DATA MINING
LECTURE 2

Frequent Itemsets

Association Rules

INTRODUCTION
SUMMARY

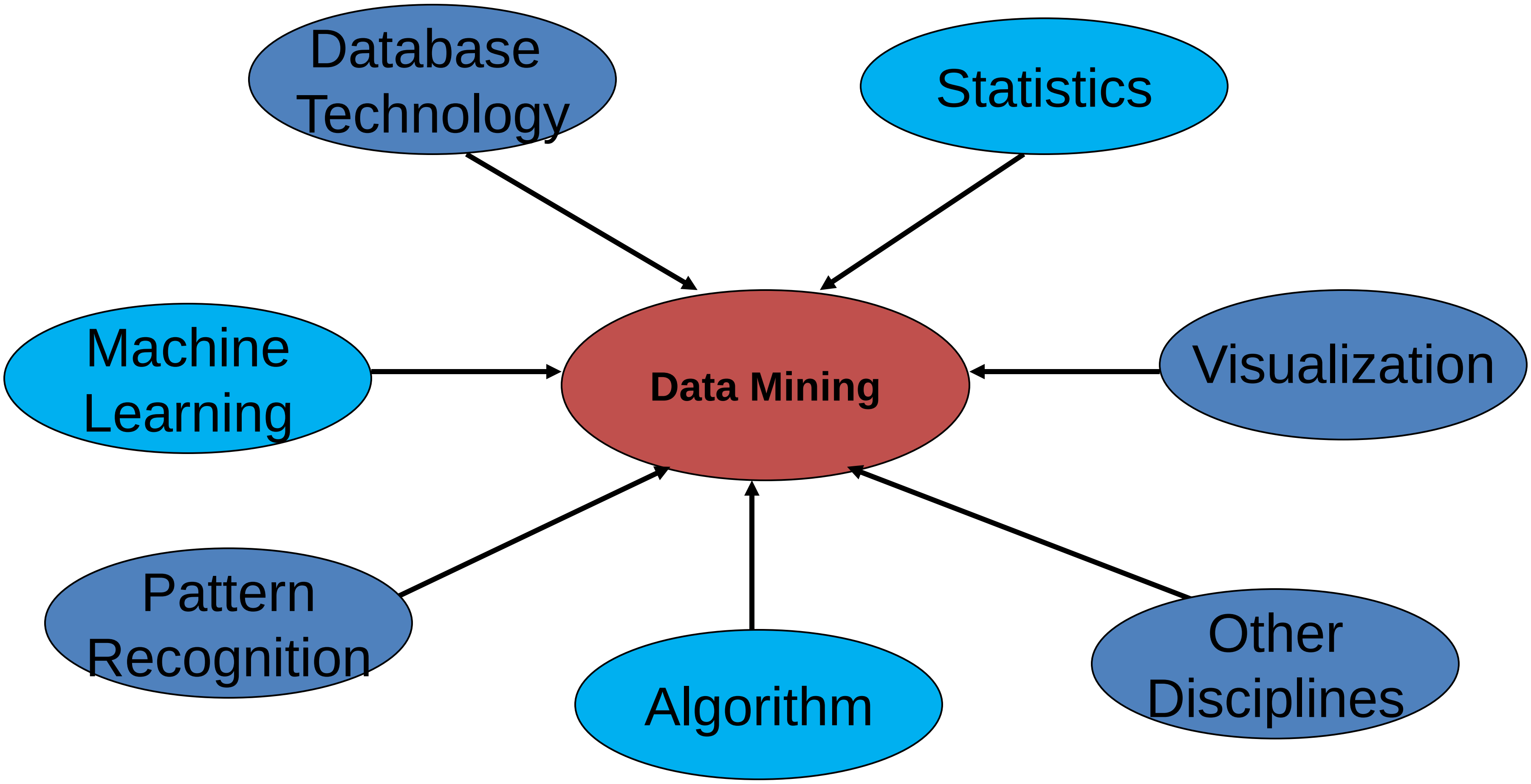
What is Data Mining?

- Data mining is the use of efficient techniques for the analysis of very large collections of data and the extraction of useful and possibly unexpected patterns in data.
- "Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data analyst" (Hand, Mannila, Smyth)
- "Data mining is the discovery of models for data" (Rajaraman, Ullman)
  - We can have the following types of models
    - Models that explain the data (e.g., a single function)
    - Models that predict the future data instances.
    - Models that summarize the data
    - Models the extract the most prominent features of the data.

Why do we need data mining?

- Really huge amounts of complex data generated from multiple sources and interconnected in different ways
  - Scientific data from different disciplines
    - Weather, astronomy, physics, biological microarrays, genomics
  - Huge text collections
    - The Web, scientific articles, news, tweets, facebook postings.
  - Transaction data
    - Retail store records, credit card records
  - Behavioral data
    - Mobile phone data, query logs, browsing behavior, ad clicks
  - Networked data
    - The Web, Social Networks, IM networks, email network, biological networks.
  - All these types of data can be combined in many ways
    - Facebook has a network, text, images, user behavior, ad transactions.
- We need to analyze this data to extract knowledge
  - Knowledge can be used for commercial or scientific purposes.
  - Our solutions should scale to the size of the data

Data Mining: Confluence of Multiple Disciplines



An example of a data mining challenge

- We are given a stream of numbers (identifiers, etc). We want to answer simple questions:
  - How many numbers are there?
  - How many distinct numbers are there?
  - What are the most frequent numbers?
  - What is the mean of the numbers? Or the median?
  - How many numbers appear at least K times?
  - Etc.
- These questions are simple if we have resources (time and memory).
- In our case we have neither, since the data is streaming.

Finding the majority element

- A stream of identifiers; one of them occurs more than 50% of the time
- How can you find it using no more than a few memory locations?
- Suggestions?

Finding the majority element (solution)

```
A = first item you see; count = 1
for each subsequent item x
        if (A == x) count = count + 1
        else {
                count = count - 1
                if (count == 0) {A=x; count = 1}
             }
  endfor
  return A
```

Why does this work correctly?

Finding the majority element (solution and correctness proof)

```
A = first item you see; count = 1
for each subsequent item x
      if (A==x) count = count + 1
      else {
                  count = count - 1
                   if (count == 0)
                      {A=x; count = 1}
                 }
endfor
return A
```

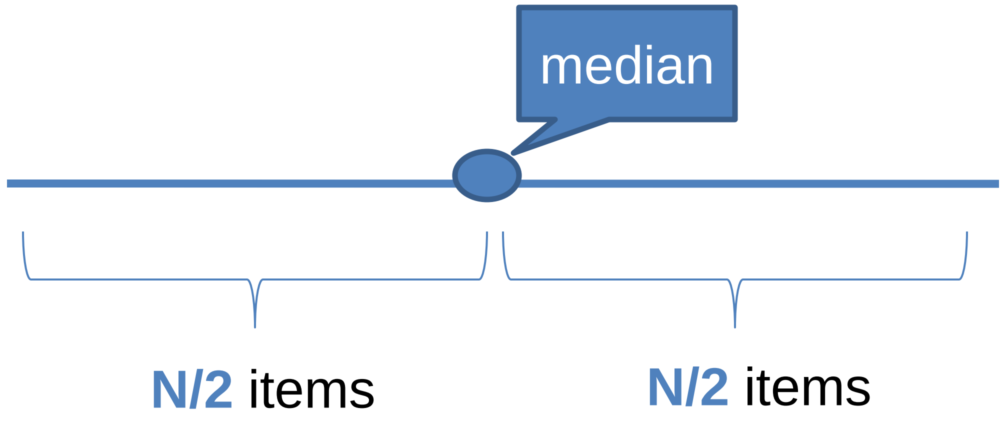
- An occurrence of an identifier **u** is discarded if
  - A=u and the counter is decreased.
  - The identifier u causes the counter to decrease
- **Basic observation:** Whenever we discard an occurrence of the majority element **m** we also discard an occurrence of an element **u** different from **m**

Finding a number in the top half

- Given a set of N numbers (N is very large)
- Find a number x such that x is ***likely*** to be larger than the **median** of the numbers
- Simple solution
  - Sort the numbers and store them in sorted array **A**
  - Any value larger than **A[N/2]** is a solution
- Other solutions?

Finding a number in the top half ***efficiently***

- A solution that uses small number of operations
  - Randomly sample **K** numbers from the file
  - Output their maximum



- Failure probability **(1/2)^K**

FREQUENT ITEMSETS & ASSOCIATION RULES

Thanks to:

Tan, Steinbach, Kumar, “Introduction to Data Mining”

Evimaria Terzi

Evaggelia Pitoura

This is how it all started…

- Rakesh Agrawal, Tomasz Imielinski, Arun N. Swami: Mining Association Rules between Sets of Items in Large Databases. SIGMOD Conference 1993: 207-216
- Rakesh Agrawal, Ramakrishnan Srikant: Fast Algorithms for Mining Association Rules in Large Databases. VLDB 1994: 487-499

- These two papers are credited with the birth of Data Mining
- For a long time people were fascinated with Association Rules and Frequent Itemsets
  - Some people (in industry and academia) still are.

Frequent Itemsets

- Given a set of transactions, find combinations of items (itemsets) that occur frequently

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

Examples of frequent itemsets

{Diaper, Beer} : 3
{Milk, Bread} : 3
{Milk, Bread, Diaper}: 2

Binary matrix representation

- Our data can also be represented as a 0/1 matrix
  - Rows: transactions
  - Columns: items
  - 1: item bought, 0: item not bought
    - Asymmetric: we care more about 1’s than 0’s

  - We lose information about counts

- A variety of data can be represented like that
  - E.g., Document-words data, biological data, etc

