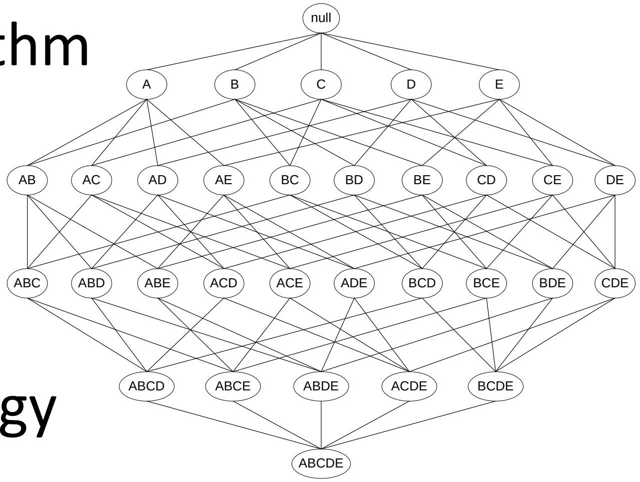
Frequent Itemset Mining & Association Rules (Part 2)

Some slides adapted from UIUC data mining course, and the data mining book by Kumar etc.

Outline

- A different frequent itemset/pattern mining algorithm
- Sequential pattern mining

Why Mining Frequent Patterns by Pattern Growth?



- Apriori: A *breadth-first search* mining algorithm
  - First find the complete set of frequent k-itemsets
  - Then derive frequent (k+1)-itemset candidates
  - Scan DB again to find true frequent (k+1)-itemsets
- Motivation for a different mining methodology
  - Can we develop a *depth-first search* mining algorithm?
  - For a frequent itemset ρ, can subsequent search be confined to only those transactions that contain ρ?
- Such thinking leads to a frequent pattern growth approach:
  - FPGrowth (Han et al "Mining Frequent Patterns without Candidate Generation," SIGMOD 2000)

Example: Frequent Pattern Growth Algorithm

- Mining FP w/o candidate generation.
- Steps:

Built FP-Tree

1. Find the frequency of 1-itemset.
2. Construct Ordered-Item set.
3. Construct FP-tree (i.e., inserting ordered-item set).

Mine FP-Tree

4. Recursively mine FP-tree and grow frequent patterns obtained so far
    a. Construct Conditional database.
    b. Construct conditional FP-tree and generate Frequent Patterns. Repeat the process on each newly created Conditional FP-tree for mining (longer) frequent patterns. *(...until the resulting FP-tree is empty, or it contains only one path.)*

Example: Frequent Pattern Growth Algorithm

- Transaction DB

Transaction ID	Items	Transaction ID	Items
T1	$\{f, a, c, d, g, i, m, p\}$	T5	$\{a, f, c, e, l, p, m, n\}$
T2	$\{a, b, c, f, l, m, o\}$	T6	$\{c, j, m, b, n\}$
T3	$\{b, f, h, j, o\}$	T7	$\{d, e, f, h\}$
T4	$\{b, c, k, s, p\}$	T8	$\{a, g, i, k, s, f\}$

→

1-Itemset Frequency

Item	Frequency
a	4
b	4
c	5
f	6
m	4
p	3
$d, e, g, h, i,$ j, k, l, n, o, s	2

- Step 1:

Find the frequency of 1-itemset.

Example: Frequent Pattern Growth Algorithm

- Transaction DB

Transaction ID	Items	Transaction ID	Items
T1	$\{f, a, c, d, g, i, m, p\}$	T5	$\{a, f, c, e, l, p, m, n\}$
T2	$\{a, b, c, f, l, m, o\}$	T6	$\{c, j, m, b, n\}$
T3	$\{b, f, h, j, o\}$	T7	$\{d, e, f, h\}$
T4	$\{b, c, k, s, p\}$	T8	$\{a, g, i, k, s, f\}$

1-Itemset Frequency

Item	Frequency
a	4
b	4
c	5
f	6
m	4
p	3
~~$d, e, g, h, i, j, k, l, n, o, s$~~	~~2~~

- Step 2:

Construct Ordered-Item set.

  - (Let the minimum support threshold $\boldsymbol{s} = \mathbf{3}$)
  - Frequent Items: (a, b, c, f, m, p)

Example: Frequent Pattern Growth Algorithm

- Transaction DB

Transaction ID	Items	Transaction ID	Items
T1	$\{f,a,c,d,g,i,m,p\}$	T5	$\{a,f,c,e,l,p,m,n\}$
T2	$\{a,b,c,f,l,m,o\}$	T6	$\{c,j,m,b,n\}$
T3	$\{b,f,h,j,o\}$	T7	$\{d,e,f,h\}$
T4	$\{b,c,k,s,p\}$	T8	$\{a,g,i,k,s,f\}$

- Step 2:

Construct Ordered-Item set.

- Sort the frequent items in a descending order of their respective frequencies to form a frequent item set and header table.

Item	Frequency
a	4
b	4
c	5
f	6
m	4
p	3

→

Frequent Item set:

$L = \{f{:}\,6, c{:}\,5, a{:}\,4, b{:}\,4, m{:}\,4, p{:}\,3\}$

Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	

Header Table references the occurrences of the frequent items in the FP-tree

Example: Frequent Pattern Growth Algorithm

- Step 2: Construct Ordered-Item Set
  - For (each transaction):
    For (each item in L):
    If (item) in (transaction):
    Insert (item) to (Ordered-Item Set)

$$L = \{\boxed{f:6}, \boxed{c:5}, \boxed{a:4}, b:4, \boxed{m:4}, \boxed{p:3}\}$$

Transaction ID	Items	Ordered-Item Set	Transaction ID	Items	Ordered-Item Set
T1	$\boxed{\{f, a, c, d, g, i, m, p\}}$	$\{f, c, a, m, p\}$	T5	$\{a, f, c, e, l, p, m, n\}$	
T2	$\{a, b, c, f, l, m, o\}$		T6	$\{c, j, m, b, n\}$	
T3	$\{b, f, h, j, o\}$		T7	$\{d, e, f, h\}$	
T4	$\{b, c, k, s, p\}$		T8	$\{a, g, i, k, s, f\}$	

Example: Frequent Pattern Growth Algorithm

- Step 2: Construct Ordered-Item Set
  - For (each transaction):
    For (each item in L):
    If (item) in (transaction):
    Insert (item) to (Ordered-Item Set)

$$L = \{f: 6, c: 5, a: 4, b: 4, m: 4, p: 3\}$$

Transaction ID	Items	Ordered-Item Set	Transaction ID	Items	Ordered-Item Set
T1	$\{f, a, c, d, g, i, m, p\}$	$\{f, c, a, m, p\}$	T5	$\{a, f, c, e, l, p, m, n\}$	$\{f, c, a, m, p\}$
T2	$\{a, b, c, f, l, m, o\}$	$\{f, c, a, b, m\}$	T6	$\{c, j, m, b, n\}$	$\{c, b, m\}$
T3	$\{b, f, h, j, o\}$	$\{f, b\}$	T7	$\{d, e, f, h\}$	$\{f\}$
T4	$\{b, c, k, s, p\}$	$\{c, b, p\}$	T8	$\{a, g, i, k, s, f\}$	$\{f, a\}$

Example: Frequent Pattern Growth Algorithm

Transaction ID	Items	Ordered-Item Set
T1	$\{f, a, c, d, g, i, m, p\}$	$\boxed{\{f, c, a, m, p\}}$
T2	$\{a, b, c, f, l, m, o\}$	$\{f, c, a, b, m\}$
T3	$\{b, f, h, j, o\}$	$\{f, b\}$
T4	$\{b, c, k, s, p\}$	$\{c, b, p\}$

- Step 3:

