BIG DATA PROGRAMMING

PROJECT-2

Case1-FacebookMutualFriendsUsingSpark

Team Members and collaboration:

Roshini varada --**Facebook Mutual Friends Using Spark**

Sarika Reddy Kota -- **Spark Data Frames**

Pallavi Arikatla – **Spark Streaming**

Zakari, Abdulmuhaymin –**Spark Graph Frames**

Idea: (Question5- Part-1)

To identify the Common Friends of any two people in a social network with the help of map-reduce using spark. In spark the MapReduce algorithm is hundred times faster than the Hadoop MapReduce and is efficient. In spark we use the transformations such as group by, map, flat-map and reduce to perform the operation.

Usage or the real time scenario: (Question5- Part-2)

Social networking sites identifies the common friends between people. When a person visits another person profile then they can see the mutual contacts

Example:

If P1 and P2 has P3,P4 as their common friends, If P1 visits P2's profile P1 can identify P3,P4 as their common friends and vice versa.

Approach and solution: (Question5- Part-3)

Flow Chart or Pictorial Representation:

1.The identification of mutual friends for a unit takes place in 5 phases. They are

a.Input

b.GroupBy

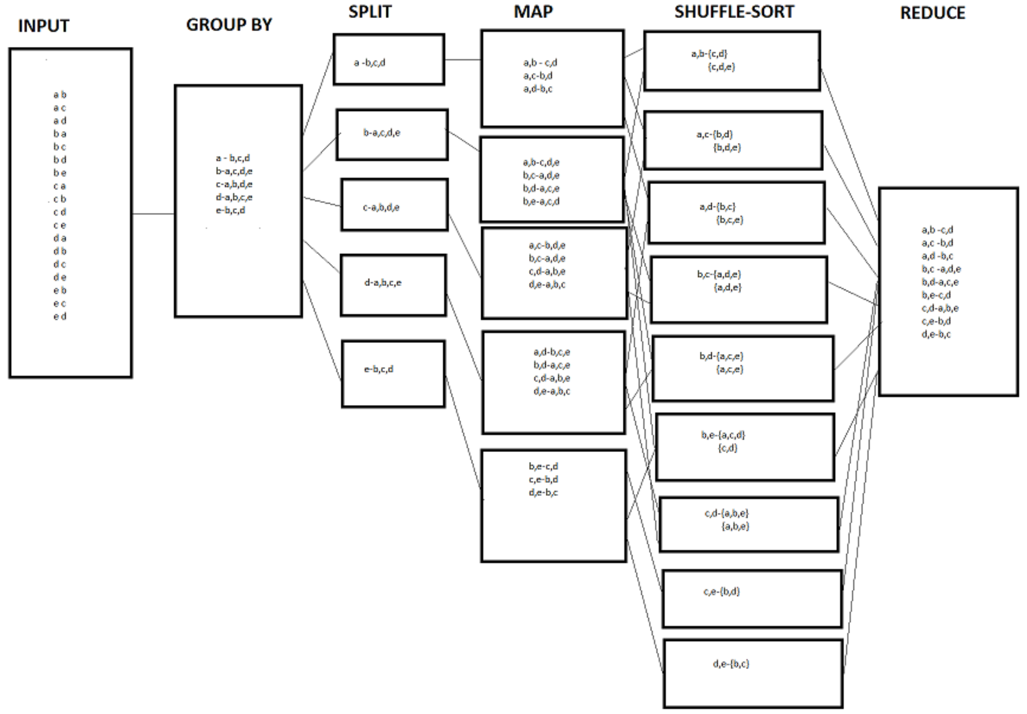
c.Split

d.Map

