Min support threshold \( \varepsilon = 3 \)

1. One scan of DB to identify the set of frequent items

2. Store the set of frequent items of each transaction in a tree

For convenience, the frequent itemsets of each transaction is listed in this ordering:

<table>
<thead>
<tr>
<th>TID</th>
<th>Ordered frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>f, c, a, m, p</td>
</tr>
<tr>
<td>200</td>
<td>f, c, a, b, m</td>
</tr>
<tr>
<td>300</td>
<td>f, c, b, m</td>
</tr>
<tr>
<td>400</td>
<td>c, b, p</td>
</tr>
<tr>
<td>500</td>
<td>f, b</td>
</tr>
</tbody>
</table>

For the second time:

1. Create a “null” root
2. Scan the DB for second time
3. Add the paths which are the ordered frequent items
4. Share the path until a different item comes up
5. Branch and create a sub-path

Frequent itemsets:

- \(<m:3, am:3, cm:3, fm:3, cam:3, pm:3>\)

For the third time:

1. One scan of DB to identify the set of frequent items \( \varepsilon = 3 \)
2. Store the set of frequent items of each transaction in a tree

a. To facilitate tree traversal, build item header table
b. Nodes with the same item-name are linked

Node p (p:3)

- \(<p:3, cp:3>\)

Node m (m:3)

- \(<m:3, am:3, cm:3, fm:3, cam:3, pm:3>\)

Node c (c:4)

- \(<c:4, fc:3>\)

Node f (f:4)

- \(<f:4, fc:3>\)
Pattern Tree Generation

Step 1: Construct a P-tree
1. Generate a P-tree by inserting sorted transactions
2. Record the actual frequency of every item into the item frequency list L
3. Sort L according to item frequency

An Example

<table>
<thead>
<tr>
<th>TID</th>
<th>Items Sorted in alphabetic</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>a, c, d, f, g, i, m, p</td>
</tr>
<tr>
<td>200</td>
<td>a, c, f, l, m, o</td>
</tr>
<tr>
<td>300</td>
<td>b, f, h, j, o</td>
</tr>
<tr>
<td>400</td>
<td>b, o, f, h, j</td>
</tr>
<tr>
<td>500</td>
<td>a, f, c, e, l, p, m, n</td>
</tr>
</tbody>
</table>

After Step 1, We Get...

Item Frequency list L: <(f:5), (a:3), (c:3), (m:3), (o:3), (b:2), (h:2), (j:2), (l:2), (p:2), (d:1), (e:1), (g:1), (i:1), (n:1)>

Step 2: Restructure the P-tree
- The restructuring of the P-tree consists of similar insertions in the first step
- Need to sort the path according to the updated item frequency list L before inserting it into a new P-tree

FP-tree Generation from P-tree
1. Get minimum support threshold from users
2. Compute frequent item list
3. Check the P-tree to get rid of infrequent ones and their sub trees
   1. Check from the root to leaves when frequent items are more than infrequent items
   2. Check from leaves to the root when infrequent items are more than frequent items

Update P-tree With New Data
- Apriori or FP-tree algorithm has to rerun the whole algorithm from scratch
  - How can we do that?
  1. Insert the new transactions into the P-tree according to the item frequency list and then restructure it according to the updated item frequency list
  2. Restructure the P-tree

Root
├ a:3-c:3-m:3
│ ├ p:1-d:1-g:1-i:1
│ │ └ o:1-l:1
│ │   └ e:1-l:1-p:1-m:1-n:1
│ └ f:2-b:2-h:2-j:2-o:2

Path Sorted Path
1. a:1, c:1, d:1, f:1, g:1, i:1, m:1, p:1
2. a:1, c:1, f:1, l:1, m:1, o:1
3. a:1, c:1, f:1, e:1, l:1, p:1, m:1, n:1
4. b:2, f:2, h:2, j:2, o:2