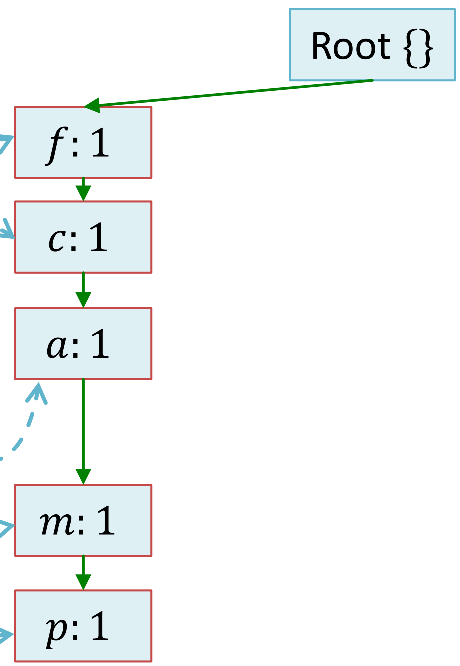
Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

Insert $\{f, c, a, m, p\}$

Header Table

Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	



Example: Frequent Pattern Growth Algorithm

Transaction ID	Items	Ordered-Item Set
T1	$\{f, a, c, d, g, i, m, p\}$	$\{f, c, a, m, p\}$
T2	$\{a, b, c, f, l, m, o\}$	$\boxed{\{f, c, a, b, m\}}$
T3	$\{b, f, h, j, o\}$	$\{f, b\}$
T4	$\{b, c, k, s, p\}$	$\{c, b, p\}$

- Step 3:

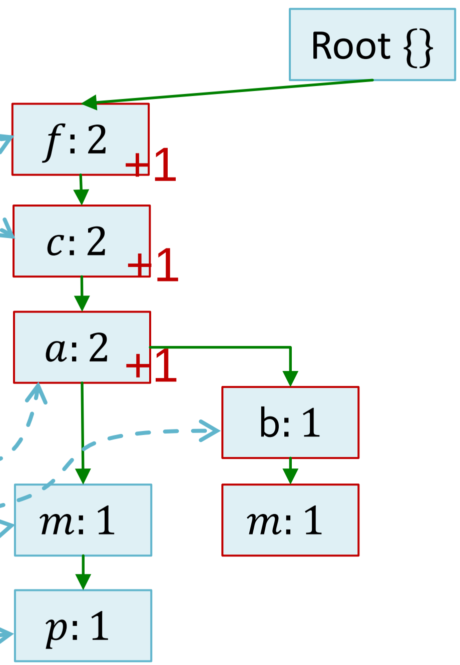
Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

Insert $\{f, c, a, b, m\}$

Header Table

Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	



Example: Frequent Pattern Growth Algorithm

Transaction ID	Items	Ordered-Item Set
T1	$\{f, a, c, d, g, i, m, p\}$	$\{f, c, a, m, p\}$
T2	$\{a, b, c, f, l, m, o\}$	$\{f, c, a, b, m\}$
T3	$\{b, f, h, j, o\}$	$\boxed{\{f, b\}}$
T4	$\{b, c, k, s, p\}$	$\{c, b, p\}$

- Step 3:

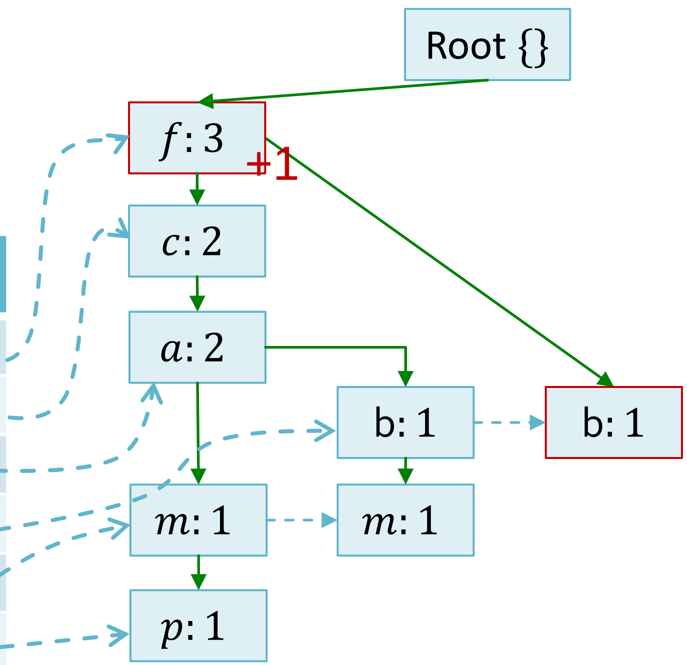
Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

Insert $\{f, b\}$

Header Table

Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	



Example: Frequent Pattern Growth Algorithm

Transaction ID	Items	Ordered-Item Set
T1	$\{f, a, c, d, g, i, m, p\}$	$\{f, c, a, m, p\}$
T2	$\{a, b, c, f, l, m, o\}$	$\{f, c, a, b, m\}$
T3	$\{b, f, h, j, o\}$	$\{f, b\}$
T4	$\{b, c, k, s, p\}$	$\boxed{\{c, b, p\}}$

- Step 3:

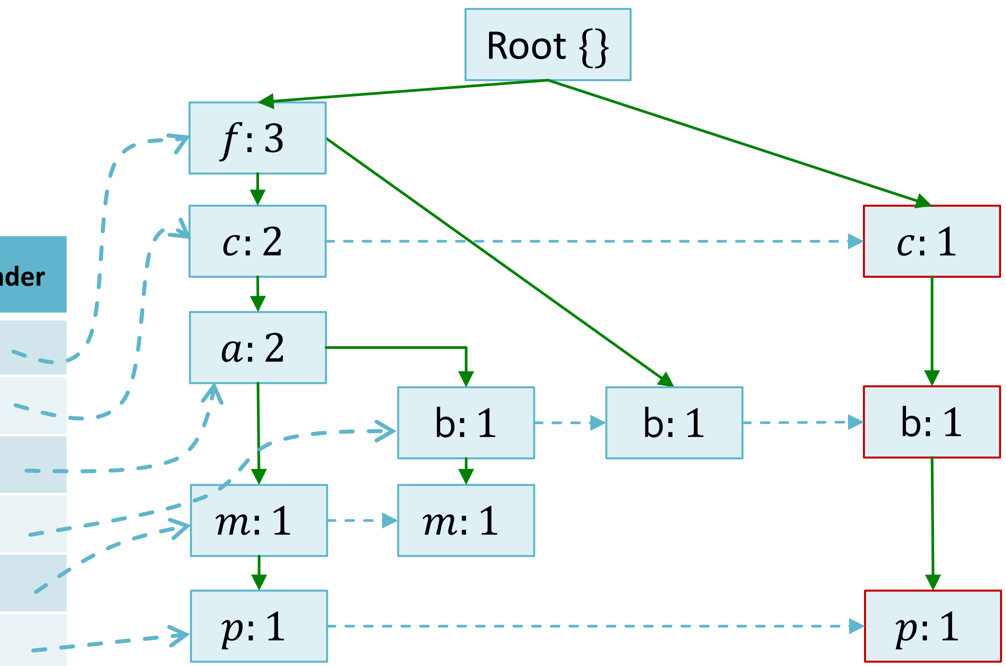
Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

