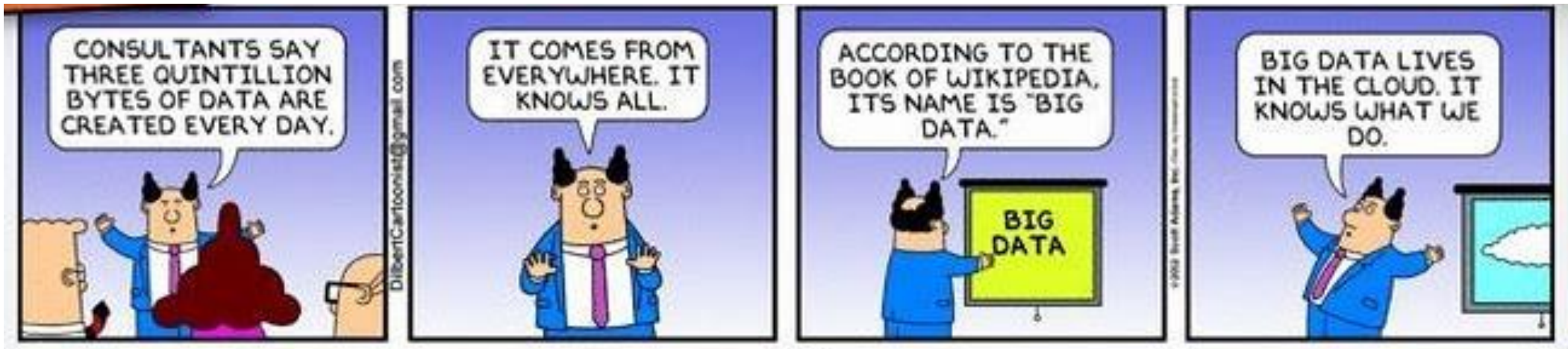
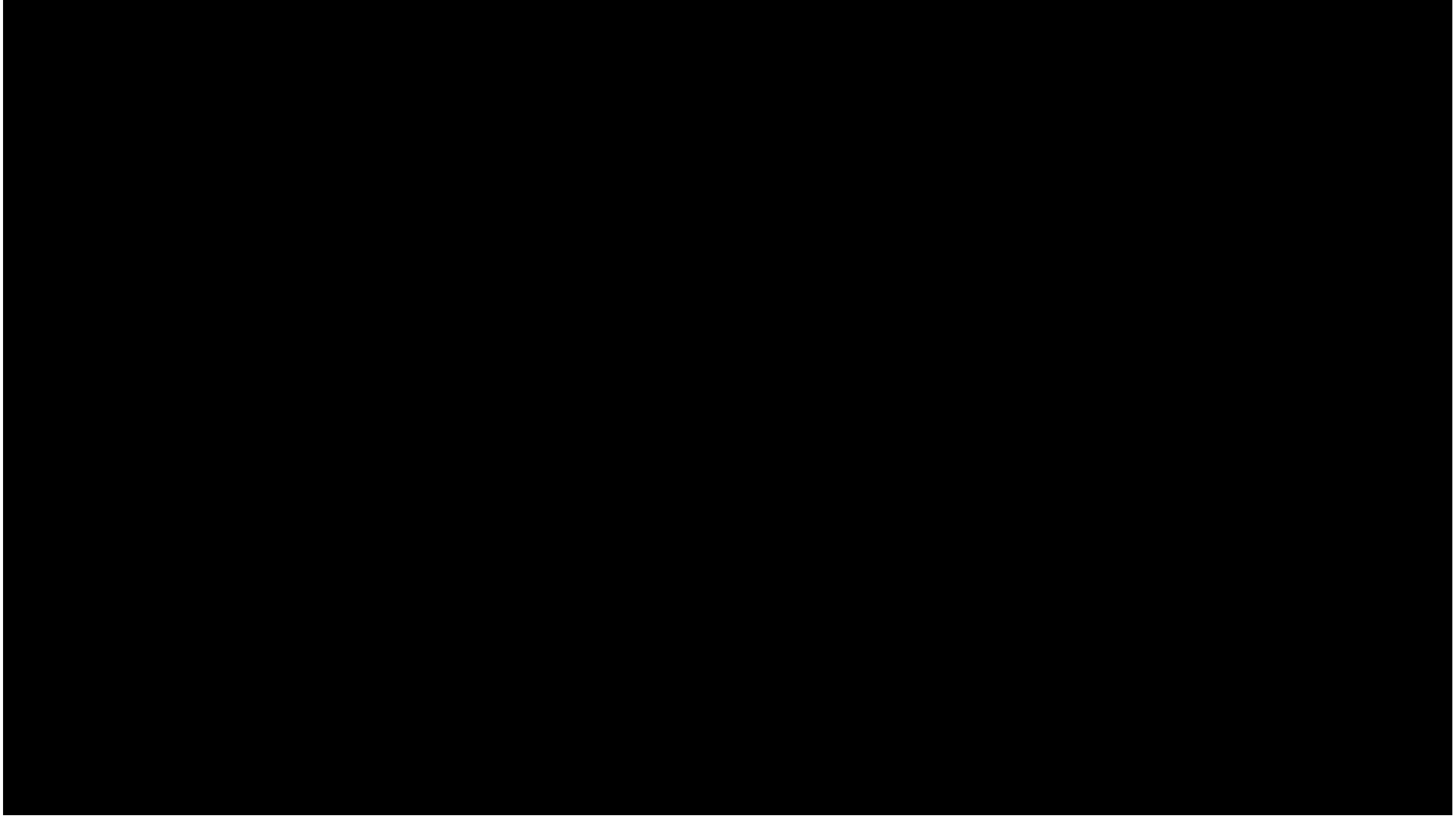

