CSC-422/522: ALDA
M/W. 8.30-9.45am. HL-Auditorium.

Ranga Raju Vatsavai
Chancellors Faculty Excellence Associate Professor in Geospatial Analytics
Department of Computer Science, North Carolina State University (NCSU)
Associate Director, Center for Geospatial Analytics, NCSU
&
Joint Faculty, Oak Ridge National Laboratory (ORNL)

Logistics

- HW4
  - Any questions?

NC STATE UNIVERSITY

Today

- Association Rules

NC STATE UNIVERSITY

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:

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

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

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

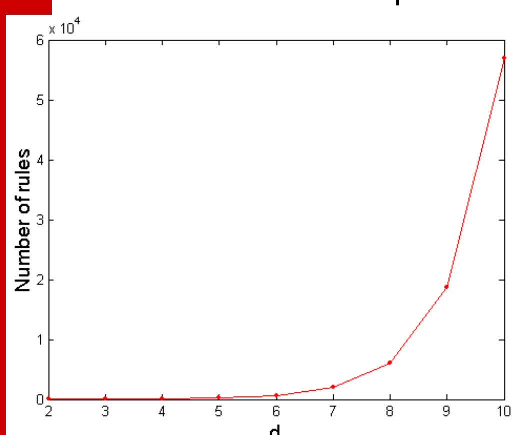
Association Rule Mining Task

- 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!

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

Amazon:
- Books ~ 2 million
- MP3 ~ 6 million

NC STATE UNIVERSITY

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

NC STATE UNIVERSITY

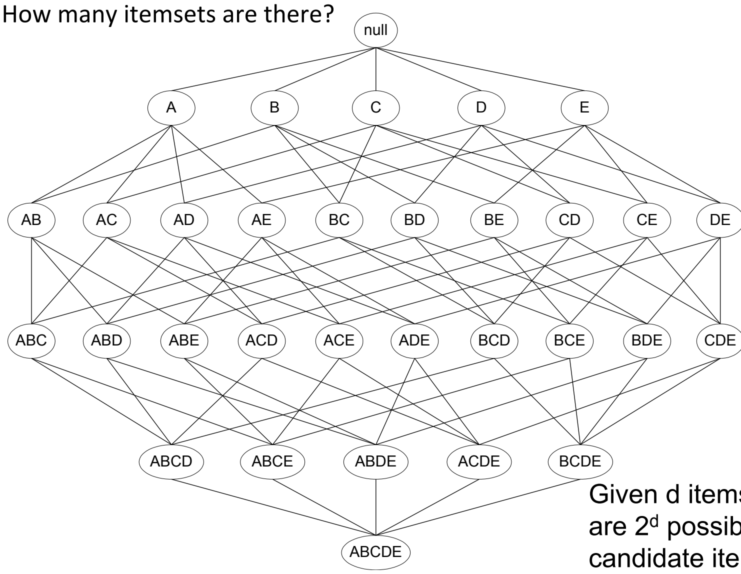
Mining Association Rules

- 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

NC STATE UNIVERSITY

Frequent Itemset Generation

How many itemsets are there?



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

NC STATE UNIVERSITY

When is this useful?

- If minsup = 0, then all subsets are frequent (2^d-1). That is, summary may be larger than input.
- If **minsup** is large, then summary (associate rules) is very small and may be interesting and potentially valuable
  - E.g., if minsup = 20% and minconf = 50%, then most likely 80% rules are discarded.
  - Somehow we should eliminate computing support/ confidence for large fraction of itemsets.

NC STATE UNIVERSITY

Frequent Itemset Generation

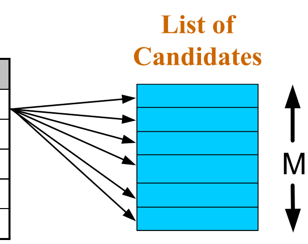
- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

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 since $M = 2^d$!!!

NC STATE UNIVERSITY

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

NC STATE UNIVERSITY

Reducing Number of Candidates

- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent
- Apriori principle holds due to the following property of the support measure:

$$\forall X,Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y) \quad \text{(Anti-Monotone)}$$

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

NC STATE UNIVERSITY

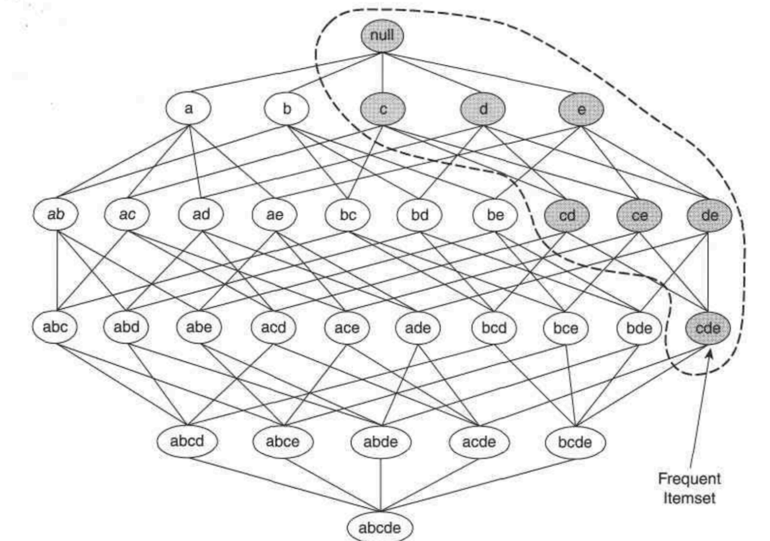
Example

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**

S(Bread) > S(Bread, Milk) [4/5 > 3/5]
S(Milk) > S(Bread, Milk)
….

NC STATE UNIVERSITY

Illustrating Apriori Principle

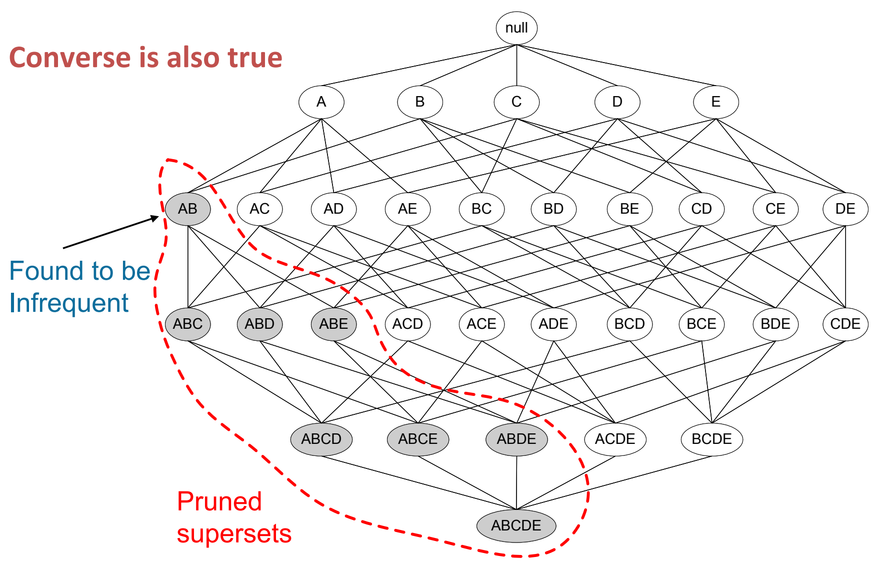


For e.g., take itemset {c,d,e}, then any transaction that contains {c,d,e} must also contain its subsets: {c,d}, {c,e}, {d,e}, {c}, {d}, {e}. Therefore, {c,d,e} is frequent then all of its subsets must also be frequent.

NC STATE UNIVERSITY

Illustrating Apriori Principle

Converse is also true



NC STATE UNIVERSITY

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

NC STATE UNIVERSITY

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

NC STATE UNIVERSITY

Illustrating Apriori Principle

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

Items (1-itemsets)

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

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

NC STATE UNIVERSITY

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
{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

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

NC STATE UNIVERSITY

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

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

NC STATE UNIVERSITY

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
{ Beer, Diaper, Milk}	2
{ Beer,Bread, Diaper}	2
{Bread, Diaper, Milk}	2
{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 + 0 = 12

NC STATE UNIVERSITY

Apriori Algorithm

- F_k: frequent k-itemsets
- L_k: candidate k-itemsets

- Algorithm
  - Let k=1
  - Generate F_1 = {frequent 1-itemsets}
  - Repeat until F_k is empty
    - **Candidate Generation**: Generate L_{k+1} from F_k
    - **Candidate Pruning**: Prune candidate itemsets in L_{k+1} containing subsets of length k that are infrequent
    - **Support Counting**: Count the support of each candidate in L_{k+1} by scanning the DB
    - **Candidate Elimination**: Eliminate candidates in L_{k+1} that are infrequent, leaving only those that are frequent => F_{k+1}

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

NC STATE UNIVERSITY

Candidate Generation: Brute-force method

Items

Item
Beer
Bread
Cola
Diapers
Milk
Eggs

→

Candidate Generation

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

→

Candidate Pruning

Itemset
{Bread, Diapers, Milk}

Candidate Generation: Merge F_{k-1} and F_1 itemsets

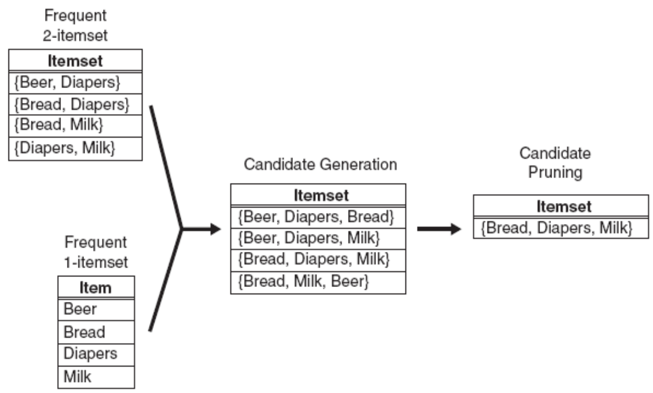


Figure 6.7. Generating and pruning candidate k-itemsets by merging a frequent $(k-1)$-itemset with a frequent item. Note that some of the candidates are unnecessary because their subsets are infrequent.

Candidate Generation: F_{k-1} x 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

NC STATE UNIVERSITY

Candidate Generation: F_{k-1} and F_{k-1} method

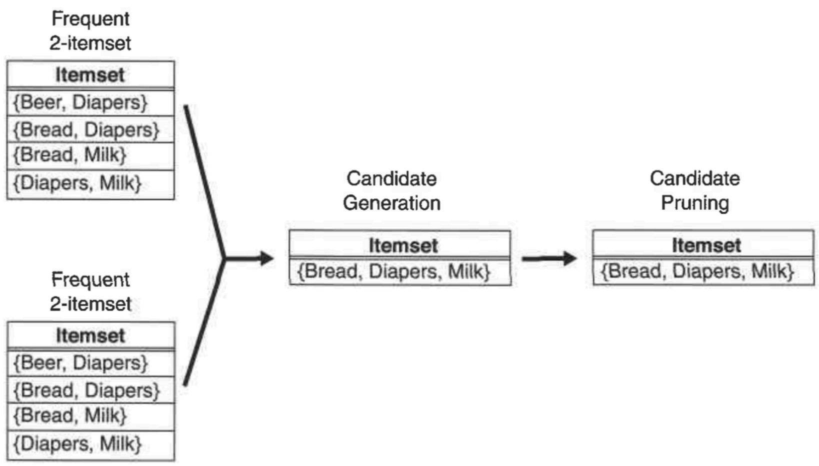


Figure 6.8. Generating and pruning candidate k-itemsets by merging pairs of frequent $(k-1)$-itemsets.

