CSE 5243 INTRO. TO DATA MINING

Advanced Frequent Pattern Mining

&

Locality Sensitivity Hashing

Huan Sun, CSE@The Ohio State University

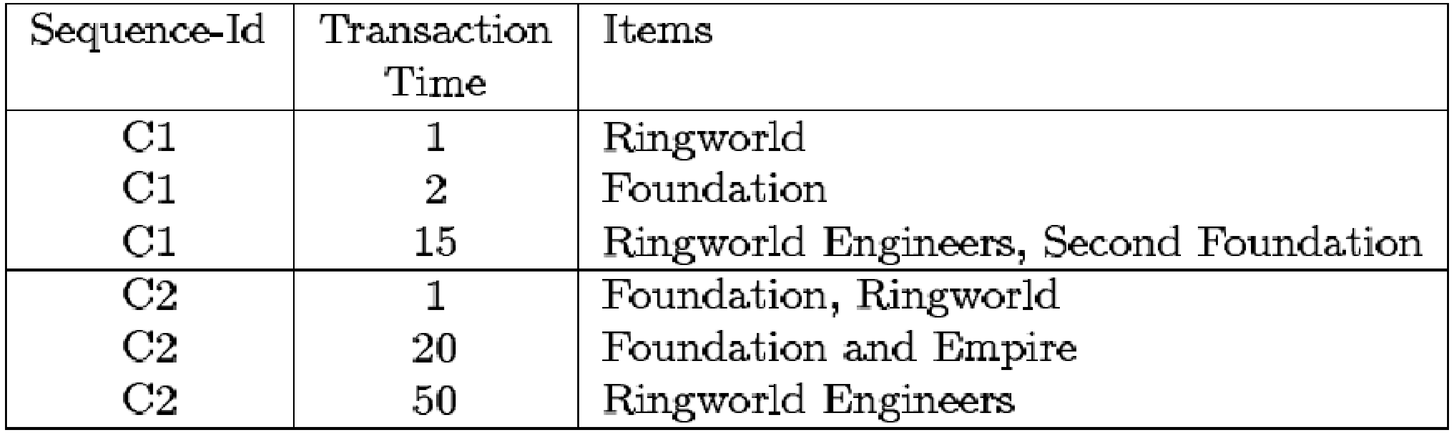
11/07/2017

Slides adapted from Prof. Jiawei Han @UIUC, Prof. Srinivasan Parthasarathy @OSU

Sequence Mining: Description

- Input
  - A database ***D*** of sequences called *data-sequences,* in which:
    - $I=\{i_1, i_2,\ldots,i_n\}$ is the set of items
    - each sequence is a list of transactions ordered by transaction-time
    - each transaction consists of fields: sequence-id, transaction-id, transaction-time and a set of items.

Database $\mathcal{D}$



Sequence-Id	Transaction Time	Items
C1	1	Ringworld
C1	2	Foundation
C1	15	Ringworld Engineers, Second Foundation
C2	1	Foundation, Ringworld
C2	20	Foundation and Empire
C2	50	Ringworld Engineers

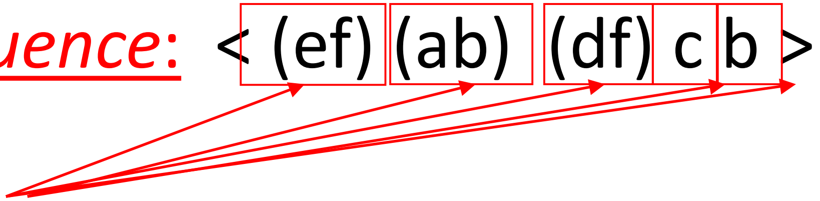
Sequential Pattern and Sequential Pattern Mining

- Sequential pattern mining: Given a set of sequences, find the complete set of *frequent* subsequences (i.e., satisfying the min_sup threshold)

A *sequence database*

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

A *sequence*: < (ef) (ab) (df) c b >



- An element may contain a set of *items* (also called *events*)
- Items within an element are unordered and we list them alphabetically

<a(bc)dc> is a *subsequence* of <a(abc)(ac)d(cf)>

Sequential Pattern and Sequential Pattern Mining

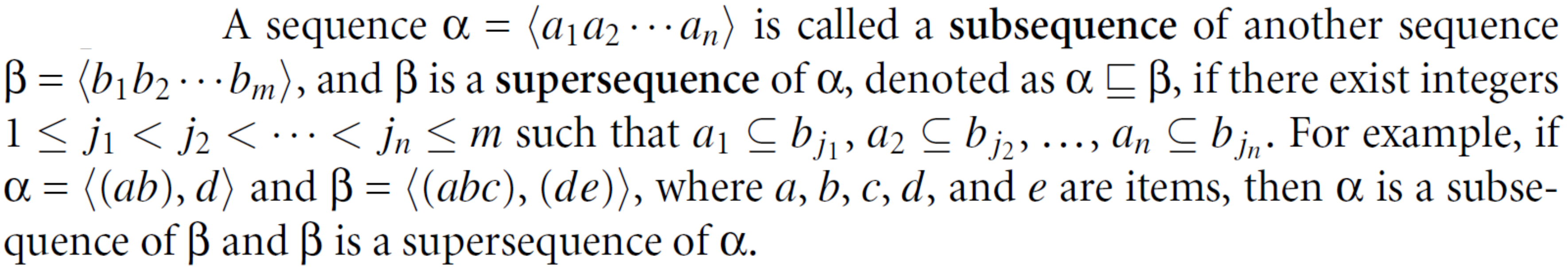
- Sequential pattern mining: Given a set of sequences, find the complete set of *frequent* subsequences (i.e., satisfying the min_sup threshold)

A *sequence database*

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a(bc)dc> is a *subsequence* of <a(abc)(ac)d(cf)>

Formal definition:


A sequence $\alpha = \langle a_1 a_2 \cdots a_n \rangle$ is called a **subsequence** of another sequence $\beta = \langle b_1 b_2 \cdots b_m \rangle$, and β is a **supersequence** of α, denoted as $\alpha \sqsubseteq \beta$, if there exist integers $1 \leq j_1 < j_2 < \cdots < j_n \leq m$ such that $a_1 \subseteq b_{j_1}$, $a_2 \subseteq b_{j_2}$, $\ldots$, $a_n \subseteq b_{j_n}$. For example, if $\alpha = \langle (ab), d \rangle$ and $\beta = \langle (abc), (de) \rangle$, where a, b, c, d, and e are items, then α is a subsequence of β and β is a supersequence of α.

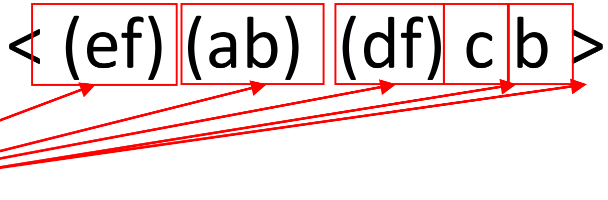
Sequential Pattern and Sequential Pattern Mining

- Sequential pattern mining: Given a set of sequences, find the complete set of *frequent* subsequences (i.e., satisfying the min_sup threshold)

A *sequence database*

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

A *sequence*:


< (ef) (ab) (df) c b >

- An element may contain a set of *items* (also called *events*)
- Items within an element are unordered and we list them alphabetically

<a(bc)dc> is a *subsequence* of <a(abc)(ac)d(cf)>

- Given *support threshold min_sup* = 2, <(ab)c> is a *sequential pattern*

A Basic Property of Sequential Patterns: Apriori

- A basic property: Apriori (Agrawal & Sirkant'94)
  - If a sequence S is not frequent
  - Then none of the super-sequences of S is frequent
  - E.g, <hb> is infrequent → so do <hab> and <(ah)b>

GSP (Generalized Sequential Patterns): Apriori-Based Sequential Pattern Mining

GSP (Generalized Sequential Patterns): Srikant & Agrawal @ EDBT'96)

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

- Initial candidates: All 8-singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan DB once, count support for each candidate

min_sup = 2

Cand.	sup
<a>	3
<b>	5
<c>	4
<d>	3
<e>	3
<f>	2
~~<g>~~	1
~~<h>~~	1

GSP (Generalized Sequential Patterns): Apriori-Based Sequential Pattern Mining

GSP (Generalized Sequential Patterns): Srikant & Agrawal @ EDBT'96)

- □ Initial candidates: All 8-singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- □ Scan DB once, count support for each candidate
- □ Generate length-2 candidate sequences

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

How?

GSP (Generalized Sequential Patterns): Apriori-Based Sequential Pattern Mining

GSP (Generalized Sequential Patterns): Srikant & Agrawal @ EDBT'96)

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

- Initial candidates: All 8-singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan DB once, count support for each candidate
- Generate length-2 candidate sequences

	<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>

Why?

GSP (Generalized Sequential Patterns): Apriori-Based Sequential Pattern Mining

GSP (Generalized Sequential Patterns): Srikant & Agrawal @ EDBT'96)

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

- □ Initial candidates: All 8-singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- □ Scan DB once, count support for each candidate
- □ Generate length-2 candidate sequences

	<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>						

Why?

GSP (Generalized Sequential Patterns): Apriori-Based Sequential Pattern Mining

GSP (Generalized Sequential Patterns): Srikant & Agrawal @ EDBT'96)

- □ Initial candidates: All 8-singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- □ Scan DB once, count support for each candidate
- □ Generate length-2 candidate sequences

	<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>						

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

- ❑ Without Apriori pruning:

  (8 singletons) 8*8+8*7/2 = 92 length-2 candidates
- ❑ With pruning, length-2 candidates: 36 + 15= 51

GSP Mining and Pruning

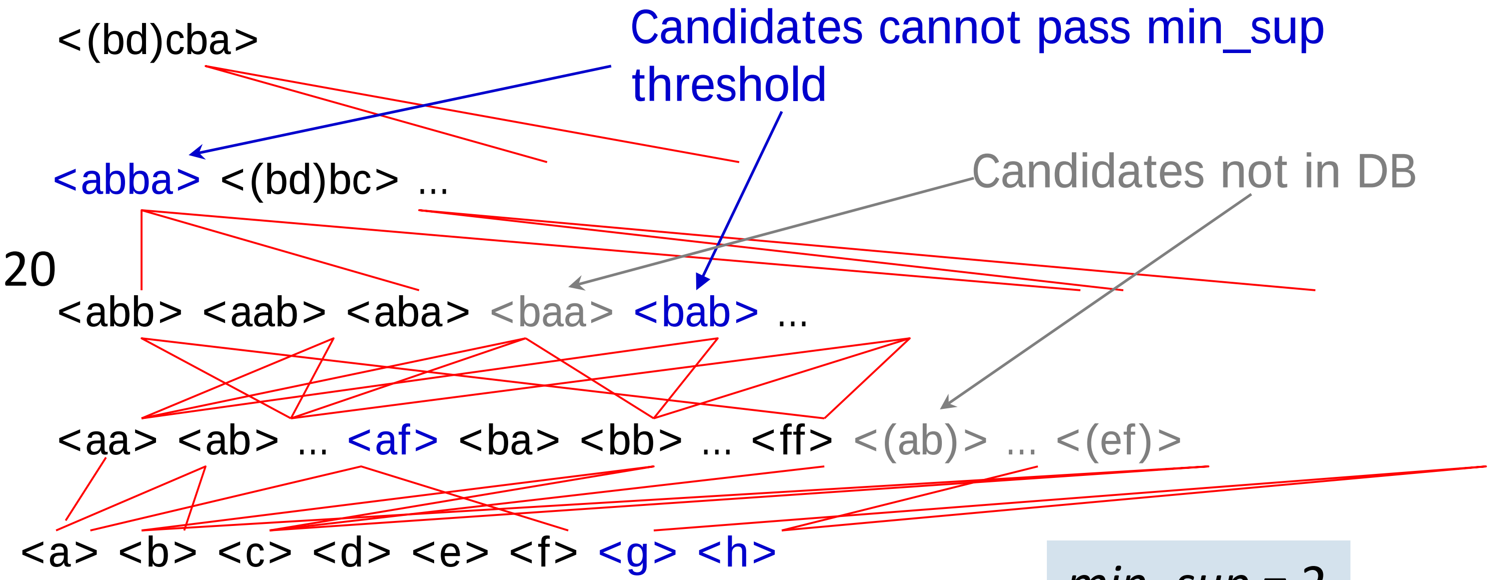
5th scan: 1 cand. 1 length-5 seq. pat.

4th scan: 8 cand. 7 length-4 seq. pat.

3rd scan: 46 cand. 20 length-3 seq. pat. 20 cand. not in DB at all

2nd scan: 51 cand. 19 length-2 seq. pat. 10 cand. not in DB at all

1st scan: 8 cand. 6 length-1 seq. pat.