CONSULTANTS SAY THREE QUINTILLION BYTES OF DATA ARE CREATED EVERY DAY.
DilbertCartoonist@gmail.com
IT COMES FROM EVERYWHERE. IT KNOWS ALL.
ACCORDING TO THE BOOK OF WIKIPEDIA, ITS NAME IS "BIG DATA."
BIG DATA
BIG DATA LIVES IN THE CLOUD. IT KNOWS WHAT WE DO.

https://www.datakind.org/ https://d4g.community/ https://www.datascienceforsocialgood.org/



Sequence Mining

Types of sequential data

- **Sequential Data**: order matters, but there is no notion of time, nor how long has passed between each item of the sequence

- **Time-series Data**: in addition to order, the timestamp, or how long has passed between items of the sequence, also matters

Sequence Data

Transaction Database

Customer Id	TransactionTime	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

Sequence Database

Customer Id	Customer Sequence
1	⟨ (30) (90) ⟩
2	⟨ (10 20) (30) (40 60 70) ⟩
3	⟨ (30 50 70) ⟩
4	⟨ (30) (40 70) (90) ⟩
5	⟨ (90) ⟩

A **sequence** contains ordered elements/events/itemsets.
Elements/events/itemsets contain unordered items.

Subsequences & Substrings

- **Subsequence**: A sequence $s_a = \langle A_1, A_2, ..., A_n \rangle$ is said to be contained in another sequence $s_b = \langle B_1, B_2, ..., B_m \rangle$ if and only if there exist integers $1 \leq i_1 < i_2 < ... < i_n \leq m$ such that $A_1 \subseteq B_{i1}, A_2 \subseteq B_{i2}, ..., A_n \subseteq B_{in}$ (denoted as $s_a \sqsubseteq s_b$).
- If a sequence s_a is contained in a sequence s_b, s_a is said to be a subsequence of s_b.

- **Substring** or **consecutive subsequence**: There are no gaps allowed between the elements of s_a in the mapping to s_b.

Ex:

s= ACTGAACG

r_1 = CGAAG

r_2 = CTGA

Ex:

s = < {7} {3 8} {9} {4 5 6} {8} >

r_1 = < {3} {4 5} {8} >

r_2 = < {3} {4} {5} {8} >

r_3 = < {3} {9} {5} >

Subsequences & Substrings

- **Subsequence**: A sequence $s_a = \langle A_1, A_2, ..., A_n \rangle$ is said to be contained in another sequence $s_b = \langle B_1, B_2, ..., B_m \rangle$ if and only if there exist integers $1 \leq i_1 < i_2 < ... < i_n \leq m$ such that $A_1 \subseteq B_{i1}, A_2 \subseteq B_{i2}, ..., A_n \subseteq B_{in}$ (denoted as $s_a \sqsubseteq s_b$).
- If a sequence s_a is contained in a sequence s_b, s_a is said to be a subsequence of s_b.

