TDT4300 — Assignment 2

ASSOCIATION ANALYSIS

By Thomas Bjerke and Trym Grande

1 - Apriori Algorithm

a)

Market basket transaction

TID	**Items**
T1	H, B, K
T2	H, B
T3	H, C, I
T4	C, I
T5	I, K
T6	H, C, I, U

↓

Items (1-itemsets)

Minimum support = 2

Item	**Count**
H	4
B	2
K	2
C	3
I	4
U	1

Eliminate U since count(U) < minsup

↓

Pairs (2-itemsets)

Itemset	**Count**
{H, B}	2
{H, K}	1
{H, C}	2
{H, I}	2
{B, K}	1

{K, I}	1
{C, I}	3

Eliminate {H, K}, {B, K}, and {K, I} since count < minsup

↓

Triplets (3-itemsets)

Itemset	**Count**
{H, B, K}	1
{H, C, I}	2

b)

Frequent itemset: {H, C, I}

Candidate rules:

HC → I HI → C CI → H

H → CI C → HI I → HC

Calculating confidence for each rule:

$c(HC \rightarrow I) = \sigma(\{H, C, I\}) / \sigma(\{H, C\}) = 2/2 = 1$

$c(HI \rightarrow C) = \sigma(\{H, C, I\}) / \sigma(\{H, I\}) = 2/2 = 1$

$c(CI \rightarrow H) = \sigma(\{H, C, I\}) / \sigma(\{C, I\}) = 2/3 = 0.67$

$c(H \rightarrow CI) = \sigma(\{H, C, I\}) / \sigma(\{H\}) = 2/4 = 0.5$

$c(C \rightarrow HI) = \sigma(\{H, C, I\}) / \sigma(\{C\}) = 2/3 = 0.67$

$c(I \rightarrow HC) = \sigma(\{H, C, I\}) / \sigma(\{I\}) = 2/4 = 0.5$

All high confidence rules:

HC → I

HI → C

CI → H

C → HI

2 - FP-Growth Algorithm

Transactions in a transaction database:

TID	Items
T1	b, e, g

T2	b, d, i
T3	b, d, e, f
T4	a, d, e
T5	d, e
T6	b, d, j
T7	b, c, d, e, f
T8	b, d, e, f
T9	b, e, h

Frequencies (support):

Item	**Frequency**
a	1
b	7
c	1
d	5
e	6

f	3
g	1
h	1

Frequency of each item.

Frequent pattern set:

L = {b:7,e:6,d:5,f:3}

Includes the item and the frequency for each item within the minimum support count, meaning a frequency of at least 3 in this case.

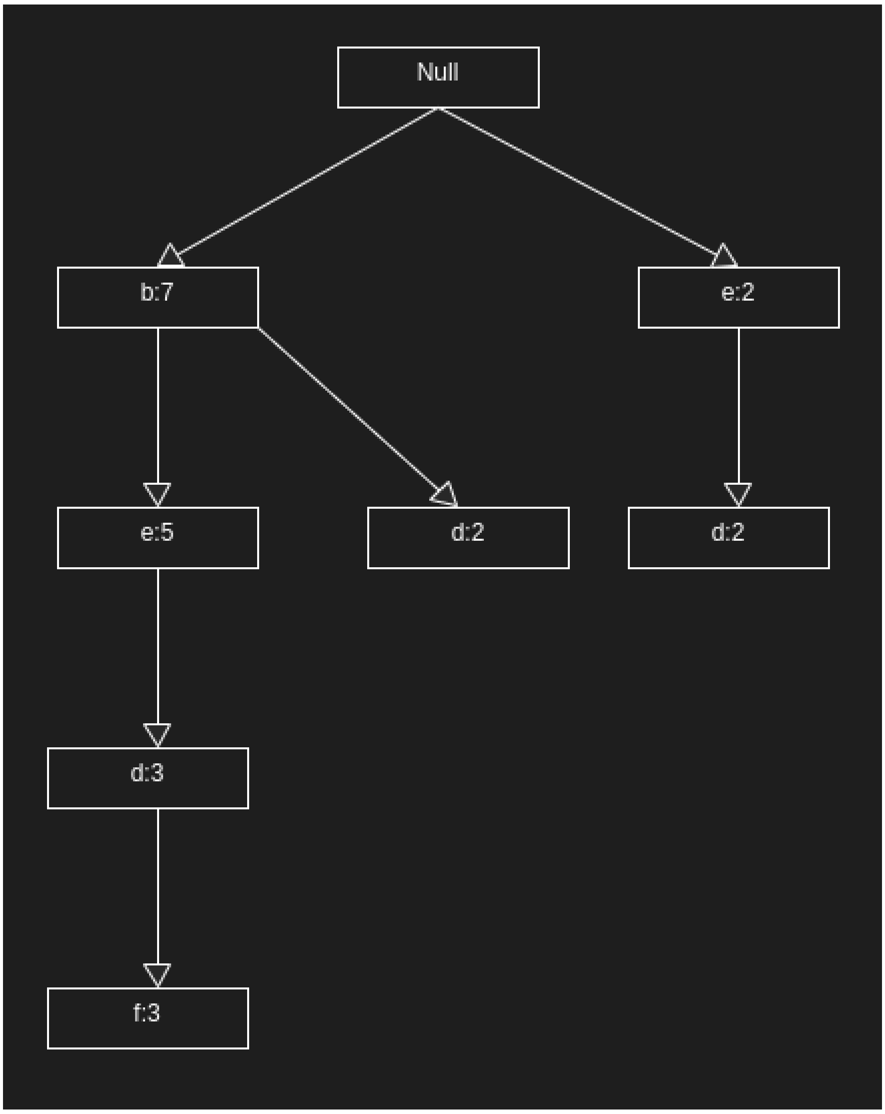
Ordered item set:

TID	**Items**	**Ordered item set**
T1	b, e, g	b, e
T2	b, d, i	b, d
T3	b, d, e, f	b, e, d, f
T4	a, d, e	e, d
T5	d, e	e, d
T6	b, d, j	b, d
T7	b, c, d, e, f	b, e, d, f

T8	b, d, e, f	b, e, d, f
T9	b, e, h	b, e

Items ordered descending where only items that exist in L are included.

Trie:



Each ordered item set is added to the trie by increasing the count of the item node if it has been walked before, and creating a new branch where the path does not exist by initializing an item node with count = 1.

Conditional pattern base:

Item	Conditional pattern base
b	
e	{b:5}, {e:2}
d	{b,e:3}, {b:2}, {e:2}
f	{b,e,d:3}

Each path to each item along with the frequency of the item by following the given path. Starting and ending nodes are not included in the pattern base.

Conditional frequent pattern tree:

Item	Conditional pattern base	Conditional frequent pattern tree
b		
e	{b:5}, {e:2}	
d	{b,e:3}, {b:2}, {e:2}	
f	{b,e,d:3}	{b,e,d:3}

Describes the pattern that is common to all conditional pattern bases, along with the total frequency count.

Frequent pattern rules:

Item	Frequent pattern generated

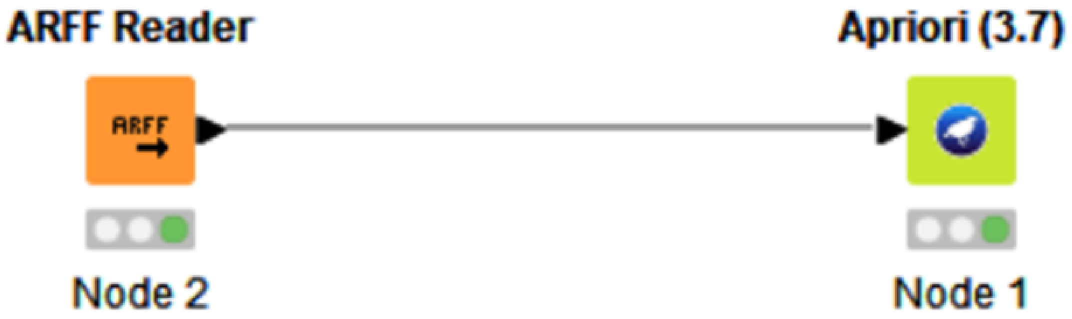
b	
e	
d	
f	<b,f:3>, <e,f:3>, <d,f:3>

These are generated by pairing the conditional frequent pattern tree with each item along with the corresponding frequency count.

