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: $I = \{i_1, \ldots, i_n\}$
- Itemset, transaction: $P, T, \subseteq I$
- Transactional dataset: $D = \{T_1, \ldots, T_m\}$
- Language of itemsets: $\mathscr{L}_I = 2^I$
- 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 $I = \{i_1, \ldots, i_n\}$
  - A transactional dataset $D = \{T_1, \ldots, T_m\}$
  - A minimum support θ

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

Example (1)

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

$\mathscr{H}_D$

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

$\mathscr{V}_D$

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

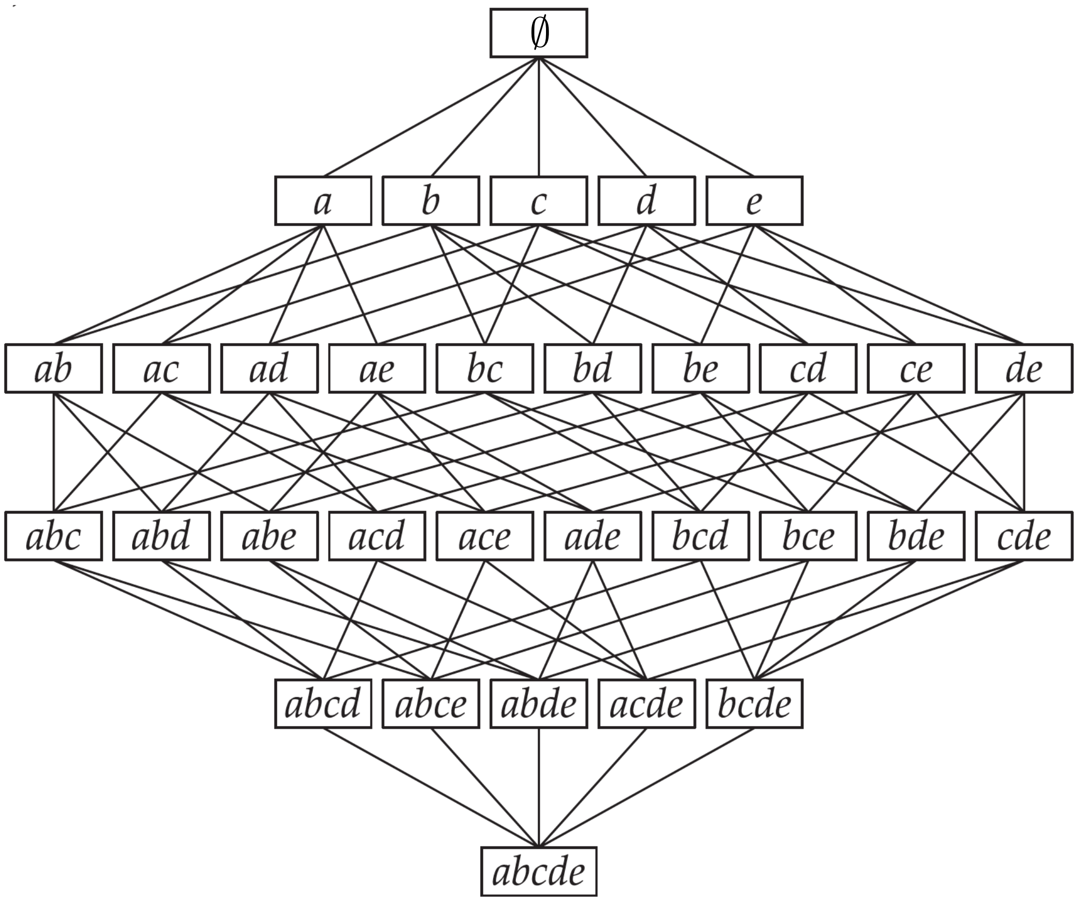
$\mathscr{M}_D$

	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

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

$$freq(bc) = 3$$

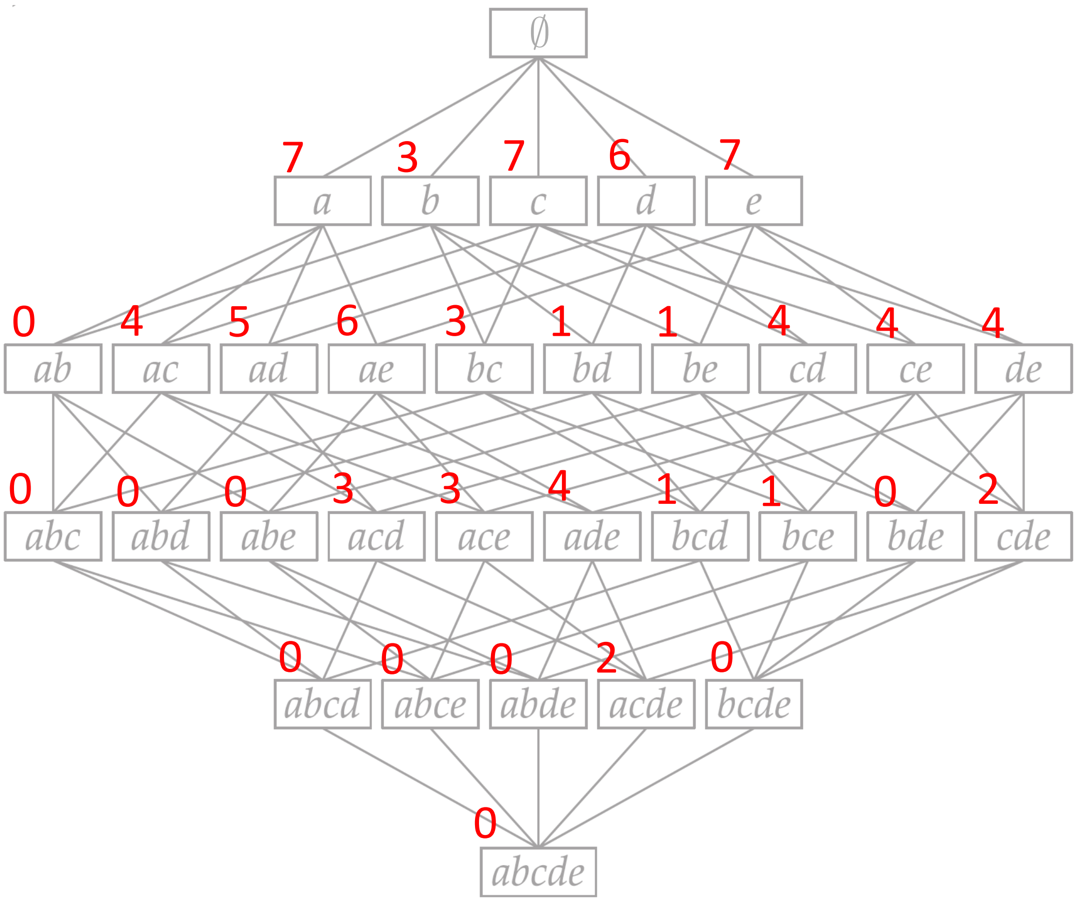
Example (1)



$\mathscr{M}_D$

	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)

