Introduction to Big Data Science

11th Period

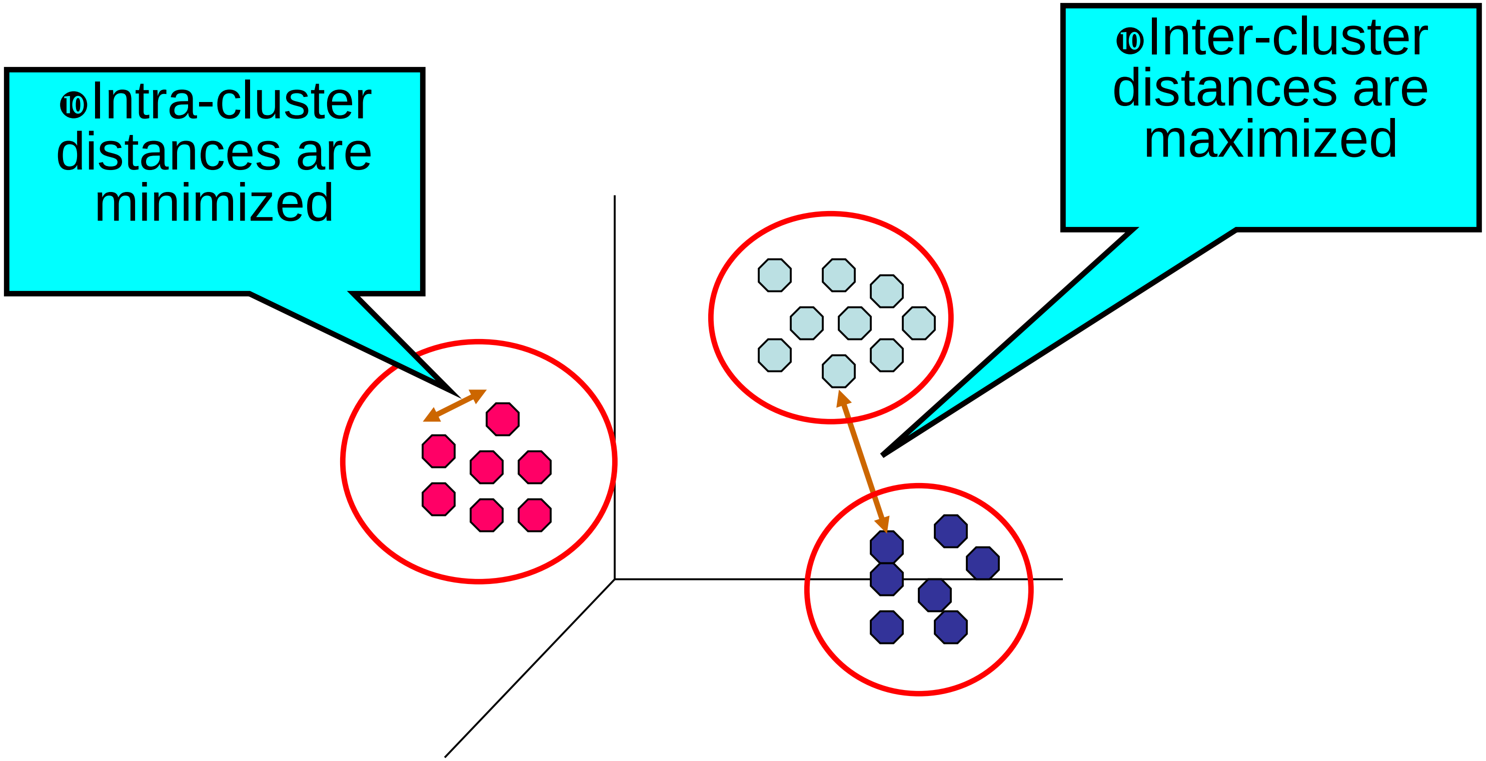
Essence in Data Mining
- Clustering and Association -

(SEC. I)

CLUSTERING ANALYSIS

What is Cluster Analysis?

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



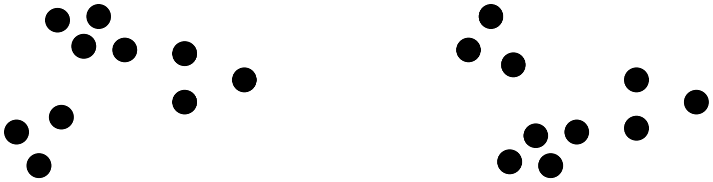
❿Intra-cluster distances are minimized

❿Inter-cluster distances are maximized

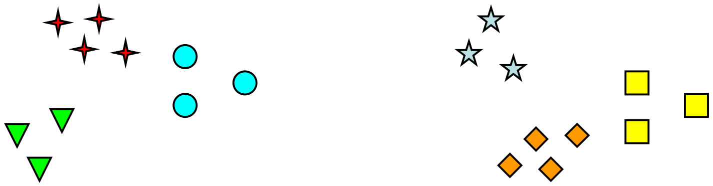
What is not Cluster Analysis?

- Supervised classification
  - Have class label information
- Simple segmentation
  - Dividing students into different registration groups alphabetically, by last name
- Results of a query
  - Groupings are a result of an external specification
- Graph partitioning
  - Some mutual relevance and synergy, but areas are not identical

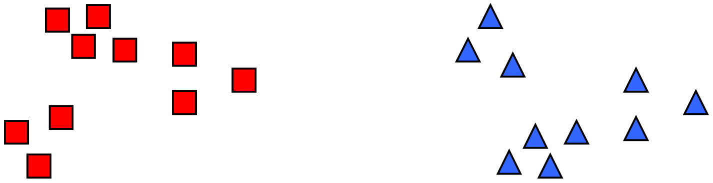
Notion of a Cluster can be Ambiguous



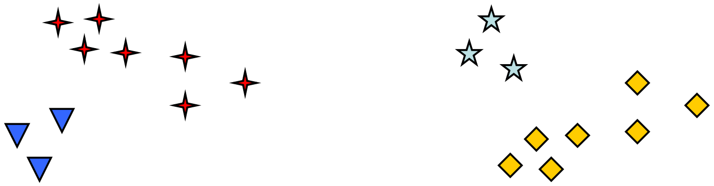
- How many clusters?



- Six Clusters



- Two Clusters

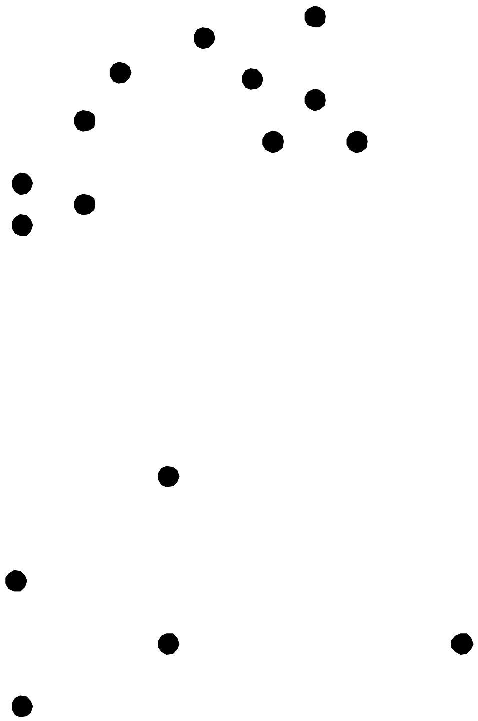


- Four Clusters

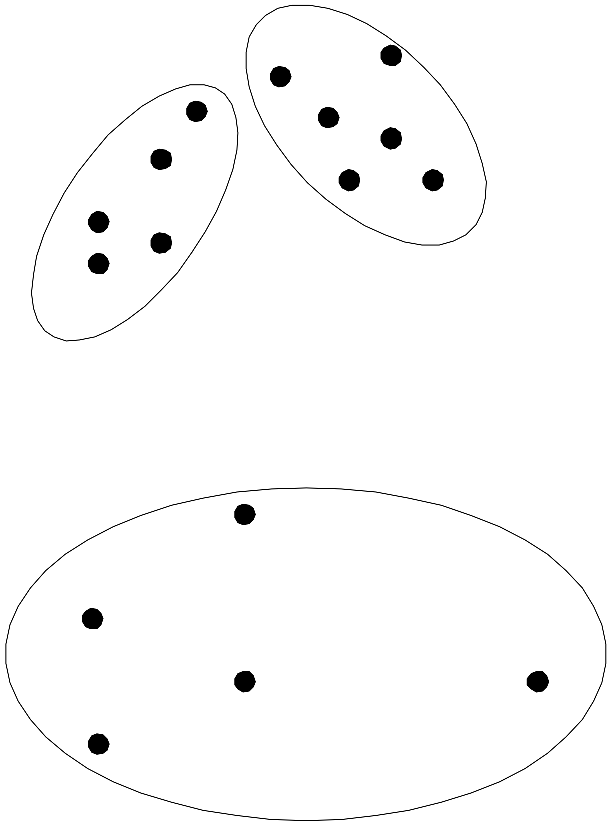
Types of Clustering

- A clustering is a set of clusters
- Important distinction between hierarchical and partitional sets of clusters
- Partitional Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