<(bd)cba>

<abba> <(bd)bc> …

<abb> <aab> <aba> <baa> <bab> …

<aa> <ab> … <af> <ba> <bb> … <ff> <(ab)> … <(ef)>

<a> <b> <c> <d> <e> <f> <g> <h>

Candidates cannot pass min_sup threshold

Candidates not in DB

min_sup = 2

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

GSP Mining and Pruning

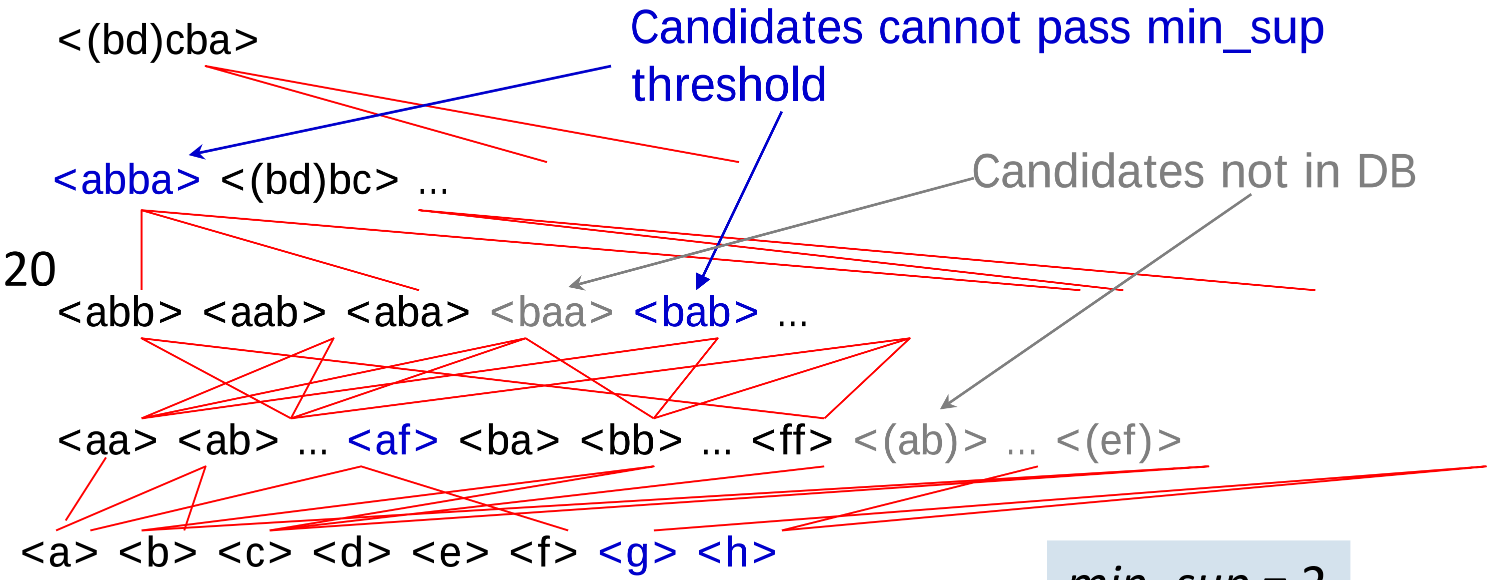
5th scan: 1 cand. 1 length-5 seq. pat.

4th scan: 8 cand. 7 length-4 seq. pat.

3rd scan: 46 cand. 20 length-3 seq. pat. 20 cand. not in DB at all

2nd scan: 51 cand. 19 length-2 seq. pat. 10 cand. not in DB at all

1st scan: 8 cand. 6 length-1 seq. pat.



<(bd)cba>

Candidates cannot pass min_sup threshold

<abba> <(bd)bc> ...

Candidates not in DB

<abb> <aab> <aba> <baa> <bab> ...

<aa> <ab> ... <af> <ba> <bb> ... <ff> <(ab)> ... <(ef)>

<a> <b> <c> <d> <e> <f> <g> <h>

- Repeat (for each level (i.e., length-k))
  - Scan DB to find length-k frequent sequences
  - Generate length-(k+1) candidate sequences from length-k frequent sequences using Apriori
  - set k = k+1
- Until no frequent sequence or no candidate can be found

min_sup = 2

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

Mining Sequential Patterns: Generalizations and Performance Improvements, Srikant and Agrawal et al. https://pdfs.semanticscholar.org/d420/ea39dc136b9e390d05e964488a65fcf6ad33.pdf

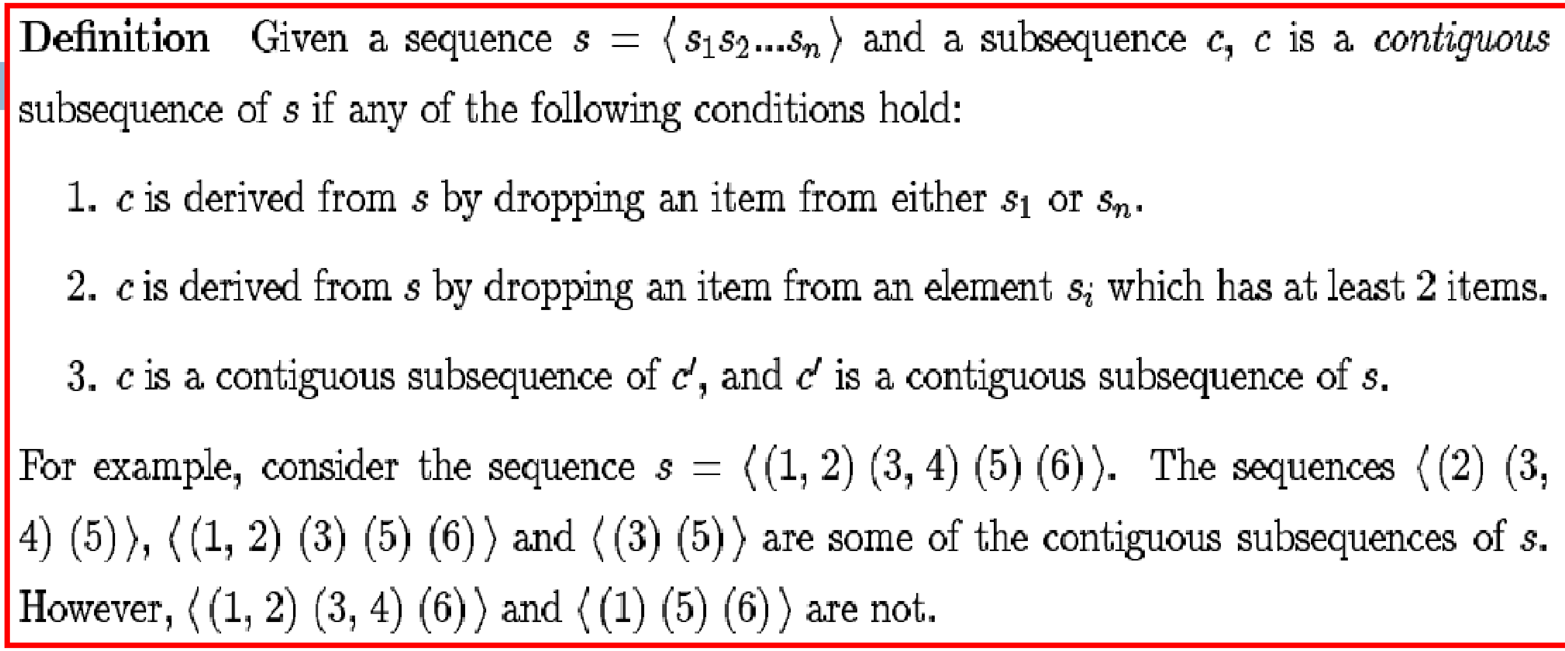
GSP: Algorithm

- **Phase 1:**
  - Scan over the database to identify all the frequent items, i.e., 1-element sequences

- **Phase 2:**
  - Iteratively scan over the database to discover all frequent sequences. Each iteration discovers all the sequences with the same length.
  - In the iteration to generate all *k*-sequences
    - Generate the set of all candidate *k*-sequences, $\mathbf{C}_k$, by joining two (*k-1*)-sequences
    - Prune the candidate sequence if any of its *k-1* contiguous subsequence is not frequent
    - Scan over the database to determine the support of the remaining candidate sequences
  - Terminate when no more frequent sequences can be found

A detailed illustration: http://simpledatamining.blogspot.com/2015/03/generalized-sequential-pattern-gsp.html

GSP: Algorithm



Definition Given a sequence $s = \langle s_1 s_2 ... s_n \rangle$ and a subsequence c, c is a *contiguous* subsequence of s if any of the following conditions hold:

1. c is derived from s by dropping an item from either s_1 or s_n.
2. c is derived from s by dropping an item from an element s_i which has at least 2 items.
3. c is a contiguous subsequence of c', and c' is a contiguous subsequence of s.

For example, consider the sequence $s = \langle (1, 2)\ (3, 4)\ (5)\ (6) \rangle$. The sequences $\langle (2)\ (3,$ $4)\ (5) \rangle$, $\langle (1, 2)\ (3)\ (5)\ (6) \rangle$ and $\langle (3)\ (5) \rangle$ are some of the contiguous subsequences of s. However, $\langle (1, 2)\ (3, 4)\ (6) \rangle$ and $\langle (1)\ (5)\ (6) \rangle$ are not.

- In the iteration to generate all *k*-sequences
  - Generate the set of all candidate *k*-sequences, **C***ₖ*, by joining two (*k-1*)-sequences
  - Prune the candidate sequence if any of its *k-1* contiguous subsequence is not frequent
  - Scan over the database to determine the support of the remaining candidate sequences
- Terminate when no more frequent sequences can be found

Bottlenecks of GSP

- A huge set of candidates could be generated
  - 1,000 frequent length-1 sequences generate length-2 candidates!

$$1000 \times 1000 + \frac{1000 \times 999}{2} = 1{,}499{,}500$$

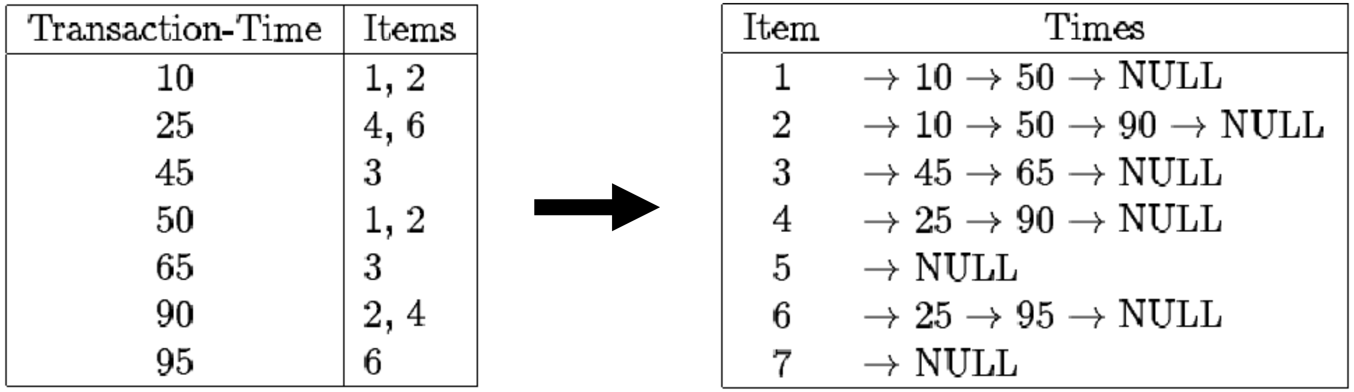
- Multiple scans of database in mining

- Real challenge: mining long sequential patterns
  - An exponential number of short candidates
  - A length-100 sequential pattern needs 10^{30} candidate sequences!

$$\sum_{i=1}^{100} \binom{100}{i} = 2^{100} - 1 \approx 10^{30}$$

GSP: Optimization Techniques

- Applied to phase 2: computation-intensive
- Technique 1: the hash-tree data structure
  - Used for counting candidates to reduce the number of candidates that need to be checked
    - Leaf: a list of sequences
    - Interior node: a hash table
- Technique 2: data-representation transformation
  - From horizontal format to vertical format



Transaction-Time	Items
10	1, 2
25	4, 6
45	3
50	1, 2
65	3
90	2, 4
95	6

Item	Times
1	→ 10 → 50 → NULL
2	→ 10 → 50 → 90 → NULL
3	→ 45 → 65 → NULL
4	→ 25 → 90 → NULL
5	→ NULL
6	→ 25 → 95 → NULL
7	→ NULL

SPADE

- **Problems in the GSP Algorithm**
  - Multiple database scans
  - Complex hash structures with poor locality
  - Scale up linearly as the size of dataset increases

- **SPADE: Sequential PAttern Discovery using Equivalence classes**
  - Use a vertical id-list database
  - Prefix-based equivalence classes
  - Frequent sequences enumerated through simple temporal joins
  - Lattice-theoretic approach to decompose search space

- **Advantages of SPADE**
  - 3 scans over the database
  - Potential for in-memory computation and parallelization

Paper Link:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.6042&rep=rep1&type=pdf

MMDS Secs. 3.2-3.4.

Slides adapted from: J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

FINDING SIMILAR ITEMS

Slides also adapted from Prof. Srinivasan Parthasarathy @OSU

Task: Finding Similar Documents

- **Goal: Given a large number (N in the millions or billions) of documents, find "near duplicate" pairs**
- **Applications:**
  - Mirror websites, or approximate mirrors → remove duplicates
  - Similar news articles at many news sites → cluster

Task: Finding Similar Documents

- **Goal: Given a large number (N in the millions or billions) of documents, find "near duplicate" pairs**
- **Applications:**
  - Mirror websites, or approximate mirrors → remove duplicates
  - Similar news articles at many news sites → cluster

What are the challenges?

