FPtree/FPGrowth

(Complete Example)

Lecture 15A

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

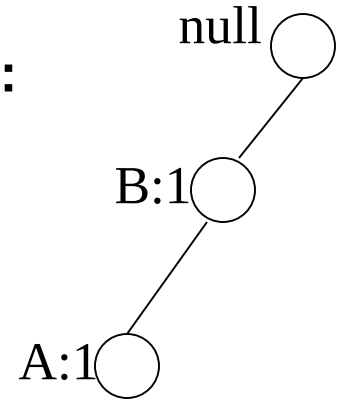
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

B	8
A	7
C	7
D	5
E	3

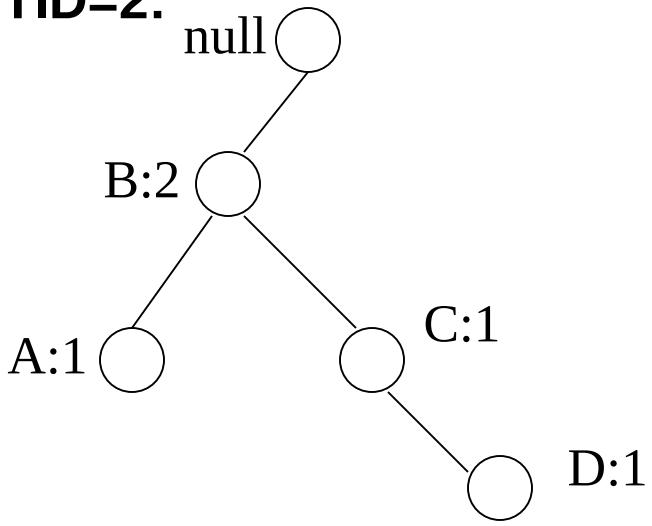
FP-tree construction

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:



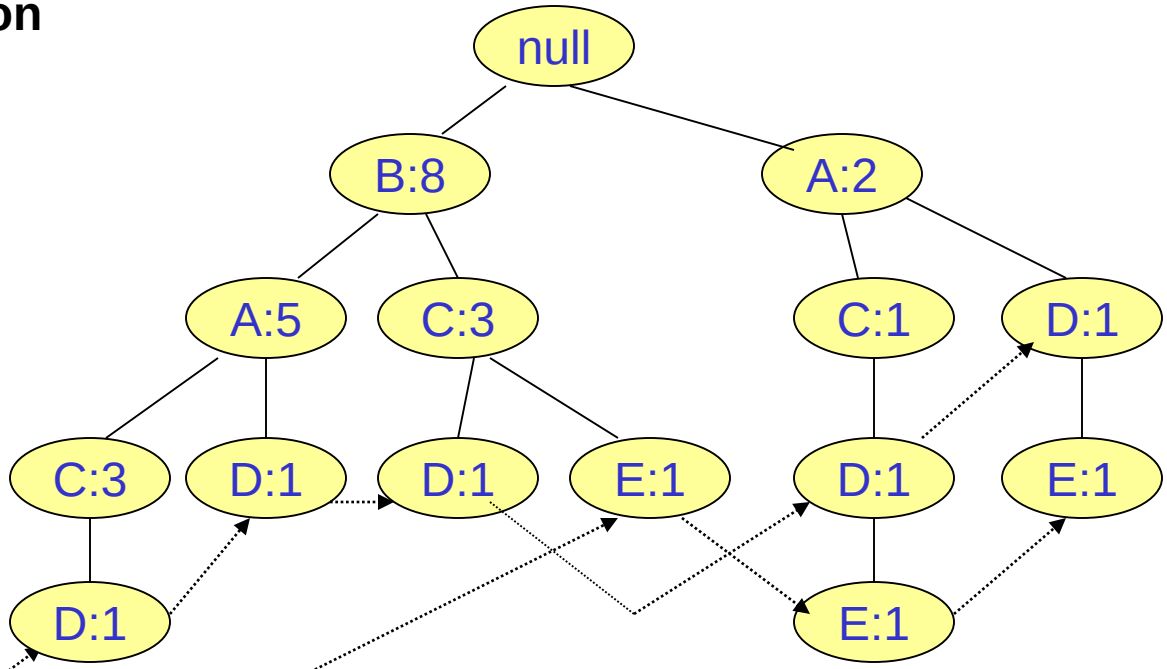
After reading TID=2:



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

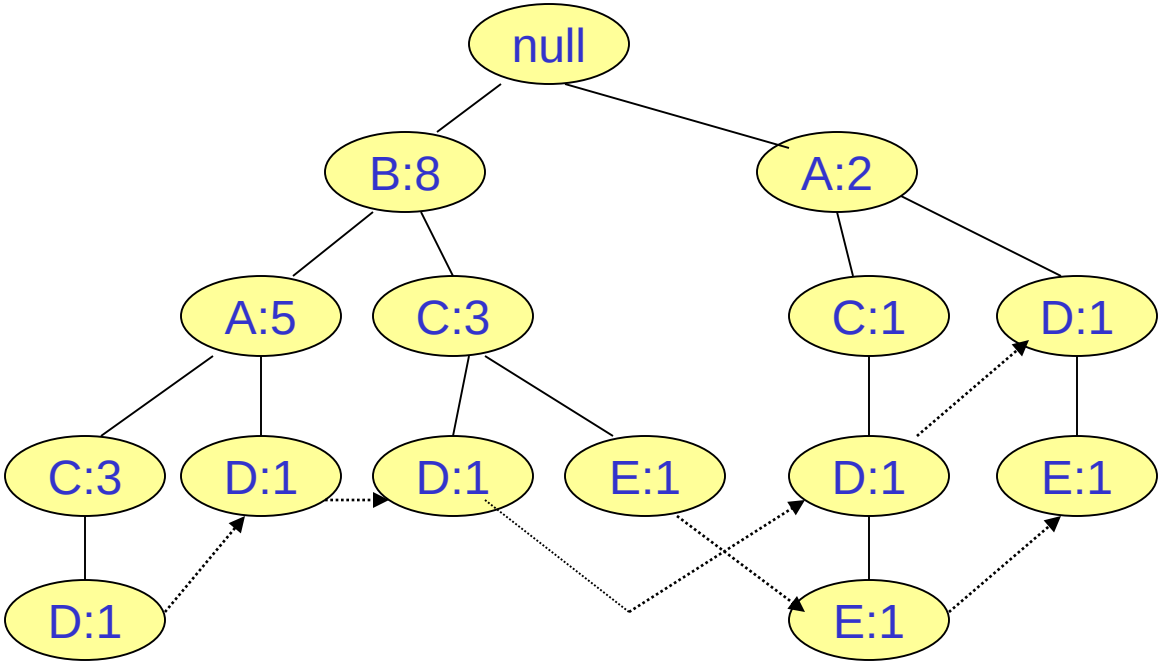


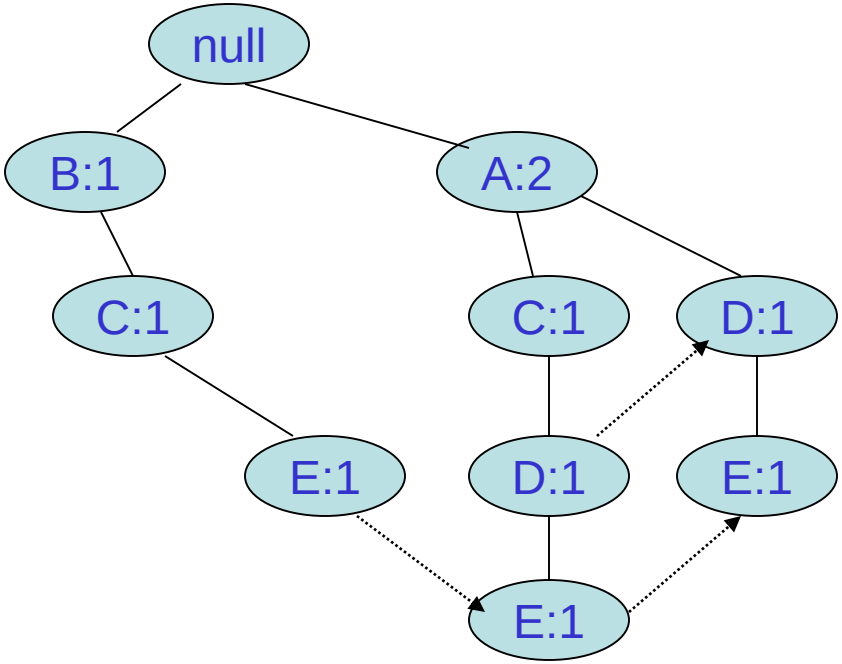
Header table

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 containing node E





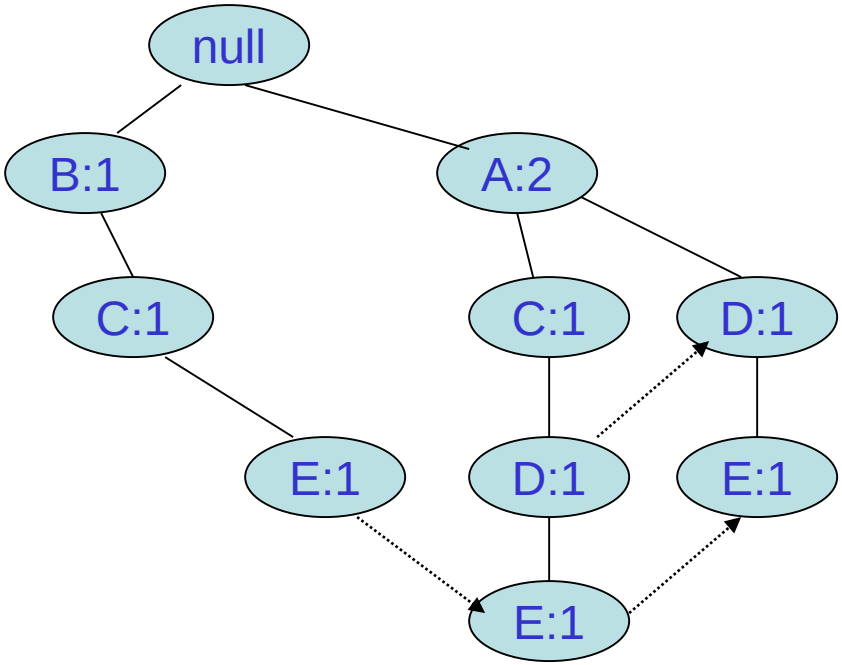
Conditional FP-Tree for E

- 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 previous slide as result of deleting nodes from the original tree. Why?

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

Suffix E

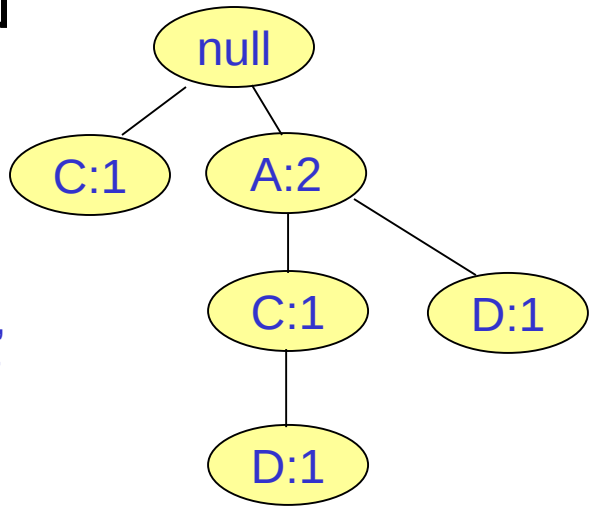


(New) Header table

A	2
C	2
D	2

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

Conditional FP-Tree for suffix E



The set of paths ending in E.

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

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

FI: E

Steps of Building Conditional FP-Trees

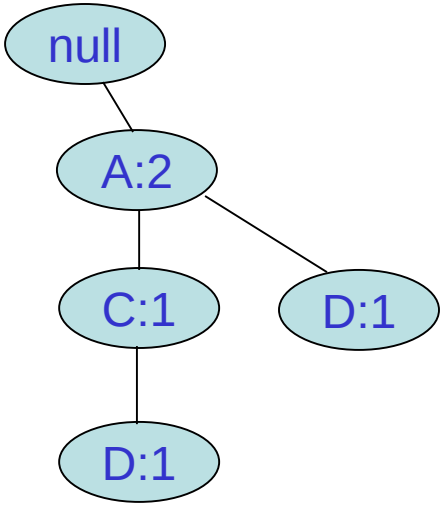
1. Find the paths containing on focus item.

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 in the conditional FP-Tree according to the new order.

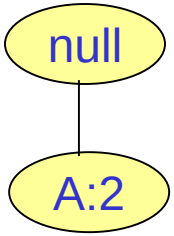
Suffix DE



(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.

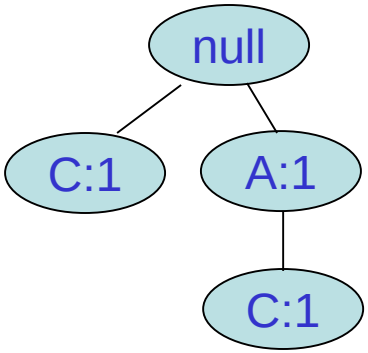
FI: DE, ADE

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 merged 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.

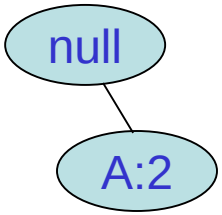
We have reached the base of recursion.

