First Homework Assignment

Due Date: **4/30/2020 at 5:00pm**

Submission Guidelines:

Please follow carefully the instructions posted on the course's github page when submitting your solutions (https://github.com/MSIA/bigdatacourse/blob/master/README.md). Failure to follow the instructions will result in lost points (-5).

Deliverables:

1. Your java source code file: name the file "exercisex.java" where x is either 1 or 2, 3, or 4. NOTE: the class name has to be "exercisex.java" as well.
2. Output of your MapReduce Job: name the file "exercisex.txt" where x is either 1, 2, 3, or 4.

Notes/tips:

1. All input files are available in /home/public/. Please copy them directly from /home/public/course to HDFS and not to your home directory on wolf.
2. The directory/home/public/WordCount-Gradle has directions on how to set up a Gradle project. You can use the same directions on your personal laptop if you wish to compile there.

Problem 1 (20 points)

Data: /home/public/google

Description: The data for this problem comes from Google's project for counting word frequencies in its entire Google Books collection. You are given two files: one file reports 1 grams (single words) and another file reports 2 grams (pairs of words; one following each other in text). The data represents the number of occurrences of a particular word (or pairs of words) in a given year across all books available by Google Books. The number of volumes/books containing a word (or pairs of words) is also reported.

Write a MapReduce program that reports the average number of volumes per year for words containing the following three substrings: 'nu,' 'chi,' 'haw'.

Example:

The 1 gram file format --the regex "\\s+" will match any kind of whitespace (space, tab etc):

word \\s+ **year** \\s+ **number of occurrences** \\s+ **number of volumes** \\s+ …

The 2 gram file format:

word \\s+ **word** \\s+ **year** \\s+ **number of occurrences** \\s+ **number of volumes** \\s+...

The final output should show the **year**, **substring**, and **average number of volumes across both bigram and unigram formats** where the substring appears in that year. For example:

2000,nu,345

2010,nu,200

1998,chi,31

The 'year' column may include erroneous values which can be a string. If the year field is a string, the record should be discarded.

If each word in the bi-gram includes the string, it should be counted twice in the average. For example, for the bi-gram "nugi hinunu" with volume of 10, when calculating the average, its contribution to the numerator should be 2 times 10 and in the denominator it should be 2. A unigram counts only once regardless of the number of occurrence of "nu" in the word.

Do this in the most efficient way with a **single** MapReduce job. Beyond the file formats described above, you are not allowed to make any structural assumptions on the data; e.g., that the 2 gram file contains more fields compared to the 1 gram file.

Problem 2 (40 points)

Data: /home/public/google

Description: The data set is the same as in the previous exercise. You need to compute the **standard deviation** of all volume values (across all records and both files) in MapReduce. The output should be a single value. You must not use more than 2 MapReduce jobs.

Problem 3 (15 points)

Data: /home/public/music

Description: The data for this problem is a subset of the million song database (https://www.kaggle.com/c/msdchallenge#description), and its size is 42GB. The file is in csv format. Your task is to extract the **song title** (column 1), **artist's name** (column 2) and **duration** (column 3) for all songs published between the years 2000 and 2010. The year is in column 166 – note that some year entries can be erroneous, and should be discarded. Additionally, you can assume that all songs are unique, so there is no need to remove any duplicates. You should do this as efficiently as possible in MapReduce.

Problem 4 (25 points)

Data: s3://msiahw2/music/

Description: The data set is the same as in the previous exercise. For each artist in the data set, compute the maximum duration across all of their songs. The output should be: **artist, max duration**.

In addition, the management of your firm wants the artists' names to be **sorted across all files based on the first character**. This means that the each output file of your MapReduce job must be sorted by the first character of an artist's name. You cannot take the output files and then sort them in a spreadsheet software. You are only allowed to concatenate the output files in the end.

In order to save computing resources for your firm, you have to use **5 reducers**.