Data Mining and Web Algorithms

Course Code: 15B22CI621

Credits: 4 [3+ 1]

Frequent Pattern Mining: An Example

Given a transaction database DB and a minimum support threshold ξ, find all frequent patterns (item sets) with support no less than ξ.

Input: DB:

TID	*Items bought*
100	{*f, a, c, d, g, i, m, p*}
200	{*a, b, c, f, l, m, o*}
300	{*b, f, h, j, o*}
400	{*b, c, k, s, p*}
500	{*a, f, c, e, l, p, m, n*}

Minimum support: ξ =3

Output: all frequent patterns, i.e., *f, a,* …, *fa, fac, fam, fm,am*…

Problem Statement: How to **efficiently** find all frequent patterns?

Apriori

- Main Steps of Apriori Algorithm:

  Candidate Generation

  – Use frequent (*k* – 1)-itemsets (L_{k-1}) to generate candidates of frequent *k*-itemsets C_k

  – Scan database and count each pattern in C_k , get frequent *k*-itemsets (L_k) .

  Candidate Test

- E.g. ,

TID	*Items bought*
100	{*f, a, c, d, g, i, m, p*}
200	{*a, b, c, f, l, m, o*}
300	{*b, f, h, j, o*}
400	{*b, c, k, s, p*}
500	{*a, f, c, e, l, p, m, n*}

Apriori	*iteration*
C1	*f,a,c,d,g,i,m,p,l,o,h,j,k,s,b,e,n*
L1	*f, a, c, m, b, p*
C2	*fa, fc, fm, fp, ac, am, …bp*
L2	*fa, fc, fm, …*
…	

Performance Bottlenecks of Apriori

- Bottlenecks of *Apriori*: candidate generation
  - Generate huge candidate sets.
  - Candidate Test incur multiple scans of database.

Overview of FP-Growth: Ideas

- Compress a large database into a compact, *Frequent-Pattern tree* (FP-tree) structure
  - highly compacted, but complete for frequent pattern mining
  - avoid costly repeated database scans
- Develop an efficient, FP-tree-based frequent pattern mining method (FP-growth)
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only.

FP-tree:
Construction and Design

Construct FP-tree

Two Steps:

1. Scan the transaction DB for the first time, find frequent items (single item patterns) and order them into a list L in frequency descending order.

   2. For each transaction, order its frequent items according to the order in L; Scan DB the second time, construct FP-tree by putting each frequency ordered transaction onto it.

FP-tree Example: step 1

Step 1: Scan DB for the first time to generate L

TID	*Items bought*
100	{*f, a, c, d, g, i, m, p*}
200	{*a, b, c, f, l, m, o*}
300	{*b, f, h, j, o*}
400	{*b, c, k, s, p*}
500	{*a, f, c, e, l, p, m, n*}

Item	*frequency*
f	*4*
c	*4*
a	*3*
b	*3*
m	*3*
p	*3*

By-Product of First Scan of Database

FP-tree Example: step 2

Step 2: scan the DB for the second time, order frequent items in each transaction

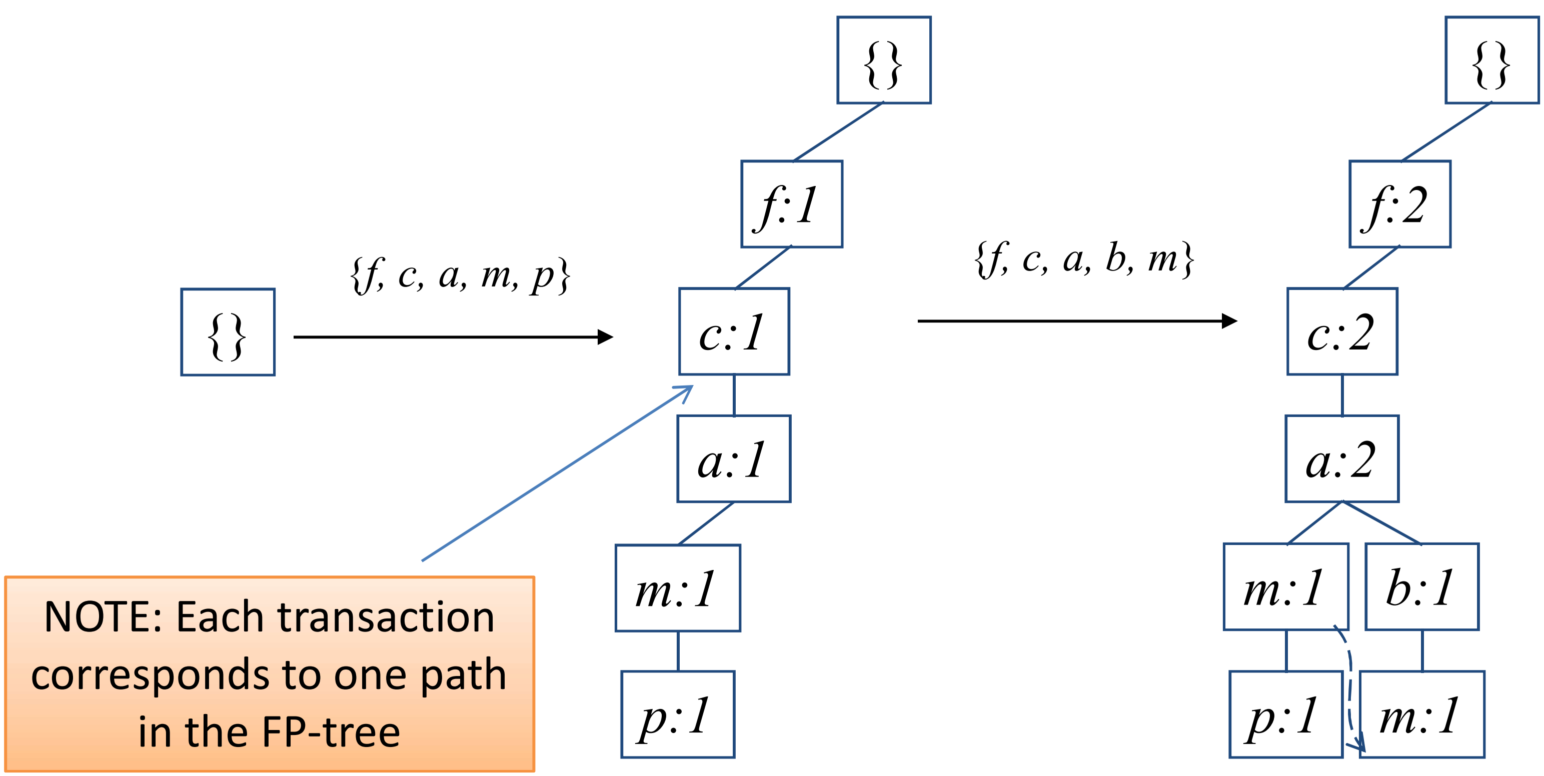
TID	*Items bought*	*(ordered) frequent items*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

