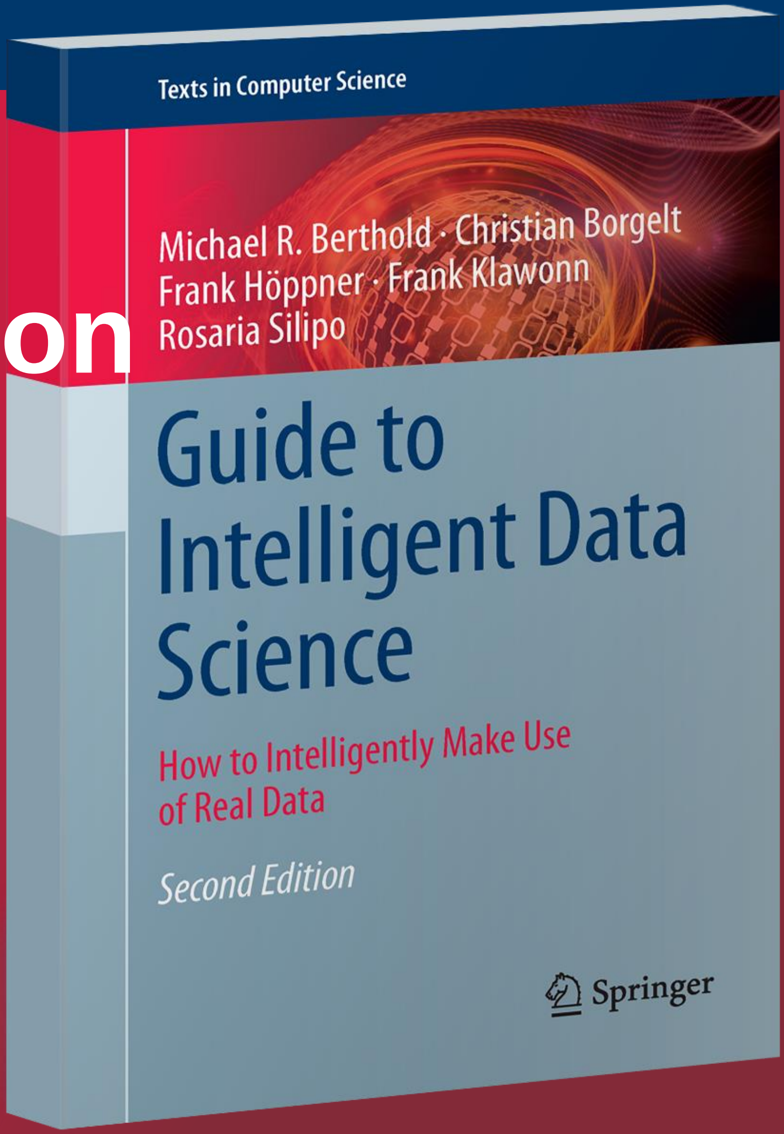
Recommendation Engines



Summary of this lesson

“We all make choices, but in the end our choices make us”
-Ken Levine

Are there events that always happen together?

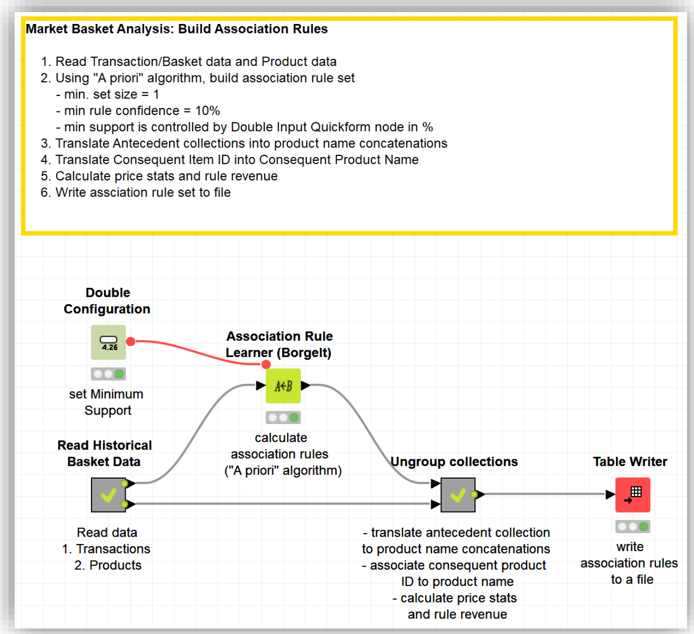
**This lesson refers to chapter 7 of the GIDS book*

Content of this lesson

- Association Rules
- Itemset Mining
- Generating Association Rules
- Collaborative Filtering

Datasets

- Datasets used : transaction data & products data
- Example Workflow:
  - „Association_Rules_for_MarketBasketAnalysis“ https://kni.me/w/fQ9yZLztzEUmAsQ0



Association Rules

Overview

- Association Rules: Motivation
- Item Set Mining
  - Breadth First Searching: The Apriori Algorithm
  - Depth First Searches: The Eclat Algorithm
  - (Compact) Representation of Itemsets
- Finding Association Rules

Association rule induction

- Originally designed for **market basket analysis**.
- Aims at finding patterns in the shopping behavior of customers of supermarkets, mail-order companies, on-line shops etc.

More specifically:

- **Find sets of products that are frequently bought together**.
- Example of an association rule:
  - *IF a customer buys bread and wine,*
  - *THEN she/he will probably also buy cheese.*

Association Rule: Example

IF

+

THEN

Antecedent

Consequent

From the analysis of many shopping baskets ...

A-priori algorithm

Recommendation

IF

+

THEN

Possible applications of found association rules:

- Improve arrangement of products in shelves, on a catalog’s pages.
- Support of cross-selling (suggestion of other products), product bundling.
- Fraud detection, technical dependence analysis.
- Finding business rules and detection of data quality problems.
- . . .

- Two step implementation:

- Find the **frequent item sets** (also called large item sets), i.e., the item sets that have at least a user-defined **minimum support**.

- Form rules using the frequent item sets found and select those that have at least a user-defined **minimum confidence**.

Assessing the quality of association rules:

Support of an item set:

- Fraction of transactions (shopping baskets/carts) that contain the item set.

Support of an association rule $X \rightarrow Y$:

- Either: Support of $X \cup Y$ (more common: rule is correct)
- Or: Support of X (more plausible: rule is applicable)

Confidence of an association rule $X \rightarrow Y$:

- Support of $X \cup Y$ divided by support of X (estimate of $P(Y \mid X)$).