Partition Clustering

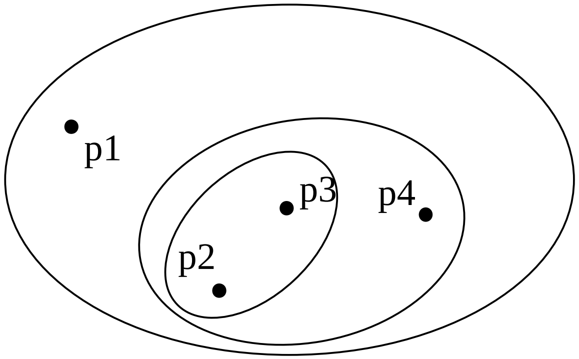


❿Original Points

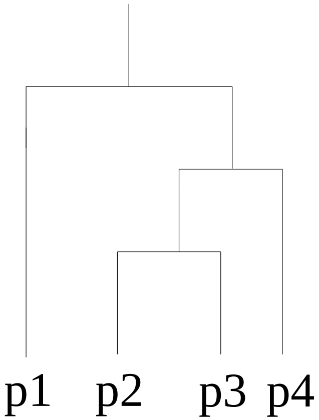


❿A Partitional Clustering

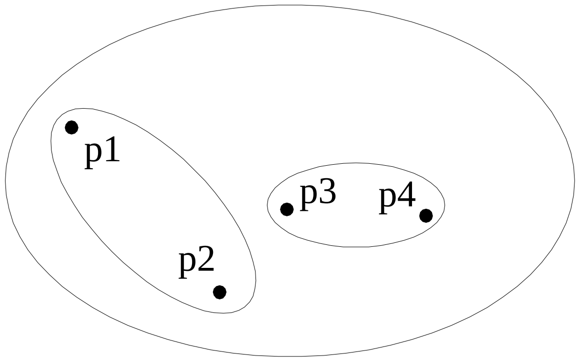
Hierarchical Clustering



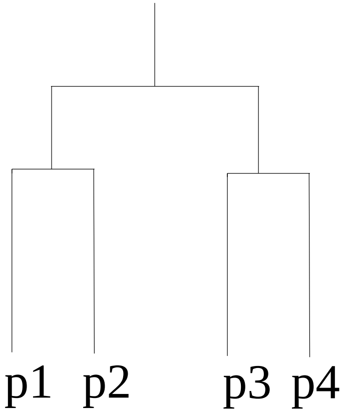
❿Traditional Hierarchical Clustering



❿Traditional Dendrogram



❿Non-traditional Hierarchical Clustering



❿Non-traditional Dendrogram

Other Distinctions Between Sets of Clusters

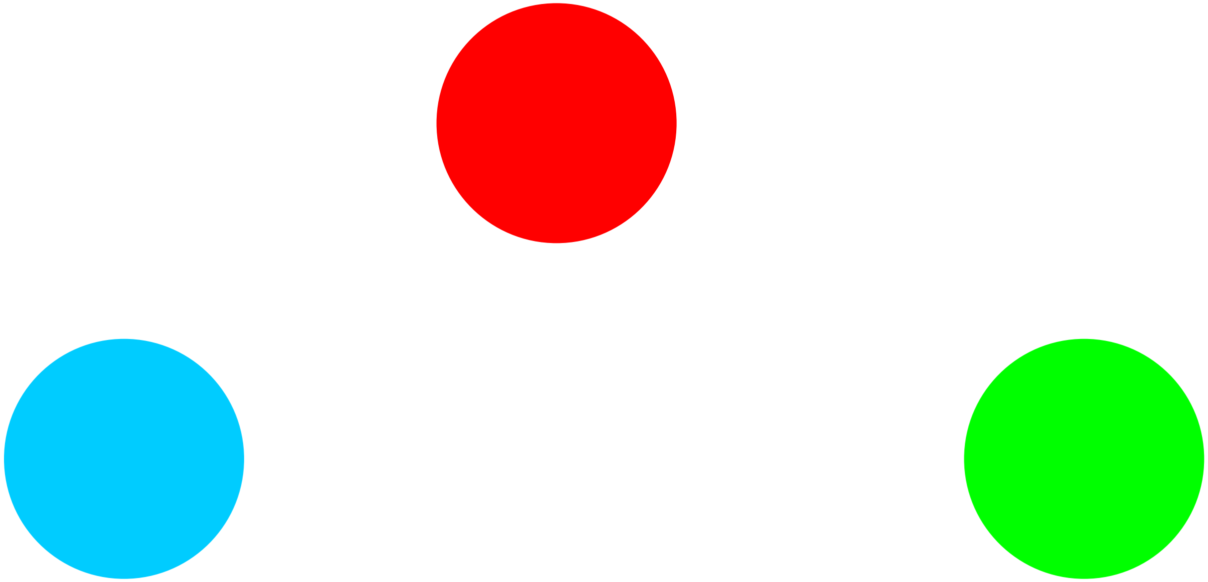
- Exclusive versus non-exclusive
  - In non-exclusive clusterings, points may belong to multiple clusters.
  - Can represent multiple classes or ‘border’ points
- Fuzzy versus non-fuzzy
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must sum to 1
  - Probabilistic clustering has similar characteristics
- Partial versus complete
  - In some cases, we only want to cluster some of the data
- Heterogeneous versus homogeneous
  - Cluster of widely different sizes, shapes, and densities

Types of Clusters

- Well-separated clusters
- Center-based clusters
- Contiguous clusters
- Density-based clusters
- Property or Conceptual
- Described by an Objective Function

Types of Clusters: Well-Separated

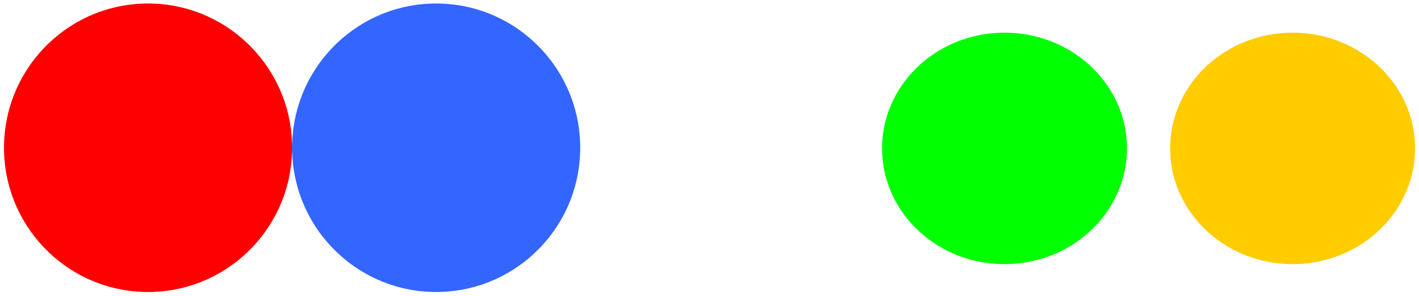
- Well-Separated Clusters:
  - A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.



❿3 well-separated clusters

Types of Clusters: Center-Based

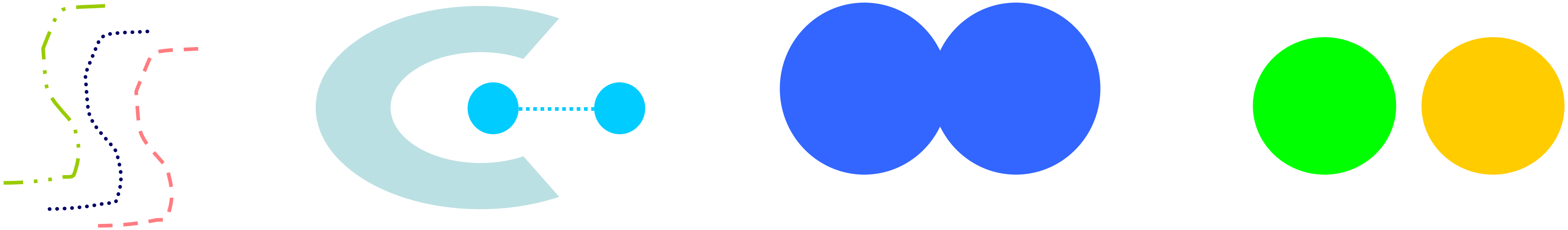
- Center-based
  - A cluster is a set of objects such that an object in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster
  - The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most "representative" point of a cluster



❿4 center-based clusters

Types of Clusters: Contiguity-Based

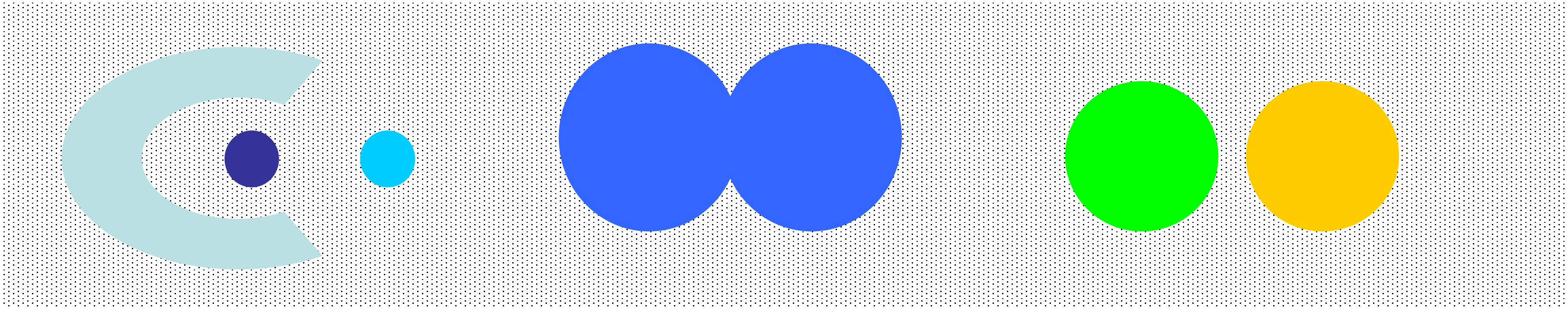
- Contiguous Cluster (Nearest neighbor or Transitive)
  - A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.



