CS 501: Data Mining

Association Analysis

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

The Task

- Two ways of defining the task
- General
  - Input: A collection of instances
  - Output: rules to predict the values of any attribute(s) (not just the class attribute) from values of other attributes
  - E.g. if temperature = cool then humidity =normal
  - If the right hand side of a rule has only the class attribute, then the rule is a classification rule
    - Distinction: Classification rules are applied together as sets of rules
- Specific - Market-basket analysis
  - Input: a collection of transactions
  - Output: rules to predict the occurrence of any item(s) from the occurrence of other items in a transaction
  - E.g. {Milk, Diaper} -> {Beer}
- General rule structure:
  - Antecedents -> Consequents

The Market-Basket Model

- A large set of *items*, e.g., things sold in a supermarket
- A large set of *baskets*, each of which is a small set of the items, e.g., the items one customer buys on one day

Market-Baskets – (2)

- Really a general many-many mapping (association) between two kinds of items
  - But we ask about connections among “items,” not “baskets”
- The technology focuses on common events, not rare events (“long tail”)

Example

Market-Basket transactions

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 Association Rules

{Diaper} $\rightarrow$ {Beer},
{Milk, Bread} $\rightarrow$ {Eggs,Coke},
{Beer, Bread} $\rightarrow$ {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. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({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:
    {Milk, Diaper} $\rightarrow$ {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

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:

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

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

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

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *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

$\Rightarrow$ 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:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

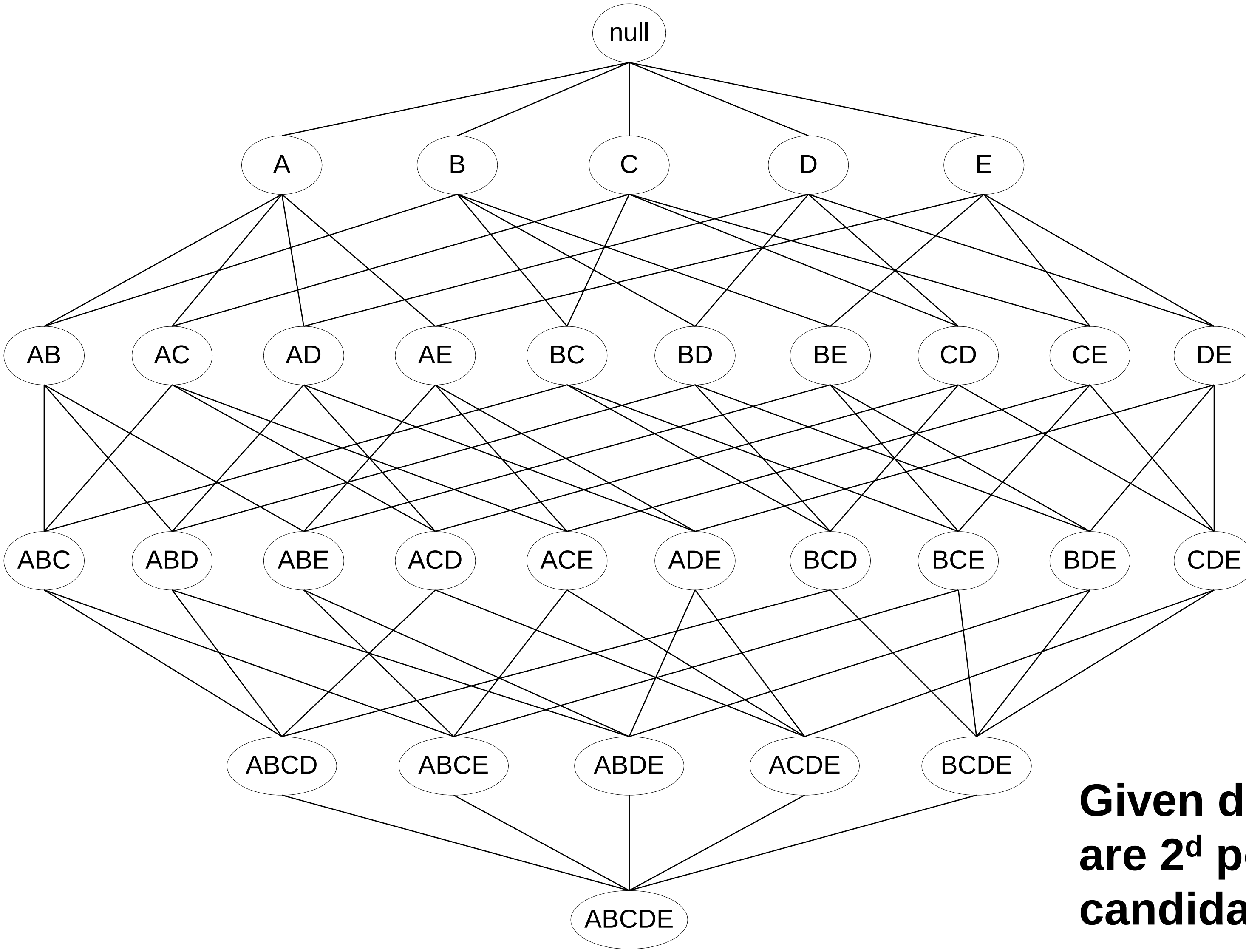
- All the above rules are binary partitions of the same itemset: {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence measures
- Thus, we may decouple the support and confidence requirements

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

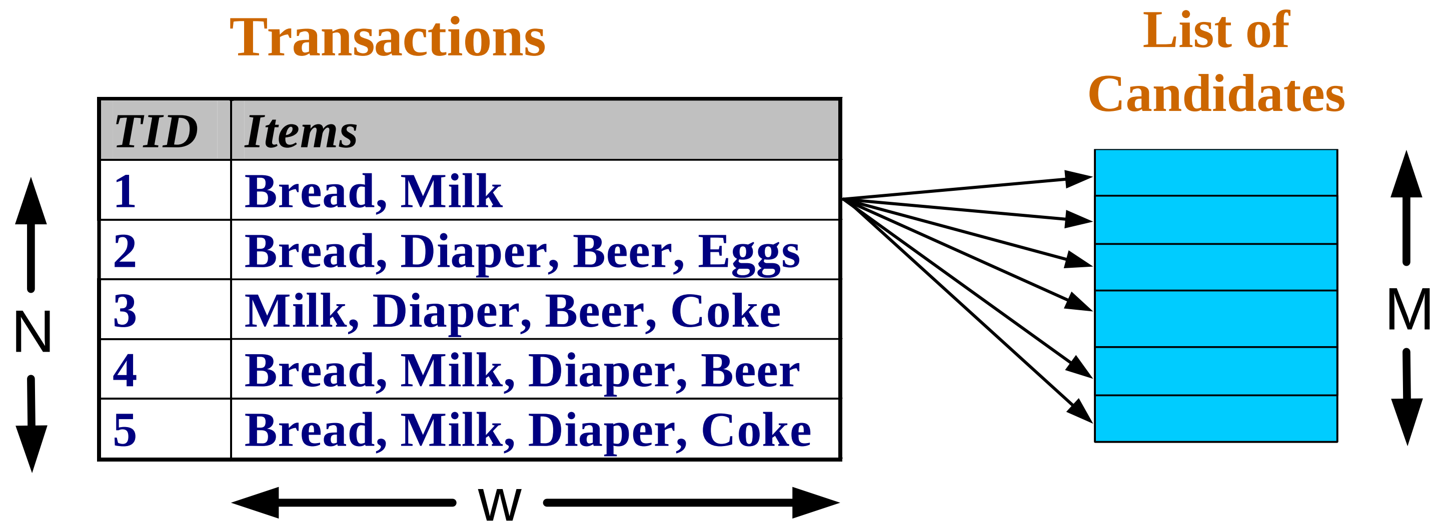
Frequent Itemset Generation



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

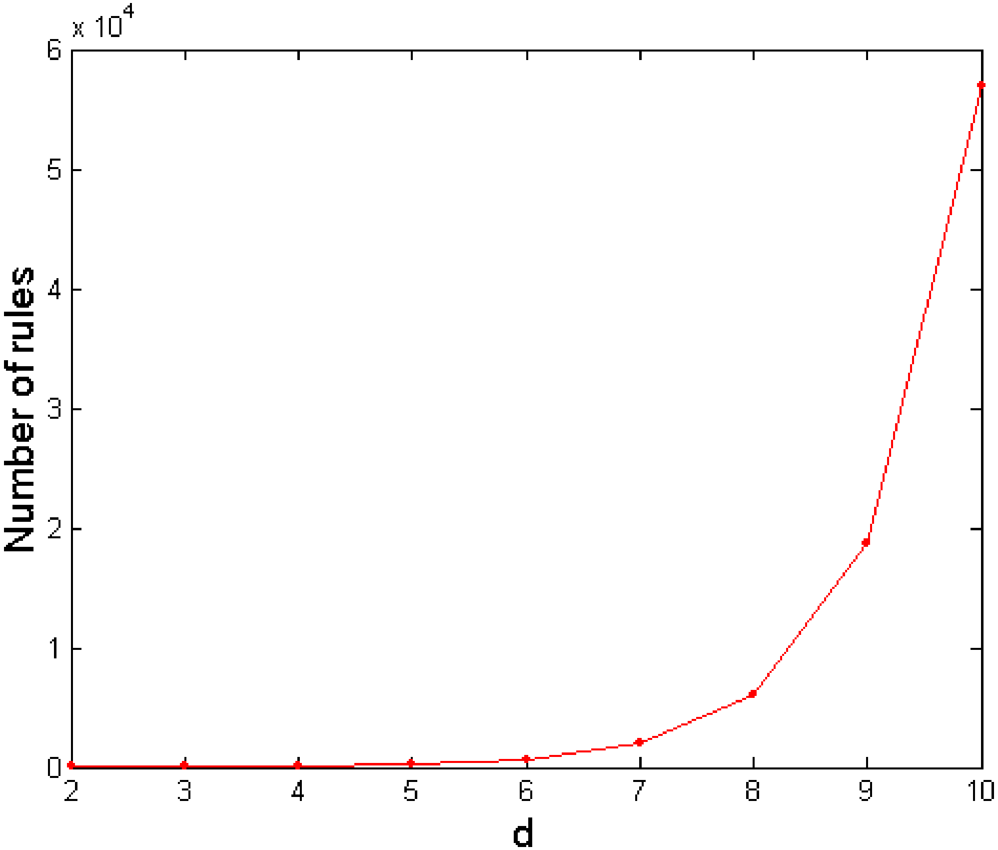


TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => 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

- 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

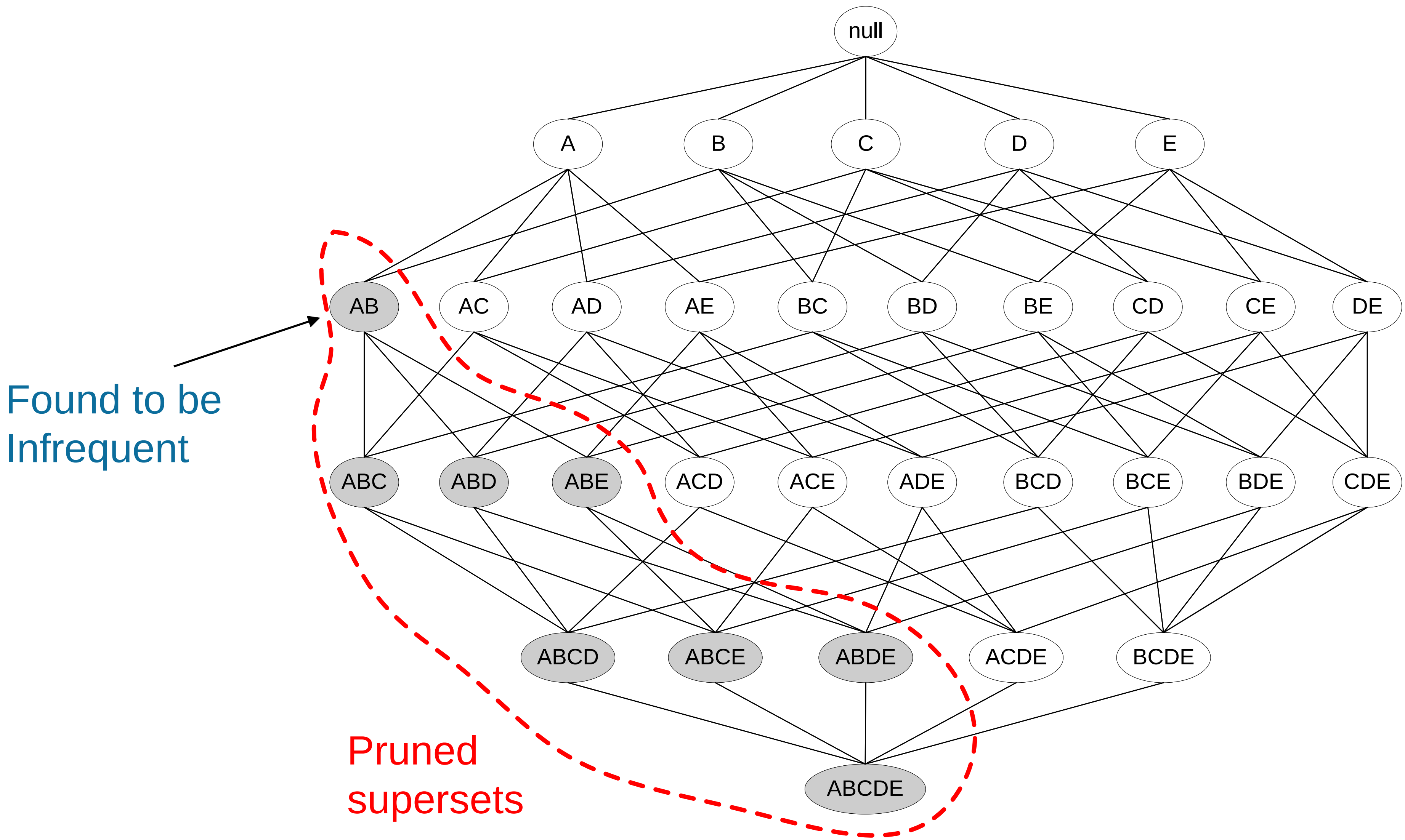
- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent

- 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
  - 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)

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

If every subset is considered,

$^{6}C_{1} + {}^{6}C_{2} + {}^{6}C_{3} = 41$

With support-based pruning,

$6 + 6 + 1 = 13$

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

The Apriori Algorithm: Basic idea

- Join Step: C_k is generated by joining L_{k-1} with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset
- Pseudo-code:

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction t in database do
    increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
return $\cup_k L_k$;

The Apriori Algorithm — Example

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

  insert into $\boldsymbol{C_k}$

  select $\boldsymbol{p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}}$

  from $\boldsymbol{L_{k-1}\ p, L_{k-1}\ q}$

  where $\boldsymbol{p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}}$
- Step 2: pruning

  for all ***itemsets c in*** $\boldsymbol{C_k}$ do

  for all ***(k-1)-subsets s of c*** do

  if *(c is not in* L_{k-1}*)* **then delete** *c* **from** C_k

Reducing Number of Comparisons

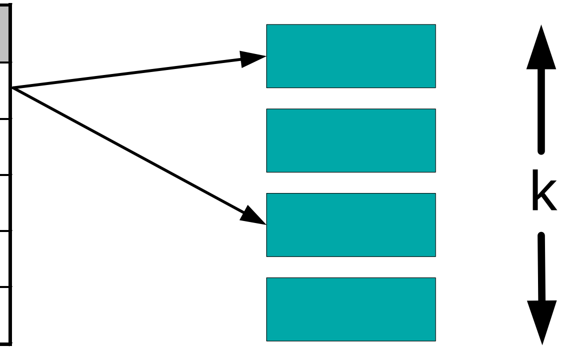
- Candidate counting:
  – Scan the database of transactions to determine the support of each candidate itemset
  – To reduce the number of comparisons, store the candidates in a hash structure
    - Instead of matching each transaction against every candidate, match it against candidates contained in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