FI: CE

Suffix AE

(New) Header table

The conditional FP-Tree for suffix AE



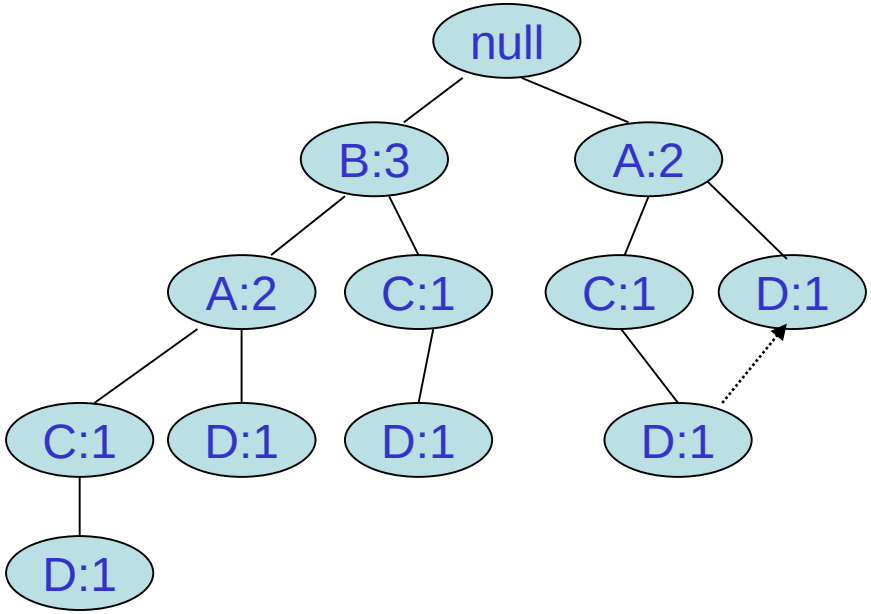
The set of paths, from the E-conditional FP-Tree, ending in A.

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

We have reached the base of recursion.

FI: AE

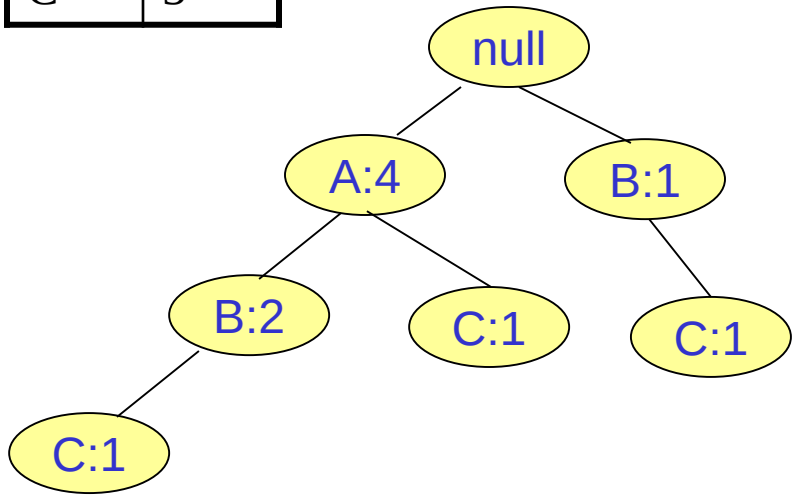
Suffix D



(New) Header table

A	4
B	3
C	3

Conditional FP-Tree for suffix D



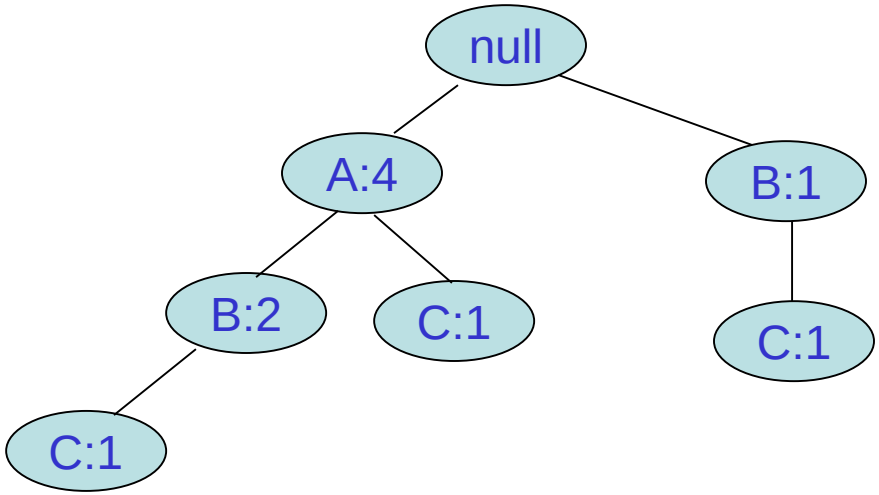
The set of paths ending in D.

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

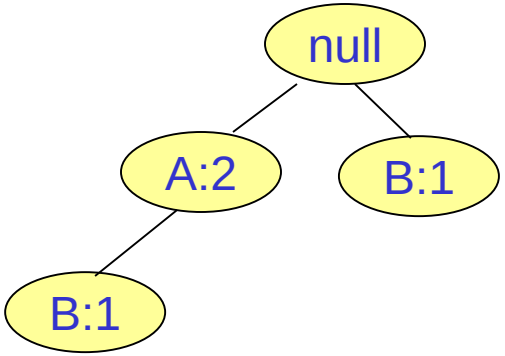
Suffix CD



(New) Header table

A	2
B	2

Conditional FP-Tree for suffix CD



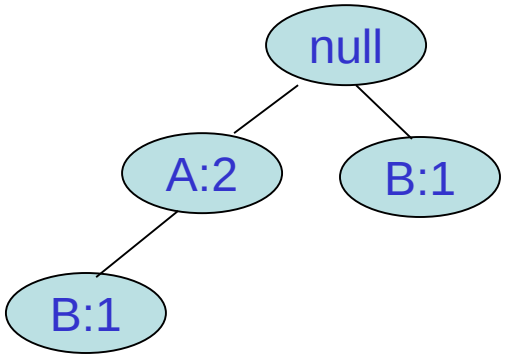
The set of paths, from the D-conditional FP-Tree, ending in C.

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



(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.

We have reached the base of recursion.

FI: BCD

Suffix ACD

(New) Header table

null

Conditional FP-Tree for suffix ACD

null

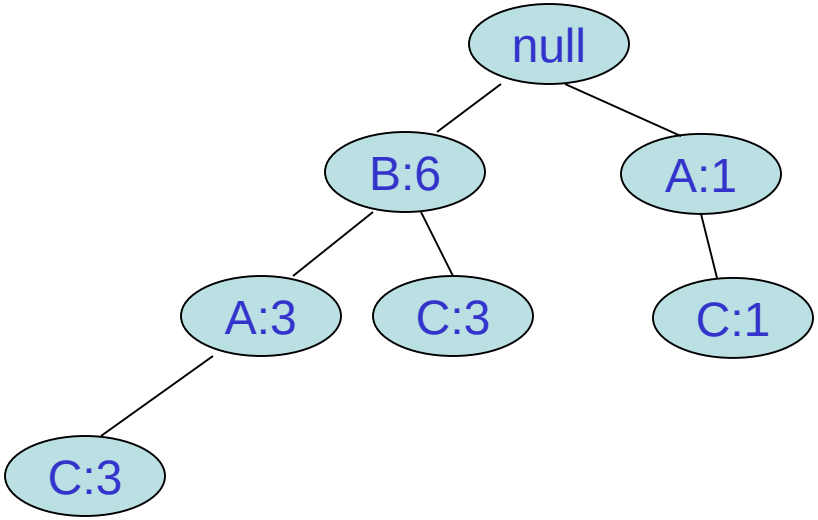
The set of paths from the CD-conditional FP-Tree, ending in A.

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

We have reached the base of recursion.

