LINGI2364: Mining Patterns in Data

Exercise session 2: Depth-First Algorithms for Itemset Mining and Sequential Pattern Mining

Gilles Peiffer

25 February 2020

2 Solutions

1. (a) For a minimum support threshold of 2, the projected databases are obtained as follows (by convention, transactions which do not appear are empty):

$$\mathcal{D}_{|\emptyset}(t) = \begin{cases} \mathcal{D}(t), & \text{if } t \in \{0, \ldots, 9\} \setminus \{7\}, \\ \mathcal{D}(7) \setminus \{G\}, & \text{otherwise.} \end{cases}$$

$$\begin{aligned}
\mathcal{D}_{|\{B\}}(2) &= \{D\}, \\
\mathcal{D}_{|\{B\}}(3) &= \{C, D\}, \\
\mathcal{D}_{|\{B\}}(4) &= \{C\}, \\
\mathcal{D}_{|\{B\}}(5) &= \{D\}, \\
\mathcal{D}_{|\{B\}}(6) &= \{D, E\}, \\
\mathcal{D}_{|\{B\}}(7) &= \{C, D, E\}, \\
\mathcal{D}_{|\{B\}}(9) &= \{D\}.
\end{aligned}$$

$$\begin{aligned}
\mathcal{D}_{|\{B,C\}}(3) &= \{D\}, \\
\mathcal{D}_{|\{B,C\}}(7) &= \{D\}.
\end{aligned}$$

(b)

$$\begin{aligned}
\mathrm{cover}_{\mathcal{D}_{|\emptyset}}(\{A\}) &= \{0, 1, 5, 9\}, \\
\mathrm{cover}_{\mathcal{D}_{|\emptyset}}(\{B\}) &= \{2, 3, 4, 5, 6, 7, 9\}, \\
\mathrm{cover}_{\mathcal{D}_{|\emptyset}}(\{C\}) &= \{1, 3, 4, 7, 8\}, \\
\mathrm{cover}_{\mathcal{D}_{|\emptyset}}(\{D\}) &= \{0, 1, 2, 3, 5, 6, 7, 8, 9\}, \\
\mathrm{cover}_{\mathcal{D}_{|\emptyset}}(\{E\}) &= \{1, 6, 7\}, \\
\mathrm{cover}_{\mathcal{D}_{|\emptyset}}(\{F\}) &= \{0, 8\}.
\end{aligned}$$

$$\begin{aligned}
\mathrm{cover}_{\mathcal{D}_{|\{B\}}}(\{C\}) &= \{3, 4, 7\}, \\
\mathrm{cover}_{\mathcal{D}_{|\{B\}}}(\{D\}) &= \{2, 3, 5, 6, 7, 9\}, \\
\mathrm{cover}_{\mathcal{D}_{|\{B\}}}(\{E\}) &= \{6, 7\}.
\end{aligned}$$

$$\mathrm{cover}_{\mathcal{D}_{|\{B,C\}}}(\{D\}) = \{3, 7\}.$$

(c) At node $\{A\}$, the projected database is

$$\begin{aligned}
\mathrm{cover}_{\mathcal{D}_{|\{A\}}}(\{B\}) &= \{5, 9\}, \\
\mathrm{cover}_{\mathcal{D}_{|\{A\}}}(\{D\}) &= \{0, 1, 5, 9\}.
\end{aligned}$$

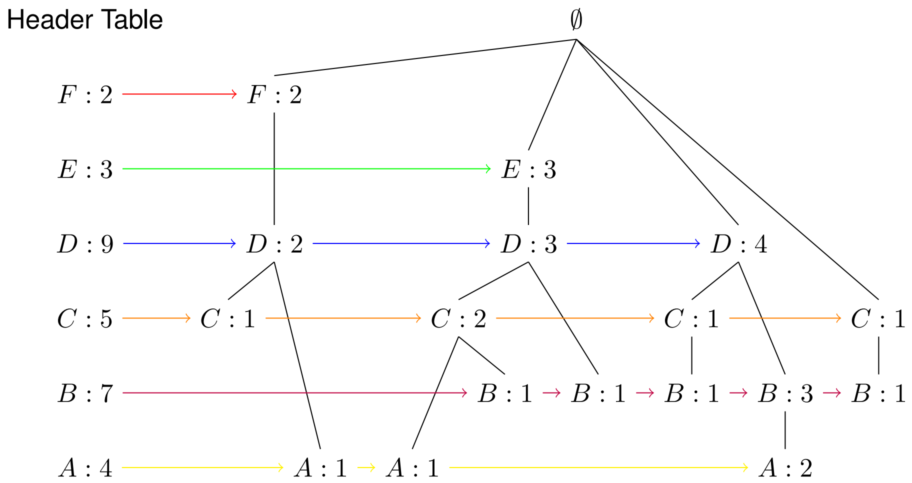
The DFS algorithm next explores node $\{A, B\}$, which has the following projected database:

$$\mathrm{cover}_{\mathcal{D}_{|\{A,B\}}}(\{D\}) = \{5, 9\}.$$

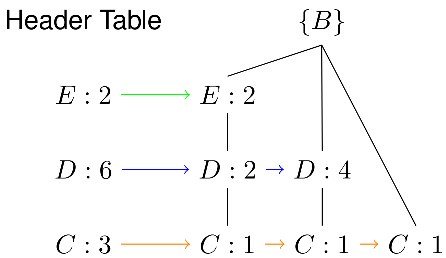
However, $\mathrm{cover}_{\mathcal{D}_{|\{A,B,D\}}}(I) = \emptyset$, for any item I.
The search thus backtracks, and explores node $\{A, D\}$. $\mathrm{cover}_{\mathcal{D}_{|\{A,D\}}}(I) = \emptyset$, for any item I, hence the search backtracks, and then once again, back to the root node. This concludes the DFS algorithm.

(d) The inverted tid-list for $\{A\}$ is $\{2, 3, 4, 6, 7, 8\}$. The inverted tid-list for $\{B\}$ is $\{0, 1, 8\}$. The inverted tid-list for $\{A, B\}$ can then be computed by taking the union of the previously identified inverted tid-lists for $\{A\}$ and $\{B\}$: $\{0, 1, 2, 3, 4, 6, 7, 8\}$.
Inverted tid-lists have the advantage that computing a set union is easier than computing an intersection. However, computing the cover is harder, and has higher memory requirements, since very quickly, inverted tid-lists are close to $\mathcal{T}$.

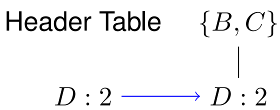
2. (a) The FP-tree for $\emptyset$ is the following:



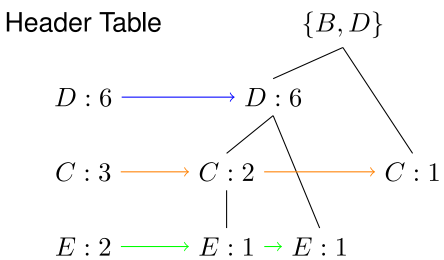
The FP-tree for $\{B\}$ is the following:



The FP-tree for $\{B, C\}$ is the following:



(b) By sorting on the frequencies instead of alphabetically ($D \to C \to E$), one finds the following FP-tree for $\{B\}$:



(c) The first node to be explored is $\{B, D\}$. However, its FP-tree is empty. The search backtracks, and visits $\{B, C\}$, which has FP-tree

$$\begin{array}{cc} \text{Header Table} & \{B,C\} \\ & | \\ D:2 \longrightarrow & D:2 \end{array}$$

The search then continues down its path, towards $\{B, C, D\}$, which has an empty FP-tree. The search backtracks twice, then visits $\{B, E\}$, with FP-tree

$$\begin{array}{cc} \text{Header Table} & \{B,E\} \\ & | \\ D:2 \longrightarrow & D:2 \end{array}$$

The next child of this node, $\{B, D, E\}$, has an empty FP-tree. The search backtracks all the way up to the root node, and the algorithm terminates.

3. (a)

$$\begin{aligned} \text{cover}(\langle\{A,B\},\{C\}\rangle) &= \{0,1\}, \\ \text{cover}(\langle\{A,B\},\{D\}\rangle) &= \{0,1\}, \\ \text{cover}(\langle\{A\},\{B\}\rangle) &= \{2,3\}, \\ \text{cover}(\langle\{A\},\{D\}\rangle) &= \{0,1,2\}. \end{aligned}$$

(b) The frequent sequences are:

- $\langle\{A\}\rangle, \langle\{B\}\rangle, \langle\{C\}\rangle, \langle\{D\}\rangle$,
- $\langle\{A,B\}\rangle, \langle\{A,C\}\rangle$,
- $\langle\{A\},\{A\}\rangle$, $\langle\{A\},\{B\}\rangle$, $\langle\{A\},\{C\}\rangle$, $\langle\{A\},\{D\}\rangle$, $\langle\{B\},\{C\}\rangle$, $\langle\{B\},\{D\}\rangle$, $\langle\{C\},\{A\}\rangle$, $\langle\{C\},\{B\}\rangle$, $\langle\{C\},\{D\}\rangle$,
- $\langle\{A,B\},\{C\}\rangle, \langle\{A,B\},\{D\}\rangle$,
- $\langle\{A\},\{A,C\}\rangle$.

(c) Let $F = \langle I_1, \ldots, I_n\rangle \in (2^{\mathcal{I}})^*$ be a frequent sequence. By anti-monotonicity, the sequence $S = \langle J_1, \ldots, J_m\rangle \in (2^{\mathcal{I}})^*$ is also frequent, if there exist integers $1 \le i_1 < \cdots < i_m \le n$ such that for all $j \in \{1, \ldots, m\} : J_j \subseteq I_{i_j}$.
Conversely, let $I = \langle I_1, \ldots, I_n\rangle \in (2^{\mathcal{I}})^*$ be an infrequent sequence. By anti-monotonicity, the sequence $S = \langle J_1, \ldots, J_m\rangle \in (2^{\mathcal{I}})^*$ is then also infrequent, if there exist integers $1 \le j_1 < \cdots < j_n \le m$ such that for all $i \in \{1, \ldots, n\} : I_i \subseteq J_{j_i}$.
This formalizes the intuition of anti-monotonicity: "If a sequence is frequent, all sequences which are contained in it are also frequent; if a sequence is infrequent, all sequences in which it is contained are also infrequent."

