Sequence Mining

Kevin Duh

2015-05-26
Today’s Agenda

Review of Apriori Algorithm

Sequence Mining

PrefixSpan Algorithm
Recall: Frequent Itemset Mining

- Given a finite set of items \( \{A, B, C, \ldots \} \)
Recall: Frequent Itemset Mining

- Given a finite set of items \( \{A, B, C, \ldots \} \)
- in several baskets, e.g.
  - Basket 1: \( \{A, B, D\} \)
  - Basket 2: \( \{A, B, C, E\} \)
  - Basket 3: \( \{B, E, F\} \)
  - Basket 4: \( \{A, B, E, F\} \)
Recall: Frequent Itemset Mining

- Given a finite set of items \( \{A, B, C, \ldots \} \)
- in several baskets, e.g.
  - Basket 1: \( \{A, B, D\} \)
  - Basket 2: \( \{A, B, C, E\} \)
  - Basket 3: \( \{B, E, F\} \)
  - Basket 4: \( \{A, B, E, F\} \)

- The support of itemset \( I \) is the number of baskets that contain \( I \)
Recall: Frequent Itemset Mining

- Given a finite set of **items** \( \{A, B, C, \ldots \} \)
- in several **baskets**, e.g.
  - Basket 1: \( \{A, B, D\} \)
  - Basket 2: \( \{A, B, C, E\} \)
  - Basket 3: \( \{B, E, F\} \)
  - Basket 4: \( \{A, B, E, F\} \)
- The **support** of itemset \( I \) is the number of baskets that contain \( I \)
- Goal: Find all **frequent itemsets**, i.e. sets of items with support \( \geq s \)
Example

- We are given:
  - Basket 1: \{A, B, D\}
  - Basket 2: \{A, B, C, E\}
  - Basket 3: \{B, E, F\}
  - Basket 4: \{A, B, E, F\}
Example

- We are given:
  - Basket 1: \{A, B, D\}
  - Basket 2: \{A, B, C, E\}
  - Basket 3: \{B, E, F\}
  - Basket 4: \{A, B, E, F\}

- 1-item Itemsets & their support:
  - \{A\}: 3, \{B\}: 4, \{C\}: 1, \{D\}: 1, \{E\}: 3, \{F\}: 2
Example

- We are given:
  - Basket 1: \{A, B, D\}
  - Basket 2: \{A, B, C, E\}
  - Basket 3: \{B, E, F\}
  - Basket 4: \{A, B, E, F\}

- 1-item Itemsets & their support:
  - \{A\}: 3, \{B\}: 4, \{C\}: 1, \{D\}: 1, \{E\}: 3, \{F\}: 2

- 2-item Itemsets & their support:
  - \{A, B\}: 3, \{A, C\}: 1, \{A, D\}: 1, \{A, E\}: 2, \{A, F\}: 1, \{B, C\}: 1, \{B, D\}: 1, ...
Example

- We are given:
  - Basket 1: \{A, B, D\}
  - Basket 2: \{A, B, C, E\}
  - Basket 3: \{B, E, F\}
  - Basket 4: \{A, B, E, F\}

- 1-item Itemsets & their support:
  - \{A\}: 3, \{B\}: 4, \{C\}: 1, \{D\}: 1, \{E\}: 3, \{F\}: 2

- 2-item Itemsets & their support:
  - \{A, B\}: 3, \{A, C\}: 1, \{A, D\}: 1, \{A, E\}: 2, \{A, F\}: 1, \{B, C\}: 1, \{B, D\}: 1, ...

