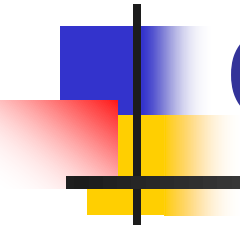


COMP1942

Association Rule Mining (Concepts)

Prepared by Raymond Wong
Presented by Raymond Wong
raywong@cse

Introduction

Supermarket Application

Item

History or **Transaction**

Raymond	 apple	 coke	 coffee	
David	 diaper	 coke	…	
Emily	 milk	 biscuit	…	
Derek	 coke	 milk	…	 diaper  beer

We want to find some **associations** between items.

An interesting association:

Diaper and **Beer** are usually bought together.

Why? Is it strange?

Introduction

Supermarket Application

An interesting association:

Diaper and **Beer** are usually bought together.

Why? Is it strange?

diaper

beer

Introduction

Supermarket Application

An interesting association:

Diaper and **Beer** are usually bought together.

Why? Is it strange?

diaper

beer

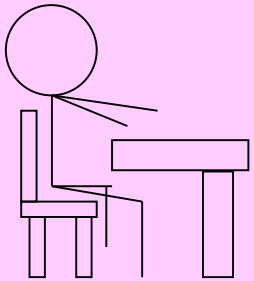
Reasons:

This pattern occurs frequently in the **early evening**.

Daytime

Office

Working…



Introduction

Supermarket Application

An interesting association:

Diaper and **Beer** are usually bought together.

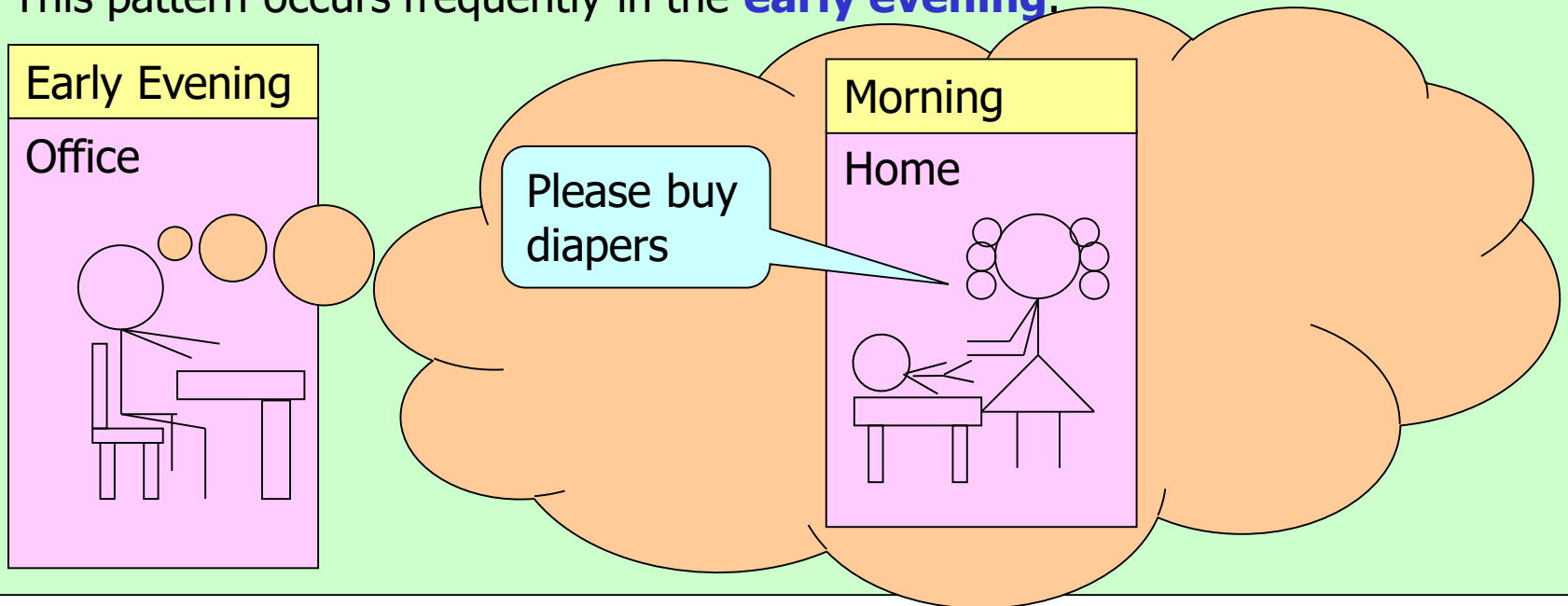
Why? Is it strange?

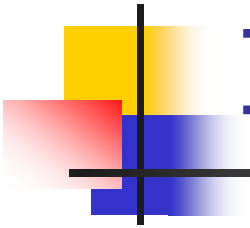
diaper

beer

Reasons:

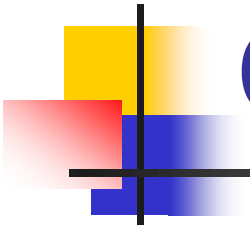
This pattern occurs frequently in the **early evening**.





Introduction

- Applications of Association Rule Mining
  - Supermarket
  - Web Mining
  - Medical analysis
  - Bioinformatics
  - Network analysis
    (e.g., Denial-of-service (DoS))
  - Programming Pattern Finding



Outline

- Association Rule Mining
  - Problem Definition
- Algorithm
- How to use the data mining tool

Association Rule Mining

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

A, D

A, B, D, E

B, C

A, B, C, D, E

B, C, E

Single Items (or simply items): A B C D E

Itemsets: {B, C} {A, B, C} {B, C, D} {A}

2-itemset 3-itemset 3-itemset 1-itemset

Association Rule Mining

Large itemsets:
itemsets with support >= a threshold (e.g., 3)

Frequent itemsets

e.g., {A}, {B}, {B, C}
but NOT {A, B, C}

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

Support = 3

Support = 4

Single Items (or simply items): A B C D E

Itemsets: {B, C} {A, B, C} {B, C, D} {A}

1-frequent itemset of size 3

Support = 3

Support = 1

3-frequent itemset of size 2

Association Rule Mining

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

Support = 2

Association rules: {B, C} → E

Association Rule Mining

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

Support = 2

Confidence = 2/3 = 66.7%

Association rules: {B, C} → E

Association Rule Mining

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

Support = 2

Confidence = 2/3 = 66.7%

Association rules:

{B, C} → E

Support = 3

B → C

Association Rule Mining

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

Support = 2

Confidence = 2/3 = 66.7%

Association rules:

{B, C} → E

Support = 3

B → C

Confidence= 3/4 = 75%

Antecedent

Consequent

Association Rule Mining

Association rules with
1. Support >= a threshold (e.g., 3)
2. Confidence >= another threshold (e.g., 50%)

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

Problem:
We want to find some "interesting" association rules

{B, C} → E

Support = 2

Confidence = 2/3 = 66.7%

B → C

Support = 3

Confidence = 3/4 = 75%

…

How can we find all "interesting" association rules?

Step 1: to find all "large" itemsets
(i.e., itemsets with support >= 3)
(e.g., itemset {B, C} has support = 3)

Step 2: to find all "interesting" rules after Step 1
- from all "large" itemsets
find the association rule with confidence >= 50%

Association Rule Mining

- The previous definition is the traditional definition of association rule mining
- The following is a new concept used in the data mining tool
  - **Lift Ratio**

$$\textbf{Lift Ratio} = \frac{\text{Confidence of the rule}}{\text{Expected Confidence of the consequent of the rule}}$$

Association Rule Mining

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

Association rules:

{B, C} → E

Confidence of the rule
= 2/3 = 66.7%

Expected Confidence of the consequent of the rule
= 3/5 = 60%

Lift Ratio of the rule
= 66.7/60 = 1.11

Association Rule Mining

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

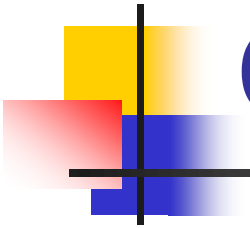
Association rules:

B → C

Confidence of the rule = 3/4 = 75%

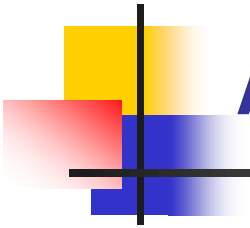
Expected Confidence of the consequent of the rule = 3/5 = 60%

Lift Ratio of the rule = 75/60 = 1.25



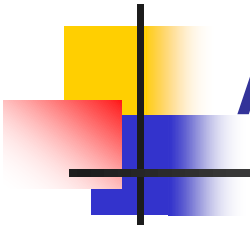
Outline

- Association Rule Mining
  - Problem Definition
- Algorithm
- How to use the data mining tool



Algorithm

- Two major algorithms
  - Apriori Approach
  - FP-growth Approach



Apriori Algorithm

- Properties
- Algorithm

Apriori

Suppose we want to find all "large" itemsets (e.g., itemsets with support >= 3)

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

{B, C} is large

Support of {B, C} = 3

Is {B} large?

Is {C} large?

Property 1: If an itemset S is large, then any proper subset of S must be large.

Apriori

Suppose we want to find all "large" itemsets (e.g., itemsets with support >= 3)

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

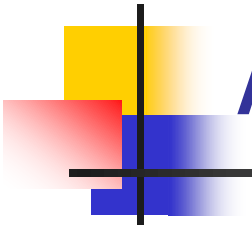
{B, C, E} is NOT large

Support of {B, C, E} = 2

Is {A, B, C, E} large?

Is {B, C, D, E} large?

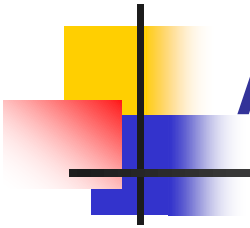
Property 2: If an itemset S is NOT large, then any proper superset of S must NOT be large.



Apriori

Property 1: If an itemset S is large, then any proper subset of S must be large.

Property 2: If an itemset S is NOT large, then any proper superset of S must NOT be large.



Apriori Algorithm

- Properties
- Algorithm

Apriori

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

Item	Count
A	3
B	
C	
D	
E	

Apriori

Suppose we want to find all "large" itemsets (e.g., itemsets with support >= 3)

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

Item	Count
A	3
B	4
C	3
D	3
E	3

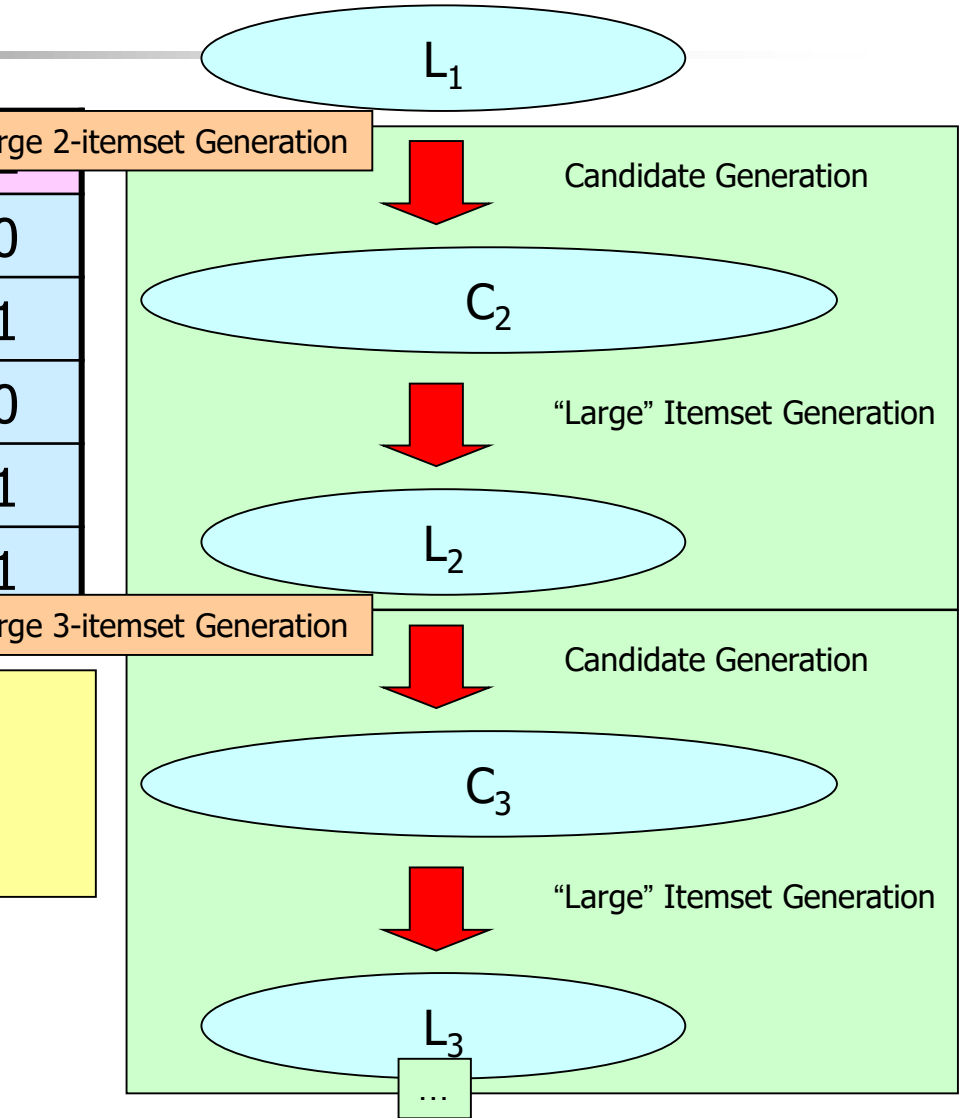
Thus, {A}, {B}, {C}, {D} and {E} are "large" itemsets of size 1 (or, "large" 1-itemsets).

We set L_1 = {{A}, {B}, {C}, {D}, {E}}

Apriori

Suppose we want to find all "large" itemsets (e.g., itemsets with support >= 3)

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1



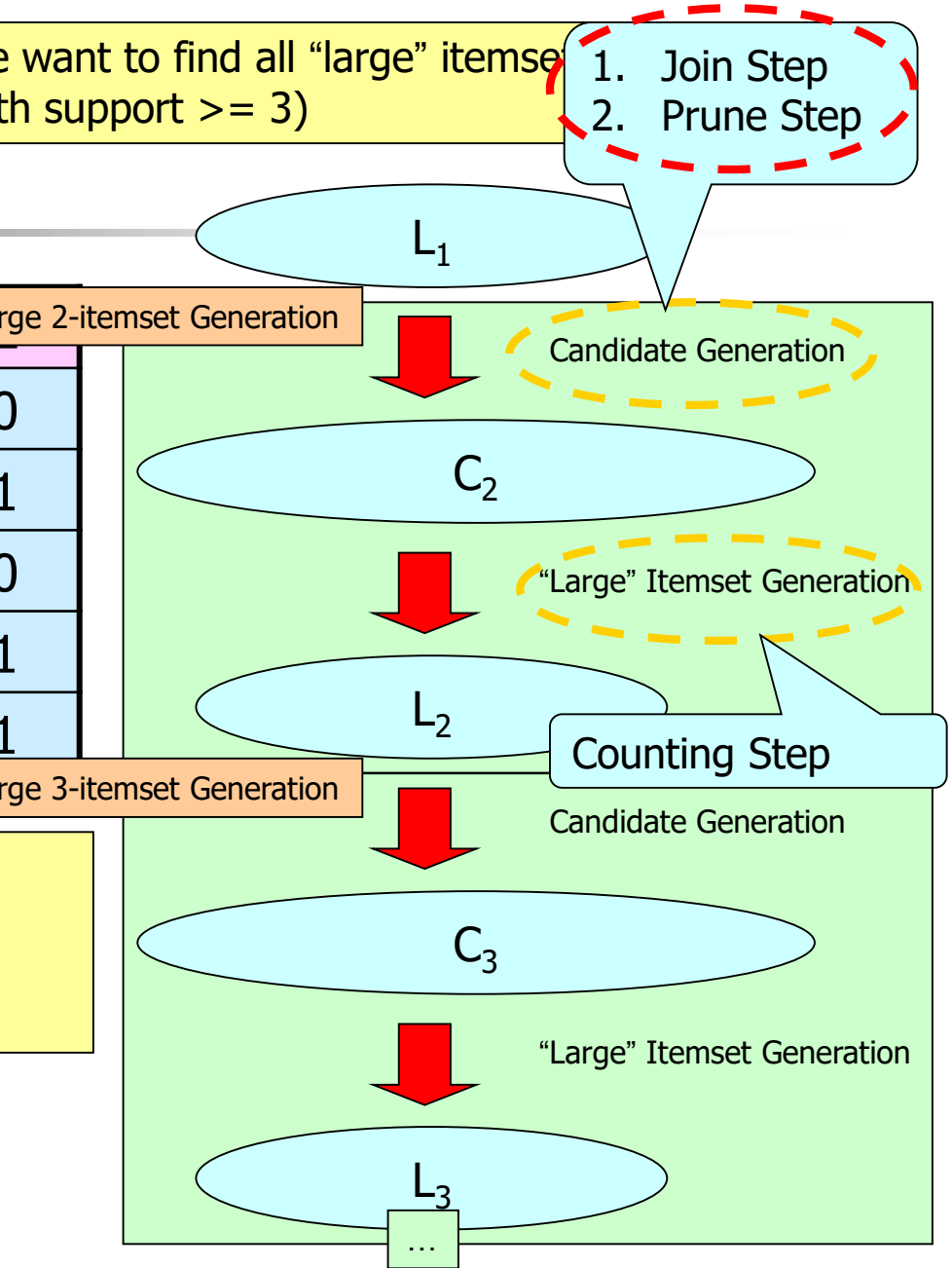
Thus, {A}, {B}, {C}, {D} and {E} are "large" itemsets of size 1 (or, "large" 1-itemsets).

