Data Mining: Association Analysis

Laura Brown

Some slides adapted from G. Piatetsky-Shapiro; Han, Kamber, & Pei; Tan, Steinbach, & Kumar

Association Rule Mining

- Given a set of transaction, 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!

Basic Definitions

- Itemset
  - collection of one or more items
    - Ex. {Milk, Bread, Diaper}
  - k-itemset
    - an itemset with *k* items
- Support count (σ), (absolute support)
  - frequency of occurrence of an itemset
    - Ex. σ({Milk, Bread, Diaper}) = 2
- Support, (relative support)
  - fraction of transactions that contain an itemset (prob. of transaction containing itemset)
    - Ex. s({Milk, Bread, Diaper}) = 2/5
- Frequent itemset
  - 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

Basic Definitions

- Association Rules
  - implication expression of the form X -> Y, where X and Y are itemsets
  - X and Y are disjoint
  - Ex: {Milk, Diaper} -> {Beer}

- Rule Evaluation Metrics
  - Support (s)
    - fraction of transactions than contain items of X and Y combined
    - probability of transaction contains X U Y
  - Confidence (c)
    - measures how often items in Y appear in transactions that contain X
    - conditional probability that a transaction having X also contains Y

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:

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

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

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

Applications

- Market Basket Analysis
  - stores keep terabytes of information about what customers buy together
    - tells how typical customers navigate stores, lets them position tempting items
    - suggests tie-in “tricks”, e.g., run sale on diapers and raise the price of beer
  - high support needed, or no $$’s

Applications

- Example 1 – text mining
  - baskets = sentences
  - items = words in those sentences
    - find words that appear together unusually frequently, i.e., linked concepts
- Example 2 – document mining
  - baskets = sentences
  - items = documents containing those sentences
    - items that appear together too often could represent plagiarism

Applications

- Example 3 – healthcare mining
  - baskets = people
  - items = genes or blood-chemistry factors
    - detect combinations of genes that results in a disease
    - requires extension: absence of an item needs to be observed as well as presence

Example: Frequent Itemsets

- Items = { milk, coke, pepsi, beer, juice }
- MinSupport = 3 baskets

B_1 = {m, c, b} B_2 = {m, p, j}

B_3 = {m, b} B_4 = {c, j}

B_5 = {m, p, b} B_6 = {m, c, b, j}

B_7 = {c, b, j} B_8 = {b, c}

- Frequent itemsets:

Example: Frequent Itemsets

- Items = { milk, coke, pepsi, beer, juice }
- MinSupport = 3 baskets

B_1 = {m, c, b} B_2 = {m, p, j}
B_3 = {m, b} B_4 = {c, j}
B_5 = {m, p, b} B_6 = {m, c, b, j}
B_7 = {c, b, j} B_8 = {b, c}

- Frequent itemsets:
  - {m}, {c}, {b}, {j}, {m, b}, {b, c}, {c, j}

Subset Property

- Every subset of a frequent set is frequent!
  - If {A, B} is frequent. Each occurrence of A, B includes both A and B, then both A and B alone must also be frequent
- A long pattern (itemsets) contains a combinatorial number of sub-patterns (itemsets)
  - A frequent set with 100 items contains

$$\binom{100}{1}+\binom{100}{2}+\cdots+\binom{100}{100}=2^{100}-1$$

- Solution: look at *closed patterns* and *max-patterns*

Closed Patterns and Max-Patterns

- An itemset X is closed if X is frequent and there exists no super-itemset $Y \supset X$, with the same support as X
- An itemset X is a max-itemset if X is frequent and there exists no frequent super-itemset Y such that $Y \supset X$ and Y is frequent
  - An itemset is maximal if none of its immediate supersets are frequent
- Closed pattern is a lossless compression of freq. patterns
  - reducing the num. of patterns and rules

Closed Patterns and Max-Patterns

- DB = {<a_1, …, a_{100}>, <a_1, …, a_{50}>}
  - minsup = 1
- What is the set of closed itemset?
  - <a_1, …, a_{100}> : 1
  - <a_1, …, a_{50}> : 2
- What is the set of max-pattern?
  - <a_1, …, a_{100}> : 1
- What is the set of all patterns?
  - !!!

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 computationally expensive

From Frequent Itemsets to Association Rules

- Given a frequent set {A, B, E}, what are possible association rules?
  - {A} -> {B, E}
  - {A, B} -> {E}
  - {A, E} -> {B}
  - {B} -> {A, E}
  - {B, E} -> {A}
  - {E} -> {A, B}

TID	List of items
1	**A, B, E**
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	**A, B, C, E**
9	A, B, C

From Frequent Itemsets to Association Rules

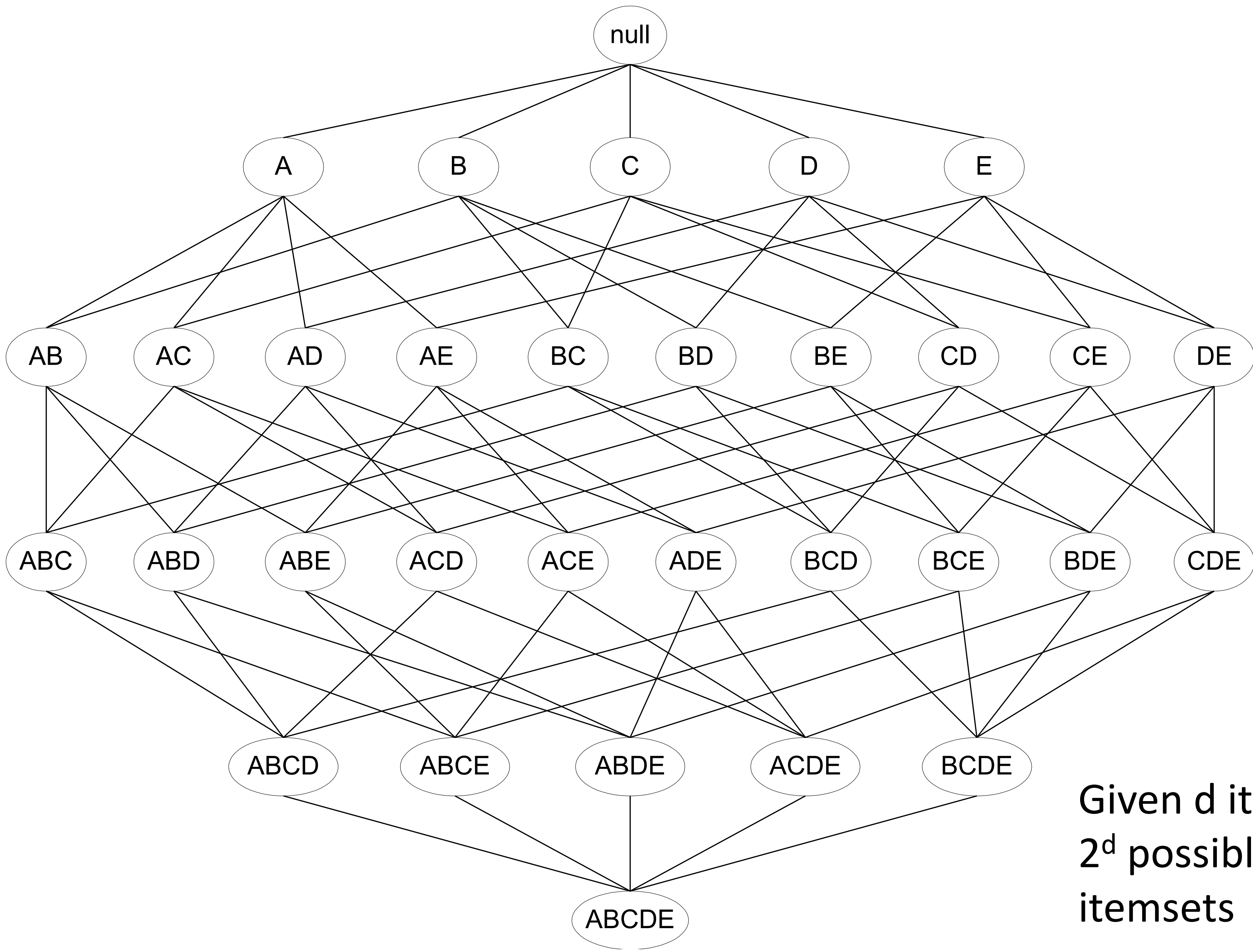
- Given a frequent set {A, B, E}, what are possible association rules?
  - {A} -> {B, E}, c = 2/6 = 0.33
  - {A, B} -> {E}, c = 2/4 = 0.50
  - {A, E} -> {B}, c = 2/2 = 1.00
  - {B} -> {A, E}, c = 2/7 = 0.28
  - {B, E} -> {A}, c = 2/2 = 1.00
  - {E} -> {A, B}, c = 2/2 = 1.00
- Each rule is binary partition of same itemset
  - have identical support, but different confidence

TID	List of items
1	**A, B, E**
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	**A, B, C, E**
9	A, B, C

Frequent Itemset Mining

- How many itemsets are potentially to be generated in the worst case?
  - number is sensitive to the *minsup* threshold
  - when *minsup* is low, there exists potentially an exponential number of frequent itemsets

Frequent Itemset Generation



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

Frequent Itemset Generation

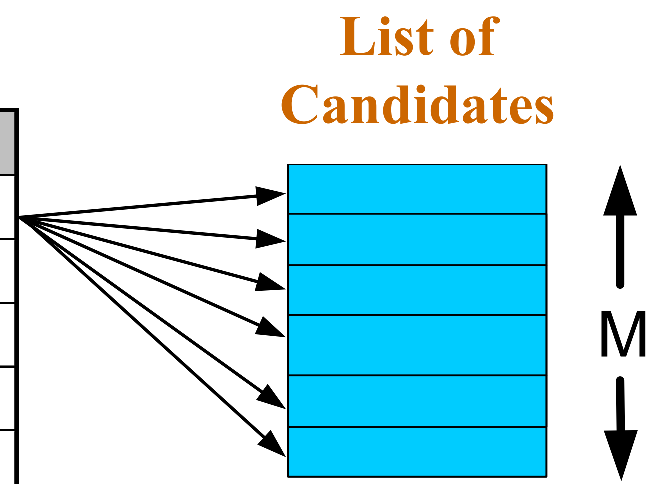
- Brute-force approach
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

