Association rules
Part 1

Mohammed Brahimi & Sami Belkacem

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, Juice, Eggs
3	Milk, Diaper, Juice, Coke
4	Bread, Milk, Diaper, Juice
5	Bread, Milk, Diaper, Coke

Example of Association Rules

{Diaper} → {Juice},
{Milk, Bread} → {Eggs, Coke},
{Juice, Bread} → {Milk},

Implication means co-occurrence, not causality!

Binary representation

- Market basket data can be represented in a binary format.
- A row corresponds to a transaction and a column corresponds to an item.
- An item is one if the item is present in a transaction and zero otherwise.
- An item is represented using a binary asymmetric variable.

TID	Bread	Milk	Diapers	Juice	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

Definition: Frequent Itemset

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

- **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 (s)**
  - 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

Definition: Association Rule

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

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Juice, Eggs
3	Milk, Diaper, Juice, Coke
4	Bread, Milk, Diaper, Juice
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:

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

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

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

Association Rule Mining Task

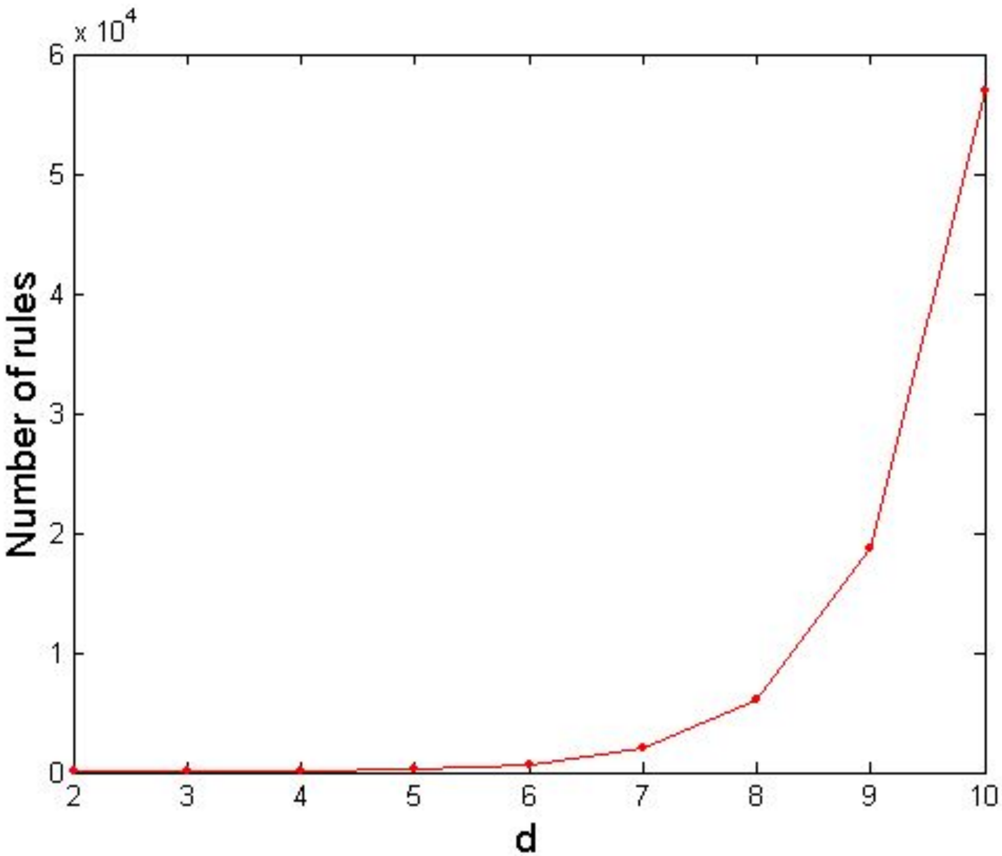
- Given a set of transactions ***T***, the goal of association rule mining is to find all rules having
  - support ≥ ***minsup*** threshold
  - confidence ≥ ***minconf*** threshold
  - **support** computes <u>frequency</u> of a rule and **confidence** its <u>reliability</u>

- 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

Mining Association Rules

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

Example of Rules:

{Milk,Diaper} → {Juice} (s=0.4, c=0.67)
{Milk,Juice} → {Diaper} (s=0.4, c=1.0)
{Diaper,Juice} → {Milk} (s=0.4, c=0.67)
{Juice} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Juice} (s=0.4, c=0.5)
{Milk} → {Diaper,Juice} (s=0.4, c=0.5)

Observations:

- All the rules are binary partitions of the same itemset: {Milk, Diaper, Juice}
- Rules originating from the same itemset have identical support but can have different confidence.
- If the itemset is infrequent, then all six candidate rules can be pruned immediately without having to compute their confidence values.

