(node1, node2, weight); //call addEdge() on both nodes and weight to add the weighted edge
                }
            }
            reader.close();
            graph.printAdjList(); //call printAdjList() to print the adjacency list representation of the graph
            long timeInit = System.nanoTime(); //records initial system time in nanoseconds
            WeightedGraph mst = graph.findMST(); //call prim's algorithm on graph and assign it to a new WeightedGraph obje
            long timeFinal = System.nanoTime(); // records final system time in nanoseconds
            long time = timeFinal - timeInit; //calculates time taken for BFS algorithm
            System.out.println(x:"The MST found using Prim's algoirthm:");
            mst.printAdjList(); //print adjacency list of MST graph
            System.out.println("Prim's Algorithm Time: " + time + " nanoseconds, " + (float)time/1000000 + " milliseconds, 
            System.out.println();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

Figure 4: HW7.java

3. Testing

Testing involved executing the code with on the following text files: “tinyDG,” “mediumDG.txt,” “largeDG.txt,” and “XtralargerDG.txt.” The execution times and resulting graphs of the functions were recorded.

Testing Screenshots:

```
The graph contains 15 edges and 8 vertices.
Adjacency list for node 1: 4 (Weight: 0.38) 2 (Weight: 0.26)
Adjacency list for node 2: 3 (Weight: 0.29)
Adjacency list for node 3: 7 (Weight: 0.34)
Adjacency list for node 4: 6 (Weight: 0.52)
Adjacency list for node 5: 5 (Weight: 0.35) 7 (Weight: 0.37)
Adjacency list for node 6: 4 (Weight: 0.35) 7 (Weight: 0.28) 1 (Weight: 0.32)
Adjacency list for node 7: 2 (Weight: 0.4) 0 (Weight: 0.58) 4 (Weight: 0.93)
Adjacency list for node 8: 5 (Weight: 0.28) 3 (Weight: 0.39)
```

Figure 5: Resulting weighted graph from tinyDG.txt

```
The MST found using Prim's algoirthm:
The graph contains 7 edges and 8 vertices.
Adjacency list for node 1: 2 (Weight: 0.26)
Adjacency list for node 2: 3 (Weight: 0.29)
Adjacency list for node 3: 7 (Weight: 0.34)
Adjacency list for node 4: 6 (Weight: 0.52)
Adjacency list for node 5:
Adjacency list for node 6: 1 (Weight: 0.32) 4 (Weight: 0.35)
Adjacency list for node 7:
Adjacency list for node 8: 5 (Weight: 0.28)

Prim's Algorithm Time: 297900 nanoseconds, 0.2979 milliseconds, or 2.979E-4 seconds
```

