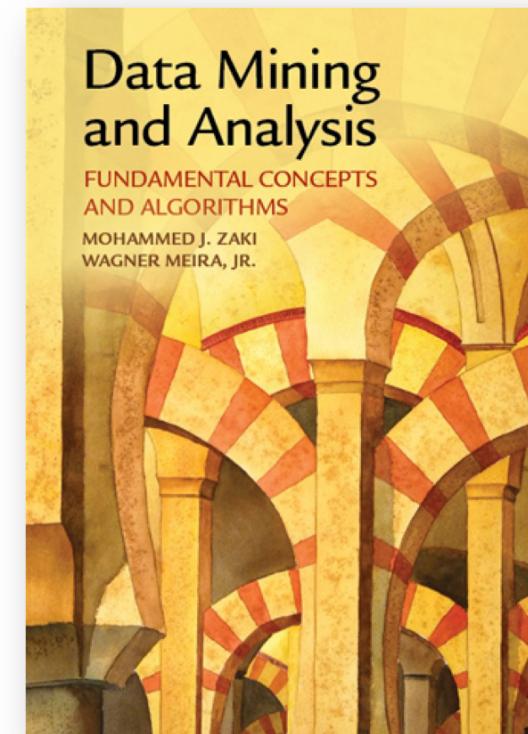




Association Rules

Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)

- “Data Mining and Analysis” by Zaki & Meira
 - Chapter 8
 - Section 9.1
 - Chapter 10 up to
Section 10.2.1 included
 - Section 12.1
- <http://www.dataminingbook.info>





Market-Basket Transactions



4

Bread
Peanuts
Milk
Fruit
Jam

Bread
Jam
Soda
Chips
Milk
Fruit

Steak
Jam
Soda
Chips
Bread

Jam
Soda
Peanuts
Milk
Fruit

Jam
Soda
Chips
Milk
Bread

Fruit
Soda
Chips
Milk

Fruit
Soda
Peanuts
Milk

Fruit
Peanuts
Cheese
Yogurt

- Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories
- Applications
 - Basket data analysis
 - Cross-marketing
 - Catalog design
 - ...

- 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
- Examples
 - $\{bread\} \Rightarrow \{milk\}$
 - $\{soda\} \Rightarrow \{chips\}$
 - $\{bread\} \Rightarrow \{jam\}$

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

- Support
 - Fraction of transactions that contain an itemset
 - $s(\{\text{Milk, Bread}\}) = 3/8$
 - $s(\{\text{Soda, Chips}\}) = 4/8$
- Frequent Itemset
 - An itemset whose support is greater than or equal to a minsup threshold
- Itemset
 - A collection of one or more items, e.g., {milk, bread, jam}
 - k-itemset, an itemset that contains k items
- Support count (σ)
 - Frequency of occurrence of an itemset
 - $\sigma(\{\text{Milk, Bread}\}) = 3$
 - $\sigma(\{\text{Soda, Chips}\}) = 4$

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

- Implication of the form $X \Rightarrow Y$, where X and Y are itemsets
- Example, $\{\text{bread}\} \Rightarrow \{\text{milk}\}$
- Rule Evaluation Metrics: Support & Confidence
- 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

$$s = \frac{\sigma(\{\text{Bread, Milk}\})}{\# \text{ of transactions}} = 0.38$$

$$c = \frac{\sigma(\{\text{Bread, Milk}\})}{\sigma(\{\text{Bread}\})} = 0.75$$

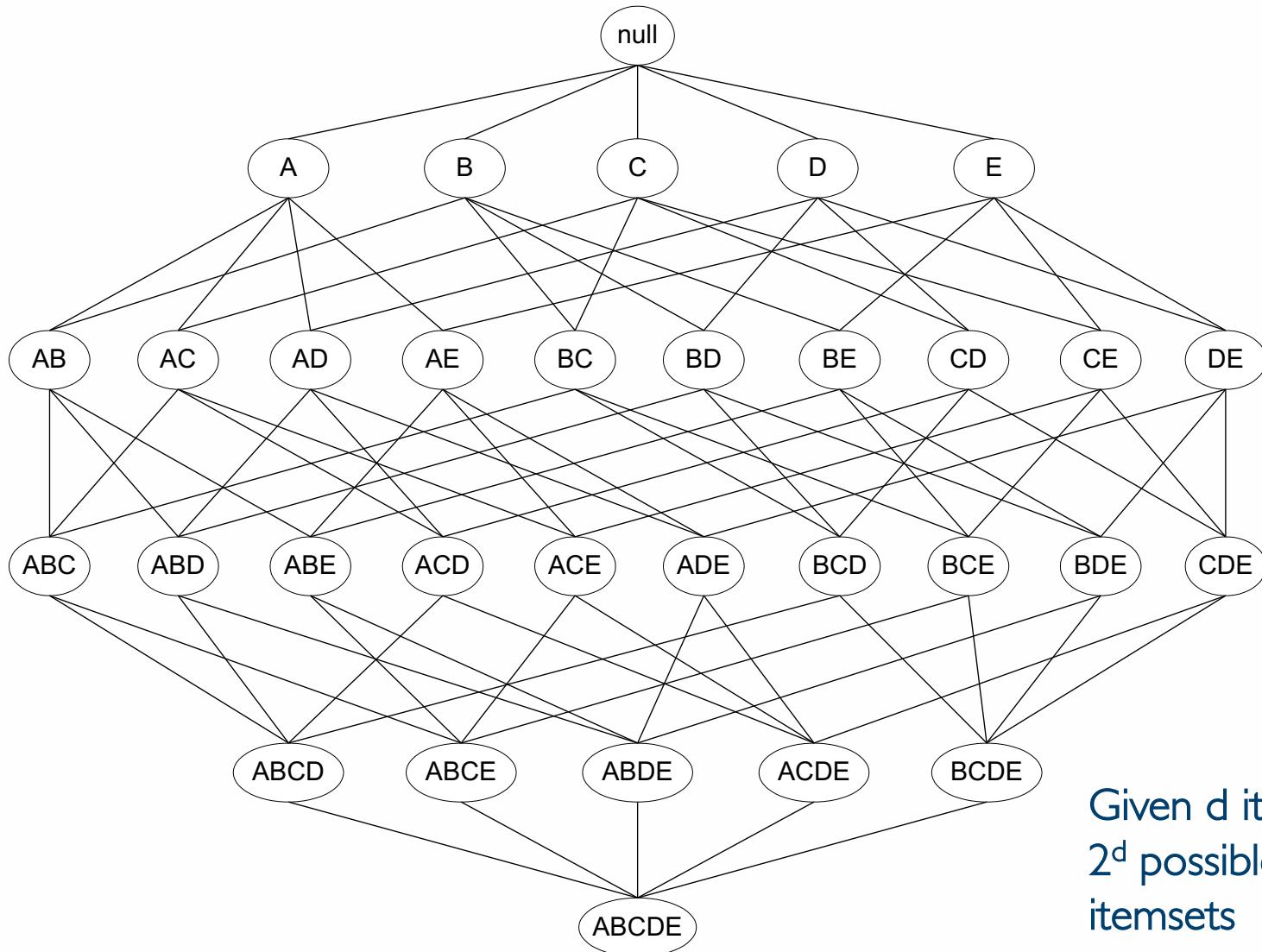
What is the Goal?

- 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
- Brute-force approach is computationally prohibitive!

- $\{\text{Bread, Jam}\} \Rightarrow \{\text{Milk}\}$ s=0.4 c=0.75
- $\{\text{Milk, Jam}\} \Rightarrow \{\text{Bread}\}$ s=0.4 c=0.75
- $\{\text{Bread}\} \Rightarrow \{\text{Milk, Jam}\}$ s=0.4 c=0.75
- $\{\text{Jam}\} \Rightarrow \{\text{Bread, Milk}\}$ s=0.4 c=0.6
- $\{\text{Milk}\} \Rightarrow \{\text{Bread, Jam}\}$ s=0.4 c=0.5
- All the above rules are binary partitions of the same itemset $\{\text{Milk, Bread, Jam}\}$
- Rules originating from the same itemset have identical support but can have different confidence
- We can decouple the support and confidence requirements!

