西安交通大學
XI'AN JIAOTONG UNIVERSITY

数据挖掘
第四章：序列模式分析

刘均

陕西省天地网技术重点实验室

西安交通大学计算机学院

本章内容

4.1 序列模式的基本概念

4.2 GSP算法

4.3 PrefixSpan算法

基本要求：掌握序列模式的基本概念，掌握GSP、PrefixSpan两种典型序列模式挖掘算法

1 序列模式的基本概念

2 GSP算法

3 PrefixSpan算法

4.1 序列模式的基本概念

序列模式(sequential pattern)的概念最早是由Agrawal和Srikant 提出的，**序列模式分析**旨在寻找事件间在顺序上的相关性。

例子:

- ✓ 凡是买了喷墨打印机的顾客中，80%的人在三个月之后又买了墨盒 。
- ✓ 两年前购买了Ford牌轿车的顾客，很可能在今年采取贴旧换新旧的购车行动。
- ✓ 购买了自行车的客户中，70%的客户会在两个月后购买打气筒。

典型应用

- Web 站点访问者访问的 Web 页面序列：

  ⟨{主页}{电子产品}{照相机和摄像机}{数码像机}{购物车}{订购确认}{返回购物}⟩
- 计算机科学主修课程序列：

  ⟨{算法与数据结构，操作系统引论}{数据库系统，计算机体系结构}{计算机网络，软

4.1 序列模式的基本概念

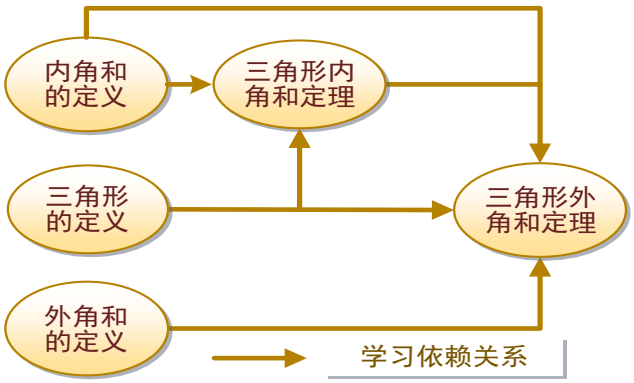
- **序列模式(sequential pattern)**的概念最早是由Agrawal和Srikant 提出的，**序列模式分析**旨在寻找事件间在顺序上的相关性。

例子:

- ✓ 凡是买了喷墨打印机的顾客中，80%的人在三个月之后又买了墨盒 。
- ✓ 两年前购买了Ford牌轿车的顾客，很可能在今年采取贴旧换新旧的购车行动。
- ✓ 购买了自行车的客户中，70%的客户会在两个月后购买打气筒。

典型应用

- ✓ E-learning中



4.1 序列模式的基本概念

A series of daily news paper articles

typhoon

flood, landslide

typhoon

flood, landslide

<typhoon (flood, landslide)>

事件中的例子

4.1 序列模式的基本概念

交易数据数据库

Transaction Time	Customer	Items Bought
June 20, 1994 10:13 am	J. Brown	Juice, Coke
June 20, 1994 11:02 am	F. Zappa	Brandy
June 20, 1994 11:47 am	J. Brown	Beer
June 20, 1994 2:32 pm	B. Moore	Beer
June 21, 1994 9:22 am	J. Brown	Wine, Water, Cider
June 21, 1994 3:19 pm	J. Mitchell	Beer, Gin, Cider
June 21, 1994 5:27 pm	B. Adams	Beer
June 21, 1994 6:17 pm	B. Moore	Wine, Cider
June 22, 1994 10:34 am	B. Adams	Brandy
June 22, 1994 5:03 pm	B. Moore	Brandy

4.1 序列模式的基本概念

交易数据数据库（按顾客与时间排序）

Customer	Transaction Time	Items Bought
B. Adams	June 21, 1994 5:27 pm	Beer
B. Adams	June 22, 1994 10:34 am	Brandy
J. Brown	June 20, 1994 10:13 am	Juice, Coke
J. Brown	June 20, 1994 11:47 am	Beer
J. Brown	June 21, 1994 9:22 am	Wine, Water, Cider
J. Mitchell	June 21, 1994 3:19 pm	Beer, Gin, Cider
B. Moore	June 20, 1994 2:32 pm	Beer
B. Moore	June 21, 1994 6:17 pm	Wine, Cider
B. Moore	June 22, 1994 5:03 pm	Brandy
F. Zappa	June 20, 1994 11:02 am	Brandy

4.1 序列模式的基本概念

序列数据库

Customer	Customer Sequence
B. Adams	(Beer) (Brandy)
J. Brown	(Juice, Coke) (Beer) (Wine, Water, Cider)
J. Mitchell	(Beer, Gin, Cider)
B. Moore	(Beer) (Wine, Cider) (Brandy)
F. Zappa	(Brandy)

序列模式

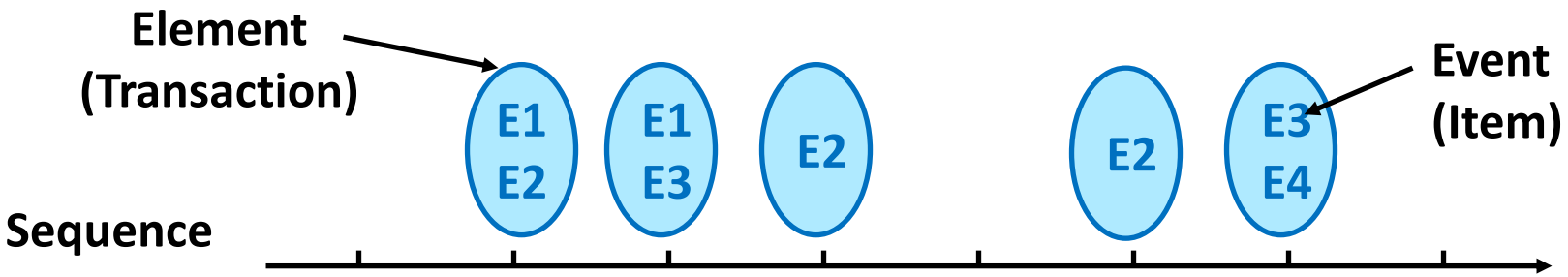
Sequential Patterns with Support > 40%	Customers Supporting it
(Beer) (Brandy)	B. Adams, B. Moore
(Beer) (Wine, Cider)	J. Brown, B. Moore