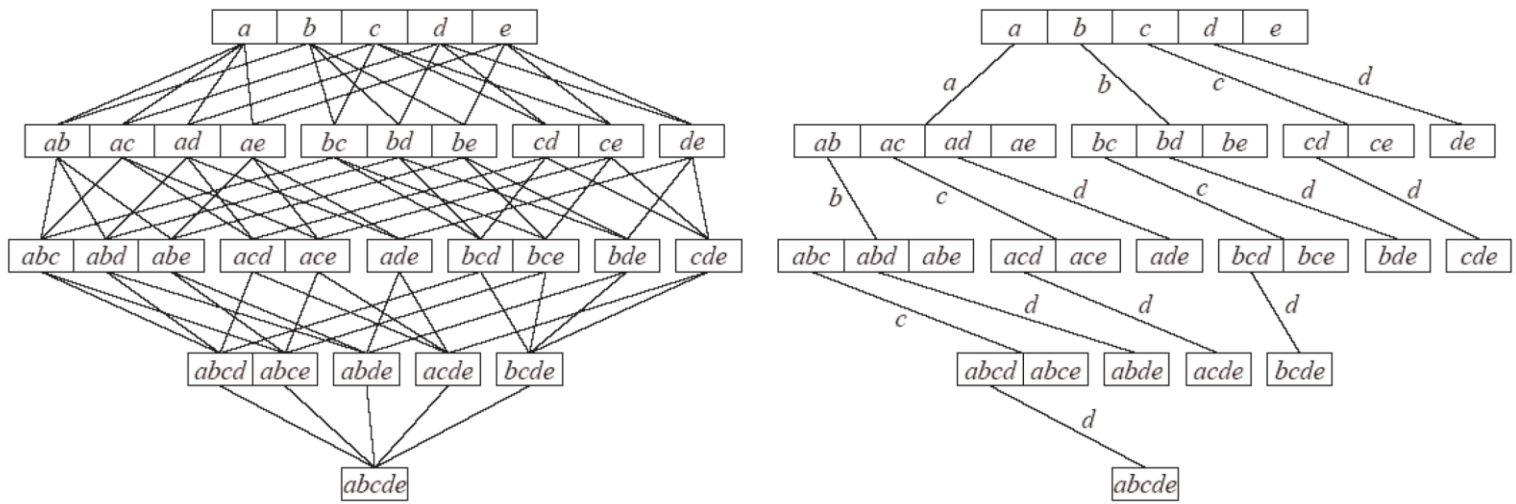
Itemset Mining

N shopping baskets

Search for
$\boxed{\text{frequent itemsets}}$

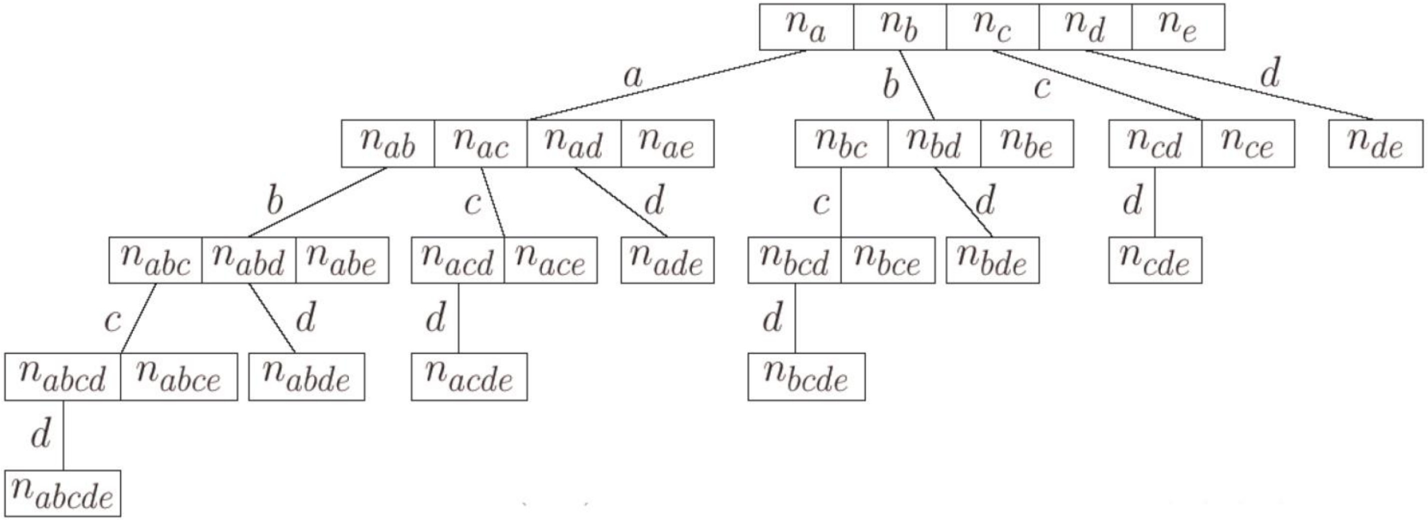
{A, B, F, H}
{A, B, C}
{B, C, H}
{D, E , F}
{D, E}
{A, B}
{A, C}
{H, F}
...

- Subset lattice and a prefix tree for five items:



- It is not possible to determine the support of all possible item sets, because their number grows exponentially with the number of items.
- Efficient methods to search the subset lattice are needed.

Item Set Trees

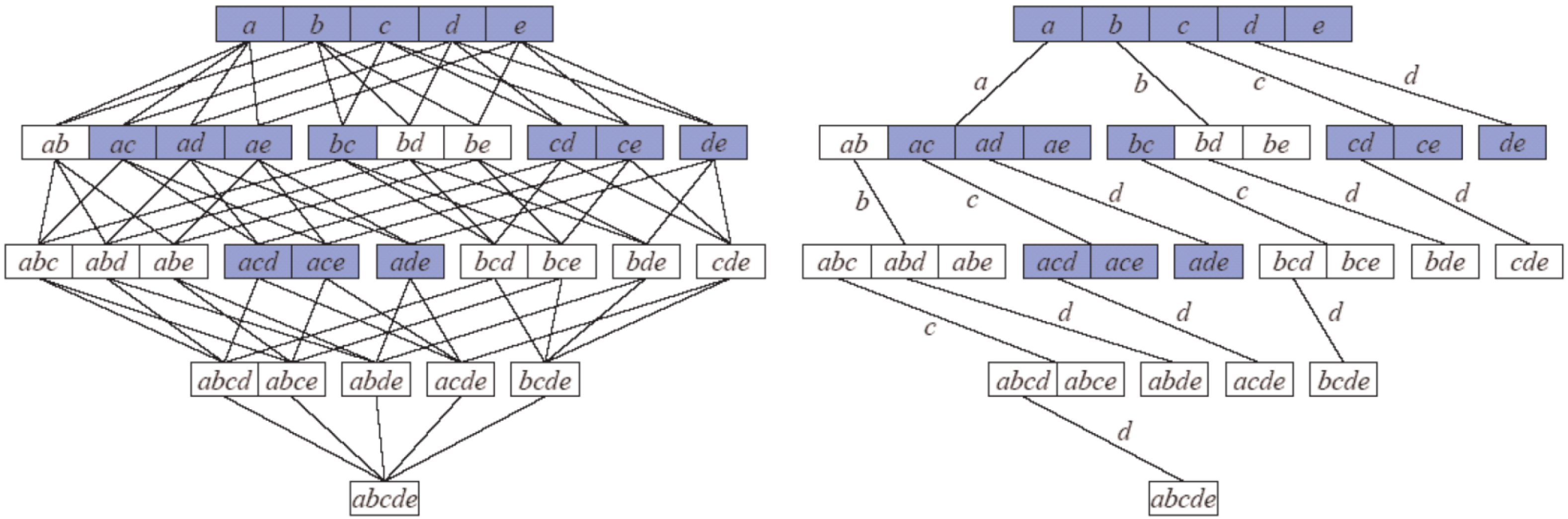


- A (full) item set tree for the five items *a, b, c, d,* and *e*.
- Based on a global order of the items.
- The item sets counted in a node consist of all items labeling the edges to the node (common prefix) and one item following the last edge label.

- In applications item set trees tend to get very large, so pruning is needed.

- **Structural Pruning:**
  - Make sure that there is only one counter for each possible item set.
  - Explains the unbalanced structure of the full item set tree.
- **Size Based Pruning:**
  - Prune the tree if a certain depth (a certain size of the item sets) is reached.
  - Idea: Rules with too many items are difficult to interpret.
- **Support Based Pruning:**
  - **No superset of an infrequent item set can be frequent.**
  - No counters for item sets having an infrequent subset are needed

- **Boundary** between frequent (blue) and infrequent (white) item sets:



- **Apriori**: Breadth-first search (item sets of same size).
- **Eclat**: Depth-first search (item sets with same prefix).

Apriori Breadth first Search

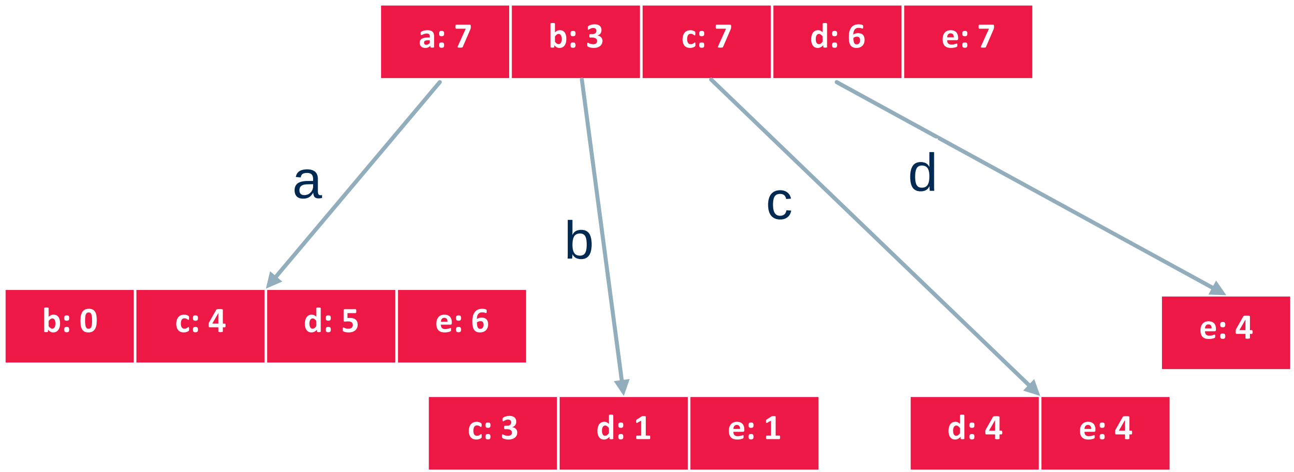
1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}

a: 7	b: 3	c: 7	d: 6	e: 7

- Example transaction database with 5 items { a,b,c,d,e } and 10 transactions.
- Minimum support: 30%, i.e., at least 3 transactions must contain the item set.
- All one item sets are frequent ➔ full second level is needed.

