Big Data Analytics – CS7070

Programming Project #2

Disha Rao, M13254448

1. *Phase-1: Write a program for Spark (in PySpark, Scala, or Spark-JAVA) that takes as input a list of edges of a graph, and outputs a node-list representation of the graph. In the output, for each node (=key) the value is a list of all the nodes to which the key node is connected.*

    a. *Items to be submitted: (i) your program source code, (ii) A list of input records used by your program, (iii) Output of your program for the TinyDataSet, (iv) Output of your program for the SmallDataSet.*

Solution:

*(**Output at each stage is highlighted in Yellow**)*

(i) **Tiny dataset source code:**

```
%pyspark
import pandas as pd

%pyspark
tiny_dataset = pd.read_excel("/home/maria_dev/pp2/TinyDataset.xlsx",header=None,names=['key','value'])
keys = tiny_dataset['key'].values
values = tiny_dataset['value'].values

%pyspark
from collections import defaultdict
adjacency_list = defaultdict(list)
for k, v in zip(keys, values):
     adjacency_list[k].append(v)

%pyspark
adjacency_list

defaultdict(<class 'list'>, {7: [10, 8, 4], 8: [9, 5], 9: [5, 10], 10: [6], 4: [5, 6], 5: [6], 1: [4, 3], 2: [3, 6], 3: [4, 6]})

%pyspark
adjacency_list_new = adjacency_list
def add_edge(node,node_list):

    for element in node_list:
        if not node in adjacency_list[element]:
            adjacency_list_new[element].append(node)


adjacency_dict = {k: add_edge(k,v) for k, v in adjacency_list.items()}

%pyspark
adjacency_list_new

```

```
defaultdict(<class 'list'>, {7: [10, 8, 4], 8: [9, 5, 7], 9: [5, 10, 8], 10: [6, 7, 9], 4: [5, 6, 7, 1, 3], 5: [6, 8, 9, 4], 1: [4, 3], 2: [3, 6], 3: [4, 6, 1, 2], 6: [10, 4, 5, 2, 3]})

%pyspark
with open('/home/maria_dev/tiny_dataset_new.txt','w') as textfile:
    for key,node_list in adjacency_list_new.items():
        for value in node_list:
            textfile.write(str(key))
            textfile.write(" ")
            textfile.write(str(value))
            textfile.write("\n")

%pyspark
adjacency_rdd = sc.textFile("/tmp/data/tiny_dataset_new.txt")
phase1_rdd = adjacency_rdd.flatMap(lambda line:line.split("\n")).map(lambda pair:pair.split(" ")).map(lambda tuple:(int(tuple[0]),[int(tuple[1])])).reduceByKey(lambda a,b:a+b).sortByKey()

%pyspark
phase1_rdd.collect()
[(1, [4, 3]), (2, [3, 6]), (3, [4, 6, 1, 2]), (4, [5, 6, 7, 1, 3]), (5, [6, 8, 9, 4]), (6, [10, 4, 5, 2, 3]), (7, [10, 8, 4]), (8, [9, 5, 7]), (9, [5, 10, 8]), (10, [6, 7, 9])]

%pyspark
phase1_rdd = phase1_rdd.map(lambda r:(r[0],sorted(r[1])))
phase1_rdd.collect()
[(1, [3, 4]), (2, [3, 6]), (3, [1, 2, 4, 6]), (4, [1, 3, 5, 6, 7]), (5, [4, 6, 8, 9]), (6, [2, 3, 4, 5, 10]), (7, [4, 8, 10]), (8, [5, 7, 9]), (9, [5, 8, 10]), (10, [6, 7, 9])]


%pyspark
phase1_rdd.coalesce(1).saveAsTextFile("/home/maria_dev/pp2/phase1_output")
```

(ii) **List of input records :**

Tiny dataset and small dataset in /home/maria_dev/pp2

(iii) **Output of Tiny dataset:**

The output of the file is present at the HDFS location: /home/maria_dev/pp2/phase1_output

```
(1, [3, 4])
(2, [3, 6])
(3, [1, 2, 4, 6])
(4, [1, 3, 5, 6, 7])
(5, [4, 6, 8, 9])
(6, [2, 3, 4, 5, 10])
(7, [4, 8, 10])
(8, [5, 7, 9])
(9, [5, 8, 10])
(10, [6, 7, 9])
```

Source code for the Small dataset:

```
%pyspark
import pandas as pd

%pyspark
small_dataset = pd.read_excel("/home/maria_dev/pp2/SmallDataSet.xlsx",header=None,names=['key','value'])
keys = small_dataset['key'].values
values = small_dataset['value'].values

%pyspark
from collections import defaultdict
adjacency_list = defaultdict(list)
for k, v in zip(keys, values):
     adjacency_list[k].append(v)

%pyspark
adjacency_list
defaultdict(<class 'list'>, {1: [2, 3, 10, 6, 37], 2: [3, 6, 7, 11], 3: [4, 5, 7, 12, 8], 4: [5, 8, 9, 13], 5: [4, 9, 33], 6: [7, 10], 8: [7, 9, 13, 12], 9: [14, 13], 7: [12, 11], 11: [6, 10, 15], 20: [19, 29, 24, 21], 14: [34, 18, 23], 13: [14, 12], 12: [11], 28: [29, 40], 30: [28, 31, 25], 31: [28], 32: [28, 31, 30, 27, 23, 36], 29: [30, 31, 24], 24: [28, 19], 25: [29, 20], 21: [30, 25, 17, 22], 26: [30, 31, 27, 21], 27: [31, 22], 23: [27, 22, 35], 22: [31, 26], 19: [28, 39], 10: [19, 15, 38], 15: [19], 40: [39], 16: [15, 11, 12, 20, 21], 17: [22, 16, 12, 13], 18: [13, 23, 22], 33: [34], 34: [35], 35: [36], 37: [38], 38: [39]})


%pyspark
adjacency_list_new = adjacency_list
def add_edge(node,node_list):

    for element in node_list:
        if not node in adjacency_list[element]:
            adjacency_list_new[element].append(node)


adjacency_dict = {k: add_edge(k,v) for k, v in adjacency_list.items()}

%pyspark
adjacency_list_new

defaultdict(<class 'list'>, {1: [2, 3, 10, 6, 37], 2: [3, 6, 7, 11, 1], 3: [4, 5, 7, 12, 8, 1, 2], 4: [5, 8, 9, 13, 3], 5: [4, 9, 33, 3], 6: [7, 10, 1, 2, 11], 8: [7, 9, 13, 12, 3, 4], 9: [14, 13, 4, 5, 8], 7: [12, 11, 2, 3, 6, 8], 11: [6, 10, 15, 2, 7, 12, 16], 20: [19, 29, 24, 21, 25, 16], 14: [34, 18, 23, 9, 13], 13: [14, 12, 4, 8, 9, 17, 18], 12: [11, 3, 8, 7, 13, 16, 17], 28: [29, 40, 30, 31, 32, 24, 19], 30: [28, 31, 25, 32, 29, 21, 26], 31: [28, 30, 32, 29, 26, 27, 22], 32: [28, 31, 30, 27, 23, 36], 29: [30, 31, 24, 20, 28, 25], 24: [28, 19, 20, 29], 25: [29, 20, 30, 21], 21: [30, 25, 17, 22, 20, 26, 16], 26: [30, 31, 27, 21, 22], 27: [31, 22, 32, 26, 23], 23: [27, 22, 35, 14, 32, 18], 22: [31, 26, 21, 27, 23, 17, 18], 19: [28, 39, 20, 24, 10, 15], 10: [19, 15, 38, 1, 6, 11], 15: [19, 11, 10, 16], 16: [15, 11, 12, 20, 21, 17], 17: [22, 16, 12, 13, 21]
```

```
    , 18: [13, 23, 22, 14], 33: [34, 5], 34: [35, 14, 33], 35: [36, 23, 34], 37: [38, 1], 3
    8: [39, 10, 37], 36: [32, 35], 39: [19, 40, 38],40: [39, 28]})
35.
36. %pyspark
37. with open('/home/maria_dev/small_dataset_new.txt','w') as textfile:
38.     for key,node_list in adjacency_list_new.items():
39.         for value in node_list:
40.             textfile.write(str(key))
41.             textfile.write(" ")
42.             textfile.write(str(value))
43.             textfile.write("\n")
44.
45. %pyspark
46. adjacency_rdd = sc.textFile("/tmp/data/small_dataset_new.txt")
47. phase1_rdd = adjacency_rdd.flatMap(lambda line:line.split("\n")).map(lambda pair:pair.s
    plit(" ")).map(lambda tuple:(int(tuple[0]),[int(tuple[1])])).reduceByKey(lambda a,b:a+b
    ).sortByKey()
48.
49.
50. %pyspark
51. phase1_rdd = phase1_rdd.map(lambda r:(r[0],sorted(r[1])))
52. phase1_rdd.collect()
53. [(1, [2, 3, 6, 10, 37]), (2, [1, 3, 6, 7, 11]), (3, [1, 2, 4, 5, 7, 8, 12]), (4, [3, 5,
     8, 9, 13]), (5, [3, 4, 9, 33]), (6, [1, 2, 7, 10, 11]), (7, [2, 3, 6, 8, 11, 12]), (8,
     [3, 4, 7, 9, 12, 13]), (9, [4, 5, 8, 13, 14]), (10, [1, 6, 11, 15, 19, 38]), (11, [2,
    6, 7, 10, 12, 15, 16]), (12, [3, 7, 8, 11, 13, 16, 17]), (13, [4, 8, 9, 12, 14, 17, 18]
    ), (14, [9, 13, 18, 23, 34]), (15, [10, 11, 16, 19]), (16, [11, 12, 15, 17, 20, 21]), (
    17, [12, 13, 16, 21, 22]), (18, [13, 14, 22, 23]), (19, [10, 15, 20, 24, 28, 39]), (20,
     [16, 19, 21, 24, 25, 29]), (21, [16, 17, 20, 22, 25, 26, 30]), (22, [17, 18, 21, 23, 2
    6, 27, 31]), (23, [14, 18, 22, 27, 32, 35]), (24, [19, 20, 28, 29]), (25, [20, 21, 29,
    30]), (26, [21, 22, 27, 30, 31]), (27, [22, 23, 26, 31, 32]), (28, [19, 24, 29, 30, 31,
     32, 40]), (29, [20, 24, 25, 28, 30, 31]), (30, [21, 25, 26, 28, 29, 31, 32]), (31, [22
    , 26, 27, 28, 29, 30, 32]), (32, [23, 27, 28, 30, 31, 36]), (33, [5, 34]), (34, [14, 33
    , 35]), (35, [23, 34, 36]), (36, [32, 35]), (37, [1, 38]), (38, [10, 37, 39]), (39, [19
    , 38, 40]), (40, [28, 39])]
54.
55.
56. %pyspark
57. phase1_rdd.coalesce(1).saveAsTextFile("/home/maria_dev/pp2/phase1_output_smalldataset")
```

