CS6220 Homework 2

Map Reduce: Friends of Friends

Include your code in this file. Make sure the below piece of code is at the top, as we will use that variable for testing.

Tips and tricks

- Besides the Spark documentation, use the REPL feature heavily, since you'll be able to see functionality and functions.
- One function you may find useful is the `collect()` function that can collects the RDD from all machines and brings it into memory. This is only feasible for small datasets, and it will allow you to effectively debug.
- You can mount the `datapath` from a Google Drive. That way you won't have to keep uploading to Google Colab.
    - Try using the following code block:

```
from google.colab import drive
drive.mount('/content/drive')
```

- The total runtime is around 10 minutes, where you'll only notice in the reduce step. Spark is a lazy evaluator, and only when there's a `collect` or other evaluator step will you notice the lag.

Data path for file. We will use the variable `data_path` for grading.

```
#@title Data path for file. We will use the variable `data_path` for grading.
datapath="/content/drive/MyDrive/<path-to-soc-LiveJournal1Adj.txt>" #@param
```

datapath: /content/drive/MyDrive/<

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

```
!pip install matplotlib-venn
```

```
!apt-get -qq install -y libfluidsynth1
```

```
# https://pypi.python.org/pypi/libarchive
!apt-get -qq install -y libarchive-dev && pip install -U libarchive
import libarchive
```

```
# https://pypi.python.org/pypi/pydot
!apt-get -qq install -y graphviz && pip install pydot
import pydot
```

```
!pip install cartopy
import cartopy
```

```
!pip install pyspark
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
Requirement already satisfied: pyspark in /usr/local/lib/python3.8/dist-packa
Requirement already satisfied: py4j==0.10.9.5 in /usr/local/lib/python3.8/dis
```

Your Code Below

```
#@title Your Code Below

from pyspark import SparkConf, SparkContext



# Create a SparkConf object and a SparkContext
conf = SparkConf().setAppName("Friends-Recommend")
sc = SparkContext(conf=conf)

# Load the data into an RDD
sc = SparkContext.getOrCreate()
data = sc.textFile("/content/drive/MyDrive/soc-LiveJournal1Adj.txt", 1)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-d46a7c4e8521> in <module>
      5 # Create a SparkConf object and a SparkContext
      6 conf = SparkConf().setAppName("Friends-Recommend")
----> 7 sc = SparkContext(conf=conf)
      8
      9 # Load the data into an RDD

                                 2 frames

ValueError: Cannot run multiple SparkContexts at once; existing
SparkContext(app=Friends-Recommend, master=local[*]) created by __init__ at
<ipython-input-4-e63422141914>:7
```

