Association rules

Part 2

Mohammed Brahimi & Sami Belkacem

Association rule mining task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ minsup threshold
  - confidence ≥ minconf threshold
- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the minsup and minconf thresholds

Computationally not feasible!

Agenda

1. Frequent Itemsets, Association Rules
2. Apriori Algorithm
3. FP-Growth Algorithm
4. Evaluation of Association Patterns

Agenda

1. **Frequent Itemsets, Association Rules**
2. **Apriori Algorithm**
3. FP-Growth Algorithm
4. Evaluation of Association Patterns

The Apriori Principle

If an itemset is frequent, then all of its subsets must also be frequent.

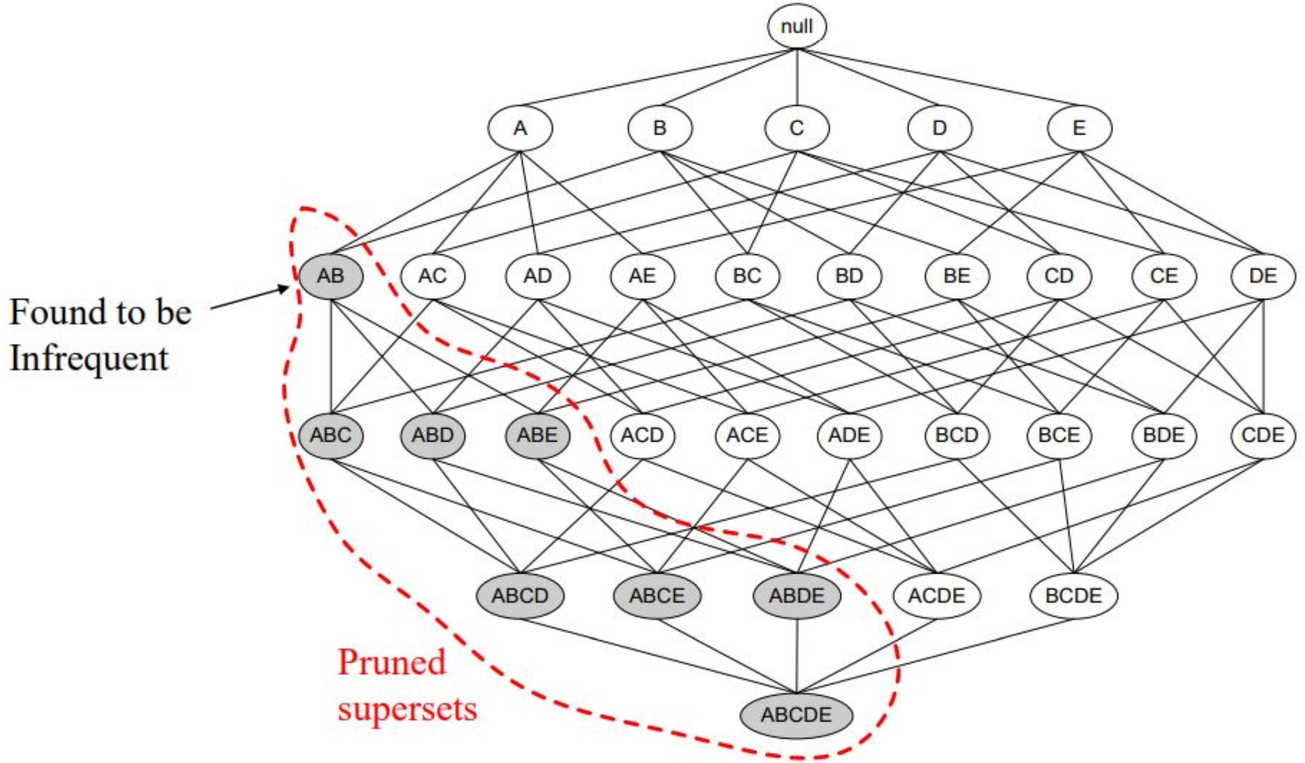
Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- Support of an itemset never exceeds the support of its subsets
- This is known as the **anti-monotone** property of support

The Apriori Principle

If an itemset is infrequent, then all of its supersets must be infrequent too.



Illustrating Apriori Principle

TID	*Items*
1	Bread, Milk
2	Juice, Bread, Diaper, Eggs
3	Juice, Coke, Diaper, Milk
4	Juice, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Juice	**3**
Diaper	**4**
Eggs	**1**

Minimum Support = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$

Illustrating Apriori Principle

TID	*Items*
1	**Bread, Milk**
2	**Juice, Bread, Diaper, Eggs**
3	**Juice, Coke, Diaper, Milk**
4	**Juice, Bread, Diaper, Milk**
5	**Bread, Coke, Diaper, Milk**

$\rightarrow$

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Juice	**3**
Diaper	**4**
Eggs	**1**

Minimum Support = 3

If every subset is considered,

$${}^{6}C_{1} + {}^{6}C_{2} + {}^{6}C_{3}$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$

Illustrating Apriori Principle

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Juice	**3**
Diaper	**4**
Eggs	**1**

Items (2-itemsets)

Itemset
{Bread,Milk}
{Juice, Bread}
{Bread,Diaper}
{Juice,Milk}
{Diaper,Milk}
{Juice,Diaper}

Pairs (2-itemsets)
(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Generate 2-itemset candidates

If every subset is considered,

$^6C_1 + {}^6C_2 + {}^6C_3$

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 4 = 16$

Illustrating Apriori Principle

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Juice	**3**
Diaper	**4**
Eggs	**1**

Items (2-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Juice, Bread}	**2**
{Bread,Diaper}	**3**
{Juice,Milk}	**2**
{Diaper,Milk}	**3**
{Juice,Diaper}	**3**

Pairs (2-itemsets)
(No need to generate candidates involving Coke or Eggs)

Eliminate infrequent 2-itemset candidates

Minimum Support = 3

If every subset is considered,

$^6C_1 + {}^6C_2 + {}^6C_3$

$6 + 15 + 20 = 41$

With support-based pruning,

$6 + 6 + 4 = 16$

Illustrating Apriori Principle

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Juice	**3**
Diaper	**4**
Eggs	**1**

Items (2-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Juice, Bread}	**2**
{Bread,Diaper}	**3**
{Juice,Milk}	**2**
{Diaper,Milk}	**3**
{Juice,Diaper}	**3**

Pairs (2-itemsets)
(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,
$^6C_1 + {}^6C_2 + {}^6C_3$
$6 + 15 + 20 = 41$
With support-based pruning,
$6 + 6 + 4 = 16$

Items (3-itemsets)

Itemset
{Juice, Diaper, Milk}
{Juice, Bread, Diaper}
{Bread, Diaper, Milk}
{Juice, Bread, Milk}

Generate 2-itemset candidates

Illustrating Apriori Principle

Items (1-itemsets)

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Juice	**3**
Diaper	**4**
Eggs	**1**

Items (2-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Juice, Bread}	**2**
{Bread,Diaper}	**3**
{Juice,Milk}	**2**
{Diaper,Milk}	**3**
{Juice,Diaper}	**3**

Pairs (2-itemsets)
(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 4 = 16$$

$$6 + 6 + 1 = 13$$

Items (3-itemsets)

Itemset	Count
{Juice, Diaper, Milk}	2
{Juice, Bread, Diaper}	2
{Bread, Diaper, Milk}	**2**
{Juice, Bread, Milk}	1

Prune 3-itemset candidates with infrequent 2-itemsets

Eliminate infrequent 3-itemset candidates

Apriori Algorithm

Fk: frequent k-itemsets **Ck**: candidate k-itemsets

- Let k=1
- Generate F1 = {frequent 1-itemsets}
- Repeat until Fk is empty
  - **Candidate Generation**
    - Generate Ck+1 from Fk
  - **Candidate Pruning**
    - Prune candidate itemsets in Ck+1 containing subsets of length k that are infrequent.
  - **Support Counting**
    - Count the support of each candidate in Ck+1 by scanning the DB
  - **Candidate Elimination**
    - Eliminate candidates in Ck+1 that are infrequent, leaving only those that are frequent => Fk+1

Apriori Algorithm: Candidate Generation

- **Brute-Force Method**
  - Exploring all possible combinations of itemsets to generate candidates.
  - Computationally intensive, especially for large datasets.
- **Merge Fk-1 and F1 Itemsets**
  - Combining frequent (k-1)-itemsets with individual items to form candidates.
  - Example: If {A, B} is frequent, combining it with {C} to create {A, B, C}.
- **Merge Fk-1 and Fk-1 Itemsets**
  - Merging (k-1)-itemsets with other (k-1)-itemsets to generate candidates.
  - Example: Merging {A, B} and {C, D} to create {A, B, C, D}.

Brute-Force Method

items

Items
Bread
Cola
Diapers
Eggs
Juice
Milk

→

Candidate Generation

Itemset
{Bread, Cola, Diapers}
{Bread, Cola, Eggs}
{Bread, Cola, Juice}
{Bread, Cola, Milk}
{Bread, Diapers, Eggs}
{Bread, Diapers, Juice}
{Bread, Diapers, Milk}
{Bread, Eggs, Juice}
{Bread, Eggs, Milk}
{Bread, Milk, Juice}
{Cola, Diapers, Eggs}
{Cola, Diapers, Juice}
{Cola, Diapers, Milk}
{Cola, Eggs, Juice}
{Cola, Eggs, Milk}
{Cola, Milk, Juice}
{Diapers, Eggs, Juice}
{Diapers, Eggs, Milk}
{Diapers, Juice, Milk}
{Eggs, Juice, Milk}

→

Candidate Pruning

Itemset
{Bread, Diapers, Milk}

Computationally intensive, especially for large datasets.

