

Elements of Machine Learning & Data Science

Winter semester 2025/26

Lecture 5 – Frequent Itemsets

04.11.2025

Prof. Bastian Leibe

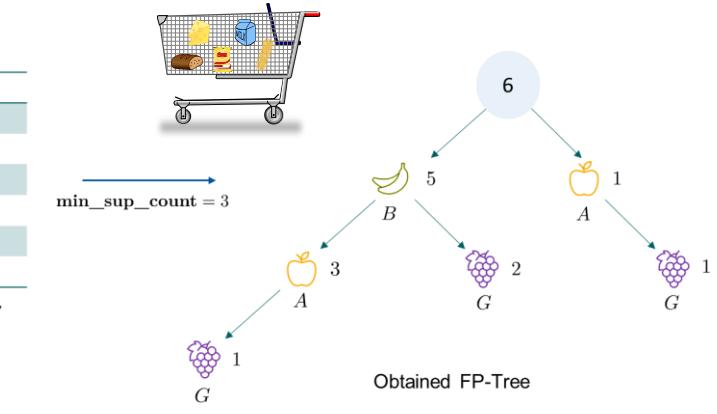
slides by Prof. Wil van der Aalst

Overview of the Lecture Topics

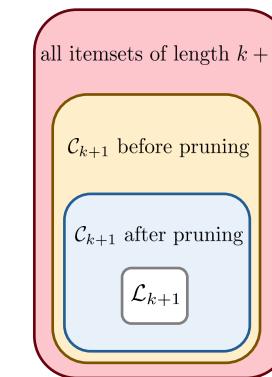
1. Introduction to Data Science
2. Decision Trees
3. Clustering
- 4. Frequent Itemsets**
5. Association Rules
6. Time Series

| TID | Bought Fruits |
|-----|-------------------------|
| 1 | {Banana, Apple} |
| 2 | {Banana, Grapes} |
| 3 | {Banana, Apple} |
| 4 | {Apple, Grapes} |
| 5 | {Banana, Grapes} |
| 6 | {Banana, Apple, Grapes} |

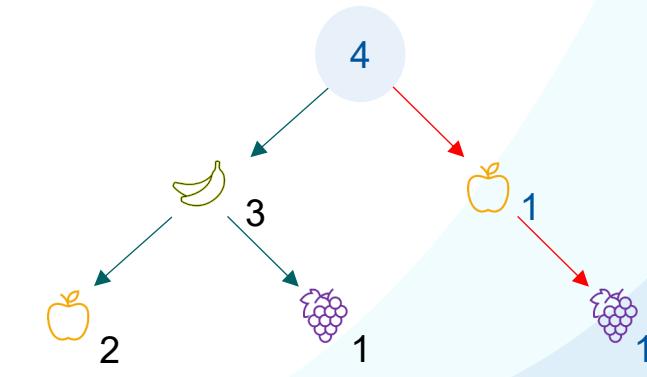
Initial transaction dataset \mathcal{X}



Frequent Itemsets



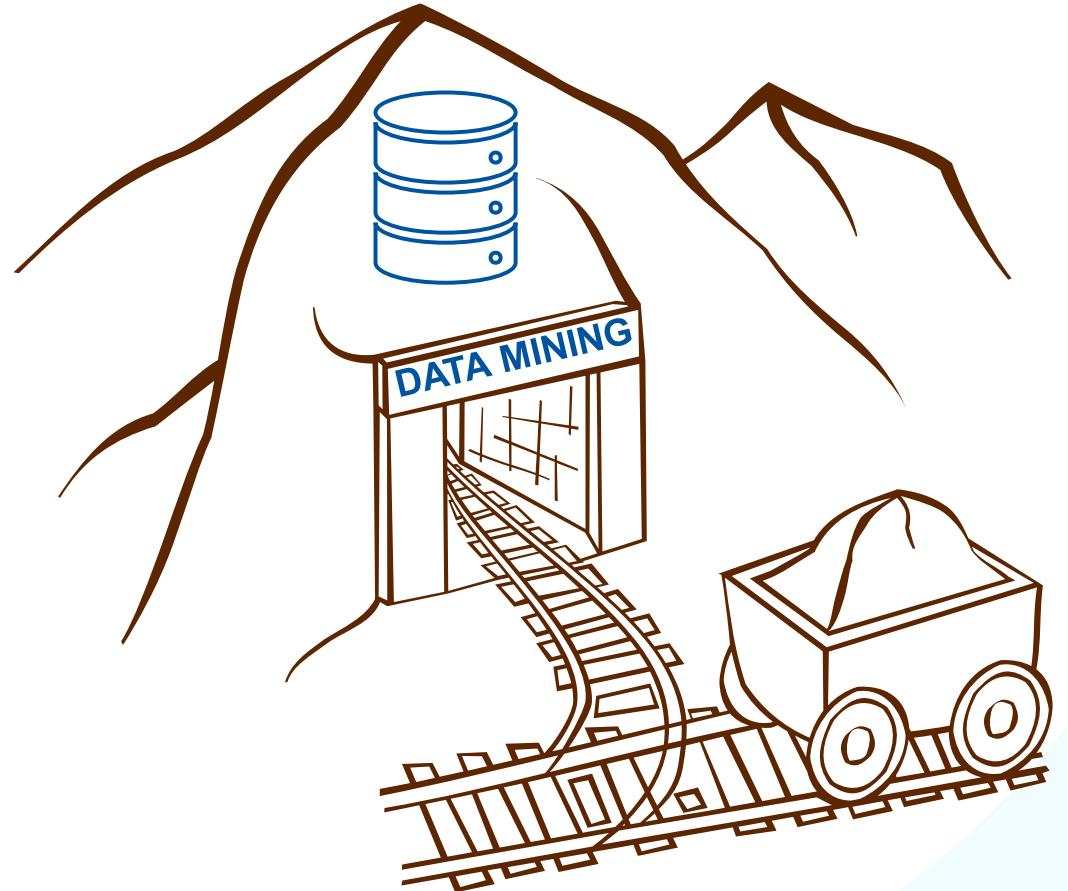
Apriori Algorithm



FP-Growth Algorithm

Frequent Itemsets

1. Introduction
2. Properties of Frequent Itemsets
3. A-Priori Algorithm
4. FP-Growth Algorithm



Pattern Mining

- Finding surprising **patterns** in the input data
- Types of patterns:
 - **Frequent itemsets**
 - Association rules
 - Sequential patterns
 - Partial orders
 - Subgraphs

Itemset Data

| ID | f_1 | f_2 | f_3 | f_4 | ... | f_D |
|-----|-------|-------|-------|-------|-----|-------|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| ... | | | | | | |

Each instance is a **transaction**

Each feature refers to an **item** (e.g., a product, disease, song, course, or error code)

Each cell describes the **presence of the item** (Boolean 0/1 or a natural number)

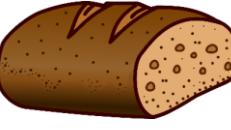
Itemset Data – Example



Itemset Data – Example

| ID |  |  |  |  | ... |  |
|-----|---|---|---|---|-----|---|
| 1 | 2 | 2 | 0 | 3 | | 2 |
| 2 | 0 | 0 | 1 | 1 | | 0 |
| 3 | 2 | 1 | 0 | 0 | | 0 |
| 4 | 0 | 1 | 0 | 0 | | 0 |
| 5 | 0 | 0 | 0 | 0 | | 2 |
| ... | ... | ... | ... | ... | ... | ... |

Itemset Data – Example

| ID |  |  |  |  | ... |  |
|-----|---|---|---|---|-----|---|
| 1 | True | True | False | True | | True |
| 2 | False | False | True | True | | False |
| 3 | True | True | False | False | | False |
| 4 | False | True | False | False | | False |
| 5 | False | False | False | False | | True |
| ... | ... | ... | ... | ... | ... | ... |

Other Itemset Data Examples

| Rows | Columns |
|----------------------|---------------------|
| EdX users | Courses taken |
| Spotify users | Songs Played |
| Netflix users | Movies Watched |
| Patients in hospital | Diseases |
| Repair bills | Components replaced |
| ... | ... |

Application of Frequent Itemsets

| ID | | | | | ... | |
|-----|-------|-------|-------|-------|-----|-------|
| 1 | True | True | False | True | | True |
| 2 | False | False | True | True | | False |
| 3 | True | True | False | True | | False |
| 4 | False | True | False | True | | False |
| 5 | False | False | False | False | | True |
| ... | ... | ... | ... | ... | ... | ... |

↑
↑
Frequent Itemsets (movies)

NETFLIX

Application of Frequent Itemsets

| ID |  |  |  |  | ... |  |
|-----|---|---|---|---|-----|---|
| 1 | True | True | False | True | | True |
| 2 | True | True | True | False | | False |
| 3 | True | True | False | True | | False |
| 4 | True | False | False | False | | False |
| 5 | False | False | False | False | | True |
| ... | ... | ... | ... | ... | | ... |

 
Frequent Itemsets (products)



Application of Frequent Itemsets

- A success story showing the potential of itemset mining: the [Tesco Clubcard](#)
- Introduced in 1995, it was the first loyalty card with [automatic data collection](#)
- Widely regarded as responsible for Tesco's supremacy in the UK
- 1bn£ of increase in sales (4%) in one year
- Today, the Clubcard program is still incredibly profitable, even though Tesco gives away about 1bn£ in rewards and discounts each year!



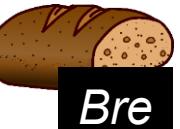
["You know more about my customers after three months than I know after 30 years."](#)

- Lord MacLaurin, chairman for Tesco,
talking to the data scientists of the Clubcard
program

Frequent Itemsets – Notation

- $\mathcal{I} = \{I_1, I_2, \dots, I_D\}$ is the set of all possible items
- $\mathcal{A} \subseteq \mathcal{I}$ is an itemset
- A transaction \mathcal{T} is a non-empty itemset
- A dataset \mathcal{X} is a collection of transactions
- Technically $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$ such that $\emptyset \notin \mathcal{X}$
(\mathbb{M} is the multiset and \mathbb{P} is the powerset operator)

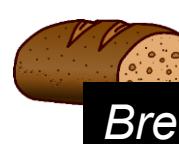
Frequent Itemsets – Example

| ID |  |  |  |  | ... |  | Pas |
|----|---|---|---|--|-----------|---|-----|
| 1 | 2 (true) | 0 (false) | 0 (false) | 3 (true) | 0 (false) | 2 (true) | |
| 2 | 0 (false) | 0 (false) | 1 (true) | 1 (true) | 0 (false) | 0 (false) | |
| 3 | 2 (true) | 1 (true) | 0 (false) | 0 (false) | 0 (false) | 0 (false) | |
| 4 | 0 (false) | 1 (true) | 0 (false) | 0 (false) | 0 (false) | 0 (false) | |

$$\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$$

- Set of all items $\mathcal{I} = \{Che, Bre, Chi, Mil, \dots, Pas\}$
- Transaction $\mathcal{T}_1 = \{Che, Mil, Pas\} \subseteq \mathcal{I}$
- Dataset with four transactions $\mathcal{X} = [\{Che, Mil, Pas\}, \{Chi, Mil\}, \{Che, Bre\}, \{Bre\}]$
- Dataset with ten transactions $\mathcal{X} = [\{Che, Mil, Pas\}^4, \{Chi, Mil\}^3, \{Che, Bre\}^2, \{Bre\}^1]$

Frequent Itemsets – Generalization To Multisets

| ID |  |  |  |  | ... |  | Pas |
|----|---|---|---|--|-----------|---|-----|
| 1 | 2 (true) | 0 (false) | 0 (false) | 3 (true) | 0 (false) | 2 (true) | |
| 2 | 0 (false) | 0 (false) | 1 (true) | 1 (true) | 0 (false) | 0 (false) | |
| 3 | 2 (true) | 1 (true) | 0 (false) | 0 (false) | 0 (false) | 0 (false) | |
| 4 | 0 (false) | 1 (true) | 0 (false) | 0 (false) | 0 (false) | 0 (false) | |

$$\mathcal{X} \in \mathbb{M}(\mathbb{M}(\mathcal{I}))$$

- Set of all items $\mathcal{I} = \{Che, Bre, Chi, Mil, \dots, Pas\}$
- Transaction $T_1 = [Che^2, Mil^3, Pas^2] \in \mathbb{M}(\mathcal{I}) = \mathcal{I} \rightarrow \mathbb{N}$
- Dataset with four transactions $\mathcal{X} = [[Che^2, Mil^3, Pas^2], [Chi, Mil], [Che^2, Bre], [Bre]]$

We will consider only itemsets that are proper sets (not multisets). However, generalization is trivial.

Frequent Itemsets – Support

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

(relative)

Fraction of transactions \mathcal{T} in dataset \mathcal{X} that cover the itemset \mathcal{A}

$$\text{support_count}(\mathcal{A}) = |[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|$$

(absolute, also called frequency or count)

Frequent Itemsets – Support

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

(relative)

Fraction of transactions \mathcal{T} in dataset \mathcal{X} that cover the itemset \mathcal{A}

$$\text{support_count}(\mathcal{A}) = |[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|$$

(absolute, also called frequency or count)

- Minimum **support threshold**:
 - *min_sup*: lower bound for $\text{support}(\mathcal{A})$
 - *min_sup_count*: lower bound for $\text{support_count}(\mathcal{A})$
- An itemset is **frequent** if its support is higher than *min_sup* (or *min_sup_count*)
- Frequent itemsets are used to find **association rules**

Support – Example

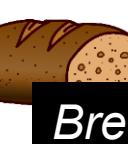
| ID | Che | Bre | Chi | Mil | ... | Pas |
|----|-----------|-----------|-----------|-----------|-----------|-----------|
| 1 | 2 (true) | 0 (false) | 0 (false) | 3 (true) | 0 (false) | 2 (true) |
| 2 | 0 (false) | 0 (false) | 1 (true) | 1 (true) | 0 (false) | 0 (false) |
| 3 | 2 (true) | 1 (true) | 0 (false) | 0 (false) | 0 (false) | 0 (false) |
| 4 | 1 (true) | 1 (true) | 0 (false) | 1 (true) | 0 (false) | 0 (false) |

$\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$

Dataset $\mathcal{X} = [\{Che, Mil, Pas\}, \{Chi, Mil\}, \{Che, Bre\}, \{Che, Bre, Mil\}]$

Support – Example

↓ ↓

| ID |  |  |  |  | ... |  | Pas |
|----|---|---|---|---|-----------|---|-----|
| 1 | 2 (true) | 0 (false) | 0 (false) | 3 (true) | 0 (false) | 2 (true) | |
| 2 | 0 (false) | 0 (false) | 1 (true) | 1 (true) | 0 (false) | 0 (false) | |
| 3 | 2 (true) | 1 (true) | 0 (false) | 0 (false) | 0 (false) | 0 (false) | |
| 4 | 1 (true) | 1 (true) | 0 (false) | 1 (true) | 0 (false) | 0 (false) | |

$\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$

Dataset $\mathcal{X} = [\{Che, Mil, Pas\}, \{Chi, Mil\}, \{Che, Bre\}, \{Che, Bre, Mil\}]$

Itemset $\mathcal{A} = \{Che, Mil\} \subseteq \mathcal{I}$

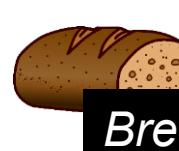
$$\text{support_count}(\mathcal{A}) = |[\mathcal{T} \in \mathcal{X} \mid \mathcal{A} \subseteq \mathcal{T}]| = |[\mathcal{T}_1, \mathcal{T}_4]| = 2$$

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} \mid \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|} = \frac{|[\mathcal{T}_1, \mathcal{T}_4]|}{4} = \frac{2}{4}$$

\mathcal{A} is **frequent** if $\text{min_sup} \leq 0.5$

Support – Example

↓

| ID |  |  |  |  | ... |  | Pas |
|----|---|---|---|--|-----------|---|-----|
| 1 | 2 (true) | 0 (false) | 0 (false) | 3 (true) | 0 (false) | 2 (true) | |
| 2 | 0 (false) | 0 (false) | 1 (true) | 1 (true) | 0 (false) | 0 (false) | |
| 3 | 2 (true) | 1 (true) | 0 (false) | 0 (false) | 0 (false) | 0 (false) | |
| 4 | 1 (true) | 1 (true) | 0 (false) | 1 (true) | 0 (false) | 0 (false) | |

$\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$

Dataset $\mathcal{X} = [\{Che, Mil, Pas\}, \{Chi, Mil\}, \{Che, Bre\}, \{Che, Bre, Mil\}]$

Itemset $\mathcal{A} = \{Che, Mil\} \subseteq \mathcal{I}$

$$\text{support_count}(\mathcal{A}) = |[\mathcal{T} \in \mathcal{X} \mid \mathcal{A} \subseteq \mathcal{T}]| = |[\mathcal{T}_1, \mathcal{T}_4]| = 2$$

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} \mid \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|} = \frac{|[\mathcal{T}_1, \mathcal{T}_4]|}{4} = \frac{2}{4}$$

\mathcal{A} is **frequent** if $\text{min_sup} \leq 0.5$

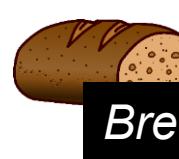
Itemset $\mathcal{B} = \{Mil\} \subseteq \mathcal{I}$

$$\text{support_count}(\mathcal{B}) = |[\mathcal{T} \in \mathcal{X} \mid \mathcal{B} \subseteq \mathcal{T}]| = |[\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_4]| = 3$$

$$\text{support}(\mathcal{B}) = \frac{|[\mathcal{T} \in \mathcal{X} \mid \mathcal{B} \subseteq \mathcal{T}]|}{|\mathcal{X}|} = \frac{|[\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_4]|}{4} = \frac{3}{4}$$

\mathcal{B} is **frequent** if $\text{min_sup} \leq 0.75$

Support – Example

| ID |  |  |  |  | ... |  | Pas |
|----|---|---|---|--|-----------|---|-----|
| 1 | 2 (true) | 0 (false) | 0 (false) | 3 (true) | 0 (false) | 2 (true) | |
| 2 | 0 (false) | 0 (false) | 1 (true) | 1 (true) | 0 (false) | 0 (false) | |
| 3 | 2 (true) | 1 (true) | 0 (false) | 0 (false) | 0 (false) | 0 (false) | |
| 4 | 1 (true) | 1 (true) | 0 (false) | 1 (true) | 0 (false) | 0 (false) | |

$\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$

Itemset $\mathcal{A} = \{Che, Mil\} \subseteq \mathcal{I}$

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|} = \frac{|[\mathcal{T}_1, \mathcal{T}_4]|}{4} = \frac{2}{4}$$

Itemset $\mathcal{B} = \{Mil\} \subseteq \mathcal{I}$

$$\text{support}(\mathcal{B}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{B} \subseteq \mathcal{T}]|}{|\mathcal{X}|} = \frac{|[\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_4]|}{4} = \frac{3}{4}$$

$$\mathcal{B} \subseteq \mathcal{A} \implies \text{support}(\mathcal{B}) \geq \text{support}(\mathcal{A})$$

General rule:
Subset of an itemset has higher or equal support than this itemset

Support – Summary

Support

- A measure of the popularity (frequency) of an itemset.
- Calculated as the fraction of transactions in a dataset that contain the itemset.

$$\text{support}(\mathcal{A}) = \frac{|[\mathcal{T} \in \mathcal{X} | \mathcal{A} \subseteq \mathcal{T}]|}{|\mathcal{X}|}$$

- Any itemset with a support below the threshold is considered to be infrequent.
- Support is also used to find association rules

Frequent Itemsets

1. Introduction
2. **Properties of Frequent Itemsets**
3. A-Priori Algorithm
4. FP-Growth Algorithm



Problem Statement

Given dataset $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$ and minimum support threshold min_sup ,
find [all frequent non-empty itemsets](#):

$$\{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup}\}$$

Naïve Approach

- Given $\mathcal{A} \subseteq \mathcal{I}$, it is possible to check whether $support(\mathcal{A}) \geq min_sup$ by testing all transactions
- If there are D unique items, then there are $2^D - 1$ candidate itemsets that can all be tested individually
- However, this can be very time consuming...



100 million songs, 626 million monthly active users

The screenshot shows a news article from The Guardian. The header includes "News", "Opinion", "Sport", "Culture", "Lifestyle", and a yellow "Eur" button. Below the header are links for "Business", "Economics", "Banking", "Money", "Markets", "Project Syndicate", "B2B", and "Retail". The main headline is "Tesco cuts range by 30% to simplify shopping" by Zoe Wood and Sarah Butler, published on Fri 30 Jan 2015 20.53 CET. It has 331 shares. A yellow box at the top right of the article area says "This article is more than 9 years old". The article text states: "By reducing number of products from 90,000, supermarket will be able to cut prices and improve availability on its shelves". A large blue speech bubble contains the text "90.000 products". Below the article is a photograph of blue shopping trolleys. At the bottom of the page, a caption reads: "Tesco's new boss Dave Lewis is pulling up to a third of products off its shelves as it calls time on policy that left shoppers baffled by a choice of up to 90,000 products on their weekly shop. Photograph: Chris Radburn/PA". To the right of the article, there is a sidebar titled "Most viewed" with several thumbnail images and titles, including "Manchester United 4-0 Everton, Chelsea 3-0 Aston Villa and more: Premier League - as it happened", "Liverpool v Manchester City: Premier League - live", "'He is one of us': US anti-vaxxers rejoice at nomination of David Weldon for CDC", "More than 600 Brazilians deported by Home Office on three secret flights", and "Middle East crisis: hundreds killed since launch of Syrian rebel offensive, says war monitor".

