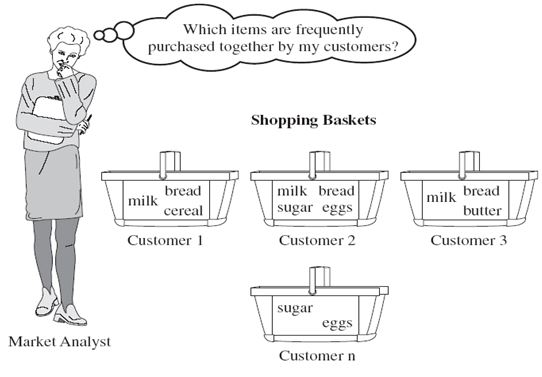
**UNIT-IV**

**ASSOCIATION ANALYSIS**

**Association Analysis:** Association Analysis Is The Task Of Uncovering Relationships Among The Data.

**Association Rules:** it is a model that identifies how the data itms are associated with each other.

***Market Basket Analysis:***

A huge amount of data is collected on movements of clients shopping in supermarkets and retail sector. The most typical example of association rules is "market basket analysis" which is a modeling  technique based on the idea that if one  buy a certain group of items, then he/she is  more likely to buy (or not to buy) another group of items. The discovery of this type of associations may provide important opportunity for market managers to develop more effective marketing strategies. For example, X% of customers buying sugar also buy eggs. This information can be found with association rule method.        

Market basket analysis

**For example,** the information that customers who purchase computers also tend to buy financial management software at the same time is represented in association Rule.

**computer =>financial management software [support = 2%; confidence = 60%]**

Example of association rule mining is **market basket analysis**. This process analyzes customer buying habits by finding associations between the different items that customers place in their “shopping baskets”.

**Association Rules techniques:**

Generate strong association rules from the frequent itemsets: those rules must satisfy minimum support and minimum confidence.

Type of Association Rules

**1.Boolean AR:**

it is a rule that checks whether an item is present or absent.

All the examples we have seen so far are Boolean AR.

**2.Quantitative AR:**

It describes associations between quantitative items or attributes.

Generally, quantitative values are partitioned into intervals.

Example:

Age(X,”30..39”) ∧ income(X,”80K..100K”)

🡺 buys(X, High Resolution TV)

**3.Single-Dimension AR:**

It is a rule that references only one dimension.

Example:

buys(X,”computer”)

🡺 buys(X,”financial\_software”)

The single dimension is “buys”

The following rule is a multi-dimensional AR:

Age(X,”30..39”) ∧ income(X,”80K..100K”)

🡺 buys(X, High Resolution TV)

**4.Multi-level AR**

* It is a set of rules that reference different levels of abstraction.

Example:

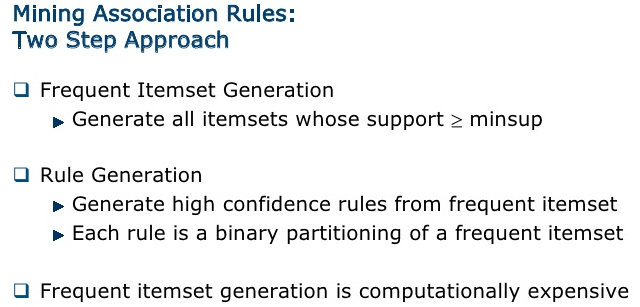
Age(X,”30..39”) 🡺 buys(X, “desktop”)

Age(X,”20..29”) 🡺 buys(X, “laptop”)

Laptop 🡪 desktop 🡪 computer

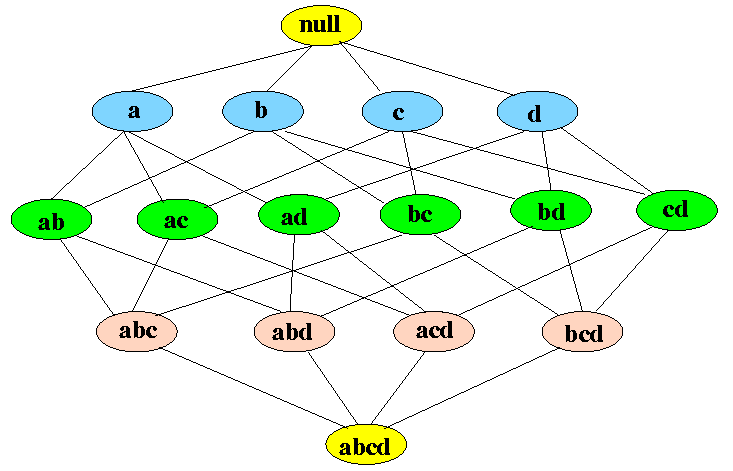
**Applications of association rules**

* Market basket data analysis, which aims to discover how items purchased by customers in a supermarket are associated.
* Shopping centres use association rules to place the items next to each other so that users buy more items.
* Amazon use association mining to recommend you the items based on the current item you are browsing/buying.
* Medical Application to make decision about medical diagnose should be assigned to this patient?

****

**1.Frequent Itemset generation: Subsets and lattices**

* A **lattice structure** can be used to **enumerate** the **list of all possible subsets**
* **Example:** **all subset** of **{a,b,c,d}**

Given d items, there are 2d possible candidate itemsets 

There are **2d - 1** **non-empty** subsets of a ***k*-item set**

**The Apriori Principle**

|  |
| --- |
| * + - If an **itemset *x*** is **frequent** (i.e., ***freq(x)* ≥ θ*N***), then:   **all subsets of *x*** is also **frequent** |

* **Example:** if **{b,c,d}** is **frequent**, then ***all* subsets** of **{b,c,d}** are also **frequent**

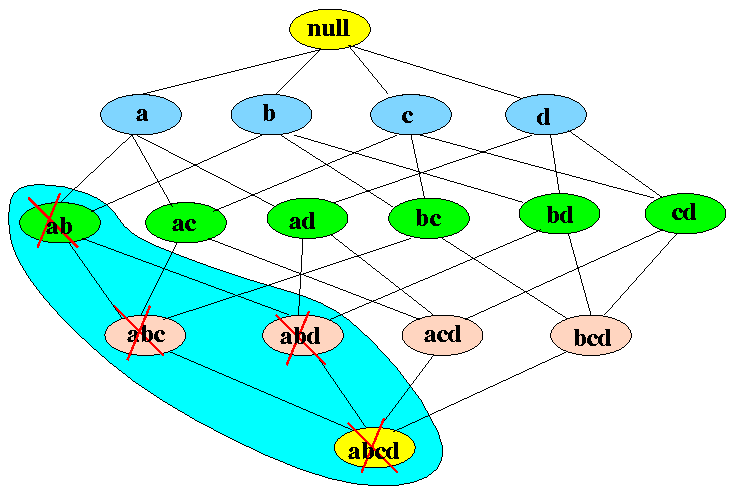
|  |
| --- |
| http://www.mathcs.emory.edu/%7Echeung/Courses/584-StreamDB/Syllabus/10-Mining/FIGS/lattice02.gif |

