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:

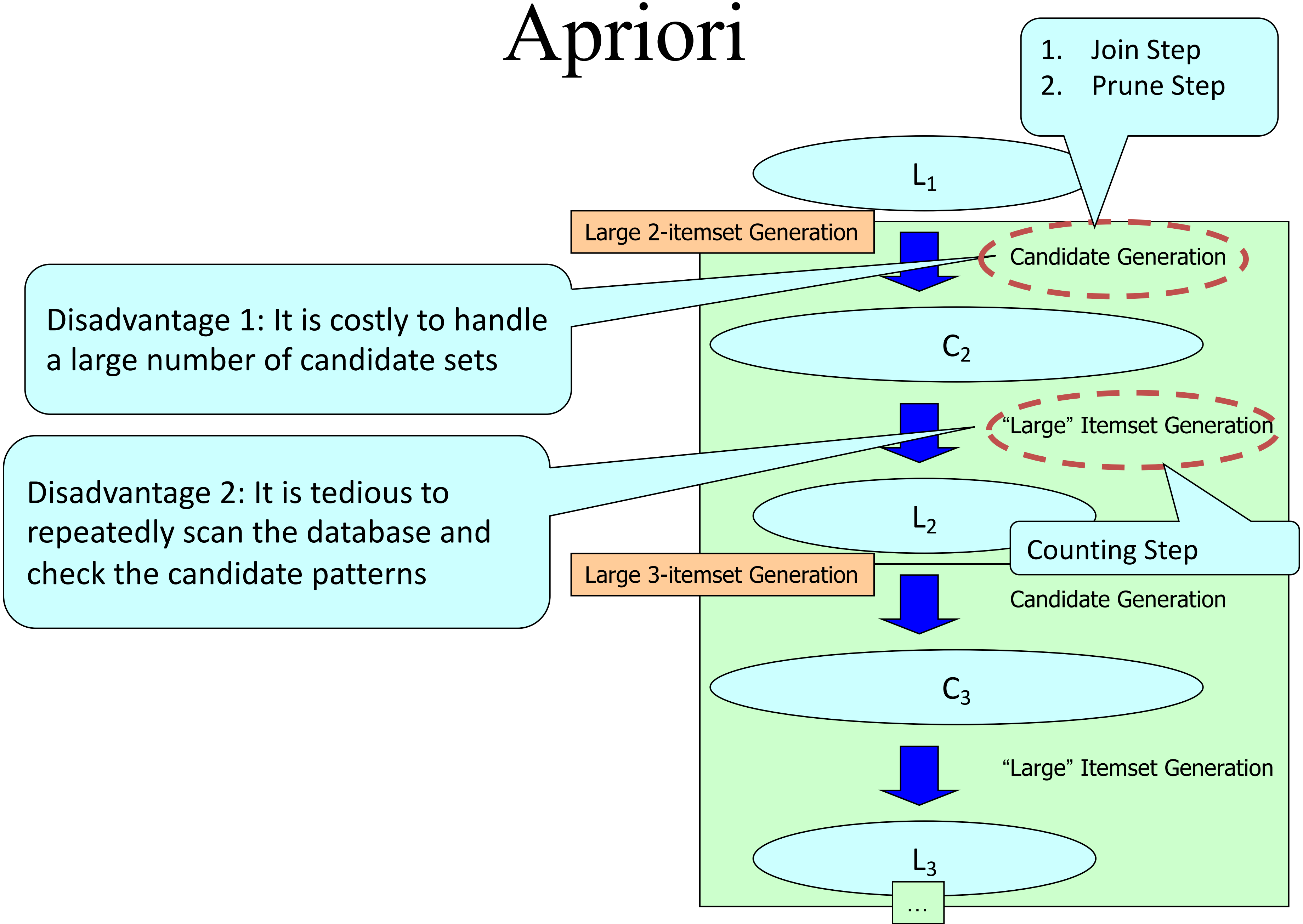
$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - 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



1. Join Step
2. Prune Step

L1

Large 2-itemset Generation

Candidate Generation

C2

"Large" Itemset Generation

L2

Counting Step

Large 3-itemset Generation

Candidate Generation

C3

"Large" Itemset Generation

L3

…

Disadvantage 1: It is costly to handle a large number of candidate sets

Disadvantage 2: It is tedious to repeatedly scan the database and check the candidate patterns

Max-patterns

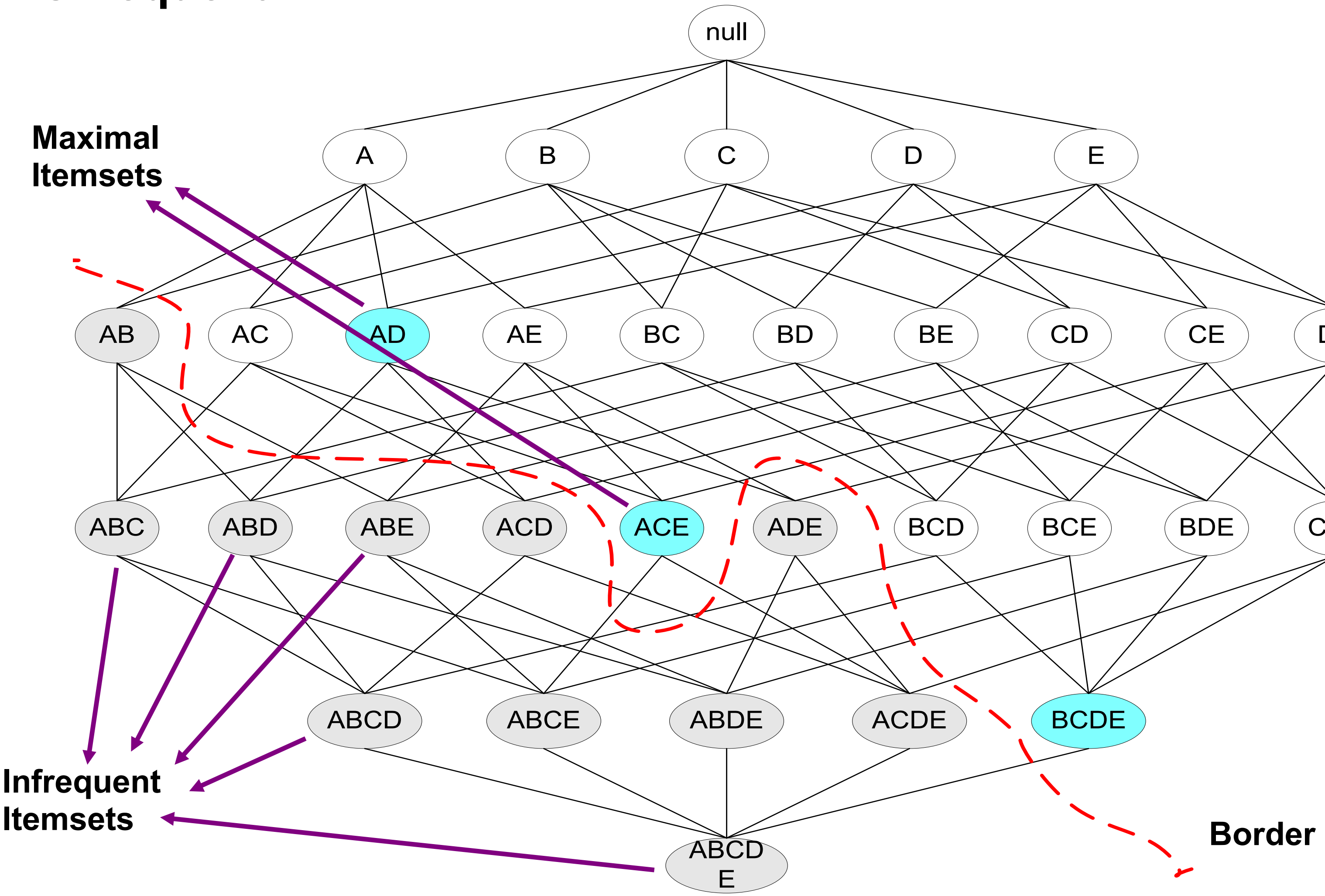
- Max-pattern: frequent patterns without proper frequent super pattern
  - BCDE, ACD are max-patterns
  - BCD is not a max-pattern

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.

1st scan: count support

Generate **frequent** itemsets of **length 1**

C_1

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

Eliminate candidates that are infrequent →

L_1

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

A Simple Example

L_1

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

Generate **length 2 candidate** itemsets from length 1 frequent itemsets

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan: **Count support**

C_2

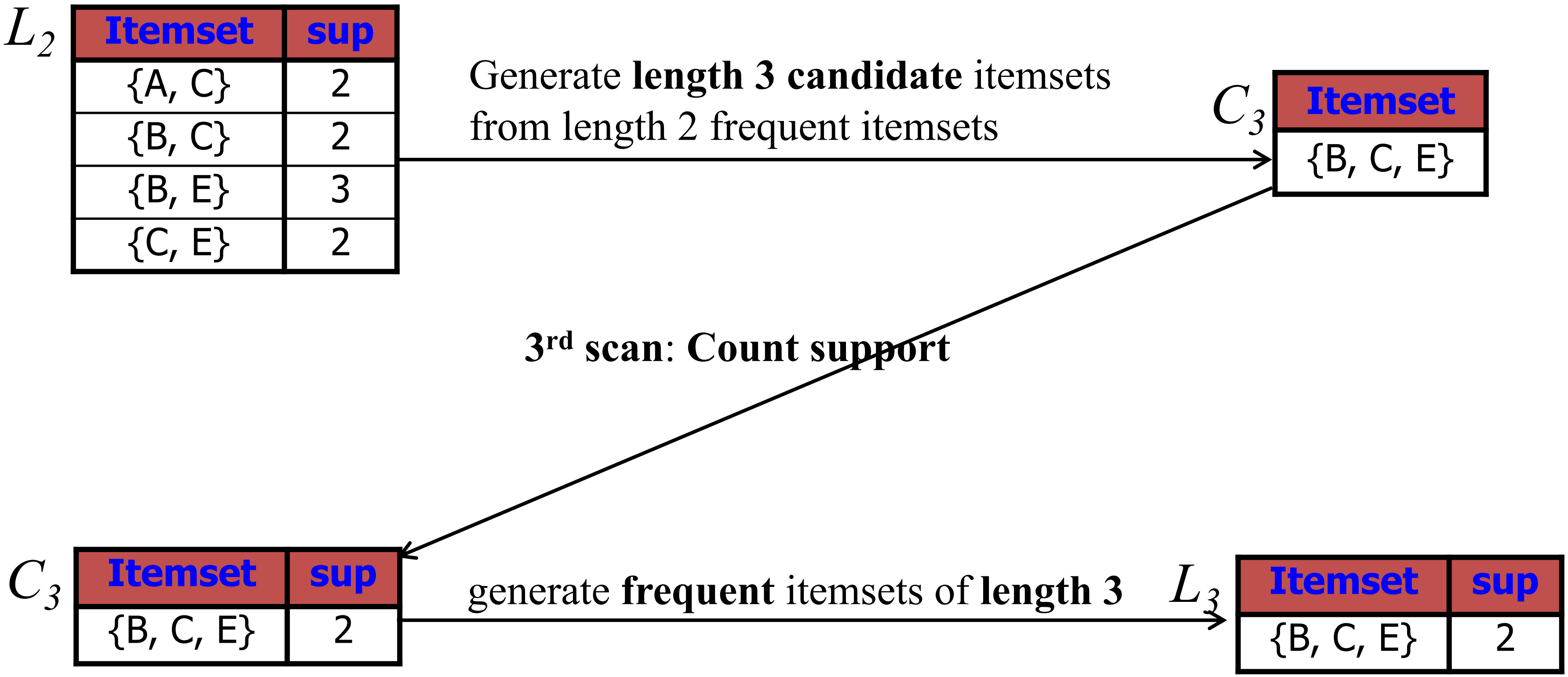
Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

