**Programming Assignment 5  
Graph Processing on Spark**

Due on Thursday November 8 before midnight

**Description**

The purpose of this project is to develop a graph analysis program using Apache Spark.

This project must be done individually. No copying is permitted. **Note: We will use a system for detecting software plagiarism, called**[**Moss**](http://theory.stanford.edu/~aiken/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](https://lambda.uta.edu/cse6331/comet.html). Optionally, you may use IntelliJ Idea or Eclipse 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 project5:

wget http://lambda.uta.edu/cse6331/project5.tgz

tar xfz project5.tgz

chmod -R g-wrx,o-wrx project5

**Project Description**

You are asked to re-implement Project #3 (Graph Processing) using Spark and Scala. An empty project5/src/main/scala/Partition.scala is provided, as well as scripts to build and run this code on Comet. **You should modify Partition.scala only**. Your main program should take the text file that contains the graph (small-graph.txt or large-graph.txt) as an argument.

The graph can be represented as RDD[ ( Long, Long, List[Long] ) ], where the first Long is the graph node ID, the second Long is the assigned cluster ID (-1 if the node has not been assigned yet), and the List[Long] is the adjacent list (the IDs of the neighbors). Here is the pseudo-code:

var graph = /\* read graph from args(0); the graph cluster ID is set to -1 except for the first 5 nodes \*/

for (i <- 1 to depth)

graph = graph.flatMap{ /\* (1) \*/ }.groupByKey.map{ /\* (2) \*/ }

/\* finally, print partition sizes \*/

where the mapper function (1) returns two kinds of tuples from a node ( id, cluster, adjacent) in the graph:

1. ( id, Left(cluster,adjacent) ) (ie, pass the graph topology)
2. ( x, Right(cluster) ) for each neighbor x in adjacent if cluster is greater than -1.

Then, the reducer function (2) takes a tuple ( Long, Iterable[Either[(Long,List[Long]),Long]] ) from groupByKey and returns a graph node (Long,Long,List[Long]) using the following pseudo-code:

reducer\_function ( id, s ):

adjacent = Nil

cluster = -1

for each p in s:

if p matches Right(c) then cluster = c

if p matches Left(c,adj) where c>0 then return (id,c,adj) /\* the node has already been assigned a cluster \*/

if p matches Left(-1,adj) then adjacent = adj

return ( id, cluster, adjacent )

Please look at the [Either class](https://www.scala-lang.org/api/current/scala/util/Either.html). Note that in the Comet script, both the Spark local and distributed modes use 2 RDD partitions. This means that in order to get 10 centroids, we need to get 5 from each partition (the first 5 of each partition).

You can compile Partition.scala using:

run partition.build

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

sbatch partition.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 partition.distr.run

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

**What to Submit**

You need to submit the following files only:

project5/src/main/scala/Partition.scala

project5/partition.local.out

project5/partition.distr.out