4.1 序列模式的基本概念

- **序列模式**：给定一个由不同**序列**组成的集合，其中每个**序列**由不同的**元素**按顺序有序排列，每个**元素**由不同**项目**组成，同时给定一个用户指定的**最小支持度阈值**，序列模式就是**频繁子序列**，即该子序列在序列集中的出现频率不低于最小支持度阈值。

4.1 序列模式的基本概念

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
Event data	History of events generated by a given sensor	Events triggered by a sensor at time t	Types of alarms generated by sensors



4.1 序列模式的基本概念

形式化表示：

- ✓ 项目集(Itemset)是各种项目组成的集合。
- ✓ 序列(Sequence)是不同项目集的有序排列，序列s可以表示为$s = <s_1 s_2 \ldots s_l>$，$s_j(1 <= j <= l)$为项目集，也称为序列s的元素。
- ✓ 序列的元素(Element)可表示为$(x_1, x_2, \ldots, x_m)$，$x_k(1 <= k <= m)$为不同的项目，如果一个序列只有一个项目，则括号可以省略。同一元素中的项目间排列没有顺序，为了表达的唯一性，同一个元素内部的项目按字典序排列。
- ✓ 一个序列包含的所有项目的个数称为序列的长度。长度为l的序列记为l-序列。

4.1 序列模式的基本概念

1-序列：$<i_1>, <i_2>, \cdots, <i_n>$

2-序列：$<\{i_1, i_2\}>, <\{i_1, i_3\}>, \cdots, <\{i_{n-1}, i_n\}>, <\{i_1\}\{i_1\}>, <\{i_1\}\{i_2\}>, \cdots, <\{i_{n-1}\}\{i_n\}>$

3-序列：$<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \cdots, <\{i_1, i_2\}\{i_1\}>, \cdots,$

$<\{i_1\}\{i_1, i_2\}>, \cdots, <\{i_1\}\{i_1\}\{i_1\}>, \cdots, <\{i_n\}\{i_n\}\{i_n\}>$

注意，候选序列的个数比候选项集的个数大得多。产生更多候选的原因有下面两个。

(1) 一个项在项集中最多出现一次，但一个事件可以在序列中出现多次。给定两个项 i_1 和 i_2，只能产生一个候选 2-项集$\{i_1, i_2\}$，但却可以产生许多候选 2-序列，如$\langle\{i_1, i_2\}\rangle$，$\langle\{i_1\}\{i_2\}\rangle$，$\langle\{i_2, i_1\}\rangle$和$\langle\{i_1, i_1\}\rangle$。

(2) 次序在序列中是重要的，但在项集中不重要。例如，{1, 2}和{2, 1}表示同一个项集，而$\langle\{i_1\}\{i_2\}\rangle$和$\langle\{i_2\}\{i_1\}\rangle$对应于不同的序列，因此必须分别产生。

4.1 序列模式的基本概念

形式化表示：

- 设$\alpha = <a_1 a_2 \ldots a_n>$，$\beta = <b_1 b_2 \ldots b_m>$，如果存在整数$1 <= j_1 < j_2 < \ldots < j_n <= m$，使得$a_1 \subseteq b_{j1}$，$a_2 \subseteq b_{j2}$，…，$a_n \subseteq b_{jn}$，则称序列$\alpha$为序列$\beta$的子序列（$\beta$为$\alpha$的超序列），又称序列$\beta$包含序列$\alpha$，记为$\alpha \subseteq \beta$。

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

4.1 序列模式的基本概念

- 给定一个n-序列，其中包含了多少个k-子序列?

<{a b} {c d e} {f} {g h i}> $n = 9$

$k=4$: Y _ _ Y Y _ _ _ Y

<{a} {d e} {i}>

西安交通大學 XI'AN JIAOTONG UNIVERSITY

4.1 序列模式的基本概念

- **形式化表示：**
  - ✓ 序列α在序列数据库S中的**支持度**为序列数据库S中包含序列α的序列个数，记为Support(α)。
  - ✓ 给定支持度阈值ξ，如果序列α在序列数据库中的支持度不低于ξ，则称序列α为**序列模式**。
  - ✓ 长度为l的序列模式记为**l-模式**。

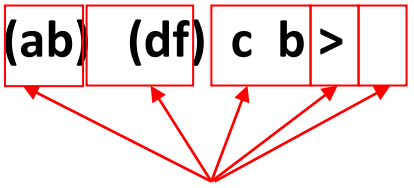
4.1 序列模式的基本概念

- **序列模式挖掘：**给定一个序列集，找出其中的所有频繁子序列

序列数据库

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

序列： < (ef) (ab) (df) c b >



一个元素是一个项集，元素中的项目是无序的，可按字母序排列。

设支持度阈值min_sup =2, <(ab)c> 是序列模式

4.1 序列模式的基本概念

以Customer_Id 及 TransactionTime 排序

Customer Id	TransactionTime	Items Bought
1	June 25 '03	30
1	June 30 '03	90
2	June 10 '03	10, 20
2	June 15 '03	30
2	June 20 '03	40, 60, 70
3	June 25 '03	30, 50, 70
4	June 25 '03	30
4	June 30 '03	40, 70
4	July 25 '03	90
5	June 12 '03	90

