

**Department of Computer Science and Engineering(UG Studies)**

**PES University, Bangalore-560085**

|  |  |
| --- | --- |
| **Session :** Aug - Dec 2017  **Credits :** 0-0-2-0-1 | UE14CS405 : Machine Learning Lab |
| **Lab # : 12** | On a real/online store purchase data set, implement FP-Growth Tree based itemset mining |

**Algorithm 1: FP-tree construction**

*Input*: A transaction database DB and a minimum support threshold.

*Output*: FP-tree, the frequent-pattern tree of DB.

*Method*: The FP-tree is constructed as follows.

1. Scan the transaction database DB once. Collect F, the set of frequent items, and the support of each frequent item. Sort F in support-descending order as FList, the list of frequent items.
2. Create the root of an FP-tree, T, and label it as “null”. For each transaction Trans in DB do the following:

* Select the frequent items in Trans and sort them according to the order of FList. Let the sorted frequent-item list in Trans be [ p | P], where p is the first element and P is the remaining list. Call insert tree([ p | P], T ).
* The function insert tree([ p | P], T ) is performed as follows. If T has a child N such that N.item-name = p.item-name, then increment N ’s count by 1; else create a new node N , with its count initialized to 1, its parent link linked to T , and its node-link linked to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert tree(P, N ) recursively.

By using this algorithm, the FP-tree is constructed in two scans of the database. The first scan collects and sort the set of frequent items, and the second constructs the FP-Tree.

### **FP-Growth Algorithm**

After constructing the FP-Tree it’s possible to mine it to find the complete set of frequent patterns.

**Algorithm 2: FP-Growth**

*Input*: A database DB, represented by FP-tree constructed according to Algorithm 1, and a minimum support threshold.

*Output*: The complete set of frequent patterns.

*Method*: call FP-growth(FP-tree, null).

Procedure FP-growth(Tree, a) {

if Tree contains a single prefix path then { // Mining single prefix-path FP-tree

let P be the single prefix-path part of Tree;

let Q be the multipath part with the top branching node replaced by a null root;

for each combination (denoted as ß) of the nodes in the path P do

generate pattern ß ∪ a with support = minimum support of nodes in ß;

let freq pattern set(P) be the set of patterns so generated;

}

else let Q be Tree;

for each item ai in Q do { // Mining multipath FP-tree

generate pattern ß = ai ∪ a with support = ai .support;

construct ß’s conditional pattern-base and then ß’s conditional FP-tree Tree ß;

if Tree ß ≠ Ø then

call FP-growth(Tree ß , ß);

let freq pattern set(Q) be the set of patterns so generated;

}

return(freq pattern set(P) ∪ freq pattern set(Q) ∪ (freq pattern set(P) × freq pattern set(Q)))

}

### **To Do:**

### 1) Fill the missing code and use real purchase store data .

### **Outcomes:**

### 1)We require only two scans for Frequent Pattern growth algorithm to scan the database unlike apriori Algorithms

### 2)It is faster compared to Apriori technique