PENNSTATE

Chapter 6-7
Association Analysis

Presentation extended from the slides of the textbook, Introduction to Data Mining by Tan et al. and supplementary material

Association Analysis

- Motivation: find inherent regularities (patterns) in data
  - What products were often purchased together?
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
- Also called *frequent pattern analysis*
  - Freq. pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
  - Association rules may be derived from freq. patterns
- Applications
  - Basket data analysis, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Association Analysis Application

- Basket Data Analysis
  - Goal: To identify items that are bought together by sufficiently many customers.
  - Approach: Process the basket data collected at the check-out counters to find dependencies among items.
  - Association rules (example):
    - If a customer buys diaper and milk, then he is very likely to buy beer.
    - Don't be surprised if you find six-packs stacked next to diapers in the supermarket!

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

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} → {Beer},
{Milk, Bread} → {Eggs,Coke}
{Beer, Bread} → {Milk},

The → here refers to implication due to *co-occurrence*, not causality!

Definition: Frequent Itemset

- Itemset: a set of items. e.g., {Milk, Bread, Diaper}
  - k-itemset: an itemset of k items
- Support count (σ)
  - # of occurrence of an itemset
  - e.g., σ({Milk,Bread,Diaper}) = 2
- Support (s)
  - Fraction of transactions that contain an itemset
  - e.g. s({Milk, Bread, Diaper}) = 2/5
- Frequent Itemset
  - An itemset with support ≥ *minsup* (a threshold)
- Association Rule
  - An implication expression of the form X → Y, where X and Y are itemsets

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

Rule Evaluation Metrics

- Consider an association rule X → Y
- Support (s) measures how interesting the itemset X ∪ Y is in terms of *frequency*
  - Fraction of transactions that contain both X and Y
- Confidence (c) measures how strong the inference (implication) a rule is in terms of *conditional probability p(Y|X)*
  - Fraction of transactions that contain X and Y to those that contain X.
- Example: {Milk, Diaper} → {Beer}

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

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

An Example

TID	Items
100	A, C, D
200	B, C, E
300	A, B, C, E
400	B, E

- Let minimum support count be 2, minimum confidence be 2/3
- Frequent itemsets
  - {A}, {B}, {C}, {E}, {A,C}, {B,C}, {B,E}, {C,E}, {B,C,E}
- Strong rules
  - {B, E}→ {C} (2/3)
  - {C}→{A} (2/3)
  - {A}→{C} (2/2)

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

  ⇒ 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*

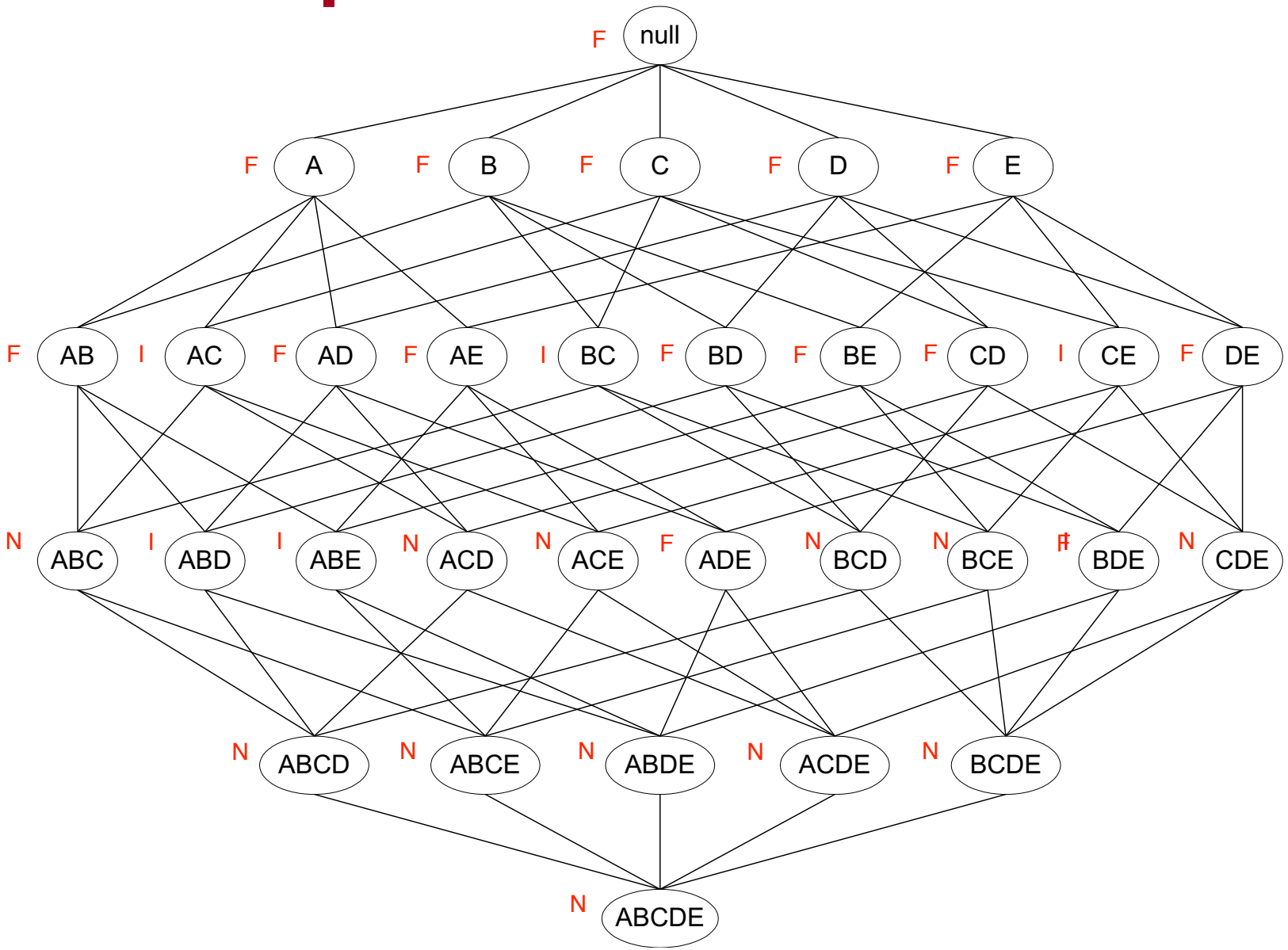
→ *decouple the support and confidence requirements*

Mining Association Rules

- Two-step approach:
  1. *Frequent Itemset Generation*
     - Generate all itemsets whose support ≥ minsup
     - i.e., *Frequent Pattern Mining*
  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

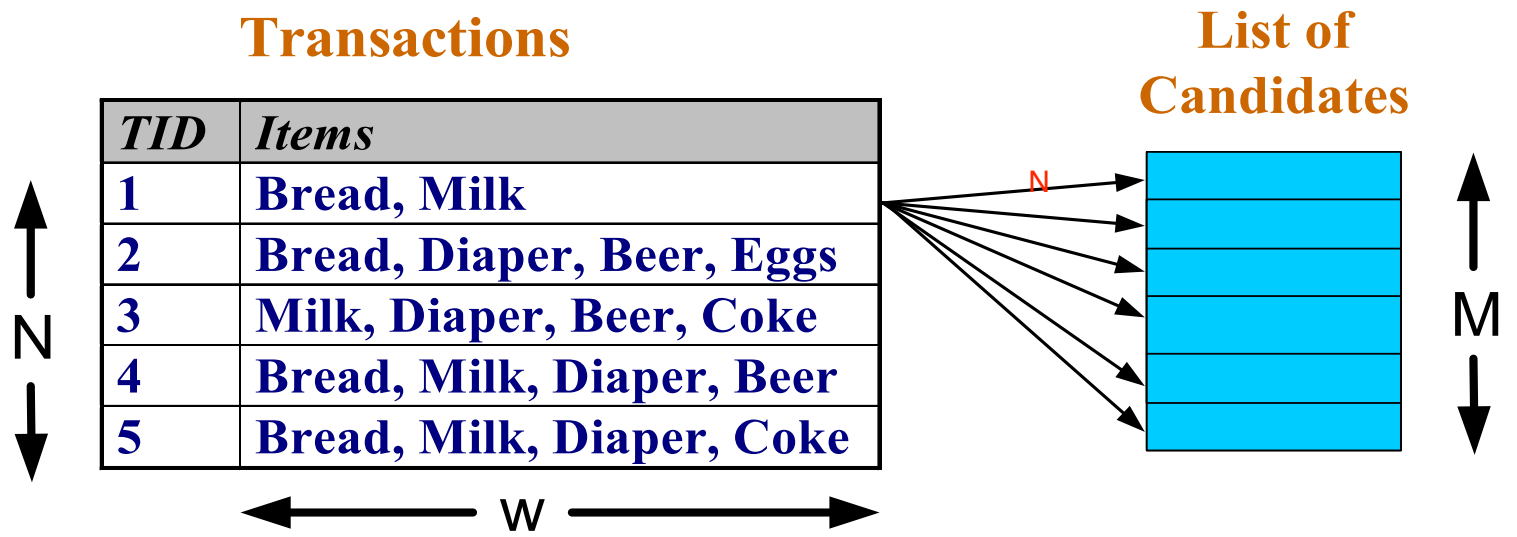
Frequent Itemset Generation



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 transaction database



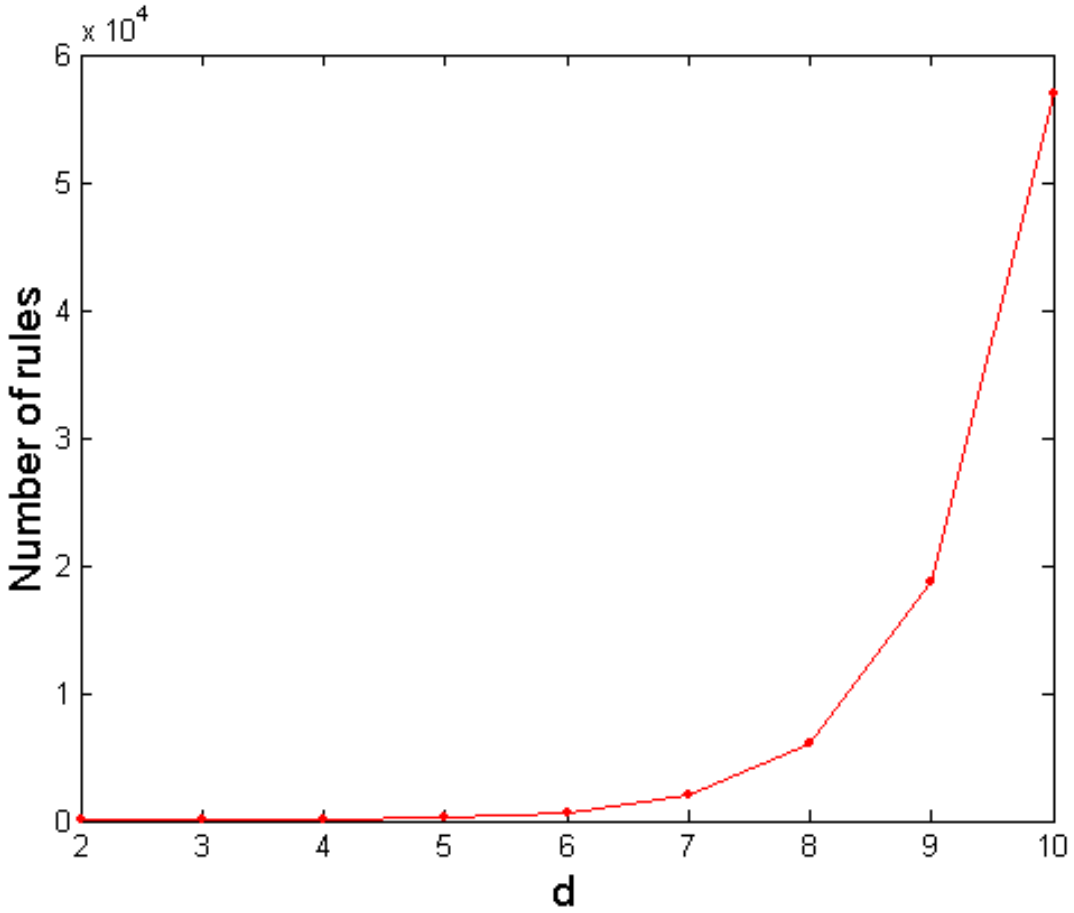
  - Match each trans. against every candidate
  - Complexity ~ O(NMw) => Expensive since M = 2^d !!!
  - *Bottlenecks?*

Challenges

- Huge number of itemset candidates to generate
- Multiple scans of transaction database
- Tedious matching workload of counting support for candidates

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

Strategies for Efficient Frequent Itemset Generation

1. Reduce the *number of* *candidates* (M)
   - Complete search: $M=2^d$
   - Use pruning techniques to reduce M
2. Reduce the *number of* *transactions* (N)
   - Reduce size of N as the size of itemset increases
   - E.g., DHP (Direct Hash and Pruning) and vertical-based mining algorithms
3. 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

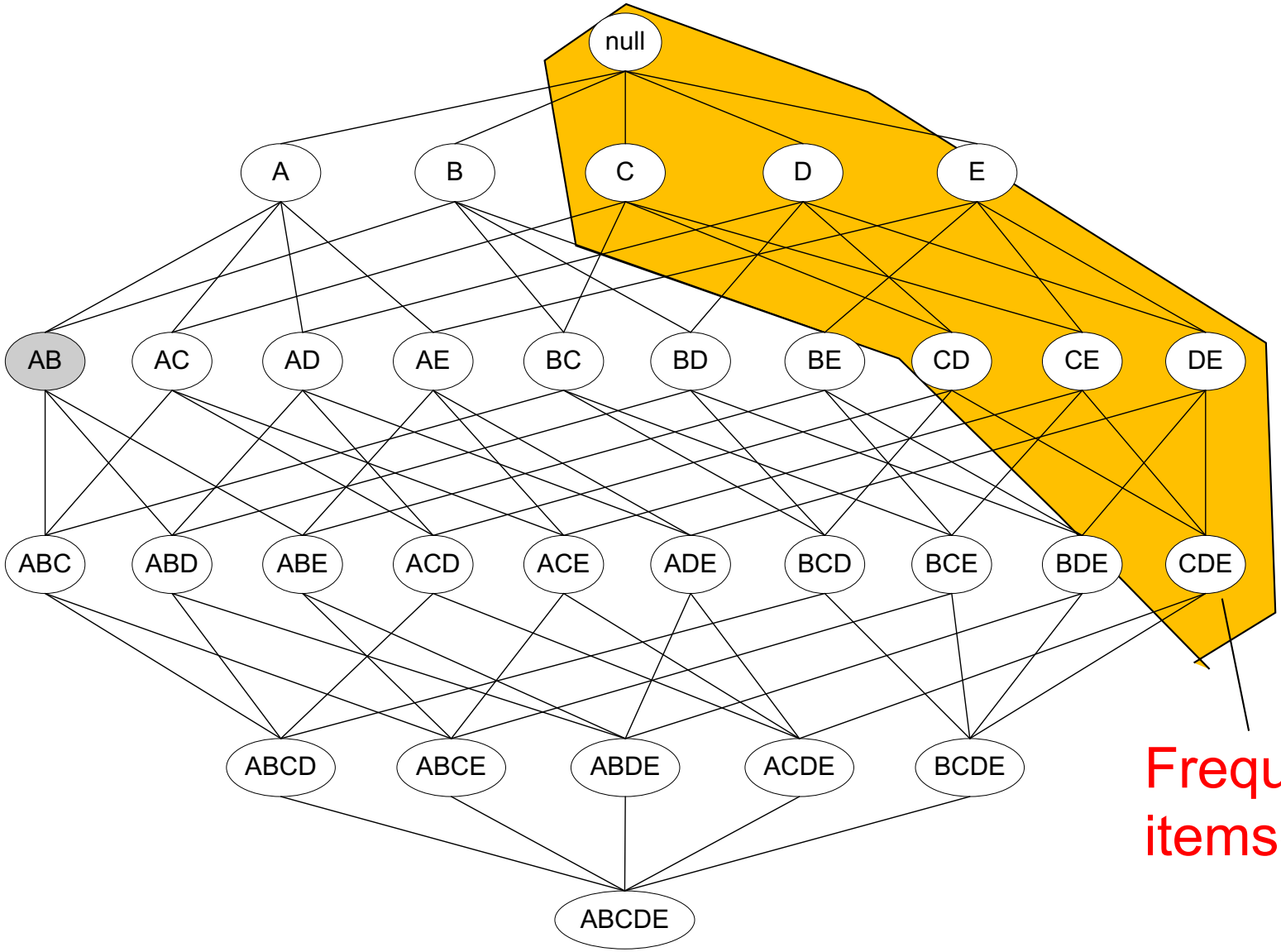
Reducing Number of Candidates

- Apriori algorithm explores strategies (1) and (3)
- *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:

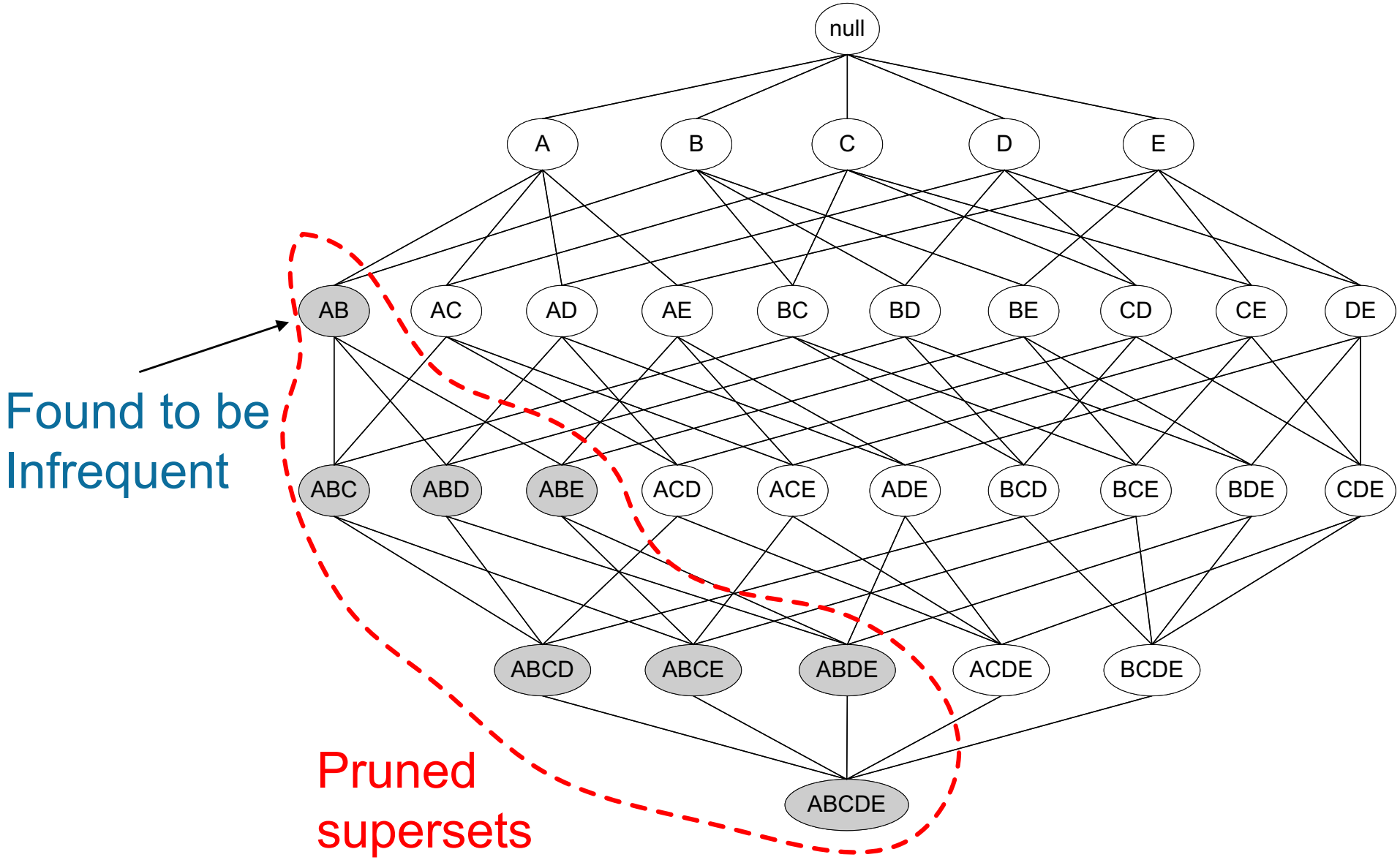
$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

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

Apriori Principle: an Illustration



Apriori Principle: an Illustration



Ideas behind Apriori Algorithm

- Strategy
  - Generate candidate k-itemsets (denoted as C_k)
    - These are possible frequent k-itemsets
  - Check frequent k-itemsets (denoted as L_k)
    - Need to satisfy minimum support
- Level-wise approach
  - Generate candidate itemsets by *lattice levels*
    - (k-1)-itemsets are used to explore k-itemsets
    - $C_k = L_{k-1} \otimes L_{k-1} = \{A \cup B | A, B \in L_{k-1}, |A \cap B| = k-2\}$
  - Prune C_k by Apriori principle
  - Generate L_k by counting support (i.e., scanning transaction DB to match subsets of transactions)

Apriori Algorithm

- Let k=1
- Generate frequent itemsets of length 1
- Repeat until no more 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

Illustration of Apriori Algorithm

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

Minimum Support = 3

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

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3 = 41$$

With support-based pruning,

$$6 + 6 + 1 = 13$$

More Detailed Example

Min. support 50% (i.e., 2 Tx's)
Conf.: 66%

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan

L_3

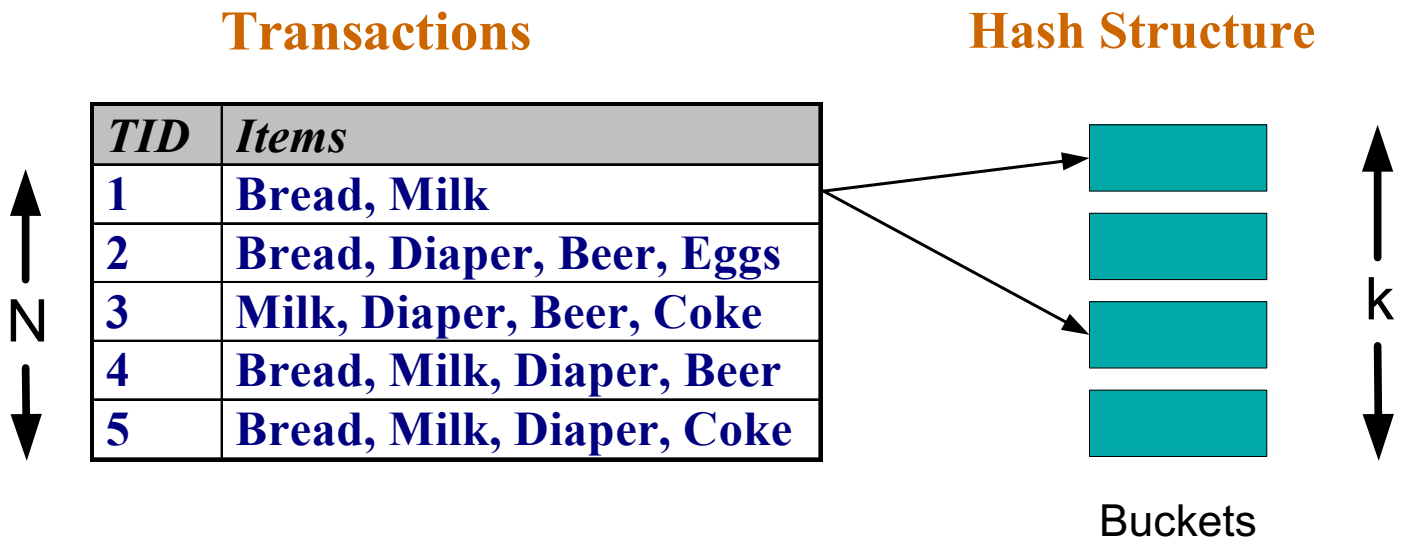
Itemset	sup
{B, C, E}	2

Example of Candidate Generation

- How to generate 4-itemset candidates?
- L_3={*abc, abd, acd, ace, bcd*}
- Self-joining: $L_3 \otimes L_3$
  - *abcd* from *abc* and *abd*
  - *acde* from *acd* and *ace*
- Pruning:
  - *acde* is removed because *ade* is not in L_3
- C_4={*abcd*}

Reduce # of Comparisons

- Support counting for candidates:
  - 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 tree*
    - Instead of matching each candidate against every transaction, match each transaction (and its subsets) against candidates stored in the hash tree (systematically and all together)!

