1. **Write an algorithm to generate all frequent itemset without Candidate generation**

Algorithm: FP_growth. Mine frequent itemsets using an FP-tree by pattern fragment growth.

Input:

- D, a transaction database;
- *min_sup*, the minimum support count threshold.

Output: The complete set of frequent patterns.

Method:

1. The FP-tree is constructed in the following steps:
   (a) Scan the transaction database D once. Collect F, the set of frequent items, and their support counts. Sort F in support count descending order as L, the *list* of frequent items.
   (b) Create the root of an FP-tree, and label it as "null." For each transaction *Trans* in D do the following. Select and sort the frequent items in *Trans* according to the order of L. 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$([p|P], T)$, which is performed as follows. If T has a child N such that $N.item\text{-}name = p.item\text{-}name$, then increment N's count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link to the nodes with the same *item-name* via the node-link structure. If P is nonempty, call insert_tree(P, N) recursively.
2. The FP-tree is mined by calling FP_growth$(FP_tree, null)$, which is implemented as follows.

```
procedure FP_growth(Tree, α)
(1)  if Tree contains a single path P then
(2)     for each combination (denoted as β) of the nodes in the path P
(3)        generate pattern β∪α with support_count = minimum support count of nodes in β;
(4)  else for each a_i in the header of Tree {
(5)     generate pattern β = a_i ∪ α with support_count = a_i.support_count;
(6)     construct β's conditional pattern base and then β's conditional FP_tree Tree_β;
(7)     if Tree_β ≠ ∅ then
(8)        call FP_growth(Tree_β, β); }
```

The FP-growth algorithm for discovering frequent itemsets without candidate generation.

2. An itemset X is called a *generator* on a data set D if there does not exist a proper sub-itemset $Y \subset X$ such that $support(X) = support(Y)$. A generator X is a *frequent generator* if $support(X)$ passes the minimum support threshold. Let G be the set of all frequent generators on a data set D.

   (a) Can you determine whether an itemset A is frequent and the support of A, if it is frequent, using only G and the support counts of all frequent generators? If yes, present your algorithm.

Otherwise, what other information is needed? Can you give an algorithm assuming the information needed is available?

(b) What is the relationship between closed itemsets and generators?

Answer:

(a) Can you determine whether an itemset A is frequent and the support of A, if it is frequent, using only G and the support counts of all frequent generators? If yes, present your algorithm. Otherwise, what other information is needed? Can you give an algorithm assuming the information needed is available?

No, the frequent generators alone do not provide enough information to represent the complete set of frequent itemsets. We need information such as the "positive border" of the frequent generators, that is, all itemsets l such that l is frequent, l is not a frequent generator, and all proper subsets of l are frequent generators. With this information we may use the following algorithm:

Algorithm: InferSupport. Determine if an itemset is frequent.

Input:

- l is an itemset;
- FG is the set of frequent generators;
- $PBd(FG)$ is the positive border of FG;

Output: Support of l if it is frequent, otherwise -1.

Method:

```
(1)  if l ∈ FG or l ∈ PBd(FG) then{
(2)      return support(l);
(3)  } else {
(4)      for all l' ⊂ l and l' ∈ PBd(FG)
(5)          Let a be the item such that l'' = l' — {a} and l'' ∈ FG
                 and support(l'') = support(l');
(6)          l = l — {a};
(7)      if l ∈ FG or l ∈ PBd(FG) then{
(8)          return support(l);
(9)      } else {
(10)         return -1;
(11)     }
(12) }
```

Reference: LIU, G., LI, J., and WONG, L. 2008. A new concise representation of frequent itemsets using generators and a positive border. *Knowledge and Information Systems 17*, 35-56.

(b) ***Very generally, they can be considered as opposites. This is because a closed itemset has no proper super-itemset with the same support, while a generator has no proper sub-itemset with the same support.***

■

3. **Write an algorithm to generate frequent itemset using Candidate generation approach**

