

Data Mining

Association Analysis: Basic Concepts and Algorithms

Lecture Notes for Chapter 6

Introduction to Data Mining
by
Tan, Steinbach, Kumar

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

<i>TID</i>	<i>Items</i>
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

$\{\text{Diaper}\} \rightarrow \{\text{Beer}\},$
 $\{\text{Milk, Bread}\} \rightarrow \{\text{Eggs, Coke}\},$
 $\{\text{Beer, Bread}\} \rightarrow \{\text{Milk}\},$

Implication means co-occurrence,
not causality!

Definition: Frequent Itemset

- **Itemset**

- A collection of one or more items
 - ◆ Example: {Milk, Bread, Diaper}
- k-itemset
 - ◆ An itemset that contains k items

- **Support count (σ)**

- Frequency of occurrence of an itemset
- E.g. $\sigma(\{\text{Milk, Bread, Diaper}\}) = 2$

- **Support**

- Fraction of transactions that contain an itemset
- E.g. $s(\{\text{Milk, Bread, Diaper}\}) = 2/5$

- **Frequent Itemset**

- An itemset whose support is greater than or equal to a *minsup* threshold

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

Definition: Association Rule

- **Association Rule**

- An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
- Example: $\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$

- **Rule Evaluation Metrics**

- Support (s)
 - ◆ Fraction of transactions that contain both X and Y
- Confidence (c)
 - ◆ Measures how often items in Y appear in transactions that contain X

Example:

$\{\text{Milk, Diaper}\} \Rightarrow \text{Beer}$

$$s = \frac{\sigma(\text{Milk, Diaper, Beer})}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} = 0.67$$

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

Association Rule Mining Task

- Given a set of transactions T , the goal of association rule mining is to find all rules having
 - support $\geq \text{minsup}$ threshold
 - confidence $\geq \text{minconf}$ threshold
- Brute-force approach:
 - List all possible association rules
 - Compute the support and confidence for each rule
 - Prune rules that fail the *minsup* and *minconf* thresholds

⇒ **Computationally prohibitive!**

Mining Association Rules

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

Example of Rules:

$\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$ ($s=0.4, c=0.67$)
 $\{\text{Milk, Beer}\} \rightarrow \{\text{Diaper}\}$ ($s=0.4, c=1.0$)
 $\{\text{Diaper, Beer}\} \rightarrow \{\text{Milk}\}$ ($s=0.4, c=0.67$)
 $\{\text{Beer}\} \rightarrow \{\text{Milk, Diaper}\}$ ($s=0.4, c=0.67$)
 $\{\text{Diaper}\} \rightarrow \{\text{Milk, Beer}\}$ ($s=0.4, c=0.5$)
 $\{\text{Milk}\} \rightarrow \{\text{Diaper, Beer}\}$ ($s=0.4, c=0.5$)

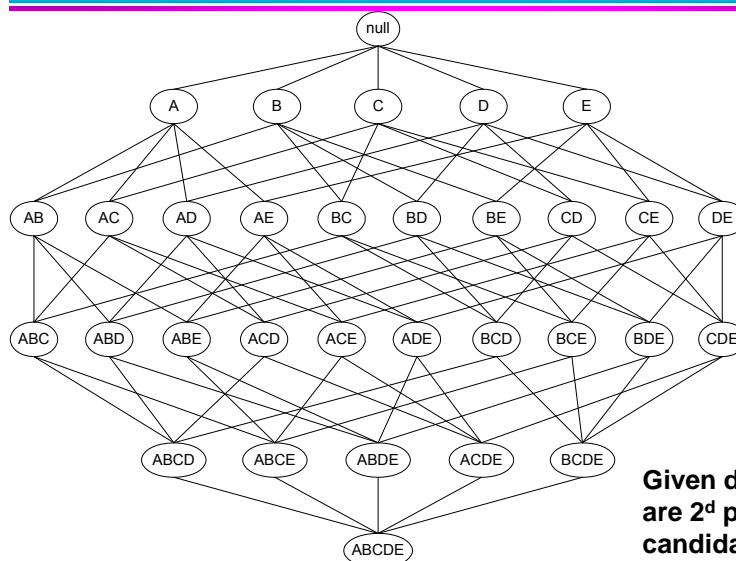
Observations:

- All the above rules are binary partitions of the same itemset:
 $\{\text{Milk, Diaper, Beer}\}$
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Mining Association Rules

- Two-step approach:
 1. **Frequent Itemset Generation**
 - Generate all itemsets whose support \geq minsup
 2. **Rule Generation**
 - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

Frequent Itemset Generation

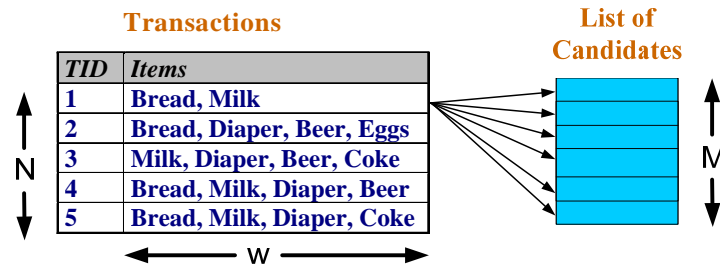


Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:

- Each itemset in the lattice is a **candidate** frequent itemset
- Count the support of each candidate by scanning the database

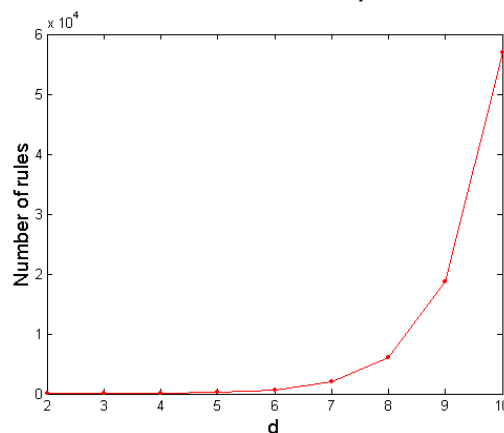


- Match each transaction against every candidate
- Complexity $\sim O(NMw) \Rightarrow$ **Expensive since $M = 2^d$!!!**