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

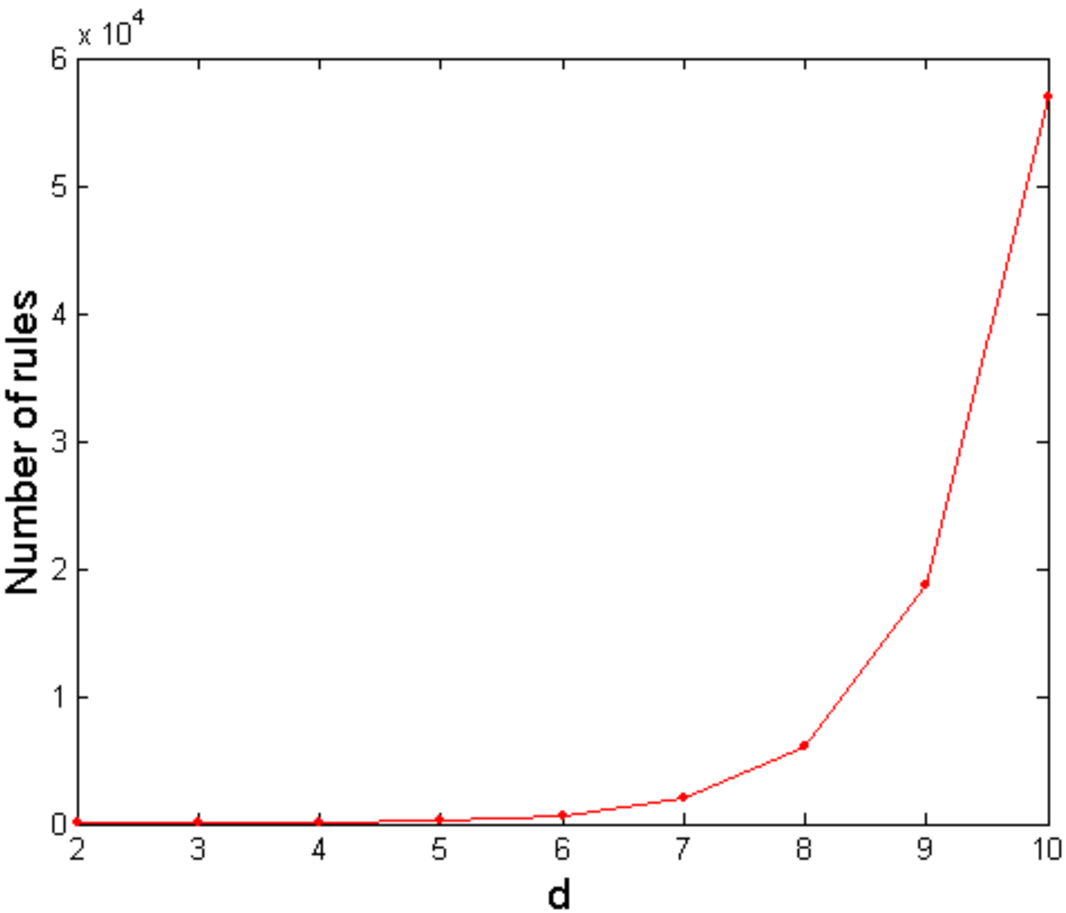
w



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

Computational Complexity

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



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

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

If d=6, R = 602 rules

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - complete search: $M=2^d$
  - use pruning methods 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 candidates or transactions
  - no need to match every candidate against every transaction

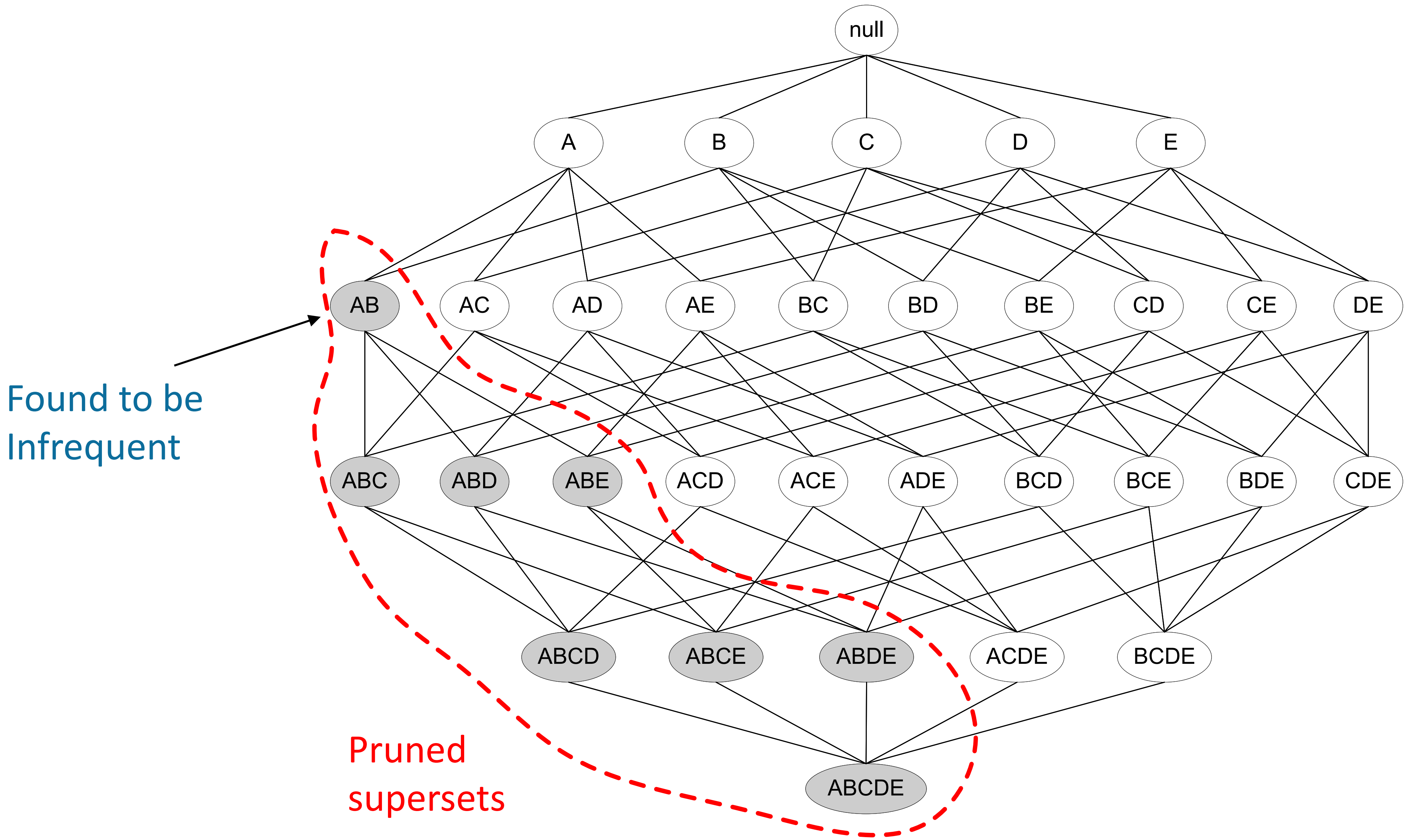
Apriori

- Apriori principle:
  - if an itemset is frequent, then all of its subsets must also be frequent
  - if there is any itemset which is infrequent, its supersets should not be generated

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

  - support of an itemset never exceeds the support of its subsets
  - the anti-monotone property of support

Illustrating Apriori Principle



Apriori Algorithm

- Method
  - let k=1
  - generate frequent itemsets of length 1
  - repeat until no new frequent itemsets identified
    - generate length (k +1) candidate itemsets from length k frequent itemsets
    - prune candidate itemsets containing subsets of length k that are infrequent
    - count support of each candidate by scanning the DB
    - eliminate candidates that are infrequent, leaving only those that are frequent

Agrawal & Srikant, VLDB 1994, Mannila, et al., KDD 1994

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 - Pseudocode

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

L_1 = {frequent items};
for (k = 1; L_k !=emptyset; k++) **do begin**
    C_{k+1} = candidates generated from L_k;
    for each transaction t in database do
        increment the count of all candidates in C_{k+1} that are contained in t
    L_{k+1} = candidates in C_{k+1} with min_support
    end
return $\cup_k L_k$;

Apriori Algorithm - 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

Apriori Algorithm Example: Details

- Consider DB of 9 transactions
- *minsup* = 2/9 = 0.28
- Let *minconf* = 0.70

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2 ,I3, I5
T100	I1, I2, I3

Step 1: Generate 1-itemset patterns

Scan D for count of each candidate

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_1

Compare candidate support count with minimum support count

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1

- Set of frequent 1-itemset, L_1, consists of candidate 1-itemsets satisfying minimum support
- In first iteration, each item is a member of the set of candidates

Step 2: Generate 2-itemset patterns

Generate C_2 candidates from L_1

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

C_2

Scan D for count of each candidate

Itemset	Sup. Count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

C_2

Compare candidate support count with minimum support count

Itemset	Sup Count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

L_2

Step 2: Generate 2-itemset patterns

- To discover the set of frequent 2-itemsets, L_2, the algorithm uses L_1 *Join* L_1 to generate a candidate set of 2-itemsets, C_2
- The transactions in D are scanned and the support count for each candidate itemset in C_2 is accumulated (middle table)
- Set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support

Step 3: Generate 3-itemset patterns

Scan D for count of each candidate →

Itemset
{I1, I2, I3}
{I1, I2, I5}

C_3

Scan D for count of each candidate →

Itemset	Sup. Count
{I1, I2, I3}	2
{I1, I2, I5}	2

C_3

Compare candidate support count with min support count →

Itemset	Sup Count
{I1, I2, I3}	2
{I1, I2, I5}	2

L_3

- Generation of set of candidate 3-itemsets, C_3, involves use of Apriori property
- To find C_3, compute L_2 *Join* L_2
- C_3 = { {I1, I2, I3}, {I1 I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5} }
- Join step complete, prune step used to reduce size of C_3

Step 3: Generate 3-itemset patterns

- Use Aprori property: all subsets of frequent itemsets must also be frequent
- Ex. {I1, I2, I3} has 2-itemsets of:
  - {I1, I2}, {I1, I3}, {I2, I3} are all in L_2,
  - keep {I1, I2, I3} in C_3
- Ex. {I2, I3, I5}
  - {I2, I3}, {I2, I5}, {I3, I5}
  - {I3, I5} not a member of L_2, thus {I2, I3, I5} not in C_3
- Therefore, C_3 = {{I1, I2, I3}, {I1, I2, I5}}

Step 4: Generate 4-itemset patterns

- Algorithm uses L_3 *Join* L_3 to generate 4-itemsets, C_4.
  - Join results in { {I1, I2, I3, I5} }
  - itemset is pruned since { {I2, I3, I5} } is not frequent
- Algorithm terminates, having found all frequent items
- Still left to do
  - generate association rules from itemsets
  - improve efficiency

Implementation Details

- How to Count Supports of Candidates?
  - candidate itemsets are stored in *hash-tree*
  - *leaf* node of hash-tree contains a list of itemsets and counts
  - *interior* node contains a hash table
  - *subset* function: finds all the candidates contained in a transaction

Reducing Number of Comparisons

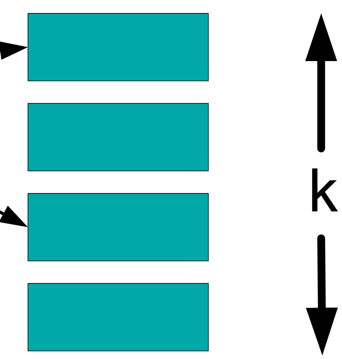
- Candidate counting
  - scan the database of transactions to determine the support of each candidate itemset
  - to reduce the number of comparisons, store the candidates in a hash structure
    - instead of matching each transaction against every candidate, match it against candidates contained in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



