Lecture 10
Sequential Pattern Mining

Zhou Shuigeng

June 3, 2007
Outline

- Sequence data
- Sequential patterns
- Basic algorithm for sequential pattern mining
- Advanced algorithms for sequential pattern mining
Outline

- **Sequence data**
- Sequential patterns
- Basic algorithm for sequential pattern mining
- Advanced algorithms for sequential pattern mining
Sequence Data

- Sequence database

### Sequence Table

<table>
<thead>
<tr>
<th>Object</th>
<th>Timestamp</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>2, 3, 5</td>
</tr>
<tr>
<td>A</td>
<td>20</td>
<td>6, 1</td>
</tr>
<tr>
<td>A</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td>4, 5, 6</td>
</tr>
<tr>
<td>B</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>7, 8, 1, 2</td>
</tr>
<tr>
<td>B</td>
<td>28</td>
<td>1, 6</td>
</tr>
<tr>
<td>C</td>
<td>14</td>
<td>1, 8, 7</td>
</tr>
</tbody>
</table>

### Timeline Diagram

- Object A:
  - Timestamp 23.5
  - Events: 2, 4, 6
  - Duration: 6

- Object B:
  - Timestamp 4.5
  - Events: 5, 2, 7, 8
  - Duration: 12

- Object C:
  - Timestamp 1.7
  - Events: 1, 6
  - Duration: 8
## Examples of Sequence Data

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

---

Data Mining: Tech. & Appl.
Formal Definition of a Sequence

- A sequence is an ordered list of elements $s = \langle e_1, e_2, e_3, \ldots \rangle$
  - Each element contains a collection of events (items), i.e., $e_i = \{i_1, i_2, \ldots, i_k\}$
- Length of a sequence, $|s|$, is given by the number of elements in the sequence
- A $k$-sequence is a sequence that contains $k$ events (items)
Examples of Sequence

- **Web sequence:**
  - `< {Homepage} {Electronics} {Digital Cameras} {Canon Digital Camera} {Shopping Cart} {Order Confirmation} {Return to Shopping} >`

- **Sequence of initiating events causing the nuclear accident at 3-mile Island:**
  - (http://stellar-one.com/nuclear/staff_reports/summary_SOE_the_initiating_event.htm)
  - `< {clogged resin} {outlet valve closure} {loss of feedwater} {condenser polisher outlet valve shut} {booster pumps trip} {main waterpump trips} {main turbine trips} {reactor pressure increases}>`

- **Sequence of books checked out at a library:**
  - `<{Fellowship of the Ring} {The Two Towers} {Return of the King}>`
Formal Definition of a Subsequence

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

<table>
<thead>
<tr>
<th>Data sequence</th>
<th>Subsequence</th>
<th>Contain?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;{2,4} {3,5,6} {8}&gt;)</td>
<td>(&lt;{2} {3,5}&gt;)</td>
<td>Yes</td>
</tr>
<tr>
<td>(&lt;{1,2} {3,4}&gt;)</td>
<td>(&lt;{1} {2}&gt;)</td>
<td>No</td>
</tr>
<tr>
<td>(&lt;{2,4} {2,4} {2,5}&gt;)</td>
<td>(&lt;{2} {4}&gt;)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- 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 $\minsup$).
Outline

- Sequence data
- **Sequential patterns mining**
- Basic algorithm for sequential pattern mining
- Advanced algorithms for sequential pattern mining
Sequential Pattern Mining: Definition

- **Given:**
  - a database of sequences
  - a user-specified minimum support threshold, \( \text{mins}up \)

- **Task:**
  - Find all subsequences with support \( \text{mins}up \)
Sequential Pattern Mining: Challenges

- Given a sequence: \(<\{a\ b\} \{c\ d\ e\} \{f\} \{g\ h\ i\}\>
- Examples of subsequences: \(<\{a\} \{c\ d\}\ {f\} \{g\}\>, \(<\{c\ d\ e\}\>, \(<\{b\} \{g\}\>, \text{ etc.}\>

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

The answer is:

\[
\binom{n}{k} = \binom{9}{4} = 126
\]
Sequential Pattern Mining: Challenges

- So, a **huge** number of possible sequential patterns are hidden in databases

- A mining algorithm should
  - find the **complete set of patterns**, when possible, satisfying the minimum support (frequency) threshold
  - be highly **efficient, scalable**, involving only a small number of database scans
  - be able to incorporate various kinds of **user-specific constraints**
Sequential Pattern Mining: Example (1)

Given a set of sequences, find the complete set of frequent subsequences

A sequence: < (ef) (ab) (df) cb >

A sequence database

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt;a(abc)(ac)d(cf)&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;(ad)c(bc)(ae)&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;(ef)(ab)(df)cb&gt;</td>
</tr>
<tr>
<td>40</td>
<td>&lt;eg(af)cbc&gt;</td>
</tr>
</tbody>
</table>

An element may contain a set of items. Items within an element are unordered and we list them alphabetically.