Transactions

N

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

Hash Structure



Buckets

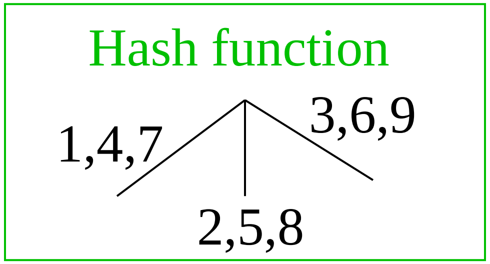
Generate Hash Tree

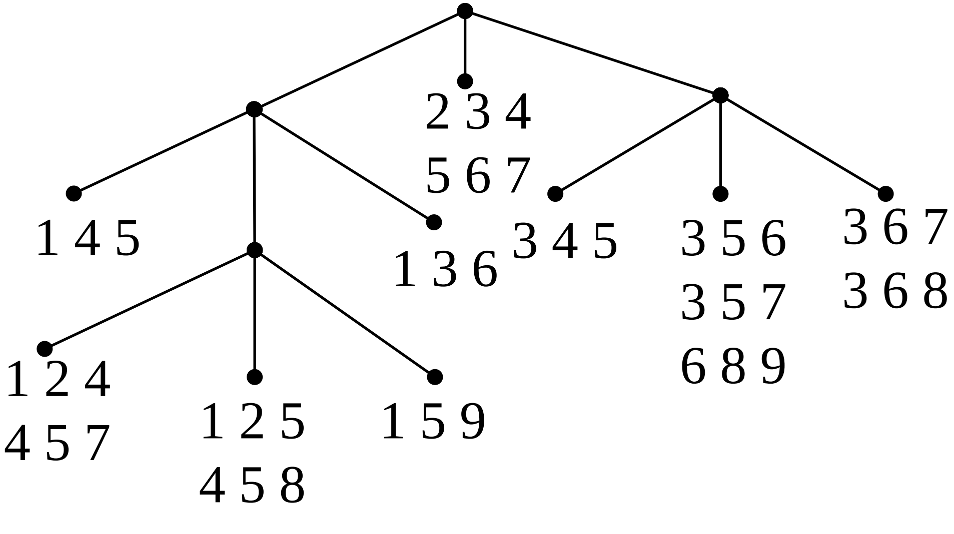
Suppose you have 15 candidate itemsets of length 3:

{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}

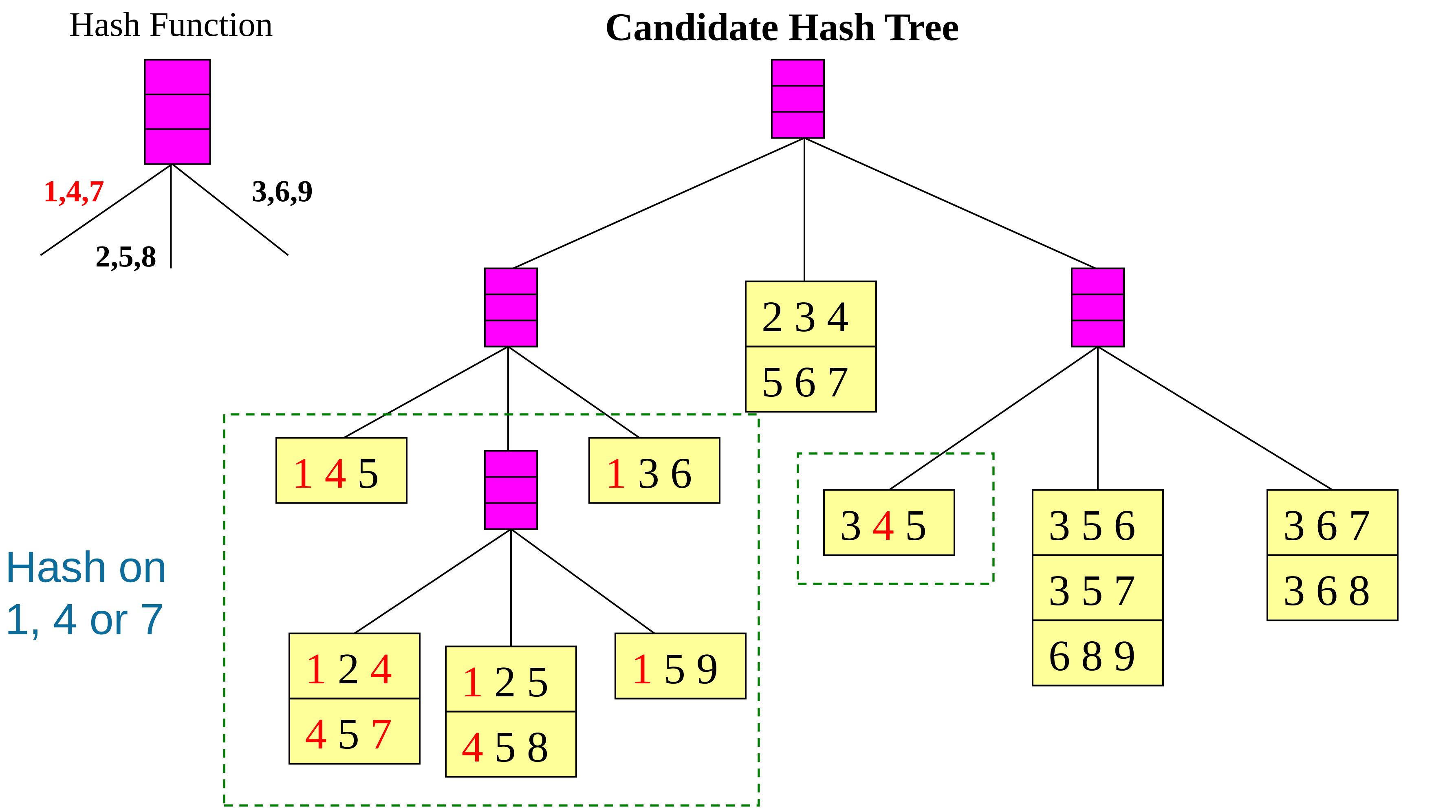
You need:

- **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)**

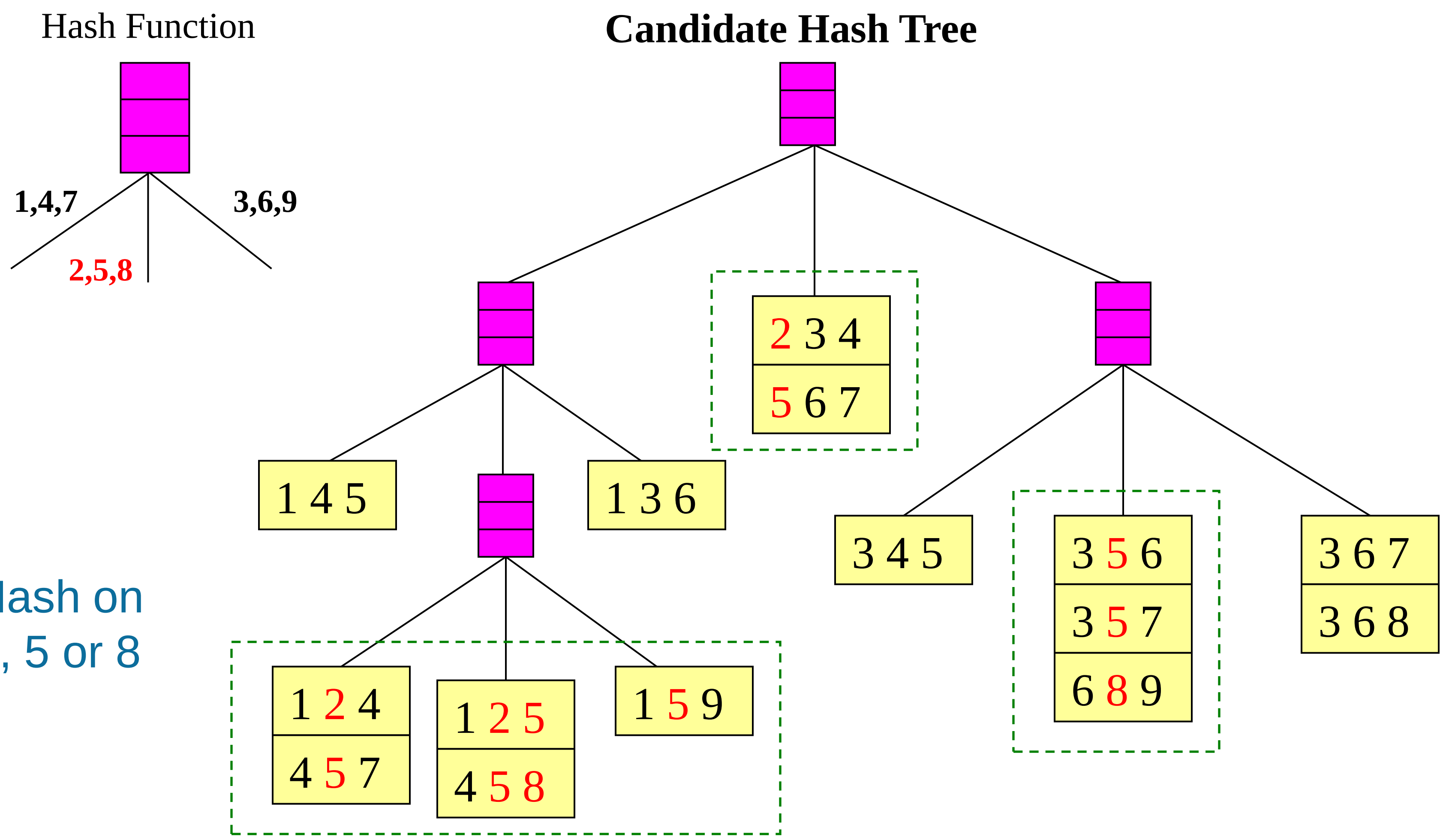




Association Rule Discovery: Hash tree

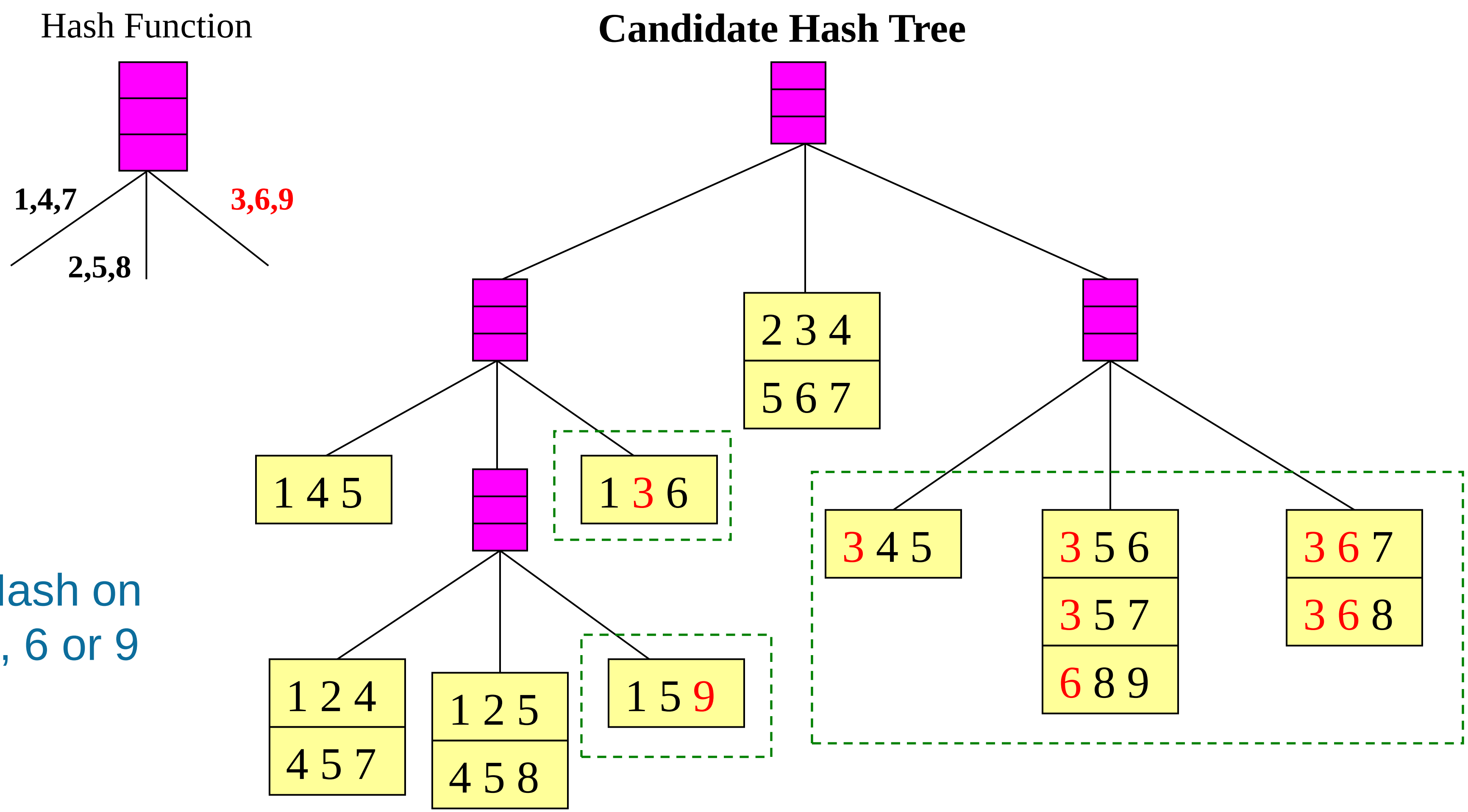


Association Rule Discovery: Hash tree



Hash on
2, 5 or 8

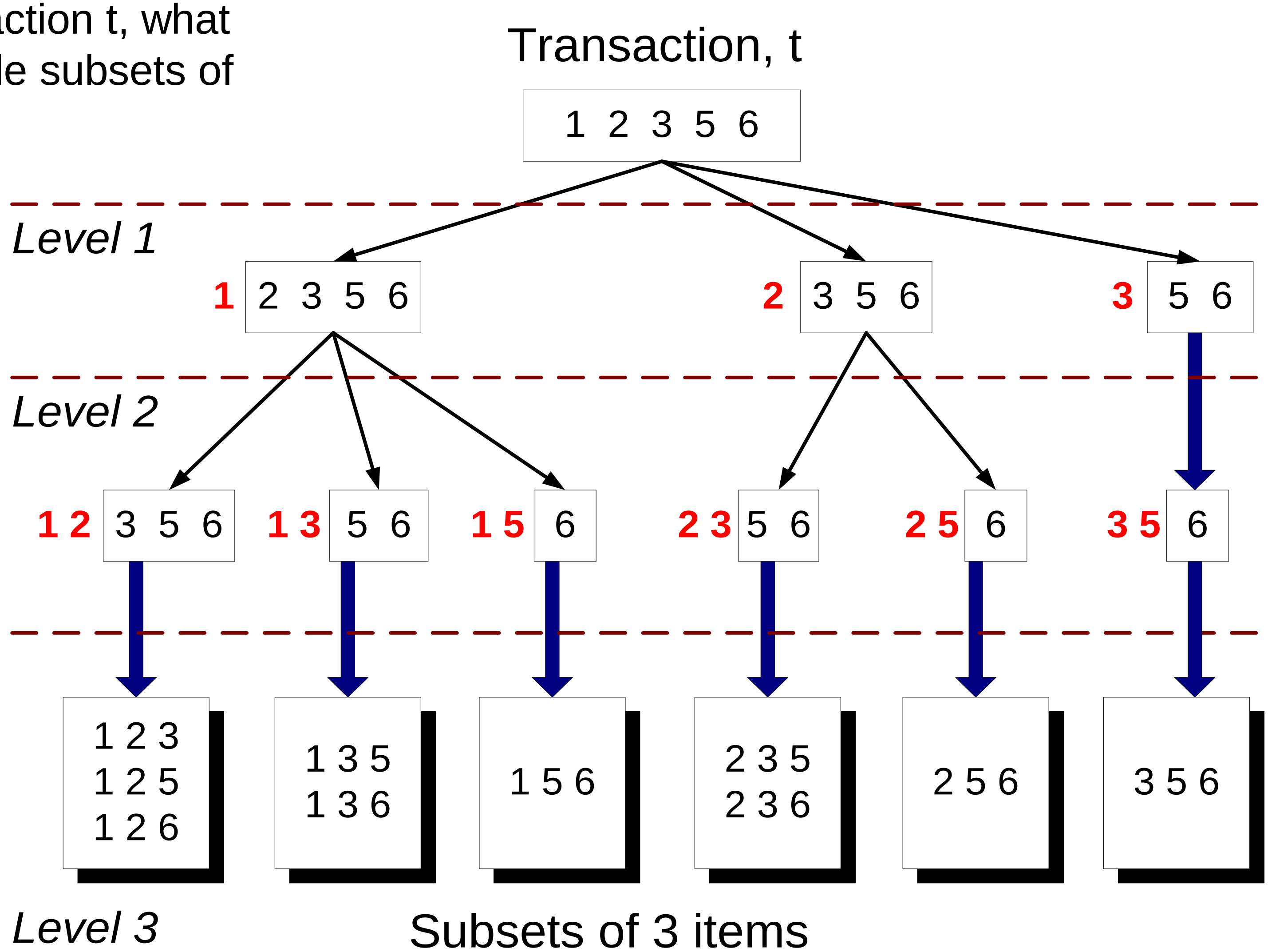
Association Rule Discovery: Hash tree



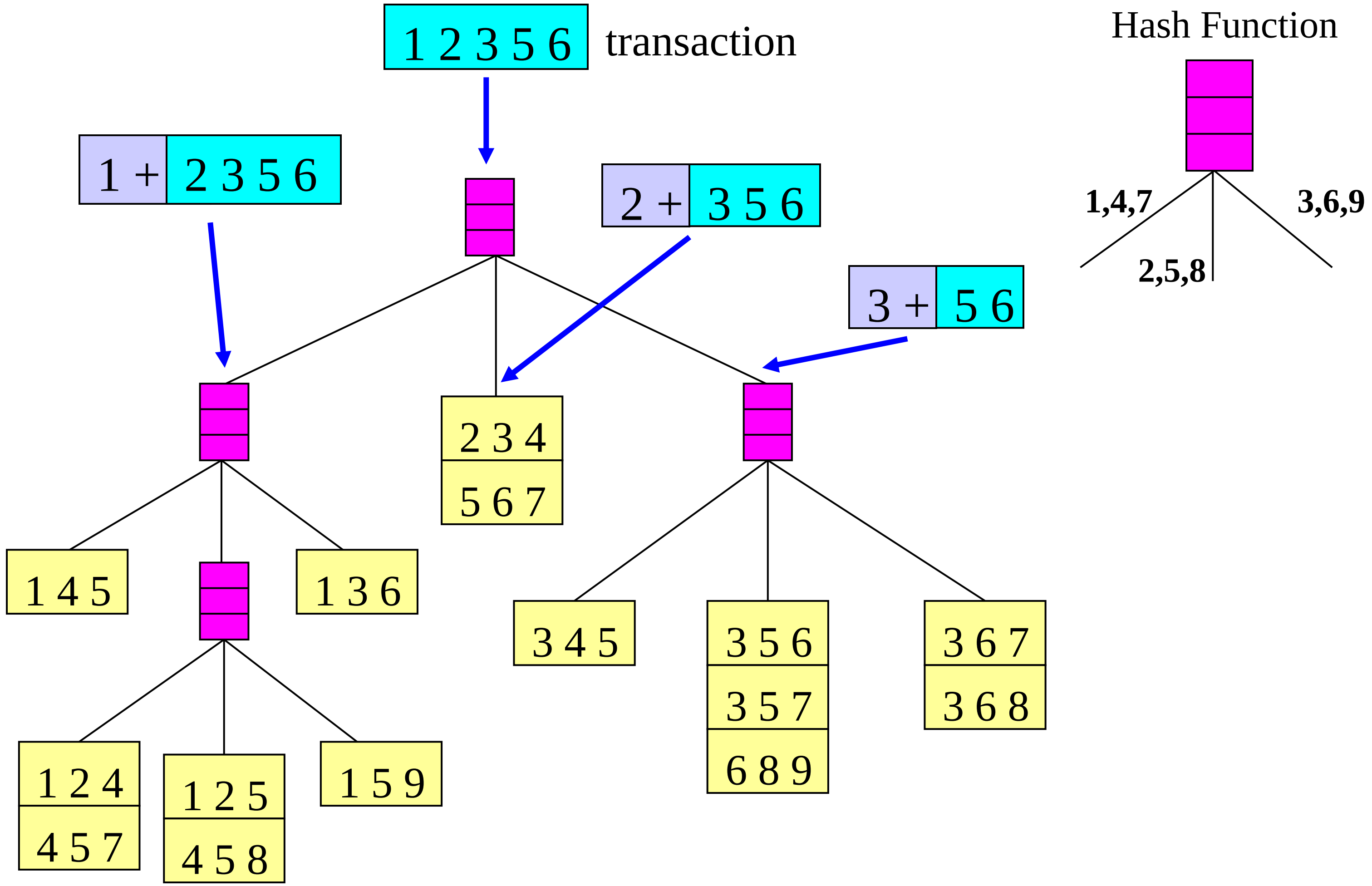
Hash on
3, 6 or 9

Subset Operation

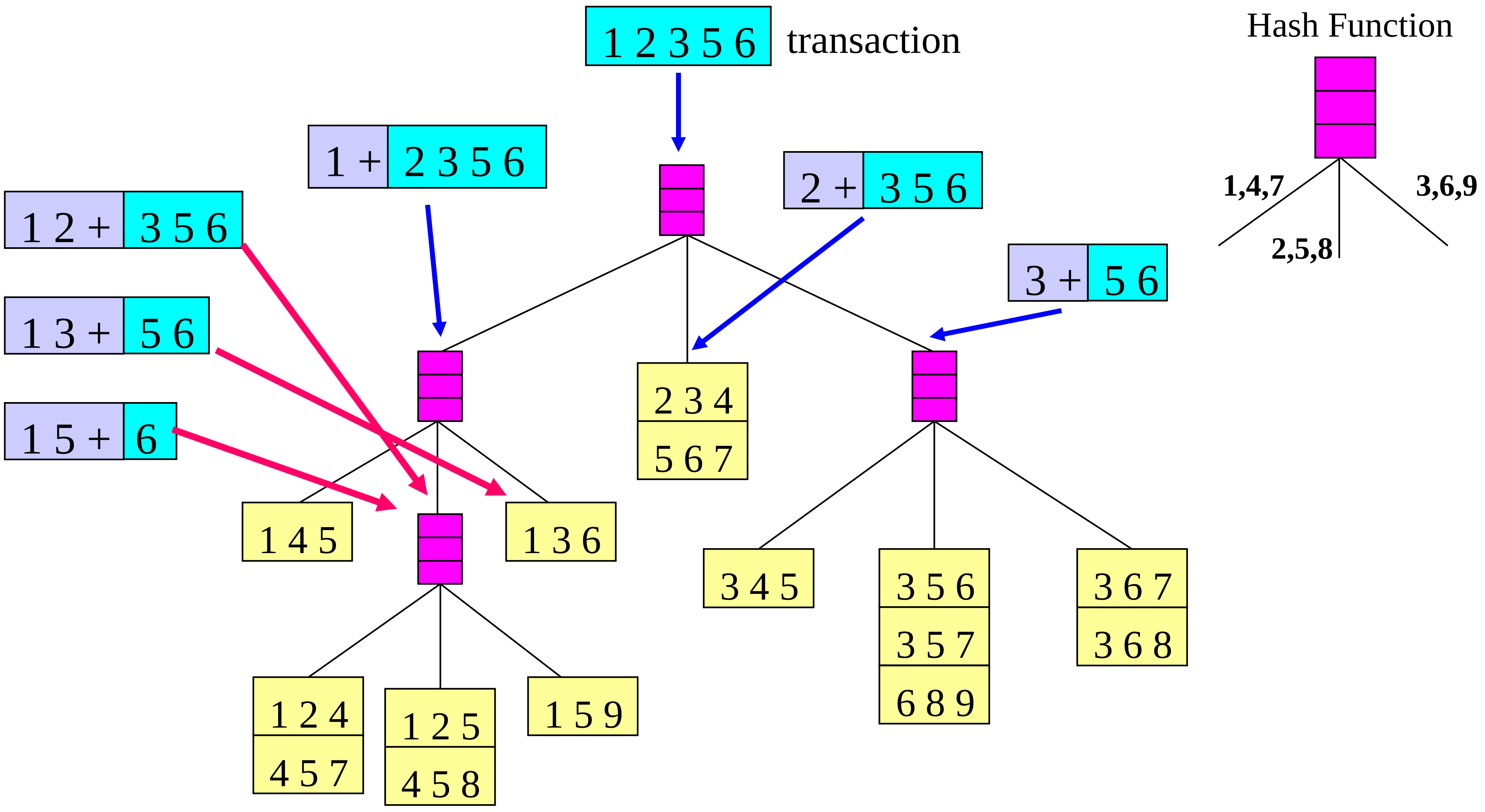
Given a transaction t, what are the possible subsets of size 3?



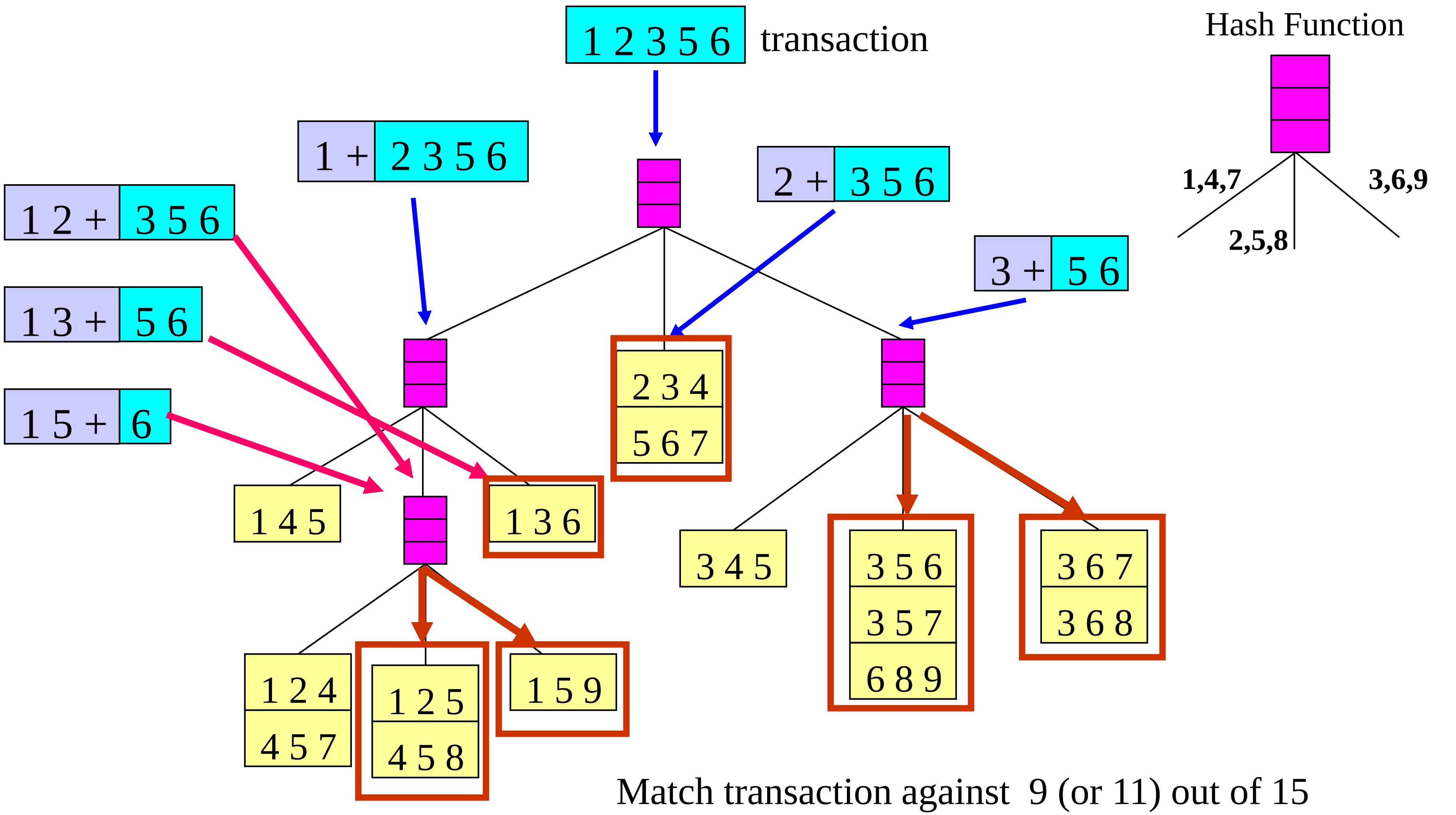
Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Match transaction against 9 (or 11) out of 15 candidates

Visit 5 (or 6) out of 9 leaf nodes

Bottlenecks of Apriori

- Candidate generation can result in huge candidate sets:
  - 10^4 frequent 1-itemset will generate 10^7 candidate 2-itemsets (how?)-*find the causes*
  - To discover a frequent pattern of size 100, e.g., $\{a_1, a_2, \ldots, a_{100}\}$, one needs to generate $2^{100} \sim 10^{30}$ candidates
- Multiple scans of database:
  - Needs (n +1) scans, n is the length of the longest pattern (?)-*find the reason*

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

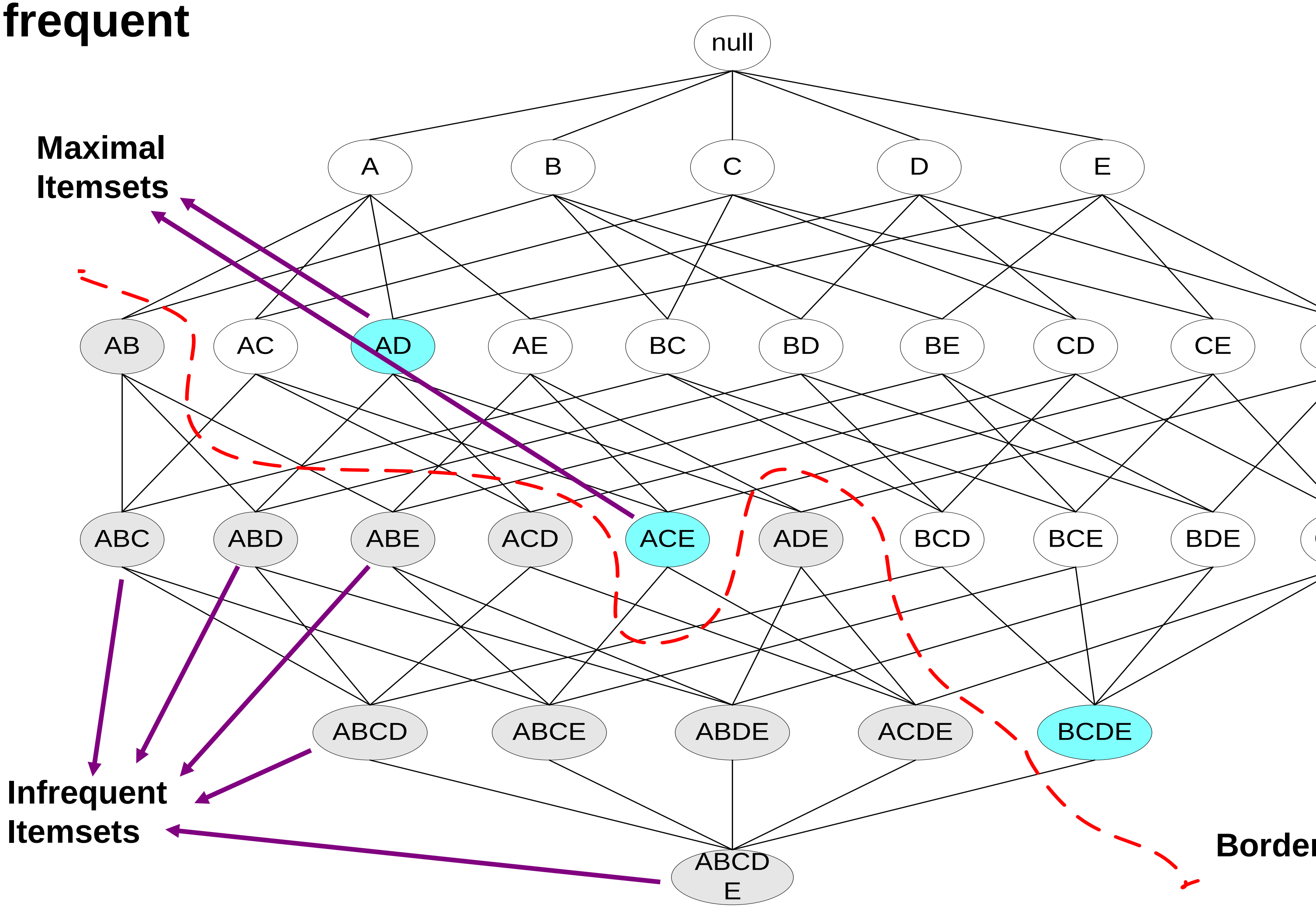
TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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**	0	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**	0	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**	0	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**	0	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**	0	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

