Introduction to
Algorithmic Data Analysis

Esther Galbrun
Autumn 2022
Part I

Frequent Itemset Mining
Problem
Discover items that often co-occur in a dataset

Classical setting: *Shopping basket data*

- Each product of the supermarket is an item
- Record customer transactions as sets of items
- Identify products that are often bought together
  
  **Frequent itemset** \{butter, bread, ham, pickles\}

- Extract rules that capture typical buying behaviour
  
  **Association rules** \{bread, ham\} \implies \{butter, pickles\}

- Insights for marketing and shelf placement
Frequent Itemset Mining

Discover items that often co-occur in a dataset

**Shopping basket data**  Customer transactions
  Identify products often bought together

**Text mining**  Bag of word model
  Identify co-occurring terms and keywords

**More complex data types**  (spatio-)temporal data, graph data

**Other analysis tasks**  Building block for clustering, classification, outlier detection
A pizzeria offers to compose your pizza by freely choosing ingredients among *ham, jalapeno, mozzarella, olives and tuna*.

To put together a menu, the pizzaiolo would like to know what are favorite combinations.
The database $\mathcal{T}$ is a collection of sets, called transactions, from a universe $U$ of items

$$\mathcal{T} = \{T_1, T_2, \ldots, T_n\}, \text{ where } T_k \subseteq U, \forall k \in [1, n]$$

The total number of items is $m = |U|$

If we fix an order over $U$, each transaction can be represented as a binary vector of size $m$

Then, the database can be represented as a binary matrix with $n$ rows and $m$ columns

Each transaction has a unique identifier, its $tid$
The universe of items is the set of five ingredients
\{\text{ham, jalapeno, mozzarella, olives, tuna}\}
For short, \(U = \{h, j, m, o, t\}\)

Each pizza constitutes a transaction, represented by the corresponding set of ingredients
For instance, a ham and mozzarella pizza is represented as \(T = \{h, m\}\), also simply denoted \(hm\)

Ordering the items alphabetically according to corresponding ingredient names, this pizza is represented by the binary vector \(\langle 1, 0, 1, 0, 0 \rangle\), also simply written 10100
The database then records all pizzas sold

<table>
<thead>
<tr>
<th>tids</th>
<th>pizzas</th>
<th>sets</th>
<th>matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>ham mozzarella olives</td>
<td>{h, m, o}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td>2)</td>
<td>mozzarella</td>
<td>{m}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td>3)</td>
<td>jalapeno mozzarella</td>
<td>{j, m}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td>4)</td>
<td>ham jalapeno mozzarella olives</td>
<td>{h, j, m, o}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td>5)</td>
<td>ham jalapeno mozzarella olives</td>
<td>{h, j, m, o}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td>6)</td>
<td>ham</td>
<td>{h}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td>7)</td>
<td>ham jalapeno mozzarella tuna</td>
<td>{h, j, m, t}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td>8)</td>
<td>mozzarella</td>
<td>{m}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td>9)</td>
<td>olives</td>
<td>{o}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td>10)</td>
<td>ham jalapeno mozzarella olives tuna</td>
<td>{h, j, m, o, t}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td>11)</td>
<td>ham mozzarella tuna</td>
<td>{h, m, t}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td>12)</td>
<td>ham mozzarella</td>
<td>{h, m}</td>
<td><img src="image" alt="matrix" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An itemset $I$ is a set of items, i.e. $I \subseteq U$

A $k$-itemset is an itemset that contains exactly $k$ items, i.e. such that $|I| = k$

The support set of an itemset $I$ in $\mathcal{T}$ is the set of transactions from $\mathcal{T}$ that contain $I$

$$\text{supp}_\mathcal{T}(I) = \{T \in \mathcal{T}, I \subseteq T\}$$

We call $|\text{supp}_\mathcal{T}(I)|$ the absolute support of $I$ in $\mathcal{T}$ and $|\text{supp}_\mathcal{T}(I)| / |\mathcal{T}|$ its fractional support.

