Project #1

Teams

This is a team project, meant to be completed in teams of three students. All students need to help the team, discuss any problems that might arise, and contribute to tasks in order to complete the project. Irrespective of who wrote a particular piece of code, all the teammates need to know what the code does and how different aspects of the project are implemented. After the completion of the project, there will be a 15-minute meeting/evaluation with all the teammates and the instructor or the TA/graders.

Team formation

You can create your own teams. If you do, send me an email with all the teammates and their emails. If you haven't found a team before Monday, I will assign you to a team on Monday (2/28).

Programming language

Teams can decide what programming language they will use. The suggested language is Python. If you choose a different one, contact the instructor to verify that it is alright to use it.

Objective

You are given part of the Million Playlist Dataset. This includes the songs appearing in different Spotify playlists. Your goal is to find all the sets of songs that are frequent, and the rules derived from them for the 2-itemsets and 3-itemsets that have enough confidence. In order to do so, you will need to implement from scratch the Apriori algorithm and all the necessary steps. For support counting, you can use dictionaries (hash tables).

Data files

You are given two different datasets (`playlists_small.txt`, `playlists_med.txt`) of different sizes. The smaller dataset will only be used for debugging purposes. In all your reports, you need to submit results using the dataset from the file `playlists_med.txt`. The format of these files is:

```
<playlistID>::<songID>
```

There is an additional file `song_info.txt` that includes information for each song. The songs are encoded in the same way for both `playlist*` files, so the songIDs from both files correspond to the same song. Each field is also separated with "::".

Code characteristics

Your code will read the input files (playlist + song info) and the min thresholds for support and confidence from the command line. I should be able to run your code by using the following command (in Python) for minsupp = 100, minconf = 0.8:

```
python myapriori.py playlists_med.txt 100 0.8
```

You will implement the Apriori algorithm to find the frequent itemsets and interesting rules. At every step/level of the algorithm during the frequent itemset generation, you need to print **on screen** the number of frequent itemsets found in the following format:

`<Level>-itemsets::<# frequent itemsets>` ---> example: `1-itemsets::100`

After the frequent itemset generation, you will create an output file (name: **`freq_<minsup>.txt`**) that has all frequent itemsets (one per line) in the following format:

`{songID,...,songID}::<support count>` ---> example: `{1,67}::10`

After generating the rules from the 2- and 3-itemsets, you will create an output file (name: **`rules_<minsup>_<minconf>.txt`**) that has all rules with high confidence (one per line) in the following format:

```
{songID,...,songID}::{songID,...,songID}::<support count>::<confidence with 4 digits>
```

---> example: `{1,4,67}::{3}::99::0.7890`

📜 Report 1 (Due March 9th)

1) Team members and assignment of responsibilities. Each teammate will lead one of the following responsibilities:
   a. Frequent Itemset Generation: Candidate generation & pruning
   b. Frequent Itemset Generation: Support counting & candidate elimination
   c. Rule generation
2) Write a paragraph for each step of the Apriori algorithm explaining how you will implement it.
3) Fill in the following table:

Number of transactions	
Number of items	
Width of longer transaction	
Width of shorter transaction	
Number of frequent 1-itemsets (Min support count threshold = 400)	
Highest support count of a 1-itemset	

4) Plot item support counts in **increasing** order (for support count > 50). Make sure that the plot is labeled properly. X-axis: “items”, Y-axis: “support count”, title: “Support counts in increasing order”. What can you notice?

For Report 1, you submit a PDF file with all the answers to the questions. As soon as you submit it, you can ask us to check if you are on the right track and help you make any corrections before the submission deadline.

📜 Report 2 (Due March 17th)

1) Create a pdf file, answering the following questions:

a. Run your code for each combination of the following sets of values for the minimum support, the minimum confidence, and option:

Min support count threshold = `minsup` = {40, 50, 60, 70, 80}

Min confidence threshold = `minconf` = {0.5, 0.6, 0.7}

For each value of `minconf`, generate two bar charts (or line charts) that will show (y-axis labels):

- the number of frequent itemsets that were found, and
- the number of high confidence rules that were generated (you will have two bars here, one for the rules of the 2-itemsets and one for the 3-itemsets)

for the different values of minimum support (x-axis labels). You need to either include a table with these values or print them on each bar in the figure (each bar is also annotated with the quantity that is plotting).

b. Based on itemsets and rules you found in the previous question, fill out this table for all the different min support count thresholds with the counts of the corresponding frequent itemsets and interesting rules found:

minsup	**1-itemsets**	**2-itemsets**	**3-itemsets**	**4-itemsets**	**Total (all itemsets)**
….	….				

	minconf=0.5		**minconf=0.7**		
minsup	**rules from 2-itemsets**	**rules from 3-itemsets**	**rules from 2-itemsets**	**rules from 3-itemsets**	**total**
….	….				

2) Submit your code, along with the output files produced when
   a. Minsup=30, minconf=0.7
   b. Minsup=60, minconf=0.5
   c. Minsup=90, minconf=0.5

   Also, include a readme file with all the needed information about how to run your code, and any additional information that you might want to include.