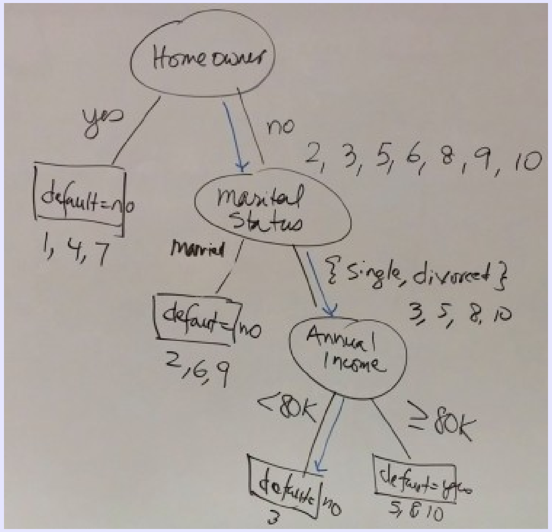
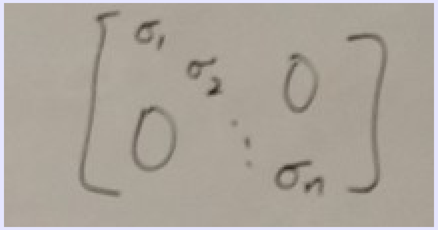
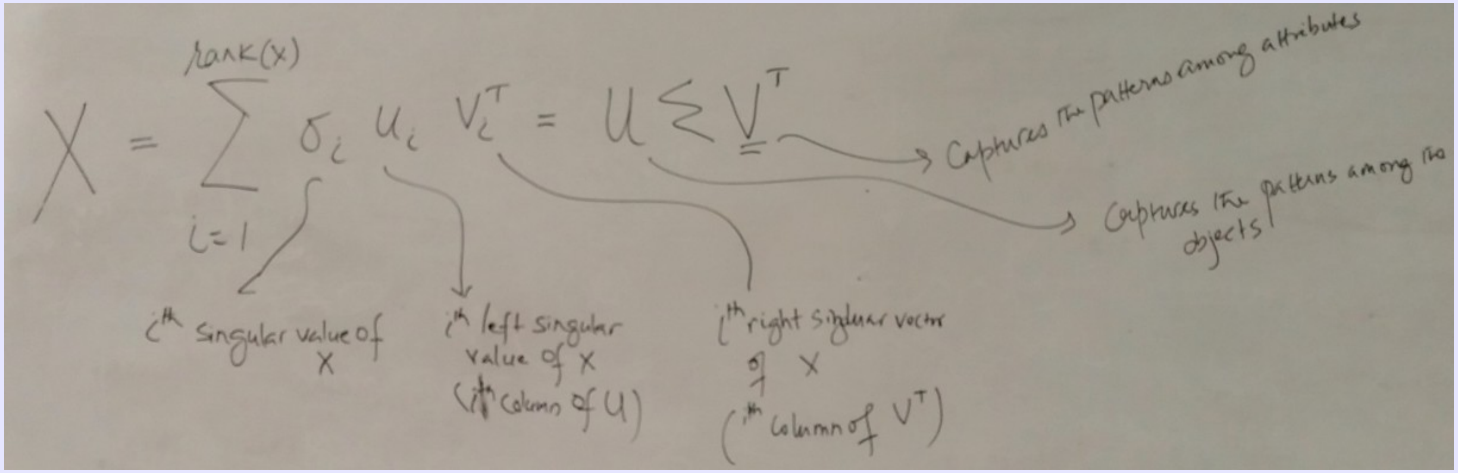
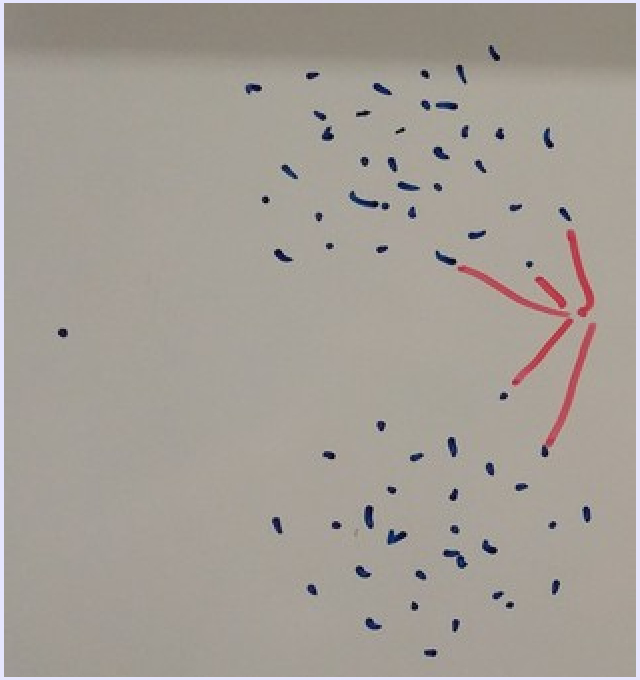








CS 422: Data Mining

Vijay K. Gurbani, Ph.D., Illinois Institute of Technology

Association Analysis (Rules)

Association Rule Mining

- One of the early examples of data mining.
- Interested in observing which objects occur together:
  - Grocery shopping (*market-basket analysis*)
  - Website visits
- Notice that we are not ***recommending*** similar items, just seeing which items co-occur.
  - Recommendation is for a later lecture.

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. (Note: Implication means co-occurrence, not causality!)

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}

Antecedent → Consequent

Association Rule Mining

Binary representation of market basket data

Image source: https://goo.gl/images/mQ3zZz

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

	Beer	Bread	Milk	Diaper	Eggs	Coke
T_1	0	1	1	0	0	0
T_2	1	1	0	1	1	0
T_3	1	0	1	1	0	1
T_4	1	1	1	1	0	0
T_5	0	1	1	1	0	1

Association Rule Mining

- Preliminaries

Let $\mathcal{I} = \{x_1, x_2, ..., x_m\}$ be a set of elements called *items*.
A set $X \subseteq \mathcal{I}$ is called an *itemset*.
An itemset of cardinality k is called a k-itemset.
$\mathcal{I}^{(k)}$ is the set of all k-itemsets.
Let $\mathcal{T} = \{t_1, t_2, ..., t_n\}$ be another set of elements called transaction identifiers, or *tids*.
A set $T \subseteq \mathcal{T}$ is called an *tidset*.
A *transaction* is a tuple of the form (t, X) where t $\in$ T is a unique transaction identifier, and X is an itemset.

Association Rule Mining

- Preliminaries

Let $\mathcal{I} = \{x_1, x_2, ..., x_m\}$ be a set of elements called *items*.
A set $X \subseteq \mathcal{I}$ is called an *itemset*.
An itemset of cardinality k is called a k-itemset.
$\mathcal{I}^{(k)}$ is the set of all k-itemsets.
Let $\mathcal{T} = \{t_1, t_2, ..., t_n\}$ be another set of elements called transaction identifiers, or *tids*.
A set $T \subseteq \mathcal{T}$ is called an *tidset*.
A *transaction* is a tuple of the form (t, X) where t $\in$ T is a unique transaction identifier, and X is an itemset.