generate **frequent** itemsets of **length 2**

L_2

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

A Simple Example



L_2

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

Generate **length 3 candidate** itemsets from length 2 frequent itemsets

C_3

Itemset
{B, C, E}

3rd scan: **Count support**

C_3

Itemset	sup
{B, C, E}	2

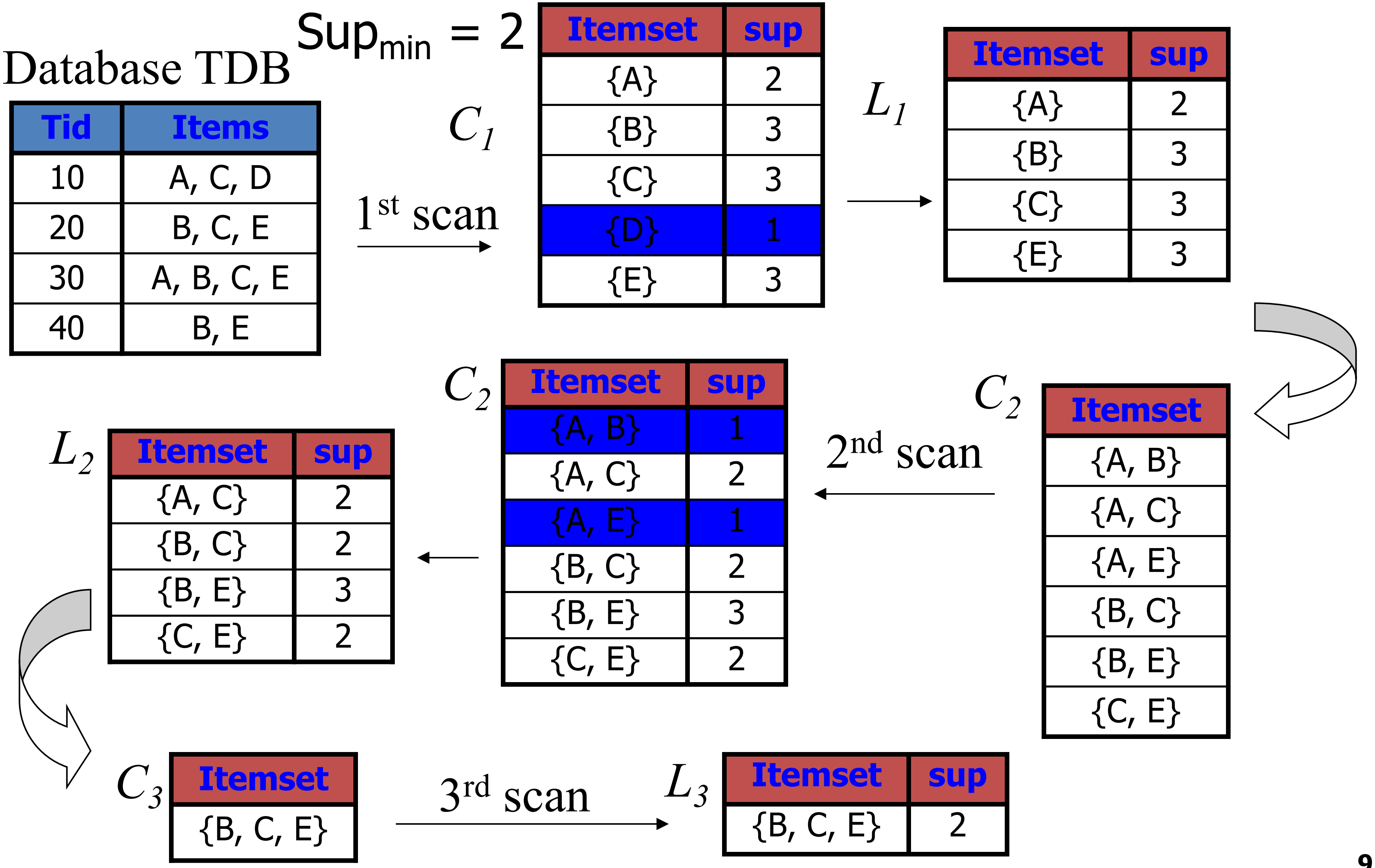
generate **frequent** itemsets of **length 3**

L_3

Itemset	sup
{B, C, E}	2

The algorithm stops here

A Simple Example



$Sup_{min} = 2$

Database TDB

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

1st scan

C_1

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

L_1

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

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

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

L_2

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

C_3

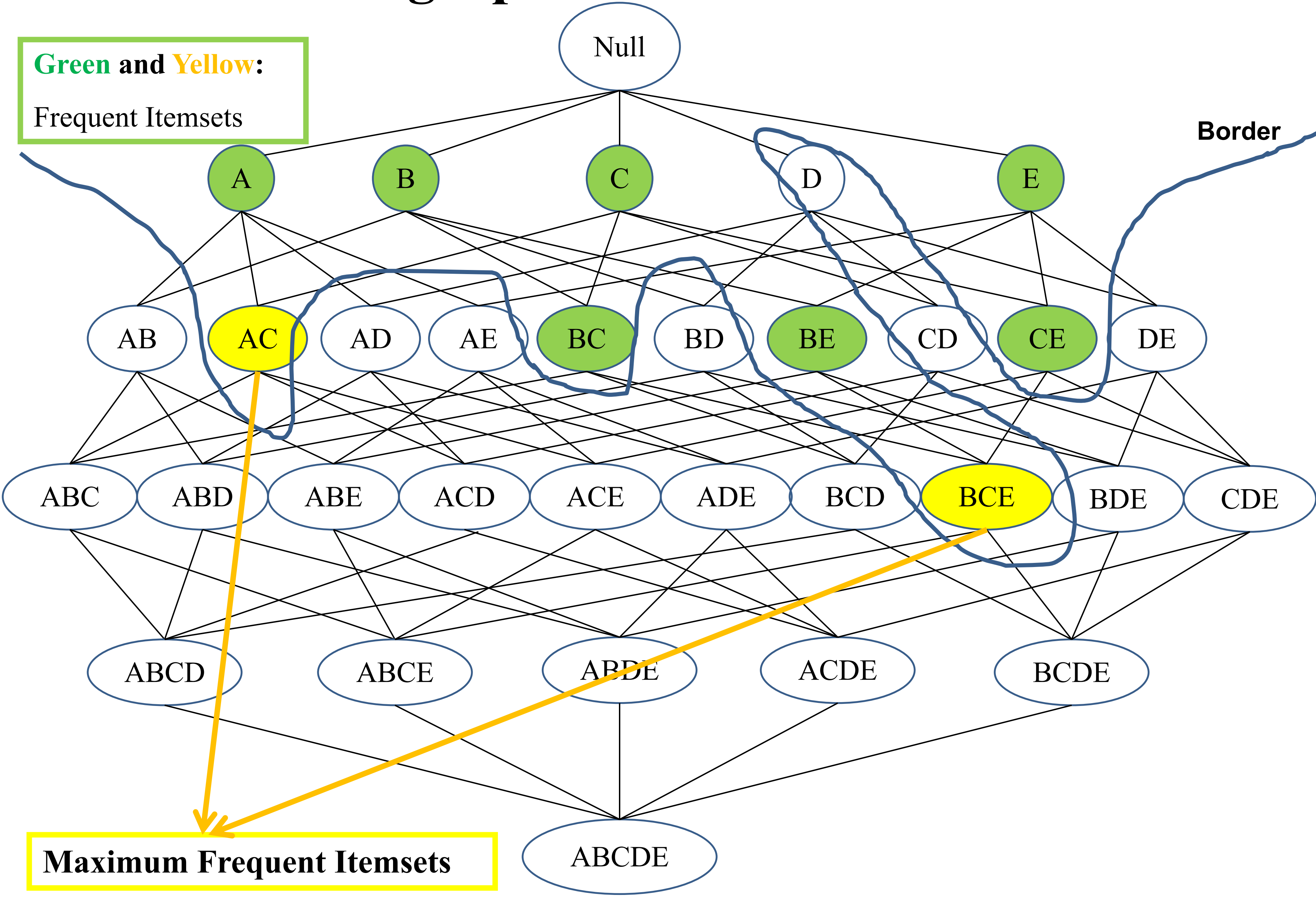
Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

Draw a graph to illustrate the result



Green and Yellow:

Frequent Itemsets

Maximum Frequent Itemsets

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.

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

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

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

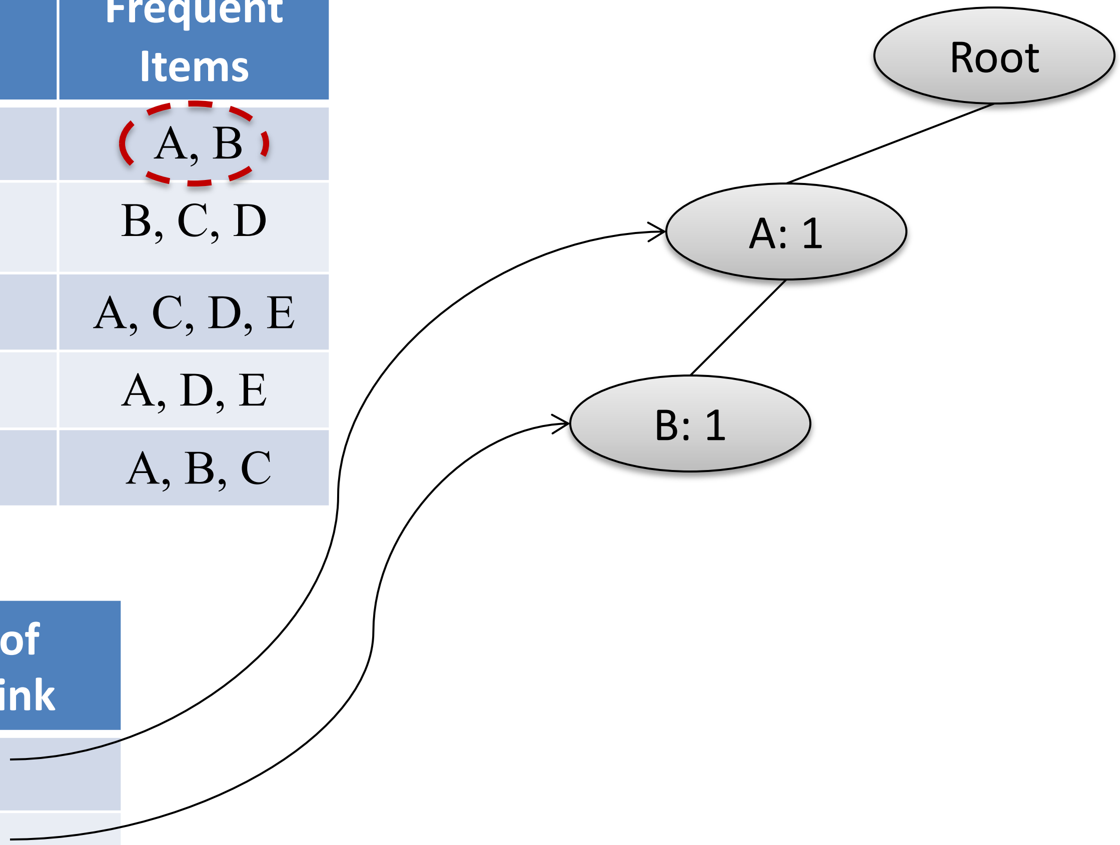
FP-growth Algorithm

- **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

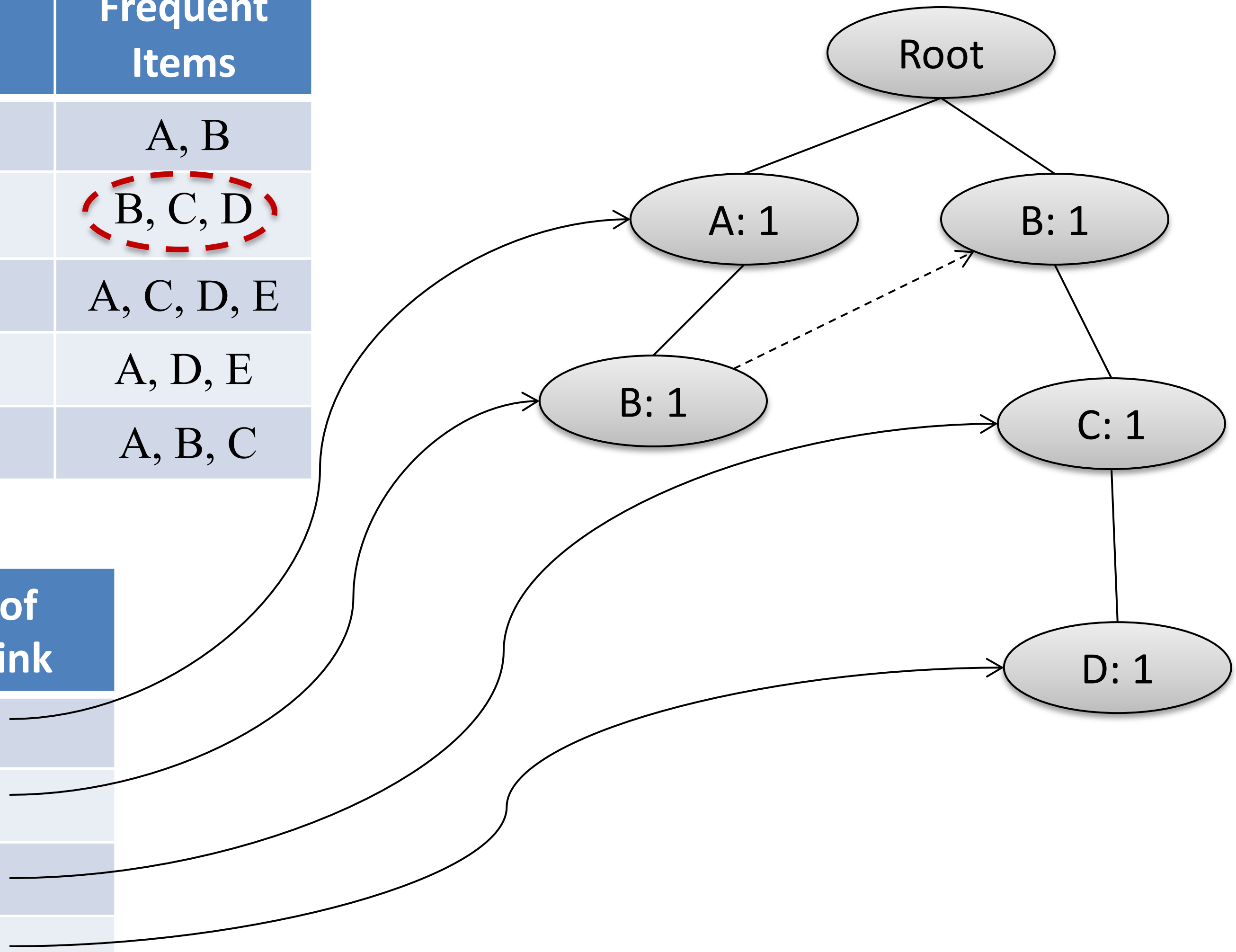
Item	Head of node-link
A	
B	
C	
D	
E	



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

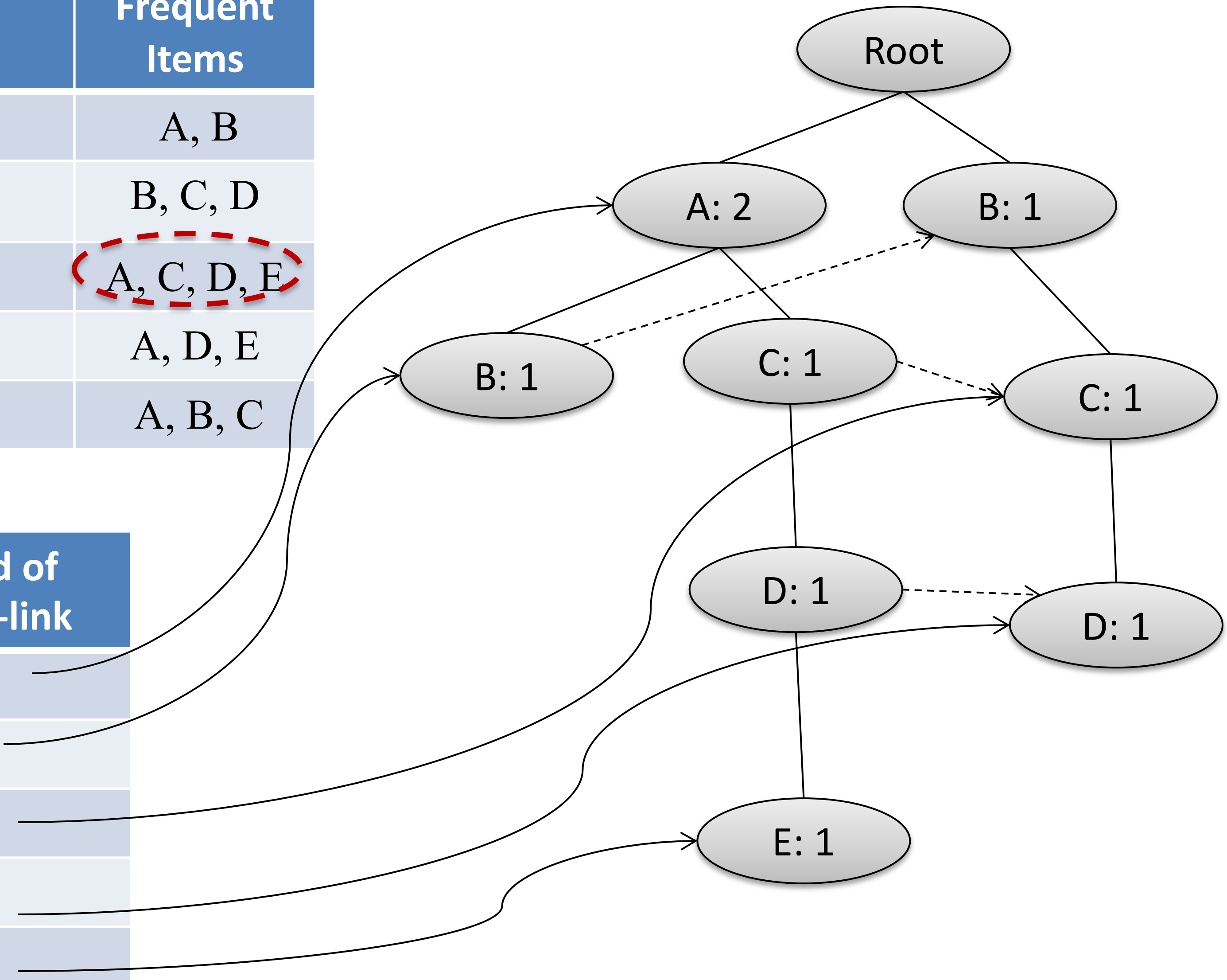
Item	Head of node-link
A	
B	
C	
D	
E	



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

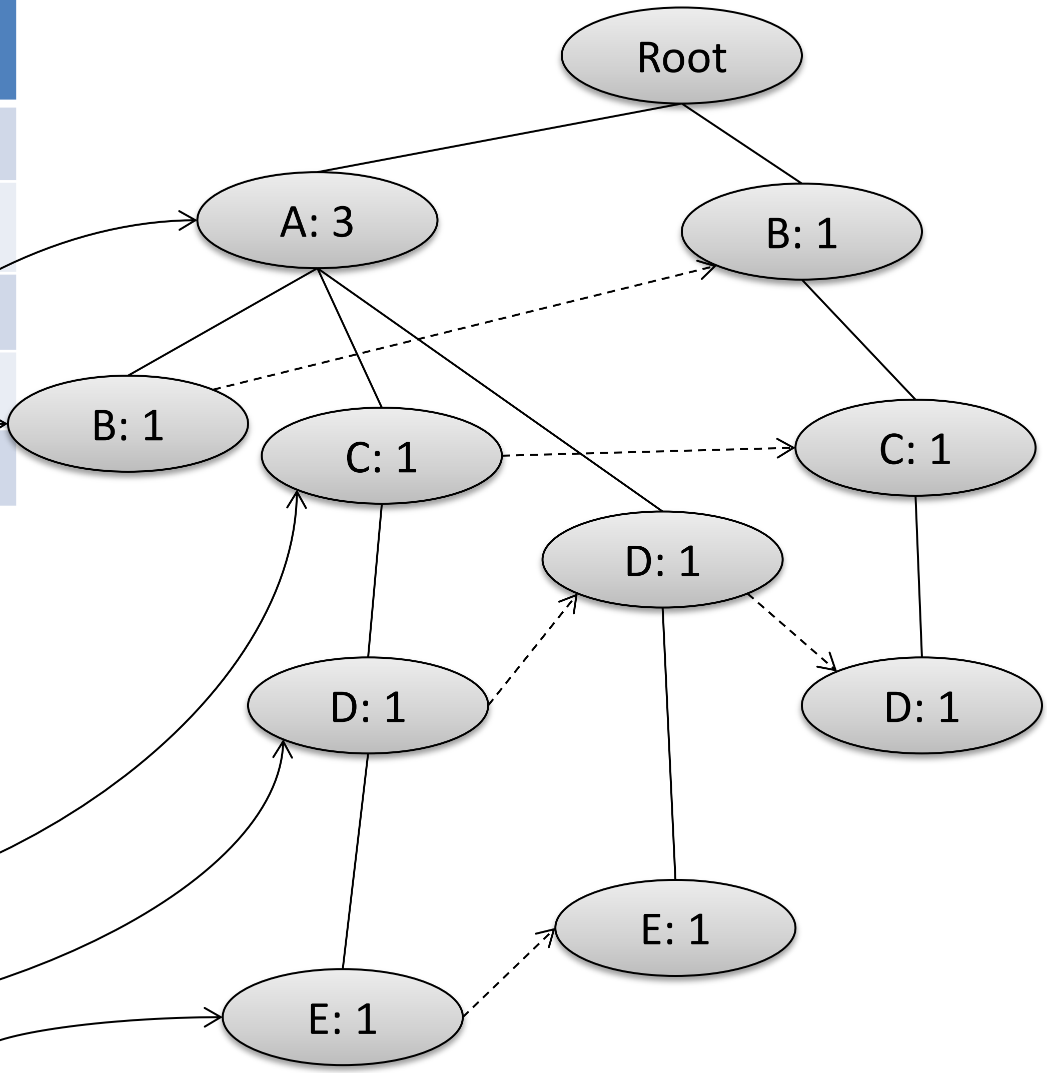
Item	Head of node-link
A	
B	
C	
D	
E	



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

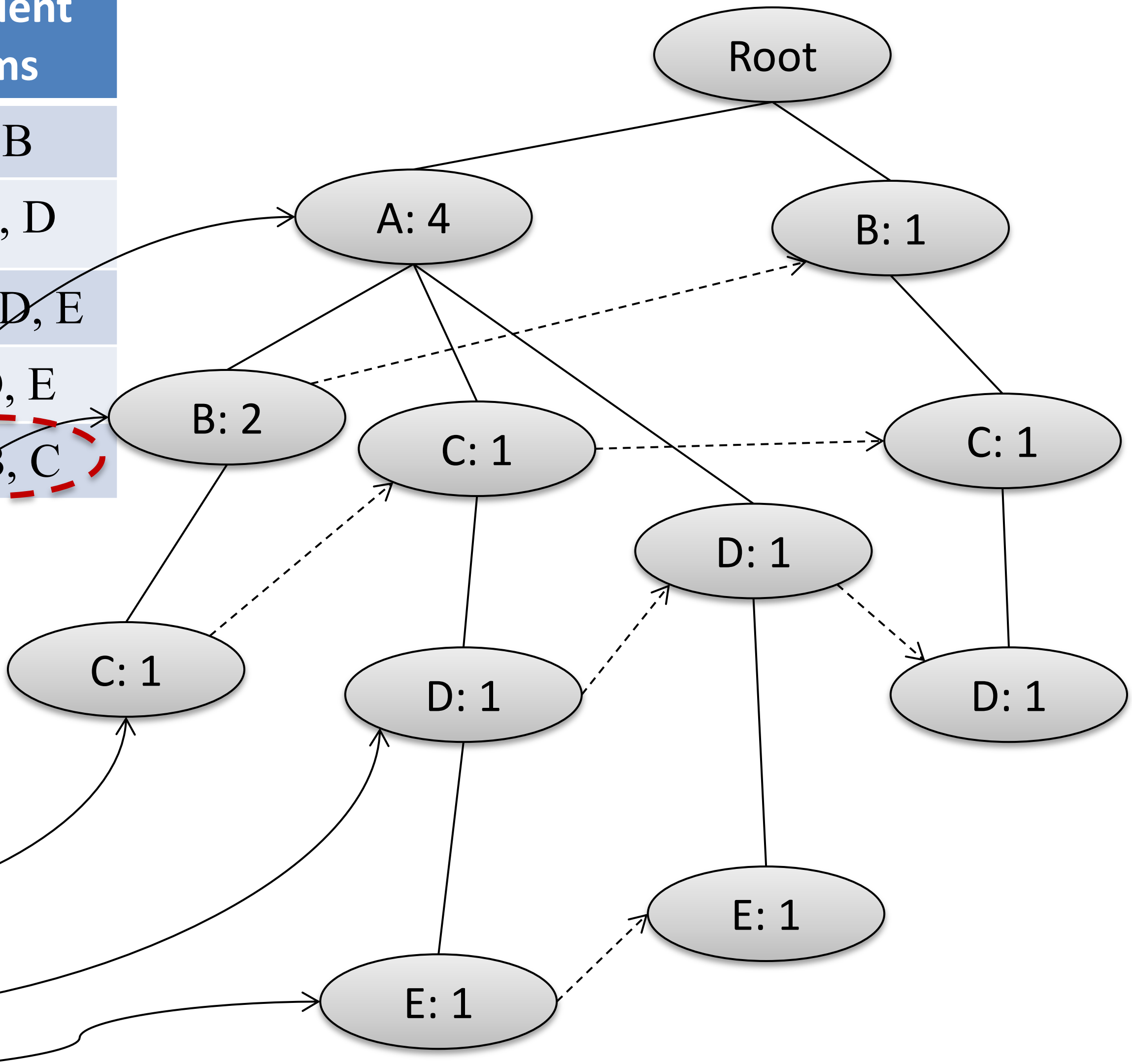
Item	Head of node-link
A	
B	
C	
D	
E	



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

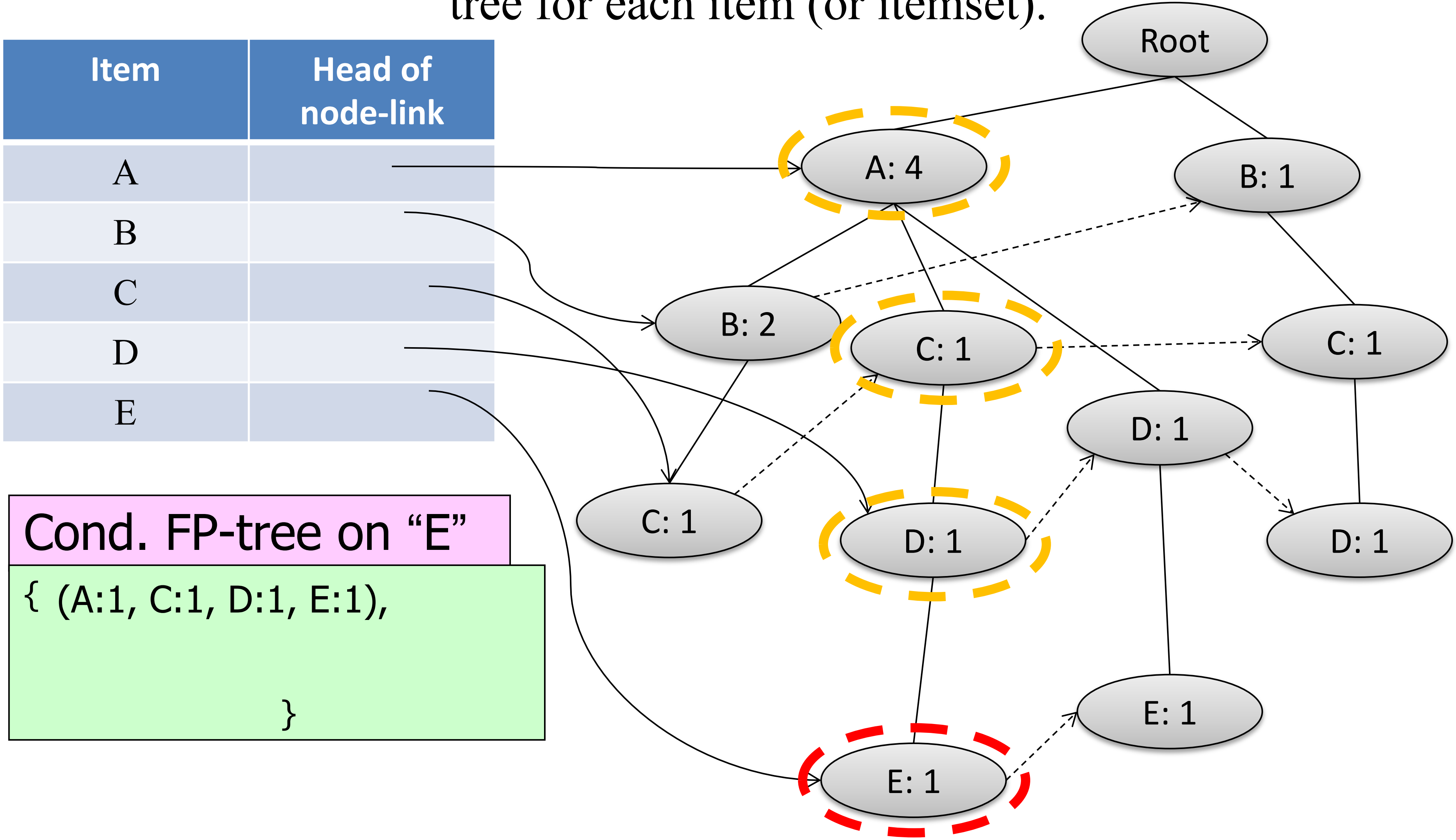
Item	Head of node-link
A	
B	
C	
D	
E	



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"