We set L_1 = {{A}, {B}, {C}, {D}, {E}}

Apriori

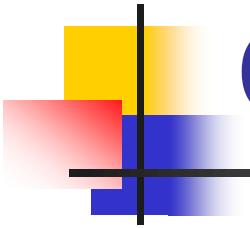
Suppose we want to find all "large" itemse
itemsets with support >= 3)

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1



Thus, {A}, {B}, {C}, {D} and {E} are "large" itemsets of size 1 (or, "large" 1-itemsets).

We set L_1 = {{A}, {B}, {C}, {D}, {E}}



Candidate Generation

- Join Step
- Prune Step

Property 1: If an itemset S is large, then any proper subset of S must be large.

Property 2: If an itemset S is NOT large, then any proper superset of S must NOT be large.

Join Step

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

Suppose we know that itemset {B, C} and itemset {B, E} are large (i.e., L_2).

It is possible that itemset {B, C, E} is also large (i.e., C_3).

Join Step

- Join Step
  - Input: L_{k-1}, a set of all large (k-1)-itemsets
  - Output: C_k, a set of candidates k-itemsets
  - Algorithm:
    - insert into C_k
      select p.item$_1$, p.item$_2$, …, p.item$_{k-1}$, q.item$_{k-1}$
      from L_{k-1} p, L_{k-1} q
      where p.item$_1$ = q.item$_1$,
      p.item$_2$ = q.item$_2$,
      …
      p.item$_{k-2}$ = q.item$_{k-2}$,
      p.item$_{k-1}$ < q.item$_{k-1}$

Property 1: If an itemset S is large, then any proper subset of S must be large.

Property 2: If an itemset S is NOT large, then any proper superset of S must NOT be large.

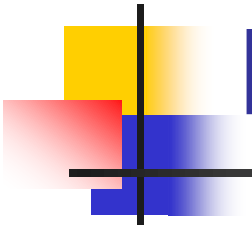
Prune Step

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

Suppose we know that itemset {B, C} and itemset {B, E} are large (i.e., L_2).

It is possible that itemset {B, C, E} is also large (i.e., C_3).

Suppose we know that {C, E} is not large. We can prune {B, C, E} in C_3.



Prune Step

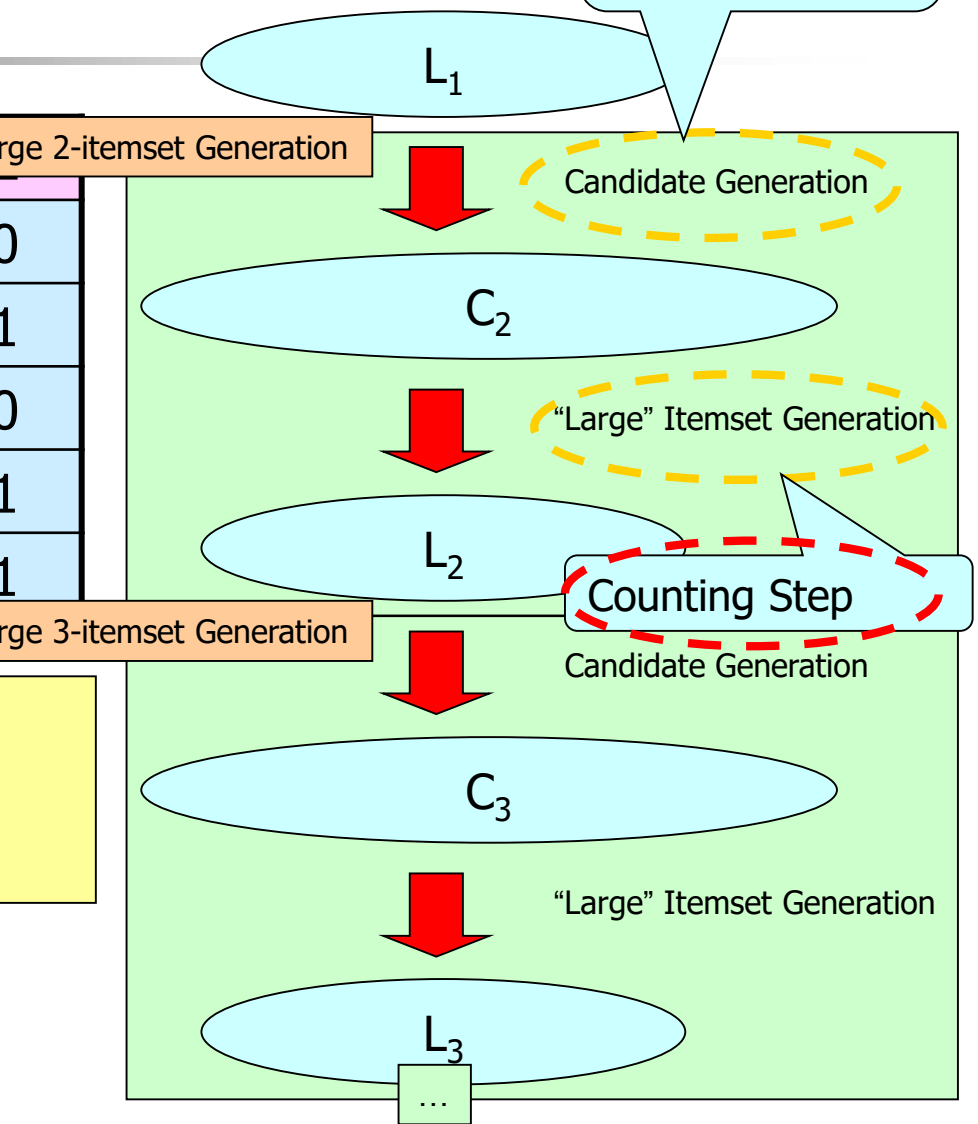
- Prune Step
  - forall itemsets c $\in C_k$ (from Join Step) do
    - for all (k-1)-subsets s of c do
      - if (s not in L_{k-1}) then
        - delete c from C_k

Apriori

Suppose we want to find all "large" itemse
itemsets with support >= 3)

1. Join Step
2. Prune Step

TID	A	B	C	D	E
t1	1	0	0	1	0
t2	1	1	0	1	1
t3	0	1	1	0	0
t4	1	1	1	1	1
t5	0	1	1	0	1

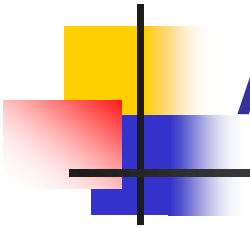


Thus, {A}, {B}, {C}, {D} and {E} are "large" itemsets of size 1 (or, "large" 1-itemsets).

We set L_1 = {{A}, {B}, {C}, {D}, {E}}

Counting Step

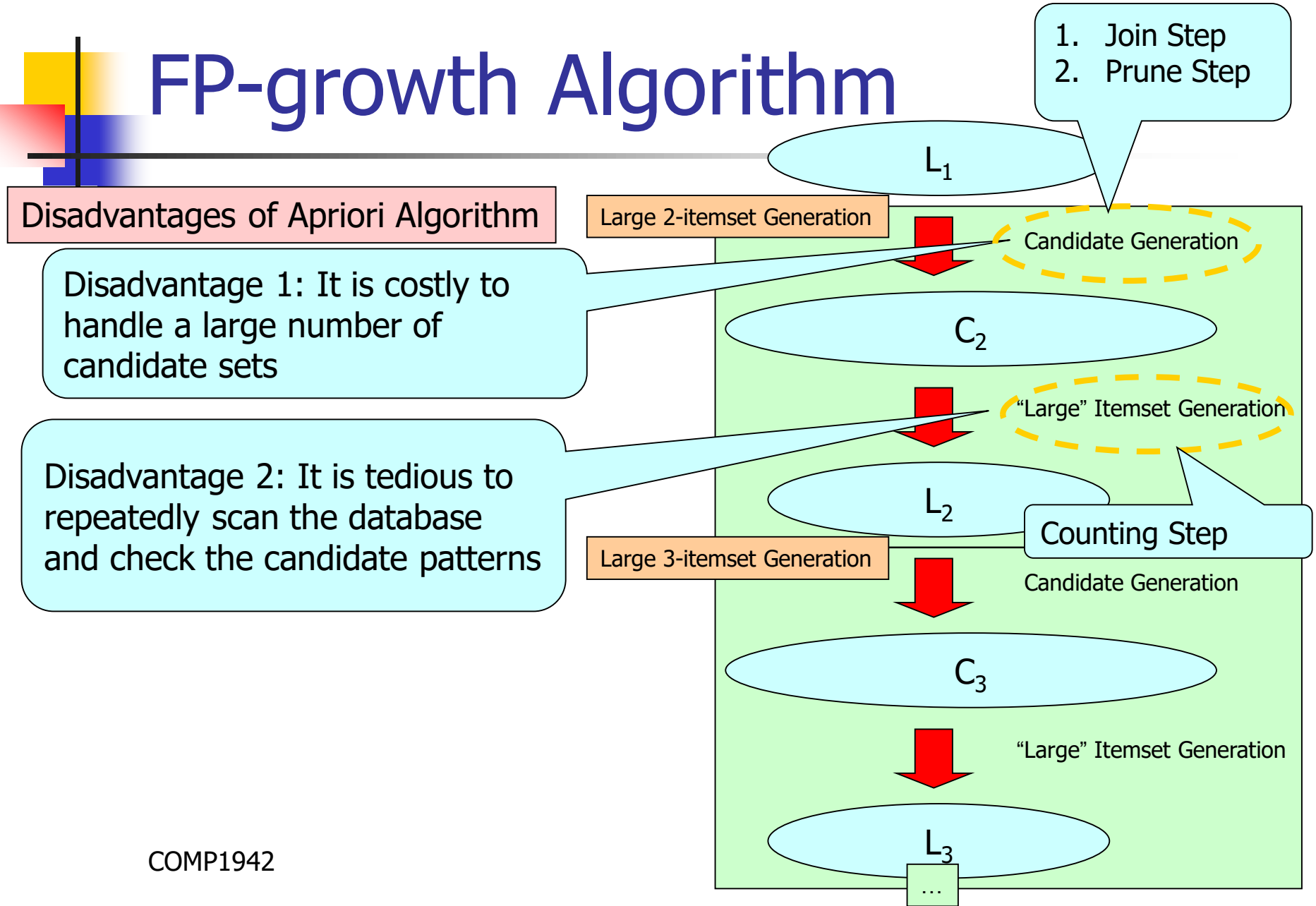
- After the candidate generation (i.e., Join Step and Prune Step), we are given a set of **candidate** itemsets
- We need to **verify** whether these candidate itemsets are large or not
- We have to scan the database to obtain the count of each itemset in the candidate set.
- Algorithm
  - For each itemset c in C_k,
    - obtain the count of c (from the database)
    - If the count of c is smaller than a given threshold,
      - remove it from C_k
  - The remaining itemsets in C_k correspond to L_k



Algorithm

- Two major algorithms
  - Apriori Approach
  - FP-growth Approach

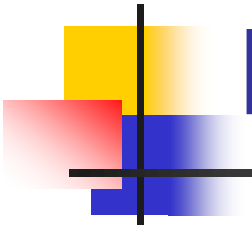
FP-growth Algorithm



Disadvantages of Apriori Algorithm

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

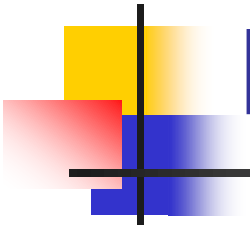


FP-growth Algorithm

- 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

FP-tree

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.

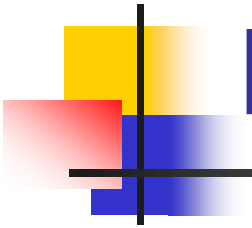


FP-tree

- Frequent Itemset Mining

Problem: to find all “large” (or frequent) itemsets with support at least a threshold (i.e., itemsets with support >= 3)

TID	Items Bought
100	a, b, c, d, e, f, g, h
200	a, f, g
300	b, d, e, f, j
400	a, b, d, i, k
500	a, b, e, g



FP-tree

TID	Items Bought
100	a, b, c, d, e, f, g, h
200	a, f, g
300	b, d, e, f, j
400	a, b, d, i, k
500	a, b, e, g

TID	Items Bought
100	a, b, c, d, e, f, g, h
200	a, f, g
300	b, d, e, f, j
400	a, b, d, i, k
500	a, b, e, g

TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	
200	a, f, g	
300	b, d, e, f, j	
400	a, b, d, i, k	
500	a, b, e, g	

Threshold = 3

Item	Frequency
a	4
b	
c	
d	
e	
f	
g	
h	
i	
j	
k	

TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	
200	a, f, g	
300	b, d, e, f, j	
400	a, b, d, i, k	
500	a, b, e, g	

Threshold = 3

Item	Frequency
a	4
b	4
c	1
d	3
e	3
f	3
g	3
h	1
i	1
j	1
k	1

TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	
200	a, f, g	
300	b, d, e, f, j	
400	a, b, d, i, k	
500	a, b, e, g	

Threshold = 3

Item	Frequency
a	4
b	4
c	1
d	3
e	3
f	3
g	3
h	1
i	1
j	1
k	1

Item	Frequency
a	4
b	4
d	3
e	3
f	3
g	3

TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

Item	Frequency
a	4
b	4
c	1
d	3
e	3
f	3
g	3
h	1
i	1
j	1
k	1

Item	Frequency
a	4
b	4
d	3
e	3
f	3
g	3

FP-tree

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

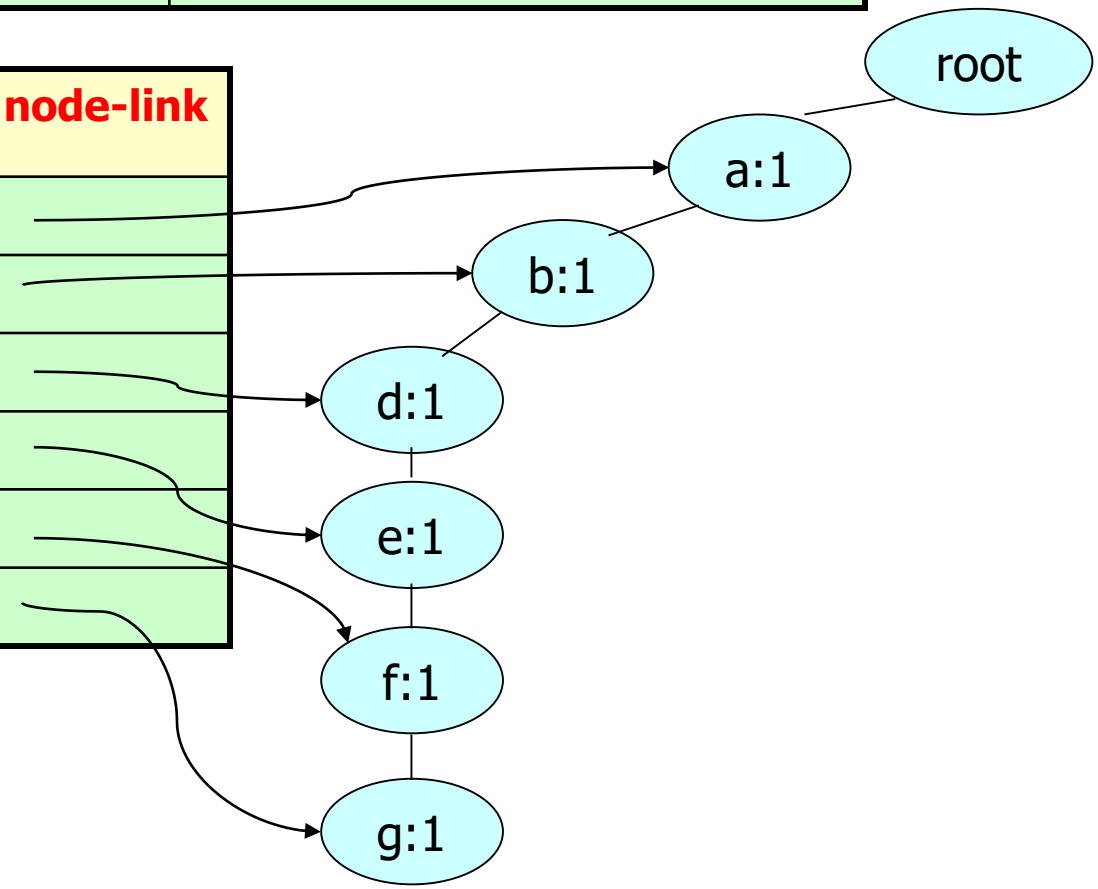
Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).

