

# 大数据计算及应用(二)

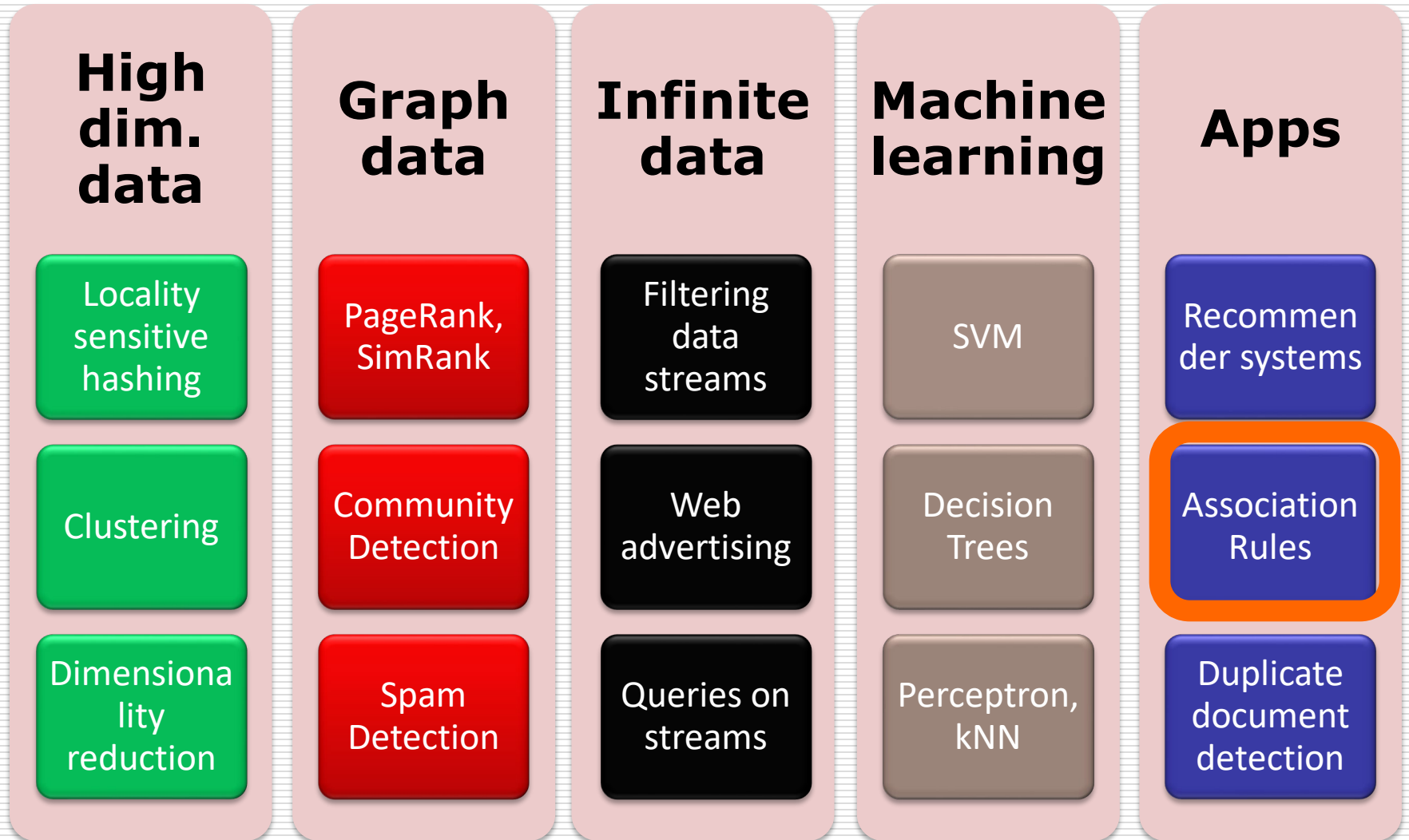
---

## Association Rules and Frequent Pattern Mining

Slides adapted from <http://www.mmds.org>

# Agenda

---



# Association Rule

---

- Items frequently purchased together:

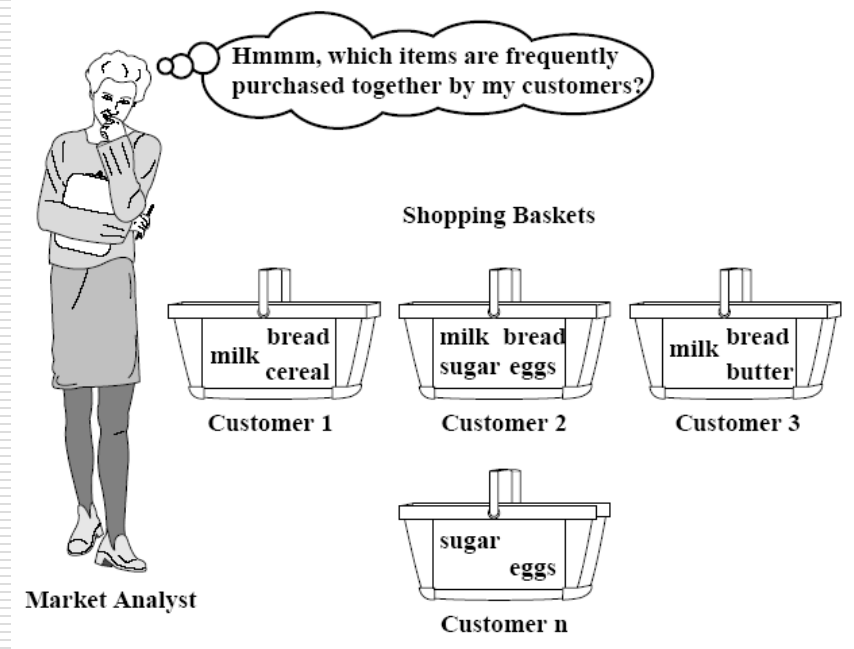
beer → diaper



- Uses:

- Placement
- Advertising
- Sales
- Coupons

- Objective: increase sales and reduce costs



# The Market-Basket Model

---

- A large set of *items*, e.g., things sold in a supermarket
- A large set of *baskets*, each of which is a small set of the items, e.g., the things one customer buys on one day

<i>TID</i>	<i>Items</i>
1	Beer, Diaper, Milk
2	Coke, Diaper, Eggs
3	Beer, Coke, Diaper, Eggs
4	Coke, Eggs

# The Market-Basket Model

---

- A general many-many mapping (association) between two kinds of things
  - But we ask about connections among “items” not “baskets”
- The technology focuses on **common events**, not rare events (“long tail”)

# Applications – (1)

---

- **Items** = products; **baskets** = sets of products someone bought in one trip to the store
- **Example application**: given that many people buy beer and diapers together
  - Run a sale on diapers; raise price of beer
- Only useful if many buy diapers & beer

# Applications – (2)

---

- **Items** = words; **Baskets** = Web pages;
- Unusual words appearing together in a large number of documents, e.g., “Brad” and “Angelina” may indicate an interesting relationship

# Applications – (3)

---

- **Items** = sentences; **baskets** = documents containing those sentences
- Items that appear together too often could represent plagiarism



# Association Rule Mining Applications

---

- ☐ Basket Data Analysis
- ☐ Genomic Data
- ☐ Telecommunication
- ☐ Credit Cards/ Banking Services
- ☐ Medical Treatments
- ☐ Web Personalization
- ☐ etc.

# Scale of the Problem

---

- ❑ WalMart sells 100,000 items and can store billions of baskets
- ❑ The Web has billions of words and many billions of pages

# Some Definition - Support

---

An itemset is **supported** by a basket (transaction) if it is included in the basket

## Market-Basket transactions

<i>TID</i>	<i>Items</i>
1	<b>Beer</b> , <b>Diaper</b> , Milk
2	Coke, Diaper, Eggs
3	<b>Beer</b> , Coke, <b>Diaper</b> , Eggs
4	Coke, Eggs

<Beer, Diaper> is **supported** by basket 1, and 3, and its support is  $2/4=50\%$ .

# Some Definition – Frequent Itemset

---

If the support of an itemset exceeds user specified *min\_support* (threshold), this itemset is called a **frequent itemset (pattern)**.

## Market-Basket transactions

<i>TID</i>	<i>Items</i>
1	<b>Beer</b> , <b>Diaper</b> , Milk
2	Coke, Diaper, Eggs
3	<b>Beer</b> , Coke, <b>Diaper</b> , Eggs
4	Coke, Eggs

min\_support=50%

<Beer, Diaper> is a frequent itemset

<Beer, Milk> is not a frequent itemset

# Association Rules

---

## □ Association Rules:

If-then rules about the contents of baskets

□  $\{i_1, i_2, \dots, i_k\} \rightarrow j$  means: “if a basket contains all of  $i_1, \dots, i_k$  then it is *likely* to contain  $j$ ”

□ In practice there are many rules, want to find significant/interesting ones!

