Association Analysis (3)
FPTree/FPGrowth

FP-Tree/FP-Growth Algorithm

- Use a compressed representation of the database using an FP-tree
- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets.

Building the FP-Tree

1. Scan data to determine the support count of each item.

   Infrequent items are discarded, while the frequent items are sorted in decreasing order of support counts.

2. Make a second pass over the data to construct the FPtree.

   As the transactions are read, their items are sorted according to the above order.

First scan – determine frequent 1-itemsets, then build header

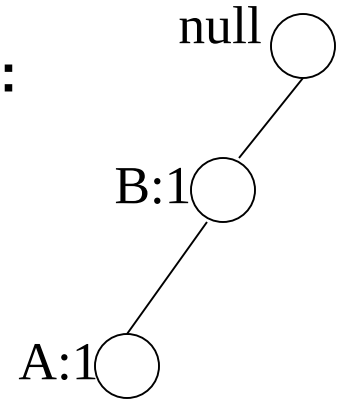
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

B	8
A	7
C	7
D	5
E	3

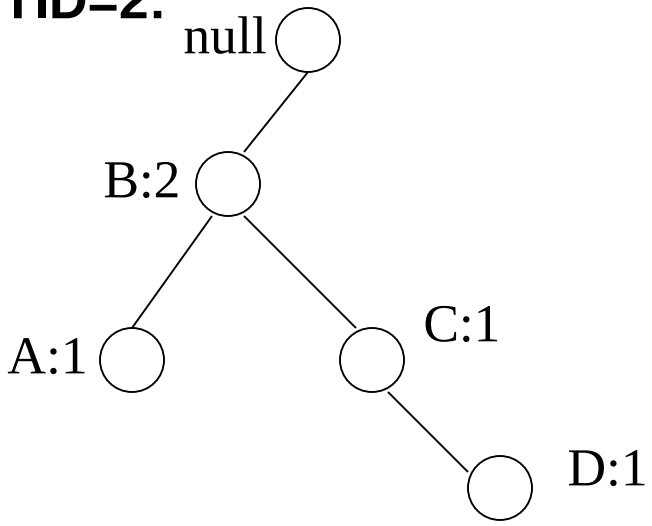
FP-tree construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

After reading TID=1:



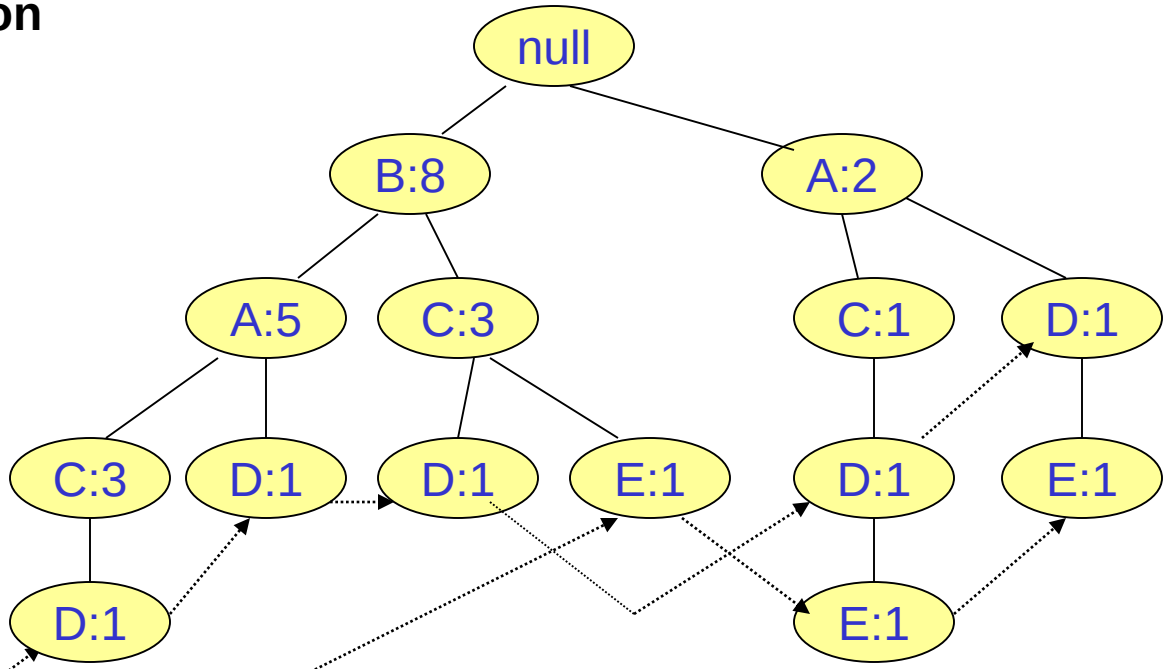
After reading TID=2:



FP-Tree Construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Transaction Database

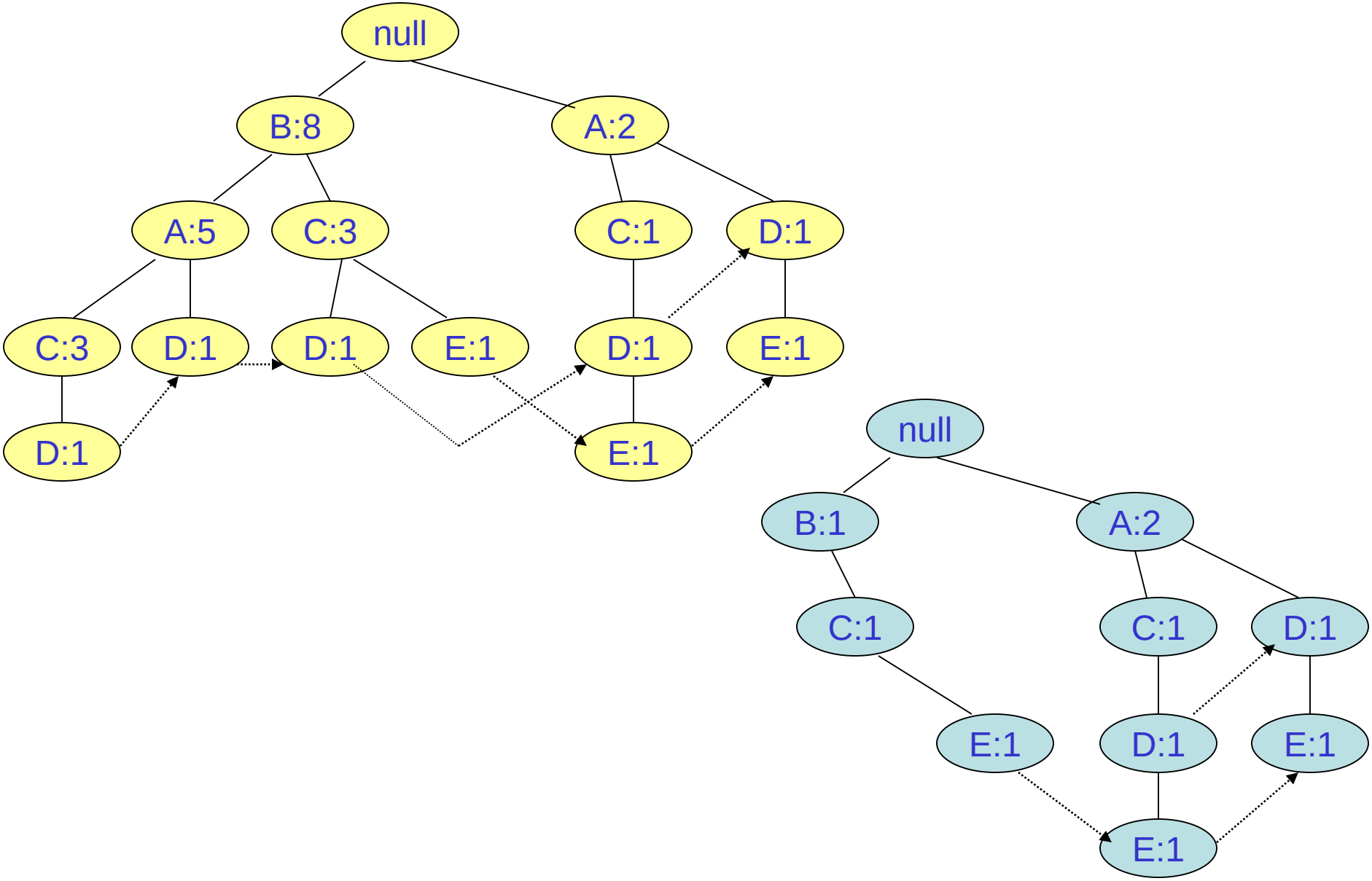


Header table

Item	Pointer
B	8
A	7
C	7
D	5
E	3

Chain pointers help in quickly finding all the paths of the tree containing some given item.

Paths (transactions) containing node E (last node in the header table)



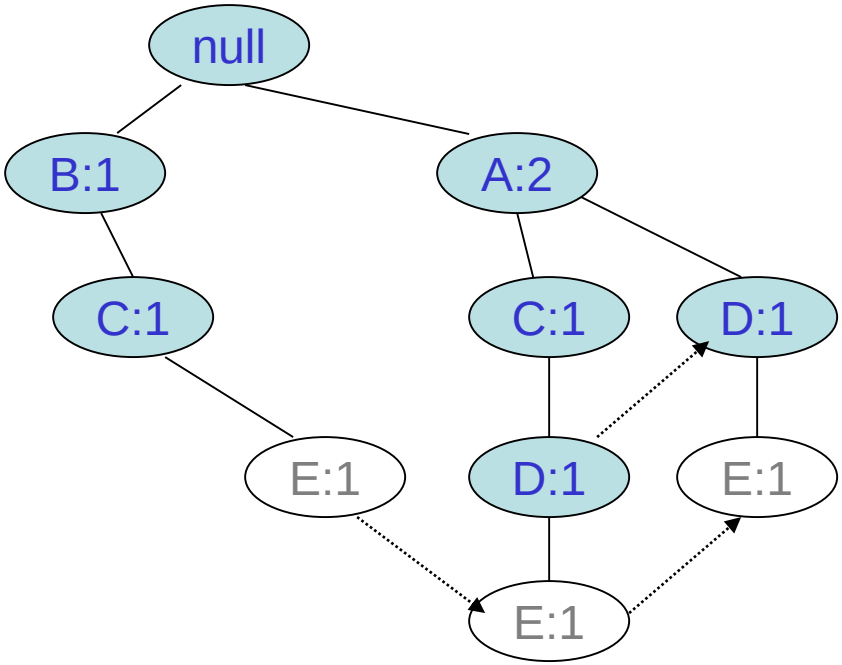
Conditional FP-Tree for E

