Homework #2: Frequent Itemset

Due: February 27, Tuesday
100 points

In this homework, we will work with frequent itemset, and implement two algorithms – PCY and SON (With Apriori).

Task 1 (40 Points, PCY):

First, consider the PCY algorithm. Recall that the algorithm leverages the unused space in the first phase to build a frequent-bucket hash table which is then used in the second phase to reduce the number of candidate itemset to be considered. You are asked to implement the algorithm in a program named **FirstName_LastName_PCY.py** in Python without Spark, that could be executed as below.

Execution format:

> *python FirstName_LastName_PCY.py baskets.txt <a> <b> <N> <s> <output-dir>*

If you do use Spark (For reading file and doing any transformations etc.), please specify in Readme.txt, in which case, execution format will be

> *bin/spark-submit FirstName_LastName_PCY.py baskets.txt <a> <b> <N> <s> <output-dir>*

baskets.txt (Attached with the homework) is a text file that contains the baskets (a list of comma-separated item numbers) per line. For example:

```
1,2,3
1,2,5
1,3,4
2,3,4
1,2,3,4
2,3,5
1,2,4
1,2
1,2,3
```

where <a>, <b>, <N> are 3 integers that are used to build the hash function used in PCY.

> Hash function for a pair (i,j), where i < j, is h(i, j) = (a*i + b*j) % N, where N is the number of buckets in the hash table.

<s> is the threshold for the support count (so an itemset is frequent if its support count is $\geq s$).

The algorithm needs to output all the discovered frequent items (Singletons and pairs) in **frequentset.txt** and all the candidate itemset the hash table helps prune in the second phase of the algorithm in **candidates.txt** in the directory **<output-dir>**. In addition, it needs to output the false positive rate of the hash table (3 decimal places) on the console as below.

```
C:\spark\bin>spark-submit pcy.py baskets.txt 1 1 7 4 output-dir
False Positive Rate: 0.571

C:\spark\bin>
```

The output files should be created in **output-dir** like below

\spark\bin\output-dir

	Name	Date modified
	candidates.txt	2/5/2018 8:27 AM
	frequentset.txt	2/5/2018 8:26 AM

Output Format:

frequentset.txt should have frequent itemset in the ascending sorted order. Frequent singletons per each line followed by frequent pairs. Pairs (i,j) are tuples in python where i < j. For example **(Not an actual output)**:

```
1    1
2    2
3    3
4    (1,2)
5    (1,3)
6    (2,4)
7    (3,4)
```

candidates.txt will have the candidate pairs in the sorted order of tuples. For example:

```
candidates.txt
1    (1,4)
2    (2,3)
```

Task 2 (60 Points, SON & Apriori):

In this task, you are asked to implement SON algorithm **"FirstName_LastName_SON.py"** in **Apache Spark** using Python. Recall that given a set of baskets, SON algorithm divides them into chunks/partitions and then proceed in two stages. First, local frequent itemsets are collected, which form candidates; next, it makes second pass through data to determine which candidates are globally frequent.

You must implement Apriori algorithm for stage one to find local frequent itemset. Make use of monotonicity concept while generating candidate itemset. You may find Python itertools package to be useful in your implementation: https://docs.python.org/3/library/itertools.html#itertools.combinations

Requirements: You must use mapPartitions() method in Spark in stage one that invokes Apriori to find frequent itemsets in each partition. You will need mapPartitions() for your second stage to find the true frequent itemsets too.
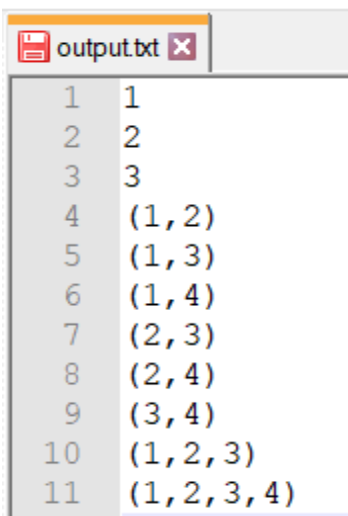
Execution Format:

bin/spark-submit FirstName_LastName_SON.py baskets.txt <support> output.txt

baskets.txt is exactly like the one shown in task 1.

<support> is a minimum support ratio in floating format like 0.1 (That is, for an itemset to be frequent, it should appear in at least 10% of the baskets). output.txt is name of the text file.

Output Format:

You should save all the frequent itemset into one file. Each line will have frequent items in sorted ascending order. Singletons followed by pairs, triples, quadruples etc. Frequent items except singletons need to be in tuple format, and are sorted in ascending order within themselves. For example: (1,2), (1,3,4) (1,4,5,8). Example output (**Not an actual output**):

```
output.txt
1    1
2    2
3    3
4    (1,2)
5    (1,3)
6    (1,4)
7    (2,3)
8    (2,4)
9    (3,4)
10   (1,2,3)
11   (1,2,3,4)
```

Submission on Blackboard:

A single zip file **FirstName_LastName_hw2.zip** containing below files:

candidates.txt	2/5/2018 9:36 AM	TXT File	
FirstName_LastName_PCY.py	2/5/2018 9:36 AM	PY File	
FirstName_LastName_SON.py	2/5/2018 9:36 AM	PY File	
frequentset.txt	2/5/2018 9:36 AM	TXT File	
output.txt	2/5/2018 9:36 AM	TXT File	
Readme.txt	2/5/2018 9:36 AM	TXT File	

Readme.txt will contain the instructions to run your both programs. Please mention Python and Spark version you are using. If you are using Spark for task 1, do mention. candidates.txt and frequents.txt are the output files from task 1, and output.txt is output file from task 2.

Grading Criteria:

a. If program does not run according to your Readme, there will be 50% penalty. In that case, grading will be done based on your output files submitted.
b. If output files format is not followed (Each frequent item in a single line in sorted order), there will be 20% penalty.
c. There will be 20 % penalty if you do not print false positive rate to the console for task 1.
d. Your programs will be run on additional baskets.txt files of similar size or less to verify the correctness. Test cases will be released after grading.
e. Assignment is due at 11:59 pm on 02/27 (3 weeks). Late homework will have 10% of penalty for every 24 hours that it is late. No credit will be given after 72 hours of its due time.

Important Note:

- **This assignment will surely take some time to implement. Start early!**
- **All the submitted work must be your own. Do not share the code with anyone! We will use Moss for plagiarism detection!**