Computer Science 477/577

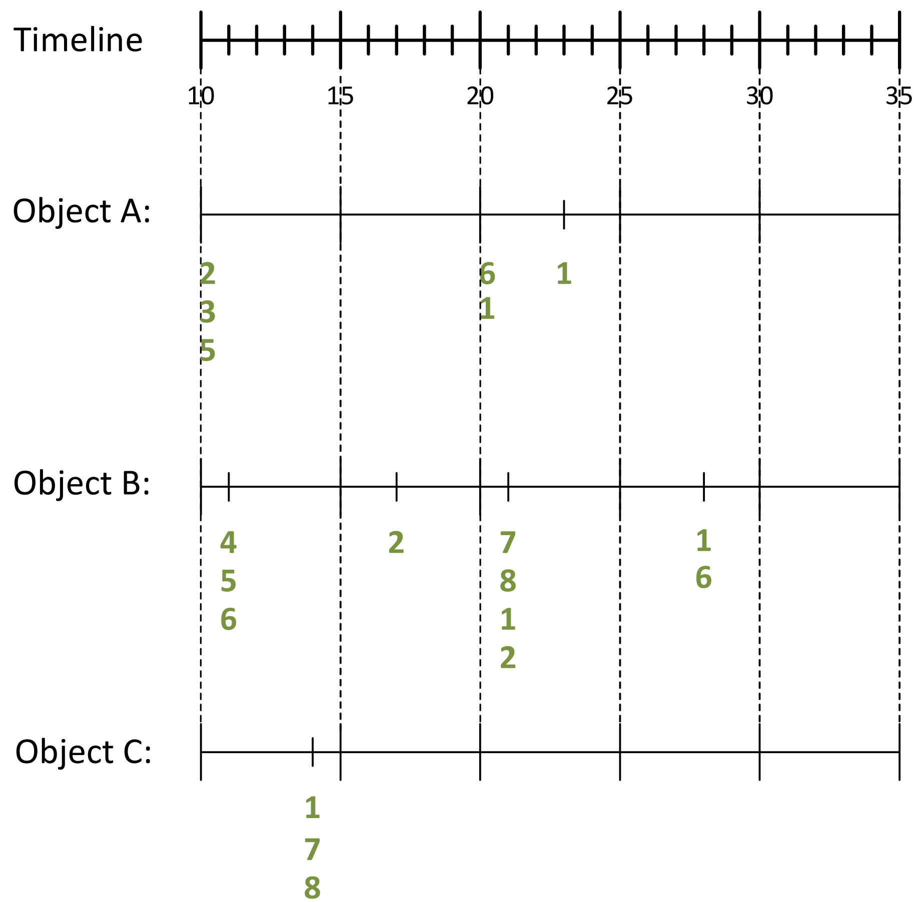
Sequence Mining

Lecture 15

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



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)

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

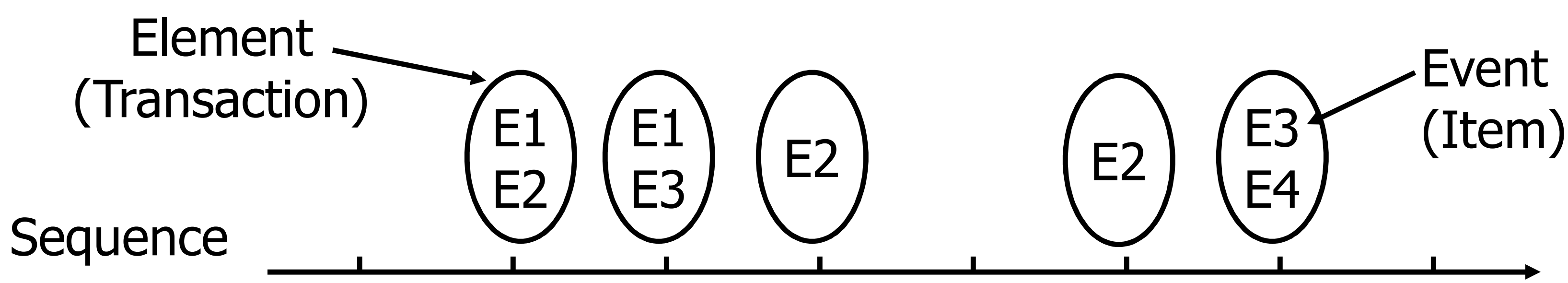
- Each element is attributed to a specific time or location