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

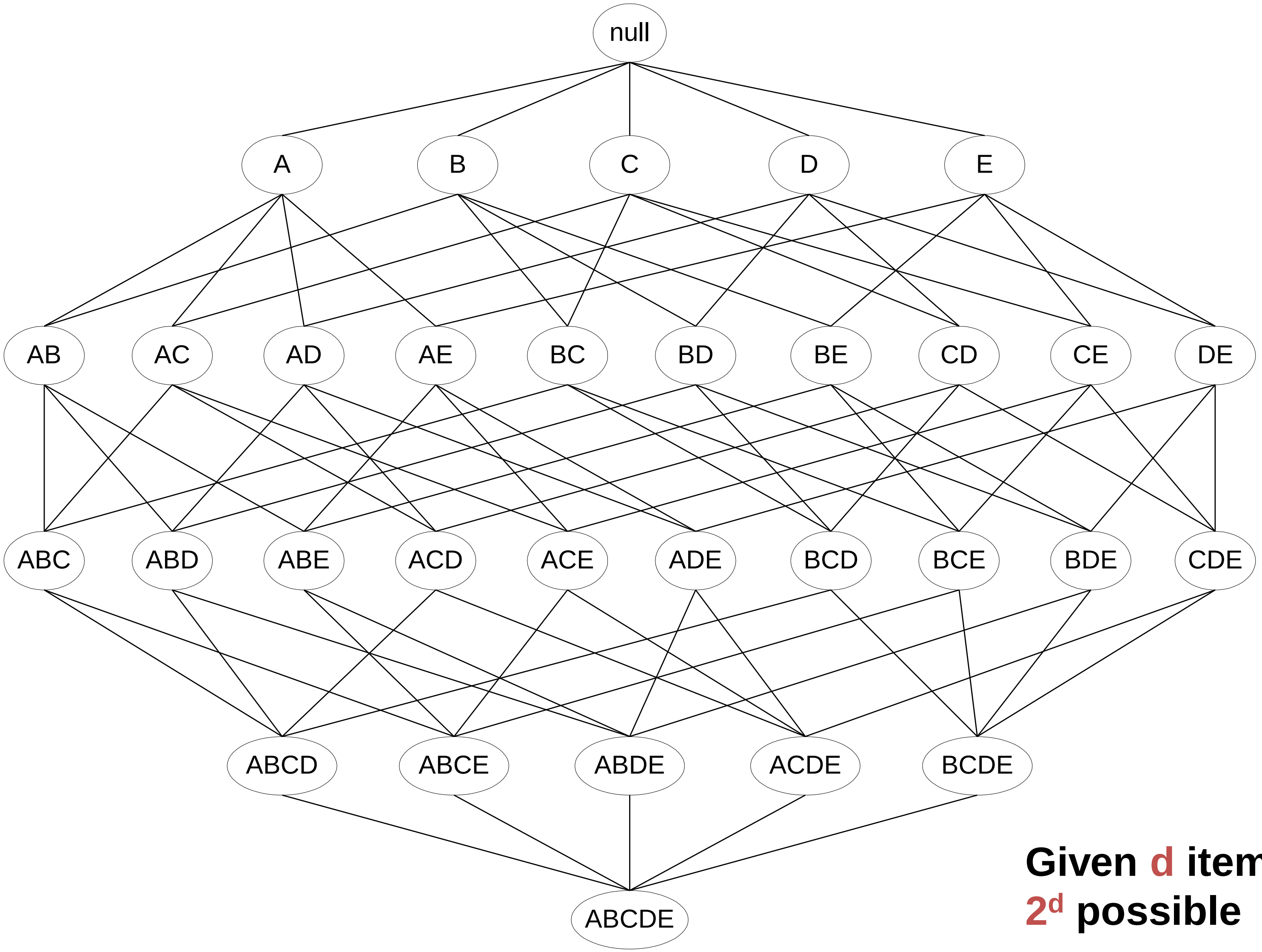
$$s(I) \geq \text{minsup}$$

Mining Frequent Itemsets task

- Input: A set of transactions T, over a set of items I
- Output: All itemsets with items in I having
  - support ≥ *minsup* threshold

- Problem parameters:
  - N = |T|: number of transactions
  - d = |I|: number of (distinct) items
  - w: max width of a transaction
  - Number of possible itemsets? $= 2^d$

The itemset lattice



Given d items, there are 2^d possible itemsets

A Naïve Algorithm

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database
  - Match each transaction against every candidate
  - Time Complexity ~ O(NMw) , Space Complexity ~ O(M)
    - Expensive since $M = 2^d$!!!

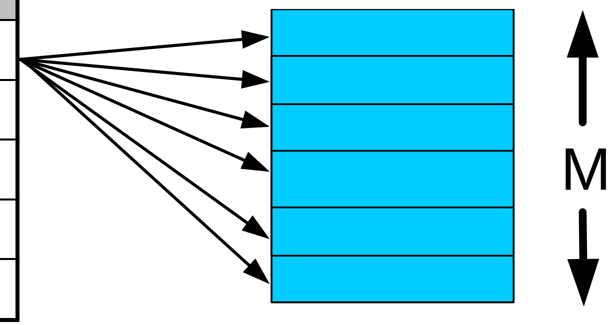
Transactions

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

List of Candidates



Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M

Any ideas?

- 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

Reduce the number of candidates

- Apriori principle (Main observation):
  - 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:

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

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

Illustration of the Apriori principle

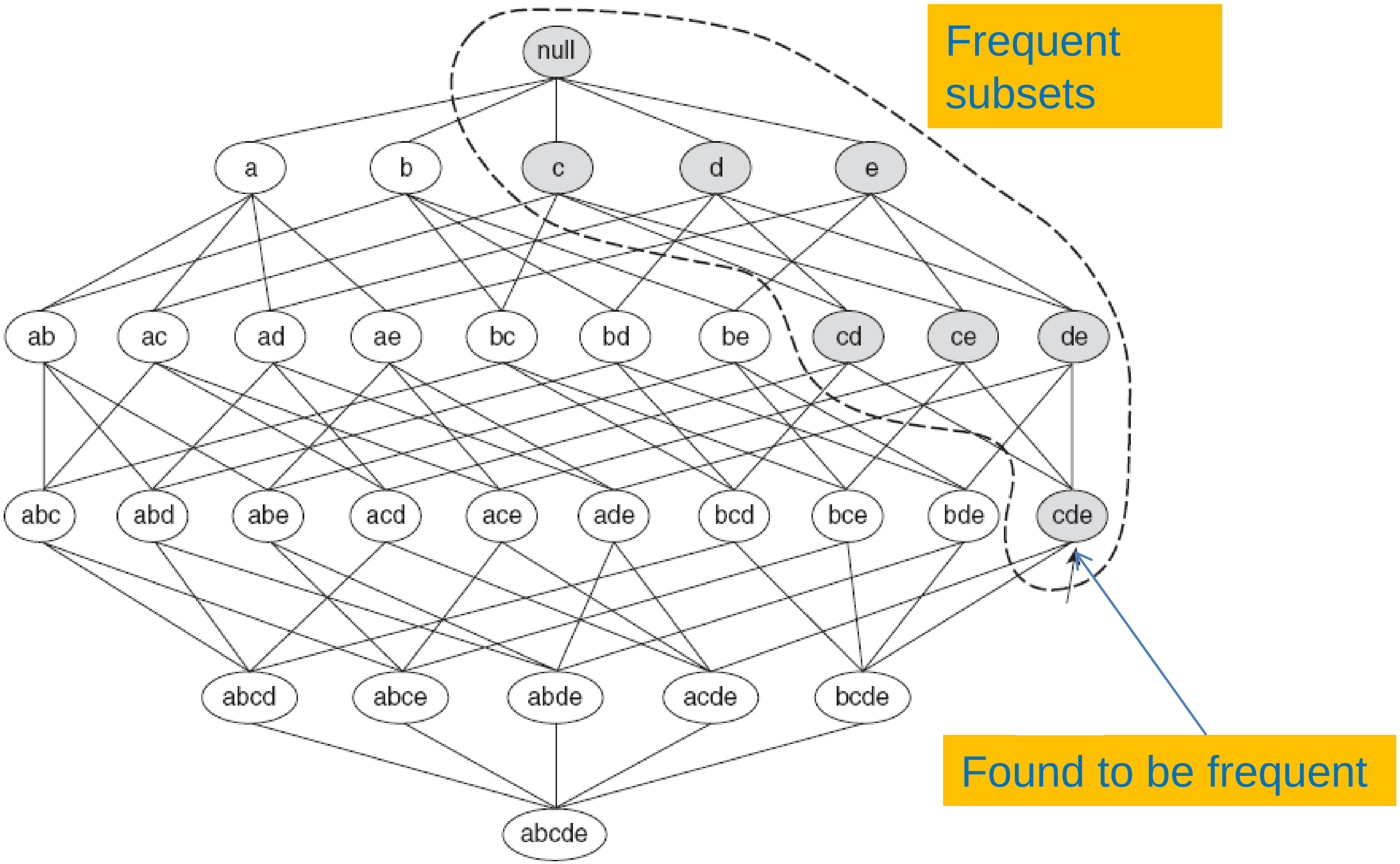


Figure 6.3. An illustration of the *Apriori* principle. If $\{c, d, e\}$ is frequent, then all subsets of this itemset are frequent.

Illustration of the Apriori principle

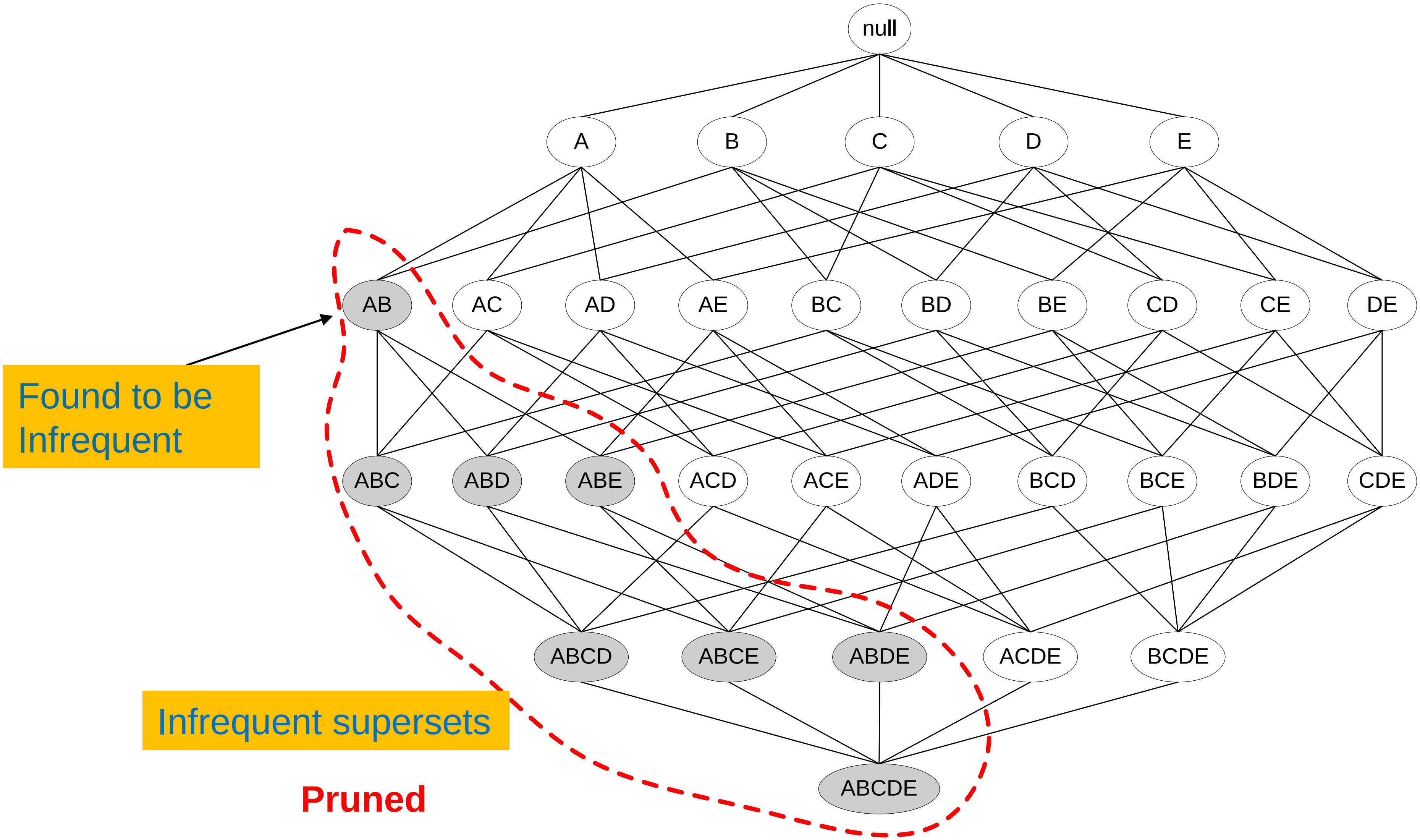


Illustration of the Apriori principle

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

minsup = 3

Items (1-itemsets)

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

Pairs (2-itemsets)

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

(No need to generate candidates involving Coke or Eggs)

Triplets (3-itemsets)

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

Only this triplet has all subsets to be frequent
But it is below the minsup threshold

If every subset is considered,

$$\binom{6}{1} + \binom{6}{2} + \binom{6}{3} = 6 + 15 + 20 = 41$$

With support-based pruning,

$$\binom{6}{1} + \binom{4}{2} + 1 = 6 + 6 + 1 = 13$$

The Apriori algorithm

1. Find frequent 1-items and put them to L_k ($k=1$)
2. Use L_k to generate a collection of *candidate* itemsets C_{k+1} with size ($k+1$)
3. Scan the database to find which itemsets in C_{k+1} are frequent and put them into L_{k+1}
4. If L_{k+1} is not empty
   - $k=k+1$
   - Goto step 2

R. Agrawal, R. Srikant: "Fast Algorithms for Mining Association Rules", *Proc. of the 20th Int'l Conference on Very Large Databases*, 1994.

The Apriori algorithm

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

L_1 = {frequent 1-itemsets};
for (k = 2; L_k !=∅; k++)
  C_{k+1} = GenerateCandidates(L_k)
  for each transaction t in database do
    increment count of candidates in C_{k+1} that are contained in **t**
  endfor
  L_{k+1} = candidates in C_{k+1} with support ≥***min_sup***
endfor
return $\cup_k L_k$;

Generate Candidates C_{k+1}

- Any ideas?

- We know the frequent itemsets of size k, L_k
- We know that every itemset in C_{k+1} should have frequent subsets

- Construct C_{k+1} from the itemsets in L_k

Generate Candidates C_{k+1}

- Assume the items in L_k are listed in an order (e.g., alphabetical)
- **Step 1:** ***self-joining*** L_k *(IN SQL)*

insert into C_{k+1}

select $p.item_1, p.item_2, \ldots, p.item_k, q.item_k$

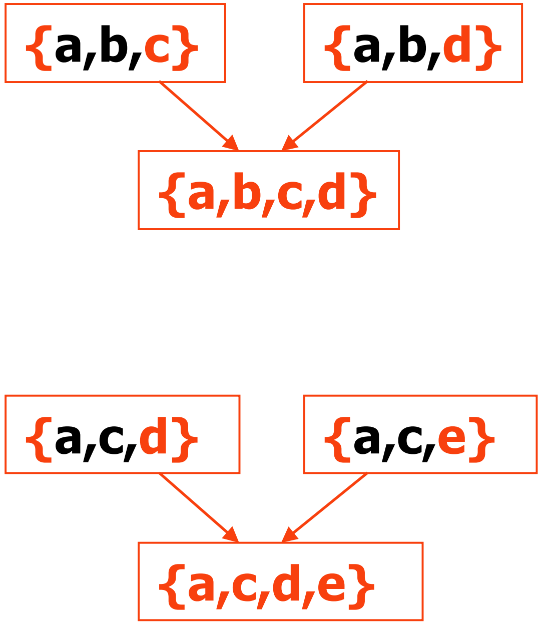
from $L_k\ p, L_k\ q$

where $p.item_1=q.item_1, \ldots, p.item_{k-1}=q.item_{k-1}, p.item_k < q.item_k$

Create an itemset of size k+1, by joining two itemsets of size k, that share the first k-1 items

Example I

- L_3={*abc, abd, acd, ace, bcd*}
- ***Self-joining***: $L_3 * L_3$
  - ***abcd*** from ***abc*** and ***abd***
  - ***acde*** from ***acd*** and ***ace***



Example II

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

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

Itemset
{Bread,Diaper,Milk}

Generate Candidates C_{k+1}

- Assume the items in L_k are listed in an order (e.g., alphabetical)
- **Step 1:** ***self-joining*** L_k *(IN SQL)*

  insert into C_{k+1}

  select $p.item_1, p.item_2, \ldots, p.item_k, q.item_k$

  from $L_k\ p, L_k\ q$

  where $p.item_1=q.item_1, \ldots, p.item_{k-1}=q.item_{k-1}, p.item_k < q.item_k$

- **Step 2:** ***pruning***

  forall ***itemsets c in*** C_{k+1} do

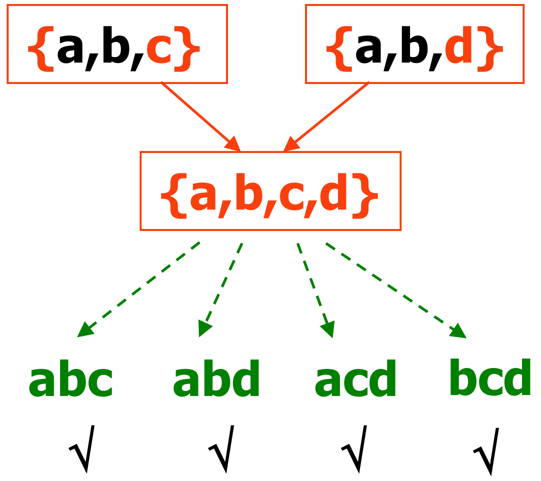
  forall ***k-subsets s of c*** do

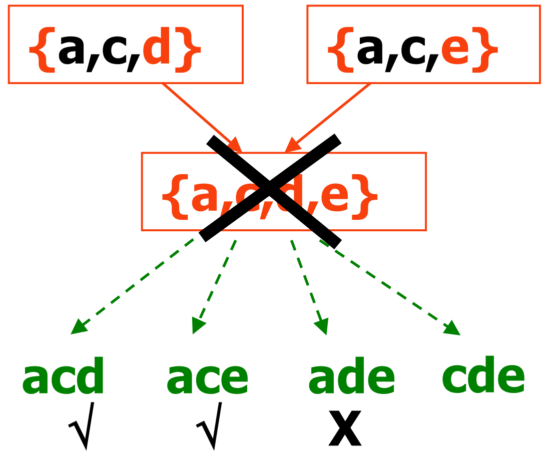
  if *(s is **not** in* L_k*)* **then delete c from** C_{k+1}

All itemsets of size k that are subsets of a new (k+1)-itemset should be frequent

Example I

- L_3={abc, abd, acd, ace, bcd}
- ***Self-joining***: L_3*L_3
  - ***abcd*** from ***abc*** and ***abd***
  - ***acde*** from ***acd*** and ***ace***
- ***Pruning:***
  - ***abcd*** is kept since all subset itemsets are in L_3
  - ***acde*** is removed because ***ade*** is not in L_3
- C_4={abcd}





Example II

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

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

Itemset
{Bread,Diaper,Milk}

{**Bread,Diaper**} √

{**Bread,Milk**} √

{**Diaper, Milk**} √

Example II – Alternative

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

Item	Count
Bread	**4**
Milk	**4**
Beer	**3**
Diaper	**4**

Itemset
{Beer,Bread,Diaper}
{Beer,Bread,Milk}
{Beer,Diaper,Milk}
{Bread,Diaper,Milk}

Joining with the L_1 set generates more candidates that need to be pruned.

The Apriori algorithm

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

L_1 = {frequent 1-itemsets};
for (k = 2; L_k !=∅; k++)
C_{k+1} = GenerateCandidates(L_k)
for each transaction ***t*** in database do
increment count of candidates in C_{k+1} that are contained in **t**
endfor
L_{k+1} = candidates in C_{k+1} with support ≥***min_sup***
endfor
return $\cup_k L_k$;

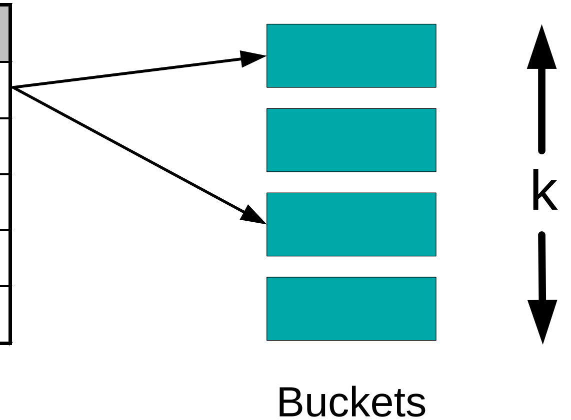
Reducing Number of Comparisons

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

Transactions

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



How to Count Supports of Candidates?