Apriori Breadth first Search

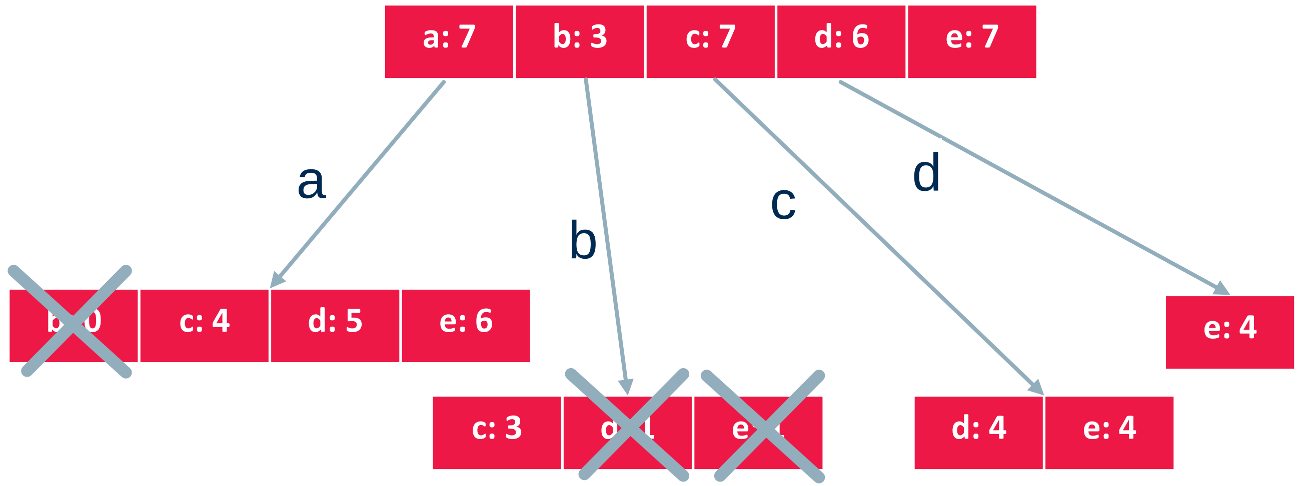
1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



- Determining the support of item sets: For each item set traverse the database and count the transactions that contain it (highly inefficient).
- Better: Traverse the tree for each transaction and find the item sets it contains (efficient: can be implemented as a simple double recursive procedure).

Apriori Breadth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



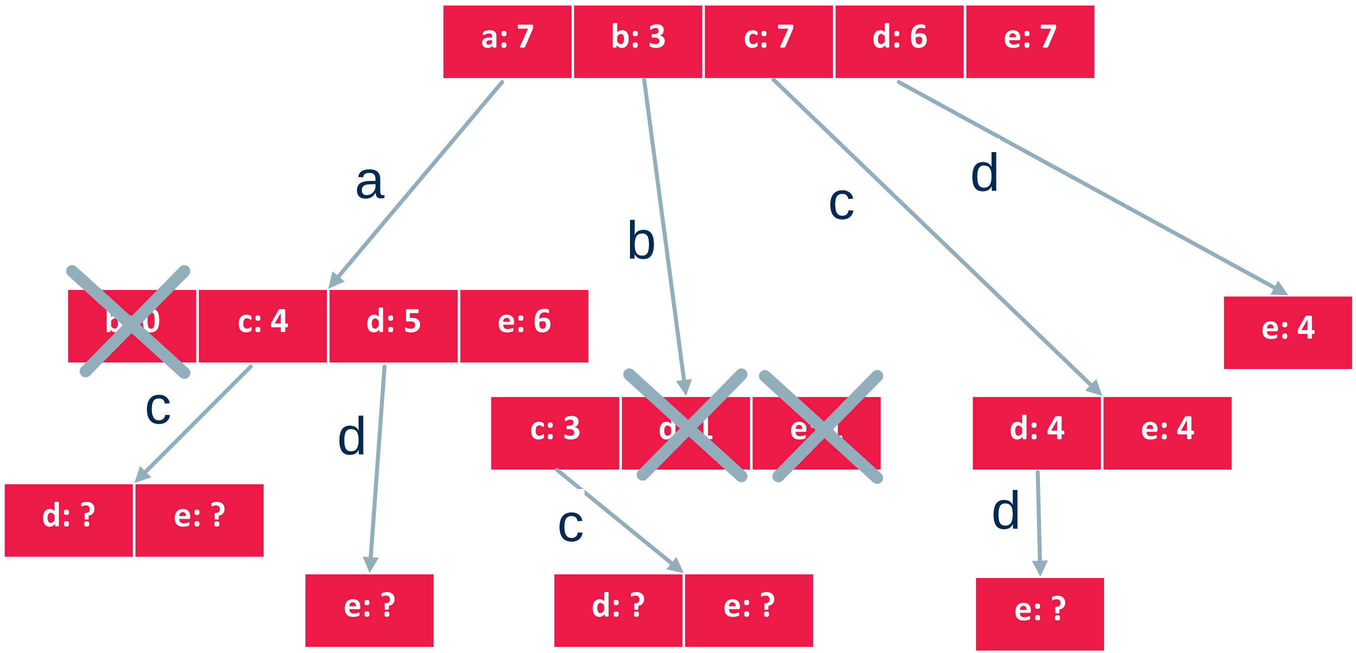
– Minimum support: 30%, i.e., at least 3 transactions must contain the item set.

– Infrequent item sets: {*a*, *b*}, {*b*, *d*}, {*b*, *e*}.

– The subtrees starting at these item sets can be pruned.

Apriori Breadth first Search

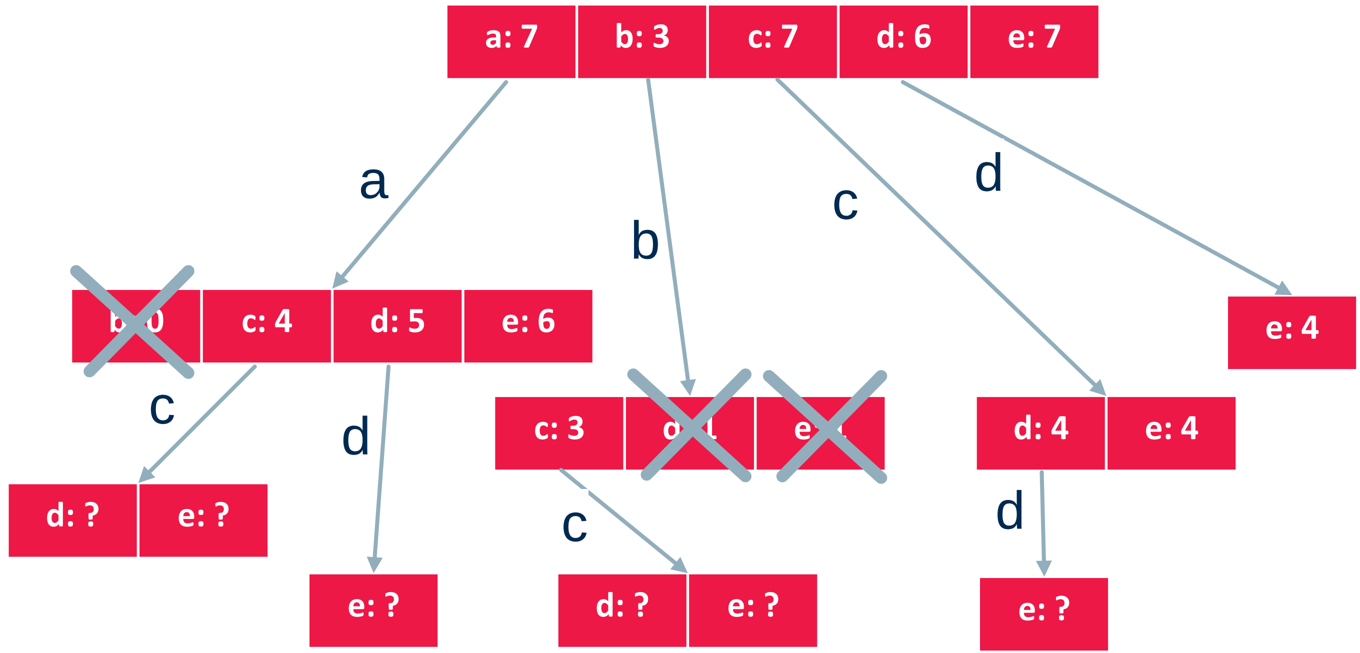
1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



– Generate candidate item sets with 3 items (parents must be frequent).

Apriori Breadth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



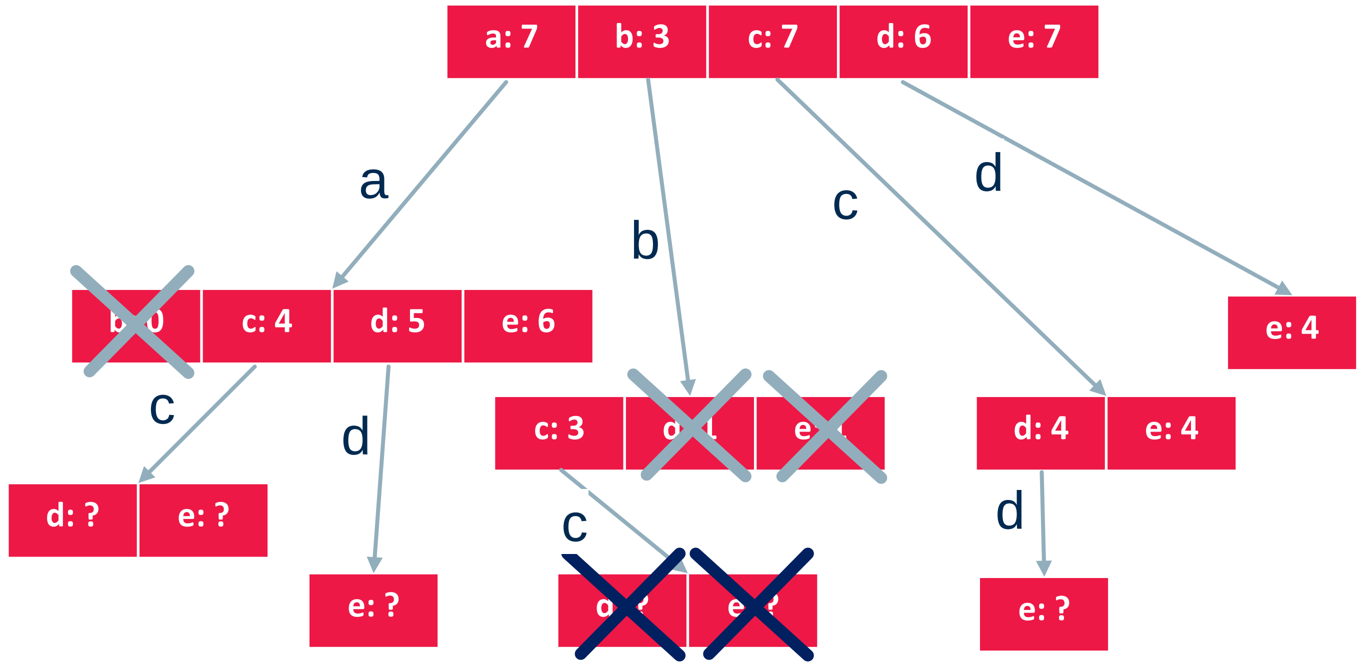
– Before counting, check whether the candidates contain an infrequent item set.