- 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 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, dairy 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 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:
  <{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 $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	
< {1,2} {3,4} >	< {1} {2} >	
< {2,4} {2,4} {2,5} >	< {2} {4} >	

- 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:

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

- Candidate 2-subsequences:

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

- Candidate 3-subsequences:

  $\langle\{i_1, i_2, i_3\}\rangle, \langle\{i_1, i_2, i_4\}\rangle, \ldots, \langle\{i_1, i_2\}\ \{i_1\}\rangle, \langle\{i_1, i_2\}\ \{i_2\}\rangle, \ldots,$
  $\langle\{i_1\}\ \{i_1, i_2\}\rangle, \langle\{i_1\}\ \{i_1, i_3\}\rangle, \ldots, \langle\{i_1\}\ \{i_1\}\ \{i_1\}\rangle, \langle\{i_1\}\ \{i_1\}\ \{i_2\}\rangle,$
  $\ldots$

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
  - **Initial Pruning**:
    - Prune if it is not the case that all of the k-1 subsequences of a k sequence are frequent
  - **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}>

Sensor	Timestamp	Events
S1	1	A, B
	2	C
	3	D, E
	4	C
S2	1	A, B
	2	C, D
	3	E
S3	1	B
	2	A
	3	B
	4	D, E
S4	1	C
	2	D, E
	3	C
	4	E
S5	1	B
	2	A
	3	B, C
	4	A, D

- S1: <{A,B}><{C}><{D,E}><{C}>
- S2: <{A,B}><{C,D}><{E}>
- S3: <{B}><{A}><{B}><{D,E}>
- S4: <{C}><{D,E}><{C}><{E}>
- S5: <{B}><{A}><{B,C}><{A,D}>

$< \{A\} >, < \{B\} >, < \{C\} >, < \{D\} >, < \{E\} >$
$< \{A\} \{C\} >, < \{A\} \{D\} >, < \{A\} \{E\} >, < \{B\} \{C\} >,$
$< \{B\} \{D\} >, < \{B\} \{E\} >, < \{C\} \{D\} >, < \{C\} \{E\} >, < \{D, E\} >$

Sensor	Timestamp	Events
S1	1	A, B
	2	C
	3	D, E
	4	C
S2	1	A, B
	2	C, D
	3	E
S3	1	B
	2	A
	3	B
	4	D, E
S4	1	C
	2	D, E
	3	C
	4	E
S5	1	B
	2	A
	3	B, C
	4	A, D

- 1-sequences?
  - <{A}> : 4/5 ≥ 50%

< {A} >, < {B} >, < {C} >, < {D} >, < {E} >
< {A} {C} >, < {A} {D} >, < {A} {E} >, < {B} {C} >,
< {B} {D} >, < {B} {E} >, < {C} {D} >, < {C} {E} >, < {D, E} >

Sensor	Timestamp	Events
S1	1	A, B
	2	C
	3	D, E
	4	C
S2	1	A, B
	2	C, D
	3	E
S3	1	B
	2	A
	3	B
	4	D, E
S4	1	C
	2	D, E
	3	C
	4	E
S5	1	B
	2	A
	3	B, C
	4	A, D

- 1-sequences?
  - <{A}> : 4/5 ≥ 50%
  - <{B}> : 4/5 ≥ 50%
  - <{E}> : 4/5 ≥ 50%

< {A} >, < {B} >, < {C} >, < {D} >, < {E} >
< {A} {C} >, < {A} {D} >, < {A} {E} >, < {B} {C} >,
< {B} {D} >, < {B} {E} >, < {C} {D} >, < {C} {E} >, < {D, E} >

Sensor	Timestamp	Events
S1	1	A, B
	2	C
	3	D, E
	4	C
S2	1	A, B
	2	C, D
	3	E
S3	1	B
	2	A
	3	B
	4	D, E
S4	1	C
	2	D, E
	3	C
	4	E
S5	1	B
	2	A
	3	B, C
	4	A, D

- 1-sequences?
  - <{A}> : 4/5 ≥ 50%
  - <{B}> : 4/5 ≥ 50%
  - <{E}> : 4/5 ≥ 50%
- 2-sequences?
  - <{A, B}> : 2/5 < 50%

< {A} >, < {B} >, < {C} >, < {D} >, < {E} >
< {A} {C} >, < {A} {D} >, < {A} {E} >, < {B} {C} >,
< {B} {D} >, < {B} {E} >, < {C} {D} >, < {C} {E} >, < {D, E} >

Sensor	Timestamp	Events
S1	1	A, B
	2	C
	3	D, E
	4	C
S2	1	A, B
	2	C, D
	3	E
S3	1	B
	2	A
	3	B
	4	D, E
S4	1	C
	2	D, E
	3	C
	4	E
S5	1	B
	2	A
	3	B, C
	4	A, D

- 1-sequences?
  - <{A}> : 4/5 ≥ 50%
  - <{B}> : 4/5 ≥ 50%
  - <{E}> : 4/5 ≥ 50%
- 2-sequences?
  - <{A, B}> : 2/5 < 50%
  - <{A,C}> : 0 < 50%
  - <{D, E}> : 3/5 ≥ 50%

Sequence Merging Procedure

- A sequence s(1) is merged with another sequence s(2) only if the subsequence obtained by dropping the first event in s(1) is identical to the subsequence obtained by dropping the last event in s(2).
- The resulting candidate is the sequence s(1), concatenated with the last event from s(2).
- The last event from s(2) can either be merged into the same element as the last event in s(1) or
- Different elements depending on the following conditions:
  - If the last two events in s(2) belong to the same element, then the last event in s(2) is part of the last element in s(1) in the merged sequence.
  - If the last two events in s(2) belong to different elements, then the last event in s(2) becomes a separate element appended to the end of s(1) in the merged sequence.

Frequent
3-sequences

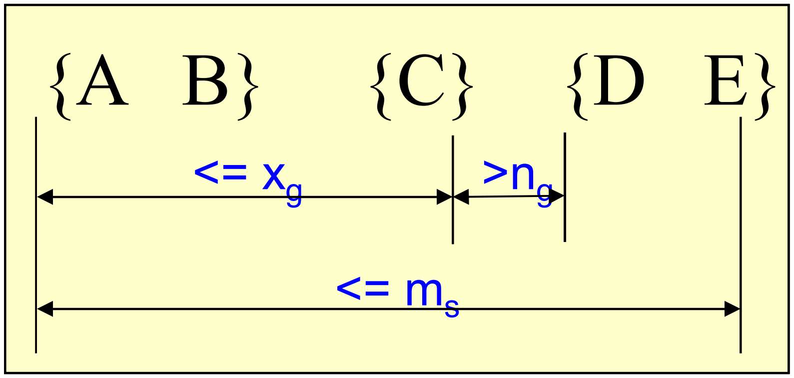
< {1} {2} {3} >
< {1} {2 5} >
< {1} {5} {3} >
< {2} {3} {4} >
< {2 5} {3} >
< {3} {4} {5} >
< {5} {3 4} >

Candidate
Generation

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

- <{1}{2}{3}{ 4}> is obtained by merging <{1}{2}{3}> with <{2}{3}{4}>.
- Merging <{1}{5}{3}> with <{5}{3,4}> →<{1}{5}{3,4}>
- <{1}{2, 5}{3}> is generated by merging a different pair of sequences, <{1}{2,5}> and <{2,5}{3}>.
- <{2,5}{3}> merges with <{2,5}{3,4}> → <{2,5}{3,4}>

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?
< {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 Generalized *Sequential Pattern* algorithm 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%

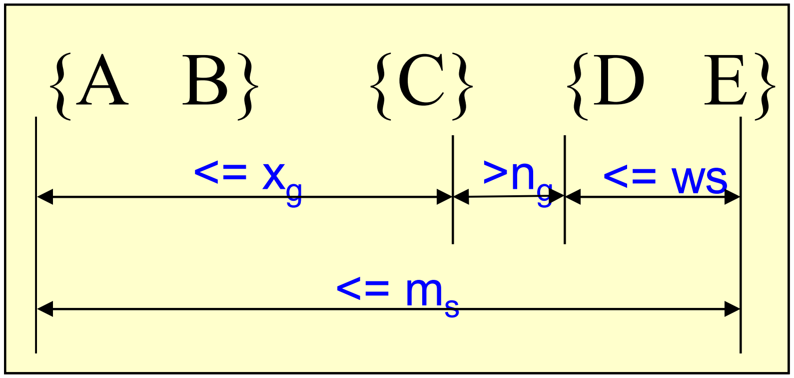
but

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

Problem exists because of max-gap constraint

No such problem if max-gap is infinite

Timing Constraints (II)



x_g: max-gap

n_g: min-gap

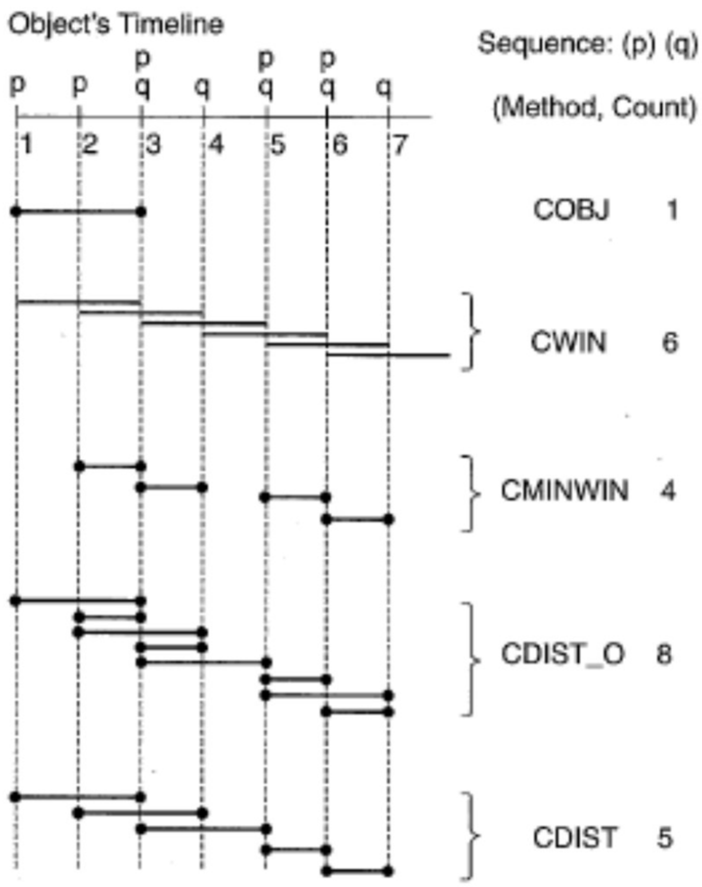
ws: window size

m_s: maximum span

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

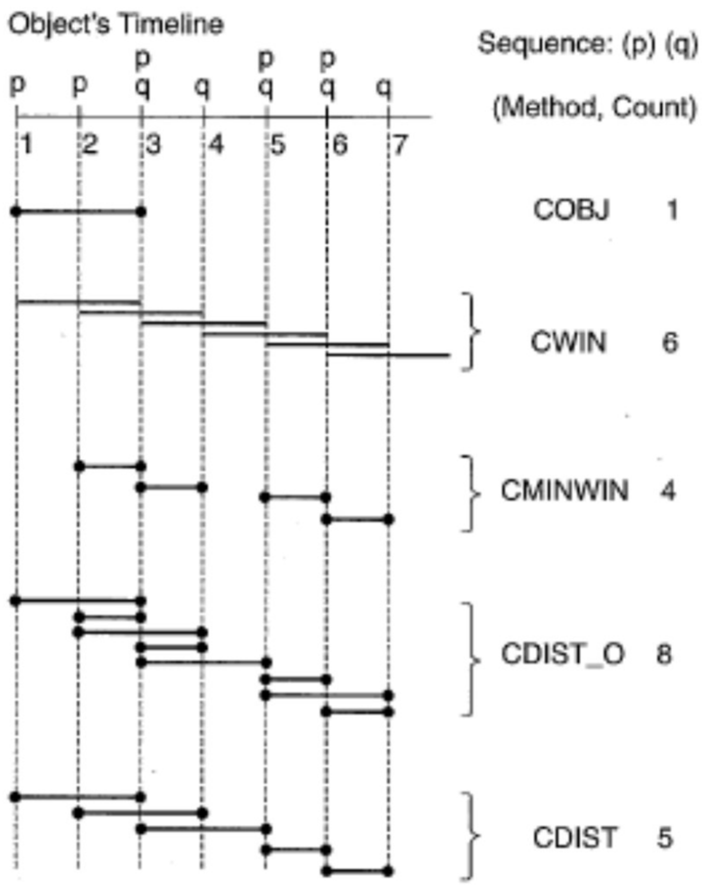
Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {4,7} {4,6} {8} >	< {3} {5} >	No
< {1} {2} {3} {4} {5}>	< {1,2} {3} >	Yes
< {1,2} {2,3} {3,4} {4,5}>	< {1,2} {3,4} >	Yes

Counting Methods



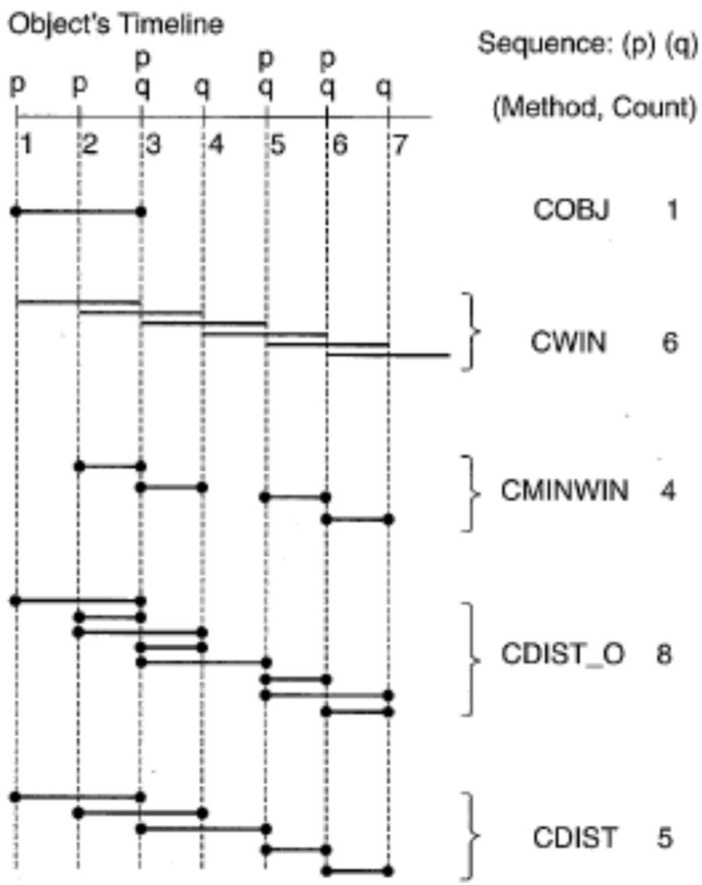
- COBJ: One occurrence per object.
- This method looks for at least one occurrence of a given sequence in an object's timeline.
- Even though the sequence <{p}{q}> appears several times in the object's timeline, it is counted only once with *p* occurring at *t* = 1 and *q* occurring at *t* = 3.

Counting Methods



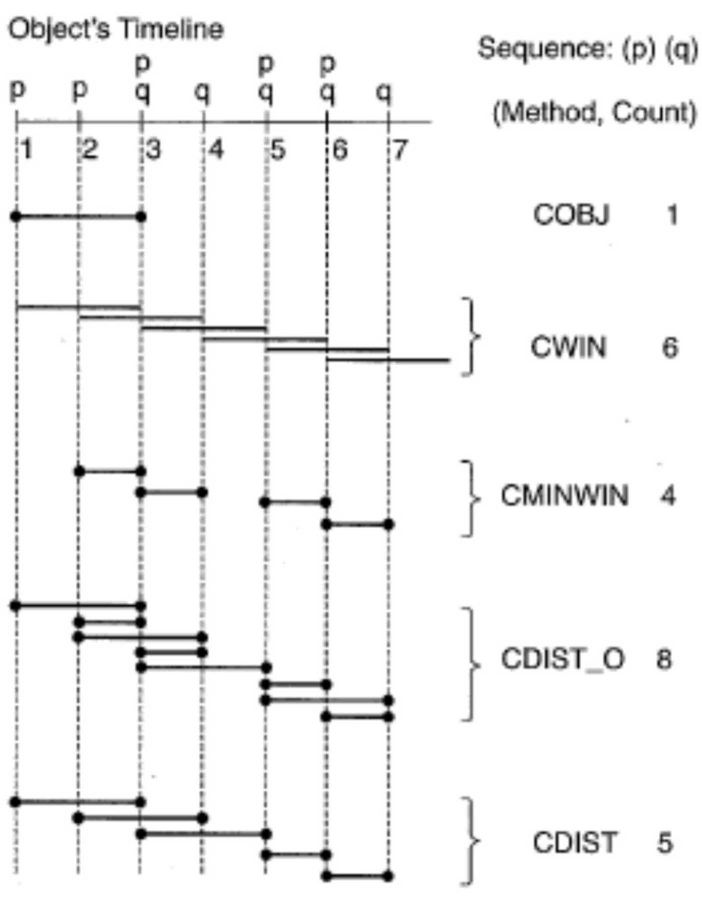
- CWIN: One occurrence per sliding window.
- In this approach, a sliding time window of fixed length *(maxspan)* is moved across an object's timeline, one unit at a time.
- The support count is incremented each time the sequence is encountered in the sliding window.
- The sequence ({p}{q}) is observed six times using this method.