- Method:
  - Candidate itemsets are stored in a *hash-tree*
  - *Leaf* node of hash-tree contains a list of itemsets and counts
  - *Interior* node contains a hash table
  - *Subset operation*: finds all the candidates contained in a transaction

Generate Hash Tree

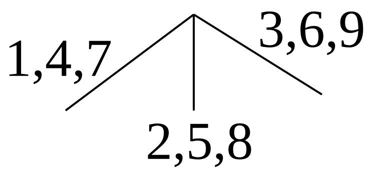
Suppose you have 15 candidate itemsets of length 3:

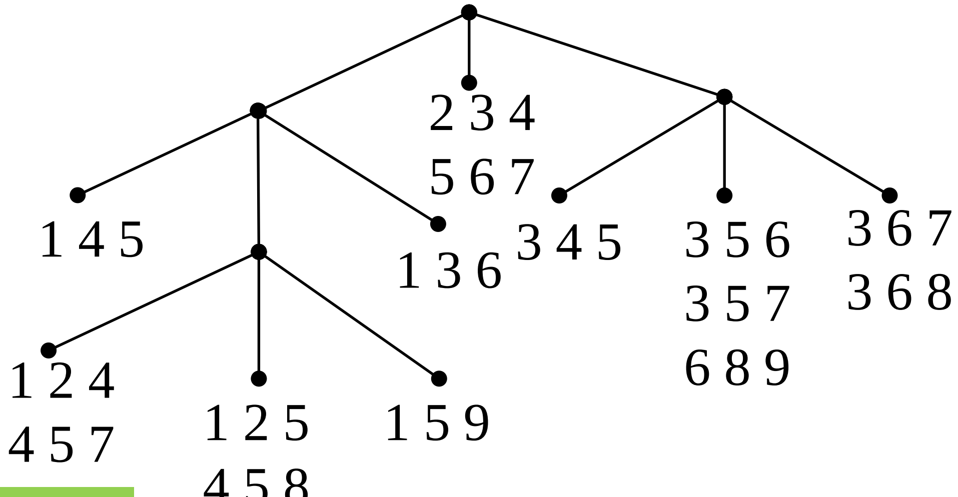
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

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)

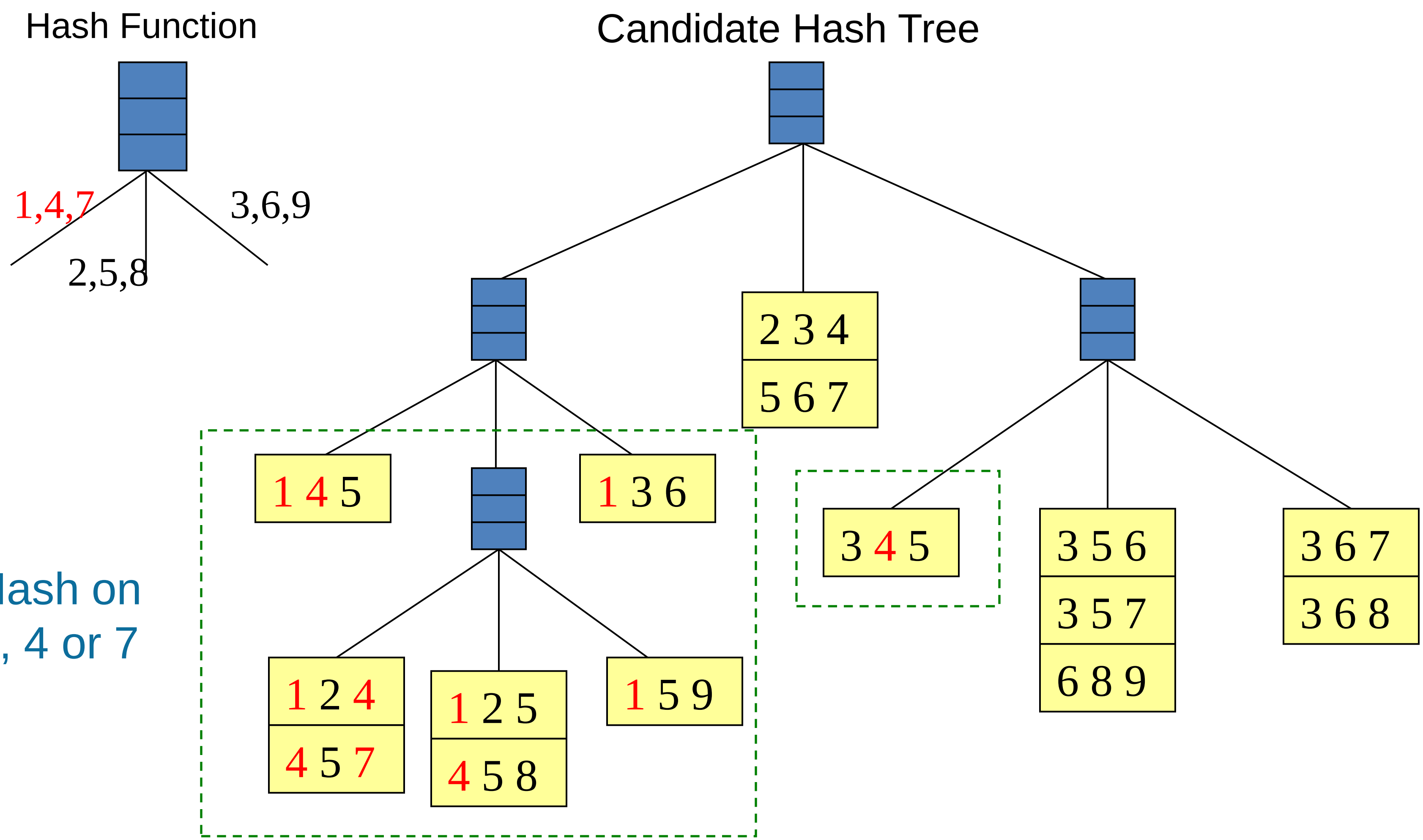
Hash function = mod 3





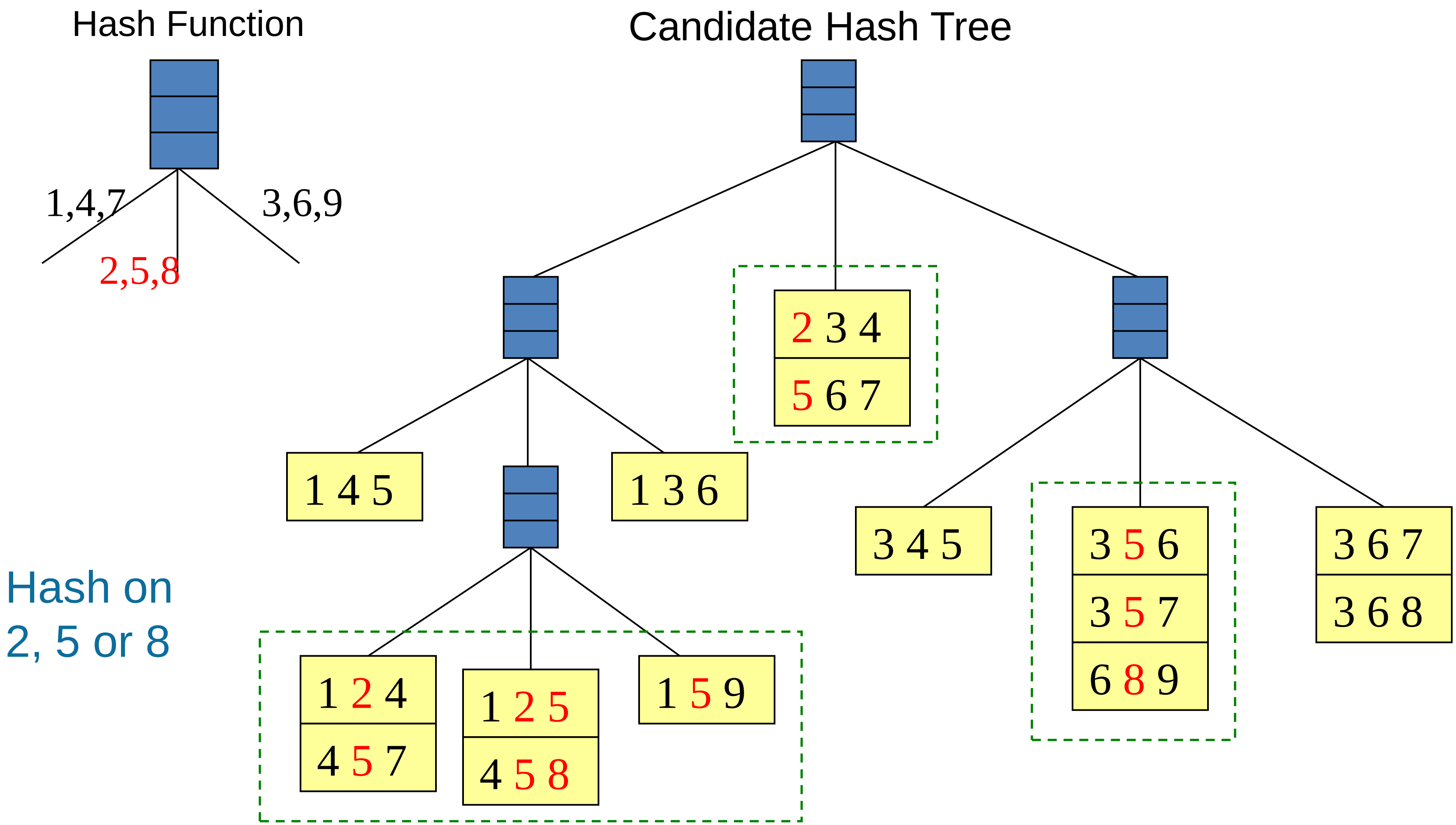
At the i-th level we hash at the i-th item