Computational Complexity

- Given d unique items:

- Total number of itemsets $= 2^d$
- Total number of possible association rules:



$$R = \sum_{k=1}^{d-1} \left[\binom{d}{k} \times \sum_{j=1}^{d-k} \binom{d-k}{j} \right]$$

$$= 3^d - 2^{d+1} + 1$$

If $d=6$, $R = 602$ rules

Frequent Itemset Generation Strategies

- Reduce the **number of candidates** (M)
 - Complete search: $M=2^d$
 - Use pruning techniques to reduce M
- Reduce the **number of transactions** (N)
 - Reduce size of N as the size of itemset increases
 - Used by hash-pruning algorithms
- Reduce the **number of comparisons** (NM)
 - Use efficient data structures to store the candidates or transactions
 - No need to match every candidate against every transaction

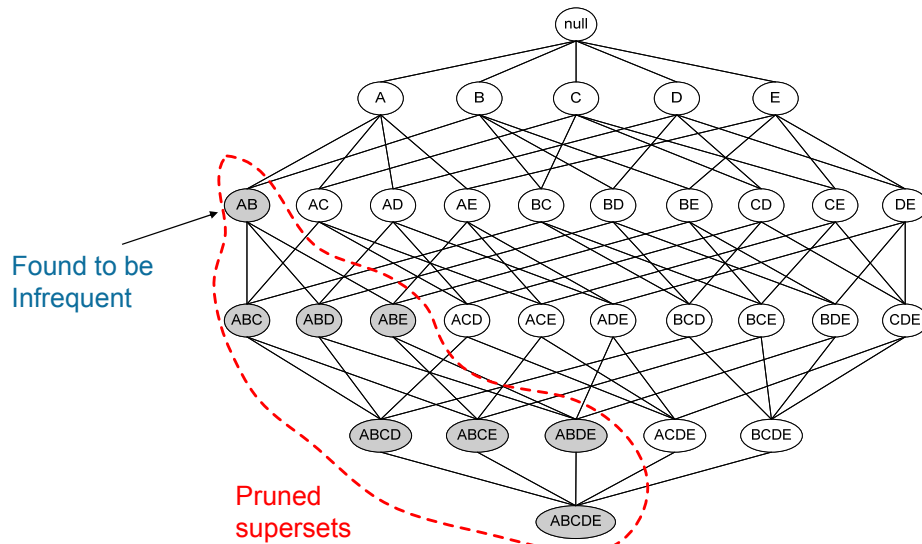
Reducing Number of Candidates

- **Apriori principle:**
 - If an itemset is frequent, then all of its subsets must also be frequent
- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- Support of an itemset never exceeds the support of its subsets
- This is known as the **anti-monotone** property of support

Illustrating Apriori Principle



Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)



Itemset	Count
{Bread, Milk}	3
{Bread, Beer}	2
{Bread, Diaper}	3
{Milk, Beer}	2
{Milk, Diaper}	3
{Beer, Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)



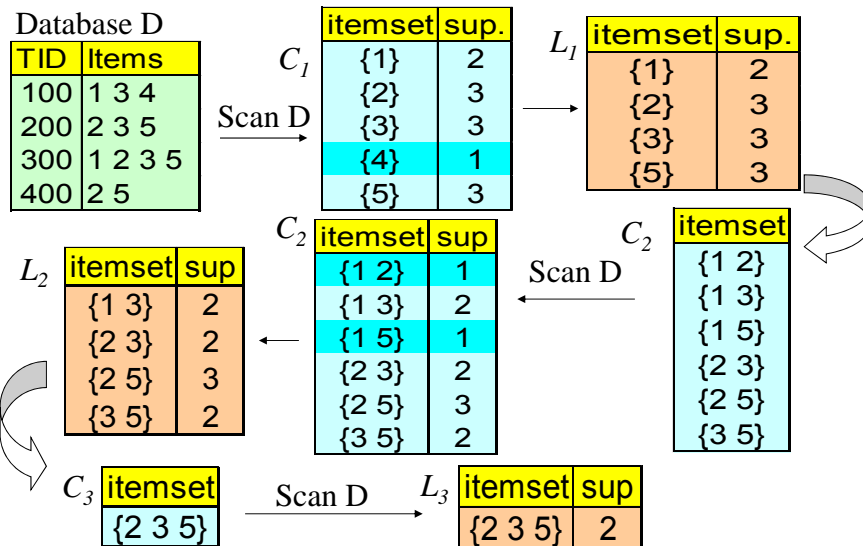
Triplets (3-itemsets)

Itemset	Count
{Bread, Milk, Diaper}	3

Minimum Support = 3

If every subset is considered,
 ${}^6C_1 + {}^6C_2 + {}^6C_3 = 41$
 With support-based pruning,
 $6 + 6 + 1 = 13$

The Apriori Algorithm — Example



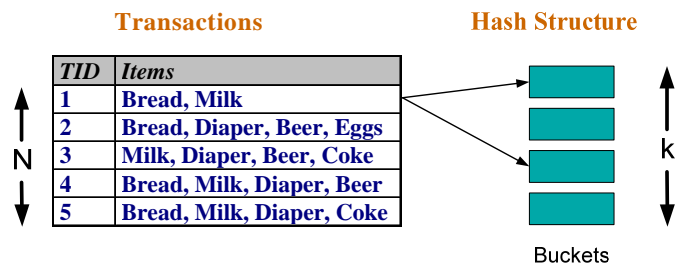
Apriori Algorithm

- Method:

- Let $k=1$
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are identified
 - ◆ Generate length $(k+1)$ candidate itemsets from length k frequent itemsets
 - ◆ Prune candidate itemsets containing subsets of length k that are infrequent
 - ◆ Count the support of each candidate by scanning the DB
 - ◆ Eliminate candidates that are infrequent, leaving only those that are frequent