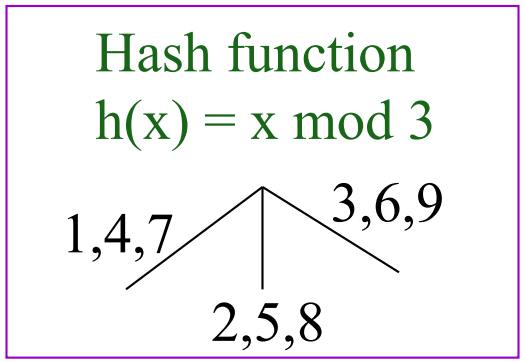


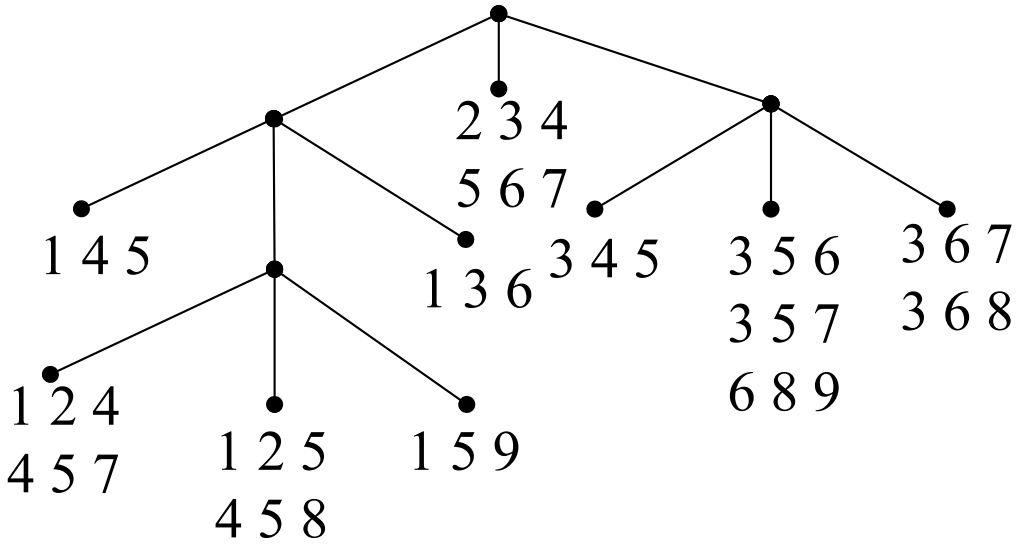
Generate Hash Tree

Hash trees are created as part of the candidate generation process. Suppose you have 15 candidate 3-itemsets:

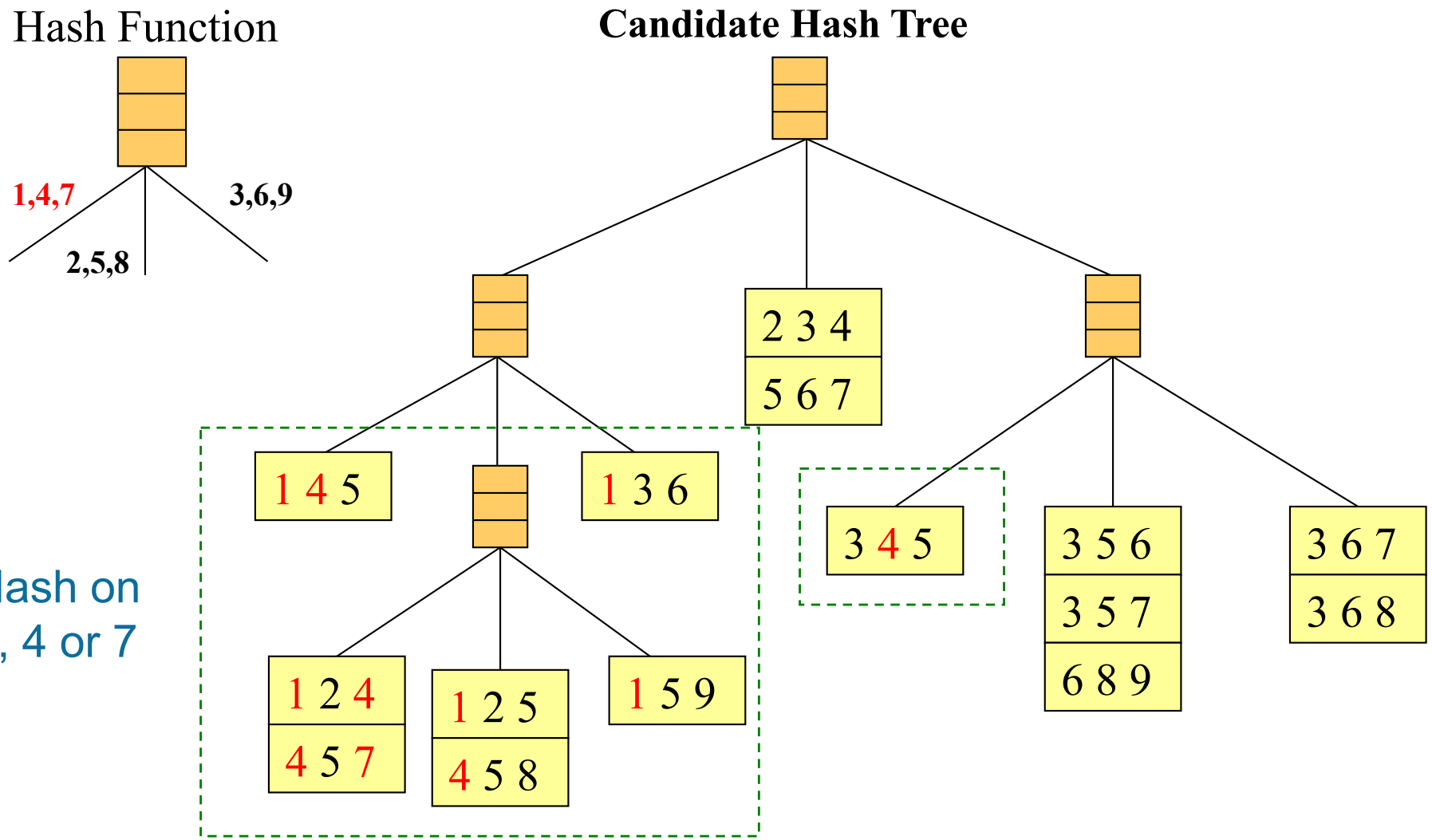
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

The tree is shaped by (i) Hash function and (ii) Max leaf size, i.e., max number of itemsets stored in a leaf node (if number of candidate itemsets exceeds max leaf size, split the node)

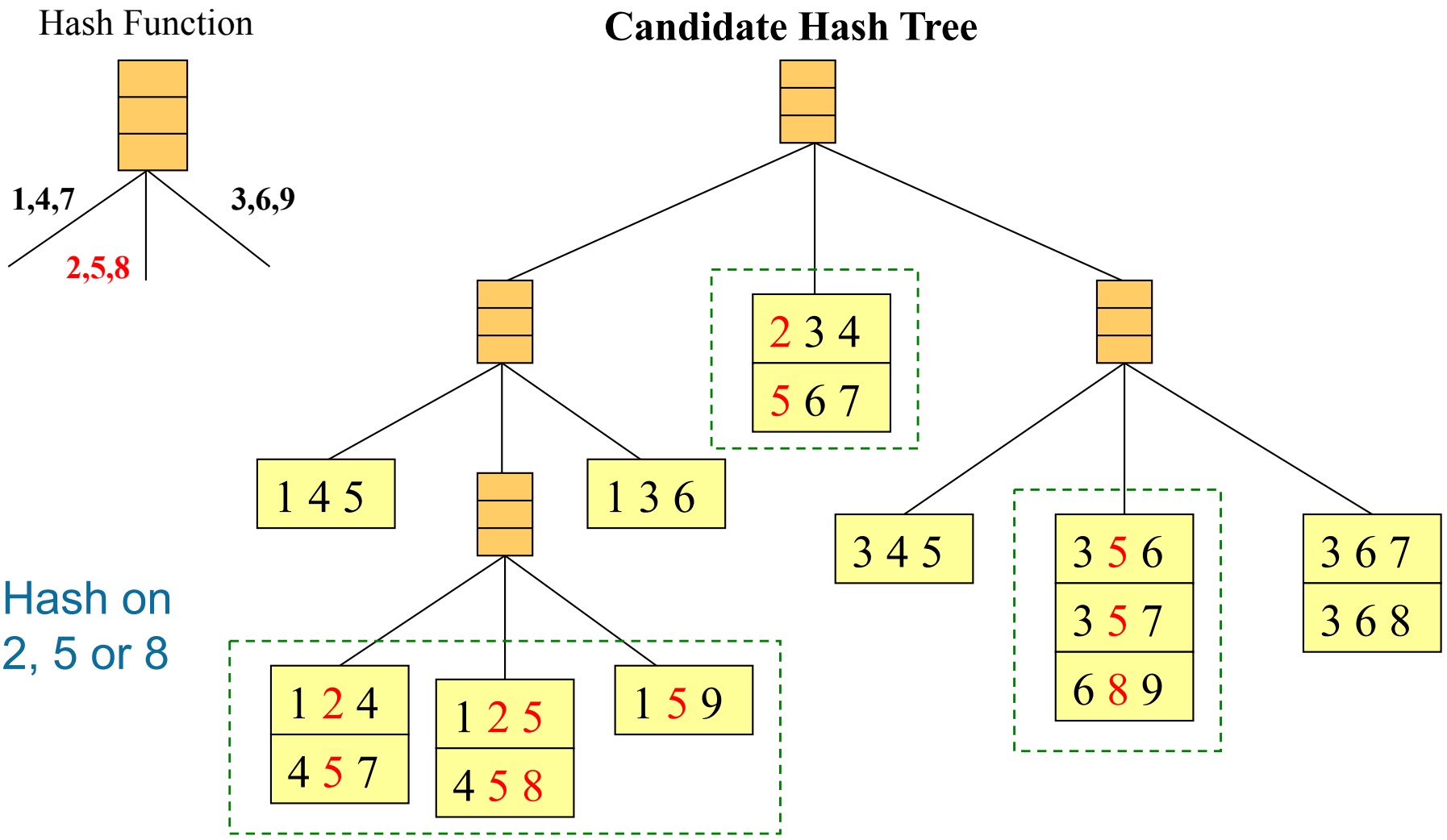




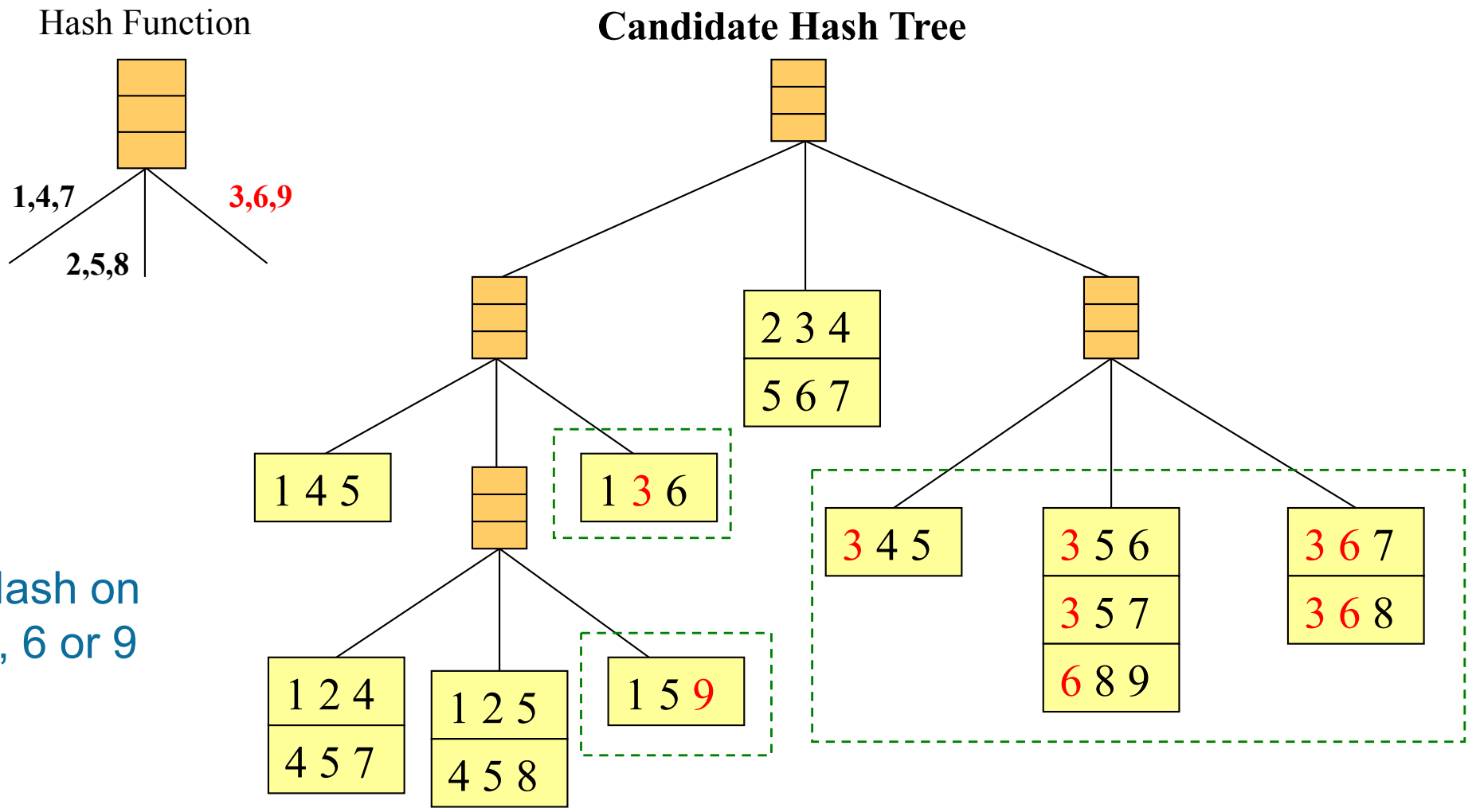
Association Rule Discovery: Hash tree



Association Rule Discovery: Hash tree



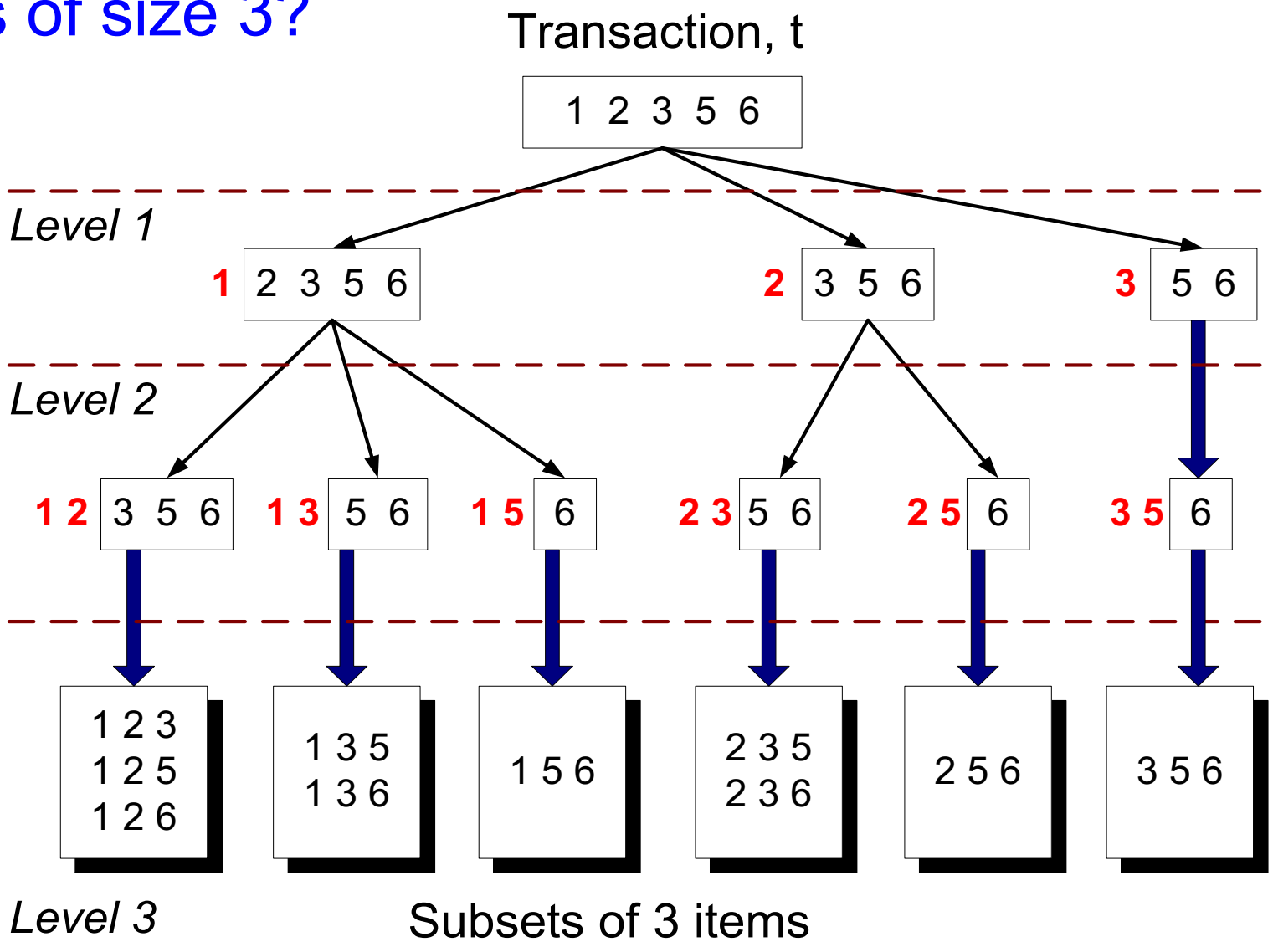
Association Rule Discovery: Hash tree



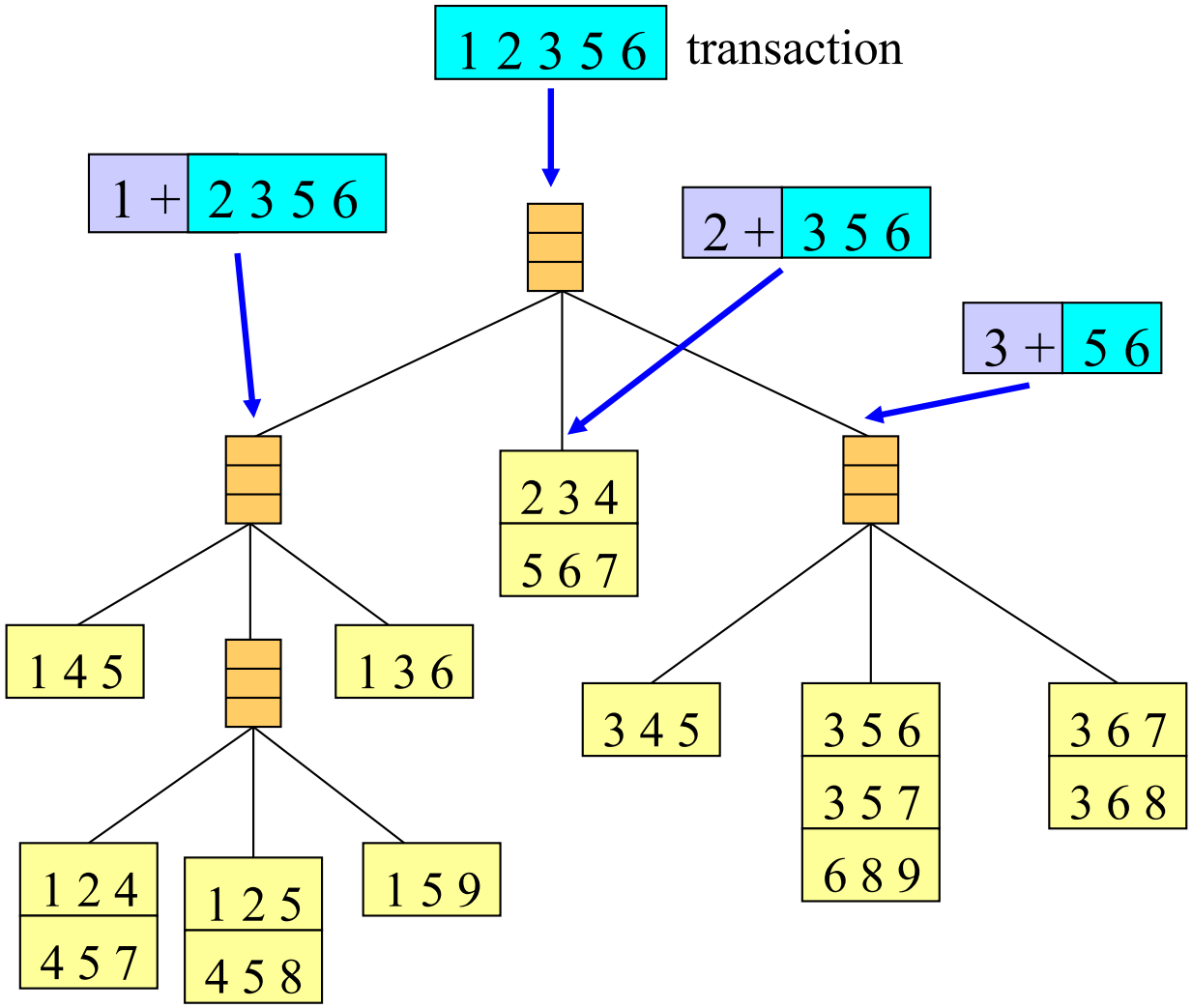
Hash on 3, 6 or 9

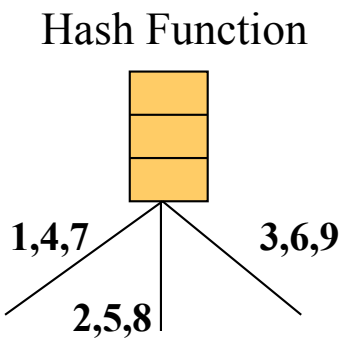
Subset Operation

Given a transaction t, how to enumerate all possible subsets of size 3?



