FREQUENT PATTERN MINING

FREQUENT PATTERN MINING

- Frequent Pattern mining/ FP discovery
  - Objective: find items that occur frequently together in a database
  - Algorithmically difficult problem
- Association Rule Learning
  - From frequent patterns,
    - Identify statistically relevant associations

MARKET BASKET ANALYSIS

- Typical example: Market Basket Analysis
  - Database: people buying products
    - One reason why supermarkets/shops propose Loyalty programs

- If you buy tomatoes, onions and hamburger patties, you will probably buy hamburger breads

- Famous unexpected association:
  - Beers and Diapers
  - (Probably a legend…)

MARKET BASKET ANALYSIS

- Usage of market basket analysis:
  - Put one object on sale, to favor selling the other ones
    - Sales on burger breads=>consumer buy tomatoes, onion and beef patty
  - Put products close/far away
    - Men buying diapers tempted to buy beers ? Put beers close to diapers
- Relevant in other contexts of course
  - Relation between medical condition and life habits
    - Smoking + cholesterol=>heart disease…
    - High pH + bacteria1=> mosquito development

DATASETS

- Type of data: list of itemsets
  - 1={milk, bread,fruit}
  - 2={butter,eggs,fruit}
  - 3={beer,diapers}
  - 4={milk, bread, butter,eggs,fruit}
  - 5={bread}

transaction ID	milk	bread	butter	beer	diapers	eggs	fruit
1	1	1	0	0	0	0	1
2	0	0	1	0	0	1	1
3	0	0	0	1	1	0	0
4	1	1	1	0	0	1	1
5	0	1	0	0	0	0	0

DEFINITIONS

- **Items**: $I = \{i_1, i_2, \ldots, i_n\}$
  - Unique item (butter, milk, etc)
- **Transaction**
  - $(t_i \subseteq I)$, arbitrary size
- **Database** $D = \{t_1, t_2, \ldots, t_m\}$
  - Collection of **transactions**
- **Itemset**: set of items of arbitrary size $(X \subseteq I)$
  - A subset we are interested in

DEFINITIONS

- Absolute Support of itemset X in D:
  - Number of transactions containing X (i.e., $|\{t \in D / X \subseteq t\}|$)
- Relative support (or simply *Support*)
  - Fraction of transactions containing X
    - $\frac{\text{abs\_support}(X)}{|D|}$
  - Estimation of $P(X)$
    - Probability for a random transaction to contain X
- **Frequent** itemset:
  - Itemset with support ≥ min_supp

SUPPORT

- Support {Milk,bread} ?
- Support {diapers,beer} ?

transaction ID	milk	bread	butter	beer	diapers	eggs	fruit
1	1	1	0	0	0	0	1
2	0	0	1	0	0	1	1
3	0	0	0	1	1	0	0
4	1	1	1	0	0	1	1
5	0	1	0	0	0	0	0

SUPPORT

- Support {Milk,bread} = 2/5
- Support {diapers,beer}=1/5

transaction ID	milk	bread	butter	beer	diapers	eggs	fruit
1	1	1	0	0	0	0	1
2	0	0	1	0	0	1	1
3	0	0	0	1	1	0	0
4	1	1	1	0	0	1	1
5	0	1	0	0	0	0	0

DEFINITIONS

- Association rule : rule of the form
  - $X \rightarrow Y$
    - $X \subseteq I, Y \subseteq I$
    - $X \cap Y = \emptyset$
  - Meaning: If X is in a transaction, then Y too

- Support of $X \rightarrow Y$:
  - => Support of itemset $W = X \cup Y$

- For an association to be interesting, we further look at interest scores
  - Else, risk of finding spurious associations

SCORES OF INTEREST

CONFIDENCE

- $\text{conf}(X \Rightarrow Y) = P(Y|X) = \dfrac{\text{supp}(X \cap Y)}{\text{supp}(X)} = \dfrac{\text{number of transactions containing } X \text{ and } Y}{\text{number of transactions containing } X}$

