Jakub Klapacz
CS 412
Homework 3

Question 1

a. The apriori algorithm is an algorithm to extract frequent patterns in a database and prune all non-frequent patterns. The algorithm works by scanning a database for 1-item itemsets and prunes all non-frequent itemsets. In the next step the algorithm performs a self join on all the 1-item itemsets to generate all 2-item itemsets, scans the database for counts of these transactions and prunes all infrequent itemsets. This pattern of joining and pruning is repeated until all there are no more itemsets generated. This algorithm is important because it serves as a basis of determining what patterns are frequent in a database of transactions and narrows down the space of possible interesting patterns.
b. Apriori and FP Growth both extract frequent patterns from a database however they differ in some ways.

   Apriori repeatedly performs self joins and then must scan the database for a count of transactions matching a particular itemset. This means that a lot of work is redundant in performing an apriori pattern mining algorithm, in addition to many patterns being generated that are simply not in the database due to the fact that apriori must check all combinations of k+1 itemsets (if we are at a k itemset). This represents a combinatorial explosion of possible patterns and can become computationally intractable.

   FP Growth begins like Apriori – it must scan a database for all 1-item itemsets and orders them by descending frequency. Where the algorithms differ is that in a subsequent scan of the transactions in a database, elements from transactions are placed into a tree according to descending frequency (calculated from the first step). This means that only transactions that have actually occurred are considered in the search space of frequent patterns. Once the tree is built we determine which patterns are frequent however we no longer have to consider all combinations of increasing-sized itemsets cutting down computation significantly.

   Apriori has the adavantage of being much simpler to implement, however this is at a significant computational cost. FP Growth is more complex in its implementation, however computation does not become intractable even as the number of possible items in transactions grows by a significant amount, something that is a huge downside of Apriori.
c. The statement “For any associate rules X->Y, the support would always be larger than or equal to the confidence” is False. This is because support is calculated by:

$$Support(X \rightarrow Y) = P(X \cup Y) = \frac{\#\ of\ itemsets\ including\ both\ X\ and\ Y}{\#\ of\ itemsets\ total}$$

And confidence is calculated by:

$$Confidence(X \rightarrow Y) = P(Y|X) = \frac{\#\ of\ itemsets\ including\ both\ X\ and\ Y}{\#\ of\ itemsets\ including\ X}$$

From these two equations we can see that confidence can be equal to support only if every itemsets includes X, otherwise the number of itemsets including X will have to be smaller than the total number of itemsets making CONFIDENCE always greater than or equal to SUPPORT.

Question 2

a. To determine all frequent 3-itemsets and 2-itemsets, we must go through the pattern mining portion of the FP Growth algorithm. We begin by looking at the frequencies of 1-item itemsets and we start with the least frequent.

E	150
A	80
B	80
C	80
D	10

Because D was the least frequent 1-item itemset, we explore starting with D, we extract the base patterns associated with D, determine the conditional FP-tree for D, and the determine which patterns including D are frequent. We then repeat this in ascending frequency order for the other items:

Item	Base Pattern	Conditional FP-Tree	Frequent Patterns
D	{e,a,b:10}, {b:20}	-	-
C	{b,a,e:30}, {a,e:30}, {b:20}	{a,e:60}	{c,a:60}, {c,e:60}, {c,a,e:60}
B	{a,e:40}	-	-
A	{e:80}	{e:80}	{a,e:80}

As a result, there is only 1 frequent 3-itemset: {c,a,e} with a support of 60 and there are 3 2-itemsets: {c,a:60} {c,e:60}, {a,e:80}

b. C's conditional (projected) database is the one from the table above: {a,e} with a support of 60.

c. To determine strong association rules that meet the min_support of 60 and min_confidence of .7, we must consider all 4 patterns that we have found:

Rule	Support	Confidence (frac)	Strong
C -> A	60/190	60/80 = .75	Yes
C -> E	60/190	60/150 = .4	No
C -> AE	60/190	60/80 = .75	Yes

A -> E	80/190	80/150 = .53	No

As we can see, the rules C->A and C->AE meet minimum support and minimum confidence, however C->E and A->E do not, therefore the only strong rules are C->A and C->AE.

Question 3 / Question 4

The python code for my implementation of Questions 3 and 4 is included in my submission. Output files are named accordingly and are a result of running my script with minimum support = 20.

The instructions for running my program are included in the program and below:

```
python apriori.py [input file] [output files suffix] [minimum support]
```

Sample usage is below:

```
python apriori.py data-assign3/topic-4.txt 4 20
```

This creates the files “patterns/pattern-4.txt”, “closed/closed-4.txt”, and “max/max-4.txt”
Minimum support can either be a count (20 in this case) or a relative frequency [0..1]