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- min_sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ ϕ; k++) {
(3)       Ck = apriori_gen(Lk−1);
(4)       for each transaction t ∈ D { // scan D for counts
(5)           Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)           for each candidate c ∈ Ct
(7)               c.count++;
(8)       }
(9)       Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;

procedure apriori_gen(Lk−1:frequent (k − 1)-itemsets)
(1)   for each itemset l1 ∈ Lk−1
(2)       for each itemset l2 ∈ Lk−1
(3)           if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)               c = l1 ⋈ l2; // join step: generate candidates
(5)               if has_infrequent_subset(c, Lk−1) then
(6)                   delete c; // prune step: remove unfruitful candidate
(7)               else add c to Ck;
(8)           }
(9)   return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
          Lk−1: frequent (k − 1)-itemsets); // use prior knowledge
(1)   for each (k − 1)-subset s of c
(2)       if s ∉ Lk−1 then
(3)           return TRUE;
(4)   return FALSE;
```

The Apriori algorithm for discovering frequent itemsets for mining Boolean association rules.

4. Let c be a candidate itemset in C_k generated by the Apriori algorithm. How many length-(k 1) subsets do we need to check in the prune step? According to your answer to the above question, can you give an improved version of procedure has infrequent subset.

Answer:

Because c was generated from two length-$(k-1)$ frequent itemsets, we do not need to check these two subsets. That is, even though there are k length-$(k-1)$ subsets of c, we only need to check $k-2$ of them. One way to push this into has infrequent subset would be to additionally pass l_1 and l_2, and prevent searching L_{k-1} for these because we already know they are frequent. ■

5. Suppose that frequent itemsets are saved for a large transactional database, *DB*. Discuss how to efficiently mine the (global) association rules under the same minimum support threshold, if a set of new transactions, denoted as ΔDB, is *(incrementally) added in*?

Answer:

We can treat ΔDB and *DB* as two partitions.

- For itemsets that are frequent in *DB*, scan ΔDB once and add their counts to see if they are still frequent in the updated database.
- For itemsets that are frequent in ΔDB but not in *DB*, scan DB once to add their counts to see if they are frequent in the updated DB.

■

6. Give a short example to show that items in a strong association rule may actually be *negatively correlated*.

Answer:

Consider the following table:

	A	$\overline{A}$	Σ_{row}
B	65	35	100
$\overline{B}$	40	10	50
Σ_{col}	105	35	150

Let the minimum support be 40%. Let the minimum confidence be 60%. $A \Rightarrow B$ is a strong rule because it satisfies minimum support and minimum confidence with a support of 65/150 = 43.3% and a confidence of 65/100 = 61.9%. However, the correlation between A and B is $corr_{A,B} = \frac{0.433}{0.700 \times 0.667} = 0.928$, which is less than 1, meaning that the occurrence of A is negatively correlated with the occurrence of B.

■

7. The following contingency table summarizes supermarket transaction data, where *hot dogs* refers to the transactions containing hot dogs, *hotdogs* refers to the transactions that do not contain hot dogs, *hamburgers* refers to the transactions containing hamburgers, and *hamburgers* refers to the transactions that do not contain hamburgers.

	hot dogs	*hotdogs*	Σ_{row}
hamburgers	2000	500	2500
hamburgers	1000	1500	2500
Σ_{col}	3000	2000	5000