We denote $\text{supp} \%_\mathcal{T}(I)$ the fractional support given as a percentage, i.e.

$$\text{supp} \%_\mathcal{T}(I) = 100 \cdot \frac{|\text{supp}_\mathcal{T}(I)|}{|\mathcal{T}|}$$
An itemset $I$ is a set of items, i.e. $I \subseteq U$

A $k$-itemset is an itemset that contains exactly $k$ items, i.e. such that $|I| = k$

The support set of an itemset $I$ in $\mathcal{T}$ is the set of transactions from $\mathcal{T}$ that contain $I$

$$\text{supp}_\mathcal{T}(I) = \{T \in \mathcal{T}, I \subseteq T\}$$

We call $|\text{supp}_\mathcal{T}(I)|$ the absolute support of $I$ in $\mathcal{T}$ and $|\text{supp}_\mathcal{T}(I)| / |\mathcal{T}|$ its fractional support

There are variations in the use of support terminology

The database is often left out from the notation, as it is clear from the context
Pizzeria example

In this database, for itemset $I = \{t\}$

$\text{supp}(l) = \{7, 10, 11\}$

$|\text{supp}(l)| = 3$

$\text{supp} \% (l) = 25$

<table>
<thead>
<tr>
<th>tid</th>
<th>set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>${h, m, o}$</td>
</tr>
<tr>
<td>2)</td>
<td>${m}$</td>
</tr>
<tr>
<td>3)</td>
<td>${j, m}$</td>
</tr>
<tr>
<td>4)</td>
<td>${h, j, m, o}$</td>
</tr>
<tr>
<td>5)</td>
<td>${h, j, m, o}$</td>
</tr>
<tr>
<td>6)</td>
<td>${h}$</td>
</tr>
<tr>
<td>7)</td>
<td>${h, j, m, t}$</td>
</tr>
<tr>
<td>8)</td>
<td>${m}$</td>
</tr>
<tr>
<td>9)</td>
<td>${o}$</td>
</tr>
<tr>
<td>10)</td>
<td>${h, j, m, o, t}$</td>
</tr>
<tr>
<td>11)</td>
<td>${h, m, t}$</td>
</tr>
<tr>
<td>12)</td>
<td>${h, m}$</td>
</tr>
</tbody>
</table>
In this database, for itemset \( I = \{h, m\} \)

\[
\text{supp}(I) = \{1, 4, 5, 7, 10, 11, 12\}
\]

\[
|\text{supp}(I)| = 7
\]

\[
\text{supp \%}(I) = 58.33
\]
Problem definition

Frequent Itemset Mining

Given a set of transactions $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$, where each transaction $T_i$ is a subset of items from $U$, and a minimum support threshold $\sigma$, determine all itemsets $I$ that occur as a subset of at least $\sigma$ transactions in $\mathcal{T}$. 
**Pizzeria example**

Enumerate all distinct pizzas

<table>
<thead>
<tr>
<th></th>
<th>count</th>
<th>tids</th>
<th></th>
<th>count</th>
<th>tids</th>
<th></th>
<th>count</th>
<th>tids</th>
</tr>
</thead>
<tbody>
<tr>
<td>hmo</td>
<td>65</td>
<td>{1...}</td>
<td>hj</td>
<td>16</td>
<td>{...}</td>
<td>hjmo</td>
<td>47</td>
<td>{4,5...}</td>
</tr>
<tr>
<td>m</td>
<td>74</td>
<td>{2,8...}</td>
<td>hjm</td>
<td>42</td>
<td>{...}</td>
<td>jmo</td>
<td>7</td>
<td>{...}</td>
</tr>
<tr>
<td>jm</td>
<td>28</td>
<td>{3...}</td>
<td>hjo</td>
<td>8</td>
<td>{...}</td>
<td>jmot</td>
<td>9</td>
<td>{...}</td>
</tr>
<tr>
<td>hjmo</td>
<td>47</td>
<td>{4,5...}</td>
<td>hjot</td>
<td>0</td>
<td>{}</td>
<td>jmt</td>
<td>3</td>
<td>{...}</td>
</tr>
<tr>
<td>h</td>
<td>74</td>
<td>{6...}</td>
<td>hjt</td>
<td>4</td>
<td>{...}</td>
<td>jo</td>
<td>16</td>
<td>{...}</td>
</tr>
<tr>
<td>hjmt</td>
<td>30</td>
<td>{7...}</td>
<td>hmot</td>
<td>108</td>
<td>{...}</td>
<td>jot</td>
<td>3</td>
<td>{...}</td>
</tr>
<tr>
<td>o</td>
<td>178</td>
<td>{9...}</td>
<td>ho</td>
<td>43</td>
<td>{...}</td>
<td>jt</td>
<td>14</td>
<td>{...}</td>
</tr>
<tr>
<td>hjmot</td>
<td>93</td>
<td>{10...}</td>
<td>hot</td>
<td>28</td>
<td>{...}</td>
<td>mo</td>
<td>20</td>
<td>{...}</td>
</tr>
<tr>
<td>hmt</td>
<td>49</td>
<td>{11...}</td>
<td>ht</td>
<td>7</td>
<td>{...}</td>
<td>mot</td>
<td>27</td>
<td>{...}</td>
</tr>
<tr>
<td>hm</td>
<td>96</td>
<td>{12...}</td>
<td>j</td>
<td>29</td>
<td>{...}</td>
<td>mt</td>
<td>13</td>
<td>{...}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>t</td>
<td>8</td>
<td>{...}</td>
<td>ot</td>
<td>17</td>
<td>{...}</td>
</tr>
</tbody>
</table>
Pizzeria example: Enumerating all distinct pizzas