Association Rule Discovery: Hash tree

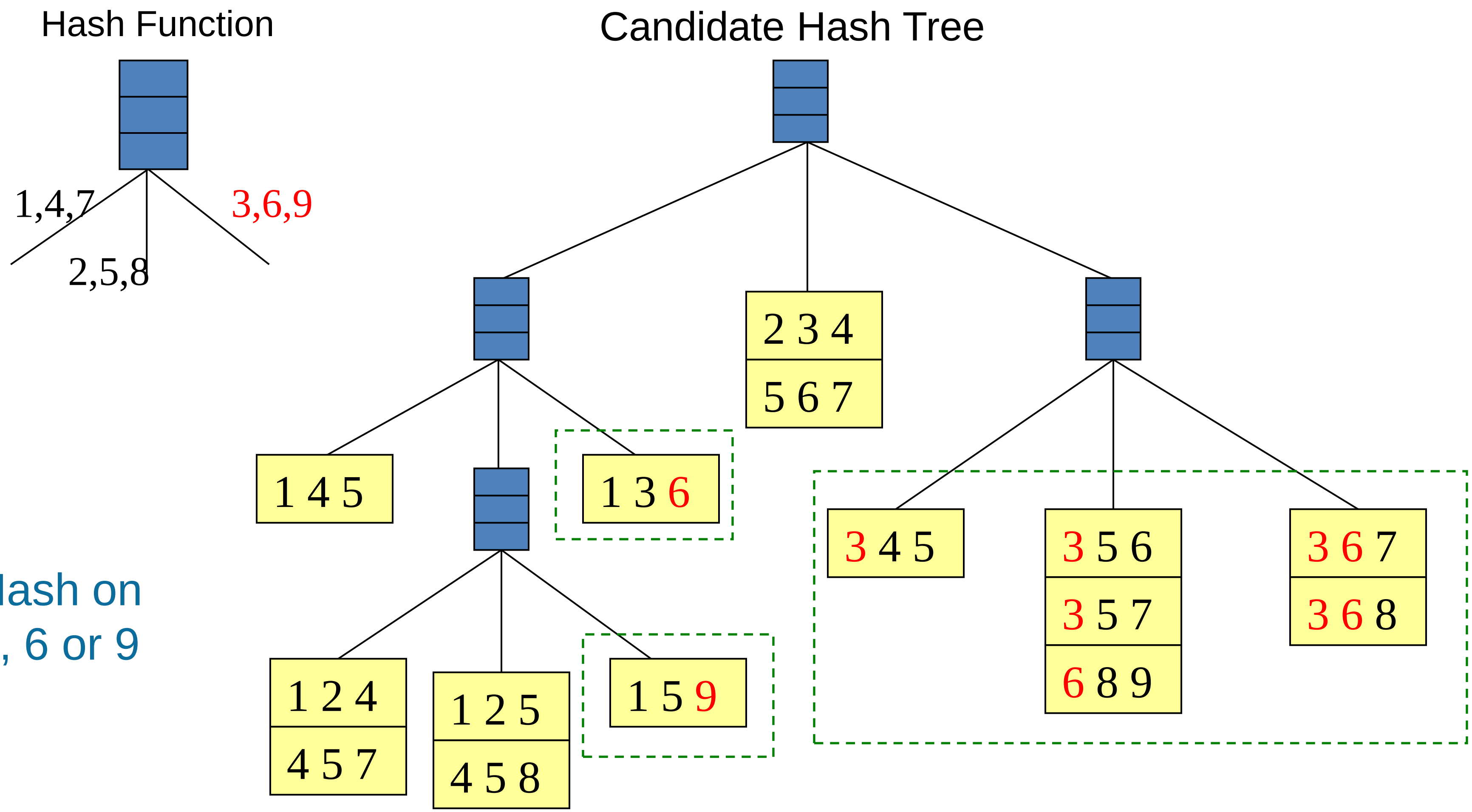


Hash on
1, 4 or 7

Association Rule Discovery: Hash tree



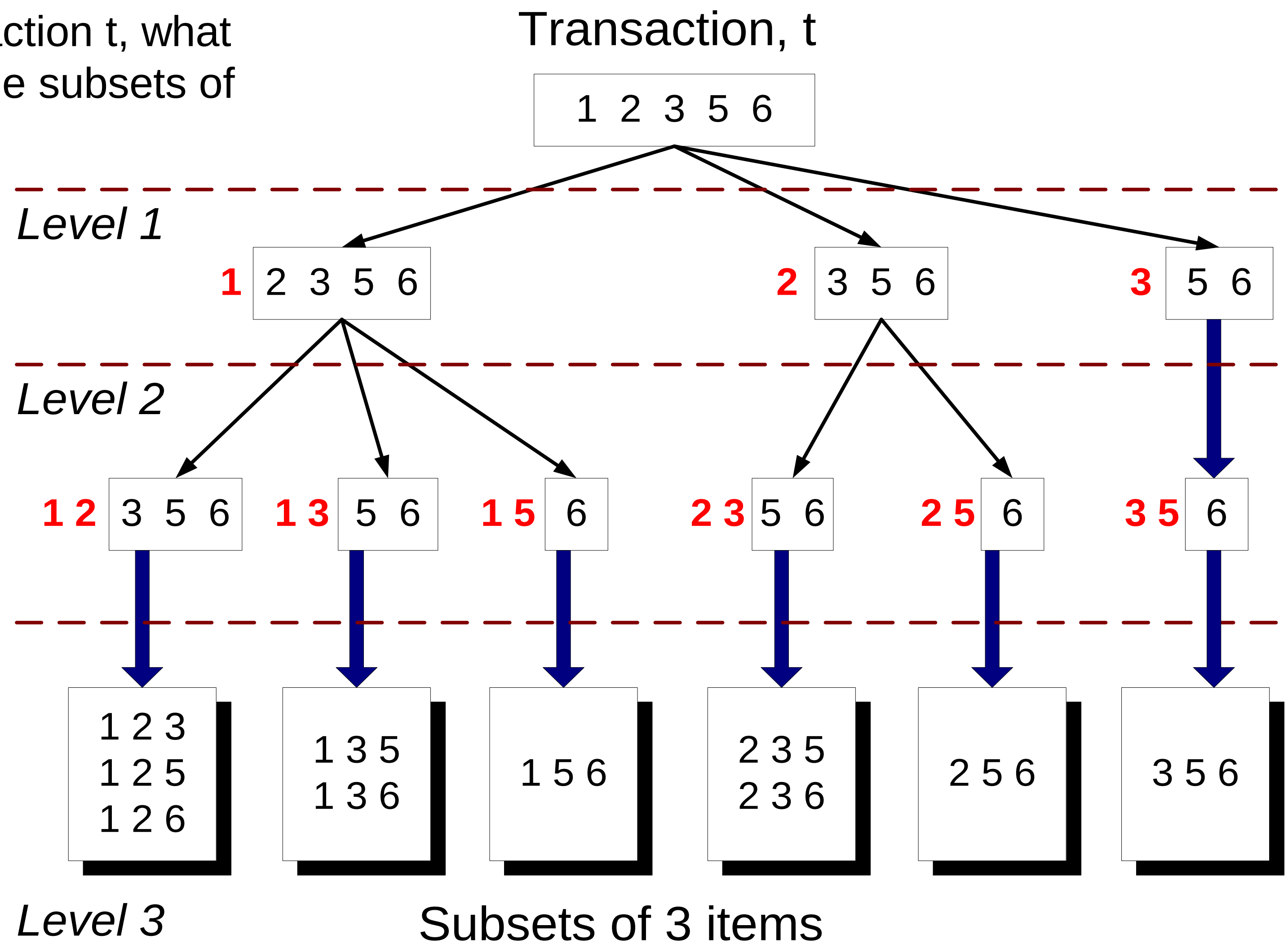
Association Rule Discovery: Hash tree



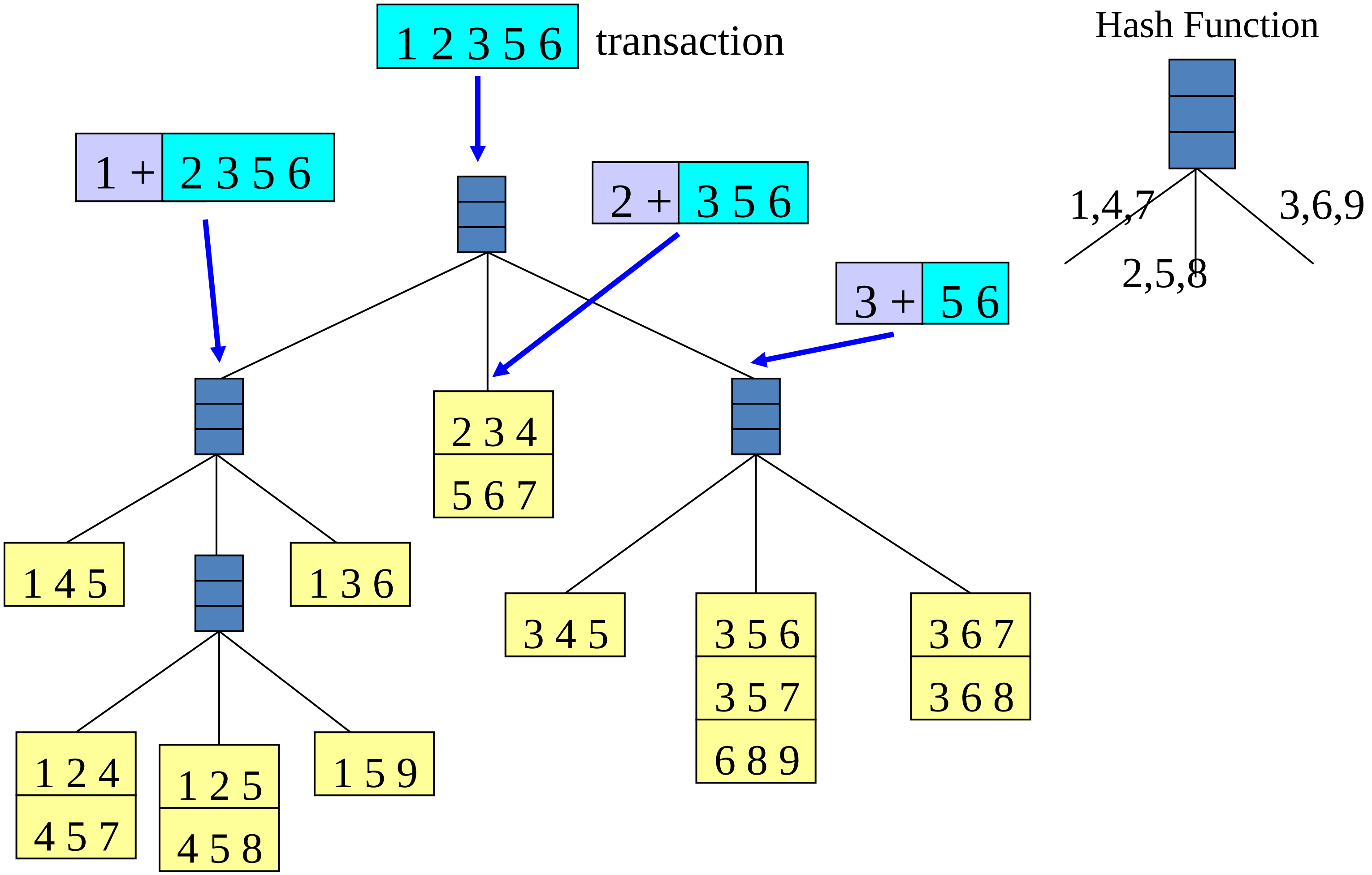
Hash on
3, 6 or 9

Subset Operation

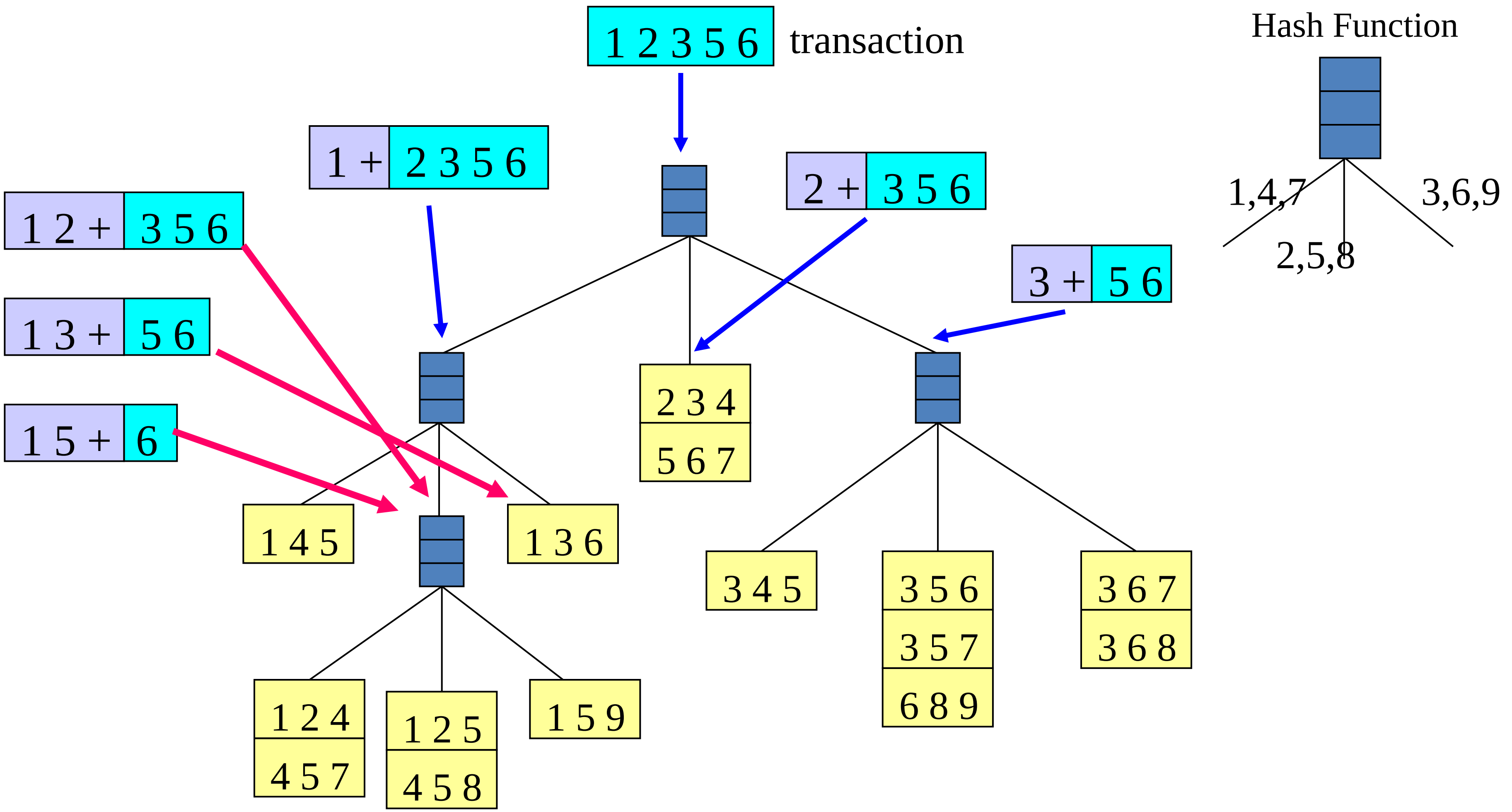
Given a transaction t, what are the possible subsets of size 3?



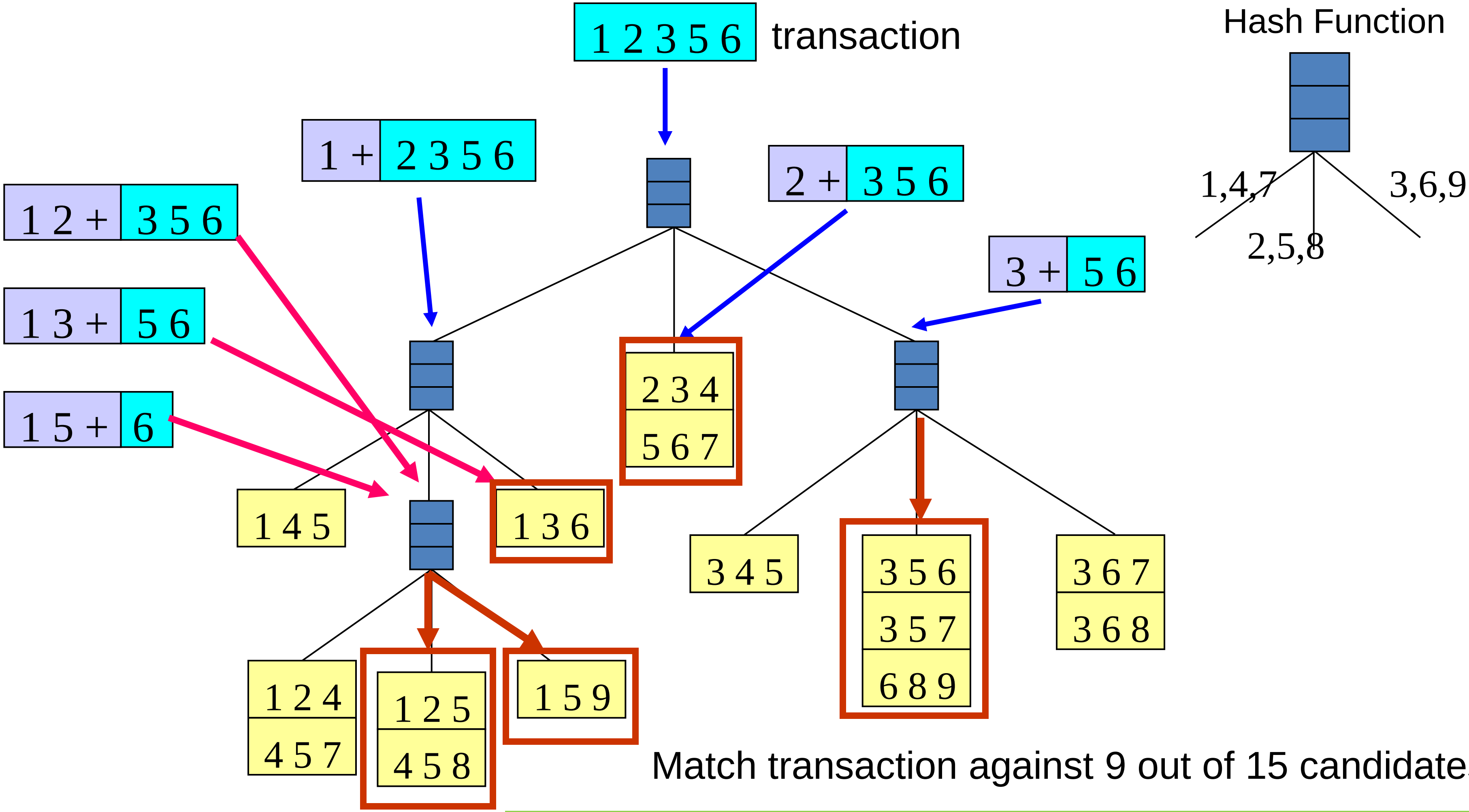
Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Match transaction against 9 out of 15 candidates

Hash-tree enables to enumerate itemsets in transaction and match them against 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)

ASSOCIATION RULES

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: Association Rule

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

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:

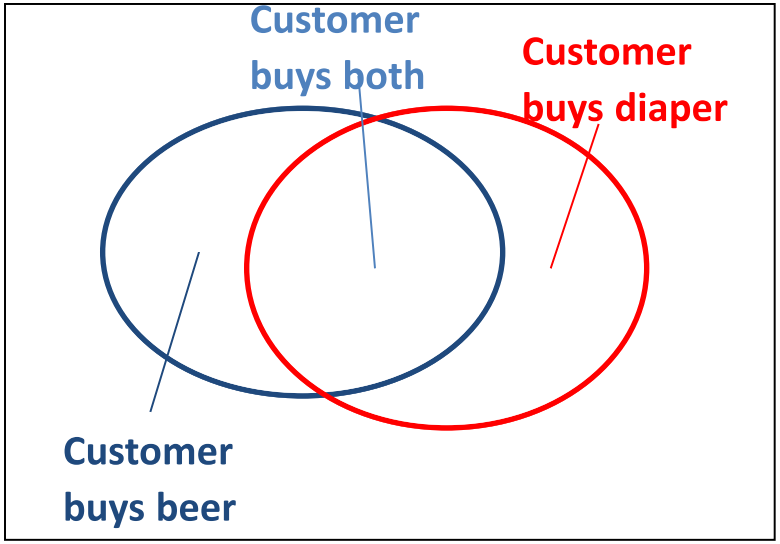
$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

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

