LINKÖPING UNIVERSITY

723A75 Advanced Data Mining
TDDD41 Data Mining - Clustering and Association Analysis

Lecture 7: FP Grow Algorithm

Johan Alenlöv
IDA, Linköping University, Sweden

Outline

- Content
  - Recap
  - Frequent Pattern (FP) Grow Algorithm
  - Exercise
  - Summary
- Litterature
  - Course Book. 2nd ed.: 5.2.4. 3rd ed.: 6.2.4
  - Han, J., Pei, J., and Yin, Y. **Mining Frequent Patterns without Candidate Generation**. In Proc. of the 2000 ACM SIGMOD Int. Conf. on Management of Data, 2000.

Recap

- Given a database of transactions we want to find association rules,

$$\text{Item}_1, \ldots, \text{Item}_m \rightarrow \text{Item}_{m+1}, \ldots, \text{Item}_n, (X \rightarrow Y)$$

  with a **user-specified** minimum **support** and **confidence**.
- **support**: Fraction of transactions that contains th full rule $\text{Item}_1, \ldots, \text{Item}_n$. $(p(X, Y))$
- **confidence**: Fraction of transactions that contain $\text{Item}_1, \ldots, \text{Item}_m$ which also contain $\text{Item}_{m+1} \rightarrow \text{Item}_n$. $(p(Y \mid X))$
- We find the rules in two steps:
  1. Find all **frequent itemsets**
  2. Find all rules with minimum confidence from these sets.

Recap: Apriori Algorithm

- Using the following **apriori propeerty**:
  - Every subset of a frequent itemset is frequent.
  - Alternatively, every superset of an infrequent itemset is infrequent.
- The Apriori Algorithm works as follows:
  1. Find all 1-itemsets.
  2. Use the previous found frequent itemsets and the apriori property to generate candidates for the next frequent itemsets.
  3. Go through the candidates to find the itemsets.

  Step 2 and 3 are repeated until no new frequent itemsets are found.
- We proved by induction that the algorithm is correct.

Recap: Generate rules

- Given a large itemset L we wish to generate rules

$$X \rightarrow L \setminus X,$$

  where $X \subset L$.
- These rules should have a **minimum confidence**.
- The algorithm uses the following **apriori property**:
  - If X does not result in a rule with minimum confidence for L, then neither does **any subset** $X' \subset X$,

$$\text{confidence}(X \rightarrow L \setminus X) = \frac{\text{support}(L)}{\text{supportX}} \geq \frac{\text{support}(L)}{\text{support}(X')} = \text{confidence}(\text{X}' \rightarrow \text{L} \setminus \text{X}')$$

```
1   for all large itemsets l_k with k ≥ 2 do
2       call genrules(l_k, l_k, minconf)

    Algorithm: genrules(l_k, a_m, minconf)
    Input: A large itemset l_k, a set a_m ⊆ l_k, the minimum confidence minconf.
    Output: All the rules of the form a → l_k \ a with a ⊆ a_m and confidence equal or above minconf.

1   𝔸 = {(m − 1)-itemsets a_{m−1} | a_{m−1} ⊆ a_m}
2   for all a_{m−1} ∈ 𝔸 do
3       conf = support(l_k) / support(a_{m−1})          // Confidence of the rule a_{m−1} → l_k \ a_{m−1}
4       if conf ≥ minconf then
5           output the rule a_{m−1} → l_k \ a_{m−1} with confidence=conf and support=support(l_k)
6           if m − 1 > 1 then call genrules(l_k, a_{m−1}, minconf)
```

Rule Generation Algorithm Proof

```
1  for all large itemsets l_k with k ≥ 2 do
2      call genrules(l_k, l_k, minconf)

   Algorithm: genrules(l_k, a_m, minconf)
   Input: A large itemset l_k, a set a_m ⊆ l_k, the minimum confidence minconf.
   Output: All the rules of the form a → l_k \ a with a ⊆ a_m and confidence equal or above minconf.

1  𝔸 = {(m − 1)-itemsets a_{m−1} | a_{m−1} ⊆ a_m}
2  for all a_{m−1} ∈ 𝔸 do
3      conf = support(l_k) / support(a_{m−1})        // Confidence of the rule a_{m−1} → l_k \ a_{m−1}
4      if conf ≥ minconf then
5          output the rule a_{m−1} → l_k \ a_{m−1} with confidence=conf and support=support(l_k)
6          if m − 1 > 1 then call genrules(l_k, a_{m−1}, minconf)
```