\[
\begin{align*}
\emptyset \\
\{\} \\
\{h\} \\
\{j\} \\
\{m\} \\
\{o\} \\
\{t\} \\
\{hj\} \\
\{hm\} \\
\{ho\} \\
\{ht\} \\
\{jm\} \\
\{jo\} \\
\{jt\} \\
\{mo\} \\
\{mt\} \\
\{ot\} \\
\{hjm\} \\
\{hjo\} \\
\{hjt\} \\
\{hmo\} \\
\{hmt\} \\
\{hot\} \\
\{jmo\} \\
\{jmt\} \\
\{jot\} \\
\{jmot\} \\
\end{align*}
\]
Pizzeria example: Aggregating supports

\[
\begin{align*}
\emptyset & \quad 0 \\
\{h\} & \quad 74 \\
\{j\} & \quad 29 \\
\{m\} & \quad 74 \\
\{o\} & \quad 178 \\
\{t\} & \quad 8 \\
\{hj\} & \quad 16 \\
\{hm\} & \quad 96 \\
\{ho\} & \quad 43 \\
\{ht\} & \quad 7 \\
\{jm\} & \quad 28 \\
\{jo\} & \quad 16 \\
\{jt\} & \quad 14 \\
\{mo\} & \quad 20 \\
\{mt\} & \quad 13 \\
\{ot\} & \quad 17 \\
\{hjm\} & \quad 42 \\
\{hjo\} & \quad 8 \\
\{hjt\} & \quad 4 \\
\{hmo\} & \quad 65 \\
\{hmt\} & \quad 49 \\
\{hot\} & \quad 28 \\
\{jmo\} & \quad 7 \\
\{jmt\} & \quad 3 \\
\{jot\} & \quad 3 \\
\{mot\} & \quad 27 \\
\{hjmo\} & \quad 140 \\
\{hjmt\} & \quad 0 \\
\{hjot\} & \quad 108 \\
\{hmot\} & \quad 9 \\
\{jmot\} & \quad \{\ldots\}
\end{align*}
\]
Pizzeria example: Aggregating supports

\[
\begin{align*}
\emptyset &\quad 259 \quad \{3, 4, 5, 7, 10 \ldots\} \\
\{6\ldots\} &\quad \{2, 8\ldots\} & \{9\ldots\} &\quad \{12\ldots\} & \{10\ldots\} \quad \{11\ldots\} & \{13\ldots\} & \{14\ldots\} & \{17\ldots\} \\
16 &\quad 96 &\quad 43 &\quad 7 &\quad 16 &\quad 14 &\quad 20 &\quad 13 &\quad 17 \\
\{\ldots\} &\quad \{\ldots\} &\quad \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} \\
42 &\quad 8 &\quad 4 &\quad 65 &\quad 49 &\quad 28 &\quad 7 &\quad 3 &\quad 3 \\
\{\ldots\} &\quad \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} \\
47 &\quad 30 &\quad 0 &\quad 108 &\quad 9 &\quad 74 &\quad 29 &\quad 74 &\quad 178 &\quad 8 \\
\{4, 5\ldots\} &\quad \{7\ldots\} &\quad \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} & \{\ldots\} \\
93 &\quad \{10\ldots\} \\
\end{align*}
\]
Support properties