□ *Confidence* of this association rule is the probability of  $j$  given  $I = \{i_1, \dots, i_k\}$

$$\text{conf}(I \rightarrow j) = \frac{\text{support}(I \cup j)}{\text{support}(I)}$$

# Example: Confidence

---

$$T_1 = \{m, c, b\}$$

$$T_2 = \{m, p, j\}$$

$$T_3 = \{m, b\}$$

$$T_4 = \{c, j\}$$

$$T_5 = \{m, p, b\}$$

$$T_6 = \{m, c, b, j\}$$

$$T_7 = \{c, b, j\}$$

$$T_8 = \{b, c\}$$

□ Association rule:  $\{m, b\} \rightarrow c$

■  $\text{Support}(m, b) = 4/8$ ,  $\text{Support}(m, b, c) = 2/8$

■ **Confidence** =  $2/4 = 0.5$

# Association Rules Mining

---

- ❑ Question: "find all association rules with support  $\geq s$  and confidence  $\geq c$ "
- ❑ Hard part: finding the frequent itemsets

# Computation Model

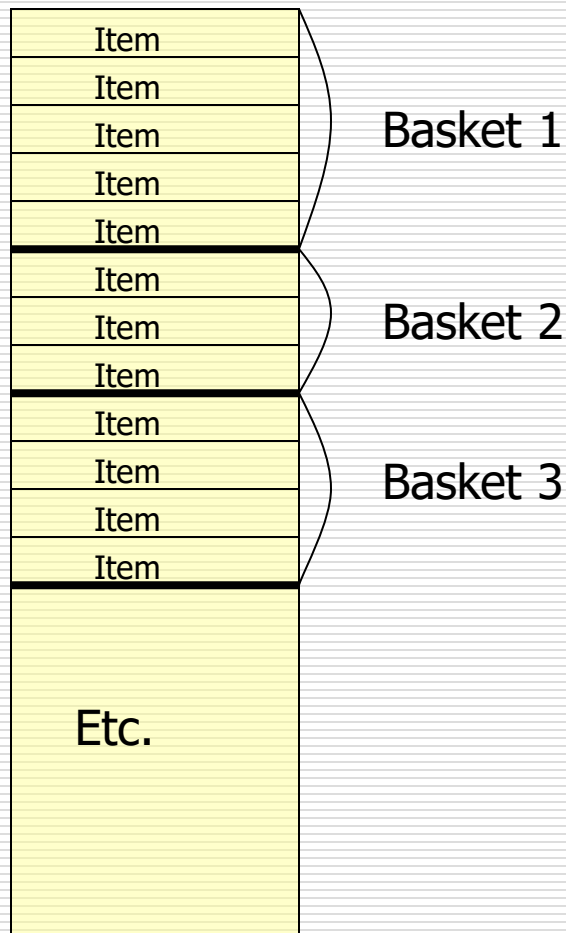
---

- Typically, data is kept in flat files rather than in a database system
  - Stored on disk
  - Stored basket-by-basket
  - Expand baskets into pairs, triples, etc. as you read baskets



# File Organization

---



**Example:** items are positive integers, and boundaries between baskets are  $-1$

# Computation Model – (2)

---

- ❑ The true cost of mining disk-resident data is usually the **number of disk I/O's**
- ❑ In practice, association-rule algorithms read the data in *passes* – all baskets read in turn
- ❑ Thus, we measure the cost by the **number of passes** an algorithm takes

# Main-Memory Bottleneck

---

- For many frequent-itemset algorithms, main memory is the critical resource
  - As we read baskets, we need to count something, e.g., occurrences of pairs
  - The number of different things we can count is limited by main memory
  - Swapping counts in/out is a disaster

# Finding Frequent Pairs

---

- The hardest problem often turns out to be finding the frequent pairs
  - Why? Often frequent pairs are common, frequent triples are rare
    - Why? Probability of being frequent drops exponentially with size; number of sets grows more slowly with size
- We'll concentrate on pairs, then extend to larger sets

# Naïve Algorithm

---

- ❑ Read file once, counting in main memory the occurrences of each pair
  - From each basket of  $n$  items, generate its  $n(n-1)/2$  pairs by two nested loops
- ❑ Fails if  $(\text{\#items})^2$  exceeds main memory
  - **Remember:** #items can be 100K (Wal-Mart) or 10B (Web pages)

## Example: Counting Pairs

---

- ❑ Suppose  $10^5$  items
- ❑ Suppose counts are 4-byte integers
- ❑ Number of pairs of items:  $10^5(10^5-1)/2 = 5*10^9$  (approximately)
- ❑ Therefore,  $2*10^{10}$  (20 gigabytes) of main memory needed

# Details of Main-Memory Counting

---

## □ Two approaches:

(1) Count all pairs, using a triangular matrix

- requires only 4 bytes/pair

always assume integers are 4 bytes

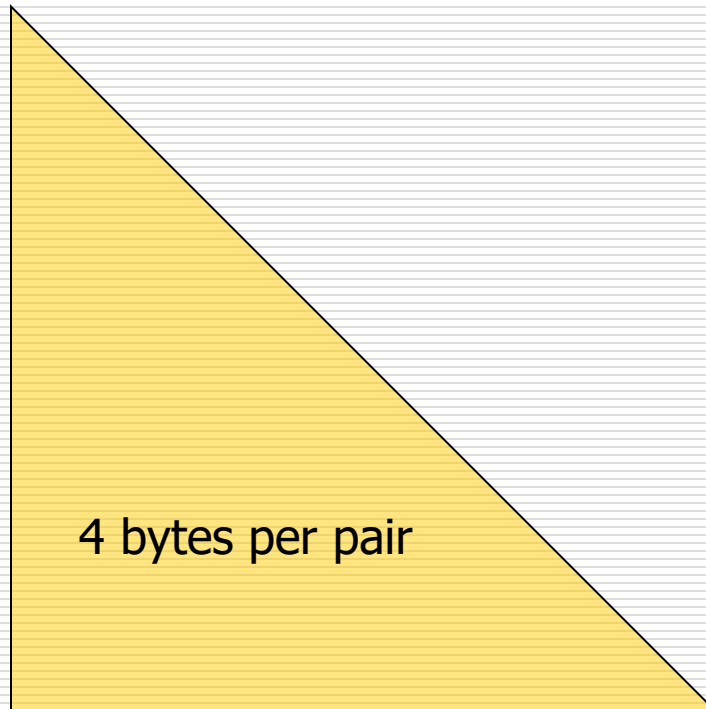
(2) Keep a table of triples  $[i, j, c]$  = "the count of the pair of items  $\{i, j\}$  is  $c$ "

- requires 12 bytes

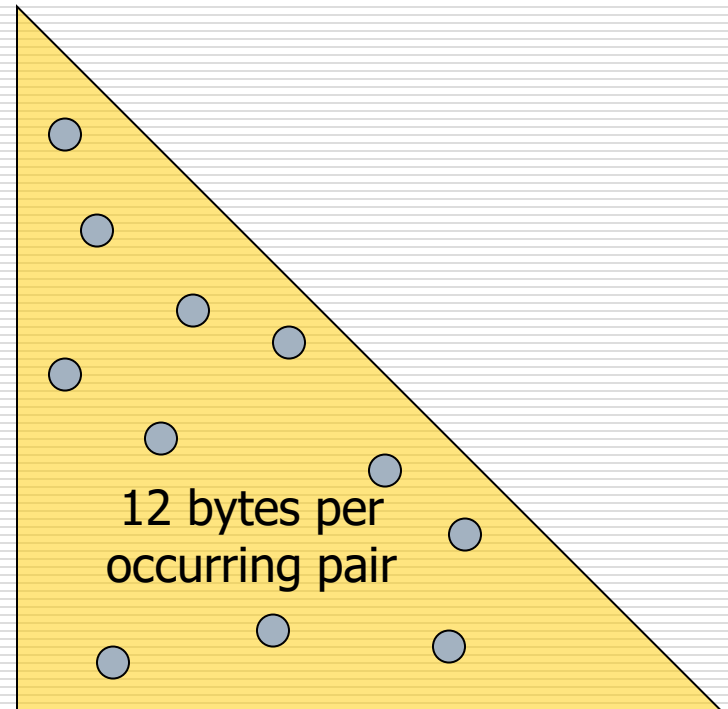
but only for those pairs with count  $> 0$

# Details of Main-Memory Counting

---



Method (1)



Method (2)



# Comparing the Two Approaches

---

## □ Approach 1: Triangular Matrix

- $n$  = total number of items
- Count pair of items  $\{i, j\}$  only if  $i < j$
- Keep pair counts in lexicographic order:
  - $\{1,2\}, \{1,3\}, \dots, \{1,n\}, \{2,3\}, \{2,4\}, \dots, \{2,n\}, \{3,4\}, \dots, \{n-1,n\}$
- Pair  $\{i, j\}$  is at position  $(i-1)(n-i/2) + j - i$
- Total number of pairs  $n(n-1)/2$ ; total bytes =  $2n^2$
- **Triangular Matrix** requires 4 bytes per pair

## □ Approach 2 uses **12 bytes** per occurring pair (*but only for pairs with count > 0*)

- Beats Approach 1 if less than **1/3** of possible pairs actually occur

# Comparing the Two Approaches

---

## □ Approach 1: Triangular Matrix

- $n$  = total number items
- Count pair of items  $\{i, j\}$  only if  $i < j$

**Problem is if we  
have too many  
items so the pairs  
do not fit into  
memory.**

□ Approach 2:  
(b)

■ **Can we do better?**

possible pairs actually occur

$s = 2n^2$

pair

# Outline

---

- Association Rules
- Frequent Itemset Mining Algorithms
  - Apriori
  - FP-growth
- Sequential Pattern Mining Algorithms

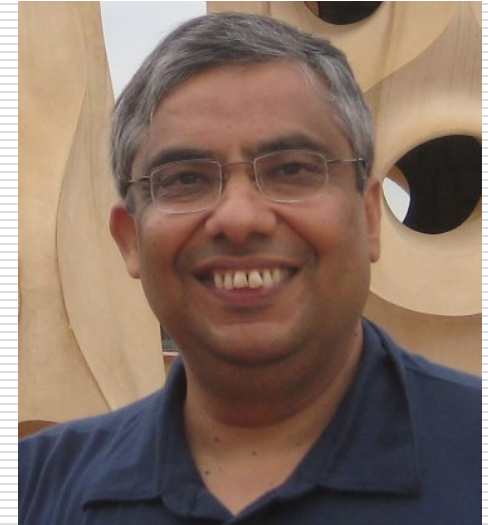
# Apriori Algorithm

---

□ Proposed by Rakesh Agrawal [VLDB'94]

□ Key idea:

- Candidate generation-and-test
- Anti-monotone property



<http://www.vldb.org> › conf PDF ⋮

## Fast Algorithms for Mining Association Rules - VLDB ...

by R Agrawal · Cited by 28692 — We consider the problem of discovering **association rules** between items in a large database of sales transactions. We present two new **algorithms** for...  
13 pages

# Apriori Algorithm – (1)

---

- A two-pass approach called *Apriori* limits the need for main memory
- *Monotonicity* : if a set of items appears at least  $s$  times, so does every subset
  - *Contrapositive for pairs*: if item  $i$  does not appear in  $s$  baskets, then no pair including  $i$  can appear in  $s$  baskets

# Apriori Algorithm – (2)

---

- **Pass 1:** Read baskets and count in main memory the occurrences of each item
  - Requires only memory proportional to #items
- Items that appear at least  $s$  times are the *frequent items*

# Apriori Algorithm – (3)

---

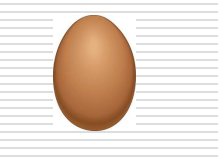
- **Pass 2:** Read baskets again and count in main memory only those pairs both of which were found in Pass 1 to be frequent
- Requires memory proportional to square of *frequent* items only (for counts), plus a list of the frequent items (so you know what must be counted)

# Apriori Algorithm

---

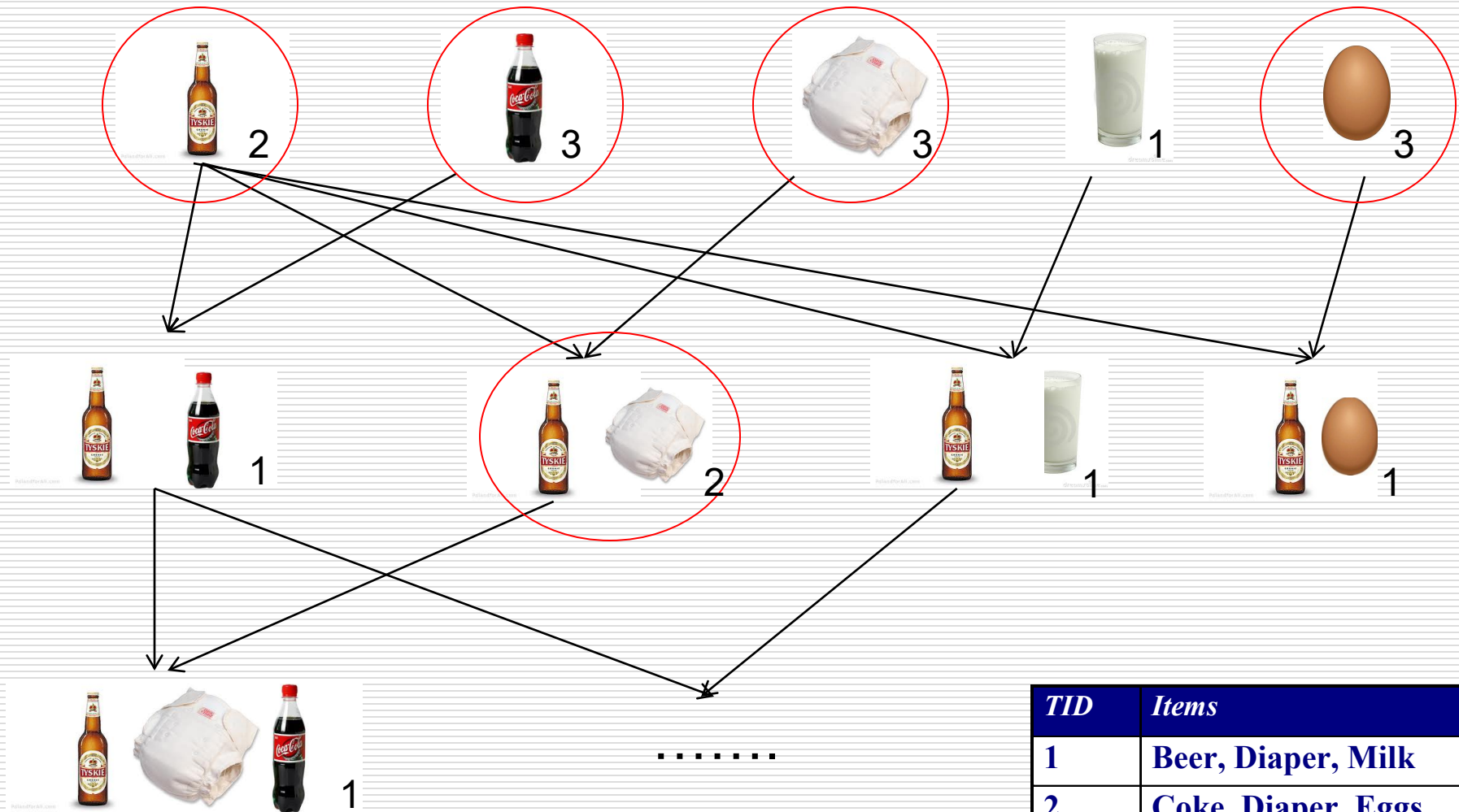
## Market-Basket transactions

<i>TID</i>	<i>Items</i>
1	Beer, Diaper, Milk
2	Coke, Diaper, Eggs
3	Beer, Coke, Diaper, Eggs
4	Coke, Eggs





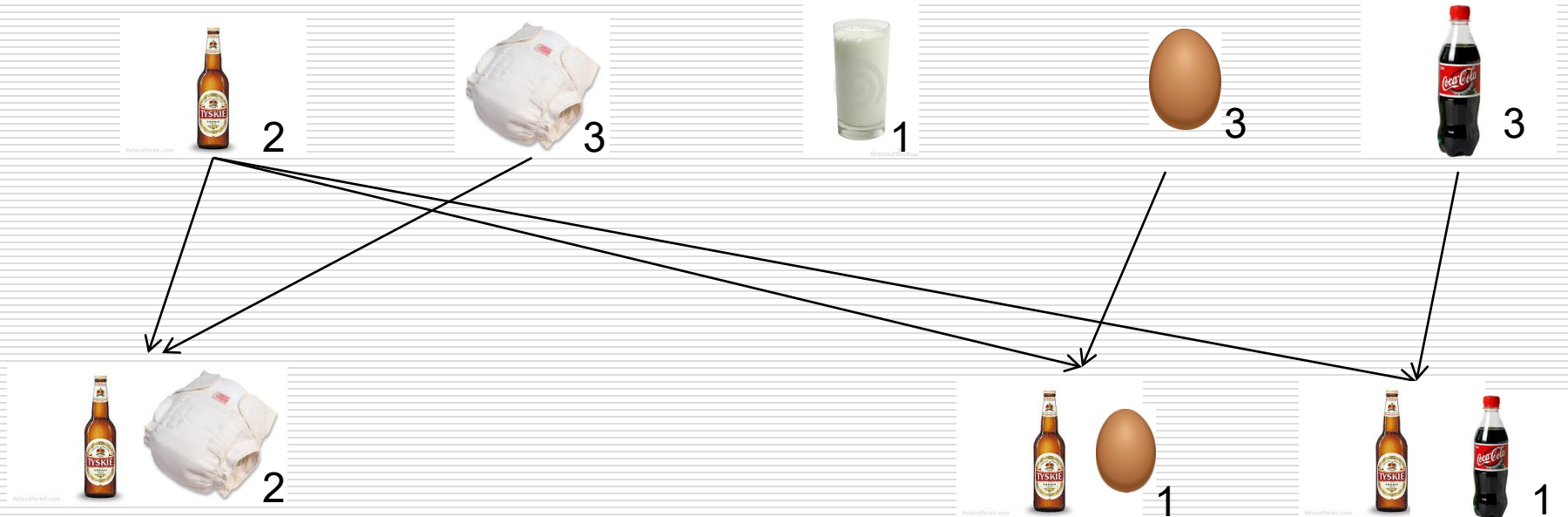
# Naive Algorithm



TID	Items
1	Beer, Diaper, Milk
2	Coke, Diaper, Eggs
3	Beer, Coke, Diaper, Eggs
4	Coke, Eggs

