

intro

- 数据挖掘: 从大量数据中发现有趣的模式
- 数据库技术的自然进化, 具有巨大需求和广泛应用
- **KDD** 过程包括数据清理, 数据集成, 数据选择, 变换, 数据挖掘, 模式评估, 和知识表示
- 挖掘可以在各种数据存储上进行
- 数据挖掘功能: 特征, 区分, 关联, 分类, 聚类, 孤立点 和趋势分析, 等.
- 数据挖掘的应用
- 数据挖掘的主要问题

data

- **Data attribute types: nominal, binary, ordinal, interval-scaled, ratio-scaled**
- **Many types of data sets, e.g., numerical, text, graph, Web, image.**
- **Gain insight into the data by:**
 - **Basic statistical data description: central tendency, dispersion, graphical displays**
 - **Data visualization: map data onto graphical primitives**
 - **Measure data similarity**
- **Above steps are the beginning of data preprocessing.**
- **Many methods have been developed but still an active area of research.**

data preprocessing

- **数据质量**：准确性、完整性、一致性、及时性、可信度、可解释性
- **数据清理**：如缺失/噪声值、异常值
- 来自多个来源的**数据集成**。
 - 实体识别问题；消除冗余；检测不一致的地方
- **数据归约、数据变换和数据离散化**
 - 数值归约；数据压缩
 - 归一化；概念层次的生成
- **降低维度**

Data warehouse and OLAP

- 数据仓库。数据仓库的多维模型
 - 一个数据立方体由 *维度表* 和 *事实表* 组成
 - 星星模式、雪花模式、事实星座
 - OLAP操作：钻孔、滚动、切片、切块和透视
- 数据仓库架构、设计和使用
 - 多层次的架构
 - 业务分析设计框架
 - 信息处理、分析处理、数据挖掘、OLAM
- 实施。高效地计算数据方块
 - 局部与全部与无物化
 - 对OLAP数据进行索引。位图索引和连接索引
 - OLAP查询处理
 - OLAP服务器。ROLLAP, MOLAP, HOLAP

datacube

- 数据立方体的计算：初步的概念
- 数据立方体的计算方法
 - 多路阵列聚合
 - BUC
 - 使用Shell-Fragments的高维OLAP
- 立方空间的多维数据分析
 - 多功能立方体
 - 发现驱动的数据立方体探索

FPBasic

- 基本概念
 - 什么是模式发现？为什么它很重要？
 - 基本概念。频繁模式和关联规则
 - 压缩表示。封闭模式和最大模式
- 高效的模式挖掘方法
 - 频繁模式的向下闭合特性
 - Apriori算法
 - Apriori的扩展或改进
 - 通过探索垂直数据格式挖掘频繁模式
 - FPGrowth: 频繁模式增长的方法
 - 挖掘封闭模式
- 模式评估
 - 模式挖掘中的趣味性度量
 - 有趣的度量。提升和 χ^2
 - 无变量度量
 - 趣味性度量的比较