❿8 contiguous clusters

Types of Clusters: Density-Based

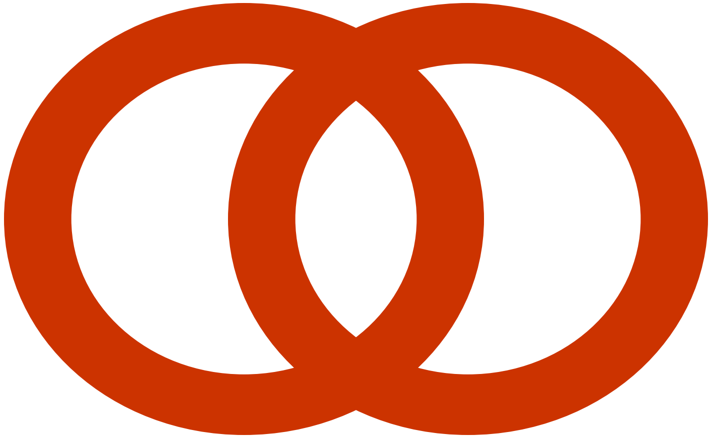
- Density-based
  - A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
  - Used when the clusters are irregular or intertwined, and when noise and outliers are present.



❿6 density-based clusters

Types of Clusters: Conceptual Clusters

- Shared Property or Conceptual Clusters
  - Finds clusters that share some common property or represent a particular concept.



❿2 Overlapping Circles

(SEC. II)

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} → {Beer},
    {Milk, Bread} → {Eggs,Coke},
    {Beer, Bread} → {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 → Y, where X and Y are itemsets
  - Example:
    {Milk, Diaper} → {Beer}

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

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

Example:

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

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

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

Association Rule Mining Task

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

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  ⇒ Computationally prohibitive!

Mining Association Rules

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

❿Example of Rules:

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

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

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

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

❿Observations:

- 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

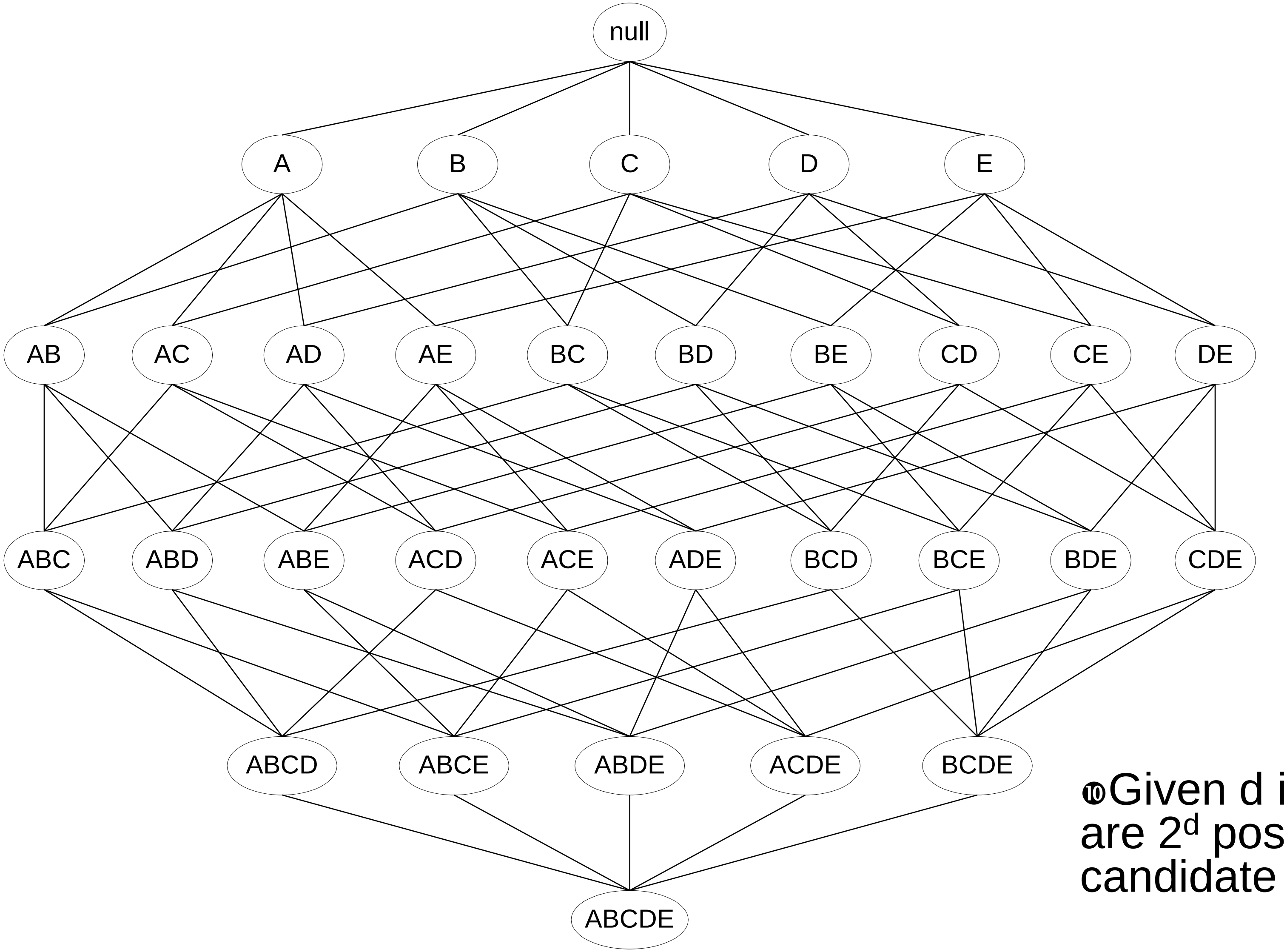
- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

- Frequent itemset generation is still computationally expensive

(SEC. III)