Mining Association Rules: Two Step Approach

- Frequent Itemset Generation
 - Generate all itemsets whose support $\geq \text{minsup}$
- Rule Generation
 - Generate high confidence rules from frequent itemset
 - Each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is computationally expensive



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

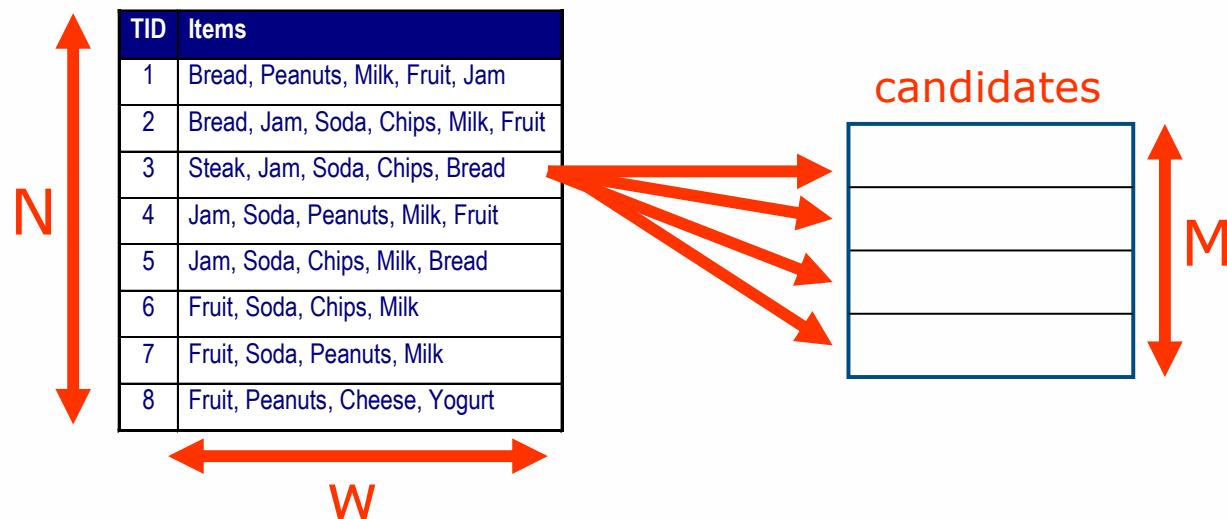
BRUTEFORCE ($D, \mathcal{I}, minsup$):

```
1  $\mathcal{F} \leftarrow \emptyset$  // set of frequent itemsets
2 foreach  $X \subseteq \mathcal{I}$  do
3    $sup(X) \leftarrow \text{COMPUTESUPPORT}(X, D)$ 
4   if  $sup(X) \geq minsup$  then
5      $\mathcal{F} \leftarrow \mathcal{F} \cup \{(X, sup(X))\}$ 
6 return  $\mathcal{F}$ 
```

COMPUTESUPPORT (X, D):

```
1  $sup(X) \leftarrow 0$ 
2 foreach  $\langle t, i(t) \rangle \in D$  do
3   if  $X \subseteq i(t)$  then
4      $sup(X) \leftarrow sup(X) + 1$ 
5 return  $sup(X)$ 
```

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



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

- Reduce the number of candidates (M)
 - Complete search has $M=2^d$
 - 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
 - No need to match every candidate against every transaction

- 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:

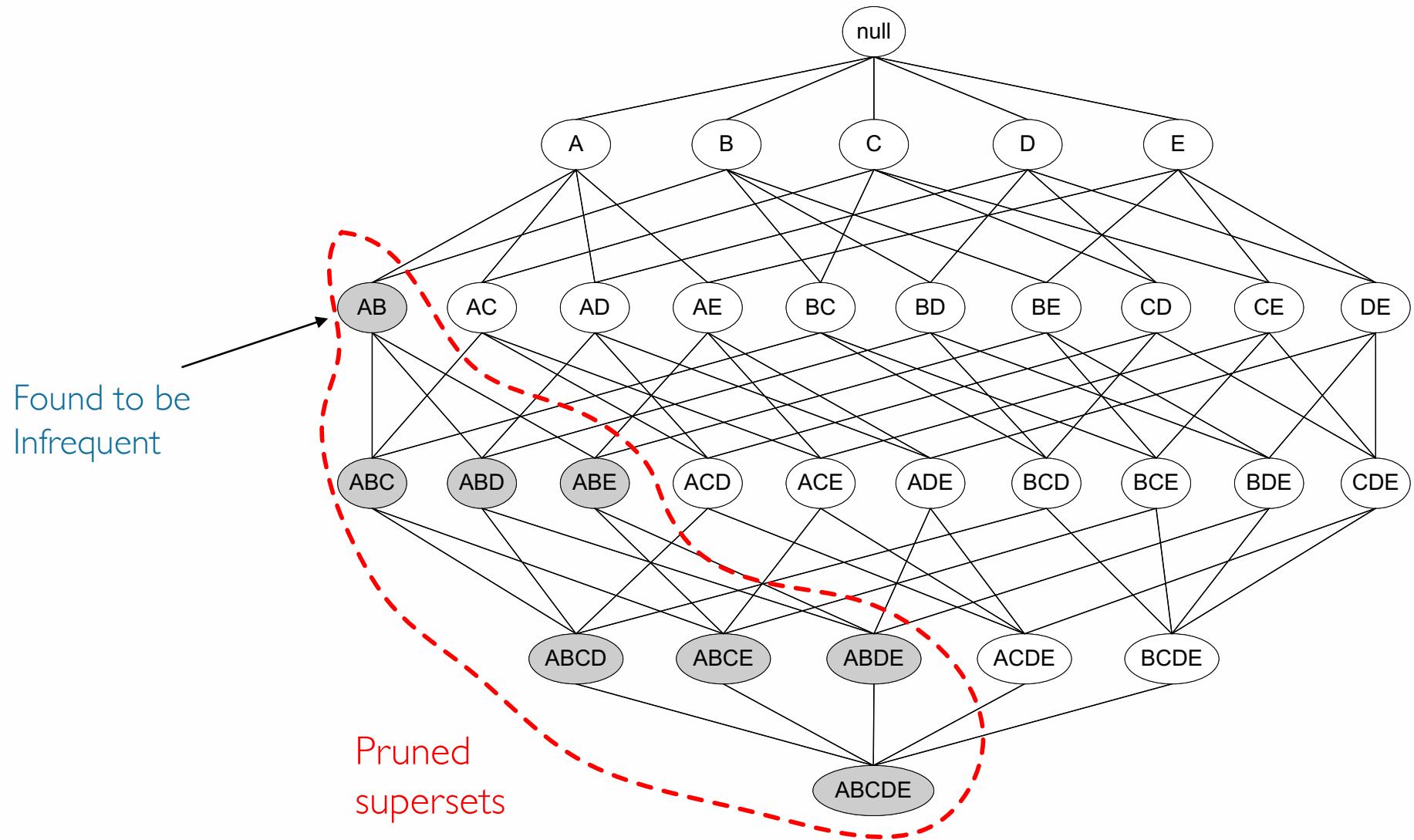
$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$



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

Illustrating Apriori Principle

17



How does the Apriori principle work?

18

Item	Count
Bread	4
Peanuts	4
Milk	6
Fruit	6
Jam	5
Soda	6
Chips	4
Steak	1
Cheese	1
Yogurt	1

1-itemsets

Minimum Support = 4

2-Itemset	Count
Bread, Jam	4
Peanuts, Fruit	4
Milk, Fruit	5
Milk, Jam	4
Milk, Soda	5
Fruit, Soda	4
Jam, Soda	4
Soda, Chips	4

2-itemsets

3-Itemset	Count
Milk, Fruit, Soda	4

3-itemsets

Example

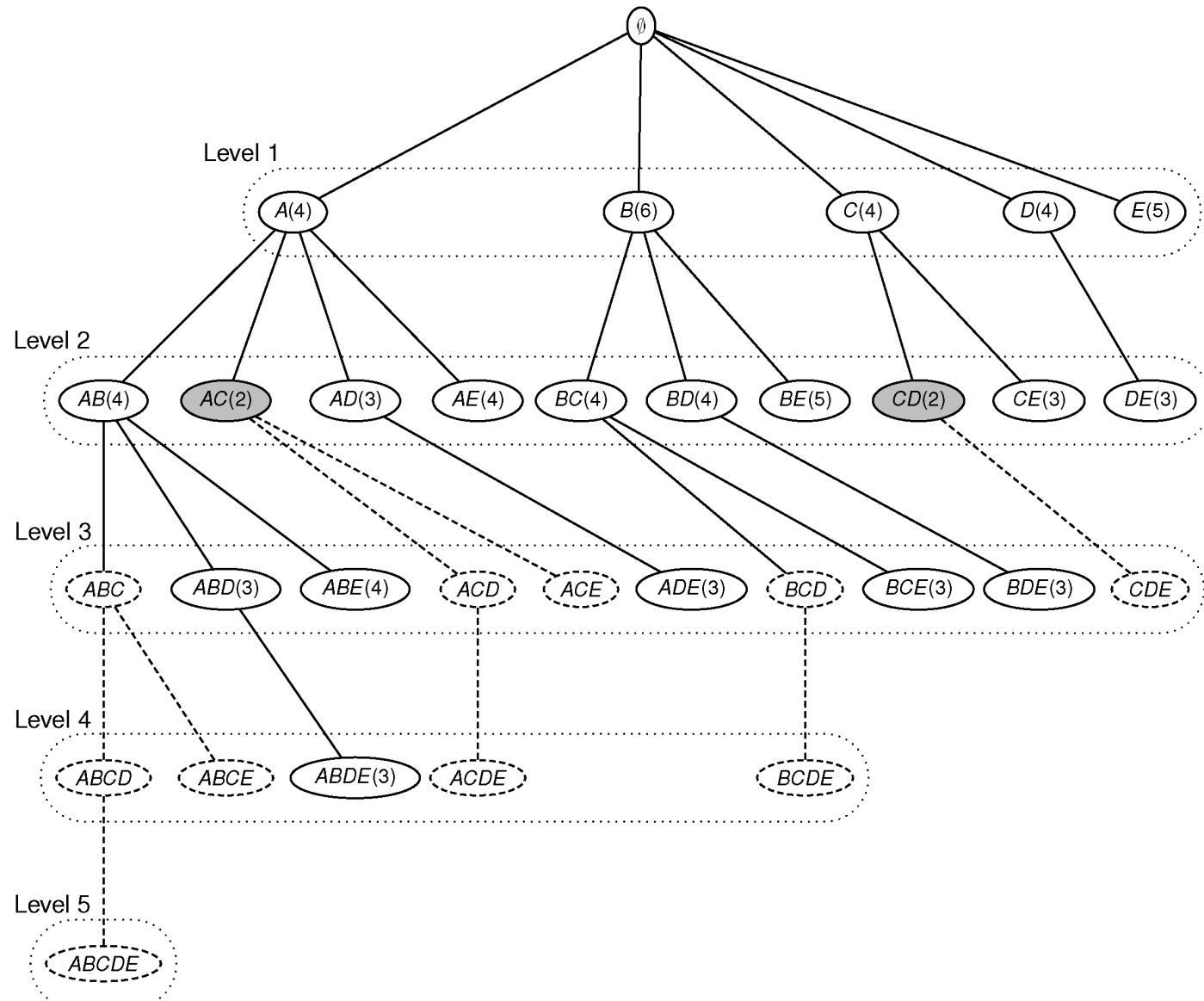
- Given the following database and a min support of 3, generate all the frequent itemsets