Figure 6: Resulting MST from tinyDG.txt

```
The graph contains 2546 edges and 250 vertices.
Adjacency list for node 1: 15 (Weight: 0.05719) 24 (Weight: 0.10191) 44 (Weight: 0.06471) 49 (Weight: 0.04849) 58 (Weight: 0.09955) 59 (Weigh
t: 0.10657) 68 (Weight: 0.11816) 80 (Weight: 0.06821) 97 (Weight: 0.07705) 114 (Weight: 0.0961) 149 (Weight: 0.09659) 160 (Weight: 0.11714) 1
63 (Weight: 0.09368) 176 (Weight: 0.08927) 191 (Weight: 0.10711) 202 (Weight: 0.04678) 204 (Weight: 0.05476) 209 (Weight: 0.09511) 211 (Weigh
t: 0.08438) 222 (Weight: 0.07573) 225 (Weight: 0.02383)
Adjacency list for node 2: 72 (Weight: 0.06506) 107 (Weight: 0.07484) 130 (Weight: 0.10203) 150 (Weight: 0.10908) 164 (Weight: 0.11039) 189 (
Weight: 0.09582) 194 (Weight: 0.11069) 200 (Weight: 0.0955) 203 (Weight: 0.08567) 220 (Weight: 0.10428)
Adjacency list for node 3: 14 (Weight: 0.08765) 18 (Weight: 0.07425) 42 (Weight: 0.11456) 51 (Weight: 0.05083) 79 (Weight: 0.11759) 86 (Weigh
t: 0.0598) 108 (Weight: 0.09627) 110 (Weight: 0.11746) 141 (Weight: 0.11373)
Adjacency list for node 4: 37 (Weight: 0.08512) 45 (Weight: 0.11902) 67 (Weight: 0.09725) 76 (Weight: 0.08069) 115 (Weight: 0.09861) 153 (Wei
ght: 0.04799) 228 (Weight: 0.07635) 241 (Weight: 0.07024)
Adjacency list for node 5: 5 (Weight: 0.11184) 26 (Weight: 0.08347) 55 (Weight: 0.06425) 77 (Weight: 0.10733) 78 (Weight: 0.02559) 112 (Weigh
t: 0.08751) 128 (Weight: 0.04751) 138 (Weight: 0.11375) 159 (Weight: 0.10114) 239 (Weight: 0.03883) 240 (Weight: 0.11344)
Adjacency list for node 6: 26 (Weight: 0.03351) 32 (Weight: 0.11054) 55 (Weight: 0.11131) 67 (Weight: 0.1088) 77 (Weight: 0.05505) 102 (Weigh
t: 0.03834) 104 (Weight: 0.11574) 217 (Weight: 0.09458) 226 (Weight: 0.11433) 4 (Weight: 0.11184)
Adjacency list for node 7: 16 (Weight: 0.04529) 54 (Weight: 0.11235) 98 (Weight: 0.09893) 99 (Weight: 0.11022) 117 (Weight: 0.08821) 129 (Wei
ght: 0.05363) 140 (Weight: 0.10829) 147 (Weight: 0.07924) 166 (Weight: 0.06998) 178 (Weight: 0.07007) 236 (Weight: 0.05556)
Adjacency list for node 8: 42 (Weight: 0.11616) 57 (Weight: 0.06795) 65 (Weight: 0.09235) 71 (Weight: 0.11091) 101 (Weight: 0.10577) 125 (Wei
ght: 0.02442) 148 (Weight: 0.02175) 157 (Weight: 0.00516) 181 (Weight: 0.05778) 184 (Weight: 0.04976) 188 (Weight: 0.10982) 197 (Weight: 0.06
984) 230 (Weight: 0.06107)
Adjacency list for node 9: 11 (Weight: 0.04709) 30 (Weight: 0.03985) 43 (Weight: 0.09334) 82 (Weight: 0.07286) 85 (Weight: 0.11331) 143 (Weig
ht: 0.07437) 152 (Weight: 0.00702) 179 (Weight: 0.09533) 207 (Weight: 0.09011) 210 (Weight: 0.10661) 212 (Weight: 0.05604) 221 (Weight: 0.118
95) 244 (Weight: 0.02711) 246 (Weight: 0.09709)
Adjacency list for node 10: 23 (Weight: 0.03526) 33 (Weight: 0.08216) 58 (Weight: 0.10398) 68 (Weight: 0.09604) 114 (Weight: 0.11445) 142 (We
ight: 0.10955) 195 (Weight: 0.04585)
Adjacency list for node 11: 105 (Weight: 0.11028) 106 (Weight: 0.11976) 123 (Weight: 0.00886) 175 (Weight: 0.07429) 246 (Weight: 0.09977)
Adjacency list for node 12: 30 (Weight: 0.08689) 43 (Weight: 0.10208) 82 (Weight: 0.03687) 85 (Weight: 0.06928) 143 (Weight: 0.08708) 152 (We
ight: 0.0414) 175 (Weight: 0.09935) 207 (Weight: 0.11101) 212 (Weight: 0.09716) 244 (Weight: 0.07397) 246 (Weight: 0.06678) 8 (Weight: 0.0470
9)
Adjacency list for node 13: 28 (Weight: 0.06032) 35 (Weight: 0.06079) 36 (Weight: 0.08058) 41 (Weight: 0.06364) 88 (Weight: 0.07461) 94 (Weig
ht: 0.08239) 113 (Weight: 0.09906) 121 (Weight: 0.08542) 170 (Weight: 0.11918) 182 (Weight: 0.06361) 198 (Weight: 0.05807) 242 (Weight: 0.084
57)
Adjacency list for node 14: 19 (Weight: 0.08927) 100 (Weight: 0.0256) 103 (Weight: 0.08741) 129 (Weight: 0.10843) 133 (Weight: 0.06257) 162 (
Weight: 0.11602) 174 (Weight: 0.09377) 192 (Weight: 0.08128)
Adjacency list for node 15: 18 (Weight: 0.07335) 51 (Weight: 0.09603) 86 (Weight: 0.09145) 129 (Weight: 0.10737) 133 (Weight: 0.06649) 166 (W
eight: 0.08096) 2 (Weight: 0.08765)
Adjacency list for node 16: 24 (Weight: 0.04507) 39 (Weight: 0.09051) 49 (Weight: 0.10519) 58 (Weight: 0.10462) 66 (Weight: 0.11797) 80 (Weig
ht: 0.08217) 114 (Weight: 0.09104) 149 (Weight: 0.04256) 163 (Weight: 0.05589) 202 (Weight: 0.08786) 204 (Weight: 0.11074) 209 (Weight: 0.045
8) 211 (Weight: 0.04) 222 (Weight: 0.07726) 225 (Weight: 0.07498) 0 (Weight: 0.05719)
Adjacency list for node 17: 54 (Weight: 0.07406) 98 (Weight: 0.11623) 99 (Weight: 0.1021) 117 (Weight: 0.05134) 129 (Weight: 0.09325) 140 (We
ight: 0.07952) 147 (Weight: 0.07171) 166 (Weight: 0.11475) 178 (Weight: 0.06981) 236 (Weight: 0.07388) 6 (Weight: 0.04529)
Adjacency list for node 18: 41 (Weight: 0.10519) 81 (Weight: 0.05763) 121 (Weight: 0.09728) 134 (Weight: 0.10171) 158 (Weight: 0.10542) 170 (
Weight: 0.07756) 182 (Weight: 0.09423) 223 (Weight: 0.11337) 229 (Weight: 0.06676)
Adjacency list for node 19: 35 (Weight: 0.11609) 51 (Weight: 0.11645) 86 (Weight: 0.02813) 94 (Weight: 0.11772) 141 (Weight: 0.06466) 14 (Wei
ght: 0.07335) 2 (Weight: 0.07425)
Adjacency list for node 20: 70 (Weight: 0.06872) 79 (Weight: 0.08364) 84 (Weight: 0.0354) 100 (Weight: 0.06397) 103 (Weight: 0.05319) 174 (We
ight: 0.01925) 179 (Weight: 0.08733) 192 (Weight: 0.09085) 243 (Weight: 0.09323) 13 (Weight: 0.08927)
Adjacency list for node 21: 40 (Weight: 0.07068) 75 (Weight: 0.02897) 89 (Weight: 0.0569) 116 (Weight: 0.02095) 127 (Weight: 0.11837) 164 (We
ight: 0.06088) 190 (Weight: 0.09529) 194 (Weight: 0.09349) 220 (Weight: 0.10072) 247 (Weight: 0.07725)
Adjacency list for node 22: 27 (Weight: 0.01873) 62 (Weight: 0.07424) 65 (Weight: 0.07811) 71 (Weight: 0.03782) 138 (Weight: 0.09594) 184 (We
ight: 0.11666) 188 (Weight: 0.0428) 230 (Weight: 0.08276) 233 (Weight: 0.04409) 240 (Weight: 0.08207)
Adjacency list for node 23: 34 (Weight: 0.03658) 53 (Weight: 0.07033) 56 (Weight: 0.01508) 73 (Weight: 0.04871) 120 (Weight: 0.06464) 145 (We