(a) Suppose that the association rule "*hot dogs* $\Rightarrow$ *hamburgers*" is mined. Given a minimum support threshold of 25% and a minimum confidence threshold of 50%, is this association rule strong?

(b) Based on the given data, is the purchase of *hot dogs* independent of the purchase of *hamburgers*? If not, what kind of *correlation* relationship exists between the two?

(c) Compare the use of the *all confidence*, *max confidence*, *Kulczynski*, and *cosine* measures with *lift* and *correlation* on the given data.

Answer:

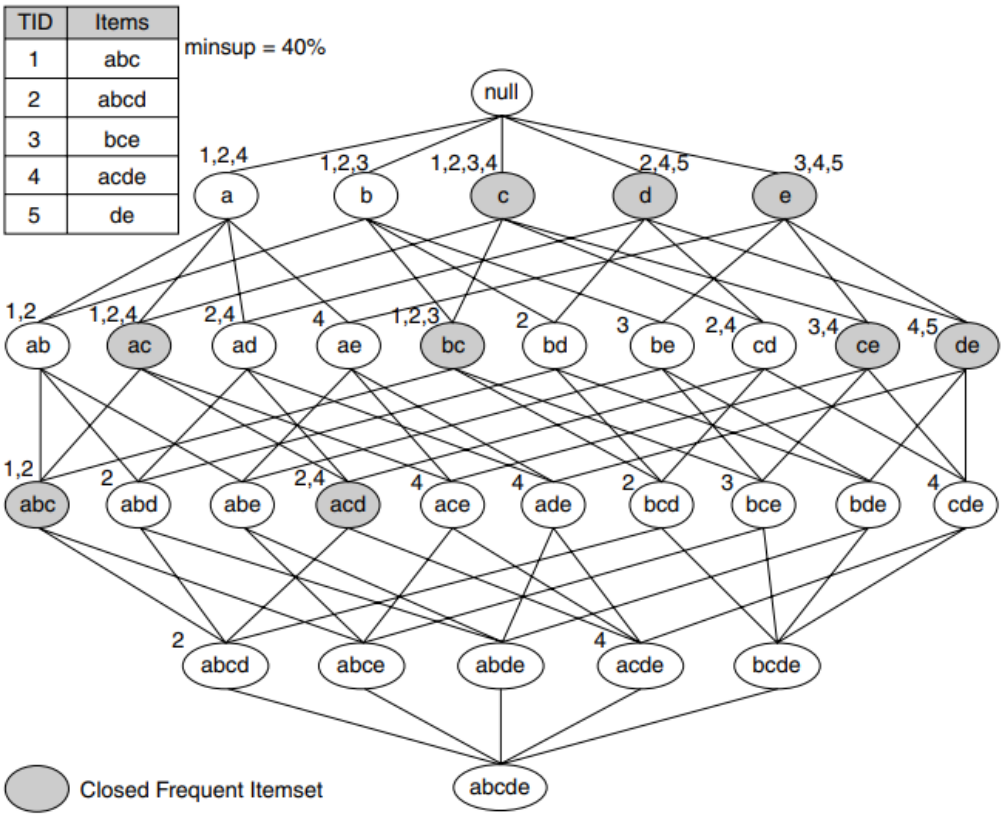
(a) Suppose that the association rule "*hotdogs -> hamburgers*" is mined. Given a minimum support threshold of 25% and a minimum confidence threshold of 50%, is this association rule strong?

For the rule, support = 2000/5000 = 40%, and confidence = 2000/3000 = 66.7%. Therefore, the association rule is strong.

(b) Based on the given data, is the purchase of *hotdogs* independent of the purchase of *hamburgers*? If not, what kind of *correlation* relationship exists between the two?

$corr_{\{hotdog,\ hamburger\}} = P(\{hot\ dog,\ hamburger\}) / (P(\{hot\ dog\}) \times P(\{hamburger\})) = 0.4/(0.5 \times 0.6) = 1.33 > 1$. So, the purchase of hotdogs is NOT independent of the purchase of hamburgers. There exists a POSITIVE correlation between the two.

- Traditional frequent itemset mining (Apriori, FP-growth) usually **ignores multiple counts** of the same item in a transaction.
- In reality, **quantities matter** (e.g., 4 cakes vs. 1 cake).
- One approach → **extend item representation** (e.g., Cake:1, Cake:2, Cake:3 … treated as separate).
- **Apriori modification**:
  - Generate candidate itemsets including quantity levels.
  - Support count must consider both presence and quantity.
- **FP-growth modification**:
  - FP-tree nodes must also record **item counts/quantities**.
  - Projected databases → keep track of item + quantity patterns.
- Optimization possible (e.g., mine **closed/quantitative itemsets**) to reduce redundancy.

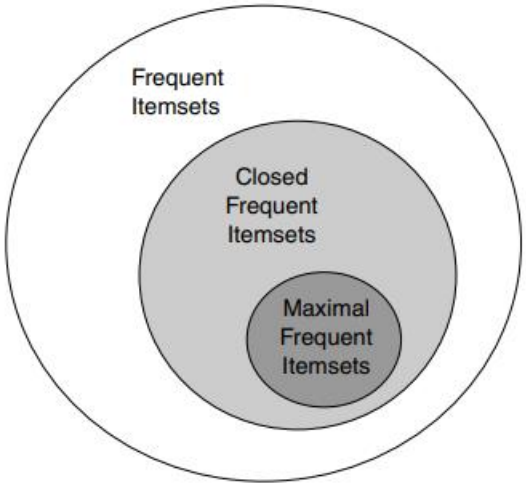


Put another way, X is not closed if at least one of its immediate supersets has the same support count as X. Examples of closed itemsets are shown in above figure. To better illustrate the support count of each itemset, we have associated each node (itemset) in the lattice with a list of its corresponding transaction IDs. For example, since the node {b, c} is associated with transaction IDs 1, 2, and 3, its support count is equal to three. From the transactions given in this diagram, notice that every transaction that contains b also contains c. Consequently, the support for {b} is identical to {b, c} and {b} should not be considered a closed itemset. Similarly, since c occurs in every transaction that contains both a and d, the itemset {a, d} is not closed.

On the other hand, {b, c} is a closed itemset because it does not have the same support count as any of its supersets. (Closed Frequent Itemset). An itemset is a closed frequent itemset if it is closed and its support is greater than or equal to minsup. In the previous example, assuming that the support threshold is 40%, {b,c} is a closed frequent itemset because its support is 60%. The rest of the closed frequent itemsets are indicated by the shaded nodes. Algorithms are available to explicitly extract closed frequent itemsets from a given data set.

