TDT4300 — ASSIGNMENT 2

ASSOCIATION ANALYSIS

GROUP 98

1 APRIORI ALGORITHM

a)

Set	Support	State
{B}	2	Selected
{C}	3	Selected
{H}	4	Selected
{I}	4	Selected
{K}	2	Selected
{U}	1	Rejected

Set	Support	State
{B,C}	0	Rejected
{B,H}	2	Selected
{B,I}	0	Rejected
{B,K}	1	Rejected
{C,H}	2	Selected
{C,I}	3	Selected
{C,K}	0	Rejected
{H,I}	2	Selected
{H,K}	1	Rejected
{I,K}	1	rejected

Set	Support	state
{B,H,C}	0	Rejected
{B,H,I}	0	Rejected
{C,H,I}	2	Selected
{C,B,C}	0	Rejected

b) {C,H} ➔ {I} $\frac{support(C,I,H)}{support(C,H)} = \frac{2}{2} = 1 \ > 60\%$

{I,H} ➔ {C} $\frac{support(C,I,H)}{support(I,H)} = \frac{2}{2} = 1 \ > 60\%$

{I,C} ➔ {H} $\frac{support(C,I,H)}{support(I,C)} = \frac{2}{3} = 0.66 > 60\%$

{I} ➔ {C,H} $\frac{support(C,I,H)}{support(I)} = \frac{2}{4} = 0.5$ < 60% ➔ Rejected

{C} ➔ {I, H} $\frac{support(C,I,H)}{support(C)} = \frac{2}{3} = 0.66 >$ 60%

{H} ➔ {C, I} $\frac{support(C,I,H)}{support(H)} = \frac{2}{4} = 0.5$ < 60% ➔ Rejected

2 FP-Growth Algorithm

Step 1 :

We sort the items by their support:

Item	Support
b	7
d	7
e	7
f	3
a	1
c	1
g	1
u	1
i	1
j	1

Step 2:

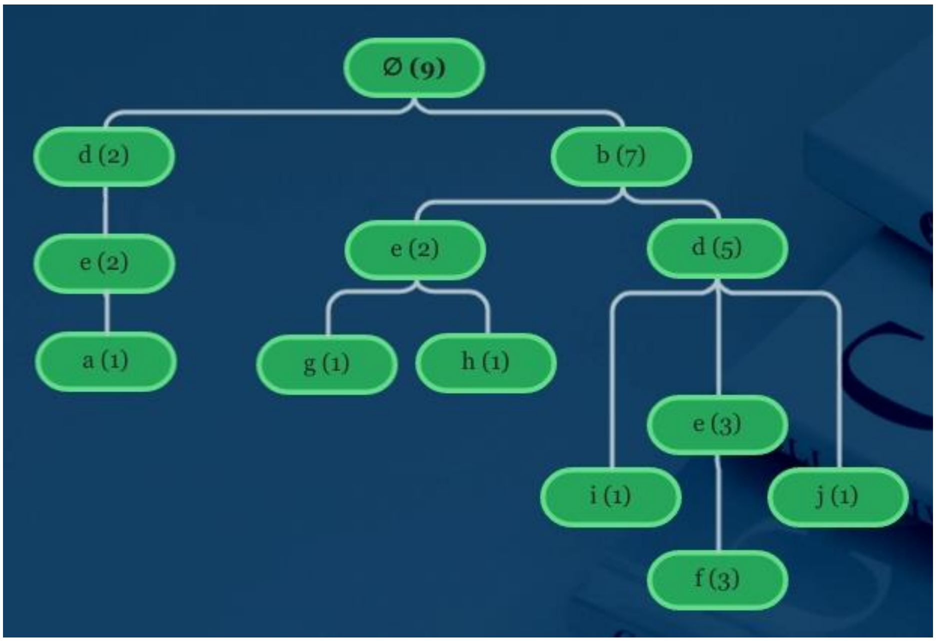
We then sort the itemsets based on step 1 :