D	A	B	C	D	E
1	1	1	0	1	1
2	0	1	1	0	1
3	1	1	0	1	1
4	1	1	1	0	1
5	1	1	1	1	1
6	0	1	1	1	0

Binary Database

t	i(t)
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

Transaction Database



- Let $k=1$
- Generate frequent itemsets of length l
- Repeat until no new frequent itemsets are identified
 - Generate length $(k+1)$ candidate itemsets from length k frequent itemsets
 - Prune candidate itemsets containing subsets of length k that are infrequent
 - Count the support of each candidate by scanning the database
 - Eliminate candidates that are infrequent, leaving only those that are frequent

APRIORI ($\mathbf{D}, \mathcal{I}, \text{minsup}$):

```
1  $\mathcal{F} \leftarrow \emptyset$ 
2  $\mathcal{C}^{(1)} \leftarrow \{\emptyset\}$  // Initial prefix tree with single items
3 foreach  $i \in \mathcal{I}$  do Add  $i$  as child of  $\emptyset$  in  $\mathcal{C}^{(1)}$  with  $\text{sup}(i) \leftarrow 0$ 
4  $k \leftarrow 1$  //  $k$  denotes the level
5 while  $\mathcal{C}^{(k)} \neq \emptyset$  do
6   COMPUTESUPPORT ( $\mathcal{C}^{(k)}, \mathbf{D}$ )
7   foreach leaf  $X \in \mathcal{C}^{(k)}$  do
8     if  $\text{sup}(X) \geq \text{minsup}$  then  $\mathcal{F} \leftarrow \mathcal{F} \cup \{(X, \text{sup}(X))\}$ 
9     else remove  $X$  from  $\mathcal{C}^{(k)}$ 
10   $\mathcal{C}^{(k+1)} \leftarrow \text{EXTENDPREFIXTREE } (\mathcal{C}^{(k)})$ 
11   $k \leftarrow k + 1$ 
12 return  $\mathcal{F}^{(k)}$ 
```

COMPUTESUPPORT ($\mathcal{C}^{(k)}$, \mathbf{D}):

```
1 foreach  $\langle t, \mathbf{i}(t) \rangle \in \mathbf{D}$  do
2   foreach  $k$ -subset  $X \subseteq \mathbf{i}(t)$  do
3     if  $X \in \mathcal{C}^{(k)}$  then  $sup(X) \leftarrow sup(X) + 1$ 
```

EXTENDPREFIXTREE ($\mathcal{C}^{(k)}$):

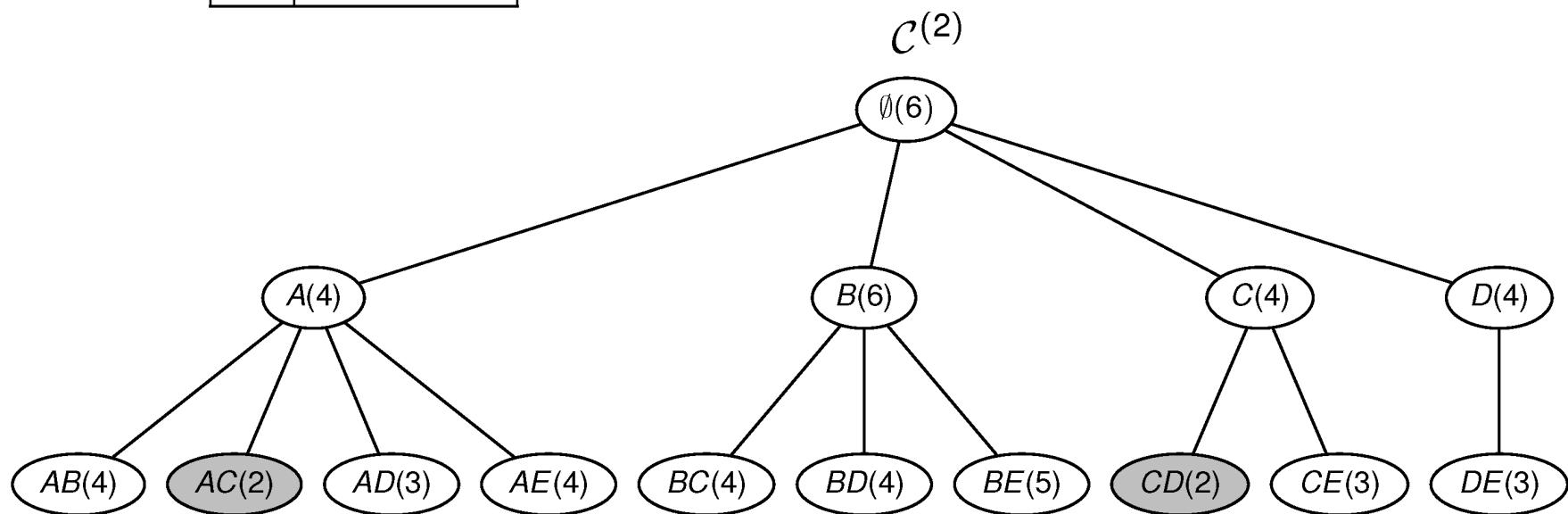
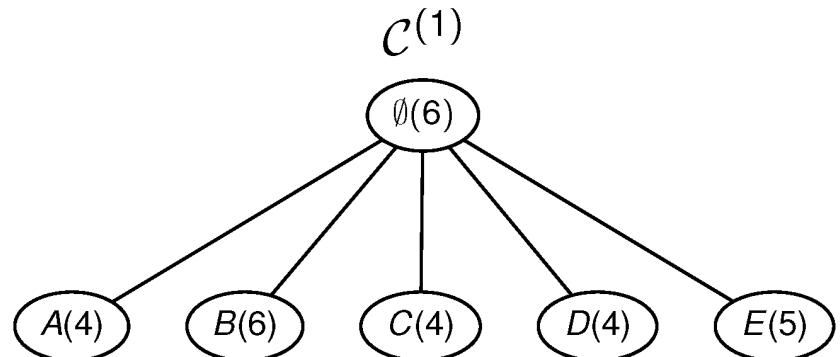
```
1 foreach leaf  $X_a \in \mathcal{C}^{(k)}$  do
2   foreach leaf  $X_b \in \text{SIBLING}(X_a)$ , such that  $b > a$  do
3      $X_{ab} \leftarrow X_a \cup X_b$ 
        // prune candidate if there are any infrequent
        subsets
4     if  $X_j \in \mathcal{C}^{(k)}$ , for all  $X_j \subset X_{ab}$ , such that  $|X_j| = |X_{ab}| - 1$  then
5       Add  $X_{ab}$  as child of  $X_a$  with  $sup(X_{ab}) \leftarrow 0$ 
6   if no extensions from  $X_a$  then
7     remove  $X_a$ , and all ancestors of  $X_a$  with no extensions, from  $\mathcal{C}^{(k)}$ 
8 return  $\mathcal{C}^{(k)}$ 
```

Apriori Algorithm Example

24

D

t	$i(t)$
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD



Eclat

- Leverages the tidsets directly for support computation.
- The support of a candidate itemset can be computed by intersecting the tidsets of suitably chosen subsets.
- Given $t(X)$ and $t(Y)$ for any two frequent itemsets X and Y , then $t(XY) = t(X) \wedge t(Y)$
- And $\text{sup}(XY) = |t(XY)|$

