CS 25100 Programming Assignment 2
Fall 2013 Due: 11:59pm, Thursday, Nov. 14

This assignment concerns finding minimum spanning trees of a graph. As you know, a minimum spanning tree is a minimum-cost way to connect up all the vertices of an undirected graph. It has application in implementing real networks and is used within algorithms that solve more complicated problems.

Many graphs have more than one possible minimum spanning tree. Given more than one minimum spanning tree, you may prefer one such tree over another. Let's investigate the possibility of finding more than one minimum spanning tree, so that you will have a choice as to which such tree is most useful in practice.

The assignment consists of two tasks. In the first task, you must code up a java procedure named *min_st* that, given a weighted undirected graph, will output a minimum spanning tree of that graph. Your code must use Kruskal's minimum spanning tree algorithm and the disjoint set union-find data structure (with weighted unions and path compression). The output must consist of the weight of a minimum spanning tree along with the edges contained in the minimum spanning tree.

In the second task, you must code up a java procedure named *two_msts* that, given a weighted undirected graph, will determine whether the graph has at least two distinct minimum spanning trees, and if so, output each of them. Your procedure for the second task should not examine all (or even most) spanning trees of a graph. One way to guarantee getting a different spanning tree is to identify an edge that is in a m.s.t. and then delete that edge. Of course this new graph may have a m.s.t. of greater cost, so you would have to be careful in pursuing this approach.

Students should write their procedures to handle standard input in the following form, where comments put between /* and */ will not appear in the actual input:

```
n /* number of vertices */
m /* number of edges */
   /* one edge per line in the form: vertex vertex weight */
   /* each vertex is an integer between 1 and n (inclusive) */
   /* each weight is a positive integer */
v1 v2 w1
vi vj wk
   /* etc. for m lines altogether */
   /* start next graph with flag 0 */
0
   /* etc. for each succeeding graph in turn */
   /* end of last graph indicated with a flag 1 */
1
```

Output should be:

/* weight of minimum spanning tree of first graph */
/* list of edges, 1 edge per line, with edges ordered by increasing value of first endpoint, followed by increasing value of second endpoint */
/* flag of 2 if there is a second m.s.t. of that graph */ 2
/* followed by list of edges for the second m.s.t. if there is a second m.s.t. */
/* start solution for the next graph with a flag 0 */
0
/* etc. */

To make it easy for the TAs to grade, list each edge $(v1, v2)$ such that $v1 < v2$, list the edges so that they are ordered by increasing first vertex, and when two edges have the same first vertex, order by increasing second vertex. Thus the edges in the set $\{(4,2),\ (2,1),\ (1,4),\ (1,5)\}$ should be output:
1 2
1 4
1 5
2 4
Also, please do not print anything extra, such as strings like "Enter your input" or "Next graph", etc., or even a blank line.

Note: The entry class must be named MSTKruskal. There must be a separate class for disjoint union-find data structure called UnionFind. Students are free to add any other helping classes. All the project files should be contained in the folder named lab2-src.

To turn in the project follow these instructions:

1. Login to lore.cs.purdue.edu
2. Remove the .class files
3. Go to the parent directory of lab2-src.
4. Type "turnin -c cs251 -p lab2 lab2-src"
5. Type "turnin -c cs251 -p lab2 -v" to make sure you have submitted the right files.