Transaction

Item

Itemset

4.1 序列模式的基本概念

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)⟩

Sequence

<(30) (90)> is supported by customer 1 and 4

<30 (40 70)> is supported by customer 2 and 4

4.1 序列模式的基本概念

- **序列模式的限制：**
  - ✓ 时间限制
    - 相邻事件之间最大与/或最小的时间间隔
    - 例如: 购买‘Foundation’, 然后购买‘Foundation and Empire’ 与‘Ringworld’ 应在三个月之内.
  - ✓ 分类体系

4.1 序列模式的基本概念

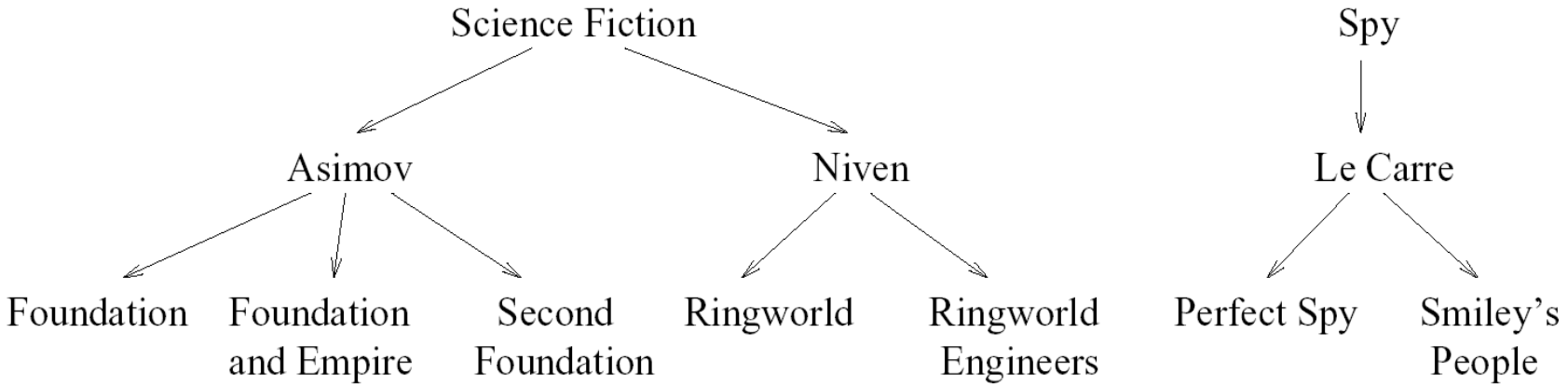


Figure 1: Example of a Taxonomy

Example: A customer who bought Foundation,then Perfect Spy, would support the following patterns:

- **Foundation, then Perfect Spy**
- **Asimov, then Perfect Spy**
- **Science Fiction, then Le Carre**

4.1 序列模式的基本概念

序列模式挖掘问题描述

✓ 输入

– 对于序列数据库 D：

- $I=\{i_1, i_2,\ldots,i_n\}$ 是所有项目的集合
- 每个序列都是按时间排列的一组交易
- 每个交易包含以下字段:sequence-id, transaction-id, transaction-time and a set of items.

✓ 问题

– 找到满足最小支持度的所有序列模式

4.1 序列模式的基本概念

- **主要算法**
  - ✓ 类Apriori算法
    - GSP(Generalized Sequential Patterns)：Srikant & Agrawal [EDBT'96]
    - SPADE：Zaki [Machine Leanining'00]
  - ✓ 基于模式增长（Pattern-Growth-based）的算法
    - PrefixSpan & FreeSpan：Han et al.KDD'00; Pei, et al. [ICDE'01]

本章内容

4.1 序列模式的基本概念

4.2 GSP算法

4.3 PrefixSpan算法

4.2 GSP算法

- **Apriori 性质**
  - ✓ 如果序列 s 是非频繁的, 则s 的所有超序列都是非频繁的
  - ✓ <hb>是非频繁的，则<hab> 与<(ah)b>都是非频繁的

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

设支持度阈值min_sup =2

4.2 GSP算法

GSP算法描述

- ✓ 扫描序列数据库，得到长度为1的序列模式L1，作为初始的种子集。
- ✓ 根据长度为i的种子集L_i，通过连接操作和剪切操作生成长度为$i+1$的候选序列模式C_{i+1}；然后扫描序列数据库，计算每个候选序列模式的支持数，产生长度为$i+1$的序列模式L_{i+1}，并将L_{i+1}作为新的种子集。
- ✓ 重复第二步，直到没有新的序列模式或新的候选序列模式产生为止

$$L_1 \Rightarrow C_2 \Rightarrow L_2 \Rightarrow C_3 \Rightarrow L_3 \Rightarrow C_4 \Rightarrow L_4 \Rightarrow \cdots\cdots$$

4.2 GSP算法

候选序列模式步骤

- ✓ **连接阶段**：如果去掉序列模式s_1的第一个项目与去掉序列模式s_2的最后一个项目所得到的序列相同，则可以将s_1与s_2进行连接，即将s_2的最后一个项目添加到s_1中。
- ✓ **剪切阶段**：若某候选序列模式的某个子序列不是序列模式，则此候选序列模式不可能是序列模式，将它从候选序列模式中删除。

$$L_1 \Rightarrow C_2 \Rightarrow L_2 \Rightarrow C_3 \Rightarrow L_3 \Rightarrow C_4 \Rightarrow L_4 \Rightarrow \cdots\cdots$$

4.2 GSP算法

- **GSP算法实现**
  - ✓ **生成候选序列模式：生成尽可能少候选模式，同时保证结果完整性**
  - ✓ **计算候选序列模式的支持度：找出序列中元素出现的位置**
  - ✓ **实现分类体系**

4.2 GSP算法

- **生成候选序列模式**
  - ✓ **目标: 给定所有的 (*k*-1)-序列, 生成所有的候选*k*-序列模式**
  - ✓ **算法:**
    - **连接阶段**: L_{k-1} 相互连接。S_1 能够连接S2 ， 当 (S1 – first item) 与 (S2 – last item)相同
    - **剪切阶段**: 删除包含“不满足最小支持度要求的 (k-1) 子序列”的候选序列模式。