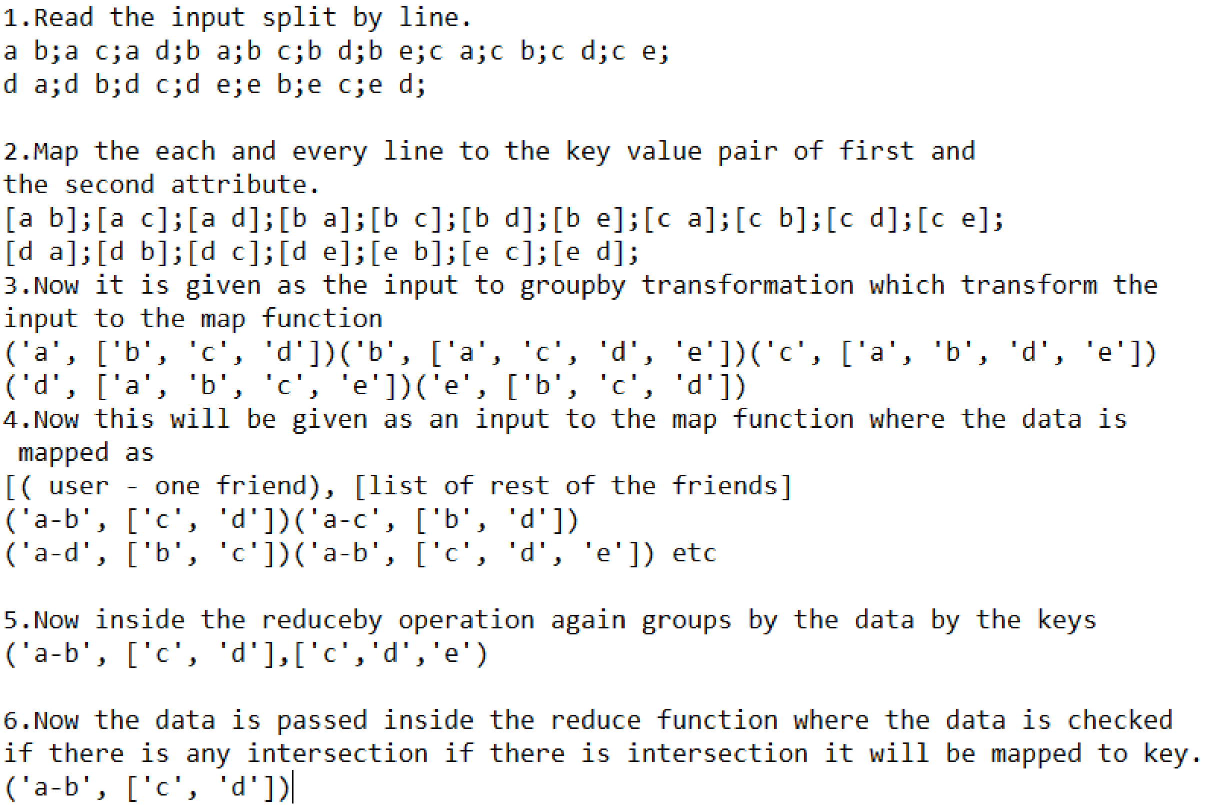
e.Shuffle

f.Reduce

The below diagram represents how the input transforms in each phase.



c)Algorithm For the approach



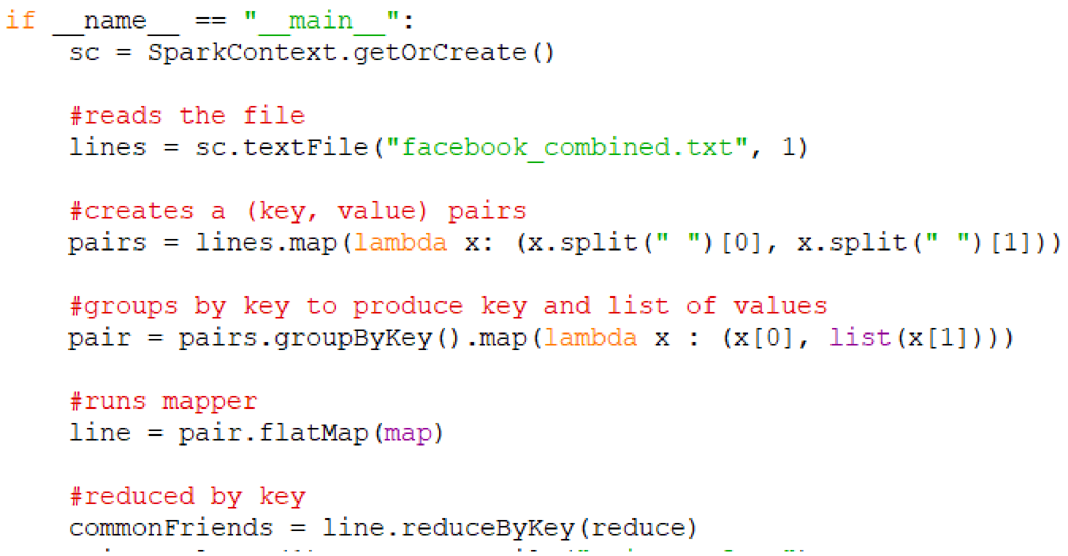
```
1.Read the input split by line.
a b;a c;a d;b a;b c;b d;b e;c a;c b;c d;c e;
d a;d b;d c;d e;e b;e c;e d;

2.Map the each and every line to the key value pair of first and
the second attribute.
[a b];[a c];[a d];[b a];[b c];[b d];[b e];[c a];[c b];[c d];[c e];
[d a];[d b];[d c];[d e];[e b];[e c];[e d];
3.Now it is given as the input to groupby transformation which transform the
input to the map function
('a', ['b', 'c', 'd'])('b', ['a', 'c', 'd', 'e'])('c', ['a', 'b', 'd', 'e'])
('d', ['a', 'b', 'c', 'e'])('e', ['b', 'c', 'd'])
4.Now this will be given as an input to the map function where the data is
 mapped as
[( user - one friend), [list of rest of the friends]
('a-b', ['c', 'd'])('a-c', ['b', 'd'])
('a-d', ['b', 'c'])('a-b', ['c', 'd', 'e']) etc

5.Now inside the reduceby operation again groups by the data by the keys
('a-b', ['c', 'd'],['c','d','e')

6.Now the data is passed inside the reduce function where the data is checked
if there is any intersection if there is intersection it will be mapped to key.
('a-b', ['c', 'd'])
```

Implementation

a)Implementing mapreduce using spark with a simple example.

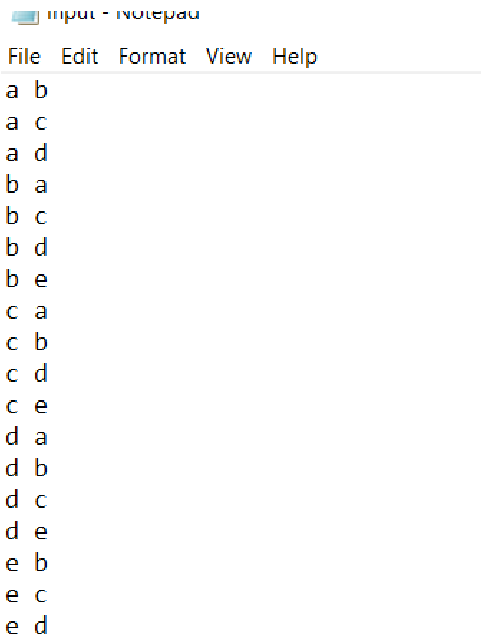
1.Here the input taken is the first attribute indicates the user and the second attribute indicates the friend of the user.

Main method



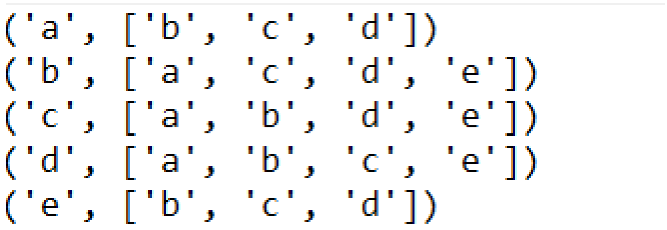
```
if __name__ == "__main__":
    sc = SparkContext.getOrCreate()

    #reads the file
    lines = sc.textFile("facebook_combined.txt", 1)

    #creates a (key, value) pairs
    pairs = lines.map(lambda x: (x.split(" ")[0], x.split(" ")[1]))

    #groups by key to produce key and list of values
    pair = pairs.groupByKey().map(lambda x : (x[0], list(x[1])))

    #runs mapper
    line = pair.flatMap(map)

    #reduced by key
    commonFriends = line.reduceByKey(reduce)
```

Input:



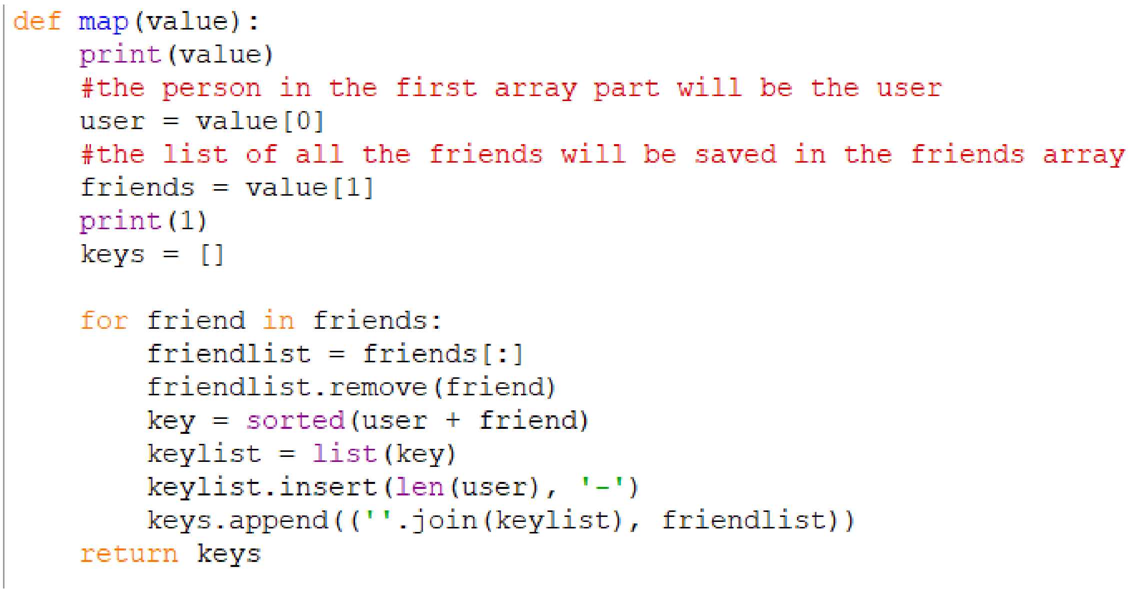
```
a b
a c
a d
b a
b c
b d
b e
c a
c b
c d
c e
d a
d b
d c
d e
e b
e c
e d
```

2.After this the users are mapped with their friends using groupby key.

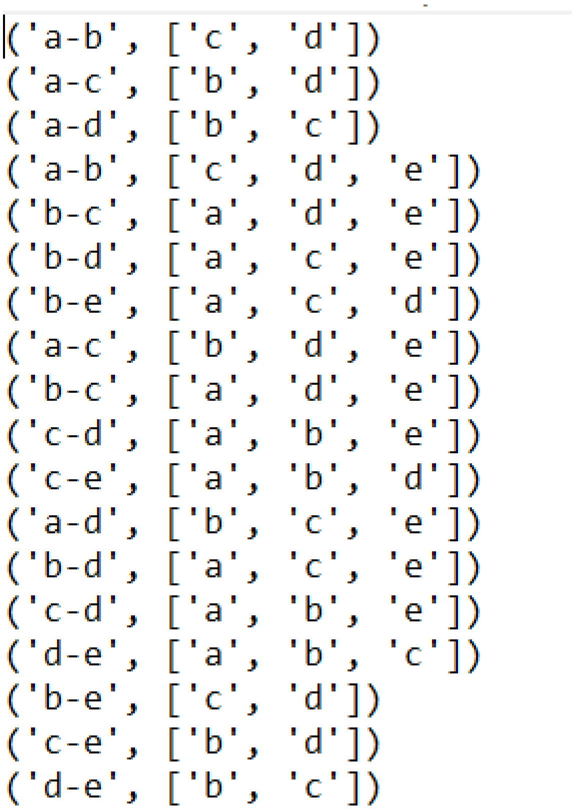


```
('a', ['b', 'c', 'd'])
('b', ['a', 'c', 'd', 'e'])
('c', ['a', 'b', 'd', 'e'])
('d', ['a', 'b', 'c', 'e'])
('e', ['b', 'c', 'd'])
```

3.After grouping the data is split and the mapping is done for each user. Here each user is paired with the other user and the and the list of the other friends is formed.

