



Frequent Itemset Mining



Nadjib LAZAAR

LIRMM- UM
COCONUT Team

(PART I)

IMAGINA 2020 / 2021

Webpage: github.com/FDSInfoMontp-HMIN233

Email: nadjib.lazaar@umontpellier.fr

Data Mining

- **Data Mining (DM)** or Knowledge Discovery in Databases (KDD) revolves around the investigation and creation of knowledge, processes, algorithms, and the mechanisms for **retrieving potential knowledge** from **data collections**.

Game Data Mining

- Data about players behavior, server performance, system functionality...
- How to convert these data into something meaningful?
- How to move from raw data to actionable insights?
- ➔ Game data mining is the answer

Frequent Itemset Mining: Motivations

Frequent Itemset Mining is a method for market basket analysis.

It aims at finding regularities in the shopping behavior of customers of supermarkets, mail-order companies, on-line shops etc.

- More specifically: Find sets of products that are frequently bought together.
- Possible applications of found frequent itemsets:
 - Improve arrangement of products in shelves, on a catalog's pages etc.
 - Support cross-selling (suggestion of other products), product bundling.
 - Fraud detection, technical dependence analysis, fault localization... etc.
- Often found patterns are expressed as association rules, for example:
 - If a customer buys bread and wine, then she/he will probably also buy cheese.

Frequent Itemset Mining: Basic notions

- Items: $I = \{i_1, \dots, i_n\}$
- Itemset, transaction: $P, T, \subseteq I$
- Transactional dataset: $D = \{T_1, \dots, T_m\}$
- Language of itemsets: $\mathcal{L}_I = 2^I$
- Cover of an itemset: $cover(P) = \{i \mid T_i \in D \wedge P \subseteq T_i\}$
- (absolute) Frequency: $freq(P) = |cover(P)|$

Absolute/relative frequency

➤ Absolute Frequency:

$$freq(P) = |cover(P)|$$

➤ Relative Frequency:

$$freq(P) = \frac{1}{|D|} |cover(P)|$$

Frequent Itemset Mining: Definition

➤ Given:

- A set of items $I = \{i_1, \dots, i_n\}$
- A transactional dataset $D = \{T_1, \dots, T_m\}$
- A minimum support θ

➤ The need:

- The set of itemset P s.t.: $freq(P) \geq \theta$

Example (1)

$$I = \{a, b, c, d, e\}, D = \{T_1, \dots, T_{10}\}$$

 \mathcal{H}_D

1:	a d e
2:	b c d
3:	a c e
4:	a c d e
5:	a e
6:	a c d
7:	b c
8:	a c d e
9:	b c e
10:	a d e

 \mathcal{V}_D

	a	b	c	d	e
1		2	2	1	1
3		7	3	2	3
4		9	4	4	4
5			6	6	5
6			7	8	8
8			8	10	9
10			9		10

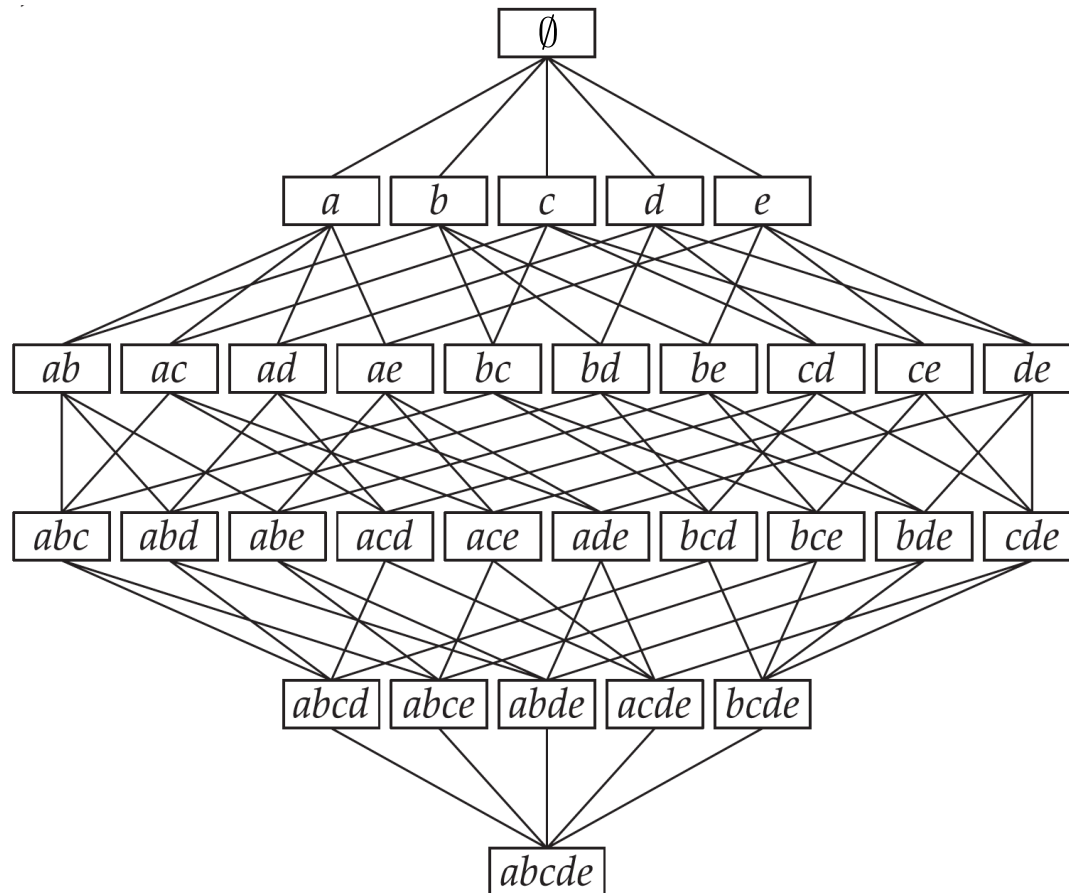
 \mathcal{M}_D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