NC STATE UNIVERSITY

F_{k-1} and F_{k-1} method: Self-Join

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

  insert into $\mathbf{C_{k+1}}$

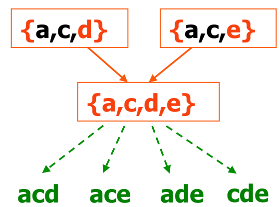
  select $\mathbf{p.item_1, p.item_2, \ldots, p.item_k, q.item_k}$

  from $\mathbf{L_k\ p, L_k\ q}$

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

Self-Join Example

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



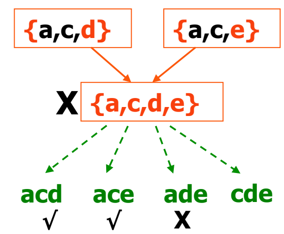
Candidate Pruning

- 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
- Candidate pruning
  - Prune ABCE because ACE and BCE are infrequent
  - Prune ABDE because ADE is infrequent
- After candidate pruning: L_4 = {ABCD}

NC STATE UNIVERSITY

Candidate Pruning

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



NC STATE UNIVERSITY

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 + 0 = 12

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.

NC STATE UNIVERSITY

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}

NC STATE UNIVERSITY

Support Counting of Candidate Itemsets

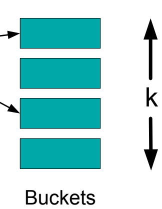
- To reduce number of comparisons, store the candidate itemsets in a hash structure
  - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets

Transactions

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



- Leaf node contains list of itemsets and counts
- Interior node contains a hash table
- Subset-function: finds all candidates contained in a transcation

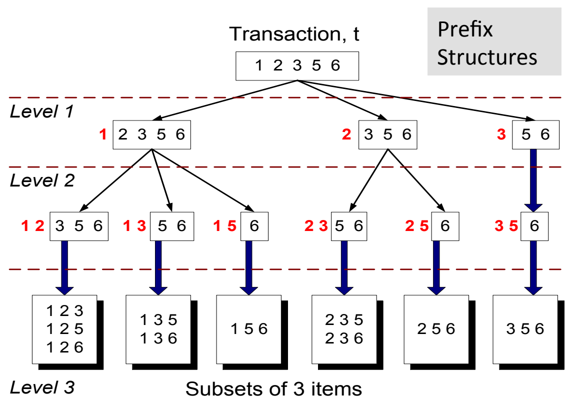
NC STATE UNIVERSITY

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)?



Still need to match with candidate itemsets. Increment support if there is a match.

NC STATE UNIVERSITY

Support Counting: An Example

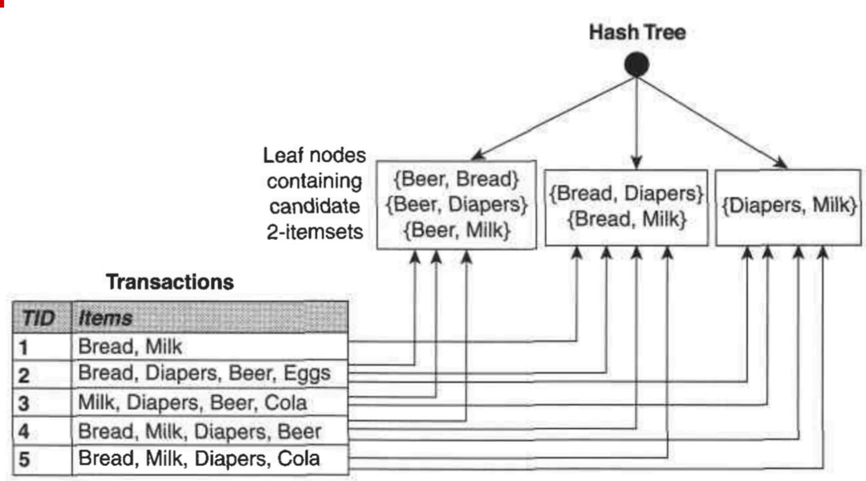


Figure 6.10. Counting the support of itemsets using hash structure.