FREQUENT ITEMSET GENERATION

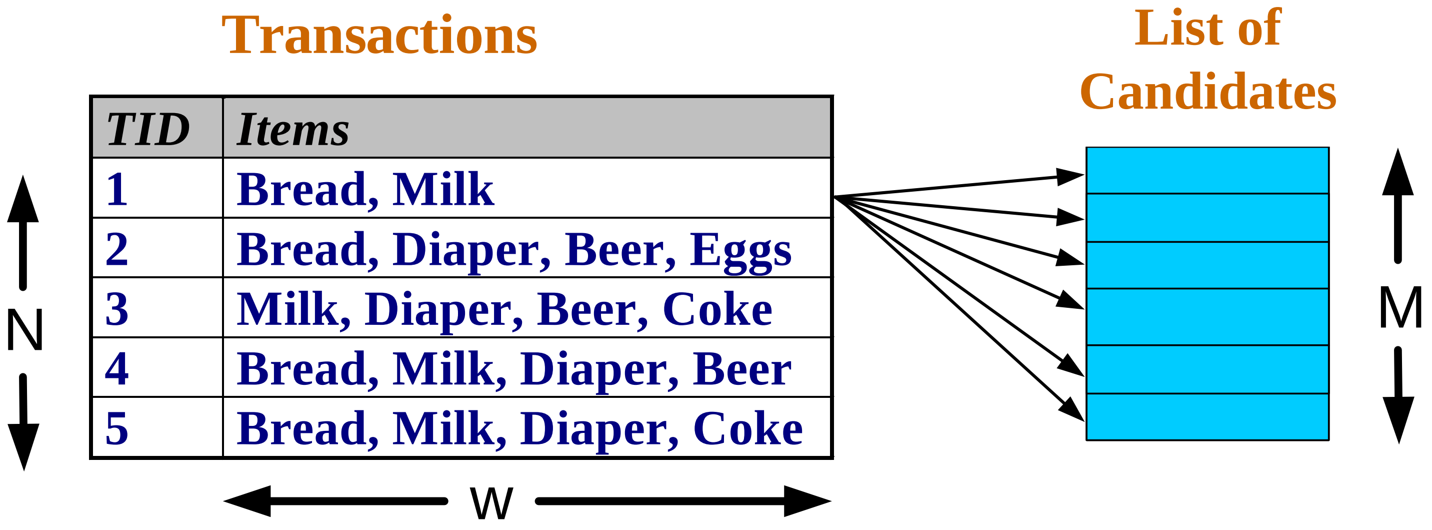
Frequent Itemset Generation



- 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

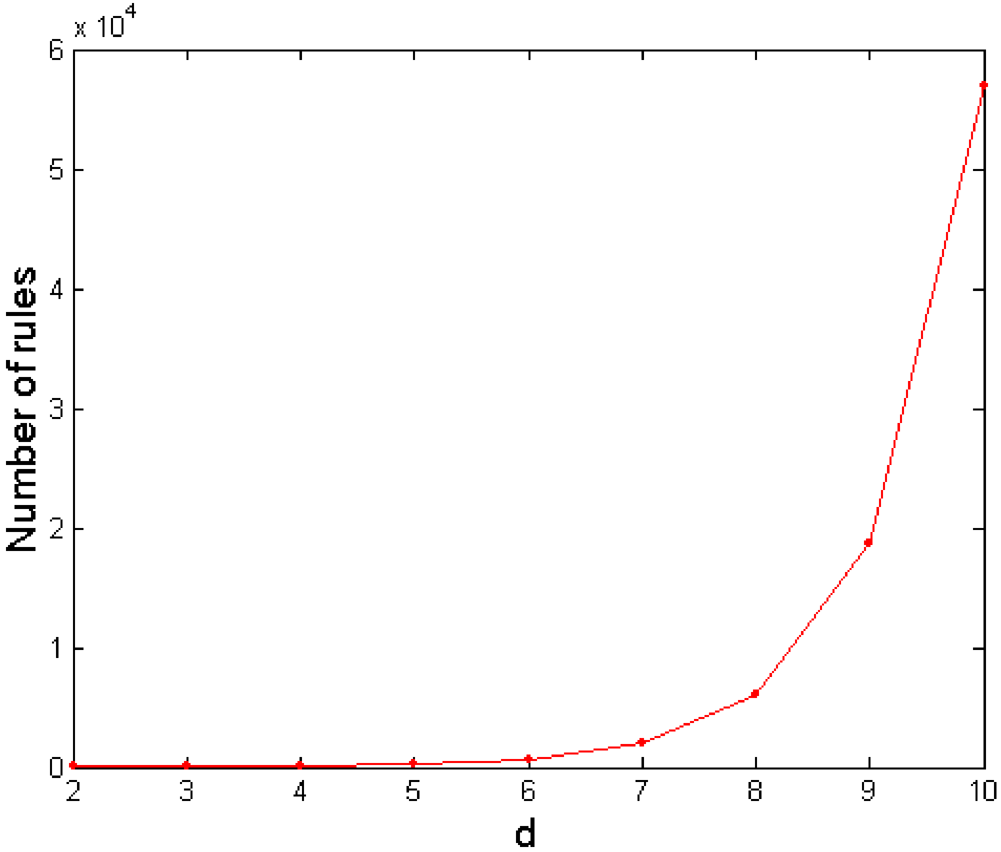
TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

List of Candidates

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

Computational Complexity

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



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

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

⑩If 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

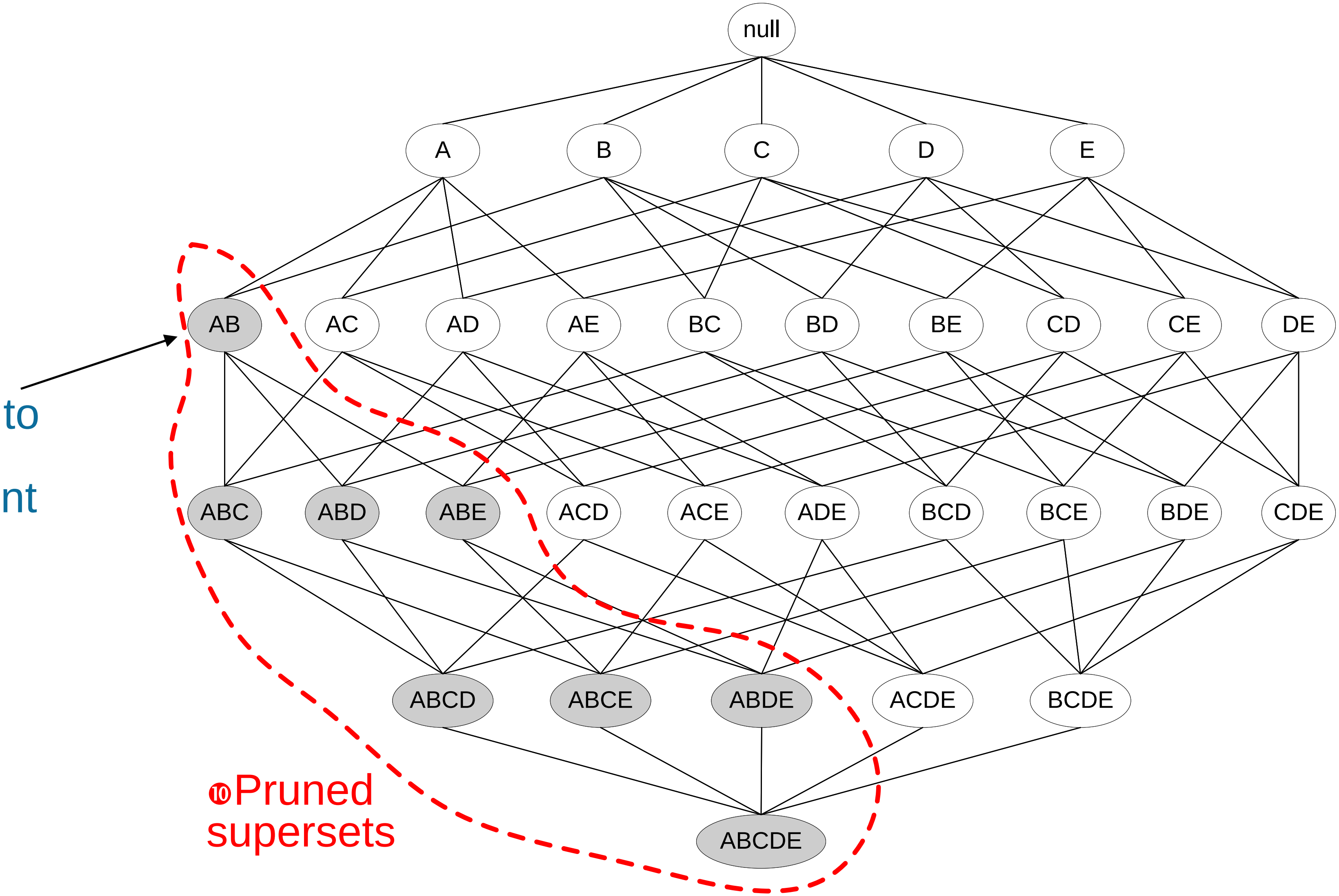
- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent

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

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

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

Illustrating Apriori Principle



❿Found to be Infrequent

❿Pruned supersets

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,
- $$6 + 6 + 1 = 13$$

Another Example

- Database TDB

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

- Sup_{min} = 2

- 1st scan

- C_1

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

- L_1

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

- L_2

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

- C_3

Itemset
{A, C, E}

- 3rd scan

- L_3

Itemset	sup
{A, C, E}	2

Apriori Algorithm

- Let k=1
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are identified
  - 1. Generate candidate (k+1)-itemsets from frequent k-itemsets
  - 2. Prune candidate (k+1)-itemsets containing some infrequent k-itemset
  - 3. Count the support of each candidate by scanning the DB
  - 4. Eliminate infrequent candidates, leaving only those that are frequent

1. Generate Candidate (k+1) itemsets

- $\text{Sup}_{min} = 2$

- L_2

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

- C_3

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

- Input: frequent k-itemsets L_k
- Output: frequent (k+1)-itemsets L_{k+1}
- Procedure:
- 1. Candidate generation, by self-join L_k*L_k
  - For each pair of $P=\{p_1, p_2, \ldots, p_k\} \in L_{k,}$ $q=\{q_1, q_2, \ldots, q_k\} \in L_{k,}$
    - if $p_1=q_1, \ldots, p_{k-1}=q_{k-1}, p_k < q_{k,}$ add $\{p_1, \ldots, p_{k-1}, p_k, q_{k\}}$ into C_{k+1}

- *Example:* L_2={AB, AC, AE, CE}
  - AB and AC => ABC
  - AB and AE => ABE
  - AC and AE => ACE

2. Prune Candidates

- $Sup_{min} = 2$

- L_2

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

- C_3

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

- Input: frequent k-itemsets L_k
- Output: frequent (k+1)-itemsets L_{k+1}
- Procedure:
- 1. Candidate generation, by self-join L_k*L_k
  - For each pair of $P=\{p_1, p_2, \ldots, p_k\} \in L_{2,}$ $q=\{q_1, q_2, \ldots, q_k\} \in L_{2,}$
    - if $p_1=q_1, \ldots, p_{k-1}=q_{k-1}, p_k < q_{k,}$ add $\{p_1, \ldots, p_{k-1}, p_k, q_{k\}}$ into C_{k+1}

- 2. Prune candidates that contain infrequent k-itemsets

- *Example:* L_2={AB, AC, AE, CE}
  - AB and AC => ABC, pruned because BC is not frequent
  - AB and AE => ABE, pruned because BE is not frequent
  - AC and AE => ACE

