Nithin Kumar
Asst professor
Dept of CS&E
VVCE, Mysuru

Module - 3

Association Analysis

* Association Analysis :-

Association discovers the Association or Connection Among a set of items in large data sets. Association Identifies the relationship among objects.

→ Association is used for Commodity Management, Advertising, Direct Marketing Etc.

→ For Ex, A retailer can Identify the products that Normally Customers purchase together or Even find the Customers who respond to the promotion of same kind of products.

→ Consider an Example of Market basket Transactions.

TID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

Each row in this table corresponds to a transaction, which contains a unique Identifier labeled "TID" & a set of items bought by a given Customer.

→ The Association Analysis is useful for discovering Interesting relationships hidden in large data sets.
The uncovered relationships can be represented in the form of "Association rules" or "Sets of frequent items".

The Basic Terminology used in Association Analysis are

* Binary representation :- The Transaction data can be represented in a binary format

TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

where Each row corresponds to a transaction & Each column corresponds to an item.

→ An item can be treated as a binary variable whose value is 'one' if the item is present in a transaction & 'zero' otherwise.

* Itemset :- is a collection of one or more items.

Ex:- {Milk, Bread, Diapers} = 3-Itemset

And K-Itemset is a Itemset which contains "K-items".

* Support Count (σ) :- frequency of occurance of an Itemset.

→ In the above table, {Milk, Bread, Diapers} items Appears for two times.

$\therefore \sigma$ {Milk, Bread, Diapers} = 2

* Support :- Fraction of Transaction that contains Itemset.

→ The Support (s) is given by = $\frac{\text{Support Count } (\sigma)}{\text{total no of Transactions}}$

$\therefore s$ {Milk, Bread, Diapers} = $\frac{\sigma \{\text{Milk, Bread, Diapers}\}}{\text{total no of Transaction}}$ = 2/5 //

* Association Rule :- An Association rule is an implication Expression form $X \rightarrow Y$, where X & Y are disjoint itemsets.

→ The Strength of an Association rule can be Measured in terms of its
  * Support
  * Confidence.

→ Consider the foll Example

$\{Milk, diapers\} \rightarrow \{Beers\}$

* Support → Fraction of transactions that Contain both X & Y

$$S = \frac{\sigma\{Milk, diapers, Beers\}}{|T|} = \frac{2}{5} = 0.4 //$$

* Confidence → Measure How often items in 'Y' Appear in Transaction that Contain X

$$C = \frac{\sigma\{Milk, diapers, Beers\}}{\sigma\{Milk, diapers\}} = \frac{2}{3} = 0.67 //$$

→ Support is an important Measure, Because a rule that has Very low Support May occur Simply by Chance.
Confidence on the other hand, Measures the Reliability of Interference Made by rule.

* Formulation of Association rule Mining problem :- Given a Set of transactions T, the goal of Association rule Mining is to find all rules having
  * Support > Minsup Threshold
  * Confidence > Minconf Threshold

"Minsup" & "Minconf" are the Corresponding Support & Confidence Thresholds.

→ A Common Strategy Adopted by Many Association rule Algorithms is to decompose the problems into two Major Subtasks.

* Frequent Itemset Generation → is a generation of all itemsets whose Support > Minsup
* Rule Generation → is a generation of High Confid-ence rules from Each frequent itemset.

* Frequent itemset Generation (Apriori Algorithm)

Apriori Algorithm is the first Association Rule Mining Algorithm that pioneered the use of Support-Based pruning to Systematically Control the Exponential Growth of Candidate Itemsets.

→ Apriori uses a "Bottom-up" Approach, where frequent Subsets are Extended one item at a time.

→ Apriori is designed to operate on Database Containing Transactions

For Ex:- Collections of Items bought by Customers, or details of a Website Frequentation.

* Steps to perform Apriori Algorithm

* Step 1:- Scan the Transaction Database to get the Support of Search 1-Itemset, Compare with "Min_Sup"

* Step 2:- Use Apriori property to prune the unfrequented K-Itemsets from the Set

* Note:- Apriori property → Any Subset of frequent Itemset Must be Frequent

* Step 3 :- Count the Support of Each Candidate by Scanning the DB
* Step 4 :- Eliminate Candidates that are Infrequent, leaving only those that are frequent

Finally, Repeat the whole procedure until no new frequent Itemsets are Identified.

→ Consider the following Example of Frequent Itemset Generation using Apriori Algorithm

TID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Egg}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

Min Sup Count = 3

→ Firstly, Generate 1-Itemset by Counting the Components

Bread	4
Diapers	4
Milk	4
Beer	3
Egg	1
Cola	2

(1-Itemset)

Eliminate the candidates that are Infrequent by Comparing with "Min Sup Count = 3" Such as "Egg & Cola"

→ Next, Generate 2-Itemset by Joining Subset of two Components from 1-Itemset

{Bread, Milk}	3
{Bread, Diapers}	3
{Bread, Beers}	2
{Milk, Diapers}	3
{Milk, Beers}	2
{Diapers, Beers}	3

(2-Itemset)

Get the count of Subsets by Scanning Base-DB, Eliminate the Candidates that have less count compare to "Min Sup Count"

→ Generate 3-Itemset by forming Subset of three Components from 2-Itemset.

