Data Warehousing and Data Mining
Assignment 2
ASSOCIATION ANALYSIS

Alicja Jonczyk, Kacper Multan

1. Apriori Algorithm

a) Show thoroughly the steps on how the frequent itemsets are generated.

We started with counting the number of appearances for each of the 1-item itemsets: A, B, C, D, I, U. Then we continued with calculating the support using the formula:

$$s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$$

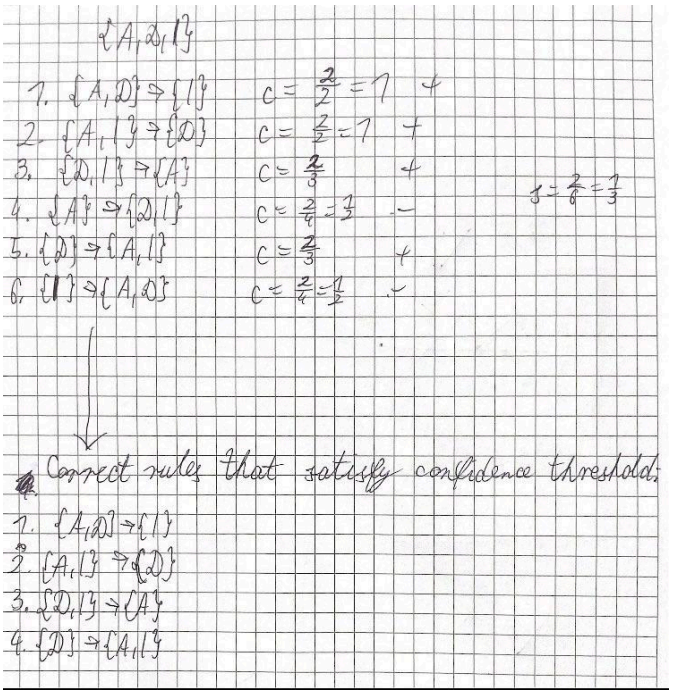
With red we have marked all those itemsets which support was below the minimum threshold. Then, using the itemsets that have passed (marked in green) we generated the 2-item itemsets. We repeated counting and calculating the support. Again, we pruned the not frequent itemsets and repeated the steps until we reached the state where all of the supersets where below the threshold.

Item	Count	Support
A	4	0,666667
B	2	0,333333
C	2	0,333333
D	3	0,5
I	4	0,666667
U	1	0,166667

Item	Count	Support
AB	2	0,333333
AC	1	0,166667
AD	2	0,333333
AI	2	0,333333
BC	1	0,166667
BD	0	0
BI	0	0
CD	0	0
CI	1	0,166667
DI	3	0,5

Item	Count	Support
ABD	0	0
ABI	0	0
ADI	2	0,333333

b) Find all association rules based on this set, given confidence threshold c = 60%.



2. ECLAT Algorithm

2.1

Implement one of three variations of the frequent itemset mining algorithms: Apriori algorithm, FP-growth, and Eclat. Which variation will you implement and why?

Given the scale of data the most suitable algorithm to implement would be the FP-growth algorithm. The FP-growth algorithm is more efficient than ECLAT and the Apriori algorithm, especially when dealing with large datasets. It doesn't require multiple passes over the data to generate candidate itemsets, instead it builds a compact data structure - the FP-tree to represent the transactions and efficiently mines frequent itemsets from it. Additionally, the FP-growth algorithm tends to be faster than both Apriori and Eclat, especially as the dataset size increases.

2.2

Use ECLAT algorithm to compute the frequent itemsets of the Table2 with minimum support count of 3.

TID	Items
T1	AED
T2	AEG
T3	ACGF
T4	ACGBD
T5	AEGF
T6	ECDF
T7	CDFG
T8	DFG

A	B	C	D	E	F	G
T1	T4	T3	T1	T1	T3	T2
T2		T4	T4	T2	T5	T3
T3		T6	T6	T5	T6	T4
T4		T7	T7	T6	T7	T5
T5			T8		T8	T7
						T8

AB	AC	AD	AE	AF	AG	CD	CE	CF	CG	DE	DF	DG	EF	EG	FG
T4	T3	T1	T1	T3	T2	T4	T6	T3	T3	T1	T6	T4	T5	T2	T3
	T4	T4	T2	T5	T3	T6		T6	T4	T6	T7	T7	T6	T5	T5
			T5		T4	T7		T7	T7		T8	T8			T7
					T5										T8

