Assignment 2: Frequent Itemsets

Write your own code!

For this assignment to be an effective learning experience, you must write your own code! I emphasize this point because you will be able to find Python implementations of most or perhaps even all of the required functions on the web. Please do not look for or at any such code! **Do no share code with other students in the class!!**

Here's why:

- The most obvious reason is that it will be a huge temptation to cheat: if you include code written by anyone else in your solution to the assignment, you will be cheating. As mentioned in the syllabus, this is a very serious offense, and may lead to you failing the class.
- However, even if you do not directly include any code you look at in your solution, it surely will influence your coding. Put another way, it will short-circuit the process of you figuring out how to solve the problem, and will thus decrease how much you learn.

So, just don't look on the web for any code relevant to these problems. Don't do it.

Please use Python 3.4

For this and future assignments, please use Python version 3.4. This will make it easier to automate the grading.

Submission Details

For each problem, you will turn in a python program.

To allow you to test your programs, sample data will be provided in the data folder, and the corresponding solutions will be provided for each problem in the solutions folder.

What you will turn in these two files:

- o Program that implements the a priori algorithm: `<firstname>_<lastname>_apriori.py`
- o Program that implements the PCY algorithm: `<firstName>_<lastname>_pcy.py`

Problem 1:

Implement Apriori Algorithm and print the frequent item sets of all possible sizes.

Input: The arguments of the program are:
1. input.txt: This file contains the transaction buckets.
- Each bucket corresponds to a single line in the file.
- Each bucket has a list of items separated by ","

2. Minimum Support: This is an integer.

Output: Please follow the format provided and make sure to output in the exact same way.
- The Output provided is for the given input file with a support 3

- Mismatch in the format might result in negative points.
- The order of the itemsets of a particular size can be different.
- Each itemset can have items in different order.

File Name: Please name your files as <firstname>_<lastname>_apriori.py.
Please note that the file name should be lower case and should match this format exactly.

Execution example: python3.4 pooja_anand_apriori.py input.txt 3
Where 3 is the minimum support

You can compare your solution to this problem with the corresponding output file output.txt

Problem 2:

Implement the PCY algorithm using a single hash.

The input and the output format are similar to part 1. You can use any hash function you choose, including the one we presented in class: convert each frequent itemset to an integer using some mapping of item to integer; then hash the resulting value based on the total number of buckets.

Input: The arguments of the program are:
1. input.txt: Same format as above
2. Support threshold: integer
3. Number of buckets: integer

Output: Same format as above.

File Name: Please name your files as <firstName>_<lastname>_pcy.py.

Execution Example: python3.4 pooja_anand_pcy.py input.txt 3 8
: Where 3 is the support, 8 is the number of buckets

You can compare your solution to this problem with the corresponding output file output.txt

General instructions:

To simplify the automated grading, please:

- Use Python3.4
- Please do not zip your files. You will lose points.
- Make sure your code compiles before submitting it.
- Make sure to include your <firstName>_<lastName> as part of the file names.