ID	Transaction
T1	b e g
T2	b d i
T3	b d e f
T4	d e a
T5	d e
T6	b d j
T7	b d e f c
T8	b d e f
T9	b e h

NTNU

Step 3:

We design the FP-Tree



We notice that some items have the number of support less than the minimum which is 2. We will re-design the previous tree and remove the nodes with less than the minimum support. We therefore get the following graph tree:

Step 4:

Project in order : f(3)e(7)d(7)b(7)

- **Projected FP-Tree for f:**

There is a path to f which is "b d e f". The conditional sub-database is:

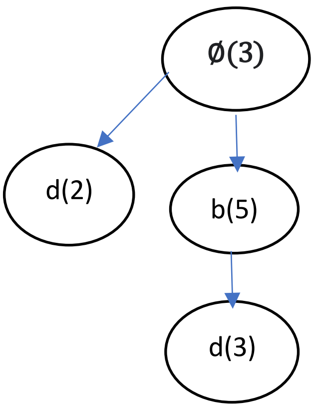
Path	Count
b d e f	3

The projected tree is Rf: Ø(3) ➔ b(3) ➔ d(3) ➔ e(3)

- **Projected FP-Tree for e:**

Path	Count
d e	2
b e	2
b d e	3

The projected tree is



Since we don't have a path and the exercise's instructions ask for the recursive method, we choose to take the next item with the lowest support count in the sub-database for e which is **d(5).**

- **Projected FP-Tree for e d:**

Path	Count
d	2
b d	3

The FP-Tree for Red: Ø(5) ➔b(3)

- **Projected FP-Tree for e b:**

Path	Count
Ø	5

The FP-Tree for Reb: Ø(5)

- **Projected FP-Tree for d:**

Path	Count
b d	5
d	2

The FP-Tree for Rd: Ø(7) ➔b(5)

- **Projected FP-Tree for b:**

Path	Count
b	7

The FP-Tree for Rb: Ø(7)

Step 5:

Frequent items for f : Because we have a path, we will immediately stop the recursion. We then mine the itemsets with prefix P=f: **{fe(3), fb(3), fd(3), fbd(3), fbe(3), fde(3), fbde(3)}**

Frequent items for e : Since we don't have a path for Re, we will recursively project in order to get Red where d is the next item with the lowest support count. Red is a path and we mine the frequent itemsets with prefix P=ed: **{ed(5), ebd(3)}**

The next recursion will be Reb where b is the following, and last, item with the lowest support count. Reb is a path and we mine the frequent itemsets prefix P=eb: **{eb(5)}**

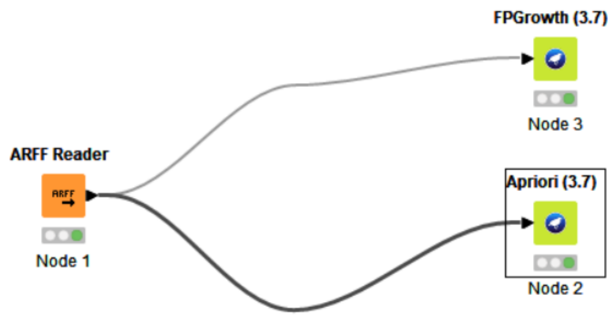
Frequent items for d: We have a path so we immediately stop the recursion. We mine the itemsets with prefix P=d: **{db(5)}**

Frequent items for b: The projected FP-Tree for b was an empty set so the itemset is **Ø.**

Item	Conditional pattern base	Conditional FP-Tree	Frequent itemsets
f	{b,d,e,f: 3}	<b:3, d:3, e:3>	{f, e:3}, i{f, d:3} {f,b:3}, {f,b,d,e:3}, {f,b, d:3}, {f,b,e:3}, {f,d,e:3}
e	{b,e:2}, {b,d,e:3}, {d,e:2}	<b:5, d:3>	
ed	{d:2}, {b,d:3}	<d:3>	{e,d: 5}, {e, d,b: 3}
eb	{b:5}	<Ø:5>	{e,b: 5}
d	{d:2}, {b,d:5}	<b:5>	{d,b: 5}
b	{b:7}	<Ø:7>	{b:7}