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

Given support threshold \( \text{min\_sup} = 2 \), <ab(c)> is a sequential pattern.
Sequential Pattern Mining: Example(2)

- **Minsup = 50%**
- **Examples of Frequent Subsequences:**
  - `< {1,2} > s=60%`
  - `< {2,3} > s=60%`
  - `< {2,4} > s=80%`
  - `< {3} {5} > s=80%`
  - `< {1} {2} > s=80%`
  - `< {2} {2} > s=60%`
  - `< {1} {2,3} > s=60%`
  - `< {2} {2,3} > s=60%`
  - `< {1,2} {2,3} > s=60%`

<table>
<thead>
<tr>
<th>Object</th>
<th>Timestamp</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1,2,4</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>2,3</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1,2</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2,3,4</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1,2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2,3,4</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>2,4,5</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>3,4</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>4,5</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>1,3</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>2,4,5</td>
</tr>
</tbody>
</table>
Sequential Pattern Mining: Applications

- Customer shopping sequences:
  - First buy computer, then CD-ROM, and then digital camera, within 3 months.
- Medical treatment, natural disasters (e.g., earthquakes), science & engineering processes, stocks and markets, etc.
- Telephone calling patterns, Weblog click streams
- DNA sequences and gene structures
Studies on Sequential Pattern Mining (early period)

- Concept introduction and an initial Apriori-like algorithm
  - R. Agrawal & R. Srikant. “Mining sequential patterns,” ICDE’95

- GSP—An Apriori-based, influential mining method (developed at IBM Almaden)

- From sequential patterns to episodes (Apriori-like + constraints)

- Mining sequential patterns with constraints
Outline

- Sequence data
- Sequential patterns
- Basic algorithm for sequential pattern mining
- Advanced algorithms for sequential pattern mining
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 $\rightarrow$ so do $<hab>$ and $<(ah)b>$

<table>
<thead>
<tr>
<th>Seq. ID</th>
<th>Sequence</th>
<th>support threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$(bd)cb(ac)$</td>
<td>$min_sup = 2$</td>
</tr>
<tr>
<td>20</td>
<td>$(bf)(ce)b(fg)$</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>$(ah)(bf)abf$</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>$(be)(ce)d$</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>$a(bd)bcb(ade)$</td>
<td></td>
</tr>
</tbody>
</table>
Extracting Sequential Patterns

Given \( n \) events: \( i_1, i_2, i_3, \ldots, i_n \)

**Candidate 1-subsequences:**
- \( \langle i_1 \rangle, \langle i_2 \rangle, \langle i_3 \rangle, \ldots, \langle i_n \rangle \)

**Candidate 2-subsequences:**
- \( \langle i_1, i_2 \rangle, \langle i_1, i_3 \rangle, \ldots, \langle i_1 \rangle \{i_1\}, \langle i_1 \rangle \{i_2\}, \ldots, \langle i_{n-1} \rangle \{i_n\} \)

**Candidate 3-subsequences:**
- \( \langle i_1, i_2, i_3 \rangle, \langle i_1, i_2, i_4 \rangle, \ldots, \langle i_1, i_2 \rangle \{i_1\}, \langle i_1, i_2 \rangle \{i_2\}, \ldots, \langle i_1 \rangle \{i_1, i_2\}, \langle i_1 \rangle \{i_1, i_3\}, \ldots, \langle i_1 \rangle \{i_1\} \{i_1\}, \langle i_1 \} \{i_1\} \{i_2\}, \ldots \)
GSP—A Generalized Sequential Pattern Mining Algorithm

