



Trường Đại học Công nghiệp Thành phố Hồ Chí Minh

Frequency Item Set

Giảng viên: Tiến sĩ Bùi Thanh Hùng
Bộ môn Khoa học dữ liệu
Khoa Công nghệ thông tin
Đại học Công nghiệp TP HCM

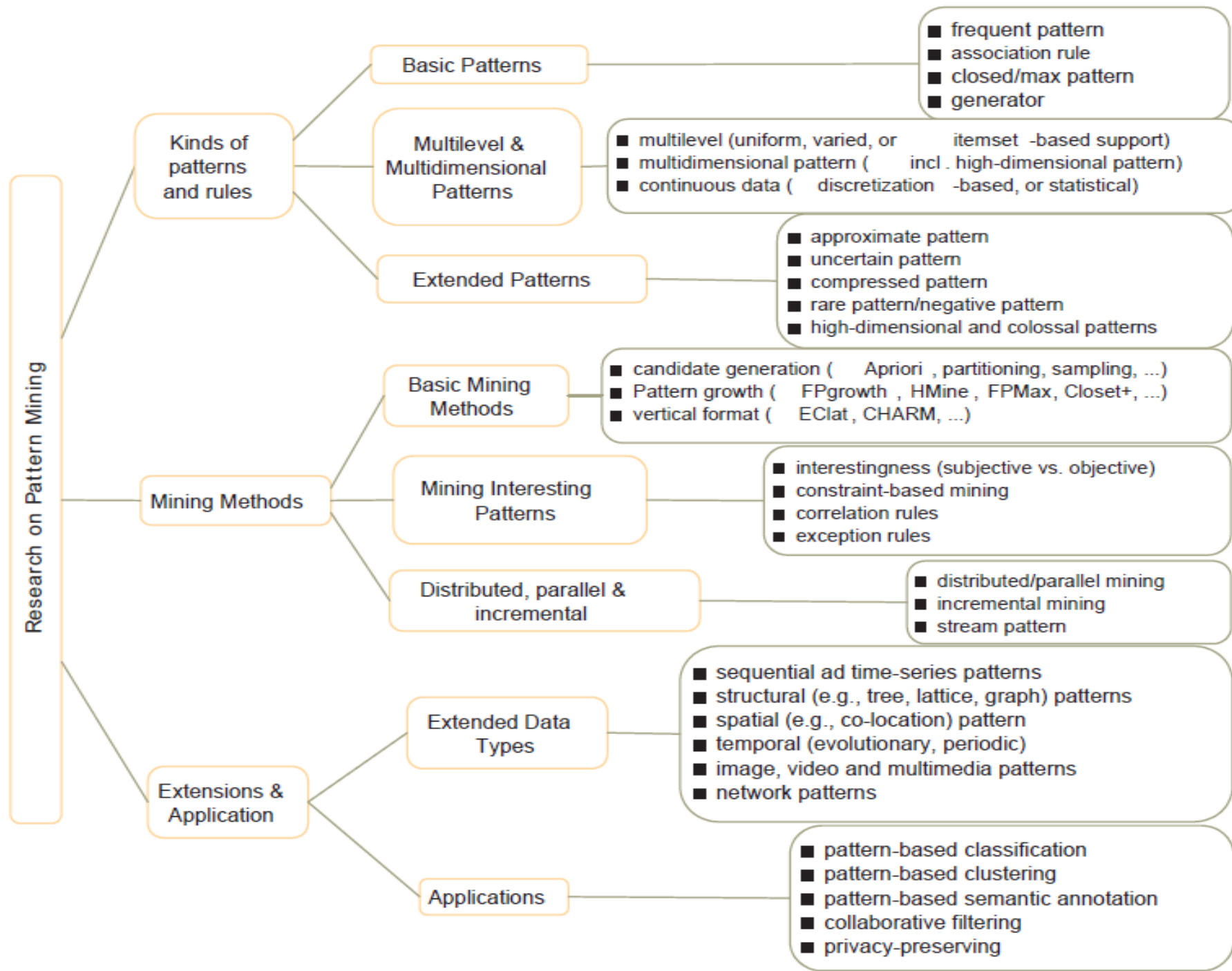
Email: buithanhhung@iuh.edu.vn

Website: <https://sites.google.com/site/hungthanhbui1980/>

Data Mining Techniques

- Frequent Pattern
- Classification
- Clustering

Research on Pattern Mining: A Road Map



Basic Frequent Pattern Mining

- Apriori: A Candidate Generation-and-Test Approach
- Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Mining Close Frequent Patterns and Maxpatterns

Advanced Frequent Pattern Mining

- Pattern Mining in Multi-Level, Multi-Dimensional Space
 - Mining Multi-Level Association
 - Mining Multi-Dimensional Association
 - Mining Quantitative Association Rules
 - Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Mining Compressed or Approximate Patterns

Rule generation

Rule generation is a common task in the mining of frequent patterns. *An association rule is an implication expression of the form $X \rightarrow Y$, where X and Y are disjoint itemsets.* A more concrete example based on consumer behaviour would be $\{\text{Diapers}\} \rightarrow \{\text{Beer}\}$ suggesting that people who buy diapers are also likely to buy beer. To evaluate the "interest" of such an association rule, different metrics have been developed. The current implementation make use of the confidence and lift metrics.

Support metrics

The currently supported metrics for evaluating association rules and setting selection thresholds are listed below. Given a rule " $A \rightarrow C$ ", A stands for antecedent and C stands for consequent.

$$\text{support}(A \rightarrow C) = \text{support}(A \cup C), \quad \text{range: } [0, 1]$$

R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in large databases. In Proc. of the ACM SIGMOD Int'l Conference on Management of Data, pages 207-216, Washington D.C., May 1993

Support metrics

The currently supported metrics for evaluating association rules and setting selection thresholds are listed below. Given a rule " $A \rightarrow C$ ", A stands for antecedent and C stands for consequent.

The support metric is defined for itemsets, not association rules. The table produced by the association rule mining algorithm contains three different support metrics: 'antecedent support', 'consequent support', and 'support'. Here, 'antecedent support' computes the proportion of transactions that contain the antecedent A , and 'consequent support' computes the support for the itemset of the consequent C . The 'support' metric then computes the support of the combined itemset $A \cup C$ -- note that 'support' depends on 'antecedent support' and 'consequent support' via $\min(\text{'antecedent support'}, \text{'consequent support'})$.

Typically, support is used to measure the abundance or frequency (often interpreted as significance or importance) of an itemset in a database. We refer to an itemset as a "frequent itemset" if you support is larger than a specified minimum-support threshold. Note that in general, due to the *downward closure* property, all subsets of a frequent itemset are also frequent.

$$\text{support}(A \rightarrow C) = \text{support}(A \cup C), \quad \text{range: } [0, 1]$$

Confidence

The confidence of a rule $A \rightarrow C$ is the probability of seeing the consequent in a transaction given that it also contains the antecedent. Note that the metric is not symmetric or directed; for instance, the confidence for $A \rightarrow C$ is different than the confidence for $C \rightarrow A$. The confidence is 1 (maximal) for a rule $A \rightarrow C$ if the consequent and antecedent always occur together.

$$\text{confidence}(A \rightarrow C) = \frac{\text{support}(A \rightarrow C)}{\text{support}(A)}, \quad \text{range: } [0, 1]$$

R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in large databases. In Proc. of the ACM SIGMOD Int'l Conference on Management of Data, pages 207-216, Washington D.C., May 1993

lift metric

The lift metric is commonly used to measure how much more often the antecedent and consequent of a rule $A \rightarrow C$ occur together than we would expect if they were statistically independent. If A and C are independent, the Lift score will be exactly 1.

$$\text{lift}(A \rightarrow C) = \frac{\text{confidence}(A \rightarrow C)}{\text{support}(C)}, \quad \text{range: } [0, \infty]$$

S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data, 1996

leverage metric

Leverage computes the difference between the observed frequency of A and C appearing together and the frequency that would be expected if A and C were independent. A leverage value of 0 indicates independence.

$$\text{leverage}(A \rightarrow C) = \text{support}(A \rightarrow C) - \text{support}(A) \times \text{support}(C), \quad \text{range: } [-1, 1]$$

Piatetsky-Shapiro, G., Discovery, analysis, and presentation of strong rules. Knowledge Discovery in Databases, 1991: p. 229-248.

conviction metric

A high conviction value means that the consequent is highly depending on the antecedent. For instance, in the case of a perfect confidence score, the denominator becomes 0 (due to $1 - 1$) for which the conviction score is defined as 'inf'. Similar to lift, if items are independent, the conviction is 1.

$$\text{conviction}(A \rightarrow C) = \frac{1 - \text{support}(C)}{1 - \text{confidence}(A \rightarrow C)}, \quad \text{range: } [0, \infty]$$

Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Turk. Dynamic itemset counting and implication rules for market basket data. In SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, pages 255-264, Tucson, Arizona, USA, May 1997

support(A->C) = **support**(A+C) [aka 'support'], range: [0, 1]

confidence(A->C) = **support**(A+C) / **support**(A), range: [0, 1]

lift(A->C) = **confidence**(A->C) / **support**(C), range: [0, inf]

leverage(A->C) = **support**(A->C) - **support**(A)***support**(C), range: [-1, 1]

conviction = [1 - **support**(C)] / [1 - **confidence**(A->C)], range: [0, inf]

Reducing Number of Candidates

- **Apriori principle:**
 - If an itemset is frequent, then all of its subsets must also be frequent
- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

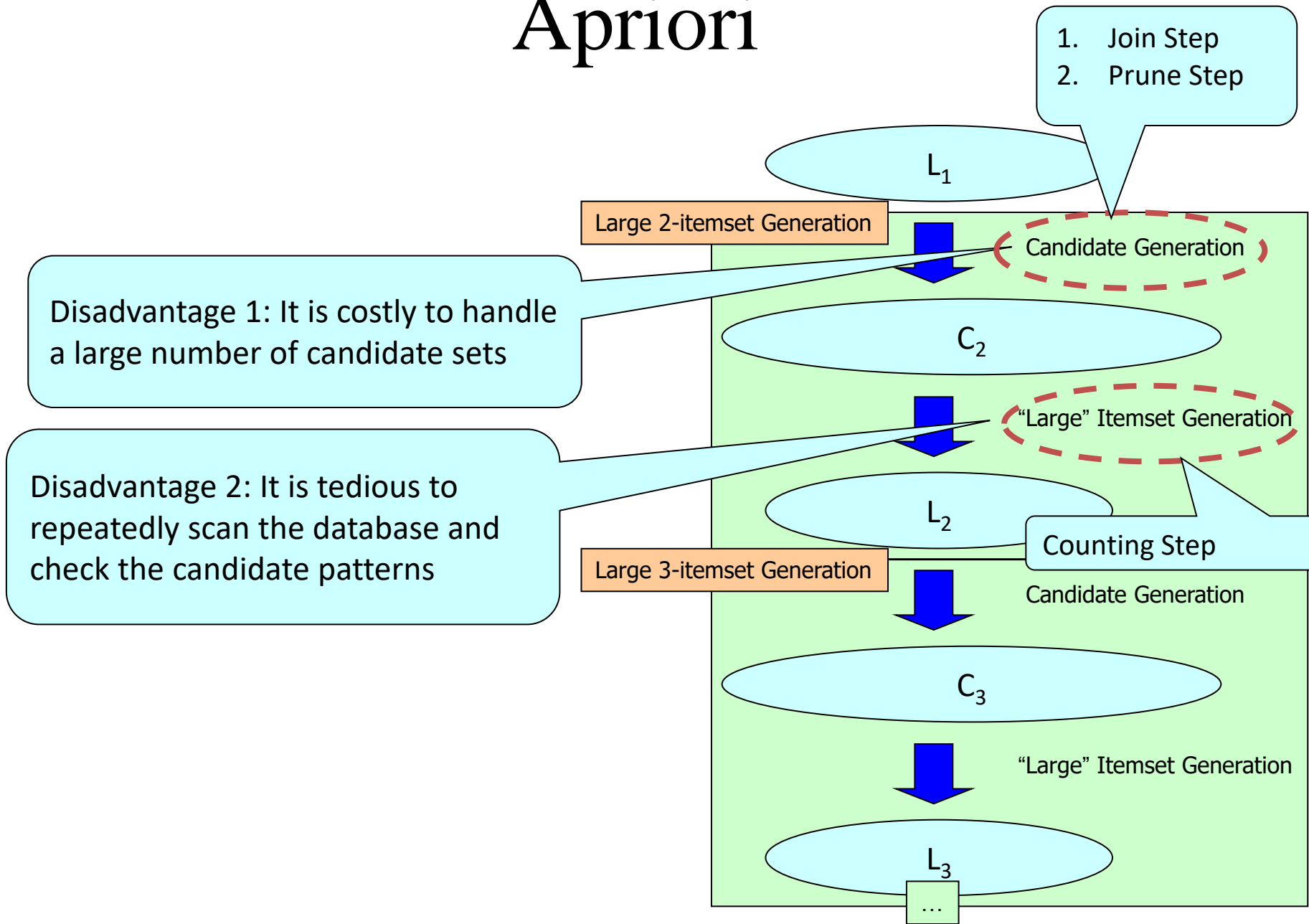
- Support of an itemset never exceeds the support of its subsets
- This is known as the **anti-monotone** property of support

