Missouri University of Science & Technology, Rolla, MO, USA
Fall 2023| HW2| CS6304| Instructor: Dr. Sanjay Madria, Professor, CS.

Finding mutual friends

1. Assume the friends are stored as Person-> [List of Friends], our friends list is then:

A -> B C D
B -> A C D E
C -> A B D E
D -> A B C E
E -> B C D

Each line will be an argument to a mapper. For every friend in the list of friends, the mapper will output a key-value pair. The key will be a friend along with the person. The value will be the list of friends. The key will be sorted so that the friends are in order, causing all pairs of friends to go to the same reducer. This is hard to explain with text, so let's just do it and see if you can see the pattern. After all the mappers are done running, you'll have a list like this:

For map (A -> B C D):

(A B) -> B C D
(A C) -> B C D
(A D) -> B C D

For map (B -> A C D E): (Note that A comes before B in the key)

(A B) -> A C D E
(B C) -> A C D E
(B D) -> A C D E
(B E) -> A C D E

For map (C -> A B D E):

(A C) -> A B D E
(B C) -> A B D E
(C D) -> A B D E
(C E) -> A B D E

For map (D -> A B C E):

(A D) -> A B C E
(B D) -> A B C E
(C D) -> A B C E
(D E) -> A B C E

And finally for map (E -> B C D):

(B E) -> B C D
(C E) -> B C D
(D E) -> B C D

Before we send these key-value pairs to the reducers, we group them by their keys and get:

(A B) -> (A C D E) (B C D)

(A C) -> (A B D E) (B C D)
(A D) -> (A B C E) (B C D)
(B C) -> (A B D E) (A C D E)
(B D) -> (A B C E) (A C D E)
(B E) -> (A C D E) (B C D)
(C D) -> (A B C E) (A B D E)
(C E) -> (A B D E) (B C D)
(D E) -> (A B C E) (B C D)

Each line will be passed as an argument to a reducer. The reduce function will simply intersect the lists of values and output the same key with the result of the intersection. For example, reduce ((A B) -> (A C D E) (B C D)) will output (A B): (C D) and means that friends A and B have C and D as common friends.

The result after reduction is:

(A B) -> (C D)
(A C) -> (B D)
(A D) -> (B C)
(B C) -> (A D E)
(B D) -> (A C E)
(B E) -> (C D)
(C D) -> (A B E)
(C E) -> (B D)
(D E) -> (B C)

Now when D visits B's profile, we can quickly look up (B D) and see that they have three friends in common, (A C E). **Implement** the above example in the dataset given below.

Perform the identical operation using both Hadoop and Spark on the provided input data, with the number of input lines specified in the table below. Create a bar chart that illustrates a comparison between the execution times of Hadoop and Spark.

Input (Number of lines from the input.txt file)	Time taken by Hadoop (ms)	Time taken by Spark (ms)
24997		
29997		
34997		
39997		
44997		
49997		

2. Find two friends who have highest number of mutual friends. Also **show** the mutual friends between two peoples who's first letter contains '1' or '5'. (For testing purposes, in this example, you can use 'a' or 'A').

Perform the identical operation using both Hadoop and Spark on the provided input data, with the number of input lines specified in the table below. Create a bar chart that illustrates a comparison between the execution times of Hadoop and Spark.

Input (Number of lines from the input.txt file)	Time taken by Hadoop (ms)	Time taken by Spark (ms)
24997		

29997		
34997		
39997		
44997		
49997		

Note : We need the mutual friends list to have values of '1' and '5'. PLEASE NOTE THAT THE 2 PEOPLE FOR WHOM YOU ARE FINDING THE MUTUAL FRIENDS DO NOT NEED TO BE '1' OR '5'.

Example of correct answers:
35678, 6789 -> 134567, 1456, 16799, 56789
45678, 23456 -> 56789, 54321, 14789

3. Find the average number of mutual friends in the dataset and show the friends who have mutual friends above the average number of mutual friends.

Perform the identical operation using both Hadoop and Spark on the provided input data, with the number of input lines specified in the table below. Create a bar chart that illustrates a comparison between the execution times of Hadoop and Spark.

Input (Number of lines from the input.txt file)	Time taken by Hadoop (ms)	Time taken by Spark (ms)
24997		
29997		
34997		
39997		
44997		
49997		

Note : The average needs to be calculated for all the pairs irrespective of the number of mutual friends. Even if the mutual friends are 0, count them.

** The best practice is to use the above example first to verify that your solution is working and then apply it to the main dataset (soc-LiveJournal1Adj.txt).*

Dataset:
Please use the following text file, where the first number is the user's ID. The friend list is separated by tab. In the friend list, friends are separated by a comma. Consider the following line from the dataset:

7 0,31993,40218,40433,1357,21843

Here, 7 is the user id and 0,31993,40218,40433,1357,21843 is the friend list of the user 7. Friends in the list are separated by comma.

soc-LiveJournal1Adj.txt

**The dataset is taken from the Internet, and it is an open dataset. I have not found any specific source to provide the credential. If you find it, please let us know.*