Monotonicity of support
The support of every subset $J$ of $I$ is at least equal to that of the support of itemset $I$

$$\forall J \subseteq I, \quad \text{supp}(I) \subseteq \text{supp}(J)$$

and hence

$$|\text{supp}(I)| \leq |\text{supp}(J)|$$

Downward closure property
Every subset of a frequent itemset is also frequent
Pizzeria example: Lattice of ingredient combinations
Pizzeria example: Frequent ingredient combinations

For minimum support

\[ \sigma = 145 = 0.125 \cdot |\mathcal{T}| \]
Pizzeria example: Frequent ingredient combinations

For minimum support

\[ \sigma = 289 = 0.250 \cdot |T| \]
Maximal and closed itemsets

A frequent itemset $I$ is **maximal** at a given minimum support level $\sigma$, if it is frequent, and no superset of it is frequent.

An itemset $I$ is **closed**, if none of its supersets have exactly the same support count as $I$.

**Condensed representations**
Knowledge of maximal frequent itemsets allows to reconstruct the set of frequent itemsets, but not their supports.
Knowledge of closed frequent itemsets allows to also recompute the supports.
Pizzeria example: Frequent and maximal itemsets

for minimum support $\sigma = 145 = 0.125 \cdot |T|$
Pizzeria example: Frequent and maximal itemsets

for minimum support

\[ \sigma = 289 = 0.25 \cdot |T| \]
Algorithms
Support counting is expensive

Explore the space of itemsets by increasing lengths, i.e. level-wise enumeration

Avoid generating itemsets twice by using a canonical order

Exploit the downward closure property to prune itemsets
Level-wise enumeration: *Apriori* algorithm

\[
\begin{align*}
k & \leftarrow 1 \\
\mathcal{F}_k & \leftarrow \{ \text{all frequent singleton itemsets} \} \\
\textbf{while } \mathcal{F}_k \neq \emptyset \textbf{ do } \\
& \quad \text{Generate } \mathcal{C}_{k+1} \text{ by extending itemsets from } \mathcal{F}_k \\
& \quad \text{Prune itemsets that violate downward closure} \\
& \quad \mathcal{F}_{k+1} \leftarrow \{ S \in \mathcal{C}_{k+1}, \text{supp}_D(S) \geq \theta \} \\
& \quad k \leftarrow k + 1 \\
\textbf{return } \bigcup_i \mathcal{F}_i
\end{align*}
\]
Level-wise enumeration: Apriori algorithm

Candidate generation

\[ k \leftarrow 1 \]
\[ F_k \leftarrow \{ \text{all frequent singleton itemsets} \} \]
while \( F_k \neq \emptyset \) do
  \[ \text{Generate } C_{k+1} \text{ by extending itemsets from } F_k \]
  \[ \text{Prune itemsets that violate downward closure} \]
  \[ F_{k+1} \leftarrow \{ S \in C_{k+1}, \text{supp}_D(S) \geq \theta \} \]
  \[ k \leftarrow k + 1 \]
return \( \bigcup_i F_i \)
Candidate pruning

\begin{align*}
k & \leftarrow 1 \\
\mathcal{F}_k & \leftarrow \{\text{all frequent singleton itemsets}\} \\
\text{while } \mathcal{F}_k & \neq \emptyset \text{ do} \\
\quad & \text{Generate } \mathcal{C}_{k+1} \text{ by extending itemsets from } \mathcal{F}_k \\
\quad & \text{Prune itemsets that violate downward closure} \\
\quad & \mathcal{F}_{k+1} \leftarrow \{\mathcal{S} \in \mathcal{C}_{k+1}, \text{supp}_D(\mathcal{S}) \geq \theta\} \\
\quad & k \leftarrow k + 1 \\
\text{return } & \bigcup_i \mathcal{F}_i
\end{align*}
Level-wise enumeration: *Apriori* algorithm

**Support counting**

\[
k \leftarrow 1 \\
\mathcal{F}_k \leftarrow \{\text{all frequent singleton itemsets}\} \\
\textbf{while } \mathcal{F}_k \neq \emptyset \textbf{ do} \\
\quad \text{Generate } \mathcal{C}_{k+1} \text{ by extending itemsets from } \mathcal{F}_k \\
\quad \text{Prune itemsets that violate downward closure} \\
\quad \mathcal{F}_{k+1} \leftarrow \{S \in \mathcal{C}_{k+1}, \text{supp}_D(S) \geq \theta\} \\
\quad k \leftarrow k + 1 \\
\textbf{return } \bigcup_i \mathcal{F}_i
Pizzeria example: \textit{Apriori} algorithm

