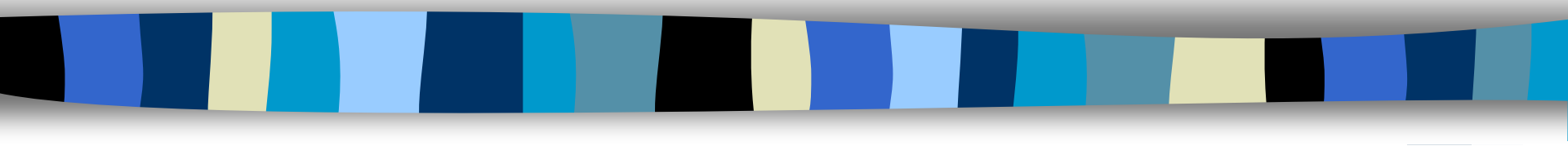
Pattern Sequenziali



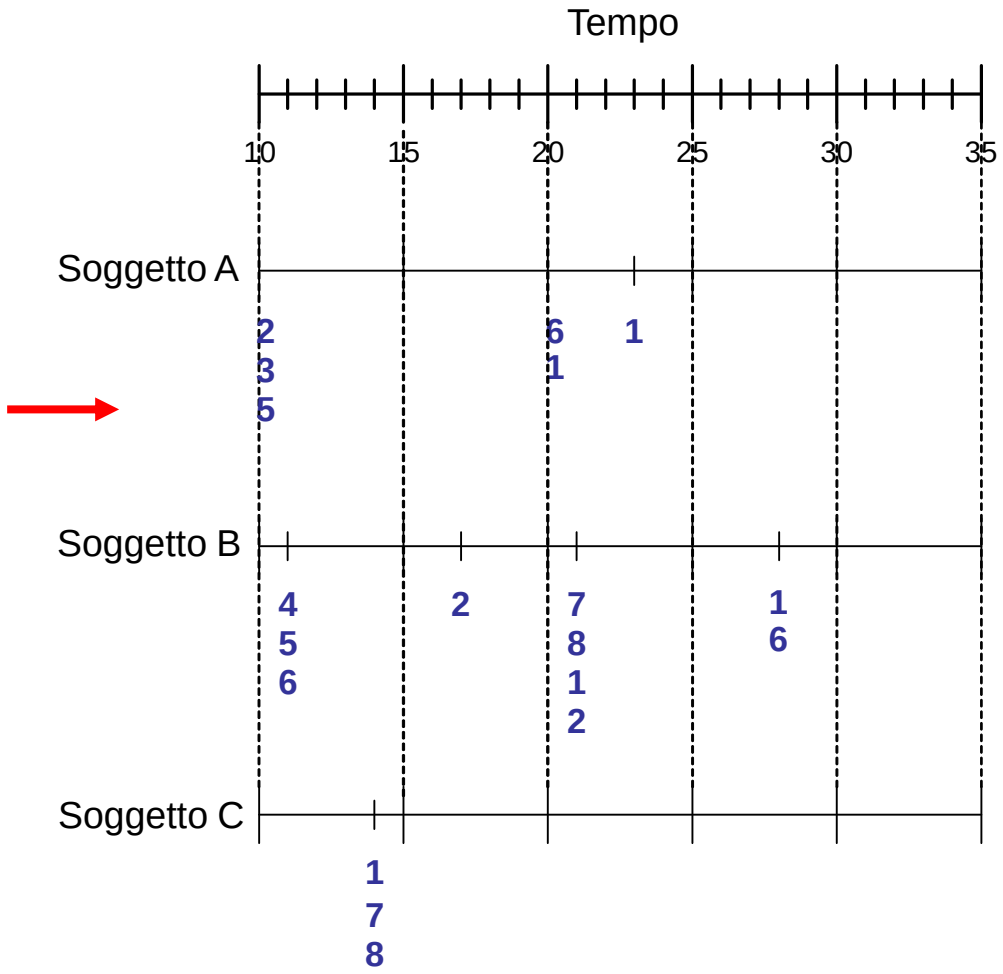
Prof. Matteo Golfarelli
Alma Mater Studiorum - Università di Bologna