Reducing Number of Comparisons

- Candidate counting:
 - Scan the database of transactions to determine the support of each candidate itemset
 - To reduce the number of comparisons, store the candidates in a hash structure
 - ◆ Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets



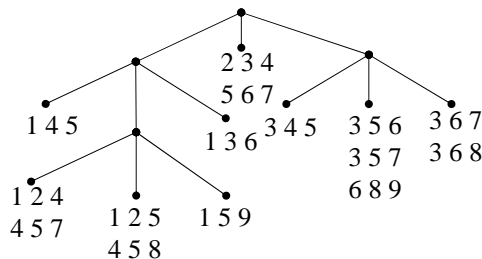
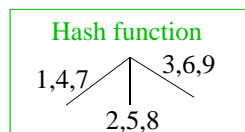
Generate Hash Tree

Suppose you have 15 candidate itemsets of length 3:

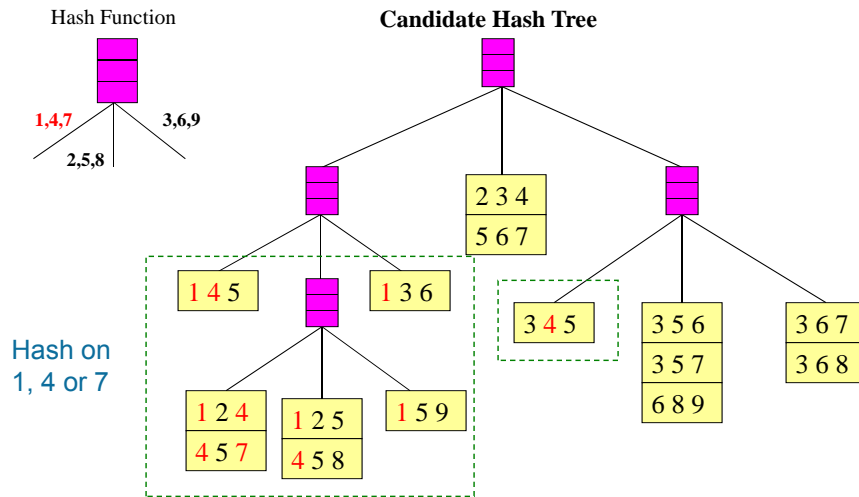
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5},
{3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

You need:

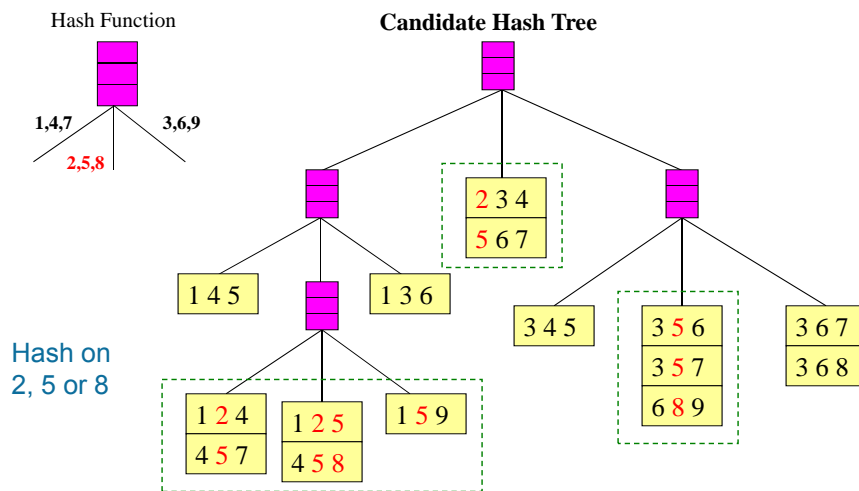
- Hash function
- Max leaf size: max number of itemsets stored in a leaf node (if number of candidate itemsets exceeds max leaf size, split the node)



