Name: Garvit Rajkumar Pugalia
UID: 504628127

Homework 5

Question 1:

(a) Using the transaction database, we can create a set of 1-itemsets. Consequently, we can check for min_support criterion and create a set of “frequent” 1-itemsets:

Itemset	Support Count
{a}	6
{b}	8
{c}	6
{d}	4
{e}	2
{i}	1
{j}	1
{k}	1

Itemset	Support Count
{a}	6
{b}	8
{c}	6
{d}	4
{e}	2

By combining the “frequent” 1-itemsets (union), we can create a list of “frequent” 2-itemsets. Note that there is no need for pruning as all single items (i.e. all subsets of 2-itemset) are already frequent.

Itemset	Support Count
{a b}	4
{a c}	4
{a d}	2
{a e}	2
{b c}	4
{b d}	4
{b e}	2
{c d}	1
{c e}	1
{d e}	0

Itemset	Support Count
{a b}	4
{a c}	4
{a d}	2
{a e}	2
{b c}	4
{b d}	4
{b e}	2

Name: Garvit Rajkumar Pugalia
UID: 504628127

Now, we can form 3-itemsets from the previous result. However, we need to prune the sets formed such that each non-zero subset of the itemset is also “frequent”. The non-pruned list will look like:

C = {{a b c}, {a b d}, {a b e}, {a c d}. {a c e}, {a d e}, {b c d}, {b c e}, {b d e} }

During the pruning step, we realize that some sets (e.g. {a c e}) have non-zero subsets (e.g. {c e}) that are not frequent. These sets can be removed to obtain:

Itemset	Support Count
{a b c}	2
{a b d}	2
{a b e}	2
{a c d}	*prune*
{a c e}	*prune*
{a d e}	*prune*
{b c d}	*prune*
{b c e}	*prune*
{b d e}	*prune*

Itemset	Support Count
{a b c}	2
{a b d}	2
{a b e}	2

These cannot be combined to form 4-itemsets since all possible unions will be pruned. Therefore, we get the final list of “frequent” patterns as:

{a}	{a b}	{a b c}
{b}	{a c}	{a b d}
{c}	{a d}	{a b e}
{d}	{a e}	
{e}	{b c}	
	{b d}	
	{b e}	

Name: Garvit Rajkumar Pugalia
UID: 504628127

(b) To draw the FP-tree, we first need the “frequent” 1-itemsets. We can use the result from part (a) and sort in descending order of support count to get:

Itemset	Support Count
{b}	8
{a}	6
{c}	6
{d}	4
{e}	2

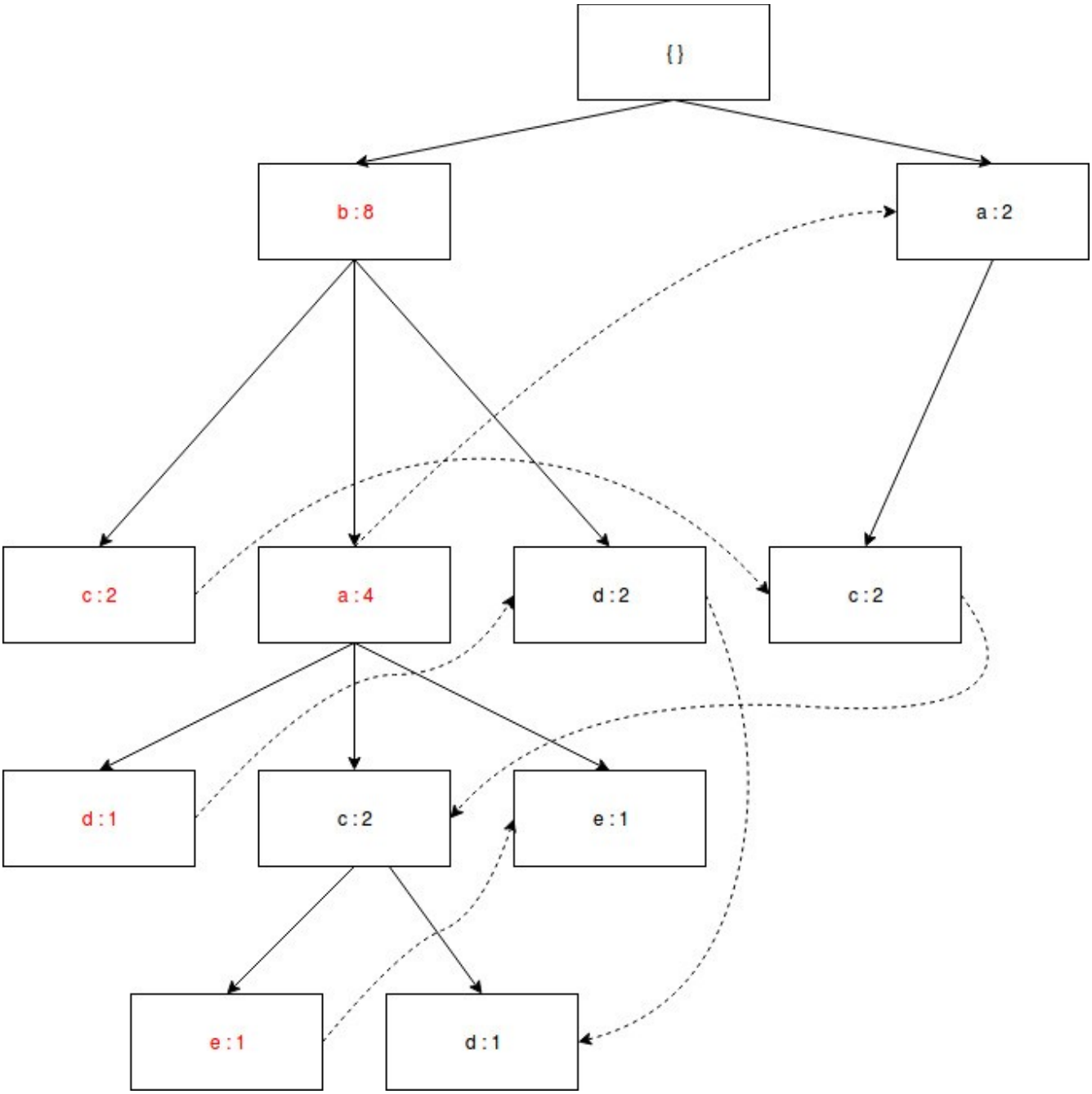
This gives us the F-list = b – a – c – d – e

