

Assignment 1

Omkumar M. Patel 222110936

Q1

Please answer the following questions.

- (a) A hospital would like to partition their patients into similar groups according to their medical and demographic profiles. What data mining task is best suited to this problem?

Answer: *Descriptive — Clustering.*

Justification: Clustering groups similar records without labels (unsupervised), so it is suitable for partitioning patients by demographics and clinical features.

- (b) Suppose the hospital already knows for some patients whether or not they developed diabetes. Which data mining task would be suited to the problem of identifying other patients at risk of developing diabetes?

Answer: *Predictive — Classification.*

Justification: Labeled examples (developed diabetes: yes/no) make this a supervised learning problem; train a classifier to predict risk on unlabeled patients.

- (c) Suppose the hospital has recorded all the treatments received by patients. What data mining task would be best suited to finding sets of treatments that are often prescribed together with insulin therapy?

Answer: *Descriptive — Association analysis / Frequent pattern mining.*

Justification: Association mining (market-basket style) finds itemsets of co-occurring treatments; it can identify treatments that frequently accompany insulin.

- (d) Suppose the hospital finds the sets of treatments often prescribed together with insulin therapy. How would this knowledge be used by the hospital?

Answer: To build recommendation systems for complementary plans, forecast various treatment plans and drugs.. Regular alerts for medicines or treatments.

- (e) Suppose that a small number of patients provide incorrect personal information, leading to mismatches between their treatment patterns and their profiles compared to the rest. Which data mining task would be best suited to finding such patients?

Answer: *Anomaly / Outlier detection.*

Justification: Outlier detection flags records whose behavior deviates from the population and is useful for identifying mismatches or likely data errors.

Q2 Prove that the join step for generating candidate set C_{k+1} in the Apriori algorithm does not miss any frequent itemset of length $k+1$

Answer

Aim. Generation of C_{k+1} by joining L_k (pairs $p, q \in L_k$) does not leave any frequent $(k + 1)$ -itemset.

Definitions

Let k be the size used for candidate generation (we form C_{k+1} from L_k).

Let L_k be the set of frequent k -itemsets (each itemset stored in sorted order under a fixed total order).

Let C_{k+1} be the set of candidate $(k+1)$ -itemsets produced by the join step.

Let $p, q \in L_k$ be two distinct k -itemsets; write

$$p = (p_1, p_2, \dots, p_k), \quad q = (q_1, q_2, \dots, q_k),$$

each sorted.

Apriori property

If an itemset of size $k+1$ is frequent, then all its non-empty k -subsets must be frequent.

Join step

During candidate generation, the Apriori algorithm joins p and q to produce candidate $c \in C_{k+1}$ iff

$$p_1 = q_1, p_2 = q_2, \dots, p_{k-1} = q_{k-1} \quad \text{and} \quad p_k < q_k.$$

The produced candidate is

$$c = (p_1, \dots, p_{k-1}, p_k, q_k).$$

Proof

Let's prove this with a concrete example, Assume there exists a frequent 5-itemset

$$X = \{7, 3, 2, 5, 4\},$$

we keep items in sorted order so $X = (2, 3, 4, 5, 7)$. Consider the two 4-item subsets p and q as:

$$p = (2, 3, 4, 5), \quad q = (2, 3, 4, 7).$$

By the Apriori property both p and q must be in L_4 . They share the same first three items $(2, 3, 4)$ and satisfy $p_4 = 5 < 7 = q_4$, so the join rule pairs p and q and produces the candidate

$$c = (2, 3, 4, 5, 7) = X.$$

Thus any frequent 5-itemset like X will be generated by joining appropriate members of L_4 .

General case (formal):

Let X be any frequent $(k+1)$ -itemset. Write X in sorted order

$$X = (i_1, i_2, \dots, i_k, i_{k+1}), \quad i_1 < i_2 < \dots < i_{k+1}.$$

Consider the two k -subsets

$$p = (i_1, i_2, \dots, i_k), \quad q = (i_1, i_2, \dots, i_{k-1}, i_{k+1}).$$

By Apriori both p and q belong to L_k . They share the same first $k - 1$ items and $p_k = i_k < i_{k+1} = q_k$, so the join rule pairs them and produces

$$c = (i_1, \dots, i_{k-1}, i_k, i_{k+1}) = X.$$

Since X was arbitrary, every frequent $(k + 1)$ -itemset is produced by the join step (and pruning cannot remove a true frequent X because all its k -subsets are in L_k). \square

Q3 A transaction dataset contains ten transactions as shown below.