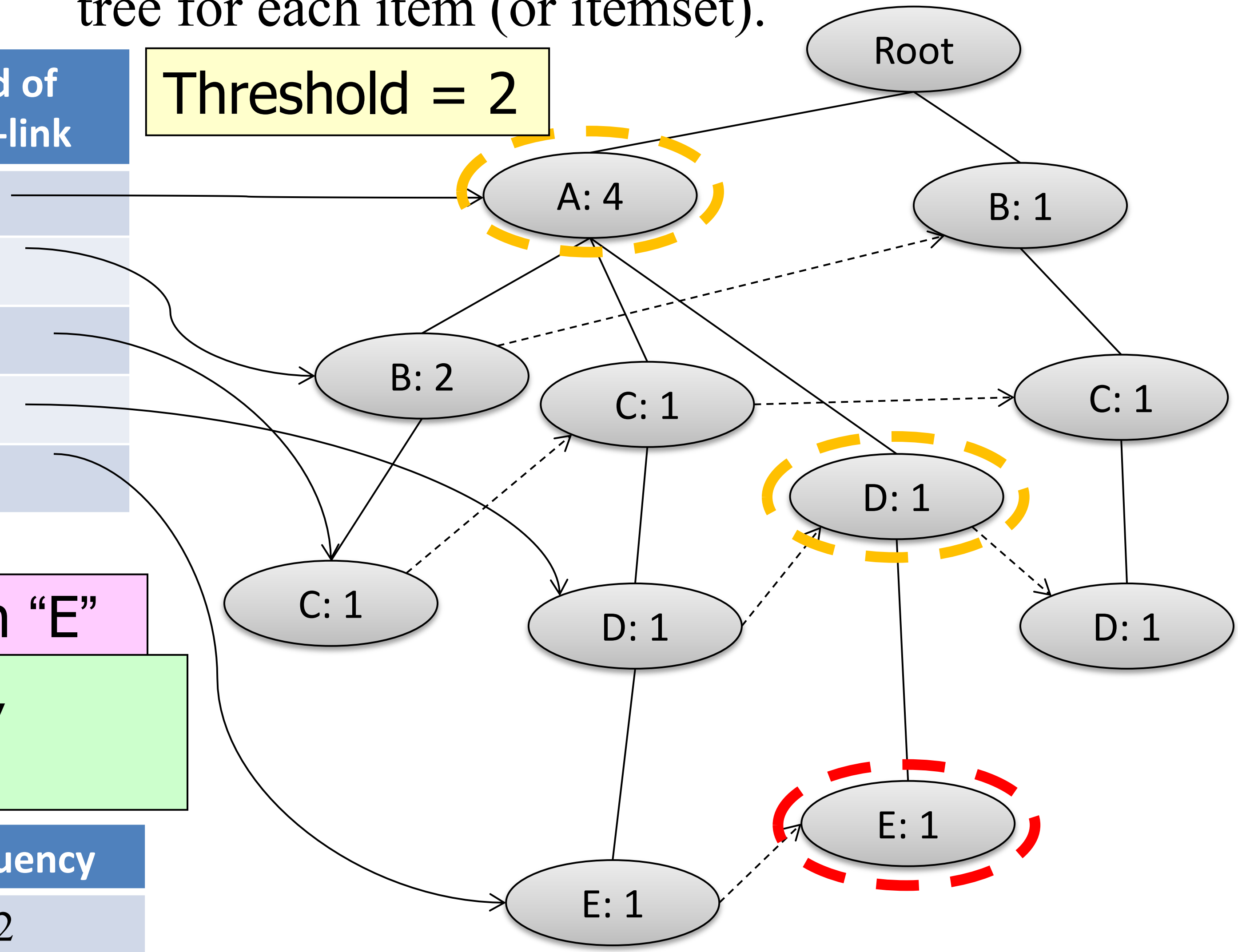
{ (A:1, C:1, D:1, E:1),

}

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

Threshold = 2

Item	Head of node-link
A	
B	
C	
D	
E	



Cond. FP-tree on "E"

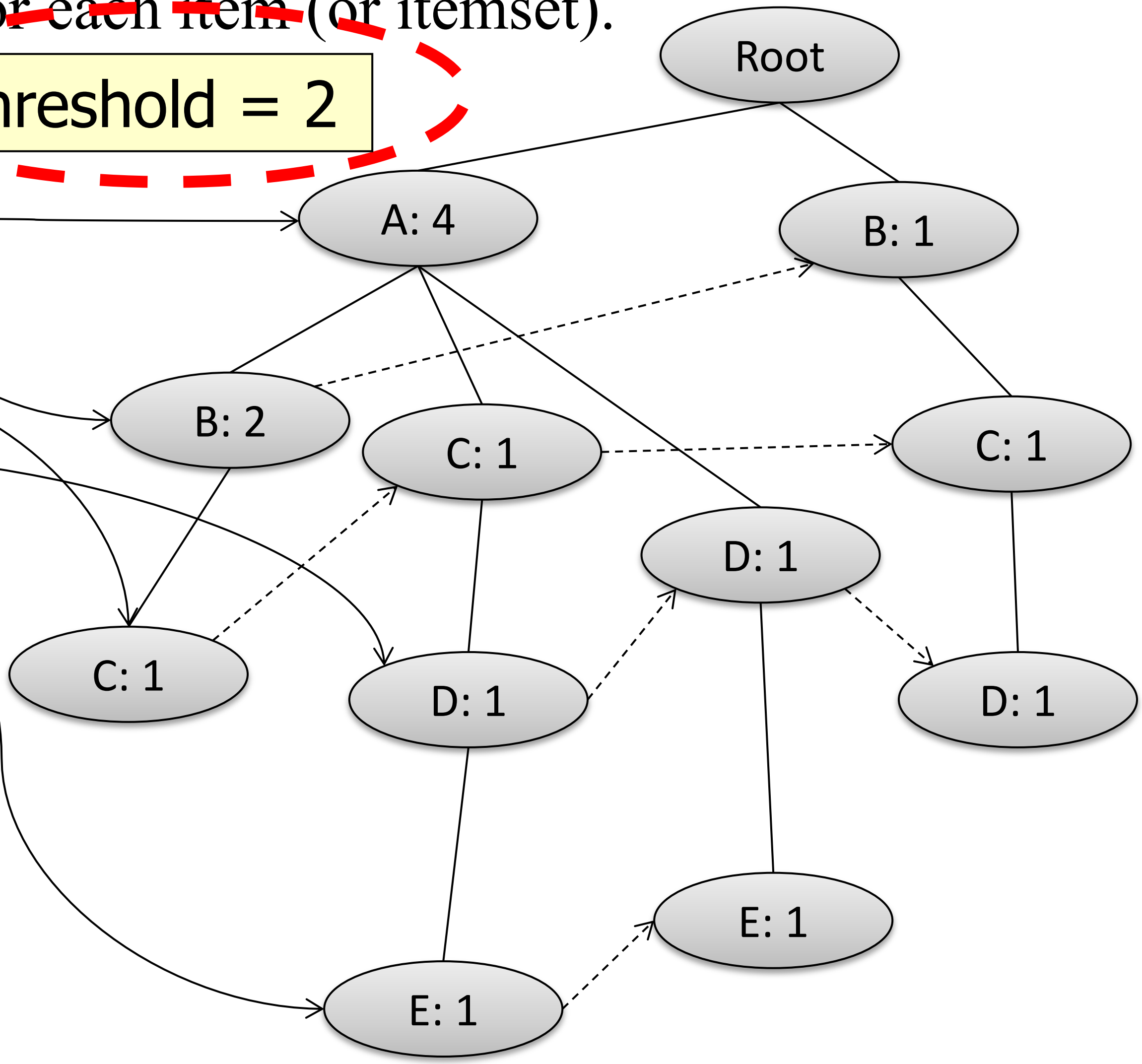
{ (A:1, C:1, D:1, E:1),
(A:1, D:1, E:1), }

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).

Threshold = 2

Item	Head of node-link
A	
B	
C	
D	
E	



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"

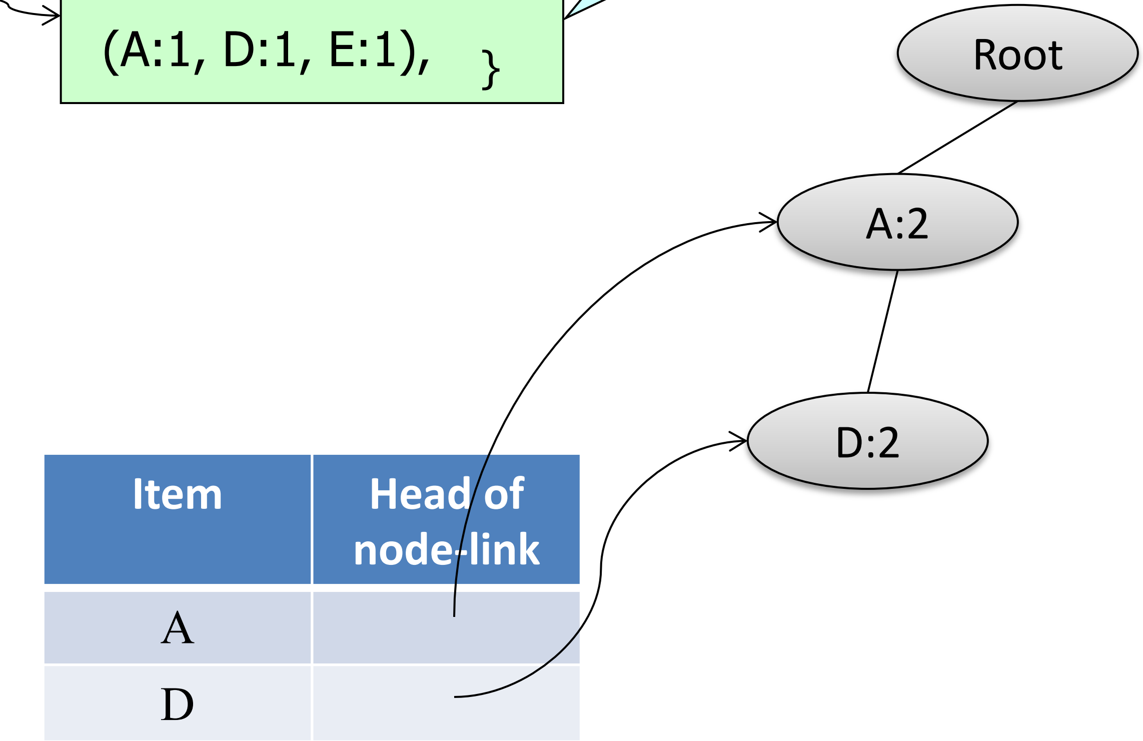
{ (A:1, C:1, D:1, E:1),
(A:1, D:1, E:1), }

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

conditional pattern base of "E"

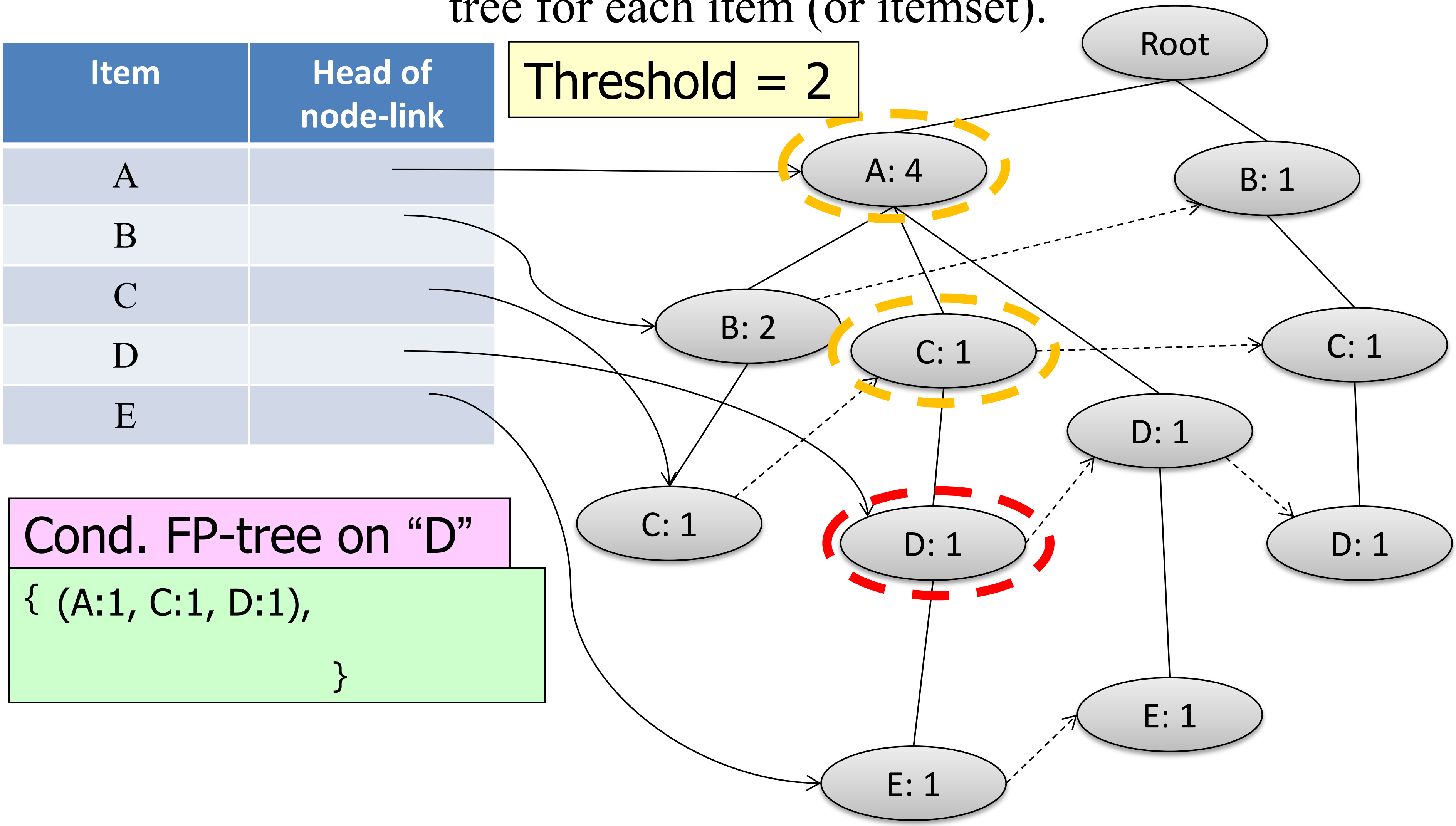
Item	Frequency
A	2
C	1
D	2
E	2

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).



