# FREQUENT PATTERN MINING—ESSENTIALS

*Hari Sundaram*

hs1@illinois.edu

http://sundaram.cs.illinois.edu

adapted from slides by Jiawei Han and Kevin Chang

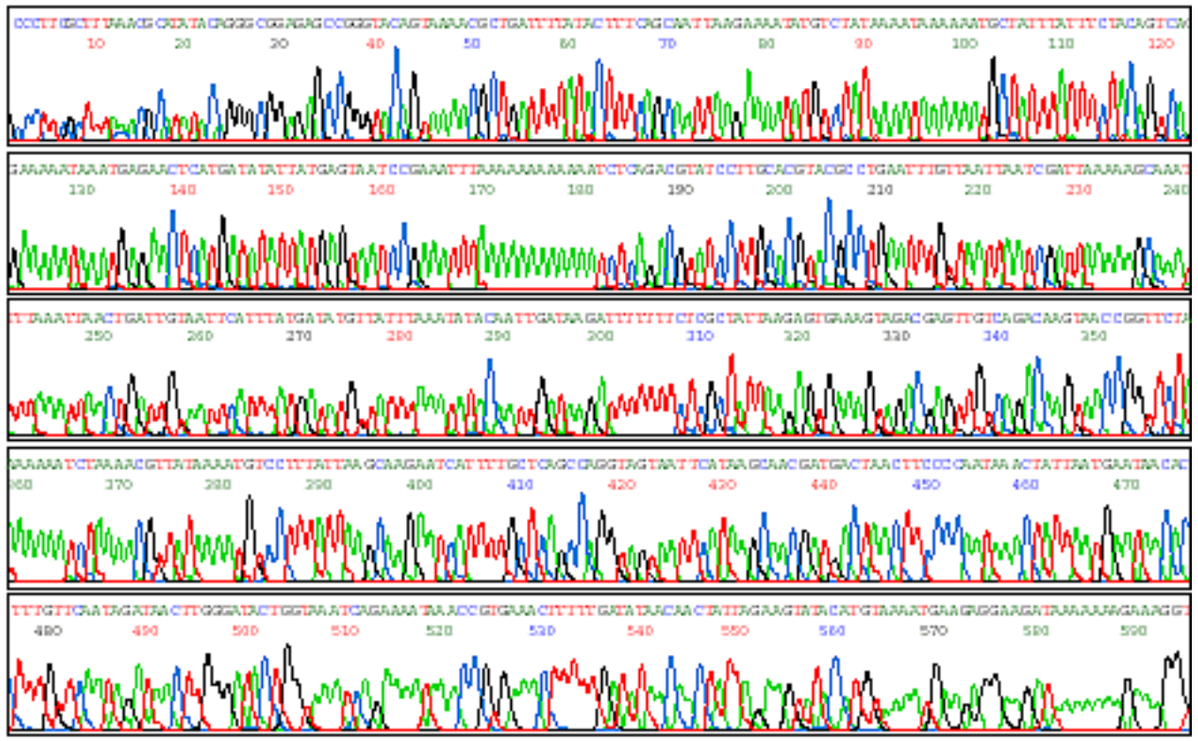
# BASIC CONCEPTS

Methods

Pattern Evaluation

Summary

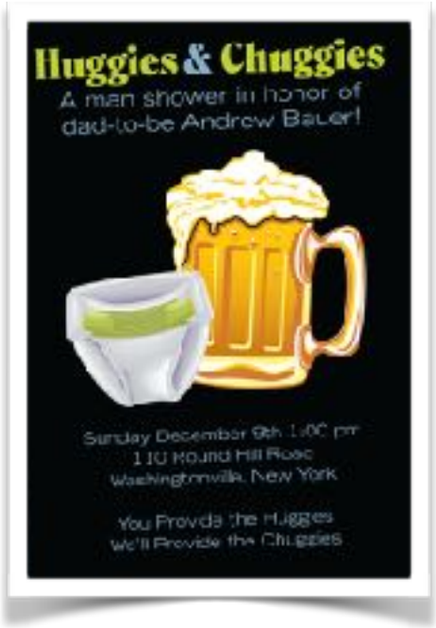
# frequent patterns

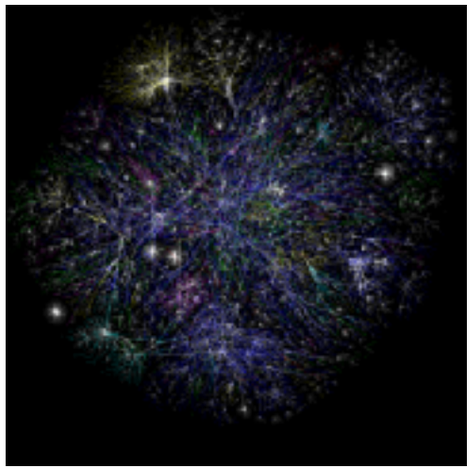


a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set

Agrawal, Rakesh, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." ACM SIGMOD Record 22.2 (1993): 207-216.

What products are often purchased together?



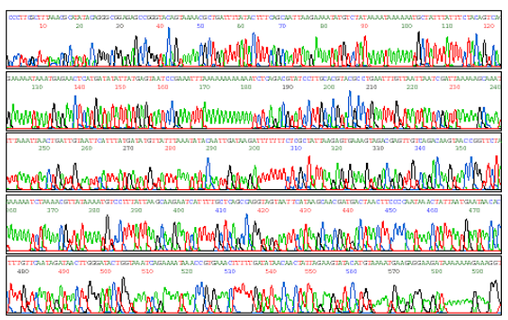



Can we automatically classify web documents?

lots of applications!

# motivation

What kinds of DNA are sensitive to this new drug?



What are the subsequent purchases after buying a PC?

# why is it important?

# frequent mining importance

Foundation for many essential data mining tasks

Cluster analysis

Classification

Semantic data compression: fascicles

Jagadish, H. V., Jason Madar, and Raymond T. Ng. "Semantic compression and pattern extraction with fascicles." VLDB. Vol. 99. 1999.

Data warehousing: iceberg cube and cube-gradient

Pattern analysis in spatiotemporal, multimedia, time-series, and stream data

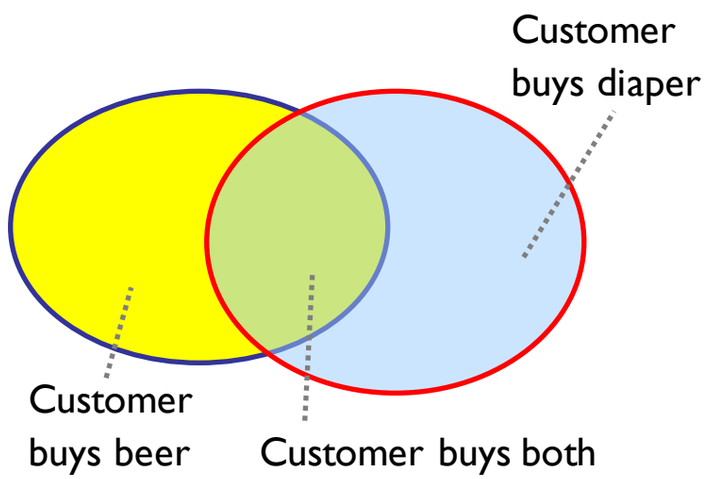
Broad applications!

Association, correlation, and causality analysis

Sequential, structural (e.g., sub-graph) patterns

itemset: A set of one or more items

| Tid | Items bought |
|---|---|
| 10 | {Beer, Nuts, Diaper} |
| 20 | Beer, Coffee, Diaper |
| 30 | Beer, Diaper, Eggs |
| 40 | Nuts, Eggs, Milk |
| 50 | Nuts, Coffee, Diaper, Eggs, Milk |




# BASIC IDEAS: FREQUENT PATTERNS

k-itemset $X = \{x_1, \ldots, x_k\}$

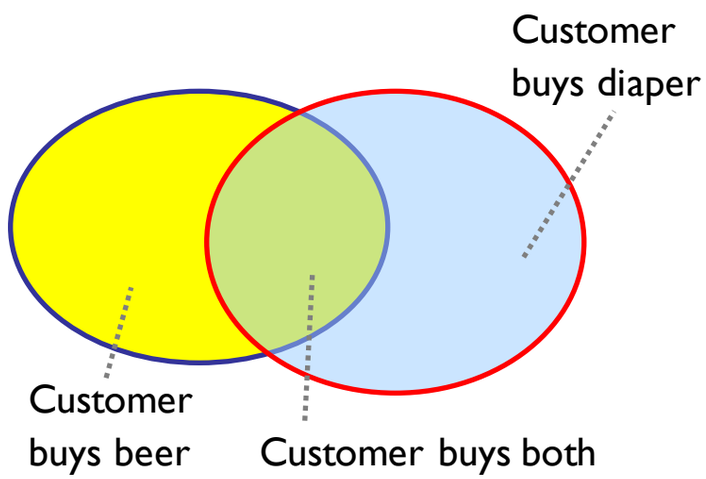
(absolute) support, or, support count of X: Frequency or occurrence of an itemset X

(relative) support, s, is the fraction of transactions that contains X (i.e., the probability that a transaction contains X)

An itemset X is frequent if X's support is no less than a **minsup** threshold

itemset: A set of one or more items

| Tid | Items bought |
|---|---|
| 10 | Beer, Nuts, Diaper |
| 20 | Beer, Coffee, Diaper |
| 30 | Beer, Diaper, Eggs |
| 40 | Nuts, Eggs, Milk |
| 50 | Nuts, Coffee, Diaper, Eggs, Milk |




# BASIC IDEAS: ASSOCIATION RULES

Find all the rules X→Y with minimum support and confidence

support, s, probability that a transaction contains X ∪ Y

confidence, c, conditional probability that a transaction having X also contains Y

Let **minsup** = 50%, **minconf** = 50%

Frequent Patterns: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

Association rules: (many more!)

Beer→Diaper (60%, 100%)

Diaper→Beer (60%, 75%)

joint probability

$$P(X \cup Y) \neq P(X \vee B)$$

X,Y are set elements, not events

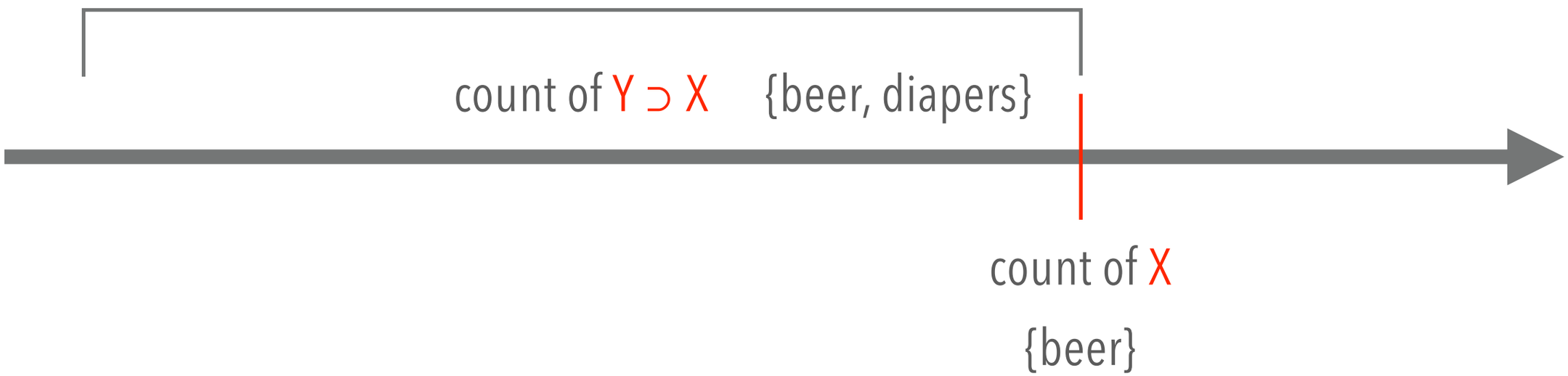
closed

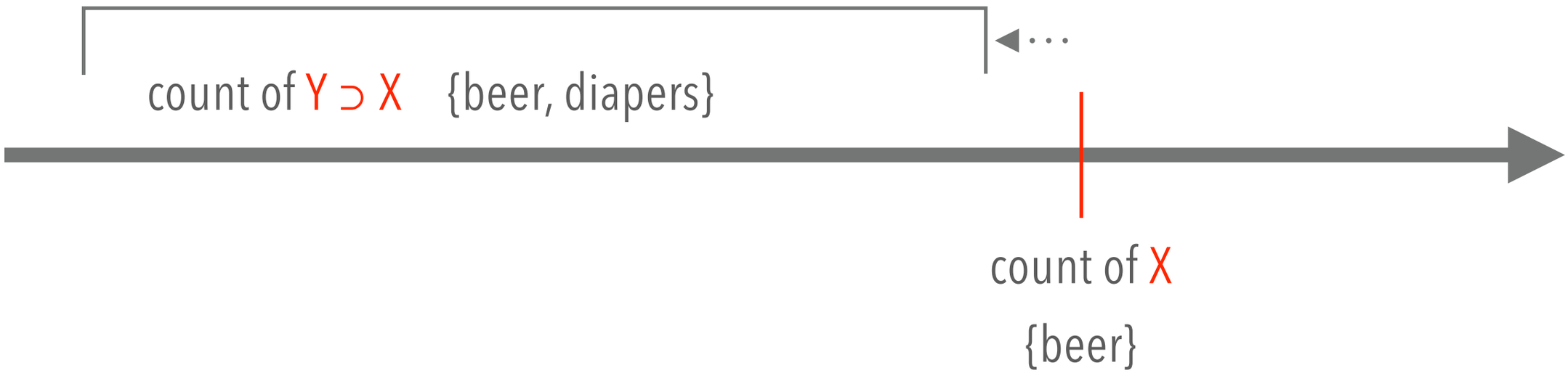
# A long pattern $\{a_1, \ldots, a_{100}\}$ contains contains a combinatorial number of sub-patterns:

$2^{100} - 1 \cong 10^{30.1}$ sub-patterns!

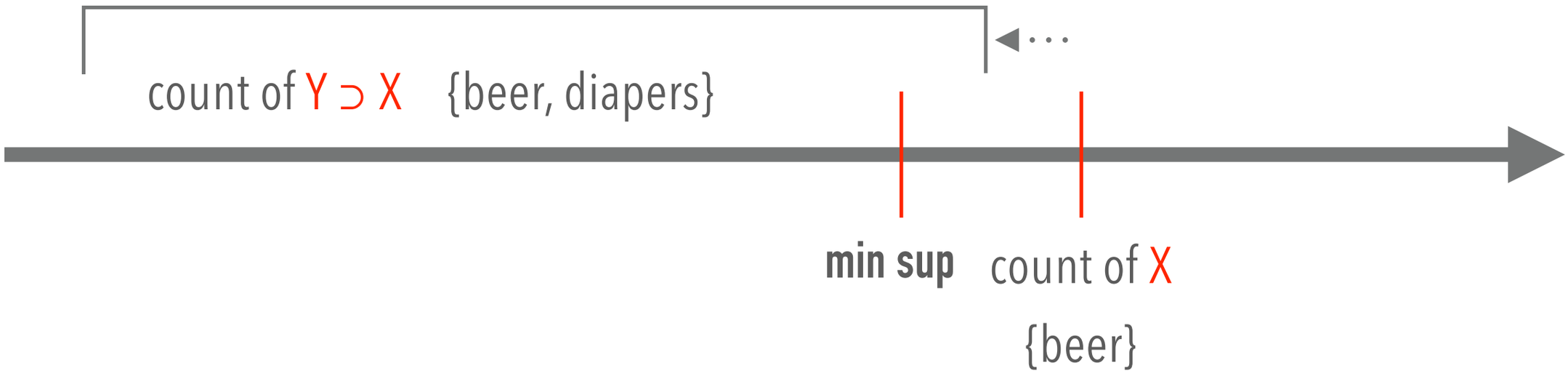
max patterns

the count of the super set Y is always less or equal to the set count X!



