DMDA IMPORATANT QUESTIONS:

1. **A database has five transactions. Let min-support = 60_ and min- confidence = 80%. Find all frequent item sets by using Apriori Algorithm T_ID is the transaction ID.**

T_ID	**Items bought**
T-1000	M,O,N,K,E,Y
T-1001	D,O,N,K,E,Y
T-1002	M,A,K,E
T-1003	M,U,C,K,Y
T-1004	C,O,O,K,E

Apriori algorithm:

The apriori algorithm solves the frequent item sets problem. The algorithm analyzes a data set to determine which combinations of items occur together frequently. The Apriori algorithm is at the core of various algorithms for data mining problems. The best known problem is finding the association rules that hold in a basket -item relation.

Numerical:

Given:

Support = 60% = 60100x560100x5 = 3
Confidence = 70%

ITERATION 1:

STEP 1: (C1)

Item	**Count**
A	1
C	2
D	1

Item	Count
E	4
I	1
K	5
M	3
N	2
0	3
U	1
Y	3

STEP 2: (L2)

Item	Count
E	4
K	5
M	3
O	3
Y	3

ITERATION 2:

STEP 3: (C2)

Item	**Count**
E,K	4
E,M	2
E,O	3
E,Y	2
K,M	3
K,O	3
K,Y	3
M,O	1
M,Y	2
O,Y	2

STEP 4: (L2)

Item	**Count**
E,K	4
E,O	3
K,M	3
K,O	3

Item	Count
K,Y	3

ITERATION 3:

STEP 5: (C3)

Item	Count
E,K,O	3
K,M,O	1
K,M,Y	2

STEP 6: (L3)

Item	Count
E,K,O	3

Now, stop since no more combinations can be made in L3.

ASSOCIATION RULE:

1. [E,K] →→ 0 = 3/4 = 75%
2. [K,O] →→ E = 3/3 = 100%
3. [E,O] →→ K = 3/3 = 100%
4. E →→ [K,O] = 3/4 = 75%
5. K →→ [E,O] = 3/5 = 60%
6. O →→ [E,K] = 3/3 = 100%

∴∴ Rule no. 5 is discarded because confidence ≥≥70%
So, Rule 1, 2,3,4,6 are selected.

2. What is FP-Growth tree? Explain FP-Growth Tree Algorithm with an Example

Apriori Algorithm was explained in detail in our previous tutorial. In this tutorial, we will learn about Frequent Pattern Growth – FP Growth is a method of mining frequent itemsets.
As we all know, Apriori is an algorithm for frequent pattern mining that focuses on generating itemsets and discovering the most frequent itemset. It greatly reduces the size of the itemset in the database, however, Apriori has its own shortcomings as well.

Shortcomings Of Apriori Algorithm

1. Using Apriori needs a generation of candidate itemsets. These itemsets may be large in number if the itemset in the database is huge.
2. Apriori needs multiple scans of the database to check the support of each itemset generated and this leads to high costs.

These shortcomings can be overcome using the FP growth algorithm.

Frequent Pattern Growth Algorithm

This algorithm is an improvement to the Apriori method. A frequent pattern is generated without the need for candidate generation. FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree.

This tree structure will maintain the association between the itemsets. The database is fragmented using one frequent item. This fragmented part is called “pattern fragment”. The itemsets of these fragmented patterns are analyzed. Thus with this method, the search for frequent itemsets is reduced comparatively.

FP Tree

Frequent Pattern Tree is a tree-like structure that is made with the initial itemsets of the database. The purpose of the FP tree is to mine the most frequent pattern. Each node of the FP tree represents an item of the itemset.

The root node represents null while the lower nodes represent the itemsets. The association of the nodes with the lower nodes that is the itemsets with the other itemsets are maintained while forming the tree.

Frequent Pattern Algorithm Steps

The frequent pattern growth method lets us find the frequent pattern without candidate generation.

Let us see the steps followed to mine the frequent pattern using frequent pattern growth algorithm:
#1) The first step is to scan the database to find the occurrences of the itemsets in the database. This step is the same as the first step of Apriori. The count of 1-itemsets in the database is called support count or frequency of 1-itemset.

#2) The second step is to construct the FP tree. For this, create the root of the tree. The root is represented by null.

#3) The next step is to scan the database again and examine the transactions. Examine the first transaction and find out the itemset in it. The itemset with the max count is taken at the top, the next itemset with lower count and so on. It means that the branch of the tree is constructed with transaction itemsets in descending order of count.

#4) The next transaction in the database is examined. The itemsets are ordered in descending order of count. If any itemset of this transaction is already present in another branch (for example in the 1st transaction), then this transaction branch would share a common prefix to the root.
This means that the common itemset is linked to the new node of another itemset in this transaction.

#5) Also, the count of the itemset is incremented as it occurs in the transactions. Both the common node and new node count is increased by 1 as they are created and linked according to transactions.
#6) The next step is to mine the created FP Tree. For this, the lowest node is examined first along with the links of the lowest nodes. The lowest node represents the frequency pattern length 1. From this, traverse the path in the FP Tree. This path or paths are called a conditional pattern base.
Conditional pattern base is a sub-database consisting of prefix paths in the FP tree occurring with the lowest node (suffix).

#7) Construct a Conditional FP Tree, which is formed by a count of itemsets in the path. The itemsets meeting the threshold support are considered in the Conditional FP Tree.
#8) Frequent Patterns are generated from the Conditional FP Tree.

Example Of FP-Growth Algorithm

Support threshold=50%, Confidence= 60%

Table 1

Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4

Transaction	List of items
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

Solution:
Support threshold=50% => 0.5*6= 3 => min_sup=3

1. Count of each item
Table 2

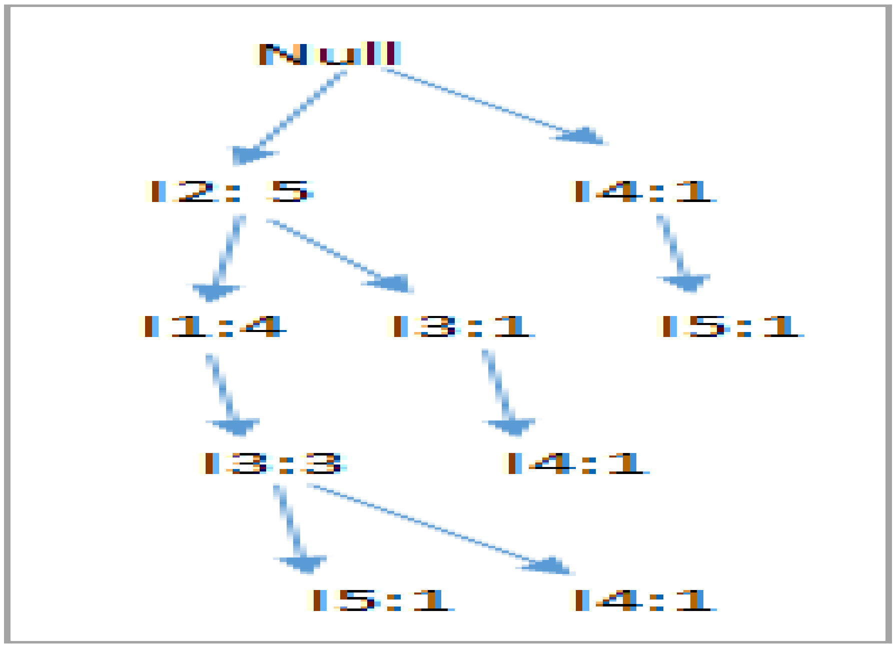
Item	Count
I1	4
I2	5
I3	4
I4	4
I5	2

2. Sort the itemset in descending order.
Table 3

Item	Count
I2	5
I1	4
I3	4
I4	4

3. Build FP Tree

1. Considering the root node null.
2. The first scan of Transaction T1: I1, I2, I3 contains three items {I1:1}, {I2:1}, {I3:1}, where I2 is linked as a child to root, I1 is linked to I2 and I3 is linked to I1.
3. T2: I2, I3, I4 contains I2, I3, and I4, where I2 is linked to root, I3 is linked to I2 and I4 is linked to I3. But this branch would share I2 node as common as it is already used in T1.
4. Increment the count of I2 by 1 and I3 is linked as a child to I2, I4 is linked as a child to I3. The count is {I2:2}, {I3:1}, {I4:1}.
5. T3: I4, I5. Similarly, a new branch with I5 is linked to I4 as a child is created.
6. T4: I1, I2, I4. The sequence will be I2, I1, and I4. I2 is already linked to the root node, hence it will be incremented by 1. Similarly I1 will be incremented by 1 as it is already linked with I2 in T1, thus {I2:3}, {I1:2}, {I4:1}.
7. T5:I1, I2, I3, I5. The sequence will be I2, I1, I3, and I5. Thus {I2:4}, {I1:3}, {I3:2}, {I5:1}.
8. T6: I1, I2, I3, I4. The sequence will be I2, I1, I3, and I4. Thus {I2:5}, {I1:4}, {I3:3}, {I4 1}.