Item	Head of node-link
A	
B	
C	
D	
E	

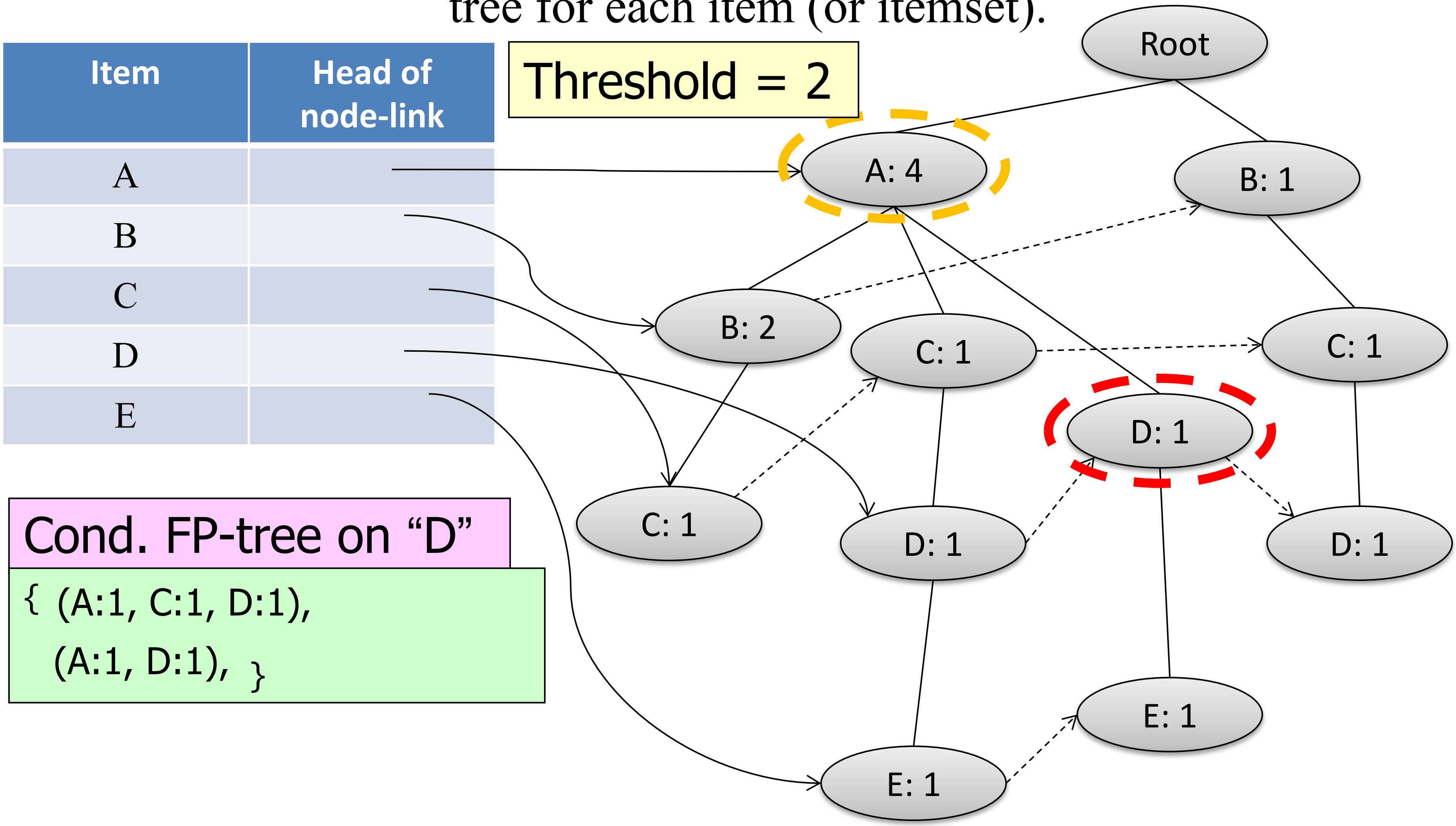
Threshold = 2

Cond. FP-tree on "D"

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

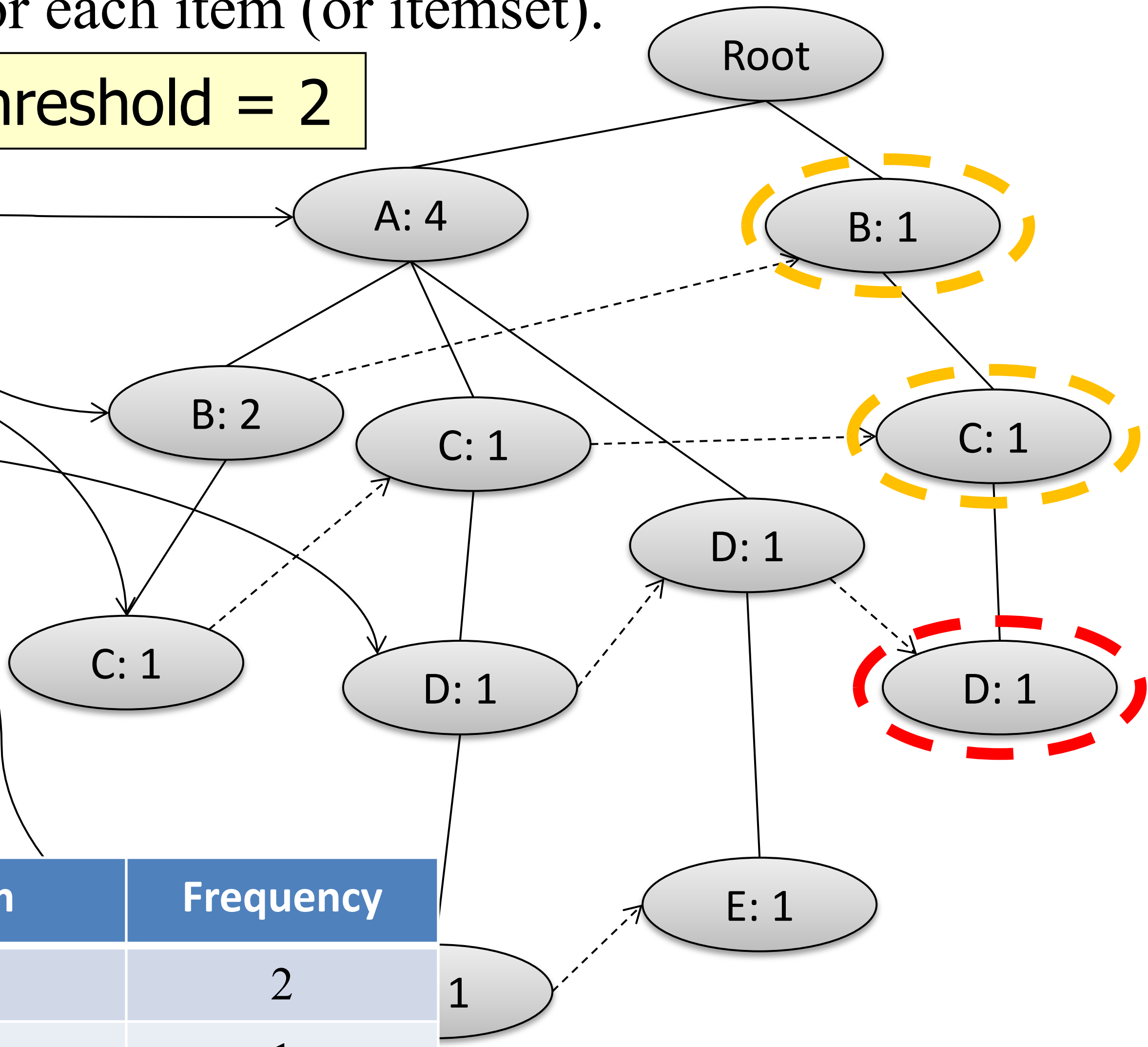
Cond. FP-tree on "D"

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

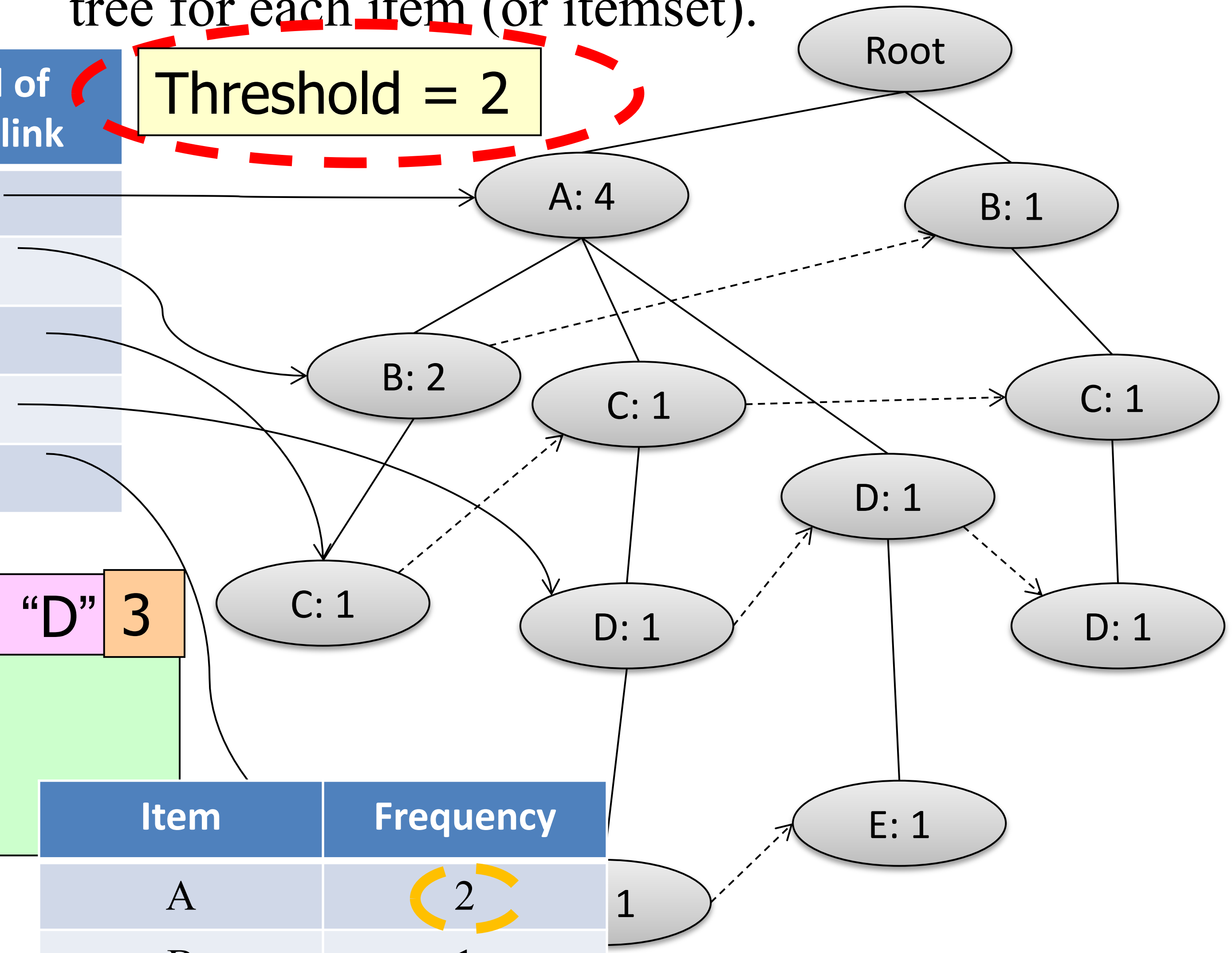
{ (A:1, C:1, D:1),
(A:1, D:1),
(B:1, C:1, D:1), }

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).

Threshold = 2

Item	Head of node-link
A	
B	
C	
D	
E	



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

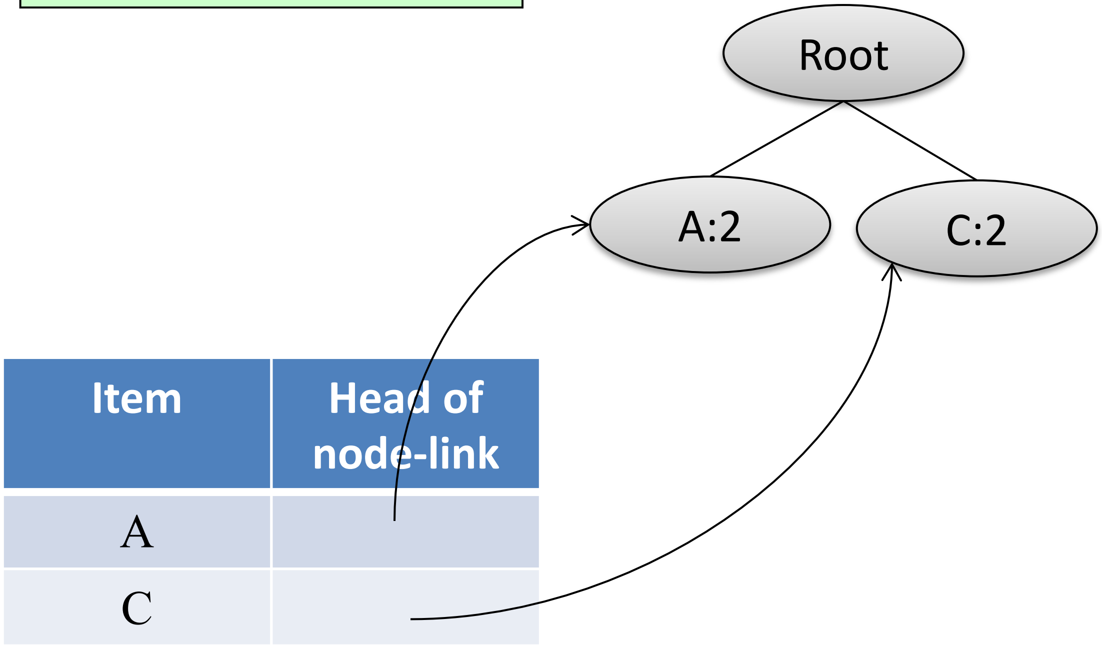
Cond. FP-tree on "D"

{ (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

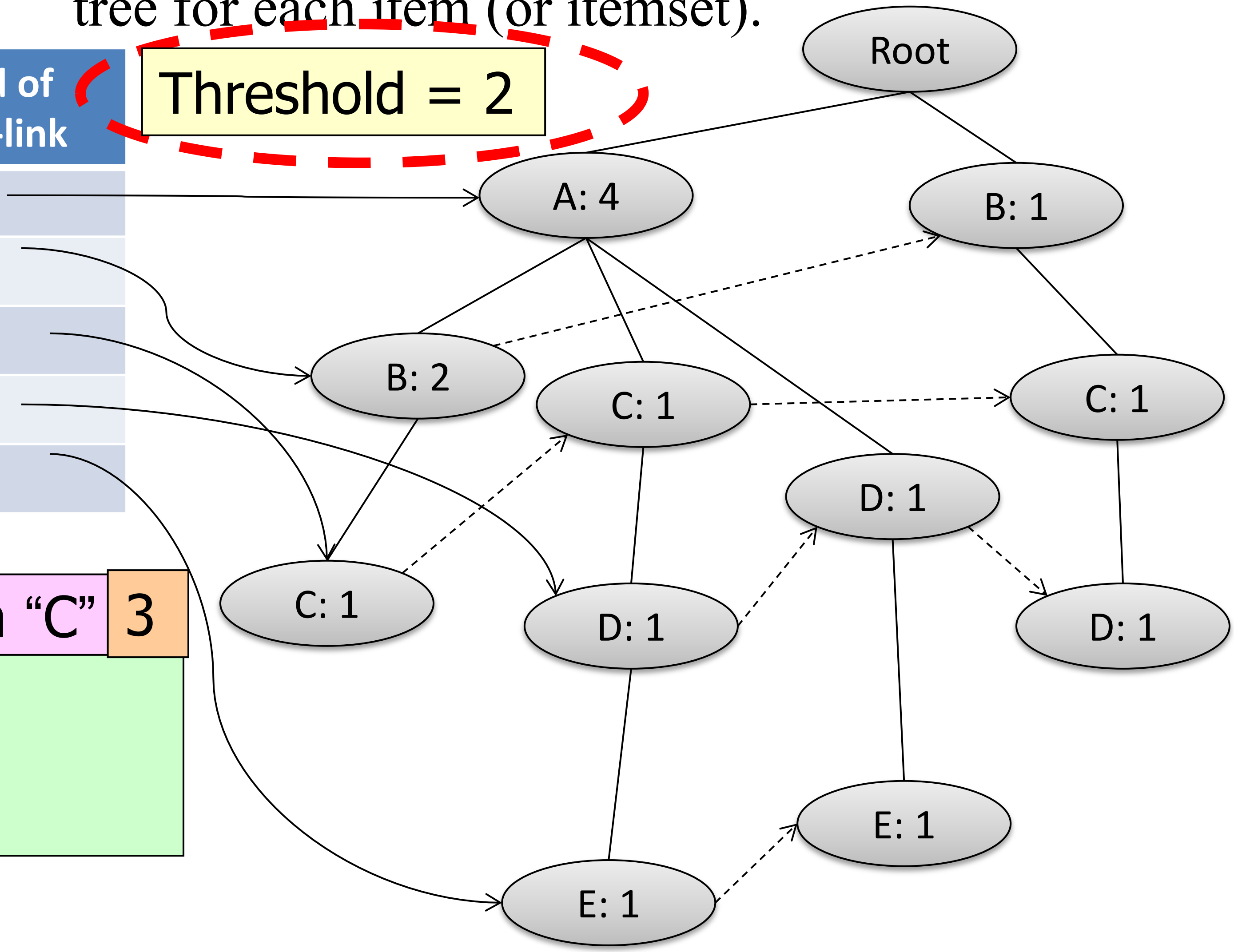


Item	Head of node-link
A	
C	

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).