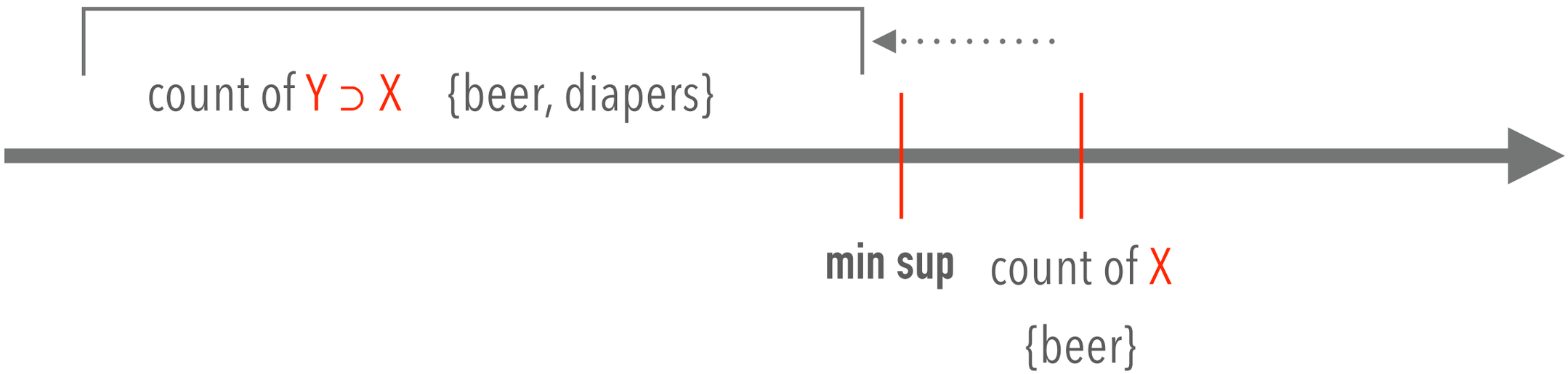



# An itemset X is closed in dataset D if there exists no proper super-pattern Y ⊃ X, with the same support as X in D

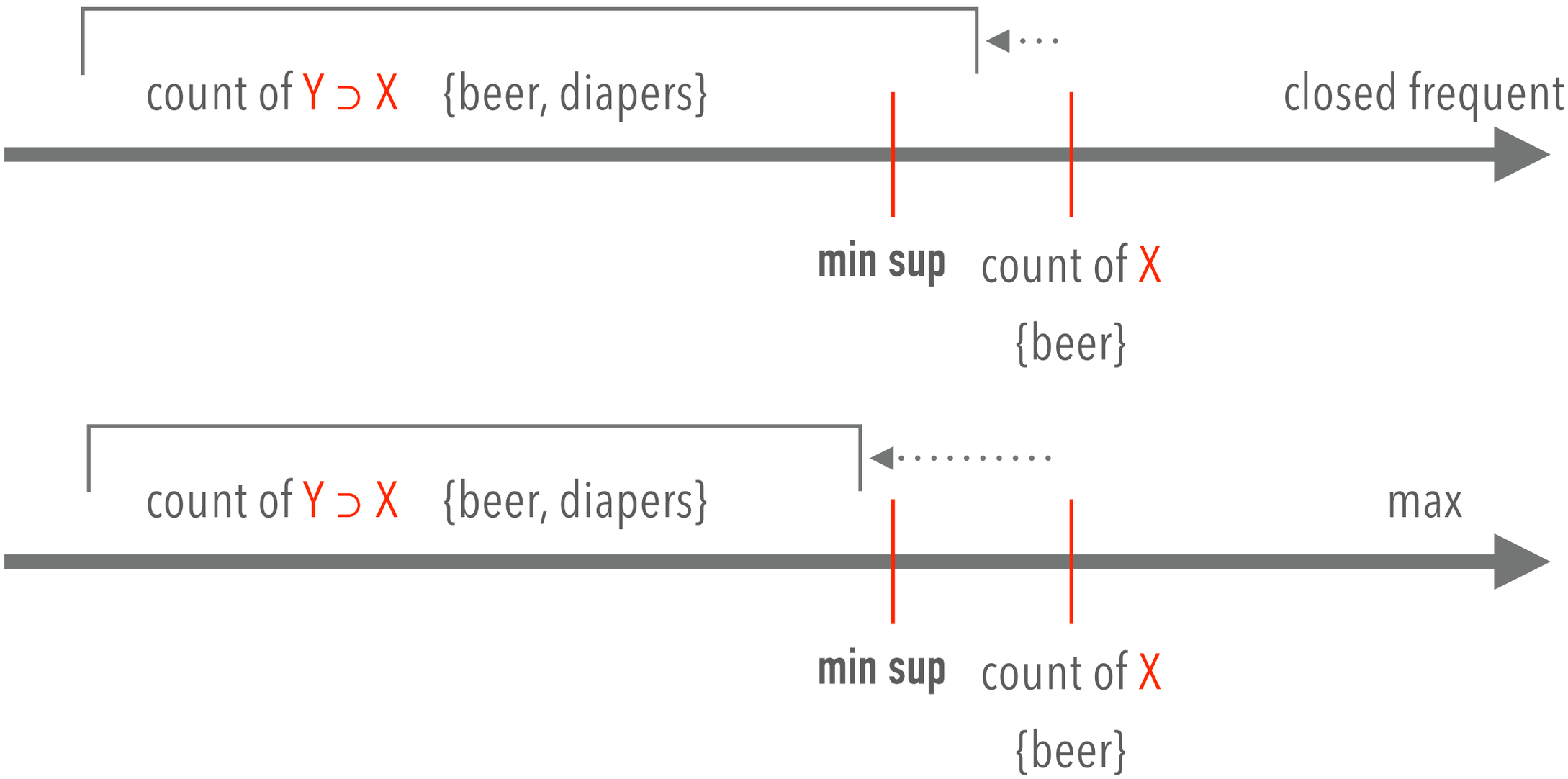



# An itemset X is a closed frequent itemset in set D if X is both closed and frequent in D

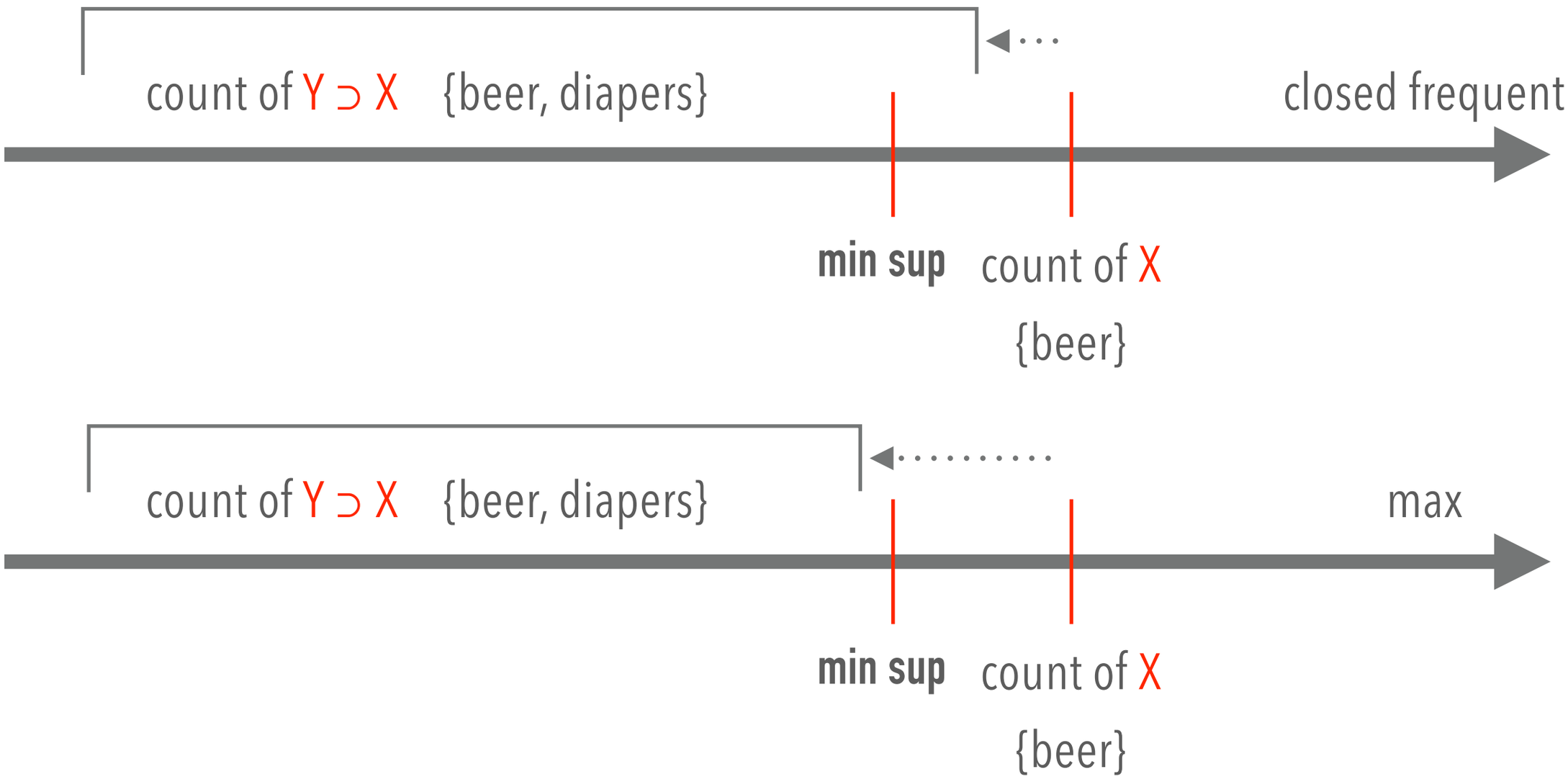
The definition does **not** say anything about Y




# An itemset X is a maximal frequent itemset (or max-itemset) in a data set D if X is frequent, and there exists no super-itemset Y such that Y ⊃ X and Y is frequent in D.




what is a key difference between the set of all **max patterns** and the set of all **closed frequent** patterns?




You can use the set of closed frequent patterns to enumerate all frequent item sets

assume the following four transactions:

{**beer, nuts, diapers**},
{beer, nuts, diapers},
{beer, nuts}, {beer, nuts}

what are closed item sets?

**min_sup=2**

**Exercise!**

$\{a_1, \ldots, a_{100}\}, \{a_1, \ldots, a_{50}\}$

the DB contains only these two transactions

**min sup=1**

**Exercise!**

$\{a_1, \ldots, a_{100}\}, \{a_1, \ldots, a_{50}\}$

min sup=1

what is the set of closed patterns?

$\{a_1, \ldots, a_{100}\}$: 1, $\{a_1, \ldots, a_{50}\}$: 2

**Exercise!**

$\{a_1, \ldots, a_{100}\}, \{a_1, \ldots, a_{50}\}$

min sup=1

what is the set of max patterns?

$\{a_1, \ldots, a_{100}\}$: 1

**Exercise!**

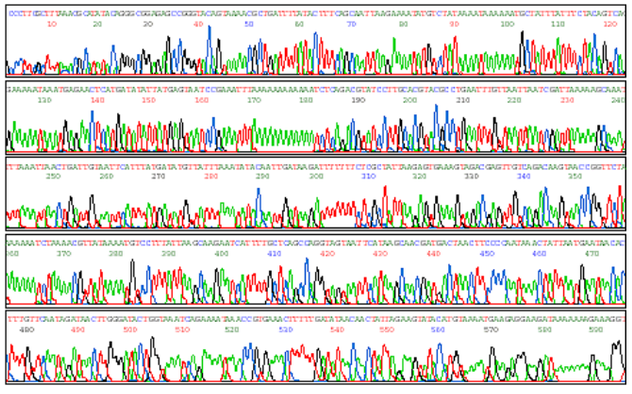
$\{a_1, \ldots, a_{100}\}, \{a_1, \ldots, a_{50}\}$

min sup=1

how many patterns in all?

$\{a_1\}, \{a_2\}, \{a_1, a_2\}, \ldots, \{a_1, \ldots, a_{100}\}$

$2^{100} - 1$



a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set

# BASIC CONCEPTS

itemset, support, frequent

confidence

closed vs. max-pattern


# SUMMARY

Methods

Pattern Evaluation

Summary

Apriori

FPGrowth

ECLAT

Max & Closed

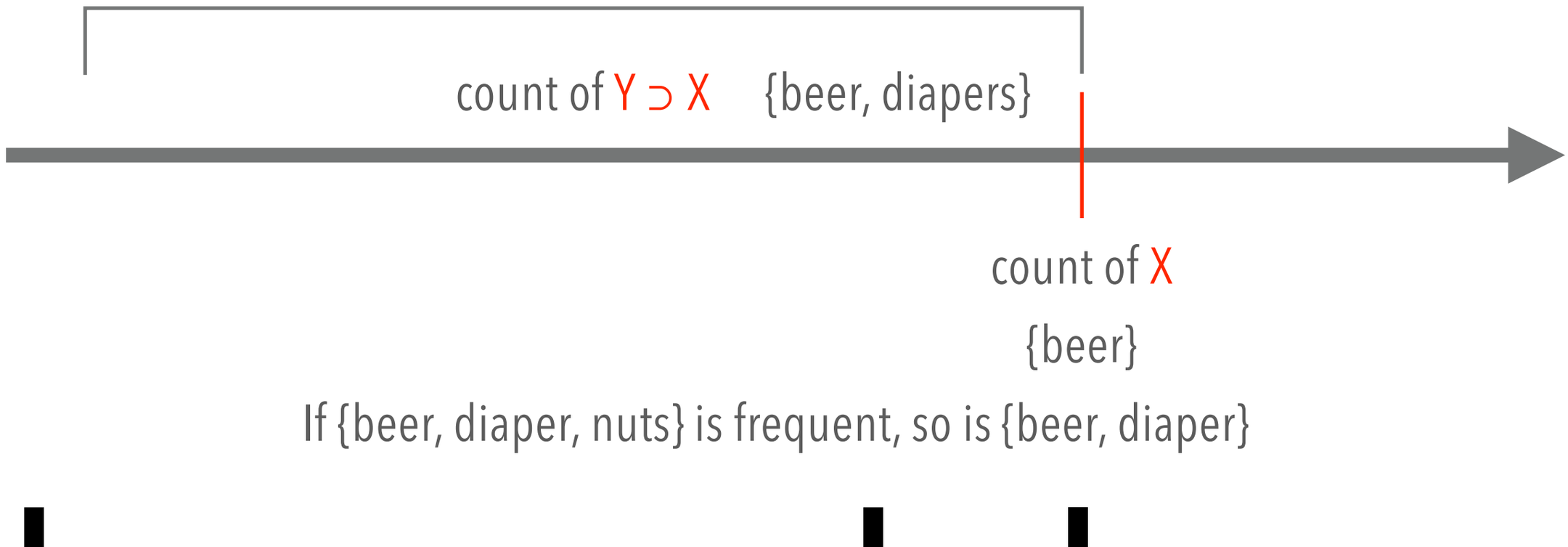
# FREQUENT ITEMSET METHODS

Basic Concepts

Pattern Evaluation

Summary

the count of a set X is always greater than or equal to the superset Y count!