- FP-Growth builds a conditional FP-Tree for E, which is the tree of itemsets ending in E.

- It is not the tree obtained in the previous slide as result of deleting nodes from the original tree. Why?

- Because the order of the items can change.
  - E.g. now, C has a higher count than B.

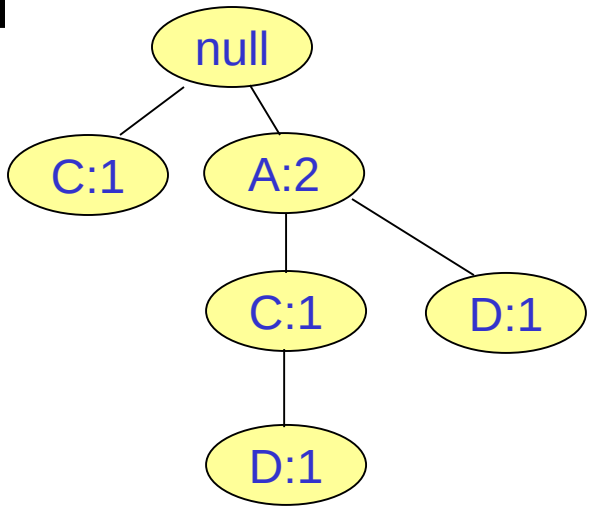
Suffix E



(New) Header table

A	2
C	2
D	2

Conditional FP-Tree for suffix E



The set of paths ending in E.

