“Association mining - 2”

Salvatore Orlando

Continuous and Categorical Attributes

How to apply association analysis formulation to non-asymmetric binary variables, like “Buy”, or continuous like “Session length”, or categorical like “Country” ?

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:

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

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 = Netscape

Handling Categorical Attributes

- **Potential Issues**
  - **What if an 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**, since these items, like “Buy=No”, should be associated with every itemset returned

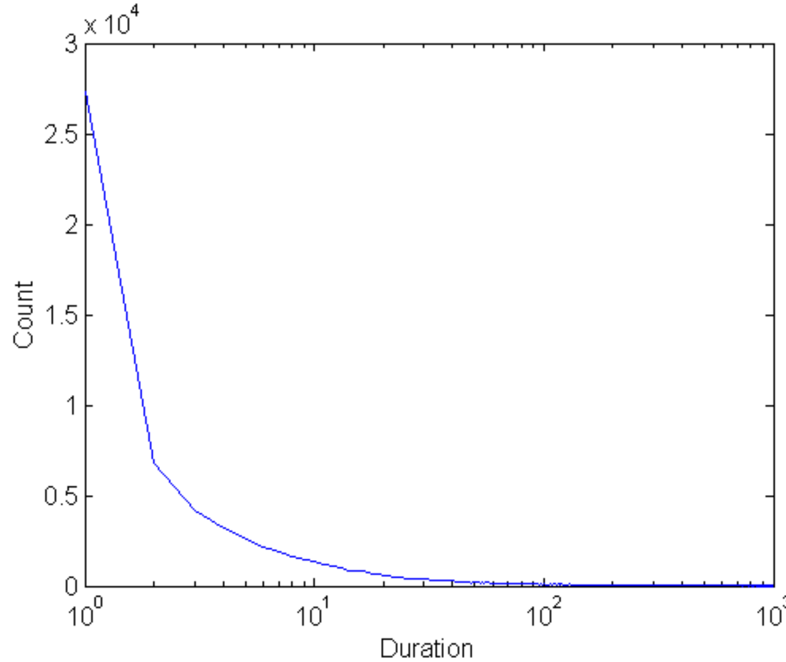
Handling Continuous Attributes

- **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**

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)} → {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 with small supports**
  - **Expensive and too many rules**

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

  {Refund = No, (51K ≤ Income ≤ 52K)} → {Cheat = No}

  {Refund = No, (50K ≤ Income ≤ 60K)} → {Cheat = No}

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 properly, as users want association among words not ranges of words (like the following rule)**

$$\{(2 \leq W2 \leq 3)\} \rightarrow \{W1 = 2\}$$

Non-discretization methods: Min-Apriori (Han et al.)

Document-term matrix, where each entry is the (normalized) frequency count of a word 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

Example:

W1 and W2 tend to appear together in the same document

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!
  - Solution
    - Normalize the word vectors (by column)
    - Each word has an overall 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:**

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

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:

Sup(W1,W2,W3)

= 0 + 0 + 0 + 0 + 0.17

= 0.17

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

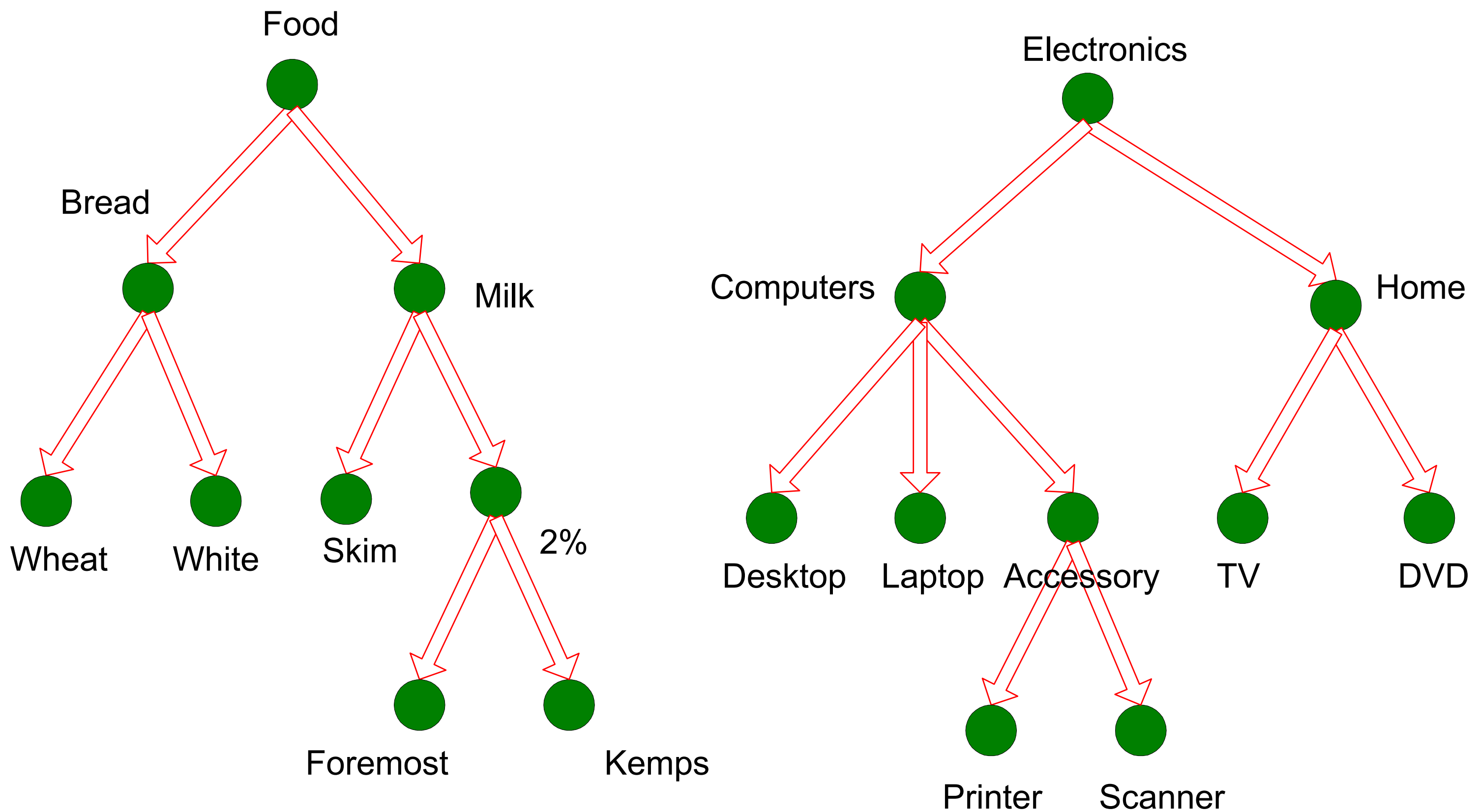
Example (anti-monotone support):

Sup(W1) = 0.4 + 0 + 0.4 + 0 + 0.2 = 1

Sup(W1, W2) = 0.33 + 0 + 0.4 + 0 + 0.17 = 0.9

Sup(W1, W2, W3) = 0 + 0 + 0 + 0 + 0.17 = 0.17

Multi-level Association Rules



- **In market basket analysis, the** concept hierarchy **becomes an** *item taxonomy*
  - **Edges are "is-a" relationships**
- **Directed Acyclic Graph**
  - **Parent-Child   Ancestor-Descendant**

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, etc.

    are indicative of association between milk and bread

Multi-level Association Rules

- **Approach 1:**
  - **Extend current association rule formulation by augmenting each transaction with higher level items, and then apply standard Apriori algorithm**

    Original Transaction, with items at the lowest level of the hierarchy:

    {skim milk, wheat bread}

    Augmented Transaction, adding the ancestors:

    {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**
  - **Redundant rules**
    - **Easily remove redundant itemsets like:** *{skim milk, milk, food}*

