

# **Data Mining**

## **Association Rules: Advanced Concepts and Algorithms**

Lecture Notes for Chapter 7

Introduction to Data Mining  
by  
Tan, Steinbach, Kumar

(modified by Predrag Radivojac, 2018)

# Continuous and Categorical Attributes

How to apply association analysis formulation to non-symmetric binary variables?

Session Id	Country	Session Length (sec)	Number of Web Pages viewed	Gender	Browser Type	Buy
1	USA	982	8	Male	IE	No
2	China	811	10	Female	Chrome	No
3	USA	2125	45	Female	Firefox	Yes
4	Germany	596	4	Male	IE	Yes
5	Australia	123	9	Male	Firefox	No
...	...	...	...	...	...	...

**Example of Association Rule:**

$\{\text{Number of Pages} \in [5, 10) \wedge (\text{Browser} = \text{Firefox})\} \rightarrow \{\text{Buy} = \text{No}\}$

# Handling Categorical Attributes

- Transform categorical attribute into asymmetric binary variables
- Introduce a new “item” for each distinct attribute-value pair
  - Example: replace Browser Type attribute with
    - ◆ Browser Type = Internet Explorer
    - ◆ Browser Type = Firefox
    - ◆ Browser Type = Chrome

# Handling Categorical Attributes

## ● 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  $\text{Buy} = \text{No}$
  - ◆ Most of the items will be associated with  $(\text{Buy} = \text{No})$  item
    - *Potential solution*: drop the highly frequent items
- Transaction width increases => more computation

# Handling Continuous Attributes

- Different kinds of rules:

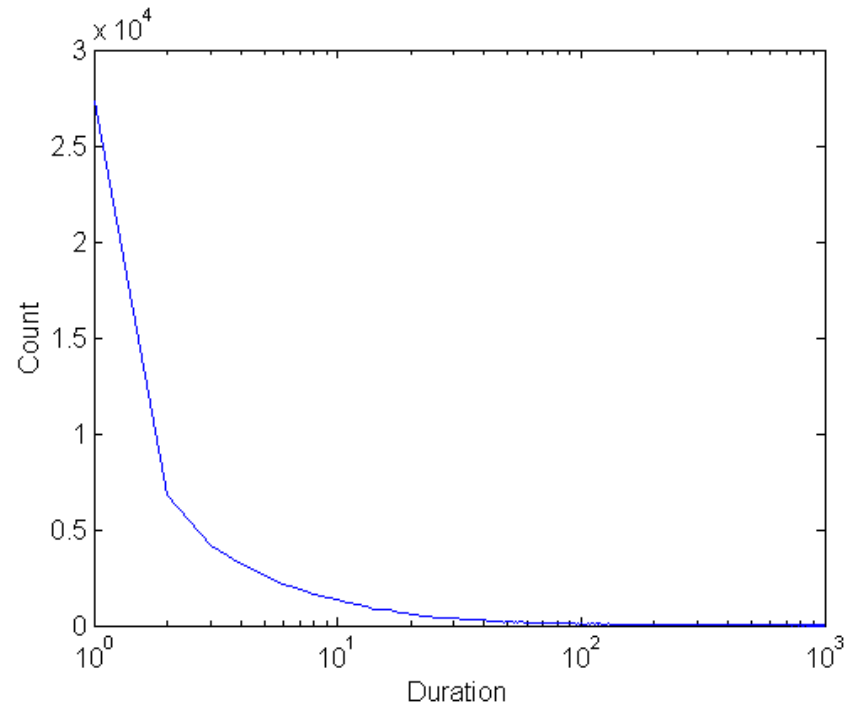
- $\text{Age} \in [21, 35) \wedge \text{Salary} \in [70\text{k}, 120\text{k}) \rightarrow \text{Buy}$
- $\text{Salary} \in [70\text{k}, 120\text{k}) \wedge \text{Buy} \rightarrow \text{Age}: \mu=28, \sigma=4$

- Different methods:

- Discretization-based
- Statistics-based
- Non-discretization based
  - ◆ minApriori

# Handling Continuous Attributes

- Use discretization
- Unsupervised:
  - Equal-width binning
  - Equal-depth binning
  - Clustering
- Supervised:



Attribute values,  $v$

Class	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_6$	$v_7$	$v_8$	$v_9$
Anomalous	0	0	20	10	20	0	0	0	0
Normal	150	100	0	0	0	100	100	150	100

bin<sub>1</sub> bin<sub>2</sub> bin<sub>3</sub>

# Discretization Issues

- Size of the discretized intervals affect support & confidence

`{Refund = No, (Income = $51,250)} → {Cheat = No}`

`{Refund = No, (60K ≤ Income ≤ 80K)} → {Cheat = No}`

`{Refund = No, (0K ≤ Income ≤ 1B)} → {Cheat = No}`

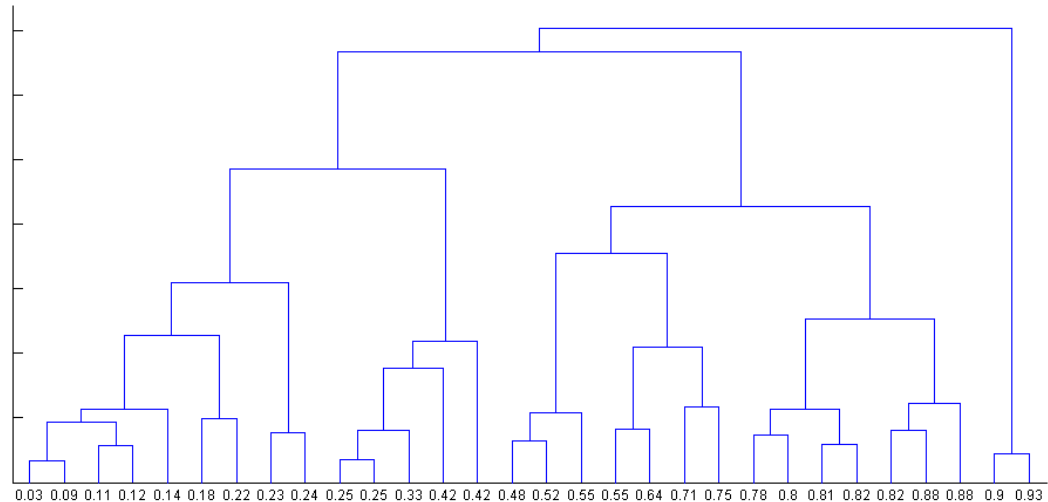
- If intervals too small
  - ◆ may not have enough support
- If intervals too large
  - ◆ may not have enough confidence

- Potential solution: use all possible intervals

# Discretization Issues

- Execution time

- If intervals contain  $n$  values, there are on average  $O(n^2)$  possible ranges



- Too many rules

$\{\text{Refund} = \text{No}, (\text{Income} = \$51,250)\} \rightarrow \{\text{Cheat} = \text{No}\}$

$\{\text{Refund} = \text{No}, (51\text{K} \leq \text{Income} \leq 52\text{K})\} \rightarrow \{\text{Cheat} = \text{No}\}$

$\{\text{Refund} = \text{No}, (50\text{K} \leq \text{Income} \leq 60\text{K})\} \rightarrow \{\text{Cheat} = \text{No}\}$

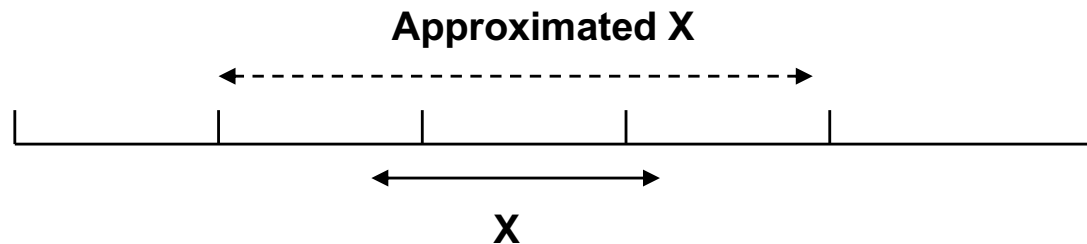


# Approach by Srikant & Agrawal

- Preprocess the data
  - Discretize attribute using equi-depth partitioning
    - ◆ Use *partial completeness measure* to determine number of partitions
    - ◆ Merge adjacent intervals as long as support is less than max-support
- Apply existing association rule mining algorithms
- Determine interesting rules in the output

# Approach by Srikant & Agrawal

- Discretization will lose information



- Use *partial completeness measure* to determine how much information is lost

C: frequent itemsets obtained by considering all ranges of attribute values

P: frequent itemsets obtained by considering all ranges over the partitions

P is *K-complete* w.r.t C if  $P \subseteq C$ , and  $\forall X \in C, \exists X' \in P$  such that:

1.  $X'$  is a generalization of  $X$  and  $\text{support}(X') \leq K \times \text{support}(X)$  ( $K \geq 1$ )
2.  $\forall Y \subseteq X, \exists Y' \subseteq X'$  such that  $\text{support}(Y') \leq K \times \text{support}(Y)$

Given  $K$  (*partial completeness level*), can determine number of intervals ( $N$ )

# Interestingness Measure

$\{\text{Refund} = \text{No}, (\text{Income} = \$51,250)\} \rightarrow \{\text{Cheat} = \text{No}\}$

$\{\text{Refund} = \text{No}, (51\text{K} \leq \text{Income} \leq 52\text{K})\} \rightarrow \{\text{Cheat} = \text{No}\}$

$\{\text{Refund} = \text{No}, (50\text{K} \leq \text{Income} \leq 60\text{K})\} \rightarrow \{\text{Cheat} = \text{No}\}$

- Given an itemset:  $Z = \{z_1, z_2, \dots, z_k\}$  and its generalization  $Z' = \{z_1', z_2', \dots, z_k'\}$

$P(Z)$ : support of  $Z$

$E_{Z'}(Z)$ : expected support of  $Z$  based on  $Z'$

$$E_{Z'}(Z) = \frac{P(z_1)}{P(z_1')} \times \frac{P(z_2)}{P(z_2')} \times \dots \times \frac{P(z_k)}{P(z_k')} \times P(Z')$$

- $Z$  is  $R$ -interesting w.r.t.  $Z'$  if  $P(Z) \geq R \times E_{Z'}(Z)$

# Interestingness Measure

- For  $S: X \rightarrow Y$ , and its generalization  $S': X' \rightarrow Y'$

$P(Y|X)$ : confidence of  $X \rightarrow Y$

$P(Y'|X')$ : confidence of  $X' \rightarrow Y'$

$E_{S'}(Y|X)$ : expected support of  $Z$  based on  $Z'$

$$E(Y | X) = \frac{P(y_1)}{P(y_1')} \times \frac{P(y_2)}{P(y_2')} \times \dots \times \frac{P(y_k)}{P(y_k')} \times P(Y' | X')$$

- Rule  $S$  is  $R$ -interesting w.r.t its ancestor rule  $S'$  if
  - Support,  $P(S) \geq R \times E_{S'}(S)$  or
  - Confidence,  $P(Y|X) \geq R \times E_{S'}(Y|X)$

# Statistics-based Methods

- Example:

`Browser = Firefox  $\wedge$  Buy = Yes  $\rightarrow$  Age:  $\mu=23$`

- Rule consequent consists of a continuous variable, characterized by their statistics
  - mean, median, standard deviation, etc.
- Approach:
  - Withhold the target variable from the rest of the data
  - Apply existing frequent itemset generation on the rest of the data
  - For each frequent itemset, compute the descriptive statistics for the corresponding target variable
    - ◆ Frequent itemset becomes a rule by introducing the target variable as rule consequent
  - Apply statistical test to determine interestingness of the rule

# Statistics-based Methods

- How to determine whether an association rule is interesting?

- Compare the statistics for segment of population covered by the rule vs segment of population not covered by the rule:

$$A \Rightarrow B: \mu \quad \text{versus} \quad \bar{A} \Rightarrow B: \mu'$$

- Statistical hypothesis testing:

- ◆ Null hypothesis:  $H_0: \mu' = \mu + \Delta$
- ◆ Alternative hypothesis:  $H_1: \mu' > \mu + \Delta$
- ◆ Z has zero mean and variance 1 under null hypothesis

$$Z = \frac{\mu' - \mu - \Delta}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

# Statistics-based Methods

- Example:

$r: \text{Browser} = \text{Firefox} \wedge \text{Buy} = \text{Yes} \rightarrow \text{Age}: \mu=23$

- Rule is interesting if difference between  $\mu$  and  $\mu'$  is greater than 5 years (i.e.,  $\Delta = 5$ )
- For  $r$ , suppose  $n_1 = 50, s_1 = 3.5$
- For  $r'$  (complement):  $n_2 = 250, s_2 = 6.5$

$$Z = \frac{\mu' - \mu - \Delta}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} = \frac{30 - 23 - 5}{\sqrt{\frac{3.5^2}{50} + \frac{6.5^2}{250}}} = 3.11$$

- For 1-sided test at 95% confidence level, critical Z-value for rejecting null hypothesis is 1.64.
- Since Z is greater than 1.64,  $r$  is an interesting rule

# Min-Apriori (Han et al.)

Document-term matrix:

TID	W1	W2	W3	W4	W5
D1	2	2	0	0	1
D2	0	0	1	2	2
D3	2	3	0	0	0
D4	0	0	1	0	1
D5	1	1	1	0	2

**Example:**

$W_1$  and  $W_2$  tend to appear together in the same document



# Min-Apriori

- Data contains only continuous attributes of the same “type”
  - e.g., frequency of words in a document

TID	W1	W2	W3	W4	W5
D1	2	2	0	0	1
D2	0	0	1	2	2
D3	2	3	0	0	0
D4	0	0	1	0	1
D5	1	1	1	0	2

- Potential solution:
  - Convert into 0/1 matrix and then apply existing algorithms
    - ◆ lose word frequency information
  - Discretization does not apply as users want association among words not ranges of words

# Min-Apriori

- How to determine the support of a word?
  - If we simply sum up its frequency, support count will be greater than total number of documents!
    - ◆ Normalize the word vectors – e.g., using  $L_1$  norm
    - ◆ Each word has a support equals to 1.0

TID	W1	W2	W3	W4	W5
D1	2	2	0	0	1
D2	0	0	1	2	2
D3	2	3	0	0	0
D4	0	0	1	0	1
D5	1	1	1	0	2

Normalize



TID	W1	W2	W3	W4	W5
D1	0.40	0.33	0.00	0.00	0.17
D2	0.00	0.00	0.33	1.00	0.33
D3	0.40	0.50	0.00	0.00	0.00
D4	0.00	0.00	0.33	0.00	0.17
D5	0.20	0.17	0.33	0.00	0.33

# Min-Apriori

- New definition of support:

$$\text{sup}(C) = \sum_{i \in T} \min_{j \in C} D(i, j)$$

TID	W1	W2	W3	W4	W5
D1	0.40	0.33	0.00	0.00	0.17
D2	0.00	0.00	0.33	1.00	0.33
D3	0.40	0.50	0.00	0.00	0.00
D4	0.00	0.00	0.33	0.00	0.17
D5	0.20	0.17	0.33	0.00	0.33

**Example:**

$$\begin{aligned} &\text{Sup}(W_1, W_2, W_3) \\ &= 0 + 0 + 0 + 0 + 0.17 \\ &= 0.17 \end{aligned}$$

# Anti-monotone property of Support

TID	W1	W2	W3	W4	W5
D1	0.40	0.33	0.00	0.00	0.17
D2	0.00	0.00	0.33	1.00	0.33
D3	0.40	0.50	0.00	0.00	0.00
D4	0.00	0.00	0.33	0.00	0.17
D5	0.20	0.17	0.33	0.00	0.33

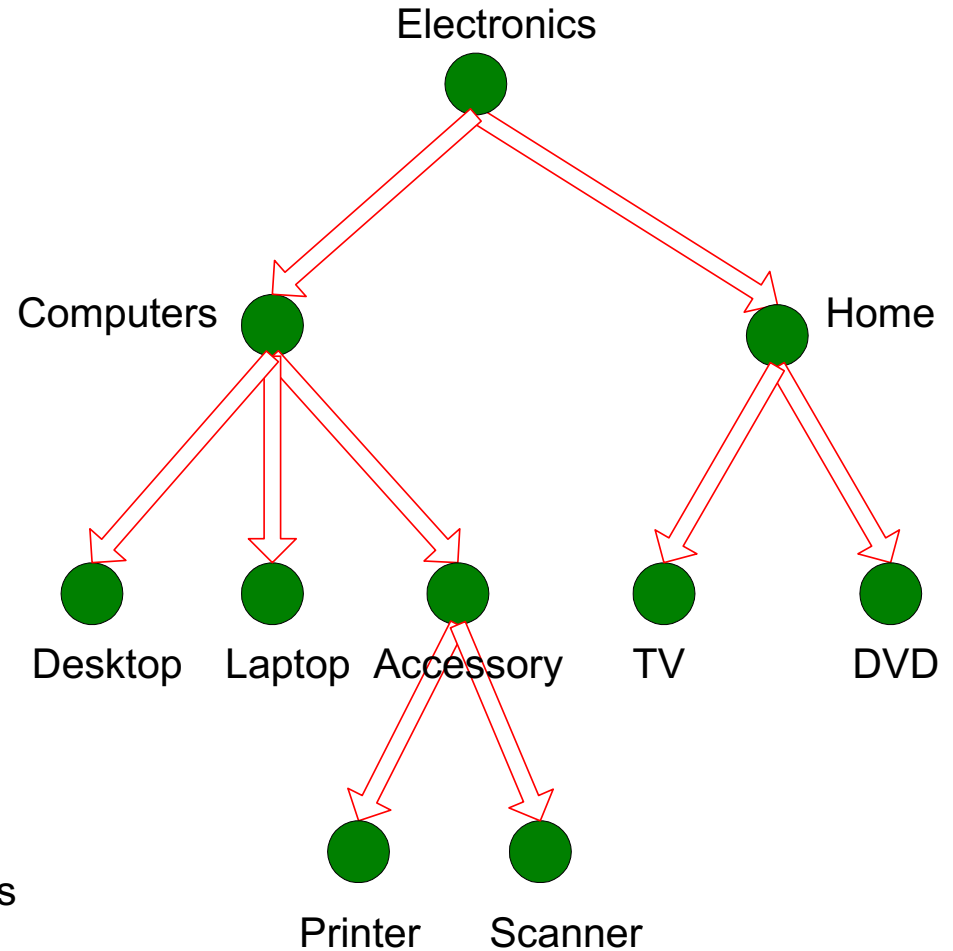
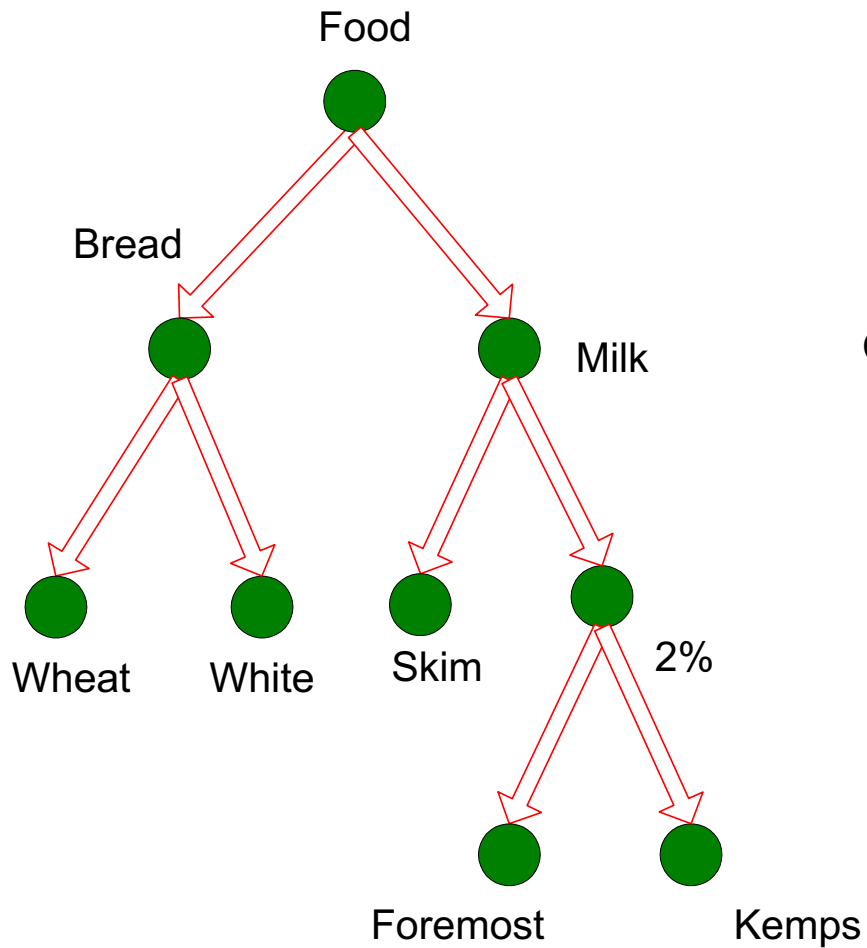
**Example:**

$$\text{Sup}(W_1) = 0.4 + 0 + 0.4 + 0 + 0.2 = 1$$

$$\text{Sup}(W_1, W_2) = 0.33 + 0 + 0.4 + 0 + 0.17 = 0.9$$

$$\text{Sup}(W_1, W_2, W_3) = 0 + 0 + 0 + 0 + 0.17 = 0.17$$

# Multi-level Association Rules



# Multi-level Association Rules

- Why should we incorporate concept hierarchy?
  - Rules at lower levels may not have enough support to appear in any frequent itemsets
  - Rules at lower levels of the hierarchy are overly specific; e.g.,

skim milk → white bread

2% milk → wheat bread

skim milk → wheat bread

are indicative of association between milk and bread

# Multi-level Association Rules

- How do support and confidence vary as we traverse the concept hierarchy?
  - If  $X$  is the parent item for both  $X_1$  and  $X_2$ , then
$$\sigma(X) \geq \sigma(X_1) + \sigma(X_2)$$
  - If
$$\sigma(X_1 \cup Y_1) \geq \text{minsup},$$
and  $X$  is parent of  $X_1$ ,  $Y$  is parent of  $Y_1$ then
$$\sigma(X \cup Y_1) \geq \text{minsup}, \sigma(X_1 \cup Y) \geq \text{minsup}$$
$$\sigma(X \cup Y) \geq \text{minsup}$$
  - If
$$\text{conf}(X_1 \Rightarrow Y_1) \geq \text{minconf},$$
then
$$\text{conf}(X_1 \Rightarrow Y) \geq \text{minconf}$$

# Multi-level Association Rules

- Approach 1:

- Extend current association rule formulation by augmenting each transaction with higher level items

Original Transaction:

`{skim milk, wheat bread}`

Augmented Transaction:

`{skim milk, wheat bread, milk, bread, food}`

- Issues:

- Items that reside at higher levels have much higher support counts
  - ◆ if support threshold is low, too many frequent patterns involving items from the higher levels
- Increased dimensionality of the data



# Multi-level Association Rules

- Approach 2:

- Generate frequent patterns at highest level first
- Then, generate frequent patterns at the next highest level, and so on

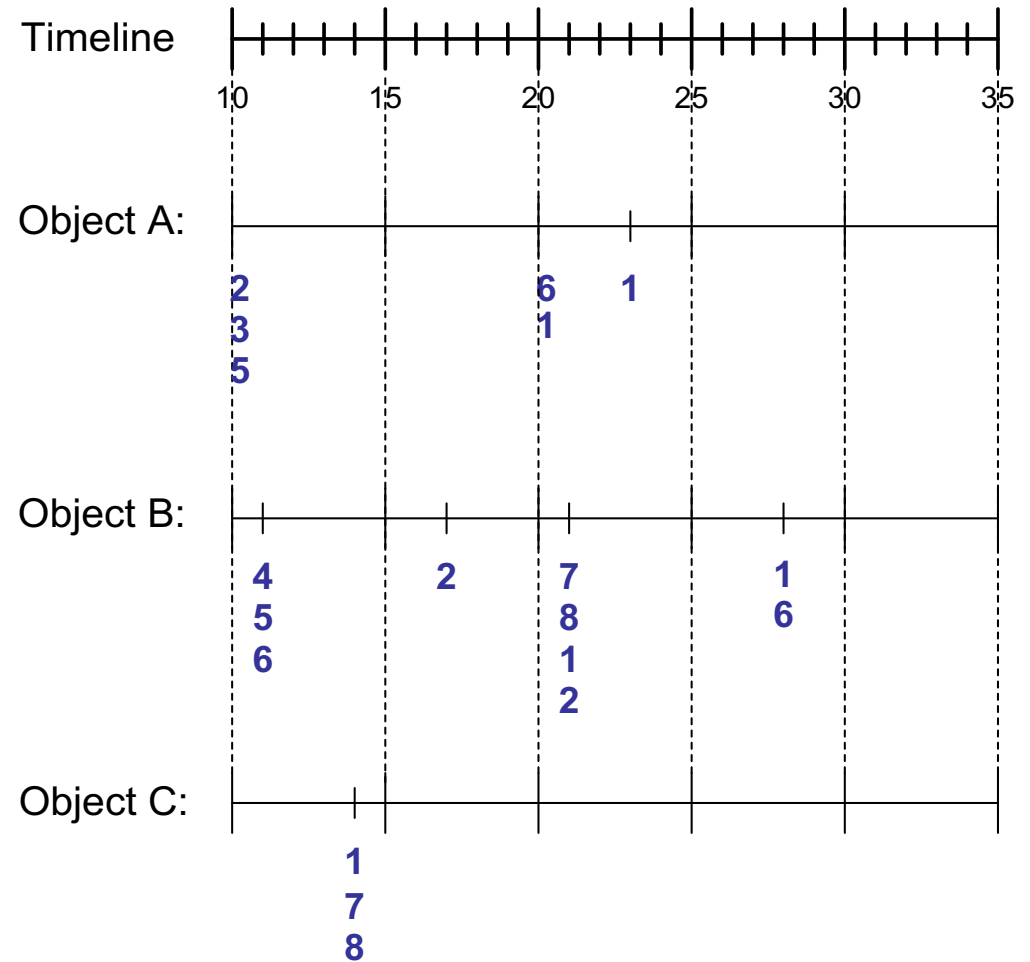
- Issues:

- I/O requirements will increase dramatically because we need to perform more passes over the data
- May miss some potentially interesting cross-level association patterns

# Sequential Patterns

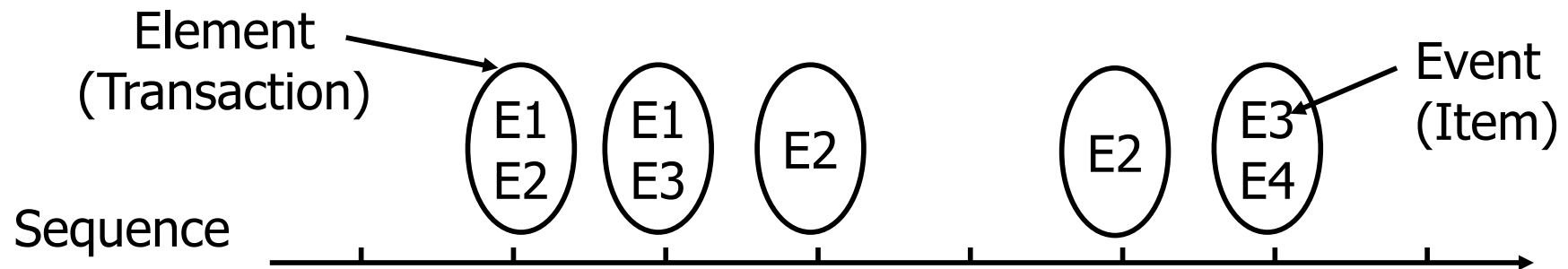
Sequence Database:

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



# Examples of Sequence Data

Sequence Database	Sequence	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor after a single mouse click	Home page, index page, contact info, etc
Event data	History of events generated by a given sensor	Events triggered by a sensor at time t	Types of alarms generated by sensors
Genome sequences	DNA sequence of a particular species	An element of the DNA sequence	Bases A,T,G,C



# Formal Definition of a Sequence

- A sequence is an ordered list of elements (transactions)

$$S = \langle e_1 \ e_2 \ e_3 \ \dots \rangle$$

- Each element contains a collection of events (items)

$$e_i = \{ i_1, \ i_2, \ \dots, \ i_k \}$$

- 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)

# Examples of Sequences

- 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:

```
< {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} >
```

# Formal Definition of a Subsequence

- A sequence  $\langle a_1 a_2 \dots a_n \rangle$  is contained in another sequence  $\langle b_1 b_2 \dots b_m \rangle$  ( $m \geq n$ ) if there exist integers  $i_1 < i_2 < \dots < i_n$  such that  $a_1 \subseteq b_{i_1}$ ,  $a_2 \subseteq b_{i_2}$ , ...,  $a_n \subseteq b_{i_n}$

Data sequence	Subsequence	Contained?
$\langle \{2,4\} \{3,5,6\} \{8\} \rangle$	$\langle \{2\} \{3,5\} \rangle$	
$\langle \{1,2\} \{3,4\} \rangle$	$\langle \{1\} \{2\} \rangle$	
$\langle \{2,4\} \{2,4\} \{2,5\} \rangle$	$\langle \{2\} \{4\} \rangle$	

- The *support* of a subsequence  $w$  is defined as the fraction of data sequences that contain  $w$
- A *sequential pattern* is a frequent subsequence (i.e., a subsequence whose support is  $\geq \text{minsup}$ )

# Formal Definition of a Subsequence

- A sequence  $\langle a_1 a_2 \dots a_n \rangle$  is contained in another sequence  $\langle b_1 b_2 \dots b_m \rangle$  ( $m \geq n$ ) if there exist integers  $i_1 < i_2 < \dots < i_n$  such that  $a_1 \subseteq b_{i_1}$ ,  $a_2 \subseteq b_{i_2}$ , ...,  $a_n \subseteq b_{i_n}$

Data sequence	Subsequence	Contained?
$\langle \{2,4\} \{3,5,6\} \{8\} \rangle$	$\langle \{2\} \{3,5\} \rangle$	Yes
$\langle \{1,2\} \{3,4\} \rangle$	$\langle \{1\} \{2\} \rangle$	No
$\langle \{2,4\} \{2,4\} \{2,5\} \rangle$	$\langle \{2\} \{4\} \rangle$	Yes

- The *support* of a subsequence  $w$  is defined as the fraction of data sequences that contain  $w$
- A *sequential pattern* is a frequent subsequence (i.e., a subsequence whose support is  $\geq \text{minsup}$ )

# Sequential Pattern Mining: Definition

- Given:

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

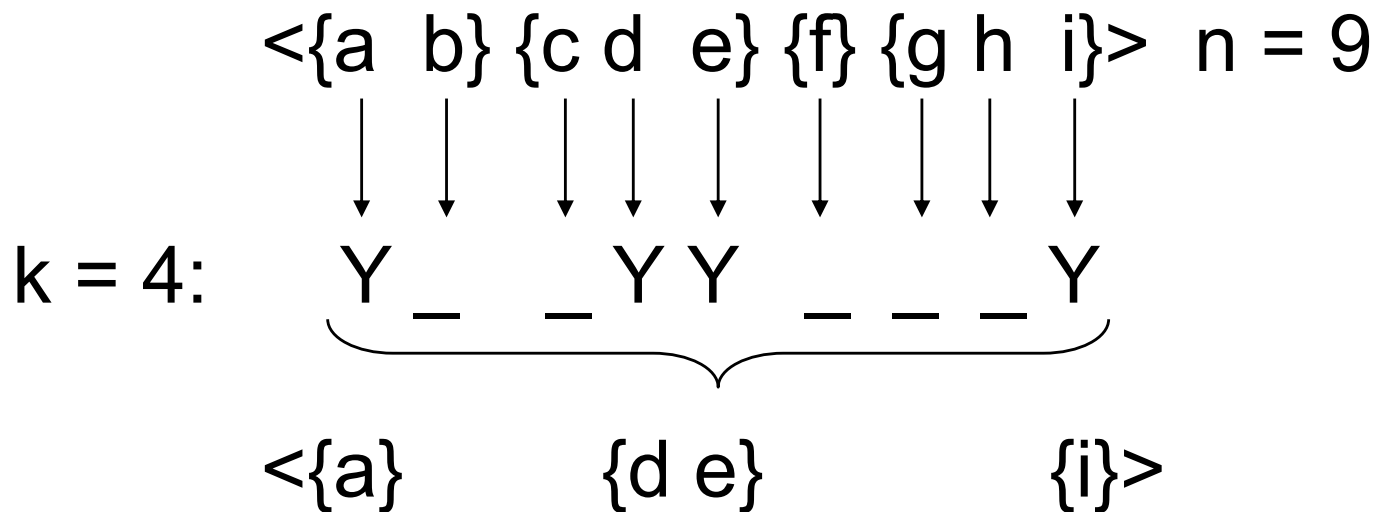
- Task:

- Find all subsequences with support  $\geq \textit{minsup}$



# Sequential Pattern Mining: Challenge

- Given a sequence:  $\langle \{a\} \{b\} \{c\} \{d\} \{e\} \{f\} \{g\} \{h\} \{i\} \rangle$ 
  - Examples of subsequences:  
 $\langle \{a\} \{c\} \{d\} \{f\} \{g\} \rangle$ ,  $\langle \{c\} \{d\} \{e\} \rangle$ ,  $\langle \{b\} \{g\} \rangle$ , etc.
- How many  $k$ -subsequences can be extracted from a given  $n$ -sequence?



Answer :

$$\binom{n}{k} = \binom{9}{4} = 126$$

# Sequential Pattern Mining: Example

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

*Minsup* = 50%

## Examples of Frequent Subsequences:

$\langle \{1,2\} \rangle$	$s = 60\%$
$\langle \{2,3\} \rangle$	$s = 60\%$
$\langle \{2,4\} \rangle$	$s = 80\%$
$\langle \{3\} \{5\} \rangle$	$s = 80\%$
$\langle \{1\} \{2\} \rangle$	$s = 80\%$
$\langle \{2\} \{2\} \rangle$	$s = 60\%$
$\langle \{1\} \{2,3\} \rangle$	$s = 60\%$
$\langle \{2\} \{2,3\} \rangle$	$s = 60\%$
$\langle \{1,2\} \{2,3\} \rangle$	$s = 60\%$

# Extracting Sequential Patterns

- Given  $n$  events:  $i_1, i_2, i_3, \dots, i_n$
- Candidate 1-subsequences:  
 $\langle \{i_1\} \rangle, \langle \{i_2\} \rangle, \langle \{i_3\} \rangle, \dots, \langle \{i_n\} \rangle$
- Candidate 2-subsequences:  
 $\langle \{i_1, i_2\} \rangle, \langle \{i_1, i_3\} \rangle, \dots, \langle \{i_1\} \{i_1\} \rangle, \langle \{i_1\} \{i_2\} \rangle, \dots, \langle \{i_n\} \{i_n\} \rangle$
- Candidate 3-subsequences:  
 $\langle \{i_1, i_2, i_3\} \rangle, \langle \{i_1, i_2, i_4\} \rangle, \dots, \langle \{i_1, i_2\} \{i_1\} \rangle, \langle \{i_1, i_2\} \{i_2\} \rangle, \dots,$   
 $\langle \{i_1\} \{i_1, i_2\} \rangle, \langle \{i_1\} \{i_1, i_3\} \rangle, \dots, \langle \{i_1\} \{i_1\} \{i_1\} \rangle, \langle \{i_1\} \{i_1\} \{i_2\} \rangle, \dots$

# Generalized Sequential Pattern (GSP)

- **Step 1:**

- Make the first pass over the sequence database  $D$  to yield all the 1-element frequent sequences

- **Step 2:**

Repeat until no new frequent sequences are found

- **Candidate Generation:**

- ◆ Merge pairs of frequent subsequences found in the  $(k - 1)th$  pass to generate candidate sequences that contain  $k$  items

- **Candidate Pruning:**

- ◆ Prune candidate  $k$ -sequences that contain infrequent  $(k - 1)$ -subsequences

- **Support Counting:**

- ◆ Make a new pass over the sequence database  $D$  to find the support for these candidate sequences

- **Candidate Elimination:**

- ◆ Eliminate candidate  $k$ -sequences whose actual support is less than *minsup*

# Candidate Generation

- **Base case ( $k = 2$ ):**

- Merging two frequent 1-sequences  $\langle \{i_1\} \rangle$  and  $\langle \{i_2\} \rangle$  will produce two candidate 2-sequences:  $\langle \{i_1\} \{i_2\} \rangle$  and  $\langle \{i_1 i_2\} \rangle$

- **General case ( $k > 2$ ):**

- A frequent  $(k - 1)$ -sequence  $w_1$  is merged with another frequent  $(k - 1)$ -sequence  $w_2$  to produce a candidate  $k$ -sequence if the subsequence obtained by removing the first event in  $w_1$  is the same as the subsequence obtained by removing the last event in  $w_2$ 
  - ◆ The resulting candidate after merging is given by the sequence  $w_1$  extended with the last event of  $w_2$ .
    - If the last two events in  $w_2$  belong to the same element, then the last event in  $w_2$  becomes part of the last element in  $w_1$
    - Otherwise, the last event in  $w_2$  becomes a separate element appended to the end of  $w_1$

# Candidate Generation Examples

- **Merging sequences**

$w_1 = \langle \{1\} \{2\ 3\} \{4\} \rangle$  and  $w_2 = \langle \{2\ 3\} \{4\ 5\} \rangle$

will produce the candidate sequence  $\langle \{1\} \{2\ 3\} \{4\ 5\} \rangle$  because the last two events in  $w_2$  (4 and 5) belong to the same element

- **Merging sequences**

$w_1 = \langle \{1\} \{2\ 3\} \{4\} \rangle$  and  $w_2 = \langle \{2\ 3\} \{4\} \{5\} \rangle$

will produce the candidate sequence  $\langle \{1\} \{2\ 3\} \{4\} \{5\} \rangle$  because the last two events in  $w_2$  (4 and 5) do not belong to the same element

- **We do not have to merge these sequences**

$w_1 = \langle \{1\} \{2\ 6\} \{4\} \rangle$  and  $w_2 = \langle \{1\} \{2\} \{4\ 5\} \rangle$

to produce the candidate  $\langle \{1\} \{2\ 6\} \{4\ 5\} \rangle$  because if the latter is a viable candidate, then it can be obtained by merging  $w_1$  with  $\langle \{1\} \{2\ 6\} \{5\} \rangle$

# GSP Example

Frequent  
3-sequences

< {1} {2} {3} >  
< {1} {2 5} >  
< {1} {5} {3} >  
< {2} {3} {4} >  
< {2 5} {3} >  
< {3} {4} {5} >  
< {5} {3 4} >

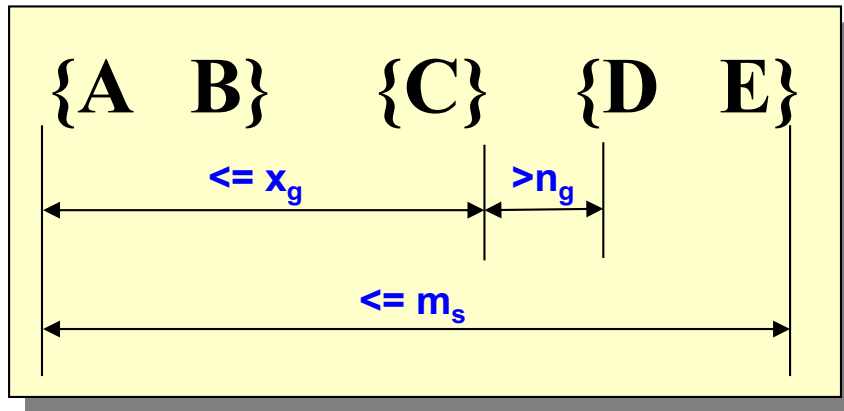
Candidate  
Generation

< {1} {2} {3} {4} >  
< {1} {2 5} {3} >  
< {1} {5} {3 4} >  
< {2} {3} {4} {5} >  
< {2 5} {3 4} >

Candidate  
Pruning

< {1} {2 5} {3} >

# Timing Constraints (I)



$x_g$ : max-gap

$n_g$ : min-gap

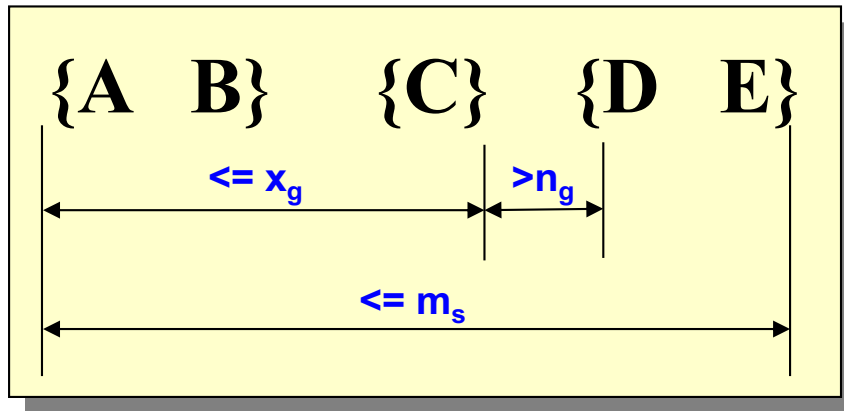
$m_s$ : maximum span

$x_g = 2, n_g = 0, m_s = 4$

Data sequence	Subsequence	Contain?
$\langle \{2,4\} \{3,5,6\} \{4,7\} \{4,5\} \{8\} \rangle$	$\langle \{6\} \{5\} \rangle$	
$\langle \{1\} \{2\} \{3\} \{4\} \{5\} \rangle$	$\langle \{1\} \{4\} \rangle$	
$\langle \{1\} \{2,3\} \{3,4\} \{4,5\} \rangle$	$\langle \{2\} \{3\} \{5\} \rangle$	
$\langle \{1,2\} \{3\} \{2,3\} \{3,4\} \{2,4\} \{4,5\} \rangle$	$\langle \{1,2\} \{5\} \rangle$	



# Timing Constraints (I)



$x_g$ : max-gap

$n_g$ : min-gap

$m_s$ : maximum span

$x_g = 2, n_g = 0, m_s = 4$

Data sequence	Subsequence	Contain?
$\langle \{2,4\} \{3,5,6\} \{4,7\} \{4,5\} \{8\} \rangle$	$\langle \{6\} \{5\} \rangle$	Yes
$\langle \{1\} \{2\} \{3\} \{4\} \{5\} \rangle$	$\langle \{1\} \{4\} \rangle$	No
$\langle \{1\} \{2,3\} \{3,4\} \{4,5\} \rangle$	$\langle \{2\} \{3\} \{5\} \rangle$	Yes
$\langle \{1,2\} \{3\} \{2,3\} \{3,4\} \{2,4\} \{4,5\} \rangle$	$\langle \{1,2\} \{5\} \rangle$	No

# Mining Sequential Patterns with Timing Constraints

- Approach 1:

- Mine sequential patterns without timing constraints
- Post-process the discovered patterns

- Approach 2:

- Modify GSP to directly prune candidates that violate timing constraints
- Question:
  - ◆ Does Apriori principle still hold?

# Apriori Principle for Sequence Data

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

Suppose:

$x_g = 1$  (max-gap)

$n_g = 0$  (min-gap)

$m_s = 5$  (maximum span)

*minsup* = 60%

$\langle \{2\} \{5\} \rangle$  support = 40%

but

$\langle \{2\} \{3\} \{5\} \rangle$  support = 60%

Problem exists because of max-gap constraint

No such problem if max-gap is infinite

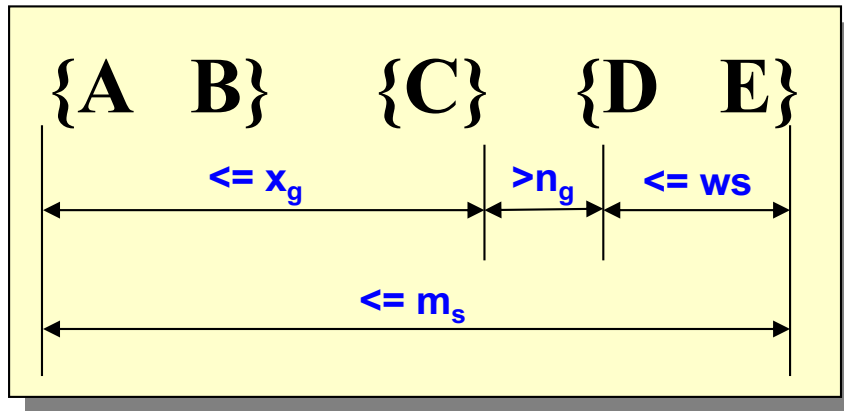
# Contiguous Subsequences

- $s$  is a contiguous subsequence of  $w = \langle e_1 \rangle \langle e_2 \rangle \dots \langle e_k \rangle$  if any of the following conditions hold:
  1.  $s$  is obtained from  $w$  by deleting an item from either  $e_1$  or  $e_k$
  2.  $s$  is obtained from  $w$  by deleting an item from any element  $e_i$  that contains at least 2 items
  3.  $s$  is a contiguous subsequence of  $s'$  and  $s'$  is a contiguous subsequence of  $w$  (recursive definition)
- Examples:  $s = \langle \{1\} \{2\} \rangle$ 
  - is a contiguous subsequence of  $\langle \{1\} \{2\} \{3\} \rangle$ ,  $\langle \{1\} \{2\} \{3\} \{4\} \rangle$ , and  $\langle \{3\} \{4\} \{1\} \{2\} \{2\} \{3\} \{4\} \rangle$
  - is not a contiguous subsequence of  $\langle \{1\} \{3\} \{2\} \rangle$  and  $\langle \{2\} \{1\} \{3\} \{2\} \rangle$

# Modified Candidate Pruning Step

- Without maxgap constraint:
  - A candidate  $k$ -sequence is pruned if at least one of its  $(k-1)$ -subsequences is infrequent
- With maxgap constraint:
  - A candidate  $k$ -sequence is pruned if at least one of its **contiguous**  $(k-1)$ -subsequences is infrequent

# Timing Constraints (II)



$x_g$ : max-gap

$n_g$ : min-gap

**ws**: window size

$m_s$ : maximum span

$x_g = 2$ ,  $n_g = 0$ , **ws = 1**,  $m_s = 5$

Data sequence	Subsequence	Contain?
$\langle \{2,4\} \{3,5,6\} \{4,7\} \{4,6\} \{8\} \rangle$	$\langle \{3\} \{5\} \rangle$	No
$\langle \{1\} \{2\} \{3\} \{4\} \{5\} \rangle$	$\langle \{1,2\} \{3\} \rangle$	Yes
$\langle \{1,2\} \{2,3\} \{3,4\} \{4,5\} \rangle$	$\langle \{1,2\} \{3,4\} \rangle$	Yes

# Modified Support Counting Step

- Given a candidate pattern:  $\langle \{a, c\} \rangle$

- Any data sequences that contain

- $\langle \dots \{a\ c\} \dots \rangle$ ,

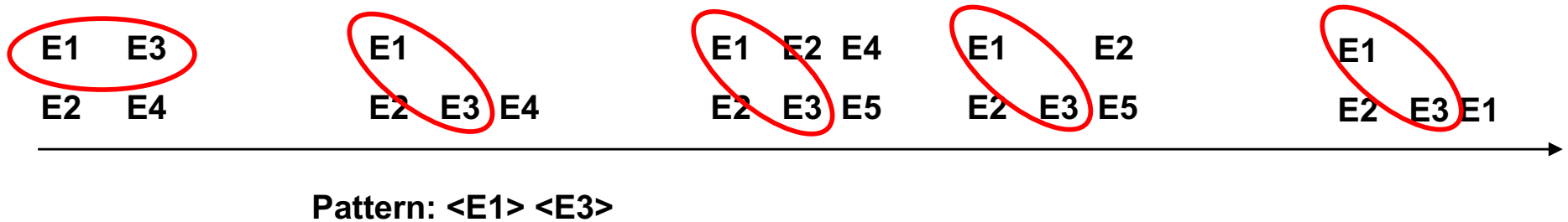
- $\langle \dots \{a\} \dots \{c\} \dots \rangle$  ( where  $\text{time}(\{c\}) - \text{time}(\{a\}) \leq ws$ )

- $\langle \dots \{c\} \dots \{a\} \dots \rangle$  (where  $\text{time}(\{a\}) - \text{time}(\{c\}) \leq ws$ )

will contribute to the support count of candidate pattern

# Other Formulation

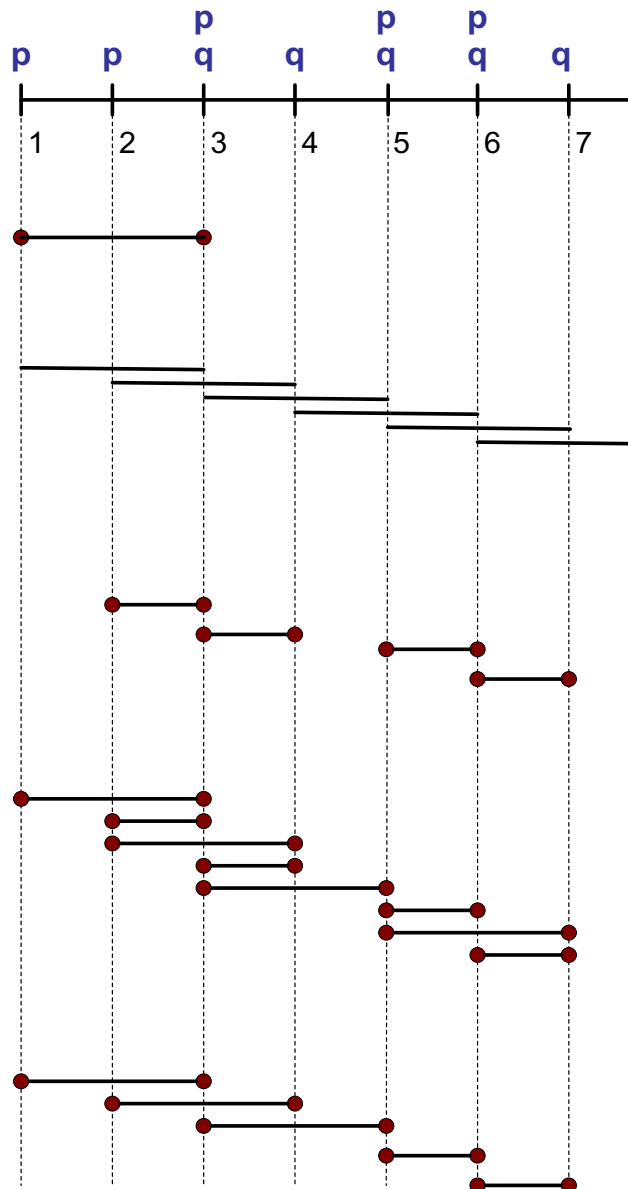
- In some domains, we may have only one very long time series
  - Example:
    - ◆ monitoring network traffic events for attacks
    - ◆ monitoring telecommunication alarm signals
- Goal is to find frequent sequences of events in the time series
  - This problem is also known as frequent episode mining





# General Support Counting Schemes

Object's Timeline



Sequence: (p) (q)

Method      Support  
                 Count

COBJ      1

CWIN      6

CMINWIN      4

CDIST\_O      8

CDIST      5

Assume:

$x_g = 2$  (max-gap)

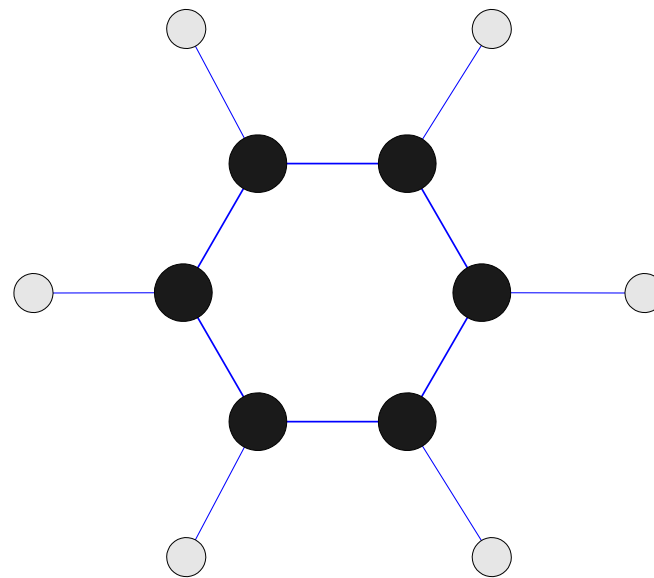
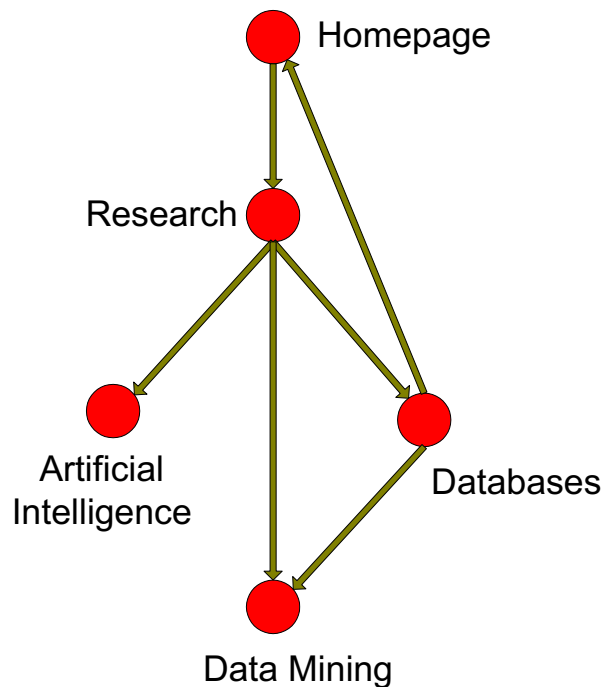
$n_g = 0$  (min-gap)

$ws = 0$  (window size)

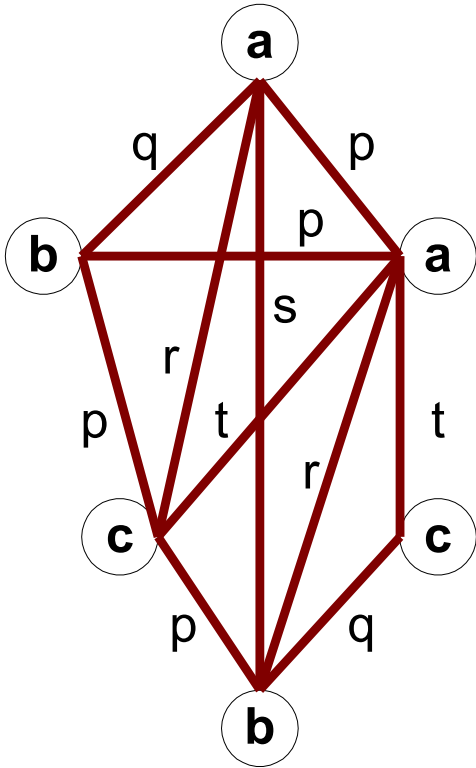
$m_s = 2$  (maximum span)

# Frequent Subgraph Mining

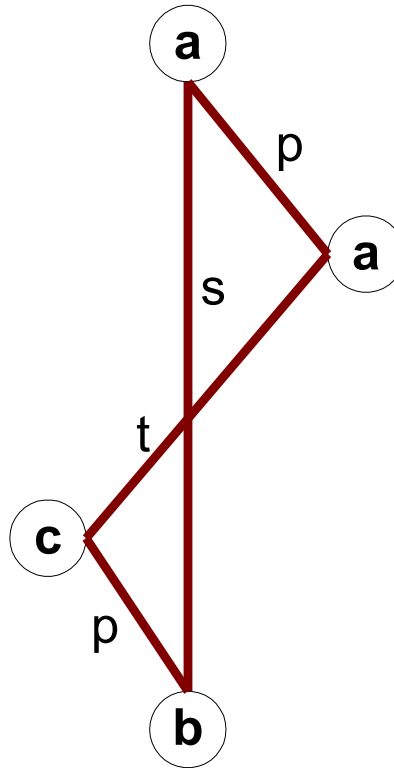
- Extend association rule mining to finding frequent subgraphs
- Useful for web mining, computational chemistry, bioinformatics, spatial data sets, etc.



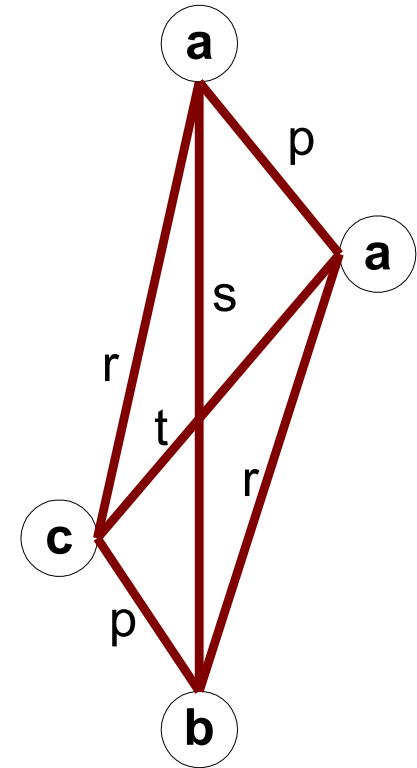
# Graph Definitions



(a) Labeled Graph



(b) Subgraph

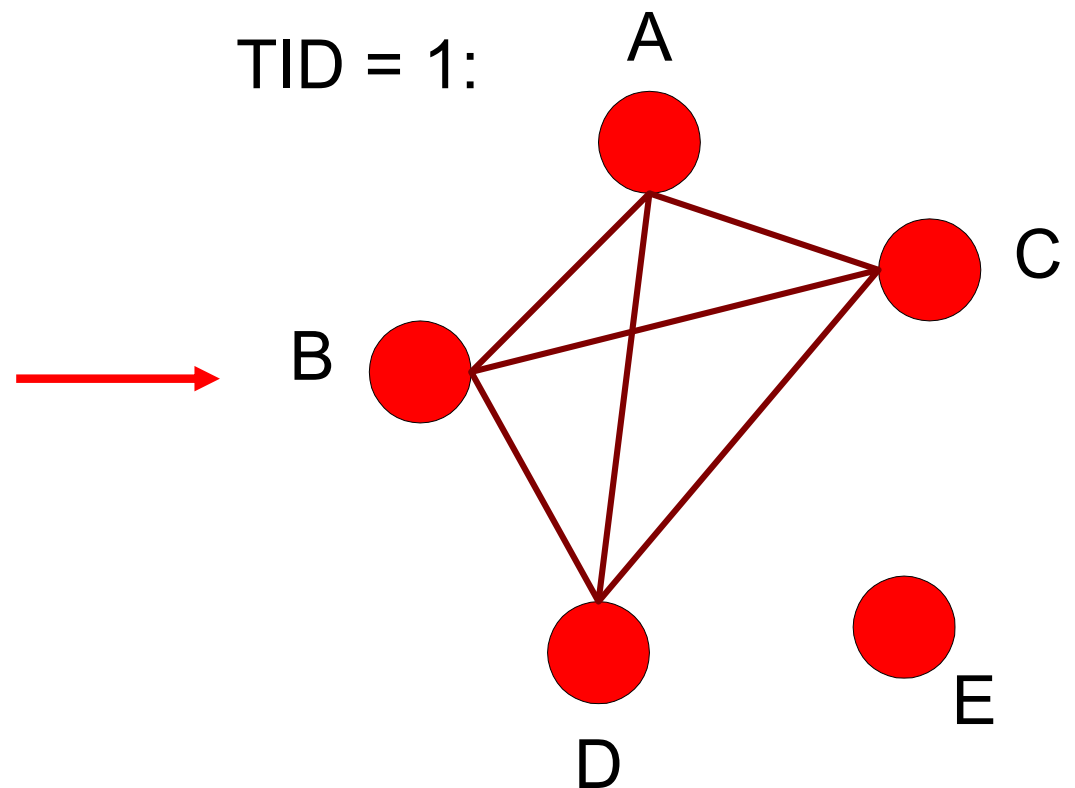


(c) Induced Subgraph

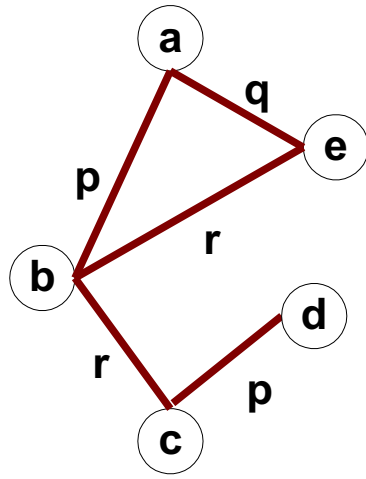
# Representing Transactions as Graphs

- Each transaction is a clique of items

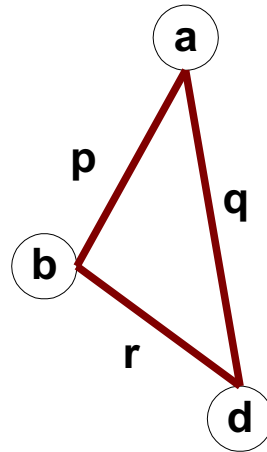
Transaction Id	Items
1	{A,B,C,D}
2	{A,B,E}
3	{B,C}
4	{A,B,D,E}
5	{B,C,D}



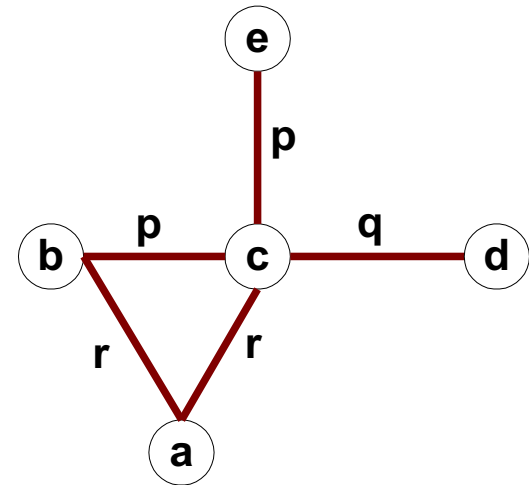
# Representing Graphs as Transactions



# G1



## G2



# G3

	(a,b,p)	(a,b,q)	(a,b,r)	(b,c,p)	(b,c,q)	(b,c,r)	...	(d,e,r)
G1	1	0	0	0	0	1	...	0
G2	1	0	0	0	0	0	...	0
G3	0	0	1	1	0	0	...	0
G3	...	...	...	...	...	...	...	...

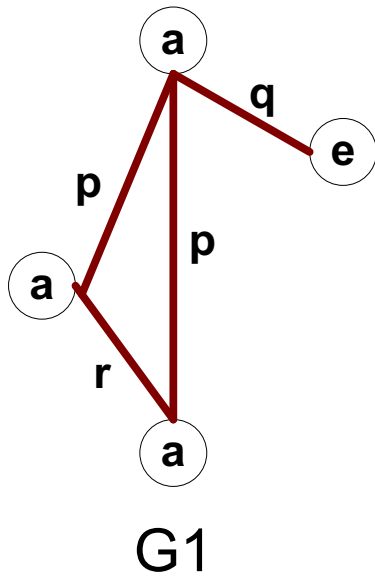
# Challenges

- Node may contain duplicate labels
- Support and confidence
  - How to define them?
- Additional constraints imposed by pattern structure
  - Support and confidence are not the only constraints
  - Assumption: frequent subgraphs must be connected
- Apriori-like approach:
  - Use frequent  $k$ -subgraphs to generate frequent  $(k+1)$  subgraphs
    - ◆ What is  $k$ ?

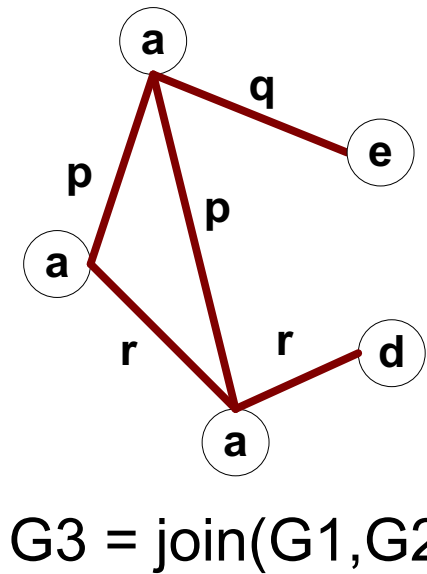
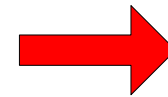
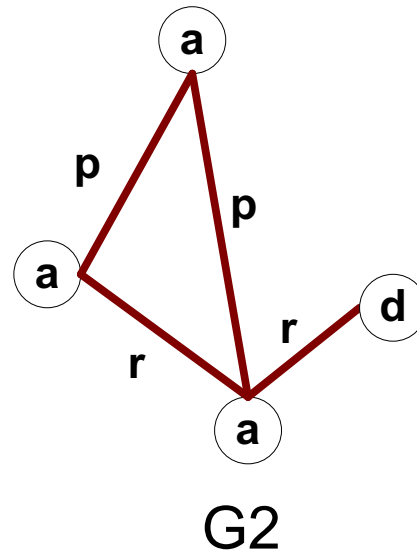
# Challenges...

- Support:
  - number of graphs that contain a particular subgraph
- Apriori principle still holds
- Level-wise (Apriori-like) approach:
  - Vertex growing:
    - ◆  $k$  is the number of vertices
  - Edge growing:
    - ◆  $k$  is the number of edges

# Vertex Growing



+



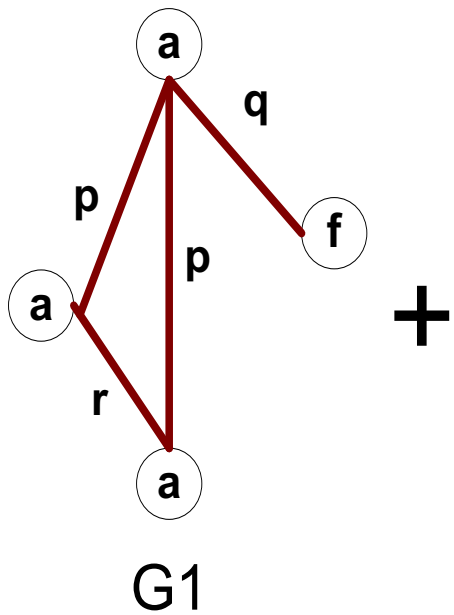
$$M_{G_1} = \begin{pmatrix} 0 & p & p & q \\ p & 0 & r & 0 \\ p & r & 0 & 0 \\ q & 0 & 0 & 0 \end{pmatrix}$$

$$M_{G_2} = \begin{pmatrix} 0 & p & p & 0 \\ p & 0 & r & 0 \\ p & r & 0 & r \\ 0 & 0 & r & 0 \end{pmatrix}$$

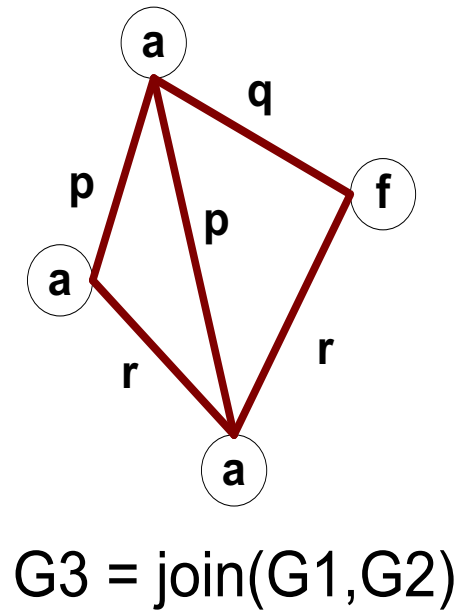
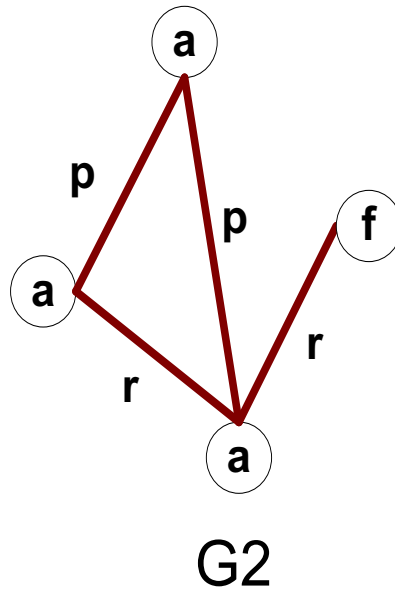
$$M_{G_3} = \begin{pmatrix} 0 & p & p & 0 & q \\ p & 0 & r & 0 & 0 \\ p & r & 0 & r & 0 \\ 0 & 0 & r & 0 & 0 \\ q & 0 & 0 & 0 & 0 \end{pmatrix}$$



# Edge Growing



+

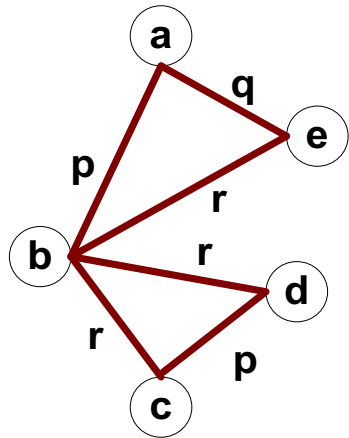


# Apriori-like Algorithm

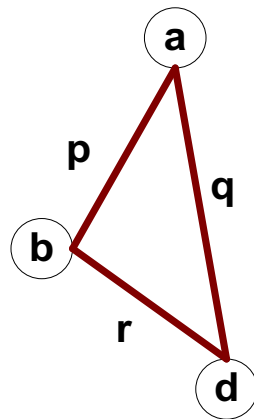
- Find frequent 1-subgraphs
- Repeat
  - Candidate generation
    - ◆ Use frequent  $(k - 1)$ -subgraphs to generate candidate  $k$ -subgraph
  - Candidate pruning
    - ◆ Prune candidate subgraphs that contain infrequent  $(k - 1)$ -subgraphs
  - Support counting
    - ◆ Count the support of each remaining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

**In practice, it is not as easy. There are many other issues.**

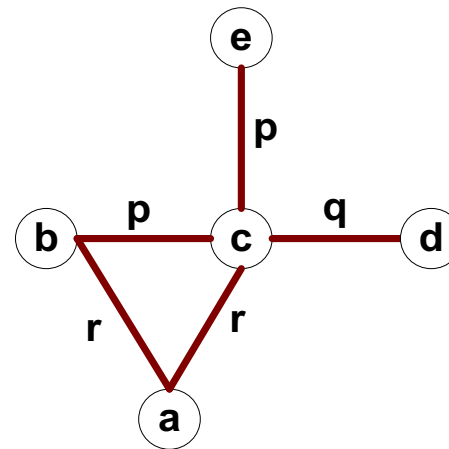
# Example: Dataset



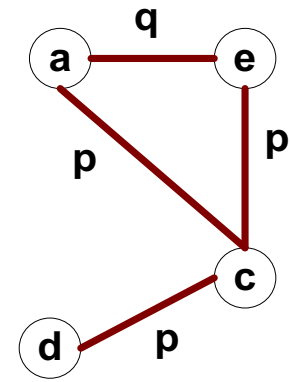
G1



G2



G3



G4

	(a,b,p)	(a,b,q)	(a,b,r)	(b,c,p)	(b,c,q)	(b,c,r)	...	(d,e,r)
G1	1	0	0	0	0	1	...	0
G2	1	0	0	0	0	0	...	0
G3	0	0	1	1	0	0	...	0
G4	0	0	0	0	0	0	...	0

# Example

Minimum support count = 2

k=1

Frequent  
Subgraphs

a

b

c

d

e

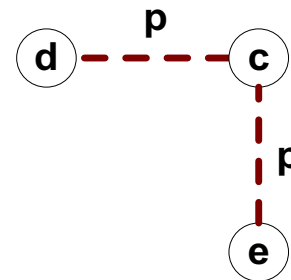
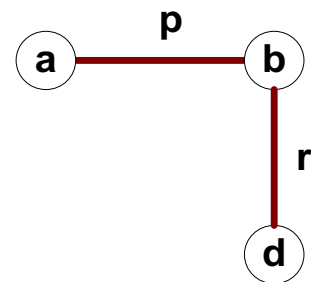
k=2

Frequent  
Subgraphs



k=3

Candidate  
Subgraphs

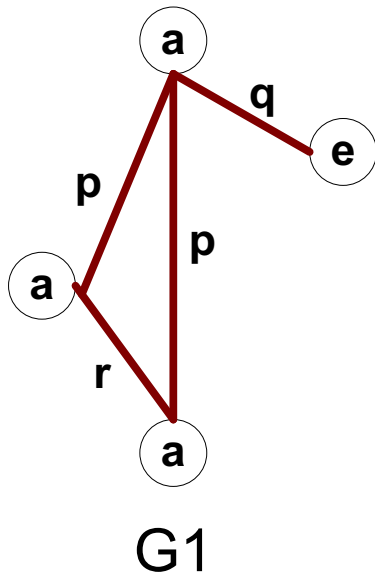


(Pruned candidate)

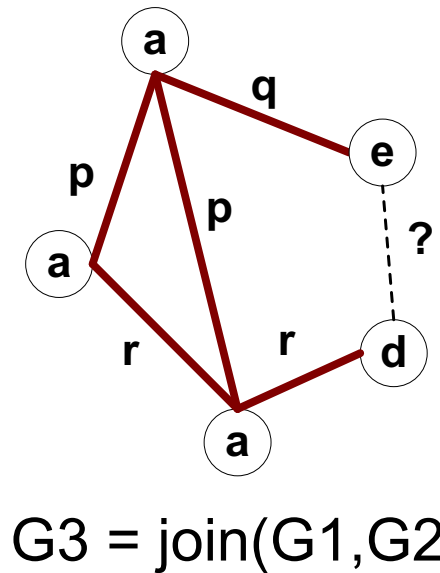
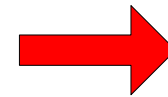
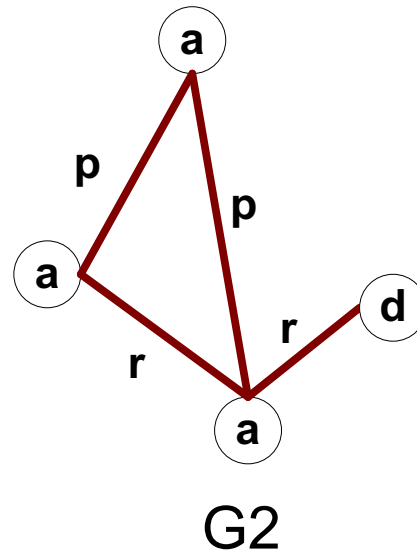
# Candidate Generation

- In Apriori:
  - Merging two frequent  $k$ -itemsets will produce a candidate  $(k + 1)$ -itemset
- In frequent subgraph mining (vertex/edge growing)
  - Merging two frequent  $k$ -subgraphs may produce more than one candidate  $(k + 1)$ -subgraph

# Multiplicity of Candidates (Vertex Growing)



+



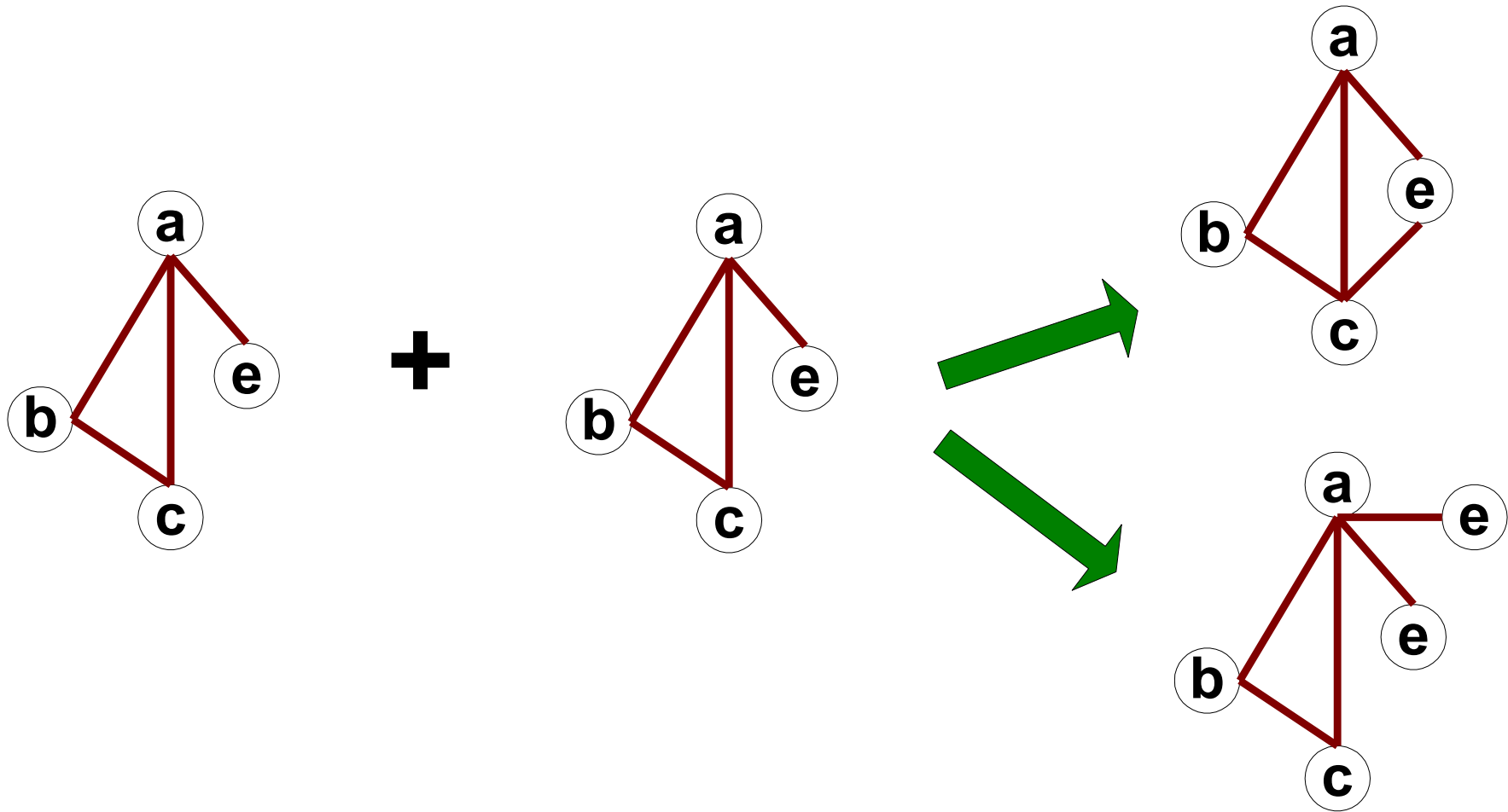
$$M_{G_1} = \begin{pmatrix} 0 & p & p & q \\ p & 0 & r & 0 \\ p & r & 0 & 0 \\ q & 0 & 0 & 0 \end{pmatrix}$$

$$M_{G_2} = \begin{pmatrix} 0 & p & p & 0 \\ p & 0 & r & 0 \\ p & r & 0 & r \\ 0 & 0 & r & 0 \end{pmatrix}$$

$$M_{G_3} = \begin{pmatrix} 0 & p & p & 0 & q \\ p & 0 & r & 0 & 0 \\ p & r & 0 & r & 0 \\ 0 & 0 & r & 0 & ? \\ q & 0 & 0 & ? & 0 \end{pmatrix}$$

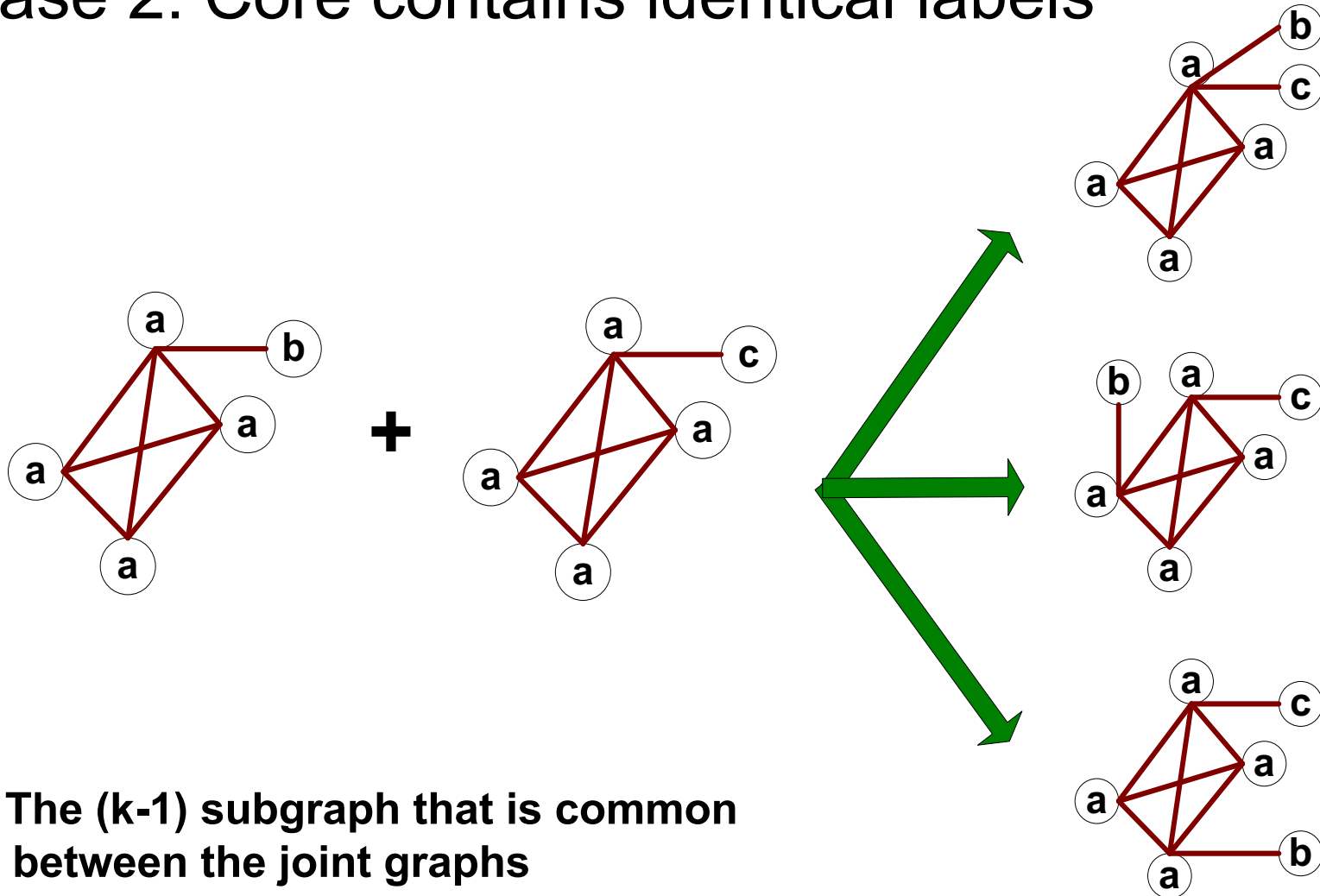
# Multiplicity of Candidates (Edge growing)

- Case 1: identical vertex labels



# Multiplicity of Candidates (Edge growing)

- Case 2: Core contains identical labels

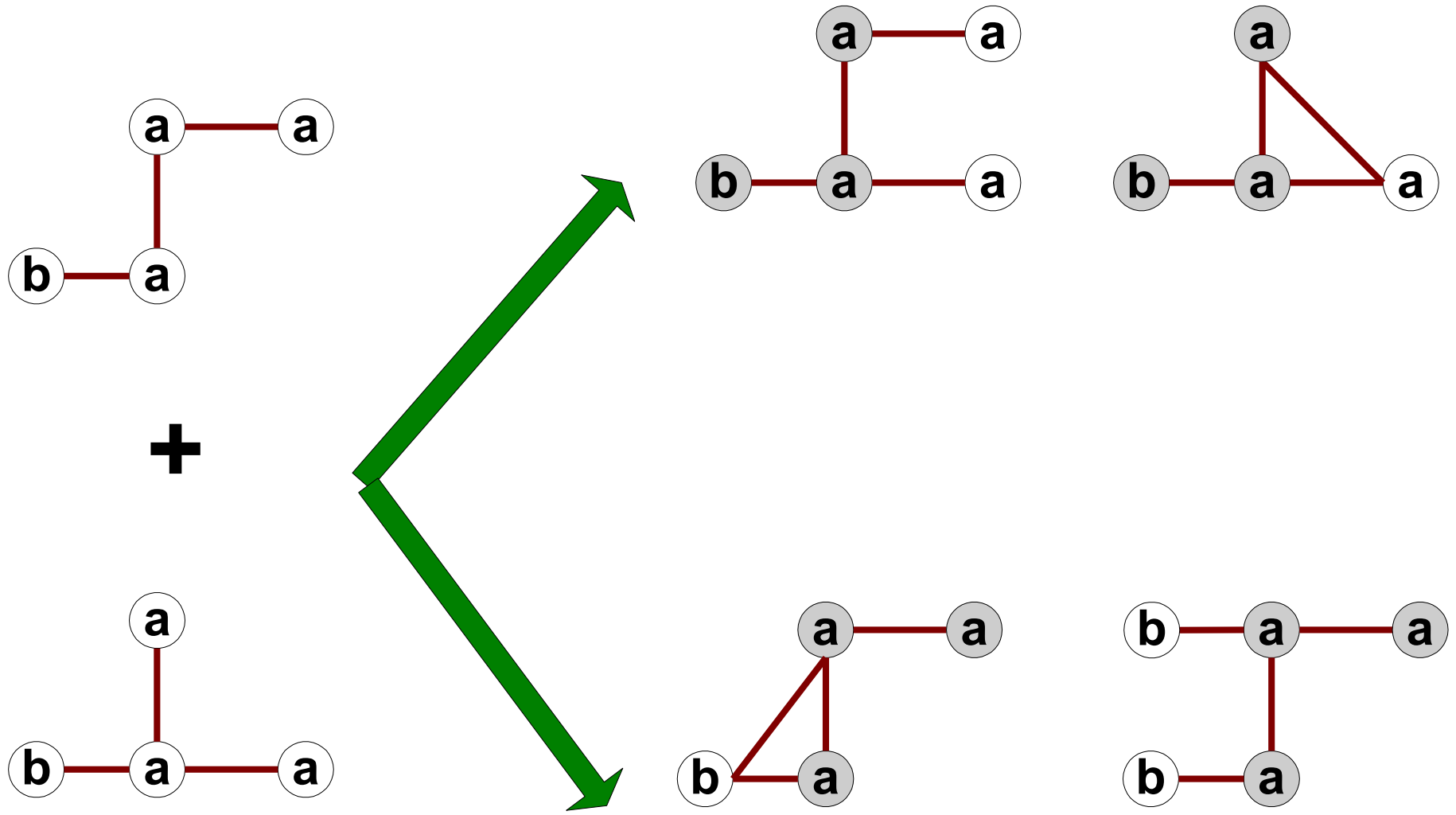


**Core:** The  $(k-1)$  subgraph that is common between the joint graphs

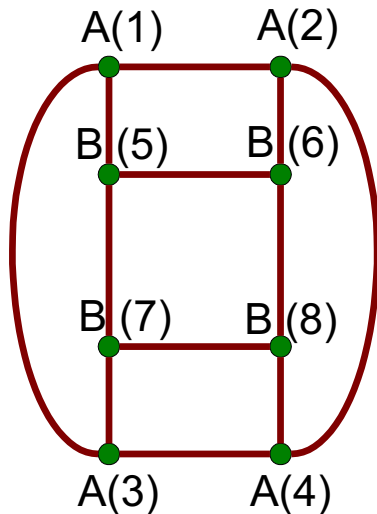


# Multiplicity of Candidates (Edge growing)

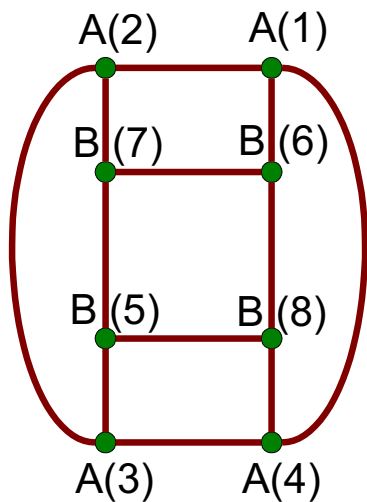
- Case 3: Core multiplicity



# Adjacency Matrix Representation



	A(1)	A(2)	A(3)	A(4)	B(5)	B(6)	B(7)	B(8)
A(1)	1	1	1	0	1	0	0	0
A(2)	1	1	0	1	0	1	0	0
A(3)	1	0	1	1	0	0	1	0
A(4)	0	1	1	1	0	0	0	1
B(5)	1	0	0	0	1	1	1	0
B(6)	0	1	0	0	1	1	0	1
B(7)	0	0	1	0	1	0	1	1
B(8)	0	0	0	1	0	1	1	1

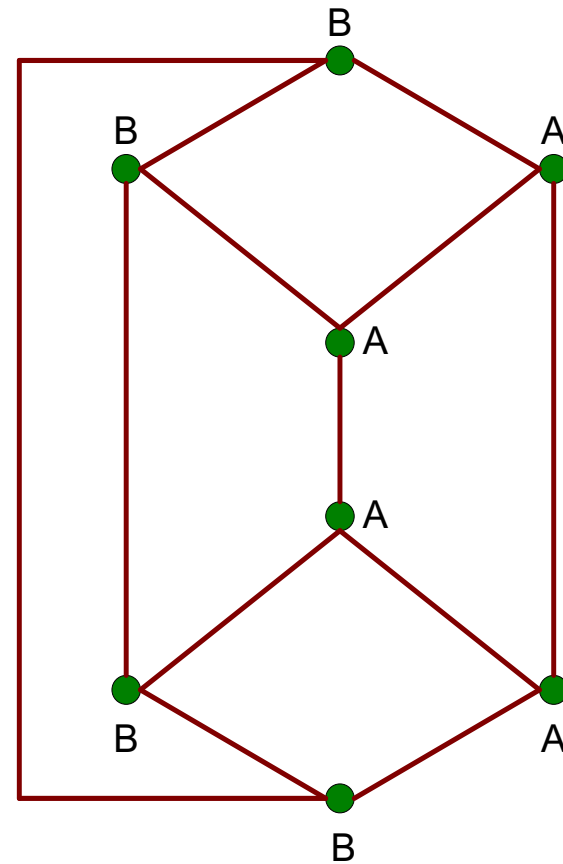
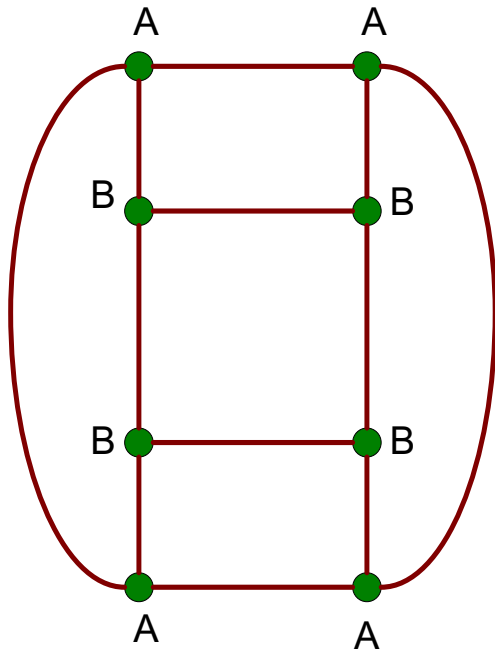


	A(1)	A(2)	A(3)	A(4)	B(5)	B(6)	B(7)	B(8)
A(1)	1	1	0	1	0	1	0	0
A(2)	1	1	1	0	0	0	1	0
A(3)	0	1	1	1	1	0	0	0
A(4)	1	0	1	1	0	0	0	1
B(5)	0	0	1	0	1	0	1	1
B(6)	1	0	0	0	0	1	1	1
B(7)	0	1	0	0	1	1	1	0
B(8)	0	0	0	1	1	1	0	1

- The same graph can be represented in many ways

# Graph Isomorphism

- A graph is isomorphic if it is topologically equivalent to another graph



# Graph Isomorphism

- Test for graph isomorphism is needed:
  - During candidate generation step, to determine whether a candidate has been generated
  - During candidate pruning step, to check whether its  $(k - 1)$ -subgraphs are frequent
  - During candidate counting, to check whether a candidate is contained within another graph

# Graph Isomorphism

- Use canonical labeling to handle isomorphism
  - Map each graph into an ordered string representation (known as its code) such that two isomorphic graphs will be mapped to the same canonical encoding
  - Example:
    - ◆ Lexicographically largest adjacency matrix