- Fraction of transactions containing X that also contains Y
  - An itemset/rule can be frequent because its elements are frequent
  - We want to know if Y is frequent when we have X

- Non-symmetric

transaction ID	milk	bread	butter	beer	diapers	eggs	fruit
1	1	1	0	0	0	0	1
2	0	0	1	0	0	1	1
3	0	0	0	1	1	0	0
4	1	1	1	0	0	1	1
5	0	1	0	0	0	0	0

- Confidence Milk=>bread
- Confidence bread=>milk
- Confidence diapers=>beer
- Confidence beer=>diapers

transaction ID	milk	bread	butter	beer	diapers	eggs	fruit
1	1	1	0	0	0	0	1
2	0	0	1	0	0	1	1
3	0	0	0	1	1	0	0
4	1	1	1	0	0	1	1
5	0	1	0	0	0	0	0

- Confidence Milk=>bread = 2/2=1
- Confidence bread=>milk = 2/3
- Confidence diapers=>beer=1/1
- Confidence beer=>diapers= 1/1

LIFT

- If Y has high confidence, but is also frequent, confidence is not enough.
  - If both are frequent, by chance, they appear frequently together
  - $\text{lift}(X \Rightarrow Y) = \dfrac{\text{confidence}(X \Rightarrow Y)}{\text{supp}(Y)},$
    - Compares Y presence when X with Y in general
  - $$\textbf{lift}(X \Rightarrow Y) = \frac{\textbf{supp}(X \cap Y)}{\textbf{supp}(X) \times \textbf{supp}(Y)}$$
    - Compares observed co-presence with expected co-presence
- [0,+inf]
  - X and Y are independent: lift=1

transaction ID	milk	bread	butter	beer	diapers	eggs	fruit
1	1	1	0	0	0	0	1
2	0	0	1	0	0	1	1
3	0	0	0	1	1	0	0
4	1	1	1	0	0	1	1
5	0	1	0	0	0	0	0

- Lift Milk=>bread?
- Lift beer=>diapers?

transaction ID	milk	bread	butter	beer	diapers	eggs	fruit
1	1	1	0	0	0	0	1
2	0	0	1	0	0	1	1
3	0	0	0	1	1	0	0
4	1	1	1	0	0	1	1
5	0	1	0	0	0	0	0

- Lift Milk=>bread
  - (2/5)/(6/25)=1.666
  - (1)/(3/5)=1.666
- Lift beer=>diapers
  - (1/5)/(1/25)=5
  - (1)/(1/5)=5

LEVERAGE

- levarage$(A \rightarrow C)$ = support$(A \rightarrow C)$ − support$(A) \times$ support(C), range: $[-1,1]$
    - Difference between the observed frequency of A and C appearing together and the frequency that would be expected if A and C were independent
- 0 indicates independence
- =>Take also into account how frequent the items are
    - On top of how exceptionally frequent

transaction ID	milk	bread	butter	beer	diapers	eggs	fruit
1	1	1	0	0	0	0	1
2	0	0	1	0	0	1	1
3	0	0	0	1	1	0	0
4	1	1	1	0	0	1	1
5	0	1	0	0	0	0	0

- Leverage Milk=>bread
- Leverage beer=>diapers

transaction ID	milk	bread	butter	beer	diapers	eggs	fruit
1	1	1	0	0	0	0	1
2	0	0	1	0	0	1	1
3	0	0	0	1	1	0	0
4	1	1	1	0	0	1	1
5	0	1	0	0	0	0	0

- Leverage Milk=>bread
  - (2/5)-(6/25)=0.16
- Leverage beer=>diapers
  - (1/5)-(1/25)=0.16

SCORES

- Other scores exist:
  - Conviction
  - zhangs metric
  - …
- https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/

FREQUENT ITEMSET OBJECTIVE

- Objective: limit the number of rules found
  - Given a minimum support threshold *min_sup*
  - Given a minimum confidence threshold *min_conf*
  - Find all association rules with *support* ≥ *min_sup* and *confidence* ≥ *min_conf*