FI: ACD

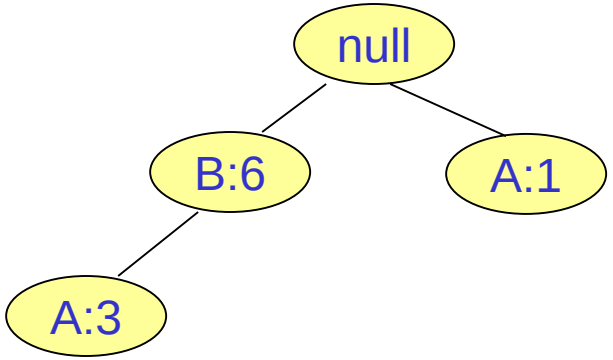
Suffix C



(New) Header table

B	6
A	4

Conditional FP-Tree for suffix C



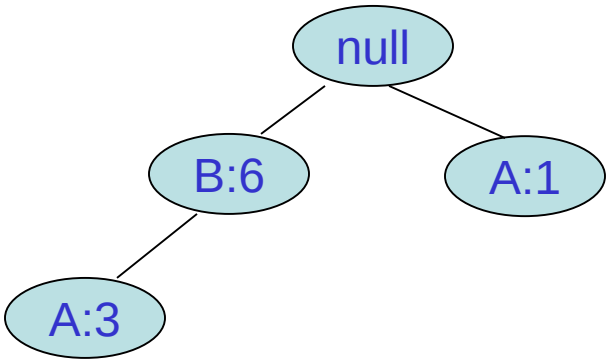
The set of paths ending in C.

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

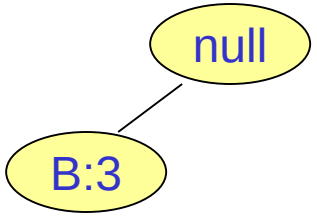
Suffix AC



(New) Header table

B	3

Conditional FP-Tree for suffix AC



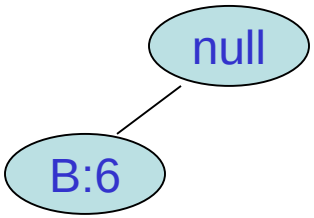
The set of paths from
the C-conditional FP-Tree, ending in A.

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

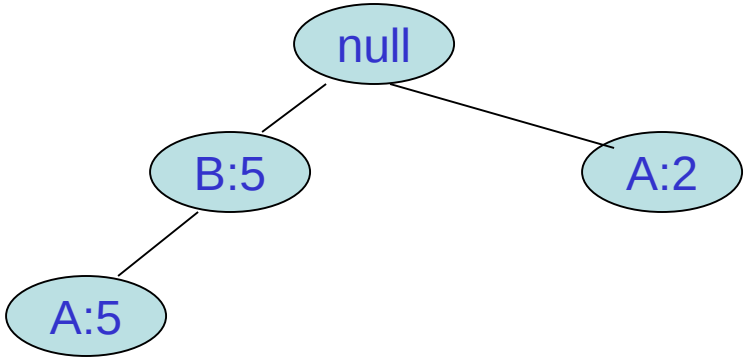
The set of paths from
the C-conditional FP-Tree, ending in B.

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

We have reached the base of recursion.

FI: BC

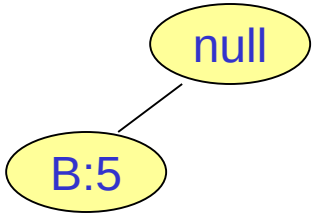
Suffix A



(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.

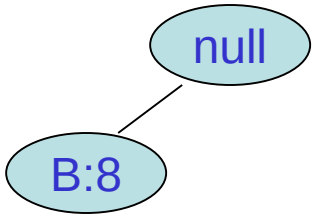
We have reached the base of recursion.

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

Array Technique

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

B	8
A	7
C	7
D	5
E	3

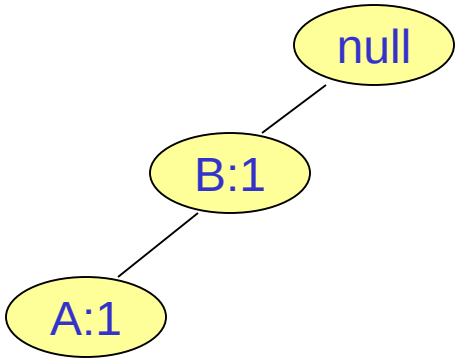
First pass on DB: Determine the header. Then sort it.

Second pass on DB: Build the FP-Tree. Also build an array of counts.

