MSCIT-MSCBD 5210/5002 : Knowledge Discovery and Data Mining

Acknowledgement: Slides modified by Dr. Lei Chen based on the slides provided by Jiawei Han, Micheline Kamber, and Jian Pei

And slides provide by Raymond Wong and Tan, Steinbach, Kumar

Association Rule Mining

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

Market-Basket transactions

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:
    $\{\text{Milk}, \text{Diaper}\} \rightarrow \{\text{Beer}\}$

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

Example:

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

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

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

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

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

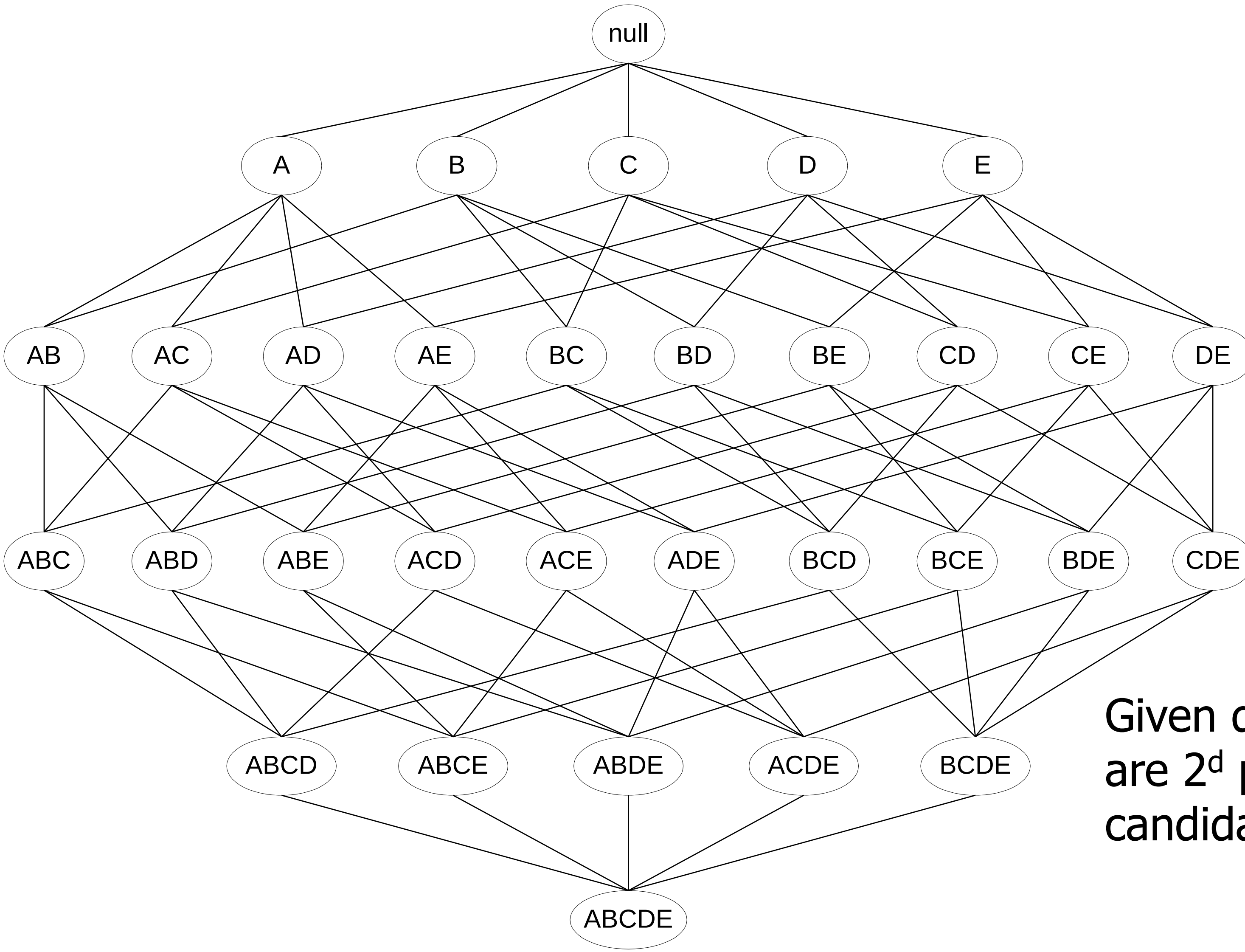
Observations:

- 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
- 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

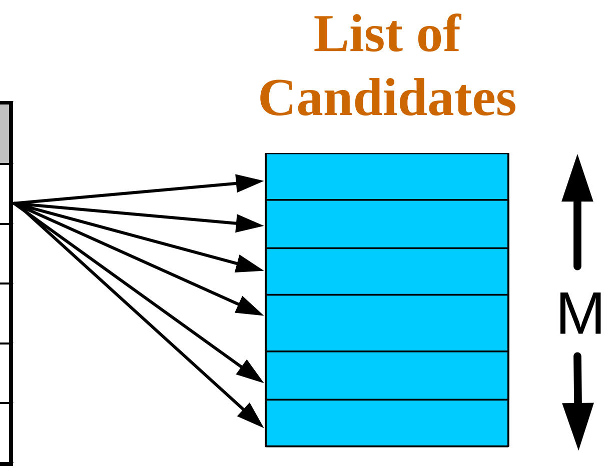
- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

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

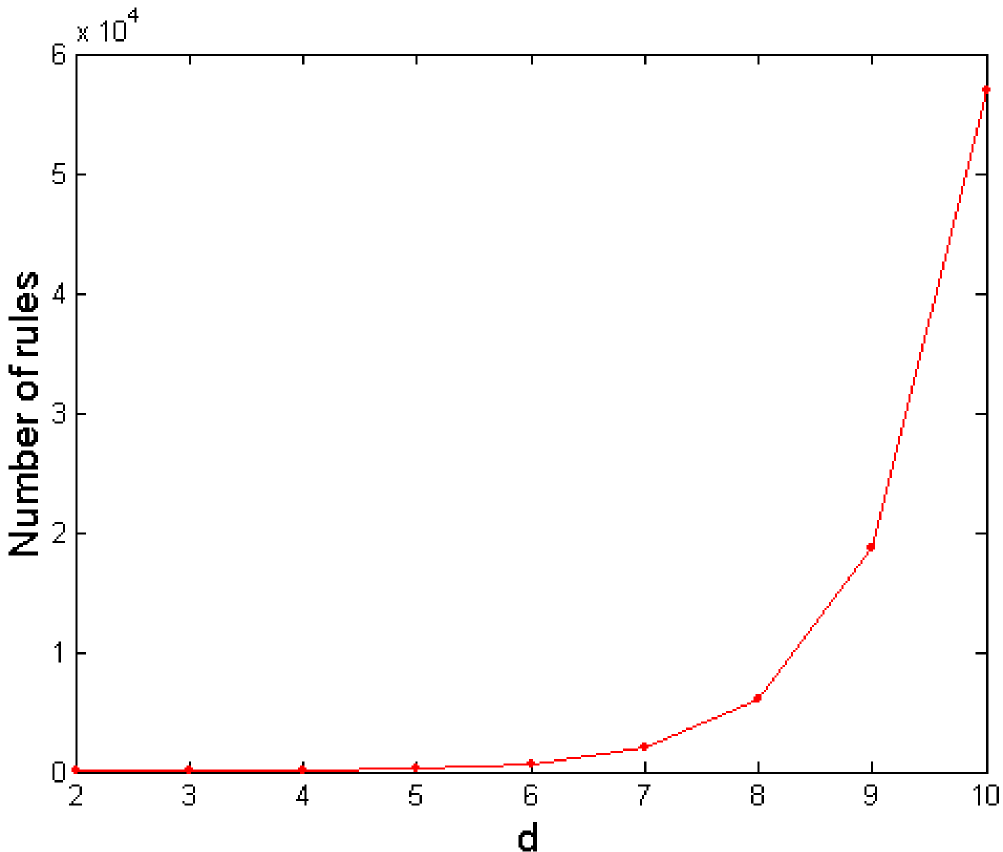
N

w



Computational Complexity

- Given d unique items:
  - Total number of itemsets = 2^d
  - Total number of possible association rules:



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

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

If d=6, R = 602 rules

Frequent Itemset Generation Strategies

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

Reducing Number of Candidates

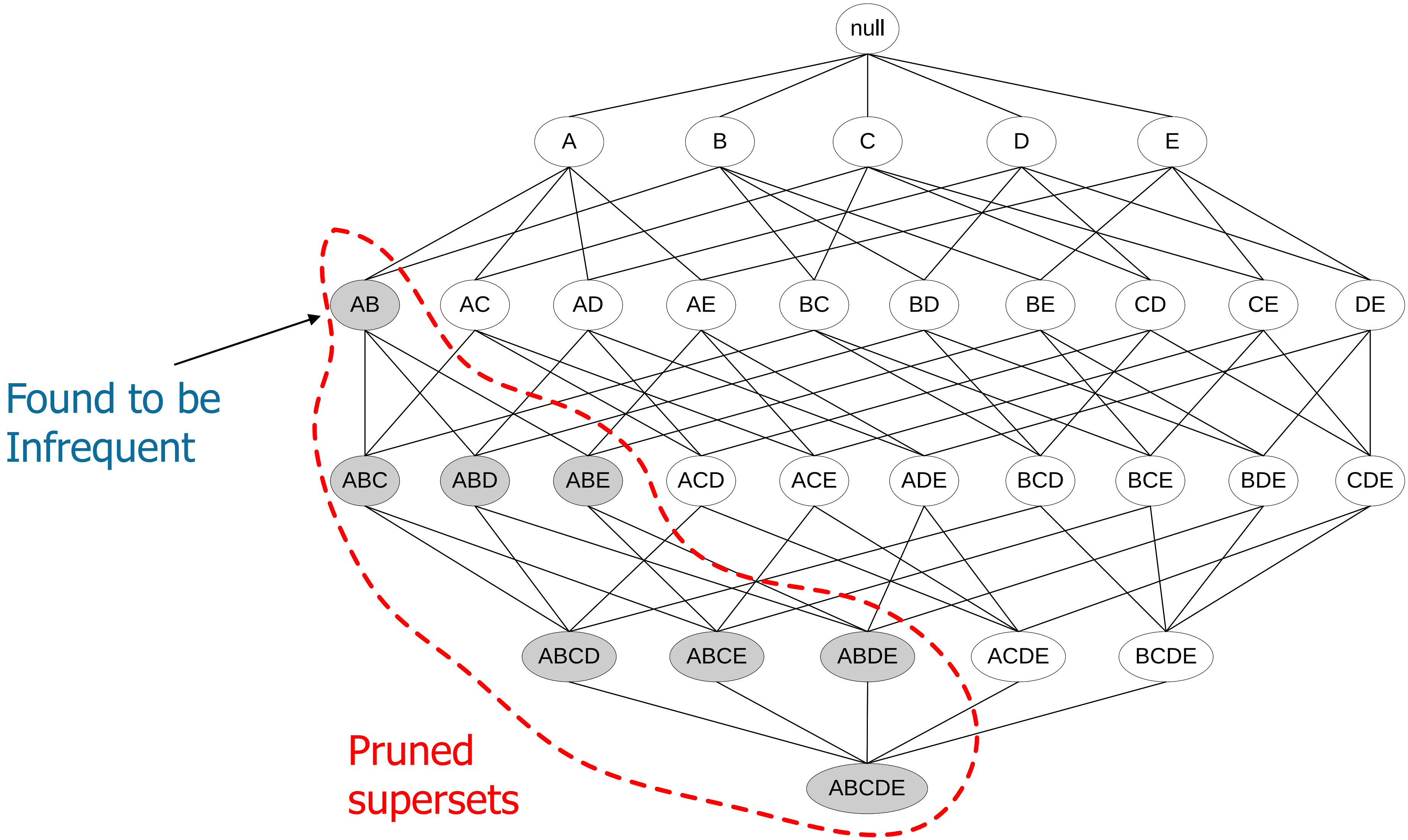
- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent

- Apriori principle holds due to the following property of the support measure:

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

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

Illustrating Apriori Principle



Illustrating Apriori Principle

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

Items (1-itemsets)

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

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

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,

$$6 + 6 + 1 = 13$$

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

Apriori: A Candidate Generation & Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example

$Sup_{min} = 2$

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

Reducing Number of Comparisons

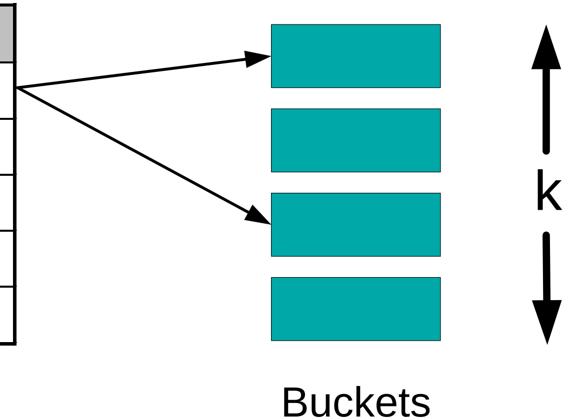
- Candidate counting:
  - Scan the database of transactions to determine the support of each candidate itemset
  - To reduce the number of comparisons, store the candidates in a hash structure
    - Instead of matching each transaction against every candidate, match it against candidates contained in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



Generate Hash Tree

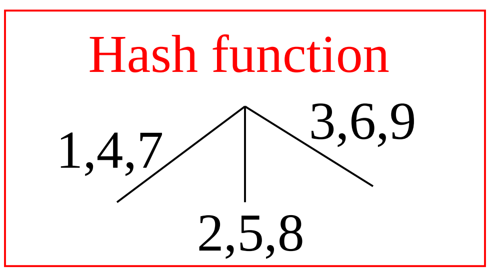
Suppose you have 15 candidate itemsets of length 3:

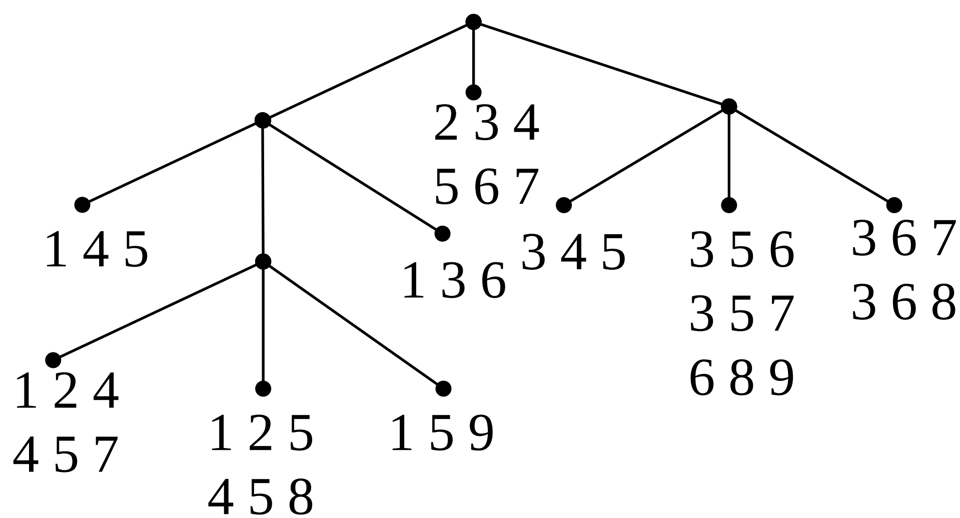
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

You need:

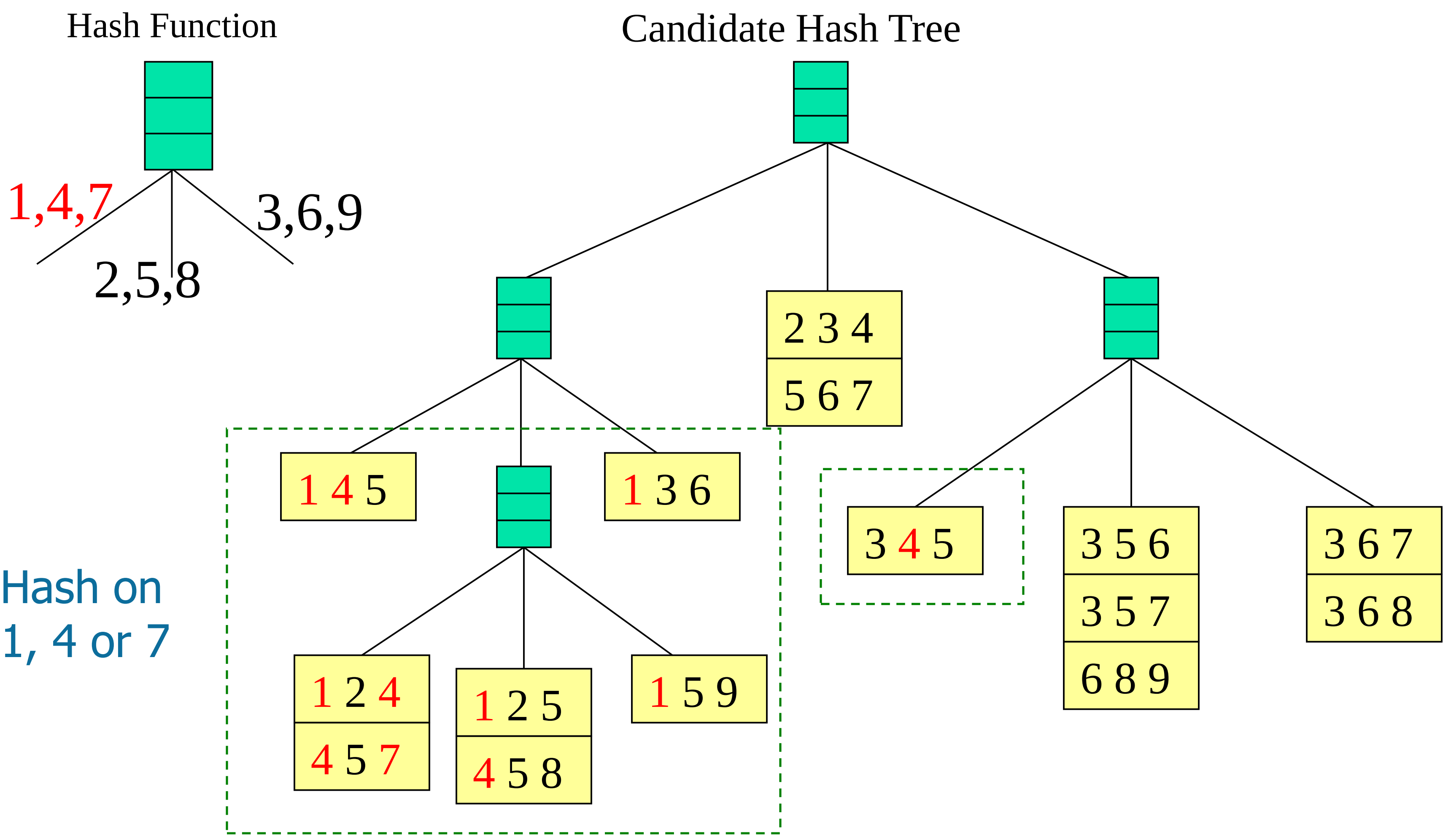
• Hash function

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



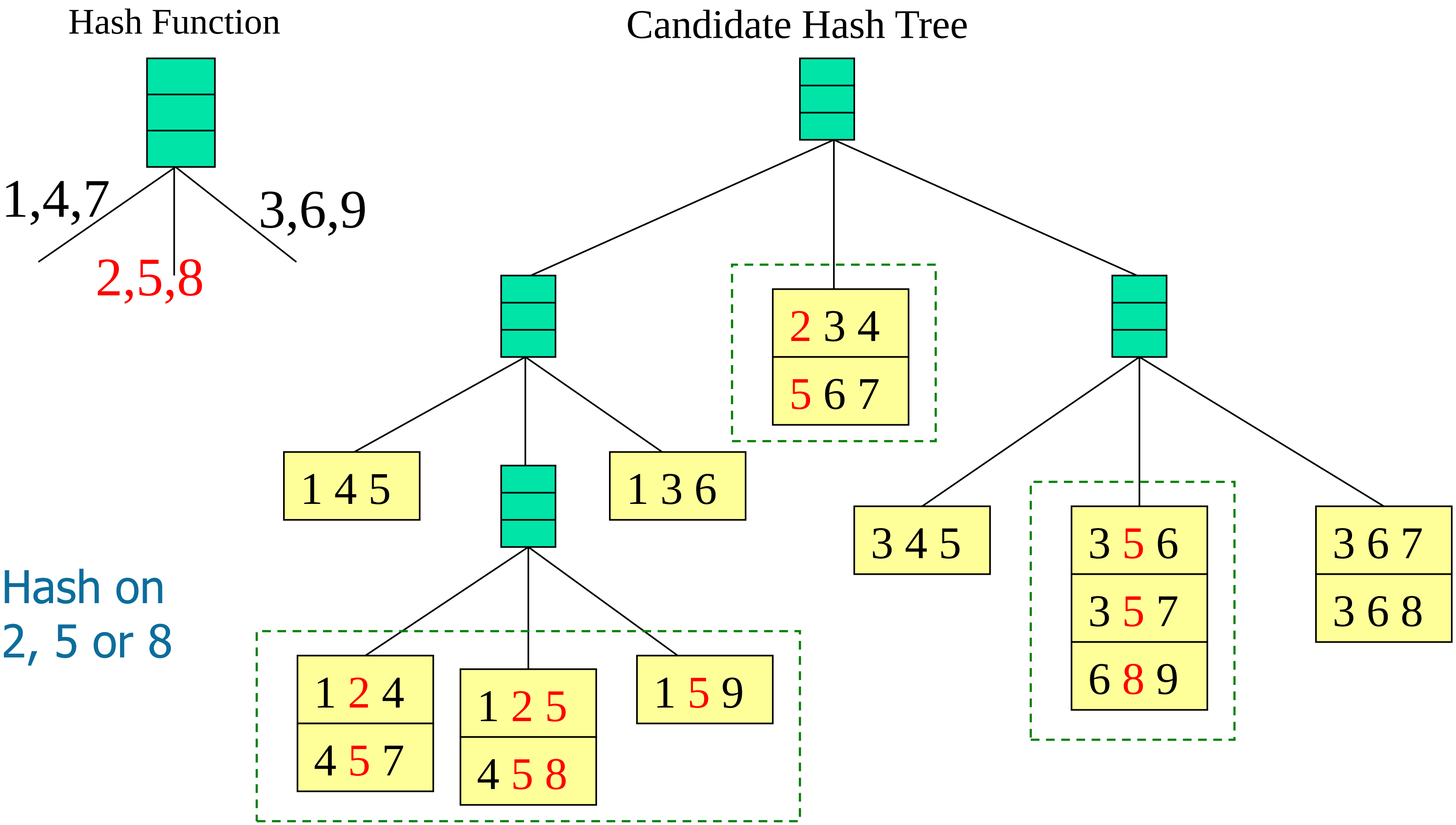


Association Rule Discovery: Hash tree



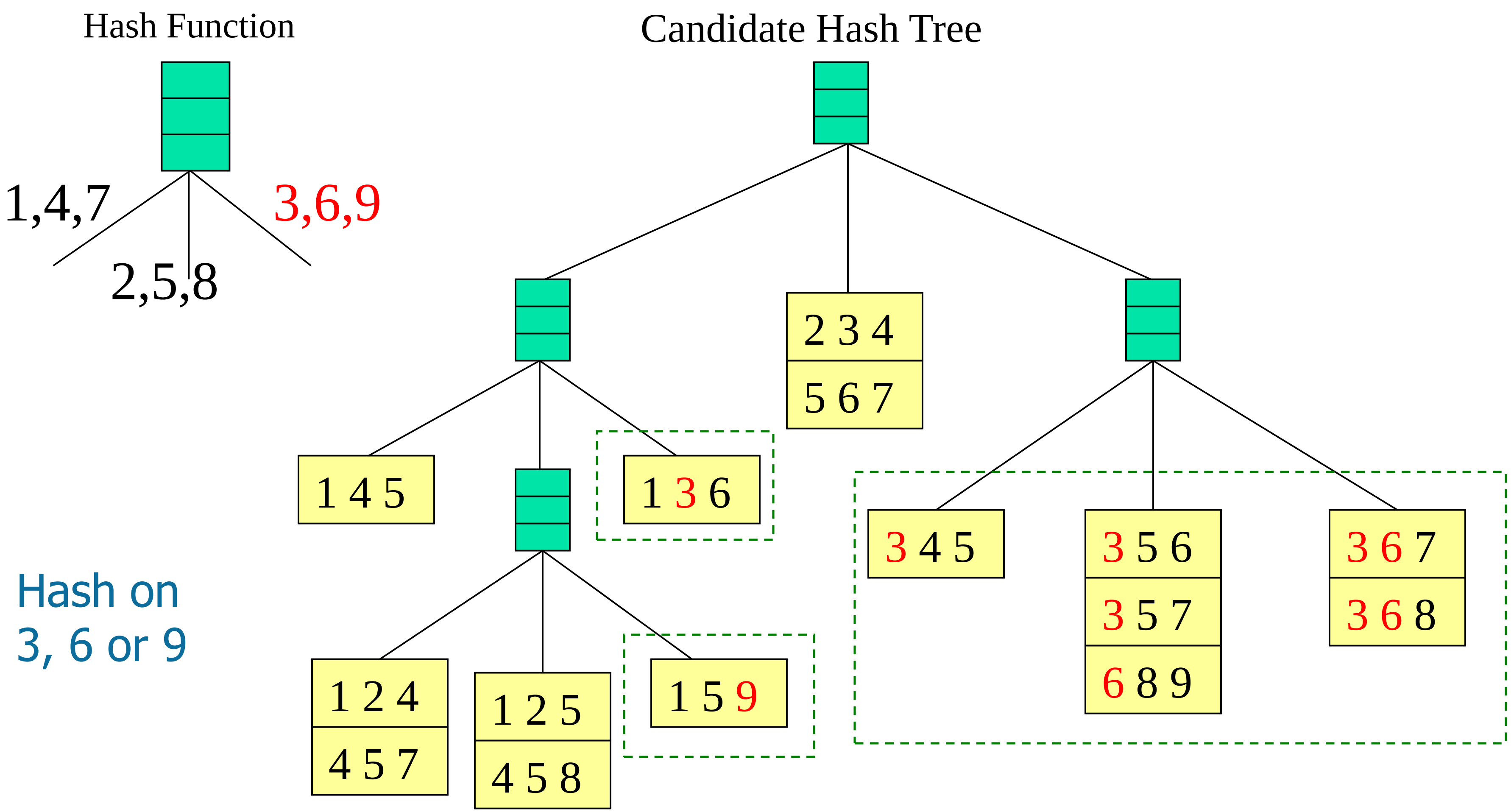
Hash on
1, 4 or 7

Association Rule Discovery: Hash tree



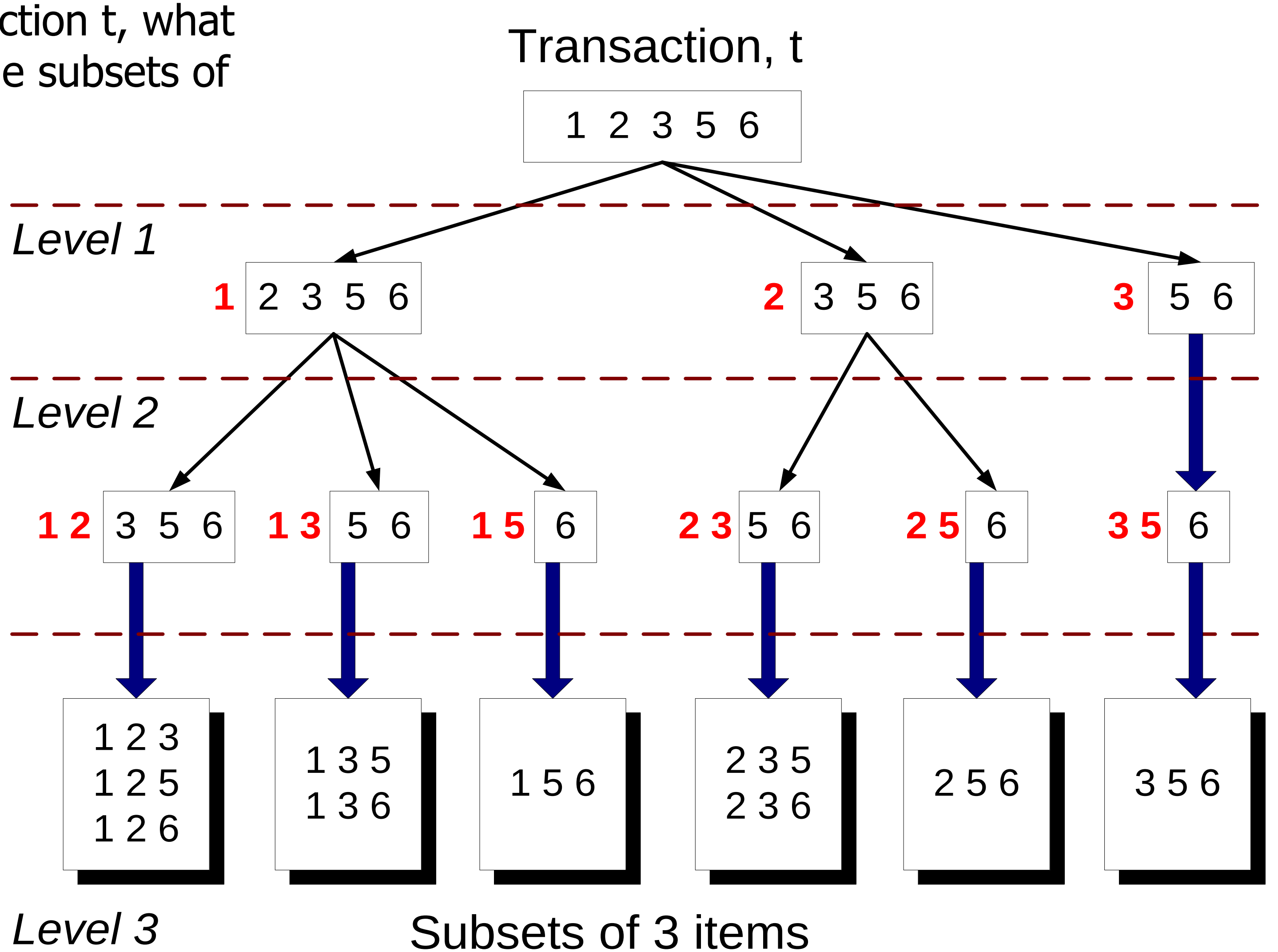
Hash on
2, 5 or 8

Association Rule Discovery: Hash tree

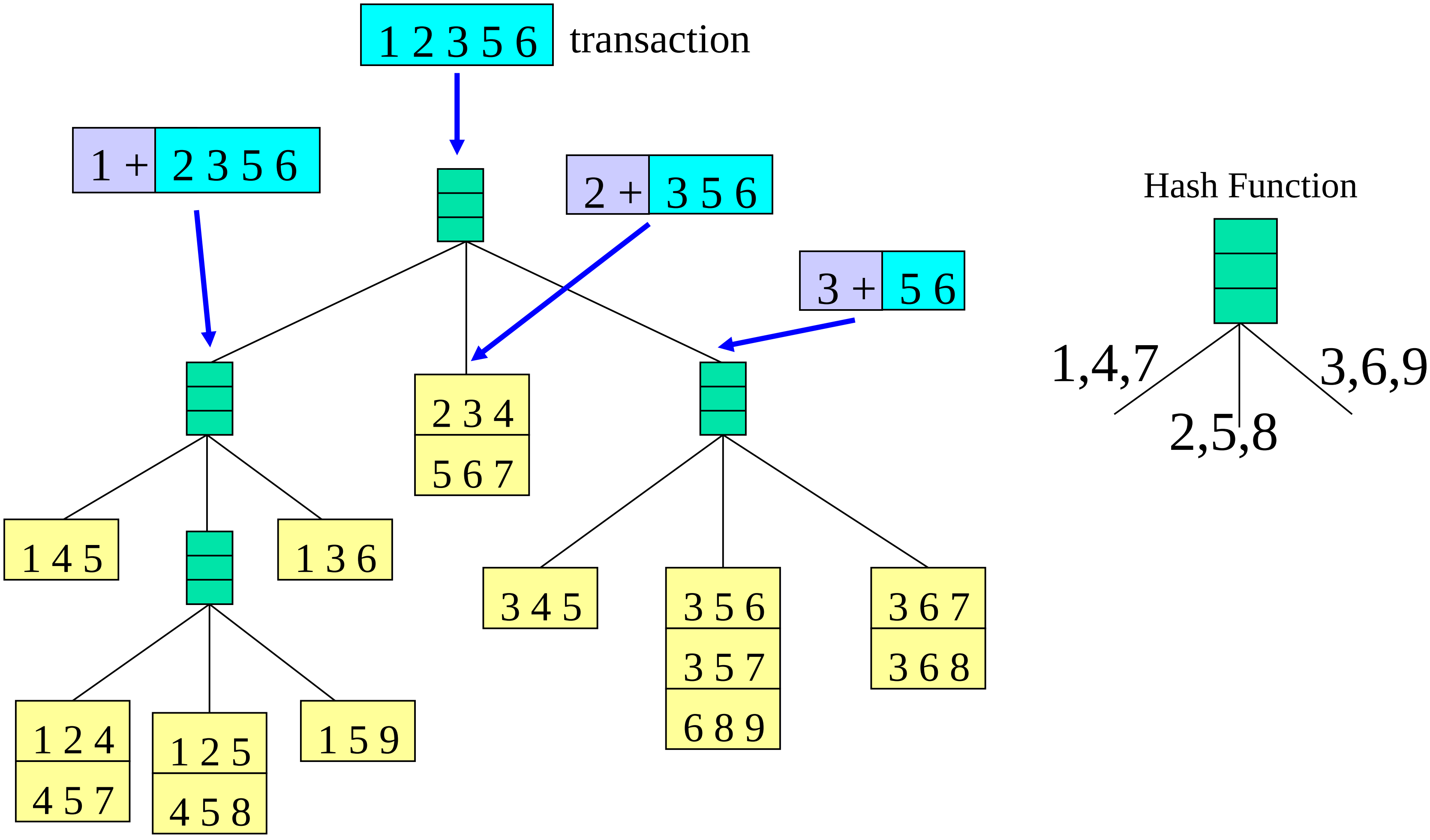


Subset Operation

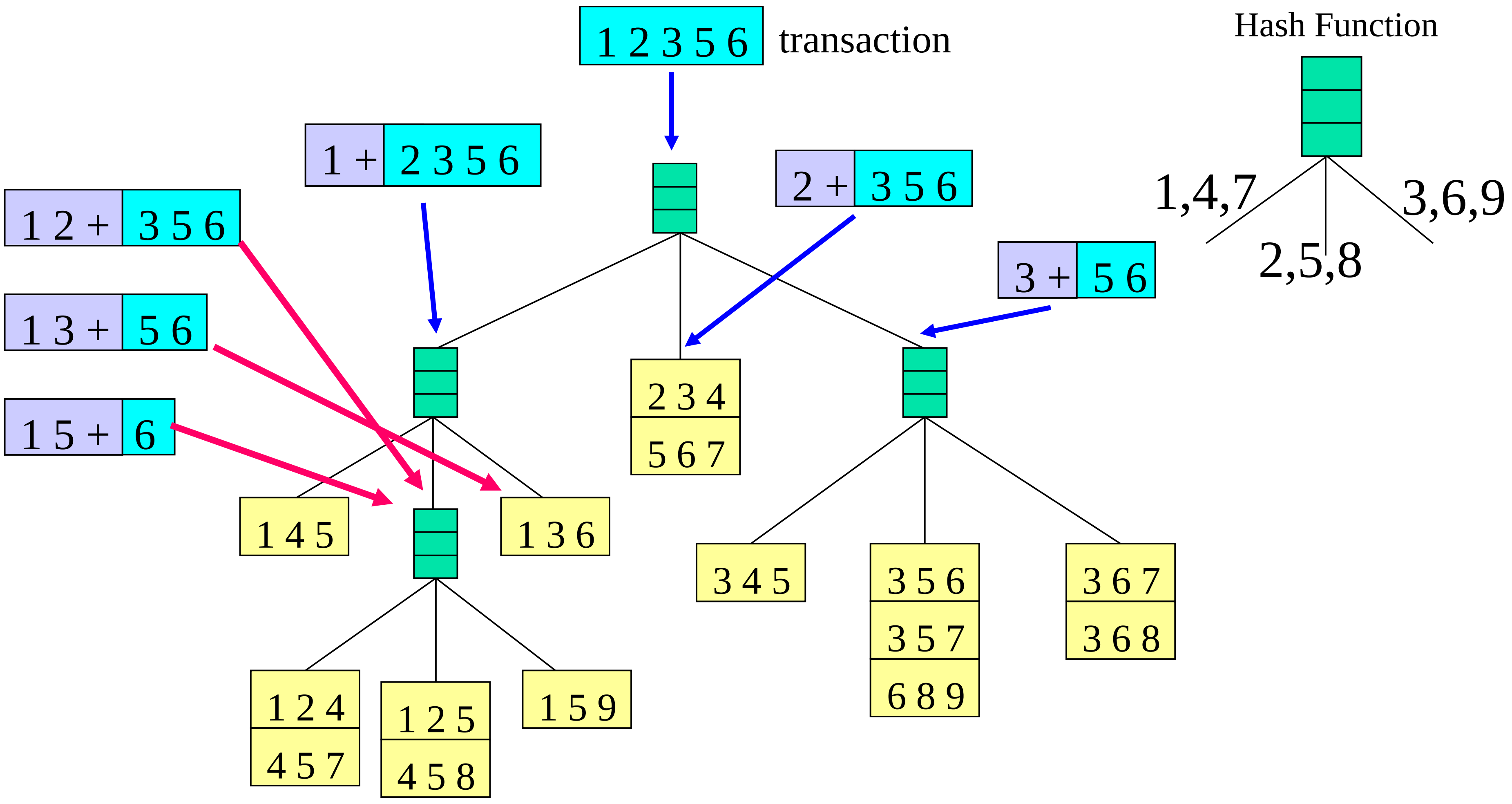
Given a transaction t, what are the possible subsets of size 3?