# Apriori Algorithm

- Anti-monotone property: If an itemset is not frequent, then any of its superset is not frequent



$\text{Sup}_{\min}=2$

<i>TID</i>	<i>Items</i>
1	Beer, Diaper, Milk
2	Coke, Diaper, Eggs
3	Beer, Coke, Diaper, Eggs
4	Coke, Eggs

# Apriori Algorithm

$\text{Sup}_{\min} = 2$

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1<sup>st</sup> scan

$C_1$

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

$L_1$

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

$L_2$

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

$C_2$

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

2<sup>nd</sup> scan

$C_2$

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

$C_3$

Itemset
{B, C, E}

3<sup>rd</sup> scan

$L_3$

Itemset	sup
{B, C, E}	2

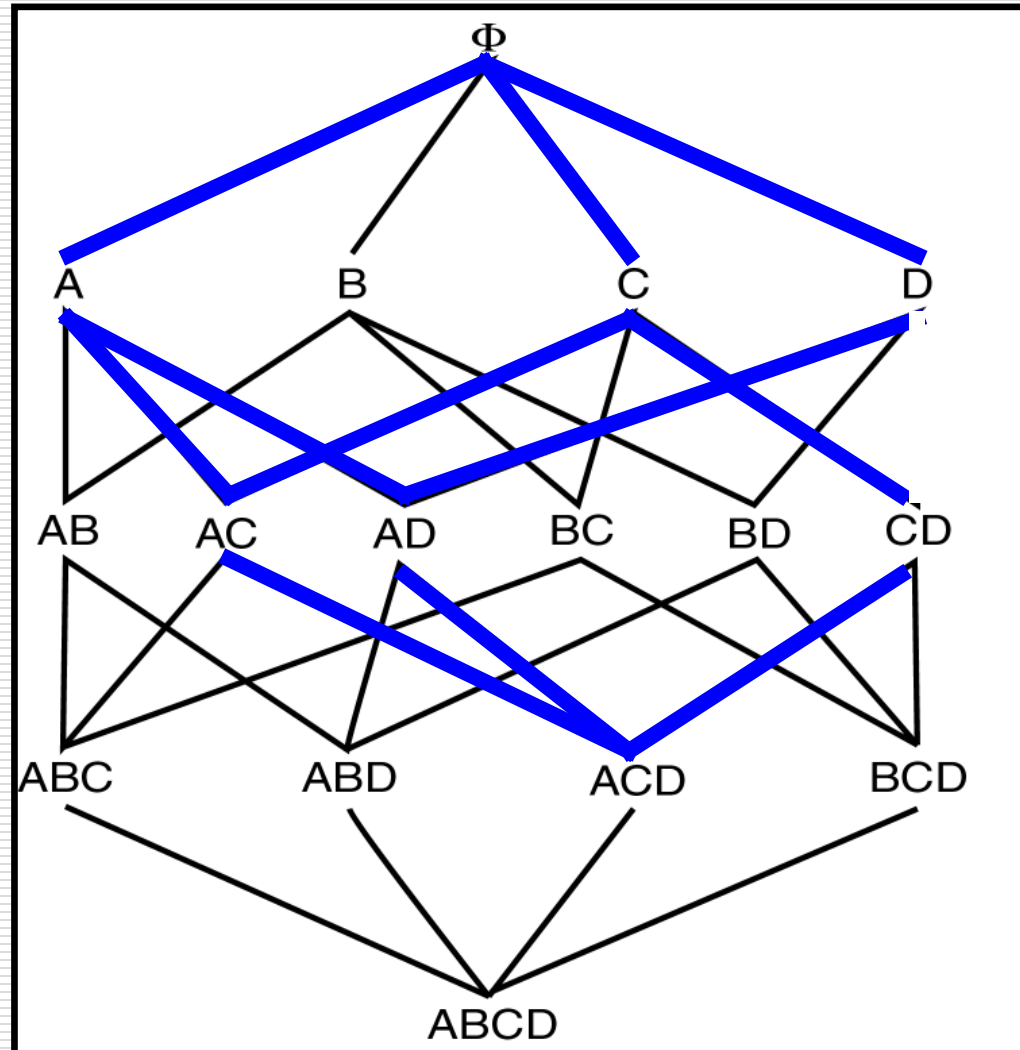
# Apriori Algorithm

---

1.  $C_1$  = Itemsets of size one in  $I$ ;
2. Determine all large itemsets of size 1,  $L_1$ ;
3.  $i = 1$ ;
4. Repeat
5.      $i = i + 1$ ;
6.      $C_i$  = Apriori-Gen( $L_{i-1}$ );
7.     Count  $C_i$  to determine  $L_i$ ;
8. until no more large itemsets found;

# Frequent Itemset Property

---



# Drawbacks of Apriori

---

- Multiple scans of transaction database
  - Multiple database scans are **costly**
- Huge number of candidates
  - To find frequent itemset  $i_1 i_2 \dots i_{100}$ 
    - # of scans: **100**
    - # of Candidates:  $2^{100}-1 = 1.27*10^{30}$

# Improving Apriori: General Ideas

---

- ❑ Reduce passes of transaction database scans
- ❑ Shrink number of candidates
- ❑ Facilitate support counting of candidates

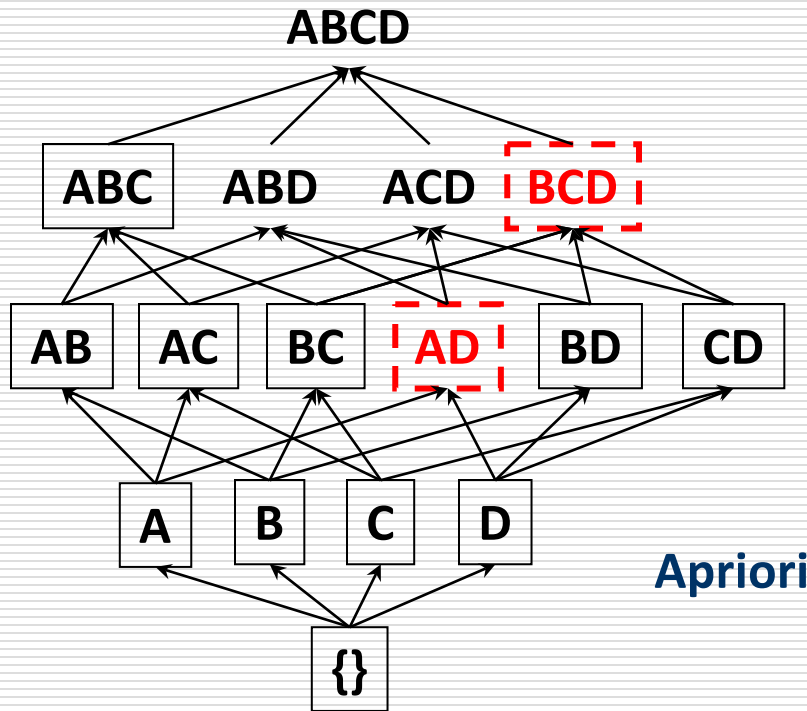
# Improving Apriori's Efficiency

---

- ❑ **Hash-based itemset counting**: A  $k$ -itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- ❑ **Transaction reduction**: A transaction that does not contain any frequent  $k$ -itemset is useless in subsequent scans
- ❑ **Partitioning**: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
- ❑ **Sampling**: mining on a subset of given data, need a lower support threshold + a method to determine the completeness
- ❑ **Dynamic itemset counting**: add new candidate itemsets immediately (unlike Apriori) when all of their subsets are estimated to be frequent



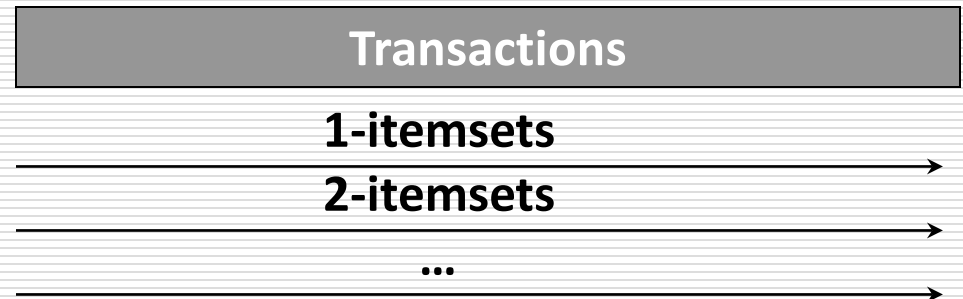
# DIC: Reduce Number of Scans



Itemset lattice

Apriori

- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins



S. Brin , R. Motwani, J. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD'97*



# FP-growth Algorithm

---

# FP-Growth

---

- Proposed by Jiawei Han [*SIGMOD'00*]
- Uses the Apriori pruning principle
- Scan DB only twice
  - Once to find frequent 1-itemset (single item pattern)
  - Once to construct FP-tree (prefix tree, Trie), the data structure of FP-growth



# Mining Frequent Patterns without Candidate Generation

---

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!

