Data Mining

Chapter 5

Association Analysis: Basic Concepts

Market Basket Transaction

- Many business enterprises accumulate large quantities of data from their day-to-day operations. For example, huge amounts of customer purchase data are collected daily at the checkout counters of grocery stores. Such data, is commonly known as **market basket transactions**.
- Each row in this table corresponds to a transaction, which contains a unique identifier labeled TID and a set of items bought by a given customer. Retailers are interested in analyzing the data to learn about the purchasing behavior of their customers.

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},

Implication means co-occurrence, not causality!

Issues

- There are two key issues that need to be addressed when applying association analysis to market basket data.
- First, discovering patterns from a large transaction data set can be computationally expensive.
- Second, some of the discovered patterns are potentially spurious because they may happen simply by chance.

Binary Representation

- Market basket data can be represented in a binary format where each row corresponds to a transaction and each column corresponds to an item.
- An item can be treated as a binary variable whose value is one if the item is present in a transaction and zero otherwise. Because the presence of an item in a transaction is often considered more important than its absence, an item is an asymmetric binary variable.

Table 6.2. A binary $0/1$ representation of market basket data.

TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Example

This representation is perhaps a very simplistic view of real market basket data because it ignores certain important aspects of the data such as the quantity of items sold or the price paid to purchase them

Definition

- **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 → Y, where X and Y are itemsets
  - Example:
    {Milk, Diaper} → {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:

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

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

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

Definition

- The transaction width is defined as the number of items present in a transaction.
- A transaction tj is said to contain an itemset X if X is a subset of tj .

Why use support

- Support is an important measure because a rule that has very low support may occur simply by chance. A low support rule is also likely to be uninteresting from a business perspective because it may not be profitable to promote items that customers seldom buy together. For these reasons, support is often used to eliminate uninteresting rules.

Why use confidence

- Confidence, on the other hand, measures the reliability of the inference made by a rule. For a given rule $X \rightarrow Y$, the higher the confidence, the more likely it is for Y to be present in transactions that contain X. Confidence also provides an estimate of the conditional probability of Y given X.

Caution

- Association analysis results should be interpreted with caution. The inference made by an association rule does not necessarily imply causality. Instead, it suggests a strong co-occurrence relationship between items in the antecedent and consequent of the rule. Causality, on the other hand, requires knowledge about the causal and effect attributes in the data and typically involves relationships occurring over time (e.g., ozone depletion leads to global warming).

Association Rule Mining Task

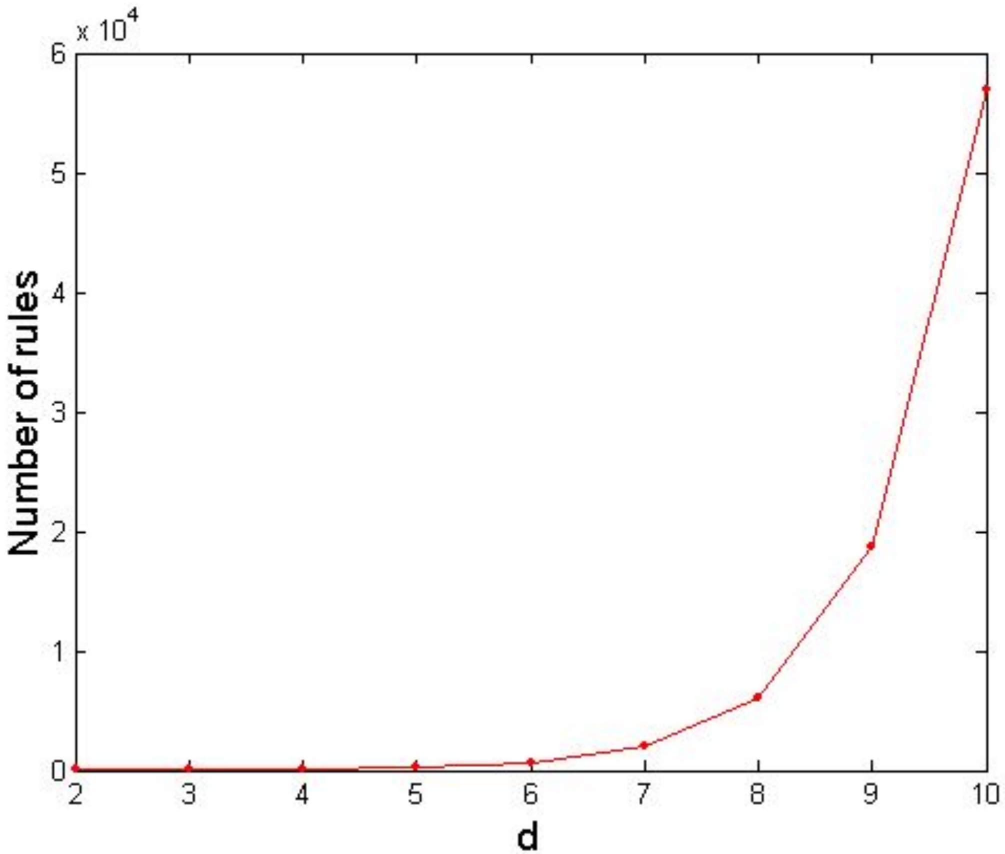
- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- 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!

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

Example

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

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

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)

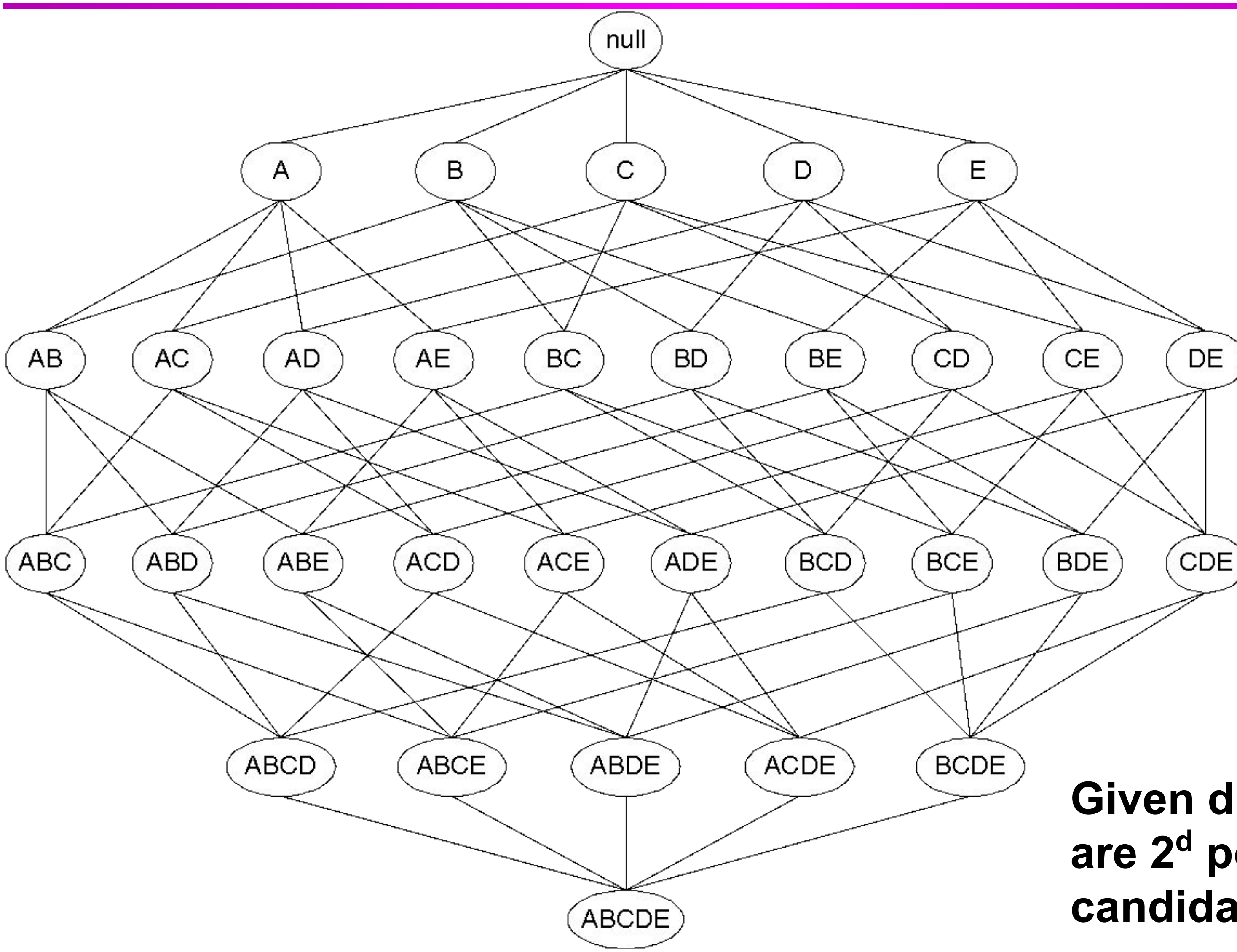
Observations:

- 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
- Thus, we may decouple the support and confidence requirements

Mining Association Rules

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

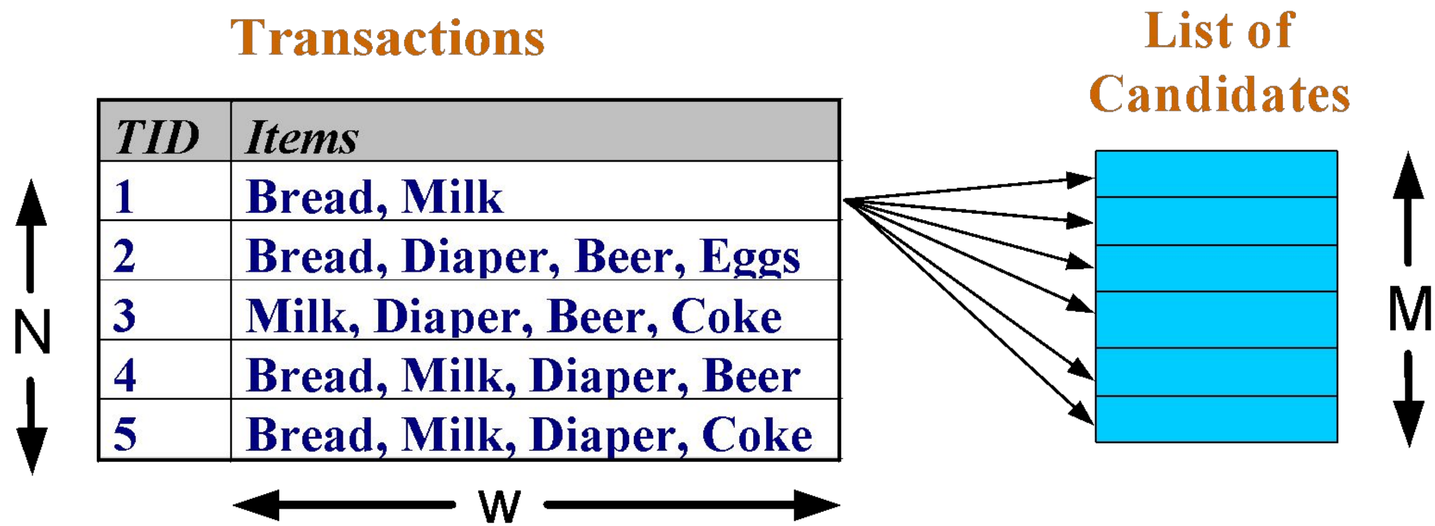
Frequent Itemset Generation



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

Frequent Itemset Generation

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



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

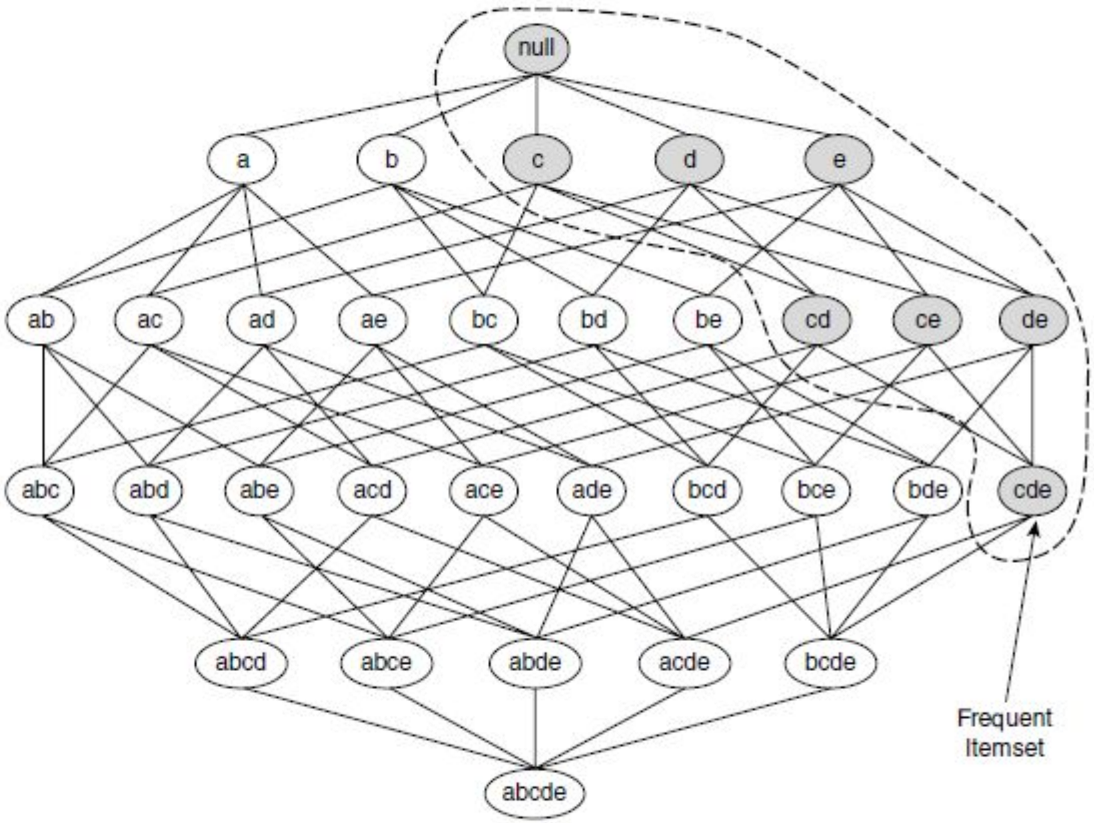
  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Frequent Itemset Generation Strategies

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

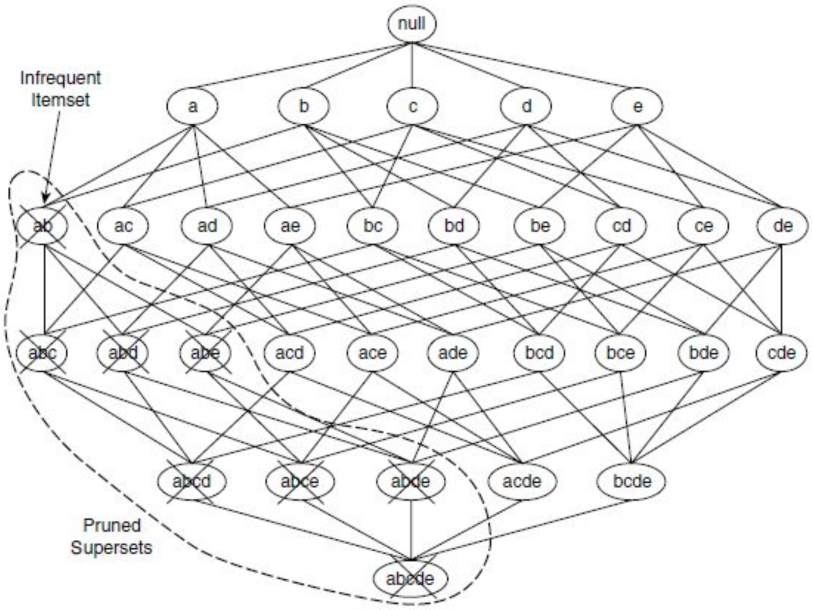
Reducing Number of Candidates

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



Reducing Number of Candidates

- Apriori principle:
- Conversely, if an itemset such as {a, b} is infrequent, then all of its supersets must be infrequent too.