Counting Methods



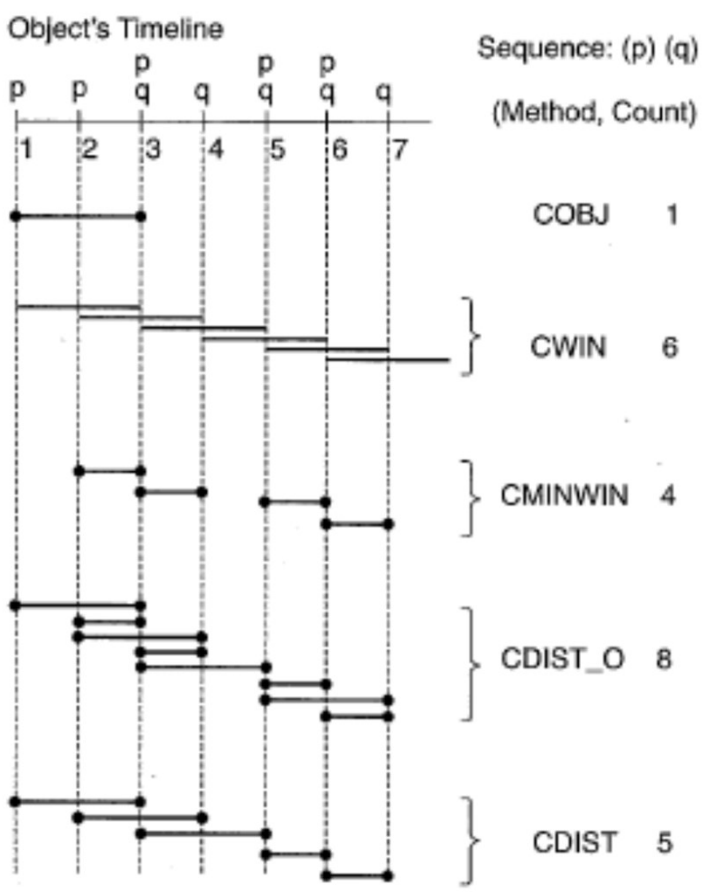
- CMINWIN: Number of minimal windows of occurrence.
- A minimal window of occurrence is the smallest window in which the sequence occurs given the timing constraints.
- In other words, a minimal window is the time interval such that the sequence occurs in that time interval, but it does not occur in any of the proper subintervals of it.
- A restrictive version of CWIN, because its effect is to shrink and collapse some of the windows that are counted by CWIN.
- Sequence <{p}{q}> has four minimal window occurrences:
  - (1) the pair *(p: t* = 2, *q: t* = 3),
  - (2) the pair *(p: t* = 3, *q:t* = 4),
  - (3) the pair *(p: t* = 5, *q: t* = 6), and
  - (4) the pair *(p: t* = 6, *q: t* = 7).
  - The occurrence of event pat *t* = 1 and event *q* at *t* = 3 is not a minimal window occurrence because it contains a smaller window with *(p: t* = 2, *q: t* = 3), which is indeed a minimal window of occurrence.

Counting Methods



- **CDIST 0:** Distinct occurrences with possibility of event-timestamp overlap.
- A distinct occurrence of a sequence is defined to be the set of event timestamp pairs such that there has to be at least one new event timestamp pair that is different from a previously counted occurrence.
- Counting all such distinct occurrences results in the CDIST 0 method.
- If the occurrence time of events *p* and *q* is denoted as a tuple (*t(p),* *t*(q)), then this method yields eight distinct occurrences of sequence (({p }{ q}) at times (1,3), (2,3), (2,4), (3,4), (3,5), (5,6), (5,7), and (6,7).

Counting Methods



- **CDIST:** Distinct occurrences with no event-timestamp overlap allowed.
- In CDIST 0 above, two occurrences of a sequence were allowed to have overlapping event-timestamp pairs, e.g., the overlap between (1,3) and (2,3). In the CDIST method, no overlap is allowed.
- Effectively, when an event-timestamp pair is considered for counting, it is marked as used and
- is never used again for subsequent counting of the same sequence.
- Example: there are five distinct, non-overlapping occurrences of the
- sequence ({p} { q}) in the diagram These occurrences happen at times (1,3), (2,4), (3,5), (5,6), and (6,7).
- Observe that these occurrences are subsets of the occurrences observed in CDIST 0.

Counting Methods - Summary

- **COBJ**: One occurrence per object
- **CWIN**: One occurrence per sliding window
- **CMINWIN**: Number of minimal windows of occurrence
- **CDIST 0:** Distinct occurrences with possibility of event-timestamp overlap
- **CDIST:** Distinct occurrences with no event-timestamp overlap allowed

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 more than 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 (III)

- The window size constraint restricts the time difference between the latest and the earliest event in any element of a sequence.
- In the above subsequences the first violates the mingap constraint since element gap is 0.
- In the second, ws is 1 time step for {1,2} and the element gap is 1 which is OK.
- For the third the ws is 0 and the element gap is 2 which is OK