D	A	B	C	D	E
1	1	1	0	1	1
2	0	1	1	0	1
3	1	1	0	1	1
4	1	1	1	0	1
5	1	1	1	1	1
6	0	1	1	1	0

Binary Database

<i>t</i>	<i>i(t)</i>
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

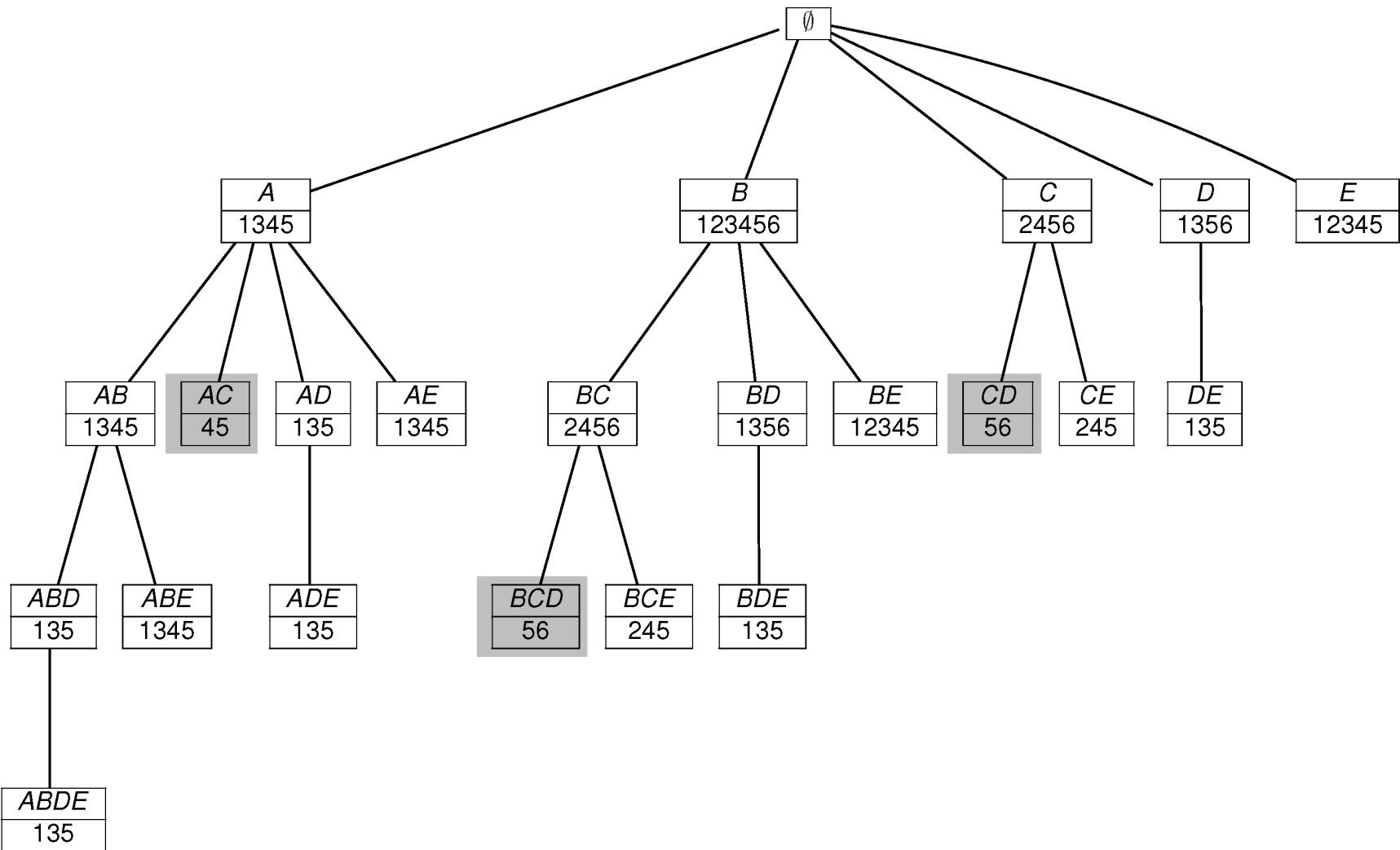
Transaction Database

<i>t(x)</i>				
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
1	1	2	1	1
3	2	4	3	2
4	3	5	5	3
5	4	6	6	4
	5			5
	6			

Vertical Database

Example

27



```

// Initial Call:  $\mathcal{F} \leftarrow \emptyset, P \leftarrow \{\langle i, \mathbf{t}(i) \rangle \mid i \in \mathcal{I}, |\mathbf{t}(i)| \geq \text{minsup}\}$ 
ECLAT ( $P, \text{minsup}, \mathcal{F}$ ):
1 foreach  $\langle X_a, \mathbf{t}(X_a) \rangle \in P$  do
2    $\mathcal{F} \leftarrow \mathcal{F} \cup \{(X_a, \text{sup}(X_a))\}$ 
3    $P_a \leftarrow \emptyset$ 
4   foreach  $\langle X_b, \mathbf{t}(X_b) \rangle \in P$ , with  $X_b > X_a$  do
5      $X_{ab} = X_a \cup X_b$ 
6      $\mathbf{t}(X_{ab}) = \mathbf{t}(X_a) \cap \mathbf{t}(X_b)$ 
7     if  $\text{sup}(X_{ab}) \geq \text{minsup}$  then
8        $P_a \leftarrow P_a \cup \{\langle X_{ab}, \mathbf{t}(X_{ab}) \rangle\}$ 
9   if  $P_a \neq \emptyset$  then ECLAT ( $P_a, \text{minsup}, \mathcal{F}$ )

```

Frequent Patterns Mining Without Candidate Generation

- The core of the Apriori algorithm
 - Use frequent $(k-1)$ -itemsets to generate candidate frequent k -itemsets
 - Use database scan and pattern matching to collect counts for the candidate itemsets
- Huge candidate sets:
 - 10^4 frequent 1-itemset will generate 10^7 candidate 2-itemsets
 - To discover a frequent pattern of size 100, e.g., $\{a_1, a_2, \dots, a_{100}\}$, needs to generate $2^{100} \sim 10^{30}$ candidates.
 - Multiple scans of database, it needs $(n+1)$ scans, n is the length of the longest pattern

Mining Frequent Patterns Without Candidate Generation

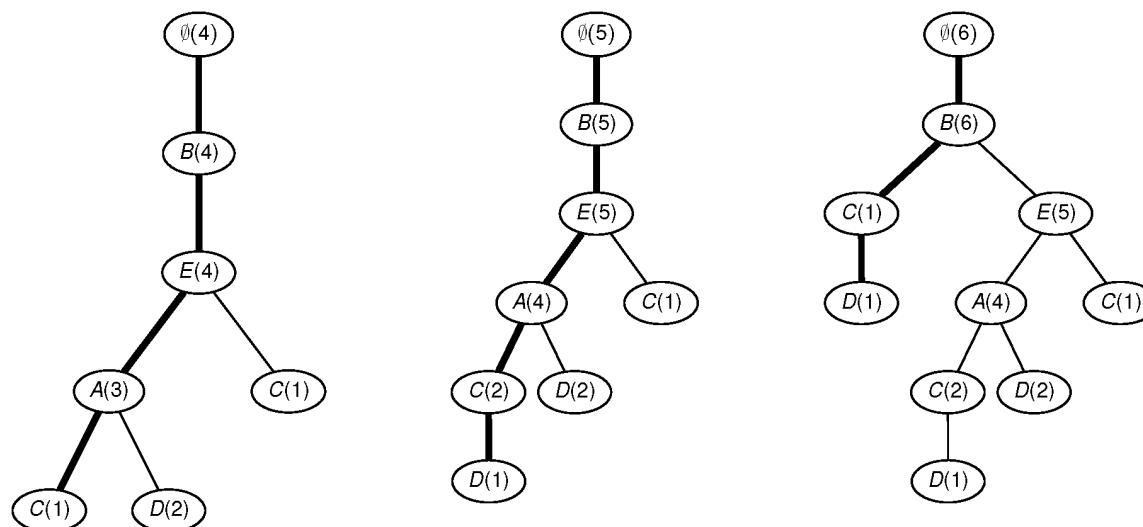
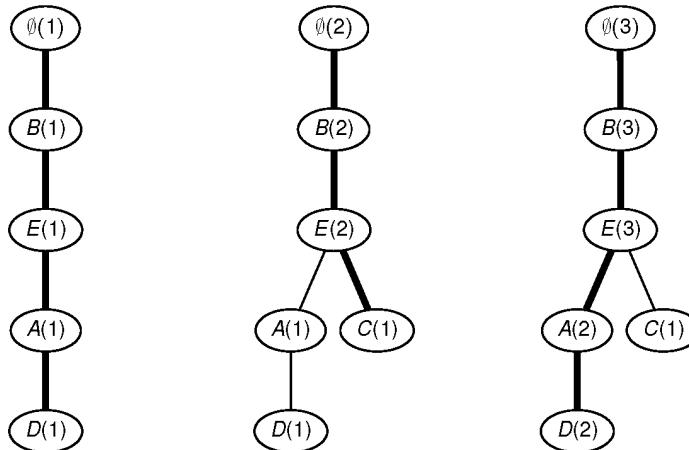
- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
 - Highly condensed, but complete for frequent pattern mining
 - Avoid costly database scans
- Use an efficient, FP-tree-based frequent pattern mining method
- A divide-and-conquer methodology: decompose mining tasks into smaller ones
- Avoid candidate generation: sub-database test only