Merge Fk-1 and F1 Itemsets

Frequent 2-items

Itemset
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}
{Diapers,Juice}

Frequent 1-items

Items
Bread
Cola
Diapers
Eggs
Milk
Juice

Candidate Generation

Itemset
{Bread, Diapers, Juice}
{Bread, Diapers, Milk}
{Bread, Juice, Milk}
{Diapers, Milk, Juice}

Candidate Pruning

Itemset
{Bread, Diapers, Milk}

Merge Fk-1 and Fk-1 Itemsets

Frequent 2-items

Itemset
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}
{Diapers,Juice}

Frequent 2-items

Itemset
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}
{Diapers,Juice}

Candidate Generation

Itemset
{Bread, Diapers,Milk}

Candidate Pruning

Itemset
{Bread, Diapers, Milk}

Candidate Generation: $F_{k-1} \times F_{k-1}$ Method

- Merge two frequent (k-1)-itemsets if their first (k-2) items are identical

- F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE}
  - Merge(**AB**C, **AB**D) = **AB**CD
  - Merge(**AB**C, **AB**E) = **AB**CE
  - Merge(**AB**D, **AB**E) = **AB**DE

- Do not merge(**A**BD,**A**CD) because they share only prefix of length 1 instead of length 2

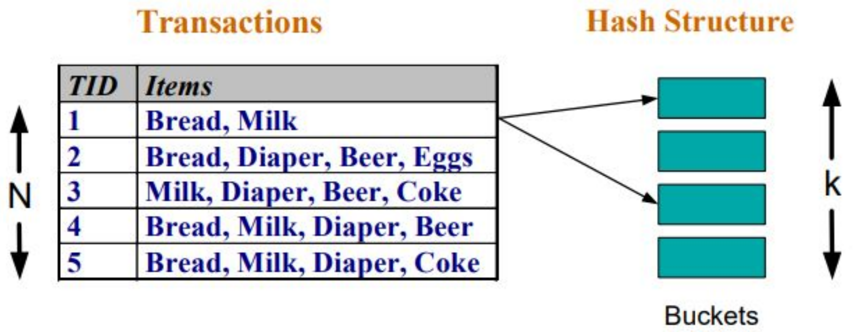
Apriori Algorithm: Candidate Pruning

- Let F3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE} be the set of frequent 3-itemsets
- C4 = {ABCD,ABCE,ABDE} is the set of candidate 4-itemsets generated
- Candidate pruning
  - Prune ABCE because ACE and BCE are infrequent
  - Prune ABDE because ADE is infrequent
- After **candidate pruning**: C4 = {ABCD}

Candidate pruning allows the removal of some itemsets without calculating the support, based on the anti-monotonic property.

Apriori Algorithm: Support Counting of Candidate Itemsets

- Scan the database of transactions to determine the support of each candidate itemset
  - Must match every candidate itemset against every transaction, which is an expensive operation
- To reduce the number of comparisons, store the candidates in a hash structure
  - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets

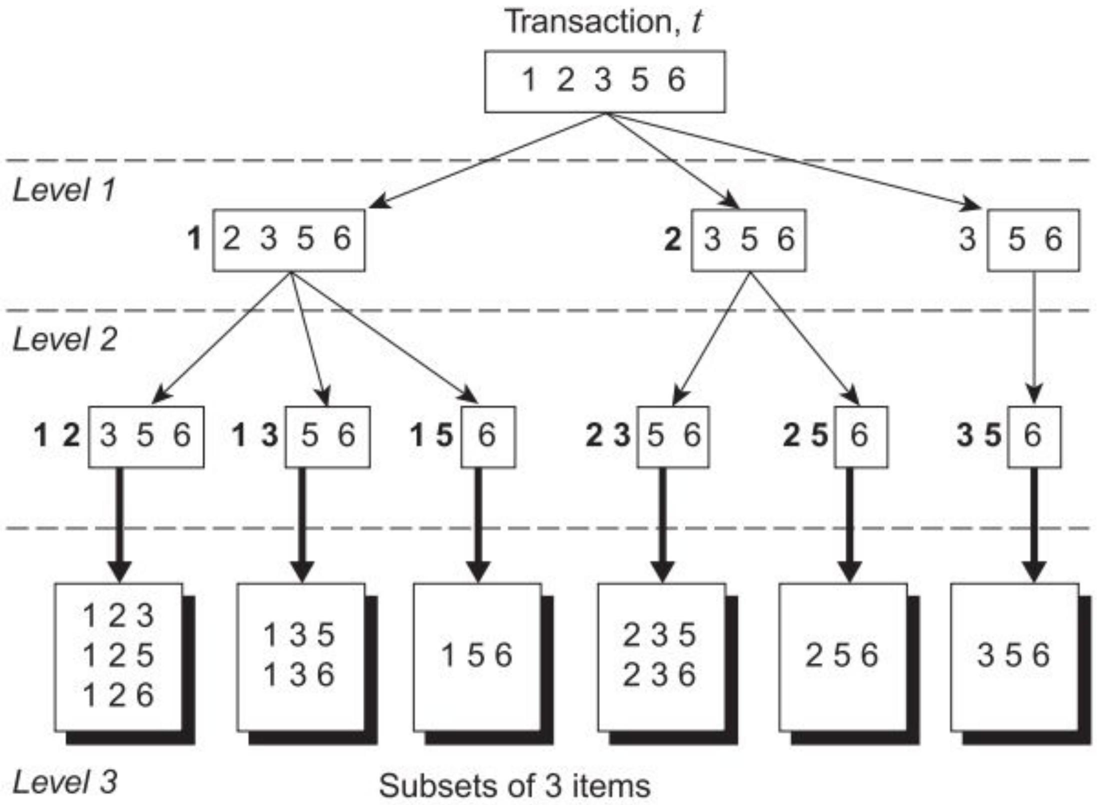


Transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

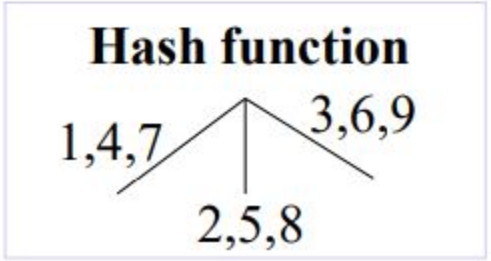
Support Counting Using Hash Tree

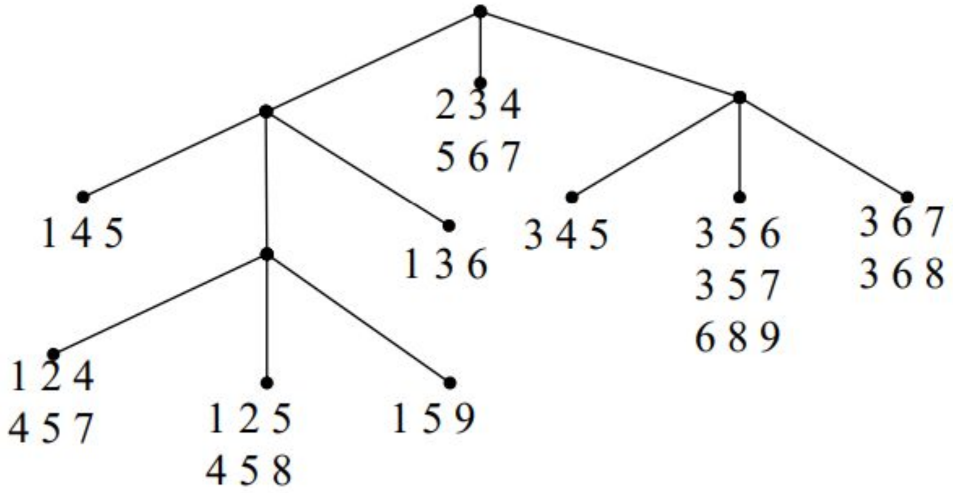
- Enumerating subsets of three items from a transaction t



Support Counting Using Hash Tree

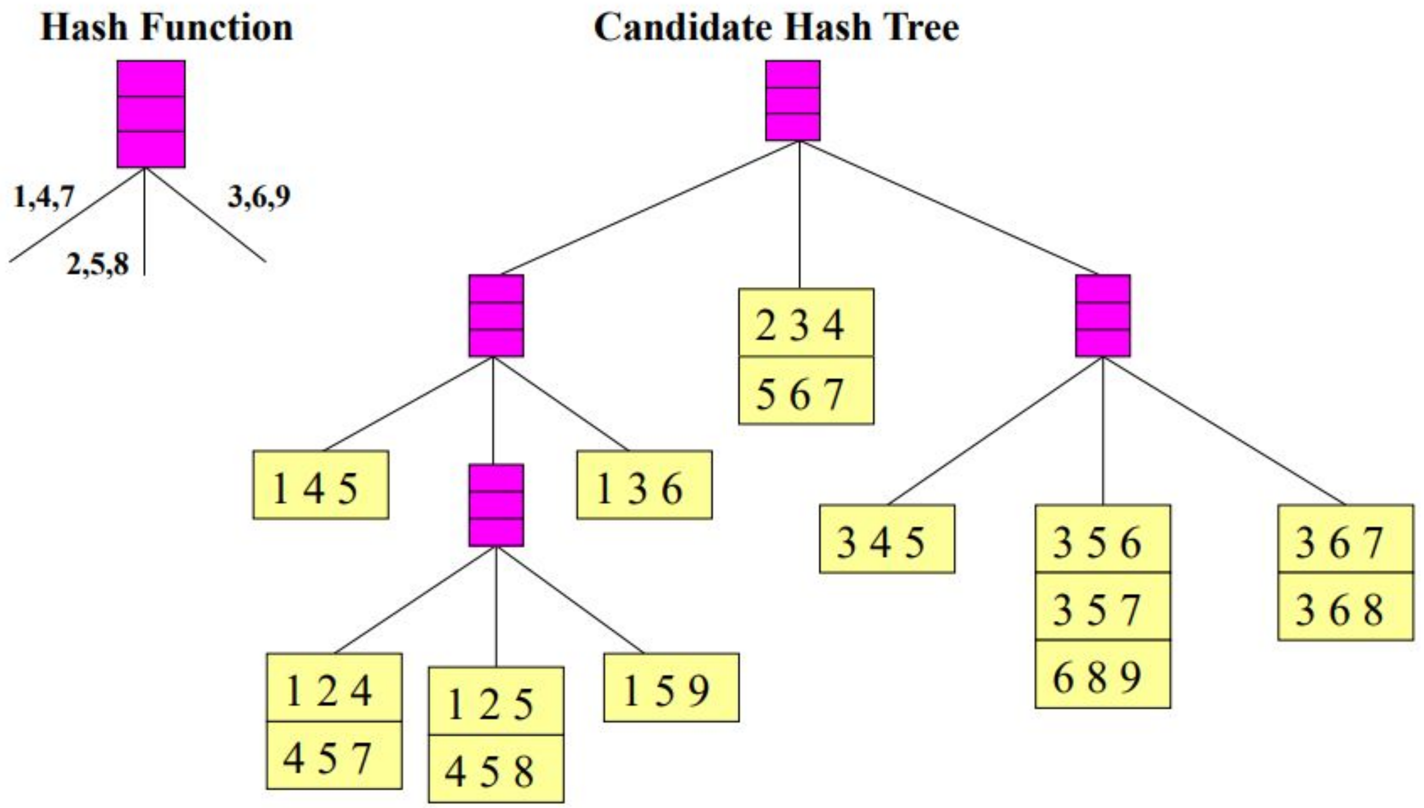
- 15 candidate itemsets of length 3:
  - {1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8},
  - {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5},
  - {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}.
- We need Hash function to locate the bucket
  - HashFunc: mod 3





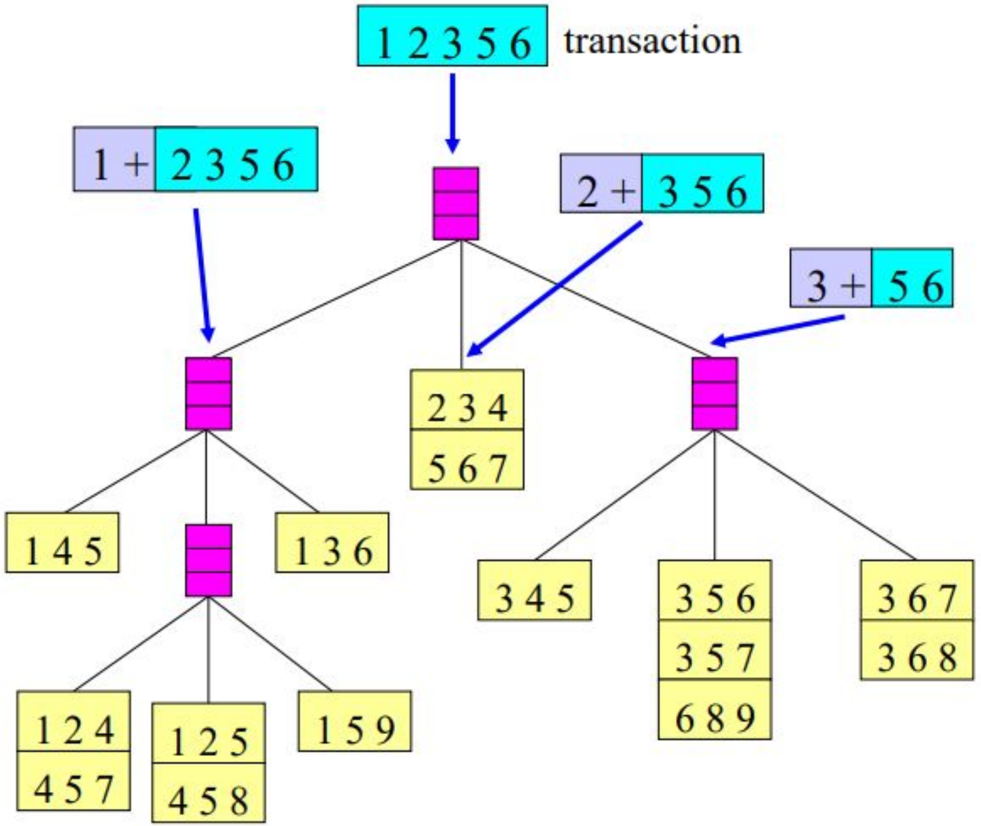
Support Counting Using a Hash Tree

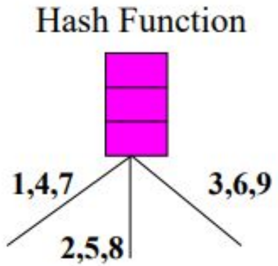
Generate Candidate Hash Tree



Support Counting Using a Hash Tree

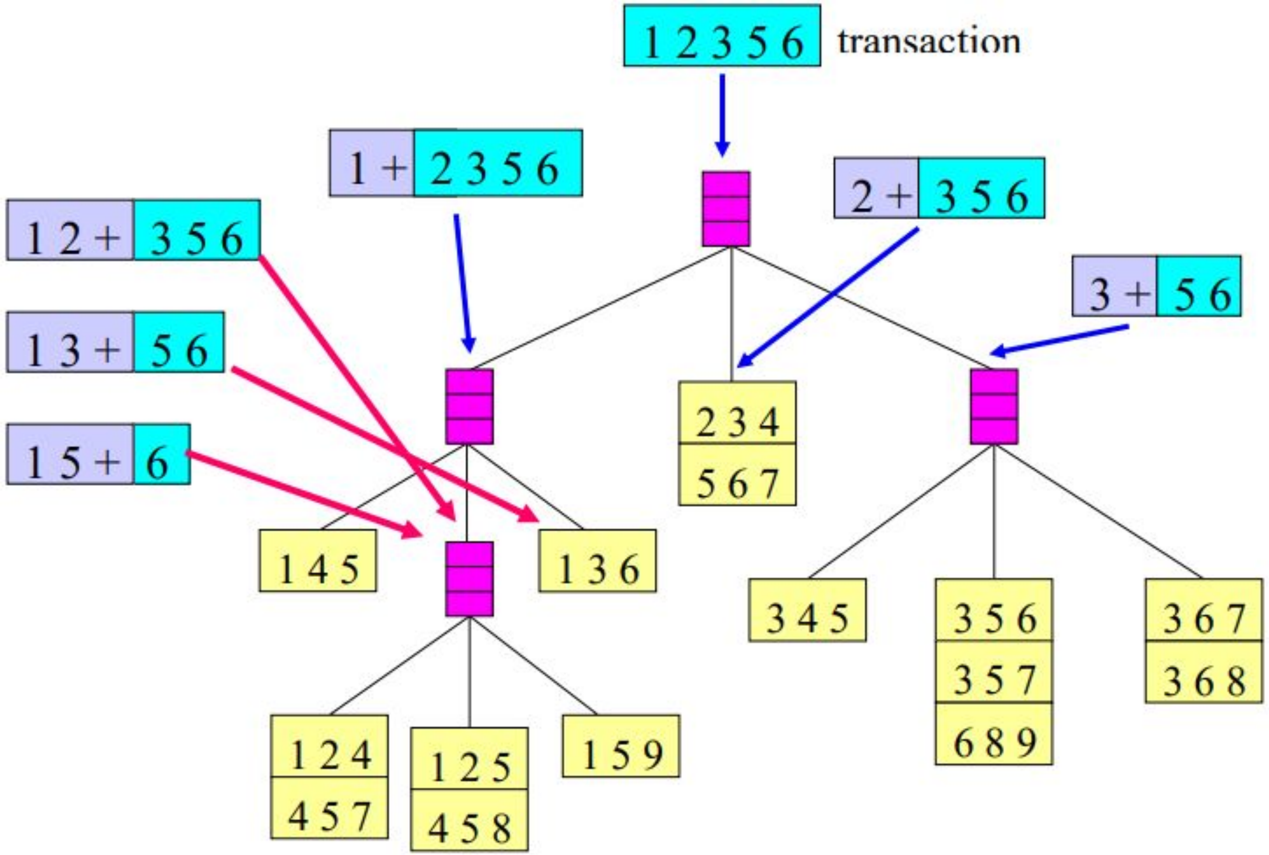
Traverse Candidate Hash Tree to Update Support Counts

