<u>**Big Data Analytics – CS7070**</u>

<u>**Programming Project #2**</u>

3. Phase-3: Write a program for Spark (in PySpark, Scala, or Spark-JAVA) that takes as input a graph in the form of 2-hop projection (value) for each node (key) and produces a list of (key, value) pairs such that for each node (key), the value is all the triangles incident on that node.

a. Items to be submitted: (i) yourprogram source code, (ii) Output of your program for the graph of TinyDataSet, and (ii) Output of your program for the graph of SmallDataSet.

<u>**Solution:**</u>

<u>(i) Source code:</u>

```
 %pyspark
phase2_tiny_rdd = sc.textFile("/user/maria_dev/Phase2_output_tinydataset.txt")
phase2_tiny_rdd.collect()

['{1: [{3: [1, 2, 4, 6], 4: [1, 3, 5, 6, 7]}]}', '{2: [{3: [1, 2, 4, 6], 6: [2, 3, 4, 5, 10]}]}', '{3: [{1: [3, 4], 2: [3, 6], 4: [1, 3, 5, 6, 7], 6: [2, 3, 4, 5, 10]}]}', '{4: [{1: [3, 4], 3: [1, 2, 4, 6], 5: [4, 6, 8, 9], 6: [2, 3, 4, 5, 10], 7: [4, 8, 10]}]}', '{5: [{4: [1, 3, 5, 6, 7], 6: [2, 3, 4, 5, 10], 8: [5, 7, 9], 9: [5, 8, 10]}]}', '{6: [{2: [3, 6], 3: [1, 2, 4, 6], 4: [1, 3, 5, 6, 7], 5: [4, 6, 8, 9], 10: [6, 7, 9]}]}', '{7: [{4: [1, 3, 5, 6, 7], 8: [5, 7, 9], 10: [6, 7, 9]}]}', '{8: [{5: [4, 6, 8, 9], 7: [4, 8, 10], 9: [5, 8, 10]}]}', '{9: [{5: [4, 6, 8, 9], 8: [5, 7, 9], 10: [6, 7, 9]}]}', '{10: [{6: [2, 3, 4, 5, 10], 7: [4, 8, 10], 9: [5, 8, 10]}]}']

%pyspark
phase2_tiny_transformed = phase2_tiny_rdd.map(lambda x:eval(x))
phase2_tiny_transformed.collect()

[{1: [{3: [1, 2, 4, 6], 4: [1, 3, 5, 6, 7]}]}, {2: [{3: [1, 2, 4, 6], 6: [2, 3, 4, 5, 10]}]}, {3: [{1: [3, 4], 2: [3, 6], 4: [1, 3, 5, 6, 7], 6: [2, 3, 4, 5, 10]}]}, {4: [{1: [3, 4], 3: [1, 2, 4, 6], 5: [4, 6, 8, 9], 6: [2, 3, 4, 5, 10], 7: [4, 8, 10]}]}, {5: [{4: [1, 3, 5, 6, 7], 6: [2, 3, 4, 5, 10], 8: [5, 7, 9], 9: [5, 8, 10]}]}, {6: [{2: [3, 6], 3: [1, 2, 4, 6], 4: [1, 3, 5, 6, 7], 5: [4, 6, 8, 9], 10: [6, 7, 9]}]}, {7: [{4: [1, 3, 5, 6, 7], 8: [5, 7, 9], 10: [6, 7, 9]}]}, {8: [{5: [4, 6, 8, 9], 7: [4, 8, 10], 9: [5, 8, 10]}]}, {9: [{5: [4, 6, 8, 9], 8: [5, 7, 9], 10: [6, 7, 9]}]}, {10: [{6: [2, 3, 4, 5, 10], 7: [4, 8, 10], 9: [5, 8, 10]}]}]

%pyspark
phase2_intermediate = phase2_tiny_transformed.map(lambda x:{k:list(dicts.keys()) for k, v in x.items() for dicts in v })
phase2_intermediate.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
from itertools import combinations
```

```
phase2_flattened = phase2_intermediate.map(lambda x:{k:sorted([k]+v) for k,v in x.items() })
phase2_flattened.collect()