- 3-item Itemsets & their support:
  - \{A, B, C\}: 1, \{A, B, D\}: 1, \{A, B, E\}: 1, \{A, B, F\}: 1, \{A, C, D\}: 0, ...
Monotonicity Principle

- If $I$ is not frequent, then no superset of $I$ can be frequent.
- Apriori Algorithm exploits this: Smart enumeration of itemset.
Apriori Algorithm

Alternate between:

- $L_k$: set of **truly frequent** itemsets of size $k$
- $C_k$: set of **candidate** itemsets of size $k$
  - constructed from $L_{k-1}$, avoids all possible enumerations

Figure from Rajamaran et. al., Mining of Massive Datasets, chapter 6
Apriori Algorithm (example run)

- Find frequent itemsets ($s = 3$):
  - Basket 1: \{A, B, D\}
  - Basket 2: \{A, B, C, E\}
  - Basket 3: \{B, E, F\}
  - Basket 4: \{A, B, E, F\}

1 First pass (1-item itemsets)
  - $C_1$: \{A\}:3, \{B\}:4, \{C\}:1, \{D\}:1, \{E\}:3, \{F\}:2
  - $L_1$: \{A\}, \{B\}, \{E\}
Apriori Algorithm (example run)

- Find frequent itemsets \( s = 3 \):
  - Basket 1: \{A, B, D\}
  - Basket 2: \{A, B, C, E\}
  - Basket 3: \{B, E, F\}
  - Basket 4: \{A, B, E, F\}

1. First pass (1-item itemsets)
   - \( C_1 \): \{A\}:3, \{B\}:4, \{C\}:1, \{D\}:1, \{E\}:3, \{F\}:2
   - \( L_1 \): \{A\}, \{B\}, \{E\}

2. Second pass (2-item itemsets)
   - \( C_2 \): \{A, B\}: 3, \{A, E\}: 2, \{B, E\}: 3
   - \( L_2 \): \{A, B\}, \{B, E\}

3. Third pass (3-item itemsets)
   - \( C_3 \): \{A, B, E\}: 2; \( L_3 \) : \emptyset
Today’s Agenda

Review of Apriori Algorithm

Sequence Mining

PrefixSpan Algorithm
From Itemsets to Sequences

- Itemset Mining
  - Purchase 1: \{camera, US B\}
  - Purchase 3: \{printer, paper\}
  - Purchase 4: \{ink, paper\}
From Itemsets to Sequences

- **Itemset Mining**
  - Purchase 1: \{camera, US B\}
  - Purchase 3: \{printer, paper\}
  - Purchase 4: \{ink, paper\}

- **Sequence Mining:**
  - Customer 1: \langle\{camera, US B\}, \{printer\}\rangle
  - Customer 2: \langle\{camera\}, \{printer\}, \{ink\}\rangle
From Itemsets to Sequences

- Itemset Mining
  - Purchase 1: \{\textit{camera}, \textit{US B}\}
  - Purchase 3: \{\textit{printer}, \textit{paper}\}
  - Purchase 4: \{\textit{ink}, \textit{paper}\}

- Sequence Mining:
  - Customer 1: \langle\{\textit{camera}, \textit{US B}\}, \{\textit{printer}\}\rangle
  - Customer 2: \langle\{\textit{camera}\}, \{\textit{printer}\}, \{\textit{ink}\}\rangle

- Customers who bought camera are likely to buy printer later
Problem Definition

- A Sequence is an **ordered list** of itemsets:
  - Customer 1: \(<\{\text{camera, USB}\}, \{\text{printer}\}\>\)
  - Customer 2: \(<\{\text{camera}\}, \{\text{printer}\}, \{\text{ink}\}\}>\)
  - Customer \(n\): \(<I_1, I_2, I_3, ...>\)

- Goal: Find frequent sub-sequences with support \(\geq s\)
  - i.e. more than \(s\) customers exhibit this buying behavior
This has 5 itemsets (aka “events”)
\langle \{A\}, \{A, B, C\}, \{A, C\}, \{D\}, \{C, F\}\rangle

- This has 5 itemsets (aka “events”)
- This has 9 items total, so is called a length-9 sequence
\langle \{A\}, \{A, B, C\}, \{A, C\}, \{D\}, \{C, F\}\rangle