Task: Finding Similar Documents

- **Goal: Given a large number (N in the millions or billions) of documents, find "near duplicate" pairs**
- **Applications:**
  - Mirror websites, or approximate mirrors → remove duplicates
  - Similar news articles at many news sites → cluster

- **Problems:**
  - Many small pieces of one document can appear out of order in another
  - Too many documents to compare all pairs
  - Documents are so large or so many (scale issues)

Two Essential Steps for Similar Docs

1. ***Shingling:*** Convert documents to sets

2. ***Min-Hashing:*** Convert large sets to short signatures, while preserving similarity

Host of follow up applications

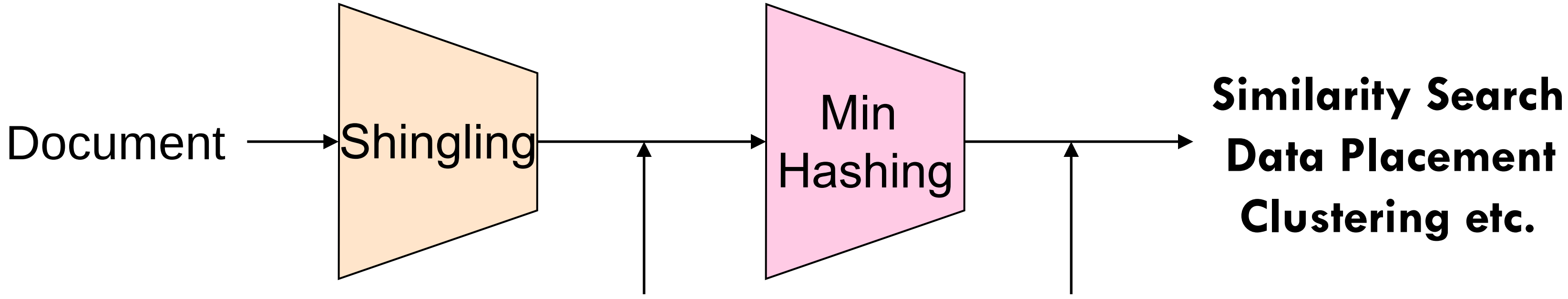
e.g. Similarity Search

Data Placement

Clustering etc.

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

The Big Picture



Document

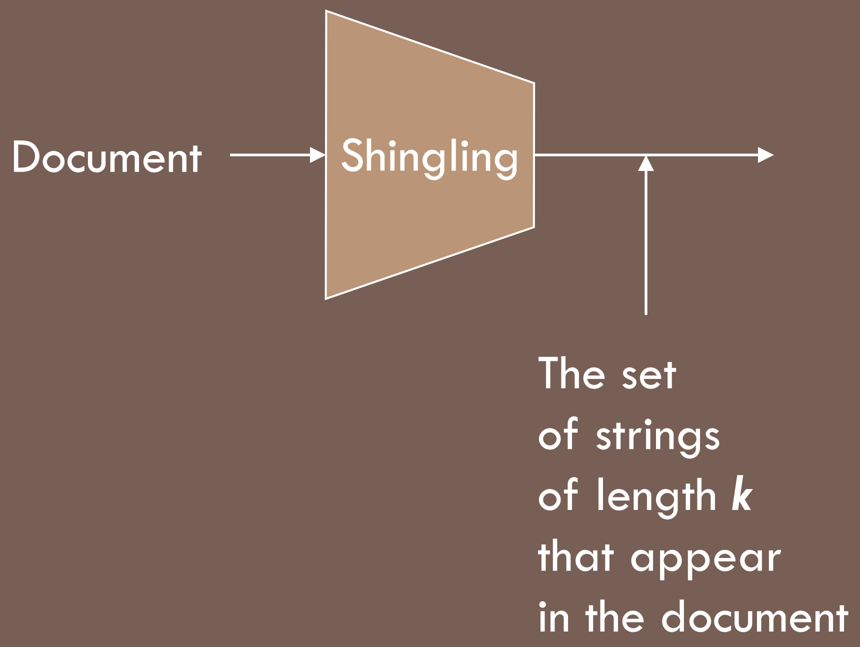
Shingling

Min Hashing

Similarity Search
Data Placement
Clustering etc.

The set of strings of length ***k*** that appear in the document

Signatures: short integer vectors that represent the sets, and reflect their similarity



SHINGLING

Step 1: ***Shingling:*** Convert documents to sets

Documents as High-Dim Data

- **Step 1: *Shingling:* Convert documents to sets**

- **Simple approaches:**
  - Document = set of words appearing in document
  - Document = set of "important" words
  - Don't work well for this application. Why?

- **Need to account for ordering of words!**
- A different way: **Shingles!**

Define: Shingles

- A *k*-shingle (or *k*-gram) for a document is a sequence of *k* tokens that appears in the doc
  - Tokens can be characters, words or something else, depending on the application
  - Assume tokens = characters for examples

Define: Shingles

- A *k*-shingle (or *k*-gram) for a document is a sequence of *k* tokens that appears in the doc
  - Tokens can be characters, words or something else, depending on the application
  - Assume tokens = characters for examples

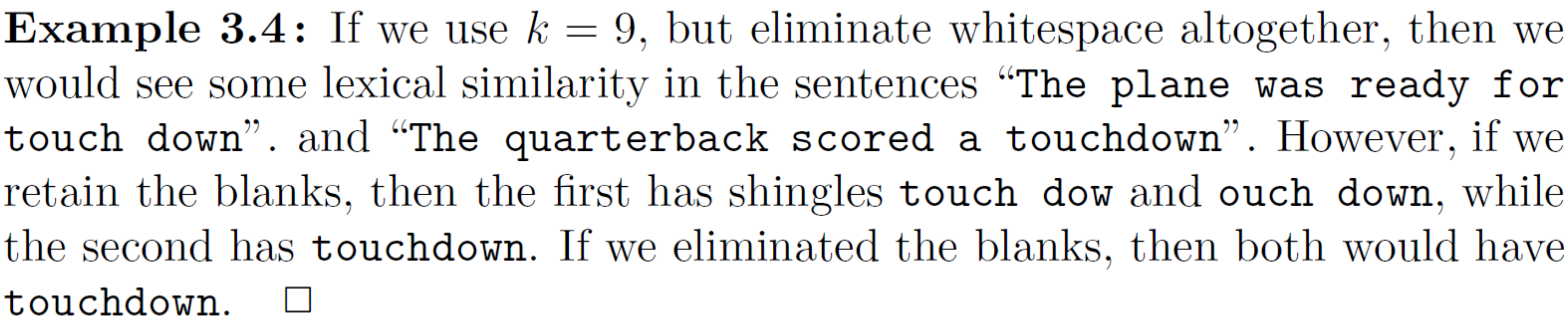
- **Example:** **k=2**; document $\mathbf{D_1}$ = abcab
  Set of 2-shingles: $\mathbf{S(D_1)}$ = {ab, bc, ca}

Define: Shingles

- A *k*-shingle (or *k*-gram) for a document is a sequence of *k* tokens that appears in the doc
  - Tokens can be characters, words or something else, depending on the application
  - Assume tokens = characters for examples

- **Example: k=2**; document **$\mathbf{D_1}$** = abcab
  Set of 2-shingles: **$\mathbf{S(D_1)}$** = {ab, bc, ca}
  - **Another option:** Shingles as a **bag** (multiset), count ab twice: **$\mathbf{S'(D_1)}$ =** {ab, bc, ca, ab}

Shingles: How to treat white-space chars?



Example 3.4: If we use $k = 9$, but eliminate whitespace altogether, then we would see some lexical similarity in the sentences "The plane was ready for touch down". and "The quarterback scored a touchdown". However, if we retain the blanks, then the first has shingles touch dow and ouch down, while the second has touchdown. If we eliminated the blanks, then both would have touchdown. □

It makes sense to replace any sequence of one or more white-space characters (blank, tab, newline, etc.) by a single blank.

This way distinguishes shingles that cover two or more words from those that do not.

How to choose K?

- **Documents that have lots of shingles in common have similar text, even if the text appears in different order**

- **Caveat:** You must pick ***k*** large enough, or most documents will have most shingles
  - ***k*** = 5 is OK for short documents
  - ***k*** = 10 is better for long documents

Compressing Shingles

- To **compress long shingles**, we can **hash** them to (say) 4 bytes
  - Like a Code Book
  - If #shingles manageable → Simple dictionary suffices

e.g., 9-shingle => bucket number [0, 2^32 - 1]

(using 4 bytes instead of 9)

Compressing Shingles

- To **compress long shingles**, we can **hash** them to (say) 4 bytes
  - Like a Code Book
  - If #shingles manageable → Simple dictionary suffices

- **Doc represented by the set of hash/dict. values of its *k*-shingles**
  - **Idea:** Two documents could (rarely) appear to have shingles in common, when in fact only the hash-values were shared

Compressing Shingles

- To **compress long shingles**, we can **hash** them to (say) 4 bytes
  - Like a Code Book
  - If #shingles manageable → Simple dictionary suffices

- **Doc represented by the set of hash/dict. values of its *k*-shingles**

- **Example:** **k=2**; document **D_1**= abcab
  Set of 2-shingles: **$S(D_1)$** = {ab, bc, ca}
  Hash the singles: **$h(D_1)$** = {1, 5, 7}

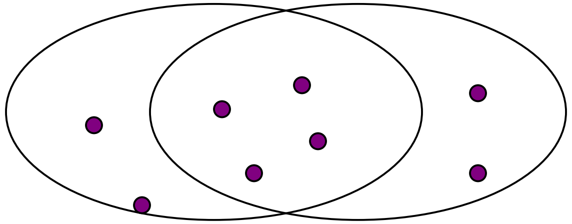
Similarity Metric for Shingles

- **Document D_1 is a set of its k-shingles C_1=S(D_1)**

- Equivalently, each document is a 0/1 vector in the space of *k*-shingles
  - Each unique shingle is a dimension
  - Vectors are very sparse

- **A natural similarity measure is the Jaccard similarity:**

$$sim(D_1, D_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$



Motivation for Minhash/LSH

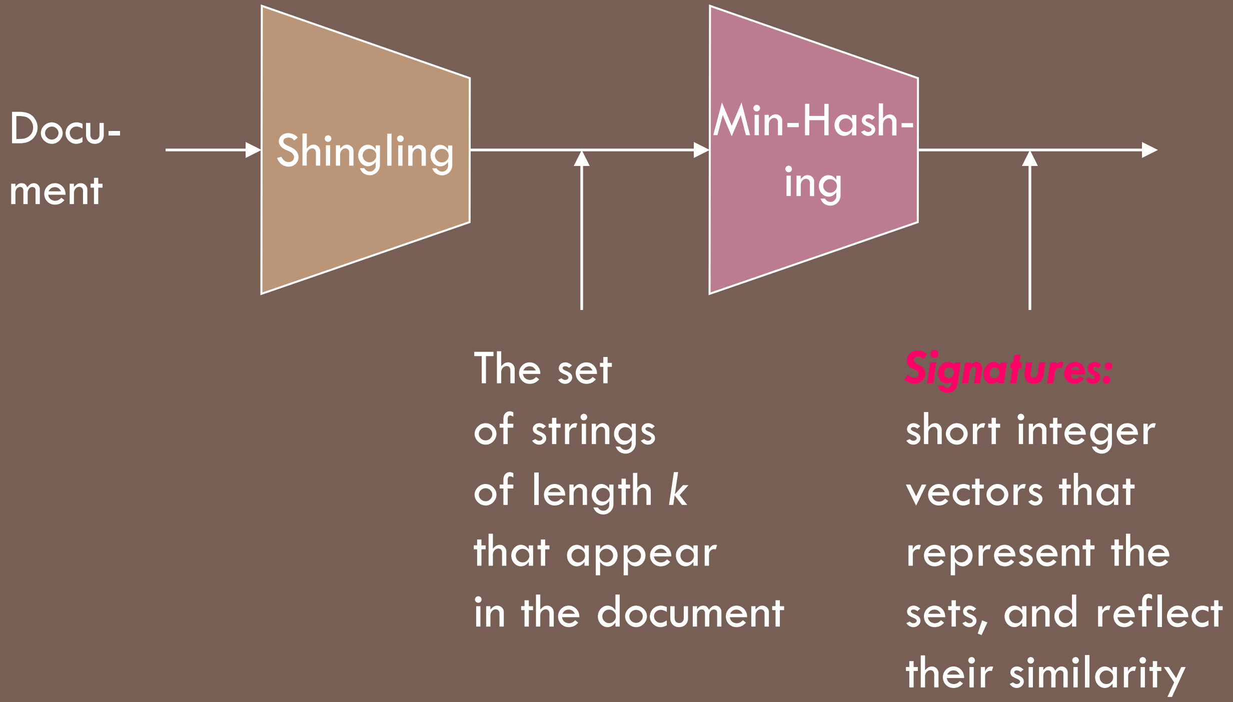
- **Suppose we need to find similar documents among $N = 1$ million documents**

- Naïvely, we would have to compute **pairwise Jaccard similarities** for **every pair of docs**

  - $N(N-1)/2 \approx$ **5*10¹¹** comparisons
  - At 10^5 secs/day and 10^6 comparisons/sec, it would take **5 days**

- For $N = 10$ million, it takes more than a year…

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

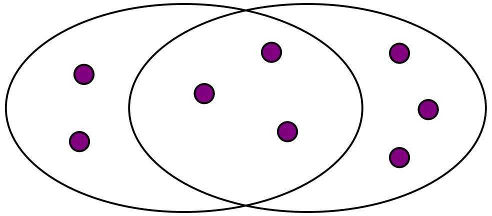

Document
Shingling
Min-Hashing
The set of strings of length *k* that appear in the document
Signatures: short integer vectors that represent the sets, and reflect their similarity

MINHASHING

Step 2: ***Minhashing:*** Convert **large variable length sets** to **short <u>fixed-length</u> signatures**, while **<u>preserving similarity</u>**

Encoding Sets as Bit Vectors

- Many similarity problems can be formalized as **finding subsets that have significant intersection**



- **Encode sets using 0/1 (bit, boolean) vectors**
  - One dimension per element in the universal set
- Interpret set intersection as bitwise **AND**, and set union as bitwise **OR**
- **Example: C_1** = 10111; **C_2** = 10011
  - Size of intersection **= 3**; size of union **= 4,**
  - **Jaccard similarity** (not distance) **= 3/4**
  - **Distance: d(C_1,C_2) = 1 – (Jaccard similarity) = 1/4**