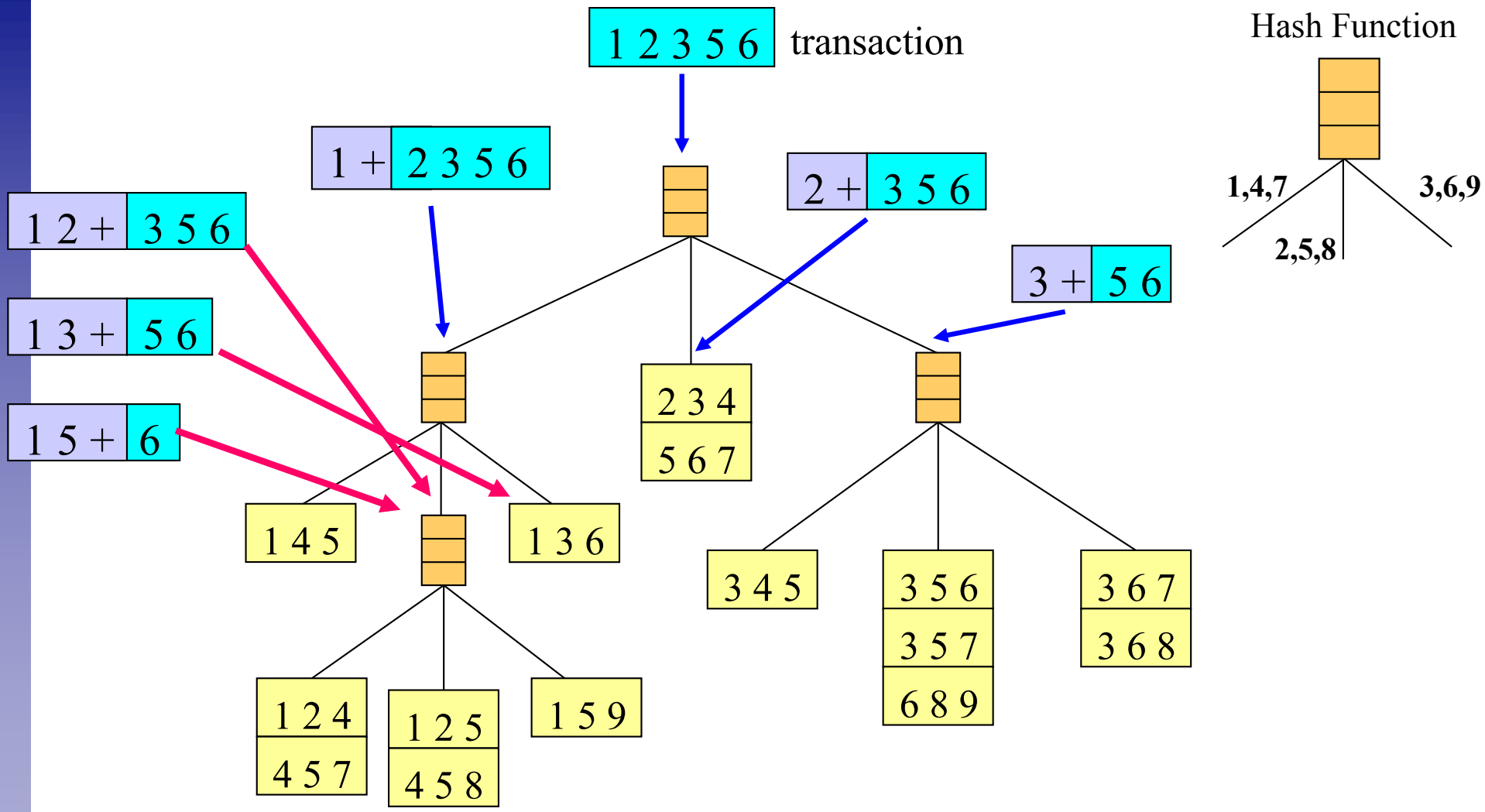
Subset Operation Using Hash Tree



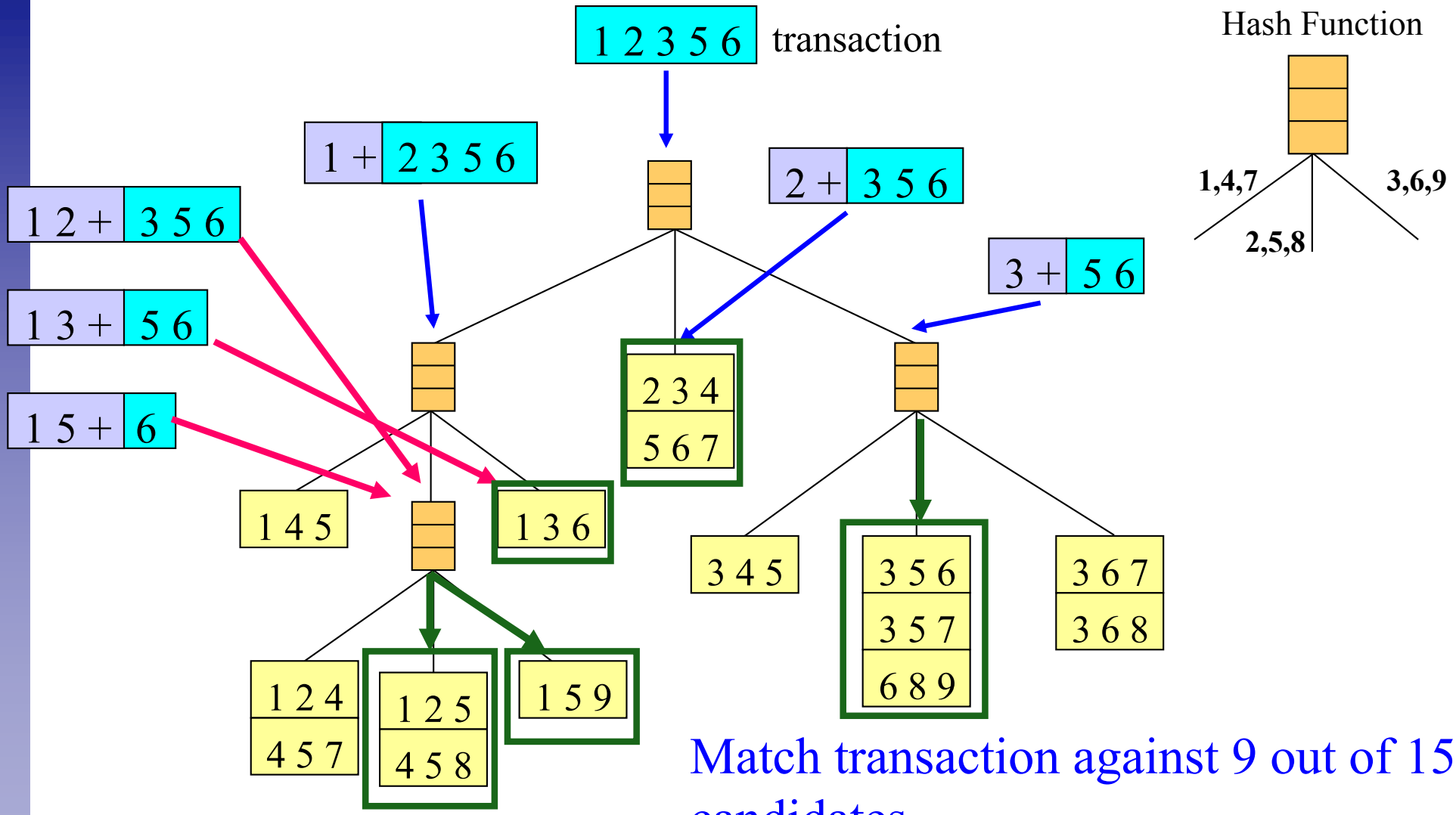


Each subset of the transaction only needs to match against corresponding subtree!

Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Match transaction against 9 out of 15 candidates

Factors Affecting Complexity

- Choice of minimum support threshold
  - lowering support threshold results in more freq. itemsets
  - may increase # 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)

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 $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

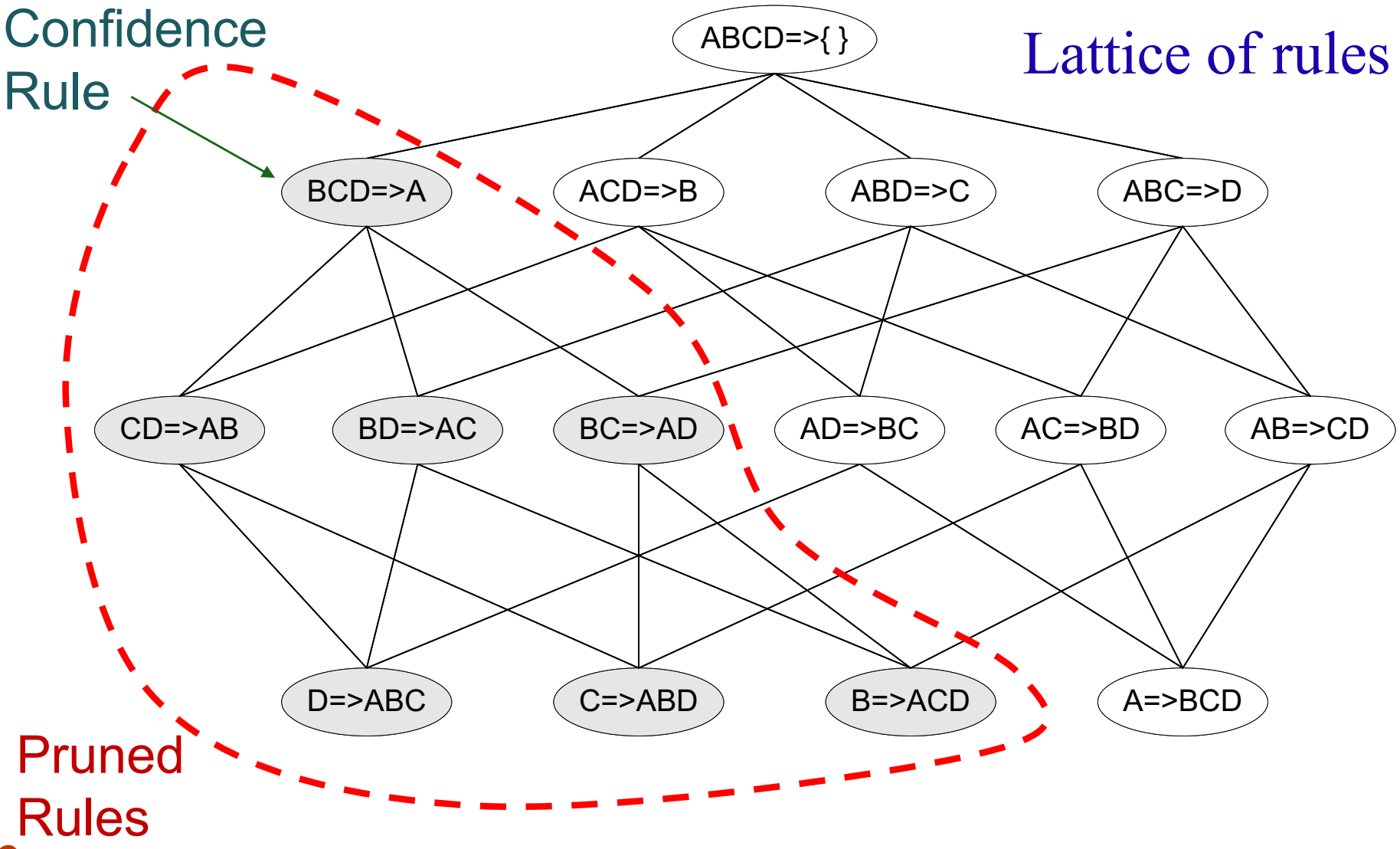
Rule Generation

- How to efficiently generate rules from frequent itemsets?
  - Similar to itemset generation, add items to the *consequent* part of rules, level by level.
  - In general, confidence does not have an anti-monotone property on subsets

    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}:

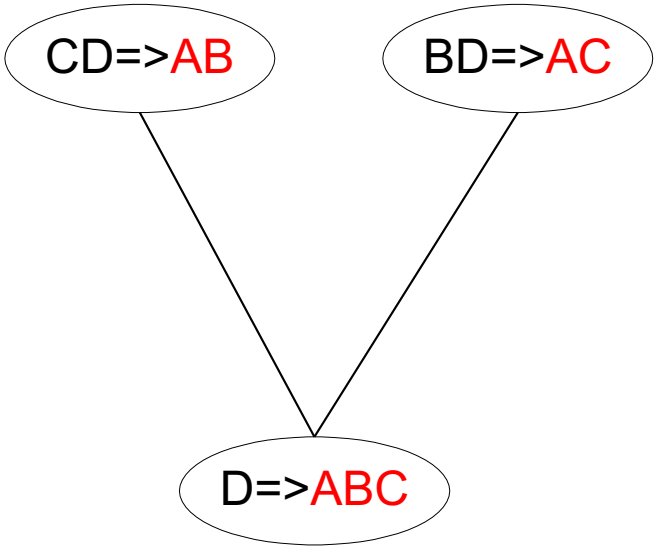
    $c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$
  - *The LHS condition is more strict, confidence is higher!*
    - $c(ABC \rightarrow D) = s(ABCD)/s(ABC) \geq s(ABCD)/s(AB) = c(AB \rightarrow CD)$

Rule Generation for Apriori Algorithm



Rule Generation with Apriori Algorithm

- Apriori Algorithm is applicable to generation of association rules.
- Candidate rule is generated by merging two rules in previous lattice level 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



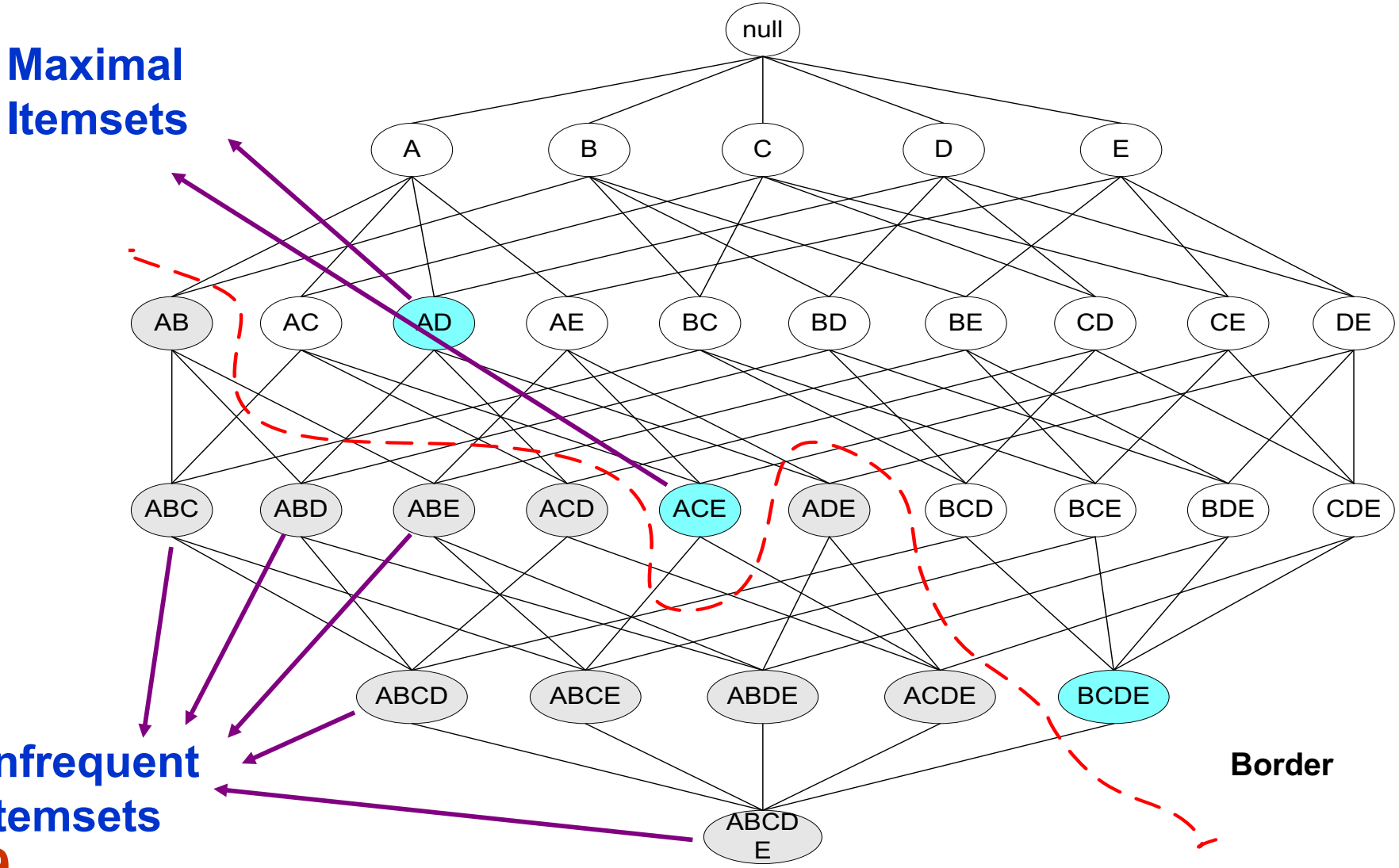
Compact Representation of Frequent Itemsets

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

- Some itemsets are redundant because they have identical support as their supersets
- Number of frequent itemsets $= 3 \times \sum_{k=1}^{10} \binom{10}{k}$
- Need a compact representation
  - Maximal and Closed frequent itemsets

Maximal Frequent Itemset

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



Closed Frequent Itemset

- An itemset is *closed* if none of its immediate supersets has the same support as the itemset
  - The highlighted itemsets below are closed itemsets.
  - The support of non-closed itemsets can be found by the maximum support of its supersets.