Item	*frequency*
f	*4*
c	*4*
a	*3*
b	*3*
m	*3*
p	*3*

FP-tree Example: step 2

Step 2: construct FP-tree

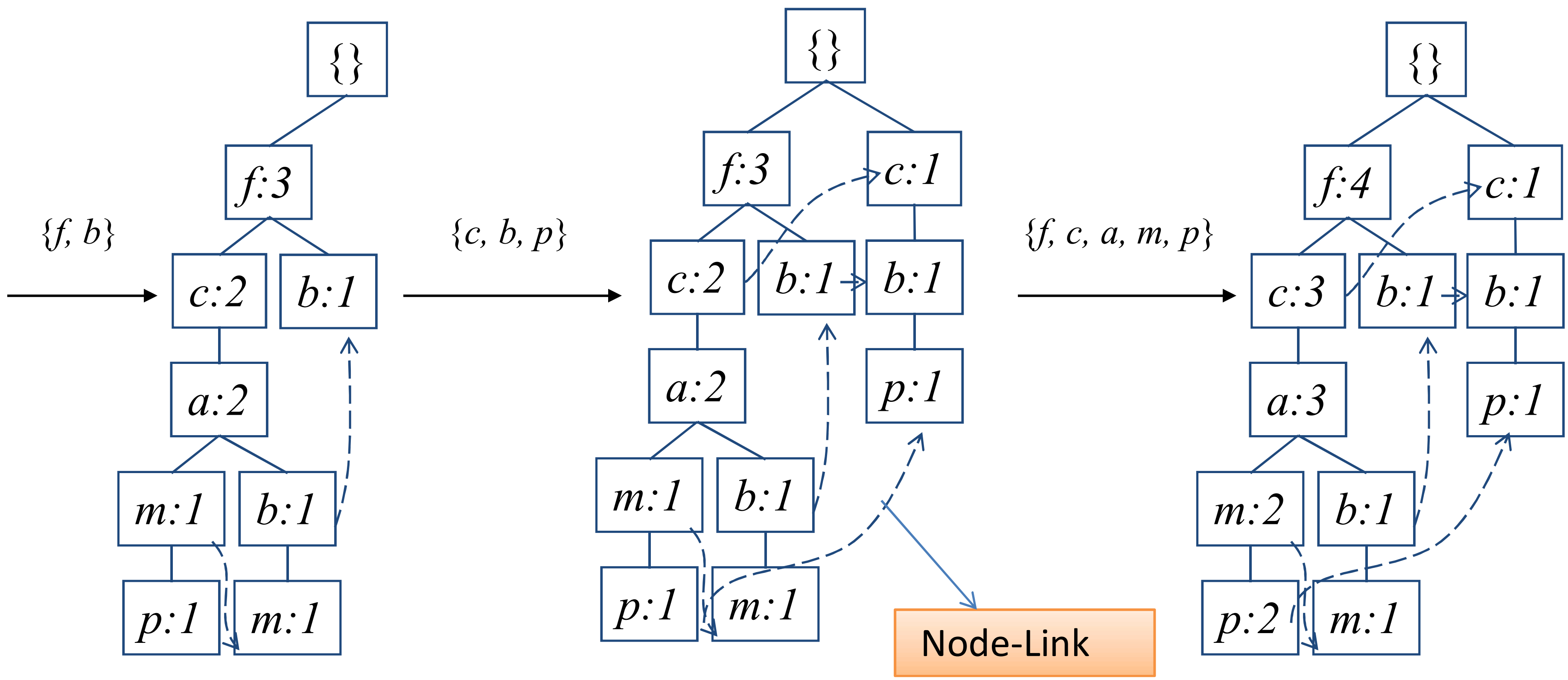
TID	*Items bought*	*(ordered)*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}



FP-tree Example: step 2

TID	Items bought	(ordered)
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o*}	**{*f, b*}**
400	{*b, c, k, s, p*}	**{*c, b, p*}**
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

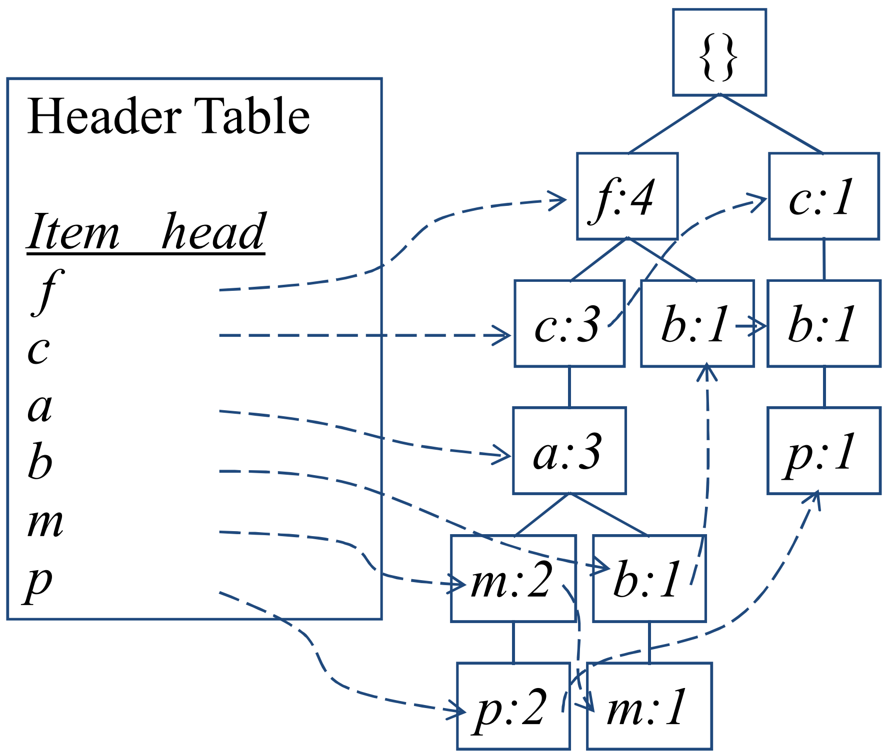
Step 2: construct FP-tree



Construction Example

TID	*Items bought*	*(ordered)*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

Final FP-tree



Advantages of the FP-tree Structure

- The most significant advantage of the FP-tree
  - Scan the DB only twice and twice only.
- Completeness:
  - the FP-tree contains all the information related to mining frequent patterns (given the min-support threshold). Why?
- Compactness:
  - The size of the tree is bounded by the occurrences of frequent items
  - The height of the tree is bounded by the maximum number of items in a transaction

FP-growth:
Mining Frequent Patterns Using FP-tree

Properties of FP-Tree

- Node-link property
  - For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header.
- Prefix path property
  - To calculate the frequent patterns for a node a_i in a path P, only the prefix sub-path of a_i in P need to be accumulated, and its frequency count should carry the same count as node a_i.

Principles of FP-Growth

- Pattern growth property
  - Let α be a frequent itemset in DB, B be α's conditional pattern base, and β be an itemset in B. Then $\alpha \cup \beta$ is a frequent itemset in DB iff β is frequent in B.
- Is "*fcabm* " a frequent pattern?
  - "fcab" is a branch of m's conditional pattern base
  - "b" is **NOT** frequent in transactions containing "fcab "
  - "bm" is **NOT** a frequent itemset.

Mining Frequent Patterns Using FP-tree

- General idea (divide-and-conquer)

  Recursively grow frequent patterns using the FP-tree: looking for shorter ones recursively and then concatenating the suffix:

  - For each frequent item, construct its conditional pattern base, and then its conditional FP-tree;
  - Repeat the process on each newly created conditional FP-tree until the resulting FP-tree is empty, or it contains only one path (single path will generate all the combinations of its sub-paths, each of which is a frequent pattern)

Major Steps

Starting the processing from the end of list L:

Step 1:

Construct **conditional pattern base** for each item in the header table

Step 2

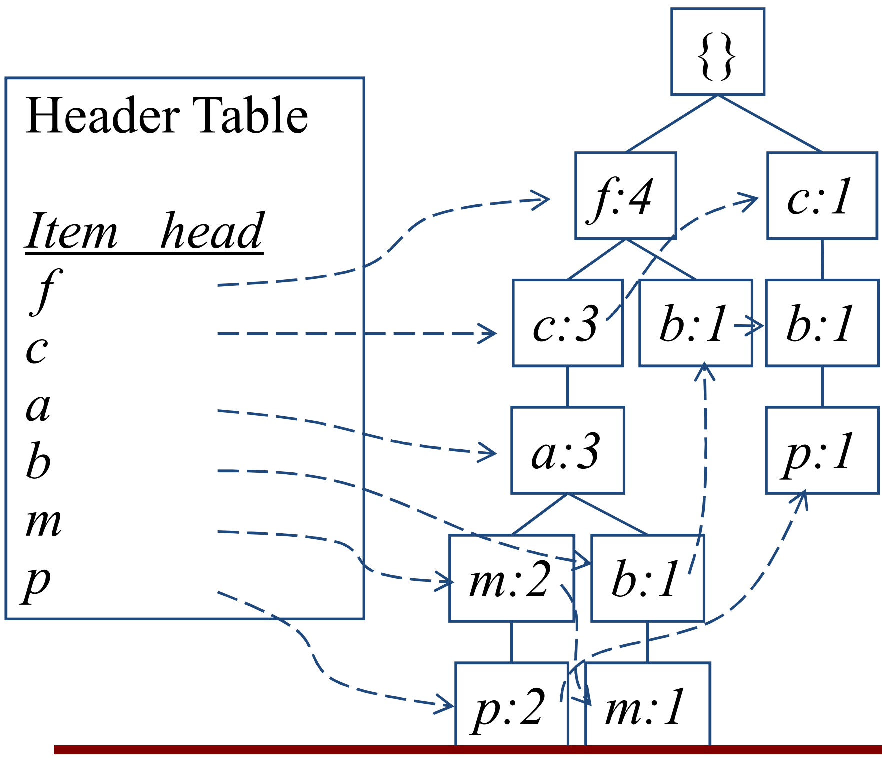
Construct **conditional FP-tree** from each conditional pattern base

Step 3

Recursively mine conditional FP-trees and grow frequent patterns obtained so far. If the conditional FP-tree contains a single path, simply enumerate all the patterns

Step 1: Construct Conditional Pattern Base

- Starting at the bottom of frequent-item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of **transformed prefix paths** of that item to form a **conditional pattern base**



***Conditional* pattern bases**

item	***cond. pattern base***
p	***fcam:2, cb:1***
m	***fca:2, fcab:1***
b	***fca:1, f:1, c:1***
a	***fc:3***
c	***f:3***
f	***{ }***

Step 2: Construct Conditional FP-tree

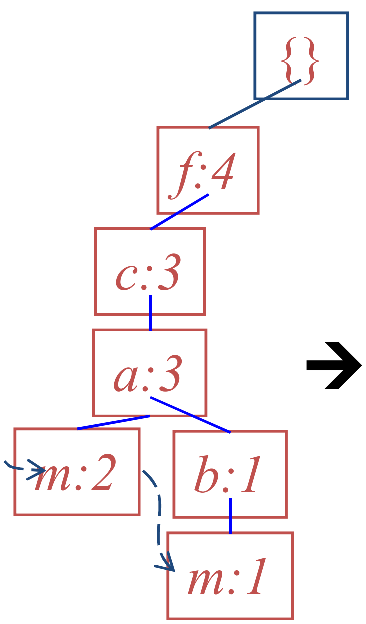
- For each pattern base
  - Accumulate the count for each item in the base
  - Construct the conditional FP-tree for the frequent
  - items of the pattern base

***Conditional* pattern bases**

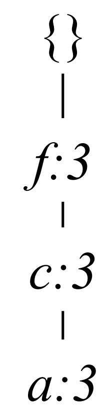
item	***cond. pattern base***
p	***fcam:2, cb:1***
m	***fca:2, fcab:1***
b	***fca:1, f:1, c:1***
a	***fc:3***
c	***f:3***
f	***{ }***

Header Table

Item	***head***
f	*4*
c	*4*
a	*3*
b	*3*
m	*3*
p	*3*



➔ ***m*- *cond.* pattern base:**
fca:2, fcab:1 ➔



***m-conditional* FP-tree**

Conditional Pattern Bases and Conditional FP-Tree

Conditional **pattern bases**

item	***cond. pattern base***
p	***fcam:2, cb:1***
m	***fca:2, fcab:1***
b	***fca:1, f:1, c:1***
a	***fc:3***
c	***f:3***
f	***{ }***

Item	Conditional pattern base	Conditional FP-tree
p	{(fcam:2), (cb:1)}	{(c:3)}|p
m	{(fca:2), (fcab:1)}	{(f:3, c:3, a:3)}|m
b	{(fca:1), (f:1), (c:1)}	Empty
a	{(fc:3)}	{(f:3, c:3)}|a
c	{(f:3)}	{(f:3)}|c
f	Empty	Empty

order of L

Single FP-tree Path Generation

- Suppose an FP-tree T has a single path P. The complete set of frequent pattern of T can be generated by enumeration of all the combinations of the sub-paths of P

{}
|
f:3
|
c:3
|
a:3

m-conditional FP-tree

➔

All frequent patterns concerning *m*: combination of {f, c, a} and *m*

m,

fm, cm, am,

fcm, fam, cam,

fcam

Example : FP Growth

TID	List of Items
T100	I1, I2, I5
T101	I2, I4
T102	I2, I3
T103	I1, I2, I4
T104	I1, I3
T105	I2, I3
T106	I1, I3
T107	I1, I2 ,I3, I5
T108	I1, I2, I3

- Consider a database, D , consisting of 9 transactions.
- Let support count =2

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Example : FP Growth cont...

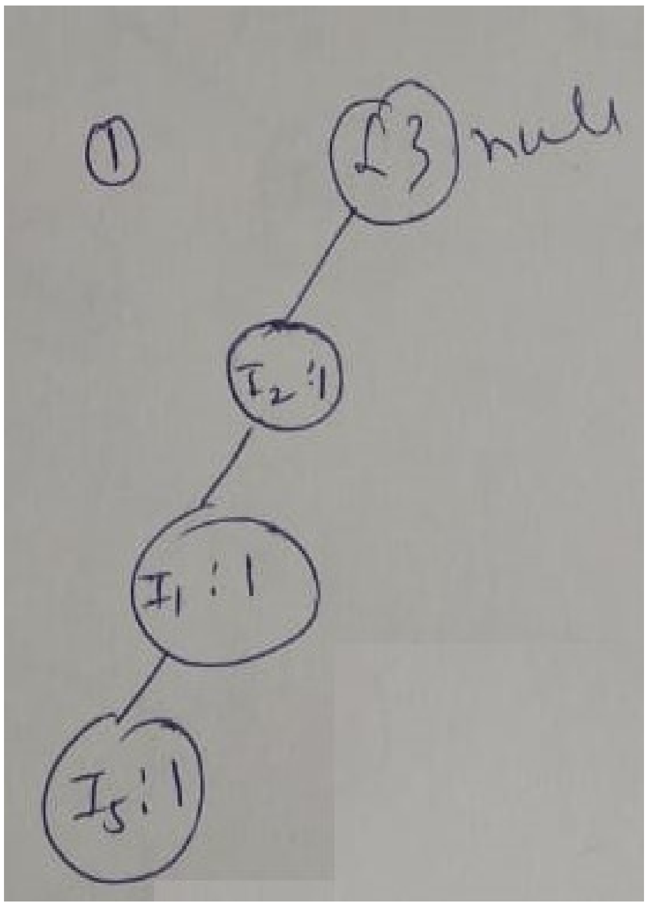
Sort the items in each transactions according to its support count

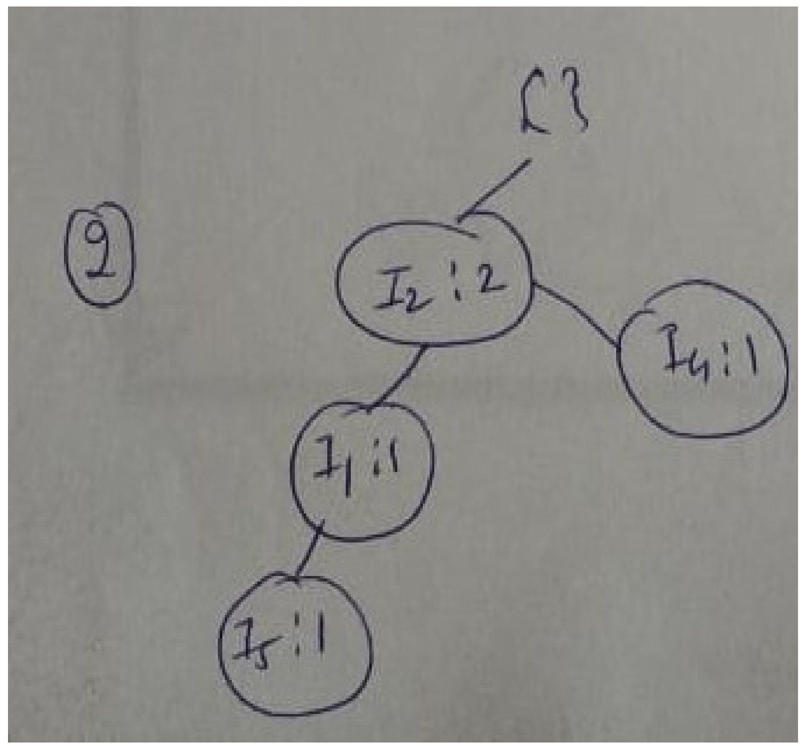
I_2 (7)
I_1 (6)
I_3 (6)
I_4 (2)
I_5 (2)

1	I_2 I_1 I_5
2	I_2 I_4
3	I_2 I_3
4	I_2 I_1 I_4
5	I_1 I_3
6	I_2 I_3
7	I_1 I_3
8	I_2 I_1 I_3 I_5
9	I_2 I_1 I_3

Example : FP Growth cont...