- **Substring** or **consecutive subsequence**: There are no gaps allowed between the elements of s_a in the mapping to s_b.

Ex:

s= ACTGAACG

r_1 = CGAAG is a subsequence of s

r_2 = CTGA is a substring of s

Ex:

s = < {7} {3 8} {9} {4 5 6} {8} >

r_1 = < {3} {4 5} {8} >

r_2 = < {3} {4} {5} {8} >

r_3 = < {3} {9} {5} >

Subsequences & Substrings

- **Subsequence**: A sequence $s_a = \langle A_1, A_2, ..., A_n \rangle$ is said to be contained in another sequence $s_b = \langle B_1, B_2, ..., B_m \rangle$ if and only if there exist integers $1 \leq i_1 < i_2 < ... < i_n \leq m$ such that $A_1 \subseteq B_{i1}, A_2 \subseteq B_{i2}, ..., A_n \subseteq B_{in}$ (denoted as $s_a \sqsubseteq s_b$).
- If a sequence s_a is contained in a sequence s_b, s_a is said to be a subsequence of s_b.

- **Substring** or **consecutive subsequence**: There are no gaps allowed between the elements of s_a in the mapping to s_b.

Ex:

s= ACTGAACG

r_1 = CGAAG is a subsequence of s

r_2 = CTGA is a substring of s

Ex:

s = < {7} {3 8} {9} {4 5 6} {8} >

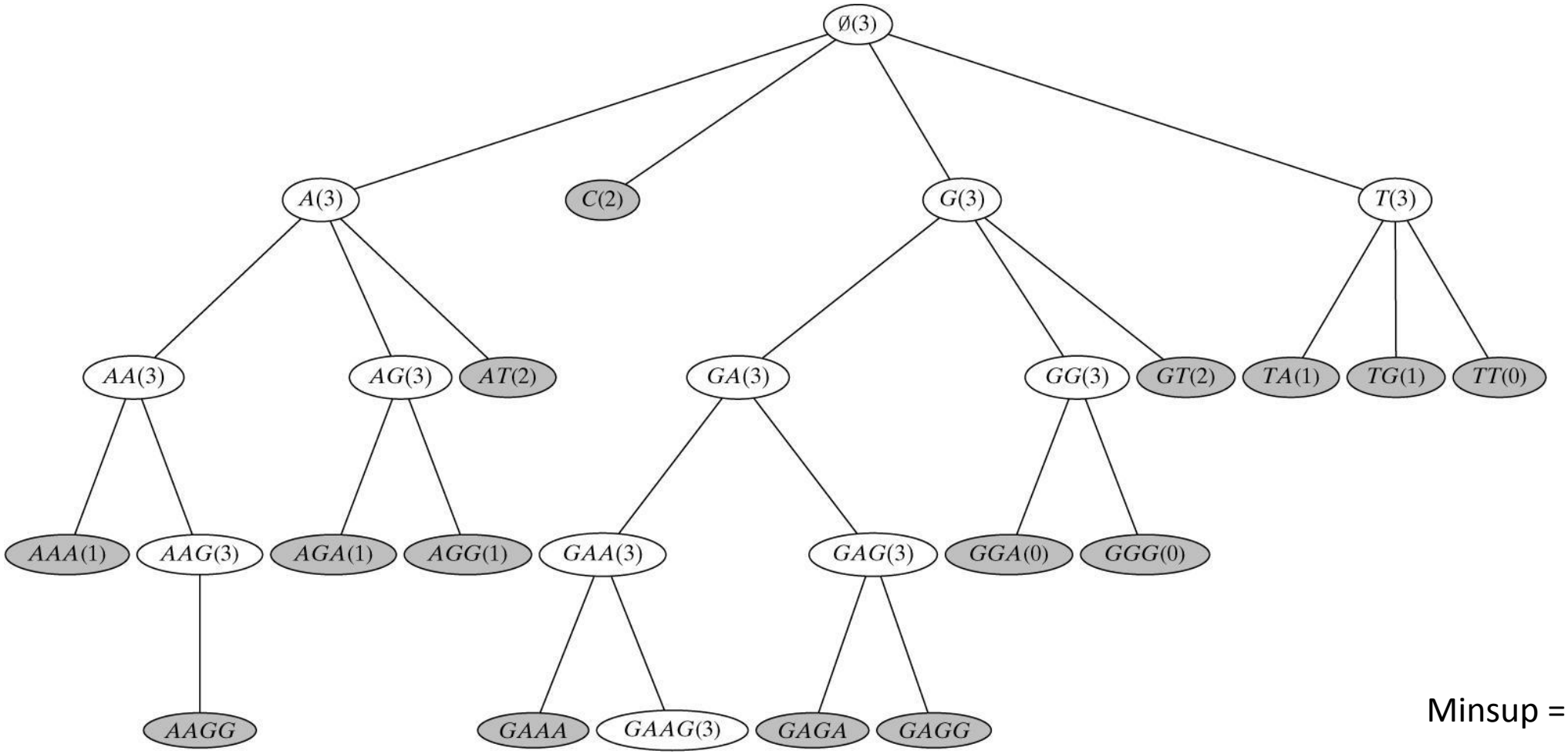
r_1 = < {3} {4 5} {8} > is a subsequence of s