FREQUENT ITEMSET EXTRACTION

NAIVE APPROACH

- Naive approach
  - 1)Generate all possible itemsets (size 1, 2, 3, 4 etc.)
  - 2)Compute their support from the database
- Problem: explosion of possible combinations
  - 1000 items
    - 1000 itemsets of size 1
    - 1000*999/2 itemsets of size 2
    - …
    - 2^{100} combinations

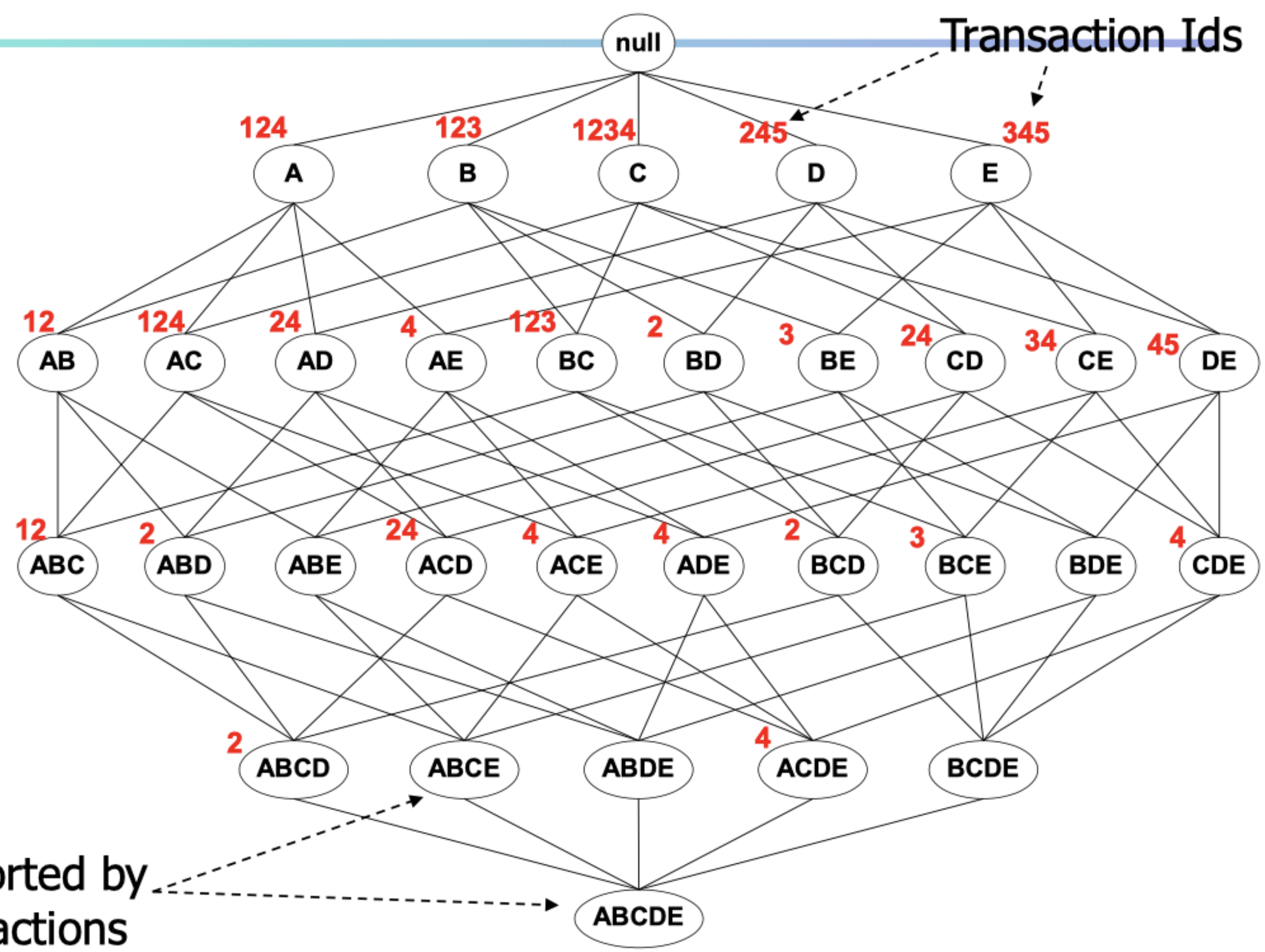
SUPPORT PROPERTY

- Anti-monotonic property of support
  - If X_1 is frequent, then $X_2 \subset X_1$ is frequent
  - If X_1 is not frequent, then $X_2,\ X_1 \subset X_2$ is not frequent
- Computation trick:
  - 1)Find frequent 1-itemsets
  - 2)Find frequent 2-itemsets
    - Among those that contains only frequent 1-itemsets
  - 3) Repeat for all size (or until reaching a threshold)

SUPPORT PROPERTY

Maximal vs Closed Itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Not supported by
any transactions

CLOSED AND MAXIMAL

- We define a **closed** pattern as a frequent pattern (support>threshold) with no sub-pattern of equal support
- We defined a **maximal** pattern as a frequent pattern that has no frequent sub-pattern