FP-Tree Construction – Reading 1

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

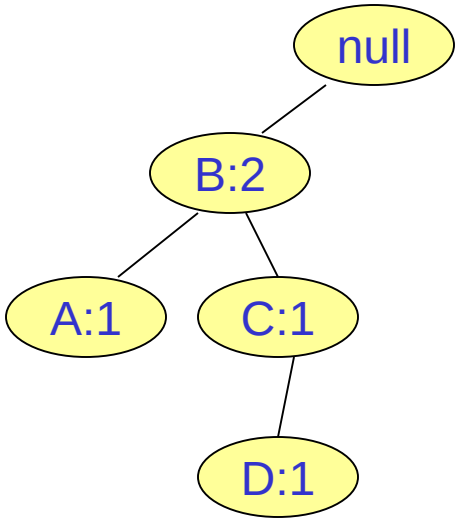
B	8
A	7
C	7
D	5
E	3

A	1			
C				
D				
E				
	B	A	C	D

FP-Tree Construction – Reading 2

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

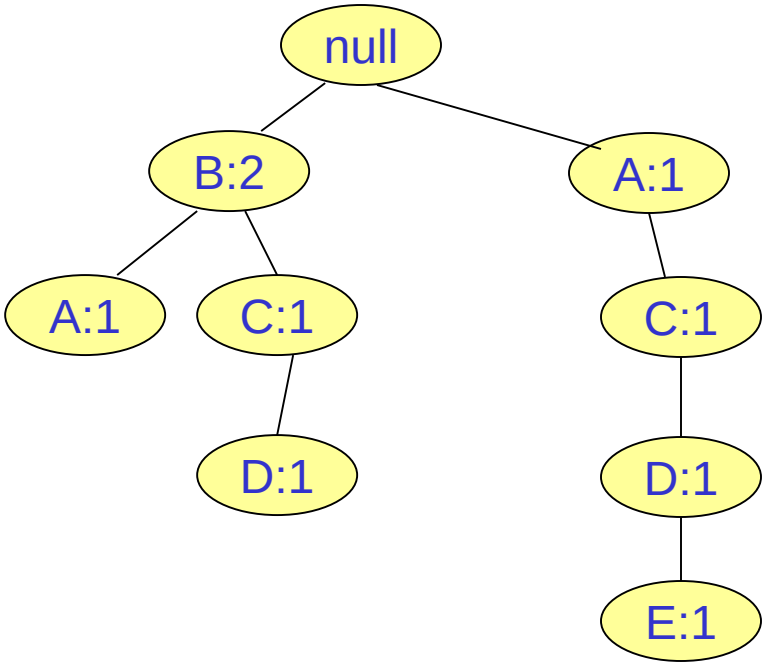
B	8
A	7
C	7
D	5
E	3

A	1			
C	1			
D	1		1	
E				
	B	A	C	D

FP-Tree Construction – Reading 3

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

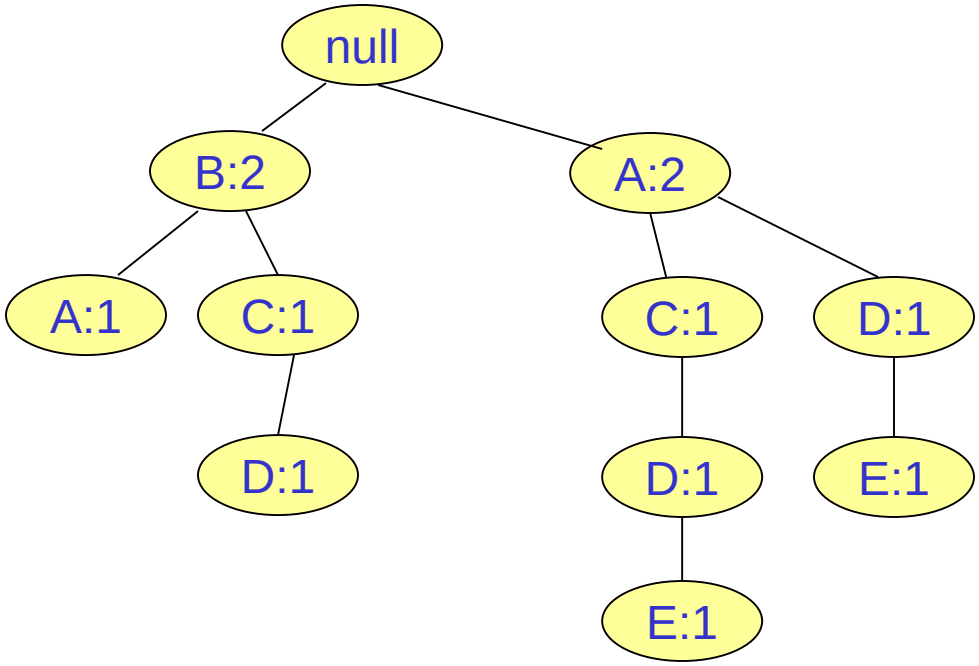
B	8
A	7
C	7
D	5
E	3

A	1			
C	1	1		
D	1	1	2	
E		1	1	1
	B	A	C	D

FP-Tree Construction – Reading 4

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

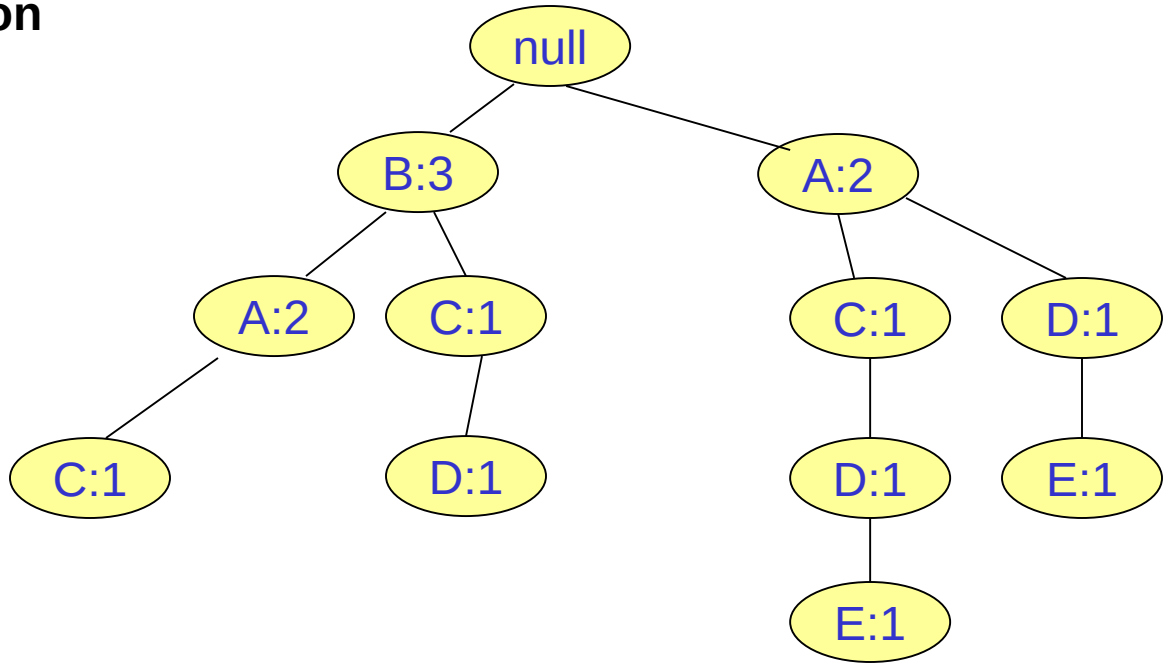
B	8
A	7
C	7
D	5
E	3

A	1			
C	1	1		
D	1	2	2	
E		2	1	2
	B	A	C	D

FP-Tree Construction – Reading 5