r_2 = < {3} {4} {5} {8} > is neither

r_3 = < {3} {9} {5} > is a substring of s

Prefix Search Tree

- The sequence search space can be ordered in a prefix search tree



Minsup = 3

GSP (Generalized Sequential Pattern) Algorithm

- Apriori-based: level-wise/breadth-first search
- Given the set of frequent sequences at level k, it generates all candidate k+1-sequences
- A sequence $s = \langle S1, S2, ..., Sn \rangle$ is said to be of length k, or a k-sequence, if it contains k items, or in other words, $k = |S1| + |S2| + \cdots + |Sn|$.

For example, the sequence $\langle (a, b), (c), (f, g), (g), (e) \rangle$ is a 7-sequence.

- Prune based on the apriori principle (the antimonotonic property of support: all subsequences of a frequent sequence must be frequent)

GSP (Generalized Sequential Pattern) Algorithm

- **Step 1**:
  - Make the first pass over the sequence database D to yield all frequent 1-subsequences
- **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*

GSP Example

Transaction Database

Transaction Date	Customer ID	Items Purchased
1	01	A
1	02	B
1	03	B
2	04	F
3	01	B
3	05	A
4	02	G
4	05	BC
5	03	F
6	04	AB
6	02	D
7	01	FG
7	05	G
8	04	C
8	03	G
9	05	F
9	01	C
9	03	AB
10	01	D
10	05	DE
10	04	D

Sequence Database

Transaction Date	Customer ID	Items Purchased
1	01	A
3	01	B
7	01	FG
9	01	C
10	01	D
1	02	B
4	02	G
6	02	D
1	03	B
5	03	F
8	03	G
9	03	AB
2	04	F
6	04	AB
8	04	C
10	04	D
3	05	A
4	05	BC
7	05	G
9	05	F
10	05	DE

GSP Example

Transaction Database

Transaction Date	Customer ID	Items Purchased
1	01	A
1	02	B
1	03	B
2	04	F
3	01	B
3	05	A
4	02	G
4	05	BC
5	03	F
6	04	AB
6	02	D
7	01	FG
7	05	G
8	04	C
8	03	G
9	05	F
9	01	C
9	03	AB
10	01	D
10	05	DE
10	04	D

Sequence Database

Transaction Date	Customer ID	Items Purchased	Customer Sequence
1	01	A	<AB(FG)CD>
3	01	B	
7	01	FG	
9	01	C	
10	01	D	
1	02	B	<BGD>
4	02	G	
6	02	D	
1	03	B	<BFG(AB)>
5	03	F	
8	03	G	
9	03	AB	
2	04	F	<F(AB)CD>
6	04	AB	
8	04	C	
10	04	D	
3	05	A	<A(BC)GF(DE)>
4	05	BC	
7	05	G	
9	05	F	
10	05	DE	

GSP Example

- Find the frequent 1-sequences

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

Support counting method COBJ: The support for a sequence is the number of customer sequences that contain this sequence as a subsequence.

minsup = 2

GSP Example

- Find the frequent 1-sequences

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

Support counting method COBJ: The support for a sequence is the number of customer sequences that contain this sequence as a subsequence.

Item	Support
A	4
B	5
C	3
D	4
~~E~~	~~1~~
F	4
G	4

minsup = 2

GSP Example

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

- Generate candidate 2-sequences from frequent 1-sequences
- Remember that order matters!
- Also, items can be bought in the same transaction.

