Frequent Itemset Mining

Nadjib LAZAAR

LIRMM- UM
COCONUT Team

(PART I)

IMAGINA 18/19

Webpage: http://www.lirmm.fr/~lazaar/teaching.html
Email: lazaar@lirmm.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)

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

$\mathcal{H}_\mathcal{D}$

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

horizontal representation

$\mathcal{V}_\mathcal{D}$

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

vertical representation

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

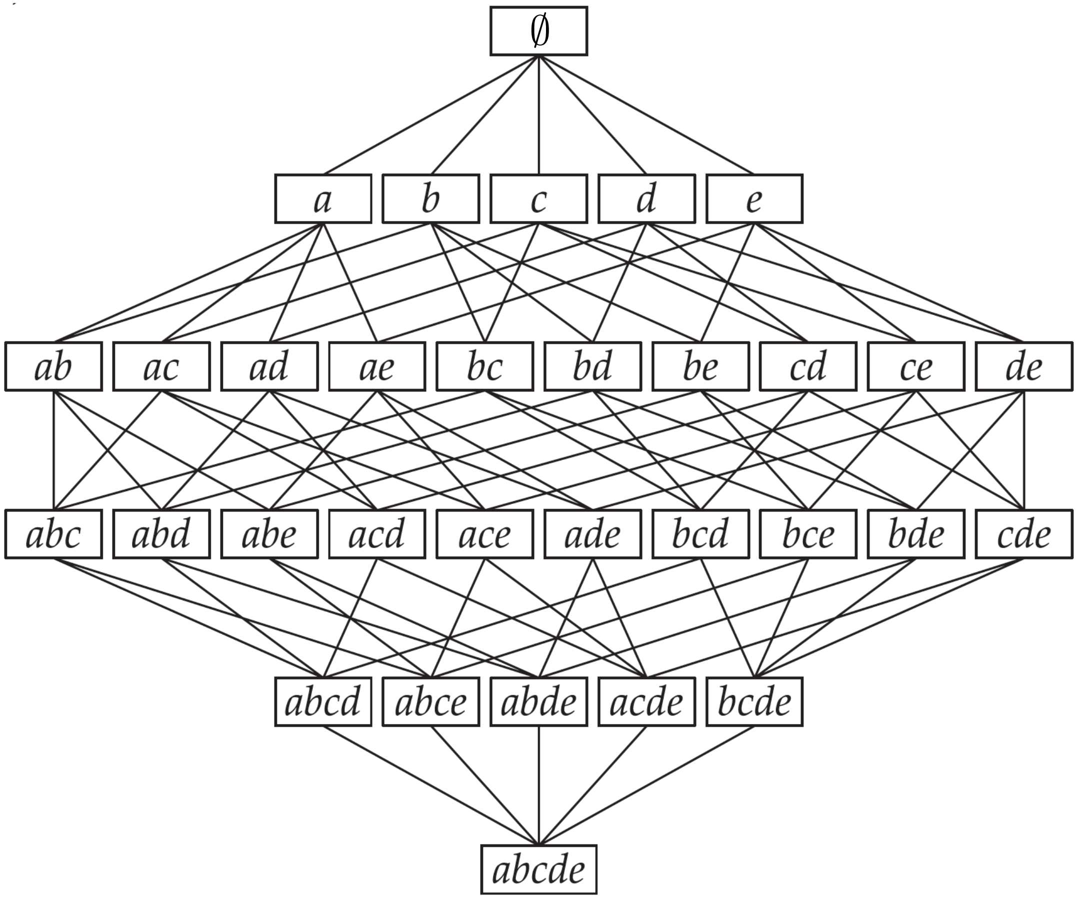
$freq(bc) = 3$

$\mathcal{M}_\mathcal{D}$

	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**

matrix representation

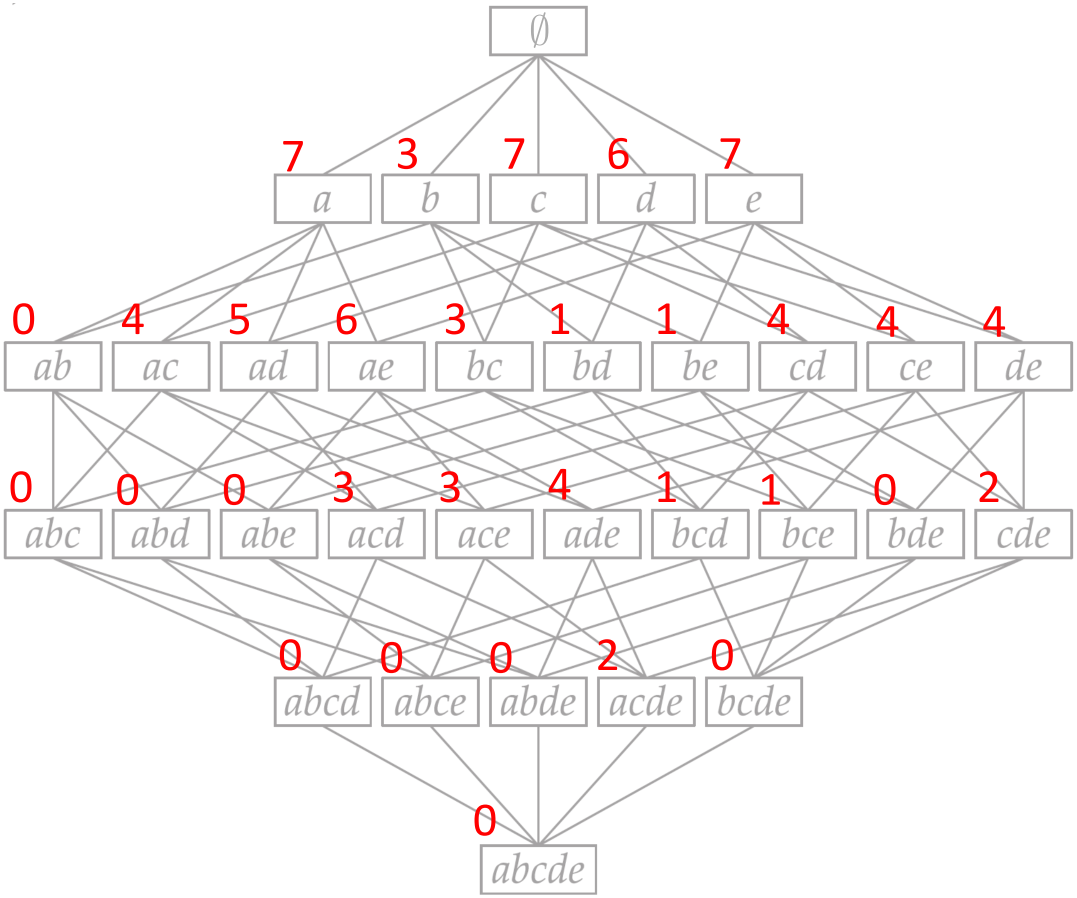
Example (1)