数据仓库与数据挖掘

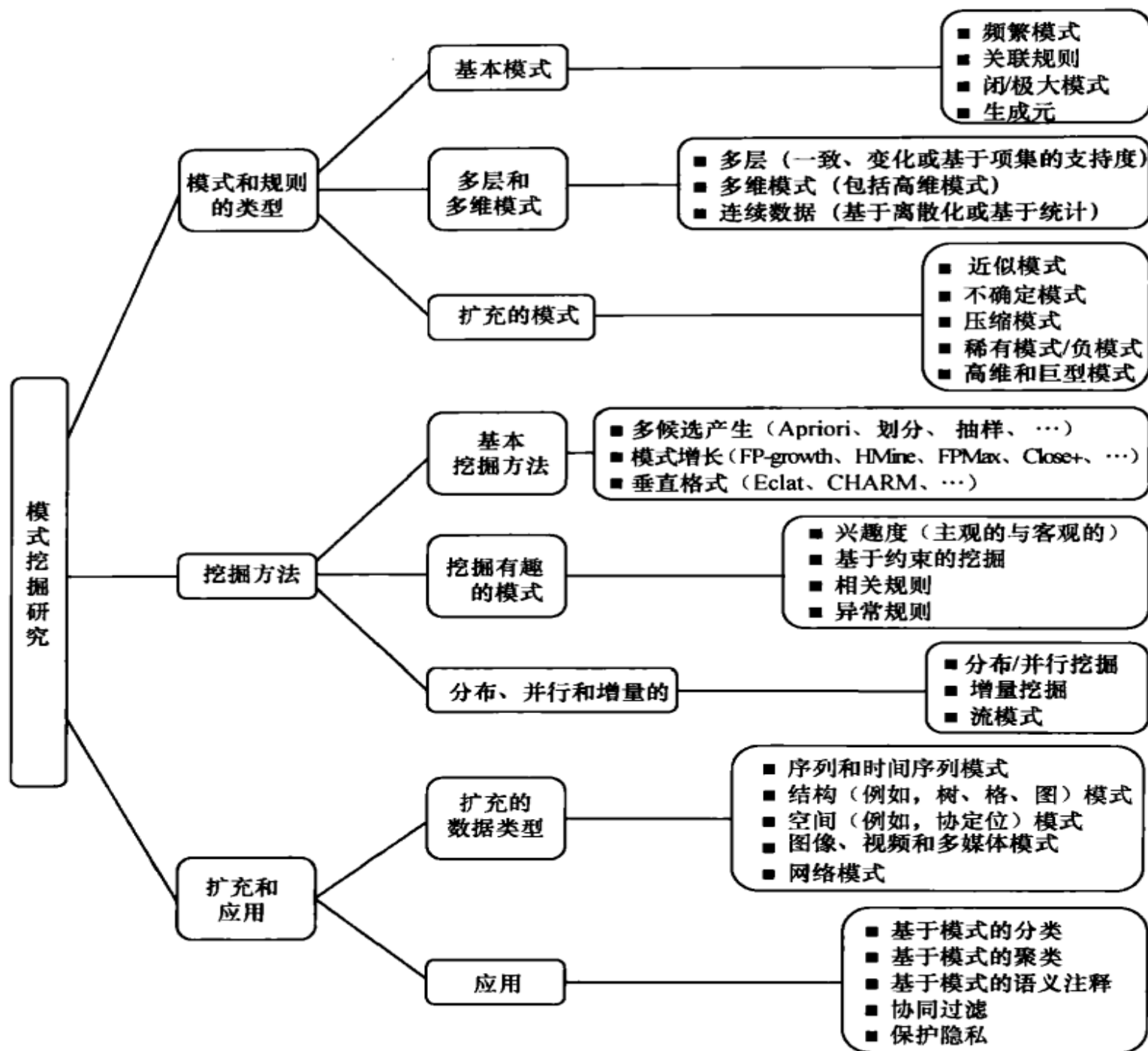
Data warehouse and data mining

丁钰

yuding@live.com

南京农业大学人工智能学院

第七章:高级模式挖掘



第7章 :高级模式挖掘

- 多层、多维空间中的模式挖掘
- 序列模式挖掘
- 基于约束的频繁模式挖掘
- 图模式挖掘
- 模式挖掘应用
- 摘要



多维、多层空间中的模式挖掘

- 挖掘多层关联规则
- 挖掘多维关联规则
- 挖掘量化关联规则
- 挖掘负模式
- 挖掘压缩和低冗余的模式



挖掘多层的频繁模式

表 7.1 任务相关的数据, D

TID	购买的商品
T100	Apple17" MacBook Pro Notebook, HP Photosmart Pro b9180
T200	Microsoft Office Professional 2010, Microsoft Wireless Optical Mouse 5000
T300	Logitech VX Nano Cordless Laser Mouse, Fellowes GEL Wrist Rest
T400	Dell Studio XPS 16 Notebook, Canon PowerShot SD1400
T500	Lenovo ThinkPad X200 Tablet PC, Symantec Norton Antivirus 2010
...	...

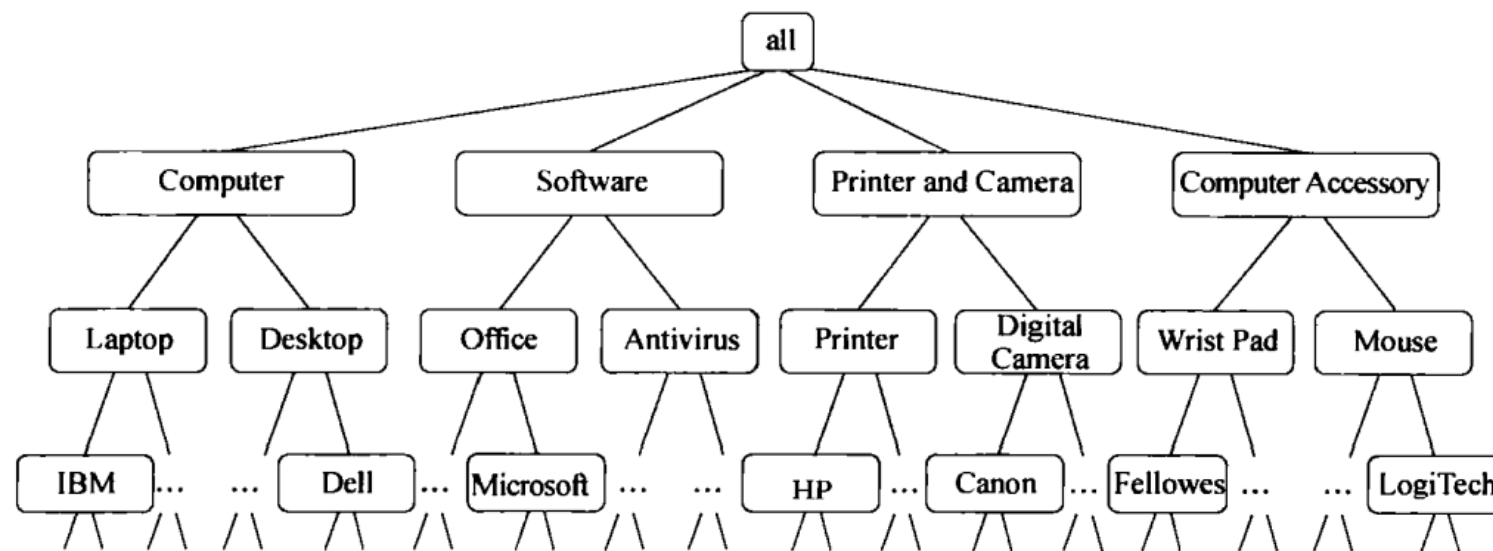
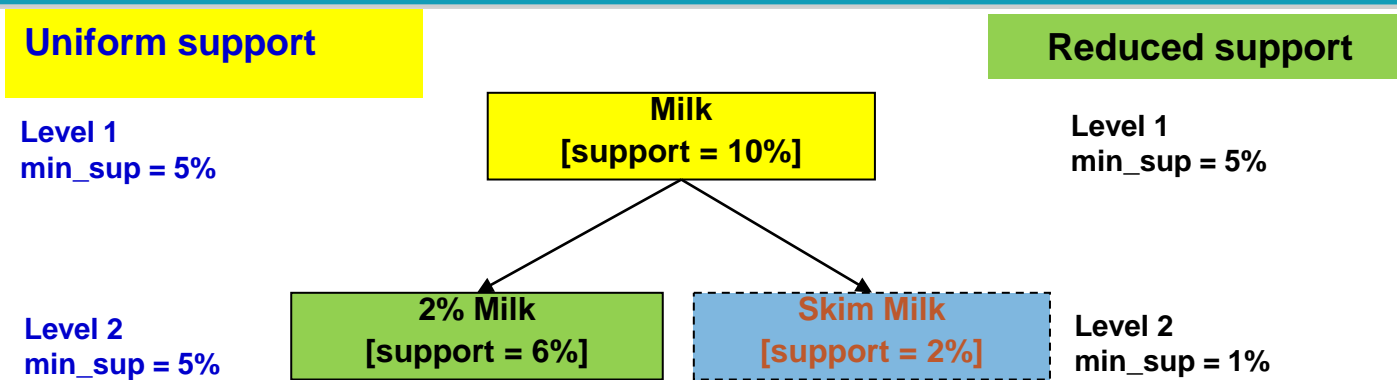


图 7.2 AllElectronics 计算机商品的概念分层

挖掘多层的频繁模式

- 项往往形成层次结构
 - 例如：Dairyland 2%牛奶；Wonder小麦面包
- 如何设置最小支持阈值？



- ☐ 多个级别的统一最小支持（合理？）
- ☐ 级别越低的支持度越小：处于较低水平的项预计会有较低的支持度
- ☐ 高效的挖掘：共享多层次挖掘
 - ☐ 使用最低的最小支持度来向下传递候选项集合