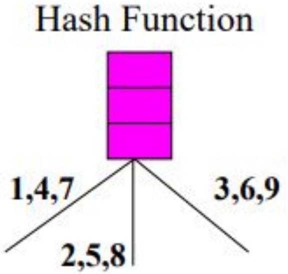




Support Counting Using a Hash Tree

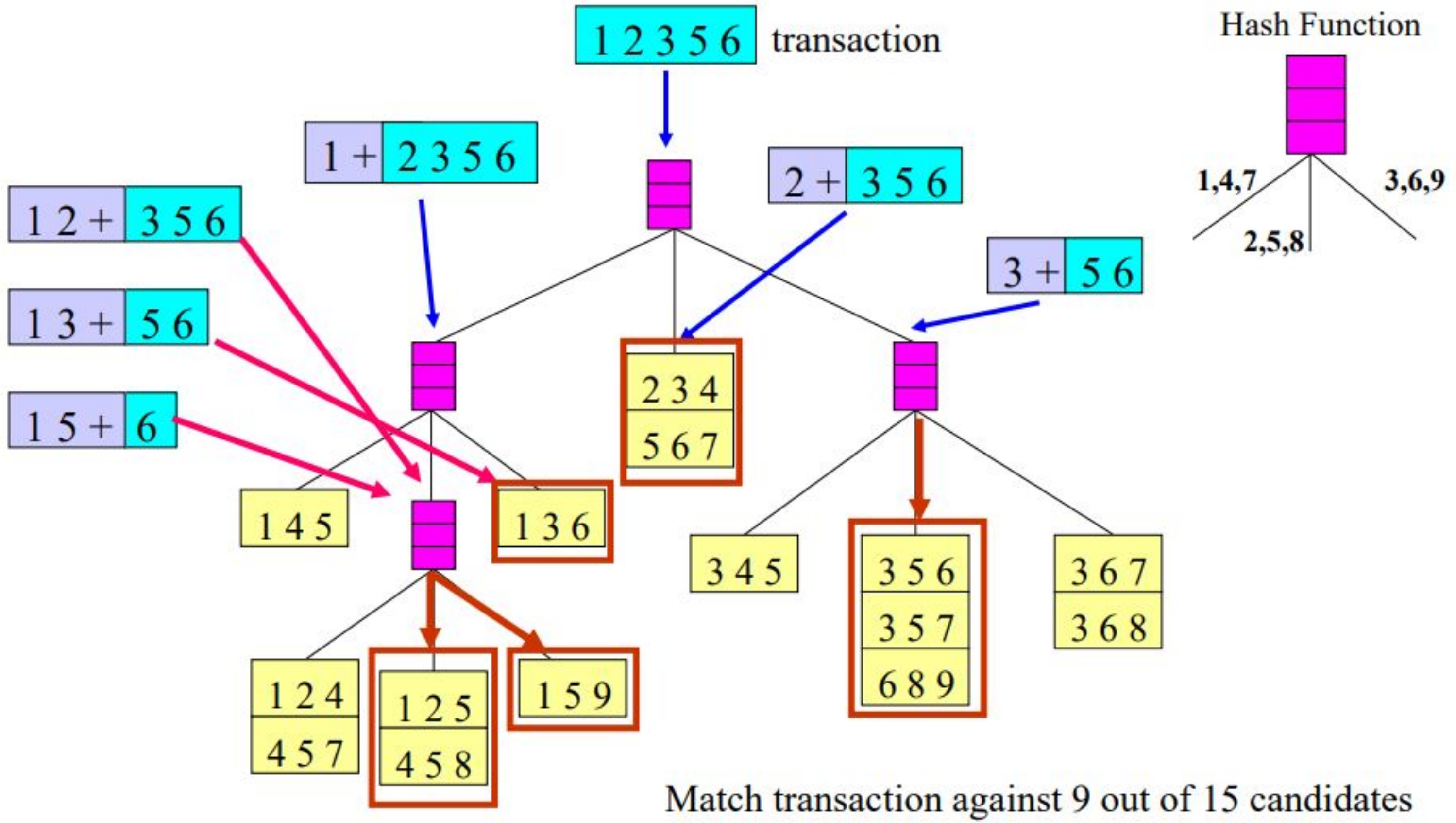
Traverse Candidate Hash Tree to Update Support Counts





Support Counting Using a Hash Tree

Traverse Candidate Hash Tree to Update Support Counts



Match transaction against 9 out of 15 candidates

Rule Generation

- Given a frequent itemset L, find all non-empty subsets f ⊂ L such that f → L – f satisfies the minimum confidence requirement
- If {A,B,C,D} is a frequent itemset, candidate rules:

  ABC →D, ABD →C, ACD →B, BCD →A,

  A →BCD, B →ACD, C →ABD, D →ABC

  AB →CD, AC → BD, AD → BC, BC →AD,

  BD →AC, CD →AB,
- If |L| = k, then there are **2^(k) – 2** candidate association rules
  - Ignoring L → ∅ and ∅ → L

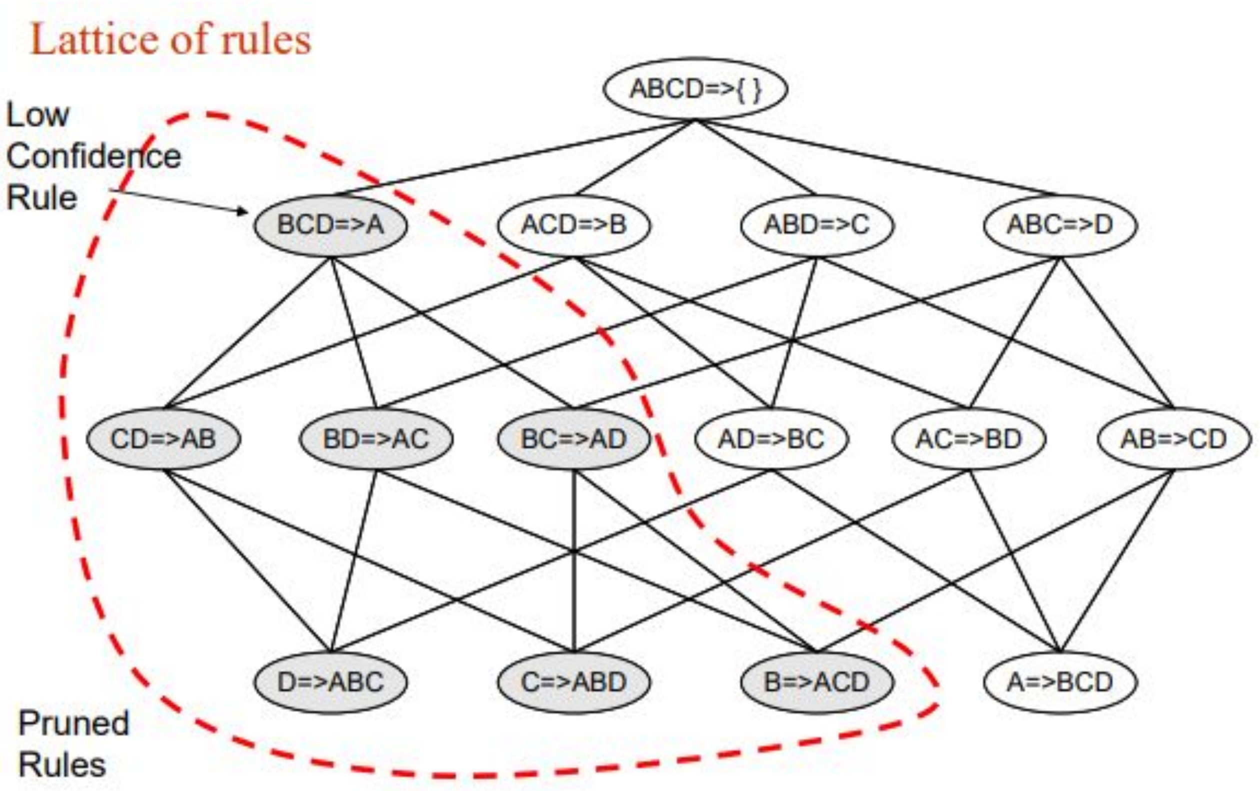
Apriori Algorithm: Rule Generation

- How to efficiently generate rules from frequent itemsets?

- In general, confidence does not have an anti-monotone property **c(ABC ⇒ D)** can be larger or smaller than **c(AB ⇒ D)**

- But confidence of rules generated from the same itemset has an anti-monotone property
  - E.g., Suppose {A,B,C,D} is a frequent 4-itemset: **c(ABC ⇒D) >= c(AB ⇒ CD) >= c(A ⇒ BCD)**
  - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Apriori Algorithm: Rule Generation



Agenda

1. **Frequent Itemsets, Association Rules**
2. **Apriori Algorithm**
3. **FP-Growth Algorithm**
4. Evaluation of Association Patterns

FP-Growth (Frequent Pattern Growth) Algorithm

- FP-growth takes a radically different approach to discovering frequent itemsets.
  - FP-growth does not subscribe to the **generate-and-test** paradigm of Apriori

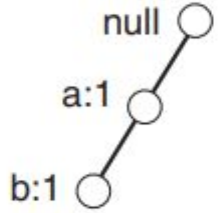
- Compressed Database Representation
  - Utilizes the FP-tree for a compressed database representation.
  - Enhances computational efficiency in capturing itemset relationships.

- FP-Tree: Compact Data Structure
  - Encodes the dataset using the efficient FP-tree.
  - Direct extraction of frequent itemsets, eliminating candidate generation.

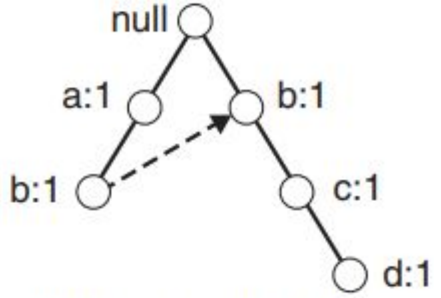
FP-Tree Construction

Transaction Data Set

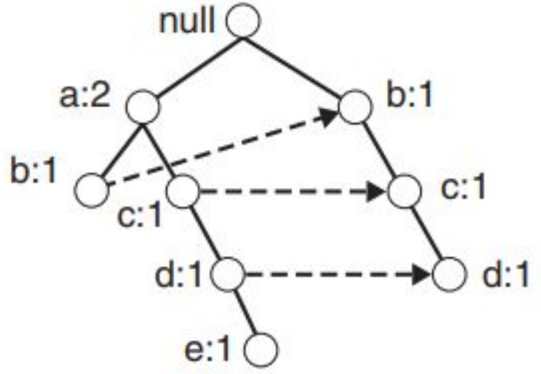
TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



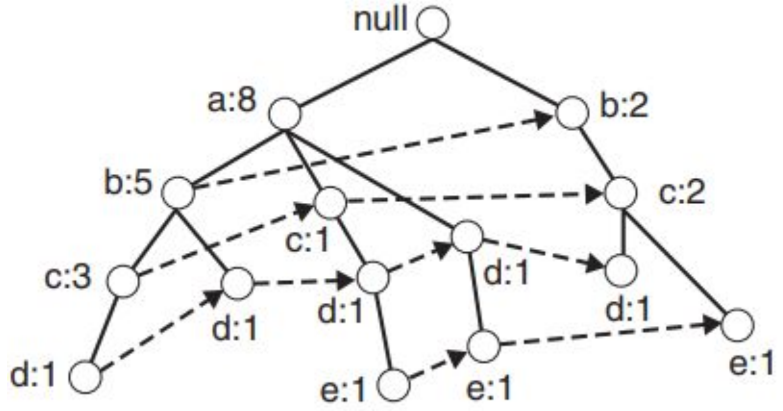
(i) After reading TID=1



(ii) After reading TID=2



(iii) After reading TID=3



(iv) After reading TID=10

FP-Tree Construction

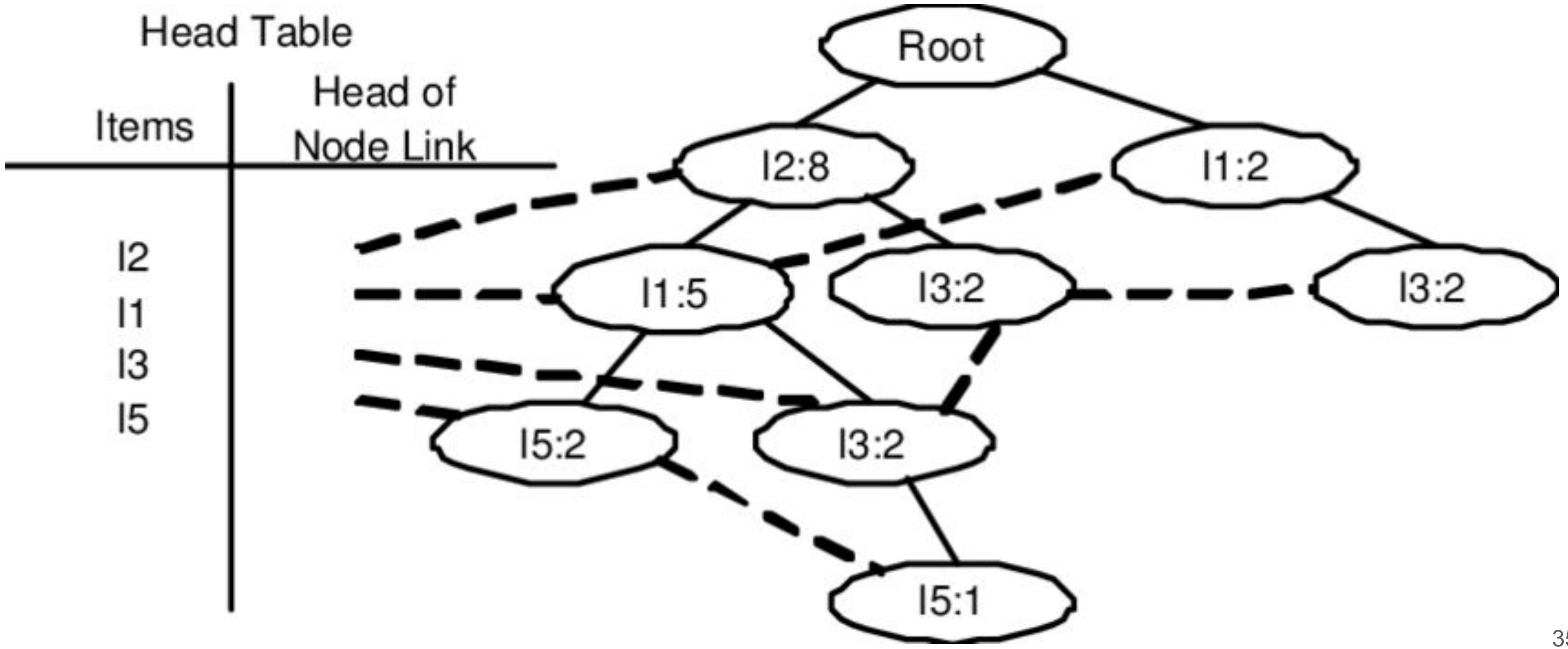
1. Scan the dataset to determine the support count of each item.
2. Discard infrequent items and sort frequent items in decreasing support counts within each transaction.
3. Make a second pass over the data to construct the FP-tree:
    a. Create nodes for each frequent item encountered.
    b. Form paths to encode transactions based on the created nodes.
    c. Increment frequency counts for nodes along the paths.
    d. Ensure disjoint paths for transactions without a common prefix.
4. Repeat steps 3a-3e for all transactions, mapping them onto paths in the FP-tree.

FP-Tree Construction

1. Scan the dataset to determine the support count of each item.
2. Discard infrequent items and sort frequent items in decreasing support counts within each t
3. Make a seco
   a. Create n
   b. Form paths to encode transactions based on the created nodes.
   c. Increment frequency counts for nodes along the paths.
   d. Ensure disjoint paths for transactions without a common prefix.
4. Repeat steps 3a-3e for all transactions, mapping them onto paths in the FP-tree.

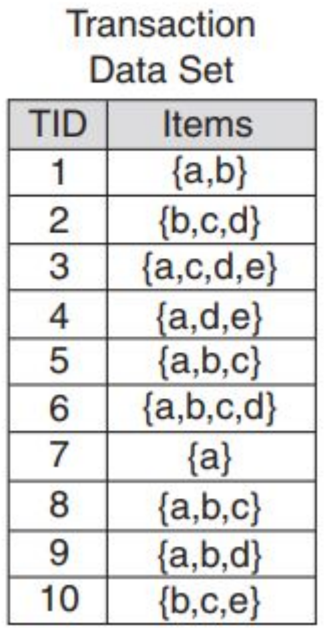
If a path is traversed by many transactions, it indicates consolidation, and their itemsets will be deemed frequent.

FP-Tree



Head Table
Items
Head of Node Link
I2
I1
I3
I5
Root
I2:8
I1:2
I1:5
I3:2
I3:2
I5:2
I3:2
I5:1

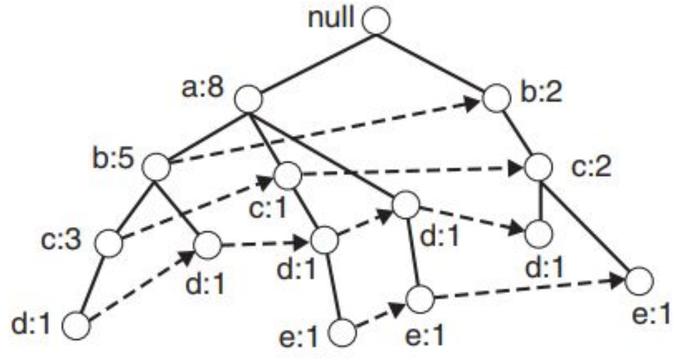
FP-Tree Construction and the order of items



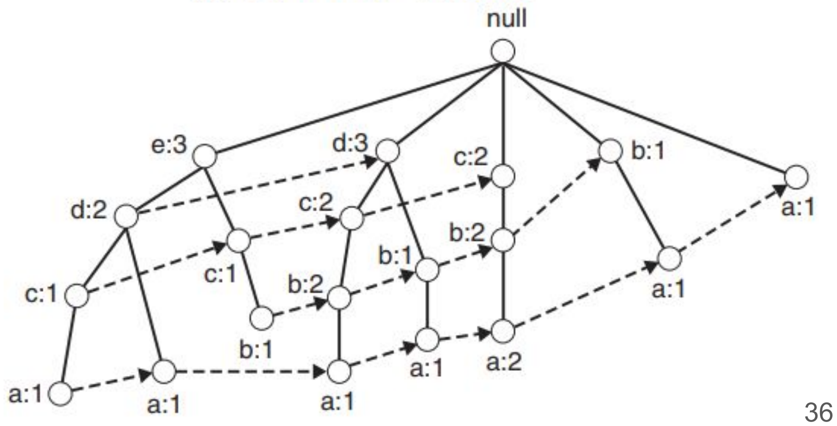
Transaction Data Set

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	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

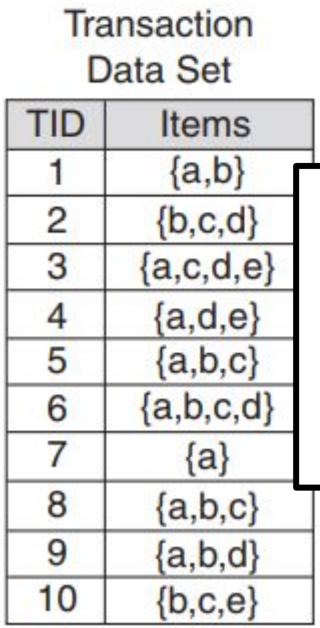
Order 1 starts by a



Order 2 starts by e

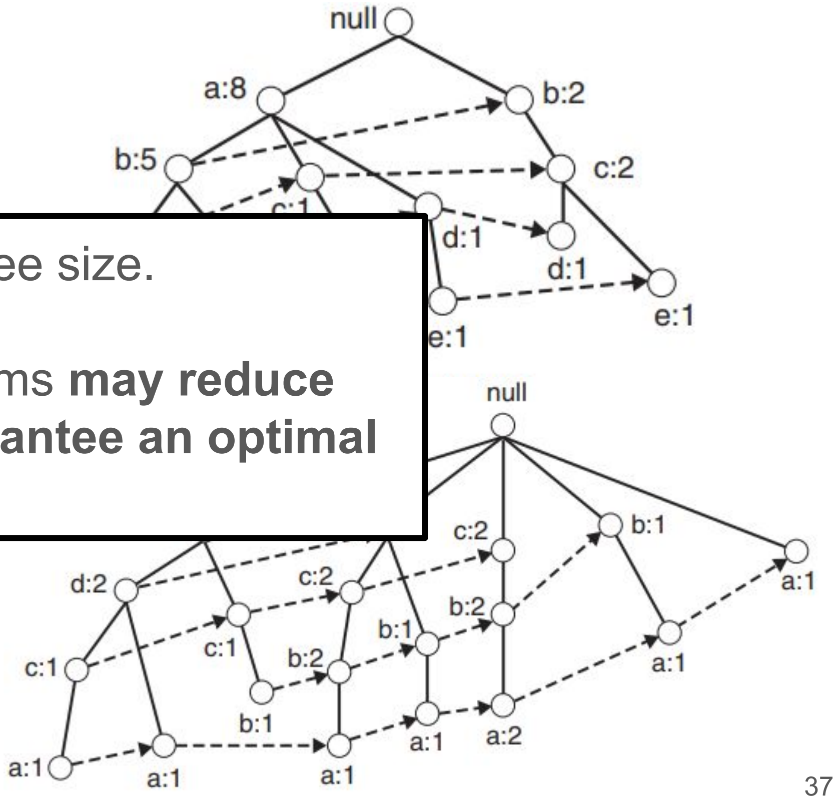


FP-Tree Construction and the order of items



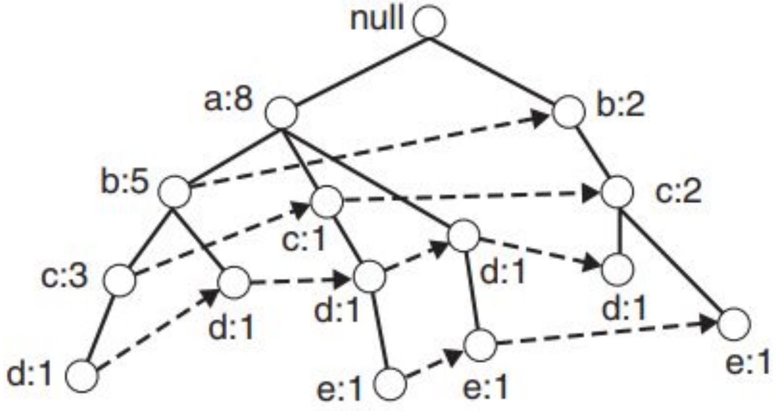
Transaction
Data Set

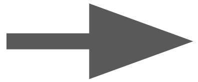
TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

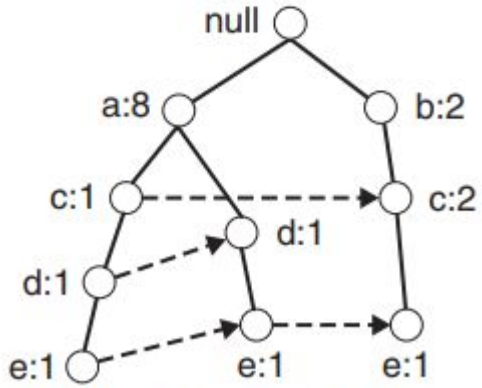


- Item order impacts FP-tree size.
- Starting with frequent items **may reduce the size**, it **doesn't guarantee an optimal FP-tree size**.

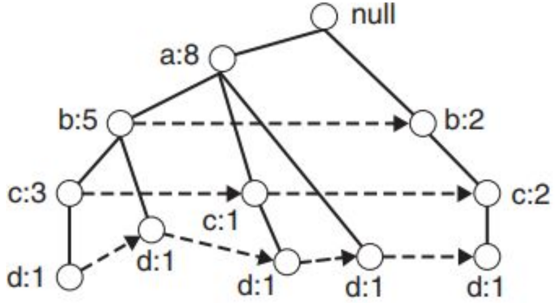
Frequent Itemset Generation in FP-Growth Algorithm





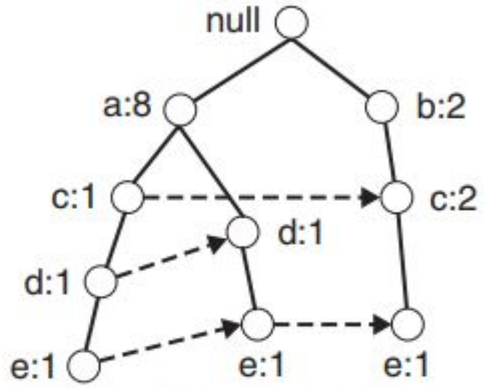


(a) Paths containing node e

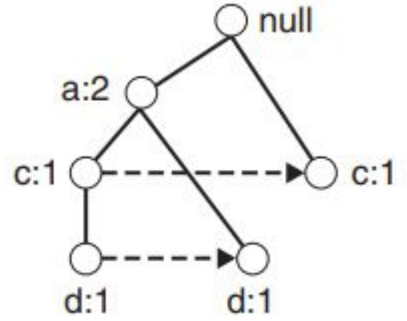


(b) Paths containing node d

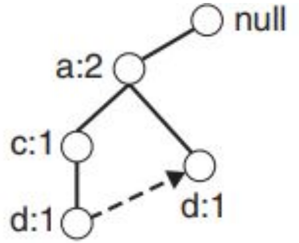
Frequent Itemset Generation in FP-Growth Algorithm



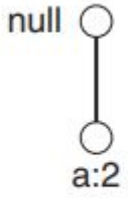
(a) Paths containing node e



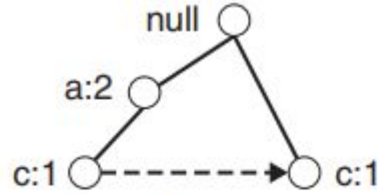
(b) Conditional FP-tree for e



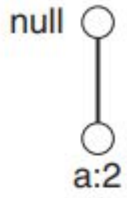
(c) Prefix paths ending in de



(d) Conditional FP-tree for de



(e) Prefix paths ending in ce



(f) Prefix paths ending in ae

FP-Growth: Example

Transaction ID	Items
T1	E,K,M,N,O,Y
T2	D,E,K,N,O,Y
T3	A,E,K,M
T4	C,K,M,U,Y
T5	C,E,I,K,O

Item	Frequency
K	5
E	4
M	3
O	3
Y	3
~~C~~	~~2~~
~~N~~	~~2~~
~~A~~	~~1~~
~~D~~	~~1~~
~~I~~	~~1~~
~~U~~	~~1~~

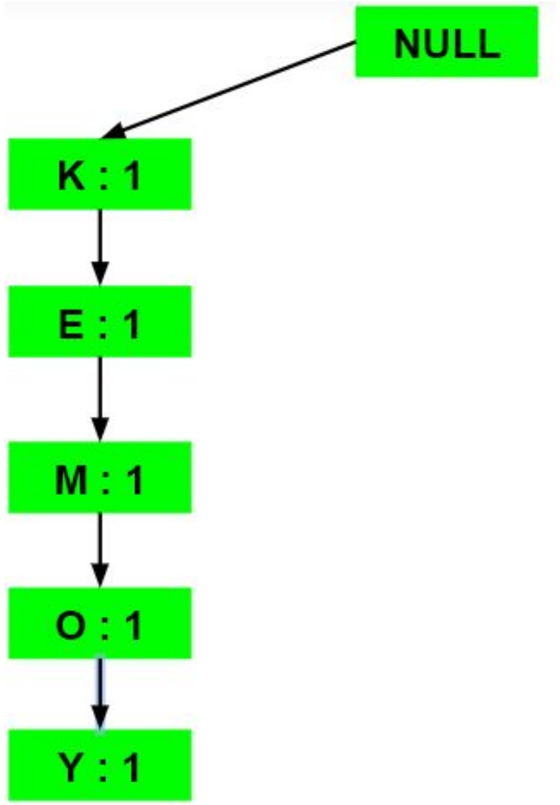
FP-Growth: Example

Item	Frequency
K	5
E	4
M	3
O	3
Y	3

Transaction ID	Items	Ordered-Item set
T1	E,K,M,N,O,Y	K,E,M,O,Y
T2	D,E,K,N,O,Y	K,E,O,Y
T3	A,E,K,M	K,E,M
T4	C,K,M,U,Y	K,M,Y
T5	C,E,I,K,O	K,E,O

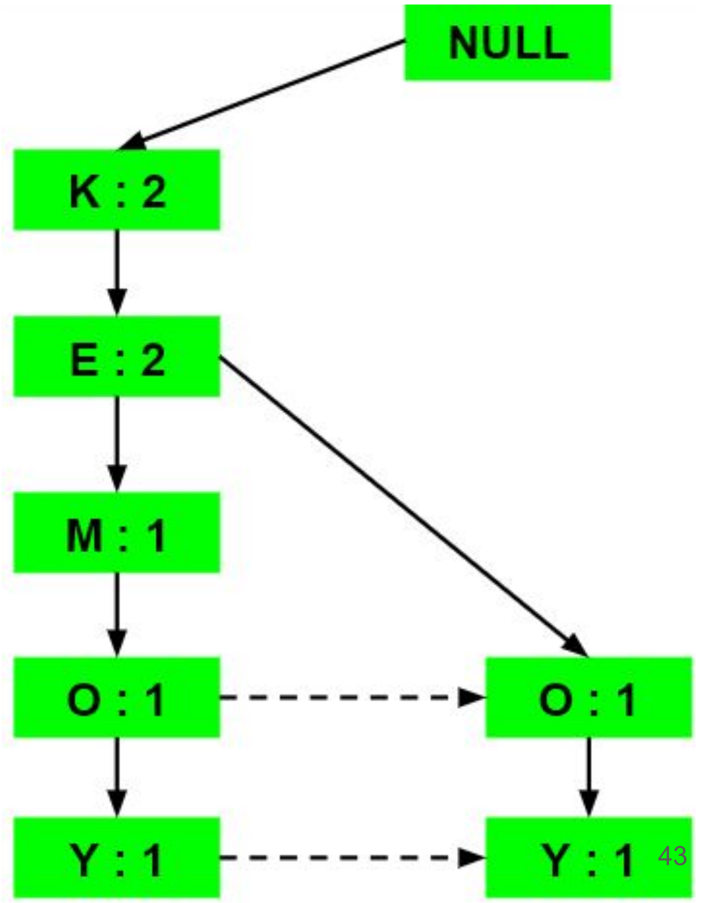
FP-Growth: Example

Transaction ID	Items	Ordered-Item set
T1	E,K,M,N,O,Y	K,E,M,O,Y
T2	D,E,K,N,O,Y	K,E,O,Y
T3	A,E,K,M	K,E,M
T4	C,K,M,U,Y	K,M,Y
T5	C,E,I,K,O	K,E,O


$$\begin{tikzcd} & \text{NULL} \arrow[dl] \\ K:1 \arrow[d] & \\ E:1 \arrow[d] & \\ M:1 \arrow[d] & \\ O:1 \arrow[d] & \\ Y:1 & \end{tikzcd}$$

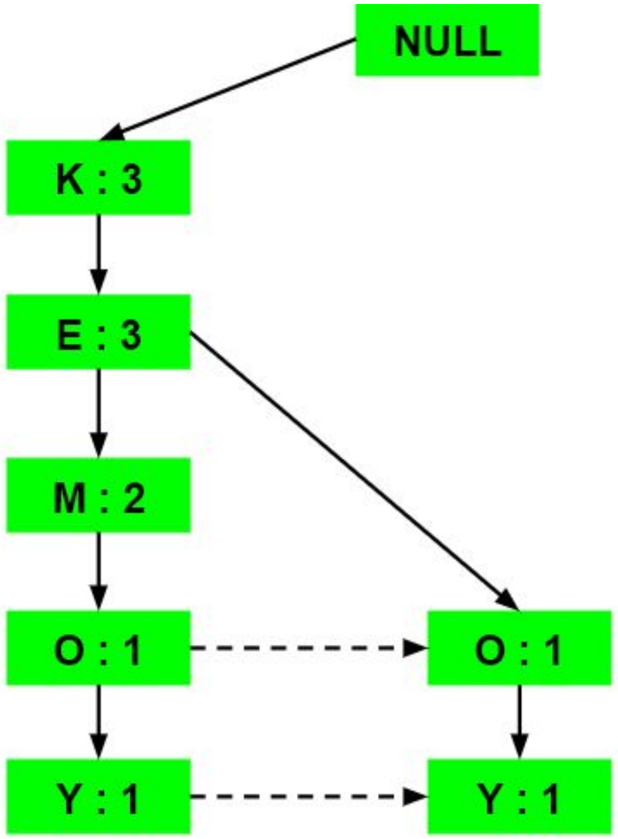
FP-Growth: Example

Transaction ID	Items	Ordered-Item set
T1	E,K,M,N,O,Y	K,E,M,O,Y
T2	D,E,K,N,O,Y	K,E,O,Y
T3	A,E,K,M	K,E,M
T4	C,K,M,U,Y	K,M,Y
T5	C,E,I,K,O	K,E,O



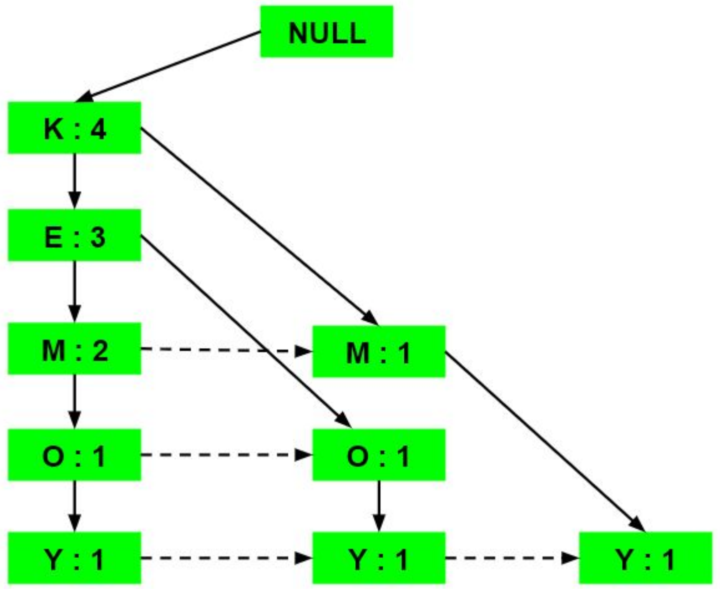
FP-Growth: Example

Transaction ID	Items	Ordered-Item set
T1	E,K,M,N,O,Y	K,E,M,O,Y
T2	D,E,K,N,O,Y	K,E,O,Y
T3	A,E,K,M	K,E,M
T4	C,K,M,U,Y	K,M,Y
T5	C,E,I,K,O	K,E,O



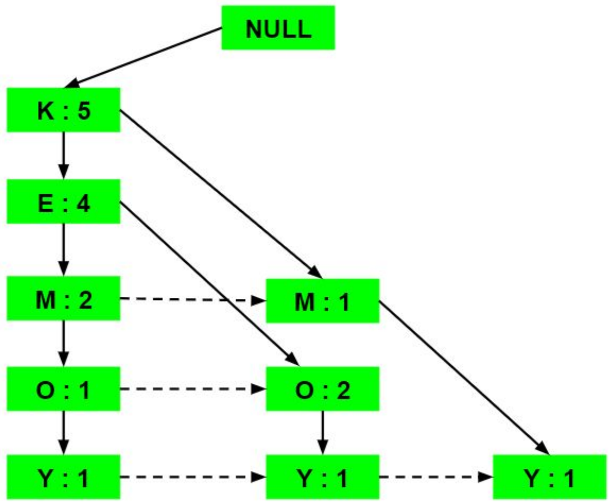
FP-Growth: Example

Transaction ID	Items	Ordered-Item set
T1	E,K,M,N,O,Y	K,E,M,O,Y
T2	D,E,K,N,O,Y	K,E,O,Y
T3	A,E,K,M	K,E,M
T4	C,K,M,U,Y	K,M,Y
T5	C,E,I,K,O	K,E,O