- This has 5 itemsets (aka “events”)
- This has 9 items total, so is called a length-9 sequence
- Item A occurs 3 times. It contributes 3 to the length but only 1 to the support
\[ \{\{A\}, \{A, B, C\}, \{A, C\}, \{D\}, \{C, F\}\} \]

- This has 5 itemsets (aka “events”)
- This has 9 items total, so is called a length-9 sequence
- Item A occurs 3 times. It contributes 3 to the length but only 1 to the support
- Sub-sequences include:
  - \( \{\{A, B, C\}, \{D\}\} \)
  - \( \{\{A\}, \{B, C\}, \{C\}, \{D\}, \{C, F\}\} \)
  - \( \{\{A\}, \{B, C\}, \{D\}, \{F\}\} \)
\{\{A\}, \{A, B, C\}, \{A, C\}, \{D\}, \{C, F\}\}

- This has 5 itemsets (aka “events”)
- This has 9 items total, so is called a length-9 sequence
- Item A occurs 3 times. It contributes 3 to the length but only 1 to the support
- Sub-sequences include:
  - \{\{A, B, C\}, \{D\}\}
  - \{\{A\}, \{B, C\}, \{C\}, \{D\}, \{C, F\}\}
  - \{\{A\}, \{B, C\}, \{D\}, \{F\}\}
- But not: \{\{D\}, \{A, B, C\}\}, etc.
From here on, for simplicity...

- We only consider sequences with 1-item events
- e.g. \( \langle\{A\}, \{A\}, \{C\}, \{D\}, \{F\}\rangle \)
  written as: \( \langle A, A, C, D, F\rangle \)
From here on, for simplicity...

- We only consider sequences with 1-item events
- e.g. \(\langle\{A\}, \{A\}, \{C\}, \{D\}, \{F\}\rangle\)
  written as: \(\langle A, A, C, D, F\rangle\)
- Suitable for sequence data such as text, DNA, browsing history
Example

- Extract frequent sub-sequence ($s = 3$)

1. $\langle A, A, A, C, C \rangle$
2. $\langle B, C, B, C, B \rangle$
3. $\langle A, D, C, A, A, B \rangle$
4. $\langle A, C, B, C, A, A \rangle$
Example

- Extract frequent sub-sequence \((s = 3)\)
  1. \(\langle A, A, A, C, C \rangle\)
  2. \(\langle B, C, B, C, B \rangle\)
  3. \(\langle A, D, C, A, A, B \rangle\)
  4. \(\langle A, C, B, C, A, A \rangle\)

- Frequent sub-sequences include:
  - \(\langle A \rangle\)
  - \(\langle A, A \rangle\)
  - \(\langle A, A, A \rangle\)
  - \(\langle A, C \rangle\)
Applying the Apriori Algorithm

- Extract frequent sub-sequence \( s = 3 \)

1. \( \langle A, A, A, C, C \rangle \)
2. \( \langle B, C, B, C, B \rangle \)
3. \( \langle A, D, C, A, A, B \rangle \)
4. \( \langle A, C, B, C, A, A \rangle \)
Applying the Apriori Algorithm

- Extract frequent sub-sequence \( s = 3 \)
  1. \( \langle A, A, A, C, C \rangle \)
  2. \( \langle B, C, B, C, B \rangle \)
  3. \( \langle A, D, C, A, A, B \rangle \)
  4. \( \langle A, C, B, C, A, A \rangle \)

- 1st Pass:
  - \( C_1 : \langle A \rangle, \langle B \rangle, \langle C \rangle, \langle D \rangle \)
Applying the Apriori Algorithm

- Extract frequent sub-sequence \((s = 3)\)
  1. \(\langle A, A, A, C, C \rangle\)
  2. \(\langle B, C, B, C, B \rangle\)
  3. \(\langle A, D, C, A, A, B \rangle\)
  4. \(\langle A, C, B, C, A, A \rangle\)