Insert $\{c, b, p\}$

Header Table

Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	



Example: Frequent Pattern Growth Algorithm

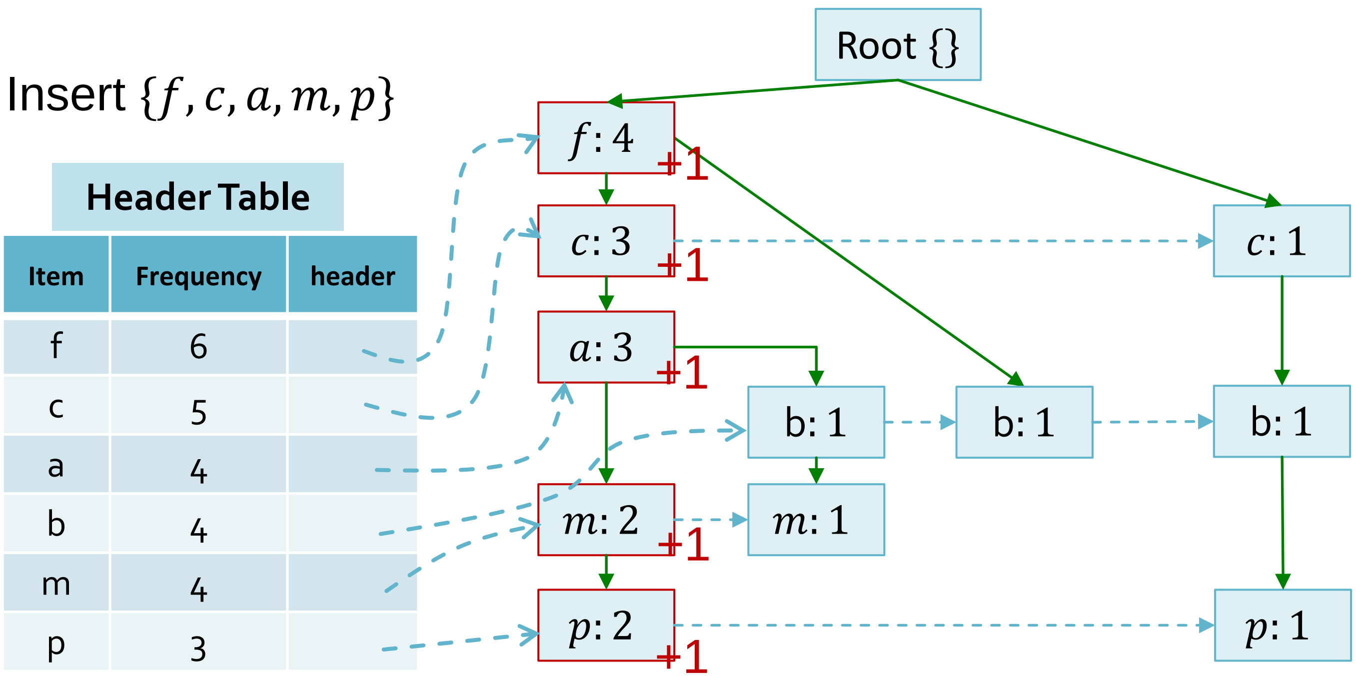
Transaction ID	Items	Ordered-Item Set
T5	$\{a, f, c, e, l, p, m, n\}$	$\boxed{\{f, c, a, m, p\}}$
T6	$\{c, j, m, b, n\}$	$\{c, b, m\}$
T7	$\{d, e, f, h\}$	$\{f\}$
T8	$\{a, g, i, k, s, f\}$	$\{f, a\}$

- Step 3:

Construct FP-tree.

  - Create the root of FP-tree (Null).
  - Insert Ordered-Item Set.

Insert $\{f, c, a, m, p\}$



Example: Frequent Pattern Growth Algorithm

Transaction ID	Items	Ordered-Item Set
T5	$\{a, f, c, e, l, p, m, n\}$	$\{f, c, a, m, p\}$
T6	$\{c, j, m, b, n\}$	$\boxed{\{c, b, m\}}$
T7	$\{d, e, f, h\}$	$\{f\}$
T8	$\{a, g, i, k, s, f\}$	$\{f, a\}$

- Step 3:

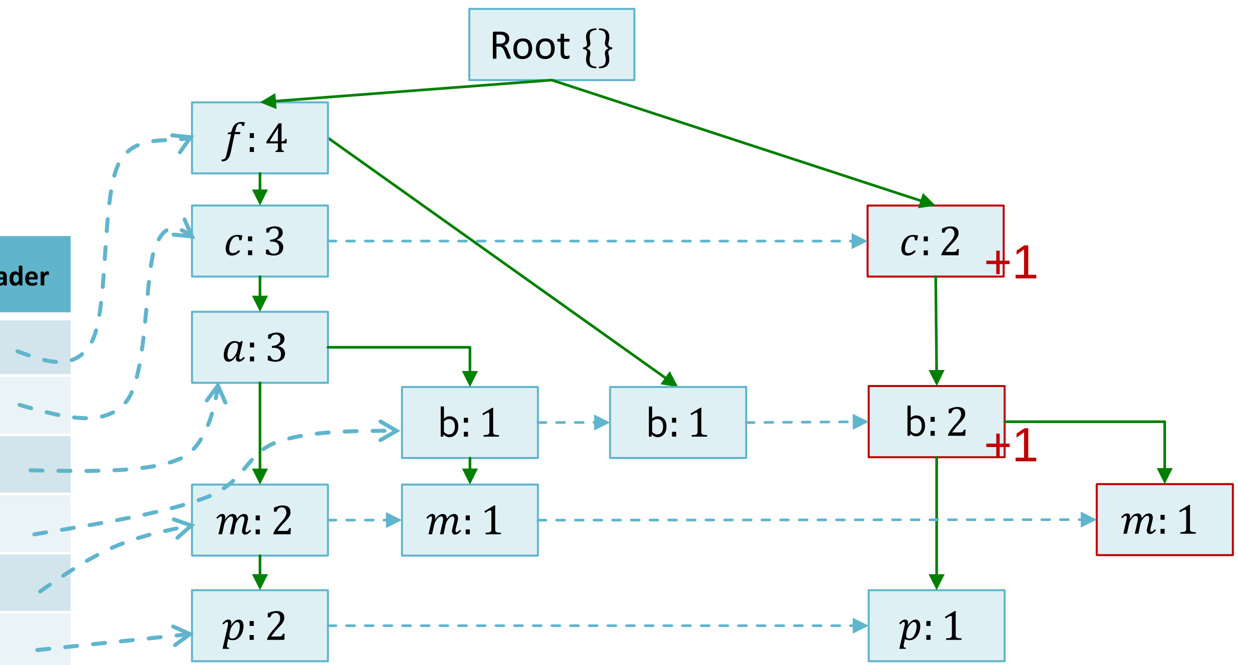
Construct FP-tree.

  - Create the root of FP-tree (Null).
  - Insert Ordered-Item Set.

Insert $\{c, b, m\}$

Header Table

Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	



Example: Frequent Pattern Growth Algorithm

Transaction ID	Items	Ordered-Item Set
T5	$\{a, f, c, e, l, p, m, n\}$	$\{f, c, a, m, p\}$
T6	$\{c, j, m, b, n\}$	$\{c, b, m\}$
T7	$\{d, e, f, h\}$	$\boxed{\{f\}}$
T8	$\{a, g, i, k, s, f\}$	$\{f, a\}$

- Step 3:

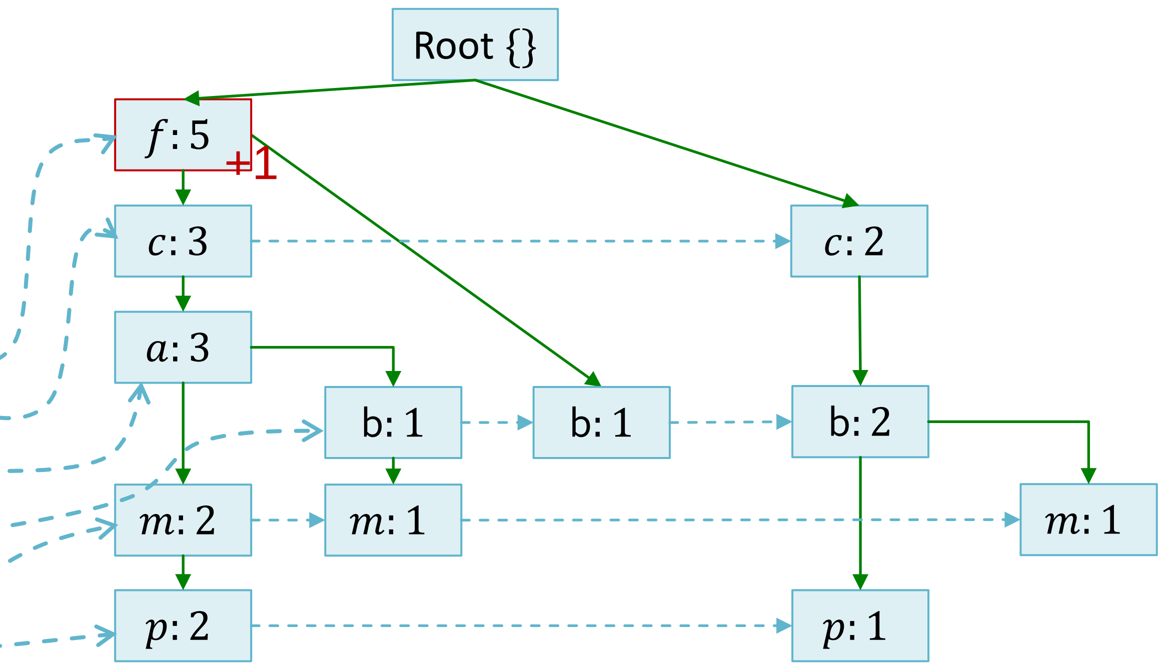
Construct FP-tree.

