Data Mining

Chapter 5

Association Analysis: Basic Concepts

Introduction to Data Mining, 2nd Edition

by

Tan, Steinbach, Karpatne, Kumar

Association Analysis

- Tarefa descritiva, não supervisionada, interpretável
- **Association analysis:** useful for discovering interesting relationships hidden in large data sets
- The uncovered relationships can be represented in the form of sets of items, which are known as **frequent itemsets**, or **association rules**, that represent relationships between two itemsets

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

Interpretáveis

Example of Association Rules

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

Implication means co-occurrence, not causality!

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

Implication means co-occurrence, not causality!

Causality requires knowledge about which attributes in the data capture cause and effect, and typically involves relationships occurring over time (e.g., greenhouse gas emissions lead to global warming) [emissões de gases de efeito estufa levam ao aquecimento global]

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) [Sup. absoluto]**
  – Frequency of occurrence of an itemset
  – E.g. σ({Milk, Bread, Diaper}) = 2
- **Support [Sup. relativo]**
  – Fraction of transactions that contain an itemset
  – E.g. s({Milk, Bread, Diaper}) = 2/5 = 40%
- **Frequent Itemset**
  – An itemset whose support is greater than or equal to a *minsup* threshold [relativo ou absoluto, a depender da implementação]

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 $X \rightarrow Y$, where X and Y are itemsets ($X \cap Y = \emptyset$; X=LHS, Y=RHS)
  - 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:

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

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

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

Association Rule Mining Task

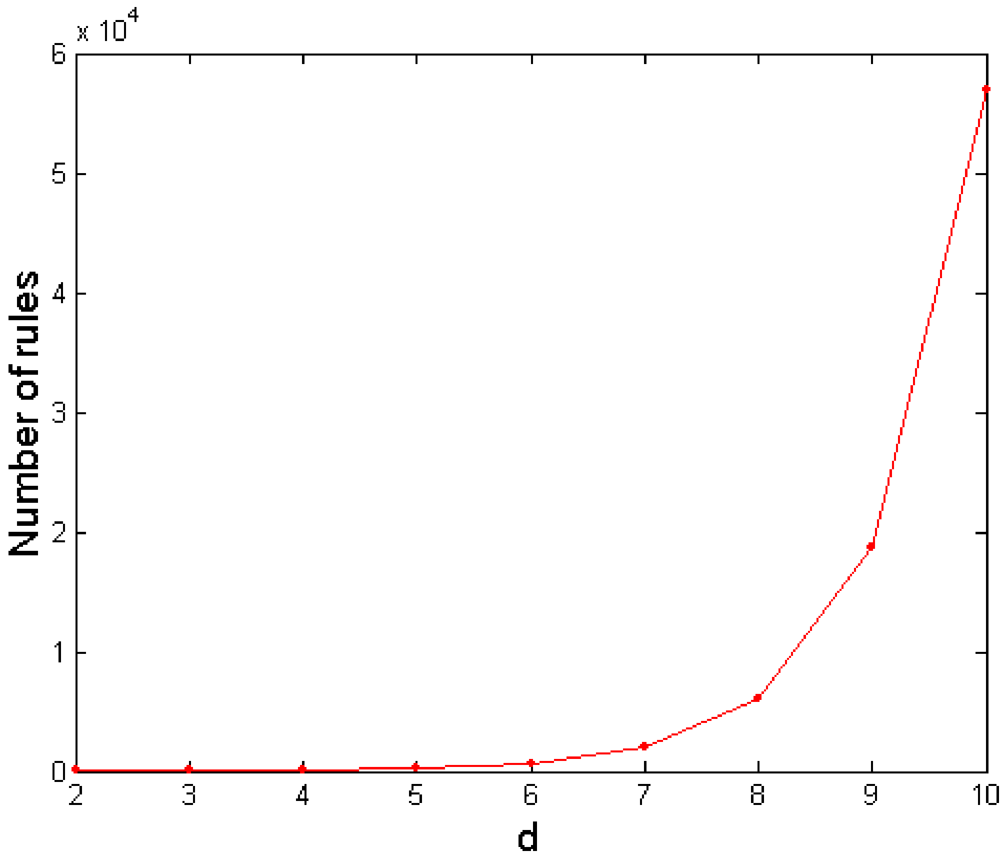
- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- 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

  $\Rightarrow$ 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, 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)

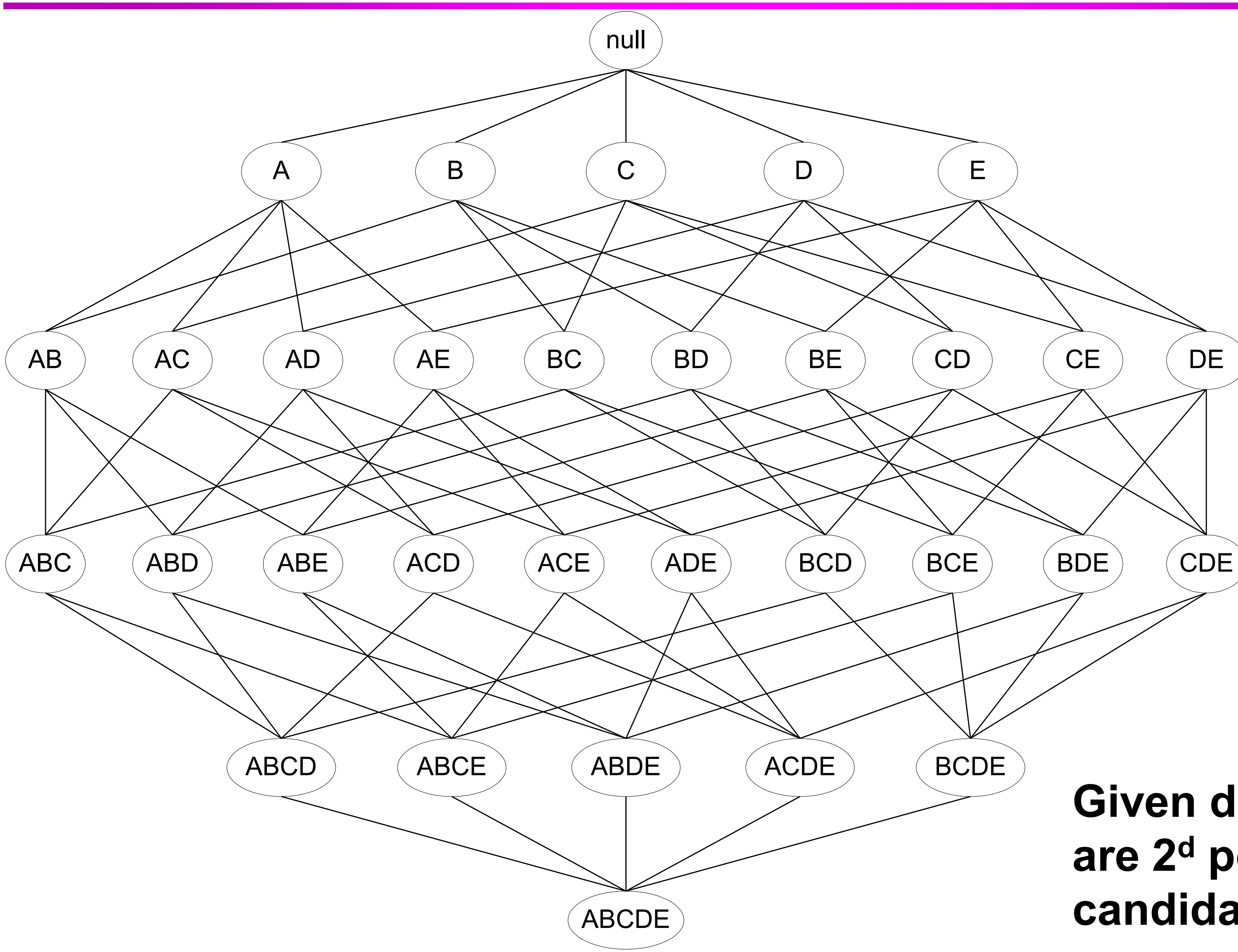
Observations:

- 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
[propriedade anti-monotônica do suporte]

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     – Generate all itemsets whose support $\geq$ 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

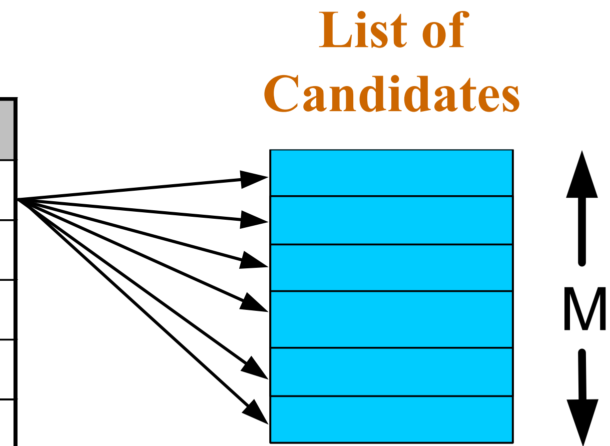
Transactions

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

w

Maximum transaction width



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

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M) [Apriori]
  – Complete search: $M=2^d$
  – Use pruning techniques to reduce M

- 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

- Reduce the number of transactions (N) [próximo]
  – Reduce size of N as the size of itemset increases
  – Used by DHP and vertical-based mining algorithms

Frequent Itemset Generation Strategies

- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
    - As the size of candidate itemsets increases, fewer transactions will be supported by the itemsets
    - For instance, since the width of the first transaction in the previous Table is 2, it would be advantageous to remove this transaction before searching for frequent itemsets of size 3 and larger
  - Used by DHP and vertical-based mining algorithms

Reducing Number of Candidates

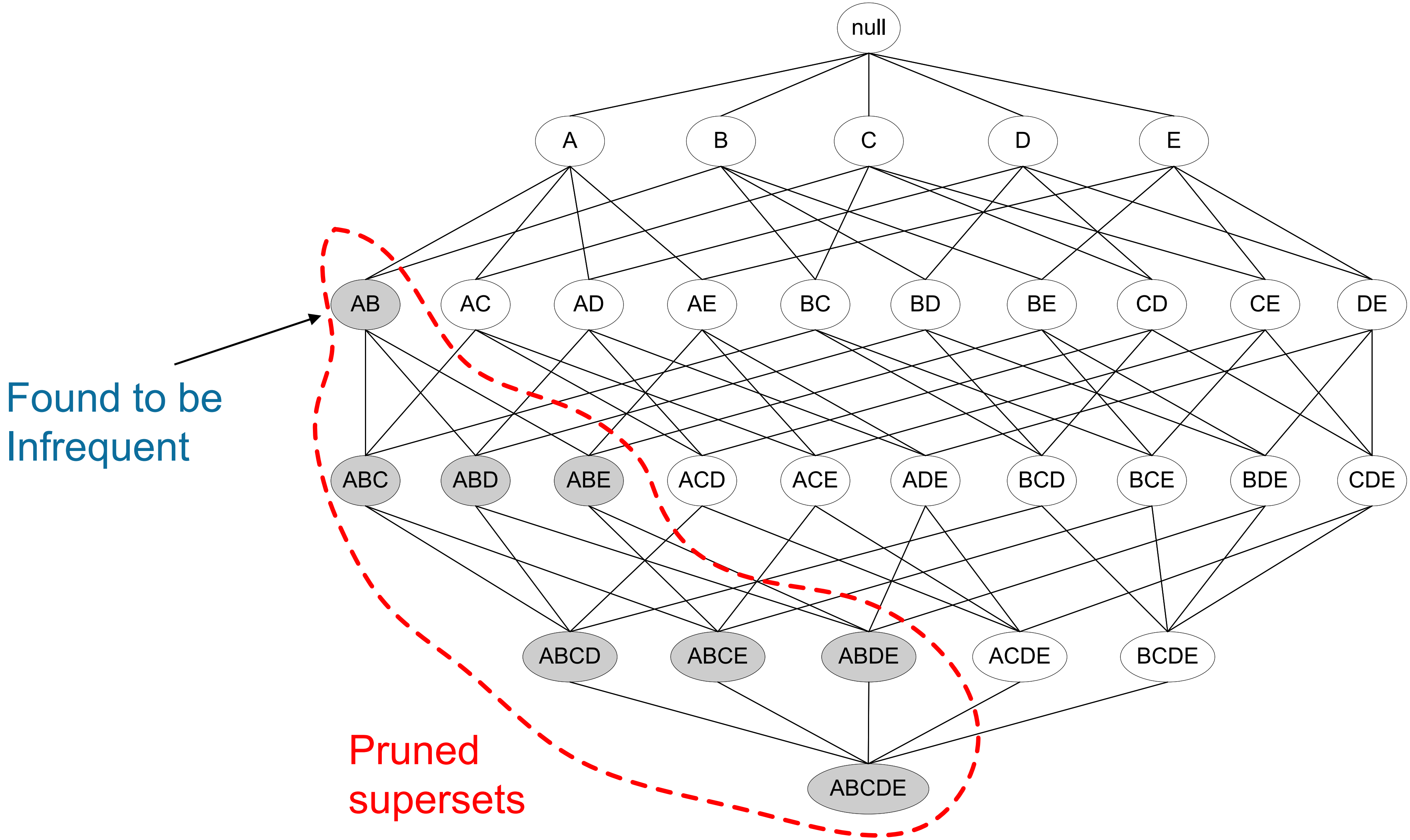
- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent

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

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

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

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,

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

$$6 + 15 + 20 = 41$$

Combinação de 6 itens 1 a 1, 6 itens 2 a 2 e 6 itens 3 a 3

With support-based pruning,

$${}^6C_1 + {}^4C_2 + {}^4C_3$$

$$6 + 6 + 4 = 16$$

Redução de 68%

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,

$$^6C_1 + {}^4C_2 + {}^4C_3$$

$$6 + 6 + 4 = 16$$

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,

$6 + 6 + 4 = 16$

$6 + 6 + 1 = 13$

Triplets (3-itemsets)

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

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}

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}

Observação

- There are many ways to generate candidate itemsets. An effective candidate generation procedure must be **complete** and **non-redundant**
  - Complete: if it does not omit any frequent itemsets
  - Non-redundant: if it does not generate the same candidate itemset more than once
    - Generation of duplicate candidates leads to wasted computations and thus should be avoided for efficiency reasons
- In addition:
  - An effective candidate generation procedure should avoid generating too many **unnecessary** candidates
    - Unnecessary: if at least one of its subsets is infrequent, and thus, eliminated in the candidate pruning step

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}

→

Frequent 2-itemset

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

Candidate Pruning

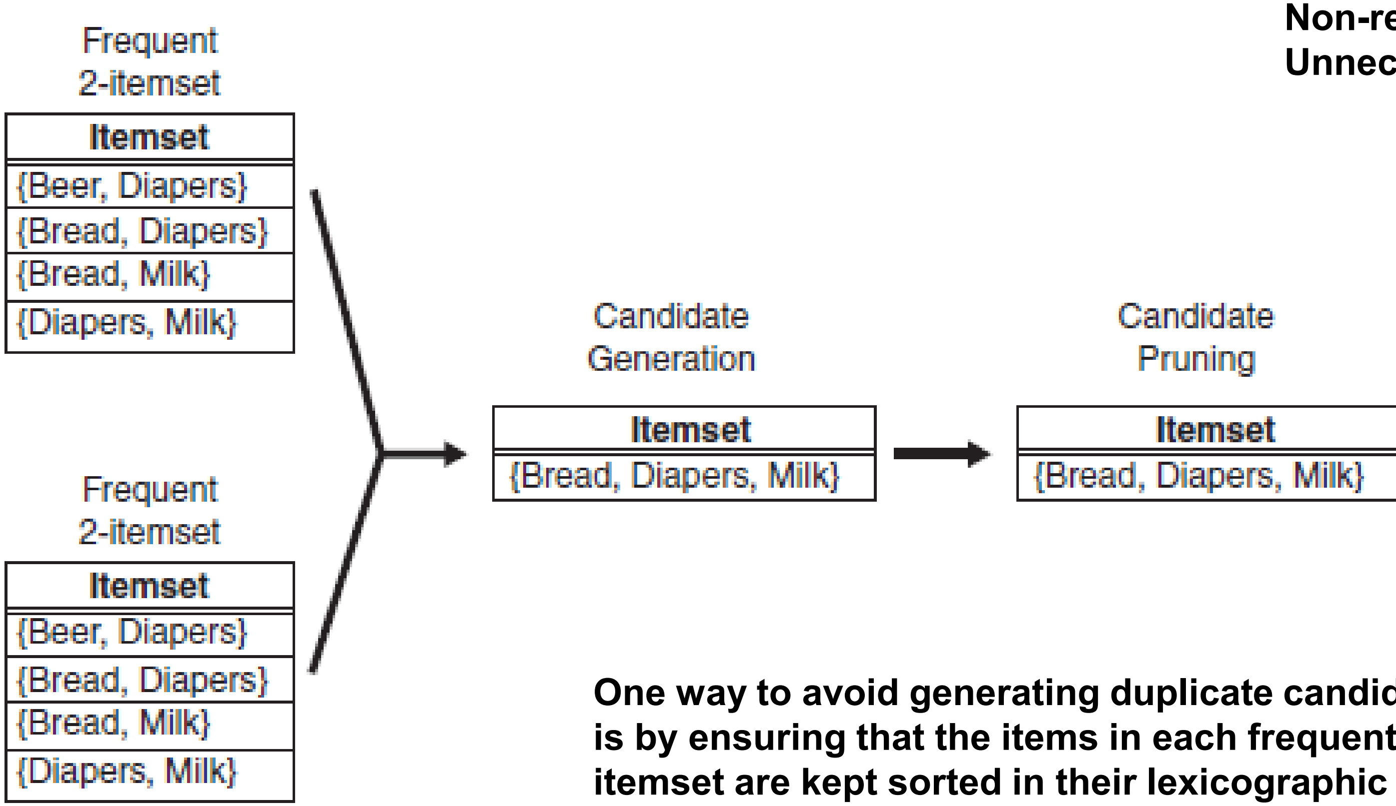
Itemset
{Bread, Diapers, Milk}

Complete: Ok
Non-redundant: ?
Unnecessary: !Ok
20 → 1

Figure 6.6. A brute-force method for generating candidate 3-itemsets.