– An item set with *k* items has *k* subsets of size *k* − 1.

– The parent is only one of these subsets

Apriori Breadth first Search

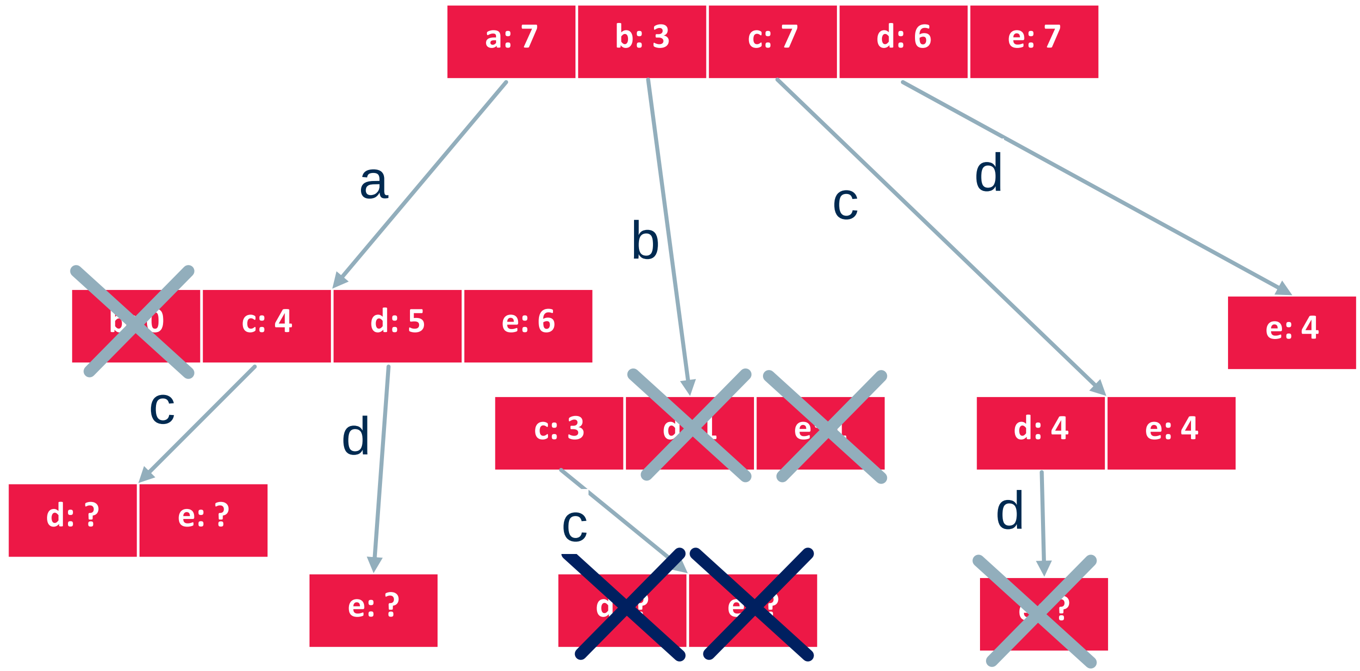
1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



– The item sets {*b*, *c*, *d*} and {*b*, *c*, *e*} can be pruned, because
  – {*b*, *c*, *d*} contains the infrequent item set {*b*, *d*} and
  – {*b*, *c*, *e*} contains the infrequent item set {*b*, *e*}.
– Only the remaining four item sets of size 3 are evaluated.

Apriori Breadth first Search

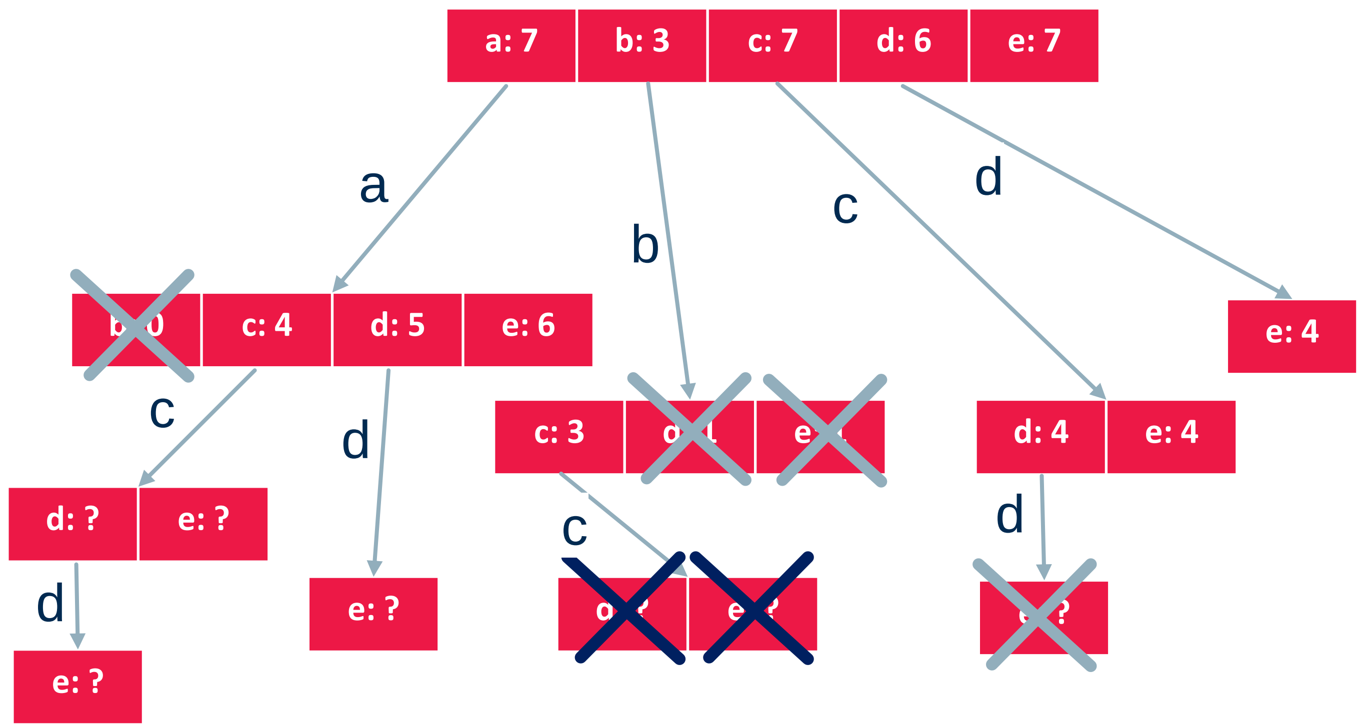
1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



– Minimum support: 30%, i.e., at least 3 transactions must contain the item set.

– Infrequent item set: {*c*, *d*, *e*}.