{Bread, Milk, Diapers}	2
{Bread, Diapers, Beers}	2
{Milk, Diapers, Beers}	2
{Bread, Milk, Beers}	1

(3-Itemset)

All 3-Itemset are Infrequent, Because Count is less than Min Sup Count = 3.

* Apriori Algorithm

1. $k = 1$
2. Find all frequent 1-Itemsets $F_k = \{i \mid i \in I \wedge \sigma(\{i\}) \geq \text{Min Sup}\}$
3. Repeat
4. $k = k+1$
5. $F_k = \text{Apriori-gen}(F_{k-1})$ // Generate Candidate Itemsets
6. For Each Transaction $t \in T$ do
7. Generate Subset (C_k, t)
8. End for
9. $F_k = \{c \mid c \in C_k \geq \text{Min Sup}\}$ // Extract k-frequent Itemsets
10. Until $F_k = 0$

11. Return F_k

* Limitations of Apriori Algorithm → Can be very slow & the Bottleneck is Candidate Generation

To Compute those with Support More than MinSup, the database need to be Scanned at Every level. It needs (n+1) Scans. Apriori Algo consumes "Huge Memory".

* Solved problem on Frequent item Generation (Apriori)

Consider the foll Transaction

TID	Items
1	{1} {3} {4}
2	{2} {3} {5}
3	{1} {2} {3} {5}
4	{2} {5}

MinSup Count = 2

Itemset	Support
{1}	2
{2}	3
{3}	3
~~{4}~~	~~1~~ < 2 (MinSup)
{5}	3

1-Itemset

Itemset	Support
~~{1,2}~~	~~1~~ < 2 (MinSup)
{1,3}	2
~~{1,5}~~	~~1~~ < 2 (MinSup)
{2,3}	2
{2,5}	3
{3,5}	2

2-Itemset

Itemset	Support
~~{1,3,2}~~	~~1~~
~~{1,2,5}~~	~~1~~
~~{1,3,5}~~	~~1~~
{2,3,5}	2

3-Itemset

∴ Frequent Itemset = {2,3,5} //

* Computational Complexity of Apriori Algorithm :-

The Computational Complexity of the Apriori Algorithm can be Affected by the following factors.

* Support Threshold :- Lowering the Support Threshold (Min Sup) often results in More itemsets being declared as frequent. This has an adverse Affect on Computational Complexity of the Algorithm.

* Number of Items (Dimensionality) :- As the no of items increases, More Space will be needed to Store the Support Counts of items.
The Computation & I/O costs will increase because of the larger no of itemsets.

* Average Transaction Width :- The Average Transaction Width can be very large. This Affects the Complexity of Apriori Algorithm in two ways
  * Increase in Max Size of Frequent itemsets
  * Increase in No of Hash Tree Traversals.

* Generation of Frequent 1-itemsets :- For Each Transaction, we need to update the Support Count for Every Item present in the Transaction.

* Candidate Generation :- To Generate Candidate k-itemsets, pairs of frequent (k-1) itemsets are Merged. Each Merging operation requires at Most k-2 Equality Comparisons.

∴ overall cost of Merging =

$$\sum_{k=2}^{w} (k-2)\,|C_k| < \text{Cost of Merging} < \sum_{k=2}^{w} (k-2)\,|F_{k-1}|^2$$

* Support Counting :- Each Transaction of length |t| produces $\binom{|t|}{k}$ itemsets of size k. This is also the effective no of Hash tree Traversals performed for Each Transaction.
  * Cost for Support Counting
  * Cost for updating the Support.

* Support Counting :-
  is the process of determining the frequency of occurrance for Every Candidate itemset that survives the Candidate pruning of the Apriori-gen function.

→ There are two Approaches to perform Support Counting
  * One Approach is to compare Each transaction Against Every Candidate itemset & update the Support Counts of Candidates Contained in the Transaction.
    This Approach is Computationally Expensive.
  * Alternative Approach is to Enumerate the itemsets Contained in Each transaction & use them to update the Support Counts of their respective Candidate itemsets.

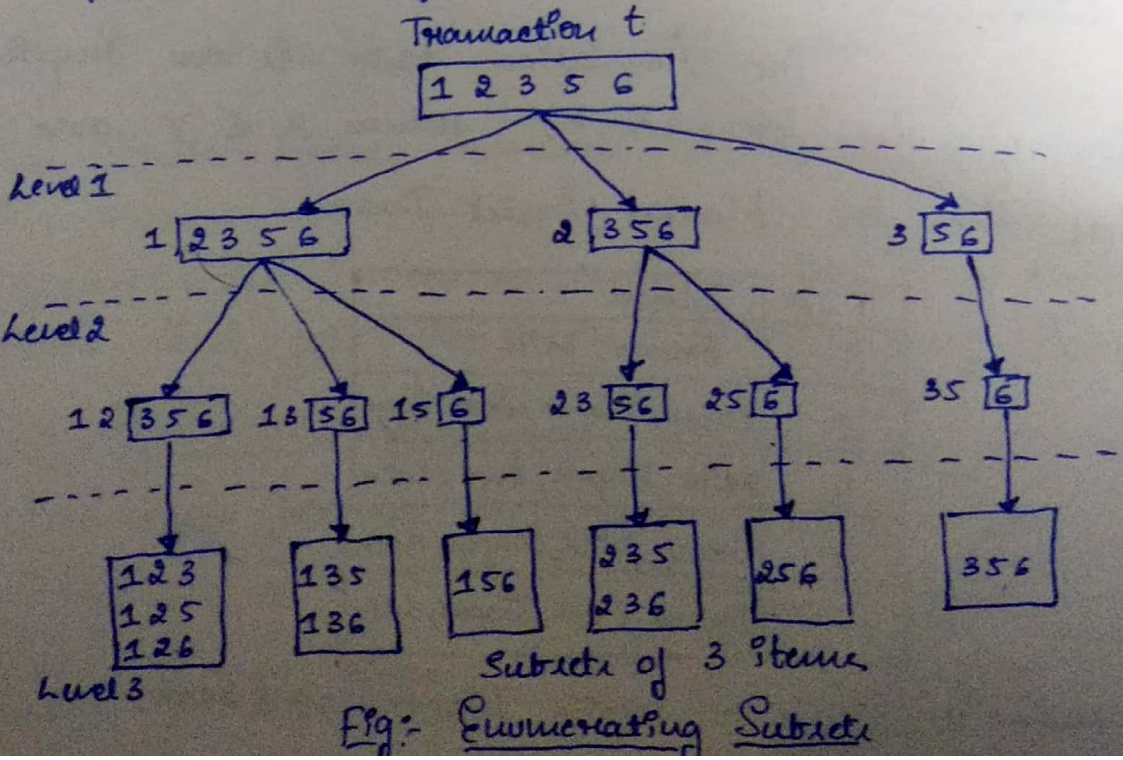


Fig:- Enumerating Subsets

→ Assuming that each itemset keeps its items in increasing lexicographic order, An itemset can be enumerated by specifying the smallest item first, followed by the larger items.

→ Consider the above fig example, where given $t = \{1, 2, 3, 5, 6\}$. all the 3-itemsets contained in t must begin with item 1, 2 or 3.
It is not possible to construct a 3-itemset that begins with 5 or 6.

* In level 1, for instance, 1 [2 3 5 6] represents a 3-itemset that begins with item 1, followed by two more items chosen from the set $\{2, 3, 5, 6\}$
* In level 2, for instance, 1 2 [3 5 6] corresponds to itemsets that begin with prefix (1 2) & are followed by items 3, 5 or 6.
* Finally, the prefix structures at level 3 represent the complete set of 3-itemsets contained in t.

* Rule Generation :-

The Association rule is an implication expression of the form $X \rightarrow Y$, where X & Y are itemsets

→ Consider the following Market Basket Data

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Ex:- Association rule = $\{Milk, Diaper\} \rightarrow \{Beer\}$

→ There are Two rule Evaluation Metrics

* Support (s) → Fraction of Transaction that contain both X & Y

* Confidence (c) → Measures how often items in Y Appear in Transactions that contain X

→ For the above Mentioned Example Association rule, How to Calculate Support & Confidence

$\underbrace{\{Milk, Diaper\}}_{X} \longrightarrow \underbrace{\{Beer\}}_{Y}$

* Support (s) = $\frac{\sigma(X+Y)}{\text{Total no of Transaction}} = \frac{\sigma\{Milk, Diaper, Beer\}}{|T|}$

$= \frac{2}{5} = 0.4$ //

* Confidence (c) = $\frac{\sigma(X+Y)}{\sigma(X)} = \frac{\sigma\{Milk, Diaper, Beer\}}{\sigma\{Milk, Diaper\}} = \frac{2}{3} = 0.67$ //

→ For Given Set of Transactions T, the goal of Association rule Mining is to find all rules Having

* Support ≥ Min Sup Threshold

* Confidence ≥ Min conf Threshold

→ The Brute-Force Approach is to List all possible Associat-ion rules.
Compute the Support & Confidence for Each rule, Remove rules that fail the "Min Sup" & "Min Conf" Thresholds

* Rule-Generation-Algorithm (Apriori-Rule-Generation)

1. For Each frequent k-Itemset f_k, $k \geq 2$ do
2. $m = |H_m|$ // Size of rule Consequent
3. If $k > m+1$ then

4. H_{m+1} = Apriori_gen (H_m)
5. for Each H_{m+1} do
6. If Conf ≥ minConf then
7. Add the rule to Rule Set (H_{m+1})
8. Else
9. Delete the rule from H_{m+1}
10. End If
11. End for
12. End for

* Solved problem on Rule Generation (Apriori_Rule_Generation)

Consider the foll Transaction details

TID	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

Min Sup = 2

Min Conf = 70%

→ Firstly, we should Apply Apriori Frequent Itemset Generation Algo to Generate frequent Itemset.

Itemset	S
A	2
B	3
C	3
D	1
E	3

< MinSup

1-Itemset

Itemset	Support
A,B	1
A,C	2
A,E	1
B,C	2
B,E	3
C,E	2

< MinSup

< MinSup

2-Itemset

3-Itemset

Itemset	Support
A,B,C	1
A,B,E	1
A,C,E	1
B,C,E	2

< MinSup

< MinSup

< MinSup

∴ Frequent Itemset = {B, C, E}

→ Next, Apply Rule Generation procedure on obtained Frequent Itemset {B, C, E} list all possible combinations.

* B, C → E
* C E → B
* B E → C

Just Interchange Rule Consequent & Antecedent

* E → B, C
* B → C, E
* C → B, E

Totally Six rules, now we should check the Confidence Value for Every Individual rule.

* B, C → E = $\frac{\sigma(B, C, E)}{\sigma(B, C)} = \frac{2}{2}$ = 100% > MinConf ∴ Accepted //

* C, E → B = $\frac{\sigma(B, C, E)}{\sigma(C, E)} = \frac{2}{2}$ = 100% > MinConf ∴ Accepted //

* B, E → C = $\frac{\sigma(B, C, E)}{\sigma(B, E)} = \frac{2}{3}$ = 66.6% < MinConf ∴ Rejected

* E → B, C = $\frac{\sigma(B, C, E)}{\sigma(E)} = \frac{2}{3}$ = 66.6% < MinConf ∴ Rejected

* B → C, E = $\frac{\sigma(B, C, E)}{\sigma(B)} = \frac{2}{3}$ = 66.6% < MinConf ∴ Rejected

* C → B, E = $\frac{\sigma(B, C, E)}{\sigma(C)} = \frac{2}{3}$ = 66.6% < MinConf ∴ Rejected

→ Out of Six all possible Combination rules, only two rules Satisfied the MinConf Threshold

∴ Generated Rules are **B, C → E** & **C, E → B** //

* FP-Growth Algorithm :-

FP-Growth take a radically diff Approach to discovering Frequent Itemsets. The Algorithm does not use the "Generate-and-Test" paradigm of Apriori. Instead

It Encodes the data set using a Compact Data Structure called an "FP-Tree" & Extracts Frequent Itemsets directly from FP-Tree.

→ FP-Tree is an Compressed Representation of Input data. Once an FP-Tree has been Constructed, it uses Recursive "Divide & Conquer" Approach to Mine Frequent Itemsets.

→ FP-Growth is an improvements of Apriori designed to Eliminate Some of the Heavy Bottlenecks in Apriori.
In FP-Tree Each node represents an Item & it's Current Count & Each branch represent a different Association.

→ The FP-Growth Algorithm is divided into 5 Simple Steps as Shown below:-

* Step 1 :- The First Step is to Count all the Items in whole Transactions to get Support Count.

* Step 2 :- Next we Apply the Threshold (MinSup) we had Set previously. & Remove Item which posses Support Count less than "MinSup"

* Step 3 :- Now, we Sort the list of Items with Support Count greater than "MinSup" in Decending order.

* Step 4 :- Now, we Build the FP-Tree, we go through Each of the Transactions & add all the Items in the order they Appear in our Sorted List.

* Step 5 :- In order to get the Associations now we go through Every Branch of the Tree & only Include in the Association all the nodes whose Count passed the threshold (MinSup).

→ Consider the following Example.

TID	Items
1	{a, b}
2	{b, c, d}
3	{a, c, d, e}
4	{a, d, e}
5	{a, b, c}
6	{a, b, c, d}
7	{a}
8	{a, b, c}
9	{a, b, d}
10	{b, c, e}

Min Support = 2

Apply FP-Growth Algorithm & Construct FP-Tree ?

⇒ *Step 1 :- Count All Items in whole Transactions to get Support count

Items	Support
a	8
b	7
c	6
d	5
e	3

*Step 2 :- Apply the Threshold (MinSup) & remove Item which poses Support Count less than MinSup

In the above table, all Items Support Count is greater than "MinSup = 2", so all Items are considered in FP-Tree.

* Step 3 :- Arrange Items in Descending order of their Support Count

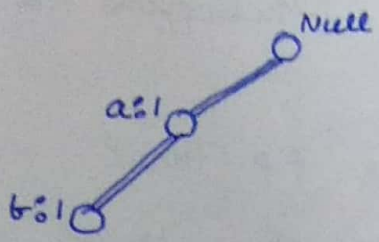
Items	Support
a	8
b	7
c	6
d	5
e	3

∴ "a, b, c, d, e" will be the FP-Tree Construction pattern. (As per Descending order)

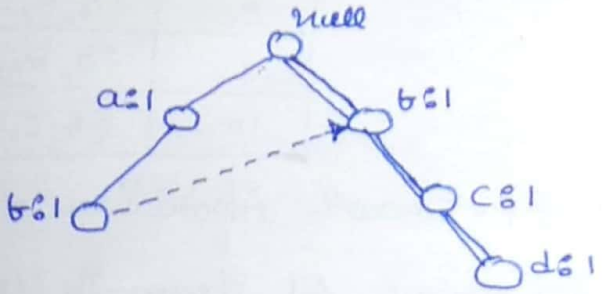
* Step 4 :- FP-Tree Construction, we go through Each of Transaction with Joll two condition.