- 1st Pass:
  - \(C_1 : \langle A \rangle, \langle B \rangle, \langle C \rangle, \langle D \rangle\)
  - \(L_1 : \langle A \rangle, \langle B \rangle, \langle C \rangle\)
Applying the Apriori Algorithm

- Extract frequent sub-sequence ($s = 3$)
  1. $\langle A, A, A, C, C \rangle$
  2. $\langle B, C, B, C, B \rangle$
  3. $\langle A, D, C, A, A, B \rangle$
  4. $\langle A, C, B, C, A, A \rangle$

- 1st Pass:
  - $C_1 : \langle A \rangle, \langle B \rangle, \langle C \rangle, \langle D \rangle$
  - $L_1 : \langle A \rangle, \langle B \rangle, \langle C \rangle$

- 2nd Pass:
  - $C_2 : 3 \times 3$ candidates,
    $\langle A, A \rangle, \langle A, B \rangle, \langle A, C \rangle,$
    $\langle B, A \rangle, \langle B, B \rangle, \langle B, C \rangle, \langle C, A \rangle, \langle C, B \rangle, \langle C, C \rangle$
Applying the Apriori Algorithm

- Extract frequent sub-sequence \((s = 3)\)
  1. \(\langle A, A, A, C, C \rangle\)
  2. \(\langle B, C, B, C, B \rangle\)
  3. \(\langle A, D, C, A, A, B \rangle\)
  4. \(\langle A, C, B, C, A, A \rangle\)

- 1st Pass:
  - \(C_1 : \langle A \rangle, \langle B \rangle, \langle C \rangle, \langle D \rangle\)
  - \(L_1 : \langle A \rangle, \langle B \rangle, \langle C \rangle\)

- 2nd Pass:
  - \(C_2 : 3 \times 3\) candidates,
    - \(\langle A, A \rangle, \langle A, B \rangle, \langle A, C \rangle, \langle B, A \rangle, \langle B, B \rangle, \langle B, C \rangle, \langle C, A \rangle, \langle C, B \rangle, \langle C, C \rangle\)
  - \(L_2 : ?\)
Issues with the Apriori Algorithm

- We still need to generate many candidates
- For each candidate, we need to scan the entire dataset
Issues with the Apriori Algorithm

- We still need to generate many candidates
- For each candidate, we need to scan the entire dataset

Next, we present the PrefixSpan algorithm.

- An instance of a family of algorithms called Frequent-Pattern (FP) Growth that addresses the above issues.
Today’s Agenda

Review of Apriori Algorithm

Sequence Mining

PrefixSpan Algorithm
## Prefix & Suffix

\[ \langle A, A, A, C, C \rangle \]

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>\langle A \rangle</td>
<td>\langle A, A, C, C \rangle</td>
</tr>
<tr>
<td>\langle A, A \rangle</td>
<td>\langle A, C, C \rangle</td>
</tr>
<tr>
<td>\langle A, A, A \rangle</td>
<td>\langle C, C \rangle</td>
</tr>
<tr>
<td>\langle A, A, A, C \rangle</td>
<td>\langle C \rangle</td>
</tr>
</tbody>
</table>
PrefixSpan Algorithm (main idea)

- Divide & Conquer:
  1. First find length-1 frequent sequences. Suppose there are $m$ such cases.
PrefixSpan Algorithm (main idea)

- Divide & Conquer:
  1. First find length-1 frequent sequences. Suppose there are $m$ such cases.
  2. The complete set of frequent patterns can be partitioned into $m$ subsets, each subset having the same prefix.
PrefixSpan Algorithm (main idea)

- Divide & Conquer:
  1. First find length-1 frequent sequences. Suppose there are $m$ such cases.
  2. The complete set of frequent patterns can be partitioned into $m$ subsets, each subset having the same prefix.
  3. Each partition is mined separately. This process is done recursively.
PrefixSpan Algorithm (main idea)