Apriori Algorithm

Method:

- Let $k=1$
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are identified
 - **Generate** length $(k+1)$ candidate itemsets from length k frequent itemsets
 - **Prune candidate** itemsets containing subsets of length k that are infrequent
 - **Count** the **support** of each candidate by scanning the DB
 - **Eliminate candidates** that are **infrequent**, leaving only those that are frequent

Apriori



Max-patterns

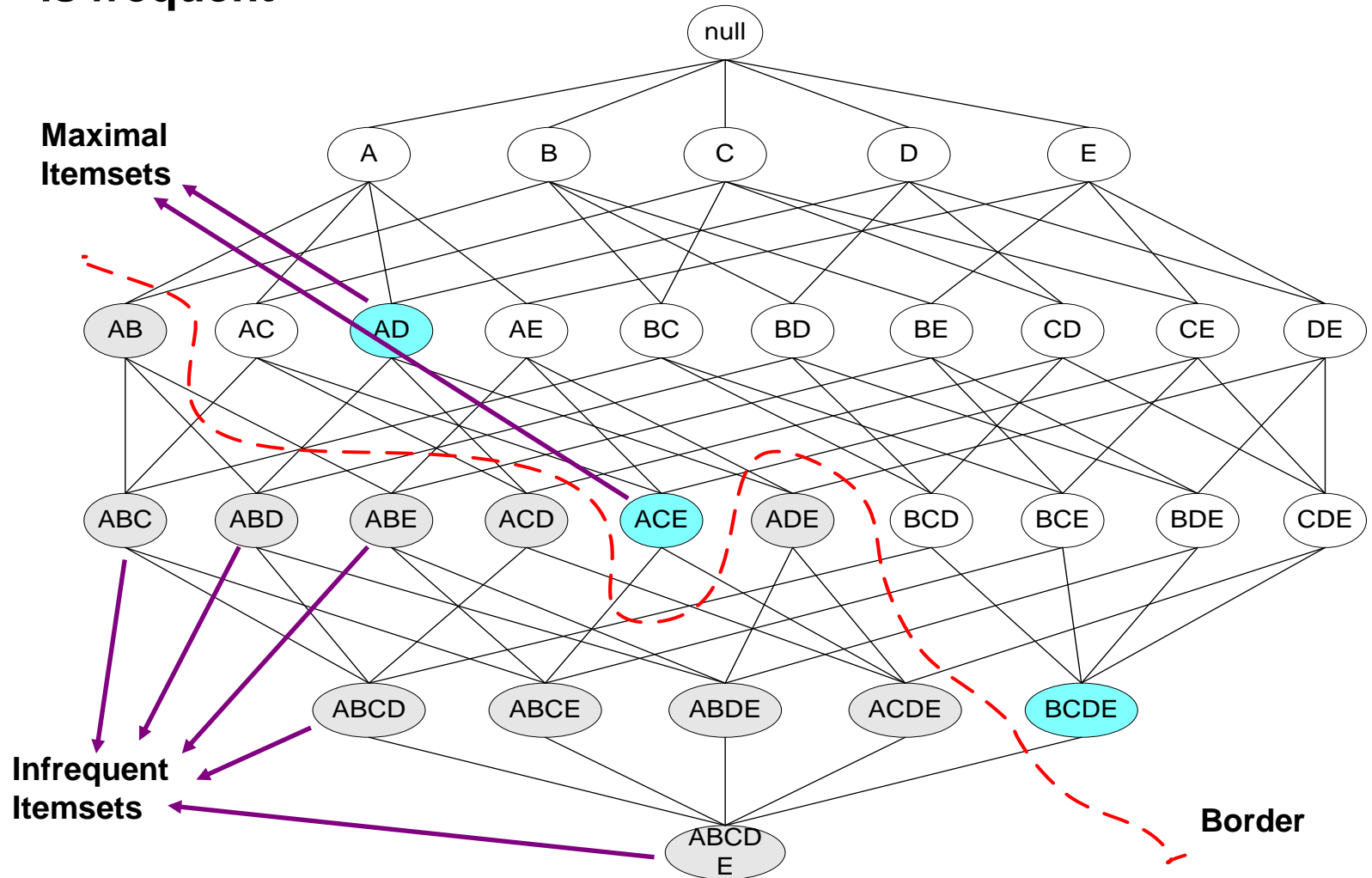
- Max-pattern: frequent patterns without proper frequent super pattern
 - BCDE, ACD are max-patterns
 - BCD is not a max-pattern

Min_sup=2

Tid	Items
10	A,B,C,D,E
20	B,C,D,E,
30	A,C,D,F

Maximal Frequent Itemset

An itemset is maximal frequent if none of its immediate supersets is frequent



A Simple Example

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

Here is a database that has 5 transactions: A, B, C, D, E.

Let the min support = 2.

Find all frequent max itemsets using Apriori algorithm.

A Simple Example

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

Here is a database that has 5 transactions: A, B, C, D, E.

Let the min support = 2.

Find all frequent max itemsets using Apriori algorithm.

1st scan: count support

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

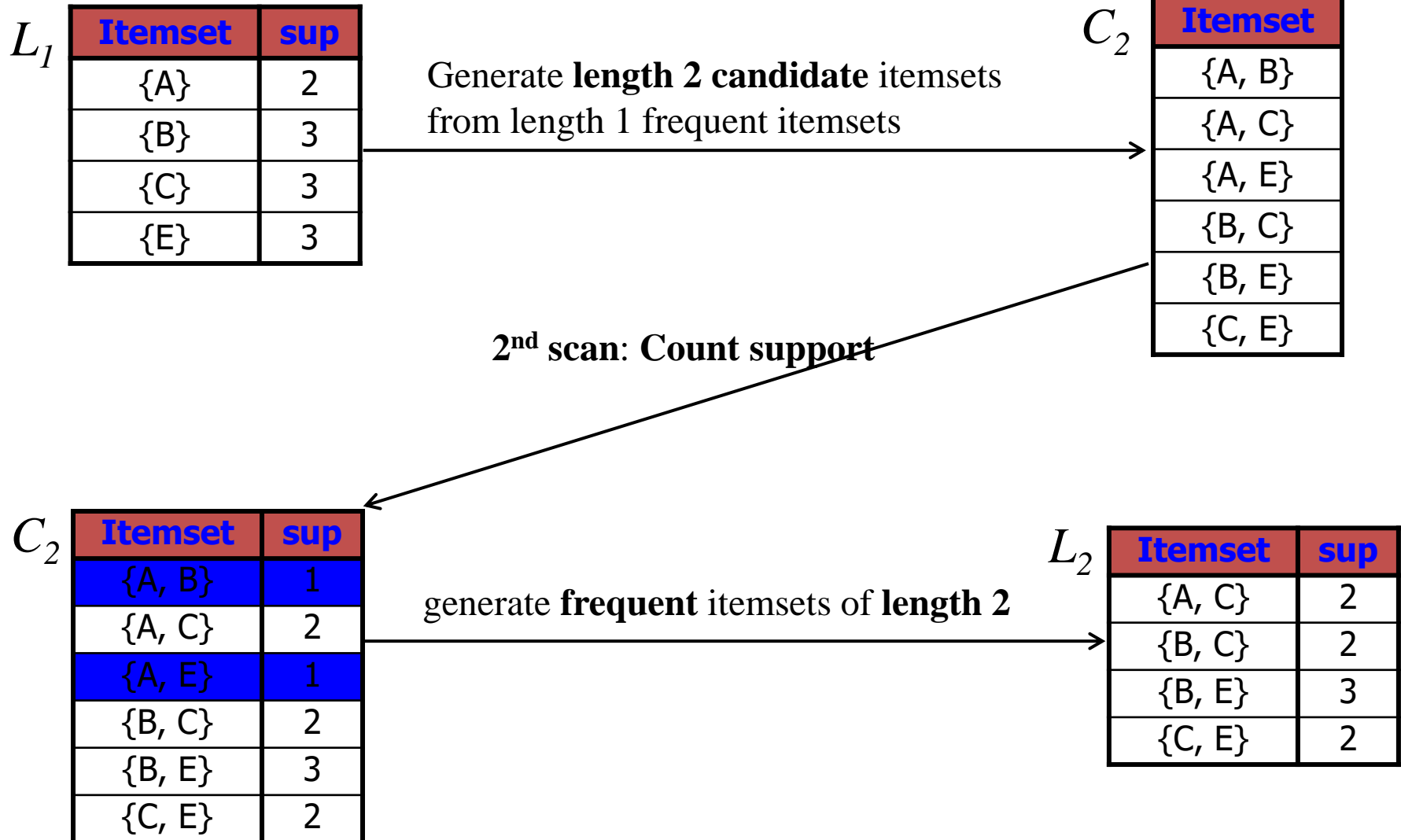
Generate **frequent** itemsets of length 1

Eliminate candidates that are infrequent

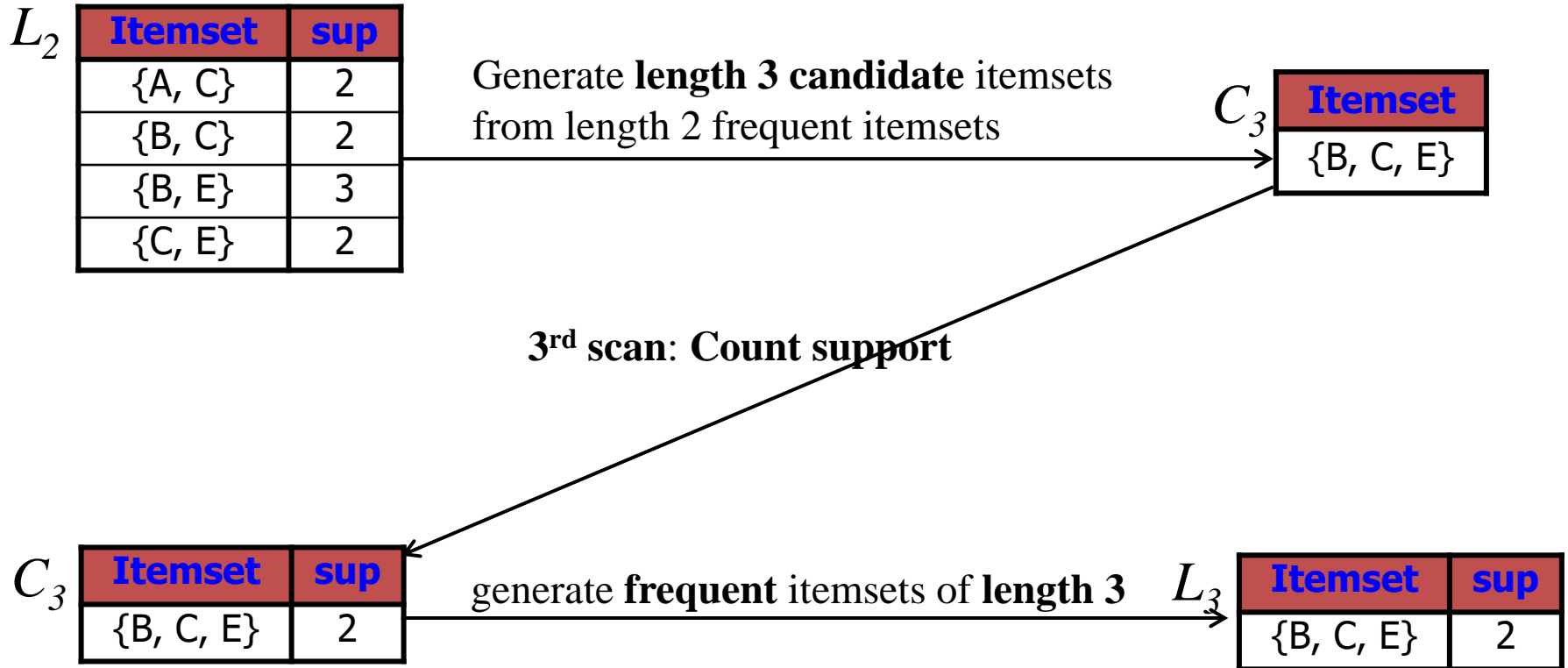
L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