Insert each path (after truncating E) into a new tree.

B doesn't survive because it has support 1, which is lower than min support of 2.

We continue recursively.

Base of recursion: When the tree has a single path only.

We output FI: E

Steps of Building Conditional FP-Trees

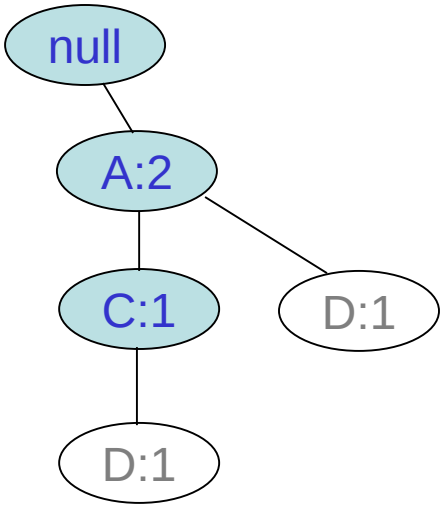
1. Find the paths containing on focus item.

2. **Read the tree** to determine the new counts of the items along those paths.

   Build a new header.

3. **Read again the tree**. Insert the paths into the conditional FP-Tree according to the new order.

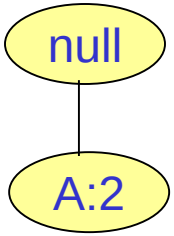
Suffix DE



(New) Header table

A	2

The conditional FP-Tree for suffix DE



The set of paths, from the E-conditional FP-Tree, ending in D.

Insert each path (after truncating D) into a new tree.

We have reached the base of recursion.

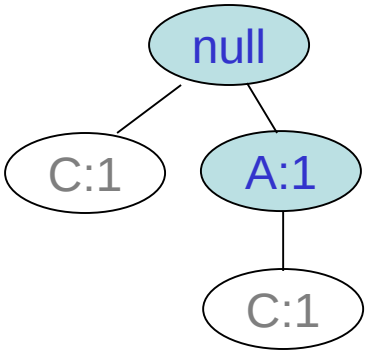
FI: DE, ADE

Base of Recursion

- We continue recursively on the conditional FP-Tree.

- Base case of recursion: when the tree is just a single path.
  - Then, we just produce all the subsets of the items on this path concatenated with the corresponding suffix.

Suffix CE



(New) Header table

The conditional FP-Tree for suffix CE

The set of paths, from the E-conditional FP-Tree, ending in C.

Insert each path (after truncating C) into a new tree.

A doesn't survive because it has support 1

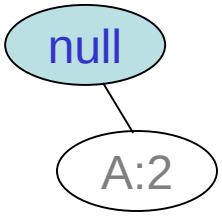
We have reached the base of recursion.

FI: CE

Suffix AE

(New) Header table

The conditional FP-Tree for suffix AE



The set of paths, from the E-conditional FP-Tree, ending in A.

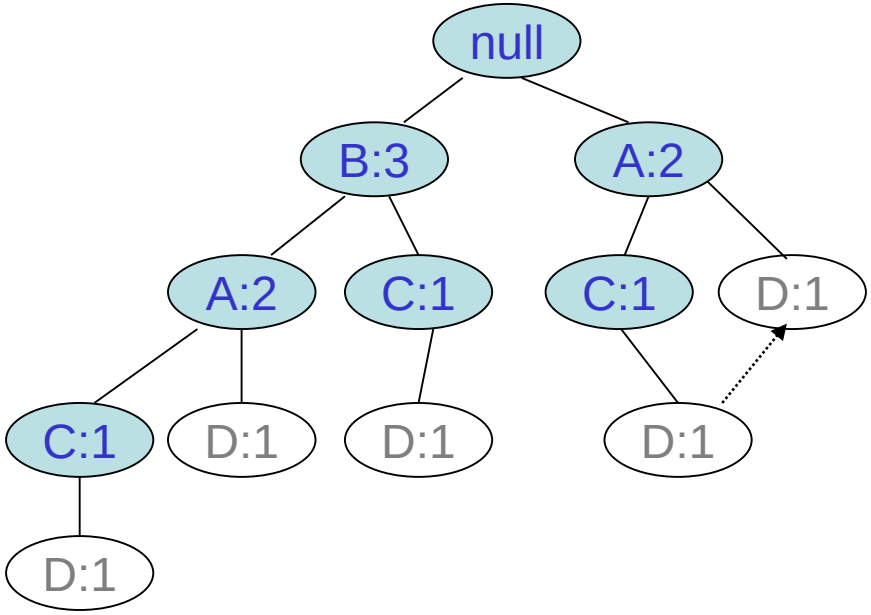
Insert each path (after truncating A) into a new tree.

We have reached the base of recursion.

FI: AE

Suffix D

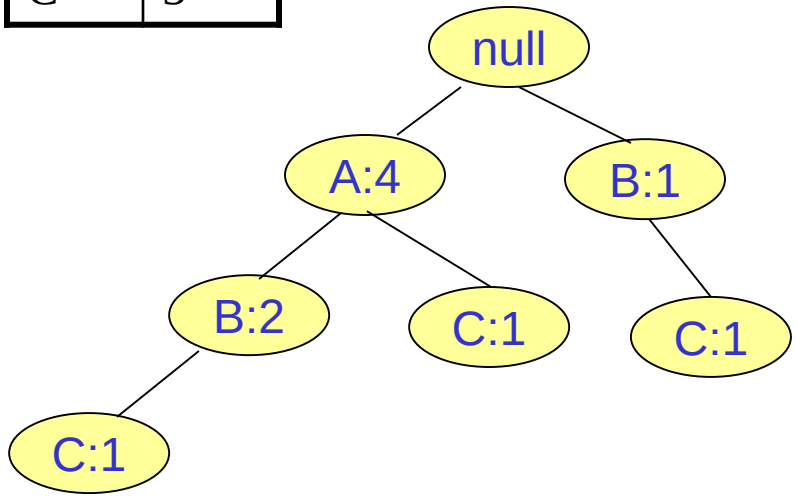
Observe how the shape of the tree changes.



(New) Header table

A	4
B	3
C	3

Conditional FP-Tree for suffix D



The set of paths ending in D.

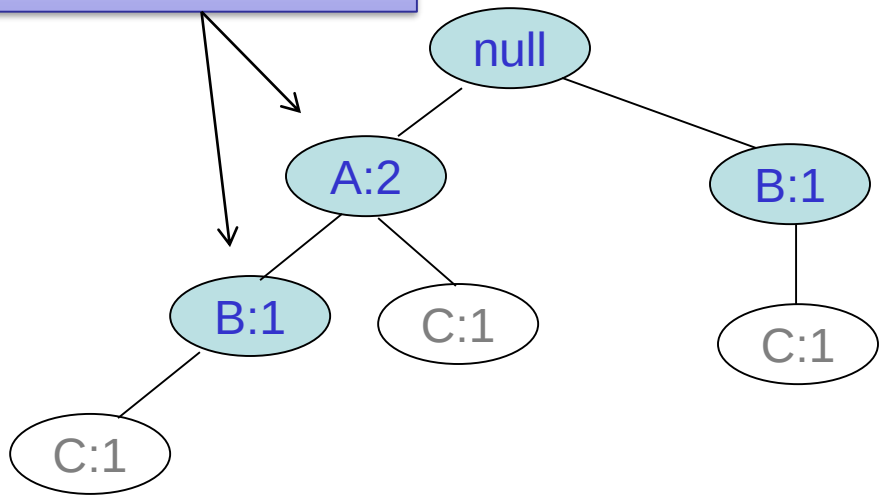
Insert each path (after truncating D) into a new tree.

We continue recursively.
Base of recursion: When the tree has a single path only.

FI: D

We update the counts, so that each count is equal to the sum of the leafs of the subtree.

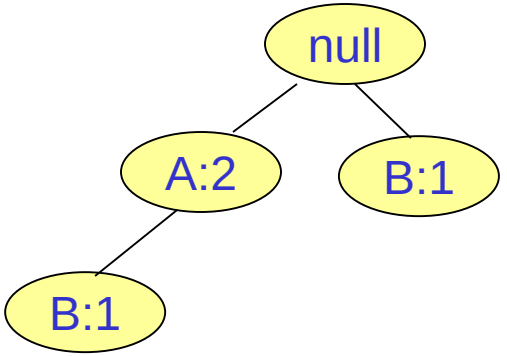
Suffix CD



(New) Header table

A	2
B	2

Conditional FP-Tree for suffix CD



The set of paths, from the D-conditional FP-Tree, ending in C.

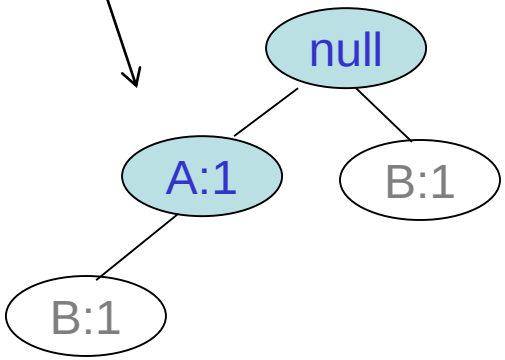
Insert each path (after truncating C) into a new tree.

We continue recursively.
Base of recursion: When the tree has a single path only.

FI: CD

Suffix BCD

We update the counts, so that each count is equal to the sum of the leafs of the subtree.



(New) Header table

Conditional FP-Tree for suffix CDB

The set of paths from the CD-conditional FP-Tree, ending in B.

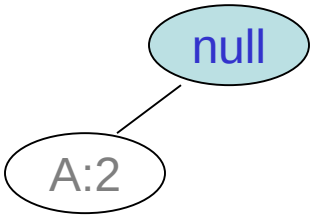
Insert each path (after truncating B) into a new tree.

A doesn't survive because it has support 1

We have reached the base of recursion.

FI: BCD

Suffix ACD



(New) Header table

Conditional FP-Tree for suffix ACD

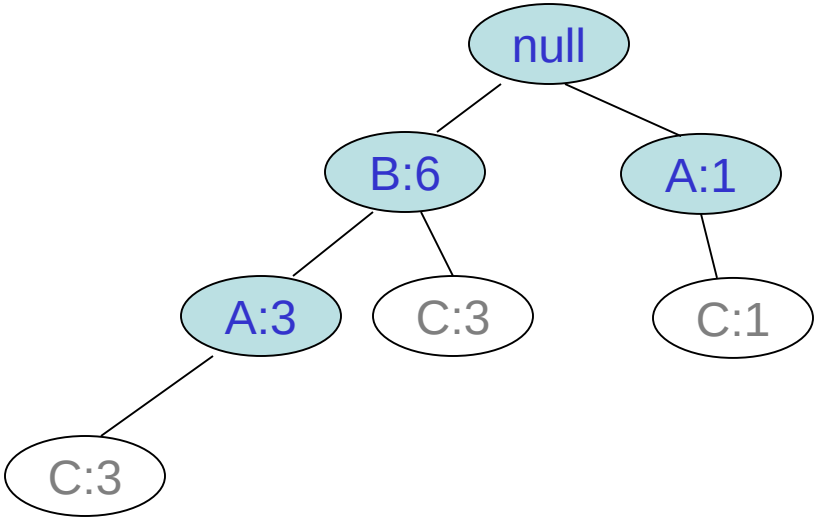
The set of paths from the CD-conditional FP-Tree, ending in A.

Insert each path (after truncating A) into a new tree.

We have reached the base of recursion.

FI: ACD

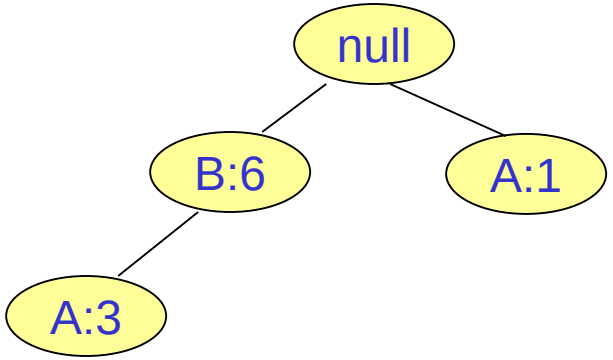
Suffix C



(New) Header table

B	6
A	4

Conditional FP-Tree for suffix C



The set of paths ending in C.