挖掘多层关联时的冗余过滤

- 多层的关联挖掘可能会产生许多冗余的规则
- 冗余过滤： 由于项目之间的 "祖先 "关系，一些规则可能是多余的
 - 牛奶 → 小麦面包[支持度=8%， 置信度=70%] (1)
 - 2%牛奶 → 小麦面包[支持度=2%， 置信度=72%] (2)
 - 假设2%的牛奶销售量约为牛奶总销量的1/4
 - (2)应该能够从(1)中 "衍生 "出来
- 如果根据其 "祖先 "规则，一个规则的支持度接近于 "预期 "值，并且它的置信度与 "祖先 "相似，那么这个规则就是多余的。
 - 规则(1)是规则(2)的祖先，要修剪哪一条？

为不同类型的项提供定制的最小支持度

- 在每个关联挖掘中，我们对所有要挖掘的项目或项目集使用相同的最小支持阈值。
- 在现实中，有些物品（如钻石、手表.....）是有价值的，但较少出现。
- 有必要为不同类型的项目定制最小支持设置
- 一种方法。使用基于小组的“个性化”支持度
 - 例如：{钻石，手表}：0.05%；{面包，牛奶}：5%；...
 - 如何有效地挖掘这种规则？
 - 现有的可扩展的挖掘算法可以很容易地扩展，以涵盖这种情况

挖掘多维关联

- 单一维度的规则（例如，项都在 "产品" 维度上）。
 - $\text{买}(X, \text{"牛奶"}) \rightarrow \text{买}(X, \text{"面包"})$
- 多维规则（即大于2个维度（谓词）的项）。
 - 维度间的关联规则（没有重复的谓词）。
 - $\text{年龄}(X, \text{"18-25"}) \wedge \text{职业}(X, \text{"学生"}) \rightarrow \text{购买}(X, \text{"可乐"})$
 - 混合维度的关联规则（重复谓词）。
 - $\text{age}(X, \text{"18-25"}) \wedge \text{buys}(X, \text{"popcorn"}) \rightarrow \text{buys}(X, \text{"coke"})$
- 属性可以是分类的或数字的
 - 分类属性（例如，职业、产品：数值之间没有排序）：用于维度间关联的数据立方体
 - 定量属性：数值、数值间的隐性排序--分层、聚类 and 梯度方法



挖掘定量关联

- 挖掘数字属性的关联
 - 例：数字属性：年龄和工资
 - 如果你要考虑每一个年龄，或者每一个薪酬，那么项的种类就会过于丰富，从而导致我们不能敏感地发现有价值的关联规则。
- 方法
 - 基于预定义概念层次的静态离散化
 - 在每个维度上进行分层细化，具有层次性
 - 年龄。 $\{0-10, 10-20, \dots, 90-100\} \rightarrow \{\text{年轻}, \text{中年}, \text{老年}\}$ 。
 - 基于数据分布的动态离散化
 - 聚类。基于距离的关联
 - 首先是一维聚类，然后是关联
 - 偏差分析。异常
 - 性别=女性 \rightarrow 工资：平均=7美元/小时（总体平均=9美元）。

量化关联挖掘中的非常态现象

- 挖掘非凡（即有趣）的现象 美国1985年实际数据
 - 例如：性别=女性 \rightarrow 工资：平均值=7美元/小时（总体平均值=9美元）。
 - LHS：人口的一个子集
 - RHS：这个子集的一个特殊行为
- 只有当统计检验（如Z检验）以高置信度证实了推论时，该规则才会被接受。
- 子规则：突出超级规则的人口子集的非凡行为
 - 例如：（性别=女性） \wedge （南方=是） \rightarrow 平均工资=6.3美元/小时
- 规则条件可以是分类的或数字的（定量规则）。
 - 例如：在[14-18]（年）的教育经历 \rightarrow 平均工资=11.64美元/小时
- 已经开发了高效的方法来挖掘这种特殊规则（例如，Aumann和Lindell@KDD'99）。

稀有模式和负模式

- 稀有模式
 - 支持率很低，但很有趣（例如，购买劳力士手表）。
 - 如何挖掘它们？为不同的项目组设置个性化的、基于群体的最小支持阈值
- 负模式
 - 负相关的。不太可能一起发生
 - 例子：由于同一个客户不太可能同时购买福特Expedition（一款SUV汽车）和福特 Fusion（一款混合动力汽车），所以购买福特Expedition和购买福特 Fusion 很可能是负相关的模式。
 - 如何定义负模式？

定义负相关模式

- 一个基于支持度的定义
 - 如果项集A和B都是频繁出现的，但很少一起出现，即 $\text{sup}(A \cup B) \ll \text{sup}(A) \times \text{sup}(B)$
 - 那么A和B是负相关的
- 这对大型交易数据集来说是一个好的定义吗？
- 例子：假设一家商店将两个针包A和B各卖出100次，但只有一次交易同时包含A和B
 - 当总共有200个交易时，我们有
 - $s(A \cup B) = 0.005, s(A) \times s(B) = 0.25, s(A \cup B) \ll s(A) \times s(B)$
 - 但当有 10^5 个交易时，我们有
 - $s(A \cup B) = 1/10^5, s(A) \times s(B) = 1/10^3 \times 1/10^3, s(A \cup B) > s(A) \times s(B)$
 - 问题是什么？-零事务。基于支持度的定义不是空变的！

这是否让你想起了 *lift* 提升度的定义？

负相关的定义：定义中需要无差异性

- 一个关于负相关的好的定义应该照顾到零事务问题
 - 两个项集A和B是否是负相关的，不应该受到零事务的影响。
- 基于库尔钦斯基Kulczynski度量的定义
 - 如果项目集A和B是频繁的，但
$$(s(A \cup B)/s(A) + s(A \cup B)/s(B))/2 < \epsilon。$$
其中 ϵ 是一个负模式阈值，那么A和B是负相关的。
- 对于同样的问题：
 - 不管总共有200个还是10个⁵交易
 - 如果 $\epsilon=0.01$ ，我们有
$$(s(A \cup B)/s(A) + s(A \cup B)/s(B))/2 = (0.01 + 0.01)/2 < \epsilon$$



挖掘压缩的模式

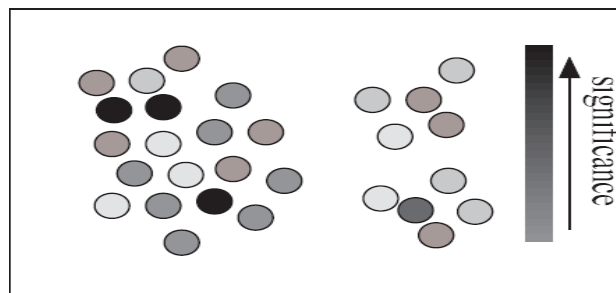
Pat-ID	项目集	支持
P1	{38,16,18,12}	205227
P2	{38,16,18,12,17}	205211
P3	{39,38,16,18,12,17}	101758
P4	{39,16,18,12,17}	161563
P5	{39,16,18,12}	161576