# FP-Growth

$\text{Sup}_{\min} = 2$

<i>TID</i>	<i>Items bought</i>
10	{f, a, c, d, g, i, m, p}
20	{a, b, c, f, l, m, o}
30	{b, f, h, j, o, w}
40	{b, c, k, s, p}
50	{a, f, c, e, l, p, m, n}

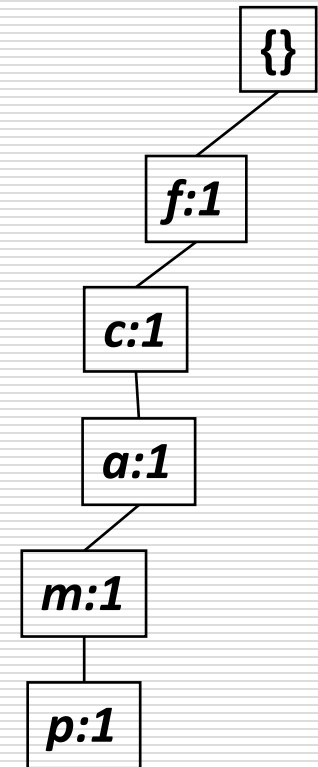
## Header Table

### *Item frequency*

<i>f</i>	4
<i>c</i>	4
<i>a</i>	3
<i>b</i>	3
<i>m</i>	3
<i>p</i>	3



<i>TID</i>	<i>(ordered) frequent items</i>
10	{f, c, a, m, p}
20	{f, c, a, b, m}
30	{f, b}
40	{c, b, p}
50	{f, c, a, m, p}



# FP-Growth

$\text{Sup}_{\min} = 2$

<i>TID</i>	<i>Items bought</i>
10	{f, a, c, d, g, i, m, p}
20	{a, b, c, f, l, m, o}
30	{b, f, h, j, o, w}
40	{b, c, k, s, p}
50	{a, f, c, e, l, p, m, n}

## Header Table

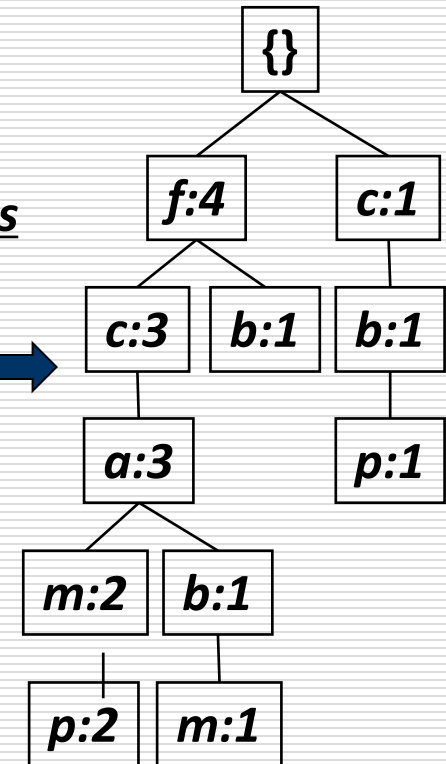
### *Item frequency*

<i>f</i>	4
<i>c</i>	4
<i>a</i>	3
<i>b</i>	3
<i>m</i>	3
<i>p</i>	3



### *TID (ordered) frequent items*

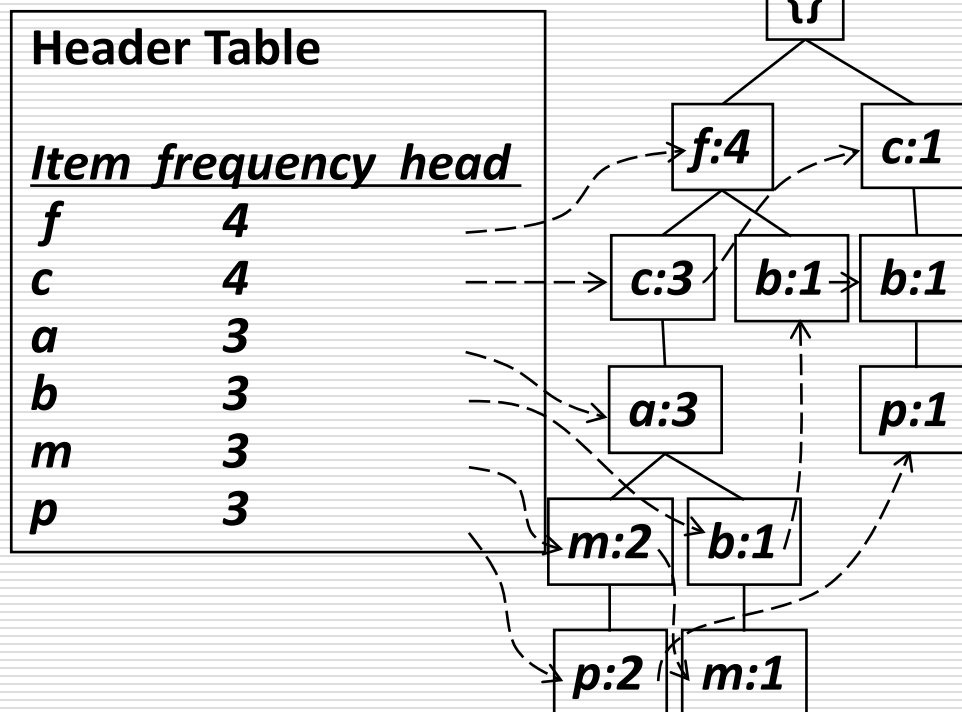
10	{f, c, a, m, p}
20	{f, c, a, b, m}
30	{f, b}
40	{c, b, p}
50	{f, c, a, m, p}



# FP-Growth

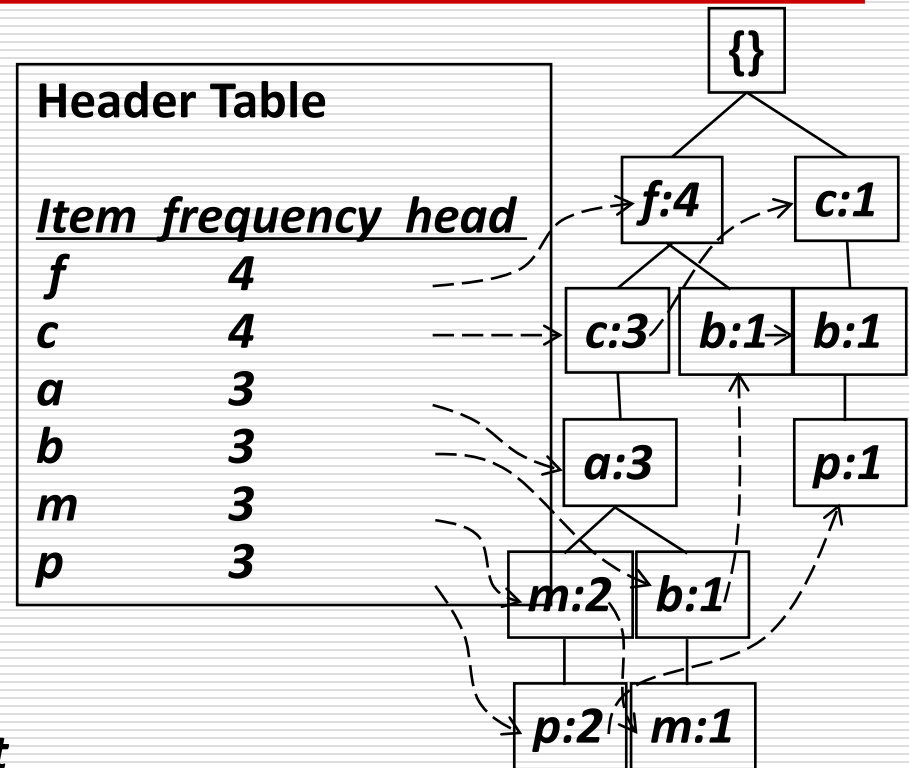
$$\text{Sup}_{\min} = 2$$

<i>TID</i>	<i>(ordered) frequent items</i>
10	{ <i>f</i> , <i>c</i> , <i>a</i> , <i>m</i> , <i>p</i> }
20	{ <i>f</i> , <i>c</i> , <i>a</i> , <i>b</i> , <i>m</i> }
30	{ <i>f</i> , <i>b</i> }
40	{ <i>c</i> , <i>b</i> , <i>p</i> }
50	{ <i>f</i> , <i>c</i> , <i>a</i> , <i>m</i> , <i>p</i> }



# FP-Growth

$$\text{Sup}_{\min} = 2$$



## Conditional pattern bases

Item   cond. pattern base   freq. itemset

<i>p</i>	<i>fcam:2, cb:1</i>	<i>fp, cp, ap, mp, fcp, fap, fmp, cap, cmp, amp, camp, facp, fcmp, famp, fcamp</i>
<i>m</i>	<i>fca:2, fcab:1</i>	<i>fm, cm, am, fcm, fam, cam, fcam</i>
<i>b</i>	<i>fca:1, f:1, c:1</i>	...
<i>a</i>	<i>fc:3</i>	...
<i>c</i>	<i>f:3</i>	...