Step 4: Determine the frequent patterns.

TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

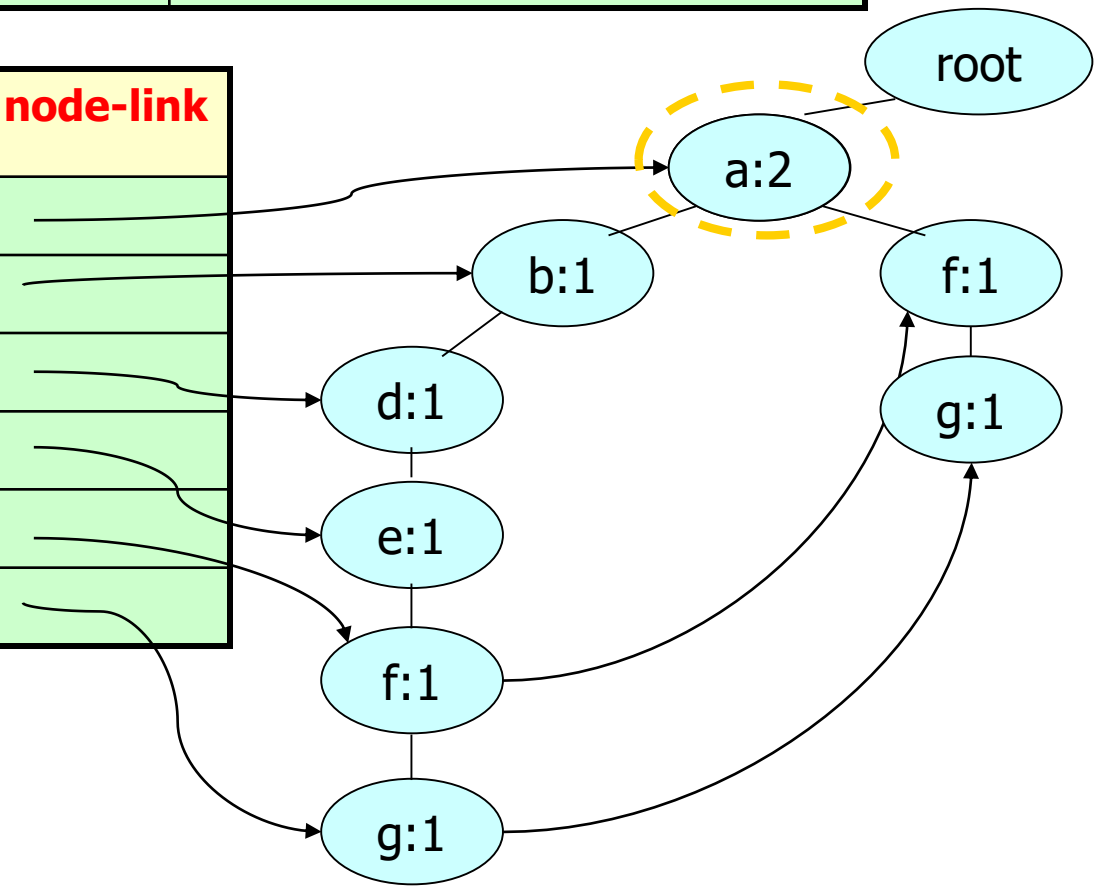
Item	Head of node-link
a	
b	
d	
e	
f	
g	



TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

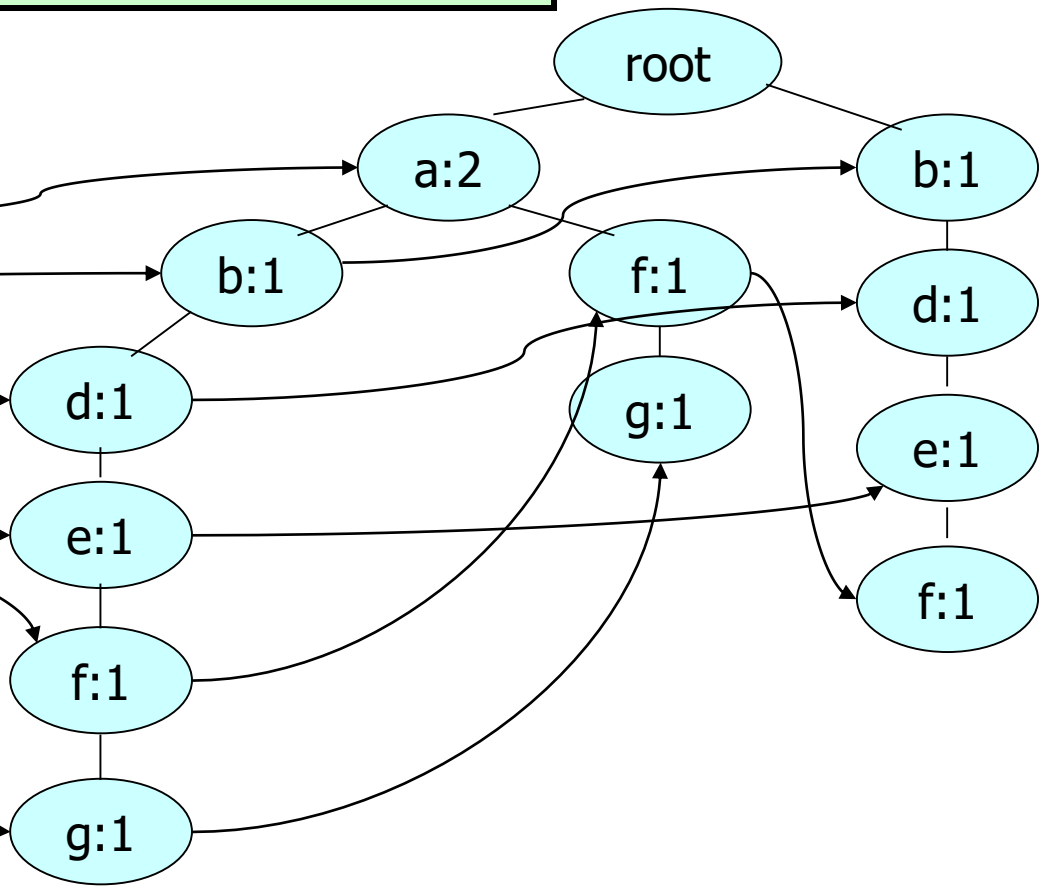
Item	Head of node-link
a	
b	
d	
e	
f	
g	



TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

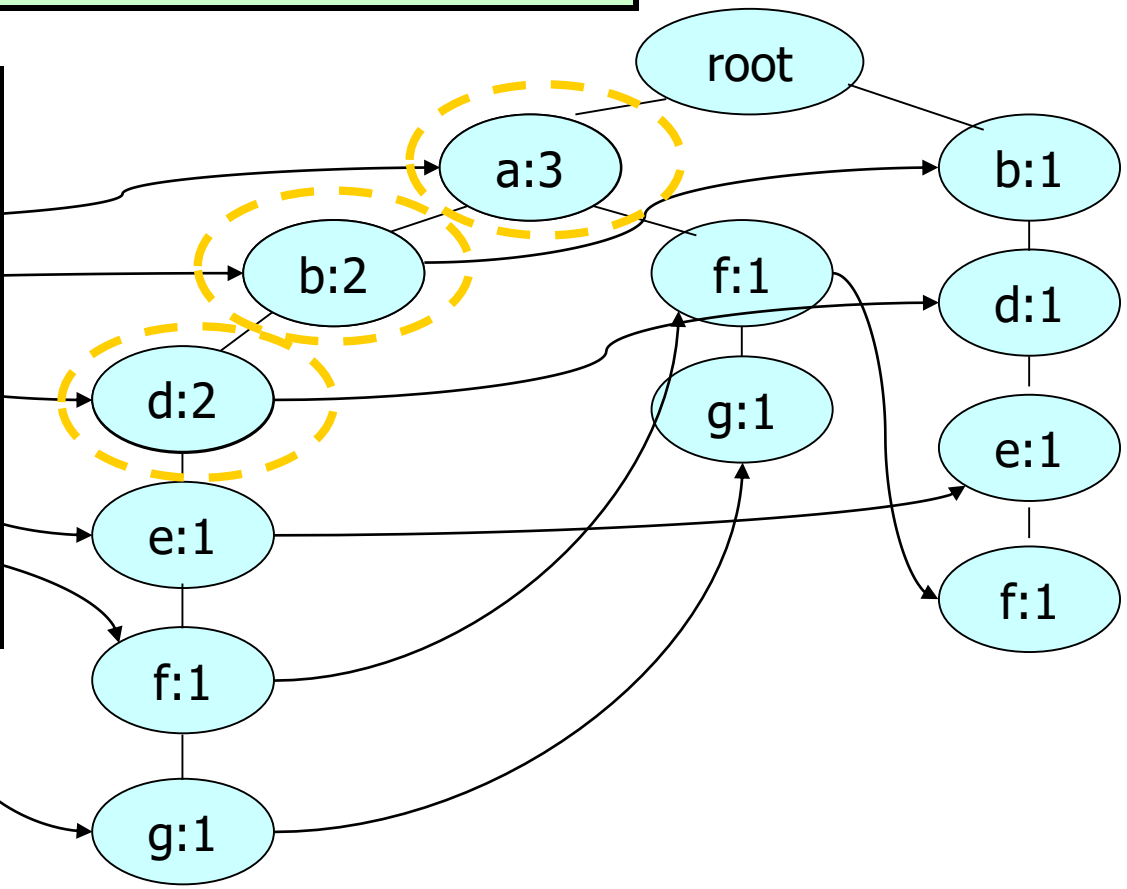
Item	Head of node-link
a	
b	
d	
e	
f	
g	



TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

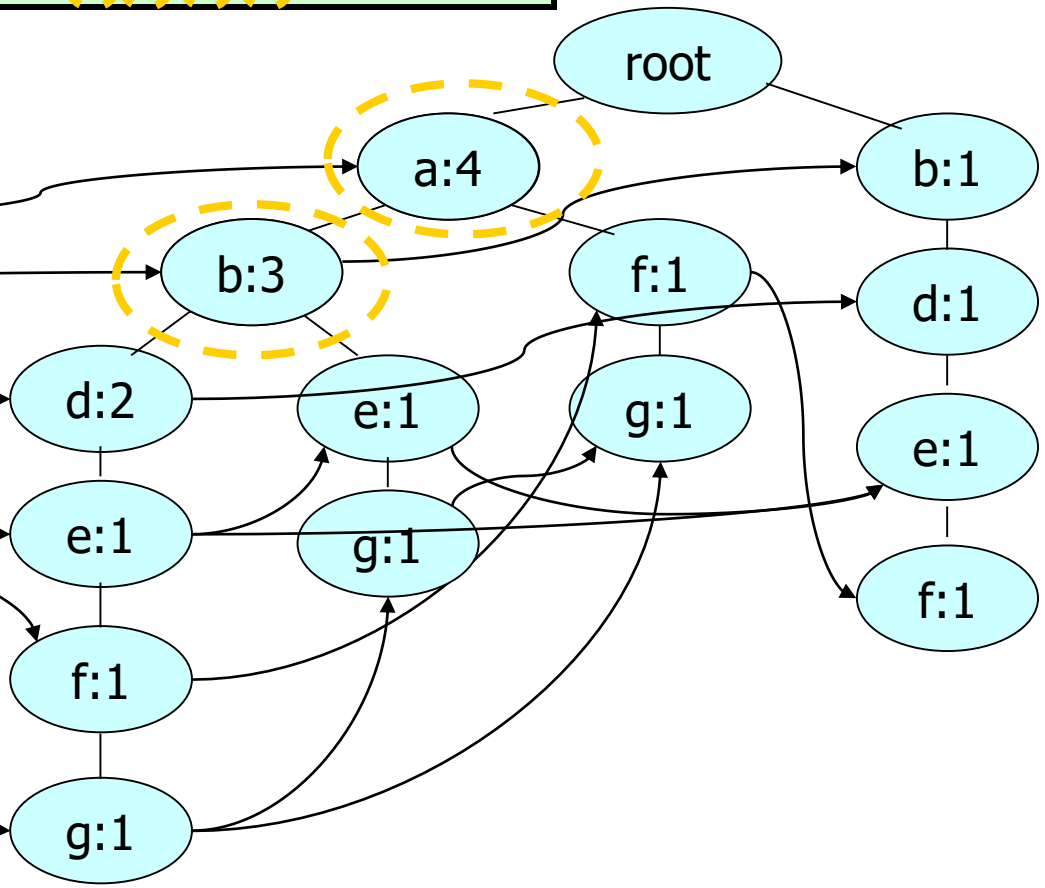
Item	Head of node-link
a	
b	
d	
e	
f	
g	



TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

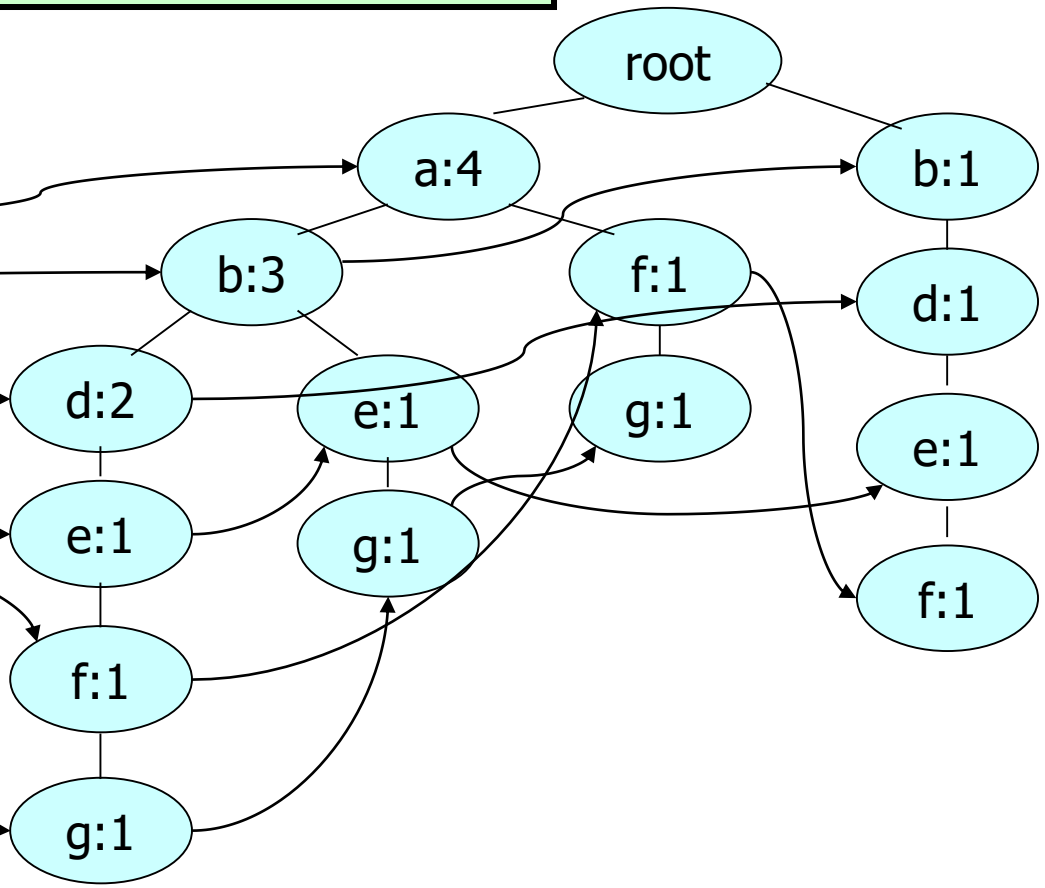
Item	Head of node-link
a	
b	
d	
e	
f	
g	



TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	



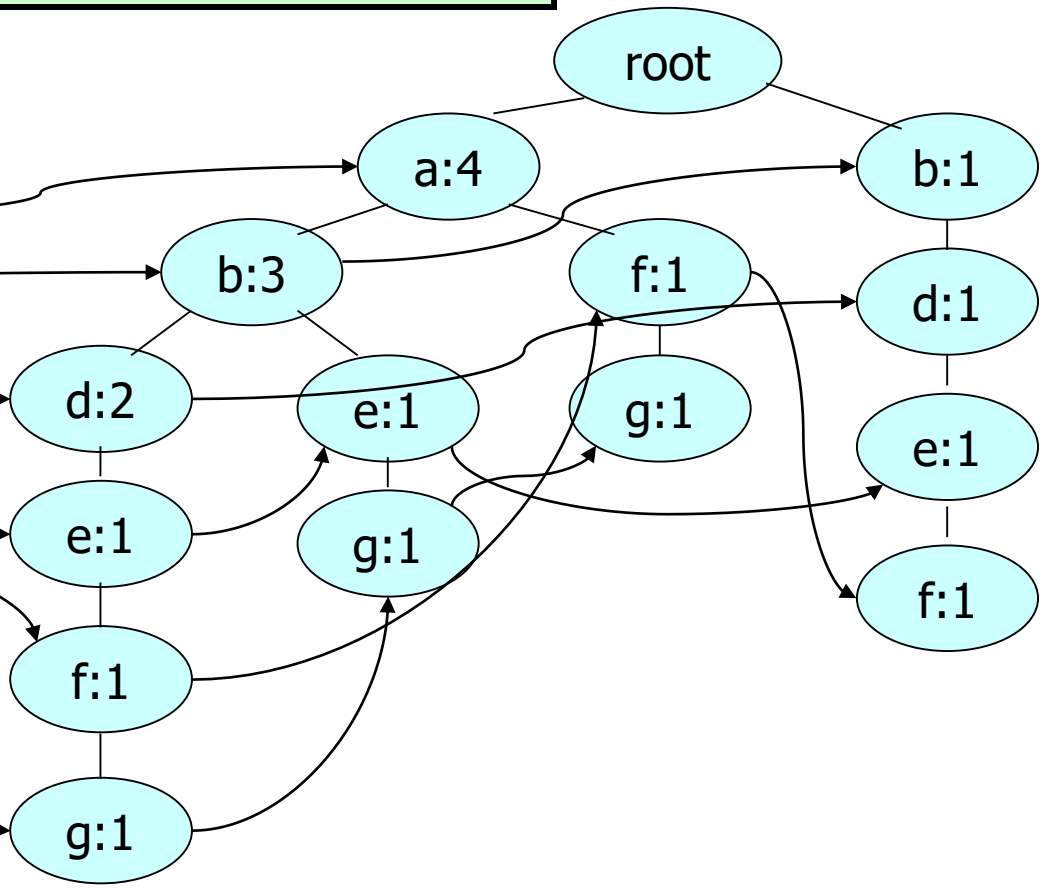