From Sets to Boolean Matrices

Note:Transposed Document Matrix

- **Rows** = elements (shingles)
- **Columns** = sets (documents)
  - 1 in row **e** and column ***s*** if and only if **e** is a valid shingle of document represented by s
  - Column similarity is the Jaccard similarity of the corresponding sets (rows with value *1*)
  - **Typical matrix is sparse!**

$$\begin{array}{c|cccc}
 & \multicolumn{4}{c}{\text{Documents}} \\
\hline
\text{Shingles} & 1 & 1 & 1 & 0 \\
 & 1 & 1 & 0 & 1 \\
 & 0 & 1 & 0 & 1 \\
 & 0 & 0 & 0 & 1 \\
 & 1 & 0 & 0 & 1 \\
 & 1 & 1 & 1 & 0 \\
 & 1 & 0 & 1 & 0
\end{array}$$

Outline: Finding Similar Columns

- **So far:**
  - Documents → Sets of shingles
  - Represent sets as boolean vectors in a matrix

- **Next goal: Find similar columns while computing small signatures**
  - **Similarity of columns == similarity of signatures**

Outline: Finding Similar Columns

- **Next Goal: Find similar columns, Small signatures**

- **Naïve approach:**
  - **1) Signatures of columns:** small summaries of columns
  - **2) Examine pairs of signatures** to find similar columns
    - **Essential:** Similarities of signatures and columns are related
  - **3) Optional:** Check that columns with similar signatures are really similar

- **Warnings:**
  - Comparing all pairs may take too much time: **Job for LSH**
    - These methods can produce false negatives, and even false positives (if the optional check is not made)

Hashing Columns (Signatures) : LSH principle

- **Key idea:** "hash" each column **C** to a small ***signature*** ***h(C)***, such that:
  - **(1)** ***h(C)*** is small enough that the signature fits in RAM
  - **(2)** ***sim*(C_1, C_2)** is the same as the "similarity" of signatures ***h*(C_1)** and ***h*(C_2)**

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Hashing Columns (Signatures) : LSH principle

- **Key idea:** "hash" each column **C** to a small ***signature*** ***h(C)***, such that:
  - (1) *h(C)* is small enough that the signature fits in RAM
  - (2) *sim*(C_1, C_2) is the same as the "similarity" of signatures $h(C_1)$ and $h(C_2)$

- **Goal: Find a hash function *h(·)* such that:**
  - If *sim*(C_1,C_2) is high, then with high prob. $h(C_1) = h(C_2)$
  - If *sim*(C_1,C_2) is low, then with high prob. $h(C_1) \neq h(C_2)$

- **Hash docs into buckets. Expect that "most" pairs of near duplicate docs hash into the same bucket!**

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Min-Hashing

- **Goal: Find a hash function *$h(\cdot)$* such that:**
  - if *$sim(C_1,C_2)$* is high, then with high prob. ***$h(C_1) = h(C_2)$***
  - if *$sim(C_1,C_2)$* is low, then with high prob. ***$h(C_1) \neq h(C_2)$***

- **Clearly, the hash function depends on the similarity metric:**
  - Not all similarity metrics have a suitable hash function

- **There is a suitable hash function for the Jaccard similarity:** It is called **Min-Hashing**

Min-Hashing

- □ Imagine the rows of the boolean matrix permuted under **random permutation** π

- □ Define a **“hash” function** $h_\pi(C)$ = the index of the **first** (in the permuted order π) row in which column **C** has value **1**:

$$h_\pi(C) = \min_\pi \pi(C)$$

- □ Use several (e.g., 100) independent hash functions (that is, permutations) to create a signature of a column

Zoo example (shingle size k=1)

Universe → { dog, cat, lion, tiger, mouse}

π_1 → [cat, mouse, lion, dog, tiger]

π_2 → [lion, cat, mouse, dog, tiger]

A = { mouse, lion }

$mh_1(A)$ = min (π_1{mouse, lion }) = mouse

$mh_2(A)$ = min (π_2{ mouse, lion }) = lion

Key Fact

For two sets A, B, and a min-hash function $mh_i()$:

$$Pr[mh_i(A) = mh_i(B)] = Sim(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Unbiased estimator for Sim using ***K*** hashes (notation policy – this is a different K from size of shingle)

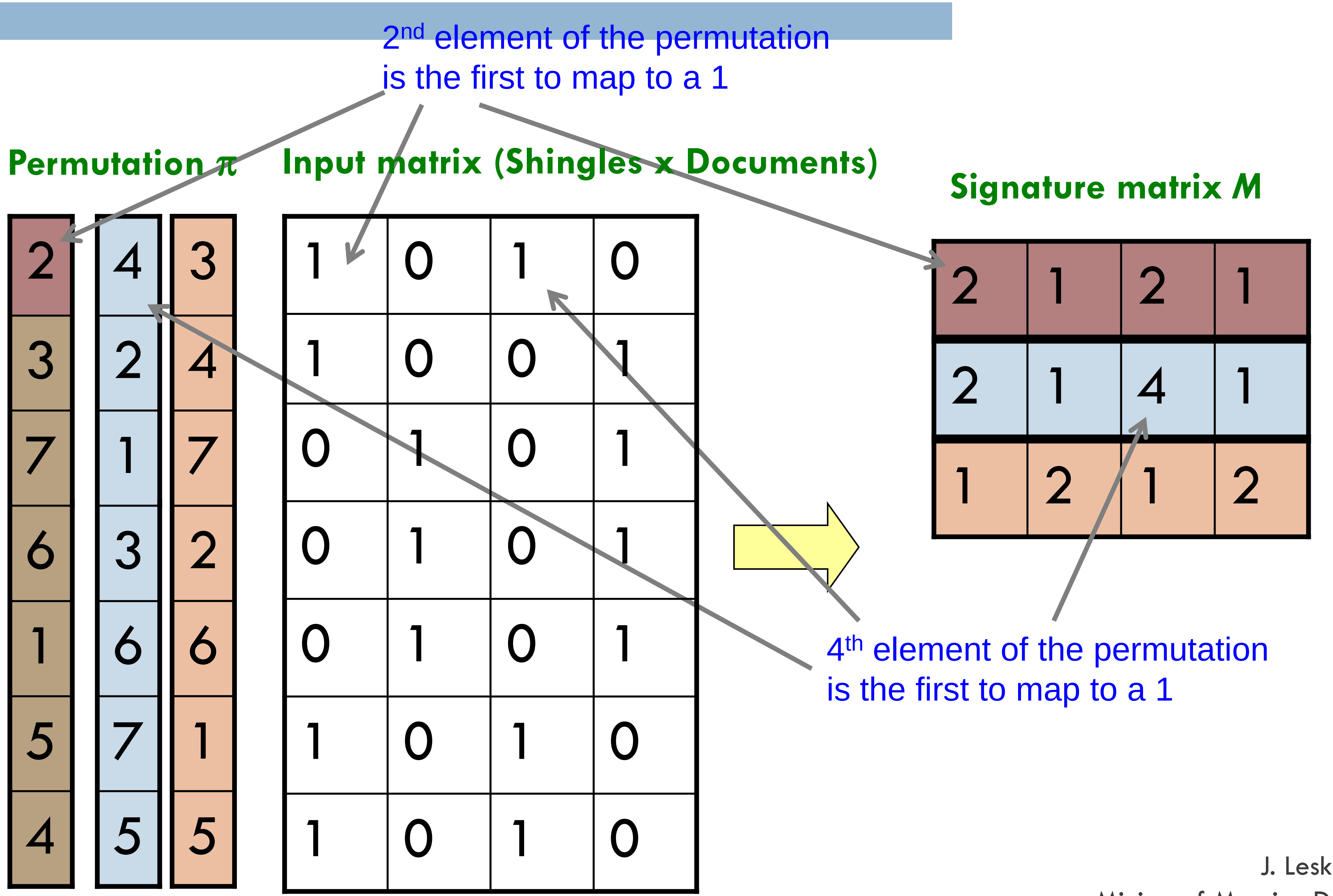
$$S\hat{i}m(A,B) = \tfrac{1}{k} \textstyle\sum_{i=1:k} I[mh_i(A) = mh_i(B)]$$

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Min-Hashing Example

Note: Another (equivalent) way is to store row indexes or raw shingles (e.g. mouse, lion):

1	5	1	5
2	3	1	3
6	4	6	4



2^{nd} element of the permutation is the first to map to a 1

Permutation π

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Input matrix (Shingles x Documents)

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix *M*

2	1	2	1
2	1	4	1
1	2	1	2

4^{th} element of the permutation is the first to map to a 1

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

The Min-Hash Property

- **Choose a random permutation π**
- **Claim:** $\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
- **Why?**
  - Let **X** be a doc (set of shingles), $y \in X$ is a shingle
  - **Then:** $\Pr[\pi(y) = \min(\pi(X))] = 1/|X|$
    - It is equally likely that any $y \in X$ is mapped to the ***min*** element
  - Let y be s.t. $\pi(y) = \min(\pi(C_1 \cup C_2))$
  - **Then either:** $\pi(y) = \min(\pi(C_1))$ if $y \in C_1$, **or**

    $\pi(y) = \min(\pi(C_2))$ if $y \in C_2$
  - So the prob. that **both** are true is the prob. $y \in C_1 \cap C_2$
  - $\Pr[\min(\pi(C_1)) = \min(\pi(C_2))] = |C_1 \cap C_2| / |C_1 \cup C_2| = sim(C_1, C_2)$

0	0
0	0
1	1
0	0
0	1
1	0

One of the two cols had to have 1 at position ***y***

The Min-Hash Property (Take 2: simpler proof)

- **Choose a random permutation π**
- **<u>Claim:</u>** **$\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$**
- **Why?**
  - Given a set X, the probability that any one element is the min-hash under π is $1/|X|$ ← (0)
    - It is equally likely that any $y \in X$ is mapped to the ***min*** element
  - Given a set X, the probability that one of any **k** elements is the min-hash under π is $\mathbf{k}/|X|$ ← (1)
  - For $C_1 \cup C_2$, the probability that any element is the min-hash under π is $1/|C_1 \cup C_2|$ (from 0) ← (2)
  - For any C_1 and C_2, the probability of choosing the same min-hash under π is $|C_1 \cap C_2|/|C_1 \cup C_2|$ ← from (1) and (2)

Similarity for Signatures

- □ We know: $\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
- □ Now generalize to multiple hash functions

- □ **The *similarity of two signatures* is the fraction of the hash functions in which they agree**

- □ **Note:** Because of the Min-Hash property, the similarity of columns is the same as the expected similarity of their signatures

Min-Hashing Example

Permutation π

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Input matrix (Shingles x Documents)

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix *M*

2	1	2	1
2	1	4	1
1	2	1	2

Similarities:

	1-3	2-4	1-2	3-4
Col/Col	0.75	0.75	0	0
Sig/Sig	0.67	1.00	0	0

Min-Hash Signatures

- □ **Pick K=100 random permutations of the rows**
- □ Think of ***sig*(C)** as a column vector
- □ ***sig*(C)[i] =** according to the *i*-th permutation, the index of the first row that has a 1 in column C

$$\boldsymbol{sig}(\mathbf{C})[\mathbf{i}] = \mathbf{min}\ (\pi_i(\mathbf{C}))$$

- □ **Note:** The sketch (signature) of document C is small **~100 bytes!**

- □ **We achieved our goal! We "compressed" long bit vectors into short signatures**

Implementation Trick

- **Permuting rows even once is prohibitive**
- **Approximate Linear Permutation Hashing**
- **Pick K independent hash functions (use a, b below)**
  - Apply the idea on ***each column (document)*** for each hash function and get minhash signature

How to pick a random hash function h(x)?

Universal hashing:

$h_{a,b}(x)=((a{\cdot}x+b) \bmod p) \bmod N$
where:
a,b … random integers
p … prime number (p > N)

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Summary: 3 Steps

- **Shingling:** Convert documents to sets
    - We used hashing to assign each shingle an ID

- **Min-Hashing:** Convert large sets to short signatures, while preserving similarity
    - We used **similarity preserving hashing** to generate signatures with property $\mathbf{Pr}[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
    - We used hashing to get around generating random permutations

Backup slides

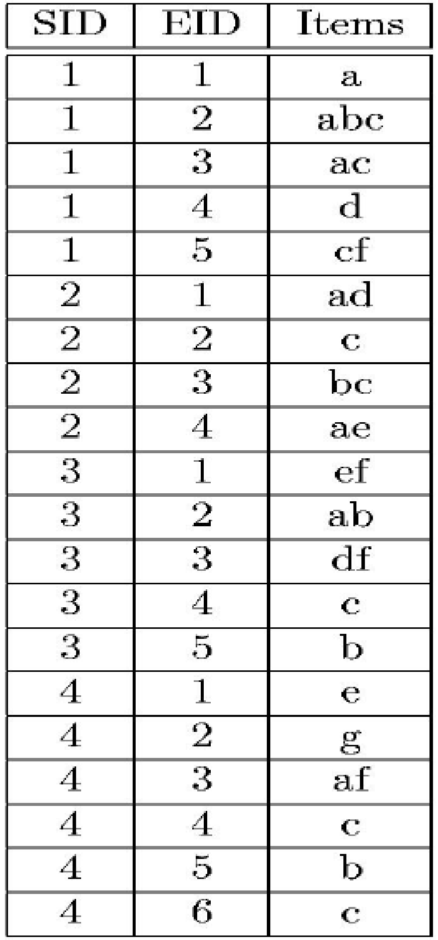
Sequential Pattern Mining in Vertical Data Format: The SPADE Algorithm

- A sequence database is mapped to: <SID, EID>
- Grow the subsequences (patterns) one item at a time by Apriori candidate generation