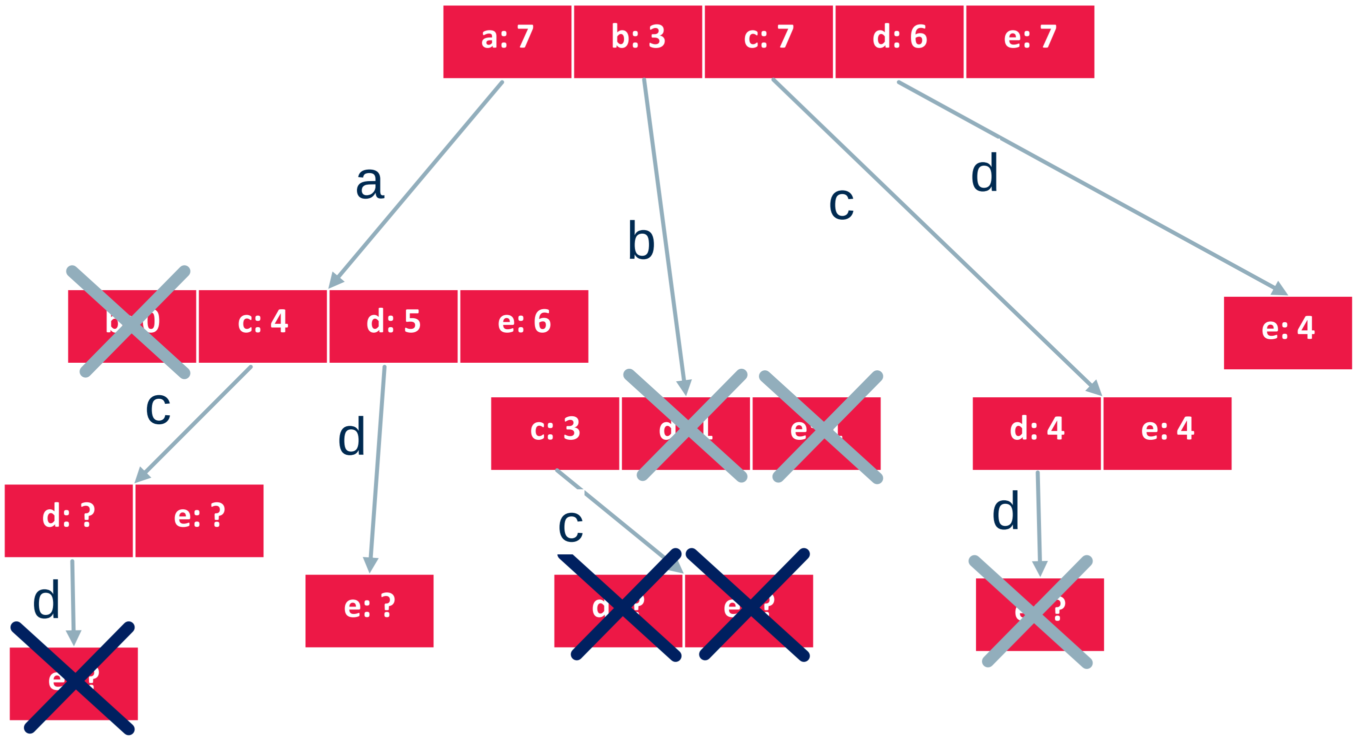
1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



– Generate candidate item sets with 4 items (parents must be frequent).
– Before counting, check whether the candidates contain an infrequent item set.

Apriori Breadth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



– The item set {*a*,*c*,*d*,*e*} can be pruned, because it contains the infrequent item set {*c*, *d*, *e*}.
– Consequence: No candidate item sets with four items. Stop.

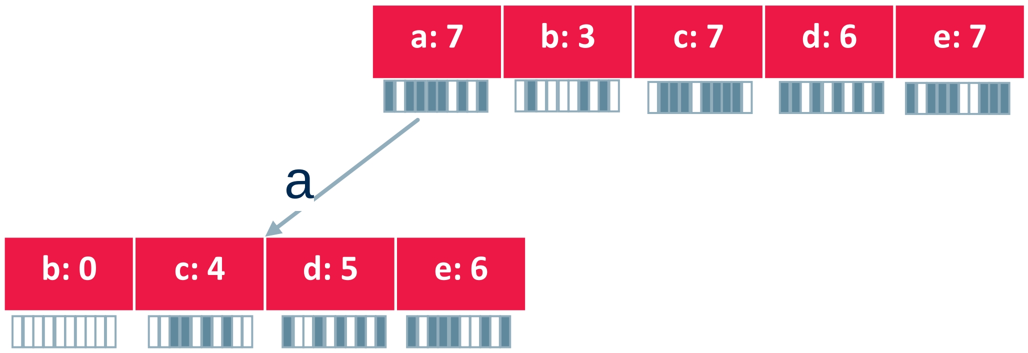
Eclat Depth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}

– Form a transaction list for each item. Here: bit vector representation.
  – grey: item is contained in transaction
  – white: item is not contained in transaction
– Transaction database is needed only once (for the single item transaction lists).

Eclat Depth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



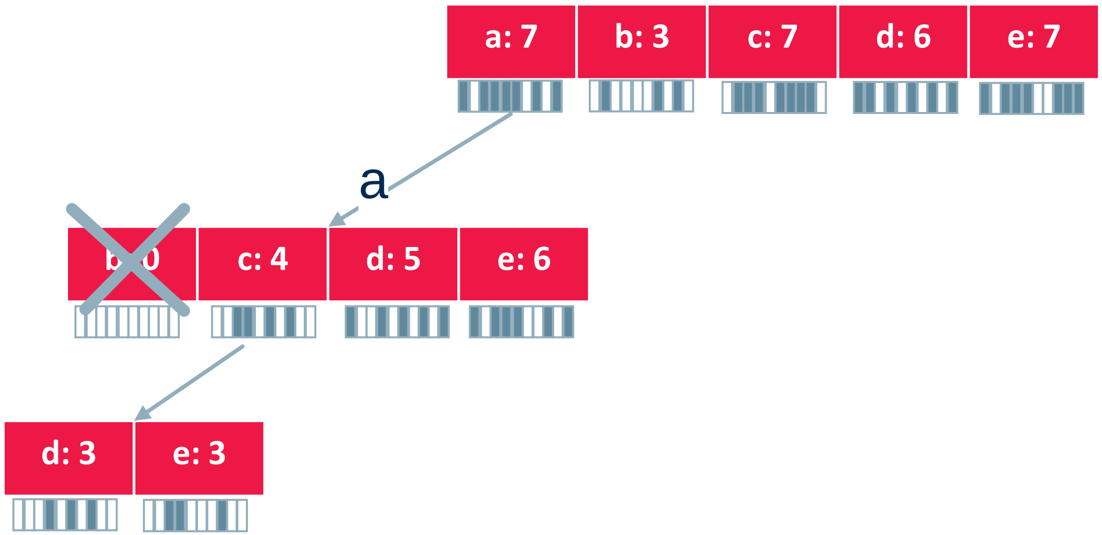
– Intersect the transaction list for item *a* with the transaction lists of all other items.

– Count the number of set bits (containing transactions).

– The item set {*a*, *b*} is infrequent and can be pruned.

Eclat Depth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}

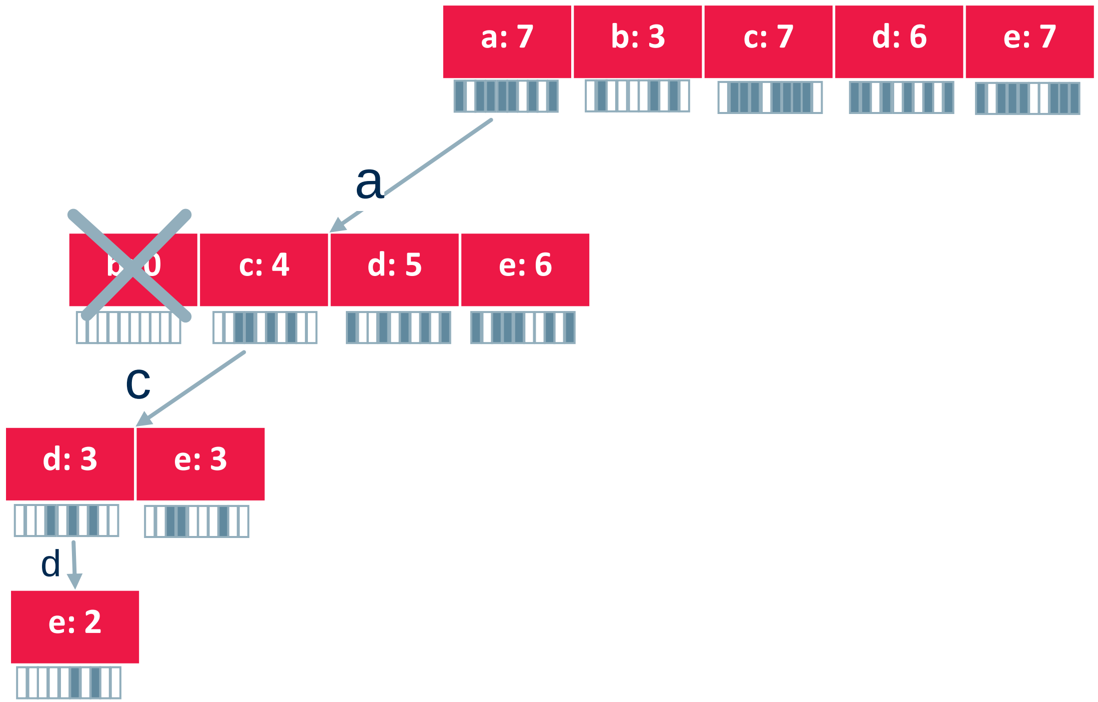


– Intersect the transaction list for {*a*, *c*} with the transaction lists of {*a*, *x*}, $x \in$ {*d*, *e*}.

– Result: Transaction lists for the item sets {*a*, *c*, *d*} and {*a*, *c*, *e*}.