FP-tree

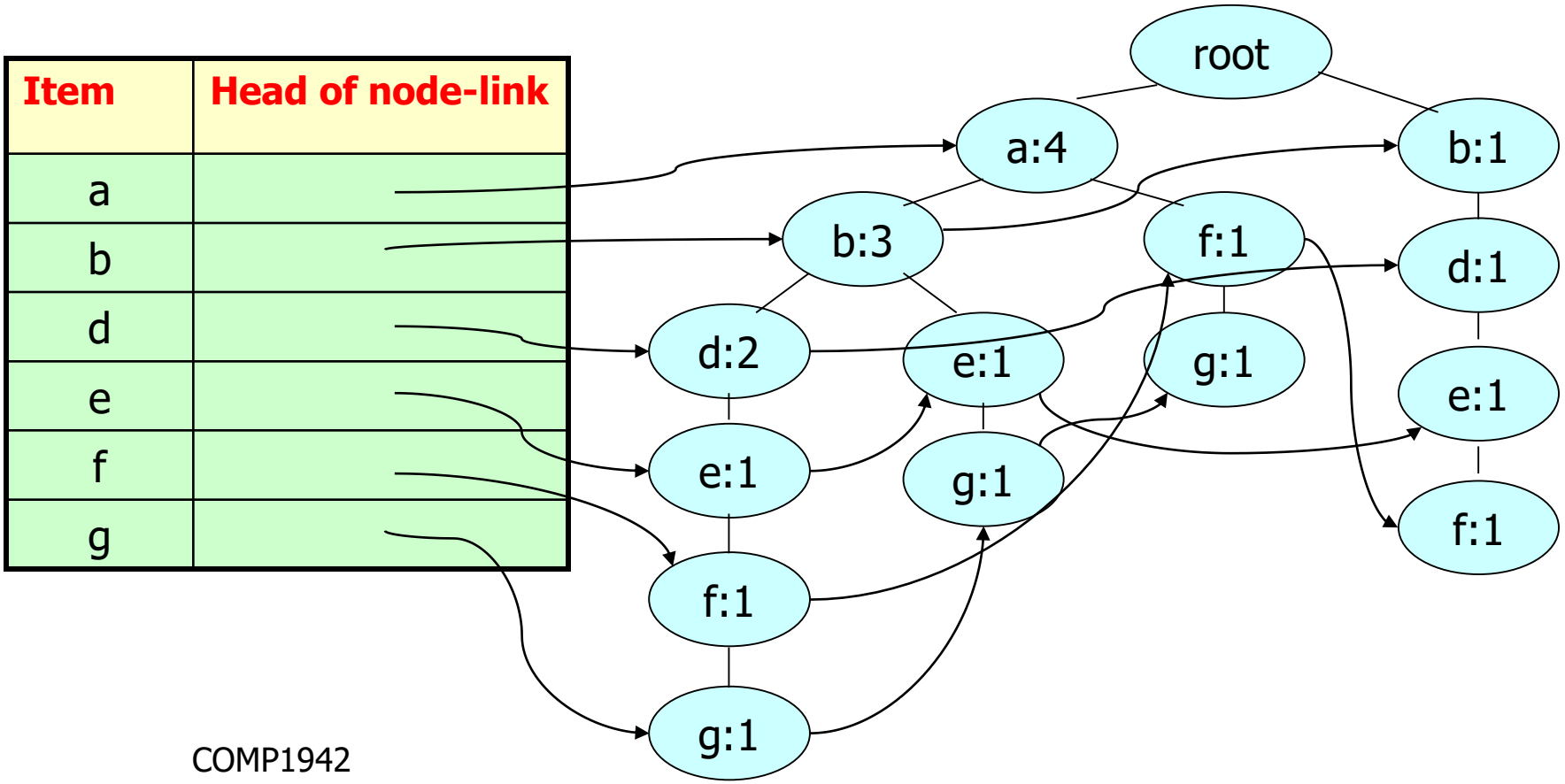
Step 1: Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
Step 2: Construct the FP-tree from the above data
Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
Step 4: Determine the frequent patterns.

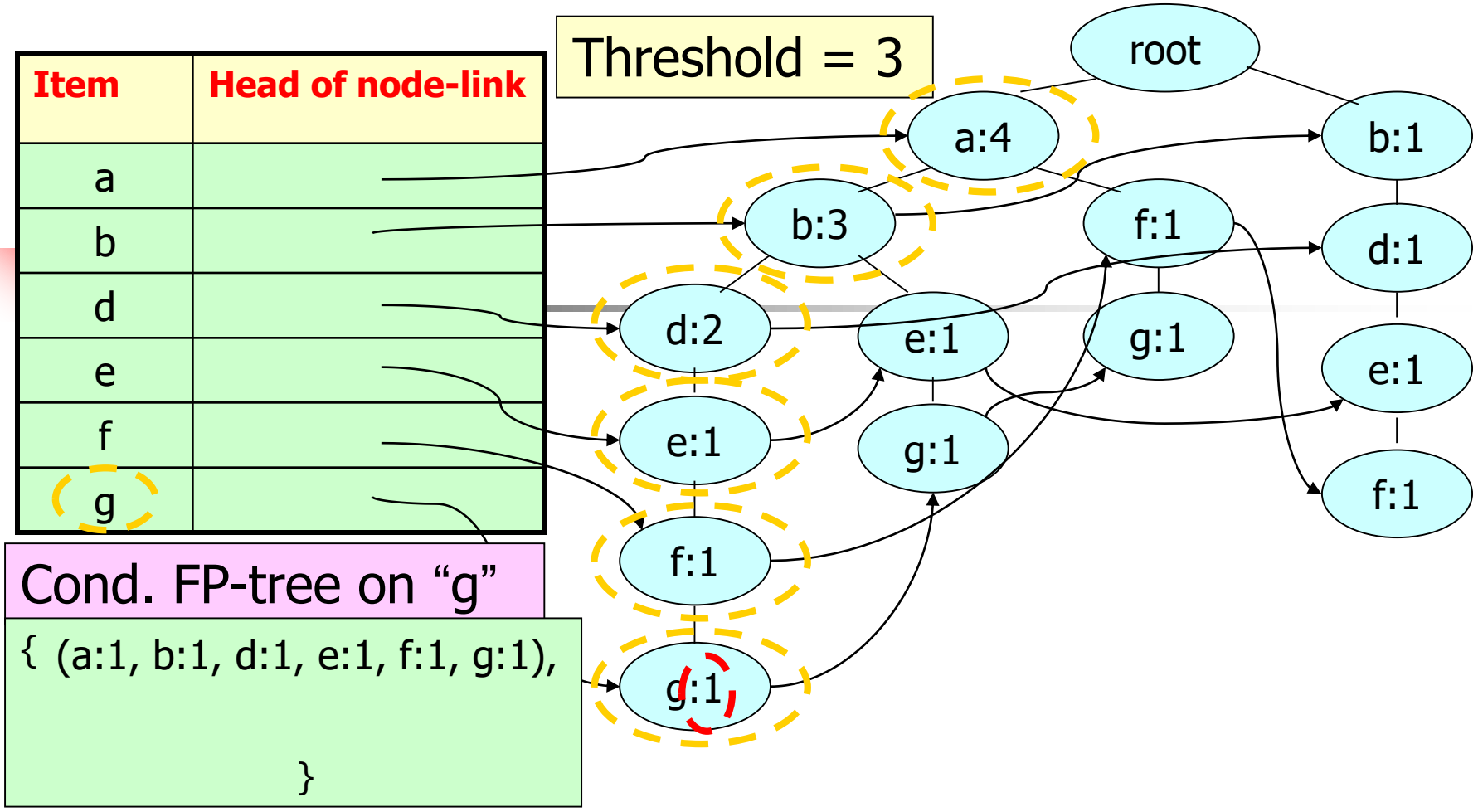
TID	Items Bought	(Ordered) Frequent Items
100	a, b, c, d, e, f, g, h	a, b, d, e, f, g
200	a, f, g	a, f, g
300	b, d, e, f, j	b, d, e, f
400	a, b, d, i, k	a, b, d
500	a, b, e, g	a, b, e, g

Threshold = 3

Item	Head of node-link
a	
b	
d	
e	
f	
g	

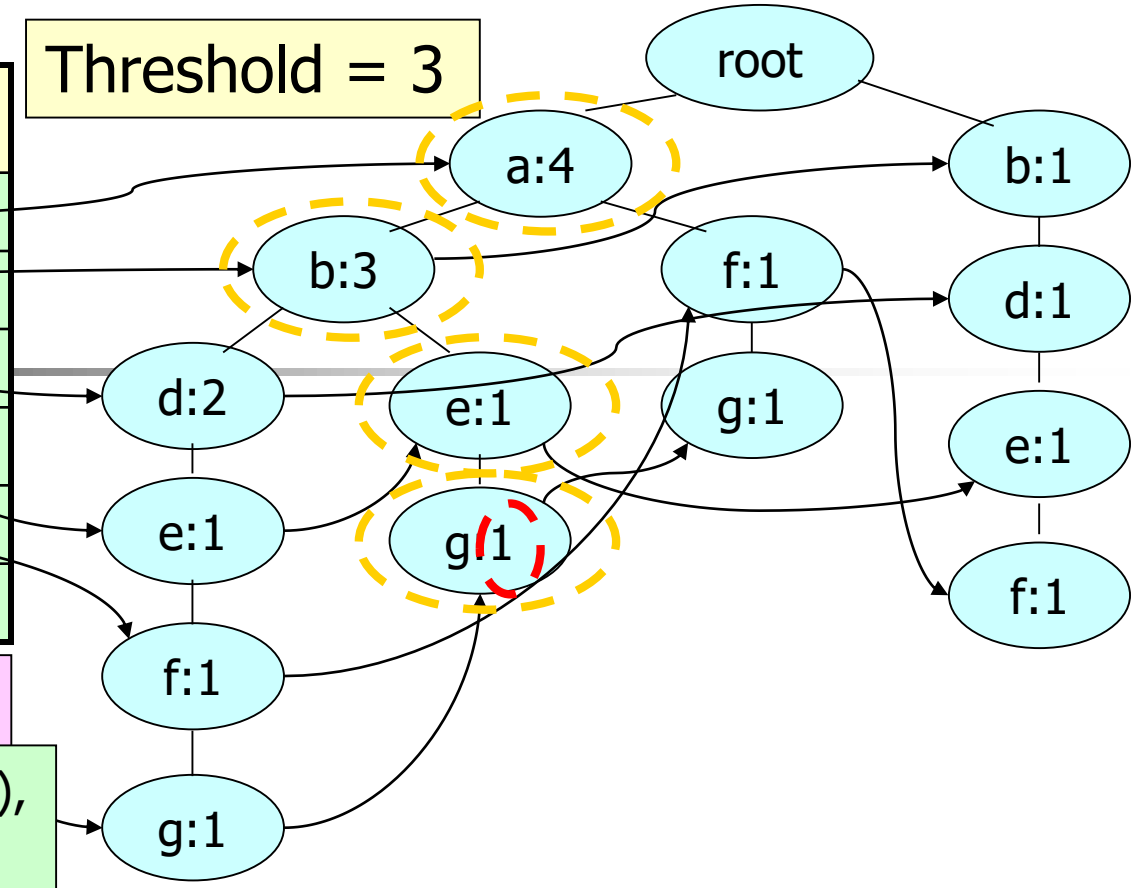



Item
Head of node-link
a
b
d
e
f
g
root
a:4
b:1
b:3
f:1
d:1
d:2
e:1
g:1
e:1
e:1
g:1
f:1
f:1
g:1


Item
Head of node-link
a
b
d
e
f
g
Threshold = 3
root
a:4
b:1
b:3
f:1
d:1
d:2
e:1
g:1
e:1
e:1
g:1
f:1
f:1
g:1
Cond. FP-tree on “g”
{ (a:1, b:1, d:1, e:1, f:1, g:1),
}