If {beer, diaper, nuts} is frequent, so is {beer, diaper}

# downward closure property

Any subset of a frequent itemset must be frequent

Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94), Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 487-499.

# Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated or tested!

Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94), Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 487-499.

# Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated or tested!

1. Initially, scan DB once to get frequent 1-itemset
2. Generate length (k+1) candidate itemsets from length k frequent itemsets
3. Test the candidates against DB
4. Terminate when no frequent or candidate set can be generated

| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Scan $D$ for count of each candidate

$C_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Compare candidate support count with minimum support count

$L_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Generate $C_2$ candidates from $L_1$ →

$C_2$

| Itemset |
|---|
| {I1, I2} |
| {I1, I3} |
| {I1, I4} |
| {I1, I5} |
| {I2, I3} |
| {I2, I4} |
| {I2, I5} |
| {I3, I4} |
| {I3, I5} |
| {I4, I5} |

Scan $D$ for count of each candidate →

$C_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I4} | 1 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |
| {I3, I4} | 0 |
| {I3, I5} | 1 |
| {I4, I5} | 0 |

Compare candidate support count with minimum support count →

$L_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |

$L_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |

| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Generate $C_3$ candidates from $L_2$ →

$C_3$

| Itemset |
|---|
| {I1, I2, I3} |
| {I1, I2, I5} |

→ Scan $D$ for count of each candidate →

$C_3$

| Itemset | Sup. count |
|---|---|
| {I1, I2, I3} | 2 |
| {I1, I2, I5} | 2 |

→ Compare candidate support count with minimum support count →

$L_3$

| Itemset | Sup. count |
|---|---|
| {I1, I2, I3} | 2 |
| {I1, I2, I5} | 2 |

**Algorithm: Apriori.** Find frequent itemsets using an iterative level-wise approach based on candidate generation.

**Input:**

- $D$, a database of transactions;
- *min_sup*, the minimum support count threshold.

**Output:** $L$, frequent itemsets in $D$.

**Method:**

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ φ; k++) {
(3)       Ck = apriori_gen(Lk−1);
(4)       for each transaction t ∈ D { // scan D for counts
(5)           Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)           for each candidate c ∈ Ct
(7)               c.count++;
(8)       }
(9)       Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;

procedure apriori_gen(Lk−1:frequent (k − 1)-itemsets)
(1)   for each itemset l1 ∈ Lk−1
(2)       for each itemset l2 ∈ Lk−1
(3)           if (l1[1] = l2[1]) ∧ (l1[2] = l2[2])
                  ∧... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)               c = l1 ⋈ l2; // join step: generate candidates
(5)               if has_infrequent_subset(c, Lk−1) then
(6)                   delete c; // prune step: remove unfruitful candidate
(7)               else add c to Ck;
(8)           }
(9)   return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
          Lk−1: frequent (k − 1)-itemsets); // use prior knowledge
(1)   for each (k − 1)-subset s of c
(2)       if s ∉ Lk−1 then
(3)           return TRUE;
(4)   return FALSE;
```

# APRIORI IMPLEMENTATION

## How to generate candidates?

### Step 1: self-joining $L_k$

### Step 2: pruning

## Example of Candidate-generation

$L_3$={abc, abd, acd, ace, bcd}

Self-joining: $L_3$*$L_3$

{abcd} from {abc} and {abd}

{acde} from {acd} and {ace}

Pruning:

{acde} is removed because {ade} is not in $L_3$

$C_4$ = {abcd}

**Algorithm: Apriori.** Find frequent itemsets using an iterative level-wise approach based on candidate generation.

**Input:**

- $D$, a database of transactions;
- *min_sup*, the minimum support count threshold.

**Output:** $L$, frequent itemsets in $D$.

**Method:**

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ φ; k++) {
(3)       Ck = apriori_gen(Lk−1);
(4)       for each transaction t ∈ D { // scan D for counts
(5)           Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)           for each candidate c ∈ Ct
(7)               c.count++;
(8)       }
(9)       Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;

procedure apriori_gen(Lk−1:frequent (k − 1)-itemsets)
(1)   for each itemset l1 ∈ Lk−1
(2)       for each itemset l2 ∈ Lk−1
(3)           if (l1[1] = l2[1]) ∧ (l1[2] = l2[2])
                  ∧... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)               c = l1 ⋈ l2; // join step: generate candidates
(5)               if has_infrequent_subset(c, Lk−1) then
(6)                   delete c; // prune step: remove unfruitful candidate
(7)               else add c to Ck;
(8)           }
(9)   return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
          Lk−1: frequent (k − 1)-itemsets); // use prior knowledge
(1)   for each (k − 1)-subset s of c
(2)       if s ∉ Lk−1 then
(3)           return TRUE;
(4)   return FALSE;
```

$L_3$={abc, abd, acd, ace, bcd}

Self-joining: $L_3$*$L_3$

{abcd} from {abc} and {abd}

{acde} from {acd} and {ace}

{acde} is removed because {ade} is not in $L_3$

$C_4$ = {abcd}

# apriori is inefficient!

# challenges

Multiple scans of transaction database — reduce passes

Huge number of candidates — shrink

Tedious workload of support counting for candidates — make it easy to keep count

Create hash table $H_2$ using hash function $h(x, y) = ((order\ of\ x) \times 10 + (order\ of\ y))\ mod\ 7$

$H_2$

| bucket address | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| bucket count | 2 | 2 | 4 | 2 | 2 | 4 | 4 |
| bucket contents | {I1, I4}<br>{I3, I5} | {I1, I5}<br>{I1, I5} | {I2, I3}<br>{I2, I3}<br>{I2, I3}<br>{I2, I3} | {I2, I4}<br>{I2, I4} | {I2, I5}<br>{I2, I5} | {I1, I2}<br>{I1, I2}<br>{I1, I2}<br>{I1, I2} | {I1, I3}<br>{I1, I3}<br>{I1, I3}<br>{I1, I3} |

# hashing

A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent

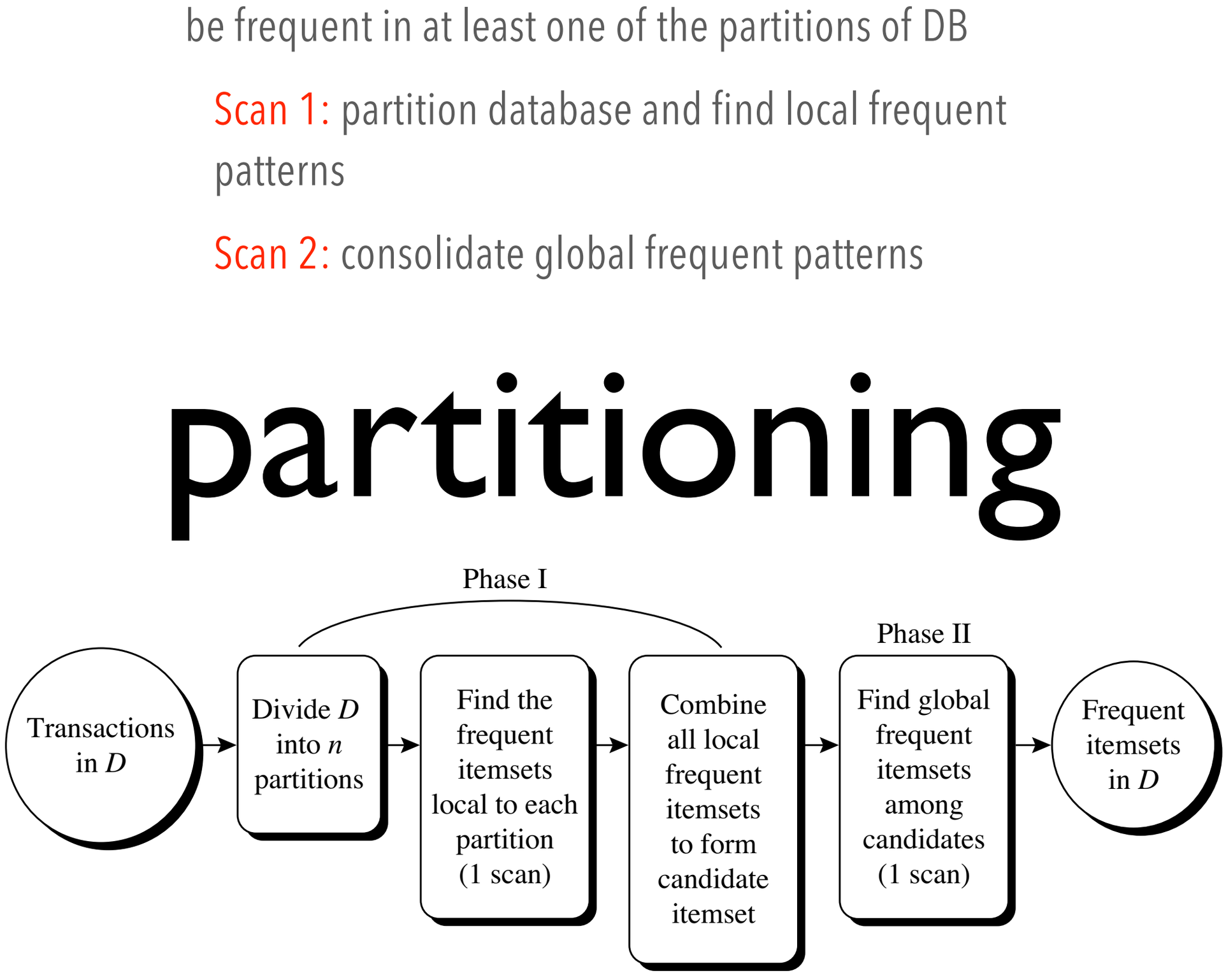
Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. 1995. An effective hash-based algorithm for mining association rules. In Proceedings of the 1995 ACM SIGMOD international conference on Management of data (SIGMOD '95), Michael Carey and Donovan Schneider (Eds.). ACM, New York, NY, USA, 175-186.

Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB

Scan 1: partition database and find local frequent patterns

Scan 2: consolidate global frequent patterns

# partitioning



# sampling

Hannu Toivonen. 1996. Sampling Large Databases for Association Rules. In Proceedings of the 22th International Conference on Very Large Data Bases (VLDB '96), T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 134-145.

Apriori

FPGrowth

ECLAT

Max & Closed

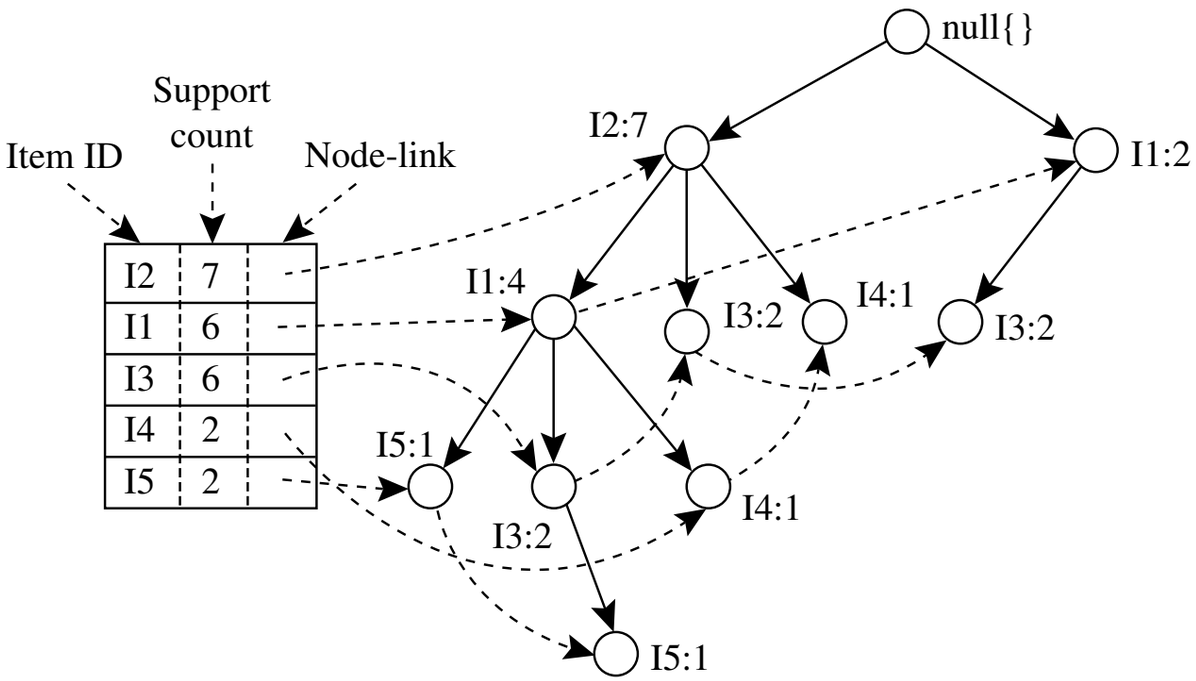
# FREQUENT ITEMSET METHODS

Basic Concepts

Pattern Evaluation

Summary

# MINING FREQUENT PATTERNS




Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data (SIGMOD '00). ACM, New York, NY, USA, 1-12.