[{1: [1, 3, 4]}, {2: [2, 3, 6]}, {3: [1, 2, 3, 4, 6]}, {4: [1, 3, 4, 5, 6, 7]}, {5: [4, 5, 6, 8, 9]}, {6: [2, 3, 4, 5, 6, 10]}, {7: [4, 7, 8, 10]}, {8: [5, 7, 8, 9]}, {9: [5, 8, 9, 10]}, {10: [6, 7, 9, 10]}]

%pyspark
phase2_rdd = phase2_flattened.map(lambda x:{k:list(combinations(v,3)) for k,v in x.items()})
phase2_rdd_new = phase2_rdd

phase2_rdd.collect()

[{1: [(1, 3, 4)]}, {2: [(2, 3, 6)]}, {3: [(1, 2, 3), (1, 2, 4), (1, 2, 6), (1, 3, 4), (1, 3, 6), (1, 4, 6), (2, 3, 4), (2, 3, 6), (2, 4, 6), (3, 4, 6)]}, {4: [(1, 3, 4), (1, 3, 5), (1, 3, 6), (1, 3, 7), (1, 4, 5), (1, 4, 6), (1, 4, 7), (1, 5, 6), (1, 5, 7), (1, 6, 7), (3, 4, 5), (3, 4, 6), (3, 4, 7), (3, 5, 6), (3, 5, 7), (3, 6, 7), (4, 5, 6), (4, 5, 7), (4, 6, 7), (5, 6, 7)]}, {5: [(4, 5, 6), (4, 5, 8), (4, 5, 9), (4, 6, 8), (4, 6, 9), (4, 8, 9), (5, 6, 8), (5, 6, 9), (5, 8, 9), (6, 8, 9)]}, {6: [(2, 3, 4), (2, 3, 5), (2, 3, 6), (2, 3, 10), (2, 4, 5), (2, 4, 6), (2, 4, 10), (2, 5, 6), (2, 5, 10), (2, 6, 10), (3, 4, 5), (3, 4, 6), (3, 4, 10), (3, 5, 6), (3, 5, 10), (3, 6, 10), (4, 5, 6), (4, 5, 10), (4, 6, 10), (5, 6, 10)]}, {7: [(4, 7, 8), (4, 7, 10), (4, 8, 10), (7, 8, 10)]}, {8: [(5, 7, 8), (5, 7, 9), (5, 8, 9), (7, 8, 9)]}, {9: [(5, 8, 9), (5, 8, 10), (5, 9, 10), (8, 9, 10)]}, {10: [(6, 7, 9), (6, 7, 10), (6, 9, 10), (7, 9, 10)]}]

%pyspark
def trans(x,y):
    x.update(y)
    return x

phase2_flat_rdd = phase2_rdd.reduce(lambda x,y:trans(x,y))
phase2_flat_rdd

{1: [(1, 3, 4)], 2: [(2, 3, 6)], 3: [(1, 2, 3), (1, 2, 4), (1, 2, 6), (1, 3, 4), (1, 3, 6), (1, 4, 6), (2, 3, 4), (2, 3, 6), (2, 4, 6), (3, 4, 6)], 4: [(1, 3, 4), (1, 3, 5), (1, 3, 6), (1, 3, 7), (1, 4, 5), (1, 4, 6), (1, 4, 7), (1, 5, 6), (1, 5, 7), (1, 6, 7), (3, 4, 5), (3, 4, 6), (3, 4, 7), (3, 5, 6), (3, 5, 7), (3, 6, 7), (4, 5, 6), (4, 5, 7), (4, 6, 7), (5, 6, 7)], 5: [(4, 5, 6), (4, 5, 8), (4, 5, 9), (4, 6, 8), (4, 6, 9), (4, 8, 9), (5, 6, 8), (5, 6, 9), (5, 8, 9), (6, 8, 9)], 6: [(2, 3, 4), (2, 3, 5), (2, 3, 6), (2, 3, 10), (2, 4, 5), (2, 4, 6), (2, 4, 10), (2, 5, 6), (2, 5, 10), (2, 6, 10), (3, 4, 5), (3, 4, 6), (3, 4, 10), (3, 5, 6), (3, 5, 10), (3, 6, 10), (4, 5, 6), (4, 5, 10), (4, 6, 10), (5, 6, 10)], 7: [(4, 7, 8), (4, 7, 10), (4, 8, 10), (7, 8, 10)], 8: [(5, 7, 8), (5, 7, 9), (5, 8, 9), (7, 8, 9)], 9: [(5, 8, 9), (5, 8, 10), (5, 9, 10), (8, 9, 10)], 10: [(6, 7, 9), (6, 7, 10), (6, 9, 10), (7, 9, 10)]}

%pyspark
rdd = sc.parallelize([phase2_flat_rdd])
rdd.collect()