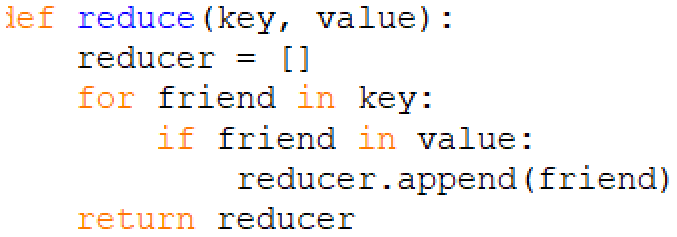


```
def map(value):
    print(value)
    #the person in the first array part will be the user
    user = value[0]
    #the list of all the friends will be saved in the friends array
    friends = value[1]
    print(1)
    keys = []

    for friend in friends:
        friendlist = friends[:]
        friendlist.remove(friend)
        key = sorted(user + friend)
        keylist = list(key)
        keylist.insert(len(user), '-')
        keys.append((''.join(keylist), friendlist))
    return keys
```

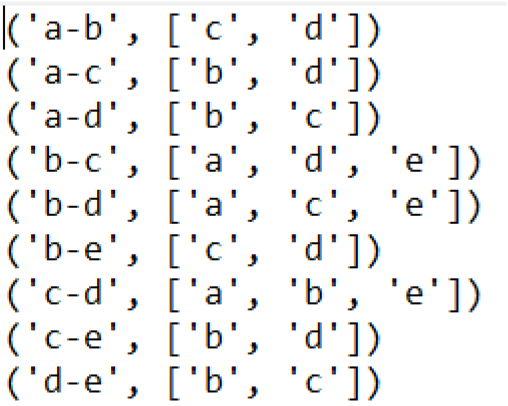


```
('a-b', ['c', 'd'])
('a-c', ['b', 'd'])
('a-d', ['b', 'c'])
('a-b', ['c', 'd', 'e'])
('b-c', ['a', 'd', 'e'])
('b-d', ['a', 'c', 'e'])
('b-e', ['a', 'c', 'd'])
('a-c', ['b', 'd', 'e'])
('b-c', ['a', 'd', 'e'])
('c-d', ['a', 'b', 'e'])
('c-e', ['a', 'b', 'd'])
('a-d', ['b', 'c', 'e'])
('b-d', ['a', 'c', 'e'])
('c-d', ['a', 'b', 'e'])
('d-e', ['a', 'b', 'c'])
('b-e', ['c', 'd'])
('c-e', ['b', 'd'])
('d-e', ['b', 'c'])
```

4.Then the data is reduced using the reduce by operation after the shuffle sorting.



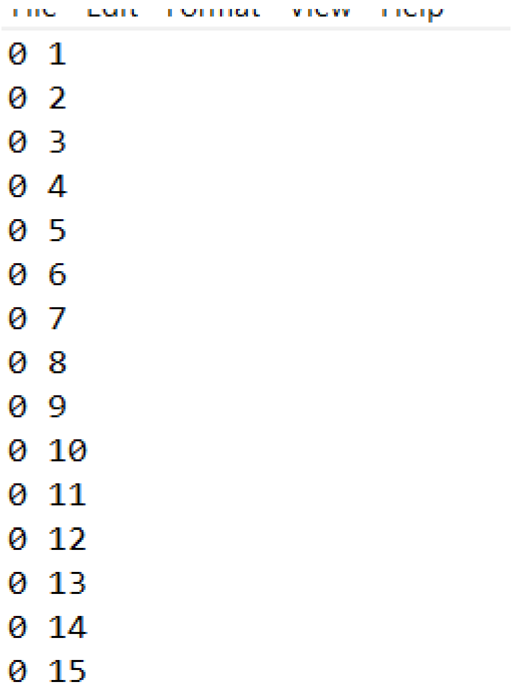
```
lef reduce(key, value):
    reducer = []
    for friend in key:
        if friend in value:
            reducer.append(friend)
    return reducer
```



```
('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'])
```

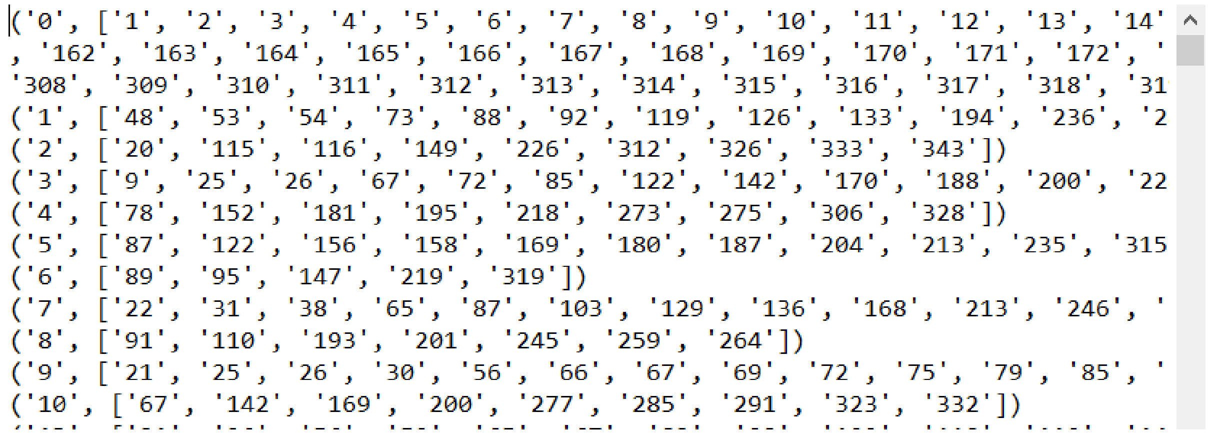
b)Implementing mapreduce using spark data set given.