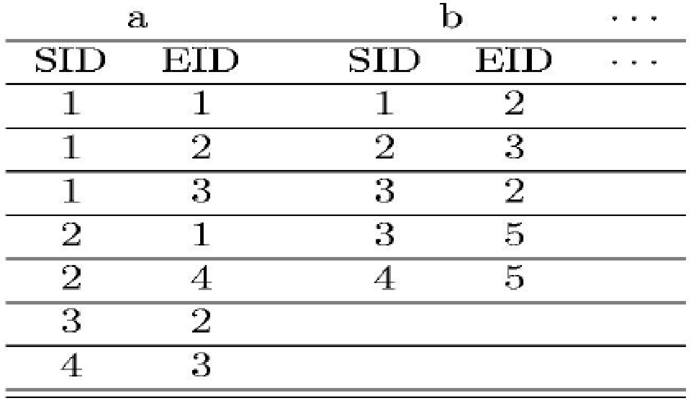
SID	Sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

min_sup = 2

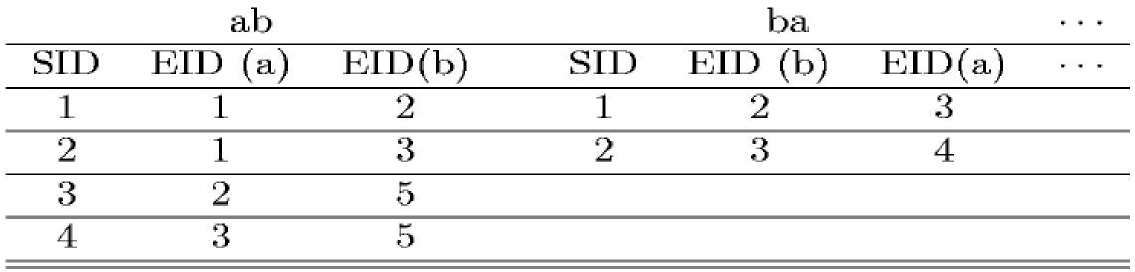
Ref: SPADE (Sequential PAttern Discovery using Equivalent Class) [M. Zaki 2001]



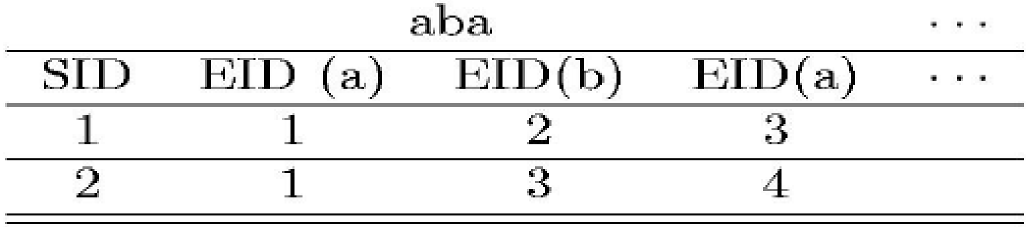
SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c



a		b		. . .
SID	EID	SID	EID	. . .
1	1	1	2	
1	2	2	3	
1	3	3	2	
2	1	3	5	
2	4	4	5	
3	2			
4	3			



ab			ba			. . .
SID	EID (a)	EID(b)	SID	EID (b)	EID(a)	. . .
1	1	2	1	2	3	
2	1	3	2	3	4	
3	2	5				
4	3	5				



aba				. . .
SID	EID (a)	EID(b)	EID(a)	. . .
1	1	2	3	
2	1	3	4	

PrefixSpan: A Pattern-Growth Approach

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

min_sup = 2

Prefix	Suffix (Projection)
<a>	<(abc)(ac)d(cf)>
<aa>	<(_bc)(ac)d(cf)>
<ab>	<(_c)(ac)d(cf)>

- Prefix and suffix
  - Given <a(abc)(ac)d(cf)>
  - Prefixes: <a>, <aa>, <a(ab)>, <a(abc)>, …
  - Suffix: Prefixes-based projection

- PrefixSpan Mining: Prefix Projections
  - Step 1: Find length-1 sequential patterns
    - <a>, <b>, <c>, <d>, <e>, <f>
  - Step 2: Divide search space and mine each projected DB
    - <a>-projected DB,
    - <b>-projected DB,
    - …
    - <f>-projected DB, …

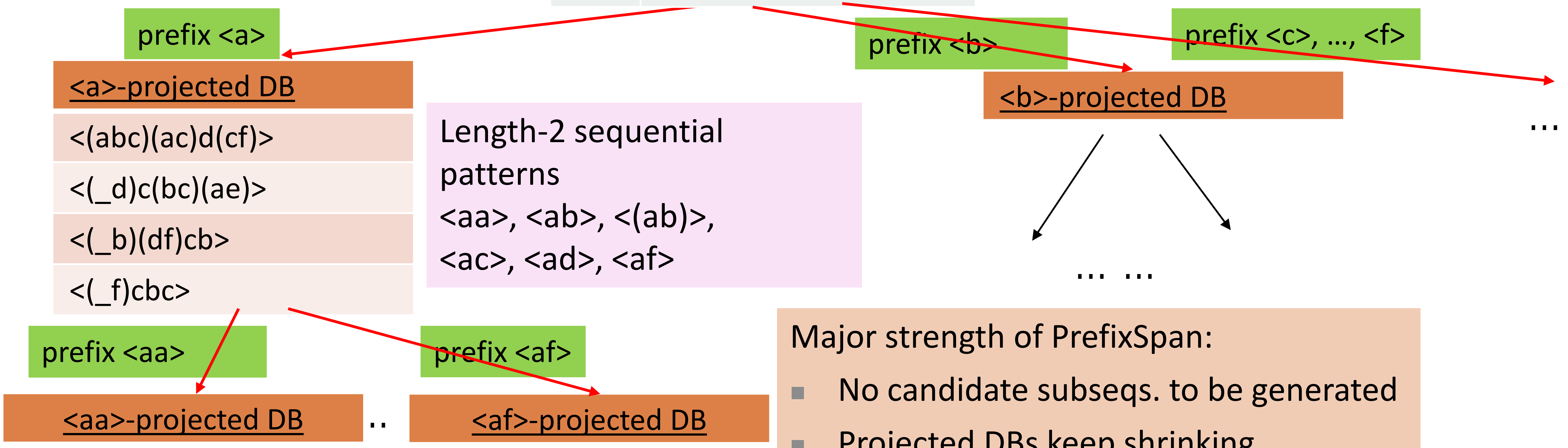
PrefixSpan (Prefix-projected Sequential pattern mining) Pei, et al. @TKDE’04

PrefixSpan: Mining Prefix-Projected DBs

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

min_sup = 2

Length-1 sequential patterns
<a>, <b>, <c>, <d>, <e>, <f>



prefix <a>

<a>-projected DB
<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

Length-2 sequential patterns
<aa>, <ab>, <(ab)>, <ac>, <ad>, <af>

prefix <b>

<b>-projected DB

prefix <c>, …, <f>

…

… …

prefix <aa>

<aa>-projected DB

..

prefix <af>

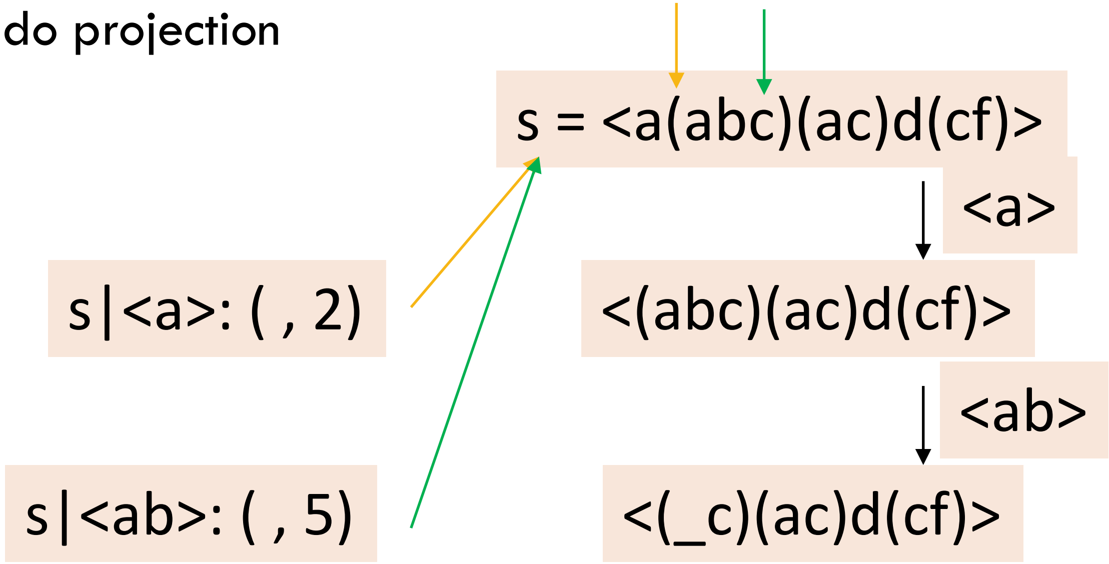
<af>-projected DB

Major strength of PrefixSpan:

- No candidate subseqs. to be generated
- Projected DBs keep shrinking

Consideration:
Pseudo-Projection vs. Physical PrImplementation ojection

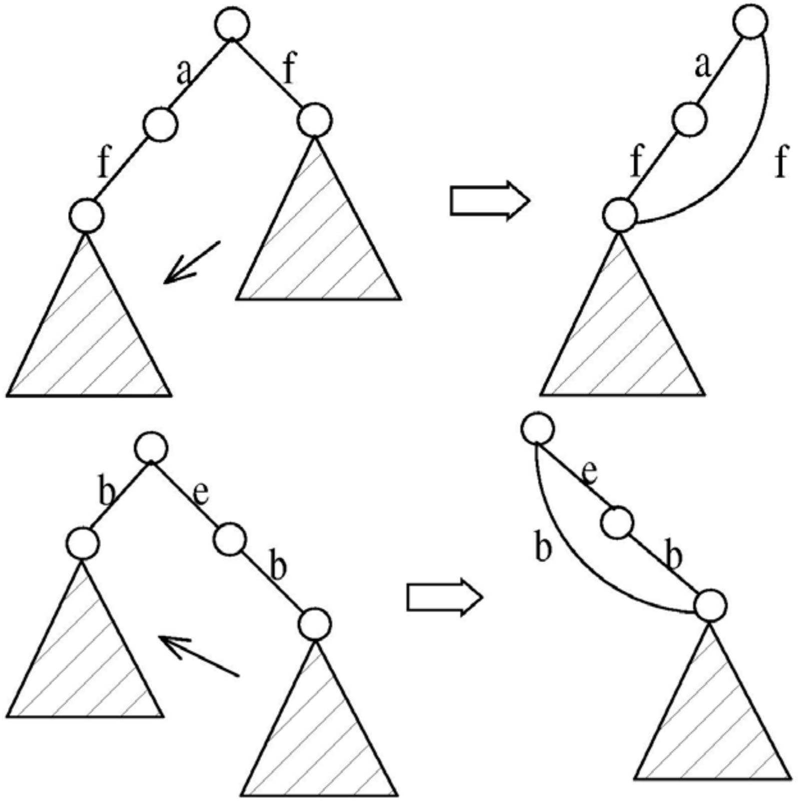
- Major cost of PrefixSpan: Constructing projected DBs
    - Suffixes largely repeating in recursive projected DBs
- When DB can be held in main memory, use pseudo projection
    - No physically copying suffixes
    - Pointer to the sequence
    - Offset of the suffix
  - But if it does not fit in memory
    - Physical projection
  - Suggested approach:
    - Integration of physical and pseudo-projection
    - Swapping to pseudo-projection when the data fits in memory



CloSpan: Mining Closed Sequential Patterns

- A closed sequential pattern *s*: There exists no superpattern *s'* such that $s' \supset s$, and *s'* and *s* have the same support
- Which ones are closed? <abc>: 20, <abcd>:20, <abcde>: 15

- Why directly mine closed sequential patterns?
  - Reduce # of (redundant) patterns
  - Attain the same expressive power
- Property P_1: If $s \supset s_1$, s is closed iff two project DBs have the same size
- Explore *Backward Subpattern* and *Backward Superpattern* pruning to prune redundant search space
- Greatly enhances efficiency (Yan, et al., SDM'03)