- Divide & Conquer:
  1. First find length-1 frequent sequences. Suppose there are \( m \) such cases.
  2. The complete set of frequent patterns can be partitioned into \( m \) subsets, each subset having the same prefix.
  3. Each partition is mined separately. This process is done recursively.
PrefixSpan Algorithm (main idea)

- Divide & Conquer:
  1. First find length-1 frequent sequences. Suppose there are \( m \) such cases.
  2. The complete set of frequent patterns can be partitioned into \( m \) subsets, each subset having the same prefix.
  3. Each partition is mined separately. This process is done recursively.

- Each partition is a (smaller) "projected" database
Projected database

- **Original database:**
  1. \(\langle A, A, A, C, C \rangle\)
  2. \(\langle B, C, B, C, B \rangle\)
  3. \(\langle A, D, C, A, A, B \rangle\)
  4. \(\langle A, C, B, C, A, A \rangle\)

- **Projected database of Prefix \(\langle A \rangle\):**
  1. \(\langle A, A, C, C \rangle\)
  2. \(\emptyset\)
  3. \(\langle D, C, A, A, B \rangle\)
  4. \(\langle C, B, C, A, A \rangle\)
- Original database:
  1. \( \langle A, A, A, C, C \rangle \)
  2. \( \langle B, C, B, C, B \rangle \)
  3. \( \langle A, D, C, A, A, B \rangle \)
  4. \( \langle A, C, B, C, A, A \rangle \)

- Projected database of Prefix \( \langle C \rangle \):
Original database:

1. \langle A, A, A, C, C \rangle
2. \langle B, C, B, C, B \rangle
3. \langle A, D, C, A, A, B \rangle
4. \langle A, C, B, C, A, A \rangle

Projected database of Prefix \langle C \rangle:

1. \langle C \rangle
- Original database:
  1 \langle A, A, A, C, C \rangle
  2 \langle B, C, B, C, B \rangle
  3 \langle A, D, C, A, A, B \rangle
  4 \langle A, C, B, C, A, A \rangle

- Projected database of Prefix \langle C \rangle:
  1 \langle C \rangle
  2 \langle B, C, B \rangle
- Original database:
  1. \langle A, A, A, C, C \rangle
  2. \langle B, C, B, C, B \rangle
  3. \langle A, D, C, A, A, B \rangle
  4. \langle A, C, B, C, A, A \rangle

- Projected database of Prefix \langle C \rangle:
  1. \langle C \rangle
  2. \langle B, C, B \rangle
  3. \langle A, A, B \rangle
- Original database:
  1. \langle A, A, A, C, C \rangle
  2. \langle B, C, B, C, B \rangle
  3. \langle A, D, C, A, A, B \rangle
  4. \langle A, C, B, C, A, A \rangle

- Projected database of Prefix \langle C \rangle:
  1. \langle C \rangle
  2. \langle B, C, B \rangle
  3. \langle A, A, B \rangle
  4. \langle B, C, A, A \rangle
Original database:

1. \langle A, A, A, C, C \rangle
2. \langle B, C, B, C, B \rangle
3. \langle A, D, C, A, A, B \rangle
4. \langle A, C, B, C, A, A \rangle

Projected database of Prefix \langle C \rangle:

1. \langle C \rangle
2. \langle B, C, B \rangle
3. \langle A, A, B \rangle
4. \langle B, C, A, A \rangle

Trick: Frequent items in projected database combines with Prefix \langle C \rangle to form frequent length-2 sequence!