Let's be conservative: Assume $D = 50\,000$ products

$$2^D - 1 =$$

Good Luck
on your
course most of them
are not
irrelevant

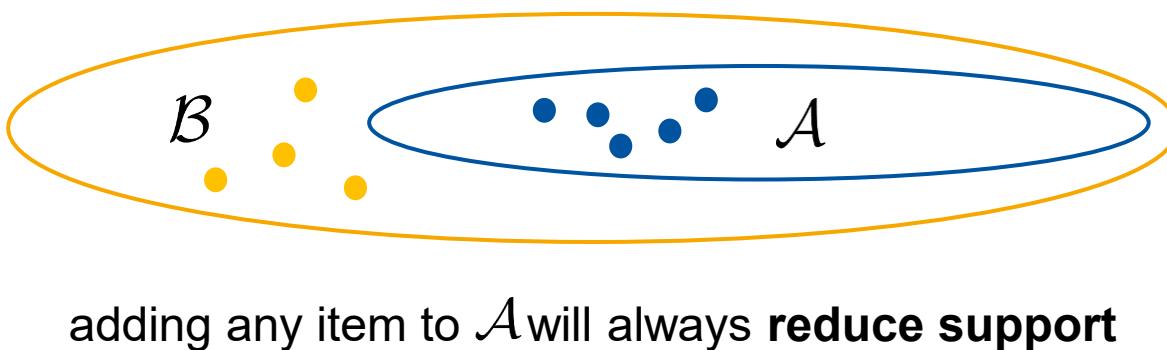
Subsets of Frequent Itemsets Are Also Frequent

- Assume $\mathcal{A} = \{I_1, I_2, \dots, I_{100}\}$ and $\text{support}(\mathcal{A}) \geq \text{min_sup}$
- All subsets of \mathcal{A} are also frequent
- There are $\binom{100}{1} = 100$ frequent itemsets having one item
- There are $\binom{100}{k}$ frequent itemsets having k items (“100 choose k “)
- There are $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{99} = 2^{100} - 2 = 1.27 \times 10^{30}$ smaller frequent itemsets contained in \mathcal{A}

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\dots(n-k+1)}{k(k-1)\dots1}$$

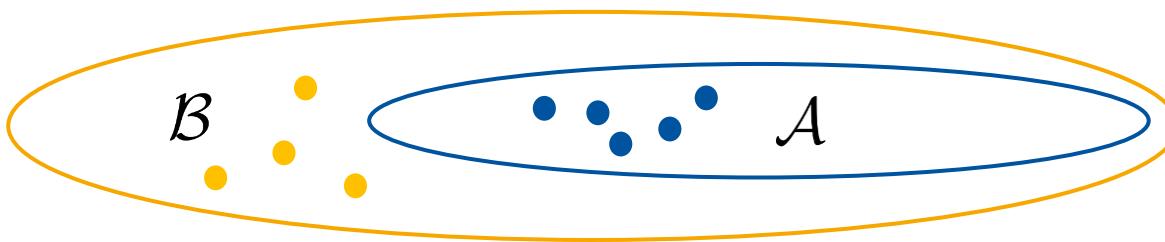
Closed Itemsets

- An itemset \mathcal{A} is **closed** if there is no **proper superset** $\mathcal{B} \supset \mathcal{A}$ that has the same support
- If \mathcal{A} is **closed**, then $\text{support}(\mathcal{A}) > \text{support}(\mathcal{B})$ for any $\mathcal{B} \supset \mathcal{A}$



Closed Frequent Itemsets

- An itemset \mathcal{A} is **closed** if there is no proper superset $\mathcal{B} \supset \mathcal{A}$ that has the same support
- If \mathcal{A} is **closed**, then $\text{support}(\mathcal{A}) > \text{support}(\mathcal{B})$ for any $\mathcal{B} \supset \mathcal{A}$
- \mathcal{A} is frequent if its support is higher than threshold **min_sup**

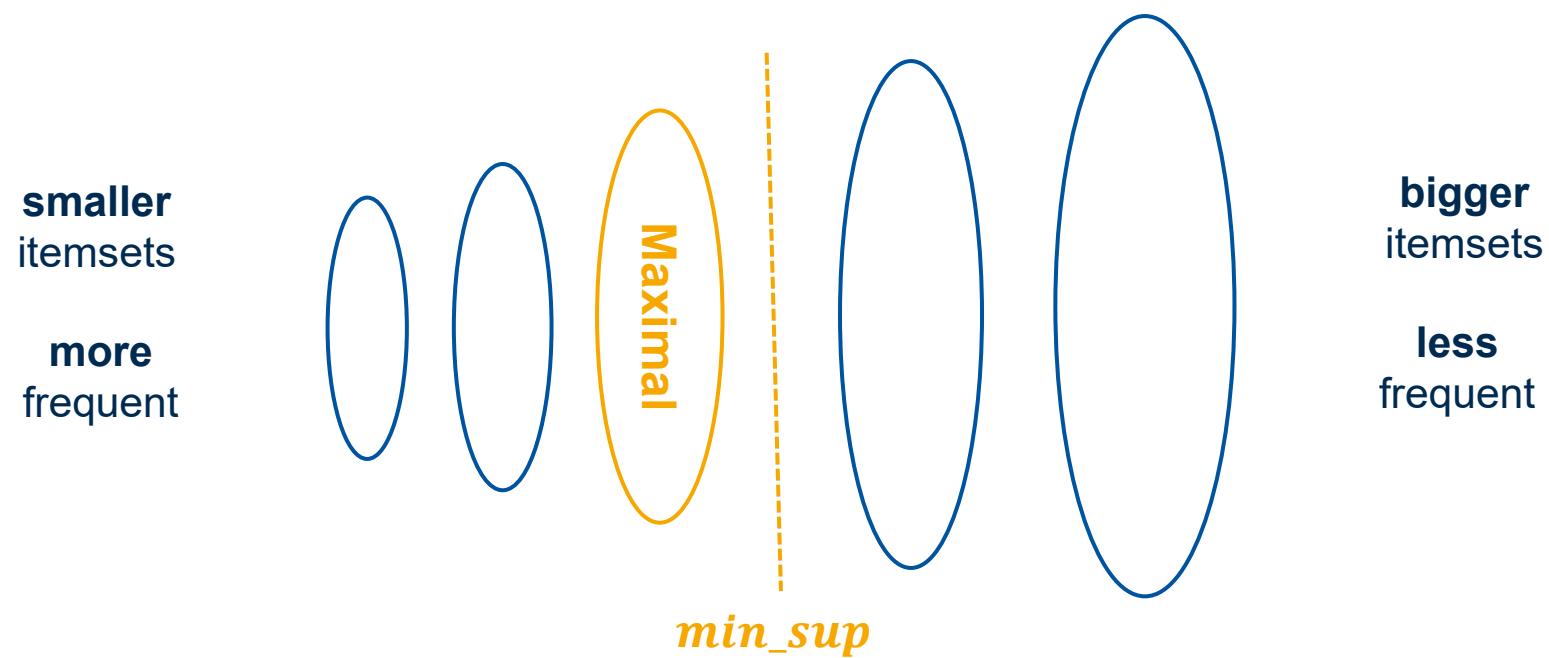


closed frequent itemsets are **closed and frequent**

Maximal Frequent Itemsets

An itemset \mathcal{A} is a maximal frequent itemset if:

- \mathcal{A} is frequent
- there is no proper superset $\mathcal{B} \supset \mathcal{A}$ that is also frequent



Relationships

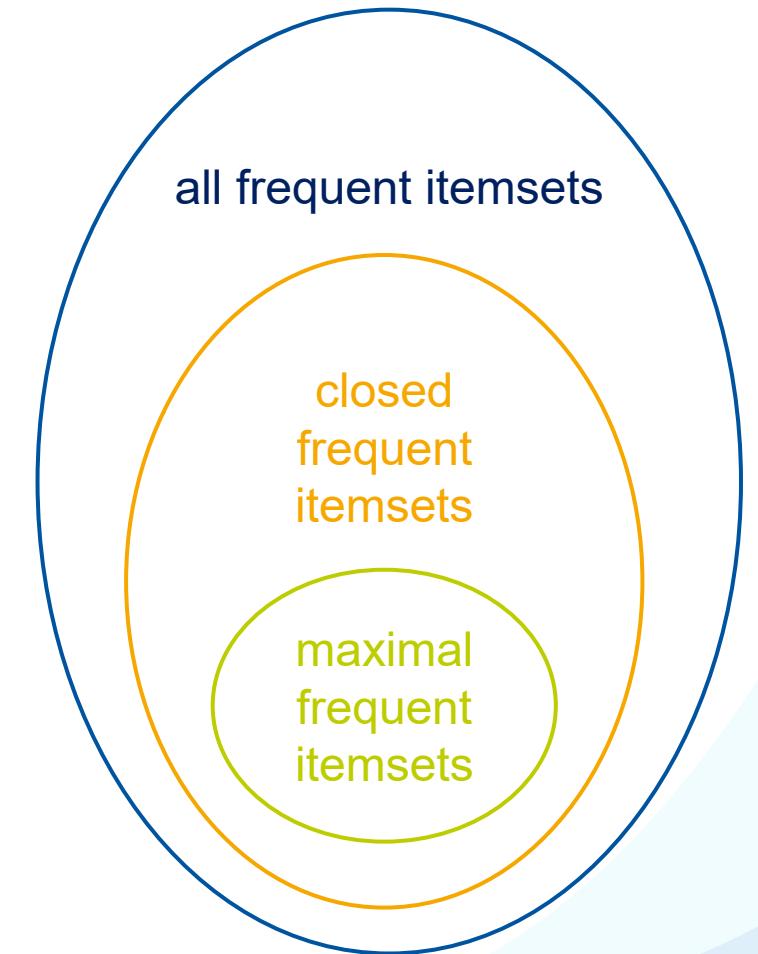
An itemset \mathcal{A} is a **closed frequent itemset** if:

- \mathcal{A} is frequent
- there is no proper superset $\mathcal{B} \supset \mathcal{A}$ that has the same support

An itemset \mathcal{A} is a **maximal frequent itemset** if:

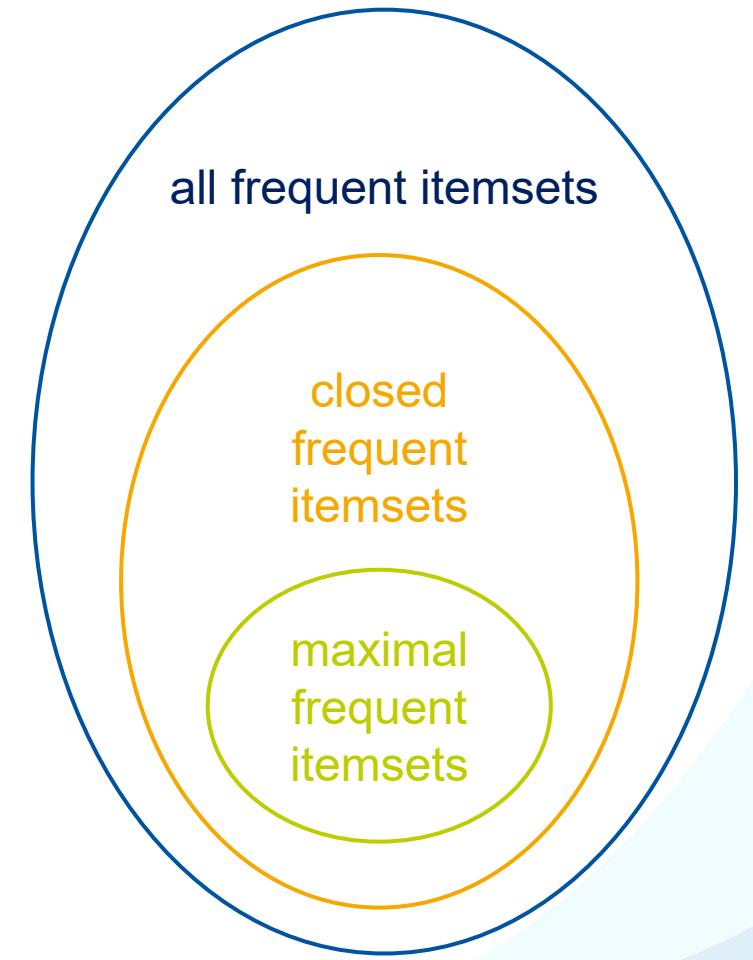
- \mathcal{A} is frequent
- there is no proper superset $\mathcal{B} \supset \mathcal{A}$ that is also frequent

Hence, maximal frequent itemsets are closed by definition.



Observations

- The supports of the **closed** frequent itemsets provide complete information about the supports of **all** frequent item sets
 - Formally, assume:
 - $\mathcal{A} \subset \mathcal{B}$,
 - \mathcal{B} is a **closed** frequent itemset, and
 - there is no **closed** frequent itemset \mathcal{B}' such that $\mathcal{A} \subseteq \mathcal{B}' \subset \mathcal{B}$.Then $support(\mathcal{A}) = support(\mathcal{B})$.
- It suffices to store **closed** frequent itemsets
(**maximal** frequent itemsets provide less information)



Frequent Itemsets

1. Introduction
2. Properties of Frequent Itemsets
3. **Apriori Algorithm**
4. FP-Growth Algorithm



Apriori Algorithm

- Introduced by Rakesh Agrawal and Ramakrishnan Srikant in “Fast Algorithms for Mining Association Rules in Large Databases. VLDB 1994: 487-499”
- Computes **frequent itemsets / association rules** in a dataset
- It illustrates the **Apriori Principle** often used to reduce the number of candidate patterns. It can be applied to sets, multisets, graphs, sequences, partial orders, etc.

Apriori Algorithm – Basic Idea

$$\mathcal{L}_k = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup} \wedge |\mathcal{A}| = k\}$$

frequent itemsets of length k

1. Candidate generation: use the set \mathcal{L}_k of frequent itemsets of length k to generate the candidate set \mathcal{C}_{k+1} of candidate itemsets with length $k+1$

Apriori Algorithm – Basic Idea

$$\mathcal{L}_k = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup} \wedge |\mathcal{A}| = k\}$$

frequent itemsets of length k

1. Candidate generation: use the set \mathcal{L}_k of frequent itemsets of length k to generate the candidate set \mathcal{C}_{k+1} of candidate itemsets with length $k+1$

does not
need the
input data
(efficient)

Apriori Algorithm – Basic Idea

$$\mathcal{L}_k = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup} \wedge |\mathcal{A}| = k\}$$

frequent itemsets of length k

1. **Candidate generation:** use the set \mathcal{L}_k of frequent itemsets of length k to generate the candidate set \mathcal{C}_{k+1} of candidate itemsets with length $k+1$
2. **Pruning (antimonotonicity):** all nonempty subsets of a frequent itemset must also be frequent → superset of an infrequent itemset cannot be frequent

does not
need the
input data
(efficient)

Apriori Algorithm – Basic Idea

$$\mathcal{L}_k = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup} \wedge |\mathcal{A}| = k\}$$

frequent itemsets of length k

1. **Candidate generation:** use the set \mathcal{L}_k of frequent itemsets of length k to generate the candidate set \mathcal{C}_{k+1} of candidate itemsets with length $k+1$
2. **Pruning (antimonotonicity):** all nonempty subsets of a frequent itemset must also be frequent → superset of an infrequent itemset cannot be frequent

does not
need the
input data
(efficient)

3. **Testing candidates:** use the dataset to filter the infrequent itemsets from \mathcal{C}_{k+1} and obtain \mathcal{L}_{k+1}

needs the
input data
(inefficient)

Apriori Algorithm – Basic Idea

$$\mathcal{L}_k = \{\mathcal{A} \subseteq \mathcal{I} \mid \text{support}(\mathcal{A}) \geq \text{min_sup} \wedge |\mathcal{A}| = k\}$$

frequent itemsets of length k

1. **Candidate generation:** use the set \mathcal{L}_k of frequent itemsets of length k to generate the candidate set \mathcal{C}_{k+1} of candidate itemsets with length $k+1$
2. **Pruning (antimonotonicity):** all nonempty subsets of a frequent itemset must also be frequent → superset of an infrequent itemset cannot be frequent
3. **Testing candidates:** use the dataset to filter the infrequent itemsets from \mathcal{C}_{k+1} and obtain \mathcal{L}_{k+1}

does not
need the
input data
(efficient)

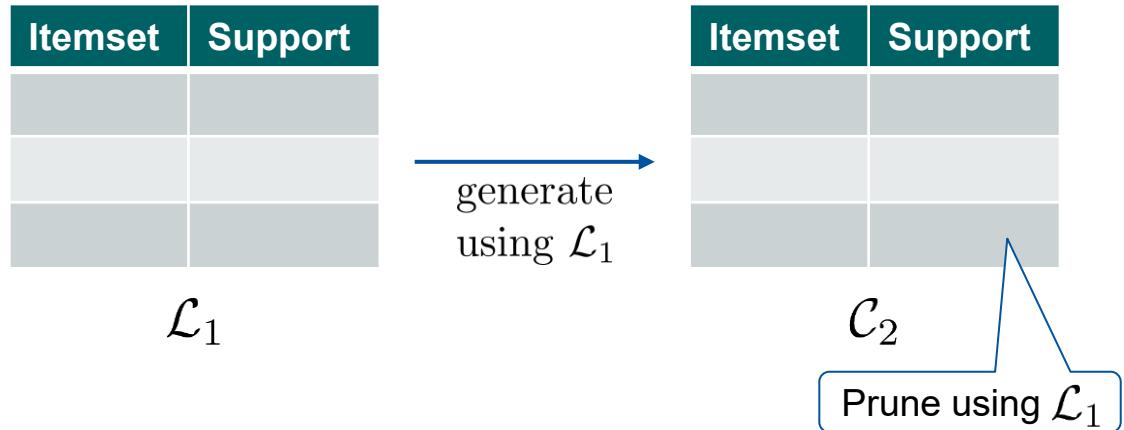
needs the
input data
(inefficient)

Apriori Algorithm – Basic Idea

| Itemset | Support |
|---------|---------|
| | |
| | |
| | |
| | |

 \mathcal{L}_1

Apriori Algorithm – Basic Idea



Apriori Algorithm – Basic Idea

| Itemset | Support |
|---------|---------|
| | |
| | |
| | |

 \mathcal{L}_1

generate
using \mathcal{L}_1

| Itemset | Support |
|---------|---------|
| | |
| | |
| | |

 \mathcal{C}_2

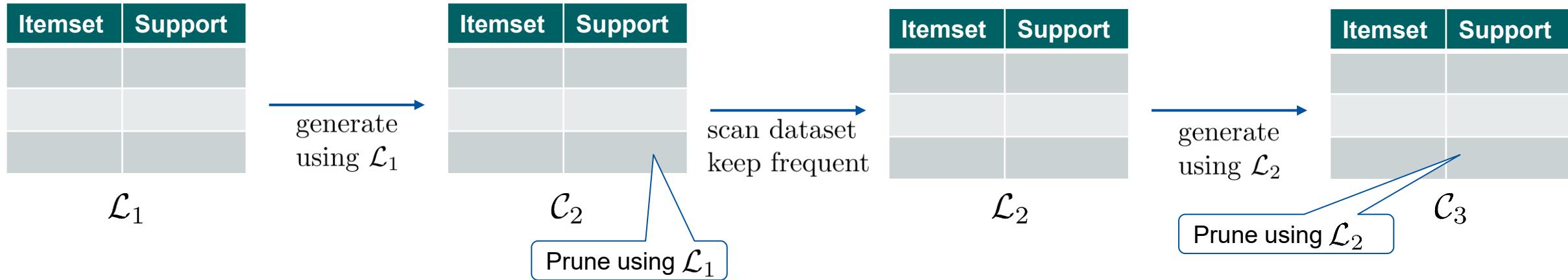
Prune using \mathcal{L}_1

| Itemset | Support |
|---------|---------|
| | |
| | |

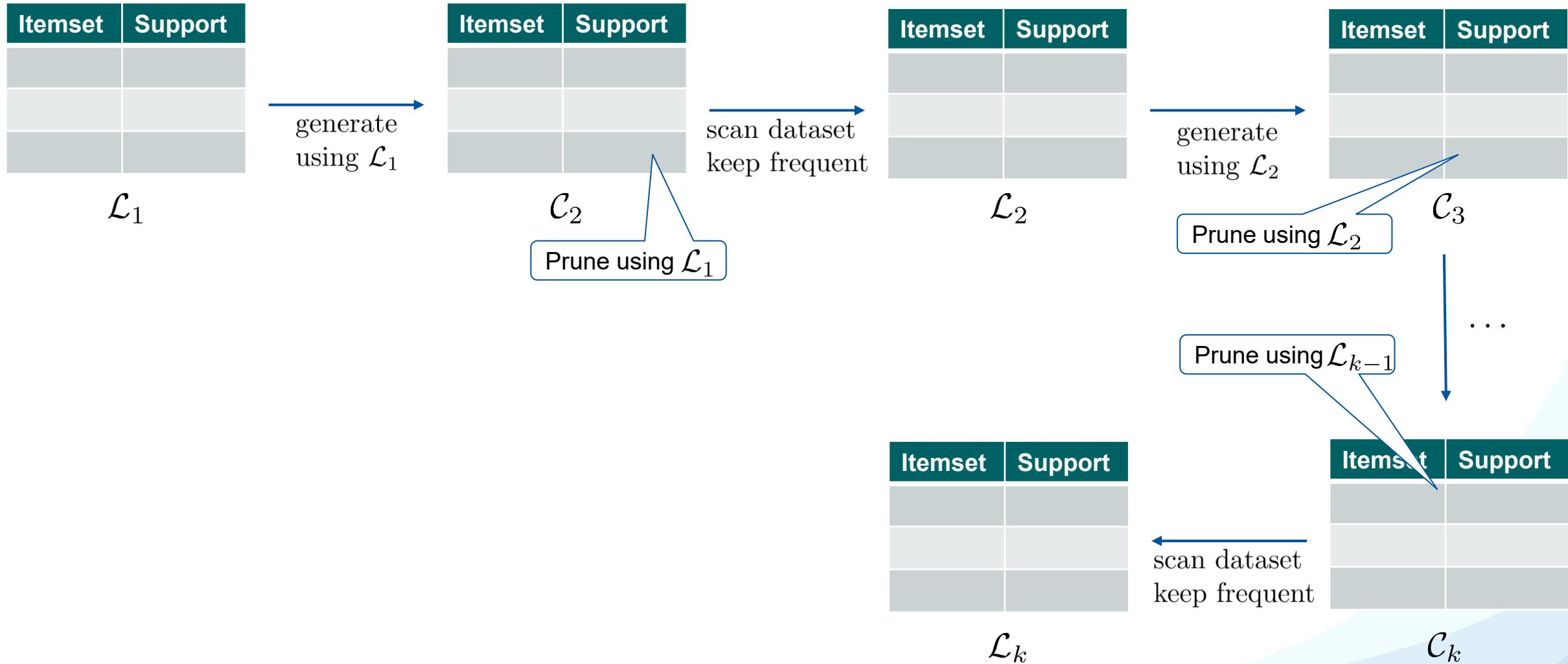
 \mathcal{L}_2

scan dataset
keep frequent

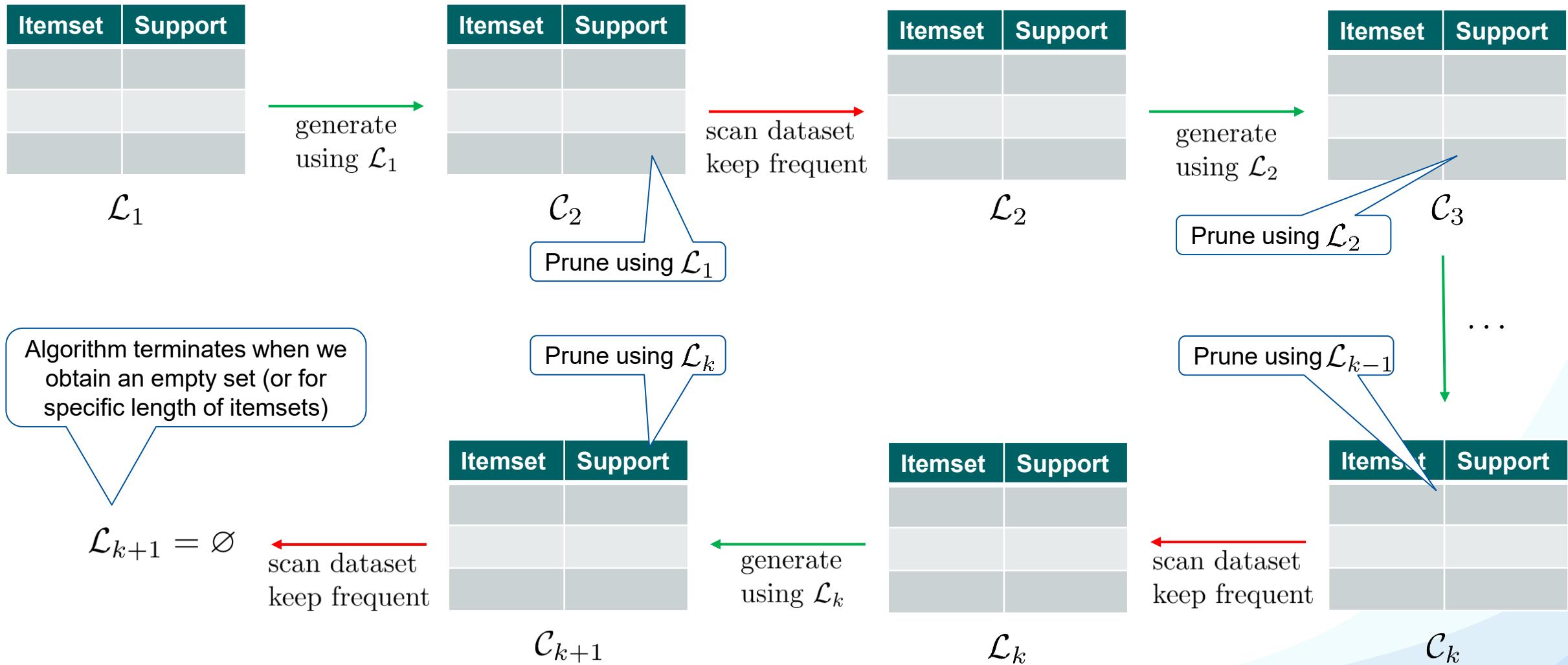
Apriori Algorithm – Basic Idea



Apriori Algorithm – Basic Idea



Apriori Algorithm – Basic Idea



Candidate Generation – Leveling

Leveling is used to generate candidate itemset \mathcal{C}_{k+1} from \mathcal{L}_k :