[{1: [(1, 3, 4)], 2: [(2, 3, 6)], 3: [(1, 2, 3), (1, 2, 4), (1, 2, 6), (1, 3, 4), (1, 3, 6), (1, 4, 6), (2, 3, 4), (2, 3, 6), (2, 4, 6), (3, 4, 6)], 4: [(1, 3, 4), (1, 3, 5), (1, 3, 6), (1, 3, 7), (1, 4, 5), (1, 4, 6), (1, 4, 7), (1, 5, 6), (1, 5, 7), (1, 6, 7), (3, 4, 5), (3, 4, 6), (3, 4, 7), (3, 5, 6), (3, 5, 7), (3, 6, 7), (4, 5, 6), (4, 5, 7), (4, 6, 7), (5, 6, 7)], 5: [(4, 5, 6), (4, 5, 8), (4, 5, 9), (4, 6, 8), (4, 6, 9), (4, 8, 9), (5, 6, 8), (5, 6, 9), (5, 8, 9), (6, 8, 9)], 6: [(2, 3, 4), (2, 3, 5), (2, 3, 6), (2, 3, 10), (2, 4, 5), (2, 4, 6), (2, 4, 10), (2, 5, 6), (2, 5, 10), (2, 6, 10), (3, 4, 5), (3, 4, 6), (3, 4, 10), (3, 5, 6), (3, 5, 10), (3, 6, 10), (4, 5, 6), (4, 5, 10), (4, 6, 10), (5, 6, 10)], 7: [(4, 7, 8), (4, 7, 10), (4, 8, 10), (7, 8, 10)], 8: [(5,
7, 8), (5, 7, 9), (5, 8, 9), (7, 8, 9)], 9: [(5, 8, 9), (5, 8, 10), (5, 9, 10), (8, 9, 10)], 10: [(6, 7, 9), (6, 7, 10), (6, 9, 10), (7, 9, 10)]}]


%pyspark
def filter_elements(x):
    for k,v in x.items():
        b = []
        for i,item in enumerate(v):
            if k in item:
                b.append(item)

    return {k:b}
phase2_rdd_2 = phase2_rdd_new.map(lambda x: filter_elements(x))
phase2_rdd_2.collect()

[{1: [(1, 3, 4)]}, {2: [(2, 3, 6)]}, {3: [(1, 2, 3), (1, 3, 4), (1, 3, 6), (2, 3, 4), (2, 3, 6), (3, 4, 6)]}, {4: [(1, 3, 4), (1, 4, 5), (1, 4, 6), (1, 4, 7), (3, 4, 5), (3, 4, 6), (3, 4, 7), (4, 5, 6), (4, 5, 7), (4, 6, 7)]}, {5: [(4, 5, 6), (4, 5, 8), (4, 5, 9), (5, 6, 8), (5, 6, 9), (5, 8, 9)]}, {6: [(2, 3, 6), (2, 4, 6), (2, 5, 6), (2, 6, 10), (3, 4, 6), (3, 5, 6), (3, 6, 10), (4, 5, 6), (4, 6, 10), (5, 6, 10)]}, {7: [(4, 7, 8), (4, 7, 10), (7, 8, 10)]}, {8: [(5, 7, 8), (5, 8, 9), (7, 8, 9)]}, {9: [(5, 8, 9), (5, 9, 10), (8, 9, 10)]}, {10: [(6, 7, 10), (6, 9, 10), (7, 9, 10)]}]

%pyspark
#same as phase2_flattened. “rr” can be replaced with phase2_flattened

phase1_rdd = sc.textFile("/user/maria_dev/Phase1_output_tinydataset.txt")
phase1_op_rdd = phase1_rdd.map(lambda x:eval(x)).map(lambda x:{x[0]:x[1]})
phase1_op_rdd.collect()

def trans_1(x,y):
    x.update(y)
    return x

rr = phase1_op_rdd.reduce(lambda x,y : trans_1(x,y))
rr
{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
def filter_non_triangles(x):

    for k,v in x.items():
        b=[]
        l=[]
        for tuples in v:
            if k not in tuples:
                b.append("a")
            else:
                l = list(tuples)
                l.remove(k)
                a = l[0]
                if l[1] in rr.get(a):
                    b.append(tuples)

    return {k:b}