Prof. Han's most cited paper! ~**7,800** citations

- Bottlenecks of the Apriori approach
  - Breadth-first (i.e., level-wise) search
  - Candidate generation and test (Often generates a huge number of candidates)
- The FPGrowth Approach
  - Depth-first search
  - Avoid explicit candidate generation
- Major philosophy: Grow long patterns from short ones using local frequent items only
  - "abc" is a frequent pattern
  - Get all transactions having "abc", i.e., project DB on abc: DB|abc
  - "d" is a local frequent item in DB|abc abcd is a frequent pattern

Branch

| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Branch

| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

**min sup=2**

most frequent

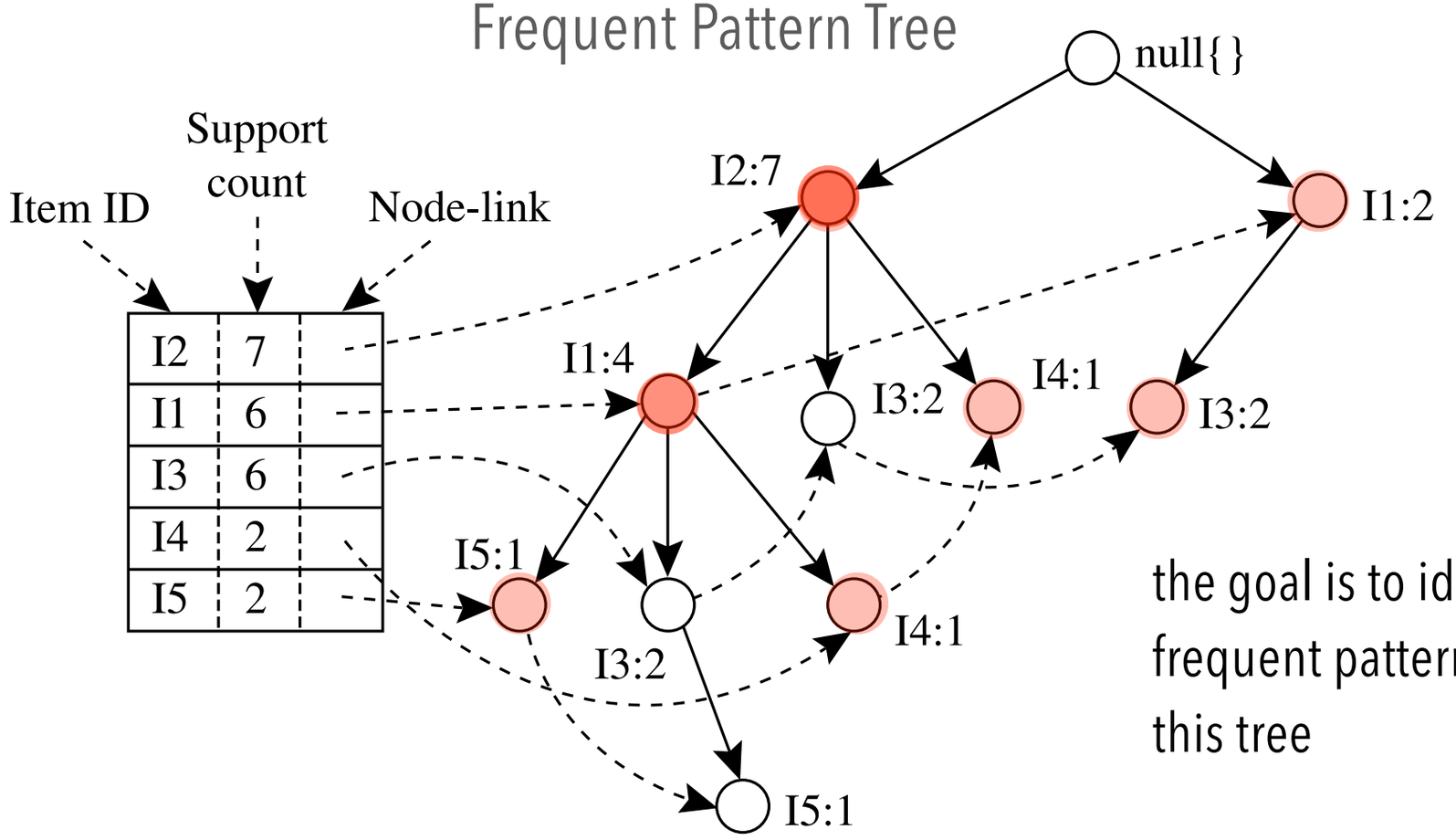
| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Branch

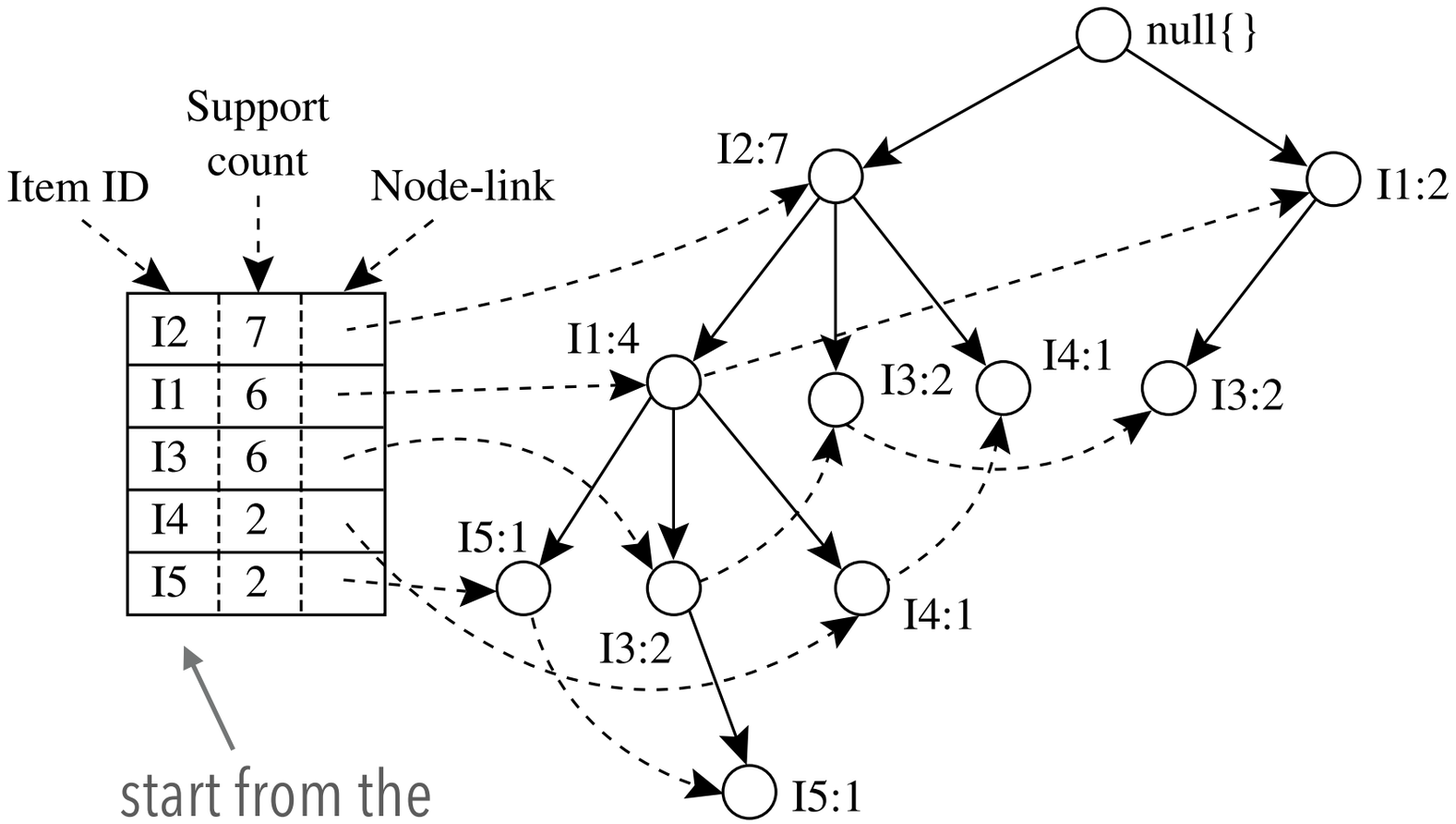
| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

## Frequent Pattern Tree




the goal is to identify frequent patterns from this tree

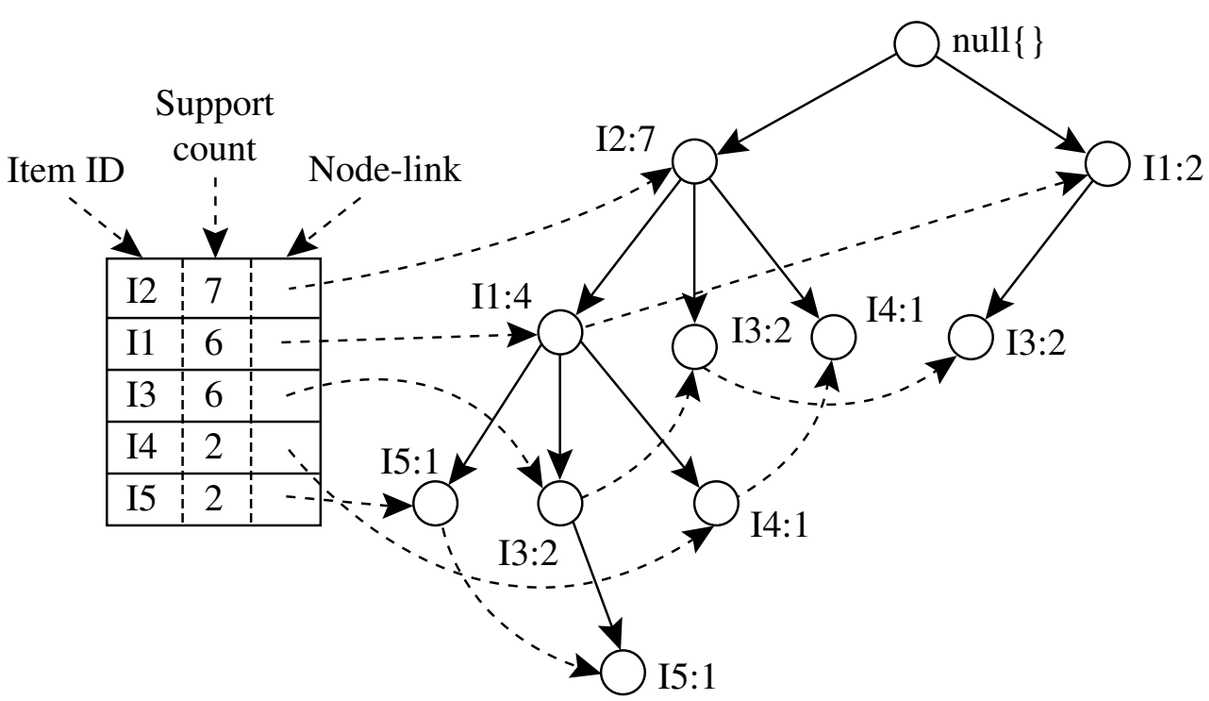


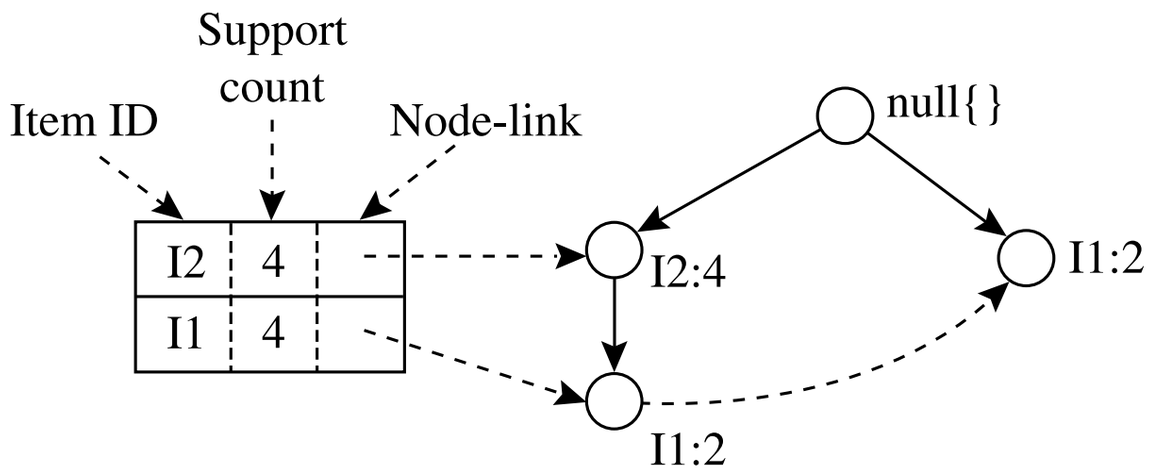

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

min sup=2

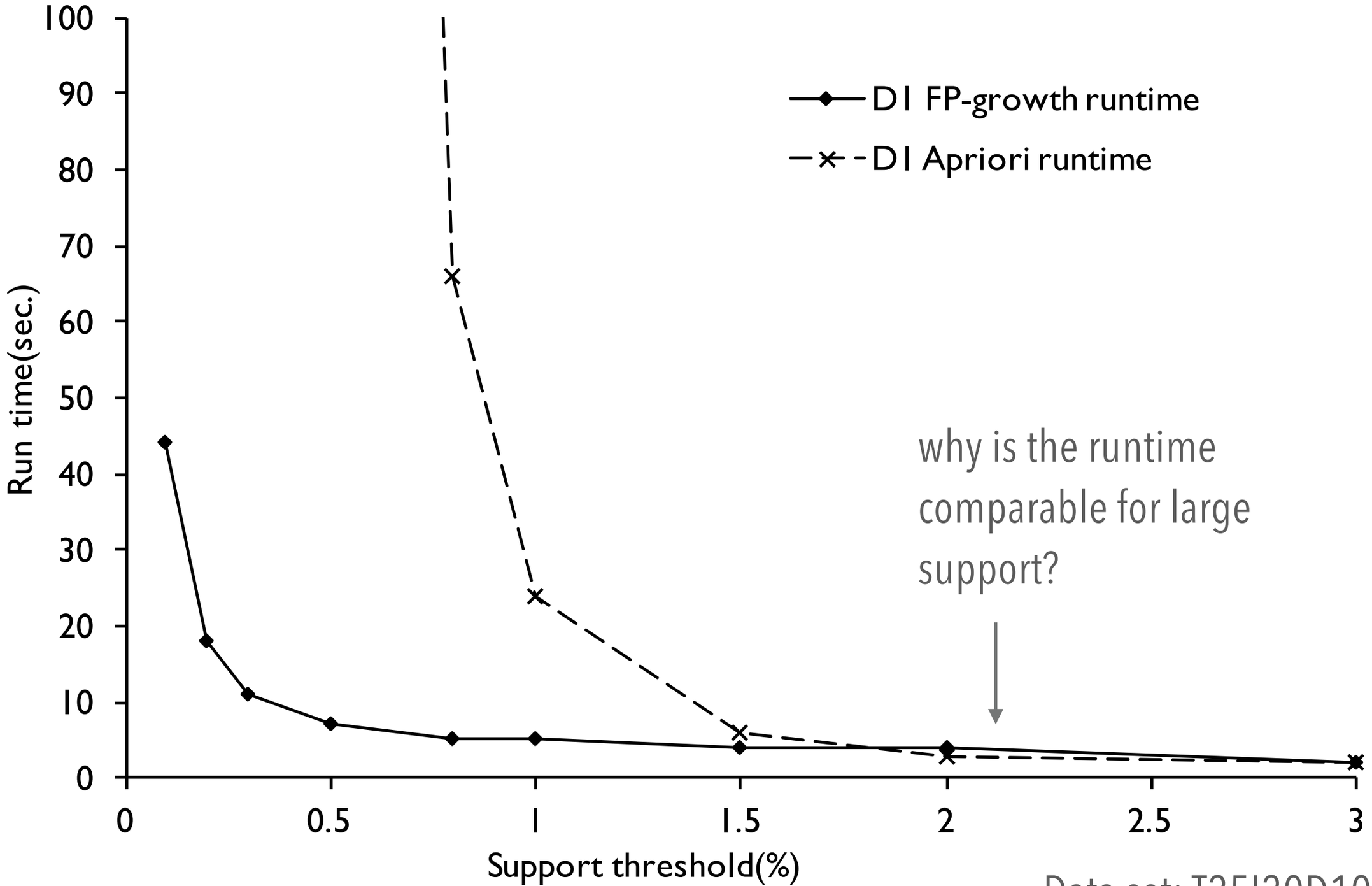
| *Item* | *Conditional Pattern Base* | *Conditional FP-tree* | *Frequent Patterns Generated* |
|---|---|---|---|
| I5 | {{I2, I1: 1}, {I2, I1, I3: 1}} | ⟨I2: 2, I1: 2⟩ | {I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2} |
| I4 | {{I2, I1: 1}, {I2: 1}} | ⟨I2: 2⟩ | {I2, I4: 2} |
| I3 | {{I2, I1: 2}, {I2: 2}, {I1: 2}} | ⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩ | {I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2} |
| I1 | {{I2: 4}} | ⟨I2: 4⟩ | {I2, I1: 4} |

| *Item* | *Conditional Pattern Base* | *Conditional FP-tree* | *Frequent Patterns Generated* |
|---|---|---|---|
| I5 | {{I2, I1: 1}, {I2, I1, I3: 1}} | ⟨I2: 2, I1: 2⟩ | {I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2} |
| I4 | {{I2, I1: 1}, {I2: 1}} | ⟨I2: 2⟩ | {I2, I4: 2} |
| I3 | {{I2, I1: 2}, {I2: 2}, {I1: 2}} | ⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩ | {I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2} |
| I1 | {{I2: 4}} | ⟨I2: 4⟩ | {I2, I1: 4} |