For any $\mathcal{A} \in \mathcal{L}_{k+1}$ there exist $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$ such that $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$

Hence, we can obtain the candidate itemsets $\mathcal{C}_{k+1} \supseteq \mathcal{L}_{k+1}$ by joining suitable $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$!

all itemsets of length $k + 1$

\mathcal{C}_{k+1} before pruning

\mathcal{C}_{k+1} after pruning

\mathcal{L}_{k+1}

Candidate Generation – Leveling

Leveling is used to generate candidate itemset \mathcal{C}_{k+1} from \mathcal{L}_k :

For any $\mathcal{A} \in \mathcal{L}_{k+1}$ there exist $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$ such that $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$

Assume that the items are ordered (I_1, I_2, \dots) and that
 $\mathcal{A} = \{I_1, I_2, \dots, I_{k-1}, I_k, I_{k+1}\} \in \mathcal{L}_{k+1}$

If \mathcal{A} is frequent, its subsets must be frequent, in particular:

$$\mathcal{A}' = \{I_1, I_2, \dots, I_{k-1}, I_k\} \in \mathcal{L}_k$$

$$\mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_{k+1}\} \in \mathcal{L}_k$$

$$\mathcal{A}' \cup \mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_k, I_{k+1}\} \in \mathcal{L}_{k+1} = \mathcal{A}$$

all itemsets of length $k + 1$

\mathcal{C}_{k+1} before pruning

\mathcal{C}_{k+1} after pruning

\mathcal{L}_{k+1}

Candidate Generation – Leveling

Leveling is used to generate candidate itemset \mathcal{C}_{k+1} from \mathcal{L}_k :

For any $\mathcal{A} \in \mathcal{L}_{k+1}$ there exist $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$ such that $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$

Assume that the items are ordered (I_1, I_2, \dots) and that
 $\mathcal{A} = \{I_1, I_2, \dots, I_{k-1}, I_k, I_{k+1}\} \in \mathcal{L}_{k+1}$

If \mathcal{A} is frequent, its subsets must be frequent, in particular:

$$\mathcal{A}' = \{I_1, I_2, \dots, I_{k-1}, I_k\} \in \mathcal{L}_k$$

$$\mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_{k+1}\} \in \mathcal{L}_k$$

$$\mathcal{A}' \cup \mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_k, I_{k+1}\} \in \mathcal{L}_{k+1} = \mathcal{A}$$

\Rightarrow We can generate \mathcal{C}_{k+1} by joining itemsets $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$ which differ in one item

all itemsets of length $k + 1$

\mathcal{C}_{k+1} before pruning

\mathcal{C}_{k+1} after pruning

\mathcal{L}_{k+1}

Candidate Generation

Thanks to **leveling**:

- Apriori creates the set of **candidate itemsets of length $k+1$** , \mathcal{C}_{k+1} , by joining two frequent itemsets of length k
- This can be done efficiently without creating duplicates
- Next, we **prune** the set \mathcal{C}_{k+1} based on infrequent subsets

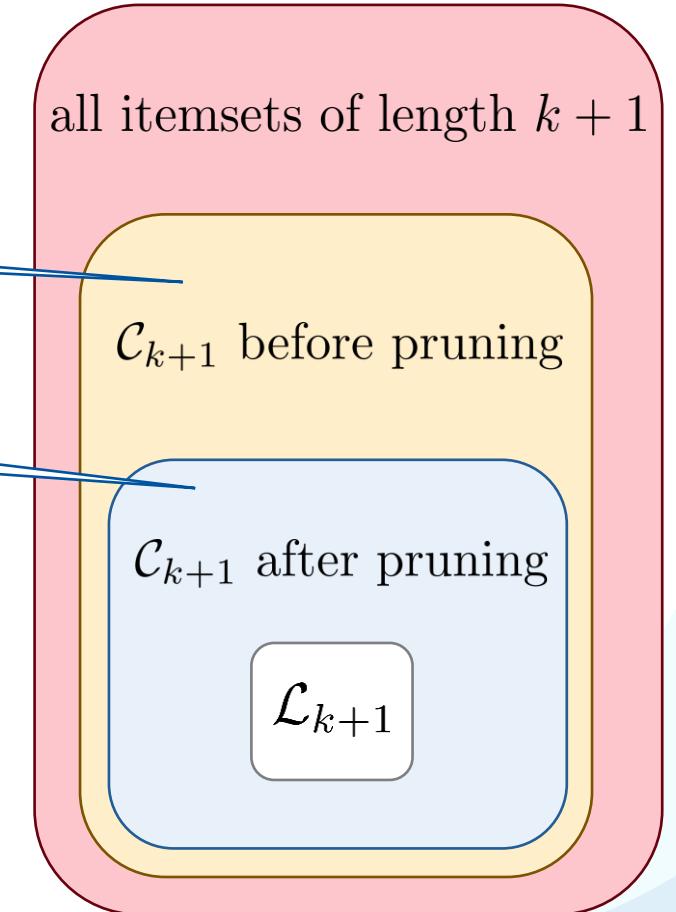
If \mathcal{A} is frequent, its subsets must be frequent, in particular:

$$\mathcal{A}' = \{I_1, I_2, \dots, I_{k-1}, I_k\} \in \mathcal{L}_k$$

$$\mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_{k+1}\} \in \mathcal{L}_k$$

$$\mathcal{A}' \cup \mathcal{A}'' = \{I_1, I_2, \dots, I_{k-1}, I_k, I_{k+1}\} \in \mathcal{L}_{k+1} = \mathcal{A}$$

⇒ We can generate \mathcal{C}_{k+1} by joining itemsets $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$ which differ in one item



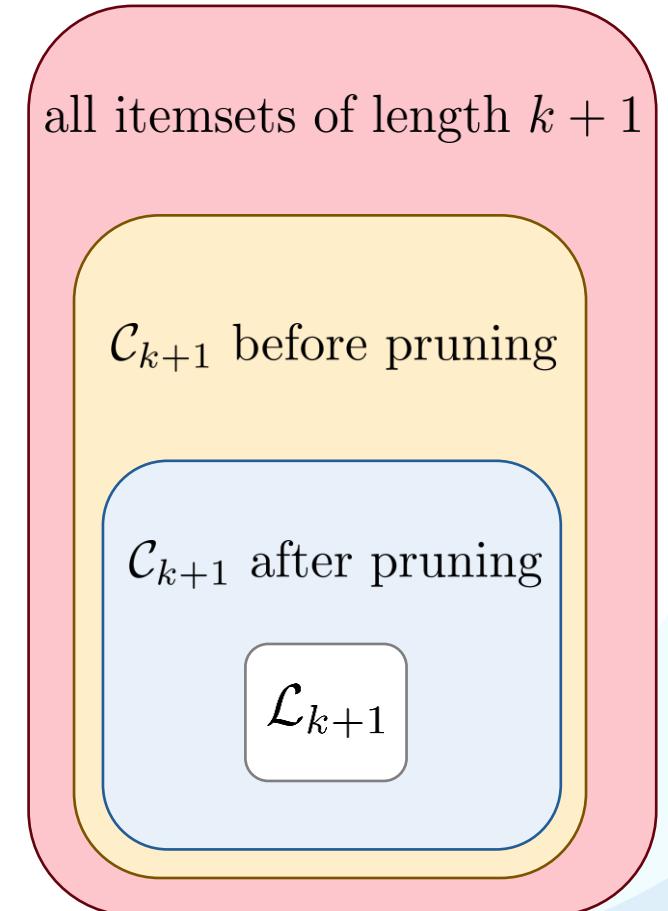
Pruning – Antimonotonicity

For any $\mathcal{A} \subseteq \mathcal{I}$ and $\mathcal{B} \subseteq \mathcal{I}$:

1. If $\mathcal{A} \subseteq \mathcal{B}$, then $\text{support}(\mathcal{A}) \geq \text{support}(\mathcal{B})$
2. If $\mathcal{A} \subseteq \mathcal{B}$ and $\text{support}(\mathcal{B}) \geq \text{min_sup}$,
then $\text{support}(\mathcal{A}) \geq \text{min_sup}$
3. If $\mathcal{A} \subseteq \mathcal{B}$ and $\text{support}(\mathcal{A}) < \text{min_sup}$,
then $\text{support}(\mathcal{B}) < \text{min_sup}$

Antimonotonicity is used to prune the candidate set:

If \mathcal{B} is a frequent itemset, any subset $\mathcal{A} \subseteq \mathcal{B}$ must be frequent
 \Rightarrow If a subset $\mathcal{A} \subseteq \mathcal{B}$ is infrequent, then \mathcal{B} is infrequent

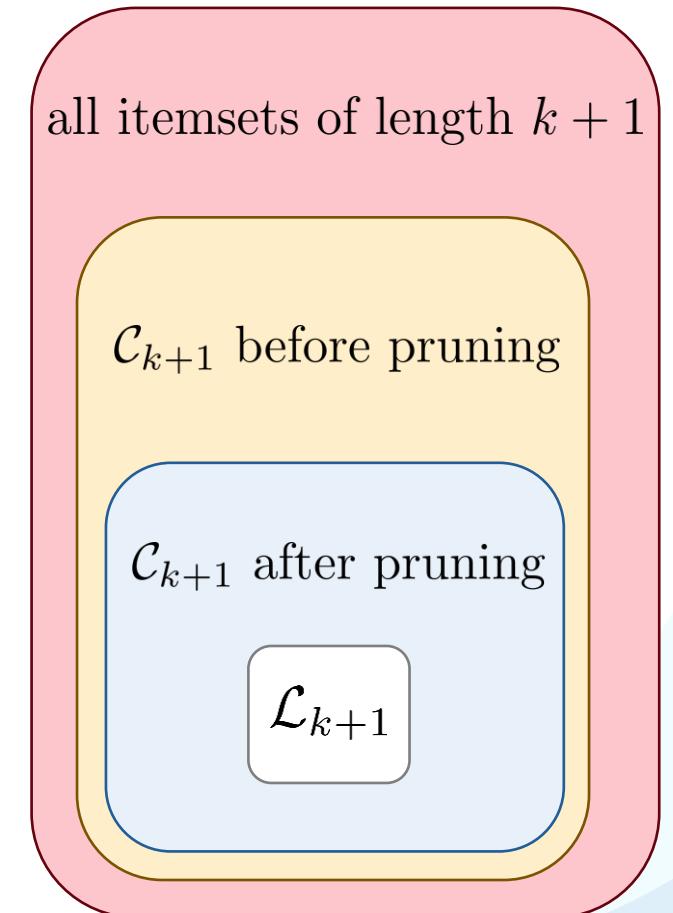


Pruning – Antimonotonicity

Antimonotonicity is used to prune the candidate set:

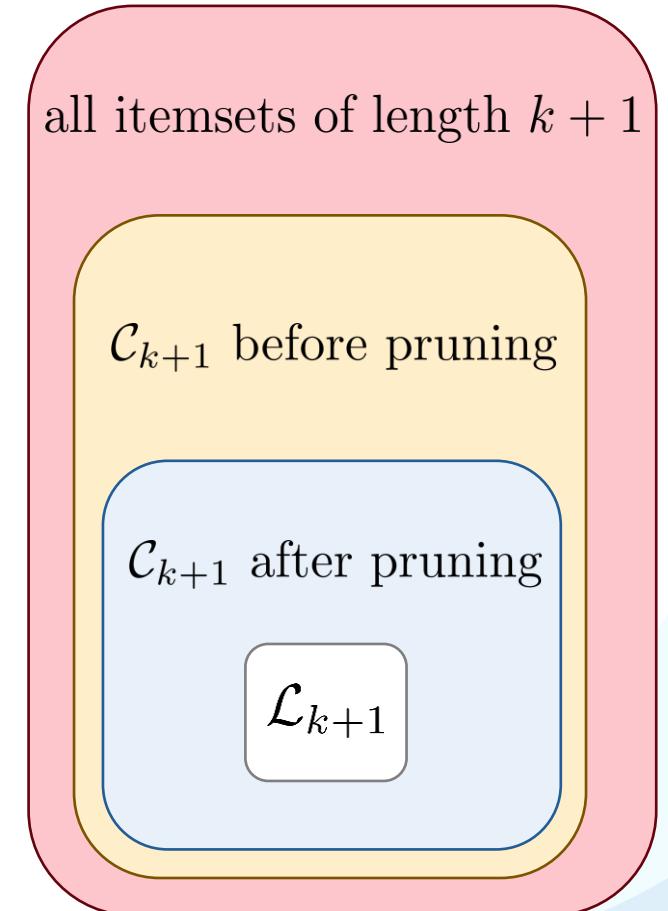
If \mathcal{B} is a frequent itemset, any subset $\mathcal{A} \subseteq \mathcal{B}$ must be frequent
 \Rightarrow If a subset $\mathcal{A} \subseteq \mathcal{B}$ is infrequent, then \mathcal{B} is infrequent

Candidate $\mathcal{A} = \{I_1, I_2, \dots, I_k\} \in \mathcal{C}_k$ can be removed if there is an $1 \leq i \leq k$ such that $\{I_1, I_2, \dots, I_k\} \setminus \{I_i\} \notin \mathcal{C}_k$



Testing Candidates

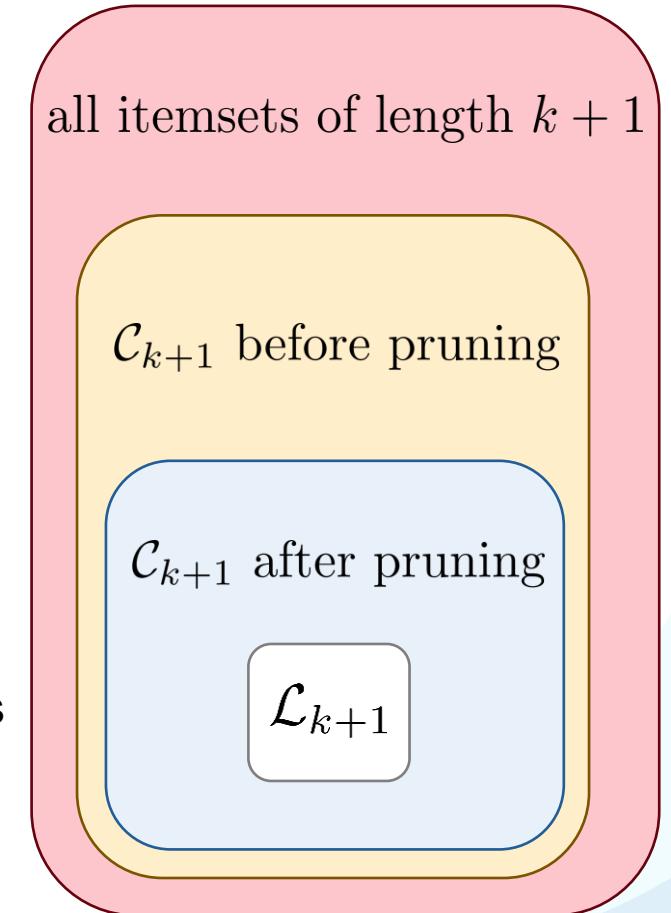
- After candidate generation and pruning, we test the remaining candidate itemsets
- We scan the dataset \mathcal{X} and remove all infrequent candidate itemsets from \mathcal{C}_{k+1} to obtain \mathcal{L}_{k+1}



Testing Candidates

- After candidate generation and pruning, we test the remaining candidate itemsets
- We scan the dataset \mathcal{X} and remove all infrequent candidate itemsets from \mathcal{C}_{k+1} to obtain \mathcal{L}_{k+1}
- Consider all transactions $\mathcal{T} \in \mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{I}))$
- For each candidate itemset $\mathcal{A} \in \mathcal{C}_k$ increment the corresponding counter if $\mathcal{A} \subseteq \mathcal{T}_k$
- This returns the frequencies (**support_count**) of the candidate itemsets and we can compute \mathcal{L}_{k+1} from \mathcal{C}_{k+1}

$$\mathcal{L}_{k+1} = \{\mathcal{A} \in \mathcal{C}_{k+1} \mid \text{support}(\mathcal{A}) \geq \text{min_sup}\}$$



Algorithm

Apriori algorithm:

1. Find the frequent itemsets of length 1 (\mathcal{L}_1)

2. Let $k \leftarrow 1$

3. **Repeat until** $\mathcal{L}_k = \emptyset$:

(a) Generate set of candidate itemsets of length $k+1$ (\mathcal{C}_{k+1})
based on \mathcal{L}_k

(b) Prune \mathcal{C}_{k+1} from itemsets that have infrequent subsets

(c) Test all remaining candidates from \mathcal{C}_{k+1} and remove
infrequent itemsets to obtain \mathcal{L}_{k+1}

(d) Assign $k \leftarrow k + 1$

4. **Return** $\bigcup_{i=1}^k \mathcal{L}_i$

(Or until we find
frequent itemsets of
pre-defined length)

all itemsets of length $k + 1$

\mathcal{C}_{k+1} before pruning

\mathcal{C}_{k+1} after pruning

\mathcal{L}_{k+1}

Example

| TID | Bought Fruits |
|-----|---------------------------------|
| 1 | {Grapes, Apple, Pineapple} |
| 2 | {Orange, Apple, Banana} |
| 3 | {Grapes, Orange, Apple, Banana} |
| 4 | {Orange, Banana} |
| 5 | {Grapes, Apple, Banana} |

 \mathcal{X} 

| Itemset | Count |
|-------------|-------|
| {Grapes} | 3 |
| {Apple} | 4 |
| {Pineapple} | 1 |
| {Orange} | 3 |
| {Banana} | 4 |

 \mathcal{C}_1

Example

| TID | Bought Fruits |
|-----|---------------------------------|
| 1 | {Grapes, Apple, Pineapple} |
| 2 | {Orange, Apple, Banana} |
| 3 | {Grapes, Orange, Apple, Banana} |
| 4 | {Orange, Banana} |
| 5 | {Grapes, Apple, Banana} |

 \mathcal{X} 

| Itemset | Count |
|-------------|-------|
| {Grapes} | 3 |
| {Apple} | 4 |
| {Pineapple} | 1 |
| {Orange} | 3 |
| {Banana} | 4 |

 \mathcal{C}_1

$\text{min_sup_count} = 2$

| Itemset | Count |
|-------------|-------|
| {Grapes} | 3 |
| {Apple} | 4 |
| {Pineapple} | 1 |
| {Orange} | 3 |
| {Banana} | 4 |

 \mathcal{L}_1

| Itemset | Count |
|-------------|-------|
| {Grapes} | 3 |
| {Apple} | 4 |
| {Pineapple} | 1 |
| {Orange} | 3 |
| {Banana} | 4 |

Example

| TID | Bought Fruits |
|-----|---------------------------------|
| 1 | {Grapes, Apple, Pineapple} |
| 2 | {Orange, Apple, Banana} |
| 3 | {Grapes, Orange, Apple, Banana} |
| 4 | {Orange, Banana} |
| 5 | {Grapes, Apple, Banana} |

 \mathcal{X}

| Itemset | Count |
|-------------|-------|
| {Grapes} | 3 |
| {Apple} | 4 |
| {Pineapple} | 1 |
| {Orange} | 3 |
| {Banana} | 4 |

 \mathcal{C}_1

| Itemset | Count |
|-------------|-------|
| {Grapes} | 3 |
| {Apple} | 4 |
| {Pineapple} | 1 |
| {Orange} | 3 |
| {Banana} | 4 |

 \mathcal{L}_1

| Itemset |
|------------------|
| {Grapes, Apple} |
| {Grapes, Orange} |
| {Grapes, Banana} |

| Itemset | Count |
|------------------|-------|
| {Grapes, Apple} | 3 |
| {Grapes, Orange} | 1 |
| {Grapes, Banana} | 2 |

generate candidates from \mathcal{L}_1

| Itemset |
|-----------------|
| {Apple, Orange} |
| {Apple, Banana} |

pruning based on \mathcal{L}_1

| Itemset |
|------------------|
| {Orange, Banana} |

 \mathcal{C}_2

| Itemset | Count |
|------------------|-------|
| {Grapes, Apple} | 3 |
| {Grapes, Orange} | 1 |
| {Grapes, Banana} | 2 |
| {Apple, Orange} | 2 |
| {Apple, Banana} | 3 |
| {Orange, Banana} | 3 |

 \mathcal{C}_2

Example

| TID | Bought Fruits |
|-----|---------------------------------|
| 1 | {Grapes, Apple, Pineapple} |
| 2 | {Orange, Apple, Banana} |
| 3 | {Grapes, Orange, Apple, Banana} |
| 4 | {Orange, Banana} |
| 5 | {Grapes, Apple, Banana} |

 \mathcal{X}

| Itemset | Count |
|-------------|-------|
| {Grapes} | 3 |
| {Apple} | 4 |
| {Pineapple} | 1 |
| {Orange} | 3 |
| {Banana} | 4 |

 \mathcal{C}_1

| Itemset | Count |
|-------------|-------|
| {Grapes} | 3 |
| {Apple} | 4 |
| {Pineapple} | 1 |
| {Orange} | 3 |
| {Banana} | 4 |

 \mathcal{L}_1

| Itemset |
|------------------|
| {Grapes, Apple} |
| {Grapes, Orange} |
| {Grapes, Banana} |
| {Apple, Orange} |
| {Apple, Banana} |
| {Orange, Banana} |

generate candidates from \mathcal{L}_1 pruning based on \mathcal{L}_1 \mathcal{C}_2

| Itemset | Count |
|------------------|-------|
| {Grapes, Apple} | 3 |
| {Grapes, Orange} | 1 |
| {Grapes, Banana} | 2 |
| {Apple, Orange} | 2 |
| {Apple, Banana} | 3 |
| {Orange, Banana} | 3 |

 \mathcal{C}_2 $\text{min_sup_count} = 2$ scan dataset
test candidates

| Itemset | Count |
|------------------|-------|
| {Grapes, Apple} | 3 |
| {Grapes, Orange} | 1 |
| {Grapes, Banana} | 2 |
| {Apple, Orange} | 2 |
| {Apple, Banana} | 3 |
| {Orange, Banana} | 3 |

 \mathcal{L}_2

Example

| TID | Bought Fruits |
|-----|---------------------------------|
| 1 | {Grapes, Apple, Pineapple} |
| 2 | {Orange, Apple, Banana} |
| 3 | {Grapes, Orange, Apple, Banana} |
| 4 | {Orange, Banana} |
| 5 | {Grapes, Apple, Banana} |

 \mathcal{X} 

...

min_sup_count = 2

| Itemset | Count |
|------------------|-------|
| {Grapes, Apple} | 3 |
| {Grapes, Banana} | 2 |
| {Apple, Orange} | 2 |
| {Apple, Banana} | 3 |
| {Orange, Banana} | 3 |

 \mathcal{L}_2

Example

| TID | Bought Fruits |
|-----|---------------------------------|
| 1 | {Grapes, Apple, Pineapple} |
| 2 | {Orange, Apple, Banana} |
| 3 | {Grapes, Orange, Apple, Banana} |
| 4 | {Orange, Banana} |
| 5 | {Grapes, Apple, Banana} |

 χ

→ ... → $\min_sup_count = 2$

| Itemset | Count |
|------------------|-------|
| {Grapes, Apple} | 3 |
| {Grapes, Banana} | 2 |
| {Apple, Orange} | 2 |
| {Apple, Banana} | 3 |
| {Orange, Banana} | 3 |

 \mathcal{L}_2

| Itemset |
|--------------------------|
| {Grapes, Apple, Banana} |
| {Grapes, Apple, Orange} |
| {Grapes, Banana, Orange} |
| {Apple, Banana, Orange} |

generate candidates from \mathcal{L}_2

| Itemset | Subsets of Length 2 |
|--------------------------|--|
| {Grapes, Apple, Banana} | {Grapes, Apple}, {Grapes, Banana}, {Apple, Banana} |
| {Grapes, Apple, Orange} | {Grapes, Apple}, {Grapes, Orange}, {Apple, Orange} |
| {Grapes, Banana, Orange} | {Grapes, Banana}, {Grapes, Orange}, {Banana, Orange} |
| {Apple, Banana, Orange} | {Apple, Banana}, {Apple, Orange}, {Banana, Orange} |

pruning based on \mathcal{L}_2

| Itemset | Count |
|-------------------------|-------|
| {Grapes, Apple, Banana} | 2 |
| {Apple, Banana, Orange} | 2 |

$\min_sup_count = 2$

scan dataset
test candidates

 \mathcal{L}_3 \mathcal{C}_3 \mathcal{C}_3

Example

| TID | Bought Fruits |
|-----|---------------------------------|
| 1 | {Grapes, Apple, Pineapple} |
| 2 | {Orange, Apple, Banana} |
| 3 | {Grapes, Orange, Apple, Banana} |
| 4 | {Orange, Banana} |
| 5 | {Grapes, Apple, Banana} |

 \mathcal{X} 

...

min_sup_count = 2

| Itemset | Count |
|-------------------------|-------|
| {Grapes, Apple, Banana} | 2 |
| {Apple, Banana, Orange} | 2 |

 \mathcal{L}_3

Example

| TID | Bought Fruits |
|-----|---------------------------------|
| 1 | {Grapes, Apple, Pineapple} |
| 2 | {Orange, Apple, Banana} |
| 3 | {Grapes, Orange, Apple, Banana} |
| 4 | {Orange, Banana} |
| 5 | {Grapes, Apple, Banana} |

 \mathcal{X} 

min_sup_count = 2

| Itemset | Count |
|-------------------------|-------|
| {Grapes, Apple, Banana} | 2 |
| {Apple, Banana, Orange} | 2 |

 \mathcal{L}_3

generate candidates from \mathcal{L}_3

| Itemset |
|---------------------------------|
| {Grapes, Apple, Banana, Orange} |

 \mathcal{C}_4

pruning based on \mathcal{L}_3

| Itemset | Subsets of length 3 |
|---------------------------------|---|
| {Grapes, Apple, Banana, Orange} | {Grapes, Apple, Banana}, {Grapes, Apple, Orange}, {Grapes, Banana, Orange}, {Apple, Banana, Orange} |
| | |
| | |

No more candidates of length 4 – algorithm terminates

Limitations

- It may remain challenging to generate the candidate sets (may be huge)
 - Each candidate needs to be tested against the whole dataset
- FP-Growth is an approach that aims to overcome these limitations

Frequent Itemsets

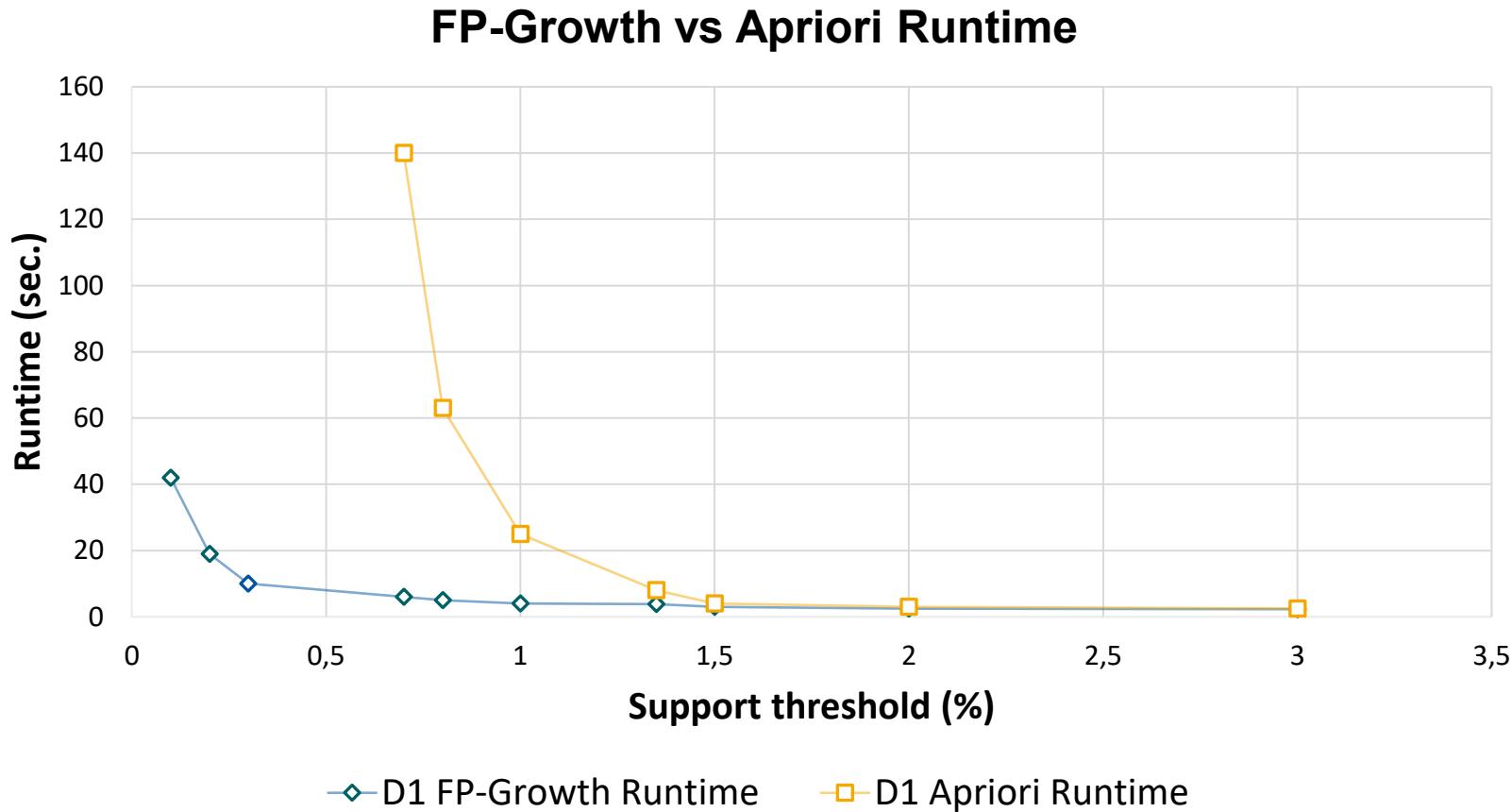
1. Introduction
2. Properties of Frequent Itemsets
3. Apriori Algorithm
4. **FP-Growth Algorithm**



Frequent Pattern Growth Algorithm

- Introduced by Jiawei Han, Jian Pei, Yiwen Yin in “Mining Frequent Patterns without Candidate Generation. SIGMOD Conference 2000: 1-12”
- Based on constructing the [Frequent Pattern Tree \(FP-Tree\)](#)
- Avoids generation of many candidates
- [Depth-first](#) rather than breadth-first
- Requires [only two passes](#) over the (potentially huge) dataset

Motivation



FP-Growth Steps

1. Determine the frequency of each item ([first](#) pass through the dataset)
2. Sort $\mathcal{I} = \{I_1, \dots, I_D\}$ based on their frequencies (I_1 is most frequent, I_D is the least frequent)
3. Remove the non-frequent items
4. The remaining items in each transactions are [ordered by frequency](#) (same as above)
5. This can be used to build a so-called [prefix tree](#) ([second](#) pass trough the dataset)

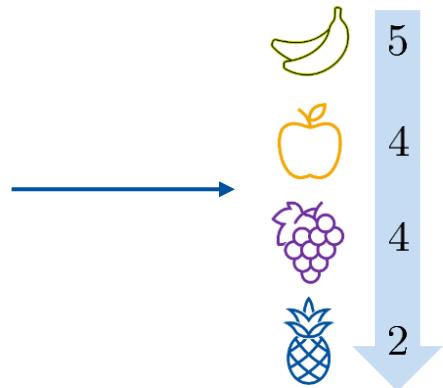
FP-Growth Steps

1. Determine the frequency of each item ([first](#) pass through the dataset)
2. Sort $\mathcal{I} = \{I_1, \dots, I_D\}$ based on their frequencies (I_1 is most frequent, I_D is the least frequent)
3. Remove the non-frequent items from all itemsets (keep the remainder of the transactions)
4. The remaining items in each transactions are [ordered by frequency](#) (same as above)
5. This can be used to build a so-called [prefix tree](#) ([second](#) pass trough the dataset)
6. The resulting FP-tree contains all information needed to find the frequent itemsets of any length
[\(no need to traverse the dataset again\)](#)

Constructing FP-Tree – Example

| TID | Bought Fruits |
|-----|-----------------------------|
| 1 | {Banana, Apple} |
| 2 | {Grapes, Banana, Pineapple} |
| 3 | {Apple, Banana} |
| 4 | {Apple, Grapes, Pineapple} |
| 5 | {Grapes, Banana} |
| 6 | {Apple, Banana, Grapes} |

Dataset $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{J}))$

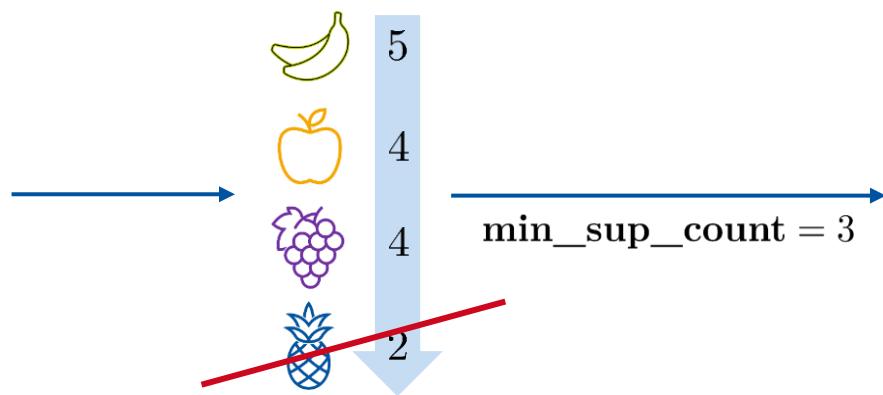


1. Determine the frequencies of items
2. Order the items based on frequency

Constructing FP-Tree – Example

| TID | Bought Fruits |
|-----|-----------------------------|
| 1 | {Banana, Apple} |
| 2 | {Grapes, Banana, Pineapple} |
| 3 | {Apple, Banana} |
| 4 | {Apple, Grapes, Pineapple} |
| 5 | {Grapes, Banana} |
| 6 | {Apple, Banana, Grapes} |

Dataset $\mathcal{X} \in \mathbb{M}(\mathbb{P}(\mathcal{J}))$



1. Determine the frequencies of items
2. Order the items based on frequency
3. Remove non-frequent items
4. Sort the items in the transactions

| TID | Bought Fruits |
|-----|-------------------------|
| 1 | {Banana, Apple} |
| 2 | {Banana, Grapes} |
| 3 | {Banana, Apple} |
| 4 | {Apple, Grapes} |
| 5 | {Banana, Grapes} |
| 6 | {Banana, Apple, Grapes} |

Constructing FP-Tree – Example

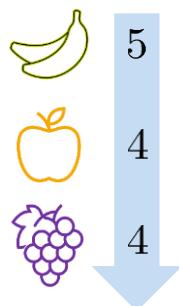
| TID | Bought Fruits |
|-----|-------------------------|
| 1 | {Banana, Apple} |
| 2 | {Banana, Grapes} |
| 3 | {Banana, Apple} |
| 4 | {Apple, Grapes} |
| 5 | {Banana, Grapes} |
| 6 | {Banana, Apple, Grapes} |

5. Build the FP-tree going through each transaction

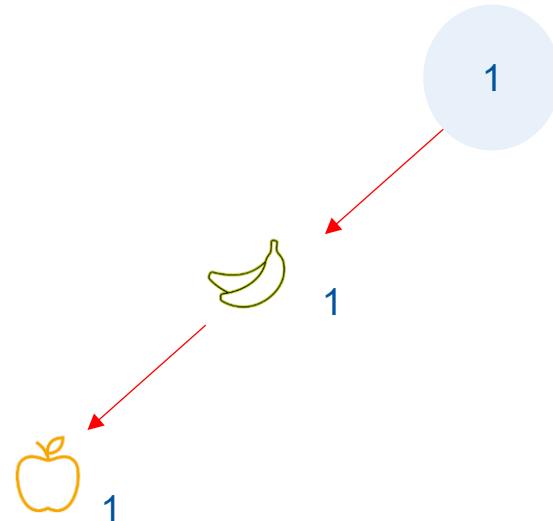


Constructing FP-Tree – Example

| TID | Bought Fruits |
|-----|-------------------------|
| 1 | {Banana, Apple} |
| 2 | {Banana, Grapes} |
| 3 | {Banana, Apple} |
| 4 | {Apple, Grapes} |
| 5 | {Banana, Grapes} |
| 6 | {Banana, Apple, Grapes} |

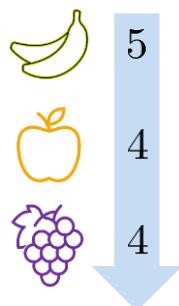


5. Build the FP-tree going through each transaction

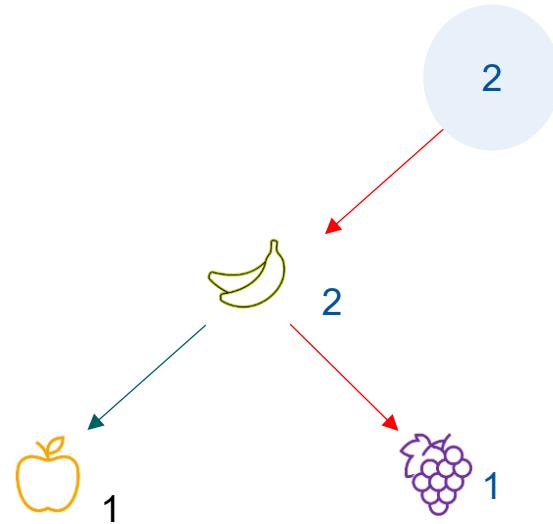


Constructing FP-Tree – Example

| TID | Bought Fruits |
|-----|-------------------------|
| 1 | {Banana, Apple} |
| 2 | {Banana, Grapes} |
| 3 | {Banana, Apple} |
| 4 | {Apple, Grapes} |
| 5 | {Banana, Grapes} |
| 6 | {Banana, Apple, Grapes} |

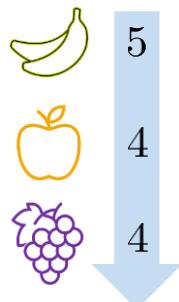


5. Build the FP-tree going through each transaction

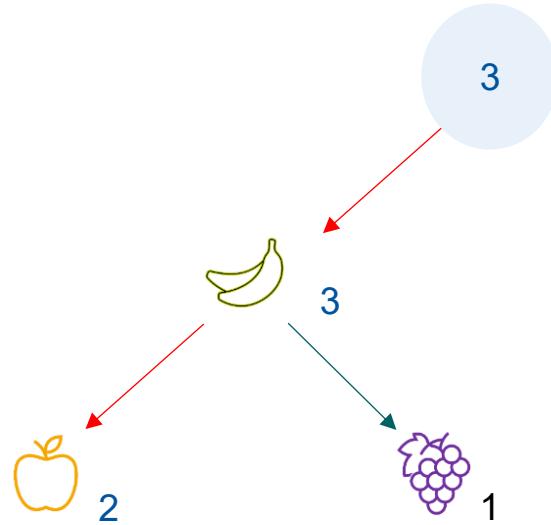


Constructing FP-Tree – Example

| TID | Bought Fruits |
|-----|-------------------------|
| 1 | {Banana, Apple} |
| 2 | {Banana, Grapes} |
| 3 | {Banana, Apple} |
| 4 | {Apple, Grapes} |
| 5 | {Banana, Grapes} |
| 6 | {Banana, Apple, Grapes} |

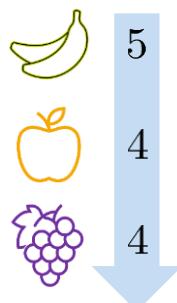


5. Build the FP-tree going through each transaction

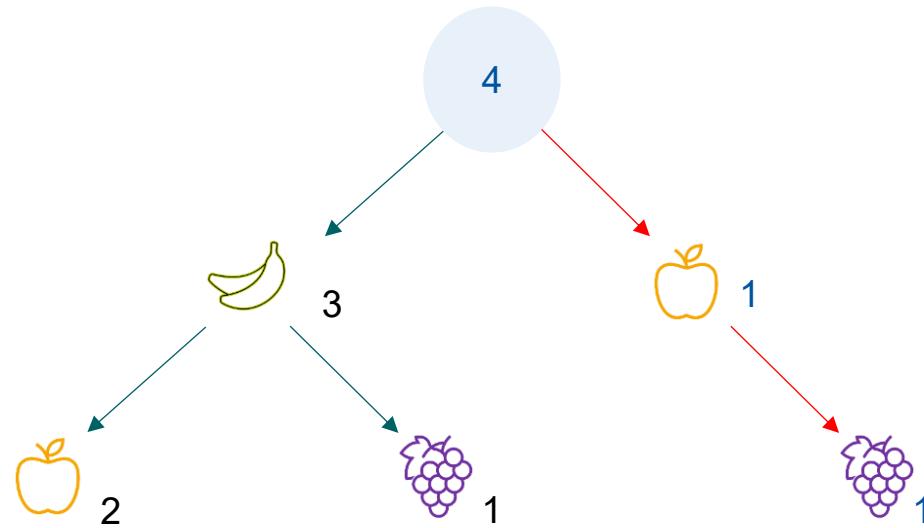


Constructing FP-Tree – Example

| TID | Bought Fruits |
|-----|-------------------------|
| 1 | {Banana, Apple} |
| 2 | {Banana, Grapes} |
| 3 | {Banana, Apple} |
| 4 | Apple, Grapes |
| 5 | {Banana, Grapes} |
| 6 | {Banana, Apple, Grapes} |

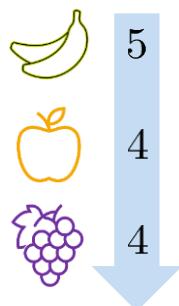


5. Build the FP-tree going through each transaction

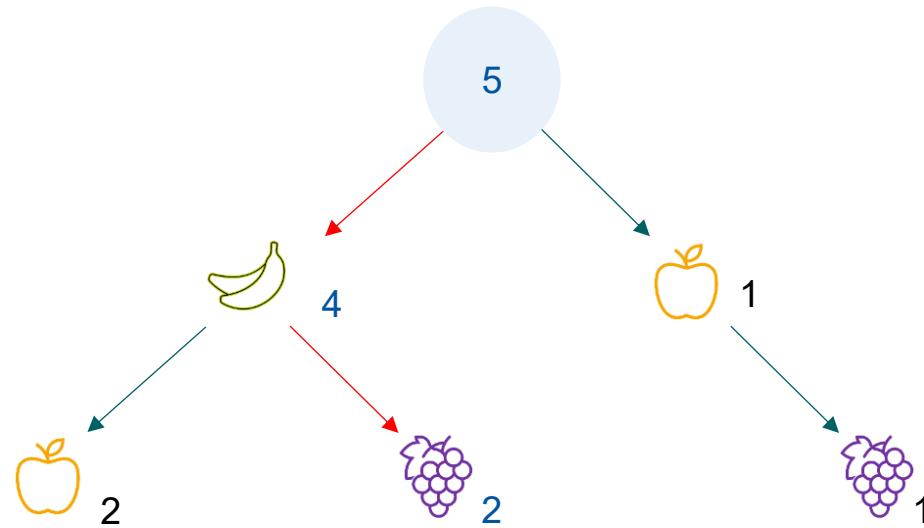


Constructing FP-Tree – Example

| TID | Bought Fruits |
|-----|-------------------------|
| 1 | {Banana, Apple} |
| 2 | {Banana, Grapes} |
| 3 | {Banana, Apple} |
| 4 | {Apple, Grapes} |
| 5 | Banana, Grapes |
| 6 | {Banana, Apple, Grapes} |

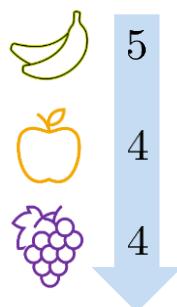


5. Build the FP-tree going through each transaction

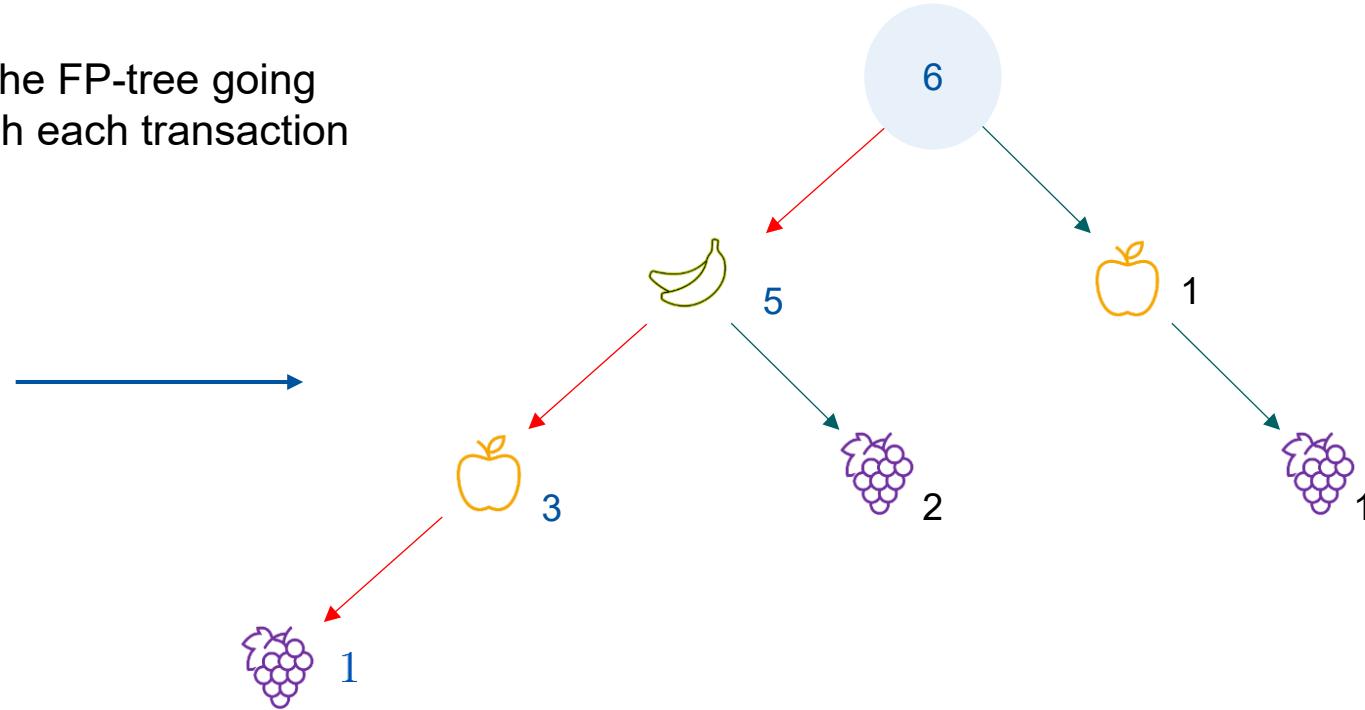


Constructing FP-Tree – Example

| TID | Bought Fruits |
|-----|--------------------------------|
| 1 | {Banana, Apple} |
| 2 | {Banana, Grapes} |
| 3 | {Banana, Apple} |
| 4 | {Apple, Grapes} |
| 5 | {Banana, Grapes} |
| 6 | {Banana, Apple, Grapes} |



5. Build the FP-tree going through each transaction

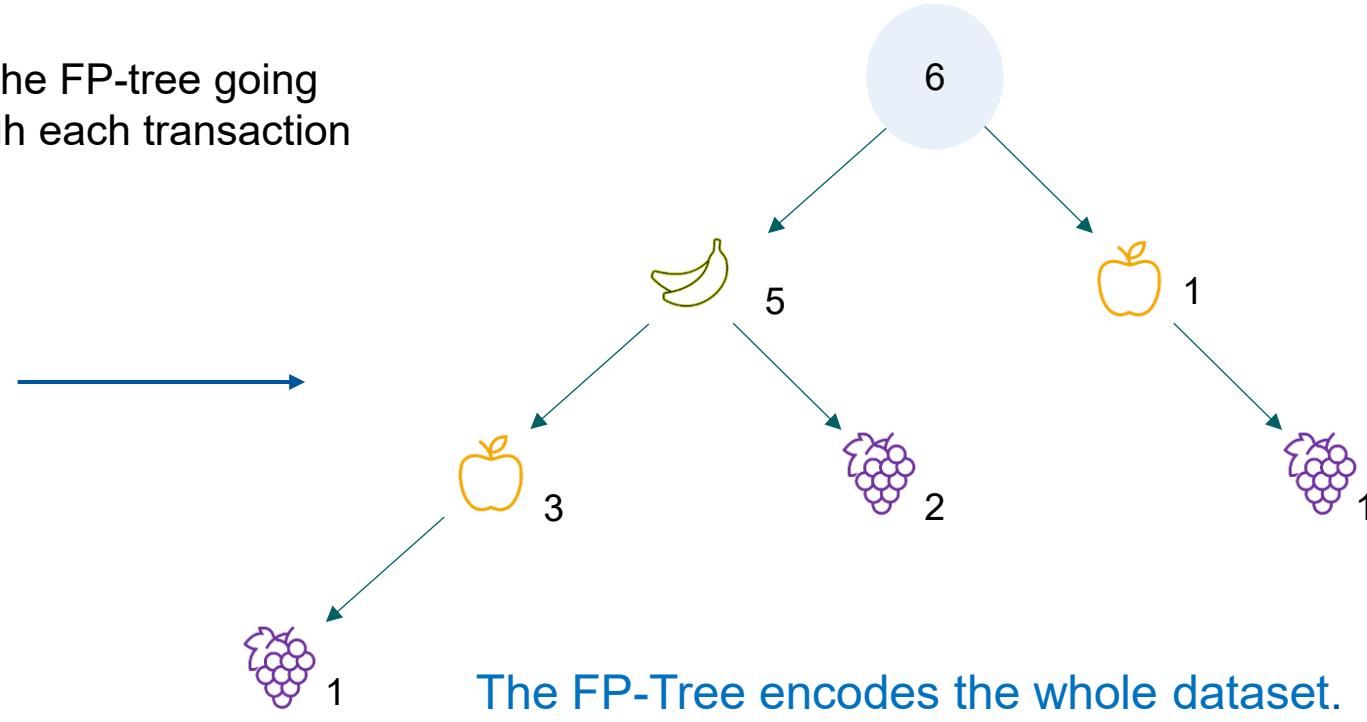


Constructing FP-Tree – Example

| TID | Bought Fruits |
|-----|-------------------------|
| 1 | {Banana, Apple} |
| 2 | {Banana, Grapes} |
| 3 | {Banana, Apple} |
| 4 | {Apple, Grapes} |
| 5 | {Banana, Grapes} |
| 6 | {Banana, Apple, Grapes} |



5. Build the FP-tree going through each transaction

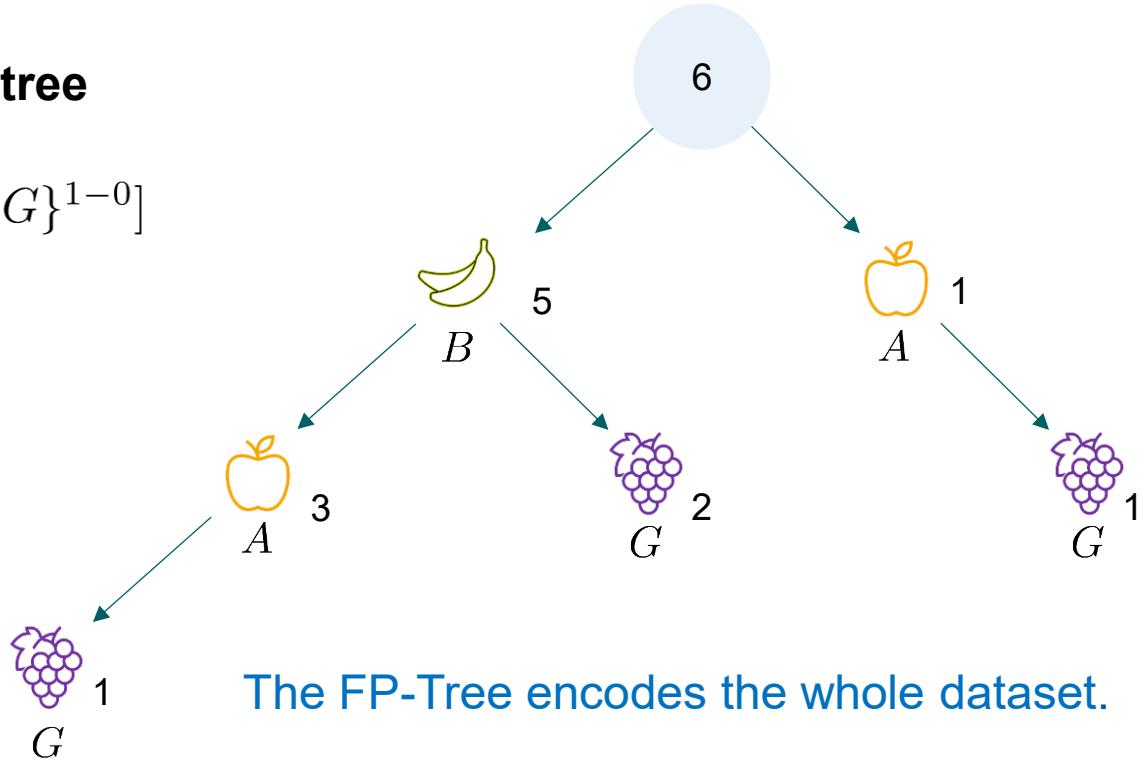


The FP-Tree encodes the whole dataset.

FP-Tree – Encodes The Dataset

We can read the transactions from the FP-tree

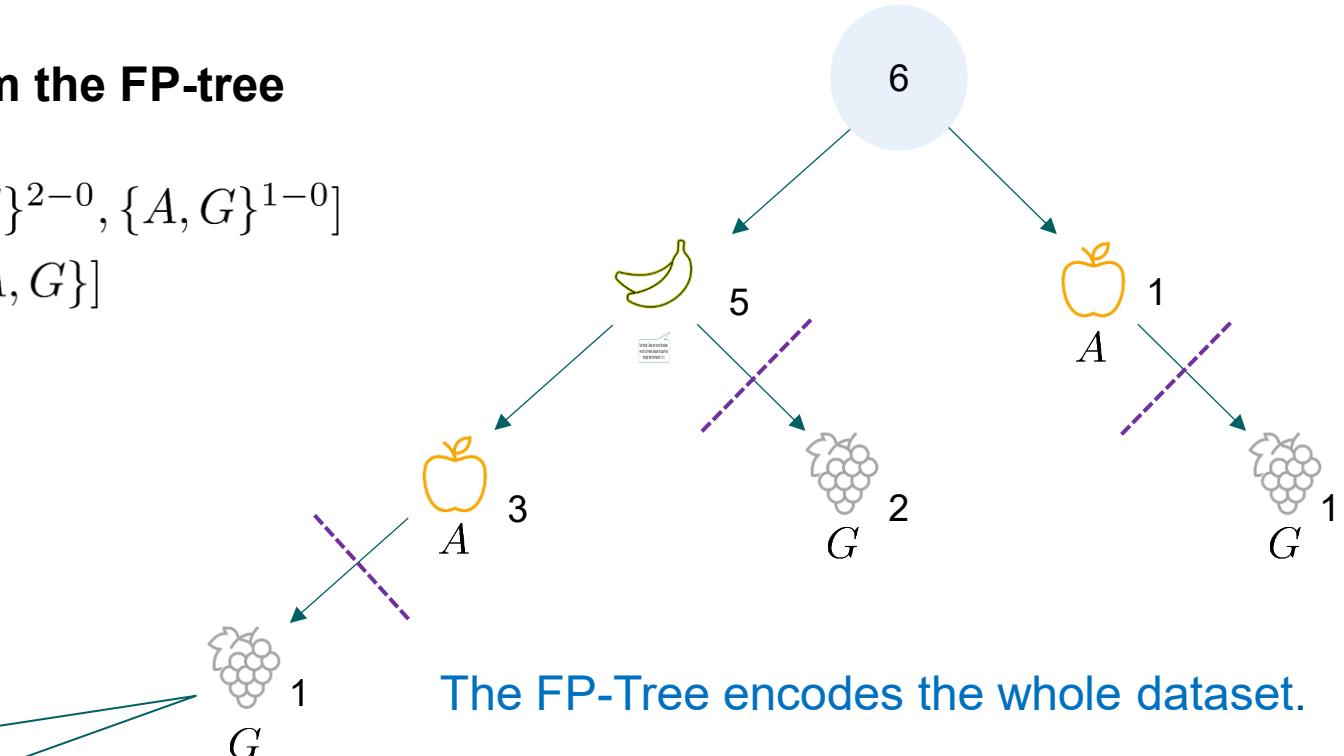
$$\begin{aligned}\mathcal{X} &= [\{B, A, G\}^{1-0}, \{B, A\}^{3-1}, \{B, G\}^{2-0}, \{A, G\}^{1-0}] \\ &= [\{B, A, G\}, \{B, A\}^2, \{B, G\}^2, \{A, G\}]\end{aligned}$$



FP-Tree – Cannot Cut Naïvely

We can read the transactions from the FP-tree

$$\begin{aligned}\mathcal{X} &= [\{B, A, G\}^{1-0}, \{B, A\}^{3-1}, \{B, G\}^{2-0}, \{A, G\}^{1-0}] \\ &= [\{B, A, G\}, \{B, A\}^2, \{B, G\}^2, \{A, G\}]\end{aligned}$$

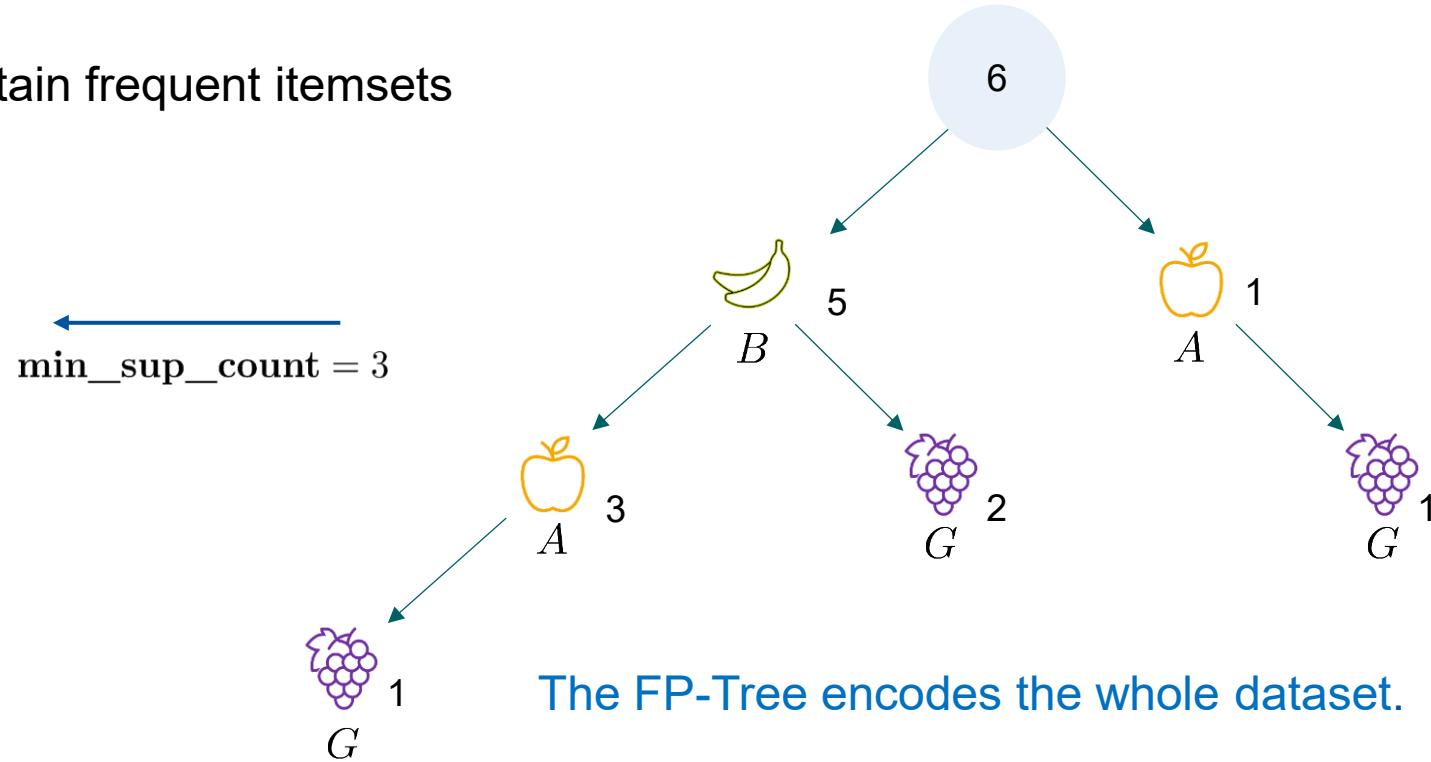


Even though G exists only once in this subtree, we can't cut it naively, because its support may be higher than the threshold ($4 \geq 3$)

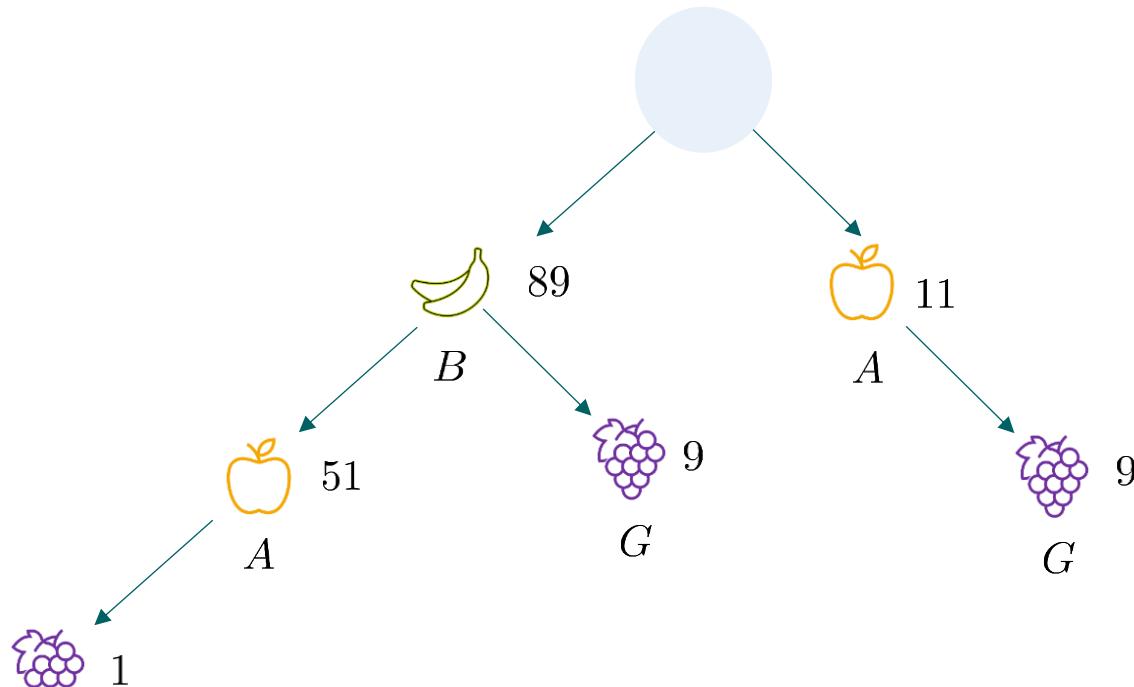
FP-Tree – Frequent Itemsets

Next: Mining the FP-tree to obtain frequent itemsets

| Frequent Itemsets | Support Count |
|-------------------|---------------|
| {B} | 5 |
| {A} | 4 |
| {G} | 4 |
| {B, A} | 3 |
| {B, G} | 3 |



FP-Tree Encodes Dataset – Another Example



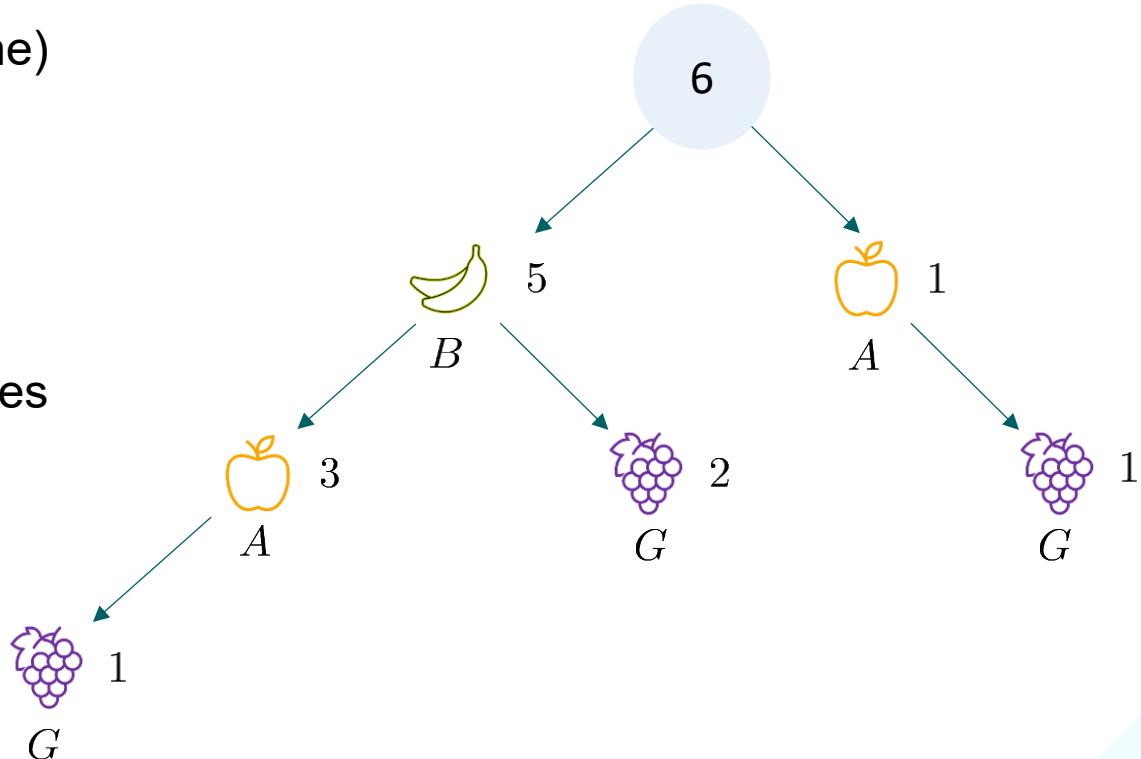
We can read the transactions from the FP-tree

$$\mathcal{X} = [\{B, A, G\}^{1-0}, \{B, A\}^{51-1}, \{B, G\}^{9-0}, \{B\}^{89-(51+9)}, \{A, G\}^{9-0}, \{A\}^{11-9}]$$

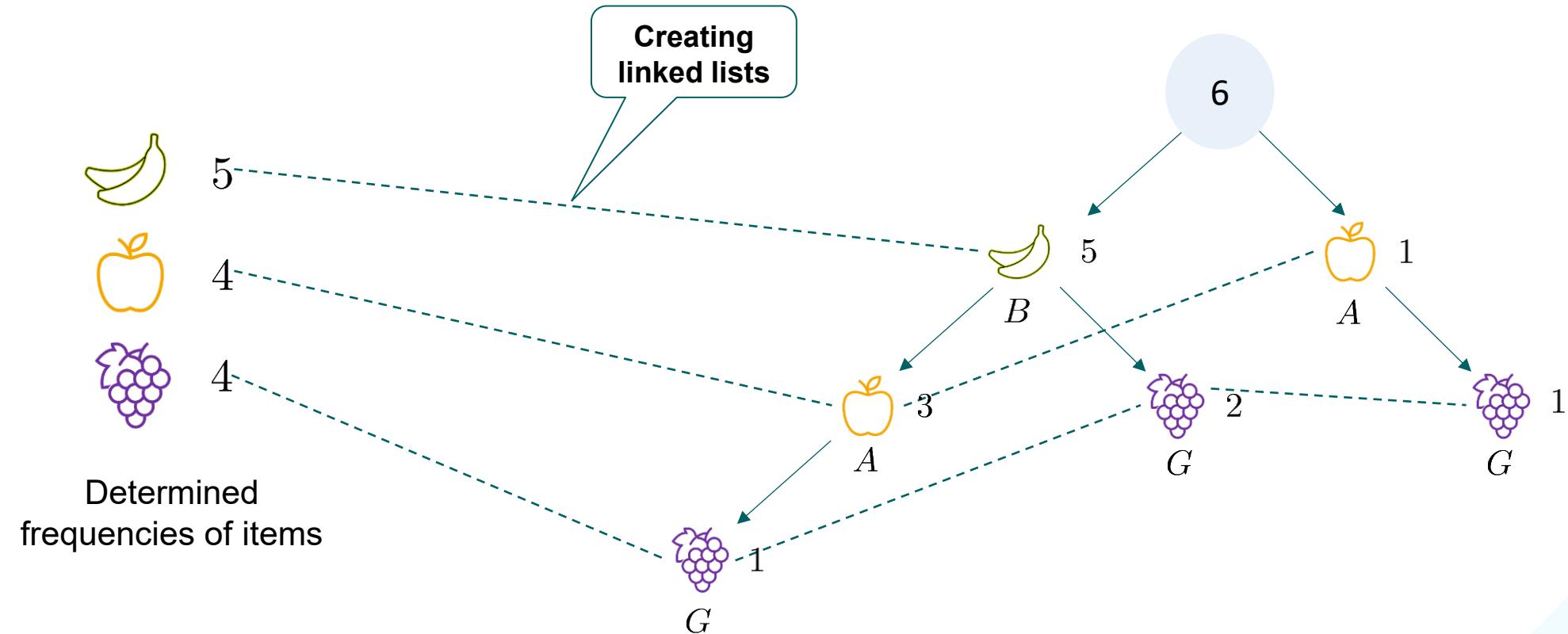
$$\mathcal{X} = [\{B, A, G\}^1, \{B, A\}^{50}, \{B, G\}^9, \{B\}^{29}, \{A, G\}^9, \{A\}^2]$$

Mining the FP-Tree – Overview

- For each frequent item, create a **conditional FP-tree** (starting with the **least** frequent one)
- The conditional FP-tree considers all transactions ending with this item
- Apply this **recursively**
- Due to recursion, we also consider postfixes that contain multiple elements
- The ordering ensures that postfixes are considered only once

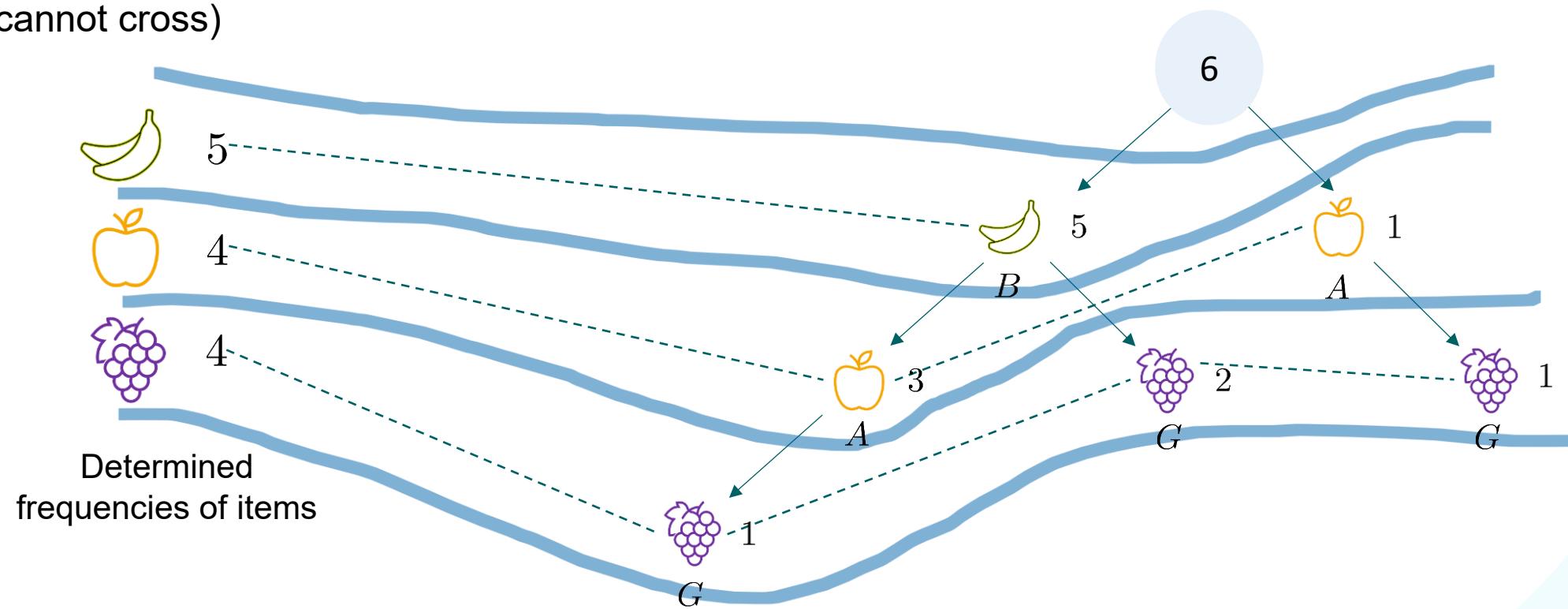


Node Links

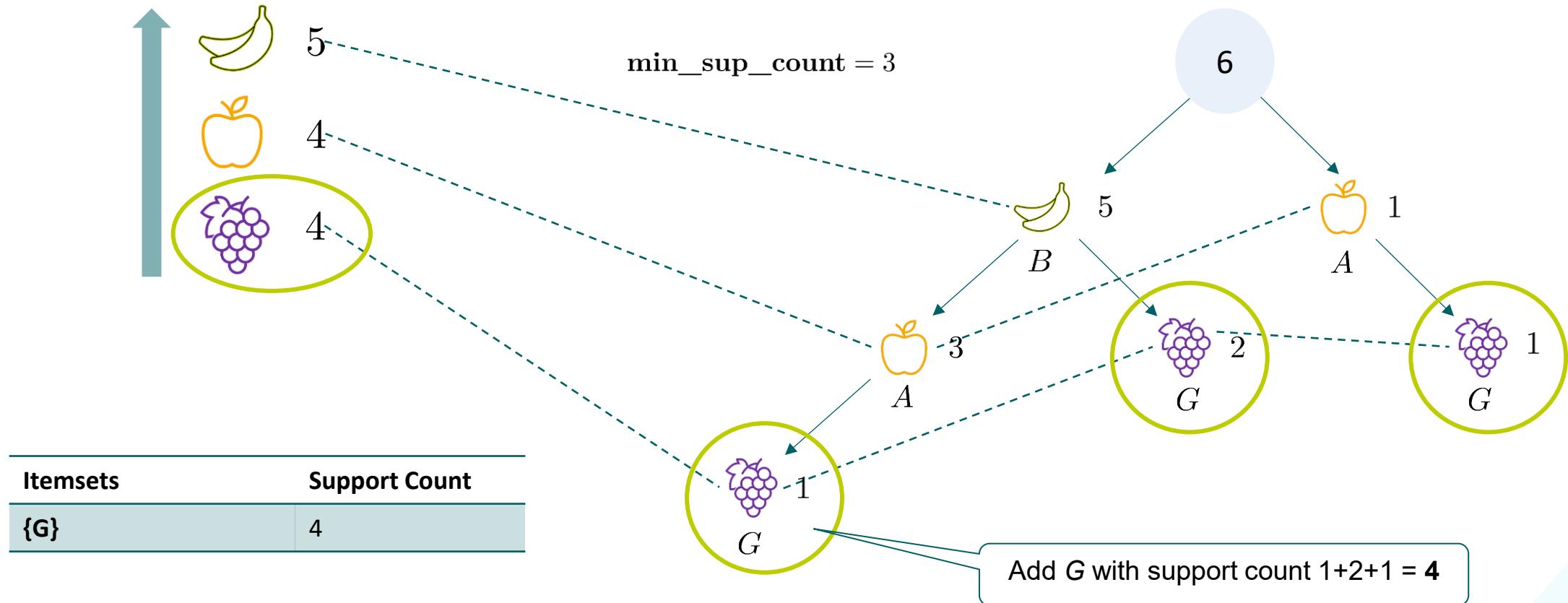


Node Links

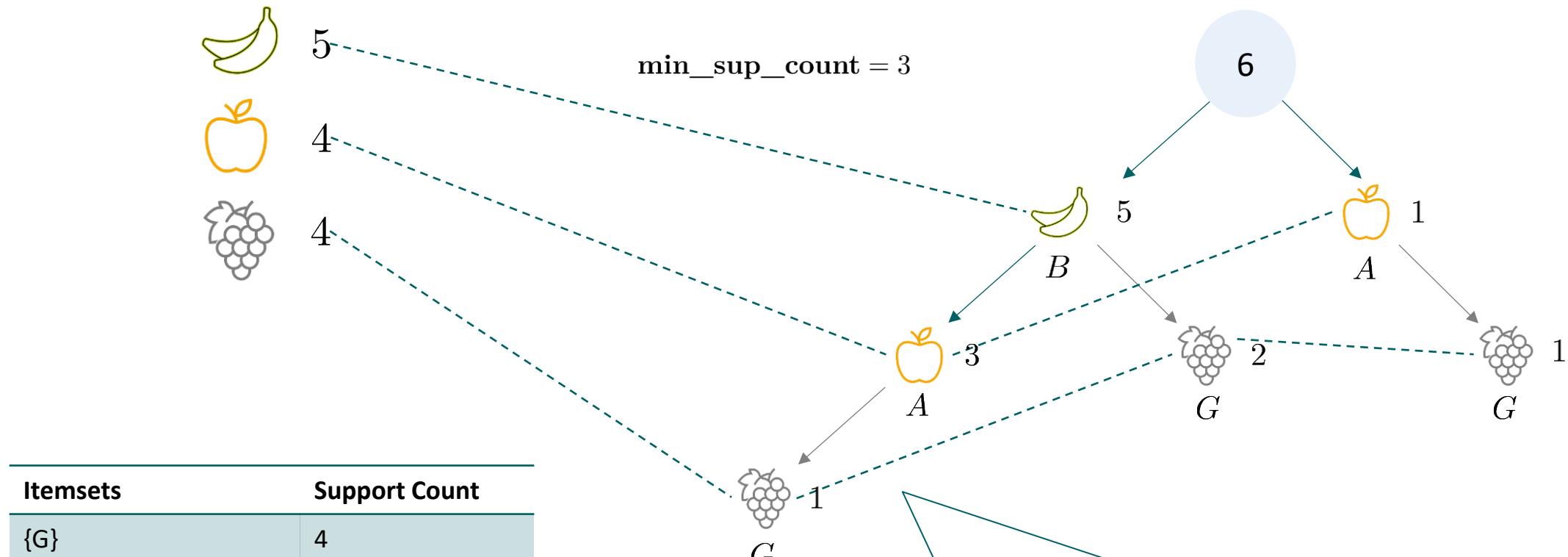
Node links are like 'altitude lines' because of total order of items
(they cannot cross)



Consider Postfix G

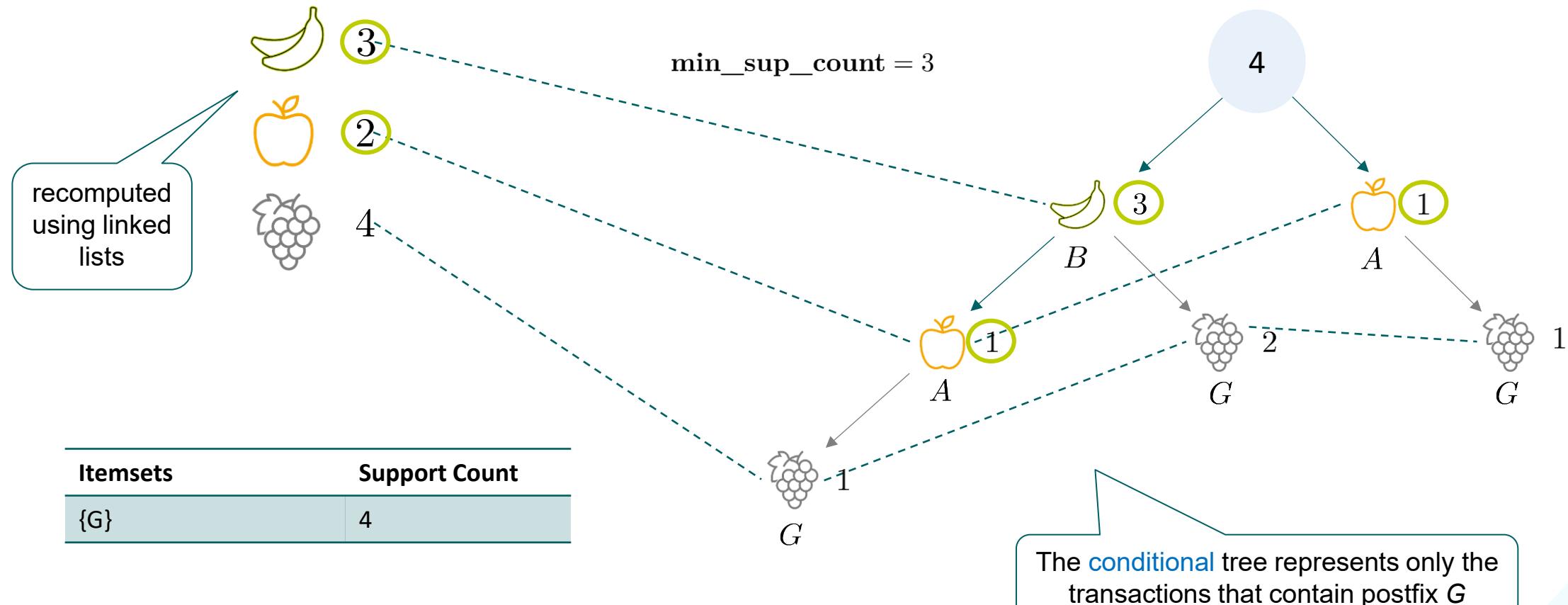


Consider Postfix G

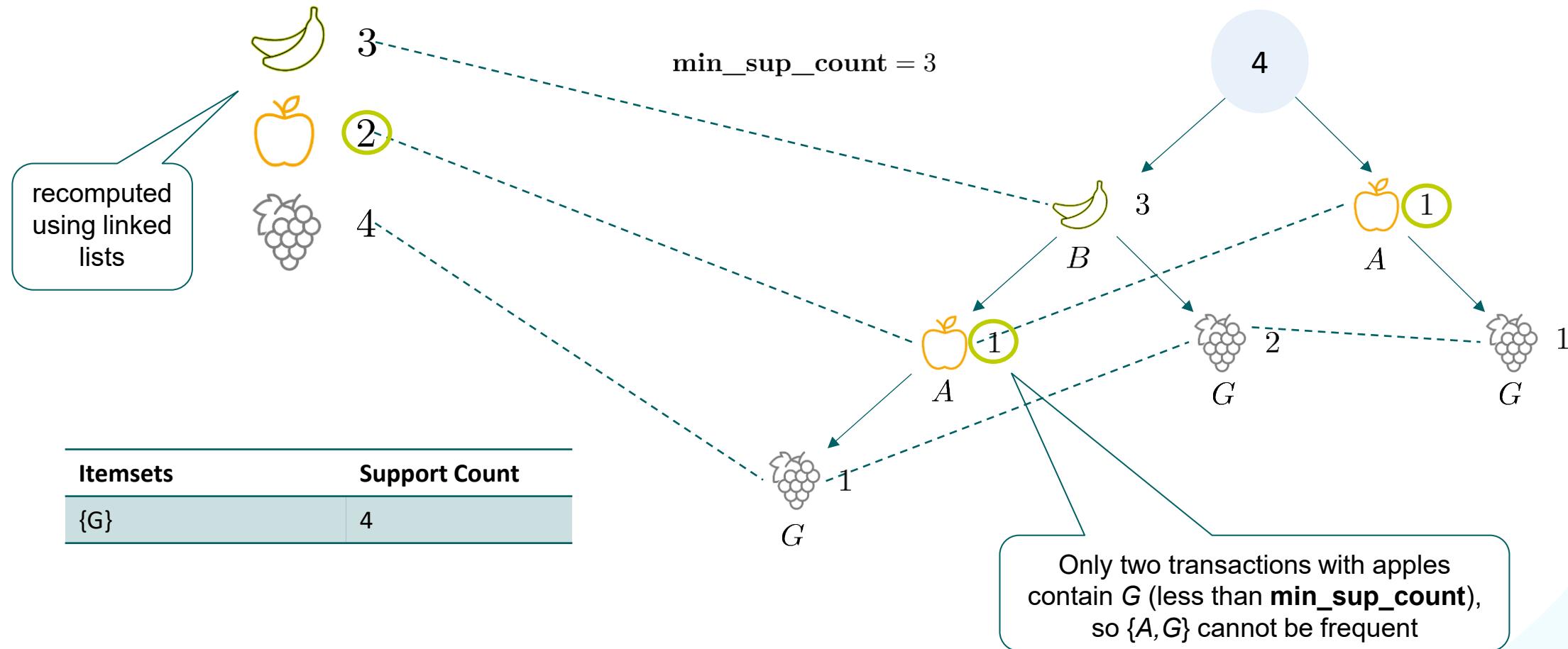


Now only consider **all the paths** leading from the root to G
(representing transactions that include G)

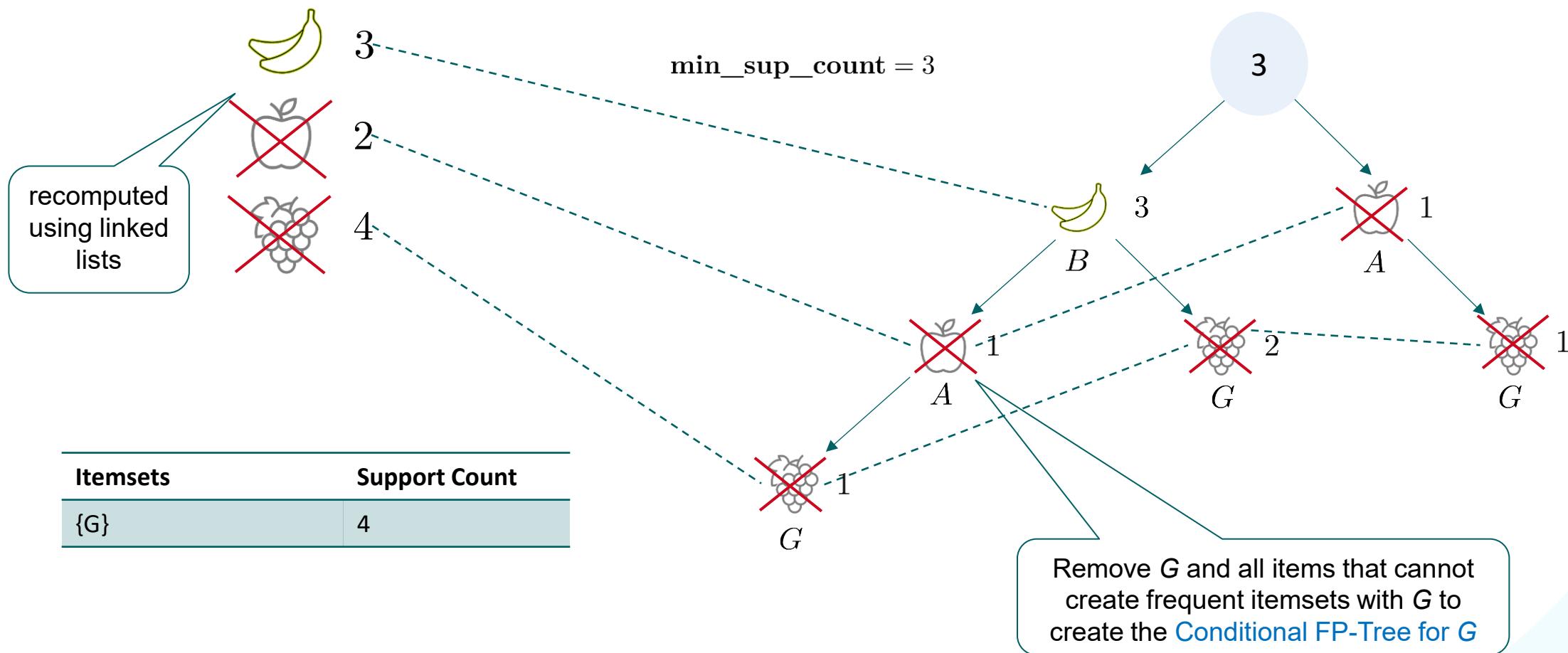
Towards the Conditional FP-Tree for Postfix G



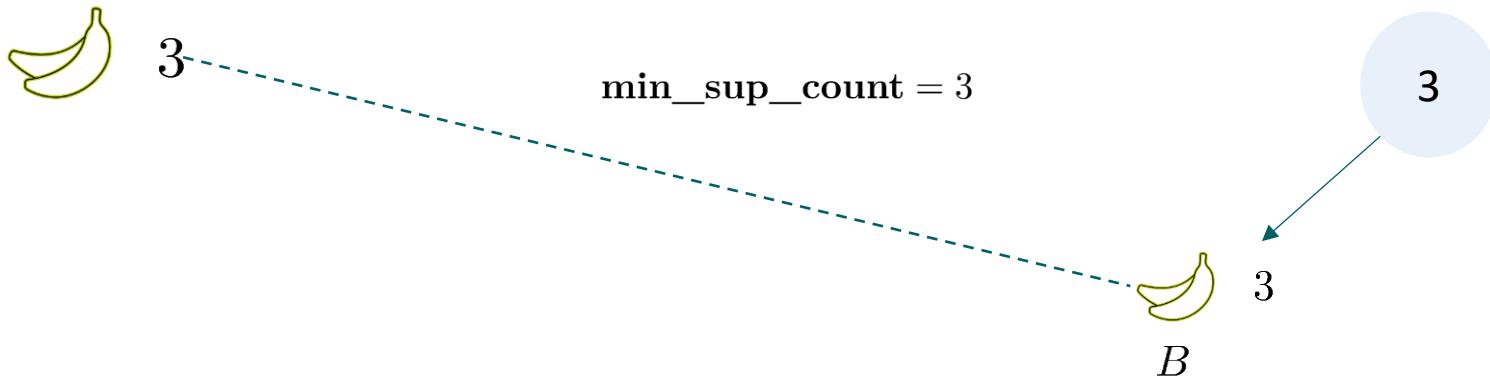
Towards the Conditional FP-Tree for Postfix G



Towards the Conditional FP-Tree for Postfix G



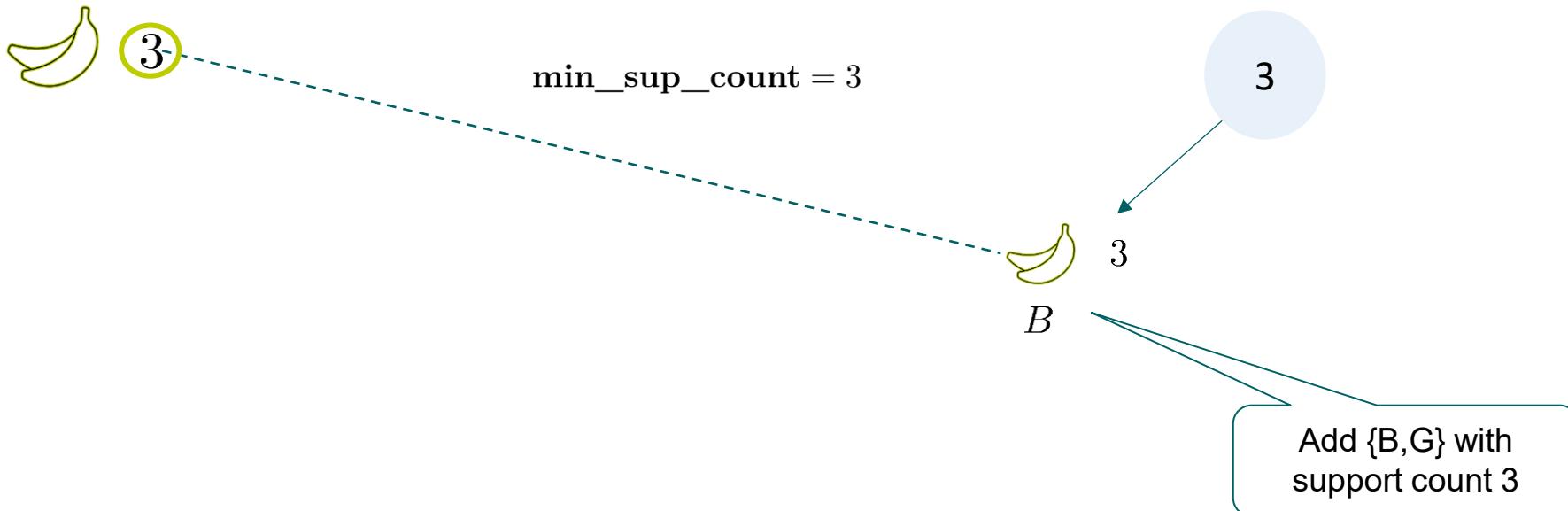
Conditional FP-Tree for Postfix G



| Itemsets | Support Count |
|----------|---------------|
| {G} | 4 |
| {..., G} | reurse |

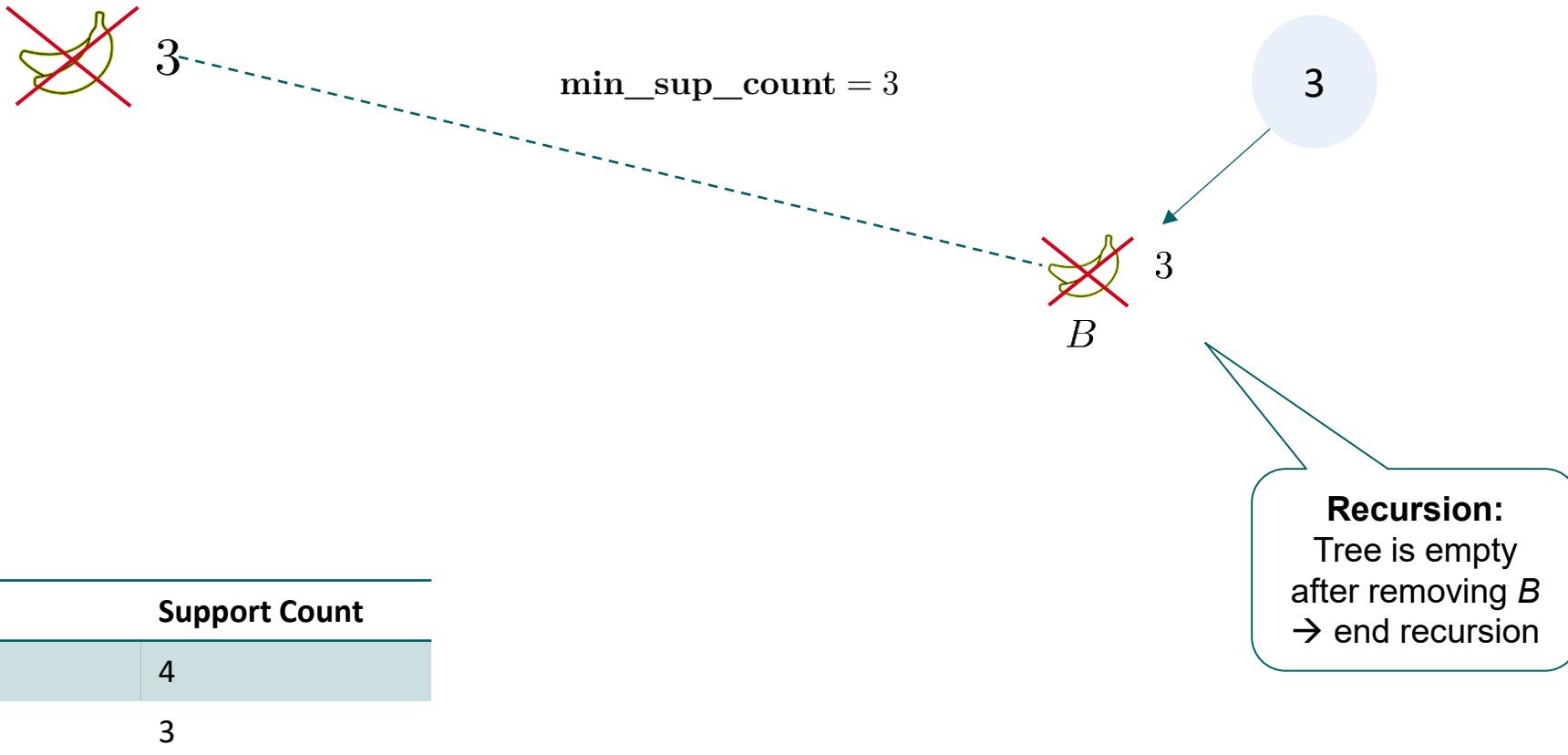
Mine the **Conditional FP-Tree for G** to
find frequent itemsets that contain G
(Recursion)

Conditional FP-Tree for Postfix G

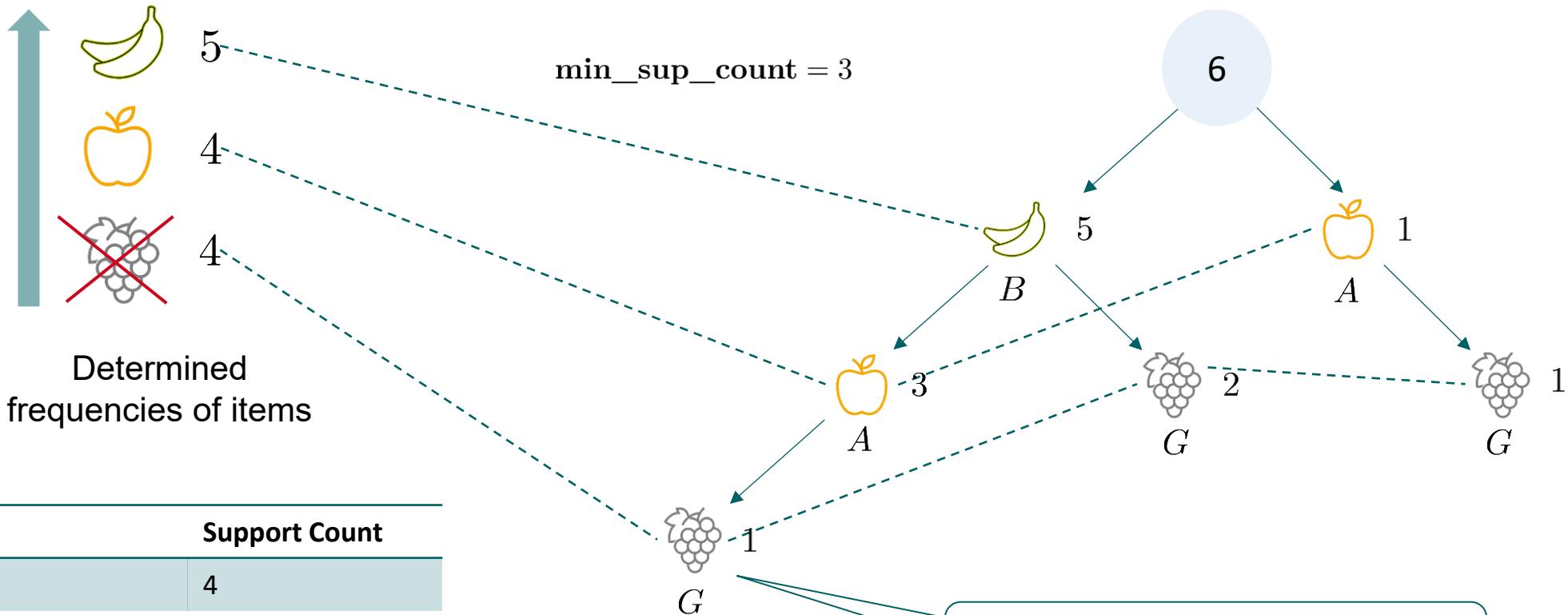


| Itemsets | Support Count |
|------------|---------------|
| $\{G\}$ | 4 |
| $\{B, G\}$ | 3 |

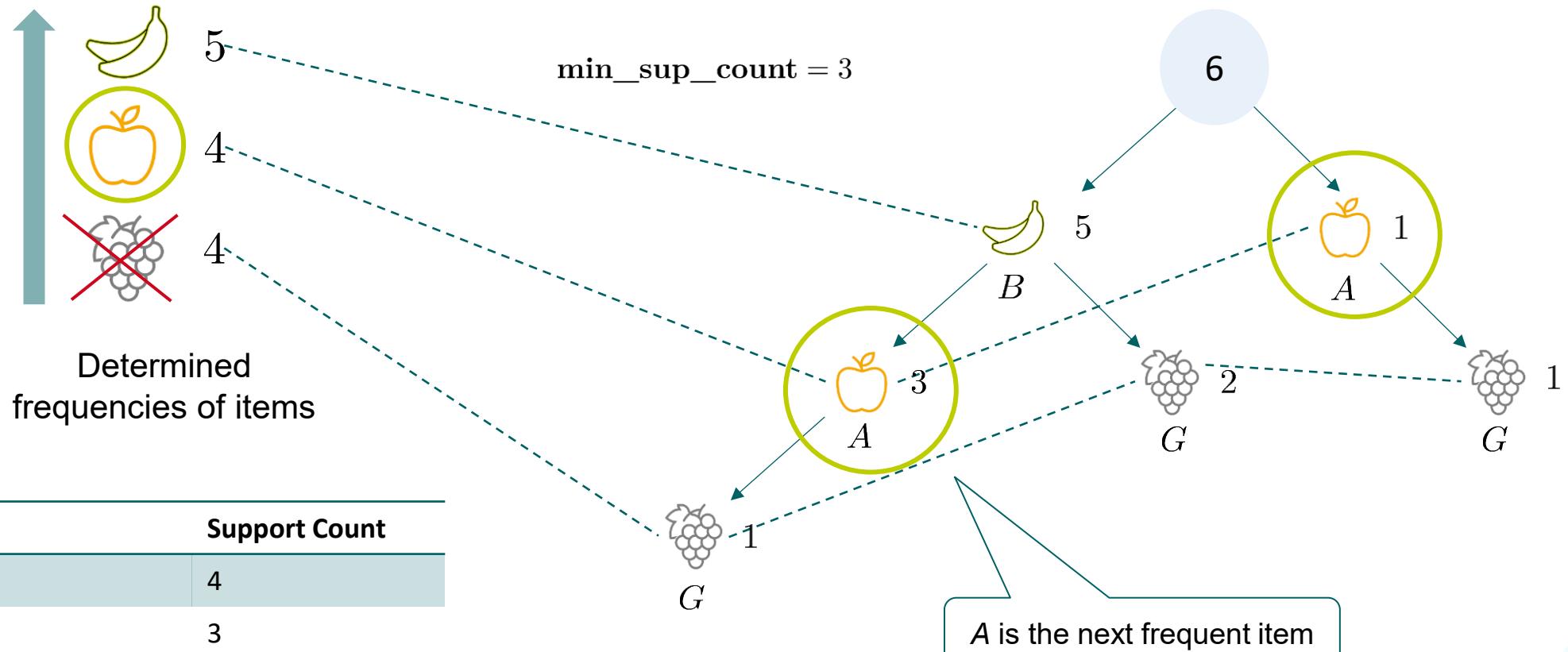
Conditional FP-Tree for Postfix G



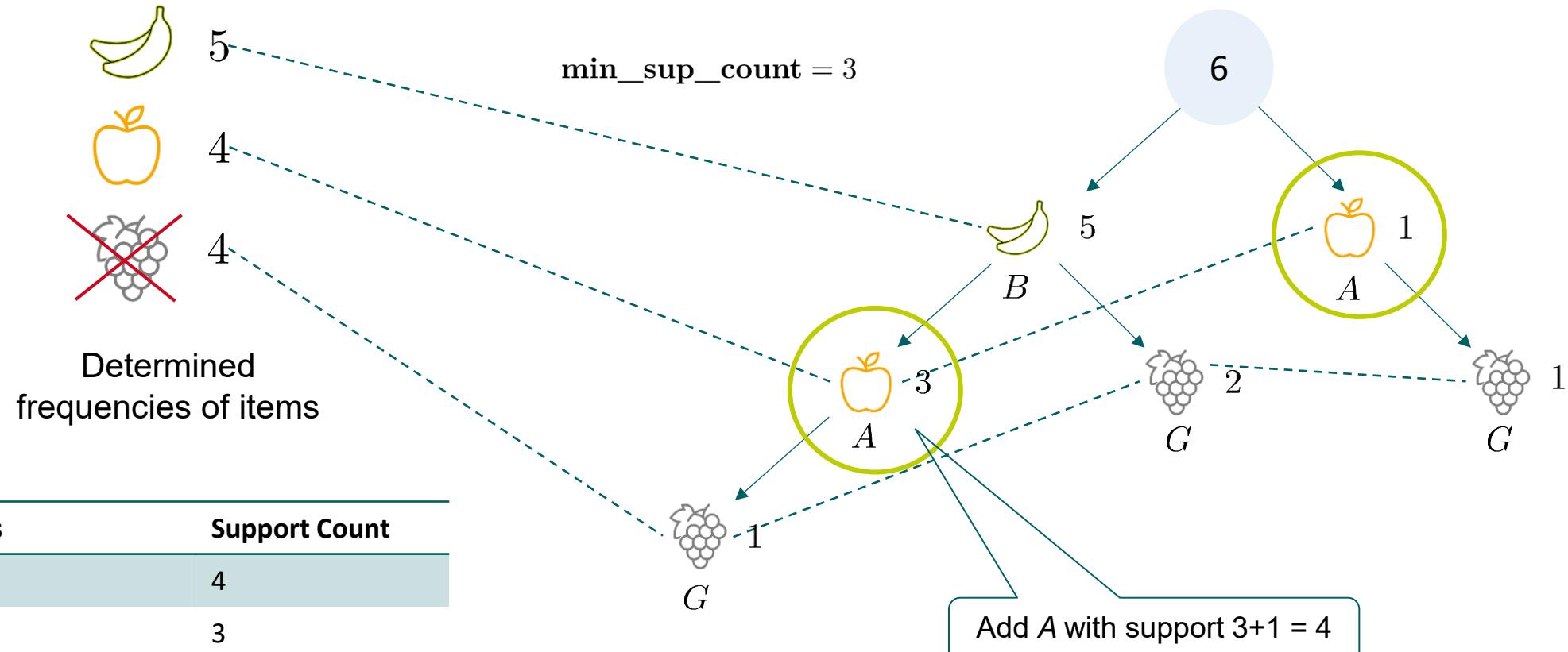
Consider Postfix A



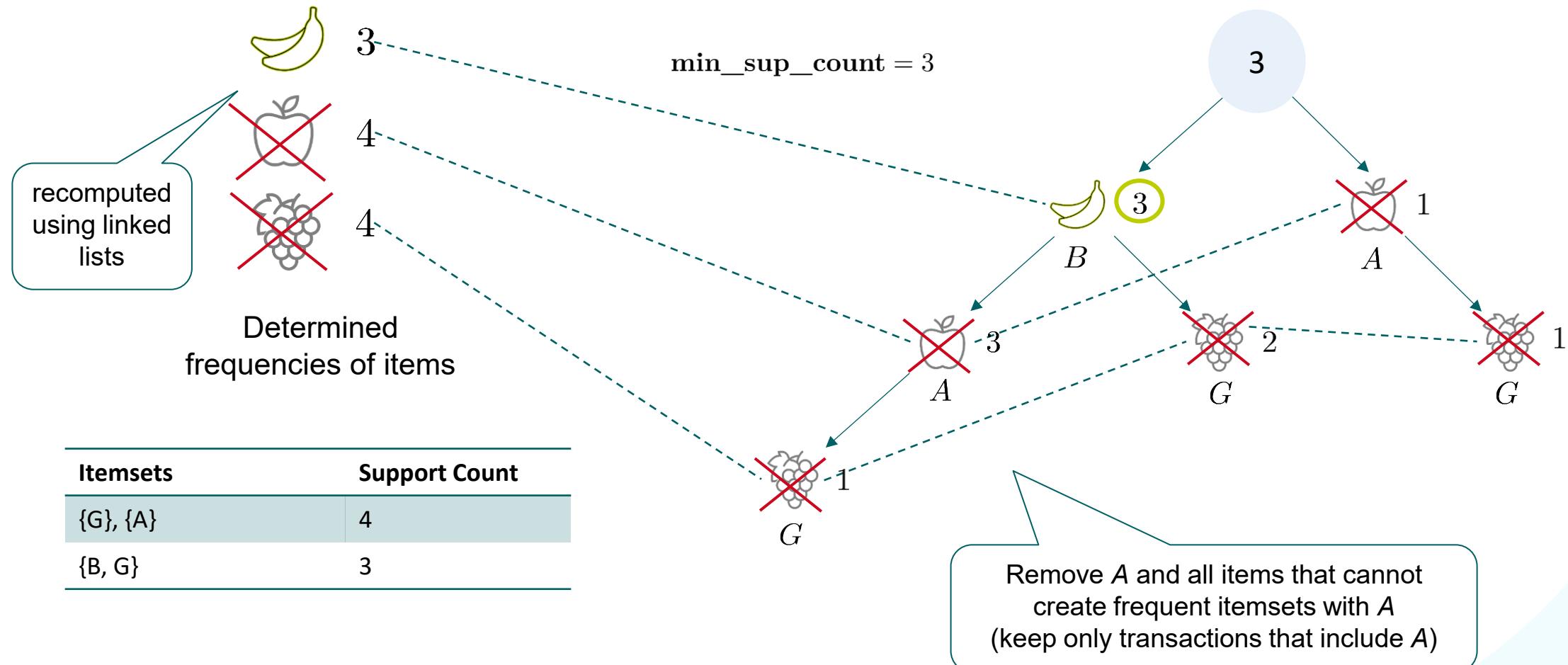
Consider Postfix A



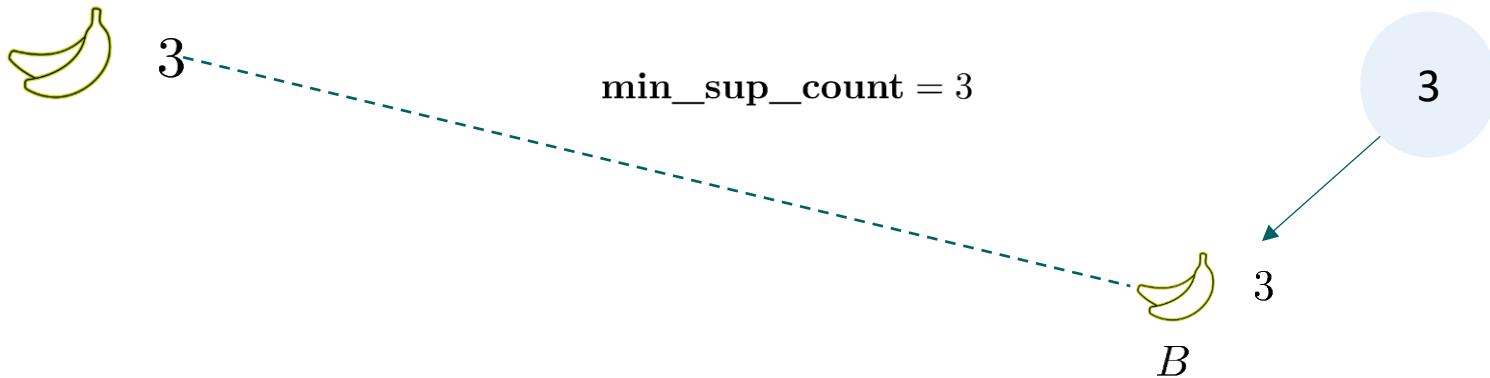
Consider Postfix A



Towards the Conditional FP-Tree for Postfix A



Towards Conditional FP-Tree for Postfix A



Determined
frequencies of items

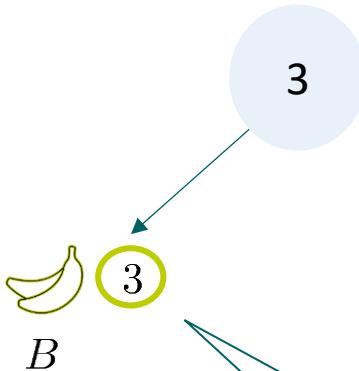
| Itemsets | Support Count |
|----------|---------------|
| {G}, {A} | 4 |
| {B, G} | 3 |

Conditional FP-Tree for Postfix A



3

min_sup_count = 3

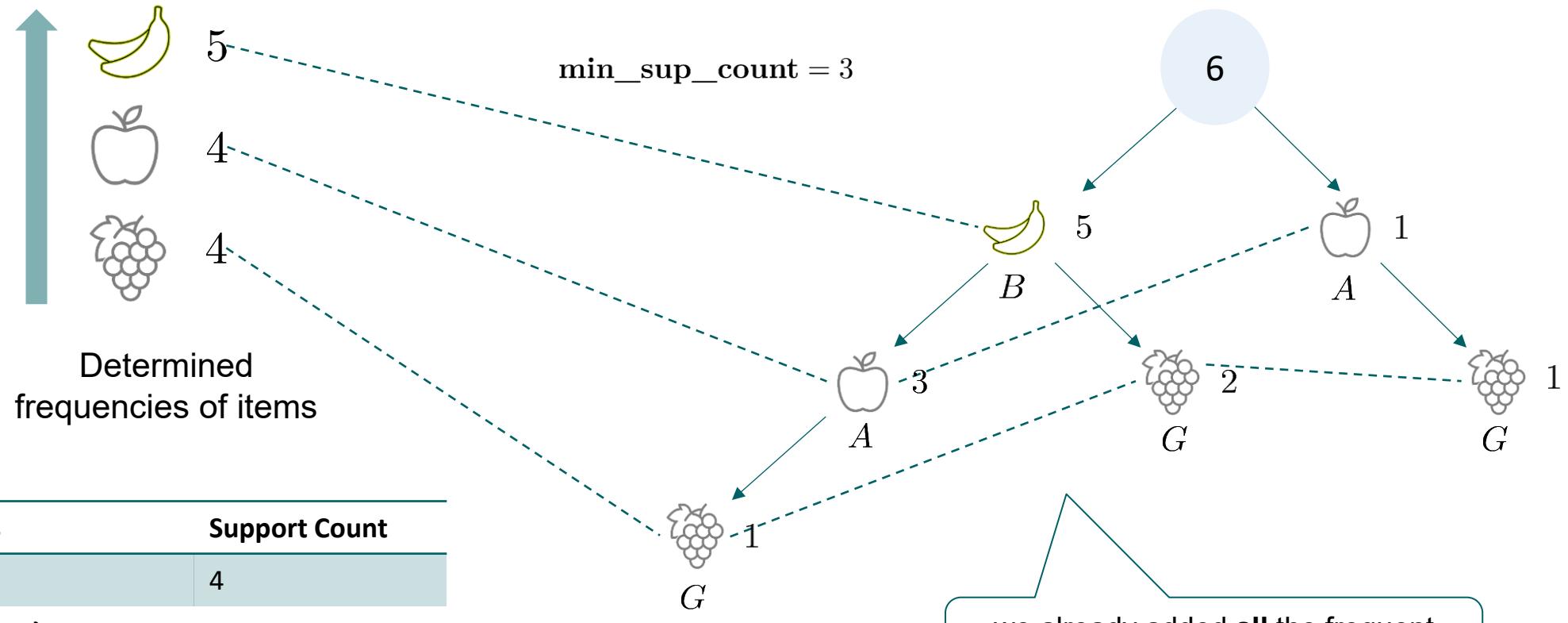


Determined
frequencies of items

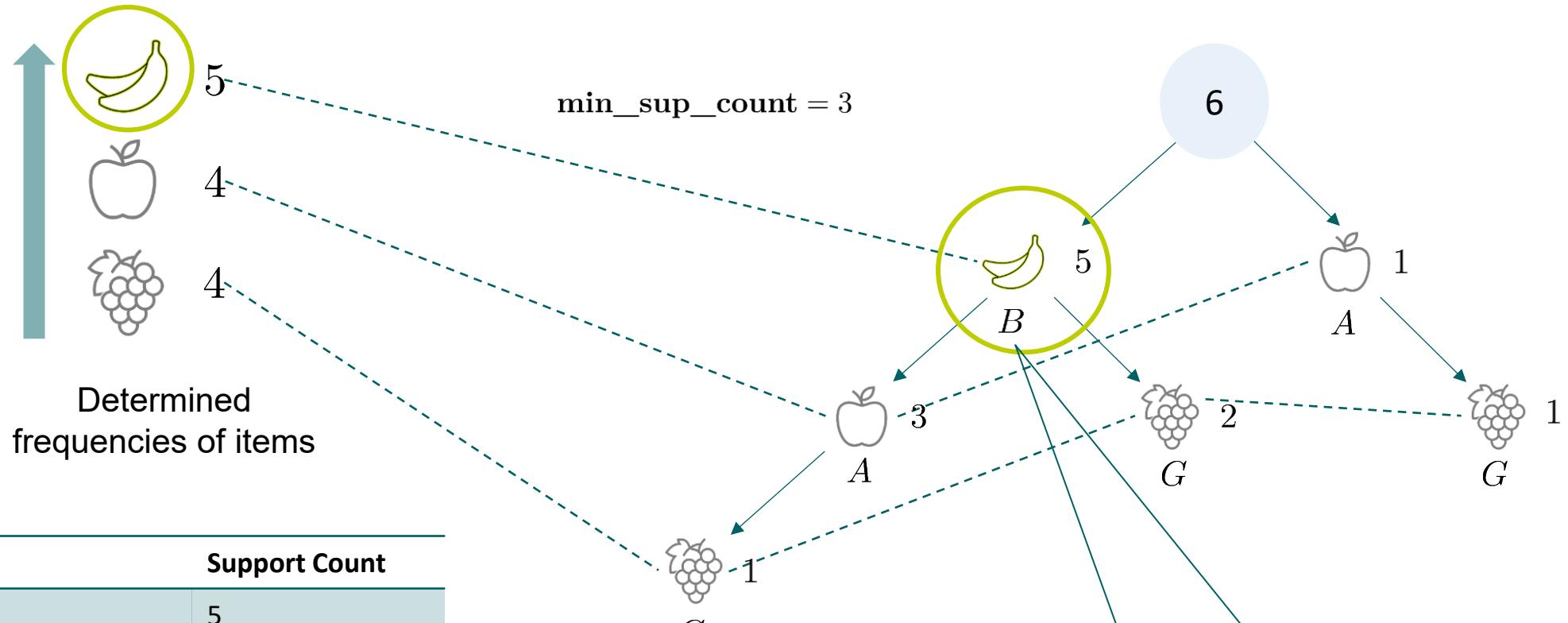
| Itemsets | Support Count |
|----------------|---------------|
| {G}, {A} | 4 |
| {B, G}, {B, A} | 3 |

Recursion:
Add {B, A} with
support 3 (then end
recursion)

Consider Postfix B

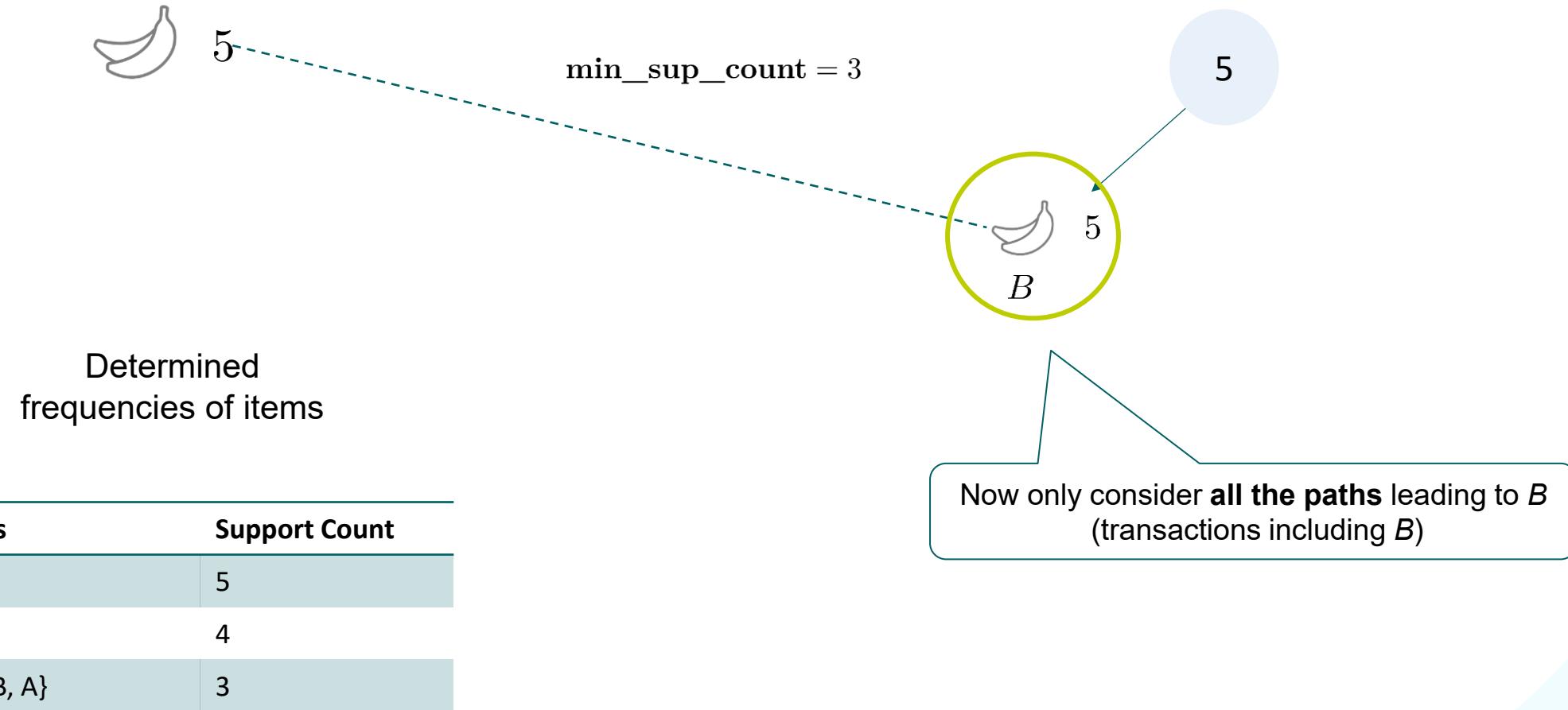


Consider Postfix B



| Itemsets | Support Count |
|----------------------|---------------|
| $\{B\}$ | 5 |
| $\{G\}, \{A\}$ | 4 |
| $\{B, G\}, \{B, A\}$ | 3 |

Towards Conditional FP-Tree for Postfix B

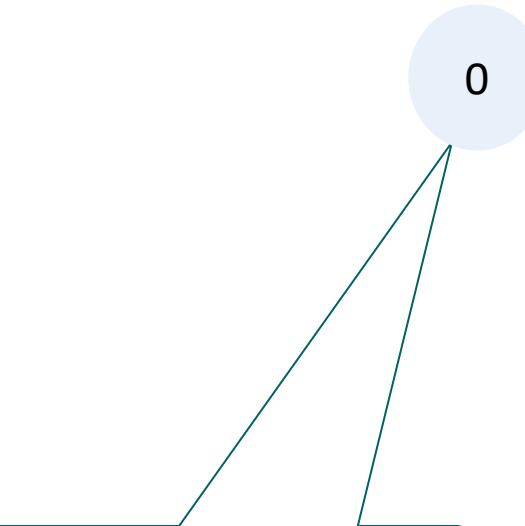


Conditional FP-Tree for Postfix B

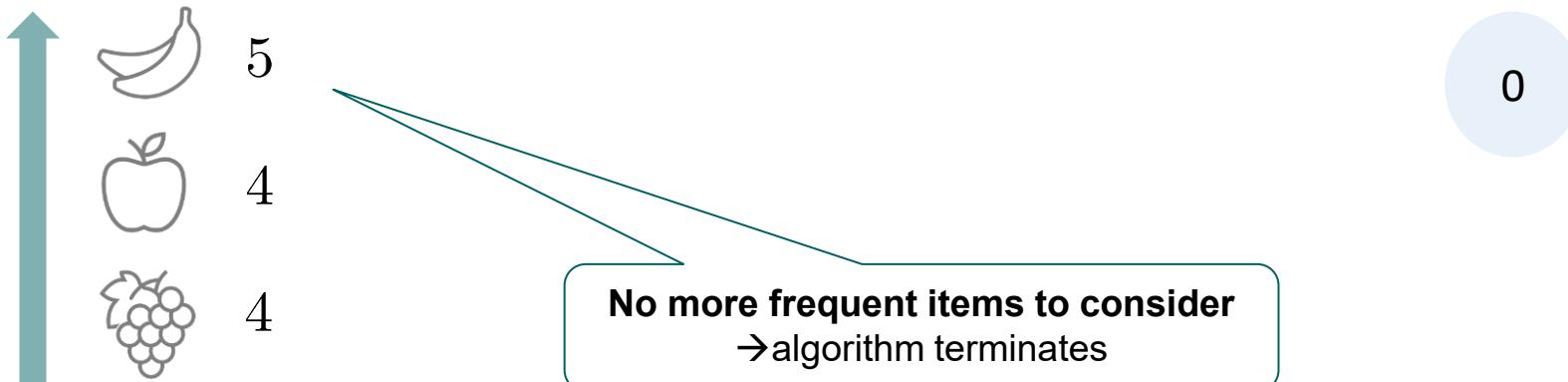


| Itemsets | Support Count |
|----------------|---------------|
| {B} | 5 |
| {G}, {A} | 4 |
| {B, G}, {B, A} | 3 |

Conditional FP-tree is empty, i.e., no additional frequent itemsets with B

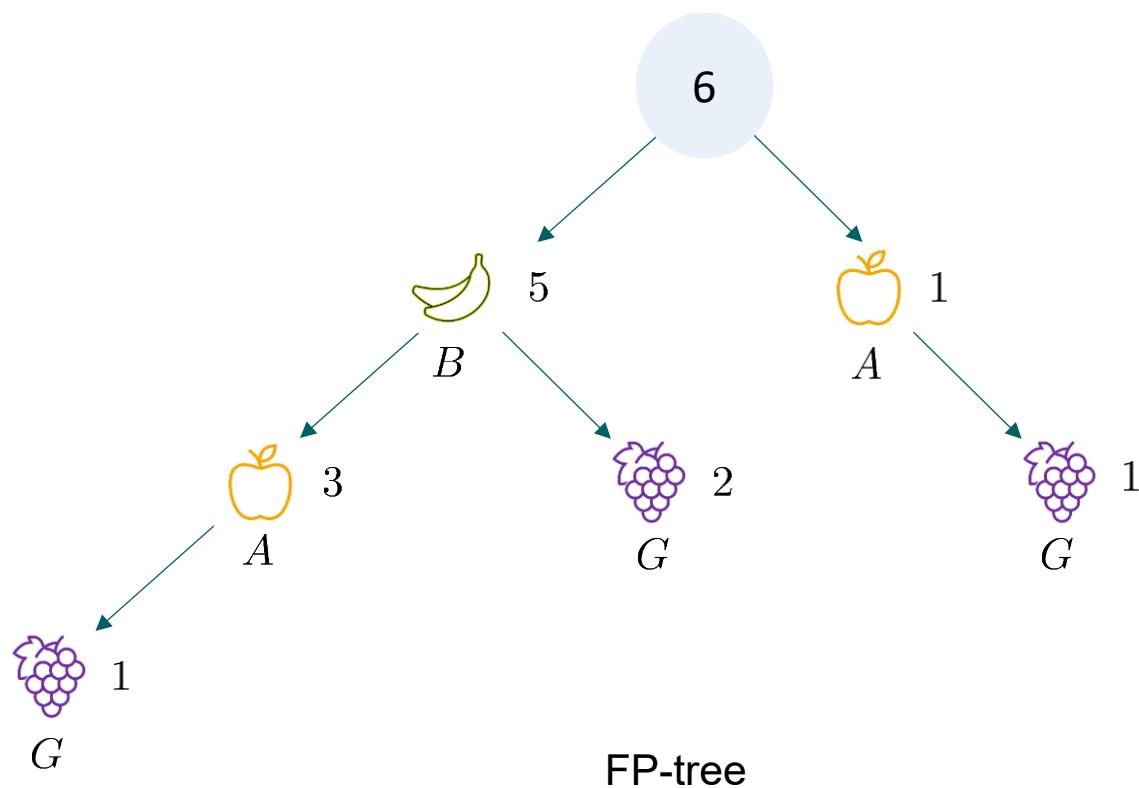


Conditional FP-Tree for Postfix B



| Itemsets | Support Count |
|----------------|---------------|
| {B} | 5 |
| {G}, {A} | 4 |
| {B, G}, {B, A} | 3 |

All Frequent Itemsets Generated



| Itemsets | Support Count |
|----------------|---------------|
| {B} | 5 |
| {G}, {A} | 4 |
| {B, G}, {B, A} | 3 |

Frequent itemsets mined

FP-Growth Algorithm – General Principle

- The example did not show all possible cases.
- Consider an item set $\mathcal{A} = \{I_{k_1}, I_{k_2}, \dots, I_{k_n}\}$ (sorted based on the first pass).
- The conditional FP-tree for \mathcal{A} is **identical to** the FP-tree created for a dataset where:
 - First, all transactions not containing all elements of \mathcal{A} are removed (remove rows)
 - Then, all items in the set $\mathcal{I}_R = \{I_j \mid j \geq k_1\}$ are removed (remove columns).
- The conditional FP-tree can be used to compute all frequent item sets “ending” with postfix \mathcal{A} .
- These are all frequent item sets \mathcal{B} such that $\mathcal{B} \cap \mathcal{I}_R = \mathcal{A}$.

FP-Growth Algorithm – Summary

- Idea: frequent pattern growth based on FP-tree
- Method:
 - Construct the FP-tree from the dataset
 - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
 - Recursively repeat the process on each newly created conditional FP-tree until the tree is empty
- Advantages:
 - ✓ Only two passes through the dataset are needed (when constructing the tree)
 - ✓ Avoiding testing many hopeless candidates
 - ✓ Very fast when FP-tree fits in main memory

Frequent Itemsets – Summary

- Pattern mining is a form of unsupervised learning
- Frequent itemsets are the basis for finding patterns (ideas can be transferred to other patterns)
- Two well-known algorithms using generally applicable concepts:
 - Apriori algorithm
 - FP-growth algorithm
- Outlook
 - There may be many frequent “patterns”
 - How to determine which ones are surprising / interesting?

Association Rules – Preview

(one of the topics of the next lecture)



$\{Cheese, Bread\} \Rightarrow \{Milk\}$

People that buy Cheese and Bread also tend to buy Milk.



$\{AC/DC, Queen\} \Rightarrow \{Metallica\}$

Users that listen to AC/DC and Queen also tend to listen to Metallica.



$\{Bitburger\} \Rightarrow \{Heineken, Palm\}$

People that buy Bitburger beer tend to buy both Heineken and Palm beer.



$\{Carbonara, Margherita\} \Rightarrow \{Espresso, Tiramisu\}$

People that buy Carbonara and Margherita also tend to buy Espresso and Tiramisu.



$\{part-245, part-345, part-456\} \Rightarrow \{part-372\}$

When Parts 245, 345, and 456 are replaced, then often also Part 372 is replaced.