Candidate Generation: Fk-1 x Fk-1 Method

Complete: Ok
Non-redundant: Ok
Unnecessary: Ok



One way to avoid generating duplicate candidates is by ensuring that the items in each frequent itemset are kept sorted in their lexicographic order

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

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(**<u>AB</u>**C, **<u>AB</u>**D) = **<u>AB</u>**CD
  - Merge(**<u>AB</u>**C, **<u>AB</u>**E) = **<u>AB</u>**CE
  - Merge(**<u>AB</u>**D, **<u>AB</u>**E) = **<u>AB</u>**DE

  - Do not merge(**<u>A</u>**BD,**<u>A</u>**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}

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

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,

$6 + 6 + 4 = 16$

$6 + 6 + 1 = 13$

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.

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}

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	Count
{Bread, Diaper, Milk}	2

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, Diapers, Beer, Eggs
3	Milk, Diapers, Beer, Coke
4	Bread, Milk, Diapers, Beer
5	Bread, Milk, Diapers, Coke

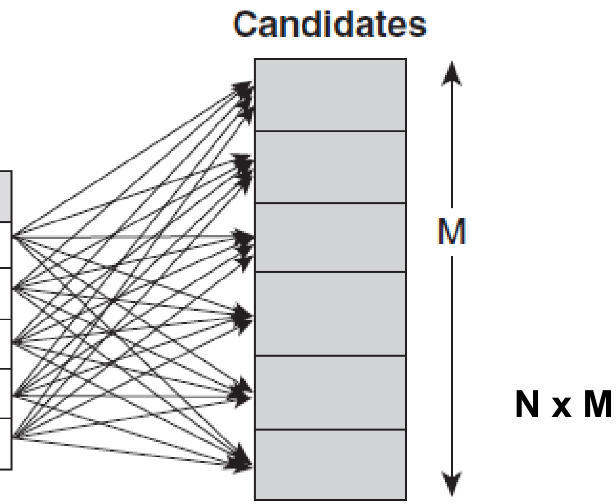


Figure 5.2. Counting the support of candidate itemsets.

Approach: enumerate the itemsets contained in each transaction and use them to update the support counts of their respective candidate itemsets

Support Counting of Candidate Itemsets

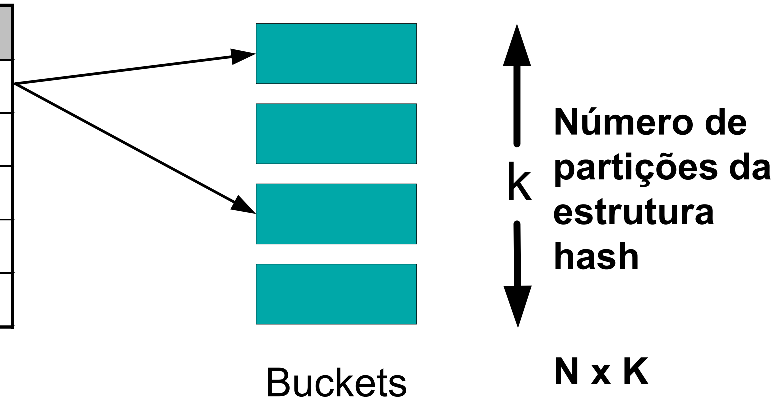
- To reduce number of comparisons, store the candidate itemsets in a hash structure
  - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets

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



N x K

Approach: enumerate the itemsets contained in each transaction and use them to update the support counts of their respective candidate itemsets

Support Counting: An Example

Suppose you have 15 candidate itemsets of length 3:

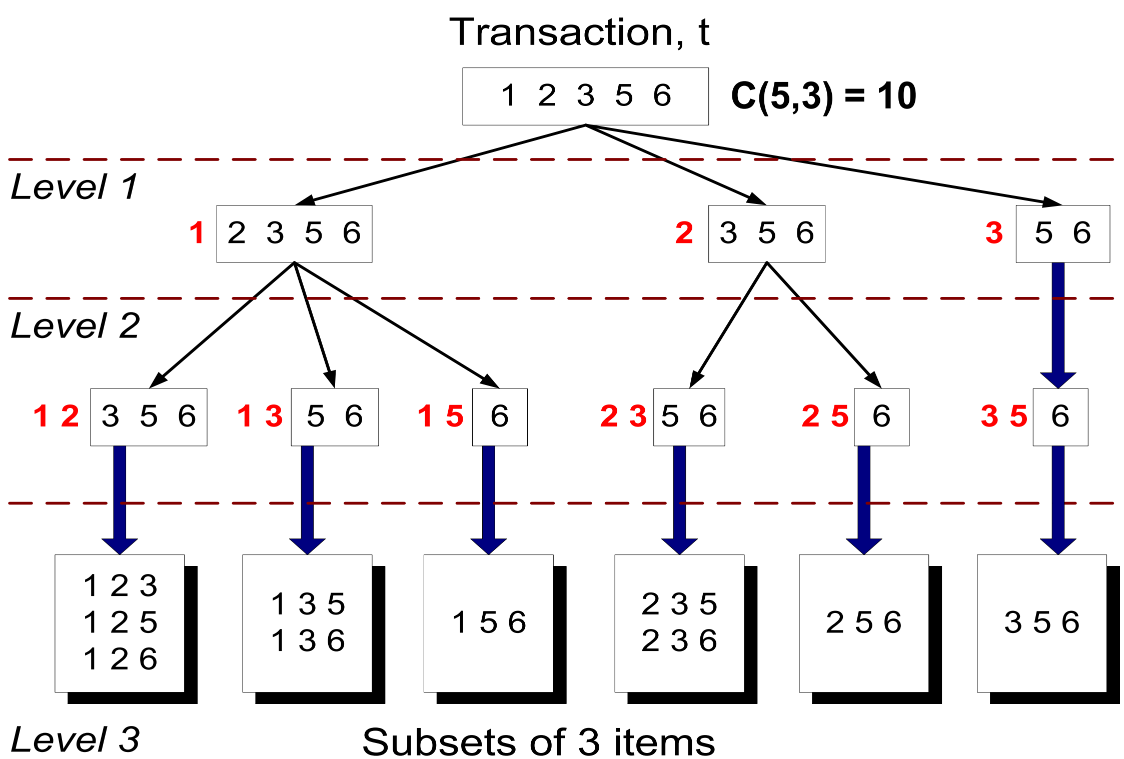
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

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

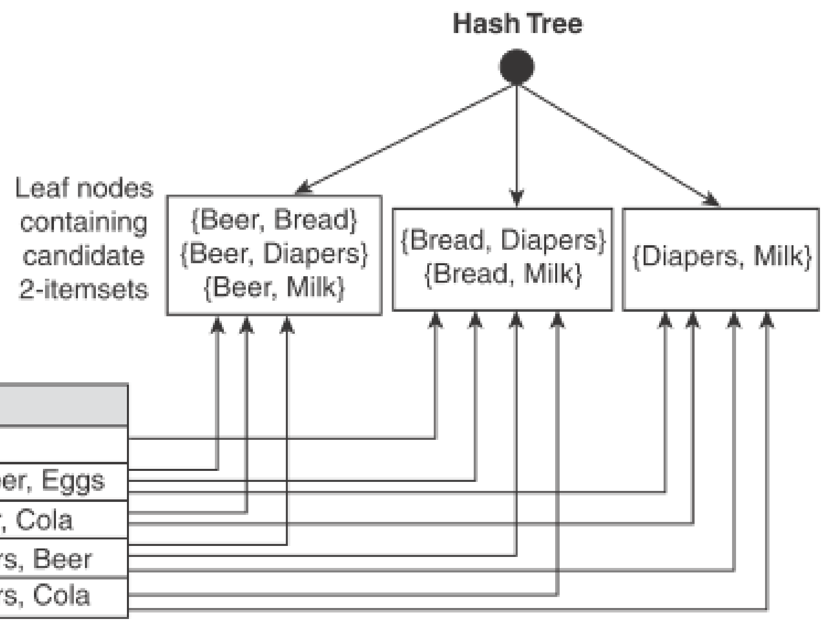
A figura apresenta uma maneira de realizar a enumeração dos itemsets candidatos contidos em uma dada transação

Funciona já que os itens estão em ordem lexicográfica - os possíveis itemsets começam com 1, 2 ou 3

Como identificar qual deles é um itemset candidato?



Support Counting Using a Hash Tree



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

Figure 5.10. Counting the support of itemsets using hash structure.

Candidate itemsets are partitioned into different buckets and stored in a hash tree

During support counting, itemsets contained in each transaction are also hashed into their appropriate buckets

That way, instead of comparing each itemset in the transaction with every candidate itemset, it is matched only against candidate itemsets that belong to the same bucket

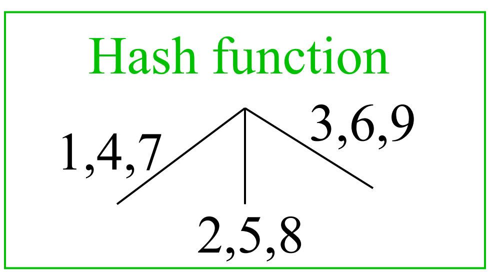
Support Counting Using a Hash Tree

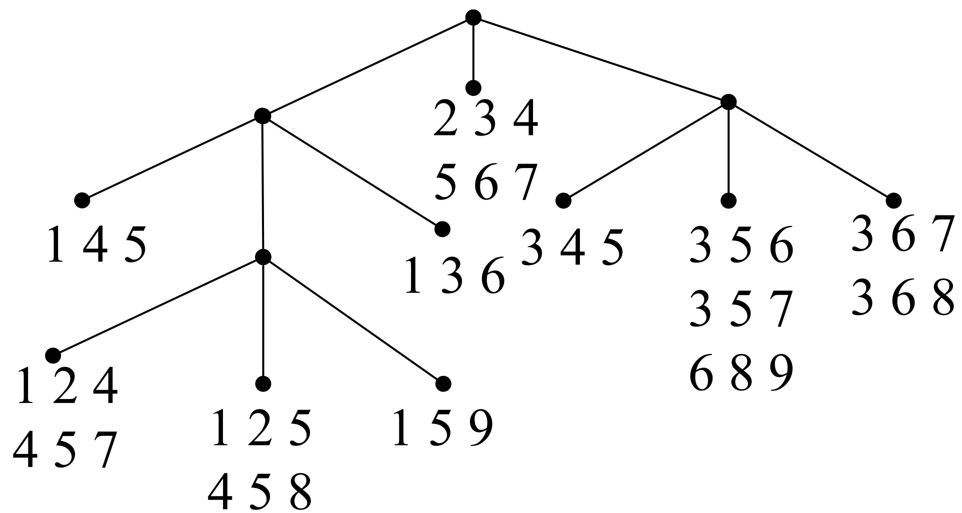
Suppose you have 15 candidate itemsets of length 3:

{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 (h(k) = (k-1) mod 3)**
- **Max leaf size: max number of itemsets stored in a leaf node (if number of candidate itemsets exceeds max leaf size, split the node)**



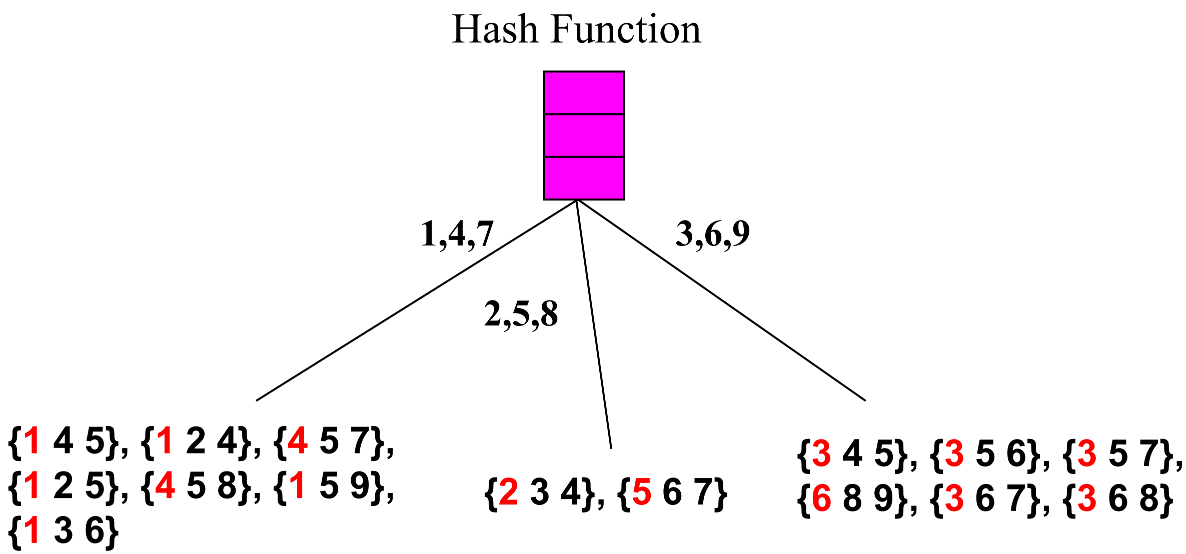


Support Counting Using a Hash Tree

h(1) = (1-1) mod 3 = 0
h(2) = (2-1) mod 3 = 1
h(3) = (3-1) mod 3 = 2
h(4) = (4-1) mod 3 = 0
h(5) = (5-1) mod 3 = 1
h(6) = (6-1) mod 3 = 2
h(7) = (7-1) mod 3 = 0
h(8) = (8-1) mod 3 = 1
h(9) = (9-1) mod 3 = 2

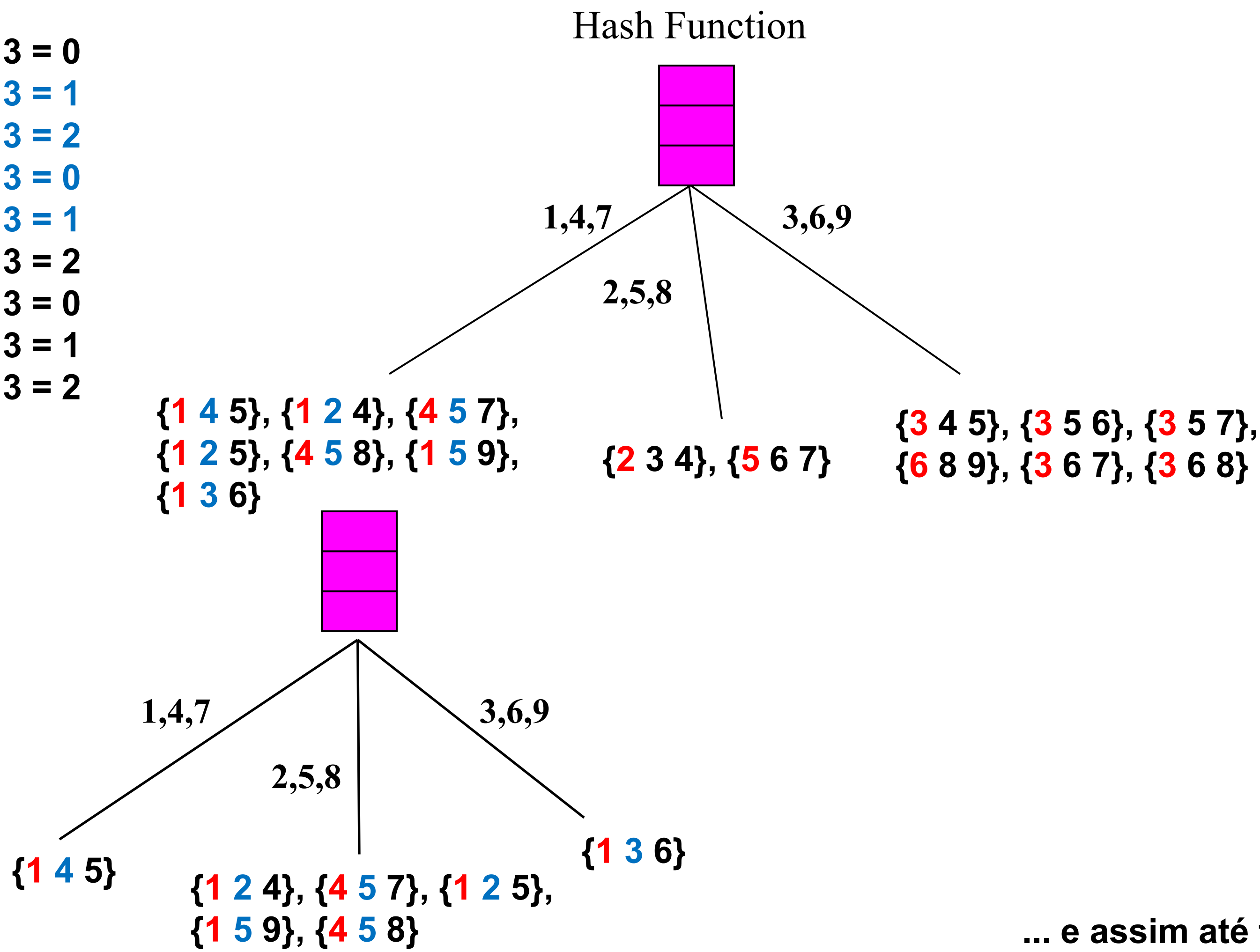
[1 a 9] (itens distintos)