Now, we can order the original transaction database using the F-list priority:

TID	Items	Ordered Frequent Items
1	*b, c, j*	*b, c*
2	*a, b, d*	*b, a, d*
3	*a, c*	*a, c*
4	*b, d*	*b, d*
5	*a, b, c, e*	*b, a, c, e*
6	*b, c, k*	*b, c*
7	*a, c*	*a, c*
8	*a, b, e, i*	*b, a, e*
9	*b, d*	*b, d*
10	*a, b, c, d*	*b, a, c, d*

Name: Garvit Rajkumar Pugalia
UID: 504628127

Now, we can go through each transaction and add the items to the FP tree using the FP-tree construction algorithm. The final tree can be seen here, with the first named-node (i.e. head) highlighted in red for each item:



Name: Garvit Rajkumar Pugalia
UID: 504628127

(c) By accumulating all of transformed prefix paths of item *d*, we can get the following conditional pattern base = ba:1, b:2, bac:1

We can construct the FP-tree for the frequent items in the pattern base to get the *d*-conditional FP-tree. This can be done by only looking at transactions with item *d* and removing *d* from every transaction:

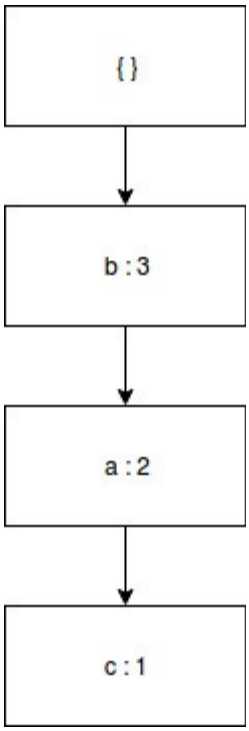
TID	Items
~~1~~	~~*b, c, j*~~
2	*a*, *b*, *d*
~~3~~	~~*a, c*~~
4	*b*, *d*
~~5~~	~~*a, b, c, e*~~
~~6~~	~~*b, c, k*~~
~~7~~	~~*a, c*~~
~~8~~	~~*a, b, e, i*~~
9	*b*, *d*
10	*a*, *b*, *c*, *d*

Therefore, we need to make a tree from:

TID	Items	Ordered Frequent Items
1	*a*, *b*	*b*, *a*
2	*b*	*b*
3	*b*	*b*
4	*a*, *b*, *c*	*b*, *a*, *c*

Name: Garvit Rajkumar Pugalia
UID: 504628127

The final conditional FP-tree will then be:



(d) Using the conditional FP-tree, we can get the following frequent patterns:

{ {d}, {b d}, {a d}, {b a d} }

After comparing with part (a), we observe that the conditional FP-tree for item *d* gives the “frequent” patterns that end with *d*.

Name: Garvit Rajkumar Pugalia
UID: 504628127

Question 2:

After running *python2.7 apriori.py*, we get the following output:

min_support: 50 min_conf: 0.25
item: "Wicked Spoon","Holsteins Shakes & Buns" , 51.000
item: "Wicked Spoon","Secret Pizza" , 52.000
item: "Wicked Spoon","Earl of Sandwich" , 52.000
item: "The Cosmopolitan of Las Vegas","Wicked Spoon" , 54.000
item: "Mon Ami Gabi","Wicked Spoon" , 57.000
item: "Bacchanal Buffet","Wicked Spoon" , 63.000

---------- RULES:
Rule: "Secret Pizza" ==> "Wicked Spoon" , 0.256
Rule: "The Cosmopolitan of Las Vegas" ==> "Wicked Spoon" , 0.277
Rule: "Holsteins Shakes & Buns" ==> "Wicked Spoon" , 0.315
106.425302029 sec

The rules relate different food joints, in a “people who went to A, also went to B” manner.