SEARCH STACK OVERFLOW

```
# Perform the mapping step
mapped_data = data.map(lambda x: (x.split("\t")[0], x.split("\t")[1]))
mapped_data.collect()
```

```
[('0',

'1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,2

 ('1',
```

```
'0,5,20,135,2409,8715,8932,10623,12347,12846,13840,13845,14005,20075,21556,22

 ('2',

'0,117,135,1220,2755,12453,24539,24714,41456,45046,49927,6893,13795,16659,328

 ('3', '0,12,41,55,1532,12636,13185,27552,38737'),
 ('4',

'0,8,14,15,18,27,72,80,15326,19068,19079,24596,42697,46126,74,77,33269,38792,

 ('5',

'0,1,20,2022,22939,23527,30257,32503,35633,41457,43262,44846,49574,31140,3282

 ('6',

'0,21,98,2203,3238,5040,8795,9843,9847,15294,17874,18286,18311,18320,20553,35

 ('7', '0,31993,40218,40433,1357,21843'),
 ('8', '0,4,38,46,72,85,24777,83,33380'),
 ('9', '0,6085,18972,19269'),
 ('10', '0,12,16,30,6027,13793,23557,29581,35477,35617,44310'),
 ('11',

'0,1754,6027,7789,11142,12633,17898,19049,22486,26970,27554,27585,27591,27679

 ('12',

'0,3,10,16,29,38,41,55,1085,1532,7714,27679,29379,35195,38737,43121,30,83,85,

 ('13',

'0,12584,32064,27,37,111,129,274,1383,1600,2141,7284,9172,13207,16519,18122,1

 ('14', '0,4,19,19079,42697,444,42748'),
 ('15', '0,4,27,80'),
 ('16',

'0,10,12,18,30,38,89,12570,19044,29319,35477,53,83,9745,15520,19010,30062,313

 ('17',

'0,19,26,28,95,128,134,150,6157,7284,12570,20016,20533,20599,42704,49678,53,2

 ('18', '0,4,16,30,89,2406,2411,12562'),
 ('19',

'0,14,17,439,1100,1694,1705,2413,2644,2646,2659,2678,3734,3926,7463,9892,1024
```

```
      ('20',
       '0,1,5,12846,22939,28193,29724,29791,30691,31232,34394,35589,44887,49574'),
      ('21'
```

```
# Perform the reducing step
reduced_data = mapped_data.reduceByKey(lambda x, y: x + y)


# Collect the results
results = reduced_data.collect()


# Print the results
for result in results:
    print(result)
```

```
Streaming output truncated to the last 5000 lines.
('35341', '35325,35326,35327,35329,35330,35331,35336,35339,35350,35353,35355,
('35342', '35325,35331,35333,35335,35339,35344,35346,35355,35361,35363,35371,
('35343', '35325,35351')
('35344', '35325,35326,35327,35328,35331,35336,35338,35339,35342,35333,35335,
('35346', '35325,35333,35335,35336,35339,35342,35344,35349,35355,35361,35363,
('35347', '35325,35327,35329,35330,35331,35339,35326,35341,35363,35385')
('35351', '35325,35343,35377')
('35352', '35325,35326,35327,35335,35336,35349,12015,35369,35408')
('35355', '35325,35326,35327,35331,35335,35336,35339,35341,35342,35346,35363,
('35356', '35325,35336,35338,35373')
('35357', '35325,35336')
('35358', '35325,35336,35344,35373')
('35359', '35325,35336,35353')
('35361', '35325,35326,35328,35331,35335,35336,35339,35341,35342,35344,35346,
('35362', '35325,35336,35389')
('35363', '35325,35326,35327,35328,35329,35330,35331,35335,35336,35339,35341,
('35366', '35325,35344,35372,35400,35402')
('35368', '35325,35336,35338,35341,35373,35386,35389,35403,35409')
('35369', '35325,35326,35335,35336,35339,35346,35349,35352,35333,35377')
('35370', '35325,35340')
('35371', '35325,35331,35335,35336,35338,35339,35341,35342,35344,35349,35363,
('35372', '35325,35336,35342,35366,35371,35373,35388,35389,35399')
('35373', '35325,35326,35328,35331,35335,35336,35338,35339,35341,35342,35344,
('35375', '35325,35339,580,16812,35336,35383,35408')
('35376', '35325,35337')
('35377', '35325,35326,35351,35369,35383,35384')
('35378', '35325,12491')
('35379', '35325,35336,35341,35349,35350,35373,35380,35382')
('35381', '35325')
('35383', '35325,35338,35339,35375,35377,3714,16812,35336,35411')
('35384', '35325,35377')
```

```
('35389', '35326,35331,35336,35338,35339,35341,35342,35344,35346,35355,35361,
('35398', '35326,35329,35330,35336,35338,35339,35342,35344,35346,35349,35373,
('35385', '35329,35330,35331,35332,35347,35325,35382')
('35403', '35331,35338,35345,35350,35361,35368,35373,35389,35325,35409,40086'
('35409', '35331,35335,35338,35339,35341,35346,35350,35355,35368,35389,35403,
('35386', '35332,35338,35368,13153,33299,35325,35371,35405,44761')
('35399', '35336,35344,35371,35373,35388,35325,35372,35402')
('35405', '35338,35386,35325')
('35395', '35340,35325')