1.The facebook_combined data is taken as the input.



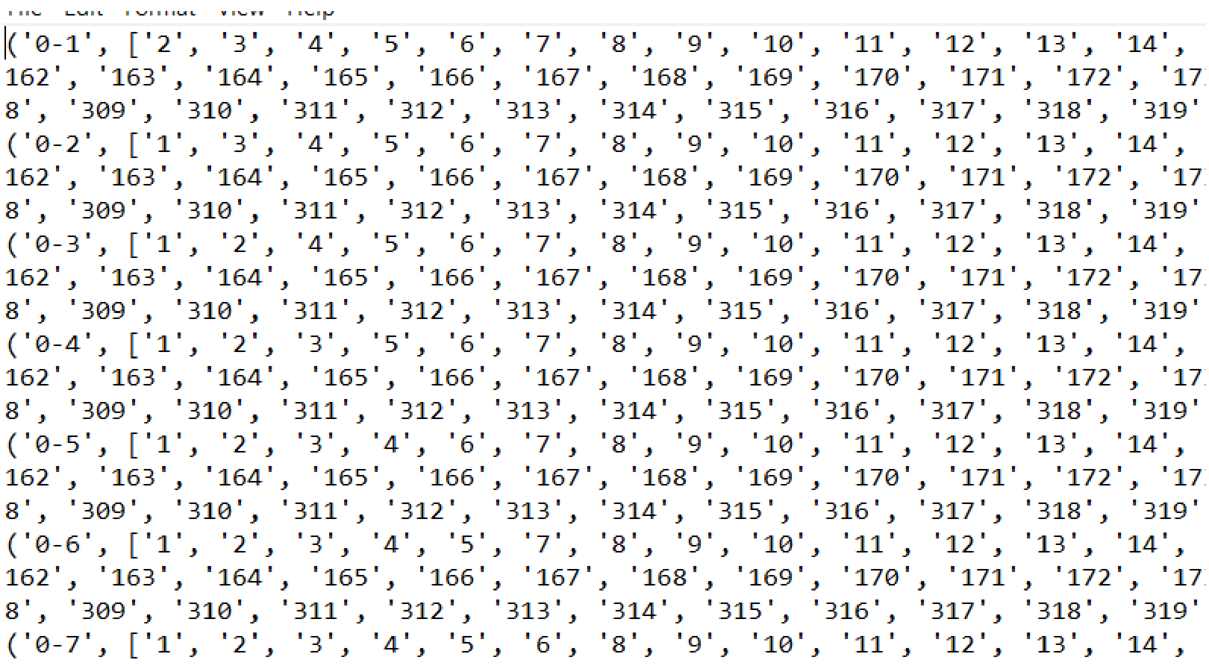
```
0 1
0 2
0 3
0 4
0 5
0 6
0 7
0 8
0 9
0 10
0 11
0 12
0 13
0 14
0 15
```

2.Output after grouping.



