Homework 9: Subject

Due: March 31, 11:55 PM

Important

There are a few guidelines you must follow in this homework. If you fail to follow any of the following guidelines you will receive a **0** for the entire assignment.

1. All submitted code must compile under **JDK 7**. This includes unused code, don't submit extra files that don't compile. (Java is backwards compatabile so if it compiles under JDK 6 it *should* compile under JDK 7)
2. Don't include any package declarations in your classes.
3. Don't change any *existing* class headers, constructors, or method signatures. (It is fine to add extra methods and classes)
4. Don't import anything that would trivialize the assignment. (e.g. don't import `java.util.LinkedList` for a Linked List assignment. Ask if you are unsure.)
5. You must submit your source code, the `.java` files, not the compiled `.class` files.

After you submit your files redownload them and run them to make sure they are what you intended to submit. We are not responsible if you submit the wrong files.

Assignment

For this homework you will be implementing an algorithm to find the minimum weight spanning tree of a graph. The specific algorithm that you will be implementing is kruskals, using a disjoint set data structure to detect cycles.

Kruskals

You will be implementing kruskals algorithm in the static method inside the MST class.

The process

1. Take in a graph and get the collection of edges
2. Find the least weighted edge in the collection
3. Add the edge to your MST if it does not create a cycle (use the disjoint set data structure)
4. Repeat step 2 until all of the edges in your collection are gone

Disjoint Set Data Structure

This data structure is used to find out if two vertices are already in the same tree (or "set"). The data structure begins with all of the vertices in defferent sets. As you add edges to your result (connect vertices), you are connecting sets of vertices. As you connect them, they are combined into the same set. Eventually all of the vertices will be in the same set, meaning that you have a spanning tree.

For this you will need to have a hashMap mapping vertices to an object you create that can keep parent pointers for the sets.

When you add an edge, you need to merge the two sets that it connects. If they are in different sets.

- First find the root of the set for both of the vertices in the edge we are adding
  - To find this root we use our map and traverse up the parent pointers until we see null
  - Once we have the root node for both vertices, we will arbitrarily choose one of the roots and point it at the other root
    - Now, all of the nodes in both sets have the same root (hence, both sets have the same root node now)

How do we know if two vertices are in the same set? They will have the same root.

There are some optimization you are required to implement.

Path Compression

- This optimization flattens out the tree by reducing the depth. When traversing up the tree, make every node along the path point directly to the root that eventually gets returned. (This is best to do recursively)

Merge By Rank

- This optimization keeps the sets balanced by keeping track of each trees estimated depth (rank)
  - Let the rank of each set be 0 initially.
  - If you are merging two sets with differnt ranks, make the smaller ranked root point to the larger root and do not change the ranks.
  - If the two ranks are the same arbitrarily choose one to point to the other, and increment the rank of the new set.

Deliverables

You must submit all of the following files.

1. `MST.java`
2. `DisjointSets.java`

You may attach them each individually, or submit them in a zip archive.