AEG	ACG	ADG	AFG	CDF	CDG	CFG	DFG
T2	T3	T4	T3	T6	T4	T3	T7
T5	T4		T5	T7	T7	T7	T8

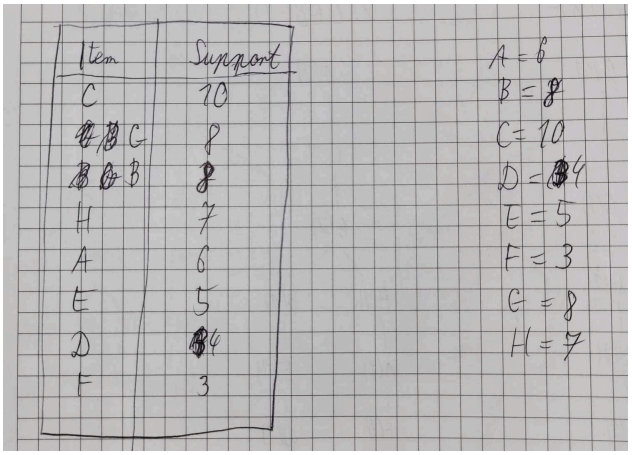
The frequent itemsets of Table 2 are: A, C, D, E, F, G, AE, AG, CD, CF, CG, DF, DG, FG.