Buckets

Generate Hash Tree

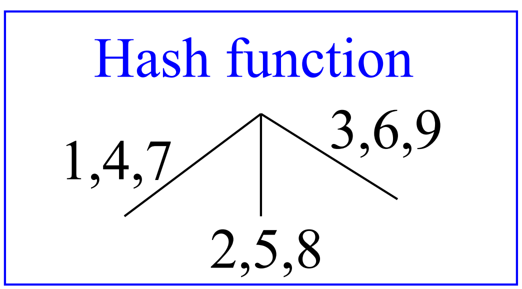
Suppose you have 15 candidate itemsets of length 3:

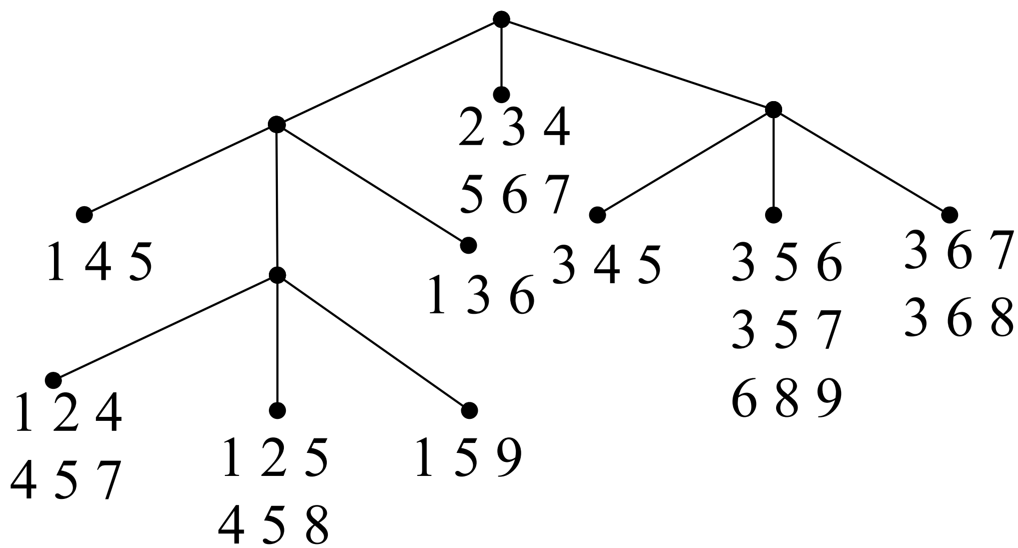
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

You need:

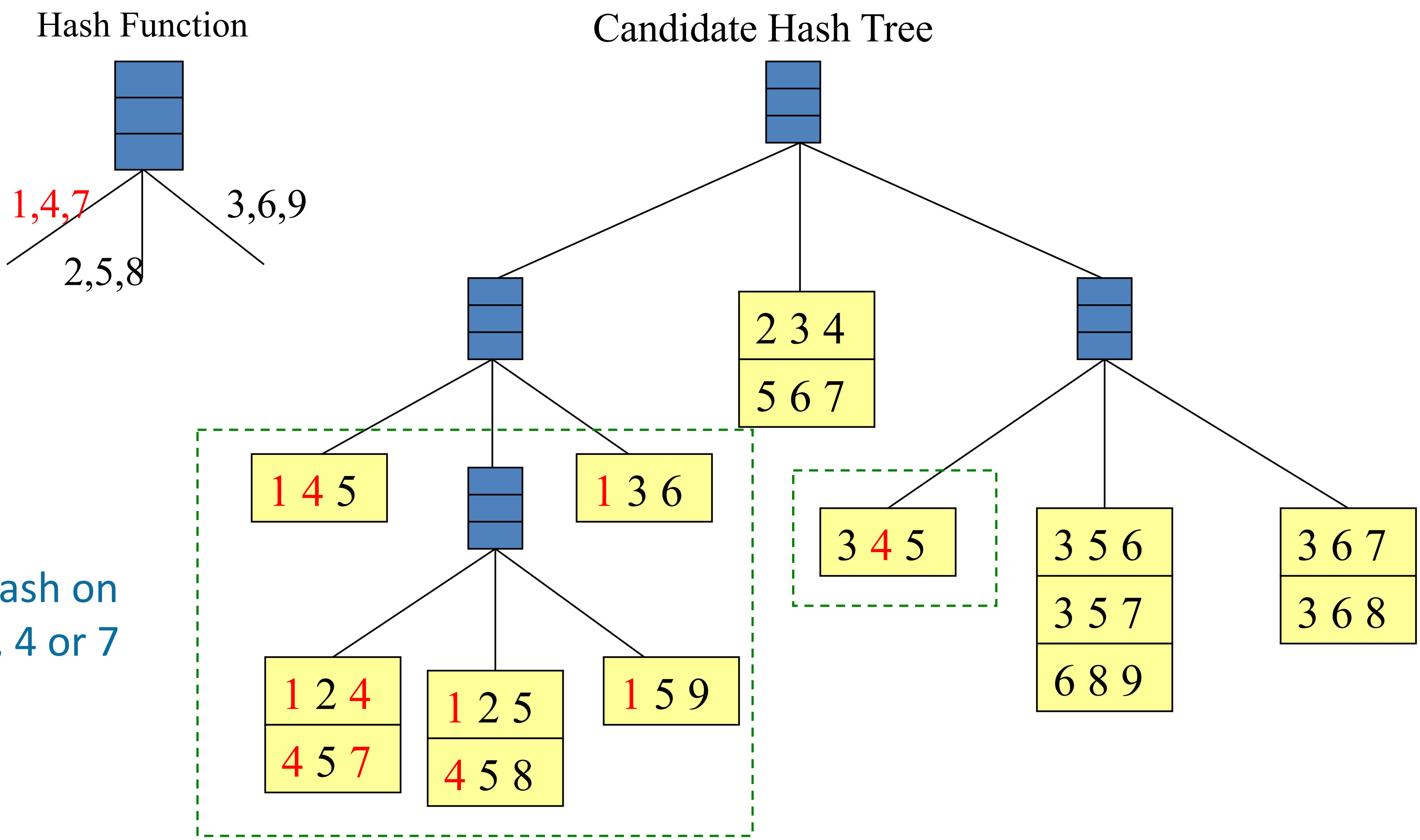
• Hash function / subset function

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



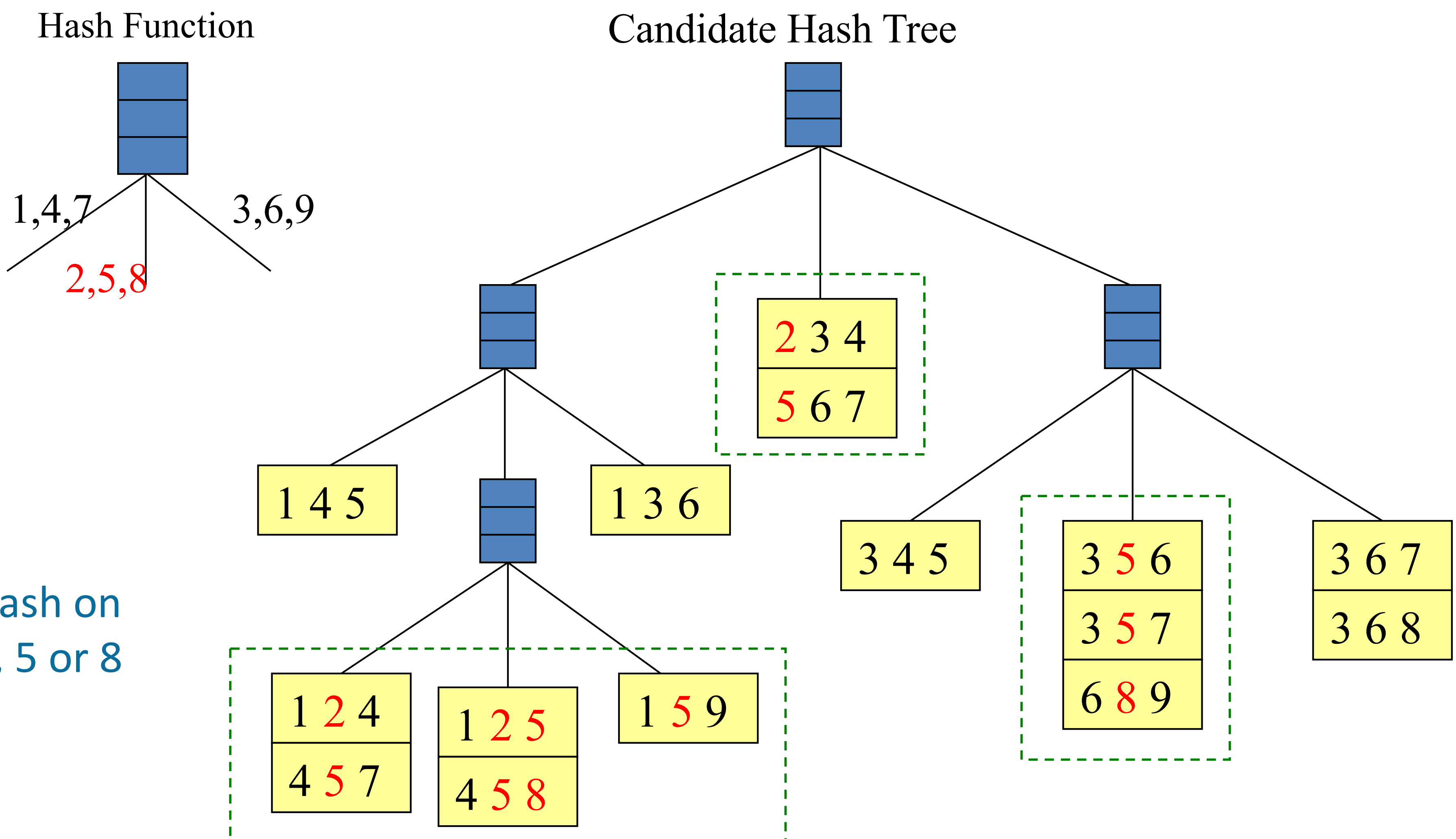


Hash Tree



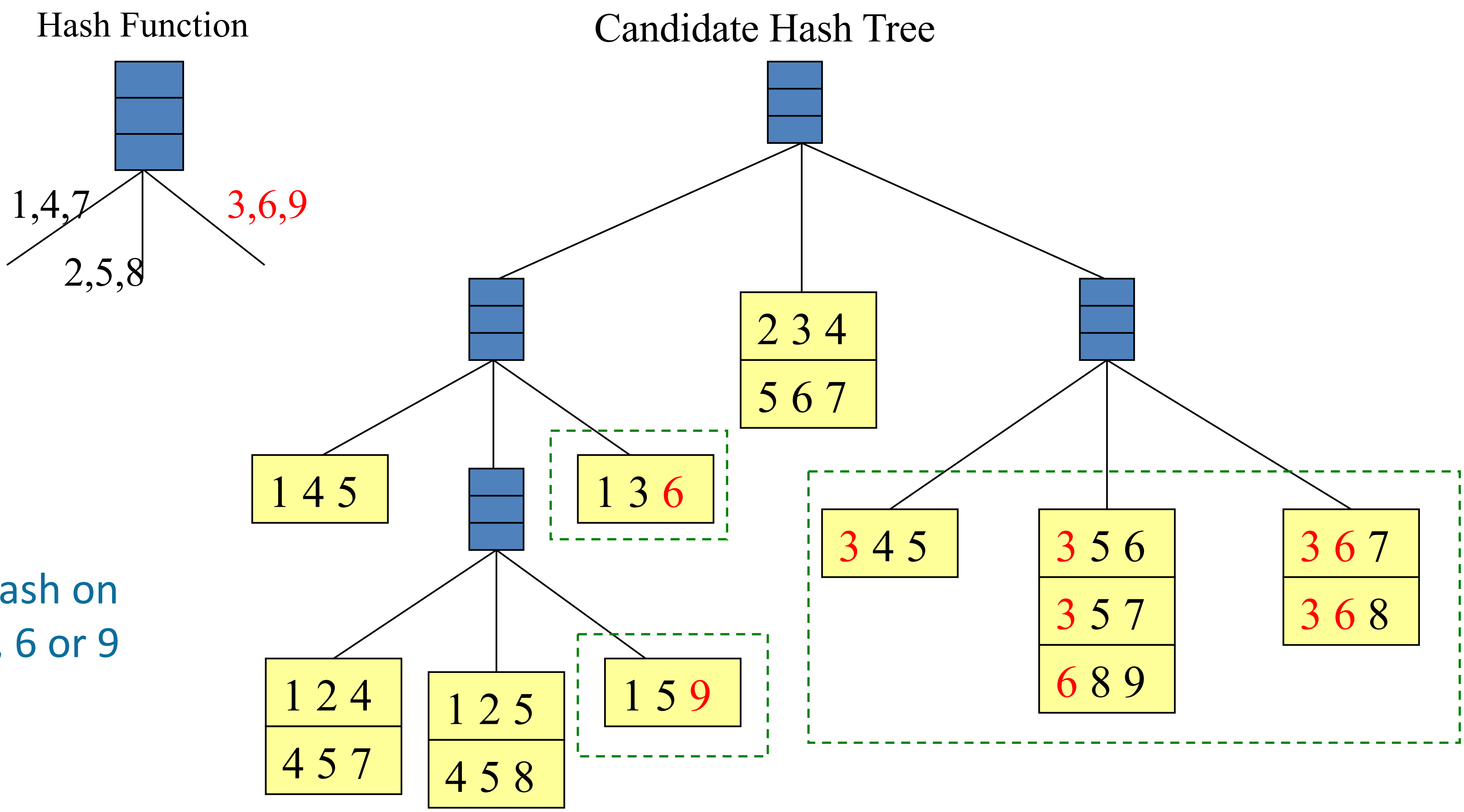
Hash on
1, 4 or 7

Hash Tree



Hash on
2, 5 or 8

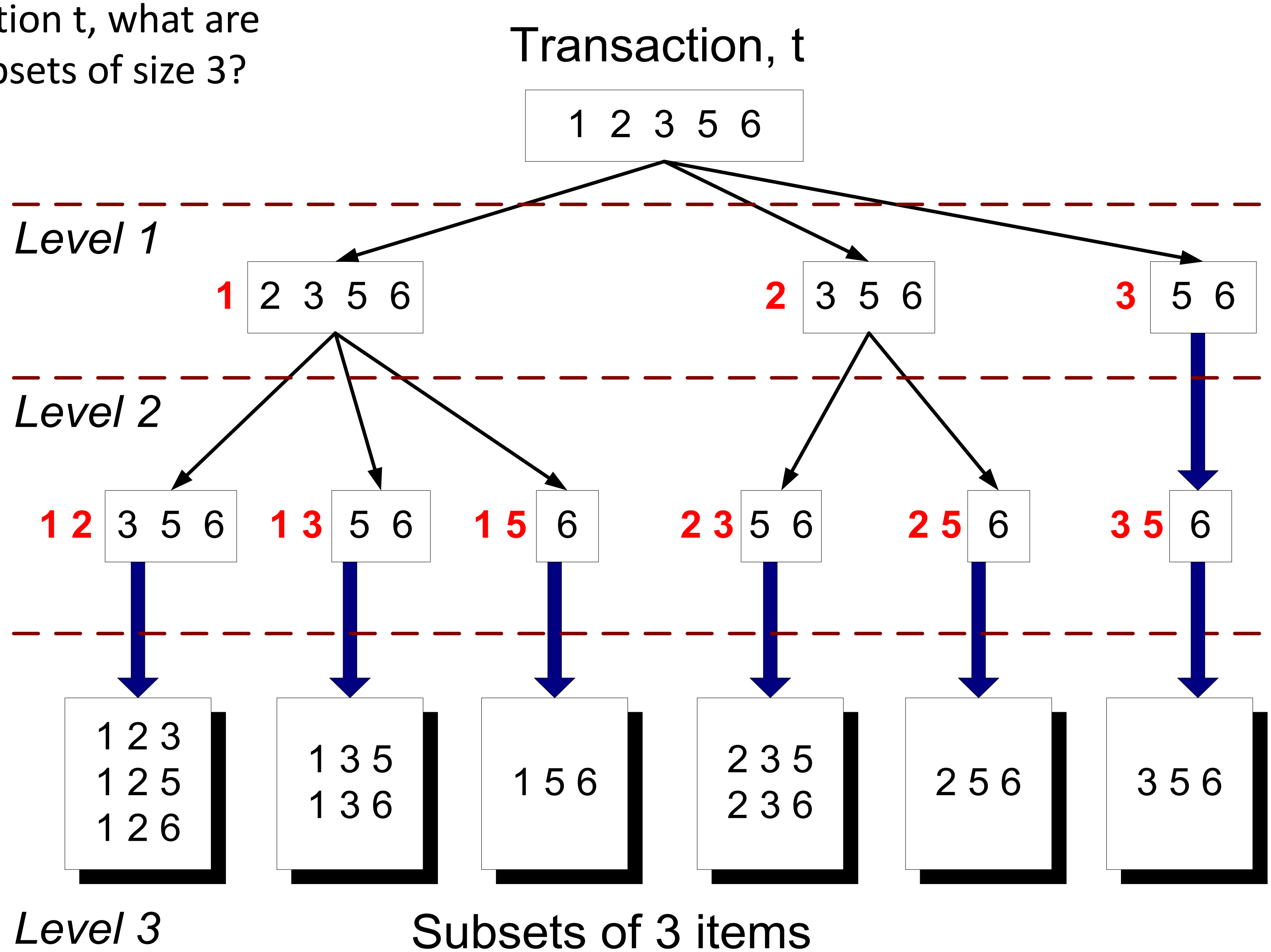
Hash Tree



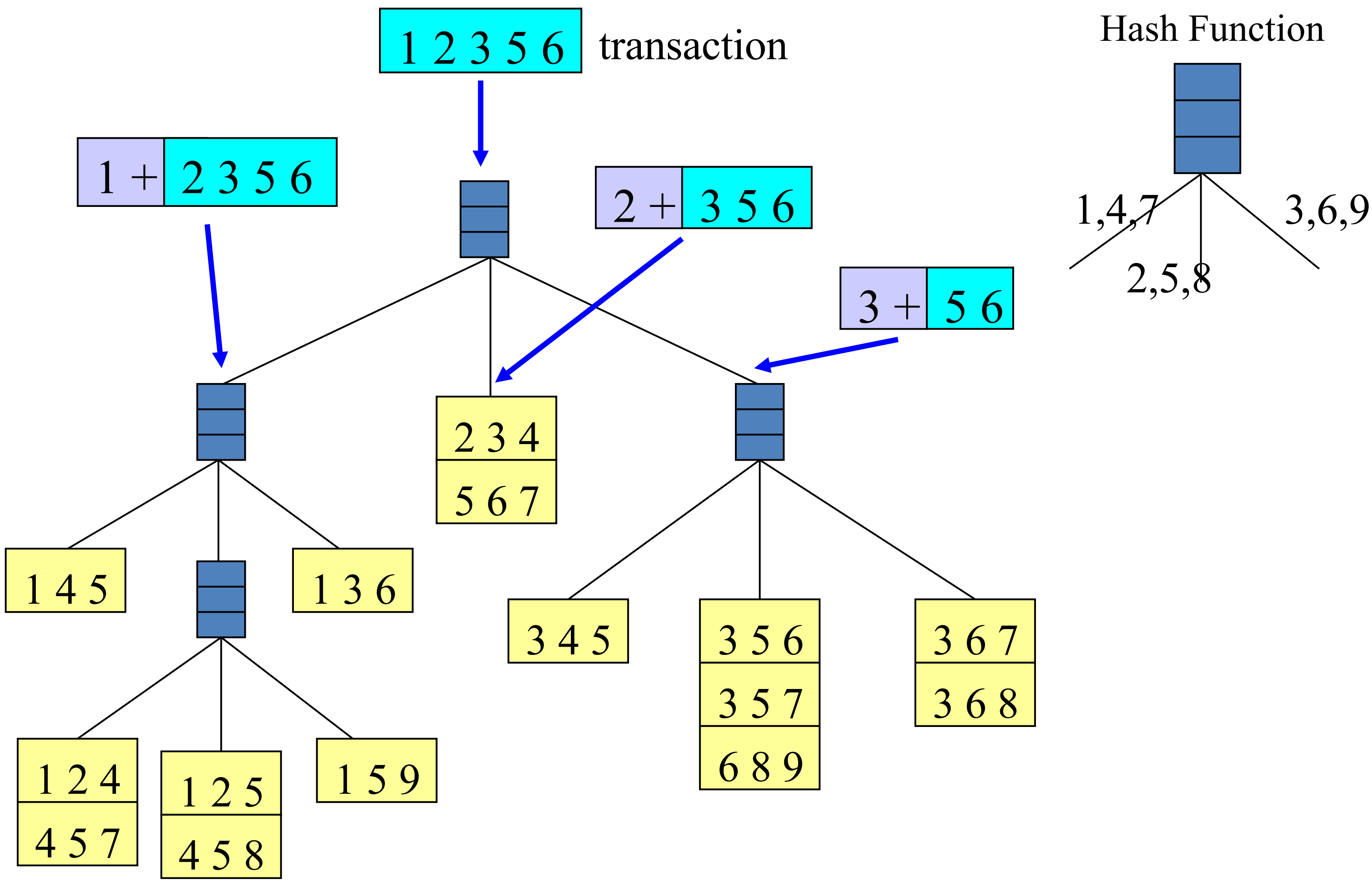
Hash on
3, 6 or 9

Subset Operation