# Why Is Frequent Pattern Growth Fast?

---

- The performance study shows
  - FP-growth is faster than Apriori (in most cases), and is also faster than tree-projection (an order of magnitude on some datasets)
- Reasoning
  - No candidate generation (claimed by the authors)
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operation is counting and FP-tree building

# Extension of Association Rule Mining

---

- ❑ Association rule mining has been extensively studied in the data mining community.
- ❑ There are many efficient algorithms and model variations.
- ❑ Other related work includes
  - Multi-level or generalized rule mining
  - Sequential pattern mining
  - Constrained rule mining
  - Incremental rule mining
  - Maximal and closed frequent itemset mining
  - Numeric association rule mining
  - Rule interestingness and visualization
  - Parallel algorithms
  - ...

# Extension of Association Rule Mining

---

- ❑ Association rule mining has been extensively studied in the data mining community.
- ❑ There are many efficient algorithms and model variations.
- ❑ Other related work includes
  - Multi-level or generalized rule mining
  - [Sequential pattern mining](#)
  - Constrained rule mining
  - Incremental rule mining
  - Maximal and closed frequent itemset mining
  - Numeric association rule mining
  - Rule interestingness and visualization
  - Parallel algorithms
  - ...

# Applications

Web Images Maps News Shopping Gmail more ▾

Google **Q1** sequence mining **P1** **Q2** sequential pattern mining

Web

**Sequence mining** - Wikipedia, the free encyclopedia  
Sequence mining is concerned with finding statistically relevant patterns between data examples where the values are delivered in a sequence. ...  
[en.wikipedia.org/wiki/Sequence\\_mining](http://en.wikipedia.org/wiki/Sequence_mining) - 15k - [Cached](#) - [Similar pages](#)

**Publications by Mohammed Javeed Zaki**  
Mohammed J. Zaki, **Sequence Mining** in Categorical Domains: Algorithms and ...  
Mohammed J. Zaki, **Parallel Sequence Mining** on Shared-Memory Machines, ...  
[www.cs.rpi.edu/~zaki/papers.html](http://www.cs.rpi.edu/~zaki/papers.html) - 66k - [Cached](#) - [Similar pages](#)

**[PDF] 7. Sequence Mining**  
File Format: Microsoft Powerpoint - [View as HTML](#)  
**Sequence Mining**. Sequences and Strings. Recognition with Strings. MM & HMM.  
**Sequence Association Rules**. 7/03. **Data Mining** - Sequences ...  
[www.cs.wright.edu/~gdong/mining03/course\\_slides/C7Seq.ppt](http://www.cs.wright.edu/~gdong/mining03/course_slides/C7Seq.ppt) - [Similar pages](#)

**Incremental and Interactive Sequence Mining** - Parthasarathy ...  
Interactive **Sequence Discovery** by Incremental **Mining** - Ming-Yen Lin And (Correct) ... 0.3:  
**Parallel Sequence Mining** on Shared-Memory Machines - Zaki (2000) ...  
[citeseer.ist.psu.edu/235156.html](http://citeseer.ist.psu.edu/235156.html) - 25k - [Cached](#) - [Similar pages](#)

**Sequence Mining** in Categorical Domains: Incorporating Constraints ...  
We present cSPADE, an efficient algorithm for **mining** frequent sequences considering a variety of syntactic constraints.  
[citeseer.ist.psu.edu/616916.html](http://citeseer.ist.psu.edu/616916.html) - 24k - [Cached](#) - [Similar pages](#)  
[More results from citeseer.ist.psu.edu »](#)

Web Images Maps News Shopping Gmail more ▾

Google **P2** sequential pattern mining

Web

**SPAM: Sequential Pattern Mining**  
SPAM: **Sequential Pattern Mining**. SPAM is a new algorithm for finding all frequent sequences within a transactional database. The algorithm is especially ...  
[himalaya-tools.sourceforge.net/Spam/](http://himalaya-tools.sourceforge.net/Spam/) - 7k - [Cached](#) - [Similar pages](#)

**[PDF] Multi-dimensional Sequential Pattern Mining**  
File Format: PDF/Adobe Acrobat - [View as HTML](#)  
Multi-dimensional **Sequential Pattern Mining**. Helen Pinto. Jiawei Han. Jian Pei .....  
**SEQUENTIAL PATTERN MINING**. "m1Z@h"kn"r4j&o1l(mb f({Hm1r,h"ka l(m1AkEnAo ...  
[www-sal.cs.uiuc.edu/~hanj/pdf/mdseq01.pdf](http://www-sal.cs.uiuc.edu/~hanj/pdf/mdseq01.pdf) - [Similar pages](#)

**[PDF] Sequential Pattern Mining Algorithms: Trade-offs between Speed and ...**  
File Format: PDF/Adobe Acrobat - [View as HTML](#)  
In this paper, we study the problem of **sequential pattern mining**, ... goal of **sequential pattern mining** is to discover all frequent sequences of itemsets in ...  
[hms.liacs.nl/mgts2004/papers/antunes.pdf](http://hms.liacs.nl/mgts2004/papers/antunes.pdf) - [Similar pages](#)

**[PDF] Sequential Pattern Mining: A Survey**  
File Format: PDF/Adobe Acrobat - [View as HTML](#)  
those techniques web **mining** and **sequential pattern mining** are also well ..... —  
**Sequential patterns: sequential pattern mining** is trying to find the ...  
[www.icar.cnr.it/.../2006/datamining/Esami2006/ArticoliSelezionatiDM/SEMINARI/Mining%20Data%20Streams/SQM.pdf](http://www.icar.cnr.it/.../2006/datamining/Esami2006/ArticoliSelezionatiDM/SEMINARI/Mining%20Data%20Streams/SQM.pdf) - [Similar pages](#)

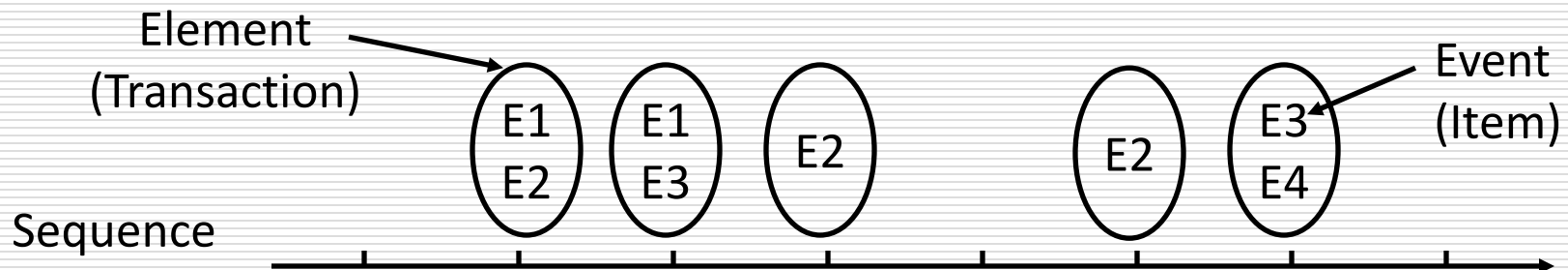
**Mining Sequential Patterns by Pattern-Growth: The PrefixSpan ...**  
**Sequential pattern mining** is an important data **mining** problem with broad applications. However, it is also a difficult problem since the **mining** may have to ...  
[citeseer.ist.psu.edu/655475.html](http://citeseer.ist.psu.edu/655475.html) - 25k - [Cached](#) - [Similar pages](#)

**Mining Sequential Patterns** - Agrawal, Srikant (ResearchIndex)  
We introduce the problem of **mining sequential patterns** over such databases. We present three algorithms to solve this problem, and empirically evaluate ...  
[citeseer.ist.psu.edu/agrawal95mining.html](http://citeseer.ist.psu.edu/agrawal95mining.html) - 25k - [Cached](#) - [Similar pages](#)

# Examples of Sequence Data

---

Sequence Database	Sequence	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time $t$	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor after a single mouse click	Home page, index page, contact info, etc
Event data	History of events generated by a given sensor	Events triggered by a sensor at time $t$	Types of alarms generated by sensors
Genome sequences	DNA sequence of a particular species	An element of the DNA sequence	Bases A,T,G,C



# Formal Definition of a Sequence

---

- A sequence is an ordered list of elements (transactions)

$$S = \langle e_1 e_2 e_3 \dots \rangle$$

- Each element contains a collection of items

$$e_i = \{i_1, i_2, \dots, i_k\}$$

- Each element is attributed to a specific time

- A k-sequence is a sequence that contains k items

# Formal Definition of a Subsequence

---

- A sequence  $\langle a_1 a_2 \dots a_n \rangle$  is contained in another sequence  $\langle b_1 b_2 \dots b_m \rangle$  ( $m \geq n$ ) if there exist integers  $i_1 < i_2 < \dots < i_n$  such that  $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots, a_n \subseteq b_{i_n}$