Pattern sequenziali

- Spesso alle transazioni sono associate informazioni temporali che permettono di collegare tra loro gli eventi che riguardano uno specifico soggetto

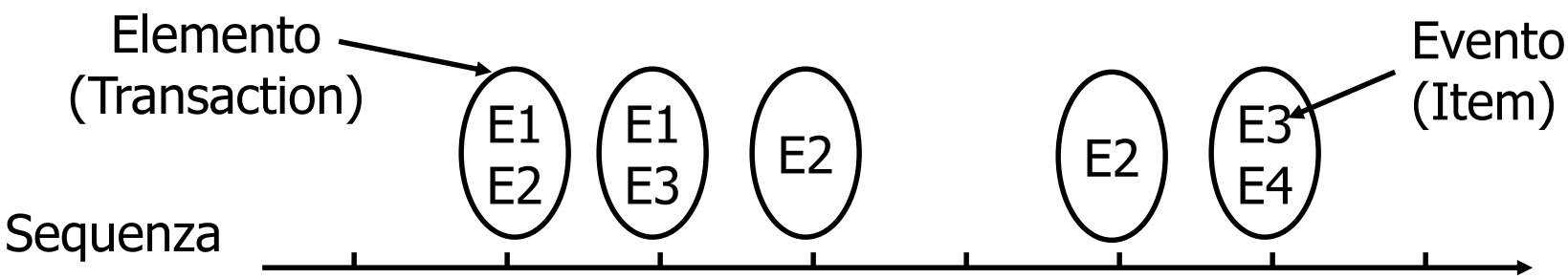
Sequence Database:

Soggetto	Tempo	Eventi
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



Dati sequenziali: alcuni esempi

Database	Sequenza	Elemento (Transaction)	Evento (Item)
Clienti	Storia degli acquisti di un cliente	L'insieme degli item comprati da un cliente al tempo t	Libri,CD, ecc.
Dati web	Attività di browsing di un particolare visitatore web	Una collezione di file visualizzati da un visitatore web dopo un singolo click del mouse	Home page, index page, contact info, ecc.
Eventi	Storia degli eventi generati da un sensore	Eventi scatenati dal sensore al tempo t	Tipi di allarmi generati dal sensore
Sequenze genomiche	Sequenze del DNA di una particolare specie	Un lemento della sequenza del DNA	Basi A,T,G,C



Definizione di sequenza

- Una sequenza è una lista ordinata di elementi (transazioni)

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

  - ✓ Ogni elemento contiene un insieme di eventi (item)

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

  - ✓ A ogni elemento è associato uno specifico istante temporale o posizione ordinale

- La lunghezza della sequenza, |s|, è data dal numero degli elementi che la compongono
- Mentre una k-sequence è una sequenza che contiene k eventi
- ATTENZIONE le sequenze formate da k eventi possono avere lunghezze diverse

  < {1,2,3} > < {1,2} {3} > < {1} {2} {3} >

Definizione di sottosequenza

- Una sequenza $\langle a_1\, a_2 \ldots a_n \rangle$ è contenuta in una sequenza $\langle b_1\, b_2 \ldots b_m \rangle$ ($m \geq n$) se esistono degli interi $i_1 < i_2 < \ldots < i_n$ tali che $a_1 \subseteq b_{i_1}$, $a_2 \subseteq b_{i_2}$, …, $a_n \subseteq b_{i_n}$

Sequenze	Sottosequenze	E' contenuta?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	Si (1,2)
< {1,2} {3,4} >	< {1} {2} >	No
< {2,4} {2,4} {2,5} >	< {2} {4} >	Si (1,2)

- Il supporto di una sottosequenza w è definito come la frazione di sequenze che contengono w
- Un *pattern sequenziale* è una sottosequenza frequente ossia il cui supporto è ≥ *minsup*

Mining di pattern sequenziali

- Dato un database di sequenze e una soglia di supporto minimo, *minsup* trovare tutte le sottosequenze il cui supporto sia ≥ *minsup*

Database di sequenze

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a(bc)dc> è una *sottosequenza* di <a(abc)(ac)d(cf)>