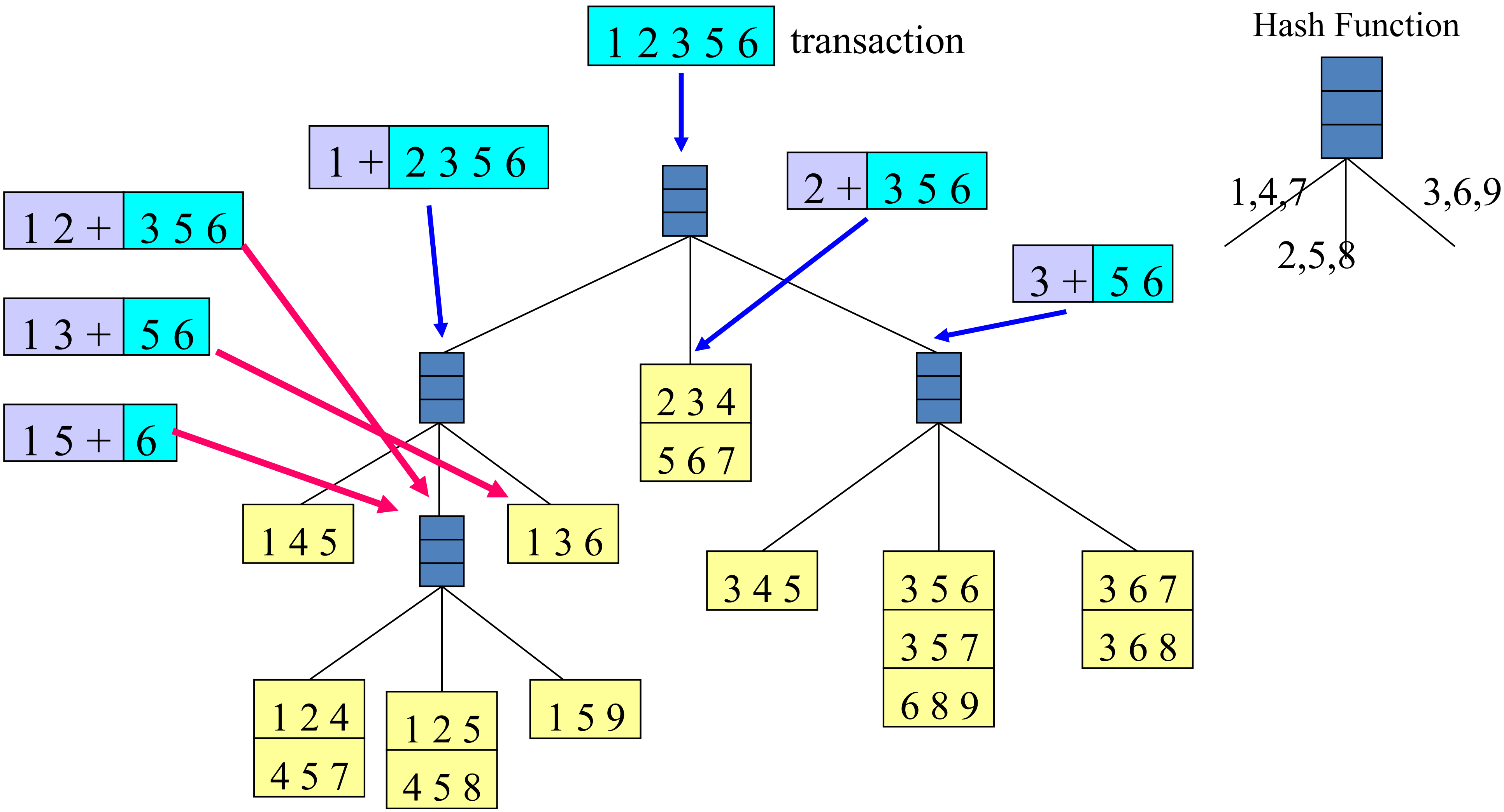
Given a transaction t, what are the possible subsets of size 3?



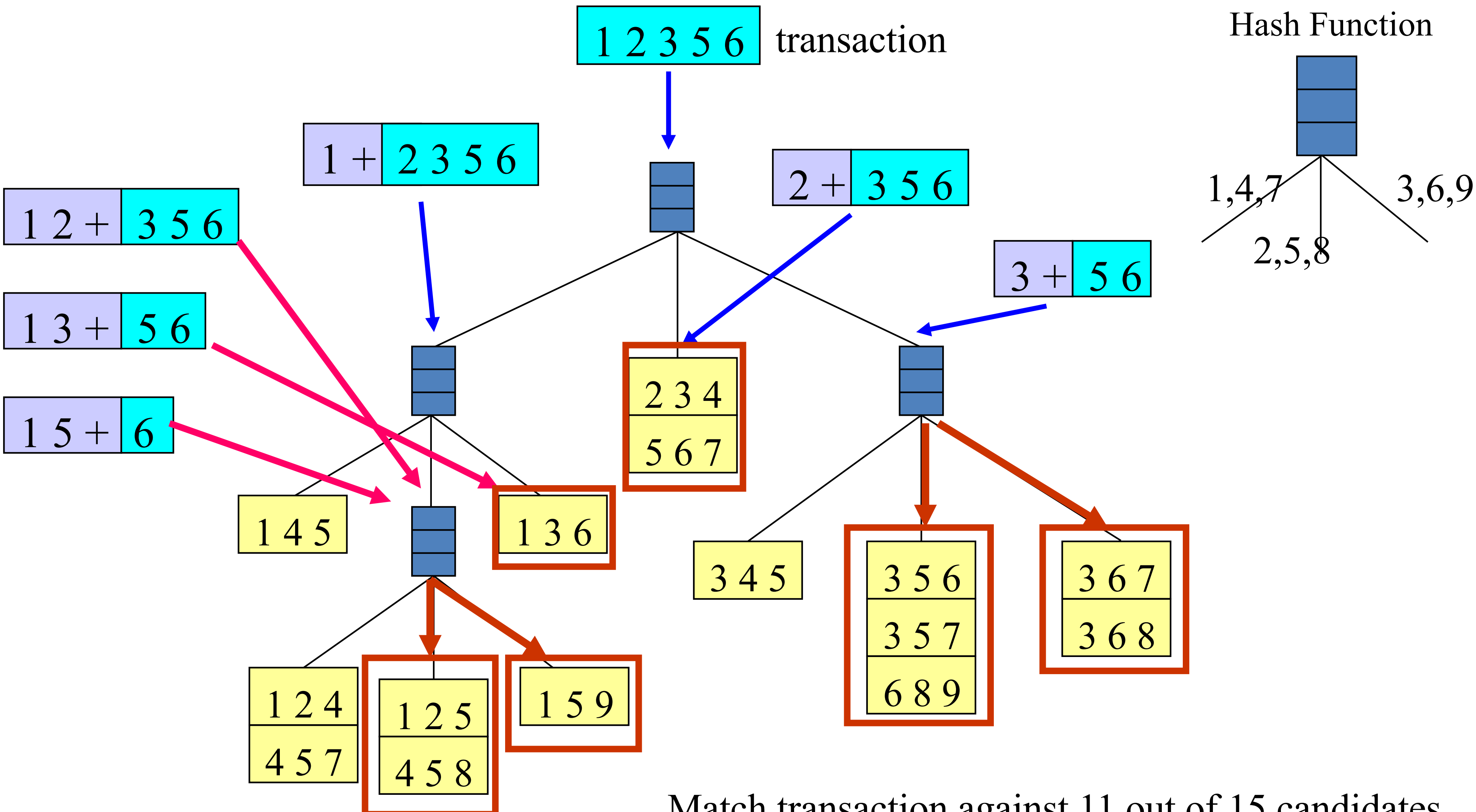
Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Match transaction against 11 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

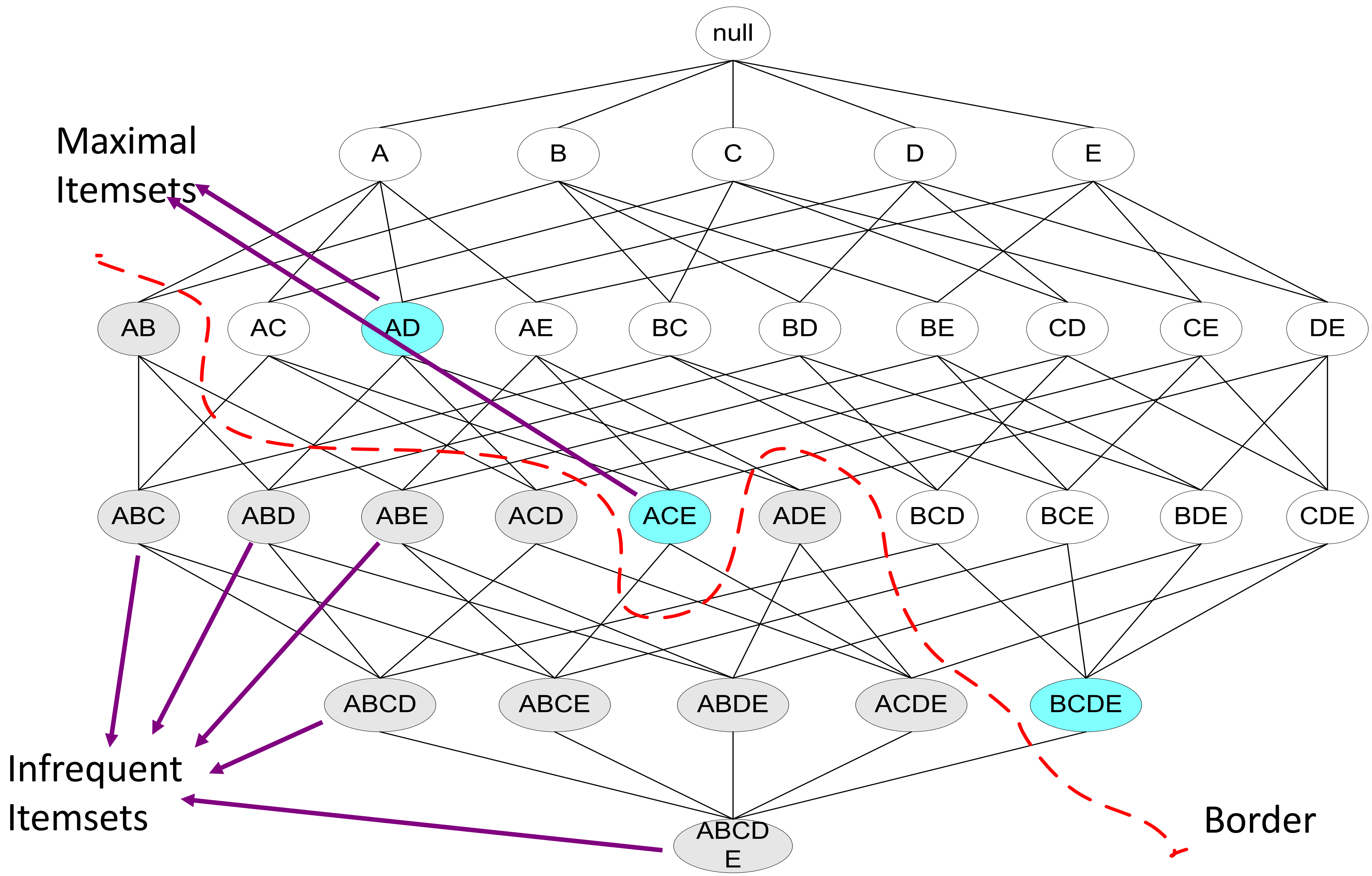
TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1	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

