Contrast Data Mining: Methods and Applications

COMP90073
Security Analytics

Sarah Erfani, CIS

Semester 2, 2021

Outline

- Introduction to Contrast Data Mining
- Apriori
- FP-Growth
- Applications of contrast mining in network traffic analysis and anomaly detection

Contrast Data Mining - What is it? [1]

Contrast – "To compare or appraise in respect to differences" (Merriam Webster Dictionary)

Contrast data mining – The mining of patterns and models contrasting two or more datasets/conditions.

"*Sometimes it's good to contrast what you like with something else. It makes you appreciate it even more*"

Darby Conley, *Get Fuzzy, 2001*

What can be Contrasted?

- Objects at different *time* periods
  - “Compare traffic patterns from yesterday with today’s”
- Objects for different *spatial* locations
  - “Find the distinguishing features of location *x* for human DNA, versus location *x* for mouse DNA”
- Object positions in a *ranking*
  - “Find the differences between high- and low-income earners”
- Objects *across* different classes
  - “Find the differences between people with brown hair, versus those with blonde hair”
- Objects *within* a class
  - “Within the academic profession, there are no rich people”
  - “Within computer science, most scientific articles come from USA or Europe”

Characteristics of Contrast Data Mining

- Applied to multivariate data
- Objects may be relational, sequential, graphs, models, classifiers, combinations of these
- *Representation* of contrasts is important. Needs to be
  - *Interpretable, non redundant, potentially actionable*
  - *Tractable* to compute
- *Quality* of contrasts is also important. Need
  - *Statistical significance*, which can be measured in multiple ways
  - Ability to *rank* contrasts is desirable, especially for classification

How is Contrast Data Mining Used ? [1]

- Reporting significant changes/differences
  - "Young children with diabetes have a greater risk of hospital admission, compared to the rest of the population"
- Alerting, notification and monitoring
  - "Tell me when the dissimilarity index falls below 0.3"
- Building *one/multi-class classifiers*
  - Many different techniques
  - Also used for weighting and ranking instances
- Constructing *synthetic instances*
  - Good for rare classes

Example – Network Traffic Analysis

- Extracting knowledge from the massive volumes of network traffic is an important task in network and security management
- Network flows that are ranked by anomaly detection systems often contain thousands of records. Analysts often check only the first few pages
- Having a concise and meaningful report of network traffic is more desirable
- An appropriate report can help managers to reduce the time and cost of security analysis and make smart decisions

Example – Network Traffic Analysis

- **Summarization:** A good summarization is trade-off between two metrics: *Compaction gain* and *information loss*

Table1: Dataset of network flows

	src IP	sPort	des IP	dPort	pro	flags	packets	bytes
T1	12.190.84.122	32178	100.10.20.4	80	tcp	APRS	[2,20]	[504,1200]
T2	88.34.224.2	51989	100.10.20.4	80	tcp	APRS	[2,20]	[220,500]
T3	12.190.19.23	2234	100.10.20.4	80	tcp	APRS	[2,20]	[220,500]
T4	98.198.66.23	27643	100.10.20.4	80	tcp	APRS	[2,20]	[42,200]
T5	192.168.22.4	5002	100.10.20.3	21	tcp	A-RSF	[2,20]	[42,200]
T6	192.168.22.4	5001	100.10.20.3	21	tcp	A-RSF	[40,68]	[220,500]
T7	67.118.25.23	44532	100.10.20.3	21	tcp	A-RS	[40,68]	[42,200]
T8	192.168.22.4	2765	100.10.20.4	113	tcp	APRS	[2,20]	[504,1200]
T9	98.198.66.23	5003	100.10.20.5	21	tcp	A-RSF	[2,20]	[220,500]

THE UNIVERSITY OF MELBOURNE

Example – Network Traffic Analysis

- **Summarization:** A good summarization is trade-off between two metrics: *Compaction gain* and *information loss*

Table1: Dataset of network flows

	src IP	sPort	des IP	dPort	pro	flags	packets	bytes
T1	12.190.84.122	32178	100.10.20.4	80	tcp	APRS	[2,20]	[504,1200]
T2	88.34.224.2	51989	100.10.20.4	80	tcp	APRS	[2,20]	[220,500]
T3	12.190.19.23	2234	100.10.20.4	80	tcp	APRS	[2,20]	[220,500]
T4	98.198.66.23	27643	100.10.20.4	80	tcp	APRS	[2,20]	[42,200]
T5	192.168.22.4	5002	100.10.20.3	21	tcp	A-RSF	[2,20]	[42,200]
T6	192.168.22.4	5001	100.10.20.3	21	tcp	A-RSF	[40,68]	[220,500]
T7	67.118.25.23	44532	100.10.20.3	21	tcp	A-RS	[40,68]	[42,200]
T8	192.168.22.4	2765	100.10.20.4	113	tcp	APRS	[2,20]	[504,1200]
T9	98.198.66.23	5003	100.10.20.5	21	tcp	A-RSF	[2,20]	[220,500]

Table 2: Summarization by clustering

	size	src IP	sPort	des IP	dPort	pro	flags	packets	bytes
S1	5	***	***	100.10.20.4	***	tcp	APRS	[2,20]	***
S2	3	***	***	100.10.20.3	21	tcp	***	***	***

THE UNIVERSITY OF MELBOURNE

Example – Reporting Significant Differences Between Multiple Datasets

- **Day1:**

	src IP	sPort	des IP	dPort	pro	flags	packets	bytes
T1	12.190.84.122	32178	100.10.20.4	80	tcp	APRS	[2,20]	[504,1200]
T2	88.34.224.2	51989	100.10.20.4	80	tcp	APRS	[2,20]	[220,500]
T3	12.190.19.23	2234	100.10.20.4	80	tcp	APRS	[2,20]	[220,500]
T4	98.198.66.23	27643	100.10.20.4	80	tcp	APRS	[2,20]	[42,200]
T5	192.168.22.4	5002	100.10.20.3	21	tcp	A-RSF	[2,20]	[42,200]
T6	192.168.22.4	5001	100.10.20.3	21	tcp	A-RSF	[40,68]	[220,500]
T7	67.118.25.23	44532	100.10.20.3	21	tcp	A-RSF	[40,68]	[42,200]
T8	98.198.66.23	5003	100.10.20.5	21	tcp	A-RSF	[2,20]	[220,500]

- **Day2:**

Differences

	src IP	sPort	des IP	dPort	pro	flags	packets	bytes
T1	12.190.84.122	32178	100.10.20.4	80	tcp	APRS	[2,20]	[504,1200]
T2	88.34.224.2	51989	100.10.20.4	80	tcp	APRS	[2,20]	[220,500]
T3	12.190.19.23	2234	100.10.20.4	80	tcp	APRS	[2,20]	[220,500]
T4	98.198.66.23	27643	100.10.20.10	90	udp	---	[2,20]	[42,200]
T5	192.168.22.4	5002	100.10.20.10	90	udp	---	[2,20]	[42,200]
T6	192.168.22.4	5001	100.10.20.3	21	tcp	A-RSF	[40,68]	[220,500]
T7	67.118.25.23	44532	100.10.20.3	21	tcp	A-RSF	[40,68]	[42,200]
T8	98.198.99.23	5003	100.10.20.20	21	tcp	APRS	[40,68]	[1200,1500]

Example – Reporting Significant Differences Between Multiple Datasets

- Output:

	src IP	sPort	des IP	dPort	pro	flags	packets	bytes
C1	98.198.66.23	27643	100.10.20.10	90	udp	---	[2,20]	[42,200]
C2	192.168.22.4	5002	100.10.20.10	90	udp	---	[2,20]	[42,200]
C3	98.198.99.23	5003	100.10.20.20	21	tcp	APRS	[40,68]	[1200,1500]

- We can use **contrast pattern mining** for finding important changes.

- **Contrast pattern mining** finds patterns whose *support* differs significantly from one dataset to another.

THE UNIVERSITY OF MELBOURNE

Definitions

- **Itemset:** A collection of one or more items
  - **k-itemset:** An itemset that contains k items
- **Count (X, D):** The number of transactions in dataset D containing pattern X

- **Support (X, D):** The percentage of transactions in dataset D containing pattern X

$$support\ (X, D) = \frac{count(X, D)}{|D|}$$

- **Frequent Itemset:** An itemset whose support is greater than or equal to a *minsup* threshold

$$support\ (X, D) \geq minsup$$

Apriori Algorithm – Revision

Method:

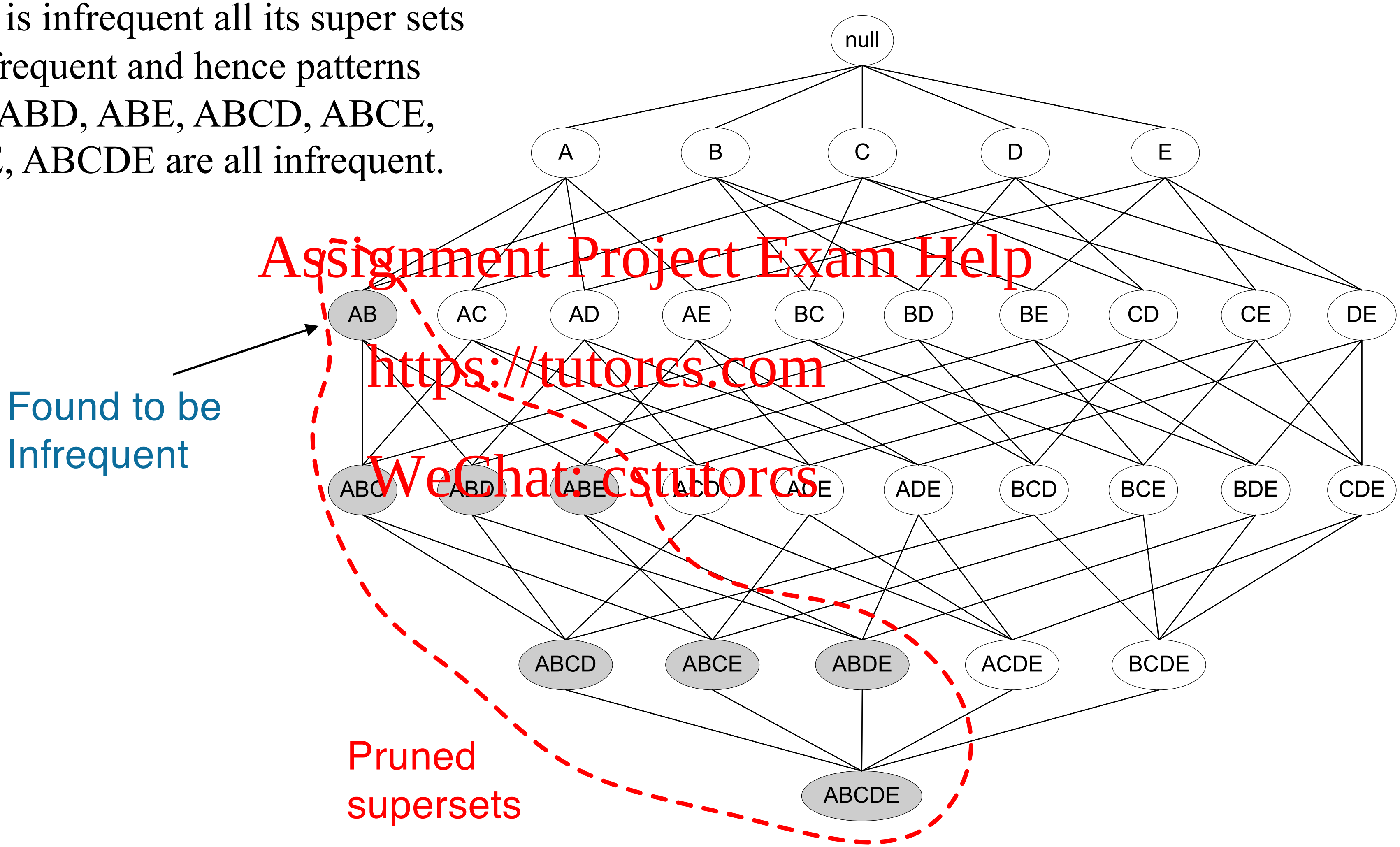
- Let k=1
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are identified
  - Prune candidate itemsets containing subsets of length k that are infrequent
  - Count the support of each candidate by scanning the database
  - Eliminate candidates that are infrequent, leaving only those that are frequent
  - Generate length (k+1) candidate itemsets from length k frequent itemsets

Apriori Principle:

- If an itemset is infrequent, then all of its superset must also be infrequent

Illustrating Apriori Principle – Revision

If AB is infrequent all its super sets are infrequent and hence patterns ABC, ABD, ABE, ABCD, ABCE, ABDE, ABCDE are all infrequent.



Bottleneck of Apriori

- It is costly to handle a huge number of candidate sets
- If there are 10^4 frequent *1-itemsts*, the Apriori algorithm will need to generate more than 10^7 *2-itemsets* and test their frequencies.
- Mining long patterns needs many passes of scanning and generates lots of candidates.
- It may need to repeatedly scan the whole database and check a large set of candidates by pattern matching.
- Bottleneck: **candidate-generation-and-test**
- Can we avoid **candidate generation**?

Overview of FP-Growth [2]

- Find frequent single items, and partition the database based on each such item
- Recursively grow frequent patterns by doing the above for each portioned databased

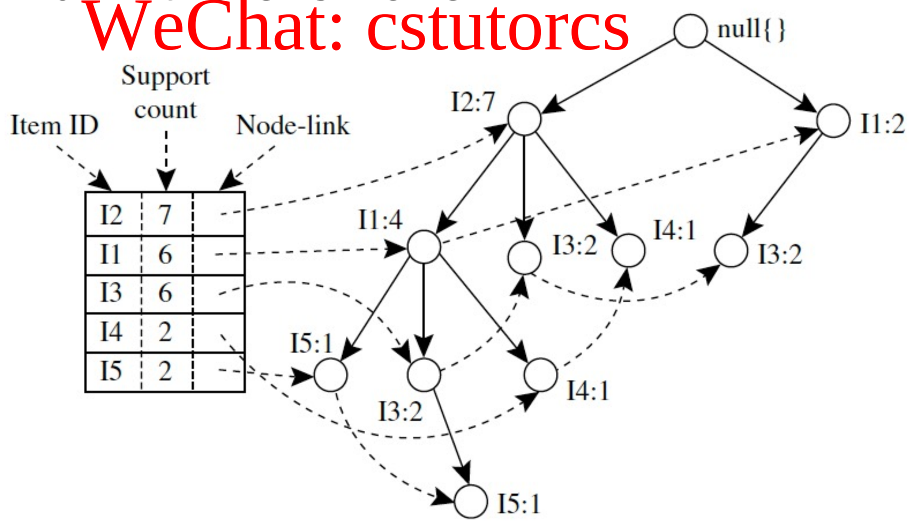
- To facilitate efficient processing, compress a large database into a compact, *Frequent-Pattern tree* (FP-tree) structure

  – Highly compacted, but complete for frequent pattern mining

  – Avoid candidate generation
  – Avoid costly repeated database scans

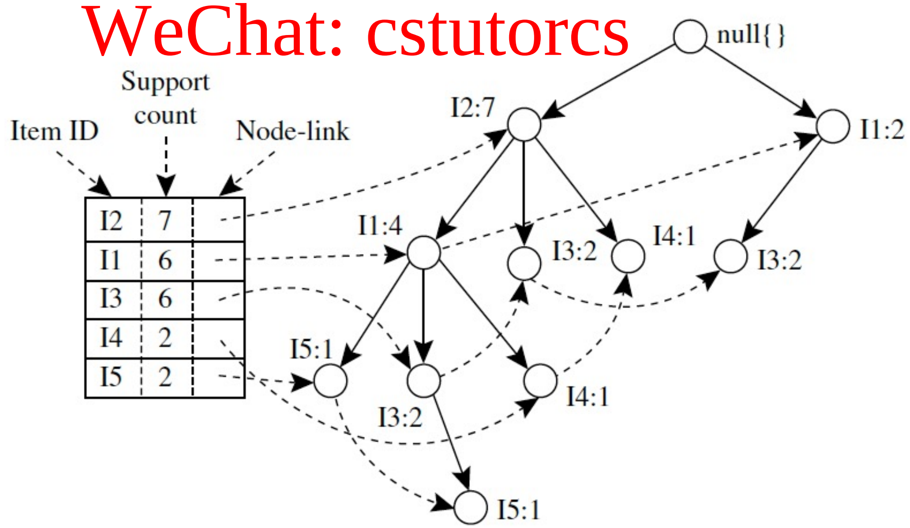
FP-Tree Definition

- FP-tree is *a frequent pattern tree*, and defined as below:
- One root labeled as “null", a set of *item prefix sub-trees* as the children of the root, and a *frequent-item header table*.
- Each node in *the item prefix sub-trees* has three fields:
  - Item-name: Registers which item this node represents,
  - Count: The number of transactions represented by the portion of the path reaching this node,
  - Node-link: Links to the next node in the FP-tree carrying the same item-name, or null if there is none.



FP-Tree Definition

- Each entry in the *frequent-item header table* has two fields,
  - Item-name,
  - Item support count, and
  - Head of node-link: Points to the first node in the FP-tree carrying the item-name.



FP-tree: Construction and Design

STEP 1: Scan the transaction database for the first time, find frequent items (single item patterns) and order them into a list in frequency descending order. In the format of (item-name, support).

TID	List of item _IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3, I6
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

FP-tree: Construction and Design

STEP 1: Scan the transaction database for the first time, find frequent items (single item patterns) and order them into a list in frequency descending order. In the format of (item-name, support).

TID	List of item _IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3, I6
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Itemset	Count
{I1}	
{I2}	
{I3}	
{I4}	
{I5}	
{I6}	

FP-tree: Construction and Design

STEP 1: Scan the transaction database for the first time, find frequent items (single item patterns) and order them into a list in frequency descending order. In the format of (item-name, support).

TID	List of item _IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3, I6
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

L

Itemset	Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2
{I6}	1

FP-tree: Construction and Design

STEP 1: Scan the transaction database for the first time, find frequent items (single item patterns) and order them into a list in frequency descending order. In the format of (item-name, support).

TID	List of item _IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3, I6
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

L

Itemset	Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2
{I6}	1

Minsup= 2 →

L

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

FP-tree: Construction and Design

STEP 1: Scan the transaction database for the first time, find frequent items (single item patterns) and order them into a list in frequency descending order. In the format of (item-name, support).