西安交通大學 XI'AN JIAOTONG UNIVERSITY

4.2 GSP算法

- **例子：从长度为3的序列模式产生长度为4的获选序列模式。**

Sequential patterns With length 3	Candidate 4-Sequences	
	After Join	After Pruning
<(1,2) 3>	<(1,2) (3,4)>	<(1,2) (3,4)>
<(1,2) 4>	<(1,2) 3 5>	
<1 (3,4)>		
<(1,3) 5>		
<2 (3,4)>		
<2 3 5>		

4.2 GSP算法

计算候选序列模式的支持度

✓ **关键问题: 找出交易数据库中包含的候选序列模式**

- Let ***d*** be a data-sequence, and let ***s = (s_1,..s_n)*** be a candidate sequence. We assume the existence of a procedure that finds the first occurrence of an element of s in **d** after a given time
- For any element s_i, the procedure always checks whether a later set of transactions contains s_i

s1	...	s2	...	...	s3	...

4.2 GSP算法

寻找单个元素

- ✓ **目的:** 找到元素（ element ）的第一个出现的位置
- ✓ 将交易数据库转化为交易链，每个链用一个项目（Item）标示
  - **从水平到垂直**

Transaction-Time	Items
10	1, 2
25	4, 6
45	3
50	1, 2
65	3
90	2, 4
95	6

Item	Times
1	→ 10 → 50 → NULL
2	→ 10 → 50 → 90 → NULL
3	→ 45 → 65 → NULL
4	→ 25 → 90 → NULL
5	→ NULL
6	→ 25 → 95 → NULL
7	→ NULL

4.2 GSP算法

寻找单个元素

例子: 假设窗口大小设置为7天，找出时间t=20 后，元素 (2, 6) 的第一个出现位置

(2) → 50, (6) → 25,
50 – 25 > 7, t=43(50-7)

(2) → 50, (6) → 95,
95 –50 > 7, t=88

(2) → 90, (6) → 95,
95 – 90 <=7

t=20

Transaction-Time	Items
10	1, 2
25	4, 6
45	3
50	1, 2
65	3
90	2, 4
95	6

Figure 4: Example Data-Sequence

4.2 GSP算法

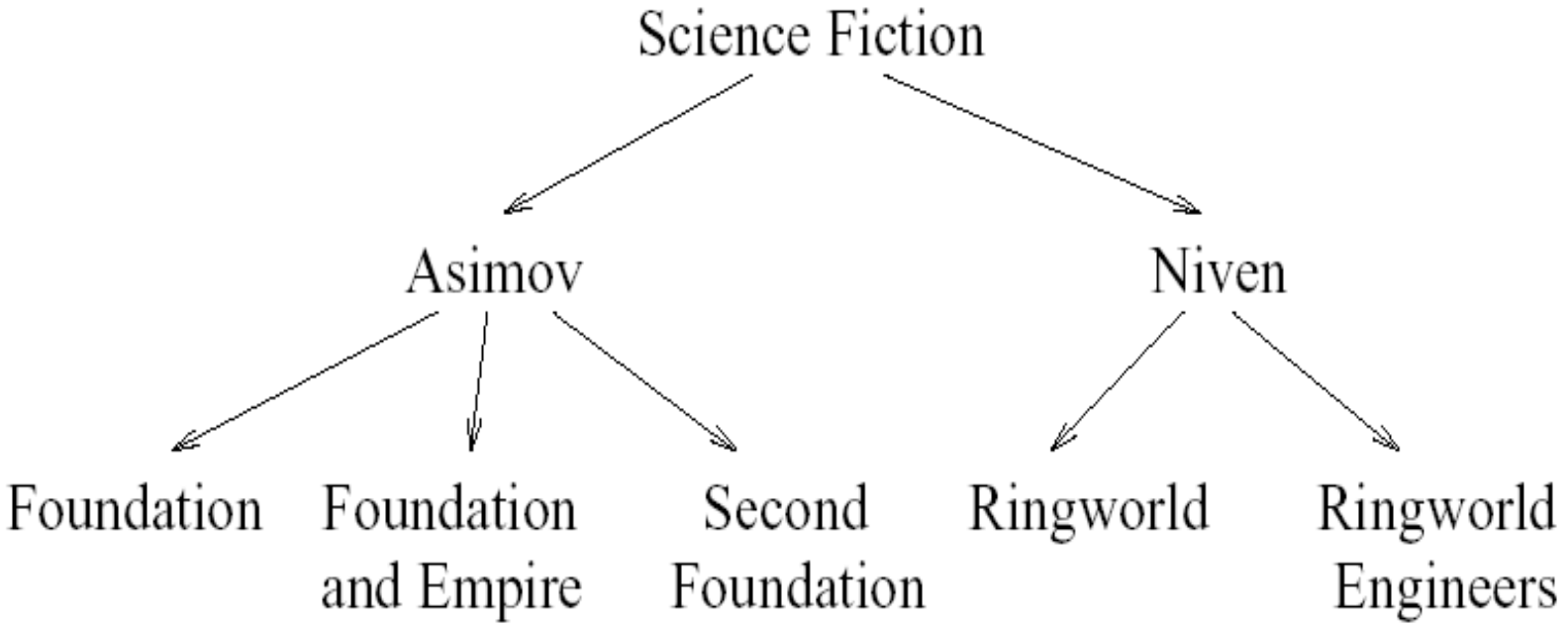
实现分类体系

- ✓ **基本思路:** 用扩展序列d'代替原序列d，扩展序列d'中除包含了原序列中所有项目外，还有其祖先
- ✓ **例子:**

- d1: <(Foundation, Ringworld)(Second Foundation)>
- d2: <(Foundation, Ringworld, Asimov, Niven, Science Fiction)(Second Foundation, Asimov, Science Fiction)>

4.2 GSP算法

- **实现分类体系**



西安交通大學 XI'AN JIAOTONG UNIVERSITY

4.2 GSP算法

- **例子 – 长度为1的序列模式**
  - ✓ **初始候选序列**

    <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
  - ✓ **扫描数据库，计算候选序列的支持度**

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

设支持度阈值min_sup =2

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

4.2 GSP算法

例子 – 生成长度为2的候选序列

- ✓ 得到51个长度为2的候选序列
- ✓ 不考虑Apriori性质将产生8*8+8*7/2=92个候选序列

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<aa>	<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>

	<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>						

- ✓ 扫描数据库，计算长度为2的 候选序列的支持度，得到19个计算长度为2 序列模式

4.2 GSP算法

例子

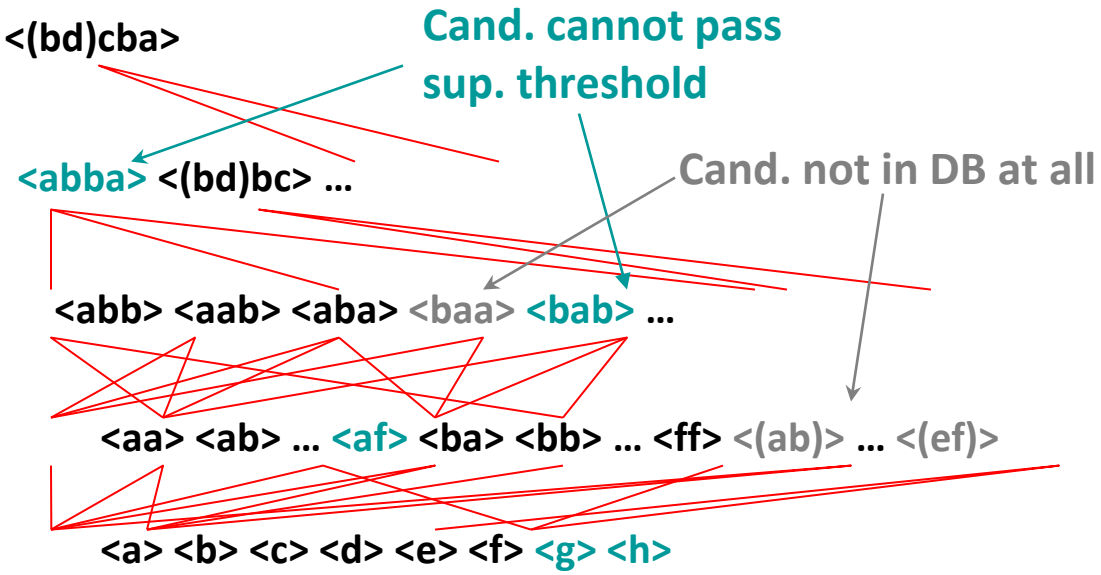
5th scan: 1 cand. 1 length-5 seq. pat.

4th scan: 8 cand. 6 length-4 seq. pat.

3rd scan: 46 cand. 19 length-3 seq. pat. 20 cand. not in DB at all

2nd scan: 51 cand. 19 length-2 seq. pat. 10 cand. not in DB at all

1st scan: 8 cand. 6 length-1 seq. pat.



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

4.2 GSP算法

GSP算法的缺点

- ✓ **需要生成大规模候选序列模式**
  - 1,000 个length-1 序列模式可生成 $1000 \times 1000 + \frac{1000 \times 999}{2} = 1{,}499{,}500$ length-2 候选序列模式!
- ✓ **挖掘过程中需要多次扫描数据库**
- ✓ **挑战: 挖掘长的序列模式**
  - 指数函数关系
  - 一个length-100序列模式需要测试10^{30}个候选序列模式!

本章内容

4.1 序列模式的基本概念

4.2 GSP算法

4.3 PrefixSpan算法

4.2 PrefixSpan算法

Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach

采用分治思想，不断产生序列数据库的多个更小的投影数据库，然后在各个投影数据库上进行序列模式挖掘

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, Meichun Hsu: **Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach**. *IEEE Trans. Knowl. Data Eng.* 16(11): 1424-1440 (2004)

