Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Thomas Seidl
Janina Sontheim, Maximilian Hünemörder

Knowledge Discovery in Databases
WS 2019/20

Exercise 10: Apriori, FP-Growth, Hash-Tree

Exercise 10-1 Apriori Algorithm

Given a set of items $\{a, b, c, d, e, f, g, h\}$ and a set of transactions T according to the following table

TID	Items
1	ag
2	bcg
3	eg
4	dg
5	dfg
6	dg
7	ag
8	ag
9	ae
10	ag
11	afh
12	af
13	ade
14	dfg

(a) Using the Apriori algorithm, compute all frequent itemsets for $minSup = 0.1$ (i.e. 2 transactions are needed for an itemset to be frequent).

k	candidate	prune	count	threshold	closed	maximal
1	a		8	a	✓	
	b		1			
	c		1			
	d		5	d	✓	
	e		3	e	✓	
	f		4	f	✓	
	g		10	g	✓	
	h		1			
2	ad		1			
	ae		2	ae	✓	✓
	af		2	af	✓	✓
	ag		4	ag	✓	✓
	de		1			
	df		2	df		
	dg		4	dg	✓	
	ef		0			
	eg		1			
	fg		2	fg		
3	aef	with ef				
	aeg	with eg				
	afg		0			
	dfg		2	dfg	✓	✓

(b) Which of the found frequent itemsets are closed/maximal? Is there a dependency between those two concepts?

Maximal implies *closed*. To this end, observe that if X is frequent and maximal, then for all $Y \supset X$ holds $supp(Y) < minSup$. As X is frequent, $supp(X) \geq minSup$. Hence, for all $Y \supset X$ holds $supp(Y) < minSup \leq supp(X)$, which implies X being closed.

Exercise 10-2 Hash-Tree

(a) **Construction**. Using the hash function

$$h(x) = x \mod 3 \tag{1}$$

construct a hash tree with maximum number of itemsets in inner nodes equal to 4 given the following set of candidates:

(1, 9, 11)	(2, 5, 10)	(3, 6, 8)	(4, 7, 9)	(6, 12, 13)	(9, 12, 14)
(1, 10, 12)	(2, 5, 12)	(3, 7, 10)	(4, 7, 13)	(6, 12, 14)	(10, 11, 15)
(2, 4, 7)	(2, 9, 10)	(3, 12, 14)	(5, 7, 9)	(8, 11, 11)	(12, 12, 15)
(2, 5, 8)	(3, 3, 5)	(4, 5, 8)	(5, 7, 13)	(8, 11, 15)	(14, 14, 15)

In the root node, the itemsets are splitted according to the hash value of the first item in the itemset. Hence, after the root node we have 3 child nodes with content:

N_0	N_1	N_2
(3, 3, 5)	(1, 9, 11)	(2, 4, 7)
(3, 6, 8)	(1, 10, 12)	(2, 5, 8)
(3, 7, 10)	(4, 5, 8)	(2, 5, 10)
(3, 12, 14)	(4, 7, 9)	(2, 5, 12)
(6, 12, 13)	(4, 7, 13)	(2, 9, 10)
(6, 12, 14)	(10, 11, 15)	(5, 7, 9)
(9, 12, 14)		(5, 7, 13)
(12, 12, 15)		(8, 11, 11)
		(8, 11, 15)
		(14, 14, 15)