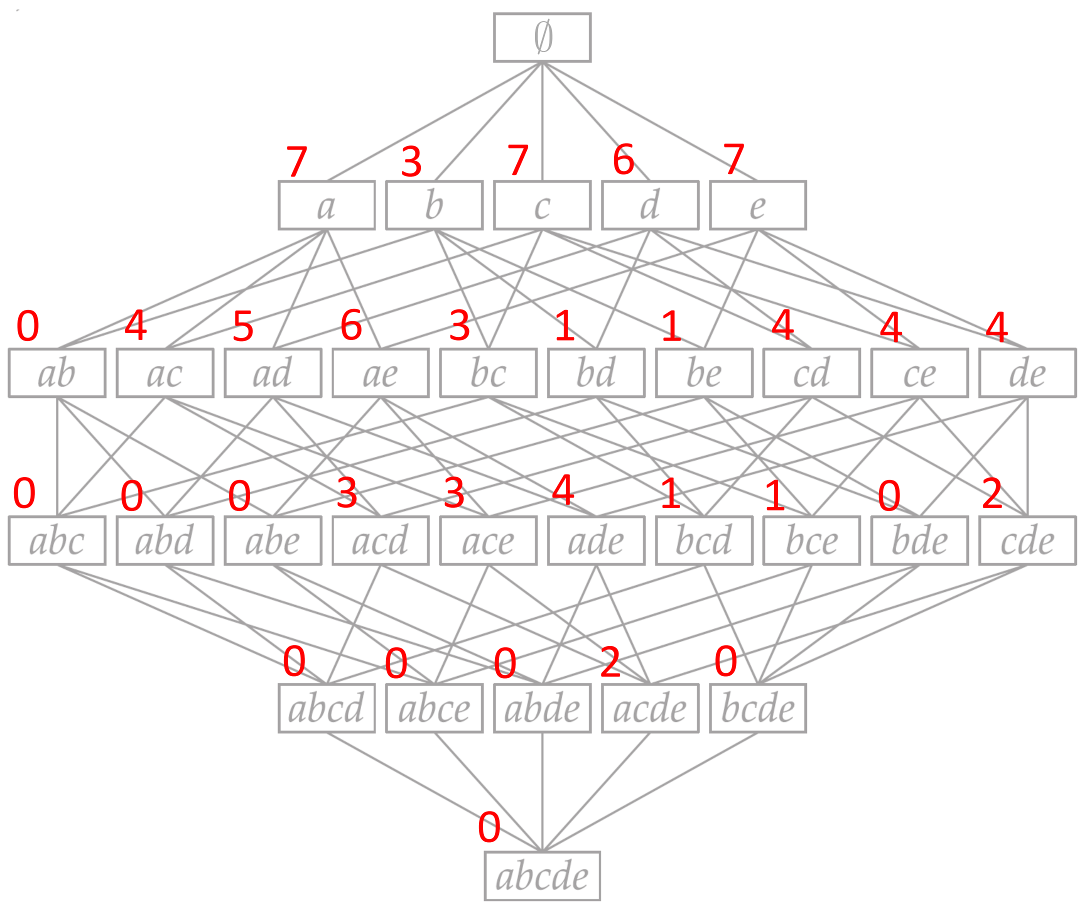


$\mathcal{M}_D$

	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?

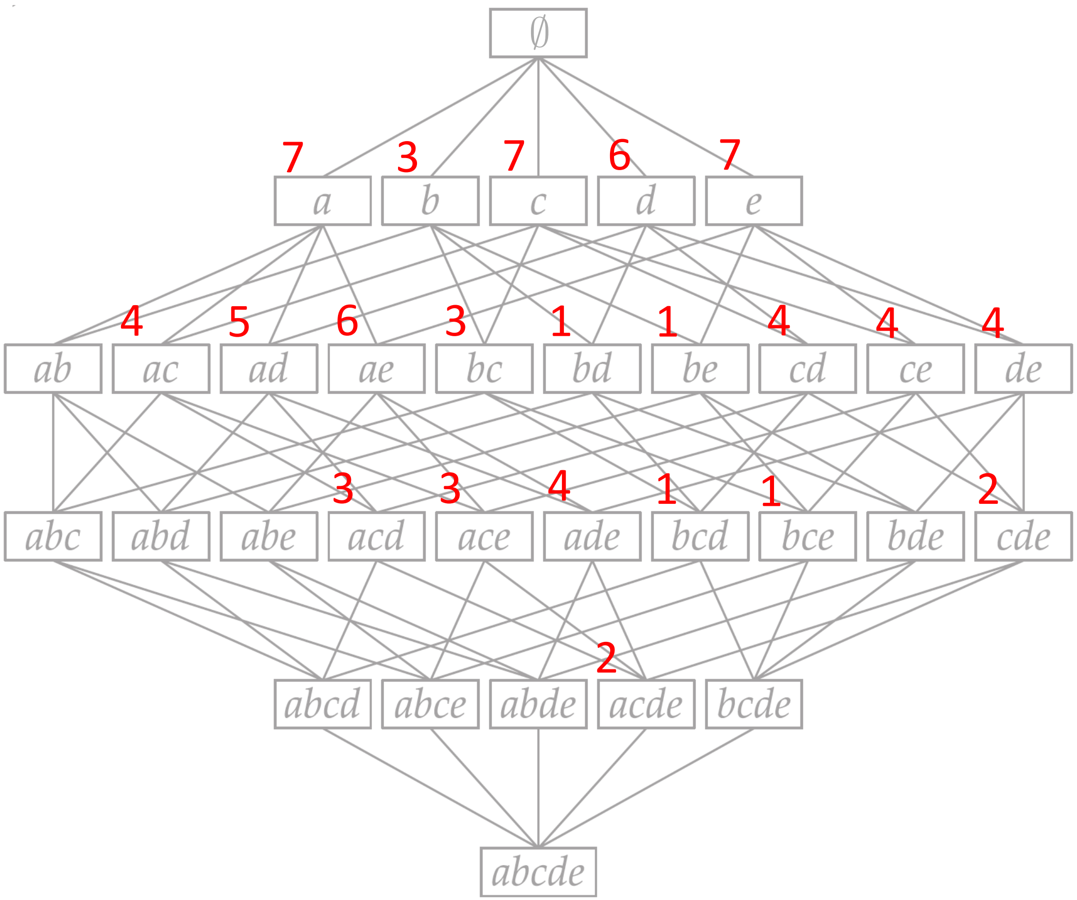


$\mathcal{M}_D$

	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 θ=3?