- ❑ 闭模式
 - ❑ P1, P2, P3, P4, P5
 - ❑ 太强调支持了
 - ❑ 没有压缩
- ❑ 最大模式
 - ❑ P3: 信息丢失
- ❑ 希望的产出（良好的平衡）。
 - ❑ P2, P3, P4

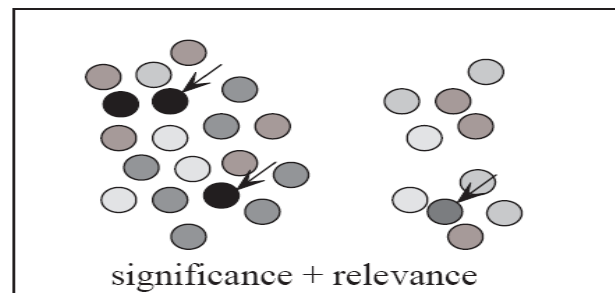
- 为什么要挖掘压缩模式？
 - 太多零散的模式，但意义不大
- 模式距离测量
$$Dist(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|}$$
- δ 聚类。对于每个模式P，找到所有可以由P表达的模式，并且与P的距离在 δ 之内（ δ -覆盖）。
- 集群中的所有模式都可以用P表示
- 高效、直接挖掘压缩频繁模式的方法（例如，D. Xin, J. Han, X. Yan, H. Cheng, "On Compressing Frequent Patterns", Knowledge and Data Engineering, 60:5-29, 2007)

提取感知冗余的Top-k模式

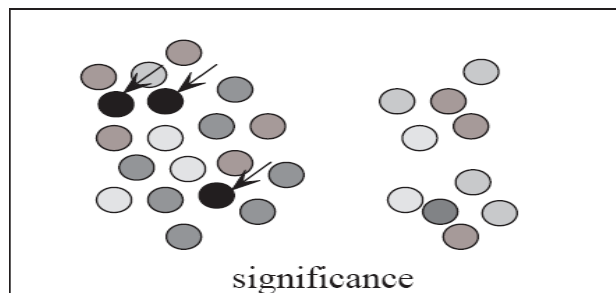
- 希望的模式：高显著性和低冗余度



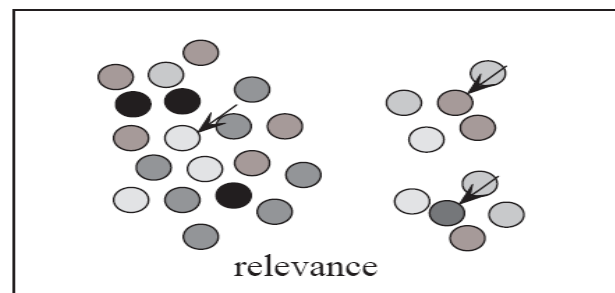
(a) a set of patterns



(b) redundancy-aware top- k




(c) traditional top- k



(d) summarization

- 方法：使用MMS（最大化边缘显著性）来衡量模式集的综合显著性
- Xin et al., Extracting Redundancy-Aware Top-K Patterns, KDD'06

第7章 :高级模式挖掘

- 多层、多维空间中的模式挖掘
- 序列模式挖掘 
- 基于约束的频繁模式挖掘
- 图模式挖掘
- 模式挖掘应用
- 摘要

序列模式挖掘

- 序列模式和序列模式挖掘
- GSP: 基于Apriori的序列模式挖掘
- SPADE: 垂直数据格式的序列模式挖掘
- PrefixSpan: 通过模式增长的序列模式挖掘
- CloSpan: 挖掘封闭的序列模式

序列数据库和序列模式

- 顺序模式挖掘有广泛的应用
 - 客户购物顺序
 - 在6个月内先购买笔记本电脑，然后是数码相机，最后是智能手机。
 - 医学治疗、自然灾害（如地震）、科学与工程过程、股票和市场.....。
 - 网络日志点击流，调用模式， ...
 - 软件工程：程序执行序列， ...
 - 生物学序列： **DNA**、蛋白质、 ...
- 事务数据库、序列数据库与时间序列数据库的关系
- 有间隙的与无间隙的顺序模式
 - 购物序列， 点击流 生物序列

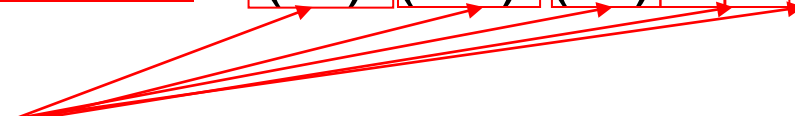
序列模式和序列模式挖掘

- 序列模式挖掘。给定一组序列，找到完整的频繁子序列集（满足min_sup阈值）

一个序列数据库

SID	Sequence
10	<a(<u>abc</u>)(a <u>c</u>)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(<u>ab</u>)(df) <u>c</u> b>
40	<eg(af)cbc>

一个序列: <(ef)(ab)(df)c b>



- 一个元素可以包含一组项（也叫事件）。
- 一个元素内的项目是无序的，我们按字母顺序排列

<a(bc)dc>是<a(abc)(ac)d(cf)>的一个子序列。

- 考虑到支持度阈值min_sup=2，<(ab)c>是一个序列模式

序列模式挖掘算法

- 算法要求：高效，可扩展，找到完整的集合，纳入各种用户特定的约束条件
- Apriori属性仍然成立。如果一个子序列 s_1 是不频繁的，那么这个 s_1 子序列的上级序列就不可能是频繁的。
- 代表性的算法
 - **GSP** (Generalized Sequential Patterns): Srikant & Agrawal @ EDBT'96)
 - 基于垂直格式的挖掘方法。 **SPADE** (Zaki @ Machine Learning'00)。
 - 模式增长的方法。 **PrefixSpan** (Pei, et al. @ TKDE'04)
- 挖掘闭的序列模式。 **CloSpan** (Yan, et al. @ SDM'03)
- 基于约束的顺序模式挖掘（将在约束挖掘部分介绍）。

GSP: 基于Apriori的顺序模式挖掘

- 最初的候选。所有8个单体序列
 - <a>, , <c>, <d>, <e>, <f>, <g>, <h>.
- 扫描一次数据库, 统计每个候选项的支持度
- 产生长度为2的候选序列

$min_sup = 2$

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

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

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

- 没有Apriori修剪。
(8个单数) $8*8+8*7/2=92$
个长度为2候选
- 通过修剪, 长度为2的
候选人。 $36 + 15 = 51$

GSP (Generalized Sequential Patterns): Srikant & Agrawal @ EDBT'96)



GSP挖掘和修剪

5th扫描: 1候选。1 length-5 seq. pat

4th扫描: 8候选。7 length-4 seq. pat

3rd扫描: 46 候选。20 length-3 seq. pat。
20个候选完全不在数据库中

2nd扫描: 51个候选。19 length-2 seq. pat。
10个候选完全不在数据库中

1st扫描: 8个候选。6 length-1 seq. pat.

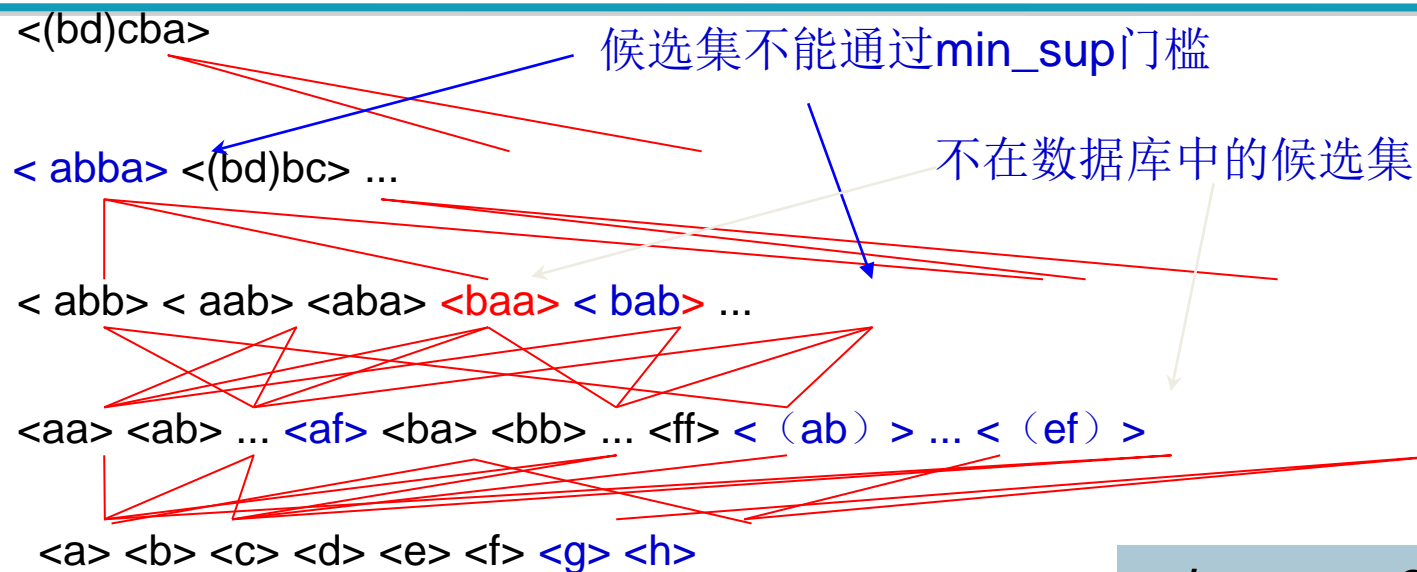
❑ 重复（对于每个级别（即长度-k））。

❑ 扫描数据库以找到长度为k的频繁序列

❑ 使用Apriori从长度为k的频繁序列中生成长度为(k+1)的候选序列

❑ 设置 $k = k+1$

❑ 直到没有频繁的序列或找不到候选为止



$min_sup = 2$

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



垂直数据格式中的序列模式挖掘：SPADE算法

- 一个序列数据库被映射到： $\langle \text{SID}, \text{EID} \rangle$ 。
- 通过Apriori候选者生成，逐项增长子序列（模式）。

SID	Sequence
1	$\langle a(\underline{abc})(ac)d(cf) \rangle$
2	$\langle (ad)c(bc)(ae) \rangle$
3	$\langle (ef)(\underline{ab})(df)\underline{cb} \rangle$
4	$\langle eg(af)cbc \rangle$

$min_sup = 2$

参考文献: SPADE
(Sequential PAttern
Discovery using Equivalent
Class) [M. Zaki 2001)

SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c

a		b		...
SID	EID	SID	EID	...
1	1	1	2	
1	2	2	3	
1	3	3	2	
2	1	3	5	
2	4	4	5	
3	2			
4	3			

ab			ba			...
SID	EID (a)	EID(b)	SID	EID (b)	EID(a)	...
1	1	2	1	2	3	
2	1	3	2	3	4	
3	2	5				
4	3	5				

aba				...
SID	EID (a)	EID(b)	EID(a)	...
1	1	2	3	
2	1	3	4	

PrefixSpan: 一种模式增长的方法

		$min_sup = 2$	
SID	Sequence	Prefix	Suffix (Projection)
10	<a(<u>a</u> bc)(a <u>c</u>)d(cf)>	<a>	<(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>	<aa>	<(_bc)(ac)d(cf)>
30	<(ef)(<u>a</u> b)(df) <u>c</u> b>	<ab>	<(_c)(ac)d(cf)>
40	<eg(af)cbc>		

前缀和后缀

给出<a(abc)(ac)d(cf)>

前缀: <a>, <aa>, <a(ab)>, <a(abc)>, ...

后缀: 基于前缀的投影

- PrefixSpan: Prefix-Projected Pattern Growth, 即前缀投影的模式挖掘

- 步骤1: 寻找长度为1的序列模式
 - <a>, , <c>, <d>, <e>, <f>.
- 第2步: 划分搜索空间, 挖掘每个投射DB
 - <a>预测的DB.
 - 预计的DB.
 - ...
 - <f>预测的DB, ...

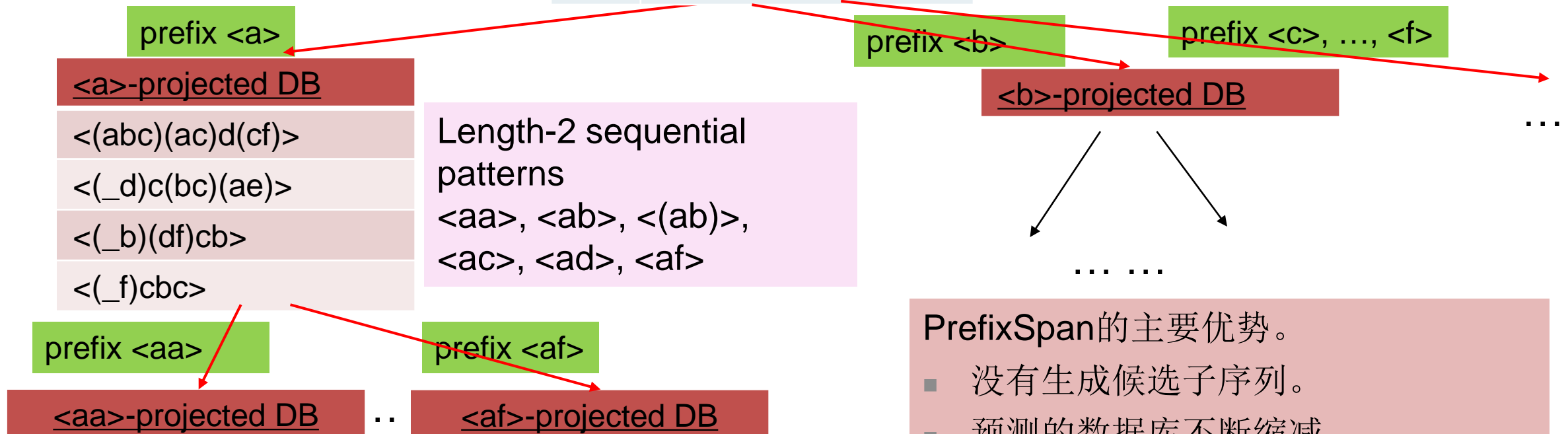
PrefixSpan (Prefix-projected Sequential pattern mining) Pei, et al. @TKDE'04

PrefixSpan: 挖掘前缀预测的数据库

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

$min_sup = 2$

Length-1 sequential patterns
<a>, , <c>, <d>, <e>, <f>



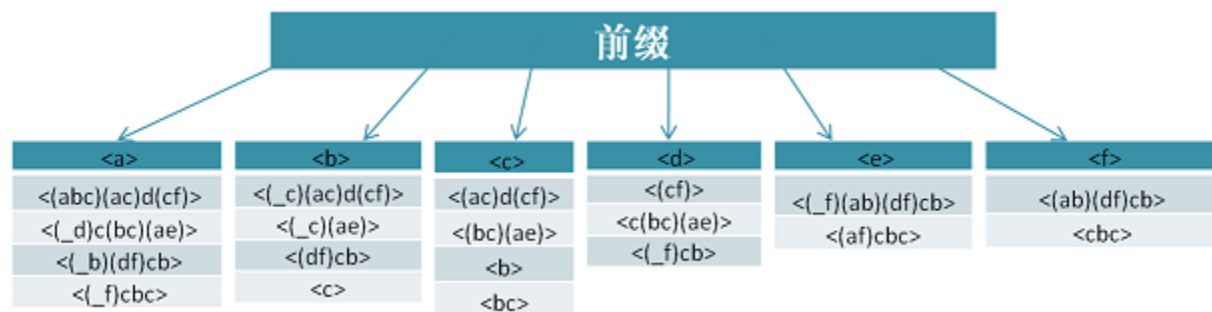
PrefixSpan的主要优势。

- 没有生成候选子序列。
- 预测的数据库不断缩减

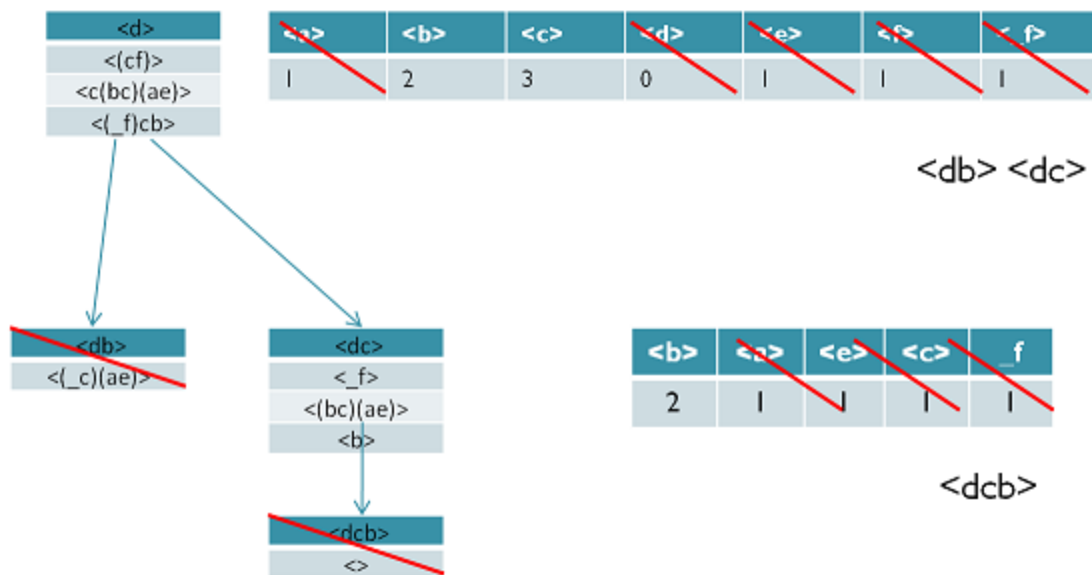
id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a><c><d><e><f>