- Leave the generate-and-test paradigm of Apriori
- Data sets are encoded using a compact structure, the FP-tree
- Frequent itemsets are extracted directly from the FP-tree
- Major Steps to mine FP-tree
 - Construct the frequent pattern tree
 - For each frequent item i compute the projected FP-tree
 - Recursively mine conditional FP-trees and grow frequent patterns obtained so far
 - If the conditional FP-tree contains a single path, simply enumerate all the patterns

Frequent Pattern Tree

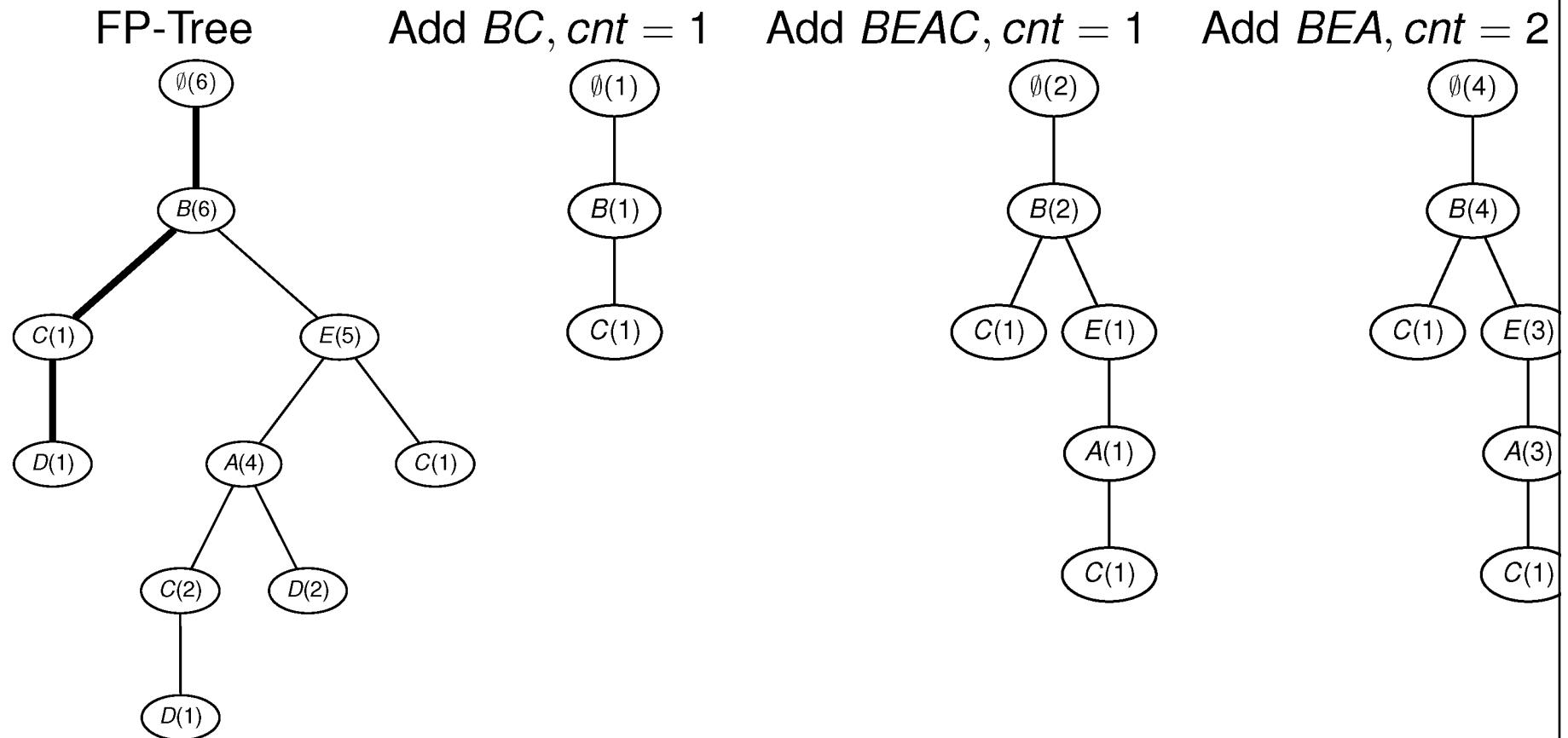
33

Transactions
BEAD
BEC
BEAD
BEAC
BEACD
BCD



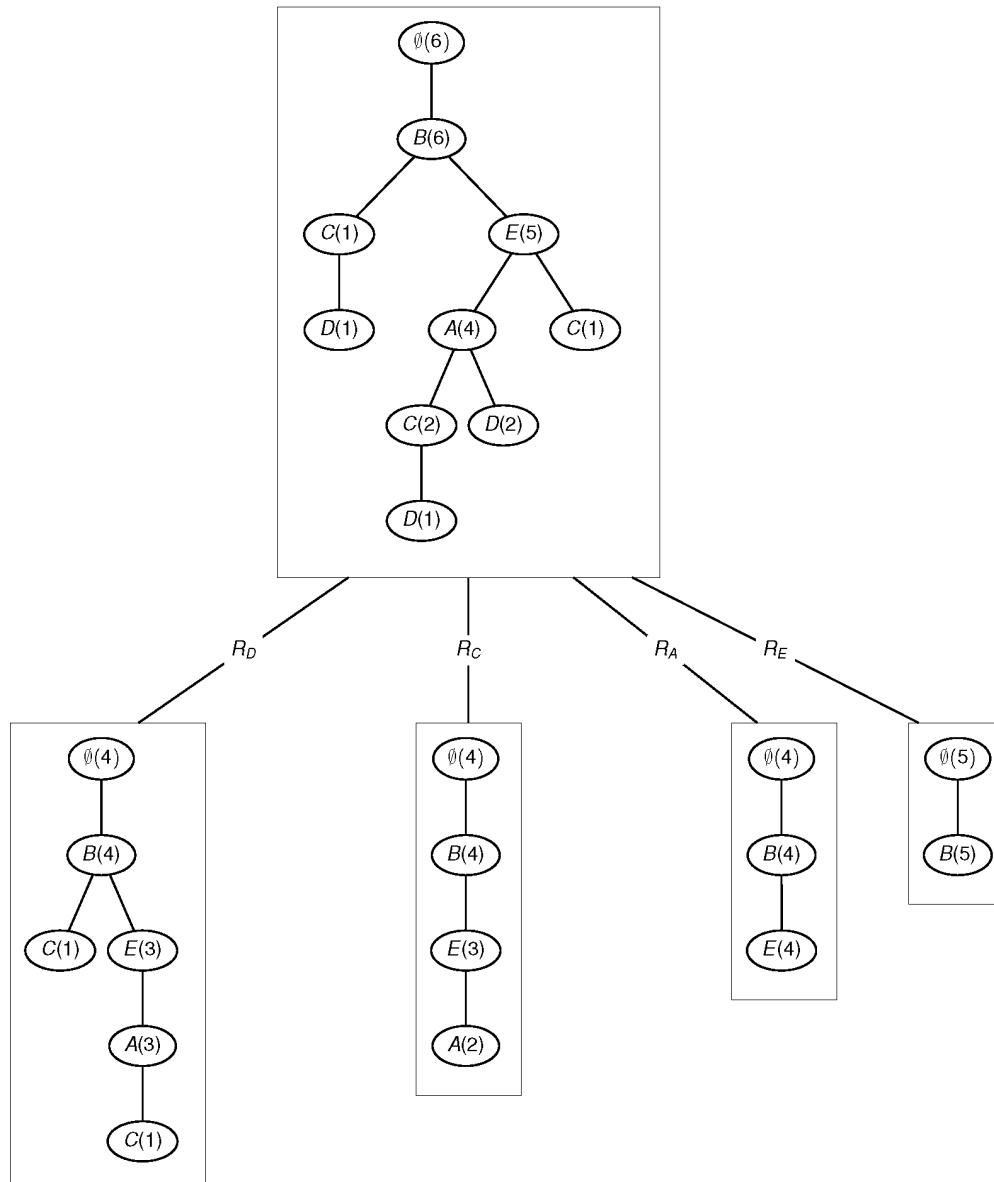
Conditional FP-tree for D

34



All Conditional FP-Trees

35



- Completeness
 - Preserve complete information for frequent pattern mining
 - Never break a long pattern of any transaction
- Compactness
 - Reduce irrelevant info—infrequent items are gone
 - Items in frequency descending order: the more frequently occurring, the more likely to be shared
 - Never be larger than the original database (not count node-links and the count field)

- Divide-and-conquer:
 - decompose both the mining task and DB according to the frequent patterns obtained so far
 - leads to focused search of smaller databases
- Other factors
 - No candidate generation, no candidate test
 - Compressed database: FP-tree structure
 - No repeated scan of entire database
 - Basic ops—counting local freq items and building sub FP-tree, no pattern search and matching

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 $\emptyset \rightarrow L$) 