SUPPORT PROPERTY

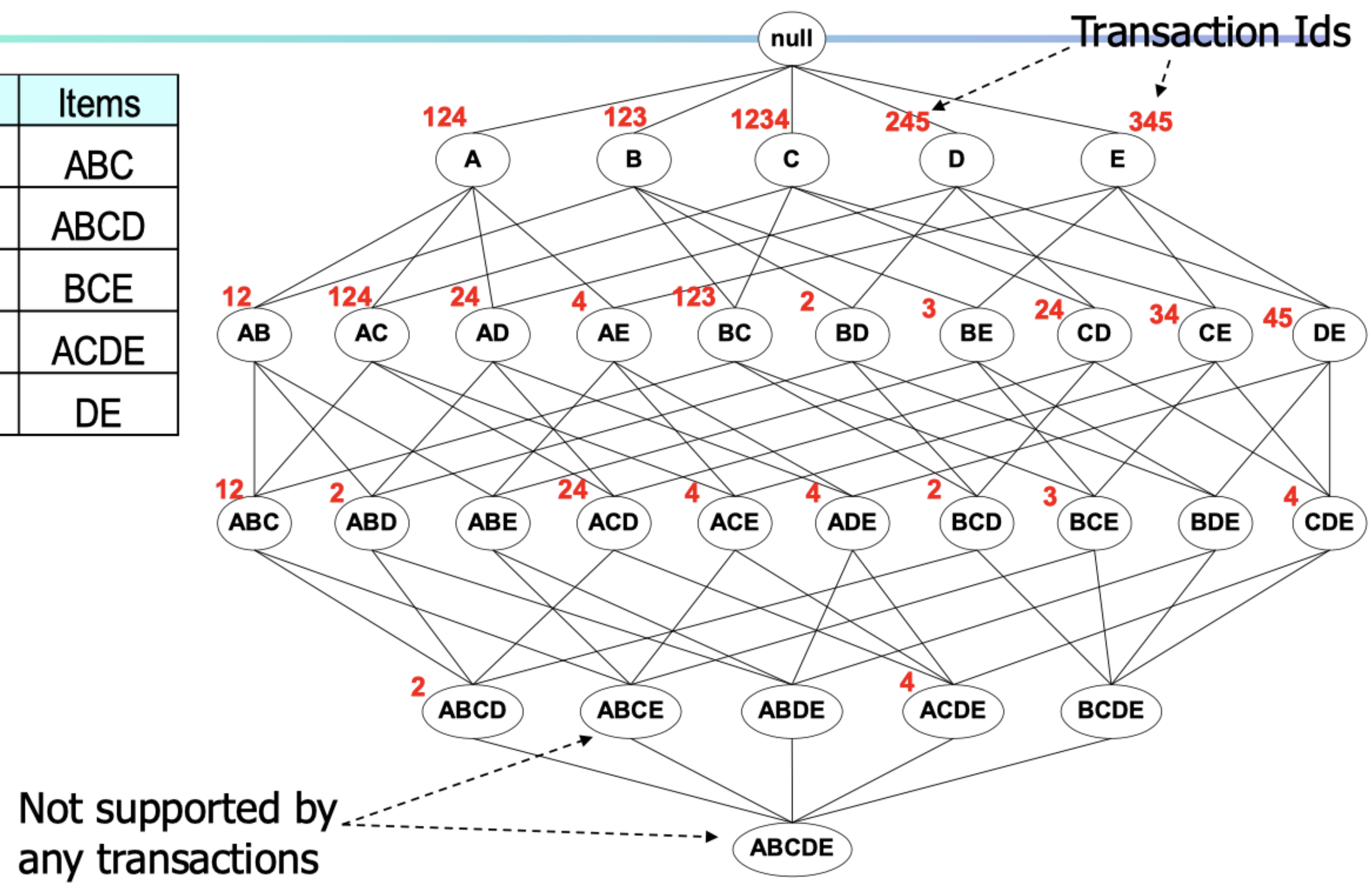
Minimum support = 2

 # Closed = 9

 # Maximal = 4