Item	Head of node-link
a	
b	
d	
e	
f	
g	

Threshold = 3

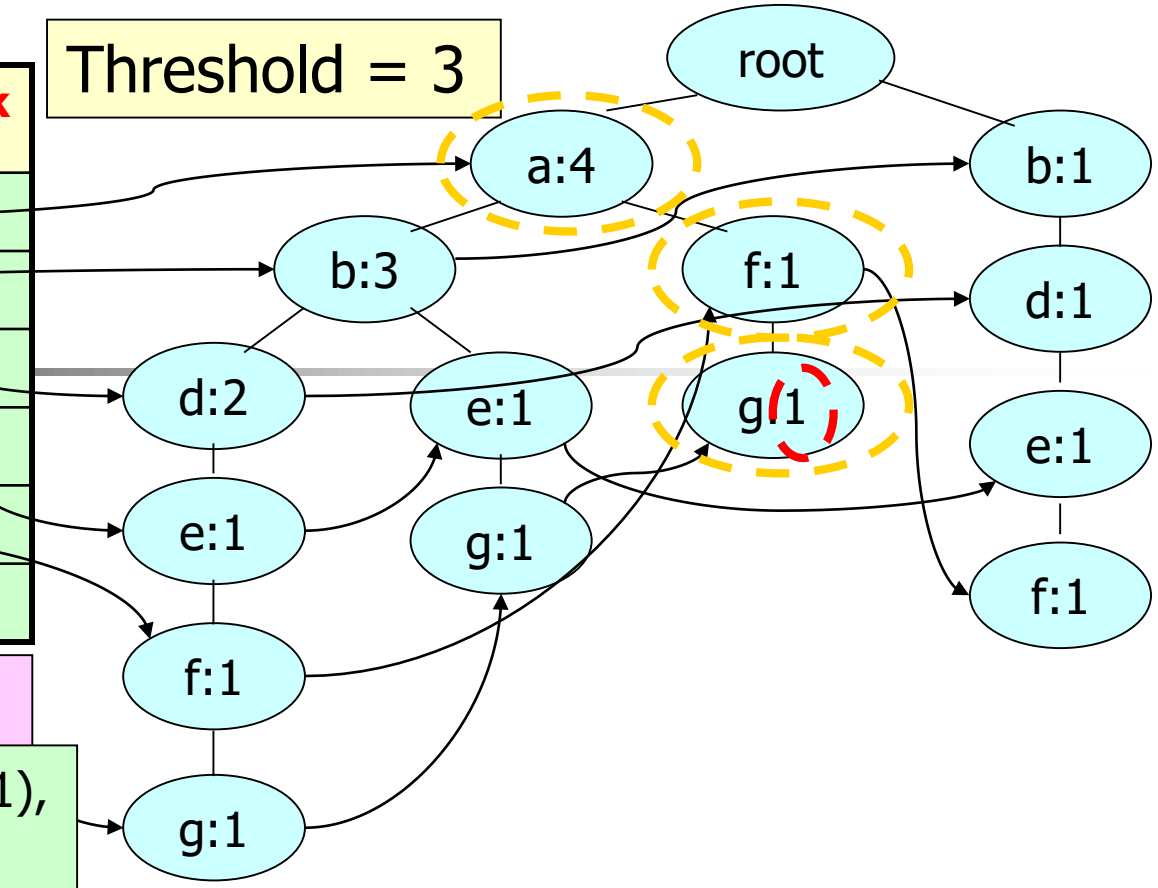


Cond. FP-tree on “g”

{ (a:1, b:1, d:1, e:1, f:1, g:1),
(a:1, b:1, e:1, g:1),
}

Item	Head of node-link
a	
b	
d	
e	
f	
g	

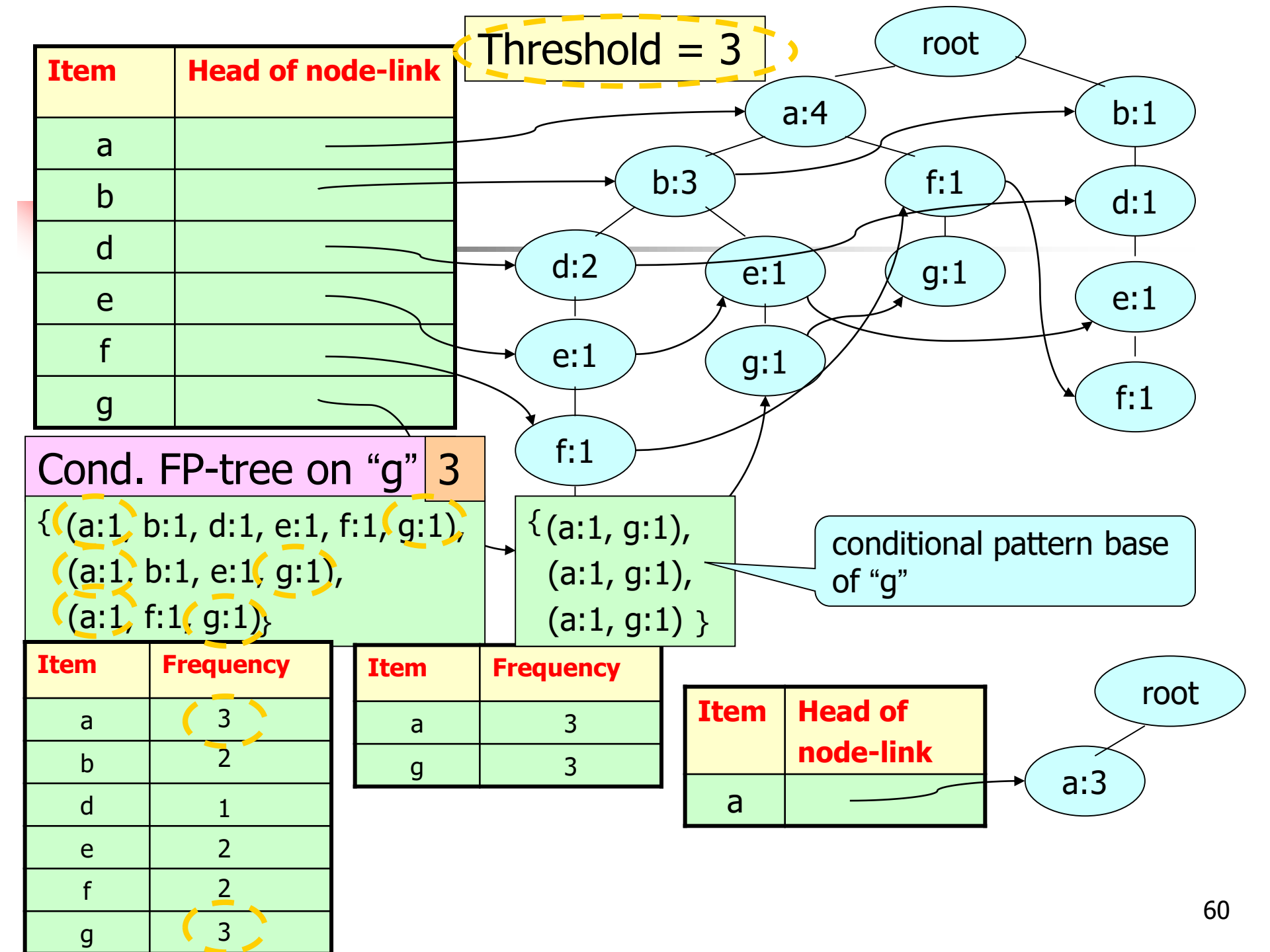
Threshold = 3

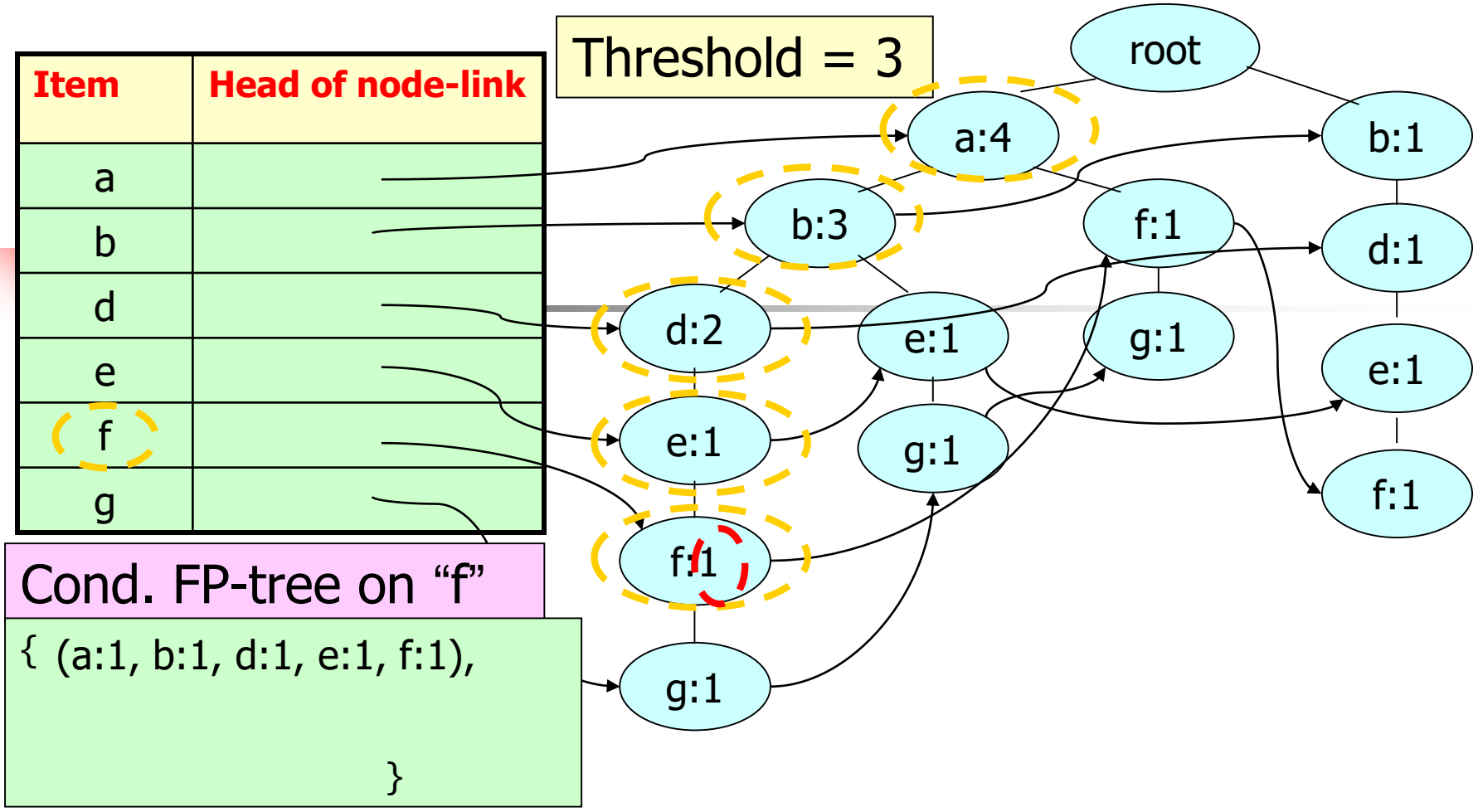


Cond. FP-tree on "g"

{ (a:1, b:1, d:1, e:1, f:1, g:1),
(a:1, b:1, e:1, g:1),
(a:1, f:1, g:1)}

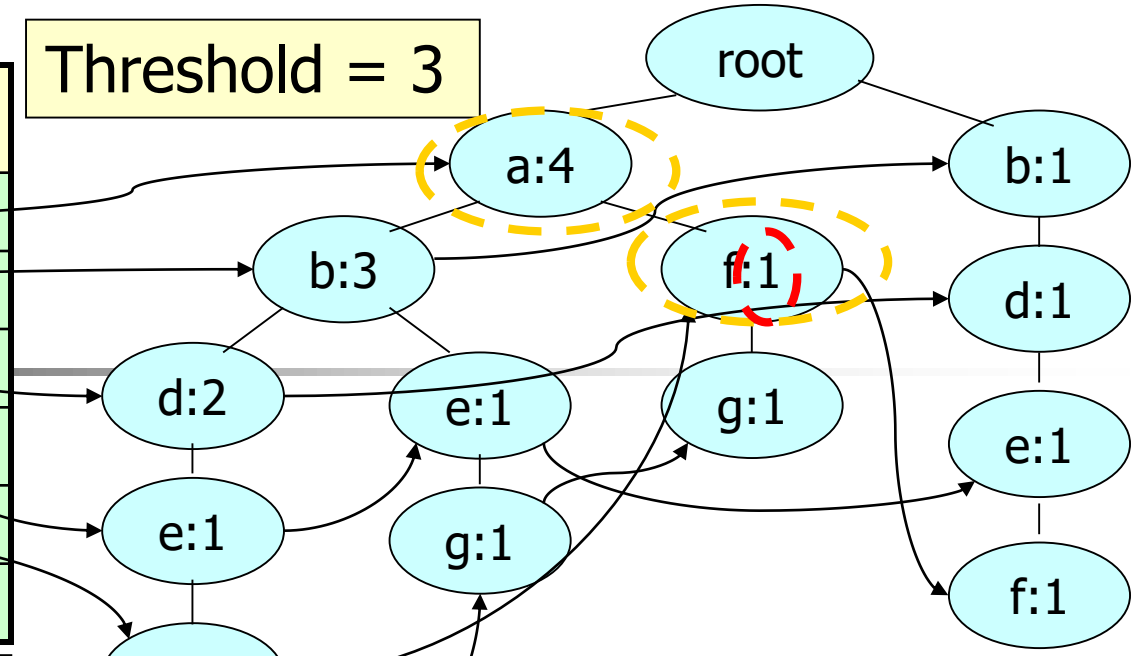
Item	Frequency
a	3
b	2
d	1
e	2
f	2
g	3


Item
Head of node-link
a
b
d
e
f
g
Threshold = 3
root
a:4
b:1
b:3
f:1
d:1
d:2
e:1
g:1
e:1
e:1
g:1
f:1
f:1
Cond. FP-tree on “g” 3
{(a:1, b:1, d:1, e:1, f:1, g:1),
(a:1, b:1, e:1, g:1),
(a:1, f:1, g:1)}
{(a:1, g:1),
(a:1, g:1),
(a:1, g:1) }
conditional pattern base of “g”
Item
Frequency
a
3
b
2
d
1
e
2
f
2
g
3
Item
Frequency
a
3
g
3
Item
Head of node-link
a
root
a:3


Item
Head of node-link
a
b
d
e
f
g
Threshold = 3
root
a:4
b:1
b:3
f:1
d:1
d:2
e:1
g:1
e:1
e:1
g:1
f:1
f:1
g:1
Cond. FP-tree on “f”
{ (a:1, b:1, d:1, e:1, f:1),
}

Item	Head of node-link
a	
b	
d	
e	
f	
g	

Threshold = 3

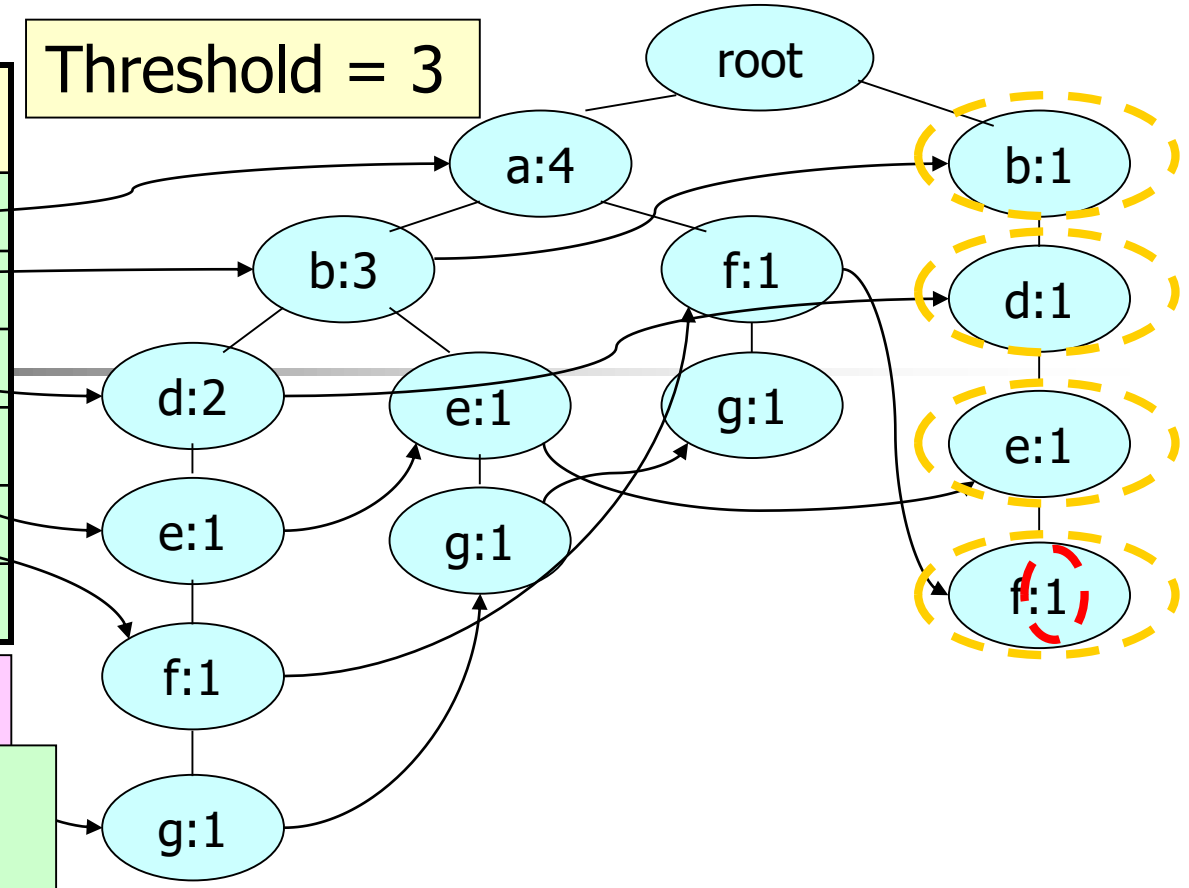


Cond. FP-tree on "f"