Candidate 2-sequences:

	A	B	C	D	F	G
A	AA	AB	AC	AD	AF	AG
B	BA	BB	BC	BD	BF	BG
C	CA	CB	CC	CD	CF	CG
D	DA	DB	DC	DD	DF	DG
F	FA	FB	FC	FD	FF	FG
G	GA	GB	GC	GD	GF	GG

	A	B	C	D	F	G
A		(AB)	(AC)	(AD)	(AF)	(AG)
B			(BC)	(BD)	(BF)	(BG)
C				(CD)	(CF)	(CG)
D					(DF)	(DG)
F						(FG)
G						

GSP Example

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

- Check all candidate 2-sequences against the database to see if they meet the support threshold.

- Here's just the first row of candidate 2-sequences…

AA	AB	AC	AD	AF	AG
<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>
<BGD>	<BGD>	<BGD>	<BGD>	<BGD>	<BGD>
<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>
<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>
<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>

- We end up with these support counts:

~~AA 0~~	AB 2	AC 3	AD 3	AF 2	AG 2
~~BA 1~~	~~BB 1~~	BC 2	BD 4	BF 3	BG 4
~~CA 0~~	~~CB 0~~	~~CC 0~~	CD 3	~~CF 1~~	~~CG 1~~
~~DA 0~~	~~DB 0~~	~~DC 0~~	~~DD 0~~	~~DF 0~~	~~DG 0~~
FA 2	FB 2	FC 2	FD 3	~~FF 0~~	~~FG 1~~
~~GA 1~~	~~GB 1~~	~~GC 1~~	GD 3	~~GF 1~~	~~GG 0~~

(AB)	2
~~(AC)~~	0
~~(AD)~~	0
~~(AF)~~	0
~~(AG)~~	0

~~(BC)~~	~~1~~
~~(BD)~~	0
~~(BF)~~	0
~~(BG)~~	0

~~(CD)~~	0
~~(CF)~~	0
~~(CG)~~	0

~~(DF)~~	0
~~(DG)~~	0

~~(FG)~~	~~1~~

- The frequent 2-sequences are:

AB	AC	AD	AF	AG	BC	BD	BF	BG	CD	FA	FB	FC	FD	GD	(AB)

GSP Example

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

- Generate candidate 3-sequences from frequent 2-sequences

- To do this:
- Remove the first item from each sequence. For AB, we remove A and we're left with B. Do this for all of the sequences (2nd column in table).
- Then you do the same thing, but remove the last item from the sequence (3rd column in table) .
- This is pretty straightforward except when you're dealing with a sequence that starts or ends in a multiple item purchase like (AB). When that happens, you have to remove all of the possible items one at a time and treat them separately. So if you're removing the "first" item from (AB) you remove A and get B leftover, and then you remove B and get A leftover.

GSP Example

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

- Combine the sequences together where their "-1st" and "-Last" columns match. So if we're starting from the top, we see that the 2-item sequence AB matches up with BC, BD, BF, BG and (AB) to create ABC, ABD, ABF, ABG and A(AB).
- The A(AB) one is tricky because you have to remind yourself that the order within the parentheses does not matter and it could also be written A(BA) which makes it easier to see that the B's match up.

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

2-seq	2-seq	3-seq
AB	BC	ABC
AB	BD	ABD
AB	BF	ABF
AB	BG	ABG
AB	(AB)	A(AB)

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

2-seq	2-seq	3-seq
AB	BC	ABC
AB	BD	ABD
AB	BF	ABF
AB	BG	ABG
AB	(AB)	A(AB)
AC	CD	ACD

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

2-seq	2-seq	3-seq
AB	BC	ABC
AB	BD	ABD
AB	BF	ABF
AB	BG	ABG
AB	(AB)	A(AB)
AC	CD	ACD

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

2-seq	2-seq	3-seq
AB	BC	ABC
AB	BD	ABD
AB	BF	ABF
AB	BG	ABG
AB	(AB)	A(AB)
AC	CD	ACD
AF	FA	AFA
AF	FB	AFB
AF	FC	AFC
AF	FD	AFD

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

2-seq	2-seq	3-seq
AB	BC	ABC
AB	BD	ABD
AB	BF	ABF
AB	BG	ABG
AB	(AB)	A(AB)
AC	CD	ACD
AF	FA	AFA
AF	FB	AFB
AF	FC	AFC
AF	FD	AFD
AG	GD	AGD
BC	CD	BCD
BF	FA	BFA
BF	FB	BFB
BF	FC	BFC
BF	FD	BFD
BG	GD	BGD