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

TID	List of Item _IDs	Ordered Frequent Items
T100	I1, I2, I5	I2, I1, I5
T200	I2, I4	I2, I4
T300	I2, I3	I2, I3
T400	I1, I2, I4	I2, I1, I4
T500	I1, I3	I1, I3
T600	I2, I3, I6	I2, I3
T700	I1, I3	I1, I3
T800	I1, I2, I3, I5	I2, I1, I3, I5
T900	I1, I2, I3	I2, I1, I3

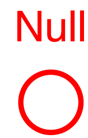
L

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

FP-tree: Construction and Design

- **STEP 2: Construct FP-tree**

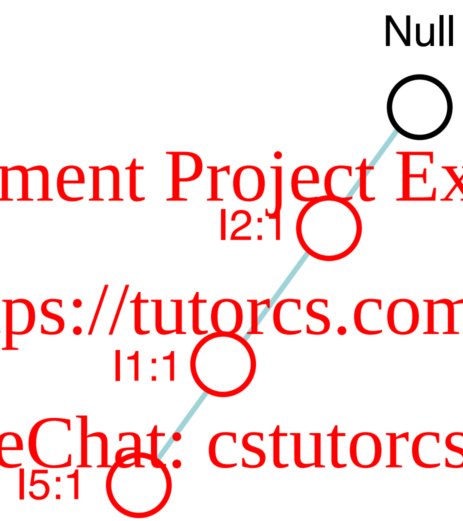
Ordered Frequent Items
I2, I1, I5
I2, I4
I2, I3
I2, I1, I4
I1, I3
I2, I3
I1, I3
I2, I1, I3, I5
I2, I1, I3



FP-tree: Construction and Design

- **STEP 2: Construct FP-tree**

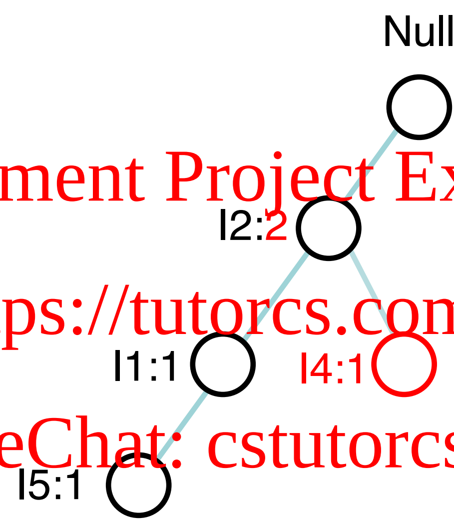
Ordered Frequent Items
I2, I1, I5
I2, I4
I2, I3
I2, I1, I4
I1, I3
I2, I3
I1, I3
I2, I1, I3, I5
I2, I1, I3



FP-tree: Construction and Design

- **STEP 2: Construct FP-tree**

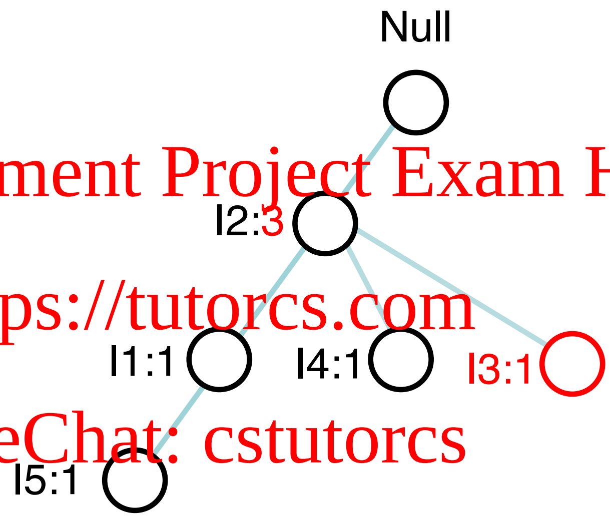
Ordered Frequent Items
I2, I1, I5
I2, I4
I2, I3
I2, I1, I4
I1, I3
I2, I3
I1, I3
I2, I1, I3, I5
I2, I1, I3



FP-tree: Construction and Design

- **STEP 2: Construct FP-tree**

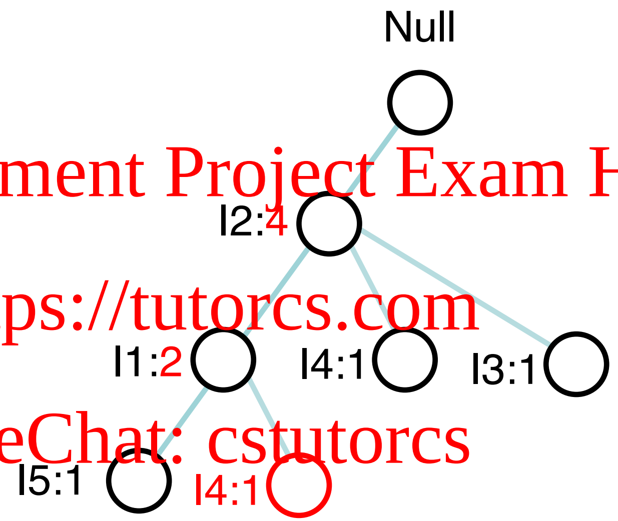
Ordered Frequent Items
I2, I1, I5
I2, I4
I2, I3
I2, I1, I4
I1, I3
I2, I3
I1, I3
I2, I1, I3, I5
I2, I1, I3



FP-tree: Construction and Design

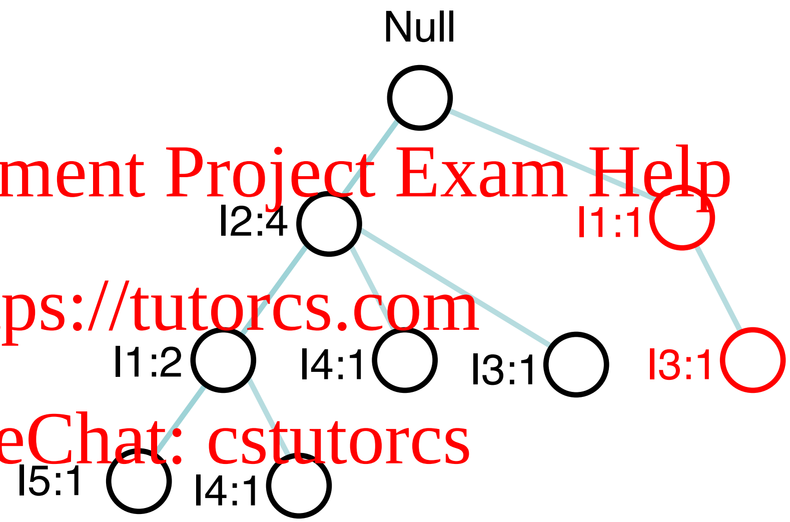
- **STEP 2: Construct FP-tree**

Ordered Frequent Items
I2, I1, I5
I2, I4
I2, I3
I2, I1, I4
I1, I3
I2, I3
I1, I3
I2, I1, I3, I5
I2, I1, I3



- **STEP 2: Construct FP-tree**

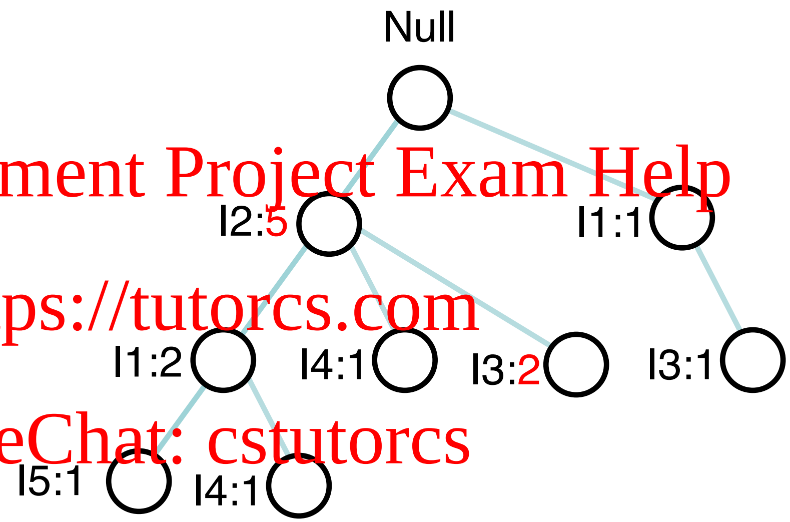
Ordered Frequent Items
I2, I1, I5
I2, I4
I2, I3
I2, I1, I4
I1, I3
I2, I3
I1, I3
I2, I1, I3, I5
I2, I1, I3



FP-tree: Construction and Design

- **STEP 2: Construct FP-tree**

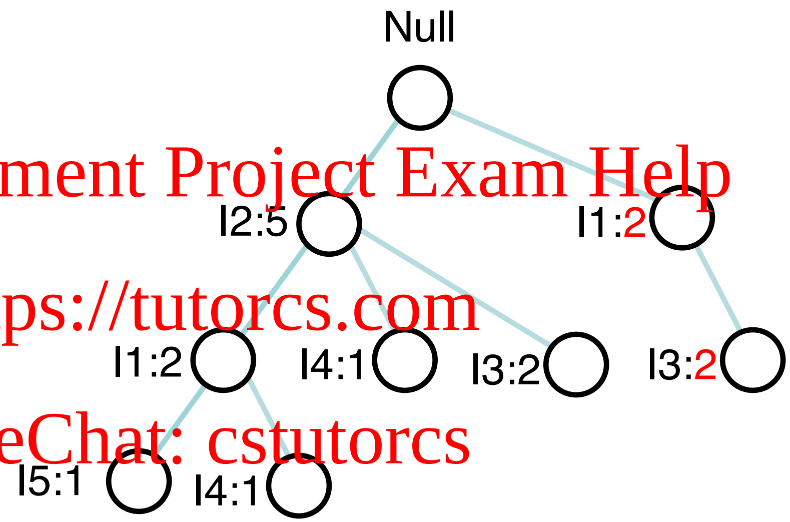
Ordered Frequent Items
I2, I1, I5
I2, I4
I2, I3
I2, I1, I4
I1, I3
I2, I3
I1, I3
I2, I1, I3, I5
I2, I1, I3



FP-tree: Construction and Design

- **STEP 2: Construct FP-tree**

Ordered Frequent Items
I2, I1, I5
I2, I4
I2, I3
I2, I1, I4
I1, I3
I2, I3
I1, I3
I2, I1, I3, I5
I2, I1, I3