- We prove by contradiction that the rule generation algorithm is correct.
- Assume that the algorithm missed a rule. Let $a_{m-1} \rightarrow l_k \setminus a_{m-1}$ denote one of the missing rules with the largest antecedent. Then,
  - Note that l_k has minimum support and, thus, it is outputted by the apriori algorithm since this is correct.
  - Then, the rule generation algorithm cannot have missed the rule if $m = k$.
  - Moreover if $m < k$, then

$$\begin{aligned}\text{confidence}(a_m \rightarrow l_k \setminus a_m) &= \text{support}(l_k)/\text{support}(a_m) \geq \text{support}(l_k)/\text{support}(a_{m-1}) \\ &= \text{confidence}(a_{m-1} \rightarrow l_k \setminus a_{m-1}) \geq \textit{minconf}.\end{aligned}$$

  - Note that the algorithm didn't miss the rule $a_m \rightarrow l_k \setminus a_m$.
  - Then the algorithm couldn't have missed the rule $a_{m-1} \rightarrow l_k \setminus a_{m-1}$.
  - This contradicts our assumption and, thus, the algorithm is correct.

FP Grow Algorithm

- As previous, assume that we have access to some transactional data,

Tid	Items
1	F, A, C, D, G, I, M, P
2	A, B, C, F, L, M, O
3	B, F, H, J, O, W
4	B, C, K, S, P
5	A, F, C, E, L, P, M, N

- The FP grow algorithm returns all frequent itemsets **without candidate generation** and may save time and space.
- First, it finds frequent 1-itemsets and **sorts the frequent items within each transaction in support decending order**, e.g. with $minsup = 3$

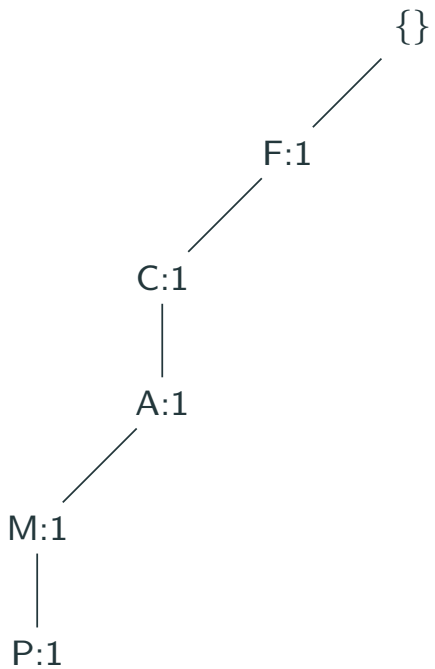
Tid	Items
1	F, C, A, M, P
2	F, C, A, B, M
3	F, B
4	C, B, P
5	F, C, A, M, P

- Then it outputs the frequent 1-itemsets, F, C, A, B, M, and P.

FP Grow Algorithm

- Given the new sorted set it constructs a so-called FP tree.

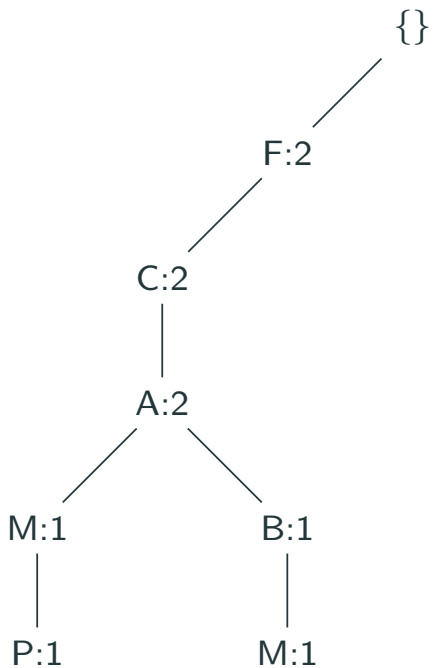
Tid	Items
1	**F, C, A, M, P**
2	F, C, A, B, M
3	F, B
4	C, B, P
5	F, C, A, M, P



FP Grow Algorithm

- Given the new sorted set it constructs a so-called FP tree.

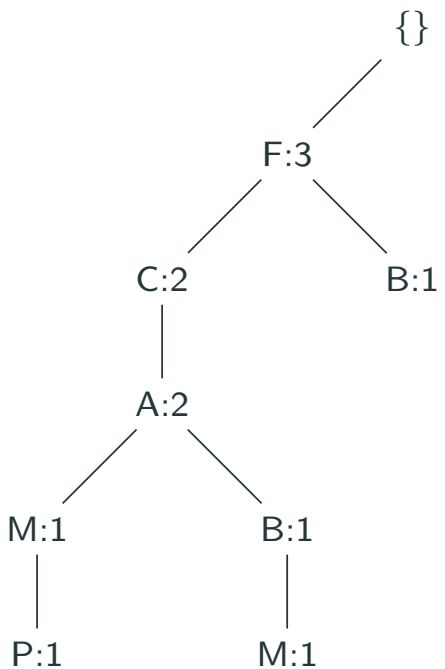
Tid	Items
1	F, C, A, M, P
2	**F, C, A, B, M**
3	F, B
4	C, B, P
5	F, C, A, M, P



FP Grow Algorithm

- Given the new sorted set it constructs a so-called FP tree.