ight: 0.09324)
Adjacency list for node 24: 33 (Weight: 0.06032) 58 (Weight: 0.07072) 68 (Weight: 0.07437) 114 (Weight: 0.0796) 176 (Weight: 0.11709) 195 (We
ight: 0.0809) 222 (Weight: 0.09692) 9 (Weight: 0.03526)
Adjacency list for node 25: 39 (Weight: 0.07951) 66 (Weight: 0.08457) 80 (Weight: 0.11814) 114 (Weight: 0.10397) 149 (Weight: 0.02915) 163 (W
eight: 0.05034) 206 (Weight: 0.09602) 209 (Weight: 0.02802) 211 (Weight: 0.03696) 222 (Weight: 0.09835) 225 (Weight: 0.11996) 15 (Weight: 0.0
4507) 0 (Weight: 0.10191)
Adjacency list for node 26: 60 (Weight: 0.03405) 63 (Weight: 0.09366) 96 (Weight: 0.08879) 111 (Weight: 0.05309) 199 (Weight: 0.07779)
Adjacency list for node 27: 55 (Weight: 0.09933) 77 (Weight: 0.03854) 78 (Weight: 0.09413) 102 (Weight: 0.0419) 138 (Weight: 0.09654) 217 (We
ight: 0.1191) 226 (Weight: 0.11644) 239 (Weight: 0.11461) 240 (Weight: 0.11097) 5 (Weight: 0.03351) 4 (Weight: 0.08347)
Adjacency list for node 28: 62 (Weight: 0.07419) 65 (Weight: 0.07207) 71 (Weight: 0.05107) 138 (Weight: 0.07814) 184 (Weight: 0.11544) 188 (W
eight: 0.03989) 230 (Weight: 0.08817) 233 (Weight: 0.04883) 240 (Weight: 0.06581) 21 (Weight: 0.01873)
Adjacency list for node 29: 35 (Weight: 0.10487) 41 (Weight: 0.10902) 94 (Weight: 0.08477) 113 (Weight: 0.05483) 121 (Weight: 0.07801) 170 (W
eight: 0.10987) 182 (Weight: 0.08584) 198 (Weight: 0.00775) 223 (Weight: 0.09677) 242 (Weight: 0.02503) 12 (Weight: 0.06032)
Adjacency list for node 30: 47 (Weight: 0.02921) 64 (Weight: 0.10752) 91 (Weight: 0.06429) 109 (Weight: 0.0944) 137 (Weight: 0.06803) 146 (We
ight: 0.08191) 167 (Weight: 0.10821) 218 (Weight: 0.02524) 224 (Weight: 0.03477) 227 (Weight: 0.10208)
Adjacency list for node 31: 43 (Weight: 0.10126) 70 (Weight: 0.10835) 79 (Weight: 0.10818) 82 (Weight: 0.10926) 143 (Weight: 0.0874) 152 (Wei
ght: 0.04565) 156 (Weight: 0.11364) 179 (Weight: 0.0657) 207 (Weight: 0.08708) 210 (Weight: 0.09124) 212 (Weight: 0.03314) 214 (Weight: 0.112
8) 219 (Weight: 0.11146) 221 (Weight: 0.11496) 244 (Weight: 0.01475) 11 (Weight: 0.08689) 8 (Weight: 0.03985)
Adjacency list for node 32: 37 (Weight: 0.09487) 115 (Weight: 0.0873) 153 (Weight: 0.10769) 228 (Weight: 0.07913) 241 (Weight: 0.07871)
Adjacency list for node 33: 52 (Weight: 0.07988) 77 (Weight: 0.11466) 93 (Weight: 0.0741) 102 (Weight: 0.09251) 104 (Weight: 0.10076) 144 (We
ight: 0.07333) 151 (Weight: 0.11408) 160 (Weight: 0.11627) 168 (Weight: 0.05434) 185 (Weight: 0.10165) 187 (Weight: 0.04594) 201 (Weight: 0.1
1801) 208 (Weight: 0.11121) 226 (Weight: 0.07421) 231 (Weight: 0.057) 248 (Weight: 0.07367) 5 (Weight: 0.11054)
Adjacency list for node 34: 58 (Weight: 0.09102) 114 (Weight: 0.08593) 163 (Weight: 0.10678) 222 (Weight: 0.10644) 23 (Weight: 0.06032) 9 (We
ight: 0.08216)
Adjacency list for node 35: 53 (Weight: 0.08565) 56 (Weight: 0.04399) 73 (Weight: 0.01812) 120 (Weight: 0.09075) 145 (Weight: 0.07784) 22 (We
ight: 0.03658)
Adjacency list for node 36: 36 (Weight: 0.08804) 41 (Weight: 0.10606) 88 (Weight: 0.07142) 94 (Weight: 0.06254) 141 (Weight: 0.11388) 198 (We
ight: 0.10645) 28 (Weight: 0.10487) 18 (Weight: 0.11609) 12 (Weight: 0.06079)
Adjacency list for node 37: 41 (Weight: 0.05329) 88 (Weight: 0.01841) 98 (Weight: 0.08515) 182 (Weight: 0.09055) 35 (Weight: 0.08804) 12 (Wei
ght: 0.08058)
Adjacency list for node 38: 76 (Weight: 0.02674) 95 (Weight: 0.09352) 115 (Weight: 0.01398) 153 (Weight: 0.03846) 228 (Weight: 0.07596) 241 (
Weight: 0.06139) 31 (Weight: 0.09487) 3 (Weight: 0.08512)
Adjacency list for node 39: 74 (Weight: 0.06907) 109 (Weight: 0.09245) 126 (Weight: 0.00845) 183 (Weight: 0.08279) 215 (Weight: 0.0529)
Adjacency list for node 40: 66 (Weight: 0.0559) 80 (Weight: 0.10277) 149 (Weight: 0.05382) 206 (Weight: 0.10412) 209 (Weight: 0.10654) 211 (W
eight: 0.11209) 24 (Weight: 0.07951) 15 (Weight: 0.09051)
Adjacency list for node 41: 75 (Weight: 0.07166) 89 (Weight: 0.10329) 116 (Weight: 0.051) 150 (Weight: 0.07051) 164 (Weight: 0.03395) 190 (We
ight: 0.03649) 194 (Weight: 0.10624) 220 (Weight: 0.0315) 247 (Weight: 0.07741) 20 (Weight: 0.07068)
Adjacency list for node 42: 81 (Weight: 0.11854) 88 (Weight: 0.06374) 121 (Weight: 0.07816) 170 (Weight: 0.09973) 182 (Weight: 0.03878) 198 (
Weight: 0.10323) 36 (Weight: 0.05329) 35 (Weight: 0.10606) 28 (Weight: 0.10902) 17 (Weight: 0.10519) 12 (Weight: 0.06364)
Adjacency list for node 43: 86 (Weight: 0.09639) 101 (Weight: 0.09688) 108 (Weight: 0.06664) 135 (Weight: 0.05008) 141 (Weight: 0.0888) 157 (
Weight: 0.11467) 181 (Weight: 0.06506) 196 (Weight: 0.09113) 7 (Weight: 0.11616) 2 (Weight: 0.11456)
Adjacency list for node 44: 82 (Weight: 0.08576) 152 (Weight: 0.08933) 156 (Weight: 0.10122) 207 (Weight: 0.02708) 210 (Weight: 0.05763) 212
(Weight: 0.07554) 219 (Weight: 0.09405) 221 (Weight: 0.03765) 244 (Weight: 0.10274) 30 (Weight: 0.10126) 11 (Weight: 0.10208) 8 (Weight: 0.09
334)
Adjacency list for node 45: 49 (Weight: 0.02107) 59 (Weight: 0.09573) 68 (Weight: 0.1193) 80 (Weight: 0.10281) 93 (Weight: 0.06793) 97 (Weigh
t: 0.03365) 144 (Weight: 0.09765) 160 (Weight: 0.06268) 168 (Weight: 0.10433) 176 (Weight: 0.07613) 185 (Weight: 0.10945) 191 (Weight: 0.0824
) 202 (Weight: 0.05971) 204 (Weight: 0.01774) 222 (Weight: 0.1137) 225 (Weight: 0.05336) 231 (Weight: 0.10384) 248 (Weight: 0.11288) 0 (Weigh
t: 0.06471)
Adjacency list for node 46: 48 (Weight: 0.11127) 67 (Weight: 0.06225) 76 (Weight: 0.11037) 83 (Weight: 0.02906) 95 (Weight: 0.10135) 104 (Wei
ght: 0.11417) 217 (Weight: 0.04535) 232 (Weight: 0.09323) 3 (Weight: 0.11902)
Adjacency list for node 47: 161 (Weight: 0.07892) 169 (Weight: 0.06854) 177 (Weight: 0.11118) 186 (Weight: 0.07689)
Adjacency list for node 48: 64 (Weight: 0.10444) 91 (Weight: 0.06658) 109 (Weight: 0.07505) 137 (Weight: 0.06569) 146 (Weight: 0.09071) 167 (
```