A Simple Example

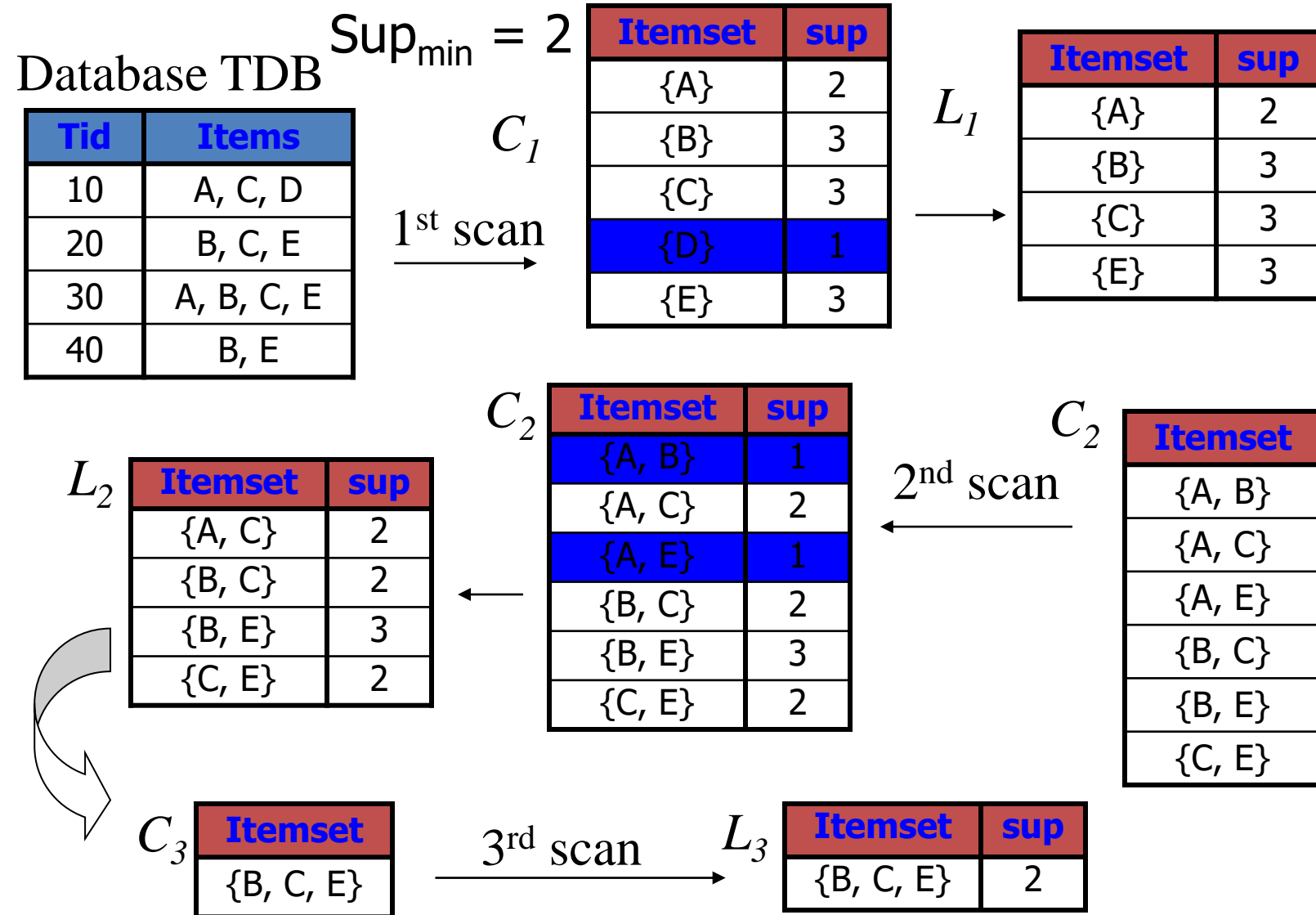


A Simple Example

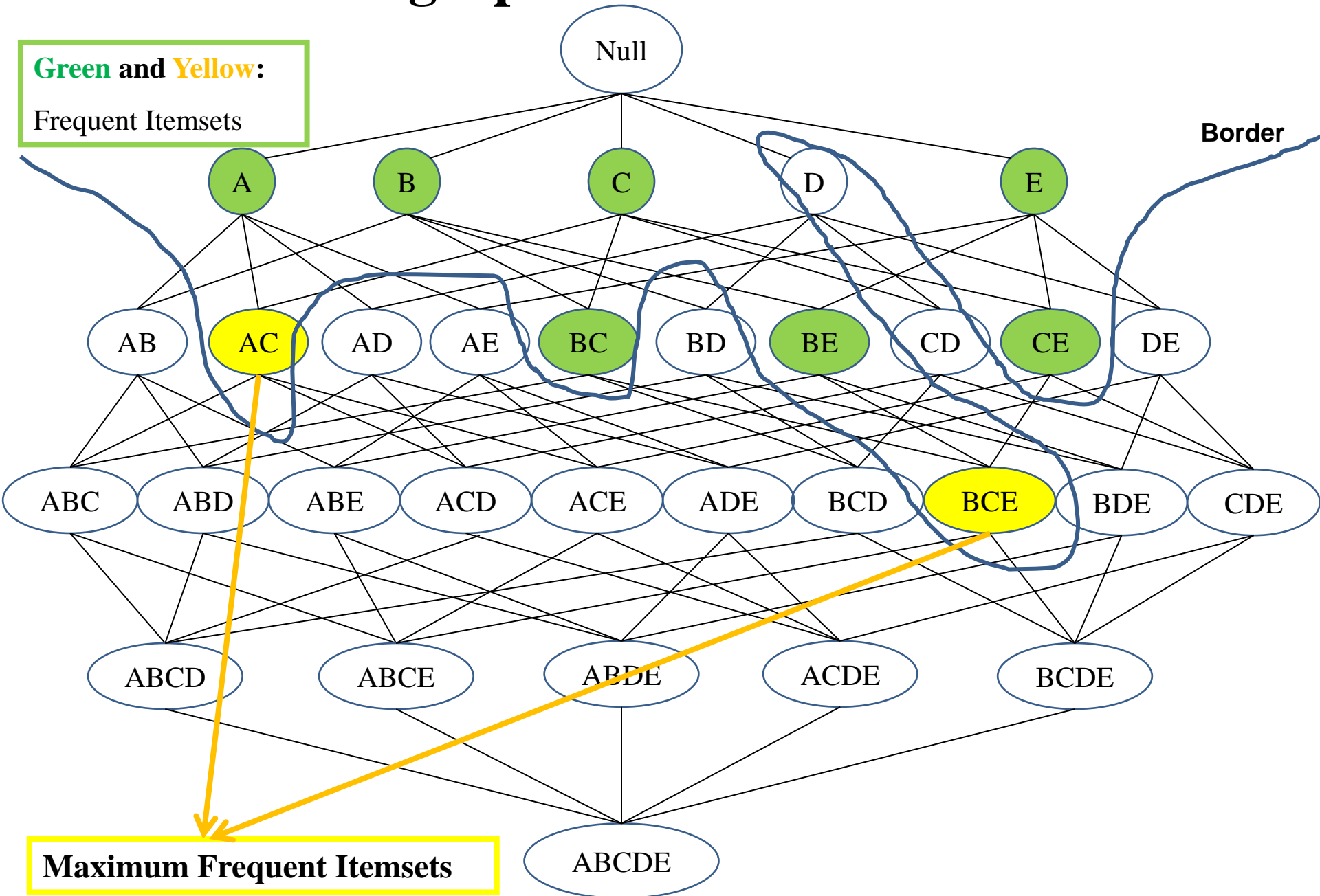


The algorithm stops here

A Simple Example



Draw a graph to illustrate the result



FP-growth Algorithm

- Scan the database once to store all essential information in a data structure called **FP-tree**(Frequent Pattern Tree)
- The FP-tree is concise and is used in directly generating large itemsets
- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets

FP-growth Algorithm

- **Step 1:** Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
- **Step 2:** Construct the FP-tree from the above data
- **Step 3:** From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
- **Step 4:** Determine the frequent patterns.

A Simple Example of FP-tree

TID	Items
1	A, B
2	B, C, D
3	A, C, D, E
4	A, D, E
5	A, B, C

Problem:

Find all frequent itemsets with support ≥ 2)

FP-growth Algorithm

- **Step 1:** Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
- **Step 2:** Construct the FP-tree from the above data
- **Step 3:** From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
- **Step 4:** Determine the frequent patterns.

- **Step 1:** Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
- **Step 2:** Construct the FP-tree from the above data
- **Step 3:** From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
- **Step 4:** Determine the frequent patterns.

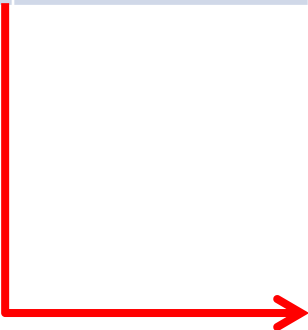
TID	Items
1	A, B
2	B, C, D
3	A, C, D, E
4	A, D, E
5	A, B, C

A Simple Example: Step 1

TID	Items
1	A, B
2	B, C, D
3	A, C, D, E
4	A, D, E
5	A, B, C

Item	Frequency
A	4
B	3
C	3
D	3
E	2

Threshold = 2



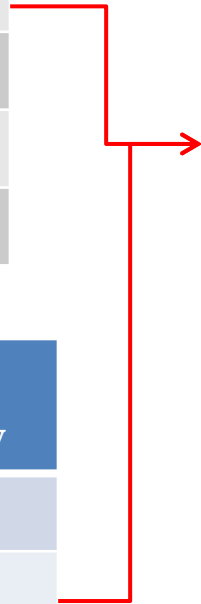
Item	Ordered Frequency
A	4
B	3
C	3
D	3
E	2

A Simple Example: **Step 1**

TID	Items
1	A, B
2	B, C, D
3	A, C, D, E
4	A, D, E
5	A, B, C

TID	Items	(Ordered) Frequent Items
1	A, B	A, B
2	B, C, D	B, C, D
3	A, C, D, E	A, C, D, E
4	A, D, E	A, D, E
5	A, B, C	A, B, C

Item	Ordered Frequency
A	4
B	3
C	3
D	3
E	2

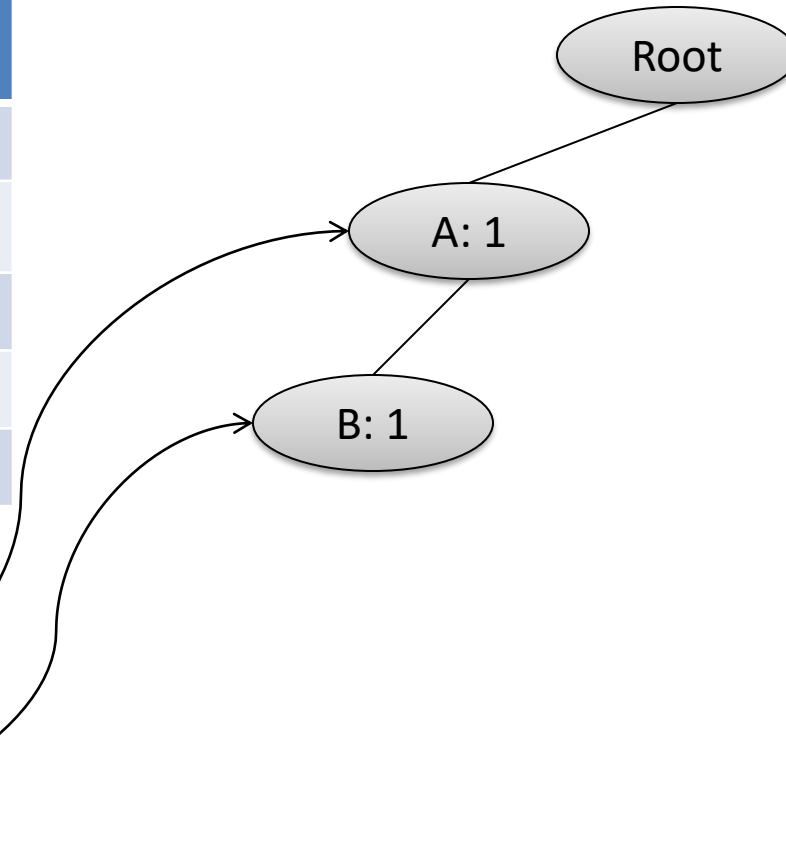


FP-growth Algorithm

- **Step 1:** Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
- **Step 2:** Construct the FP-tree from the above data.
- **Step 3:** From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
- **Step 4:** Determine the frequent patterns.