1	I_2 I_1 I_5
2	I_2 I_4
3	I_2 I_3
4	I_2 I_1 I_4

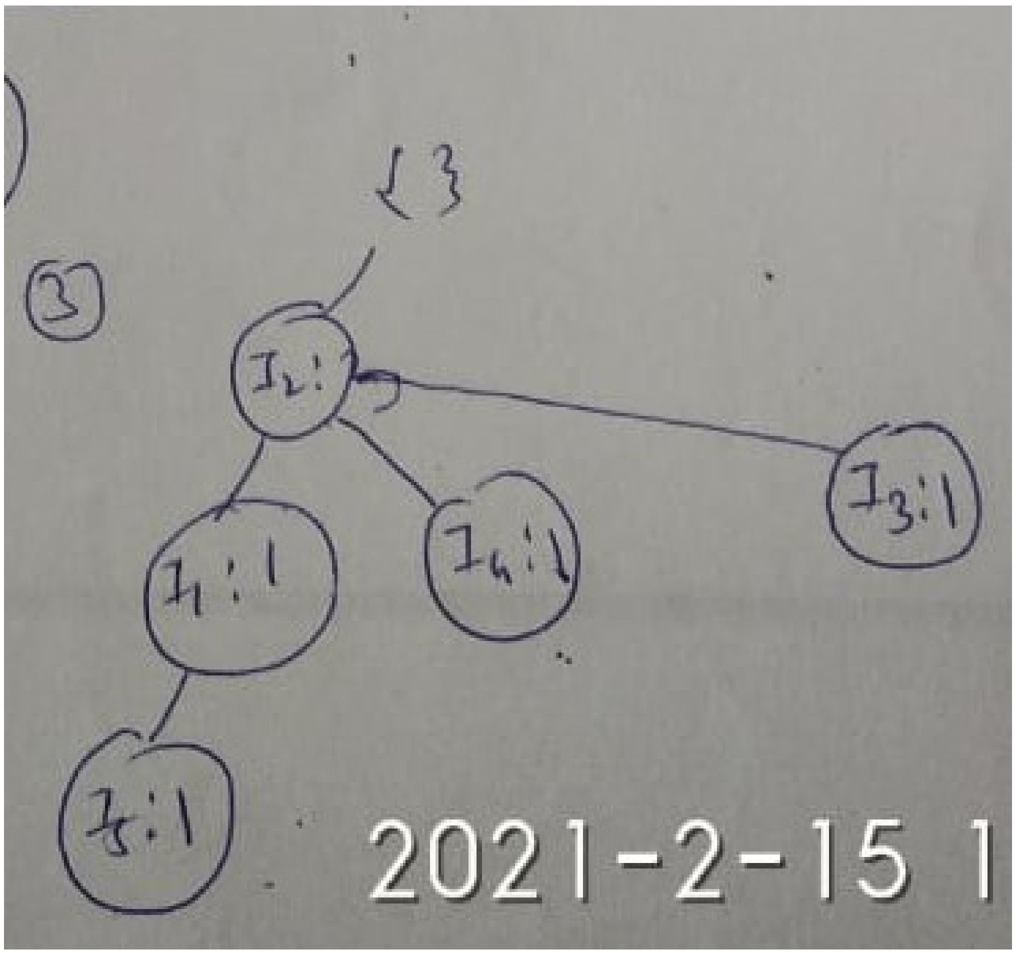




Example : FP Growth cont…

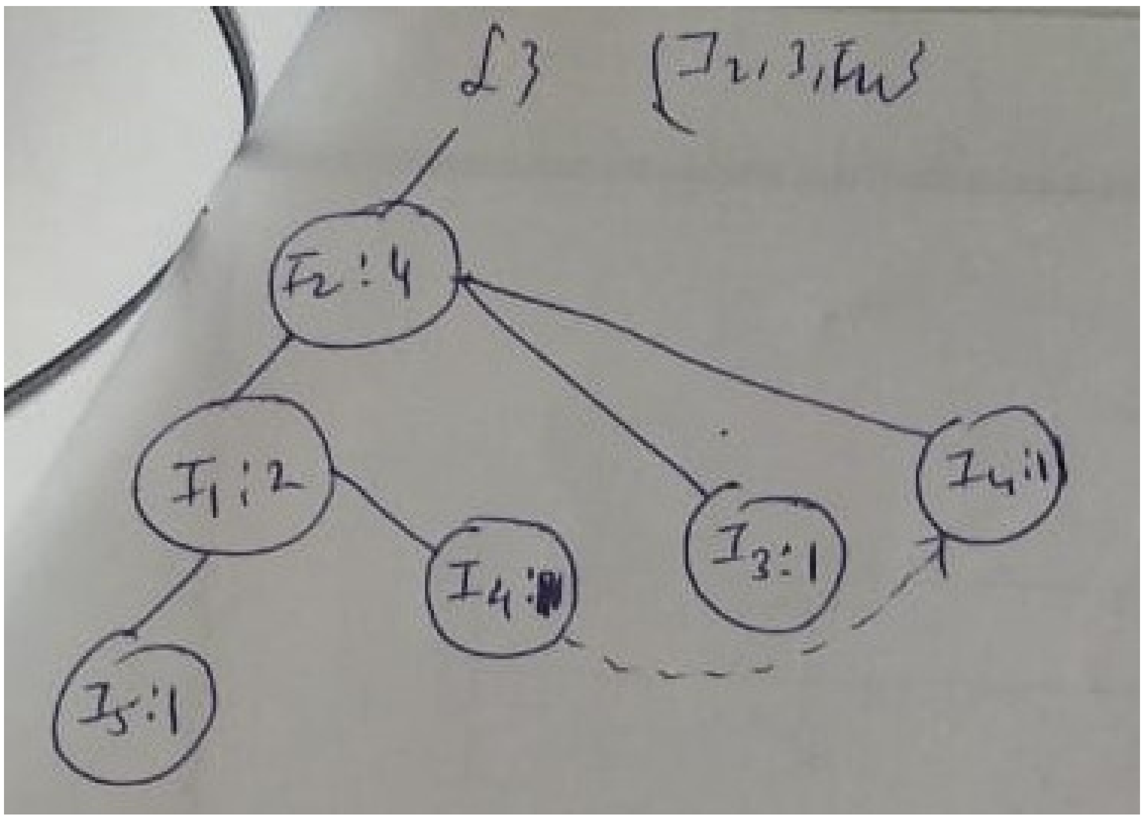
3	I_2 I_3

1	I_2 I_1 I_5
2	I_2 I_4
3	I_2 I_3
4	I_2 I_1 I_4

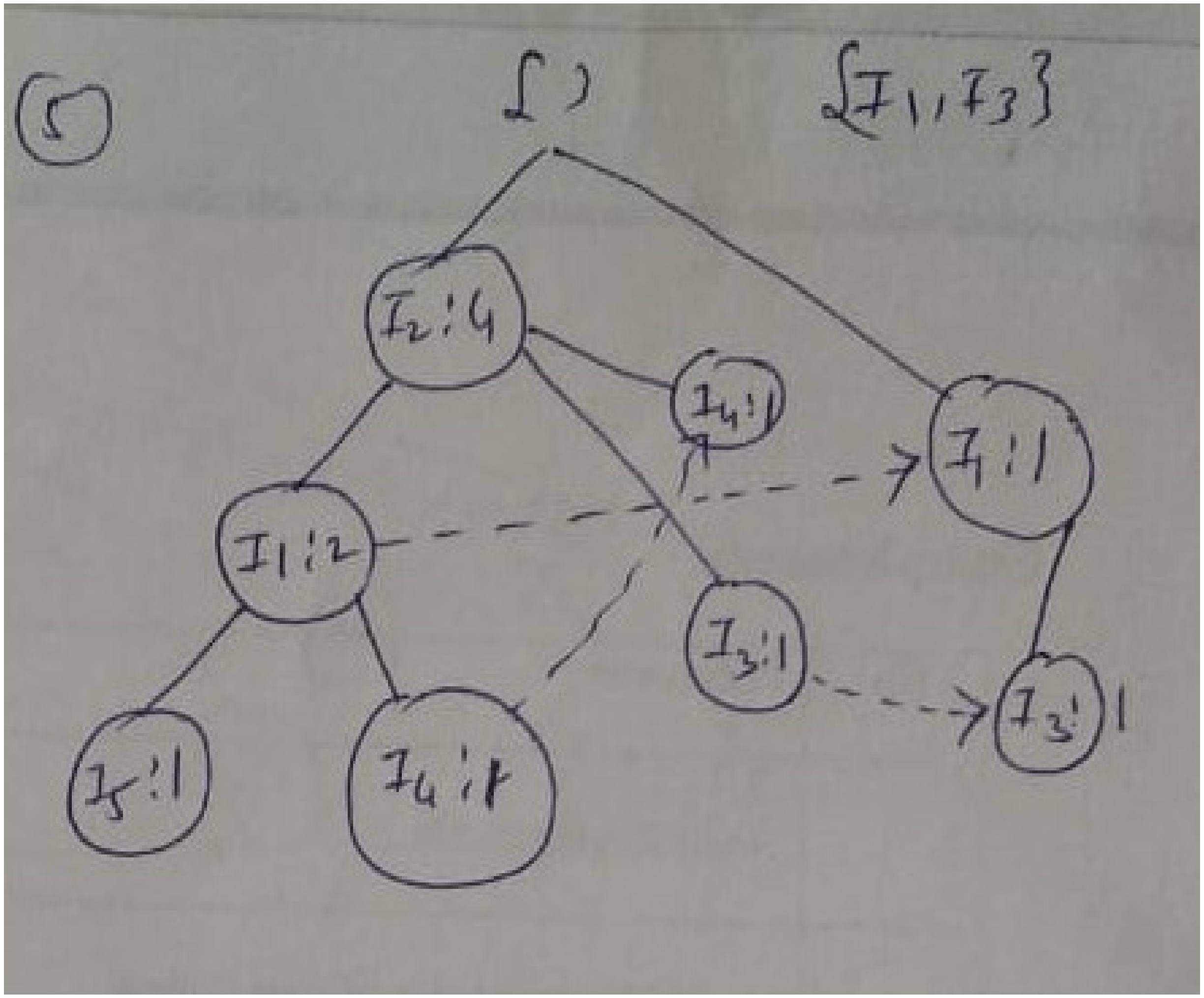


Example : FP Growth cont…

