Liqi Zhu Assignment 3

> 6.1 Suppose you have the set C of all frequent closed itemsets on a data set D, as well as the support count for each frequent closed itemset. Describe an algorithm to determine whether a given itemset X is frequent or not, and the support of X if it is frequent.

Assume s=∅
for each itemset in C, I ∈ C
if X ⊂ I and length(X)<length(I)
s=I
return support(s)
if s=∅
return 0

> 6.3 The Apriori algorithm makes use of prior knowledge of subset support properties.
> (a) Prove that all nonempty subsets of a frequent itemset must also be frequent.
> (b) Prove that the support of any nonempty subset s' of itemset s must be at least as great as the support of s.
> (c) Given frequent itemset l and subset s of l, prove that the confidence of the rule "$s' \Rightarrow (l - s')$" cannot be more than the confidence of " "$s \Rightarrow (l - s)$ " where s' is a subset of s.
> (d) A partitioning variation of Apriori subdivides the transactions of a database D into n nonoverlapping partitions. Prove that any itemset that is frequent in D must be frequent in at least one partition of D.

(a)Assume itemset s, nonempty subsets s'. Assume there's a set of data transaction P, |P|equals the number of transactions.
Since $support\_count(s) \geq minsup \times |P|$
Any transaction includes s also includes s'
$support\_count(s') \geq support\_count(s)$

(b)

$$support\_count(s') \geq support\_count(s)$$

$$\frac{support\_count(s')}{|P|} \geq \frac{support\_count(s)}{|P|}$$

$$support(s') \geq support(s)$$

(c)

$$support\_count(s') \geq support\_count(s)$$

$$\frac{support(l)}{support(s')} \leq \frac{support(l)}{support(s)}$$

$$confidence(s' \Rightarrow (l - s')) \leq confidence(s' \Rightarrow (l - s)$$

proved

(d)Assume the itemset is not frequent in any partition of D. Let F be any frequent itemet. Let A represents the number of transactions including F.
$A \geq minsup \times |D|$
$(a_1 + a_2 + a_3 + ... + a_n) \geq (|d_1| + |d_2| + |d_3| + ... + |d_n|) \times minsup$
since F is assumed not frequent
$a_n \leq |d_n| \times minsup$
which conflicts
so it's proved

> 6.4 Let c be a candidate itemset in C_k generated by the Apriori algorithm. How many
> length-(k-1) subsets do we need to check in the prune step? Per your previous answer, can you give an improved version of procedure has infrequent subset in Figure 6.4?

```
for each itemset l1 in  L_{k-1}
    for each itemset l2 in  L_{k-1}
```

```
        if (l1[1]=l2[1])and(l1[2]=l2[2])and...and(l1[k-1]<l2[k-1])
            c=l1|l2
            add c to Ck
    for each subset s of c
        if s not in L_{k-1}
            delete c
return Ck
end
```

Number of subsets we need to check: $length(Ck) \times k$

modificatoin of the has_infrequent_subset

Since there's multible methods of improving Apriori, such as hashing table, Transaction reduction, Partitioning, Sampling and Dynamic itemset counting.

The first improvement I tried is based on the concept that every itemset in L_k has all its nonempty subset in C_{k-1}, and there will be k subsets. I choose the different elements in each two items to form a detect item. As long as there is k-2 (because of the item I choose) items include the detect item. The union set of the two item must be a frequent item. Also using this method can directly pick the frequent Lk and saves the time of deleting infrequent c s. This method is applicable when k>2.

```
for each itemset l1 in  L_{k-1}
    for each itemset l2 in  L_{k-1}
        if (l1[1]=l2[1])and(l1[2]=l2[2])and...and(l1[k-1]<l2[k-1])
            t=(l1-l1&l2)|(l2-l1&l2)
            c=l1|l2
            add c to Ck
              s=0
                for i in range(1,count(L_{k-1})):
                    if t in L_{k-1}[i]
                      s++
                        if s=k-2
                        add c to Lk
return Ck,Lk

end
```

The second improvement I tried is based on the hashing table method. The only difference is the size of the candidate k-itemsets. The code of the last part may be the same.

> 6.5 Section 6.2.2 describes a method for generating association rules from frequent itemsets. Propose a more efficient method. Explain why it is more efficient than the one proposed there. (Hint: Consider incorporating the properties of Exercises 6.3(b), (c) into your design.)

```
i:set of frequent items
s:subset of i
min_con: minimum confidence threshold
for each i
    rule(s,i,min_con):
    k=length(s)
    if k>1
        for each length(k-1) subset m of s
            if(support_count(i)/support_count(m)=min_con)
                print ("rule:"rule)
```

It tests only necessary subsets.