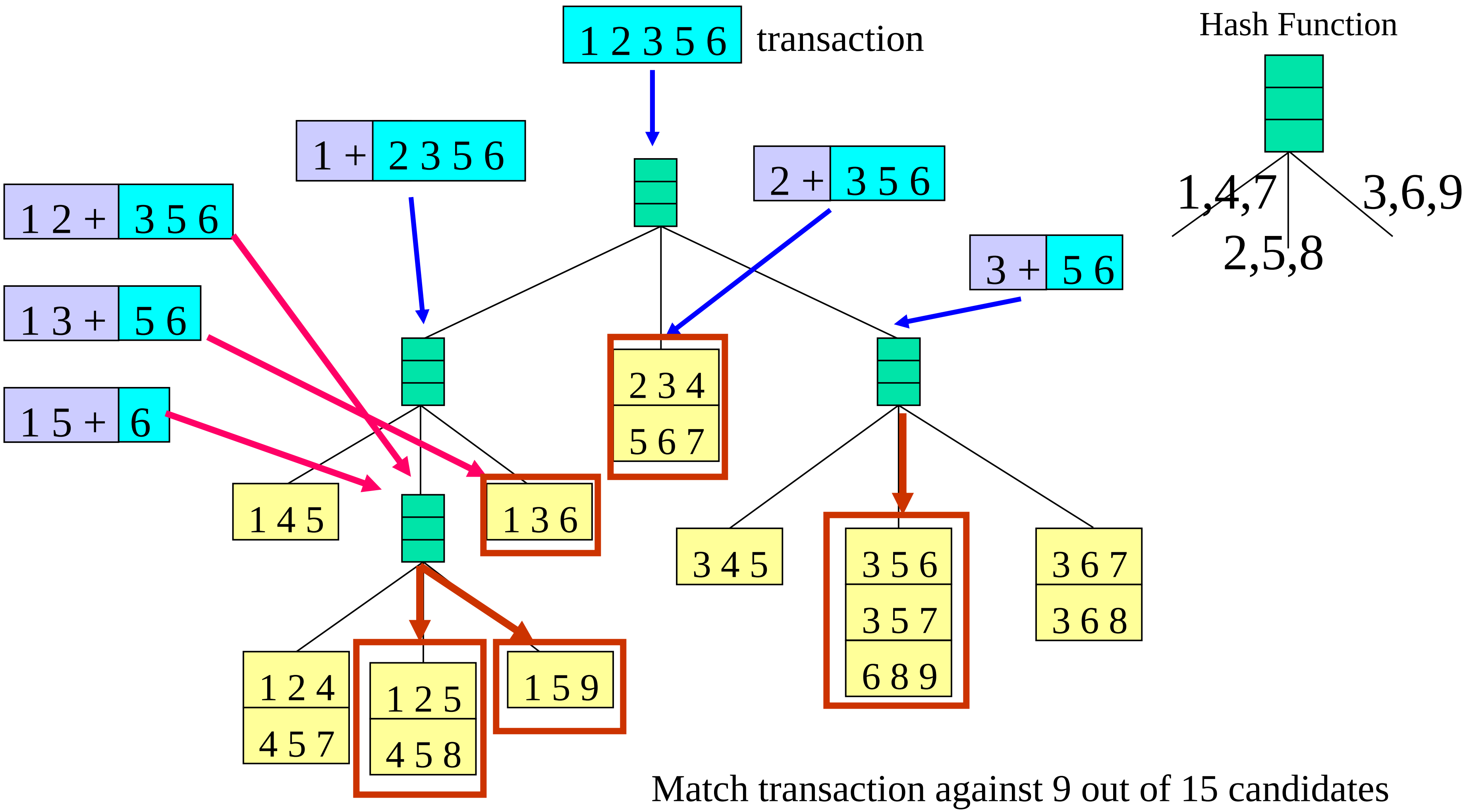
Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Match transaction against 9 out of 15 candidates

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

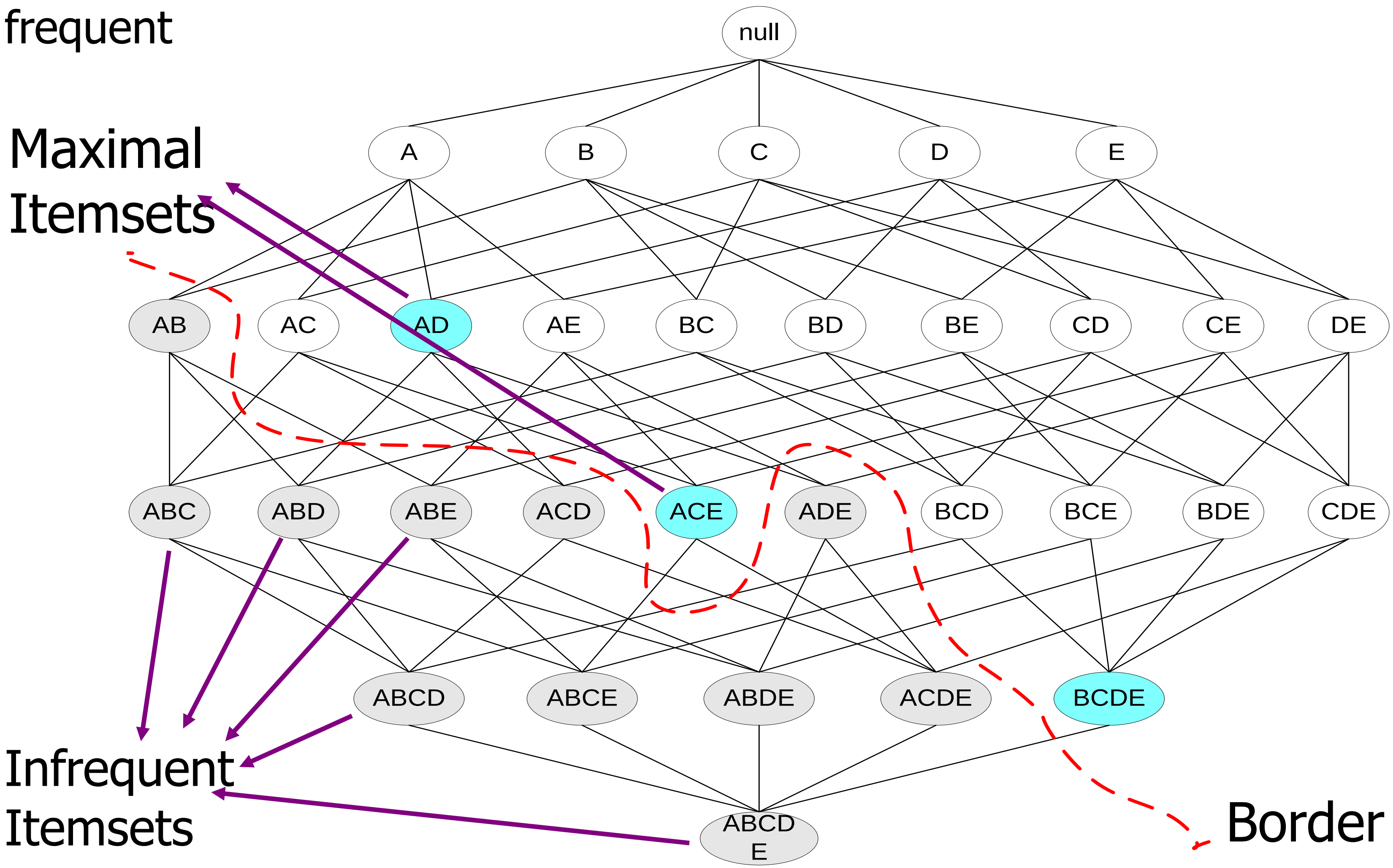
TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0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

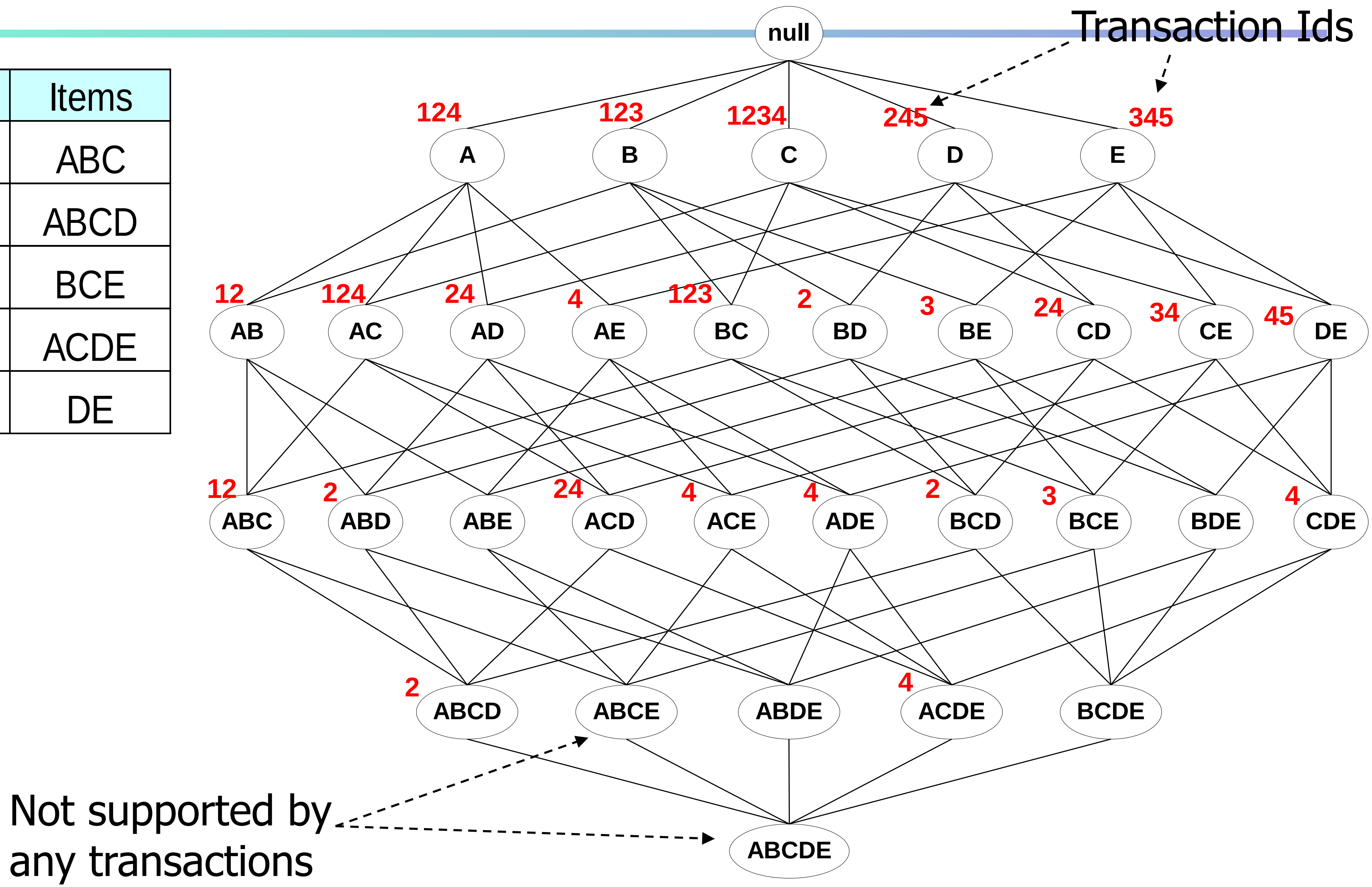
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

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

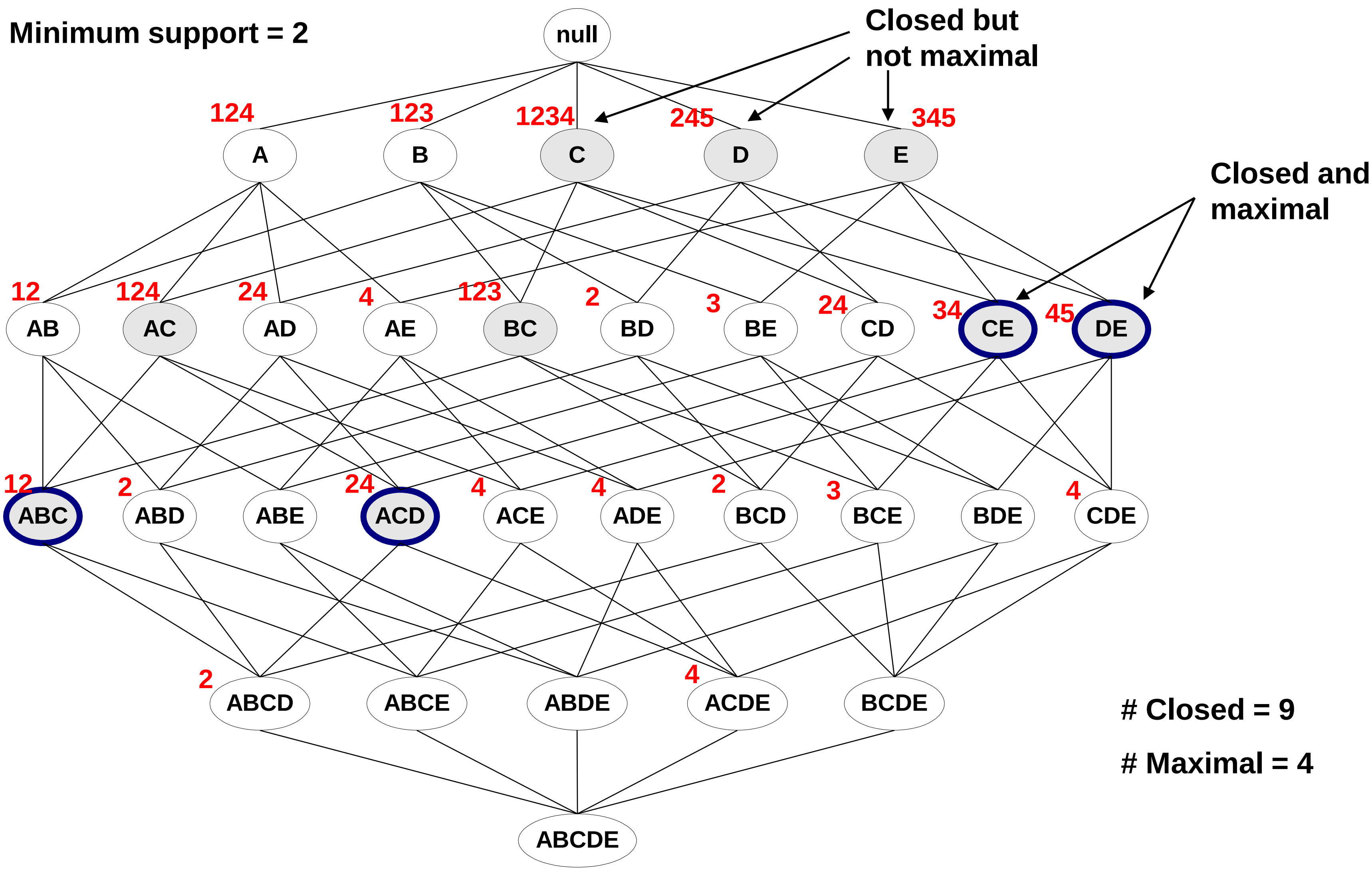
Maximal vs Closed Itemsets

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



Maximal vs Closed Frequent Itemsets

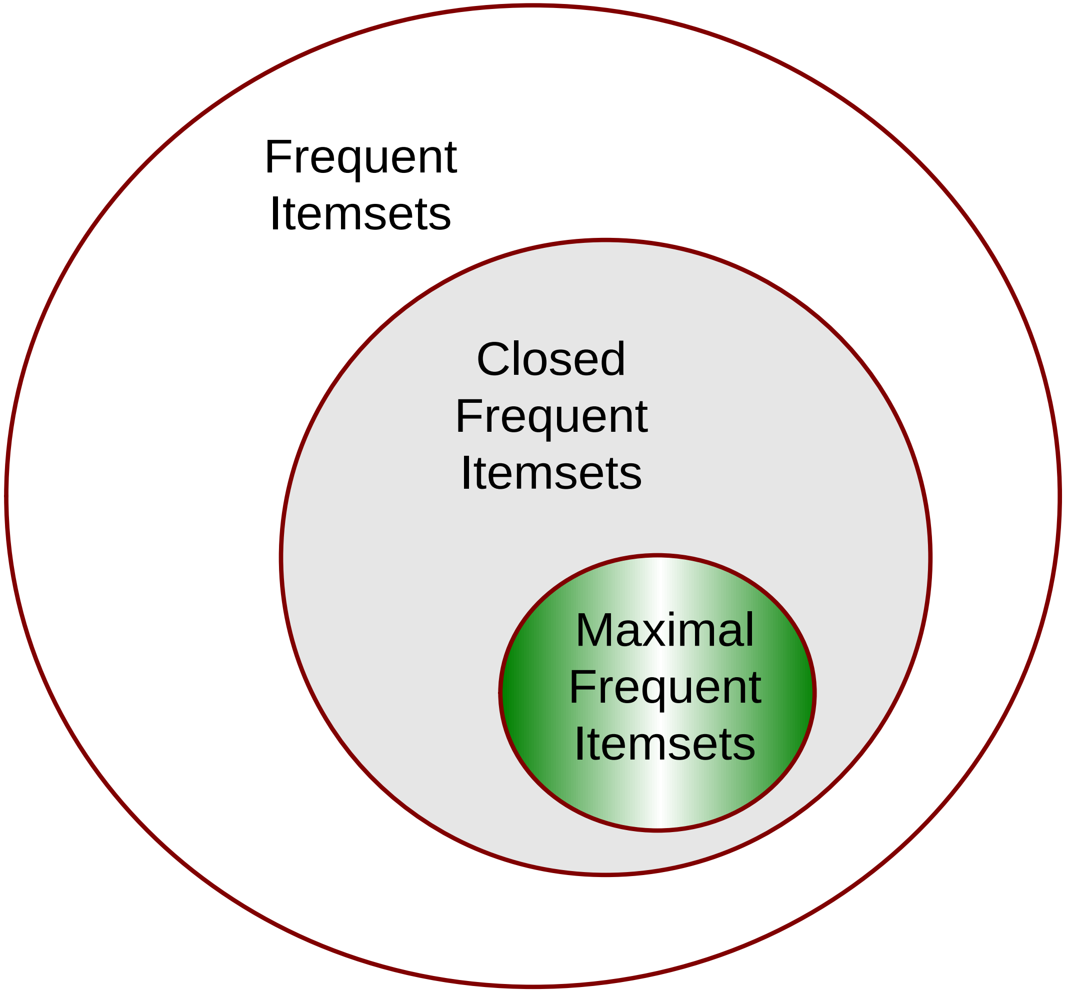
Minimum support = 2



\# Closed = 9

\# Maximal = 4

Maximal vs Closed Itemsets



Large Itemset Mining

- Frequent Itemset Mining

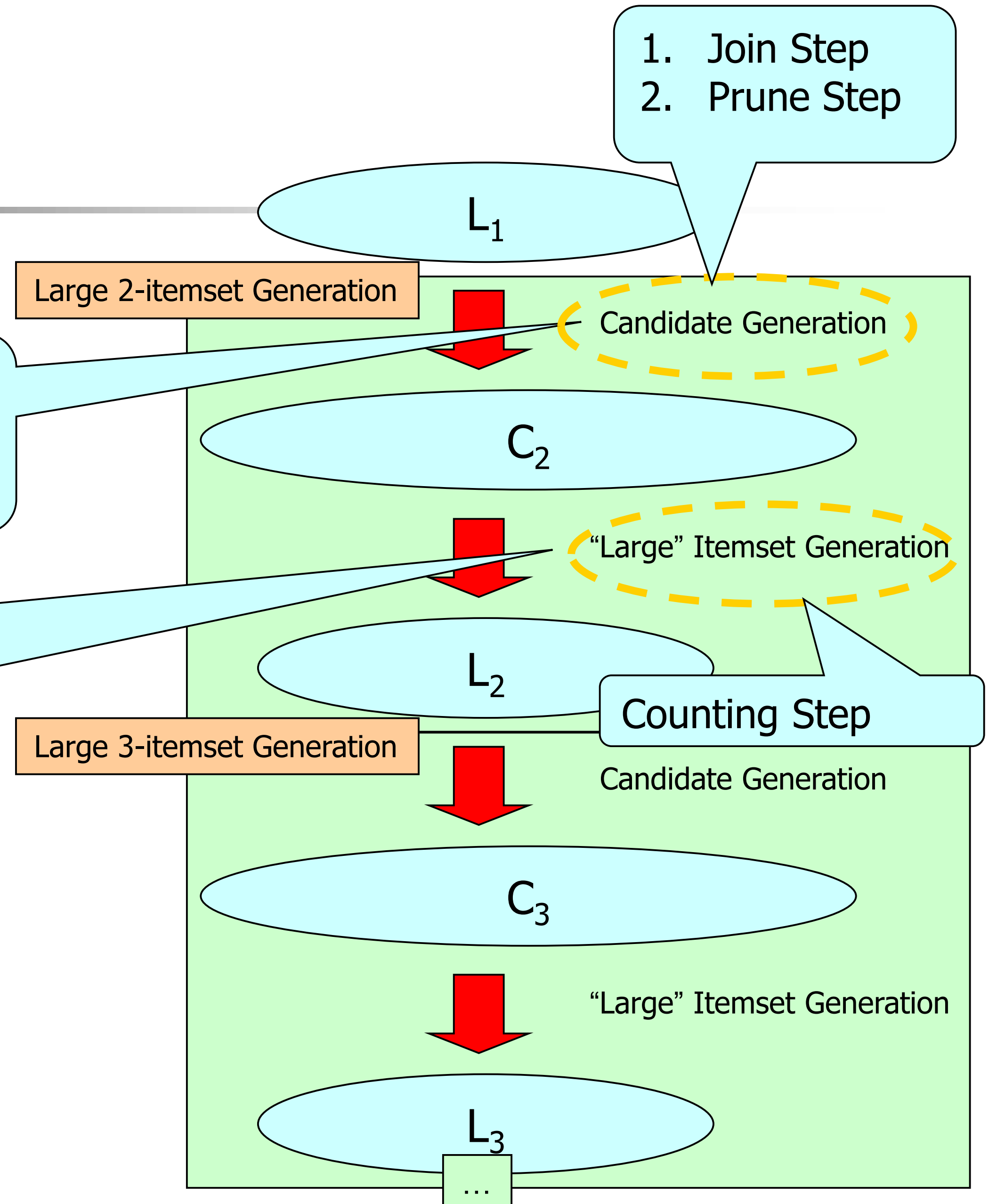
Problem: to find all “large” (or frequent) itemsets
with support at least a threshold
(i.e., itemsets with support >= 3)

TID	Items Bought
100	a, b, c, d, e, f, g, h
200	a, f, g
300	b, d, e, f, j
400	a, b, d, i, k
500	a, b, e, g

Apriori

Disadvantage 1: It is costly to handle a large number of candidate sets

Disadvantage 2: It is tedious to repeatedly scan the database and check the candidate patterns



FP-tree

- Scan the database once to store all essential information in a data structure called FP-tree (Frequent Pattern Tree)
- The FP-tree is concise and is used in directly generating large itemsets

FP-tree

Step 1: Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
Step 2: Construct the FP-tree from the above data
Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
Step 4: Determine the frequent patterns.