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

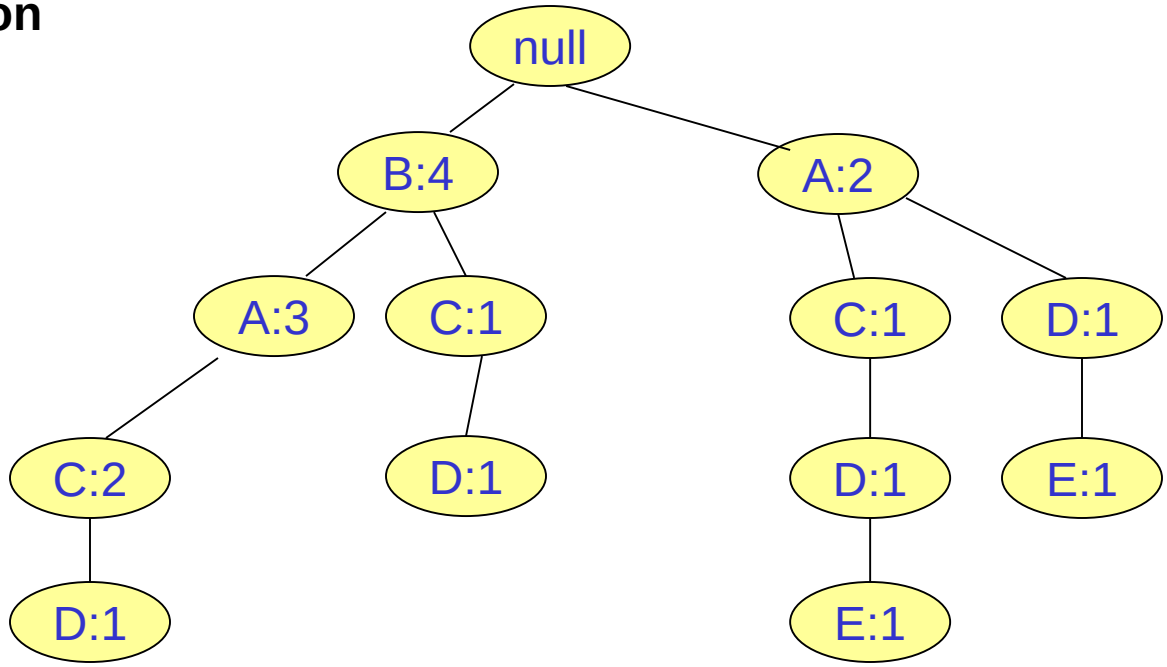
B	8
A	7
C	7
D	5
E	3

A	2			
C	2	2		
D	1	2	2	
E		2	1	2
	B	A	C	D

FP-Tree Construction – Reading 6

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

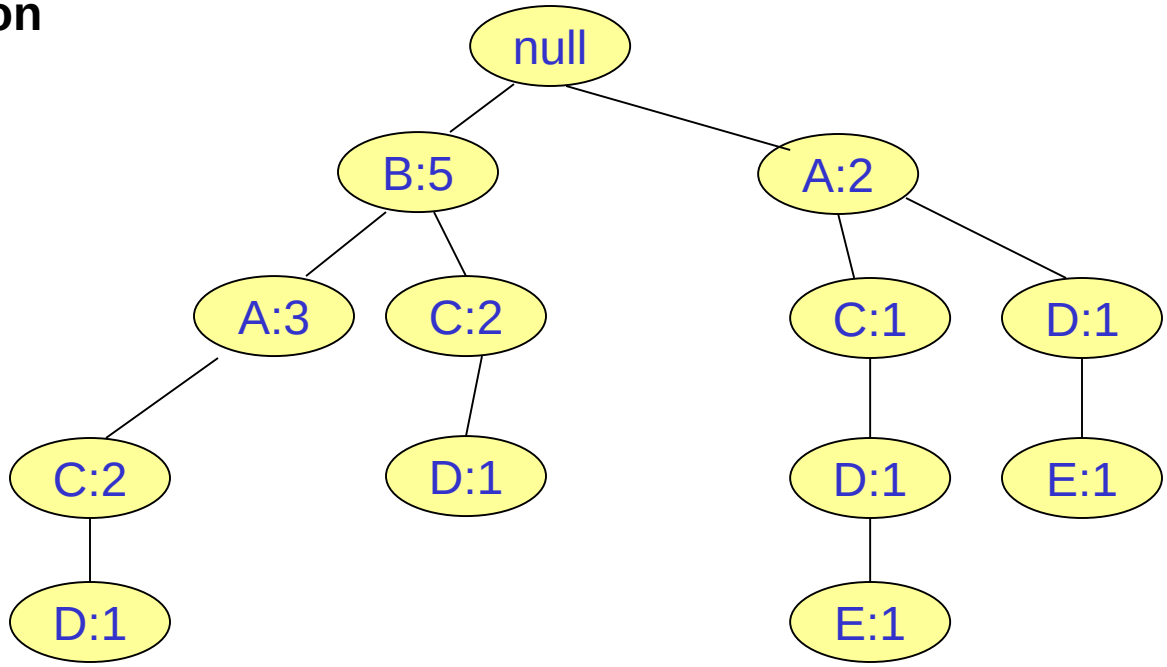
B	8
A	7
C	7
D	5
E	3

A	3			
C	3	3		
D	2	3	3	
E		2	1	2
	B	A	C	D

FP-Tree Construction – Reading 7

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

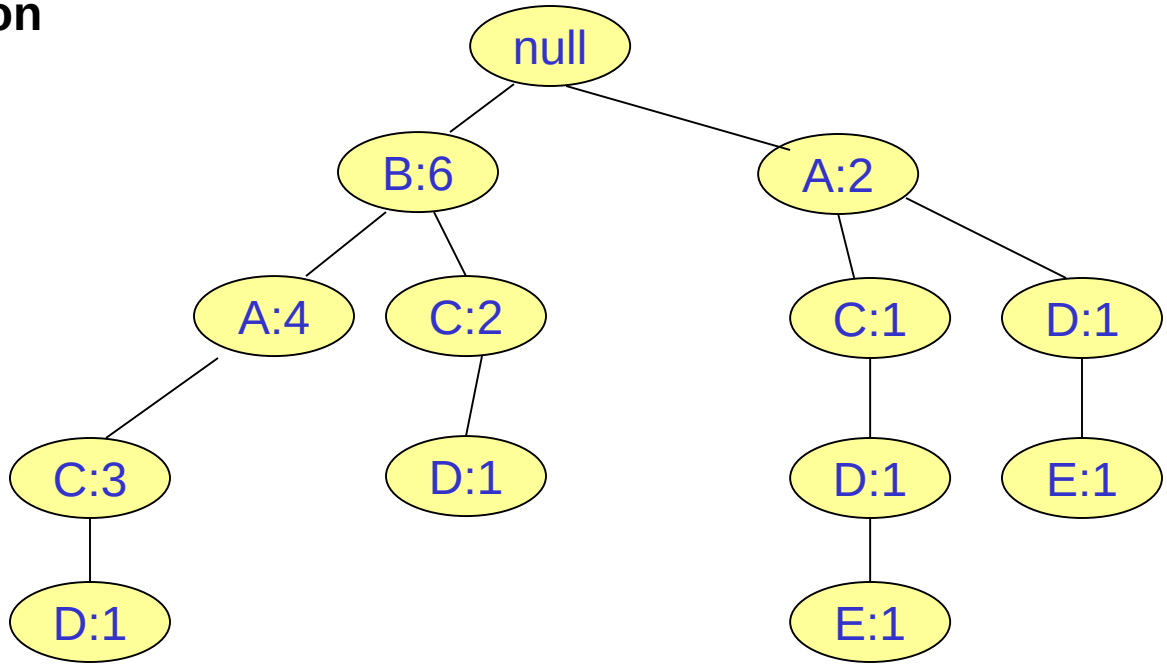
B	8
A	7
C	7
D	5
E	3

A	3			
C	4	3		
D	2	3	3	
E		2	1	2
	B	A	C	D

FP-Tree Construction – Reading 8

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

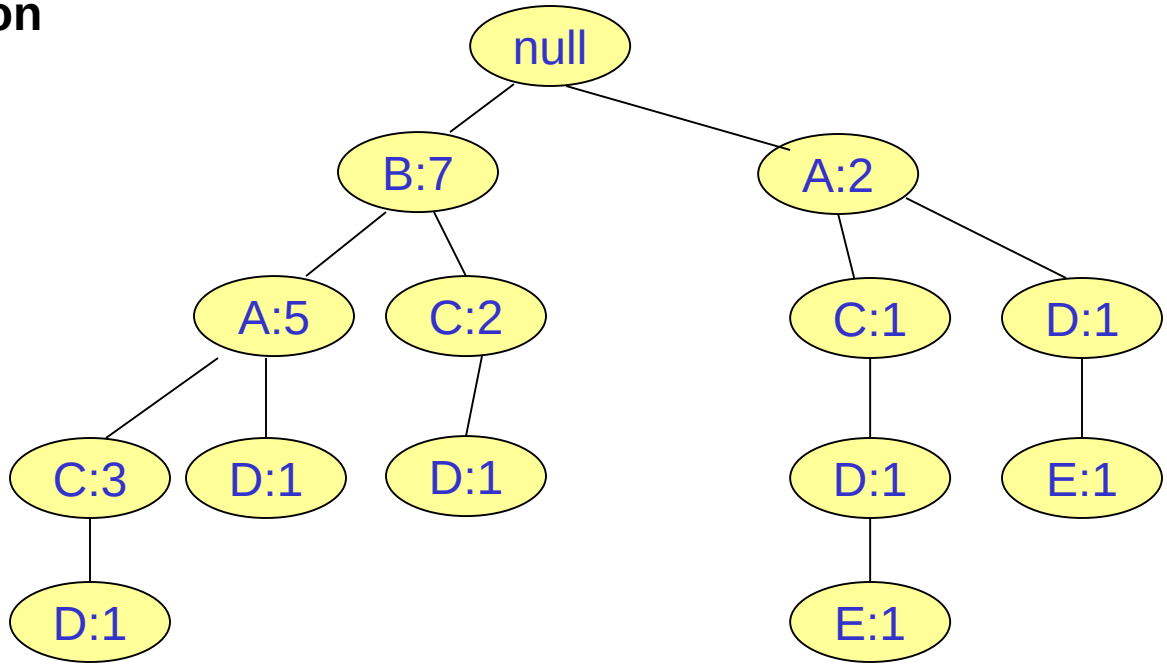
B	8
A	7
C	7
D	5
E	3

A	4			
C	5	4		
D	2	3	3	
E		2	1	2
	B	A	C	D

FP-Tree Construction – Reading 9

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