- If $B$ is frequent, then so is \langle C, B \rangle
- If $C$ is frequent, then so is \langle C, C \rangle
PrefixSpan Algorithm (example run)

- Extract frequent sub-sequence \( s = 3 \)

1. \( \langle A, A, A, C, C \rangle \)
2. \( \langle B, C, B, C, B \rangle \)
3. \( \langle A, D, C, A, A, B \rangle \)
4. \( \langle A, C, B, C, A, A \rangle \)
PrefixSpan Algorithm (example run)

- Extract frequent sub-sequence \( s = 3 \)
  1. \( \langle A, A, A, C, C \rangle \)
  2. \( \langle B, C, B, C, B \rangle \)
  3. \( \langle A, D, C, A, A, B \rangle \)
  4. \( \langle A, C, B, C, A, A \rangle \)

- 1st pass: \( A : 3, B : 3, C : 4, D : 1 \)
PrefixSpan Algorithm (example run)

- Extract frequent sub-sequence ($s = 3$)
  1. $\langle A, A, A, C, C \rangle$
  2. $\langle B, C, B, C, B \rangle$
  3. $\langle A, D, C, A, A, B \rangle$
  4. $\langle A, C, B, C, A, A \rangle$

- 1st pass: $A : 3$, $B : 3$, $C : 4$, $D : 1$
  - Frequent length-1 seq: $\langle A \rangle, \langle B \rangle, \langle C \rangle$
PrefixSpan Algorithm (example run)

- Extract frequent sub-sequence ($s = 3$)
  
  1. $\langle A, A, A, C, C \rangle$
  2. $\langle B, C, B, C, B \rangle$
  3. $\langle A, D, C, A, A, B \rangle$
  4. $\langle A, C, B, C, A, A \rangle$

- 1st pass: $A : 3, B : 3, C : 4, D : 1$
  
  - Frequent length-1 seq: $\langle A \rangle, \langle B \rangle, \langle C \rangle$
  - No frequent seq (any length) w/ prefix $D$
PrefixSpan Algorithm (example run)

- Extract frequent sub-sequence ($s = 3$)
  1. $\langle A, A, A, C, C \rangle$
  2. $\langle B, C, B, C, B \rangle$
  3. $\langle A, D, C, A, A, B \rangle$
  4. $\langle A, C, B, C, A, A \rangle$

- 1st pass: $A : 3, B : 3, C : 4, D : 1$
  - Frequent length-1 seq: $\langle A \rangle, \langle B \rangle, \langle C \rangle$
  - No frequent seq (any length) w/ prefix $D$

- Projected database with Prefix $\langle A \rangle$:
  1. $\langle A, A, C, C \rangle$
  2. $\emptyset$
  3. $\langle D, C, A, A, B \rangle$
  4. $\langle C, B, C, A, A \rangle$
Projected database with Prefix \( \langle A \rangle \):

1. \( \langle A, A, C, C \rangle \)
2. \( \emptyset \)
3. \( \langle D, C, A, A, B \rangle \)
4. \( \langle C, B, C, A, A \rangle \)
- Projected database with Prefix $\langle A \rangle$:
  1. $\langle A, A, C, C \rangle$
  2. $\emptyset$
  3. $\langle D, C, A, A, B \rangle$
  4. $\langle C, B, C, A, A \rangle$

- Frequent items ($s = 3$): $A: 3$, $B: 2$, $C: 3$
  - Frequent length-2 seq: $\langle A, A \rangle, \langle A, C \rangle$
- Projected database with Prefix $\langle A \rangle$:

1. $\langle A, A, C, C \rangle$
2. $\emptyset$
3. $\langle D, C, A, A, B \rangle$
4. $\langle C, B, C, A, A \rangle$

- Frequent items ($s = 3$): A: 3, B: 2, C: 3
  - Frequent length-2 seq: $\langle A, A \rangle, \langle A, C \rangle$

- Projected database with Prefix $\langle A, A \rangle$:

1. $\langle A, C, C \rangle$
2. $\emptyset$
3. $\langle A, B \rangle$
4. $\langle A \rangle$
Projected database with Prefix $\langle A \rangle$:

1. $\langle A, A, C, C \rangle$
2. $\emptyset$
3. $\langle D, C, A, A, B \rangle$
4. $\langle C, B, C, A, A \rangle$

Frequent items ($s = 3$): A: 3, B: 2, C: 3

- Frequent length-2 seq: $\langle A, A \rangle, \langle A, C \rangle$

Projected database with Prefix $\langle A, A \rangle$:

1. $\langle A, C, C \rangle$
2. $\emptyset$
3. $\langle A, B \rangle$
4. $\langle A \rangle$

Frequent items ($s = 3$): A: 3, B: 1, C: 1

- Frequent length-3 seq: $\langle A, A, A \rangle$
Projected database w/ Prefix \(\langle A, A, A \rangle\):

1. \(\langle C, C \rangle\)
2. \(\emptyset\)
3. \(\langle B \rangle\)
4. \(\emptyset\)
- Projected database w/ Prefix $\langle A, A, A \rangle$:
  1. $\langle C, C \rangle$
  2. $\emptyset$
  3. $\langle B \rangle$
  4. $\emptyset$

- Frequent items ($s = 3$): B: 1, C: 1
  - No Frequent length-4 seq with prefix $\langle A, A, A \rangle$
- Projected database w/ Prefix \( \langle A, A, A \rangle \):
  - 1 \( \langle C, C \rangle \)
  - 2 \( \emptyset \)
  - 3 \( \langle B \rangle \)
  - 4 \( \emptyset \)

- Frequent items \((s = 3)\): B: 1, C: 1
  - No Frequent length-4 seq with prefix \( \langle A, A, A \rangle \)

- Repeat recursively for Projected databases with Prefix \( \langle A, C \rangle \)
- Repeat recursively for Projected databases with Prefix \( \langle B \rangle \)
- Repeat recursively for Projected databases with Prefix \( \langle C \rangle \)
## PrefixSpan vs. Apriori Algorithm

<table>
<thead>
<tr>
<th>PrefixSpan</th>
<th>Apriori</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate 1-item only, then combine with prefix</td>
<td>Generates candidate sequences</td>
</tr>
<tr>
<td>Scan projected database</td>
<td>Scan whole database per candidate</td>
</tr>
<tr>
<td>Depth-first search</td>
<td>Breadth-first search</td>
</tr>
</tbody>
</table>

Main cost of PrefixSpan is construction of projected database. Can be implemented by pointers.
Summary

- Sequence Mining problem:
  - Customer 1: ⟨{camera, US B}, {printer}⟩
  - Customer 2: ⟨{camera}, {printer}, {ink}⟩
  - Customers who bought camera are likely to buy printer later
Summary

- Sequence Mining problem:
  - Customer 1: $\langle \{\text{camera}, \text{USB}\}, \{\text{printer}\} \rangle$
  - Customer 2: $\langle \{\text{camera}\}, \{\text{printer}\}, \{\text{ink}\} \rangle$
  - Customers who bought camera are likely to buy printer later

- Apriori Algorithm: works ok but costly
Summary

- Sequence Mining problem:
  - Customer 1: ⟨{camera, US B}, {printer}⟩
  - Customer 2: ⟨{camera}, {printer}, {ink}⟩
  - Customers who bought camera are likely to buy printer later

- Apriori Algorithm: works ok but costly

- PrefixSpan: Divide & Conquer
  - Partition data by prefix.
  - Mine frequent item on smaller database then combine with prefix
Summary

- Sequence Mining problem:
  - Customer 1: \langle \{camera, US B\}, \{printer\} \rangle
  - Customer 2: \langle \{camera\}, \{printer\}, \{ink\} \rangle
  - Customers who bought camera are likely to buy printer later

- Apriori Algorithm: works ok but costly
- PrefixSpan: Divide & Conquer
  - Partition data by prefix.
  - Mine frequent item on smaller database then combine with prefix

- Both still exploit Monotonicity
Next Week

- Graph Mining
- Homework posted online