Database Representation

A binary database $\mathbf{D}$ is a binary relation on the set of tids and items, that is, $\mathbf{D} \subseteq \mathcal{T} \times \mathcal{I}$. We say that tid $t \in \mathcal{T}$ *contains* item $x \in \mathcal{I}$ iff $(t, x) \in \mathbf{D}$. In other words, $(t, x) \in \mathbf{D}$ iff $x \in X$ in the tuple $\langle t, X \rangle$. We say that tid t *contains* itemset $X = \{x_1, x_2, \ldots, x_k\}$ iff $(t, x_i) \in \mathbf{D}$ for all $i = 1, 2, \ldots, k$.

For a set X, we denote by 2^X the powerset of X, that is, the set of all subsets of X. Let $\mathbf{i} : 2^{\mathcal{T}} \to 2^{\mathcal{I}}$ be a function, defined as follows:

$$\mathbf{i}(T) = \{x \mid \forall t \in T,\ t \text{ contains } x\} \tag{8.1}$$

where $T \subseteq \mathcal{T}$, and $\mathbf{i}(T)$ is the set of items that are common to *all* the transactions in the tidset T. In particular, $\mathbf{i}(t)$ is the set of items contained in tid $t \in \mathcal{T}$.

Association Rule Mining

- Preliminaries

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

(a) Binary database

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

(b) Transaction database

x	A	B	C	D	E
$\mathbf{t}(x)$	1	1	2	1	1
	3	2	4	3	2
	4	3	5	5	3
	5	4	6	6	4
		5			5
		6			

(c) Vertical database

Figure 8.1. An example database.

Association Rule Mining

- Preliminaries

Support and Frequent Itemsets

The *support* of an itemset X in a dataset $\mathbf{D}$, denoted $sup(X, \mathbf{D})$, is the number of transactions in $\mathbf{D}$ that contain X:

$$sup(X, \mathbf{D}) = \left|\{t \mid \langle t, \mathbf{i}(t)\rangle \in \mathbf{D} \text{ and } X \subseteq \mathbf{i}(t)\}\right| = |\mathbf{t}(X)|$$

The *relative support* of X is the fraction of transactions that contain X:

$$rsup(X, \mathbf{D}) = \frac{sup(X, \mathbf{D})}{|\mathbf{D}|}$$

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

(a) Binary database

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

(b) Transaction database

sup({A,B}) = 4 rsup({A,B}) = 4/6 = 0.67
sup({B}) = 6 rsup({B}) = 6/6 = 1.00

Association Rule Mining

- Preliminaries

Support and Frequent Itemsets

The *support* of an itemset X in a dataset $\mathbf{D}$, denoted $sup(X, \mathbf{D})$, is the number of transactions in $\mathbf{D}$ that contain X:

$$sup(X, \mathbf{D}) = \left|\{t \mid \langle t, \mathbf{i}(t)\rangle \in \mathbf{D} \text{ and } X \subseteq \mathbf{i}(t)\}\right| = |\mathbf{t}(X)|$$

The *relative support* of X is the fraction of transactions that contain X:

$$rsup(X, \mathbf{D}) = \frac{sup(X, \mathbf{D})}{|\mathbf{D}|}$$

$\mathbf{D}$	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

(a) Binary database

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

(b) Transaction database

sup({A,B}) = 4 rsup({A,B}) = 4/6 = 0.67
sup({B}) = 6 rsup({B}) = 6/6 = 1.00

We use $\mathcal{F}$ to denote the set of all itemsets, and $\mathcal{F}^{(k)}$ to denote the set of k-itemsets.

Thus, in our transaction database shown above,
$\mathcal{F}^{(3)} = \{\text{BCE, BCD}\}$
$\mathcal{F}^{(4)} = \{\text{ABDE}\}$
$\mathcal{F}^{(5)} = \{\text{ABCDE}\}$

Association Rule Mining

- Preliminaries

Frequent itemsets: An itemset X is frequent if sup(X) ≥ *minsup*, where *minsup* is a user specified minimum support threshold. (If *minsup* is a fraction, then relative support is implied.)

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

(a) Binary database

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

(b) Transaction database

Example: Let *minsup* = 3 (in relative support term, *minsup* = 0.5). The set of all 19 frequent itemsets grouped by their support value is:

Table 8.1. Frequent itemsets with $minsup = 3$

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

Association Rule Mining

- Preliminaries

Association Rules

An *association rule* is an expression $X \xrightarrow{s,c} Y$, where X and Y are itemsets and they are disjoint, that is, $X, Y \subseteq \mathcal{I}$, and $X \cap Y = \emptyset$. Let the itemset $X \cup Y$ be denoted as XY. The *support* of the rule is the number of transactions in which both X and Y co-occur as subsets:

$$s = sup(X \longrightarrow Y) = |\mathbf{t}(XY)| = sup(XY)$$

The *relative support* of the rule is defined as the fraction of transactions where X and Y co-occur, and it provides an estimate of the joint probability of X and Y:

$$rsup(X \longrightarrow Y) = \frac{sup(XY)}{|\mathbf{D}|} = P(X \wedge Y)$$

The *confidence* of a rule is the conditional probability that a transaction contains Y given that it contains X:

$$c = conf(X \longrightarrow Y) = P(Y|X) = \frac{P(X \wedge Y)}{P(X)} = \frac{sup(XY)}{sup(X)}$$

A rule is *frequent* if the itemset XY is frequent, that is, $sup(XY) \geq minsup$ and a rule is *strong* if $conf \geq minconf$, where $minconf$ is a user-specified minimum confidence threshold.

Association Rule Mining

- Preliminaries

Table 8.1. Frequent itemsets with $minsup = 3$

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

Association Rules

An *association rule* is an expression $X \xrightarrow{s,c} Y$, where X and Y are itemsets and they are disjoint, that is, $X, Y \subseteq \mathcal{I}$, and $X \cap Y = \emptyset$. Let the itemset $X \cup Y$ be denoted as XY. The *support* of the rule is the number of transactions in which both X and Y co-occur as subsets:

$$s = sup(X \longrightarrow Y) = |\mathbf{t}(XY)| = sup(XY)$$

The *relative support* of the rule is defined as the fraction of transactions where X and Y co-occur, and it provides an estimate of the joint probability of X and Y:

$$rsup(X \longrightarrow Y) = \frac{sup(XY)}{|\mathbf{D}|} = P(X \wedge Y)$$

The *confidence* of a rule is the conditional probability that a transaction contains Y given that it contains X:

$$c = conf(X \longrightarrow Y) = P(Y|X) = \frac{P(X \wedge Y)}{P(X)} = \frac{sup(XY)}{sup(X)}$$

A rule is *frequent* if the itemset XY is frequent, that is, $sup(XY) \geq minsup$ and a rule is *strong* if $conf \geq minconf$, where $minconf$ is a user-specified minimum confidence threshold.

Example: BC → E.

sup(BC → E) = sup(BCE) = 3

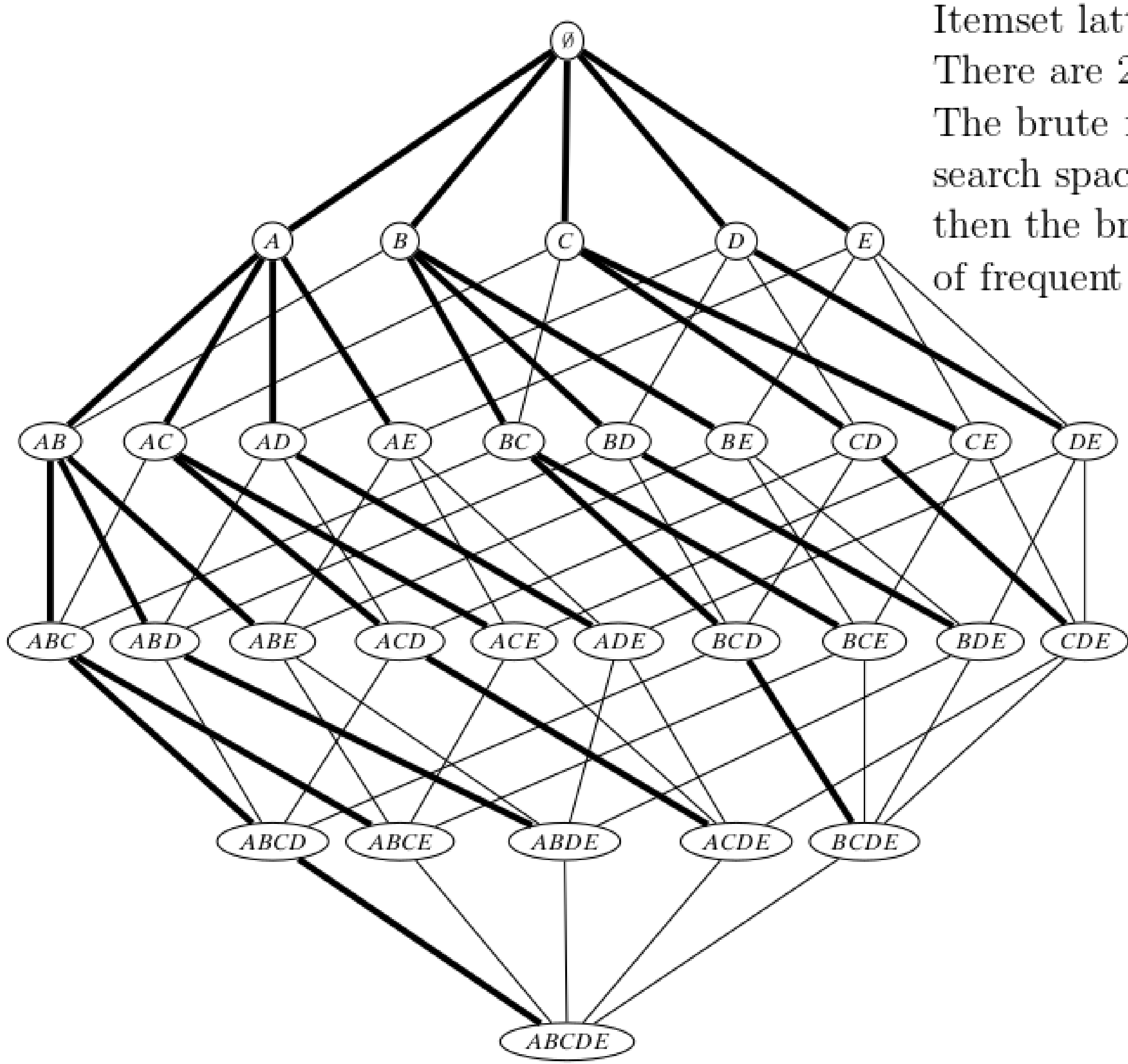
conf(BC → E) = sup(BCE) / sup(BC)

= ¾ = 0.75

Association Rule Mining

- Goal of Association Rule Mining: Given a set of transactions, T, find all rules having:
  - support >= *minsup*
  - confidence >= *minconf*
- How do we get there?
- Two steps:
  - Frequent itemset generation: find all items that satisfy *minsup* threshold (frequent itemsets). (Is computationally expensive!!)
  - Rule generation: extract all high-confidence rules from the frequent itemsets (strong rules).