('35388', '35344,35325,35372,35399,35402')
('35400', '35344,35366,35398,35325')
('35402', '35344,35366,35388,35399,35325')
('35387', '35348,35364,35325')
('35449', '35391,35435,35445,8735,35447,35450,35452,35455,35464,35465,35487,3
('35393', '35325,35348')
('35397', '35325')
('46497', '35401,46498,46499,46500,46501,46502,46503,46504,46506,46507,46508,
('46510', '35401,46497,46502,46527,46517')
('46524', '35401,46497,46499,46502,46507,46509,46511,46518,46520,46522,46523,
('46527', '35401,46497,46502,46507,46509,46510,46511,46512,46520,46521,46529,
('46528', '35401,46497,46502,46506,46507,46511,46518,46522,46523,46524,46525,
('46531', '35401,46497,46502,46507,46520,46527,46543,46544,46546')
('46532', '35401,46497,46502,46507,46518,46520,46523,46524,46528,46533,46536,
('46533', '35401,46497,46502,46511,46518,46521,46523,46524,46525,46528,46530,
('46543', '35401,46497,46502,46507,46511,46518,46522,46523,46524,46527,46529,
('46544', '35401,46497,46502,46507,46521,46527,46529,46531,46543,46545,46546,
('46546', '35401,46497,46502,46507,46527,46531,46543,46544,46545,46547')
('46548', '35401,46497,46502,46520,46522,46527,46529,46541,46542,46545,46547'
```

```
# Split the values for each key into individual elements
flat_data = reduced_data.flatMap(lambda x: [(x[0], value) for value in x[1].split(
flat_data.collect()
```

```
[('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'),
 ('0', '16'),
 ('0', '17'),
```

```
 ('0', '18'),
 ('0', '19'),
 ('0', '20'),
 ('0', '21'),
 ('0', '22'),
 ('0', '23'),
 ('0', '24'),
 ('0', '25'),
 ('0', '26'),
 ('0', '27'),
 ('0', '28'),
 ('0', '29'),
 ('0', '30'),
 ('0', '31'),
 ('0', '32'),
 ('0', '33'),
 ('0', '34'),
 ('0', '35'),
 ('0', '36'),
 ('0', '37'),
 ('0', '38'),
 ('0', '39'),
 ('0', '40'),
 ('0', '41'),
 ('0', '42'),
 ('0', '43'),
 ('0', '44'),
 ('0', '45'),
 ('0', '46'),
 ('0', '47'),
 ('0', '48'),
 ('0', '49'),
 ('0', '50'),
 ('0', '51'),
 ('0', '52'),
 ('0', '53'),
 ('0', '54'),
 ('0', '55'),
 ('0', '56'),
 ('0', '57'),
 ('0', '58'),
 ('0', '59'),
 ('0', '60')
```

```
# Create a tuple for each element with the key as the first element and the elemen
count_data = flat_data.map(lambda x: (x, 1))
count_data.collect()
```

```
[(('0', '1'), 1),
 (('0', '2'), 1),
 (('0', '3'), 1)
```

```
(('0', '3'), 1),
(('0', '4'), 1),
(('0', '5'), 1),
(('0', '6'), 1),
(('0', '7'), 1),
(('0', '8'), 1),
(('0', '9'), 1),
(('0', '10'), 1),
(('0', '11'), 1),
(('0', '12'), 1),
(('0', '13'), 1),
(('0', '14'), 1),
(('0', '15'), 1),
(('0', '16'), 1),
(('0', '17'), 1),
(('0', '18'), 1),
(('0', '19'), 1),
(('0', '20'), 1),
(('0', '21'), 1),
(('0', '22'), 1),
(('0', '23'), 1),
(('0', '24'), 1),
(('0', '25'), 1),
(('0', '26'), 1),
(('0', '27'), 1),
(('0', '28'), 1),
(('0', '29'), 1),
(('0', '30'), 1),
(('0', '31'), 1),
(('0', '32'), 1),
(('0', '33'), 1),
(('0', '34'), 1),
(('0', '35'), 1),
(('0', '36'), 1),
(('0', '37'), 1),
(('0', '38'), 1),
(('0', '39'), 1),
(('0', '40'), 1),
(('0', '41'), 1),
(('0', '42'), 1),
(('0', '43'), 1),
(('0', '44'), 1),
(('0', '45'), 1),
(('0', '46'), 1),
(('0', '47'), 1),
(('0', '48'), 1),
(('0', '49'), 1),
(('0', '50'), 1),
(('0', '51'), 1),
(('0', '52'), 1),
(('0', '53'), 1),
```

```
 (('0', '54'), 1),
 (('0', '55'), 1),
 (('0', '56'), 1),
 (('0', '57'), 1),
 (('0', '58'), 1),
 (('0', '59'), 1),
 (('0', '60'), 1)
```

```
# Count the occurrences of each element for each key
final_data = count_data.reduceByKey(lambda x, y: x + y)



# Collect the results
results = final_data.collect()


# Print the results
for result in results:
    print(result)
```

```
 (('48003', '48014'), 1)
 (('48003', '48019'), 1)
 (('48004', '47989'), 1)
 (('48004', '47991'), 1)
 (('48004', '47994'), 1)
 (('48004', '48000'), 1)
 (('48004', '48001'), 1)
 (('48004', '48002'), 1)
 (('48004', '47990'), 1)
 (('48004', '47992'), 1)
 (('48004', '47993'), 1)
 (('48004', '47995'), 1)
 (('48004', '47996'), 1)
 (('48004', '47997'), 1)
 (('48004', '47998'), 1)
 (('48004', '47999'), 1)
 (('48004', '48003'), 1)
 (('48004', '48005'), 1)
 (('48004', '48006'), 1)
 (('48004', '48007'), 1)
 (('48004', '48008'), 1)
 (('48004', '48009'), 1)
 (('48004', '48012'), 1)