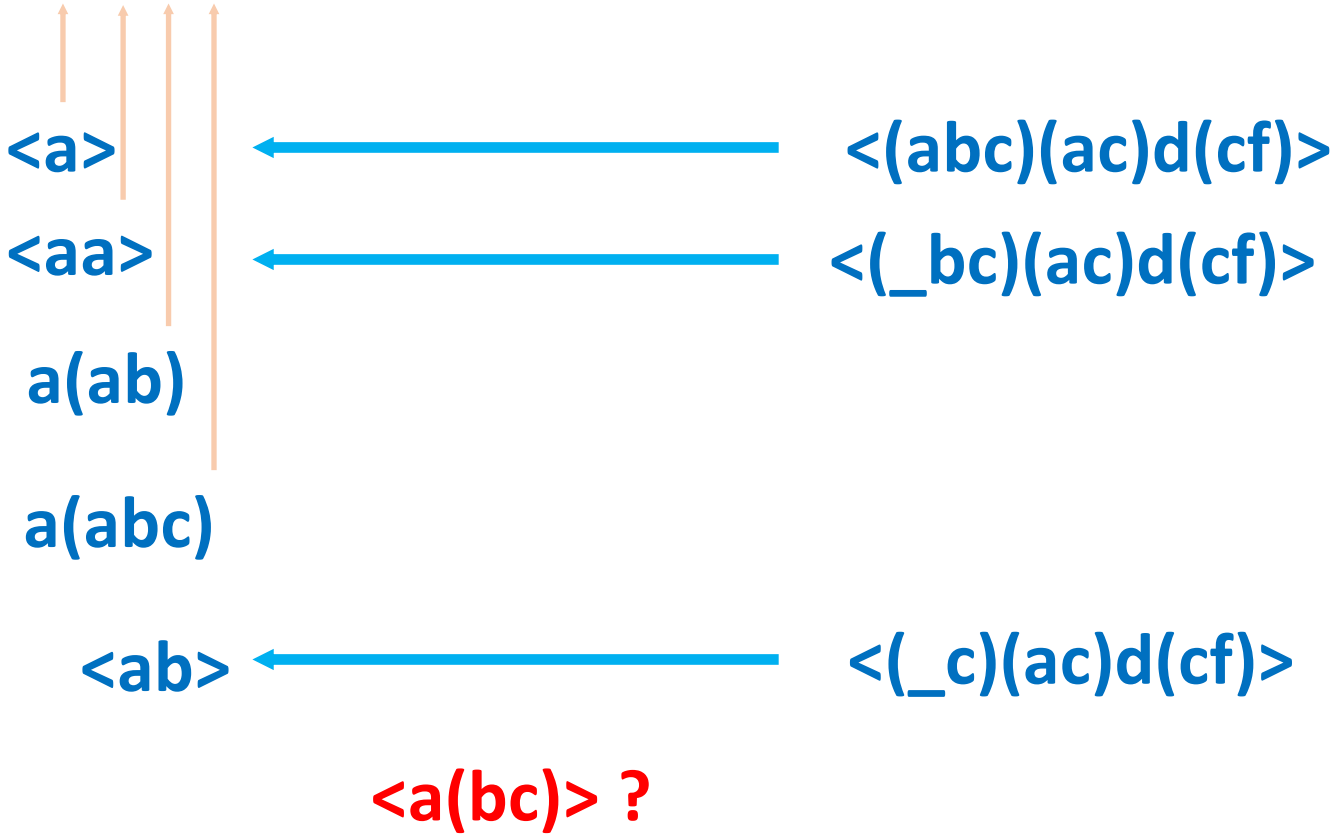
4.2 PrefixSpan算法

相关定义

- **前缀**：设每个元素中的所有项目按照字典序排列。给定序列 $\alpha = <e_1 e_2 \ldots e_n>$，$\beta = <e_1' e_2' \ldots e_m'>$ $(m \le n)$，如果 $e_i' = e_i$ $(i \le m-1)$，$e_m' \subseteq e_m$，并且 $(e_m - e_m')$ 中的项目均在 e_m' 中项目的后面，则称 β 是 α 的前缀
- **投影**：给定序列 α 和 β，如果 β 是 α 的子序列，则 α 关于 β 的投影 α' 必须满足：β 是 α' 的前缀，α' 是 α 的满足上述条件的最大子序列 （如果 β 是 α 的前缀？）
- **后缀**： 序列 α 关于子序列 $\beta = <e_1 e_2 \ldots e_{m-1} e_m'>$ 的投影为 $\alpha' = <e_1 e_2 \ldots e_n>$ $(n >= m)$，则序列 α 关于子序列 β 的后缀为 $<e_m'' e_{m+1} \ldots e_n>$，其中 $e_m'' = (e_m - e_m')$

4.2 PrefixSpan算法

- 例子：<a(abc)(ac)d(cf)>



<a(bc)> ?

4.2 PrefixSpan算法

- **例子：前缀**

  α =<a(abc)(ac)d(cf)>

  β =<a(abc)a>

- **例子：投影**

  α =<a(abc)(ac)d(cf)>

  β =<(bc)a>

  α’ =<(bc)(ac)d(cf)>

- **例子：后缀**

  α’ =<a(abc)(ac)d(cf)>,

  β =<a(abc)a>,

  γ= <(_c)d(cf)>.

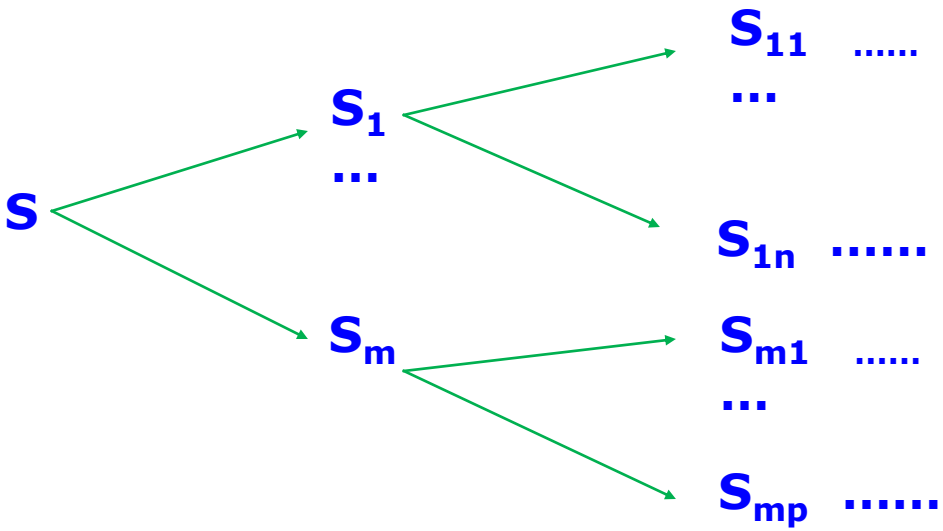
  α’ =<a(abc)(ac)d(cf)>,

  β =<ac>,

  γ= <(ac)d(cf)>.