	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**

matrix representation

Example (1)

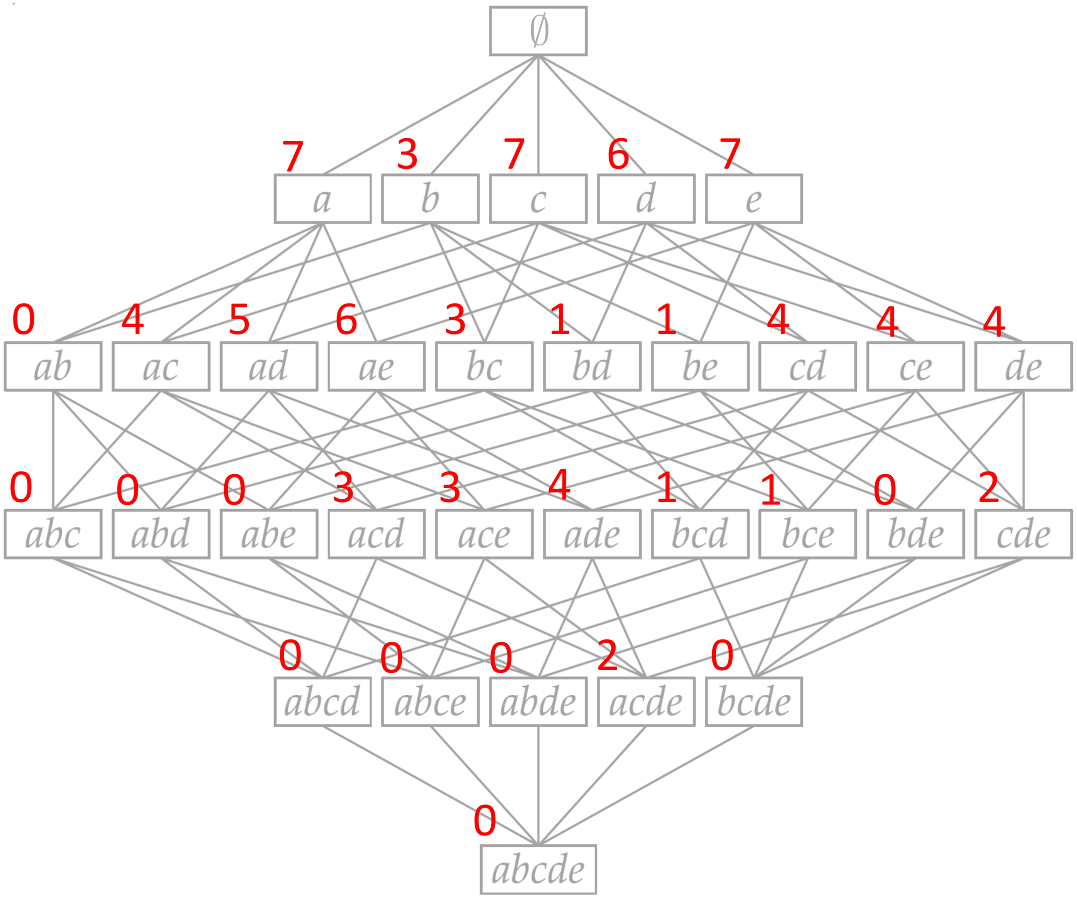


	a	b	c	d	e
1:	**1**	0	0	**1**	**1**
2:	0	**1**	**1**	**1**	0
3:	**1**	0	**1**	0	**1**
4:	**1**	0	**1**	**1**	**1**
5:	**1**	0	0	0	**1**
6:	**1**	0	**1**	**1**	0
7:	0	**1**	**1**	0	0
8:	**1**	0	**1**	**1**	**1**
9:	0	**1**	**1**	0	**1**
10:	**1**	0	0	**1**	**1**

matrix representation

Example (1)

Frequent itemset?

