FALL 2015 - CS 145 Homework Assignment #1 Solution

Q1: Level-wise Frequent Itemset Mining using Apriori and DIC (40pts)

1.1 Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n items. How many possible non-empty itemsets can be generated from I ? (2pts)

Ans. Maximum number of possible non-empty itemsets = $\sum_{i=1}^{n} \binom{n}{i} = 2^n - 1$ (2pts)

1.2 What is A*priori* property? How does it help in candidate pruning during the Apriori algorithm? (3pts)

Ans. Apriori property claims that any subset of a frequent itemset must be frequent. (1pt)
During the Apriori algorithm, we use this property to prune a k-itemset candidate, if any of its immediate subset ($(k-1)$-itemset) is not frequent. (2pt)

Consider the transactional database shown below to answer the following questions:

Transaction ID	***Items***
100	i,j,k,p,s,t
200	r,s,t
300	k,p,q,r,t
400	i,p,q,r
500	p,q,r,s
600	j,r,s,t

1.3 Assuming minimum support ≥ 3, find out **all** the frequent itemsets with their supports from the above database using the Apriori algorithm. Show every step of the computation. How many database scans did the Apriori algorithm require in this example? (20pts)

PASS #1	*Support of 1-candidates*	i:2, j:2, k:2, p:4, q:3, r:5, s:4, t:4 (2pts)
	1-Frequent	p, q, r, s, t (2pts)
	Self-join	pq, pr, ps, pt, qr, qs, qt, rs, rt, st (2pts)
	2-candidate (after prune)	pq, pr, ps, pt, qr, qs, qt, rs, rt, st (2pts)
PASS #2	*Support of 2-candidates*	pq:3, pr:3, ps:2, pt:2, qr:3, qs:1, qt:1, rs:3, rt:3, st:3 (2pts)
	2-Frequent	pq, pr, qr, rs, rt, st (2pts)
	Self-join	pqr, rst (2pts)
	3-candidate (after prune)	pqr, rst (all subsets are frequent) (2pts)
PASS #3	*Support of 3-candidates*	pqr:3, rst:2 (2pts)
	3-Frequent	pqr (1pt)
	Self-join	Ø
	4-candidate (after prune)	Ø

Ans. In this example, the Apriori required 3 complete database scans to generate all the frequent itemsets. 4th scan is not required as there are no more candidates to examine. (1pt)

1.4 Apply Dynamic Itemset Counting (DIC) to the above problem. You can assume that the transactions are read one by one, starting from the transaction ID 100. You **only** need to show **when an itemset becomes frequent** ***(during which scan and after reading which transaction)***. In this example, is there any frequent k-itemset that became frequent **before** the k^{th} scan? (15pts)

Frequent Itemset	***Becomes Frequent During***	
	Scan	***After Reading Transaction***
p	1	400
q	1	500
r	1	400
s	1	500
t	1	300
{p,q}	2	500
{p,r}	2	400
{q,r}	2	500
{r,s}	2	500
{r,t}	2	300
{s,t}	2	200
{p,q,r}	3	500

Ans. No, all frequent k-itemsets became frequent during the k^{th} scan. (12x2 + 1 = 15pts)

Note: Solution is based on the original DIC paper "*Dynamic Itemset Counting and Implication Rules for Market Basket Data*". But, full credit will be given even if counting of new frequent itemset is started from the current transaction, instead of the next one; whichever rule is decided, it has to be consistent throughout the solution.

Q2: Frequent Itemset Mining using FP-Tree (40pts)

Use the *same transactional database and minimum support constraint as in Q1* to answer the following questions:

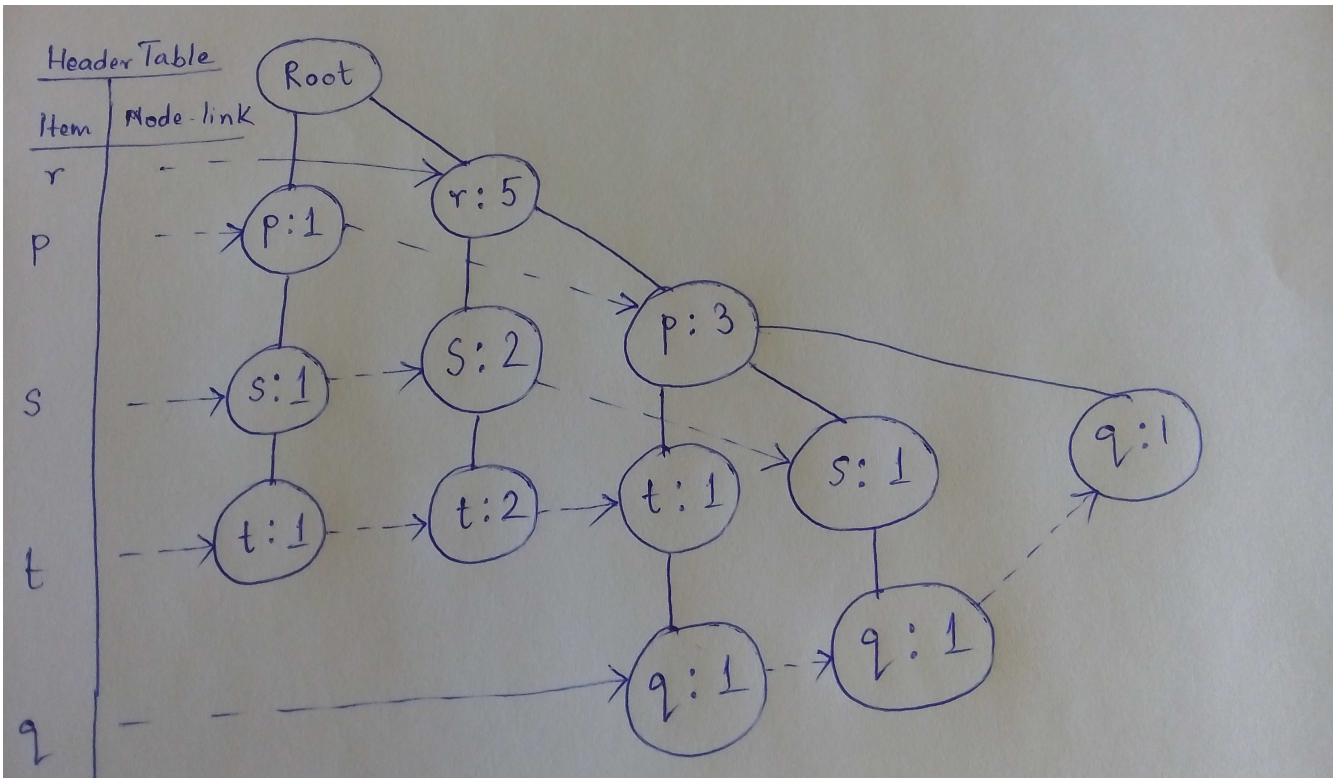
2.1 Build the FP-tree step by step for the given transactional database. While ordering the frequent items, use alphabetical order to break ties between the items having same support. (20pts)

Scan #1

Find Support of 1-items: i:2, j:2, k:2, p:4, q:3, r:5, s:4, t:4 (3pts)
Find 1-frequent items: p:4, q:3, r:5, s:4, t:4 (minimum support ≥ 3) (3pts)
Order among 1-frequent items: r, p, s, t, q (descending order of support; ties broken according to alphabetical order) (2pts)

Scan #2

For each transaction, remove infrequent items, arrange the frequent items by the new order and insert it into the FP-tree. The final FP-tree at the end of the scan is shown below. (12pts)



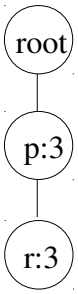
2.2 How many database scans did you need to build the FP-tree in this example? In general for any transactional database, how does the number of scans required to build a FP-tree compare to the Apriori algorithm? (3pts)

Ans. For any transactional database, building a FP-tree always require 2 database scans. However, the Apriori requires $\Theta(k)$ database scans, where k is the max. length of a frequent itemset in the database under the support constraint.