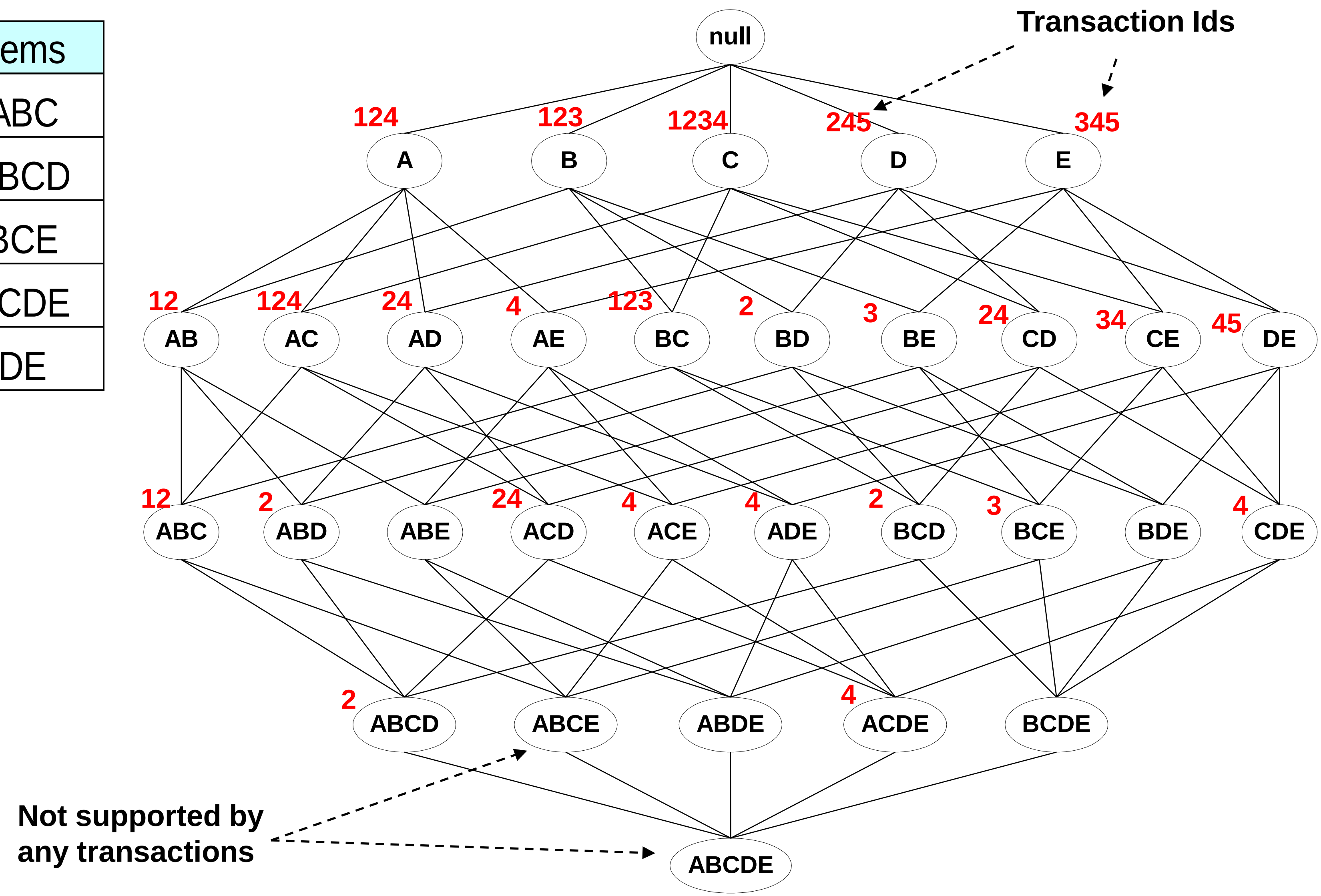
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

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

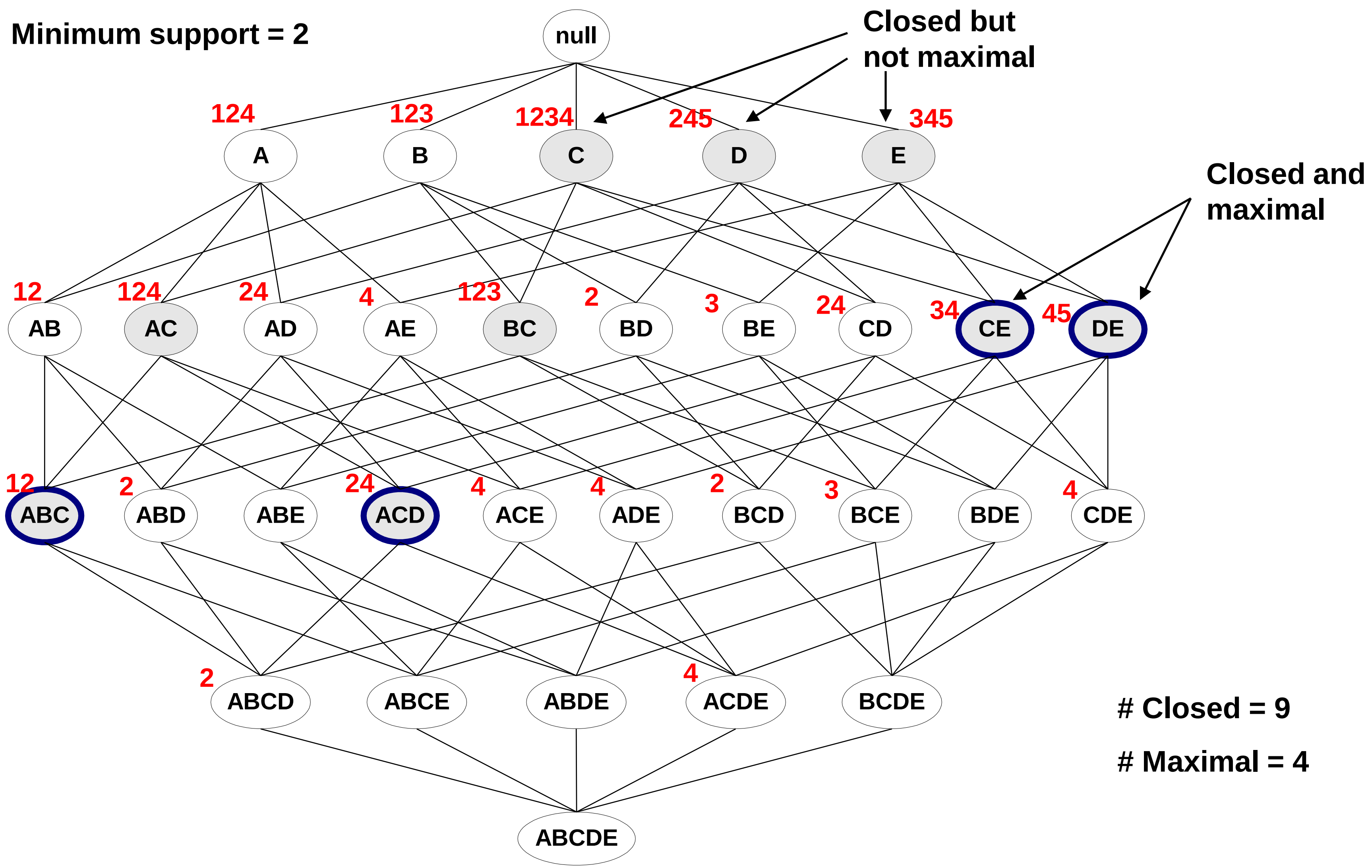
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Maximal vs Closed Itemsets

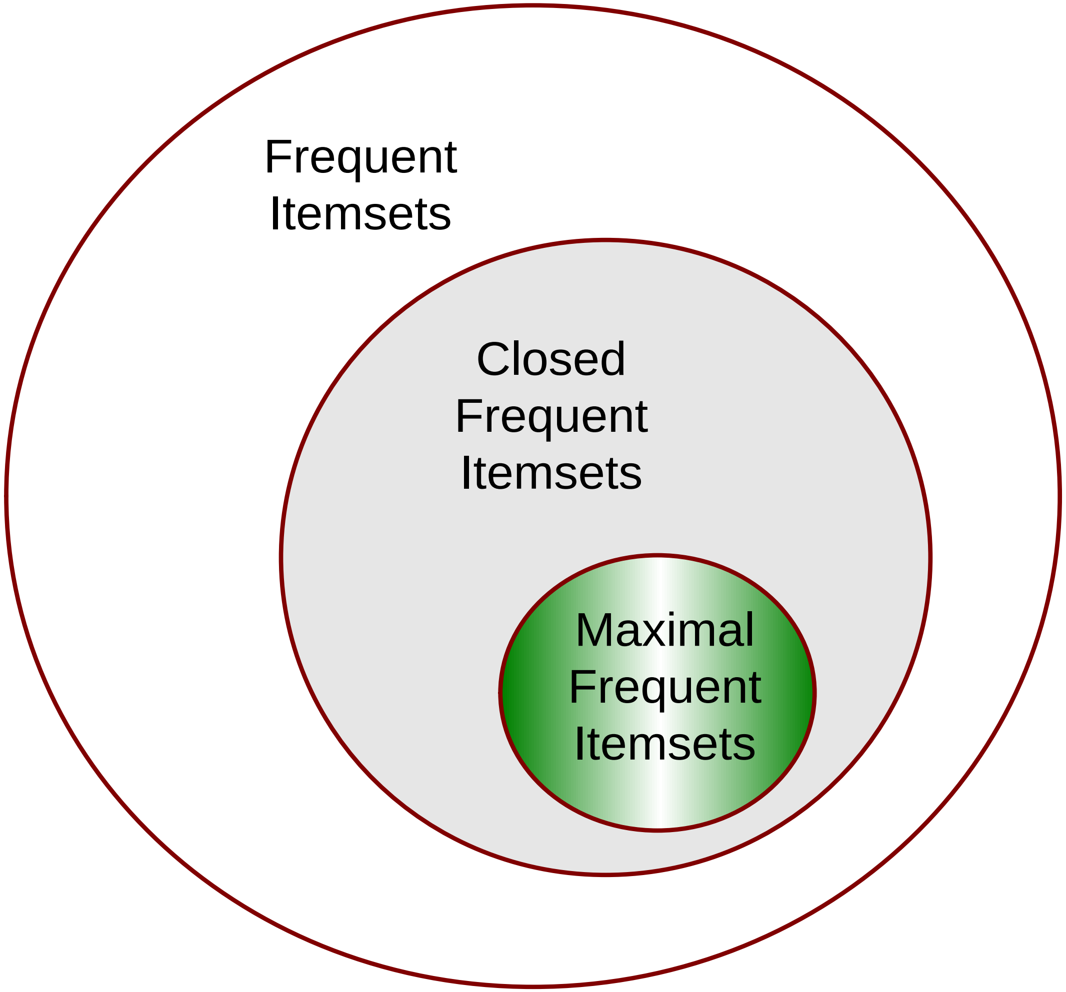
TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Maximal vs Closed Frequent Itemsets