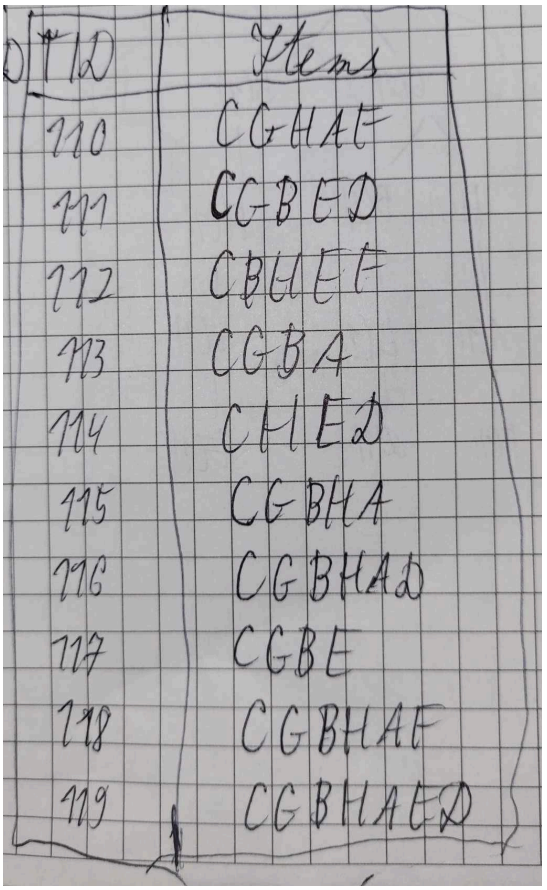
3 FP-Growth Algorithm

3.1 Assignment

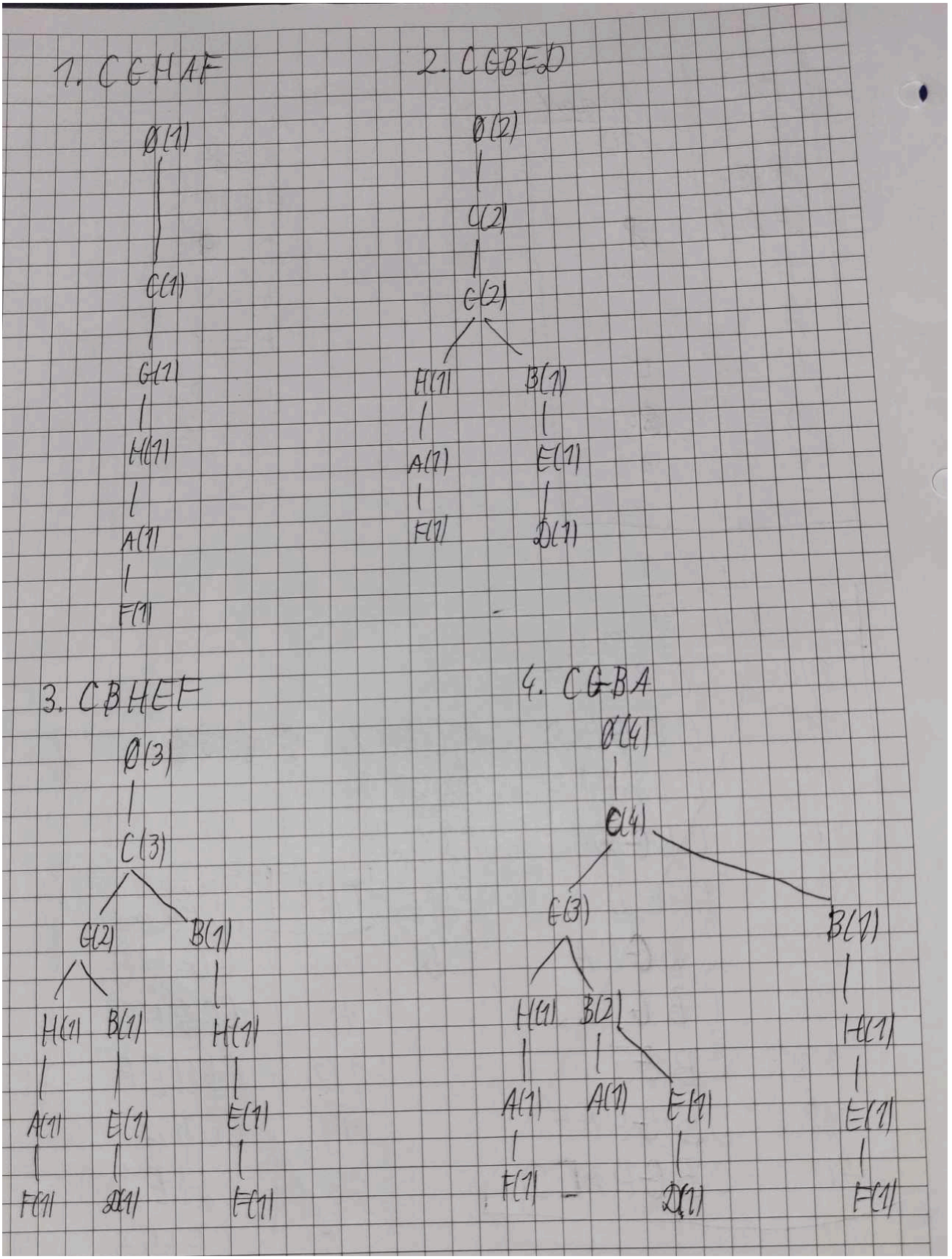
Note: We have overseen the note about the support count ties and put the item G before item B, which has resulted in slightly different trees. But according to our analysis, it didn't change the exercise's logic, because the found frequent itemsets are the same ones we would get if we would change this order. (only difference happens in Rg and Rb trees where the mined frequent itemsets from Rb tree swapped places with the frequent itemsets from Rg tree).

Ordering items and transactions by descending support count:

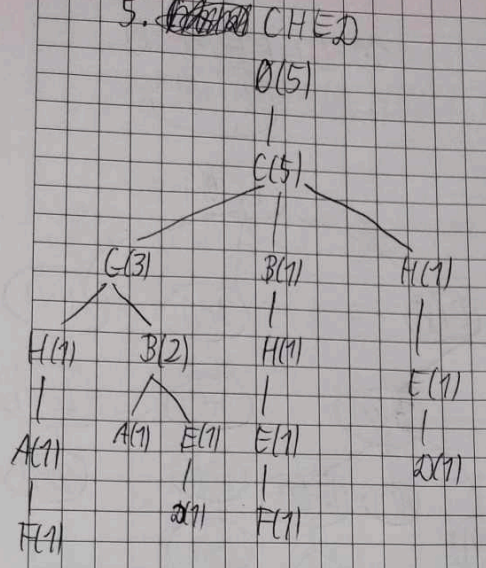




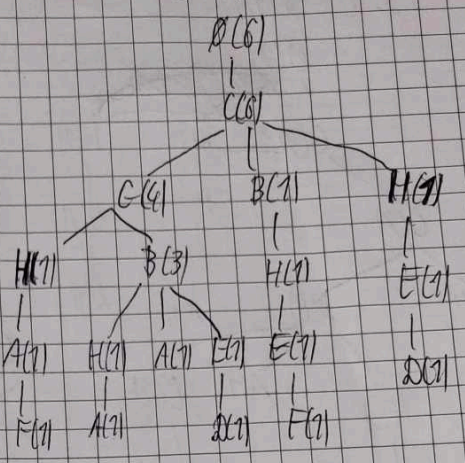
Creating main FP-tree:



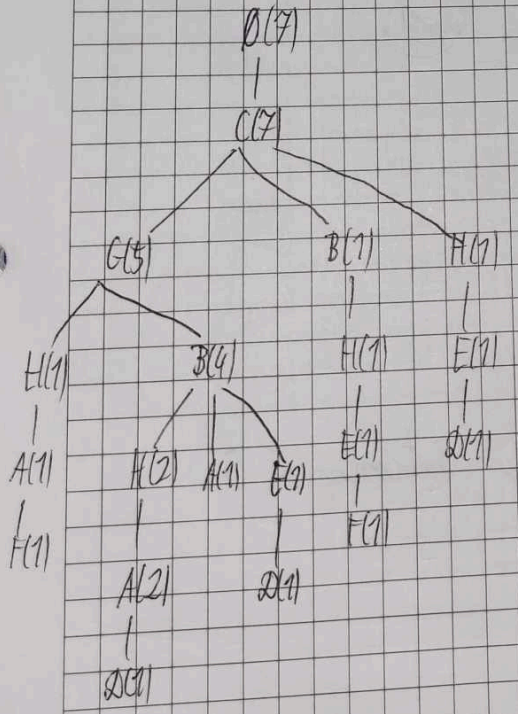
5. CHED



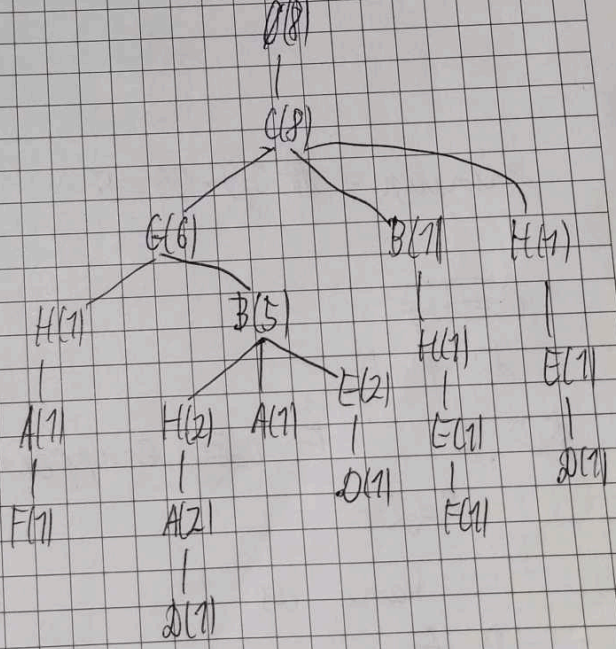
6. CGBHA



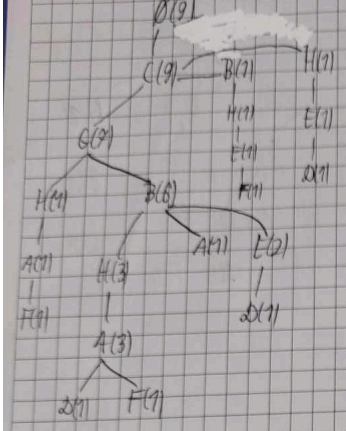
7. CGBHAD



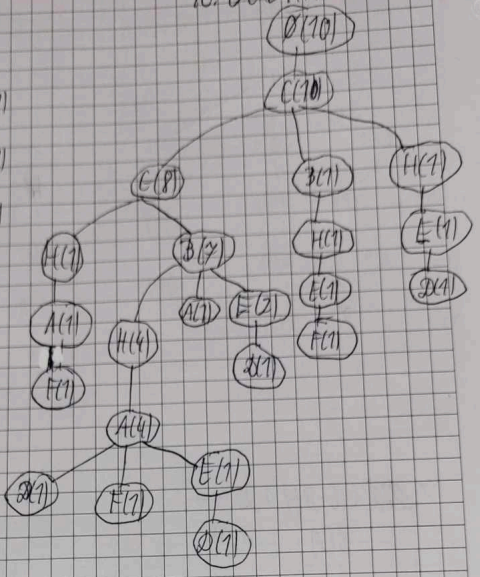
8. CGBE



9. C G B H A F



10. C G B H A E D



minsup = 0,5 · 10 = 5

P = H:

P = F:

F appears only 3 times < 5

P = D:

Same as F

P = E:

{EC(5)}

P = A:

{AC(6), AG(6), AB(5), ABC(5), AGC(6), ABGC(5), AH(5)}

P = H:

{HC(7), HG(5), HB(5), HBC(5), HGC(5)}

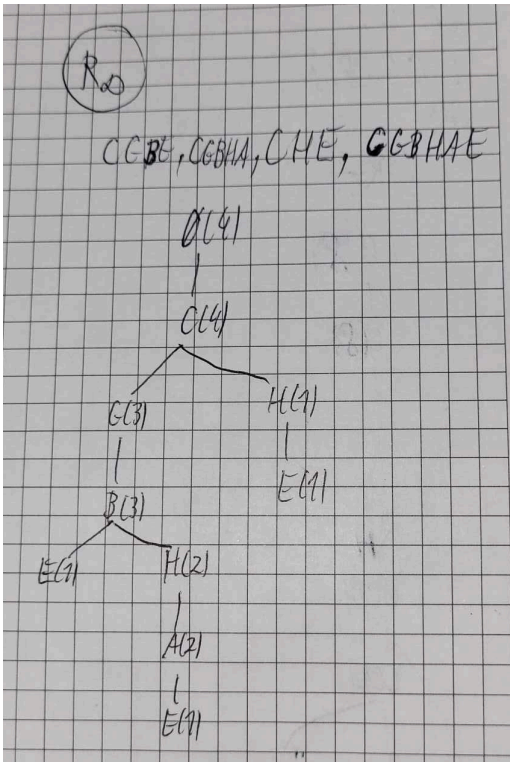
P = B:

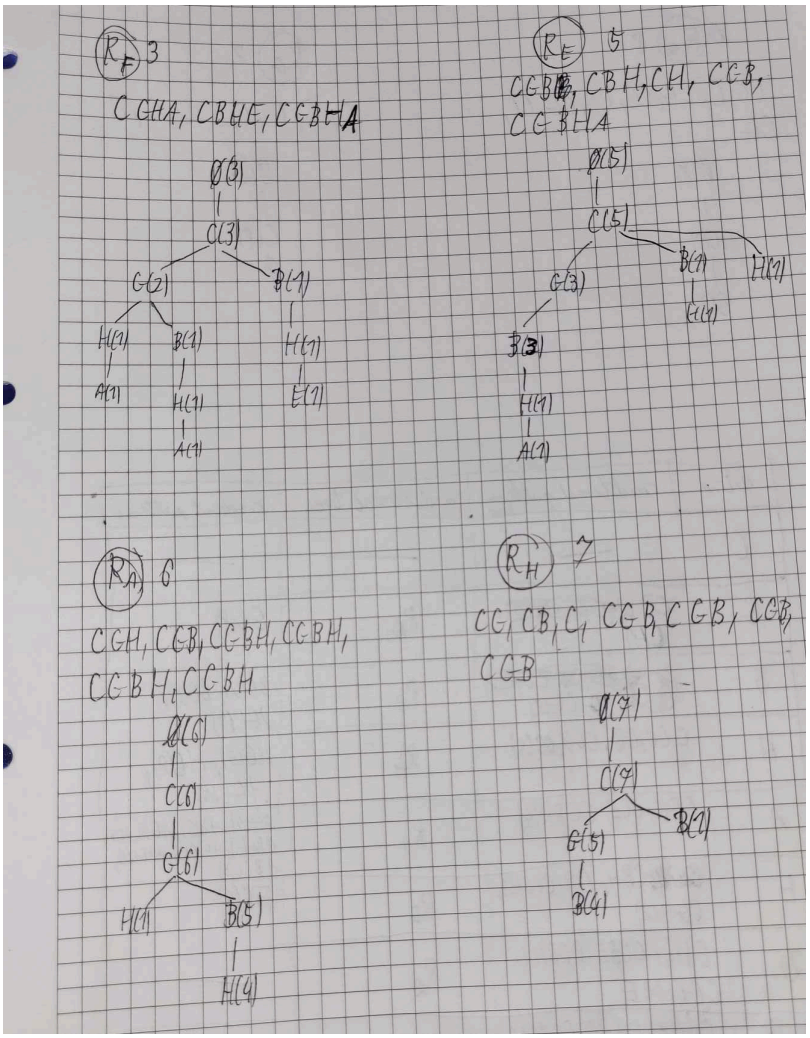
{BC(8), BG(7), BGC(7)}

P = G:

{GC(8)}

Conditional FP-trees:





R_B 8

CG, C, CG, CG, CG, CG, CG, CG

Ø(8)
|
C(8)
|
G(7)

R_G 8

C, C, C, C, C, C, C, C

Ø(8)
|
C(8)
|
8

item	conditional pattern base	conditional tree	frequent patterns
C	—	—	—
G	~~CG, CG, CG, CG, CG~~ C(8)	R_G	GC(8)
B	~~CB, CB, CB, CB~~ C, CG(7)	R_B	BC(7), BC(8), BGC(7)
H	C, CG, CB, CGB(4)	R_H	HG(5), HC(7), HGC(5)
A	CGH, CGB, CGBH(4)	R_A	AH(5), AB(5), AG(6), AC(6), ABG(5), AHG(5), AHC(5), ABGC(5)
E	~~CBH~~, CH, CGBHA, CGB(2)	R_E	EC(5)
D	CHE, CGBE, CGBHA, CGBHAE	R_D	—
F	CGHA, CBHE, CGBHA	R_F	—

4. Mlxtend

```
FP-growth Result:
    support                    itemsets
0       1.0              (Kidney Beans)
1       0.8                      (Eggs)
2       0.6                     (Bread)
3       0.6                    (Yogurt)
4       0.6                     (Onion)
5       0.8        (Eggs, Kidney Beans)
6       0.6       (Bread, Kidney Beans)
7       0.6      (Yogurt, Kidney Beans)
8       0.6               (Eggs, Onion)
9       0.6       (Onion, Kidney Beans)
10      0.6  (Eggs, Onion, Kidney Beans)

Apriori Result:
    support                    itemsets
0       0.6                     (Bread)
1       0.8                      (Eggs)
2       1.0              (Kidney Beans)
3       0.6                     (Onion)
4       0.6                    (Yogurt)
5       0.6       (Bread, Kidney Beans)
6       0.8        (Eggs, Kidney Beans)
7       0.6               (Eggs, Onion)
8       0.6       (Onion, Kidney Beans)
9       0.6      (Yogurt, Kidney Beans)
10      0.6  (Eggs, Onion, Kidney Beans)
```

5. Compact Representation of Frequent Itemsets

To solve this problem we decided to use this algorithm:

Algorithm 5.4 Support counting using closed frequent itemsets.

1: Let C denote the set of closed frequent itemsets and F denote the set of all frequent itemsets.
2: Let $k_{\max}$ denote the maximum size of closed frequent itemsets
3: $F_{k_{\max}} = \{f | f \in C,\ |f| = k_{\max}\}$ {Find all frequent itemsets of size $k_{\max}$.}
4: **for** $k = k_{\max} - 1$ down to 1 **do**
5: $F_k = \{f | f \in F,\ |f| = k\}$ {Find all frequent itemsets of size k.}
6: **for** each $f \in F_k$ **do**
7: **if** $f \notin C$ **then**
8: $f.support = \max\{f'.support | f' \in F_{k+1},\ f \subset f'\}$
9: **end if**
10: **end for**
11: **end for**

With yellow we have marked all the closed frequent itemset given in Table 4. Then, following the algorithm we denoted kmax = 4, the maximum size of the closed frequent itemsets. Then we presided to find all the itemsets of length k = kmax - 1 where k equaled 3 in our case. For each of those itemsets, that were not part of the closed frequent itemsets, we performed the operation: f.support = max{f .support|f ∈ Fk+1, f ⊂ f }. This meant, we assigned every itemset that was the subset of the closed frequent itemset from Fk+1 the same support count value. We repeated the operation until k = 1 and all the frequent itemsets where given their support count.

Items	Support Count	Items	Support Count	Items	Support Count	Items	Support Count
a	11	ab	7	abc		abcd	
b	10	ac	6	abd		abce	
c	6	ad	11	abe	7	abde	
d	13	ae	7	acd	6	acde	5
e	8	bc		ace	5	bcde	
		bd	7	ade	5		
		be	8	bcd			
		cd	6	bce			
		ce	5	bde	4		
		de	6	cde	5		