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

- Relative Frequency: $freq(P) = \frac{1}{|D|}|cover(P)|$
- Absolute Frequency: $freq(P) = |cover(P)|$

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

$\mathscr{H}_D$

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	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

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

$freq(bc) = 3$

$\mathscr{M}_D$

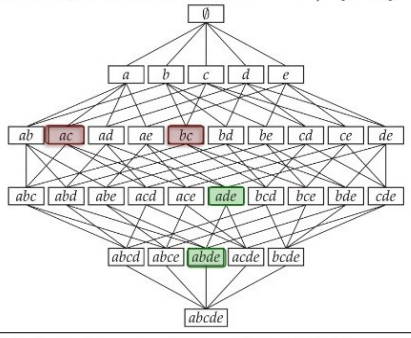
	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

- 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.: $freq(P) \geq \theta$

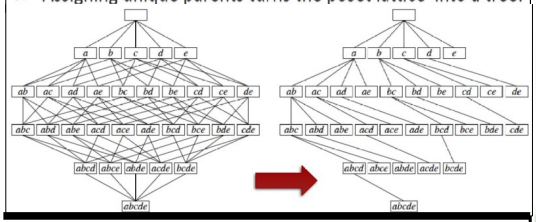
```
Apriori(D, θ):
1. k ← 1
2. L_k ← {i | i ∈ I ∧ freq(i) ≥ θ}
3. while(L_k ≠ ∅)
     1. C ← aprioriGen(L_k)   // new candidates
     2. k++
     3. L_k ← {c | c ∈ C ∧ freq(c) ≥ θ}
4. return ⋃_i L_i
```

```
aprioriGen(L_k):
1. E ← ∅
2. Foreach P', P'' ∈ L_k st:
   (P' = {i_1, ..., i_{k-1}, i_k}) ∧ (P'' = {i_1, ..., i_{k-1}, i'_k}) do
     1. P ← P' ∪ P''   // P = {i_1, ..., i_{k-1}, i_k, i'_k}
     2. if ∀i ∈ P : P\{i} ∈ L_k then
          1. E ← E ∪ {P}
3. return E
```

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



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



- Foreach P of a given level, generate all possible extension of P by one item such that:
- Using aprioriGen function, an item of k+1 size can be generated in a j possible ways:

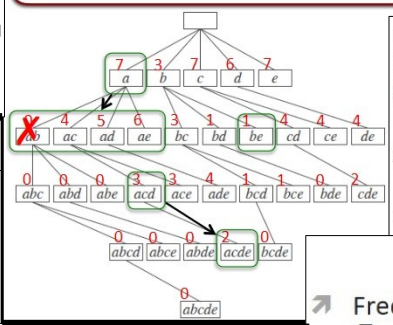
$$j = \frac{k(k+1)}{2}$$

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

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	—

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



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

Maximal Itemsets

- Every frequent itemset has a maximal superset:
  $\forall \theta, \forall P \in F_\theta : (\exists P' \in M_\theta : P \subseteq P')$
- The maximal itemsets are a condensed representation of the frequent itemsets where: $\forall \theta : F_\theta = \bigcup_{P \in M_\theta} 2^P$

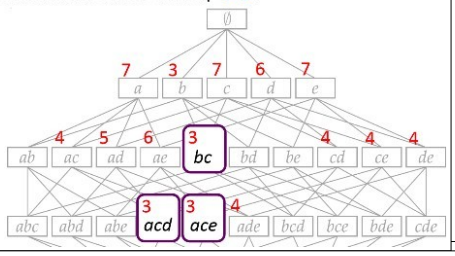
$M_\theta = \{P \subset I \mid freq(P) \geq \theta \wedge \forall P' \supset P : freq(P') < \theta\}$

Number of items (n)
Search Space (2^n)

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.

Here are the Frequent itemset with minsup θ=3
Q: What are the maximal itemsets minsup θ=3?



Ensemble des maximaux d'un minimum support θ = 1 D = {abef, bdef, acd, ac}
M 1 = {abef, bdef, acd} M 1 est toujours un sous ensemble de la base

Vocabulaire:
Itemset maximaux : feuille de l'abre (plus long mot avec fréquence >=θ)
ItemSet fréquents : tout les élément avec fréquence >= θ
ItemSet clos : Union des ItemSet maximaux avec frequence de θ à maxFrequence

$C_\theta = \{P \subset I \mid freq(P) \geq \theta \wedge \forall P' \supset P : freq(P') < freq(P)\}$