{ (a:1, b:1, d:1, e:1, f:1),
(a:1, f:1),
}

Item	Head of node-link
a	
b	
d	
e	
f	
g	

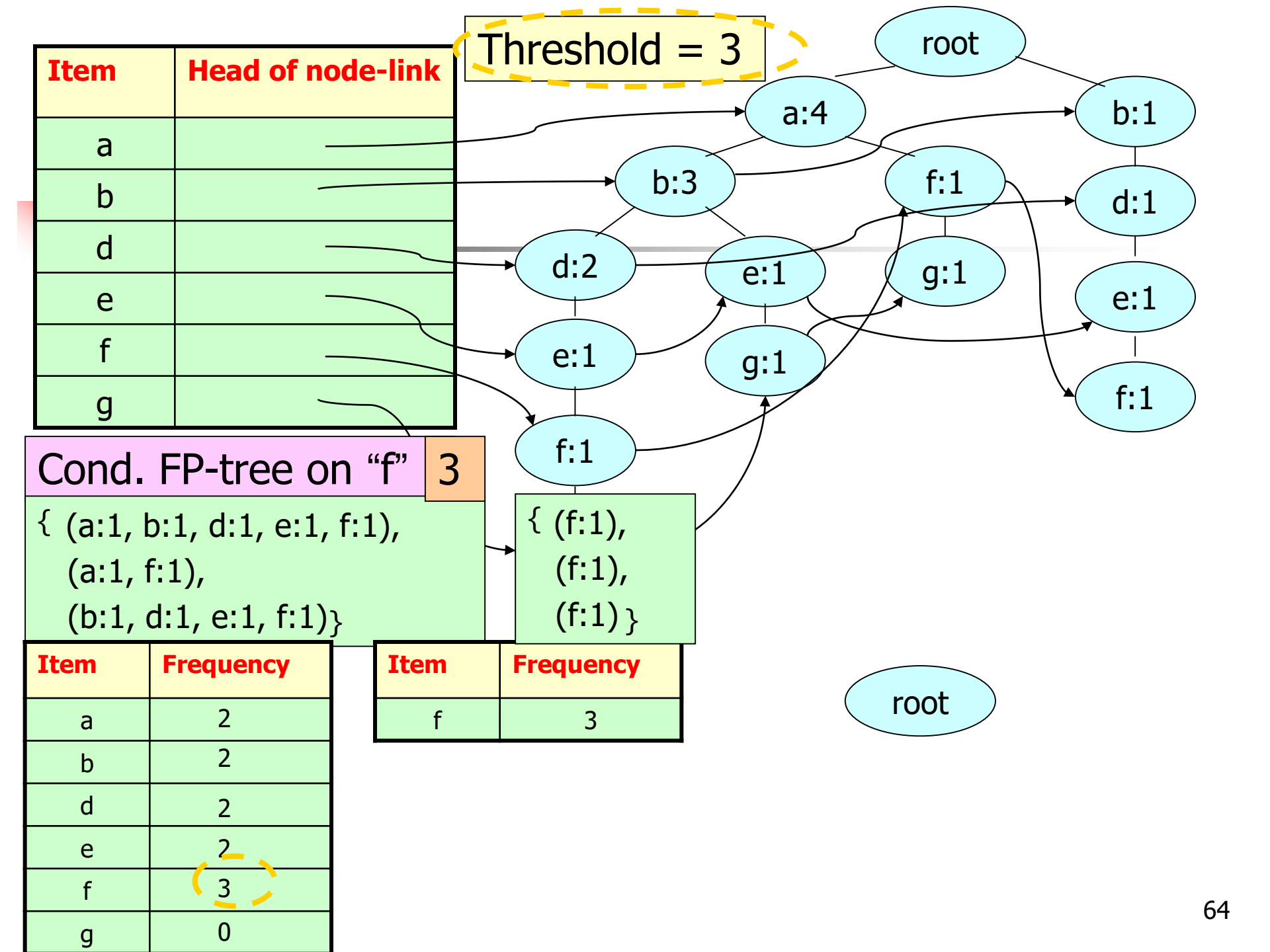
Threshold = 3



Cond. FP-tree on "f"

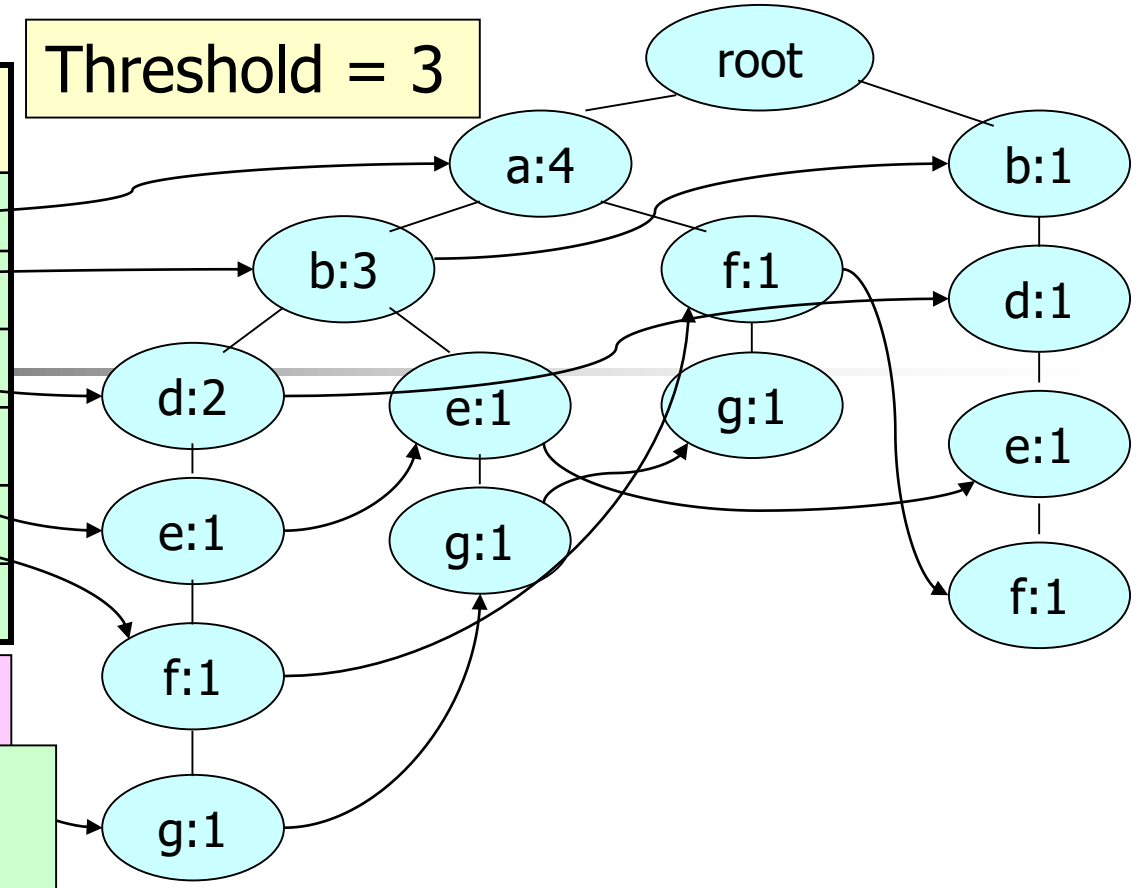
{ (a:1, b:1, d:1, e:1, f:1),
(a:1, f:1),
(b:1, d:1, e:1, f:1)}

Item	Frequency
a	2
b	2
d	2
e	2
f	3
g	0


Threshold = 3
Item
Head of node-link
a
b
d
e
f
g
root
a:4
b:1
b:3
f:1
d:1
d:2
e:1
g:1
e:1
e:1
g:1
f:1
f:1
Cond. FP-tree on “f”
3
{ (a:1, b:1, d:1, e:1, f:1),
(a:1, f:1),
(b:1, d:1, e:1, f:1)}
{ (f:1),
(f:1),
(f:1) }
Item
Frequency
a
2
b
2
d
2
e
2
f
3
g
0
Item
Frequency
f
3
root

Threshold = 3

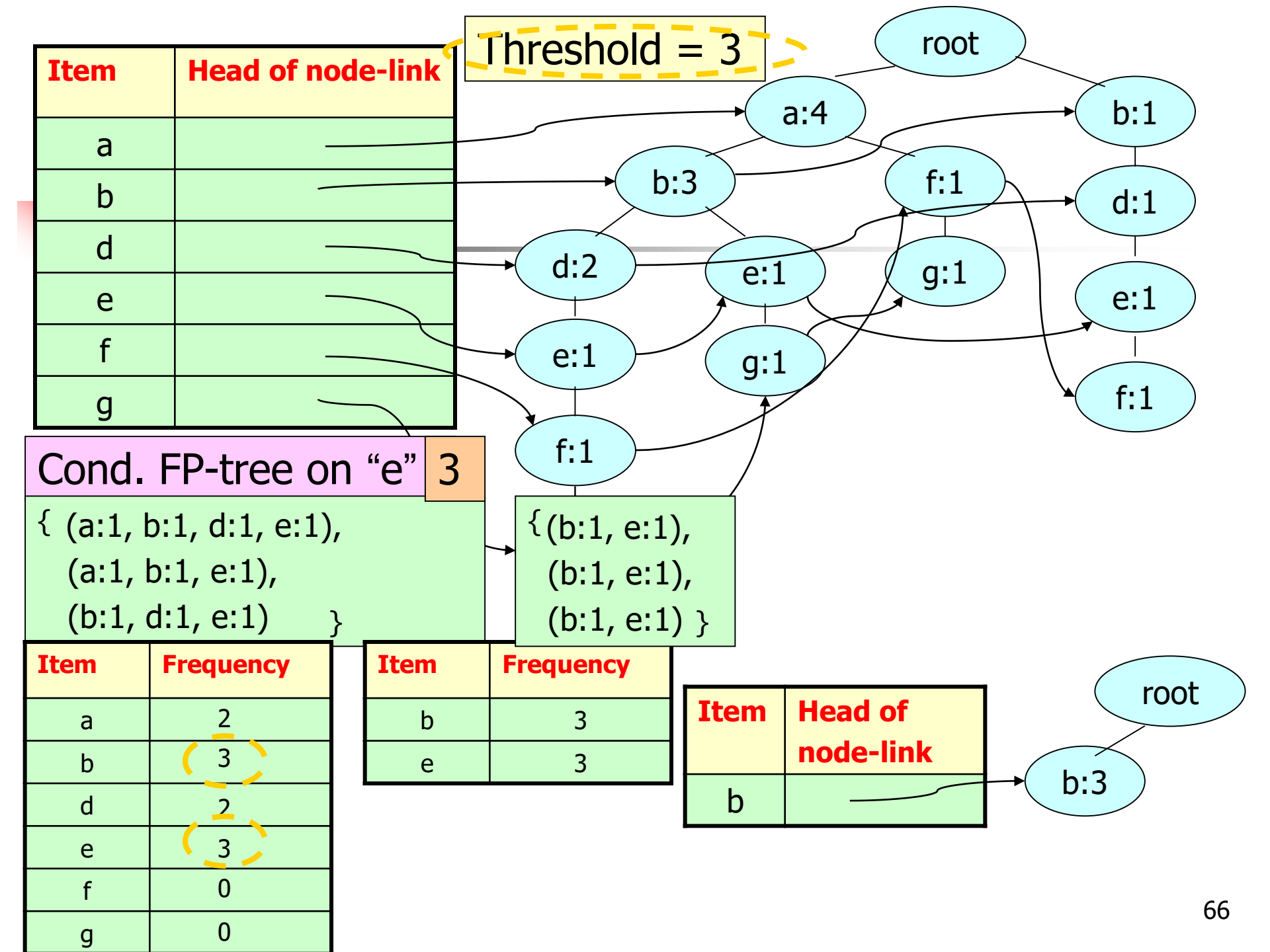
Item	Head of node-link
a	
b	
d	
e	
f	
g	



Cond. FP-tree on "e"

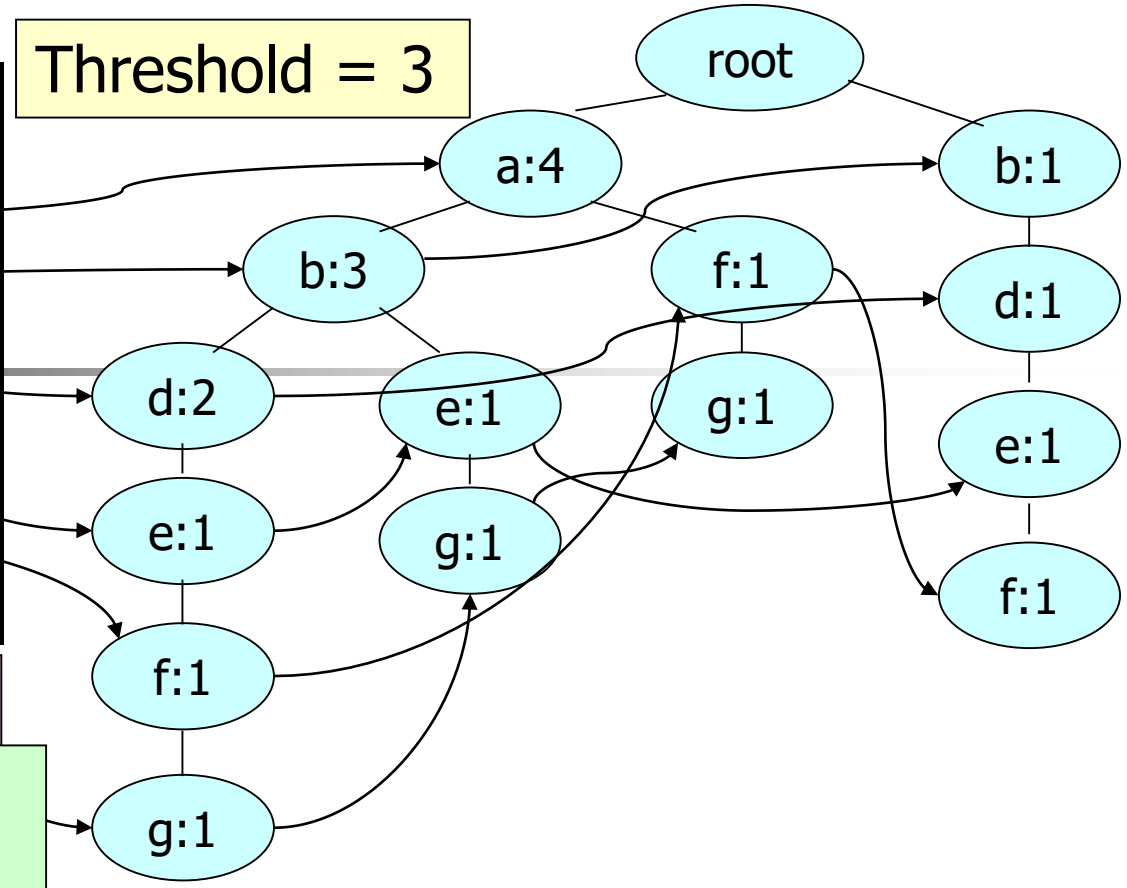
{ (a:1, b:1, d:1, e:1),
(a:1, b:1, e:1),
(b:1, d:1, e:1) }

Item	Frequency
a	2
b	3
d	2
e	3
f	0
g	0


Threshold = 3
Item
Head of node-link
a
b
d
e
f
g
root
a:4
b:1
b:3
f:1
d:1
d:2
e:1
g:1
e:1
e:1
g:1
f:1
f:1
Cond. FP-tree on "e"
3
{ (a:1, b:1, d:1, e:1),
(a:1, b:1, e:1),
(b:1, d:1, e:1) }
{(b:1, e:1),
(b:1, e:1),
(b:1, e:1) }
Item
Frequency
a
2
b
3
d
2
e
3
f
0
g
0
Item
Frequency
b
3
e
3
Item
Head of node-link
b
root
b:3