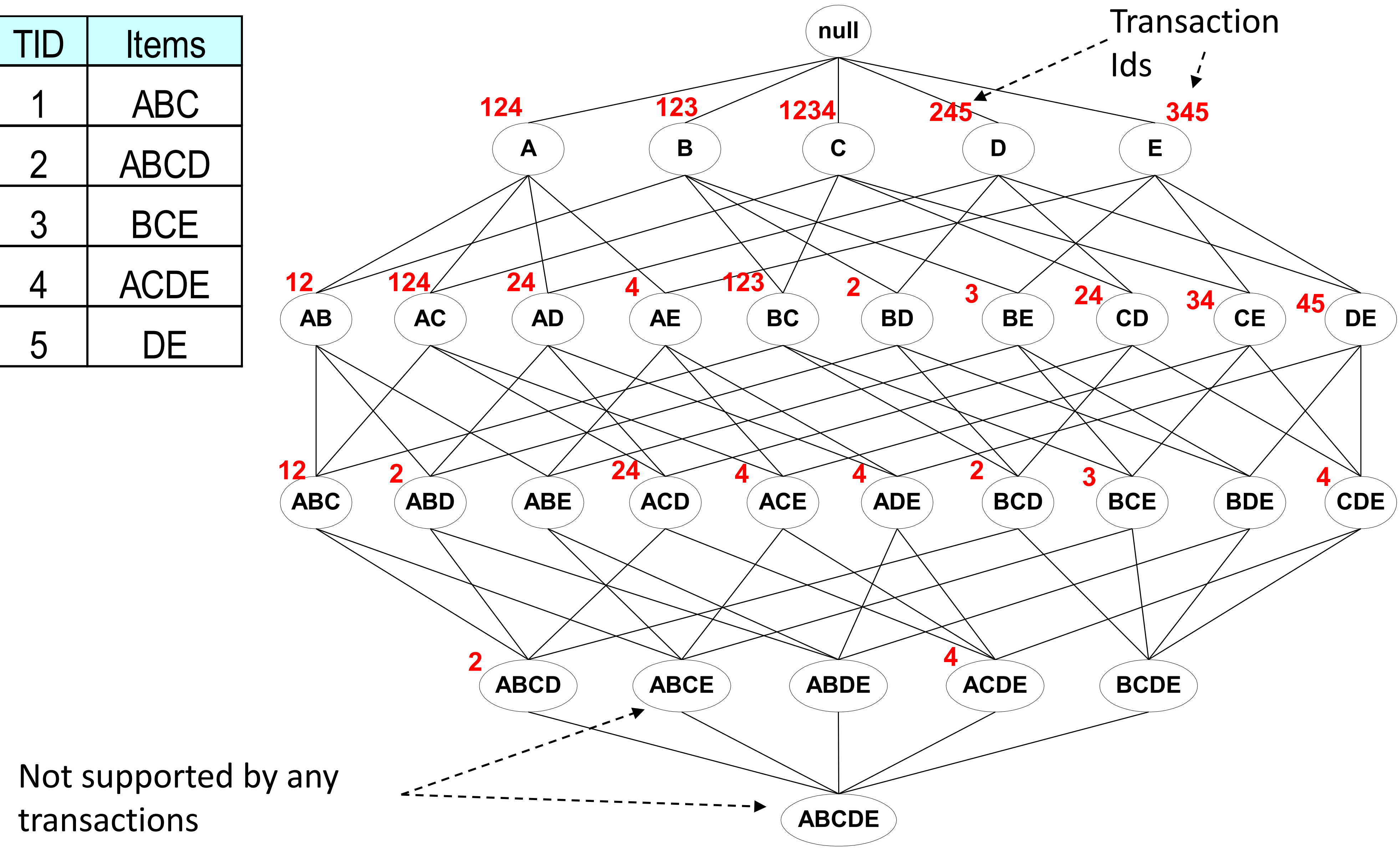
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

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

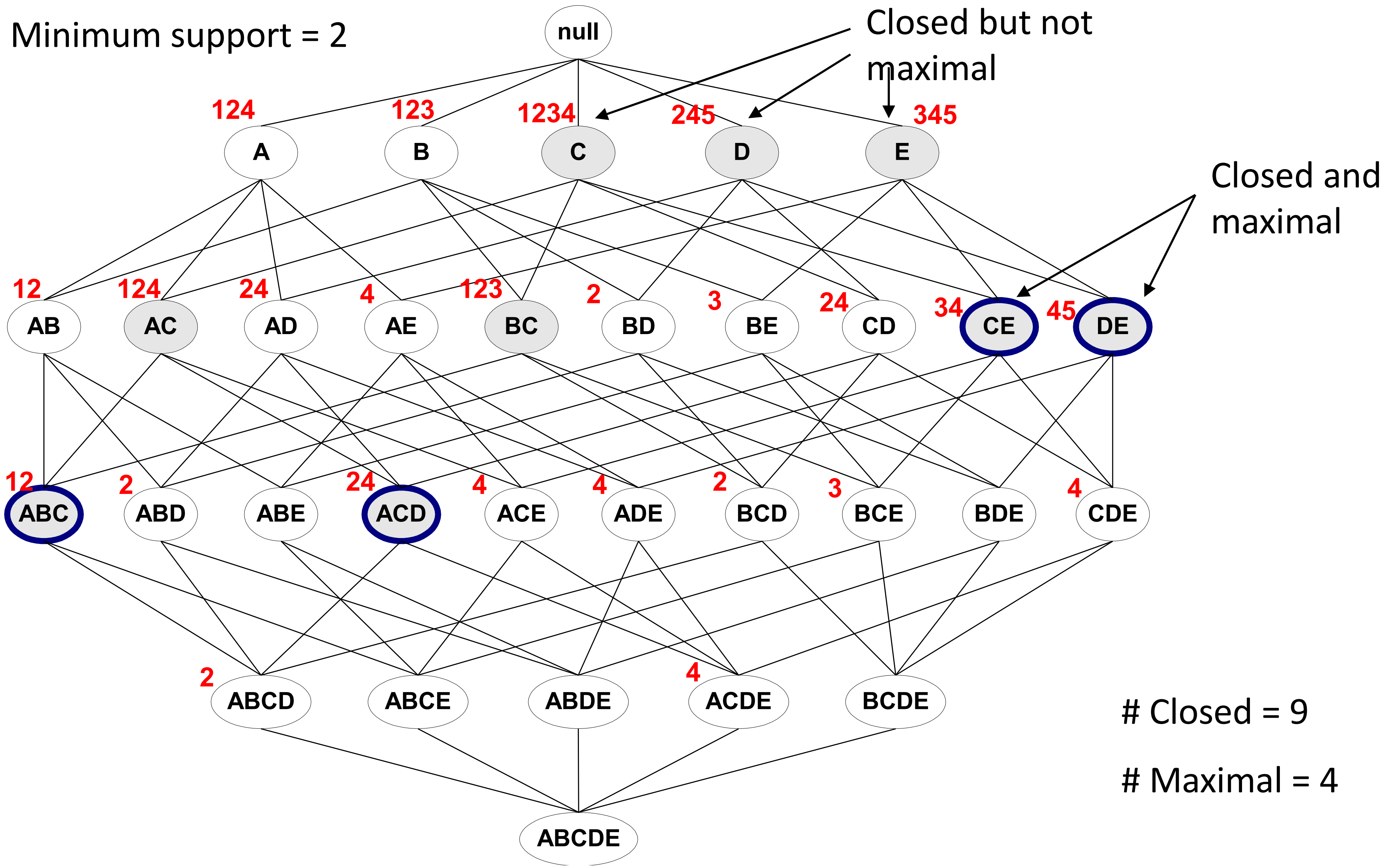
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Maximal vs. Closed Itemsets

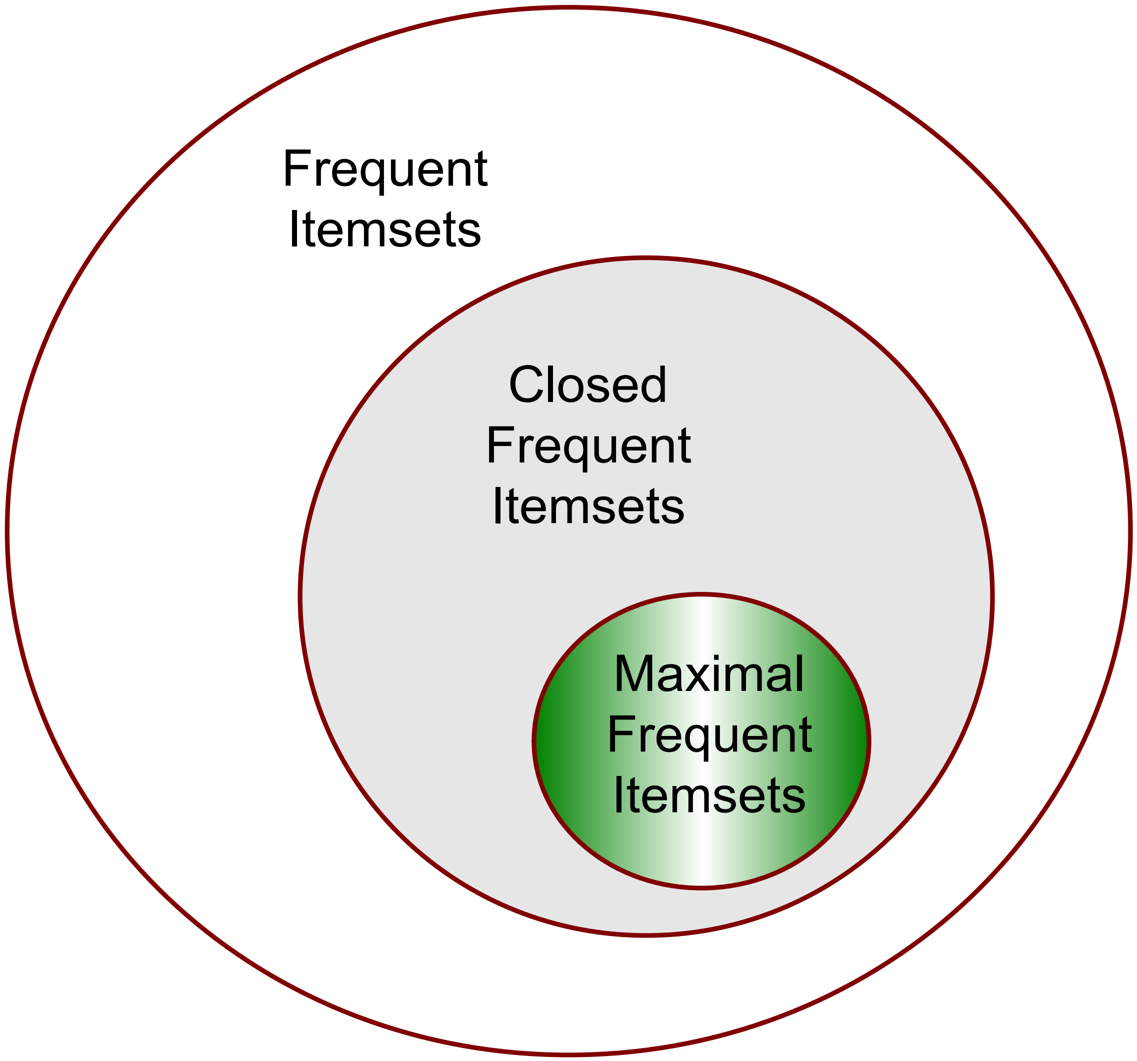
TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Maximal vs. Closed Frequent Itemsets