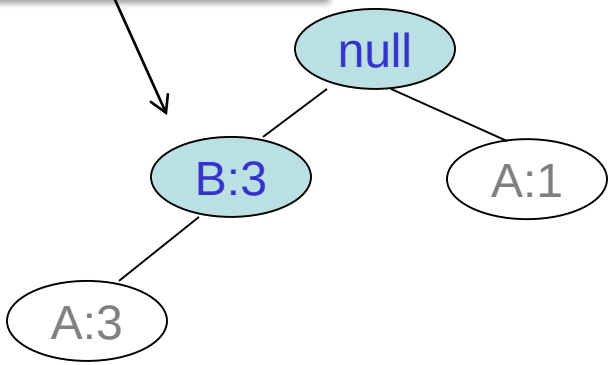
Insert each path (after truncating C) into a new tree.

We continue recursively.
Base of recursion: When the tree has a single path only.

FI: C

We update the counts, so that each count is equal to the sum of the leafs of the subtree.

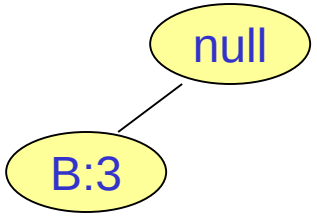
Suffix AC



(New) Header table

B	3

Conditional FP-Tree for suffix AC



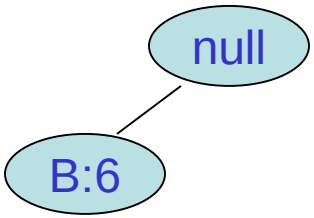
The set of paths from the C-conditional FP-Tree, ending in A.

Insert each path (after truncating A) into a new tree.

We have reached the base of recursion.

FI: AC, BAC

Suffix BC



(New) Header table

B	3

Conditional FP-Tree for suffix BC

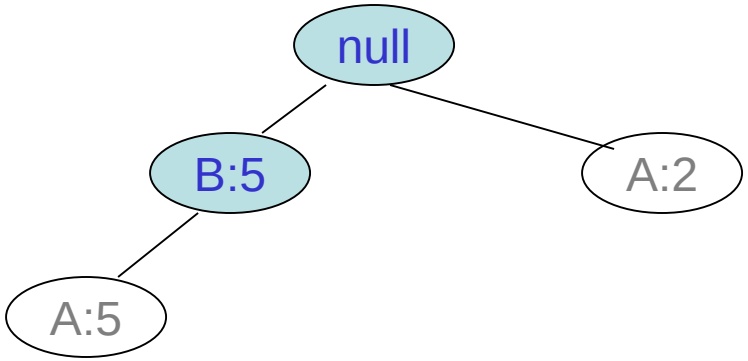
The set of paths from
the C-conditional FP-Tree, ending in B.

Insert each path (after truncating B)
into a new tree.

We have reached the base of recursion.

FI: BC

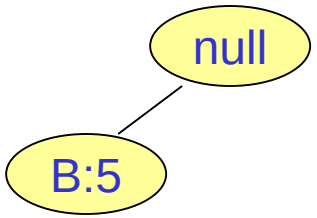
Suffix A



(New) Header table

B	5

Conditional FP-Tree for suffix A



The set of paths ending in A.

Insert each path (after truncating A) into a new tree.

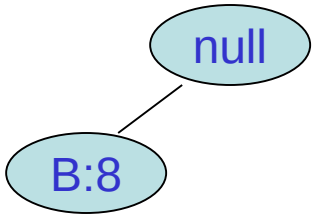
We have reached the base of recursion.

FI: A, BA

Suffix B

(New) Header table

Conditional FP-Tree for suffix B



The set of paths ending in B.

Insert each path (after truncating B) into a new tree.

We have reached the base of recursion.

FI: B