1	I_2 I_1 I_5
2	I_2 I_4
3	I_2 I_3
4	I_2 I_1 I_4

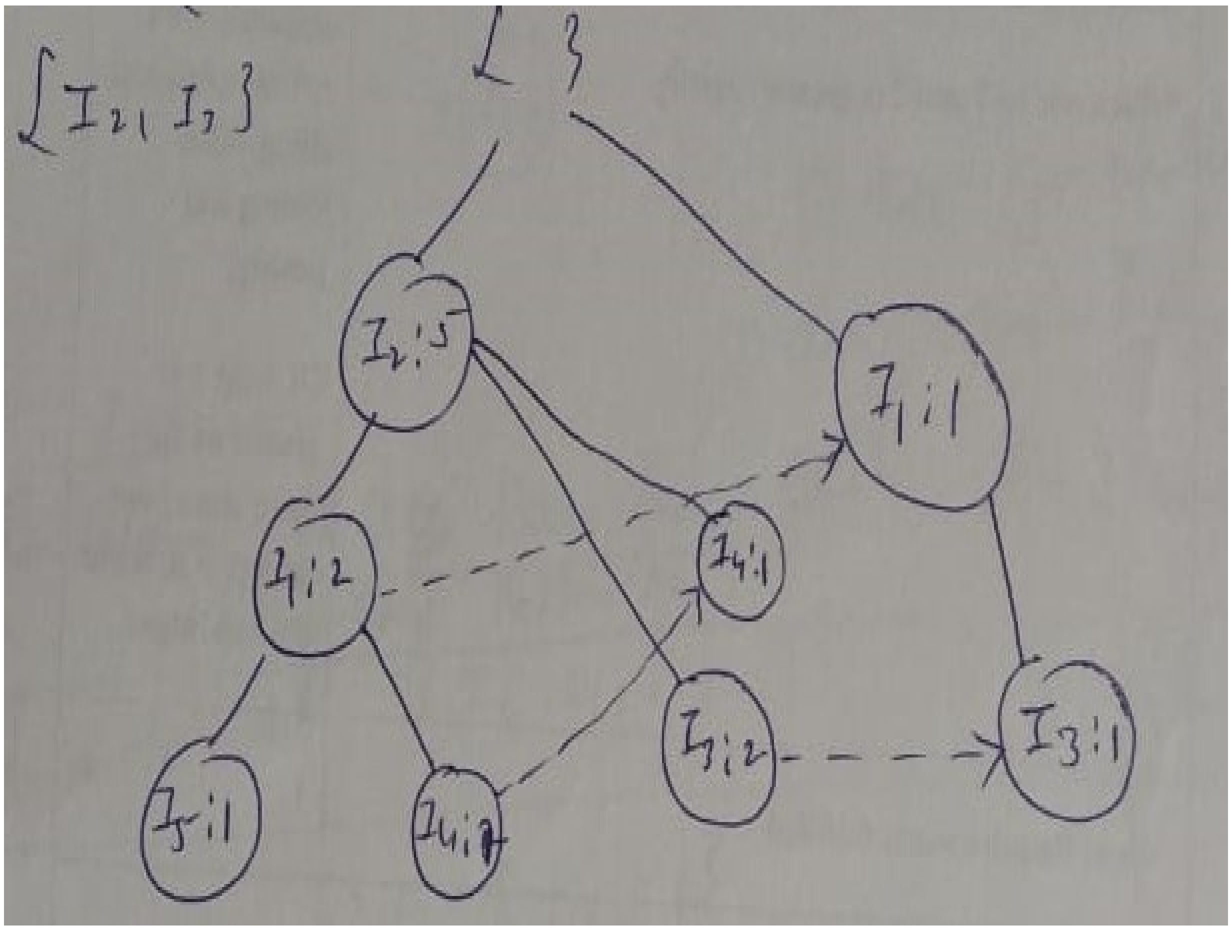


Example : FP Growth cont…



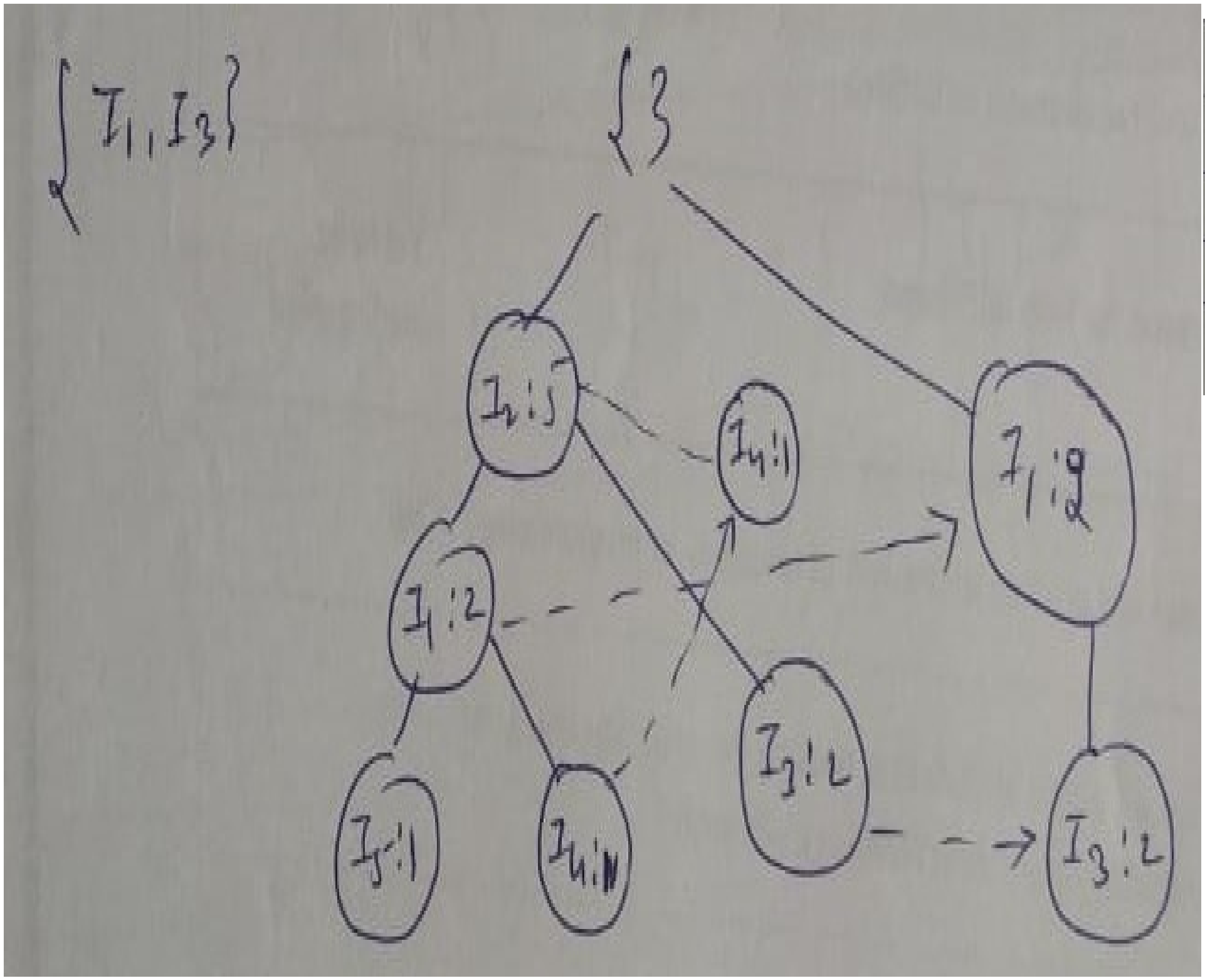
1	I_2 I_1 I_5
2	I_2 I_4
3	I_2 I_3
4	I_2 I_1 I_4
5	I_1 I_3
6	I_2 I_3
7	I_1 I_3
8	I_2 I_1 I_3 I_5
9	I_2 I_1 I_3