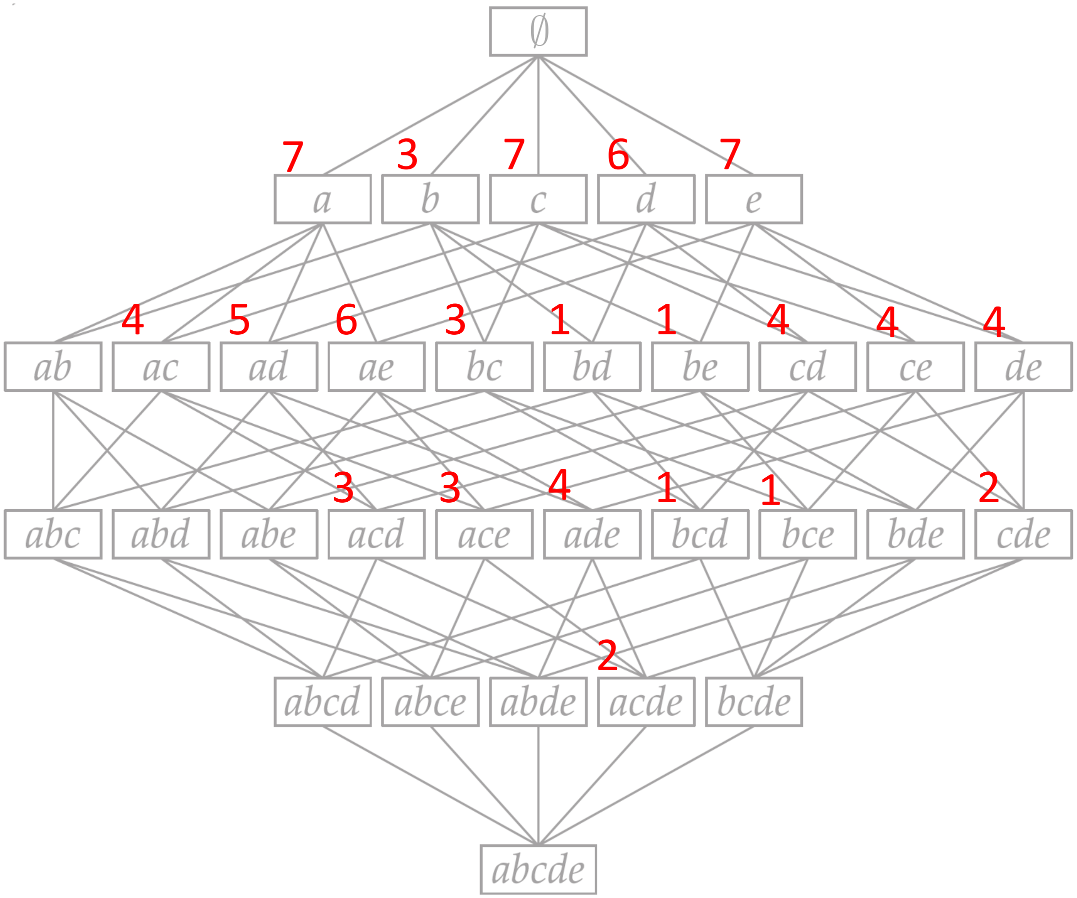


	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

matrix representation

Example (1)

Frequent itemset with minimum support θ=3?



	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**

matrix representation

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	≈ 10^3
20	≈ 10^6
30	≈ 10^9
100	≈ 10^{30}
128	≈ 10^{68} (atoms in the universe)
1000	≈ 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,

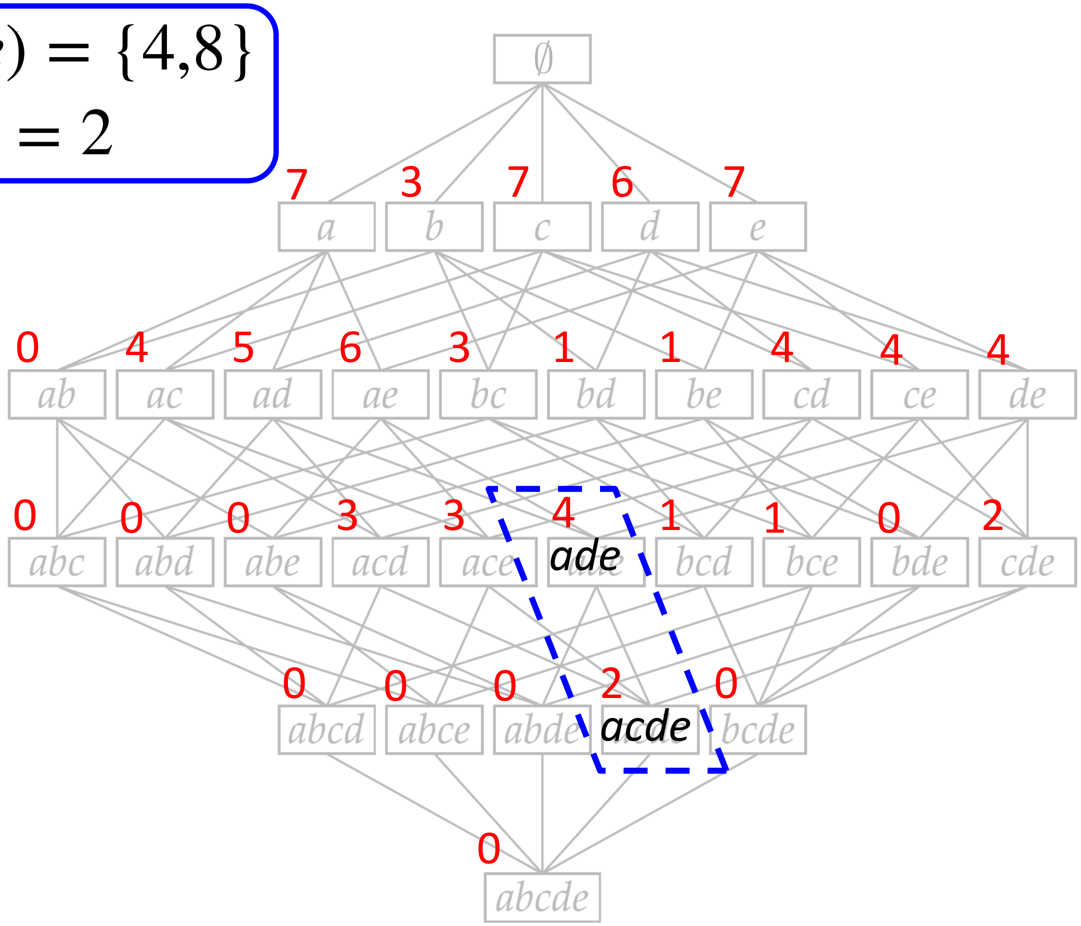
$$X \subseteq Y \Rightarrow freq(Y) \leq freq(X)$$