Data *una soglia minsup* =2, <(ab)c> è un *pattern sequenziale*

- La ricerca di pattern sequenziali è un problema difficile visto il numero esponenziale di sottosequenze contenute in una sequenza
  - ✓Il numero di k-sottosequenze contenute in una sequenza con n eventi è: $\binom{n}{k}$
  - ✓Una sequenza con 9 elementi contiene: $\binom{9}{1}+\binom{9}{2}+\ldots+\binom{9}{9}=2^9-1=516$ sequenze

Tecniche per il mining di pattern sequenziali

- Approcci basati sul principio Apriori
  - ✓ **GSP (implementato in Weka)**
  - ✓ SPADE
- Approcci basati sul principio Pattern-Growth
  - ✓ FreeSpan
  - ✓ PrefixSpan

Approccio naive

- Dati n eventi: i_1, i_2, i_3, …, i_n, enumerare tutte le possibili sequenze e calcolare il relativo supporto
  - ✓ 1-sottosequenze candidate:
    $<\{i_1\}>, <\{i_2\}>, <\{i_3\}>, \ldots, <\{i_n\}>$
  - ✓ 2-sottosequenze candidate:
    $<\{i_1, i_2\}>, <\{i_1, i_3\}>, \ldots, <\{i_1\}\ \{i_1\}>, <\{i_1\}\ \{i_2\}>, \ldots, <\{i_{n-1}\}\ \{i_n\}>$
  - ✓ 3-sottosequenze candidate:
    $<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \ldots, <\{i_1, i_2\}\ \{i_1\}>, <\{i_1, i_2\}\ \{i_2\}>, \ldots,$
    $<\{i_1\}\ \{i_1, i_2\}>, <\{i_1\}\ \{i_1, i_3\}>, \ldots, <\{i_1\}\ \{i_1\}\ \{i_1\}>, <\{i_1\}\ \{i_1\}\ \{i_2\}>, \ldots$

- Si noti che rispetto alle regole associative il numero di sottosequenze candidate è di molto superiore al numero degli itemset candidati poiché:
  - ✓ Un item può apparire una sola volta, ma un evento può apparire più volte, poiché nelle sequenze conta l'ordinamento
    $<\{i_1, i_2\}>, <\{i_1\}\ \{i_2\}>, <\{i_2\}\ \{i_1\}>$

Principio Apriori e algoritmo GSP

- Il principio Apriori si può applicare anche nel caso di pattern sequenziali poiché:
  - ✓ qualsiasi sequenza che contenga una particolare k-sequenza *s* deve contenere tutte le (k-1)-sottosequenze di *s*

$k=1$

$F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsupp\}$

// trova le 1-sequence frequenti

repeat

  $k=k+1$

  C_k**=apriori-gen(**F_{k-1}**)** // genera le k-subsequence candidate

  for each sequence $t \in T$

  C_t = subsequence(C_k, t)

  // determina le sottosequenze candidate che compaiono in t

  for each candidate k-subsequce c $\in C_t$

  $\sigma(c) = \sigma(c)+1$ // incrementa il supporto

  end for

  end for

  $F_k = \{c \mid c \in C_k \wedge \sigma(c) \geq N \times minsupp\}$

  // identifica le k-sequenze frequenti

until $F_k = \emptyset$

$Risultato = \bigcup F_k$

Algoritmo GSP: Generalized Sequential Pattern

- **Step 1**:
  - ✓ Fai una prima scansione del DB delle sequenze per individuare tutte le 1-sequenze
- **Step 2**:

  Ripeti fino a che sono scoperte nuove sequenze frequenti
  - ✓ **Generazione dei candidati**:
    - Fondi coppie di sottoseqenze frequenti trovate al passo *k*-1 per generare sequenze candidate che contengono *k* item
  - ✓ **Pruning dei candidati**:
    - Elimina le *k*-sequenze candidate che contengono (*k-1)*-sottosequenze non frequenti
  - ✓ **Conteggio del supporto**:
    - Fai una scansione del DB per trovare il supporto delle sequenze candidate
  - ✓ **Eliminazione dei candidati**:
    - Elimina le *k*-sequenze candidate il cui supporto è effettivamente inferiore a *minsup*