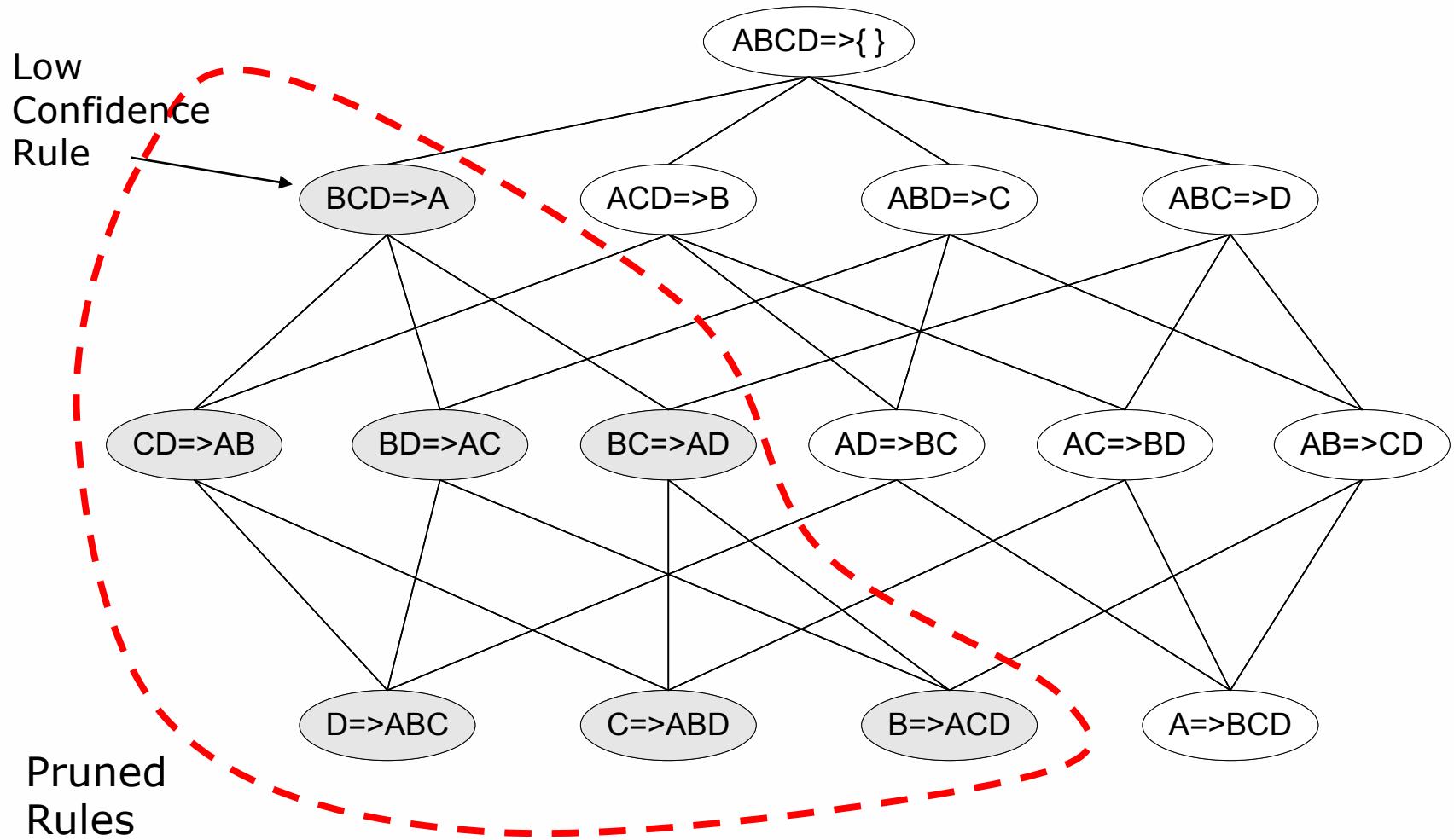
How to efficiently generate rules from frequent itemsets?

40

- 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
- $L = \{A,B,C,D\}$: $c(ABC \Rightarrow D) \geq c(AB \Rightarrow CD) \geq c(A \Rightarrow BCD)$
- Confidence is anti-monotone with respect to the number of items on the right hand side of the rule

Rule Generation for Apriori Algorithm

41



```
ASSOCIATIONRULES ( $\mathcal{F}$ , minconf):
1 foreach  $Z \in \mathcal{F}$ , such that  $|Z| \geq 2$  do
2    $\mathcal{A} \leftarrow \{X \mid X \subset Z, X \neq \emptyset\}$ 
3   while  $\mathcal{A} \neq \emptyset$  do
4      $X \leftarrow$  maximal element in  $\mathcal{A}$ 
5      $\mathcal{A} \leftarrow \mathcal{A} \setminus X$  // remove  $X$  from  $\mathcal{A}$ 
6      $c \leftarrow sup(Z)/sup(X)$ 
7     if  $c \geq minconf$  then
8       print  $X \rightarrow Y$ ,  $sup(Z)$ ,  $c$ 
9     else
10       $\mathcal{A} \leftarrow \mathcal{A} \setminus \{W \mid W \subset X\}$ 
           // remove all subsets of  $X$  from  $\mathcal{A}$ 
```

- Suppose that we computed the following frequent itemsets and we have a minconf of 0.9

sup	itemsets
6	<i>B</i>
5	<i>E, BE</i>
4	<i>A, C, D, AB, AE, BC, BD, ABE</i>
3	<i>AD, CE, DE, ABD, ADE, BCE, BDE, ABDE</i>

- We start from the largest itemset (ABDE) and consider all the possible subsets $\mathcal{A} = \{\text{ABDE}(3), \text{ABD}(3), \text{ABE}(4), \text{ADE}(3), \text{BDE}(3), \text{AB}(4), \text{AD}(3), \text{AE}(4), \text{BD}(4), \text{BE}(5), \text{DE}(3), \text{A}(4), \text{B}(6), \text{D}(4), \text{E}(5)\}$

- The first subset is $X = ABD$, and the confidence of $ABD \Rightarrow E$ is $3/3 = 1.0$, so we output the rule.
- The next subset is $X = ABE$, but the corresponding rule $ABE \Rightarrow D$ is not strong since $c(ABE \Rightarrow D) = 3/4 = 0.75$
- We can thus remove from \mathcal{A} all subsets of ABE
- The updated set of antecedents is therefore
$$\mathcal{A} = \{ADE(3), BDE(3), AD(3), BD(4), DE(3), D(4)\}$$
- When the algorithm ends, it will output the rules,
 $ABD \Rightarrow E \text{ conf}=1.0$; $ADE \Rightarrow B \text{ conf}=1.0$; $BDE \Rightarrow A, \text{ conf}=1.0$;
 $AD \Rightarrow BE \text{ conf}=1.0$; $DE \Rightarrow AB \text{ conf}=1.0$

Rule Assessment Measures

- If minsup is set too high, we could miss itemsets involving interesting rare items (e.g., expensive products)
- If minsup is set too low, it is computationally expensive and the number of itemsets is very large
- A single minimum support threshold may not be effective

- Lift is the ratio of the observed joint probability of X and Y to the expected joint probability if they were statistically independent,

$$\text{lift}(X \Rightarrow Y) = \text{sup}(X \cup Y)/\text{sup}(X)\text{sup}(Y) = \text{conf}(X \Rightarrow Y)/\text{sup}(Y)$$

- Lift is a measure of the deviation from stochastic independence (if it is 1 then X and Y are independent)
- Lift also measures the surprise of the rule. A lift close to 1 means that the support of a rule is expected considering the supports of its components.
- We typically look for values of lift that are much larger (i.e., above expectation) or much smaller (i.e., below expectation) than one.

- Leverage is computed as the difference between the observed and expected joint probability of XY assuming that X and Y are independent

$$\text{leverage}(X \Rightarrow Y) = \text{sup}(X \cup Y) - \text{sup}(X)\text{sup}(Y)$$

- Leverage gives an “absolute” measure of how surprising a rule is and it should be used together with lift. Like lift it is symmetric.

Summarizing Itemsets

- A frequent itemset X is called maximal if it has no frequent supersets
- The set of all maximal frequent itemsets, given as

$$M = \{X \mid X \in F \text{ and } \nexists Y \supset X, \text{ such that } Y \in F\}$$

- M is a condensed representation of the set of all frequent itemset F , because we can determine whether any itemset is frequent or not using M
- If there is a maximal itemset Z such that $X \subseteq Z$, then X must be frequent, otherwise X cannot be frequent
- However, M alone cannot be used to determine $\text{sup}(X)$, we can only use to have a lower-bound, that is, $\text{sup}(X) \geq \text{sup}(Z)$ if $X \subseteq Z \in M$.