Figure 7: Resulting weighted graph from mediumDG.txt

```
The graph contains 249 edges and 250 vertices.
Adjacency list for node 1: 225 (Weight: 0.02383)
Adjacency list for node 2: 107 (Weight: 0.07484)
Adjacency list for node 3: 51 (Weight: 0.05083)
Adjacency list for node 4: 153 (Weight: 0.04799)
Adjacency list for node 5: 78 (Weight: 0.02559)
Adjacency list for node 6: 26 (Weight: 0.03351)
Adjacency list for node 7: 16 (Weight: 0.04529) 236 (Weight: 0.05556)
Adjacency list for node 8: 125 (Weight: 0.02442)
Adjacency list for node 9: 152 (Weight: 0.00702) 143 (Weight: 0.07437)
Adjacency list for node 10: 195 (Weight: 0.04585)
Adjacency list for node 11: 123 (Weight: 0.00886)
Adjacency list for node 12: 82 (Weight: 0.03687) 246 (Weight: 0.06678)
Adjacency list for node 13: 198 (Weight: 0.05807)
Adjacency list for node 14: 133 (Weight: 0.06257)
Adjacency list for node 15:
Adjacency list for node 16:
Adjacency list for node 17: 117 (Weight: 0.05134) 147 (Weight: 0.07171)
Adjacency list for node 18: 81 (Weight: 0.05763) 229 (Weight: 0.06676)
Adjacency list for node 19:
Adjacency list for node 20: 174 (Weight: 0.01925) 100 (Weight: 0.06397)
Adjacency list for node 21:
Adjacency list for node 22: 71 (Weight: 0.03782) 233 (Weight: 0.04409)
Adjacency list for node 23: 34 (Weight: 0.03658)
Adjacency list for node 24: 9 (Weight: 0.03526) 33 (Weight: 0.06032)
Adjacency list for node 25: 149 (Weight: 0.02915)
Adjacency list for node 26: 60 (Weight: 0.03405)
Adjacency list for node 27:
Adjacency list for node 28: 21 (Weight: 0.01873)
Adjacency list for node 29: 242 (Weight: 0.02503)
Adjacency list for node 30: 224 (Weight: 0.03477)
Adjacency list for node 31: 244 (Weight: 0.01475) 179 (Weight: 0.0657)
Adjacency list for node 32:
Adjacency list for node 33:
Adjacency list for node 34:
Adjacency list for node 35: 73 (Weight: 0.01812)
Adjacency list for node 36: 12 (Weight: 0.06079)
Adjacency list for node 37: 88 (Weight: 0.01841)
Adjacency list for node 38: 115 (Weight: 0.01398) 76 (Weight: 0.02674)
Adjacency list for node 39: 126 (Weight: 0.00845)
Adjacency list for node 40: 66 (Weight: 0.0559)
Adjacency list for node 41: 164 (Weight: 0.03395) 190 (Weight: 0.03649)
Adjacency list for node 42: 36 (Weight: 0.05329)
Adjacency list for node 43: 135 (Weight: 0.05008) 108 (Weight: 0.06664)
Adjacency list for node 44:
Adjacency list for node 45: 204 (Weight: 0.01774)
Adjacency list for node 46:
Adjacency list for node 47:
Adjacency list for node 48:
Adjacency list for node 49: 216 (Weight: 0.07531)
Adjacency list for node 50: 44 (Weight: 0.02107) 97 (Weight: 0.03121)
Adjacency list for node 51: 185 (Weight: 0.05942)
Adjacency list for node 52:
Adjacency list for node 53: 226 (Weight: 0.02384) 208 (Weight: 0.03135)
Adjacency list for node 54: 120 (Weight: 0.02782)
Adjacency list for node 55:
Adjacency list for node 56: 4 (Weight: 0.06425)
Adjacency list for node 57: 22 (Weight: 0.01508)
Adjacency list for node 58: 197 (Weight: 0.02583)
Adjacency list for node 59: 114 (Weight: 0.01947) 23 (Weight: 0.07072)
Adjacency list for node 60: 50 (Weight: 0.04226)
Adjacency list for node 61: 199 (Weight: 0.04868)
Adjacency list for node 62: 111 (Weight: 0.06739)
Adjacency list for node 63:
Adjacency list for node 64: 237 (Weight: 0.04962)
Adjacency list for node 65:
Adjacency list for node 66: 184 (Weight: 0.0479)
Adjacency list for node 67: 206 (Weight: 0.05154)
Adjacency list for node 68: 3 (Weight: 0.09725)
Adjacency list for node 69: 58 (Weight: 0.04795)
Adjacency list for node 70: 173 (Weight: 0.05282)
Adjacency list for node 71: 79 (Weight: 0.01576) 19 (Weight: 0.06872)
Adjacency list for node 72: 188 (Weight: 0.03894)
Adjacency list for node 73: 189 (Weight: 0.03706) 1 (Weight: 0.06506)
Adjacency list for node 74:
Adjacency list for node 75:
Adjacency list for node 76: 89 (Weight: 0.03394)
Adjacency list for node 77: 95 (Weight: 0.0737)
Adjacency list for node 78: 138 (Weight: 0.07171)
Adjacency list for node 79: 239 (Weight: 0.02065)
Adjacency list for node 80:
Adjacency list for node 81:
Adjacency list for node 82: 134 (Weight: 0.04508)
Adjacency list for node 83: 85 (Weight: 0.04344)
Adjacency list for node 84: 45 (Weight: 0.02906) 217 (Weight: 0.03695)
Adjacency list for node 85:
Adjacency list for node 86:
Adjacency list for node 87: 18 (Weight: 0.02813) 2 (Weight: 0.0598)
Adjacency list for node 88: 61 (Weight: 0.05163) 234 (Weight: 0.08302)
Adjacency list for node 89:
Adjacency list for node 90: 194 (Weight: 0.0511) 127 (Weight: 0.10682)
Adjacency list for node 91:
Adjacency list for node 92: 137 (Weight: 0.01061) 218 (Weight: 0.04088)
Adjacency list for node 93:
Adjacency list for node 94: 231 (Weight: 0.03594)
Adjacency list for node 95: 35 (Weight: 0.06254)
Adjacency list for node 96:
Adjacency list for node 97:
Adjacency list for node 98: 59 (Weight: 0.06442)
Adjacency list for node 99:
Adjacency list for node 100: 162 (Weight: 0.06455)
Adjacency list for node 101: 13 (Weight: 0.0256)
Adjacency list for node 102: 139 (Weight: 0.03983)
Adjacency list for node 103: 77 (Weight: 0.02737) 5 (Weight: 0.03834)
```

Figure 8: Resulting MST from mediumDG.txt

```
Adjacency list for node 783: 794 (Weight: 0.05188) 848 (Weight: 0.0478) 951 (Weight: 0.03947) 750
(Weight: 0.05254) 685 (Weight: 0.06611) 639 (Weight: 0.07158) 638 (Weight: 0.03244) 582 (Weight: 0
.05527) 530 (Weight: 0.05517) 496 (Weight: 0.03913) 458 (Weight: 0.05622) 375 (Weight: 0.07177) 32
5 (Weight: 0.03781) 298 (Weight: 0.04162) 240 (Weight: 0.04579) 159 (Weight: 0.05736) 116 (Weight:
 0.03676) 110 (Weight: 0.03969) 80 (Weight: 0.04742) 11 (Weight: 0.04616)
Adjacency list for node 784: 807 (Weight: 0.01378) 972 (Weight: 0.03186) 764 (Weight: 0.03675) 748
 (Weight: 0.02999) 722 (Weight: 0.06669) 645 (Weight: 0.04081) 635 (Weight: 0.04224) 584 (Weight:
0.0572) 538 (Weight: 0.03279) 516 (Weight: 0.01357) 512 (Weight: 0.03699) 428 (Weight: 0.07352) 25
6 (Weight: 0.02896) 100 (Weight: 0.03421) 91 (Weight: 0.04531) 44 (Weight: 0.05542) 27 (Weight: 0.
0478) 14 (Weight: 0.06664)
Adjacency list for node 785: 808 (Weight: 0.07481) 850 (Weight: 0.05672) 906 (Weight: 0.04794) 919
 (Weight: 0.06186) 949 (Weight: 0.02072) 954 (Weight: 0.07207) 776 (Weight: 0.05087) 672 (Weight:
0.05306) 643 (Weight: 0.02997) 610 (Weight: 0.04416) 577 (Weight: 0.06166) 573 (Weight: 0.06219) 5
11 (Weight: 0.06186) 488 (Weight: 0.04688) 487 (Weight: 0.03548) 457 (Weight: 0.0557) 446 (Weight:
 0.06721) 419 (Weight: 0.05954) 368 (Weight: 0.06342) 346 (Weight: 0.07469) 253 (Weight: 0.07069)
219 (Weight: 0.04959) 154 (Weight: 0.04402) 143 (Weight: 0.02824) 75 (Weight: 0.06388)
Adjacency list for node 786: 820 (Weight: 0.02927) 931 (Weight: 0.0523) 958 (Weight: 0.05687) 710
(Weight: 0.07443) 576 (Weight: 0.02258) 558 (Weight: 0.03501) 387 (Weight: 0.07436) 358 (Weight: 0
.01403) 337 (Weight: 0.01707) 335 (Weight: 0.05458) 284 (Weight: 0.01383) 274 (Weight: 0.05876) 21
7 (Weight: 0.06359) 119 (Weight: 0.01607) 8 (Weight: 0.0126) 0 (Weight: 0.0514)
Adjacency list for node 787: 887 (Weight: 0.01898) 892 (Weight: 0.03585) 898 (Weight: 0.05483) 917
 (Weight: 0.07486) 964 (Weight: 0.04885) 977 (Weight: 0.07171) 777 (Weight: 0.05991) 724 (Weight:
0.03859) 636 (Weight: 0.04163) 603 (Weight: 0.039) 600 (Weight: 0.04947) 537 (Weight: 0.05947) 520
 (Weight: 0.01388) 515 (Weight: 0.03166) 415 (Weight: 0.04937) 382 (Weight: 0.01762) 333 (Weight:
0.07065) 258 (Weight: 0.06082) 238 (Weight: 0.07141) 88 (Weight: 0.06243) 49 (Weight: 0.04555)
Adjacency list for node 788: 870 (Weight: 0.04439) 751 (Weight: 0.05605) 721 (Weight: 0.04371) 659
 (Weight: 0.02895) 620 (Weight: 0.03132) 615 (Weight: 0.06496) 526 (Weight: 0.06386) 469 (Weight:
0.05891) 346 (Weight: 0.06376) 167 (Weight: 0.03456) 106 (Weight: 0.03751) 101 (Weight: 0.05308)
Adjacency list for node 789: 803 (Weight: 0.02928) 872 (Weight: 0.06988) 969 (Weight: 0.07373) 970
 (Weight: 0.03216) 780 (Weight: 0.03882) 759 (Weight: 0.05976) 573 (Weight: 0.04307) 488 (Weight:
0.05926) 430 (Weight: 0.0425) 344 (Weight: 0.05169) 310 (Weight: 0.06201) 219 (Weight: 0.06937) 21
4 (Weight: 0.04991) 158 (Weight: 0.02125) 131 (Weight: 0.04732)
Adjacency list for node 790: 836 (Weight: 0.06684) 839 (Weight: 0.01003) 899 (Weight: 0.06032) 973
 (Weight: 0.0423) 984 (Weight: 0.02287) 986 (Weight: 0.04317) 995 (Weight: 0.06611) 775 (Weight: 0
.02137) 702 (Weight: 0.0325) 641 (Weight: 0.06839) 606 (Weight: 0.02935) 525 (Weight: 0.06574) 465
 (Weight: 0.03655) 414 (Weight: 0.01457) 197 (Weight: 0.04569) 157 (Weight: 0.06723) 124 (Weight:
0.0371) 102 (Weight: 0.06254)
Adjacency list for node 791: 845 (Weight: 0.05458) 861 (Weight: 0.03531) 915 (Weight: 0.0642) 974
(Weight: 0.04779) 773 (Weight: 0.05886) 656 (Weight: 0.03902) 477 (Weight: 0.03731) 315 (Weight: 0
.03737) 297 (Weight: 0.0611) 262 (Weight: 0.06235) 12 (Weight: 0.05374) 10 (Weight: 0.04476)
Adjacency list for node 792: 867 (Weight: 0.0469) 734 (Weight: 0.04852) 698 (Weight: 0.03252) 676
(Weight: 0.06615) 666 (Weight: 0.07078) 596 (Weight: 0.00908) 528 (Weight: 0.04844) 371 (Weight: 0
.06611) 292 (Weight: 0.038) 255 (Weight: 0.04066) 251 (Weight: 0.06538) 248 (Weight: 0.0207) 207 (
Weight: 0.01797) 47 (Weight: 0.01836) 32 (Weight: 0.04436) 25 (Weight: 0.05242)
Adjacency list for node 793: 848 (Weight: 0.04852) 893 (Weight: 0.02626) 750 (Weight: 0.04353) 743
 (Weight: 0.03379) 699 (Weight: 0.01228) 685 (Weight: 0.04882) 644 (Weight: 0.05038) 622 (Weight:
0.06476) 458 (Weight: 0.06601) 417 (Weight: 0.06345) 311 (Weight: 0.06037) 298 (Weight: 0.0604) 24
0 (Weight: 0.06861) 200 (Weight: 0.06939) 186 (Weight: 0.05949) 139 (Weight: 0.03871)
Adjacency list for node 794: 832 (Weight: 0.05167) 866 (Weight: 0.04735) 975 (Weight: 0.03493) 978
 (Weight: 0.04567) 763 (Weight: 0.05711) 746 (Weight: 0.01093) 697 (Weight: 0.02597) 686 (Weight:
0.06712) 665 (Weight: 0.06693) 369 (Weight: 0.0702) 336 (Weight: 0.04535) 189 (Weight: 0.03307) 12
5 (Weight: 0.01735) 113 (Weight: 0.0601) 92 (Weight: 0.01955) 60 (Weight: 0.06844)
Adjacency list for node 795: 848 (Weight: 0.07418) 937 (Weight: 0.06707) 782 (Weight: 0.05188) 750
 (Weight: 0.05747) 639 (Weight: 0.0197) 638 (Weight: 0.06591) 559 (Weight: 0.05138) 530 (Weight: 0
.02456) 281 (Weight: 0.03436) 221 (Weight: 0.07024) 156 (Weight: 0.06982) 116 (Weight: 0.01729) 80
 (Weight: 0.02883) 11 (Weight: 0.06305)
Adjacency list for node 796: 916 (Weight: 0.06779) 950 (Weight: 0.05019) 977 (Weight: 0.05807) 992
 (Weight: 0.07146) 777 (Weight: 0.0561) 744 (Weight: 0.05478) 708 (Weight: 0.05222) 684 (Weight: 0
.06598) 551 (Weight: 0.06711) 515 (Weight: 0.07375) 476 (Weight: 0.03253) 438 (Weight: 0.05382) 41
5 (Weight: 0.06378) 403 (Weight: 0.02925) 235 (Weight: 0.0643) 138 (Weight: 0.0364) 88 (Weight: 0.
04814) 70 (Weight: 0.06128) 39 (Weight: 0.05766)
Adjacency list for node 797: 834 (Weight: 0.05698) 695 (Weight: 0.06794) 631 (Weight: 0.03511) 625
 (Weight: 0.06463) 519 (Weight: 0.04376) 439 (Weight: 0.07112) 436 (Weight: 0.06309) 367 (Weight:
0.03915) 326 (Weight: 0.06678) 317 (Weight: 0.0573) 270 (Weight: 0.05364)
Adjacency list for node 798: 802 (Weight: 0.0476) 830 (Weight: 0.05217) 874 (Weight: 0.06422) 719
(Weight: 0.06246) 650 (Weight: 0.02281) 612 (Weight: 0.06077) 598 (Weight: 0.0562) 581 (Weight: 0.
04121) 579 (Weight: 0.06533) 562 (Weight: 0.06775) 552 (Weight: 0.02835) 549 (Weight: 0.03949) 474
 (Weight: 0.07316) 454 (Weight: 0.03836) 272 (Weight: 0.02648) 257 (Weight: 0.06038) 145 (Weight:
0.0638) 118 (Weight: 0.06982) 40 (Weight: 0.07007)
Adjacency list for node 799: 818 (Weight: 0.05168) 859 (Weight: 0.03802) 871 (Weight: 0.00824) 747
 (Weight: 0.06255) 731 (Weight: 0.03302) 726 (Weight: 0.04783) 703 (Weight: 0.05474) 587 (Weight:
0.04194) 548 (Weight: 0.04648) 510 (Weight: 0.06792) 463 (Weight: 0.05152) 389 (Weight: 0.05047) 2
45 (Weight: 0.07468) 208 (Weight: 0.03644) 185 (Weight: 0.02001) 149 (Weight: 0.0623) 45 (Weight:
0.06181) 7 (Weight: 0.02963) 1 (Weight: 0.06835)
Adjacency list for node 800: 809 (Weight: 0.03319) 825 (Weight: 0.04149) 864 (Weight: 0.05685) 905
 (Weight: 0.03718) 939 (Weight: 0.0357) 778 (Weight: 0.01555) 762 (Weight: 0.03893) 727 (Weight: 0
.05152) 707 (Weight: 0.06115) 602 (Weight: 0.06654) 562 (Weight: 0.07054) 471 (Weight: 0.07175) 41
2 (Weight: 0.03376) 399 (Weight: 0.04714) 353 (Weight: 0.06477) 323 (Weight: 0.05328) 287 (Weight:
 0.05946) 230 (Weight: 0.06345) 81 (Weight: 0.04361) 3 (Weight: 0.03799)
Adjacency list for node 801: 827 (Weight: 0.04595) 854 (Weight: 0.01693) 924 (Weight: 0.05634) 763
 (Weight: 0.06138) 742 (Weight: 0.06813) 379 (Weight: 0.06551) 374 (Weight: 0.0292) 372 (Weight: 0
.05967) 347 (Weight: 0.05478) 305 (Weight: 0.05595) 233 (Weight: 0.06562) 173 (Weight: 0.07455) 34
 (Weight: 0.04956)
Adjacency list for node 802: 860 (Weight: 0.03185) 890 (Weight: 0.0647) 907 (Weight: 0.07395) 932
(Weight: 0.05301) 934 (Weight: 0.03616) 973 (Weight: 0.05818) 757 (Weight: 0.0648) 752 (Weight: 0.
06002) 736 (Weight: 0.06715) 678 (Weight: 0.03326) 619 (Weight: 0.06202) 534 (Weight: 0.06152) 525
 (Weight: 0.06107) 499 (Weight: 0.02796) 424 (Weight: 0.06047) 411 (Weight: 0.05528) 409 (Weight:
0.02004) 342 (Weight: 0.03511) 313 (Weight: 0.0524) 279 (Weight: 0.02133) 261 (Weight: 0.00893) 23
6 (Weight: 0.03155) 226 (Weight: 0.07037) 174 (Weight: 0.00584) 153 (Weight: 0.02465) 105 (Weight:
 0.0711) 69 (Weight: 0.05292) 19 (Weight: 0.03842)
Adjacency list for node 803: 983 (Weight: 0.04903) 797 (Weight: 0.0476) 781 (Weight: 0.04819) 768
(Weight: 0.065) 719 (Weight: 0.0607) 705 (Weight: 0.06161) 691 (Weight: 0.04746) 650 (Weight: 0.04
142) 642 (Weight: 0.04732) 612 (Weight: 0.06767) 588 (Weight: 0.06602) 581 (Weight: 0.06735) 552 (
Weight: 0.06735) 549 (Weight: 0.0257) 474 (Weight: 0.05928) 454 (Weight: 0.07373) 380 (Weight: 0.0
6594) 316 (Weight: 0.06708) 272 (Weight: 0.06302) 118 (Weight: 0.02339) 40 (Weight: 0.05712) 20 (W
eight: 0.04155)
Adjacency list for node 804: 881 (Weight: 0.0701) 969 (Weight: 0.04701) 970 (Weight: 0.04201) 788
(Weight: 0.02928) 780 (Weight: 0.06547) 677 (Weight: 0.0643) 627 (Weight: 0.06111) 573 (Weight: 0.
06048) 495 (Weight: 0.06459) 430 (Weight: 0.06814) 344 (Weight: 0.03242) 310 (Weight: 0.04578) 214
 (Weight: 0.03965) 158 (Weight: 0.03607) 133 (Weight: 0.06562) 131 (Weight: 0.01847)
Adjacency list for node 805: 872 (Weight: 0.0477) 948 (Weight: 0.05106) 987 (Weight: 0.06532) 546
(Weight: 0.04743) 490 (Weight: 0.05713) 481 (Weight: 0.04279) 214 (Weight: 0.07304) 56 (Weight: 0.
04127) 53 (Weight: 0.06768)
Adjacency list for node 806: 814 (Weight: 0.04254) 756 (Weight: 0.06751) 714 (Weight: 0.06019) 546
```

Figure 9: Resulting weighted graph from largeDG.txt

```
The MST found using Prim's algoirthm:
The graph contains 999 edges and 1000 vertices.
Adjacency list for node 1: 958 (Weight: 0.01083) 558 (Weight: 0.01791)
Adjacency list for node 2:
Adjacency list for node 3: 475 (Weight: 0.00978) 602 (Weight: 0.02599)
Adjacency list for node 4: 905 (Weight: 0.02536)
Adjacency list for node 5: 879 (Weight: 0.02345)
Adjacency list for node 6: 956 (Weight: 0.00858) 340 (Weight: 0.01895)
Adjacency list for node 7:
Adjacency list for node 8: 185 (Weight: 0.0106) 389 (Weight: 0.02442)
Adjacency list for node 9: 284 (Weight: 0.00905)
Adjacency list for node 10: 891 (Weight: 0.0205)
Adjacency list for node 11: 656 (Weight: 0.01064)
Adjacency list for node 12:
Adjacency list for node 13: 297 (Weight: 0.0082)
Adjacency list for node 14:
Adjacency list for node 15: 722 (Weight: 0.0262)
Adjacency list for node 16: 565 (Weight: 0.01496) 928 (Weight: 0.01865)
Adjacency list for node 17:
Adjacency list for node 18: 630 (Weight: 0.01398) 273 (Weight: 0.01546)
Adjacency list for node 19: 539 (Weight: 0.01009) 242 (Weight: 0.03682)
Adjacency list for node 20: 313 (Weight: 0.01896) 424 (Weight: 0.02284)
Adjacency list for node 21: 118 (Weight: 0.01873)
Adjacency list for node 22: 644 (Weight: 0.02602) 616 (Weight: 0.03327)
Adjacency list for node 23:
Adjacency list for node 24: 105 (Weight: 0.0188)
Adjacency list for node 25:
Adjacency list for node 26:
Adjacency list for node 27: 688 (Weight: 0.00285)
Adjacency list for node 28: 512 (Weight: 0.01323)
Adjacency list for node 29:
Adjacency list for node 30: 182 (Weight: 0.01675)
Adjacency list for node 31: 589 (Weight: 0.02029)
Adjacency list for node 32: 706 (Weight: 0.01357)
Adjacency list for node 33: 25 (Weight: 0.00981)
Adjacency list for node 34: 462 (Weight: 0.02049)
Adjacency list for node 35: 305 (Weight: 0.01028)
Adjacency list for node 36:
Adjacency list for node 37: 772 (Weight: 0.039)
Adjacency list for node 38: 107 (Weight: 0.02497)
Adjacency list for node 39: 554 (Weight: 0.03256)
Adjacency list for node 40:
Adjacency list for node 41:
Adjacency list for node 42: 137 (Weight: 0.00173)
Adjacency list for node 43:
Adjacency list for node 44: 86 (Weight: 0.0141)
Adjacency list for node 45: 38 (Weight: 0.03717)
Adjacency list for node 46: 813 (Weight: 0.0165)
Adjacency list for node 47: 525 (Weight: 0.03412)
Adjacency list for node 48: 207 (Weight: 0.01773) 32 (Weight: 0.02629)
Adjacency list for node 49: 357 (Weight: 0.02157)
Adjacency list for node 50: 964 (Weight: 0.00337) 537 (Weight: 0.02937)
Adjacency list for node 51:
Adjacency list for node 52:
Adjacency list for node 53: 321 (Weight: 0.02587)
Adjacency list for node 54: 386 (Weight: 0.01189) 362 (Weight: 0.01536)
Adjacency list for node 55: 293 (Weight: 0.01749) 926 (Weight: 0.01833)
Adjacency list for node 56: 838 (Weight: 0.0096)
Adjacency list for node 57: 53 (Weight: 0.02878) 804 (Weight: 0.04127)
Adjacency list for node 58: 902 (Weight: 0.01478)
Adjacency list for node 59: 679 (Weight: 0.01397) 129 (Weight: 0.02704)
Adjacency list for node 60: 5 (Weight: 0.00784) 381 (Weight: 0.01556)
Adjacency list for node 61: 336 (Weight: 0.02511)
Adjacency list for node 62:
Adjacency list for node 63: 17 (Weight: 0.02185)
Adjacency list for node 64:
Adjacency list for node 65:
Adjacency list for node 66: 472 (Weight: 0.01524) 397 (Weight: 0.02014)
Adjacency list for node 67: 142 (Weight: 0.02254)
Adjacency list for node 68:
Adjacency list for node 69: 350 (Weight: 0.00706)
Adjacency list for node 70: 934 (Weight: 0.01728) 932 (Weight: 0.02484)
Adjacency list for node 71: 263 (Weight: 0.02307)
Adjacency list for node 72: 366 (Weight: 0.00555)
Adjacency list for node 73: 247 (Weight: 0.03583)
Adjacency list for node 74: 895 (Weight: 0.02766)
Adjacency list for node 75: 586 (Weight: 0.02605)
Adjacency list for node 76: 929 (Weight: 0.01865)
Adjacency list for node 77: 199 (Weight: 0.00774)
Adjacency list for node 78: 816 (Weight: 0.02268) 6 (Weight: 0.02295)
Adjacency list for node 79: 111 (Weight: 0.04636)
Adjacency list for node 80: 406 (Weight: 0.01827) 133 (Weight: 0.02782)
Adjacency list for node 81:
Adjacency list for node 82: 323 (Weight: 0.01702)
Adjacency list for node 83:
Adjacency list for node 84: 166 (Weight: 0.02625)
Adjacency list for node 85: 239 (Weight: 0.01019)
Adjacency list for node 86: 339 (Weight: 0.02462)
Adjacency list for node 87: 383 (Weight: 0.05014)
Adjacency list for node 88: 99 (Weight: 0.0211) 668 (Weight: 0.02919) 821 (Weight: 0.03399)
Adjacency list for node 89: 777 (Weight: 0.00938)
Adjacency list for node 90: 491 (Weight: 0.01243)
Adjacency list for node 91: 74 (Weight: 0.01226) 326 (Weight: 0.02726) 270 (Weight: 0.03088)
Adjacency list for node 92: 428 (Weight: 0.03028)
Adjacency list for node 93: 975 (Weight: 0.01584)
Adjacency list for node 94: 566 (Weight: 0.02496)
Adjacency list for node 95: 509 (Weight: 0.01437) 123 (Weight: 0.03587)
Adjacency list for node 96: 588 (Weight: 0.01276)
Adjacency list for node 97: 283 (Weight: 0.00978)
Adjacency list for node 98:
Adjacency list for node 99:
Adjacency list for node 100: 175 (Weight: 0.02563)
Adjacency list for node 101: 256 (Weight: 0.02651)
```