* Items where Support Count is less than Min-Sup should be removed from Transaction
* Descending-order Item list (pattern) for FP-Tree Construction Should be Maintained.

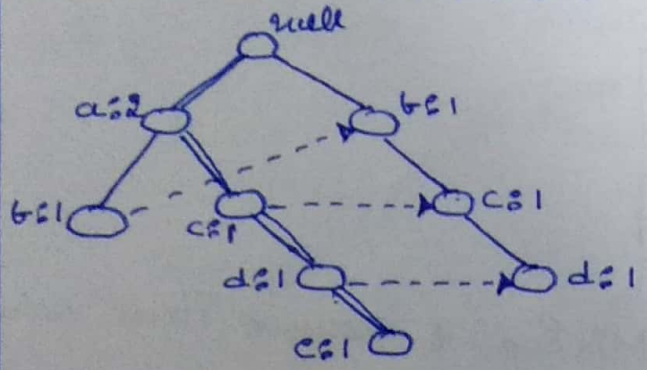
(i) Transaction - 1 {a, b}



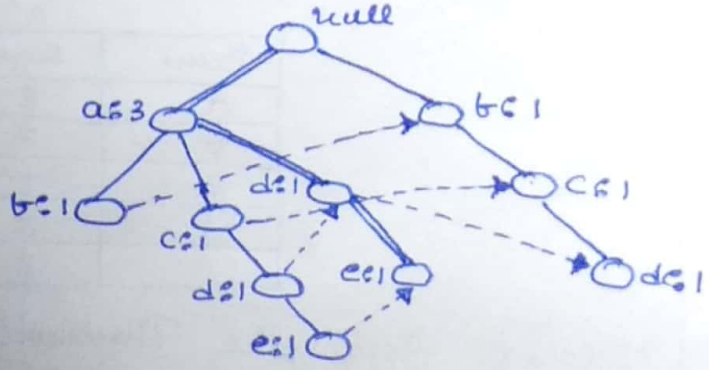
(ii) Transaction - 2 {b, c, d}



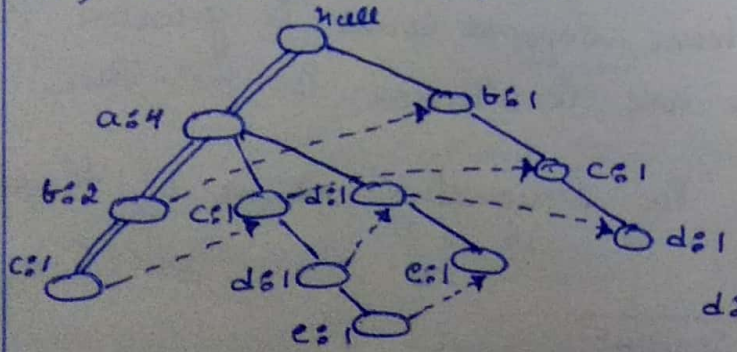
(iii) Transaction - 3 {a, c, d, e}



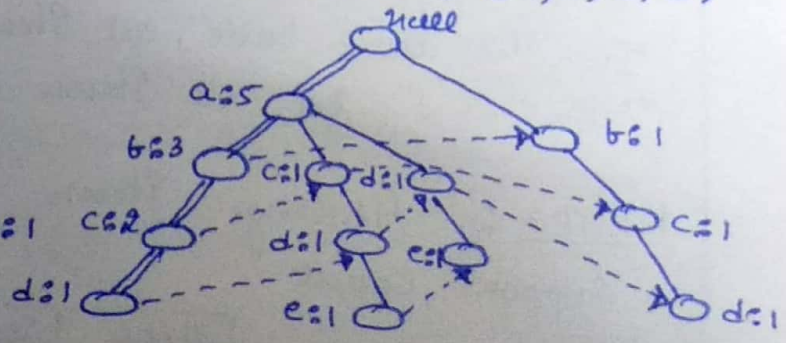
(iv) Transaction - 4 {a, d, e}



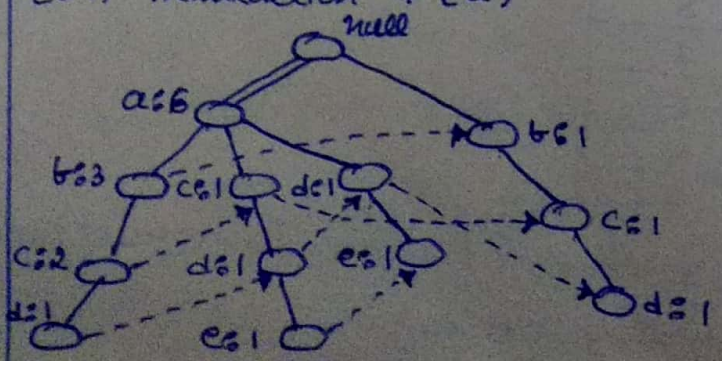
(v) Transaction - 5 {a, b, c}