Maximal vs. Closed Itemsets

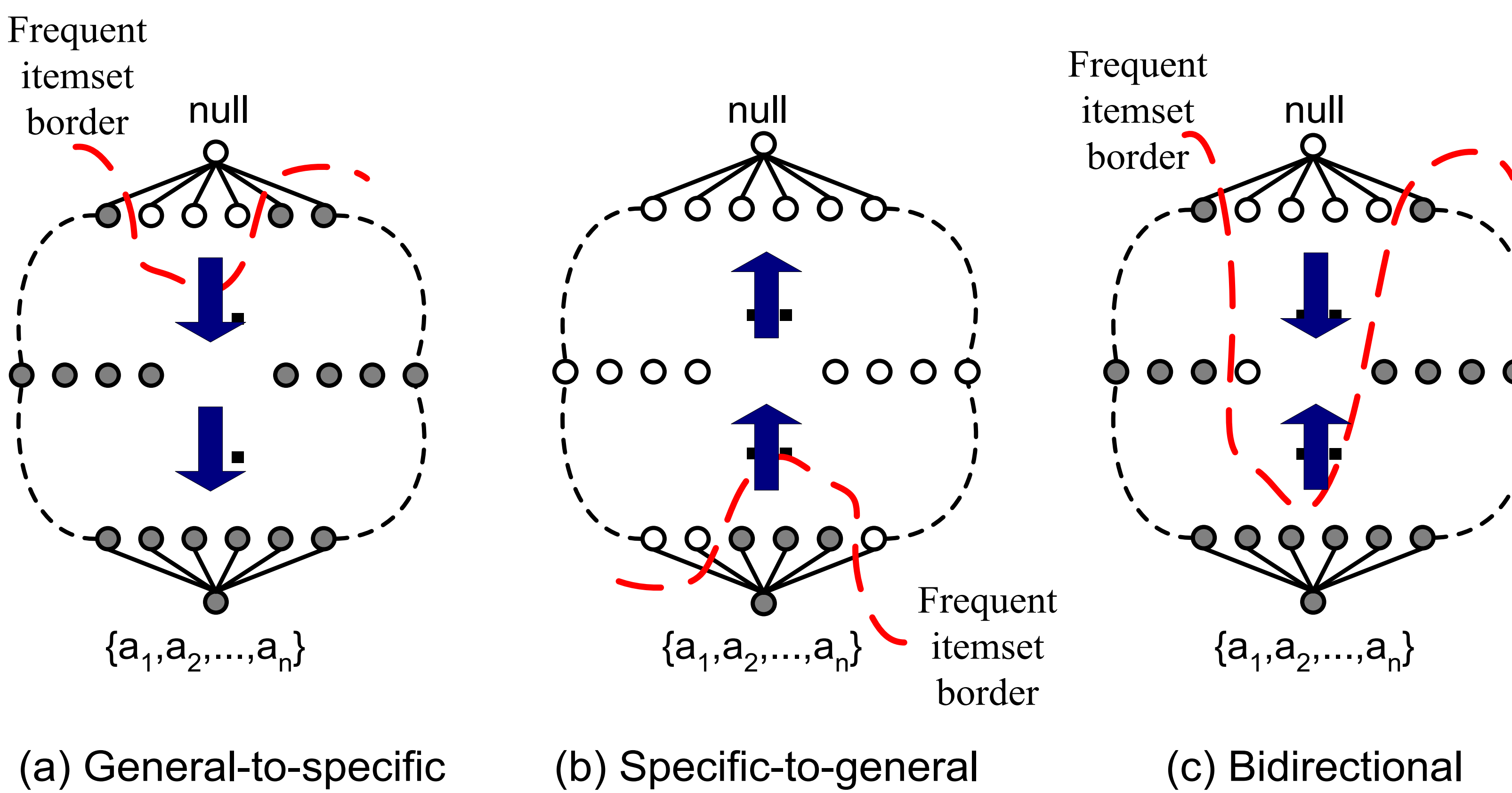


Maximal vs. Closed Itemsets

- Closed itemsets allow simpler computation of the support of their subsets than maximal itemsets
  - Ex. support of (AB) must be the same as one of its supersets: (ABC), (ABD), (ABE)
  - But,
    - s(AB) >= s(ABC),
    - s(AB) >= s(ABD), and
    - s(AB) >= s(ABE)
  - Implies,
    - s(AB) = max(s(ABC), s(ABD), s(ABE))

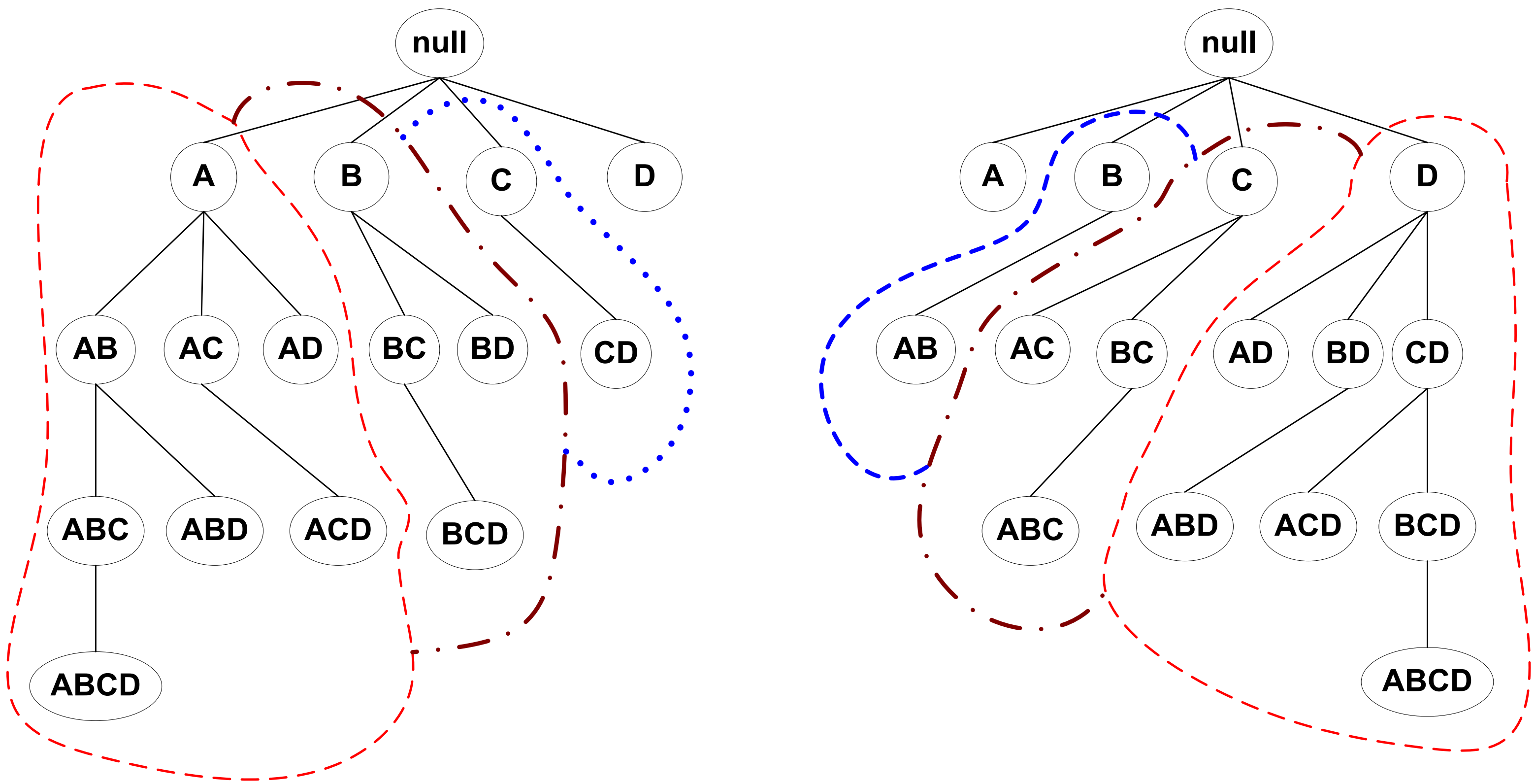
Alternative Method: Frequent Itemset Gen.

- Traversal of Itemset Lattice
  - general-to-specific vs. specific-to-general



Alternative Method: Frequent Itemset Gen.