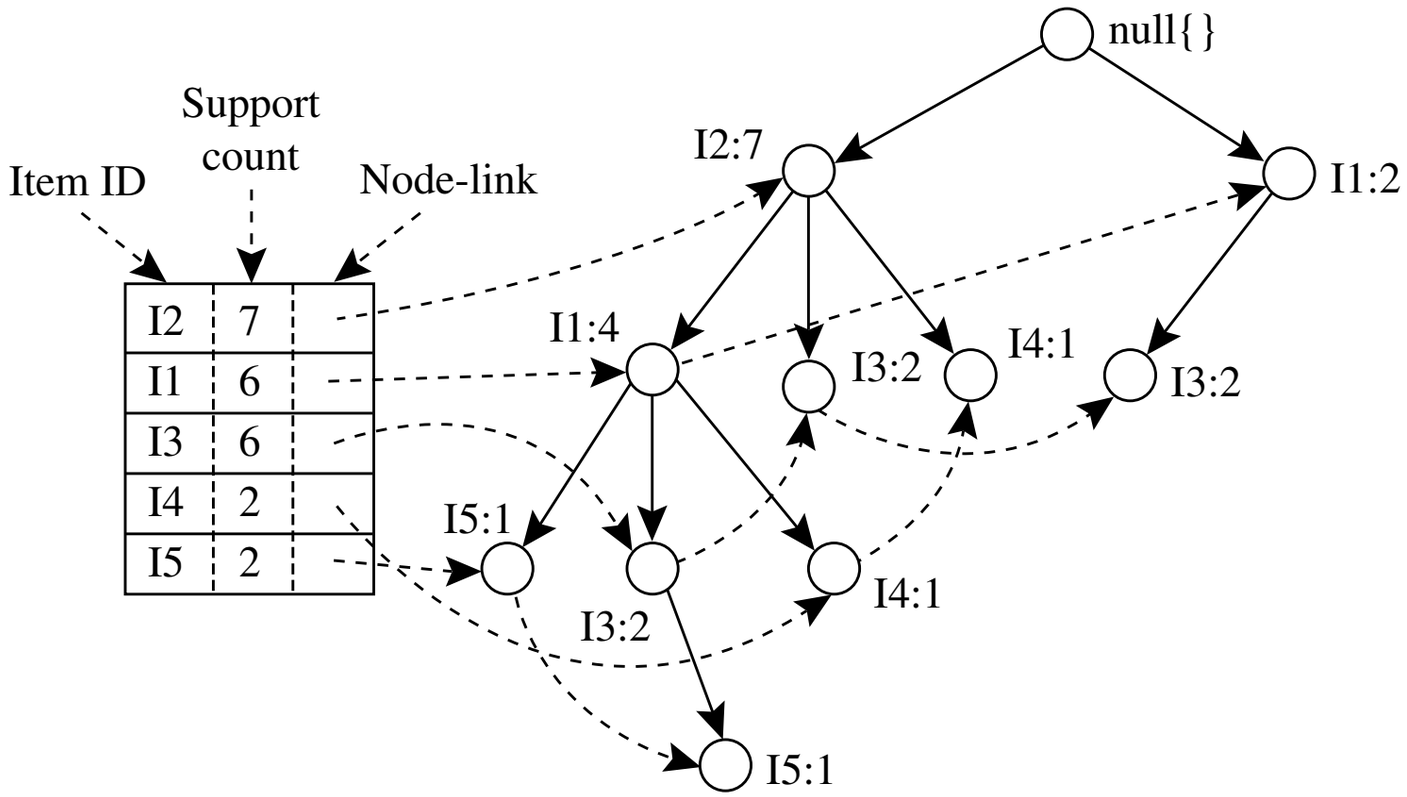
# FP-Growth vs. Apriori: Scalability With the Support Threshold




why is the runtime comparable for large support?

Data set: T25I20D10K

# FP-GROWTH SUMMARY




Idea: Frequent pattern growth

Recursively grow frequent patterns by pattern and database partition

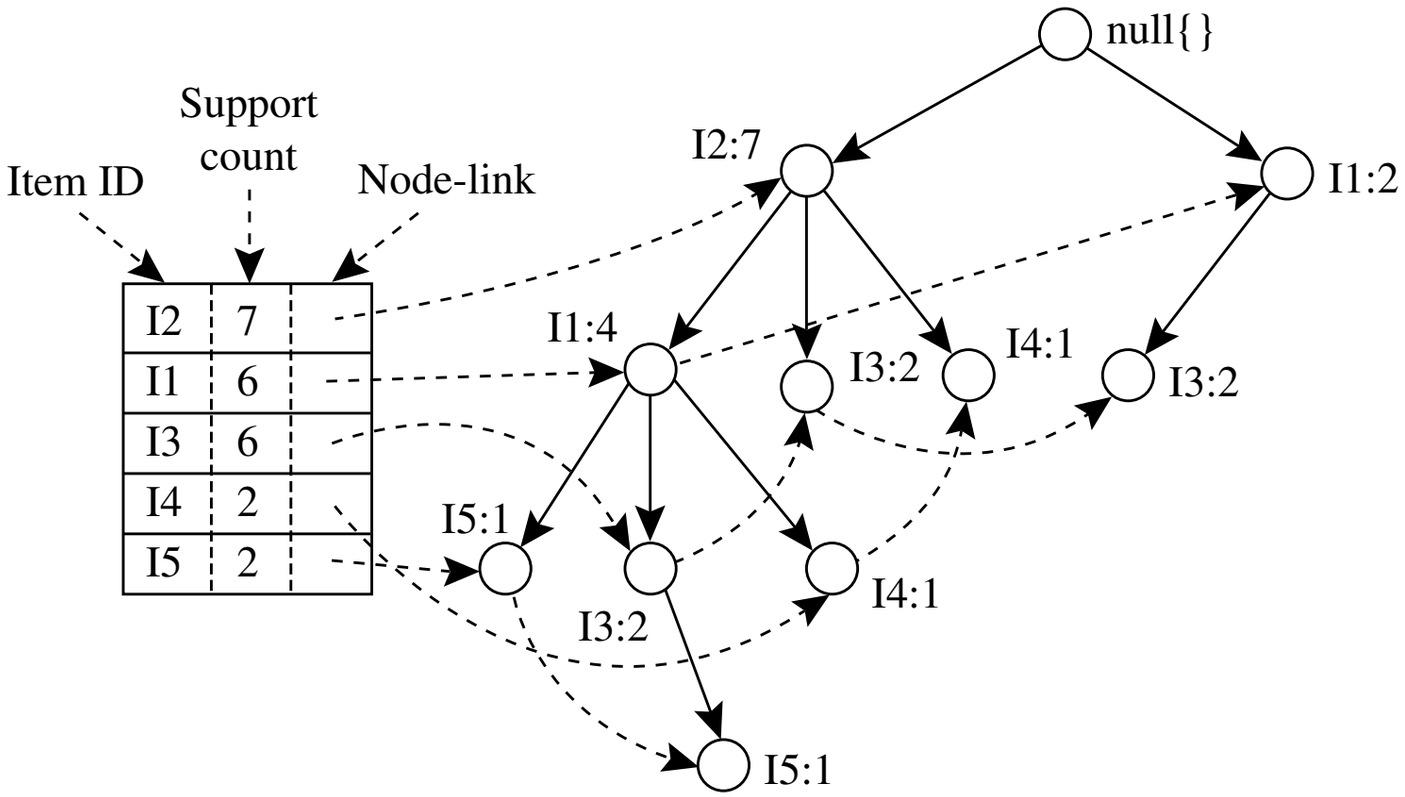
Method

For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree

Repeat the process on each newly created conditional FP-tree

Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

# FP-TREE BENEFITS



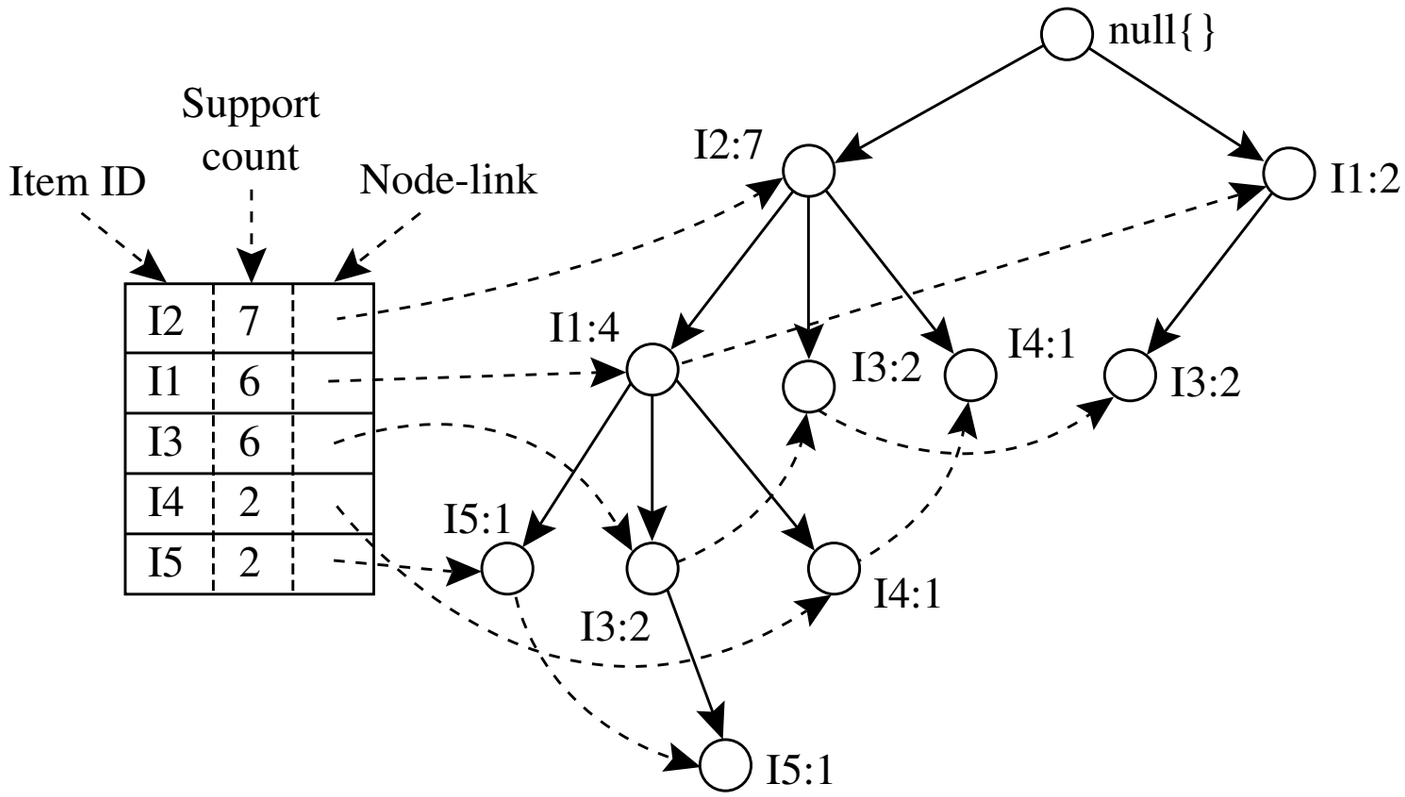

Divide-and-conquer:

- Decompose both the mining task and DB according to the frequent patterns obtained so far
- Lead to focused search of smaller databases

Other factors

- No candidate generation, no candidate test
- Compressed database: FP-tree structure
- No repeated scan of entire database
- Basic operations: counting local frequent items and building sub FP-tree, no pattern search and matching

# FP-TREE BENEFITS




## Completeness

Preserves complete information for frequent pattern mining

Never breaks a long pattern of any transaction

## Compactness

Reduce irrelevant information—infrequent items are gone

Items in frequency descending order: the more frequently occurring pattern, the more likely it will be shared

Is never larger than the original database (not counting node-links and the count field)

# what if the tree cannot fit in main memory?

partition!

Apriori

FPGrowth

ECLAT

Max & Closed

# FREQUENT ITEMSET METHODS

Basic Concepts

Pattern Evaluation

Summary

| *itemset* | *TID_set* |
|---|---|
| I1 | {T100, T400, T500, T700, T800, T900} |
| I2 | {T100, T200, T300, T400, T600, T800, T900} |
| I3 | {T300, T500, T600, T700, T800, T900} |
| I4 | {T200, T400} |
| I5 | {T100, T800} |

| *itemset* | *TID_set* |
|---|---|
| {I1, I2} | {T100, T400, T800, T900} |
| {I1, I3} | {T500, T700, T800, T900} |
| {I1, I4} | {T400} |
| {I1, I5} | {T100, T800} |
| {I2, I3} | {T300, T600, T800, T900} |
| {I2, I4} | {T200, T400} |
| {I2, I5} | {T100, T800} |
| {I3, I5} | {T800} |

Zaki, M.J., Parthasarathy, S., Ogihara, M. and Li, W., 1997, August. New Algorithms for Fast Discovery of Association Rules. In KDD (Vol. 97, pp. 283-286).

