Unit 2
Association Analysis

Basic concept, Use of Association Analysis, Apriori algorithm, pruning

Objective

- Association Analysis Concepts
- Application
- Algorithm for Association Analysis
- Solving a problem

Corelation between data

- It is the measure of degree of dependency between two variables
- Statistical Approach
  - Corelation Analysis
    - statistical method used to measure the strength of the linear relationship between two variables and compute their association
    - A high correlation points to a strong relationship between the two variables, while a low correlation means that the variables are weakly related

Problems

- Market Basket Analysis
  - How to arrange the items (placements) to increase the cross-selling opportunities
  - Where to display the new items
    - For example, if customers often buy bread and milk together, a store might place these items closer to each other
- Cross-Selling in Online Retail
  - Suggest additional products to customers based on the items they have added to their shopping carts or purchased.
  - This helps in increasing revenue through cross-selling.

Problems

- Customer Behavior Analysis
  - Understand purchasing patterns and preferences of customers to improve marketing strategies and personalized recommendations
    - This is widely used in e-commerce and online platforms to enhance the user experience

Problems

- Fraud Detection
  - Identify unusual patterns or associations in financial transactions that may indicate fraudulent activities.
  - For example, detecting instances where certain products are consistently bought together in fraudulent transactions.

Problems

- Telecommunications Network Optimization
  - Analyze call records to identify patterns of co-occurring calls and optimize network performance, leading to better resource allocation and improved service quality

Association Analysis

- Mining for associations among items in a large database of transactions is an important data mining function
- Association rules are statements of the form
  - {X1, X2, …, Xn} => Y, meaning that if we

    find all of X1, X2, ……… , Xn in the transaction then we have good chance of finding Y.
- Association analysis mostly applied in the field of market basket analysis, web-based mining, intruder detection

Market Basket Analysis

- It is the study of items that are purchased or grouped together in a single transaction or multiple, sequential transactions
- Used for
  - Make recommendations
  - Cross-sell
  - Up-sell
  - Offer coupons / discounts

Market Basket Analysis

- The analysis can be applied in various ways:
  - Develop combo offers based on products sold together.
  - Organize and place associated products/categories nearby inside a store.
  - Determine the layout of the catalog of an e-commerce site.
  - Control inventory based on product demands and what products sell together.

Few Terminologies

- Support
  - The support of an association pattern is the percentage of task-relevant data transaction for which the pattern is true

    Support (A): Number of tuples containing A / Total number of tuples

    Support (A = > B): Number of tuples containing A and B / Total number of tuples

  - While computing the association Minimum Support is used as threshold for computing

Few Terminologies

- Support
  - If minimum support is set too high, we could miss itemsets involving interesting rare items
    - e.g., expensive products
  - If minimum support is set too low, it is computationally expensive and the number of itemsets is very large

Few Terminologies

- Confidence
  - Confidence is defined as the measure of certainty or trustworthiness associated with each discovered pattern

    Confidence (A = > B): Number of tuples containing A and B / Total count of A

  - Confidence is usually given in percentage

Few Terminologies

- Item Set
  - A collection of one or more items.

    Example: {Milk, Bread, Diaper}

  - An itemset that contains k items is called k-itemset

- Frequent Itemset
  - An itemset whose support is greater than or equal to a minimum support threshold.

Few Terminologies

- Association Rule
  - An implication expression of the form X => Y, where X and Y are itemsets.

Example: {Milk, Diaper} => {Beer}

Few Terminologies

- Maximal Frequent Itemset
  - An itemset is maximal if none of its immediate supersets is frequent
- Closed Itemset
  - An itemset is closed if none of its immediate supersets has same support as of the itmeset

Few Terminologies

- Lift
  - Lift is a measure of the performance of a targeting model (association rule) at predicting or classifying cases as having an enhanced response with respect to the population as a whole, measured against a random choice targeting model.

$$\text{Lift} = P(Y \mid X) / P(Y)$$

  -

Association Rules Mining

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ min_sup threshold and
  - confidence ≥ min_conf threshold
- Some of approaches for association rules mining are:
  - Brute-Force Approach
  - Frequent Itemset Generation Techniques