- Create the root of FP-tree (Null).
- Insert Ordered-Item Set.

Insert $\{f\}$

Header Table

Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	



Example: Frequent Pattern Growth Algorithm

Transaction ID	Items	Ordered-Item Set
T5	$\{a, f, c, e, l, p, m, n\}$	$\{f, c, a, m, p\}$
T6	$\{c, j, m, b, n\}$	$\{c, b, m\}$
T7	$\{d, e, f, h\}$	$\{f\}$
T8	$\{a, g, i, k, s, f\}$	$\boxed{\{f, a\}}$

- Step 3:

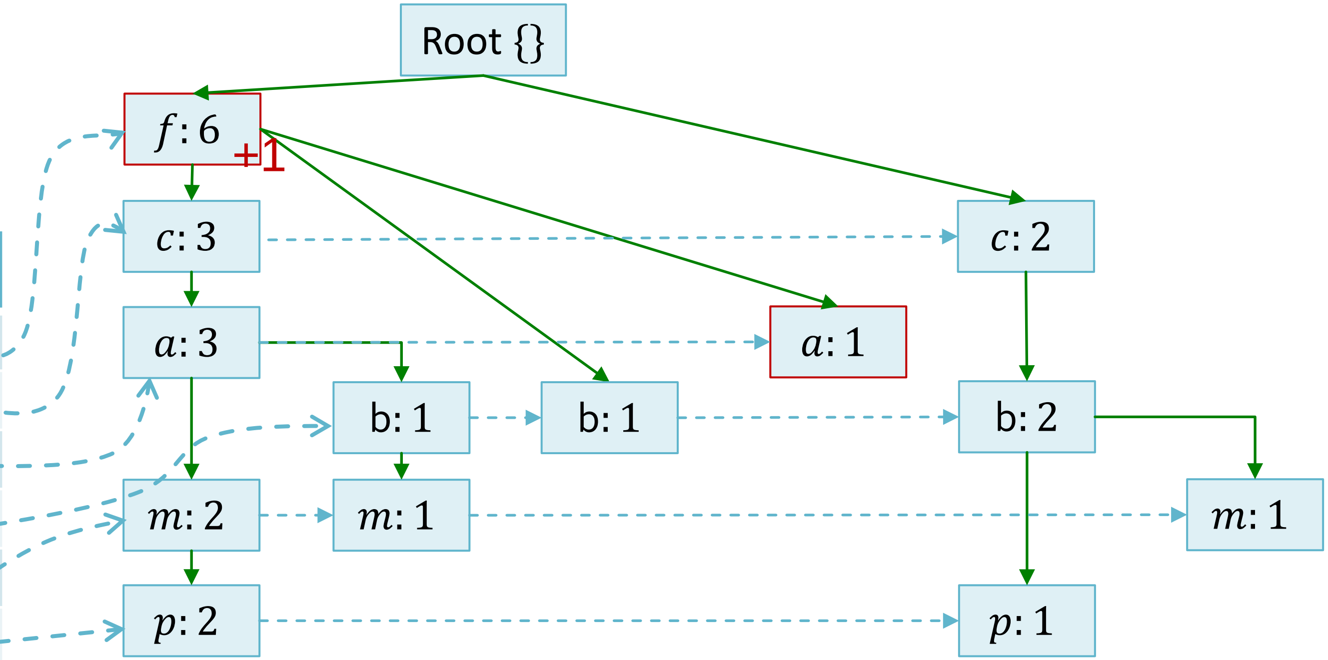
Construct FP-tree.

  - Create the root of FP-tree (Null).
  - Insert Ordered-Item Set.

Insert $\{f, a\}$

Header Table

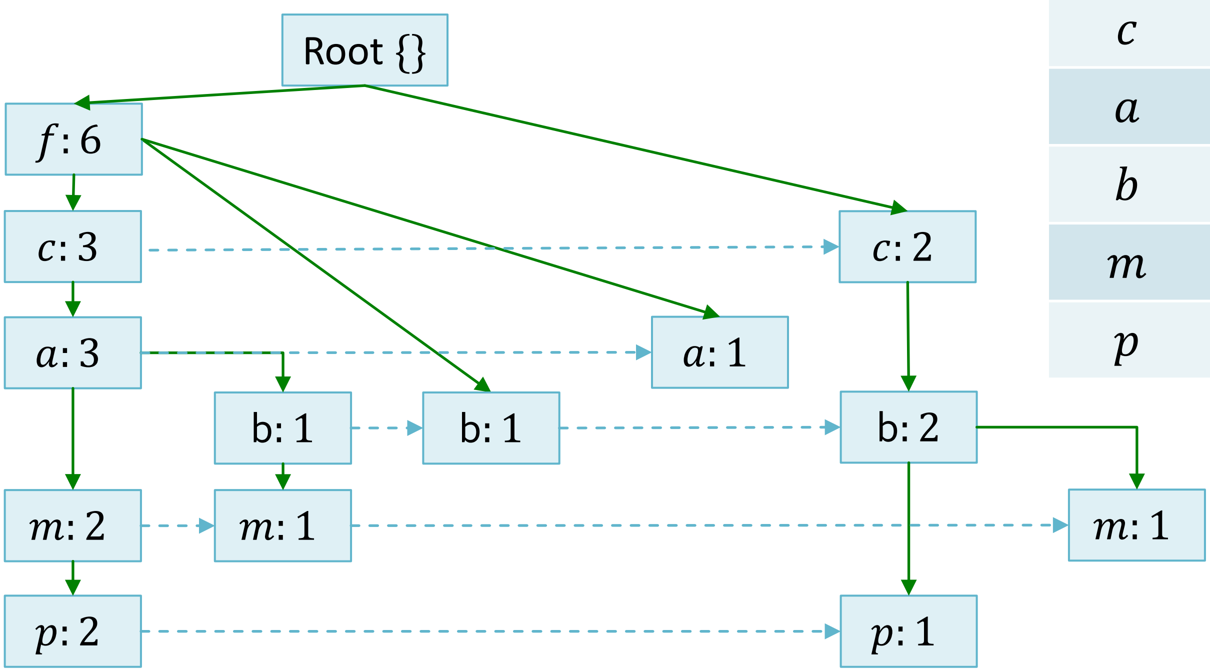
Item	Frequency	header
f	6	
c	5	
a	4	
b	4	
m	4	
p	3	



Example: Frequent Pattern Growth Algorithm

- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).