Generazione dei candidati

- Caso base (k=2):
  - ✓ La fusione di due 1-sequenze frequenti $<\{i_1\}>$ e $<\{i_2\}>$ produrrà due 2-sequenze candidate: $<\{i_1\}\ \{i_2\}>$ e $<\{i_1, i_2\}>$
- Caso generale (k>2):
  - ✓ Una (*k-1*)-sequenza frequente w_1 è fusa con un'altra (*k-1*)-sequenza frequente w_2 per produrre una *k*-sequenza candidata se rimuovendo il primo evento in w_1 e rimuovendo l'ultimo evento in w_2 si ottiene la stessa sottosequenza
  - ✓ La k-sequenza ottenuta corrisponde a w_1 estesa con l'ultimo evento in w_2.
    - Se gli ultimi due eventi in w_2 appartengono allo stesso elemento, allora l'ultimo evento in w_2 diventa parte dell'ultimo elemento in w_1
    - Altrimenti, l'ultimo elemento in w_2 diventa un elemento separato aggiunto alla fine di w_1

GSP: un esempio

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

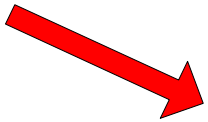
< {1} {2 5} {3} >

- La fusione delle sequenze w_1=<{1} {2} {3}> e w_4 =<{2} {3} {4}> produce la sequenza candidata < {1} {2} {3} {4}> dato che gli eventi {3} e {4} appartengono a elementi separati in w_4