- GSP (Generalized Sequential Pattern) mining algorithm
  - proposed by Agrawal and Srikant, EDBT’96
- Outline of the method
  - **Step 1:**
    - Make the first pass over the sequence database D to yield all the 1-element frequent sequences
  - **Step 2:** Repeat until no new frequent sequences are found
GSP—A Generalized Sequential Pattern Mining Algorithm

Outline of the method

Step 2: Repeat until no new frequent sequences are found

- **Candidate Generation:**
  - Merge pairs of frequent subsequences found in the \((k-1)th\) pass to generate candidate sequences that contain \(k\) items

- **Candidate Pruning:**
  - Prune candidate \(k\)-sequences that contain infrequent \((k-1)\)-subsequences

- **Support Counting:**
  - Make a new pass over the sequence database \(D\) to find the support for these candidate sequences

- **Candidate Elimination:**
  - Eliminate candidate \(k\)-sequences whose actual support is less than \(\text{minsup}\)
Candidate Generation

- **Base case (k=2):**
  - Merging two frequent 1-sequences \(<\{i_1\}\) and \(<\{i_2\}\) will produce two candidate 2-sequences: \(<\{i_1\} \{i_2\}\) and \(<\{i_1 \ i_2\}\>

- **General case (k>2):**
  - A frequent \((k-1)\)-sequence \(w_1\) is merged with another frequent \((k-1)\)-sequence \(w_2\) to produce a candidate \(k\)-sequence if the subsequence obtained by removing the first event in \(w_1\) is the same as the subsequence obtained by removing the last event in \(w_2\)
    - The resulting candidate after merging is given by the sequence \(w_1\) extended with the last event of \(w_2\)
      - If the last two events in \(w_2\) belong to the same element, then the last event in \(w_2\) becomes part of the last element in \(w_1\)
      - Otherwise, the last event in \(w_2\) becomes a separate element appended to the end of \(w_1\)
Candidate Generation

Examples

- Merging $w_1 = \{1\} \{2, 3\} \{4\}$ and $w_2 = \{2, 3\} \{4, 5\}$ produces the candidate sequence $\{1\} \{2, 3\} \{4, 5\}$ because the last two events in $w_2$ (4 and 5) belong to the same element.

- Merging $w_1 = \{1\} \{2, 3\} \{4\}$ and $w_2 = \{2, 3\} \{4\} \{5\}$ produces the candidate sequence $\{1\} \{2, 3\} \{4\} \{5\}$ because the last two events in $w_2$ (4 and 5) do not belong to the same element.

- We do not have to merge the sequences $w_1 = \{1\} \{2, 6\} \{4\}$ and $w_2 = \{1\} \{2\} \{4, 5\}$ to produce the candidate $\{1\} \{2, 6\} \{4, 5\}$ because if the latter is a viable candidate, then it can be obtained by merging $w_1$ with $\{2, 6\} \{4, 5\}$.
GSP Example

Frequent 3-sequences

Candidate Generation

Candidate Pruning

Data Mining: Tech. & Appl.
Finding Length-1 Sequential Patterns

- Examine GSP using an example
- Initial candidates: all singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan database once, count support for candidates

\[
\text{min}_\text{sup} = 2
\]

<table>
<thead>
<tr>
<th>Seq. ID</th>
<th>Sequence</th>
<th>Cand</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>(bd)cb(ac)</td>
<td>&lt;a&gt;</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>(bf)(ce)b(fg)</td>
<td>&lt;b&gt;</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>(ah)(bf)abf</td>
<td>&lt;c&gt;</td>
<td>4</td>
</tr>
<tr>
<td>40</td>
<td>(be)(ce)d</td>
<td>&lt;d&gt;</td>
<td>3</td>
</tr>
<tr>
<td>50</td>
<td>a(bd)bcb(ade)</td>
<td>&lt;e&gt;</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;f&gt;</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;g&gt;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;h&gt;</td>
<td>1</td>
</tr>
</tbody>
</table>