 (('48004', '48013'), 1)
 (('48004', '48014'), 1)
 (('48005', '47989'), 1)
 (('48005', '47991'), 1)
 (('48005', '47993'), 1)
 (('48005', '47994'), 1)
 (('48005', '48000'), 1)
```

```
(('48005', '48000'), 1)
(('48005', '48001'), 1)
(('48005', '48002'), 1)
(('48005', '48004'), 1)
(('48005', '47990'), 1)
(('48005', '47992'), 1)
(('48005', '47995'), 1)
(('48005', '47996'), 1)
(('48005', '47998'), 1)
(('48005', '47999'), 1)
(('48005', '48003'), 1)
(('48005', '48006'), 1)
(('48006', '47989'), 1)
(('48006', '47991'), 1)
(('48006', '47994'), 1)
(('48006', '48000'), 1)
(('48006', '48001'), 1)
(('48006', '48002'), 1)
(('48006', '48004'), 1)
(('48006', '48005'), 1)
(('48006', '47990'), 1)
(('48006', '47992'), 1)
(('48006', '47993'), 1)
(('48006', '47995'), 1)
(('48006', '47996'), 1)
(('48006', '47998'), 1)
(('48006', '47999'), 1)
(('48006', '48003'), 1)
(('48006', '48008'), 1)
(('48006', '48012'), 1)
(('48006', '48013'), 1)
```

```
# Define a list of desired user IDs
user_ids = ["924", "8941", "8942", "9019", "9020", "9021", "9022", "9990", "9992",


# Use the map function and lambda to extract the user ID and friends, and swap the
swapped_data = data.map(lambda x: (x[0][1], x[0][0])).groupByKey()


# Use the flatMap function and lambda to generate the recommendations for each use
recommendations = swapped_data.flatMap(lambda x: ((x[0], y) for y in list(x[1]) if


# Take the top 10 recommendations for each user
top_10_recs = recommendations.groupByKey().mapValues(lambda x: list(x)[:10])
```

```
# Filter the RDD for the desired user IDs
desired_recs = top_10_recs.filter(lambda x: x[0] in user_ids)


# Access the recommendations
# Print the top 10 recommendations for each user
for rec in desired_recs.collect():
    print("User ID:", rec[0], "Recommendations:", rec[1])

sc.stop()
```

```
---------------------------------------------------------------------------
Py4JJavaError                             Traceback (most recent call last)
<ipython-input-42-972027640a3a> in <module>
      1 # Access the recommendations
      2 # Print the top 10 recommendations for each user
----> 3 for rec in desired_recs.collect():
      4     print("User ID:", rec[0], "Recommendations:", rec[1])
      5

                                  2 frames
/usr/local/lib/python3.8/dist-packages/py4j/protocol.py in
get_return_value(answer, gateway_client, target_id, name)
    324             value = OUTPUT_CONVERTER[type](answer[2:],
gateway_client)
    325             if answer[1] == REFERENCE_TYPE:
--> 326                 raise Py4JJavaError(
    327                     "An error occurred while calling {0}{1}{2}.\n".
    328                     format(target_id, ".", name), value)

Py4JJavaError: An error occurred while calling
z:org.apache.spark.api.python.PythonRDD.collectAndServe.
: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0
in stage 34.0 failed 1 times, most recent failure: Lost task 0.0 in stage
34.0 (TID 15) (deede2af1554 executor driver):
org.apache.spark.api.python.PythonException: Traceback (most recent call
last):
  File "/usr/local/lib/python3.8/dist-
packages/pyspark/python/lib/pyspark.zip/pyspark/worker.py", line 686, in main
    process()
  File "/usr/local/lib/python3.8/dist-
packages/pyspark/python/lib/pyspark.zip/pyspark/worker.py", line 676, in
process
    out_iter = func(split_index, iterator)
  File "/usr/local/lib/python3.8/dist-packages/pyspark/rdd.py", line 3472, in
pipeline_func
    return func(split, prev_func(split, iterator))
  File "/usr/local/lib/python3.8/dist-packages/pyspark/rdd.py", line 3472, in
pipeline_func
```

```
pipeline_func
    return func(split, prev_func(split, iterator))
  File "/usr/local/lib/python3.8/dist-packages/pyspark/rdd.py", line 540, in
func
    return f(iterator)
  File "/usr/local/lib/python3.8/dist-packages/pyspark/rdd.py", line 2665, in
combine
    merger.mergeValues(iterator)
  File "/usr/local/lib/python3.8/dist-
packages/pyspark/python/lib/pyspark.zip/pyspark/shuffle.py", line 253, in
mergeValues
    for k, v in iterator:
  File "/usr/local/lib/python3.8/dist-
packages/pyspark/python/lib/pyspark.zip/pyspark/util.py", line 81, in wrapper