Example (2)

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

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

$freq(acde) = 2$



	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

matrix representation

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!

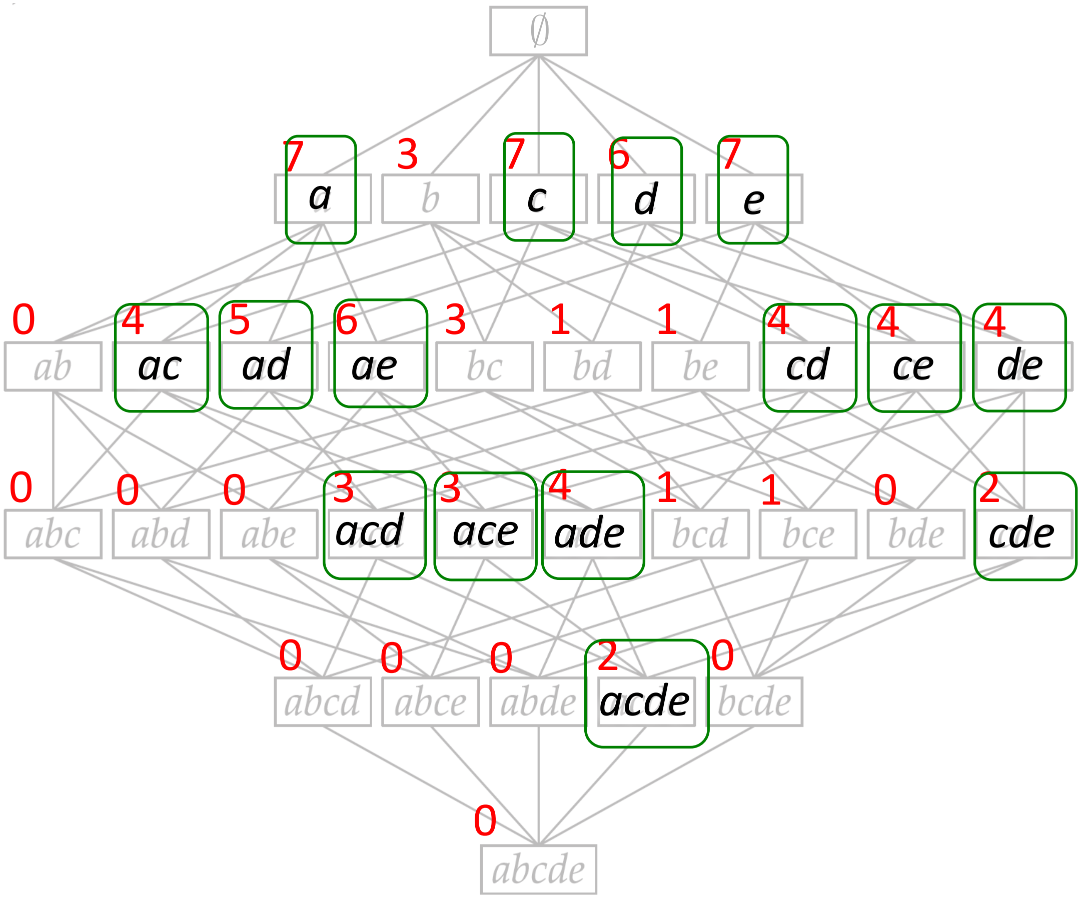
- Contraposition: $X \subseteq Y \Rightarrow (freq(X) < \theta \Rightarrow freq(Y) < \theta)$

All supersets of an infrequent itemset are infrequent!

Example (3)

All subsets of a frequent itemset are frequent!