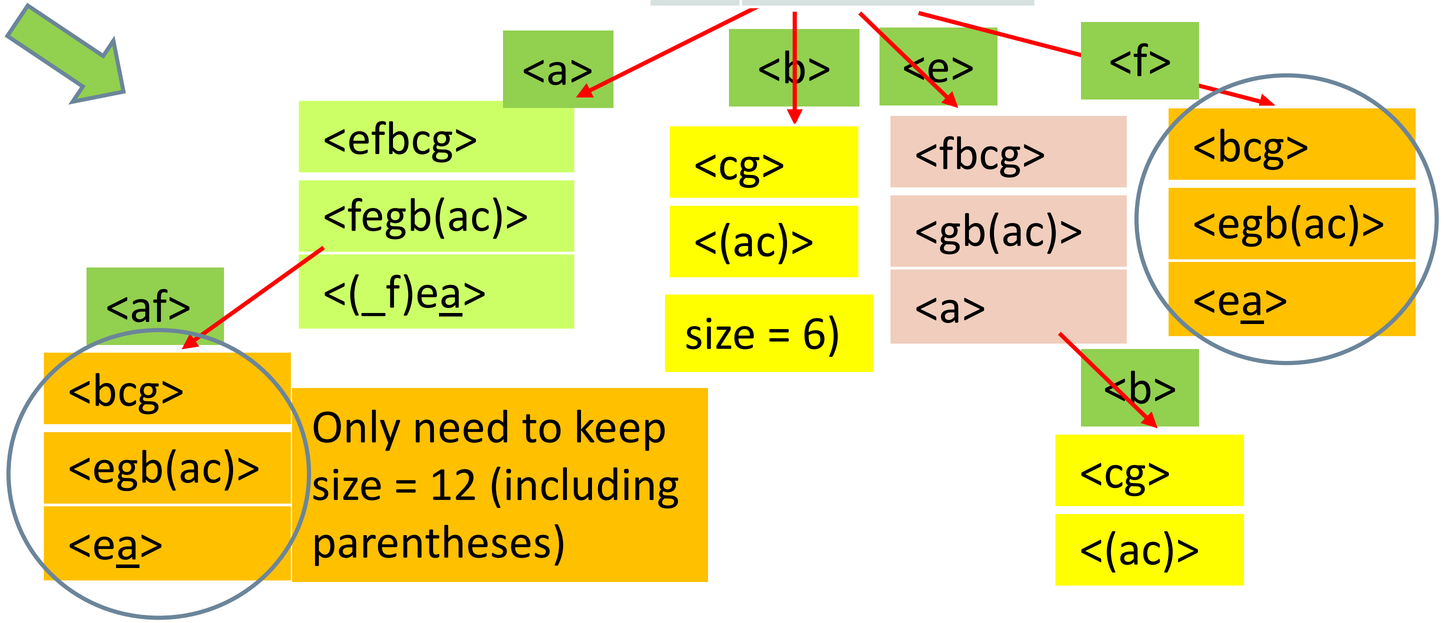


CloSpan: When Two Projected DBs Have the Same Size

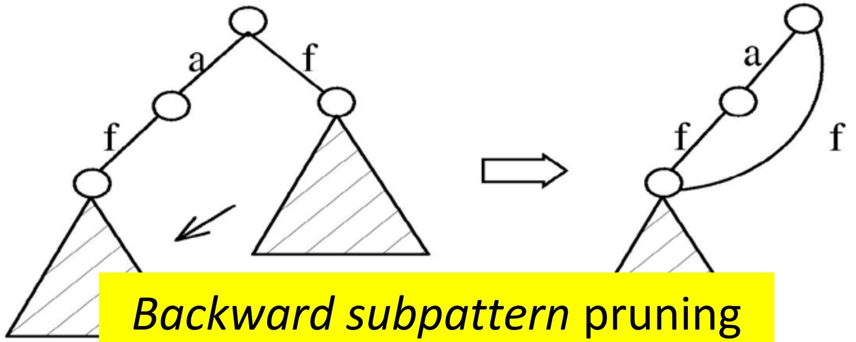
- If $s \supset s_1$, s is closed iff two project DBs have the same size
  - When two projected sequence DBs have the same size?
    - Here is one example:

ID	Sequence
1	<aefbcg>
2	<afegb(ac)>
3	<(af)ea>

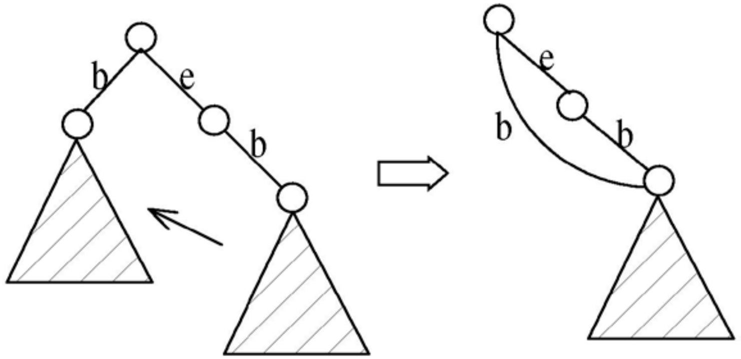
min_sup = 2



Only need to keep size = 12 (including parentheses)



Backward subpattern pruning



Backward superpattern pruning

Chapter 7 : Advanced Frequent Pattern Mining

□ Mining Diverse Patterns

□ Sequential Pattern Mining

□ Constraint-Based Frequent Pattern Mining

□ Graph Pattern Mining

□ Pattern Mining Application: Mining Software Copy-and-Paste Bugs

□ Summary

Constraint-Based Pattern Mining

- Why Constraint-Based Mining?
- Different Kinds of Constraints: Different Pruning Strategies
- Constrained Mining with Pattern Anti-Monotonicity
- Constrained Mining with Pattern Monotonicity
- Constrained Mining with Data Anti-Monotonicity
- Constrained Mining with Succinct Constraints
- Constrained Mining with Convertible Constraints
- Handling Multiple Constraints
- Constraint-Based Sequential-Pattern Mining

Why Constraint-Based Mining?

- Finding all the patterns in a dataset autonomously?—unrealistic!
  - Too many patterns but not necessarily user-interested!
- Pattern mining in practice: Often a user-guided, interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface), specifying various kinds of constraints
- What is constraint-based mining?
  - Mine together with user-provided constraints
- Why constraint-based mining?
  - User flexibility: User provides constraints on what to be mined
  - Optimization: System explores such constraints for mining efficiency
    - E.g., Push constraints deeply into the mining process

Various Kinds of User-Specified Constraints in Data Mining

- Knowledge type constraint—Specifying what kinds of knowledge to mine
  - Ex.: Classification, association, clustering, outlier finding, …
- Data constraint—using SQL-like queries
  - Ex.: Find products sold together in NY stores this year
- Dimension/level constraint—similar to projection in relational database
  - Ex.: In relevance to region, price, brand, customer category
- Interestingness constraint—various kinds of thresholds
  - Ex.: Strong rules: min_sup $\geq$ 0.02, min_conf $\geq$ 0.6, min_correlation $\geq$ 0.7
- Rule (or pattern) constraint ← The focus of this study
  - Ex.: Small sales (price < \$10) triggers big sales (sum > \$200)

Pattern Space Pruning with Pattern Anti-Monotonicity

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

Item	Price	Profit
a	100	40
b	40	0
c	150	−20
d	35	−15
e	55	−30
f	45	−10
g	80	20
h	10	5

- A constraint *c* is ***anti-monotone***
  - If an itemset S **violates** constraint *c*, so does any of its superset
  - That is, mining on itemset S can be terminated
- Ex. 1: c_1: $sum(S.price) \leq v$ is anti-monotone
- Ex. 2: c_2: $\text{range}(S.\text{profit}) \leq 15$ is anti-monotone
  - Itemset *ab* violates c_2 (range(ab) = 40)
  - So does every superset of *ab*
- Ex. 3. c_3: $sum(S.Price) \geq v$ is not anti-monotone
- Ex. 4. Is c_4: $support(S) \geq \sigma$ anti-monotone?
  - Yes! Apriori pruning is essentially pruning with an anti-monotonic constraint!

Note: item.price > 0
Profit can be negative

Pattern Monotonicity and Its Roles

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

Item	Price	Profit
a	100	40
b	40	0
c	150	−20
d	35	−15
e	55	−30
f	45	−10
g	80	20
h	10	5

- A constraint *c* is *monotone*: If an itemset S **satisfies** the constraint c, so does any of its superset
  - That is, we do not need to check *c* in subsequent mining
- Ex. 1: c_1: $sum(S.Price) \geq v$ is monotone
- Ex. 2: c_2: $min(S.Price) \leq v$ is monotone
- Ex. 3: c_3: range(S.profit) $\geq$ 15 is monotone
  - Itemset *ab* satisfies c_3
  - So does every superset of *ab*

Note: item.price > 0
Profit can be negative

Data Space Pruning with Data Anti-Monotonicity

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

Item	Price	Profit
a	100	40
b	40	0
c	150	−20
d	35	−15
e	55	−30
f	45	−10
g	80	20
h	10	5

- A constraint c is *data anti-monotone*: In the mining process, if a data entry *t* cannot satisfy a pattern *p* under c, *t* cannot satisfy *p*'s superset either
  - Data space pruning: Data entry *t* can be pruned
- Ex. 1: c_1: $sum(S.Profit) \geq v$ is data anti-monotone
  - Let constraint c_1 be: $sum(S.Profit) \geq 25$
    - T_{30}: {b, c, d, f, g} can be removed since none of their combinations can make an S whose sum of the profit is ≥ 25
- Ex. 2: c_2: $min(S.Price) \leq v$ is data anti-monotone
    - Consider $v = 5$ but every item in a transaction, say T_{50}, has a price higher than 10
- Ex. 3: c_3: $range(S.Profit) > 25$ is data anti-monotone

Note: item.price > 0
Profit can be negative

Expressing Patterns in Compressed Form: Closed Patterns

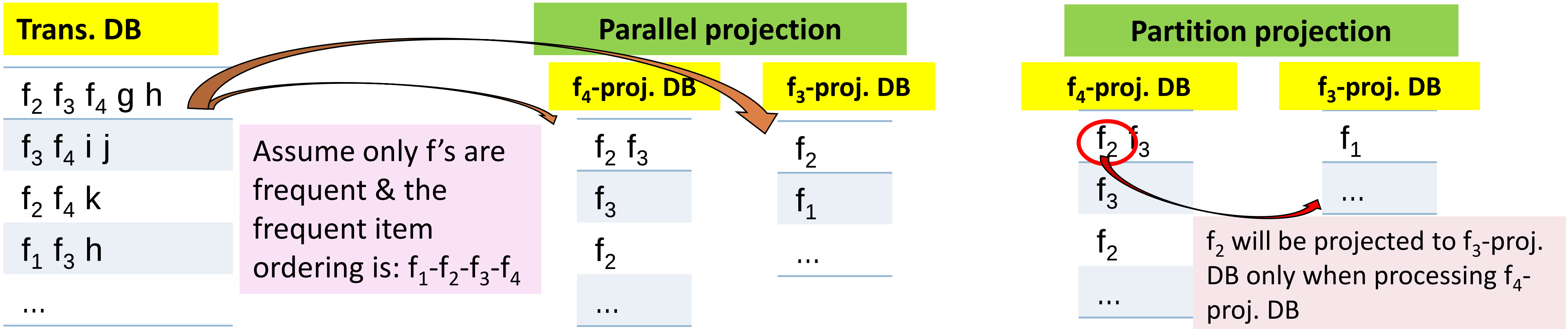
- How to handle such a challenge?
- Solution 1: **Closed patterns:** A pattern (itemset) X is closed if X is *frequent*, and there exists *no super-pattern* Y ⊃ X, *with the same support* as X
  - Let Transaction DB TDB$_1$: T$_1$: $\{a_1, \ldots, a_{50}\}$; T$_2$: $\{a_1, \ldots, a_{100}\}$
  - Suppose *minsup* = 1. How many closed patterns does TDB$_1$ contain?
    - Two: P$_1$: "$\{a_1, \ldots, a_{50}\}$: 2"; P$_2$: "$\{a_1, \ldots, a_{100}\}$: 1"
- Closed pattern is a lossless compression of frequent patterns
  - Reduces the # of patterns but does not lose the support information!
  - You will still be able to say: "$\{a_2, \ldots, a_{40}\}$: 2", "$\{a_5, a_{51}\}$: 1"

Expressing Patterns in Compressed Form: Max-Patterns

- Solution 2: **Max-patterns:** A pattern X is a maximal frequent pattern or max-pattern if X is frequent and there exists no frequent super-pattern Y ɔ X
- Difference from close-patterns?
  - Do not care the real support of the sub-patterns of a max-pattern
  - Let Transaction DB TDB_1: T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$
  - Suppose *minsup* = 1. How many max-patterns does TDB_1 contain?
    - One: P: "$\{a_1, \ldots, a_{100}\}$: 1"
- Max-pattern is a lossy compression!
  - We only know $\{a_1, \ldots, a_{40}\}$ is frequent
  - But we do not know the real support of $\{a_1, \ldots, a_{40}\}$, …, any more!
  - Thus in many applications, close-patterns are more desirable than max-patterns

Scaling FP-growth by Item-Based Data Projection

- What if FP-tree cannot fit in memory?—Do not construct FP-tree
  - "Project" the database based on frequent single items
  - Construct & mine FP-tree for each projected DB
- Parallel projection vs. partition projection
  - Parallel projection: Project the DB on each frequent item
    - Space costly, all partitions can be processed in parallel
  - Partition projection: Partition the DB in order
    - Passing the unprocessed parts to subsequent partitions



Trans. DB
f_2 f_3 f_4 g h
f_3 f_4 i j
f_2 f_4 k
f_1 f_3 h
…

Assume only f's are frequent & the frequent item ordering is: f_1-f_2-f_3-f_4

Parallel projection

f_4-proj. DB	f_3-proj. DB
f_2 f_3	f_2
f_3	f_1
f_2	…
…	

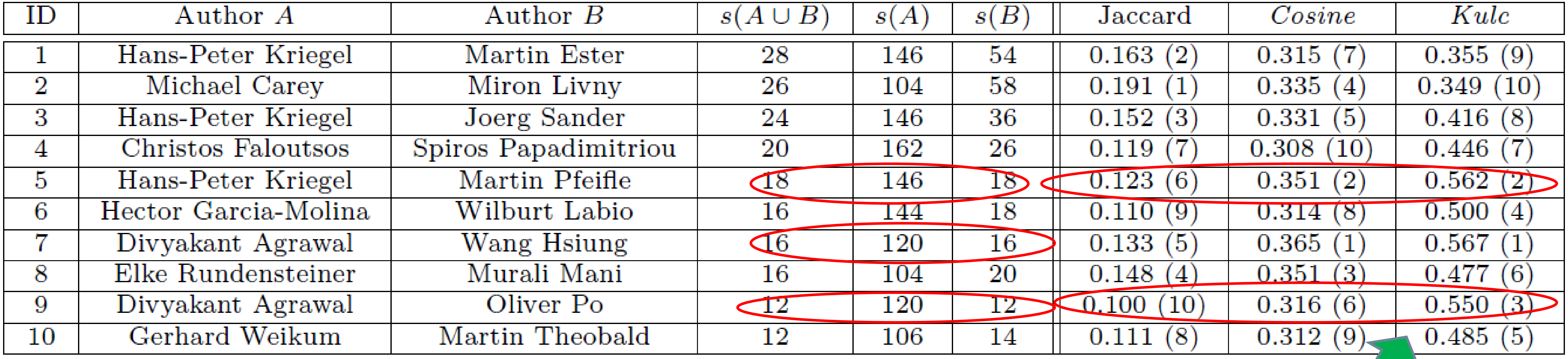
Partition projection

f_4-proj. DB	f_3-proj. DB
f_2 f_3	f_1
f_3	…
f_2	
…	

f_2 will be projected to f_3-proj. DB only when processing f_4-proj. DB

Analysis of DBLP Coauthor Relationships

- DBLP: Computer science research publication bibliographic database
  - > 3.8 million entries on authors, paper, venue, year, and other information