2-seq	2-seq	3-seq
FA	AB	FAB
FA	AC	FAC
FA	AD	FAD
FA	AF	FAF
FA	AG	FAG
FA	(AB)	F(AB)
FB	BC	FBC
FB	BD	FBD
FB	BF	FBF
FB	BG	FBG
FC	CD	FCD

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

2-seq	2-seq	3-seq
AB	BC	ABC
AB	BD	ABD
AB	BF	ABF
AB	BG	ABG
AB	(AB)	A(AB)
AC	CD	ACD
AF	FA	AFA
AF	FB	AFB
AF	FC	AFC
AF	FD	AFD
AG	GD	AGD
BC	CD	BCD
BF	FA	BFA
BF	FB	BFB
BF	FC	BFC
BF	FD	BFD
BG	GD	BGD

2-seq	2-seq	3-seq
FA	AB	FAB
FA	AC	FAC
FA	AD	FAD
FA	AF	FAF
FA	AG	FAG
FA	(AB)	F(AB)
FB	BC	FBC
FB	BD	FBD
FB	BF	FBF
FB	BG	FBG
FC	CD	FCD
(AB)	BC	(AB)C
(AB)	BD	(AB)D
(AB)	BF	(AB)F
(AB)	BG	(AB)G

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

2-seq	2-seq	3-seq
AB	BC	ABC
AB	BD	ABD
AB	BF	ABF
AB	BG	ABG
AB	(AB)	A(AB)
AC	CD	ACD
AF	FA	AFA
AF	FB	AFB
AF	FC	AFC
AF	FD	AFD
AG	GD	AGD
BC	CD	BCD
BF	FA	BFA
BF	FB	BFB
BF	FC	BFC
BF	FD	BFD
BG	GD	BGD

2-seq	2-seq	3-seq
FA	AB	FAB
FA	AC	FAC
FA	AD	FAD
FA	AF	FAF
FA	AG	FAG
FA	(AB)	F(AB)
FB	BC	FBC
FB	BD	FBD
FB	BF	FBF
FB	BG	FBG
FC	CD	FCD
(AB)	BC	(AB)C
(AB)	BD	(AB)D
(AB)	BF	(AB)F
(AB)	BG	(AB)G
(AB)	AB	(AB)B

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

Have generated all of the candidate 3-sequences.

Now, prune them based on the apriori principle: Prune any 3-sequence that contains an infrequent 2-sequence.

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

2-seq	2-seq	3-seq	support
AB	BC	ABC	
AB	BD	ABD	
AB	BF	ABF	
AB	BG	ABG	
AB	(AB)	A(AB)	X (AA is not frequent)
AC	CD	ACD	
AF	FA	AFA	X (AA is not frequent)
AF	FB	AFB	
AF	FC	AFC	
AF	FD	AFD	
AG	GD	AGD	
BC	CD	BCD	
BF	FA	BFA	X (BA is not frequent)
BF	FB	BFB	X (BB is not frequent)
BF	FC	BFC	
BF	FD	BFD	
BG	GD	BGD	

2-seq	2-seq	3-seq	support
FA	AB	FAB	
FA	AC	FAC	
FA	AD	FAD	
FA	AF	FAF	X (FF is not frequent)
FA	AG	FAG	X (FG is not frequent)
FA	(AB)	F(AB)	
FB	BC	FBC	
FB	BD	FBD	
FB	BF	FBF	X (FF is not frequent)
FB	BG	FBG	X (FG is not frequent)
FC	CD	FCD	
(AB)	BC	(AB)C	
(AB)	BD	(AB)D	
(AB)	BF	(AB)F	
(AB)	BG	(AB)G	
(AB)	AB	(AB)B	X (BB is not frequent)

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

2-seq	2-seq	3-seq	support
AB	BC	ABC	1
AB	BD	ABD	2
AB	BF	ABF	2
AB	BG	ABG	2
AB	(AB)	A(AB)	X (AA is not frequent)
AC	CD	ACD	3
AF	FA	AFA	X (AA is not frequent)
AF	FB	AFB	0
AF	FC	AFC	1
AF	FD	AFD	2
AG	GD	AGD	2
BC	CD	BCD	2
BF	FA	BFA	X (BA is not frequent)
BF	FB	BFB	X (BB is not frequent)
BF	FC	BFC	1
BF	FD	BFD	2
BG	GD	BGD	3