$$cover(bc) = \{2, 7, 9\}$$

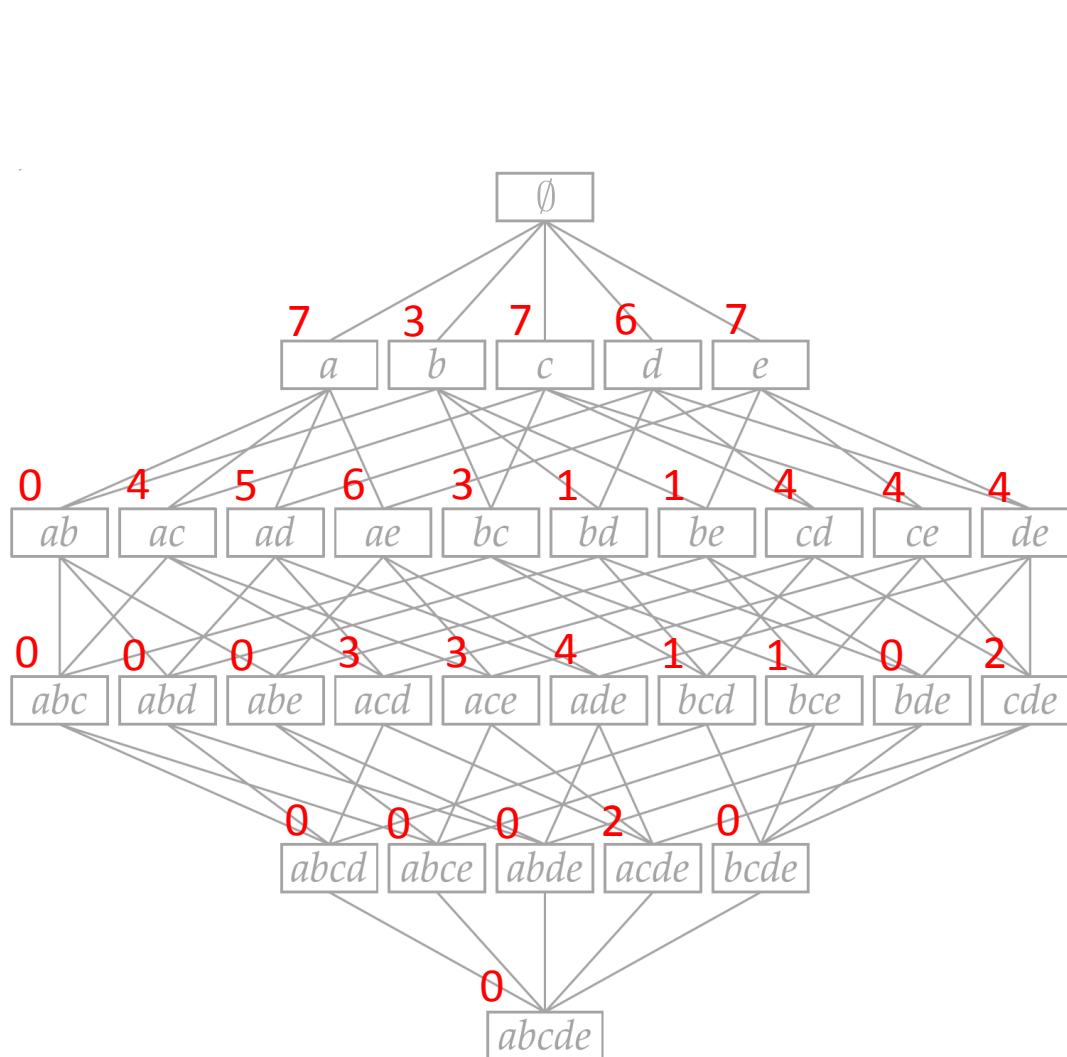
$$freq(bc) = 3$$

Example (1)