For example, consider the frequent itemset {a, d}. Because the itemset is not closed, its support count must be identical to one of its immediate supersets. The key is to determine which superset (among {a, b, d}, {a, c, d}, or {a, d, e}) has exactly the same support count as {a, d}. The Apriori principle states that any transaction that contains the superset of {a, d} must also contain {a, d}. However, any transaction that contains {a, d} does not have to contain the supersets of {a, d}. For this reason, the support for {a, d} must be equal to the largest support among its supersets. Since {a, c, d} has a larger support than both {a, b, d} and {a, d, e}, the support for {a, d} must be identical to the support for {a, c, d}.

Using this methodology, an algorithm can be developed to compute the support for the non-closed frequent itemsets.



Relationships among frequent, maximal frequent, and closed frequent itemsets.

Closed frequent itemsets are useful for removing some of the redundant association rules. An association rule $X \longrightarrow Y$ is redundant if there exists another rule $X \longrightarrow Y$, where X is a subset of X and Y is a subset of Y , such that the support and confidence for both rules are identical. In the example shown in Figure 6.17, {b} is not a closed frequent itemset while {b, c} is closed. The association rule {b} $\longrightarrow$ {d, e} is therefore redundant because it has the same support and confidence as {b, c} $\longrightarrow$ {d, e}. Such redundant rules are not generated if closed frequent itemsets are used for rule generation. Finally, note that all maximal frequent itemsets are closed because none of the maximal frequent itemsets can have the same support count as their

immediate supersets. The relationships among frequent, maximal frequent, and closed frequent itemsets are shown in Figure

Apriori and FP growth Practice problems

TID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

Min supp=50% and Min conf 80%

(a) Compute the support for itemsets {e}, {b, d}, and {b, d, e} by treating each transaction ID as a market basket.
(b) Use the results in part (a) to compute the confidence for the association rules {b, d} $\longrightarrow$ {e} and {e} $\longrightarrow$ {b, d}. Is confidence a symmetric measure?
(c) Repeat part (a) by treating each customer ID as a market basket. Each item should be treated as a binary variable (1 if an item appears in at least one transaction bought by the customer, and 0 otherwise.)
(d) Use the results in part (c) to compute the confidence for the association rules {b, d} $\longrightarrow$ {e} and {e} $\longrightarrow$ {b, d}.
(e) Suppose s1 and c1 are the support and confidence values of an association rule r when treating each transaction ID as a market basket. Also, let s2 and c2 be the support and confidence values of r when treating each customer ID as a market basket. Discuss whether there are any relationships between s1 and s2 or c1 and c2.

Transaction ID	Items Bought
1	{Milk, Beer, Diapers}
2	{Bread, Butter, Milk}
3	{Milk, Diapers, Cookies}
4	{Bread, Butter, Cookies}
5	{Beer, Cookies, Diapers}
6	{Milk, Diapers, Bread, Butter}
7	{Bread, Butter, Diapers}
8	{Beer, Diapers}
9	{Milk, Diapers, Bread, Butter}
10	{Beer, Cookies}

(a) What is the maximum number of association rules that can be extracted from this data (including rules that have zero support)?
(b) What is the maximum size of frequent itemsets that can be extracted (assuming minsup > 0)?
(c) Write an expression for the maximum number of size-3 itemsets that can be derived from this data set.
(d) Find an itemset (of size 2 or larger) that has the largest support.
(e) Find a pair of items, a and b, such that the rules {a} $\longrightarrow$ {b} and {b} $\longrightarrow$ {a} have the same confidence.