Data Mining: Tech. & Appl.
Generating Length-2 Candidates

51 length-2 Candidates

<table>
<thead>
<tr>
<th></th>
<th>&lt;a&gt;</th>
<th>&lt;b&gt;</th>
<th>&lt;c&gt;</th>
<th>&lt;d&gt;</th>
<th>&lt;e&gt;</th>
<th>&lt;f&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;a&gt;</td>
<td></td>
<td>&lt;aa&gt;</td>
<td>&lt;ab&gt;</td>
<td>&lt;ac&gt;</td>
<td>&lt;ad&gt;</td>
<td>&lt;ae&gt;</td>
</tr>
<tr>
<td>&lt;b&gt;</td>
<td>&lt;ba&gt;</td>
<td></td>
<td>&lt;bb&gt;</td>
<td>&lt;bc&gt;</td>
<td>&lt;bd&gt;</td>
<td>&lt;be&gt;</td>
</tr>
<tr>
<td>&lt;c&gt;</td>
<td>&lt;ca&gt;</td>
<td>&lt;cb&gt;</td>
<td></td>
<td>&lt;cc&gt;</td>
<td>&lt;cd&gt;</td>
<td>&lt;ce&gt;</td>
</tr>
<tr>
<td>&lt;d&gt;</td>
<td>&lt;da&gt;</td>
<td>&lt;db&gt;</td>
<td>&lt;dc&gt;</td>
<td></td>
<td>&lt;dd&gt;</td>
<td>&lt;de&gt;</td>
</tr>
<tr>
<td>&lt;e&gt;</td>
<td>&lt;ea&gt;</td>
<td>&lt;eb&gt;</td>
<td>&lt;ec&gt;</td>
<td>&lt;ed&gt;</td>
<td></td>
<td>&lt;ee&gt;</td>
</tr>
<tr>
<td>&lt;f&gt;</td>
<td>&lt;fa&gt;</td>
<td>&lt;fb&gt;</td>
<td>&lt;fc&gt;</td>
<td>&lt;fd&gt;</td>
<td>&lt;fe&gt;</td>
<td></td>
</tr>
</tbody>
</table>

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

Apriori prunes 44.57% candidates
Generating Length-3 Candidates and Finding Length-3 Patterns

- **Generate Length-3 Candidates**
  - **Self-join length-2 sequential patterns**
    - Based on the Apriori property
    - \(<ab>, <aa> \text{ and } <ba> \text{ are all length-2 sequential patterns} \rightarrow <aba> \text{ is a length-3 candidate}\)
    - \<(bd)>, <bb> \text{ and } <db> \text{ are all length-2 sequential patterns} \rightarrow <(bd)b> \text{ is a length-3 candidate}\)
  - 46 candidates are generated

- **Find Length-3 Sequential Patterns**
  - **Scan database once more, collect support counts for candidates**
  - 19 out of 46 candidates pass support threshold
The GSP Mining Process

5th scan: 1 cand. 1 length-5 seq. pat.  \[(bd)cba\]
4th scan: 8 cand. 6 length-4 seq. pat.  \[\text{\ldots} \quad \text{\ldots} \quad \text{\ldots} \quad \text{\ldots} \quad \text{\ldots} \quad \text{\ldots}\]
3rd scan: 46 cand. 19 length-3 seq. pat. 20 cand. not in DB at all  \[\text{\ldots} \quad \text{\ldots} \quad \text{\ldots} \quad \text{\ldots} \quad \text{\ldots} \quad \text{\ldots}\]
2nd scan: 51 cand. 19 length-2 seq. pat. 10 cand. not in DB at all  \[\text{\ldots} \quad \text{\ldots} \quad \text{\ldots} \quad \text{\ldots} \quad \text{\ldots} \quad \text{\ldots}\]
1st scan: 8 cand. 6 length-1 seq. pat.

<table>
<thead>
<tr>
<th>Seq. ID</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>[(bd)cb(ac)]</td>
</tr>
<tr>
<td>20</td>
<td>[(bf)(ce)b(fg)]</td>
</tr>
<tr>
<td>30</td>
<td>[(ah)(bf)abf]</td>
</tr>
<tr>
<td>40</td>
<td>[(be)(ce)d]</td>
</tr>
<tr>
<td>50</td>
<td>[a(bd)bcb(ade)]</td>
</tr>
</tbody>
</table>

\[\text{min\_sup} = 2\]
Time Constraints (I)

\[
\begin{array}{c}
\{A, B\} \quad \{C\} \quad \{D, E\} \\
\begin{array}{c}
\leq x_g \\
\leq m_g
\end{array}
\begin{array}{c}
> n_g
\end{array}
\end{array}
\]

\[x_g = 2, \ n_g = 0, \ m_g = 4\]