$\theta = 2$



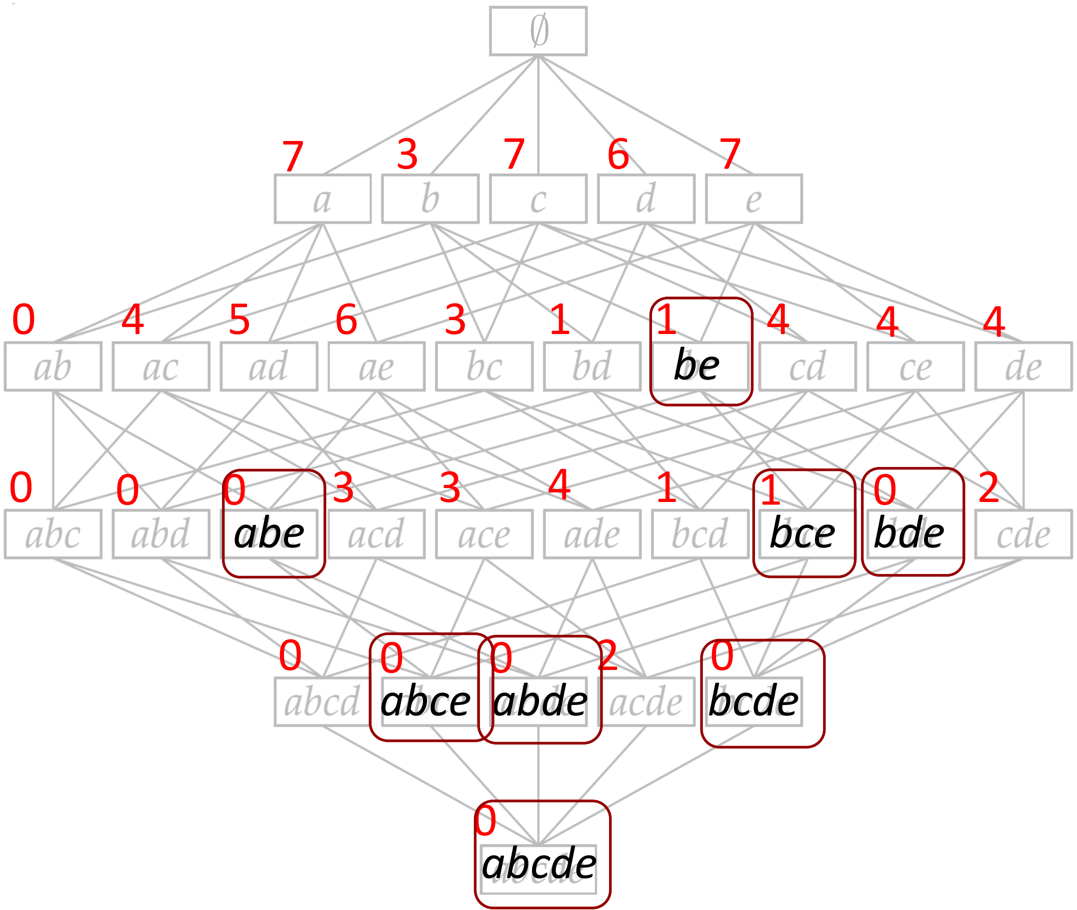
	a	b	c	d	e
1:	**1**	0	0	**1**	**1**
2:	0	**1**	**1**	**1**	0
3:	**1**	0	**1**	0	**1**
4:	**1**	0	**1**	**1**	**1**
5:	**1**	0	0	0	**1**
6:	**1**	0	**1**	**1**	0
7:	0	**1**	**1**	0	0
8:	**1**	0	**1**	**1**	**1**
9:	0	**1**	**1**	0	**1**
10:	**1**	0	0	**1**	**1**

matrix representation

Example (3)

All supersets of an infrequent itemset are infrequent!

$\theta = 2$



	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

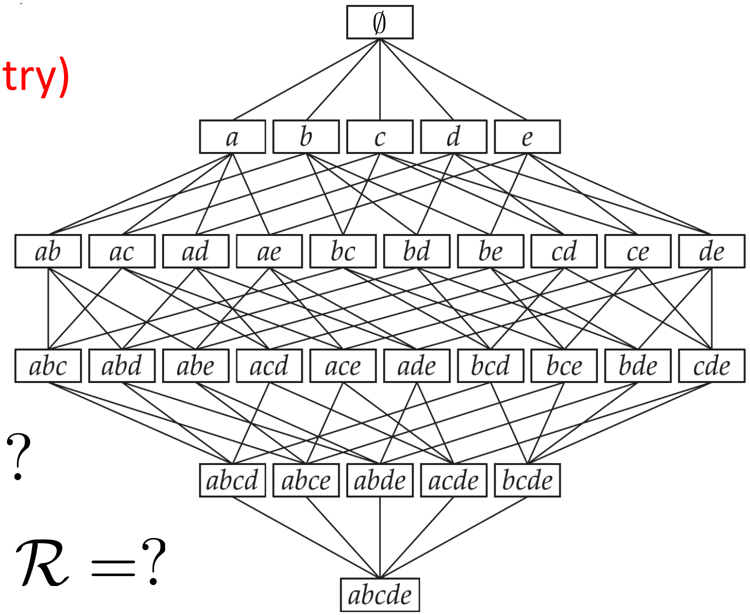
matrix representation

Partially ordered sets

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

$\forall x, y, z \in \mathcal{S}$

- $x \ \mathcal{R} \ x$ (reflexivity)
- $x \ \mathcal{R} \ y \wedge y \ \mathcal{R} \ x \Rightarrow x = y$ (anti-symmetry)
- $x \ \mathcal{R} \ y \wedge y \ \mathcal{R} \ z \Rightarrow x \ \mathcal{R} \ z$ (transitivity)

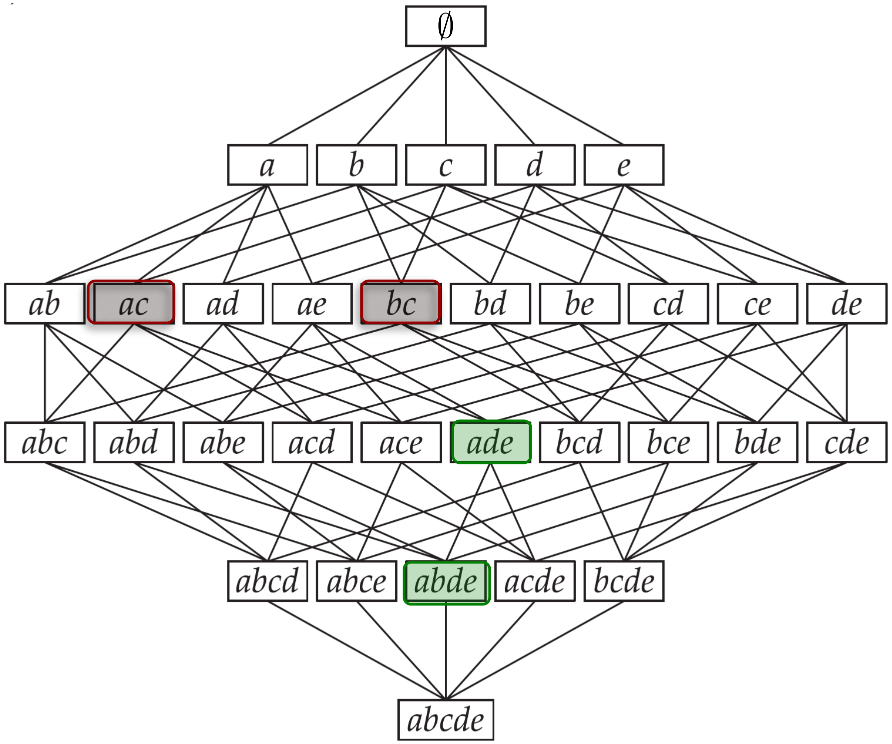


$\mathcal{S} = ?$

$\mathcal{R} = ?$

Poset $(2^{\mathcal{I}}, \subseteq)$

- **Comparable** itemsets: $x \subseteq y \vee y \subseteq x$
- **Incomparable** itemsets: $x \not\subseteq y \wedge y \not\subseteq x$

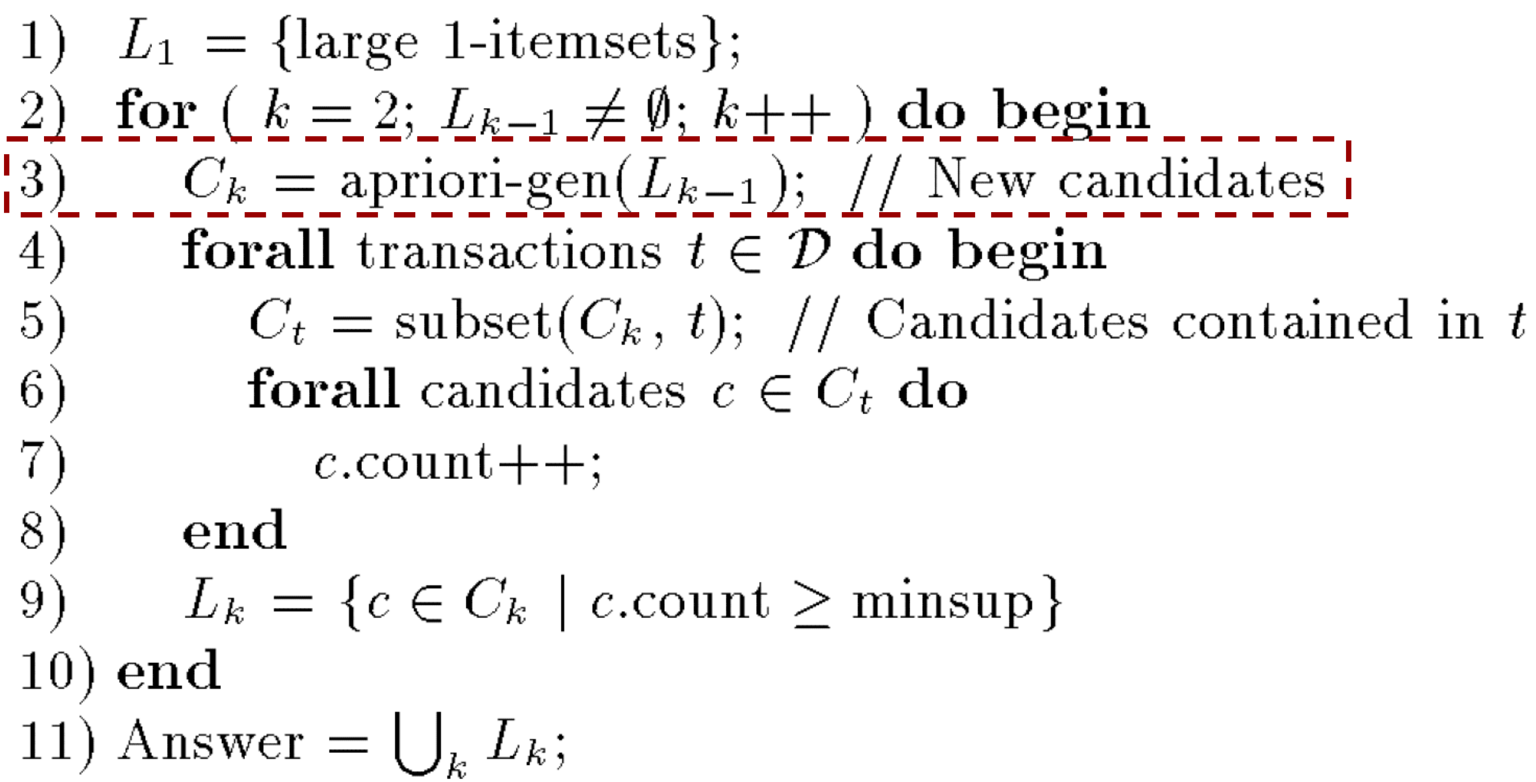


Apriori Algorithm [Agrawal and Srikant 1994]

- 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!