**Applying the Apriori Principle to eleminate Candidate Sets**

* **Converse of the Apriori Principle:**

|  |
| --- |
| * + - If an **itemset *x*** is ***not* frequent** (i.e., ***freq(x)* < θ*N***), then:   **all *super* sets of *x*** are ***not* frequent** |

* **Example:** if **{a,b}** is **infrequent**, then all its **super sets ({a,b,c}, {a,b,d}, and {a,b,c,d})** are also **infrequent**:



* **Conclusion:**

|  |
| --- |
| * + - If a **node** ***x*** is **infrequent**, the **entire subgraph** rooted at ***x*** can be **pruned** away from the **candidate set** |

**BRUTE-FORCE APPROACH:** Each itemset in the lattice is a candidate frequent itemset.Count the support of each candidate by scanning the database

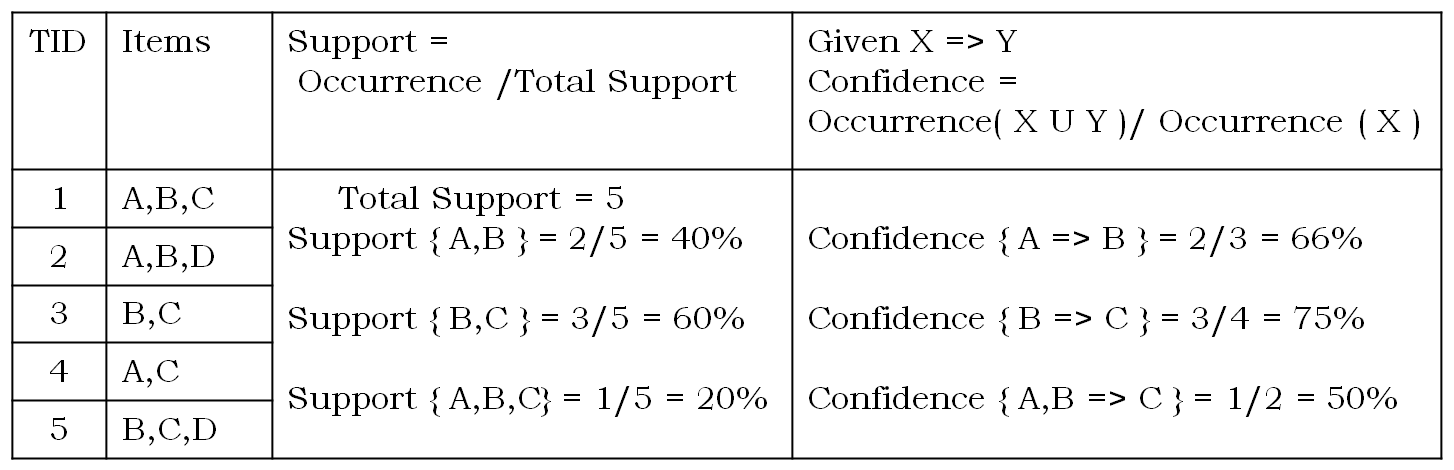
**What is Frequent Item Set ?**

It refers to set of items that frequently appear together and satisfies both [**minimum**](http://athena.ecs.csus.edu/%7Ekhambamb/Association.html#Minimum) [**support**](http://athena.ecs.csus.edu/%7Ekhambamb/Association.html#Support) threshold and [**minimum**](http://athena.ecs.csus.edu/%7Ekhambamb/Association.html#Minimum) [**confidence**](http://athena.ecs.csus.edu/%7Ekhambamb/Association.html#Confidence) threshold.

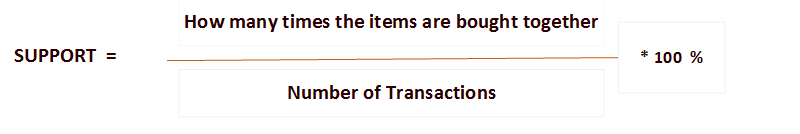
**For example** :Consider the below transaction where items A, B, C are brought together in first transaction (TID 1) , items A,B,D are brought together in second transaction (TID 2) and so on.

Given that minimum threshold support is 60% and minimum threshold confidence is 70%.

We can clearly see that items { B, C } has support (60%) >= minimum threshold support and confidence (75%) >= minimum threshold confidence . So { B, C } is a frequent item set.



**What is support :**It is the percentage of the transaction, in which all the items in the item set is bought together.

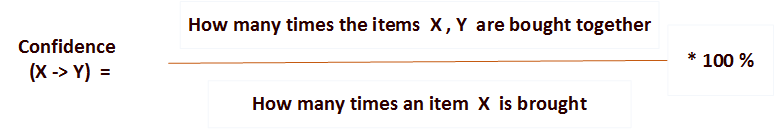


**For example** :Consider the itemset { B, C } in the above transaction, as items B and C are bought together in 3 out of 5 transactions.

**Support (B, C)** = ( 3 / 5 ) \* 100 % = 60%

**Formulae:** Support (A , B)   =   Probability(A U B)

**What is confidence :**The rule X => Y holds with confidence C if in C% of the transaction, customers who purchased a X also bought the Y.



**For example** :Consider the itemset { B, C } in the above transaction, As both items B and C are bought together in 3 transactions and item B is bought in 4 transactions.

**Confidence (B -> C)** = ( 3 / 4 ) \* 100 % = 75%

**Formulae:** Confidence (A -> B)   =   Probability(A U B)/Probability(A) [7].

**What is minimum support/confidence threshold ?**

Association rules are considered interesting if they satisfy minimum support threshold and minimum confidence threshold which is set by users or domain experts. If an itemset I does not satisfy the minimum support threshold, then I is not frequent

## Association Rule Mining Algorithms : this are two types

* **Apriori Algorithm.: finding frequent item sets using candidate generation.**
* **FP Growth Algorithm:finding frequent item sets using with out candidate generation.**

**Apriori Algorithm :**

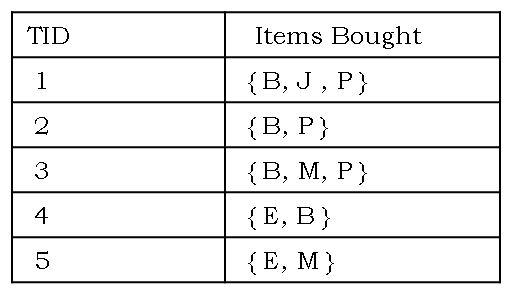
* **A priori is a seminal algorithm pro**posed by R.agarwal and R.srikanth in 1994 for mining frequent item set for boolen association rule.
* Apriori employes an iterative approach known as level wise approach.
* To improve the efficiency of the level wise generation of frequent item sets an important property called apriori property.
* Apriori property:all non empty subsets of frequent item sets must be also frequent

It is a classic algorithm used in data mining for finding association rules based on the principle "Any subset of a large item set must be large". It uses a generate-and-test approach – generates candidate itemsets and tests if they are frequent.