<table>
<thead>
<tr>
<th>Data sequence, d</th>
<th>Sequential Pattern, s</th>
<th>d contains s?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;{2,4} {3,5,6} {4,7} {4,5} {8}&gt;)</td>
<td>(&lt;{6} {5}&gt;)</td>
<td>Yes</td>
</tr>
<tr>
<td>(&lt;{1} {2} {3} {4} {5}&gt;)</td>
<td>(&lt;{1} {4}&gt;)</td>
<td>No</td>
</tr>
<tr>
<td>(&lt;{1} {2,3} {3,4} {4,5}&gt;)</td>
<td>(&lt;{2} {3} {5}&gt;)</td>
<td>Yes</td>
</tr>
<tr>
<td>(&lt;{1,2} {3} {2,3} {3,4} {2,4} {4,5}&gt;)</td>
<td>(&lt;{1,2} {5}&gt;)</td>
<td>No</td>
</tr>
</tbody>
</table>

Data Mining: Tech. & Appl.
Mining Sequential Patterns with Time Constraints

- **Approach 1:**
  - Mine sequential patterns without timing constraints
  - Postprocess the discovered patterns

- **Approach 2:**
  - Modify GSP to directly prune candidates that violate timing constraints

**Question:**
- Does Apriori principle still hold?
Apriori Principle for Sequence Data

<table>
<thead>
<tr>
<th>Object</th>
<th>Timestamp</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1,2,4</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>2,3</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1,2</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2,3,4</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1, 2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2,3,4</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>2,4,5</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>3, 4</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>4, 5</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>1, 3</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>2, 4, 5</td>
</tr>
</tbody>
</table>

Suppose:
\[ x_g = 1 \text{ (max-gap)} \]
\[ n_g = 0 \text{ (min-gap)} \]
\[ m_s = 5 \text{ (maximum span)} \]
\[ \text{minsup} = 60\% \]

\{2\} \{5\} \text{ support} = 40\% 
but
\{2\} \{3\} \{5\} \text{ support} = 60\%

Problem exists because of max-gap constraint
No such problem if max-gap is infinite
Contiguous Subsequences

- $s$ is a contiguous subsequence of $w = <e_1><e_2>...<e_k>$, if any of the following conditions hold:
  - $s$ is obtained from $w$ by deleting an item from either $e_1$ or $e_k$
  - $s$ is obtained from $w$ by deleting an item from any element $e_i$ that contains at least 2 items
  - $s$ is a contiguous subsequence of $s'$ and $s'$ is a contiguous subsequence of $w$ (recursive definition)

- Examples: $s = <\{1\}\{2\}>$
  - is a contiguous subsequence of $<\{1\}\{2\}3>$, $<\{1\}2\{2\}\{3\}>$, and $<\{3\}4\{1\}\{2\}\{2\}3\{4\}>$
  - is not a contiguous subsequence of $<\{1\}\{3\}\{2\}>$ and $<\{2\}\{1\}\{3\}\{2\}>$
Modified Candidate Pruning Step

- **Without maxgap constraint:**
  - A candidate \( k \)-sequence is pruned if at least one of its \((k-1)\)-subsequences is infrequent

- **With maxgap constraint:**
  - A candidate \( k \)-sequence is pruned if at least one of its **contiguous** \((k-1)\)-subsequences is infrequent
### Time Constraints (II)

- \( x_g \): max-gap
- \( n_g \): min-gap
- \( ws \): window size
- \( m_s \): maximum span

\[ x_g = 2, \quad n_g = 0, \quad ws = 1, \quad m_s = 5 \]

<table>
<thead>
<tr>
<th>Data sequence, ( d )</th>
<th>Sequential Pattern, ( s )</th>
<th>( d ) contains ( s )?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; {2, 4} {3, 5, 6} {4, 7} {4, 5} {8}&gt;)</td>
<td>(&lt; {3, 4, 5}&gt;)</td>
<td>Yes</td>
</tr>
<tr>
<td>(&lt; {1} {2} {3} {4} {5}&gt;)</td>
<td>(&lt; {1, 2} {3, 4}&gt;)</td>
<td>No</td>
</tr>
<tr>
<td>(&lt; {1, 2} {2, 3} {3, 4} {4, 5}&gt;)</td>
<td>(&lt; {1, 2} {3, 4}&gt;)</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Modified Support Counting Step

- Given a candidate sequential pattern: \( \langle \{a, c\} \rangle \)
  - Any data sequences that contain
    - \( \langle \ldots \{a, c\} \ldots \rangle \),
    - \( \langle \ldots \{a\} \ldots \{c\} \ldots \rangle \) (where \( \text{time}(\{c\}) - \text{time}(\{a\}) \leq ws \))
    - \( \langle \ldots \{c\} \ldots \{a\} \ldots \rangle \) (where \( \text{time}(\{a\}) - \text{time}(\{c\}) \leq ws \))
  - will contribute to the support count of candidate pattern
Other Formulation

- In some domains, we may have only one very long time series
  - Example:
    - monitoring network traffic events for attacks
    - monitoring telecommunication alarm signals
  - Goal is to find frequent sequences of events in the time series
    - This problem is also known as frequent episode mining
General Support Counting Schemes

**COBJ**: one occurrence per object;
**CWIN**: one occurrence per sliding window;
**CMINWIN**: number of minimal windows of occurrence;
**CDIST_O**: distinct occurrences with possibility of event-timestamp overlap;
**CDIST**: distinct occurrences with no event-timestamp overlap.

Refer to P.-N. Tan, M. Steinbach, and V. Kumar’s text book “Introduction to Data Mining”
Outline

- Sequence data
- Sequential patterns
- Basic algorithms for sequential pattern mining
- Advanced algorithms for sequential pattern mining
# What we covered so far

<table>
<thead>
<tr>
<th></th>
<th>Read-based</th>
<th>Write-based</th>
<th>Point-based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Association Mining</strong></td>
<td>Apriori[AgSr94]</td>
<td>Eclat, MaxClique[Zaki01], FP Growth [HaPe00]</td>
<td>Hmine</td>
</tr>
<tr>
<td><strong>Sequential Pattern Discovery</strong></td>
<td>GSP[AgSr96]</td>
<td>SPADE [Zaki98,Zaki01], PrefixSpan [PHPC01]</td>
<td></td>
</tr>
<tr>
<td><strong>Iceberg Cube</strong></td>
<td>Apriori[AgSr94]</td>
<td></td>
<td>BUC[BeRa99], H-cubing [HPDW01]</td>
</tr>
</tbody>
</table>
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!
    \[\sum_{i=1}^{100} \binom{100}{i} = 2^{100} - 1 \approx 10^{30}\]
FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining

- A divide-and-conquer approach
  - Recursively *project* a sequence database into a set of smaller databases based on the current set of frequent patterns
  - Mine each projected database to find its patterns

- J. Han J. Pei, B. Mortazavi-Asi, Q. Chen, U. Dayal, M.C. Hsu, FreeSpan: Frequent pattern-projected sequential pattern mining. In KDD’00.

\[
\text{f_list: } b\!:5, c\!:4, a\!:3, d\!:3, e\!:3, f\!:2
\]

All seq. pat. can be divided into 6 subsets:
- Seq. pat. containing item \( f \)
- Those containing \( d \) but no \( e \) nor \( f \)
- Those containing \( a \) but no \( d, e \) or \( f \)
- Those containing \( c \) but no \( a, d, e \) or \( f \)
- Those containing only item \( b \)

<table>
<thead>
<tr>
<th>Sequence Database</th>
<th>SDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; (bd)) c b (ac) &gt;</td>
<td></td>
</tr>
<tr>
<td>(&lt; (bf) (ce)) b (fg) &gt;</td>
<td></td>
</tr>
<tr>
<td>(&lt; (ah) (bf)) a b f &gt;</td>
<td></td>
</tr>
<tr>
<td>(&lt; (be) (ce)) d &gt;</td>
<td></td>
</tr>
<tr>
<td>(&lt; a (bd)) b c b (ade) &gt;</td>
<td></td>
</tr>
</tbody>
</table>
From FreeSpan to PrefixSpan: Why?