Reducing Number of Candidates

- Apriori principle:
- This strategy of trimming the exponential search space based on the support measure is known as support-based pruning.
- Such a pruning strategy is made possible by a key property of the support measure, namely, that the support for an itemset never exceeds the support for its subsets. This property is also known as the anti-monotone property

Monotone and Antimonotone Property

- A measure f is monotone (or upward closed) if

$\forall X, Y \in J : (X \subseteq Y) \rightarrow f(X) \leq f(Y)$

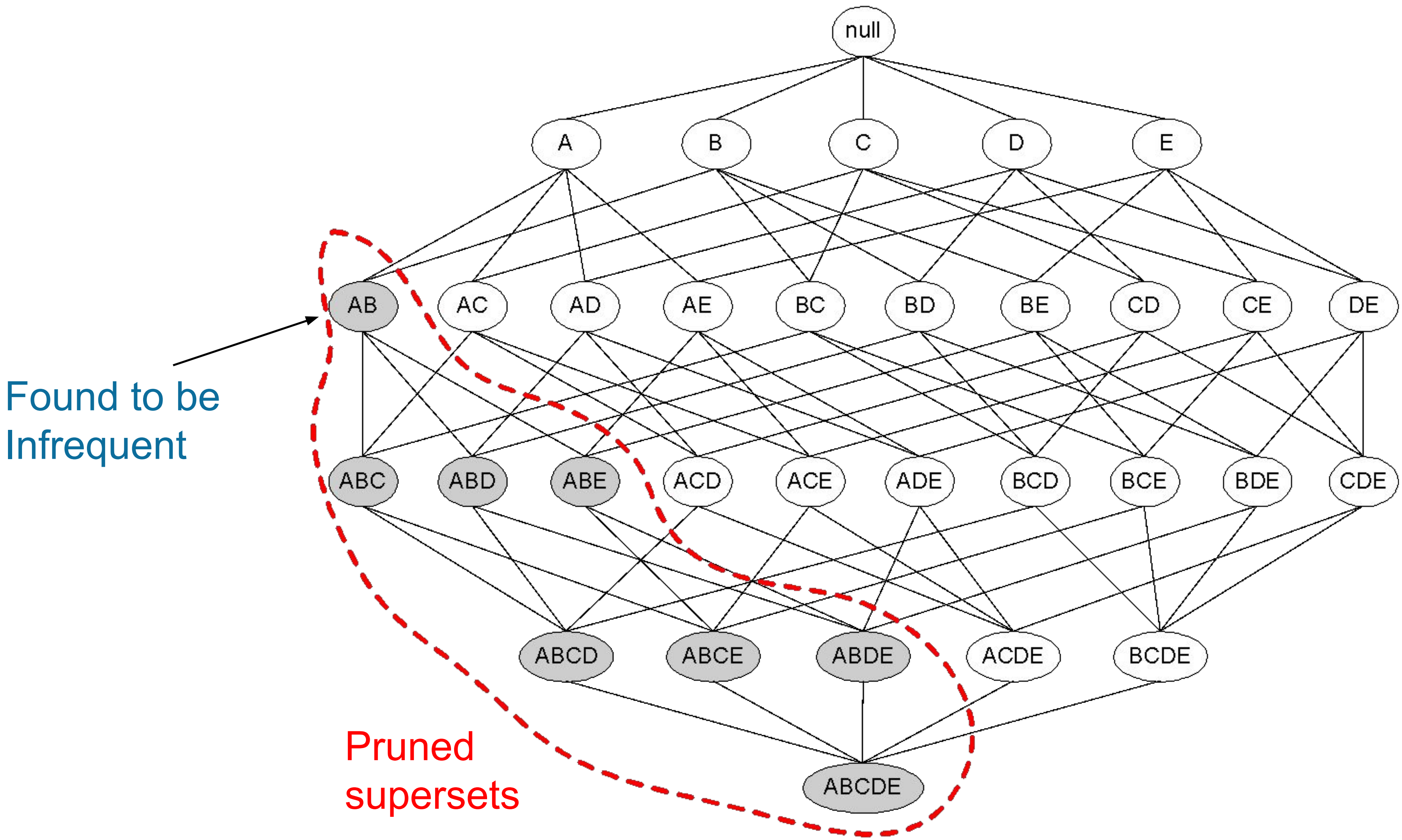
- which means that if X is a subset of Y, then f(X) must not exceed f(Y).
- f is anti-monotone (or downward closed) if
- $\forall X, Y \in J : (X \subseteq Y) \rightarrow f(Y) \leq f(X)$

which means that if X is a subset of Y , then f(Y) must not exceed f(X).

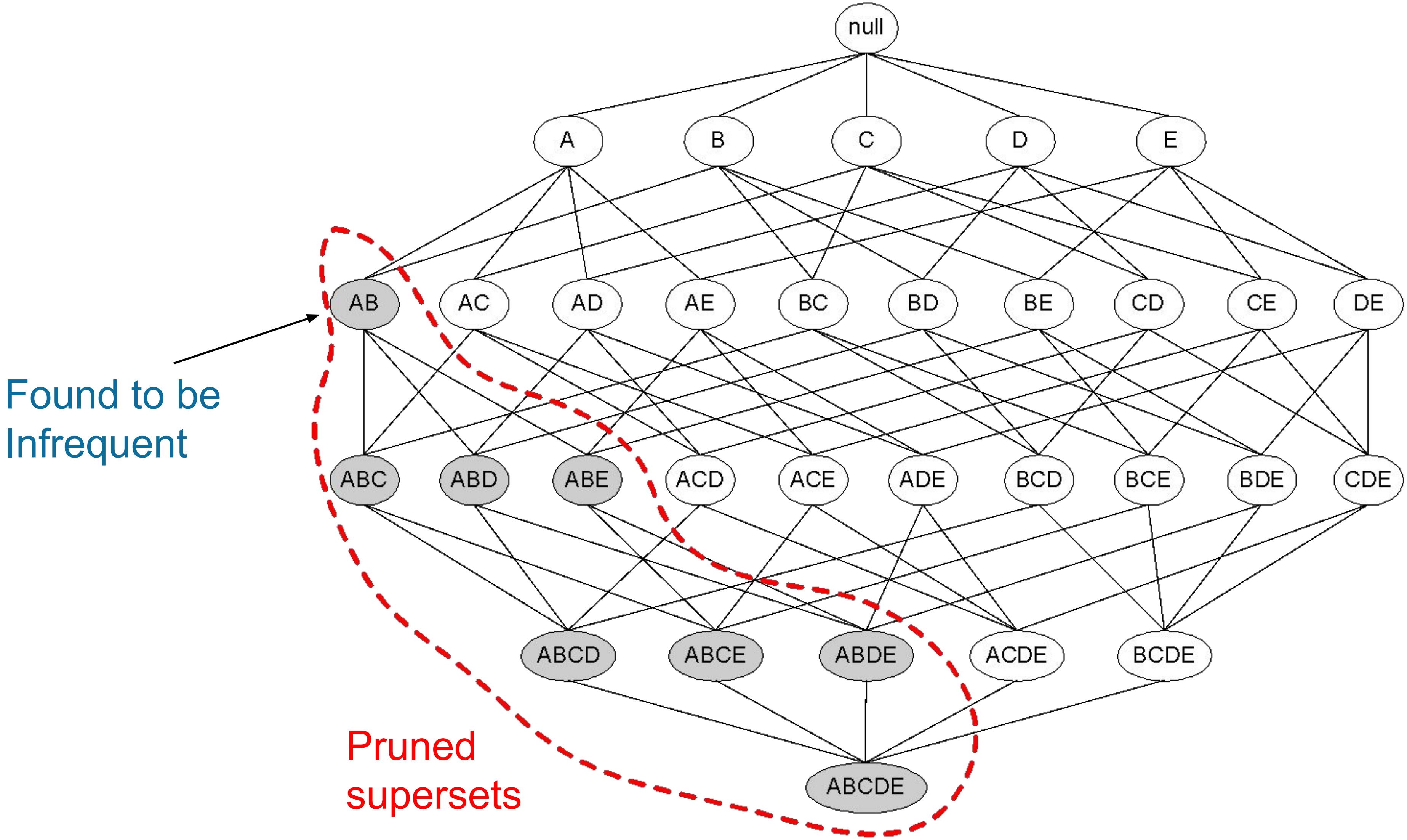
Antimonotone

- Any measure that possesses an anti-monotone property can be incorporated directly into the mining algorithm to effectively prune the exponential search space of candidate itemsets