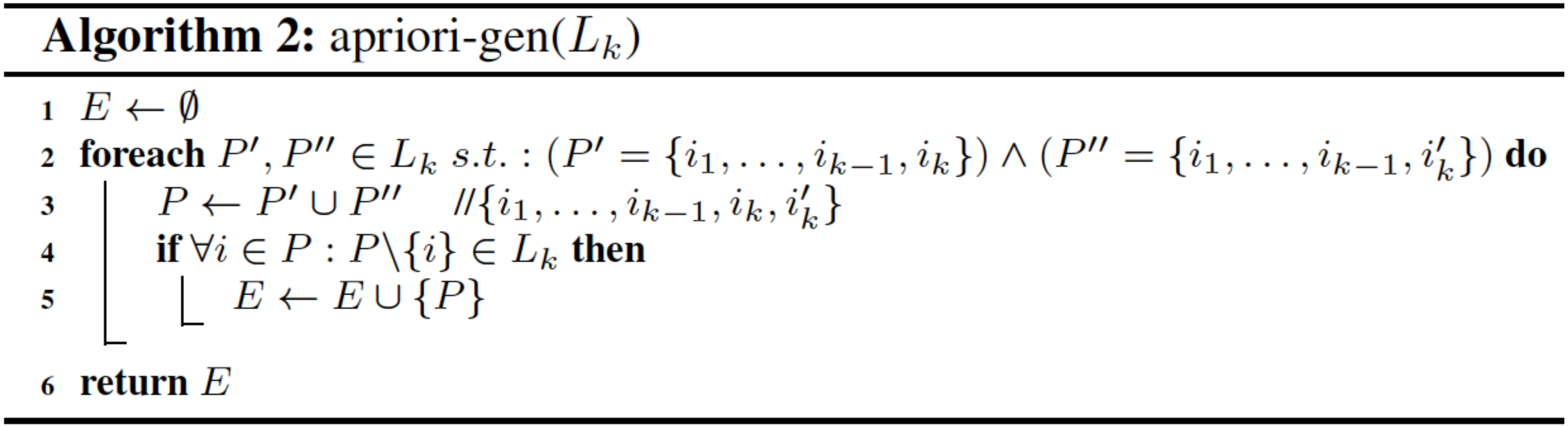
Based on candidate generation and pruning

Apriori Algorithm [Agrawal and Srikant 1994]



```
$L_1$ = {large 1-itemsets};
for ( $k = 2$; $L_{k-1} \neq \emptyset$; $k$++ ) do begin
    $C_k$ = apriori-gen($L_{k-1}$); // New candidates
    forall transactions $t \in \mathcal{D}$ do begin
        $C_t$ = subset($C_k$, $t$); // Candidates contained in $t$
        forall candidates $c \in C_t$ do
            $c$.count++;
    end
    $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$
end
Answer = $\bigcup_k L_k$;
```

Apriori candidates generation



Algorithm 2: apriori-gen(L_k)

```
$E \leftarrow \emptyset$
foreach $P', P'' \in L_k$ $s.t.:(P' = \{i_1, \dots, i_{k-1}, i_k\}) \wedge (P'' = \{i_1, \dots, i_{k-1}, i'_k\})$ do
    $P \leftarrow P' \cup P''$    //$\{i_1, \dots, i_{k-1}, i_k, i'_k\}$
    if $\forall i \in P : P \backslash \{i\} \in L_k$ then
        $E \leftarrow E \cup \{P\}$
return $E$
```