{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}



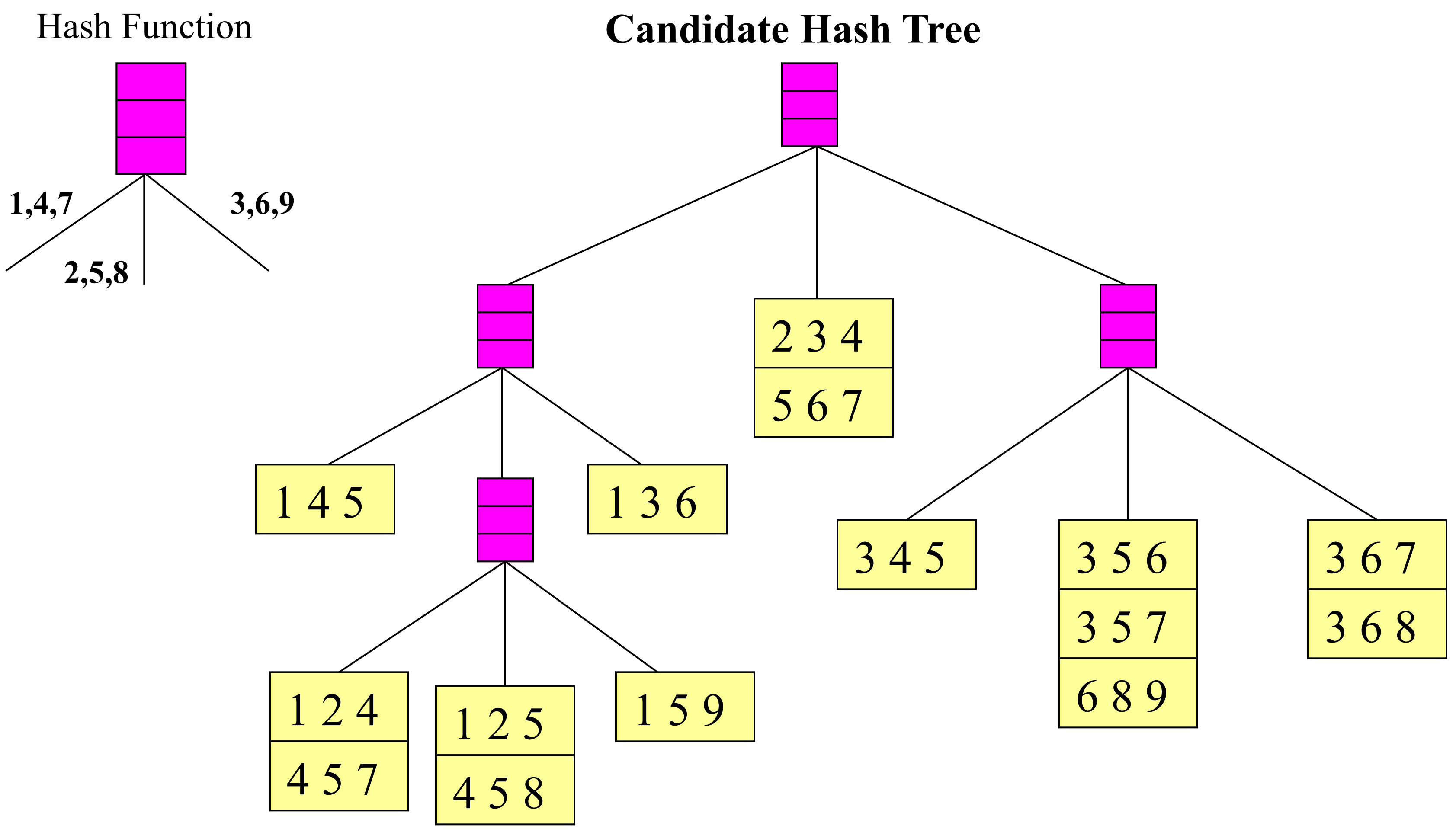
Support Counting Using a Hash Tree

h(1) = (1-1) mod 3 = 0
h(2) = (2-1) mod 3 = 1
h(3) = (3-1) mod 3 = 2
h(4) = (4-1) mod 3 = 0
h(5) = (5-1) mod 3 = 1
h(6) = (6-1) mod 3 = 2
h(7) = (7-1) mod 3 = 0
h(8) = (8-1) mod 3 = 1
h(9) = (9-1) mod 3 = 2

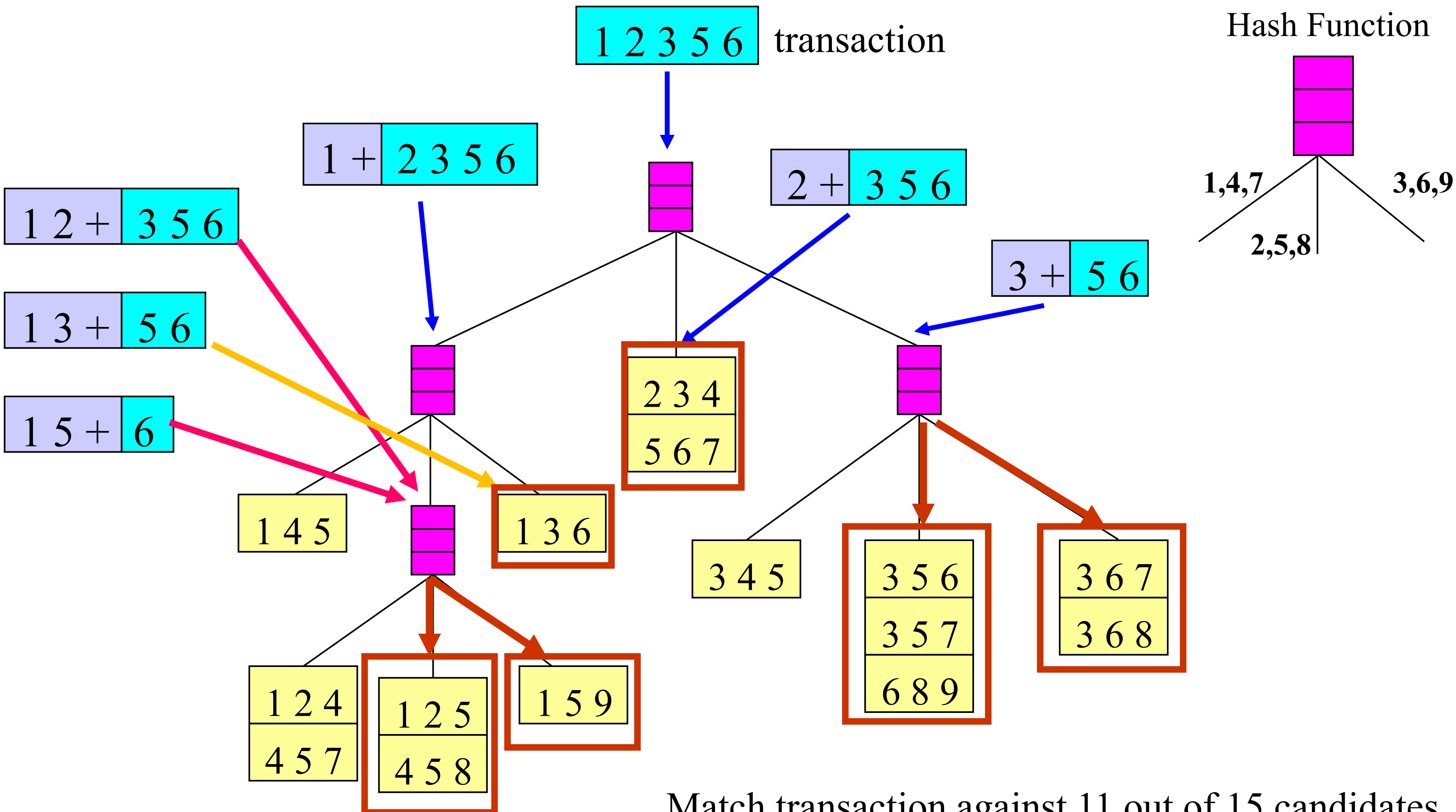


... e assim até terminar

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:

$ABC \rightarrow D$,	$ABD \rightarrow C$,	$ACD \rightarrow B$,	$BCD \rightarrow A$,
$A \rightarrow BCD$,	$B \rightarrow ACD$,	$C \rightarrow ABD$,	$D \rightarrow ABC$
$AB \rightarrow CD$,	$AC \rightarrow BD$,	$AD \rightarrow BC$,	$BC \rightarrow AD$,
$BD \rightarrow AC$,	$CD \rightarrow AB$,		

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

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:

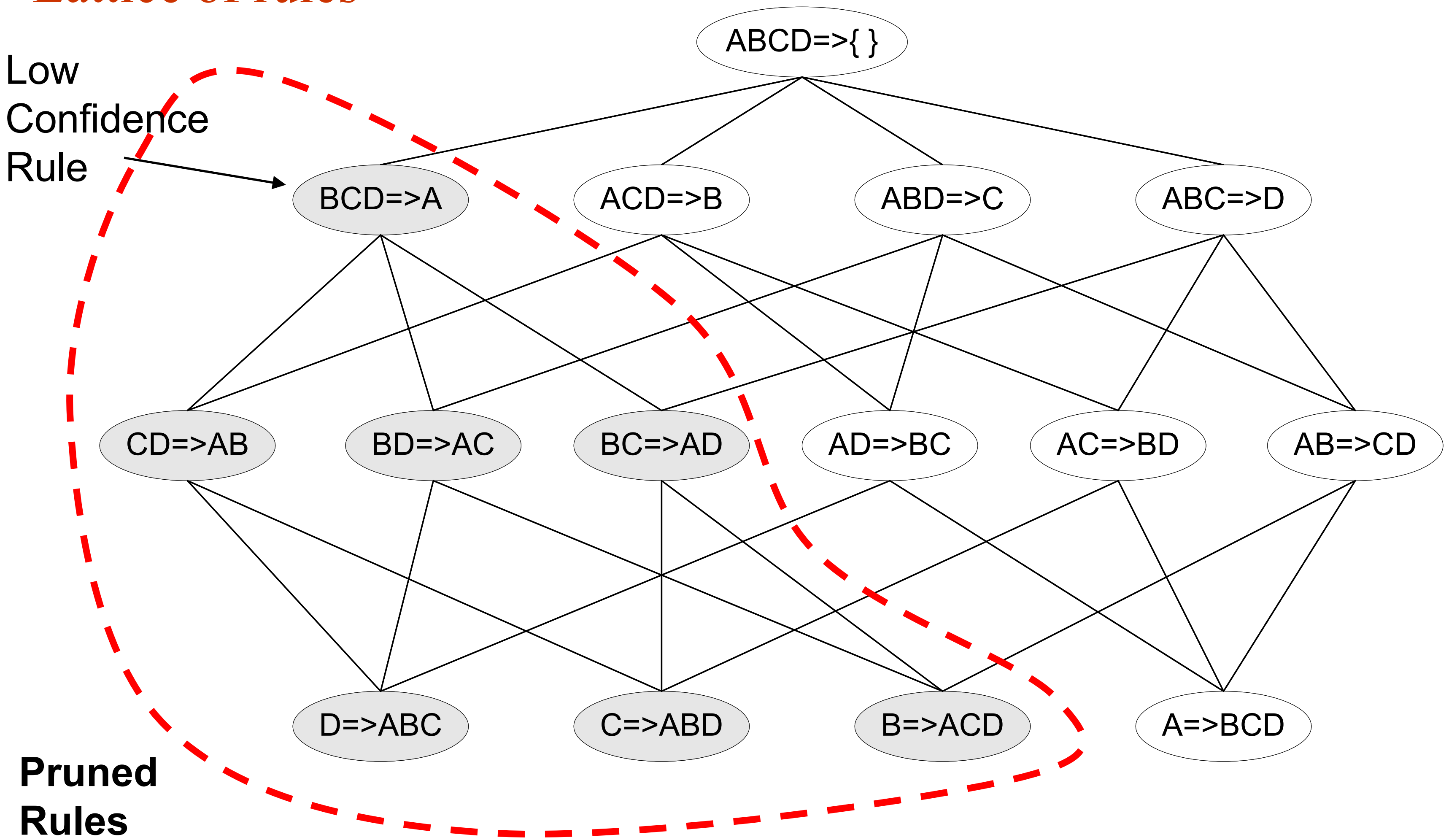
$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

Aumenta-se ou mantém-se o denominador (se aumentar, a confiança vai diminuir, já que o numerador é fixo)

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

Rule Generation for Apriori Algorithm

Lattice of rules



Factors Affecting Complexity of Apriori

- 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: useful to identify a small representative set of frequent itemsets from which all other frequent itemsets can be derived

Maximal Frequent Itemset

An itemset is maximal frequent if it is frequent and none of its immediate supersets is frequent

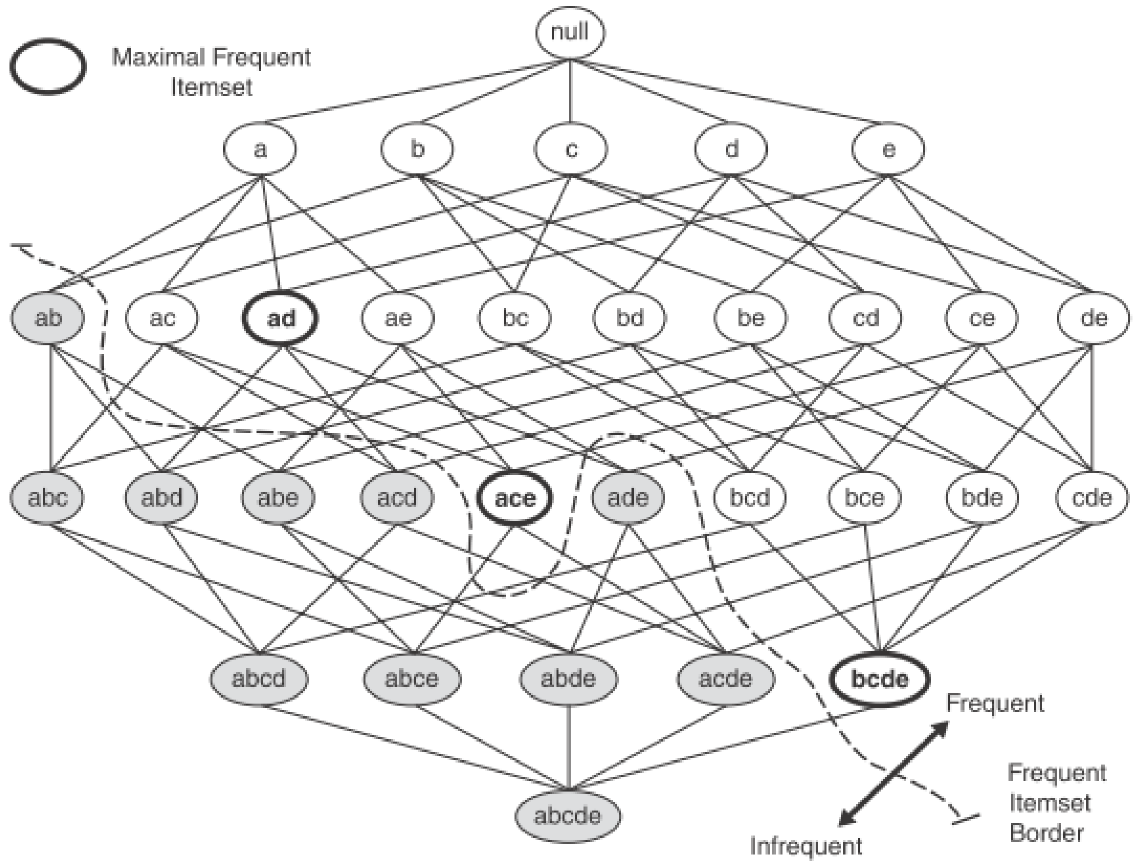


Figure 5.16. Maximal frequent itemset.

Maximal Frequent Itemset (MFI)

An itemset is maximal frequent if it is frequent and none of its immediate supersets is frequent

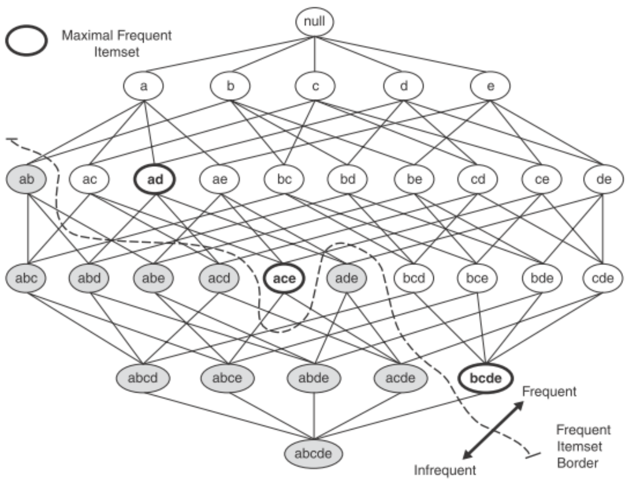


Figure 5.16. Maximal frequent itemset.