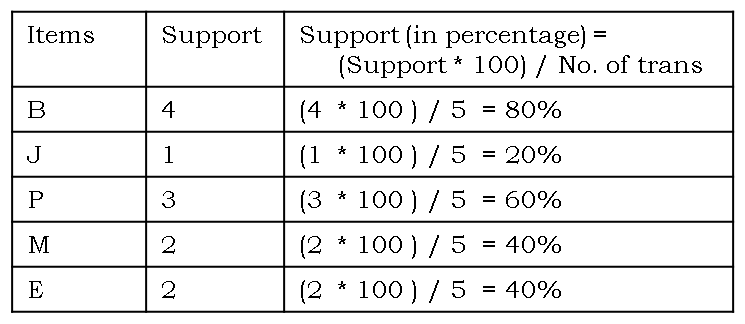
**Frequent Itemset Generation:**

Given the [**mininum threshold support**](http://athena.ecs.csus.edu/%7Ekhambamb/Association.html#Minimum), Generating large item sets (only keep [**frequent item sets**](http://athena.ecs.csus.edu/%7Ekhambamb/Association.html#frequentItem) – large item sets with enough support).

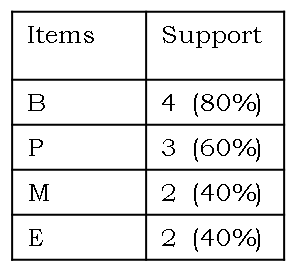
**Illustration:**Consider the below transaction in which B = Bread, J = Jelly, P = Peanut Butter, M = Milk and E = Eggs. Given that minimum threshold support = 40% and minimum threshold confidence = 80% [13].



**Step-1:** Count the number of transactions in which each item occurs (Bread B occurs in 4 transactions and so on).

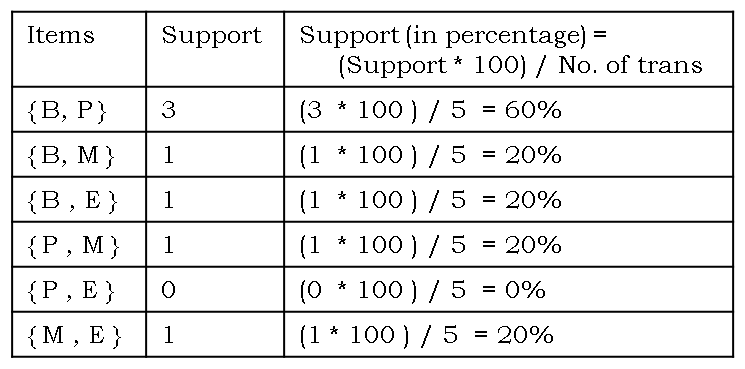


**Step-2:** As minimum threshold support = 40%, So in this step we will remove all the items that are bought less than 40% of support or support less than 2.

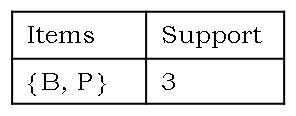


The above table has single items that are bought frequently. Now let’s find a pair of items that are bought frequently. We continue from the above table (Table in step 2)

**Step-3:** We start making pairs from the first item and below items like {B,P} ,{B,M} ,{B,E} and then we start with the second item and below items like {P,M} ,{P,E}. We do not make pair {P,B} because we already made {P,B} pair when we were making pairs of B. As buying a bread and Peanut Butter together is same as buying Peanut Butter and bread together. After making all the pairs we get,



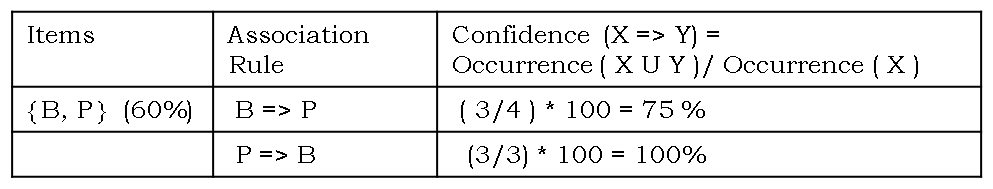
**Step-4:** As minimum threshold support = 40%, So in this step we will remove all the items that are bought less than 40% of support and we are left with



The above table has two items {B , P } that are bought together frequently.

**Association Rule Generation :**

**Step-5:** As we cannot generate large frequent item (itemset of 3) further because we are left with 1 frequent item set. We will start generating association rules from the frequent item set. As we have frequent item set of two, only two association rules will be generated which is shown below :



As P -> B has confidence 100% which is greater than minimum confidence threshold 80%, thus P -> B is a **Strong Association Rule.**

**Disadvantages of Apriori Algorithm ?**

* Generation of itemsets is expensive(in both space and time)
* Support counting is expensive

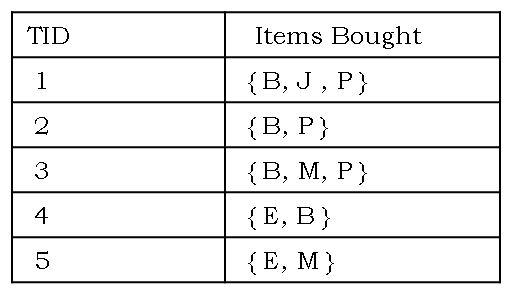
## What is FP Growth Algorithm ?

An efficient and scalable method to find frequent patterns. It allows [**frequent itemset**](http://athena.ecs.csus.edu/%7Ekhambamb/Association.html#frequentItem) discovery without candidate itemset generation.

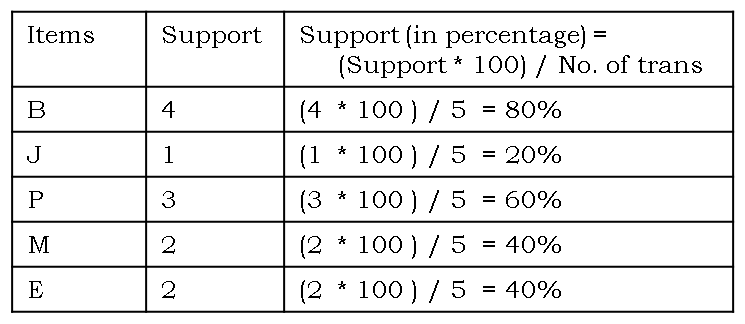
**Following are the steps for FP Growth Algorithm**

* Scan DB once, find frequent 1-itemset (single item pattern)
* Sort frequent items in frequency descending order, f-list
* Scan DB again, construct FP-tree
* Construct the conditional FP tree in the sequence of reverse order of F - List - generate frequent item set

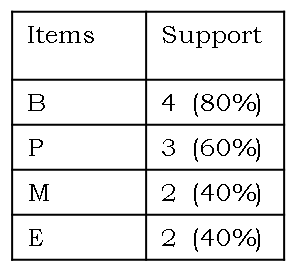
**Illustration:**Consider the below tansaction in which B = Bread, J = Jelly, P = Peanut Butter, M = Milk and E = Eggs. Given that minimum threshold support = 40% and minimum threshold confidence = 80% [13].