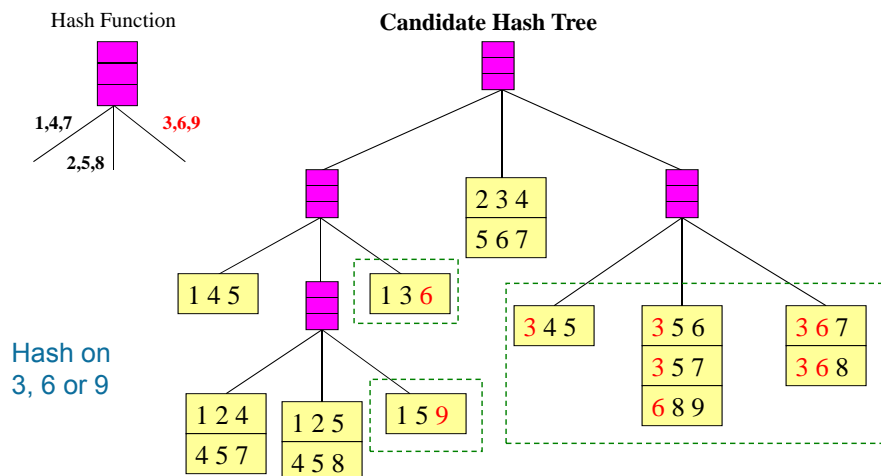
Association Rule Discovery: Hash tree



Association Rule Discovery: Hash tree

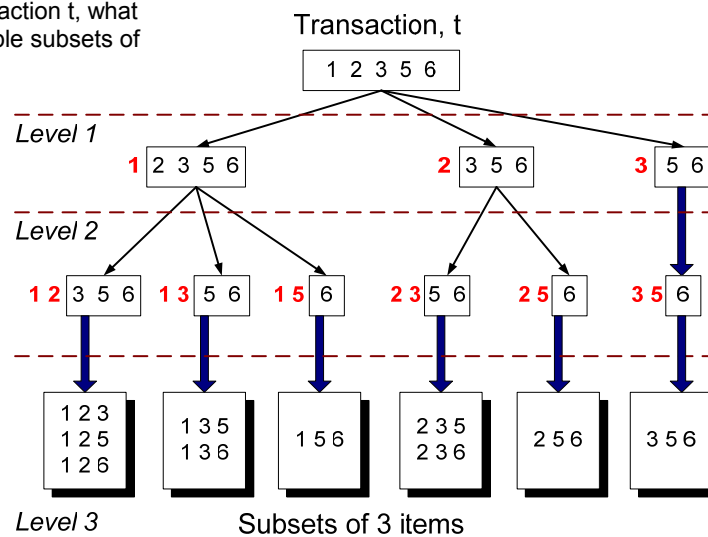


Association Rule Discovery: Hash tree

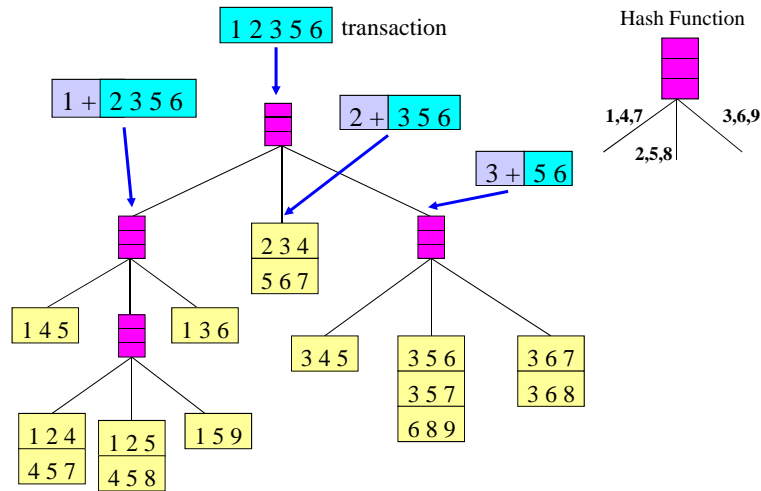


Subset Operation

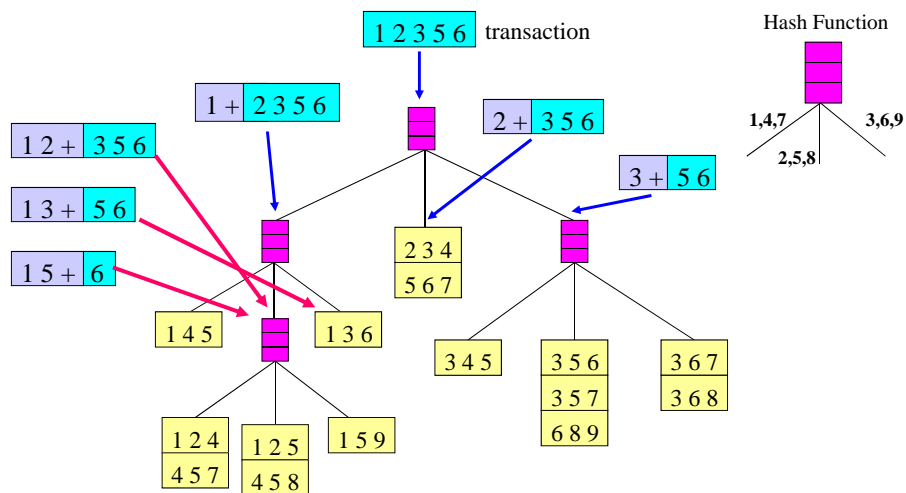
Given a transaction t , what are the possible subsets of size 3?



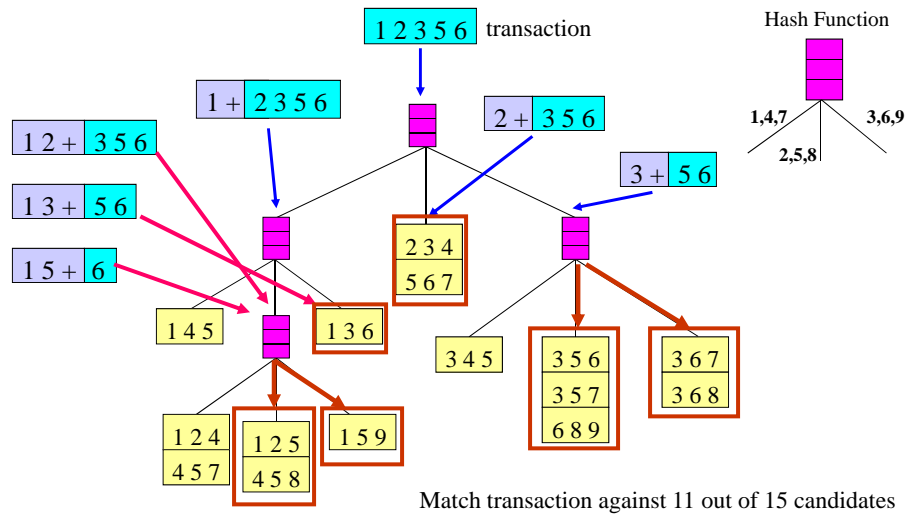
Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Factors Affecting Complexity

- Choice of minimum support threshold
 - lowering support threshold results in more frequent itemsets
 - this may increase number of candidates and max length of frequent itemsets
- Dimensionality (number of items) of the data set
 - more space is needed to store support count of each item
 - if number of frequent items also increases, both computation and I/O costs may also increase
- Size of database
 - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
 - transaction width increases with denser data sets
 - This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