- An itemset X is closed if all supersets of X have strictly less support, that is,
$$\text{sup}(X) > \text{sup}(Y), \text{ for all } Y \supset X$$
- The set of all closed frequent itemsets C is a condensed representation, as we can determine whether an itemset X is frequent, as well as the exact support of X using C alone

- A frequent itemset X is a minimal generator if it has no subsets with the same support

$$G = \{ X \mid X \in F \text{ and } \nexists Y \subset X, \text{ such that } \text{sup}(X) = \text{sup}(Y) \}$$

- Thus, all subsets of X have strictly higher support, that is,

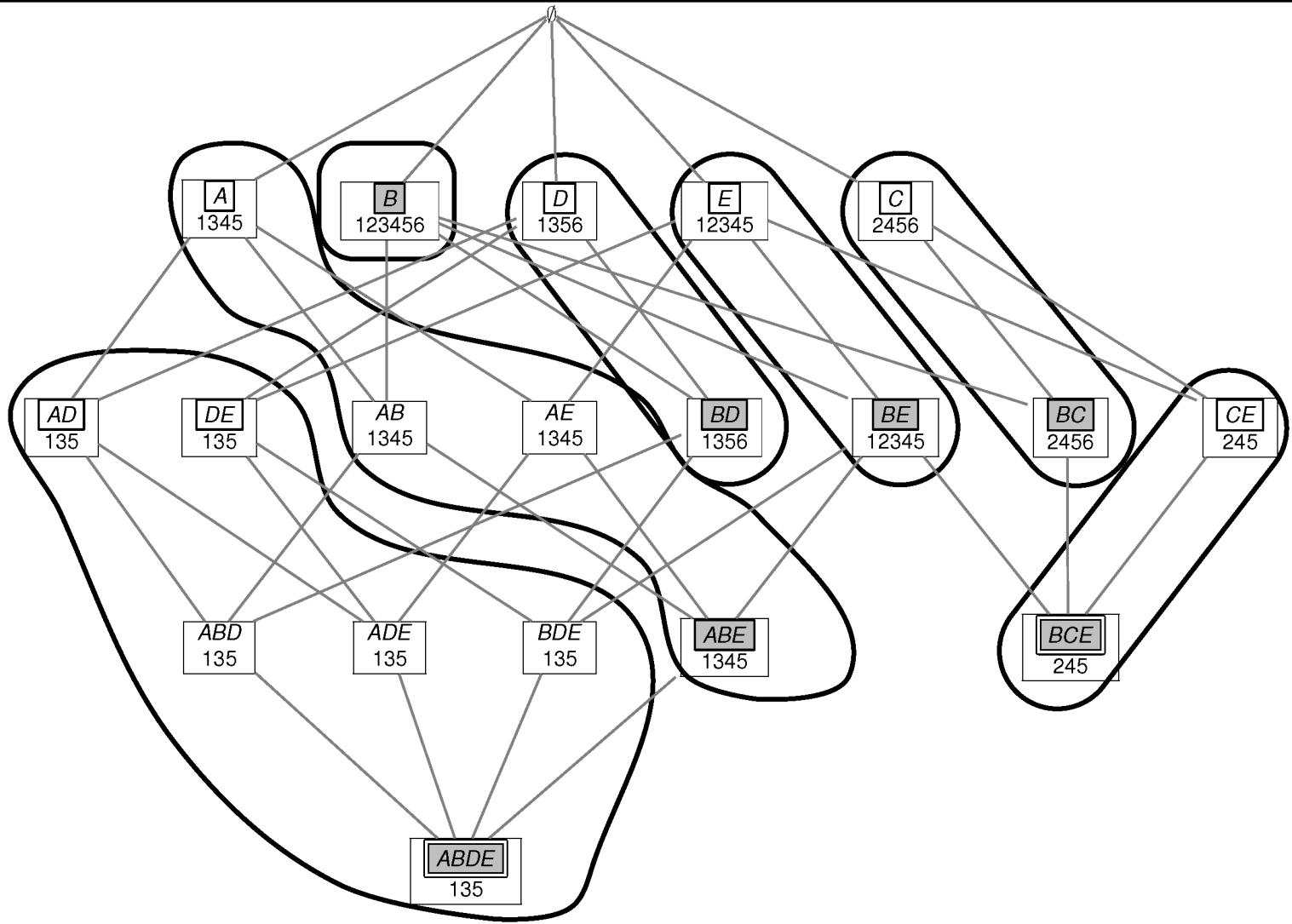
$$\text{sup}(X) < \text{sup}(Y)$$

Transaction database

Tid	Itemset
1	<i>ABDE</i>
2	<i>BCE</i>
3	<i>ABDE</i>
4	<i>ABCE</i>
5	<i>ABCDE</i>
6	<i>BCD</i>

Frequent itemsets ($\text{minsup} = 3$)

<i>sup</i>	Itemsets
6	<i>B</i>
5	<i>E, BE</i>
4	<i>A, C, D, AB, AE, BC, BD, ABE</i>
3	<i>AD, CE, DE, ABD, ADE, BCE, BDE, ABDE</i>



Shaded boxes are closed itemsets

Simple boxes are generators

Shaded boxes with double lines are maximal itemsets

Mining Frequent Sequences

- Association rules do not consider the order of transactions
- In many applications such orderings are significant
- In market basket analysis, it is interesting to know whether people buy some items in sequence
- Example
 - Buying bed first and then bed sheets later
 - Navigational patterns of users in a Web site from sequences of page visits of users

- A sequence is an ordered list of symbols $s = s_1, s_2, \dots, s_k$ where s_i or $s[i]$ denote the symbol at position i
- Notation $s[i:j]$ denotes $s_{i+1}, s_{i+2}, \dots, s_j$
- Let $s = s_1, s_2, \dots, s_n$ and $r = r_1, r_2, \dots, r_m$ we say that r is a subsequence of s denoted $s \subseteq r$ if there is a function ϕ from $[1,m]$ to $[1,n]$ such that $r[i] = s[\phi(i)]$ and for any position i, j in r ,

$$i < j \Rightarrow \phi(i) < \phi(j)$$

- r is a consecutive subsequence when

$$r_1, r_2, \dots, r_m = s_j, s_{j+1}, \dots, s_{j+m-1}$$

- Given $s = \text{ACTGAAACG}$
- $r_1 = \text{CGAAG}$ is a subsequence of s
- $r_2 = \text{CTGA}$ is a consecutive subsequence of s
- $r_3 = \text{ACT}$ is a prefix of s
- $r_4 = \text{GAACG}$ is a suffix of s

- Given a database D containing N sequences, the support of a sequence r in D is defined as the total number of sequences in D that contains r

$$sup(r) = |\{s_i \in D | r \subseteq s_i\}|$$

- The relative support of r is the percentage of sequences that contain r ,

$$rsup(r) = sup(r)/N$$

- Given a minsup threshold we that r is frequent if $sup(r) \geq \text{minsup}$
- Note that, in sequence mining, we need to consider all possible permutations of the items not just the combinations

Example

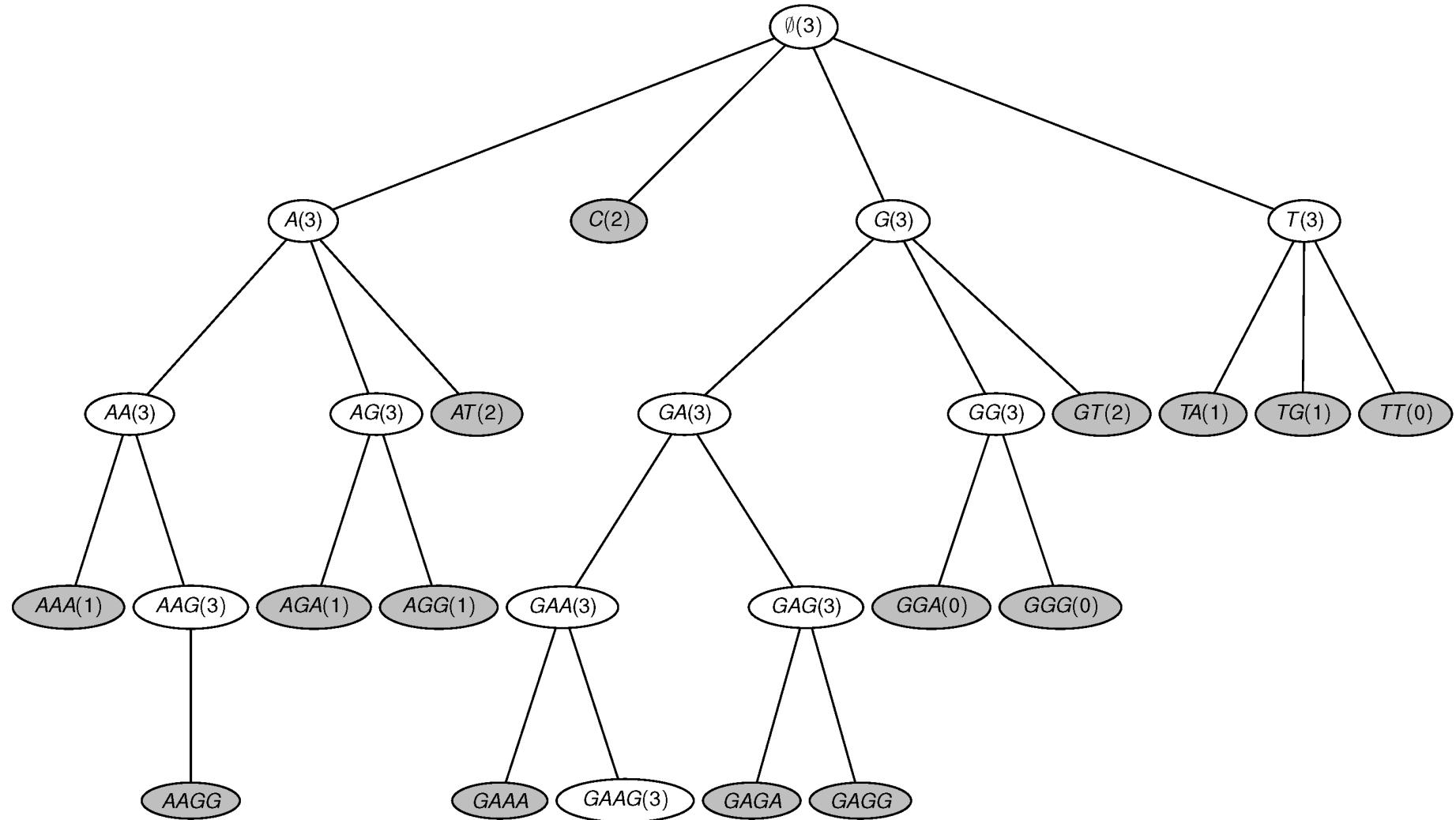
- Given the following sequence database

Id	Sequence
s_1	CAGAAGT
s_2	TGACAG
s_3	GAAGT



- With a minsup of 3, the set of frequent subsequences is
 - A(3), G(3), T(3)
 - AA(3), AG(3), GA(3), GG(3)
 - AAG(3), GAA(3), GAG(3)
 - GAAG(3)