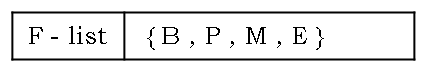
**Step-1:** Scan DB once, find frequent 1-itemset (single item in itemset)



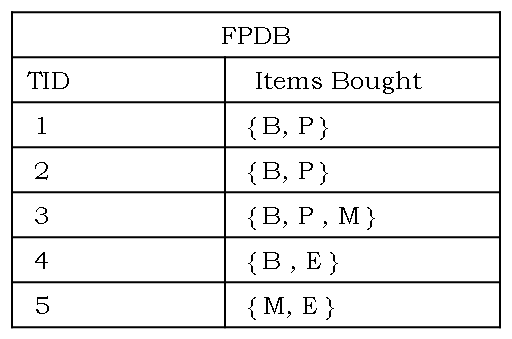
**Step-2:** As minimum threshold support = 40%, So in this step we will remove all the items that are bought less than 40% of support or support less than 2.



**Step-3:** Create a F -list in which frequent items are sorted in the descending order based on the [**support**](http://athena.ecs.csus.edu/%7Ekhambamb/Association.html#Support).

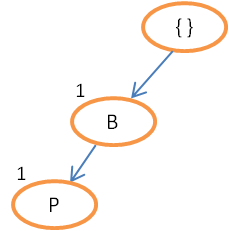


**Step-4:** Sort frequent items in transactions based on F-list. It is also known as FPDP.

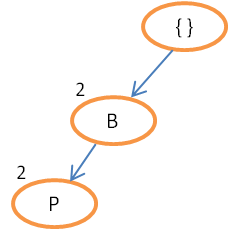


**Step-5:** Construct the FP tree

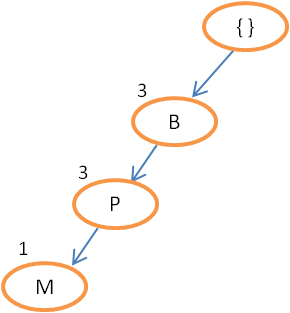
* Read transaction 1: {B,P} -> Create 2 nodes B and P. Set the path as null -> B -> P and the count of B and P as 1 as shown below :



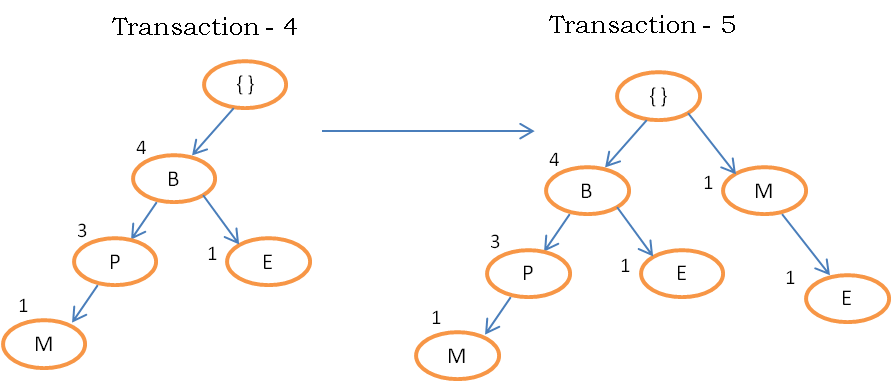
* Read transaction 2: {B,P} -> The path will be null -> B -> P. As transaction 1 and 2 share the same path. Set counts of B and P to 2.



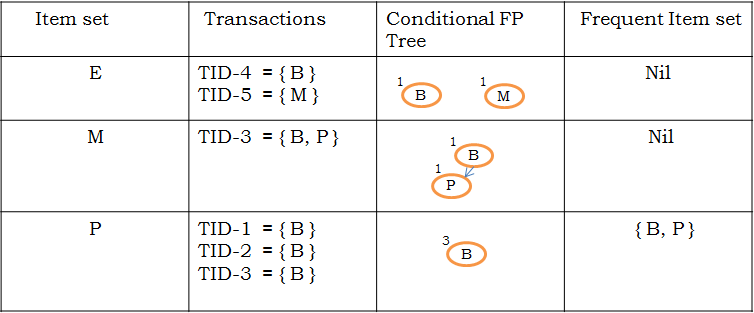
* Read transaction 3: {B,P,M} -> The path will be null -> B -> P -> M. As transaction 2 and 3 share the same path till node P. Therefore, set count of B and P as 3 and create node M having count 1.



* Continue until all the transactions are mapped to a path in FP-tree.



**Step-6:** Construct the conditional FP tree in the sequence of reverse order of F - List {E,M,P,B} and generate frequent item set. The conditional FP tree is sub tree which is built by considering the transactions of a particular item and then removing that item from all the transaction.



The above table has two items {B , P} that are bought together frequently.

As for items E and M, nodes in the conditional FP tree has a count(support) of 1 (less than minimum threshold support 2). Therefore frequent itemset are nil. In case of item P, node B in the conditional FP tree has a count(support) of 3 (satisfying minimum threshold support). Hence frequent itemset is generated by adding the item P to the B.

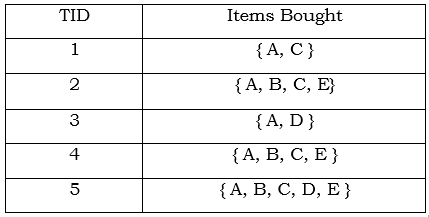
## Association Rule Generation :

Given the [**mininum threshold confidence**](http://athena.ecs.csus.edu/%7Ekhambamb/Association.html#Minimum), Generating association rules by going through all possible combinations of frequent item sets and pruning the rules according to confidence criterion.

**Following are the steps for strong Association Rule Generation:**

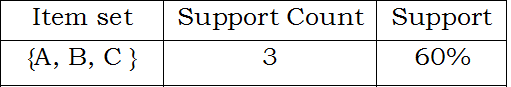
* Generate all nonempty subsets for each frequent itemset
* For every nonempty subset S of Itemset I , output of the rule:
  + S --> (I - S )
  + **If** support\_count (I) / support\_count (S) > = minimum confidence threshold **then** rule is a **strong Association Rule.**

**For Example** :Consider the below transaction:



Given that minimum threshold support = 60% (support count = 3) and minimum threshold confidence = 80% ( Confidence =

We have already generated the frequent item sets for this example



* Generate all nonempty subsets for each frequent itemset
  + For Itemset - { A, B, C } , all non empty subsets are {A,B}, {B,C}, {A,C}, {A}, {B}, {C}
* For every nonempty subset S of Itemset I , output of the rule:
  + S --> (I - S )
    - {A,B} -> {C}
    - {B,C} -> {A}
    - {A,C} -> {C}
    - {A} -> {B,C}
    - {B} -> {A,C}
    - {C} -> {A,B}
  + **If** support\_count (I) / support\_count (S) > = minimum confidence threshold **then** rule is a **strong Association Rule.**
    - {A,B} -> {C}, Confidence = 3/3 \* 100 = 100% - **Yes**, it is a strong association rules
    - {B,C} -> {A}, Confidence = 3/3 \* 100 = 100% - **Yes**, it is a strong association rules
    - {A,C} -> {C}, Confidence = 3/4 \* 100 = 80% - **Yes**, it is a strong association rules
    - {A} -> {B,C}, Confidence = 3/5 \* 100 = 60% - **No**, it is not a strong association rules
    - {B} -> {A,C}, Confidence = 3/3 \* 100 = 100% - **Yes**, it is a strong association rules
    - {C} -> {A,B}, Confidence = 3/4 \* 100 = 80% - **Yes**, it is a strong association rules

**3.2.1 The Apriori algorithm: Finding frequent itemsets**

Apriori is an influential algorithm for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties.

Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets. First, the set of frequent 1-itemsets is found. This set is denoted L1. L1 is used to find L2, the frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found. The finding of each Lk requires one full scan of the database.

To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the **Apriori property** is used to reduce the search space.

**The Apriori property.** All non-empty subsets of a frequent itemset must also be frequent.

By definition, if an itemset I does not satisfy the minimum support threshold, s, then I is not frequent, i.e., P(I) < s. If an item A is added to the itemset I, then the resulting itemset cannot occur more frequently than I. This property belongs to a special category of properties called anti-monotone in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well. It is called anti-monotone because the property is monotonic in the context of failing a test.

1. **The join step**: To find Lk, a set of candidate k-itemsets is generated by joining Lk-1 with itself. This set of candidates is denoted Ck. Let l1 and l2 be itemsets in Lk\_1. The notation li[j] refers to the jth item in li.

By convention, Apriori assumes that items within a transaction or itemset are sorted in increasing lexicographic order. It also ensures that no duplicates are generated.

1. **The prune step**: Ck is a superset of Lk, that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in Ck. A scan of the database to determine the count of each candidate in Ck would result in the determination of Lk. Ck can be huge, and so this could involve heavy computation.

To reduce the size of Ck, the Apriori property is used as follows. Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k-1)-subset of a candidate k-itemset is not in Lk-1, then the candidate cannot be frequent either and so can be removed from Ck. This subset testing can be done quickly by maintaining a hash tree of all frequent itemsets.

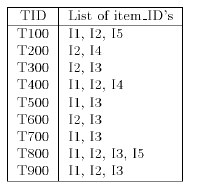


Fig. 3.2.1.1 Transactional data for an All Electronics branch

.

Let's look at a concrete example of Apriori, based on the All Electronics transaction database, D, of There are nine transactions in this database.

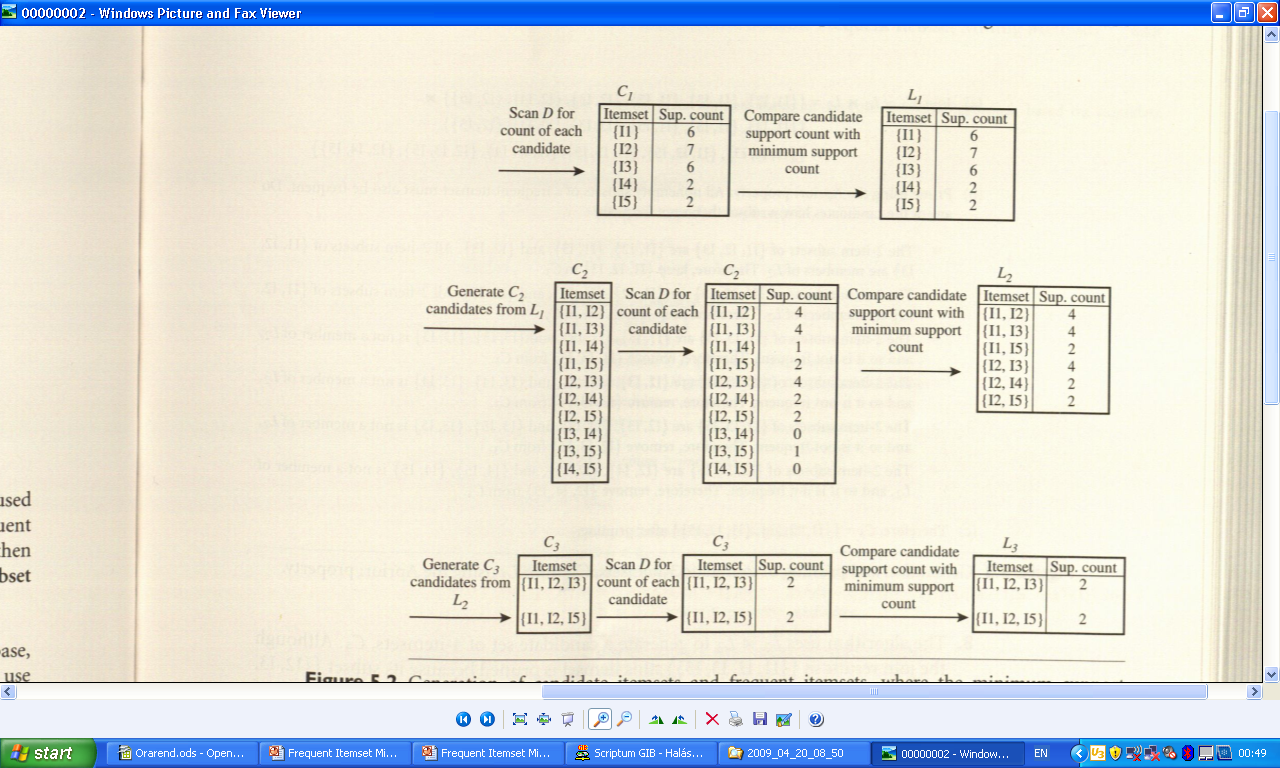
1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, C1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.

2. Suppose that the minimum transaction support count required is 2 (i.e., min sup = 2). The set of frequent 1-itemsets, L1, can then be determined. It consists of the candidate 1-itemsets having minimum support.

3. To discover the set of frequent 2-itemsets, L2, the algorithm uses L1×L1 to generate a candidate set of 2-itemsets, C2.

4. Next, the transactions in D are scanned and the support count of each candidate itemset in C2 is accumulated.

5. The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support.



6. The generation of the set of candidate 3-itemsets, C3. Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent. We therefore remove them from C3, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of D to determine L3.

7. The transactions in D are scanned in order to determine L3, consisting of those candidate 3-itemsets in C3 having minimum support.