 \mathcal{M}_D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

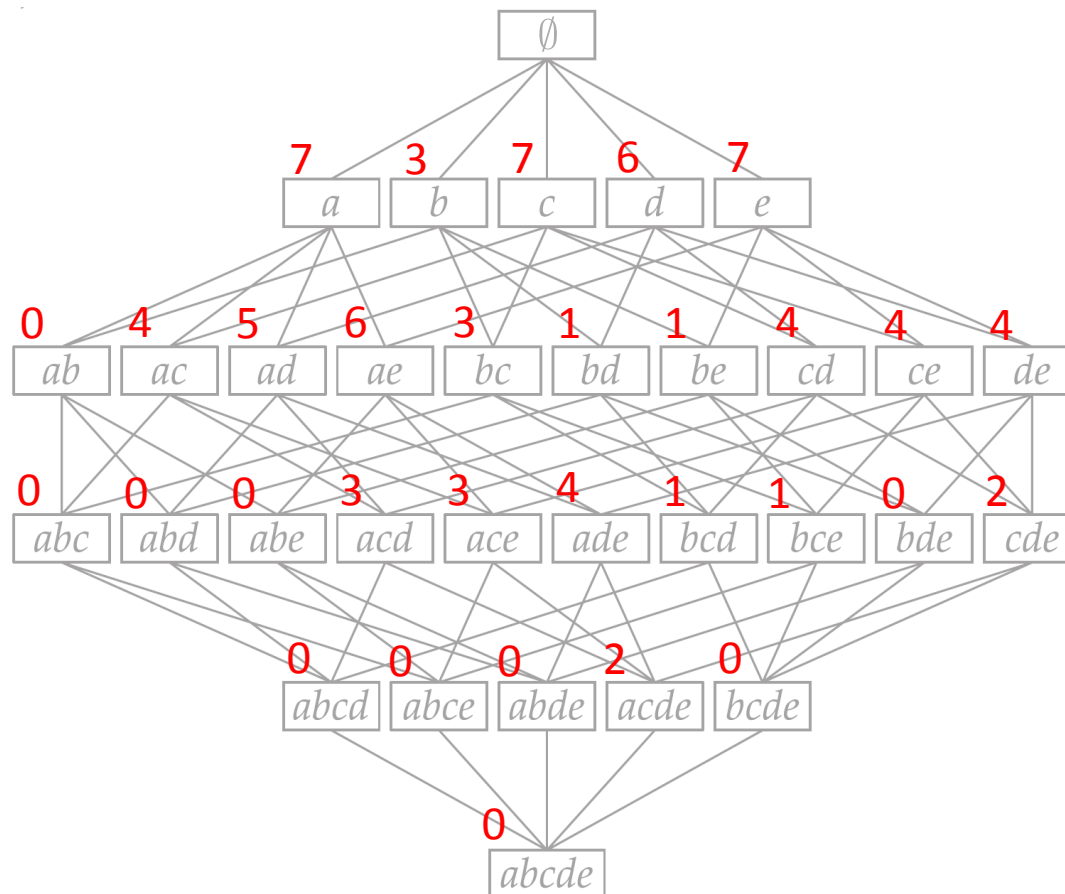
Example (1)


 \mathcal{M}_D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Example (1)

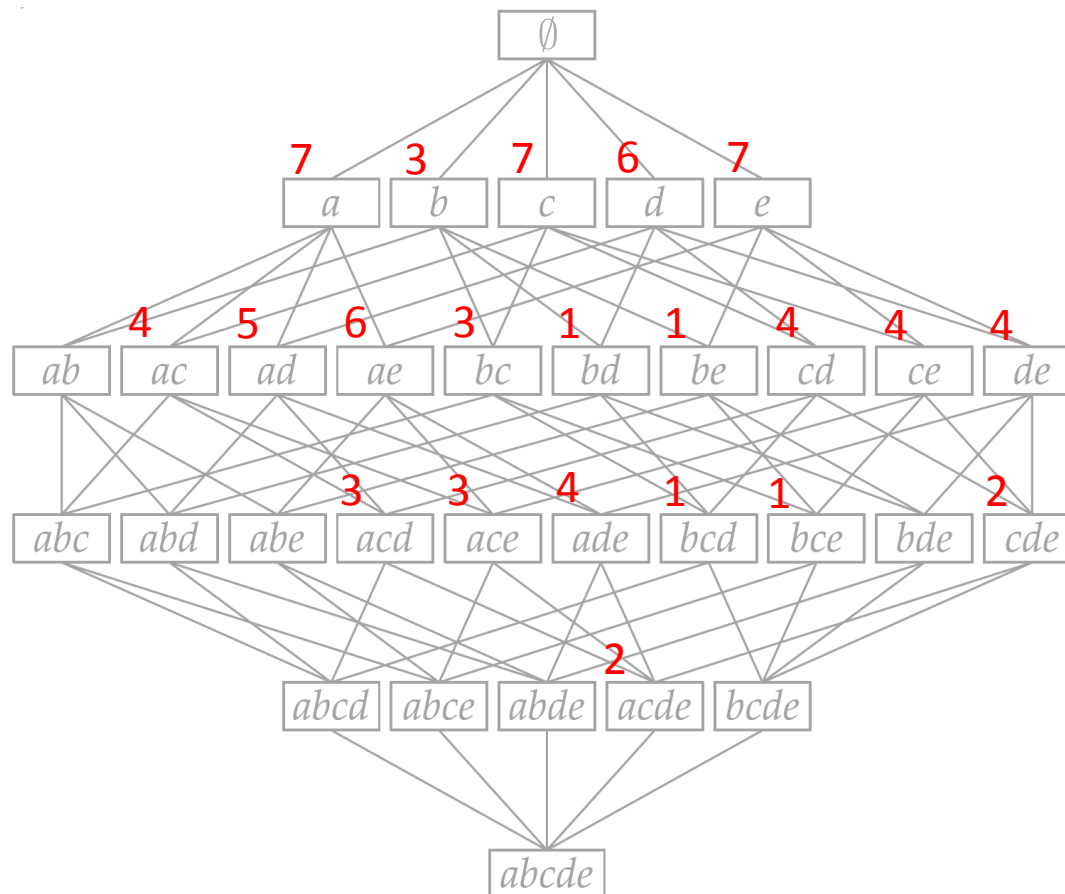
Frequent itemset?