FP-tree

- Frequent Itemset Mining

Problem: to find all “large” (or frequent) itemsets
with support at least a threshold
(i.e., itemsets with support >= 3)

TID	Items Bought
100	a, b, c, d, e, f, g, h
200	a, f, g
300	b, d, e, f, j
400	a, b, d, i, k
500	a, b, e, g

FP-tree

TID	Items Bought
100	a, b, c, d, e, f, g, h
200	a, f, g
300	b, d, e, f, j
400	a, b, d, i, k
500	a, b, e, g

TID	Items Bought
100	a, b, c, d, e, f, g, h
200	a, f, g
300	b, d, e, f, j
400	a, b, d, i, k
500	a, b, e, g

TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	
200	a, f, g	
300	b, d, e, f, j	
400	a, b, d, i, k	
500	a, b, e, g	

Threshold = 3

Item	Frequency
a	4
b	
c	
d	
e	
f	
g	
h	
i	
j	
k	

Threshold = 3

TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	
200	a, f, g	
300	b, d, e, f, j	
400	a, b, d, i, k	
500	a, b, e, g	

Item	Frequency
a	4
b	4
c	1
d	3
e	3
f	3
g	3
h	1
i	1
j	1
k	1

TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	
200	a, f, g	
300	b, d, e, f, j	
400	a, b, d, i, k	
500	a, b, e, g	

Threshold = 3

Item	Frequency
a	4
b	4
c	1
d	3
e	3
f	3
g	3
h	1
i	1
j	1
k	1

Item	Frequency
a	4
b	4
d	3
e	3
f	3
g	3

Threshold = 3

TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Item	Frequency
a	4
b	4
c	1
d	3
e	3
f	3
g	3
h	1
i	1
j	1
k	1

Item	Frequency
a	4
b	4
d	3
e	3
f	3
g	3

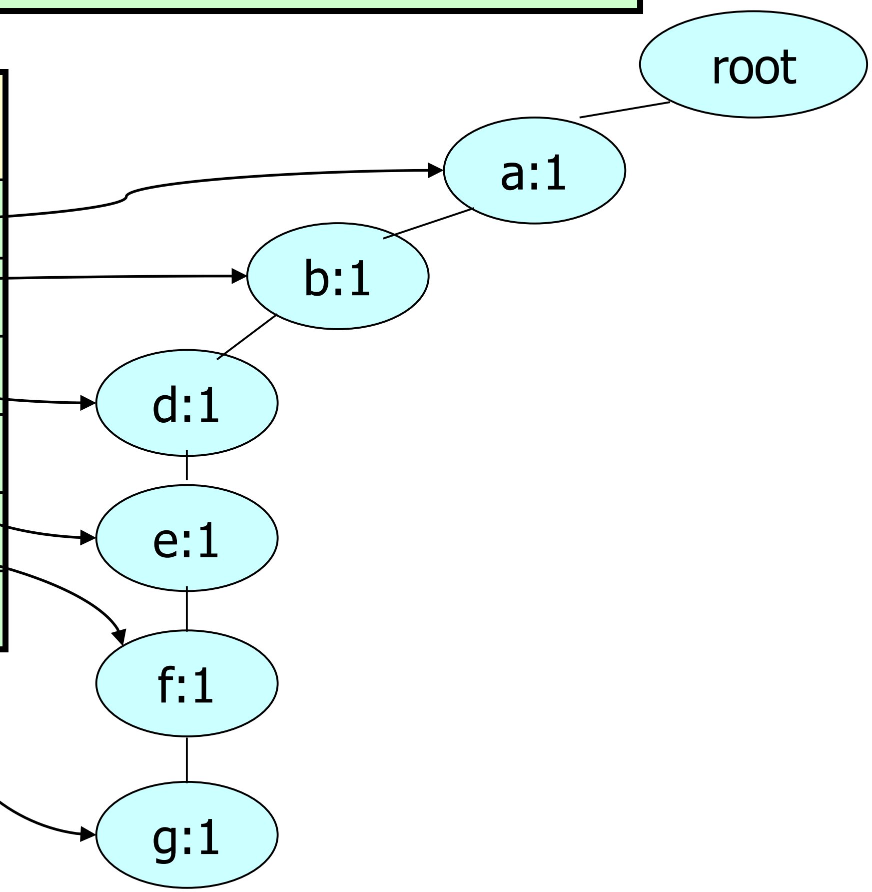
FP-tree

Step 1: Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
Step 2: Construct the FP-tree from the above data
Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
Step 4: Determine the frequent patterns.

TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

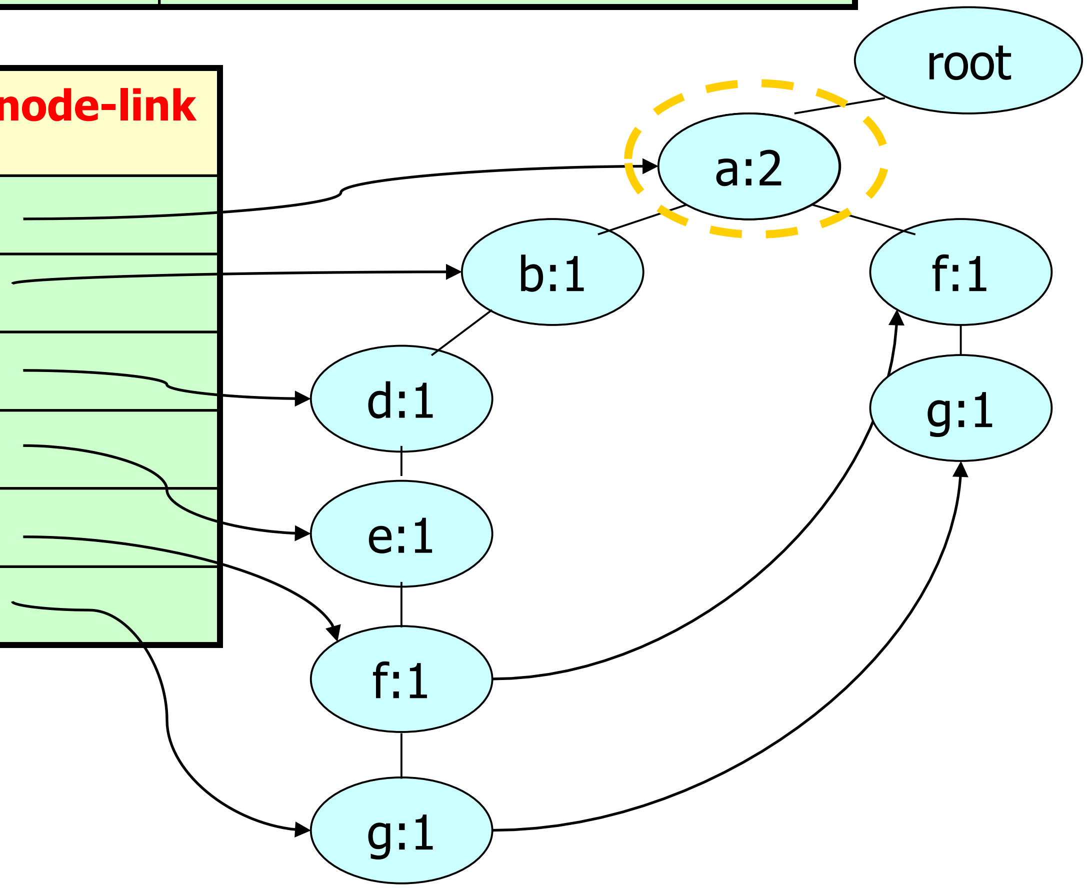
Item	Head of node-link
a	
b	
d	
e	
f	
g	



TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

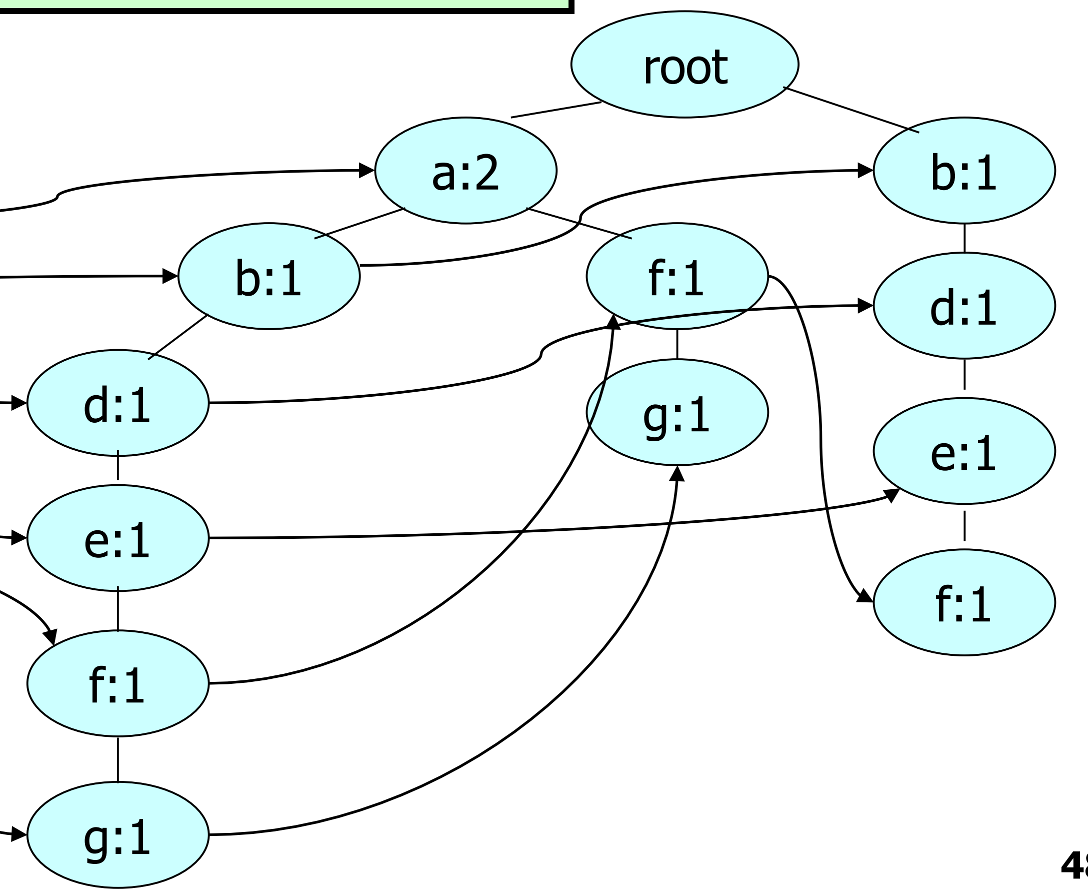
Item	Head of node-link
a	
b	
d	
e	
f	
g	



TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

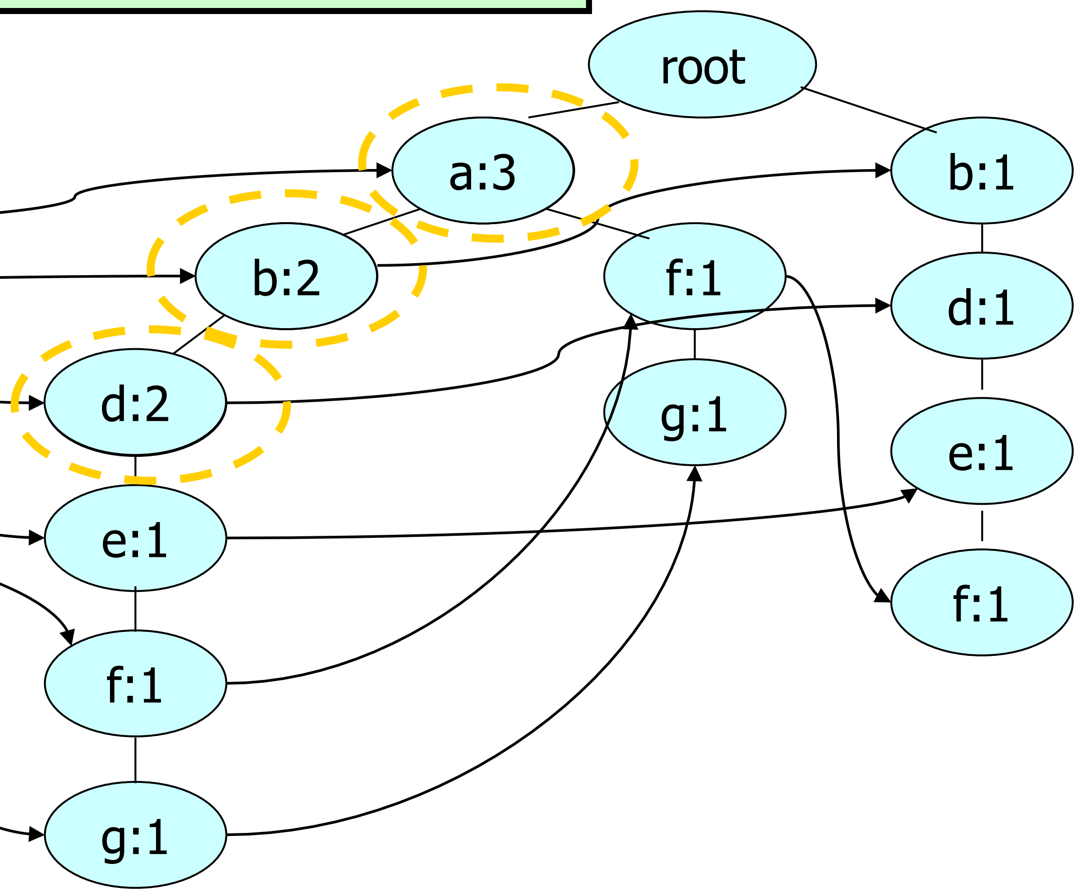
Item	Head of node-link
a	
b	
d	
e	
f	
g	



TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

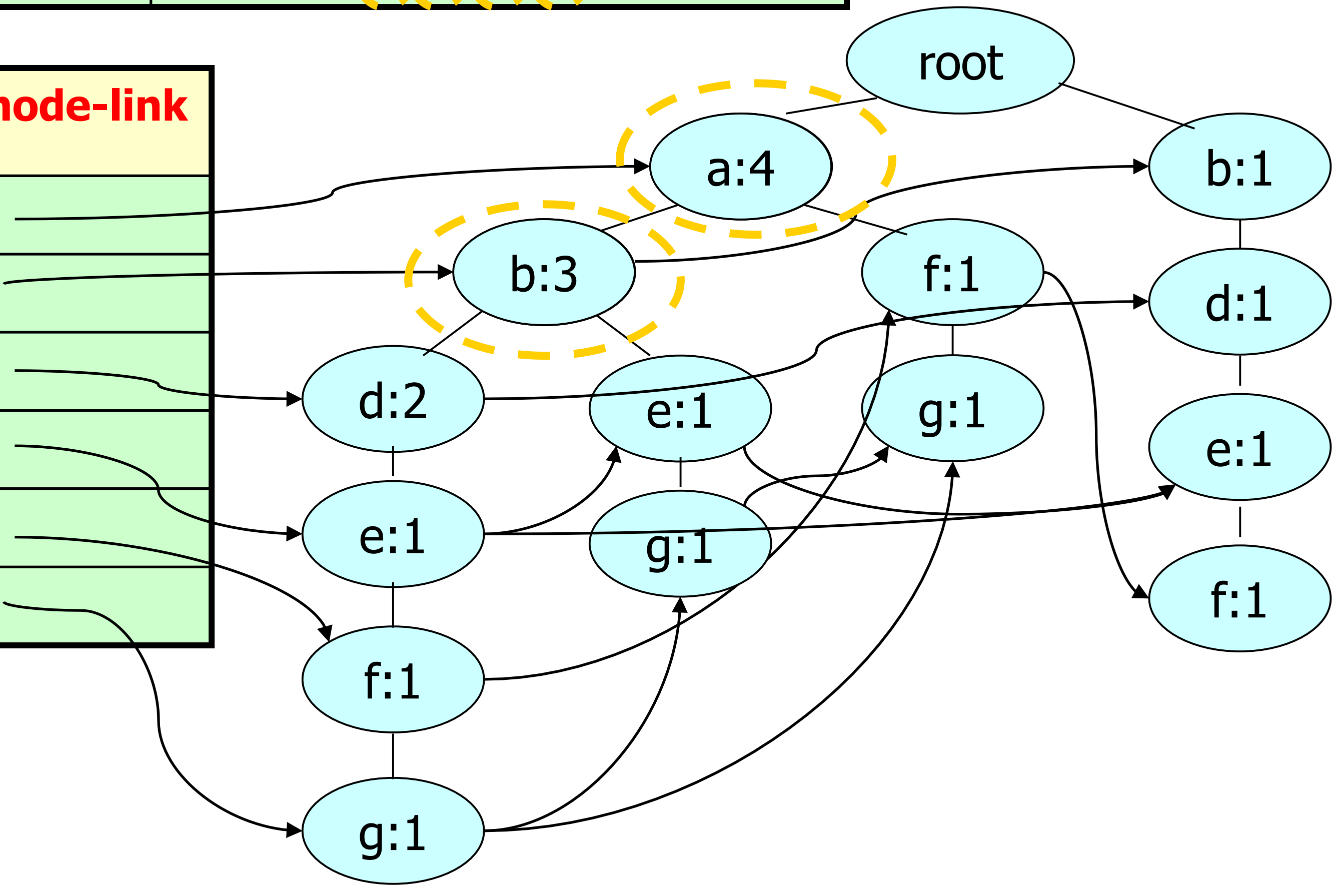
Item	Head of node-link
a	
b	
d	
e	
f	
g	



TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

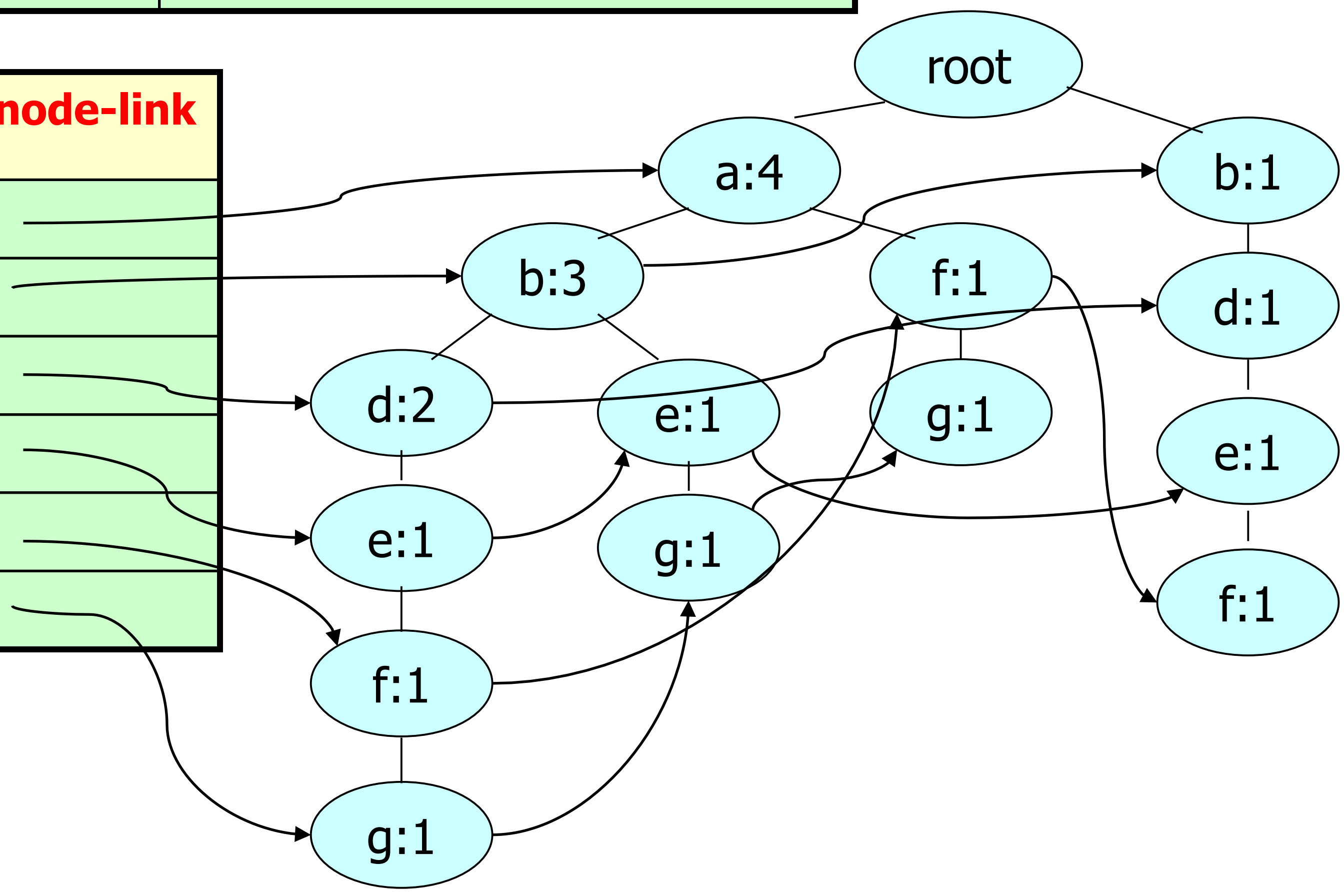
Item	Head of node-link
a	
b	
d	
e	
f	
g	



TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	



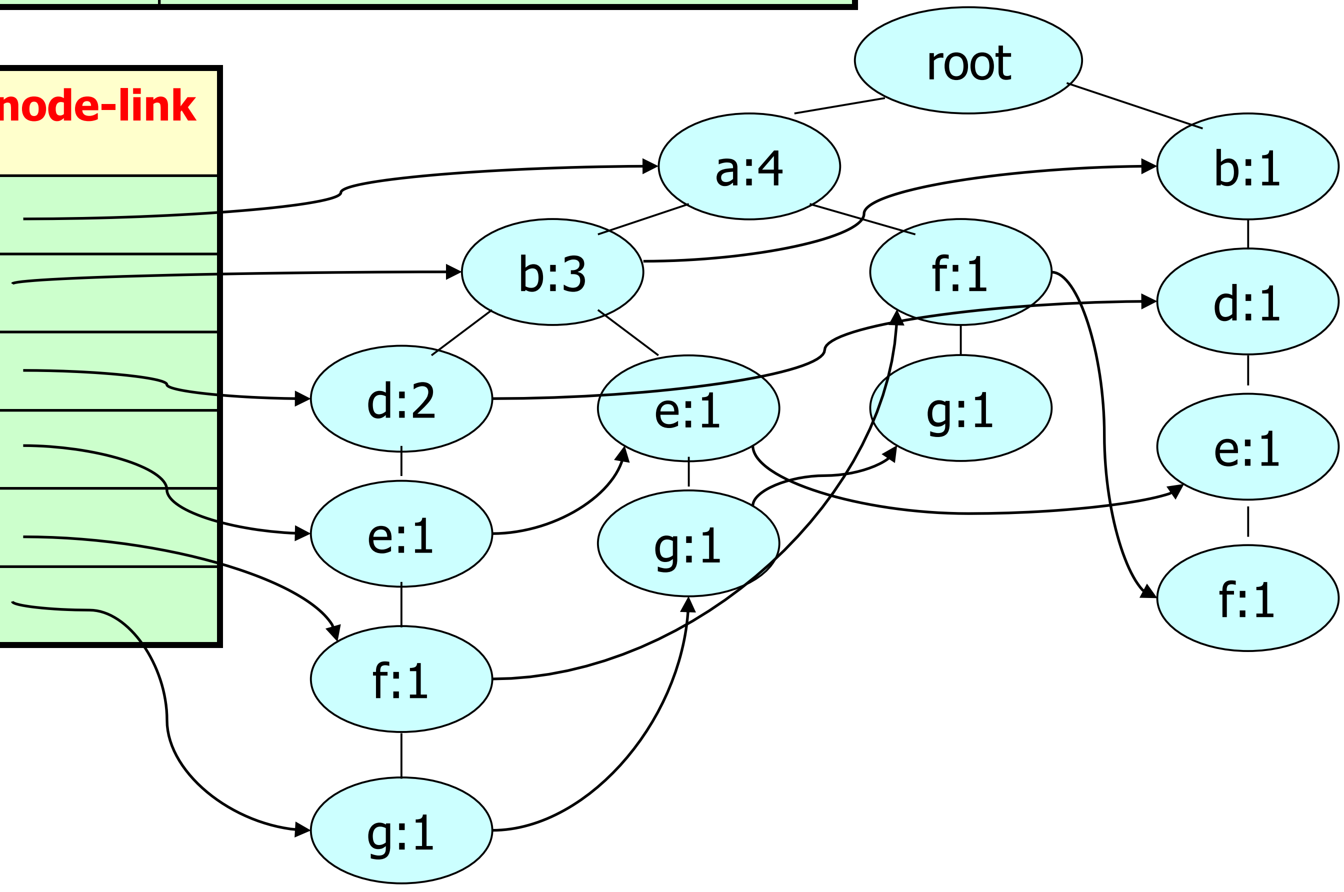
FP-tree

Step 1: Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
Step 2: Construct the FP-tree from the above data
Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
Step 4: Determine the frequent patterns.

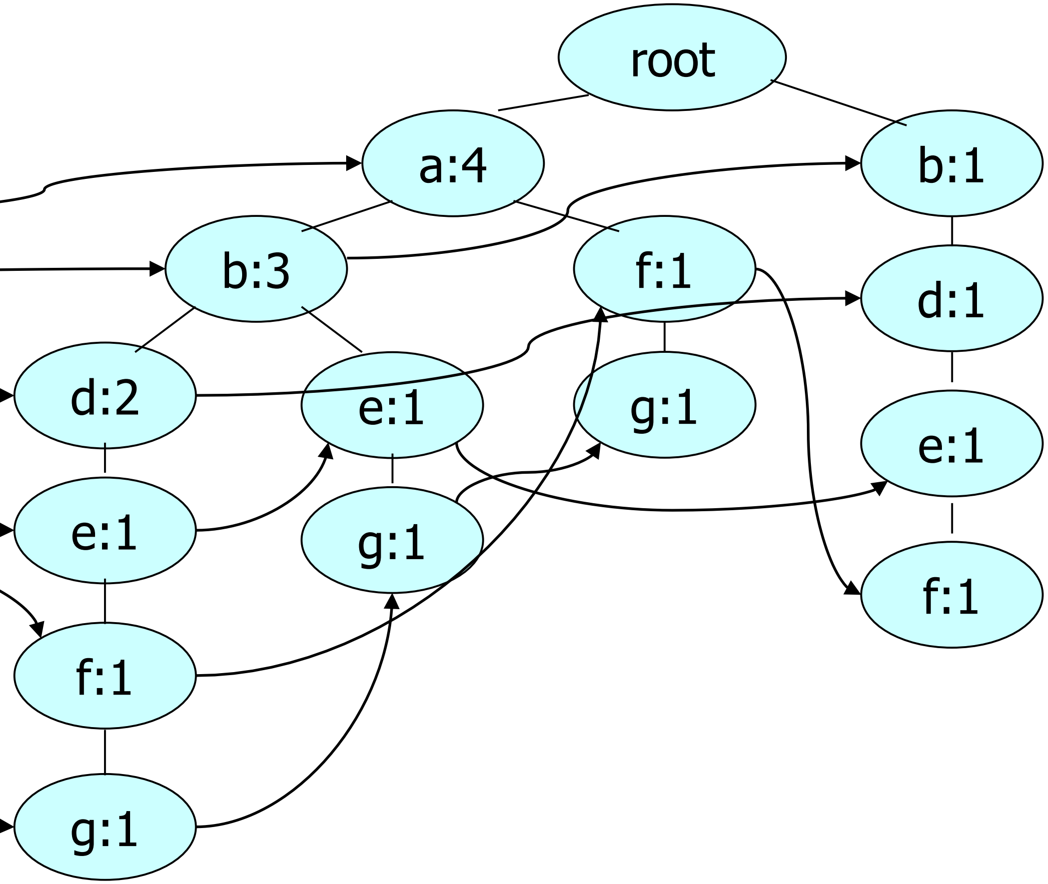
TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	

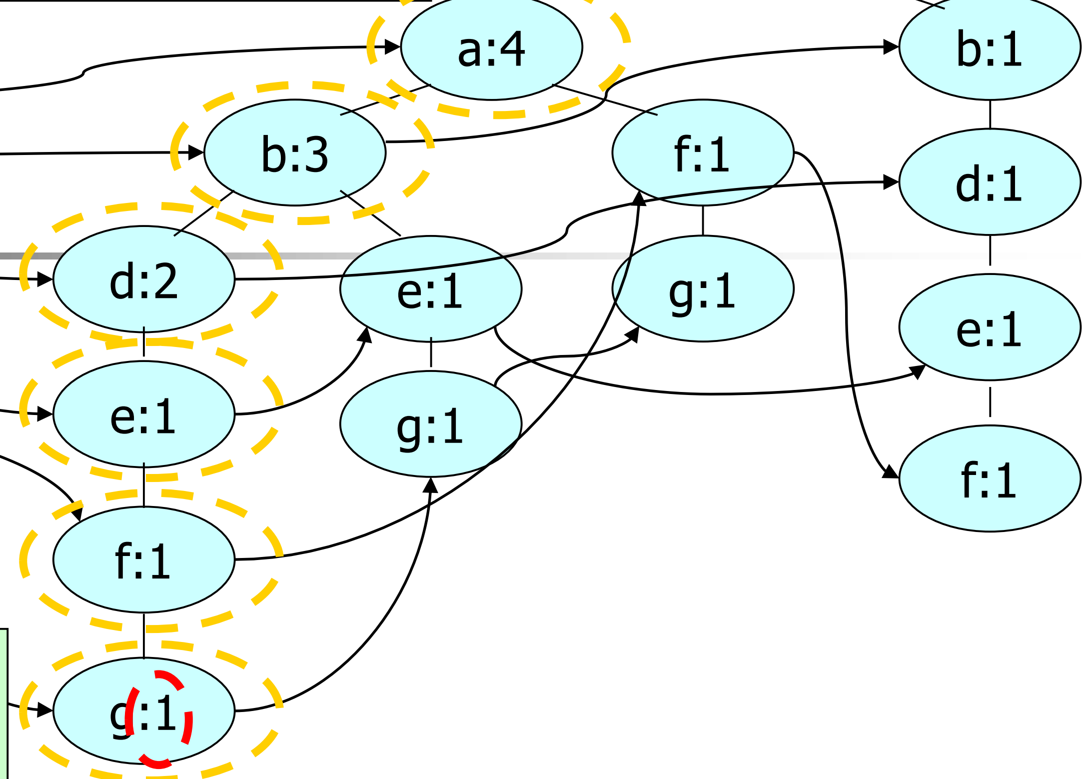


Item	Head of node-link
a	
b	
d	
e	
f	
g	



Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	



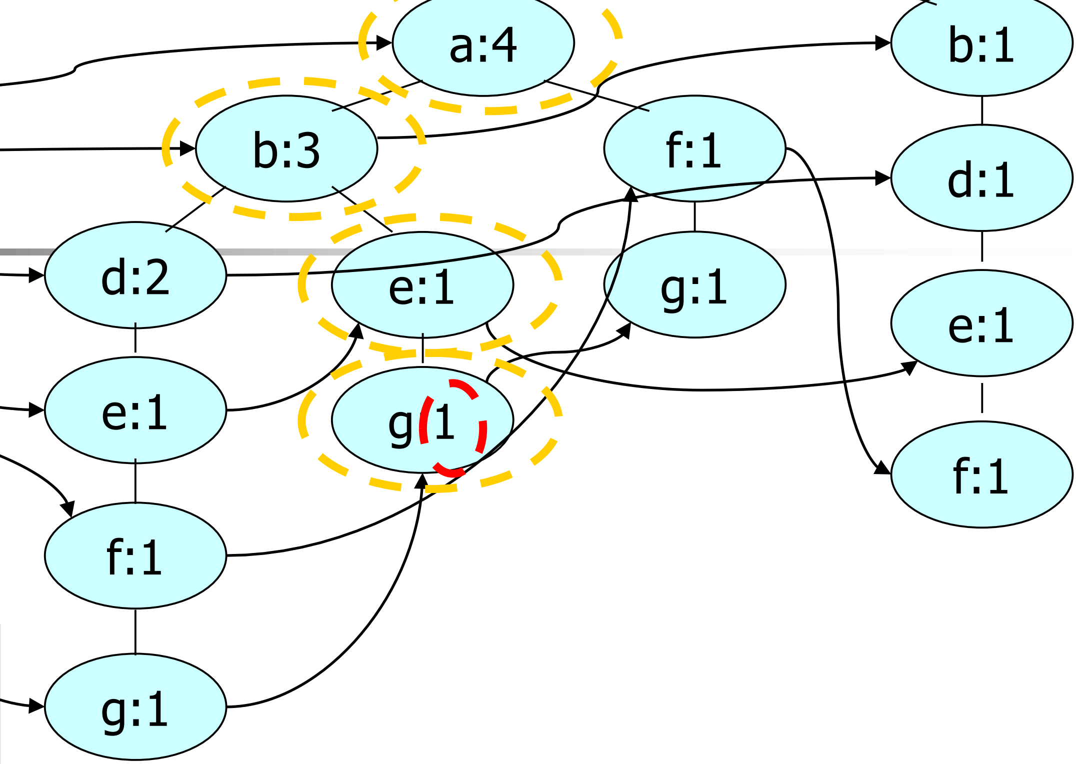
Cond. FP-tree on "g"

{ (a:1, b:1, d:1, e:1, f:1, g:1),

}

Item	Head of node-link
a	
b	
d	
e	
f	
g	

Threshold = 3

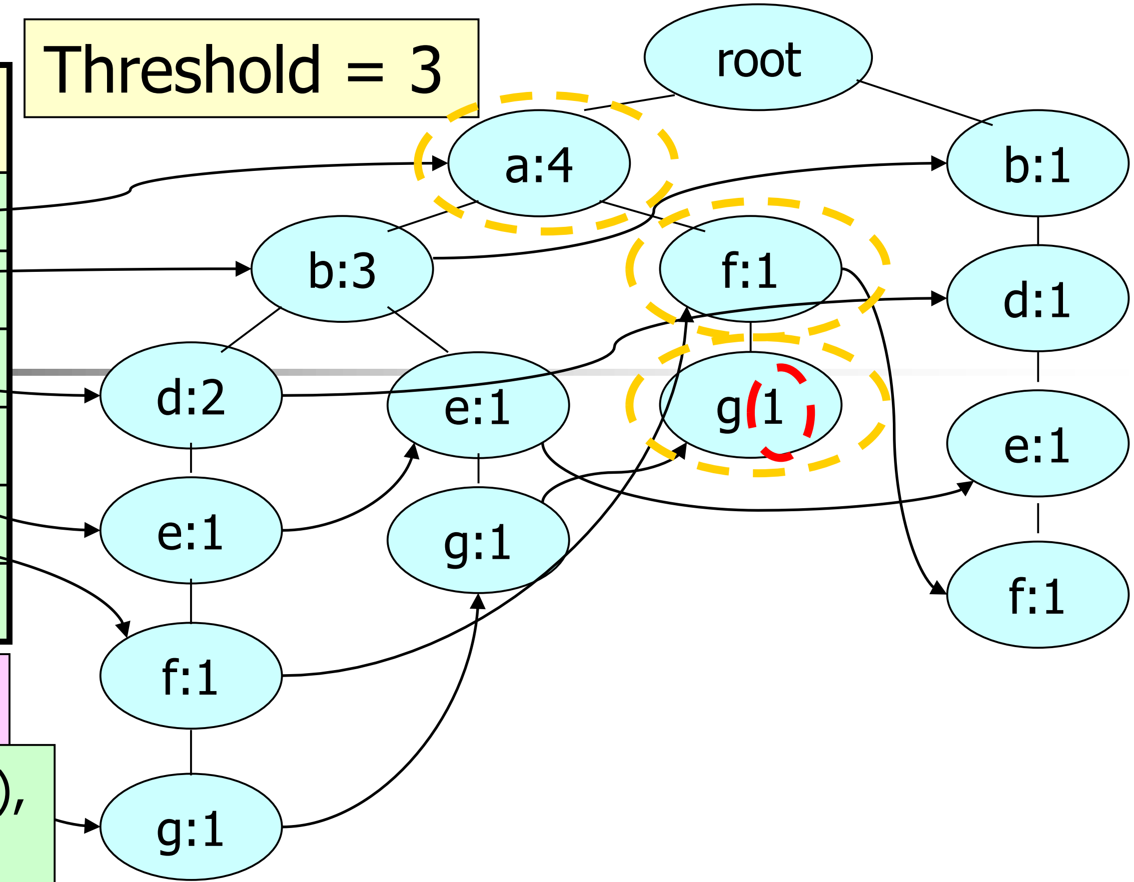


Cond. FP-tree on "g"