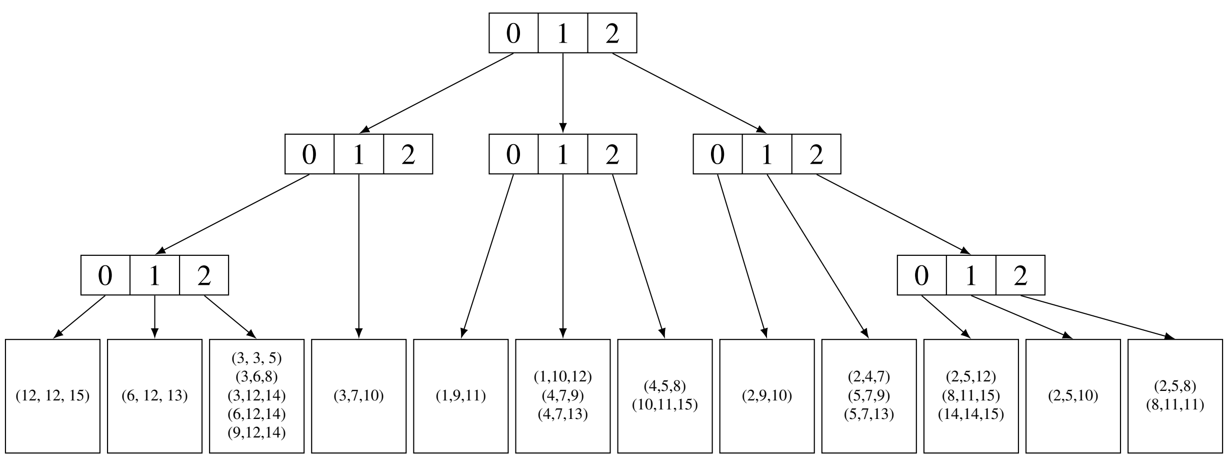
As the fill degree of all nodes is larger 4, all have to be split, now according to the second item.

N_{00}	N_{01}^*	N_{10}^*	N_{11}^*	N_{12}^*	N_{20}^*	N_{21}^*	N_{22}
(3, 3, 5)	(3, 7, 10)	(1, 9, 11)	(1, 10, 12)	(4, 5, 8)	(2, 9, 10)	(2, 4, 7)	(2, 5, 8)
(3, 6, 8)			(4, 7, 9)	(10, 11, 15)		(5, 7, 9)	(2, 5, 10)
(3, 12, 14)			(4, 7, 13)			(5, 7, 13)	(2, 5, 12)
(6, 12, 13)							(8, 11, 11)
(6, 12, 14)							(8, 11, 15)
(9, 12, 14)							(14, 14, 15)
(12, 12, 15)							

Here, only N_{00} and N_{22} have a higher fill degree than allowed (the leaf nodes are marked with *). Hence, they are splitted again, this time using the third item.

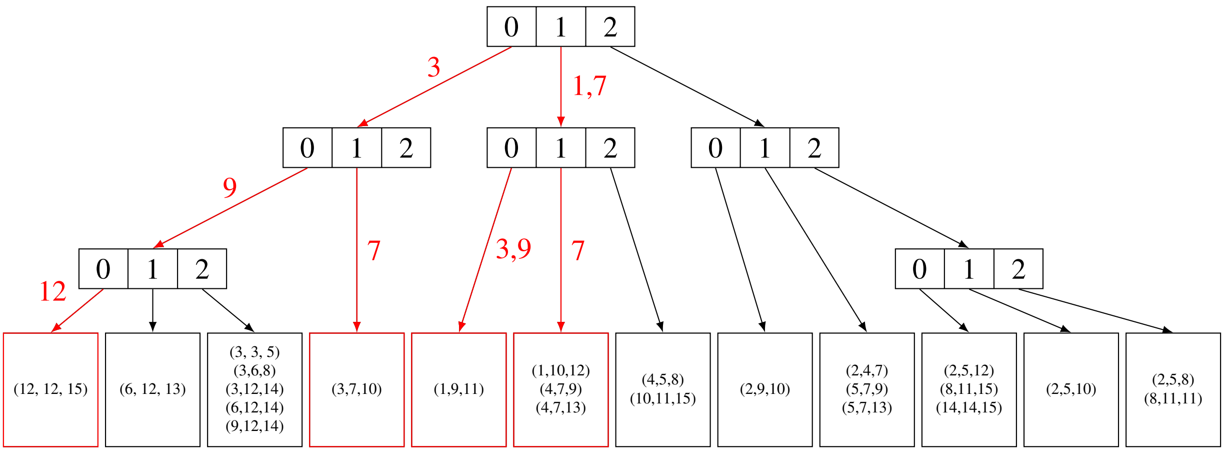
N_{000}^*	N_{001}^*	N_{002}^*	N_{220}^*	N_{221}^*	N_{222}^*
(12, 12, 15)	(6, 12, 13)	(3, 3, 5)	(2, 5, 12)	(2, 5, 10)	(2,5,8)
		(3, 6, 8)	(8, 11, 15)		(8, 11, 11)
		(3, 12, 14)	(14, 14, 15)		
		(6, 12, 14)			
		(9, 12, 14)			