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

Support and Confidence

- For association rule $X \rightarrow Y$:
  - Support $s(X \rightarrow Y)$: the probability $P(X,Y)$ that X and Y occur together
  - Confidence $c(X \rightarrow Y)$: the conditional probability $P(X|Y)$ that X occurs given that Y has occurred.



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

Support, Confidence of rule

- *A → C* (50%, 66.6%)
- *C → A* (50%, 100%)

Association Rule Mining Task

- Input: A set of transactions T, over a set of items I
- Output: All rules with items in I having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

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

Rule Generation

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

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

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

Rule Generation

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

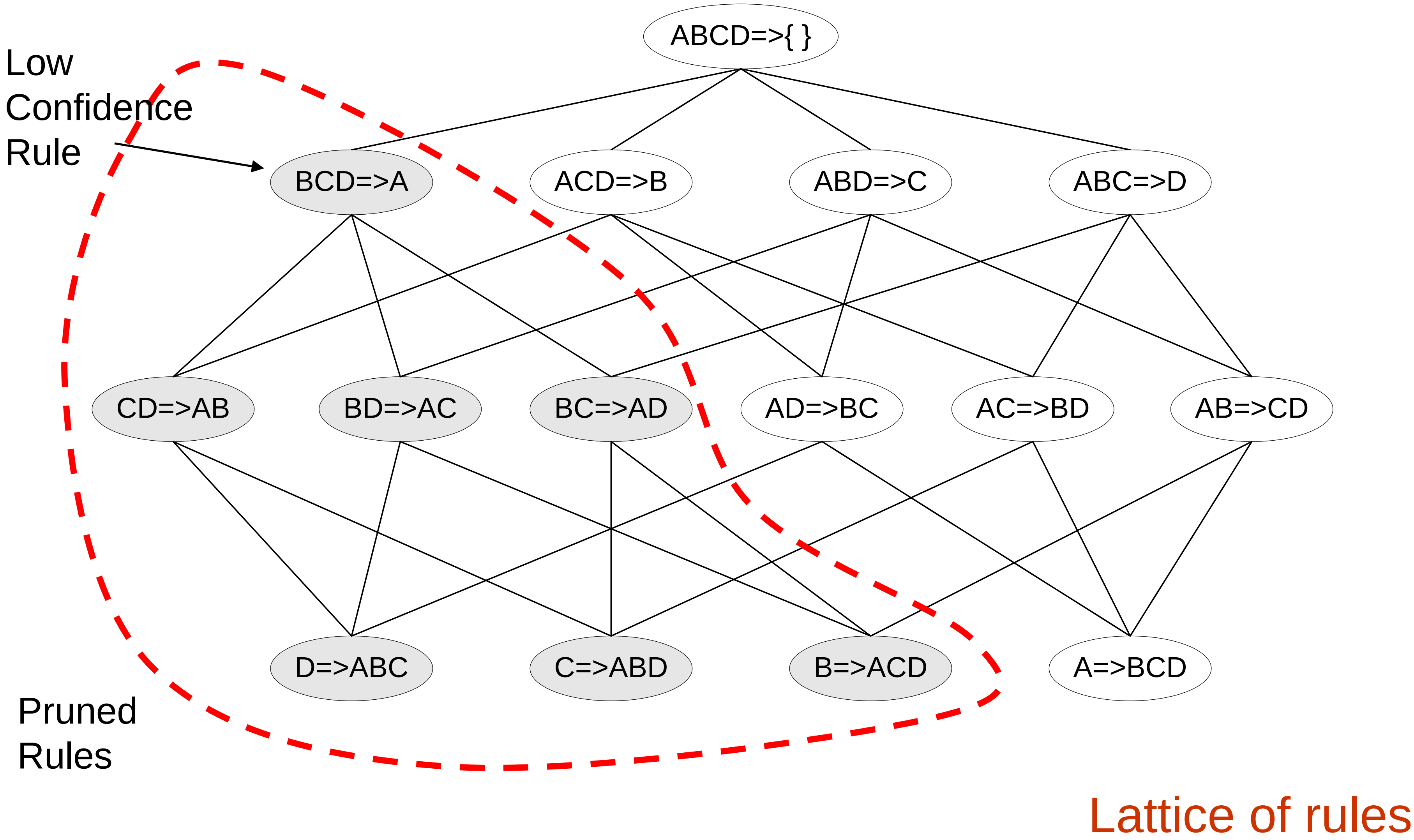
    $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