3. Count support of candidates and 4. Eliminate infrequent candidates

- $\text{Sup}_{min} = 2$

- L_2

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

- C_3

Itemset
{A, C, E}

- 3rd scan

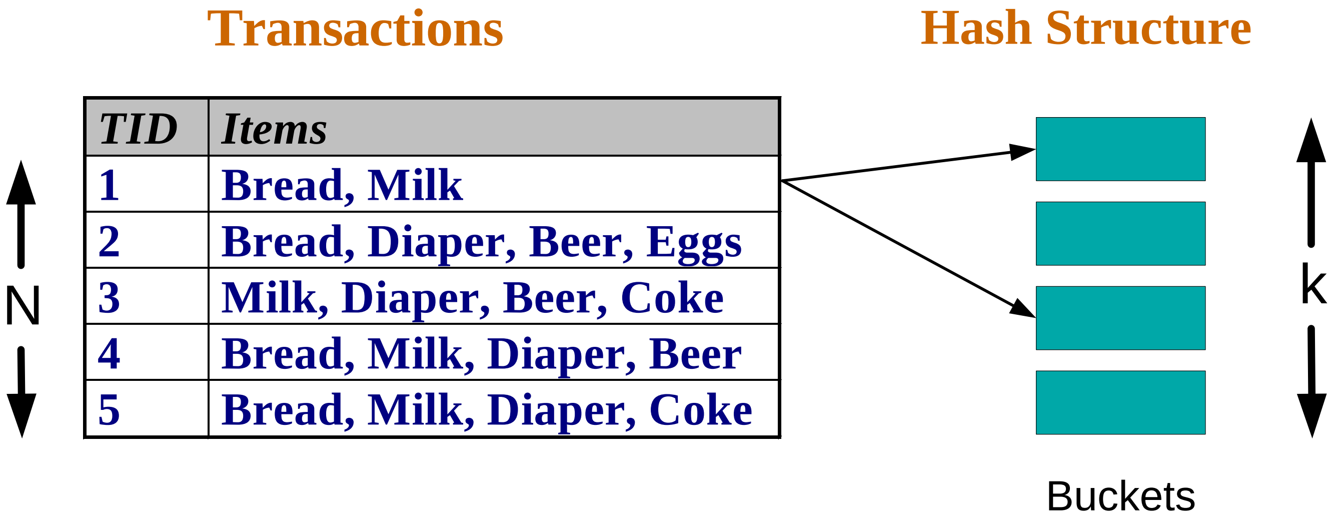
- L_3

Itemset	sup
{A, C, E}	2

- Input: frequent k-itemsets L_k
- Output: frequent (k+1)-itemsets L_{k+1}
- Procedure:
- 1. Candidate generation, by self-join L_k*L_k
  - For each pair of $P=\{p_1, p_2, \ldots, p_k\} \in L_2$, $q=\{q_1, q_2, \ldots, q_k\} \in L_2$,
    - if $p_1=q_1, \ldots, p_{k-1}=q_{k-1}, p_k < q_k$, add $\{p_1, \ldots, p_{k-1}, p_k, q_k\}$ into C_{k+1}
- 2. Prune candidates that contain infrequent k-itemsets
- 3. Count the support of each candidate by scanning the DB
  - 4. Eliminate infrequent candidates

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



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

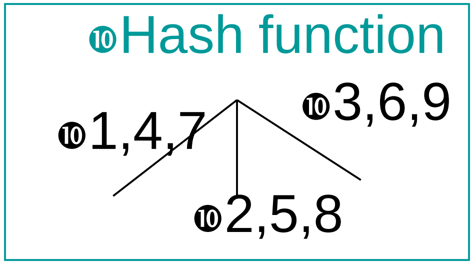
Generate Hash Tree

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)
- An order on the items (e.g., 1 .. 9,   Beer, Bread, Coke, Diaper, Egg, Milk)

