Module-5

Frequent Pattern (FP) Growth algorithm.

Disadvantages 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.

Steps:

The 1st scan of the database is the same as Apriori — It derives the set of frequent 1- itemsets and their support counts. The set is sorted in the descending order (sup-count).

Itemset	support-count
I2	7
I1	6
I3	6
I4	2
I5	2

L = {{I2 : 7}, {I1 : 6}, {I3 : 6}, {I4 : 2}, {I5 : 2}}

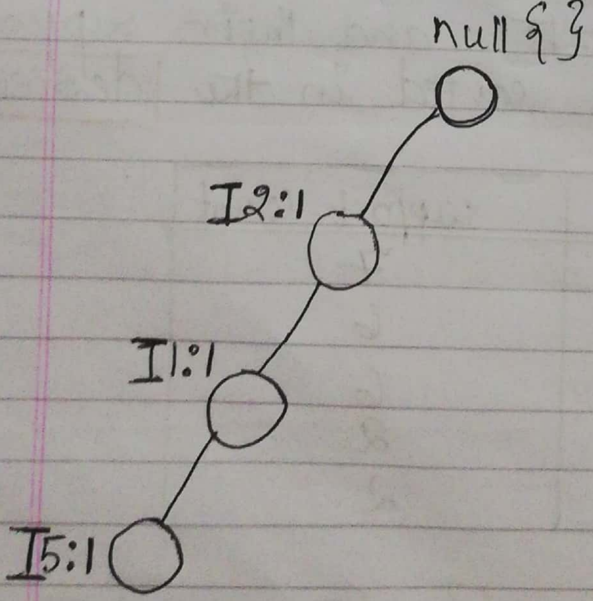
An FP- tree is then constructed :-

→ First, create the root of the tree, labell it "null"

→ The items in each transaction are processed in L-order and a branch is created for each transaction.

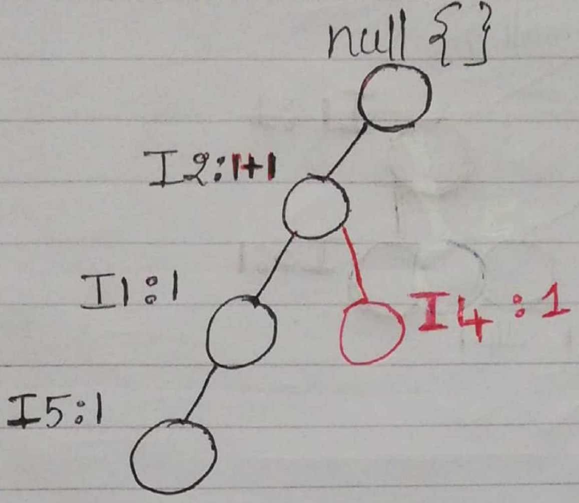
→ eg: Scan the 1st transaction
T100 : I1, I2, I5
which has 3 items in L order as
T100 : I2, I1, I5
leads to the construction of the 1st branch of the tree with 3 nodes
⟨I2 : 1⟩, ⟨I1 : 1⟩, ⟨I5 : 1⟩.

Step-1



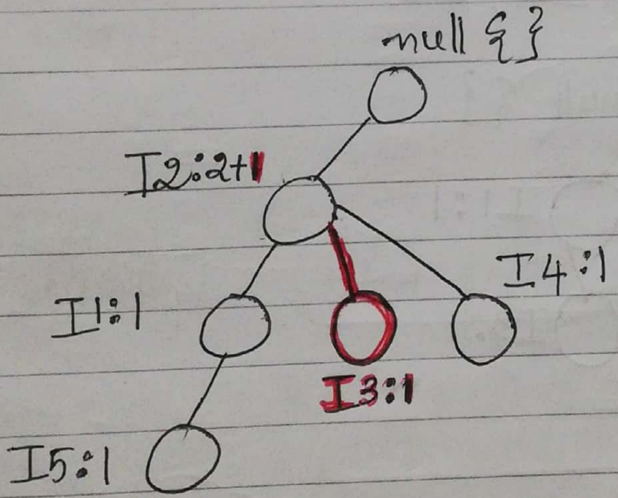
→ 2nd transaction :

T200 : I2 , I4 in L order.