2.3 Use this FP-tree to compute all the frequent itemsets that contain '*q*' with their respective supports. Show every step of the computation (including all the projected databases and conditional FP-trees at each step)? (15pts)

Ans. First Frequent itemset **q:3** and, (3pts)

q-projected DB: {rpt:1, rps:1, rp:1}. (3pts)
Local frequent items: r, p (3pts)
Conditional FP-tree built on *q*-projected DB (shown without the header table): (3pts)



This is a single branch, so no need to create conditional FP-trees any more.
Simply enumerate all combinations to find the frequent itemsets from this path: **qp:3, qr:3, qpr:3** (1x3=3pts)

2.4 Without making any changes to the original FP-tree (computed in 2.1), is it possible to mine ***all*** the frequent itemsets for (i) minimum support ≥ 4, and (ii) minimum support ≥ 2? Explain your answer. (2pts)

Ans. (i) We can always mine all the frequent items from a FP-tree built using lower minimum support constraint, by selectively projecting databases from items, which remains frequent under the new constraint. For example, with minimum support ≥ 4, we do not need to examine *q*-projected database. (1pt)

(ii) However, we cannot mine all the frequent items from a FP-tree built using higher minimum support constraint, as it does not consider previously infrequent items, which became frequent under the new support constraint. For example, this FP-tree does not contains items *i*, *j* and *k* which are frequent for minimum support ≥ 2. (1pt)

Q3: Condensed Representations (20pts)

3.1 What are closed frequent itemsets? List them out from the frequent itemsets computed in Q1.3 (3pts)

Ans. A frequent itemset becomes closed frequent it has no superset with the same support. (1pt)
See below for the list of closed frequent itemsets.

3.2 What are maximal itemsets? List them out from the frequent itemsets computed in Q1.3 (3pts)

Ans. A frequent itemset is defined to be a maximal itemset if it has no superset which is also frequent. (1pt)

Closed Frequent Itemset (2pts)	**Is Maximal?** (2pts)
pqr	Yes
st	Yes
rs	Yes
s	No
rt	Yes
t	No
p	No
r	No

3.3 Prove that maximal itemsets $\subseteq$ closed frequent itemsets. (6pts)

Ans. We can prove this by contradiction, as shown below.

Let us assume there exists at least one maximal itemset X which is not closed.
Since, X is not closed, there must exist a superset Y $\supset$ X, such that support(Y) = support(X).
Now, since X is frequent (by the definition of maximal itemset), Y must also be frequent.
However this leads to a contradiction that X is a maximal itemset, since its superset Y is also frequent.
Therefore, every maximal itemset must be closed.
This implies maximal itemsets $\subseteq$ closed frequent itemsets (***Q.E.D.***)

3.4 Consider the following itemsets and their supports:
{X,A}:5, {X,Y,Z}:3, {X,B}:4,

(a) Assume the above itemsets are *closed* and *frequent*. Using this information, find out **all** the frequent itemsets with their respective supports, if possible? Explain your answer. (4pts)

Ans. One can always generate all the frequent itemsets with their corresponding supports from closed frequent itemsets, as shown below.

Process the itemsets in descending order of support and generate from it every possible non-empty *unique* subset (which has not been previously generated) (2pts).

{X,A}:5 → {X}:5,{A}:5 and {X,A}:5
{X,B}:4 → {B}:4 and {X,B}:4 ({X} was already generated before)

{X,Y,Z}:3 → {Y}:3, {Z}:3, {X,Y}:3, {X,Z}:3, {Y,Z}:3, {X,Y,Z}:3 ({X} is skipped) (2pts)

(b) Assume the above itemsets are *maximal*. Using this information, find out **all** the frequent itemsets with their respective supports, if possible? Explain your answer. (4pts)

Ans. We can always get all the frequent itemsets from the maximal ones by generating all of their possible subsets. However, we cannot always get the respective supports. At best, we can ascertain the lower bound of the supports (2pts).

{X,A}:5 → support({X}) ≥ 5,{A} ≥ 5
{X,B}:4 → support({B}) ≥ 4
{X,Y,Z}:3 → support({Y}) ≥ 3, support({Z}) ≥ 3, support({X,Y}) ≥ 3, support({X,Z}) ≥ 3, support({Y,Z}) ≥ 3 (2pts)