FP-Tree

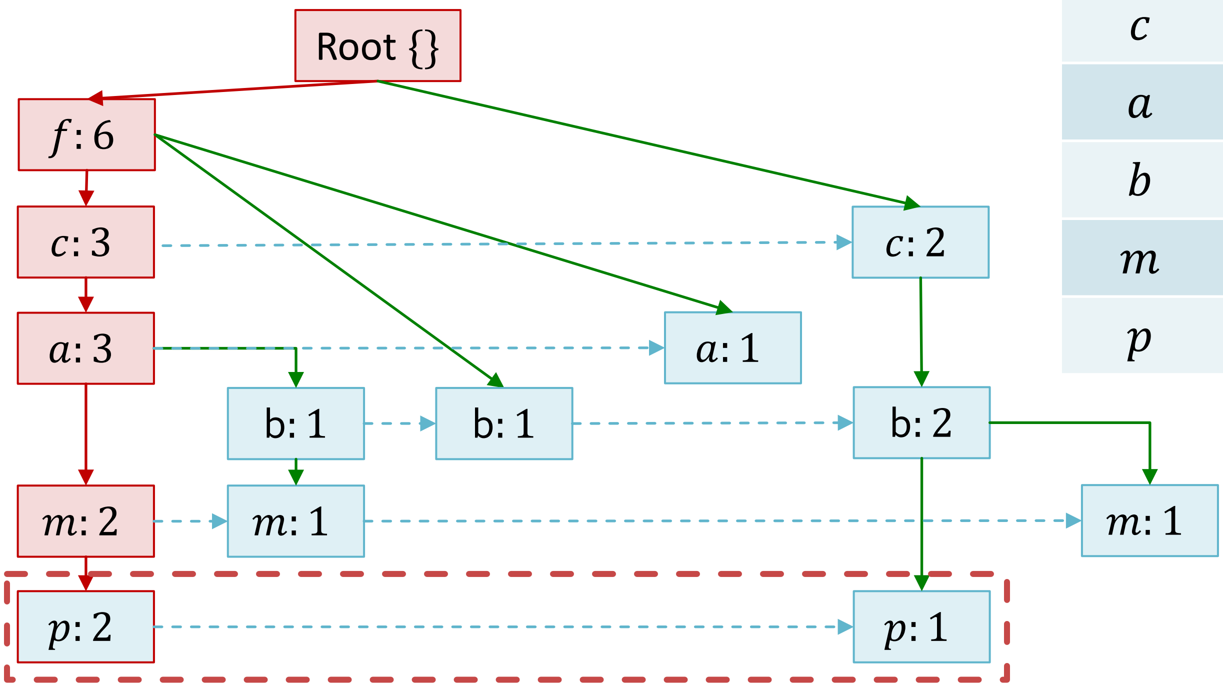


Items	Conditional Data Base
f	
c	
a	
b	
m	
p	

Example: Frequent Pattern Growth Algorithm

- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).

FP-Tree

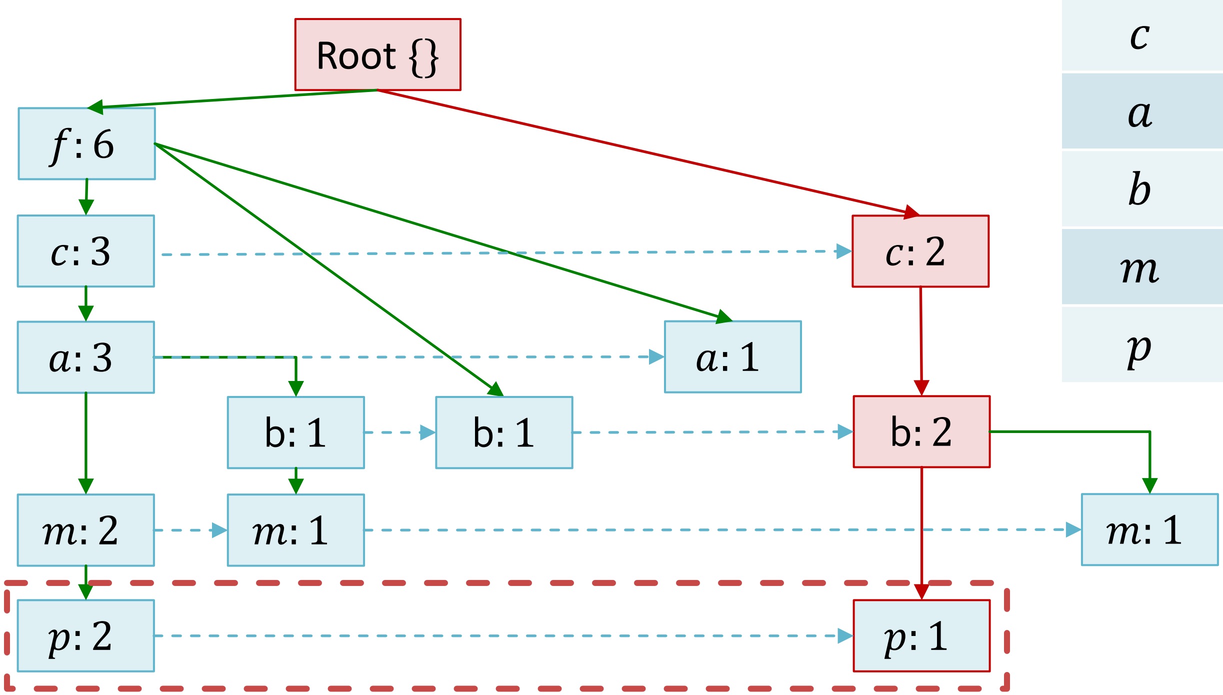


Items	Conditional Data Base
f	
c	
a	
b	
m	
p	$\{f, c, a, m: 2\}$

Example: Frequent Pattern Growth Algorithm

- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).

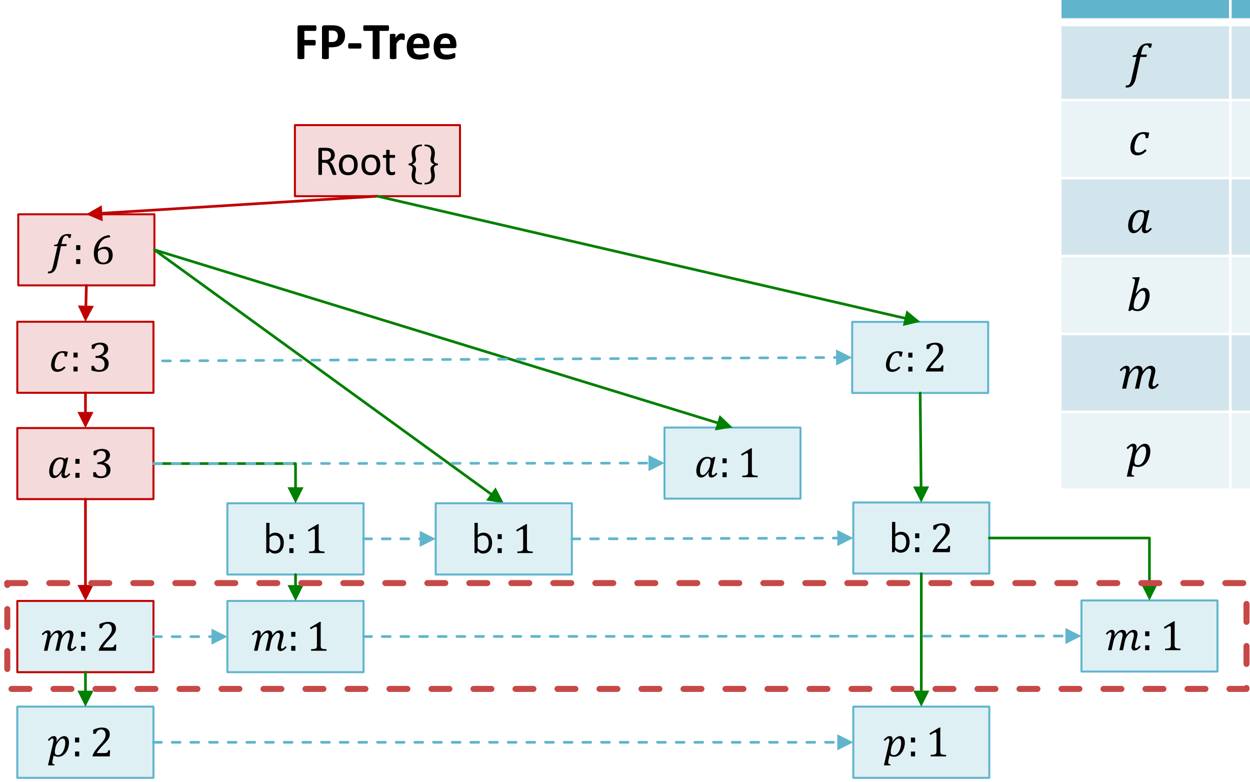
FP-Tree



Items	Conditional Data Base
f	
c	
a	
b	
m	
p	$\{f, c, a, m: 2\}$, $\{c, b: 1\}$

Example: Frequent Pattern Growth Algorithm

- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).