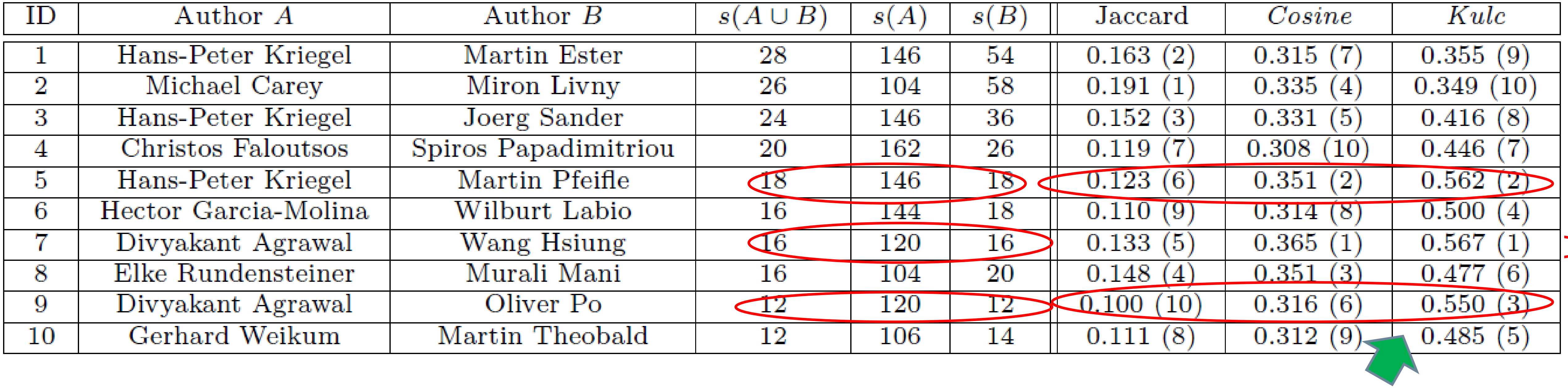
ID	Author A	Author B	$s(A \cup B)$	$s(A)$	$s(B)$	*Jaccard*	*Cosine*	*Kulc*
1	Hans-Peter Kriegel	Martin Ester	28	146	54	0.163 (2)	0.315 (7)	0.355 (9)
2	Michael Carey	Miron Livny	26	104	58	0.191 (1)	0.335 (4)	0.349 (10)
3	Hans-Peter Kriegel	Joerg Sander	24	146	36	0.152 (3)	0.331 (5)	0.416 (8)
4	Christos Faloutsos	Spiros Papadimitriou	20	162	26	0.119 (7)	0.308 (10)	0.446 (7)
5	Hans-Peter Kriegel	Martin Pfeifle	18	146	18	0.123 (6)	0.351 (2)	0.562 (2)
6	Hector Garcia-Molina	Wilburt Labio	16	144	18	0.110 (9)	0.314 (8)	0.500 (4)
7	Divyakant Agrawal	Wang Hsiung	16	120	16	0.133 (5)	0.365 (1)	0.567 (1)
8	Elke Rundensteiner	Murali Mani	16	104	20	0.148 (4)	0.351 (3)	0.477 (6)
9	Divyakant Agrawal	Oliver Po	12	120	12	0.100 (10)	0.316 (6)	0.550 (3)
10	Gerhard Weikum	Martin Theobald	12	106	14	0.111 (8)	0.312 (9)	0.485 (5)

Advisor-advisee relation: Kulc: high, Jaccard: low, cosine: middle

- Which pairs of authors are strongly related?
  - Use Kulc to find Advisor-advisee, close collaborators

Analysis of DBLP Coauthor Relationships

- DBLP: Computer science research publication bibliographic database
  - > 3.8 million entries on authors, paper, venue, year, and other information



ID	Author A	Author B	$s(A \cup B)$	$s(A)$	$s(B)$	Jaccard	*Cosine*	*Kulc*
1	Hans-Peter Kriegel	Martin Ester	28	146	54	0.163 (2)	0.315 (7)	0.355 (9)
2	Michael Carey	Miron Livny	26	104	58	0.191 (1)	0.335 (4)	0.349 (10)
3	Hans-Peter Kriegel	Joerg Sander	24	146	36	0.152 (3)	0.331 (5)	0.416 (8)
4	Christos Faloutsos	Spiros Papadimitriou	20	162	26	0.119 (7)	0.308 (10)	0.446 (7)
5	Hans-Peter Kriegel	Martin Pfeifle	18	146	18	0.123 (6)	0.351 (2)	0.562 (2)
6	Hector Garcia-Molina	Wilburt Labio	16	144	18	0.110 (9)	0.314 (8)	0.500 (4)
7	Divyakant Agrawal	Wang Hsiung	16	120	16	0.133 (5)	0.365 (1)	0.567 (1)
8	Elke Rundensteiner	Murali Mani	16	104	20	0.148 (4)	0.351 (3)	0.477 (6)
9	Divyakant Agrawal	Oliver Po	12	120	12	0.100 (10)	0.316 (6)	0.550 (3)
10	Gerhard Weikum	Martin Theobald	12	106	14	0.111 (8)	0.312 (9)	0.485 (5)

Advisor-advisee relation: Kulc: high, Jaccard: low, cosine: middle

- Which pairs of authors are strongly related?
  - Use Kulc to find Advisor-advisee, close collaborators

What Measures to Choose for Effective Pattern Evaluation?

- Null value cases are predominant in many large datasets
  - Neither milk nor coffee is in most of the baskets; neither Mike nor Jim is an author in most of the papers; ……
- *Null-invariance* is an important property
- Lift, χ^2 and cosine are good measures if null transactions are not predominant
  - Otherwise, *Kulczynski* + *Imbalance Ratio* should be used to judge the interestingness of a pattern
- Exercise: Mining research collaborations from research bibliographic data
  - Find a group of frequent collaborators from research bibliographic data (e.g., DBLP)
  - Can you find the likely advisor-advisee relationship and during which years such a relationship happened?
  - Ref.: C. Wang, J. Han, Y. Jia, J. Tang, D. Zhang, Y. Yu, and J. Guo, "Mining Advisor-Advisee Relationships from Research Publication Networks", KDD'10

Mining Compressed Patterns

Pat-ID	Item-Sets	Support
P1	{38,16,18,12}	205227
P2	{38,16,18,12,17}	205211
P3	{39,38,16,18,12,17}	101758
P4	{39,16,18,12,17}	161563
P5	{39,16,18,12}	161576

- Closed patterns
  - P1, P2, P3, P4, P5
  - Emphasizes too much on support
  - There is no compression
- Max-patterns
  - P3: information loss
- Desired output (a good balance):
  - P2, P3, P4

- Why mining compressed patterns?
  - Too many scattered patterns but not so meaningful
- Pattern distance measure

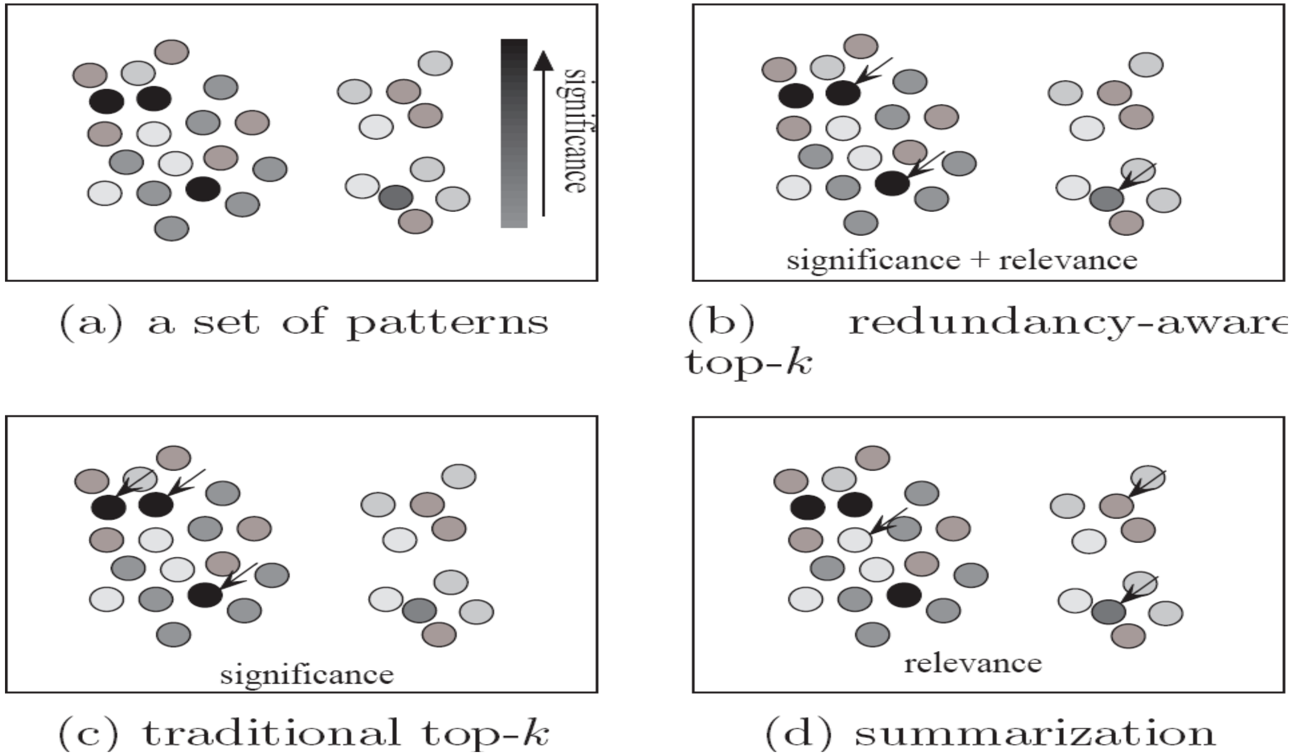
$$Dist(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|}$$

- δ-clustering: For each pattern P, find all patterns which can be expressed by P and whose distance to P is within δ (δ-cover)
- All patterns in the cluster can be represented by P

- Method for efficient, direct mining of compressed frequent patterns (e.g., D. Xin, J. Han, X. Yan, H. Cheng, "On Compressing Frequent Patterns", Knowledge and Data Engineering, 60:5-29, 2007)

Redundancy-Aware Top-k Patterns

- Desired patterns: high significance & low redundancy



(a) a set of patterns

(b) redundancy-aware top-k

(c) traditional top-k

(d) summarization

- Method: Use MMS (Maximal Marginal Significance) for measuring the combined significance of a pattern set
- Xin et al., Extracting Redundancy-Aware Top-K Patterns, KDD'06

Redundancy Filtering at Mining Multi-Level Associations

- Multi-level association mining may generate many redundant rules
- Redundancy filtering: Some rules may be redundant due to "ancestor" relationships between items
  - milk ⇒ wheat bread [support = 8%, confidence = 70%] (1)
  - 2% milk ⇒ wheat bread [support = 2%, confidence = 72%] (2)
    - Suppose the "2% milk" sold is about "¼" of milk sold
      - Does (2) provide any novel information?
- A rule is *redundant* if its support is close to the "expected" value, according to its "ancestor" rule, and it has a similar confidence as its "ancestor"
  - Rule (1) is an ancestor of rule (2), which one to prune?