Although N_{002}'s fill degree is larger then 4, there is no remaining item to be used for further splitting. Hence, the hash-tree construction finishes. The final hash-tree is depicted below:



(b) **Counting**. Given the transaction $t = (t_1, \ldots, t_5) = (1, 3, 7, 9, 12)$, find all candidates of length $k = 3$ in the previously constructed tree from exercise (a). In absolute and relative numbers: How many candidates need to be refined? How many nodes are visited?

Applying the hash function to the transaction gives $(1, 0, 1, 0, 0)$. The following diagram shows the accessed nodes. A detailed explanation follows below.



(i) Depth $d = 1$. Compute hash values for $t_1, \ldots, t_{n-k+d} = t_3$:

$$h(1) = 1 \qquad h(3) = 0 \qquad h(7) = 1 \tag{2}$$

. Continue search in N_0, N_1 (i.e. exclude N_2).

(ii) Depth $d = 2$. Additionally compute $h(t_4) = h(9) = 0$.

- In N_0 reached by item t_2, the nodes for hash values 0 (N_{00} reached by t_4) and 1 (N_{01}^* reached by t_3) are of interest.
- In N_1 reached by item t_1 and t_3, the nodes for hash values 0 (N_{10}^* reached by t_2 and t_4) and 1 (N_{11}^* reached by t_3) are of interest.

(iii) Depth $d = 3$. Additionally compute $h(t_5) = h(12) = 0$.

- In N_{00} reached by $t_2, t_4 = 3, 9$ continue with N_{000}^*.
- In N_{01}^* reached by $t_2, t_3 = 3, 7$ search for $t_2, t_3, t_4 = 3, 7, 9$ and $t_2, t_3, t_5 = 3, 7, 12$. Both are not found.
- In N_{10}^* reached by
  - $t_1 t_2 = 1, 3,$
  - $t_1 t_4 = 1, 9$, or
  - $t_3 t_4 = 7, 9$

  search for
  - $t_1 t_2 t_3 = 1, 3, 7$
  - $t_1 t_2 t_4 = 1, 3, 9$
  - $t_1 t_2 t_5 = 1, 3, 12$
  - $t_1 t_4 t_5 = 1, 9, 12$
  - $t_3 t_4 t_5 = 7, 9, 12$

  None of them is found.
- In N_{11}^* reached by $t_1, t_3 = 1, 9$ search for $t_1, t_3, t_4 = 1, 7, 9$ and $t_1, t_3, t_5 = 1, 7, 12$. Both are not found.

(iv) Depth $d = 4$.

- In N_{000}^* reached by $t_2, t_4, t_5 = 3, 9, 12$ search for this transaction. It is not found.

In total, $4/12 \approx 33\%$ of the leaf nodes are visited, $8/18 \approx 44\%$ of the nodes are visited and $6/24 = 25\%$ of the candidates are compared. As result, none of the candidates is supported by the transaction.

Exercise 10-3 FP-Tree and FP-Growth Algorithm

Given a set of items $\{a, b, c, d, e, f, g, h\}$ and a set of transactions T according to the following table, construct the FP-tree and use the FP-Growth algorithm to compute all frequent itemsets for $minSup = 0.1$ (i.e. 2 transactions are needed for an itemset to be frequent).