(iv)**Output of Small dataset:**

```
1.  (1, [2, 3, 6, 10, 37])
2.  (2, [1, 3, 6, 7, 11])
3.  (3, [1, 2, 4, 5, 7, 8, 12])
4.  (4, [3, 5, 8, 9, 13])
5.  (5, [3, 4, 9, 33])
6.  (6, [1, 2, 7, 10, 11])
7.  (7, [2, 3, 6, 8, 11, 12])
8.  (8, [3, 4, 7, 9, 12, 13])
9.  (9, [4, 5, 8, 13, 14])
10. (10, [1, 6, 11, 15, 19, 38])
11. (11, [2, 6, 7, 10, 12, 15, 16])
12. (12, [3, 7, 8, 11, 13, 16, 17])
13. (13, [4, 8, 9, 12, 14, 17, 18])
14. (14, [9, 13, 18, 23, 34])
15. (15, [10, 11, 16, 19])
```

```
16. (16, [11, 12, 15, 17, 20, 21])
17. (17, [12, 13, 16, 21, 22])
18. (18, [13, 14, 22, 23])
19. (19, [10, 15, 20, 24, 28, 39])
20. (20, [16, 19, 21, 24, 25, 29])
21. (21, [16, 17, 20, 22, 25, 26, 30])
22. (22, [17, 18, 21, 23, 26, 27, 31])
23. (23, [14, 18, 22, 27, 32, 35])
24. (24, [19, 20, 28, 29])
25. (25, [20, 21, 29, 30])
26. (26, [21, 22, 27, 30, 31])
27. (27, [22, 23, 26, 31, 32])
28. (28, [19, 24, 29, 30, 31, 32, 40])
29. (29, [20, 24, 25, 28, 30, 31])
30. (30, [21, 25, 26, 28, 29, 31, 32])
31. (31, [22, 26, 27, 28, 29, 30, 32])
32. (32, [23, 27, 28, 30, 31, 36])
33. (33, [5, 34])
34. (34, [14, 33, 35])
35. (35, [23, 34, 36])
36. (36, [32, 35])
37. (37, [1, 38])
38. (38, [10, 37, 39])
39. (39, [19, 38, 40])
40. (40, [28, 39])
```