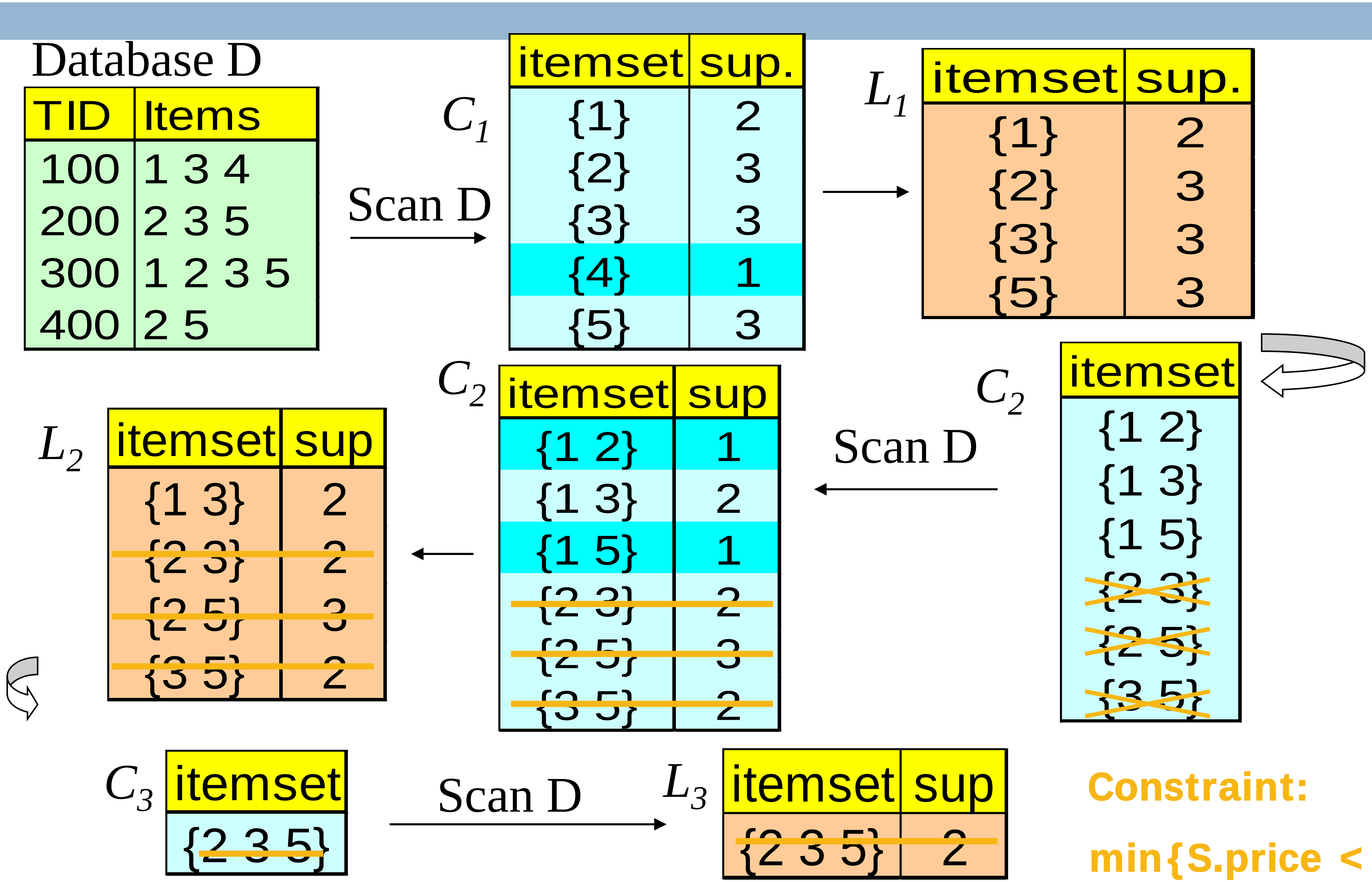
Succinctness

- Succinctness:
  - Given $A_{1,}$ the set of items satisfying a succinctness constraint C, then any set *S* satisfying C is based on A_1 , i.e., *S* contains a subset belonging to A_1
  - Idea: Without looking at the transaction database, whether an itemset *S* satisfies constraint C can be determined based on the selection of items
  - $min(S.Price) \leq v$ is succinct
  - $sum(S.Price) \geq v$ is not succinct
- Optimization: If C is succinct, C is pre-counting pushable

Which Constraints Are Succinct?

Constraint	Succinct
$v \in S$	yes
$S \supseteq V$	yes
$S \subseteq V$	yes
$\min(S) \leq v$	yes
$\min(S) \geq v$	yes
$\max(S) \leq v$	yes
$\max(S) \geq v$	yes
$\text{sum}(S) \leq v\ (a \in S, a \geq 0)$	no
$\text{sum}(S) \geq v\ (a \in S, a \geq 0)$	no
$\text{range}(S) \leq v$	no
$\text{range}(S) \geq v$	no
$\text{avg}(S)\ \theta\ v, \theta \in \{=, \leq, \geq\}$	no
$\text{support}(S) \geq \xi$	no
$\text{support}(S) \leq \xi$	no

Push a Succinct Constraint Deep



Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
~~{2 3}~~
~~{2 5}~~
~~{3 5}~~

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

L_2

itemset	sup
{1 3}	2
~~{2 3}~~	~~2~~
~~{2 5}~~	~~3~~
~~{3 5}~~	~~2~~

C_3

itemset
~~{2 3 5}~~

Scan D

L_3

itemset	sup
~~{2 3 5}~~	~~2~~

Constraint:

min{S.price <= 1 }

Sequential Pattern Mining

□ Sequential Pattern and Sequential Pattern Mining

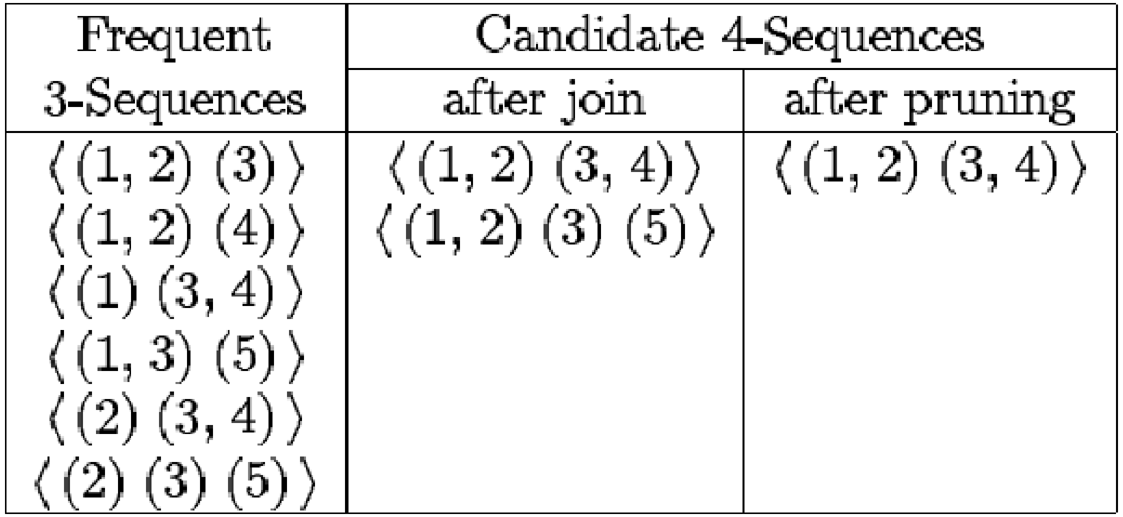
□ GSP: Apriori-Based Sequential Pattern Mining

□ SPADE: Sequential Pattern Mining in Vertical Data Format

□ PrefixSpan: Sequential Pattern Mining by Pattern-Growth

□ CloSpan: Mining Closed Sequential Patterns

GSP: Candidate Generation

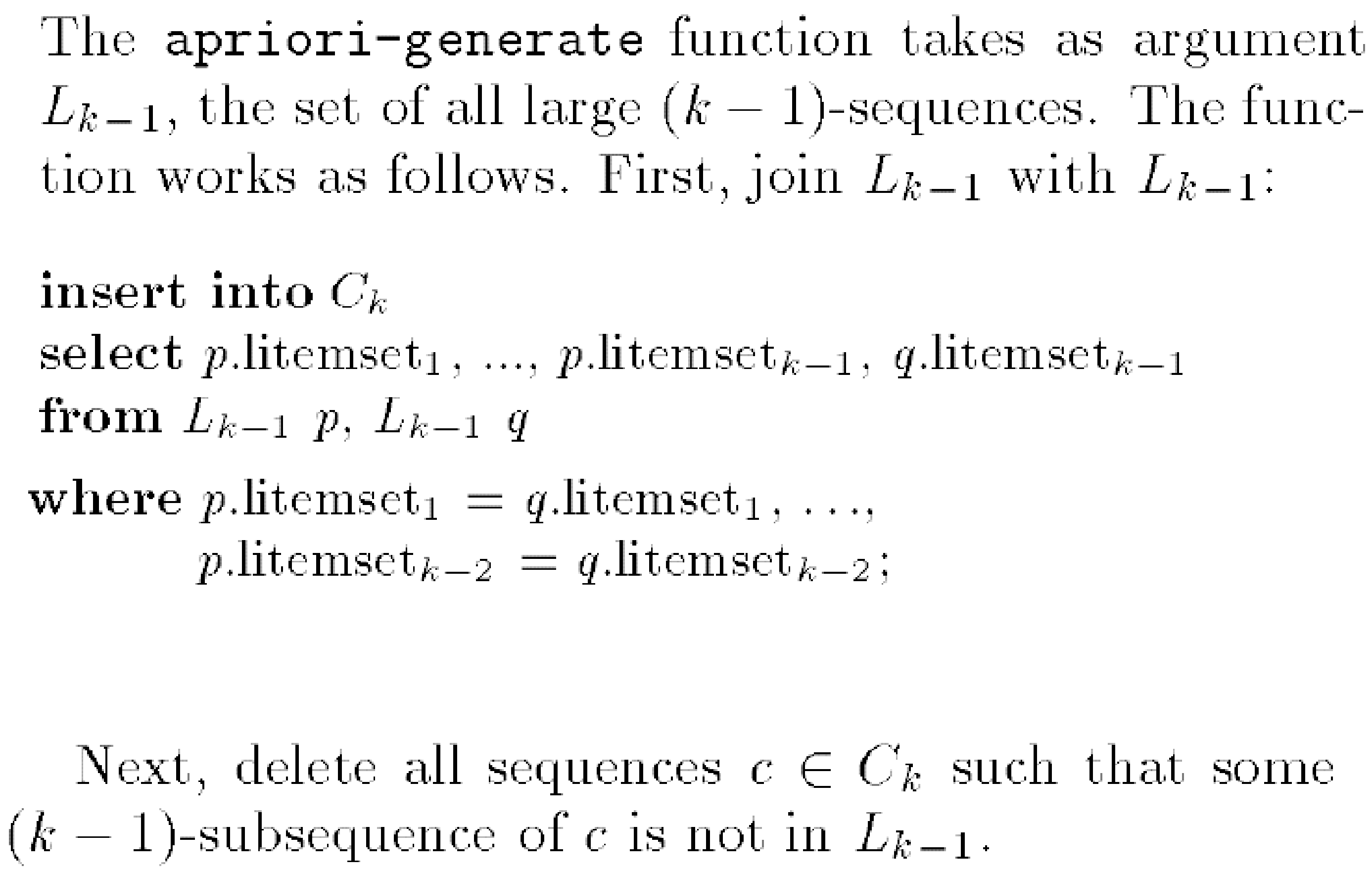


Frequent	Candidate 4-Sequences	
3-Sequences	after join	after pruning
$\langle (1, 2)\ (3) \rangle$	$\langle (1, 2)\ (3, 4) \rangle$	$\langle (1, 2)\ (3, 4) \rangle$
$\langle (1, 2)\ (4) \rangle$	$\langle (1, 2)\ (3)\ (5) \rangle$	
$\langle (1)\ (3, 4) \rangle$		
$\langle (1, 3)\ (5) \rangle$		
$\langle (2)\ (3, 4) \rangle$		
$\langle (2)\ (3)\ (5) \rangle$		

Figure 3: Candidate Generation: Example

The sequence **< (1,2) (3) (5) >** is dropped in the pruning phase, since its contiguous subsequence **< (1) (3) (5) >** is not frequent.

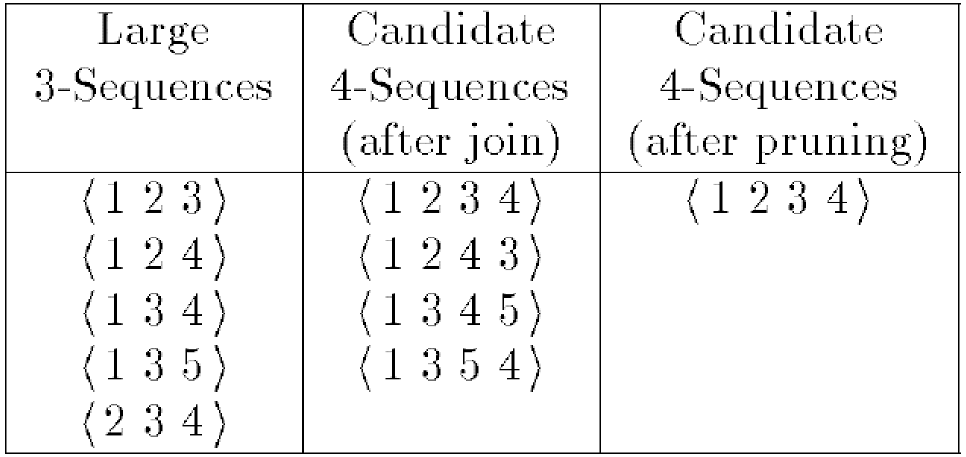
GSP Algorithm: Apriori Candidate Generation



The `apriori-generate` function takes as argument L_{k-1}, the set of all large $(k-1)$-sequences. The function works as follows. First, join L_{k-1} with L_{k-1}:

```
insert into $C_k$
select $p$.litemset$_1$, ..., $p$.litemset$_{k-1}$, $q$.litemset$_{k-1}$
from $L_{k-1}$ $p$, $L_{k-1}$ $q$
where $p$.litemset$_1$ $=$ $q$.litemset$_1$, . . .,
        $p$.litemset$_{k-2}$ $=$ $q$.litemset$_{k-2}$;
```

Next, delete all sequences $c \in C_k$ such that some $(k-1)$-subsequence of c is not in L_{k-1}.



Large 3-Sequences	Candidate 4-Sequences (after join)	Candidate 4-Sequences (after pruning)
$\langle 1\ 2\ 3 \rangle$	$\langle 1\ 2\ 3\ 4 \rangle$	$\langle 1\ 2\ 3\ 4 \rangle$
$\langle 1\ 2\ 4 \rangle$	$\langle 1\ 2\ 4\ 3 \rangle$	
$\langle 1\ 3\ 4 \rangle$	$\langle 1\ 3\ 4\ 5 \rangle$	
$\langle 1\ 3\ 5 \rangle$	$\langle 1\ 3\ 5\ 4 \rangle$	
$\langle 2\ 3\ 4 \rangle$		

Figure 7: Candidate Generation

Mining Sequential Patterns, Agrawal et al., ICDE'95