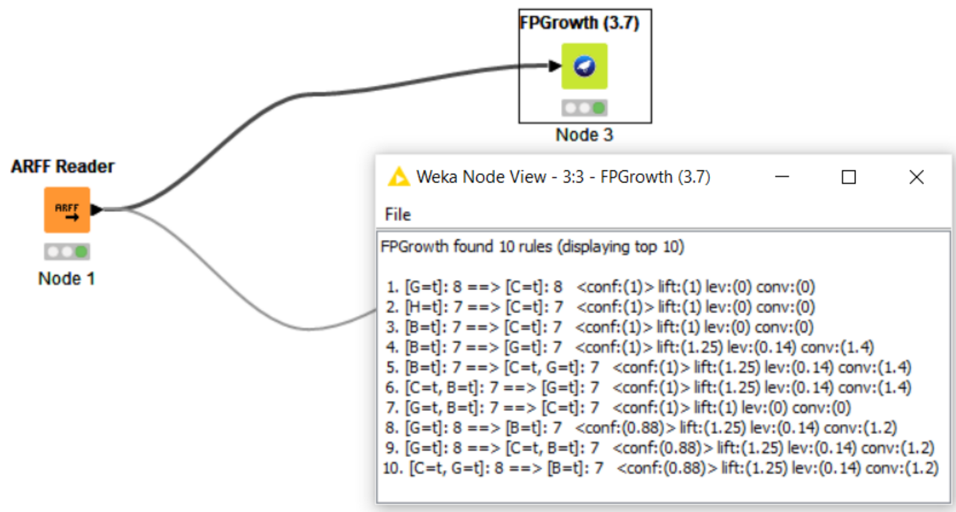
3 KNIME

The workflow:

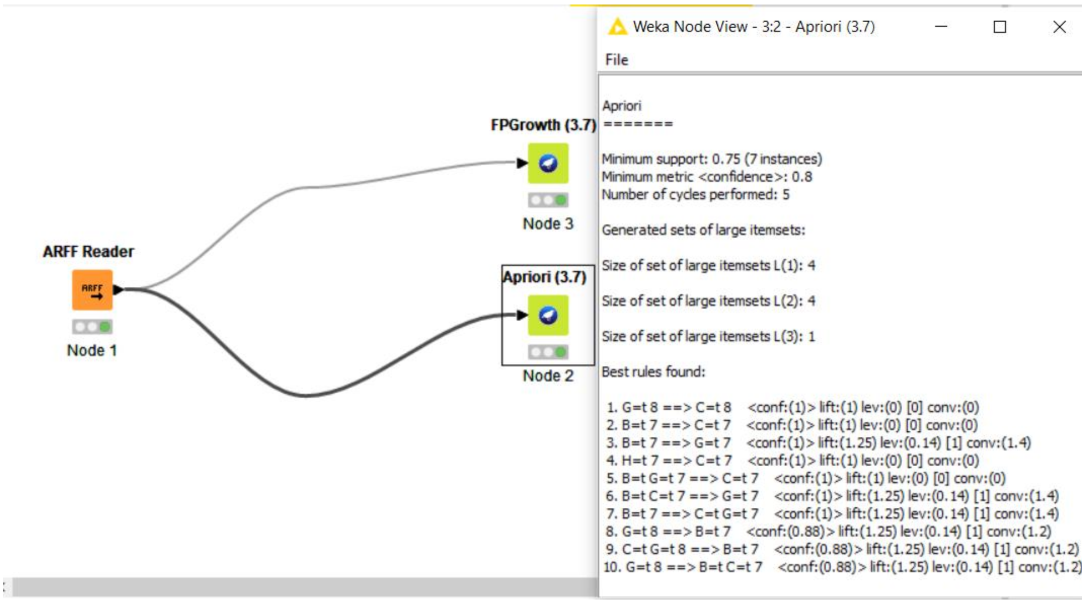


The results:

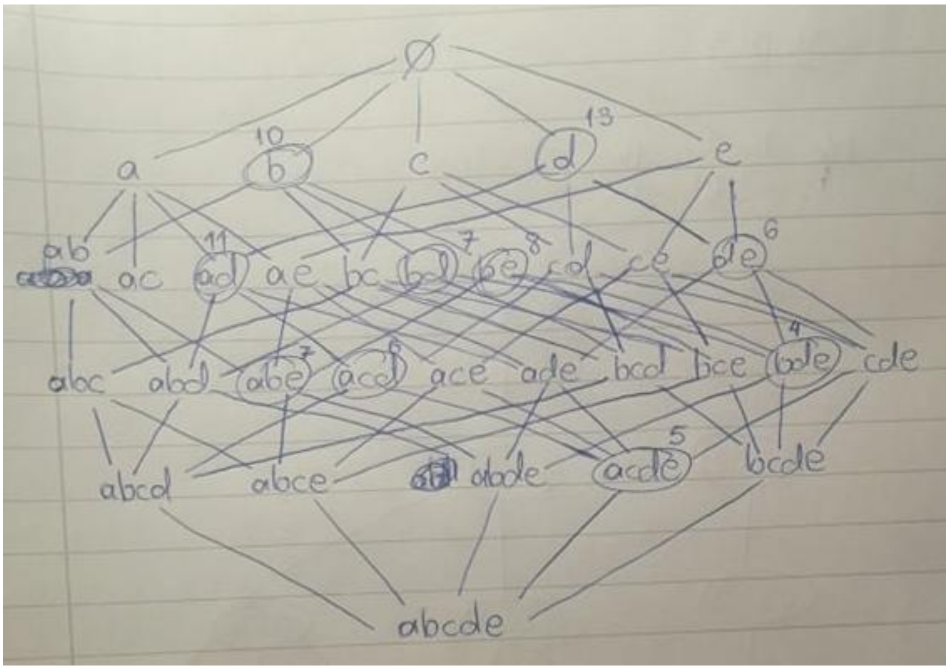
- FP-Growth:



- Apriori:



4 Compact Representation of Frequent Itemsets



We will use a recursive algorithm that will calculate the frequent itemsets bottom-up. The algorithms is based on two rules using the closed frequent itemsets given.

- **1st rule:**
  A superset of closed frequent set can't have the same support count.

- **2nd rule:**
  If a set is not closed, it has the same support count of one of its immediate supersets and specifically, the support of the superset that has the biggest support of all.

We start with k=4 since the maximum closed frequent itemset has length=4.

- **For k=4:** Fk={{a,c,d,e}}

- **For k=3:** Fk={{a,b,e}, {a,c,d}, {a,c,e}, {a,d,e}, {b,d,e}, {c,d,e}}
  except {a,b,e}, {a,c,d}, {b,d,e} for whom we already know the support count, the rest have support of {a,b,c,d,e} which is their immediate superset.

- **For k=2:** Fk={ {a,b}, {a,c}, {a,d}, {a,e}, {b,d}, {b,e}, {c,d}, {c,e}, {d,e} }
  The support of the frequent itemsets that are not closed is generated similarly.

- **For k=1:** Fk={ {a}, {b}, {c}, {d}, {e} }

- **For k=0:** The algorithm is over.

Frequent itemsets	Support Count
{ {a}, {b}, {c}, {d}, {e} }	5
{a,b,e}	7
{a,c,d}	6
{a,c,e}	5
{a,d,e}	5
{b,d,e}	4
{c,d,e}	5
{a,b}	7

{a,c}	6
{a,d}	11
{a,e}	7
{b,d}	7
{b,e}	8
{c,d}	6
{c,e}	5
{d,e}	6
{a}	11
{b}	10
{c}	6
{d}	8
{e}	13