Frequent Itemset Generation: Brute Force Method

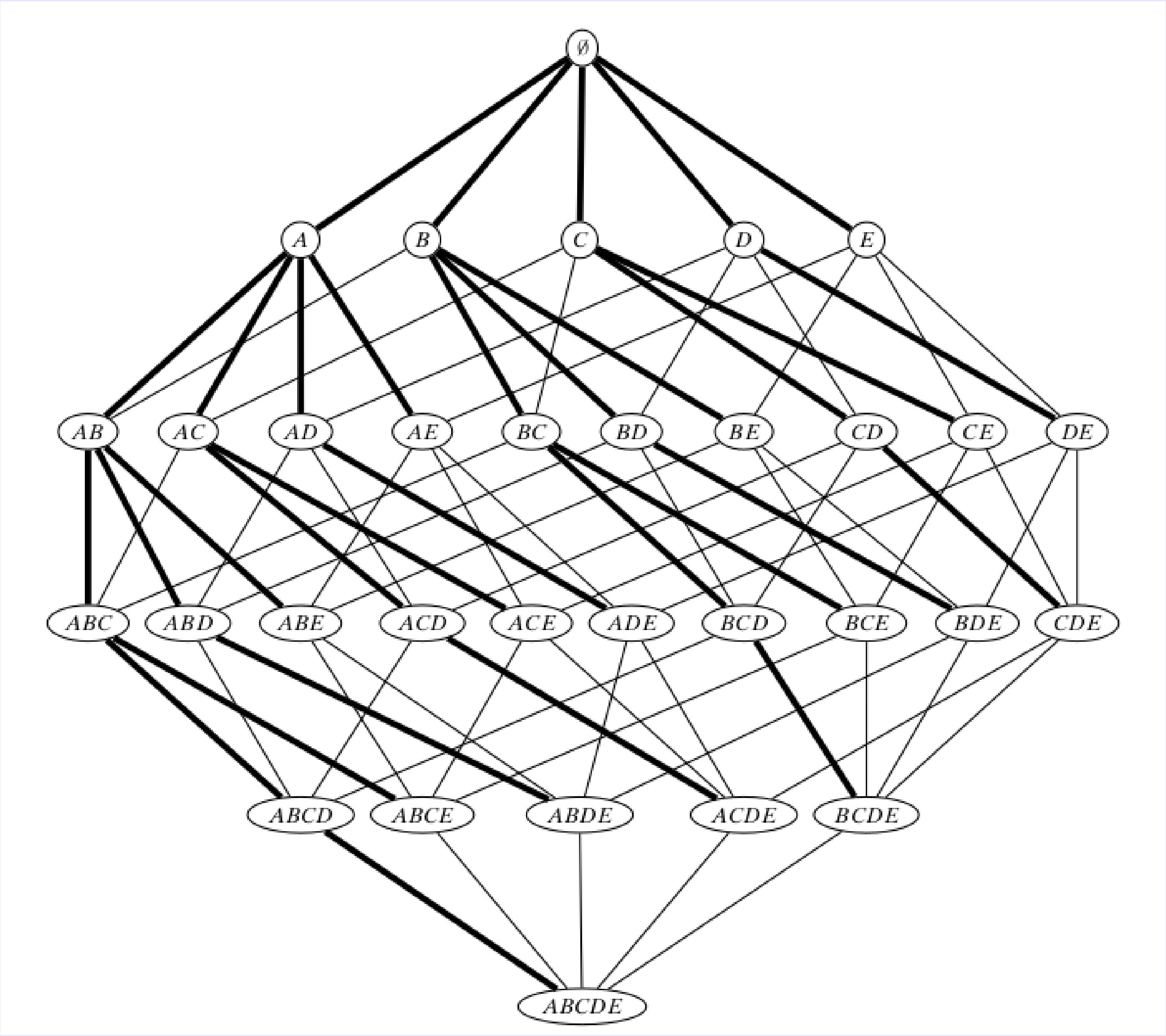


Itemset lattice for $\mathcal{I} = \{A, B, C, D, E\}$.
There are $2^{|\mathcal{I}|} = 32$ possible itemsets.
The brute force method explores the entire itemset search space, regardless of $minsup$. If $minsup = 3$, then the brute-force search method would output the set of frequent itemsets shown below.

Table 8.1. Frequent itemsets with $minsup = 3$

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

Frequent Itemset Generation: Brute Force Method



ALGORITHM 8.1. Algorithm BRUTEFORCE

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

$\mathcal{F} \leftarrow \emptyset$ `// set of frequent itemsets`

foreach $X \subseteq \mathcal{I}$ **do**

  $sup(X) \leftarrow$ COMPUTESUPPORT $(X, \mathbf{D})$

  if $sup(X) \geq minsup$ **then**

    $\mathcal{F} \leftarrow \mathcal{F} \cup \{(X, sup(X))\}$

return $\mathcal{F}$

COMPUTESUPPORT $(X, \mathbf{D})$:

$sup(X) \leftarrow 0$

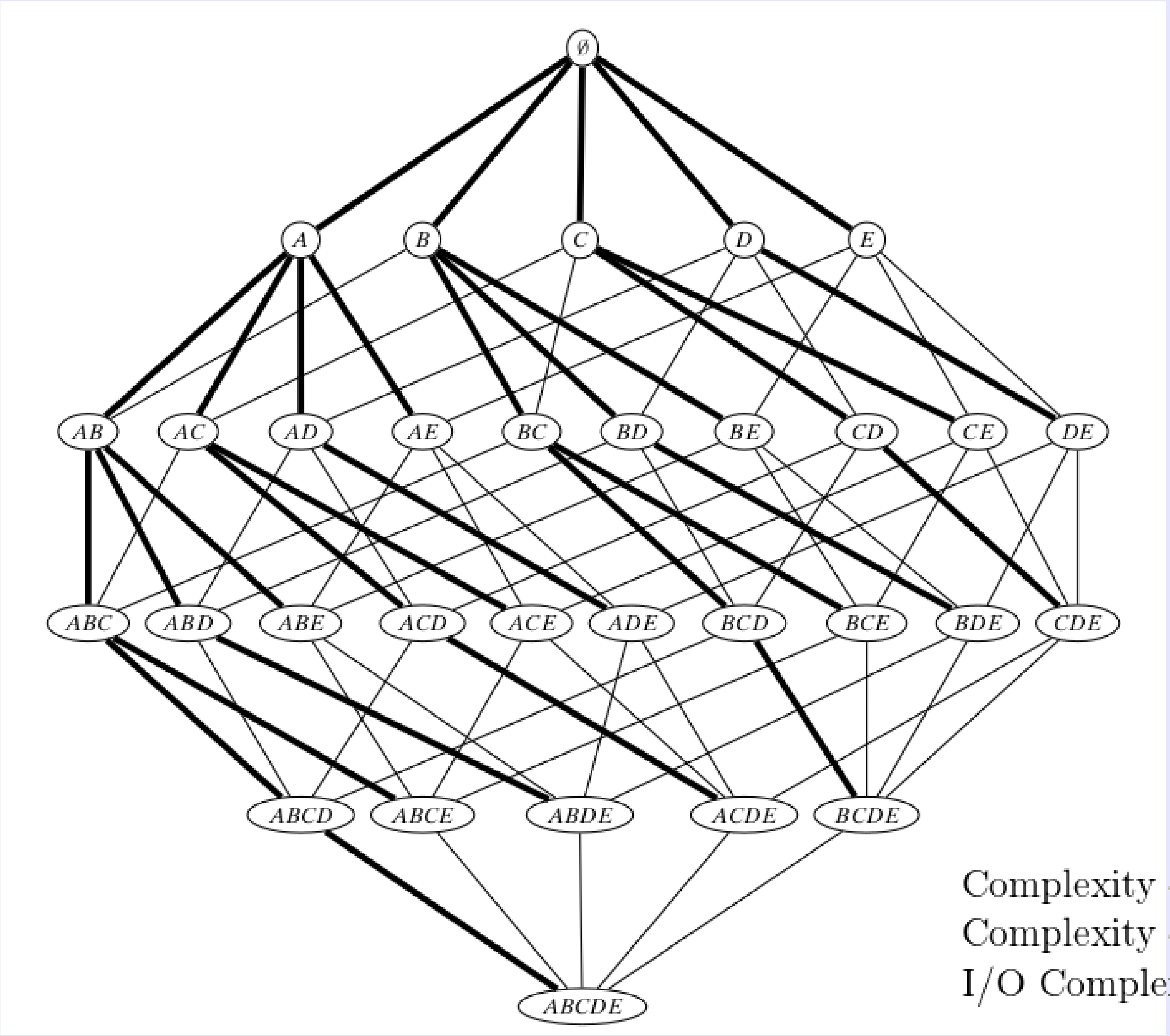
foreach $\langle t, \mathbf{i}(t) \rangle \in \mathbf{D}$ **do**

  if $X \subseteq \mathbf{i}(t)$ **then**

    $sup(X) \leftarrow sup(X) + 1$

return $sup(X)$

Frequent Itemset Generation: Brute Force Method



ALGORITHM 8.1. Algorithm BRUTEFORCE

```
BRUTEFORCE (D, I, minsup):
F ← ∅ // set of frequent itemsets
foreach X ⊆ I do
    sup(X) ← COMPUTESUPPORT (X, D)
    if sup(X) ≥ minsup then
        F ← F ∪ {(X, sup(X))}
return F

COMPUTESUPPORT (X, D):
sup(X) ← 0
foreach ⟨t, i(t)⟩ ∈ D do
    if X ⊆ i(t) then
        sup(X) ← sup(X) + 1
return sup(X)
```