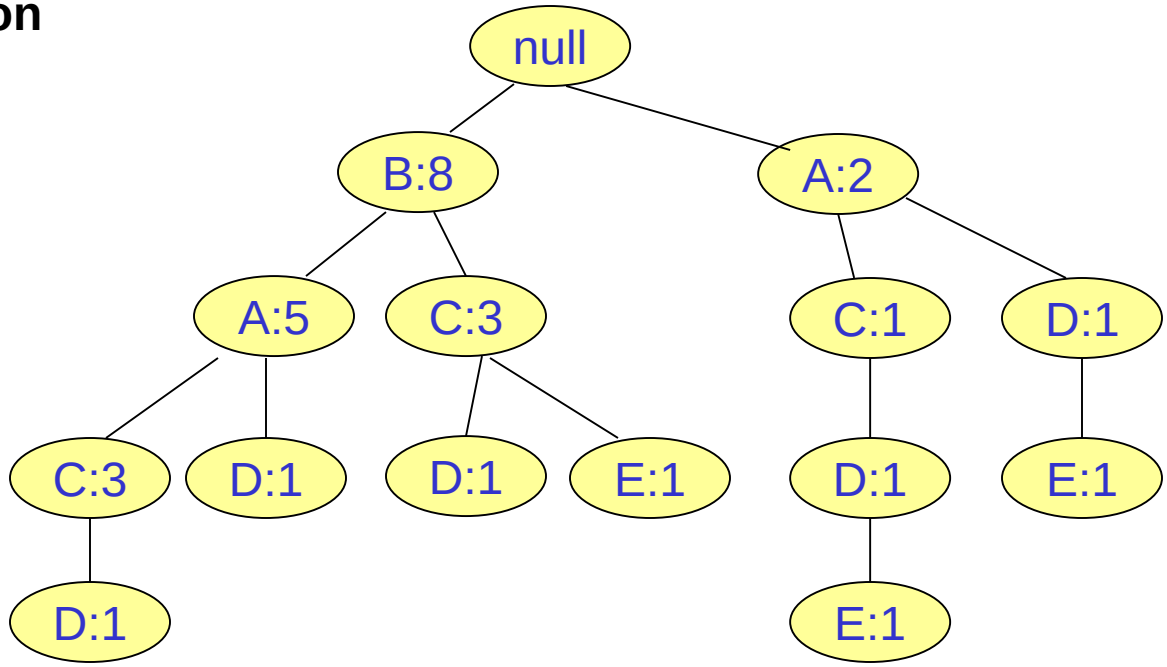
B	8
A	7
C	7
D	5
E	3

A	5			
C	5	4		
D	3	4	3	
E		2	1	2
	B	A	C	D

FP-Tree Construction – Reading 10

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

B	8
A	7
C	7
D	5
E	3

A	5			
C	6	4		
D	3	4	3	
E	1	2	2	2
	B	A	C	D

Why have the array?

Constructing conditional FP-Trees.

Without array

- Traverse the base FP-Tree to determine the new item counts.
  - Construct a new header.
- Traverse again the base FP-Tree and construct the conditional FP-Tree.

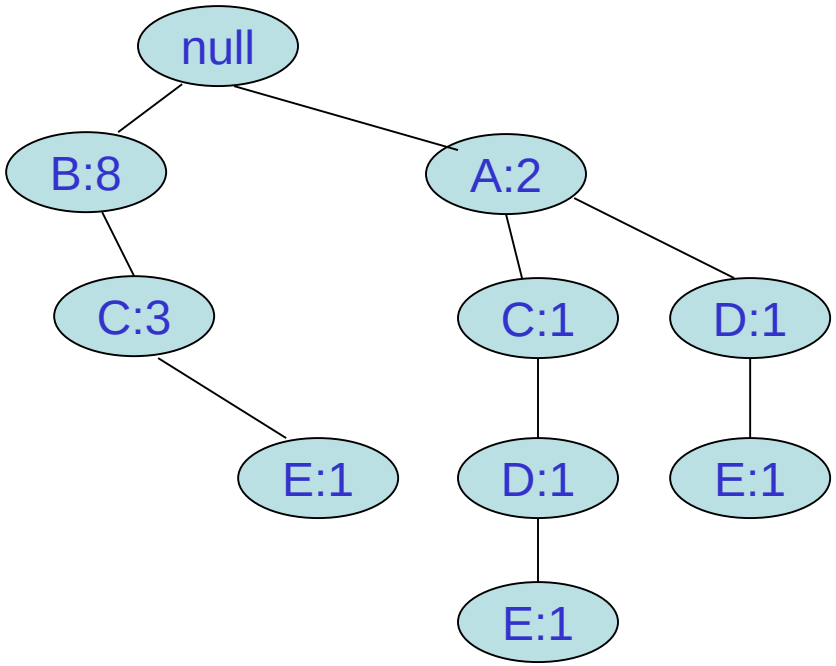
With array

- Construct a new header helped by the array.
- Traverse the base FP-Tree and construct the conditional FP-Tree.

Saving

- One tree traversal.
- Important because experimentally it's shown that 80% of time is spent on tree traversals.