Example : FP Growth cont...



5	I_1 I_3
6	I_2 I_3
7	I_1 I_3
8	I_2 I_1 I_3 I_5
9	I_2 I_1 I_3

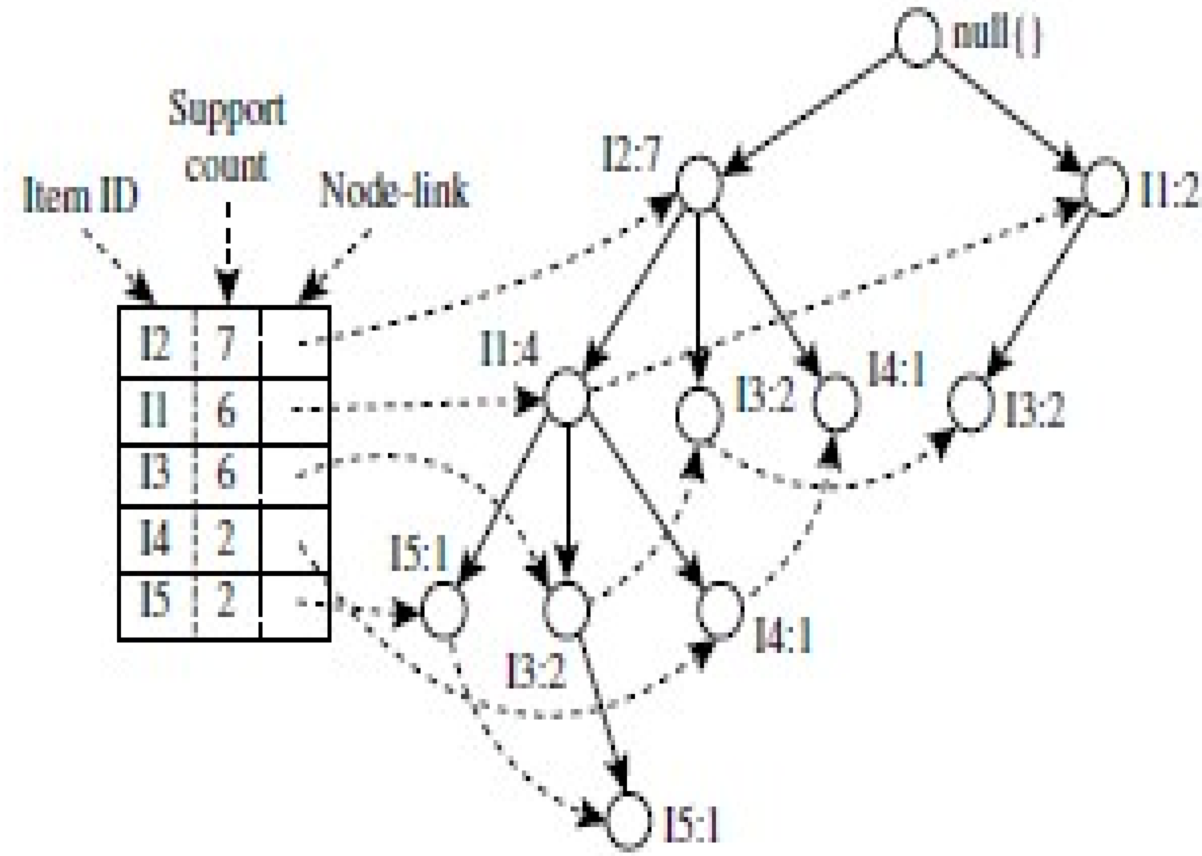
Example : FP Growth cont…



5	I_1 I_3
6	I_2 I_3
7	I_1 I_3
8	I_2 I_1 I_3 I_5
9	I_2 I_1 I_3

Add 8 and 9 transaction

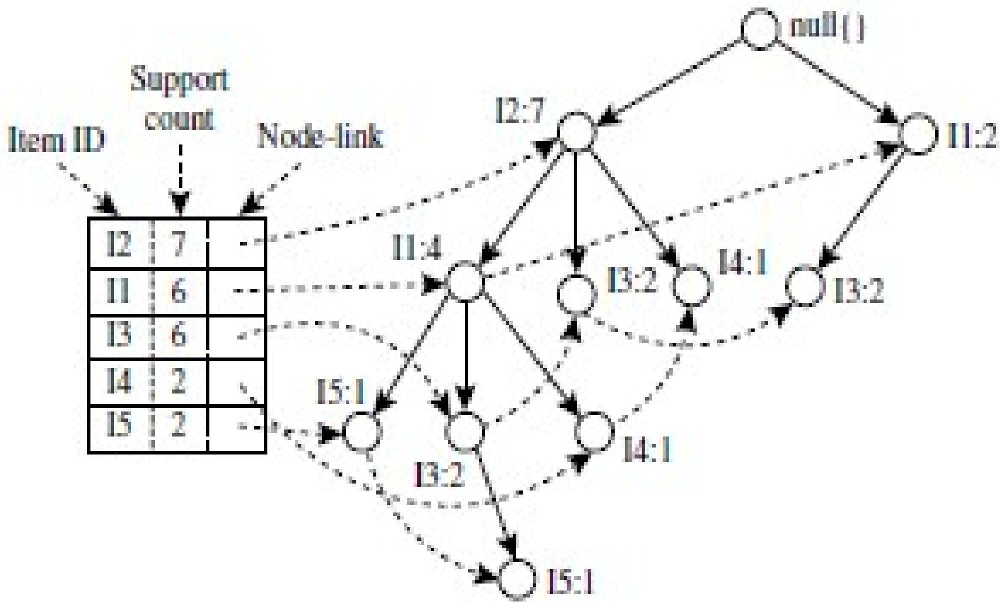
Final FP-Tree



1	I_2 I_1 I_5
2	I_2 I_4
3	I_2 I_3
4	I_2 I_1 I_4
5	I_1 I_3
6	I_2 I_3
7	I_1 I_3
8	I_2 I_1 I_3 I_5
9	I_2 I_1 I_3

Image Source: [2]

Mine FP pattern from Tree



Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

Image Source: [2]

Evaluation of Association Patterns

Well accepted criteria for evaluating the quality of patterns:

- **Objective interestingness measure:**
  - Uses statistics derived from data to find interestingness.
  - E.g. Support, confidence, lift, and others.
- **Subjective interestingness measure**:
  - Visualization (human user in loop).
  - Template based approach (it allows the user to restrict the type of patterns extracted by algorithm using user-specified template)
  - Use domain information such as concept hierarchy(lower level of the hierarchy are considered redundant→ eliminated

Criticism to Support and Confidence

- Example 1: (*Aggarwal & Yu, PODS98*)
  - Among 5000 students
    - 3000 play basketball
    - 3750 eat cereal
    - 2000 both play basket ball and eat cereal

	basketball	not basketball	sum(row)	
cereal	2000	1750	3750	75%
not cereal	1000	250	1250	25%
sum(col.)	3000	2000	5000	
	60%	40%		

play basketball ⇒ eat cereal [40%, 66.7%]

misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%.

play basketball ⇒ not eat cereal [20%, 33.3%]

is more accurate, although with lower support and confidence

Objective interestingness measure

- Lift (A→B) = confidence (A→B)/ support (B)

f_{1+} = support count of A

f_{+1} = support count of B

$$Lift(A \rightarrow B) = \frac{sup(A, B)}{sup(A) \cdot sup(B)} = \frac{P(B|A)}{P(B)}$$

- A and B negatively correlated, if the value is less than 1; otherwise A and B positively correlated

	B	$\overline{B}$	
A	f_{11}	f_{10}	f_{1+}
$\overline{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

Beverage preferences among a group of 1000 people.

	Coffee	$\overline{Coffee}$	
Tea	150	50	200
$\overline{Tea}$	650	150	800
	800	200	1000

1. If the rule had a lift of 1, it would imply that the probability of occurrence of the antecedent and that of the consequent are independent of each other. Hence , no rule can be drawn involving those two events.

2. **If the lift is > 1**, that lets us know the degree to which those two occurrences are dependent on one another, and makes those rules potentially useful for predicting the consequent in future data sets.

3. If the lift is < 1, that lets us know the items are substitute to each other. This means that presence of one item has negative effect on presence of other item and vice versa.

Image Source: [3,5]

Lift of a Rule

- Example 1 (cont)
  - *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] $LIFT = \dfrac{\frac{2000}{5000}}{\frac{3000}{5000} \times \frac{3750}{5000}} = 0.89$
