Yuan Wang
440 Data Mining
Prof. Jiebo Luo
Homework 3

6.1

Algorithm: Determine if an itemset is frequent.
Input: *C*, set of all frequent closed itemset along with their support counts; test itemset, *X*.
Output: Support of X if it is frequent, otherwise -1.
Method:

```
s = none
for l in C:
   if X in l and len(l) < len(s) or s = none:
      s=l
if s =! none:
      return support(s)
   else:
      return -1
```

6.3

a.

Suppose *s* is the frequent itemset, *min_sup* is the minimum support. *D* is task relevant data, a set of data transactions. And $|D|$ is the number of transactions of *D*.
So, $support\_count(s) \geq min\_sup \times |D|$
If *s'* is a nonempty subset of *s*. So, any transaction itemset contains *s* will also contain itemset *s'*.
Thus, $support\_count(s') \geq support\_count(s) \geq min\_sup \times |D|$.
Therefore, *s*' is a frequent itemset as well.

b.

From question (a) we know that $support\_count(s') \geq support\_count(s)$, so
Support (s') = $\frac{support_{count}(s')}{|D|} \geq$ Support (s) = $\frac{support_{count}(s)}{|D|}$. Therefore, the support of any nonempty subset s ′ of itemset *s* must be as great as the support of *s*.

c.

s is a subset of *l*, then confidence $(s \Rightarrow (l - s)) = \frac{support\ (l)}{support\ (s)}$
s' is a subset of s, then confidence $(s' \Rightarrow (l - s')) = \frac{support\ (l)}{support\ (s')}$
Because $support\ (s') \geq Support\ (s')$, so $confidence\ (s' \Rightarrow (l - s')) \leq confidence\ (s \Rightarrow (l - s))$.
Therefore, the confidence of the rule "$s' \Rightarrow (l - s')$" cannot be more than the confidence of the rule "$s \Rightarrow (l - s)$".

d.
Proof by Contradiction: Assume that the itemset is not frequent in any of the partitions of D.

Suppose *F* is any frequent itemset. *D* is task relevant data, a set of data transactions. *C* is the total number of transactions in *D*. *A* is the total number of transactions in *D* containing the itemset F. So, $A = C \times min\_sup$.
In the beginning, we suppose *F* is not frequent in any of partitions of *D*. So, $A \leq C \times min\_sup$.
This contradicts with what we defined that *F* is frequent itemset.
Therefore, any itemset that is frequent in *D* must be frequent in at least one partition of *D*.

6.4
Because C_k is generated from 2 itemsets from L_{k-1}, so these 2 subsets of C_k should not need to check. You only need to check the rest of *length-(k − 1)* subsets of C_k, that is total of *k-2* subsets.

One possible improvement is to pass l_1 and l_2, and prevent searching L_{k-1} for these two subsets because they are frequent itemsets.

6.5
The method in Section 6.2.2 generates all the nonempty subsets of a frequent itemset *l* and then tests all of them for potential rules. It may generate and test many unnecessary subsets. The proposed method as below only generates and tests the necessary subsets.

- If a subset *x* of length *k* does not meet the minimum confidence, then there is no need to generate any of its nonempty subsets as their respective confidences will never be greater than the confidence of *x*. (6.3b)
- If *x* meets the minimum confidence then we generate and test its *(k − 1)-subsets*. It will start with the *(n − 1)-subsets* of an n-itemset and progressively work our way down to the *1-subsets*.

6.6
a.
C1

Itemset	Sup.Count		min_sup = 60%	3
M	3		min_conf=80%	4
O	3			
N	2			
K	4			
E	4			
Y	3			
D	1			
A	1			
U	1			
C	2			
I	1			

Apriori:

L_1 = { E, K, M, O, Y } #after delete infrequent subset

C_2 = { EK, EM, EO, EY, KM, KO, KY, MO, MY, OY }
L_2 = { EK, EO, KM, KO, KY }

C_3= { EKO }
L_3 = { EKO}

C4 = ∅
L4 = ∅

Results of frequent itemsets: { E, K, M, O, Y, EK, EO, KM, KO, KY, EKO }

FP-growth

L1	
E	4
K	4
M	3
O	3
Y	3

Item	Conditional Pattern Base	Conditional FP-Tree	Frequent Patterns Generated
Y	{E, K, M, O : 1}, { E,K,O: 1 }, { K,M: 1 }	K:3	{K,Y : 3}
O	{ E,K,M: 1 } , { E,K: 2 }	E: 3, K: 3	{ E,K,O: 3 } , { K,O: 3 } , { E,O: 3 }
M	{ E,K: 2 } , { K: 1 }	K: 3	{ K,M: 3 }
K	{ E: 4 }	E: 4	{ E,K: 4 }

Results of frequent itemsets:
{ { E: 4 } , { K: 4 } , { M: 3 } , { O: 3 } , { Y: 3 } , { K,Y: 3 } , { E,K,O: 3 } , { K,O: 3 } , { E,O: 3 } , { K,M: 3 } , { E,K: 4 } }

Because FP-growth can mine in the conditional pattern bases, it reduces the size of the data sets to be searched. So especially in big data set, FP-growth is more efficient than APriori.

b.
∀X ∈ transaction, buys(X, E)∧buys(X, O) ⇒ buys(X, K) [60%, 100%]
∀X ∈ transaction, buys(X, K)∧ buys(X, O) ⇒ buys(X, E) [60%, 100%]

6.11
Apriori:
When finding the multiple occurrences of items, we need to treat each item with different count value as different items, then check if the minimal support is met. For example, B:1 and B:2, that is, B with single count or 2 counts, as different items. Then we construct frequent 2-itemsets, 3-itemsets, etc. We need to check the generated itemsets to decide whether they are frequent or not.

FP Growth:

When use FP growth method, we also need to consider the frequency of the generated itemset or itemsets. For example, when we do projected DBs, we need to make sure each item is associated with different counts.