Illustrating Apriori Principle



Illustrating Apriori Principle



Illustrating Apriori Principle

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

Items (1-itemsets)

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

Minimum Support = 3

If every subset is considered,

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

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 4 = 16$

Illustrating Apriori Principle

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

Items (1-itemsets)

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

Pairs (2-itemsets)

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

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,

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

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 4 = 16$

Illustrating Apriori Principle

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

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

Items (1-itemsets)

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

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,
$^6C_1 + {}^6C_2 + {}^6C_3$
$6 + 15 + 20 = 41$
With support-based pruning,
$6 + 6 + 4 = 16$

Illustrating Apriori Principle

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

Items (1-itemsets)

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

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

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

Triplets (3-itemsets)

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

Minimum Support = 3

If every subset is considered,

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

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$

Illustrating Apriori Principle

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

Items (1-itemsets)

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

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

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

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{ Beer, Diaper, Milk}	2
{ Beer,Bread, Diaper}	2
{Bread, Diaper, Milk}	3
{Beer, Bread, Milk}	1

If every subset is considered,

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

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 4 = 16$

Illustrating Apriori Principle

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

Items (1-itemsets)

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

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

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

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{ Beer, Diaper, Milk}	2
{ Beer,Bread, Diaper}	2
{Bread, Diaper, Milk}	3
{Beer, Bread, Milk}	1

If every subset is considered,

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

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 4 = 16$

$6 + 6 + 1 = 13$

Apriori Algorithm

Algorithm 6.1 Frequent itemset generation of the *Apriori* algorithm.

$k = 1$.
$F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup \}$. {Find all frequent 1-itemsets}
repeat
  $k = k + 1$.
  $C_k = \text{apriori-gen}(F_{k-1})$. {Generate candidate itemsets}
  for each transaction $t \in T$ **do**
    $C_t = \text{subset}(C_k, t)$. {Identify all candidates that belong to t}
    for each candidate itemset $c \in C_t$ **do**
      $\sigma(c) = \sigma(c) + 1$. {Increment support count}
    end for
  end for
  $F_k = \{ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup \}$. {Extract the frequent k-itemsets}
until $F_k = \emptyset$
Result $= \bigcup F_k$.

Characteristics

- The frequent itemset generation part of the *Apriori* algorithm has two important characteristics.
- First, it is a **level-wise** algorithm; i.e., it traverses the itemset lattice one level at a time, from frequent 1-itemsets to the maximum size of frequent itemsets.
- Second, it employs a **generate-and-test** strategy for finding frequent itemsets. At each iteration, new candidate itemsets are generated from the frequent itemsets found in the previous iteration. The support for each candidate is then counted and tested against the *minsup* threshold.
- The total number of iterations needed by the algorithm is $k\max+1$, where $k\max$ is the maximum size of the frequent itemsets.

Candidate Generation and Prunning

- **Candidate Generation.** This operation generates new candidate k *itemsets* based on the frequent $(k-1)$-itemsets found in the previous iteration.
- **Candidate Pruning.** This operation eliminates some of the candidate k-itemsets using the support-based pruning strategy.

Requirements (one)

- It should avoid generating too many unnecessary candidates. A candidate itemset is unnecessary if at least one of its subsets is infrequent.
- Such a candidate is guaranteed to be infrequent according to the antimonotone property of support.

Requirement (two)

- It must ensure that the candidate set is complete, i.e., no frequent itemsets are left out by the candidate generation procedure. To ensure completeness, the set of candidate itemsets must subsume the set of all frequent itemsets, i.e., $\forall k : F_k \subseteq C_k$.

Requirement (three)

- It should not generate the same candidate itemset more than once. For example, the candidate itemset {a, b, c, d} can be generated in many ways—by merging {a, b, c} with {d}, {b, d} with {a, c}, {c} with {a, b, d}, etc. Generation of duplicate candidates leads to wasted computations and thus should be avoided for efficiency reasons.

Candidate Generation: Brute-force method

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

Items

Item
Beer
Bread
Cola
Diapers
Eggs
Milk

→

Candidate Generation

Itemset
{Beer, Bread, Cola}
{Beer, Bread, Diapers}
{Beer, Bread, Eggs}
{Beer, Bread, Milk}
{Beer, Cola, Diapers}
{Beer, Cola, Eggs}
{Beer, Cola, Milk}
{Beer, Diapers, Eggs}
{Beer, Diapers, Milk}
{Beer, Eggs, Milk}
{Bread, Cola, Diapers}
{Bread, Cola, Eggs}
{Bread, Cola, Milk}
{Bread, Diapers, Eggs}
{Bread, Diapers, Milk}
{Bread, Eggs, Milk}
{Cola, Diapers, Eggs}
{Cola, Diapers, Milk}
{Cola, Eggs, Milk}
{Diapers, Eggs, Milk}

→

Candidate Pruning

Itemset
{Bread, Diapers, Milk}

Frequent 2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

Figure 5.6. A brute-force method for generating candidate 3-itemsets.

Candidate Generation: Merge Fk-1 and F1 itemsets

Frequent 2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

Frequent 1-itemset

Item
Beer
Bread
Diapers
Milk

Candidate Generation

Itemset
{Beer, Bread, Diapers}
{Beer, Diapers, Milk}
{Bread, Diapers, Milk}
{Beer, Bread, Milk}

Candidate Pruning

Itemset
{Bread, Diapers, Milk}

Figure 5.7. Generating and pruning candidate k-itemsets by merging a frequent $(k-1)$-itemset with a frequent item. Note that some of the candidates are unnecessary because their subsets are infrequent.

Candidate Generation: $F_{k-1} \times F_{k-1}$ Method

- Merge two frequent (k-1)-itemsets if their first (k-2) items are identical

- F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE}
  - Merge(**AB**C, **AB**D) = **AB**CD
  - Merge(**AB**C, **AB**E) = **AB**CE
  - Merge(**AB**D, **AB**E) = **AB**DE

  - Do not merge(**A**BD,**A**CD) because they share only prefix of length 1 instead of length 2

Candidate Pruning

- Let F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE} be the set of frequent 3-itemsets

- L_4 = {ABCD,ABCE,ABDE} is the set of candidate 4-itemsets generated (from previous slide)

- Candidate pruning
  - Prune ABCE because ACE and BCE are infrequent
  - Prune ABDE because ADE is infrequent