Let min-sup=30% and min-conf=50%

- a) Using FP-growth, find all frequent itemsets that contain item m. Show the steps.
- b) Find all the strong association rules whose antecedent is m.
- c) Are there any misleading rules in the result of (b)? If yes, identify them and explain why they are misleading.

Answer

Transactional Database

TID	Items bought
1	m, n, p, q
2	n, o, p
3	m, n, p, q
4	m, o, p, q
5	n, o, p, q
6	n, p, q
7	o, p
8	m, n, o
9	m, p, q
10	n, p

Header Table (sorted by descending frequency)

Items	Count
p	9
n	7
q	6
m	5
o	5

Header Table for m-conditional Database(sorted)

Items	Count
pnq	2
pq	2
nm	1

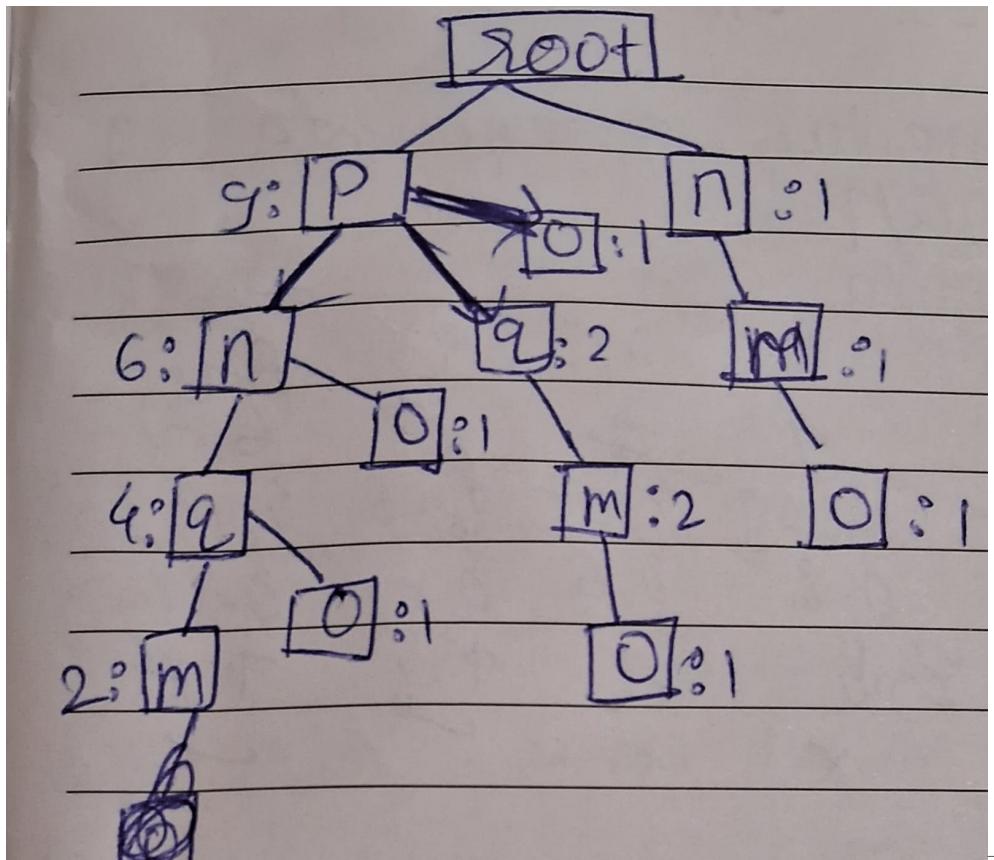


Fig 3.1 FP-tree image

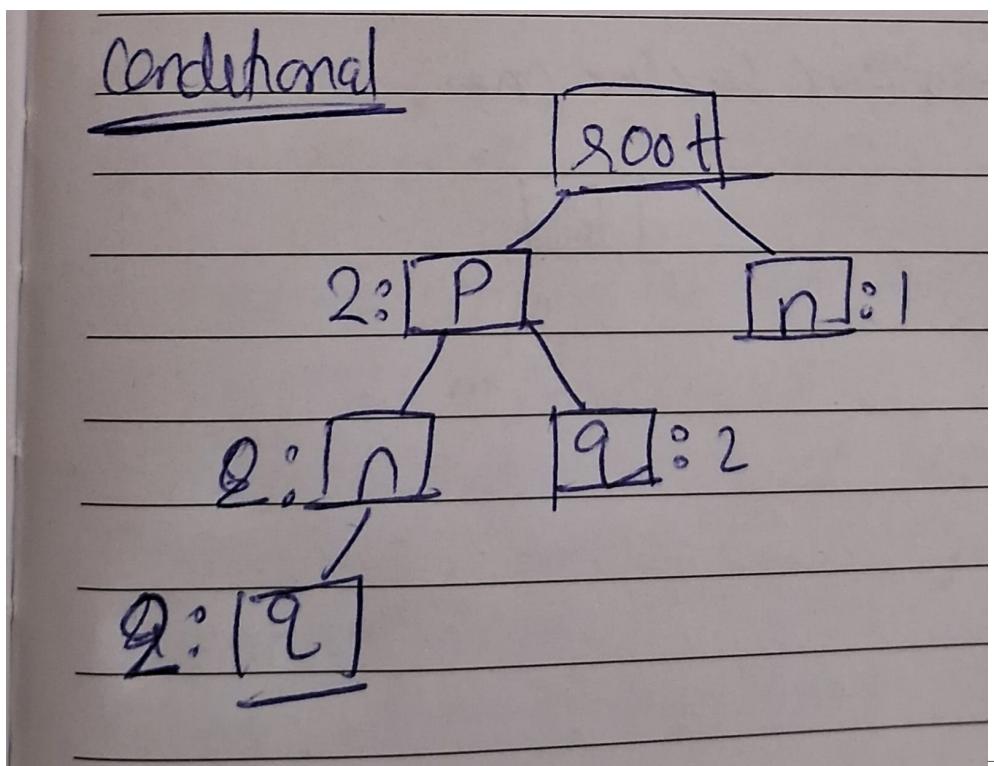


Fig 3.2 Conditional FP-tree image

Part (a)

Frequent itemsets that contain item **m**, the m-conditional pattern base are {p,n,q}:2, {p,q}:2 and {n}:1.

conditional counts (given m) : $p : 4, q : 4, n : 3$.

So the frequent itemsets that include **m** (support = count / 10) are:

$$\{m\} : 5, \{p, m\} : 4, \{q, m\} : 4, \{m, n\} : 3, \{p, q, m\} : 4.$$

(Other supersets such as {p,n,m} or {n,q,m} or {p,n,m,q} had counts < 3 and are not frequent.)

Part (b)

The support of m is 5 and confidences are:

$$\begin{aligned} \text{conf}(m \rightarrow p) &= \frac{\text{supp}(\{m, p\})}{\text{supp}(\{m\})} = \frac{4}{5} = 0.80, \\ \text{conf}(m \rightarrow q) &= \frac{4}{5} = 0.80, \\ \text{conf}(m \rightarrow n) &= \frac{3}{5} = 0.60, \\ \text{conf}(m \rightarrow \{p, q\}) &= \frac{4}{5} = 0.80. \end{aligned}$$

	$\{m\}$	$\{p, m\}$	$\{q, m\}$	$\{n, m\}$	$\{p, q, m\}$
Support (count)	5	4	4	3	4
Confidence	$5/5 = 1.00$	$4/5 = 0.80$	$4/5 = 0.80$	$3/5 = 0.60$	$4/5 = 0.80$
Strong ($\text{conf} \geq 0.5$)?	—	Yes	Yes	Yes	Yes

All four rules have confidence ≥ 0.50 , so they are *strong* by the given threshold.

Part (c)

A rule can be misleading when the consequent is very common overall, confidence alone can be high, yet the rule is not informative. We compute *lift*:

$$\text{lift}(m \rightarrow X) = \frac{\text{conf}(m \rightarrow X)}{P(X)} = \frac{\text{supp}(\{m\} \cup X)/\text{supp}(\{m\})}{\text{supp}(X)/N}.$$

Use marginal supports (from entire dataset of $N = 10$):

$$P(p) = 9/10 = 0.90, P(q) = 6/10 = 0.60, P(n) = 7/10 = 0.70, P(p \wedge q) = 6/10 = 0.60.$$

Calculate lift and mark misleading rules (lift < 1 means antecedent reduces chance of consequent):

	$\{m\}$	$\{p, m\}$	$\{q, m\}$	$\{n, m\}$	$\{p, q, m\}$
Support (count)	5	4	4	3	4
Marginal support	—	$P(p) = 0.90$	$P(q) = 0.60$	$P(n) = 0.70$	$P(p \wedge q) = 0.60$
Confidence	1.00	0.80	0.80	0.60	0.80
Lift	—	$0.80/0.90 \approx 0.889$	$0.80/0.60 \approx 1.333$	$0.60/0.70 \approx 0.857$	$0.80/0.60 \approx 1.333$
Correct? ($\text{lift} > 1$)	—	No	Yes	No	Yes