Data sequence	Subsequence	Contain?
$\langle \{2,4\} \{3,5,6\} \{8\} \rangle$	$\langle \{2\} \{3,5\} \rangle$	Yes
$\langle \{1,2\} \{3,4\} \rangle$	$\langle \{1\} \{2\} \rangle$	No
$\langle \{2,4\} \{2,4\} \{2,5\} \rangle$	$\langle \{2\} \{4\} \rangle$	Yes

- The support of a subsequence  $w$  is defined as the fraction of data sequences that contain  $w$
- A *sequential pattern* is a frequent subsequence (i.e., a subsequence whose support is  $\geq \text{minsup}$ )

# Sequential Pattern Mining: Definition

---

## □ Given:

- a database of sequences
- a user-specified minimum support threshold, *minsup*

## □ Task:

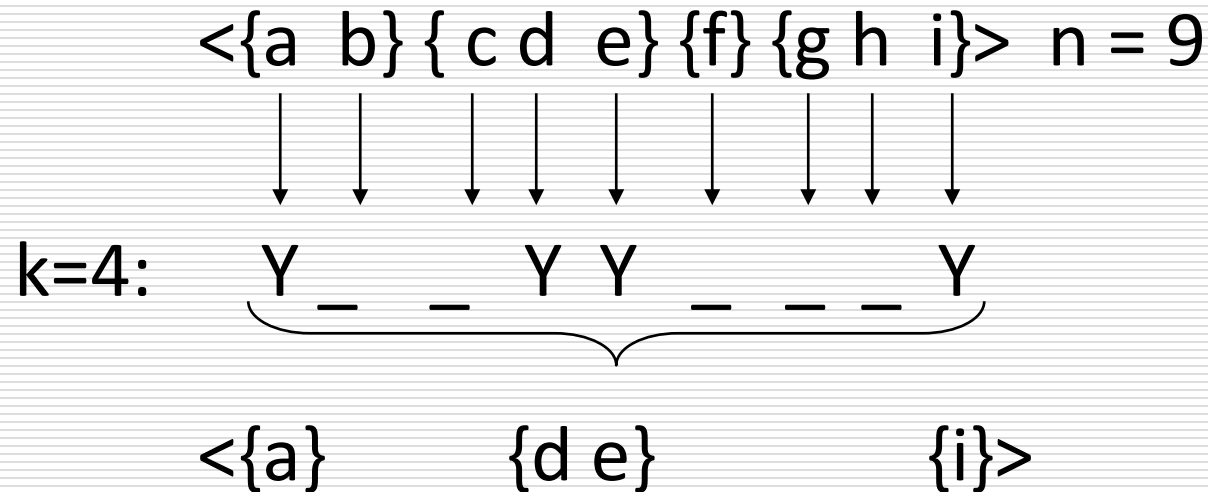
- Find all subsequences with support  $\geq \textit{minsup}$



# Sequential Pattern Mining: Challenge

---

- How many k-subsequences can be extracted from a given n-sequence?



Answer :

$$\binom{n}{k} = \binom{9}{4} = 126$$

# Outline

---

- Association Rules
- Frequent Itemset Mining Algorithms
- Sequential Pattern Mining Algorithms
  - GSP
  - SPADE
  - SPAM

# GSP (Generalized Sequential Pattern Mining)

---

- ❑ Proposed by Srikant and Agrawal [*EDBT'96*]
- ❑ Uses the Apriori pruning principle

# Finding Length-1 Sequential Patterns

---

## □ Initial candidates:

■  $\langle a \rangle, \langle b \rangle, \langle c \rangle, \langle d \rangle, \langle e \rangle, \langle f \rangle, \langle g \rangle, \langle h \rangle$

## □ Scan database once, count support for candidates

$min\_sup = 2$

Seq. ID	Sequence
10	$\langle (bd)cb(ac) \rangle$
20	$\langle (bf)(ce)b(fg) \rangle$
30	$\langle (ah)(bf)abf \rangle$
40	$\langle (be)(ce)d \rangle$
50	$\langle a(bd)bcb(ade) \rangle$

Cand	Sup
$\langle a \rangle$	3
$\langle b \rangle$	5
$\langle c \rangle$	4
$\langle d \rangle$	3
$\langle e \rangle$	3
$\langle f \rangle$	2
<del><math>\langle g \rangle</math></del>	1
<del><math>\langle h \rangle</math></del>	1

# Generating Length-2 Candidates

---

51 length-2  
Candidates

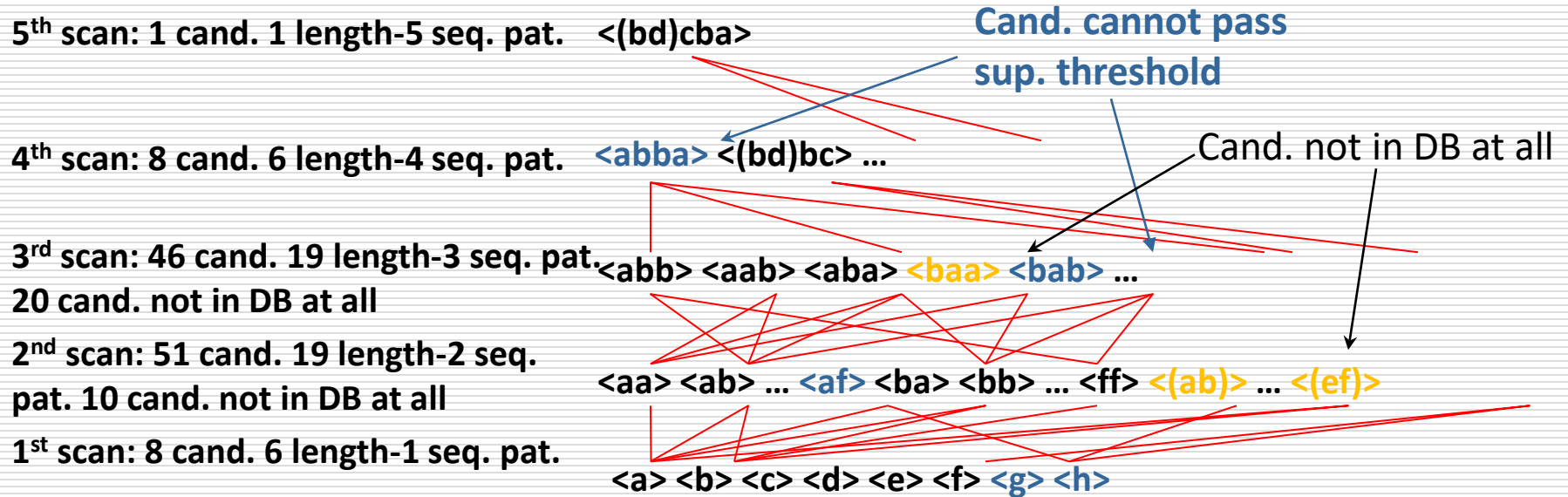
	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<aa>	<ab>	<ac>	<ad>	<ae>	<af>
<b>	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
<c>	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
<d>	<da>	<db>	<dc>	<dd>	<de>	<df>
<e>	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
<f>	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<(ab)>	<(ac)>	<(ad)>	<(ae)>	<(af)>
<b>			<(bc)>	<(bd)>	<(be)>	<(bf)>
<c>				<(cd)>	<(ce)>	<(cf)>
<d>					<(de)>	<(df)>
<e>						<(ef)>
<f>						

Without Apriori  
property,  
 $8*8+8*7/2=92$   
candidates

Apriori prunes  
44.57%  
candidates

# GSP Mining Process



*min\_sup = 2*

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

# GSP Algorithm

---

- ❑ Take sequences in form of  $\langle x \rangle$  as length-1 candidates
- ❑ Scan database once, find  $F_1$ , the set of length-1 sequential patterns
- ❑ Let  $k=1$ ; while  $F_k$  is not empty do
  - Form  $C_{k+1}$ , the set of length- $(k+1)$  candidates from  $F_k$ ;
  - If  $C_{k+1}$  is not empty, scan database once, find  $F_{k+1}$ , the set of length- $(k+1)$  sequential patterns
  - Let  $k=k+1$ ;

# GSP Algorithm

---

- Benefits from the Apriori pruning
  - Reduces search space
- Bottlenecks
  - Scans the database multiple times
  - Generates a huge set of candidate sequences



# SPADE Algorithm

---

# SPADE Algorithm

---

- ❑ Proposed by Zaki *et al.* [*MLJ'01*]
- ❑ Candidate generation-and-test
- ❑ Vertical ID-list data representation based on Lattice-theory
- ❑ Counting support through temporal joins
- ❑ Reduced I/O costs (three DB scans)