Item	Head of node-link
a	
b	
d	
e	
f	
g	

Threshold = 3



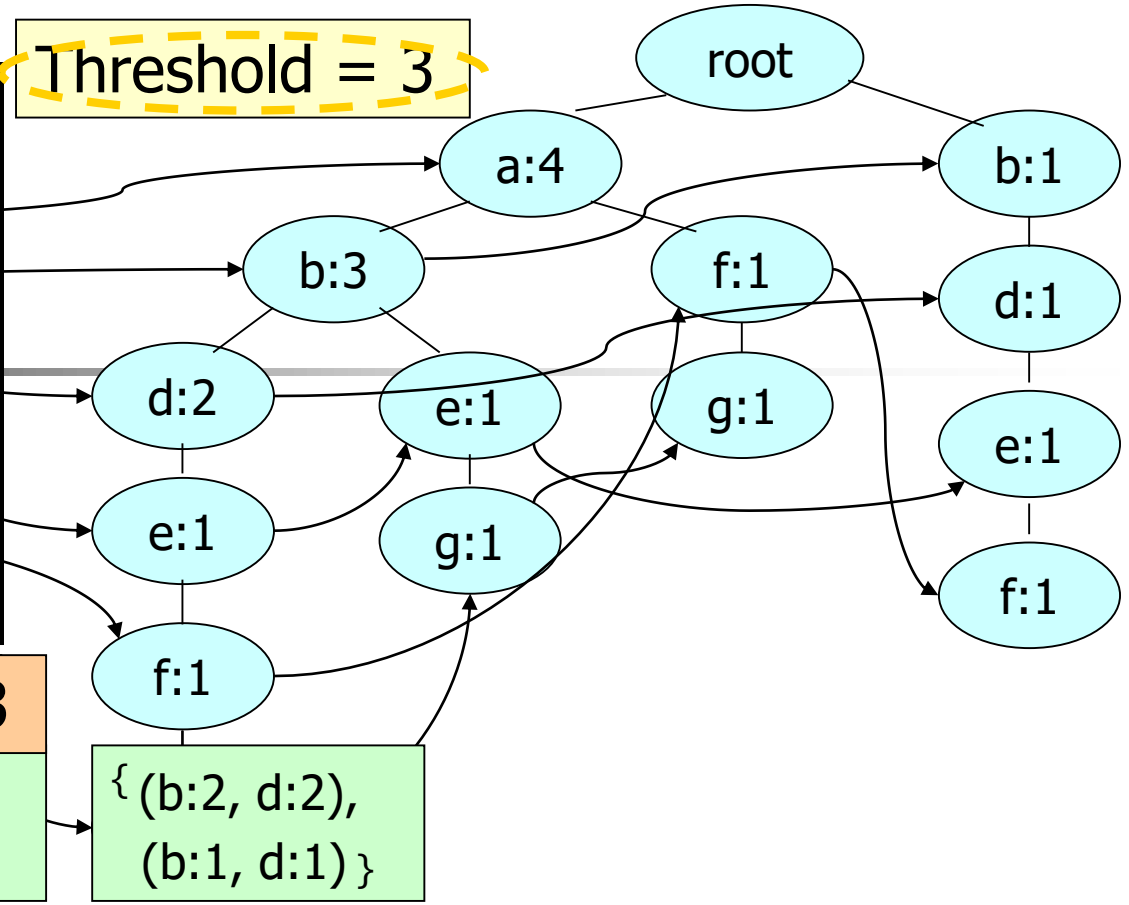
Cond. FP-tree on “d”

{ (a:2, b:2, d:2),
(b:1, d:1) }

Item	Frequency
a	2
b	3
d	3
e	0
f	0
g	0

Item	Head of node-link
a	
b	
d	
e	
f	
g	

Threshold = 3



Cond. FP-tree on "d" 3

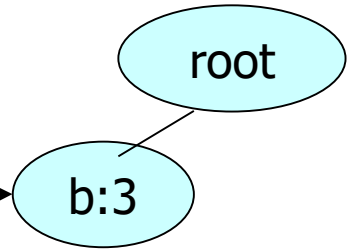
{ (a:2, b:2, d:2), (b:1, d:1) }

{ (b:2, d:2), (b:1, d:1) }

Item	Frequency
a	2
b	3
d	3
e	0
f	0
g	0

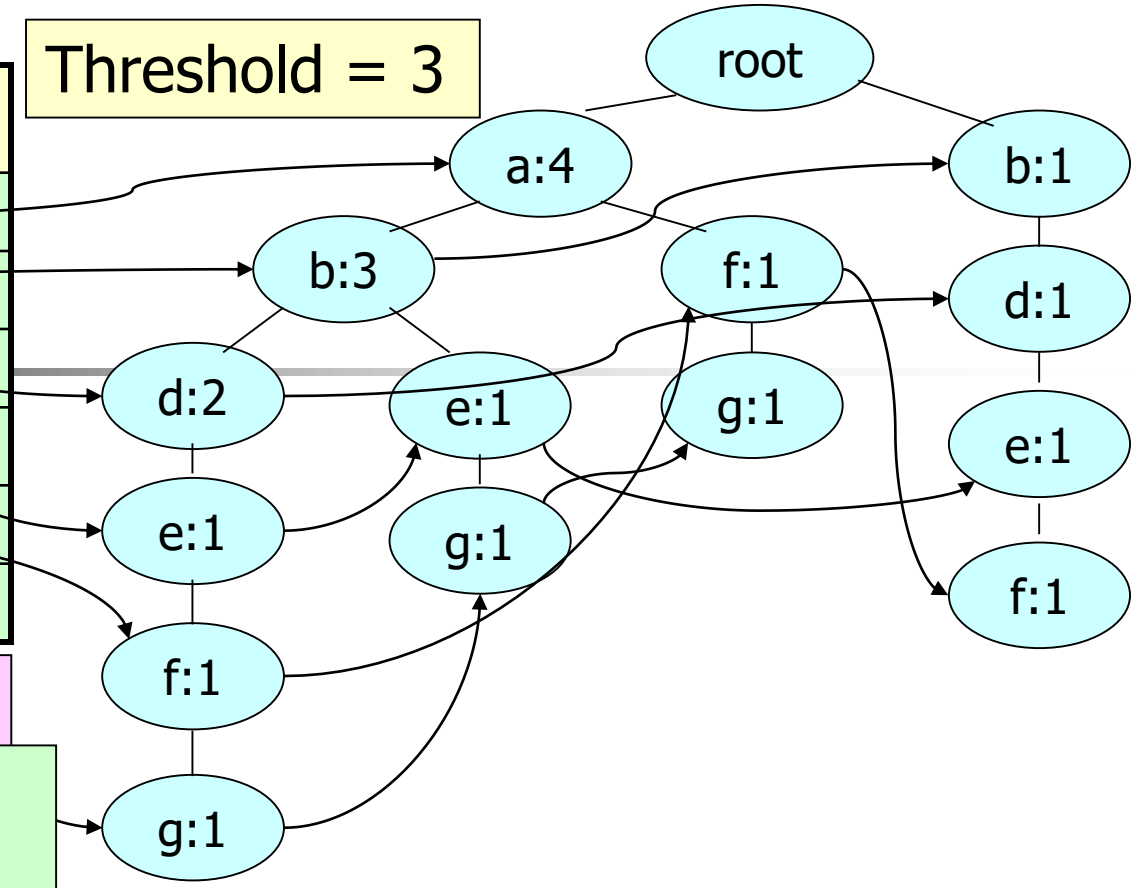
Item	Frequency
b	3
d	3

Item	Head of node-link
b	



Item	Head of node-link
a	
b	
d	
e	
f	
g	

Threshold = 3



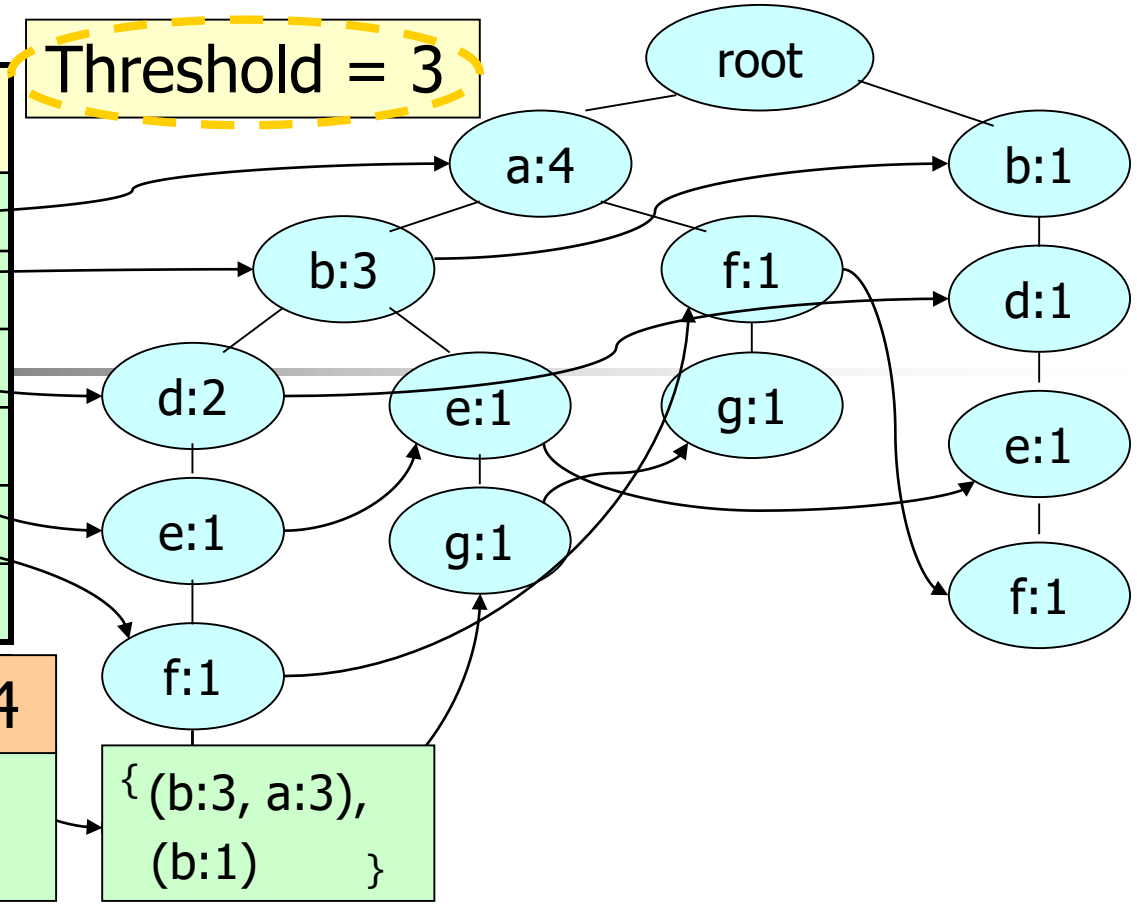
Cond. FP-tree on "b"

{ (a:3, b:3),
(b:1) }

Item	Frequency
a	3
b	4
d	0
e	0
f	0
g	0

Item	Head of node-link
a	
b	
d	
e	
f	
g	

Threshold = 3



Cond. FP-tree on "b" 4

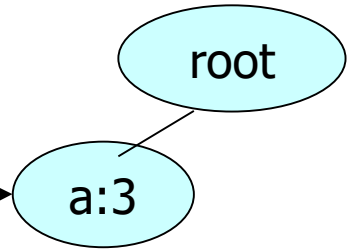
{ (a:3, b:3), (b:1) }

{ (b:3, a:3), (b:1) }

Item	Frequency
a	3
b	4
d	0
e	0
f	0
g	0

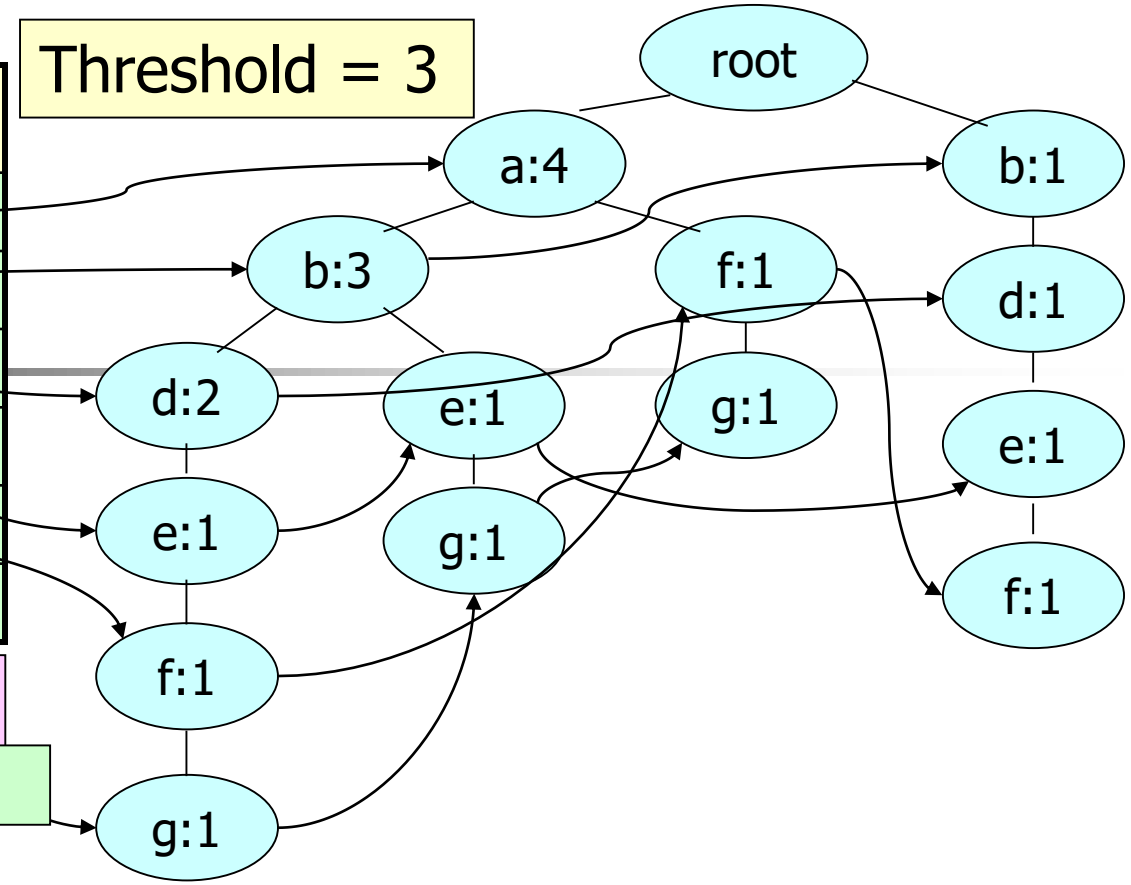
Item	Frequency
b	4
a	3

Item	Head of node-link
a	



Item	Head of node-link
a	
b	
d	
e	
f	
g	

Threshold = 3



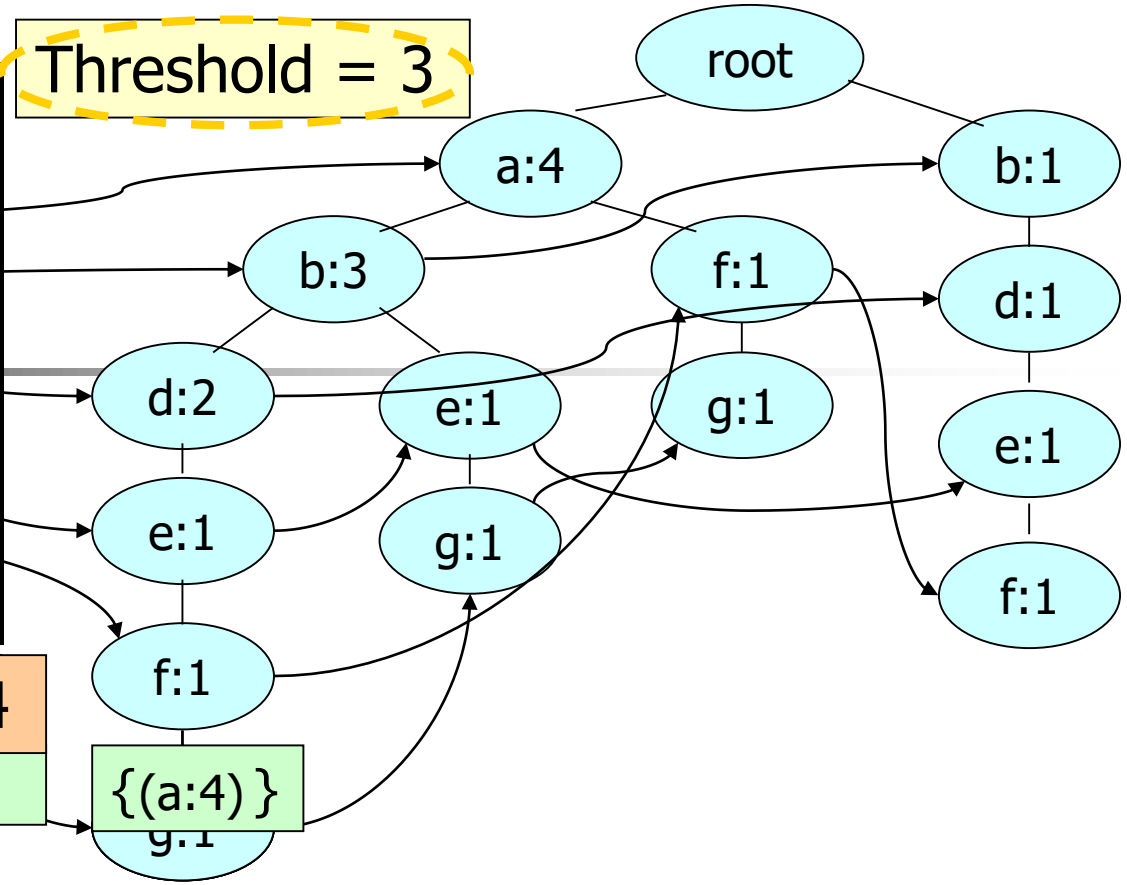
Cond. FP-tree on “a”