  - *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] $LIFT = \dfrac{\frac{1000}{5000}}{\frac{3000}{5000} \times \frac{1250}{5000}} = 1.33$

	basketball	not basketball	sum(row)
cereal	2000	1750	3750
not cereal	1000	250	1250
sum(col.)	3000	2000	5000

Image Source: [5]

Problems With Lift

- Rules that hold 100% of the time may not have the highest possible lift. For example, if 5% of people are Vietnam veterans and 90% of the people are more than 5 years old, we get a lift of 0.05/(0.05*0.9)=1.11 which is only slightly above 1 for the rule
- Vietnam veterans -> more than 5 years old.

- And, lift is symmetric:
- *not eat cereal* $\Rightarrow$ *play basketball* [20%, 80%]

$$\text{LIFT} = \frac{\frac{1000}{5000}}{\frac{1250}{5000} \times \frac{3000}{5000}} = 1.33$$

Objective interestingness measure

- Conviction $= (f_{1+} f_{+0})/(N f_{10})$

	B	$\overline{B}$	
A	f_{11}	f_{10}	f_{1+}
$\overline{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

$$\text{conviction}(A \to C) = \frac{1 - \text{support}(C)}{1 - \text{confidence}(A \to C)}$$

It is the ratio of the expected frequency that X occurs without Y (that is to say, the frequency that the rule makes an incorrect prediction) if X and Y were independent divided by the observed frequency of incorrect predictions.

Beverage preferences among a group of 1000 people.

	$Coffee$	$\overline{Coffee}$	
Tea	150	50	200
$\overline{Tea}$	650	150	800
	800	200	1000

Image Source: [3]

$$Conv(A \to B) = \frac{sup(A)\cdot sup(\bar{B})}{sup(A,\bar{B})} = \frac{P(A)\cdot P(\bar{B})}{P(A,\bar{B})} = \frac{P(A)(1-P(B))}{P(A)-P(A,B)}$$

- Conviction is a measure of the implication and has value 1 if items are unrelated.

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%]
  - *eat cereal* $\Rightarrow$ *play basketball* conv:0.85

$$Conv = \frac{\frac{3000}{5000}\left(1-\frac{3750}{5000}\right)}{\frac{3000}{5000}-\frac{2000}{5000}} = 0.75$$

- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%]
  - *not eat cereal* $\Rightarrow$ *play basketball* conv:1.43

$$Conv = \frac{\frac{3000}{5000}\left(1-\frac{1250}{5000}\right)}{\frac{3000}{5000}-\frac{1000}{5000}} = 1.125$$

Summary of FP-Growth Algorithm

- Mining frequent patterns can be viewed as first mining 1-itemset and progressively growing each 1-itemset by mining on its conditional pattern base recursively

- Transform a frequent k-itemset mining problem into a sequence of k frequent 1-itemset mining problems via a set of conditional pattern bases

References

[1] Han, Jiawei, Jian Pei, and Yiwen Yin. "Mining frequent patterns without candidate generation." *ACM sigmod record* 29.2 (2000): 1-12.

[2] Han, Jiawei, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2012(Third Edition).

[3] Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. Introduction to data mining. Pearson Education India, 2016.

[4] .http://www.cs.sunysb.edu/~cse634/lecture_notes/07apriori.pdf

[5] https://paginas.fe.up.pt/~ec/files_0506/slides/04_AssociationRules.pdf