t1. HDFS commands

a. hadoop fs -mkdir /user/hadoop
b. hadoop fs -copyFromLocal patients.csv /user/Hadoop
c. hadoop fs -mv /user/Hadoop/patients.csv /user/Hadoop/data.csv
d. hadoop fs -ls -R /user/cse482
e. hadoop fs -mv /user/Hadoop/data.csv /user/cse482/data.csv
f. hadoop fs -cp /user/cse482/data.csv /user/cse482/patients.csv
g. hadoop fs -cat /user/patients/patients.csv
h. hadoop fs -rm /user/cse482/data.csv
i. hadoop fs -getmerge /user/cse482/results ./output.txt
j. hadoop fs -get /user/patients/medication.txt ./medication.txt

2.

a.

Data set:

Amazon book ratings data. Each line in the data file has 4 columns (reviewer id, book id, book genre, rating), where ratings are integer-valued ranging from 1 to 4.

Problem:

Identify the highest rated book, i.e., the book with highest average rating, for each book genre. Note that each book can have more than one ratings (e.g., by different reviewers).

Mapper function:

Tokenize each line into a set of columns, filtering out reviewer id.

Mapper output:

key is genre, value is tuple of book id and rating.

Reducer input:

key is genre, value is a list of tuples of book id and rating.

Reducer function:

Sums up the ratings for each book for each genre, divides by amount of ratings for each book. Selects max book rating per genre.

Reducer output:

key is genre, value is book id with highest rating.

b.

Data set:

Movie preference data. Each record in the data file contains the movie title and list of users who liked the movie. For example, the record

jaws user111 user134 user313 user5812
star_wars user111 user313 user388 user4422

Problem:

For each pair of users, count the number of movies they both liked. The output may exclude pairs of users who do not have any movies they both liked.

Mapper function:

Tokenize each line into a set of words, selects pairs of users for each movie

Mapper output:

key is pair of users, value is 1.

Reducer input:

key is pair of users, value is a list of 1s.

Reducer function:

sums up the 1's for each key

Reducer output:

key is pair of users, value is number of movies they both liked.

c.

Data set:

Maximum and minimum daily temperature readings for weather stations from around the world. Each line in the data files has 4 columns (station id, date, max temperature, min temperature).

Problem:

Find the station id and date of anomalous temperature readings in the dataset. A temperature reading is anomalous if the minimum daily temperature exceeds the maximum temperature for the given day.

Mapper function:

Tokenize each line into a set of columns.

Mapper output:

key is tuple of station id and date, value is tuple of max temperature and min temperature.

Reducer input:

key is tuple of station id and date, value is tuple of max temperature and min temperature.

Reducer function:

Finds values with min temperatures larger than max temperatures.

Reducer output:

key is station id, value date of anomalous temperature readings.

d.

Data set:

Instagram friendship graph. Each record corresponds to an Instagram user, followed by a list of his/her friends. For example, the graph data may contain the following records:

john123 mary456 tom312 lee222
mary456 john123
tom312 john123 lee222
lee222 john123 tom312

The first line above states that mary456, tom312, and lee222 are friends of john123.

Problem:

Find pairs of Instagram users who are not friends with each other but who share one or more common friends. This is known as the "friend-of-a-friend" (FOF) problem. For example, mary456 and tom312 are both friends of john123, but they are not friends with each other. The

Hadoop program should only output the pair (u,v) if u < v. In the previous example, the program should only output the pair (mary456, tom312) but not (tom312, mary456).

Mapper function 1:

Tokenize each line into a set of users. User and friends are individually grouped with the known friends, and friends of the user are grouped as unknown. Known friends are also added to the unknown friends list.

Mapper output 1:

key is tuple of user and known or potential friend of friends, value is name of user that the key is or may be friends with, grouping only considered with original user (left most) and users to the right, see example in tom312 - mary is not added since she is to the left and is not the original user. Some examples are below:

((john123, known), mary456)
((john123, known), tom312)
((john123, known), lee222)
((john123, unknown), mary456)
((john123, unknown), tom312)
((john123, unknown), lee222)

((mary456, unknown), john123)
((mary456, unknown), tom312)
((mary456, unknown), lee222)

((tom312, unknown), john123)
((tom312, unknown), lee222)

((mary456, known), john123)

((tom312, known), john123)
((tom312, known), lee222)

((john123, unknown), tom312)
((john123, unknown), lee222)

((lee222, known),john123 )
((lee222, known),tom312 )

Reducer input 1:

key is tuple of user and wether friendship is known or potentially unknown, value is list of users that are the known friend or unknown friend.

((john123, known), [mary456, tom312, lee222])
((john123, unknown), [mary456, tom312, lee222])
((mary456, known), [john123])
((mary456, unknown), [john123, tom312, lee222])

((tom312, known), [john123, lee222])
((tom312, unknown), [john123, lee222])
((lee222, known),[john123, tom312] )
((lee222, known),[john123, tom312] )

Reducer function 1:

Just sends input through

Reducer output 1:

Same as input

Mapper function 2:

Seperates the list out of the key, value pair

Mapper output 2:

key is name of user, value is a list of users in relation with them

Reducer input 2:

key is name of user, value is two lists of relation
(john123, [mary456, tom312, lee222],[mary456, tom312, lee222])
(mary456, [john123],[john123, tom312, lee222])
(tom312, [john123, lee222],[john123, lee222])
(lee222,[john123, tom312], [john123, tom312] )

Reducer function 2:

Finds users that are not in both lists for each user

Reducer output 2:

(mary456, tom312)
(mary456, lee222)

e.

Data set:

Cancer data. Each line in the data file corresponds to a patient with the following nominal-valued attributes: patientID, gender, marital status, Smoker, Weight class, and Class, where the Class attribute has value yes or no to indicate whether the patient has cancer.

12345, female, married, smoker, normal, yes.
136666, male, single, nonsmoker, normal, no.
14423, male, married, smoker, overweight, yes.

Problem:

Compute the gini index for each of the following attributes: gender, marital status, smoker, and weight class, based on the distribution of their class values.

Mapper function:

Tokenize each line into a set of columns.

Mapper output:

key is attributes (gender, marital status...), value is tuple of attribute value and class.

Reducer input:

key is attribute, list of tuples of values and class.

Reducer function:

Calculates gini index for each attribute, counting values for each values of each attribute.

Reducer output:

key is attribute, value is gini index.