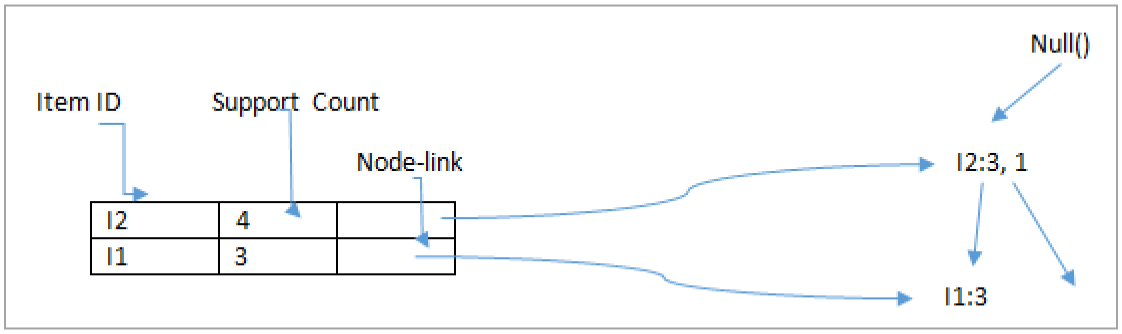


4. Mining of FP-tree is summarized below:

1. The lowest node item I5 is not considered as it does not have a min support count, hence it is deleted.
2. The next lower node is I4. I4 occurs in 2 branches , {I2,I1,I3:,I41},{I2,I3,I4:1}. Therefore considering I4 as suffix the prefix paths will be {I2, I1, I3:1}, {I2, I3: 1}. This forms the conditional pattern base.
3. The conditional pattern base is considered a transaction database, an FP-tree is constructed. This will contain {I2:2, I3:2}, I1 is not considered as it does not meet the min support count.
4. This path will generate all combinations of frequent patterns : {I2,I4:2},{I3,I4:2},{I2,I3,I4:2}
5. For I3, the prefix path would be: {I2,I1:3},{I2:1}, this will generate a 2 node FP-tree : {I2:4, I1:3} and frequent patterns are generated: {I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}.
6. For I1, the prefix path would be: {I2:4} this will generate a single node FP-tree: {I2:4} and frequent patterns are generated: {I2, I1:4}.

Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
I4	{I2,I1,I3:1},{I2,I3:1}	{I2:2, I3:2}	{I2,I4:2},{I3,I4:2},{I2,I3,I4:2}
I3	{I2,I1:3},{I2:1}	{I2:4, I1:3}	{I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}
I1	{I2:4}	{I2:4}	{I2,I1:4}

The diagram given below depicts the conditional FP tree associated with the conditional node I3.



Advantages Of FP Growth Algorithm

1. This algorithm needs to scan the database only twice when compared to Apriori which scans the transactions for each iteration.
2. The pairing of items is not done in this algorithm and this makes it faster.
3. The database is stored in a compact version in memory.
4. It is efficient and scalable for mining both long and short frequent patterns.

Disadvantages Of FP-Growth Algorithm

1. FP Tree is more cumbersome and difficult to build than Apriori.
2. It may be expensive.
3. When the database is large, the algorithm may not fit in the shared memory.

FP Growth vs Apriori

FP Growth	**Apriori**
Pattern Generation	
FP growth generates pattern by constructing a FP tree	Apriori generates pattern by pairing the items into singletons, pairs and triplets.
Candidate Generation	
There is no candidate generation	Apriori uses candidate generation
Process	
The process is faster as compared to Apriori. The runtime of process increases linearly with increase in number of itemsets.	The process is comparatively slower than FP Growth, the runtime increases exponentially with increase in number of itemsets
Memory Usage	
A compact version of database is saved	The candidates combinations are saved in memory

Conclusion

The Apriori algorithm is used for mining association rules. It works on the principle, "the non-empty subsets of frequent itemsets must also be frequent". It forms k-itemset candidates from (k-1) itemsets and scans the database to find the frequent itemsets.

Frequent Pattern Growth Algorithm is the method of finding frequent patterns without candidate generation. It constructs an FP Tree rather than using the generate and test strategy of Apriori. The focus of the FP Growth algorithm is on fragmenting the paths of the items and mining frequent patterns.