Nguyen Nguyen UID#:004870721

Problem A. *Apriori and Condensed Representations* **(10 pts)**

1. What is Apriori property? (2 pt)

1) Apriori property is a property such that any subset of a frequent itemset must be also frequent. It is also an anti-monotone property. For example, if a transaction containing {A,B,C} is frequent, then its subsets {A}, {B}, {C}, {AB}, {AC}, {BC} are also frequent.

2. Consider the transactional database shown in table 1. Assuming the **minimum support as 2**, find out all the frequent itemsets with their supports from the given database using the Apriori algorithm. (8 pts)

2)

Database D	
Trans. ID	Items
1	c, r, a
2	p, b
3	r, c, b
4	p, c, a
5	a, b, s, c
6	t, a, b, c

Min_Sup = 2

Scan D

1-candidates	
ItemSet	Sup
a	4
b	4
c	5
r	2
p	2
s	1
t	1

Freq 1-itemsets	
ItemSet	Sup
a	4
b	4
c	5
r	2
p	2

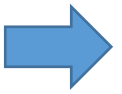
2-candidates
ItemSet
ab
ac
ar
ap
bc
br
bp
cr
cp
rp

Scan D

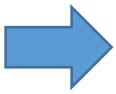
Counting	
ItemSet	Sup
ab	2
ac	4
ar	1
ap	1
bc	3
br	1
bp	1
cr	2
cp	1
rp	0

Freq 2-itemsets	
ItemSet	Sup
ab	2
ac	4
bc	3
cr	2

Scan D | 3-candidates | Freq 3-itemsets



ItemSet
abc



ItemSet	Sup
abc	2

Therefore, all frequent itemsets with their supports include:
{a:4, b:4, c:5, r:2, p:2, ab:2, ac:4, bc:3, cr:2, abc:2}

Problem B. *Dynamic Itemset Counting (DIC)* (10 pts)

1. Apply Dynamic Itemset Counting (DIC) to the above problem. You can assume that the transactions are read one by one (sequentially) from disk, starting from the transaction ID #1. You ***only*** need to show when an itemset becomes frequent (during which scan and after reading which transaction). Compare the number of scans required in DIC to Apriori. (10 pts)

1) Scan 1:
– b, r, c becomes frequent after transaction ID # 3, so start counting support of br,bc,cr 2-itemsets immediately from next transaction ID #4, while continue counting 1-itemset for the rest of the scan 1.
– p, a becomes frequent after transaction ID #4, so start counting support for ap, ab, ac, ar, br, bc, cr 2 –itemsets at transaction ID #5.
– ab, ac becomes frequent after transaction ID #6, so start counting support for abc, while counting the rest of 2-item support.
– bc becomes frequent after transaction ID #6.
Scan 2:
– after transaction ID #3, cr become frequent but no way to become 3-itemset.
– after transaction ID #6, abc will become frequent thus ending the scan.

Overall, DIC has reduced one database scan. Meanwhile, Apriori algorithm requires 3 total database scans for each k-itemset, where k = 3.

Problem C. *FP-Tree* (20 pts)