4.2 PrefixSpan算法

- **算法描述(分治思想):**
  1. 扫描序列数据库，生成所有长度为1的序列模式
  2. 根据长度为1的序列模式，生成相应的投影数据库
  3. 在相应的投影数据库上重复上述步骤，直到在相应的投影数据库上不能产生长度为1的序列模式为止



西安交通大学 XI'AN JIAOTONG UNIVERSITY

4.2 PrefixSpan算法

■ 例子： min_sup=2； sequence database S

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

4.2 PrefixSpan算法

- 扫描序列数据库S，产生长度为1的序列模式有：

  <a> : 4, <b> : 4, <c> : 4, <d> : 3, <e> : 3, <f> : 3

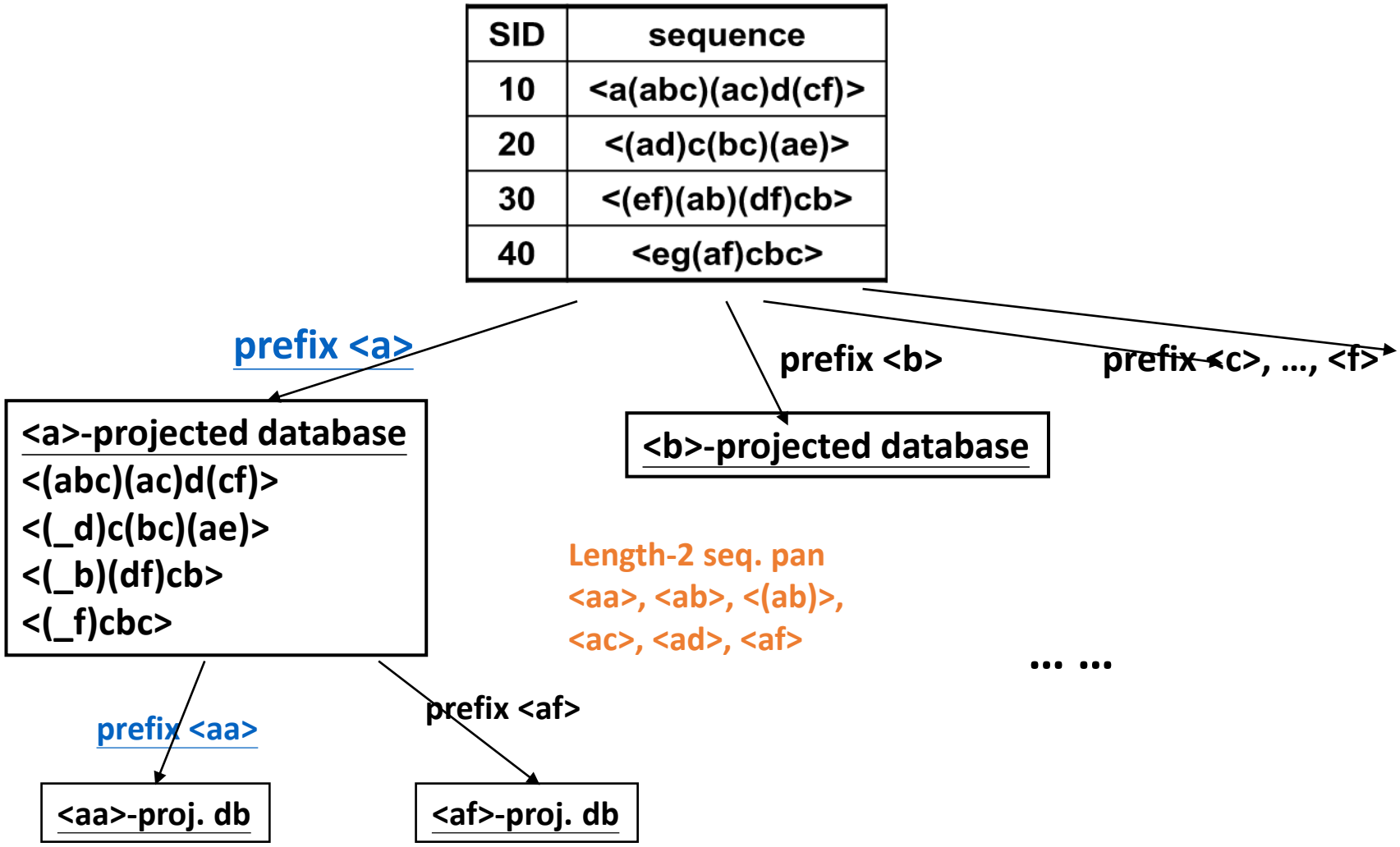
- 序列模式的全集必然可以分为以<a>, <b>, <c>, <d>, <e>和<f>为前缀的序列模式的集合，构造不同前缀所对应的投影数据库

- 分别对不同的投影数据库重复上述过程，直到没有新的长度为1的序列模式产生为止

<α> - 投影数据库：设 α 为序列数据库 S 中的一个序列模式，则 α 的投影数据库为 S中所有以 α 为前缀的序列相对于 α 的后缀

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

4.2 PrefixSpan算法



4.2 PrefixSpan算法

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

Step 2:

形成<a>-projected database,

包含4个postfix sequences:

< (abc) (ac)d (cf) >,

< (_d)c (bc) (ae) >,

< (_b) (df)cb >,

< (_f)cbc >

4.2 PrefixSpan算法

Prefix	Project Database
<a>	<(abc)(ac)d(cf)> <(_d)c(bc)(ae)> <(_b)(df)cb> <(_f)cbc>
<b>	
<c>	
<d>	
<e>	
<f>	

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

4.2 PrefixSpan算法

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

Prefix	Projected(postfix) database
<a>	<(abc) (ac)d (cf) >, < (_d)c (bc) (ae) >, < (_b) (df)cb >, < (_f)cbc >

Step 3:

1 扫描 <a>-projected database 一次, 找到所有具有<a> 前缀的length-2序列模式.

<aa>:2, <ab>:4, <(ab)>:?, <ac>:?, <ad>:?, <af>:?

2. 所有具有<a> 前缀的序列被划分为6个子集

(1) having prefix <aa>,

...