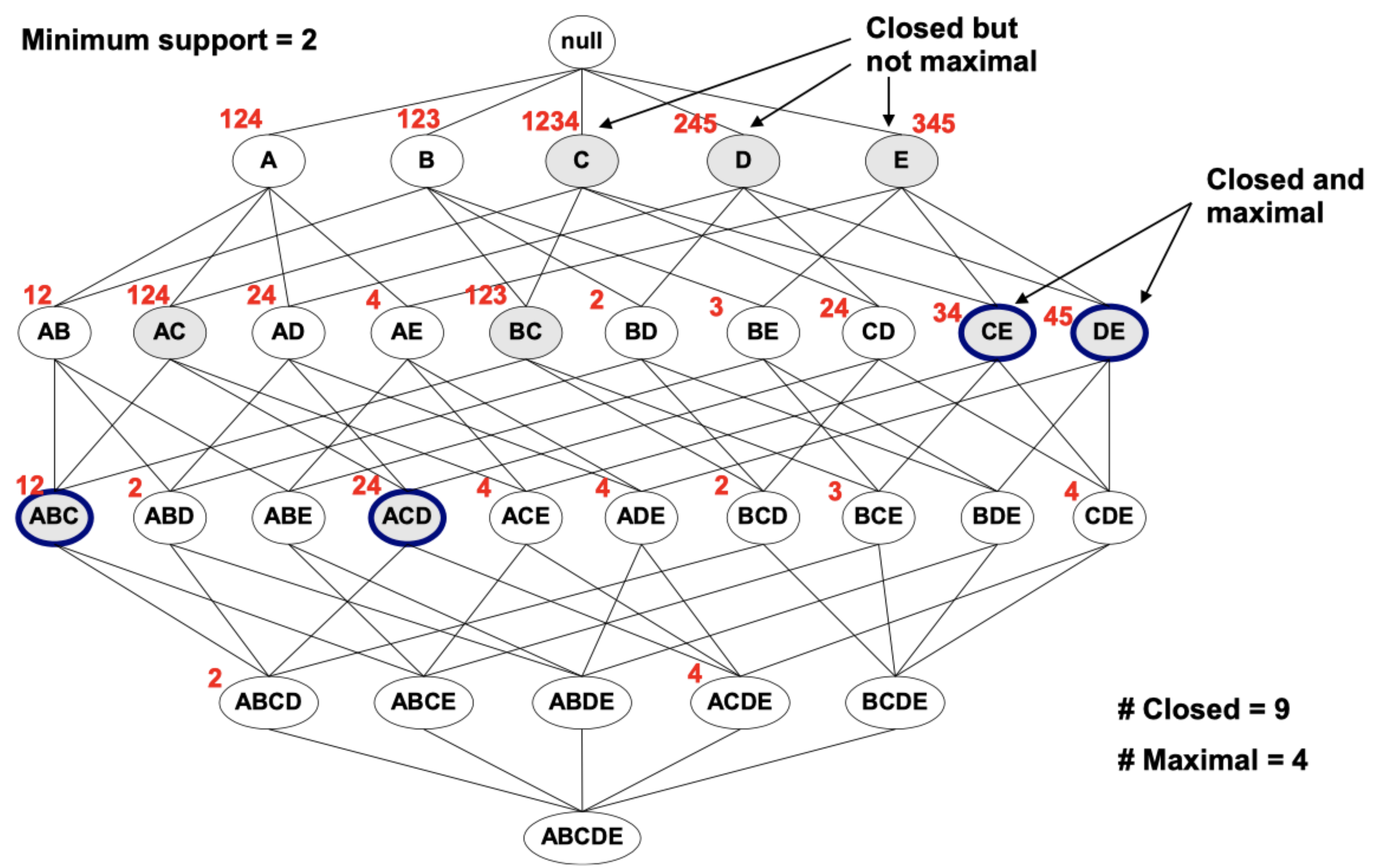
Maximal vs Closed Itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