{ (a:4) }

Item	Frequency
a	4
b	0
d	0
e	0
f	0
g	0

Item	Head of node-link
a	
b	
d	
e	
f	
g	

Threshold = 3



Cond. FP-tree on "a" 4

{ (a:4) }

Item	Frequency
a	4
b	0
d	0
e	0
f	0
g	0

Item	Frequency
a	4

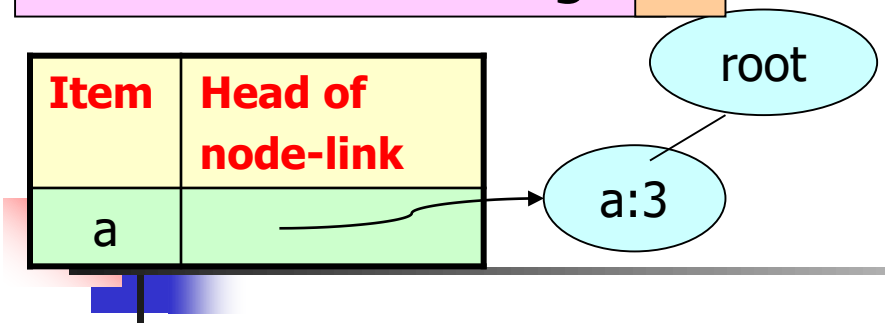
root

FP-tree

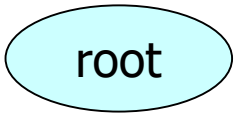
Step 1: Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
Step 2: Construct the FP-tree from the above data
Step 3: From the FP-tree above, construct the FP-conditional tree for each item (or itemset).
Step 4: Determine the frequent patterns.

Cond. FP-tree on “g” 3

Cond. FP-tree on “g” | 3

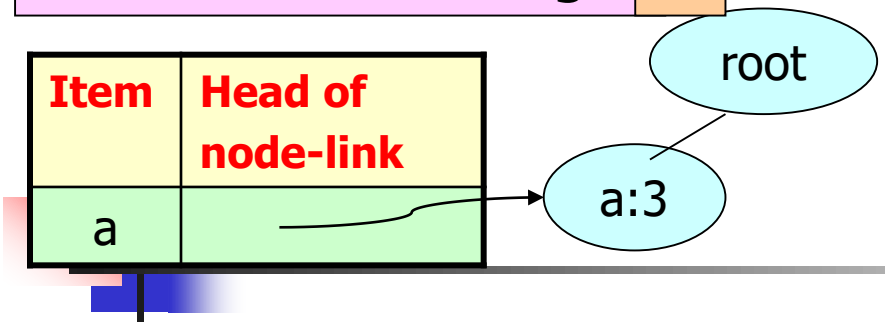


Cond. FP-tree on “f” | 3



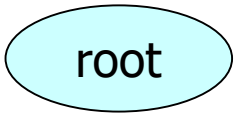
Cond. FP-tree on “e” | 3

Cond. FP-tree on “g” 3

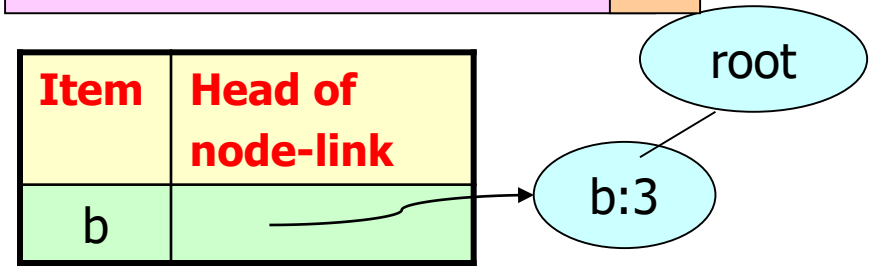


Cond. FP-tree on “d” 3

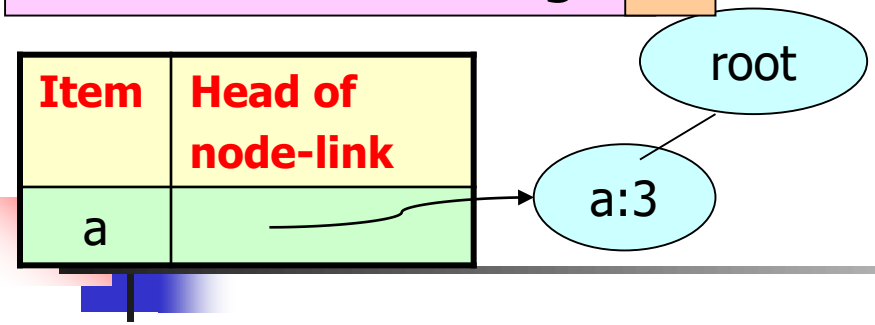
Cond. FP-tree on “f” 3



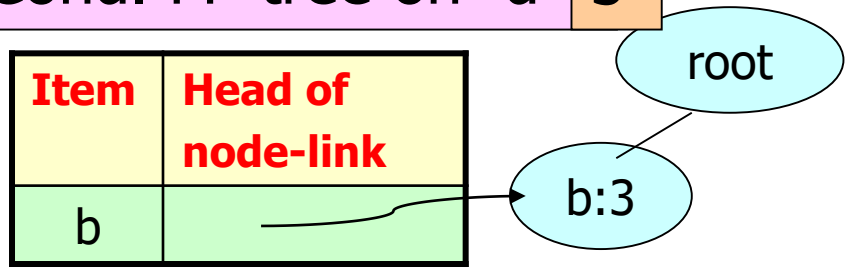
Cond. FP-tree on “e” 3



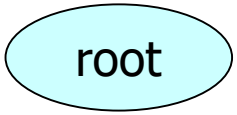
Cond. FP-tree on "g" 3



Cond. FP-tree on "d" 3

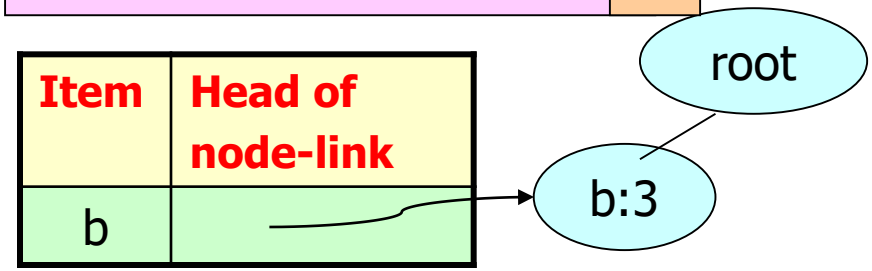


Cond. FP-tree on "f" 3

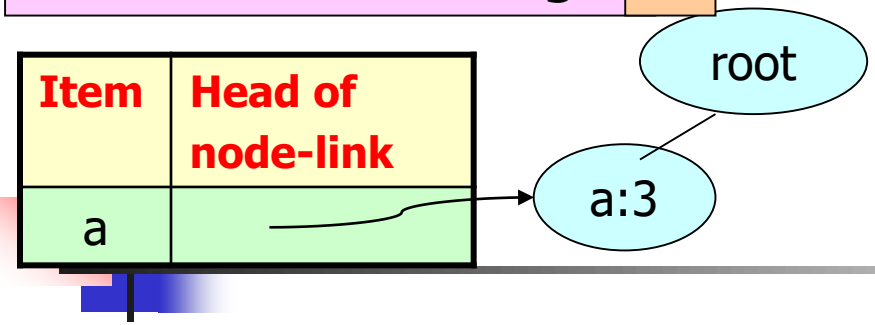


Cond. FP-tree on "b" 4

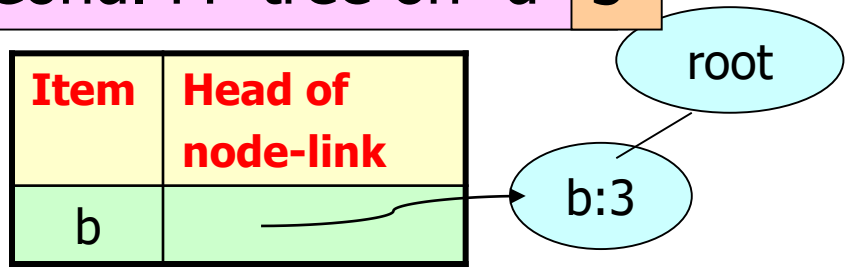
Cond. FP-tree on "e" 3



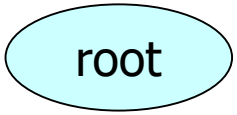
Cond. FP-tree on "g" 3



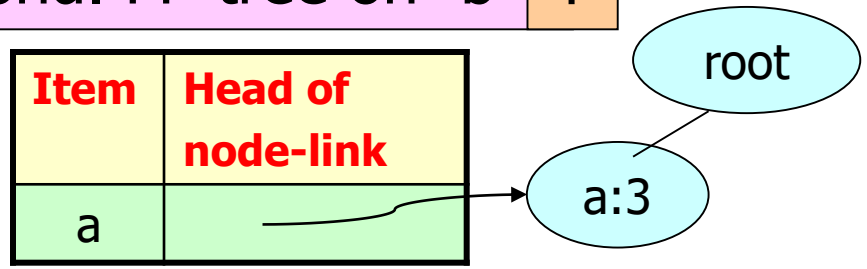
Cond. FP-tree on "d" 3



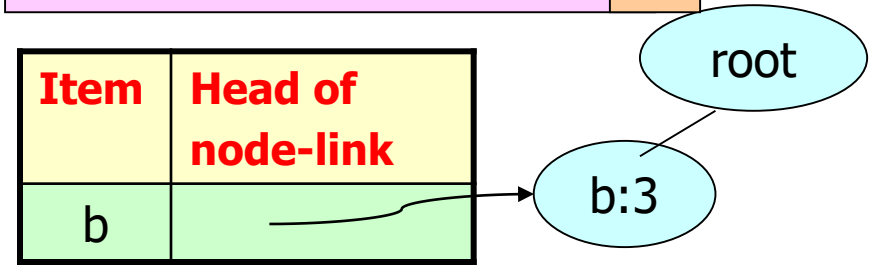
Cond. FP-tree on "f" 3



Cond. FP-tree on "b" 4



Cond. FP-tree on "e" 3

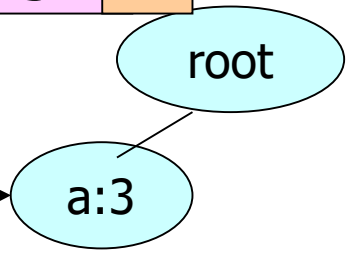


Cond. FP-tree on "a" 4

Cond. FP-tree on "g" 3

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

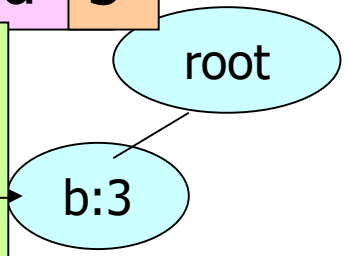
2. After generating this cond. tree, we generate {a, g} (support = 3)



Cond. FP-tree on "d" 3

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

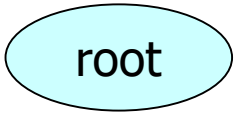
2. After generating this cond. tree, we generate {b, d} (support = 3)



Cond. FP-tree on "f" 3

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

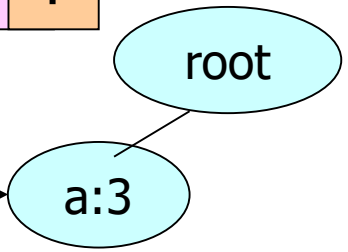
2. After generating this cond. tree, we do not generate any itemset.



Cond. FP-tree on "b" 4

1. Before generating this cond. tree, we generate {b} (support = 4)

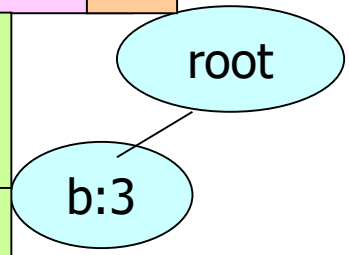
2. After generating this cond. tree, we generate {a, b} (support = 3)



Cond. FP-tree on "e" 3

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

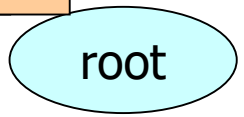
2. After generating this cond. tree, we generate {b, e} (support = 3)

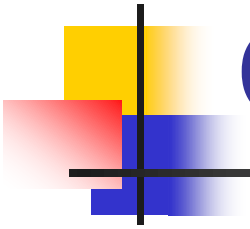


Cond. FP-tree on "a" 4

1. Before generating this cond. tree, we generate {a} (support = 4)

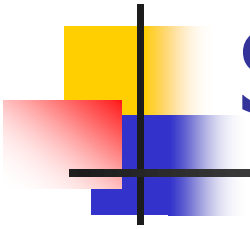
2. After generating this cond. tree, we do not generate any itemset.





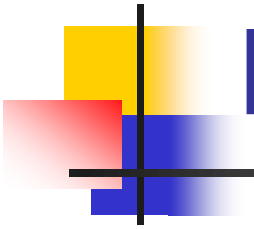
Complexity

- Complexity in building FP-tree
  - Two scans of the transactions DB
    - Collect frequent items
    - Construct the FP-tree
- Cost to insert one transaction
  - Number of frequent items in this transaction



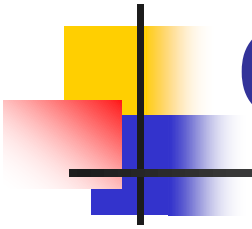
Size of the FP-tree

- The size of the FP-tree is bounded by the overall occurrences of the frequent items in the database



Height of the Tree

- The height of the tree is bounded by the maximum number of frequent items in any transaction in the database

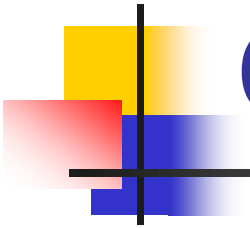


Compression

- With respect to the total number of items stored,
  - is FP-tree more compressed compared with the original databases?

Details of the Algorithm

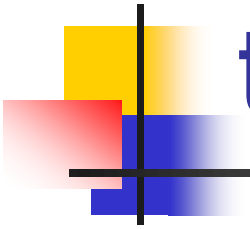
- Procedure FP-growth (Tree, α)
  - if Tree contains a single path P
    - for each combination (denoted by β) of the nodes in the path P do
      - generate pattern $\beta \cup \alpha$ with support = minimum support of nodes in β
  - else
    - for each a_i in the header table of Tree do
      - generate pattern $\beta = a_i \cup \alpha$ with support = a_i.support
      - construct β's conditional pattern base and then β's conditional FP-tree $Tree_\beta$
      - if $Tree_\beta \neq \varnothing$
        - Call FP-growth($Tree_\beta$, β)



Outline

- Association Rule Mining
  - Problem Definition
- Algorithm
- How to use the data mining tool

How to use the data mining tool



- Please see the next set of slides.