Conclusion : Rules $m \rightarrow p$ and $m \rightarrow n$ are *misleading* because their lift < 1 (the consequents p and n are already very common; knowing m lowers their probability). The two informative rules are:

$$m \rightarrow q \quad (\text{conf} = 0.80, \text{lift } \approx 1.33), \quad m \rightarrow \{p, q\} \quad (\text{conf} = 0.80, \text{lift } \approx 1.33).$$

Q4 The FP-growth algorithm has the drawback that it can generate a stack of conditional FP-trees during its recursive process and these conditional FP-trees contain redundant information, which is a waste of memory space. How can you modify the FP-growth algorithm so that no conditional FP-trees need to be generated? Note that you can't change the FP-tree to a totally different data structure. The modification should be based on the FP-tree structure and each tree node can only store one count (as in the original FP-tree structure). But you are allowed to have multiple header-tables (of the same structure as the original header-table where each row contains an item, a count, and a pointer) for the FP-tree.

Answer:

Thanks for all the direct and indirect hints professor, Have tried to understand them and think wisely. As we have been given in question:

- the FP-tree node format should be unchanged (each node has a single count).
- We are allowed multiple header-tables with same data types/structure, instead of building multiple new trees.
- When constructing conditional pattern bases, header-tables should point to nodes in the existing tree (because we do not copy nodes). Additionally, once globally the ordering of items against min-support is fixed, we don't change them while recursion to maintain fix order.

As we know FP-growth normally builds a conditional FP-tree to compress the Database to conditional base to enable fewer and small scale scans. The conditional pattern base has *contains* all information needed (it has collection of prefix paths with occurrences/frequencies). Instead of converting that base into a tree structure, we can work directly with the base: aggregate counts into a small header table (items \rightarrow conditional counts), prune infrequent items just like earlier we used to prune links/nodes, and add extensions by filtering the prefix-path list. Because we extract prefix paths from the *global* FP-tree via node-links, we never copy or create tree nodes as we only create small lists and header tables. By keeping global item order fixed throughout, we can expect the algorithm to enumerate patterns in a constant order to allow specific pruning.

Pseudo-Algorithm

1. Build the main and only *global* FP-tree T from our main dataset and its global header table HT_{global} . Fix the order of items in it with descending support and do not change it.
2. To mine frequent patterns with prefix α , iterate items i in HT_{global} (in the chosen order). For each i :
 - (a) Extract CPB_i (the conditional pattern base for i) by following the node-link chain for item i in the global tree T . Each occurrence yields a prefix path (items above that node) and the node count.
 - (b) From CPB_i , compute a small conditional header table HT_i by summing counts per item (maintain global order). Prune entries with count $< \text{min_sup}$.
 - (c) If HT_i is empty, output $\alpha \cup \{i\}$ (it is frequent) and continue. Otherwise, output $\alpha \cup \{i\}$ and *recursively* enumerate extensions of $\alpha \cup \{i\}$ using (CPB_i, HT_i) but **do not** build a conditional FP-tree.
 - (d) The recursion step works by filtering CPB_i : for a chosen extension $j \in HT_i$, keep only those prefix paths in CPB_i that contain j , shorten each such path to the portion before j (preserving global order), aggregate counts to make HT_{ij} , prune, and recurse if non-empty.
3. Use stack discipline: free temporary prefix-path lists (CPBs) after returning from a recursion branch so memory stays small.

Acknowledgements

- I used the course slides from eclass along with web articles,papers to prepare and justify the answers.
- **LLM assistance:** I used ChatGPT (model: **GPT-5 Thinking mini**) to draft and refine the LaTeX formatting and wording of this document.

Prompt summary: 1 “Refine my LaTeX file: remove date, set margins to 0.6in, adjust font sizes for title/sections/answers, keep answers unchanged, add page numbers and add references to uploaded course slides and Apriori and other papers, and add LLM acknowledgement.”

Prompt summary: 1 “ Help refine my pseudo-algorithm and LaTeX phrasing”

(Algorithm refining also changed a bit of my thinking and brought me closer to the implementation)

Date/time of use: 2025-09-30 (America/Toronto timezone).

References

- [1] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules,” Proc. of the 20th Int. Conf. on Very Large Data Bases (VLDB), Santiago, Chile, 1994.
- [2] J. Han, J. Pei, and Y. Yin, “Mining Frequent Patterns without Candidate Generation,” SIGMOD 2000. (FP-growth original paper)