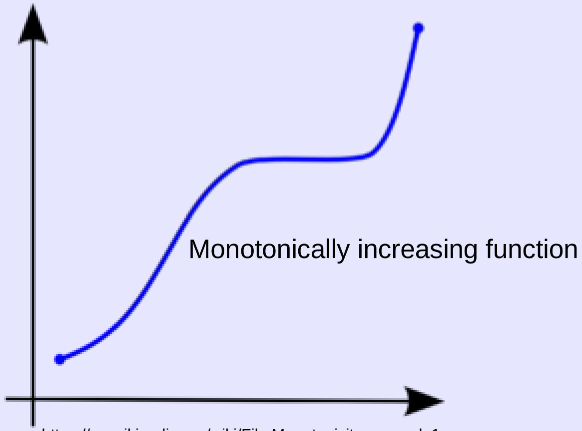
Complexity of ComputeSupport: $\mathcal{O}(|\mathcal{I}| * D)$

Complexity of BruteForce: $\mathcal{O}(|\mathcal{I}| * D * 2^{|\mathcal{I}|})$

I/O Complexity of BruteForce: $\mathcal{O}(2^{|\mathcal{I}|})$ database scans

Frequent Itemset Generation: The Apriori Approach

- Can we do better?
- Yes; thanks to the monotone property.



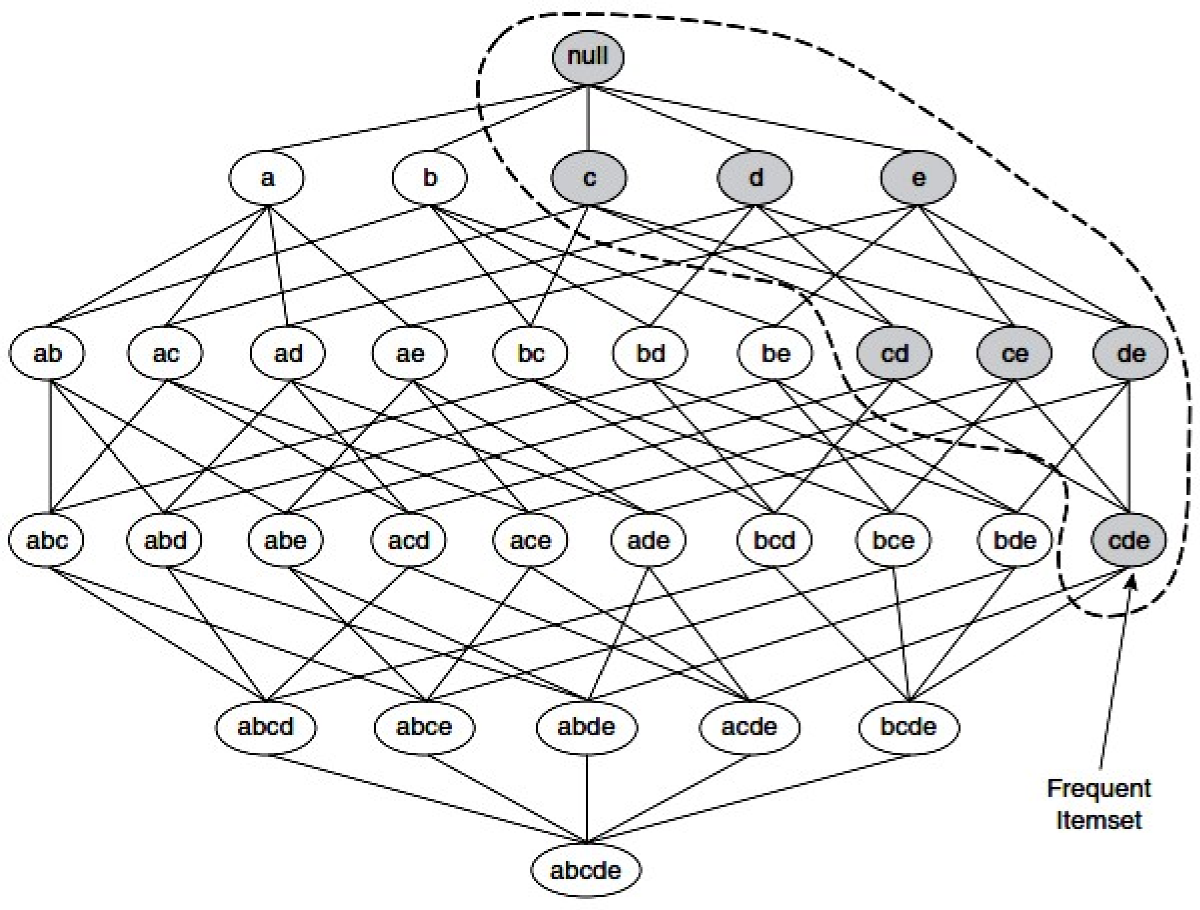
Source: https://en.wikipedia.org/wiki/File:Monotonicity_example1.png

Let X and Y be two itemsets $\in \mathcal{I}$ such that $X \subseteq Y$.
If so, then $\sup(X) \geq \sup(Y)$

E.g. X=ABCD, Y=ABCDE, then sup(ABCD) >= sup(ABCDE).

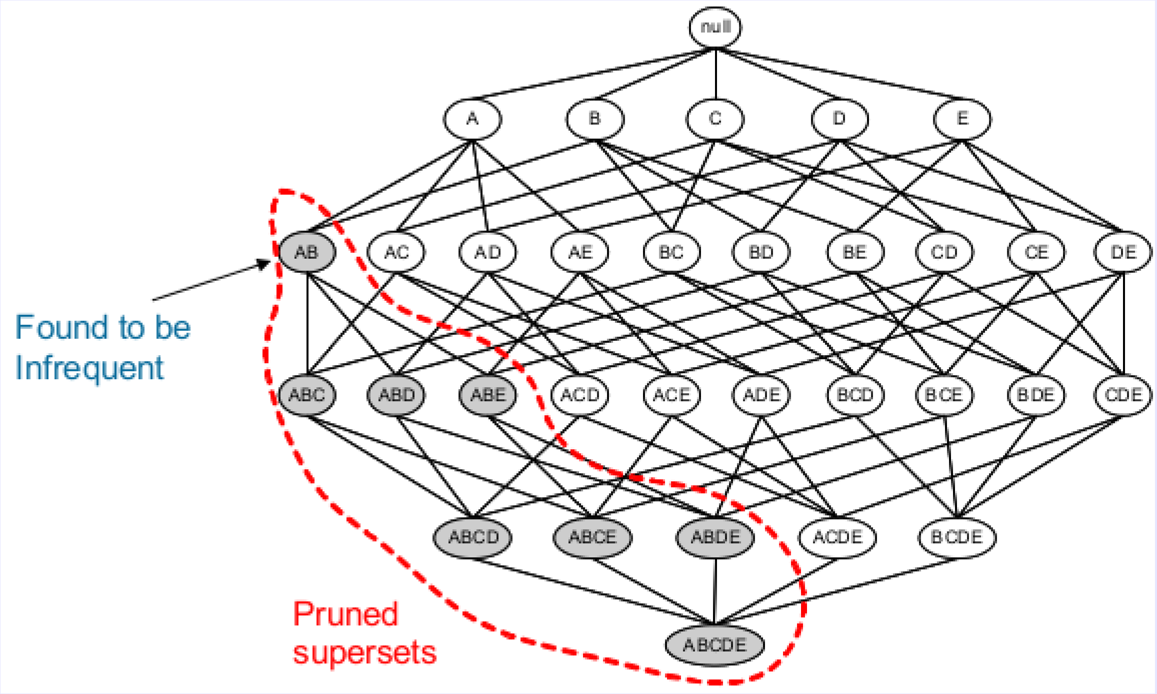
Frequent Itemset Generation: The Apriori Approach