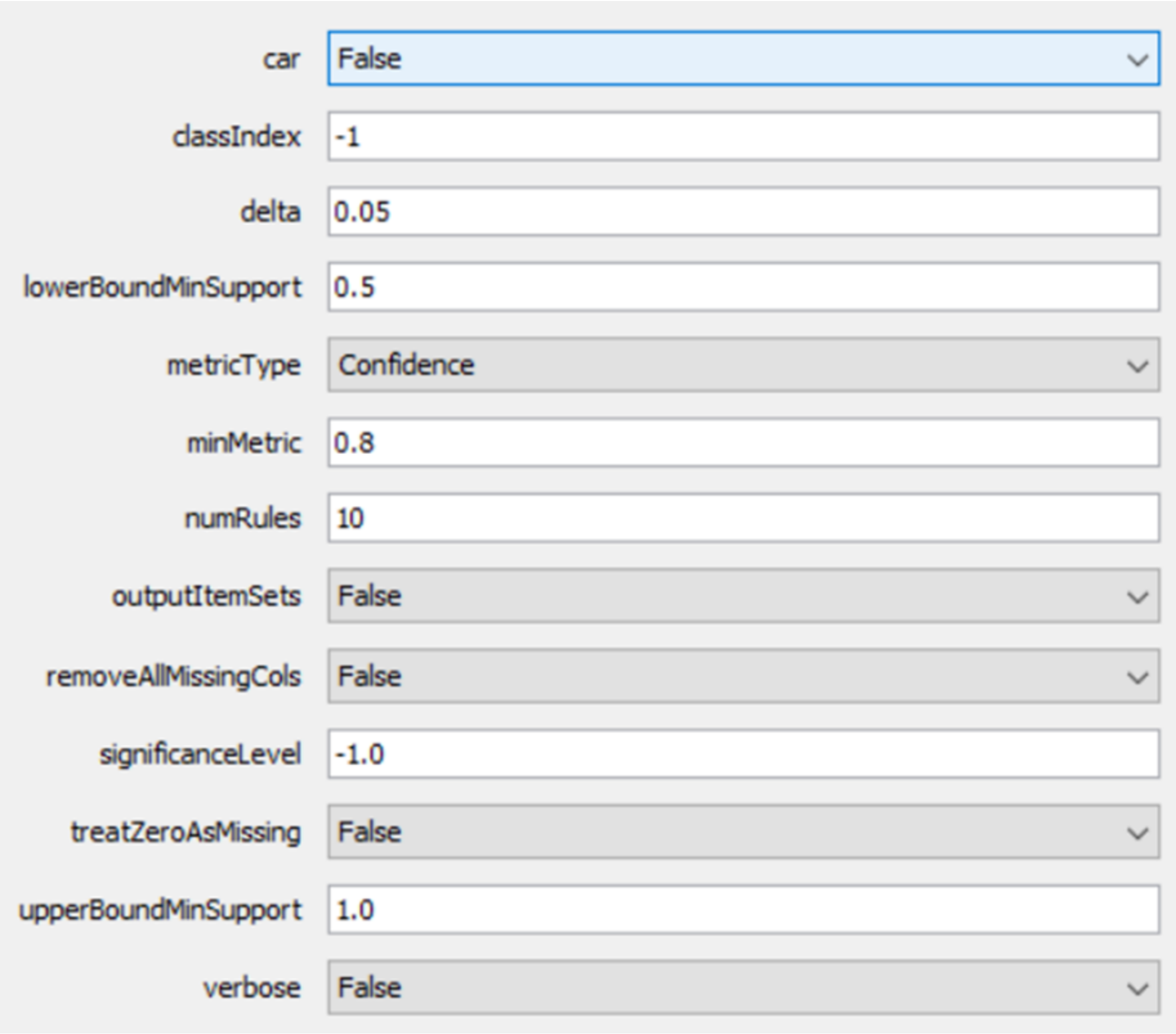
3 – KNIME

Apriori:

Workflow:

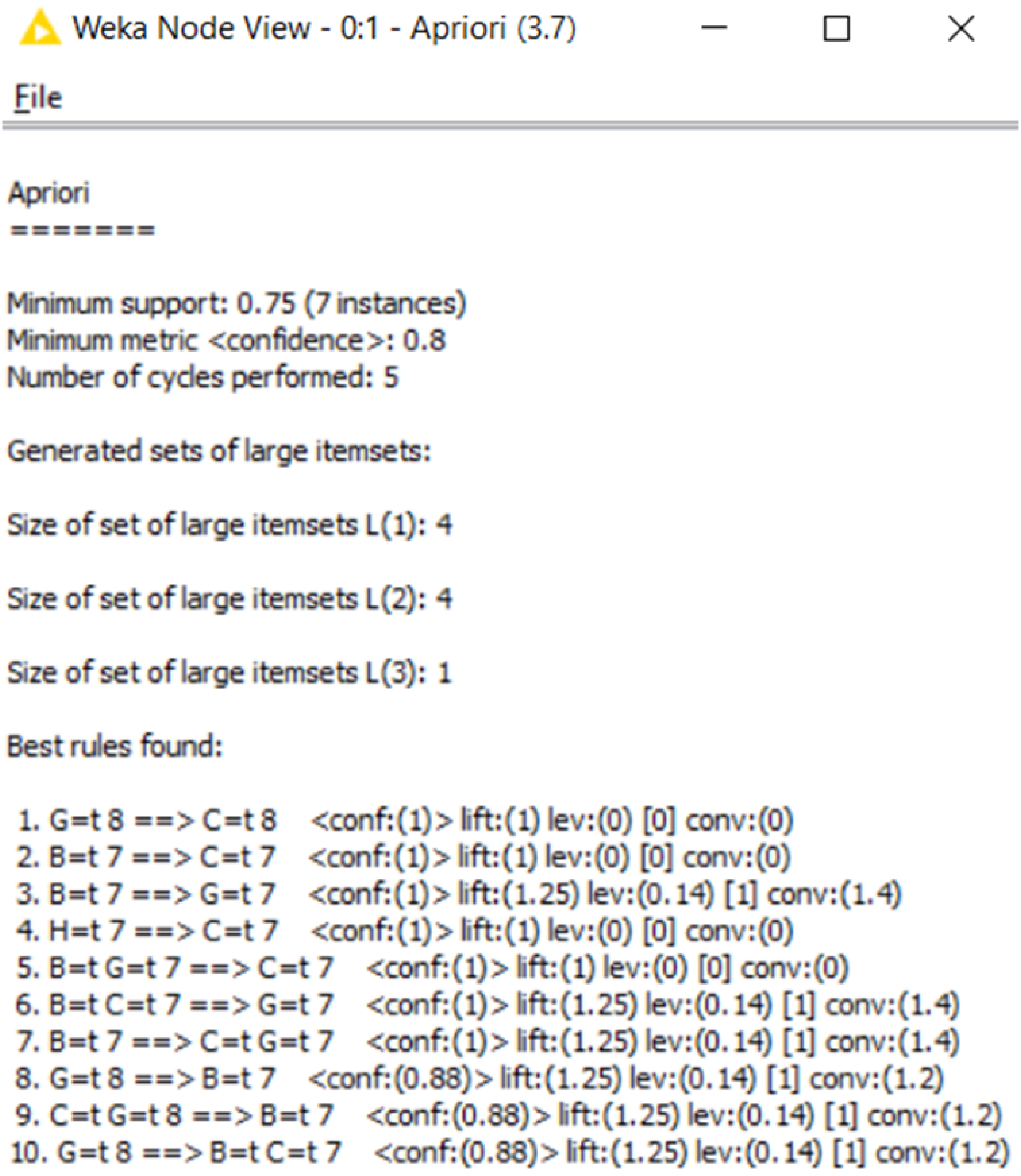


Apriori node configuration (as instructed by instructor on Piazza):



car	False
classIndex	-1
delta	0.05
lowerBoundMinSupport	0.5
metricType	Confidence
minMetric	0.8
numRules	10
outputItemSets	False
removeAllMissingCols	False
significanceLevel	-1.0
treatZeroAsMissing	False
upperBoundMinSupport	1.0
verbose	False

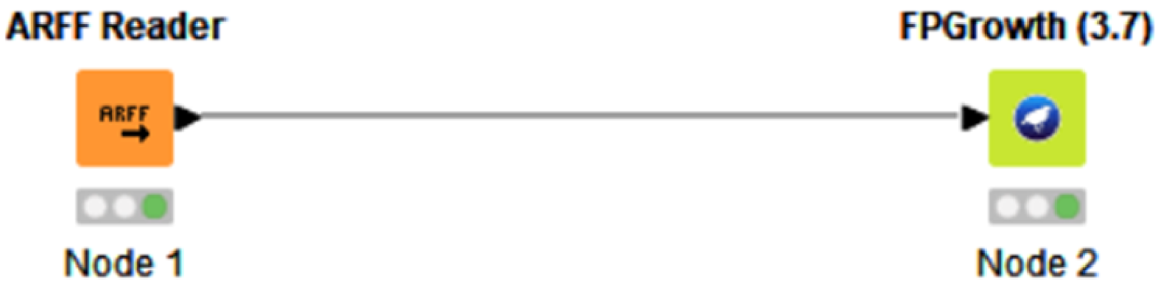
Output:



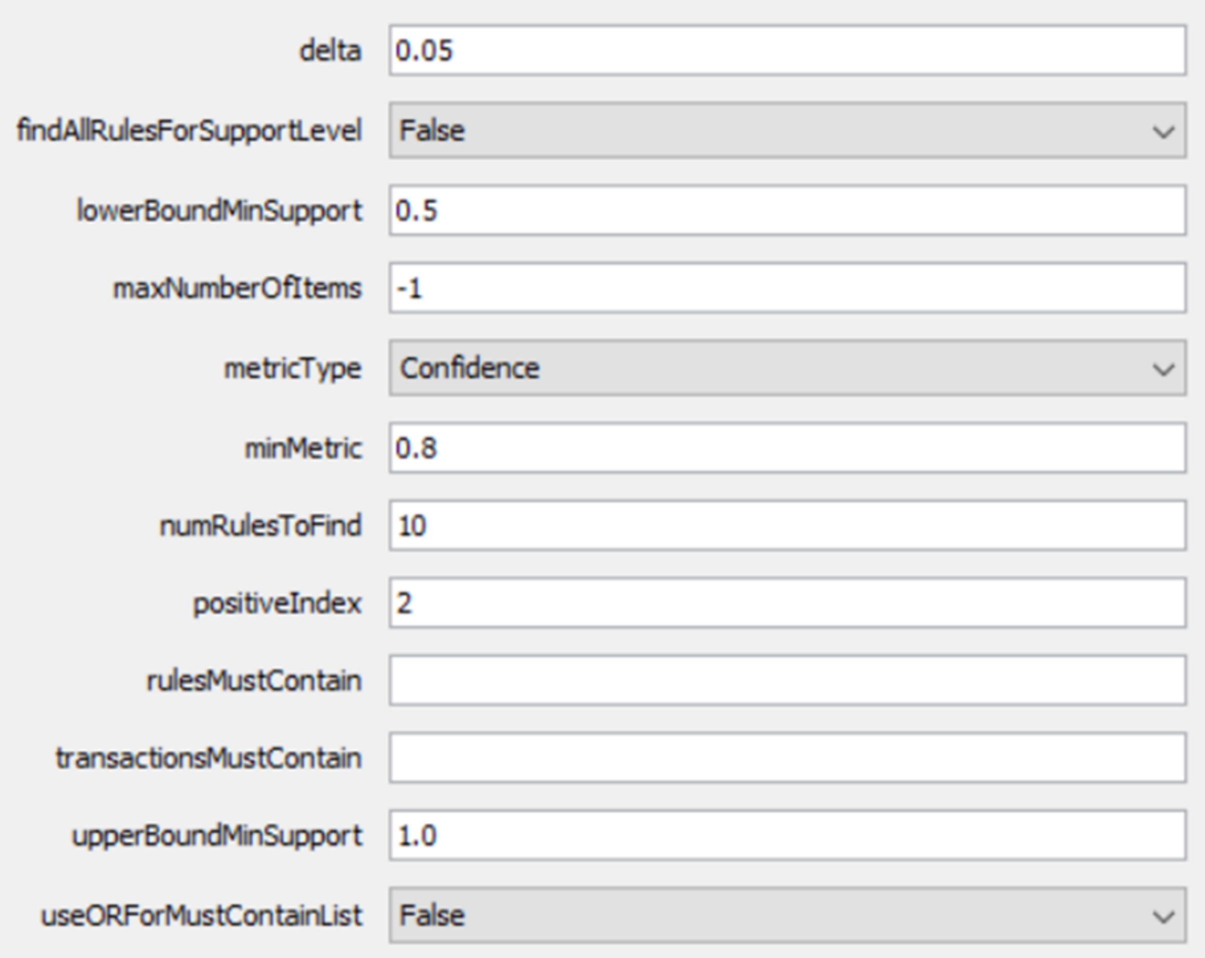
```
Apriori
=======

Minimum support: 0.75 (7 instances)
Minimum metric <confidence>: 0.8
Number of cycles performed: 5

Generated sets of large itemsets:

Size of set of large itemsets L(1): 4

Size of set of large itemsets L(2): 4

Size of set of large itemsets L(3): 1

Best rules found:

 1. G=t 8 ==> C=t 8    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 2. B=t 7 ==> C=t 7    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 3. B=t 7 ==> G=t 7    <conf:(1)> lift:(1.25) lev:(0.14) [1] conv:(1.4)
 4. H=t 7 ==> C=t 7    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 5. B=t G=t 7 ==> C=t 7    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 6. B=t C=t 7 ==> G=t 7    <conf:(1)> lift:(1.25) lev:(0.14) [1] conv:(1.4)
 7. B=t 7 ==> C=t G=t 7    <conf:(1)> lift:(1.25) lev:(0.14) [1] conv:(1.4)
 8. G=t 8 ==> B=t 7    <conf:(0.88)> lift:(1.25) lev:(0.14) [1] conv:(1.2)
 9. C=t G=t 8 ==> B=t 7    <conf:(0.88)> lift:(1.25) lev:(0.14) [1] conv:(1.2)
10. G=t 8 ==> B=t C=t 7    <conf:(0.88)> lift:(1.25) lev:(0.14) [1] conv:(1.2)
```