$\mathscr{M}_D$

	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:

$$X \subseteq Y \Rightarrow cover(Y) \subseteq cover(X)$$

- That is,

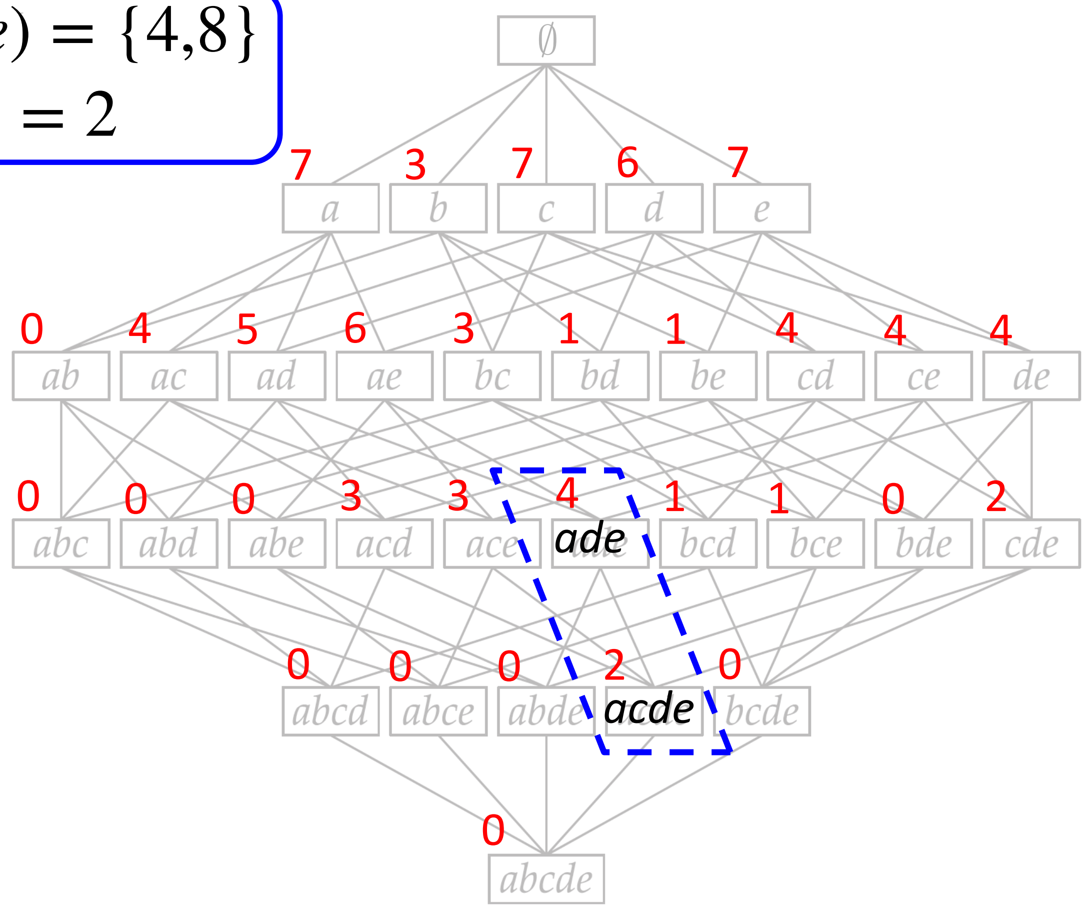
$$X \subseteq Y \Rightarrow freq(Y) \leq freq(X)$$

Example (2)

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

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

$$freq(acde) = 2$$



$\mathscr{M}_D$

	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

Apriori property

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

$$X \subseteq Y \Rightarrow freq(Y) \leq freq(X)$$

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

All subsets of a frequent itemset are frequent!