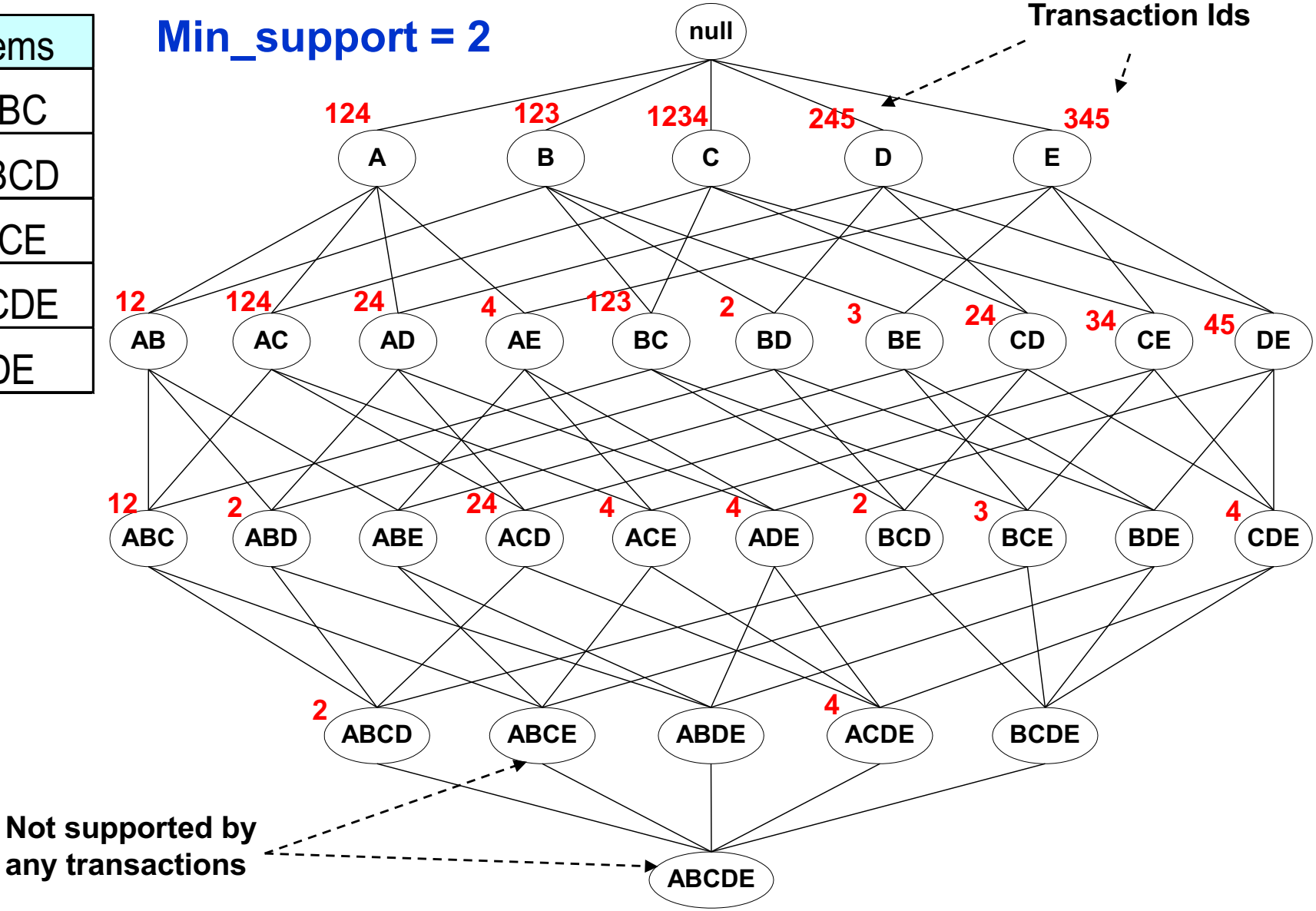
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

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

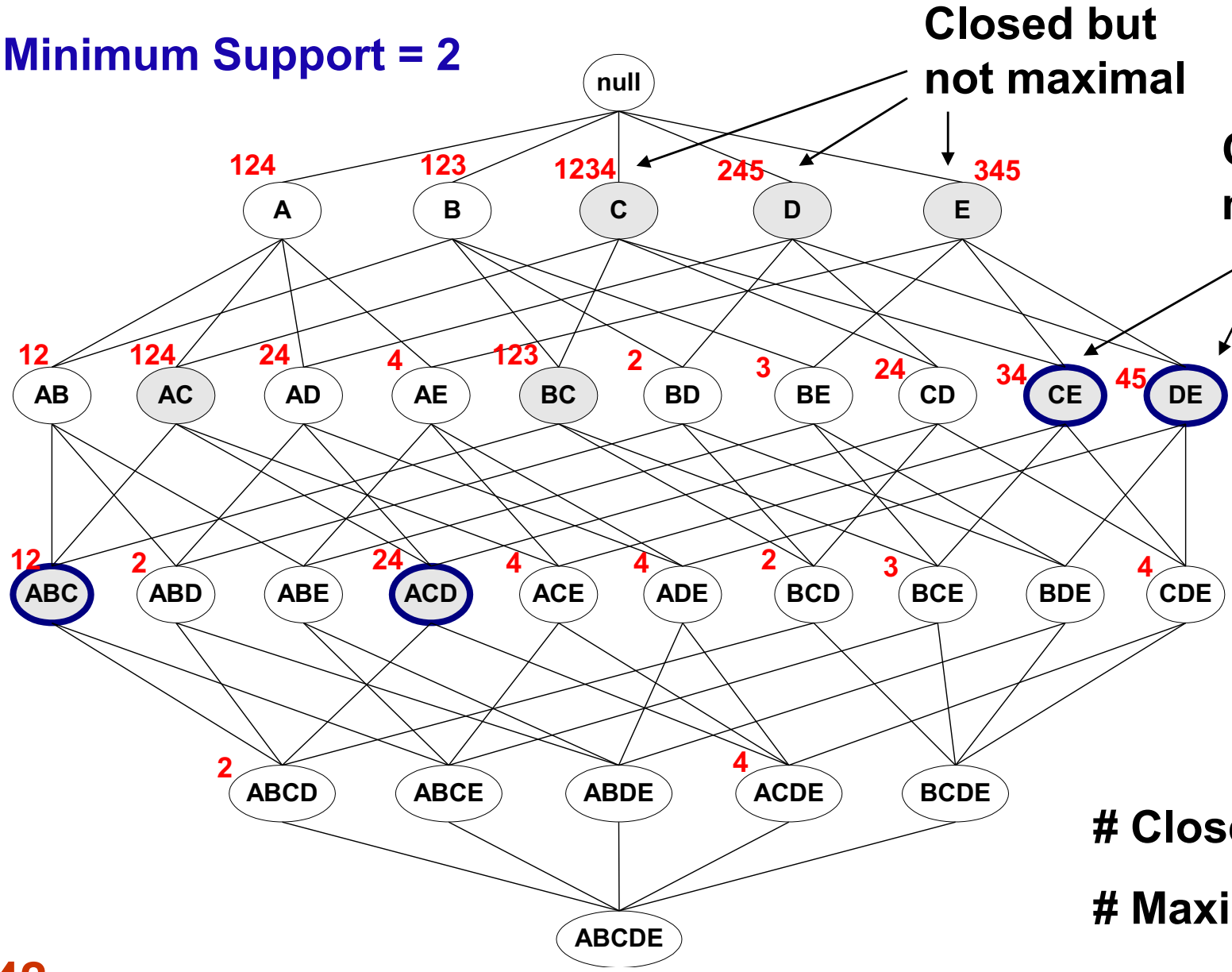
Lattice with Support Counts

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



Maximal vs Closed Frequent Itemsets

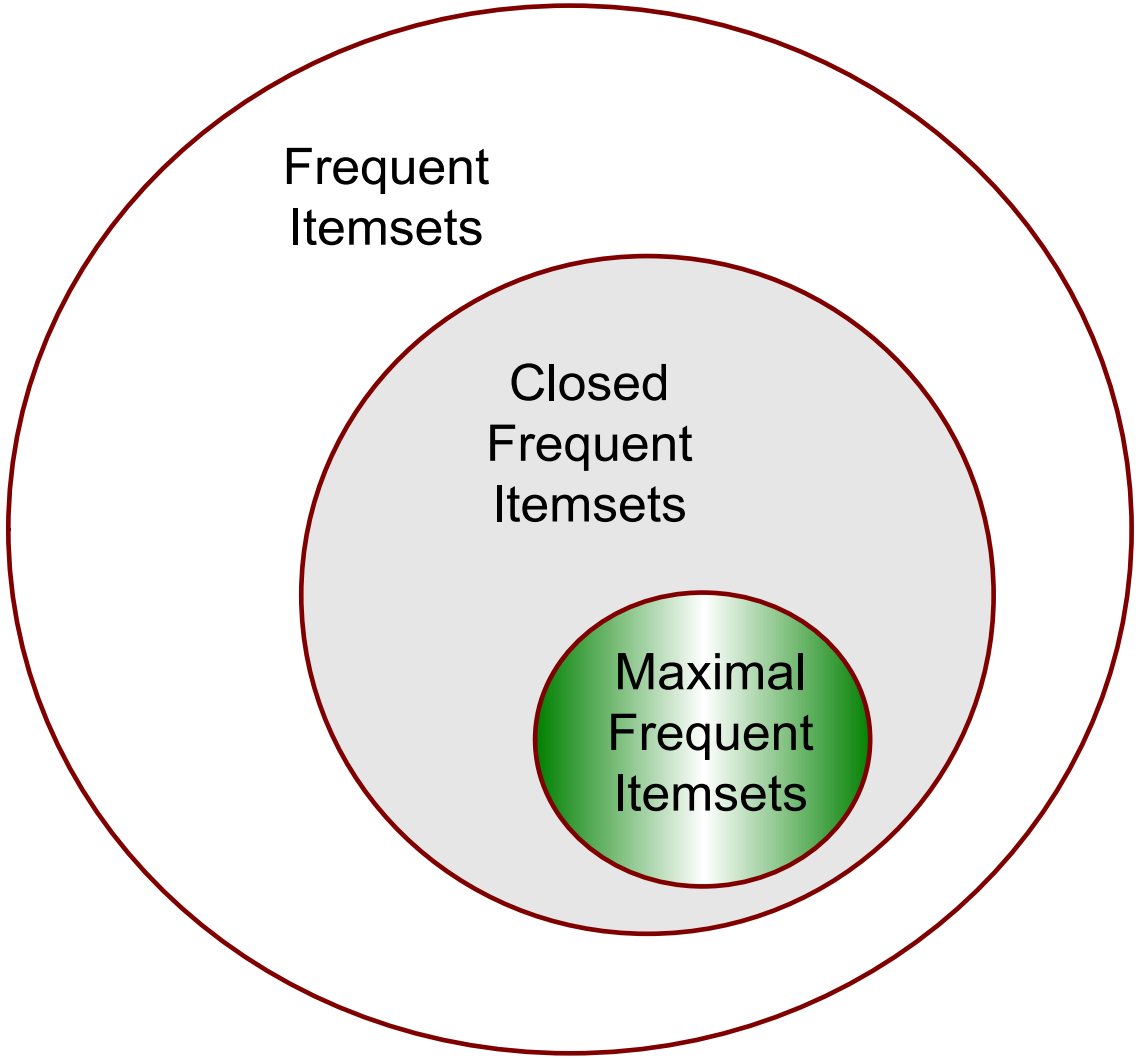
Minimum Support = 2



 # Closed = 9

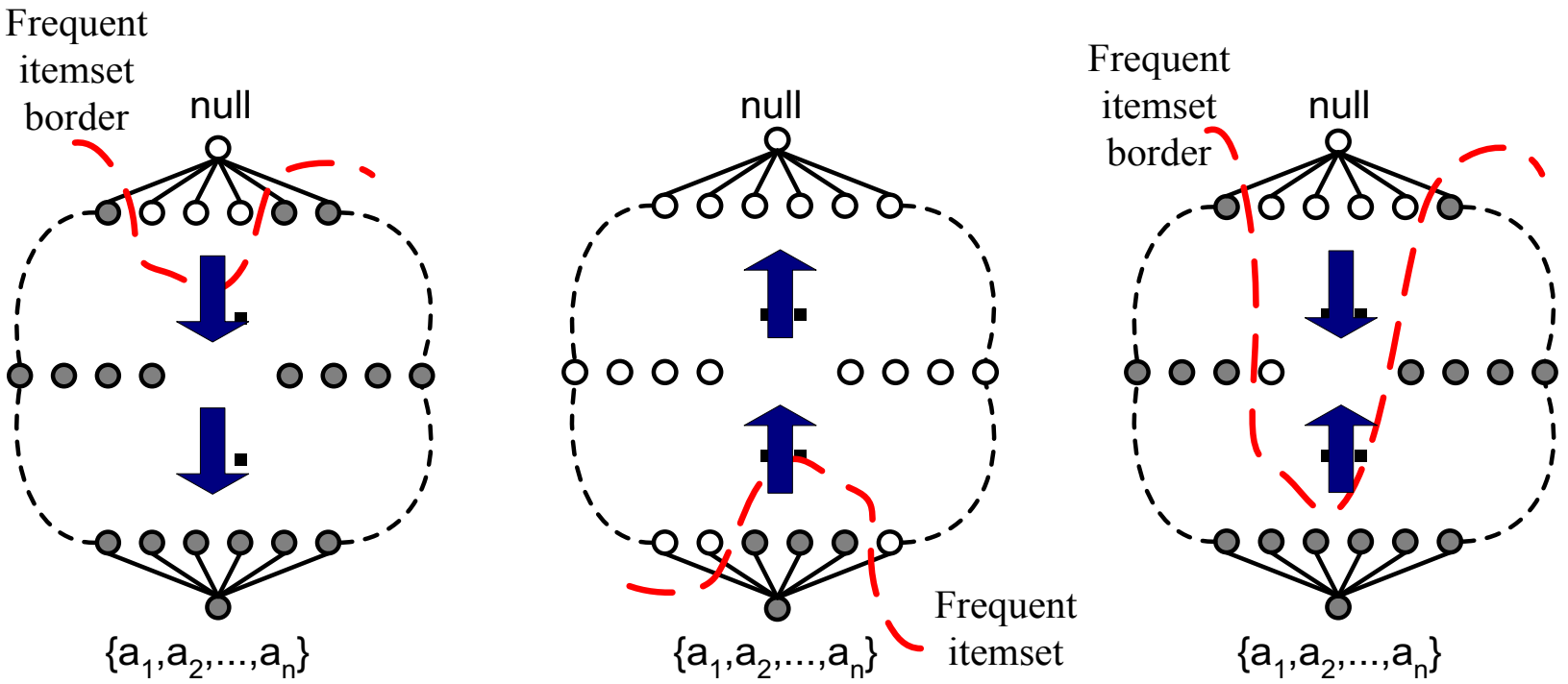
 # Maximal = 4

Maximal vs Closed Itemsets



Alternative Methods for Frequent Itemset Generation

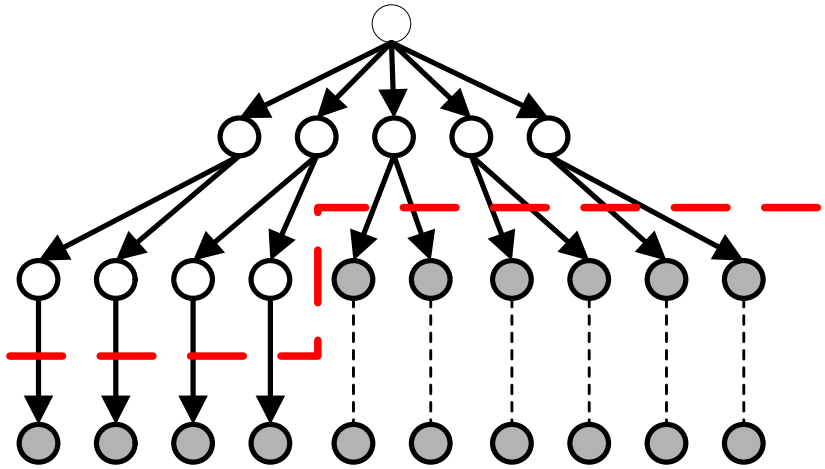
- Traversal of Itemset Lattice
  - General-to-specific vs Specific-to-general



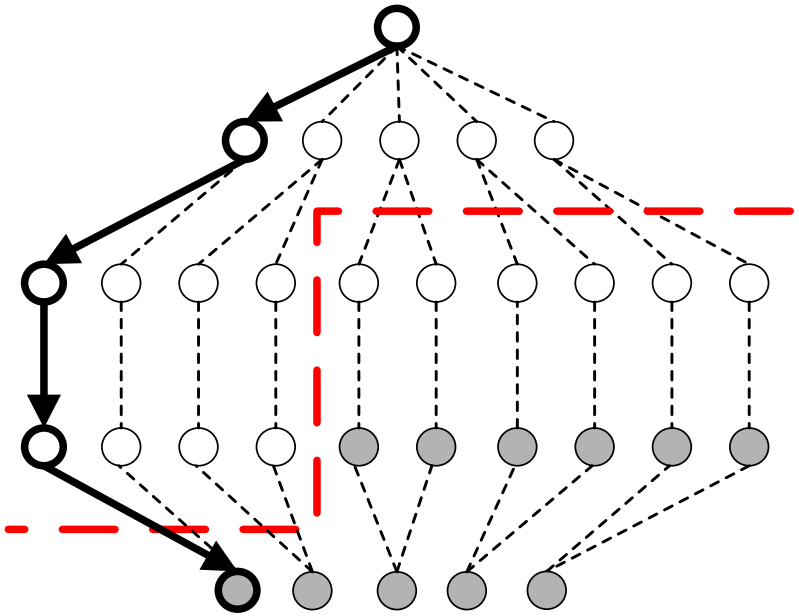
(a) General-to-specific (b) Specific-to-general (c) Bidirectional

Alternative Methods for Frequent Itemset Generation

- Traversal（遍历） of Itemset Lattice
  - Breadth-first vs Depth-first



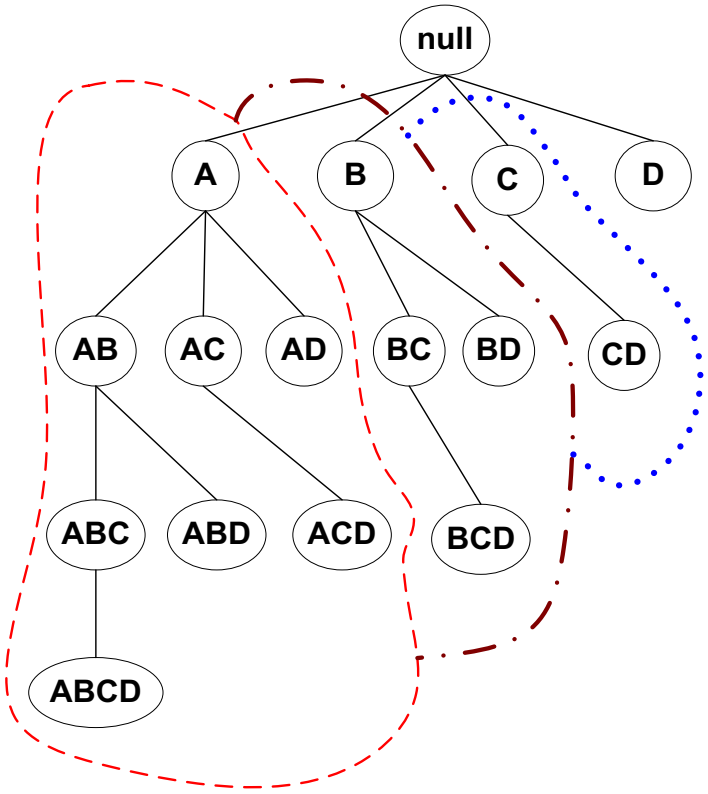
(a) Breadth first



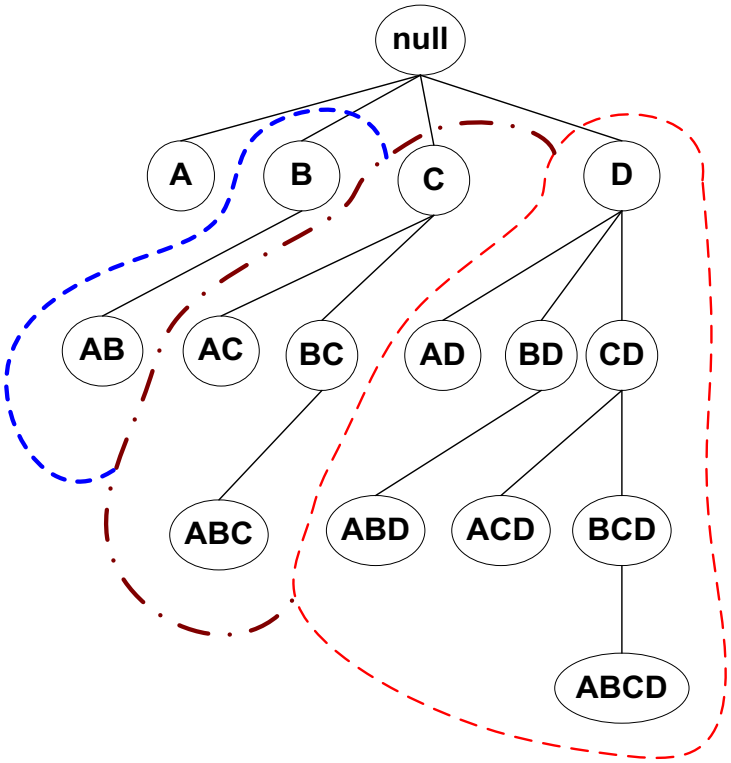
(b) Depth first

Alternative Methods for Frequent Itemset Generation

- Traversal of Itemset Lattice (Depth-First)
  - Equivalent Classes



(a) Prefix tree



(b) Suffix tree

Alternative Methods for Frequent Itemset Generation

- Representation of Database (i.e., transactions)
  - 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	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

TID-List of a larger itemsets can be easily derived by intersecting the TID-Lists of smaller itemsets.

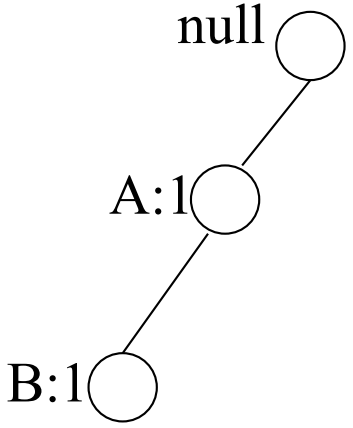
FP-growth Algorithm

- Different from the generate-and-test paradigm of Aprori,
- FP-growth builds an *FP-tree,* to generate frequent itemsets and count their support efficiently.
  - FP-tree is a *compressed* representation of the transaction database.
  - Each transaction is mapped onto a path in the FP-tree.
  - The compression is achieved through common prefix of paths.
- Once an FP-tree has been constructed, it uses a divide-and-conquer approach recursively to mine the frequent itemsets

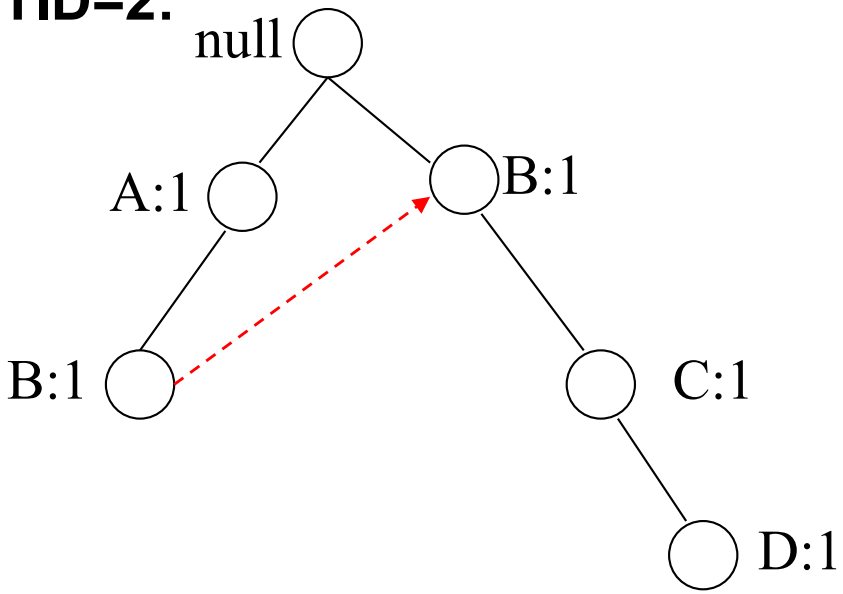
FP-tree construction

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:



Preprocessing: Scan the transactions to filter non-frequent items.