SUPPORT PROPERTY

Maximal vs Closed Frequent Itemsets



ALGORITHM: APRIORI

APRIORI

```
Apriori(T, ε)
    L₁ ← {frequent 1–itemsets}
    k ← 2
    while Lₖ₋₁ is not empty
        Cₖ ← Apriori_gen(Lₖ₋₁, k)
        for transactions t in T
            Dₜ ← {c in Cₖ : c ⊆ t}
            for candidates c in Dₜ
                count[c] ← count[c] + 1

        Lₖ ← {c in Cₖ : count[c] ≥ ε}
        k ← k + 1

    return Union(Lₖ)
```

```
Apriori_gen(L, k)
    result ← list()
    for all p ∈ L, q ∈ L where p₁ = q₁, p₂ = q₂, ..., pₖ₋₂ = qₖ₋₂ and pₖ₋₁ < qₖ₋₁
        c = p ∪ {qₖ₋₁}
        if u ∈ L for all u ⊆ c where |u| = k–1
            result.add(c)
    return result
```

APRIORI

```
Apriori(T, ε)
    L₁ ← {frequent 1-itemsets}
    k ← 2
    while Lₖ₋₁ is not empty
        Cₖ ← Apriori_gen(Lₖ₋₁, k)
        for transactions t in T
            Dₜ ← {c in Cₖ : c ⊆ t}
            for candidates c in Dₜ
                count[c] ← count[c] + 1

        Lₖ ← {c in Cₖ : count[c] ≥ ε}
        k ← k + 1

    return Union(Lₖ)
```

Start at level 2 of the tree

Treat that level

Generate candidates at current level

Compute support

Check threshold

Go to next level

```
Apriori_gen(L, k)
    result ← list()
    for all p ∈ L, q ∈ L where p₁ = q₁, p₂ = q₂, ..., pₖ₋₂ = qₖ₋₂ and pₖ₋₁ < qₖ₋₁
        c = p ∪ {qₖ₋₁}
        if u ∈ L for all u ⊆ c where |u| = k-1
            result.add(c)
    return result
```

L=frequent sets at previous level

k=current level

Generate unique combinations without repetitions

c=new candidate

Pruning: Also check that all subsets are frequent

APRIORI

- Limits: multiple scans over the dataset
  - Each level
- Many alternatives
  - FP-Growth
  - ECLAT
  - PrefixSpan
  - Distributed approaches (Spark…)
  - …

GOING FURTHER

- Many other works in this domain
  - Sequential Pattern Mining: Take order into account
    - If we first buy a printer, then we will buy ink (and not the opposite)
  - Numeric target value: Find relevant intervals
    - If {a,b}=>z∈[12,25], if {a,c}=z∈[25,32]
  - Subgraph frequent itemsets
    - e.g.: Common substructures across a database of chemical compounds
  - Spatial frequent itemsets
    - Supermarkets close to schools…
  - …

FREQUENT PATTERN=>GRAPHS

- Frequent patterns can represent another way to do data transformation:
  - Extract rules such as item1 => item2
  - Consider this information as an edge
  - Create a network out of it
    - Can be more relevant than a simple distance