<a>		<c>	<d>	<e>	<f>	<g>
4	4	4	3	3	3	1



去除所有序列中的g, 即第4条记录变成<e(af)cbc>

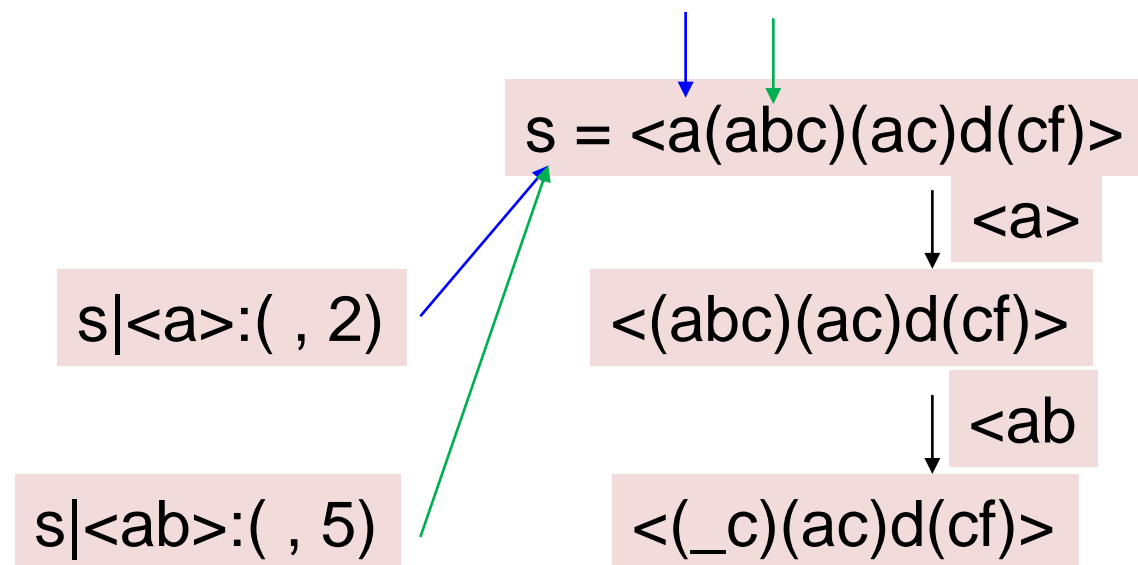


实现方面的考虑：伪投射与物理投射

- PrefixSpan的主要成本：递归的构造投影数据库
 - 在递归预测数据库中，后缀基本上是重复的。
- 当DB可以被保存在主内存中时，使用伪投射

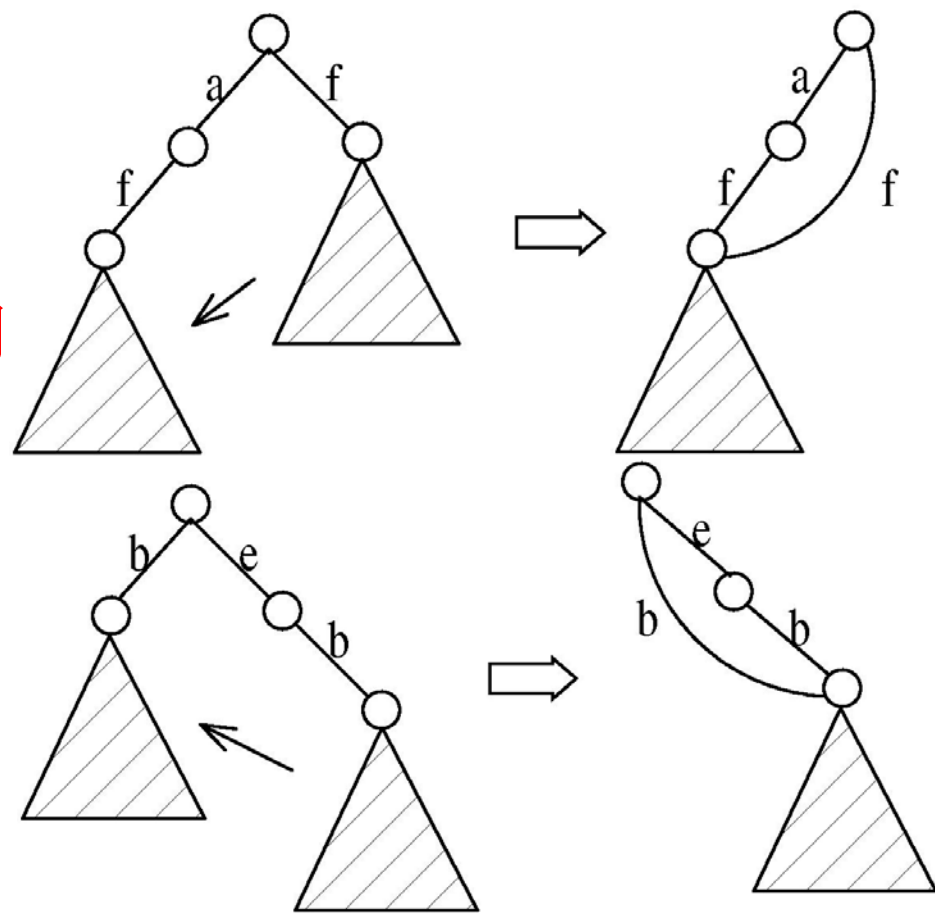
- 没有物理复制的后缀
- 指向序列的指针
- 后缀的偏移
- 但如果它不适合在内存中
- 物理投射
- 建议的方法。

- 物理和伪投影的整合
- 当数据适合在内存中时，换成伪投影法
- 伪投影避免了物理的复制后缀，这样它在运行时间和空间上就很高效，但它不适用于基于硬盘的存取。因为随机存取硬盘空间是十分耗时的。基于此原因，只要主存中可以容下一个投影数据库，就使用伪投影技术，如果原始序列数据库或投影数据库太大而不适应于主存，就用物理投影。



CloSpan:挖掘封闭的序列模式

- 一个**封闭的序列模式** s 。 不存在任何超级模式 s' , 使得 $s' \supset s$, 并且 s' 和 s 具有相同的支持度。
- 哪些是闭的? $\langle abc \rangle: 20, \langle abcd \rangle: 20, \langle abcde \rangle: 15$
 - 为什么要直接挖掘封闭的序列模式?
 - 减少 (多余的) 模式的数量
 - 达到同样的表现力
 - 属性 P_1 : 如果 $s \supset s_1$, 当且仅当两个投影DB具有相同的大小时 s 是封闭的。
 - 探索**后向子模式**和**后向超级模式**的修剪, 以修剪多余的搜索空间
 - 极大地提高了效率 (Yan, et al., SDM'03)。