2-seq	2-seq	3-seq	support
FA	AB	FAB	0
FA	AC	FAC	1
FA	AD	FAD	1
FA	AF	FAF	X (FF is not frequent)
FA	AG	FAG	X (FG is not frequent)
FA	(AB)	F(AB)	2
FB	BC	FBC	1
FB	BD	FBD	1
FB	BF	FBF	X (FF is not frequent)
FB	BG	FBG	X (FG is not frequent)
FC	CD	FCD	2
(AB)	BC	(AB)C	1
(AB)	BD	(AB)D	1
(AB)	BF	(AB)F	0
(AB)	BG	(AB)G	0
(AB)	AB	(AB)B	X (BB is not frequent)

2-seq	2-seq	3-seq	support
~~AB~~	~~BC~~	~~ABC~~	~~1~~
AB	BD	ABD	2
AB	BF	ABF	2
AB	BG	ABG	2
~~AB~~	~~(AB)~~	~~A(AB)~~	~~X (AA is not frequent)~~
AC	CD	ACD	3
~~AF~~	~~FA~~	~~AFA~~	~~X (AA is not frequent)~~
~~AF~~	~~FB~~	~~AFB~~	~~0~~
~~AF~~	~~FC~~	~~AFC~~	~~1~~
AF	FD	AFD	2
AG	GD	AGD	2
BC	CD	BCD	2
~~BF~~	~~FA~~	~~BFA~~	~~X (BA is not frequent)~~
~~BF~~	~~FB~~	~~BFB~~	~~X (BB is not frequent)~~
~~BF~~	~~FC~~	~~BFC~~	~~1~~
BF	FD	BFD	2
BG	GD	BGD	3

2-seq	2-seq	3-seq	support
~~FA~~	~~AB~~	~~FAB~~	~~0~~
~~FA~~	~~AC~~	~~FAC~~	~~1~~
~~FA~~	~~AD~~	~~FAD~~	~~1~~
~~FA~~	~~AF~~	~~FAF~~	~~X (FF is not frequent)~~
~~FA~~	~~AG~~	~~FAG~~	~~X (FG is not frequent)~~
FA	(AB)	F(AB)	2
~~FB~~	~~BC~~	~~FBC~~	~~1~~
~~FB~~	~~BD~~	~~FBD~~	~~1~~
~~FB~~	~~BF~~	~~FBF~~	~~X (FF is not frequent)~~
~~FB~~	~~BG~~	~~FBG~~	~~X (FG is not frequent)~~
FC	CD	FCD	2
~~(AB)~~	~~BC~~	~~(AB)C~~	~~1~~
~~(AB)~~	~~BD~~	~~(AB)D~~	~~1~~
~~(AB)~~	~~BF~~	~~(AB)F~~	~~0~~
~~(AB)~~	~~BG~~	~~(AB)G~~	~~0~~
~~(AB)~~	~~AB~~	~~(AB)B~~	~~X (BB is not frequent)~~

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

Frequent 3-sequences are:

ABD
ABF
ABG
ACD
AFD
AGD
BCD
BFD
BGD
F(AB)
FCD

GSP Example

SID	Sequence
1	<AB(FG)CD>
2	<BGD>
3	<BFG(AB)>
4	<F(AB)CD>
5	<A(BC)GF(DE)>

3-seq	-1st	-Last
ABD	BD	AB
ABF	BF	AB
ABG	BG	AB
ACD	CD	AC
AFD	FD	AF
AGD	GD	AG
BCD	CD	BC
BFD	FD	BF
BGD	GD	BG
F(AB)	(AB)	FA
		FB
FCD	CD	FC

3-seq. (1)	3-seq. (2)	4-seq. after join	Support Count
ABF	BFD	ABFD	2
ABG	BGD	ABGD	2