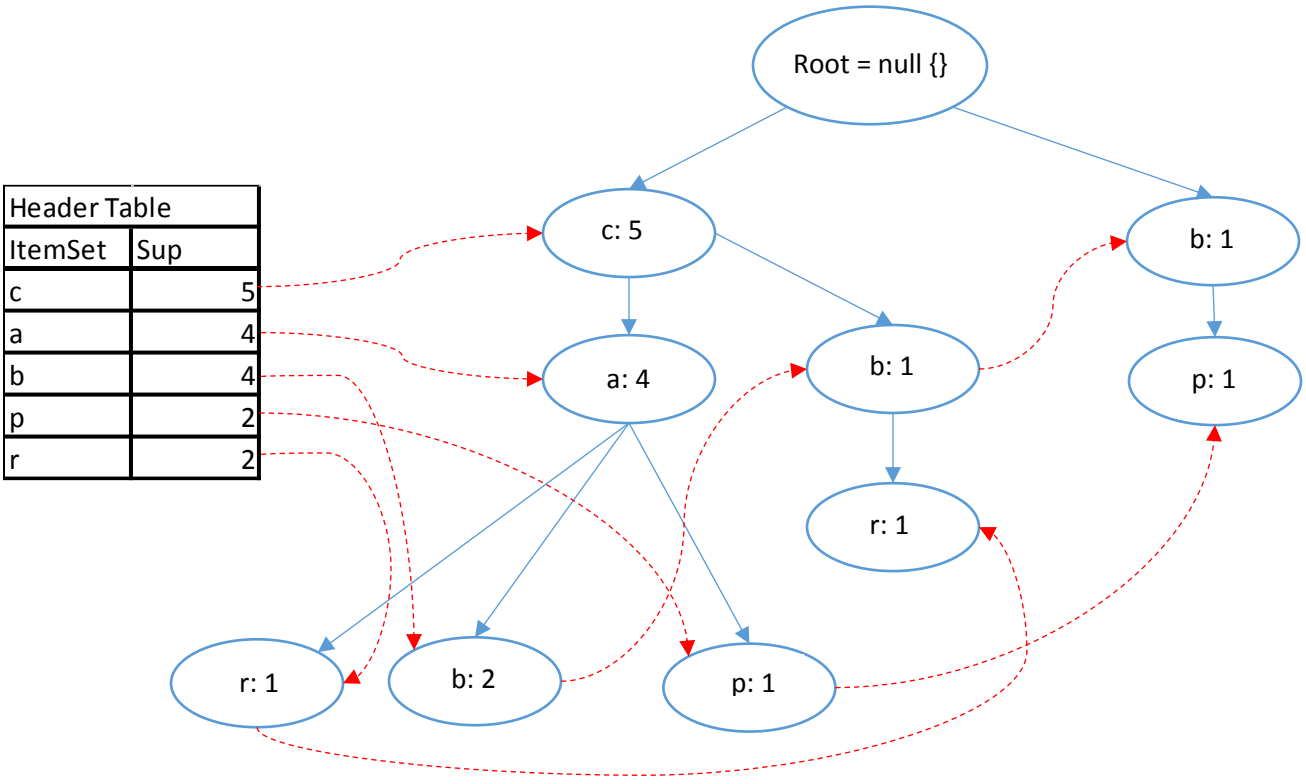
1. Using the same transactional database and minimum support constraint as problem A, build the corresponding FP-tree step by step. While ordering the frequent items, use alphabetical order to break ties between the items having same support. (12 pts)

2. Use this FP-tree to find *all* the frequent itemsets that contain 'b' with their supports. You must show *all* the projected databases and conditional FP-trees generated step by step for this computation. (8 pts)

1) - First scan of the database is the same as Apriori which to derive the set of 1-itemsets and their support counts. This is for the Header table which is sorted in descending support count.

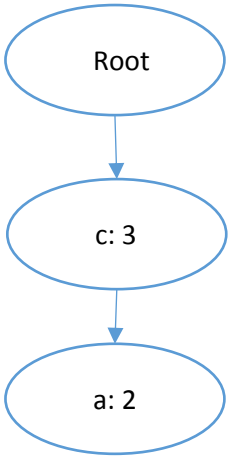
Database D		
Trans. ID	Items	freq items ordered
1	c, r, a	c, a, r
2	p, b	b, p
3	r, c, b	c, b, r
4	p, c, a	c, a, p
5	a, b, s, c	c, a, b
6	t, a, b, c	c, a, b

- Second scan is to construct the FP-tree.



Header Table	
ItemSet	Sup
c	5
a	4
b	4
p	2
r	2

2) First frequent itemset **b:4**
Condition pattern base: {ca: 2, c:1}. Thus, we can build a b-projected FP-Tree (as seen below).
The local frequent items include: c:3, a:2 because they meet min_sup = 2



However, it is a single branch, therefore we do not need to recursively mine again. We can find all the combinations of b-projected FP tree. The sets are {c,a,ac}
In conclusion, we find that the FP tree generates the set of patterns **{bc, ba, bac}** which contains 'b' in their support.