Brute- Force Approach

- List all possible association rules
- Compute the support and confidence for each rule
- Prune rules that fail to minimum support and minimum confidence level

- Pros
  - Easy Computation
  - Easy Implementation
  - Works perfect for smaller number of itemset
- Cons
  - Computationally expensive

Frequent Itemset Generation

- Formulate some ways to
  - Reduce the number of candidates
  - Reduce the number of transactions
  - Reduce the number of comparison
- Pros
  - Faster computation
  - Faster convergence toward solution
- Cons
  - Still slower (mostly depends on the min_support threshold)

Apriori Approach

- It is based on the Apriori Principle
  - Supersets of non-frequent item are also non-frequent
  - Or, If an itemset is frequent, then all of its subset also be frequent
- Two step Approach
  1) Frequent Itemset generation
  2) Rule Generation
- It use a level-wise search, k-itemsets are used to explore k+1 itemsets.
- At first, the set of frequent itemset is found and used to generate to frequent itemset at next level and so on

Apriori Algorithm

- Algorithm
  - Read the transaction database and get support for each itemset, compare the support with minimum support to generate frequent itemset at level 1.
  - Use join to generate a set of candidate k-itmesets at next level.
  - Generate frequent ietmsets at next level using minimum support.
  - Repeat step 2 and 3 until no frequent itme sets can be generated.
  - Generate rules form frequent itemsets from level 2 onwards using minimum confidence.

Example

- Separate PDF

Reference

- Reference Reading:
  - Book: Data Mining Concepts and Techniques – Morgan
    Chapter 6 Mining Frequent Patterns, Associations, and Correlations

Limitation of Apriori Algorithm

- Issues of Apriori Algorithm
  - Speed
  - High computational cost
  - Difficult to handle parallelism

Frequent Pattern Growth Algorithm

- Commonly Known as FP-Growth
- Improved version of Apriori Algorithm
- FP-growth algorithm is a tree-based algorithm for frequent itemset mining

- The algorithm represents the data in a tree structure known as FP-tree, responsible for maintaining the association information between the frequent items

FP-Growth

- The algorithm compresses frequent items into an FP-tree from the database while retaining association rules.

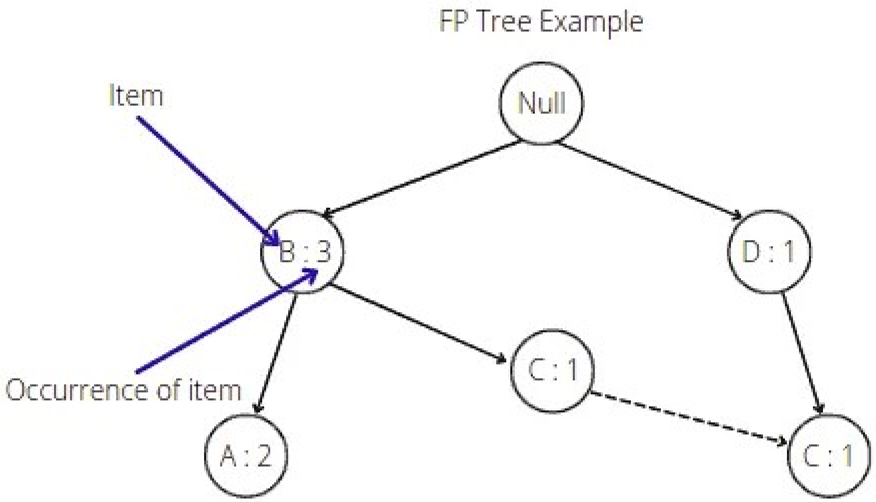
- Then it splits the database data into a set of conditional databases (a special kind of projected database), each of which is associated with one frequent data item.

FP-Tree

- FP-tree is the core concept of the FP-growth algorithm.
- The FP-tree is a compressed representation of the database itemset, storing the DB itemset in memory and keeping track of the association between items.
- The tree is constructed by taking each itemset and adding it as a subtree.
- The FP-tree's whole idea is that items that occur more frequently will be more likely to be shared.