- **FreeSpan:**
  - Projection-based: No candidate sequence needs to be generated
  - But, projection can be performed at any point in the sequence, and the projected sequences do not shrink much

- **PrefixSpan**
  - Projection-based
  - But only prefix-based projection: less projections and quickly shrinking sequences
Prefix and Suffix (Projection)

- `<a>`, `<aa>`, `<a(ab)>` and `<a(abc)>` are *prefixes* of sequence `<a(abc)(ac)d(cf)>`
- *Given* sequence `<a(abc)(ac)d(cf)>`

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Suffix (Prefix-Based Projection)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;(abc)(ac)d(cf)&gt;</code></td>
</tr>
<tr>
<td><code>&lt;aa&gt;</code></td>
<td><code>&lt;(_bc)(ac)d(cf)&gt;</code></td>
</tr>
<tr>
<td><code>&lt;ab&gt;</code></td>
<td><code>&lt;(_c)(ac)d(cf)&gt;</code></td>
</tr>
</tbody>
</table>
Mining Sequential Patterns by Prefix Projections

- **Step 1:** find length-1 sequential patterns
  - \(<a>, <b>, <c>, <d>, <e>, <f>\)

- **Step 2:** divide search space. The complete set of seq. pat. can be partitioned into 6 subsets:
  - The ones having prefix \(<a>\);
  - The ones having prefix \(<b>\);
  - ...
  - The ones having prefix \(<f>\)

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>(&lt;a(abc)(ac)d(cf)&gt;)</td>
</tr>
<tr>
<td>20</td>
<td>(&lt;(ad)c(bc)(ae)&gt;)</td>
</tr>
<tr>
<td>30</td>
<td>(&lt;(ef)(ab)(df)cb&gt;)</td>
</tr>
<tr>
<td>40</td>
<td>(&lt;eg(af)cbc&gt;)</td>
</tr>
</tbody>
</table>

Data Mining: Tech. & Appl.
Finding Seq. Patterns with Prefix <a>

- Only need to consider projections w.r.t. <a>
  - <a>-projected database: <(abc)(ac)d(cf)>, <(_d)c(bc)(ae)>, <(_b)(df)cb>, <(_f)cbc>

- Find all the length-2 seq. pat. Having prefix <a>: <aa>, <ab>, <(ab)>, <ac>, <ad>, <af>
  - Further partition into 6 subsets
    - Having prefix <aa>
    - ...
    - Having prefix <af>

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt;a(abc)(ac)d(cf)&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;(ad)c(bc)(ae)&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;(ef)(ab)(df)cb&gt;</td>
</tr>
<tr>
<td>40</td>
<td>&lt;eg(af)cbc&gt;</td>
</tr>
</tbody>
</table>

Data Mining: Tech. & Appl.
Completeness of PrefixSpan

SDB

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&lt;a(abc)(ac)d(cf)&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;(ad)c(bc)(ae)&gt;</td>
</tr>
<tr>
<td>30</td>
<td>(ef)(ab)(df)cb</td>
</tr>
<tr>
<td>40</td>
<td>&lt;eg(af)cbc&gt;</td>
</tr>
</tbody>
</table>

Length-1 sequential patterns

- <a>, <b>, <c>, <d>, <e>, <f>

Length-2 sequential patterns

- <aa>, <ab>, <ac>, <ad>, <af>

Having prefix <a>

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

Having prefix <b>

- <b>-projected database
  - ...
Efficiency of PrefixSpan

- No candidate sequence needs to be generated
- Projected databases keep shrinking
- Major cost of PrefixSpan: constructing projected databases
  - Can be improved by bi-level projections
Optimization Techniques in PrefixSpan

- Physical projection vs. pseudo-projection
  - Pseudo-projection may reduce the effort of projection when the projected database fits in main memory

- Parallel projection vs. partition projection
  - Partition projection may avoid the blowup of disk space
Speed-up by Pseudo-projection

- Major cost of PrefixSpan: projection
  - Postfixes of sequences often appear repeatedly in recursive projected databases

- When (projected) database can be held in main memory, use pointers to form projections
  - Pointer to the sequence
  - Offset of the postfix

$$s = \langle a(abc)(ac)d(cf) \rangle$$

$$s|<a>: ( , 2) \quad \langle (abc)(ac)d(cf) \rangle$$

$$s|<ab>: ( , 4) \quad \langle (_c)(ac)d(cf) \rangle$$
Pseudo-Projection vs. Physical Projection

- Pseudo-projection avoids physically copying postfixes
  - Efficient in running time and space when database can be held in main memory
- However, it is not efficient when database cannot fit in main memory
  - Disk-based random accessing is very costly
- Suggested Approach:
  - Integration of physical and pseudo-projection
  - Swapping to pseudo-projection when the data set fits in memory
PrefixSpan is Faster than GSP and FreeSpan

Data Mining: Tech. & Appl.
Effect of Pseudo-Projection

Data Mining: Tech. & Appl.