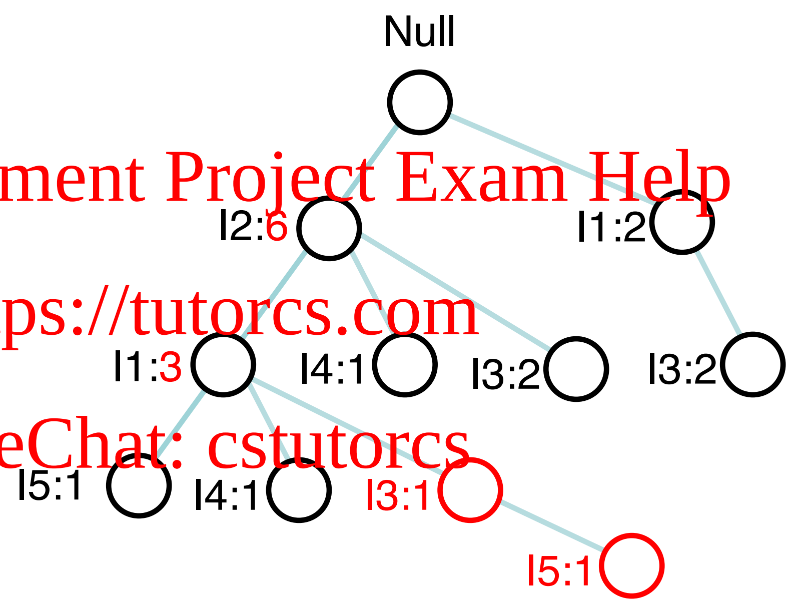


THE UNIVERSITY OF MELBOURNE

FP-tree: Construction and Design

- **STEP 2: Construct FP-tree**

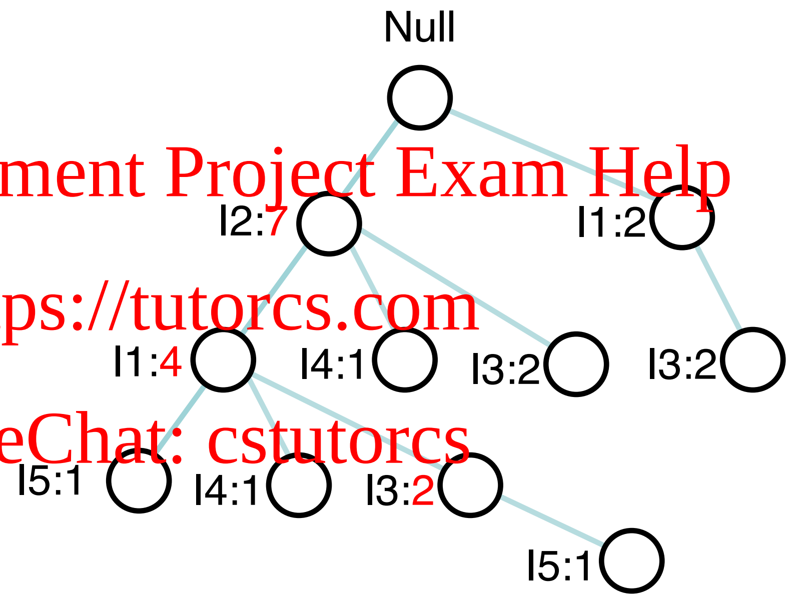
Ordered Frequent Items
I2, I1, I5
I2, I4
I2, I3
I2, I1, I4
I1, I3
I2, I3
I1, I3
I2, I1, I3, I5
I2, I1, I3



FP-tree: Construction and Design

- **STEP 2: Construct FP-tree**

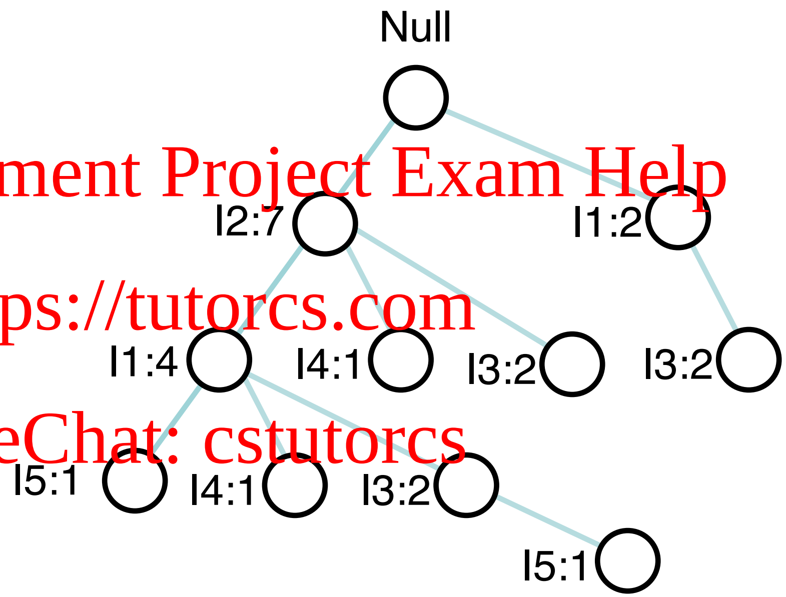
Ordered Frequent Items
I2, I1, I5
I2, I4
I2, I3
I2, I1, I4
I1, I3
I2, I3
I1, I3
I2, I1, I3, I5
I2, I1, I3



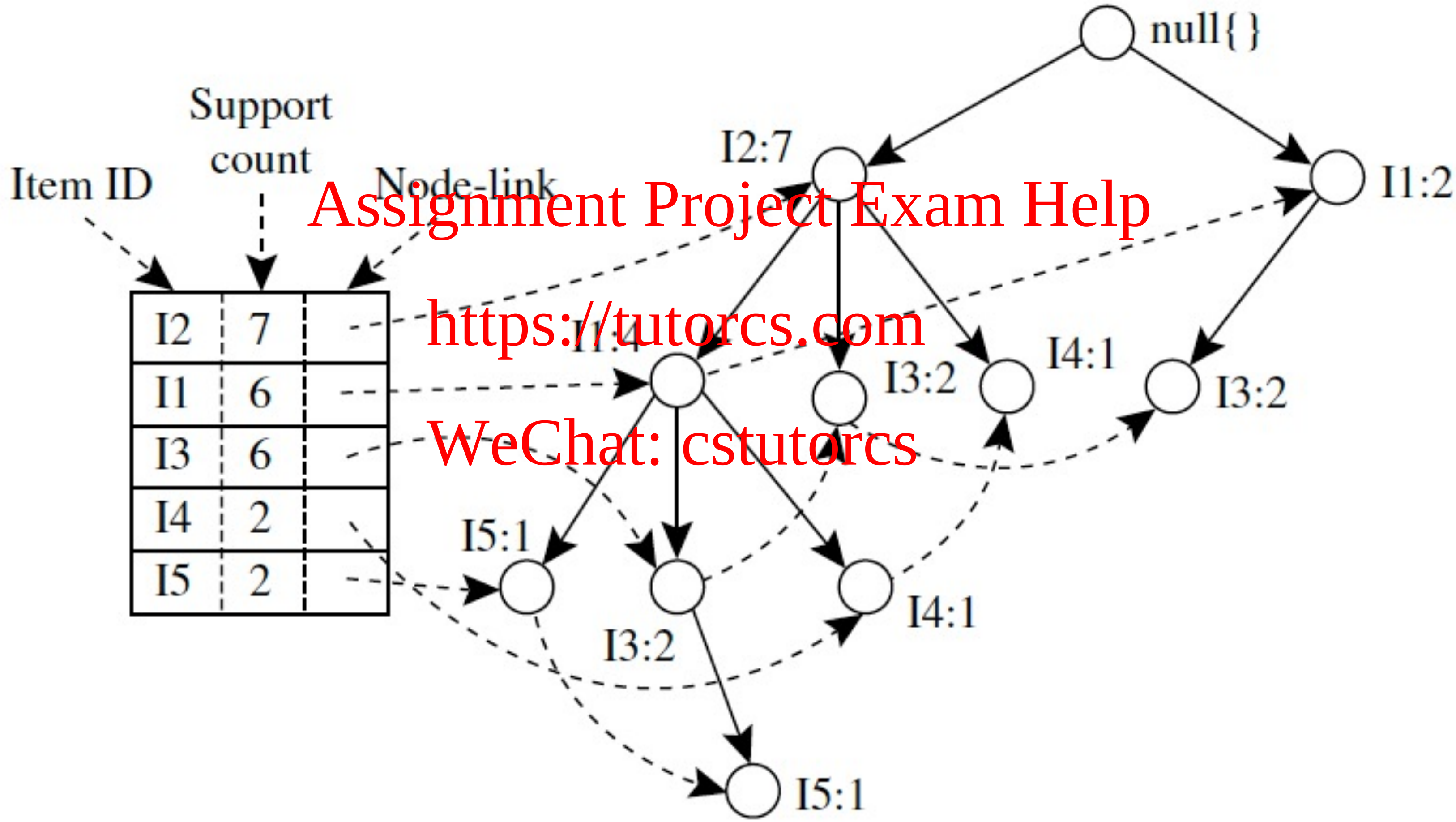
FP-tree: Construction and Design

- **STEP 2: Construct FP-tree**

Ordered Frequent Items
I2, I1, I5
I2, I4
I2, I3
I2, I1, I4
I1, I3
I2, I3
I1, I3
I2, I1, I3, I5
I2, I1, I3



FP-tree: Construction and Design



Mining Frequent Patterns Using FP-tree

Starting the processing from the end of list L:

Step 1:

Construct **conditional pattern base** for each item in L

Step 2:

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

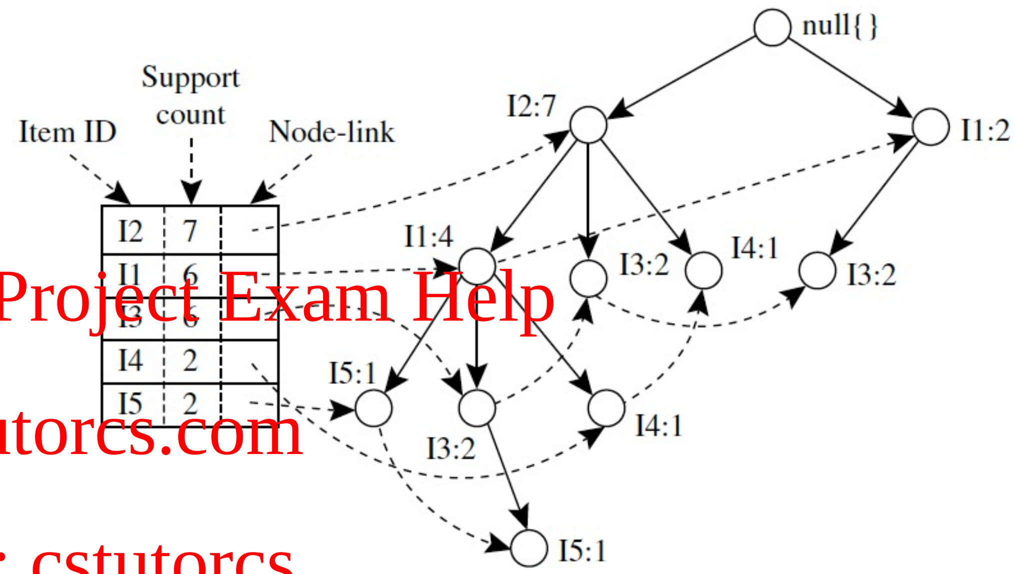
Step 3:

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

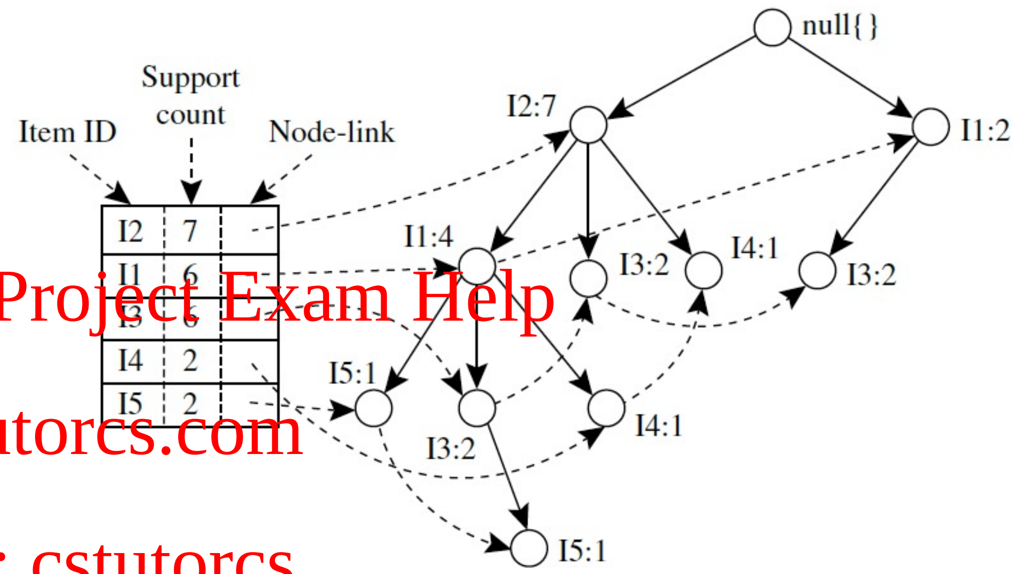
- Starting at the bottom of frequent-item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of **transformed prefix paths** of that item to form a **conditional pattern base**



Item	Conditional Pattern Base
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}

Step 1: Construct Conditional Pattern Base

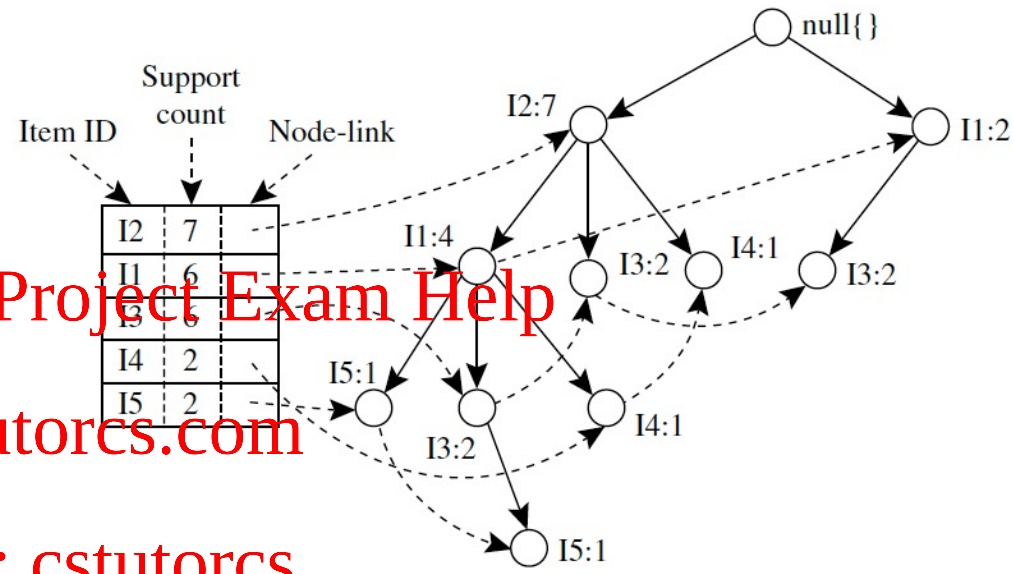
- Starting at the bottom of frequent-item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of **transformed prefix paths** of that item to form a **conditional pattern base**



Item	*Conditional Pattern Base*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}
I4	{{I2, I1: 1}, {I2: 1}}

Step 1: Construct Conditional Pattern Base

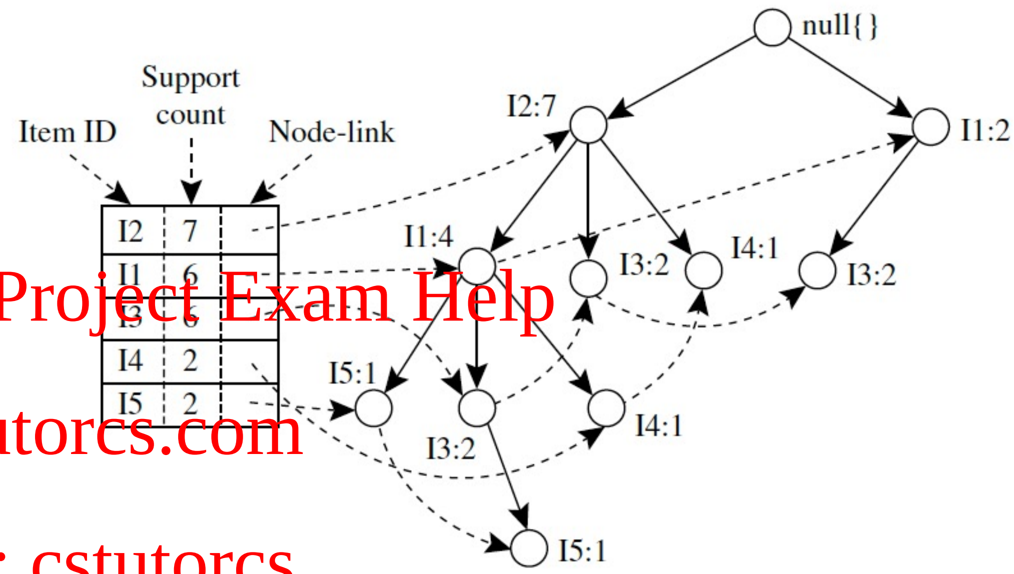
- Starting at the bottom of frequent-item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of **transformed prefix paths** of that item to form a **conditional pattern base**



Item	*Conditional Pattern Base*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}
I4	{{I2, I1: 1}, {I2: 1}}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}

Step 1: Construct Conditional Pattern Base

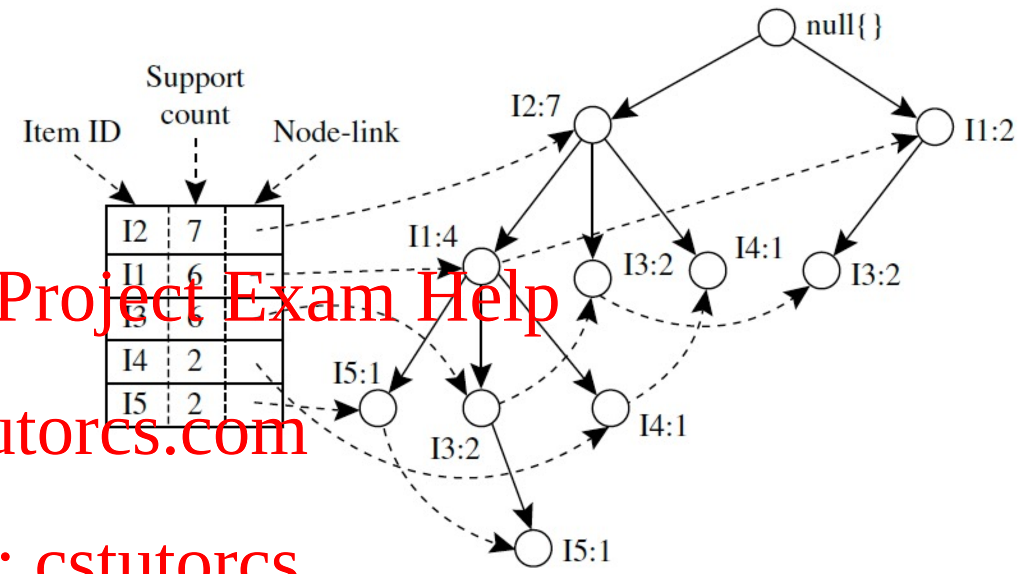
- Starting at the bottom of frequent-item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of **transformed prefix paths** of that item to form a **conditional pattern base**