Enumerate singleton itemsets

\begin{itemize}
\item \texttt{\{\}}
\item \texttt{h}
\item \texttt{j}
\item \texttt{m}
\item \texttt{o}
\item \texttt{t}
\end{itemize}

for $\sigma = 289$
Pizzeria example: Apriori algorithm

Count supports

\[
\begin{align*}
\emptyset & \rightarrow 1156 \\
h & \rightarrow 710 \\
j & \rightarrow 349 \\
m & \rightarrow 711 \\
o & \rightarrow 669 \\
t & \rightarrow 413
\end{align*}
\]

for \( \sigma = 289 \)
Pizzeria example: *Apriori* algorithm

Frequent singleton itemsets

For $\sigma = 289$

\[
\begin{align*}
\emptyset & : 1156 \\
h & : 710 \\
j & : 349 \\
m & : 711 \\
o & : 669 \\
t & : 413
\end{align*}
\]
Pizzeria example: \textit{Apriori} algorithm

Generate candidates itemsets of length 2

for $\sigma = 289$
Pizzeria example: *Apriori* algorithm

Count supports

```
for \( \sigma = 289 \)
```
Pizzeria example: *Apriori* algorithm

Frequent itemsets of length up to 2

```
for \( \sigma = 289 \)
```
Pizzeria example: *Apriori* algorithm

Generate candidates itemsets of length 3

for $\sigma = 289$
Pizzeria example: \textit{Apriori} algorithm

Prune candidates

\begin{itemize}
\item $h$, $j$, $m$, $o$, $t$
\item $hj$, $hm$, $ho$, $ht$, $jm$, $jo$, $jt$, $mo$, $mt$
\item $hmo$, $hmt$, $hot$
\item $mot$
\end{itemize}

\textbf{for } $\sigma = 289$
Pizzeria example: *Apriori* algorithm

Count supports

\[
\begin{align*}
\emptyset &: 1156 \\
h &: 710 \\
j &: 349 \\
m &: 711 \\
o &: 669 \\
t &: 413 \\
hj &: 240 \\
hm &: 530 \\
ho &: 392 \\
ht &: 319 \\
mj &: 259 \\
jo &: 183 \\
jt &: 156 \\
mo &: 376 \\
mt &: 332 \\
hot &: 285 \\
hmo &: 313 \\
hmt &: 280 \\
\end{align*}
\]

for \( \sigma = 289 \)
Pizzeria example: *Apriori* algorithm

Frequent itemsets of length up to 3

for $\sigma = 289$
Pizzeria example: *Apriori* algorithm

Generate candidates itemsets of length 4

for $\sigma = 289$
Pizzeria example: *Apriori* algorithm

Prune candidates

![Diagram showing the Apriori algorithm with candidate pruning for \(\sigma = 289\)]
Pizzeria example: *Apriori* algorithm

Frequent itemsets of length up to 4

for $\sigma = 289$
Pizzeria example: Enumeration tree

Items ordered alphabetically, prefix growth

for $\sigma = 289$
Pizzeria example: Enumeration tree

Items ordered by decreasing frequency, prefix growth

for $\sigma = 289$
Pizzeria example: Enumeration tree

Items ordered by increasing frequency, prefix growth

for $\sigma = 289$
Support counting is expensive

According to the monotonicity of support

\[ \forall J \subseteq I, \quad \text{supp}(l) \subseteq \text{supp}(J) \]

Make support counting more efficient

- Prune irrelevant transactions
- Reuse support counting from previous steps

Recursively project the database down the enumeration tree
Vertical apriori algorithm

\[ k \leftarrow 1 \]
\[ \mathcal{F}_k \leftarrow \{ \text{all frequent singleton itemsets} \} \]
Generate \( tid \) list for each frequent singleton itemsets

while \( \mathcal{F}_k \neq \emptyset \) do

\[ \text{Generate } \mathcal{C}_{k+1} \text{ by joining pairs of itemsets from } \mathcal{F}_k \]
Prune itemsets that violate downward closure