FP-Growth:

Workflow:

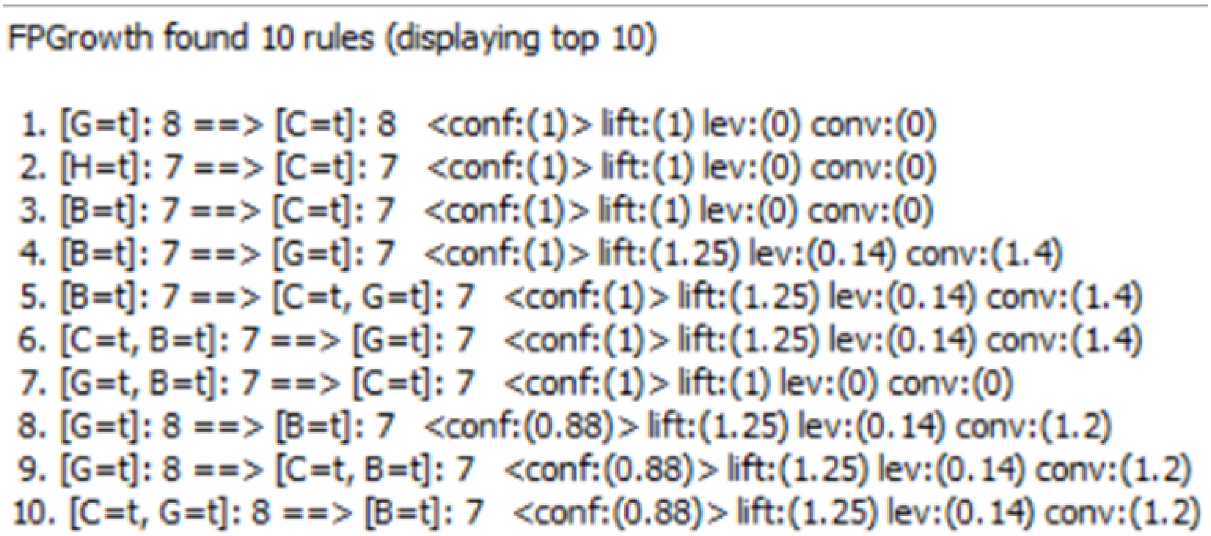


FP-Growth node configuration:



```
delta                          0.05
findAllRulesForSupportLevel    False
lowerBoundMinSupport           0.5
maxNumberOfItems               -1
metricType                     Confidence
minMetric                      0.8
numRulesToFind                 10
positiveIndex                  2
rulesMustContain
transactionsMustContain
upperBoundMinSupport           1.0
useORForMustContainList        False
```