- Searches the sequence prefix tree using a level-wise (breath-first search).
- Given the set of frequent sequences at level k , the algorithm generates the candidate for level $k+1$ and compute the support of each candidate and prune the not frequent ones.
- For each sequence s_i in D , we check if a candidate r is a subsequence of s_i , if it is the support of r is incremented. Once the frequent sequences at level k are computed the $k+1$ candidates are generated.
- For each leaf r_a , the sequence is extended with the last symbol of any other leaf r_b that shares the same prefix (it has the same parent) so $r_{ab} = r_a + r_b[k]$. If r_{ab} is infrequent, we prune it.



GSP (\mathbf{D} , Σ , $minsup$):

```
1  $\mathcal{F} \leftarrow \emptyset$ 
2  $\mathcal{C}^{(1)} \leftarrow \{\emptyset\}$  // Initial prefix tree with single symbols
3 foreach  $s \in \Sigma$  do Add  $s$  as child of  $\emptyset$  in  $\mathcal{C}^{(1)}$  with  $sup(s) \leftarrow 0$ 
4  $k \leftarrow 1$  //  $k$  denotes the level
5 while  $\mathcal{C}^{(k)} \neq \emptyset$  do
6   COMPUTESUPPORT ( $\mathcal{C}^{(k)}$ ,  $\mathbf{D}$ )
7   foreach leaf  $s \in \mathcal{C}^{(k)}$  do
8     if  $sup(r) \geq minsup$  then  $\mathcal{F} \leftarrow \mathcal{F} \cup \{(r, sup(r))\}$ 
9     else remove  $s$  from  $\mathcal{C}^{(k)}$ 
10   $\mathcal{C}^{(k+1)} \leftarrow \text{EXTENDPREFIXTREE } (\mathcal{C}^{(k)})$ 
11   $k \leftarrow k + 1$ 
12 return  $\mathcal{F}^{(k)}$ 
```

COMPUTESUPPORT ($\mathcal{C}^{(k)}$, \mathbf{D}):

```
1 foreach  $s_i \in \mathbf{D}$  do
2   foreach  $r \in \mathcal{C}^{(k)}$  do
3     if  $r \subseteq s_i$  then  $sup(r) \leftarrow sup(r) + 1$ 
```

EXTENDPREFIXTREE ($\mathcal{C}^{(k)}$):

```
1 foreach leaf  $r_a \in \mathcal{C}^{(k)}$  do
2   foreach leaf  $r_b \in \text{CHILDREN}(\text{PARENT}(r_a))$  do
3      $r_{ab} \leftarrow r_a + r_b[k]$  // extend  $r_a$  with last item of  $r_b$ 
      // prune if there are any infrequent
      subsequences
4     if  $r_c \in \mathcal{C}^{(k)}$ , for all  $r_c \subset r_{ab}$ , such that  $|r_c| = |r_{ab}| - 1$  then
5       Add  $r_{ab}$  as child of  $r_a$  with  $sup(r_{ab}) \leftarrow 0$ 
6     if no extensions from  $r_a$  then
7       remove  $r_a$ , and all ancestors of  $r_a$  with no extensions, from  $\mathcal{C}^{(k)}$ 
8 return  $\mathcal{C}^{(k)}$ 
```

Association Rules for Classification

- Association rule mining assumes that the data consist of a set of transactions. Thus, the typical tabular representation of data used in classification must be mapped into such a format.
- Association rule mining is then applied to the new dataset and the search is focused on association rules in which the tail identifies a class label

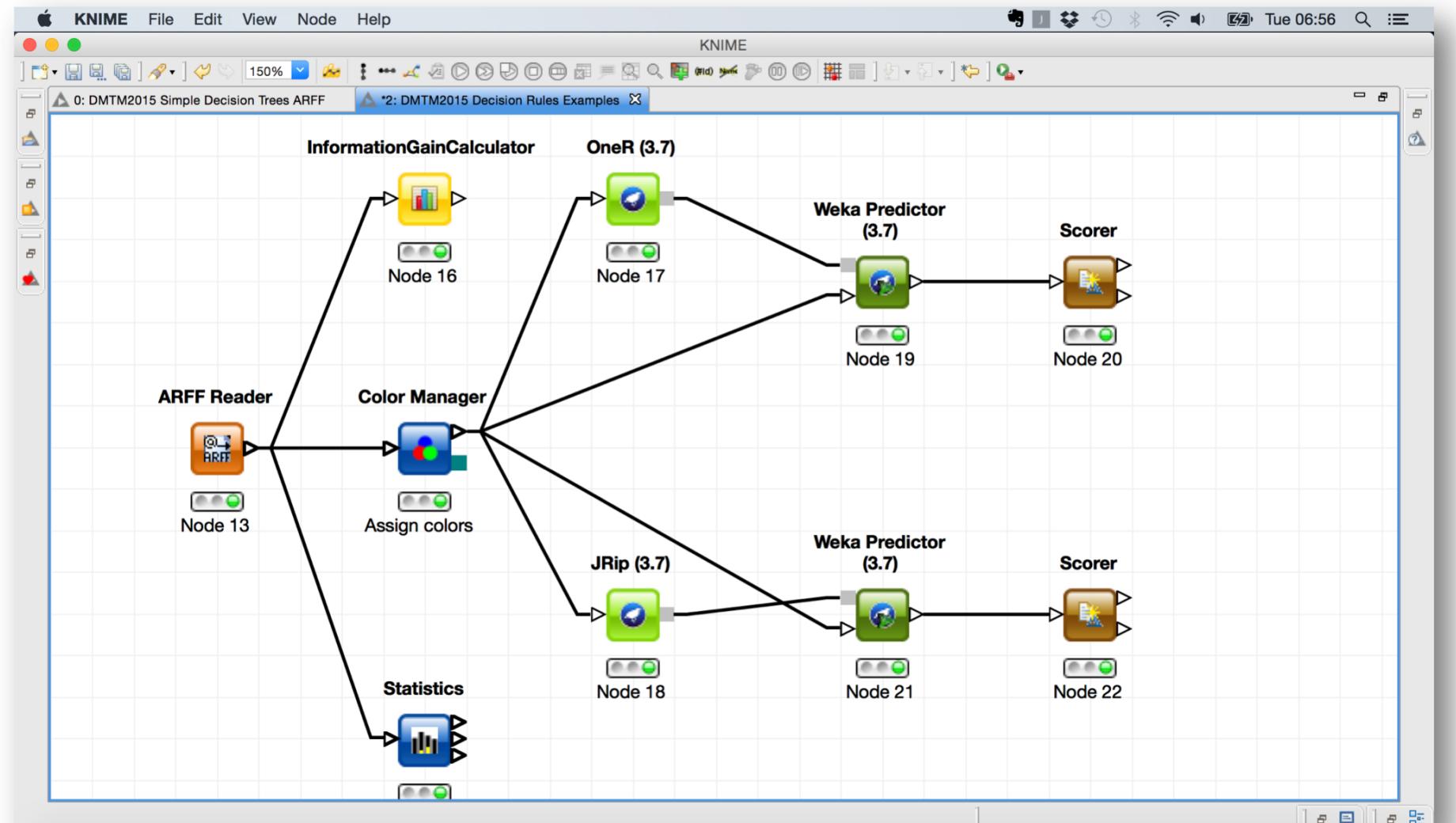
$$X \rightarrow c_i \text{ (where } c_i \text{ is a class label)}$$

- The association rules are pruned using the pessimistic error-based method used in C4.5
- Finally, rules are sorted to build the final classifier.

The Weather Dataset

67

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



(humidity = high) and (outlook = sunny) =>
play=no (3.0/0.0)

(outlook = rainy) and (windy = TRUE) =>
play=no (2.0/0.0)

=> play=yes (9.0/0.0)

outlook:

overcast -> yes

rainy-> yes

sunny-> no

(10/14 instances correct)

outlook=overcast => play=yes
humidity=normal windy=FALSE => play=yes
outlook=rainy windy=FALSE => play=yes
outlook=sunny humidity=high => play=no
outlook=rainy windy=TRUE => play=no

(default class is the majority class)

Run the notebooks for this lecture

Download and run SPMF library

<http://www.philippe-fournier-viger.com/spmf/index.php?link=download.php>