# ECLAT: VERTICAL DATA MINING

Vertical format: $t(AB) = \{T_{11}, T_{25}, \ldots\}$

**tid-list**: list of transaction ids containing an itemset

Deriving frequent patterns based on vertical intersections

$t(X) = t(Y)$: X and Y always happen together

$t(X) \subset t(Y)$: transaction having X always has Y

Using diffset to accelerate mining

Only keep track of differences of **tids**

$t(X) = \{T_1, T_2, T_3\}$, $t(XY) = \{T_1, T_3\}$

Diffset (XY, X) = $\{T_2\}$

Apriori

FPGrowth

ECLAT

Max & Closed

# FREQUENT ITEMSET METHODS

Basic Concepts

Pattern Evaluation

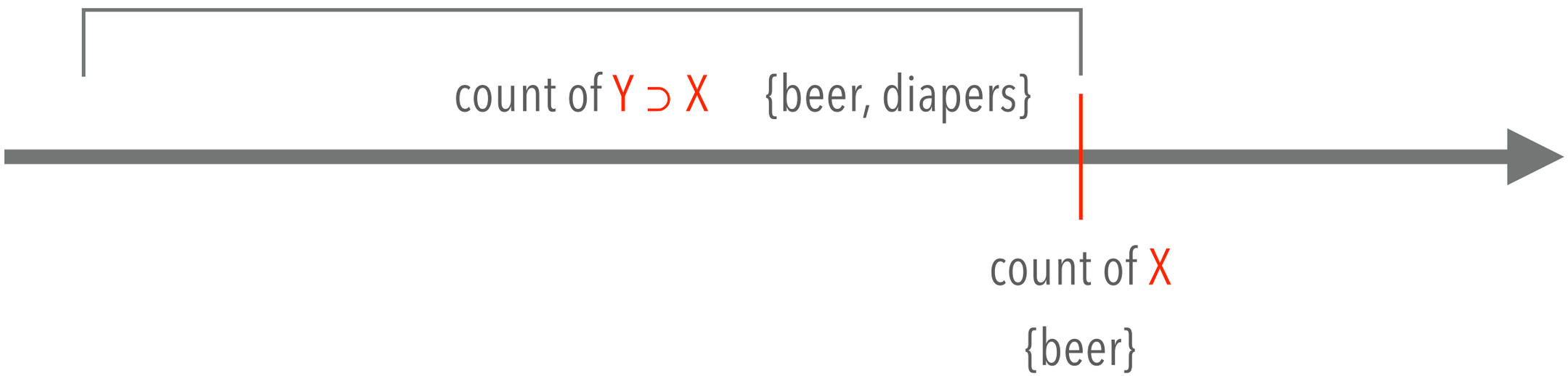
Summary

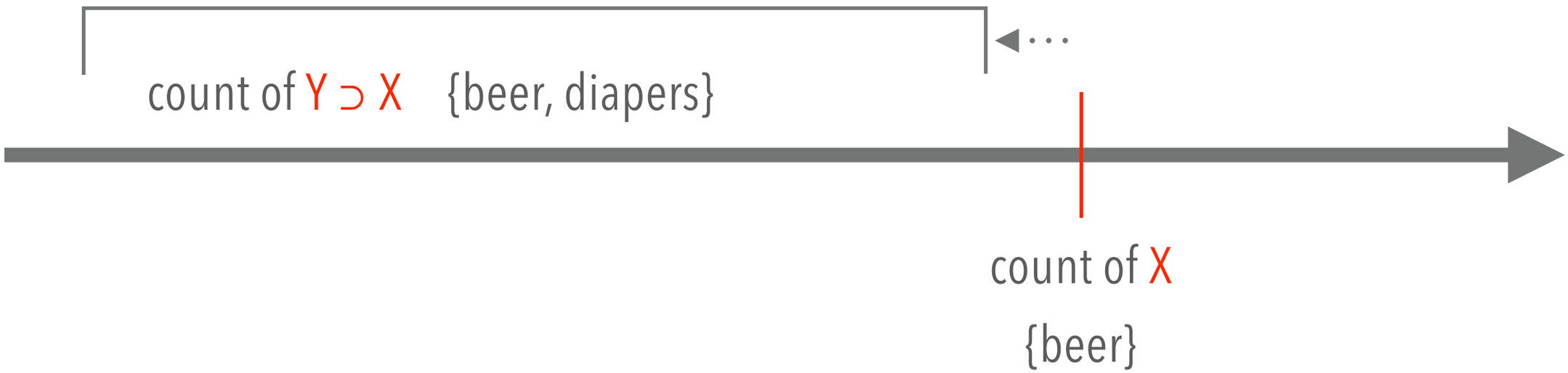
# why be interested in **closed** and **max**-patterns?

A long pattern $\{a_1, \ldots, a_{100}\}$ contains contains a **combinatorial** number of sub-patterns:

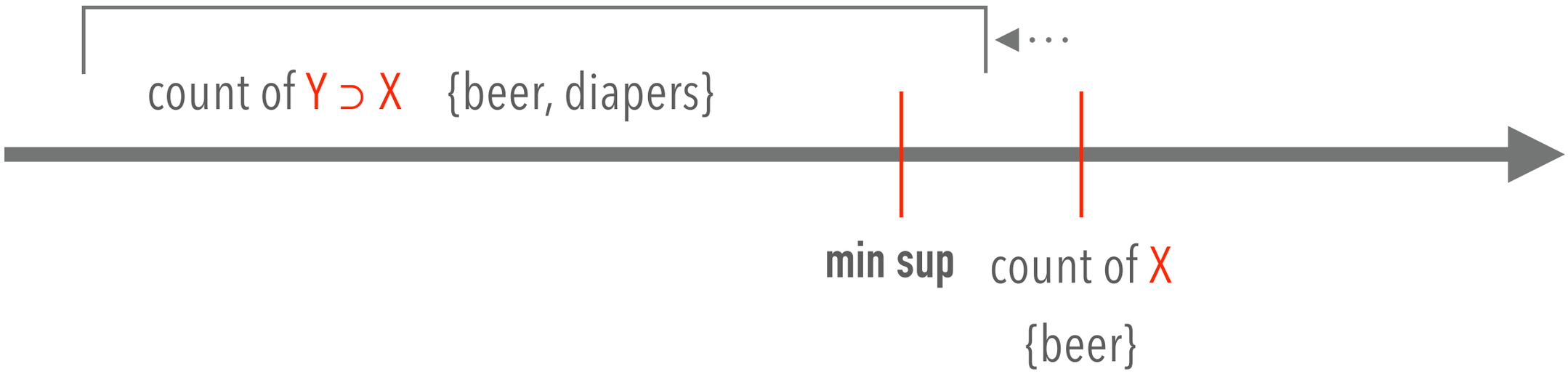
$2^{100} - 1 \cong 10^{30.1}$ **sub-patterns!**

the count of the super set Y is always less or equal to the count of set X!



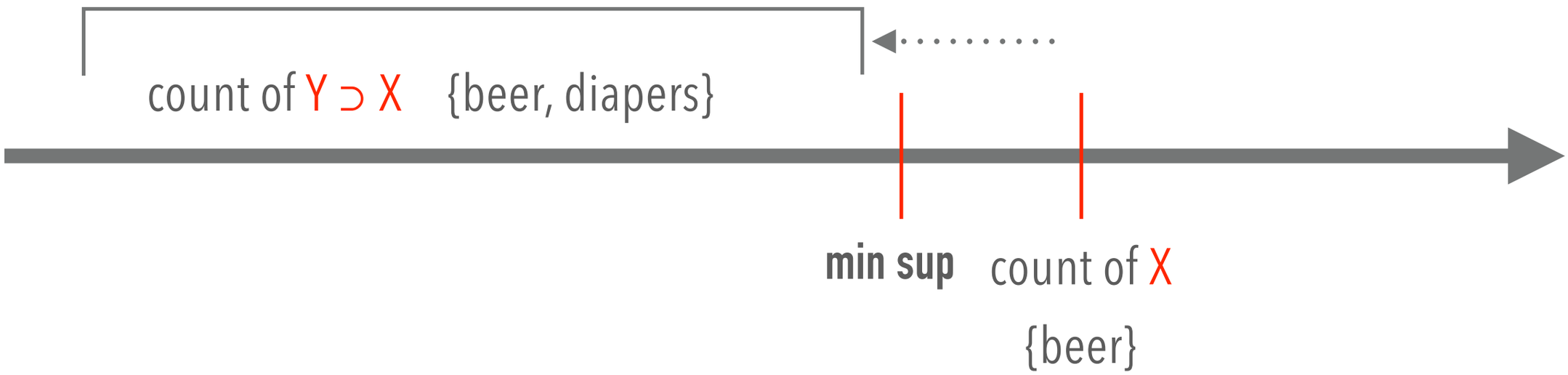



An itemset X is closed in dataset D if there exists no proper super-pattern Y ⊃ X, with the same support as X in D

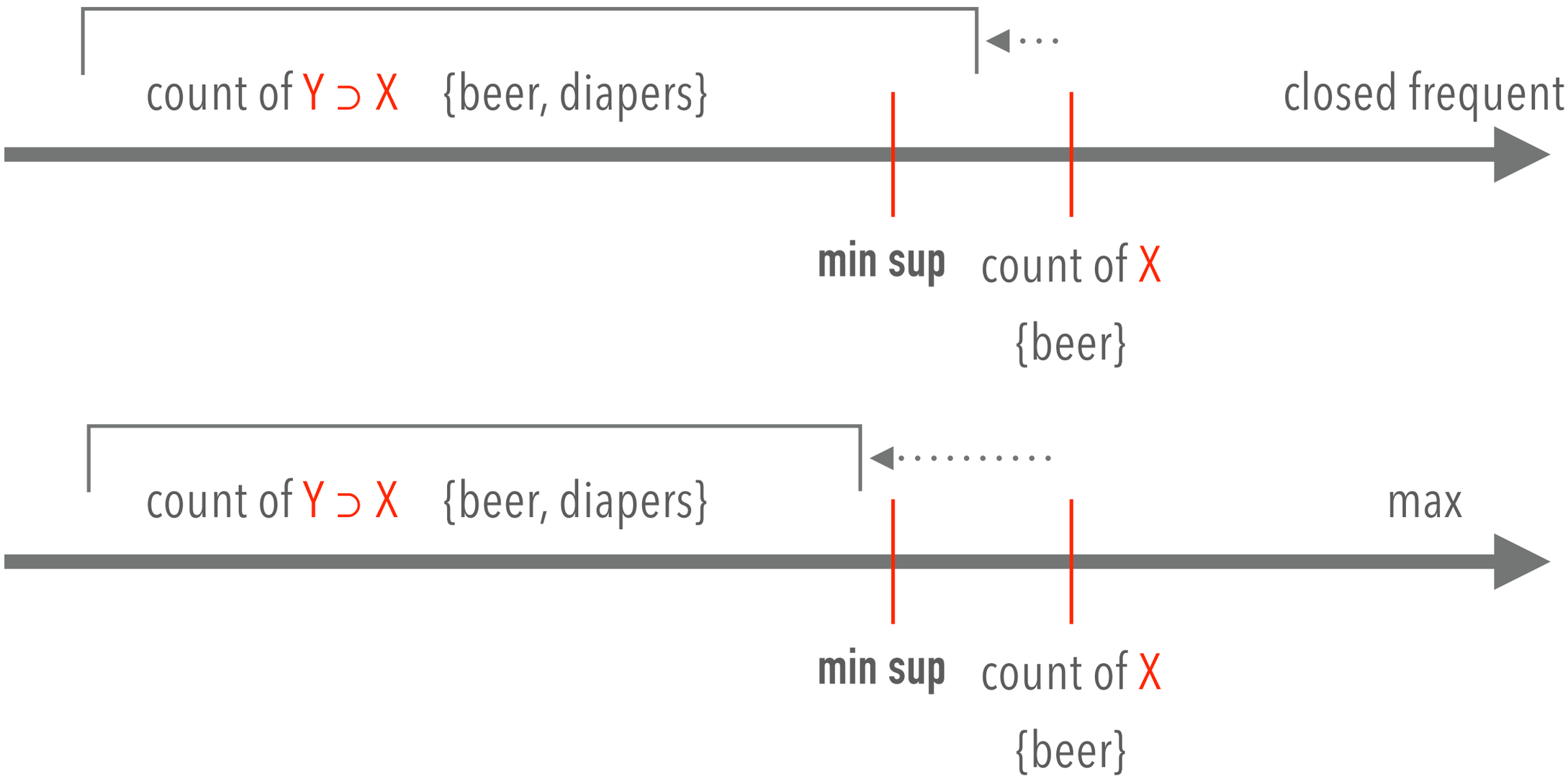



# An itemset X is a closed frequent itemset in set D if X is both closed and frequent in D

The definition does **not** say anything about Y




# An itemset X is a maximal frequent itemset (or max-itemset) in a data set D if X is frequent, and there exists no super-itemset Y such that Y ⊃ X and Y is frequent in D.




what is a key difference between the set of all **max patterns** and the set of all **closed frequent** patterns?

min_sup=2

| TID | Items |
|---|---|
| 10 | a, c, d, e, f |
| 20 | a, b, e |
| 30 | c, e, f |
| 40 | a, c, d, f |
| 50 | c, e, f |

# CLOSET: CLOSED PATTERN MINING

Flist: list of all frequent items in support descending order

f_list:<c:4, e:4, f:4, a:3, d:2>

Divide search space

Patterns having d

Patterns having a but no d, etc.

Find frequent closed pattern recursively

item set merging

Every transaction having d also has cfa→cfad is a frequent closed pattern

Pei, Jian, Jiawei Han, and Runying Mao. "CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets." In ACM SIGMOD workshop on research issues in data mining and knowledge discovery, vol. 4, no. 2, pp. 21-30. 2000.

# CLOSET: DIVIDE SEARCH SPACE

| TID | Items |
|---|---|
| 10 | a, c, d, e, f |
| 20 | a, b, e |
| 30 | c, e, f |
| 40 | a, c, d, f |
| 50 | c, e, f |

b does not satisfy **min_sup**

d has the **lowest** count satisfying **min_sup**

**f_list**:<c:4, e:4, f:4, a:3, d:2>

| Conditional DB | Itemsets |
|---|---|
| $TDB\|_d$ | cefa, cfa |
| $TDB\|_a$ | cef, e, cf |
| $TDB\|_f$ | ce:3, c |
| $TDB\|_e$ | c:3 |
| $TDB\|_c$ | {} |

b is missing!

e does not satisfy **min_sup** in TDB|d

since we've already covered item sets containing d, d is omitted

All possible frequent closed patterns can be divided into 5 non-overlapping search spaces based on f_list, i.e. x-conditional database (TDB|x)

TDB|d: The ones containing d

TDB|a: The ones containing a but no d

TDB|f: The ones containing f but no a nor d

TDB|e: The ones containing e but no f, a nor d

TDB|c: The ones containing only c

# find closed patterns containing d

| TID | Items |
|---|---|
| 10 | a, c, d, e, f |
| 20 | a, b, e |
| 30 | c, e, f |
| 40 | a, c, d, f |
| 50 | c, e, f |

**TDB**
cefad
ea
cef
cfad
cef

ordered by **f_list**;
infrequent items
are dropped

**f_list**: <c:4, e:4, f:4, a:3, d:2>

**TDB|d** (d:2)
cefa
cfa

**TDB|a** (a:3)
cef
e
cf

**TDB|f** (f:4)
ce:3
c

**TDB|e** (e:4)
c:3

<c:2, f:2, a:2>

F.C.P: cfad

The support of d=2; Items c, f and a appear twice respectively in TDB|d That is every transaction containing d also contains c, f and a Moreover e is infrequent since it appears only once in TDB|d Therefore, cfad:2 is a frequent closed itemset. Since cfad covers all frequent itemsets containing d, the mining of TDB|d is complete

# find closed patterns containing a but **not** d

Since sup(a)=3 and there no item appearing in every transaction in the a-conditional database, a:3 is a frequent closed itemset.