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

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

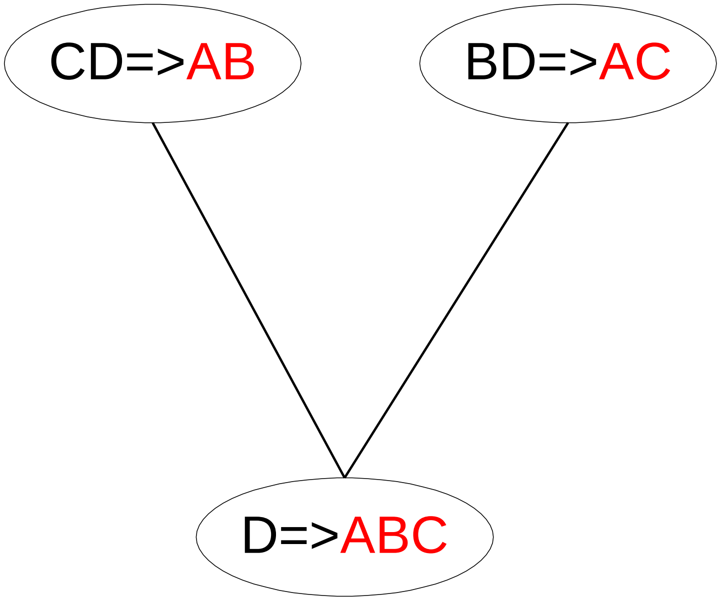
- Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule Generation for Apriori Algorithm



Rule Generation for Apriori Algorithm

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



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

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

RESULT POST-PROCESSING

Compact Representation of Frequent Itemsets

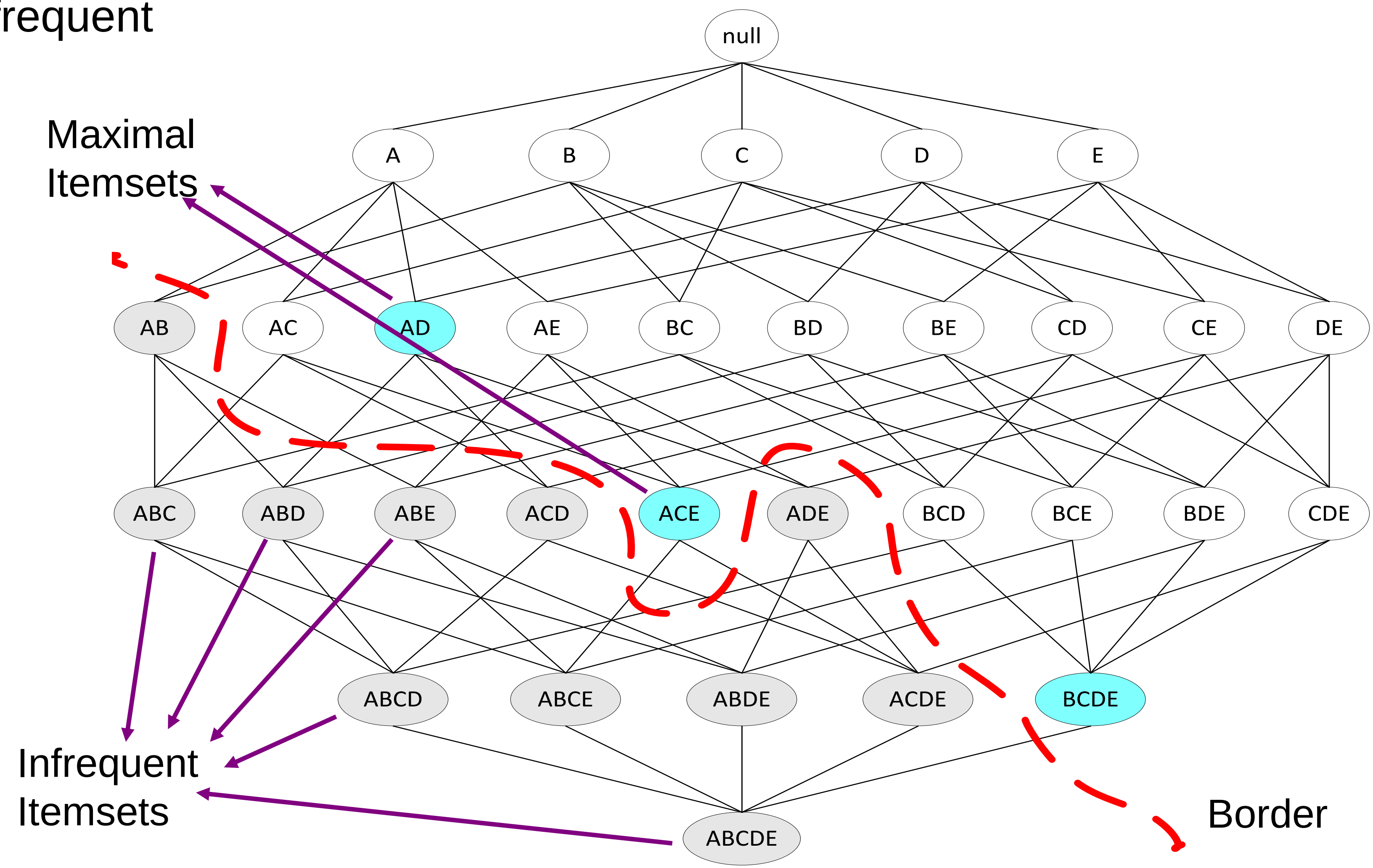
- Some itemsets are redundant because they have identical support as their supersets

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

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

Maximal Frequent Itemset

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



Closed Itemset

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

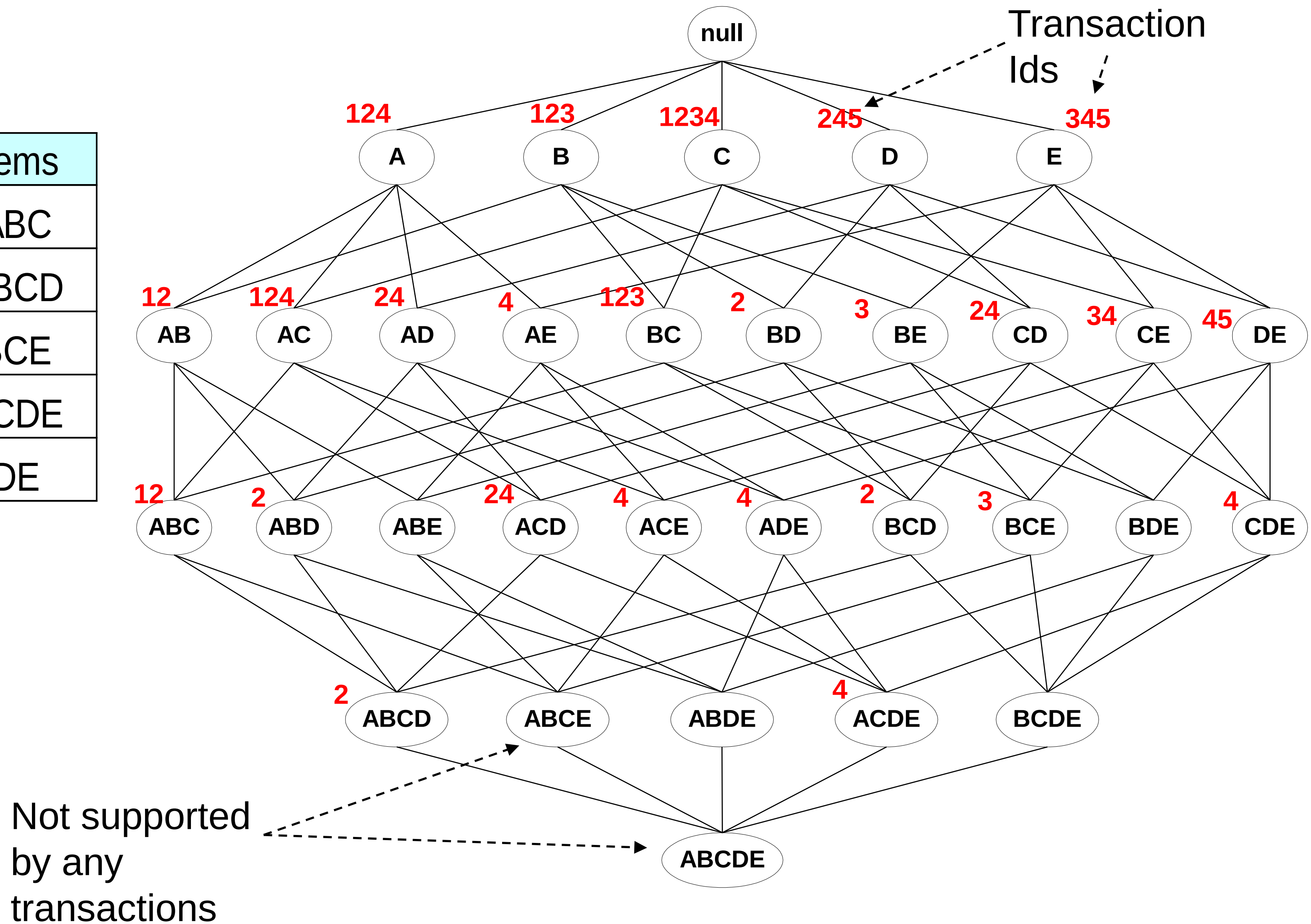
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