Step 2: Construct the FP-tree from the above data

TID	Items	(Ordered) Frequent Items
1	A, B	A, B
2	B, C, D	B, C, D
3	A, C, D, E	A, C, D, E
4	A, D, E	A, D, E
5	A, B, C	A, B, C

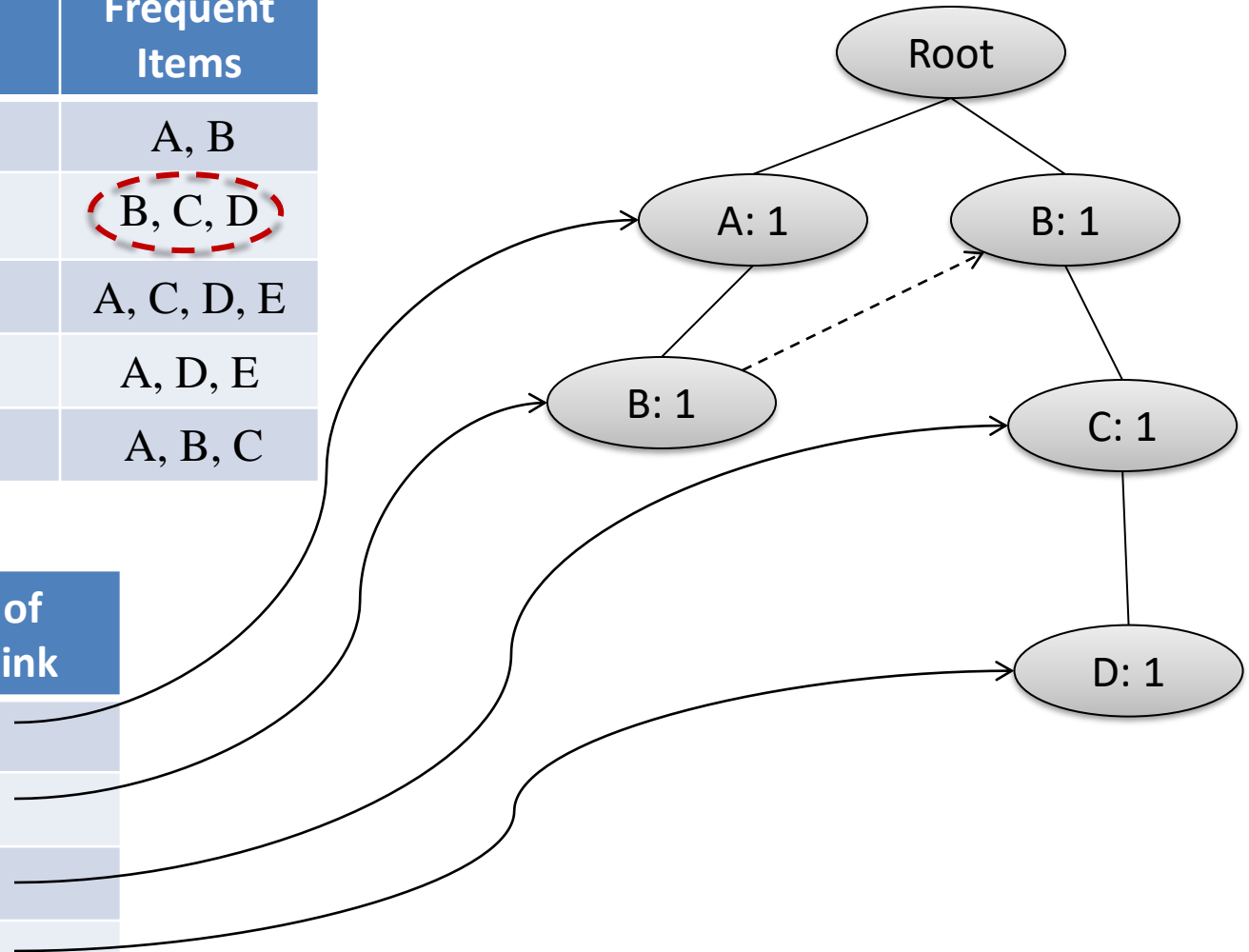


Item	Head of node-link
A	
B	
C	
D	
E	

Step 2: Construct the FP-tree from the above data

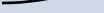
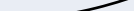

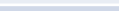

TID	Items	(Ordered) Frequent Items
1	A, B	A, B
2	B, C, D	B, C, D
3	A, C, D, E	A, C, D, E
4	A, D, E	A, D, E
5	A, B, C	A, B, C

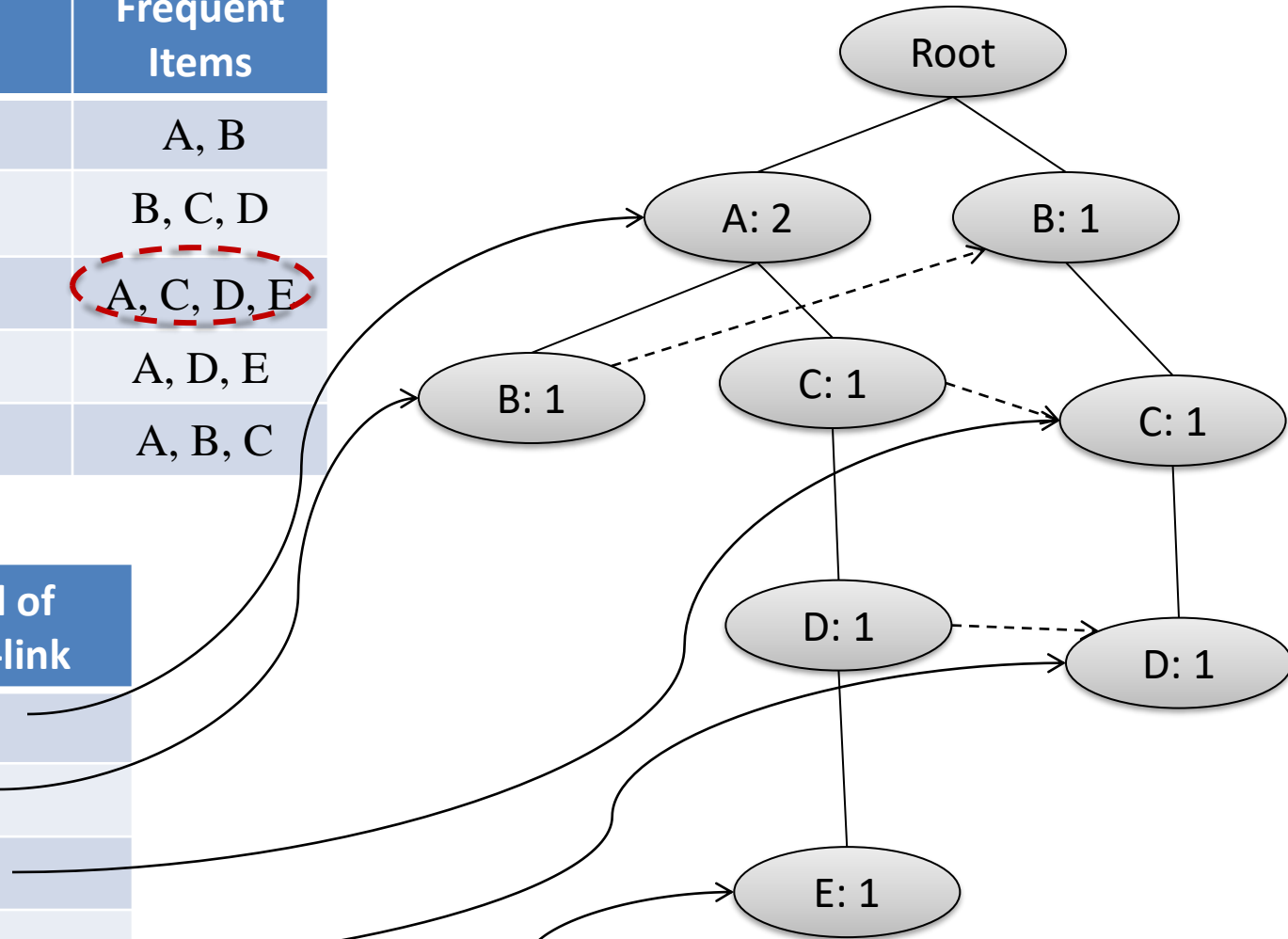
Item	Head of node-link
A	
B	
C	
D	
E	



Step 2: Construct the FP-tree from the above data

TID	Items	(Ordered) Frequent Items
1	A, B	A, B
2	B, C, D	B, C, D
3	A, C, D, E	A, C, D, E
4	A, D, E	A, D, E
5	A, B, C	A, B, C

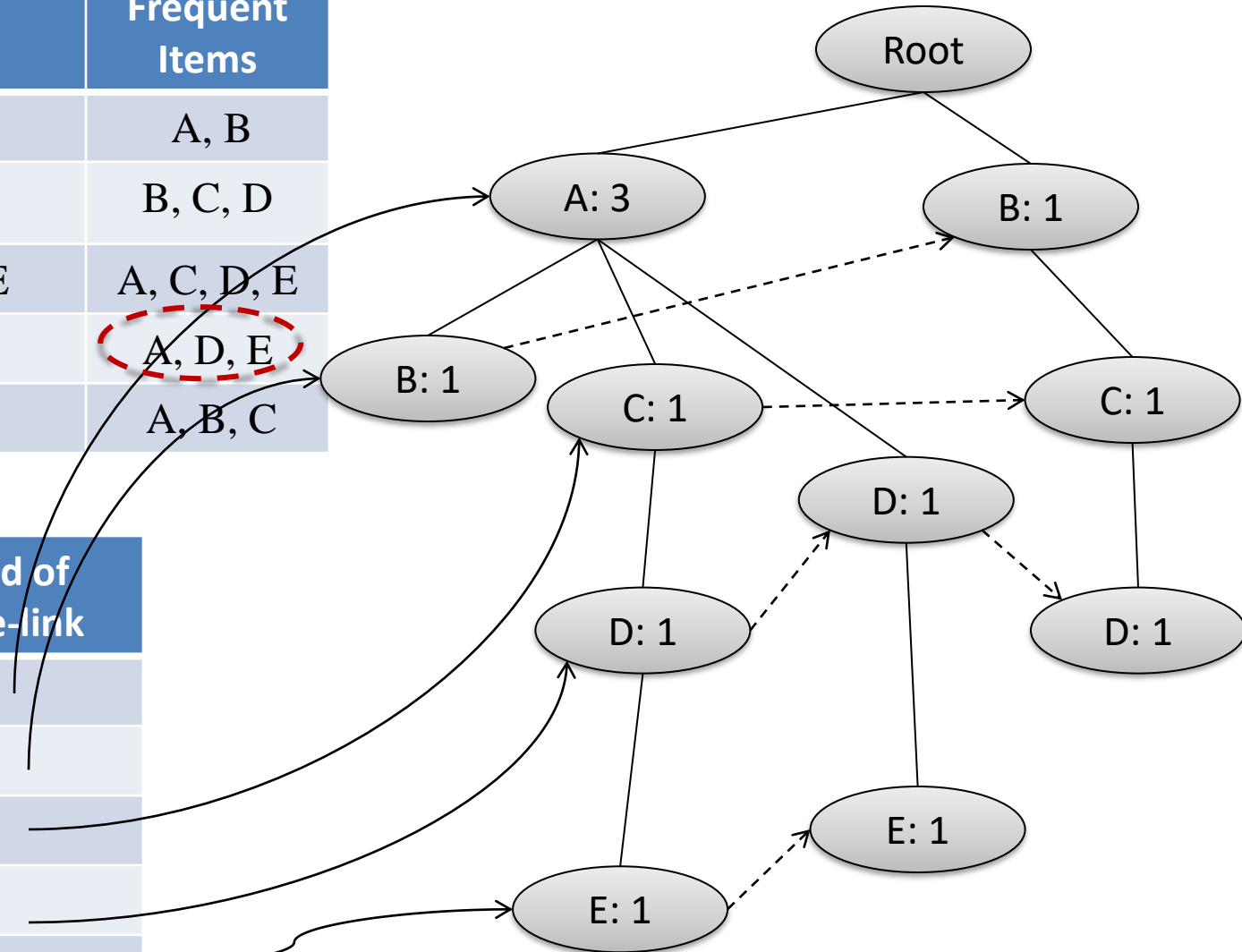
Item	Head of node-link
A	
B	
C	
D	
E	



Step 2: Construct the FP-tree from the above data

TID	Items	(Ordered) Frequent Items
1	A, B	A, B
2	B, C, D	B, C, D
3	A, C, D, E	A, C, D, E
4	A, D, E	A, D, E
5	A, B, C	A, B, C