Figure 10: Resulting MST from largeDG.txt

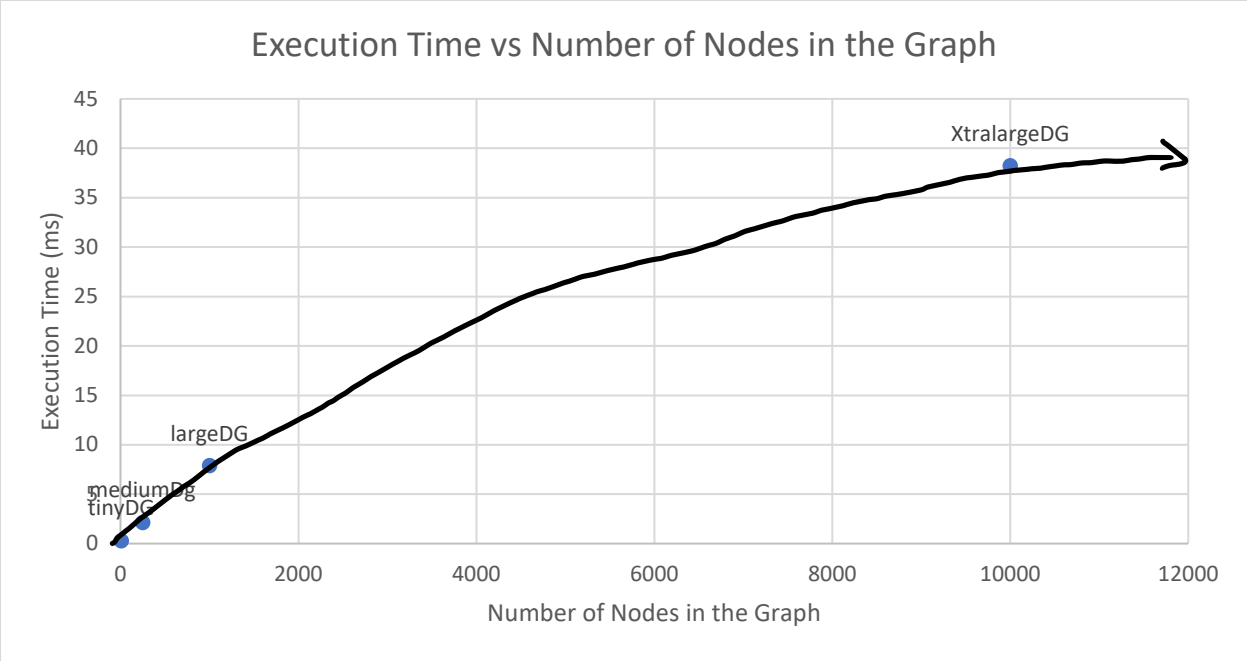
8684 (Weight: 0.01901) 8382 (Weight: 0.01018) 8178 (Weight: 0.01963) 7449 (Weight: 0.01669) 6932 (Weight: 0.00869) 6616 (Weight: 0.01027) 5313 (Weight: 0.01931) 4433 (Weight: 0.01774) 3382 (Weight: 0.00809) 2892 (Weight: 0.01359) 1826 (Weight: 0.01964)
Adjacency list for node 9728: 9028 (Weight: 0.01667) 8791 (Weight: 0.01873) 8361 (Weight: 0.00615) 7179 (Weight: 0.01884) 6607 (Weight: 0.0145) 6024 (Weight: 0.01892) 5485 (Weight: 0.01533) 5453 (Weight: 0.01939) 4274 (Weight: 0.01975) 2680 (Weight: 0.0093) 2383 (Weight: 0.01771) 2037 (Weight: 0.01329) 1942 (Weight: 0.01486)
Adjacency list for node 9729: 8789 (Weight: 0.01472) 8090 (Weight: 0.01589) 8054 (Weight: 0.01562) 7836 (Weight: 0.01666) 7172 (Weight: 0.01059) 6674 (Weight: 0.01477) 6495 (Weight: 0.01635) 5945 (Weight: 0.01863) 5507 (Weight: 0.01981) 5461 (Weight: 0.01859) 3881 (Weight: 0.01445) 3793 (Weight: 0.00368) 3251 (Weight: 0.01049) 2715 (Weight: 0.01718) 670 (Weight: 0.01836) 226 (Weight: 0.01093) 212 (Weight: 0.00602)
Adjacency list for node 9730: 9042 (Weight: 0.01921) 8941 (Weight: 0.01494) 6558 (Weight: 0.00904) 6201 (Weight: 0.01556) 6160 (Weight: 0.01186) 3105 (Weight: 0.01612) 2594 (Weight: 0.01012) 2072 (Weight: 0.01958) 1782 (Weight: 0.01001) 1780 (Weight: 0.01789) 1751 (Weight: 0.01287) 42 (Weight: 0.00756)
Adjacency list for node 9731: 9112 (Weight: 0.01276) 8593 (Weight: 0.01924) 8569 (Weight: 0.01474) 8030 (Weight: 0.008) 7415 (Weight: 0.00528) 6516 (Weight: 0.01202) 5254 (Weight: 0.00642) 4566 (Weight: 0.01429) 4001 (Weight: 0.01916) 3110 (Weight: 0.01019) 298 (Weight: 0.01714) 174 (Weight: 0.01925) 110 (Weight: 0.0043)
Adjacency list for node 9732: 9413 (Weight: 0.01927) 7288 (Weight: 0.00498) 5990 (Weight: 0.00984) 5739 (Weight: 0.01912) 5330 (Weight: 0.01919) 4459 (Weight: 0.00663) 2329 (Weight: 0.00535) 2047 (Weight: 0.01683)
Adjacency list for node 9733: 7790 (Weight: 0.01976) 6275 (Weight: 0.01326) 2476 (Weight: 0.01313) 1821 (Weight: 0.01367) 1551 (Weight: 0.01279) 1238 (Weight: 0.01023) 718 (Weight: 0.00587)
Adjacency list for node 9734: 8232 (Weight: 0.01639) 8197 (Weight: 0.01177) 8177 (Weight: 0.00212) 7528 (Weight: 0.01315) 7054 (Weight: 0.01545) 6402 (Weight: 0.01429) 6281 (Weight: 0.01846) 5653 (Weight: 0.01439) 5034 (Weight: 0.0126) 4831 (Weight: 0.01013) 4776 (Weight: 0.01401) 4728 (Weight: 0.01721) 1432 (Weight: 0.01966) 1387 (Weight: 0.0183)
Adjacency list for node 9735: 8616 (Weight: 0.01996) 8308 (Weight: 0.00617) 8306 (Weight: 0.00154) 7104 (Weight: 0.01977) 6870 (Weight: 0.0118) 6048 (Weight: 0.0159) 4430 (Weight: 0.0096) 4225 (Weight: 0.01686) 3816 (Weight: 0.0128) 3298 (Weight: 0.01793) 1771 (Weight: 0.00727)
Adjacency list for node 9736: 8605 (Weight: 0.01186) 7597 (Weight: 0.01473) 6659 (Weight: 0.00943) 6412 (Weight: 0.00782) 4819 (Weight: 0.00269) 4400 (Weight: 0.01926) 2741 (Weight: 0.00758) 2099 (Weight: 0.00262) 2049 (Weight: 0.01312) 221 (Weight: 0.01319)
Adjacency list for node 9737: 8020 (Weight: 0.01968) 7416 (Weight: 0.01954) 7271 (Weight: 0.01563) 5359 (Weight: 0.01091) 4797 (Weight: 0.01631) 4584 (Weight: 0.01207) 3308 (Weight: 0.01278) 1558 (Weight: 0.01806) 1237 (Weight: 0.00545)
Adjacency list for node 9738: 9040 (Weight: 0.00932) 8006 (Weight: 0.01344) 7013 (Weight: 0.00822) 6444 (Weight: 0.00431) 5266 (Weight: 0.01552) 3978 (Weight: 0.00912) 3656 (Weight: 0.01699) 1753 (Weight: 0.00587) 527 (Weight: 0.00591)
Adjacency list for node 9739: 9862 (Weight: 0.00358) 5501 (Weight: 0.01721) 5302 (Weight: 0.0175) 3813 (Weight: 0.00725) 354 (Weight: 0.01964)
Adjacency list for node 9740: 6291 (Weight: 0.00867) 6172 (Weight: 0.01155) 5460 (Weight: 0.01055) 4258 (Weight: 0.0168) 4248 (Weight: 0.01888) 4172 (Weight: 0.00879) 2966 (Weight: 0.00165) 2440 (Weight: 0.01404) 2292 (Weight: 0.01293) 2248 (Weight: 0.01558) 1849 (Weight: 0.01363) 1506 (Weight: 0.01325) 1483 (Weight: 0.01922) 371 (Weight: 0.01986)
Adjacency list for node 9741: 9417 (Weight: 0.0114) 9287 (Weight: 0.01369) 9081 (Weight: 0.01425) 8985 (Weight: 0.01712) 8752 (Weight: 0.01657) 8471 (Weight: 0.00477) 6504 (Weight: 0.01635) 4722 (Weight: 0.01618) 3487 (Weight: 0.01654) 2423 (Weight: 0.01602) 2094 (Weight: 0.01692) 1161 (Weight: 0.0159)
Adjacency list for node 9742: 8651 (Weight: 0.01318) 8545 (Weight: 0.01785) 6752 (Weight: 0.00538) 5707 (Weight: 0.01998) 5433 (Weight: 0.01223) 3840 (Weight: 0.01737) 3444 (Weight: 0.01749) 2564 (Weight: 0.01946) 1936 (Weight: 0.01188) 1644 (Weight: 0.01619) 797 (Weight: 0.01382) 361 (Weight: 0.00686)
Adjacency list for node 9743: 9837 (Weight: 0.00882) 9061 (Weight: 0.00955) 3161 (Weight: 0.01032) 1637 (Weight: 0.01887)
Adjacency list for node 9744: 9786 (Weight: 0.01126) 8096 (Weight: 0.0082) 7807 (Weight: 0.00818) 7044 (Weight: 0.01151) 6543 (Weight: 0.01557) 5978 (Weight: 0.01249) 1371 (Weight: 0.01071) 668 (Weight: 0.00813) 514 (Weight: 0.0134) 321 (Weight: 0.01376)
Adjacency list for node 9745: 9482 (Weight: 0.01567) 6665 (Weight: 0.01984) 6645 (Weight: 0.01753) 6624 (Weight: 0.01671) 5261 (Weight: 0.01633) 5253 (Weight: 0.01826) 5196 (Weight: 0.01899) 1741 (Weight: 0.01991) 117 (Weight: 0.01237)
Adjacency list for node 9746: 9533 (Weight: 0.00399) 8863 (Weight: 0.00946) 8652 (Weight: 0.00936) 7155 (Weight: 0.0156) 7068 (Weight: 0.01873) 7011 (Weight: 0.01943) 5666 (Weight: 0.00994) 4460 (Weight: 0.01509) 1075 (Weight: 0.01383) 304 (Weight: 0.00807)
Adjacency list for node 9747: 9560 (Weight: 0.01441) 9439 (Weight: 0.0146) 7583 (Weight: 0.00381) 7326 (Weight: 0.01306) 6070 (Weight: 0.00582) 6037 (Weight: 0.01598) 5992 (Weight: 0.00833) 5331 (Weight: 0.01679) 5197 (Weight: 0.00695) 2860 (Weight: 0.01836) 1925 (Weight: 0.01619) 1833 (Weight: 0.01878) 395 (Weight: 0.01875) 70 (Weight: 0.0176)
Adjacency list for node 9748: 9318 (Weight: 0.01883) 8679 (Weight: 0.01792) 8582 (Weight: 0.01185) 8481 (Weight: 0.01384) 8311 (Weight: 0.01287) 8085 (Weight: 0.01998) 7505 (Weight: 0.00774) 7138 (Weight: 0.01998) 7047 (Weight: 0.01463) 6073 (Weight: 0.00949) 5886 (Weight: 0.00962) 5869 (Weight: 0.01696) 5547 (Weight: 0.0199) 5168 (Weight: 0.00465) 5083 (Weight: 0.00813) 3480 (Weight: 0.0064) 1466 (Weight: 0.00929) 673 (Weight: 0.01793) 591 (Weight: 0.01768) 587 (Weight: 0.00861)
Adjacency list for node 9749: 9670 (Weight: 0.0154) 9649 (Weight: 0.01882) 9437 (Weight: 0.01854) 9234 (Weight: 0.01186) 9055 (Weight: 0.01504) 8895 (Weight: 0.01247) 8815 (Weight: 0.00837) 7736 (Weight: 0.01706) 7562 (Weight: 0.01171) 7241 (Weight: 0.01192) 6794 (Weight: 0.01919) 5400 (Weight: 0.01938) 5349 (Weight: 0.0166) 5232 (Weight: 0.00729) 4908 (Weight: 0.01668) 4882 (Weight: 0.01015) 4818 (Weight: 0.01381) 4109 (Weight: 0.01047) 3603 (Weight: 0.00767) 2311 (Weight: 0.01591) 2280 (Weight: 0.01687)
Adjacency list for node 9750: 9626 (Weight: 0.00755) 9445 (Weight: 0.01334) 8801 (Weight: 0.00131) 8700 (Weight: 0.01731) 7926 (Weight: 0.00854) 7305 (Weight: 0.00445) 6885 (Weight: 0.01506) 6792 (Weight: 0.01613) 6072 (Weight: 0.00306) 4536 (Weight: 0.00812) 3691 (Weight: 0.01116) 3335 (Weight: 0.0195) 2078 (Weight: 0.01769) 1403 (Weight: 0.01367) 943 (Weight: 0.01295) 688 (Weight: 0.0109) 384 (Weight: 0.01168)
Adjacency list for node 9751: 9571 (Weight: 0.01154) 9235 (Weight: 0.01882) 9125 (Weight: 0.01289) 9099 (Weight: 0.01979) 8145 (Weight: 0.01409) 7619 (Weight: 0.01412) 6880 (Weight: 0.00627) 6796 (Weight: 0.01683) 5721 (Weight: 0.00775) 4133 (Weight: 0.01885) 145 (Weight: 0.01755)
Adjacency list for node 9752: 9859 (Weight: 0.00761) 9630 (Weight: 0.01243) 8771 (Weight: 0.01851) 8389 (Weight: 0.00906) 7894 (Weight: 0.00629) 7015 (Weight: 0.01257) 4892 (Weight: 0.00794) 4294 (Weight: 0.01601) 4269 (Weight: 0.01368) 2822 (Weight: 0.014) 756 (Weight: 0.01172)
Adjacency list for node 9753: 8810 (Weight: 0.00415) 4766 (Weight: 0.01957) 3743 (Weight: 0.01708) 3207 (Weight: 0.01526) 3039 (Weight: 0.01376) 2985 (Weight: 0.01143) 2842 (Weight: 0.01621) 2192 (Weight: 0.00298) 1606 (Weight: 0.01867) 1386 (Weight: 0.00856) 953 (Weight: 0.01908) 575 (Weight: 0.01912) 539 (Weight: 0.00928) 503 (Weight: 0.009)
Adjacency list for node 9754: 9942 (Weight: 0.01926) 9604 (Weight: 0.01616) 9294 (Weight: 0.01549) 8845 (Weight: 0.0167) 8501 (Weight: 0.01052) 8114 (Weight: 0.01795) 7991 (Weight: 0.01638) 7471 (Weight: 0.0176) 7264 (Weight: 0.01641) 5741 (Weight: 0.00276) 5392 (Weight: 0.00938) 1126 (Weight: 0.01812)
Adjacency list for node 9755: 9541 (Weight: 0.01643) 8876 (Weight: 0.01653) 8449 (Weight: 0.00969) 8310 (Weight: 0.0125) 7022 (Weight: 0.00688) 6961 (Weight: 0.01179) 5557 (Weight: 0.01566) 5213 (Weight: 0.00632) 4912 (Weight: 0.01302) 4260 (Weight: 0.0196) 3693 (Weight: 0.0144) 2322 (Weight: 0.01113) 2230 (Weight: 0.01916) 1735 (Weight: 0.0159) 1611 (Weight: 0.01533) 1327 (Weight: 0.01705

