**CS6350:**

**Big data Management Analytics and Management**

**Spring 2021**

**Homework #1**

**Submission Deadline: March 22 (11.55 p.m.)**

In this homework, you will be using hadoop/mapreduce to analyze social network data.

**Q1**: Write a MapReduce program in Hadoop that implements a **simple “Mutual/Common friend list of two friends"**. The key idea is that if two people are friend then they have a lot of mutual/common friends. This program will find the common/mutual friend list for them.

For example,

Alice’s friends are Bob, Sam, Sara, Nancy

Bob’s friends are Alice, Sam, Clara, Nancy

Sara’s friends are Alice, Sam, Clara, Nancy

As Alice and Bob are friend and so, their mutual friend list is [Sam, Nancy]

As Sara and Bob are not friend and so, their mutual friend list is empty. (**In this case you may exclude them from your output**).

Input:

Input files

1. soc-LiveJournal1Adj.txt

**The input contains the adjacency list and has multiple lines in the following format:**

<User><TAB><Friends>

Hence, each line represents a particular user’s friend list separated by comma.

2. userdata.txt

The userdata.txt contains dummy data which consist of

column1 : userid

column2 : firstname

column3 : lastname

column4 : address

column5: city

column6 :state

column7 : zipcode

column8 :country

column9 :username

column10 : date of birth.

Here, <User> is a unique integer ID corresponding to a unique user and <Friends> is a comma-separated list of unique IDs corresponding to the friends of the user with the unique ID <User>. Note that the friendships are mutual (i.e., edges are undirected): if A is friend with B then B is also friend with A. The data provided is consistent with that rule as there is an explicit entry for each side of each edge. So when you make the pair, always consider (A, B) or (B, A) for user A and B but not both.

**Output: The output should contain one line per user in the following format:**

<User\_A>, <User\_B><TAB><Mutual/Common Friend List>

where <User\_A> & <User\_B> are unique IDs corresponding to a user A and B (A and B are friend). < Mutual/Common Friend List > is a comma-separated list of unique IDs corresponding to mutual friend list of User A and B.

Please generate/print the Mutual/Common Friend list for the following users.

(0,1), (20, 28193), (1, 29826), (6222, 19272), (28041, 28056)

**Q2.**

Please answer this question by using dataset from Q1.

Find friend pair(s) whose number of common friends is the maximum in all the pairs.

Output Format:

<User\_A>, <User\_B><TAB><Mutual/Common Friend Number>

**Q3.**

Please **use in-memory join at the Mapper** to answer the following question.

Given any two Users (they are friend) as input, output the list of the names and the date of birth (**mm/dd/yyyy**) of their mutual friends.

Note that the userdata.txt will be used to get the extra user information and cached/replicated at each mapper.

Output format:

UserA id, UserB id, list of [names: date of birth (**mm/dd/yyyy**)] of their mutual Friends.

Sample Output:

1234 4312 [John:12/05/1985, Jane : 10/04/1983, Ted: 08/06/1982]

**Q4.**

Please use **in-memory join at the Reducer** to answer the following question.

For each user print User ID and maximum age of direct friends of this user.

Sample output:

User A, 60

where User A is the id of a user and 60 represents the maximum age of direct friends for this particular user.

**Inverted Index:**

**Q5.**

Write a program that will construct inverted index in the following ways.

The **map** function parses each line in an input file, userdata.txt, and emits a sequence of <word, line number> pairs. The **reduce** function accepts all pairs for a given word, sorts the corresponding line numbers, and emits a <word, list(line numbers)> pair. The set of all the output pairs forms a simple inverted index.

What to submit

(i) Submit the source code via the eLearning website. (ii) Submit the output file for each question.