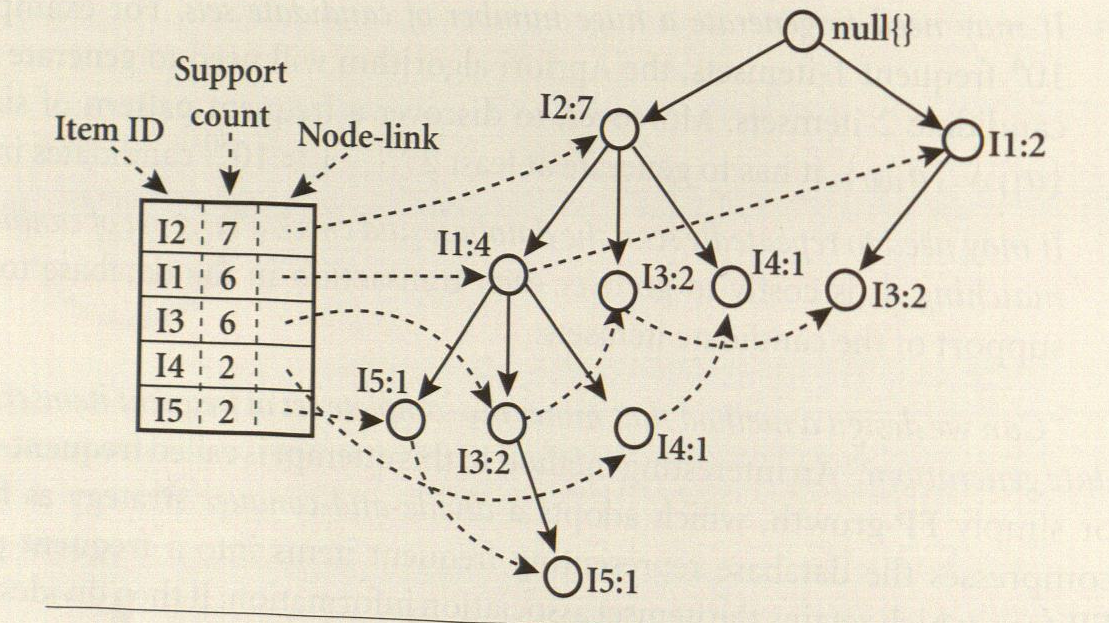
8. The algorithm uses L3×L3 to generate a candidate set of 4-itemsets, C4.

**FPGROWTH ALGORITHM**

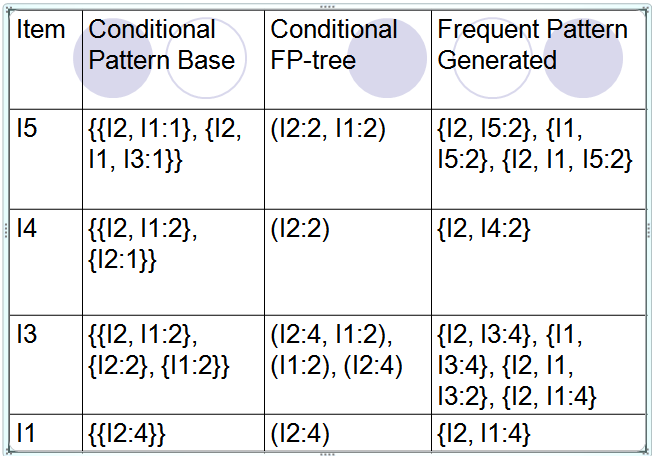
Mining Frequent Itemsets without candidate generation using fp growth algorithm:

* The candidate generate and test method
  + Reduces the size of candidates sets
  + Good performance
  + It may need to generate a huge number of candidate sets
  + It may need to repeatedly scan the database and check a large set of candidates by pattern matching

Frequent-pattern growth method(FP-growth)–frequent pattern tree(FP-tree):

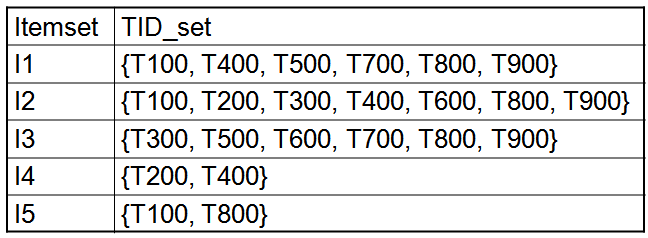
****

* I5
  + (I2, I1, I5:1)
  + (I2, I1, I3, I5:1)
* I5 is a suffix, so the two prefixes are
  + (I2, I1:1)
  + (I2, I1, I3:1)
* FP tree: (I2:2, I1:2), I3 is removed because <2
* The combinations of frequent pattenrs:
  + {I2,I5:2}
  + {I1,I5:2}
  + {I2, I1, I5:2}
  + For I4 exist 2 prefixes:
  + {{I2, I1:1},{I2:1}}
* Generation of the conditional FP-tree:
  + (I2:2)
* The frequent pattern: {I2, I1:2}

****

**Mining frequent itemsets using vertical data format:**

* **Transforming the horizontal data format of the transaction database D into a vertical data format:**

****

**Benefits of the FP-tree Structure**

* Completeness:
  + never breaks a long pattern of any transaction
  + preserves complete information for frequent pattern mining
* Compactness
  + reduce irrelevant information—infrequent items are gone
  + frequency descending ordering: more frequent items are more likely to be shared
  + never be larger than the original database (if not count node-links and counts)
  + Example: For Connect-4 DB, compression ratio could be over 100

**Mining Frequent Patterns Using FP-tree**

* General idea (divide-and-conquer)
  + Recursively grow frequent pattern path using the FP-tree
* Method
  + For each 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 to Mine FP-tree**

1. Construct conditional pattern base for each node in the FP-tree
2. Construct conditional FP-tree from each conditional pattern-base
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

# ****Compact Representation of Frequent Itemset****

## Introduction

What happens when you have a large market basket data with over a hundred items?   
The number of frequent itemsets grows exponentially and this in turn creates an issue with storage and it is for this purpose that alternative representations have been derived which reduce the initial set but can be used to generate all other frequent itemsets.  The Maximal and Closed Frequent Itemsets are two such representations that are subsets of the larger frequent itemset that will be discussed in this section.

## ****Maximal Frequent Itemset****

### ****Definition****

It is a frequent itemset for which none of its immediate supersets are frequent.

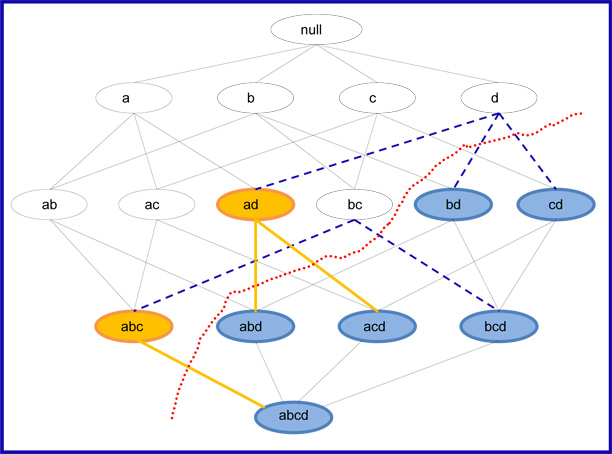
### ****Identification****

1. Examine the frequent itemsets that appear at the border between the infrequent and frequent itemsets.
2. Identify all of its immediate supersets.
3. If none of the immediate supersets are frequent, the itemset is maximal frequent.

### ****Illustration****

For instance consider the diagram shown below, the lattice is divided into two groups, red dashed line serves as the dermarcation, the itemsets above the line that are blank are frequent itemsets and the blue ones below the red dashed line are infrequent.