Multi-level Association Rules

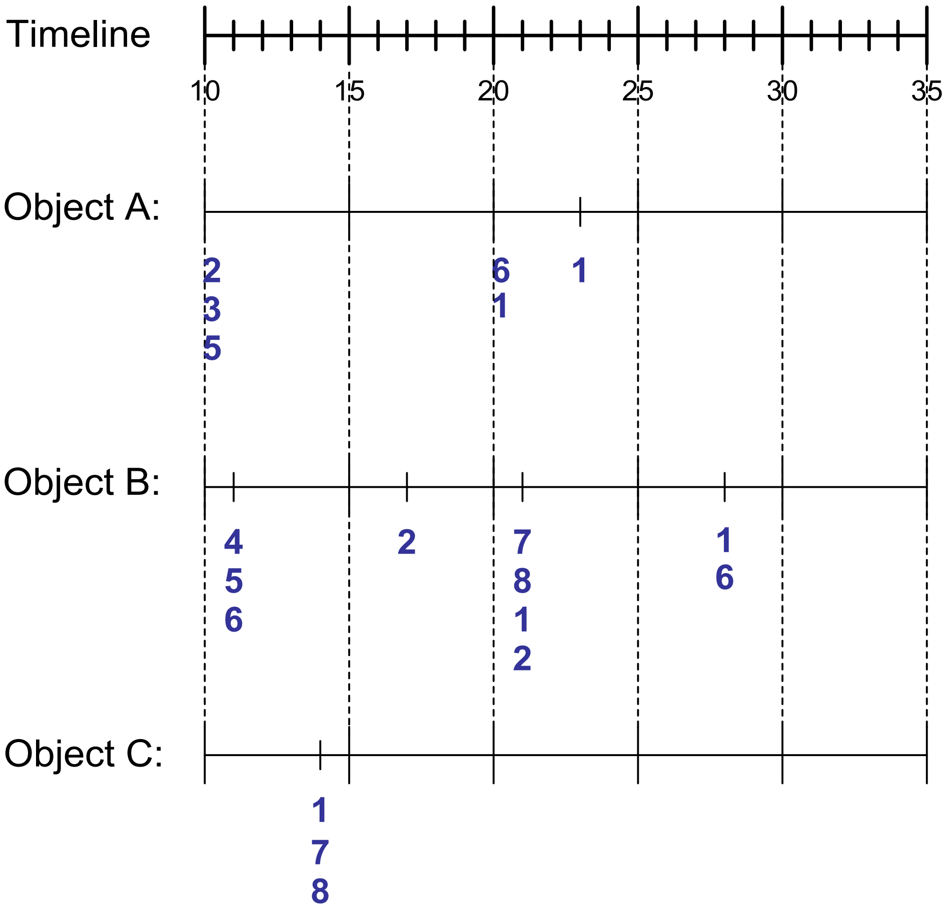
- **Approach 2:**
  - Generate frequent patterns at highest level first
  - Then, generate frequent patterns at the next highest level, and so on, by reducing the support threshold
- **Issues:**
  - I/O requirements will increase dramatically because we need to perform more passes over the data
  - May miss some potentially interesting cross-level association patterns

Sequence Data

Sequence Database:

Sequence

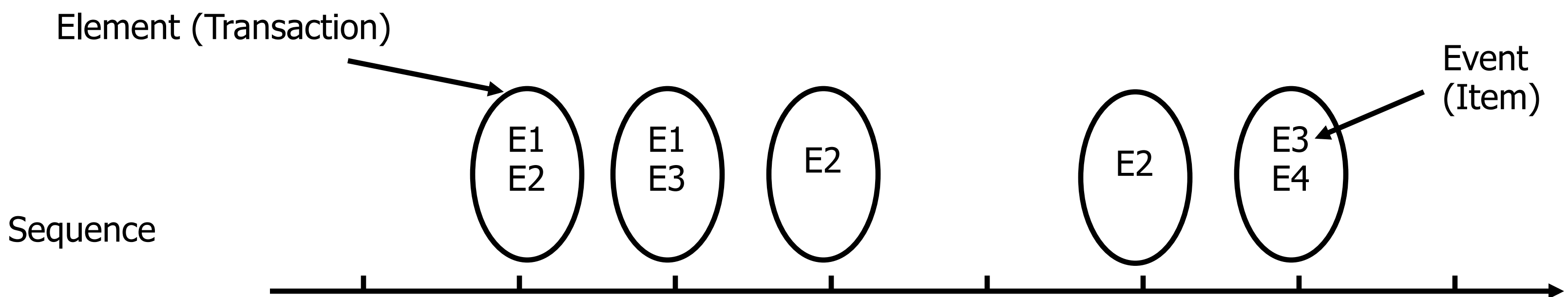
Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



Element composed of multiple **events/items**, i.e., each record is a **transaction** associated with a timestamp and an object

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



Examples of Mined 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:**
  (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 Sequence

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

$$s = < e_1 \; e_2 \; e_3 \; \ldots >$$

  - 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 (even location)

- Length of a sequence, $|s|$, is given by the number of elements/transactions in the sequence
- A k-sequence is a sequence that *globally* contains k events (items)

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** $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$

The *timestamps* associated with the transactions are not shown

subsequence	DATA sequence	Contained
< (3) (4 5) (8) >	< (7) (3 8) (9) (4 5 6) (8) >	Y
< (3) (5) >	< (3 5) >	N
< (3) (5) >	< (3 5) (3 5) (3 4 5) >	Y

- **The support of a subsequence w is defined as the fraction of input data sequences that contain w**
  - **Note that a sequence can be contained in a given "data sequence" in different ways (see the last example)**
  - **A *sequential pattern* is frequent if it is a <u>subsequence</u> of σ input sequences (σ = support of the pattern), and $\sigma \geq$ *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

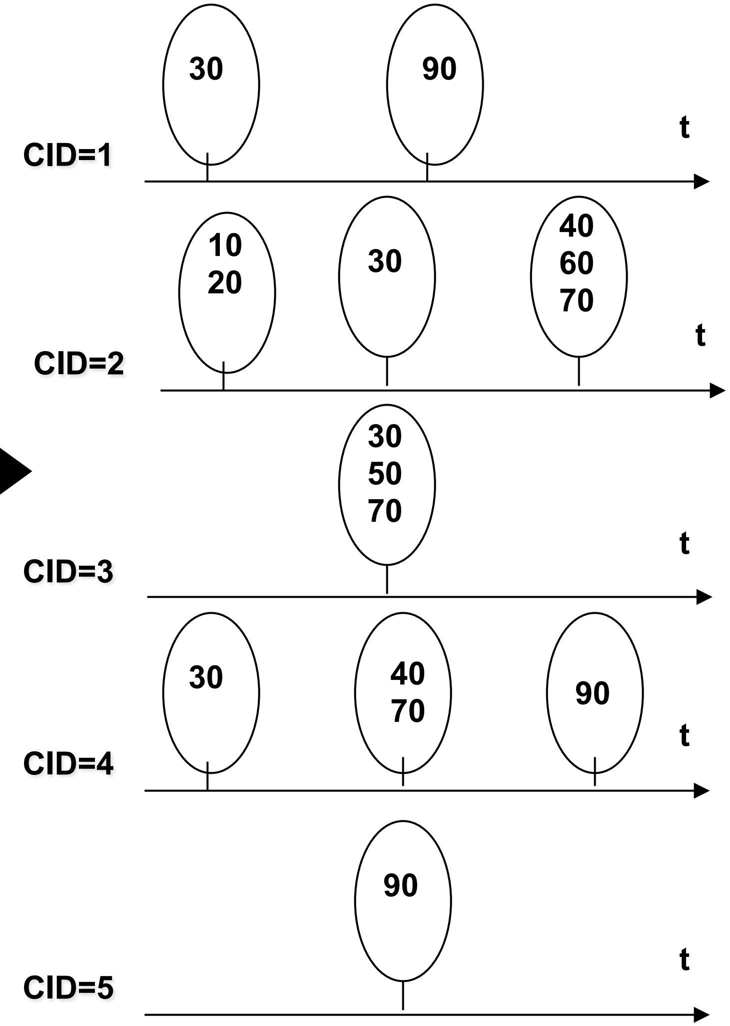
<{a} {d e} {i}>

Answer :

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

Sequential Pattern Mining: Example

Customer Id	Transaction Time	Items Bought
1	June 25,93	30
1	June 30,93	90
2	June 10,93	10,20
2	June 15,93	30
2	June 20,93	40,60,70
3	June 25,93	30,50,70
4	June 25,93	30
4	June 30,93	40,70
4	July 25,93	90
5	June 12,93	90



MinSupp=40%, i.e. 2 customers of 5:

<30><90>	(supported by 1,4)
<30><40,70>	(supported by 2,4)
<10 20 ><30>	Infrequent

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: $i_1, i_2, i_3, \dots, i_n$**
- **Candidate 1-subsequences:**

  $\langle\{i_1\}\rangle, \langle\{i_2\}\rangle, \langle\{i_3\}\rangle, \dots, \langle\{i_n\}\rangle$

- **Candidate 2-subsequences:**

  $\langle\{i_1, i_2\}\rangle, \langle\{i_1, i_3\}\rangle, \dots, \langle\{i_1\}\ \{i_1\}\rangle, \langle\{i_1\}\ \{i_2\}\rangle, \dots, \langle\{i_{n-1}\}\ \{i_n\}\rangle$

- **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 <{i_1}> and <{i_2}> will produce two candidate 2-sequences: <{i_1} {i_2}> and <{i_1 i_2}>**
- **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 (suffix of w_1 = prefix of 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 the sequences (in red the common portion)**
  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 (in red the common portion)**
  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

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} >

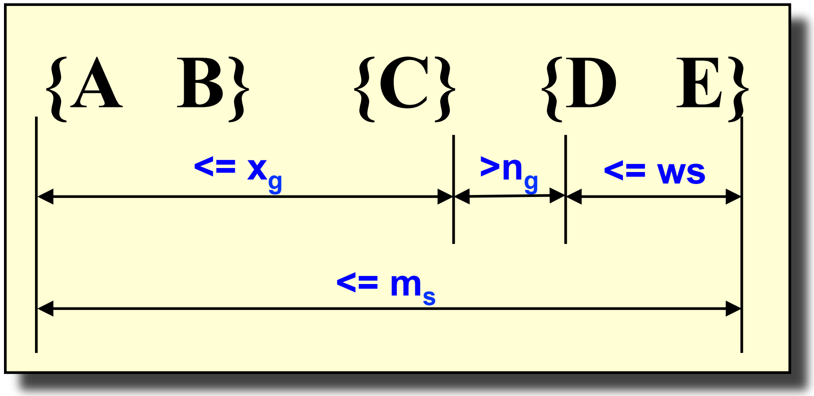
Candidate Generation

< {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

w_s: window size

m_s: maximum span

$x_g = 2$, $n_g = 0$, $w_s = 1$, $m_s = 5$

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

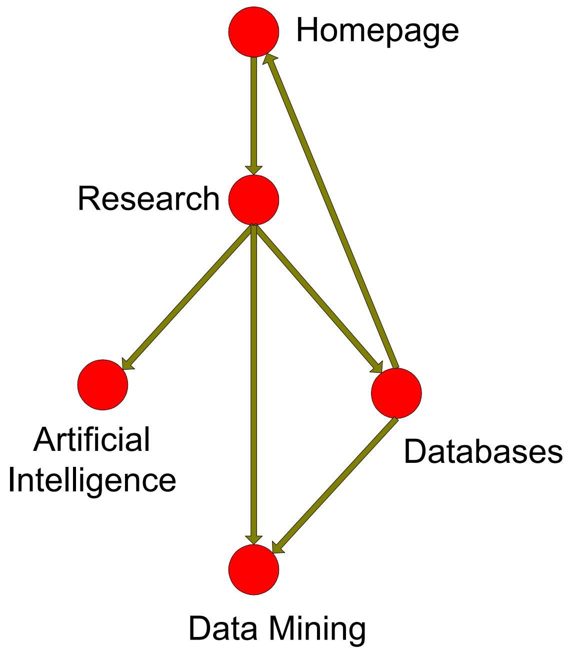
Mining Sequential Patterns with Timing Constraints

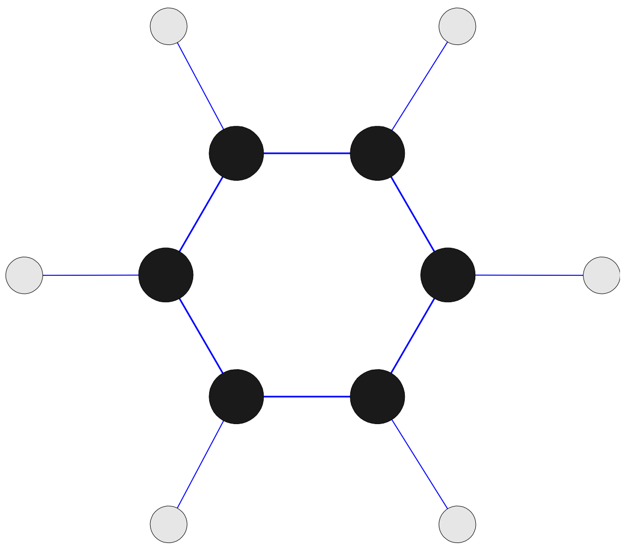
- **Approach 1:**
  - **Mine sequential patterns without timing constraints**
  - **Postprocess the discovered patterns**

- **Approach 2:**
  - **Modify GSP to directly prune candidates that violate timing constraints**

Frequent Subgraph Mining

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

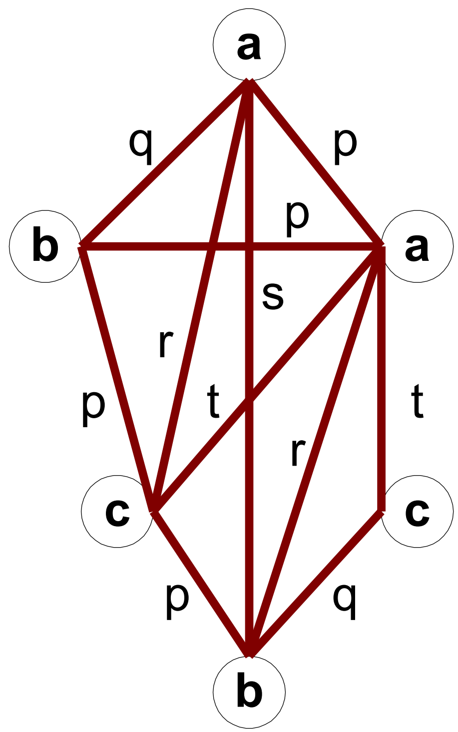




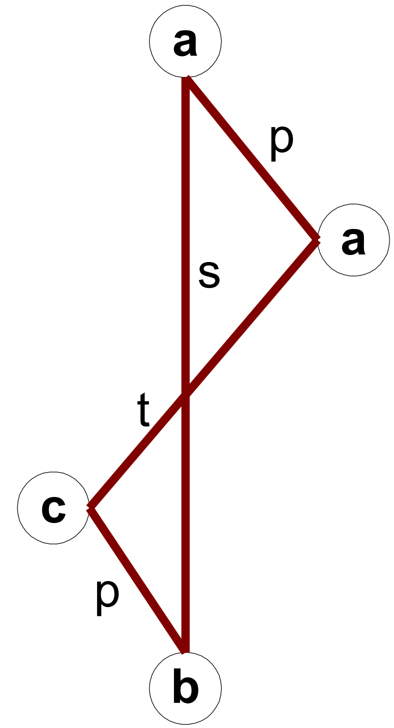
Graph representation of entities

Application	Graphs	Vertices	Edges
Web mining	Web browsing patterns	Web pages	Hyperlinks between pages
Computational chemistry	Structure of chemical compounds	Atoms or ions	Bond between atoms or ions
Sematic Web	Collection of XML documents	XML elements	Parent-child relationships between elements
Bioinformatics	Protein structures	Amino acids	Contact residue
Network computing	Computer networks	Computer and servers	Interconnections between machines

Graph Definitions



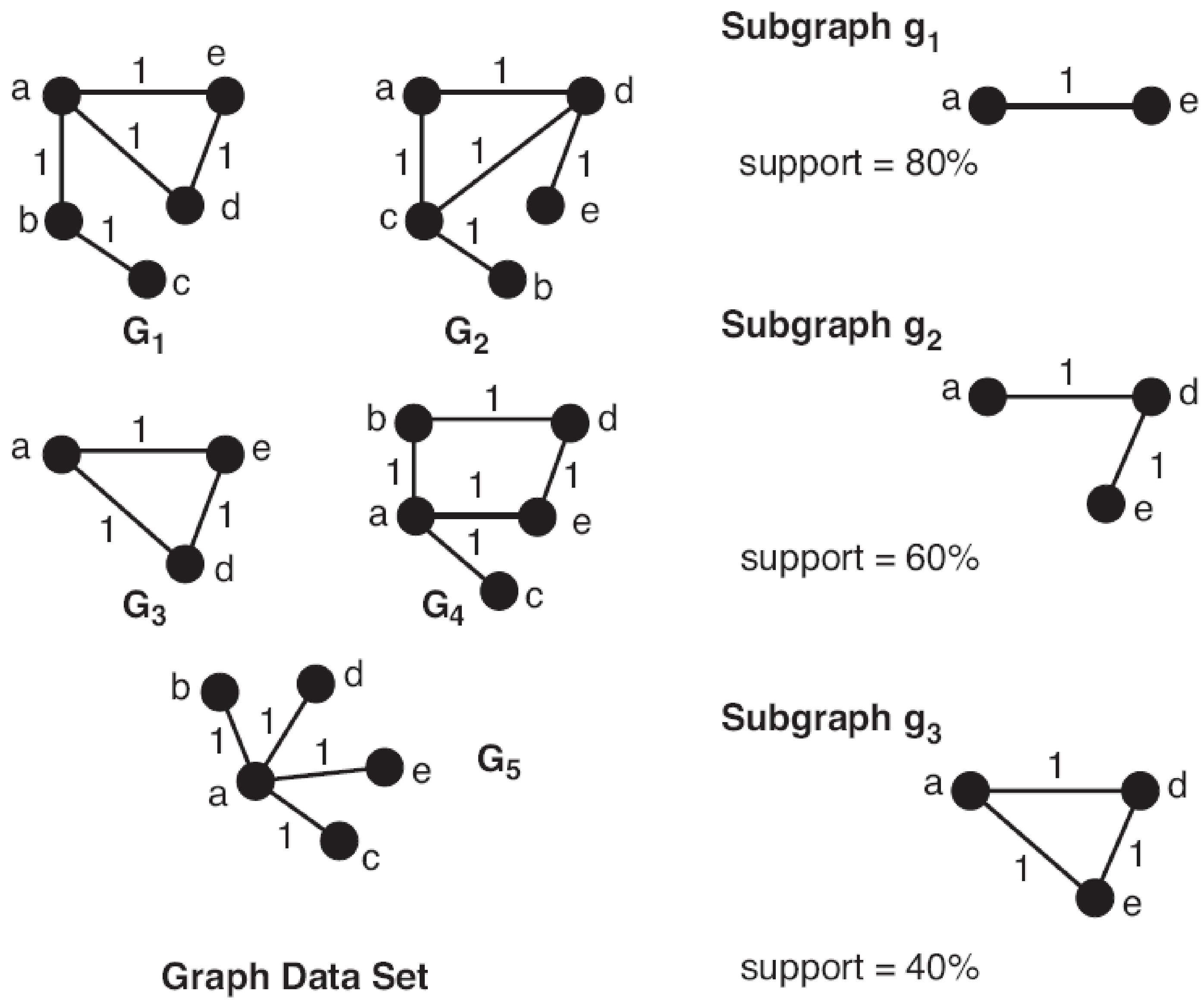
(a) Labeled Graph



(b) Subgraph

Frequent Subgraph Mining

Computing the Subgraph Support

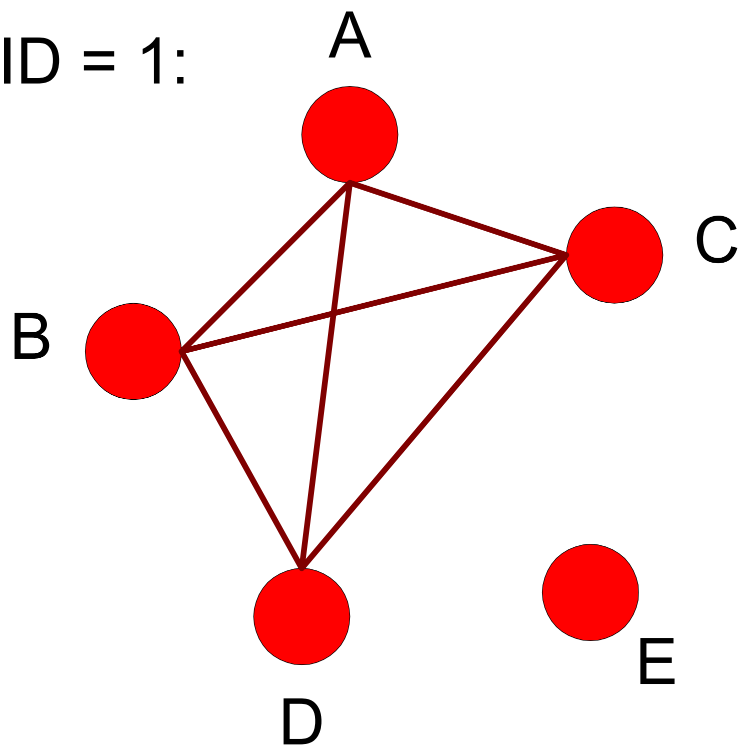


Representing Transactions as (unlabeled) Graphs

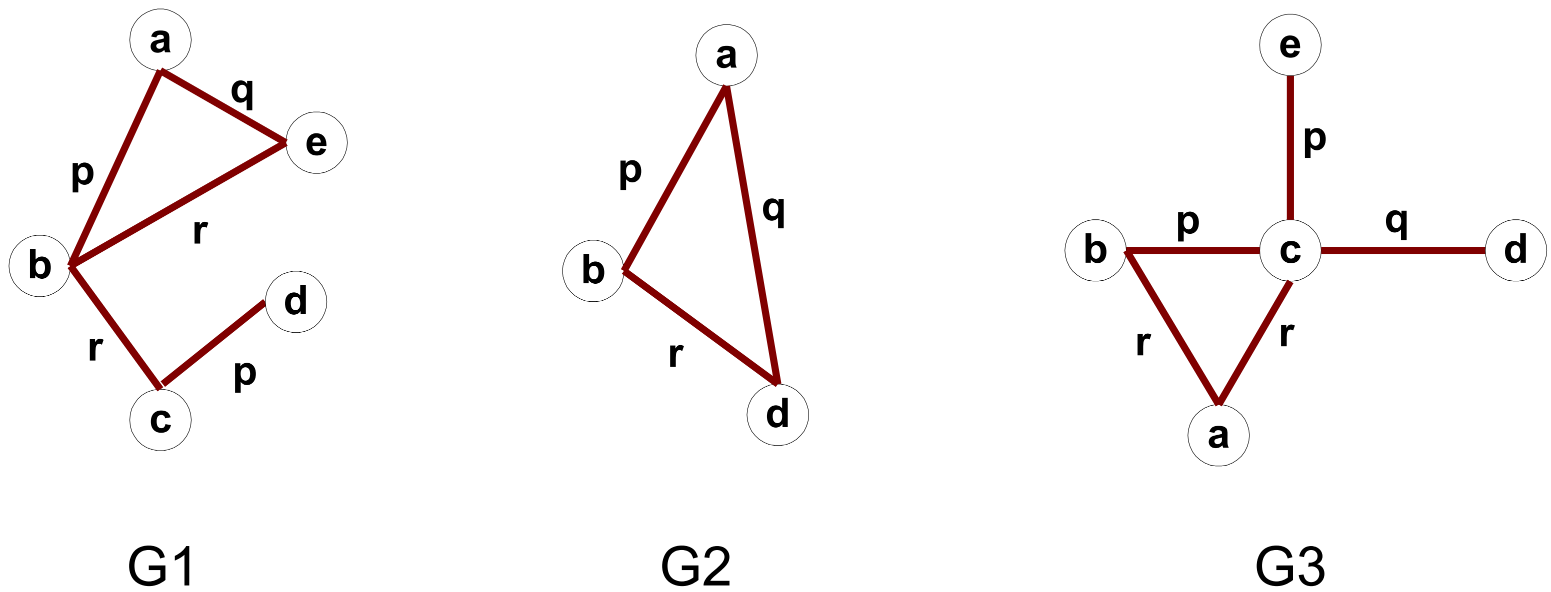
- **Each transaction is a clique of items**

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

- Nodes 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?

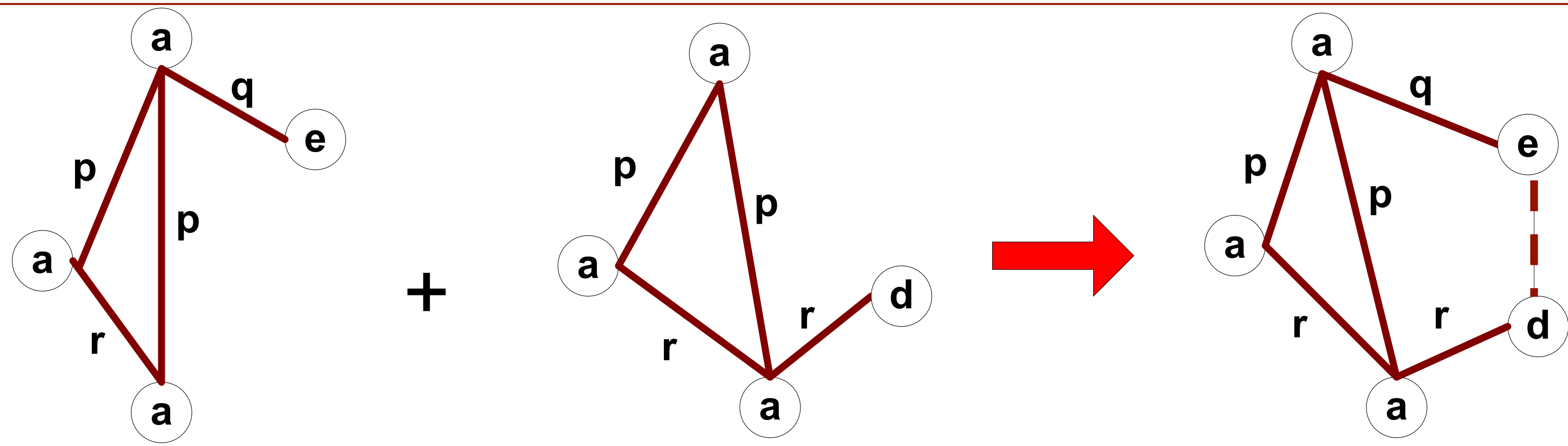
Challenges…

- **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**

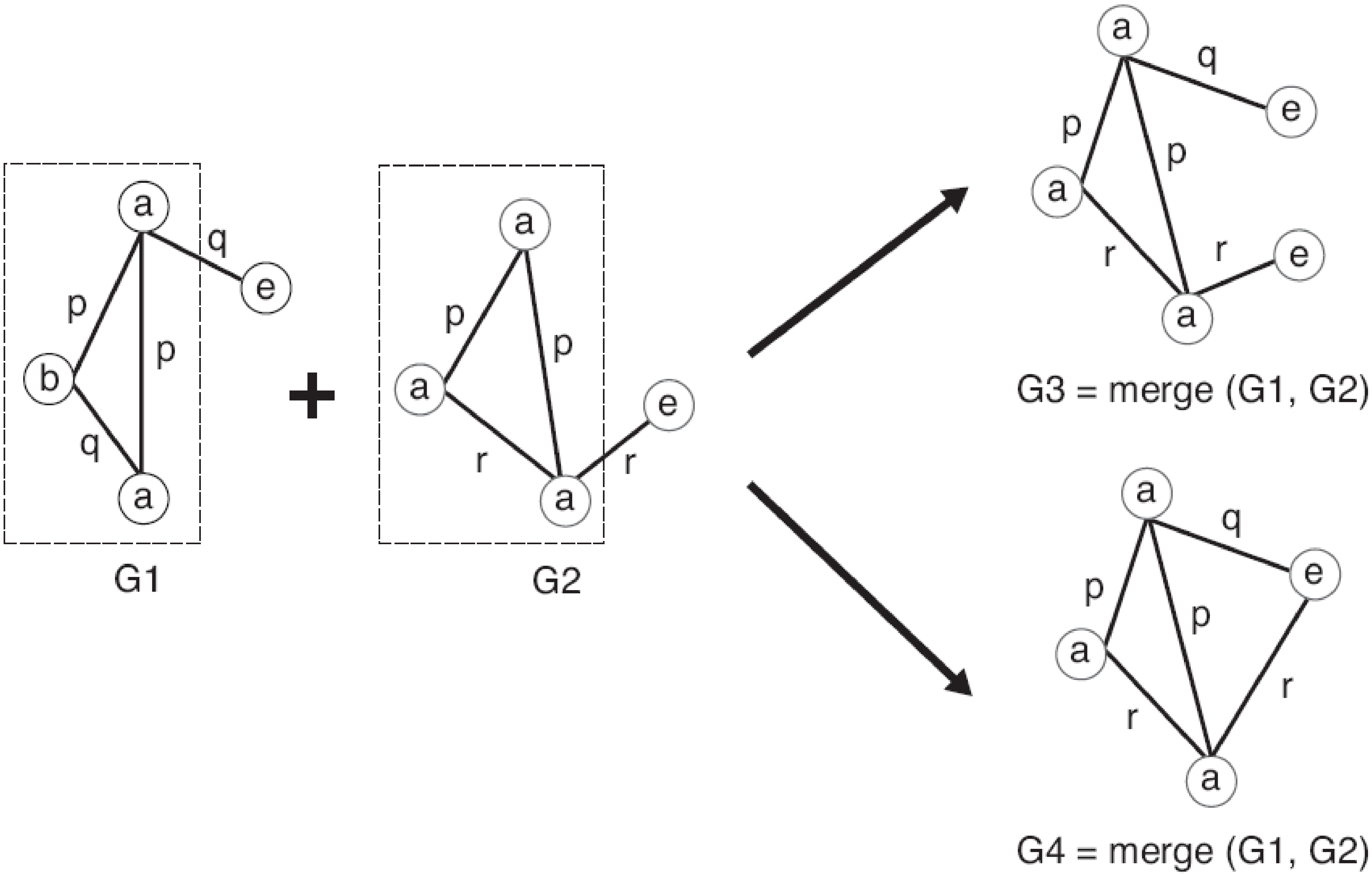
Vertex Growing



G1 G2 G3 = join(G1,G2)

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

Edge Growing

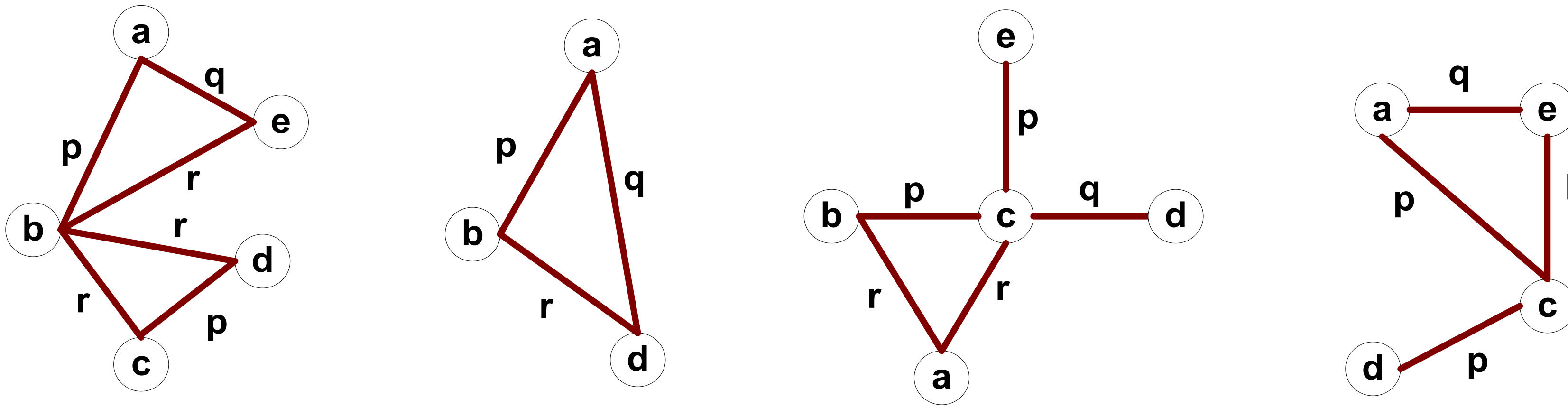


Apriori-like Algorithm

- **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



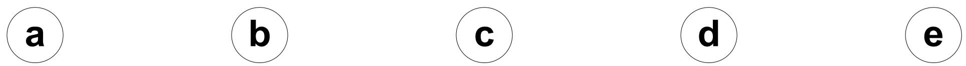
G1 G2 G3 G4

	(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

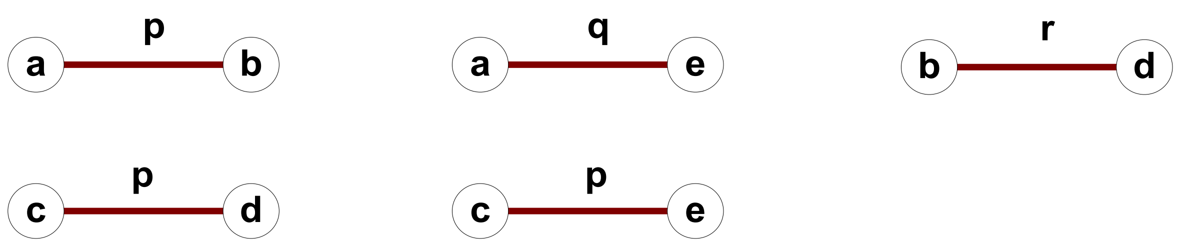
Example

Minimum support count = 2

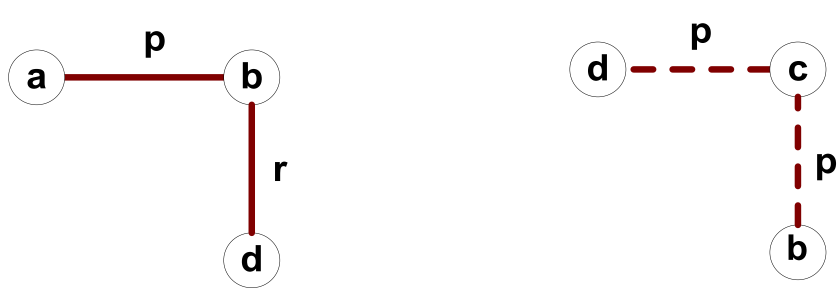
k=1
Frequent
Subgraphs



k=2
Frequent
Subgraphs



k=3
Candidate
Subgraphs

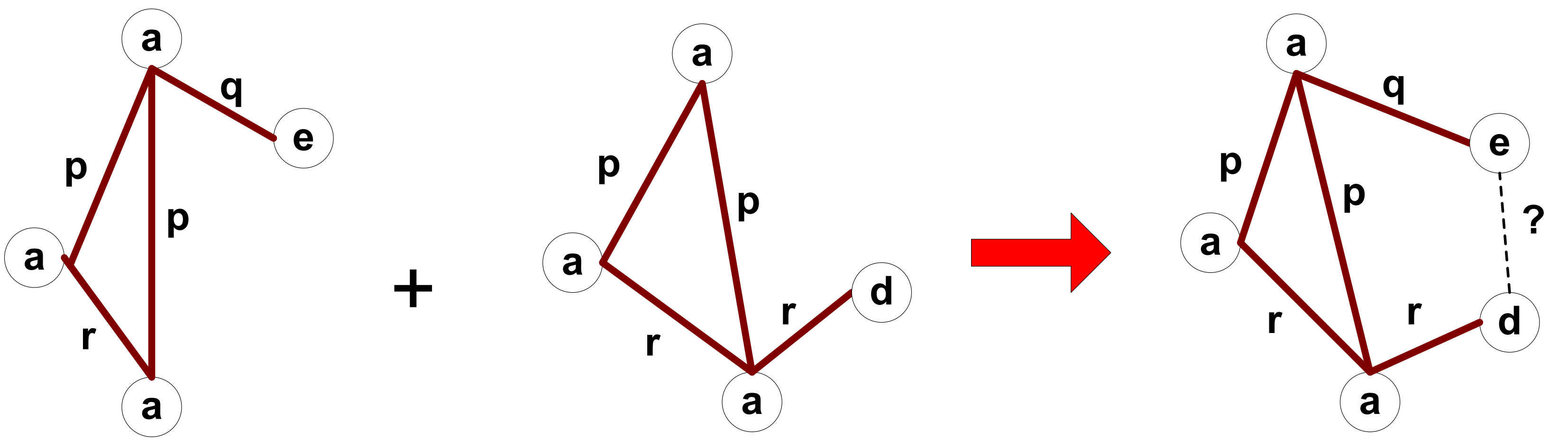


(Pruned candidate)

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)

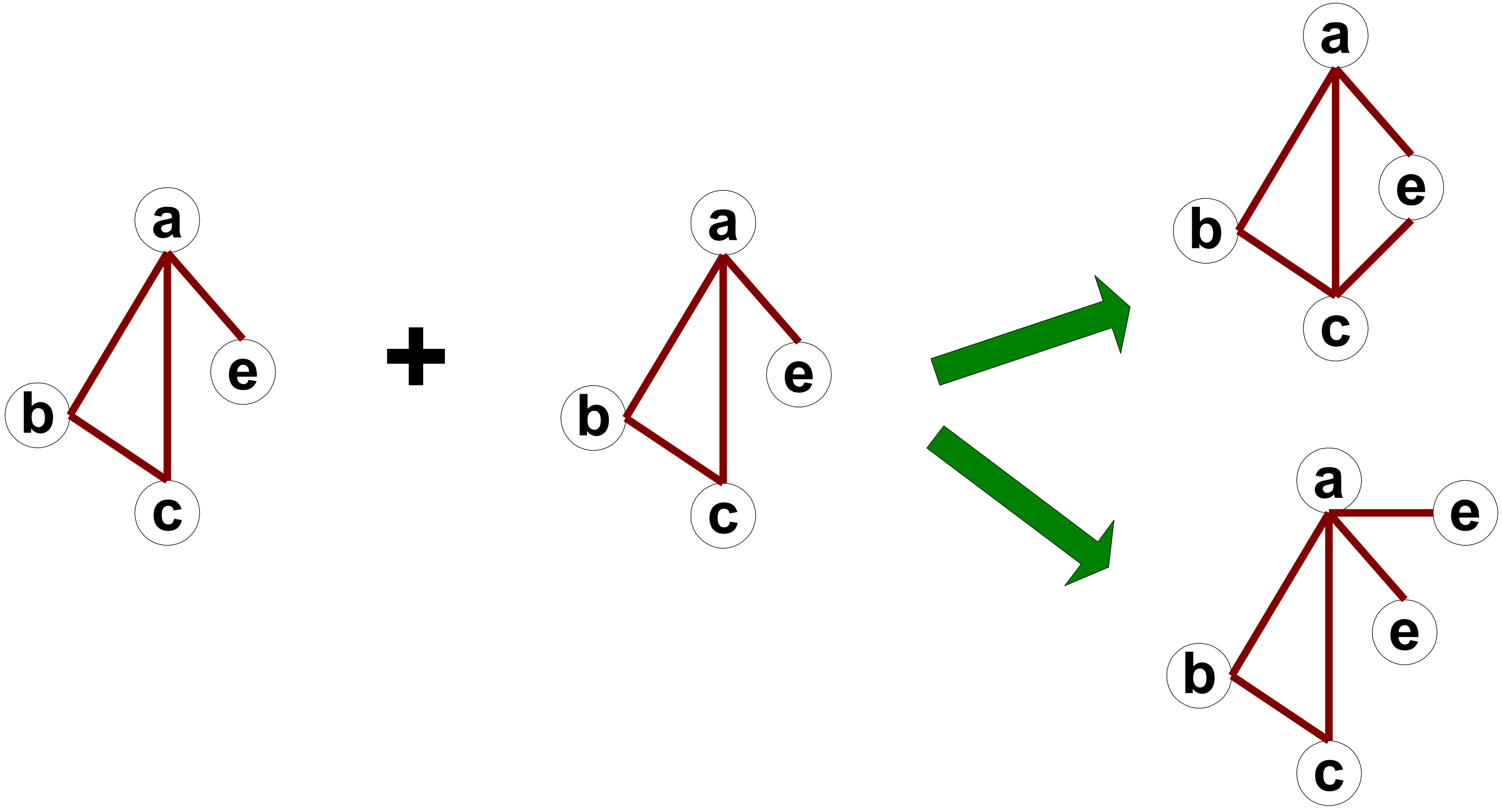


G1 G2 G3 = join(G1,G2)

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

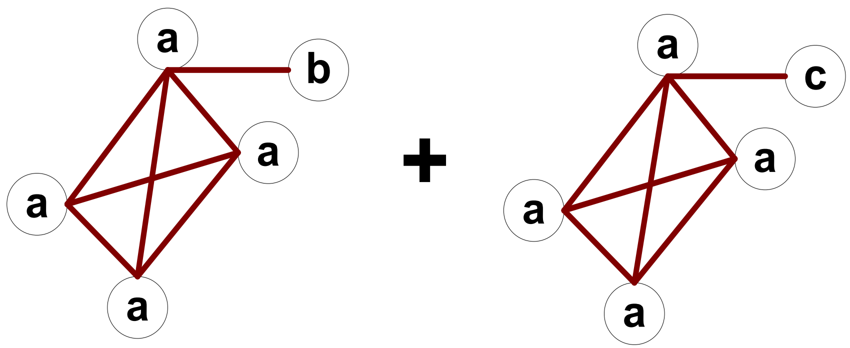
Multiplicity of Candidates (Edge growing)

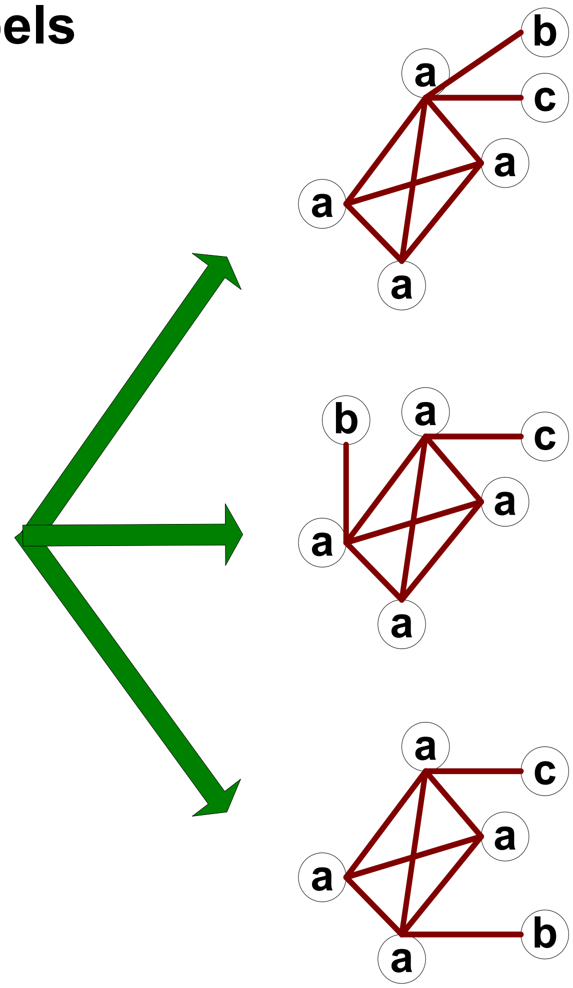
- **Case 1: identical vertex labels**



Multiplicity of Candidates (Edge growing)

- **Case 2: Core contains identical labels**

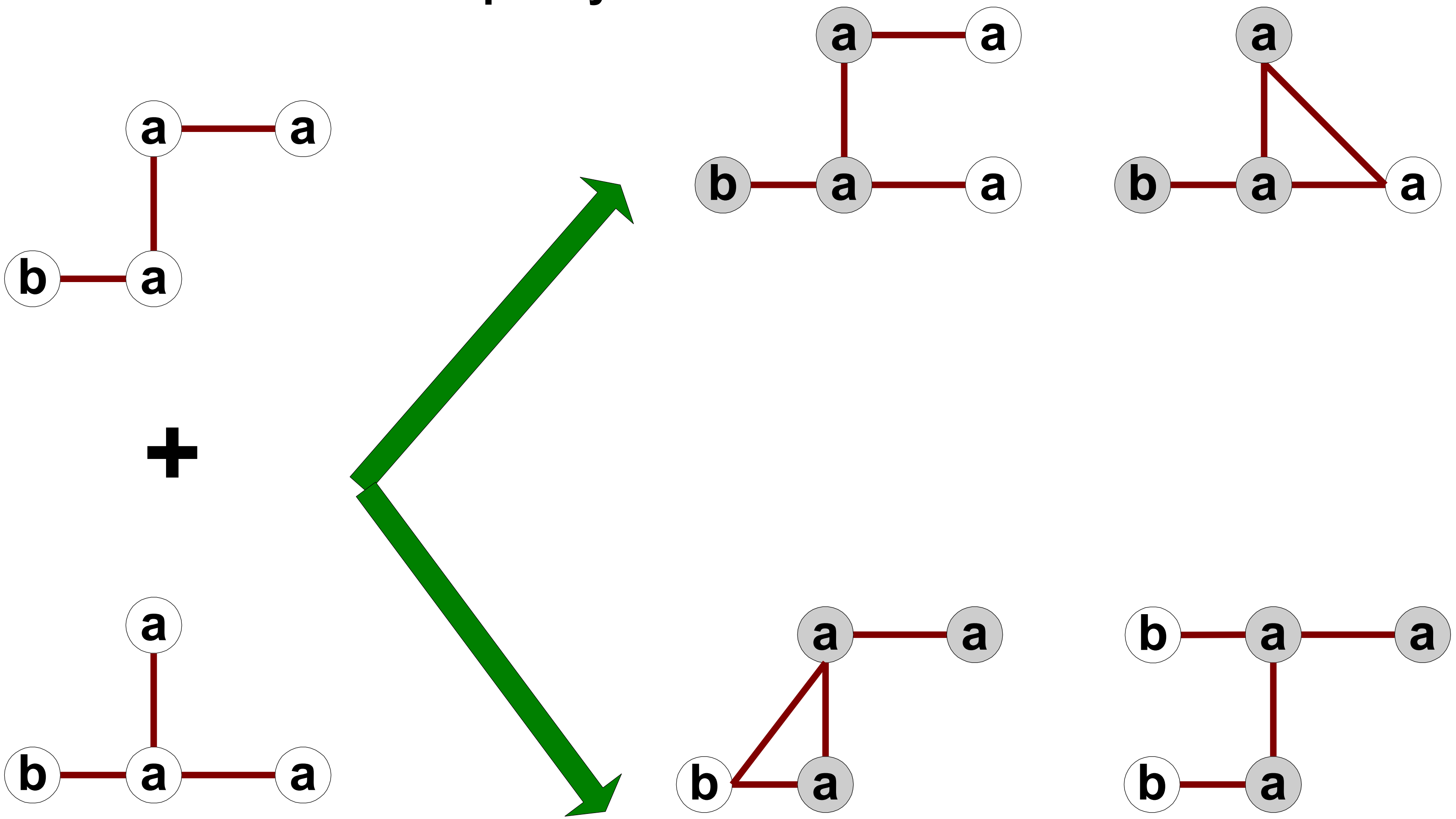




Core: The (k-1) subgraph that is common between the joint graphs

Multiplicity of Candidates (Edge growing)

- **Case 3: Core multiplicity**

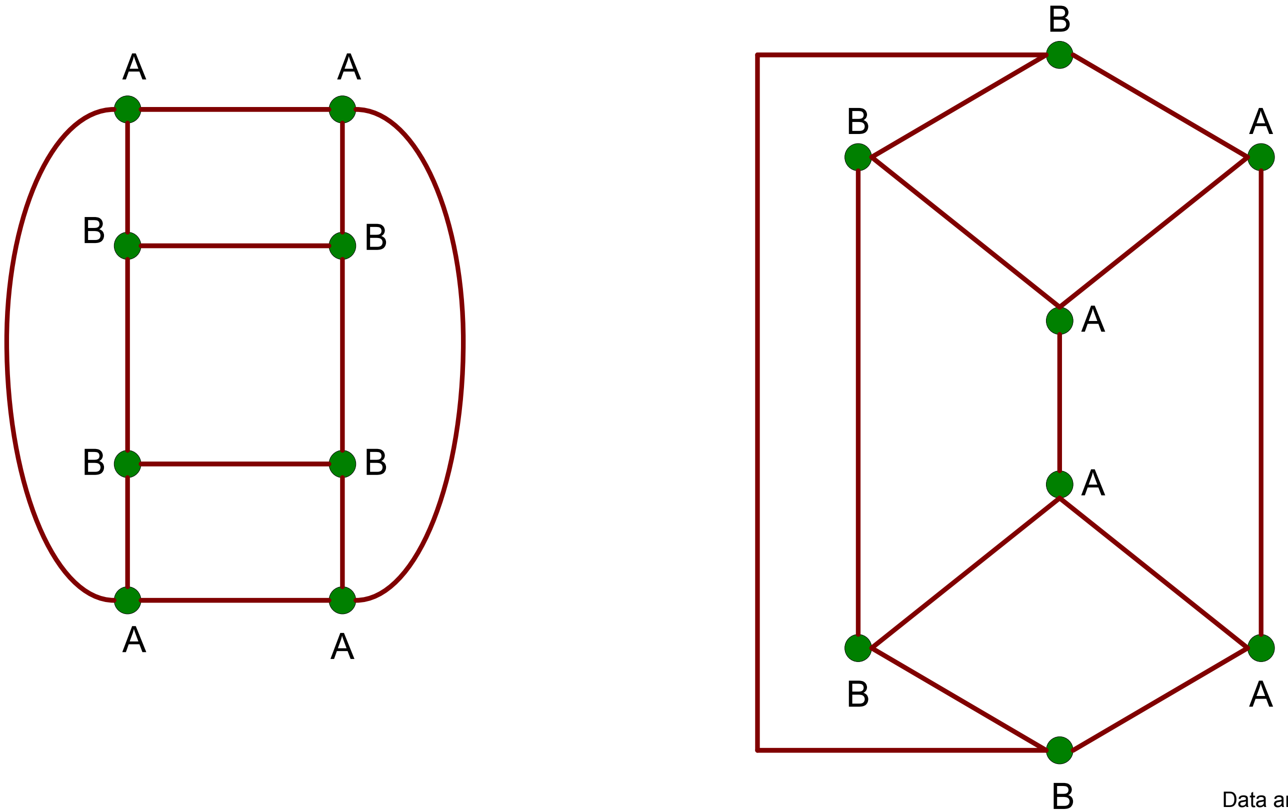


Graph Isomorphism

- **A graph G is isomorphic to a graph H, if it is topologically equivalent to H**
- **There exists a bijection between the vertex sets of the two graph**

$$f: V(G) \to V(H)$$

such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H.

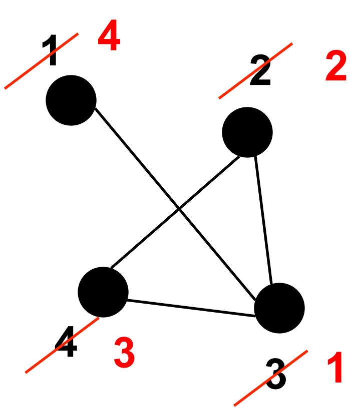


Graph Isomorphism

- **Test for graph isomorphism is needed:**
  - **During candidate generation step, to determine whether a candidate has been already 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**
    - **Find the permutations of the vertices so that the adjacency matrix is lexicographically maximized when read off from left to right, one row at a time**

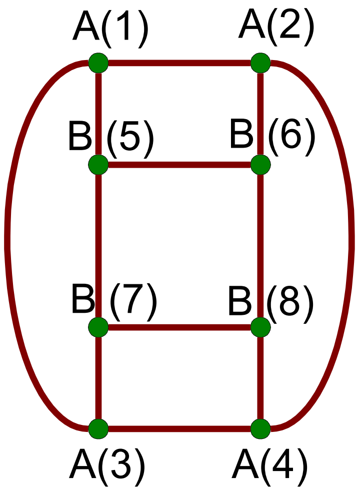


$$\longrightarrow \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

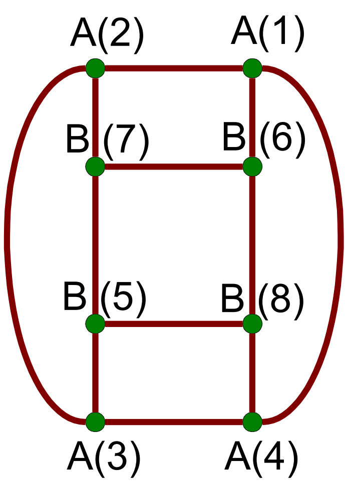
String: 0010001111010110

Canonical: 0111101011001000

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

It is sufficient to consider the string representation (canonical encoding) of the upper triangular matrix