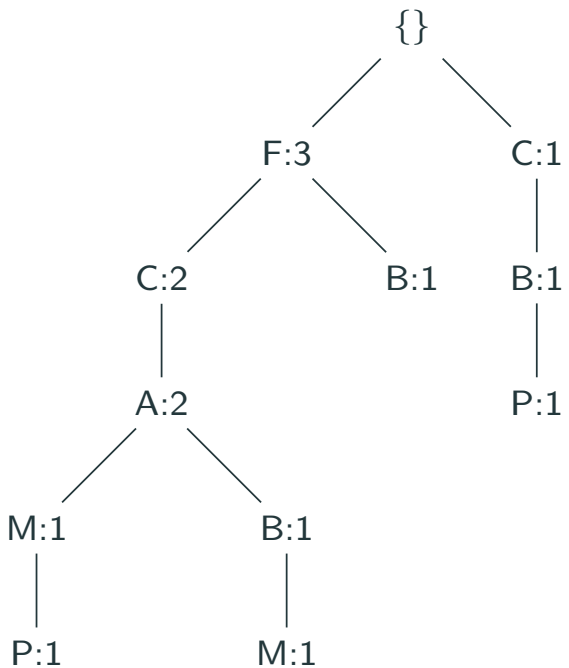
Tid	Items
1	F, C, A, M, P
2	F, C, A, B, M
3	**F, B**
4	C, B, P
5	F, C, A, M, P



FP Grow Algorithm

- Given the new sorted set it constructs a so-called FP tree.

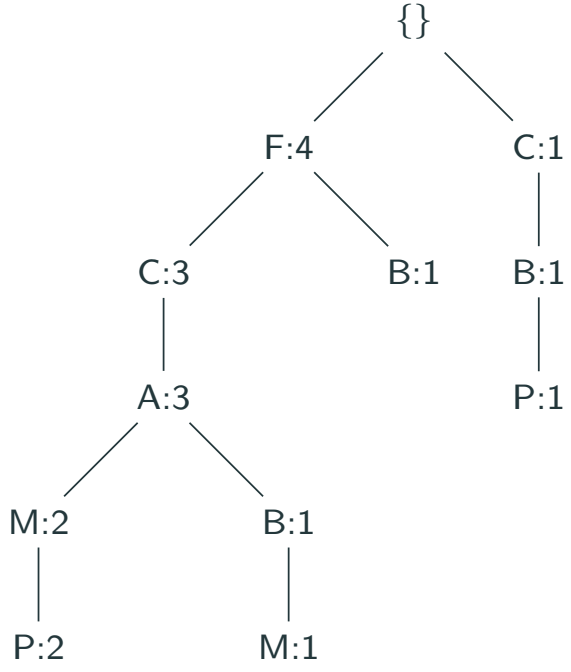
Tid	Items
1	F, C, A, M, P
2	F, C, A, B, M
3	F, B
4	**C, B, P**
5	F, C, A, M, P



FP Grow Algorithm

- Given the new sorted set it constructs a so-called FP tree.

Tid	Items
1	F, C, A, M, P
2	F, C, A, B, M
3	F, B
4	C, B, P
5	**F, C, A, M, P**



- Finally, it mines the FP tree for frequent itemsets instead of the original database.

FP Grow Algorithm

Algorithm: FP-tree(D, *minsup*)
Input: A transactional database D, and the minimum support *minsup*.
Output: The FP tree for D and *minsup*.

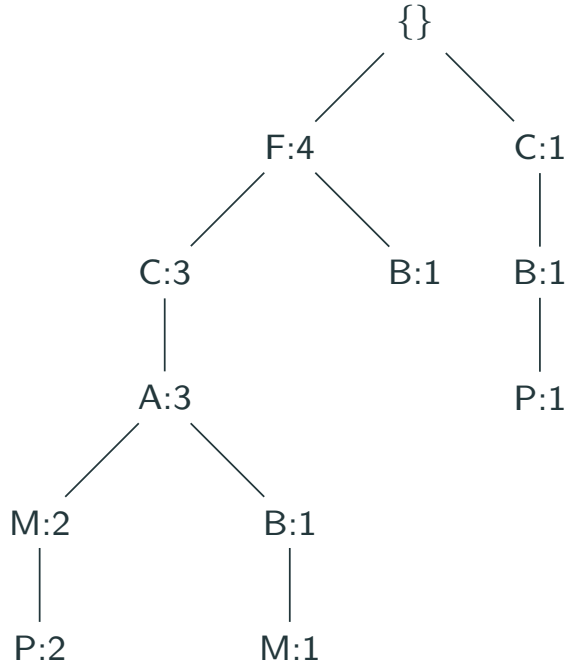
1 Count support for each item in D
2 Remove the infrequent items from the transactions in D
3 Sort the items in each transaction in D in support descending order
4 Create a FP tree with a single node T with $T.name = NULL$
5 for each transaction $I \in D$ do
6 insert-tree(I, T)

Algorithm: insert-tree($I_1, \ldots I_m$, T)
Input: An itemset $I_1, \ldots, I_m$, and a node T in the FP tree.
Output: Modified FP tree.

1 if T has a child N such that $N.name = I_1.name$ then
2 $N.count++$
3 else
4 create a new child N of T with $N.name = I_1.name$ and $N.count = 1$
5 if $m > 1$ then
6 insert-tree $(I_2, \ldots, I_m, N)$

FP Grow Algorithm

- The X-conditional database consists of all the prefix paths leading to X in the FP tree.



Item	Conditional database
F	-
C	F:3
A	FC:3
B	FCA:1, F:1, C:1
M	FCA:2, FCAB:1
P	FCAM:2, CB:1

- The support of each prefix path in the conditional database is equal to the count of X for that prefix path.
- The X-conditional database contains all the itemsets in D that end with X.
- It is enoug hto mone the X-conditional database to find all the frequent itemsets in D that end with X.
- Re-start the algorithm for the X-conditional database, i.e. call the FP grow algorithm recursively.

FP Grow Algorithm

- If we look at the M-conditional database, ($\{FCA : 2, FCAB : 1\}$)

Tip	Items
1	F, C, A
2	F, C, A
3	F, C, A, B

- After finding the frequent 1-itemsets and sorting the transactions we have

Tid	Items
1	F, C, A
2	F, C, A
3	F, C, A

- Output the frequent 1-itemsets, adding M as suffix (FM, CM, AM)
- Build the FP tree and the conditional databases.

Item	Conditional database
F	-
C	F:3
A	FC:3

- Restart the algorithm for the FM, CM, and AM condtional databases.

FP Grow Algorithm

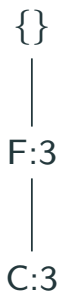
- For the AM-conditional database ({FC : 3}), or

Tid	Items
1	F,C
2	F,C
3	F,C

- After finding the 1-itemsets and sorting the transactions we have

Tid	Items
1	F,C
2	F,C
3	F,C

- Output th 1-itemsets, adding AM as a suffix. (FAM and CAM).
- Build the FP tree and the econditional databases.



Item	Conditional database
F	-
C	F:3

- Restart the algorithm for the FAM and CAM conditional databases.

FP Grow Algorithm

- For the CAM-conditional database ($\{F : 3\}$), or

Tid	Items
1	F
2	F
3	F

- After finding the 1-itemsets and sorting the transactions we have

Tid	Items
1	F
2	F
3	F

- Output th 1-itemsets, adding CAM as a suffix. (FCAM).
- Build the FP tree and the econditional databases.

Item	Conditional database
F	-

- Conditional database is empty.
  Backtrack.

FP Grow Algorithm

- To mine the FP tree *Tree*, call FP-grow(*Tree*, NULL, *minsup*).

Algorithm: FP-grow(*Tree*, α, *minsup*)
Input: A FP tree *Tree*, an itemset α, and the minimum support *minsup*.
Output: All the itemsets in *Tree* that end with α and have *minsup*.

```
1   for each item X in Tree do
2       output the itemset β = X ∪ α with support=X.count
3       build the β conditional database and the corresponding FP tree Tree_β
4       if Tree_β is not empty then call FP-grow(Tree_β, β, minsup)
```

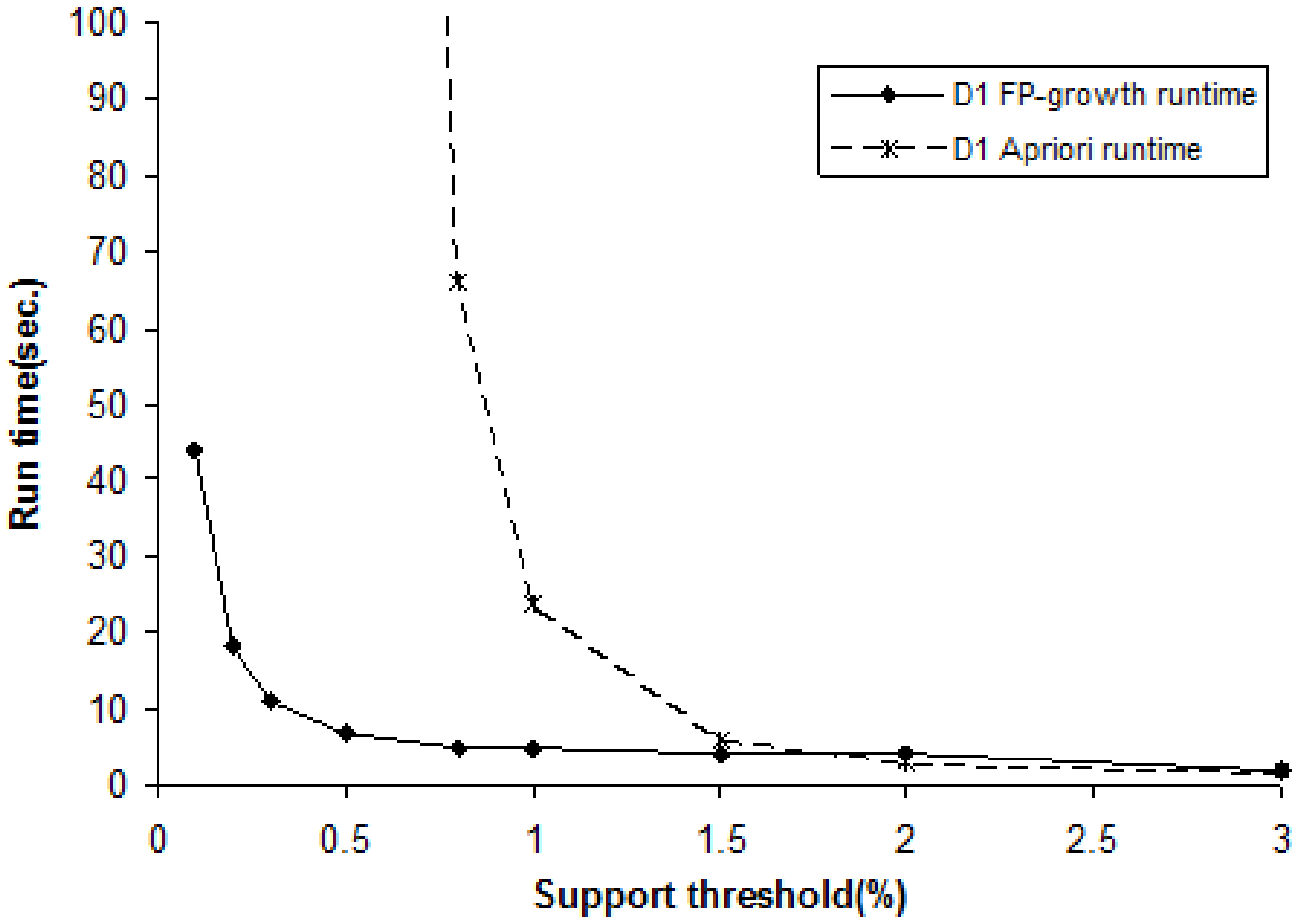
- The algorithm above can be made more efficient by adding the lines below.

```
0.1   if Tree has a single branch then
0.2       for each combination β of the nodes in the branch do
0.3           output the itemset β ∪ α with support = min_{X∈β} X.count
0.4   else
```

- The FP grow algorithm is correct.

FP Grow Algorithm

- With small values for *minsup*, there are many and long candidates, which implies long runtime due to expensive operations such as pattern matching, subset checking, storing, etc.



Exercise

- Run the FP grow algorithm on the database below with *minsup* 2.

Tid	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

- Show the execution details (i.e. FP tree construction, conditional databases, recursive calls), not just the frequent itemsets found.

Summary

- Mining transactions to find rules of the form

$$\text{Item}_1, \ldots, \text{Item}_m \rightarrow \text{Item}_{m+1}, \ldots, \text{Item}_n$$

  with **user-defined minimum support** and **confidence**.
- Two-step solution:
  1. Find all the large itemsets.
  2. Generate all the rules with minimum confidence.
- We have seen two solutions for step 1. Apriori and FP grow algorithm.
- The runtimee can differ a lot for small values of *minsup*.