FP-Tree

- The root node in the FP-tree is null.
- Each node of the subtree stores at least the item name and the support (or item occurrence) number.
- Additionally, the node may contain a link to the node with the same name from another subtree (represents another itemset from the database).



Building FP-Tree

- The FP-growth algorithm uses the following steps to build FP-tree from the database.
  - Scan itemsets from the database for the first time
  - Find frequent items (single item patterns) and order them into a list L in frequency descending order.

    For example, L = {A:5, C:3, D;2, B:1}
  - For each transaction order its frequent items according to the order in L
  - Scan the database the second time and construct FP-tree by putting each frequency ordered transaction onto it

FP-Tree (Example)

- Create a FP Tree of following dataset

ID	Items bought
100	{f, a, c, d, g, i, m, p}
200	{a, b, c, f, l, m, o}
300	{b, f, h, j, o}
400	{b, c, k, s, p}
500	{a, f, c, e, l, p, m, n}

FP-Tree (Example)

- Step 1: Item wise support count and eliminate the item that has support < min_support
  - Scan the dataset,
  - Create a frequency table containing each item from the database
  - Arrange them in descending order.
  - Filter items with a support value less than the minimum support
    - For example, let's set up the minimum support value equal to 3. In that case, we will get the following frequency table:

Item	Frequecy
{f}	4
{c}	3
{a}	3
{b}	3
{m}	3
{p}	3

FP-Tree (Example)

- Step 2: Rebuild dataset with items that created in step 1
  - Scan the database the second time and arrange elements based on the frequency table
  - Items with higher a frequency number will come first
    - if two items have the same frequency number they will be arranged in alphabetical order

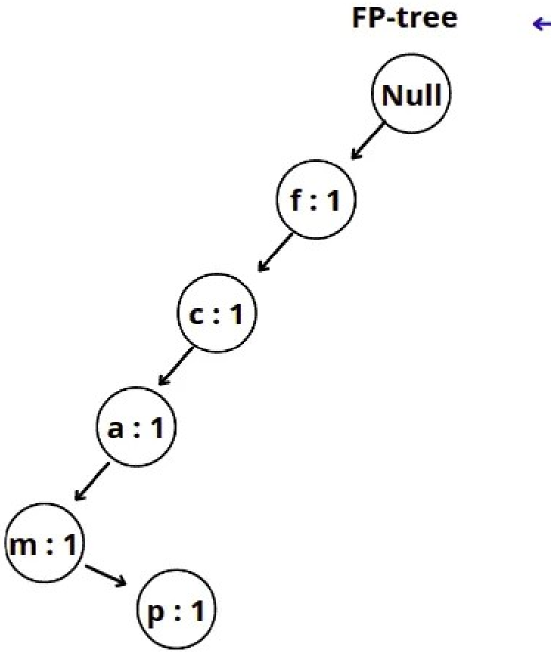
Item	Frequecy
{f}	4
{c}	3
{a}	3
{b}	3
{m}	3
{p}	3

ID	Items bought
100	{f, a, c, d, g, i, m, p}
200	{a, b, c, f, l, m, o}
300	{b, f, h, j, o}
400	{b, c, k, s, p}
500	{a, f, c, e, l, p, m, n}

Frequent Items (ordered)
{f, c, a, m, p}
{f, c, a, b, m}
{f, b}
{c, b, p}
{f, c, a, m, p}

FP-Tree (Example)