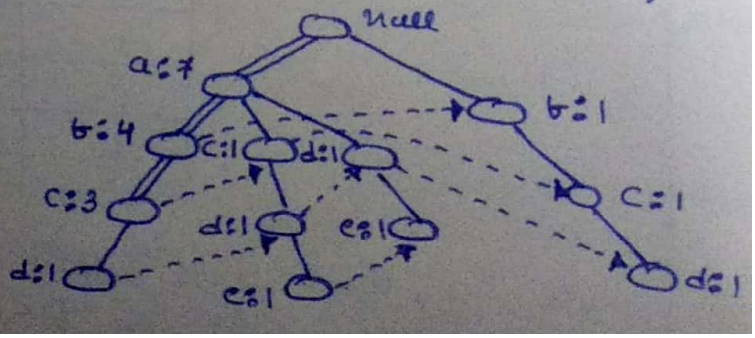
(vi) Transaction - 6 {a, b, c, d}



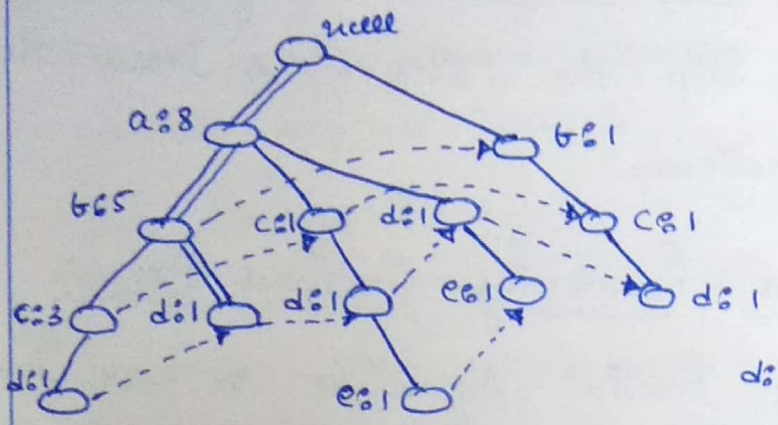
(vii) Transaction - 7 {a}



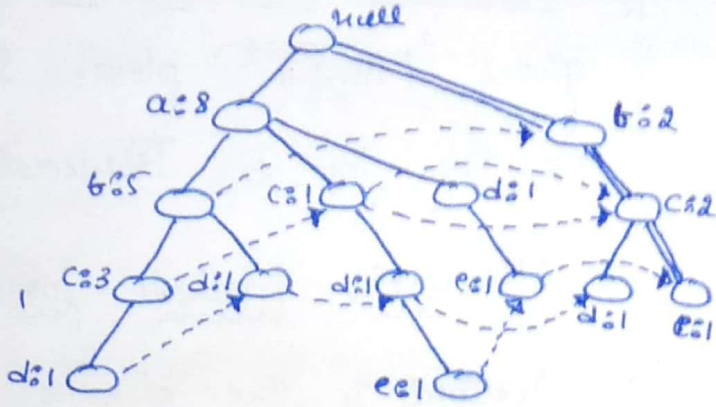
(viii) Transaction - 8 {a, b, c}



(IX) Transaction-9 {a, b, d}

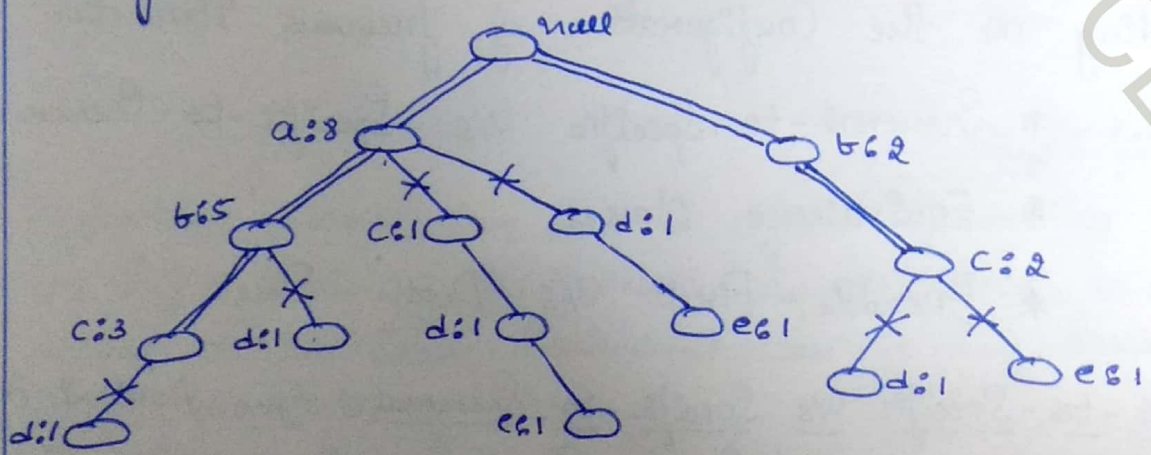


(X) Transaction-10 {b, c, e}



(For Students Convenience, I've shown FP-Tree Construction Transaction-by-Transaction, but while solving FP-Tree problem it's better show Max Transactions construction so that you'll get Full Marks). Note