- After candidate pruning: L_4 = {ABCD}

Candidate Generation: Fk-1 x Fk-1 Method

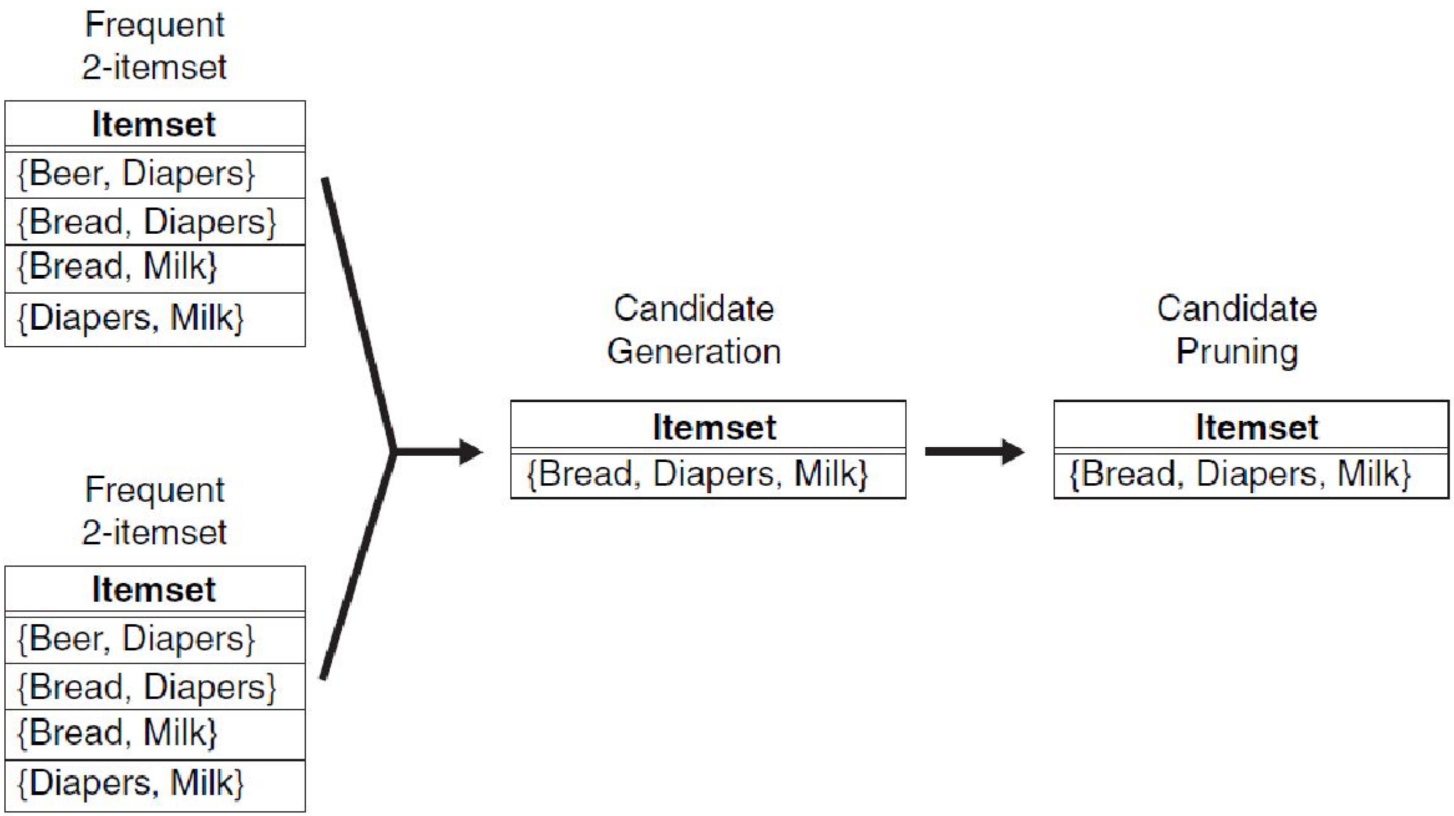


Figure 5.8. Generating and pruning candidate k-itemsets by merging pairs of frequent $(k-1)$-itemsets.

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, Diaper, Milk}	**2**

If every subset is considered,

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

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 1 = 13$

Use of F_{k-1}xF_{k-1} method for candidate generation results in only one 3-itemset. This is eliminated after the support counting step.

Alternate F_{k-1} x F_{k-1} Method

- Merge two frequent (k-1)-itemsets if the last (k-2) items of the first one is identical to the first (k-2) items of the second.

- F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE}
  - Merge(A**BC**, **BC**D) = A**BC**D
  - Merge(A**BD**, **BD**E) = A**BD**E
  - Merge(A**CD**, **CD**E) = A**CD**E
  - Merge(B**CD**, **CD**E) = B**CD**E

Candidate Pruning for Alternate $F_{k-1} \times F_{k-1}$ Method

- Let F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE} be the set of frequent 3-itemsets

- L_4 = {ABCD,ABDE,ACDE,BCDE} is the set of candidate 4-itemsets generated (from previous slide)
- Candidate pruning
  - Prune ABDE because ADE is infrequent
  - Prune ACDE because ACE and ADE are infrequent
  - Prune BCDE because BCE
- After candidate pruning: L_4 = {ABCD}

Support Counting of Candidate Itemsets

- Scan the database of transactions to determine the support of each candidate itemset
  - Must match every candidate itemset against every transaction, which is an expensive operation

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

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

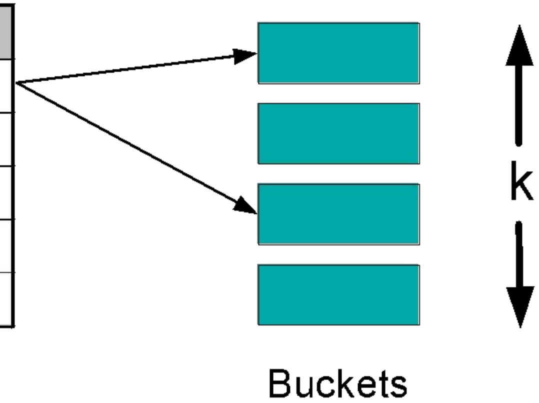
Support Counting of Candidate Itemsets

- To reduce number of comparisons, store the candidate itemsets 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

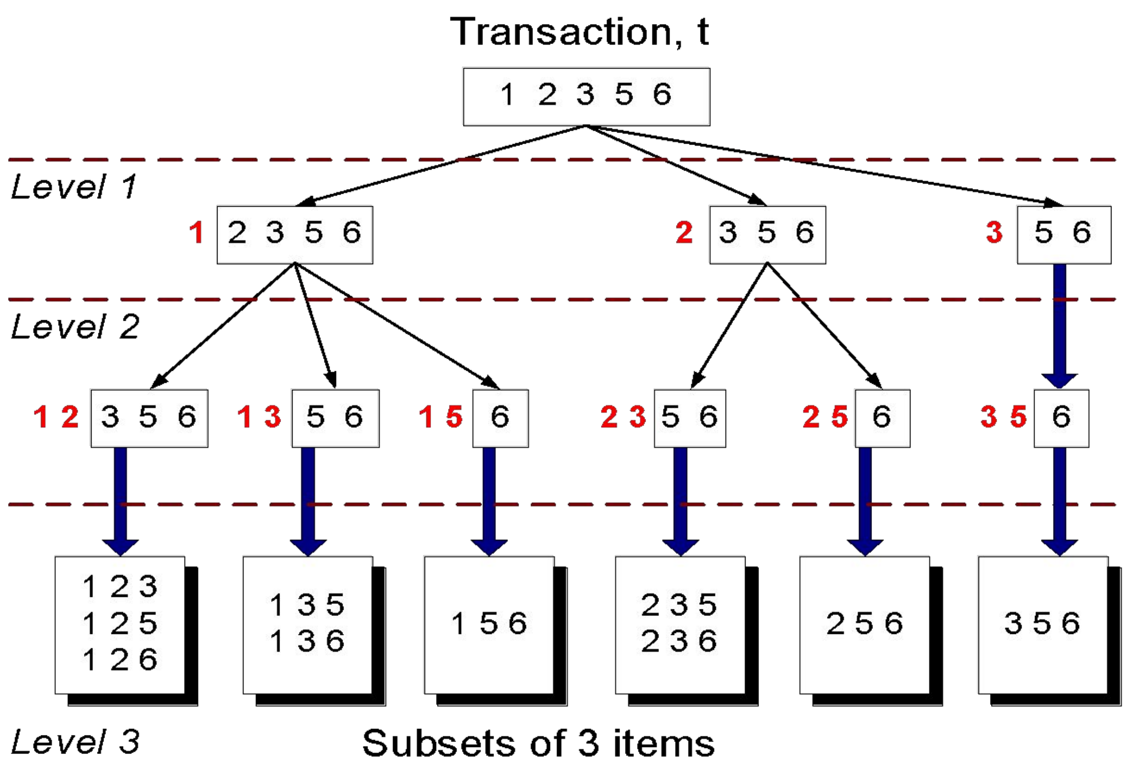


Support Counting: An Example

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}

How many of these itemsets are supported by transaction (1,2,3,5,6)?