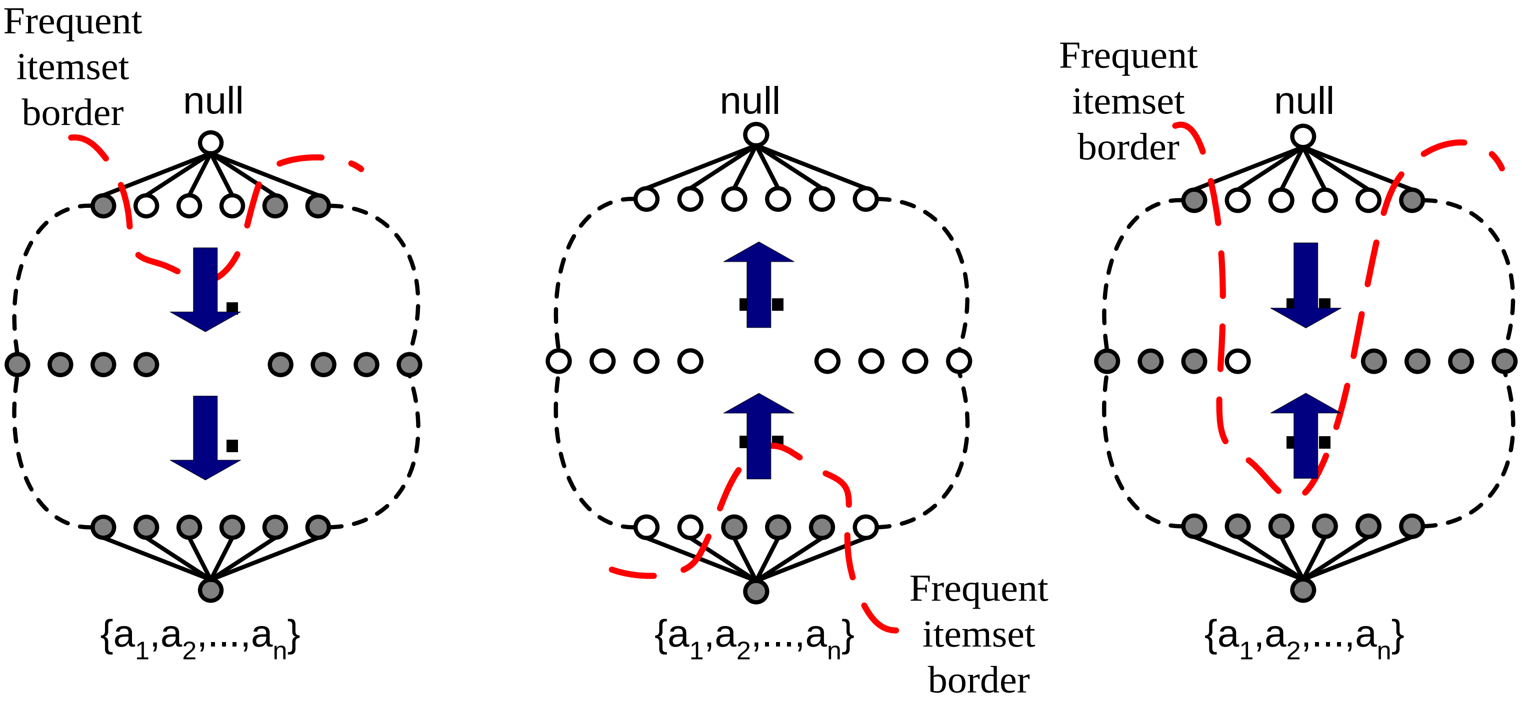


Maximal vs Closed Itemsets



Alternative Methods for Frequent Itemset Generation

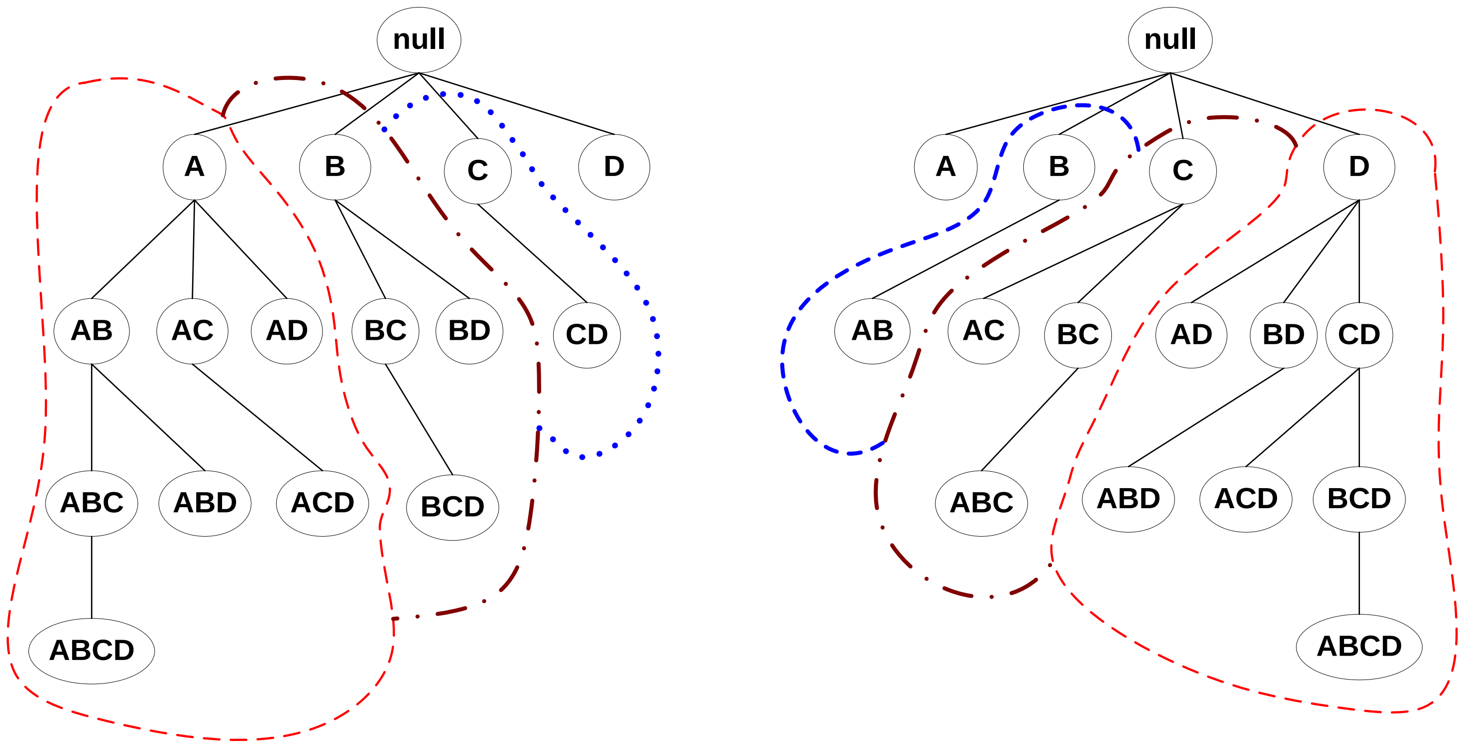
- Traversal of Itemset Lattice
  - General-to-specific vs Specific-to-general



(a) General-to-specific (b) Specific-to-general (c) Bidirectional

Alternative Methods for Frequent Itemset Generation

- Traversal of Itemset Lattice
  - Equivalent Classes (two itemsets belong to the same class if they share same common prefix or suffix)

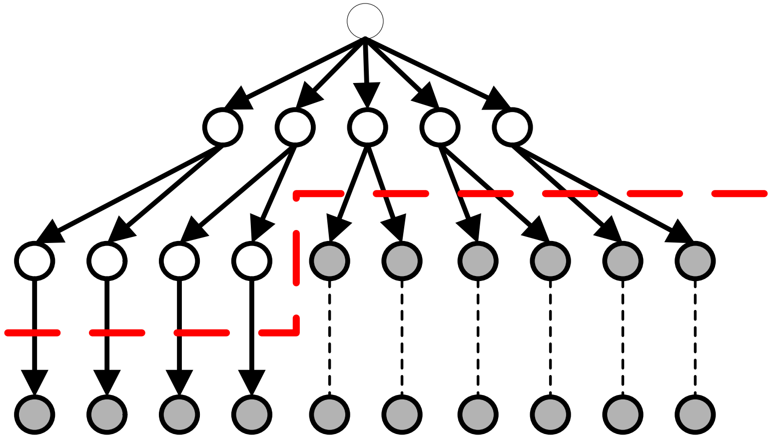


(a) Prefix tree

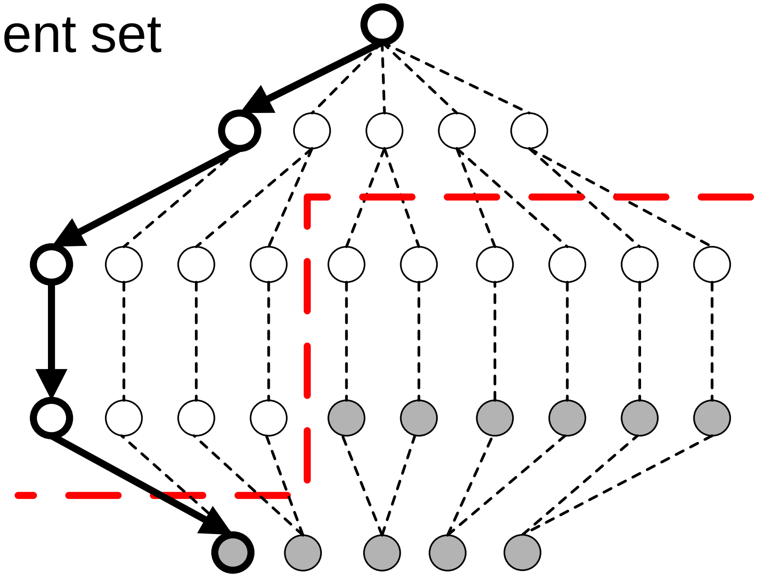
(b) Suffix tree

Alternative Methods for Frequent Itemset Generation

- Traversal of Itemset Lattice
  - Breadth-first vs Depth-first
  - Apriori traverses in BFS manner
  - DFS quickly finds maximal frequent set



(a) Breadth first



(b) Depth first

ECLAT: Another Method for Frequent Itemset Generation

- ECLAT: for each item, store a list of transaction ids (tids); 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	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

TID-list

ECLAT: Another Method for Frequent Itemset Generation

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

A
1
4
5
6
7
8
9

$\wedge$

B
1
2
5
7
8
10

$\rightarrow$

AB
1
5
7
8

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

FP-growth: Another Method for Frequent Itemset Generation

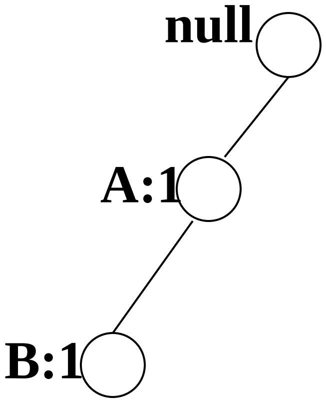
- Use a compressed representation of the database using an FP-tree

- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets

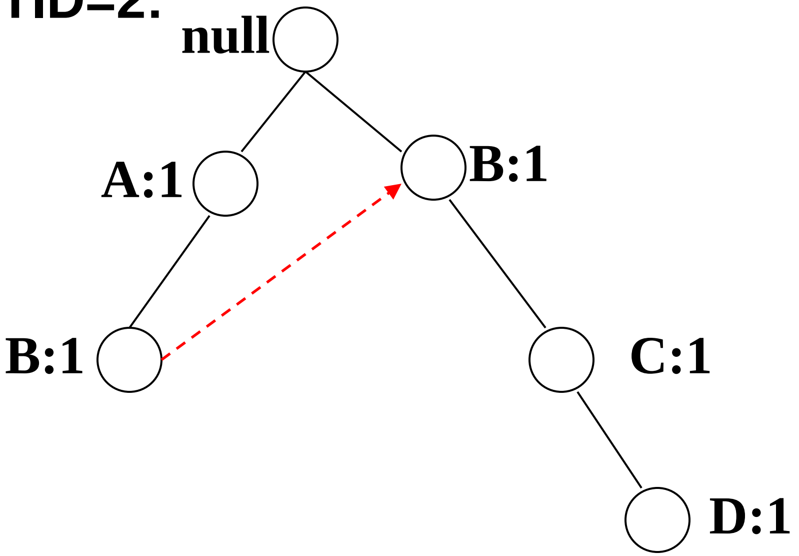
FP-Tree Construction

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	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

After reading TID=1:



After reading TID=2:



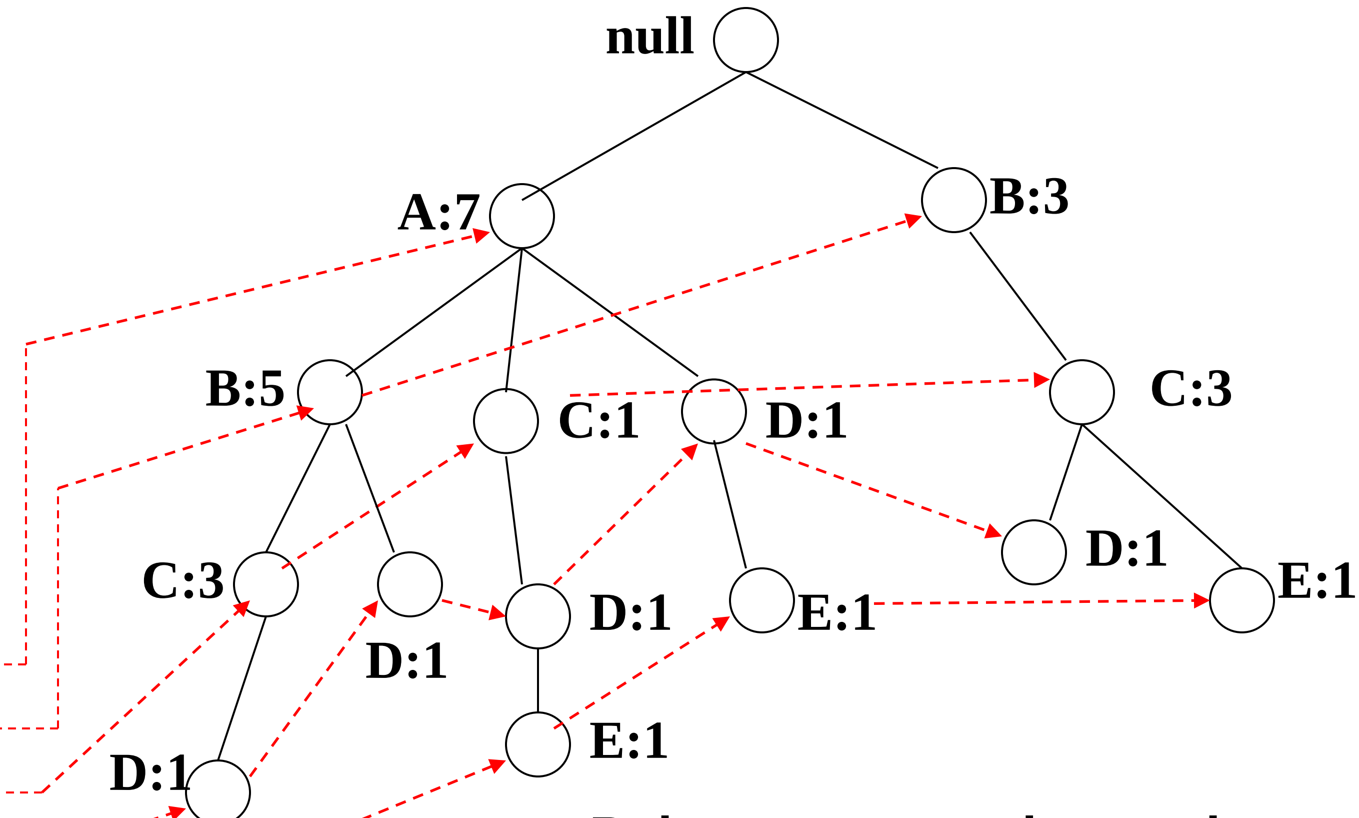
FP-Tree Construction

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

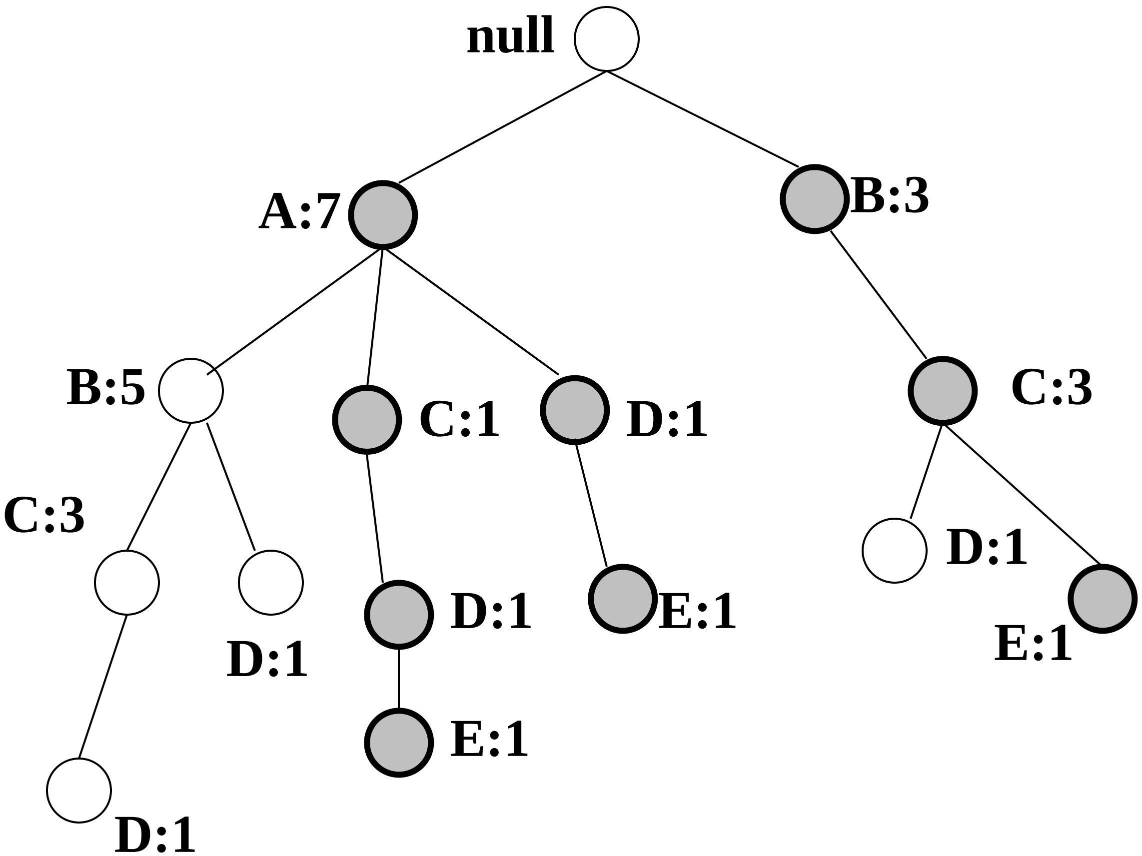
Header table

Item	Pointer
A	
B	
C	
D	
E	



Pointers are used to assist frequent itemset generation

FP-growth

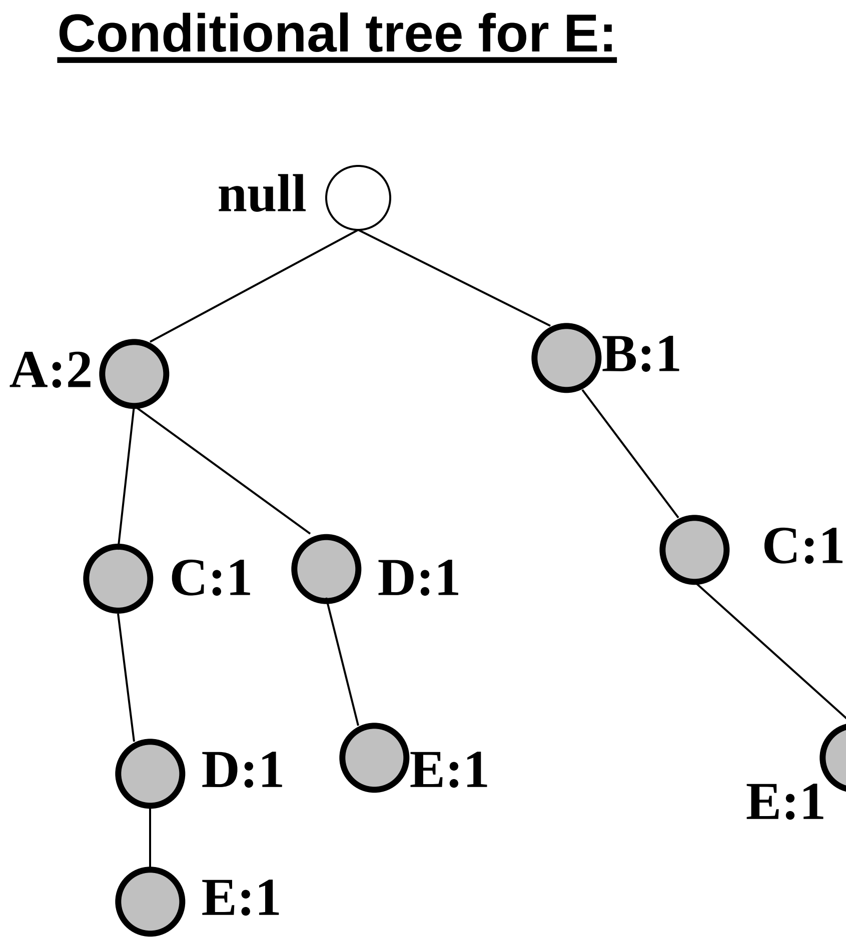


Build conditional pattern base for E:

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

Recursively apply FP-growth on P

FP-growth



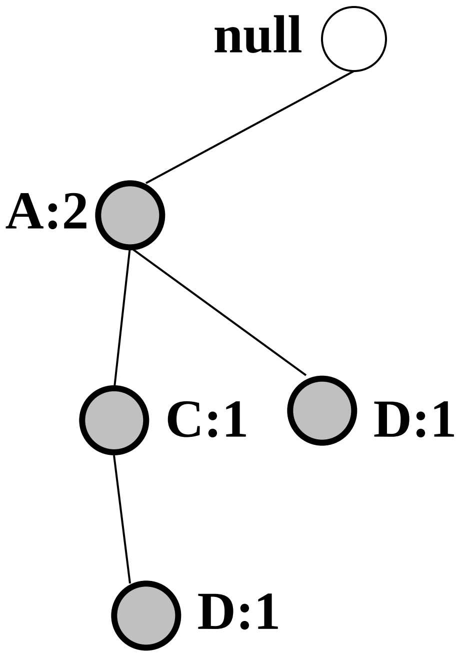
Conditional Pattern base for E:
P = {(A:1,C:1,D:1,E:1),
(A:1,D:1,E:1),
(B:1,C:1,E:1)}

Count for E is 3: {E} is frequent itemset

Recursively apply FP-growth on P

FP-growth

Conditional tree for D within conditional tree for E:



Conditional pattern base for D within conditional base for E:

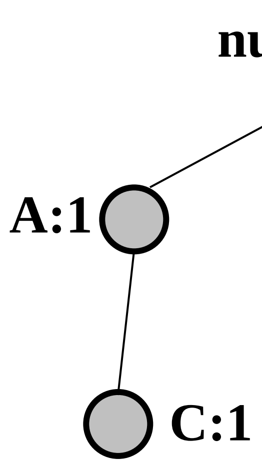
P = {(A:1,C:1,D:1),
(A:1,D:1)}

Count for D is 2: {D,E} is frequent itemset

Recursively apply FP-growth on P

FP-growth

<u>Conditional tree for C within D within E:</u>

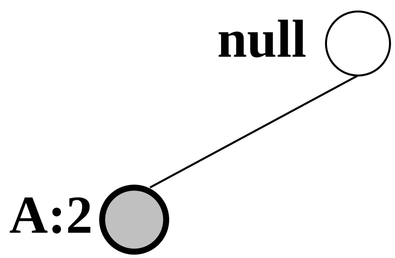


Conditional pattern base for C within D within E:
P = {(A:1,C:1)}

Count for C is 1: {C,D,E} is NOT frequent itemset

FP-growth

<u>Conditional tree for A within D within E:</u>



Count for A is 2: {A,D,E} is frequent itemset

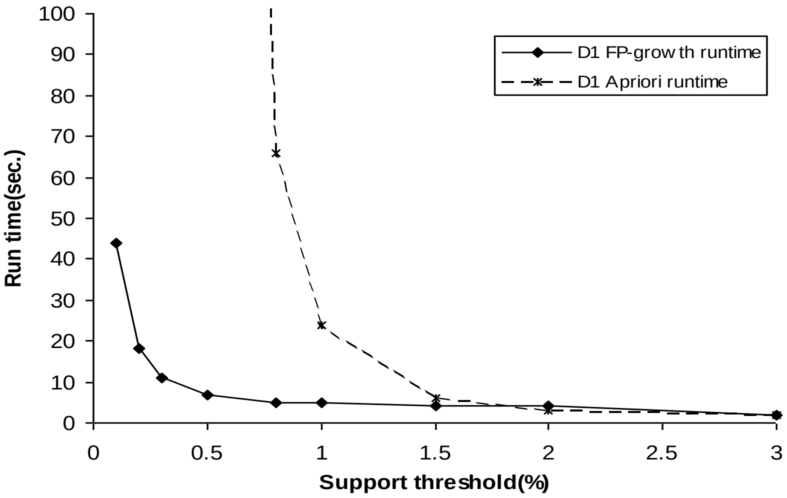
Next step:

Construct conditional tree C within conditional tree E

Continue until exploring conditional tree for A (which has only node A)

Benefits of the FP-tree Structure

- Performance study shows
  - FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection
- Reasoning
  - No candidate generation, no candidate test
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operation is counting and FP-tree building

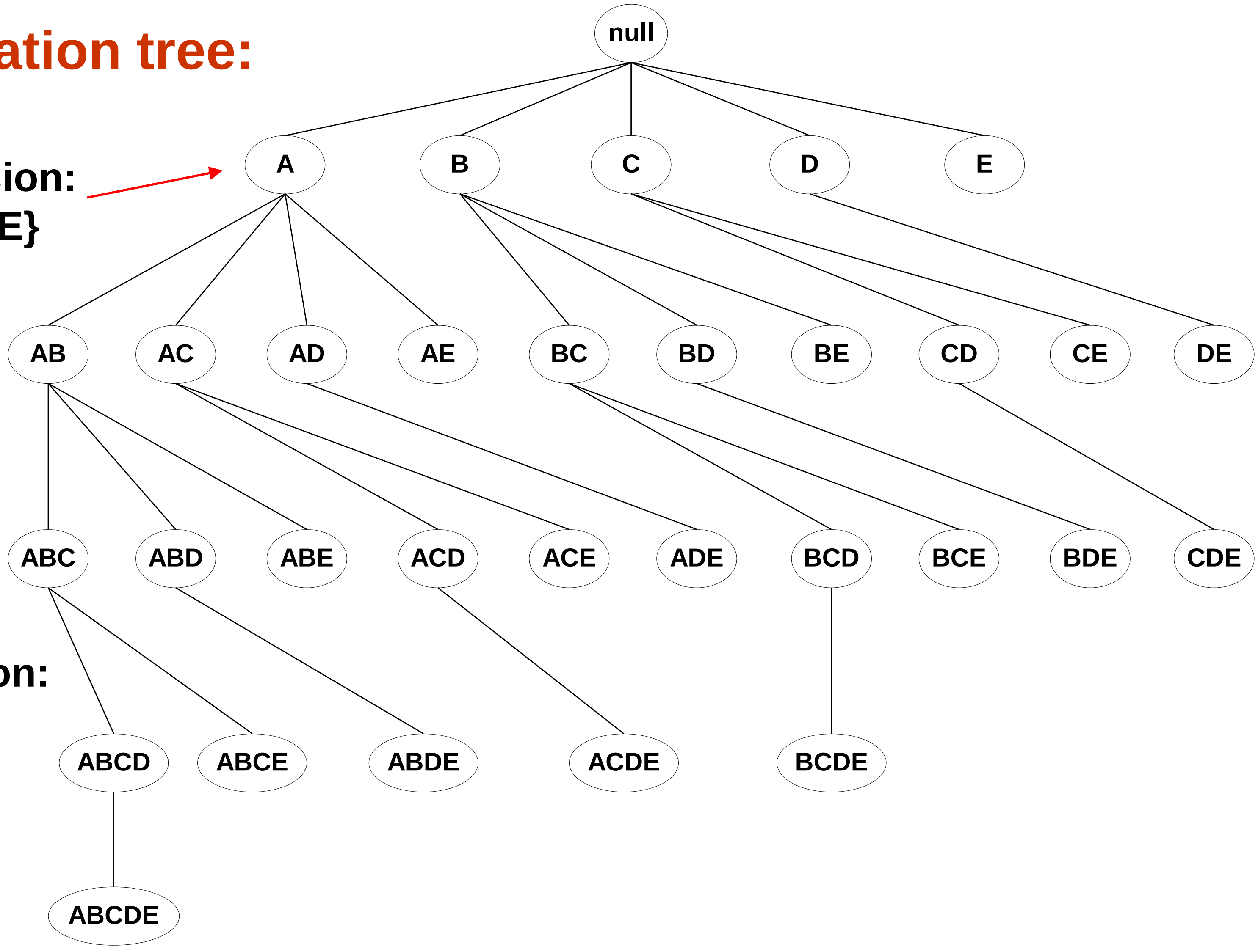


Tree Projection

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 $T \cap E(A)$

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

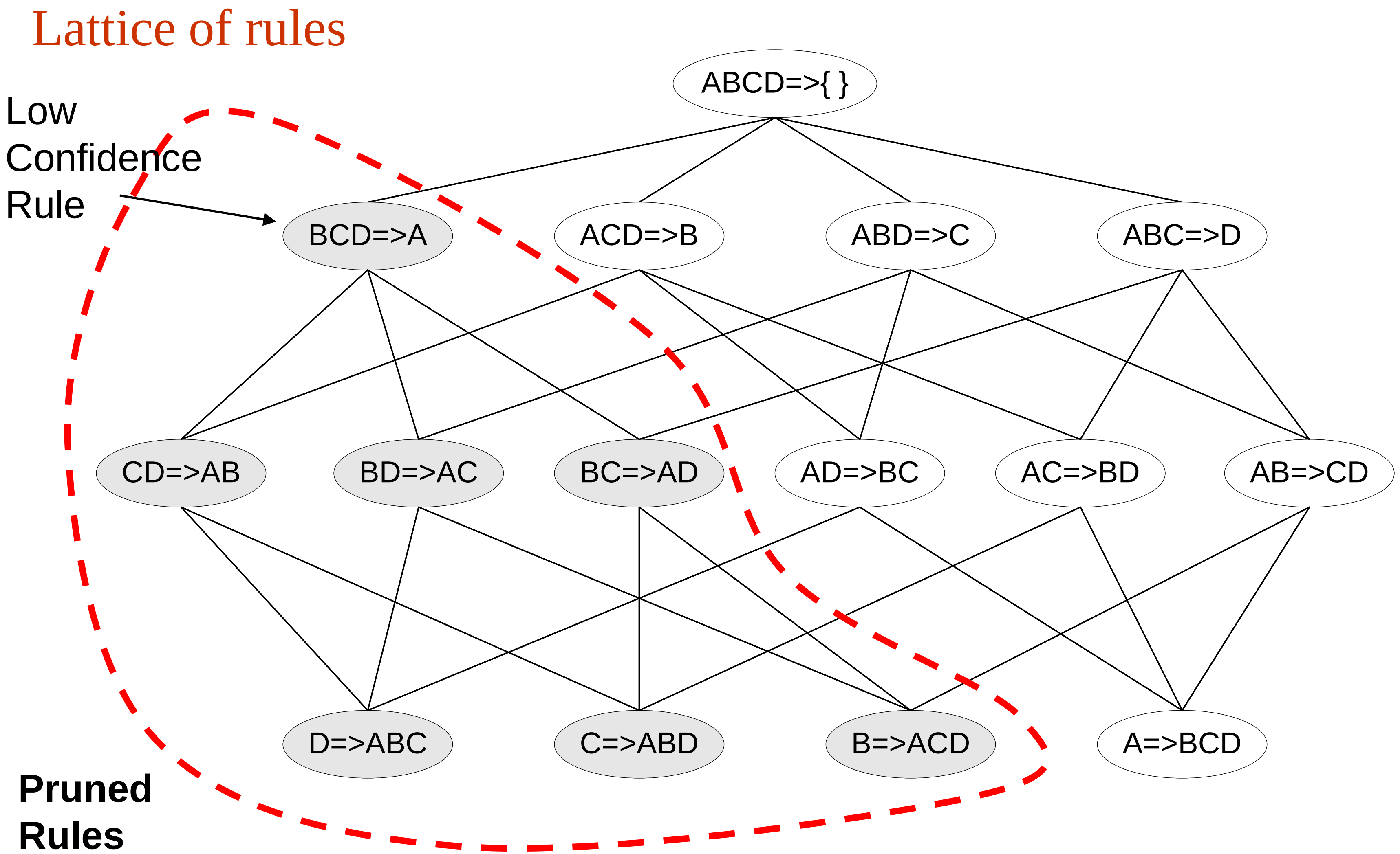
    $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., $L = \{A,B,C,D\}$:

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

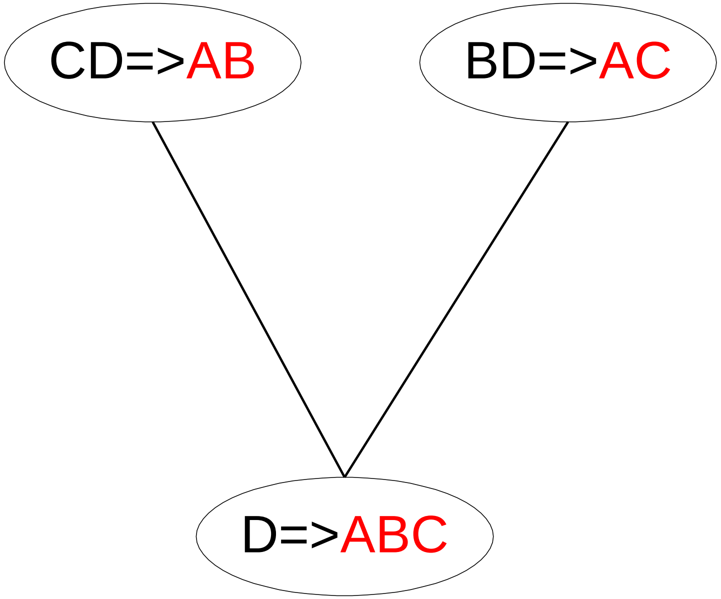
- Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule Generation for Apriori Algorithm