 \mathcal{M}_D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Example (1)

Frequent itemset with minimum support $\theta=3$?


 \mathcal{M}_D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Searching for Frequent Itemsets

➤ A **naïve search** that consists of enumerating and testing the frequency of itemset candidates in a given dataset is usually **infeasible**.

➤ Why?

Number of items (n)	Search space (2^n)
10	$\approx 10^3$
20	$\approx 10^6$
30	$\approx 10^9$
100	$\approx 10^{30}$
128	$\approx 10^{68}$ (atoms in the universe)
1000	$\approx 10^{301}$

Anti-monotonicity property

- Given a transaction database D over items I and two itemsets X , Y :

$$X \subseteq Y \Rightarrow \text{cover}(Y) \subseteq \text{cover}(X)$$

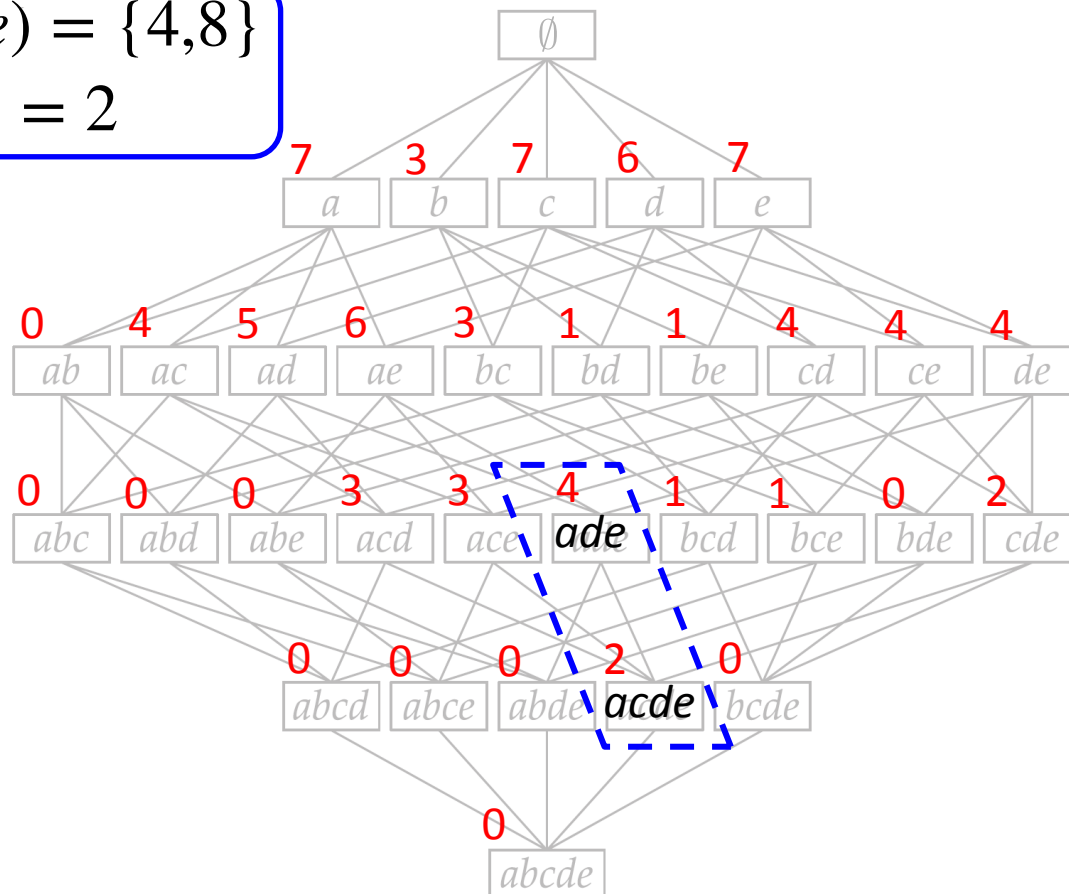
- That is,

$$X \subseteq Y \Rightarrow \text{freq}(Y) \leq \text{freq}(X)$$

Example (2)

$cover(ade) = \{1,4,8,10\}, freq(ade) = 4$

$cover(acde) = \{4,8\}$
 $freq(acde) = 2$



	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

matrix representation

Apriori property

- Given a transaction database D over items I , a minsup θ and two itemsets X, Y :

$$X \subseteq Y \Rightarrow \text{freq}(Y) \leq \text{freq}(X)$$

- It follows: $X \subseteq Y \Rightarrow (\text{freq}(Y) \geq \theta \Rightarrow \text{freq}(X) \geq \theta)$

All subsets of a frequent itemset are frequent!