GSP: un esempio

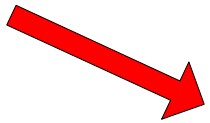
Frequent 3-sequences

< {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} >

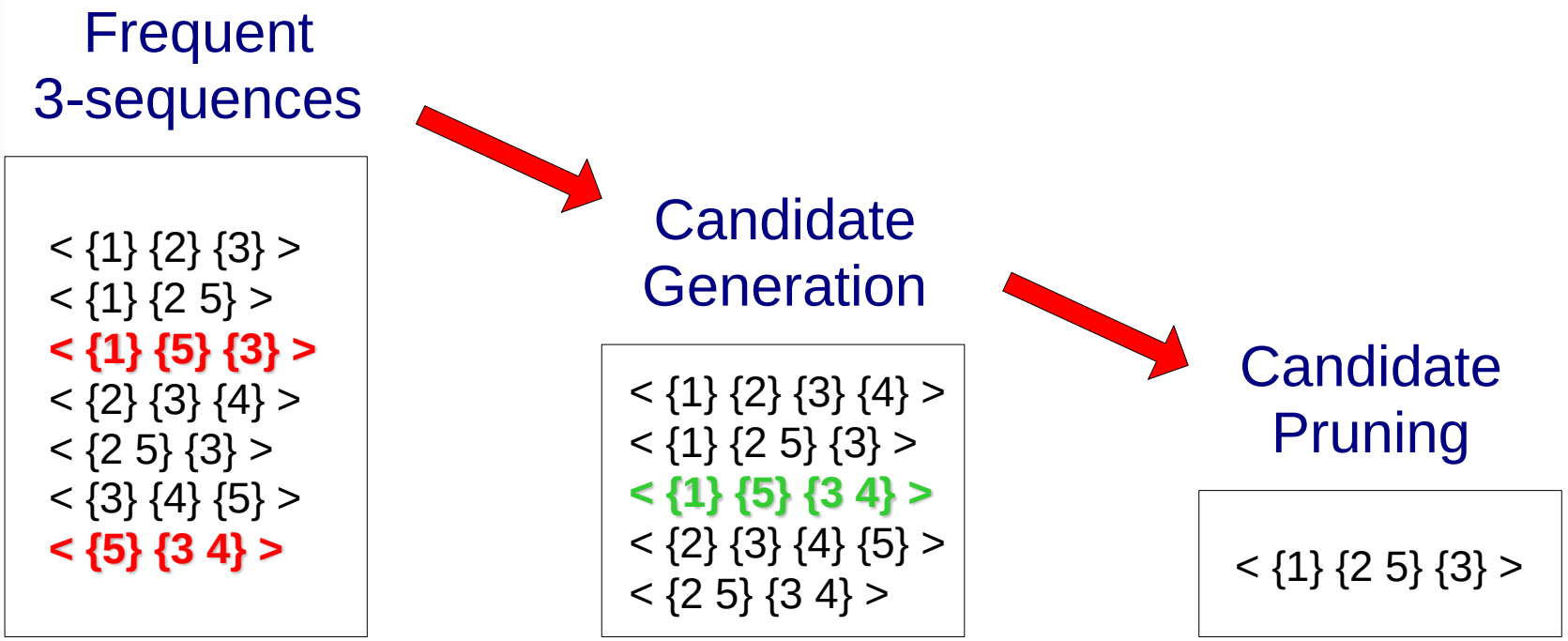


Candidate Pruning

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

- Le sequence w_1 =<{1} {2} {3}> e w_2 =<{1} {2,5}> non devono essere fuse poichè rimuovendo il primo elemento da w_1 e l'ultimo da w_2 non si ottiene la medesima sotto sequenza (<{2} {3}> ≠< {1}>)
- <{1} {2,5} {3}> è un candidato generato fondendo <{1} {2,5} > e <{2,5} {3}> poichè <{1} **{2,5}** > = <**{2,5}** {3}>

GSP: un esempio



- La fusione delle sequenze w_3=<{1} {5} {3}> e w_7 =<{5} {3,4}> produce la sequenza candidata < {1} {5} {3,4}> dato che gli eventi {3} e {4} appartengono allo stesso elemento in w_7

Vincoli temporali

- La ricerca di pattern sequenziali significativi può essere resa più efficace imponendo vincoli temporali sulla struttura delle sequenze:

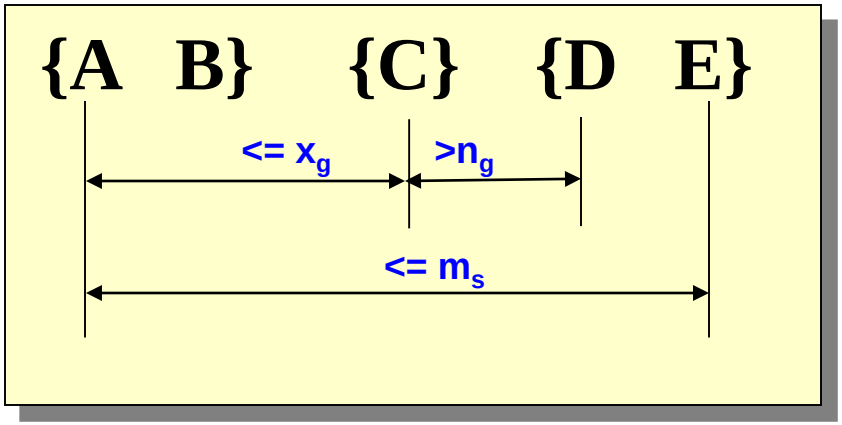
  Studente A < {CPS} {Basi di dati} {Data mining} >

  Studente b < {Basi di dati} {CPS} {Data mining} >

  - ✓ Entambe gli studenti rispondono al requisito in base al quale per poter seguire l'esame di data mining è necessario avere sostenuto gli esami di Basi di dati e Calcolo delle probabilità
  - ✓ Tuttavia i pattern non esprimono il vincolo per cui tali esami non possono essere sostenuti 10 anni prima poiché l'intervallo temporale sarebbe troppo elevato

Vincoli temporali

- **MaxSpan:** specifica il massimo intervallo temporale tra il primo e l'ultimo evento nella sequenza
  - ✓ Aumentando MaxSpan aumenta la probabilità di trovare una sottosequenza in una sequenza ma aumenta anche il rischio di correllare due eventi troppo distanti temporalmente
- **MinGap:** specifica il minimo intervallo temporale che deve trascorrere tra il verificarsi di eventi contenuti in due elementi diversi
- **MaxGap:** specifica il massimo intervallo temporale entro il quale gli eventi contenuti in un elemento devono svolgersi rispetto a quelli contenuti nell'evento precedente



x_g: MaxGap

n_g: MinGap

m_s: MaxSpan

Vincoli temporali: un esempio

- Assumendo che gli elementi siano eseguiti in istanti successivi, si valuti se le seguenti sottosequenze soddisfano i seguenti vincoli temporali
  - ✓ MaxSpan=4
  - ✓ MinGap=1
  - ✓ MaxGap=2