Improving candidates generation

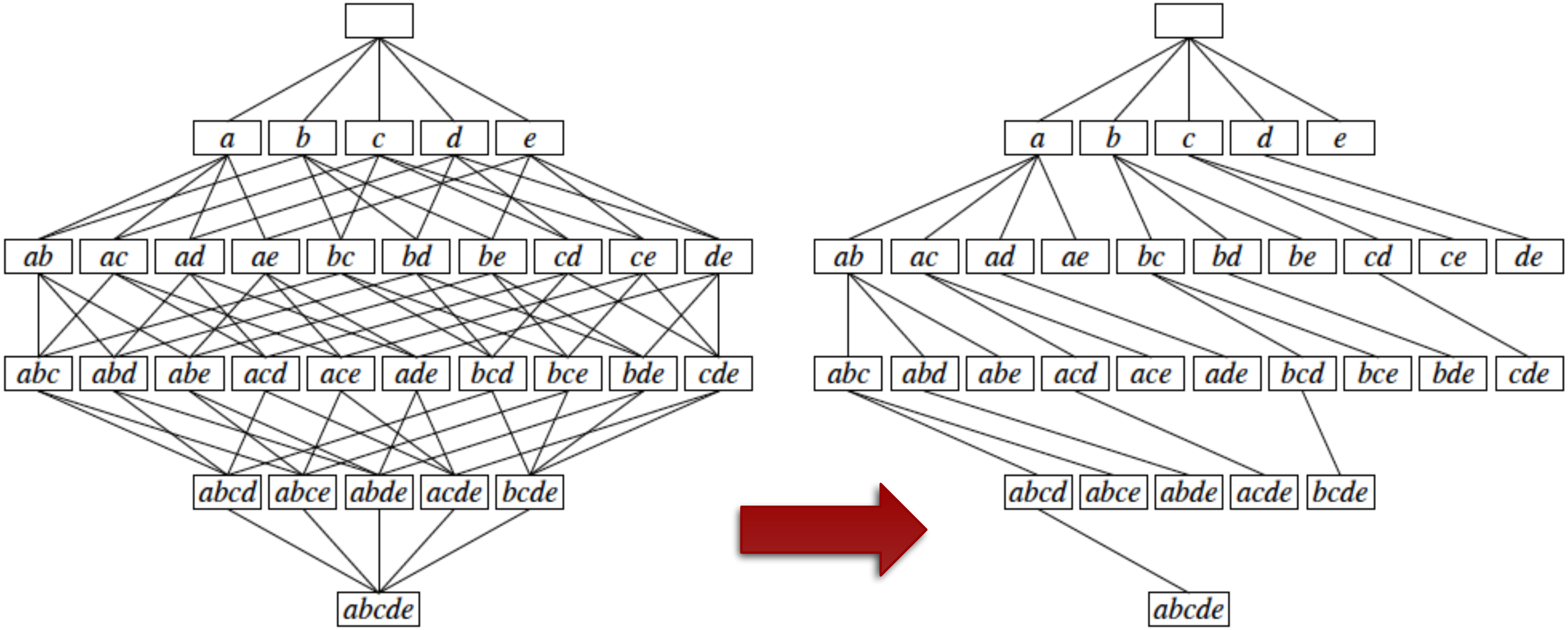
- Using `apriori-gen` function, an item of k+1 size can be generated in a j possible ways:

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

- 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 $\mathcal{I}$
- Q: how many words of 3 items can we have? Of 4 items? 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 :

$$abc < acb < bac < bca \ldots$$

Recursive processing with Canonical forms

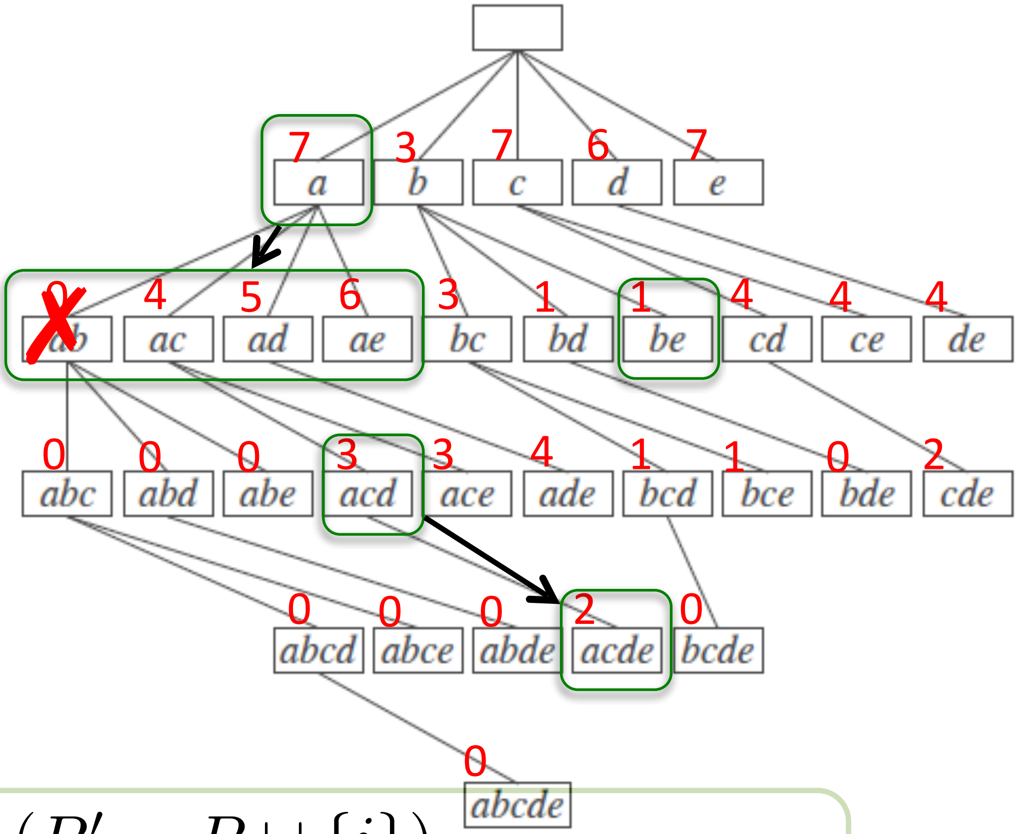
- Foreach P of a given level, generate all possible extension of P by one item such that:

$$
\begin{aligned}
child(P) = \{P' : (i \notin P) \wedge (P' = P \cup \{i\}) \\
\wedge (c(P).last < i) \wedge (P' \text{ is frequent})\}
\end{aligned}
$$

- Foreach P’, process it recursively.

Example (4)

Q: what are the children of:



$$child(P) = \{P' : (i \notin P) \wedge (P' = P \cup \{i\}) \\ \wedge (c(P').last < i) \wedge (P' \text{ is frequent})\}$$

Items Ordering

- Any order can be used, that is, the order is arbitrary
- 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

link: http://www.lirmm.fr/~lazaar/imagina/TD1.pdf