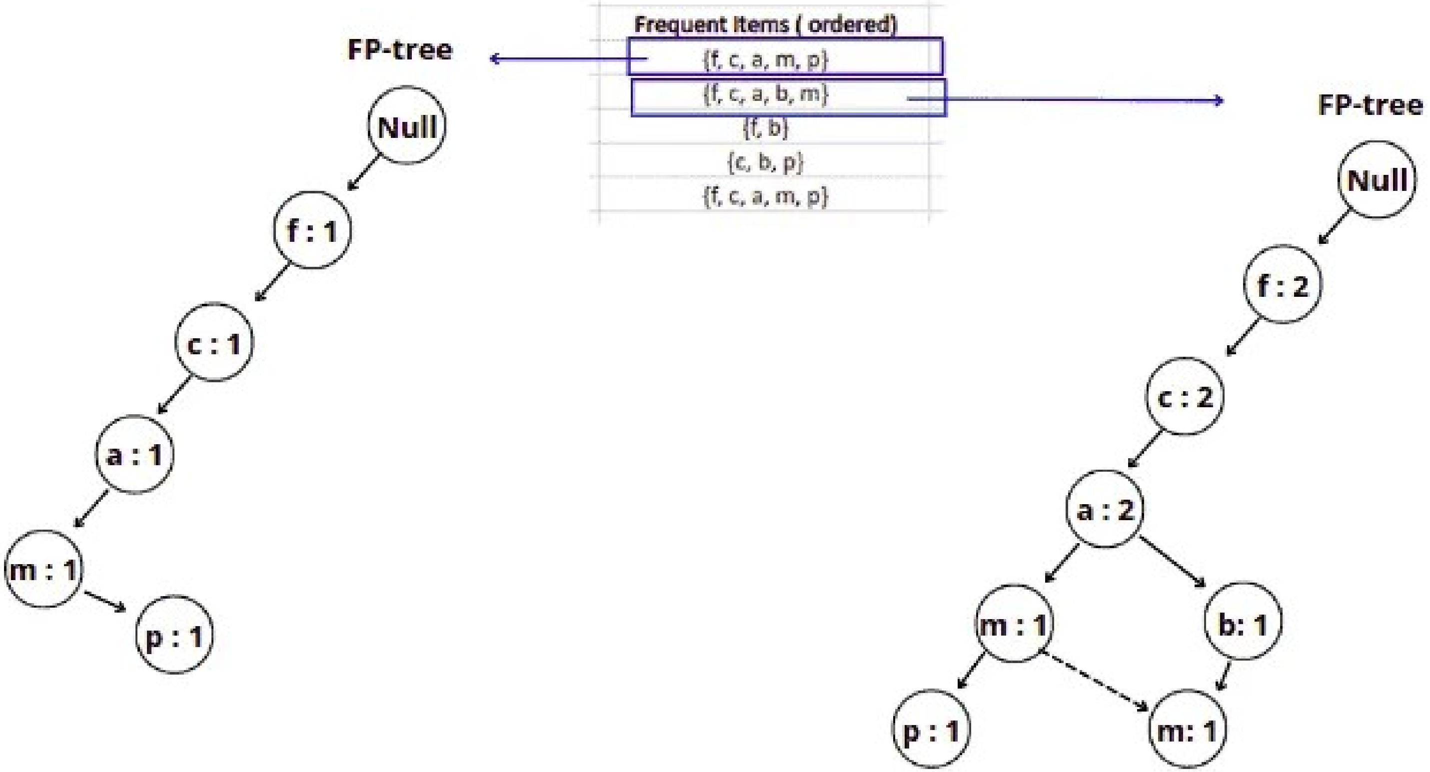
- Step 3: Build Tree
  - We will create a tree based on the frequent items table of step 2
  - Scan through each dataset and build the tree => Parent Node = NULL
  - Add the first item



Frequent Items (ordered)
{f, c, a, m, p}
{f, c, a, b, m}
{f, b}
{c, b, p}
{f, c, a, m, p}

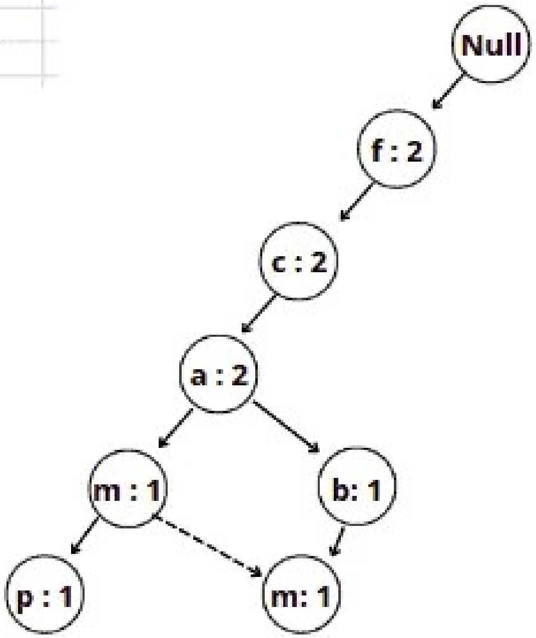
FP-Tree (Example)