MFI forms the smallest set of itemsets from which all frequent itemsets can be derived

Every frequent itemset is a subset of one of the three maximal frequent itemsets: {a, d}, {a, c, e}, {b, c, d, e}

Enumerating all the subsets of maximal frequent itemsets generates the complete list of all frequent itemsets

However, maximal frequent itemsets do not contain the support information of their subsets, except that they meet the support threshold

An additional pass over the data set is therefore needed to determine the support counts of the non-maximal frequent itemsets

Therefore, it is desirable to have a minimal representation of itemsets that preserves the support information (closed)

Closed Itemset

40% = 2 transações

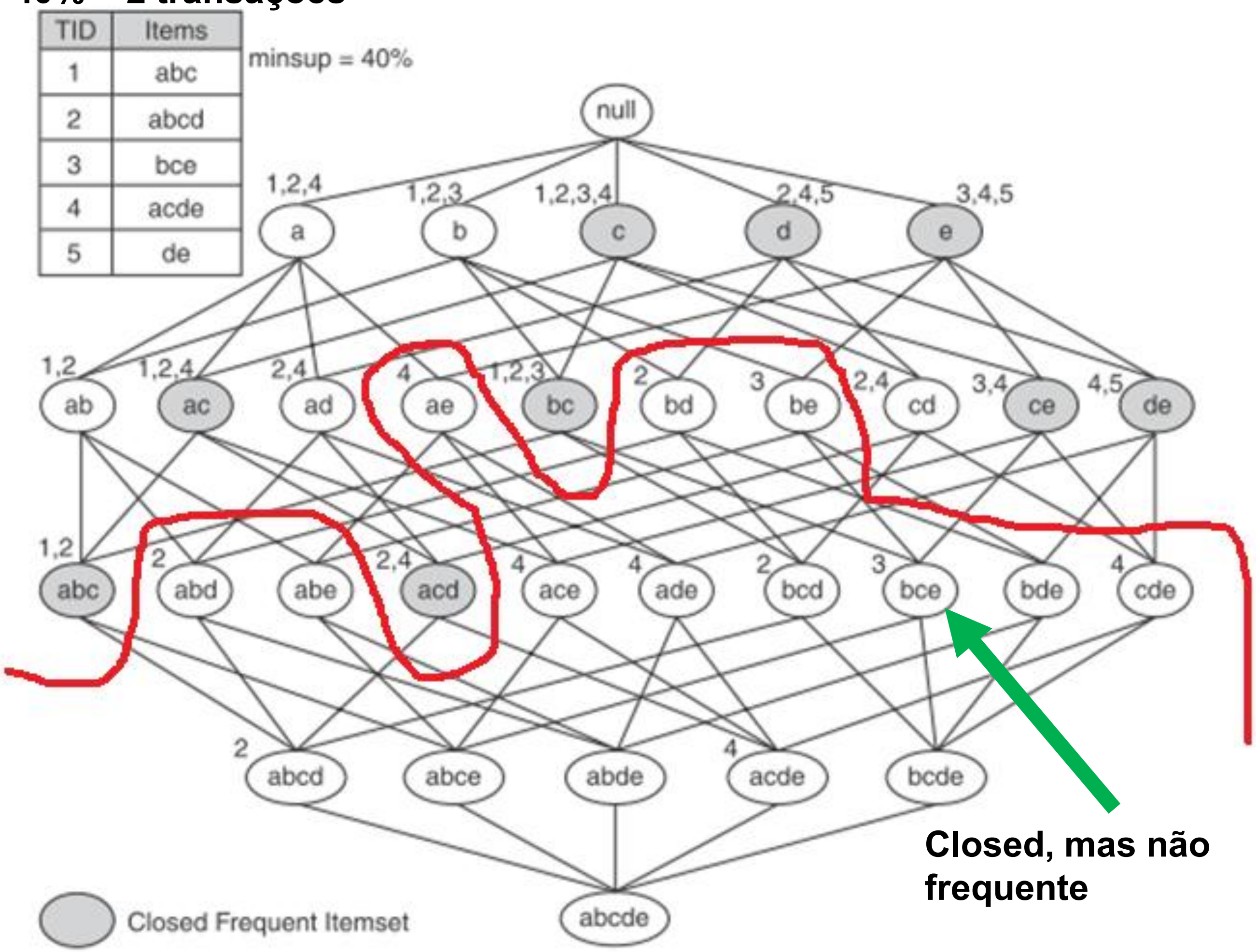


Figure 5.17. An example of the closed frequent itemsets (with minimum support equal to 40%).

→ An itemset X is closed if none of its immediate supersets has the same support as the itemset X.

→ X is not closed if at least one of its immediate supersets has support count as X.

Maximal vs Closed Frequent Itemsets

40% = 2 transações

TID	Items
1	abc
2	abcd
3	bce
4	acde
5	de

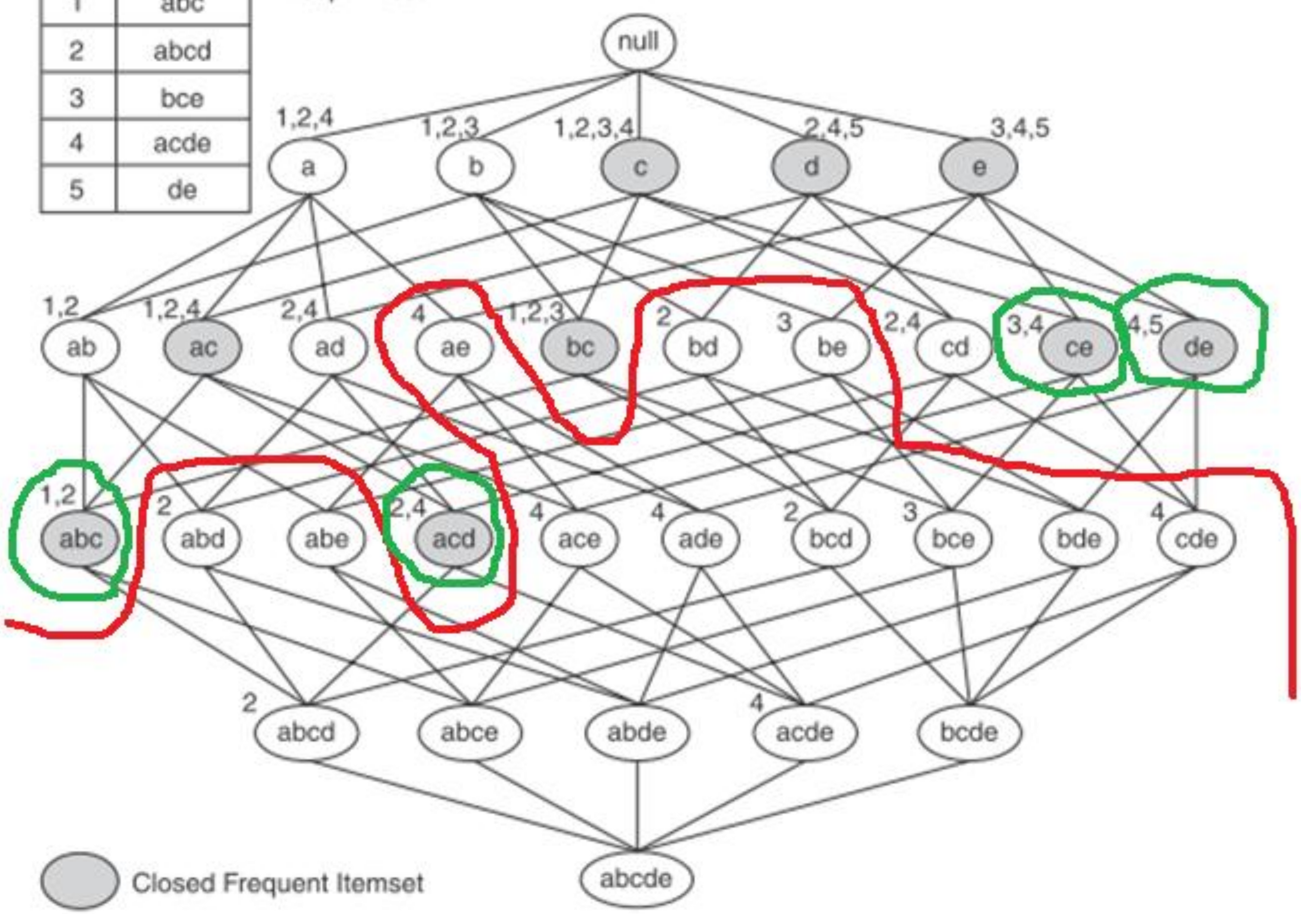


Figure 5.17. An example of the closed frequent itemsets (with minimum support equal to 40%).

#Closed = 9
#Maximal = 4

→ Closed Frequent Itemset: an itemset is a closed frequent itemset if it is closed and its support is greater than or equal to minsup.

Maximal vs Closed Frequent Itemsets

40% = 2 transações

TID	Items
1	abc
2	abcd
3	bce
4	acde
5	de

minsup = 40%

If an itemset is not closed, its support count must be equal to the maximum support count of its immediate supersets

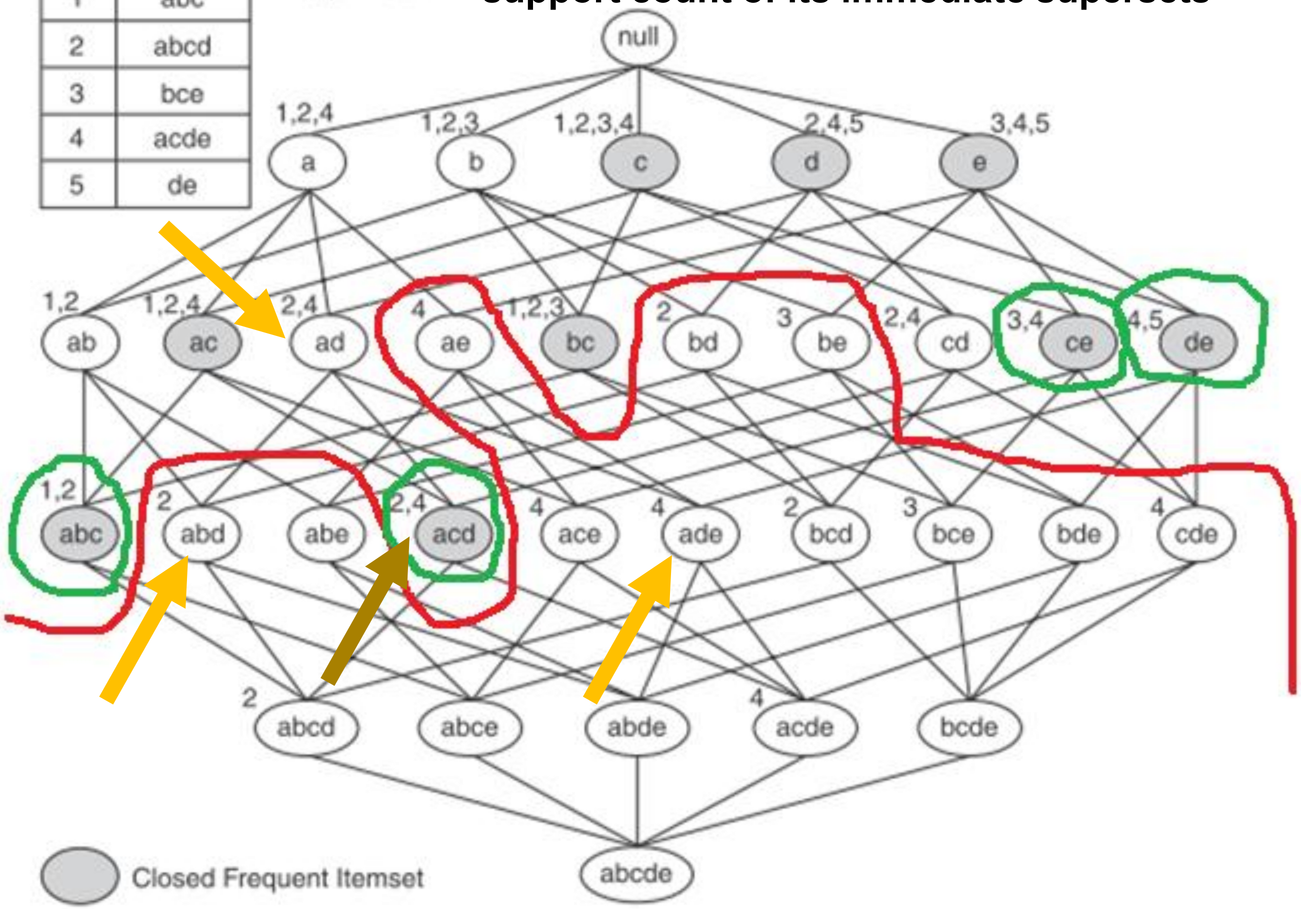
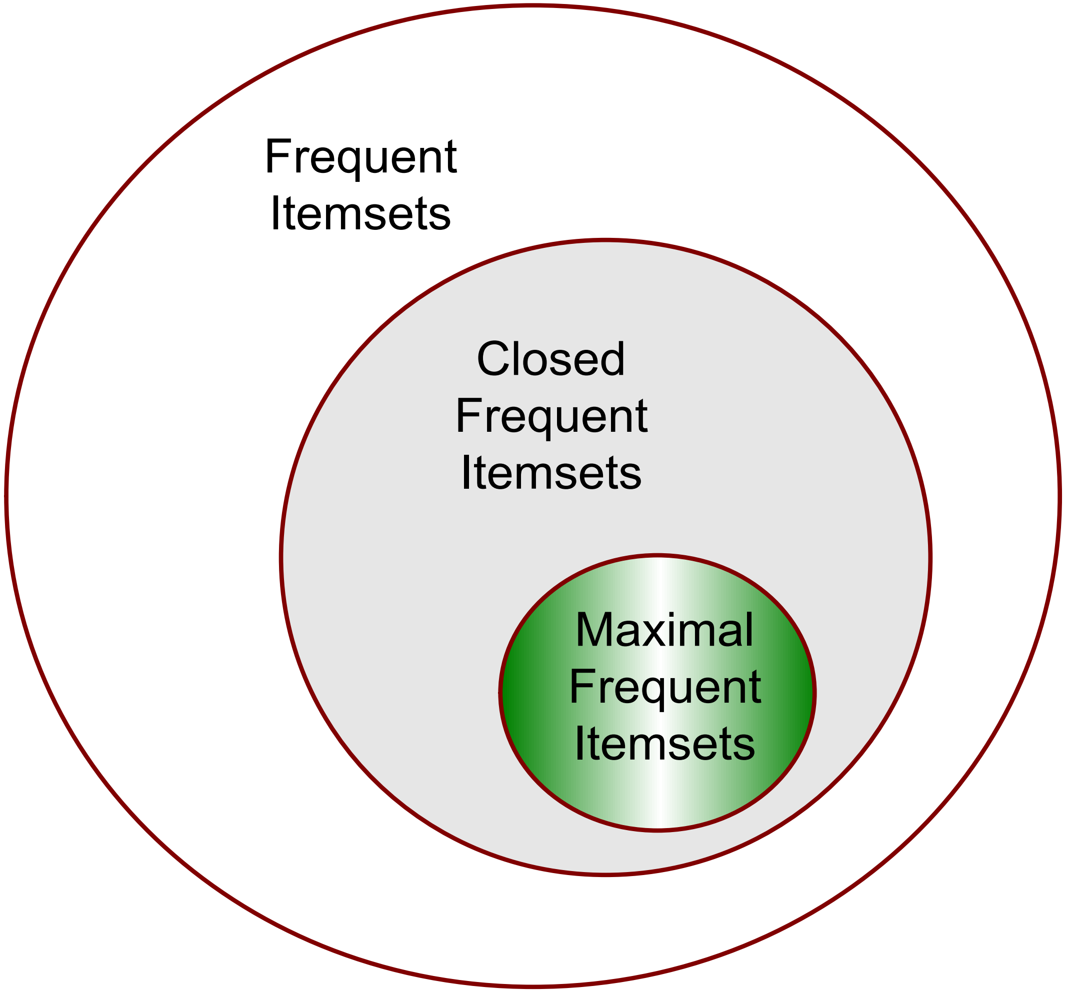


Figure 5.17. An example of the closed frequent itemsets (with minimum support equal to 40%).

