Penambangan Data [Data Mining]

Kode : SIT5255
Bobot : 2 SKS
Dosen Pengampu : Dr. Heny Pratiwi, S.Kom., M.Pd., M.TI

Advanced Frequent Pattern Mining

Continuous and Categorical Attributes

How to apply association analysis formulation to non-asymmetric 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	Netscape	No
3	USA	2125	45	Female	Mozilla	Yes
4	Germany	596	4	Male	IE	Yes
5	Australia	123	9	Male	Mozilla	No
…	…	…	…	…	…	…

Example of Association Rule:

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

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 = Mozilla
    - Browser Type = Mozilla

Handling Categorical Attributes

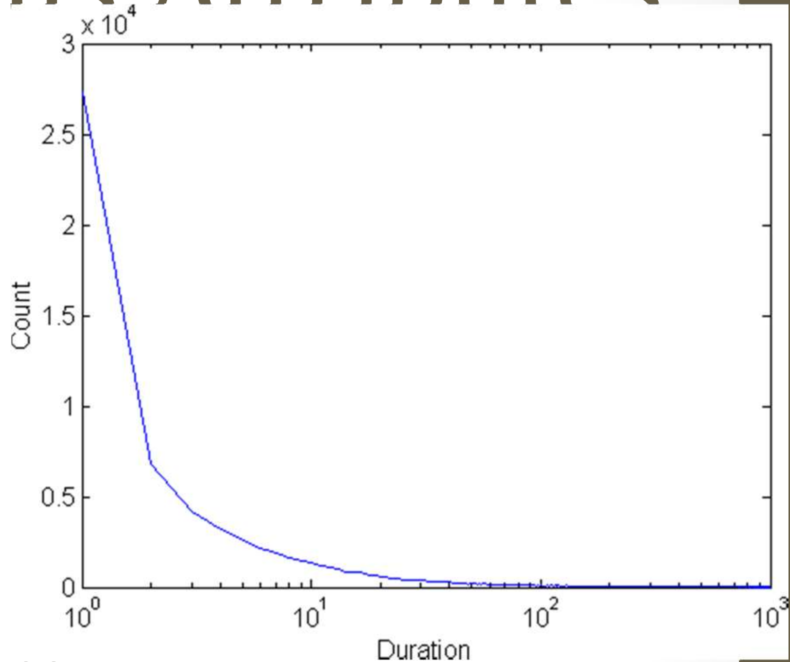
- Potential Issues
  - What if attribute has many possible values
    - Example: attribute country has more than 200 possible values
    - Many of the attribute values may have very low support
      - Potential solution: Aggregate the low-support attribute values
  - What if distribution of attribute values is highly skewed
    - Example: 95% of the visitors have Buy = No
    - Most of the items will be associated with (Buy=No) item
      - Potential solution: drop the highly frequent items

Handling Continuous Attributes

- Different kinds of rules:
  - $Age \in [21,35) \wedge Salary \in [70k,120k) \rightarrow Buy$
  - $Salary \in [70k,120k) \wedge Buy \rightarrow Age: \mu=28, \sigma=4$
- 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

$\underbrace{v_1 \; v_2}_{bin_1}$ $\underbrace{v_3 \; v_4 \; v_5}_{bin_2}$ $\underbrace{v_6 \; v_7 \; v_8 \; v_9}_{bin_3}$

Discretization Issues

- Size of the discretized intervals affect support & confidence

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

$$\{Refund = No, (60K \leq Income \leq 80K)\} \rightarrow \{Cheat = No\}$$

$$\{Refund = No, (0K \leq Income \leq 1B)\} \rightarrow \{Cheat = No\}$$

  - If intervals too small
    - may not have enough support
  - If intervals too large
    - may not have enough confidence
- Potential solution: use all possible intervals

Statistics-based Methods

- Example:

  Browser=Mozilla $\wedge$ Buy=Yes $\rightarrow$ Age: μ=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 interesting?
  - Compare the statistics for segment of population covered by the rule vs segment of population not covered by the rule:

    $A \Rightarrow B: \mu$ versus $A \Rightarrow B: \mu'$

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

  - Statistical hypothesis testing:
    - Null hypothesis: H0: $\mu' = \mu + \Delta$
    - Alternative hypothesis: H1: $\mu' > \mu + \Delta$
    - Z has zero mean and variance 1 under null hypothesis

Statistics-based Methods

- Example:

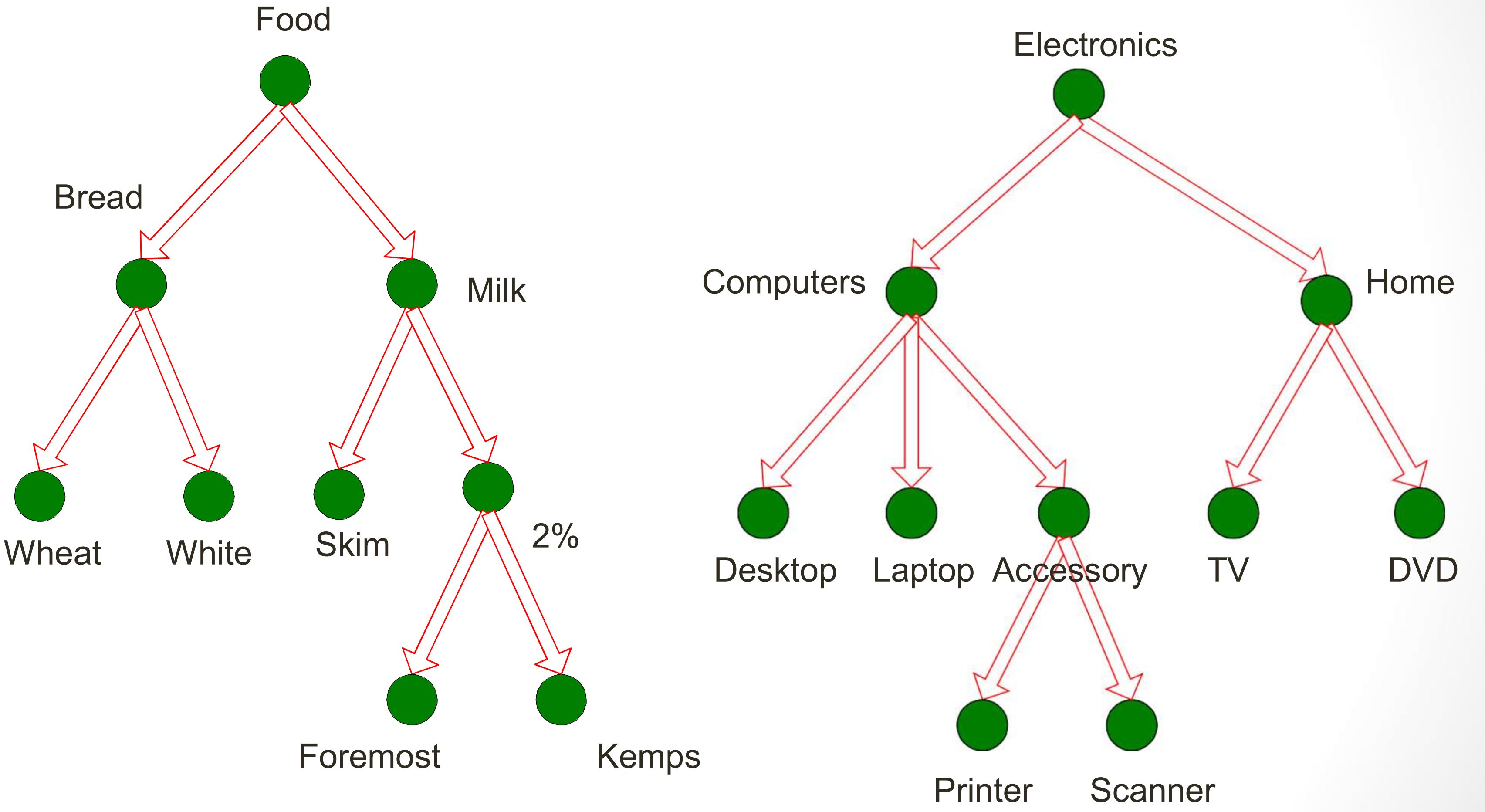
  r: Browser=Mozilla $\wedge$ Buy=Yes $\rightarrow$ Age: μ=23

  - Rule is interesting if difference between μ and μ' is greater than 5 years (i.e., Δ = 5)
  - For r, suppose n1 = 50, s1 = 3.5
  - For r' (complement): n2 = 250, s2 = 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

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 $\rightarrow$ white bread, 2% milk $\rightarrow$ wheat bread,
      skim milk $\rightarrow$ wheat bread, etc.
      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 X1 and X2, then $\sigma(X) \leq \sigma(X1) + \sigma(X2)$
  - If $\sigma(X1 \cup Y1) \geq$ minsup,
and X is parent of X1, Y is parent of Y1
then $\sigma(X \cup Y1) \geq$ minsup, $\sigma(X1 \cup Y) \geq$ minsup
$\sigma(X \cup Y) \geq$ minsup
  - If conf$(X1 \Rightarrow Y1) \geq$ minconf,
then conf$(X1 \Rightarrow Y) \geq$ 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

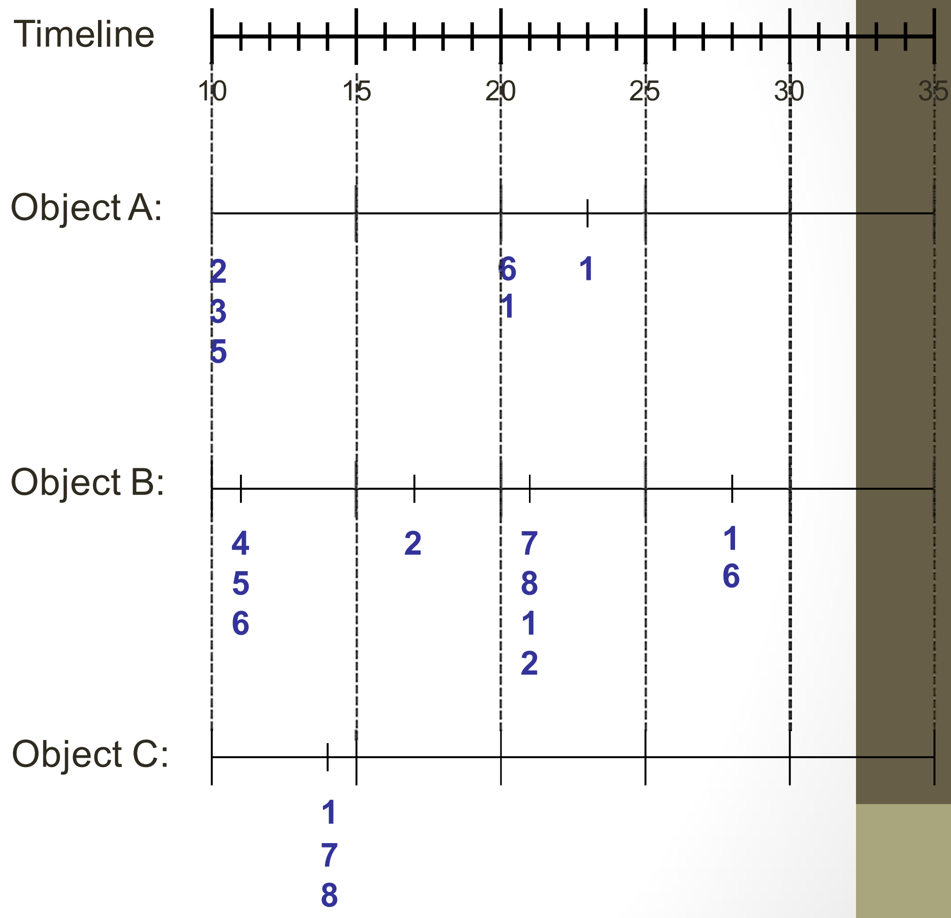
Beyond Itemsets

- Sequence Mining
  - Finding frequent subsequences from a collection of sequences
  - Time Series Motifs
  - DNA/Protein Sequence Motifs
- Graph Mining
  - Finding frequent (connected) subgraphs from a collection of graphs
- Tree Mining
  - Finding frequent (embedded) subtrees from a set of trees/graphs
- Geometric Structure Mining
  - Finding frequent substructures from 3-D or 2-D geometric graphs
- Among others…

Sequence Data

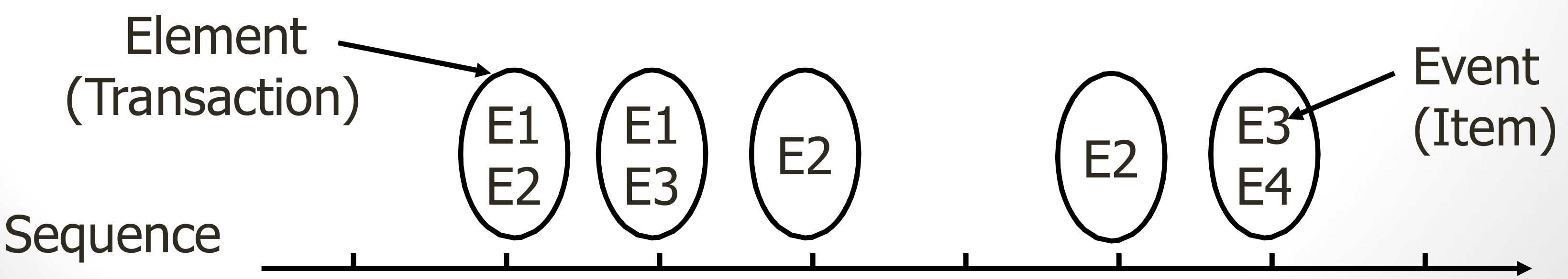
Sequence Database:

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)

$$s = \langle e_1 e_2 e_3 \ldots \rangle$$

  - Each element contains a collection of events (items)

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

  - 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 Sequence

- 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}>

Formal Definition of a Subsequence

- A sequence $<a_1\ a_2 \ldots a_n>$ is contained in another sequence $<b_1\ b_2 \ldots b_m>$ $(m \geq n)$ if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i1}$, …, $a_n \subseteq b_{in}$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	Yes
< {1,2} {3,4} >	< {1} {2} >	No
< {2,4} {2,4} {2,5} >	< {2} {4} >	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 ≥ *minsup*)

Sequential Pattern Mining: Definition

- Given:
  - a database of sequences
  - a user-specified minimum support threshold, *minsup*
- Task:
  - Find all subsequences with support ≥ *minsup*

Sequential Pattern Mining: Challenge

- 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}>

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:

< {1,2} >	s=60%
< {2,3} >	s=60%
< {2,4}>	s=80%
< {3} {5}>	s=80%
< {1} {2} >	s=80%
< {2} {2} >	s=60%
< {1} {2,3} >	s=60%
< {2} {2,3} >	s=60%
< {1,2} {2,3} >	s=60%

Extracting Sequential Patterns

- Given n events: $i_1, i_2, i_3, \ldots, i_n$
- Candidate 1-subsequences:
  - $<\{i_1\}>, <\{i_2\}>, <\{i_3\}>, \ldots, <\{i_n\}>$
- Candidate 2-subsequences:
  - $<\{i_1, i_2\}>, <\{i_1, i_3\}>, \ldots, <\{i_1\} \{i_1\}>, <\{i_1\} \{i_2\}>, \ldots, <\{i_{n-1}\} \{i_n\}>$
- Candidate 3-subsequences:
  - $<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \ldots, <\{i_1, i_2\} \{i_1\}>, <\{i_1, i_2\} \{i_2\}>, \ldots,$
  - $<\{i_1\} \{i_1, i_2\}>, <\{i_1\} \{i_1, i_3\}>, \ldots, <\{i_1\} \{i_1\} \{i_1\}>, <\{i_1\} \{i_1\} \{i_2\}>, \ldots$

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 Examples

- Merging the sequences
  w_1=<{1} {2 3} {4}> and w_2 =<{2 3} {4 5}>
  will produce the candidate sequence < {1} {2 3} {4 5}> because the last two events in w_2 (4 and 5) belong to the same element

- Merging the sequences
  w_1=<{1} {2 3} {4}> and w_2 =<{2 3} {4} {5}>
  will produce the candidate sequence < {1} {2 3} {4} {5}> because the last two events in w_2 (4 and 5) do not belong to the same element

- We do not have to merge the sequences
  w_1 =<{1} {2 6} {4}> and w_2 =<{1} {2} {4 5}>
  to produce the candidate < {1} {2 6} {4 5}> because if the latter is a viable candidate, then it can be obtained by merging w_1 with
  < {1} {2 6} {5}>

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} >

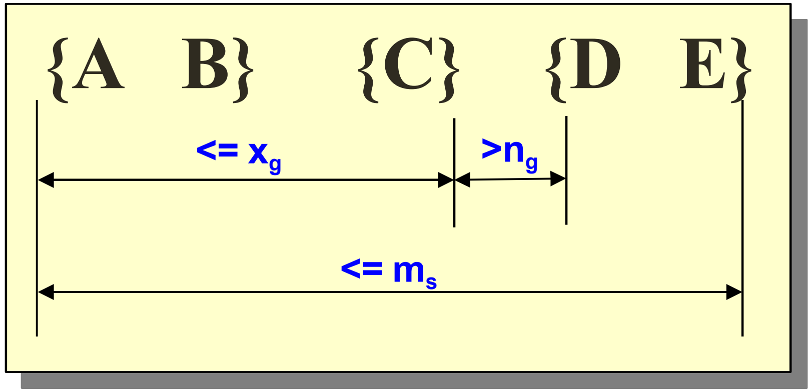
Candidate Generation

< {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} >

TimingConstraints (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?
< {2,4} {3,5,6} {4,7} {4,5} {8} >	< {6} {5} >	Yes
< {1} {2} {3} {4} {5}>	< {1} {4} >	No
< {1} {2,3} {3,4} {4,5}>	< {2} {3} {5} >	Yes
< {1,2} {3} {2,3} {3,4} {2,4} {4,5}>	< {1,2} {5} >	No

Mining Sequential Patterns with Timing Constraints

- Approach 1:
  - Mine sequential patterns without timing constraints
  - Postprocess 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%

<{2} {5}> support = 40%

but

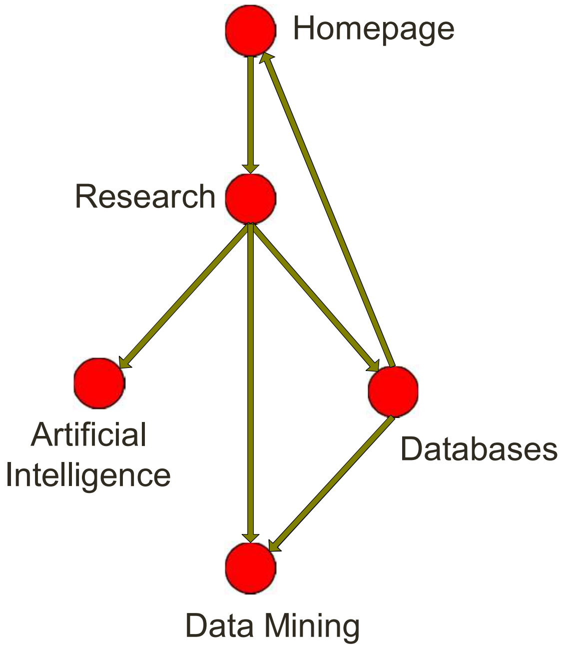
<{2} {3} {5}> support = 60%

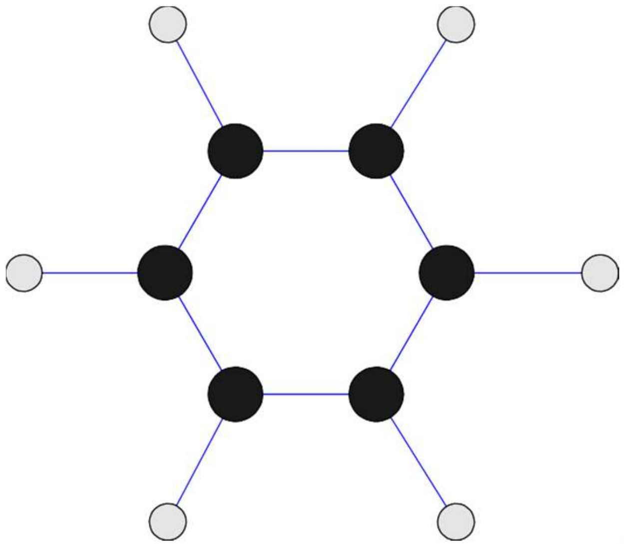
Problem exists because of max-gap constraint

No such problem if max-gap is infinite

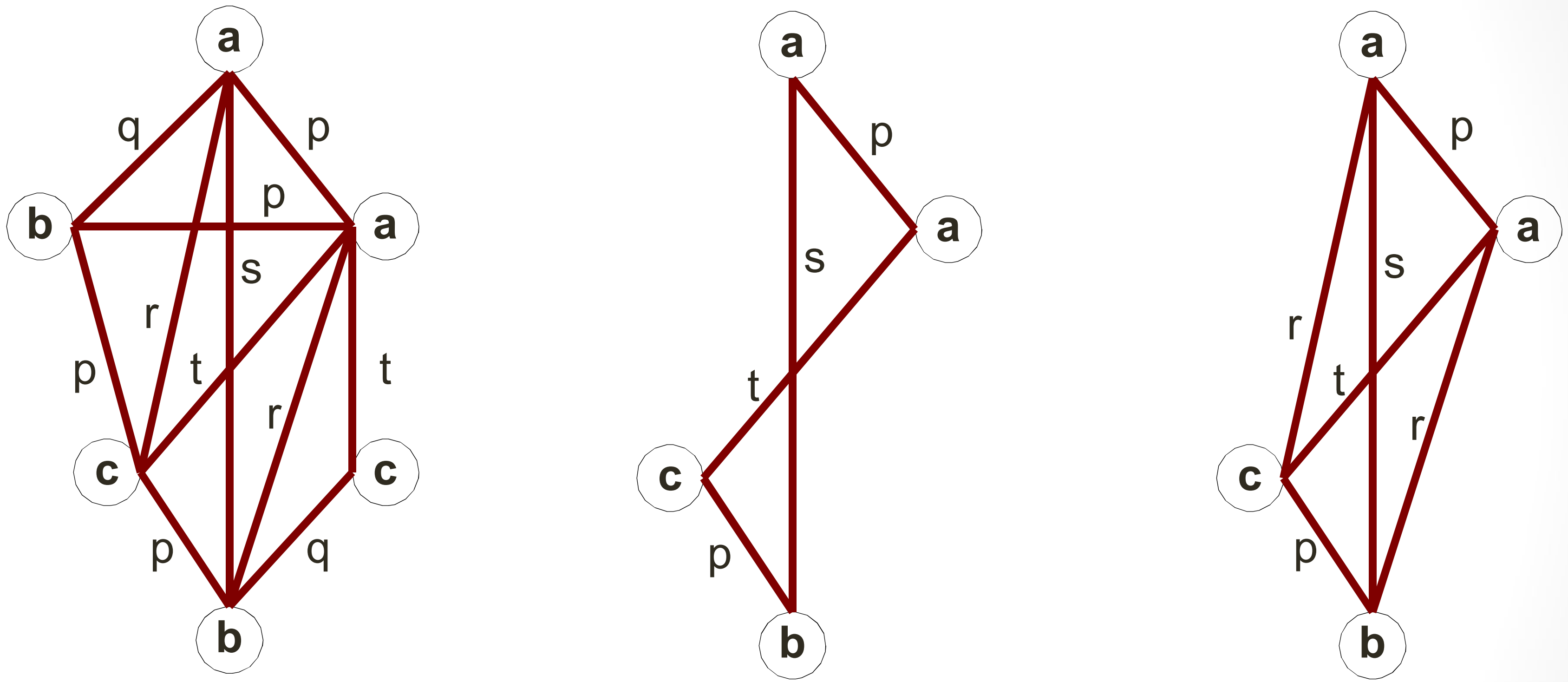
Frequent Subgraph Mining

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





Graph Definitions



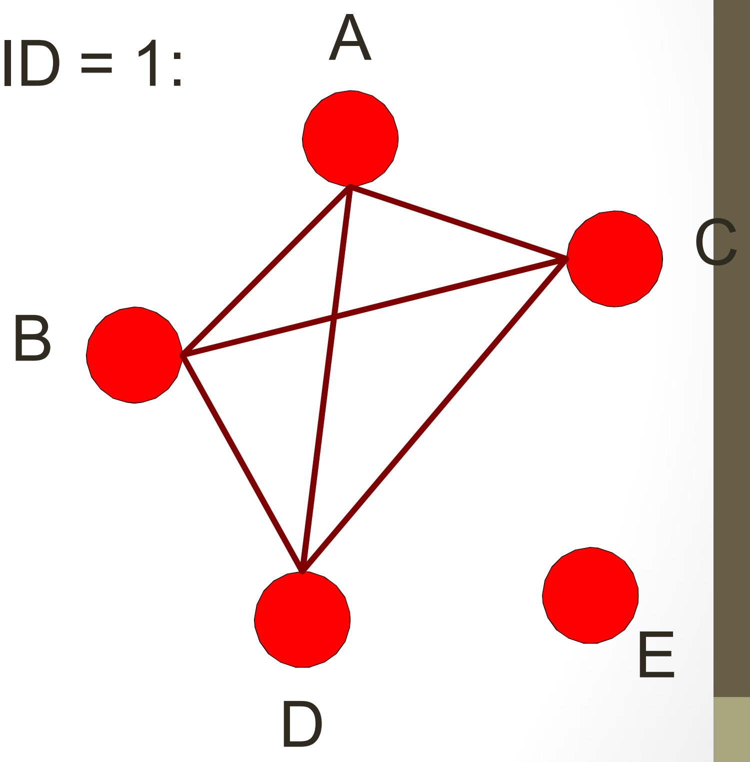
(a) Labeled Graph (b) Subgraph (c) Induced Subgraph

Representing Transactions as Graphs

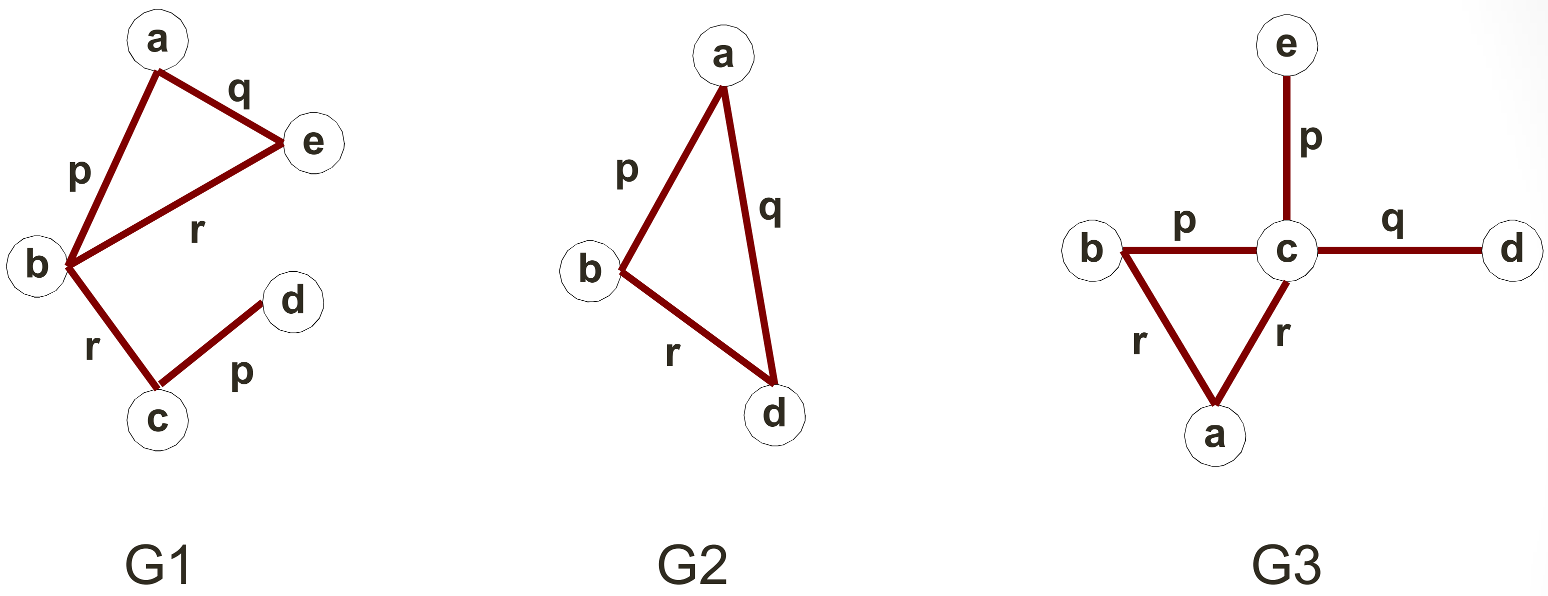
- Each transaction is a clique of items

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

TID = 1:



Representing Graphs as Transactions



	(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	…	…	…	…	…	…	…	…

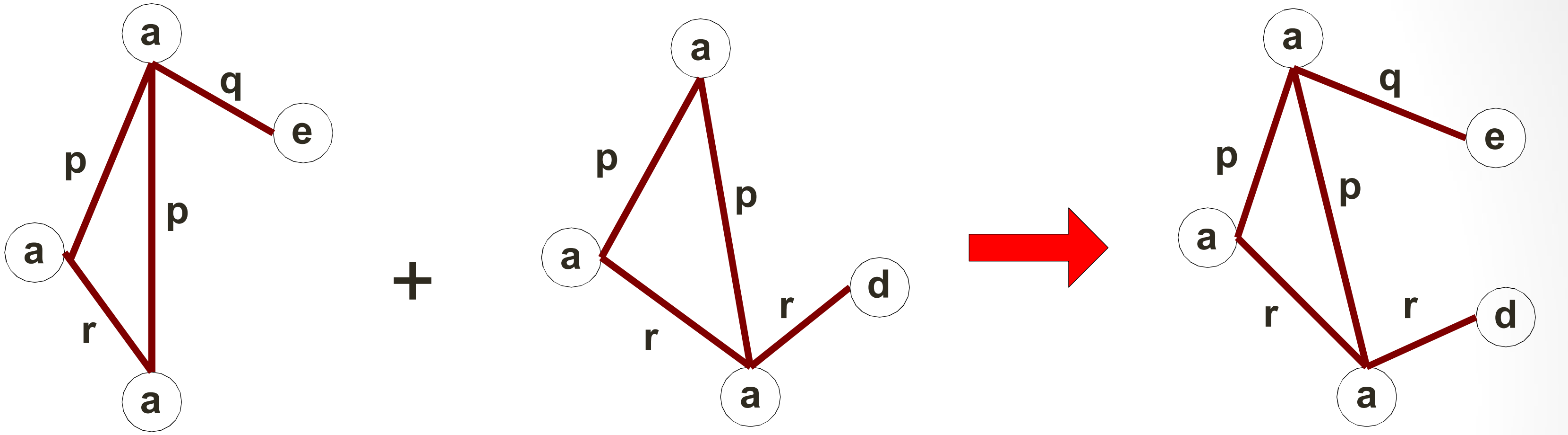
Challenges

- 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



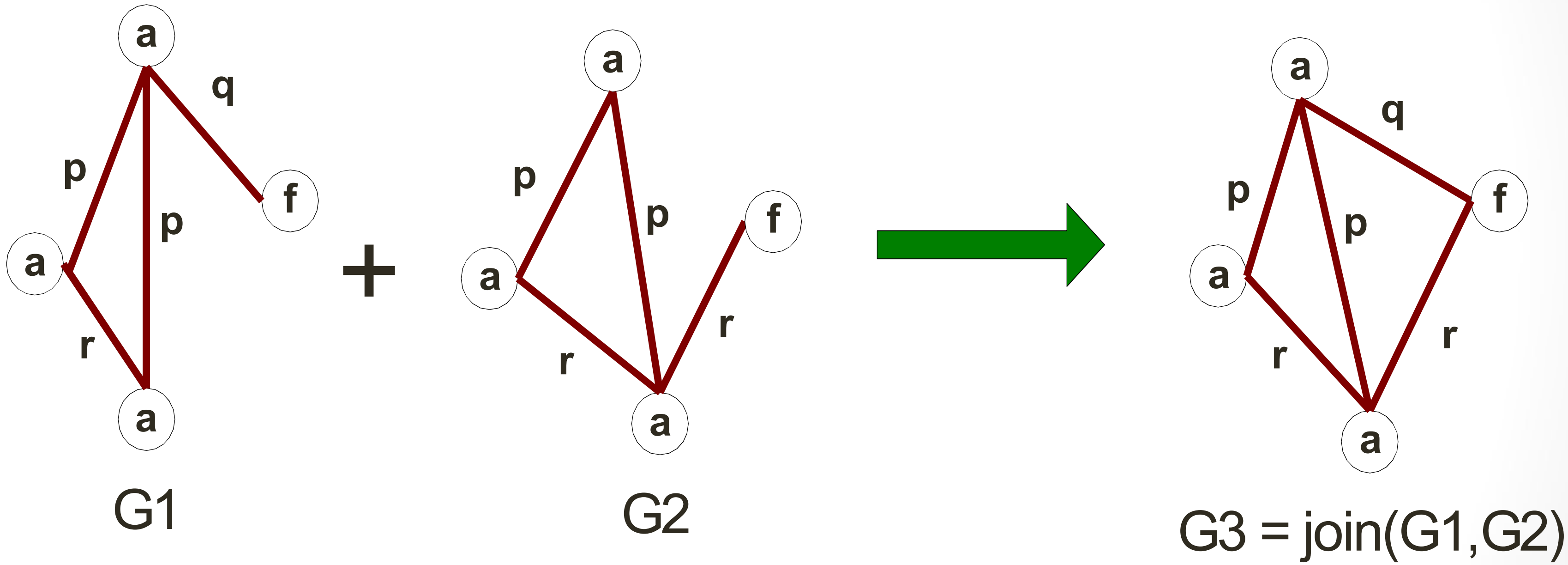
G1 G2 G3 = join(G1,G2)

$$M_{G1} = \begin{pmatrix} 0 & p & p & q \\ p & 0 & r & 0 \\ p & r & 0 & 0 \\ q & 0 & 0 & 0 \end{pmatrix}$$

$$M_{G2} = \begin{pmatrix} 0 & p & p & 0 \\ p & 0 & r & 0 \\ p & r & 0 & r \\ 0 & 0 & r & 0 \end{pmatrix}$$

$$M_{G3} = \begin{pmatrix} 0 & p & p & 0 & q \\ p & 0 & r & 0 & 0 \\ p & r & 0 & r & 0 \\ 0 & 0 & r & 0 & 0 \\ q & 0 & 0 & 0 & 0 \end{pmatrix}$$

Edge Growing

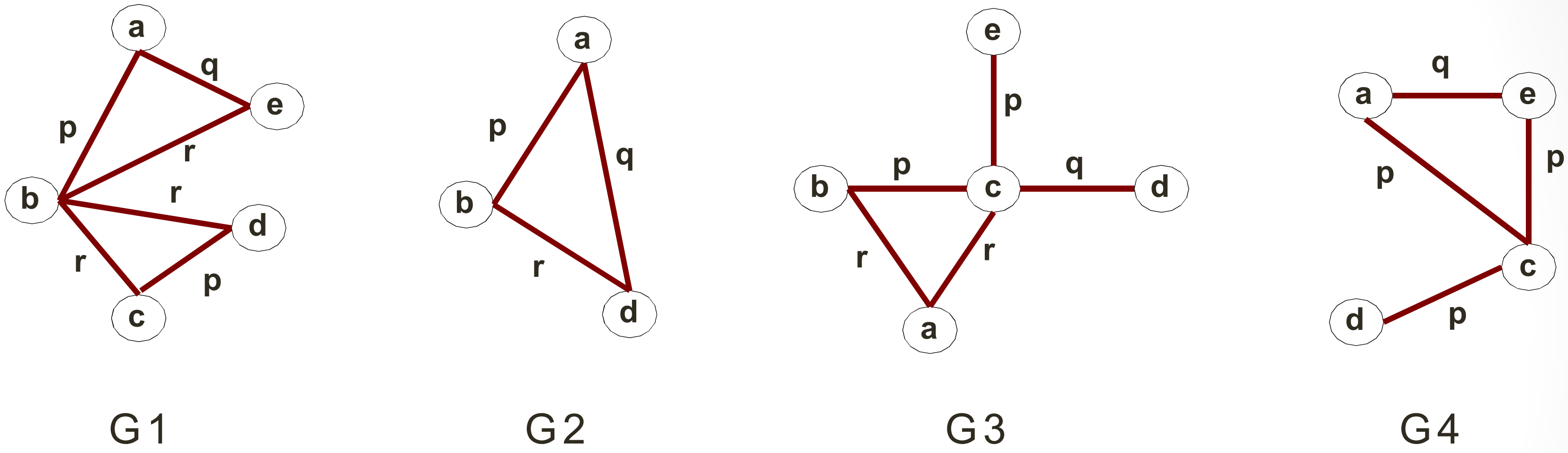


Apriori-like Algorithm

- 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

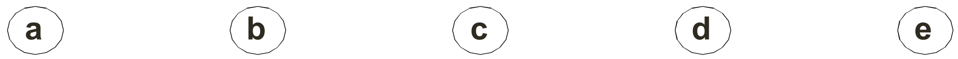


	(a,b,p)	(a,b,q)	(a,b,r)	(b,c,p)	(b,c,q)	(b,c,r)	…	(d,e,r)
G1	1	0	0	0	0	1	…	0
G2	1	0	0	0	0	0	…	0
G3	0	0	1	1	0	0	…	0
G4	0	0	0	0	0	0	…	0

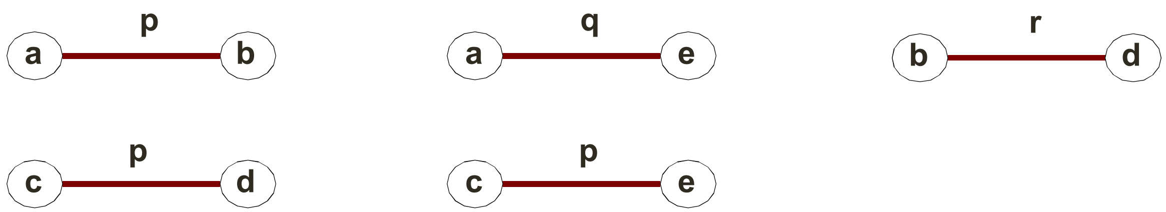
Example

Minimum support count = 2

k=1
Frequent
Subgraphs



k=2
Frequent
Subgraphs



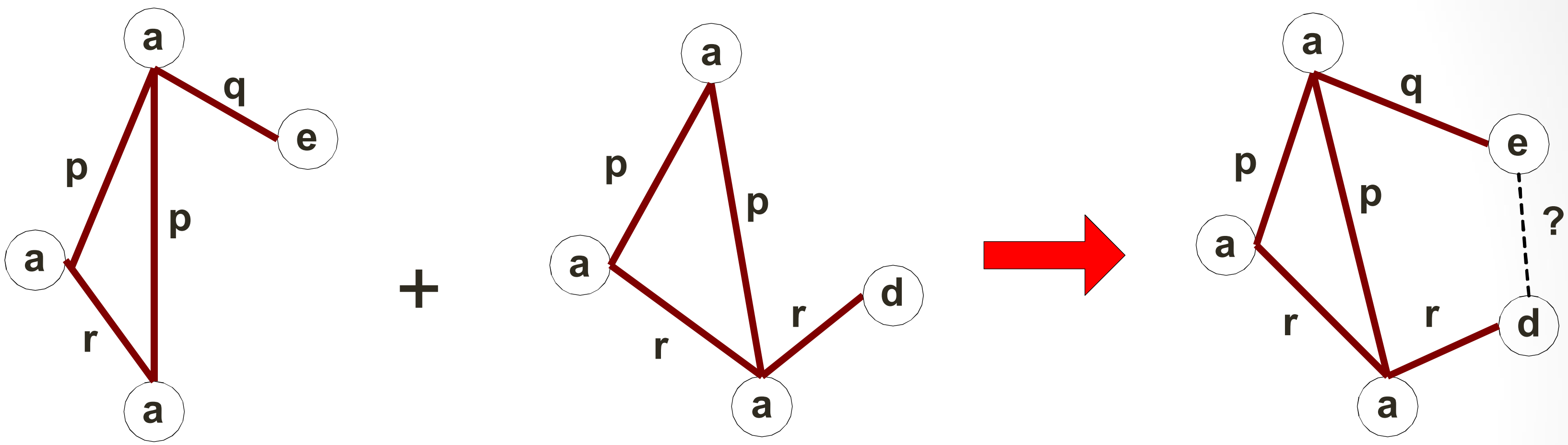
k=3
Candidate
Subgraphs

(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)

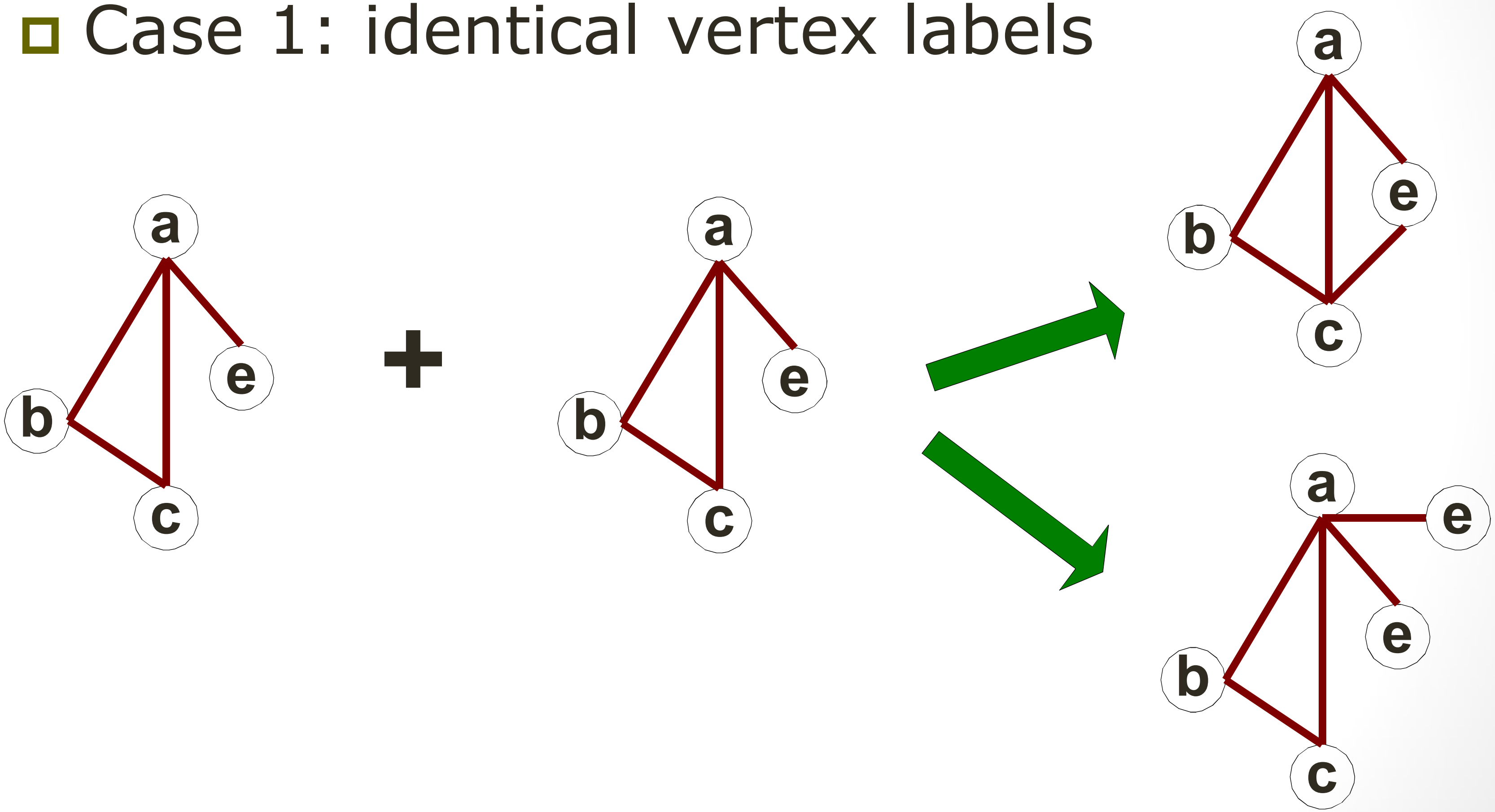


G1 G2 G3 = join(G1,G2)

$$M_{G1} = \begin{pmatrix} 0 & p & p & q \\ p & 0 & r & 0 \\ p & r & 0 & 0 \\ q & 0 & 0 & 0 \end{pmatrix} \qquad M_{G2} = \begin{pmatrix} 0 & p & p & 0 \\ p & 0 & r & 0 \\ p & r & 0 & r \\ 0 & 0 & r & 0 \end{pmatrix} \qquad M_{G3} = \begin{pmatrix} 0 & p & p & 0 & q \\ p & 0 & r & 0 & 0 \\ p & r & 0 & r & 0 \\ 0 & 0 & r & 0 & ? \\ q & 0 & 0 & ? & 0 \end{pmatrix}$$

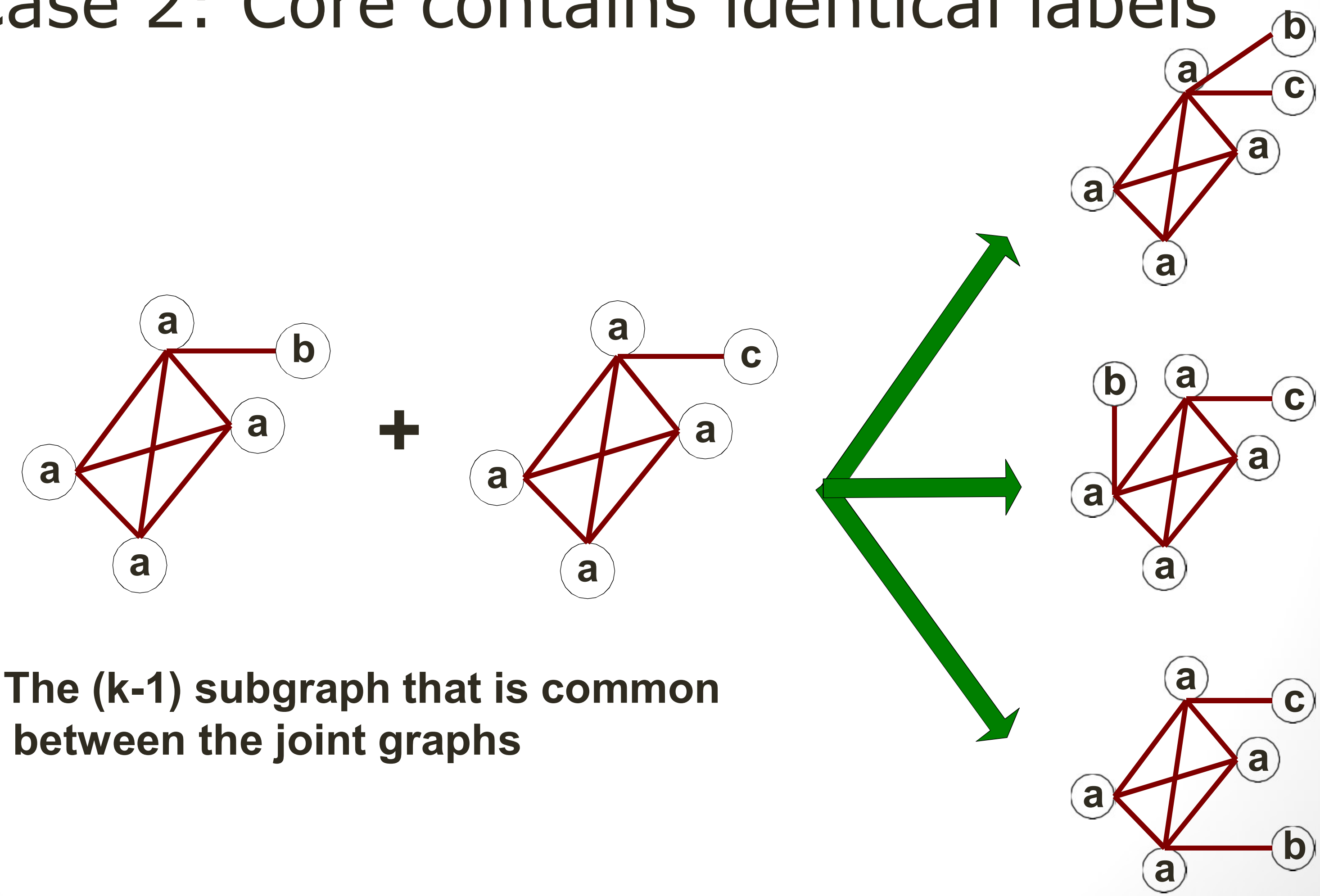
Multiplicity of Candidates (Edge growing)

- Case 1: identical vertex labels



Multiplicity of Candidates (Edge growing)

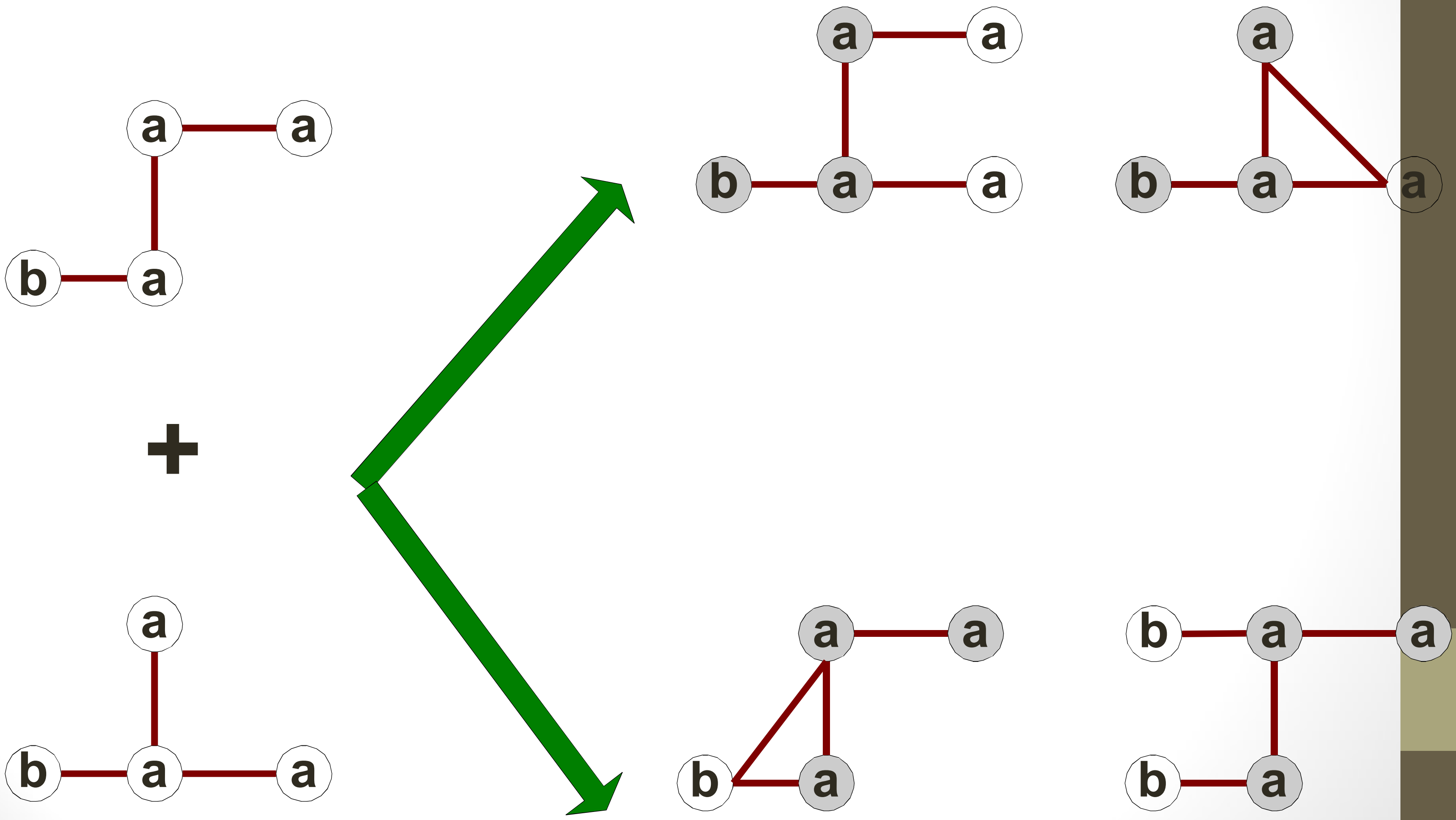
- Case 2: Core contains identical labels



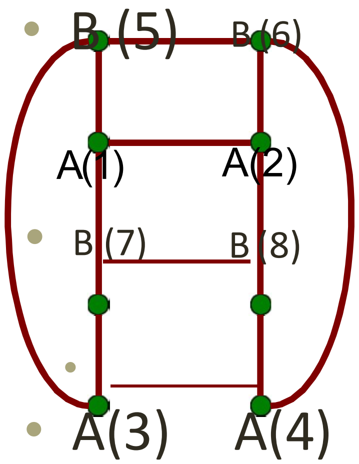
Core: The (k-1) subgraph that is common between the joint graphs

Multiplicity of Candidates (Edge growing)

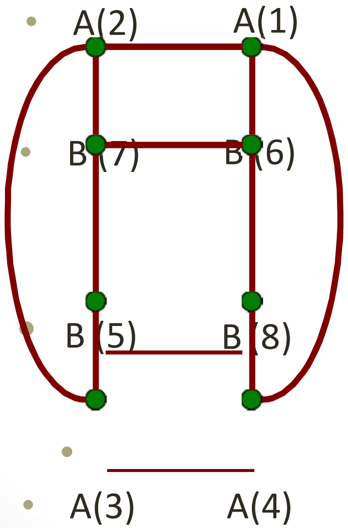
- Case 3: Core multiplicity



Matrix Representatio n



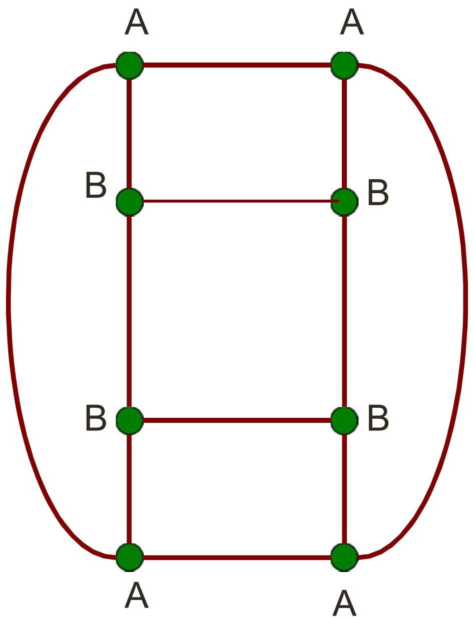
	A(1)	A(2)	A(3)	A(4)	B(5)	B(6)	B(7)	B(8)
A(1)	1	1	1	0	1	0	0	0
A(2)	1	1	0	1	0	1	0	0
A(3)	1	0	1	1	0	0	1	0
A(4)	0	1	1	1	0	0	0	1
B(5)	1	0	0	0	1	1	1	0
B(6)	0	1	0	0	1	1	0	1
B(7)	0	0	1	0	1	0	1	1
B(8)	0	0	0	1	0	1	1	1

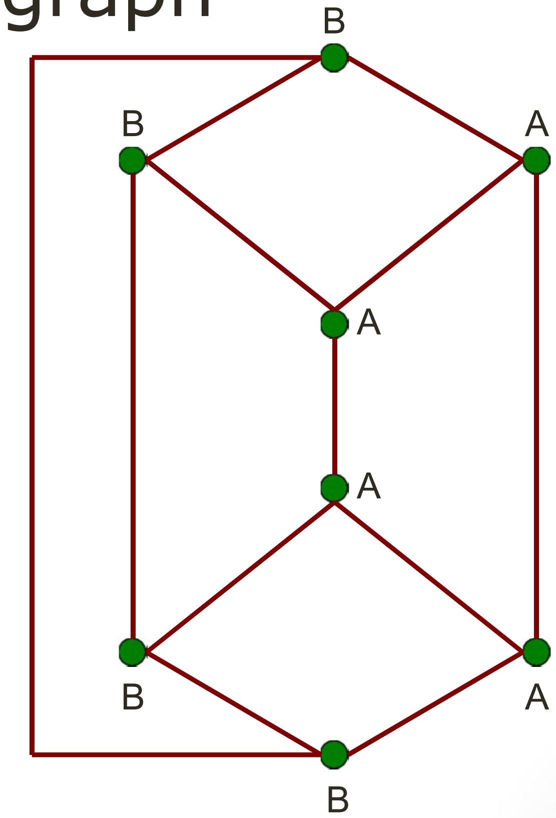


	A(1)	A(2)	A(3)	A(4)	B(5)	B(6)	B(7)	B(8)
A(1)	1	1	0	1	0	1	0	0
A(2)	1	1	1	0	0	0	1	0
A(3)	0	1	1	1	1	0	0	0
A(4)	1	0	1	1	0	0	0	1
B(5)	0	0	1	0	1	0	1	1
B(6)	1	0	0	0	0	1	1	1
B(7)	0	1	0	0	1	1	1	0
B(8)	0	0	0	1	1	1	0	1

Graph Isomorphism

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



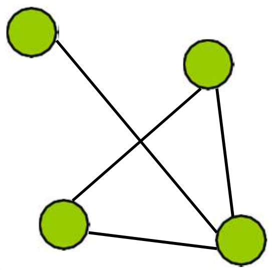


Graph Isomorphism

- 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 $(k\text{-}1)$-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



$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

String: 0010001111010110

Canonical: 0111101011001000

Frequent Subgraph Mining Approaches

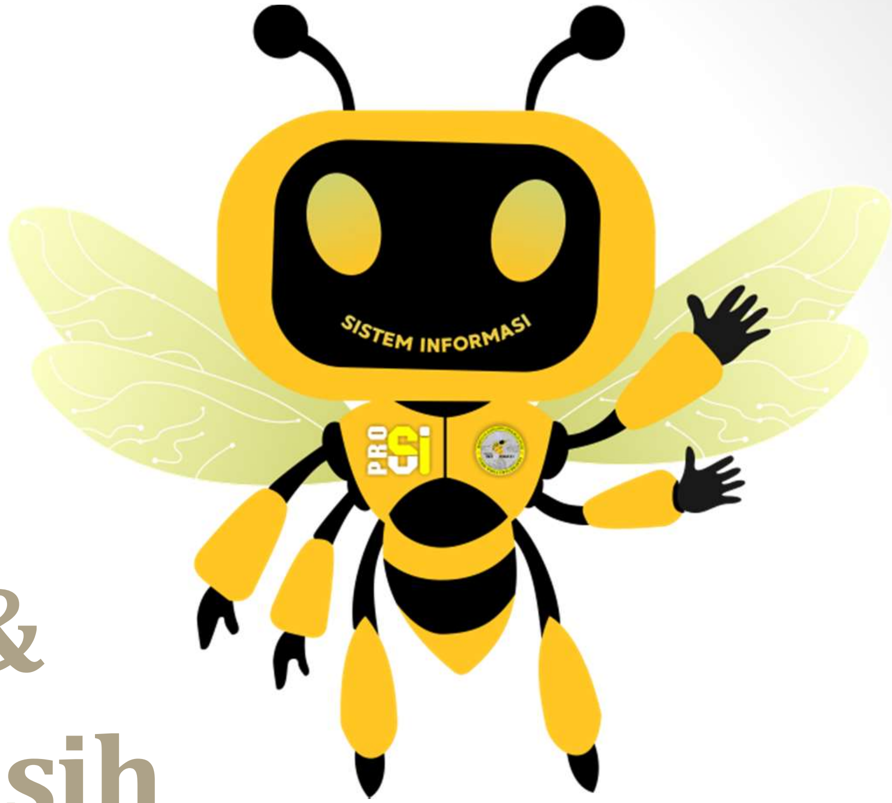
- Apriori-based approach
  - AGM/AcGM: Inokuchi, et al. (PKDD'00)
  - FSG: Kuramochi and Karypis (ICDM'01)
  - PATH#: Vanetik and Gudes (ICDM'02, ICDM'04)
  - FFSM: Huan, et al. (ICDM'03)
- Pattern growth approach
  - MoFa, Borgelt and Berthold (ICDM'02)
  - gSpan: Yan and Han (ICDM'02)
  - Gaston: Nijssen and Kok (KDD'04)

Properties of Graph Mining Algorithms

- Search order
  - breadth vs. depth
- Generation of candidate subgraphs
  - apriori vs. pattern growth
- Elimination of duplicate subgraphs
  - passive vs. active
- Support calculation
  - embedding store or not
- Discover order of patterns
  - path → tree → graph

Mining Frequent Subgraphs in a Single Graph

- A large graph is more interesting
  - Software, social network, Internet, biological networks
- What are the frequent subgraphs in a single graph?
  - How to define frequency concept?
  - Apriori property



Sekian &
Terima Kasih