- Add 2nd Item of itemset



FP-Tree (Example)

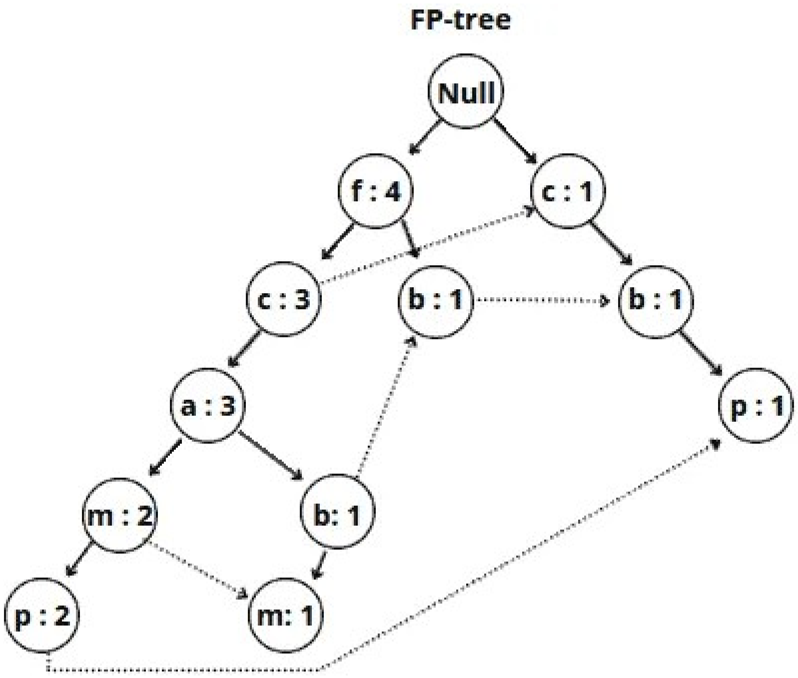
- As we add the same element to the tree, we increment the support.
- But after item a we created a new node for item b because there was no item b in our initial tree after item a.
- And we have linked items m together because this is the same element located in different subtrees.



FP-Tree (Example)

- Add the all the dataset and populate the tree

Frequent Items (ordered)
{f, c, a, m, p}
{f, c, a, b, m}
{f, b}
{c, b, p}
{f, c, a, m, p}

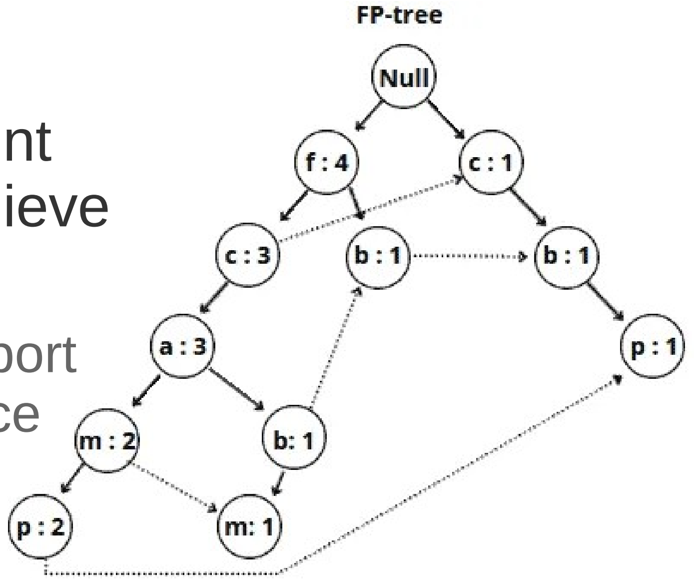


FP-Tree (Example)

- Step 4: Build Association Rules
  - It will take the item with the minor support count and trace that item through the FP-tree to achieve that goal.
    - In our example, the item p has the lowest support count, and the FP-growth algorithm will produce the following paths:

      { {f, c, a, m, p : 2}, {c, b, p : 1} }.

