# Assignment 1 : People You Might Know

Roll Number : 17CS60R81

Name : Ashish Malgawa

Solution:

In this Program, I have used Multiple Mappers and reducers.

**First Mapper Phase (flat map to pair):**

In this phase two users are mapped into a tuple which will contain a tuple of two users as the key and if we say the user is the one before the tab and then we will create tuples of all the users on the right will the user before the tab and we will use this as a key and the value will be 0 and then we will create all possible combinations of the users on the right and we will pass the value one.

For eg., if A B,C

then ((A,B),0) ((A,C),0)

((B,C),1) ((C,B),1)

**First Reducer Phase (reduce by key):**

Since in spark after each line flatmaptopair() passes a list of the tuples to Reducer which it is connected to the reducer will check if the value of the tuple is 0 or was 0 then, in both the cases the user1 and user2 of the tuple are friends and when the value is not 0 or it was never 0 then we will add the earlier and the new value of the pair and after all the processing we will get number of mutual friends.

**Filter Phase:**

The output of the above phase is a JavaPairRDD object into a filter which will discard all the pairs with value zero ( the i.e pair of friends ).

**Second Mapper Phase:**

Now we will have tuples of users who are not friends with each other as the key and the value will contain number of mutual friends and our tuple will look like this ((user1,user2), no of mutual friends) and since we need to compare the mutual friends on the basis of mutual friends we need (user1,(user2, no of mutual friends)) we need a mapper to convert the tuples in this format and the same thing is done in this phase.

**Group by key + Map value phase:**

The above mapper functions makes it feasible to group the users who are not friends with the number of mutual friends and now after each line is being grouped we will use a custom class which implements comparable interface so that we can call collections.sort on our tuples and sort them on the basis of mutual friends then we will choose top 10 among them and add them to our JavaPairRDD object

**Final Mapper Phase:**

Since we have got final output we need to transform it into the output format and the final mapper will do the same.

# Output

| | |
|---|---|
| 924 | 439,2409,6995,11860,15416,43748,45881 |
| 8941 | 8493,8488,8489,8490,8492,8494,8499,8501,8503,8504 |
| 8942 | 8939,8940,8943,8944 |
| 9019 | 9022,317,9023 |
| 9020 | 9021,9016,9017,9022,317,9023 |
| 9021 | 9020,9016,9017,9022,317,9023 |
| 9022 | 9019,9020,9021,317,9016,9017,9023 |
| 9990 | 13134,13478,13877,34299,34485,34642,37941 |
| 9992 | 9987,9989,35667,9991 |
| 9993 | 9991,13134,13478,13877,34299,34485,34642,37941 |