→ 3rd transaction :

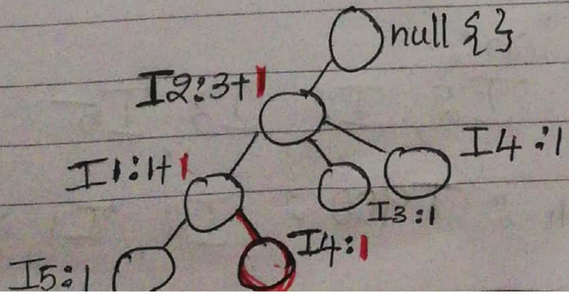
T300 : I2, I3 in L order



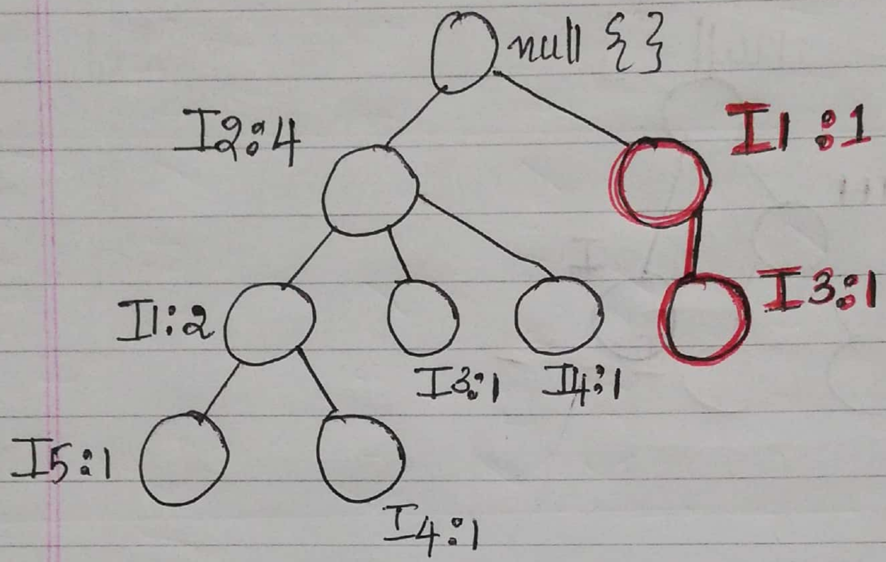
→ 4th transaction :

T400 : I1, I2 , I4 arrange in L order a

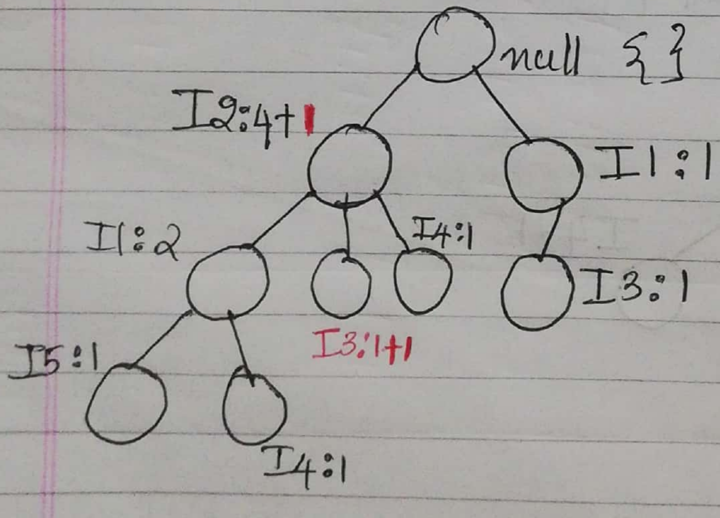
T400 : I2, I1, I4



→ 5th transaction :
T500 : I1 , I3 in L order



→ 6th transaction :
T600 : I2, I3.

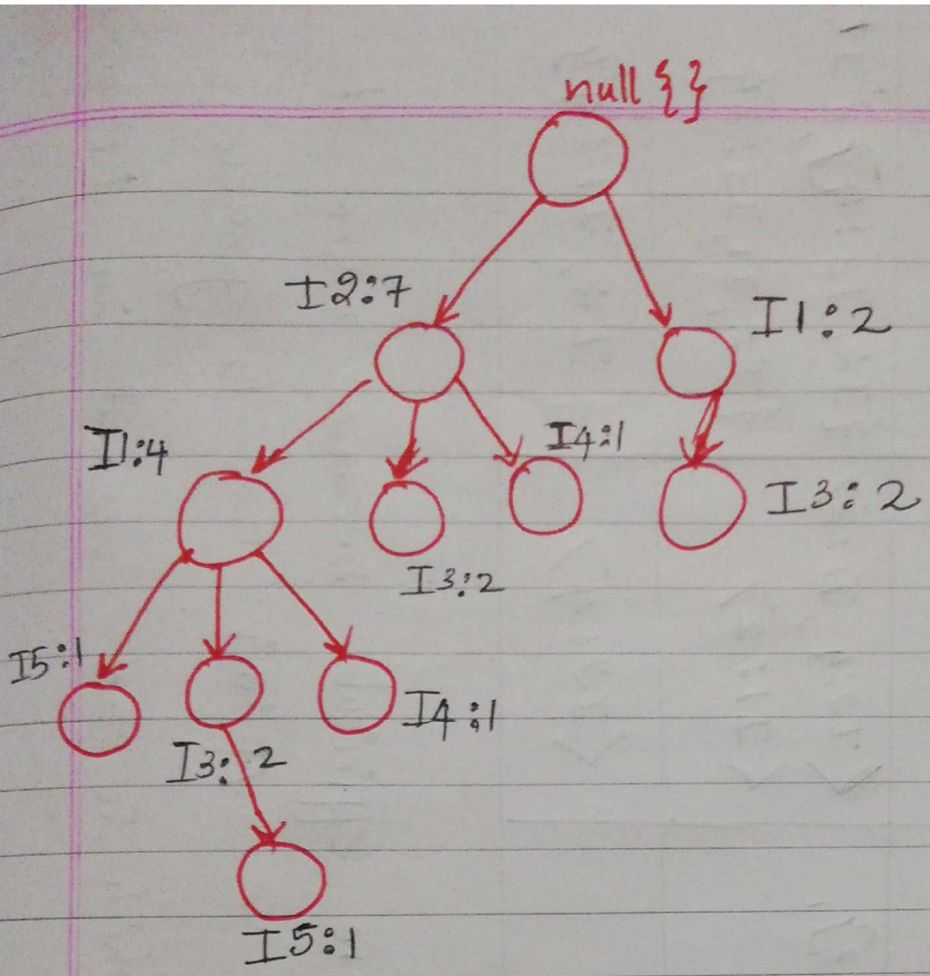


→ 7th transaction :
T700 : I1 , I3

→ 8th transaction :
T800 : I2, I1, I3 , I5

→ 9th transaction : T900 : I2, I1, I3

Scanned with CamScanner



this transforms mining frequent pattens in databases to mining the FP-tree.

Step -2

Mining the FP- tree by creating conditional (sub-) pattern bases.

conditional Base pattern: ① Start from the node with min- support value [I5].
② exclude the node with max support value [I2].

Item	Conditional Pattern Base	Conditional FP tree	Frequent pattern Generated

I5	{I2, I1 : 1}, {I2, I1, I3 : 1}	⟨I2 : 2, I1 : 2⟩	{I2, I5 : 2} {I1, I5 : 2}
			{I2, I1, I5 : 2}

I4	{{I2, I1 : 1}, {I2 : 1}}	⟨I2 : 2⟩	{I2, I4 : 2}

I3	{{I2, I1 : 2}, {I2 : 2},
	{I1 : 2}}
		⟨I2 : 4, I1 : 2⟩
		⟨I1 : 2⟩
			{I2, I3 : 4}, {I1, I3 : 4}
			{I2, I1, I3 : 2}

I1	{{I2 : 4}}	⟨I2 : 4⟩	{I2, I1 : 4}

FP Growth algorithm is more efficient because when it comes to large databases - Apriori has to generate large no. of candidates which is not needed in FP Growth.