- The *Apriori* principle:
  - If an itemset is frequent, then all of its subsets must be frequent as well.



Frequent Itemset Generation: The Apriori Approach

- The *Apriori* principle:
  - Conversely, if an itemset is infrequent, then all of its supersets must be infrequent as well.



Frequent Itemset Generation: The Apriori Approach

minsup = 3

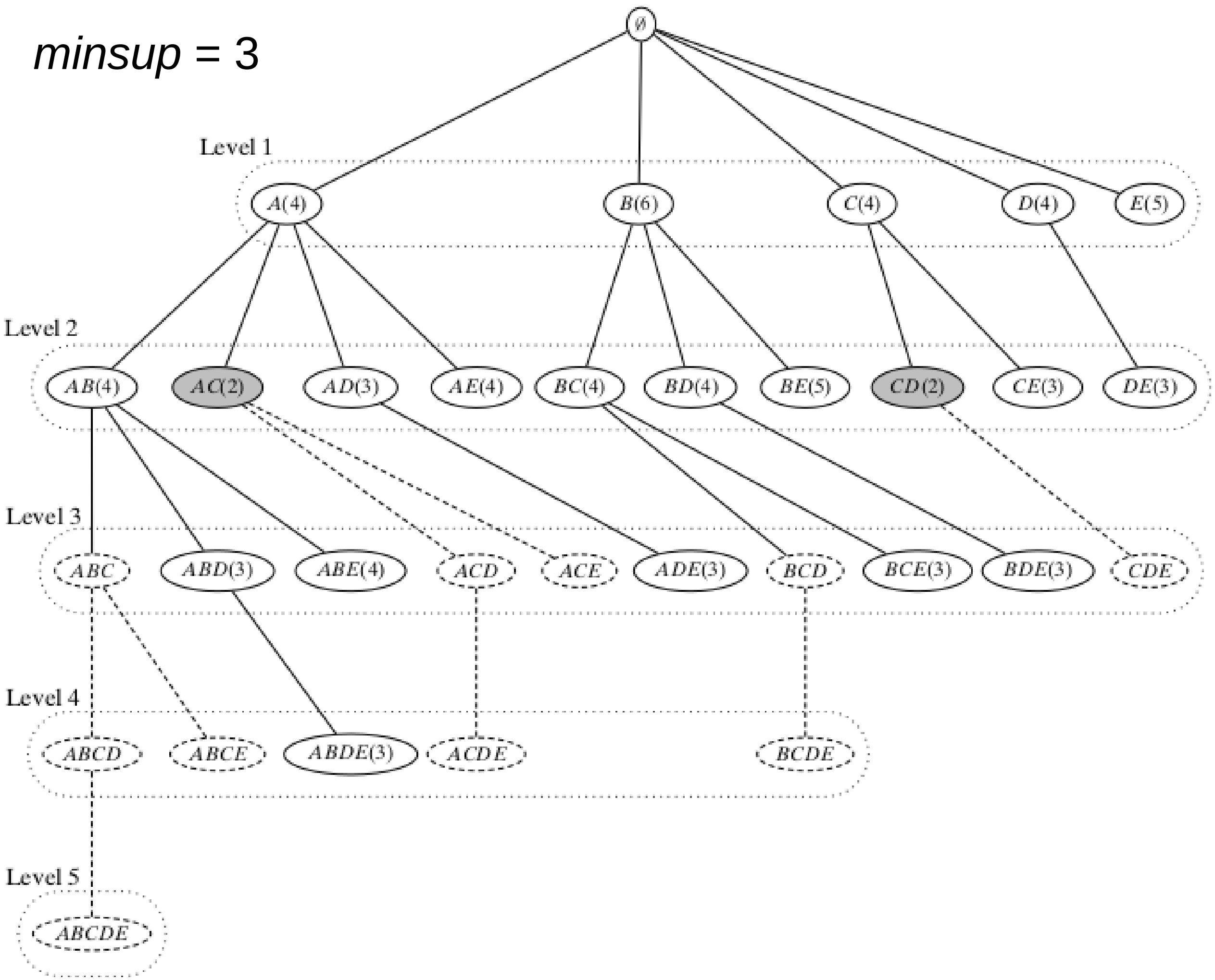


Figure 8.3. Apriori: prefix search tree and effect of pruning. Shaded nodes indicate infrequent itemsets, whereas dashed nodes and lines indicate all of the pruned nodes and branches. Solid lines indicate frequent itemsets.

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

(a) Binary database

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

(b) Transaction database

Frequent Itemset Generation: The Apriori Approach

ALGORITHM 8.2. Algorithm APRIORI

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

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

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

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

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

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

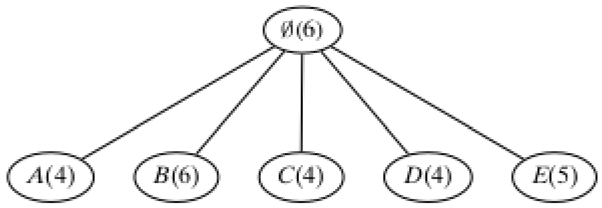
minsup = 3

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

(a) Binary database

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

(b) Transaction database



(a) $\mathcal{C}^{(1)}$

Frequent Itemset Generation: The Apriori Approach

ALGORITHM 8.2. Algorithm APRIORI

APRIORI ($\mathbf{D}, \mathcal{I}, minsup$):
- $\mathcal{F} \leftarrow \emptyset$
- $\mathcal{C}^{(1)} \leftarrow \{\emptyset\}$ // Initial prefix tree with single items
- **foreach** $i \in \mathcal{I}$ **do** Add i as child of $\emptyset$ in $\mathcal{C}^{(1)}$ with $sup(i) \leftarrow 0$
- $k \leftarrow 1$ // k denotes the level
- **while** $\mathcal{C}^{(k)} \neq \emptyset$ **do**
  - COMPUTESUPPORT ($\mathcal{C}^{(k)}, \mathbf{D}$)
  - **foreach** *leaf* $X \in \mathcal{C}^{(k)}$ **do**
    - **if** $sup(X) \geq minsup$ **then** $\mathcal{F} \leftarrow \mathcal{F} \cup \{(X, sup(X))\}$
    - **else** remove X from $\mathcal{C}^{(k)}$
  - $\mathcal{C}^{(k+1)} \leftarrow$ EXTENDPREFIXTREE ($\mathcal{C}^{(k)}$)
  - $k \leftarrow k+1$
- **return** $\mathcal{F}^{(k)}$

