INF 553 – Spring 2017

Homework #1: MapReduce & Spark

Due: February 10, Friday
100 points

1. **[Hadoop MapReduce, 60 points]** In this problem, you are asked to write a Hadoop MapReduce program, TwoPhase.java, that computes the multiplication of two given matrices using the two-phase approach described in class.

   A template file is provided to you, which contains the skeleton for mapper and reducer of phase one and two. You only need to supply codes for the block indicated by "// fill in your code" add import statements at the beginning of the file if necessary (you can only import packages from java.*). **DO NOT** modify other parts of the template.

   Note that the input matrices A and B are stored in two separate directories, e.g., mat-A and mat-B. Each directory contains a single text file, each line of which is a tuple (row-index, column-index, value) that specifies a non-zero value in the matrix.

$$\begin{bmatrix} 2 & 2 & 1 \\ 2 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 0 & 2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 7 \\ 2 & 4 \\ 3 & 3 \end{bmatrix}$$

A_{3X3} B_{3X2} C_{3X2}

   For example, the following is the content of mat-A/values.txt which stores the entries for the matrix A shown above.

```
0,0,2
0,1,2
0,2,1
1,0,2
1,1,1
2,0,1
2,2,2
```

   MultileInputs class is used to specify different mappers for different input directory.
   You also need to submit your jar file with name <FirstName>_<LastName>_2p.jar.

   Example invocation of your program is as follows:

```
bin/hadoop jar <FirstName>_<LastName>_2p.jar TwoPhase mat-A
mat-B output
```

   Your output directory (the file part-r-00000) should contain the entries of matrix C (= A * B) in the following format (tab-separated). For example,

```
1,1   3
1,2   7
2,1   2
2,2   4
3,1   3
3,2   3
```

Submissions: Submit both <FirstName>_<LastName>_TwoPhase.java and <FirstName>_<LastName>_2p.jar. DO NOT make them into folder or zip file.

2. **[40 points]** Write a Spark program in Python, TwoPhase.py, which implements the same 2-phase approach as in Problem 1.

   Restrictions: you can NOT use join (and any outer join variations) in your code, but may use any other transformations and actions provided in the lecture. This gives you a chance to think about how to implement join using other methods in Spark. It will also maximize the benefits of this exercise. In the end, you will implement a very similar program in Spark as you have done in Hadoop MapReduce for Problem 1.

   Your program should be invoked as follows.

```
bin/spark-submit <FirstName>_<LastName>_TwoPhase.py mat-A/values.txt mat-B/values.txt output.txt
```

   It should produce the same output as Problem 1 and store the output in a file "output.txt".

   Submissions: Submit <FirstName>_<LastName>_TwoPhase.py.

   Notes:

   - In both questions, you may assume that matrix entries are all integers.
   - Make sure to follow the output format (The order of output entries doesn't matter) and the naming format. If you don't follow either one or both of them, 20% points will be deducted.
   - You **MUST** implement your program using the two-phase approach described in class, and you **CAN NOT** use join transformation in your python program. If you don't follow these instructions, 80% points will be deducted.
   - Make sure that your program can be invoked correctly. If not, no more than 50% of the points can be earned.