Compact Representation of Frequent Itemsets

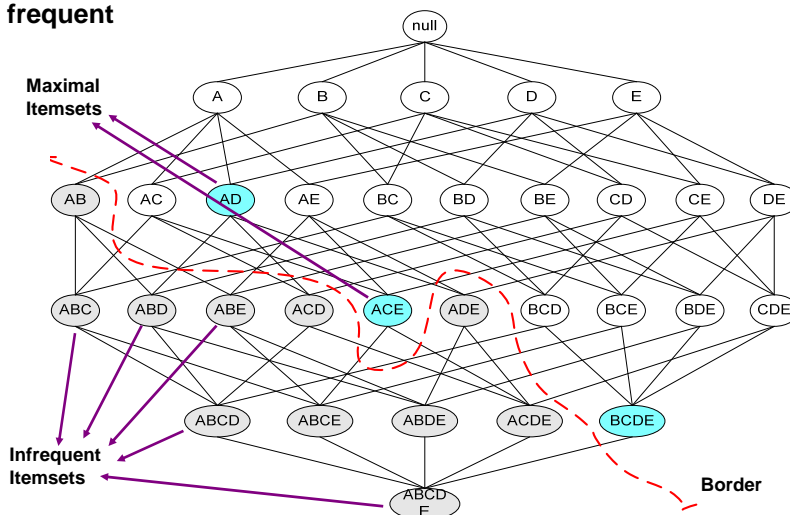
- Some itemsets are redundant because they have identical support as their supersets

TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

- Number of frequent itemsets = $3 \times \sum_{k=1}^{10} \binom{10}{k}$
- Need a compact representation

Maximal Frequent Itemset

An itemset is maximal frequent if none of its immediate supersets is frequent



Closed Itemset

- An itemset is closed if none of its immediate supersets has the same support as the itemset

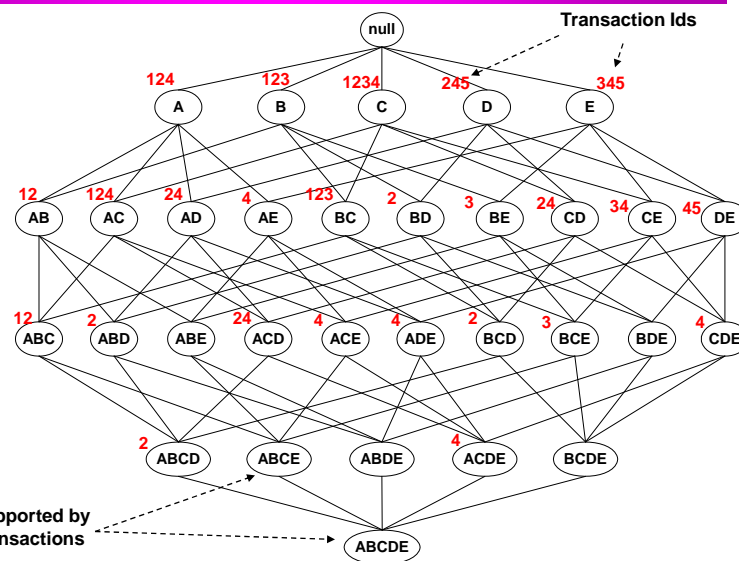
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

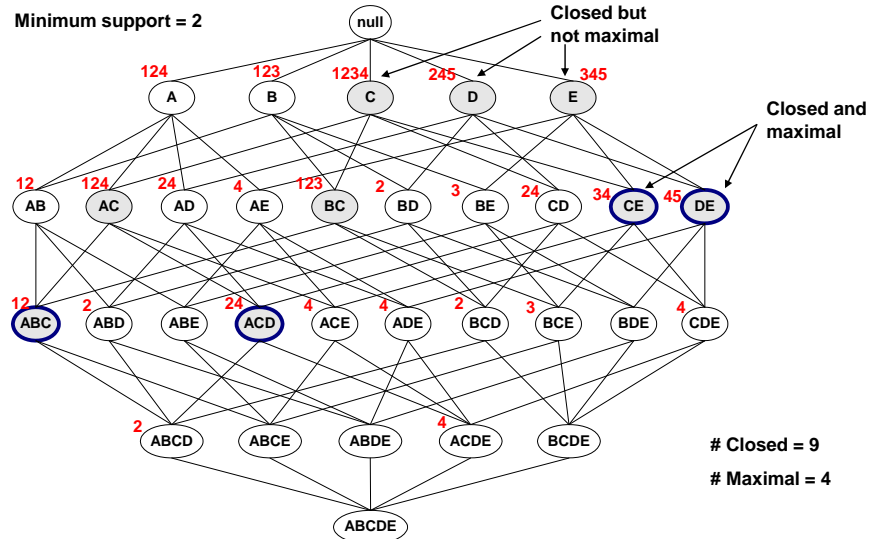
Maximal vs Closed Itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