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

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

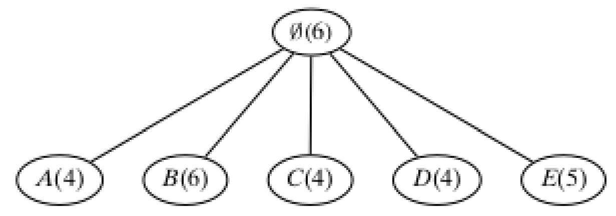
minsup = 3

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

(a) Binary database

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

(b) Transaction database



(a) $\mathcal{C}^{(1)}$

We now extend the prefix tree from Level k to Level k+1: given two frequent k-itemsets (X_a and X_b), with common k-1 length prefix (i.e., two siblings with common parent), we generate (k+1) length candidates
$X_{ab} = X_a \cup X_b$.
- X_{ab} retained only if it has no infrequent subset.

Frequent Itemset Generation: The Apriori Approach

ALGORITHM 8.2. Algorithm APRIORI

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

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

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

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

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

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

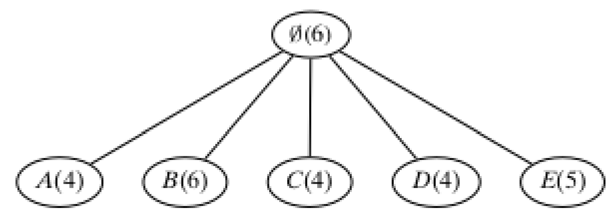
minsup = 3

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

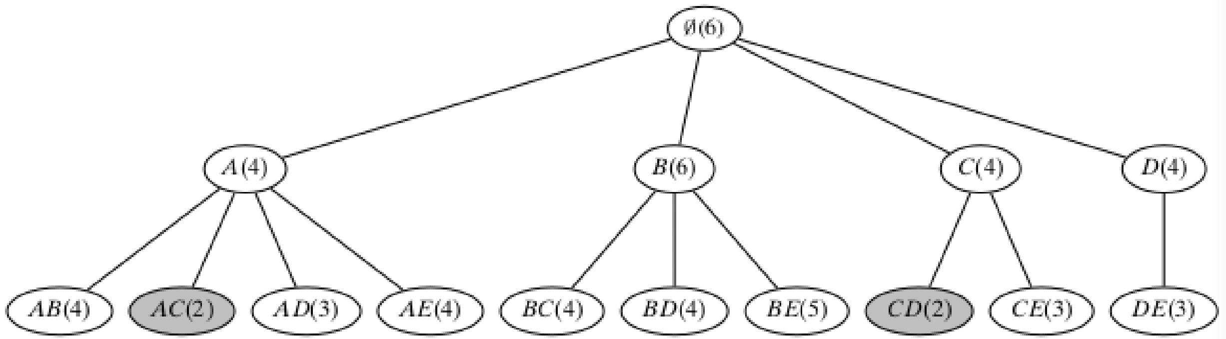
(a) Binary database

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

(b) Transaction database



(a) $\mathcal{C}^{(1)}$



(b) $\mathcal{C}^{(2)}$

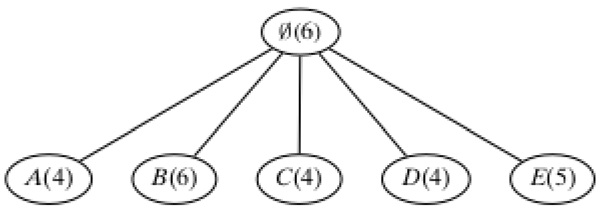
Frequent Itemset Generation: The Apriori Approach

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

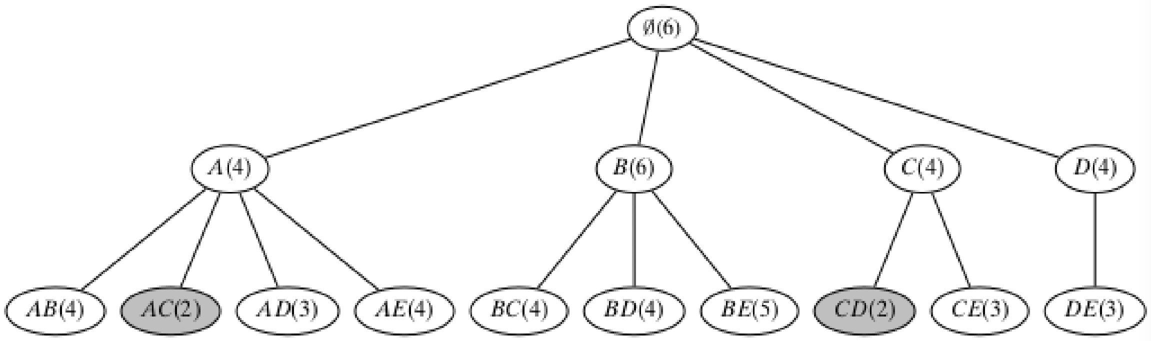
(a) Binary database

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

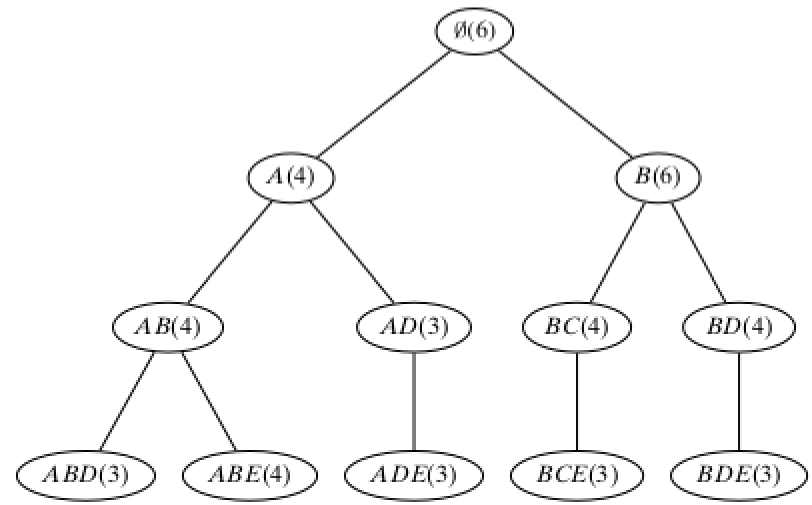
(b) Transaction database



(a) $\mathcal{C}^{(1)}$



(b) $\mathcal{C}^{(2)}$



(c) $\mathcal{C}^{(3)}$

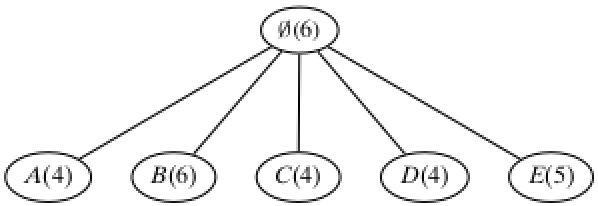
Frequent Itemset Generation: The Apriori Approach

