May 13, 2021

1.DFS Listing

First we find all frequent items and delete all infrequent items from the database

item	count
A	1
C	2
D	1
E	4
K	5
M	3
N	2
O	3
U	1
Y	3

So F = {E, K, M, O, Y}
we start with the database and $\emptyset$
so the new database is

TID	items
1	M,O,K,E,Y
2	O,K,E,Y
3	M,K,E
4	M,K,Y
5	O,K,E

and we have a total order E < K < M < O < Y
we take E and remove it from the database with every element that is smaller than it and we add E to the frequent list
the projected database is

TID	items
1	M,O,K,Y
2	O,K,Y
3	M,K
5	O,K

and F ={E}
we can see that M and Y are not frequent in the projected database so we delete them then we take k and add EK to the frequent list

TID	items
1	O
2	O
5	O

Now F ={E, EK}
then we add EKO to the frequent list and no more databases can be projected so we backtrack to E
Now F = {E, EK, EKO}
Only O is frequent so we add EO to the frequent list and delete O and all items smaller than it from the database so the projected database is

TID	items
1	M,Y
2	Y

and F ={E, EK, EO, EKO}
in the previous projected database there are no frequent items so we backtrack to the first database because there are no other possibilities for E so we take another element
Now we take K and remove it with all items smaller than it from the database and add it to the frequent list
so F = {E,K,EK,EO,EKO}
and the new database is

TID	items
1	M,O,Y
2	O,Y
3	M
4	M,Y
5	O

We take M and add KM to F so F ={E, K, EK, EO, KM, EKO}
and the projected database is

TID	items
1	O,Y
4	Y

None of the items is frequent so we backtrack to K
Now We take O and add KO to F so F ={E, K, EK, EO, KM,KO, EKO}
and the projected database is

TID	items
1	Y
2	Y

None of the items is frequent so we backtrack to K
and then we take Y and add kY to F so F ={E, K, EK, EO, KM,KO,KY, EKO}
and the projected database is empty so we backtrack to K and then backtrack again to take another item which is M and we add M to F so F ={E,K,M, EK, EO, KM,KO,KY, EKO}
and the projected database is

TID	items
1	O,Y
4	Y

None of the items is frequent so we backtrack to the first database and take O and add it to F
so F ={E,K,M,O, EK, EO, KM,KO,KY, EKO}
and the projected database is

TID	items
1	Y
2	Y

None of the items is frequent so we backtrack to the first database and take Y and add to F so F ={E,K,M,O,Y, EK, EO, KM,KO,KY, EKO}
and as Y is the biggest item the projected database will be empty and the algorithm stops with F ={E,K,M,O,Y, EK, EO, KM,KO,KY, EKO}

Exercise 2

Claim

The *DFS_Listing* Algorithm is correct and enumerates all frequent itemsets with polynomial delay.