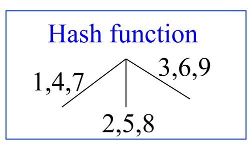
Support Counting Using a Hash Tree

Suppose you have 15 candidate itemsets of length 3:

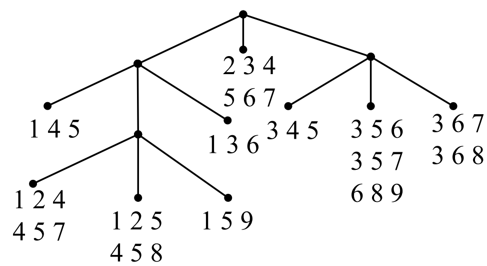
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

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)

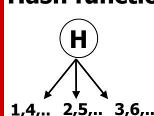


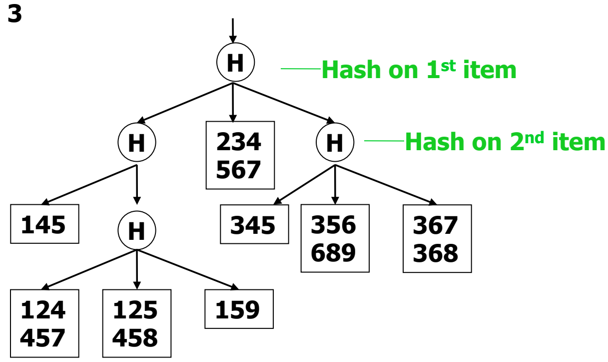
Mod 3



Example of the hash-tree for C_3

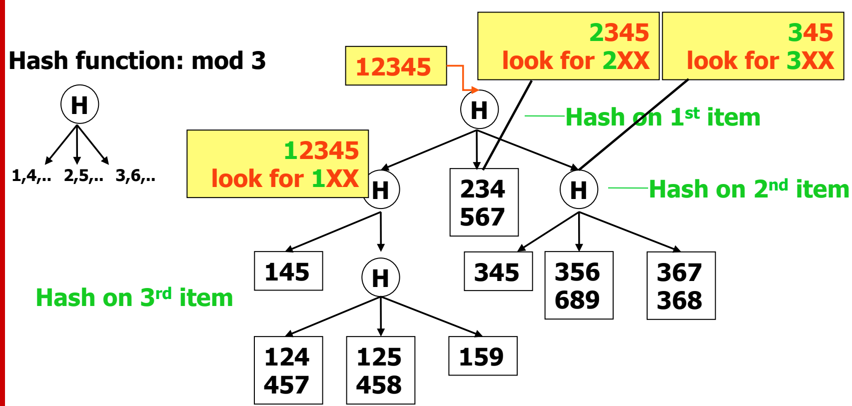
Hash function: mod 3





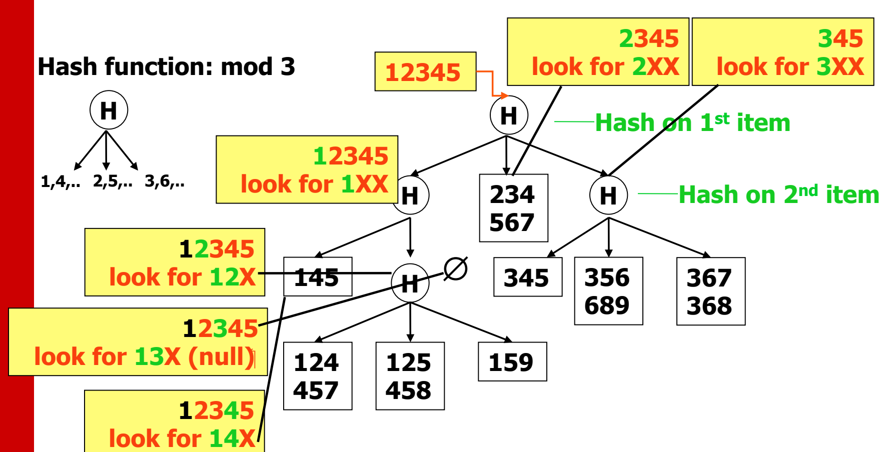
NC STATE UNIVERSITY

Example of the hash-tree for C_3



NC STATE UNIVERSITY

Example of the hash-tree for C_3



The subset-function finds all the candidates contained in a transaction:
- At the root level it hashes on all items in the transaction
- At level i it hashes on all items in the transaction that come after item the i-th item

NC STATE UNIVERSITY

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 $L \rightarrow \emptyset$ and $\emptyset \rightarrow L$)

NC STATE UNIVERSITY

Rule Generation

- 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., 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