Item	*Conditional Pattern Base*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}
I4	{{I2, I1: 1}, {I2: 1}}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}
I1	{{I2: 4}}

Step 2: Construct Conditional FP-tree

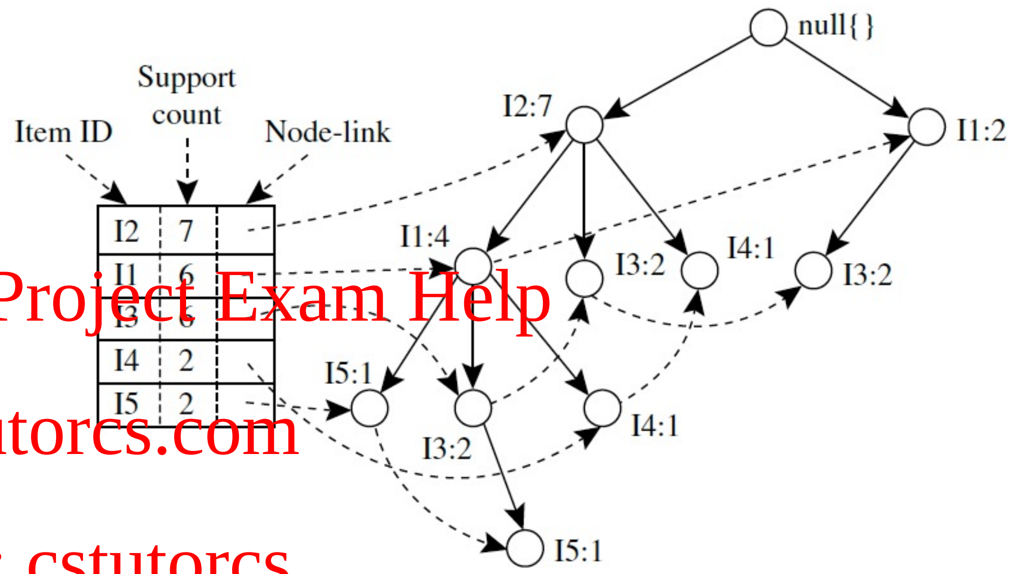
- For each pattern base
  - Accumulate the count for each item in the base
  - Construct the conditional FP-tree for the frequent items of the pattern base
- Minsup=2



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

Step 2: Construct Conditional FP-tree

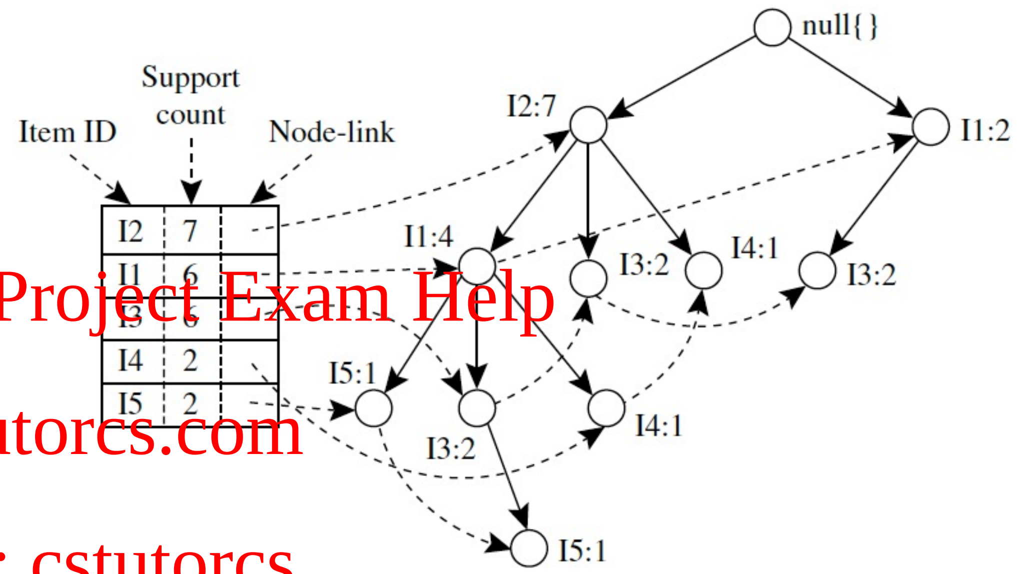
- For each pattern base
  - Accumulate the count for each item in the base
  - Construct the conditional FP-tree for the frequent items of the pattern base
- Minsup=2



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

Step 2: Construct Conditional FP-tree

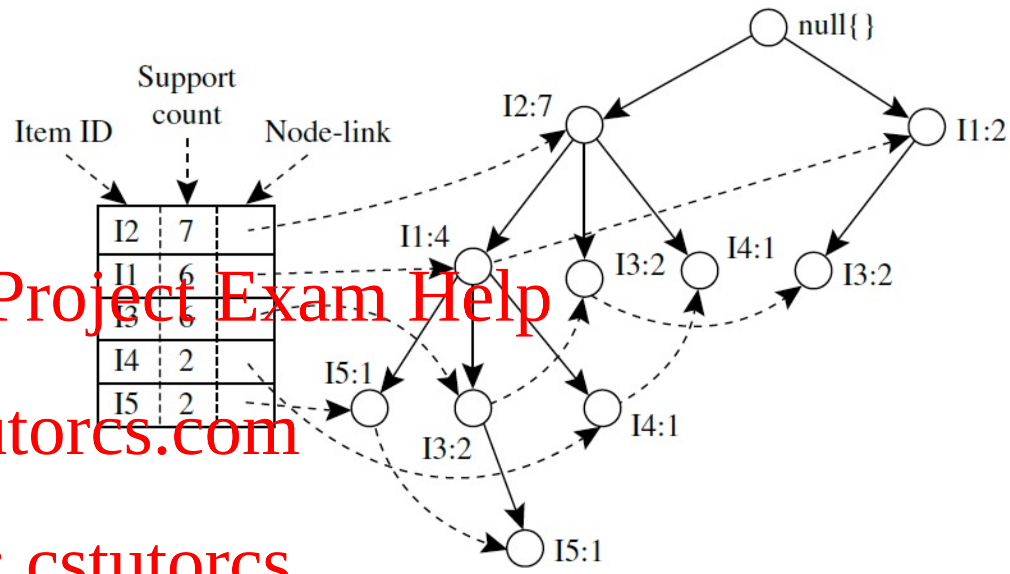
- For each pattern base
  - Accumulate the count for each item in the base
  - Construct the conditional FP-tree for the frequent items of the pattern base
- Minsup=2



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

Step 2: Construct Conditional FP-tree

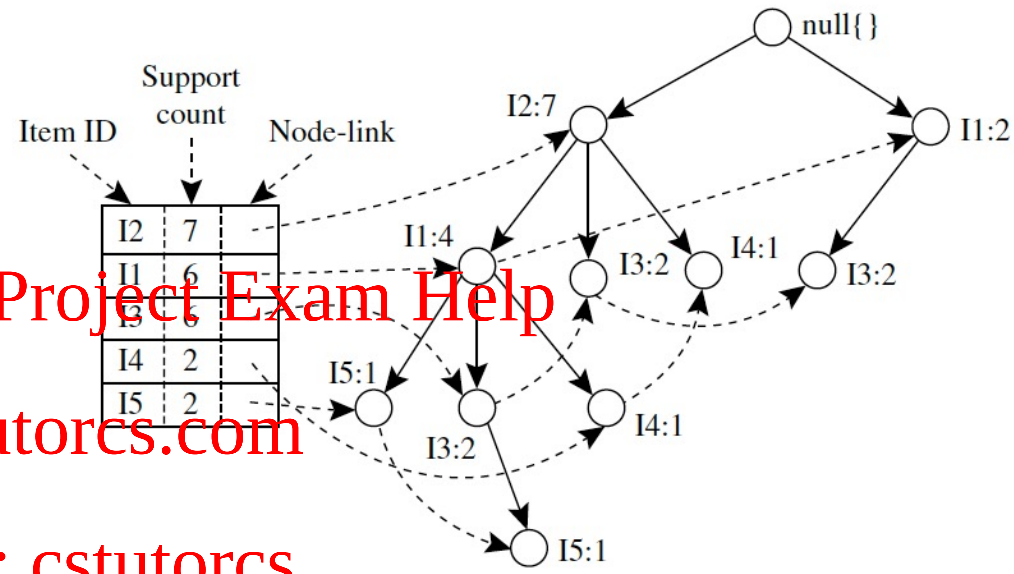
- For each pattern base
  - Accumulate the count for each item in the base
  - Construct the conditional FP-tree for the frequent items of the pattern base
- Minsup=2