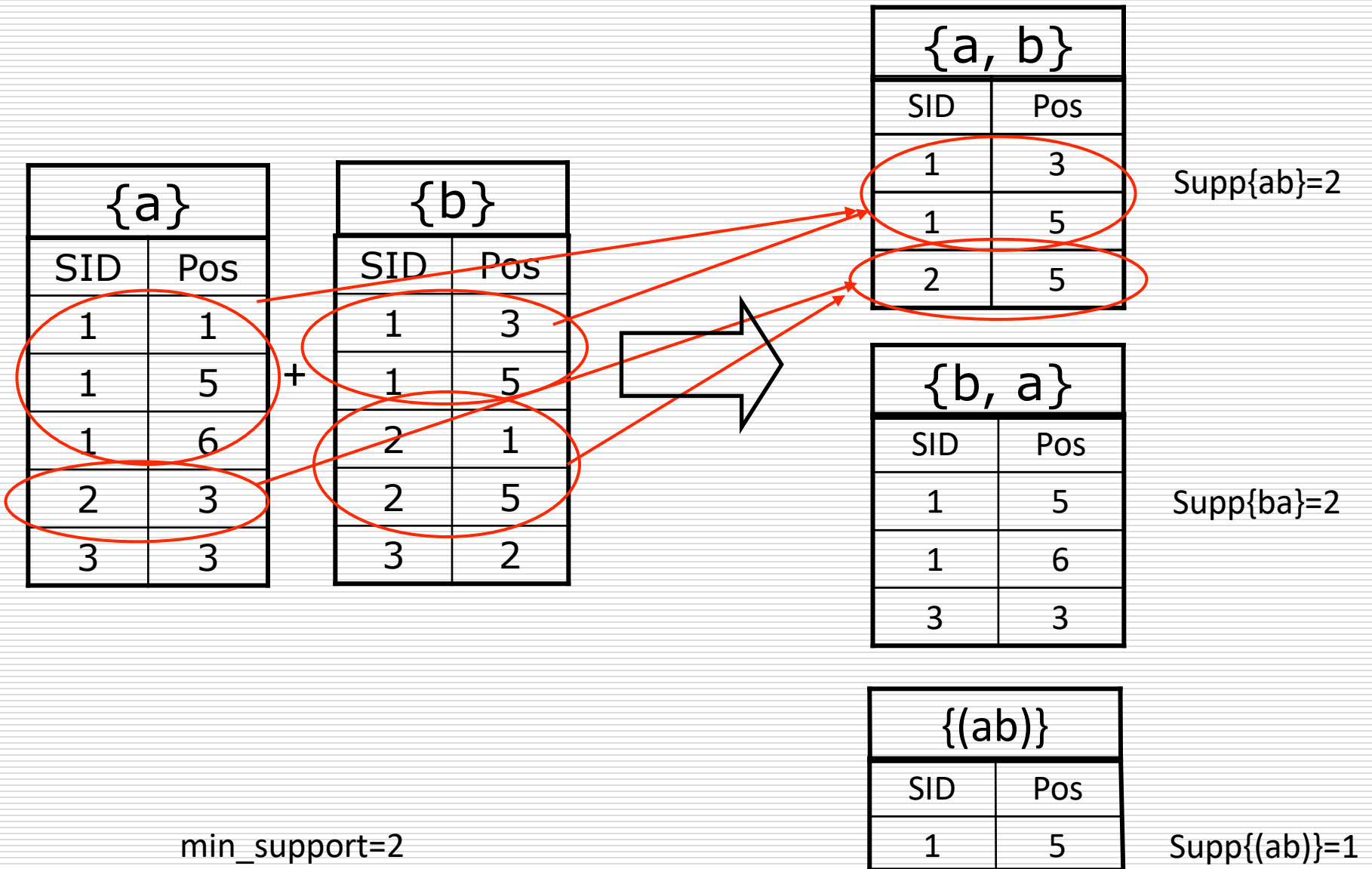
# SPADE Algorithm

ID	Data Sequence
1	< a c ( b c ) d ( a b c ) a d >
2	< b ( c d ) a c ( b d ) >
3	< d ( b c ) ( a c ) ( c d ) >

ID-List DB

a		b		c		d	
SID	Pos	SID	Pos	SID	Pos	SID	Pos
1	1	1	3	1	2	1	4
1	5	1	5	1	3	1	7
1	6	2	1	1	5	2	2
2	3	2	5	2	2	2	5
3	3	3	2	2	4	3	1
				3	2	3	4
				3	3		
				3	4		

# Temporal Joins



# SPAM Algorithm

---

# SPAM Algorithm

---

- ❑ Proposed by Ayres *et al.* [*KDD'02*]
- ❑ Key idea based on SPADE
- ❑ Using bitmap data representation
- ❑ Faster than SPADE yet space consuming

# SPAM Algorithm

---

ID	Data Sequence
10	<a c (b c) d (a b c) a d>
20	<b (c d) a c (b d)>
30	<d (b c) (a c) (c d)>

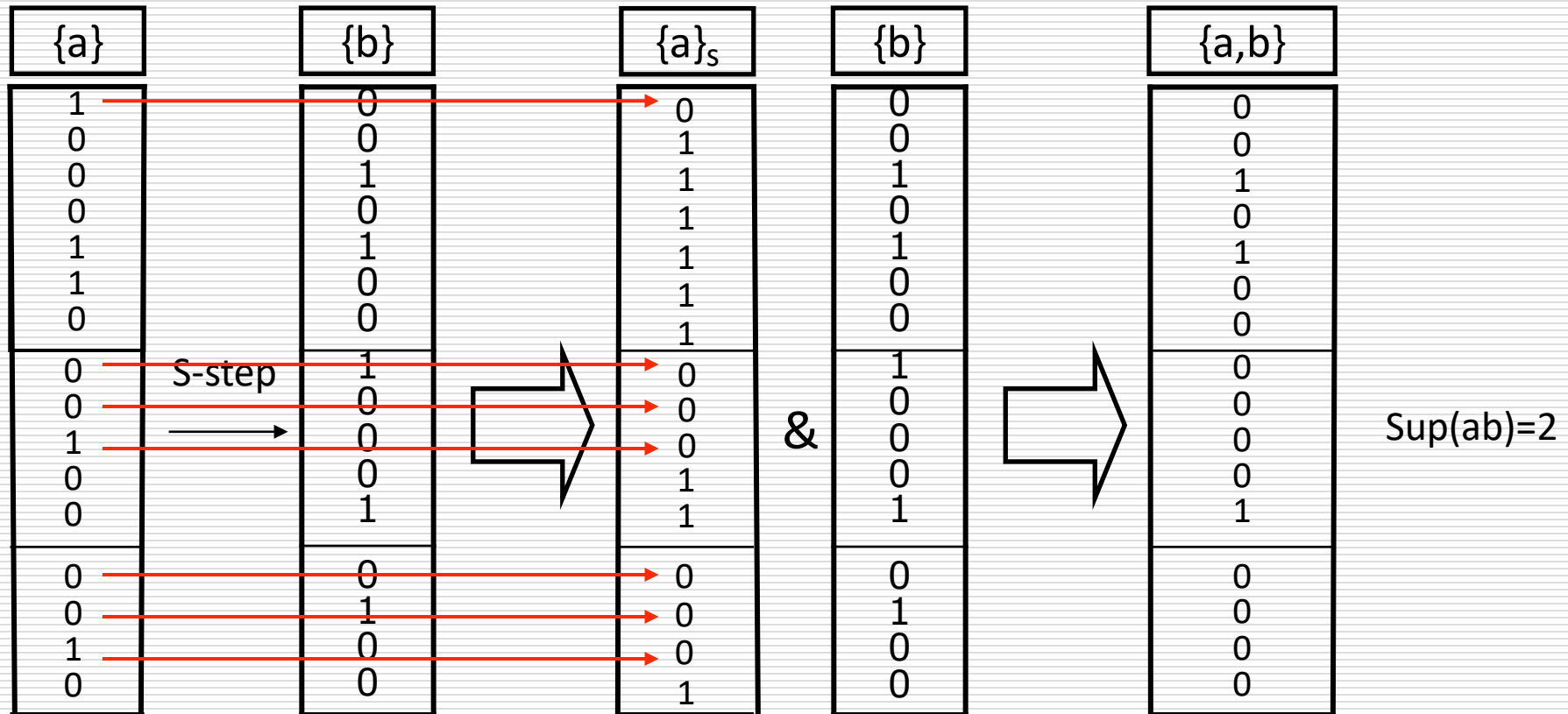
SID EID

10 1  
10 2  
10 3  
10 4  
10 5  
10 6  
10 7  
20 1  
20 2  
20 3  
20 4  
20 5  
30 1  
30 2  
30 3  
30 4

{a}	{b}	{c}	{d}
1	0	0	0
0	0	1	0
0	1	1	0
0	0	0	1
1	1	1	0
1	0	0	0
0	0	0	1
0	1	0	0
0	0	1	1
0	0	0	1
1	0	1	0
0	1	1	0
0	0	1	1

# SPAM Temporal Joins

Sequence extended step:



min\_support=2



# Problem of SPAM

---

- Bitmap representation is space consuming  
i.e., data is commonly very sparse

# Acknowledgement

---

- Slides are adapted from:
  - Prof. Jeffrey D. Ullman
  - Dr. Jure Leskovec
  - Dr. Wujun Li

# 课堂问题

---

TID	Items
1	{a, d, e}
2	{a, b, c, e}
3	{a, b, d}
4	{a, c, d, e}
5	{b, c, e}

Support(b,c) = ?

Confidence({e}  $\rightarrow$  {b,c}) = ?

需要注明：(1) 姓名；(2) 学号；(3) 专业