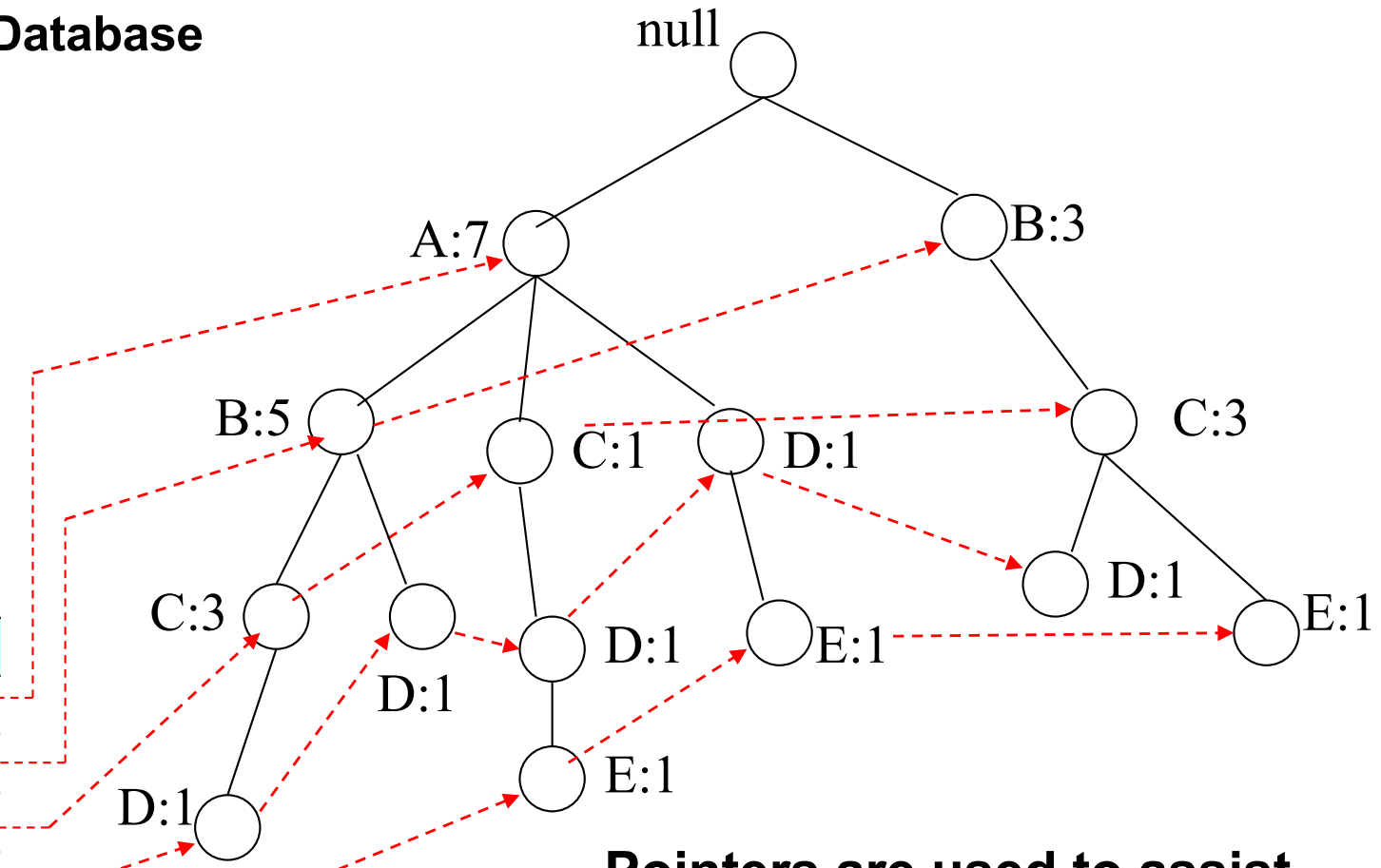
FP-Tree Construction

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	

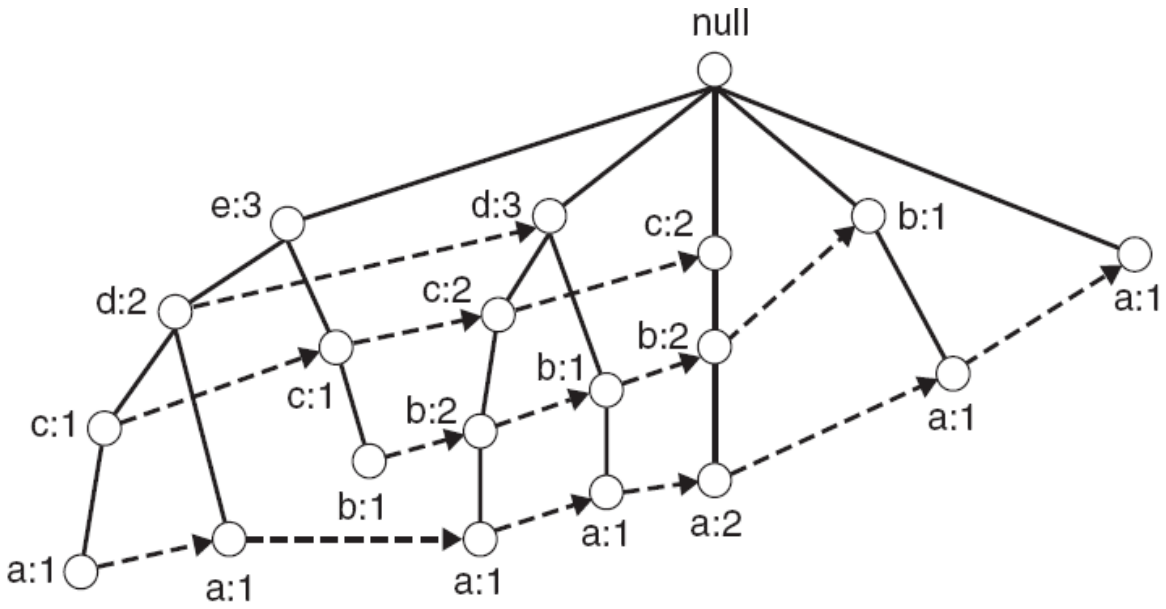


Pointers are used to assist frequent itemset generation

Items are usually sorted based on their individual supports in decreasing order. Why?

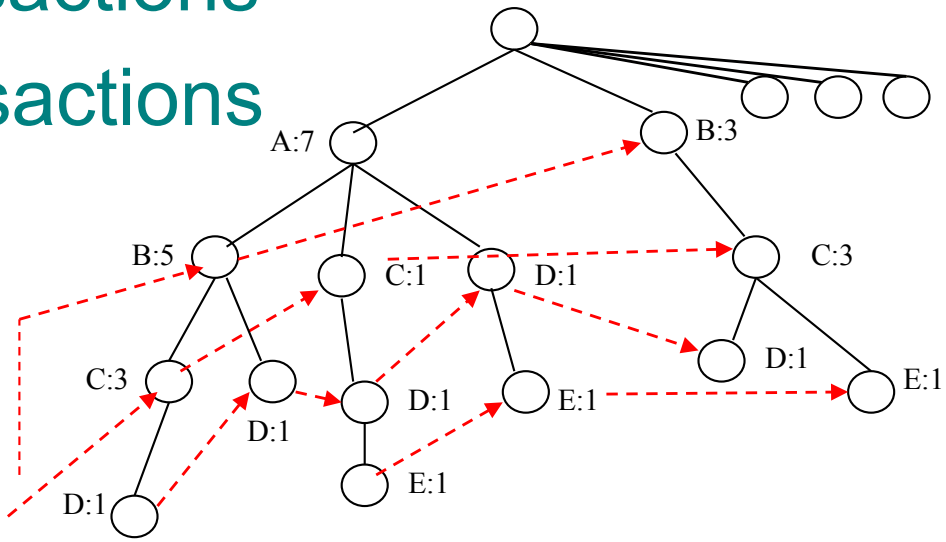
FP-Tree Construction

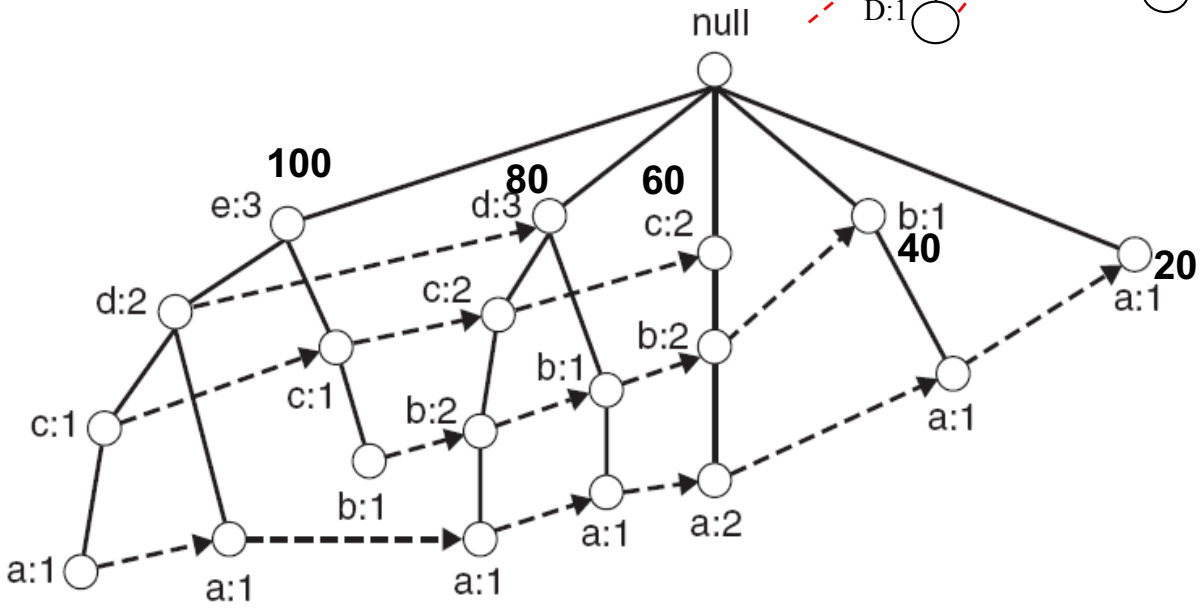
- The shape and sizes of the tree will be different, which may impact the performance, if the construction is based on different order
  - The following figure is constructed with the same dataset in increasing support of items.



(Counter) Example

- Assuming 100 transactions contain {e}, 80 transactions contain {d}, …

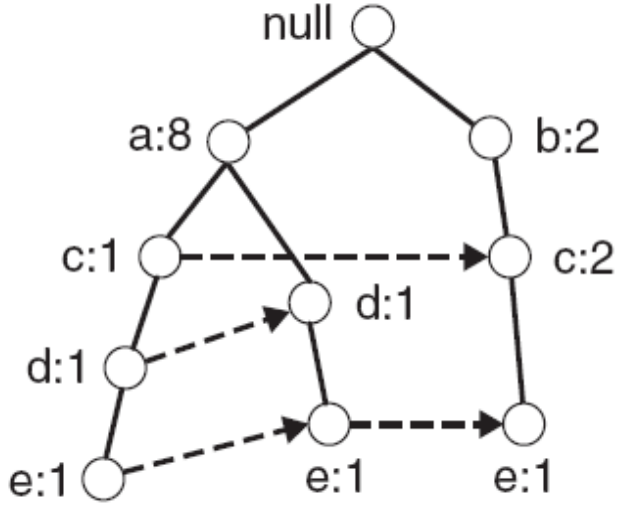




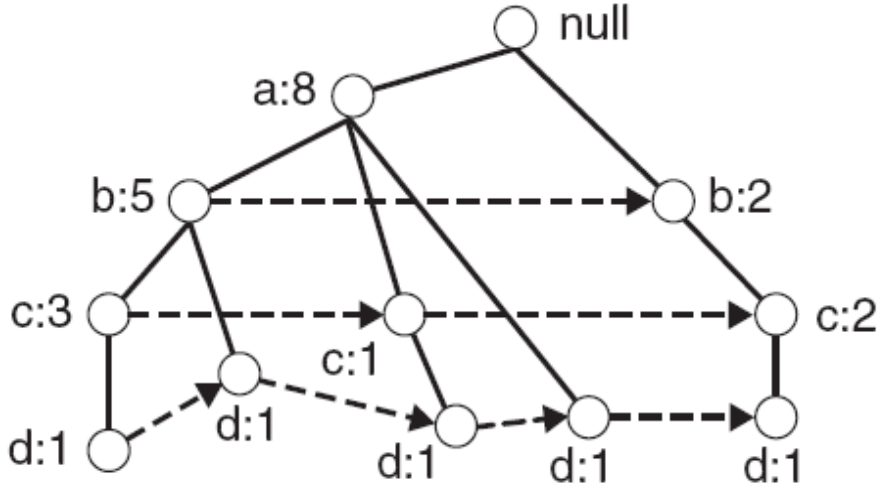
FP-Growth Overview

- Generate frequent items in a bottom-up fashion
  - Starting with the item with least support
- For each frequent 1-pattern (initial suffix pattern),
  - Construct a *conditional pattern base*
    - i.e., a subtree that has the same suffix in leaf nodes
    - consists of a set of prefix paths in FP-tree with the same given suffix pattern
- For each pattern base
  - Count support for the suffix pattern
  - Constructs corresponding *conditional FP-tree*
- Mining recursively on each conditional FP-tree.
  - Pattern growth is achieved by the concatenation of suffix pattern with frequent patterns generated from conditional FP-tree

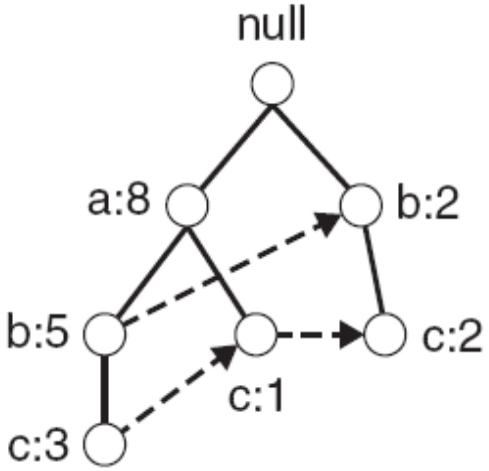
Conditional Pattern Bases



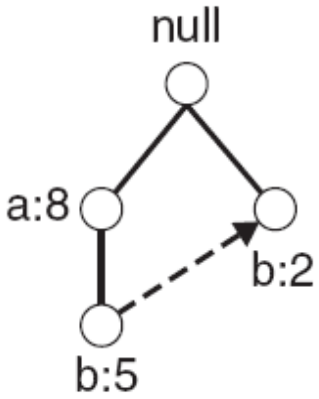
(a) Paths containing node e



(b) Paths containing node d



(c) Paths containing node c

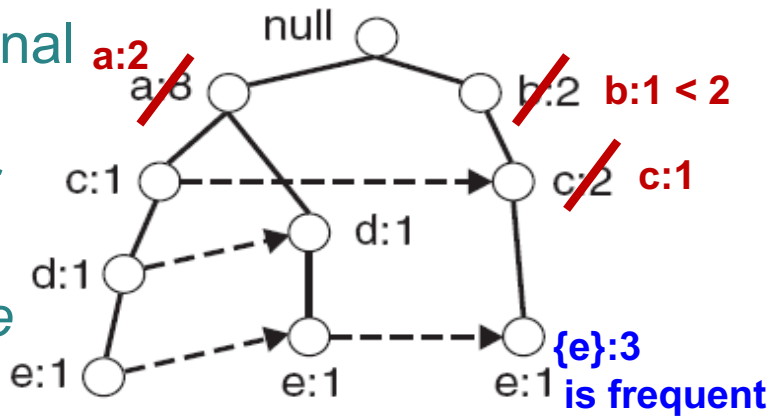


(d) Paths containing node b

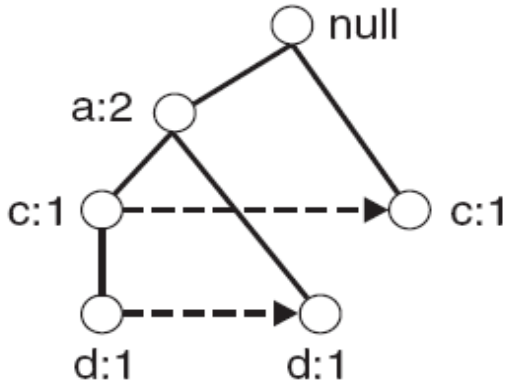
(e) Paths containing node a

Conditional Pattern FP-Trees

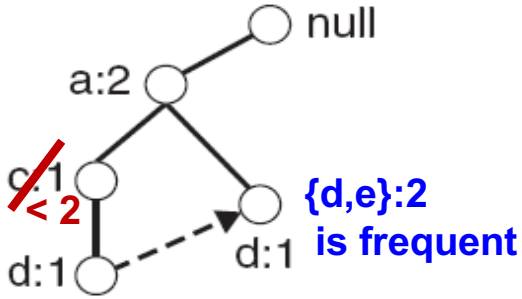
Conditional Pattern Base for Suffice pattern *e*



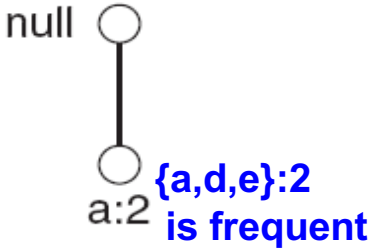
(a) Prefix paths ending in e



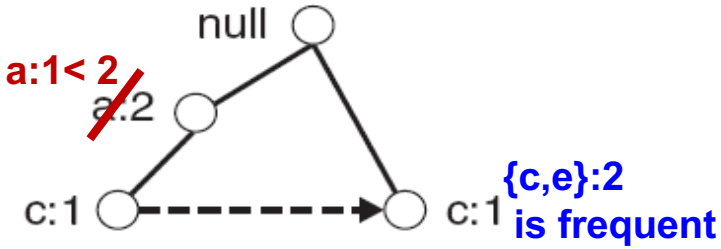
(b) Conditional FP-tree for e



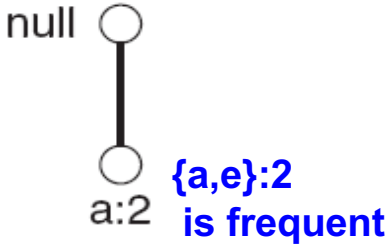
(c) Prefix paths ending in de



(d) Conditional FP-tree for de

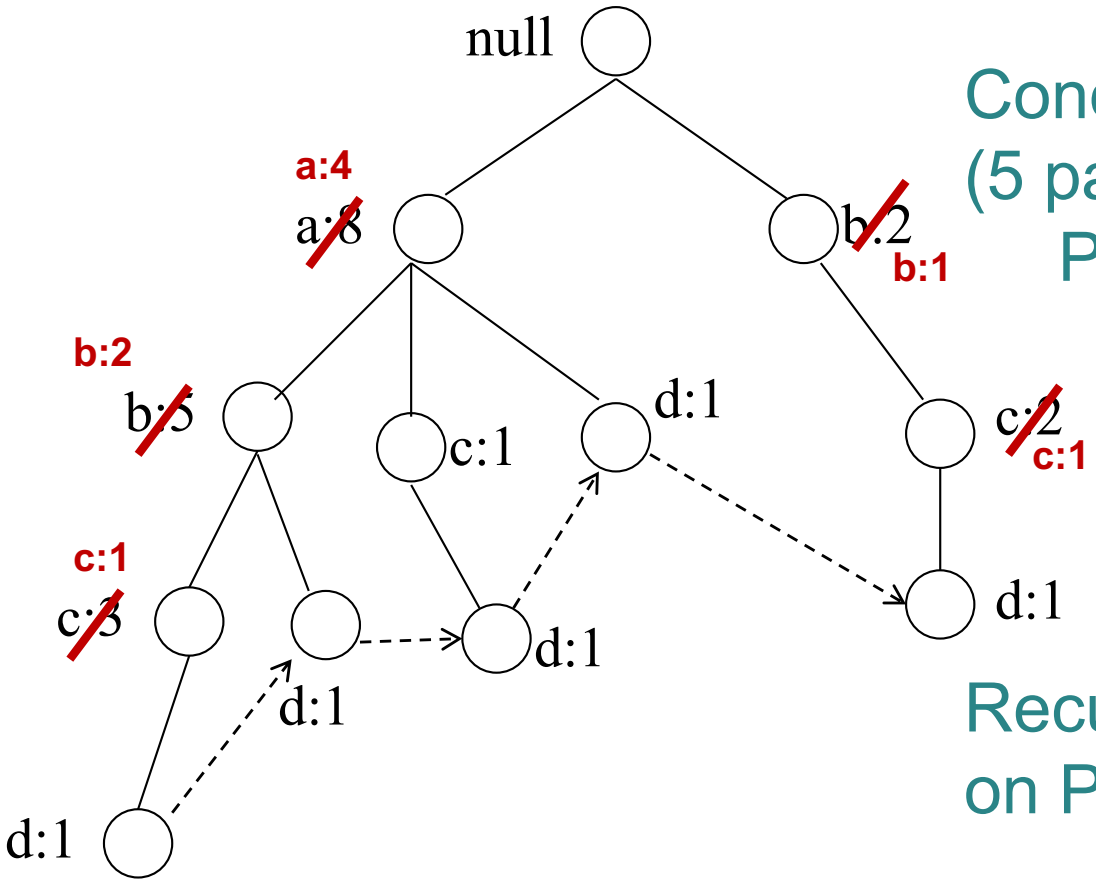


(e) Prefix paths ending in ce



(f) Prefix paths ending in ae

Conditional Pattern Bases (continued)



Conditional Pattern Base for d (5 paths ending with 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

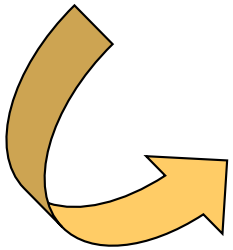
Frequent Itemsets found:
{d}, {c,d}.{b,c,d}.{a,c,d}, {b,d},{a,b,d}, {a,d}

FP-Trees Construction (Example)

- Step 1: Find frequent 1-item, sorted items in descending order of support by scanning DB

TID	Items bought
100	{a, c, d, f, g, i, m, p}
200	{a, b, c, f, i, m, o}
300	{b, f, h, j, o}
400	{b, c, k, s, p}
500	{a, c, e, f, l, m, n, p}

min_support = 3



a	3
b	3
c	4
f	4
m	3
p	3

Sorting based on support

f	4
c	4
a	3
b	3
m	3
p	3

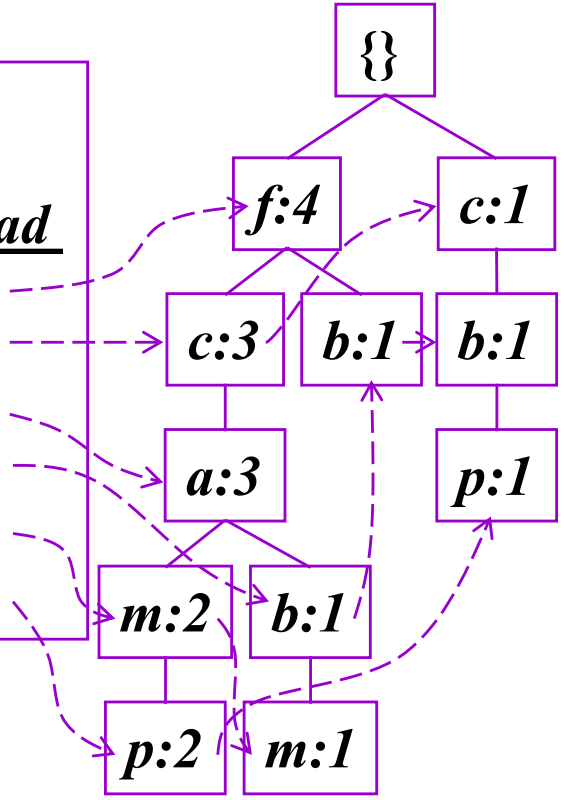
FP-Trees Construction (cont.)

