1. Basic Terminologies

The basic terminologies and definitions related to BIN-Tree are explained and illustrated using examples based on Example 1.

Example 1
Consider a sample transaction database with eight transactions and 67 items as shown in Table 1. The first column represents the transaction ID (TID), seconds represents the items in each transaction and the last column represents the transaction weight. Assume that the user defined support thresholds are given as $min_{freq} = 0 \cdot 5$ and $min_{rare} = 0 \cdot 2$.

Table 1. Sample Database with transactions' weight

TID	Transaction items	Binary Representation (Transaction Weight)
1	1,2	00...0110
2	3,4	00...011000
3	1,3	00...01010
4	1,2,3	00...01110
5	4	00...010000
6	1,2,3,65,66,67	1110...01110
7	1,2,3,57,58,59,60,61	00...0111110...01110
8	1,2,3,62,63,64	00...01110...01110

Definition 1.1. ***Transaction database:*** *A transaction database is a collection of r transactions represented as T_d=$\{t_1, t_2,....,t_r\}$ where each t_i, i=1,2,....r represents individual transaction.*

Definition 1.2. ***Itemset:*** *A collection of n items in T_d is called as itemset which is represented as $I = \{i_1, i_2,....,i_n\}$ where $i_1, i_2,....,i_n$ represents individual items.*

In Example 1, I=$\{1,2,3,4,57,58,59,60,61,62,63,64,65,66,67\}$

Definition 1.3. ***k-Itemset:*** *A collection of any k items from I is known as k-Itemset.*

Definition 1.4. ***Support:*** *The support of any itemset I is its total number of occurrences in a database with r transactions. The support of an item I is represented as $Sup(I)$ and it is calculated as given in equation (1).*

$$Sup(I) = \frac{|\{t \in T_d : I \subseteq t\}|}{|T_d|} \tag{1}$$

In Example 1, Sup(1)= $\frac{6}{8}$ =0.75, Sup(1,2)= $\frac{5}{8}$ = 0.625 and Sup(4)= $\frac{2}{8}$ =0.25

Definition 1.5. ***Frequent Itemset:*** *If the occurrence of an itemset in the transaction database is greater than or equal to the user defined minimum threshold min_{freq} then that itemset is considered to be frequent itemset.*

In Example 1, itemset $\{1\}$ and $\{1,2\}$ are considered to be frequent itemset since its support is greater than the user defined min_{freq} support threshold.

Definition 1.6. ***Rare Itemset:*** *If the occurrence of an itemset in the database is between two thresholds min_{rare} and min_{freq} then it is considered to be rare itemset.*

In Example 1, itemset $\{2\}$ is considered to be rare itemset since its support is between the user defined min_{freq} and min_{rare} support thresholds.

Definition 1.7. ***Bitset:*** *Bitset is a collection of 0s and 1s where 0 represents the absence of the position value in the array and 1 represents its presence. The leftmost bit represents a higher place value and the rightmost bit represents the lower place value.*

Definition 1.8. ***Weight:*** *A Weight W_i is an encoded representation of a transaction t_i in a database. It is an array of bitset where 0 represents the absence of the item in the transaction and 1 represents its presence.*

In Example 1, the weight of transaction 1 is 64 zeroes followed by 110 (00.....00110).

Definition 1.9. $\succ$ ***relation:*** *$\forall T_i, T_j \in T_d$; $T_j \succ T_i$ if and only if $T_i \subset T_j$. In Example 1, T6 $\succ$T4 $\succ$ T1.*

Definition 1.10. ***BIN-Tree:****. A BIN-Tree has following characteristics:*

1. *It has a root node that contains NULL and the transactions from the database are inserted as its child nodes.*
2. *Every other node in the tree represents an encoded transaction.*
3. *The parent(P) and child (C) node, except root node, in the tree has C$\succ$P relation and all the common items are stored in the parent node before updating the tree.*
4. *Each node in the tree has four fields: Weight, Count, Child Pointer and Next Pointer. Weight contains the transaction weight, Count contains the support count of each transaction, Child Pointer holds the pointer to child node and Next Pointer holds the pointer to sibling nodes at same level.*

2. BIN-Tree Construction Algorithm

Algorithm 1 Binary_Count_Tree()

Input: Transactional Database T_D
Output: Binary Count Tree: a compact and complete tree data structure

for each transaction $T_k \in T_D$ **do**
  $Weight_k \leftarrow$ bitset representation of all items in T_k
  if the tree is empty **then**
    create node $N_{new} \leftarrow$ createNode($Weight_k$)
  else
    traverse $\leftarrow$ root.child
    complete $\leftarrow 0$
    while complete $\neq 1$ **do**
      if weight(traverse) = $Weight_k$ **then**
        count(traverse) $\leftarrow$ count(traverse) + 1
        complete $\leftarrow 1$
      else if $Weight_k \wedge$ weight(traverse)=weight(traverse) **then**
        if traverse has no child **then**
          create a node $N_{new} \leftarrow$ createNode($Weight_k$)
          remove all factors in $Weight_k$ that are in common with traverse
          traverse.child $\leftarrow N_{new}$
          complete $\leftarrow 1$
        else
          remove all factors in traverse that are in common with $Weight_k$
          assign traverse to the child of traverse
        end if
      else if $Weight_k \wedge$ weight(traverse)=$Weight_k$ **then**
        create $Node_{new} \leftarrow$ createNode($Weight_k$)
        place $Node_{new}$ in place of traverse
        assign traverse as the child of $Node_{new}$
        remove all factors in traverse that are in common with $Node_{new}$
        complete $\leftarrow 1$
      else
        assign traverse to the right of traverse
      end if
    end while
  end if
end for each