> 6.6 A database has five transactions. Let min sup D 60% and min conf D 80%.
> (a) Find all frequent itemsets using Apriori and FP-growth, respectively. Compare the efficiency of the two mining processes.
> (b) List all the strong association rules (with support s and confidence c) matching the following metarule, where X is a variable representing customers, and itemi denotes variables representing items (e.g., "A," "B,"):

(a)
Apriori:
min_sup=60 support=3

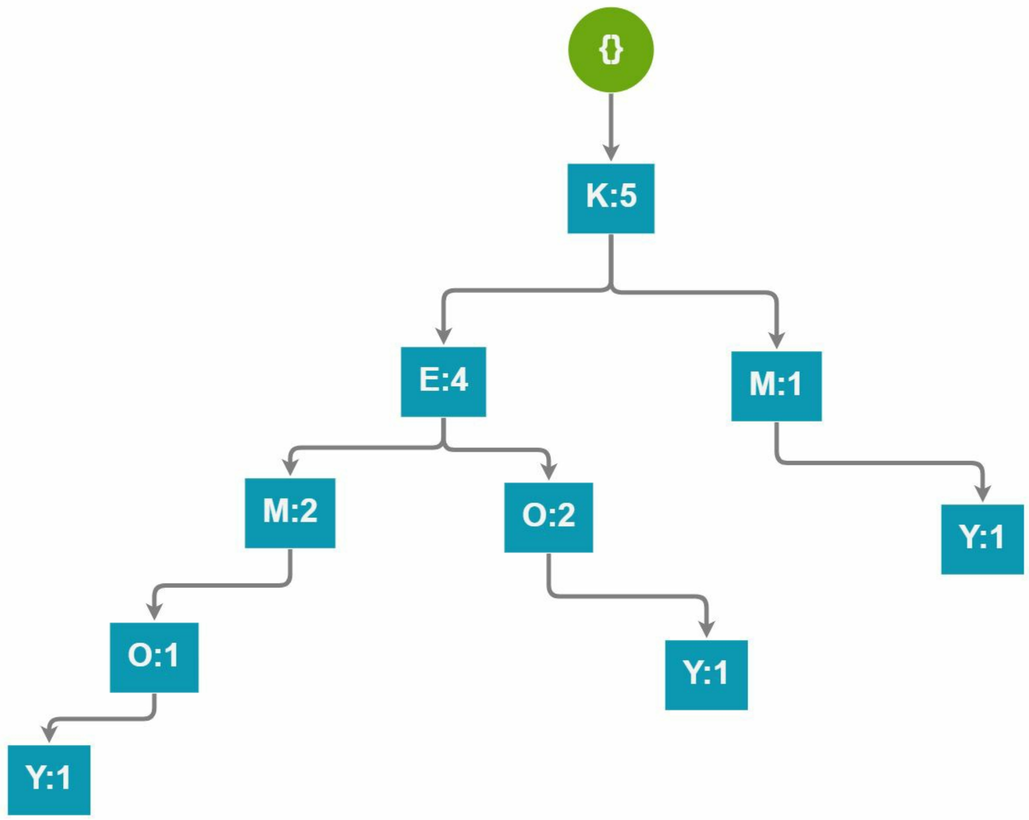
C1		**L1**		**C2**		**L2**		**C3**		**L3**	
m	3	m	3	mo	1	mk	3	oke	3	oke	3
o	3	o	3	mk	3	ok	3	key	2		

C1		L1		C2		L2		C3		L3	
n	2	k	5	me	2	oe	3				
k	5	e	4	my	2	ke	4				
e	4	y	3	ok	3	ky	3				
y	3			oe	3						
d	1			oy	2						
a	1			ke	4						
u	1			ky	3						
c	2			ey	2						
i	1										

FP Growth

ordered items
k,e,m,o,y
k,e,o,y
k,e,m
k,m,y
k,e,o

FP Tree:



Comparison of efficiency: Apriori has self-join process with multiple scans while the fp growth builds the tree with one scan, and also in a large itemset, the scan of Aprior can be time consuming and expensive.

> 6.11 Most frequent pattern mining algorithms consider only distinct items in a transaction.
> However, multiple occurrences of an item in the same shopping basket, such as four cakes and three jugs of milk, can be important in transactional data analysis.
> How can one mine frequent itemsets efficiently considering multiple occurrences of items? Propose modifications to the well-known algorithms, such as Apriori and
> FP-growth, to adapt to such a situation.

Assume there's a combined item consistes of an item and its occurrence count such as $(i, count)$, in the first scan for single frequent, if it meets the minimum threshold, $count = maxcount$, then apply the method of Apriori or FP-Growth to find k-itemsets for $count$ from 1 to $maxcount$. Consider the combined item with different counts as seperate item. The first scan C1 should include the combined item such as (A,1),(A,2),(B,1),etc. The other steps are the same.