Mining Association Rules

- Two-step approach:

    1. **Frequent Itemset Generation**

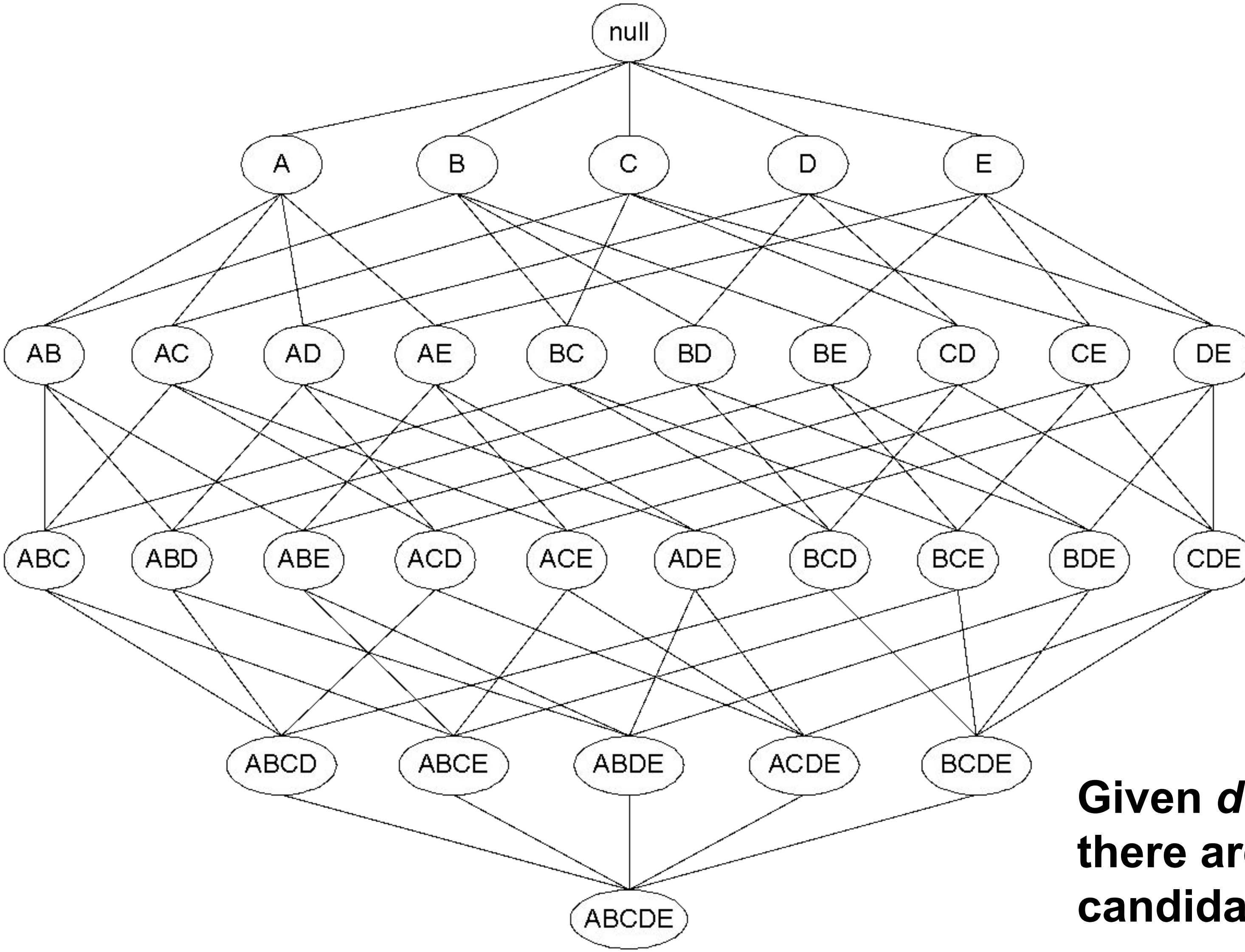
       First generate all itemsets whose support ≥ ***minsup***