Rule Generation for Apriori Algorithm

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent



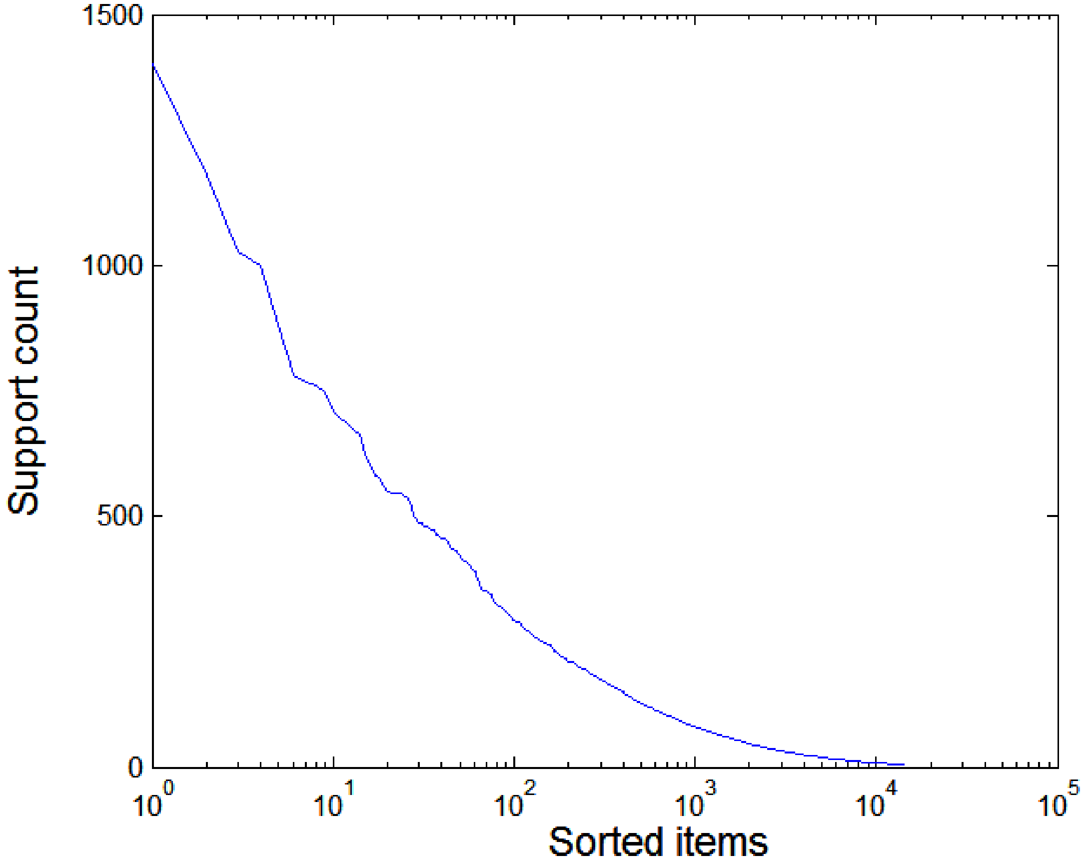
- join(CD=>AB,BD=>AC) would produce the candidate rule D => ABC

- Prune rule D=>ABC if its subset AD=>BC does not have high confidence (i.e. *confidence below threshold*)

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

Problems with the association mining

- Single minsup: It assumes that all items in the data are of the same nature and/or have similar frequencies

- Not true: In many applications, some items appear very frequently in the data, while others rarely appear

  E.g., in a supermarket, people buy *food processor* and *cooking pan* much less frequently than they buy *bread* and *milk*

Rare Item Problem

- If the frequencies of items vary a great deal, we will encounter two problems
  - If minsup is set too high, those rules that involve rare items will not be found
  - To find rules that involve both frequent and rare items, minsup has to be set very low
    - May cause combinatorial explosion because those frequent items will be associated with one another in all possible ways

Multiple minsups model

- Minimum support of a rule is expressed in terms of *minimum item supports* (MIS) of the items that appear in the rule

- Each item can have a minimum item support

- By providing different MIS values for different items, the user effectively expresses different support requirements for different rules

Minsup of a rule

- Let MIS(*i*) be the MIS value of item *i*. The *minsup* of a rule *R* is the lowest MIS value of the items in the rule
  - i.e., a rule *R*: $a_1, a_2, \ldots, a_k \rightarrow a_{k+1}, \ldots, a_r$ satisfies its minimum support if its actual support is $\geq$

    $\min(\mathrm{MIS}(a_1), \mathrm{MIS}(a_2), \ldots, \mathrm{MIS}(a_r))$.

An Example

- Consider the following items:

  bread, *shoes*, *clothes*

  The user-specified MIS values are as follows:

  MIS(*bread*) = 2%   MIS(*shoes*) = 0.1%

  MIS(*clothes*) = 0.2%

  The following rule doesn't satisfy its minsup:

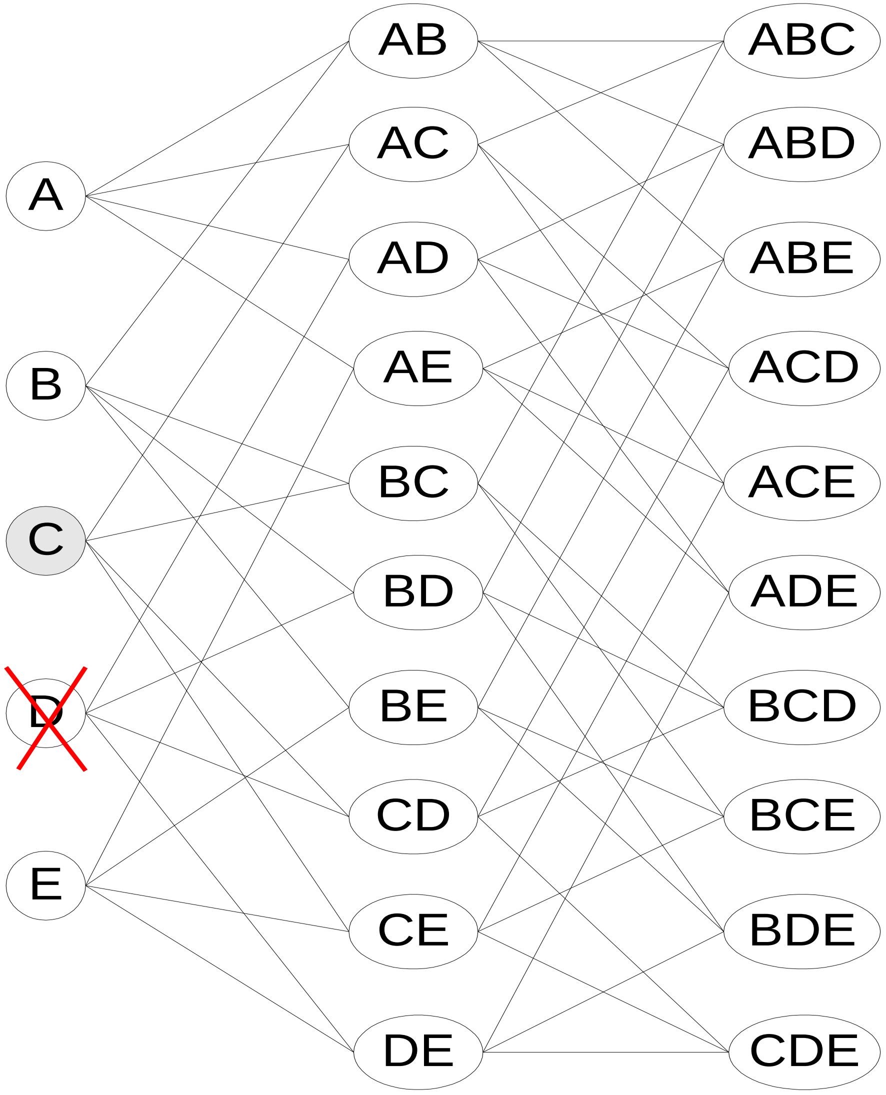
  clothes $\rightarrow$ *bread* [sup=0.15%,conf =70%]

  The following rule satisfies its minsup:

  clothes $\rightarrow$ *shoes* [sup=0.15%,conf =70%]

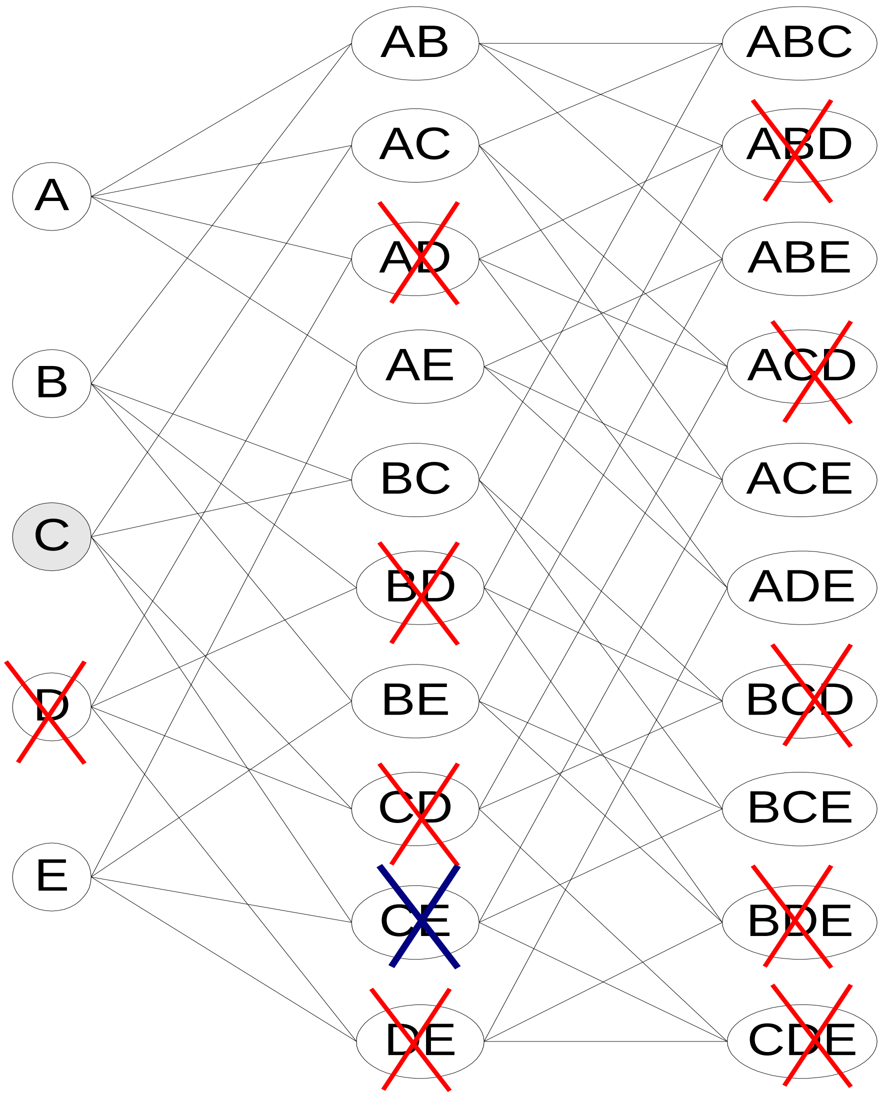
Multiple Minimum Support

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%



Downward closure property (or, anti-monotone property)

- In the new model, the property no longer holds (?)

E.g., Consider four items 1, 2, 3 and 4 in a database. Their minimum item supports are

MIS(1) = 10%   MIS(2) = 20%

MIS(3) = 5%   MIS(4) = 6%

{1, 2} with support 9% is infrequent, but {1, 2, 3} and {1, 2, 4} could be frequent.

To deal with the problem

- We sort all items in I according to their MIS values (make it a total order)
- Order is used throughout the algorithm in each itemset
- Each itemset w is of the following form:

  $\{w[1], w[2], \ldots, w[k]\}$, consisting of items,

  $w[1], w[2], \ldots, w[k]$,

  where $\text{MIS}(w[1]) \leq \text{MIS}(w[2]) \leq \ldots \leq \text{MIS}(w[k])$.

Multiple Minimum Support (Liu 1999)

- Order the items according to their minimum support (in ascending order)
  - e.g.: MS(Milk)=5%, MS(Coke) = 3%, MS(Broccoli)=0.1%, MS(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
    - Candidate is pruned if it contains any infrequent subsets of size k
  - Modify Pruning step:
    - 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.

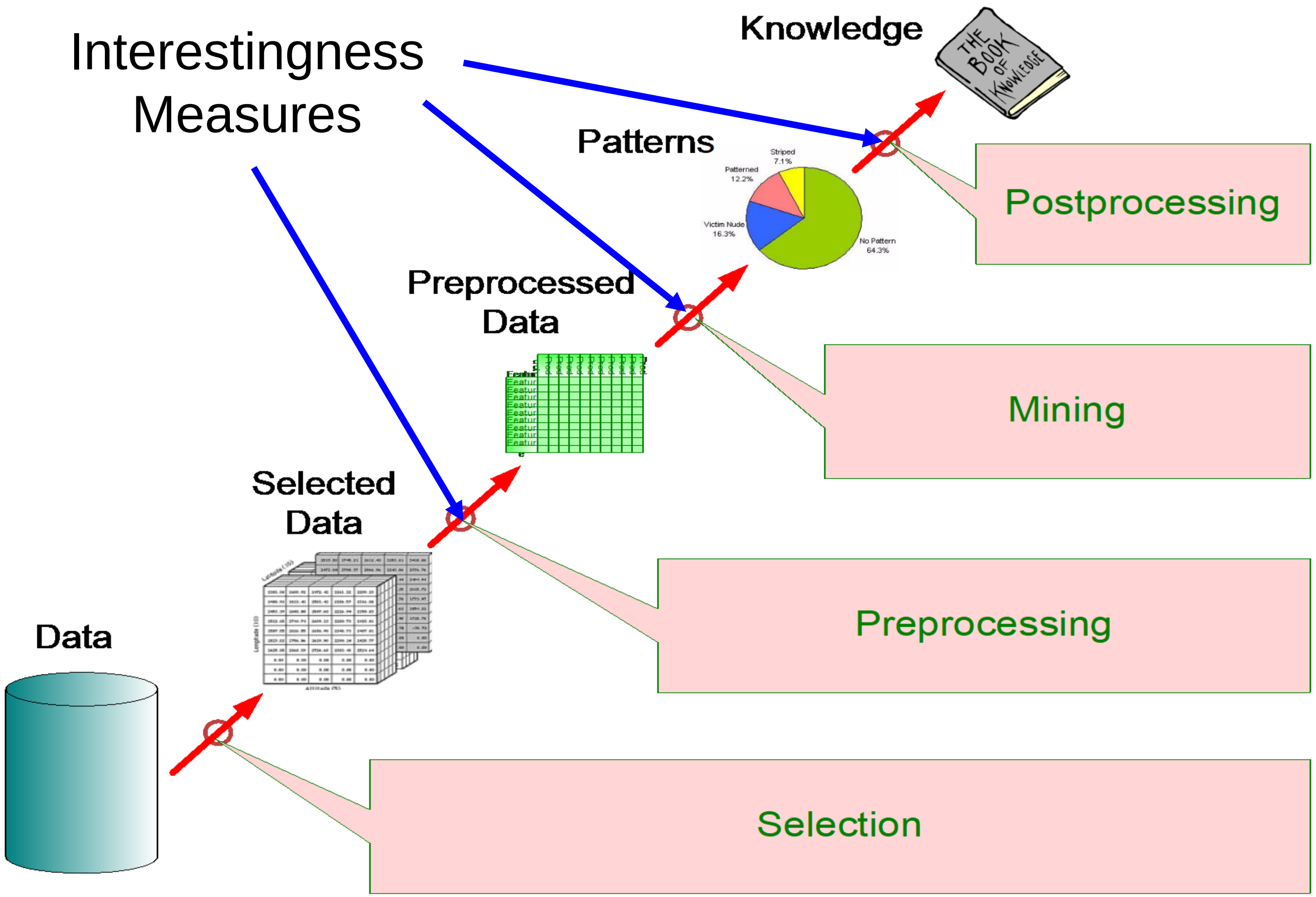
Pattern Evaluation

- Association rule algorithms tend to produce too many rules
  - many of them are uninteresting or redundant
  - redundant if $\{A,B,C\} \rightarrow \{D\}$ and $\{A,B\} \rightarrow \{D\}$ have same support & confidence

- *Interestingness* measures can be used to prune/rank the derived patterns

- In the original formulation of association rules, *support* & *confidence* are the only measures used

Application of Interestingness Measure



Computing Interestingness Measure

- Given a rule $X \rightarrow Y$, information needed to compute rule interestingness can be obtained from a contingency table

Contingency table for $X \rightarrow Y$

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	$\|T\|$

f_{11}: support of X and Y
f_{10}: support of X and $\overline{Y}$
f_{01}: support of $\overline{X}$ and Y
f_{00}: support of $\overline{X}$ and $\overline{Y}$

Used to define various measures

- support, confidence, lift, Gini, J-measure, etc.