Suffix E



(New) Header table

A	2
C	2
D	2

A	5			
C	6	4		
D	3	4	3	
E	1	2	2	2
	B	A	C	D

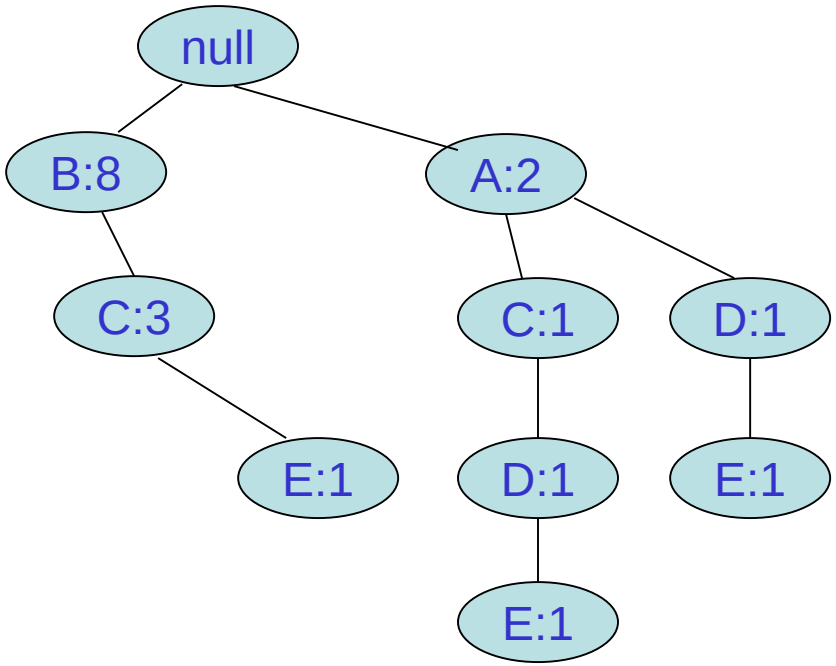
Conditional FP-Tree for suffix E

The set of paths ending in E.

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

C		
D		
	A	C

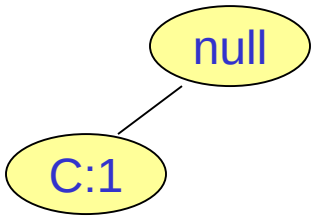
Suffix E (inserting BCE)



(New) Header table

A	2
C	2
D	2

Conditional FP-Tree for suffix E

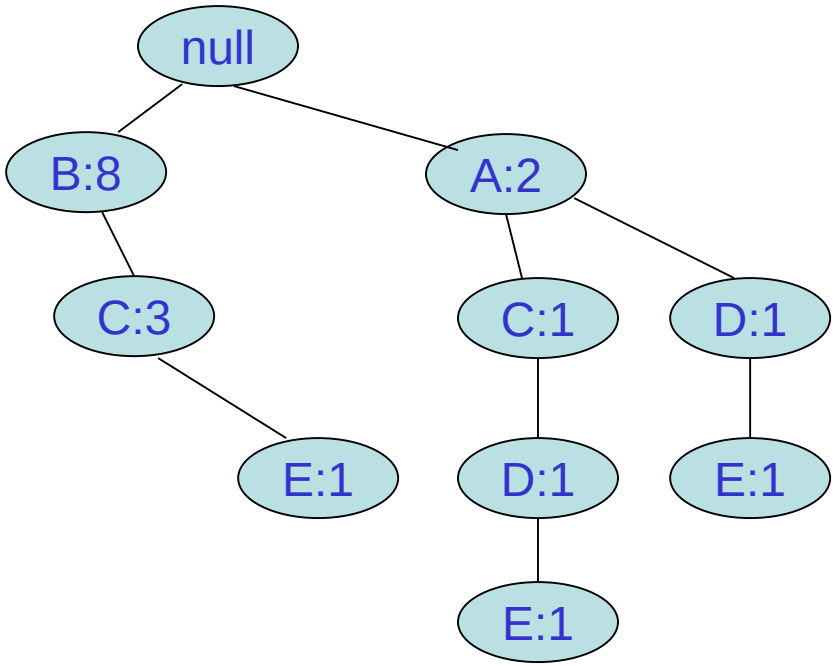


The set of paths ending in E.

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

C		
D		
	A	C

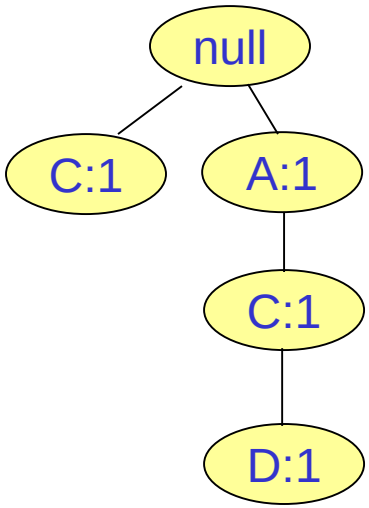
Suffix E (inserting **ACD**E)



(New) Header table

A	2
C	2
D	2

Conditional FP-Tree for suffix E

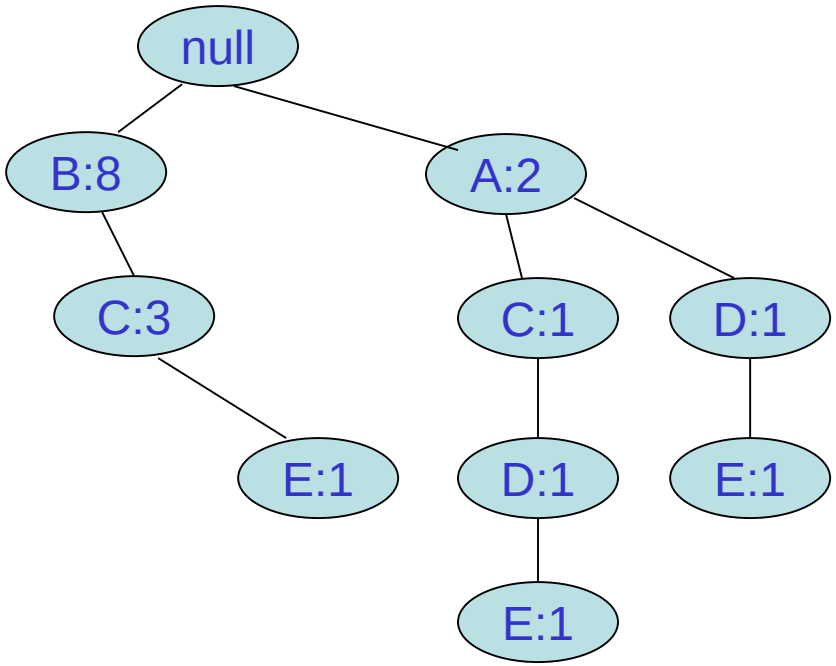


The set of paths ending in E.

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

C	1	
D	1	1
	A	C

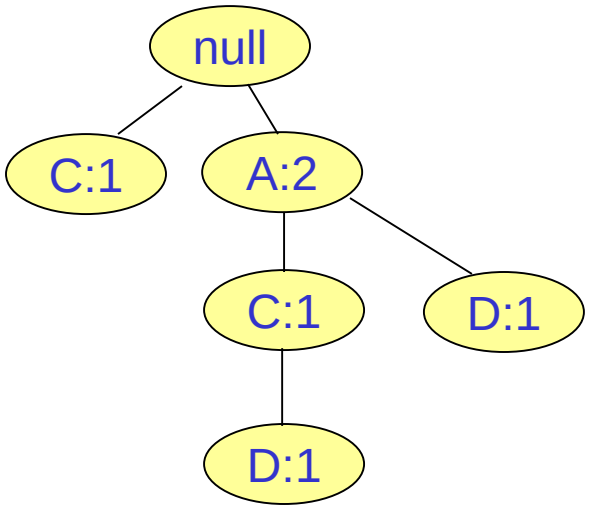
Suffix E (inserting **AD**E)



(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.

C	1	
D	2	1
	A	C