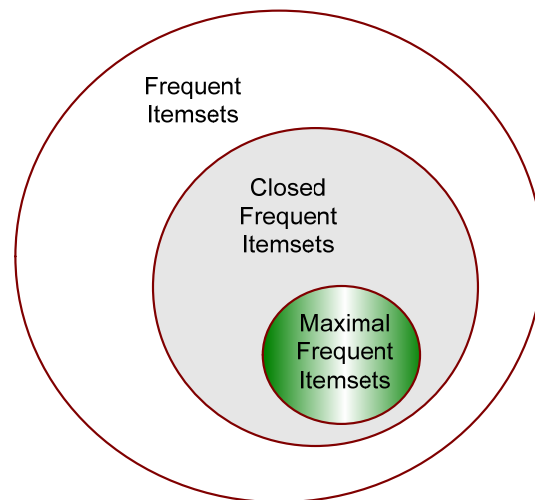


Maximal vs Closed Frequent Itemsets

Minimum support = 2



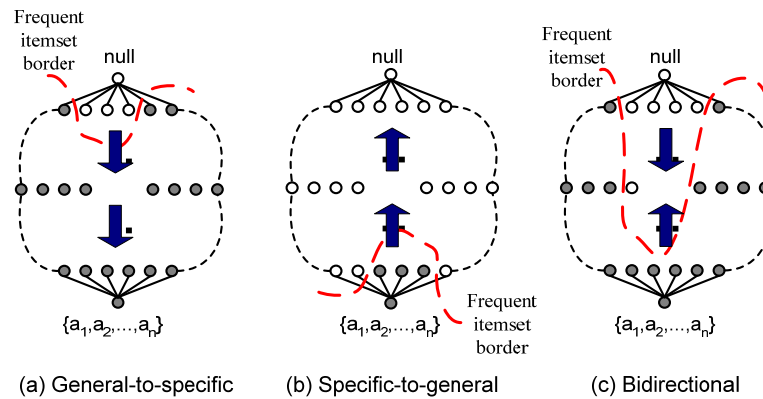
Maximal vs Closed Itemsets



Alternative Methods for Frequent Itemset Generation

• Traversal of Itemset Lattice

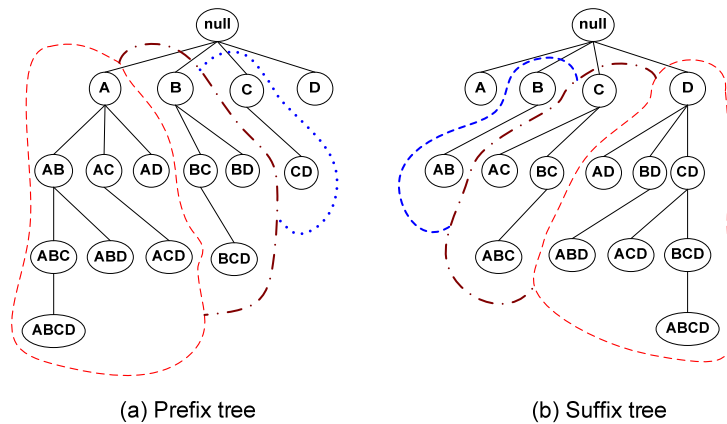
– General-to-specific vs Specific-to-general



Alternative Methods for Frequent Itemset Generation

• Traversal of Itemset Lattice

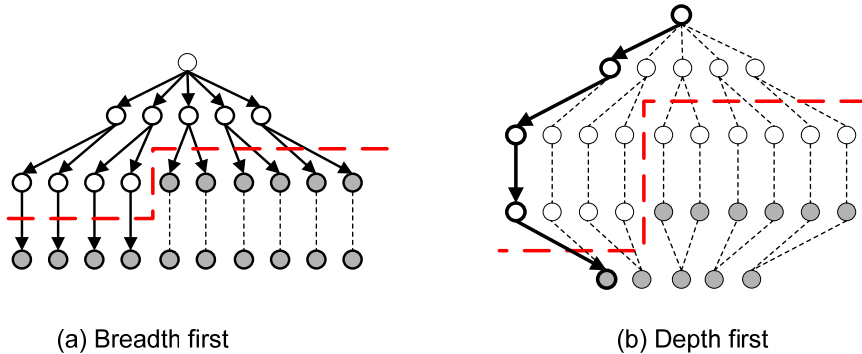
– Equivalent Classes



Alternative Methods for Frequent Itemset Generation

• Traversal of Itemset Lattice

- Breadth-first vs Depth-first



Alternative Methods for Frequent Itemset Generation

• Representation of Database

- horizontal vs vertical data layout

Horizontal
Data Layout

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B

Vertical Data Layout

	A	B	C	D	E
1	1	2	2	2	1
4	2	3	3	4	3
5	5	4	4	5	6
6	7	8	8	9	
7	8	9	9		
8	10				
9					

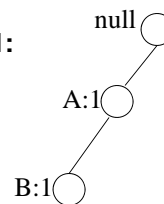
FP-growth Algorithm

- Use a compressed representation of the database using an **FP-tree**
- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets

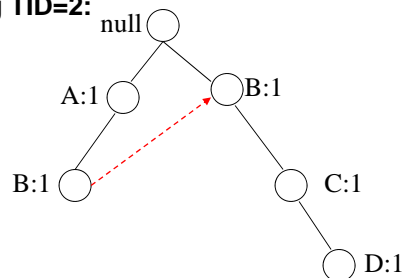
FP-tree construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

After reading TID=1:



After reading TID=2:



FP-Tree Construction

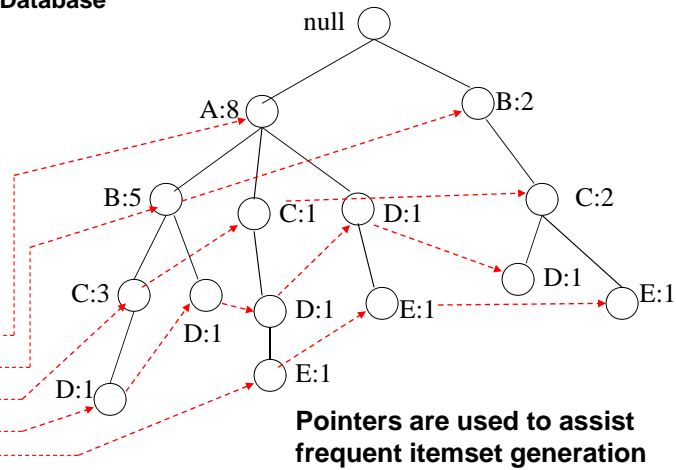
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Transaction
Database

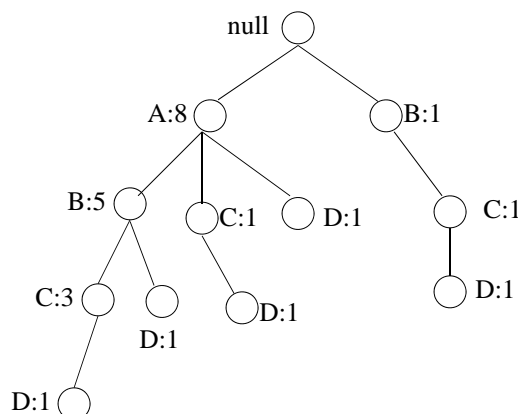
Reorder and prune DB by item frequency
Order header table by item frequency

Header table

Item	Pointer
A	
B	
C	
D	
E	



FP-growth



Conditional Pattern base
for D:

$P = \{(A:1, B:1, C:1),$
 $(A:1, B:1),$
 $(A:1, C:1),$
 $(A:1),$
 $(B:1, C:1)\}$

Recursively apply FP-
growth on P

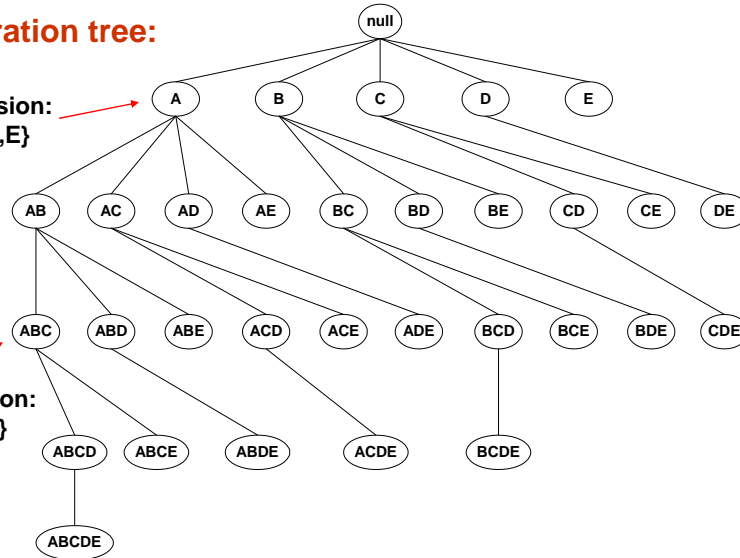
Frequent Itemsets found
(with sup > 1):
 AD, BD, CD, ACD, BCD

Tree Projection

Set enumeration tree:

Possible Extension:
 $E(A) = \{B, C, D, E\}$

Possible Extension:
 $E(ABC) = \{D, E\}$



Tree Projection

- Items are listed in lexicographic order
- Each node P stores the following information:
 - Itemset for node P
 - List of possible lexicographic extensions of P: $E(P)$
 - Pointer to projected database of its ancestor node
 - Bitvector containing information about which transactions in the projected database contain the itemset

Projected Database

Original Database:

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Projected Database
for node A:

TID	Items
1	{B}
2	{}
3	{C,D,E}
4	{D,E}
5	{B,C}
6	{B,C,D}
7	{}
8	{B,C}
9	{B,D}
10	{}

For each transaction T, projected transaction at node A is $T \cap E(A)$

ECLAT

- For each item, store a list of transaction ids (tids)

Horizontal
Data Layout

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B

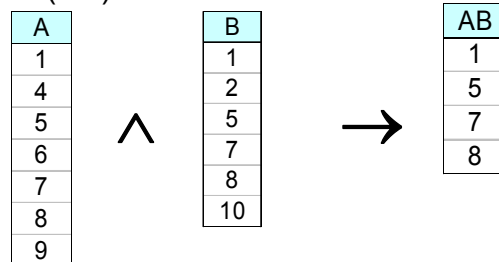
Vertical Data Layout

A	B	C	D	E
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

↓
TID-list

ECLAT

- Determine support of any k-itemset by intersecting tid-lists of two of its (k-1) subsets.



- 3 traversal approaches:
 - top-down, bottom-up and hybrid
- Advantage: very fast support counting
- Disadvantage: intermediate tid-lists may become too large for memory

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
 - If $\{A, B, C, D\}$ is a frequent itemset, candidate rules:

ABC \rightarrow D,	ABD \rightarrow C,	ACD \rightarrow B,	BCD \rightarrow A,
A \rightarrow BCD,	B \rightarrow ACD,	C \rightarrow ABD,	D \rightarrow ABC
AB \rightarrow CD,	AC \rightarrow BD,	AD \rightarrow BC,	BC \rightarrow AD,
BD \rightarrow AC,	CD \rightarrow AB,		
- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \emptyset$ and $\emptyset \rightarrow L$)

Rule Generation

- How to efficiently generate rules from frequent itemsets?

- In general, confidence does not have an anti-monotone property

$c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

- But confidence of rules generated from the same itemset has an anti-monotone property

- e.g., $L = \{A, B, C, D\}$:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

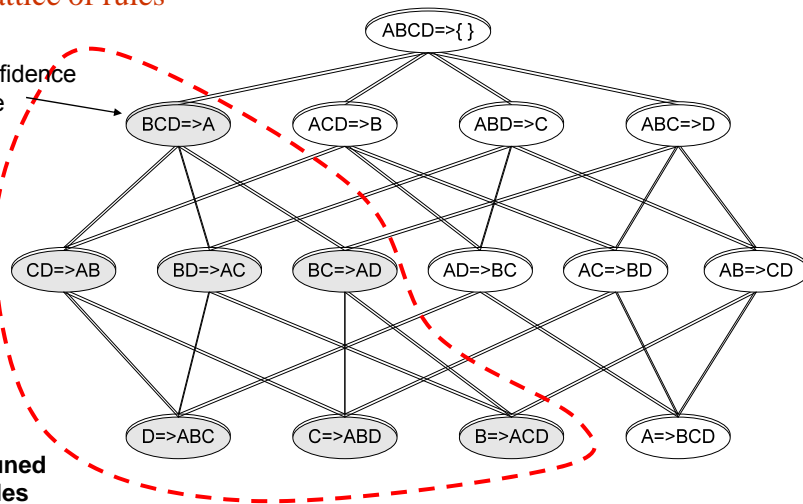
◆ Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule Generation for Apriori Algorithm