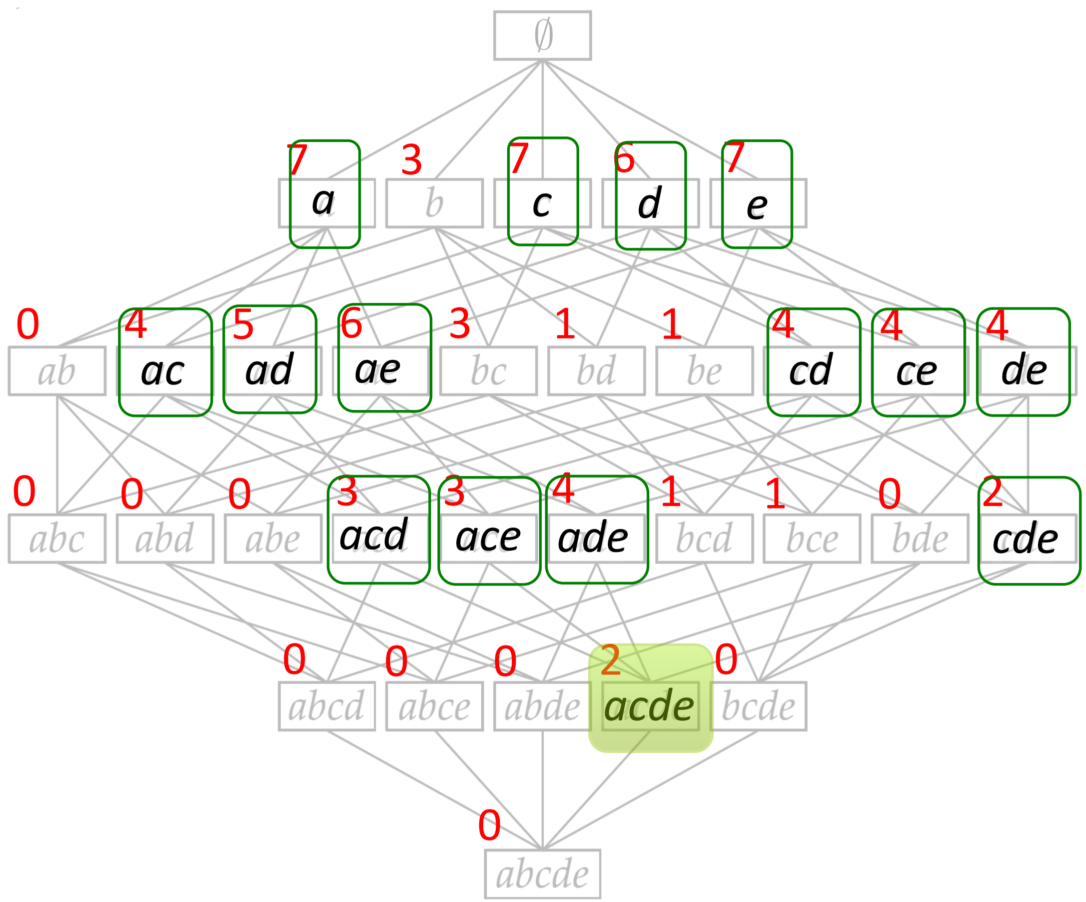
- Contraposition: $X \subseteq Y \Rightarrow (freq(X) < \theta \Rightarrow freq(Y) < \theta)$

All supersets of an infrequent itemset are infrequent!

Example (3)

All subsets of a frequent itemset are frequent!