Sequenze	Sottosequenze	Soddisfa?
< {2,4} {3,5,**6**} {4,7} {4,**5**} {8} >	< {6} {5} >	SI
< {**1**} {2} {3} {**4**} {5}>	< {1} {4} >	No MaxGap
< {1} {**2**,3} {**3**,4} {4,**5**}>	< {2} {3} {5} >	SI
< {**1**,**2**} {3} {2,3} {3,4} {2,4} {4,**5**}>	< {1,2} {5} >	No MaxSpan+MaxGap

Mining di pattern sequenziali con vincoli temporali

- I vincoli precedenti incidono sul supporto dei pattern riducendolo
  - ✓ Alcuni pattern conteggiati come frequenti potrebbero non esserlo poichè alcune delle sequenze nel loro supporto potrebbero violare un vincolo temporale
  - ✓ E' necessario modificare le tecniche di conteggio per tenere conto di questo problema
- Sono possibili due soluzioni
  - ✓ Approccio 1
    - Calcolare le sottosequenze frequenti senza considerare i vincoli temporali
    - Applicare i vincoli temporali a posteriori
  - ✓ Approccio 2
    - Modificare GSP per eliminare direttamente i candidati che violano i vincoli temporali
    - ATTENZIONE questa soluzione può portare alla violazione del principio APriori per il vincolo MaxGap

Mining di pattern sequenziali con vincoli temporali

Soggetto	Timestamp	Eventi
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

Supponiamo che:

x_g = 1 (max-gap)

n_g = 1 (min-gap)

m_s = 5 (maximum span)

minsup = 60%

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

ma

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

Esplicitare le sequenze contenute nelle transazioni

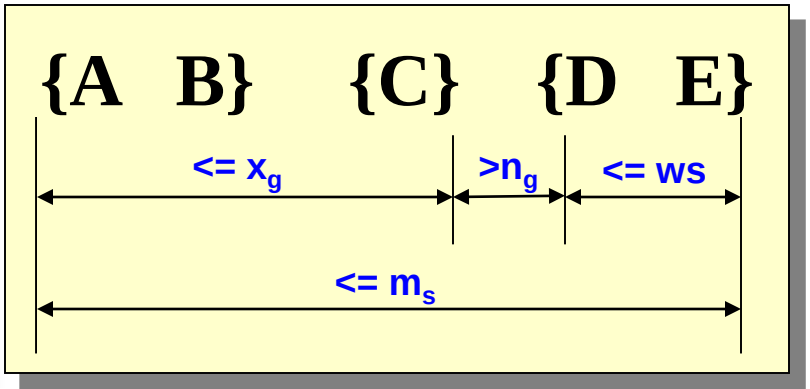
Il problema nasce dalla violazione del vincolo MaxGap che è invece soddisfatto se si inserisce l'elemento {3} che riduce i tempi tra elementi successivi

Vincoli temporali

- Un ulteriore tipo di vincolo temporale che però rilassa la definizione di base è quello di **Time Window Size** (ws) ossia l'intervallo temporale entro il quale due eventi avvenuti in tempi diversi devono essere considerati *contemporanei*

- Dato un pattern candidato <{a, c}> qualsiasi sequenza che contenga:
  - ✓ <… {a c} … >,
  - ✓ <… {a} … {c}…> (con time({c}) – time({a}) ≤ ws)
  - ✓ <…{c} … {a} …> (con time({a}) – time({c}) ≤ ws)

  contribuisce al supporto del pattern candidato



x_g: max-gap

n_g: min-gap

ws: window size

m_s: maximum span

Vincoli temporali: un esempio

- Assumendo che gli elementi siano eseguiti in istanti successivi, si valuti se le seguenti sottosequenze soddisfano i seguenti vincoli temporali
  - ✓ MaxSpan=5
  - ✓ MinGap=1
  - ✓ MaxGap=2
  - ✓ WindowSize=1

Sequenze	Sottosequenze	Soddisfa?
< {2,4} {**3**,**5**,6} {4,7} {4,6} {8} >	< {3} {5} >	No
< {**1**} {**2**} {**3**} {4} {5}>	< {1,2} {3} >	Si
< {**1**,**2**} {2,3} {**3**,**4**} {4,5}>	< {1,2} {3,4} >	Si