Eclat Depth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



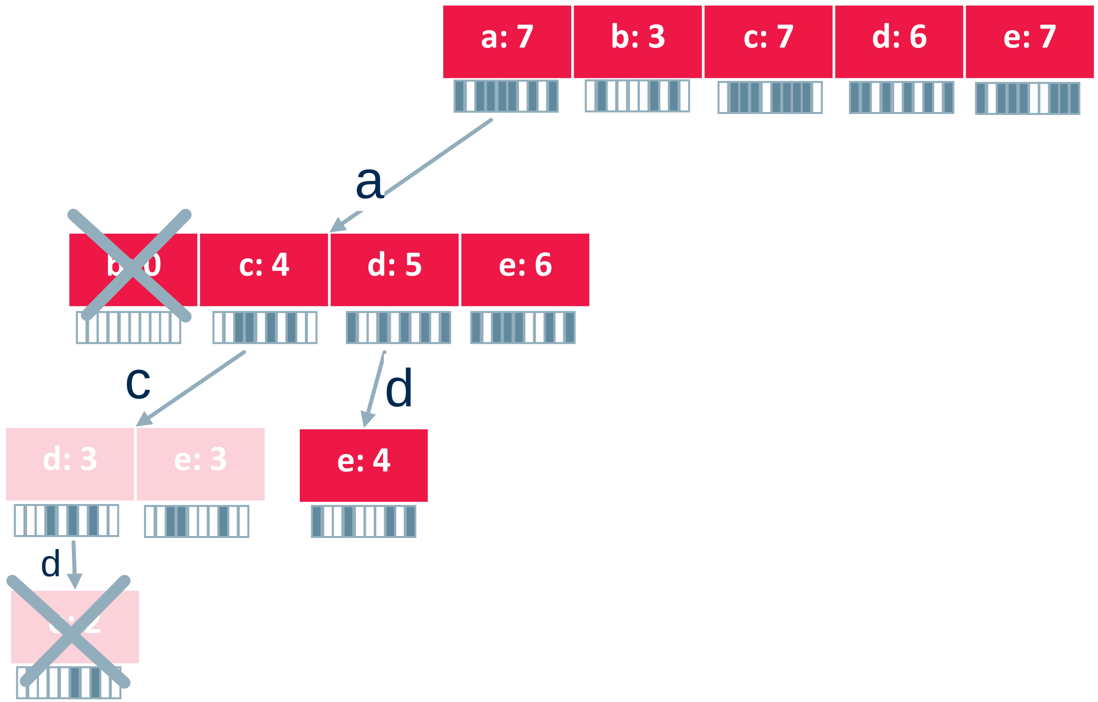
– Intersect the transaction list for {*a*, *c*, *d*} and {*a*, *c*, *e*}.

– Result: Transaction list for the item set {*a*, *c*, *d*, *e*}.

– With Apriori this item set could be pruned before counting, because it was known that {*c*, *d*, *e*} is infrequent.

Eclat Depth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}

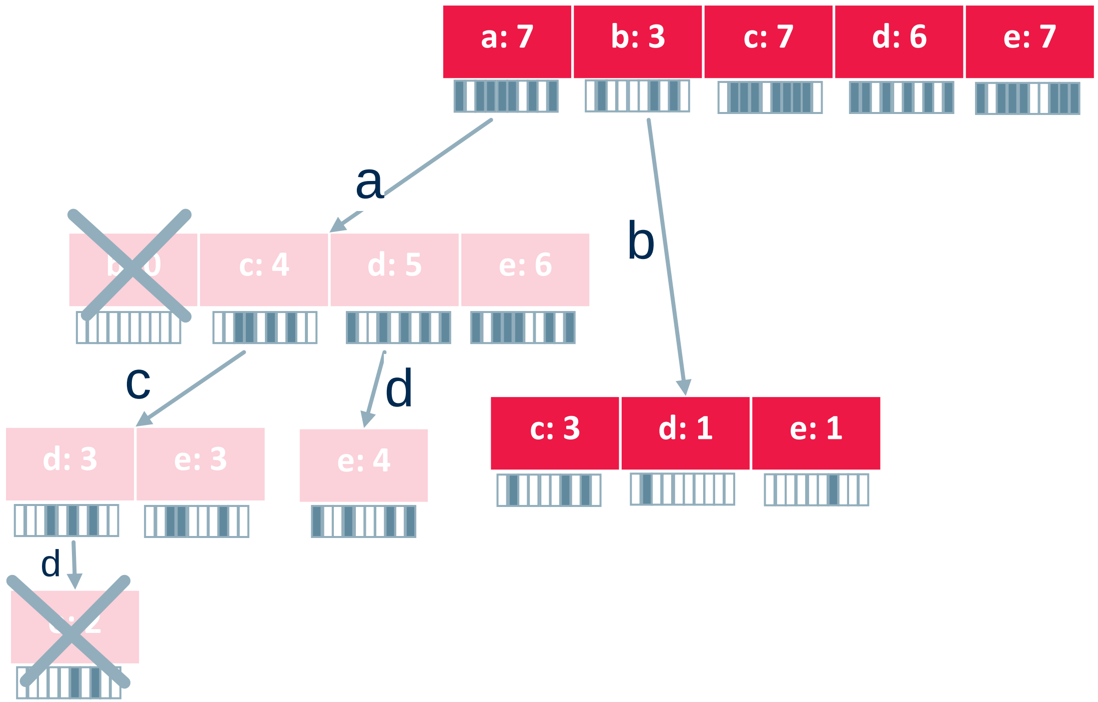


– Backtrack to the second level of the search tree and intersect the transaction list for {*a*, *d*} and {*a*, *e*}.

– Result: Transaction list for {*a*, *d*, *e*}.

Eclat Depth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



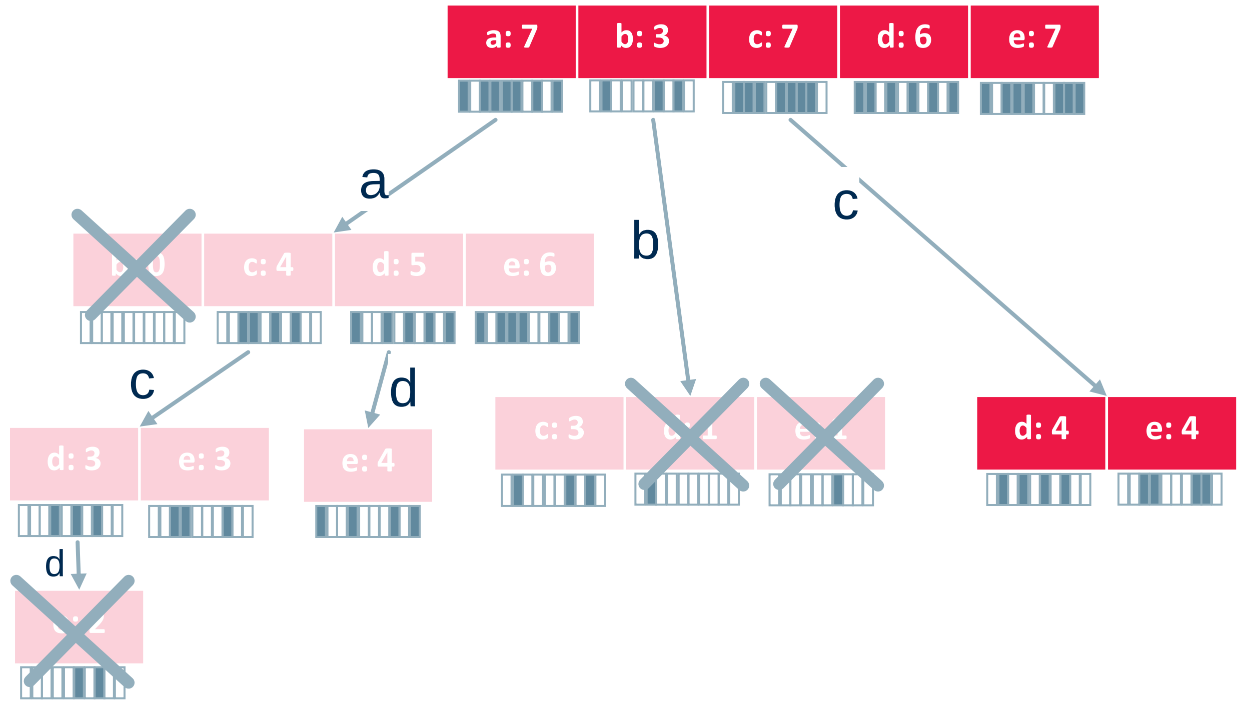
– Backtrack to the first level of the search tree and intersect the transaction list for *b* with the transaction lists for *c*, *d*, and *e*.

– Result: Transaction lists for the item sets {*b*, *c*}, {*b*, *d*}, and {*b*, *e*}.

– Only one item set with sufficient support -> prune all subtrees.