$\theta = 2$



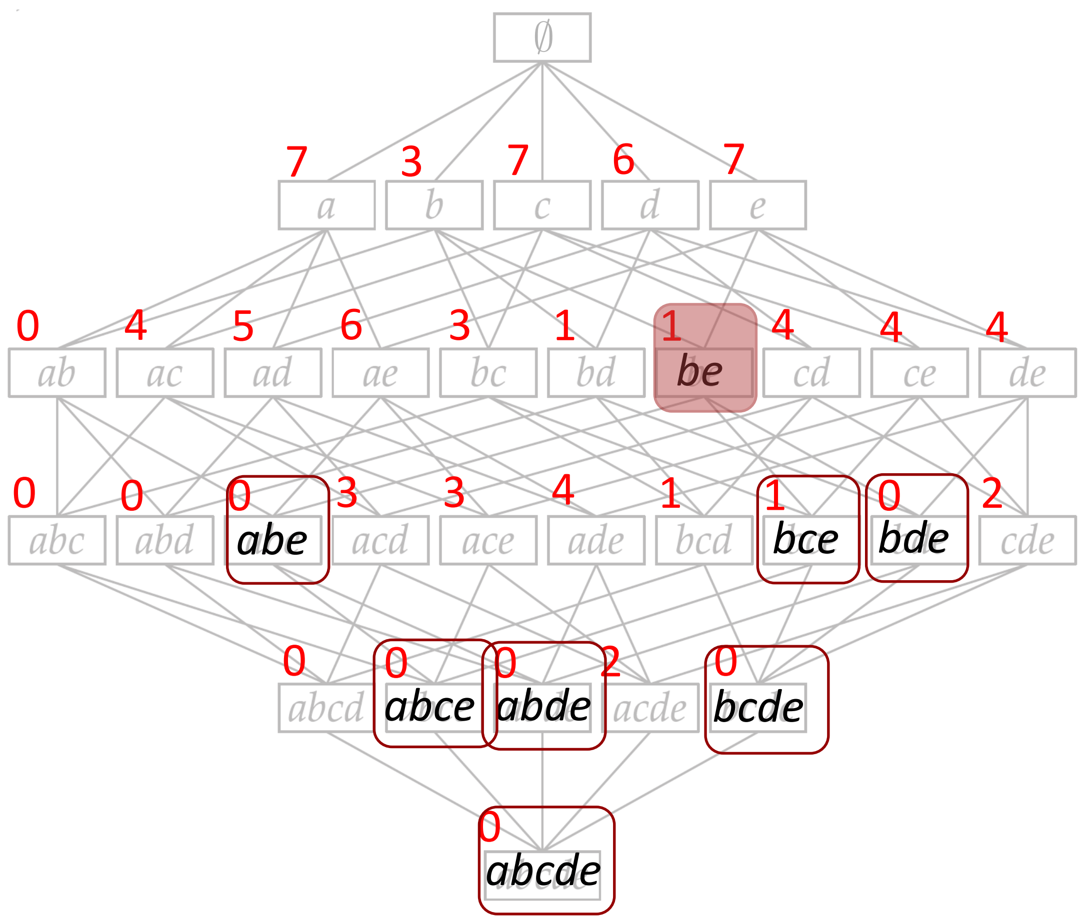
$\mathcal{M}_D$

	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 (3)

All supersets of an infrequent itemset are infrequent!

$\theta = 2$



$\mathscr{M}_D$

	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

Partially ordered sets

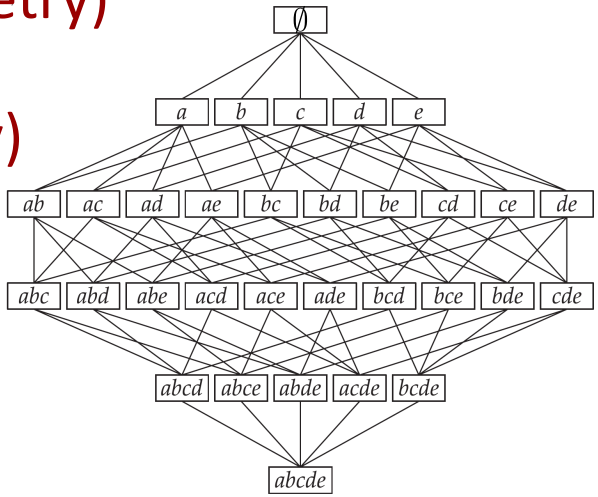
- A partial order is a binary relation $\mathscr{R}$ over a set $\mathscr{S}$:
- $\forall x, y, z \in \mathscr{S}$

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

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

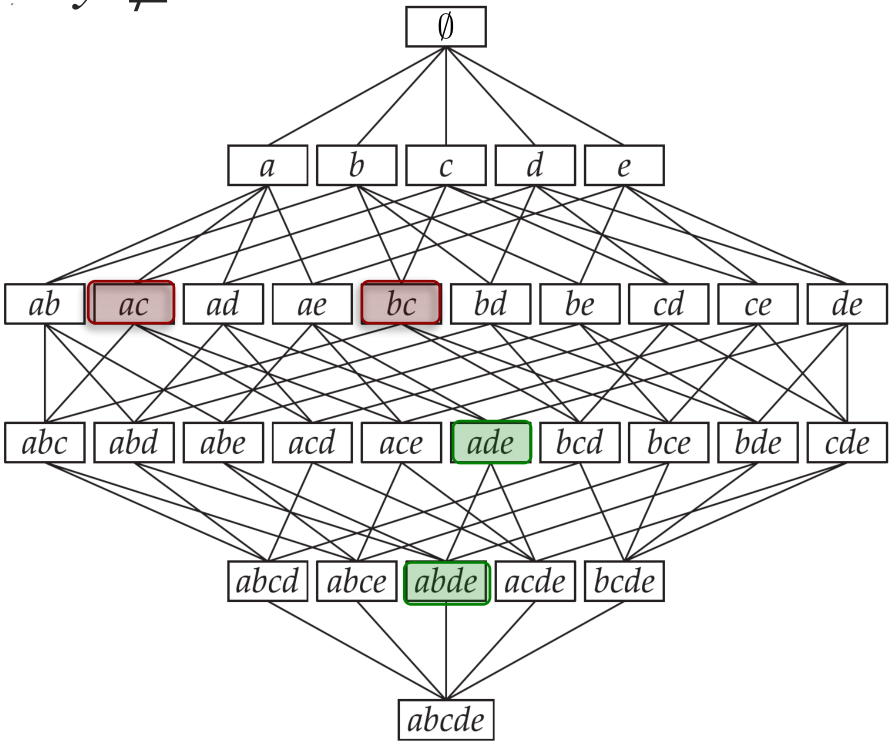
  3. $x \mathscr{R} y \wedge y \mathscr{R} z \Rightarrow x \mathscr{R} z$ (transitivity)