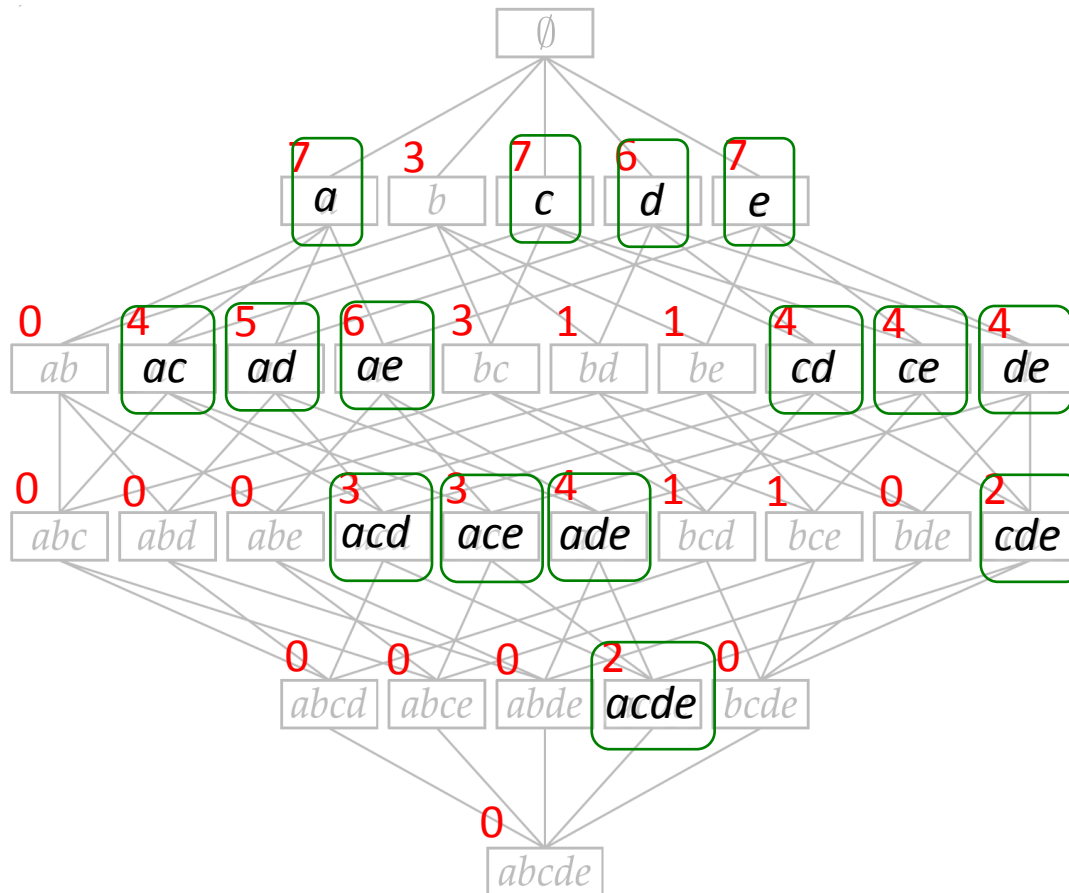
- Contraposition: $X \subseteq Y \Rightarrow (\text{freq}(X) < \theta \Rightarrow \text{freq}(Y) < \theta)$

All supersets of an infrequent itemset are infrequent!

Example (3)

All subsets of a frequent itemset are frequent!

$\theta = 2$



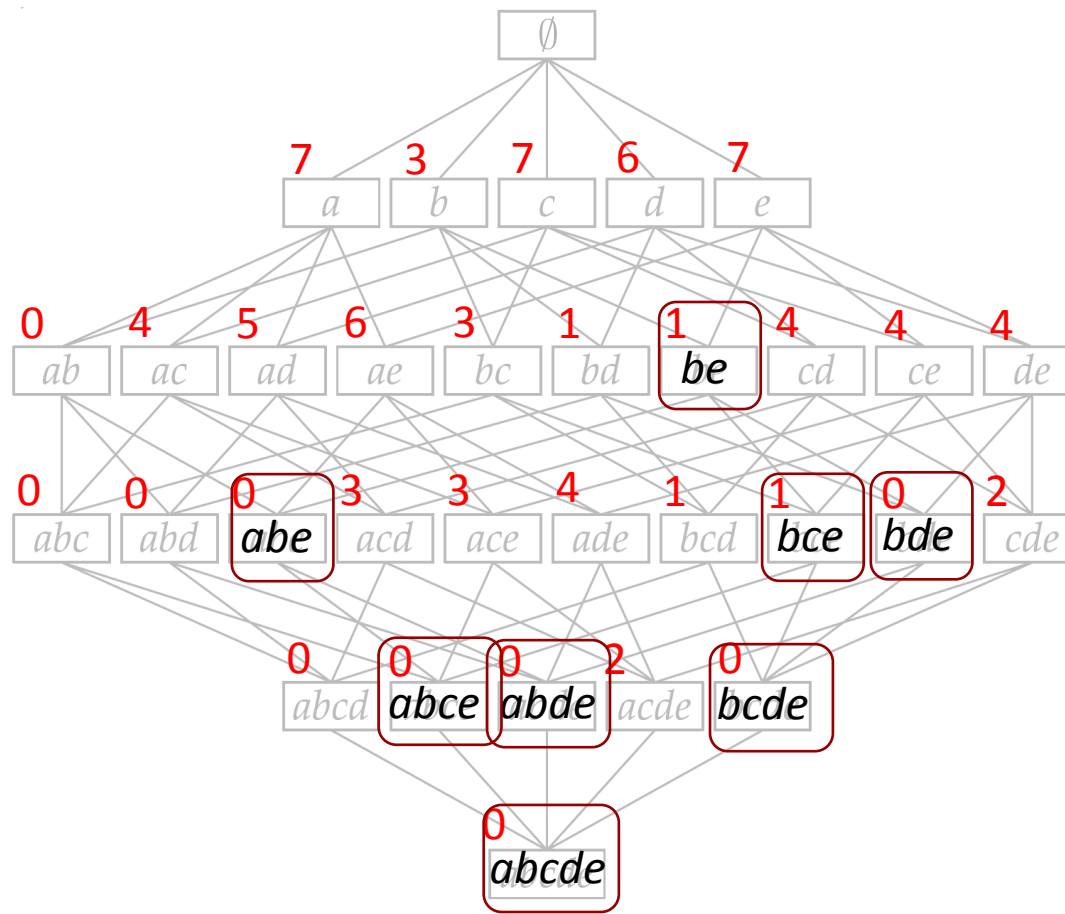
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