    return f(*args, **kwargs)
  File "<ipython-input-34-a26d4d95eddd>", line 2, in <lambda>
IndexError: string index out of range

        at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonExcep

        at
org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:765)
        at
org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:747)
        at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRun

        at
org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37

        at
scala.collection.Iterator$GroupedIterator.fill(Iterator.scala:1211)
        at
scala.collection.Iterator$GroupedIterator.hasNext(Iterator.scala:1217)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at
org.apache.spark.shuffle.sort.BypassMergeSortShuffleWriter.write(BypassMergeS

        at
org.apache.spark.shuffle.ShuffleWriteProcessor.write(ShuffleWriteProcessor.sc

        at
org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:99)
        at
org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:52)
        at org.apache.spark.scheduler.Task.run(Task.scala:136)
        at
org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:5

        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1504)
```

```
        at
org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:551)
        at
java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecuto

        at
java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecut

        at java.base/java.lang.Thread.run(Thread.java:829)

Driver stacktrace:
        at
org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGSchedu

        at
org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.sc

        at
org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGSche

        at
scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)
        at
scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)
        at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)
        at
org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2607)
        at
org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGSch

        at
org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapte

        at scala.Option.foreach(Option.scala:407)
        at
org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scal

        at
org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGSchedu

        at
org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGSchedule

        at
org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGSchedule

        at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)
        at
org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:952)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:2228)
```

```
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:2249)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:2268)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:2293)
        at org.apache.spark.rdd.RDD.$anonfun$collect$1(RDD.scala:1021)
        at
org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151

        at
org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112

        at org.apache.spark.rdd.RDD.withScope(RDD.scala:406)
        at org.apache.spark.rdd.RDD.collect(RDD.scala:1020)
        at
org.apache.spark.api.python.PythonRDD$.collectAndServe(PythonRDD.scala:180)
        at
org.apache.spark.api.python.PythonRDD.collectAndServe(PythonRDD.scala)
        at
java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native
Method)
        at
java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAc

        at
java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(Delegating

        at java.base/java.lang.reflect.Method.invoke(Method.java:566)
        at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
        at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
        at py4j.Gateway.invoke(Gateway.java:282)
        at
py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
        at py4j.commands.CallCommand.execute(CallCommand.java:79)
        at
py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)
```