Frequent 4-sequences:
ABFD
ABGD

GSP Example

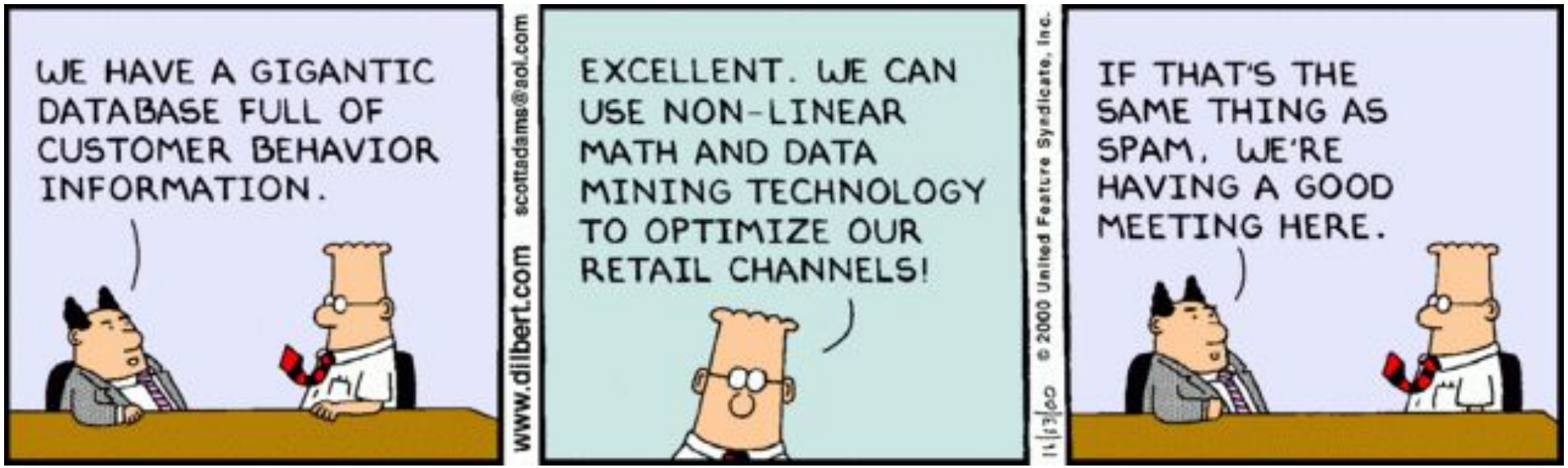
4-seq	-1st	-last
ABFD	BFD	ABF
ABGD	BGD	ABG

No matches between -1st and -last, so the algorithm stops.

Final Frequent Sequences:

Sequences			
1-Item	2-Items	3-Items	4-Items
A	AB	ABD	ABFD
B	AC	ABF	ABGD
C	AD	ABG	
D	AF	ACD	
F	AG	AFD	
G	BC	AGD	
	BD	BCD	
	BF	BFD	
	BG	BGD	
	CD	F(AB)	
	FA	FCD	
	FB		
	FC		
	FD		
	GD		
	(AB)		

There is no rule-gen step. The frequent sequences are the patterns.
Ex: The frequent sequence BGD tells us that when people buy B, they then come back and buy G, then they come back and buy D.


WE HAVE A GIGANTIC DATABASE FULL OF CUSTOMER BEHAVIOR INFORMATION.
www.dilbert.com scottadams@aol.com
EXCELLENT. WE CAN USE NON-LINEAR MATH AND DATA MINING TECHNOLOGY TO OPTIMIZE OUR RETAIL CHANNELS!
© 2000 United Feature Syndicate, Inc.
IF THAT'S THE SAME THING AS SPAM, WE'RE HAVING A GOOD MEETING HERE.

Time Series Constraints

- Maxspan: The maximum allowed time span of the entire sequence
- Maxgap: The maximum allowed time span between two elements of the sequence
- Mingap: The minimum allowed time span between two elements of the sequence

- Important! Using maxgap can cause the apriori principle to be violated

SID	Sequence
1	<{1,2,4}{2,3}{5}>
2	<{1,2}{2,3,4}{2,4,5}>
3	<{2}{3,4}{4,5}>

With no maxgap:
Sup<2 5> = 3
Sup<2 3 5> = 3

With maxgap=1:
Sup<2 5> = 2
Sup<2 3 5> = 3

Support Counting Alternatives

- COBJ: One occurrence per object
- CWIN: One occurrence per sliding window of size maxspan
- CMINWIN: Number of minimal windows of occurrence
- CDIST_O: Distinct occurrences within maxspan, with possibility of event-timestamp overlap
- CDIST: Distinct occurrences within maxspan, with no event-timestamp overlap allowed