TID	Items
1	ag
2	cg
3	eg
4	dg
5	bdfg
6	dg
7	ag
8	ag
9	ae
10	ag
11	afh
12	af
13	ade
14	bdfg

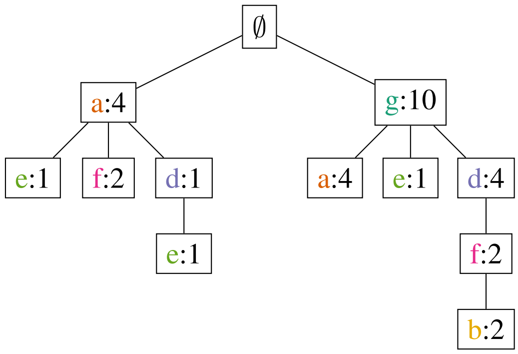
1. Scan database, count frequency of single items, remove infrequent, and sort the items by decreasing frequency.

Item	Frequency	
g	10	
a	8	
d	5	frequent
f	4	
e	3	
b	2	
c	1	infrequent
h	1	

2. Scan database again: Remove infrequent items from itemsets, and sort them descending by frequency (although the algorithm constructs the FP-Tree on-the-fly, this is done in the next step for more clarity).

TID	Items
1	ga
2	g
3	ge
4	gd
5	gdfb
6	gd
7	ga
8	ga
9	ae
10	ga
11	af
12	af
13	ade
14	gdfb

3. Construct FP-Tree:



Hint: In order to check the correctness of the FP-tree construction you can verify:

- The most frequent item has only a single node directly under the root.
- The sum of counts for each item equals the total count calculated in the step before.
- The sum of counts of the children of a node is less than or equal to the count of the node itself.
- There are x itemsets having prefix p before y, where y is the label of a node in the tree, p is the prefix on the path from the root, and x the count of the node.

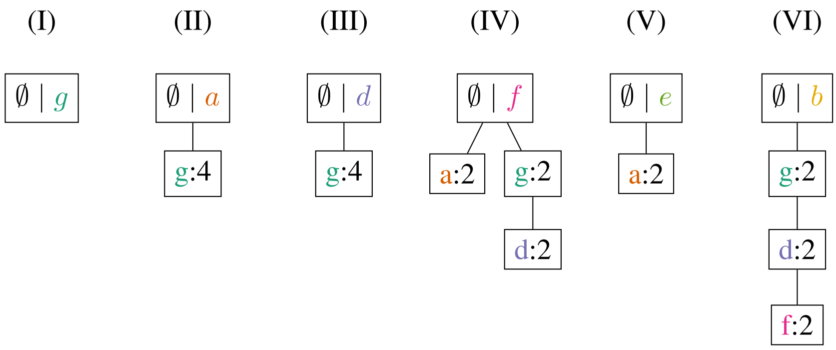
4. In order to extract the frequent patterns from the FP-tree, the FP-Growth algorithm is used. We start by constructing the conditional pattern base:

Item	Conditional Pattern Base
g	∅
a	g:4, ∅
d	a:1, g:4
f	a:2, gd:2
e	a:1, g:1, ad:1
b	gdf:2

5. Here, all conditional patterns with too small support are pruned:

Item	Conditional Pattern Base
g	∅
a	g:4, ∅
d	g:4
f	a:2, gd:2
e	a:2
b	gdf:2

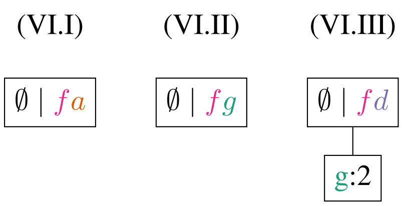
6. For each item we build a conditional FP-tree.



7. If the FP-tree is a single path, we can enumerate all frequent patterns:
   (II) ag
   (III) dg
   (V) ae
   (VI) bd, bf, bg, bdf, bdg, bfg, bdfg

8. For conditional FP-Tree (IV) we have to recurse. We first count f as frequent pattern, and then build the conditional pattern base for f:

Item	Conditional Pattern Base
a	∅
g	∅
d	g:2



All resulting FT-trees are linear, yielding patterns: fa, fg, fd, fdg

In total the frequent patterns are (in shortlex ordering[1]):

- a, b, d, e, f, g
- ae, af, ag, bd, bf, bg, dg, df, fg
- bdf, bdg, bfg, dfg
- bdfg

[1]The shortlex ordering first orders words by length, and within the same length lexicographic.