matrix representation

Example (3)

All supersets of an infrequent itemset are infrequent!

$$\theta = 2$$



	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

matrix representation

Partially ordered sets

➤ A partial order is a binary relation \mathcal{R} over a set \mathcal{S} :

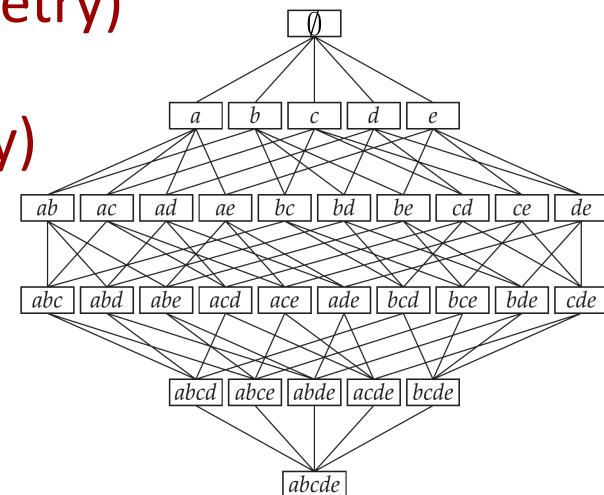
➤ $\forall x, y, z \in \mathcal{S}$

1. $x \mathcal{R} x$ (reflexivity)

2. $x \mathcal{R} y \wedge y \mathcal{R} x \Rightarrow x = y$ (anti-symmetry)

3. $x \mathcal{R} y \wedge y \mathcal{R} z \Rightarrow x \mathcal{R} z$ (transitivity)

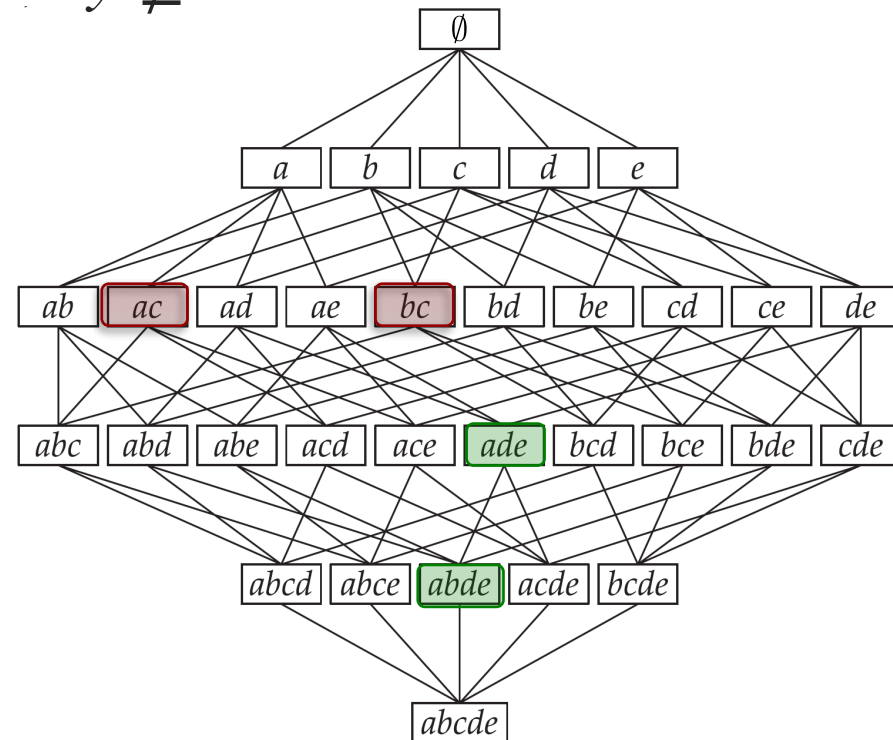
➤ What is \mathcal{S} and \mathcal{S} ?



Poset $(2^I, \subseteq)$

➤ **Comparable** itemsets: $x \subseteq y \vee y \subseteq x$

➤ **Incomparable** itemsets: $x \not\subseteq y \wedge y \not\subseteq x$



Apriori Algorithm [Agrawal and Srikant 1994]

- Determine the support of the **one-element** item sets (i.e. singletons) and discard the **infrequent items**.
- Form candidate itemsets with **two items** (both items must be frequent), determine their support, and discard the **infrequent itemsets**.
- Form candidate item sets with **three items** (all contained pairs must be frequent), determine their support, and discard the **infrequent itemsets**.
- And so on!