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

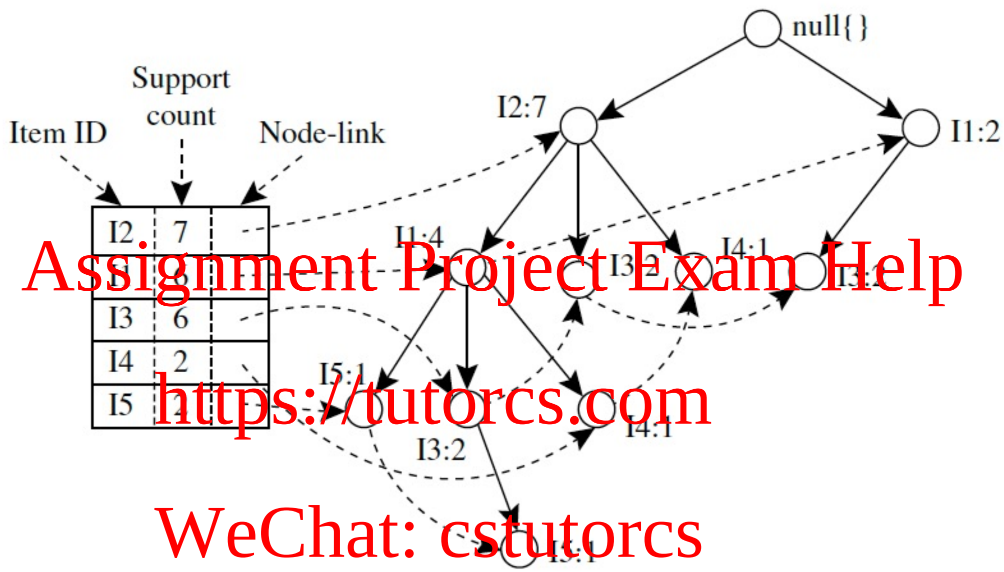
Step 2: Construct Conditional FP-tree

- 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
- Minsup=2



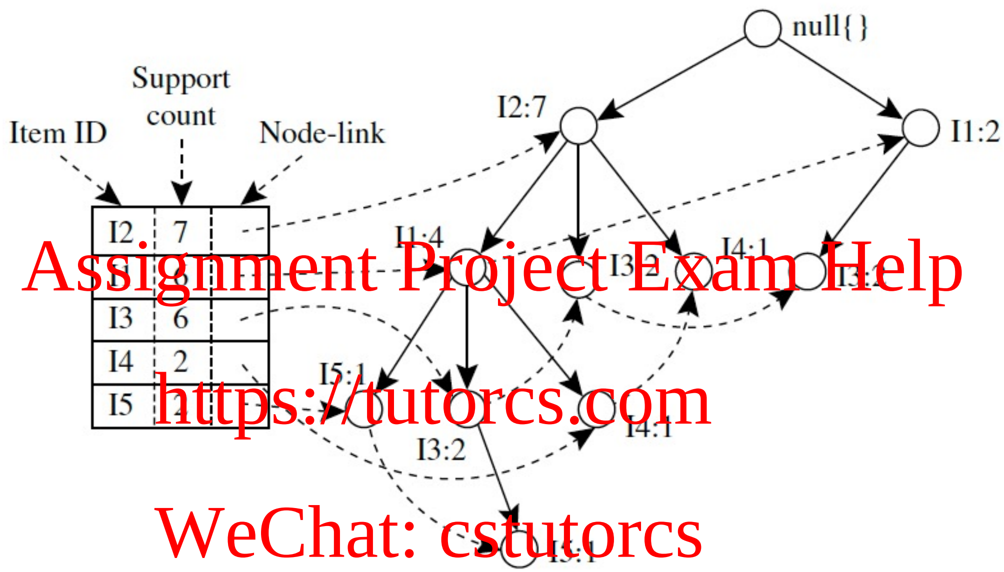
Item	*Conditional Pattern Base*	*Conditional FP-tree*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩
I1	{{I2: 4}}	⟨I2: 4⟩

Step 3: Recursively Mine the Conditional FP-tree



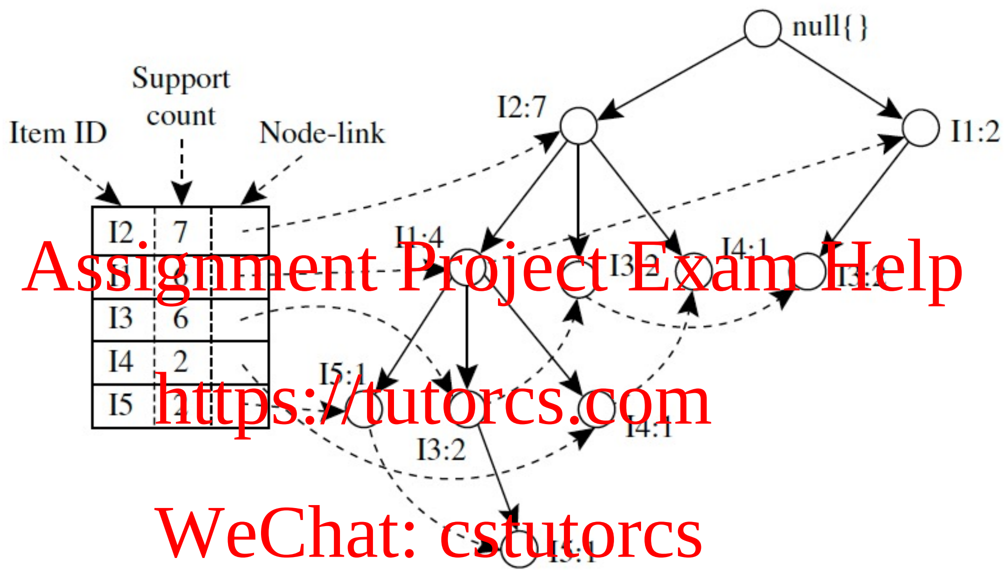
Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	
I1	{{I2: 4}}	⟨I2: 4⟩	

Step 3: Recursively Mine the Conditional FP-tree



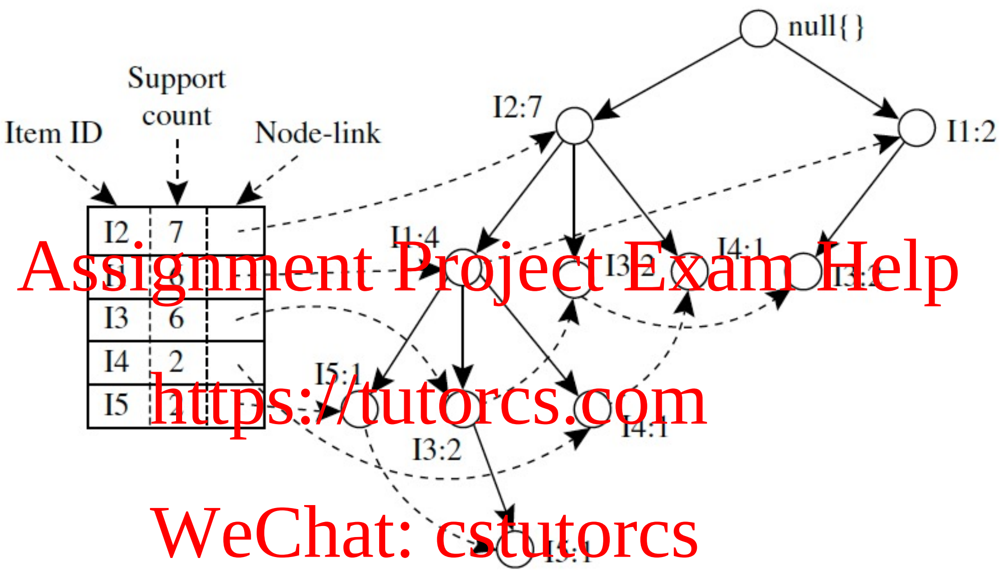
Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	
I1	{{I2: 4}}	⟨I2: 4⟩	

Step 3: Recursively Mine the Conditional FP-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⟩	

Step 3: Recursively Mine the Conditional FP-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}

Properties of the FP-tree Structure

Advantages

- Only needs to read the file twice, as opposed to Apriori who reads it once for every iteration.

- Removes the need to calculate the pairs to be counted, which is very processing heavy, because it uses the FP-Tree. This makes it O(n) (which is much faster than Apriori).

- Stores a compact version of the database in memory.

Bottlenecks

- The interdependency problem is that for the parallelization of the algorithm some that still needs to be shared, which creates a bottleneck in the shared memory.

Definitions

- **Growth Rate:** Given a pair of dataset D_p (positive/target dataset) and D_n (negative/source dataset):

$$gr(X, D_p) = \frac{supp(X, D_p)}{supp(X, D_n)}$$

- **Emerging Patterns (EPs):** Patterns whose support is significantly different from one dataset to another. If $gr(X, D_p) \geq \rho$, pattern X is an emerging pattern for dataset D_p.
  - *Emerging patterns also known as Contrast patterns (CP), and Discriminative patterns.*

- **Jumping Emerging Pattern (JEP):** An emerging pattern whose support is non-zero in the positive dataset but zero in the negative dataset is called a, and $gr(X, D_p) = \infty$.