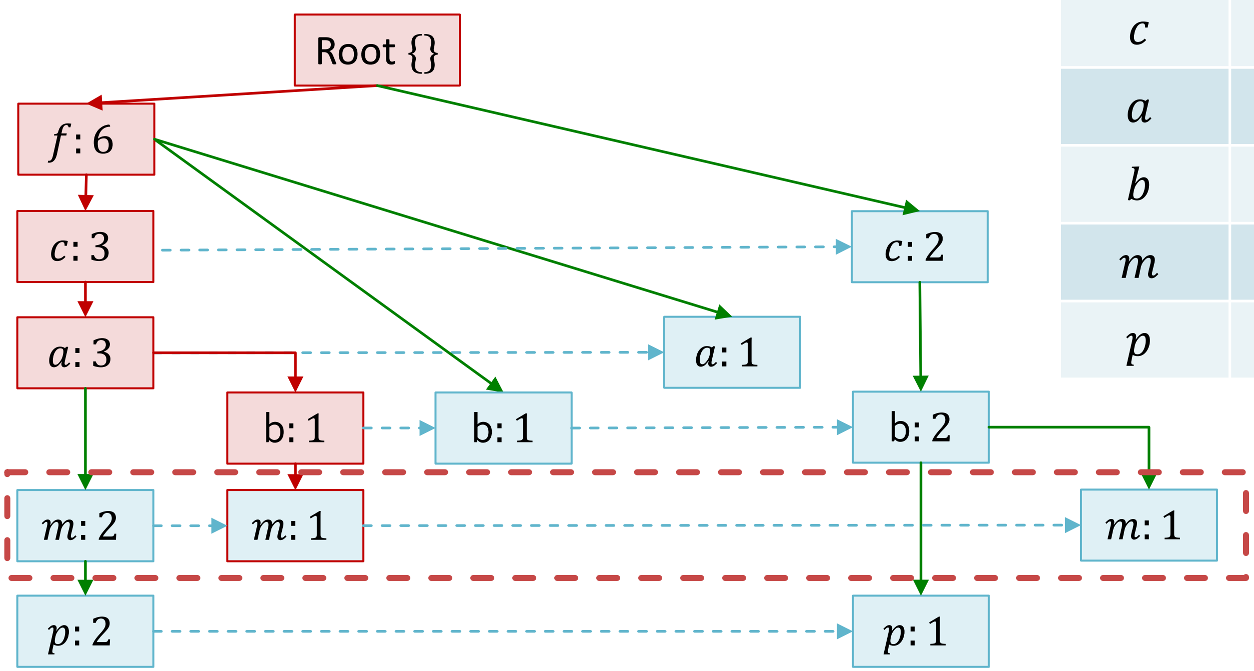


Items	Conditional Data Base
f	
c	
a	
b	
m	$\{f, c, a: 2\}$
p	$\{f, c, a, m: 2\}, \{c, b: 1\}$

Example: Frequent Pattern Growth Algorithm

- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).

FP-Tree

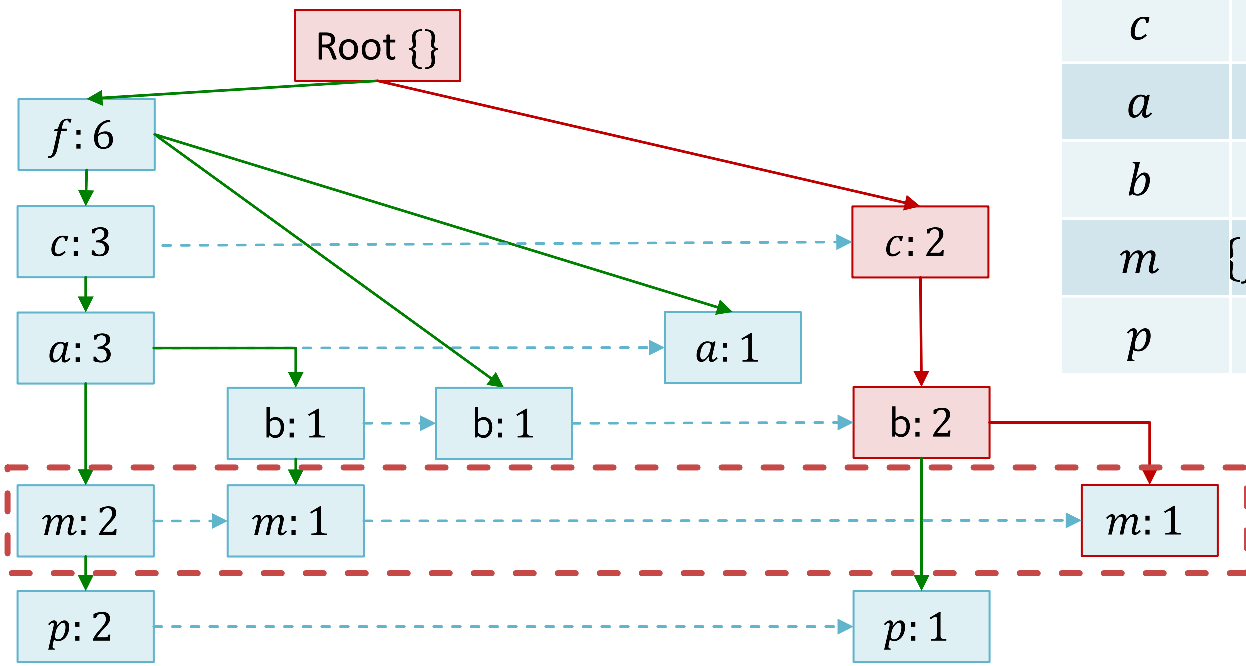


Items	Conditional Data Base
f	
c	
a	
b	
m	$\{f, c, a: 2\}, \{f, c, a, b: 1\}$
p	$\{f, c, a, m: 2\}, \{c, b: 1\}$

Example: Frequent Pattern Growth Algorithm

- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).

FP-Tree

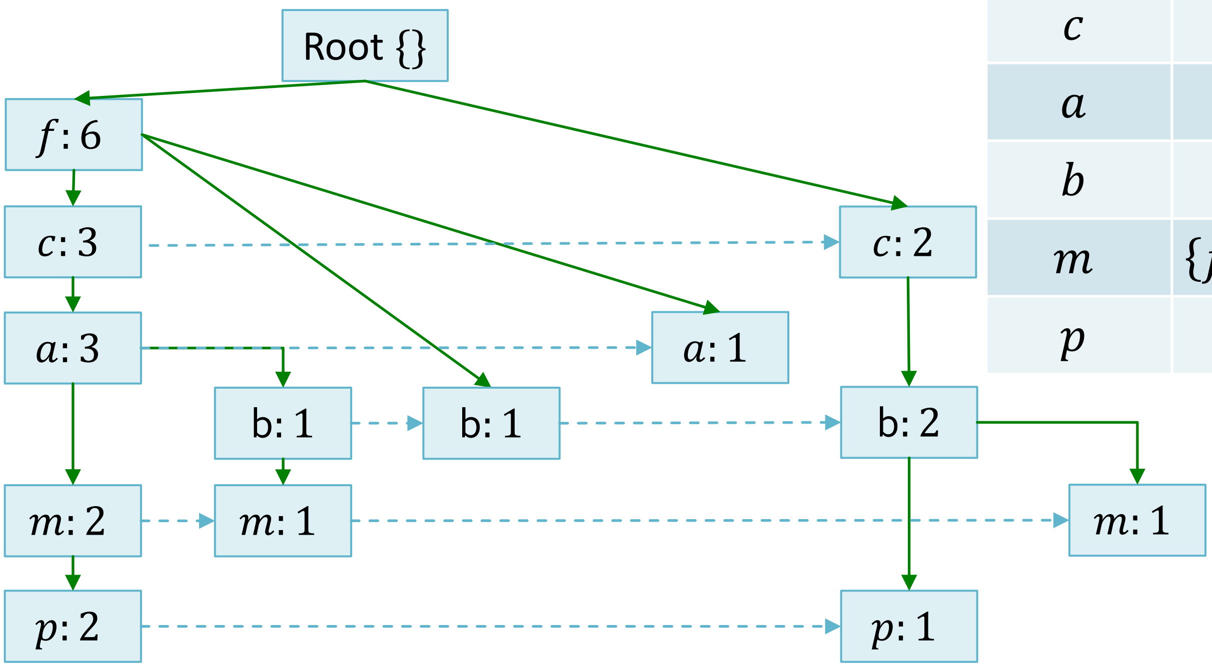


Items	Conditional Data Base
f	
c	
a	
b	
m	$\{f, c, a: 2\}, \{f, c, a, b: 1\}, \{c, b: 1\}$
p	$\{f, c, a, m: 2\}, \{c, b: 1\}$

Example: Frequent Pattern Growth Algorithm

- Step 4.a: Construct Conditional Data Base for each frequent item. (Mining FP-Tree)
  - Finding all prefix paths to the node (i.e., item).

