EE412 Foundation of Big Data Analytics, Fall 2021
HW1

Name: 고건호

Student ID: 20160025

Discussion Group (People with whom you discussed ideas used in your answers):

On-line or hardcopy documents used as part of your answers:

Answer to Problem 1

(1) the explanation about your algorithm

Read and Split

```
lines = sc.textFile(sys.argv[1])
friends = lines.map(lambda r: r.split('\t')) \
               .map(lambda r: (r[0], r[1].split(',')))
```

Convert each line into a pair.
Final form of RDD will be (User, [friend1, friend2, ⋯])

Map - Combination

```
def combination(r):
   user, friends = r
   comb_list = [
       (a, b)
       for i, a in enumerate(friends)
       for b in friends[i+1:]
   ]
   comb_list += [
       (user, a) if user < a else (a, user)
       for a in friends
   ]
   return comb_list

common_friends = friends.flatMap(lambda r: [(pair, r[0]) for pair in combination(r)])
```

Map friends list to a combination of all common friends. (L2-7)

- (1, [2, 3, 4]) → [(2, 3), (2, 4), (3, 4)]

Then add pairs with the user and each of his friends. This is to filter out users who are already friends in the next step. (L8-11)

- (1, [2, 3, 4]) → [(2, 3), (2, 4), (3, 4)] → [(2, 3), (2, 4), (3, 4), (1, 2), (1, 3), (1, 4)]

Use flatMap to apply “combination”. (L13)

- (1, [2, 3, 4]) → [((2, 3), 1), ((2, 4), 1), ((3, 4), 1), ((1, 2), 1), ((1, 3), 1), ((1, 4), 1)]

Reduce

```
common_pair = common_friends.groupByKey().mapValues(list)
common_pair = common_pair.filter(lambda r: r[0][0] not in r[1])
```

Reduce with “groupByKey”, and filter out keys that the first element of the key(r[0][0]) is in value(r[1]).

In mapping step, user himself was also added as a combination pair, so that when key and value contain same element, it means that two users are already friend.

- [… , ((2, 3), 1), ((2, 4), 1), ((3, 4), 1), ((1, 2), 1), ((1, 3), 1), ((1, 4), 1), …]
  → (groupByKey) → […, ((2, 3), [1, …]), ((2, 4), [1, …]), ((3, 4), [1, …]), ((1, 2), [1, …]), ((1, 3),
  [1, …]), ((1, 4), [1, …]), …]
  → (filter) → […, ((2, 3), [1, …]), ((2, 4), [1, …]), ((3, 4), [1, …]), ~~((1, 2), [1, …]), ((1, 3), [1, …]), ((1, 4), [1, …])~~, …]

(Why did I do this?)

Filtering out pairs that are already friends is the key process of this task. However, in the mapping stage, we don’t know if the pair of users are friends or not, because it may have to refer to information from other devices. So we should wait until the reducing stage. And adding himself as pairs, will give hint on the reducing stage.

Print

```
num_common = common_pair.map(lambda r: (r[0], len(r[1])))
top_common = num_common.top(10, key=lambda r: r[1])
for (a, b), n in sorted(top_common, key=lambda r: r[1], reverse=True):
   print(f"{a}\t{b}\t{n}")
```

In order to reduce computation, select the top 10 rows first, and then sort.

(2) the program’s elapsed time

22/09/18 20:59:41 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
22/09/18 20:59:43 INFO SparkContext: Running Spark version 3.1.2
22/09/18 20:59:44 INFO ResourceUtils:
==
22/09/18 20:59:44 INFO ResourceUtils: No custom resources configured for spark.driver.
22/09/18 20:59:44 INFO ResourceUtils:
==
22/09/18 20:59:44 INFO SparkContext: Submitted application: hw1_1.py
…
22/09/18 21:07:01 INFO DAGScheduler: Job 0 finished: top at /mnt/home/20160025/20160025_hw1/hw1_1.py:29, took 428.867828 s
18739 18740 100
31506 31530 99
31555 31560 96
31533 31559 96
31519 31568 95
31492 31511 95
31511 31533 95
31519 31554 95
31542 31568 95
31511 31556 95
22/09/18 21:07:01 INFO SparkUI: Stopped Spark web UI at http://eelab6.kaist.ac.kr:4042
22/09/18 21:07:01 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
22/09/18 21:07:01 INFO MemoryStore: MemoryStore cleared
22/09/18 21:07:01 INFO BlockManager: BlockManager stopped
22/09/18 21:07:01 INFO BlockManagerMaster: BlockManagerMaster stopped
22/09/18 21:07:01 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
22/09/18 21:07:01 INFO SparkContext: Successfully stopped SparkContext
22/09/18 21:07:02 INFO ShutdownHookManager: Shutdown hook called
22/09/18 21:07:02 INFO ShutdownHookManager: Deleting directory /tmp/spark-e96c9ba3-99d4-451b-afac-12a2a18a7f33
22/09/18 21:07:02 INFO ShutdownHookManager: Deleting directory /tmp/spark-ce587cc9-478e-4ffe-97cf-4ac1b46214fe
22/09/18 21:07:02 INFO ShutdownHookManager: Deleting directory /tmp/spark-e96c9ba3-99d4-451b-afac-12a2a18a7f33/pyspark-1fde3232-2e7a-42e2-8bfe-65756bf9c680

Answer to Problem 2

(a) [10 pts] Solve the following problems which are based on the exercises in the MMDS textbook

Properties

- Number of frequent items: N
- Number of frequent pair candidates: M+1,000,000

Memory

- Size of triangular-matrix: $2(N-1)(N-2)\ bytes$
- Size of item-item-count triples: $12(M+1,000,000)\ bytes$

Answer

$$min(2(N-1)(N-2),\ 12(M+1,000,000))$$

(b) [20 pts] Find frequent itemsets using the A-Priori algorithm

(i) The program's elapsed time

28.837302207946777s

Answer to Problem 3

(a) [10 pts] Solve the following problems which are based on the exercises in the MMDS textbook

(a) A 2-way AND construction followed by a 3-way OR construction.

p	1-(1-p^2)^3
0	0
0.2	0.115264
0.4	0.407296
0.6	0.737856
0.8	0.953344
1	1

p below 0.4 decreased, p above 0.4 increased

(b) A 3-way OR construction followed by a 2-way AND construction.

p	(1-(1-p^3)^2
0	0
0.2	0.238144
0.4	0.614656
0.6	0.876096
0.8	0.984064
1	1

Overall increased

(c) A 2-way AND construction followed by a 2-way OR construction, followed by a 2-way AND construction.

p	(1-(1-p^2)^2)^2
0	0
0.2	0.00614656
0.4	0.08667136
0.6	0.34857216
0.8	0.75759616
1	1

Overall decreased

(d) A 2-way OR construction followed by a 2-way AND construction, followed by a 2-way OR construction followed by a 2-way AND construction.

p	(1-(1-(1-p^2)^2)^2)^2
0	0
0.2	0.05875962
0.4	0.42435823
0.6	0.8341692
0.8	0.98774466
1	1

p below 0.4 decreased, p above 0.4 increased

(b) [30 pts] Find similar documents using minhash-based LSH

(i) The program's elapsed time

9.63370156288147s