* Step 5 :- By using constructed FP-Tree compare Min Support to get Associations. MinSup = 2



∴ Associations are {a, b, c} & {b, c} //

* Advantages of FP-Growth Algorithm :-

* The Biggest Advantage found in FP-Growth is the fact that Algorithm only need to read the file twice, as opposed to Apriori who reads it once for Every Iteration.
* It removes the need to calculate the pairs to be counted, which is very processing Heavy.
* FP-Algo stores in Memory a Compact Version of DB.

* Important Note on FP-Tree Construction :- If you're not given "MinSup" please Skip the "Step-2" & Draw the FP-Tree for all Transactions.

* Alternative Methods for Generating Frequent itemsets

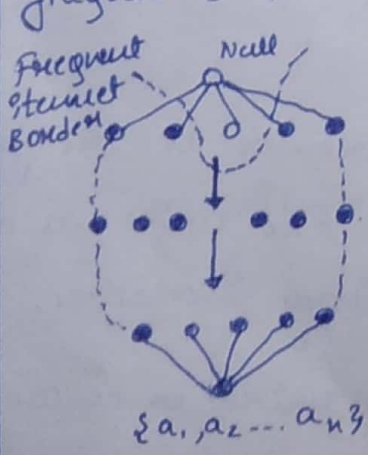
Apriori is one of the Earliest Algorithm to have Successfully addressed the Combinational Explosion of frequent Itemset Generation.

Despite its Significant performance improvement, the Algorithm Still incurs Considerable I/O overhead Since it requires Several passes over the Transaction Data-Set.

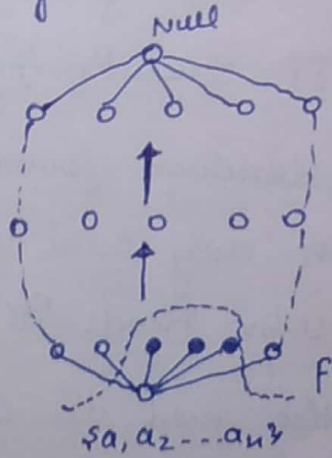
→ Some of the Search Strategies are better than Apriori depending on the Configuration of frequent itemsets.

* General-to-Specific V/S Specific-to-General
* Equivalence Classes
* Breadth-First V/S Depth-First

* General-to-Specific V/S Specific-to-General :- The Apriori Algo uses a "general-to-specific" Search Strategy, where pairs of frequent (k-1) Itemsets are Merged to obtain k-Itemsets.



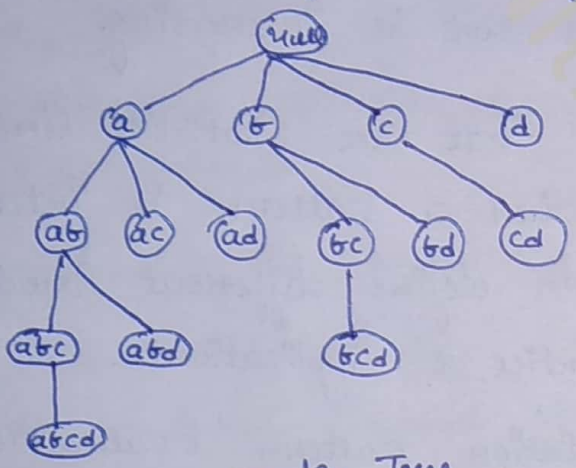
General-to-Specific



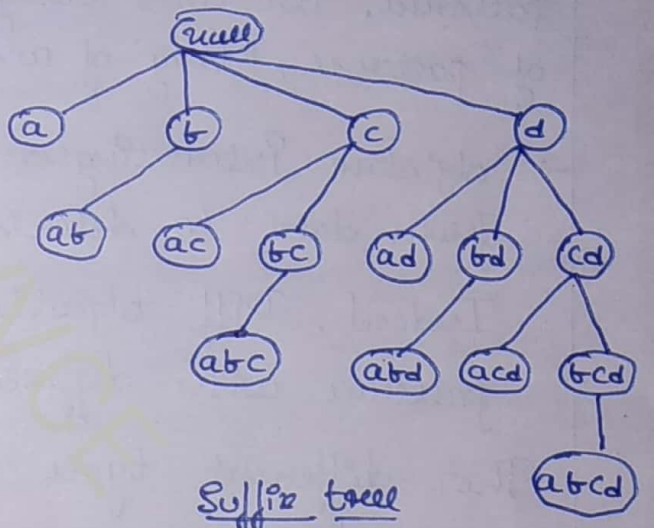
Specific-to-General

→ Alternatively, a "Specific-to-general" Search Strategy looks for More Specific frequent Itemsets first, before finding the More general frequent itemsets.
This Strategy is useful to discover Maximal frequent Itemsets in Dense Transactions.