(6) having <af>

4.2 PrefixSpan算法

Prefix	Projected(postfix) database	Sequential patterns
<a>	<(abc) (ac)d (cf) >, < (_d)c (bc) (ae) >, < (_b) (df)cb >, < (_f)cbc >	<a>,<aa>,<ab>,<a(bc)>,<a(bc)a>,<aba>,<abc>,<(ab)>,<(ab)c>,<(ab)d>,<(ab)f>,<(ab)dc>,<ac>,<aca>,<acb>,<acc>,<ad>,<adc>,<af>

3. <aa>-projected database 仅包含两个以<aa>为前缀的非空后缀

   <(_bc) (ac)d (cf) > <(_e) >

   <aa>-projected database 终止

4.2 PrefixSpan算法

Prefix	Projected(postfix) database	Sequential patterns
<a>	<(abc) (ac)d (cf) >, < (_d)c (bc) (ae) >, < (_b) (df)cb >, < (_f)cbc >	<a>,<aa>,<ab>, <a(bc)>,<a(bc)a>,<aba>, <abc>,<(ab)>,<(ab)c>,<(ab)d>,<(ab)f>, <(ab)dc>,<ac>,<aca>,<acb>,<acc>,<ad>, <adc>,<af>

4. <ab>-projected database 包含三个以<ab>为前缀的非空后缀

<(_c)(ac)d(cf)>, <(_c)(ae)>, <c>

通过迭代挖掘， <ab>-projected database 返回四个序列模式：

<(_c)>,<(_c)a>,<a>, and <c>，即 <a(bc)>,<a(bc)a>,<aba>, and <abc>

4.2 PrefixSpan算法

Prefix	Projected(postfix) database	Sequential patterns
<a>	<(abc) (ac)d (cf) >, < (_d)c (bc) (ae) >, < (_b) (df)cb >, < (_f)cbc >	<a>,<aa>,<ab>, <a(bc)>,<a(bc)a>,<aba>, <abc>,<(ab)>, <(ab)c>,<(ab)d>,<(ab)f>, <(ab)dc>,<ac>,<aca>,<acb>,<acc>,<ad>, <adc>,<af>

5. <(ab)>-projected database 包含两个以<(ab)>为前缀的非空后缀

<(_c)(ac)d(cf)> 与 <(df)cb>

<(ab)>-projected database 返回四个序列模式：

<(ab)c>,<(ab)d>,<(ab)f>, <(ab)dc>

4.2 PrefixSpan算法

6. <ac>-, <ad>- and <af>- projected databases 同样以迭代方式挖掘

   同样地, 通过构建<b>-, <c>-,<d>-, <e>- and <f>-projected databases， 可以挖掘到以<b>, <c>, <d>, <e> and <f>为前缀的序列模式

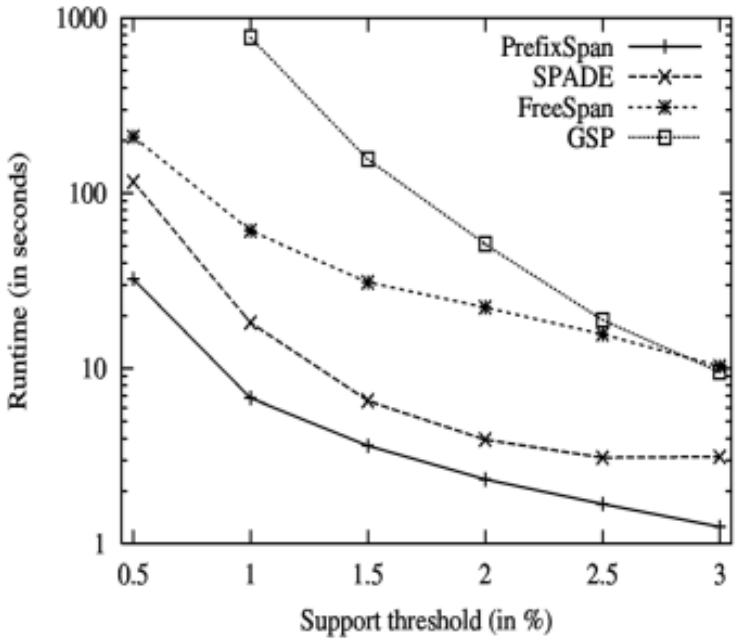
4.2 PrefixSpan算法

Prefix	Projected(postfix)database	Sequential patterns
<a>	<(abc)(ac)d(cf)>, <(_d)c(bc)(ae)>, <(_b) (df)cb>, <(_f)cbc>	<a>, <aa>, <ab>, <a(bc)>, <a(bc)a>, <aba>, <abc>, <(ab)>, <(ab)c>, <(ab)d>, <(ab)f>, <(ab)dc>, <ac>, <aca>, <acb>, <acc>, <ad>, <adc>, <af>
<b>	<(_c) (ac)d(cf)>, <(_c) (ae)>, <(df)cb>, <c>	<b>, <ba>, <bc>, <(bc)>, <(bc)a>, <bd>, <bdc>, <bf>
<c>	<(ac)d(cf)>, <(bc) (ae)>, <b>, <bc>	<c>, <ca>, <cb>, <cc>
<d>	<(cf)>, <c(bc) (ae)>, <(_f)cb>	<d>, <db>, <dc>, <dcb>
<e>	<(_f) (ab) (df)cb>, <(af)cbc>	<e>, <ea>, <eab>, <eac>, <eacb>, <eb>, <ebc>, <ec>, <ecb>, <ef>, <efb>, <efc>, <efcb>
<f>	<(ab) (df)cb>, <cbc>	<f>, <fb>, <fbc>, <fc>, <fcb>

4.2 PrefixSpan算法

- **分析:**
  - ✓ **无需要产生候选序列模式**

    => PrefixSpan 在较小的空间进行查找
  - ✓ **投影数据库不断缩小**
  - ✓ **PrefixSpan主要开销在于构建投影数据库**



总结

- A sequence database consists of sequences of ordered elements or events. Examples of sequence data include customer shopping sequences, Web clickstreams, and biological sequences.
- Sequential pattern mining is the mining of frequently occurring ordered events or subsequences as patterns. Given a sequence database, any sequence that satisfies minimum support is frequent and is called a sequential pattern.
- Algorithms for sequential pattern mining include GSP, SPADE, and PrefixSpan, as well as CloSpan.