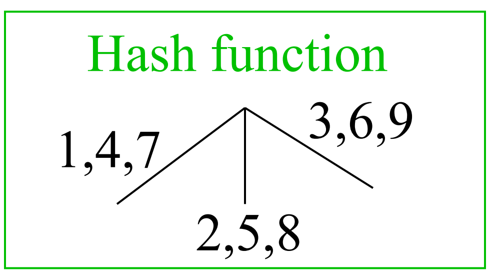
Support Counting Using a Hash Tree

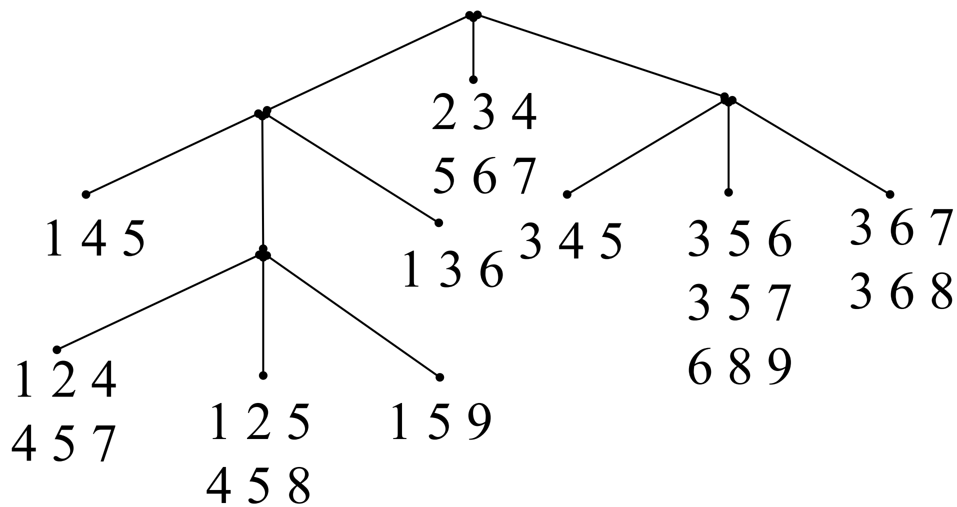
Suppose you have 15 candidate itemsets of length 3:

{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}

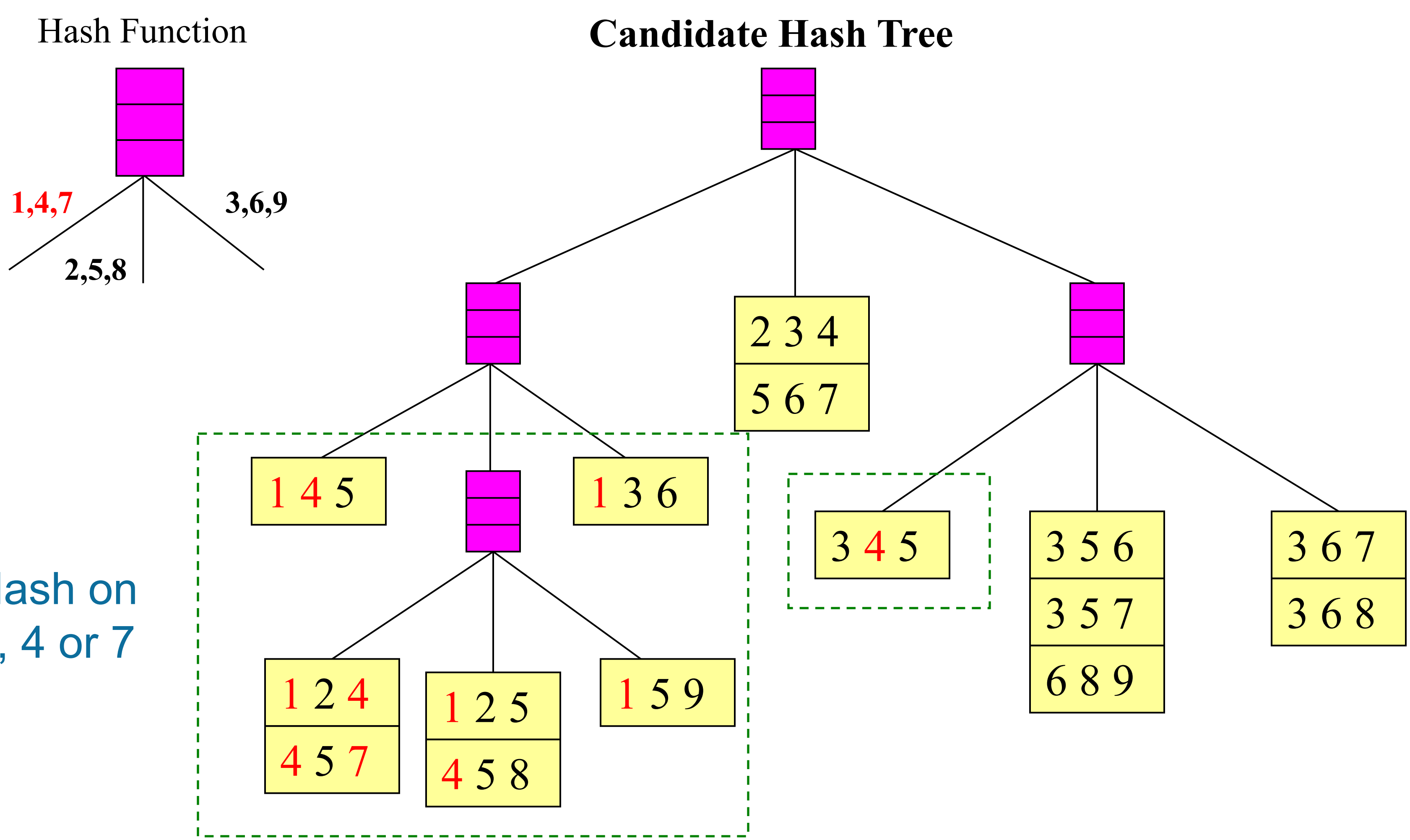
You need:

- 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)