* In order to find the maximal frequent itemset, you first identify the frequent itemsets at the border namely d, bc, ad and abc.
* Then identify their immediate supersets,   
  the supersets for d, bc  are characterized by the blue dashed line and if you trace the lattice you notice that for d, there are three supersets and one of them, ad is frequent and this can’t be maximal frequent,   
  for bc there are two supersets namely abc and bcd abc is frequent and so bc is NOT maximal frequent.
* The supersets for ad and abc are characterized by a solid orange line, the superset for abc is abcd and being that it is infrequent, abcd is maximal frequent. For ad, there are two supersets abd and acd, both of them are infrequent and so **ad** is also maximal frequent.



The slideshow below shows a dynamic illustration of the example given above

## ****Closed Frequent Itemset****

### ****Definition:****

It is a frequent itemset that is both closed and its support is greater than or equal to minsup.   
An itemset is closed in a data set if there exists no superset that has the same support count as this original itemset.

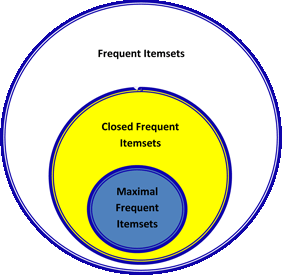
### ****Identification****

1. First identify all frequent itemsets.
2. Then from this group find those that are closed by checking to see if there exists a superset that has the same support as the frequent itemset, if there is, the itemset is disqualified, but if none can be found, the itemset is closed.  
   An alternative method is to first identify the closed itemsets and then use the minsup to determine which ones are frequent.

### ****Illustration****Closed Frequent Itemset

The lattice diagram above shows the maximal, closed and frequent itemsets. The itemsets that are circled with blue are the frequent itemsets. The itemsets that are circled with the thick blue are the closed frequent itemsets. The itemsets that are circled with the thick blue and have the yellow fill are the maximal frequent itemsets. In order to determine which of the frequent itemsets are closed, all you have to do is check to see if they have the same support as their supersets, if they do they are not closed.   
For example **ad** is a frequent itemset but has the same support as **abd** so it is NOT a closed frequent itemset; c on the other hand is a closed frequent itemset because all of its supersets, ac, bc, and cd have supports that are less than 3.   
As you can see there are a total of 9 frequent itemsets, 4 of them are closed frequent itemsets and out of these 4, 2 of them are maximal frequent itemsets. This brings us to the relationship between the three representations of frequent itemsets.

## ****Relationship between Frequent Itemset Representations****

  
In conclusion, it is important to point out the relationship between frequent itemsets, closed frequent itemsets and maximal frequent itemsets. As mentioned earlier closed and maximal frequent itemsets are subsets of frequent itemsets but maximal frequent itemsets are a more compact representation because it is a subset of closed frequent itemsets. The diagram to the right shows the relationship between these three types of itemsets. Closed frequent itemsets are more widely used than maximal frequent itemset because when efficiency is more important that space, they provide us with the support of the subsets so no additional pass is needed to find this information.

**ASSOCIATION ANALYSIS ADVANCED CONCEPTS:**

**CONTINUOUS AND CATEGORICAL ATTRIBUTES:** How to apply association analysis formulation to non-asymmetric binary variables?

****

Example of Association Rule:

{Number of Pages ∈[5,10) ∧ (Browser=Mozilla)} → {Buy = No}

Handling Categorical Attributes:

1.Transform categorical attribute into asymmetric binary variables 2.Introduce a new “item” for each distinct attribute-value pair

Example: replace Browser Type attribute with

Browser Type = Internet Explorer

Browser Type = Mozilla

Browser Type = Mozilla

**3.Potential Issues**

What if attribute has many possible values

Example: attribute country has more than 200 possible values

Many of the attribute values may have very low support

Potential solution: Aggregate the low-support attribute values

What if distribution of attribute values is highly skewed

Example: 95% of the visitors have Buy = No

Most of the items will be associated with (Buy=No) item

Potential solution: drop the highly frequent items

Different kinds of rules:

Age∈[21,35) ∧ Salary∈[70k,120k) → Buy

Salary∈[70k,120k) ∧ Buy → Age: μ=28, σ=4

**Different methods:**

Discretization-based

Statistics-based

Non-discretization based

minApriori

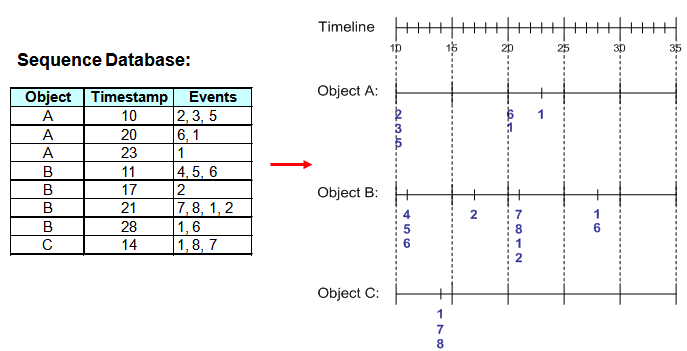
Use discretization

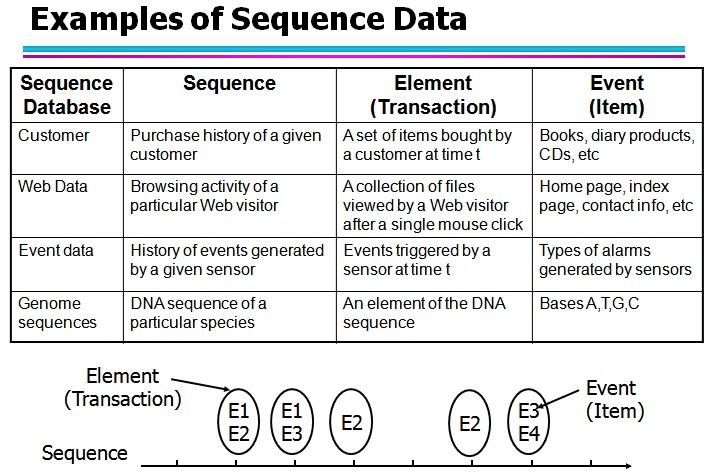
Unsupervised:

Equal-width binning

Equal-depth binning

Clustering

**SequenceData:**

****

**Formal Definition of a Sequence:** A sequence is an ordered list of elements (transactions).

s = < e1 e2 e3 … >

Each element contains a collection of events (items)

ei = {i1, i2, …, ik}

Each element is attributed to a specific time or location

Length of a sequence, |s|, is given by the number of elements of the sequence

A k-sequence is a sequence that contains k events (items)

**Web sequence:**

< {Homepage} {Electronics} {Digital Cameras} {Canon Digital Camera} {Shopping Cart} {Order Confirmation} {Return to Shopping} >

Sequence of initiating events causing the nuclear accident at 3-mile Island:  
(http://stellar-one.com/nuclear/staff\_reports/summary\_SOE\_the\_initiating\_event.htm)

<{clogged resin} {outlet valve closure} {loss of feedwater}   
{condenser polisher outlet valve shut} {booster pumps trip}   
{main waterpump trips} {main turbine trips} {reactor pressure increases}>

Sequence of books checked out at a library:

<{Fellowship of the Ring} {The Two Towers} {Return of the King}>

**Sequential Pattern Mining: Definition**

Given:

* + a database of sequences
  + a user-specified minimum support threshold, *minsup*

Task:

* + Find all subsequences with support ≥ *minsup*

Given a sequence: <{a b} {c d e} {f} {g h i}>

* + Examples of subsequences:

<{a} {c d} {f} {g} >, < {c d e} >, < {b} {g} >, etc.