Problem D. *Sequential Pattern Mining* (10 pts)

1. Compute the support of these three sequences for Table 2: (4 pts)
   (a) <ac>
   (b) <(a,c)>
2. Assume that the minimum support is 2, use PrefixScan to find out all the frequent sequential patterns. (6 pts)

1) a) **<ac>** means that they 'a' and 'c' do not have to be in the same component of a sequence, as long 'b' is after 'a', then it will count as a support. Therefore, the number of support for **<ac>: 3**.
b) **<(a,c)>** means that 'a' and 'c' must be in the same component (hence the parenthesis). Therefore, the number of support for **<(a,c)>: 1**

2) assuming min_sup = 2

We find the frequent patterns for the prefix as seen below:

Prefix	Sequential Patterns
<a>	<a><aa><ac><ad><(ad)><(ad)c><adc>
<b>	does not meet support
<c>	<c>
<d>	<d>, <da>, <dc>
<e>	does not meet support
<f>	does not meet support

Below is the step by step instruction of how to obtain the sequential patterns.

1st Scan:	<a>:3, <c>:5, <d>:4 (count from original)	
Prefix	Create Projected(suffix) databases from original	Support
<a>	<(_, d)(c,d)>	3
	<dca>	
	<(_,b,d)(a,c)(c,e)>	
<a> scan:	<a>:2, <c>:3, <d>:3 (scan from projected databse of <a>)	
<aa>	Support met, create projected database of <aa>	2
	<(_c)(c,e)>	
<aa> scan:	<c>:1, <e>:1 (scan from projected database of <aa>	
	Nothing met min_sup, go to next	
<(a,a)>	<>	0
<ac>	Support met, create projected database of <ac>	3
	<d>	
	<a>	
	<(c,e)>	
<ac> scan:	<d>:1, <a>:1, <c>:1, <e>: 1 (scan from projected database of <ac>	
	Nothing met min_sup, go to next	
<(a,c)>	<>	0
<(ad)>		2
<ad>	Support met, create projected database of <ad>	3
	<(cd)>	
	<ca>	
	<(a,c)(c,e)>	
<ad> scan:	<a>: 1, <b>:1, <c>:3, <d>:3 (scan from projected database of <ad>)	
<adc>	<>	1
<(ad)c>	Support met, create projected database of <(ad)c>	2
	<d>	
	<(c,e)>	

<(ad)c>		
scan:	<d>:1, <c>:1, <e>:1 (scan from projected database <(ad)c>	
	Nothing met min_sup, go to next	
<adc>		2
<a(dc)>	<>	0
<(adc)>	<>	0
<add>	<>	0
<(ad)d>	Nothing met min_sup, go to next	1
<a(dd)>	<>	0
<(add)>	<>	0

1st Scan:	<a>:3, <c>:5, <d>:4	
Prefix	Create Projected(suffix) databases from original	Support
<c>	<(_,d)>	5
	<a>	
	<(c,e)>	
<c> scan:	<d>:1, <a>:1, <c>1, <e>:1 (scan from projected databse of <c>)	
	Nothing met min_sup, go to next	

1st Scan:	<a>:3, <c>:5, <d>:4	
Prefix	Create Projected(suffix) databases from original	Support
<d>	<(c,d)>	4
	<ca>	
	<(a,c)(c,e)>	
	<c>	
<d> scan:	<a>:2, <c>:4, <d>:1	
<da>	Support met, create projected database of <da>	2
	<_c><(c,e)>	
<da> scan:	<c>:1, <e>:1	
	Nothing met min_sup, go to next	
<(d,a)>	<>	0
<dc>	Support met, create projected database of <dc>	3
	<d>	
	<a>	
	<(c,e)>	
<dc> scan:	<d>: 1, <a>: 1, <c>:1, <e>:1	
	Nothing met min_sup, go to next	
<(d,c)>	<>	0
<dd>	Nothing met min_sup, go to next	1
<(d,d)>	Nothing met min_sup, go to next	0