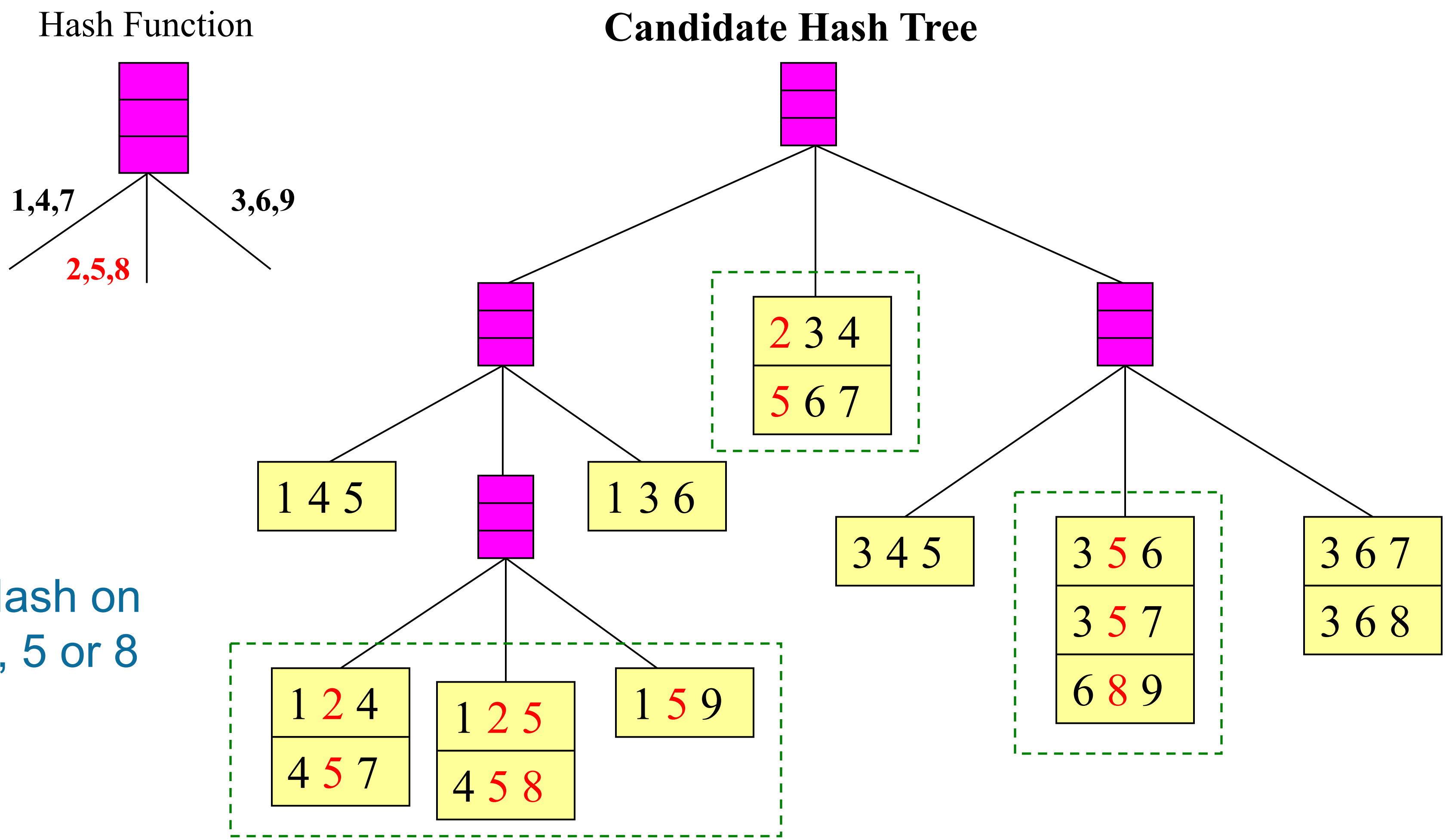


Support Counting Using a Hash Tree



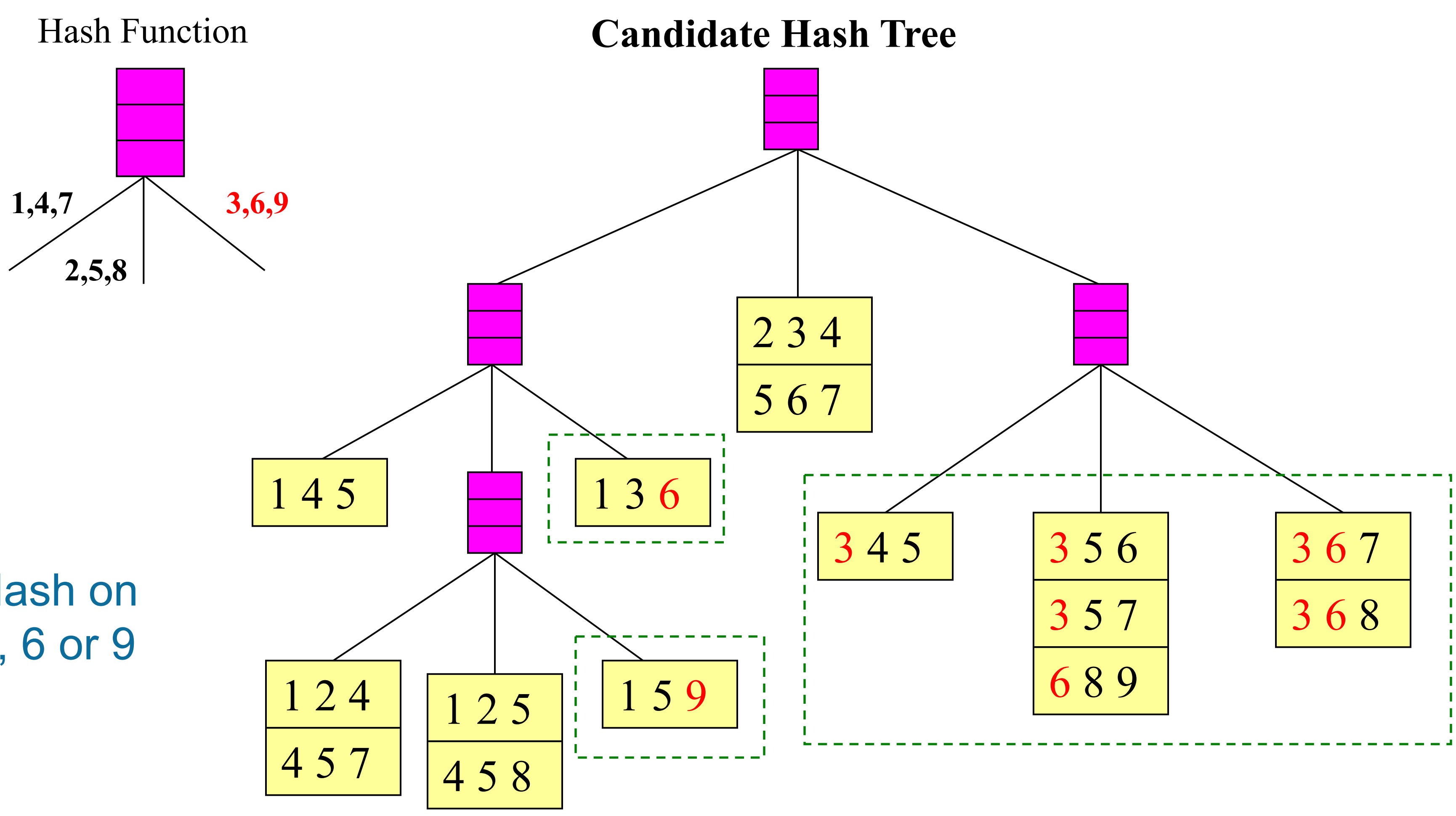
Hash on
1, 4 or 7

Support Counting Using a Hash Tree



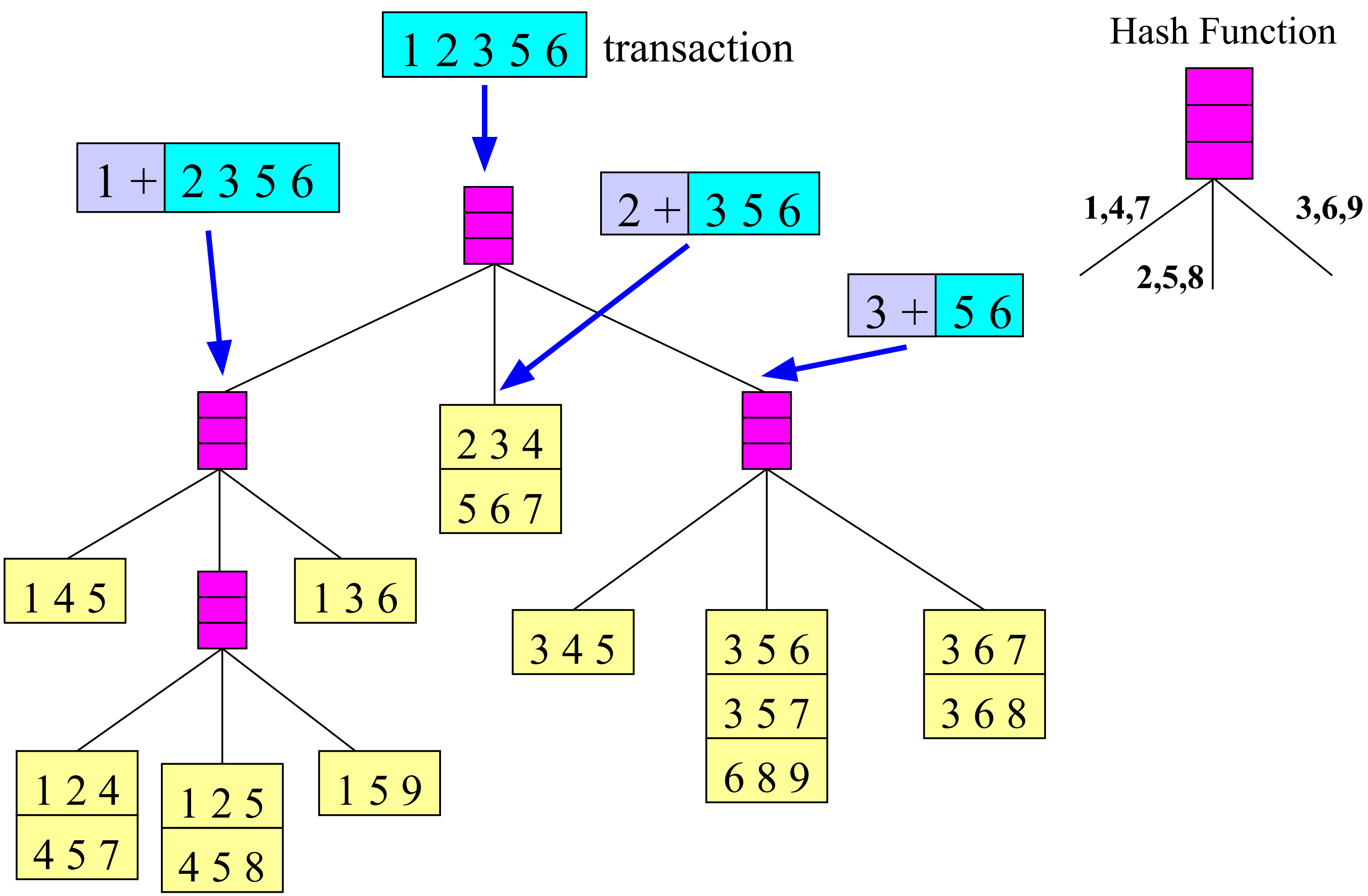
Hash on
2, 5 or 8

Support Counting Using a Hash Tree

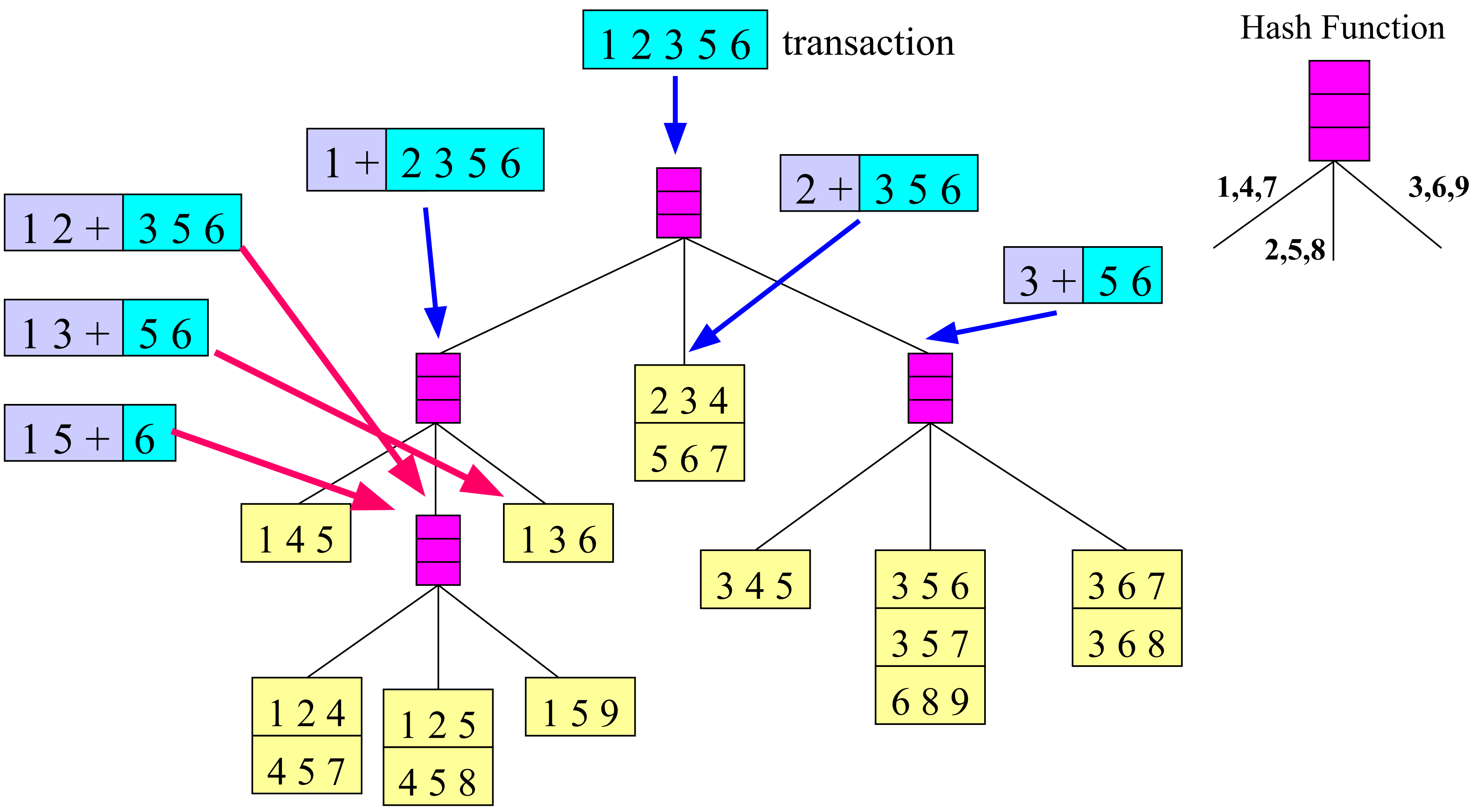


Hash on
3, 6 or 9

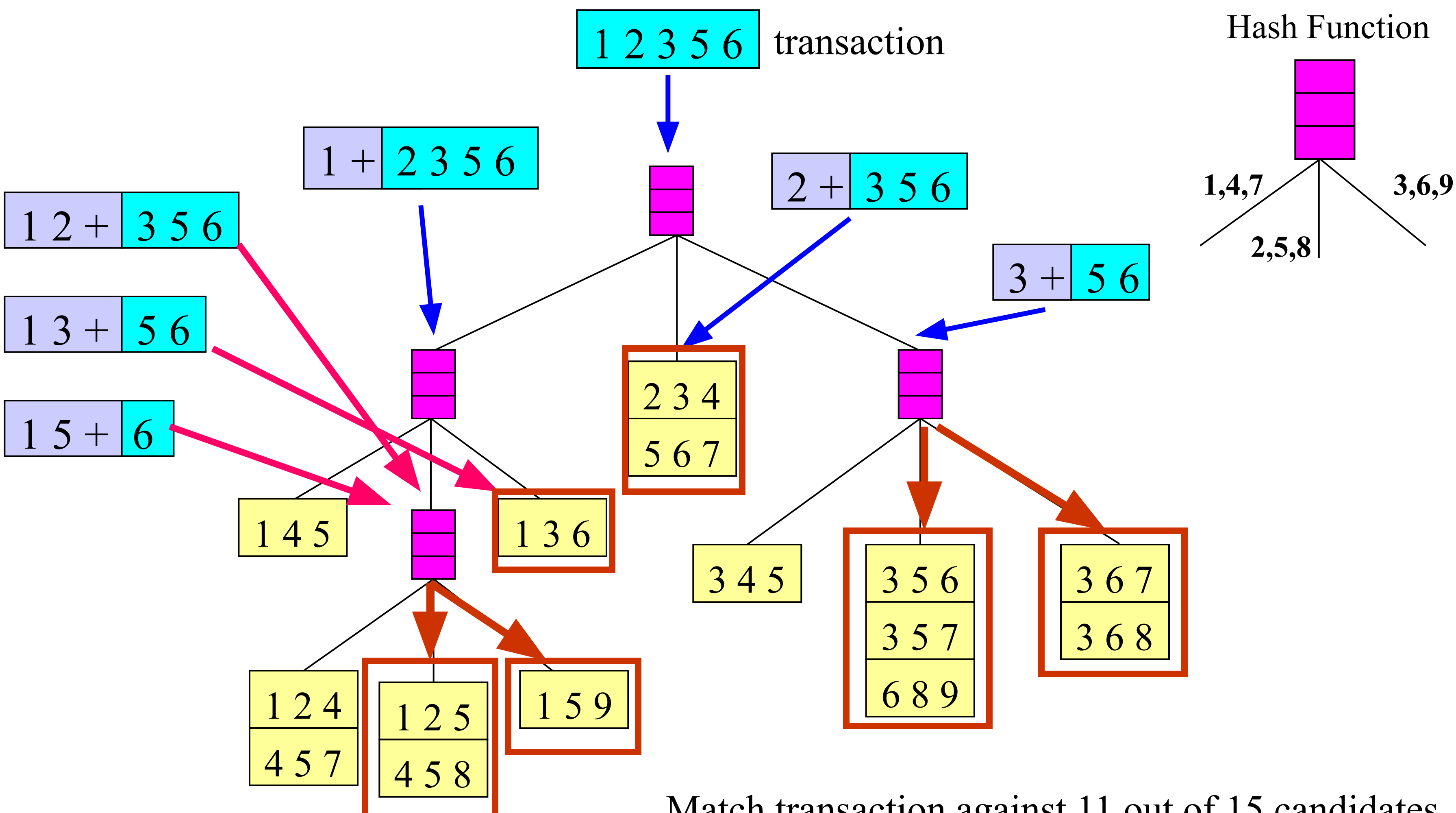
Support Counting Using a Hash Tree



Support Counting Using a Hash Tree



Support Counting Using a Hash Tree



Match transaction against 11 out of 15 candidates

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 →D, ABD →C, ACD →B, BCD →A,
    A →BCD, B →ACD, C →ABD, D →ABC
    AB →CD, AC → BD, AD → BC, BC →AD,
    BD →AC, CD →AB,

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

Rule Generation

- 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., Suppose {A,B,C,D} is a frequent 4-itemset:

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

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

Rule Generation for Apriori Algorithm

Lattice of rules