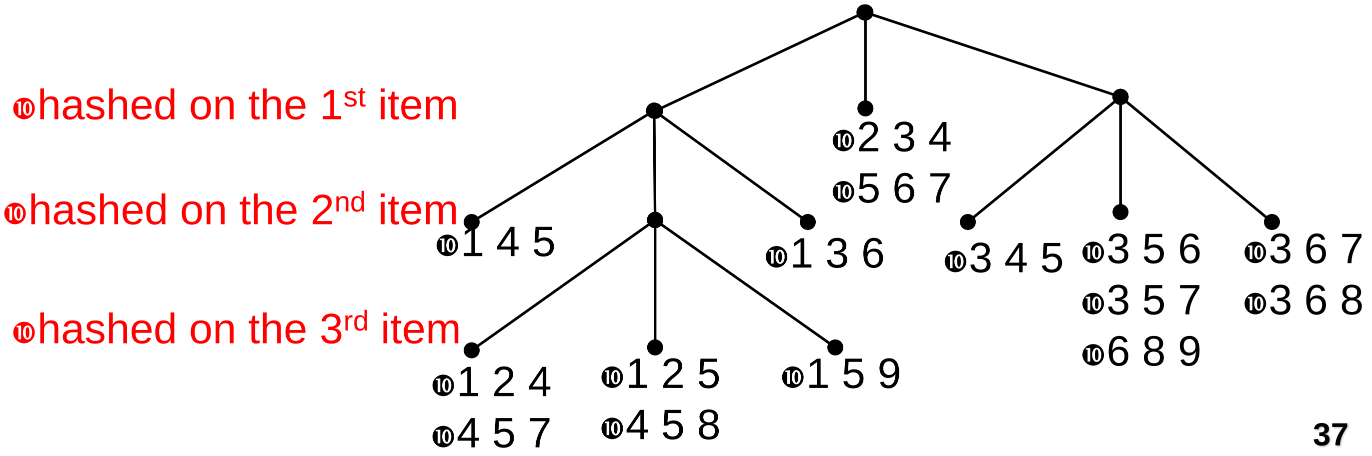


⑩Hash function

⑩1,4,7

⑩3,6,9

⑩2,5,8



⑩hashed on the 1st item

⑩hashed on the 2nd item

⑩hashed on the 3rd item

⑩2 3 4
⑩5 6 7

⑩1 4 5

⑩1 3 6

⑩3 4 5

⑩3 5 6
⑩3 5 7
⑩6 8 9

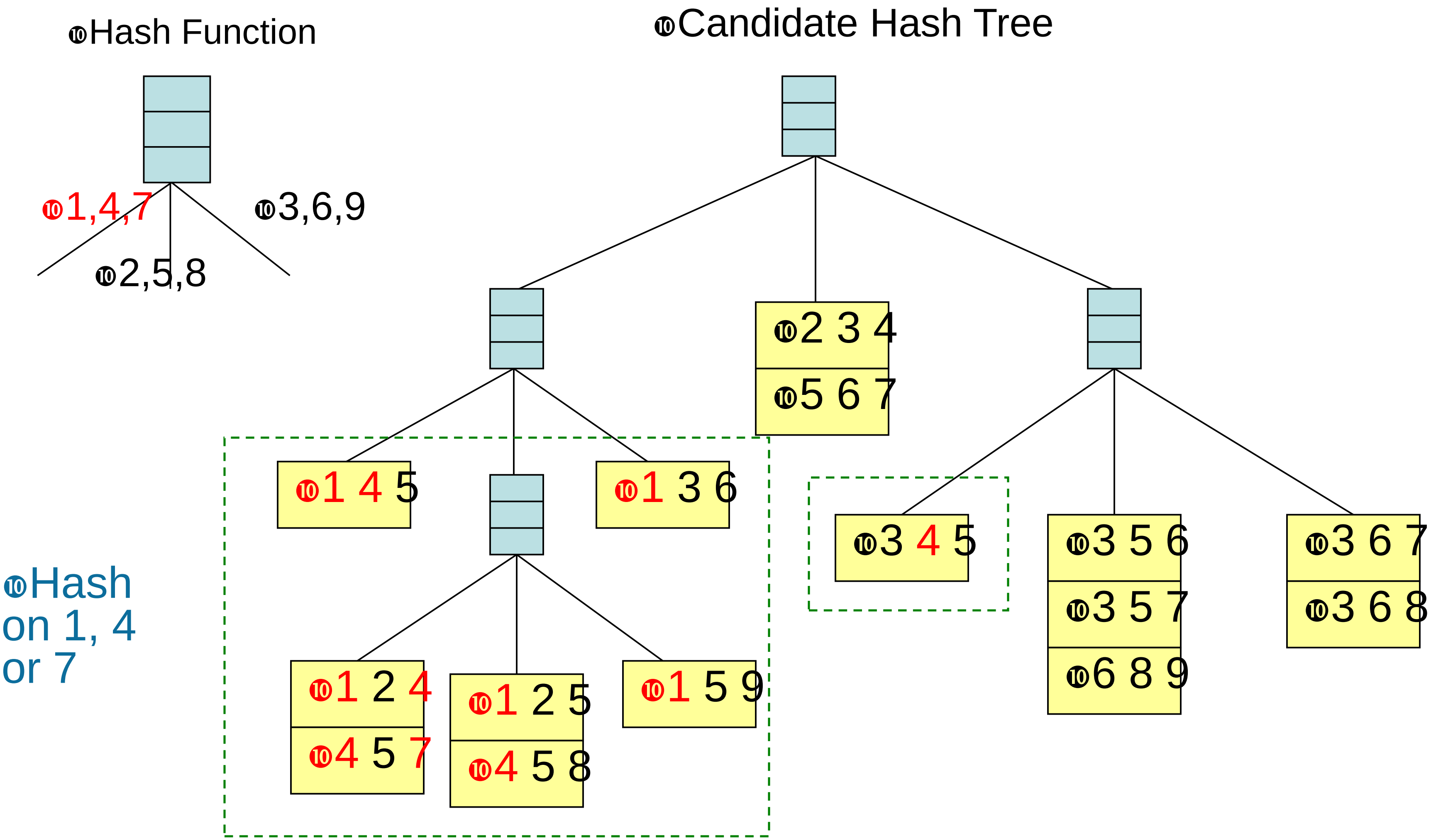
⑩3 6 7
⑩3 6 8

⑩1 2 4
⑩4 5 7

⑩1 2 5
⑩4 5 8

⑩1 5 9

Association Rule Discovery: Hash tree



Hash Function

1,4,7

2,5,8

3,6,9

Candidate Hash Tree

2 3 4

5 6 7

1 4 5

1 3 6

1 2 4

4 5 7

1 2 5

4 5 8

1 5 9

3 4 5

3 5 6

3 5 7

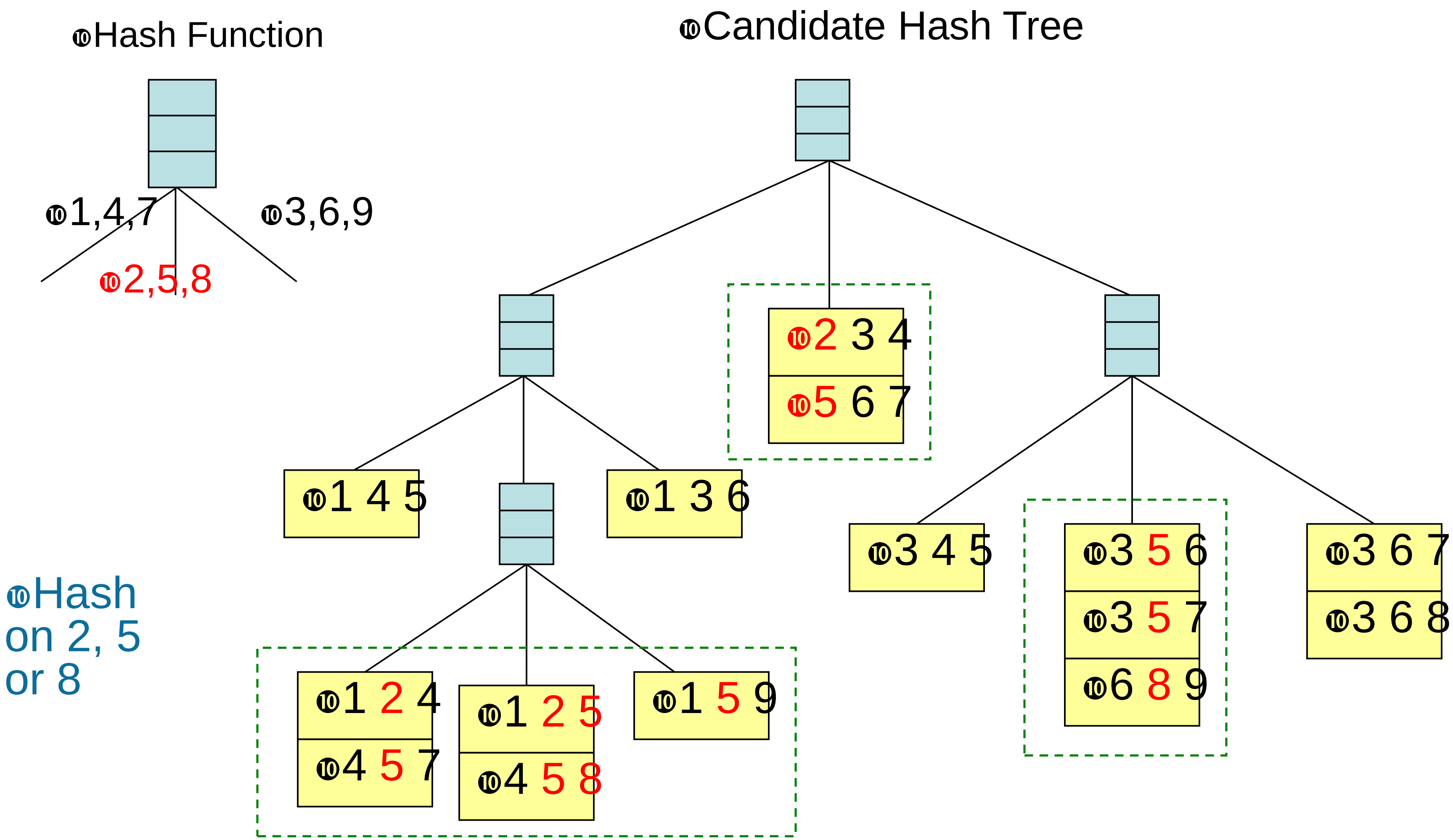
6 8 9

3 6 7

3 6 8

Hash on 1, 4 or 7

Association Rule Discovery: Hash tree

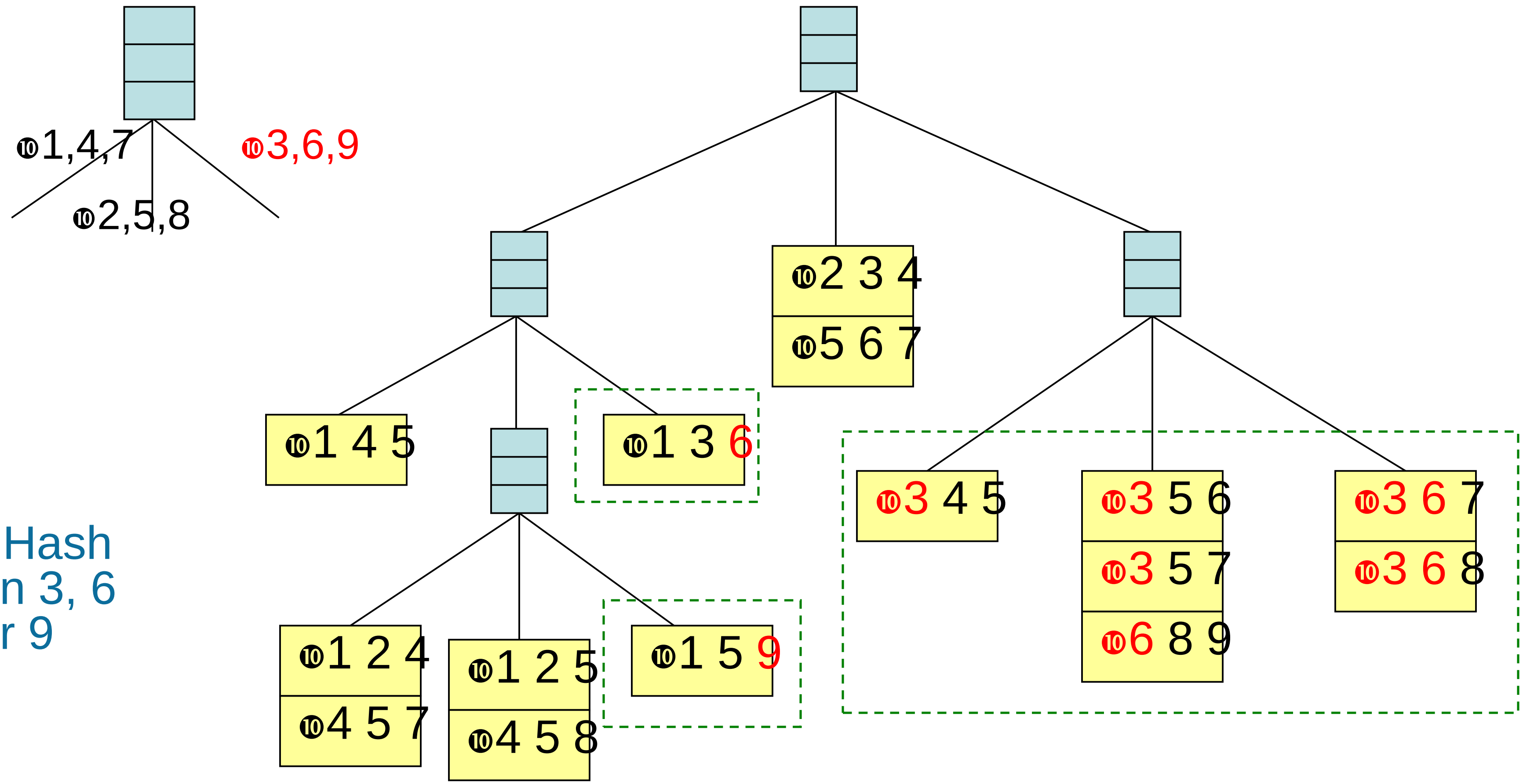


Hash on 2, 5 or 8

Association Rule Discovery: Hash tree

⑩Hash Function

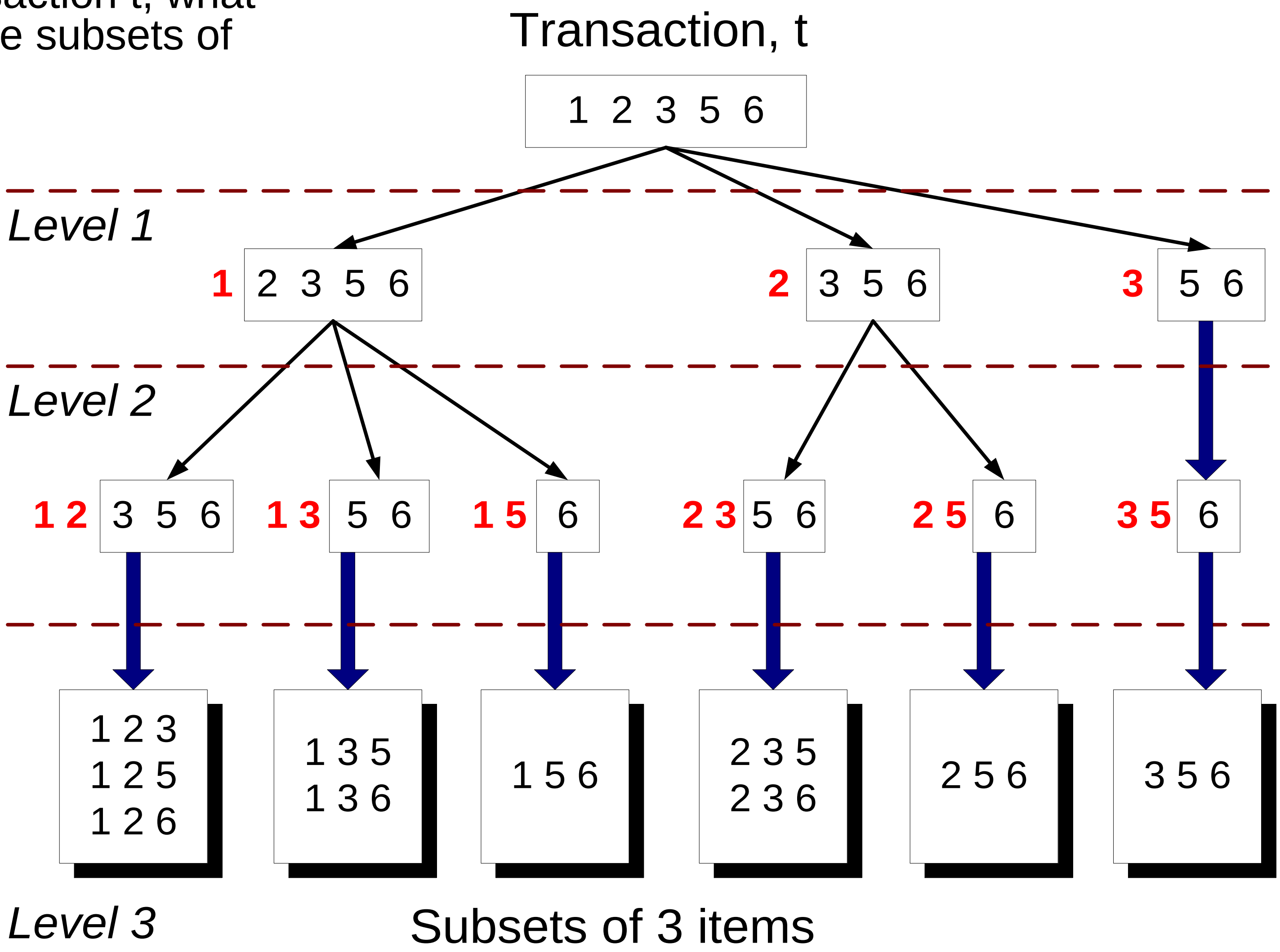
⑩Candidate Hash Tree



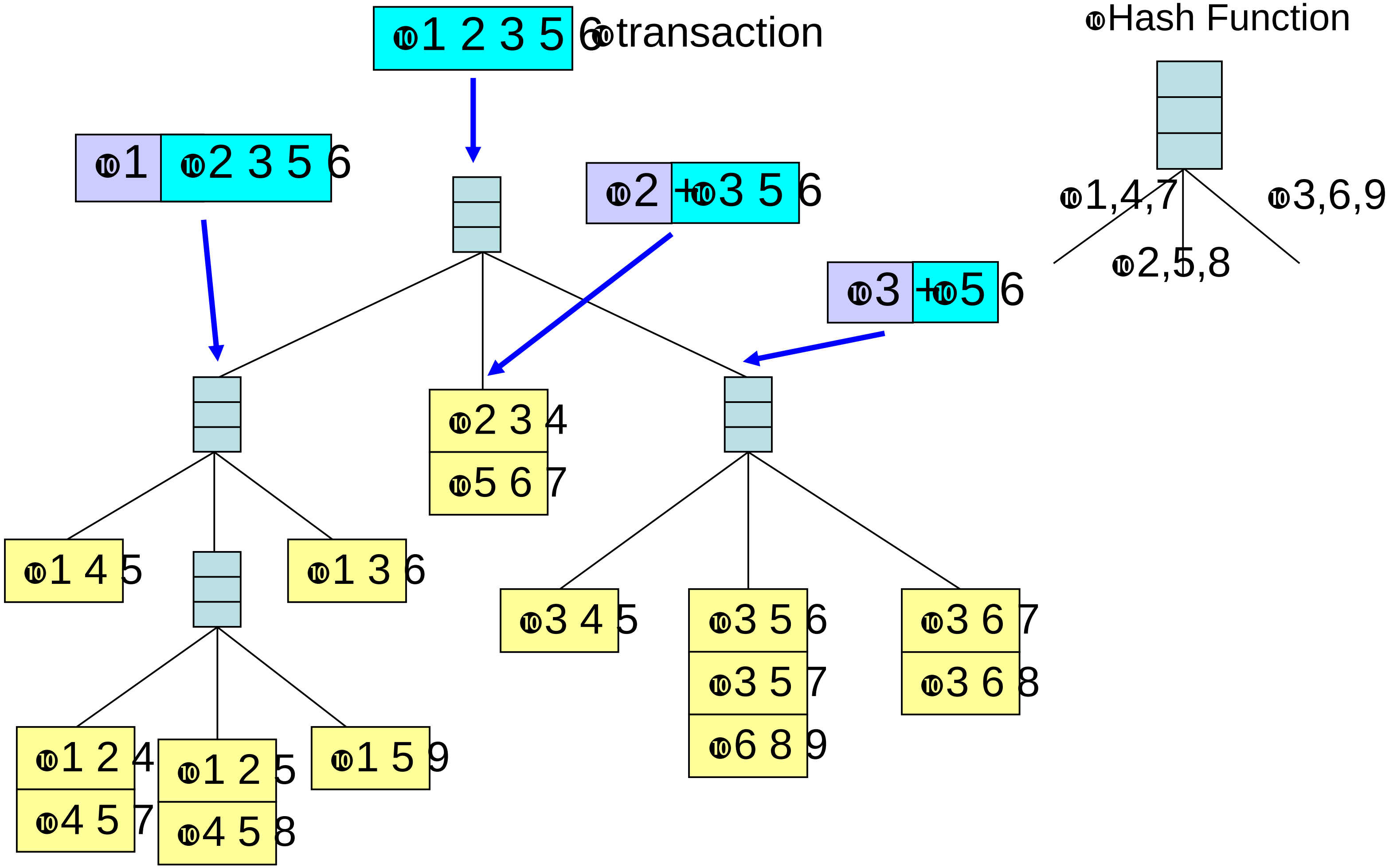
⑩Hash on 3, 6 or 9

Subset Operation

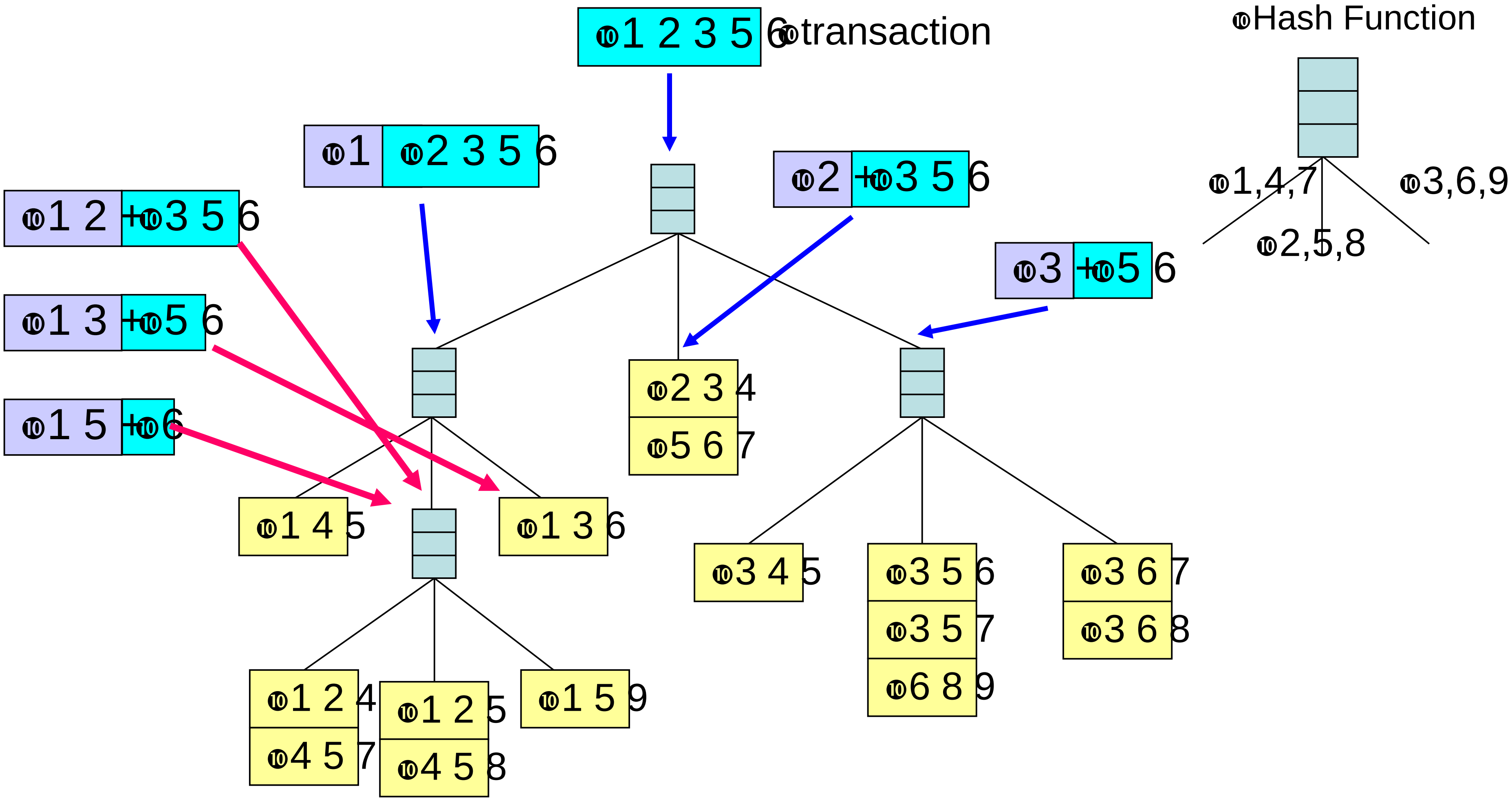
- Given a transaction t, what are the possible subsets of size 3?



