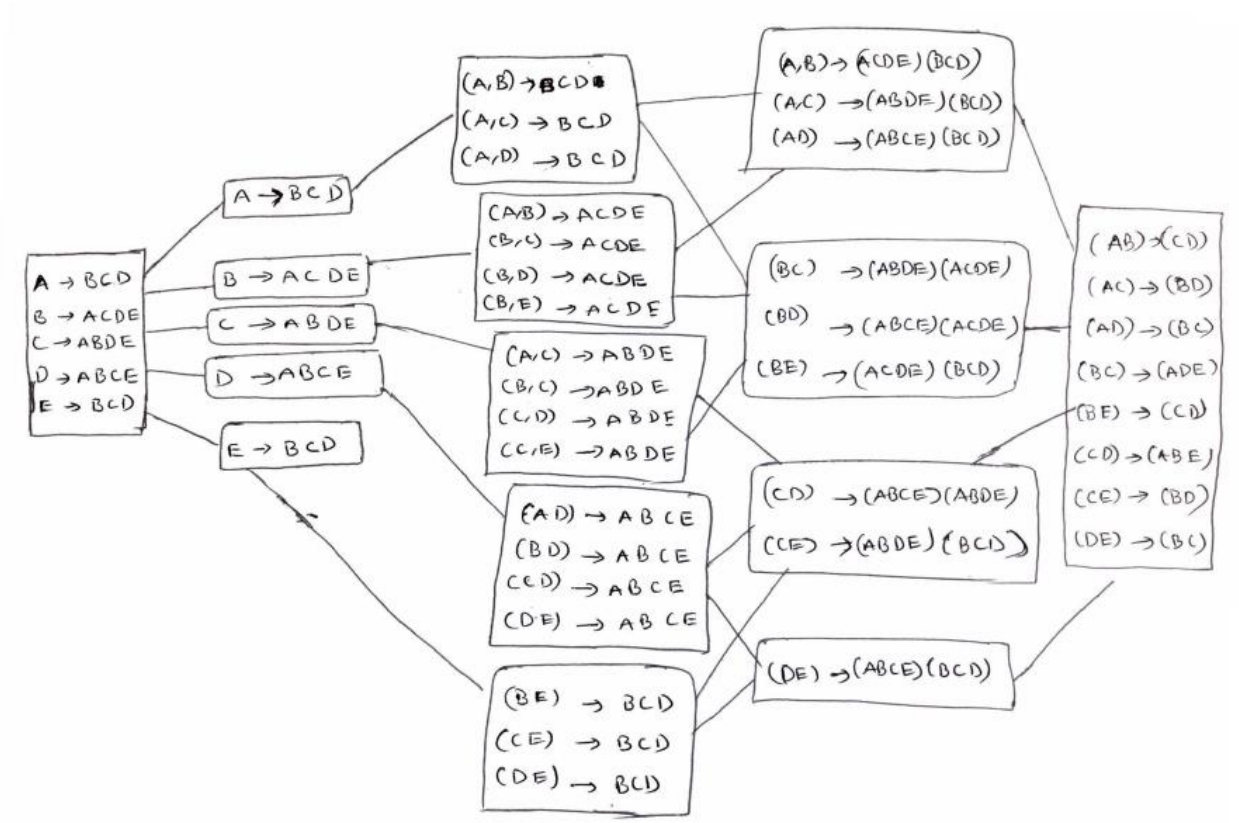
**Name: Mudunuri Sri Sai Sarat Chandra Varma**

**Class ID: 14**

# Problem Set (PS -2B)

**1)Draw a MapReduce diagram similar to the word count diagram below.**

**Answer:**




**2) Sketch a MapReduce algorithm for the common Facebook friends.**

**Answer:**

**MapReduce Algorithm for common Facebook friends:**

```
Class Mapper(key, value) {
        friends_list = (A, B, C…… N);
        foreach value in (A, B, C……N) {
                friendKey = buildSortedkey(person, value);
```

```
                emit(friendKey, friends_list);
        }
}

 Class Reducer(key, value) {
        method buildSortedkey(friend1, friend2) {
        If (friend1 < friend2) {
                return method(friend1, friend2)
        {
        Else {
                Return method(friend2, friend1)
        }
}
}
```

Mapper's output keys are sorted and this property will prevent duplicate keys. Note that we assume that friendship is a bi-directional thing: if A is a friend of B, then B is a friend of A. The reduce() function finds the common friends for every pair of users by intersecting all associated friends in between.

**3) Sketch Spark Scala implementation**

**Answer:**

**Scala Implementation:**

```
Class TokenizerMapper extends Mapper[Object,Text,Text,Intwritable] {

  val common = userToFriendsListdata.mapValues(friends =>{
    val Count = friends.foldLeft(0)((sum,_) => sum + 1)
    val common  = friends.flatten
      .groupBy(identity)
      .mapValues(f => f.foldLeft(0)((sum,_) => sum + 1))
      .filter(e => e._2 == listsCount) .keys.toList.sorted
    common
  }).filter(row => row._2.nonEmpty)
    .sortBy(row => (row._1._1, row._1._2))

}

}

Class sumReducer extends Reducer[Text, IntWritable,Text,IntWritable]{
```

```
  println("Common friends")Friendsdata.take(10).foreach(println(_))
  spark.stop()
}
```

The above code is the scala implementation for common friends. It takes friends of each person and group them by other persons mutual friends using ‘groupBy’, sort them using ‘sortBy’and finally for each person it gives their common friends. 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. 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.