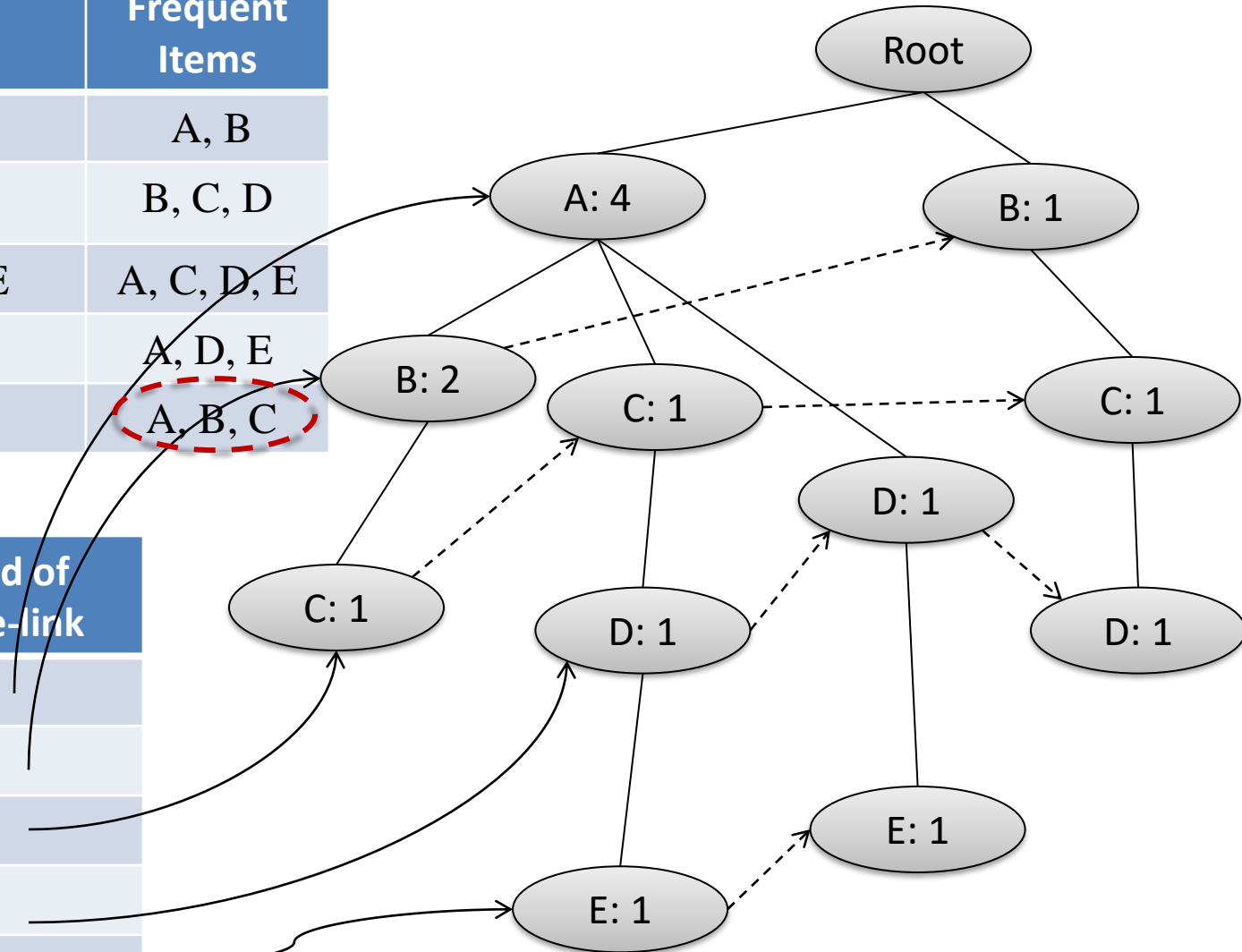
Item	Head of node-link
A	
B	
C	
D	
E	



Step 2: Construct the FP-tree from the above data

TID	Items	(Ordered) Frequent Items
1	A, B	A, B
2	B, C, D	B, C, D
3	A, C, D, E	A, C, D, E
4	A, D, E	A, D, E
5	A, B, C	A, B, C

Item	Head of node-link
A	
B	
C	
D	
E	

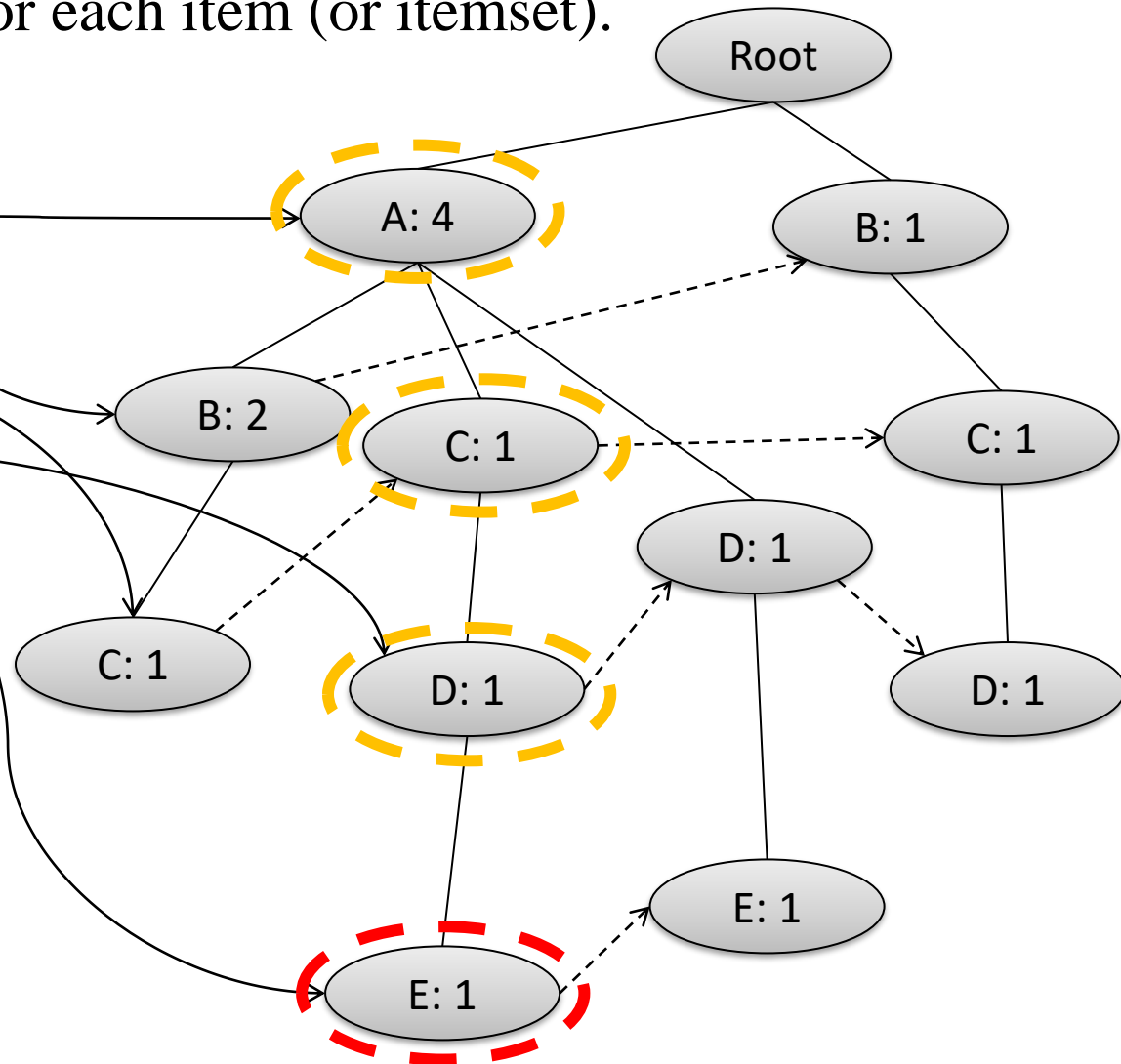


FP-growth Algorithm

- **Step 1:** Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
- **Step 2:** Construct the FP-tree from the above data
- **Step 3:** From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
- **Step 4:** Determine the frequent patterns.

Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).

Item	Head of node-link
A	
B	
C	
D	
E	



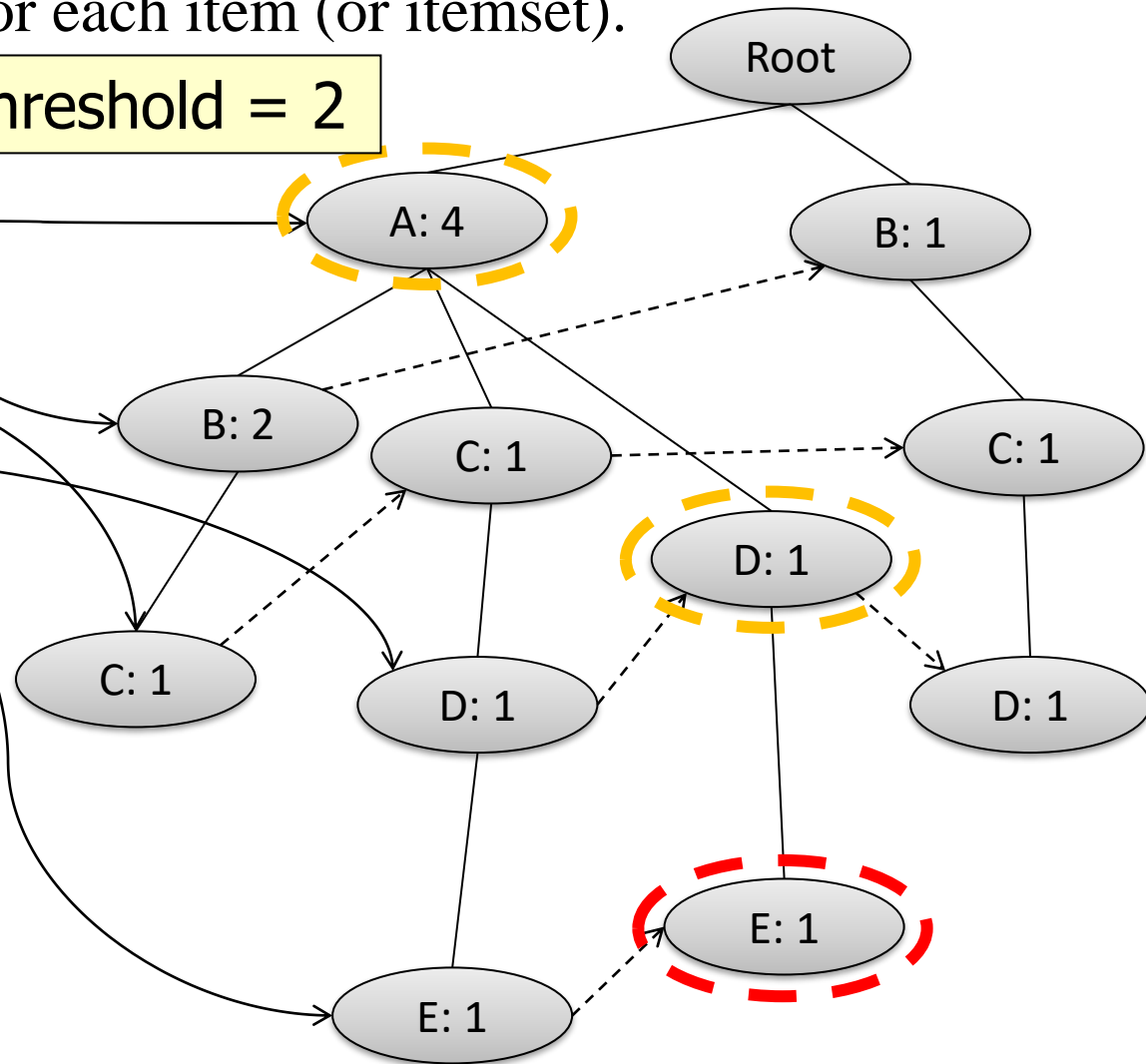
Cond. FP-tree on "E"

{ (A:1, C:1, D:1, E:1),
}

Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).

