About the Algorithm and Implementation

APRIORI

Algorithm: Most of the algorithm is based iteratively where all possible item sets are checked (2^k, where k is the total number of items). Some of the sets are skipped according to the apriori rule.

Implementation: A data structure is made that has a sparse [0,1] matrix with items as attribute and transactions as rows. Initially item sets with length 1 are checked for their support count and the iteratively the length of the item set is increased and checked if it is meeting the support count.

For every item set meeting the support criteria it is added to the list along with its support count. If a particular item when added to the item set does not meet the criteria it is added to the skip list and in further iterations those item sets are not included.

Thus time complexity is: $O(n*(2^k-m))$, where n is the number of transactions, k is the number of items and m is the number of skips.

In order to generate the rules, we have simply checked the frequent item sets of length greater than 1, and created bipartite partitions of the sets. As we were already storing the support count we didn’t have to go through the database again only through the frequent item sets and again implemented the skip rule if an item to the right did not satisfy the confidence rule.

FP-Growth

Algorithm: First the FP tree is constructed then recursively mined for frequent item sets. Thus the frequent sets are mined much faster as the time depends on the length of the tree, rather than the number of transactions.

Implementation: First the FP tree is built after each transaction is arranged according to support count and then each item is added to the tree. Essentially every node of tree has a link to its parent and has a dictionary of nodes of its children along with its support count.

In order to mine the tree for frequent item sets, conditional trees are built for each item and then each conditional item tree is recursively mined for frequent items giving the whole set of frequent item sets.

Time complexity for mining frequent item sets is approximately: O(n*n*k) where n is the length of the tree and k is the number of items.

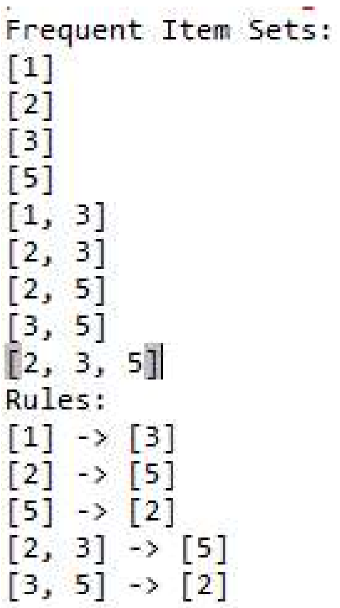
Rule mining in FP Tree was not done as it will be pretty much same as the apriori rule mining on frequent item sets.

EXAMPLE RUN:

DATA:

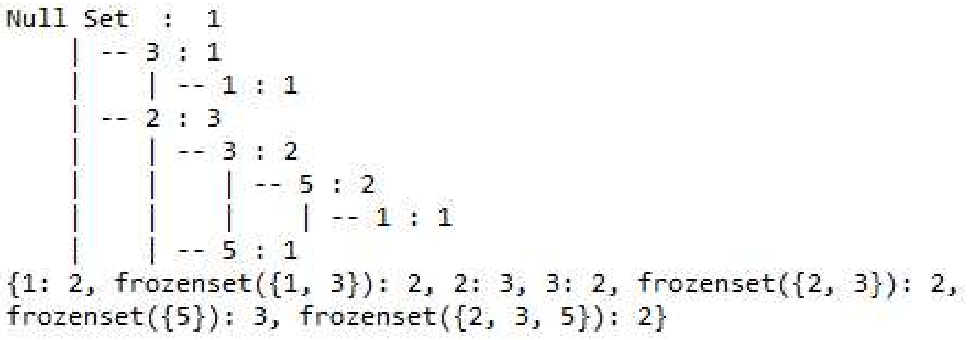
3 1
2 3 5
2 3 5 1
2 5

APRIORI:



```
Frequent Item Sets:
[1]
[2]
[3]
[5]
[1, 3]
[2, 3]
[2, 5]
[3, 5]
[2, 3, 5]
Rules:
[1] -> [3]
[2] -> [5]
[5] -> [2]
[2, 3] -> [5]
[3, 5] -> [2]
```

FP TREE:



```
Null Set  :  1
    | -- 3 : 1
    |    | -- 1 : 1
    | -- 2 : 3
    |    | -- 3 : 2
    |    |    | -- 5 : 2
    |    |    |    | -- 1 : 1
    |    | -- 5 : 1
{1: 2, frozenset({1, 3}): 2, 2: 3, 3: 2, frozenset({2, 3}): 2,
frozenset({5}): 3, frozenset({2, 3, 5}): 2}
```