Example – Emerging and Jumping Emerging Patterns

Positive Dataset

	Src IP	des IP	pro	packets
T1	192.168.22.1	10.10.10.1	udp	[2,20]
T2	192.168.55.2	10.10.10.4	udp	[40,68]
T3	192.168.22.1	10.10.10.1	tcp	[2,20]
T4	192.168.20.1	10.10.10.2	tcp	[2,20]

Negative Dataset

	Src IP	des IP	pro	packets
T1	192.168.44.2	10.10.10.2	tcp	[40,68]
T2	192.168.20.1	10.10.10.2	tcp	[2,20]
T3	192.168.20.1	10.10.10.2	tcp	[2,20]
T4	192.168.22.1	10.10.10.1	udp	[2,20]

- Find an EP and JEP given $\rho = 1$

THE UNIVERSITY OF MELBOURNE

Example – Emerging and Jumping Emerging Patterns

Positive Dataset

	Src IP	des IP	pro	packets
T1	192.168.22.1	10.10.10.1	udp	[2,20]
T2	192.168.55.2	10.10.10.4	udp	[40,68]
T3	192.168.22.1	10.10.10.1	tcp	[2,20]
T4	192.168.20.1	10.10.10.2	tcp	[2,20]

Negative Dataset

	Src IP	des IP	pro	packets
T1	192.168.44.2	10.10.10.2	tcp	[40,68]
T2	192.168.20.1	10.10.10.2	tcp	[2,20]
T3	192.168.20.1	10.10.10.2	tcp	[2,20]
T4	192.168.22.1	10.10.10.1	udp	[2,20]

	Emerging and Jumping Emerging Patterns	Growth rate
C1	{srcIP=192.168.22.1, destIP=10.10.10.1, Pkt=[2,20]}	2
C2	{srcIP=192.168.55.2, destIP=10.10.10.4, pro=udp, Pkt=[2,20]}	∞
⋮		

Case Study – Goodness of Contrast Data Mining for Network Traffic Analysis [2]

- **Objective:** Provide a concise and meaningful report of significant changes in multiple datasets.
  - Evaluate the quality of generated patterns.
  - Select the best set of patterns, emerging patterns belong to either an attack class or a normal class.
  - Emerging patterns can efficiently distinguish between attack and normal traffic.

- **Extracting contrast patterns:**
  - GC-Growth algorithm

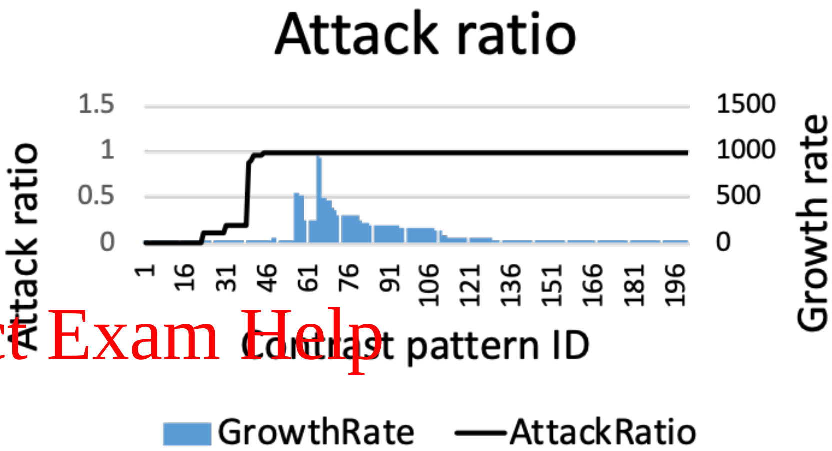
THE UNIVERSITY OF MELBOURNE

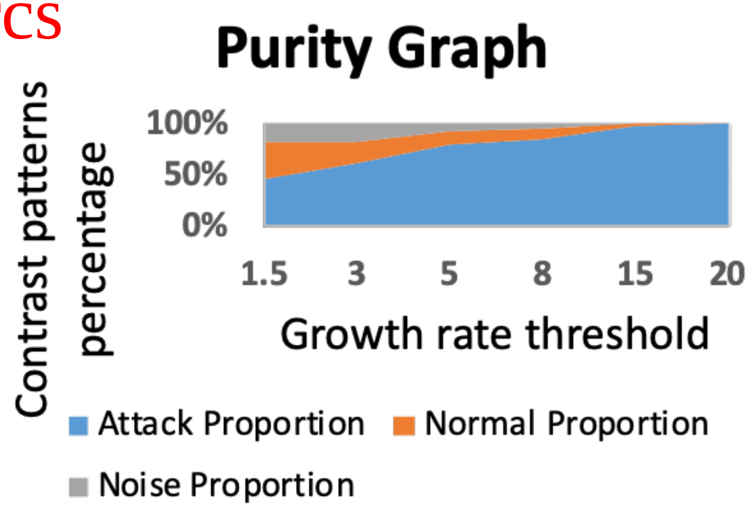
Case Study – Goodness of Contrast Data Mining for Network Traffic Analysis

- Attack ratio: Measures the probability that a given contrast pattern X belongs to the attack class

$$Attack\ Ratio = \frac{Count(X, D_{pos}(att))}{Count(X, D_{pos})}$$

  - All contrast patterns with a high growth rate are attack patterns
  - Most of the pure patterns (i.e., patterns belong to either an attack class or a normal class) correspond to attacks
  - The proportion of attack patterns increases significantly with an increase of the growth rate threshold





Application: One-class Classification [1]

- **OCLEP:** Build some CP length statistics
  - Uses the training data to derive the length statistics
  - For each new test case, compare the length statistics for the test case and the length statistics of the training data.

- **Property:** Provided that all transactions of T come from one class, length statistic S tends to contain long EPs when test X’ and train X come from the same class, and it tends to contain short EPs when test and train come from different classes.

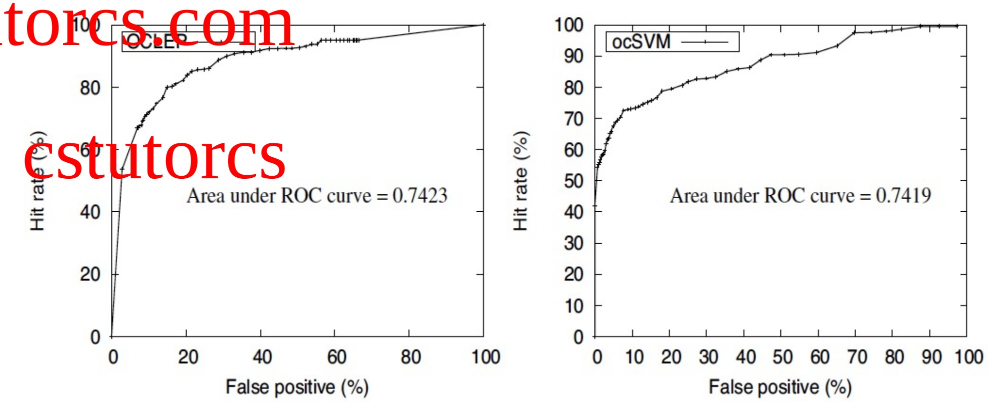
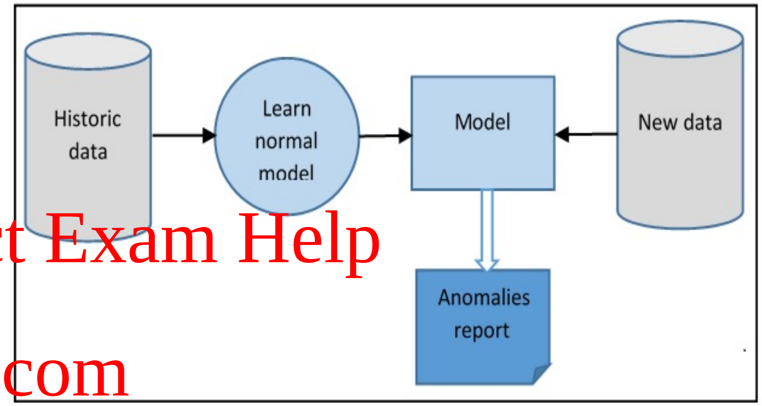


Figure: ROC curves for OCLEP and OCSVM.

Application: Comparing Anomaly Detection Models

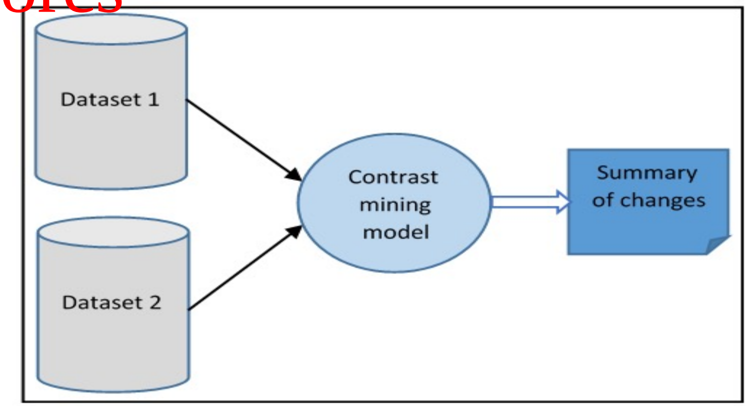
- **Anomaly detection model:**
  - Learns a model of normal behaviours from historic dataset
  - Applies the learned model to the current data
  - Detects anomalies

Anomaly detection model



- **CPM technique:**
  - Compares two current dataset and historic dataset
  - Extracts significant changes
  - Presents a succinct report

Contrast mining model



THE UNIVERSITY OF MELBOURNE

Summary

- Why contrast data mining is important and when it can be used?
- What algorithms can be used for contrast data mining?
- How it can be used for network traffic analysis and unsupervised learning?

Next: Adversarial Machine Learning

References

1. Guozhu Dong and James Bailey. “Contrast data mining: concepts, algorithms, and applications”. CRC Press, 2012.

2. Jiawei Han, Micheline Kamber, Jian Pei, “Data Mining: Concepts and Techniques ”, 2011, Chapter 6.2.4.

3. Elaheh Alipour Chavary, Sarah Erfani, Christopher Leckie, “Summarizing Significant Changes in Network Traffic Using Contrast Pattern Mining”, ACM International Conference on Information and Knowledge Management (CIKM), 2017.