**TDB**
cefad
ea
cef
cfad
cef

f_list: <c:4, e:4, f:4, a:3, d:2>

**TDB|d** (d:2)
cefa
cfa

<c:2, f:2, a:2>

F.C.P: cfad:2

**TDB|a** (a:3)
cef
e
cf

<c:2, f:2, e:2>

F.C.P: a:3

**TDB|f** (f:4)
ce
c

**TDB|e** (e:4)
c

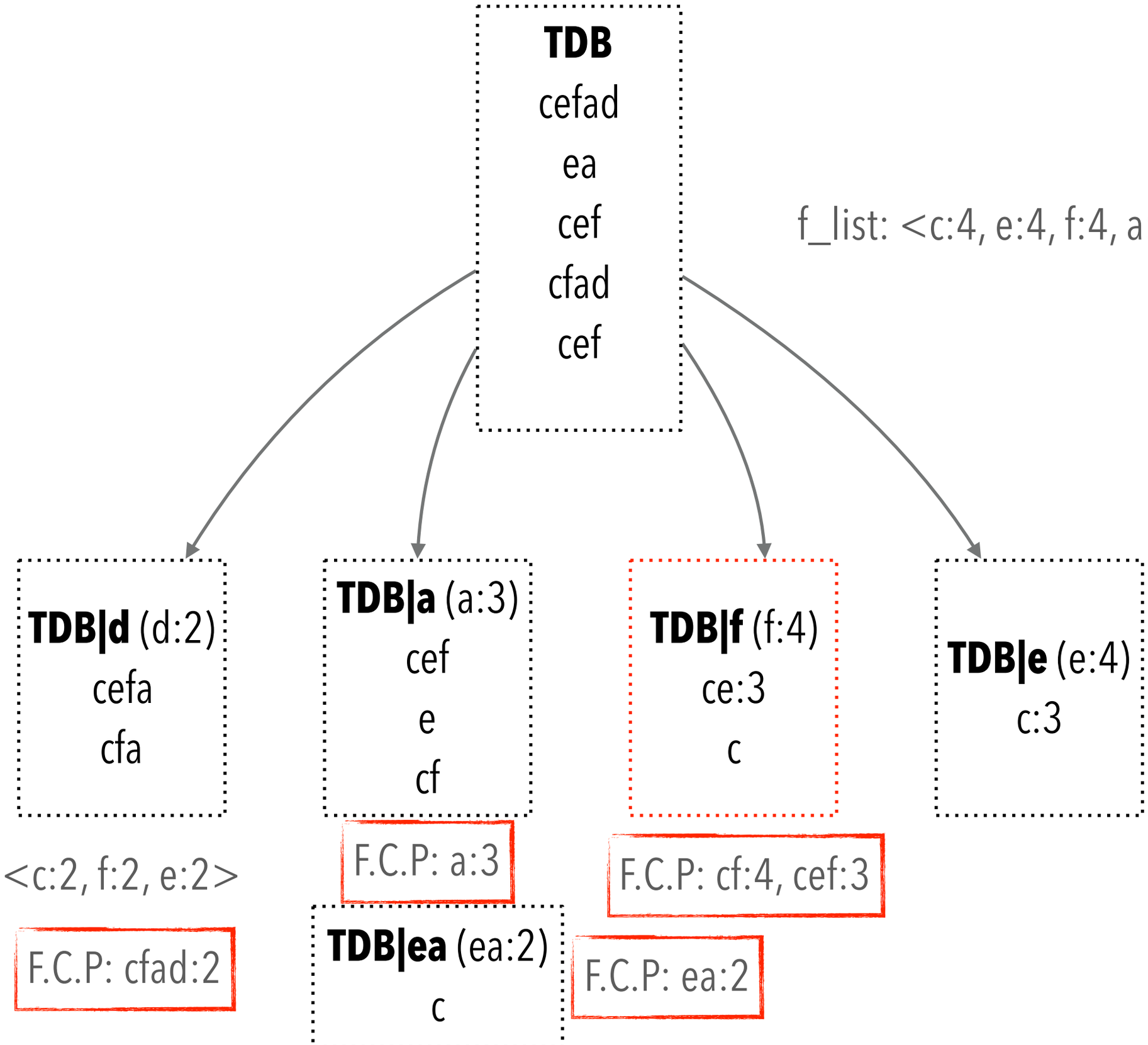
**TDB|fa** (fa:2)
ce
c

**TDB|ea** (ea:2)
c

F.C.P: ea:2

**TDB|ca** (ca:2)

# find closed patterns containing f but **not** a,d



# find closed patterns containing e but **not** a,f,d

**TDB**
cefad
ea
cef
cfad
cef

f_list: <c:4, e:4, f:4, a:3, d:2>

**sup**(c) = **sup**(cf)

**TDB|d** (d:2)
cefa
cfa

<c:2, f:2, e:2>

F.C.P: cfad:2

**TDB|a** (a:3)
cef
e
cf

F.C.P: a:3

**TDB|ea** (ea:2)
c

**TDB|f** (f:4)
ce
c

F.C.P: cf:4, cef3

F.C.P: ea:2

**TDB|e** (e:4)
c

F.C.P: e:4

**TDB|c** (c:4)

min_sup=2

# MAX MINER

1st scan: find frequent items

potential max patterns

A, B, C, D, E

2nd scan: find support for

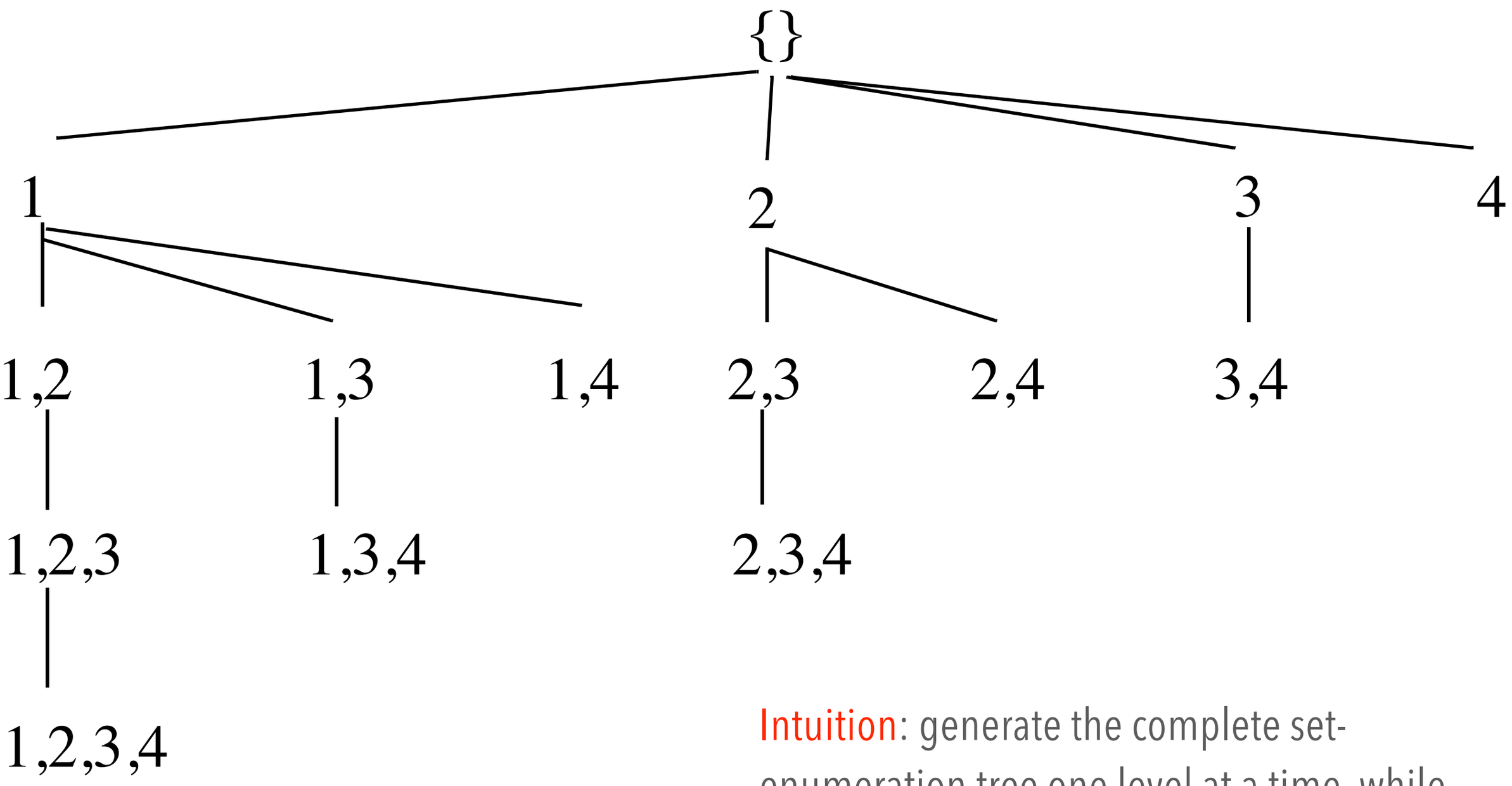
AB, AC, AD, AE, ABCDE

BC, BD, BE, BCDE

CD, CE, CDE, DE

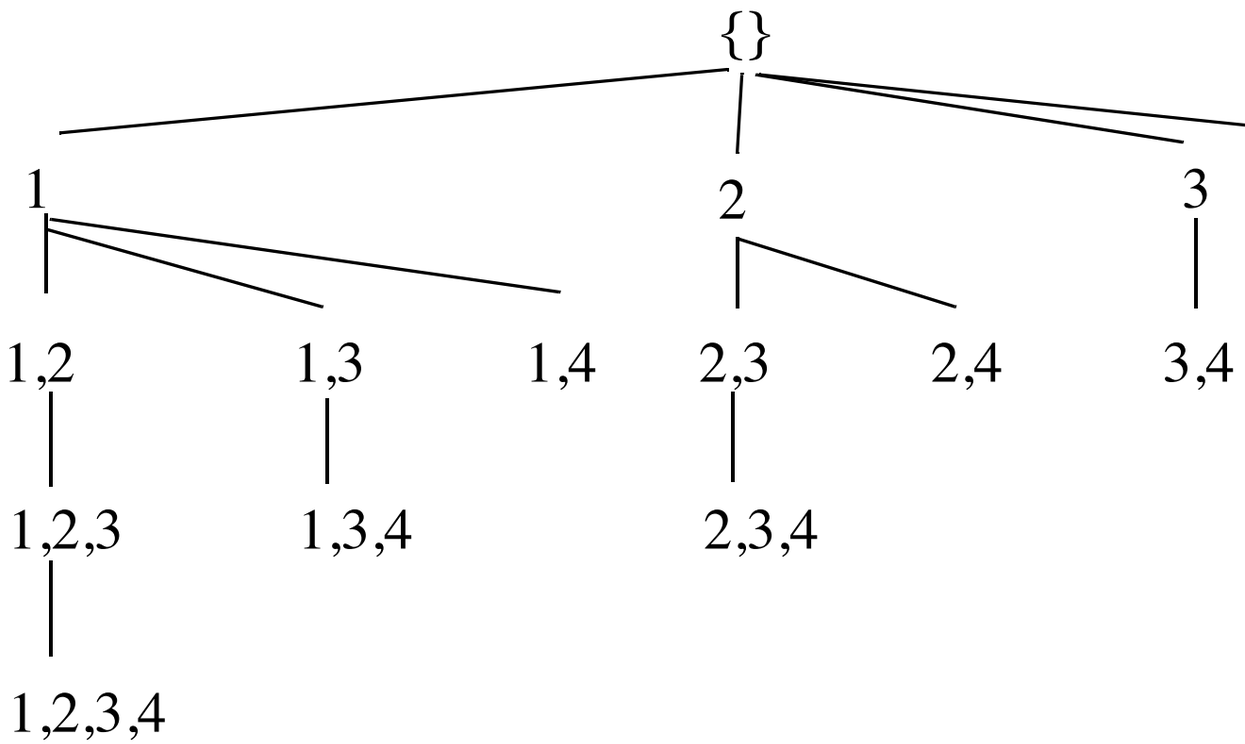
Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan

Bayardo Jr, Roberto J. "Efficiently mining long patterns from databases." In ACM Sigmod Record, vol. 27, no. 2, pp. 85-93. ACM, 1998.




Intuition: generate the complete set-enumeration tree one level at a time, while prune if applicable.

# MAXMINER




Initially, generate one node N=(1234), where head of N is h(N)=Φ and tail of N is t(N)={1,2,3,4}.

Consider expanding N,

If h(N) ∪ t(N) is frequent, **do not** expand N and prune the whole sub-tree.