- Step 2: Scan DB and construct the FP-tree

f	4
c	4
a	3
b	3
m	3
p	3

Header Table

Item	*frequency*	*head*
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	

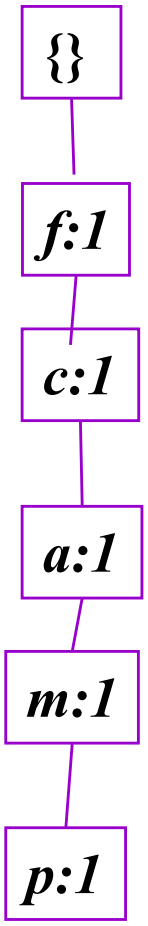


TID	Items bought	Ordered
100	{a, c, d, f, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, i, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, c, e, f, l, m, n, p}	{f, c, a, m, p}

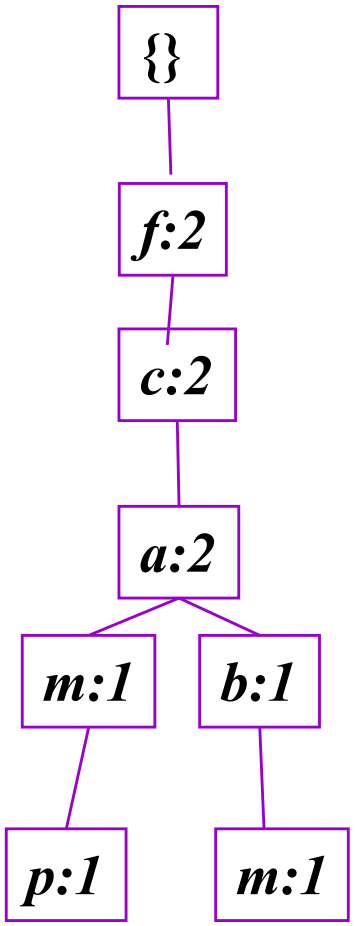
FP-Tree Construction (Illustration)

TID	(ordered) frequent items
100	{f, c, a, m, p}
200	{f, c, a, b, m}
300	{f, b}
400	{c, b, p}
500	{f, c, a, m, p}

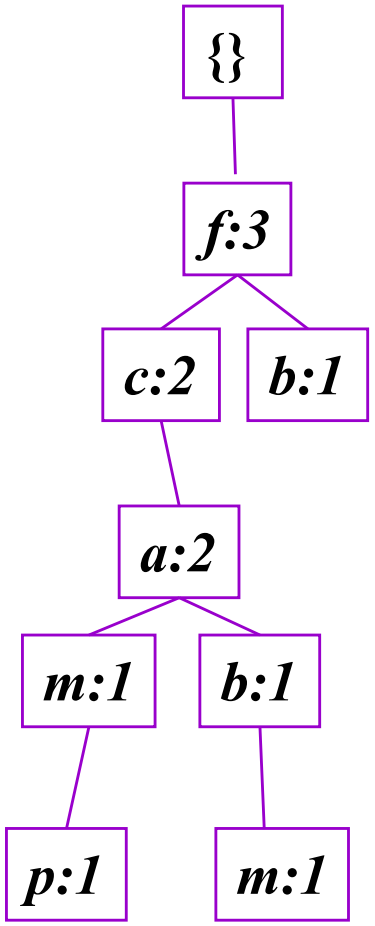
TID:100



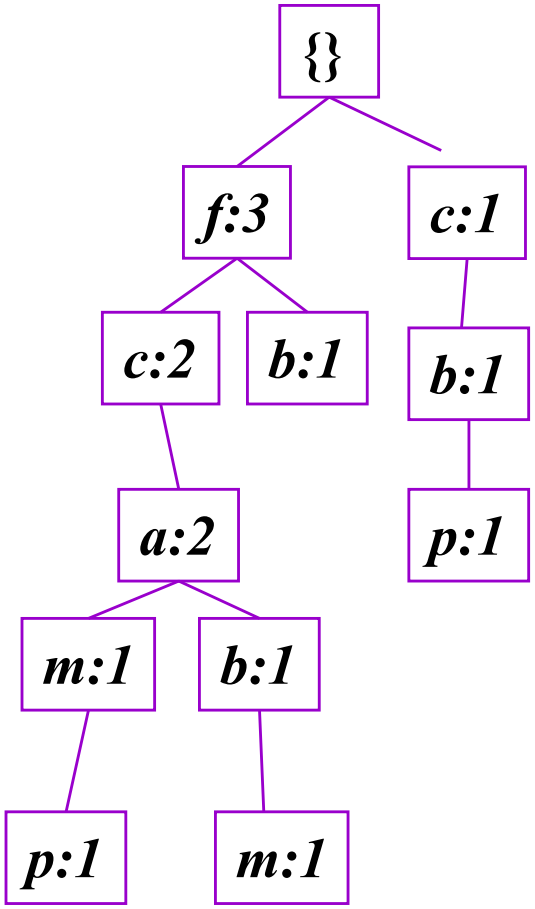
TID:200



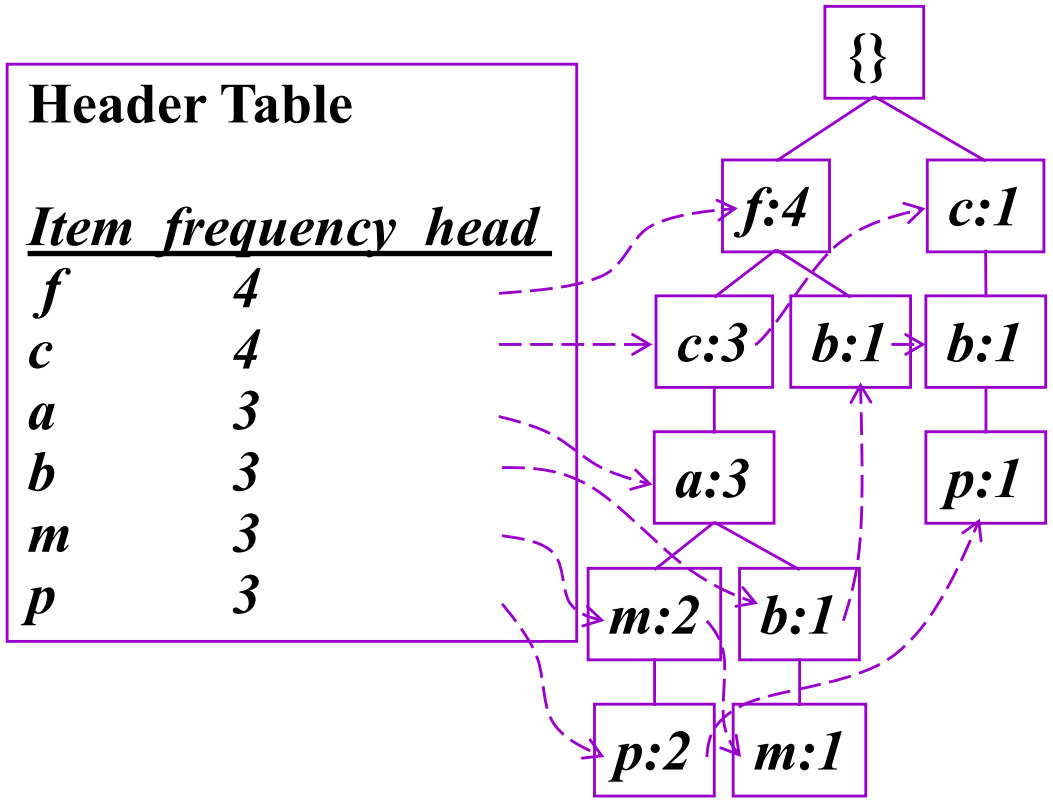
TID:300



TID:400



FP-Growth (Example)



min_support = 3

Item	Cond. Pattern base
c	f:3
a	fc:3
b	fca:1,f:1,c:1
m	fca:2,fcab:1
p	fcam:2,cb:1

FP-Growth (cont.)

- Conditional Pattern Base for p: fcam:2,cb:1
  - Update support/count for each item in the base
    - f:2,c:3,a:2,m:2,b:1
  - Construct corresponding Conditional FP trees for next items that form frequent patterns with p
    - i.e., only c is qualified in this example

Min_support=3

Paths	Count
fcam:2	c:2
cb:1	c:1

→ {} → **All frequent patterns relating to *p***

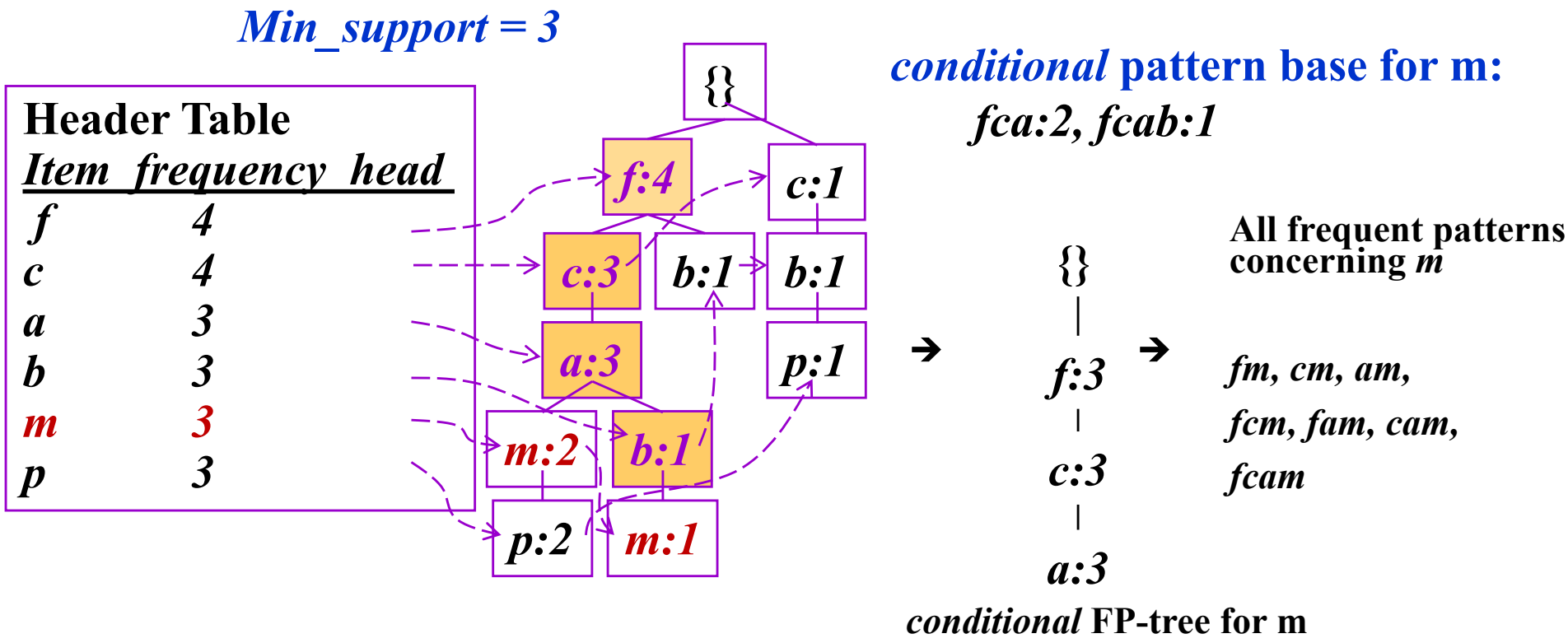
c:3

cp

Conditional FP-tree for cp

FP-Growth (Cont.)

- Pattern-base for m
  - Update support/count for each item in the base
  - Construct the conditional FP-tree for the frequent items of the pattern base



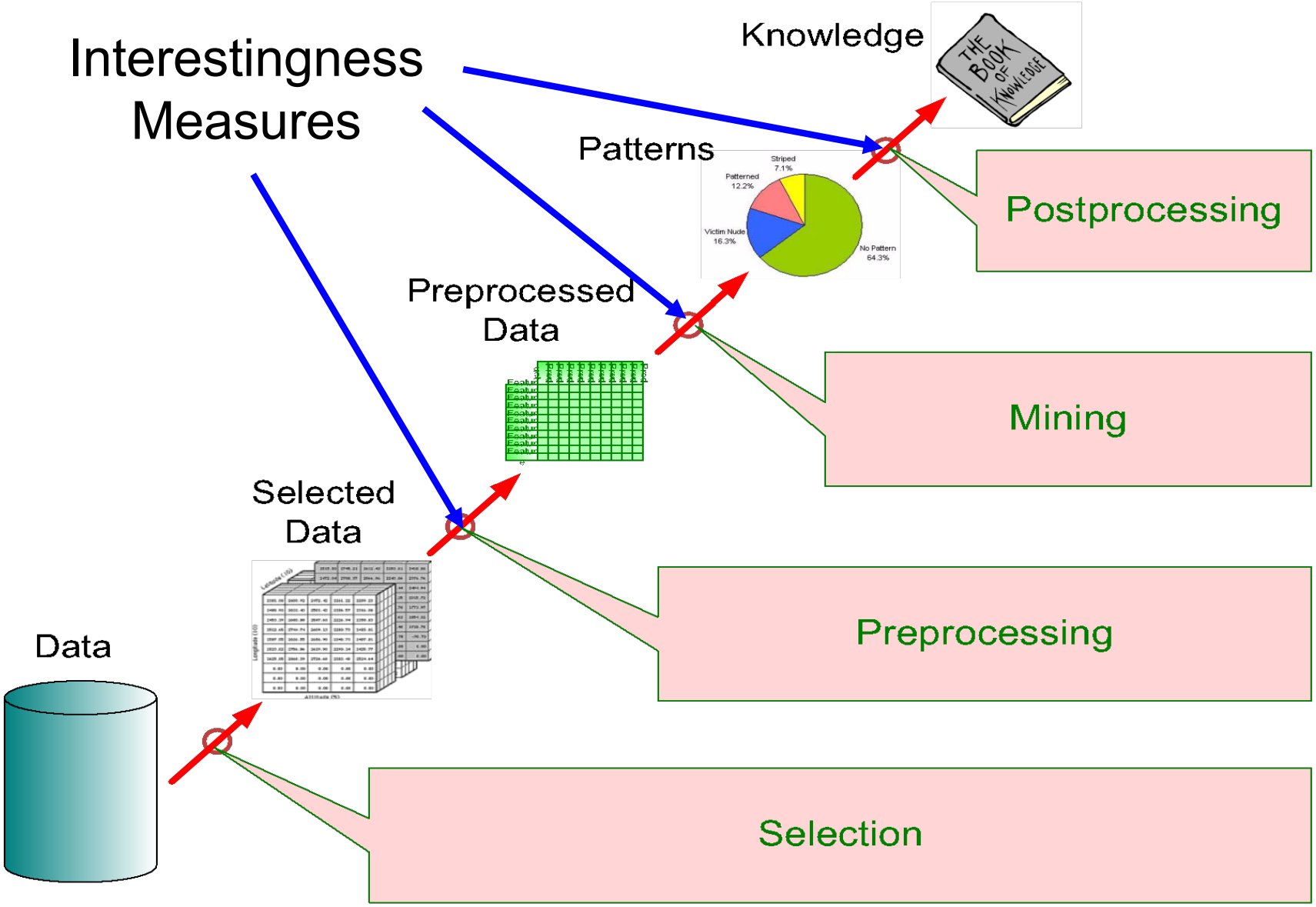
Why is FP-Growth the Winner?

- Divide-and-Conquer:
  - Decompose both the mining task and DB according to the frequent patterns obtained so far
  - Leads to focused search in smaller databases
- Other factors
  - No candidate generation, no candidate test
  - Compressed database: FP-tree structure
  - No repeated scan of entire database
  - Basic operations:
    - Counting local freq items
    - Building sub FP-tree
    - No pattern search and matching

Pattern Evaluation

- Association rule algorithms tend to produce too many rules
  - many of them are uninteresting or redundant
  - E.g., redundant if {A,B,C} → {D} and {A,B} → {D} have same support & confidence
- In the original formulation of association rules, support & confidence are the only measures used
  - There are some limitations with the support-confidence approach
- Interestingness measures based on *objective measure* or *subjective domain information* can be used to prune/rank the derived patterns

Application of Interestingness Measure



Contingency table

- 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.
  - E.g., confidence($X \rightarrow Y$) = f_{11}/f_{1+}

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) = 15/20 = 0.75

but P(Coffee) = 90/100 = 0.9 (higher than the above!)

$\Rightarrow$ Although confidence is high, the rule is misleading because 90% of people drink coffee

$\Rightarrow$ P(Coffee|$\overline{\text{Tea}}$) = 75/80 = 0.9375 (even higher!)

Statistical Independence

- Some phenomena (dependency among observed attributes) may exist in human behavior!
- 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

- A number of statistical-based measures can be used to evaluate the quality of association rules.

$$Lift = \frac{c(X \rightarrow Y)}{s(Y)/N} = \frac{P(Y|X)}{P(Y)}$$

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

$$PS = P(X,Y) - P(X)P(Y)$$

PS: Piatetsky-Shapiro

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

Example: Lift/Interest

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

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

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75

P(Coffee) = 0.9

$\Rightarrow$ Lift = 0.75/0.9= 0.8333 (< 1, i.e., negatively correlated)

Statistical independence: If P(X,Y)=P(X)P(Y), Lift = 1

Drawback of Lift & Interest

	Y	$\bar{Y}$	
X	10	0	10
$\bar{X}$	0	90	90
	10	90	100

	Y	$\bar{Y}$	
X	90	0	90
$\bar{X}$	0	10	10
	90	10	100

$$Lift = \frac{0.1}{(0.1)(0.1)} = 10$$

$$Lift = \frac{0.9}{(0.9)(0.9)} = 1.11$$

If Lift > 1 , X and Y are positively correlated. Thus, both cases are positively correlated.

Is the left one stronger than the right one?

Measures for Association Rules

- 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)	$\max\left(P(A,B)\log(\frac{P(B\|A)}{P(B)}) + P(A\overline{B})\log(\frac{P(\overline{B}\|A)}{P(\overline{B})}),\right.$ $\left. P(A,B)\log(\frac{P(A\|B)}{P(A)}) + P(\overline{A}B)\log(\frac{P(\overline{A}\|B)}{P(\overline{A})})\right)$
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)	$\max\left(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2}\right)$
13	Conviction (V)	$\max\left(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})}\right)$
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)	$\max\left(\frac{P(B\|A)-P(B)}{1-P(B)}, \frac{P(A\|B)-P(A)}{1-P(A)}\right)$
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)	$\sqrt{P(A,B)}\max(P(B\|A)-P(B), P(A\|B)-P(A))$

Comparing Different Measures

10 examples of contingency tables:

Example	f_{11}	f_{10}	f_{01}	f_{00}
E1	8123	83	424	1370
E2	8330	2	622	1046
E3	9481	94	127	298
E4	3954	3080	5	2961
E5	2886	1363	1320	4431
E6	1500	2000	500	6000
E7	4000	2000	1000	3000
E8	4000	2000	2000	2000
E9	1720	7121	5	1154
E10	61	2483	4	7452

Rankings of contingency tables using various measures:

#	ϕ	λ	α	Q	Y	κ	M	J	G	s	c	L	V	I	IS	PS	F	AV	S	ζ	K
E1	1	1	3	3	3	1	2	2	1	3	5	5	4	6	2	2	4	6	1	2	5
E2	2	2	1	1	1	2	1	3	2	2	1	1	1	8	3	5	1	8	2	3	6
E3	3	3	4	4	4	3	3	8	7	1	4	4	6	10	1	8	6	10	3	1	10
E4	4	7	2	2	2	5	4	1	3	6	2	2	2	4	4	1	2	3	4	5	1
E5	5	4	8	8	8	4	7	5	4	7	9	9	9	3	6	3	9	4	5	6	3
E6	6	6	7	7	7	7	6	4	6	9	8	8	7	2	8	6	7	2	7	8	2
E7	7	5	9	9	9	6	8	6	5	4	7	7	8	5	5	4	8	5	6	4	4
E8	8	9	10	10	10	8	10	10	8	4	10	10	10	9	7	7	10	9	8	7	9
E9	9	9	5	5	5	9	9	7	9	8	3	3	3	7	9	9	3	7	9	9	8
E10	10	8	6	6	6	10	5	9	10	10	6	6	5	1	10	10	5	1	10	10	7

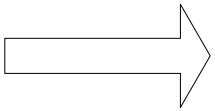
skip

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) increase 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
- Other properties (in next few slides)
  - Variable Permutation
  - Row/Column Scaling
  - Inversion
  - Null Addition

Property of Measures 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):

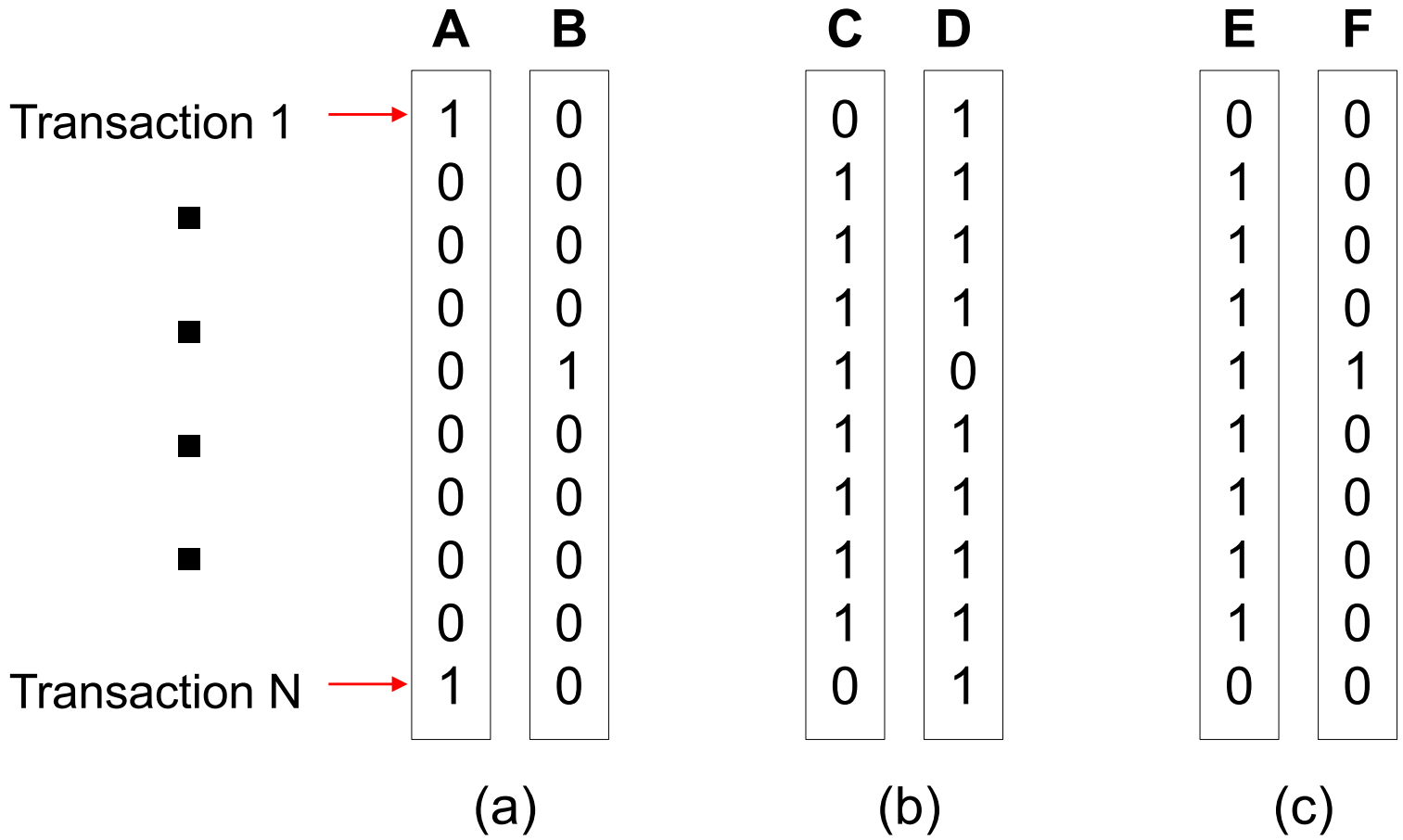
	Male	Female	
High	2	3	5
Low	1	4	5
	3	7	10

	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



- Invariant measures: correlation, odd ratio, etc.
- Non-invariant measures: Jaccard, cosine, etc

Inversion Example: ϕ-Coefficient

- Inversion Property: A measure M is invariant under inversion, if M remains the same by exchanging f_{11} with f_{00} and f_{10} with f_{01}.

	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

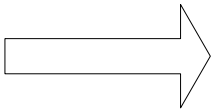
$$\phi = \frac{0.6 - 0.7 \times 0.7}{\sqrt{0.7 \times 0.3 \times 0.7 \times 0.3}} = 0.5238$$

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

ϕ-coefficient is the same for both tables

Property under Null Addition

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



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

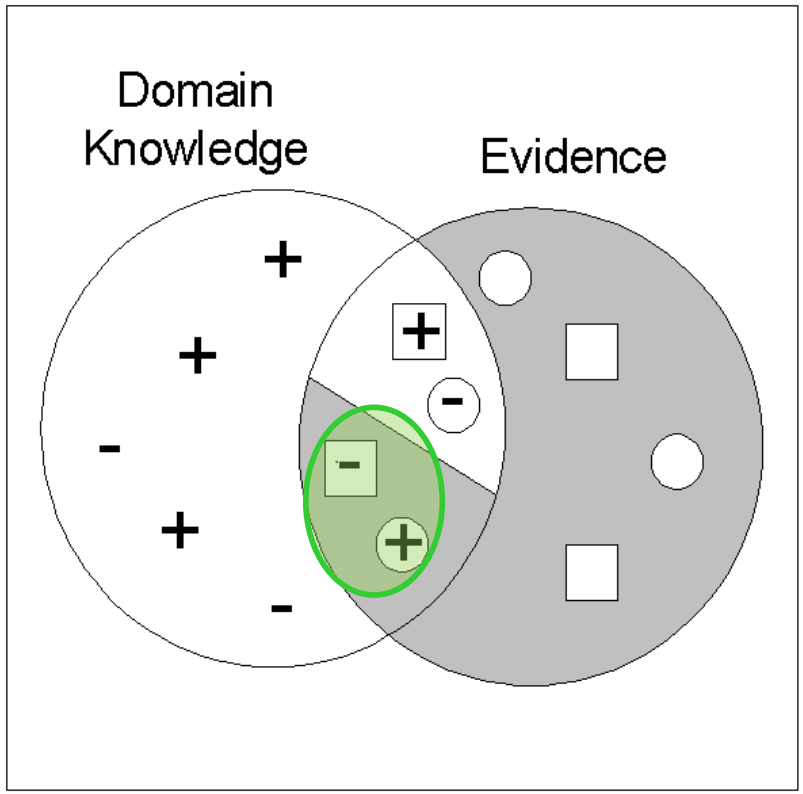
- Null Addition: a process of adding unrelated data to a given dataset, i.e., increase f_{00}.
  - If we are interested in finding the association between the words data and mining, should it matter if a collection of articles about fly fishing is added to a dataset of CS papers?
- Invariant measures:
  - support, cosine, Jaccard, etc
- Non-invariant measures:
  - correlation, Gini, mutual information, odds ratio, etc

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)

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)

Categorical and Continuous Attributes

Apriori and FP-Growth algorithms focus on *asymmetric binary variables*. How to apply association analysis on other types of variables?

Session Id	Country	Session Length (sec)	Number of Web Pages viewed	Gender	Browser Type	Buy
1	USA	982	8	Male	IE	No
2	China	811	10	Female	Chrome	No
3	USA	2125	45	Female	Mozilla	Yes
4	Germany	596	4	Male	IE	Yes
5	Australia	123	9	Male	Mozilla	No
…	…	…	…	…	…	…

Example of Association Rule:

$\{\text{Number of Pages} \in [5,10) \wedge (\text{Browser=Mozilla})\} \rightarrow \{\text{Buy = No}\}$

Handling Categorical Attributes

- Transform categorical attribute into asymmetric binary variables

- Introduce a new "item" for each distinct attribute-value pair
  - Example: replace Browser Type attribute with
    - Browser Type = Internet Explorer
    - Browser Type = Chrome
    - Browser Type = Mozilla

Handling Categorical Attributes

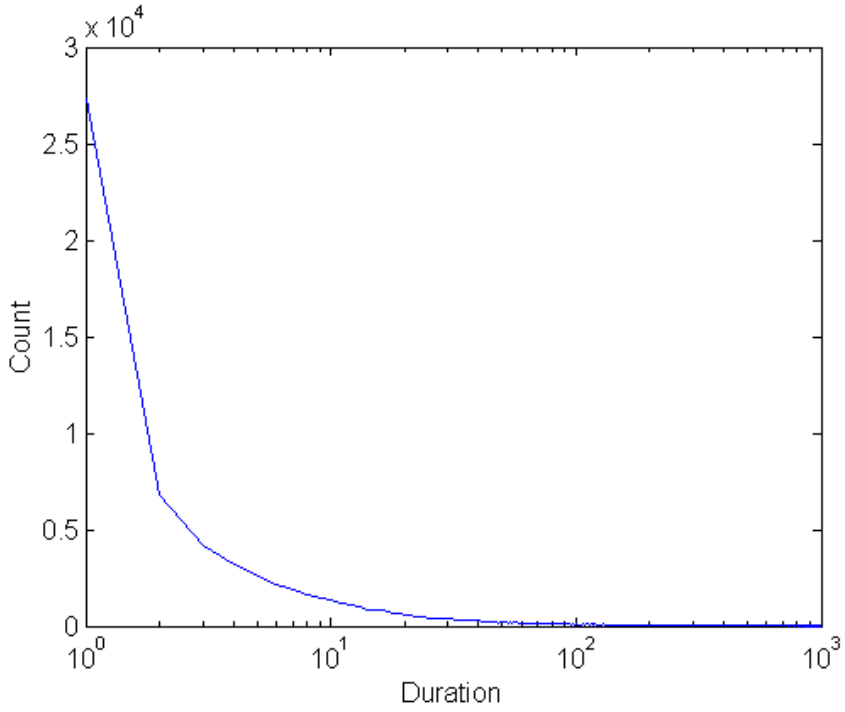
- Potential Issues
  - What if attribute has many possible values
    - Example: attribute country has more than 200 possible values
    - Many of the attribute values may have very low support
      - Potential solution: *Aggregate the low-support attribute values (if it's meaningful!)*
      - E.g., Illinois, Indiana, Iowa… → Midwest
  - What if distribution of attribute values is highly skewed
    - Example: 95% of the visitors have Buy = No
    - Most of the items will be associated with (Buy=No) item
      - Potential solution: *drop the highly frequent items because they usually do not carry new information*.

Handling Continuous Attributes

- *Quantitative Association Rules:* association rules that contains continuous attributed
- Different kinds of rules:
  - Age$\in$[21,35) $\wedge$ Salary$\in$[70k,120k) $\rightarrow$ Buy
  - Salary$\in$[70k,120k) $\wedge$ Buy $\rightarrow$ Age: μ=28, σ=4
- Different methods:
  - Discretization-based
  - Statistics-based
  - Non-discretization based
    - minApriori

Handling Continuous Attributes

- Use discretization
  - Group adjacent values into a number of intervals.
- Unsupervised:
  - Equal-width binning
  - Equal-depth binning
  - Clustering
- Supervised:



Attribute values, v

Class	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
Anomalous	0	0	20	10	20	0	0	0	0
Normal	150	100	0	0	0	100	100	150	100

bin_1 bin_2 bin_3

Discretization Issues

- Size of the discretized intervals affect support & confidence

  {Refund = No, (Income = \$51,250)} $\rightarrow$ {Cheat = No}

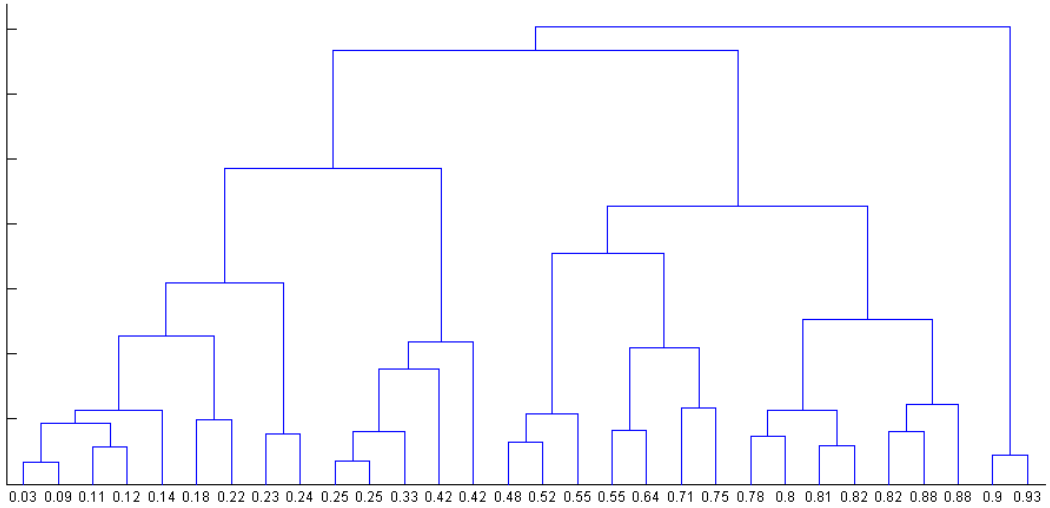
  {Refund = No, (60K $\leq$ Income $\leq$ 80K)} $\rightarrow$ {Cheat = No}

  {Refund = No, (0K $\leq$ Income $\leq$ 1B)} $\rightarrow$ {Cheat = No}

  - If intervals too small
    - may not have enough support
  - If intervals too large
    - may not have enough confidence
- Potential solution: use all possible intervals
  - Start with small interval and merge as appropriate.

Discretization Issues

- Execution time
  - If intervals contain n values, there are on average $O(n^2)$ possible ranges



- Too many rules

$\{\text{Refund} = \text{No}, (\text{Income} = \$51{,}250)\} \rightarrow \{\text{Cheat} = \text{No}\}$

$\{\text{Refund} = \text{No}, (51\text{K} \leq \text{Income} \leq 52\text{K})\} \rightarrow \{\text{Cheat} = \text{No}\}$

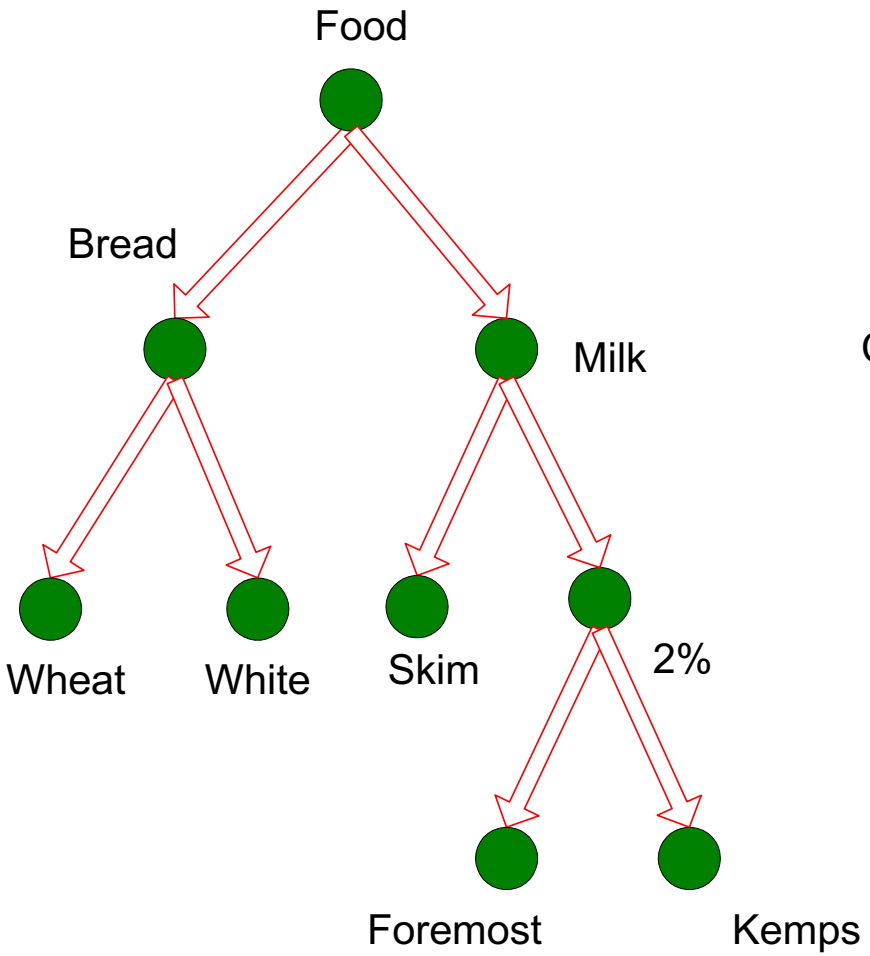
$\{\text{Refund} = \text{No}, (50\text{K} \leq \text{Income} \leq 60\text{K})\} \rightarrow \{\text{Cheat} = \text{No}\}$

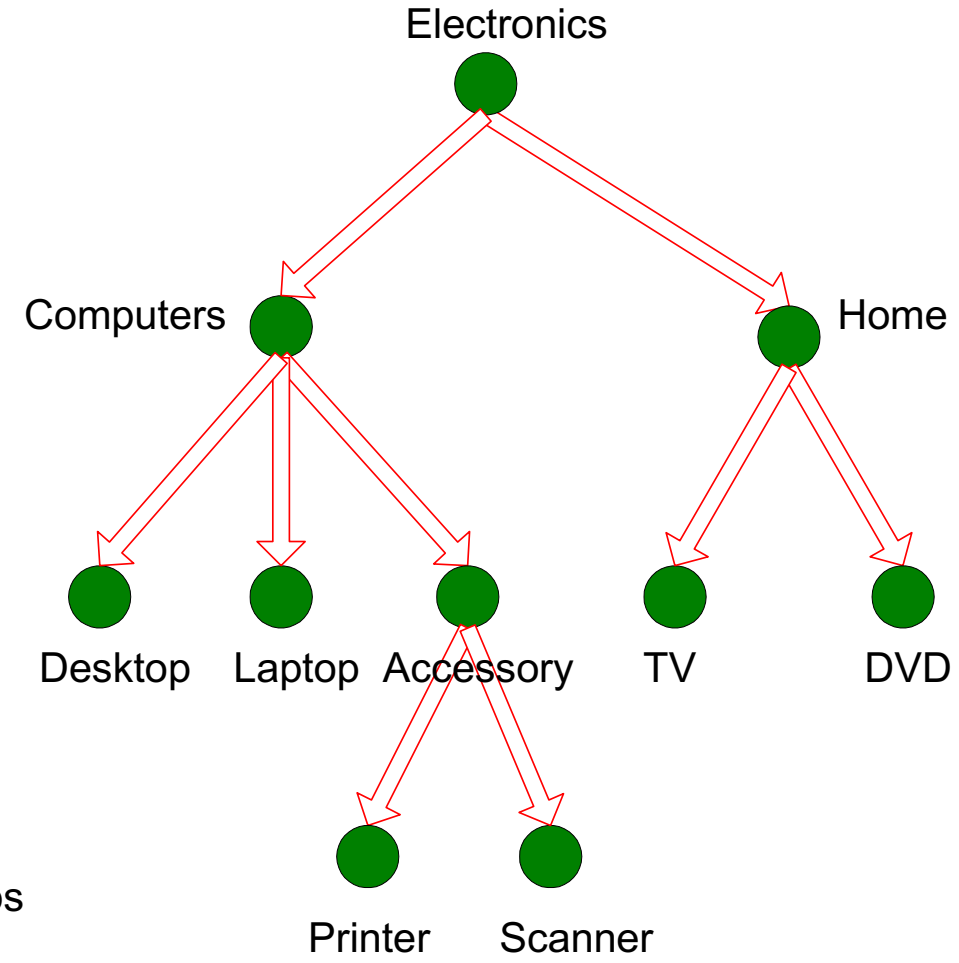
Approach by Srikant & Agrawal

- Preprocess the data
  - Discretize attribute using equi-depth partitioning
    - Use *partial completeness measure*, based on the amount of information loss after discretization, to determine number of partitions
    - Merge adjacent intervals as long as support is less than max-support
- Apply existing association rule mining algorithms
- Determine interesting rules in the output

Concept Hierarchy

- Items may be organized in a concept hierarchy.





~~Multi-level Association Rules~~

- Advantages for incorporating concept hierarchy into association rules:
  - Rules at lower levels may not have enough support to appear in any frequent itemsets
  - Rules at lower levels of the hierarchy are overly specific
    - e.g., skim milk $\rightarrow$ white bread,

      2% milk $\rightarrow$ wheat bread,

      skim milk $\rightarrow$ wheat bread, etc.

      are indicative of association between milk and bread
  - However, electronics $\rightarrow$ food may satisfy the support and confidence requirements, it may *over-generalize* the co-occurrence of items.

~~Multi-level Association Rules~~

- Approach 1:
  - Extend current association rule formulation by augmenting each transaction with higher level items

  Original Transaction: {skim milk, wheat bread}

  Augmented Transaction:
  {skim milk, wheat bread, *milk, bread, food*}
- Issues:
  - Items that reside at higher levels have much higher support counts
    - if support threshold is low, *too many frequent patterns involving items from the higher levels*
  - Increased dimensionality of the data and computational cost

~~Multi-level Association Rules~~

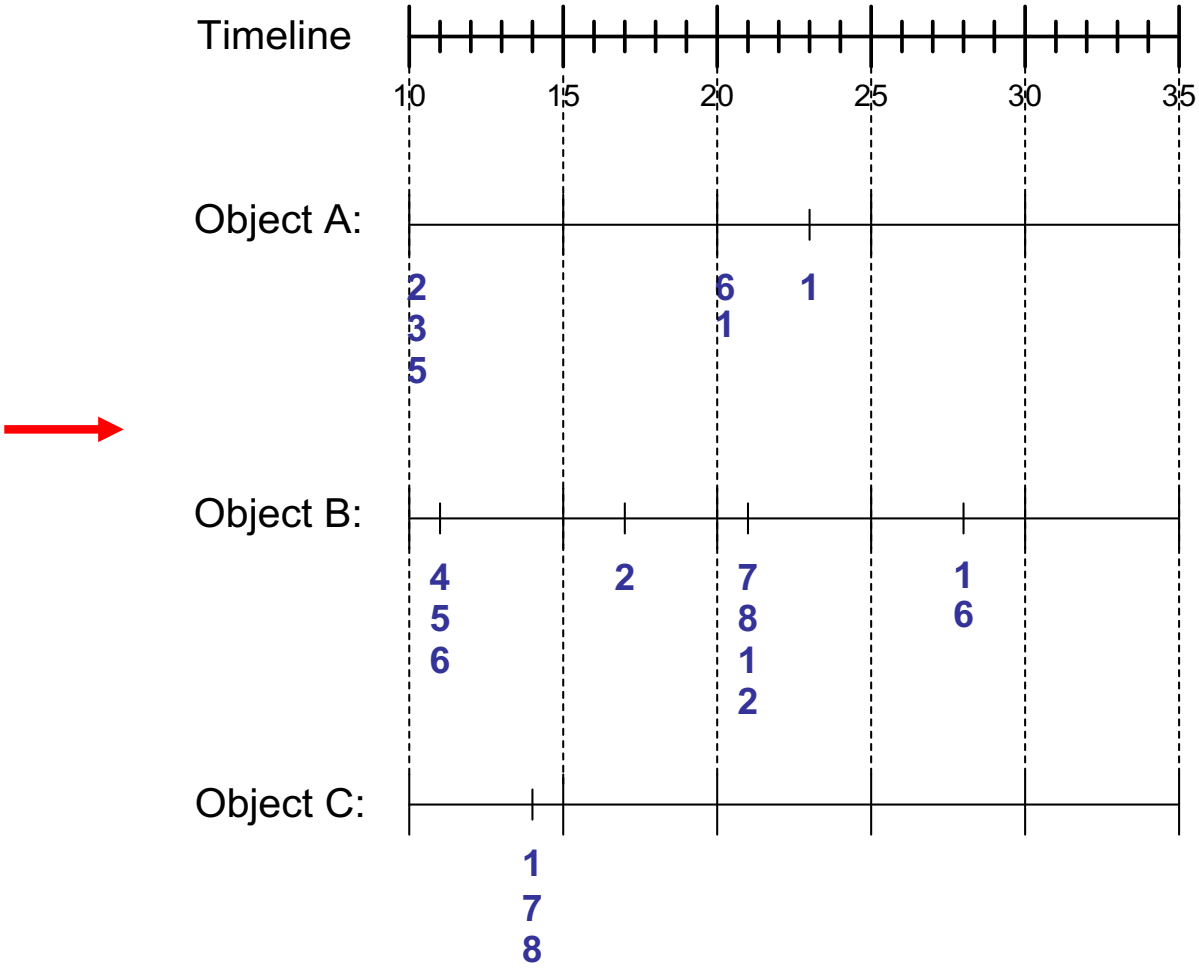
- Approach 2:
  - Generate frequent patterns at highest level first
  - Then, generate frequent patterns at the next highest level, and so on
- Issues:
  - I/O requirements will increase dramatically because we need to perform more passes over the data
  - *May miss some potentially interesting cross-level association patterns*

Sequence Data

- Many datasets contain temporal information, i.e., have inherent sequential nature.

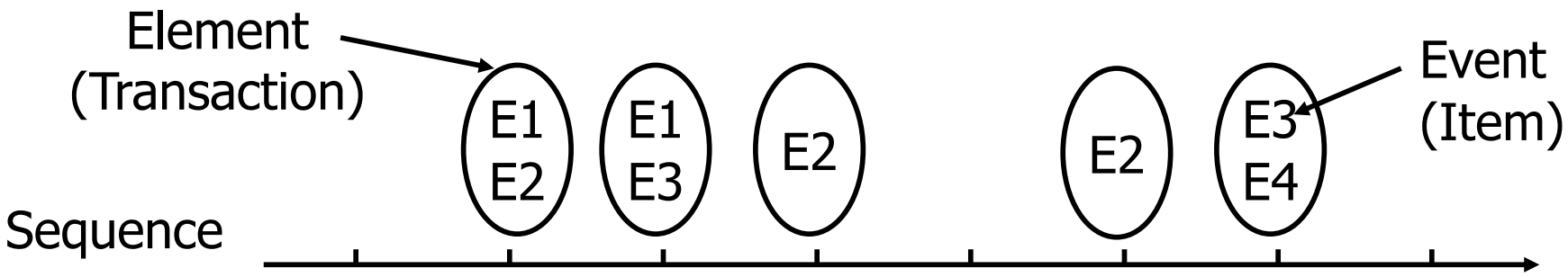
Sequence Database:

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



Examples of Sequence Data

Sequence Database	Sequence	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor after a single mouse click	Home page, index page, contact info, etc
Event data	History of events generated by a given sensor	Events triggered by a sensor at time t	Types of alarms generated by sensors
Genome sequences	DNA sequence of a particular species	An element of the DNA sequence	Bases A,T,G,C



Formal Definition of a Sequence

- A *sequence* is an ordered list of elements (transactions)

$$s = < e_1\ e_2\ e_3\ \ldots >$$

  - Each element contains a collection of events (items)

$$e_i = \{i_1, i_2, \ldots, i_k\}$$

  - Each element is attributed to a specific time or location

- Length of a sequence, |s|, is given by the number of elements of the sequence

- A *k-sequence* is a sequence that contains k *events* (items)

Examples of Sequence

- Web page viewing sequence:

  < {Homepage} {Electronics} {Digital Cameras} {Canon Digital Camera} {Shopping Cart} {Order Confirmation} {Return to Shopping} >

- Sequence of initiating events causing the nuclear accident at 3-mile Island:

  (http://stellar-one.com/nuclear/staff_reports/summary_SOE_the_initiating_event.htm)

  < {clogged resin} {outlet valve closure} {loss of feedwater} {condenser polisher outlet valve shut} {booster pumps trip} {main waterpump trips} {main turbine trips} {reactor pressure increases}>

- Sequence of library books checked out by a user:

  <{Fellowship of the Ring, The Two Towers} {Return of the King}>

Formal Definition of a Subsequence

- A *subsequence* $\langle a_1\ a_2 \ldots a_n \rangle$ is contained in another sequence $\langle b_1\ b_2 \ldots b_m \rangle$ $(m \geq n)$ if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i1}$, …, $a_n \subseteq b_{in}$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	Yes
< {1,2} {3,4} >	< {1} {2} >	No
< {2,4} {2,4} {2,5} >	< {2} {4} >	Yes

- The *support of a subsequence* w is defined as the fraction of data sequences that contain w
- A *sequential pattern* is a frequent subsequence (i.e., a subsequence whose support is ≥ *minsup*)

Sequential Pattern Mining: Definition

- Given:
  - a database of sequences
  - a user-specified minimum support threshold, *minsup*
- Task:
  - Find all subsequences with support ≥ *minsup*

Sequential Pattern Mining: Example

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

Minsup = 50%

Examples of Frequent Subsequences:

< {1,2} >	s=60%
< {2,3} >	s=60%
< {2,4}>	s=80%
< {3} {5}>	s=80%
< {1} {2} >	s=80%
< {2} {2} >	s=60%
< {1} {2,3} >	s=60%
< {2} {2,3} >	s=60%
< {1,2} {2,3} >	s=60%

Sequential Pattern Mining: Challenge

- Given a sequence: <{a b} {c d e} {f} {g h i}>
  - Examples of subsequences:

    <{a} {c d} {f} {g} >, < {c d e} >, < {b} {g} >, etc.

- How many k-subsequences can be extracted from a given n-sequence?

<{a b} {c d e} {f} {g h i}> n = 9

k=4: Y _ _ Y Y _ _ _ Y

<{a} {d e} {i}>

Answer :

$$\binom{n}{k} = \binom{9}{4} = 126$$

Extracting Sequential Patterns

- Given n events: $i_1, i_2, i_3, \ldots, i_n$
- Candidate 1-subsequences:

  $\langle\{i_1\}\rangle, \langle\{i_2\}\rangle, \langle\{i_3\}\rangle, \ldots, \langle\{i_n\}\rangle$
- Candidate 2-subsequences:

  $\langle\{i_1, i_2\}\rangle, \langle\{i_1, i_3\}\rangle, \ldots, \langle\{i_1\}\ \{i_1\}\rangle, \langle\{i_1\}\ \{i_2\}\rangle, \ldots, \langle\{i_1\}\ \{i_n\}\rangle, \ldots, \langle\{i_n\}\ \{i_n\}\rangle$
- Candidate 3-subsequences:

  $\langle\{i_1, i_2, i_3\}\rangle, \langle\{i_1, i_2, i_4\}\rangle, \ldots, \langle\{i_1, i_2\}\ \{i_1\}\rangle, \langle\{i_1, i_2\}\ \{i_2\}\rangle, \ldots,$
  $\langle\{i_1\}\ \{i_1, i_2\}\rangle, \langle\{i_1\}\ \{i_1, i_3\}\rangle, \ldots, \langle\{i_1\}\ \{i_1\}\ \{i_1\}\rangle, \langle\{i_1\}\ \{i_1\}\ \{i_2\}\rangle, \ldots$

How is the number of candidate subsequences compared to than the number of candidate itemsets?

Generalized Sequential Pattern (GSP)

1. Initialization
   - Make the first pass over the sequence database D to yield all the 1-element frequent sequences
2. Repeat until no new frequent sequences found
   - **Candidate Generation**:
     - Merge pairs of frequent subsequences found in the (k-1)*th* pass to generate candidate sequences that contain k items
   - **Candidate Pruning**:
     - Prune candidate *k*-sequences that contain infrequent (*k-1)*-subsequences
   - **Support Counting**:
     - Make a new pass over the sequence database D to find the support for these candidate sequences
   - **Candidate Elimination**:
     - Eliminate candidate *k*-sequences whose actual support is less than *minsup*

Candidate Generation

- Base case (k=2):
  - Merging two frequent 1-sequences $<\{i_1\}>$ and $<\{i_2\}>$ will produce the following candidate 2-sequences:

    $<\{i_1\}\ \{i_1\}>$ $<\{i_1\}\ \{i_2\}>$ $<\{i_2\}\ \{i_1\}>$ $<\{i_2\}\ \{i_2\}>$ $<\{i_1\ i_2\}>$
- General case (k>2):
  - A frequent *(k-1)*-sequence w_1 is merged with another frequent *(k-1)*-sequence w_2 to produce a candidate *k*-sequence if the subsequence obtained by removing the first event in w_1 is the same as the subsequence obtained by removing the last event in w_2
    - The resulting candidate after merge is given by the sequence w_1 extended with the last event of w_2 (which remains how it was in w_2)
      - If the last two events in w_2 belong to the same element, then the last event in w_2 becomes part of the last element in w_1
      - Otherwise, the last event in w_2 becomes a separate element appended to the end of w_1

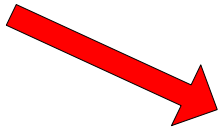
Candidate Generation Examples

- Merging the sequences w_1=<{1} {2 3} {4}> and w_2 =<{2 3} {4 5}> will produce the candidate sequence < {1} {2 3} {4 5}> because the last two events in w_2 (4 and 5) belong to the same element
- Merging the sequences w_1=<{1} {2 3} {4}> and w_2 =<{2 3} {4} {5}> will produce the candidate sequence < {1} {2 3} {4} {5}> because the last two events in w_2 (4 and 5) not belong to same element
- We do not have to merge the sequences w_1 =<{1} {2 6} {4}> and w_2 =<{1} {2} {4 5}> to produce the candidate < {1} {2 6} {4 5}> because if the latter is a viable candidate, then it can be obtained by merging w_1 with < {2 6} {4 5}>

GSP Example

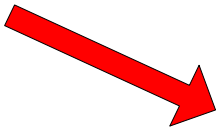
Frequent 3-sequences

< {1} {2} {3} >
< {1} {2 5} >
< {1} {5} {3} >
< {2} {3} {4} >
< {2 5} {3} >
< {3} {4} {5} >
< {5} {3 4} >



Candidate Generation

< {1} {2} {3} {4} >
< {1} {2 5} {3} >
< {1} {5} {3 4} >
< {2} {3} {4} {5} >
< {2 5} {3 4} >



Candidate Pruning

< {1} {2 5} {3} >

Sequence Mining with Time Constraints

- Student A:

  <{Statistics} {Database Systems} {Data Mining}>

- Student B:

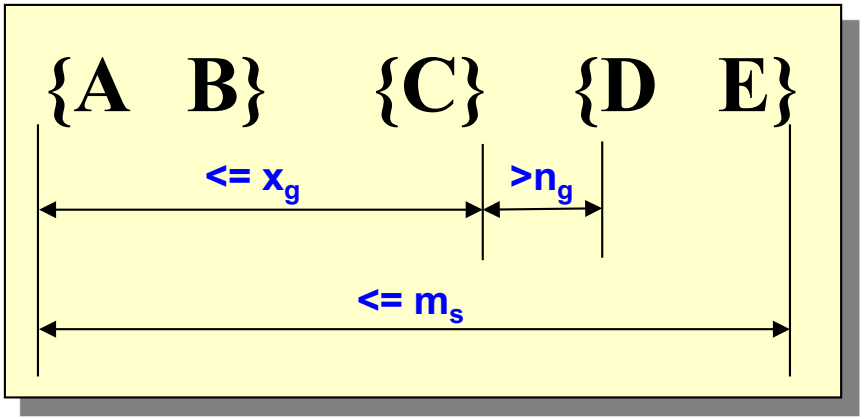
  <{Database Systems} {Statistics} {Data Mining}>

- Students taking Database before taking Data Mining:

  <{Database Systems} {Data Mining}>

  - How should we count support of this sequence?

Timing Constraints (I)



Each element has a time window: [l, u]

x_g: max-gap

n_g: min-gap

m_s: maximum span

$x_g = 2$, $n_g = 0$, $m_s = 4$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {4,7} {4,5} {8} >	< {6} {5} >	Yes
< {1} {2} {3} {4} {5}>	< {1} {4} >	No
< {1} {2,3} {3,4} {4,5}>	< {2} {3} {5} >	Yes
< {1,2} {3} {2,3} {3,4} {2,4} {4,5}>	< {1,2} {5} >	No

Mining Sequential Patterns with Timing Constraints

- Approach 1:
  - Mine sequential patterns without timing constraints
  - Postprocess the discovered patterns
  - *Time consuming!*
- Approach 2:
  - Modify GSP to directly prune candidates that violate timing constraints
  - Question:
    - *Does Apriori principle still hold?*

Apriori for Sequence Data (Bad News!)

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

Suppose:

x_g = 1 (max-gap)

n_g = 0 (min-gap)

m_s = 5 (maximum span)

minsup = 60%

<{2} {5}> support = 40%

but

<{2} {3} {5}> support = 60%

Violate the Apriori principle!

- **Problem exists because of max-gap**
- **No problem if max-gap is infinite**

Contiguous Subsequences (Good News!)

- *s* is a *contiguous subsequence* of

  $w = \langle e_1 \rangle \langle e_2 \rangle \ldots \langle e_k \rangle$

  if any of the following conditions hold:

  1. *s* is obtained from *w* by deleting an item from either e_1 or e_k
  2. *s* is obtained from *w* by deleting an item from any element e_i that contains more than 2 items
  3. *s* is a contiguous subsequence of *s'* and *s'* is a contiguous subsequence of *w* (recursive definition)

- Examples: s = < {1} {2} >
  - is a contiguous subsequence of
    < {1} {2 3}>, < {1 2} {2} {3}>, and < {3 4} {1 2} {2 3} {4} >
  - is not a contiguous subsequence of
    < {1} {3} {2}> and < {2} {1} {3} {2}>

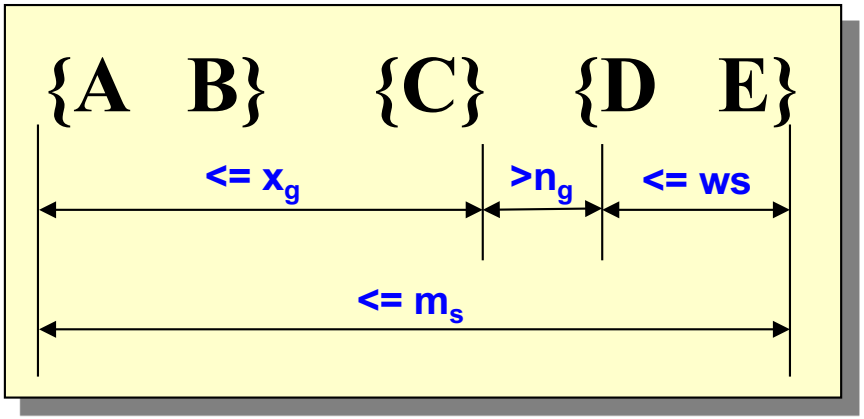
Modified Candidate Pruning Step

- Without maxgap constraint:
  - A candidate k-sequence is pruned if at least one of its (k-1)-subsequences is infrequent

- With maxgap constraint:
  - A candidate *k*-sequence is pruned if at least one of its *contiguous* (*k-1*)-subsequences is infrequent

If we cannot use the original Apriori principle to prune candidates (due to maxgap), use the modified Apriori to prune some unqualified candidates!

Timing Constraints (II)



x_g: max-gap

n_g: min-gap

ws: window size

m_s: maximum span

x_g = 2, n_g = 0, ws = 1, m_s= 5

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {4,7} {4,6} {8} >	< {3} {5} >	No
< {1} {2} {3} {4} {5}>	< {1,2} {3} >	Yes
< {1,2} {2,3} {3,4} {4,5}>	< {1,2} {3,4} >	Yes

Modified Support Counting Step

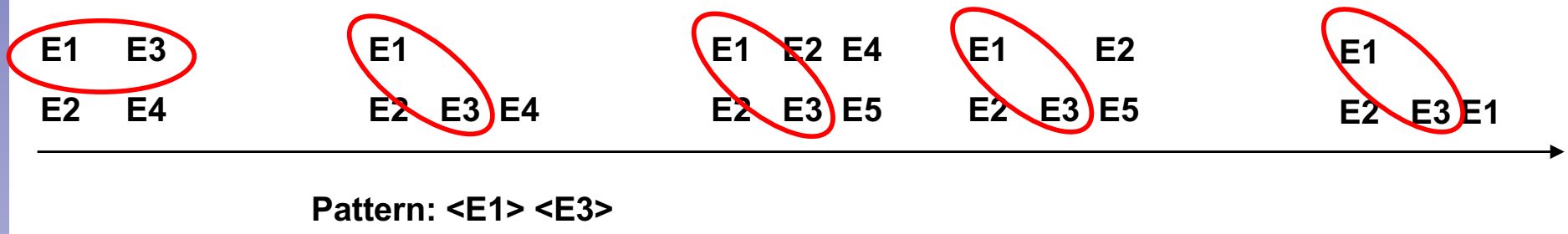
- Given a candidate pattern: <{a, c}>
  - Any data sequences that contain

    <… {a c} … >,
    <… {a} … {c}…> (where time({c}) – time({a}) ≤ ws)
    <…{c} … {a} …> (where time({a}) – time({c}) ≤ ws)

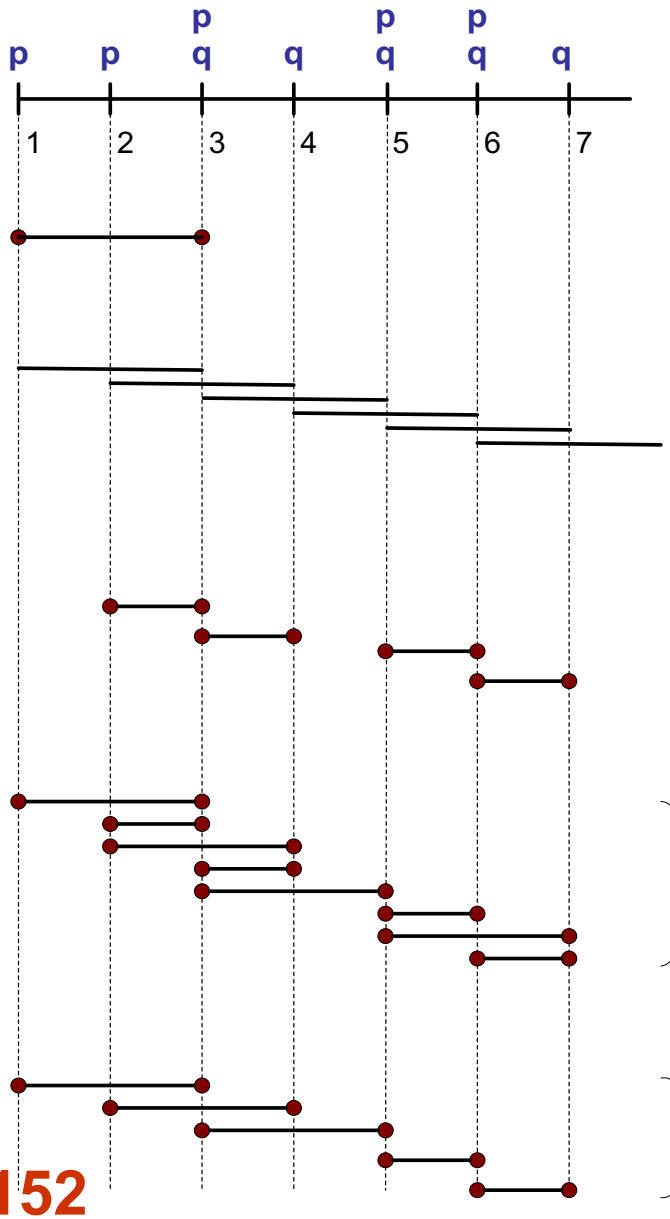
    will contribute to the support count of candidate pattern

Frequent Episode Mining

- In some domains, we may have only one very long time series
  - Example:
    - monitoring network traffic events for attacks
    - monitoring telecommunication alarm signals
- Goal is to find frequent sequences of events in the time series



General Support Counting Schemes



Sequence: (p) (q)

Method	Support Count
COBJ	1
CWIN	6
CMINWIN	4
CDIST_O	8
CDIST	5

Assume:

x_g = 2 (max-gap)

n_g = 0 (min-gap)

ws = 0 (window size)

m_s = 2 (maximum span)

COBJ: count one occurrence per object.
CWIN: one occurrence per sliding window (i.e., maxspan)
CMINWIN: number based on minimum sliding window
CDIST_O: distinct occurrence (with overlap allowed)
CDIST: distinct occurrence (with no overlap allowed)