Data Mining

UNIT- IV
Association Pattern Mining
(Advanced Concepts)

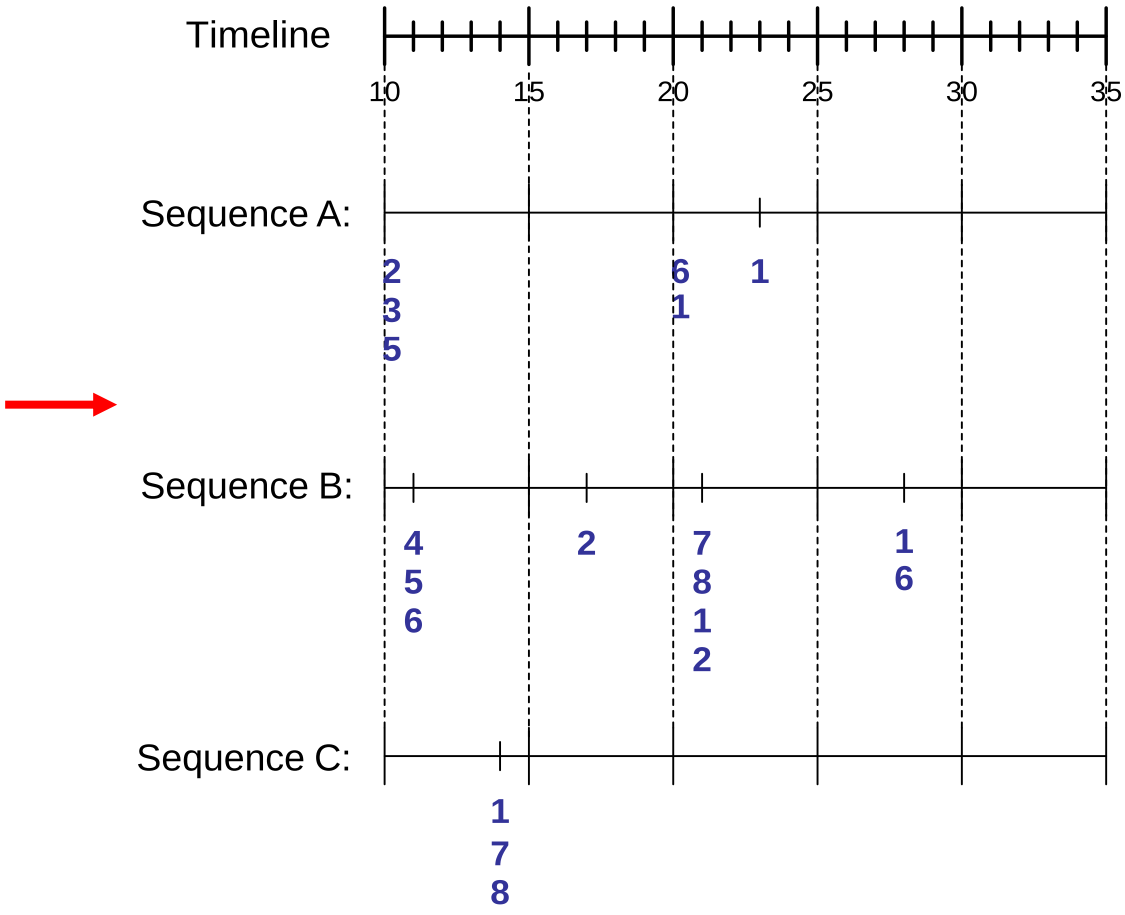
Association Analysis: Advanced Concepts

Sequential Patterns

Sequence Data

Sequence Database:

Sequence ID	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

- Sequence of different transactions by a customer at an online store:

  < {Digital Camera,iPad} {memory card} {headphone,iPad cover} >

- 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 Discovery: Examples

- In telecommunications alarm logs,
  - Inverter_Problem:

    (Excessive_Line_Current) (Rectifier_Alarm) --> (Fire_Alarm)

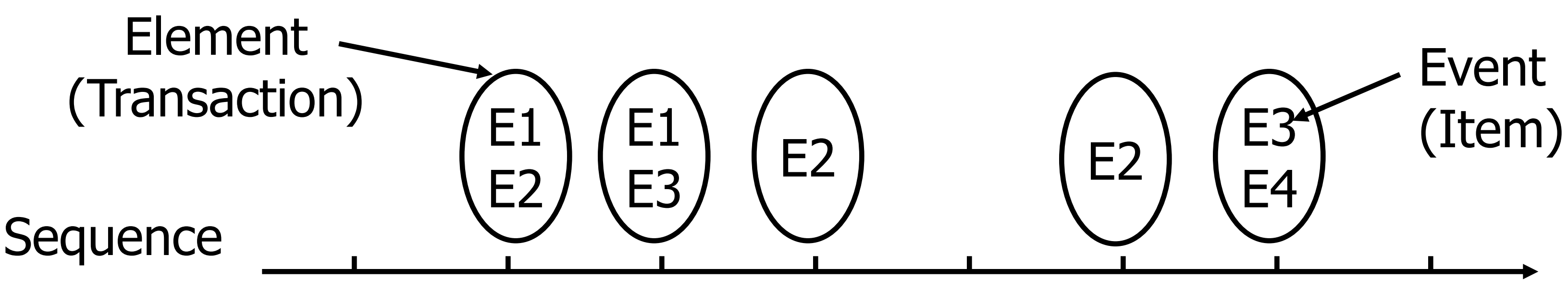
- In point-of-sale transaction sequences,
  - Computer Bookstore:

    (Intro_To_Visual_C) (C++_Primer) --> (Perl_for_dummies,Tcl_Tk)
  - Athletic Apparel Store:

    (Shoes) (Racket, Racketball) --> (Sports_Jacket)

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



Sequence Data vs. Market-basket Data

Sequence Database:

Customer	Date	Items bought
A	10	2, 3, 5
A	20	1,6
A	23	1
B	11	4, 5, 6
B	17	2
B	21	1,2,7,8
B	28	1, 6
C	14	1,7,8

Market- basket Data

Events
2, 3, 5
1,6
1
4,5,6
2
1,2,7,8
1,6
1,7,8

Sequence Data vs. Market-basket Data

Sequence Database:

Customer	Date	Items bought
A	10	2, 3, 5
A	20	1,6
A	23	1
B	11	4, 5, 6
B	17	2
B	21	1,2,7,8
B	28	1, 6
C	14	1,7,8

Market- basket Data

Events
2, 3, 5
1,6
1
4,5,6
2
1,2,7,8
1,6
1,7,8

Formal Definition of a Sequence

- A sequence is an ordered list of elements

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

  – Each element contains a collection of events (items)

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

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

- A k-sequence is a sequence that contains k events (items)
  – <{a,b} {a}> has a length of 2 and it is a 3-sequence

Formal Definition of a Subsequence

- A sequence t: $<a_1\ a_2 \ldots a_n>$ **is contained** in another sequence s: $<b_1\ b_2 \ldots b_m>$ $(m \geq n)$ if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$
- Illustrative Example:

s:	b_1	b_2	b_3	b_4	b_5
t:		a_1	a_2		a_3

t **is a subsequence** of s if $a_1 \subseteq b_2$, $a_2 \subseteq b_3$, $a_3 \subseteq b_5$.

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

Sequential Pattern Mining: Definition

- 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*)

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

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%

Sequence Data vs. Market-basket Data

Sequence Database:

Customer	Date	Items bought
A	10	2, 3, 5
A	20	1,6
A	23	1
B	11	4, 5, 6
B	17	2
B	21	1,2,7,8
B	28	1, 6
C	14	1,7,8

{2} -> {1}

$$conf(\{2\} \rightarrow \{1\}) = \frac{\sigma(\{2\}\{1\})}{\sigma(\{2\})}$$

Market- basket Data

Events
2, 3, 5
1,6
1
4,5,6
2
1,2,7,8
1,6
1,7,8

(1,8) -> (7)

$$conf(1,8) \rightarrow (7)) = \frac{\sigma(1,7,8)}{\sigma(\{1,8\})}$$

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:

  $<\{i_1, i_2\}>, <\{i_1, i_3\}>, \ldots,$

  $<\{i_1\} \{i_1\}>, <\{i_1\} \{i_2\}>, \ldots, <\{i_n\} \{i_n\}>$
- Candidate 3-subsequences:

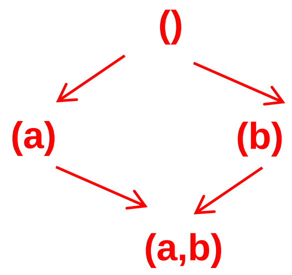
  $<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \ldots,$

  $<\{i_1, i_2\} \{i_1\}>, <\{i_1, i_2\} \{i_2\}>, \ldots,$

  $<\{i_1\} \{i_1, i_2\}>, <\{i_1\} \{i_1, i_3\}>, \ldots,$

  $<\{i_1\} \{i_1\} \{i_1\}>, <\{i_1\} \{i_1\} \{i_2\}>, \ldots$

Extracting Sequential Patterns: Simple example



Item-set patterns

- Given 2 events: a, b
- Candidate 1-subsequences:

  <{a}>, <{b}>.
- Candidate 2-subsequences:

  <{a} {a}>, <{a} {b}>, <{b} {a}>, <{b} {b}>, <{a, b}>.
- Candidate 3-subsequences:

  <{a} {a} {a}>, <{a} {a} {b}>, <{a} {b} {a}>, <{a} {b} {b}>,

  <{b} {b} {b}>, <{b} {b} {a}>, <{b} {a} {b}>, <{b} {a} {a}>

  <{a, b} {a}>, <{a, b} {b}>, <{a} {a, b}>, <{b} {a, b}>

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 the following candidate 2-sequences: <$\{i_1\}$ $\{i_1\}$>, <$\{i_1\}$ $\{i_2\}$>, <$\{i_2\}$ $\{i_2\}$>, <$\{i_2\}$ $\{i_1\}$> and <$\{i_1, i_2\}$>. (**Note**: <$\{i_1\}$> can be merged with itself to produce: <$\{i_1\}$ $\{i_1\}$>)

- 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 an event from the first element in w_1 is the same as the subsequence obtained by removing an event from the last element in w_2

Candidate Generation

- Base case (k=2):
  - Merging two frequent 1-sequences <{i_1}> and <{i_2}> will produce the following candidate 2-sequences: <{i_1} {i_1}>, <{i_1} {i_2}>, <{i_2} {i_2}>, <{i_2} {i_1}> and <{i_1 i_2}>. (**Note**: <{i_1}> can be merged with itself to produce: <{i_1} {i_1}>)

- 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 an event from the first element in w_1 is the same as the subsequence obtained by removing an event from the last element in w_2
    - The resulting candidate after merging is given by extending the sequence w_1 as follows-
      - If the last element of w_2 has only one event, append it to w_1
      - Otherwise add the event from the last element of w_2 (which is absent in the last element of w_1) to the last element of w_1

Candidate Generation Examples

- Merging w_1=<{1 2 3} {4 6}> and w_2 =<{2 3} {4 6} {5}>
  produces the candidate sequence < {1 2 3} {4 6} {5}> because the last element of w_2 has only one event
- Merging w_1=<{1} {2 3} {4}> and w_2 =<{2 3} {4 5}>
  produces the candidate sequence < {1} {2 3} {4 5}> because the last element in w_2 has more than one event
- Merging w_1=<{1 2 3} > and w_2 =<{2 3 4} >
  produces the candidate sequence < {1 2 3 4}> because the last element in w_2 has more than one event
- 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
  < {2 6} {4 5}>

Candidate Generation: Examples (ctd)

- Can <{a},{b},{c}> merge with <{b},{c},{f}> ?
- Can <{a},{b},{c}> merge with <{b,c},{f}>?
- Can <{a},{b},{c}> merge with <{b},{c,f}>?
- Can <{a,b},{c}> merge with <{b},{c,f}> ?
- Can <{a,b,c}> merge with <{b,c,f}>?
- Can <{a}> merge with <{a}>?

Candidate Generation: Examples (ctd)

- <{a},{b},{c}> can be merged with <{b},{c},{f}> to produce <{a},{b},{c},{f}>
- <{a},{b},{c}> cannot be merged with <{b,c},{f}>
- <{a},{b},{c}> can be merged with <{b},{c,f}> to produce <{a},{b},{c,f}>
- <{a,b},{c}> can be merged with <{b},{c,f}> to produce <{a,b},{c,f}>
- <{a,b,c}> can be merged with <{b,c,f}> to produce <{a,b,c,f}>
- <{a}{b}{a}> can be merged with <{b}{a}{b}> to produce <{a},{b},{a},{b}>
- <{b}{a}{b}> can be merged with <{a}{b}{a}> to produce <{b},{a},{b},{a}>

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} >

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} >

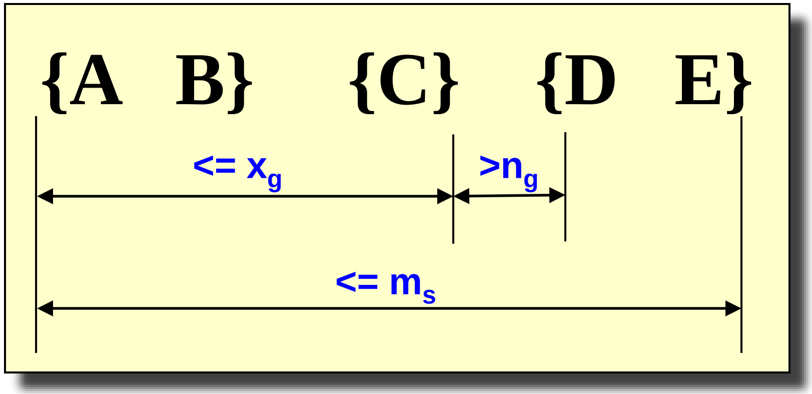
< {2} {3} {4} {5} >

< {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, d	Sequential Pattern, s	d contains s?
< {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

Contiguous Subsequences

- s is a contiguous subsequence of

  $w = \langle e_1 \rangle \langle e_2 \rangle \ldots \langle e_k \rangle$

  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 = < {1} {2} >
  - is a contiguous subsequence of

    < {1} {2 3}>, < {1 2} {2} {3}>, and < {3 4} {1 2} {2 3} {4} >
  - is not a contiguous subsequence of

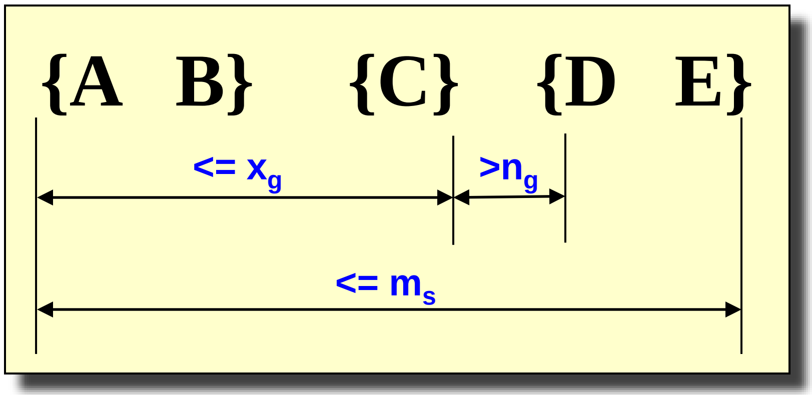
    < {1} {3} {2}> and < {2} {1} {3} {2}>

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 (I)



x_g: max-gap

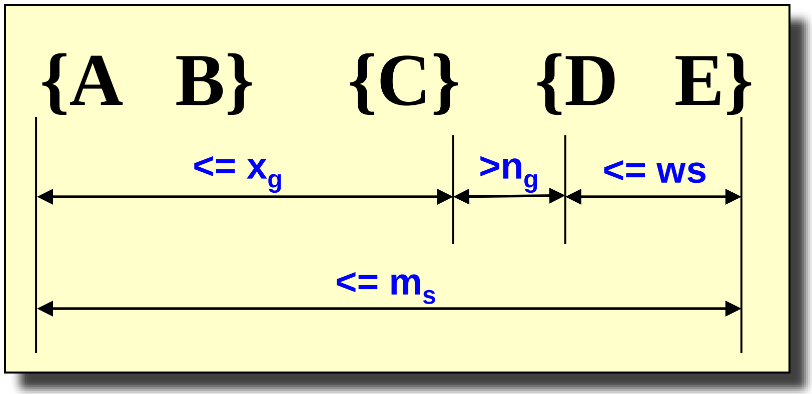
n_g: min-gap

m_s: maximum span

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

Data sequence, d	Sequential Pattern, s	d contains s?
< {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

Timing Constraints (II)



x_g: max-gap

n_g: min-gap

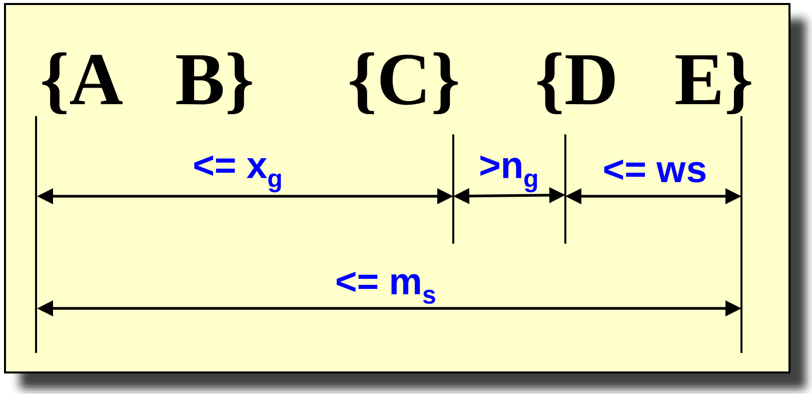
ws: window size

m_s: maximum span

$x_g = 3$, $n_g = 0$, ws = 2, $m_s = 10$

Data sequence, d	Sequential Pattern, s	d contains s?
< {1,3} {3,4} {4} {5} {6,7} {8}>	< {3,4} {5}>	Yes
< {1,3} {3,4} {4} {5} {6,7} {8}>	< {4,6} {8} >	Yes
< {1,3} {3,4} {4} {5} {6,7} {8}>	< {3,4,6} {8} >	No

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= 4

Data sequence, d	Sequential Pattern, s	d contains s?
< {2,4} {3,5,6} {4,7} {4,5} {8} >	< {3,4,5}>	No
< {1} {2} {3} {4} {5}>	< {1,2} {3,4} >	No
< {1,2} {2,3} {3,4} {4,5} {6} {7,8}>	< {1,2} {3,4} {6} {8} >	No

Association Analysis: Advanced Concepts

Subgraph Mining

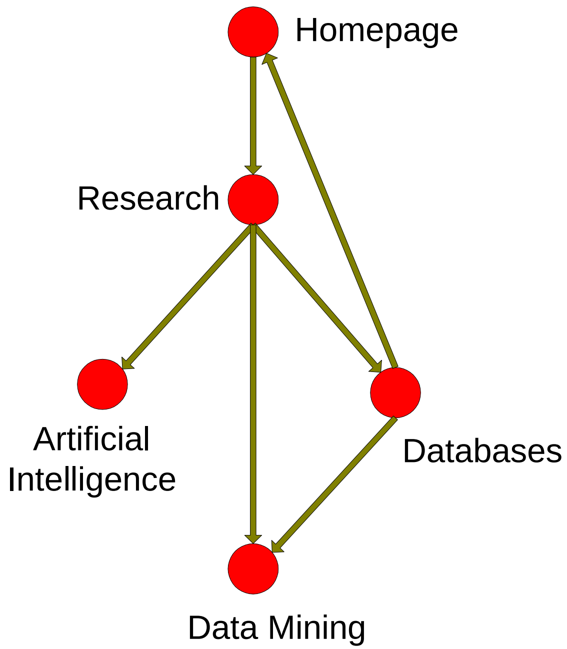
Subgraph Pattern

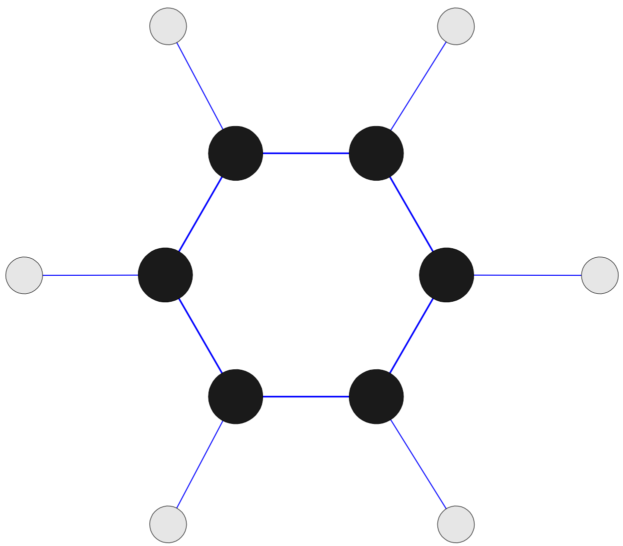
- Graph representation of entities in various application domains

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

Frequent Subgraph Mining

- Extends association analysis to finding frequent subgraphs
- Useful for Web Mining, computational chemistry, bioinformatics, spatial data sets, etc

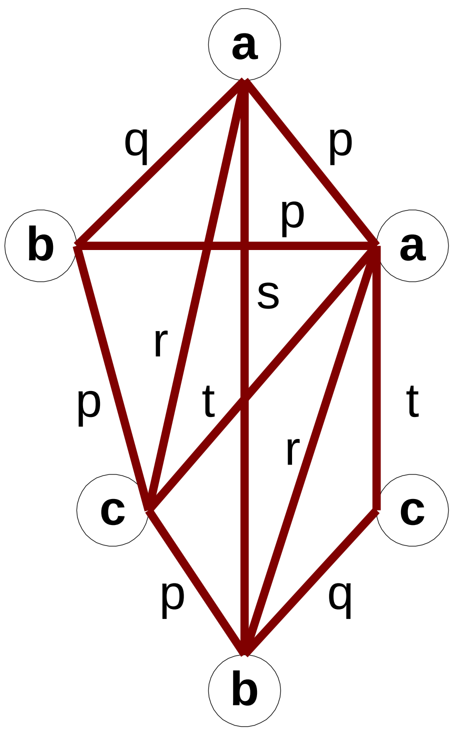




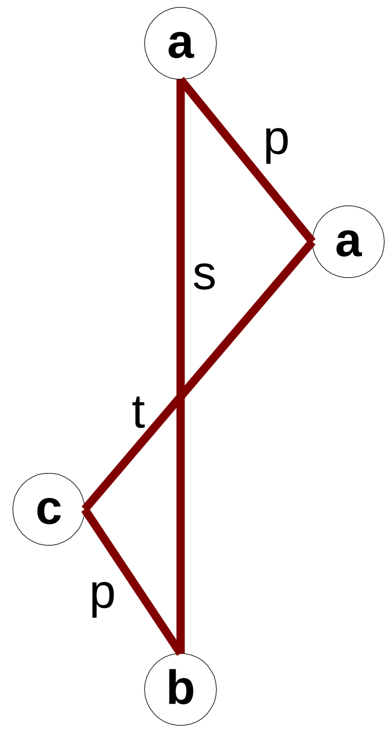
Graph Definitions

- A graph is a composition of a set of vertices V and a set of edges E. Formally defined as
  - $\mathrm{G} = \{V, E\}$
  - $v_i \in V$,
  - $(v_i, v_j) \in E$
- A graph $\mathrm{G}' = \{V', E'\}$ is a subgraph of another graph $\mathrm{G} = \{V, E\}$ if
  - $V' \subseteq V$ and $E' \subseteq E$

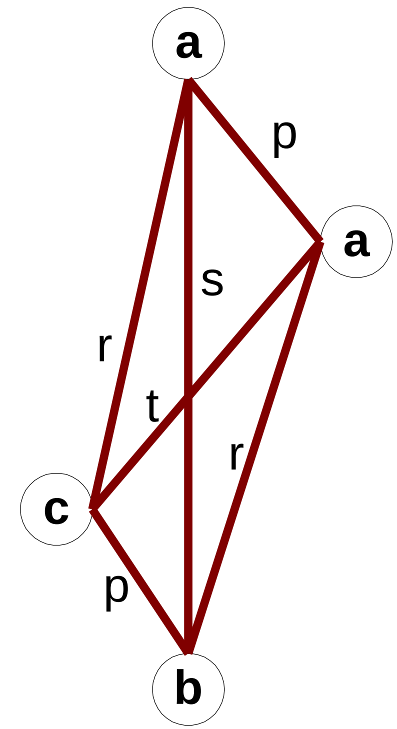
Graph Definitions



(a) Labeled Graph



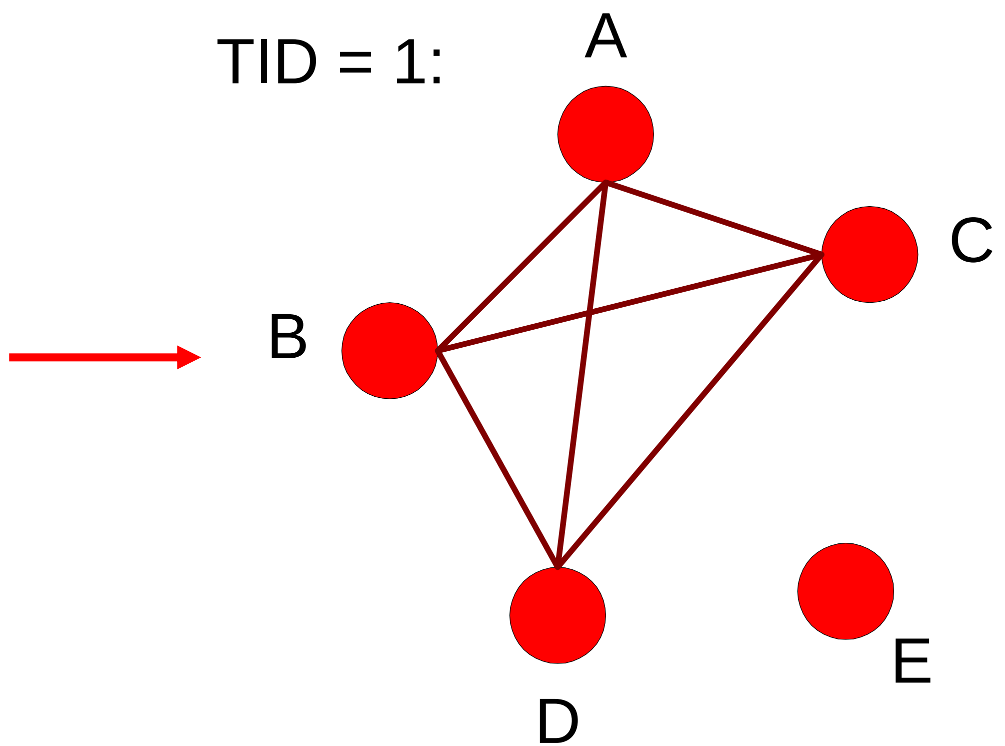
(b) Subgraph



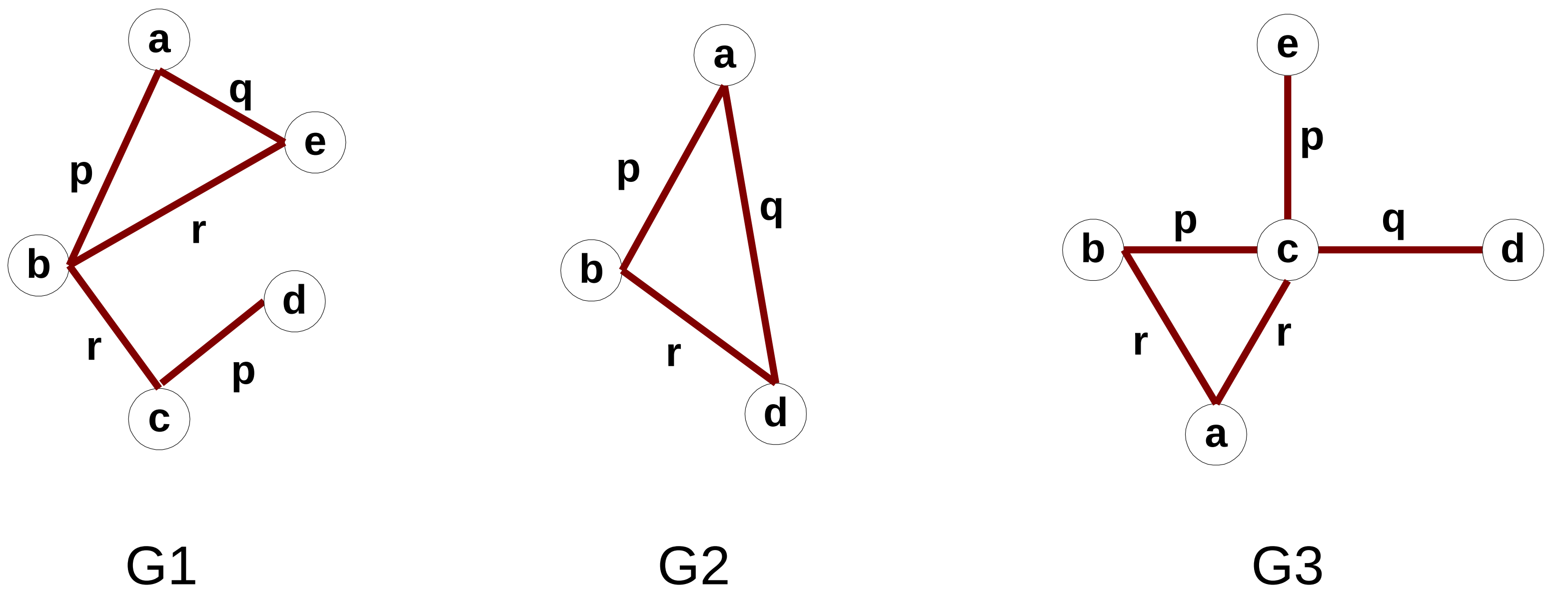
(c) Induced Subgraph

Representing Transactions as Graphs

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



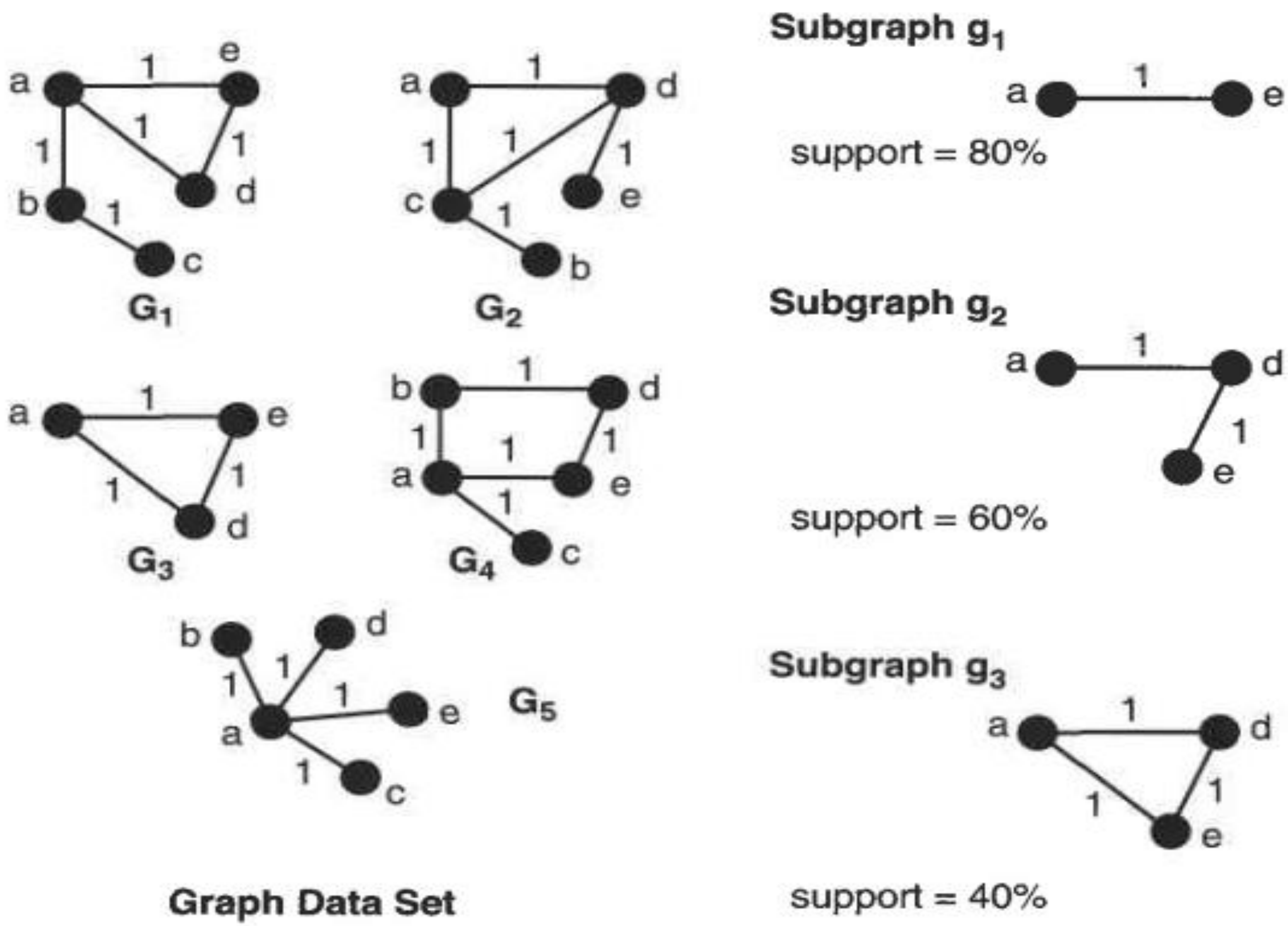
	(a,b,p)	(a,b,q)	(a,b,r)	(b,c,p)	(b,c,q)	(b,c,r)	…	(d,e,r)
G1	1	0	0	0	0	1	…	0
G2	1	0	0	0	0	0	…	0
G3	0	0	1	1	0	0	…	0
G3	…	…	…	…	…	…	…	…

Support

- A support of a subgraph g for a given collections of graph **G** is defined by

  – $s(g) = \frac{|\{G_i | g \subseteq G_i, G_i \in \boldsymbol{G}\}|}{|\boldsymbol{G}|}$

Support



a
e
b
d
c
1
G1
G2
G3
G4
G5
Graph Data Set
Subgraph g1
support = 80%
Subgraph g2
support = 60%
Subgraph g3
support = 40%

Frequent Subgraph Mining

- Tha goal of subgraph mining is to find the subgraphs g such that $s(g) \geq minsup$.

- While this formulation is applicable to any type of graph.

- Here, focus on **undirected**, **connected** graphs.

Frequent Subgraph Mining

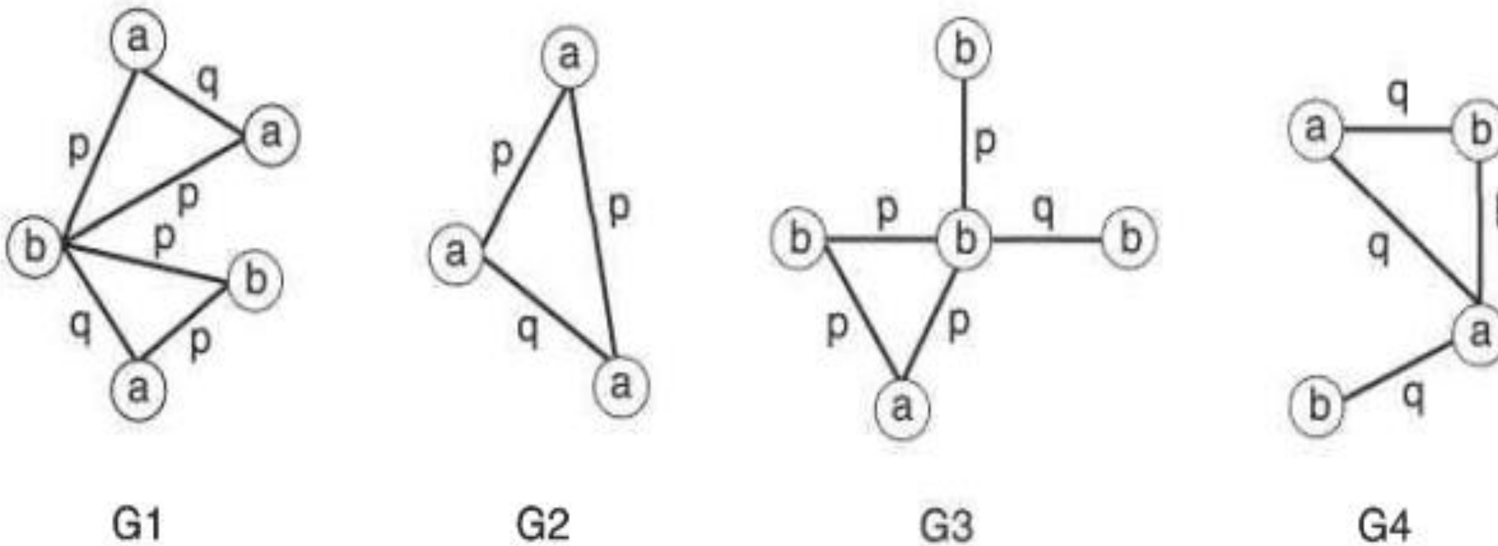
- Mining frequent subgraphs is a computationally expensive task because of the exponential scale of the search space.
- For example data set contains d entities.

- $\text{total subgraphs} = \sum_{i=1}^{d} \binom{d}{i} \times 2^{i(i-1)/2}$

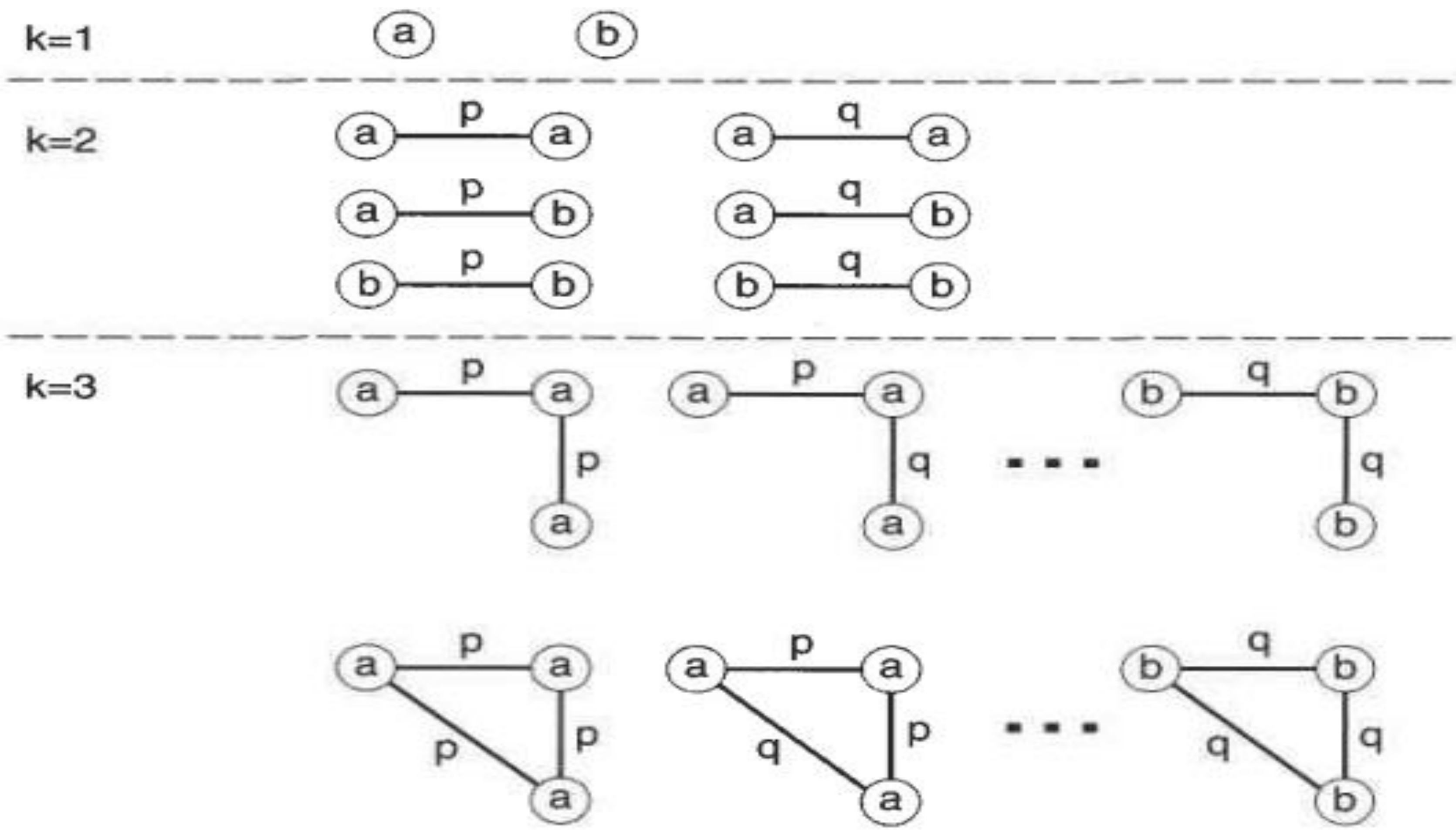
Number of entities, d	1	2	3	4	5	6	7
Number of itemsets	2	4	8	16	32	64	128
Number of subgraphs	2	5	18	113	1,450	40,069	2,350,602

Frequent Subgraph Mining

- The number of candidate ***subgraphs*** is considerably larger than the number of candidates ***itemsets***.
  - An item can appear at most once in an itemset, whereas a vertex label can appear more than once in a graph.
  - The same pair of vertex labels can have multiple choices of edge labels.



(a) Example of a graph data set.



(b) List of connected subgraphs.

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?

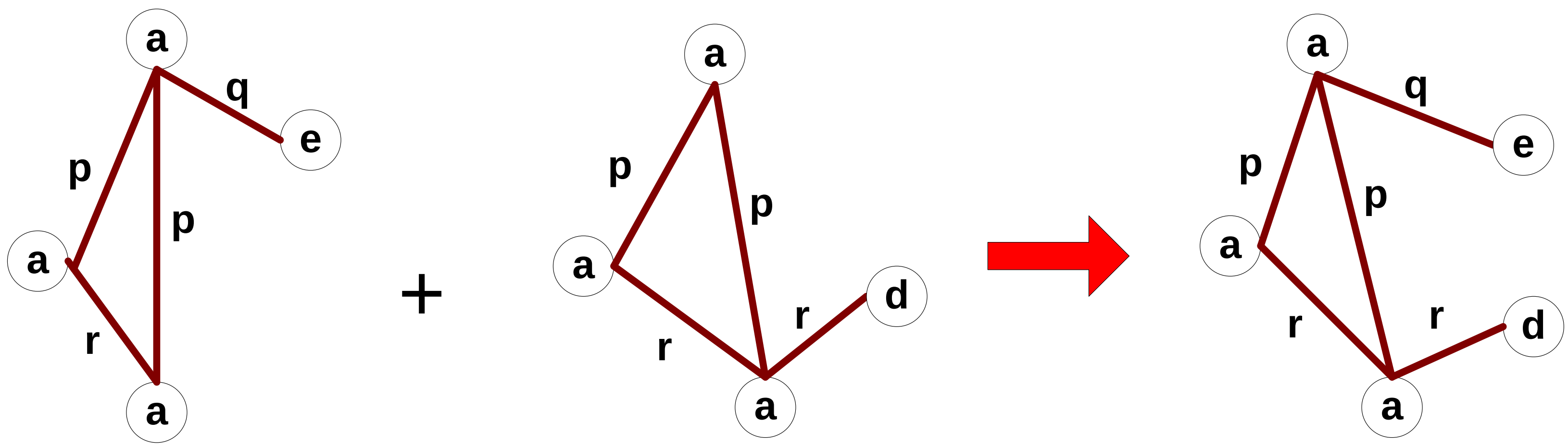
Challenges…

- 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

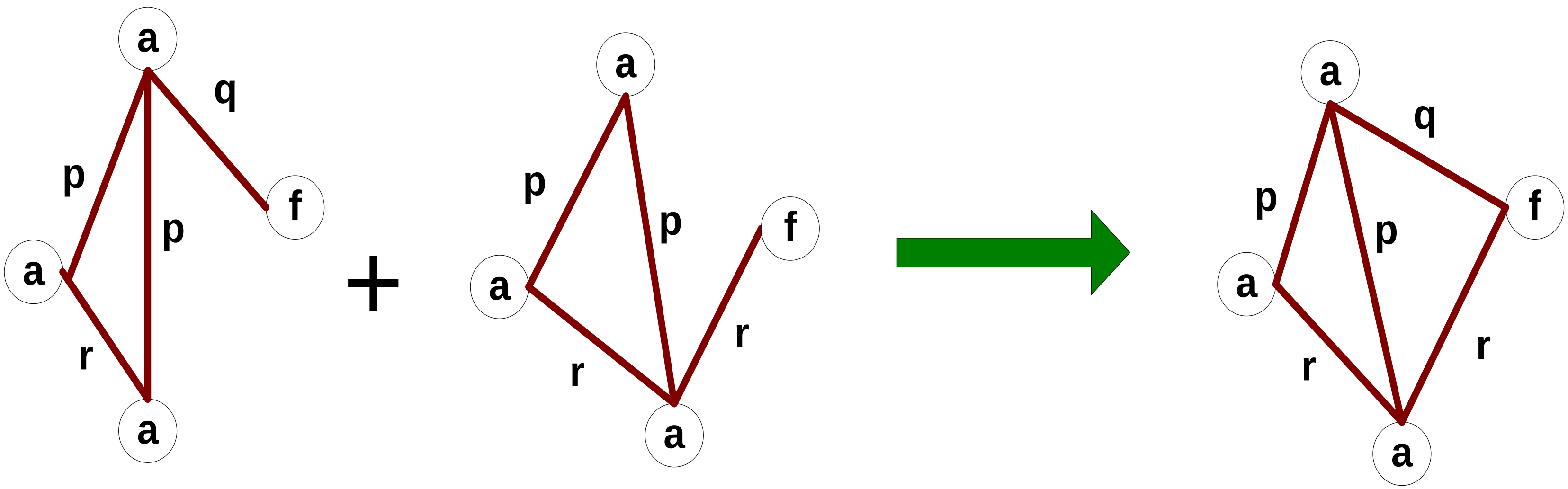
Vertex Growing



G1 G2 G3 = join(G1,G2)

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

Edge Growing



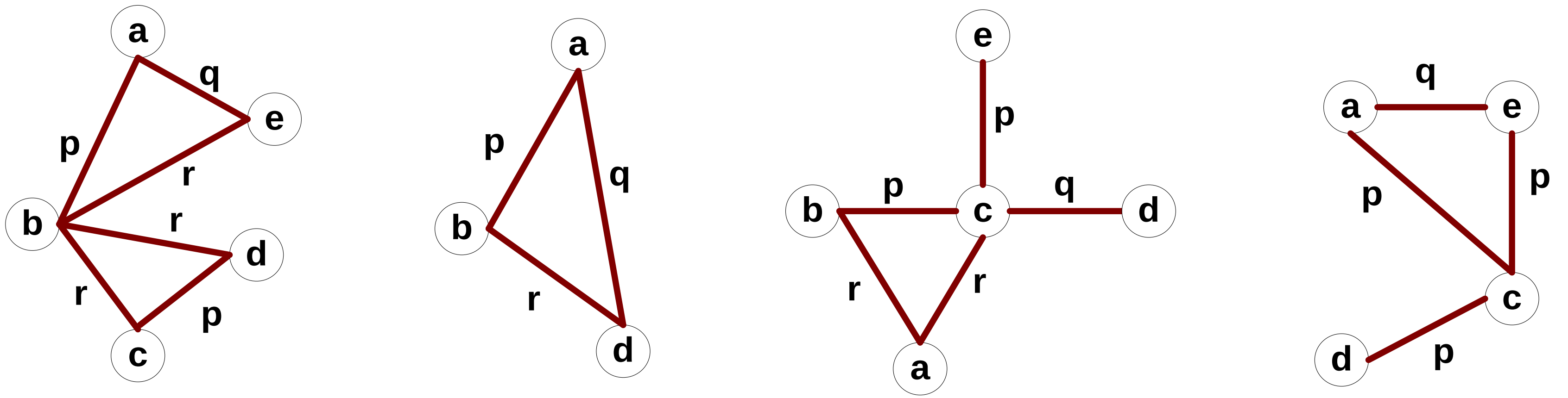
G1 G2 G3 = join(G1,G2)

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



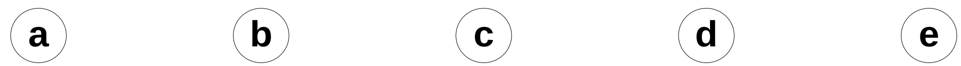
G1 G2 G3 G4

	(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

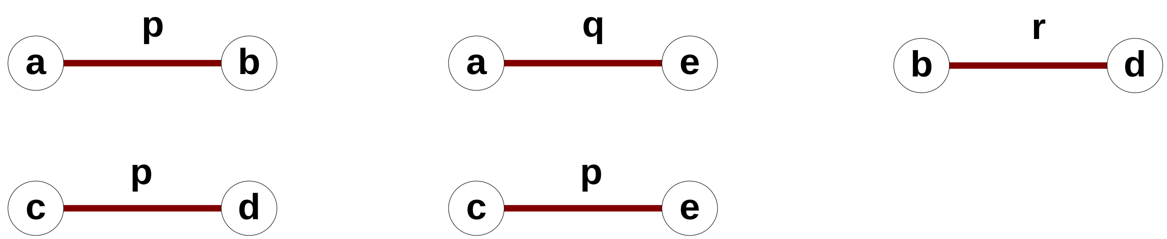
Example

Minimum support count = 2

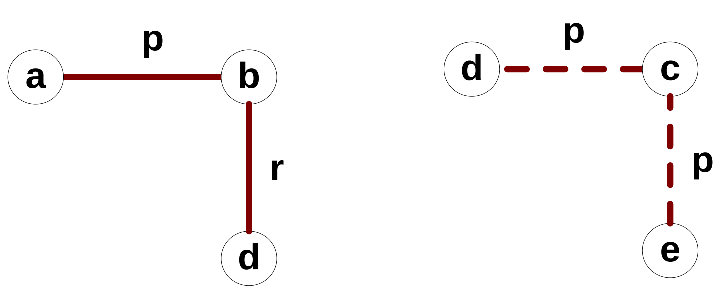
k=1
Frequent
Subgraphs



k=2
Frequent
Subgraphs



k=3
Candidate
Subgraphs



(Pruned candidate
due to low support)

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

Questions