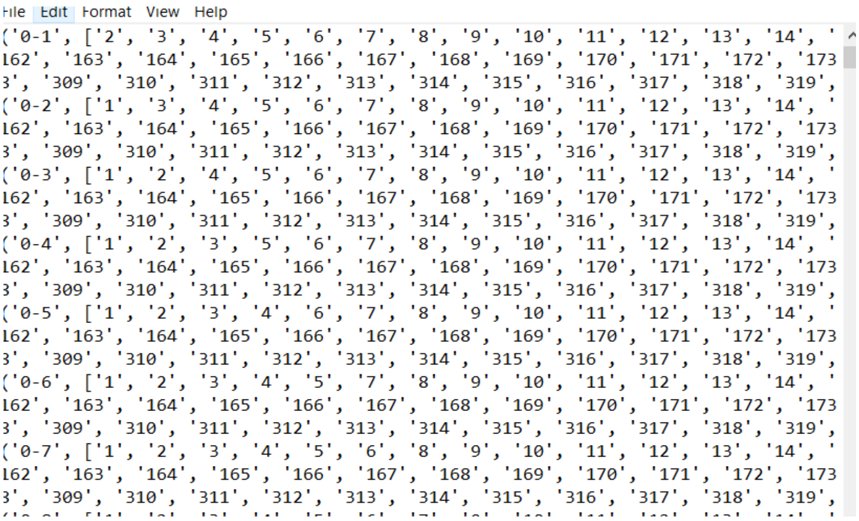
```
('0', ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14'
, '162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '
'308', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '31
('1', ['48', '53', '54', '73', '88', '92', '119', '126', '133', '194', '236', '2
('2', ['20', '115', '116', '149', '226', '312', '326', '333', '343'])
('3', ['9', '25', '26', '67', '72', '85', '122', '142', '170', '188', '200', '22
('4', ['78', '152', '181', '195', '218', '273', '275', '306', '328'])
('5', ['87', '122', '156', '158', '169', '180', '187', '204', '213', '235', '315
('6', ['89', '95', '147', '219', '319'])
('7', ['22', '31', '38', '65', '87', '103', '129', '136', '168', '213', '246', '
('8', ['91', '110', '193', '201', '245', '259', '264'])
('9', ['21', '25', '26', '30', '56', '66', '67', '69', '72', '75', '79', '85', '
('10', ['67', '142', '169', '200', '277', '285', '291', '323', '332'])
```

3.Output after mapping



```
('0-1', ['2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14',
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '17
8', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319'
('0-2', ['1', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14',
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '17
8', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319'
('0-3', ['1', '2', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14',
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '17
8', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319'
('0-4', ['1', '2', '3', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14',
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '17
8', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319'
('0-5', ['1', '2', '3', '4', '6', '7', '8', '9', '10', '11', '12', '13', '14',
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '17
8', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319'
('0-6', ['1', '2', '3', '4', '5', '7', '8', '9', '10', '11', '12', '13', '14',
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '17
8', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319'
('0-7', ['1', '2', '3', '4', '5', '6', '8', '9', '10', '11', '12', '13', '14',
```

4.The final output after reduction.



```
('0-1', ['2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '173
3', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319',
('0-2', ['1', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '173
3', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319',
('0-3', ['1', '2', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '173
3', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319',
('0-4', ['1', '2', '3', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '173
3', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319',
('0-5', ['1', '2', '3', '4', '6', '7', '8', '9', '10', '11', '12', '13', '14', '
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '173
3', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319',
('0-6', ['1', '2', '3', '4', '5', '7', '8', '9', '10', '11', '12', '13', '14', '
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '173
3', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319',
('0-7', ['1', '2', '3', '4', '5', '6', '8', '9', '10', '11', '12', '13', '14', '
162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '173
3', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319',
```

Challenges faced (Question5-Part-4)

1.The coding part was smooth without any difficulties.

Integration and Milestones (Question5- Part-5)

Since all the cases are independent from each other there are no difficulties faced in the integration part.

Links

Github Link

https://github.com/RoshiniVarada/BDP_Project2/tree/master/CASE-1

Link for Sourcecode

https://github.com/RoshiniVarada/BDP_Project2/tree/master/CASE-1/Sourcecode

Link for YouTube Video

`https://youtu.be/FPTmD63Nh9A`

Team Members Links

Use-case1 -FacebookMutualFriendsUsingSpark

Wiki-link- https://github.com/RoshiniVarada/BDP_Project2/wiki/CASE1

Use-case2-Spark Data Frames

Wiki-link- https://github.com/RoshiniVarada/BDP_Project2/wiki/CASE2

Use-case3- Spark Streaming

Wiki-link- https://github.com/RoshiniVarada/BDP_Project2/wiki/CASE3

Use-case4-Spark-Graphx

Wiki-link- https://github.com/RoshiniVarada/BDP_Project2/wiki/CASE4

Submitted by

Roshini Varada(16302628)