Lattice of rules

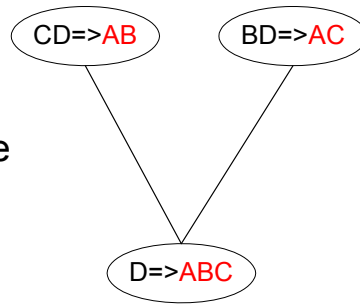
Low Confidence Rule

Pruned Rules



Rule Generation for Apriori Algorithm

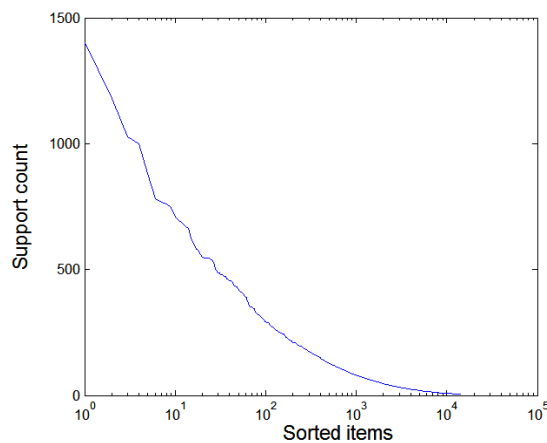
- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent
- $\text{join}(\text{CD} \Rightarrow \text{AB}, \text{BD} \Rightarrow \text{AC})$ would produce the candidate rule $\text{D} \Rightarrow \text{ABC}$
- Prune rule $\text{D} \Rightarrow \text{ABC}$ if its subset $\text{AD} \Rightarrow \text{BC}$ does not have high confidence



Effect of Support Distribution

- Many real data sets have skewed support distribution

Support distribution of a retail data set



Effect of Support Distribution

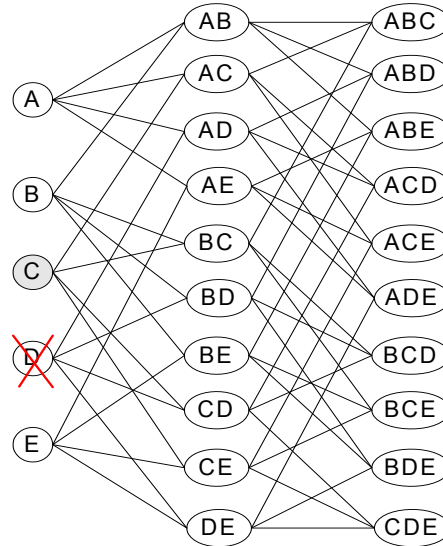
- How to set the appropriate *minsup* threshold?
 - If *minsup* is set too high, we could miss itemsets involving interesting rare items (e.g., expensive products)
 - If *minsup* is set too low, it is computationally expensive and the number of itemsets is very large
- Using a single minimum support threshold may not be effective

Multiple Minimum Support

- How to apply multiple minimum supports?
 - $MS(i)$: minimum support for item i
 - e.g.: $MS(\text{Milk})=5\%$, $MS(\text{Coke}) = 3\%$,
 $MS(\text{Broccoli})=0.1\%$, $MS(\text{Salmon})=0.5\%$
 - $MS(\{\text{Milk}, \text{Broccoli}\}) = \min (MS(\text{Milk}), MS(\text{Broccoli}))$
 $= 0.1\%$
 - Challenge: Support is no longer anti-monotone
 - ♦ Suppose: $\text{Support}(\text{Milk}, \text{Coke}) = 1.5\%$ and
 $\text{Support}(\text{Milk}, \text{Coke}, \text{Broccoli}) = 0.5\%$
 - ♦ $\{\text{Milk}, \text{Coke}\}$ is infrequent but $\{\text{Milk}, \text{Coke}, \text{Broccoli}\}$ is frequent

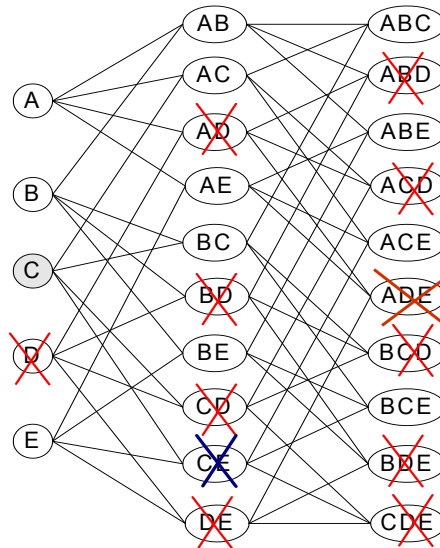
Multiple Minimum Support

Item	MS(I)	Sup(I)
A	0.10%	0.25%
B	0.20%	0.26%
C	0.30%	0.29%
D	0.50%	0.05%
E	3%	4.20%



Multiple Minimum Support

Item	MS(I)	Sup(I)
A	0.10%	0.25%
B	0.20%	0.26%
C	0.30%	0.29%
D	0.50%	0.05%
E	3%	4.20%



Multiple Minimum Support (Liu 1999)

- Order the items according to their minimum support (in ascending order)
 - e.g.: $MS(\text{Milk})=5\%$, $MS(\text{Coke}) = 3\%$,
 $MS(\text{Broccoli})=0.1\%$, $MS(\text{Salmon})=0.5\%$
 - Ordering: Broccoli, Salmon, Coke, Milk
- Need to modify Apriori such that:
 - L_1 : set of frequent items
 - F_1 : set of items whose support is $\geq MS(1)$
where $MS(1)$ is $\min_i(MS(i))$
 - C_2 : candidate itemsets of size 2 is generated from F_1
instead of L_1

Multiple Minimum Support (Liu 1999)

- Modifications to Apriori:
 - In traditional Apriori,
 - ◆ A candidate $(k+1)$ -itemset is generated by merging two frequent itemsets of size k
 - ◆ The candidate is pruned if it contains any infrequent subsets of size k
 - Pruning step has to be modified:
 - ◆ Prune only if subset contains the first item
 - ◆ e.g.: Candidate = {Broccoli, Coke, Milk} (ordered according to minimum support)
 - ◆ {Broccoli, Coke} and {Broccoli, Milk} are frequent but {Coke, Milk} is infrequent
 - Candidate is not pruned because {Coke, Milk} does not contain the first item, i.e., Broccoli.