#Closed = 9
#Maximal = 4

→ The support count of every non-closed frequent k-itemset can be obtained by considering the support of all its frequent supersets of size k + 1 (maximum)

Maximal vs Closed Frequent Itemsets



Alternative Methods for Generating Frequent Itemsets

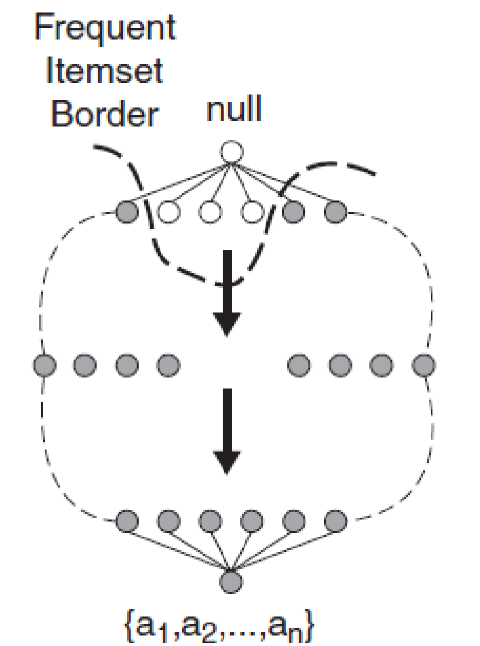
- Apriori is one of the earliest algorithms to have successfully addressed the combinatorial explosion of frequent itemset generation
  - However, the algorithm requires making several passes over the transaction data set
  - Besides, the performance of the algorithm may degrade significantly for dense data sets because of the increasing width of transactions
- Several alternative methods have been developed to overcome these limitations and improve upon the efficiency of the Apriori algorithm

Alternative Methods for Generating Frequent Itemsets

- Traversal of Itemset Lattice
  - General-to-Specific versus Specific-to-General
  - Equivalence Classes
  - Breadth-First versus Depth-First
- Representation of Transaction Data Set

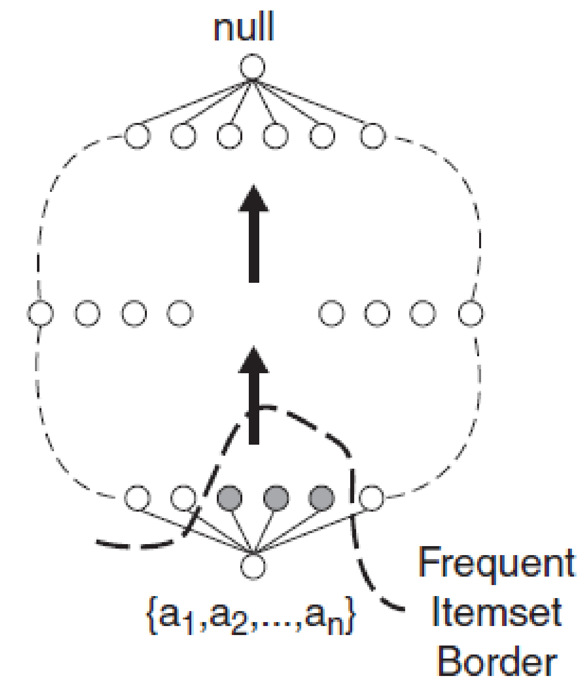
Alternative Methods for Generating Frequent Itemsets

- Traversal of Itemset Lattice
  - General-to-Specific versus Specific-to-General



(a) General-to-specific

The darker nodes represent infrequent itemsets

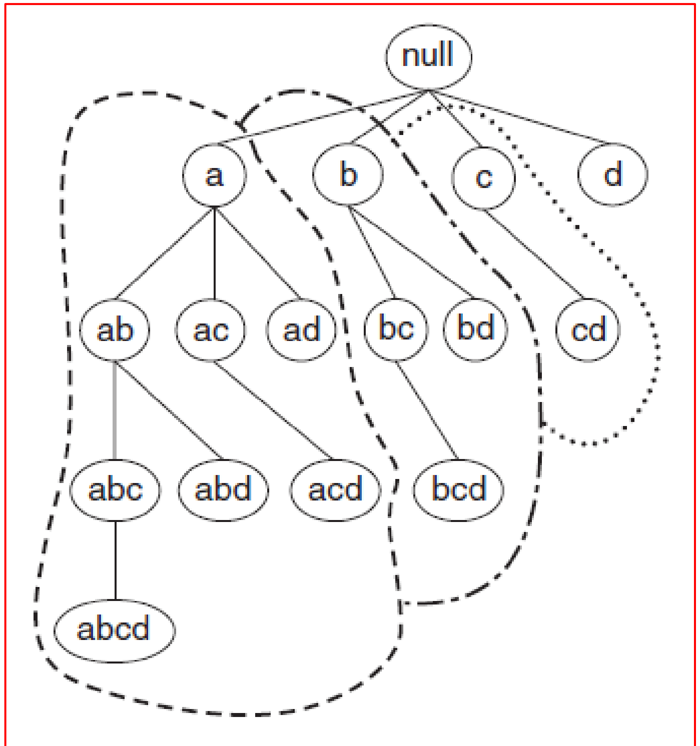


(b) Specific-to-general

Useful to discover <u>maximal frequent itemsets</u> in dense transactions, where the frequent itemset border is located near the bottom of the lattice

Alternative Methods for Generating Frequent Itemsets

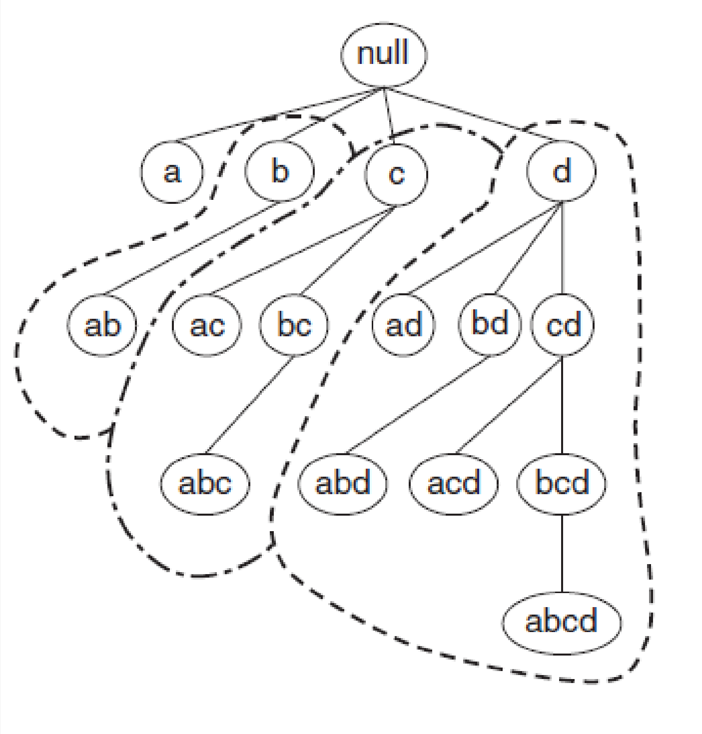
- Traversal of Itemset Lattice
  - Equivalence Classes



(a) Prefix tree.

Equivalence classes = disjoint groups of nodes

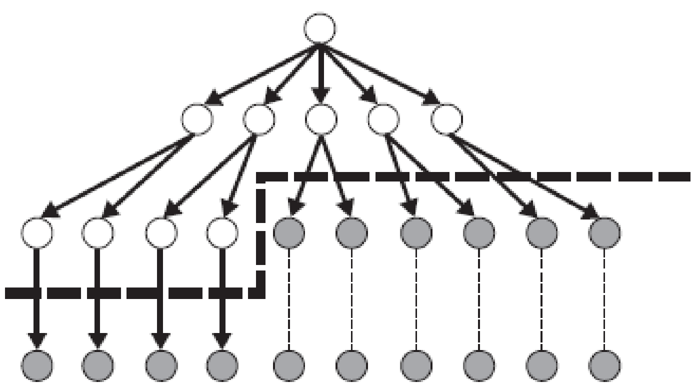
Apriori = partitioning the lattice on the basis of itemset sizes; i.e., the algorithm discovers all frequent 1-itemsets first before proceeding to larger-sized itemsets



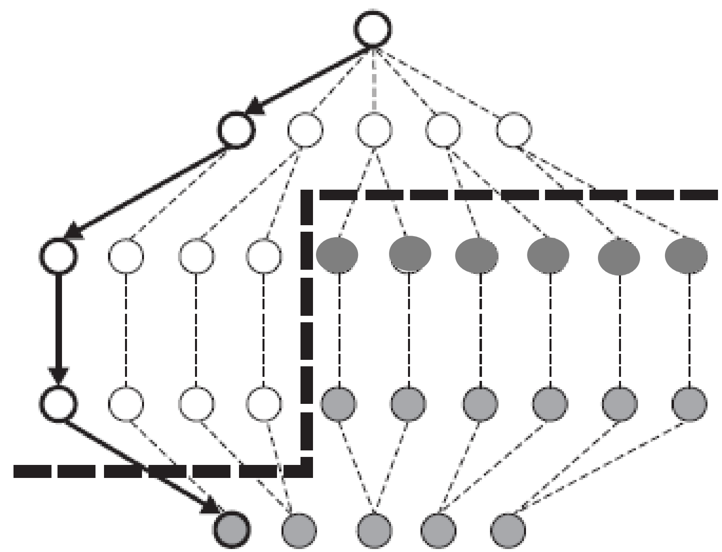
(b) Suffix tree.

Alternative Methods for Generating Frequent Itemsets

- Traversal of Itemset Lattice
  - Breadth-First versus Depth-First



(a) Breadth first



(b) Depth first

Often used by algorithm designed to find maximal frequente itemsets

Alternative Methods for Generating Frequent Itemsets

- Representation of Transaction Data Set
  - There are many ways to represent a transaction data set
  - The choice of representation can affect the I/O costs incurred when computing the support of candidate itemsets

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				

FP-Growth

- FP-growth takes a radically different approach to discovering frequent itemsets
- It encodes the data set using a compact data structure called an FP-tree and extracts frequent itemsets directly from this structure
- It scans the data set twice and doesn't generate candidate itemsets

FP-Growth: FP-Tree

- An FP-tree is a compressed representation of the input data
- The size of an FP-tree is typically smaller than the size of the uncompressed data because many transactions in market basket data often share a few items in common
  - If the size of the FP-tree is small enough to fit into main memory, this will allow us to extract frequent itemsets directly from the structure in memory instead of making repeated passes over the data stored on disk

FP-Growth: FP-Tree

https://www.mygreatlearning.com/blog/understanding-fp-growth-algorithm/ **[example]**

Transaction ID	List of items in the transaction
T1	B , A , T
T2	A , C
T3	A , S
T4	B , A , C
T5	B , S
T6	A , S
T7	B , S
T8	B , A , S , T
T9	B , A , S

Item	Support Count
Asparagus (A)	7
Beans (B)	6
Squash (S)	6
Corn (C)	2
Tomatoes (T)	2

FP-Growth: FP-Tree

Transaction ID	List of items in the transaction
T1	B , A , T → A, B, T
T2	A , C → A, C
T3	A , S → A, S
T4	B , A , C → A, B, C
T5	B , S → B, S
T6	A , S → A, S
T7	B , S → B, S
T8	B , A , S , T → A, B, S, T
T9	B , A , S → A, B, S

Item	Support Count
Asparagus (A)	7
Beans (B)	6
Squash (S)	6
Corn (C)	2
Tomatoes (T)	2

FP-Growth: FP-Tree

Transaction ID	List of items in the transaction
T1	**A, B, T**
T2	A, C
T3	A, S
T4	A, B, C
T5	B, S
T6	A, S
T7	B, S
T8	A, B, S, T
T9	A, B, S

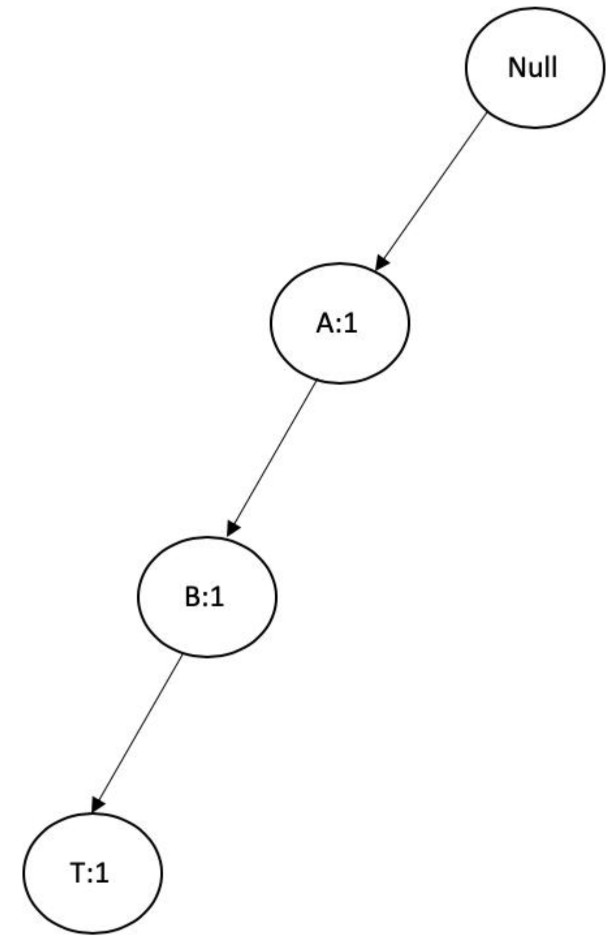


Figure 1 : Transaction ABT

FP-Growth: FP-Tree

Transaction ID	List of items in the transaction
T1	A, B, T
T2	**A, C**
T3	A, S
T4	A, B, C
T5	B, S
T6	A, S
T7	B, S
T8	A, B, S, T
T9	A, B, S

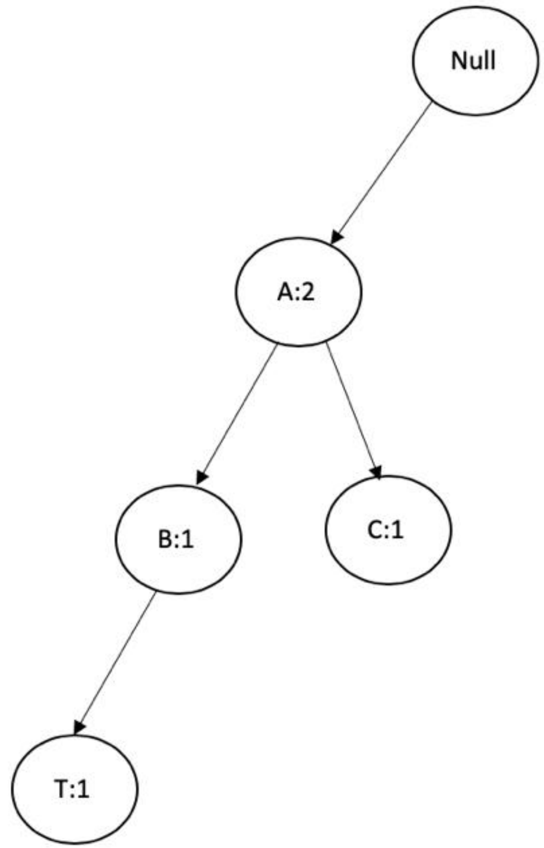


Figure 2 : Transaction AC

FP-Growth: FP-Tree

Transaction ID	List of items in the transaction
T1	A, B, T
T2	A, C
T3	**A, S**
T4	A, B, C
T5	B, S
T6	A, S
T7	B, S
T8	A, B, S, T
T9	A, B, S

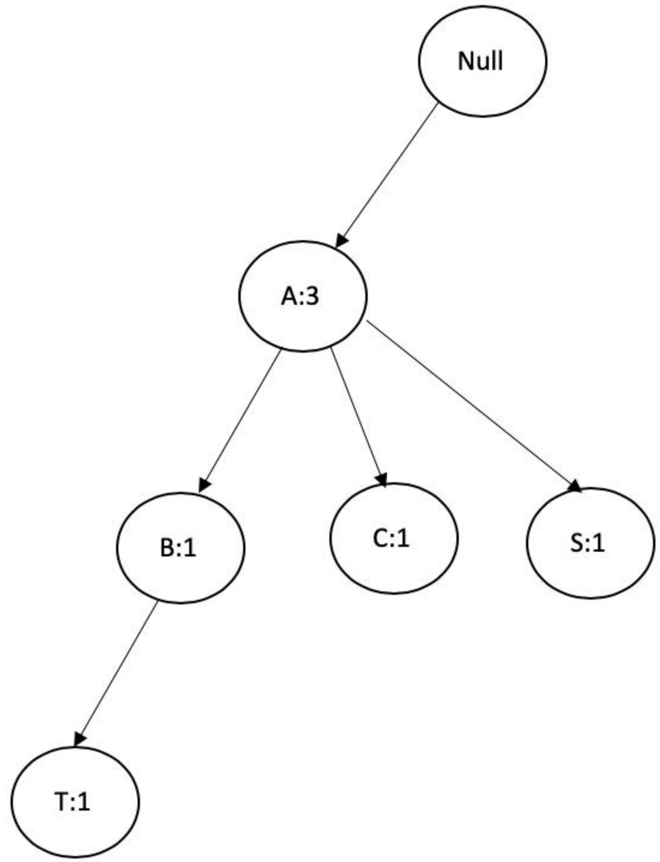


Figure 3 : Transaction AS

FP-Growth: FP-Tree

Transaction ID	List of items in the transaction
T1	A, B, T
T2	A, C
T3	A, S
T4	**A, B, C**
T5	B, S
T6	A, S
T7	B, S
T8	A, B, S, T
T9	A, B, S

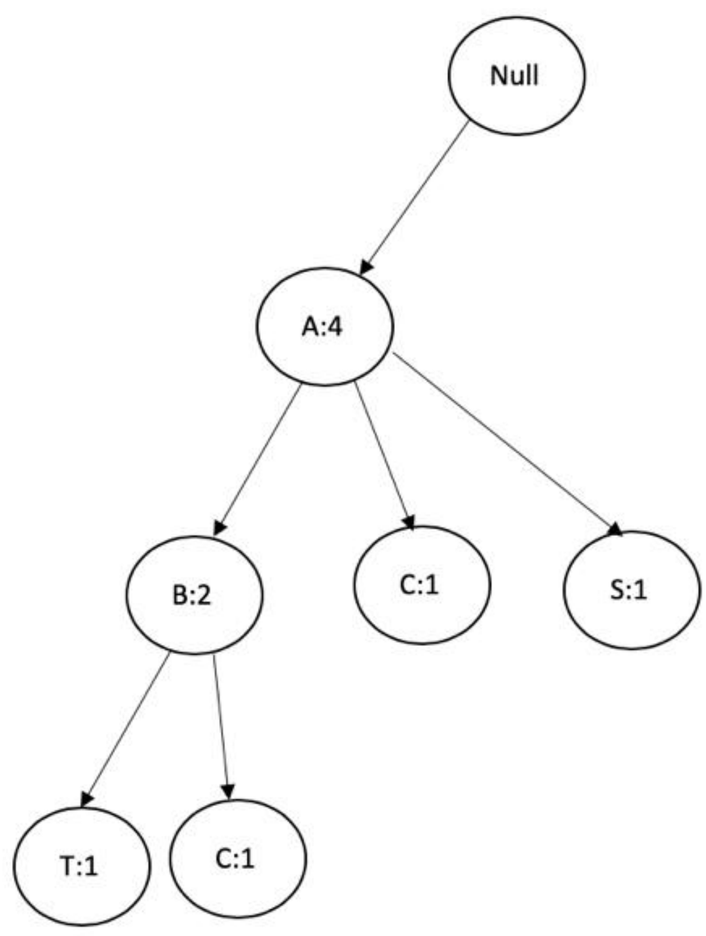


Figure 4 : Transaction ABC

FP-Growth: FP-Tree

Transaction ID	List of items in the transaction
T1	A, B, T
T2	A, C
T3	A, S
T4	A, B, C
T5	**B, S**
T6	A, S
T7	B, S
T8	A, B, S, T
T9	A, B, S

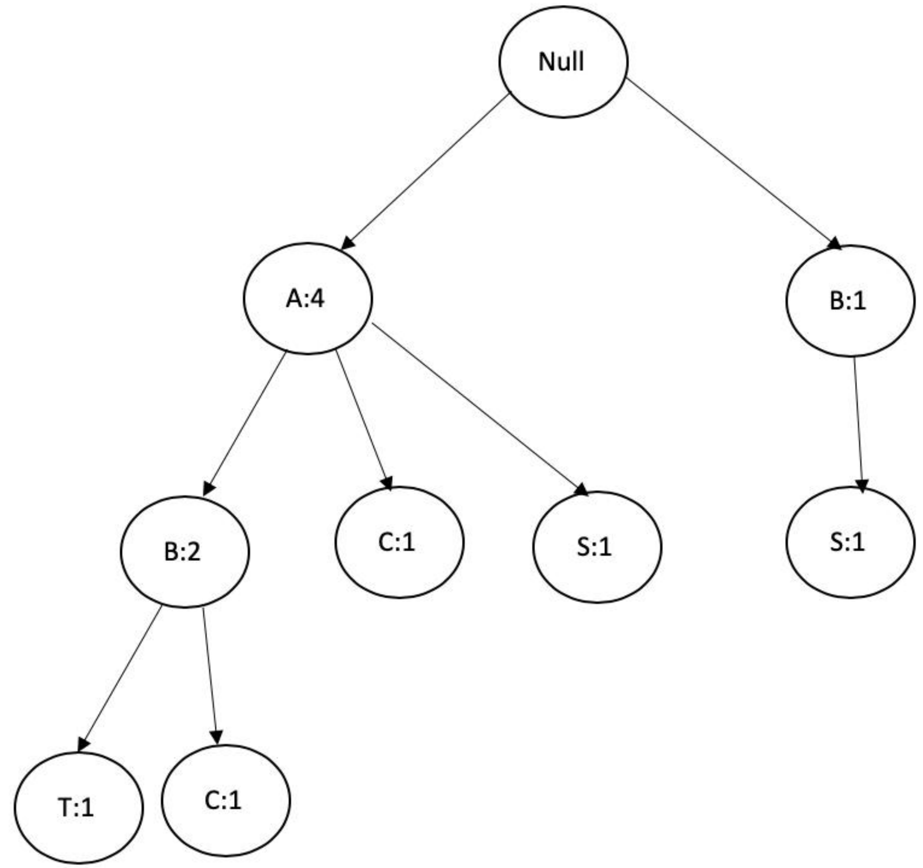


Figure 5 : Transaction BS

FP-Growth: FP-Tree

Transaction ID	List of items in the transaction
T1	A, B, T
T2	A, C
T3	A, S
T4	A, B, C
T5	B, S
T6	**A, S**
T7	B, S
T8	A, B, S, T
T9	A, B, S

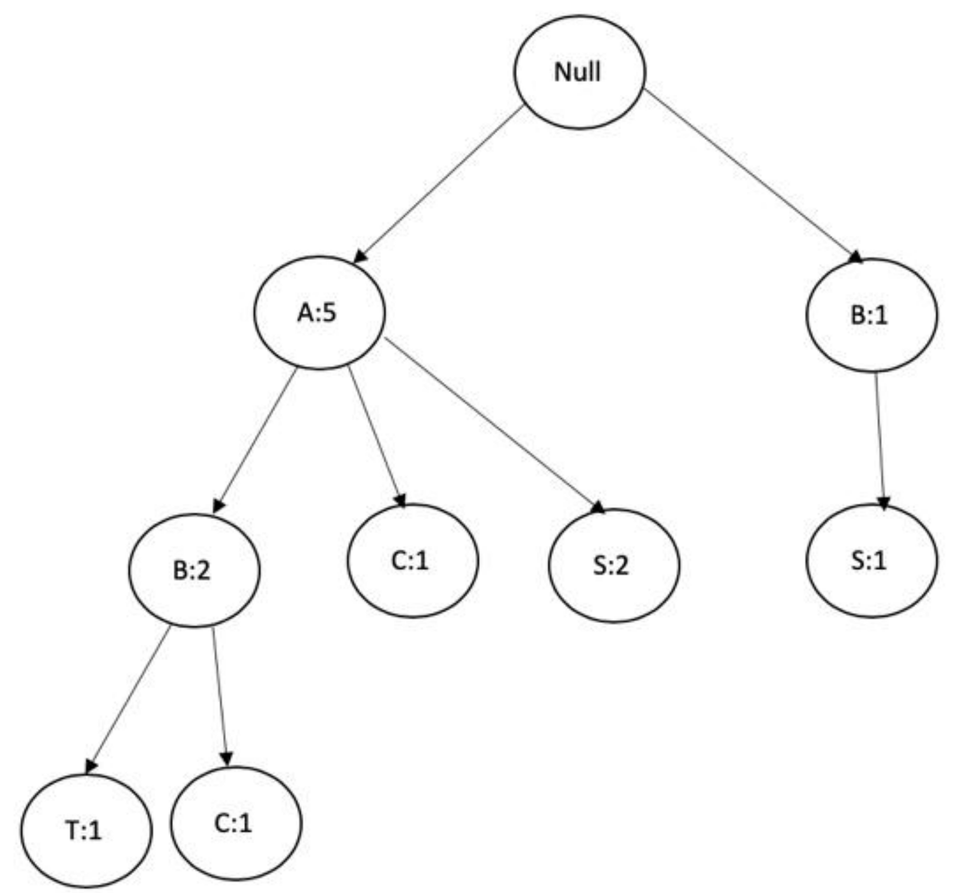


Figure 6 : Transaction AS

FP-Growth: FP-Tree

Transaction ID	List of items in the transaction
T1	A, B, T
T2	A, C
T3	A, S
T4	A, B, C
T5	B, S
T6	A, S
T7	**B, S**
T8	A, B, S, T
T9	A, B, S

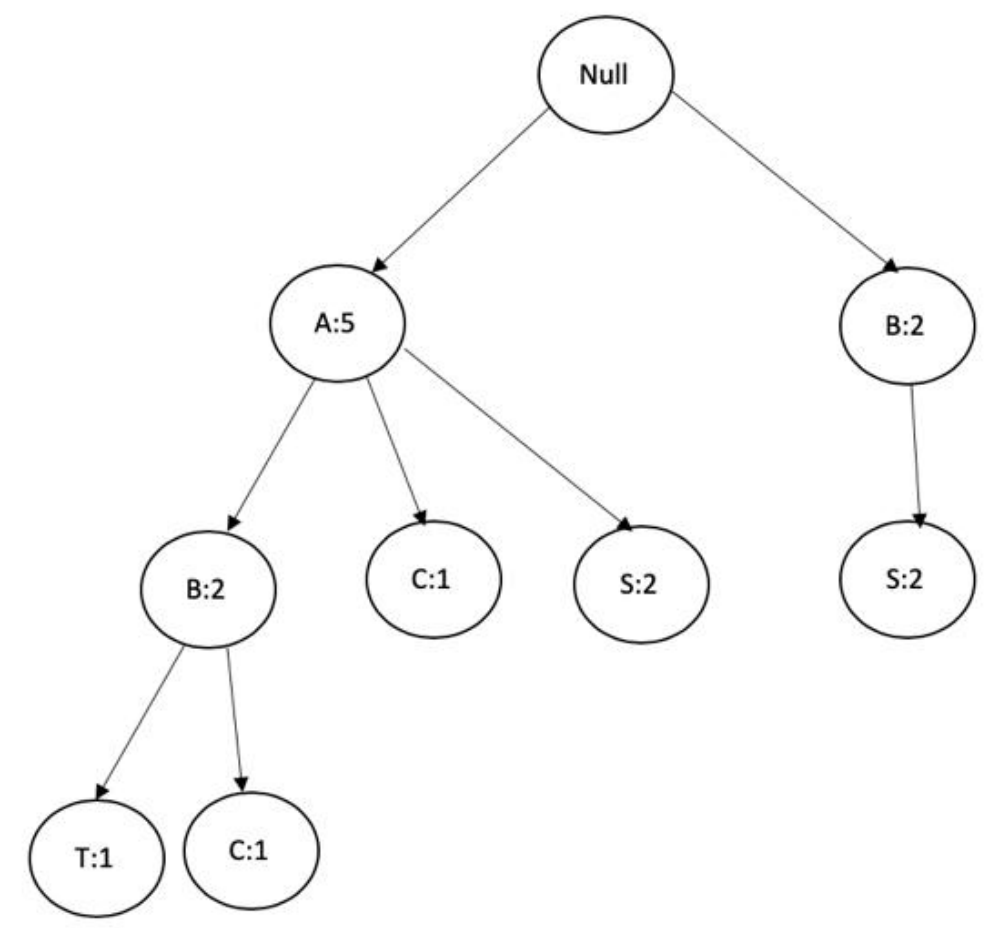


Figure 7 : Transaction BS

FP-Growth: FP-Tree

Transaction ID	List of items in the transaction
T1	A, B, T
T2	A, C
T3	A, S
T4	A, B, C
T5	B, S
T6	A, S
T7	B, S
T8	**A, B, S, T**
T9	A, B, S

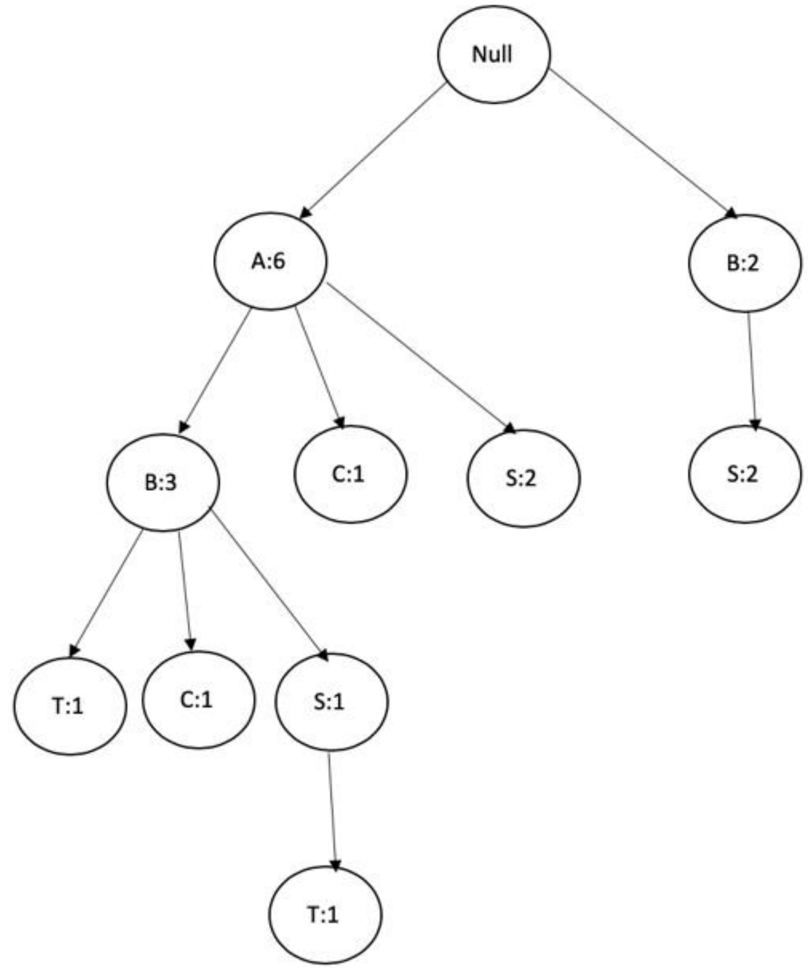


Figure 8 : Transaction ABST

FP-Growth: FP-Tree

Transaction ID	List of items in the transaction
T1	A, B, T
T2	A, C
T3	A, S
T4	A, B, C
T5	B, S
T6	A, S
T7	B, S
T8	A, B, S, T
T9	**A, B, S**

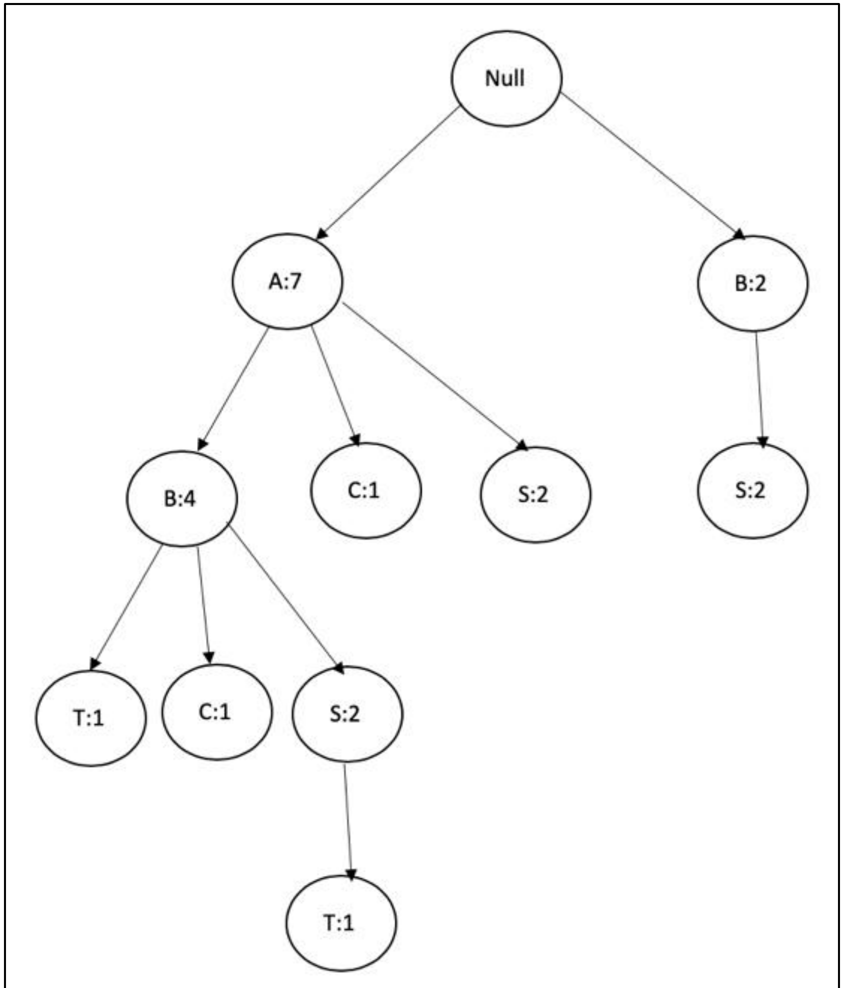
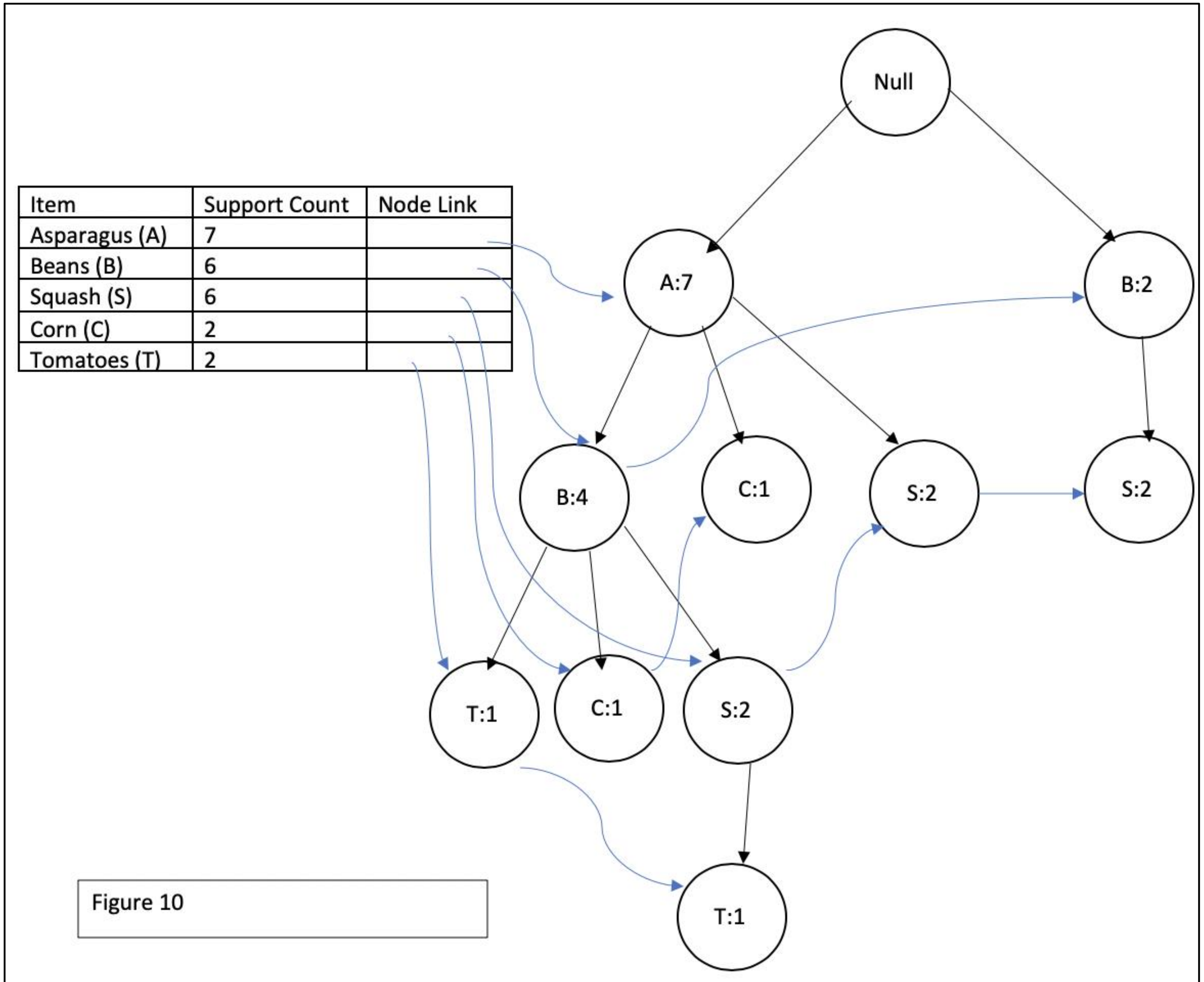