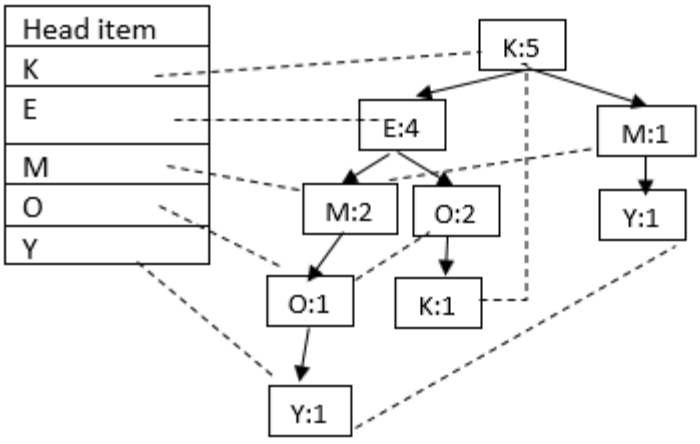
Proof

Correctness: The itemsets are printed in step 1: "print F". At first the emptyset gets printed what is trivially correct. Then all infrequent items from D are removed and the algorithm gets called with D_i and $F \cup \{i\}$. Where D_i is the set of all transactions that contain i restricted to those that are strictly greater then i according to the linear order defined on the items. In the next step $F \cup \{i\}$ gets printed. That are all frequent items in D (frequent itemsets of size 1) since $F = \emptyset$. In general each step takes a frequent k-itemset F and appends a frequent item i and get $F \cup \{i\}$. Since F is a frequent itemset and D_i contains itemsets that are locally frequent without the item i the union of that is frequent in the complete transaction set. This is the general idea of that algorithm: Find all itemsets that are frequent containing the first item according to the linear order and then find those that are frequent containing the second item and not the first and then build then join those to find all frequent itemsets int the transaction table. $\Rightarrow$ the algorithm prints only frequent itemsets.
Now assume there exists an itemset F' that is frequent but does not get printed by the algorithm. That means there is no frequent F'' and frequent i such that $DFS\_Listing(D_i, F'' \cup \{i\})$ is called. That means F'' gets not printed. If F' is a k-itemset then F'' is a k-1-itemset. When we continue that we have a frequent 1-itemset that does not get printed. But that is a contradiction because every frequent 1-itemset j gets printed via $F = \emptyset \cup \{j\}$. $\Rightarrow$ every frequent itemset gets printed. Now assume G is not frequent but gets printed. Then there needs to be a frequent itemset F and a frequent item i such that $G = F \cup \{i\}$ but thats not possible because F is locally frequent in D without i and i is frequent then the union has to be frequent too.
$\Rightarrow$ the algorithm is correct.
Complexity: The print statement is the first one that gets called. So we need to proof that step 2-5 have polynomial time complexity. Lets say $|D| = n$. Removing all infrequent items from a set of size at most n can be done polynomially in n (quadratic). Defining a linear order is a linear operation. Depending on the implementation one might want to order that. But we know sorting algorithms with polynomial runtime. In the next step the alogorithm loops over all sets in D_i. In each iteration the set D_i is defined by picking all itemsets in D that contain elements greater then i this can easily be done in linear time when we sort (in poly time) D beforehand. Nevertheless this can be done in polynomial time. After that *DFS_Listing* gets called again and the next frequent set gets printed. $\Rightarrow$ the algorithm has polynomial delay.
Q.E.D.

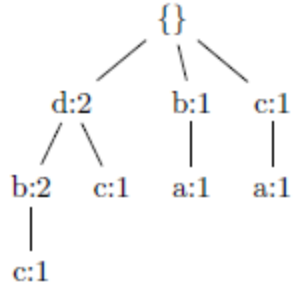
3.FP-Tree

(i)
First of all we order the frequent items according to the frequency
I' = {K:5, E:4, M:3, O:3, Y:3 }

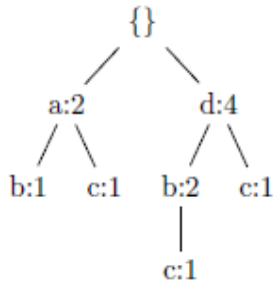
TID	ordered items
1	K,E,M,O,Y
2	K,E,O,Y
3	K,E,M
4	K,M,Y
5	K,E,O



(ii)
with t = 1 all items are frequent withe the following frequencies
I' = {d:4, b:3, c:3, a:2 }
Then we order the items in the database according to the frequency
D'={ba, ca, dbc, db, dc, d }
Now we construct the FP-Tree



The tree consists of 8 nodes without the node {}
Now if we order the database like this
D"={ba, ca, dbc, db, dc, d}
then we get the following tree



and we see that the tree with the previous heuristic consists of 7 nodes without the node {}
so ordering the items according the frequency didn't give the minimal number of nodes

Exercise 4

We define the set T of transaction identifiers with $T = V_1$ and the groundset I with $I = V_2$. Now we create a transaction database D in which the entry with tid v_1 contains all items v_2 so that $(v_1, v_2) \in E$. D must have size polynomial in $|V_1|$ and $|V_2|$ because the number of rows is bounded by $|V_1|$ and the number of items per row is bounded by $|V_2|$.
If there are k elements in V_1 that all connect with the same k elements from V_2, there must be k rows/tids containing the same itemset. And if there are k rows containing elements of the same k-subset of V_2, all of the vertices corresponding to the tids of these rows must be connected to each vertex in this k-subset. All in all, the existence of a balanced bibartite clique of size k is equivalent to the existence of a k-frequent itemset of size at least k.
In conclusion, it cannot be decided in polynomial time if there is a frequent itemset consisting of at least n items for a threshold t. Otherwise, one could solve the Balanced Bipartite Clique Problem in polyonomial time by building the transaction database D and checking for the existence of a k-frequent itemset of size at least k.