Else: do local pruning. If for some i ∈ t(N), h(N) ∪ {i} is **not** frequent, remove i from t(N) before expanding N:

Apply global pruning: When a max pattern is identified, prune all nodes across sub-tree where h(N) ∪ t(N) is a sub-set of the pattern

## Apriori

Generate $C_2$ candidates from $L_1$ →

$C_2$

| Itemset |
|---|
| {I1, I2} |
| {I1, I3} |
| {I1, I4} |
| {I1, I5} |
| {I2, I3} |
| {I2, I4} |
| {I2, I5} |
| {I3, I4} |
| {I3, I5} |
| {I4, I5} |

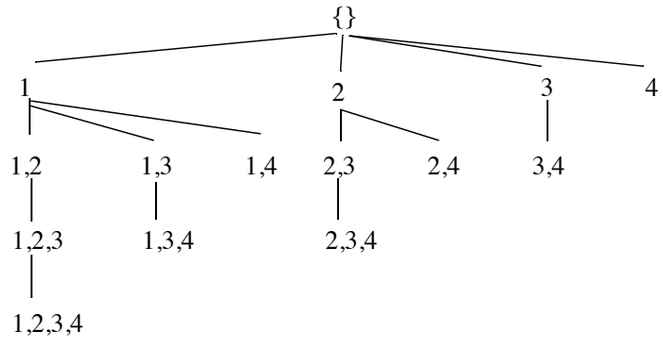
Scan $D$ for count of each candidate →

$C_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I4} | 1 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |
| {I3, I4} | 0 |
| {I3, I5} | 1 |
| {I4, I5} | 0 |

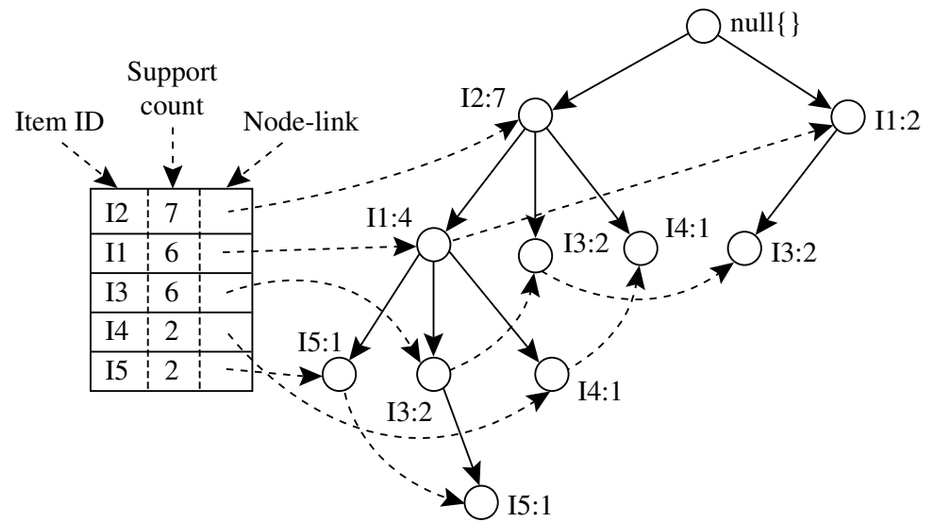
| Conditional DB | Itemsets |
|---|---|
| $TDB\|_d$ | cefa, cfa |
| $TDB\|_a$ | cef, e, cf |
| $TDB\|_f$ | ce:3, c |
| $TDB\|_e$ | c:3 |
| $TDB\|_c$ | {} |

## Max & Closed




# SUMMARY

Frequent Pattern Mining Methods




ECLAT FPGrowth

# PATTERN EVALUATION

Basic Concepts Methods Summary

# CORRELATION

| | Basketball | Not basketball | Sum (row) |
|---|---|---|---|
| Cereal | 2000 | 1750 | 3750 |
| Not cereal | 1000 | 250 | 1250 |
| Sum(col.) | 3000 | 2000 | 5000 |

play basketball→eat cereal [40%, 66.7%] is misleading

This is because the overall percentage of students eating cereal is
75% > 66.7%.

play basketball→not eat cereal [20%, 33.3%] is more accurate, although with lower support and confidence

lift: Measure of dependent or correlated events:

**Why is this happening?**

$$lift(A,B) = \frac{P(A \cup B)}{P(A)P(B)}$$

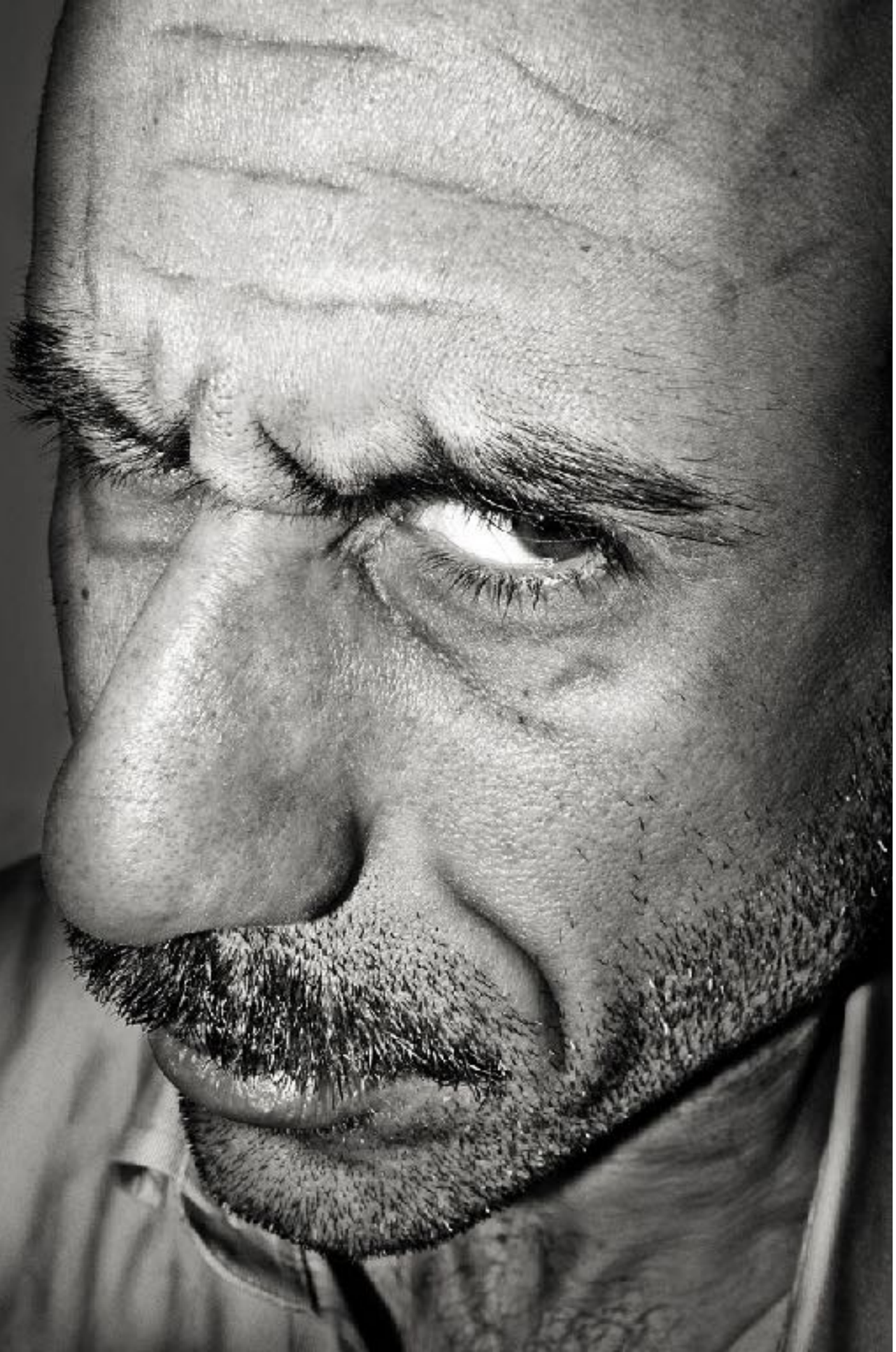
$$lift(B,C) = \frac{2000/5000}{3000/5000 \times 3750/5000}$$

$$lift(B,\bar{C}) = \frac{1000/5000}{3000/5000 \times 1250/5000}$$

**Exercise!**

| | $game$ | $\overline{game}$ | $\Sigma_{row}$ |
|---|---|---|---|
| $video$ | 4000 | 3500 | 7500 |
| $\overline{video}$ | 2000 | 500 | 2500 |
| $\Sigma_{col}$ | 6000 | 4000 | 10,000 |

games → videos?



## WHAT COULD BE A PROBLEM WITH LIFT?

$$lift(A,B) = \frac{P(A \cup B)}{P(A)P(B)}$$

video and games are **negatively** correlated!

| | *game* | $\overline{game}$ | $\Sigma_{row}$ |
|---|---|---|---|
| *video* | 4000 (4500) | 3500 (3000) | 7500 |
| $\overline{video}$ | 2000 (1500) | 500 (1000) | 2500 |
| $\Sigma_{col}$ | 6000 | 4000 | 10,000 |

expected values

$$\chi^2 = \Sigma \frac{(observed - expected)^2}{expected} = \frac{(4000-4500)^2}{4500} + \frac{(3500-3000)^2}{3000}$$
$$+ \frac{(2000-1500)^2}{1500} + \frac{(500-1000)^2}{1000} = 555.6. \quad p < 0.001$$

$$all\_conf(A,B) = \frac{sup(A \cup B)}{max\{sup(A), sup(B)\}} = min\{P(A|B), P(B|A)\}$$

$$max\_conf(A, B) = max\{P(A \mid B), P(B \mid A)\}.$$

# Null invariant measures

$$Kulc(A, B) = \frac{1}{2}(P(A|B) + P(B|A)).$$

$$cosine(A, B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}}$$
$$= \sqrt{P(A|B) \times P(B|A)}.$$

$$all\_conf(A,B) = \frac{sup(A \cup B)}{max\{sup(A), sup(B)\}} = min\{P(A|B), P(B|A)\}$$

$$lift(A,B) = \frac{P(A \cup B)}{P(A)P(B)}$$

$$max\_conf(A, B) = max\{P(A \mid B), P(B \mid A)\}.$$

$$Kulc(A, B) = \frac{1}{2}(P(A|B) + P(B|A)).$$

$$cosine(A, B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}} = \sqrt{P(A|B) \times P(B|A)}.$$

| | $milk$ | $\overline{milk}$ | $\Sigma_{row}$ |
|---|---|---|---|
| $coffee$ | $mc$ | $\overline{m}c$ | $c$ |
| $\overline{coffee}$ | $m\overline{c}$ | $\overline{mc}$ | $\overline{c}$ |
| $\Sigma_{col}$ | $m$ | $\overline{m}$ | $\Sigma$ |

# canonical contingency table

| Data Set | $mc$ | $\overline{m}c$ | $m\overline{c}$ | $\overline{mc}$ | $\chi^2$ | *lift* | *all_conf.* | *max_conf.* | *Kulc.* | *cosine* |
|---|---|---|---|---|---|---|---|---|---|---|
| $D_1$ | 10,000 | 1000 | 1000 | 100,000 | 90557 | 9.26 | 0.91 | 0.91 | 0.91 | 0.91 |
| $D_2$ | 10,000 | 1000 | 1000 | 100 | 0 | 1 | 0.91 | 0.91 | 0.91 | 0.91 |
| $D_3$ | 100 | 1000 | 1000 | 100,000 | 670 | 8.44 | 0.09 | 0.09 | 0.09 | 0.09 |
| $D_4$ | 1000 | 1000 | 1000 | 100,000 | 24740 | 25.75 | 0.5 | 0.5 | 0.5 | 0.5 |
| $D_5$ | 1000 | 100 | 10,000 | 100,000 | 8173 | 9.18 | 0.09 | 0.91 | 0.5 | 0.29 |
| $D_6$ | 1000 | 10 | 100,000 | 100,000 | 965 | 1.97 | 0.01 | 0.99 | 0.5 | 0.10 |

$$all\_conf(A,B) = \frac{sup(A \cup B)}{max\{sup(A), sup(B)\}} = min\{P(A|B), P(B|A)\}$$

$$lift(A,B) = \frac{P(A \cup B)}{P(A)P(B)}$$

$$max\_conf(A, B) = max\{P(A \mid B), P(B \mid A)\}.$$

$$Kulc(A, B) = \frac{1}{2}(P(A|B) + P(B|A)).$$

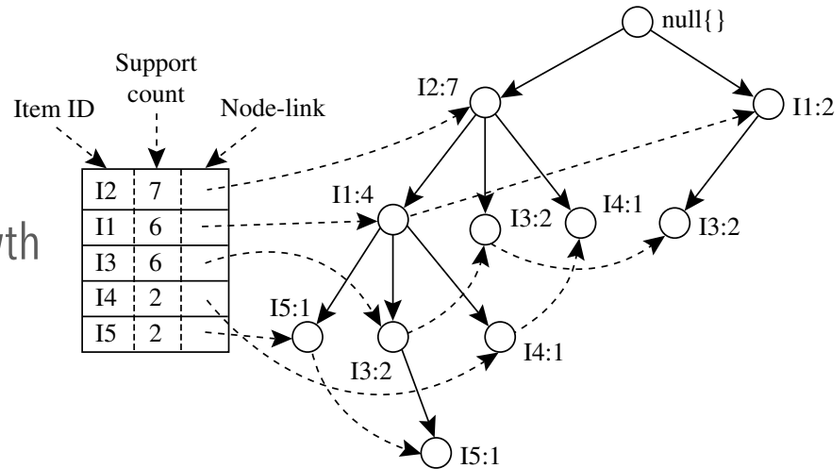
$$cosine(A, B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}} = \sqrt{P(A|B) \times P(B|A)}.$$

**imbalance ratio**

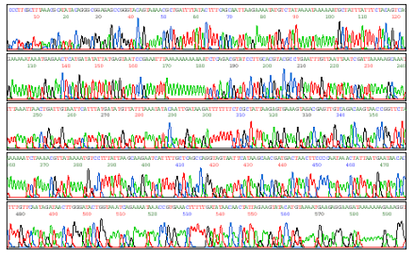
$$IR(A,B) = \frac{|sup(A) - sup(B)|}{sup(A) + sup(B) - sup(A \cup B)}$$

# canonical contingency table

| Data Set | $mc$ | $\overline{m}c$ | $m\overline{c}$ | $\overline{mc}$ | $\chi^2$ | *lift* | *all_conf.* | *max_conf.* | *Kulc.* | *cosine* |
|---|---|---|---|---|---|---|---|---|---|---|
| $D_1$ | 10,000 | 1000 | 1000 | 100,000 | 90557 | 9.26 | 0.91 | 0.91 | 0.91 | 0.91 |
| $D_2$ | 10,000 | 1000 | 1000 | 100 | 0 | 1 | 0.91 | 0.91 | 0.91 | 0.91 |
| $D_3$ | 100 | 1000 | 1000 | 100,000 | 670 | 8.44 | 0.09 | 0.09 | 0.09 | 0.09 |
| $D_4$ | 1000 | 1000 | 1000 | 100,000 | 24740 | 25.75 | 0.5 | 0.5 | 0.5 | 0.5 |
| IR=0.89 $D_5$ | 1000 | 100 | 10,000 | 100,000 | 8173 | 9.18 | 0.09 | 0.91 | 0.5 | 0.29 |
| IR=0.99 $D_6$ | 1000 | 10 | 100,000 | 100,000 | 965 | 1.97 | 0.01 | 0.99 | 0.5 | 0.10 |




FP Growth

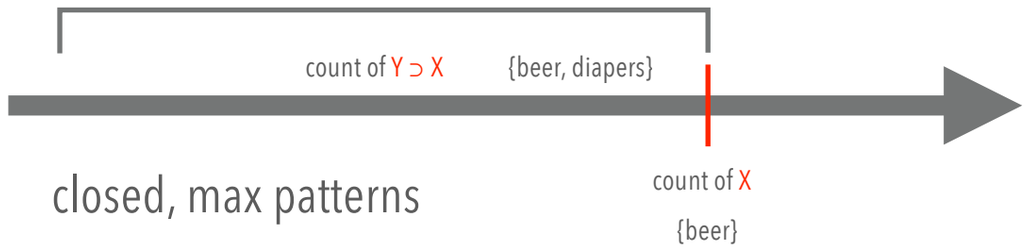


a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set

itemset, support, frequent, confidence

# FREQUENT ITEMSET METHODS

Closed itemset mining, max-pattern mining




closed, max patterns

Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated or tested!

**challenges**

hashing, partitioning, sampling

**Evaluation**

lift, $\chi^2$, min, max, Kulc, cosine

**null invariant**

Summary