Based on **candidate generation** and **pruning**

Apriori Algorithm [Agrawal and Srikant 1994]

Apriori(D, θ):

1. $k \leftarrow 1$

2. $L_k \leftarrow \{i \mid i \in I \wedge \text{freq}(i) \geq \theta\}$

3. while($L_k \neq \emptyset$)

 1. $C \leftarrow \text{aprioriGen}(L_k)$ // new candidates

 2. $k++$

 3. $L_k \leftarrow \{c \mid c \in C \wedge \text{freq}(c) \geq \theta\}$

4. return $\bigcup_i L_i$

Apriori Algorithm [Agrawal and Srikant 1994]

Apriori(D, θ):

1. $k \leftarrow 1$

2. $L_k \leftarrow \{i \mid i \in I \wedge \text{freq}(i) \geq \theta\}$

3. while($L_k \neq \emptyset$)

 1. $C \leftarrow \text{aprioriGen}(L_k)$ // new candidates

 2. $k++$

 3. $L_k \leftarrow \{c \mid c \in C \wedge \text{freq}(c) \geq \theta\}$

4. return $\bigcup_i L_i$

Apriori Algorithm [Agrawal and Srikant 1994]

`aprioriGen(L_k):`

1. $E \leftarrow \emptyset$

2. Foreach $P', P'' \in L_k$ st:

$(P' = \{i_1, \dots, i_{k-1}, i_k\}) \wedge (P'' = \{i_1, \dots, i_{k-1}, i'_k\})$ do

1. $P \leftarrow P' \cup P''$ *// $P = \{i_1, \dots, i_{k-1}, i_k, i'_k\}$*

2. if $\forall i \in P : P \setminus \{i\} \in L_k$ then

1. $E \leftarrow E \cup \{P\}$

3. return E

Improving candidates generation

- Using `aprioriGen` function, an item of $k+1$ size can be generated in a j possible ways:

$$j = \frac{k(k+1)}{2}$$

6 possibilities to generate (abcd)

	abc	abd	acd	bcd
abc	—	abcd	abcd	abcd
abd	abcd	—	abcd	abcd
acd	abcd	abcd	—	abcd
bcd	abcd	abcd	abcd	—

Improving candidates generation

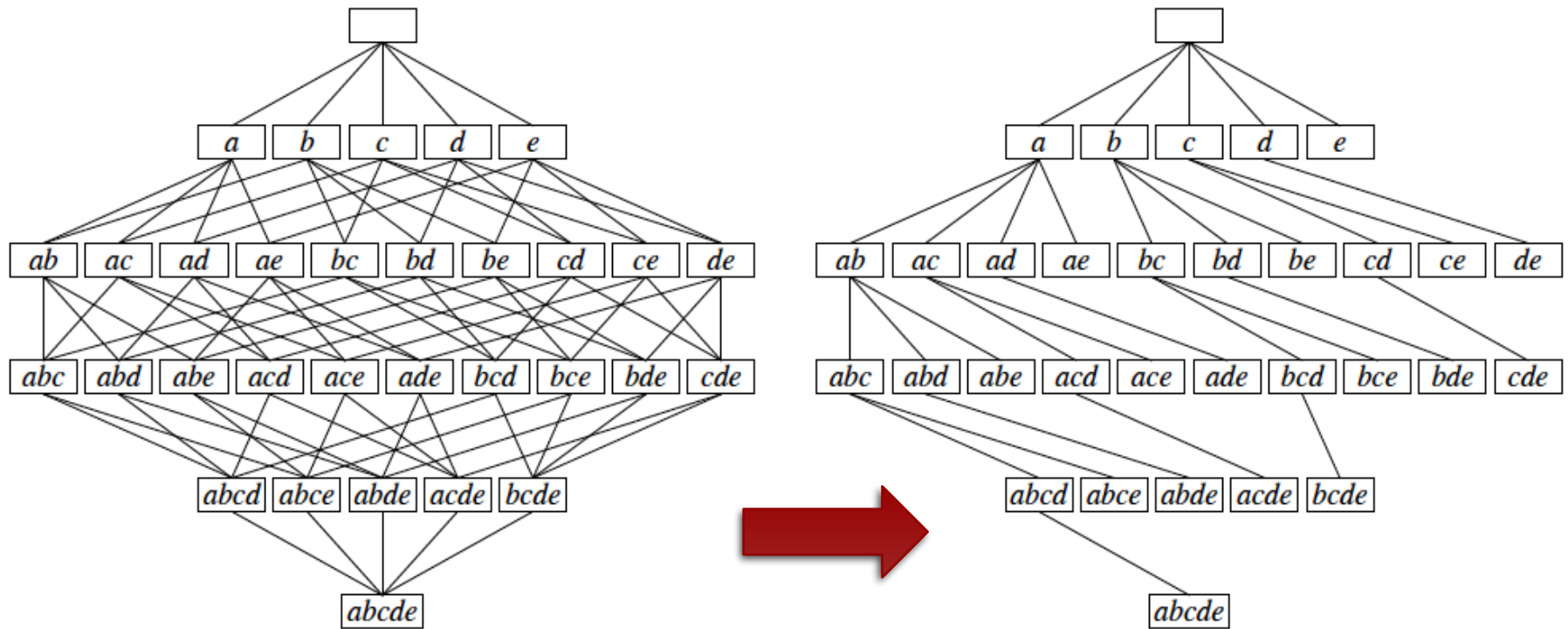
- Using `aprioriGen` function, an item of $k+1$ size can be generated in a j possible ways:

$$j = \frac{k(k+1)}{2}$$

- **Need:** Generate itemset candidate at most once.
- **How:** Assign to each itemset a unique parent itemset, from which this itemset is to be generated