    2. **Rule Generation**

       Then 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

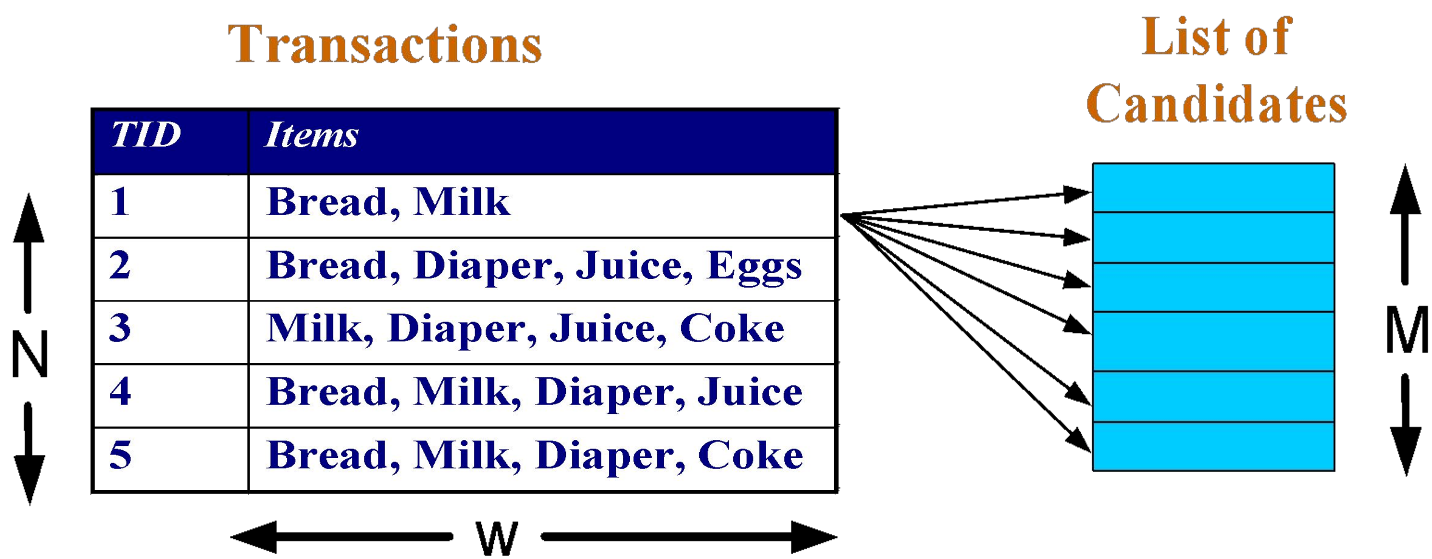
Frequent Itemset Generation: Lattice with 5 items



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, Juice, Eggs
3	Milk, Diaper, Juice, Coke
4	Bread, Milk, Diaper, Juice
5	Bread, Milk, Diaper, Coke

  - Match each transaction against every candidate in the lattice
  - **N**: # of transactions, **M:** # of candidate itemsets, **w:** maximum transaction width
  - Complexity ~ O(NMw) => Expensive since M = 2^d !!!

Frequent Itemset Generation Strategies

- Reduce the number of candidates (**M**)
  - Complete search: $\mathbf{M}=2^d$
  - 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
  - E.g. a hash structure
  - No need to match every candidate against every transaction

Reducing Number of Candidates

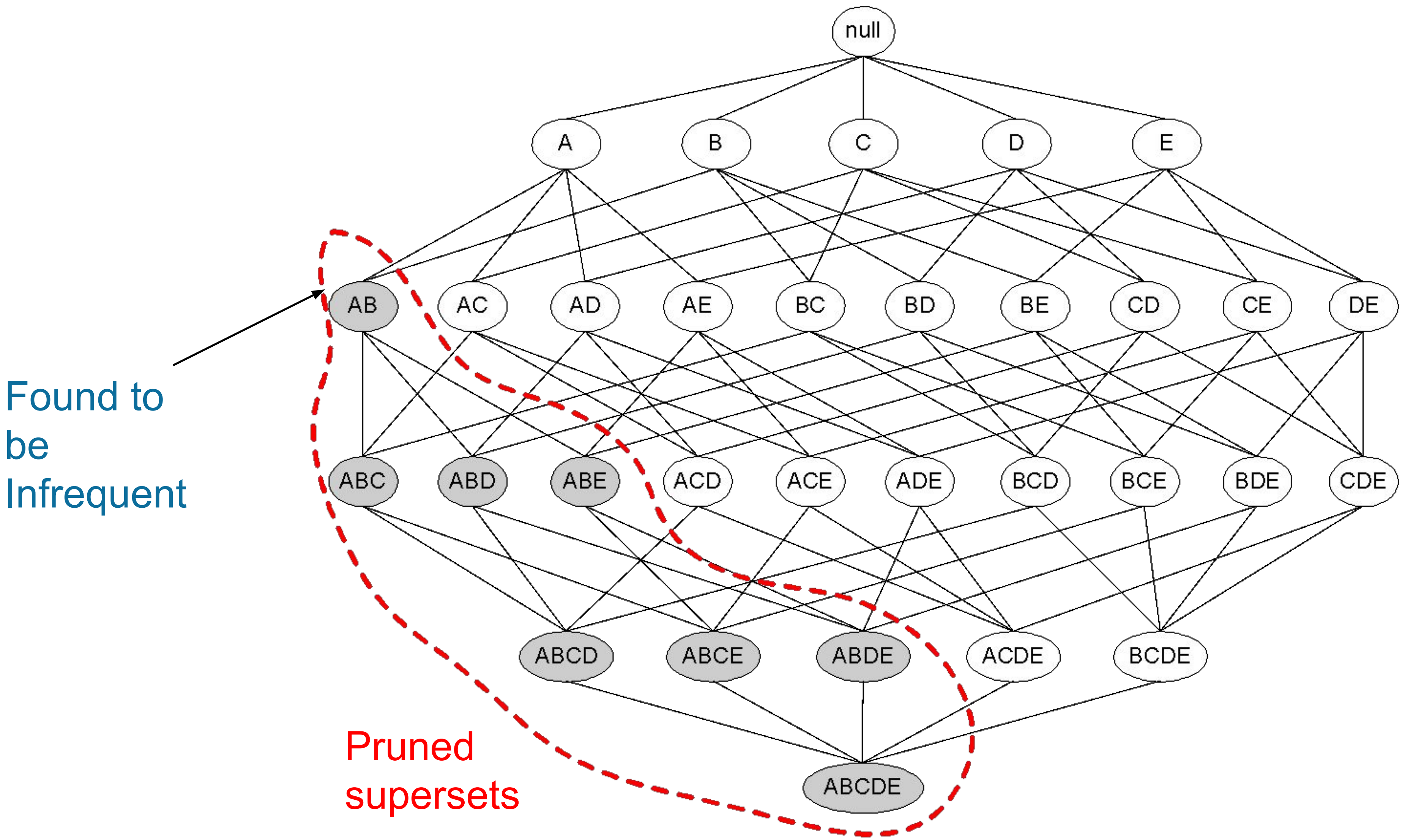
Apriori principle:

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

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

- Apriori principle holds due to the following property of the support measure:
  - Support of an itemset never exceeds the support of its subsets
- If a subset of an itemset is infrequent, then this itemset is infrequent
- If an itemset is infrequent, then all itemsets that include this infrequent itemset should be infrequent

Illustrating Apriori Principle



Illustrating Apriori Principle

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

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Juice	**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	**Juice, Bread, Diaper, Eggs**
3	**Juice, Coke, Diaper, Milk**
4	**Juice, Bread, Diaper, Milk**
5	**Bread, Coke, Diaper, Milk**

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Juice	**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

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

Items (1-itemsets)

Itemset
{Bread,Milk}
{Juice, Bread}
{Bread,Diaper}
{Juice,Milk}
{Diaper,Milk}
{Juice,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,

$6 + 6 + 4 = 16$

Illustrating Apriori Principle

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

Items (1-itemsets)

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

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

Items (1-itemsets)

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

$6 + 6 + 4 = 16$

Triplets (3-itemsets)

Itemset
{Juice, Diaper, Milk}
{Juice, Bread, Diaper}
{Bread, Diaper, Milk}
{Juice, Bread, Milk}

Illustrating Apriori Principle

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

Items (1-itemsets)

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

$6 + 6 + 4 = 16$

Triplets (3-itemsets)

Itemset	Count
{Juice, Diaper, Milk}	**2**
{Juice, Bread, Diaper}	**2**
{Bread, Diaper, Milk}	**2**
{Juice, Bread, Milk}	**1**

Illustrating Apriori Principle

Items (1-itemsets)

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

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

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

Minimum Support = 3

Triplets (3-itemsets)

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

F_k: frequent k-itemsets

L_k: candidate k-itemsets

- 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}