After researching the three places, the initial observation was that all three eateries are situated in Las Vegas. However, after looking into details, we find that all three places (Wicked Spoon, Secret Pizza, and Holsteins Shakes & Buns) are located within The Cosmopolitan of Las Vegas. The Yelp mechanism probably forms an association between the places because, in such a touristy area, people tend to visit multiple restaurants within the famous Cosmopolitan.

Name: Garvit Rajkumar Pugalia
UID: 504628127

Question 3:

	Beer	No Beer	Total
Nuts	150	700	850
No Nuts	350	8800	9150
Total	500	9500	10000

Using the contingency table, we can calculate many correlation metrics.

(a) Confidence can be calculated in two different ways:

$$c(\text{Beer} \rightarrow \text{Nuts}) = \text{sup(Beer and Nuts)} / \text{sup(Beer)} = 150 / 500 = \mathbf{30\%}$$
$$c(\text{Nuts} \rightarrow \text{Beer}) = \text{sup(Beer and Nuts)} / \text{sup(Nuts)} = 150 / 850 = \mathbf{17.6\%}$$

Lift is just the ratio of confidence to support of the item in the rule consequent. Clearly, the two confidence values above will provide the same Lift value given as:

$$\text{Lift} = \text{sup(Beer and Nuts)}/ \text{sup(Beer)}*\text{sup(Nuts)} = 150/10000 / (500/10000 * 850/10000) = \mathbf{3.529}$$

Finally, the all_confidence metric is defined as the minimum of $c(A \rightarrow B)$ and $c(B \rightarrow A)$, which gives us:

$$\text{all\_confidence} = \min(30, 17.6) = \mathbf{17.6\%}$$

(b) These metrics can be used to analyze the association between buying beer and buying nuts. The confidence levels are relatively low, which implies that there is no relation between buying beer and buying nuts. This is also supported by the value of all_confidence.

However, if we look at the Lift, we see that the value is much greater than 1. This implies that occurrence of the rule body has a strong, positive effect on the occurrence of the rule head. The confidence values don’t take the total number of non-examples into consideration (no beer and no nuts are bought). In the table, clearly, this is a majority!

By taking the total number of transactions into account, the lift value indicates (in the big picture) that buying beer and buying nuts are strongly related.

Name: Garvit Rajkumar Pugalia
UID: 504628127

Question 4:

(a) An event/element is a non-empty set of items within the sequence. The given sequence has **4 elements** as shown below:

S = <

a	b	(c d)	(e f)

>

The length of the sequence is the number of instances of items. Therefore, the sequence has **length = 6** (a, b, c, d, e, f).

For each instance, we can either include it in the subsequence or not include it in the subsequence. Therefore, we get:

total # of subsequences = 2^6 = 64

However, one subsequence will have all instances removed (i.e. empty subsequence). After removing this case, we get total number of non-empty subsequences as **63.**

(b) We know that two sequences can be joined if dropping the first item in s1 is the same as dropping the last item in s2. Therefore, we will remove the first item (- first) and remove the last item (- last) to compare different sequences:

S	- first	- last
(a c) e	c e	(a c)
	a e	
b (c d)	(c d)	b c
		b d
b c e	c e	b c
a (c d)	(c d)	a c
		a d
(a b) d	b d	(a b)
	a d	
(a b) c	b c	(a b)
	a c	

Name: Garvit Rajkumar Pugalia
UID: 504628127

Now, we can identify sequences s1 and s2 such that s1(- first) = s2(- last). These sequences were identified to create the following table:

s1	s1 (- first)	s2	s2 (- last)	Result
(a b) d	b d	**b (d c)**	b d	**(a b) (d c)**
(b a) d	a d	**a (d c)**	a d	**(b a) (d c)**
(a b) c	b c	**b (c d)**	b c	**(a b) (c d)**
(a b) c	b c	**b c e**	b c	**(a b) c e**
(b a) c	a c	**a (c d)**	a c	**(b a) (c d)**

Therefore, we get two sequences: <**(a b) (c d)**> and <**(a b) c e**>. These values will be pruned if the 3-length subsequences are not frequent. For example, for <(a b) c e>, the subsequence <a b e> doesn't appear in L3 and is not frequent. This sequence can be pruned!

However, all the subsequences of <(a b) (c d)> appear in the list of 3-length subsequences. Therefore, we get the final list of candidate 4-sequences: <**(a b) (c d)>.**