\[ \text{Generate } tid \text{ list for each candidate by intersecting } \]
tid lists of associated pair of \( k \)-itemsets

\[ \mathcal{F}_{k+1} \leftarrow \{ S \in \mathcal{C}_{k+1}, \operatorname{supp}_D(S) \geq \theta \} \]
\[ k \leftarrow k + 1 \]

return \( \bigcup_i \mathcal{F}_i \)
Vertical apriori algorithm

Vertical database representation

\[
k \leftarrow 1 \\
\mathcal{F}_k \leftarrow \{ \text{all frequent singleton itemsets} \} \\
\text{Generate } \textit{tid} \text{ list for each frequent singleton itemsets} \\
\textbf{while } \mathcal{F}_k \neq \emptyset \textbf{ do} \\
\hspace{1em} \text{Generate } \mathcal{C}_{k+1} \text{ by joining pairs of itemsets from } \mathcal{F}_k \\
\hspace{1em} \text{Prune itemsets that violate downward closure} \\
\hspace{1em} \text{Generate } \textit{tid} \text{ list for each candidate by intersecting } \\
\hspace{1em} \text{tid lists of associated pair of } k\text{-itemsets} \\
\hspace{1em} \mathcal{F}_{k+1} \leftarrow \{ S \in \mathcal{C}_{k+1}, \text{supp}_D(S) \geq \theta \} \\
\hspace{1em} k \leftarrow k + 1 \\
\textbf{return } \bigcup_i \mathcal{F}_i
\]
Space–time trade-off

Tid lists

- Allow to compute supports faster
- Require memory space for storage

Use dedicated data structures that support efficient counting
The **FP-tree** is a compact representation of the database

- Extract conditional projected database for a given suffix
- Update counts efficiently

**FP-growth** is a recursive suffix-based pattern growth algorithm
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 1
Transaction mho
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 3
Transaction mj
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 4
Transaction mhoj
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 5
Transaction mhoj
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 6
Transaction h
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 7
Transaction mhtj
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 8
Transaction m
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 9
Transaction 0
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 10
Transaction mhotj
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 11
Transaction mht
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step 12
Transaction mh
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)

Step# 100
Transaction mho
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)
Construction of the FP-tree

Inserting transactions (items sorted by decreasing frequency)
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix $m$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix $h$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix $mh$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix  o
Recursive pattern growth

for $\sigma = 289$

Suffix $mo$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth for $\sigma = 289$

Suffix $ho$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix mho
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix t
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix $mt$
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$
Suffix ht
Suffix-based pattern growth with the FP-tree

Recursive pattern growth

for $\sigma = 289$

Suffix $j$
Items ordered by decreasing frequency, suffix growth
Association rules
Association rules

Frequent itemsets can be used to generate association rules

Classical setting: *Shopping basket data*

- Identify products that are often bought together
  
  **Frequent itemset** \{butter, bread, ham, pickles\}

- Extract rules that capture typical buying behaviour
  
  **Association rules** \{bread, ham\} ⇒ \{butter, pickles\}

- Insights for marketing and shelf placement
Association rules

Frequent itemsets can be used to generate association rules. Consider two itemsets $X$ and $Y$ such that

$$X \subset U, \quad \emptyset \neq Y \subseteq U, \quad X \cap Y = \emptyset$$

The confidence of the association rule $X \Rightarrow Y$ is the conditional probability that a transaction contains $X \cup Y$ given that it contains $X$

$$\text{conf}(X \Rightarrow Y) = \frac{|\text{supp}(X \cup Y)|}{|\text{supp}(X)|}$$

$X$ and $Y$ are called the antecedent and consequent of the rule, respectively.
$X \Rightarrow Y$ is an association rule at minimum support $\sigma$ and minimum confidence $\gamma$ if

$$\text{supp}(X \cup Y) \geq \sigma \quad \text{and} \quad \text{conf}(X \Rightarrow Y) \geq \gamma$$
Mining association rules

1. Mine all the frequent itemsets for minimum support $\sigma$
2. Split the frequent itemsets into association rules of minimum confidence $\gamma$

Monotonicity of confidence
Let $X_a$, $X_b$ and $I$ be itemsets such that $X_a \subset X_b \subset I$, then

$$\text{conf}(X_b \Rightarrow I \setminus X_b) \geq \text{conf}(X_a \Rightarrow I \setminus X_a)$$