Improving candidates generation

➤ Assigning unique parents turns the poset lattice into a tree:



Canonical form for itemsets

- An itemset can be represented as a word over an alphabet I
 - Q: how many words of k items can we have?
 - A: k -permutations of k items: $k!$
- An arbitrary order (e.g., lexicography order) on items can give a canonical form, a unique representation of itemsets by breaking symmetries.
 - Lex on items : $abc < acb < bac < bca \dots$
 - $\kappa(abc) = \kappa(acb) = \kappa(bac) = \kappa(bca) = abc$
 - $\kappa(abc, 1) = a; \kappa(abc, 2) = b; \kappa(abc, 3) = \kappa(abc, |abc|) = c$

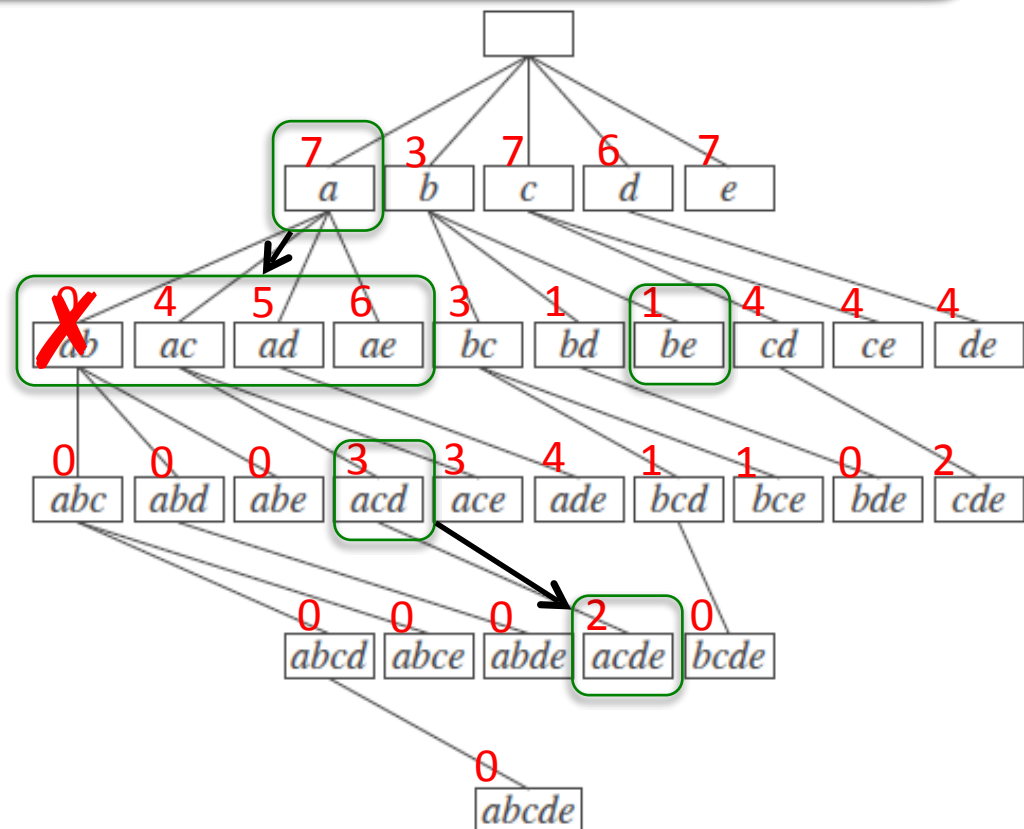
Recursive processing with Canonical forms

- Foreach P of a given level, generate all possible extension of P by one item such that:

$$child(P, \theta) = \{P' : (P' \cup P = \{i\}) \wedge (i \notin P) \wedge (\kappa(P, |P|) < i) \wedge (freq(P') \geq \theta)\}$$

Example (4)

$$child(P, \theta) = \{P' : (P' \cup P = \{i\}) \wedge (i \notin P) \wedge (\kappa(P, |P|) < i) \wedge (freq(P') \geq \theta)\}$$



Items Ordering

- Any order can be used
- The search space differs considerably depending on the order
- Thus, the efficiency of the Frequent Itemset Mining algorithms can differ considerably depending on the item order
- Advanced methods even adapt the order of the items during the search: use different, but “compatible” orders in different branches

Items Ordering (heuristics)

- Frequent itemsets consist of frequent items
 - Sort the items w.r.t. their frequency. (decreasing/increasing)
- The sum of transaction sizes, transaction containing a given item, which captures implicitly the frequency of pairs, triplets etc.
 - Sort items w.r.t. the sum of the sizes of the transactions that cover them.



Tutorials

github.com/FDSInfoMontp-HMIN233/FIM1