(d) No, there is no anti-monotonicity property.

Proof. By contradiction. Take the single transaction database $\langle\{A\},\{B\},\{B\}\rangle$. For this database, $\text{support}(\langle\{A\}\rangle) = 1 < \text{support}(\langle\{A\}\{B\}\rangle) = 2$, despite the fact that $\langle\{A\}\rangle \subseteq \langle\{A\}\{B\}\rangle$. $\square$

4. (a) Let the relation be written as $\preceq$. $\preceq$ is a partial order if and only if the three following properties are satisfied:

- Reflexivity: $a \preceq a$.
- Antisymmetry: if $a \preceq b$ and $b \preceq a$, then $a = b$.
- Transitivity: if $a \preceq b$ and $b \preceq c$, then $a \preceq c$.

(b) We assume all elements are different. This relation is then not a partial order.

Proof. By contradiction. Since $A \succeq C$ and $C \succeq D$, by transitivity one would obtain $A \succeq D$, however this is not the case. $\square$

(c) We do not prove it formally, but *match* is a partial order, since it (intuitively) verifies all three required properties.

(d) This constraint is a gap constraint. It expresses the fact that the number of itemsets between matched itemsets in a patter must not exceed some limit $g-1$.

(e) No, it is no longer a partial order.

Proof. By contradiction. Take $X = \langle\{A\},\{D\}\rangle, Y = \langle\{A\},\{C\},\{D\}\rangle, Z = \langle\{A\},\{B\},\{C\},\{D\}\rangle$, with $g = 2$. It is easy to verify that $match(X,Y)$ and $match(Y,Z)$, however, $\neg match(X,Z)$ since the gap between $\{A\}$ and $\{D\}$ in Z is $3 > g = 2$. $\square$

(f) With the length constraint, *match* is still a partial order.

When combining the two constraints, *match* is a partial order if $g \geq \ell$, as this makes the gap constraint redundant.

5. (a)
- At level 1, we generate $\langle\{A\}\rangle$, $\langle\{B\}\rangle$, $\langle\{C\}\rangle$, $\langle\{D\}\rangle$. All are frequent.
- At level 2, we generate $\langle\{A,B\}\rangle$, $\langle\{A,C\}\rangle$, $\langle\{A,D\}\rangle$, $\langle\{A\},\{A\}\rangle$, $\langle\{A\},\{B\}\rangle$, $\langle\{A\},\{C\}\rangle$, $\langle\{A\},\{D\}\rangle$, $\langle\{B\},\{A\}\rangle$, $\langle\{B\},\{B\}\rangle$, $\langle\{B\},\{C\}\rangle$, $\langle\{B\},\{D\}\rangle$, $\langle\{C\},\{A\}\rangle$, $\langle\{C\},\{B\}\rangle$, $\langle\{C\},\{C\}\rangle$, $\langle\{C\},\{D\}\rangle$, $\langle\{D\},\{A\}\rangle$, $\langle\{D\},\{B\}\rangle$, $\langle\{D\},\{C\}\rangle$, $\langle\{D\},\{D\}\rangle$.
  The frequent sequences are $\langle\{A,B\}\rangle$, $\langle\{A,C\}\rangle$, $\langle\{A\},\{A\}\rangle$, $\langle\{A\},\{B\}\rangle$, $\langle\{A\},\{C\}\rangle$, $\langle\{A\},\{D\}\rangle$, $\langle\{B\},\{C\}\rangle$, $\langle\{B\},\{D\}\rangle$, $\langle\{C\},\{A\}\rangle$, $\langle\{C\},\{B\}\rangle$, $\langle\{C\},\{D\}\rangle$.
- At level 3, we generate $\langle\{A,B,C\}\rangle$, $\langle\{A,B\},\{A\}\rangle$, $\langle\{A,B\},\{B\}\rangle$, $\langle\{A,B\},\{C\}\rangle$, $\langle\{A,B\},\{D\}\rangle$, $\langle\{A,C\},\{A\}\rangle, \ldots$
  The frequent sequences are $\langle\{A,B\},\{C\}\rangle, \langle\{A,B\},\{D\}\rangle$, $\langle\{A\},\{A,C\}\rangle$.
- At level 4, we generate $\langle\{A,B\},\{C,D\}\rangle$, $\langle\{A,B\},\{C\},\{D\}\rangle$, $\langle\{A,B\},\{D\},\{C\}\rangle$.
  There are no more frequent sequences, and the GSP algorithm terminates.

Once this is done, we can eliminate all sequences which do not contain the item A.

(b) We denote X being added to the last itemset of a sequence as branching on X, and a new itemset being added starting with X by branching on $|X$. The trie of frequent sequences looks like this: