

Elements of Machine Learning & Data Science

Winter semester 2025/26

Lecture 5 – Frequent Itemsets

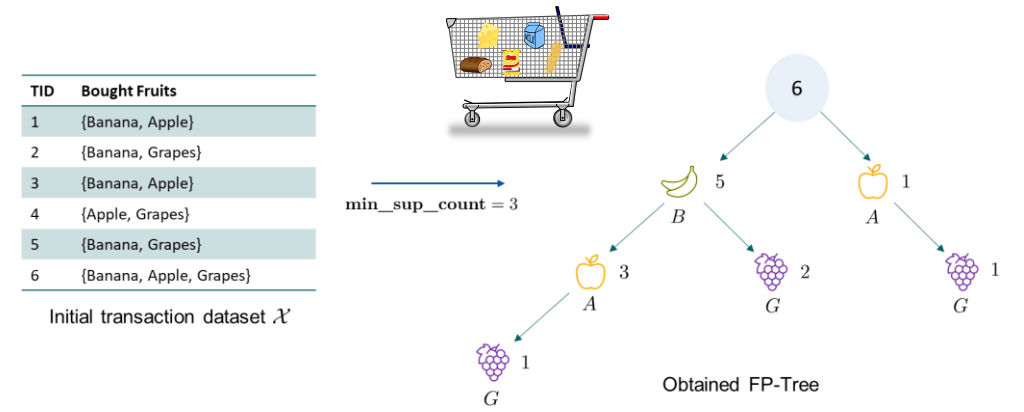
04.11.2025

Prof. Bastian Leibe

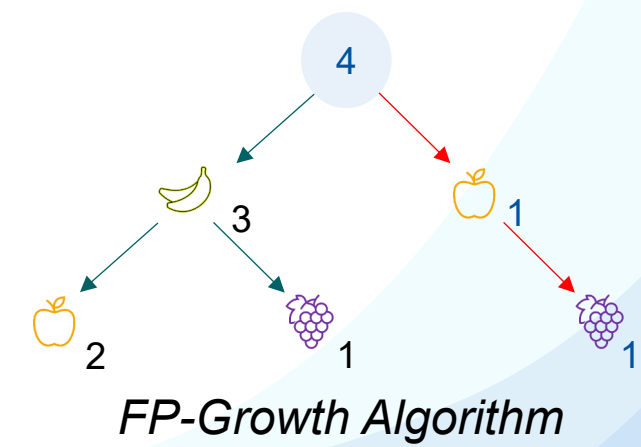
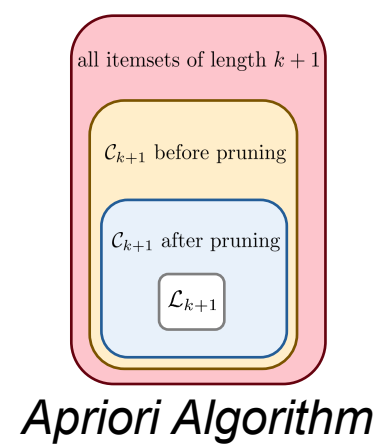
slides by Prof. Wil van der Aalst

Overview of the Lecture Topics

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

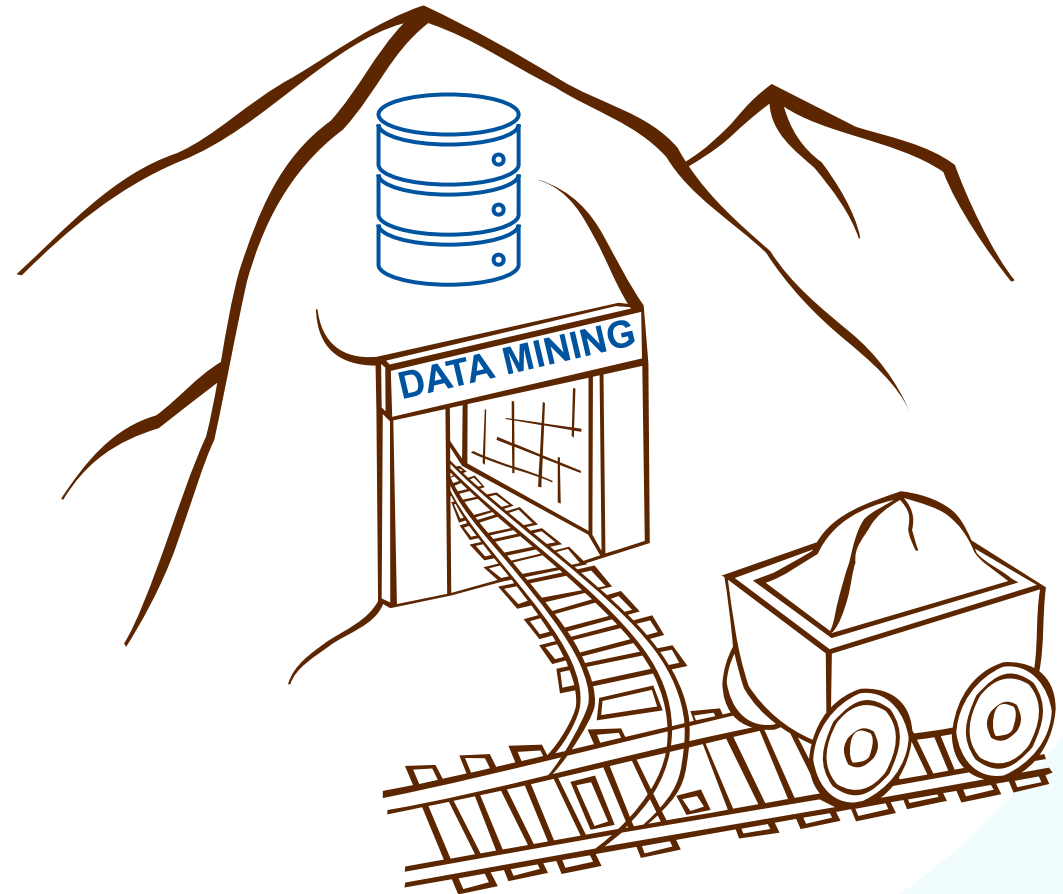


Frequent Itemsets



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**


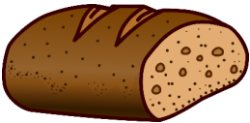



Each cell describes the **presence of the item** (Boolean 0/1 or a natural number)

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


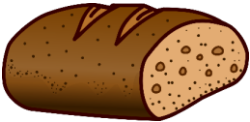



Itemset Data – Example



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
...





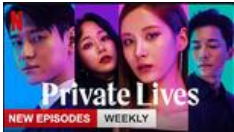
Itemset Data – Example

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	 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	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	 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	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 $\mathcal{T}_1 = [Che^2, Mil^3, Pas^2] \in \mathbb{M}(\mathcal{I}) = \mathcal{I} \rightarrow \mathbb{N}$
- 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 *support*(A)
 - *min_sup_count*: lower bound for *support_count*(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	 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 $\mathcal{X} = [\{Che, Mil, Pas\}, \{Chi, Mil\}, \{Che, Bre\}, \{Che, Bre, Mil\}]$


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

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

$$\text{support_count}(\mathcal{A}) = |\{T \in \mathcal{X} \mid \mathcal{A} \subseteq T\}| = |\{T_1, T_4\}| = 2$$

$$\text{support}(\mathcal{A}) = \frac{|\{T \in \mathcal{X} \mid \mathcal{A} \subseteq T\}|}{|\mathcal{X}|} = \frac{|\{T_1, T_4\}|}{4} = \frac{2}{4}$$

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






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

$$\text{support_count}(\mathcal{B}) = |\{T \in \mathcal{X} \mid \mathcal{B} \subseteq T\}| = |\{T_1, T_2, T_4\}| = 3$$

$$\text{support}(\mathcal{B}) = \frac{|\{T \in \mathcal{X} \mid \mathcal{B} \subseteq T\}|}{|\mathcal{X}|} = \frac{|\{T_1, T_2, T_4\}|}{4} = \frac{3}{4}$$

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

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)

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

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

$$\text{support}(\mathcal{A}) = \frac{|\{T \in \mathcal{X} \mid \mathcal{A} \subseteq T\}|}{|\mathcal{X}|} = \frac{|\{T_1, T_4\}|}{4} = \frac{2}{4}$$

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

$$\text{support}(\mathcal{B}) = \frac{|\{T \in \mathcal{X} \mid \mathcal{B} \subseteq T\}|}{|\mathcal{X}|} = \frac{|\{T_1, T_2, T_4\}|}{4} = \frac{3}{4}$$

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

General rule:

Subset of an itemset has higher or equal support than this itemset

Support – Summary

Support

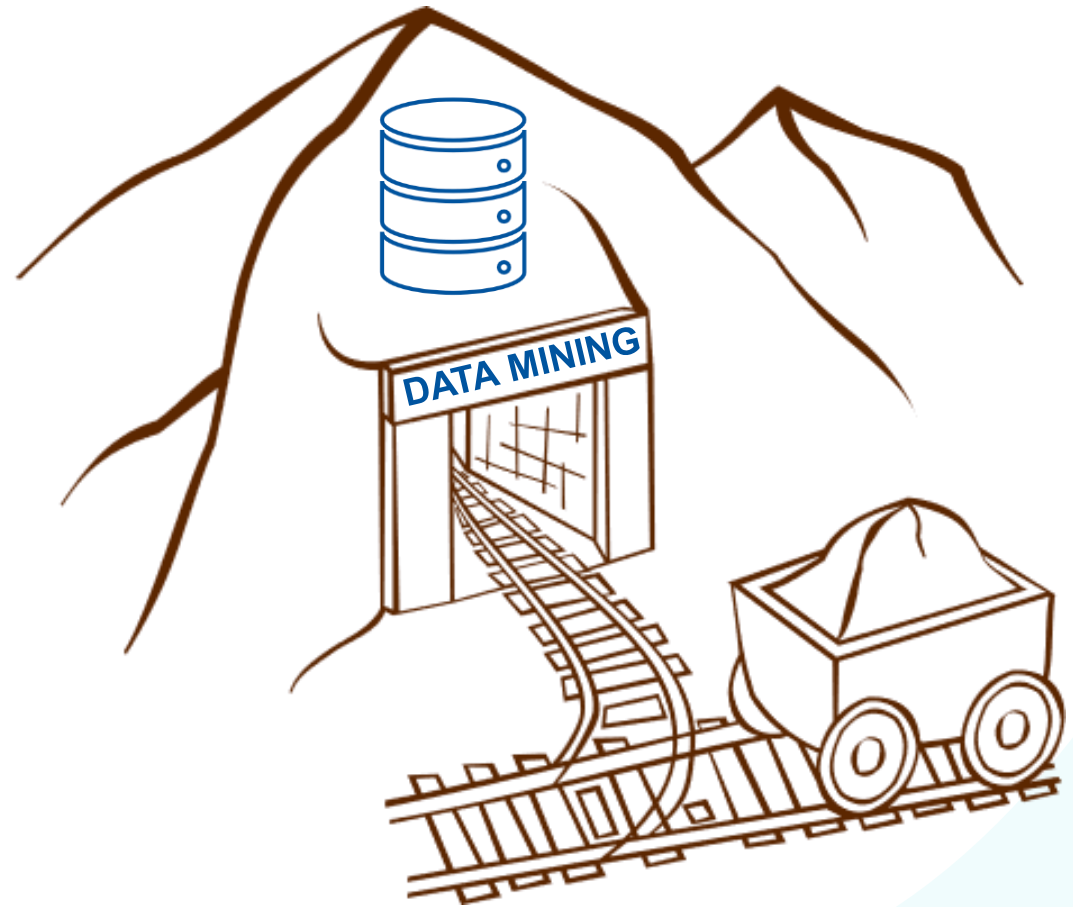
- A measure of the popularity (frequency) of an itemset.
- Calculated as the fraction of transactions in a dataset that contain the itemset.

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

- 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 $\text{support}(\mathcal{A}) \geq \text{min_sup}$ 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



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

$2^D - 1 =$

3,160,699,436,856,317,896,133,924,659,945,691,788,984,676,387,834,935,666,847,743,155,564,943,937,902,009,550,651,067,144,922,529,420,974,282,690,343,798,061,622,891,650,247,060,091,533,595,130,170,365,868,108,099,970,116,531,087,467,047,583,722,093,787,639,674,649,765,662,074,366,466,883,324,927,932,743,926,222,226,256,325,646,619,479,597,070,853,065,410,126,319,556,64,5,095,487,584,255,731,625,229,939,513,738,335,892,649,026,005,867,435,951,184,963,615,454,162,198,009,836,540,553,902,746,189,836,926,616,803,054,602,986,713,771,300,764,236,103,912,584,316,984,387,046,421,891,044,904,862,409,285,760,387,607,642,178,660,447,084,278,582,343,741,975,788,782,575,397,255,679,835,385,509,255,617,699,530,378,773,285,612,149,746,612,966,192,65,1,253,693,114,313,273,858,994,605,409,771,445,163,688,490,715,356,137,720,041,811,636,018,280,718,733,780,759,203,810,695,576,005,864,810,237,900,789,639,882,550,370,233,732,760,551,157,423,186,221,777,437,962,235,113,685,057,004,875,812,625,837,741,639,074,014,635,869,254,467,870,669,671,658,811,749,944,572,950,947,120,522,955,828,413,262,134,950,123,343,450,944,893,38,1,086,445,704,327,516,608,478,853,295,735,214,794,279,795,422,828,861,127,478,241,396,081,419,738,759,639,852,247,634,596,698,774,287,547,984,270,270,829,223,110,217,158,518,692,621,481,994,384,293,987,162,186,350,943,113,710,209,892,772,971,716,110,929,927,731,636,487,052,992,987,749,952,582,075,199,999,823,623,336,378,201,206,536,182,409,135,673,260,858,434,397,680,32,4,294,431,695,532,313,988,094,705,825,127,170,281,671,342,756,132,130,834,630,626,735,161,518,877,001,479,353,099,675,421,568,669,161,387,971,545,385,350,702,911,535,042,214,105,270,064,184,049,195,321,693,402,645,502,131,017,185,839,794,236,528,392,026,118,916,801,534,722,139,870,222,746,741,395,902,492,128,586,493,084,122,499,921,428,423,167,659,386,991,199,053,145,4,0,232,918,025,337,294,792,121,480,817,320,527,444,814,166,560,131,328,380,547,089,162,451,960,863,019,896,841,277,608,267,473,071,032,610,361,884,348,741,832,609,100,033,764,267,372,951,754,668,207,124,976,344,881,559,366,912,679,239,124,697,823,644,196,584,984,332,119,466,217,601,083,183,663,311,742,410,337,555,621,104,734,292,489,903,235,192,160,821,330,89,8,029,535,914,395,120,861,478,951,172,55,966,695,019,256,259,017,910,724,870,171,631,683,422,158,829,652,626,998,675,935,495,968,355,706,034,794,377,249,704,862,237,385,428,041,477,614,977,159,200,366,168,819,484,882,369,473,577,673,620,234,304,059,320,536,368,181,780,663,852,770,156,176,045,997,023,854,188,210,931,505,764,603,497,418,111,16,833,045,59,4,683,460,761,810,060,347,938,666,107,376,407,376,581,61,433,794,082,201,152,452,755,038,680,915,248,667,576,037,816,448,107,114,928,454,957,358,973,769,979,011,775,081,460,363,858,425,659,955,551,853,002,945,198,136,847,407,803,098,731,016,476,002,188,721,167,044,725,016,609,620,117,517,264,461,388,623,721,099,785,827,561,842,233,211,027,265,707,652,869,227,549,971,36,0,421,138,832,935,337,610,829,930,530,940,245,818,717,397,877,474,556,856,115,648,381,712,850,949,057,429,834,288,786,930,238,199,668,462,420,124,975,014,260,176,937,57,543,706,397,088,013,809,808,763,009,055,350,430,278,490,509,069,412,005,546,729,309,455,095,193,839,024,869,890,246,164,588,776,426,256,822,325,723,140,640,821,450,894,163,789,455,395,044,742,35,8,602,248,211,119,402,554,274,364,607,815,242,974,440,790,954,476,462,235,421,343,042,061,888,188,431,752,832,527,921,982,019,539,829,676,482,175,124,356,602,558,024,358,281,709,710,521,718,828,251,944,407,816,772,714,544,977,673,460,983,558,935,168,237,819,654,683,418,376,457,904,080,539,202,530,627,669,869,881,215,398,947,957,922,545,005,739,842,763,519,814,852,704,76,5,560,828,463,466,036,149,792,281,182,326,943,206,886,419,926,850,135,063,162,179,241,871,274,796,024,802,742,686,225,024,907,605,632,447,471,332,642,008,409,940,104,913,983,385,874,283,001,746,786,647,879,747,994,089,213,330,239,015,474,018,013,177,685,271,043,254,385,416,804,411,944,936,017,477,858,365,910,618,800,586,138,894,786,098,262,095,903,664,034,479,708,143,59,1,509,590,289,319,757,152,692,646,380,447,474,820,059,221,354,059,445,093,575,442,716,973,331,098,176,716,707,848,689,462,376,523,334,556,192,193,952,316,116,218,228,442,266,069,078,290,033,668,168,882,789,937,930,106,606,528,021,679,497,563,218,701,433,041,385,362,553,384,155,785,144,110,504,431,662,251,892,992,899,207,988,896,988,077,055,532,889,768,442,011,243,43,6,599,859,452,840,889,686,495,279,989,903,477,096,078,575,541,951,328,806,322,851,950,252,552,592,344,168,743,316,523,924,091,677,756,555,035,300,546,896,564,711,246,599,096,501,538,015,732,460,568,505,159,309,026,876,942,822,257,942,805,338,777,947,949,083,385,366,853,887,931,563,547,342,972,750,038,735,441,012,362,088,807,826,002,678,323,209,075,341,664,682,039,970,24,4,371,752,439,627,477,589,922,704,036,287,386,433,079,400,082,357,329,563,904,616,267,987,268,587,201,847,267,650,982,070,274,204,118,716,145,673,577,347,087,894,405,829,670,915,892,444,598,008,311,679,006,927,556,585,863,235,062,177,393,323,753,925,769,530,080,667,314,748,109,769,340,705,616,829,158,713,742,366,956,323,811,658,939,980,376,490,362,206,536,509,163,262,89,5,994,035,528,548,104,714,311,529,907,097,590,979,874,192,495,481,809,959,345,140,169,036,189,397,692,571,708,295,085,680,035,848,878,511,103,780,211,394,150,671,490,415,999,523,971,903,105,532,567,495,107,879,712,588,387,210,084,586,358,044,694,457,753,354,516,325,347,062,681,260,366,814,109,779,422,050,191,032,581,851,452,046,060,881,359,376,620,137,966,167,740,23,9,764,843,152,438,225,370,822,153,538,376,483,436,617,116,334,960,092,927,481,817,805,369,608,748,651,839,786,120,991,198,073,428,704,485,016,027,868,995,877,751,620,428,289,400,641,132,459,288,145,560,902,751,347,098,644,695,563,477,637,451,893,696,302,793,057,389,201,058,088,342,606,342,922,981,190,034,432,698,490,915,886,271,311,961,857,652,026,361,053,628,705,171,60,0,303,150,123,538,685,001,308,779,479,681,857,312,254,943,518,449,704,275,479,642,437,275,140,137,984,669,941,944,836,642,040,223,878,486,538,067,789,693,348,083,730,346,825,273,845,460,061,706,852,290,775,466,985,780,968,592,555,948,972,413,863,120,063,221,143,185,629,834,828,284,332,040,441,845,171,583,418,684,231,860,145,392,992,409,922,598,320,558,427,635,546,149,64,0,259,499,303,896,687,655,889,853,235,173,231,563,594,894,331,355,085,387,107,697,230,905,654,705,259,068,821,648,397,387,329,097,578,047,593,092,846,447,686,613,216,298,865,047,063,582,525,768,691,383,901,918,628,632,995,284,123,553,924,531,341,699,613,651,503,475,557,720,432,902,819,498,996,966,702,546,557,122,665,864,293,765,800,457,150,514,318,233,415,015,125,84,4,481,860,456,154,735,333,547,800,438,057,205,778,685,409,314,260,011,156,961,657,480,785,671,481,976,603,269,586,121,683,413,177,475,131,231,965,528,530,942,225,372,500,539,202,963,473,718,122,984,588,259,769,774,091,345,713,910,425,096,108,544,088,658,248,204,579,751,148,634,012,322,625,699,314,067,337,213,094,08,6,024,904,721,125,159,205,464,697,477,837,951,501,543,061,510,253,460,227,330,486,801,150,033,856,688,083,604,379,379,279,999,371,994,208,371,160,152,723,047,097,310,816,144,163,452,1,006,854,700,418,024,650,230,525,919,19,301,515,727,67,531,657,567,580,163,596,380,227,833,665,239,370,333,047,582,063,688,793,679,363,684,486,168,061,587,575,930,547,105,424,04,0,975,094,918,638,314,255,917,502,944,267,332,343,175,376,082,638,428,085,697,931,895,281,841,412,624,618,678,162,929,064,936,227,990,052,308,716,078,511,409,520,429,551,751,0,802,73,485,1,006,854,700,418,024,650,230,525,919,19,301,515,727,67,531,657,567,580,163,596,380,227,833,665,239,370,333,047,582,063,688,793,679,363,684,486,168,061,587,575,930,547,105,424,04,0,834,551,988,621,361,739,581,412,576,048,062,774,658,251,424,538,120,236,005,887,704,520,244,777,805,903,136,11,154,130,093,555,699,675,759,909,511,071,459,657,675,202,737,485,1,006,854,700,418,024,650,230,525,919,19,301,515,727,67,531,657,567,580,163,596,380,227,833,665,239,370,333,047,582,063,688,793,679,363,684,486,168,061,587,575,930,547,105,424,04,0,307,903,520,541,402,450,242,719,240,797,326,025,484,993,704,399,679,794,871,910,927,023,654,282,339,225,177,224,226,226,765,633,552,919,301,515,727,67,531,657,567,580,163,596,380,227,833,665,239,370,333,047,582,063,688,793,679,363,684,486,168,061,587,575,930,547,105,424,04,4,143,464,823,162,427,053,006,177,894,952,451,951,244,958,914,648,418,596,424,889,789,008,679,263,496,096,289,78,850,418,514,767,339,021,389,7,27,47,705,7,009,882,939,36,06,41,476,3,340,55,50,574,33,99,819,38,516,844,803,259,941,917,998,483,757,110,114,029,135,092,405,018,779,285,891,809,345,880,290,584,450,371,392,611,286,191,050,79,2,529,551,802,851,823,99,309,632,705,746,023,254,826,875,146,129,983,877,945,558,650,786,101,744,970,452,299,404,607,487,078,3,39,996,706,7,44,098,573,12,82,116,8,648,576,088,1,25,34,151,2,15,47,756,385,37,19,000,89,433,649,234,393,079,979,131,039,365,729,356,796,938,706,763,568,193,651,332,269,863,240,182,237,146,533,295,405,342,240,355,24,2,109,979,044,395,569,527,874,572,748,166,968,677,276,493,457,207,047,533,055,279,088,933,682,692,014,140,504,874,3,39,996,706,7,44,098,573,12,82,116,8,648,576,088,1,25,34,151,2,15,47,756,385,37,19,000,89,433,649,234,393,079,979,131,039,365,729,356,796,938,706,763,568,193,651,332,269,863,240,182,237,146,533,295,405,342,240,355,24,0,488,918,480,236,589,450,225,691,849,471,266,440,724,147,764,862,569,534,948,241,458,843,759,034,565,370,282,54,1,006,854,700,418,024,650,230,525,919,19,301,515,727,67,531,657,567,580,163,596,380,227,833,665,239,370,333,047,582,063,688,793,679,363,684,486,168,061,587,575,930,547,105,424,04,0,085,119,099,979,544,055,147,487,323,341,215,334,555,709,196,780,139,933,682,500,741,839,812,932,156,351,120,796,247,468,214,657,244,618,195,095,030,506,657,808,740,291,107,993,818,182,376,397,882,737,632,741,745,747,195,351,545,715,871,008,258,681,361,684,966,548,660,619,181,449,534,458,489,173,580,647,791,296,269,246,341,186,055,766,932,761,559,150,521,309,231,891,73,8,476,118,424,685,199,962,976,418,495,044,193,590,453,686,856,782,078,504,306,975,602,130,322,378,117,418,749,641,128,299,141,724,823,021,053,565,651,713,404,997,370,697,438,118,519,620,081,631,942,455,323,286,536,535,509,426,261,805,775,933,903,944,172,951,093,268,252,243,360,784,794,424,939,667,547,678,023,938,008,044,711,347,695,593,952,595,645,764,20,8,697,262,776,008,005,259,823,852,668,363,210,327,895,409,895,710,606,975,967,654,501,181,761,916,387,719,452,260,299,629,849,433,609,758,473,948,413,349,455,194,564,403,806,274,966,798,391,604,464,531,940,912,243,207,508,481,482,784,616,291,051,255,667,740,576,936,388,024,757,675,016,888,156,274,288,245,856,121,912,221,021,943,453,864,780,323,759,118,283,507,272,635,996,50,5,745,462,224,497,094,163,590,997,074,700,818,017,606,283,851,984,670,403,315,773,016,215,268,600,152,326,505,929,064,380,061,985,889,686,166,593,526,608,260,345,504,672,816,451,020,882,755,976,378,512,567,192,578,233,745,944,174,487,961,334,187,784,916,308,168,713,658,999,300,284,134,750,439,153,982,921,779,323,332,343,834,027,743,015,241,922,957,054,075,061,365,948,32,6,552,811,778,580,197,735,305,520,304,113,659,696,933,224,042,243,414,282,607,459,803,128,399,687,495,473,309,434,678,003,887,995,387,400,170,190,820,290,583,561,307,179,115,229,188,565,097,240,042,602,940,124,326,205,808,731,721,022,643,993,908,246,960,833,228,017,215,327,927,490,179,886,175,843,896,484,778,643,410,817,269,690,317,212,721,645,512,393,883,995,44,7,732,760,535,159,680,365,360,118,842,821,109,089,391,140,333,773,312,517,258,060,363,877,298,255,941,817,162,718,561,438,303,458,401,089,912,150,516,256,341,442,449,652,054,770,023,413,946,313,724,050,167,525,187,041,208,646,911,228,350,990,403,668,199,576,324,233,819,946,029,869,949,279,451,686,312,076,828,179,937,708,739,396,387,134,174,085,385,333,643,463,257,874,58,0,597,240,792,690,571,190,641,405,631,195,239,721,208,985,551,093,571,538,370,792,573,232,757,096,258,464,554,855,642,311,617,374,673,942,354,602,528,830,612,341,852,932,686,815,306,450,185,341,484,849,099,491,762,837,620,237,386,842,973,381,520,626,080,117,262,315,168,261,863,461,361,340,918,577,375,417,476,262,366,454,025,183,582,024,337,301,164,714,138,

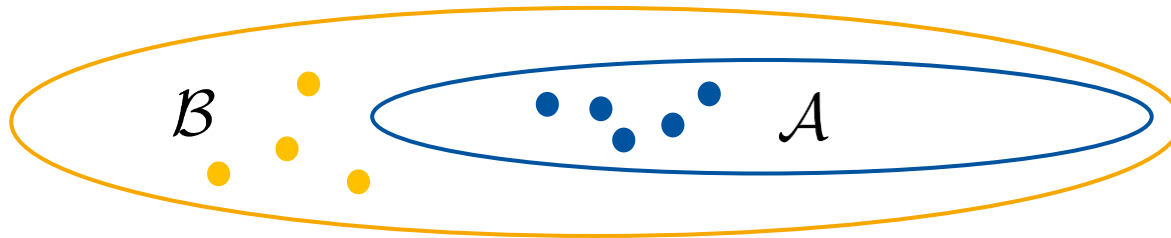
Subsets of Frequent Itemsets Are Also Frequent

- 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}

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

Closed Itemsets

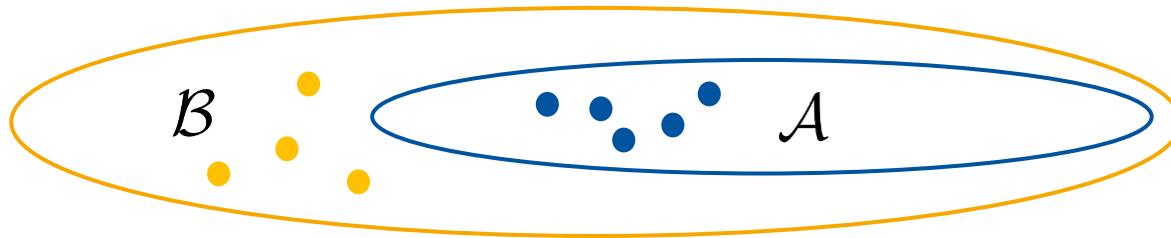
- An itemset \mathcal{A} is **closed** if there is no **proper superset** $\mathcal{B} \supset \mathcal{A}$ that has the same support
- If \mathcal{A} is **closed**, then $\text{support}(\mathcal{A}) > \text{support}(\mathcal{B})$ for any $\mathcal{B} \supset \mathcal{A}$



adding any item to \mathcal{A} will always **reduce support**

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

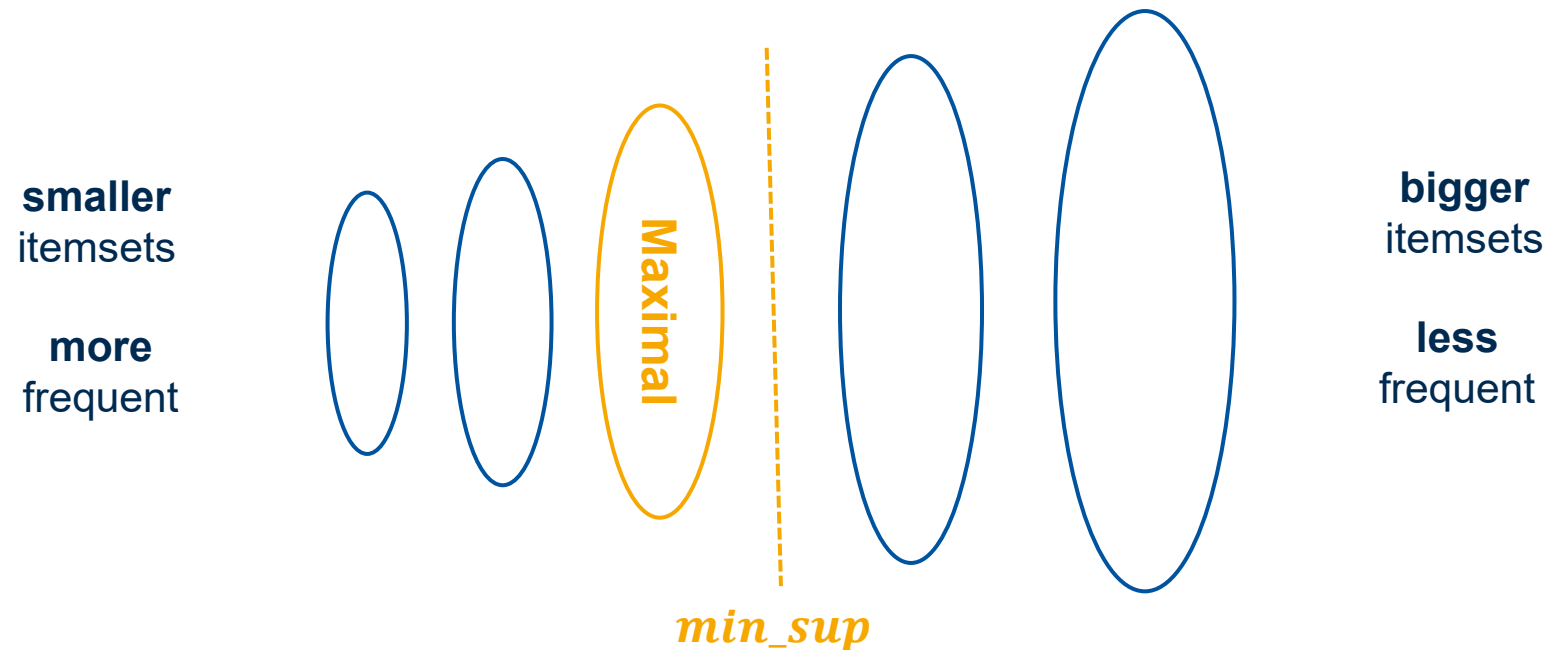


closed frequent itemsets are **closed and frequent**

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

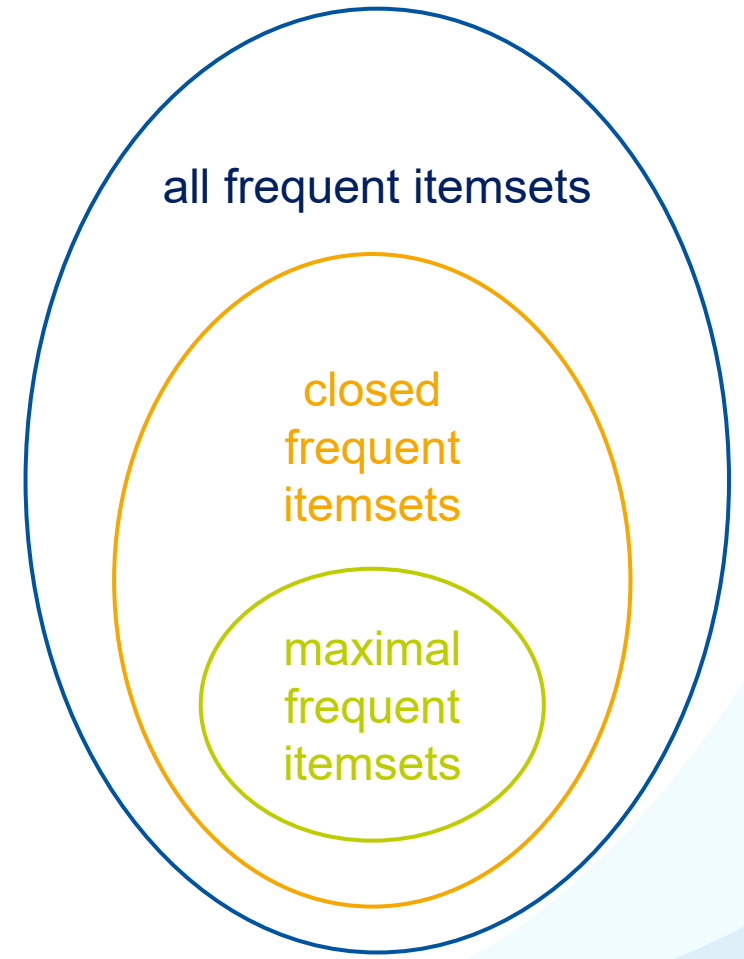
An itemset \mathcal{A} is a **closed frequent itemset** if:

- \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

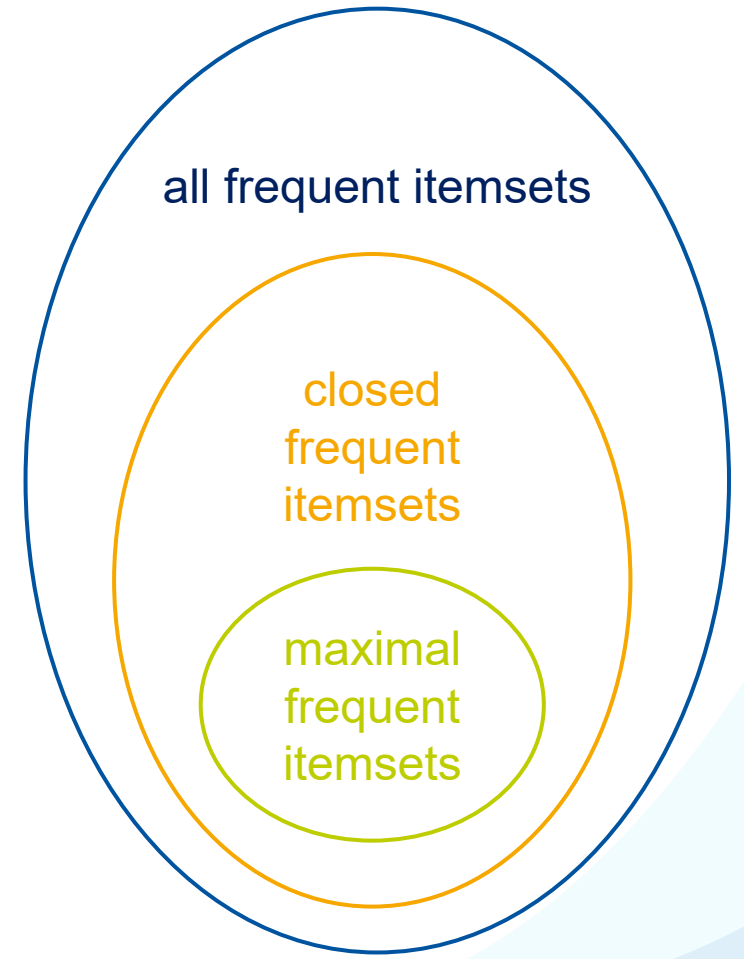
Hence, maximal frequent itemsets are closed by definition.



Observations

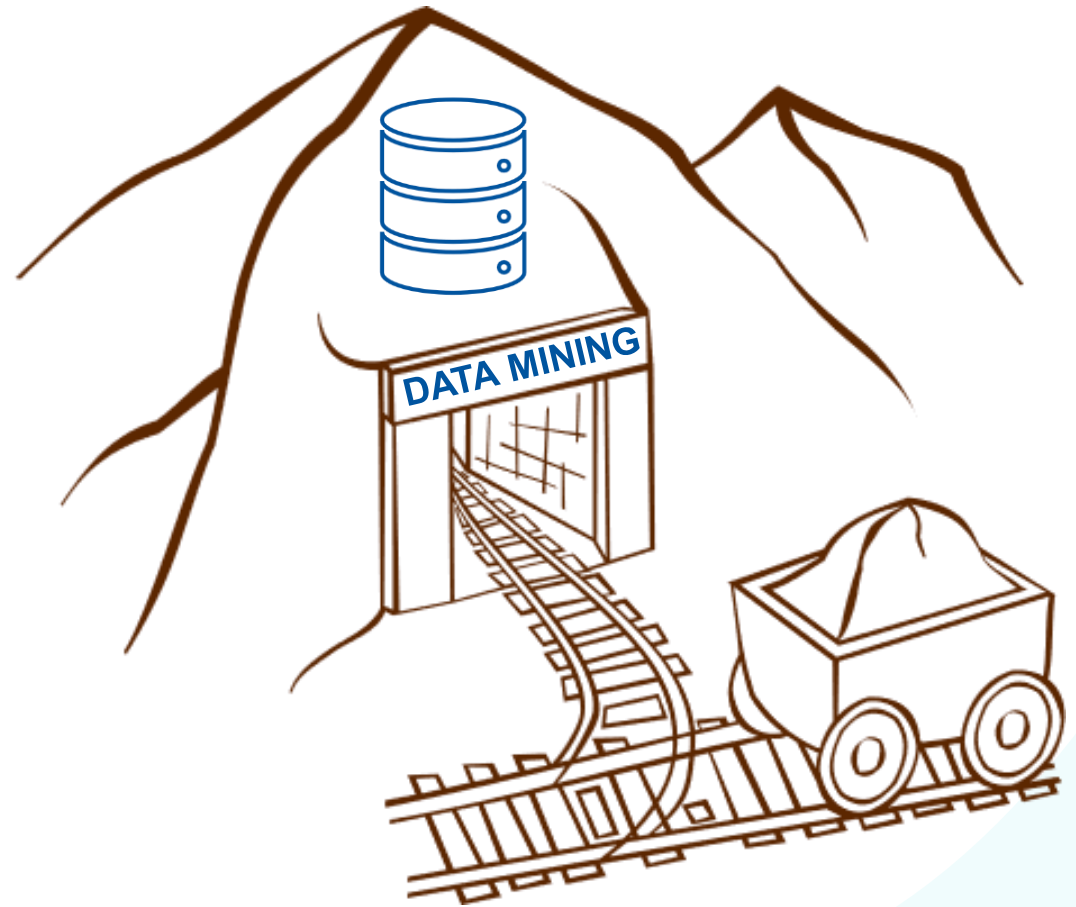
- The supports of the **closed** frequent itemsets provide complete information about the supports of **all** frequent item sets
- Formally, assume:
 - $\mathcal{A} \subset \mathcal{B}$,
 - \mathcal{B} is a **closed** frequent itemset, and
 - there is no **closed** frequent itemset \mathcal{B}' such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

→ 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 \rightarrow 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 \rightarrow 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 \rightarrow 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)

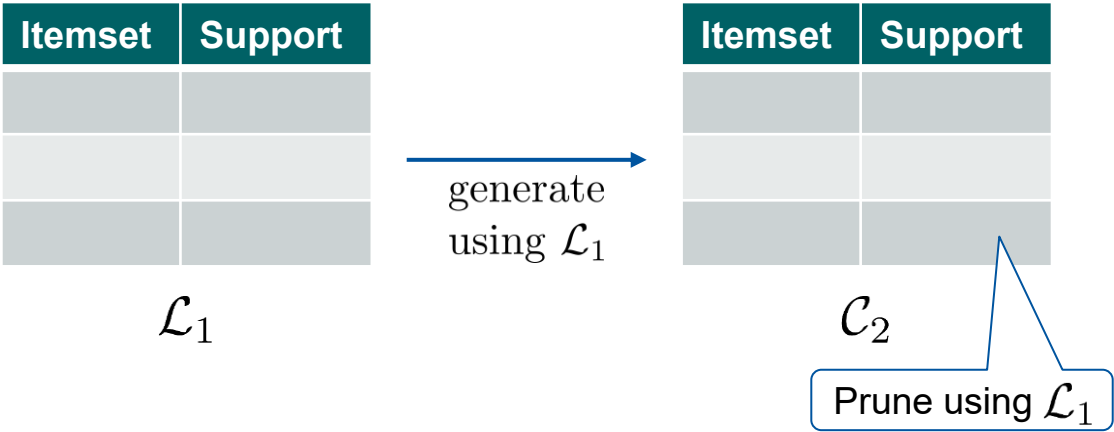
needs the
input data
(inefficient)

Apriori Algorithm – Basic Idea

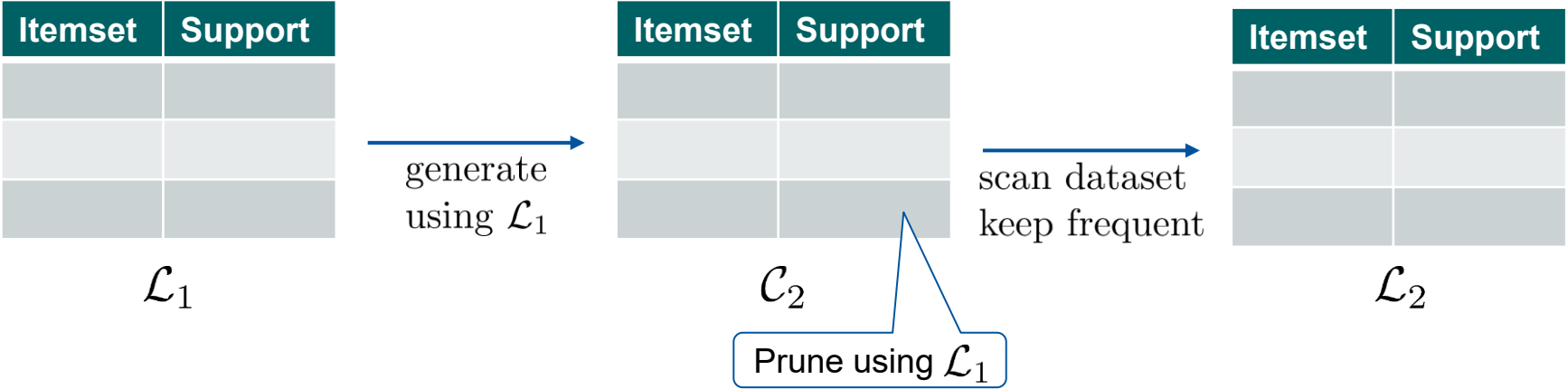
Itemset	Support

\mathcal{L}_1

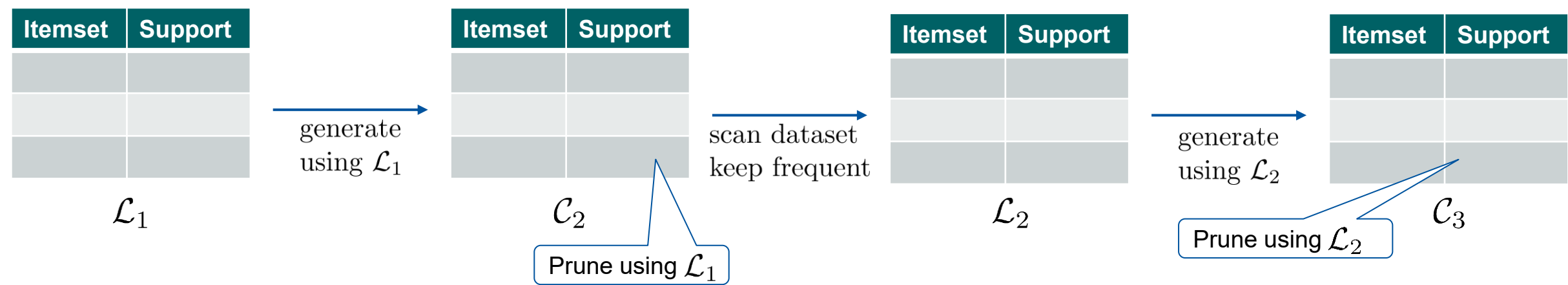
Apriori Algorithm – Basic Idea



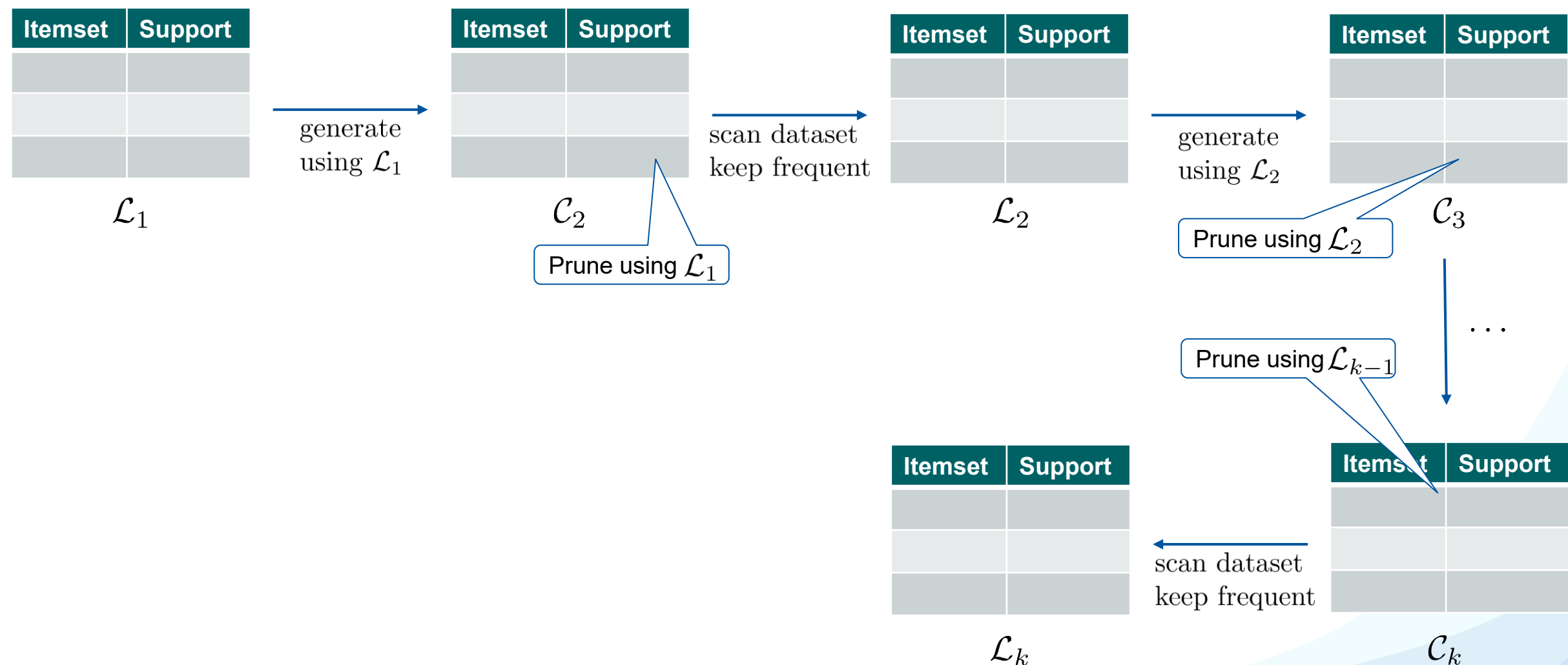
Apriori Algorithm – Basic Idea



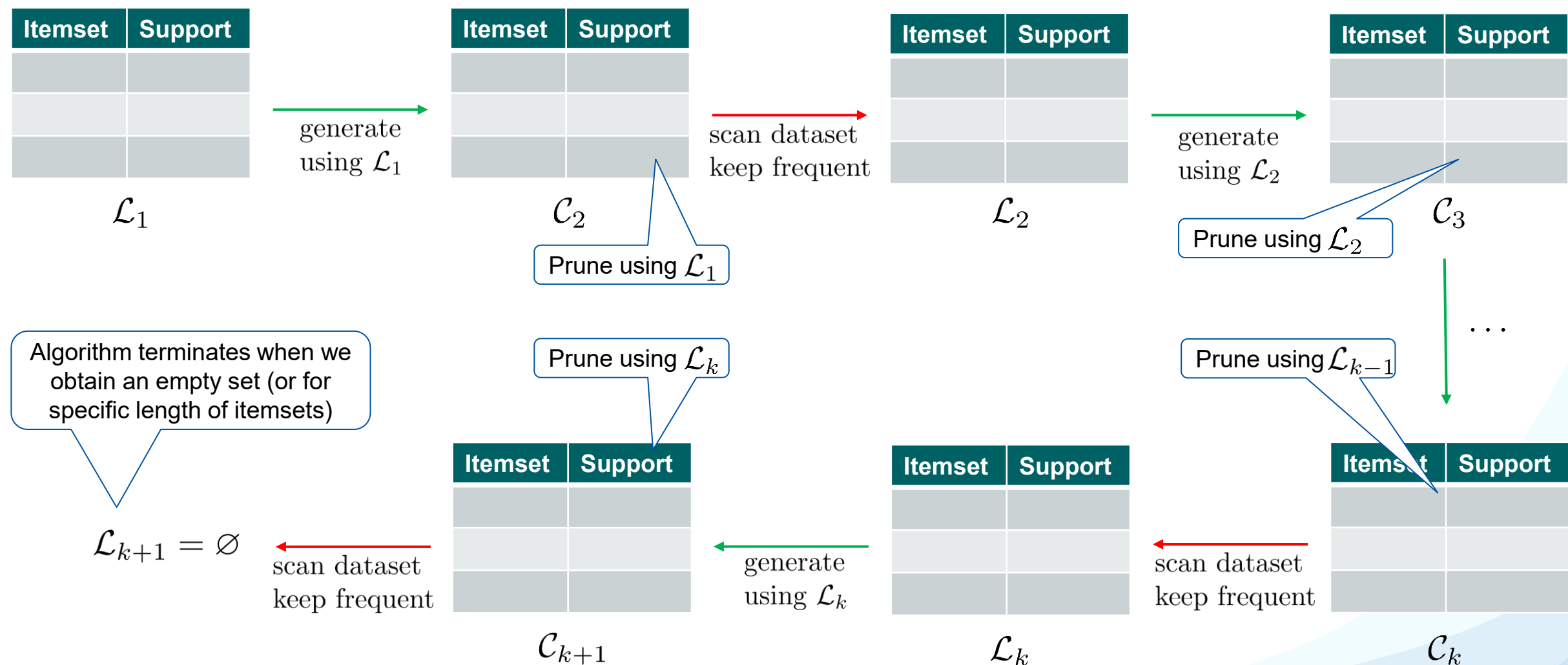
Apriori Algorithm – Basic Idea



Apriori Algorithm – Basic Idea



Apriori Algorithm – Basic Idea

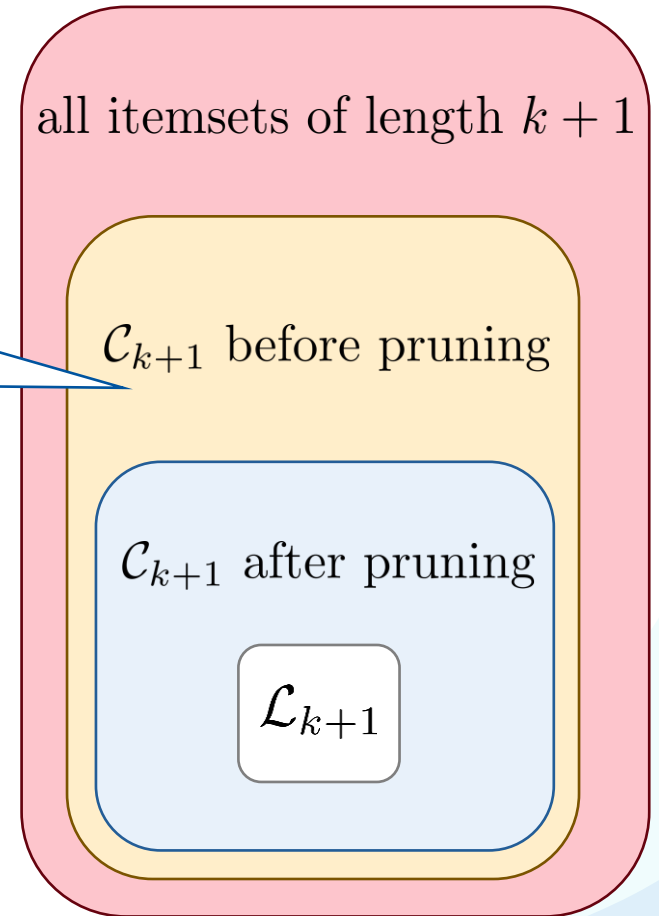


Candidate Generation – Leveling

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 $\mathcal{A}', \mathcal{A}'' \in \mathcal{L}_k$!



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 $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$

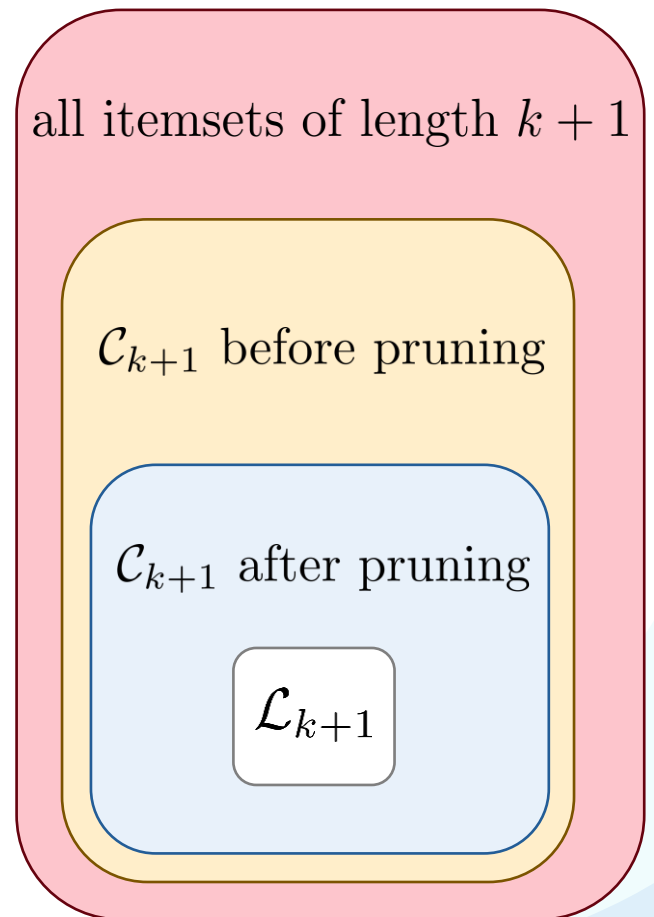
Assume that the items are ordered (I_1, I_2, \dots) and that $\mathcal{A} = \{I_1, I_2, \dots, I_{k-1}, I_k, I_{k+1}\} \in \mathcal{L}_{k+1}$

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

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

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

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



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

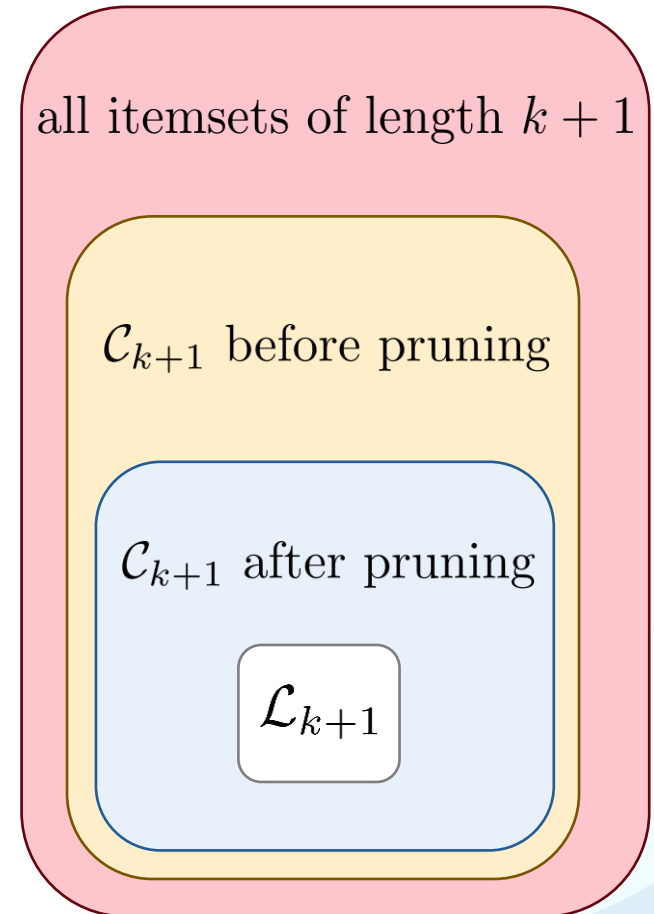
If \mathcal{A} is frequent, its subsets must be frequent, in particular:

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

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

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

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



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

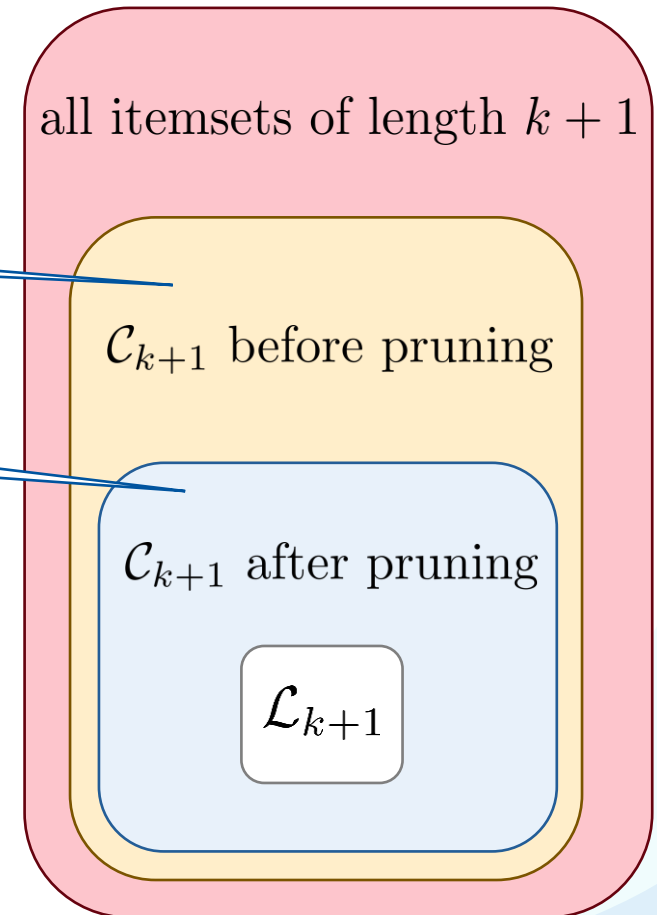
If \mathcal{A} is frequent, its subsets must be frequent, in particular:

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

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

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

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



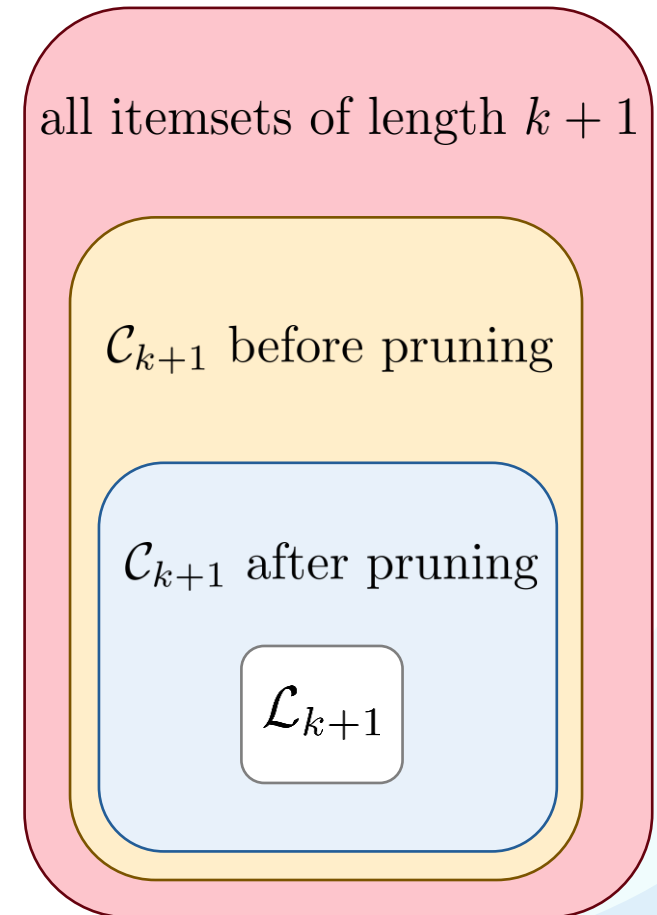
Pruning – Antimonotonicity

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

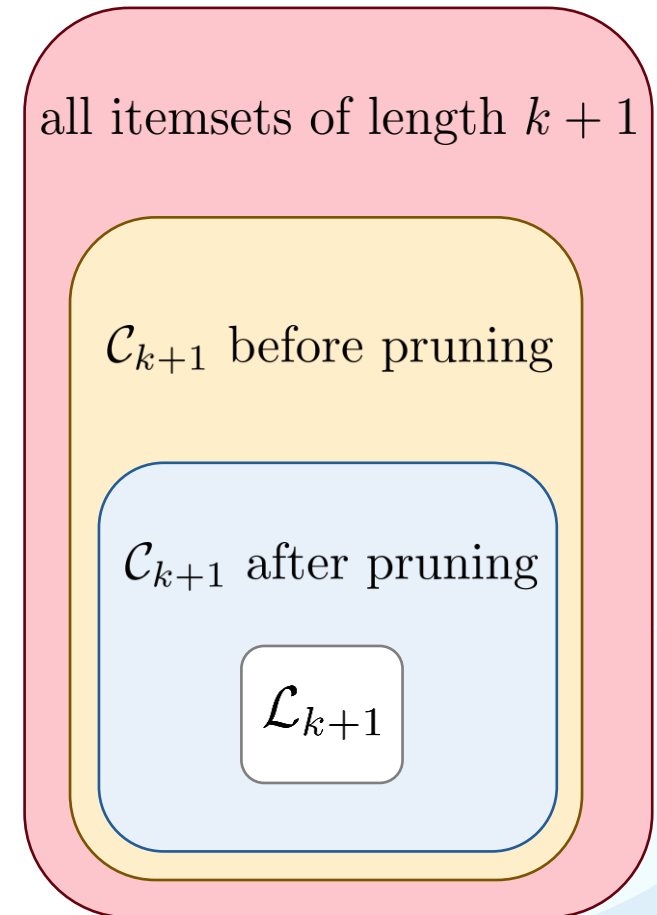


Pruning – Antimonotonicity

Antimonotonicity is used to **prune** the candidate set:

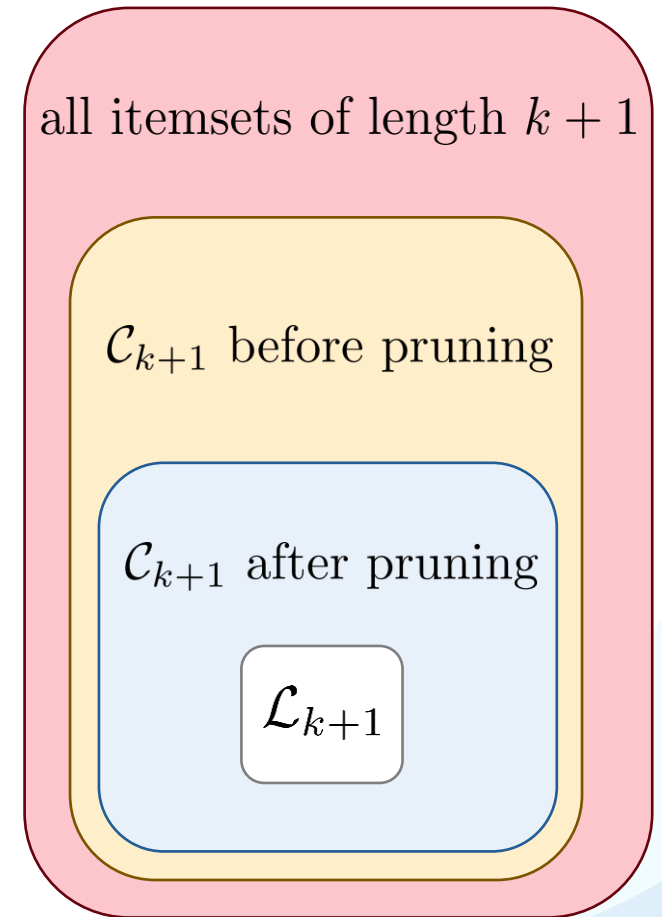
If \mathcal{B} is a frequent itemset, any subset $\mathcal{A} \subseteq \mathcal{B}$ must be frequent
 \Rightarrow If a subset $\mathcal{A} \subseteq \mathcal{B}$ is infrequent, then \mathcal{B} is infrequent

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

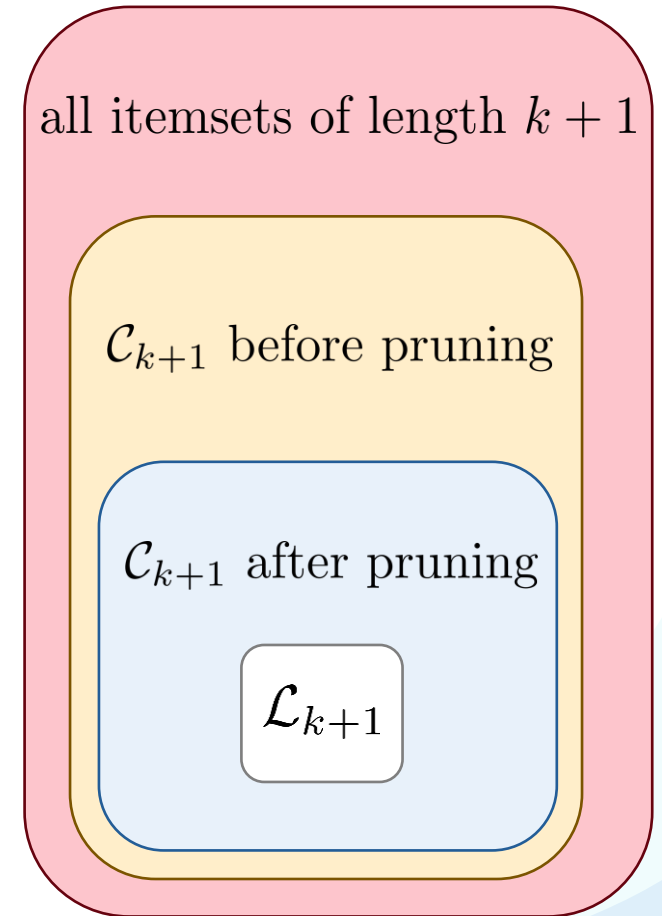
- After candidate generation and pruning, we test the remaining candidate itemsets
- We scan the dataset \mathcal{X} and **remove all infrequent candidate itemsets** from \mathcal{C}_{k+1} to obtain \mathcal{L}_{k+1}



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** $\mathcal{L}_k = \emptyset$:

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

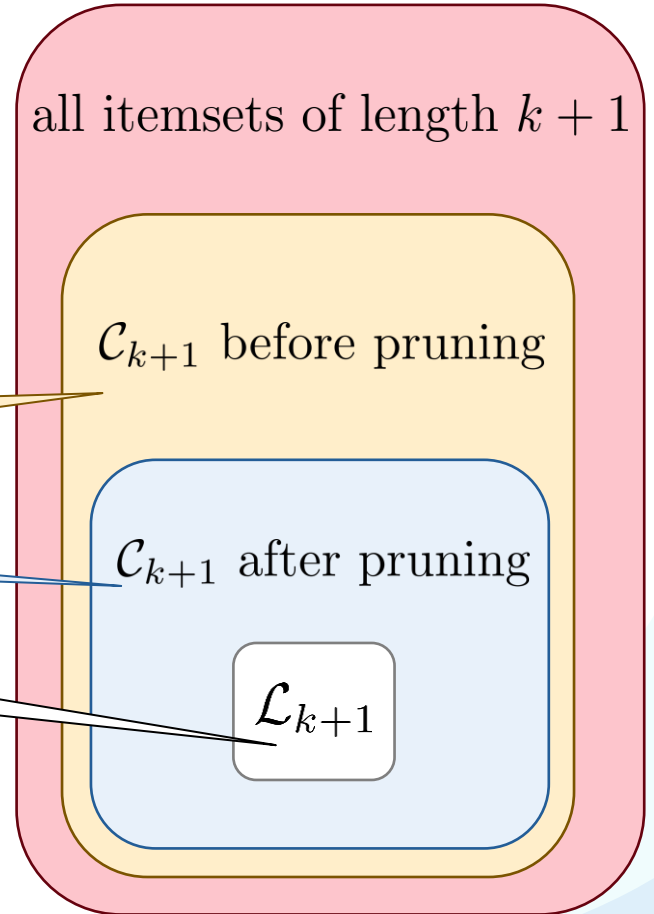
(a) Generate set of candidate itemsets of length $k+1$ (\mathcal{C}_{k+1})
based on \mathcal{L}_k

(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** $\bigcup_{i=1}^k \mathcal{L}_i$



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

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

min_sup_count = 2

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

\mathcal{L}_1

generate candidates from \mathcal{L}_1

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

\mathcal{C}_2

pruning based on \mathcal{L}_1

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

min_sup_count = 2

generate candidates from \mathcal{L}_1

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

\mathcal{C}_2

pruning based on \mathcal{L}_1

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

\mathcal{C}_2

min_sup_count = 2

scan dataset
test candidates

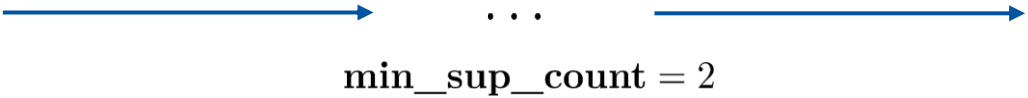
Itemset	Count
{Grapes, Apple}	3
{Grapes, Orange}	1
{Grapes, Banana}	2
{Apple, Orange}	2
{Apple, Banana}	3
{Orange, Banana}	3

\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}



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

\mathcal{L}_2

Example

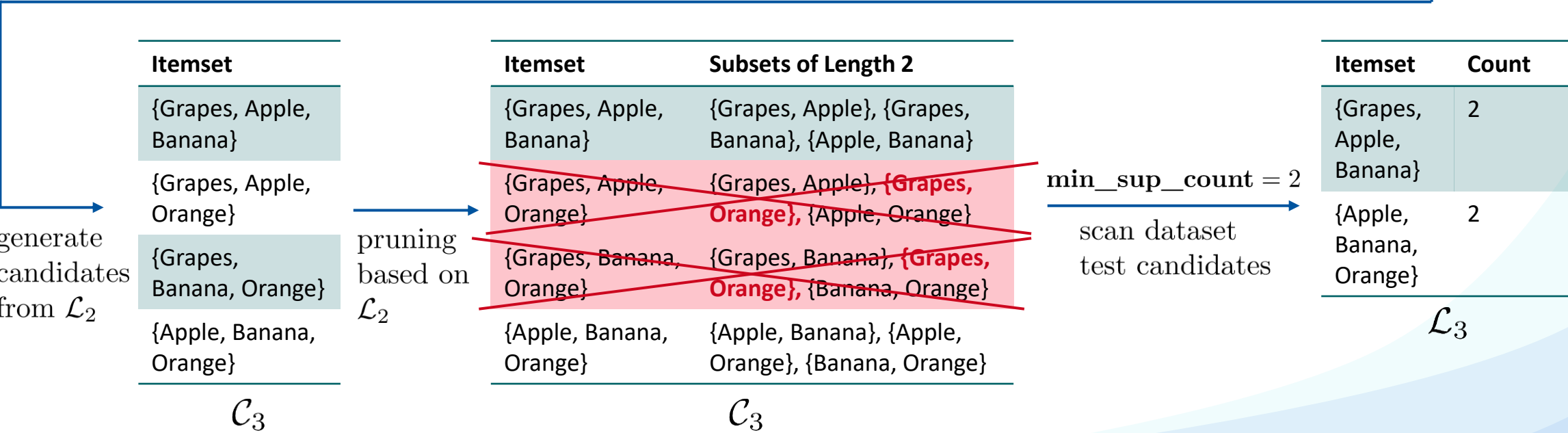
TID	Bought Fruits
1	{Grapes, Apple, Pineapple}
2	{Orange, Apple, Banana}
3	{Grapes, Orange, Apple, Banana}
4	{Orange, Banana}
5	{Grapes, Apple, Banana}

\mathcal{X}

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

\mathcal{L}_2

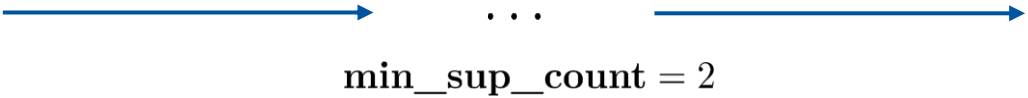
min_sup_count = 2



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, 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

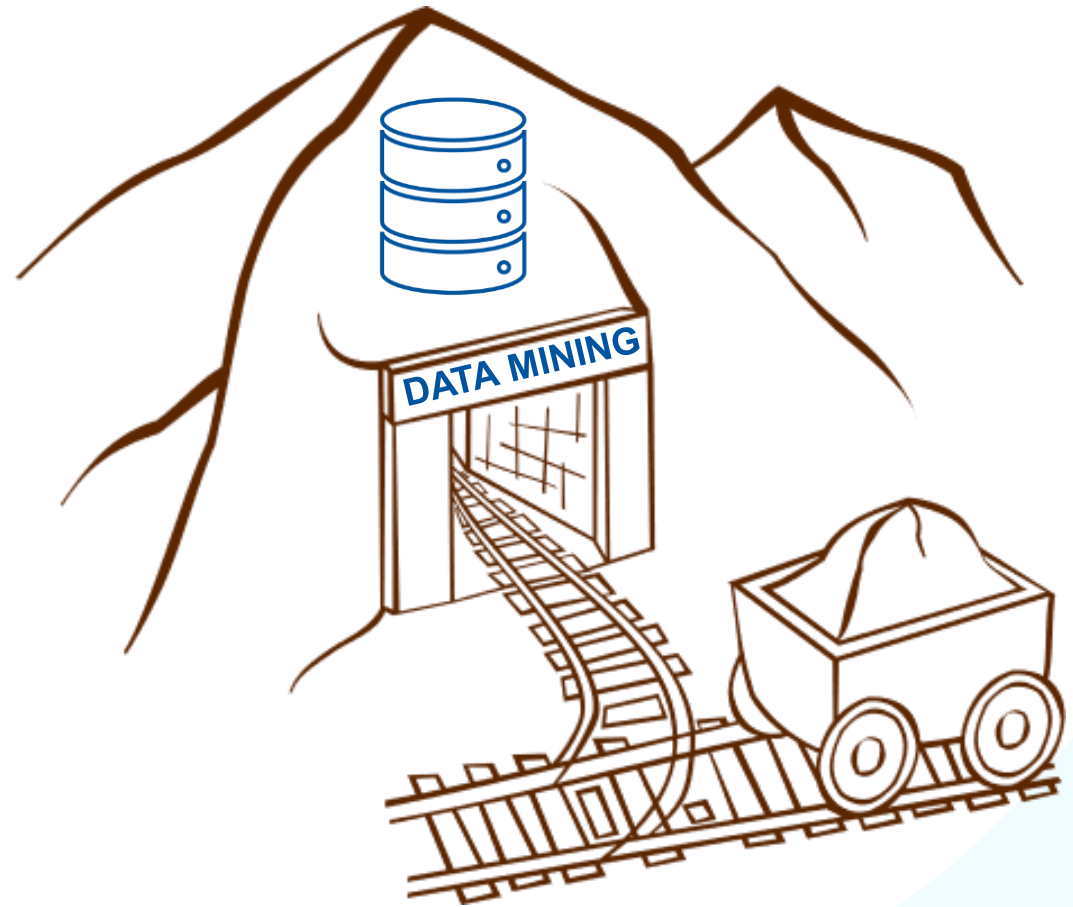
Limitations

- 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

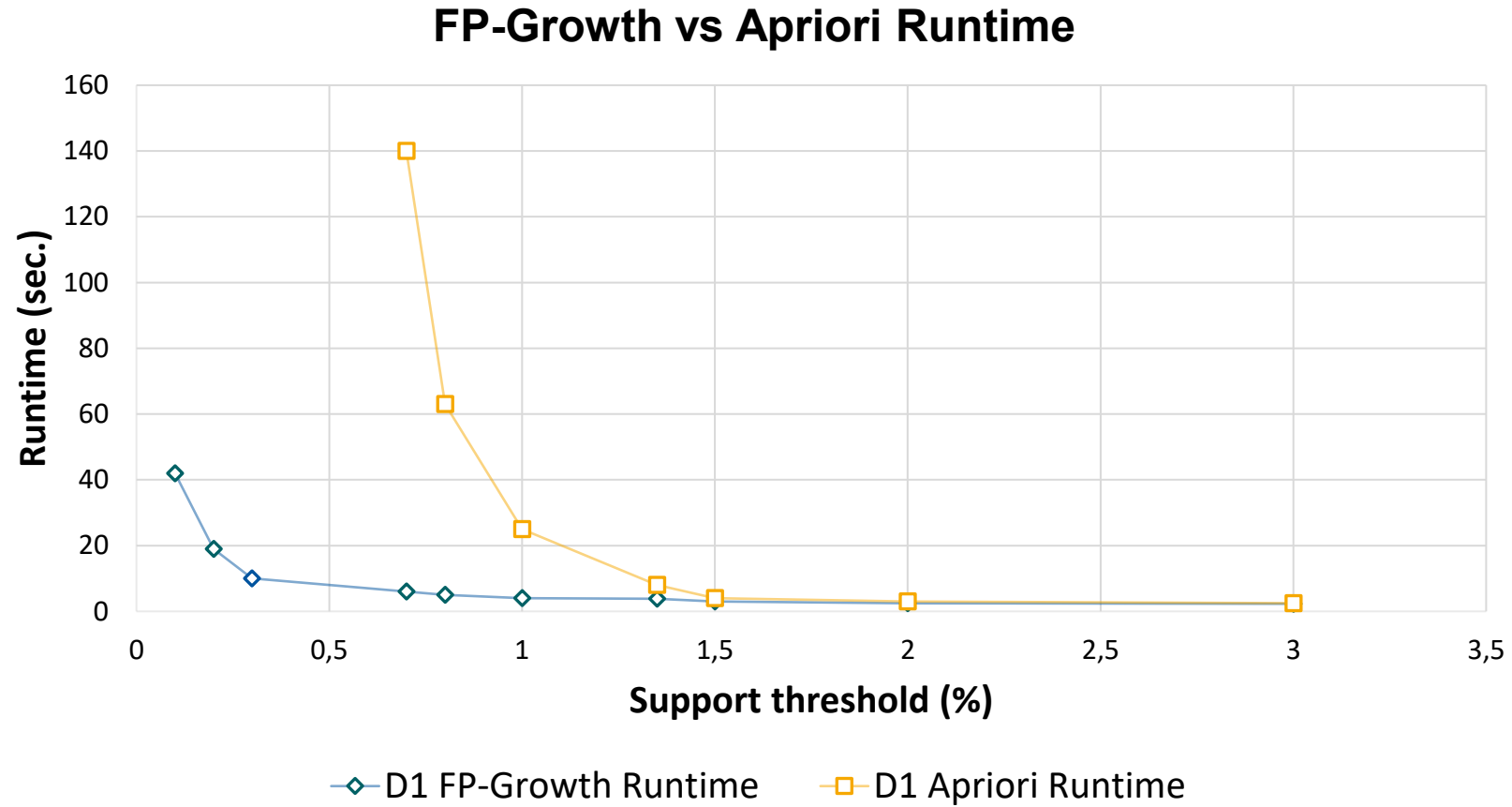
1. Introduction
2. Properties of Frequent Itemsets
3. Apriori Algorithm
4. **FP-Growth Algorithm**



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 through the dataset)

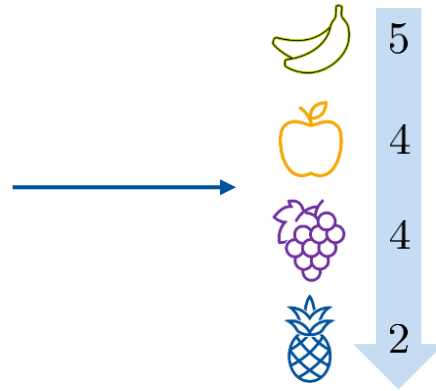
FP-Growth Steps

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

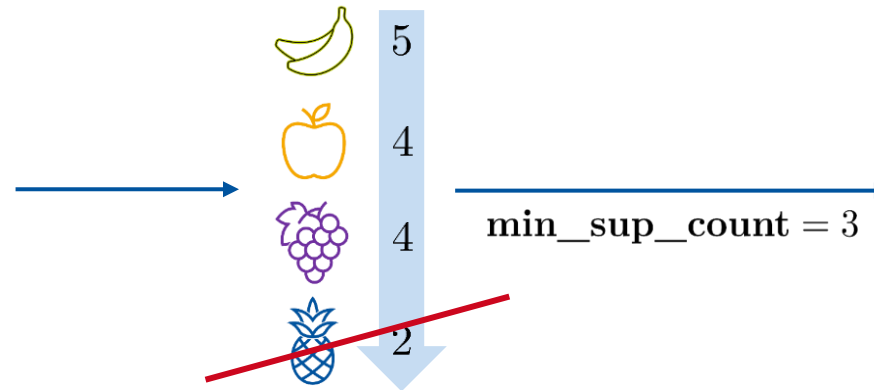


1. Determine the frequencies of items
2. Order the items based on frequency

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

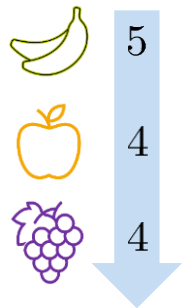
TID	Bought Fruits
1	{Banana, Apple}
2	{Banana, Grapes}
3	{Banana, Apple}
4	{Apple, Grapes}
5	{Banana, Grapes}
6	{Banana, Apple, Grapes}

3. Remove non-frequent items
4. Sort the items in the transactions

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}

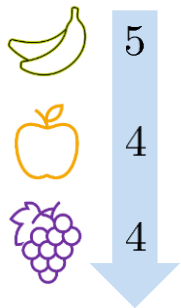
5. Build the FP-tree going through each transaction



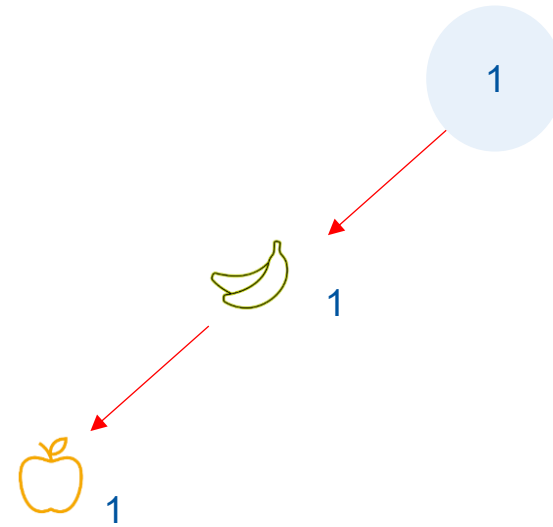
0

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}

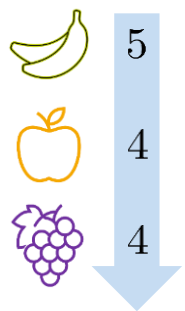


5. Build the FP-tree going through each transaction

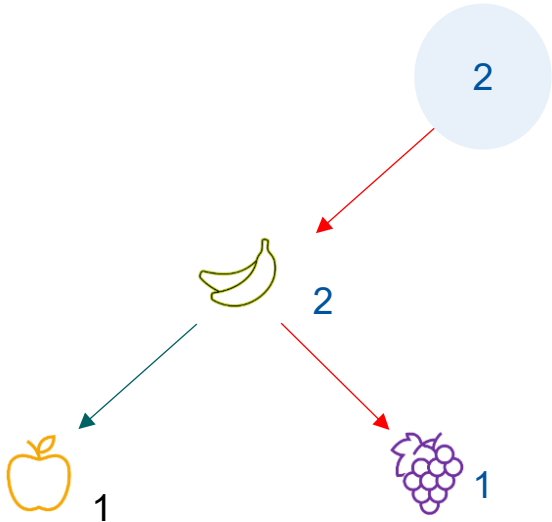


Constructing FP-Tree – Example

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

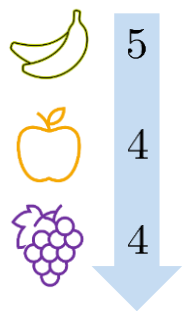


5. Build the FP-tree going through each transaction

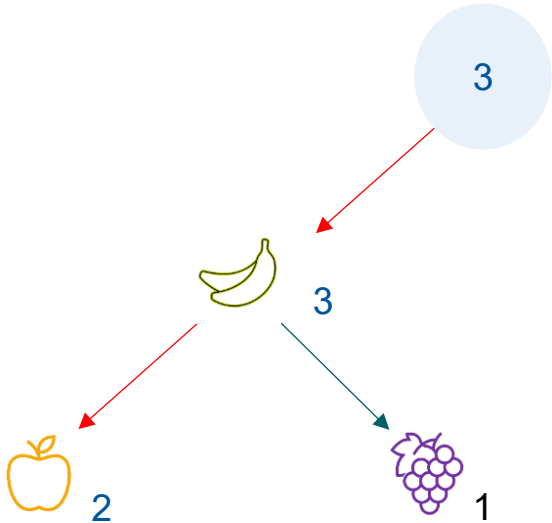


Constructing FP-Tree – Example

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

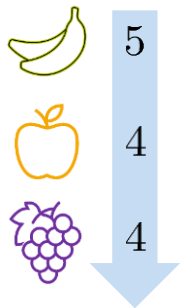


5. Build the FP-tree going through each transaction

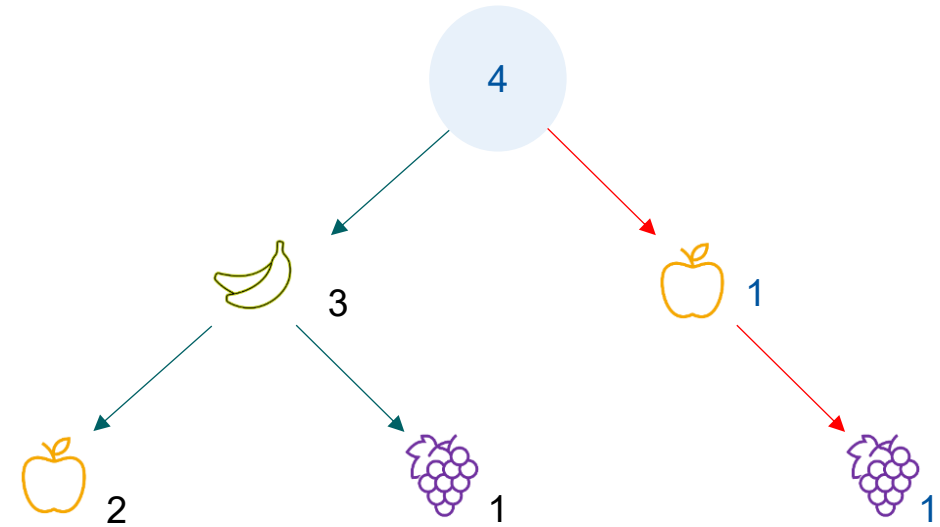


Constructing FP-Tree – Example

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

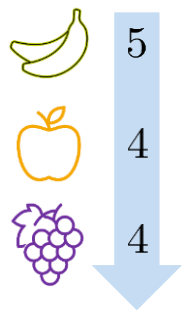


5. Build the FP-tree going through each transaction

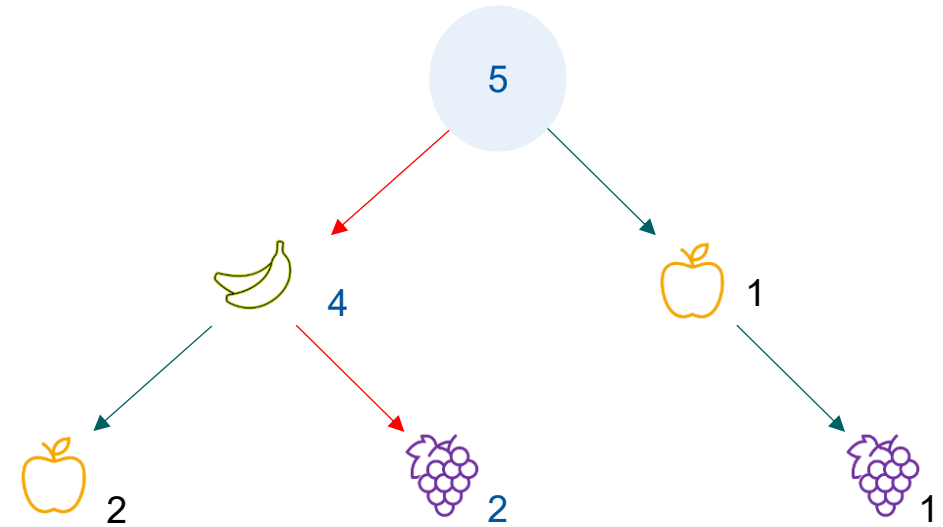


Constructing FP-Tree – Example

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

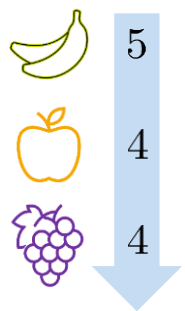


5. Build the FP-tree going through each transaction

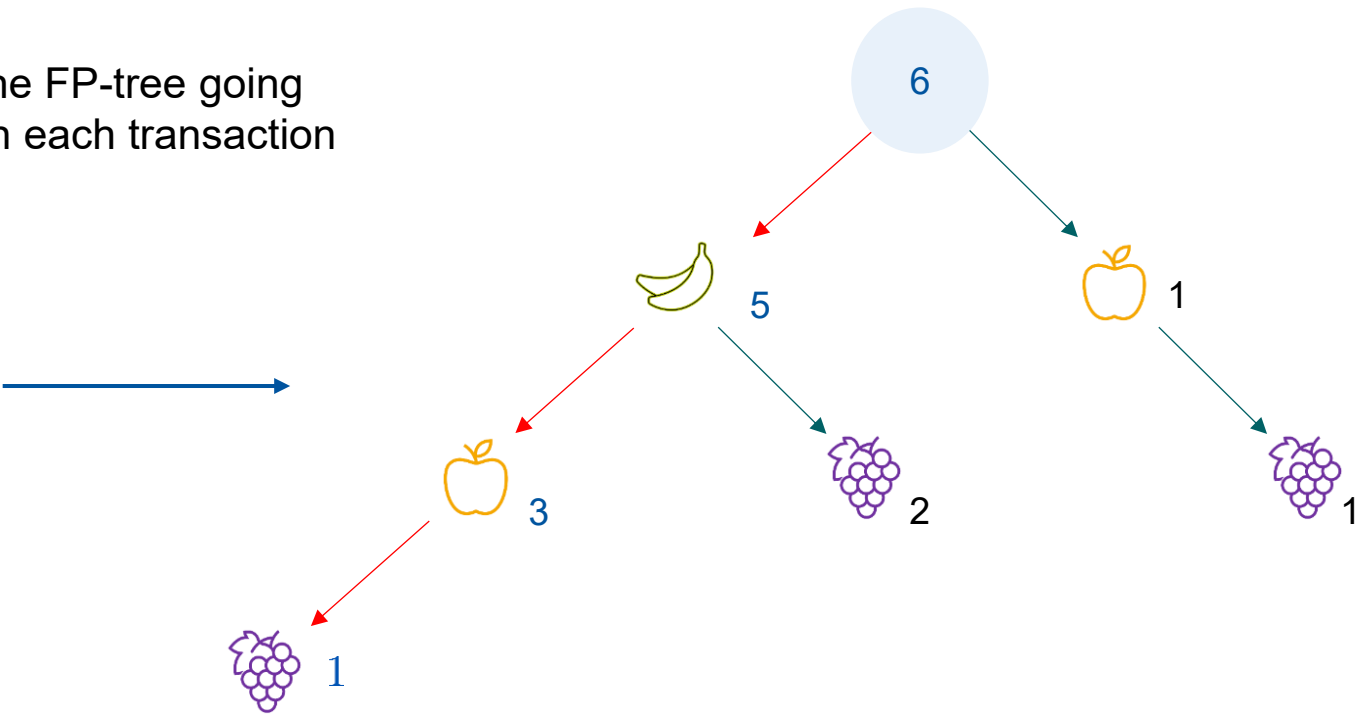


Constructing FP-Tree – Example

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

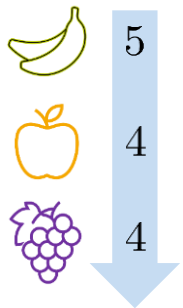


5. Build the FP-tree going through each transaction

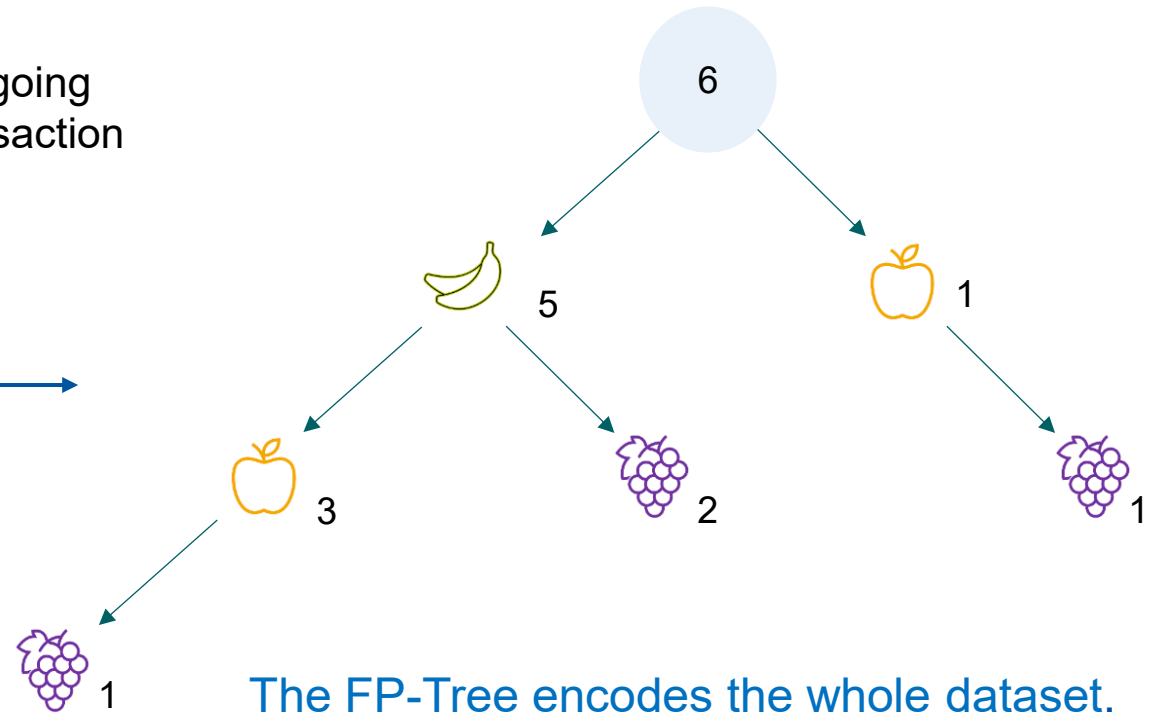


Constructing FP-Tree – Example

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



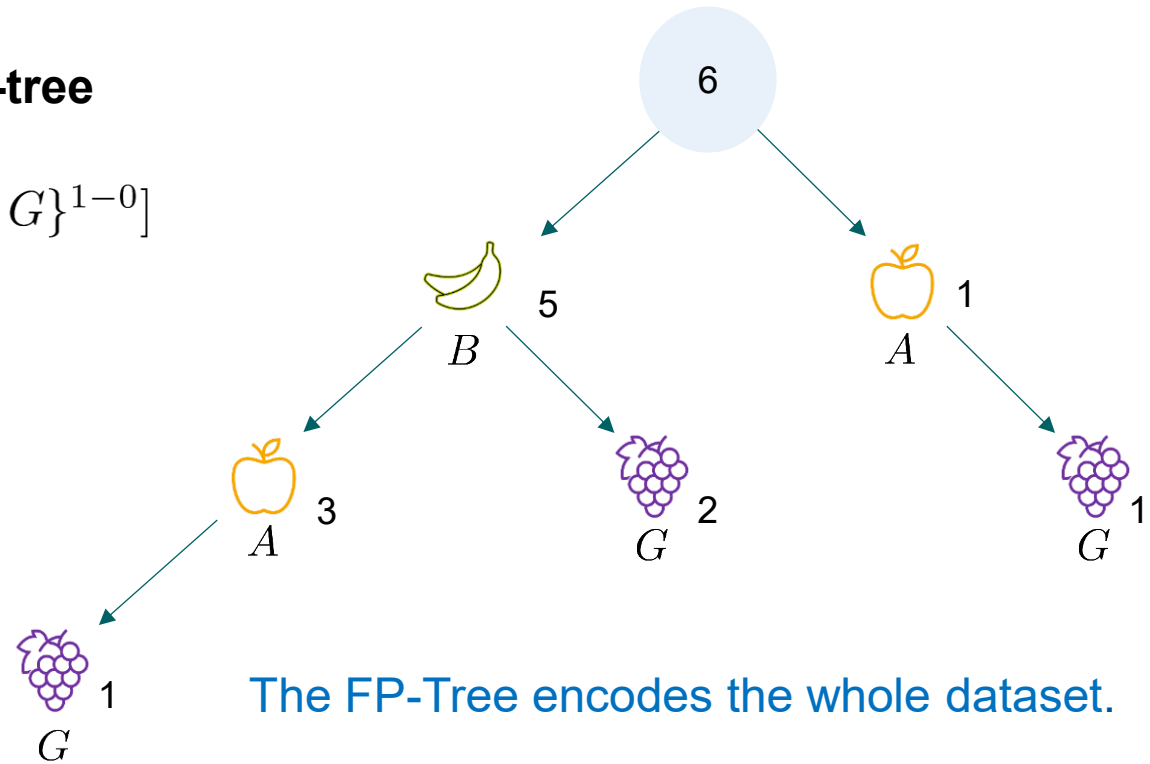
5. Build the FP-tree going through each transaction



FP-Tree – Encodes The Dataset

We can read the transactions from the FP-tree

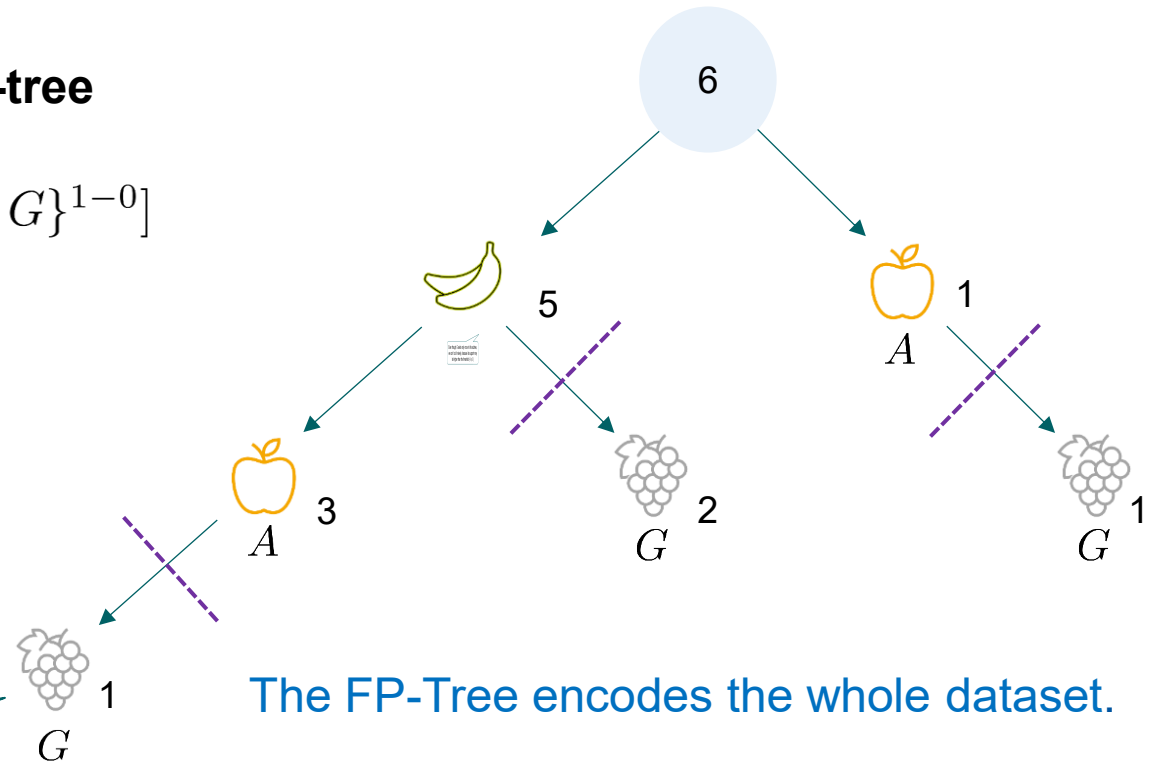
$$\begin{aligned}\mathcal{X} &= [\{B, A, G\}^{1-0}, \{B, A\}^{3-1}, \{B, G\}^{2-0}, \{A, G\}^{1-0}] \\ &= [\{B, A, G\}, \{B, A\}^2, \{B, G\}^2, \{A, G\}]\end{aligned}$$



FP-Tree – Cannot Cut Naïvely

We can read the transactions from the FP-tree

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



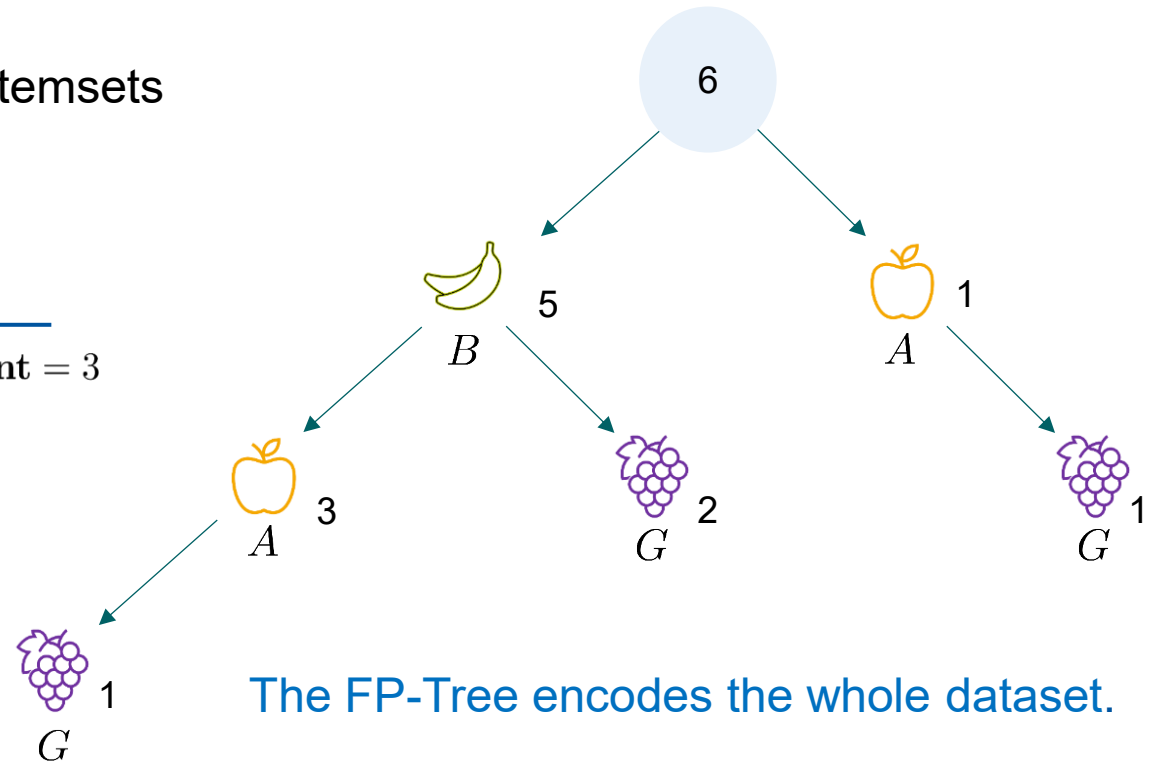
Even though G exists only once in this subtree, we can't cut it naively, because its support may be higher than the threshold ($4 \geq 3$)

FP-Tree – Frequent Itemsets

Next: Mining the FP-tree to obtain frequent itemsets

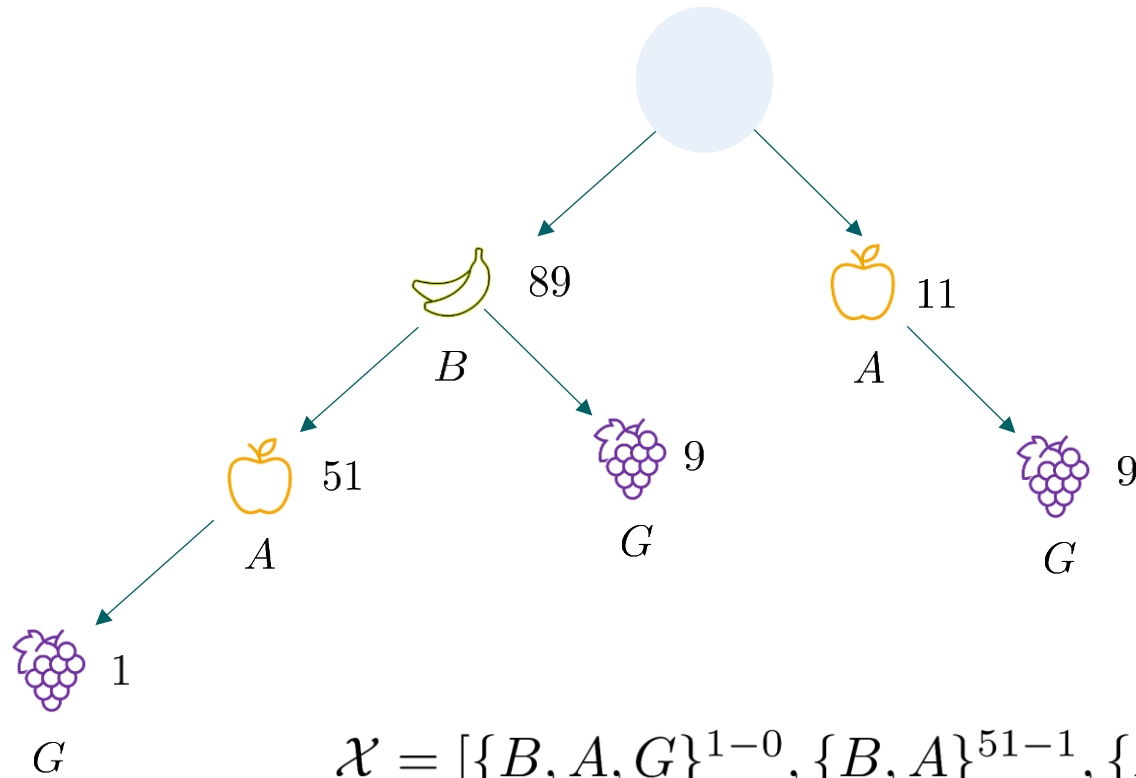
Frequent Itemsets	Support Count
{B}	5
{A}	4
{G}	4
{B, A}	3
{B, G}	3

← min_sup_count = 3



FP-Tree Encodes Dataset – Another Example

We can read the transactions from the FP-tree

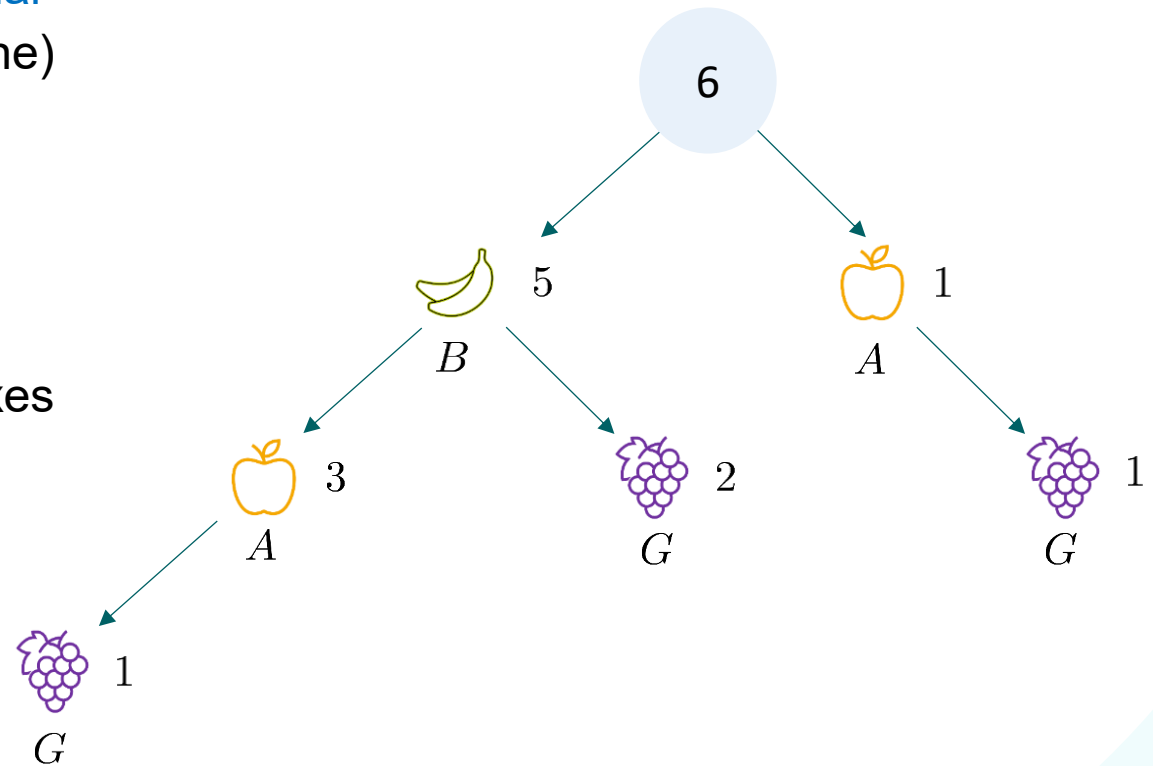


$$\mathcal{X} = [\{B, A, G\}^{1-0}, \{B, A\}^{51-1}, \{B, G\}^{9-0}, \{B\}^{89-(51+9)}, \{A, G\}^{9-0}, \{A\}^{11-9}]$$

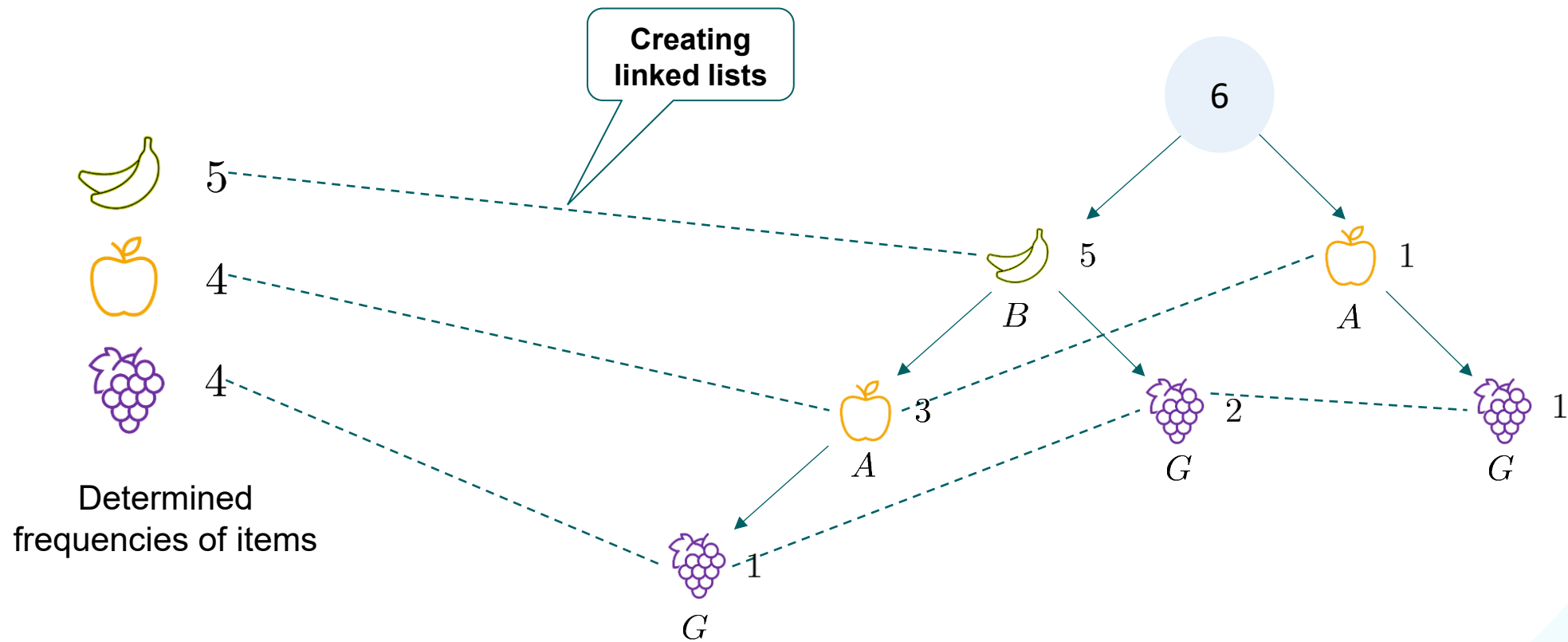
$$\mathcal{X} = [\{B, A, G\}^1, \{B, A\}^{50}, \{B, G\}^9, \{B\}^{29}, \{A, G\}^9, \{A\}^2]$$

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

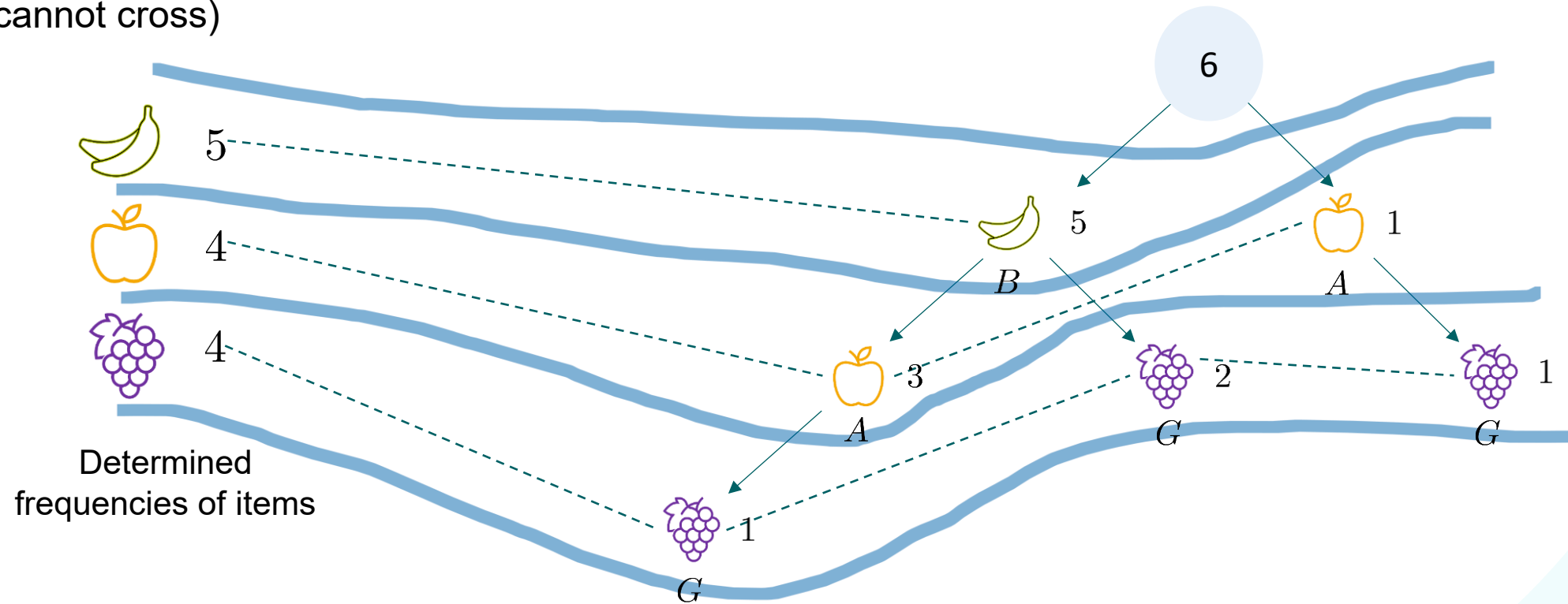


Node Links

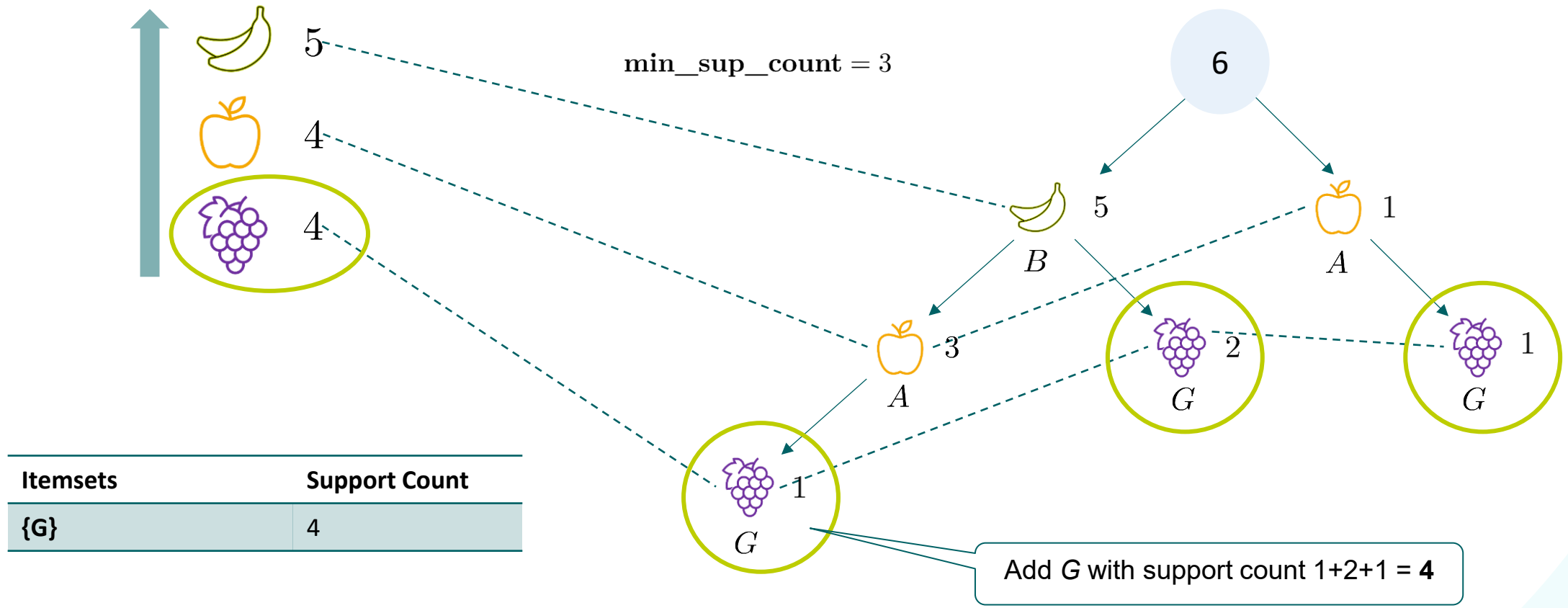


Node Links

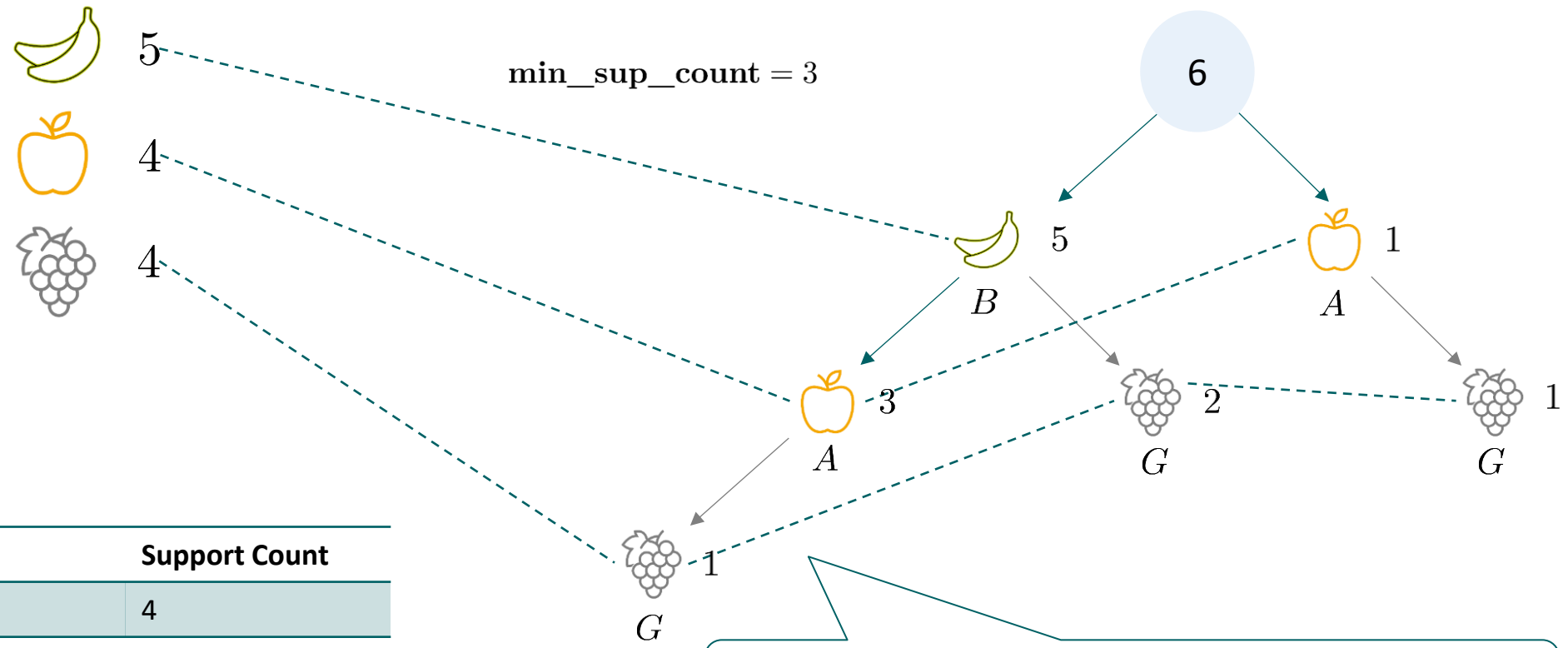
Node links are like 'altitude lines' because of total order of items
(they cannot cross)



Consider Postfix G

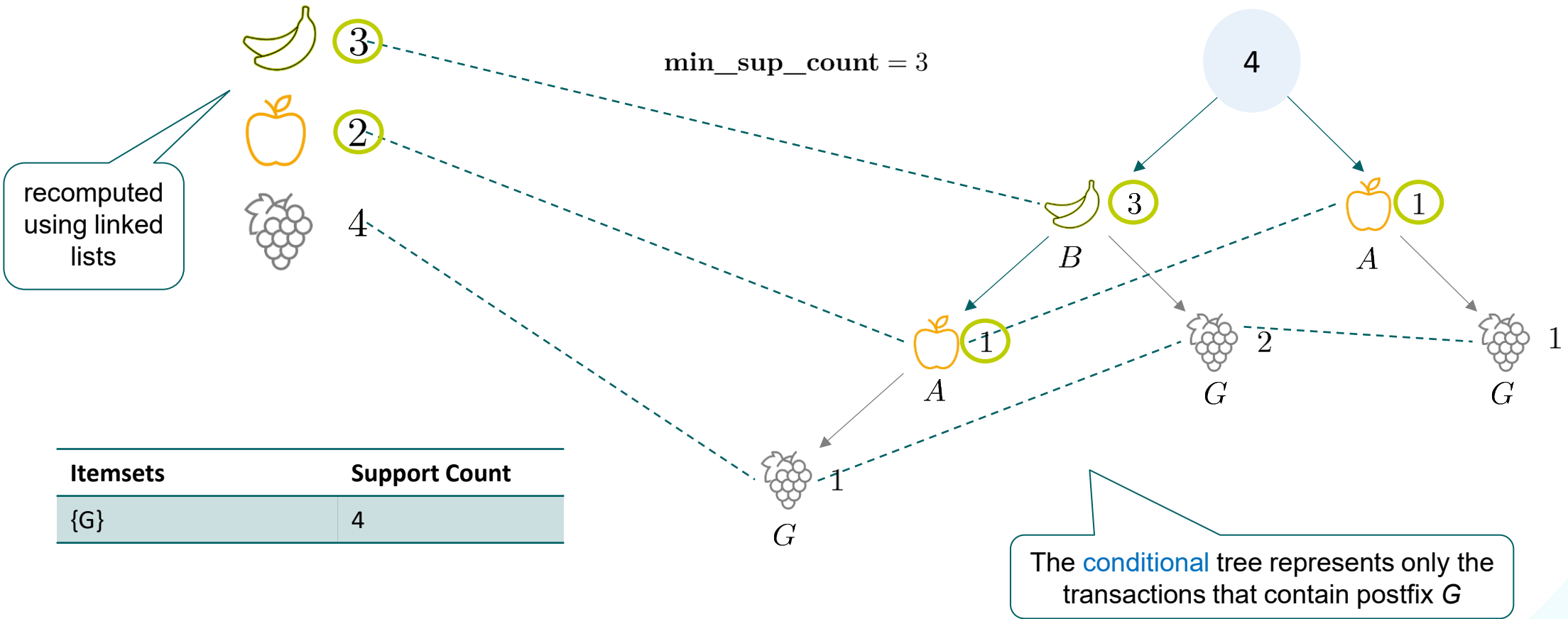


Consider Postfix G

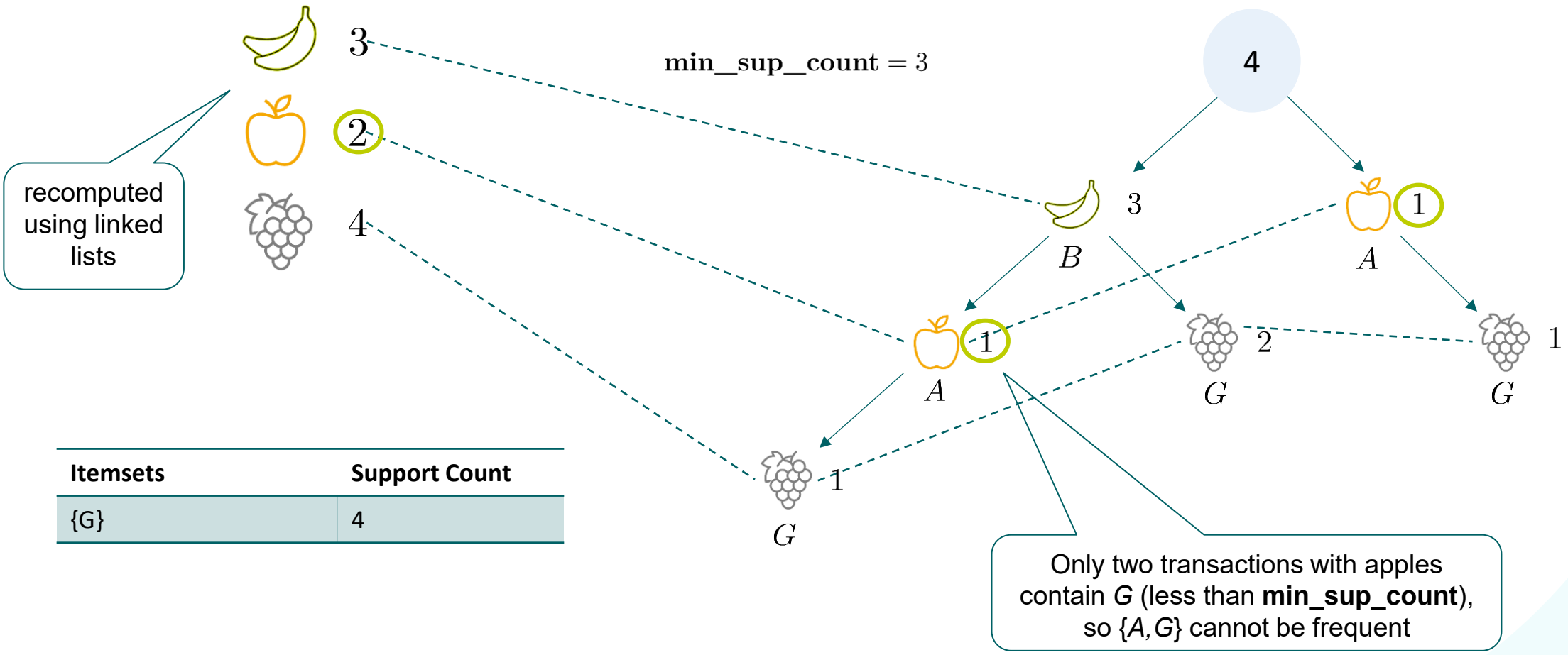


Now only consider **all the paths** leading from the root to G (representing transactions that include G)

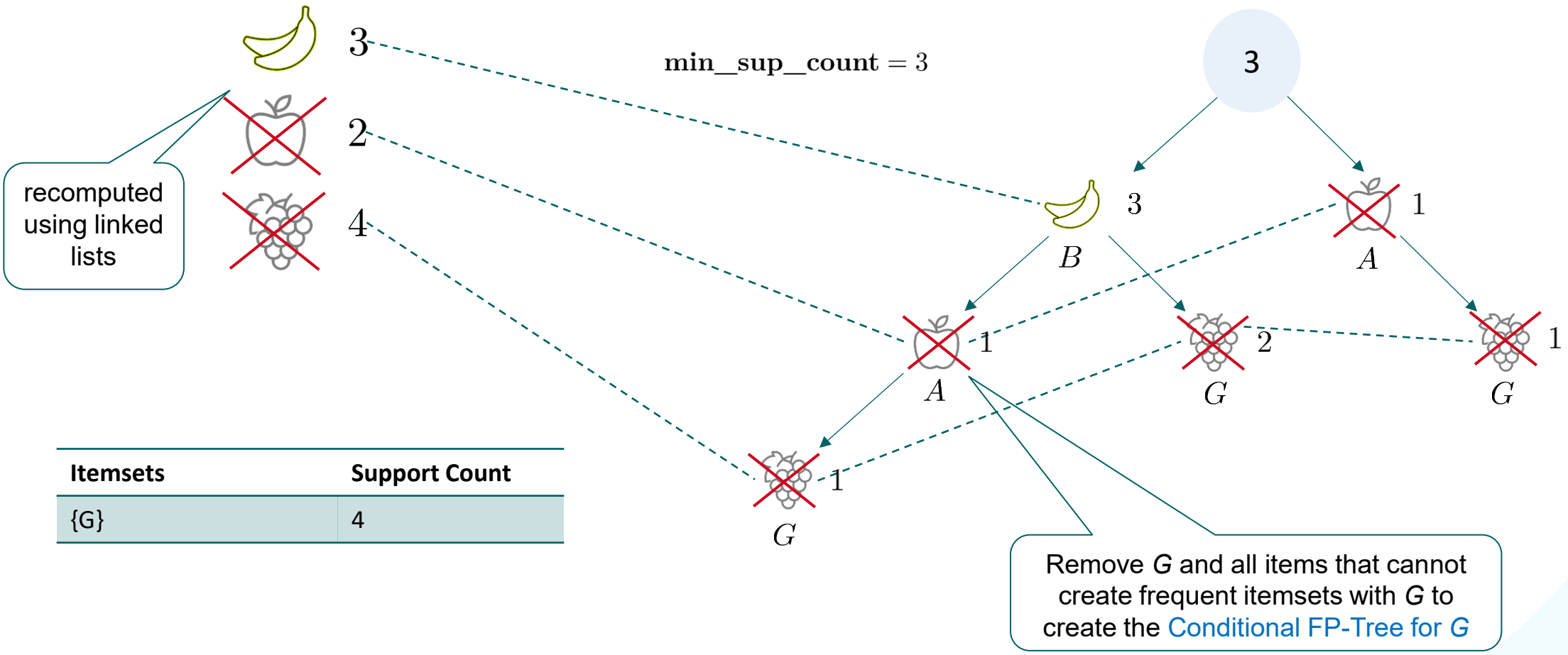
Towards the Conditional FP-Tree for Postfix G



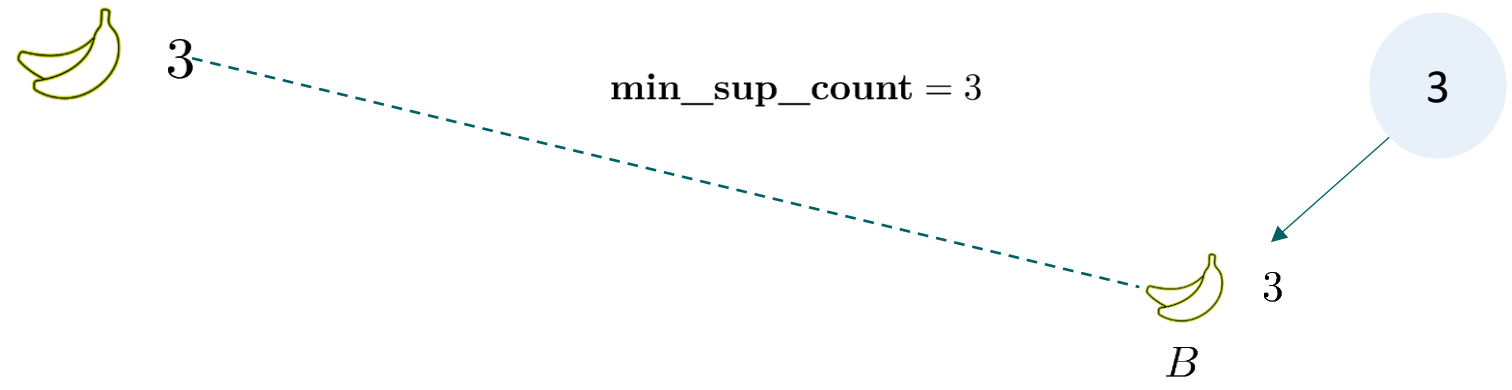
Towards the Conditional FP-Tree for Postfix G



Towards the Conditional FP-Tree for Postfix G



Conditional FP-Tree for Postfix G

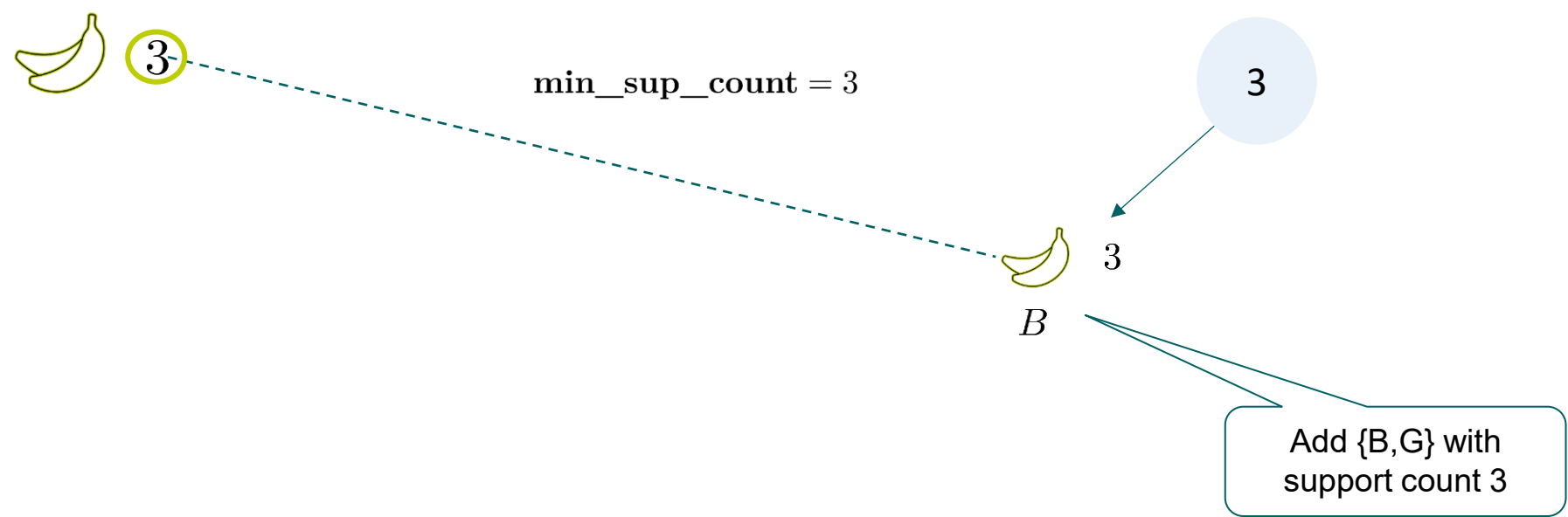


Itemsets	Support Count
{G}	4
{ ..., G }	

recurse

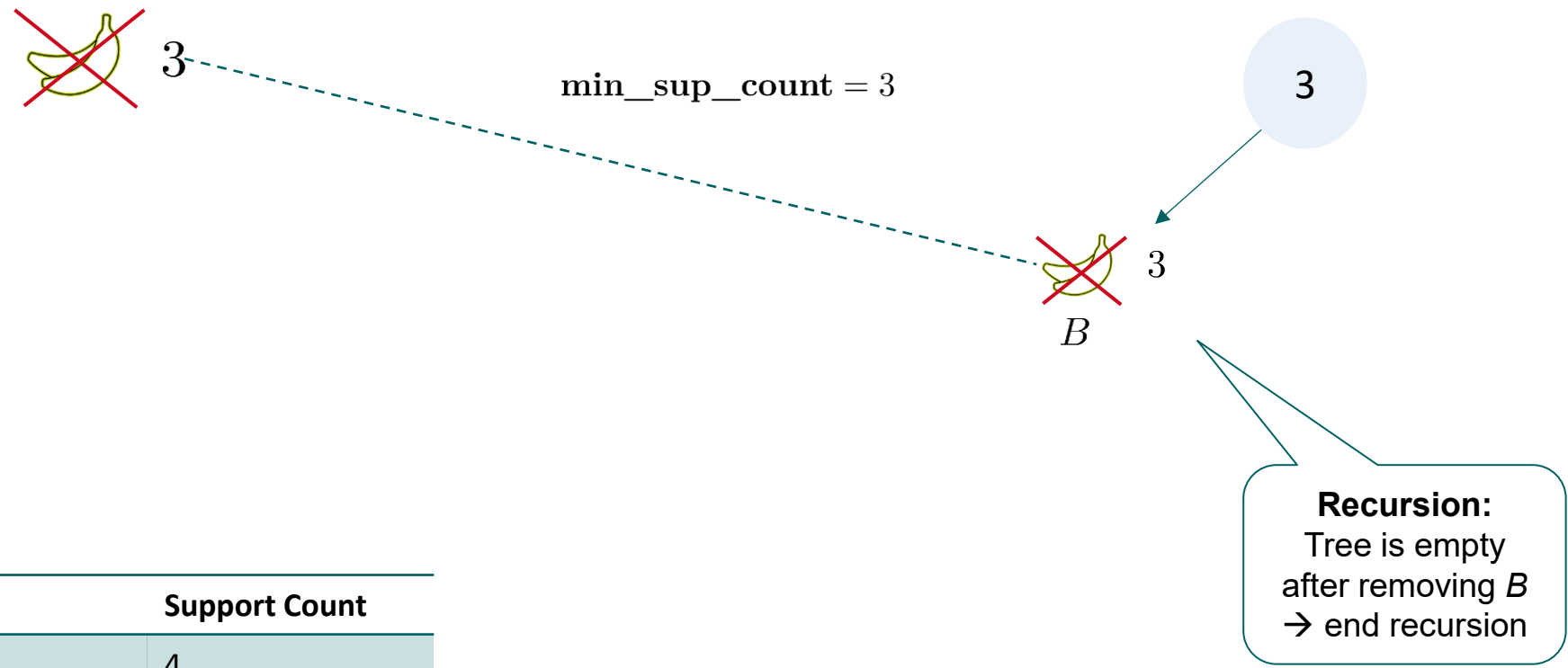
Mine the Conditional FP-Tree for G to find frequent itemsets that contain G (Recursion)