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

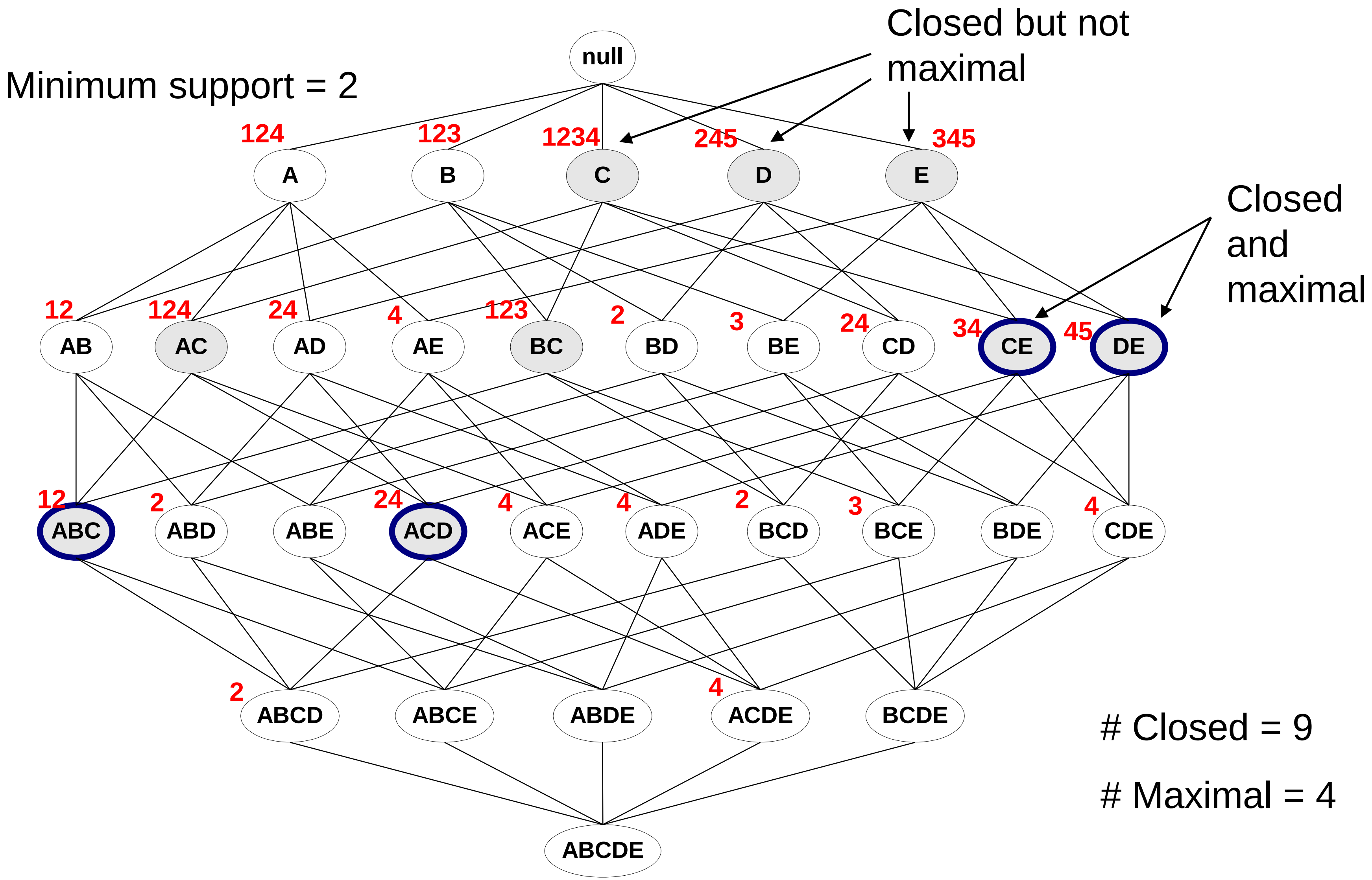
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Maximal vs Closed Itemsets

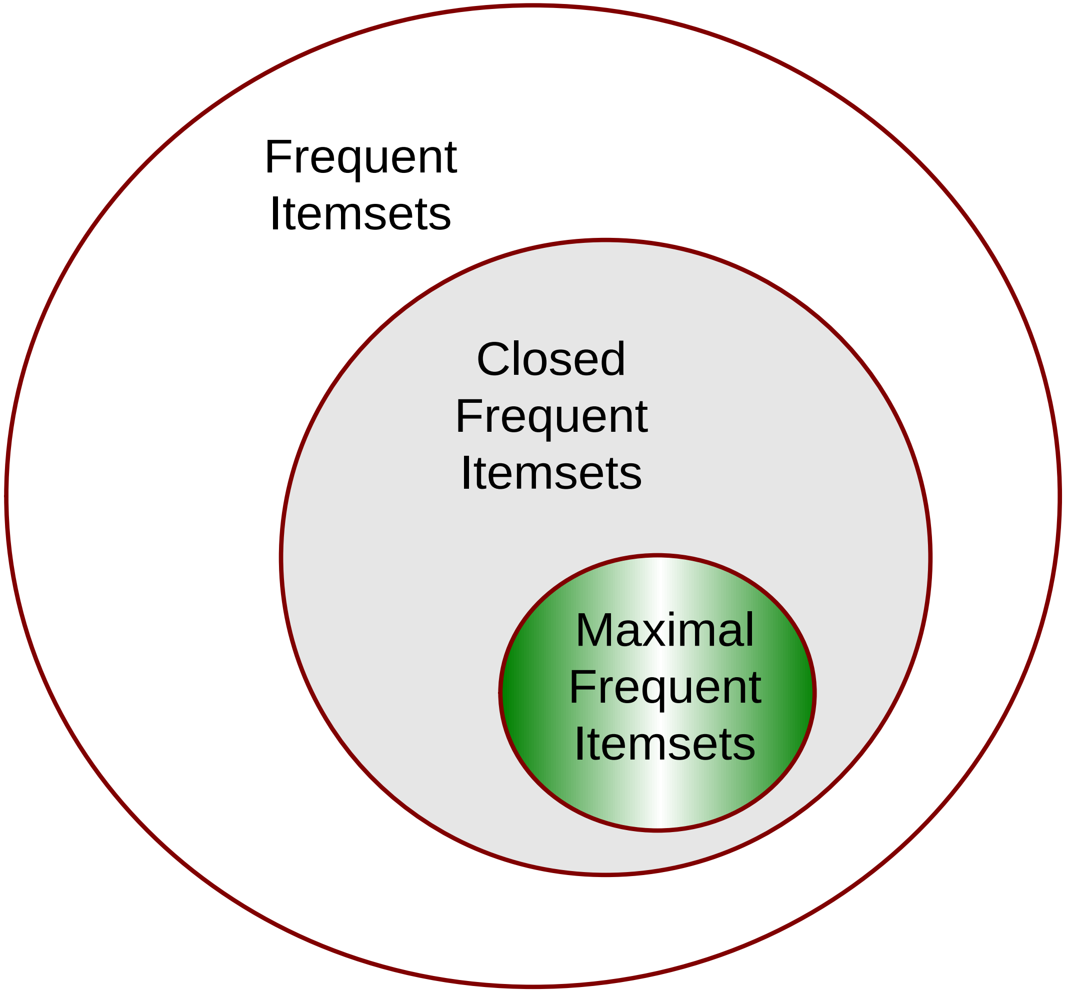
TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Maximal vs Closed Frequent Itemsets



Maximal vs Closed Itemsets



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

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

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

Text mining: Pointwise Mutual Information

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

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

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

Rare co-occurrences are deemed more interesting

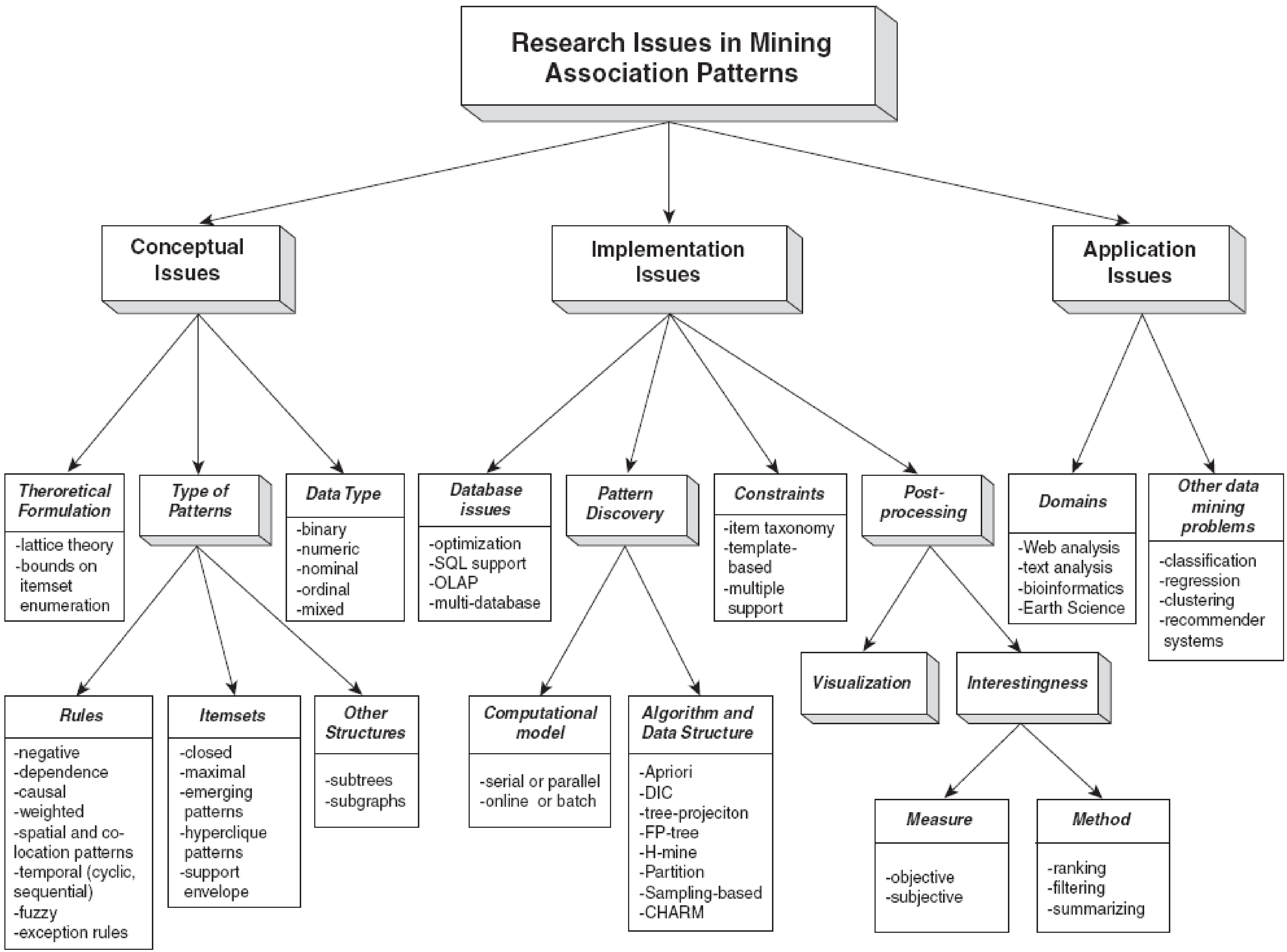


Figure 6.31. A summary of the various research activities in association analysis.