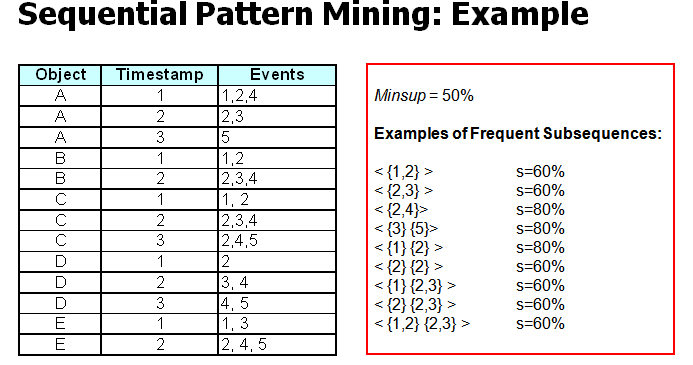
How many k-subsequences can be extracted from a given n-sequence?

<{a b} {c d e} {f} {g h i}> n = 9



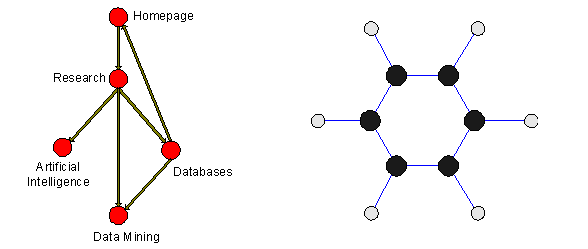
k=4: Y \_ \_ Y Y \_ \_ \_ Y

<{a} {d e} {i}>

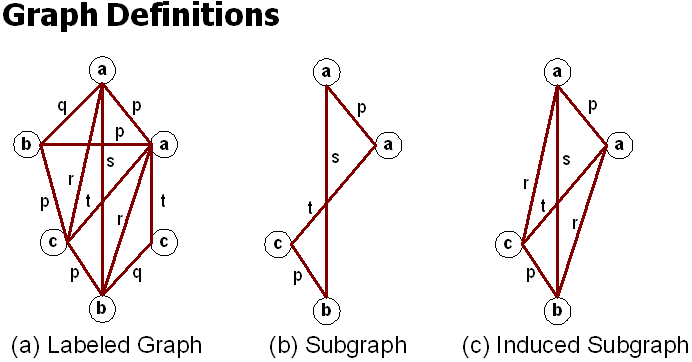


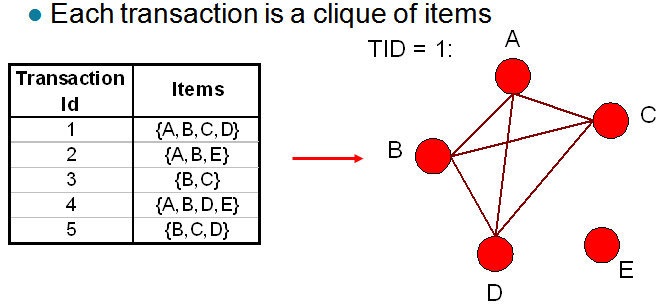
**Frequent Subgraph Mining:** Extend association rule mining to finding frequent subgraphs

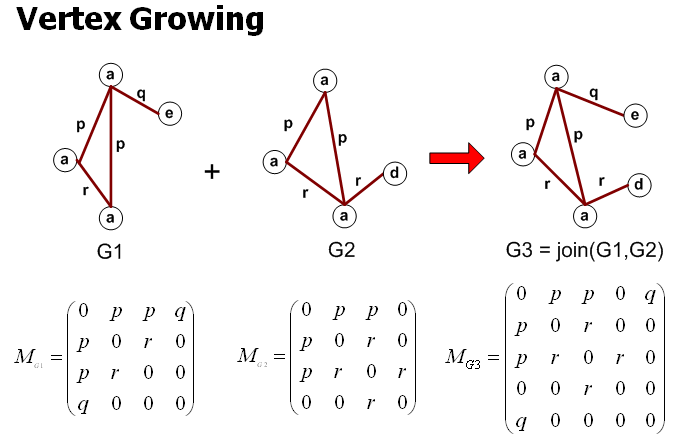
Useful for Web Mining, computational chemistry, bioinformatics, spatial data sets, etc.

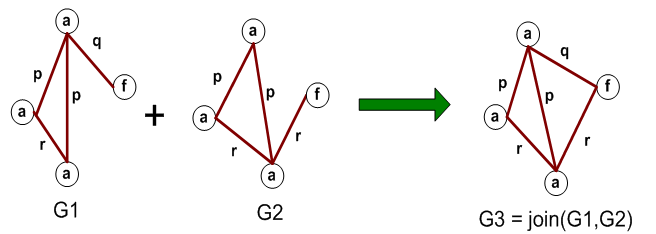
****

* A graph is said to be *connected* if there is a path between every pair of vertices
* A graph Gs (Vs, Es) is a *subgraph* of another graph G(V, E) iff
  + Vs is subset of V and Es is subset of E
* Two graphs G1(V1, E1) and G2(V2, E2) are *isomorphic* if they are topologically identical
  + There is a mapping from V1 to V2 such that each edge in E1 is mapped to a single edge in E2 and vice-versa

****

**Representing Transactions as Graphs:**





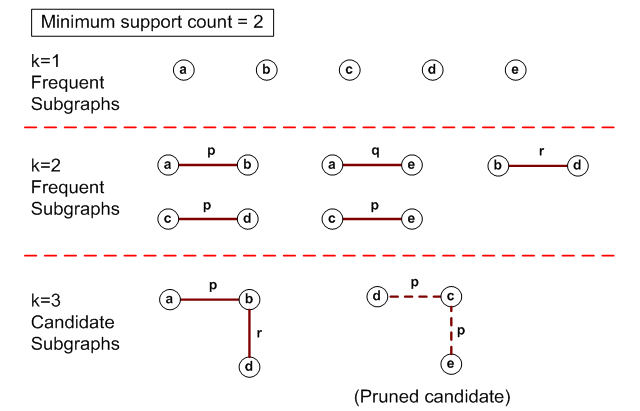
**Apriori-like Algorithm:** **Find frequent 1-subgraphs**

**Repeat**

* + **Candidate generation**
    - **Use frequent (*k-1*)-subgraphs to generate candidate *k*-subgraph**
  + **Candidate pruning**
    - **Prune candidate subgraphs that contain infrequent   
      (*k-1*)-subgraphs**
  + **Support counting**
    - **Count the support of each remaining candidate**
  + **Eliminate candidate *k*-subgraphs that are infrequent**

**Example: Dataset**

****

**Example:**