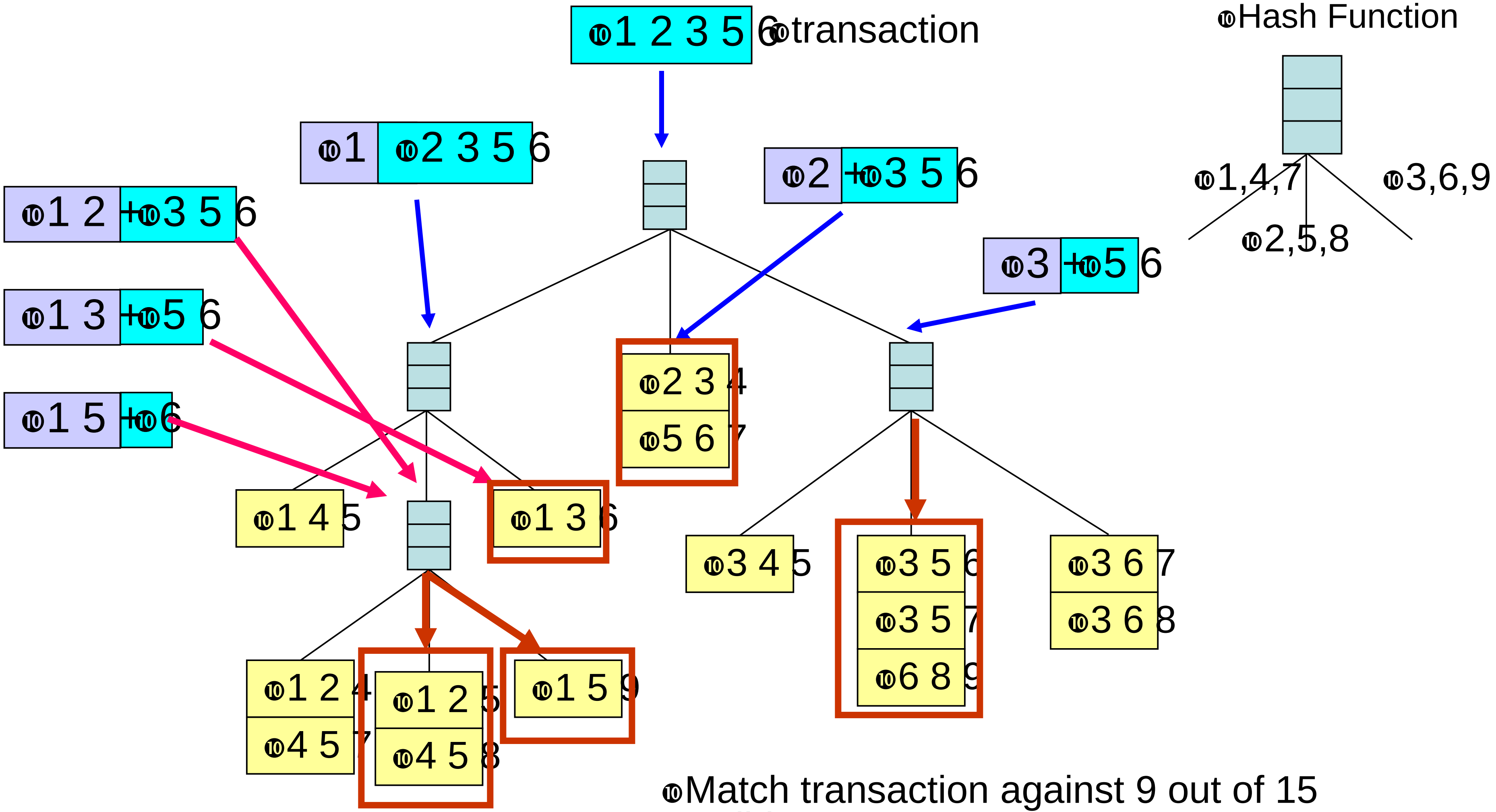
Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Match transaction against 9 out of 15 candidates

(SEC. IV)

ASSOCIATION RULES GENERATION

Rule Generation

- Given a frequent itemset L, find all non-empty subsets f ⊂ L such that f → 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 → ∅ and ∅ → L)

Rule Generation

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

    c(ABC →D) can be larger or smaller than c(AB →D)

  - But confidence of rules generated from the same itemset has an anti-monotone property

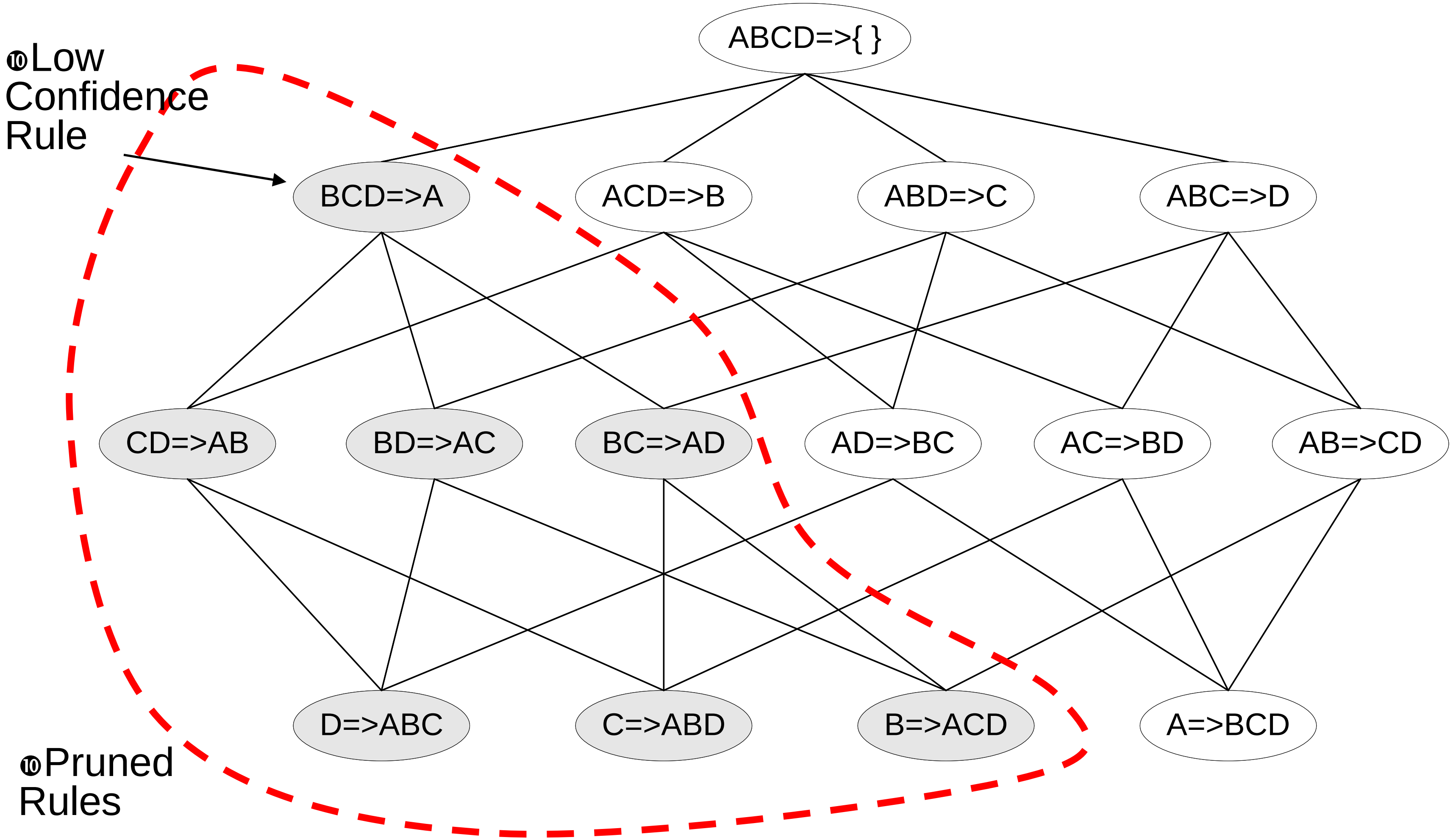
    e.g., L = {A,B,C,D}:

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

    – Confidence is anti-monotone w.r.t. the RHS of the rule

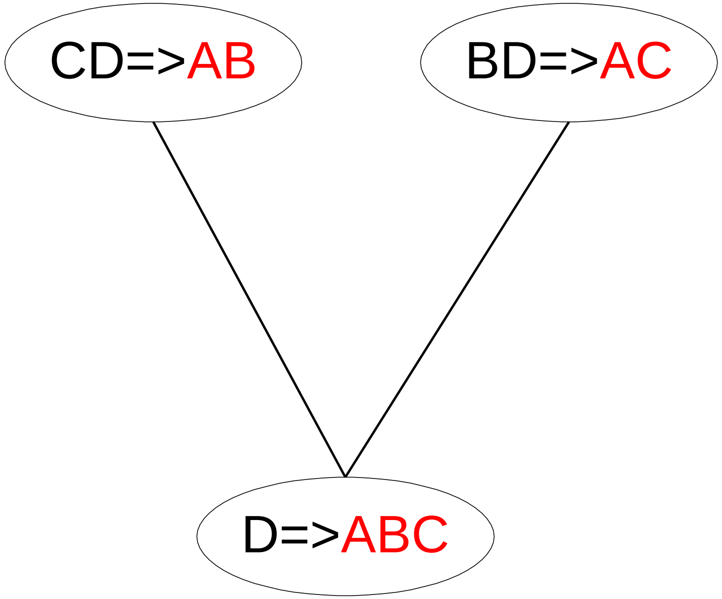
Rule Generation for Apriori Algorithm

- 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