**Project #5**

**Due Dates: Wednesday, December 7 at 11:59pm**

**Submit: eLearning**

**Late Policy: 24-hour late period, then zero**

**Instructions: This is an individual assignment. All work should be your own.**

**Objective:**

**Work with graphs and Kruskals algorithm for minimum spanning trees.**

**Overview:**

**The pseudocode for Kruskals algorithm is given in the textbook to find a minimum spanning tree of a graph. Your program will find the minimum spanning tree among a set of cities in Texas.**

**Details:**

**Write a command-line program that uses Kruskal's algorithm to find a minimum spanning tree of a graph. The graph will be provided as a file named assn9\_data.csv. The data in the file is in the form of an adjacency list.**

**You must use the author's DisjSets class without modifying it. You can either use one of the author's priority queue classes or you can use the PriorityQueue class provided in Java.**

**You should output each edge of your minimum spanning tree as the names of the two cities and the distance between them. You should also print the sum of all of the distances in the tree.**

**Submit to eLearning:**

**Kruskals.java**

**int NumElements = 128;**

**DisjSets ds = new DisjSets( NumElements );**

**int set1, set2;**

**set1 = ds.find( j );**

**set2 = ds.find( j + k );**

**ds.union( set1, set2 )**

**public void kruskal()**

**{**

**int edgesAccepted = 0;**

**DisjSet ds = new DisjSet(NUM\_VERTICES);**

**PriorityQueue<Edge> pq = new PriorityQueue<Edge>( getEdges() );**

**Edge e; Vertex u, v;**

**while (edgesAccepted < NUM\_VERTICES – 1)**

**{**

**e = pq.deleteMin( ); // get minimum edge = (u,v)**

**SetType uset = ds.find( u ); // find set vertex u is in.**

**SetType vset = ds.find( v ); // find set vertex v is in.**

**if (uset != vset) // if not same set (not yet connected)**

**{**

**// accept the edge**

**edgesAccepted++;**

**ds.union(uset, vset); // connect them**

**}**

**}**

**}**

**ArrayList<Edge> kruskal( List<Edge> edges, int numVertices )**

**{**

**DisjSets ds = new DisjSets( numVertices );**

**PriorityQueue<Edge> pq = new PriorityQueue<>( edges );**

**List<Edge> mst = new ArrayList<>( );**

**while( mst.size( ) != numVertices - 1 )**

**{**

**Edge e = pq.deleteMin( ); // Edge e = (u, v)**

**SetType uset = ds.find( e.getu( ) );**

**SetType vset = ds.find( e.getv( ) );**

**if( uset != vset )**

**{**

**// Accept the edge**

**mst.add( e );**

**ds.union( uset, vset );**

**}**

**}**

**return mst;**

**}**