Output:



```
FPGrowth found 10 rules (displaying top 10)

 1. [G=t]: 8 ==> [C=t]: 8   <conf:(1)> lift:(1) lev:(0) conv:(0)
 2. [H=t]: 7 ==> [C=t]: 7   <conf:(1)> lift:(1) lev:(0) conv:(0)
 3. [B=t]: 7 ==> [C=t]: 7   <conf:(1)> lift:(1) lev:(0) conv:(0)
 4. [B=t]: 7 ==> [G=t]: 7   <conf:(1)> lift:(1.25) lev:(0.14) conv:(1.4)
 5. [B=t]: 7 ==> [C=t, G=t]: 7   <conf:(1)> lift:(1.25) lev:(0.14) conv:(1.4)
 6. [C=t, B=t]: 7 ==> [G=t]: 7   <conf:(1)> lift:(1.25) lev:(0.14) conv:(1.4)
 7. [G=t, B=t]: 7 ==> [C=t]: 7   <conf:(1)> lift:(1) lev:(0) conv:(0)
 8. [G=t]: 8 ==> [B=t]: 7   <conf:(0.88)> lift:(1.25) lev:(0.14) conv:(1.2)
 9. [G=t]: 8 ==> [C=t, B=t]: 7   <conf:(0.88)> lift:(1.25) lev:(0.14) conv:(1.2)
10. [C=t, G=t]: 8 ==> [B=t]: 7   <conf:(0.88)> lift:(1.25) lev:(0.14) conv:(1.2)
```

4 - Compact Representation of Frequent Itemsets

Closed Frequent Itemsets	Support Count
{b}	10
{d}	13
{a, d}	11
{b, d}	7
{b, e}	8
{d, e}	6
{a, b, e}	7
{a, c, d}	6
{b, d, e}	4
{a, c, d, e}	5

The closed frequent itemsets are candidates for all frequent itemsets.

Want to generate all frequent itemsets including the support counts.

k_max = 4 // the maximum size of the closed frequent itemsets ({a, c, d, e})

find all frequent itemsets of size k_max (4) down to 1 by iterating k:

k=4:

{a,c,d,e} = 5 // already know this

// support for the rest will be {a,b,c,d,e}.support (since there is only one superset) = {a,c,d,e}.support + {b}.support = 5 + 10 = 15

{b,a,c,d} = 15

{b,c,d,e} = 15

{b,a,d,e} = 15

{b,a,c,e} = 15

k=3:

{a,d,e} = 4 // already know this

{a,b,e} = 7 // already know this

{a,c,d} = 6 // already know this

{a,b,c} = max({b,a,c,d},{b,a,c,e}) = max(15,15) = 15

{a,b,d} = max({b,a,c,d},{b,a,d,e}) = max(15,15) = 15

{a,c,e} = max({a,c,d,e},{b,a,c,e}) = max(5,15) = 15

{b,c,d} = max({b,a,c,d},{b,c,d,e}) = max(15,15) = 15

{b,c,e} = max({b,c,d,e}, {b,a,c,e}) = max(15,15) = 15

{b,d,e} = max({b,c,d,e},{b,a,d,e}) = max(15,15) = 15

{c,d,e} = max({a,c,d,e},{b,c,d,e}) = max(5,15) = 15

k=2:

{a,d} = 11 // already know this

{b,d} = 7 // already know this

{b,e} = 8 // already know this

{a,b} = max({a,b,e},{a,b,c},{a,b,d}) = max(7,15,15) = 15

{a,c} = max({a,c,d},{a,b,c},{a,c,e}) = max(15,15,15) = 15

{a,e} = max({a,d,e},{a,b,e},{a,c,e}) = max(4,7,15) = 15 {b,c} = max({a,b,c},{b,c,d},{b,c,e}) = max(15,15,15) = 15

{c,d} = max({a,c,d},{b,c,d},{c,d,e}) = max(6,15,15) = 15

{c,e} = max({a,c,e},{b,c,e},{c,d,e}) = max(15,15,15) = 15

{d,e} = max({a,d,e},{b,d,e},{c,d,e}) = max(15,15,15) = 15

k=1:

{b} = 10 // already know this

{d} = 13 // already know this

{a} = max({a,d},{a,b},{a,c}{a,e}) = max(11,15,15,15) = 15

{c} = max({a,c},{b,c},{c,d},{c,e}) = max(15,15,15,15) = 15

{e} = max({b,e},{a,e},{c,e},{d,e}) = max({b,e},{a,e},{c,e},{d,e}) = max(8,15,15,15) = 15