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

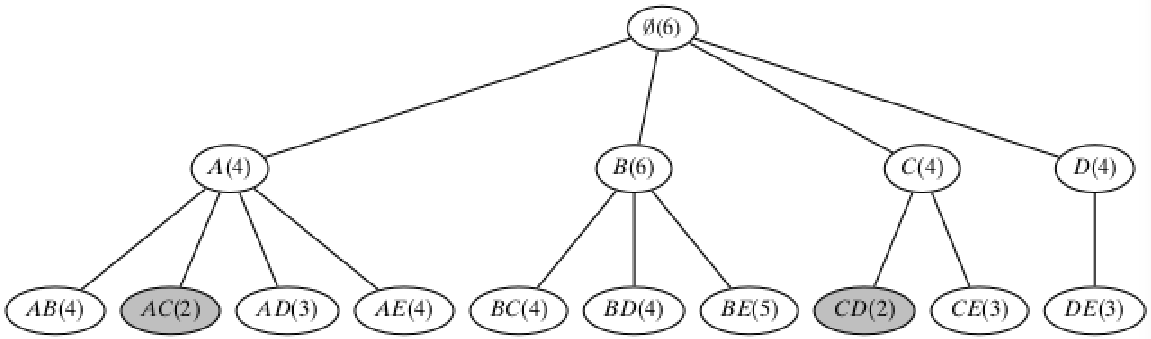
(a) Binary database

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

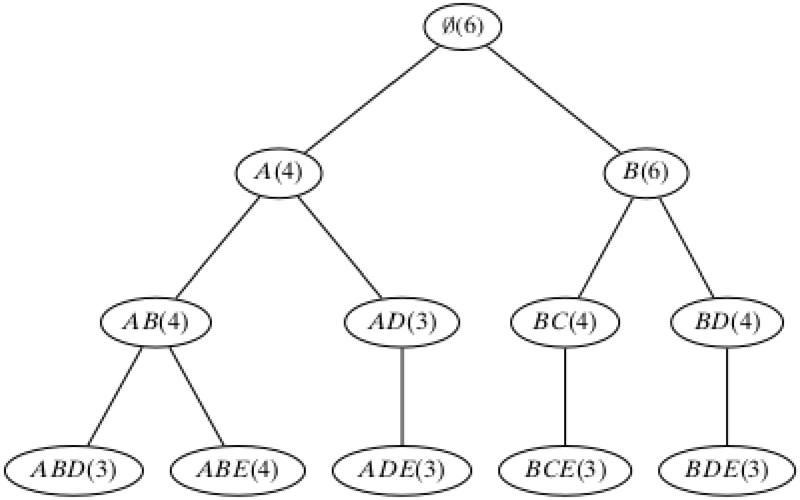
(b) Transaction database



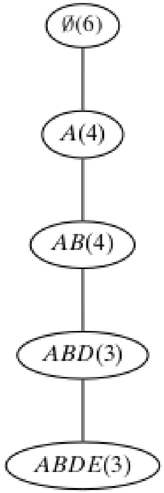
(a) $\mathcal{C}^{(1)}$



(b) $\mathcal{C}^{(2)}$



(c) $\mathcal{C}^{(3)}$



(d) $\mathcal{C}^{(4)}$

Frequent Itemset Generation: The Apriori Approach

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

(a) Binary database

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

(b) Transaction database

ALGORITHM 8.2. Algorithm APRIORI

```
APRIORI (D, I, minsup):
F ← ∅
C^(1) ← {∅} // Initial prefix tree with single items
foreach i ∈ I do Add i as child of ∅ in C^(1) with sup(i) ← 0
k ← 1 // k denotes the level
while C^(k) ≠ ∅ do
    COMPUTESUPPORT (C^(k), D)
    foreach leaf X ∈ C^(k) do
        if sup(X) ≥ minsup then F ← F ∪ {(X, sup(X))}
        else remove X from C^(k)
    C^(k+1) ← EXTENDPREFIXTREE (C^(k))
    k ← k + 1
return F^(k)

COMPUTESUPPORT (C^(k), D):
foreach ⟨t, i(t)⟩ ∈ D do
    foreach k-subset X ⊆ i(t) do
        if X ∈ C^(k) then sup(X) ← sup(X) + 1

EXTENDPREFIXTREE (C^(k)):
foreach leaf X_a ∈ C^(k) do
    foreach leaf X_b ∈ SIBLING(X_a), such that b > a do
        X_ab ← X_a ∪ X_b
        // prune candidate if there are any infrequent subsets
        if X_j ∈ C^(k), for all X_j ⊂ X_ab, such that |X_j| = |X_ab| − 1 then
            Add X_ab as child of X_a with sup(X_ab) ← 0
    if no extensions from X_a then
        remove X_a, and all ancestors of X_a with no extensions, from C^(k)
return C^(k)
```

Worst case complexity:

Complexity of Apriori: $\mathcal{O}(|\mathcal{I}| * D * 2^{|\mathcal{I}|})$ as all itemsets may be frequent. In practice, much lower due to pruning.
I/O costs are much lower, to the tune of $\mathcal{O}(|\mathcal{I}|)$ database scans as opposed to $\mathcal{O}(2^{|\mathcal{I}|})$ scans for brute-force. In practice, the algorithm only requires l database scans, where l is the length of the longest frequent itemset.

Frequent Itemset Generation: FP-Growth Algorithm

- A more optimized tree-based approach to discovering frequent itemsets.
  - Radically different than Apriori.
  - Maps each transaction to a path in a compact data structure called a FP-tree.
  - Extracts frequent itemsets directly from the tree
    - No candidate itemset generation and pruning
    - No multiple database lookups
  - The more paths overlap, the more the tree compresses, saving space.

Frequent Itemset Generation: FP-Growth Algorithm

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

(a) Binary database

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

(b) Transaction database

No slides yet; follow on board.

Frequent Itemset Generation

How many item-sets?

k items generate up to 2^k-*1* frequent itemsets.

$$\text{Number of itemsets for k items} = \binom{k}{1} + \binom{k}{2} + \ldots + \binom{k}{k} = 2^k - 1$$

For a 3-itemset {a,b,c} the candidate rules will be:
ab → c, ac → b, a → bc, b → ac, …, abc → 0 and 0 → abc

How many rules?

$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right] = 3^d - 2^{d+1} + 1$$

d = No. of items
For d = 3, R = 12
For d = 6, R = 602