Eclat Depth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}

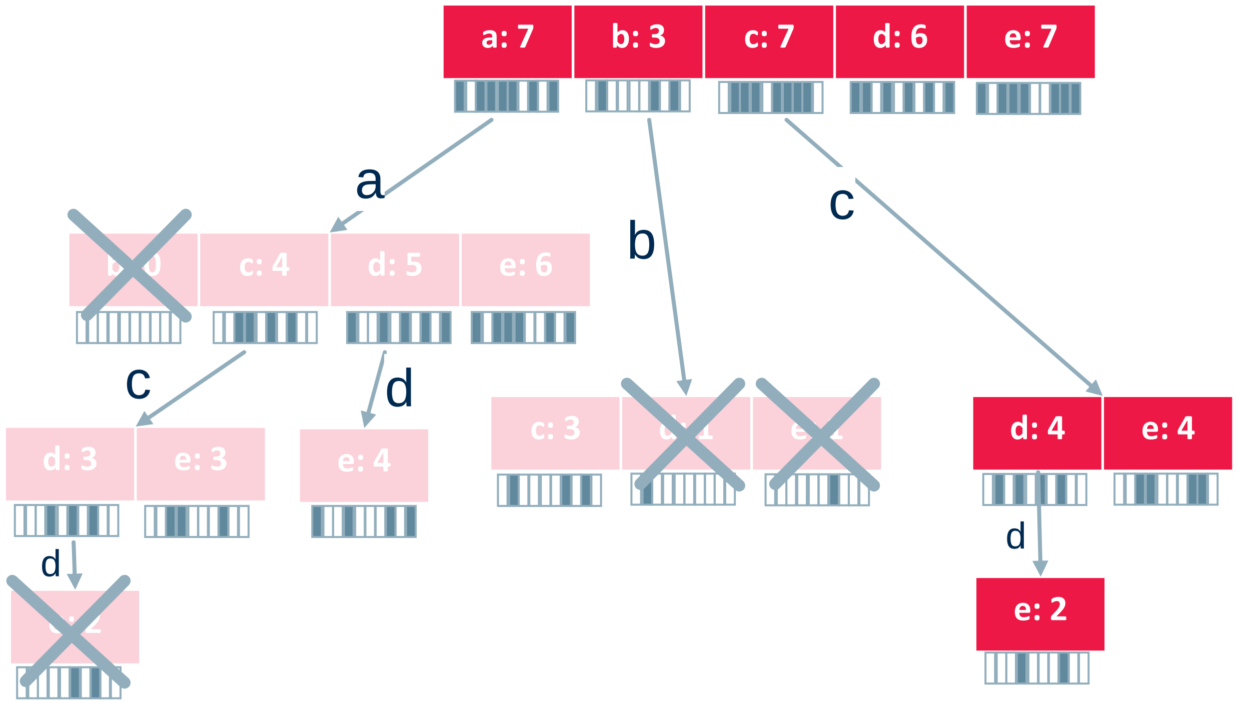


– Backtrack to the first level of the search tree and intersect the transaction list for *c* with the transaction lists for *d* and *e*.

– Result: Transaction lists for the item sets {*c*, *d*} and {*c*, *e*}.

Eclat Depth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



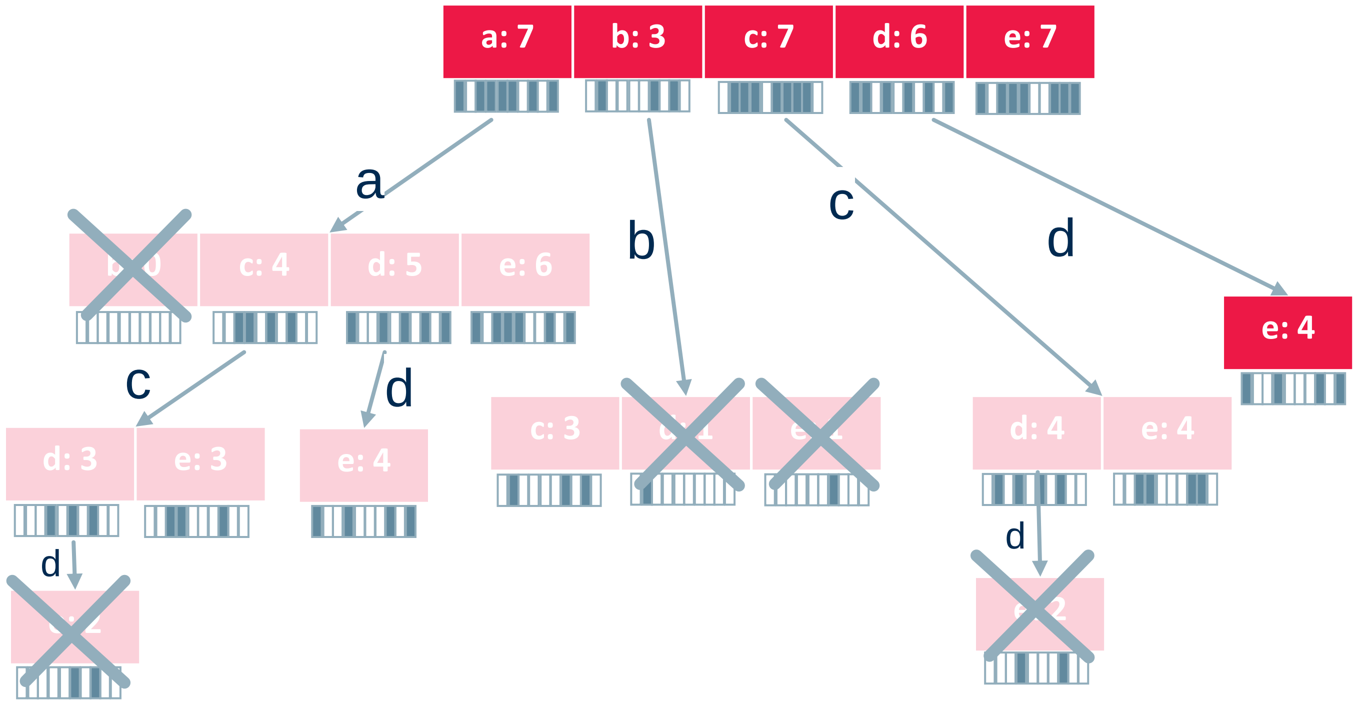
– Intersect the transaction list for {*c*, *d*} and {*c*, *e*}.

– Result: Transaction list for {*c*, *d*, *e*}.

– Infrequent item set: {*c*, *d*, *e*}.

Eclat Depth first Search

1. {*a*, *d*, *e*}
2. {*b*, *c*, *d*}
3. {*a*, *c*, *e*}
4. {*a*, *c*, *d*, *e*}
5. {*a*, *e*}
6. {*a*, *c*, *d*}
7. {*b*, *c*}
8. {*a*, *c*, *d*, *e*}
9. {*c*, *b*, *e*}
10. {*a*, *d*, *e*}



– Backtrack to the first level of the search tree and intersect the transaction list for *d* with the transaction list for *e*.

– Result: Transaction list for the item set {*d*, *e*}.

– With this step the search is finished.

Frequent Item Sets

1 item	2 items		3 items
$\{a\}^{+}: 70\%$	$\{a, c\}^{+}: 40\%$	$\{c, e\}^{+}: 40\%$	$\{a, c, d\}^{+*}: 30\%$
$\{b\} \quad : 30\%$	$\{a, d\}^{+}: 50\%$	$\{d, e\}^{+}: 40\%$	$\{a, c, e\}^{+*}: 30\%$
$\{c\}^{+}: 70\%$	$\{a, e\}^{+}: 60\%$		$\{a, d, e\}^{+*}: 40\%$
$\{d\}^{+}: 60\%$	$\{b, c\}^{+*}: 30\%$		
$\{e\}^{+}: 70\%$	$\{c, d\}^{+}: 40\%$		

- **Types of frequent item sets**
- **Free Item Set**: Any frequent item set (support is higher than the minimal support).
- **Closed Item Set** (marked with +): A frequent item set is called **closed** if no superset has the same support.
- **Maximal Item Set** (marked with): A frequent item set is called **maximal** if no superset is frequent.

Generating Association Rules

- For each frequent item set *S*:

- Consider all pairs of sets $X, Y \in S$ with $X \cup Y = S$ and $X \cap Y = \emptyset$.
  Common restriction: $|Y| = 1$, i.e. only one item in consequent

- X = antecedent, Y = consequent

- Form the association rule $X \rightarrow Y$ and compute its confidence.

$$conf(X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} = \frac{supp(S)}{supp(X)}$$

- Report rules with a confidence higher than the minimum confidence.

From “Frequent Itemsets“ to “Rules“

{A, B, F, H} →

- {A, B, F} → H
- {A, B, H} → F
- {A, F, H} → B
- {B, F, H} → A

Which rules shall I choose?