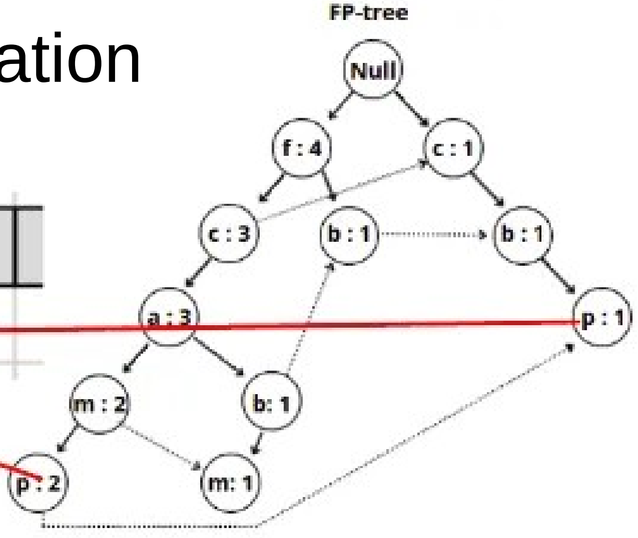
    - Note: The item p is located in two different subtrees of the FP-tree, so the algorithm traced both paths and added the minimum support value for every path.



FP-Tree (Example)

- Conditional Pattern Base Generation

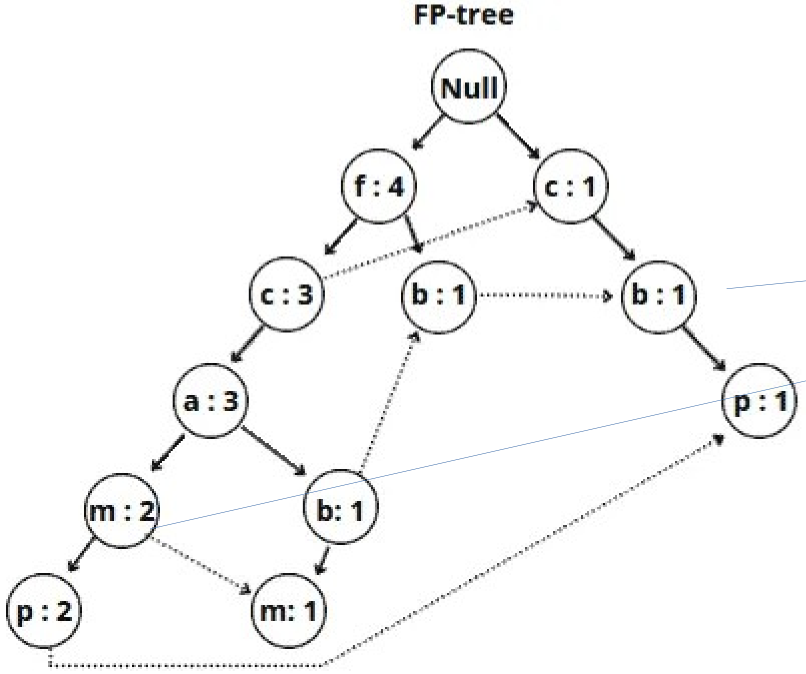
Item	Conditional Pattern Base
p	{{ f, c, a, m, :2}, { c, b :1}}



  – Similarly, the FP-growth will build the conditional pattern base table for all of the items from the FP-tree.

FP-Tree (Example)

- Conditional Pattern Base Generation
  - Start with leaf node and traverse upward (except those attached to NULL node)



Item	Conditional Pattern Base
p	{{ f, c, a, m : 2}, { c, b : 1}}
m	{{ f, c, a : 2}, {f, c, a, b : 1}}
b	{{f :1}, { c:1}, {f, c, a : 1}}
a	{{f, c : 3}}
c	{{ f : 3}}

FP-Tree (Example)

- Conditional FP Tree Generation

  Minimum support =3

  - get all items from the Conditional Pattern Base column that satisfy the minimum support requirement.
  - Let's calculate elements' occurrences for the p item:

    { f, c, a, m : 2 }, { c, b : 1 } - > { f: 2, c:3, a:2, m:2, b:1 }

  - Only item c appears three times and satisfies the minimum support requirement.
  - That means the algorithm will remove all other items except c.