Modified Support Counting Step

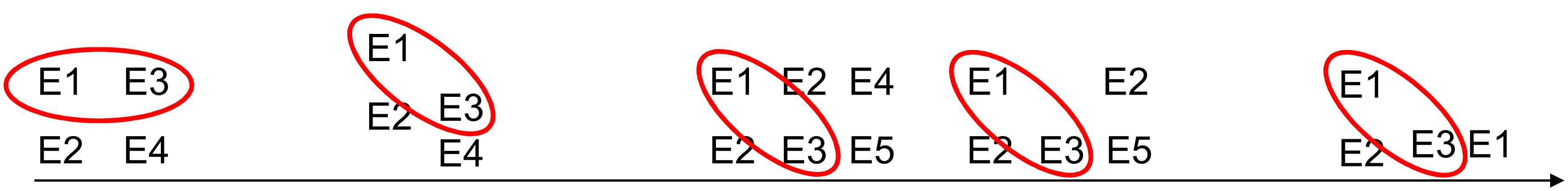
- Given a candidate pattern: <{a, c}>
  - Any data sequences that contain

    <… {a c} … >,
    <… {a} … {c}…> (where time({c}) – time({a}) ≤ ws)
    <…{c} … {a} …> (where time({a}) – time({c}) ≤ ws)

    will contribute to the support count of candidate pattern

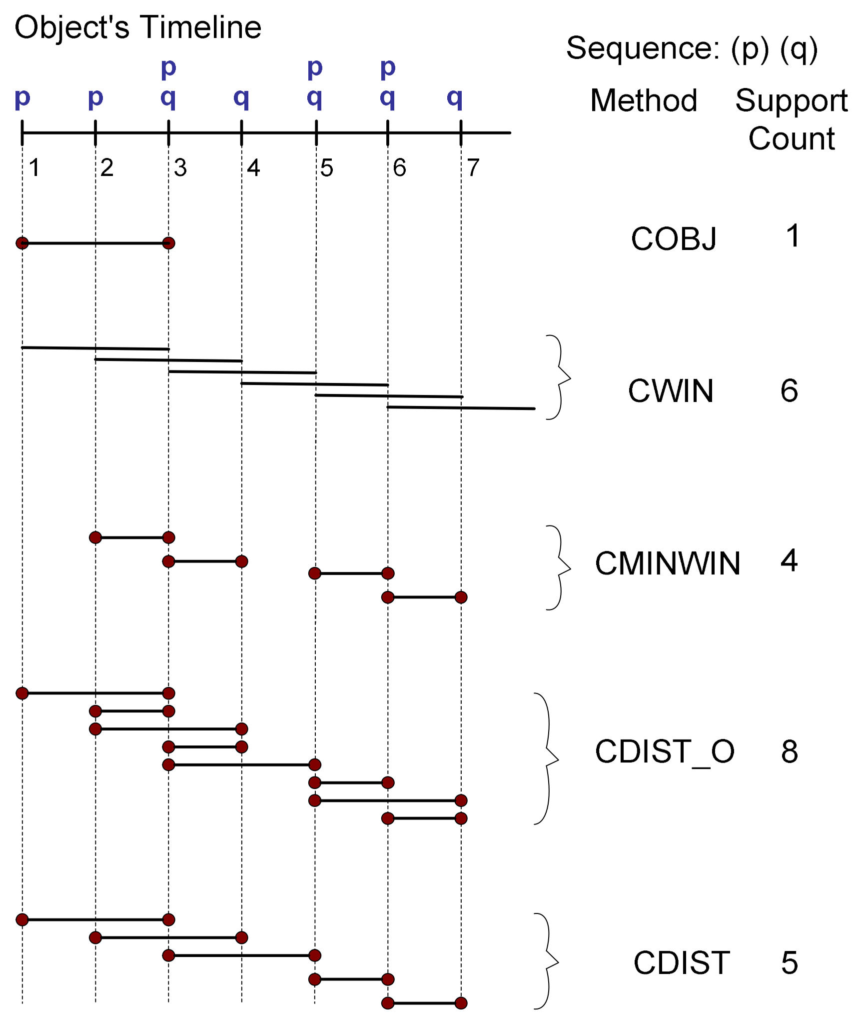
Other Formulation

- 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



Pattern: <E1> <E3>

General Support Counting Schemes



Assume:

x_g = 2 (max-gap)

n_g = 0 (min-gap)

ws = 0 (window size)

m_s = 2 (maximum span)