Item	Head of node-link
A	
B	
C	
D	
E	

Threshold = 2



Cond. FP-tree on "E"

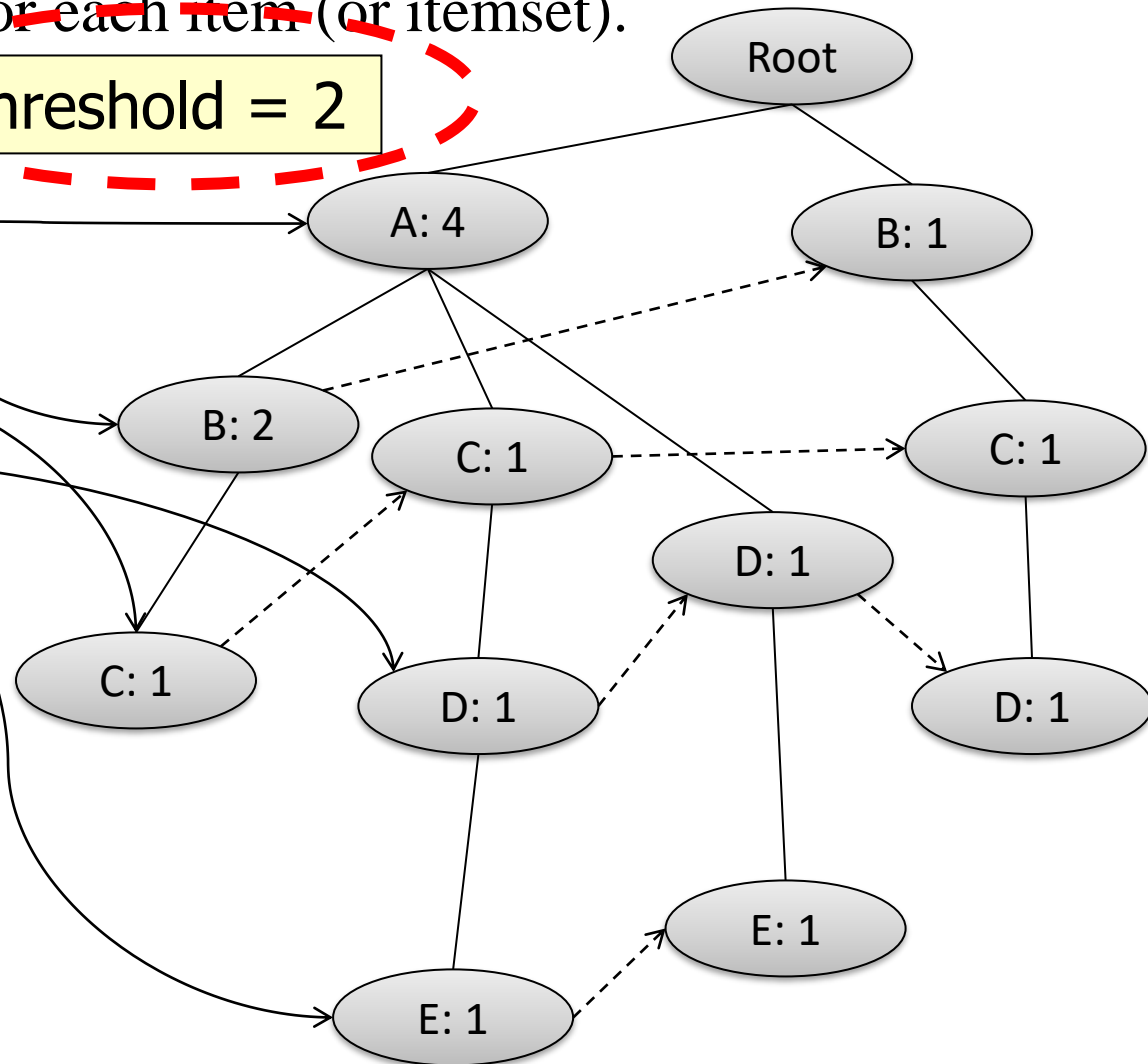
{ (A:1, C:1, D:1, E:1),
(A:1, D:1, E:1), }

Item	Frequency
A	2
C	1
D	2
E	2

Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).

Item	Head of node-link
A	
B	
C	
D	
E	

Threshold = 2



Cond. FP-tree on "E"

2

{ (A:1, C:1, D:1, E:1),
(A:1, D:1, E:1), }

Item	Frequency
A	2
C	1
D	2
E	2

Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).

Threshold = 2

Cond. FP-tree on "E"

{ (A:1, C:1, D:1, E:1),
(A:1, D:1, E:1), }

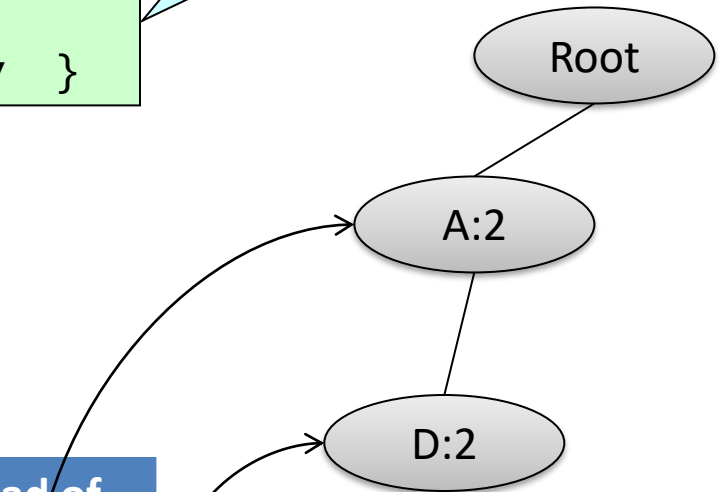
{ (A:1, D:1, E:1),
(A:1, D:1, E:1), }

conditional pattern base of
"E"

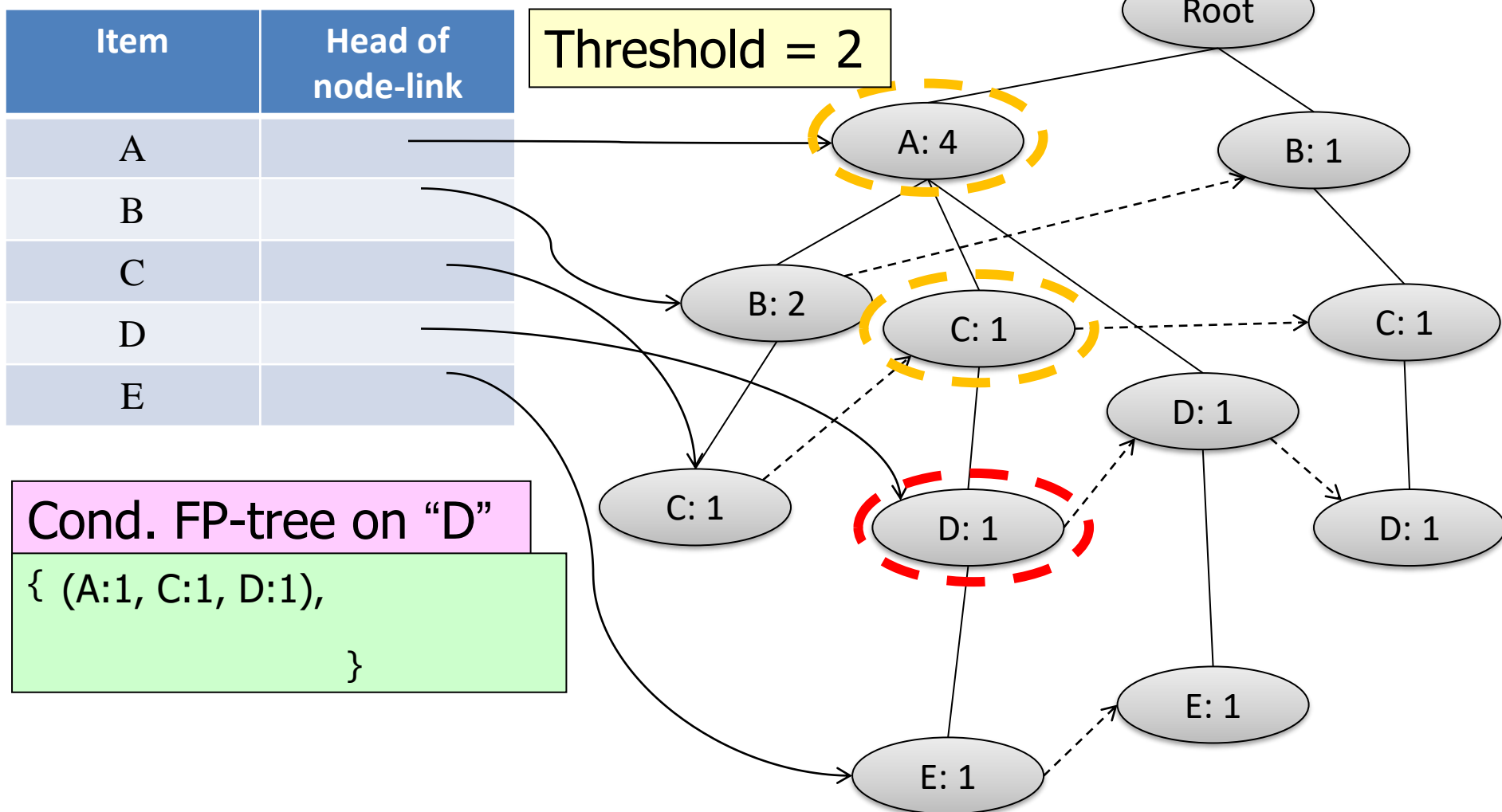
Item	Frequency
A	2
C	1
D	2
E	2

Item	Frequency
A	2
D	2
E	2

Item	Head of node-link
A	
D	



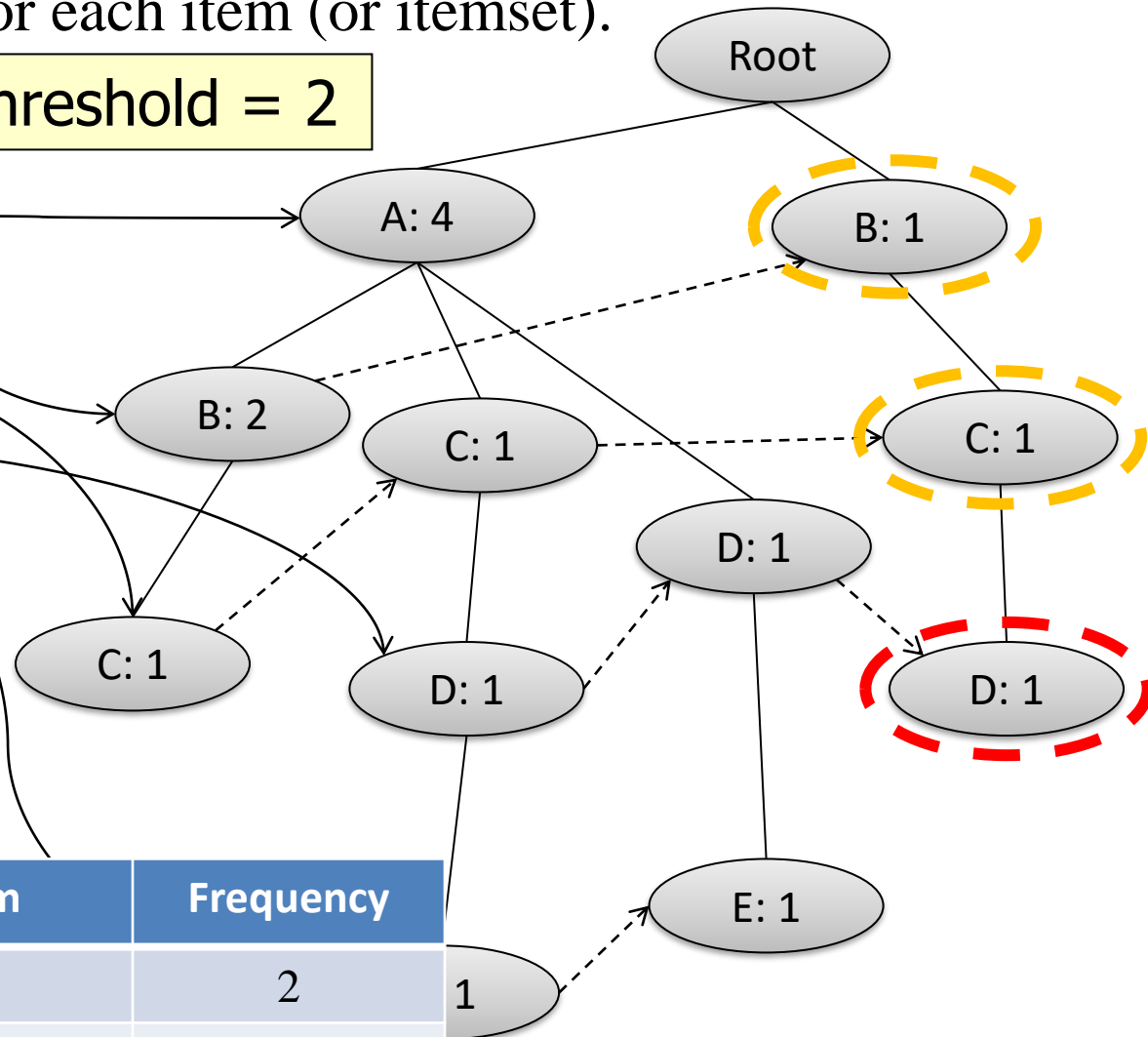
Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).



Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).

Item	Head of node-link
A	
B	
C	
D	
E	

Threshold = 2



Cond. FP-tree on "D"

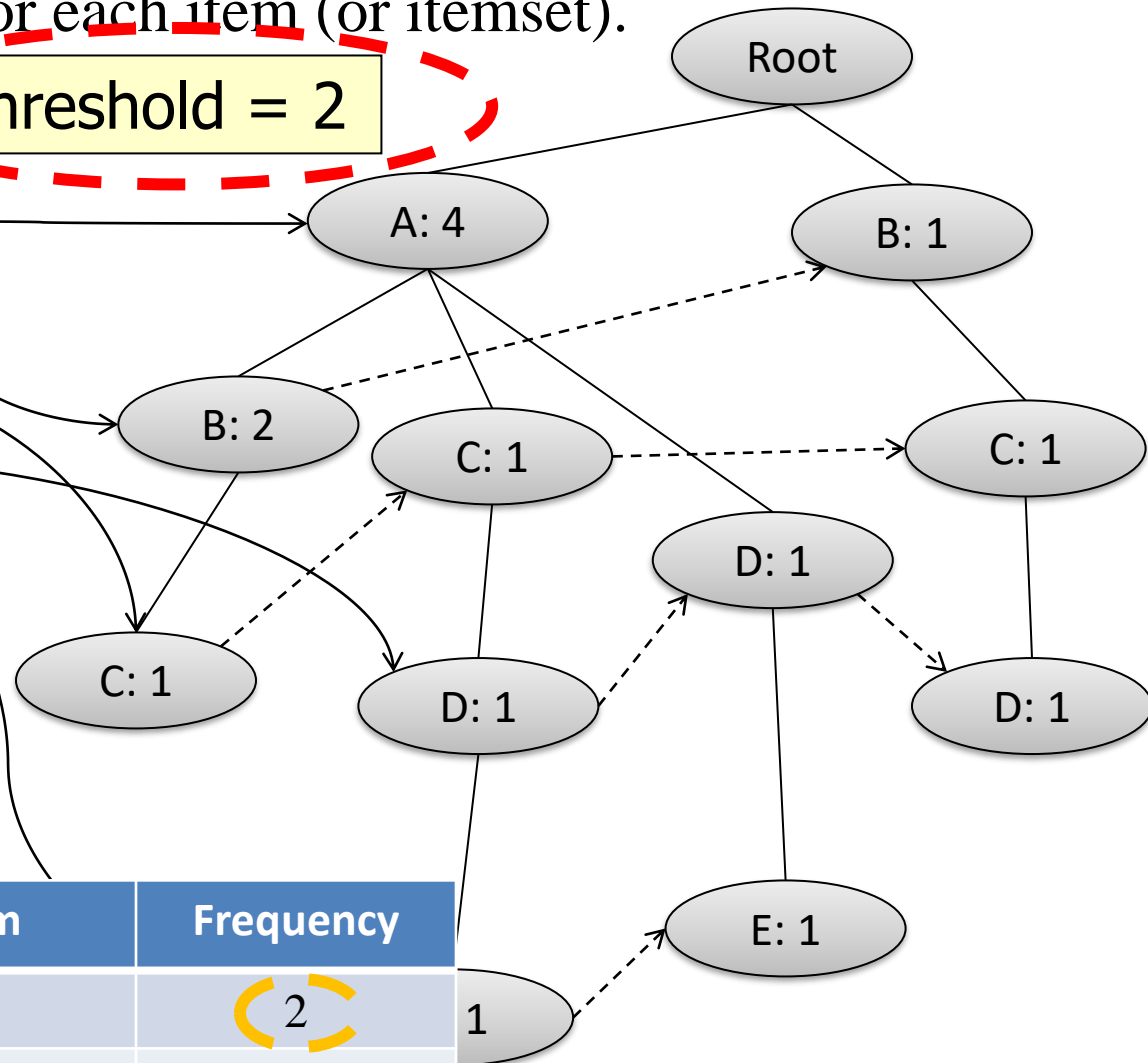
{ (A:1, C:1, D:1),
(A:1, D:1),
(B:1, C:1, D:1), }

Item	Frequency
A	2
B	1
C	2
D	3

Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).

Item	Head of node-link
A	
B	
C	
D	
E	

Threshold = 2



Cond. FP-tree on "D" 3

{ (A:1, C:1, D:1),
(A:1, D:1),
(B:1, C:1, D:1), }

Item	Frequency
A	2
B	1
C	2
D	3

Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).

Threshold = 2

Cond. FP-tree on "D"

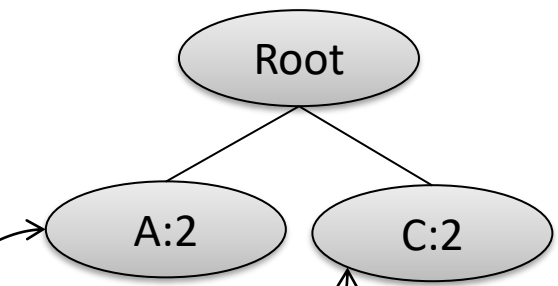
{ (A:1, C:1, D:1),
(A:1, D:1),
(B:1, C:1, D:1), }

{(A:1, D:1),
(C:1, D:1), }

Item	Frequency
A	2
B	1
C	2
D	3

Item	Frequency
A	2
C	2
D	2

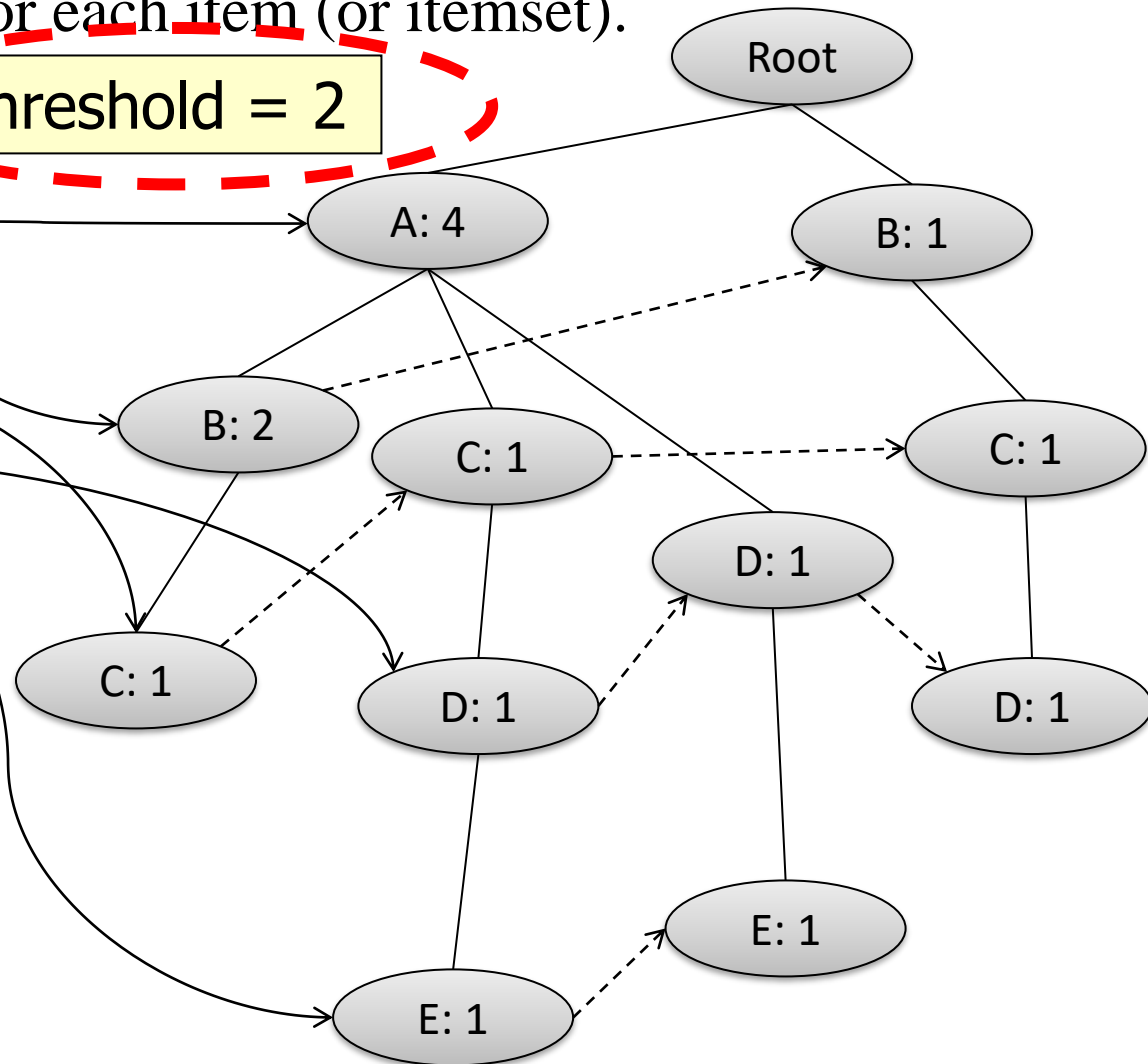
Item	Head of node-link
A	
C	



Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).

Item	Head of node-link
A	
B	
C	
D	
E	

Threshold = 2



Cond. FP-tree on "C" 3

{ (A:1, B:1, C:1),
 (A:1, C:1),
 (B:1, C:1), }

Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).

Threshold = 2

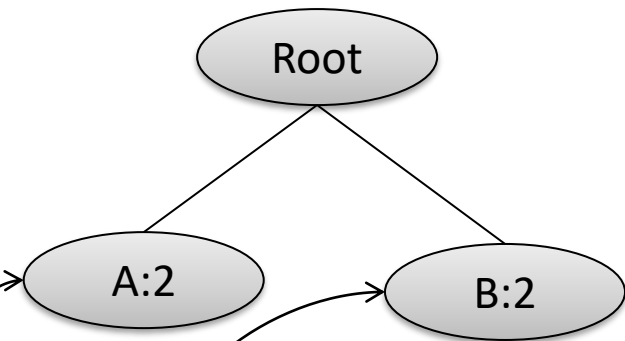
Cond. FP-tree on "C"

{ (A:1, B:1, C:1),
(A:1, C:1),
(B:1, C:1), }

{(A:1, C:1),
(B:1, C:1), }

Item	Frequency
A	2
B	2
C	3

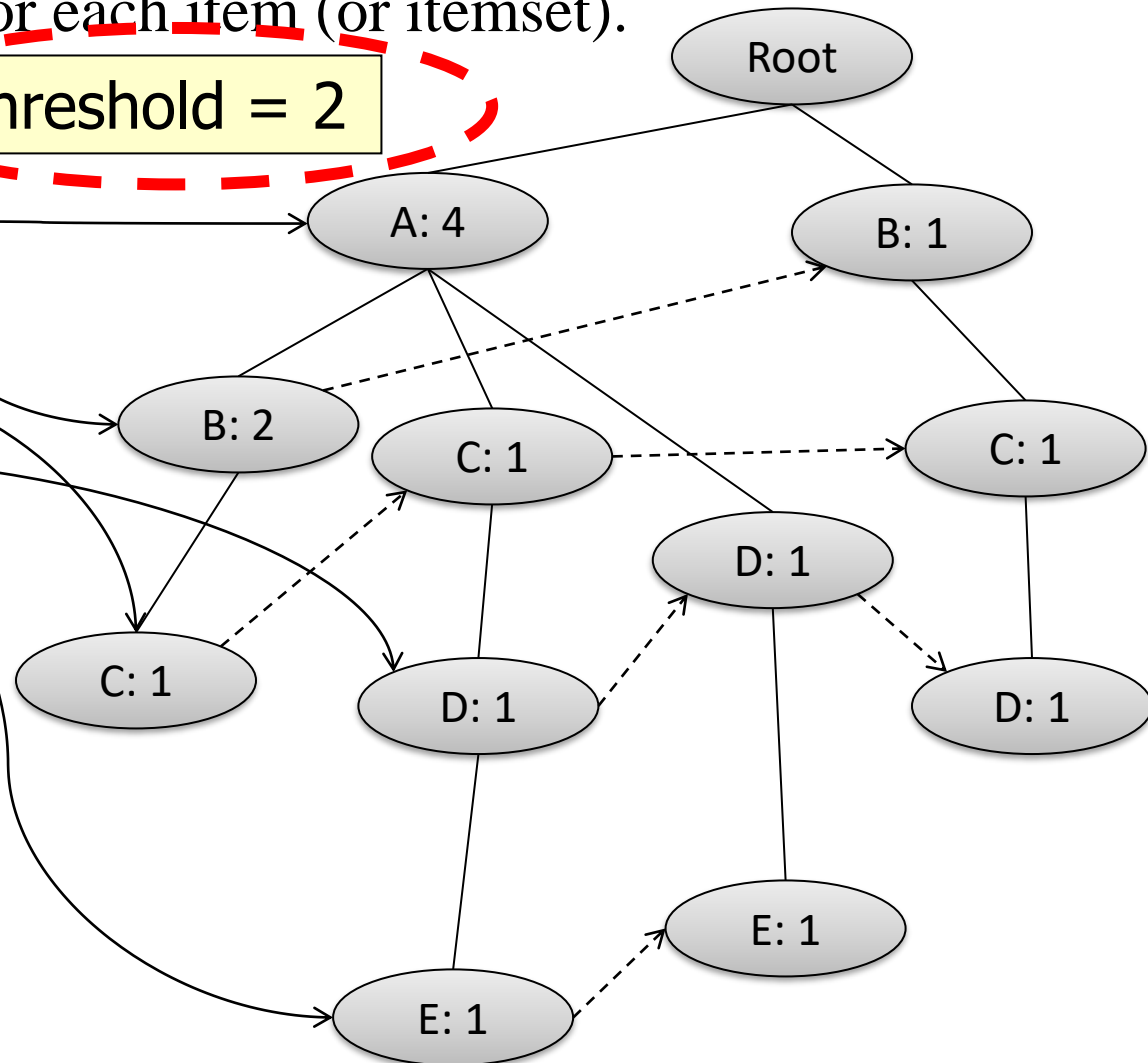
Item	Head of node-link
A	
B	



Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).