FP-Tree (Example)

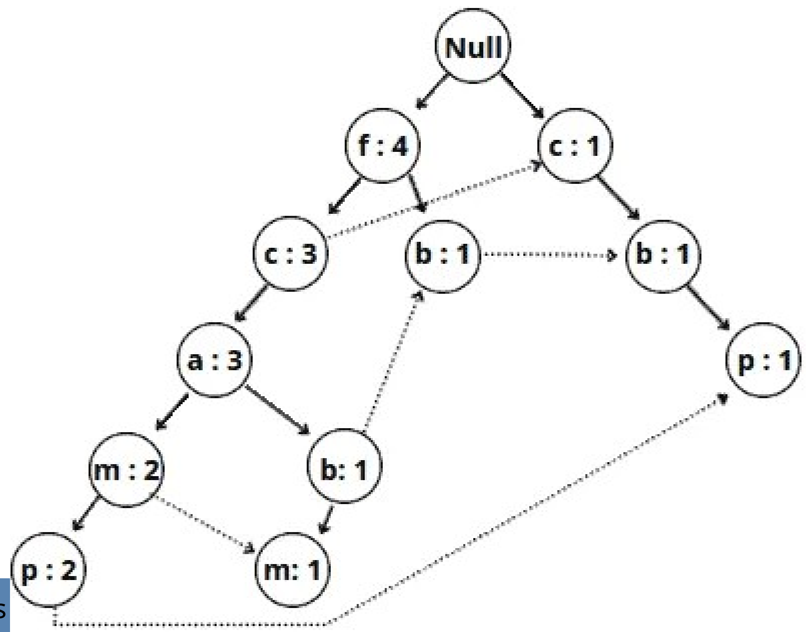
- Conditional FP Tree Generation

  Minimum support =3

  – After removing items that do not meet the minimum support requirement, the algorithm will construct the following table:

Item	Conditional Pattern Base	Conditional FP-tree
p	{{f, c, a, m : 2 }, { c, b : 1 }}	{ c:3 }
m	{{ f, c, a : 2}, { f, c, a, b : 1 }}	{ f:3, c:3, a:3 }
b	{{ f : 1 }, { c : 1}, { f, c, a : 1 }}	--
a	{{ f, c : 3 }}	{ f:3, c:3 }
c	{{ f : 3 }}	{ f:3 }

FP-tree



FP-Tree (Example)

- Generate frequent patterns
  - Generate frequent patterns by pairing the items of the Conditional FP-tree column with the corresponding item from the Item column.
    - For example, for the first row
      - { c:3 } from the Conditional FP-tree column,
      - create its combination with the p element and add the support count value

Item	Conditional Pattern Base	Conditional FP-tree	Generated Frequent Patterns
p	{{f, c, a, m : 2 }, { c, b : 1 } }	{ c:3 }	{ c, p : 3 }
m	{{ f, c, a : 2}, { f, c, a, b : 1 } }	{ f:3, c:3, a:3 }	{ f, m : 3 }, { c, p : 3 }, { a, m : 3 }, { f, c, m : 3}, { f, a, m : 3}, { c, a, m : 3 }, { f, c, a, m : 3 }
b	{{ f : 1 }, { c : 1}, { f, c, a : 1 } }	--	--
a	{{ f, c : 3 } }	{ f:3, c:3 }	{ f, a : 3 }, { c, a : 3 }, { f, c, a : 3 }
c	{{ f : 3 } }	{ f:3 }	{ f, c : 3 }

FP-Tree (Example)

- Generate Association Rules

Confidence : 70%

- Calculate the support and confidence for items in generated frequent pattern as done in Apriori Algorithm

Do it Yourself

- Explore of FP-growth algorithm using Python
  - mlextend library

```
from mlxtend.frequent_patterns import fpgrowth
from mlxtend.frequent_patterns import association_rules

res    = fpgrowth(dataset,min_support=0.05, use_colnames=True)
rules = association_rules(res, metric="lift", min_threshold=1)
```

- Reference: https://hands-on.cloud/implementation-of-fp-growth-algorithm-using-python/

Thank you