Drawback of Confidence

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75 and Support= 0.15

but P(Coffee) = 0.9

$\Rightarrow$ Although confidence is high, rule is misleading

$\Rightarrow$Fraction of tea drinkers who drink coffee is actually less than the overall fraction of people who actually drink coffee

Drawbacks of Confidence

- Measure ignores the support of the itemset in consequent

- With the support of coffee drinkers
  - Many people who drink tea also drink coffee

Statistical Independence

- Population of 1000 students
  - 600 students know how to swim (S)
  - 700 students know how to bike (B)
  - 420 students know how to swim and bike (S,B)

  - $P(S \wedge B) = 420/1000 = 0.42$
  - $P(S) \times P(B) = 0.6 \times 0.7 = 0.42$

  - $P(S \wedge B) = P(S) \times P(B)$ => Statistical independence
  - $P(S \wedge B) > P(S) \times P(B)$ => Positively correlated
  - $P(S \wedge B) < P(S) \times P(B)$ => Negatively correlated

Statistical-based Measures

- Measures that take into account statistical dependence

$$Lift = \frac{P(Y \mid X)}{P(Y)}$$

$$Interest = \frac{P(X,Y)}{P(X)P(Y)}$$

$$\phi - coefficient = \frac{P(X,Y) - P(X)P(Y)}{\sqrt{P(X)[1-P(X)]P(Y)[1-P(Y)]}}$$

Example: Lift/Interest

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

$\Rightarrow$ Lift = 0.75/0.9= 0.8333 (< 1, therefore is negatively associated)

Drawback of Interest

	Q	$\overline{Q}$	
P	880	50	930
$\overline{P}$	50	20	70
	930	70	1000

	Y	$\overline{Y}$	
X	20	50	70
$\overline{X}$	50	880	930
	70	930	1000

$$Interest = \frac{0.88}{(0.93)(0.93)} = 1.02$$

$$Interest = \frac{0.02}{(0.07)(0.07)} = 4.08$$

Statistical independence:

If P(X,Y)=P(X)P(Y) => Interest = 1

Drawback of Lift & Interest

- P and Q appear together 88% of the time
  - interest factor is close to 1 (possible only when P and Q are statistically independent)
- Interest factor of X and Y is higher than P and Q even though X and Y seldom appear together

- Let us consider the confidence values:

C (P→Q)= 94.6%

C (x→y)= 28.6%

better measure

There are lots of measures proposed in the literature

Some measures are good for certain applications, but not for others

What criteria should we use to determine whether a measure is good or bad?

What about Apriori-style support based pruning? How does it affect these measures?

#	Measure	Formula
1	ϕ-coefficient	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
2	Goodman-Kruskal's (λ)	$\frac{\sum_j \max_k P(A_j,B_k)+\sum_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-\max_k P(B_k)}$
3	Odds ratio (α)	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(A,\overline{B})P(\overline{A},B)}$
4	Yule's Q	$\frac{P(A,B)P(\overline{AB})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{AB})+P(A,\overline{B})P(\overline{A},B)} = \frac{\alpha-1}{\alpha+1}$
5	Yule's Y	$\frac{\sqrt{P(A,B)P(\overline{AB})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{AB})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}} = \frac{\sqrt{\alpha}-1}{\sqrt{\alpha}+1}$
6	Kappa (κ)	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
7	Mutual Information (M)	$\frac{\sum_i \sum_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i)\log P(A_i), -\sum_j P(B_j)\log P(B_j))}$
8	J-Measure (J)	$\max\Big(P(A,B)\log(\frac{P(B\|A)}{P(B)}) + P(A\overline{B})\log(\frac{P(\overline{B}\|A)}{P(\overline{B})}),$ $P(A,B)\log(\frac{P(A\|B)}{P(A)}) + P(\overline{A}B)\log(\frac{P(\overline{A}\|B)}{P(\overline{A})})\Big)$
9	Gini index (G)	$\max\Big(P(A)[P(B\|A)^2+P(\overline{B}\|A)^2]+P(\overline{A})[P(B\|\overline{A})^2+P(\overline{B}\|\overline{A})^2]$ $-P(B)^2-P(\overline{B})^2,$ $P(B)[P(A\|B)^2+P(\overline{A}\|B)^2]+P(\overline{B})[P(A\|\overline{B})^2+P(\overline{A}\|\overline{B})^2]$ $-P(A)^2-P(\overline{A})^2\Big)$
10	Support (s)	$P(A,B)$
11	Confidence (c)	$\max(P(B\|A), P(A\|B))$
12	Laplace (L)	$\max\Big(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2}\Big)$
13	Conviction (V)	$\max\Big(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})}\Big)$
14	Interest (I)	$\frac{P(A,B)}{P(A)P(B)}$
15	cosine (IS)	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
16	Piatetsky-Shapiro's (PS)	$P(A,B)-P(A)P(B)$
17	Certainty factor (F)	$\max\Big(\frac{P(B\|A)-P(B)}{1-P(B)}, \frac{P(A\|B)-P(A)}{1-P(A)}\Big)$
18	Added Value (AV)	$\max(P(B\|A)-P(B), P(A\|B)-P(A))$
19	Collective strength (S)	$\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{AB})}$
20	Jaccard (ζ)	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
21	Klosgen (K)	$\sqrt{P(A,B)}\max(P(B\|A)-P(B), P(A\|B)-P(A))$

Properties of A Good Measure

- Piatetsky-Shapiro:
  3 properties a good measure M must satisfy:
  - M(A,B) = 0 if A and B are statistically independent
  - M(A,B) increases monotonically with P(A,B) when P(A) and P(B) remain unchanged
  - M(A,B) decreases monotonically with P(A) [or P(B)] when P(A,B) and P(B) [or P(A)] remain unchanged

Property under Variable Permutation

	B	$\overline{B}$
A	p	q
$\overline{A}$	r	s

	A	$\overline{A}$
B	p	r
$\overline{B}$	q	s

Does M(A,B) = M(B,A)?

Symmetric measures:

- support, lift, collective strength, cosine, Jaccard, etc

Asymmetric measures:

- confidence, conviction, Laplace, J-measure, etc

Property under Row/Column Scaling

Grade-Gender Example (Mosteller, 1968):

	Male	Female	
High	2	3	5
Low	1	4	5
	3	7	10

	Male	Female	
High	4	30	34
Low	2	40	42
	6	70	76

2x 10x

Mosteller:

Underlying association should be independent of the relative number of male and female students in the samples

Property under Inversion Operation

	A	B
Transaction 1	1	0
	0	0
	0	0
	0	0
	0	1
	0	0
	0	0
	0	0
	0	0
Transaction N	1	0

(a)

C	D
0	1
1	1
1	1
1	1
1	0
1	1
1	1
1	1
1	1
0	1

(b)

E	F
0	0
1	0
1	0
1	0
1	1
1	0
1	0
1	0
1	0
0	0

(c)

An objective measure M is invariant under the inversion operation if its value remains the same when exchanging the frequency counts f_{11} with f_{00} and f_{10} with f_{01}

Example: ϕ-Coefficient

- ϕ-coefficient is analogous to correlation coefficient for continuous variables

	Y	$\overline{Y}$	
X	60	10	70
$\overline{X}$	10	20	30
	70	30	100

	Y	$\overline{Y}$	
X	20	10	30
$\overline{X}$	10	60	70
	30	70	100

$$\phi = \frac{0.6 - 0.7 \times 0.7}{\sqrt{0.7 \times 0.3 \times 0.7 \times 0.3}} = 0.5238$$

$$\phi = \frac{0.2 - 0.3 \times 0.3}{\sqrt{0.7 \times 0.3 \times 0.7 \times 0.3}} = 0.5238$$

ϕ Coefficient is the same for both tables

Property under Null Addition

	B	$\overline{B}$
A	p	q
$\overline{A}$	r	s

$\Rightarrow$

	B	$\overline{B}$
A	p	q
$\overline{A}$	r	s + k

Invariant measures:

- support, cosine, Jaccard, etc

Non-invariant measures:

- correlation, Gini, mutual information, odds ratio, etc

Different Measures have Different Properties

Symbol	Measure	Range	P1	P2	P3	O1	O2	O3	O3'	O4
Φ	Correlation	-1 … 0 … 1	Yes	Yes	Yes	Yes	No	Yes	Yes	No
λ	Lambda	0 … 1	Yes	No	No	Yes	No	No*	Yes	No
α	Odds ratio	0 … 1 … ∞	Yes*	Yes	Yes	Yes	Yes	Yes*	Yes	No
Q	Yule's Q	-1 … 0 … 1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Y	Yule's Y	-1 … 0 … 1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
κ	Cohen's	-1 … 0 … 1	Yes	Yes	Yes	Yes	No	No	Yes	No
M	Mutual Information	0 … 1	Yes	Yes	Yes	Yes	No	No*	Yes	No
J	J-Measure	0 … 1	Yes	No	No	No	No	No	No	No
G	Gini Index	0 … 1	Yes	No	No	No	No	No*	Yes	No
s	Support	0 … 1	No	Yes	No	Yes	No	No	No	No
c	Confidence	0 … 1	No	Yes	No	Yes	No	No	No	Yes
L	Laplace	0 … 1	No	Yes	No	Yes	No	No	No	No
V	Conviction	0.5 … 1 … ∞	No	Yes	No	Yes**	No	No	Yes	No
I	Interest	0 … 1 … ∞	Yes*	Yes	Yes	Yes	No	No	No	No
IS	IS (cosine)	0 .. 1	No	Yes	Yes	Yes	No	No	No	Yes
PS	Piatetsky-Shapiro's	-0.25 … 0 … 0.25	Yes	Yes	Yes	Yes	No	Yes	Yes	No
F	Certainty factor	-1 … 0 … 1	Yes	Yes	Yes	No	No	No	Yes	No
AV	Added value	0.5 … 1 … 1	Yes	Yes	Yes	No	No	No	No	No
S	Collective strength	0 … 1 … ∞	No	Yes	Yes	Yes	No	Yes*	Yes	No
ζ	Jaccard	0 .. 1	No	Yes	Yes	Yes	No	No	No	Yes
K	Klosgen's	$\left(\sqrt{\frac{2}{\sqrt{3}}}-1\right)\left(2-\sqrt{3}-\frac{1}{\sqrt{3}}\right)\ldots 0 \ldots \frac{2}{3\sqrt{3}}$	Yes	Yes	Yes	No	No	No	No	No

Subjective Interestingness Measure

- Objective measure:
  - Rank patterns based on statistics computed from data
  - e.g., 21 measures of association (support, confidence, Laplace, Gini, mutual information, Jaccard, etc).
- Subjective measure:
  - Rank patterns according to user's interpretation
    - A pattern is subjectively interesting if it contradicts the expectation of a user (Silberschatz & Tuzhilin)
    - A pattern is subjectively interesting if it is actionable (Silberschatz & Tuzhilin)

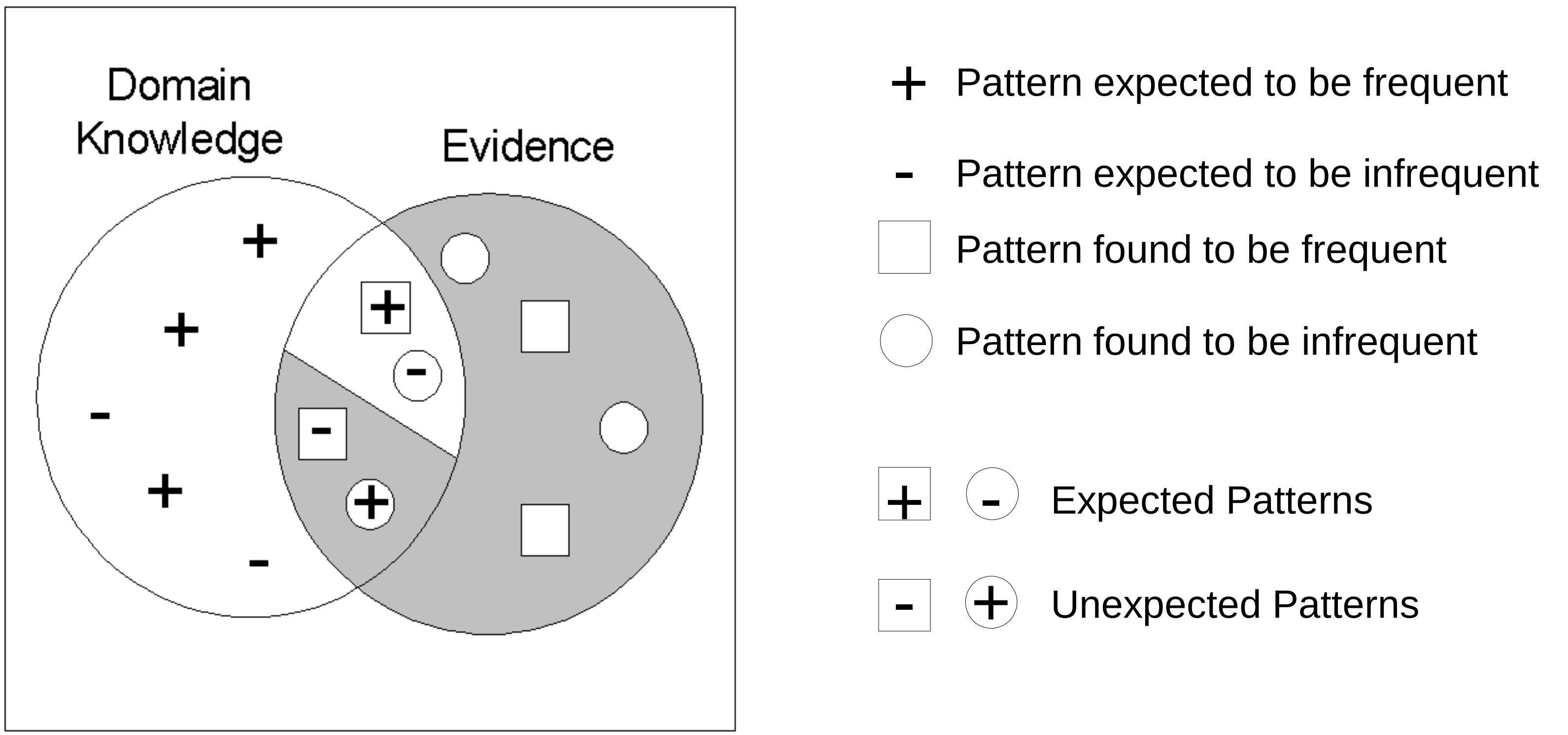
1. IF (used_seat_belt = ‘yes’) THEN (injury = ‘no’)
2. IF ((used_seat_belt = ‘yes’) Λ (passenger = child)) THEN (injury = ‘yes’)

- Rule (1) is a general and an obvious rule
- Rule (2)
  - Contradicts the knowledge represented by rule (1) and so the user's belief

- This kind of knowledge is unexpected from users preset beliefs and it is always interesting to extract this interesting (or surprising) knowledge from data sets
- *Unexpectedness* means knowledge which is unexpected from the beliefs of users
- A decision rule is considered to be interesting (or surprising) if it represents knowledge that was not only previously unknown to the users but also contradicts the original beliefs of the users

Interestingness via Unexpectedness

- Need to model expectation of users (domain knowledge)



- Need to combine expectation of users with evidence from data (i.e., extracted patterns)

Interestingness via Unexpectedness

- Web Data (Cooley et al 2001)
  - Domain knowledge in the form of site structure
  - Given an itemset $F = \{X_1, X_2, \ldots, X_k\}$ (X_i : Web pages)
    - L: number of links connecting the pages
    - lfactor = L / (k $\times$ k-1)
    - cfactor = 1 (if graph is connected), 0 (disconnected graph)
  - Structure evidence = cfactor $\times$ lfactor
  - Usage evidence $= \dfrac{P(X_1 \cap X_2 \cap \ldots \cap X_k)}{P(X_1 \cup X_2 \cup \ldots \cup X_k)}$
  - Use Dempster-Shafer theory to combine domain knowledge and evidence from data