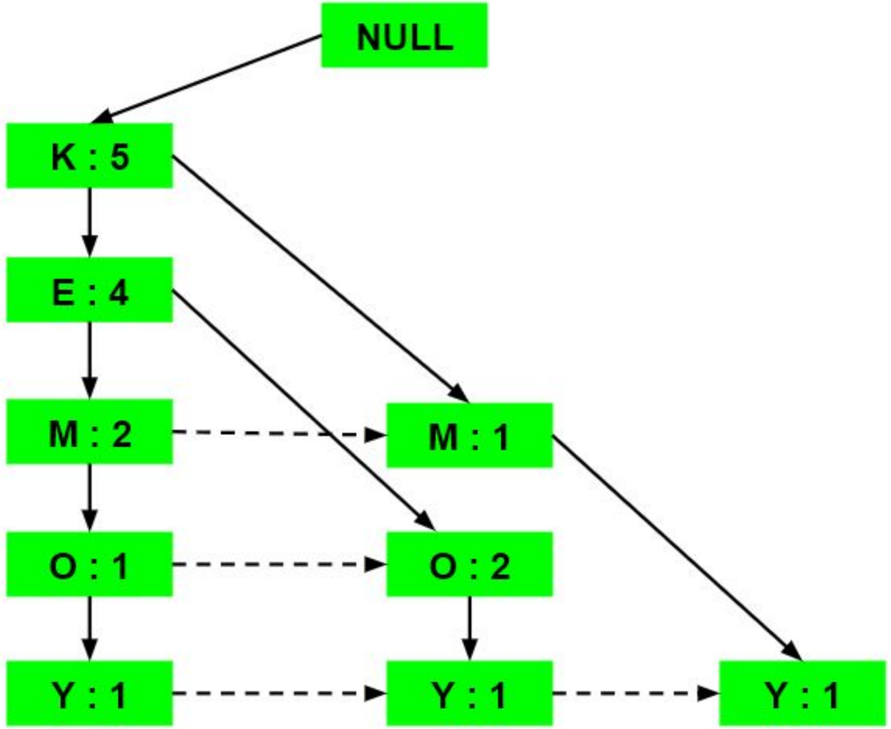
FP-Growth: Example

Transaction ID	Items	Ordered-Item set
T1	E,K,M,N,O,Y	K,E,M,O,Y
T2	D,E,K,N,O,Y	K,E,O,Y
T3	A,E,K,M	K,E,M
T4	C,K,M,U,Y	K,M,Y
T5	C,E,I,K,O	K,E,O

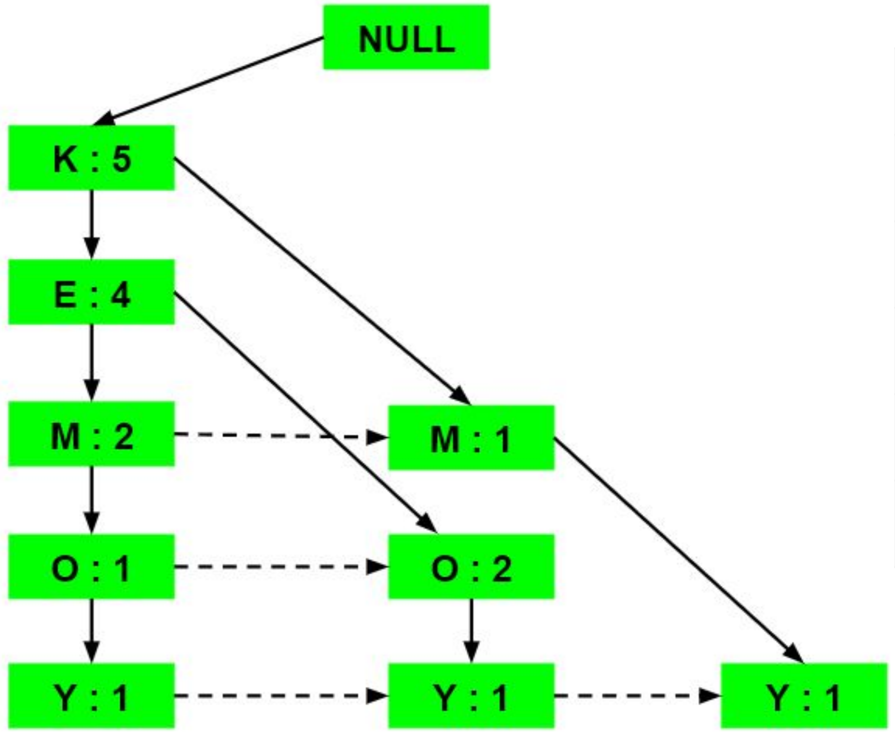


FP-Growth: Example

Transaction ID	Ordered-Item set
T1	K,E,M,O,Y
T2	K,E,O,Y
T3	K,E,M
T4	K,M,Y
T5	K,E,O



FP-Growth: Example



Item	Conditional Pattern Base
Y	{{K, E, M, O: 1}, {K, E, O: 1}, {K, M: 1}}
O	{{K, E, M: 1}, {K, E: 2}}
M	{{K, E: 2}, {K: 1}}
E	{K: 4}
K	

FP-Growth: Example

Item	Conditional Pattern Base	Conditional Frequent Pattern
Y	{{K, E, M, O: 1}, {K, E, O: 1}, {K, M: 1}}	{K:3}
O	{{K, E, M: 1}, {K, E: 2}}	{K,E:3}
M	{{K, E: 2}, {K: 1}}	{K:3}
E	{K: 4}	{K:4}
K		

FP-Growth: Example

Item	Conditional Frequent Pattern	Frequent Pattern rules
Y	{K:3}	{K,Y}:3
O	{K,E:3}	{K,O}:3 {E,O}:3 {E,K,O}:3
M	{K:3}	{K,M}:3
E	{K:4}	{E,K}:3
K		

Agenda

1. **Frequent Itemsets, Association Rules**
2. **Apriori Algorithm**
3. **FP-Growth Algorithm**
4. **Evaluation of Association Patterns**

Pattern Evaluation

- Association rule algorithms can produce large number of rules
- **Interestingness measures** can be used to prune/rank the patterns
- In the original formulation, support & confidence are the only measures used

Computing Interestingness Measure

- Given X → Y or {X,Y}, information needed to compute interestingness can be obtained from a contingency table

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	N

Contingency table

f_{11}: support of X and Y
f_{10}: support of X and Y
f_{01}: support of X and Y
f_{00}: support of X and Y

Used to define various measures

- support, confidence, Lift……

Drawback of Confidence

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

Association Rule: Tea → Coffee

- Confidence= P(Coffee|Tea) = 150/200 = 0.75
- P(Coffee) = 0.8
  - Knowing that a person drinks tea reduces the probability that the person drinks coffee!
- P(Coffee|$\overline{\text{Tea}}$) = 650/800 = 0.8125

Drawback of Confidence

Customers	Tea	Honey	…
C1	0	1	…
C2	1	0	…
C3	1	1	…
C4	1	0	…
…			

	$Honey$	$\overline{Honey}$	
Tea	100	100	200
$\overline{Tea}$	20	780	800
	120	880	1000

Association Rule: Tea $\rightarrow$ Honey

- Confidence ≅ P(Honey|Tea) = 100/200 = 0.50
- Confidence = 50% $\Rightarrow$ drinking tea has little influence on honey usage
  - The rule seems uninteresting
- P(Honey) = 120/1000 = 0.12
  - Tea drinkers are far more likely to have honey

Statistical-Based Measures for Interestingness

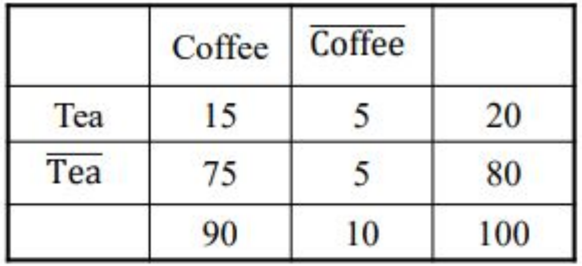
- Statistical-Based Measures use statistical dependence information.

- Lift = P(Y|X) / P(Y)
- Lift(A,B) = conf(A→B) / support(B) = support(A ∪ B) / support(A) support(B)

- Interest = P(X,Y) / P(X) P(Y)
- Interest(A,B) = support(A ∪ B) / support(A) support(B)

$$\text{Interest(A,B)} \begin{cases} = 1 & if\ A\ \ and\ B\ are\ independent \\ > 1 & if\ A\ and\ B\ are\ positively\ correlated \\ < 1 & if\ A\ and\ B\ are\ negatively\ correlated \end{cases}$$

Example: Lift/Interest



	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea ⇒ Coffee

- Confidence= P(Coffee|Tea) = 0.75 = support({Tea,Coffee}) / support({Tea})
- P(Coffee) = 0.9
- Lift = 0.75/0.9 = 0.8333 (< 1, therefore is negatively correlated)