- Traversal of Itemset Lattice
  - Equivalent Classes

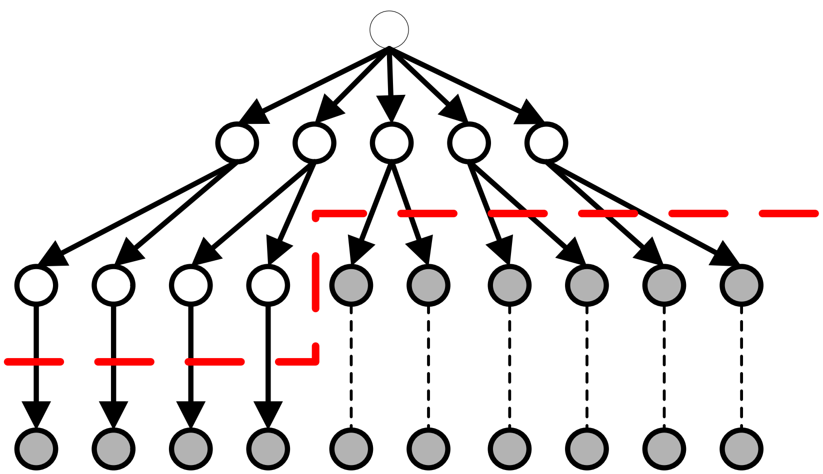


(a) Prefix tree

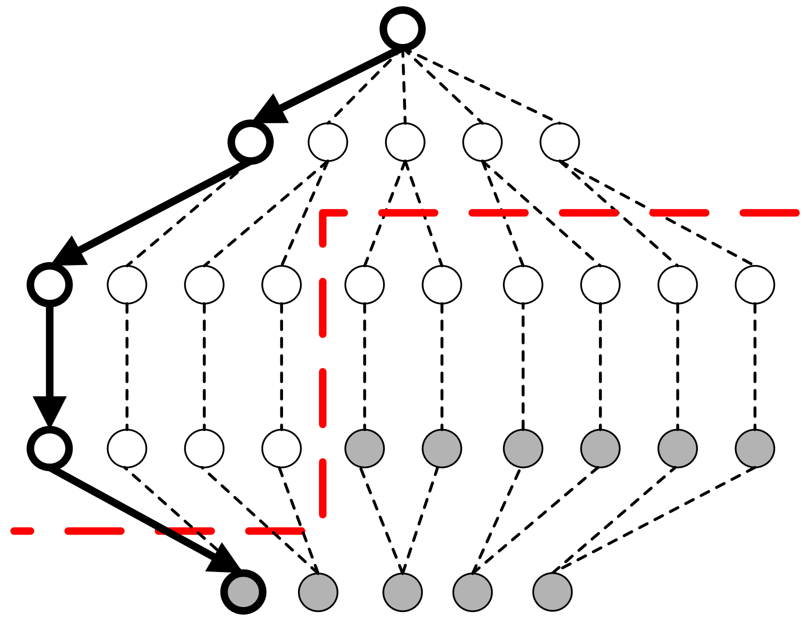
(b) Suffix tree

Alternative Method: Frequent Itemset Gen.

- Traversal of Itemset Lattice
  - Breadth-first vs. Depth-first



(a) Breadth first



(b) Depth first

Alternative Methods: Frequent Itemset Gen.

- Representation of Database
  - horizontal vs. vertical data layout

Horizontal Data Layout

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B

Vertical Data Layout

A	B	C	D	E
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

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 computationally expensive

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ 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	BDC -> A
D -> ABC	C -> ABD	B -> ACD	A -> ABC
AB -> CD	AC -> BD	AD -> BC	BC -> AD
BD -> AC	CD -> AB		

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

Efficient Rule Generation

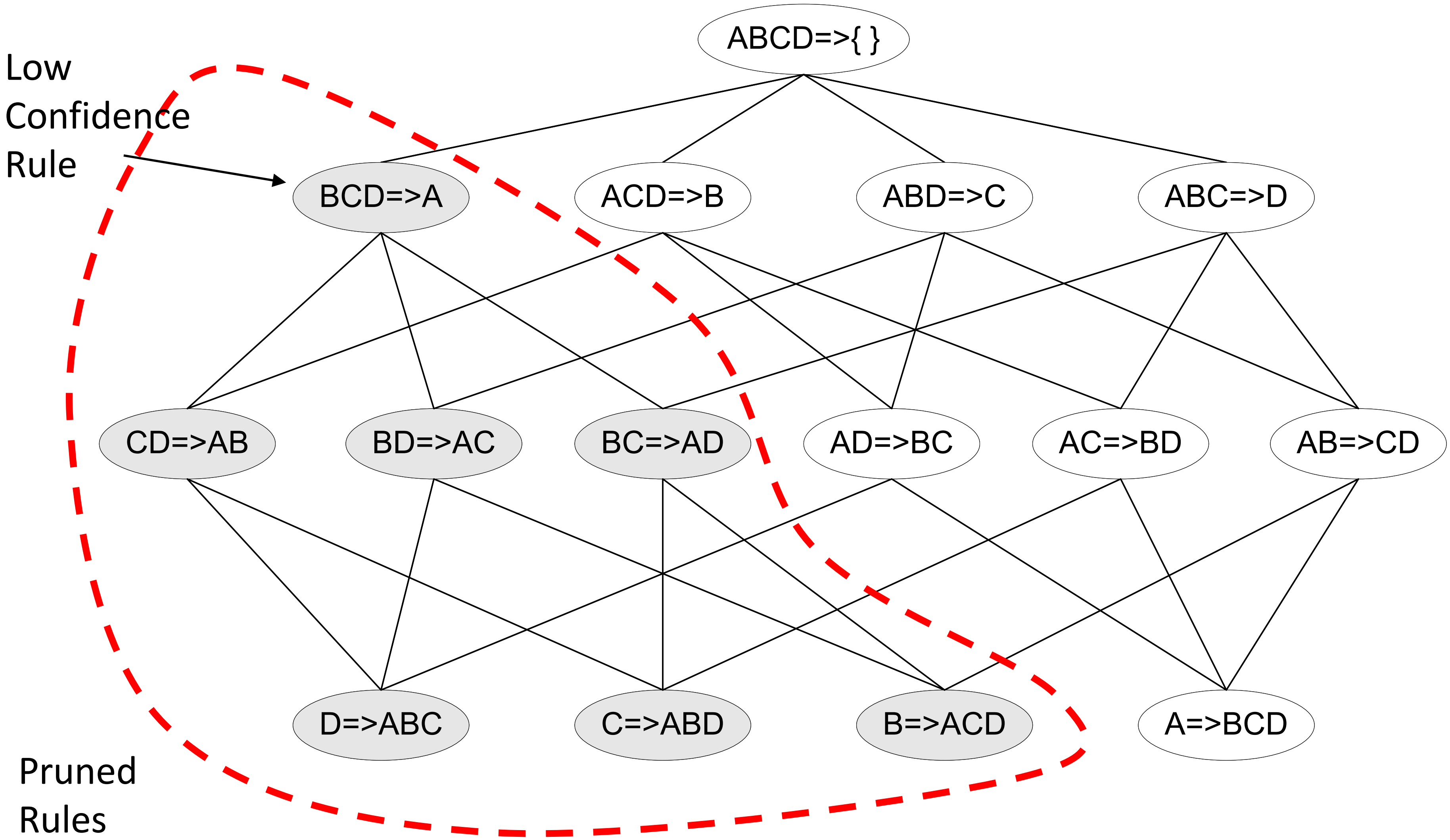
- How to efficiently generate rules from frequent analysis?
  - In general, confidence does not have an anti-monotone property

    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 -> D) ≥ c(AB -> CD) ≥ c(A -> BCD)
    - confidence is anti-monotone w.r.t. inclusion on the RHS of the rule

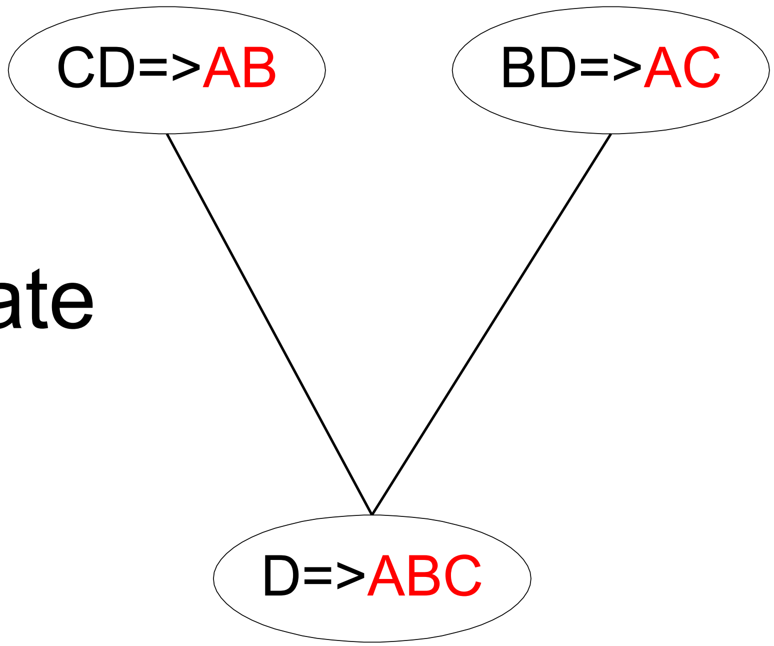
Rule Generation

Lattice of rules



Rule Generation

- A 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

Example: Rule Generation

- L = { {I1}, {I2}, {I3}, {I4}, {I5}, {I1, I2}, {I1, I3}, {I1, I5}, {I2, I3}, {I2, I4}, {I2, I5}, {I1, I2, I3}, {I1, I2, I5} }
  - Look at {I1, I2, I5}
- *minconf* is 70%
  - R1: I1 ^ I2 -> I5

$$conf = \frac{s(\{\mathrm{I1,I2,I5}\})}{s(\{\mathrm{I1,I2}\})} = \frac{2}{4} = 50\%$$

  - R2: I1 ^ I5 -> I2

$$conf = \frac{s(\{\mathrm{I1,I2,I5}\})}{s(\{\mathrm{I1,I5}\})} = \frac{2}{2} = 100\%$$

  - R3: I2 ^ I5 -> I1

$$conf = \frac{s(\{\mathrm{I1,I2,I5}\})}{s(\{\mathrm{I2,I5}\})} = \frac{2}{2} = 100\%$$

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2 ,I3, I5
T100	I1, I2, I3

Example: Rule Generation

- R4: I1 -> I2 ^ I5

$$conf = \frac{s(\{\mathrm{I1,I2,I5}\})}{s(\{\mathrm{I1}\})} = \frac{2}{6} = 33\%$$

- R5: I2 -> I1 ^ I5

$$conf = \frac{s(\{\mathrm{I1,I2,I5}\})}{s(\{\mathrm{I2}\})} = \frac{2}{7} = 29\%$$

- R6: I5 -> I1 ^ I2

$$conf = \frac{s(\{\mathrm{I1,I2,I5}\})}{s(\{\mathrm{I5}\})} = \frac{2}{2} = 100\%$$

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2 ,I3, I5
T100	I1, I2, I3

- R2, R3, and R6 are selected

Extensions to Apriori: Improve Efficiency

- Partition DB, find local frequent patterns, consolidate to global patterns
  - Savasere, Omiecinski, and Navathe, VLDB, 1995
- Reduce number of candidates with DHP
  - Park, Chen, and Yu, SIGMOD, 1995
- Sampling for frequent patterns, verify pattern in db
  - Toivonen, VLDB, 1996.
- Dynamic Itemset counting (DIC)
  - Brin, Motwani, Ullman, Tsur, SIGMOD, 1997

Mining Frequent Patterns w/o Cand. Gen.

- Bottlenecks of Apriori
  - breadth-first (i.e., level-wise) search
  - candidate generation and test
    - may generate huge number of candidates
- FPGrowth Approach (Han, Pei, Yin SIGMOD, 2000)
  - depth-first search
  - avoid explicit candidate generation
- Main Idea – grow long patterns from short ones using local frequent items only
  - “abc” is a frequent pattern
  - get all trans. with “abc”, project DB on abc: DB | abc
  - “d” is local frequent item in DB | abc, then abcd is freq. pattern