Item	Head of node-link
A	
B	
C	
D	
E	

Threshold = 2



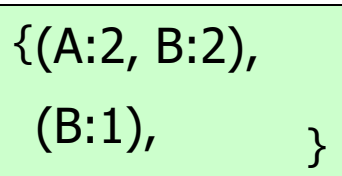
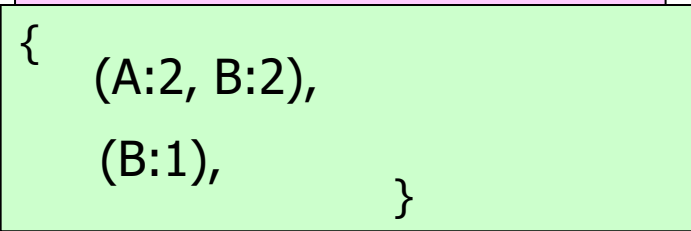
Cond. FP-tree on "B" 3

{ (A:2, B:2),
(B:1), }

Step 3: From the FP-tree above, construct the FP-conditional tree for each ~~item~~ (or itemset).

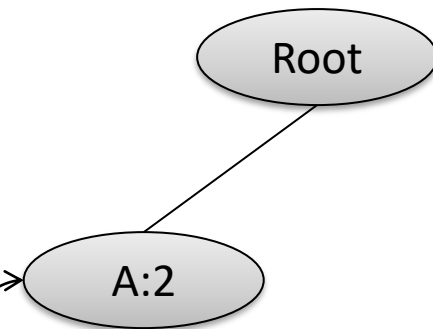
Threshold = 2

Cond. FP-tree on "B"

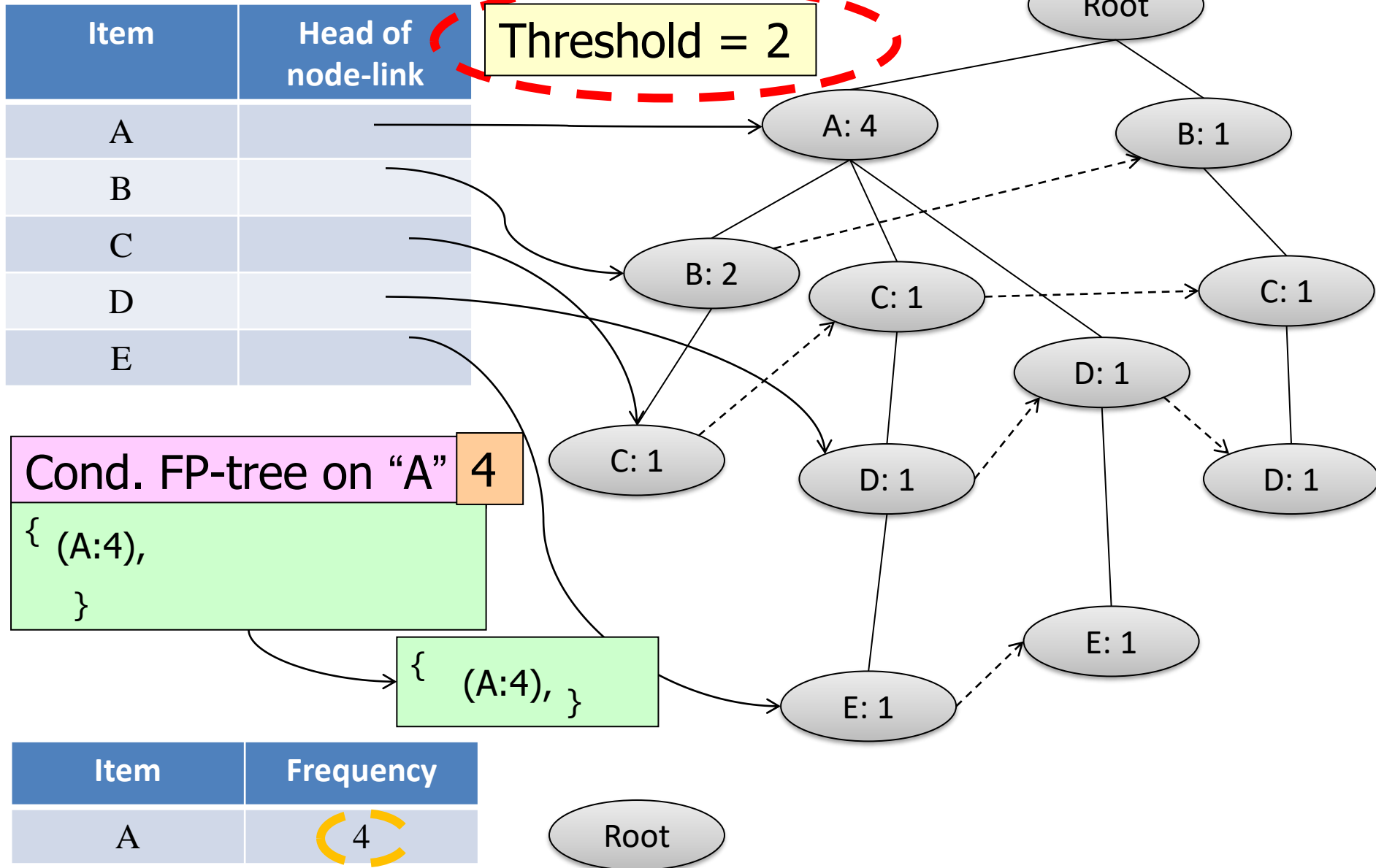


Item	Frequency
A	2
B	3

Item	Head of node-link
A	



Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).



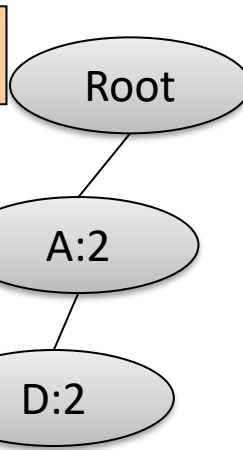
FP-growth Algorithm

- **Step 1:** Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
- **Step 2:** Construct the FP-tree from the above data
- **Step 3:** From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
- **Step 4:** Determine the frequent patterns.

Cond. FP-tree on "E"

2

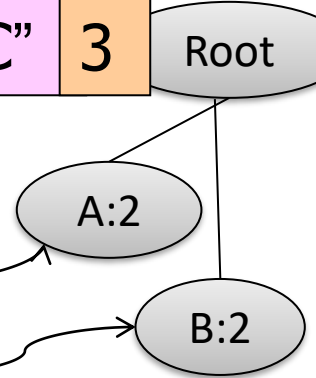
Item	Head of node-link
A	
D	



Cond. FP-tree on "C"

3

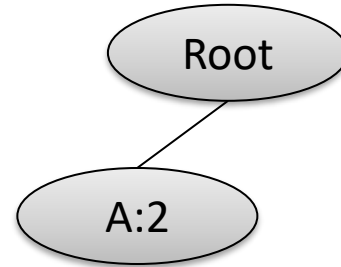
Item	Head of node-link
A	
B	



Cond. FP-tree on "B"

3

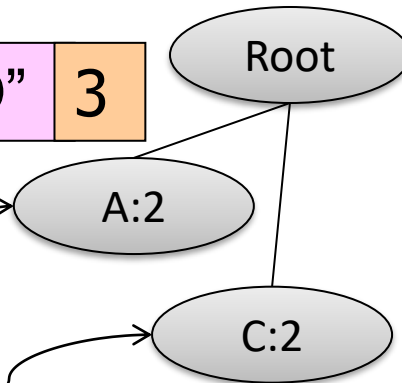
Item	Head of node-link
A	



Cond. FP-tree on "D"

3

Item	Head of node-link
A	
C	



Cond. FP-tree on "A"

4



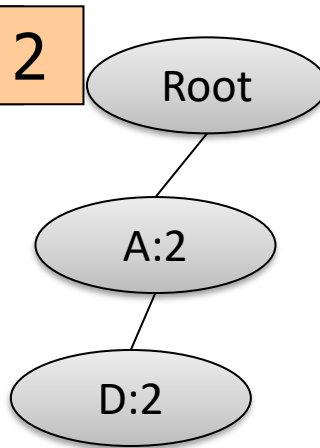
Cond. FP-tree on "E" 2

1. Before generating this cond. tree, we generate

$\{E\}$ (support = 2)

2. After generating this cond. tree, we generate

$\{A, D, E\}$, $\{A, E\}$, $\{D, E\}$ (support = 2)

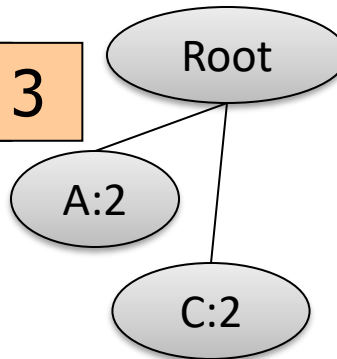


Cond. FP-tree on "D" 3

1. Before generating this cond. tree, we generate $\{D\}$ (support = 3)

2. After generating this cond. tree, we generate

$\{A, D\}$, $\{C, D\}$ (support = 2)



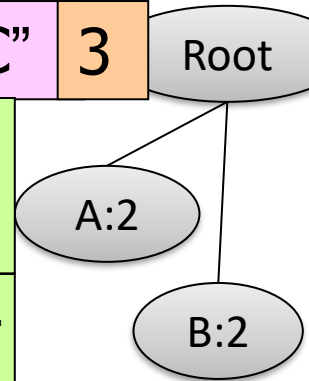
Cond. FP-tree on "C" 3

1. Before generating this cond. tree, we generate

$\{C\}$ (support = 3)

2. After generating this cond. tree, we generate

$\{A, C\}$, $\{B, C\}$ (support = 2)



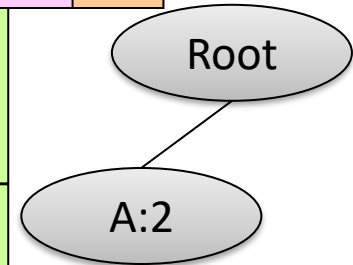
Cond. FP-tree on "B" 3

1. Before generating this cond. tree, we generate

$\{B\}$ (support = 3)

2. After generating this cond. tree, we generate

$\{A, B\}$ (support = 2)



Cond. FP-tree on "A" 4

1. Before generating this cond. tree, we generate

$\{A\}$ (support = 4)

2. After generating this cond. tree, we do not generate any itemset.



Step 4: Determine the frequent patterns

Answer:

According to the above step 4, we can find all frequent itemsets with support ≥ 2):

TID	Items
1	A, B
2	B, C, D
3	A, C, D, E
4	A, D, E
5	A, B, C

$\{E\}, \{A,D,E\}, \{A,E\}, \{D,E\},$
 $\{D\}, \{A,D\}, \{C,D\},$
 $\{C\}, \{A,C\}, \{B,C\},$
 $\{B\}, \{A,B\}$
 $\{A\}$