FP-Tree



Items	Conditional Data Base
f	$\{\}$
c	$\{f: 3\}$
a	$\{f, c: 3\}, \{f: 1\}$
b	$\{f, c, a: 1\}, \{f: 1\}, \{c: 2\}$
m	$\{f, c, a: 2\}, \{f, c, a, , b: 1\}, \{c, b: 1\}$
p	$\{f, c, a, m: 2\}, \{c, b: 1\}$

Example: Frequent Pattern Growth Algorithm

- Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns
  - Find frequent items from each {X}'s conditional DB. Each frequent item i and X will form a new frequent pattern {i, X}
  - Respective conditional FP-tree is constructed.

Items	Conditional Data Base
f	$\{\}$
c	$\{f: 3\}$
a	$\{f, c: 3\}, \{f: 1\}$
b	$\{f, c, a: 1\}, \{f: 1\}, \{c: 2\}$
m	$\{f, c, a: 2\}, \{f, c, a, , b: 1\}, \{c, b: 1\}$
p	$\{f, c, a, m: 2\}, \{c, b: 1\}$

Item	#
$\cancel{f}$	$\cancel{2}$
c	3
$\cancel{a}$	$\cancel{2}$
$\cancel{b}$	$\cancel{1}$
$\cancel{m}$	$\cancel{2}$

generate $\{c, p\}$

$\{\boldsymbol{p}\}'\boldsymbol{s}$

Conditional FP-Tree

(single path)

Root $\{\}|p$

$c: 3$

Frequent patterns contain p:

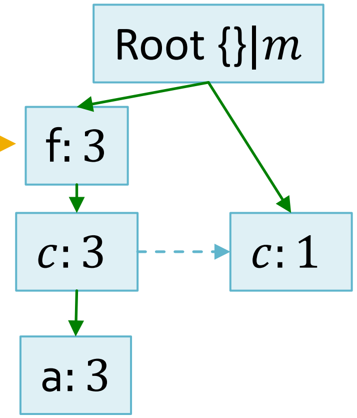
$\{\{p\}, \{c, p\}\}$

Example: Frequent Pattern Growth Algorithm

- Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns
  - Find frequent items from each {X}'s conditional DB. Each frequent item i and X will form a new frequent pattern {i, X}
  - Respective conditional FP-tree is constructed.

Items	Conditional Data Base
f	$\{\}$
c	$\{f{:}3\}$
a	$\{f,c{:}3\},\{f{:}1\}$
b	$\{f,c,a{:}1\},\{f{:}1\},\{c{:}2\}$
m	$\{f,c,a{:}2\},\{f,c,a,,b{:}1\},\{c,b{:}1\}$
p	$\{f,c,a,m{:}2\},\{c,b{:}1\}$

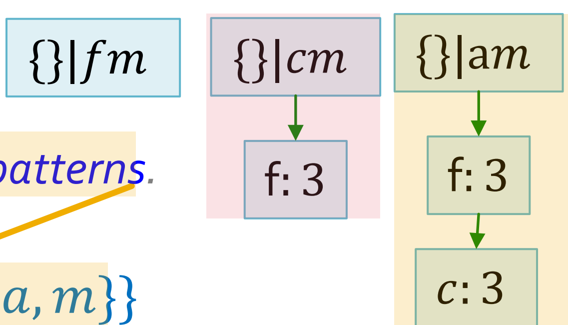
Conditional FP-Tree
(multiple paths)



Conditional Database 2

Items	Cond DB
mf	$\{\}$
mc	$\{f{:}3\}$
ma	$\{f,c{:}3\}$

Conditional FP-Tree 2



If the conditional FP-tree contains a single path, simply enumerate all patterns.

Frequent patterns contain m but not p:

$\{\{m\},\{f,m\},\{c,m\},\{f,c,m\},\{a,m\},\{f,c,a,m\},\{c,a,m\},\{f,a,m\}\}$

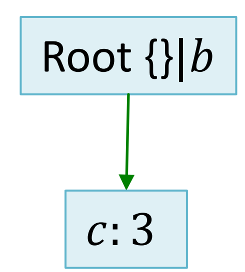
Example: Frequent Pattern Growth Algorithm

- Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns
  - Find frequent items from each {X}'s conditional DB. Each frequent item i and X will form a new frequent pattern {i, X}
  - Respective conditional FP-tree is constructed.

Items	Conditional Data Base
f	$\{\}$
c	$\{f: 3\}$
a	$\{f, c: 3\}, \{f: 1\}$
b	$\{f, c, a: 1\}, \{f: 1\}, \{c: 2\}$
m	$\{f, c, a: 2\}, \{f, c, a, , b: 1\}, \{c, b: 1\}$
p	$\{f, c, a, m: 2\}, \{c, b: 1\}$

Conditional FP-Tree

(single path)



Frequent patterns contain b but not m or p:

$\{\{b\}, \{c, b\}\}$

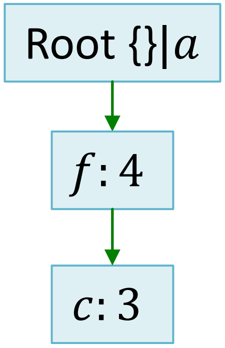
Example: Frequent Pattern Growth Algorithm

- Step 4.b: Generate Frequent Patterns and construct Conditional FP-tree for mining (longer) frequent patterns
  - Find frequent items from each {X}'s conditional DB. Each frequent item i and X will form a new frequent pattern {i, X}
  - Respective conditional FP-tree is constructed.

Items	Conditional Data Base
f	$\{\}$
c	$\{f:3\}$
a	$\{f,c:3\},\{f:1\}$
b	$\{f,c,a:1\},\{f:1\},\{c:2\}$
m	$\{f,c,a:2\},\{f,c,a,,b:1\},\{c,b:1\}$
p	$\{f,c,a,m:2\},\{c,b:1\}$

Conditional FP-Tree

(single path)



If the conditional FP-tree contains a single path, simply enumerate all patterns.

Frequent patterns contain a but not m $or\ p\ or\ b$:

$\{\{a\},\{f,a\},\{c,a\},\{f,c,a\}\}$

Example: Frequent Pattern Growth Algorithm

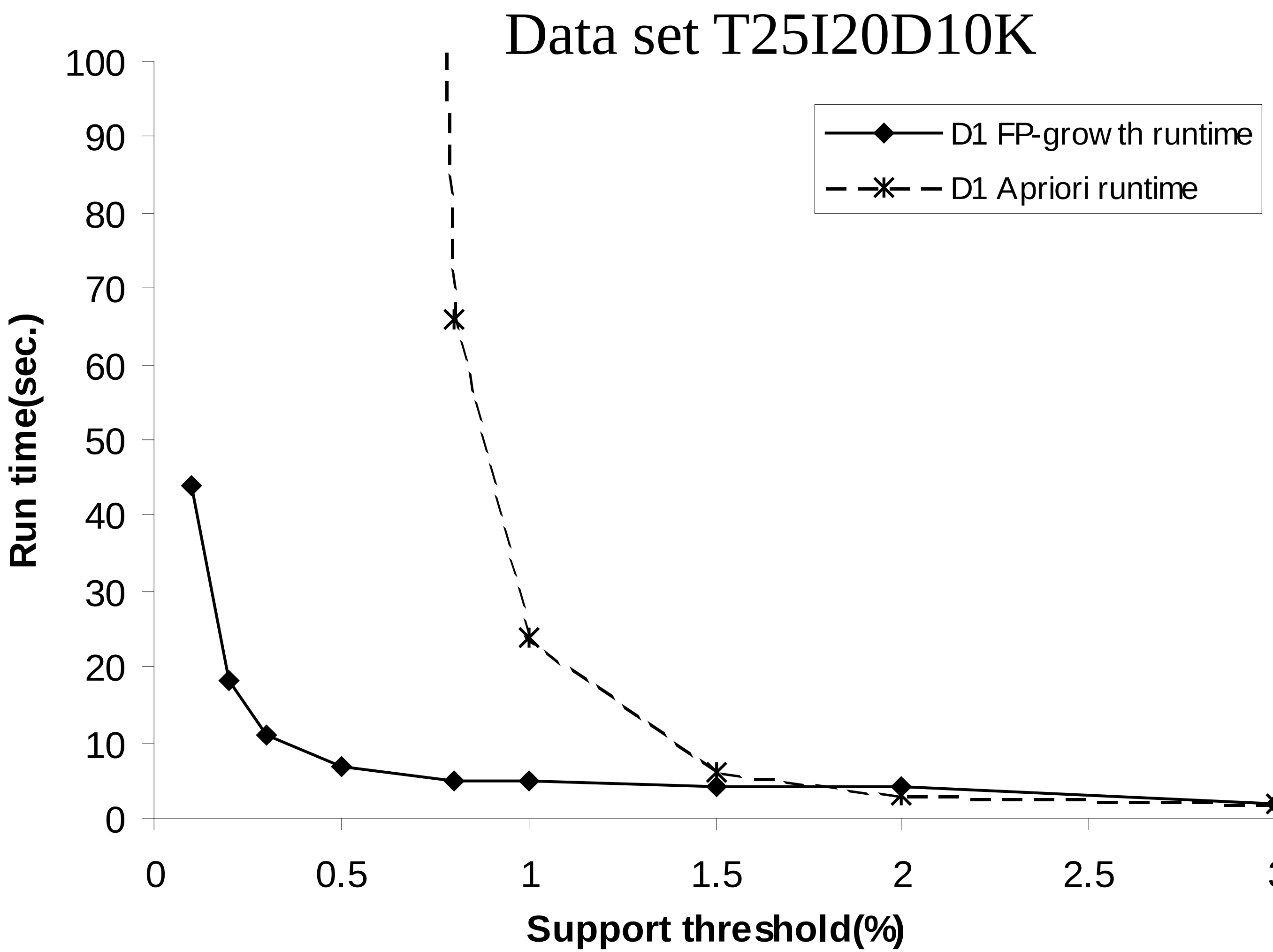
- Final results

Frequent Pattern Subsets	Frequent Patterns
Patterns contain p	$\{\{p\},\{c,p\}\}$
Patterns contain m but not p	$\{\{m\},\{f,m\},\{c,m\},\{f,c,m\},$ $\{a,m\},\{f,c,a,m\},\{c,a,m\},\{f,a,m\}\}$
Patterns contain b but not p or m	$\{\{b\},\{c,b\}\}$
Patterns contain a but not p or m or b	$\{\{a\},\{f,a\},\{c,a\},\{f,c,a\}\}$
Patterns contain c but not p,m,b,a	$\{\{c\},\{f,c\}\}$
Patterns contain f but not p,m,b,a,c	$\{\{f\}\}$

Summary of Mining Frequent Patterns Using FP-tree

- General idea (divide-and-conquer)
  - Recursively grow frequent patterns using FP-tree
- Frequent patterns can be partitioned into subsets according to L-order
  - L-order= $f: 6, c: 5, a: 4, b: 4, m: 4, p: 3$
  - Patterns containing p
  - Patterns having m but no p
  - Patterns having b but no m or p
  - …
  - Patterns having c but no a nor b, m, p
  - Pattern f

FP-growth vs. Apriori: Scalability With the Support Threshold



Why Is Frequent Pattern Growth Fast?

- Performance study shows
  - FP-growth can be an order of magnitude faster than Apriori
- Reasons
  - No candidate generation, no candidate test
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operations are counting and FP-tree building
- Challenges
  - When FP-tree cannot fit in memory

Outline

- A different frequent itemset/pattern mining algorithm
- Sequential pattern mining

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

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

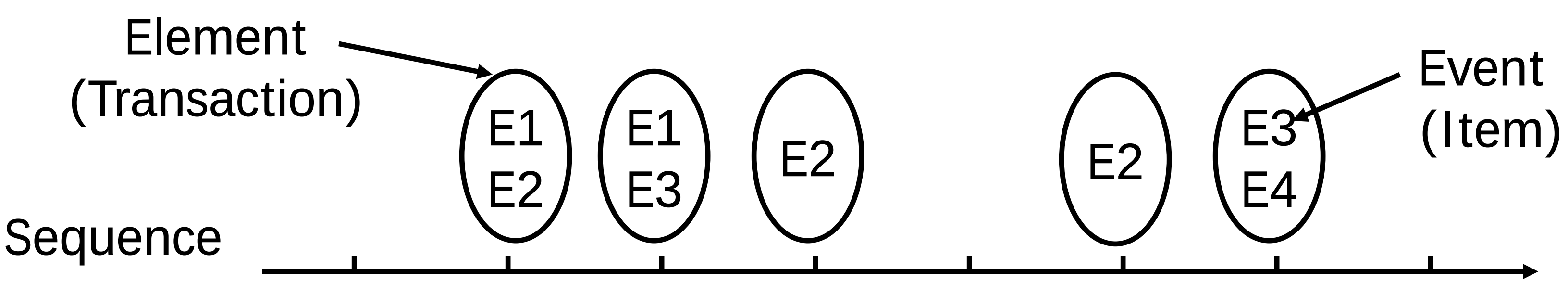
    (Shoes) (Racket, Racketball) --> (Sports_Jacket)

- Detecting Erroneous Sentences using Automatically Mined Sequential Patterns [1]
  - <the, more, the, JJ (base form of adjective)> in erroneous sentences

[1] *Guihua Sun, Xiaohua Liu, Gao Cong, Ming Zhou, Zhongyang Xiong, John Lee, Chin-Yew Lin: Detecting Erroneous Sentences using Automatically Mined Sequential Patterns. ACL 2007*

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



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

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

  - Each element contains a collection of events (items)

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

- 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 (or number) of data sequences that contain w. *We use fraction in the rest slides*
- 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 = 3

Examples of Frequent Subsequences:

< {1,2} >	s=3
< {2,3} >	s=3
< {2,4}>	s=4
< {3} {5}>	s=4
< {1} {2} >	s=4
< {2} {2} >	s=3
< {1} {2,3} >	s=3
< {2} {2,3} >	s=3
< {1,2} {2,3} >	s=3

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:

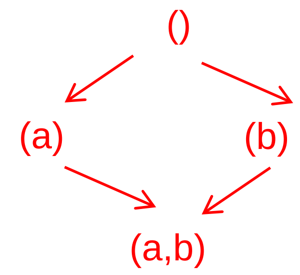
  $<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \ldots,$

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

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

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

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 (Apriori)**:
    - 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

GSP: Apriori-Based Sequential Pattern Mining

- Initial candidates: All 8-singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan DB once, count support for each candidate
- Base case (k=2): Generate candidate 2 -subsequences

SID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

minsup = 2

Cand.	sup
<a>	3
<b>	5
<c>	4
<d>	3
<e>	3
<f>	2
~~<g>~~	1
~~<h>~~	1

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<{a}{a}>	<ab>	<ac>	<ad>	<ae>	<af>
<b>	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
<c>	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
<d>	<da>	<db>	<dc>	<dd>	<de>	<df>
<e>	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
<f>	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>

Due to the space limit, we do not show {} in the above table

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<(ab)>	<(ac)>	<(ad)>	<(ae)>	<(af)>
<b>			<(bc)>	<(bd)>	<(be)>	<(bf)>
<c>				<(cd)>	<(ce)>	<(cf)>
<d>					<(de)>	<(df)>
<e>						<(ef)>
<f>						

- Without Apriori pruning: (8 singletons) 8*8+8*7/2 = 92 candidates
- With pruning, candidates: 36 + 15= 51

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** 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}>

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} >

Constraints on sequence patterns

- We can impose different constraints for two items in a sequence pattern. For example
  - a max-gap for two items
  - A min-gap for two items (e.g., contiguous)
- Need to check these constraints when deciding if a pattern is contained by a data sequence
- One solution: Mine sequential patterns without constraints
  - Postprocess the discovered patterns

Summary

- FP-tree algorithms
  - Understand the algorithm
- Sequential pattern mining
  - Understand the algorithm