Figure 11: Resulting weighted graph from XtralargeDG.txt

```
Adjacency list for node 9899: 1277 (Weight: 0.00726) 4649 (Weight: 0.00732)
Adjacency list for node 9900: 3801 (Weight: 0.00338)
Adjacency list for node 9901:
Adjacency list for node 9902: 9190 (Weight: 0.00537)
Adjacency list for node 9903: 4399 (Weight: 0.00683)
Adjacency list for node 9904: 3072 (Weight: 0.00184) 2465 (Weight: 0.00803)
Adjacency list for node 9905: 3747 (Weight: 0.00635)
Adjacency list for node 9906: 8039 (Weight: 0.00573)
Adjacency list for node 9907: 7372 (Weight: 0.01105)
Adjacency list for node 9908: 9225 (Weight: 0.00794)
Adjacency list for node 9909:
Adjacency list for node 9910: 3780 (Weight: 0.00787)
Adjacency list for node 9911: 8246 (Weight: 0.00421)
Adjacency list for node 9912: 1352 (Weight: 0.00381)
Adjacency list for node 9913: 3742 (Weight: 0.00604)
Adjacency list for node 9914: 369 (Weight: 0.00257) 8722 (Weight: 0.00775)
Adjacency list for node 9915: 9015 (Weight: 0.0013)
Adjacency list for node 9916: 9271 (Weight: 0.0058) 6125 (Weight: 0.00626)
Adjacency list for node 9917: 4739 (Weight: 0.00183)
Adjacency list for node 9918: 4706 (Weight: 0.00742)
Adjacency list for node 9919: 8027 (Weight: 0.01133)
Adjacency list for node 9920: 6467 (Weight: 0.00903)
Adjacency list for node 9921: 3210 (Weight: 0.00909) 6208 (Weight: 0.01025)
Adjacency list for node 9922: 59 (Weight: 0.00311)
Adjacency list for node 9923: 4203 (Weight: 0.00412) 8981 (Weight: 0.01133)
Adjacency list for node 9924: 5184 (Weight: 0.00798)
Adjacency list for node 9925: 9698 (Weight: 0.00863)
Adjacency list for node 9926: 4632 (Weight: 0.00587)
Adjacency list for node 9927: 6820 (Weight: 0.00231) 8293 (Weight: 0.00541)
Adjacency list for node 9928: 5787 (Weight: 0.00884)
Adjacency list for node 9929: 8998 (Weight: 0.00658)
Adjacency list for node 9930: 3999 (Weight: 0.00604) 2612 (Weight: 0.00609)
Adjacency list for node 9931: 1106 (Weight: 0.00306)
Adjacency list for node 9932:
Adjacency list for node 9933:
Adjacency list for node 9934: 4265 (Weight: 0.00564)
Adjacency list for node 9935: 8778 (Weight: 0.00837)
Adjacency list for node 9936:
Adjacency list for node 9937: 3837 (Weight: 0.00647)
Adjacency list for node 9938:
Adjacency list for node 9939: 8511 (Weight: 0.00491) 9628 (Weight: 0.00567)
Adjacency list for node 9940:
Adjacency list for node 9941: 1941 (Weight: 0.00161)
Adjacency list for node 9942: 327 (Weight: 9.8E-4)
Adjacency list for node 9943: 1326 (Weight: 0.00316)
Adjacency list for node 9944: 8980 (Weight: 0.01124)
Adjacency list for node 9945: 806 (Weight: 0.00969)
Adjacency list for node 9946: 5568 (Weight: 0.00898)
Adjacency list for node 9947: 8340 (Weight: 0.00796) 1790 (Weight: 0.00803)
Adjacency list for node 9948: 6747 (Weight: 0.00898)
Adjacency list for node 9949: 9197 (Weight: 0.01045)
Adjacency list for node 9950: 708 (Weight: 0.00385)
Adjacency list for node 9951: 1470 (Weight: 0.0072) 6154 (Weight: 0.01174)
Adjacency list for node 9952: 3494 (Weight: 0.00812)
Adjacency list for node 9953:
Adjacency list for node 9954: 8793 (Weight: 0.0049)
Adjacency list for node 9955: 2228 (Weight: 0.00542)
Adjacency list for node 9956: 9888 (Weight: 0.00657) 2625 (Weight: 0.00678)
Adjacency list for node 9957:
Adjacency list for node 9958:
Adjacency list for node 9959: 9513 (Weight: 0.00299)
Adjacency list for node 9960: 8235 (Weight: 0.00362)
Adjacency list for node 9961: 9144 (Weight: 0.00553)
Adjacency list for node 9962: 862 (Weight: 0.00398)
Adjacency list for node 9963:
Adjacency list for node 9964: 3049 (Weight: 0.00912)
Adjacency list for node 9965: 3144 (Weight: 0.00408) 9386 (Weight: 0.00721)
Adjacency list for node 9966: 574 (Weight: 0.00901)
Adjacency list for node 9967: 2149 (Weight: 0.00506) 7770 (Weight: 0.00817)
Adjacency list for node 9968: 2193 (Weight: 0.0016)
Adjacency list for node 9969: 9152 (Weight: 0.00389)
Adjacency list for node 9970: 5670 (Weight: 0.005)
Adjacency list for node 9971: 157 (Weight: 0.00725)
Adjacency list for node 9972: 1953 (Weight: 0.00308)
Adjacency list for node 9973: 7070 (Weight: 0.00491)
Adjacency list for node 9974: 2780 (Weight: 0.01278)
Adjacency list for node 9975: 2746 (Weight: 0.00557) 7569 (Weight: 0.00803)
Adjacency list for node 9976: 3975 (Weight: 0.00345)
Adjacency list for node 9977: 2444 (Weight: 0.0092)
Adjacency list for node 9978: 8355 (Weight: 0.00641)
Adjacency list for node 9979: 6708 (Weight: 0.00365)
Adjacency list for node 9980: 4361 (Weight: 0.00335) 5259 (Weight: 0.00468)
Adjacency list for node 9981:
Adjacency list for node 9982: 4394 (Weight: 0.0072)
Adjacency list for node 9983:
Adjacency list for node 9984: 6947 (Weight: 0.00734)
Adjacency list for node 9985: 5023 (Weight: 0.01219)
Adjacency list for node 9986: 2955 (Weight: 0.00258)
Adjacency list for node 9987:
Adjacency list for node 9988: 2820 (Weight: 0.00497) 2942 (Weight: 0.00539)
Adjacency list for node 9989: 5506 (Weight: 0.01047)
Adjacency list for node 9990:
Adjacency list for node 9991: 8328 (Weight: 0.00242)
Adjacency list for node 9992: 3448 (Weight: 0.00934)
Adjacency list for node 9993: 9929 (Weight: 0.00389) 238 (Weight: 0.00797)
Adjacency list for node 9994: 7875 (Weight: 0.00662)
Adjacency list for node 9995: 6185 (Weight: 0.01069)
Adjacency list for node 9996: 782 (Weight: 0.00291)
Adjacency list for node 9997: 8892 (Weight: 0.00223) 1479 (Weight: 0.00566)
Adjacency list for node 9998: 9898 (Weight: 0.00474)
Adjacency list for node 9999: 8258 (Weight: 0.00235) 1110 (Weight: 0.00312)
Adjacency list for node 10000: 1859 (Weight: 0.0031)

Prim's Algorithm Time: 39251799 nanoseconds, 39.2518 milliseconds, or 0.0392518 seconds
```

Figure 12: Resulting MST from XtralargeDG.txt



The graph above shows the execution time of the findMST() function that implements Prim's algorithm. This algorithm is supposed to have a time complexity of O(E*log(V)) where E is the number of edges and V is the number of vertices, or nodes. This is supported by the above graph as it has a vaguely logarithmic trendline.

4. Sources

https://stackoverflow.com/questions/26448352/counting-the-number-of-lines-in-a-text-file-java

https://www.youtube.com/watch?v=X1LdtRW88c0

https://chat.openai.com/share/037460fb-c57f-412a-8cd0-43e7559d09d2

https://chat.openai.com/share/8b4c2e60-b4e3-4b2a-909b-4a3300ec4287

https://stackoverflow.com/questions/44831436/java-implementing-weighted-graph

https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/

https://chat.openai.com/share/c6ed91be-1c2b-4dbf-b7a9-749de512e504