Programming Assignment 8
A Graph Processing Program using GraphX

Due on Tuesday May 5 before midnight

Description

The purpose of this project is to develop a graph processing program using Pregel on Spark GraphX.

This project must be done individually. No copying is permitted. **Note: We will use a system for detecting software plagiarism, called Moss, which is an automatic system for determining the similarity of programs.** That is, your program will be compared with the programs of the other students in class as well as with the programs submitted in previous years. This program will find similarities even if you rename variables, move code, change code structure, etc.

Note that, if you use a Search Engine to find similar programs on the web, we will find these programs too. So don't do it because you will get caught and you will get an F in the course (this is cheating). Don't look for code to use for your project on the web or from other students (current or past). Just do your project alone using the help given in this project description and from your instructor and GTA only.

Platform

As in the previous projects, you will develop your program on SDSC Comet. Optionally, you may use your laaptop to help you develop your program, but you should test your programs on Comet before you submit them.

Setting up your Project

Login into Comet and download and untar project8:

```
wget http://lambda.uta.edu/cse6331/project8.tgz
tar xfz project8.tgz
chmod -R g-wrx,o-wrx project8
```

Look at the example example/src/main/scala/SSSPExample.scala

Project Description

You are asked to re-implement Project #5 (Graph Processing) using Pregel on Spark GraphX. That is, your program must find the connected components of any undirected graph and print the size of these connected components. An empty `project8/src/main/scala/Graph.scala` is provided, as well as scripts to build and run this code on Comet. **You should modify Graph.scala only**. **You should use the pregel method from the GraphX Pregel API only to write your code**. Your main program should take the text file that contains the graph (small-graph.txt or large-graph.txt) as an argument and print the results to the output. The stopping condition is when the number of repetition reaches 5.

You can compile Graph.scala using:

```
run graph.build
```

and you can run it in local mode over the small graph using:

```
sbatch graph.local.run
```

You should modify and run your programs in local mode until you get the correct result. After you make sure that your program runs correctly in local mode, you run it in distributed mode using:

```
sbatch graph.distr.run
```

This will work on the moderate-sized graph and will print the results to the output.

The following pseudo-code finds the connected components using Pregel:

1. Read the input graph and construct the RDD of edges
2. Use the graph builder Graph.fromEdges to construct a Graph from the RDD of edges
3. Access the VertexRDD and change the value of each vertex to be the vertex ID (initial group number)
4. Call the Graph.pregel method in the GraphX Pregel API to find the connected components. For each vertex, this method changes its group number to the minimum group number of its neighbors (if it is less than its current group number)
5. Group the graph vertices by their group number and print the group sizes.

Optional: Use your laptop to develop your project