```

```
phase2_final = phase2_rdd_2.map(lambda x: filter_non_triangles(x))
phase2_final.collect()

[{1: [(1, 3, 4)]}, {2: [(2, 3, 6)]}, {3: [(1, 3, 4), (2, 3, 6), (3, 4, 6)]}, {4: [(1, 3, 4), (3, 4, 6), (4, 5, 6)]}, {5: [(4, 5, 6), (5, 8, 9)]}, {6: [(2, 3, 6), (3, 4, 6), (4, 5, 6)]}, {7: []}, {8: [(5, 8, 9)]}, {9: [(5, 8, 9)]}, {10: []}]

%pyspark
phase2_final.coalesce(1).saveAsTextFile("/home/maria_dev/pp2/phase3_tiny_output")
```

(ii) Output for Tiny Dataset:

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

(iii) Output for Small Dataset:

```
{1: [(1, 2, 3), (1, 2, 6), (1, 6, 10)]}
{2: [(1, 2, 3), (1, 2, 6), (2, 3, 7), (2, 6, 7), (2, 6, 11), (2, 7, 11)]}
{3: [(1, 2, 3), (2, 3, 7), (3, 4, 5), (3, 4, 8), (3, 7, 8), (3, 7, 12), (3, 8, 12)]}
{4: [(3, 4, 5), (3, 4, 8), (4, 5, 9), (4, 8, 9), (4, 8, 13), (4, 9, 13)]}
{5: [(3, 4, 5), (4, 5, 9)]}
{6: [(1, 2, 6), (1, 6, 10), (2, 6, 7), (2, 6, 11), (6, 7, 11), (6, 10, 11)]}
{7: [(2, 3, 7), (2, 6, 7), (2, 7, 11), (3, 7, 8), (3, 7, 12), (6, 7, 11), (7, 8, 12), (7, 11, 12)]}
{8: [(3, 4, 8), (3, 7, 8), (3, 8, 12), (4, 8, 9), (4, 8, 13), (7, 8, 12), (8, 9, 13), (8, 12, 13)]}
{9: [(4, 5, 9), (4, 8, 9), (4, 9, 13), (8, 9, 13), (9, 13, 14)]}
{10: [(1, 6, 10), (6, 10, 11), (10, 11, 15), (10, 15, 19)]}
{11: [(2, 6, 11), (2, 7, 11), (6, 7, 11), (6, 10, 11), (7, 11, 12), (10, 11, 15), (11, 12, 16), (11, 15, 16)]}
{12: [(3, 7, 12), (3, 8, 12), (7, 8, 12), (7, 11, 12), (8, 12, 13), (11, 12, 16), (12, 13, 17), (12, 16, 17)]}
{13: [(4, 8, 13), (4, 9, 13), (8, 9, 13), (8, 12, 13), (9, 13, 14), (12, 13, 17), (13, 14, 18)]}
{14: [(9, 13, 14), (13, 14, 18), (14, 18, 23)]}
{15: [(10, 11, 15), (10, 15, 19), (11, 15, 16)]}
{16: [(11, 12, 16), (11, 15, 16), (12, 16, 17), (16, 17, 21), (16, 20, 21)]}
{17: [(12, 13, 17), (12, 16, 17), (16, 17, 21), (17, 21, 22)]}
{18: [(13, 14, 18), (14, 18, 23), (18, 22, 23)]}
{19: [(10, 15, 19), (19, 20, 24), (19, 24, 28)]}
{20: [(16, 20, 21), (19, 20, 24), (20, 21, 25), (20, 24, 29), (20, 25, 29)]}
{21: [(16, 17, 21), (16, 20, 21), (17, 21, 22), (20, 21, 25), (21, 22, 26), (21, 25, 30), (21, 26, 30)]}
{22: [(17, 21, 22), (18, 22, 23), (21, 22, 26), (22, 23, 27), (22, 26, 27), (22, 26, 31), (22, 27, 31)]}
{23: [(14, 18, 23), (18, 22, 23), (22, 23, 27), (23, 27, 32)]}
{24: [(19, 20, 24), (19, 24, 28), (20, 24, 29), (24, 28, 29)]}
```

```
{25: [(20, 21, 25), (20, 25, 29), (21, 25, 30), (25, 29, 30)]}
{26: [(21, 22, 26), (21, 26, 30), (22, 26, 27), (22, 26, 31), (26, 27, 31), (26, 30, 31)]}
{27: [(22, 23, 27), (22, 26, 27), (22, 27, 31), (23, 27, 32), (26, 27, 31), (27, 31, 32)]}
{28: [(19, 24, 28), (24, 28, 29), (28, 29, 30), (28, 29, 31), (28, 30, 31), (28, 30, 32), (28, 31, 32)]}
{29: [(20, 24, 29), (20, 25, 29), (24, 28, 29), (25, 29, 30), (28, 29, 30), (28, 29, 31), (29, 30, 31)]}
{30: [(21, 25, 30), (21, 26, 30), (25, 29, 30), (26, 30, 31), (28, 29, 30), (28, 30, 31), (28, 30, 32), (29, 30, 31), (30, 31, 32)]}
{31: [(22, 26, 31), (22, 27, 31), (26, 27, 31), (26, 30, 31), (27, 31, 32), (28, 29, 31), (28, 30, 31), (28, 31, 32), (29, 30, 31), (30, 31, 32)]}
{32: [(23, 27, 32), (27, 31, 32), (28, 30, 32), (28, 31, 32), (30, 31, 32)]}
{33: []}
{34: []}
{35: []}
{36: []}
{37: []}
{38: []}
{39: []}
{40: []}
```