* Equivalence Classes :- A frequent Itemset generation Algorithm Searches for frequent Itemsets within a particular Equivalence Class before Moving to another Equivalence Class



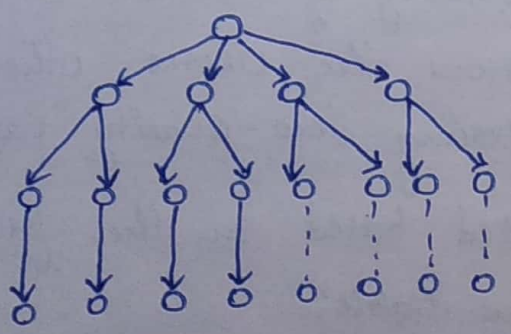
Prefix Tree



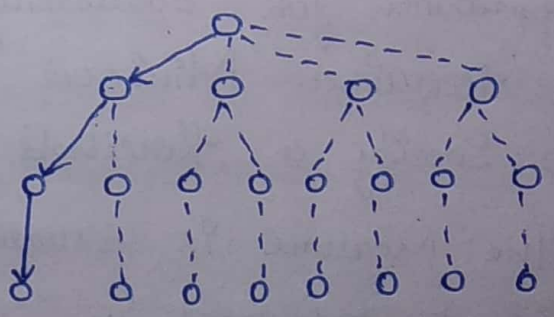
Suffix tree

Equivalence Classes can also be defined according to the prefix or Suffix labels of an Itemset. In this case, two itemsets belong to the same Equivalence Class if they share a common prefix or Suffix of length k.

* Breadth-First V/s Depth-First :- The Apriori Algo traverses in a "Breadth-First" Manner. It first Discovers all the frequent 1-Itemsets followed by 2-itemsets & so on.



Breadth First



Depth First

→ The "Depth-First" Approach is often used by Algorithms designed to find Maximal Frequent Itemsets.
This Approach allows the frequent Itemset Border to be detected More Quickly than Breadth-First Approach.

* Evaluation of Association patterns :-
The Association Analysis Algorithms have the potential to generate a large no of patterns, we could Easily End up with thousands or Even Millions of patterns, Many of which Might not be Interesting.

→ "objective Interestingness Measure" that uses Statistics derived from data to determine whether a pattern is Interesting.
Indeed, Diff objective Measures define different Association patterns with different properties & Applications.

The different types of Association pattern Evaluation Criteria's are

* Support-Confidence Framework
* Interest Factor
* Correlation Analysis
* IS Measure

* Support-Confidence Framework :- is a data-driven Approach for Evaluating the quality of Association patterns It requires Minimal Input from the Users, other than to Specify a threshold for filtering low-quality patterns.

→ The Measure is Usually computed based on the frequency counts tabulated in "Contingency table"
The Contingency table is as Shown below

	B	$\bar{B}$	
A	f_{11}	f_{10}	f_{1+}
$\bar{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

2- way Contingency table

* Limitations of Support - Confidence Framework → Existing Association rule Mining formulation relies on the Support & Confidence Measure.
The Drawback of Support was Elimination of potentially Interesting patterns & The Drawback of Confidence is More Subtle.

* Interest Factor :- The High - Confidence Rules can Some - times be Misleading Because the Confidence Measure Ignores the Support of Itemset Appearing in the rule - Consequent.
One way to address this problem is by Applying Metric Known as "LIFT"

$$LIFT = \frac{C(A \rightarrow B)}{S(B)}$$

which computes the ratio between rule Confidence & the Support of the Itemset in rule Consequent.

* Limitations of interest factor → An Example from the text Mining Domain, It is reasonable to Assume that the Association between a pair of words depends on the no of Documents that Contain both words.
We Expect the words "Data" & "Mining" to appear More frequently than words "Compiler" & "Mining" in Computer Articles

* Correlation Analysis :- Correlation Analysis is "Statistical Based Technique" for Analyzing Relationship between a pair of Variables.

→ For Continous Variables, Correlation is defined using "pearson's Correlation Co-efficient", for the Binary Variable is given by

$$\phi = \frac{F_{11} F_{00} - F_{01} F_{10}}{\sqrt{F_{1+} F_{+1} F_{0+} F_{+0}}}$$

* Limitation of Correlation Analysis → The Drawback of using Correlation can be seen from the word Association.
Because the ϕ-co-efficient gives Equal Importance to both "co-presence" & "co-Absence" of items in a Transaction.

* IS-Measure :- IS is an alternative Measure that has been proposed for Handling Asymmetric Binary Variables.

→ The IS Measure is defined as follows

$$* \; IS(A,B) = \sqrt{I(A,B) \times S(A,B)} = \frac{S(A,B)}{\sqrt{S(A)\, S(B)}} //$$

→ The IS Measure is large when the Interest factor & Support of the patterns are large.

* Limitations of IS-Measure → Shares a Similar problem as Confidence Measure that the Value of the Measure can be quite large Even for uncorrelated & Negatively correlated patterns.