Support, Confidence, and Lift

$$\{A, B, F\} \rightarrow H$$

- Item set support $\boldsymbol{s} = \dfrac{freq(A,B,F,H)}{N}$ ⬅ How often these items are found together

- Rule confidence $\boldsymbol{c} = \dfrac{freq(A,B,F,H)}{freq(A,B,F)}$ ⬅ How often the antecedent is together with the consequent

- Rule lift $= \dfrac{support\ (\{A,B,F\} \Rightarrow H)}{support\ (A,B,F) \times support(H)}$ ⬅ How often antecedent and consequent happen together compared with random chance

The rules with support, confidence and lift above a threshold → most reliable ones

Association Rule Mining (ARM): Two Phases

Discover all frequent and strong association rules

X ⇒ Y → "if X then Y"

with sufficient support and confidence

Two phases:

1. find all frequent itemsets (FI) ← Most of the complexity
   – Select itemsets with a minimum support

$$FI = \{\{X,Y\}, X,Y \subset I \mid s(X,Y) \geq S_{min}\}$$

2. build strong association rules
   – Select rules with a minimum confidence:

$$Rules: \{X \Rightarrow Y, X,Y \subset FI, \mid c(X \Rightarrow Y) \geq C_{min}\}$$

User parameters

Generating Association Rules: Example

- **Example:** $S = \{a, c, e\}, X = \{c, e\}, Y = \{a\}$.

$$conf(c, e \rightarrow a) = \frac{supp(\{a, c, e\})}{supp(\{c, e\})} = \frac{30\%}{40\%} = 75\%$$

- **Minimum confidence: 80%**

Association Rule	Support of all items	Support of antecedent	confidence
$b \rightarrow c$	30%	30%	100%
$d \rightarrow a$	50%	60%	83.3%
$e \rightarrow a$	60%	70%	85.7%
$a \rightarrow e$	60%	70%	85.7%
$d, e \rightarrow a$	40%	40%	100%
$a, d \rightarrow e$	40%	50%	80%

A-Priori Algorithm: Example

- Let's consider milk, diaper, and beer: $\{milk, diaper\} \Rightarrow beer$

- How often are they found together across all shopping baskets?
- How often are they found together across all shopping baskets containing the antecedents?

TID	Transactions
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

support

$$s(milk, diaper, beer) = \frac{P(milk, diaper, beer)}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{P(milk, diaper, beer)}{P(milk, diaper)} = \frac{2}{3} = 0.67$$

confidence

A-priori algorithm: an example

- Let‘s consider milk, diaper, and beer: $\{milk, diaper\} \Rightarrow beer$
- How often are they found together across all shooping baskets?
- How often are they found together across all shopping baskets containing the antecedents?

TID	Transactions
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

$$s(milk, diaper) = \frac{P(milk, diaper)}{|T|} = \frac{3}{5} = 0.6$$

$$s(beer) = \frac{P(beer)}{|T|} = \frac{3}{5} = 0.6$$

$$Rule\ lift = \frac{s(milk, diaper, beer)}{s(milk, diaper) \times s(beer)}$$

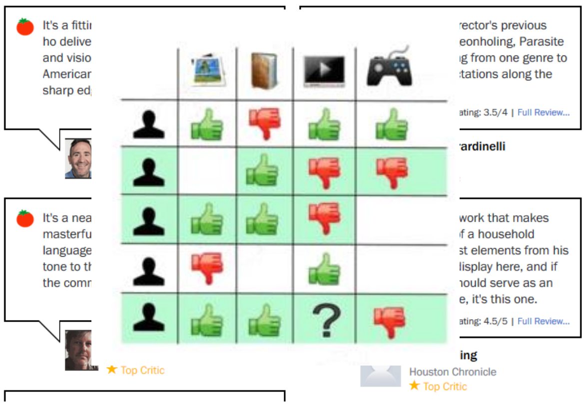
$$= \frac{0.4}{0.6 \times 0.6} = 1.11$$

Summary Association Rules

- **Association Rule Induction is a Two Step Process**
  - Find the frequent item sets (minimum support).
  - Form the relevant association rules (minimum confidence).
- **Finding the Frequent Item Sets**
  - Top-down search in the subset lattice / item set tree.
  - **Apriori:** Breadth first search;
- **Eclat**: Depth first search.
  - Other algorithms: FP-growth, H-Mine, LCM, Mafia, Relim etc.
  - Search Tree Pruning: *No superset of an infrequent item set can be frequent.*
- **Generating the Association Rules**
  - Form all possible association rules from the frequent item sets.
  - Filter “interesting” association rules.

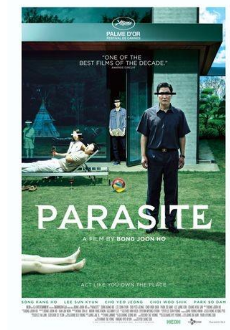
Collaborative Filtering

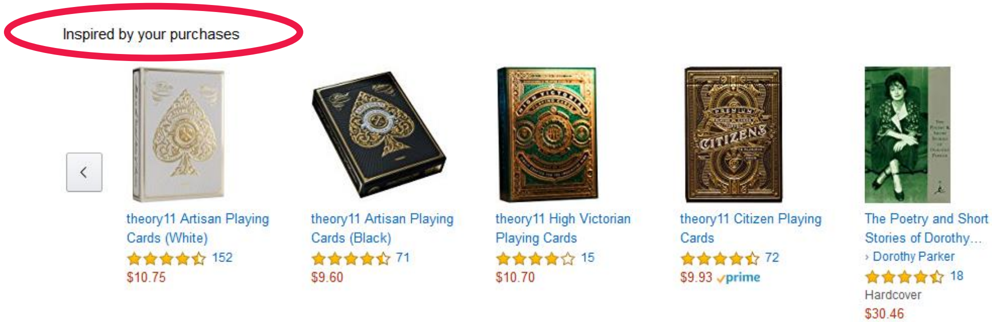
From the analysis of the reactions of many people to the same item ...



Collaborative Filtering

Recommendation

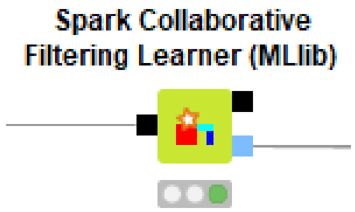




IF *A* has the same opinion as *B* on an item,
THEN A is more likely to have B's opinion on another item than that of a randomly chosen person

Collaborative filtering systems have many forms, but many common systems can be reduced to two steps:

1. Look for users who share the same rating patterns with the active user (the user whom the recommendation is for)
2. Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user
3. Implemented in Spark



https://www.knime.com/blog/movie-recommendations-with-spark-collaborative-filtering

- User u to give recommendations to
- U = set of top N users most similar to user u
- Rating of user u on item i calculated as average of ratings of all similar users in U:

$$r_{u,i} = \frac{1}{N}\sum_{u' \in U} r_{u',i} \quad \text{or weighted} \quad r_{u,i} = \frac{1}{N}\sum_{u' \in U} simil(u,u')\, r_{u',i}$$

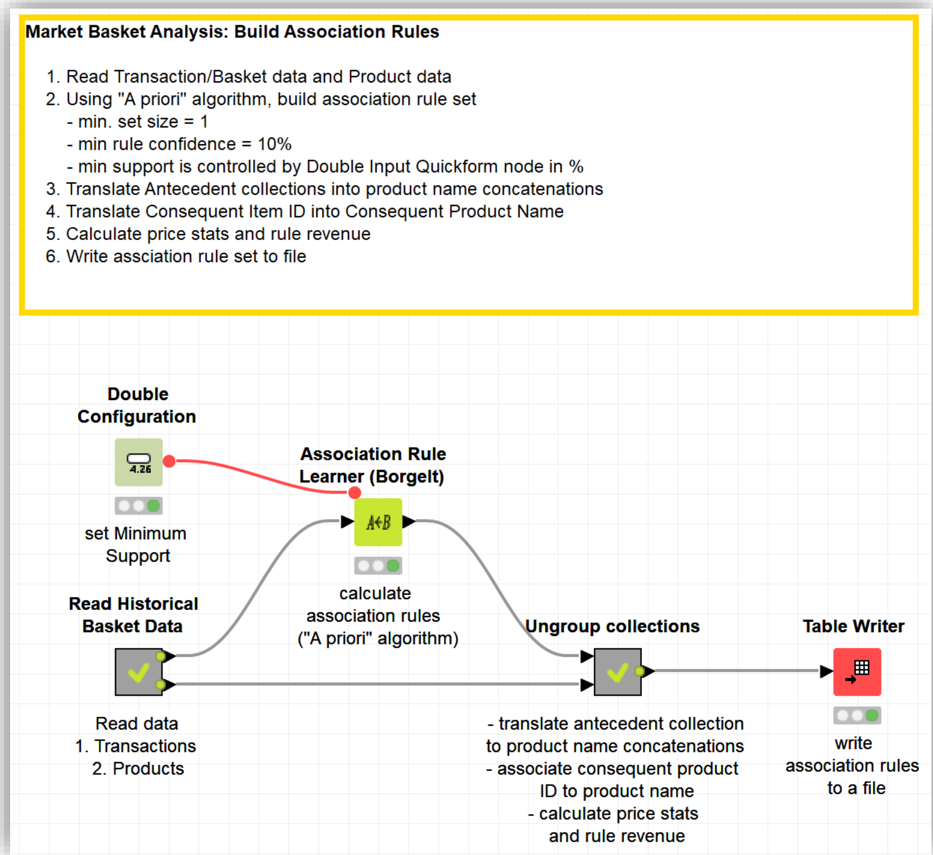
Pearson correlation

$$simil(u,u') = \frac{\sum_{i \in I_{xy}} (r_{u,i} - \overline{r_u})(r_{u',i} - \overline{r_{u'}})}{\sqrt{\sum_{i \in I_{xy}} (r_{u,i} - \overline{r_u})^2} \sqrt{\sum_{i \in I_{xy}} (r_{u',i} - \overline{r_{u'}})^2}}$$

Set of items rated by both user x and y

Practical Examples with KNIME Analytics Platform

KNIME Workflow



Thank you

For any questions please contact: education@knime.com