{ (a:1, b:1, d:1, e:1, f:1, g:1),
(a:1, b:1, e:1, g:1),
}

Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	



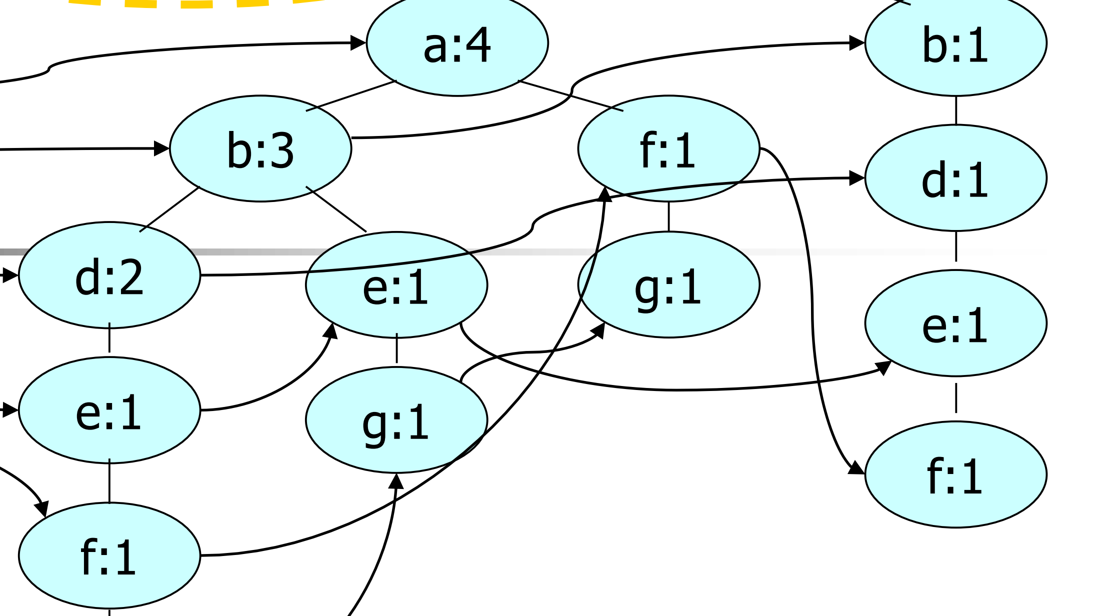
Cond. FP-tree on “g”

{ (a:1, b:1, d:1, e:1, f:1, g:1),
(a:1, b:1, e:1, g:1),
(a:1, f:1, g:1)}

Item	Frequency
a	3
b	2
d	1
e	2
f	2
g	3

Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	



Cond. FP-tree on "g"	3

{(a:1, b:1, d:1, e:1, f:1, g:1),
(a:1, b:1, e:1, g:1),
(a:1, f:1, g:1)}

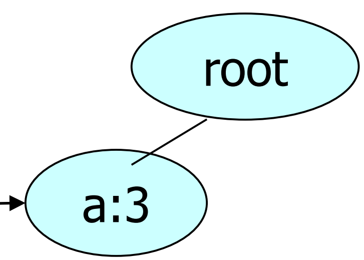
{(a:1, g:1),
(a:1, g:1),
(a:1, g:1) }

conditional pattern base of "g"

Item	Frequency
a	3
b	2
d	1
e	2
f	2
g	3

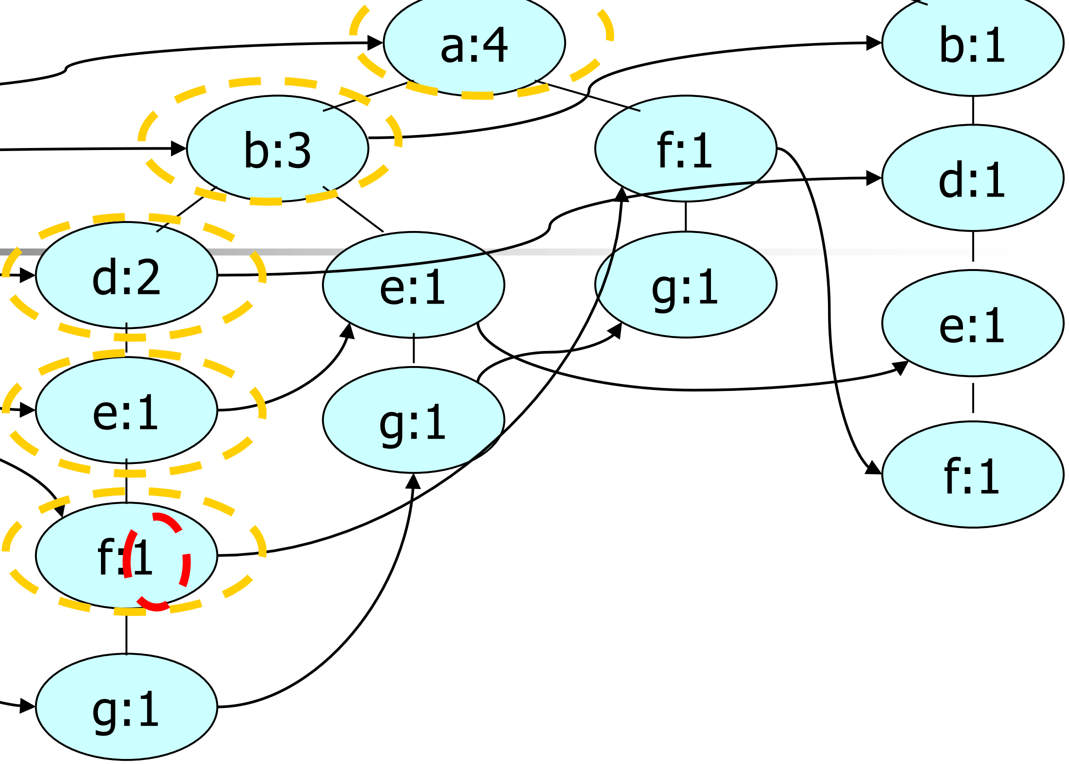
Item	Frequency
a	3
g	3

Item	Head of node-link
a	



Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	



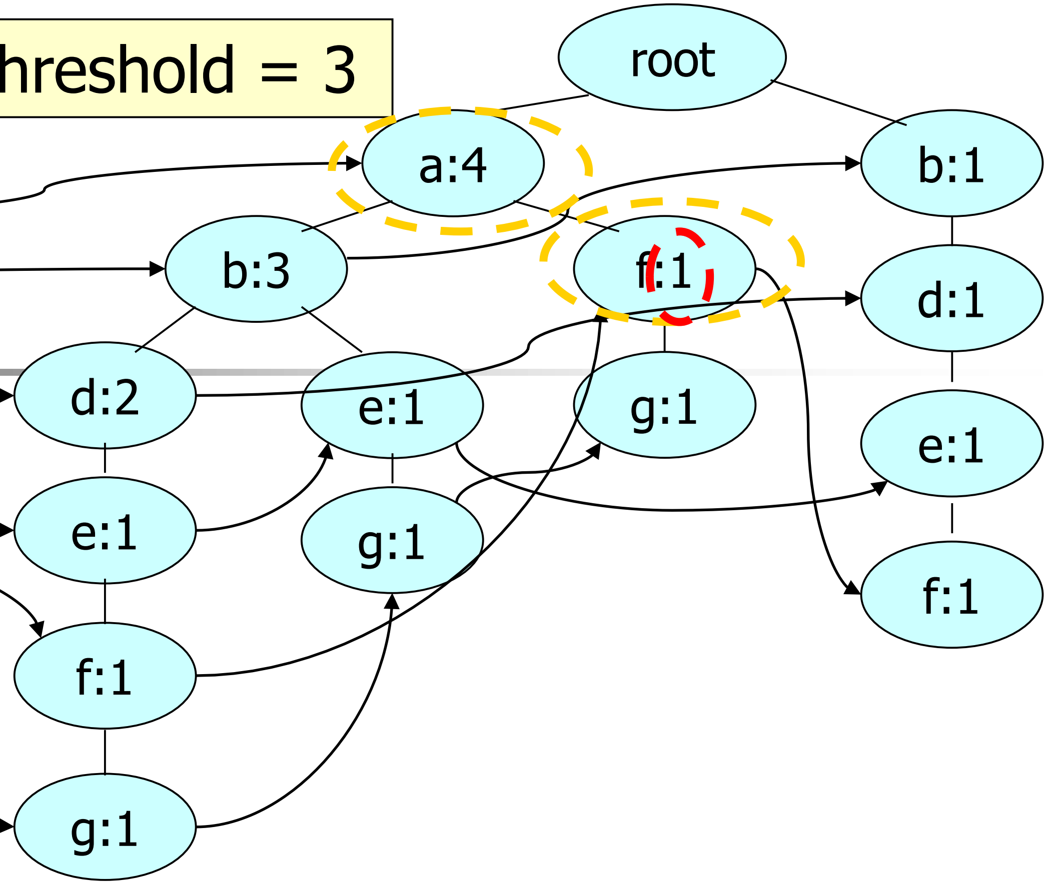
Cond. FP-tree on “f”

{ (a:1, b:1, d:1, e:1, f:1),

}

Threshold = 3

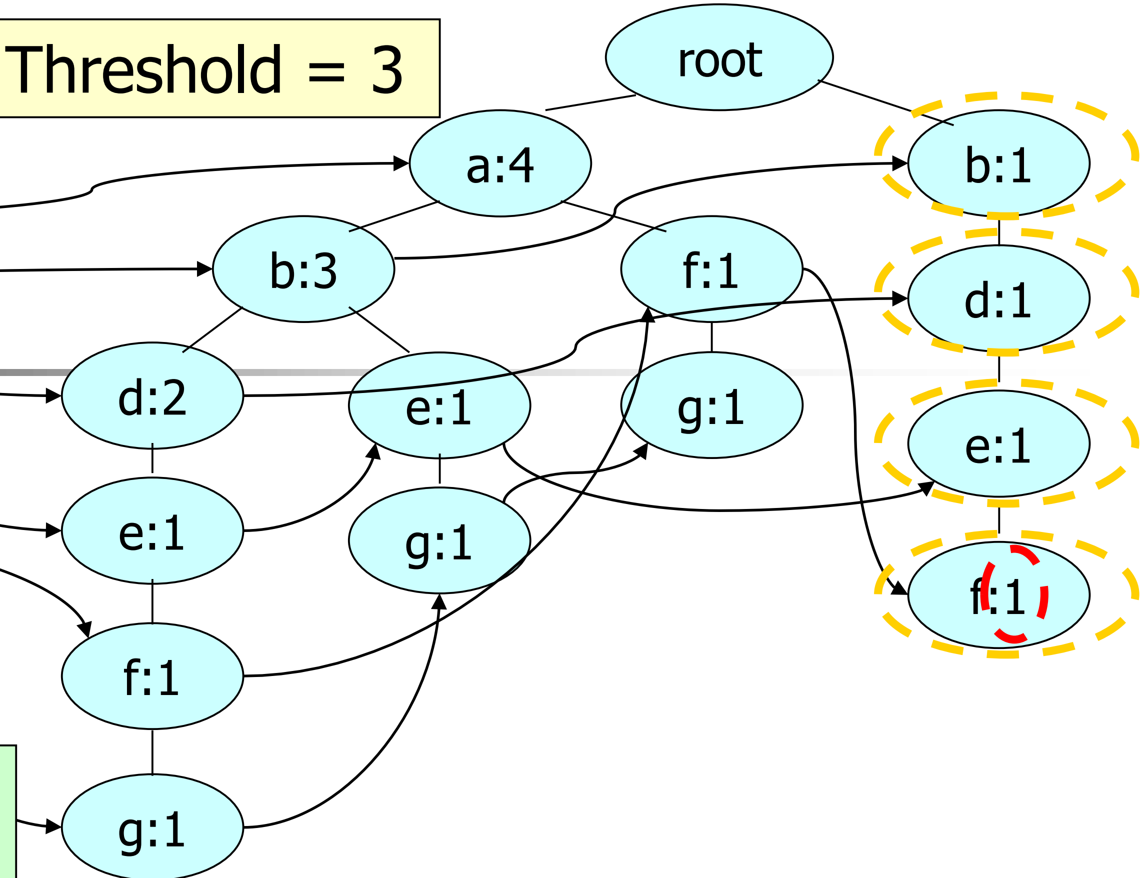
Item	Head of node-link
a	
b	
d	
e	
f	
g	



Cond. FP-tree on "f"

{ (a:1, b:1, d:1, e:1, f:1),
(a:1, f:1),
}

Item	Head of node-link
a	
b	
d	
e	
f	
g	



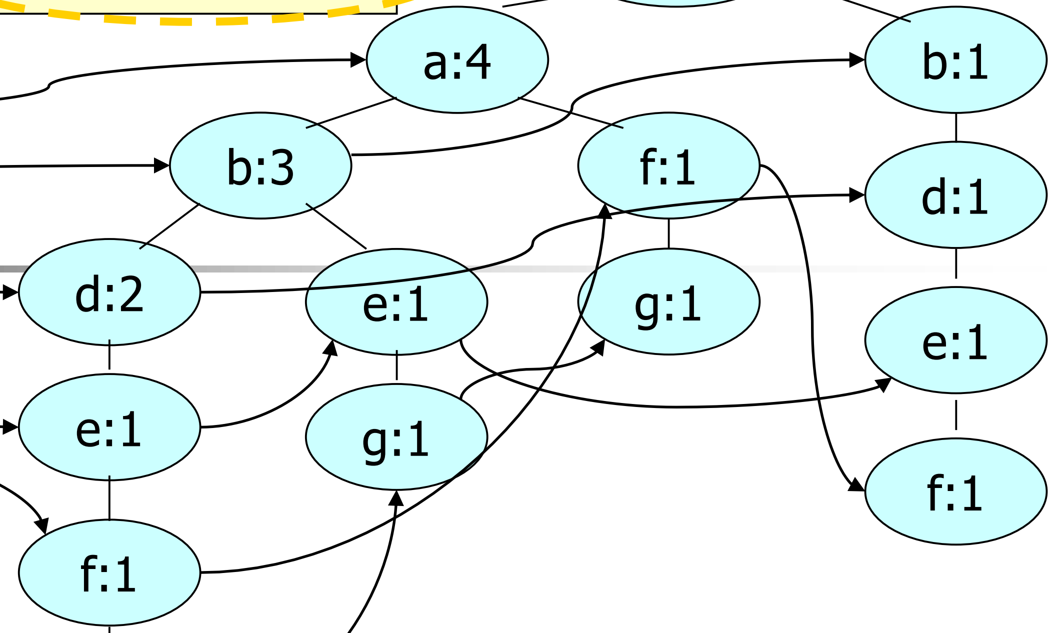
Cond. FP-tree on "f"

{ (a:1, b:1, d:1, e:1, f:1),
(a:1, f:1),
(b:1, d:1, e:1, f:1)}

Item	Frequency
a	2
b	2
d	2
e	2
f	3
g	0

Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	



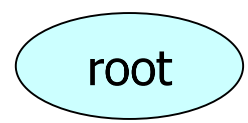
Cond. FP-tree on “f”	3

{ (a:1, b:1, d:1, e:1, f:1),
(a:1, f:1),
(b:1, d:1, e:1, f:1)}

{ (f:1),
(f:1),
(f:1) }

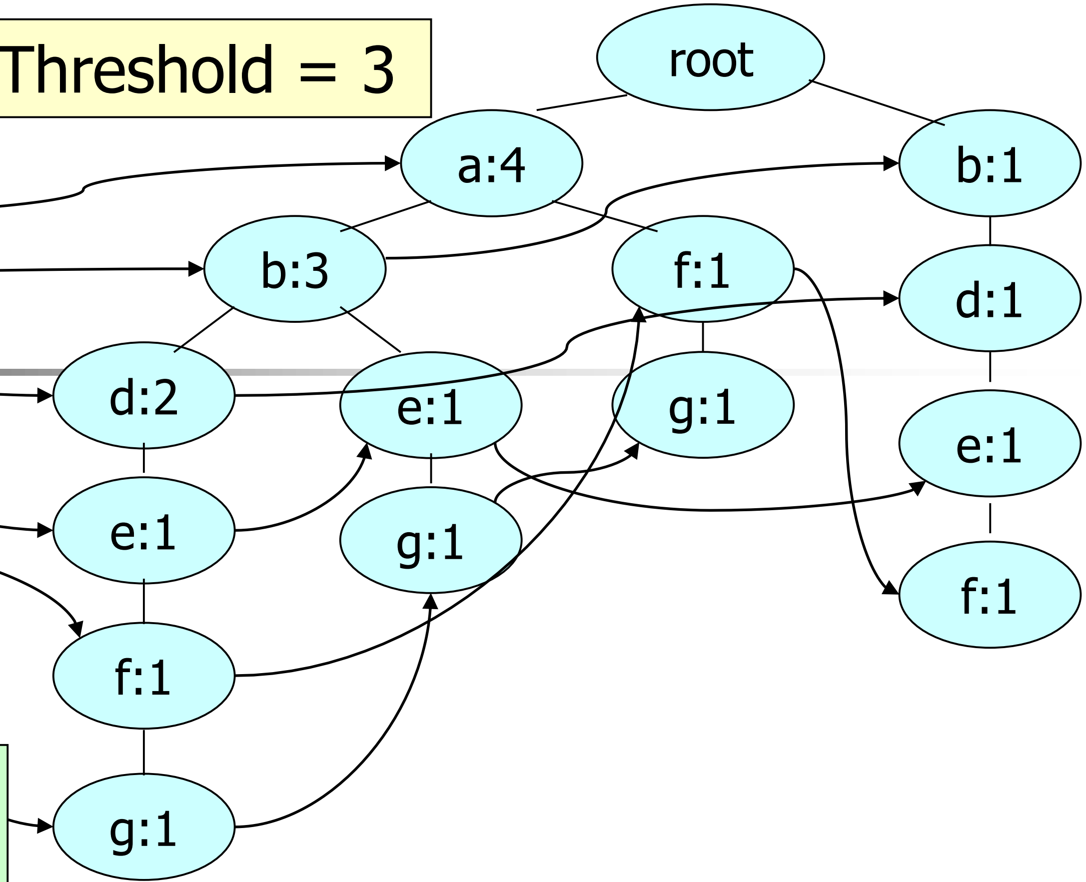
Item	Frequency
a	2
b	2
d	2
e	2
f	3
g	0

Item	Frequency
f	3



Item	Head of node-link
a	
b	
d	
e	
f	
g	

Threshold = 3



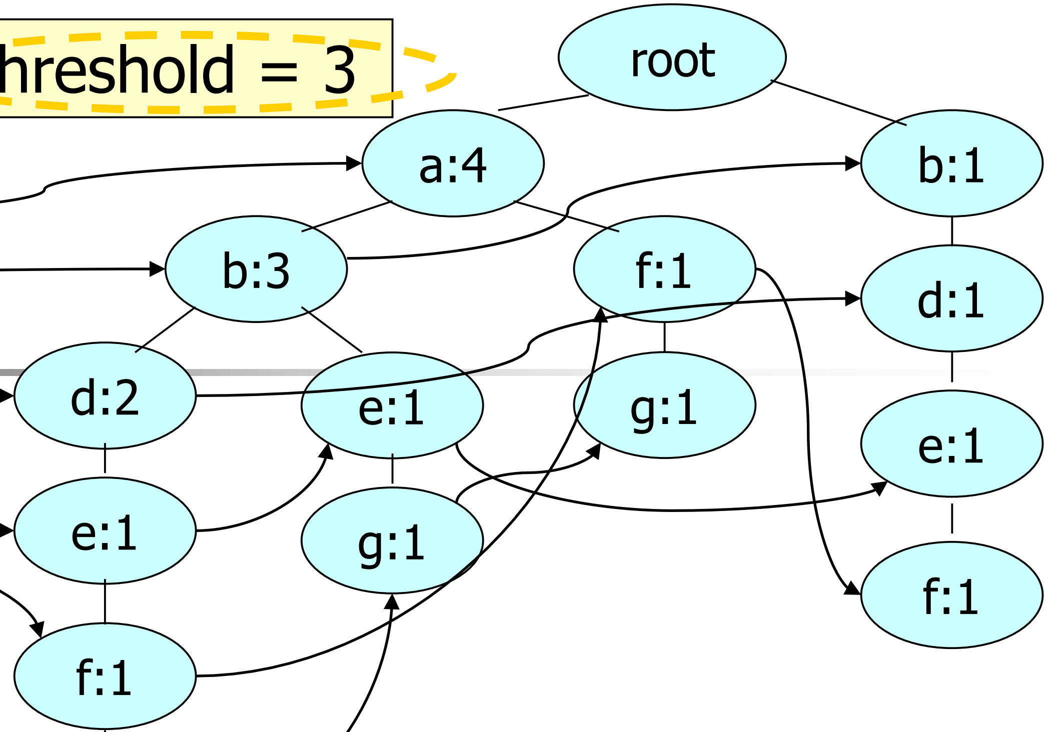
Cond. FP-tree on "e"

{ (a:1, b:1, d:1, e:1),
(a:1, b:1, e:1),
(b:1, d:1, e:1) }

Item	Frequency
a	2
b	3
d	2
e	3
f	0
g	0

Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	



Cond. FP-tree on "e" 3

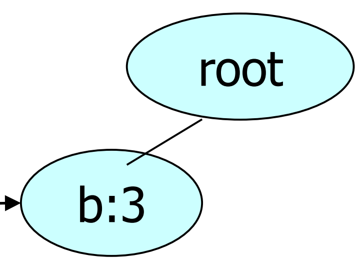
{ (a:1, b:1, d:1, e:1),
(a:1, b:1, e:1),
(b:1, d:1, e:1) }

Item	Frequency
a	2
b	3
d	2
e	3
f	0
g	0

{ (b:1, e:1),
(b:1, e:1),
(b:1, e:1) }

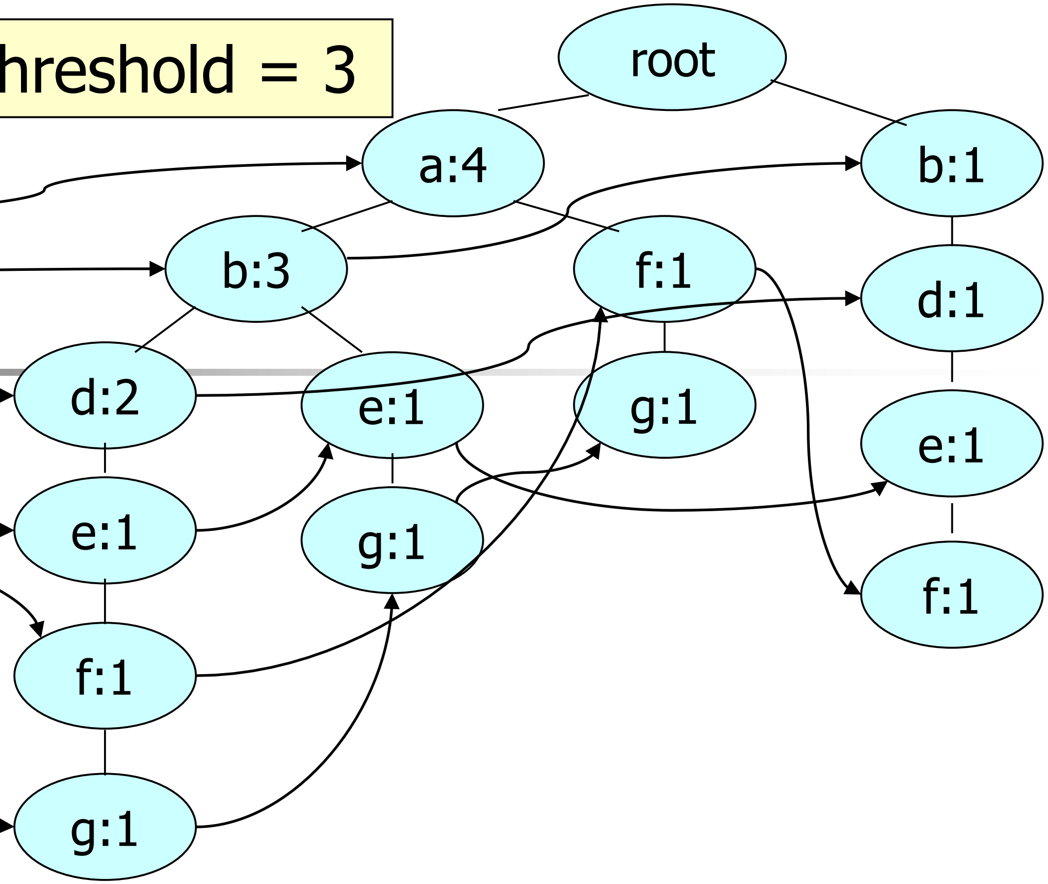
Item	Frequency
b	3
e	3

Item	Head of node-link
b	



Item	Head of node-link
a	
b	
d	
e	
f	
g	

Threshold = 3



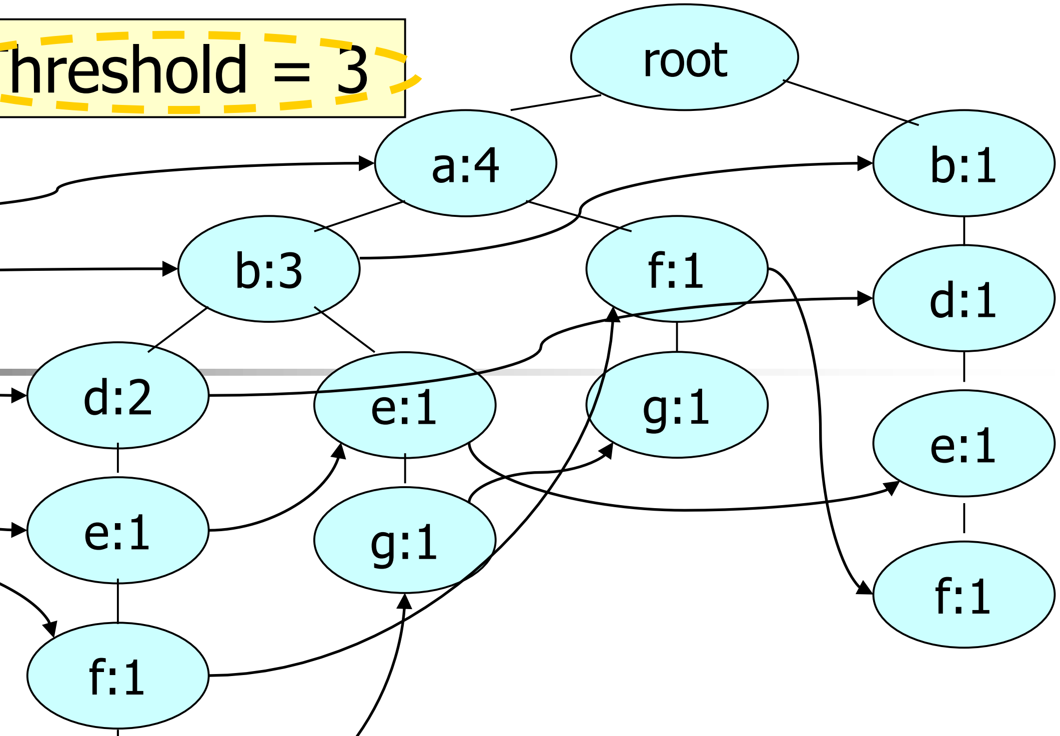
Cond. FP-tree on “d”

{ (a:2, b:2, d:2),
(b:1, d:1) }

Item	Frequency
a	2
b	3
d	3
e	0
f	0
g	0

Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	



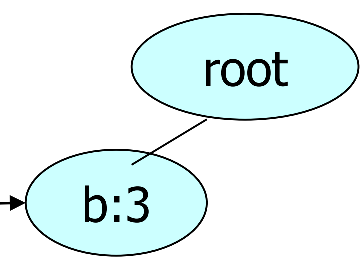
Cond. FP-tree on "d"	3
{ (a:2, b:2, d:2), (b:1, d:1) }	

{ (b:2, d:2), (b:1, d:1) }

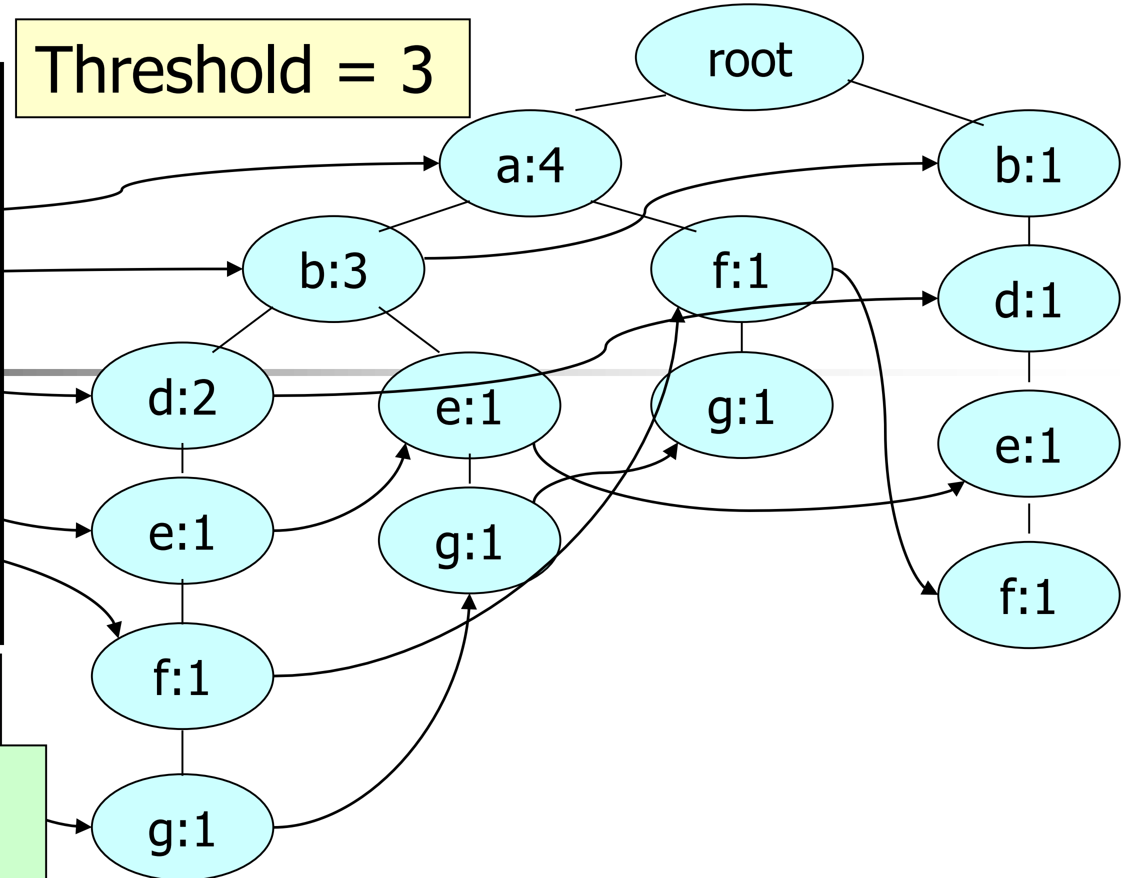
Item	Frequency
a	2
b	3
d	3
e	0
f	0
g	0

Item	Frequency
b	3
d	3

Item	Head of node-link
b	



Item	Head of node-link
a	
b	
d	
e	
f	
g	



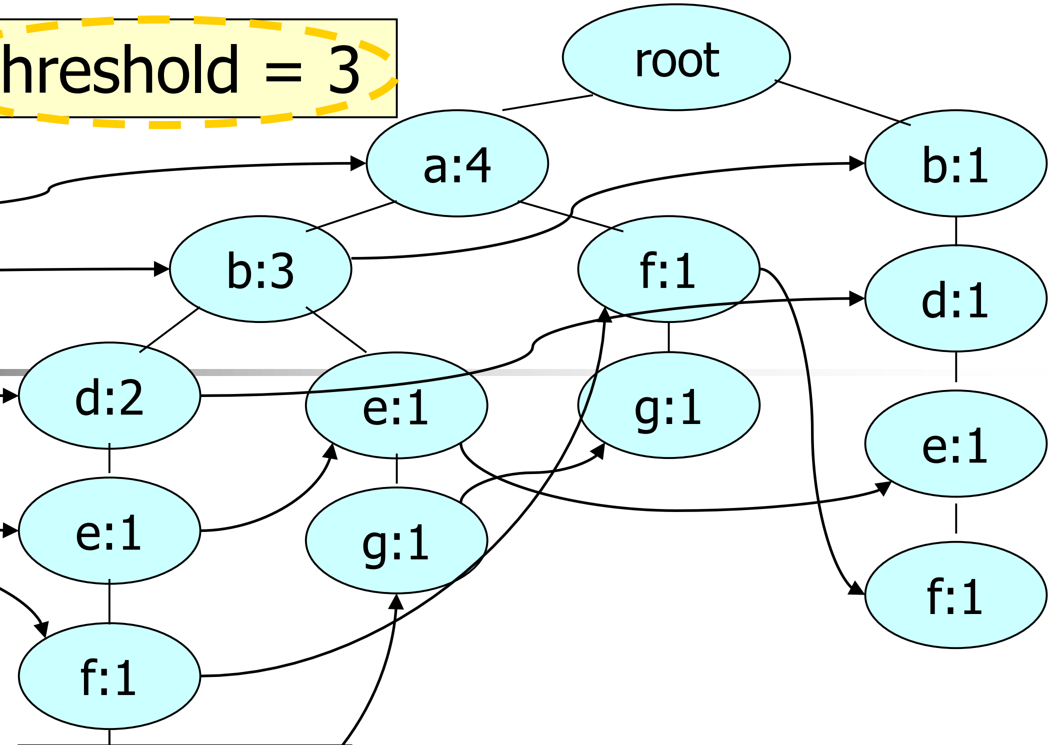
Cond. FP-tree on “b”

{ (a:3, b:3),
(b:1) }

Item	Frequency
a	3
b	4
d	0
e	0
f	0
g	0

Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	



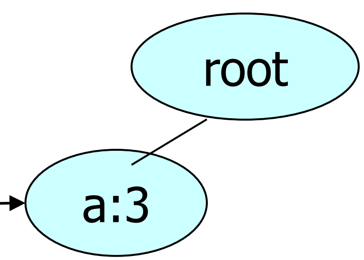
Cond. FP-tree on "b"	4
{ (a:3, b:3), (b:1) }	

{ (a:3, b:3),
(b:1) }

Item	Frequency
a	3
b	4
d	0
e	0
f	0
g	0

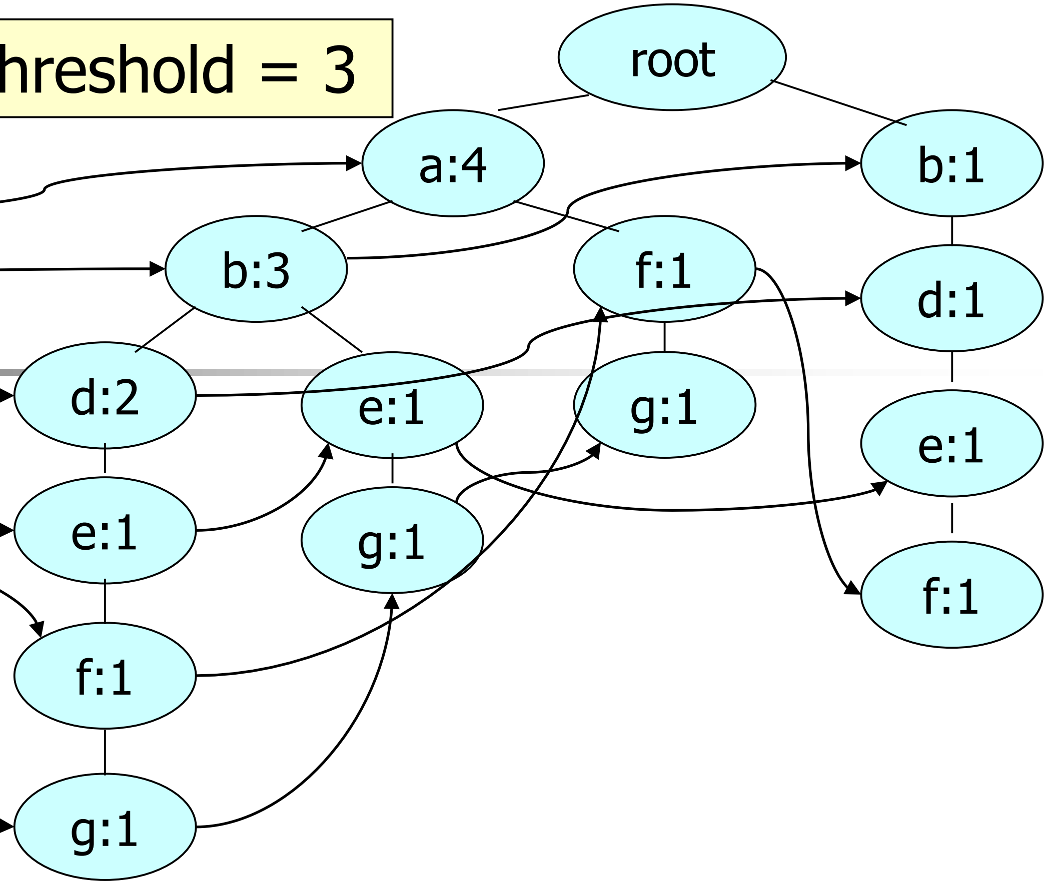
Item	Frequency
a	3
b	4

Item	Head of node-link
a	



Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	



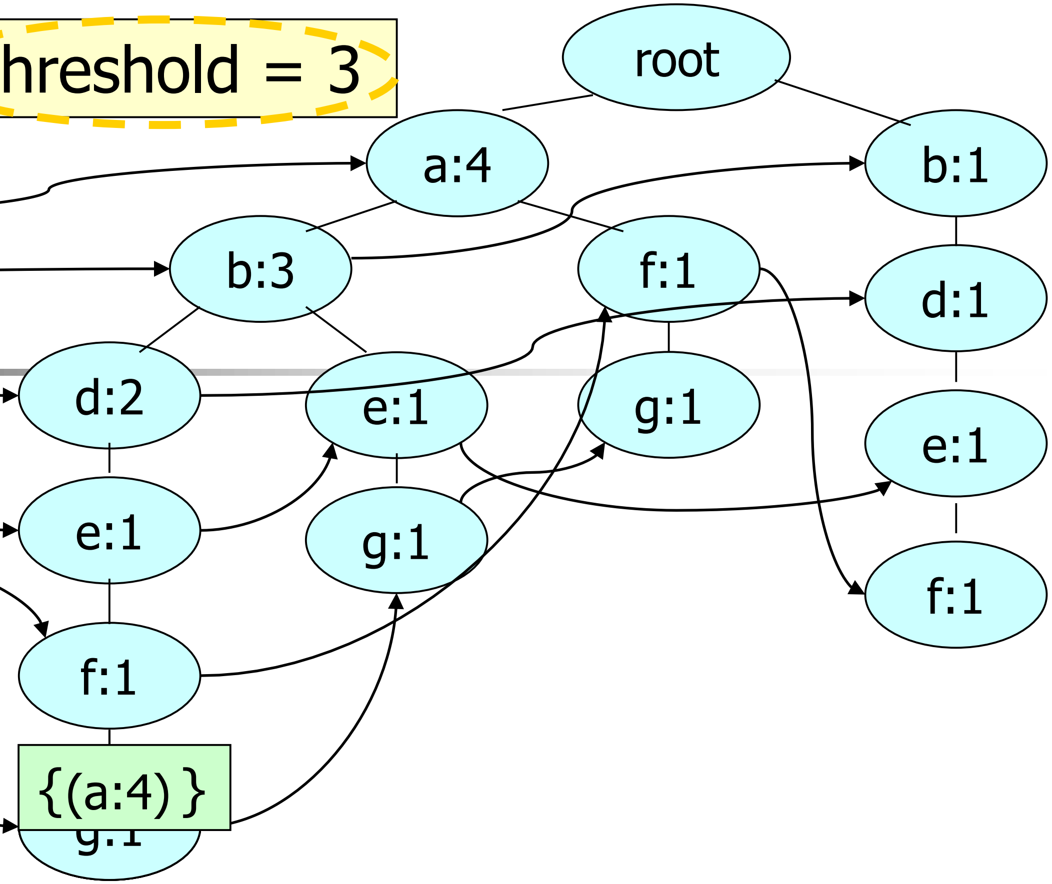
Cond. FP-tree on “a”

{ (a:4) }

Item	Frequency
a	4
b	0
d	0
e	0
f	0
g	0

Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	

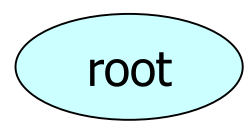


Cond. FP-tree on "a"	4
{ (a:4) }	

{(a:4)}

Item	Frequency
a	4
b	0
d	0
e	0
f	0
g	0

Item	Frequency
a	4



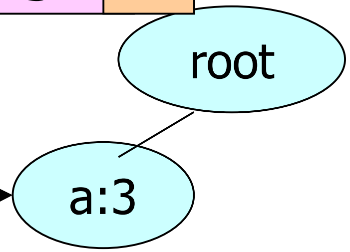
FP-tree

Step 1: Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
Step 2: Construct the FP-tree from the above data
Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
Step 4: Determine the frequent patterns.