- What is $\mathscr{S}$ and $\mathscr{R}$?



Poset $(2^I, \subseteq)$

- Comparable itemsets: $x \subseteq y \lor y \subseteq x$
- Incomparable itemsets: $x \nsubseteq y \land y \nsubseteq x$



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. $L_k \leftarrow \{i \mid i \in I \wedge freq(i) \geq \theta\}$
3. while($L_k \neq \emptyset$)
    1. $C \leftarrow$ aprioriGen(L_k) // new candidates
    2. $k++$
    3. $L_k \leftarrow \{c \mid c \in C \wedge freq(c) \geq \theta\}$
4. return $\bigcup_i L_i$

Apriori Algorithm [Agrawal and Srikant 1994]

```
Apriori(D, θ):
1. k ← 1
2. L_k ← {i | i ∈ I ∧ freq(i) ≥ θ}
3. while(L_k ≠ ∅)
    1. C ← aprioriGen(L_k)   // new candidates
    2. k++
    3. L_k ← {c | c ∈ C ∧ freq(c) ≥ θ}
4. return ⋃_i L_i
```

Apriori Algorithm [Agrawal and Srikant 1994]

aprioriGen(L_k):

1. $E \leftarrow \emptyset$
2. Foreach $P', P'' \in L_k$ st: $(P' = \{i_1, \ldots, i_{k-1}, i_k\}) \wedge (P'' = \{i_1, \ldots, i_{k-1}, i'_k\})$ do
    1. $P \leftarrow P' \cup P''$ // $P = \{i_1, \ldots, i_{k-1}, i_k, i'_k\}$
    2. if $\forall i \in P : P \backslash \{i\} \in L_k$ 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

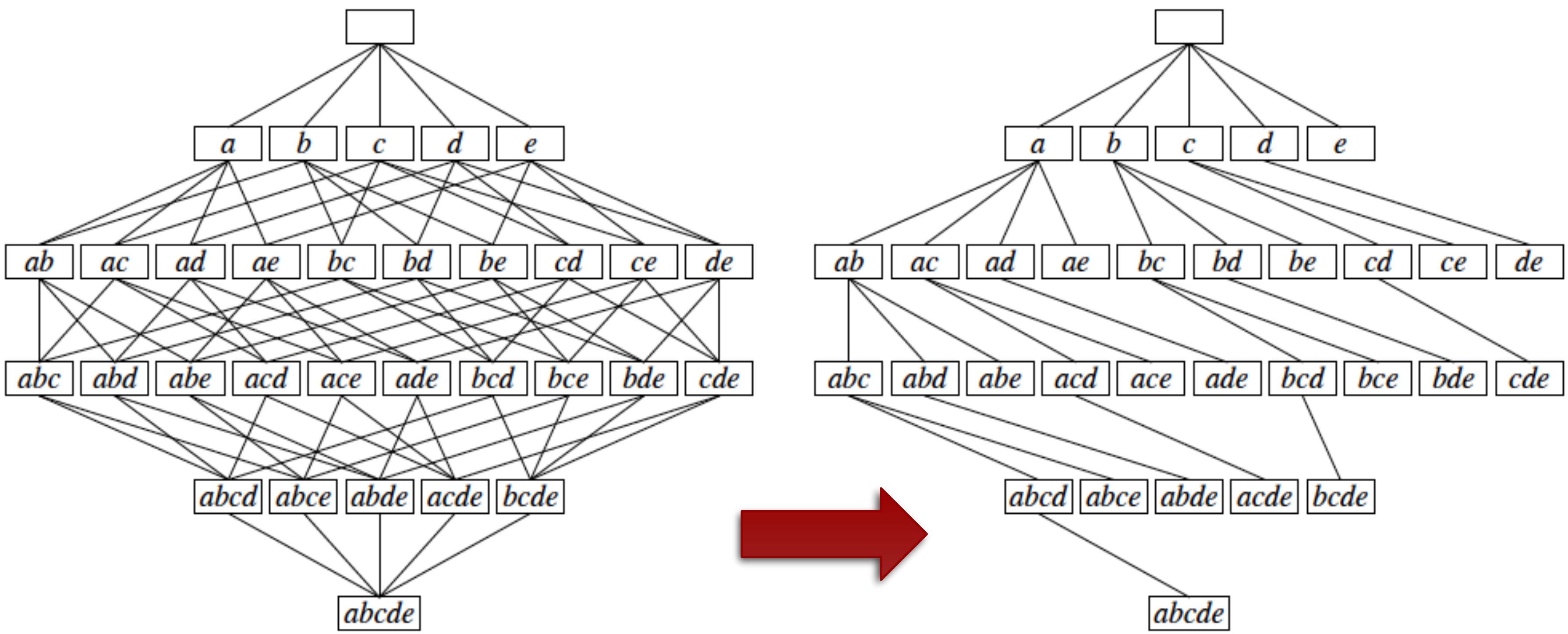
- Using `aprioriGen` function, an item of k+1 size can be generated in a j possible ways:

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

- 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 : $abc < acb < bac < bca\ldots$
  - $\kappa(abc) = \kappa(acb) = \kappa(bac) = \kappa(bca) = abc$
  - $\kappa(abc,1) = a;\ \kappa(abc,2) = b;\ \kappa(abc,3) = \kappa(abc,|abc|) = c$

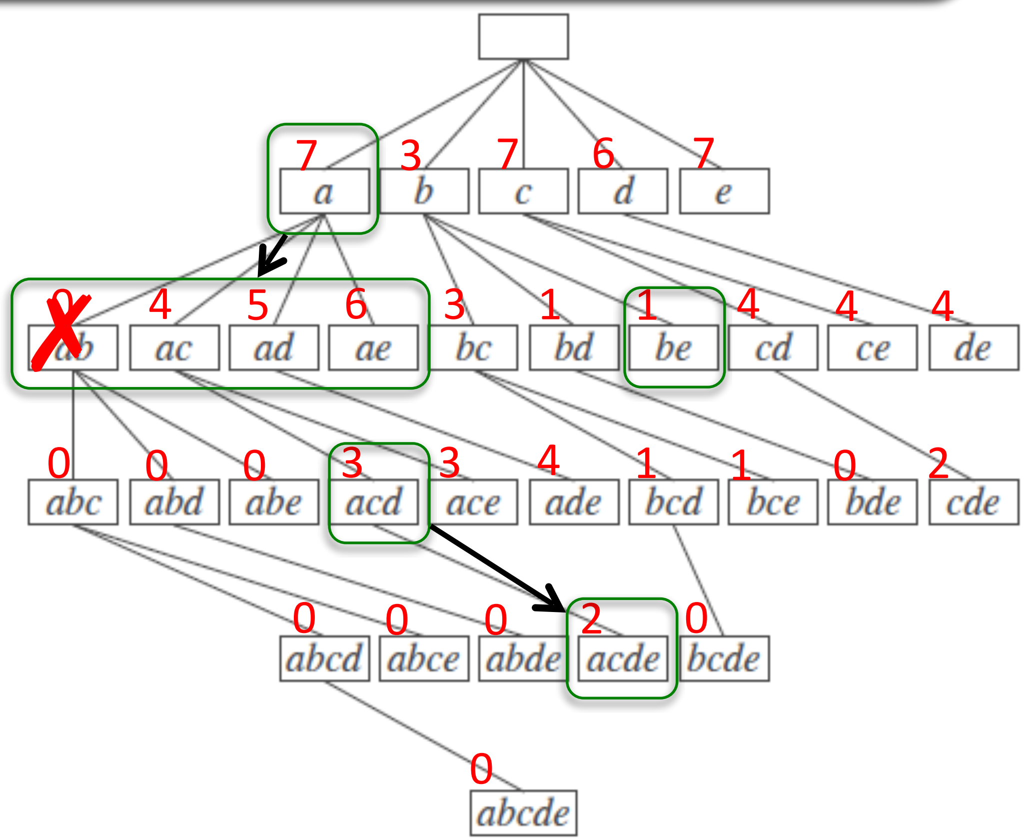
Recursive processing with Canonical forms

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

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

Example (4)

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



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