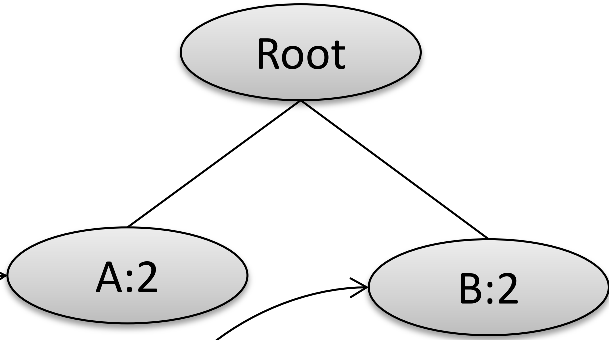
Threshold = 2

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

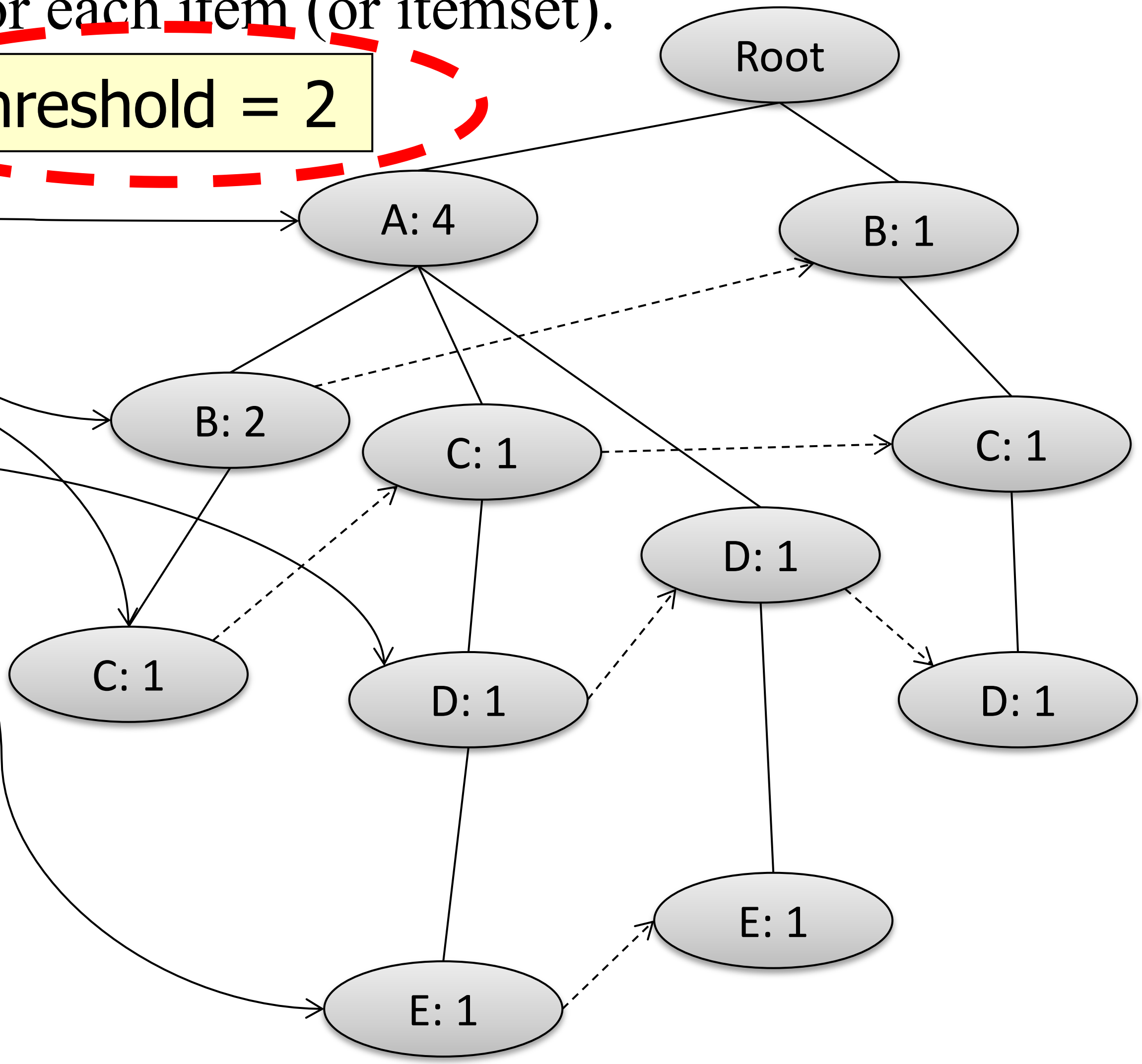


Item	Head of node-link
A	
B	

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

Threshold = 2

Item	Head of node-link
A	
B	
C	
D	
E	



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).

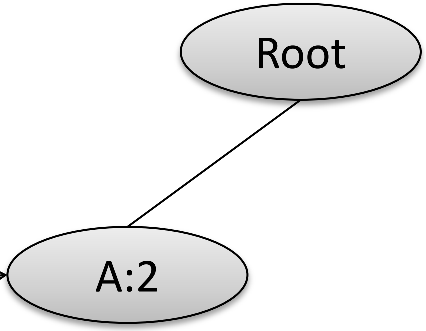
Threshold = 2

Cond. FP-tree on “B”

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

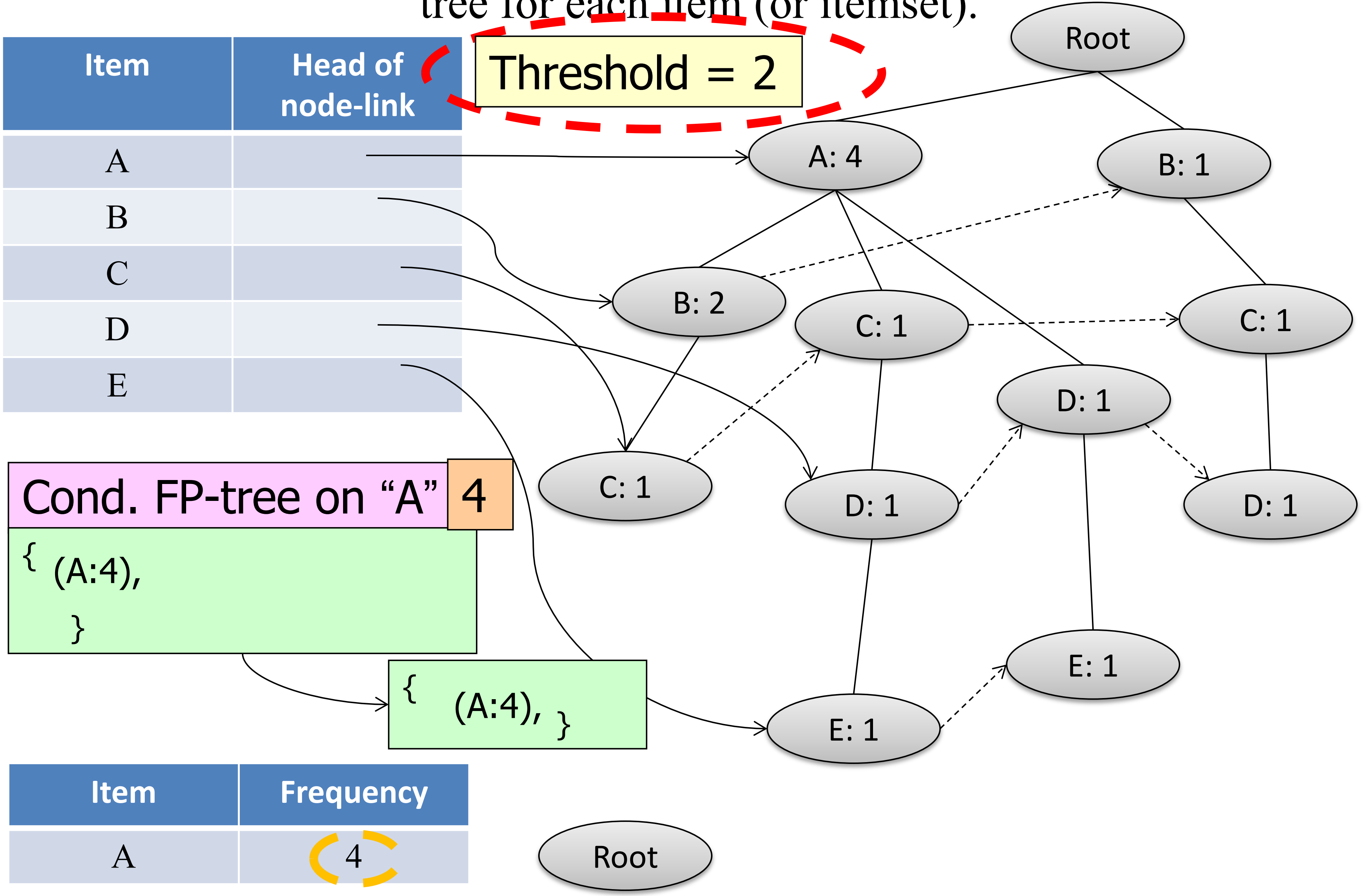
{(A:2, B:2),
(B:1), }

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).



Threshold = 2

Item	Head of node-link
A	
B	
C	
D	
E	

Cond. FP-tree on “A” 4

{ (A:4),
}

{ (A:4), }

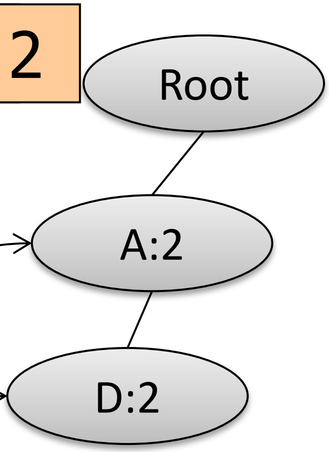
Item	Frequency
A	4

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.

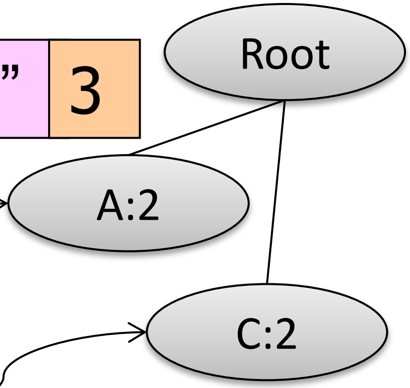
Cond. FP-tree on “E” 2

Item	Head of node-link
A	
D	



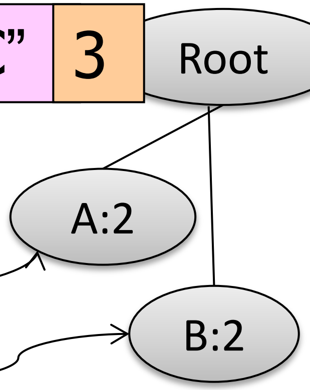
Cond. FP-tree on “D” 3

Item	Head of node-link
A	
C	



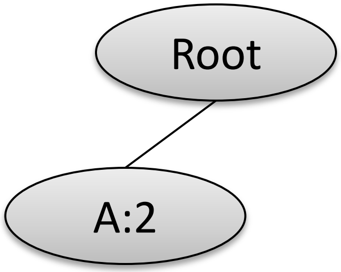
Cond. FP-tree on “C” 3

Item	Head of node-link
A	
B	

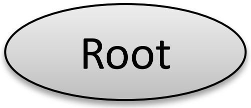


Cond. FP-tree on “B” 3

Item	Head of node-link
A	



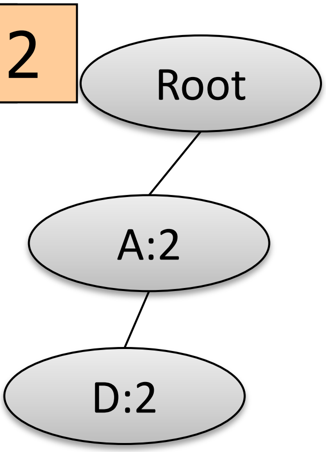
Cond. FP-tree on “A” 4



Cond. FP-tree on “E” 2

1. Before generating this cond. tree, we generate {E} (support = 2)

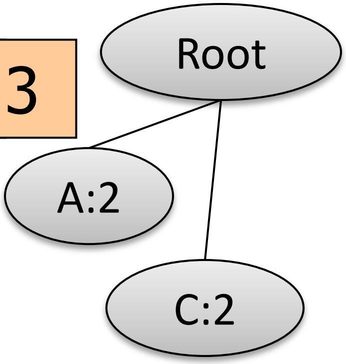
2. After generating this cond. tree, we generate {A, D, E}, {A, E}, {D, E} (support = 2)



Cond. FP-tree on “D” 3

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

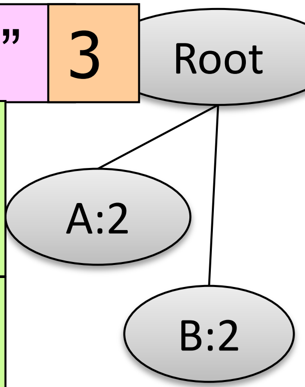
2. After generating this cond. tree, we generate {A, D}, {C, D} (support = 2)



Cond. FP-tree on “C” 3

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

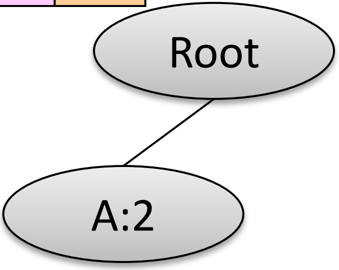
2. After generating this cond. tree, we generate {A, C}, {B, C}(support = 2)



Cond. FP-tree on “B” 3

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

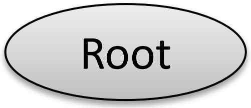
2. After generating this cond. tree, we generate {A, B} (support = 2)



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

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

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

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