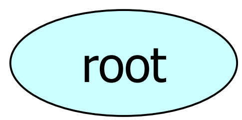
Cond. FP-tree on “g” 3

Cond. FP-tree on “g” 3

Item	Head of node-link
a	



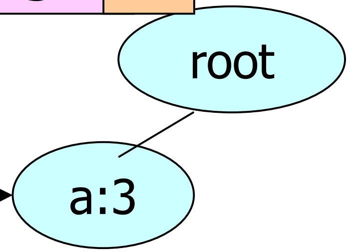
Cond. FP-tree on “f” 3



Cond. FP-tree on “e” 3

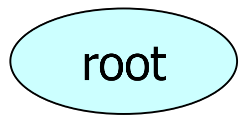
Cond. FP-tree on “g” 3

Item	Head of node-link
a	



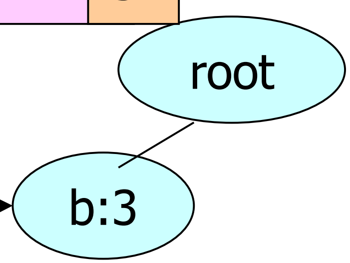
Cond. FP-tree on “d” 3

Cond. FP-tree on “f” 3



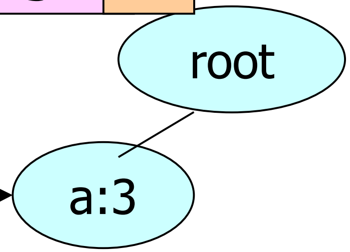
Cond. FP-tree on “e” 3

Item	Head of node-link
b	



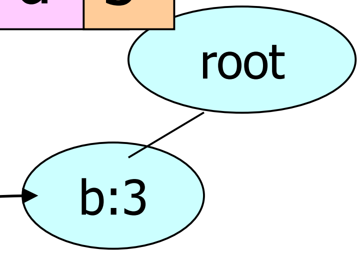
Cond. FP-tree on “g” 3

Item	Head of node-link
a	

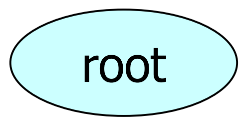


Cond. FP-tree on “d” 3

Item	Head of node-link
b	



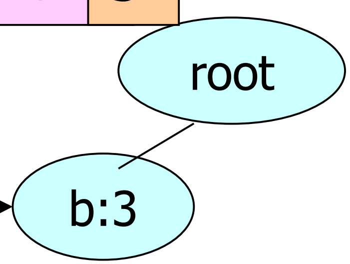
Cond. FP-tree on “f” 3



Cond. FP-tree on “b” 4

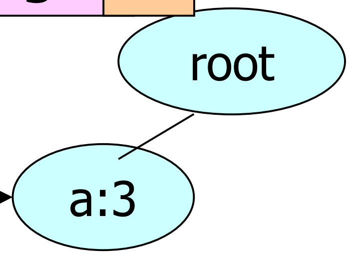
Cond. FP-tree on “e” 3

Item	Head of node-link
b	



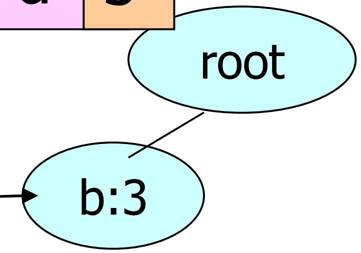
Cond. FP-tree on “g” 3

Item	Head of node-link
a	

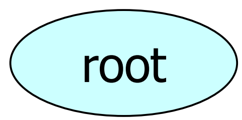


Cond. FP-tree on “d” 3

Item	Head of node-link
b	

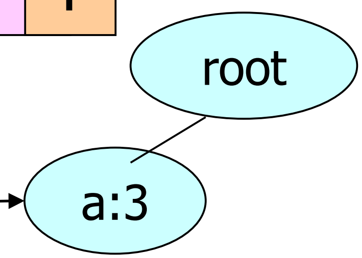


Cond. FP-tree on “f” 3



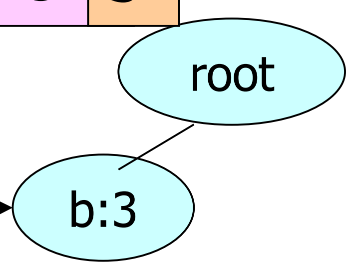
Cond. FP-tree on “b” 4

Item	Head of node-link
a	

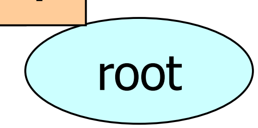


Cond. FP-tree on “e” 3

Item	Head of node-link
b	



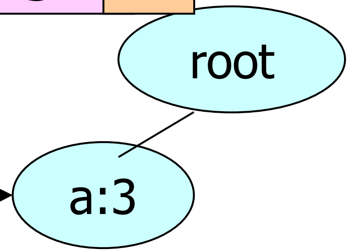
Cond. FP-tree on “a” 4



Cond. FP-tree on “g” 3

1. Before generating this cond. tree, we generate {g} (support = 3)

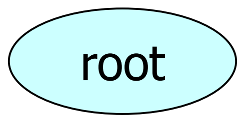
2. After generating this cond. tree, we generate {a, g} (support = 3)



Cond. FP-tree on “f” 3

1. Before generating this cond. tree, we generate {f} (support = 3)

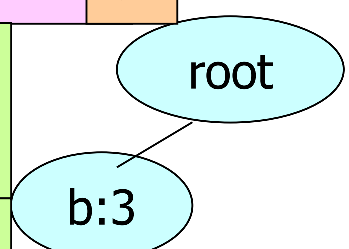
2. After generating this cond. tree, we do not generate any itemset.



Cond. FP-tree on “e” 3

1. Before generating this cond. tree, we generate {e} (support = 3)

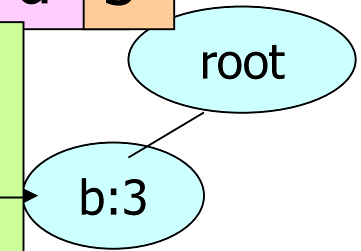
2. After generating this cond. tree, we generate {b, e} (support = 3)



Cond. FP-tree on “d” 3

1. Before generating this cond. tree, we generate {d} (support = 3)

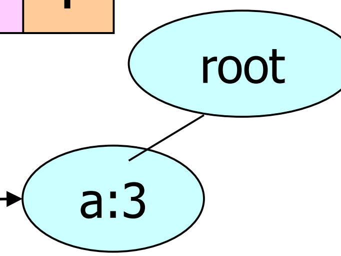
2. After generating this cond. tree, we generate {b, d} (support = 3)



Cond. FP-tree on “b” 4

1. Before generating this cond. tree, we generate {b} (support = 4)

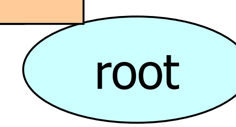
2. After generating this cond. tree, we generate {a, b} (support = 3)



Cond. FP-tree on “a” 4

1. Before generating this cond. tree, we generate {a} (support = 4)

2. After generating this cond. tree, we do not generate any itemset.



Complexity

- Complexity in building FP-tree
  - Two scans of the transactions DB
    - Collect frequent items
    - Construct the FP-tree
- Cost to insert one transaction
  - Number of frequent items in this transaction

Size of the FP-tree

- The size of the FP-tree is bounded by the overall occurrences of the frequent items in the database

Height of the Tree

- The height of the tree is bounded by the maximum number of frequent items in any transaction in the database

Compression

- With respect to the total number of items stored,
  - is FP-tree more compressed compared with the original databases?

Details of the Algorithm

- Procedure FP-growth (Tree, α)
  - if Tree contains a single path P
    - for each combination (denoted by β) of the nodes in the path P do
      - generate pattern $\beta \cup \alpha$ with support = minimum support of nodes in β
  - else
    - for each a_i in the header table of Tree do
      - generate pattern $\beta = a_i \cup \alpha$ with support = a_i.support
      - construct β's conditional pattern base and then β's conditional FP-tree $Tree_\beta$
      - if $Tree_\beta \neq \emptyset$
        - Call FP-growth($Tree_\beta$, β)

Rule Generation

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

    ABC $\rightarrow$D, ABD $\rightarrow$C, ACD $\rightarrow$B, BCD $\rightarrow$A,
    A $\rightarrow$BCD, B $\rightarrow$ACD, C $\rightarrow$ABD, D $\rightarrow$ABC
    AB $\rightarrow$CD, AC $\rightarrow$ BD, AD $\rightarrow$ BC, BC $\rightarrow$AD,
    BD $\rightarrow$AC, CD $\rightarrow$AB,

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \emptyset$ and $\emptyset \rightarrow L$)

Rule Generation

- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an anti-monotone property

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

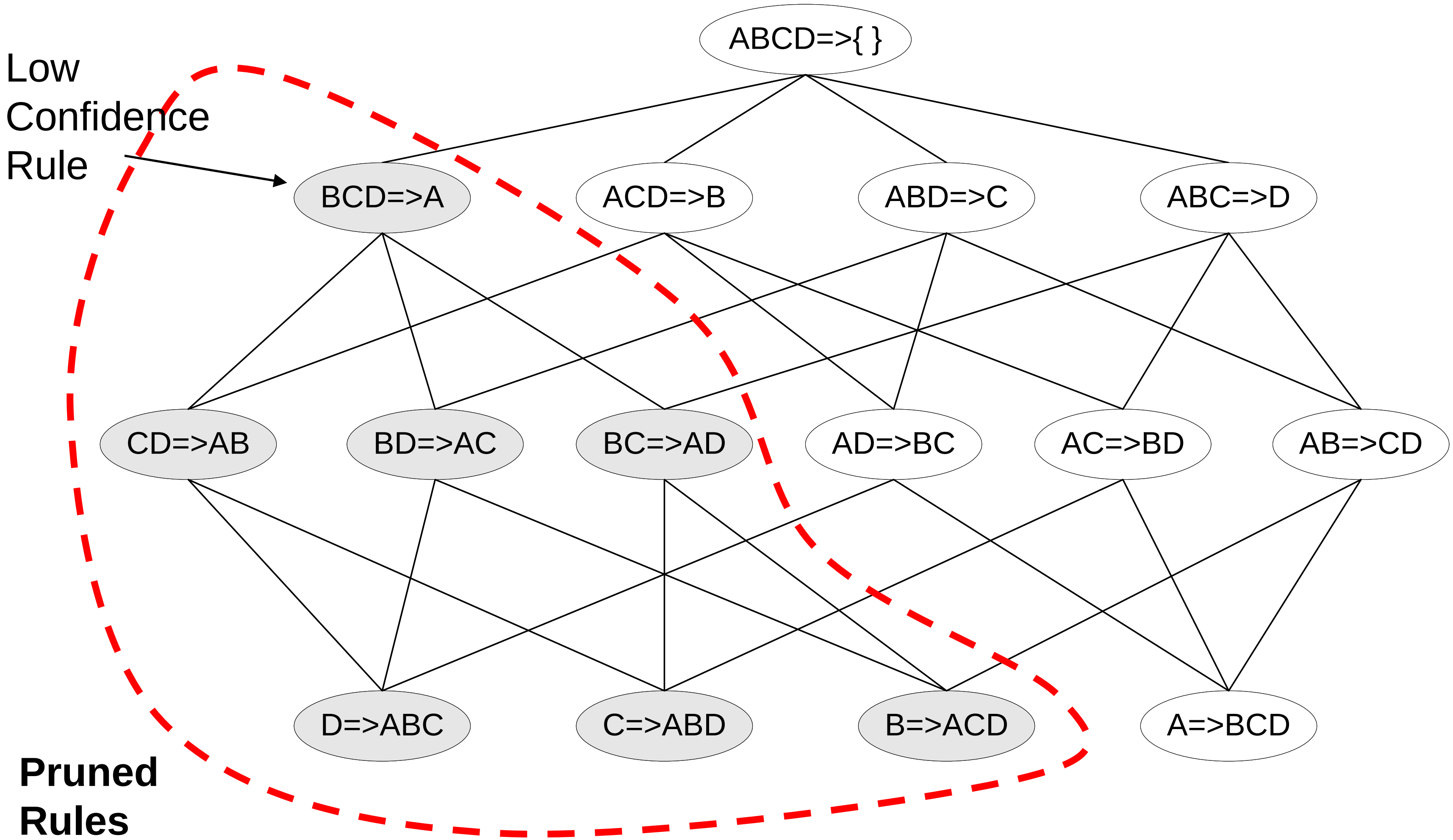
  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., L = {A,B,C,D}:

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

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

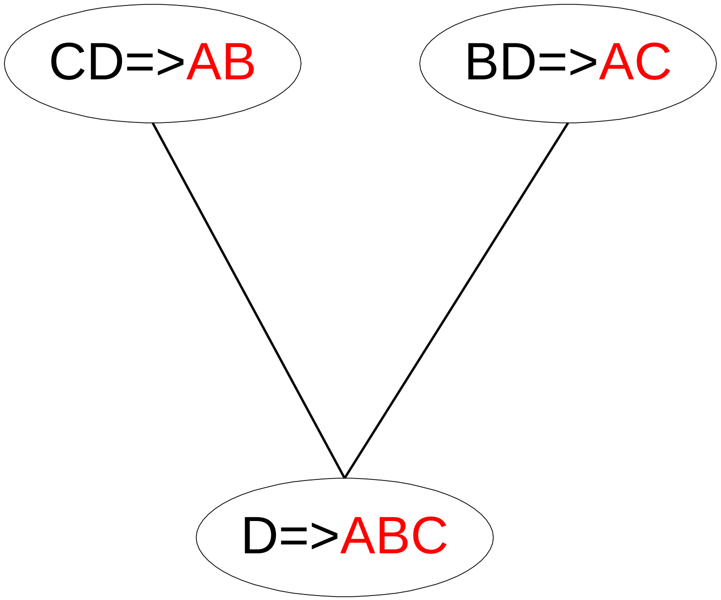
Rule Generation for Apriori Algorithm

Lattice of rules



Rule Generation for Apriori Algorithm

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



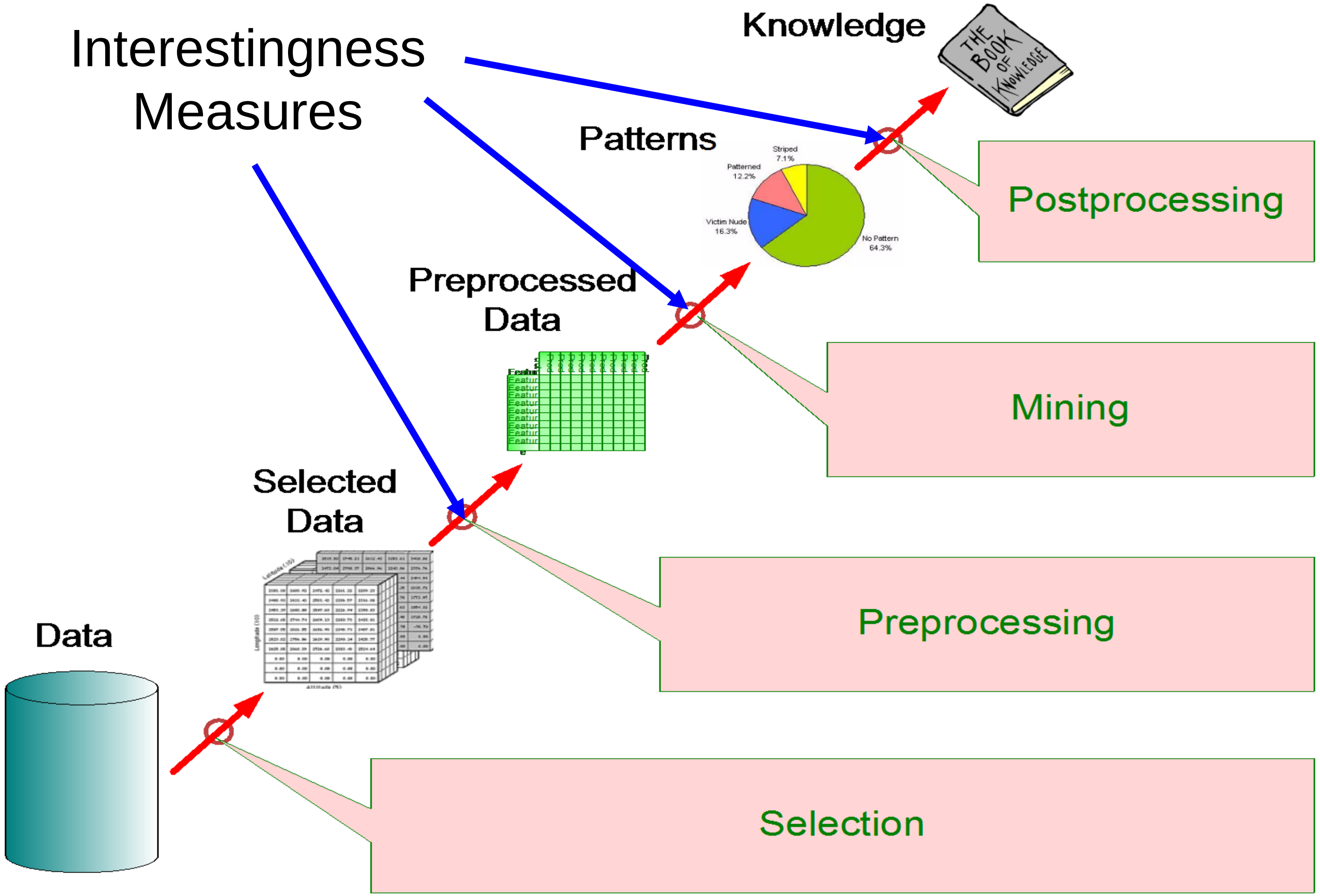
- join(CD=>AB,BD=>AC) would produce the candidate rule D => ABC

- Prune rule D=>ABC if its subset AD=>BC does not have high confidence

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

but P(Coffee) = 0.9

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

$\Rightarrow$ P(Coffee|$\overline{\text{Tea}}$) = 0.9375

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

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

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

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

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

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 Lift & Interest

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

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

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

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

Statistical independence:

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

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) 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

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

Property under Variable Permutation

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

	A	$\overline{\mathbf{A}}$
B	p	r
$\overline{\mathbf{B}}$	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	C	D	E	F
Transaction 1	1	0	0	1	0	0
.	0	0	1	1	1	0
.	0	0	1	1	1	0
.	0	0	1	1	1	0
.	0	1	1	0	1	1
.	0	0	1	1	1	0
	0	0	1	1	1	0
	0	0	1	1	1	0
	0	0	1	1	1	0
Transaction N	1	0	0	1	0	0

(a) — A, B; (b) — C, D; (c) — E, F

Example: ϕ-Coefficient

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

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

	Y	$\overline{\text{Y}}$	
X	20	10	30
$\overline{\text{X}}$	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

	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