Conditional FP-Tree for Postfix G



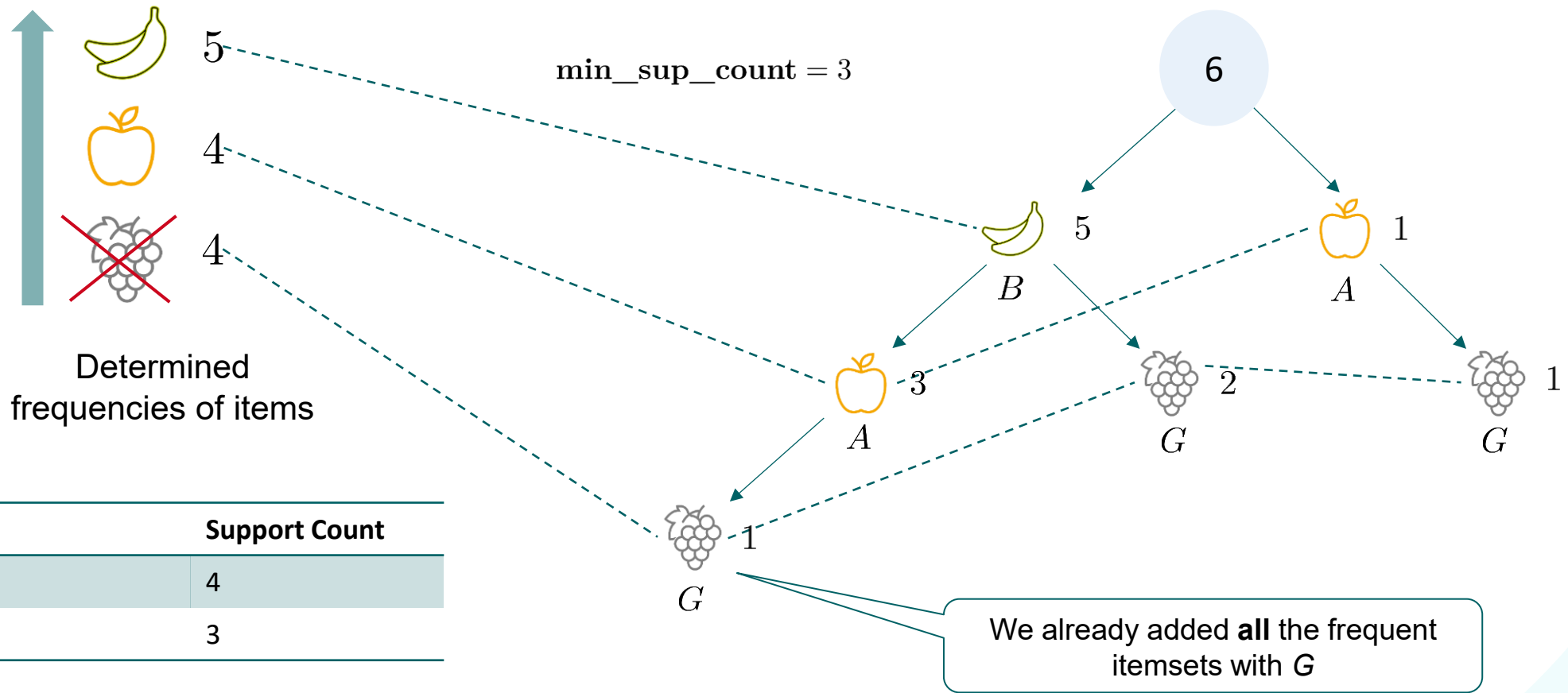
Itemsets	Support Count
{G}	4
{B, G}	3

Conditional FP-Tree for Postfix G



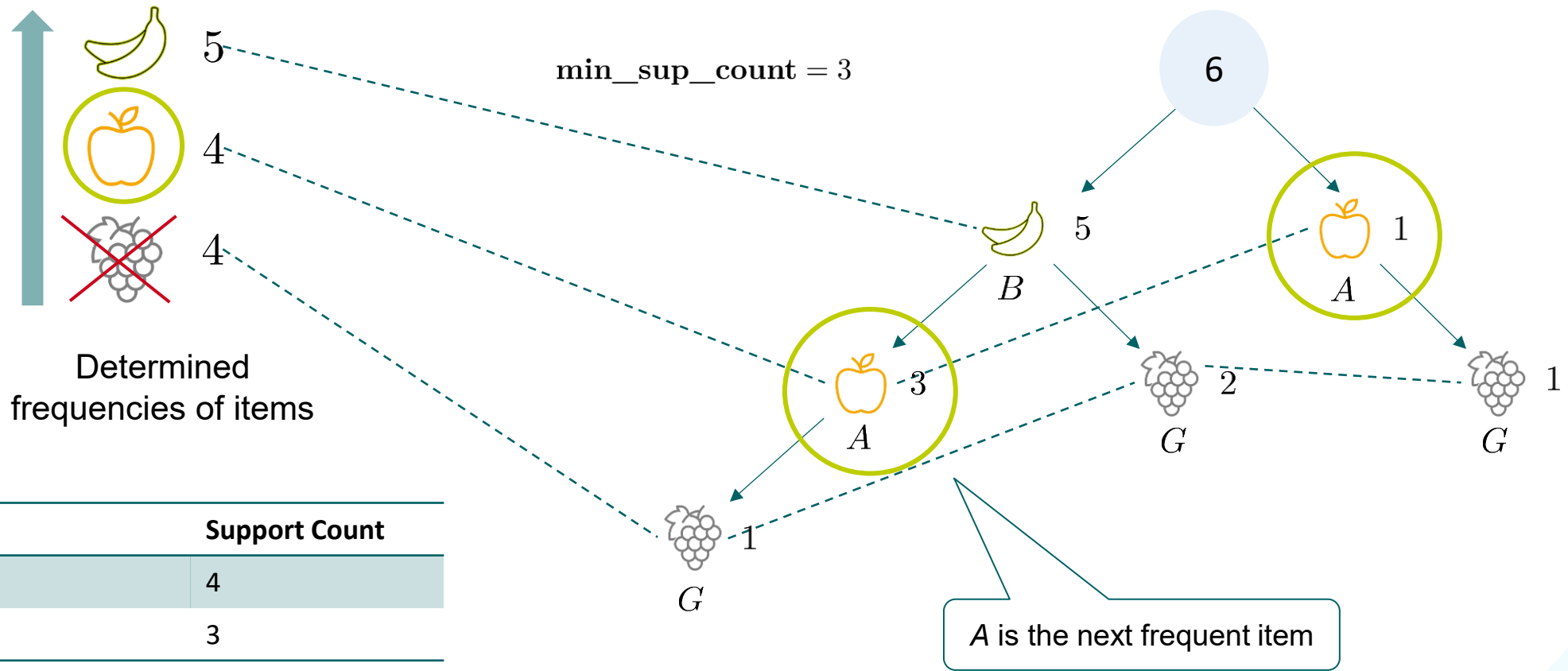
Itemsets	Support Count
{G}	4
{B, G}	3

Consider Postfix A



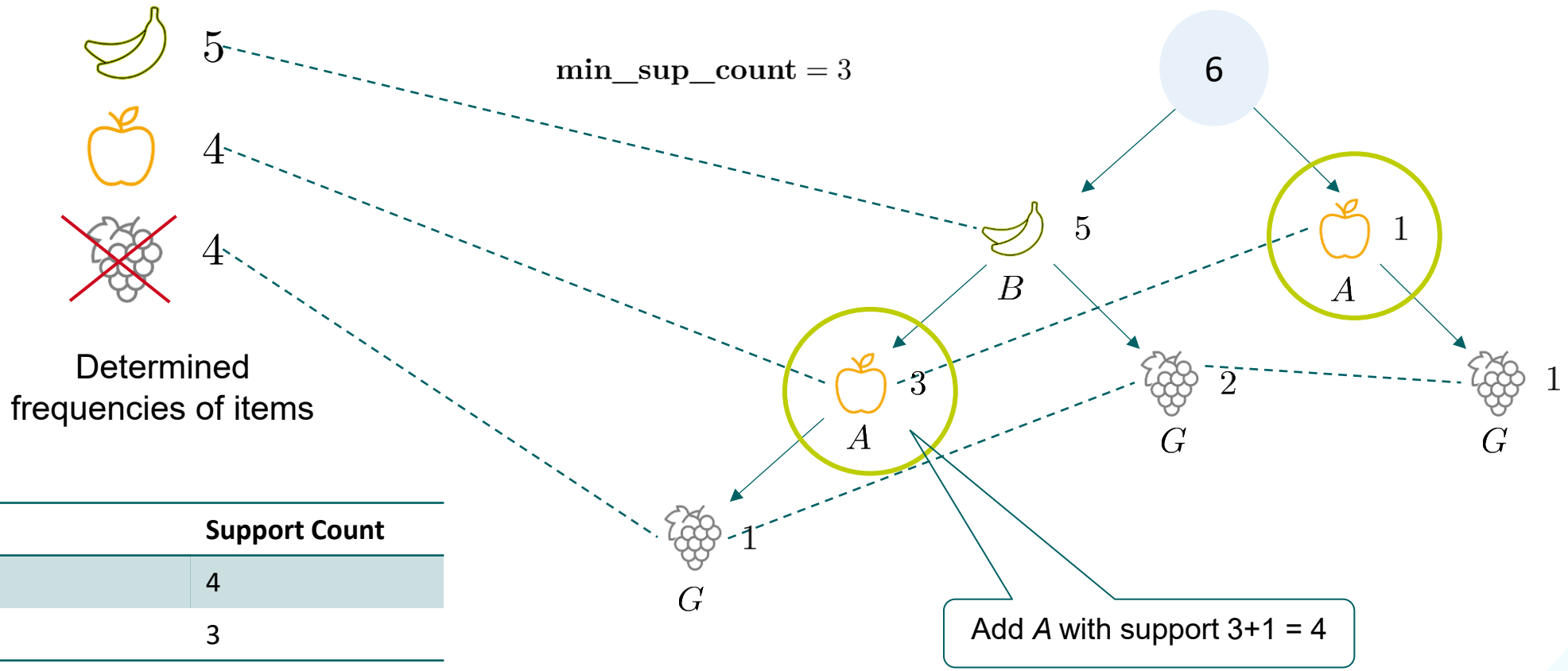
Itemsets	Support Count
{G}	4
{B, G}	3

Consider Postfix A



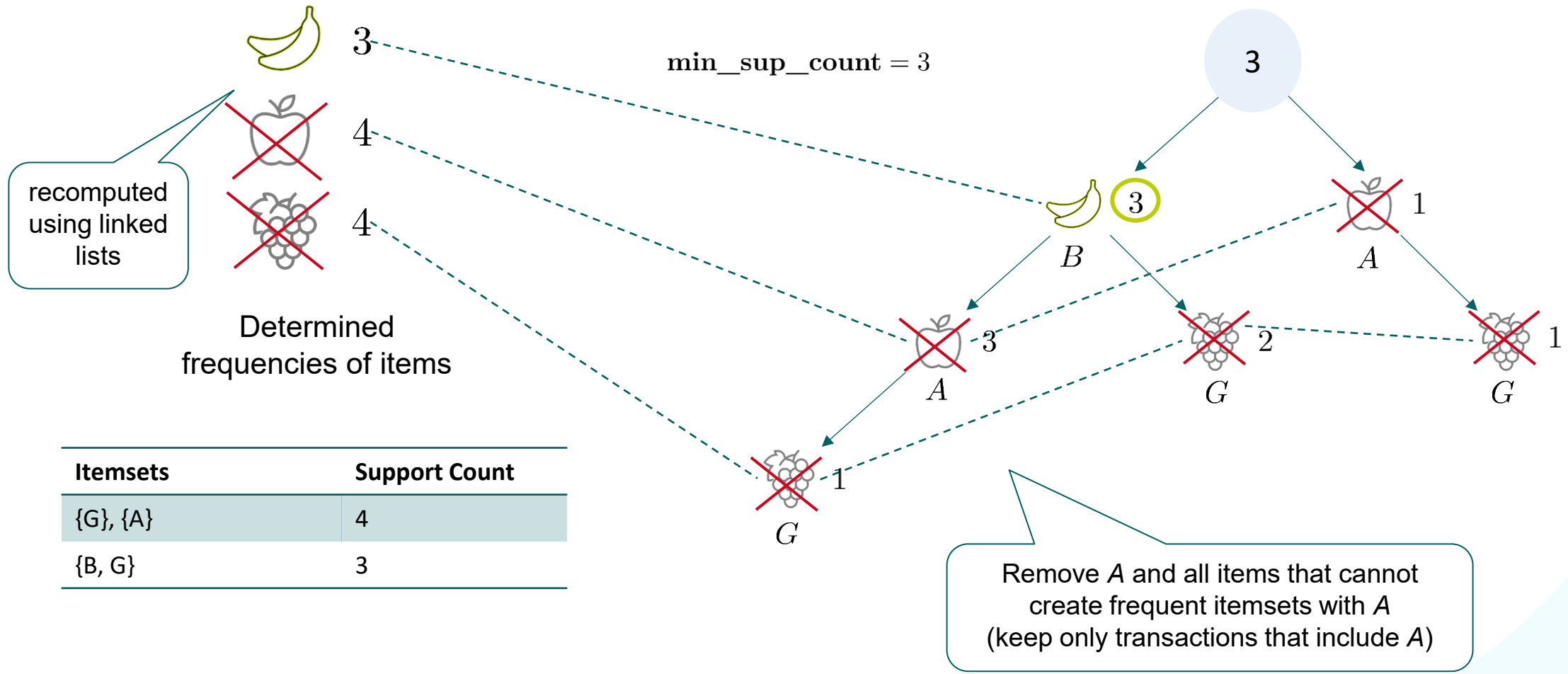
Itemsets	Support Count
{G}	4
{B, G}	3

Consider Postfix A

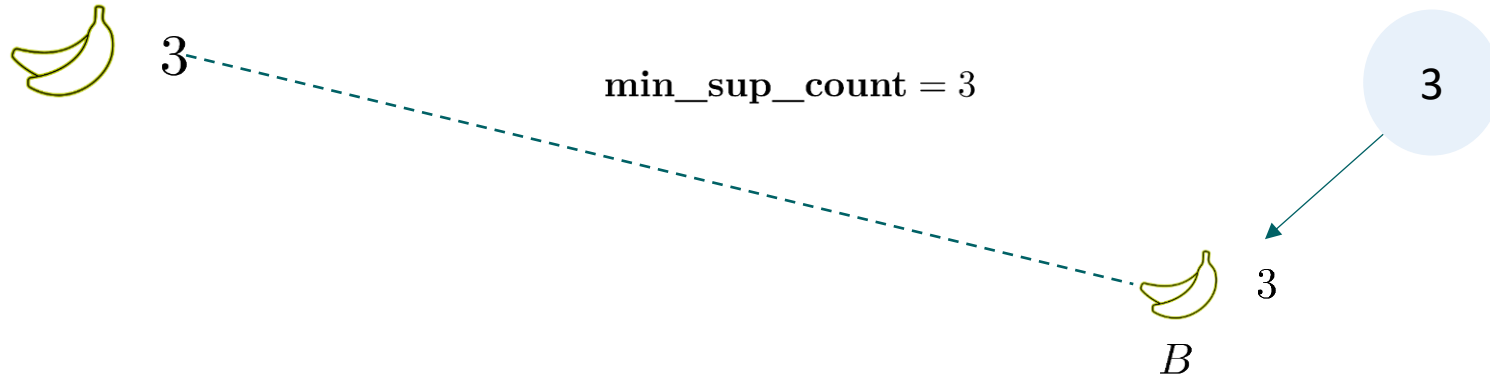


Itemsets	Support Count
{G}, {A}	4
{B, G}	3

Towards the Conditional FP-Tree for Postfix A



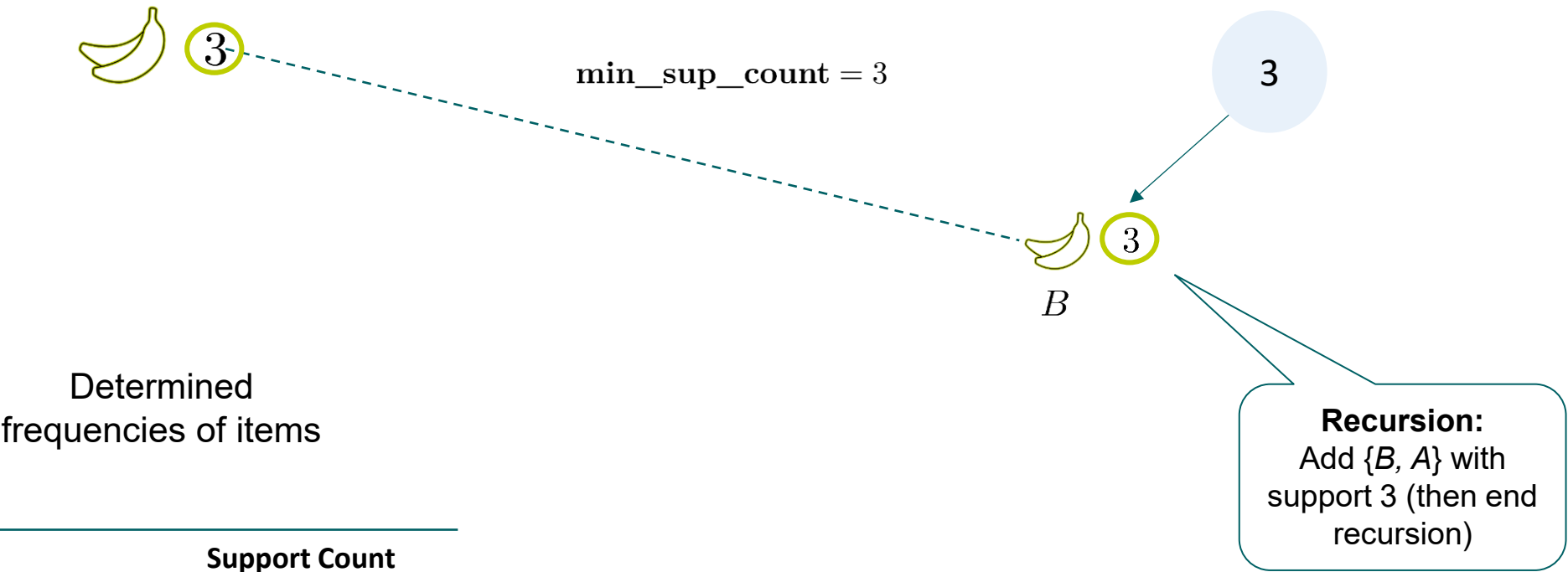
Towards Conditional FP-Tree for Postfix A



Determined
frequencies of items

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

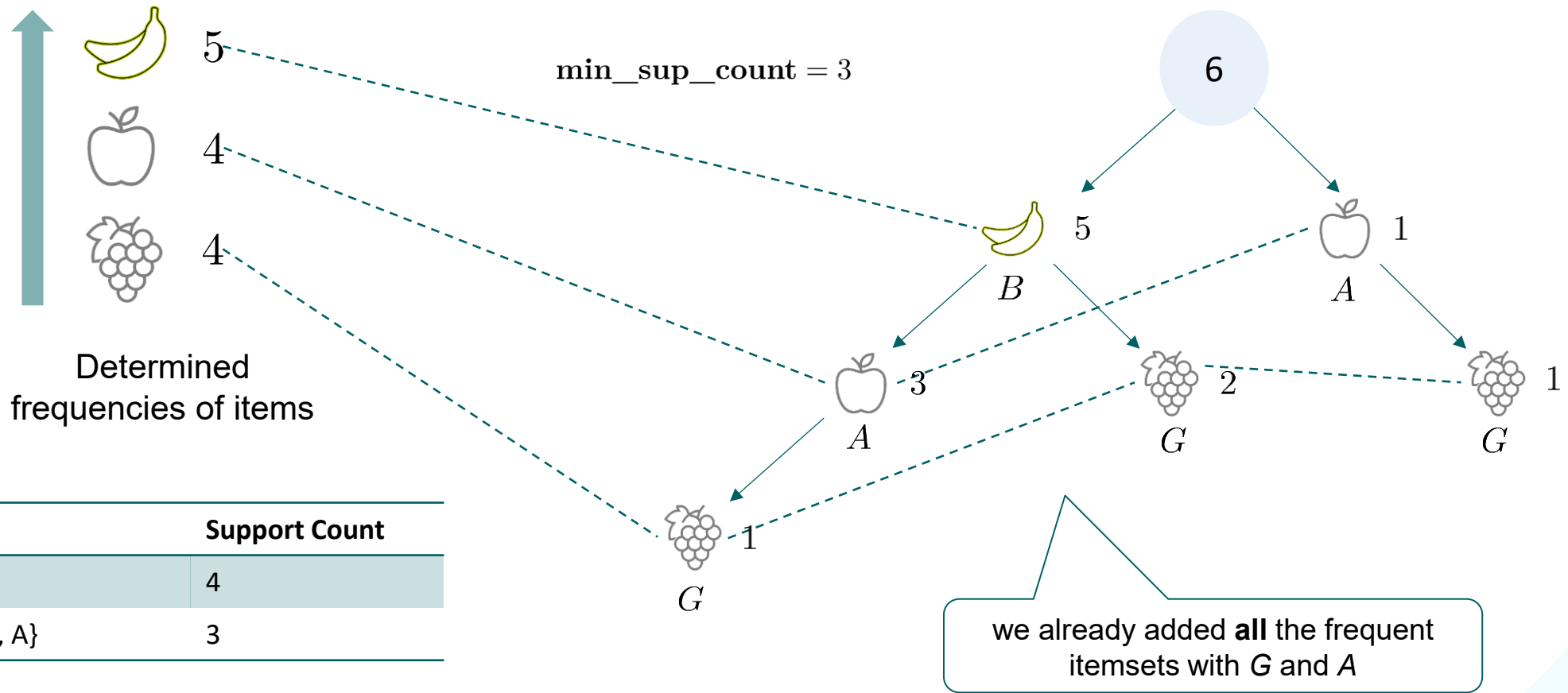
Conditional FP-Tree for Postfix A



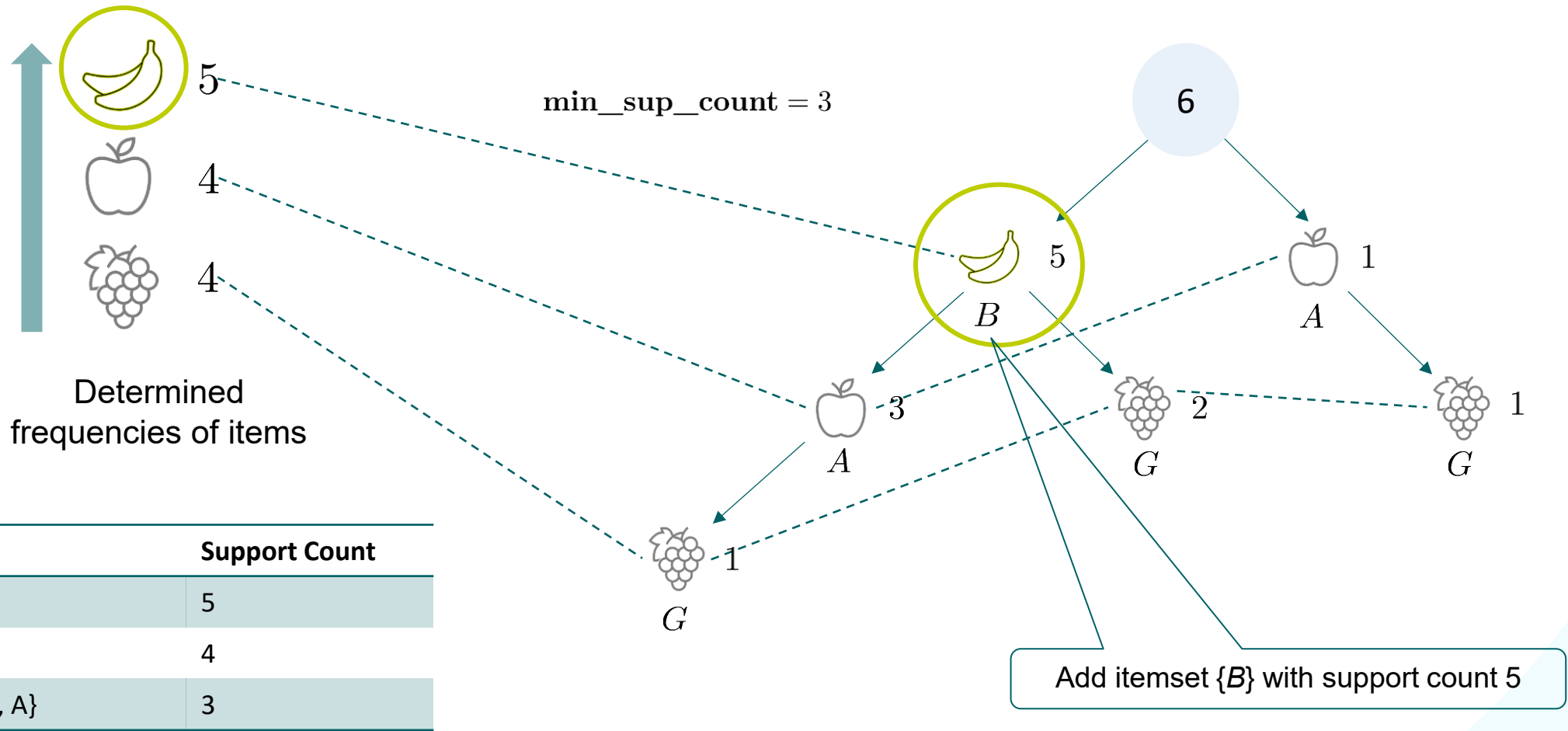
Determined frequencies of items

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

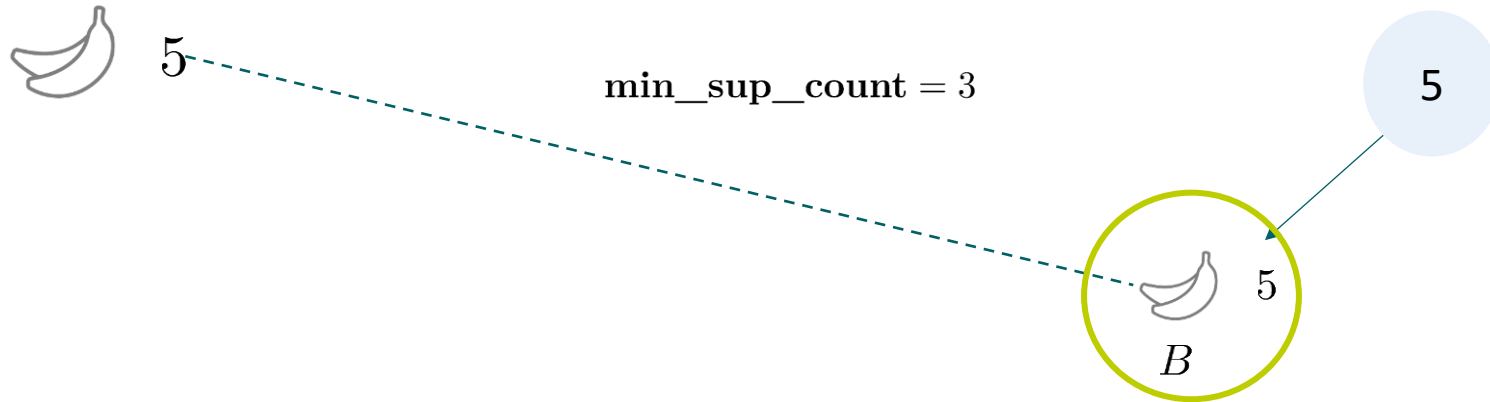
Consider Postfix B



Consider Postfix B



Towards Conditional FP-Tree for Postfix B

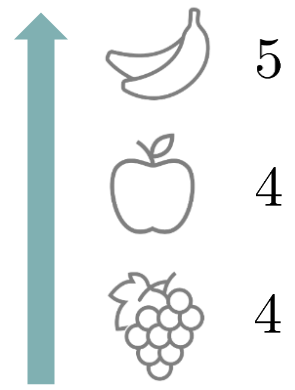


Determined
frequencies of items

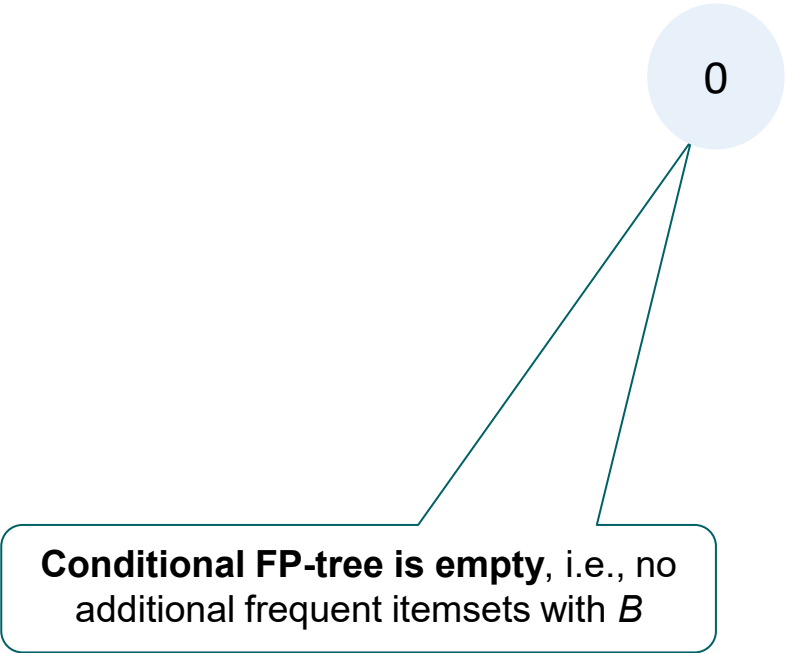
Itemsets	Support Count
{B}	5
{G}, {A}	4
{B, G}, {B, A}	3

Now only consider **all the paths** leading to *B*
(transactions including *B*)

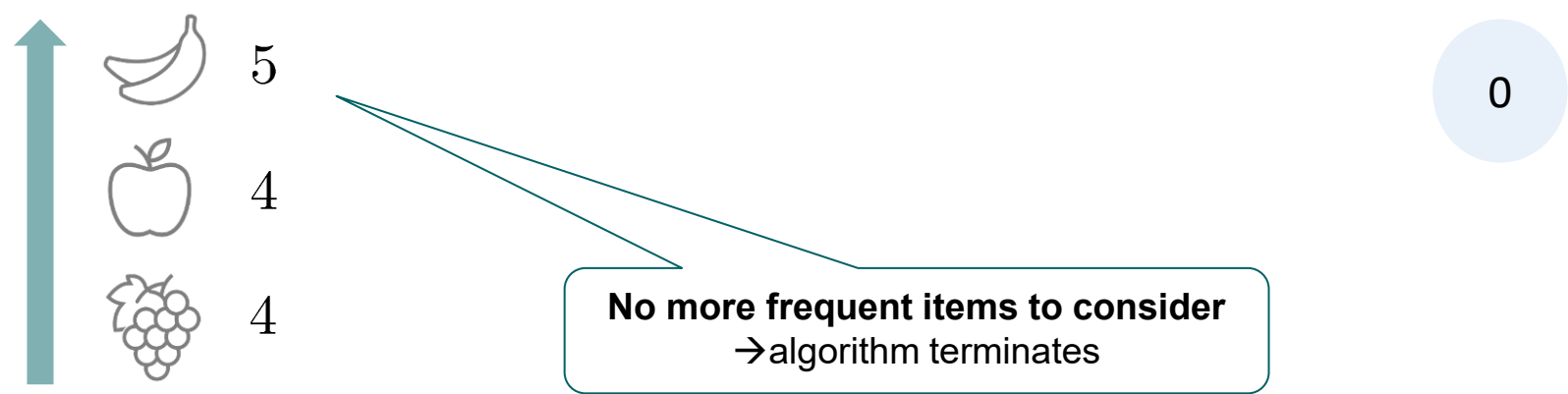
Conditional FP-Tree for Postfix B



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

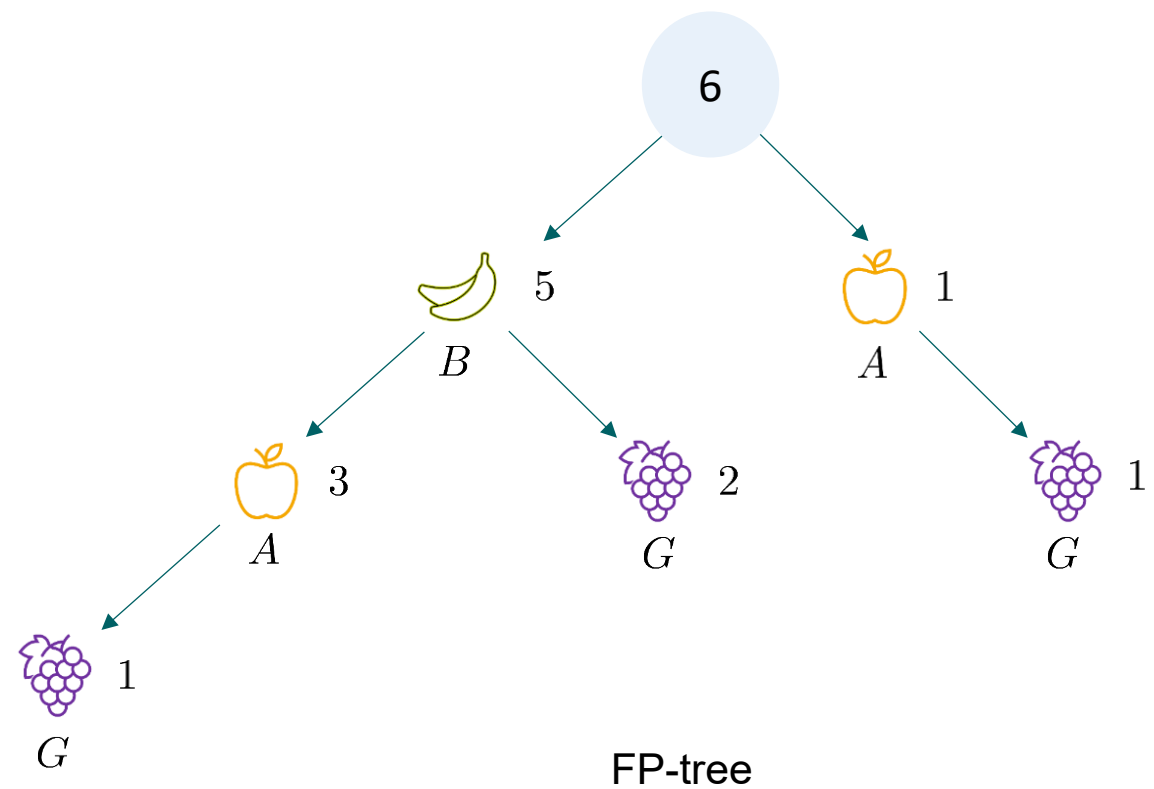


Conditional FP-Tree for Postfix B



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)



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

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



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

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



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

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



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

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



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

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