Figure 9 : Transaction ABS

FP-Growth: FP-Tree



Item	Support Count	Node Link
Asparagus (A)	7	
Beans (B)	6	
Squash (S)	6	
Corn (C)	2	
Tomatoes (T)	2	

Figure 10

FP-Growth: FP-Tree

Item	Support Count	Node Link
Asparagus (A)	7	
Beans (B)	6	
Squash (S)	6	
Corn (C)	2	
Tomatoes (T)	2	

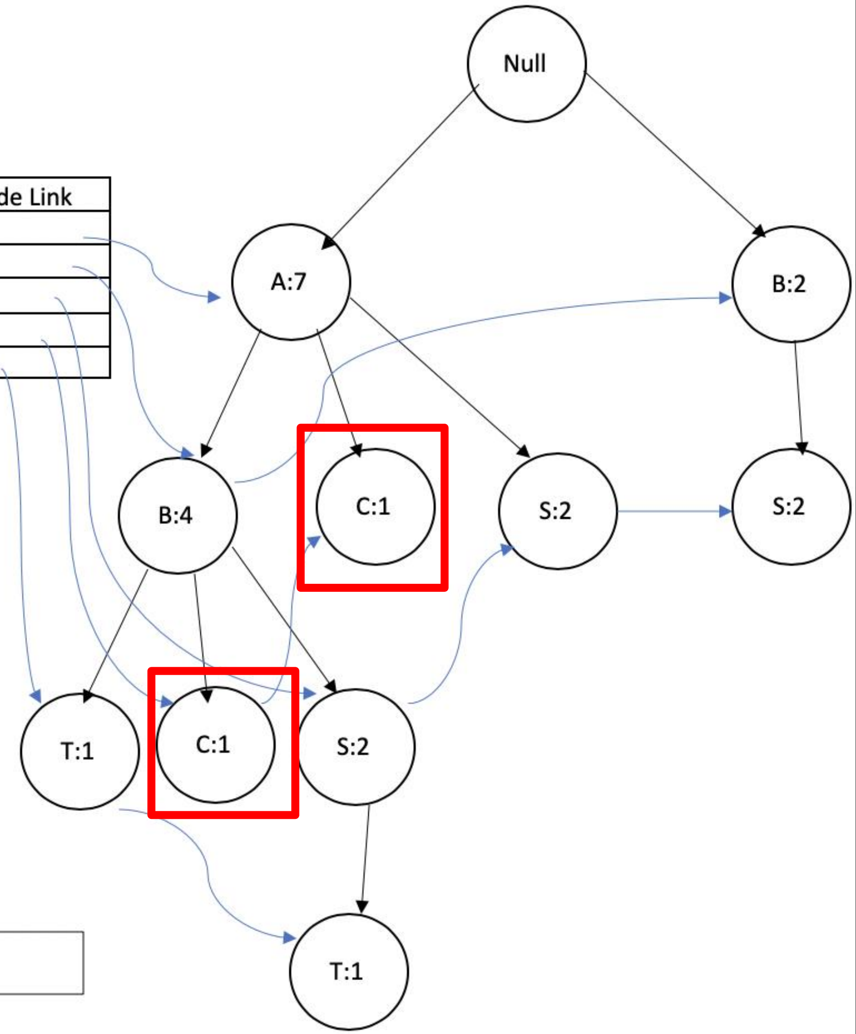


Figure 10

FP-Growth: FP-Tree

- In the best-case scenario, where all the transactions have the same set of items, the FP-tree contains only a single branch of nodes
- The worst case scenario happens when every transaction has a unique set of items
  - As none of the transactions have any items in common, the size of the FP-tree is effectively the same as the size of the original data
  - However, the physical storage requirement for the FP-tree is higher because it requires additional space to store pointers between nodes and counters for each item

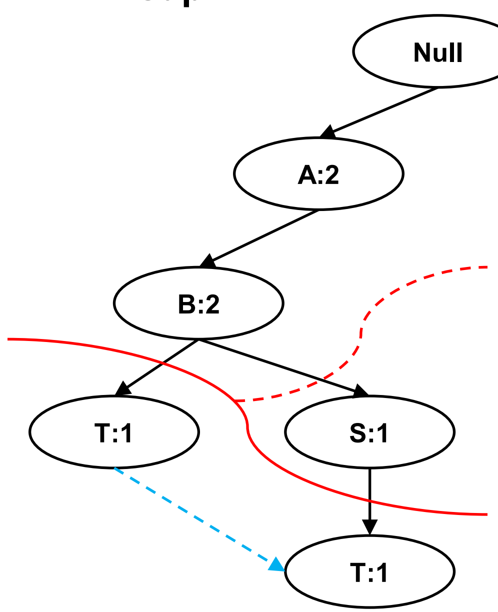
FP-Growth: Frequent Itemset Generation

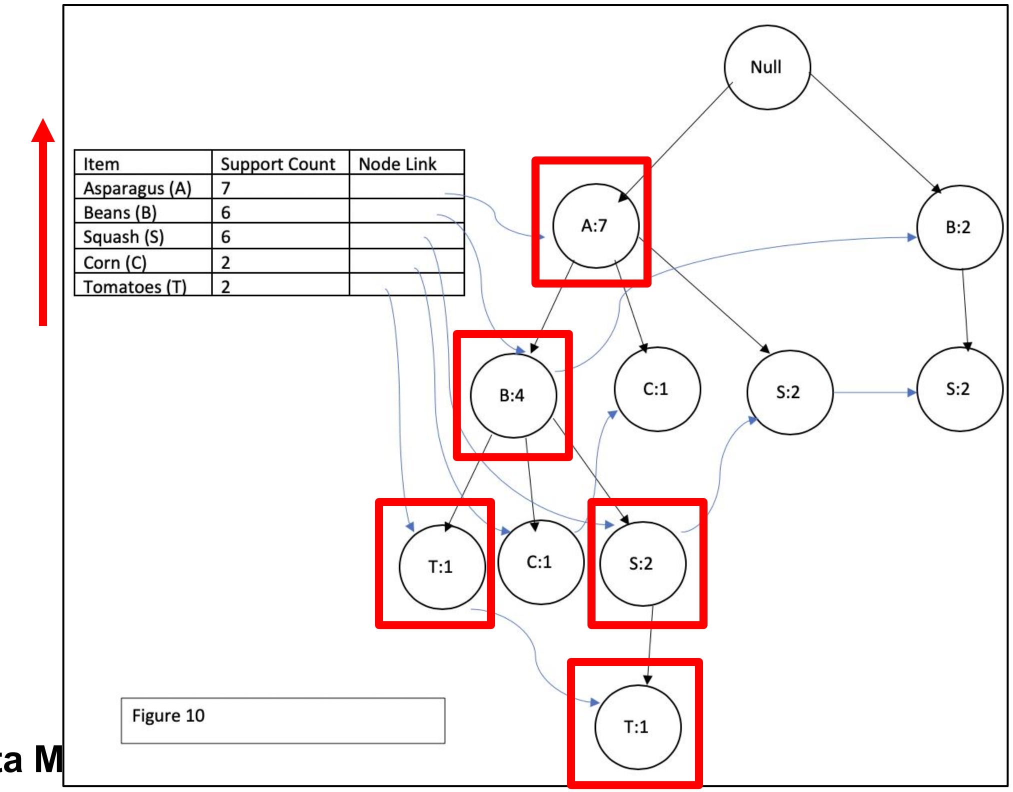
- FP-growth is an algorithm that generates frequent itemsets from an FP-tree by exploring the tree in a bottom-up fashion
  - The bottom-up strategy is equivalent to the suffix based approach early mentioned

FP-Growth: Frequent Itemset Generation

Item	Conditional Pattern base	Conditional FP-tree	Frequent Pattern Generation
Tomatoes (T)	{{A,B:1}, {A,B,S:1}}	<A:2, B:2>	{A,T:2}, {B,T:2}, {A,B,T:2}
Corn (C)	{{A,B:1}, {A:1}}	<A:2>	{A,C:2}
Squash (S)	{{A,B:2}, {A:2}, {B:2}}	<A:4,B:2>, <B:2>	{A,S:4}, {B,S:4}, {A,B,S:2}
Bean (B)	{{A:4}}	<A:4>	{A,B:4}

min-sup = 2

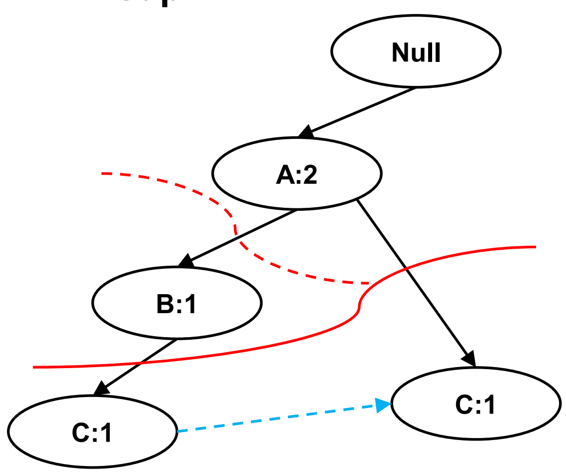


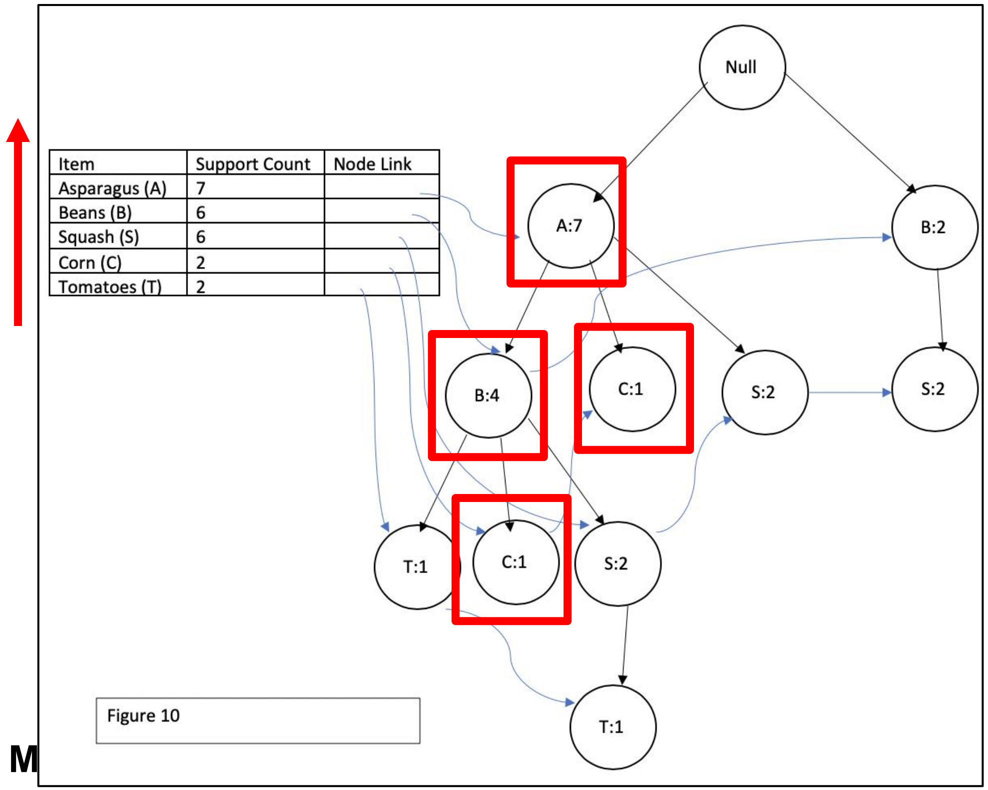


FP-Growth: Frequent Itemset Generation

Item	Conditional Pattern base	Conditional FP-tree	Frequent Pattern Generation
Tomatoes (T)	{{A,B:1}, {A,B,S:1}}	<A:2, B:2>	{A,T:2}, {B,T:2}, {A,B,T:2}
Corn (C)	{{A,B:1}, {A:1}}	<A:2>	{A,C:2}
Squash (S)	{{A,B:2}, {A:2}, {B:2}}	<A:4,B:2>, <B:2>	{A,S:4}, {B,S:4}, {A,B,S:2}
Bean (B)	{{A:4}}	<A:4>	{A,B:4}

min-sup = 2

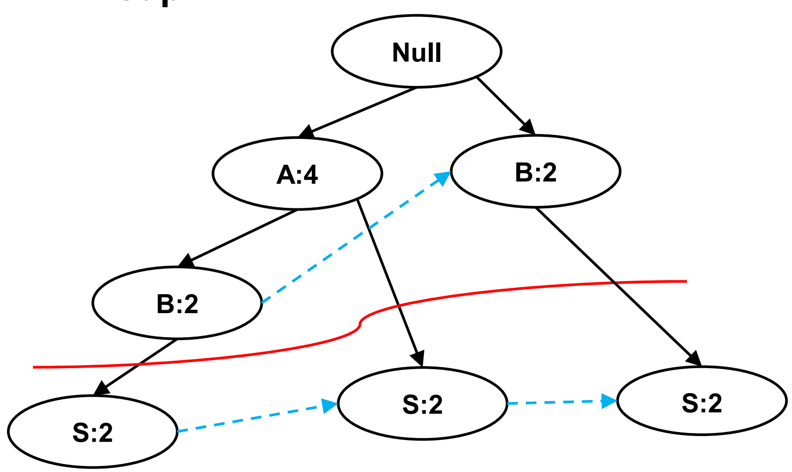


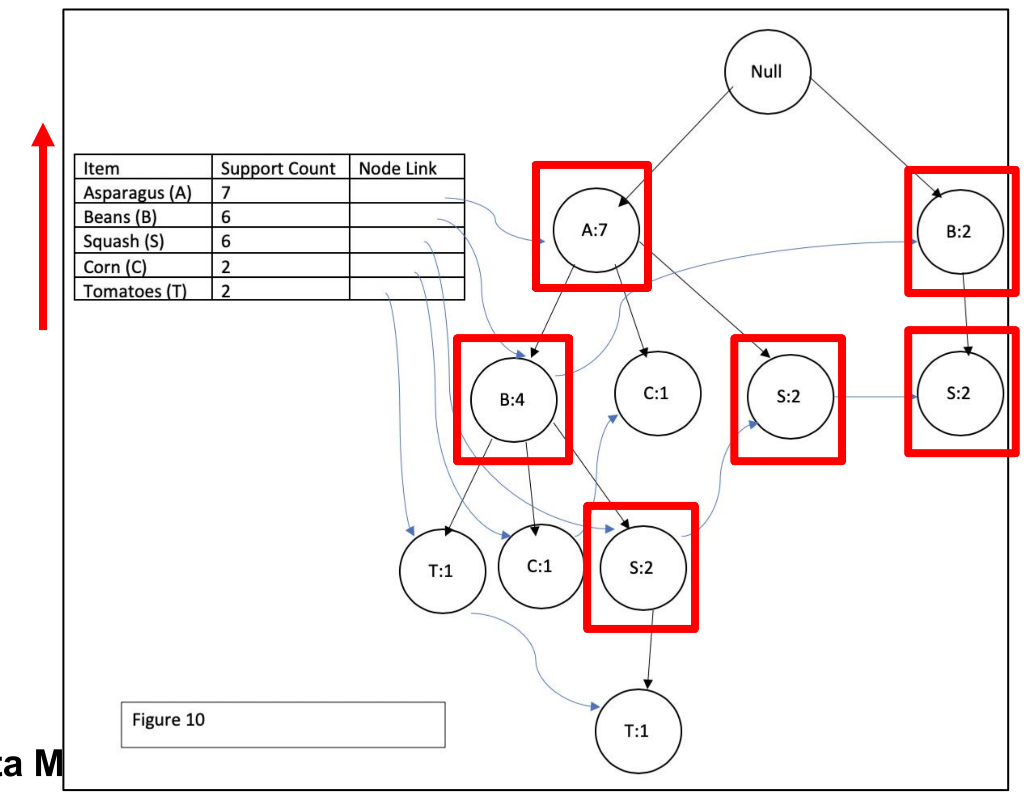


FP-Growth: Frequent Itemset Generation

Item	Conditional Pattern base	Conditional FP-tree	Frequent Pattern Generation
Tomatoes (T)	{{A,B:1}, {A,B,S:1}}	<A:2, B:2>	{A,T:2}, {B,T:2}, {A,B,T:2}
Corn (C)	{{A,B:1}, {A:1}}	<A:2>	{A,C:2}
Squash (S)	{{A,B:2}, {A:2}, {B:2}}	<A:4,B:2>, <B:2>	{A,S:4}, {B,S:4}, {A,B,S:2}
Bean (B)	{{A:4}}	<A:4>	{A,B:4}

min-sup = 2

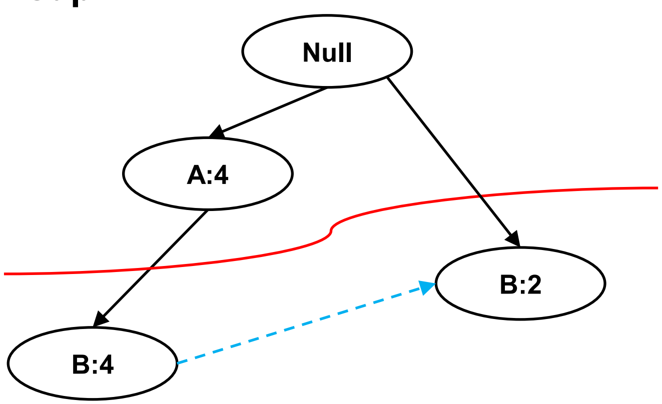




FP-Growth: Frequent Itemset Generation

Item	Conditional Pattern base	Conditional FP-tree	Frequent Pattern Generation
Tomatoes (T)	{{A,B:1}, {A,B,S:1}}	<A:2, B:2>	{A,T:2}, {B,T:2}, {A,B,T:2}
Corn (C)	{{A,B:1}, {A:1}}	<A:2>	{A,C:2}
Squash (S)	{{A,B:2}, {A:2}, {B:2}}	<A:4,B:2>, <B:2>	{A,S:4}, {B,S:4}, {A,B,S:2}
Bean (B)	{{A:4}}	<A:4>	{A,B:4}

min-sup = 2



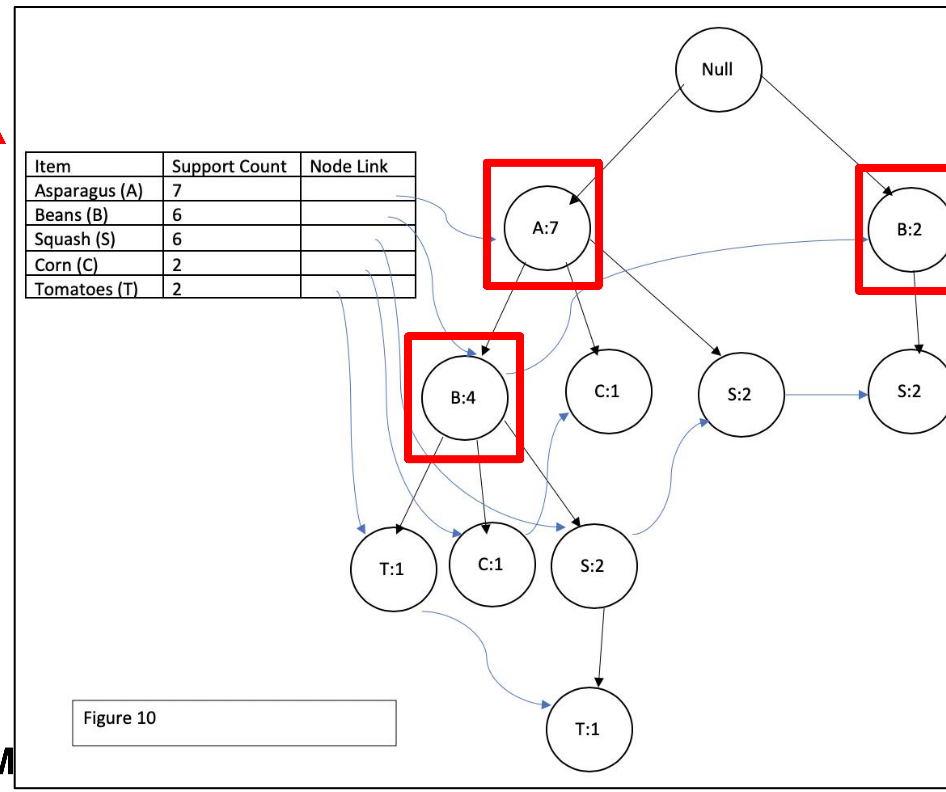


Figure 10

Pattern Evaluation

- Association rule algorithms can produce large number of rules

- Interestingness measures can be used to prune/rank the patterns
  - In the original formulation, support & confidence are the only measures used

Computing Interestingness Measure

- Given $X \rightarrow Y$ or {X,Y}, information needed to compute interestingness can be obtained from a contingency table

Contingency table

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	N

f_{11}: support of X and Y
f_{10}: support of X and $\overline{Y}$
f_{01}: support of $\overline{X}$ and Y
f_{00}: support of $\overline{X}$ and $\overline{Y}$

Used to define various measures (objetivas)

- support, confidence, Gini, entropy, etc.

Drawback of Confidence

Customers	Tea	Coffee	…
C1	0	1	…
C2	1	0	…
C3	1	1	…
C4	1	0	…
…			

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea $\rightarrow$ Coffee

Confidence $\cong$ P(Coffee|Tea) = 15/20 = 0.75

Confidence > 50%, meaning people who drink tea are more likely to drink coffee than not drink coffee

So rule seems reasonable

Drawback of Confidence

Correlação negativa

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 15/20 = 0.75

but P(Coffee) = 0.9, which means knowing that a person drinks tea reduces the probability that the person drinks coffee!

$\Rightarrow$ Note that P(Coffee|$\overline{\text{Tea}}$) = 75/80 = 0.9375

Drawback of Confidence

Correlação positiva

	Honey	$\overline{\text{Honey}}$	
Tea	100	100	200
$\overline{\text{Tea}}$	20	780	800
	120	880	1000

Association Rule: Tea $\rightarrow$ Honey

Confidence= P(Honey|Tea) = 100/200 = 0.50 //might be rejected using a threshold of 70%, for example (would be falsely rejected)

but P(Honey) = 0.12, which means knowing that a person drinks tea significantly increases the probability of using honey from 12% to 50%!

$\Rightarrow$ Note that P(Honey|$\overline{\text{Tea}}$) = 20/800 = 0.025 (2,5%)

Statistical Independence

- The criterion

  $$\text{confidence}(X \rightarrow Y) = \text{support}(Y)$$

  Grande parte das medidas tentam detectar se existe correlação entre o antecedente e o consequente ou se os mesmos são independentes (se não existe relação entre os mesmos)

  is equivalent to:
  - $P(Y|X) = P(Y)$
  - $P(X,Y) = P(X) \times P(Y)$//X and Y are statistically independent, i.e., there is no relationship between the occurrences of X and Y

If $P(X,Y) > P(X) \times P(Y)$: X & Y are positively correlated

If $P(X,Y) < P(X) \times P(Y)$: X & Y are negatively correlated

Measures that take into account statistical dependence

Medidas simétricas e assimétricas

$$Lift = \frac{P(Y \mid X)}{P(Y)}$$ **[0…1…∞]**

$$PS = P(X,Y) - P(X)P(Y)$$ **[-0,25…0…0,25]**

$$\phi - coefficient = \frac{P(X,Y) - P(X)P(Y)}{\sqrt{P(X)[1-P(X)]P(Y)[1-P(Y)]}}$$

[-1…0…1]

Example: Lift

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

$\Rightarrow$ Lift = 0.75/0.9= 0.8333 (< 1, therefore is negatively associated)

So, is it enough to use confidence/lift for pruning?

There are lots of measures proposed in the literature

[Artigo](#)
Behavior-based clustering and analysis of interestingness measures for association rule mining, 2014

#	Measure	Formula
1	ϕ-coefficient	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
2	Goodman-Kruskal's (λ)	$\frac{\sum_j \max_k P(A_j,B_k)+\sum_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-\max_k P(B_k)}$
3	Odds ratio (α)	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(A,\overline{B})P(\overline{A},B)}$
4	Yule's Q	$\frac{P(A,B)P(\overline{AB})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{AB})+P(A,\overline{B})P(\overline{A},B)} = \frac{\alpha-1}{\alpha+1}$
5	Yule's Y	$\frac{\sqrt{P(A,B)P(\overline{AB})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{AB})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}} = \frac{\sqrt{\alpha}-1}{\sqrt{\alpha}+1}$
6	Kappa (κ)	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
7	Mutual Information (M)	$\frac{\sum_i \sum_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i)\log P(A_i), -\sum_j P(B_j)\log P(B_j))}$
8	J-Measure (J)	$\max\Big(P(A,B)\log(\frac{P(B\|A)}{P(B)}) + P(A\overline{B})\log(\frac{P(\overline{B}\|A)}{P(\overline{B})}),$ $P(A,B)\log(\frac{P(A\|B)}{P(A)}) + P(\overline{A}B)\log(\frac{P(\overline{A}\|B)}{P(A)})\Big)$
9	Gini index (G)	$\max\Big(P(A)[P(B\|A)^2+P(\overline{B}\|A)^2]+P(\overline{A})[P(B\|\overline{A})^2+P(\overline{B}\|\overline{A})^2]$ $-P(B)^2-P(\overline{B})^2,$ $P(B)[P(A\|B)^2+P(\overline{A}\|B)^2]+P(\overline{B})[P(A\|\overline{B})^2+P(\overline{A}\|\overline{B})^2]$ $-P(A)^2-P(\overline{A})^2\Big)$
10	Support (s)	$P(A,B)$
11	Confidence (c)	$\max(P(B\|A),P(A\|B))$
12	Laplace (L)	$\max\Big(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2}\Big)$
13	Conviction (V)	$\max\Big(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})}\Big)$
14	Interest (I)	$\frac{P(A,B)}{P(A)P(B)}$
15	cosine (IS)	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
16	Piatetsky-Shapiro's (PS)	$P(A,B)-P(A)P(B)$
17	Certainty factor (F)	$\max\Big(\frac{P(B\|A)-P(B)}{1-P(B)}, \frac{P(A\|B)-P(A)}{1-P(A)}\Big)$
18	Added Value (AV)	$\max(P(B\|A)-P(B),P(A\|B)-P(A))$
19	Collective strength (S)	$\frac{P(A,B)+P(\overline{AB})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{AB})}$
20	Jaccard (ζ)	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
21	Klosgen (K)	$\sqrt{P(A,B)}\max(P(B\|A)-P(B),P(A\|B)-P(A))$

Comparing Different Measures

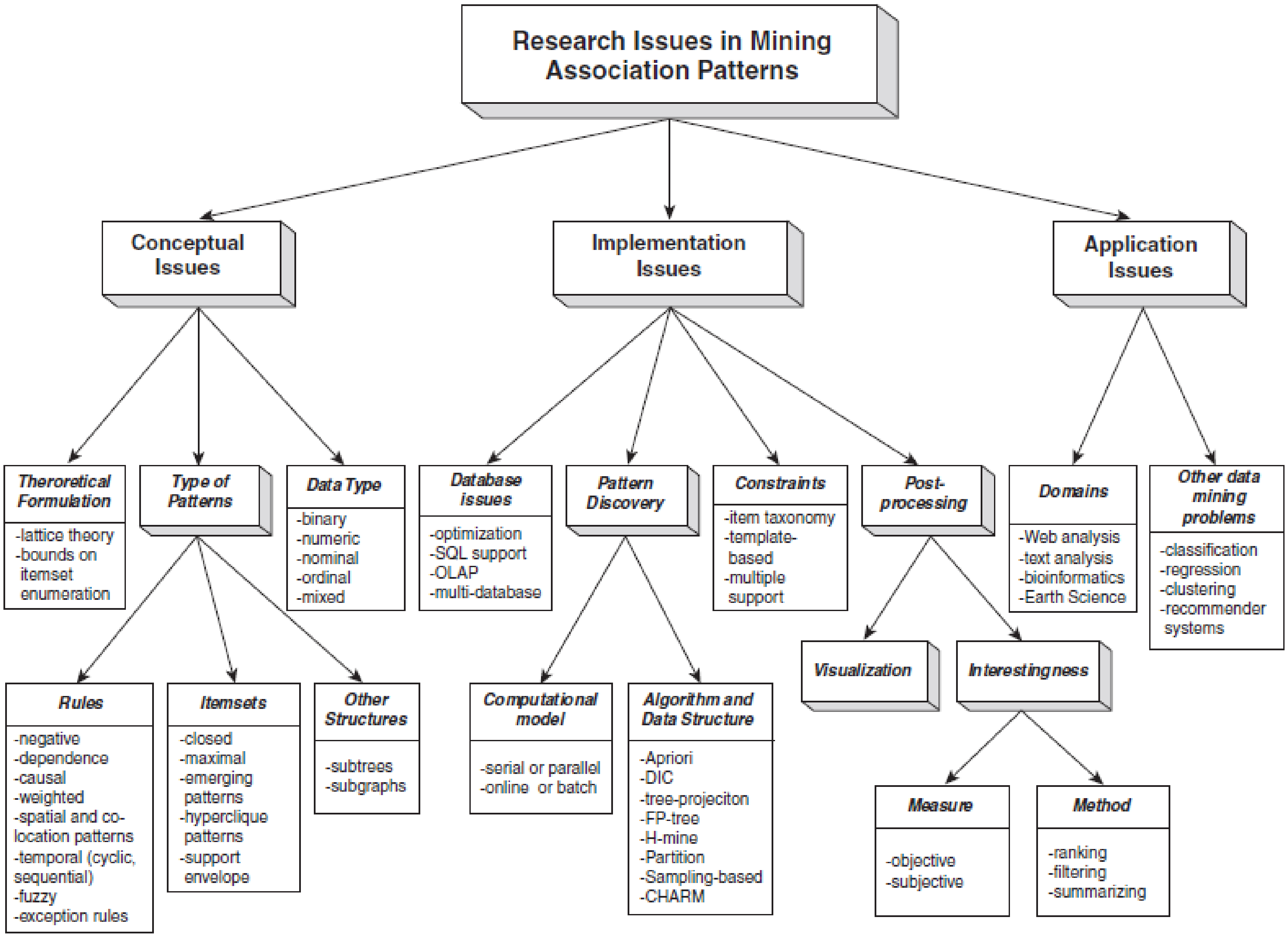
10 examples of contingency tables:

Example	f_{11}	f_{10}	f_{01}	f_{00}
E1	8123	83	424	1370
E2	8330	2	622	1046
E3	9481	94	127	298
E4	3954	3080	5	2961
E5	2886	1363	1320	4431
E6	1500	2000	500	6000
E7	4000	2000	1000	3000
E8	4000	2000	2000	2000
E9	1720	7121	5	1154
E10	61	2483	4	7452

Rankings of contingency tables using various measures:

#	ϕ	λ	α	Q	Y	κ	M	J	G	s	c	L	V	I	IS	PS	F	AV	S	ζ	K
E1	1	1	3	3	3	1	2	2	1	3	5	5	4	6	2	2	4	6	1	2	5
E2	2	2	1	1	1	2	1	3	2	2	1	1	1	8	3	5	1	8	2	3	6
E3	3	3	4	4	4	3	3	8	7	1	4	4	6	10	1	8	6	10	3	1	10
E4	4	7	2	2	2	5	4	1	3	6	2	2	2	4	4	1	2	3	4	5	1
E5	5	4	8	8	8	4	7	5	4	7	9	9	9	3	6	3	9	4	5	6	3
E6	6	6	7	7	7	7	6	4	6	9	8	8	7	2	8	6	7	2	7	8	2
E7	7	5	9	9	9	6	8	6	5	4	7	7	8	5	5	4	8	5	6	4	4
E8	8	9	10	10	10	8	10	10	8	4	10	10	10	9	7	7	10	9	8	7	9
E9	9	9	5	5	5	9	9	7	9	8	3	3	3	7	9	9	3	7	9	9	8
E10	10	8	6	6	6	10	5	9	10	10	6	6	5	1	10	10	5	1	10	10	7

An overview of the various research directions in association analysis



Considerações Finais

- SPMF: http://www.philippe-fournier-viger.com/spmf/
- A Survey of Sequential Pattern Mining, 2017 [http://www.philippe-fournier-viger.com/dspr-paper5.pdf]
- A Survey of Itemset Mining, 2017 [http://www.philippe-fournier-viger.com/Survey_Itemset_Mining.pdf]