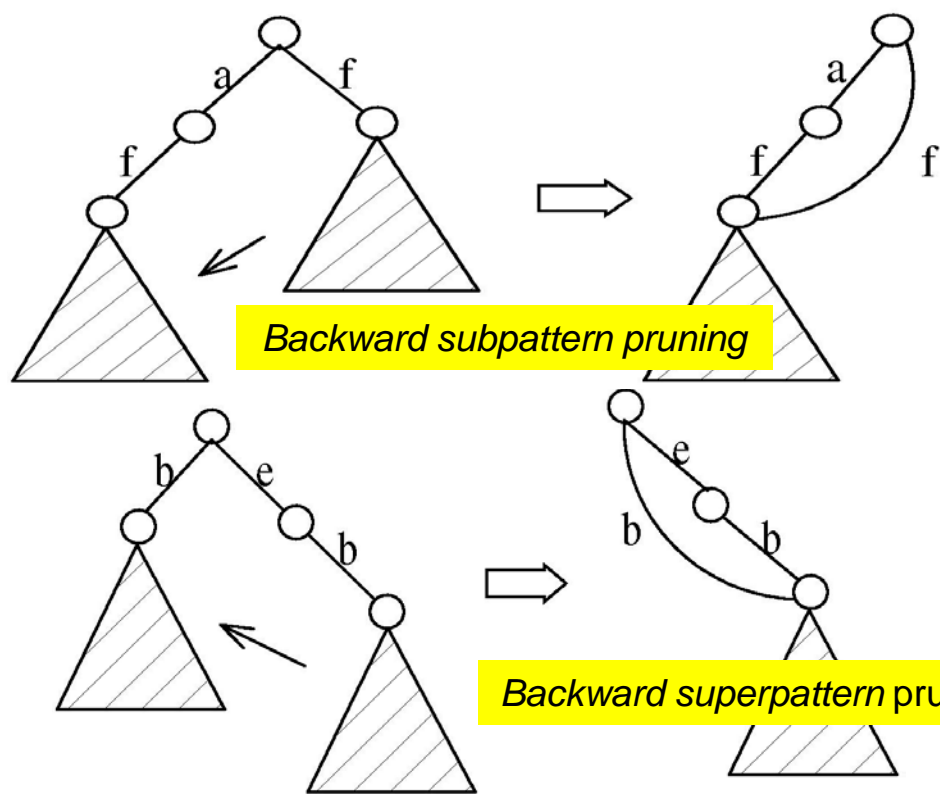


CloSpan:当两个预测的DB具有相同的尺寸时

如果 $s \supset s_1$, s 是封闭的如果 $s \supset s_1$, 当且仅当两个投影DB具有相同的大小时 s 是封闭的。

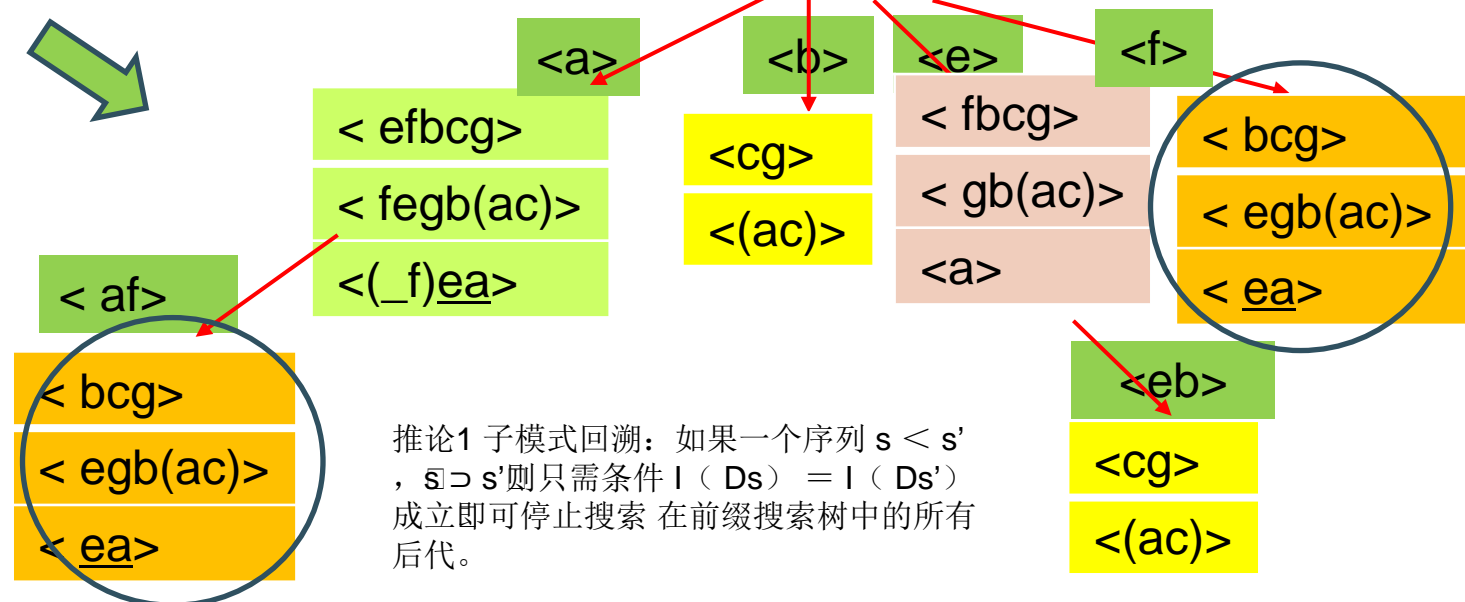
当两个投影序列DB具有相同的大小时？

这里有一个例子。



id	Sequence
1	< aefbcg>
2	<afegb(ac)>
3	<(af)ea>


$min_sup = 2$



推论1 子模式回溯：如果一个序列 $s < s'$, $s \supset s'$ 则只需条件 $I(Ds) = I(Ds')$ 成立即可停止搜索 在前缀搜索树中的所有后代。

推论2 父模式回溯：如果一个序列 $s < s'$, $s \subset s'$ 则只需条件 $I(Ds) = I(Ds')$ 成立 则将 s 在前缀搜索树中的所有子树移植给 s' 而不必搜索 s' 任何后代。

第7章 :高级模式挖掘

- 多层、多维空间中的模式挖掘
- 序列模式挖掘
- 基于约束的频繁模式挖掘 
- 图模式挖掘
- 模式挖掘应用
- 摘要

基于约束的模式挖掘

- 为什么要基于约束挖掘？
- 不同种类的约束：不同的修剪策略
- 带模式反单调性的约束性挖掘
- 带有模式单调性的约束性挖掘
- 带有数据反单调性的约束性挖掘
- 带有简洁约束的约束性挖掘
- 可转换约束条件下的约束性挖掘
- 处理多个约束条件
- 基于约束的序列模式挖掘

为什么要基于约束挖掘？

- 自主地找到数据集中的所有模式？--不现实!
 - 模式太多，但不一定是用户感兴趣的!
- 实践中的模式挖掘。通常是一个由用户引导的互动过程
 - 用户使用数据挖掘查询语言（或图形用户界面）指导要挖掘的内容，指定各种约束条件。
- 什么是基于约束条件的挖掘？
 - 与用户提供的约束条件一起挖掘
- 为什么要基于约束挖掘？
 - 用户使用的灵活性：用户对要挖掘的内容提供约束
 - 优化：系统探索这种制约因素，以提高挖掘效率
 - 例如，将约束条件深入到挖掘过程中去

数据挖掘中各种类型的用户指定的约束条件

- **知识类型约束**--指定要挖掘哪些类型的知识
 - 例如：分类、关联、聚类、离群点查找等。
- **数据约束**-使用类似SQL的查询
 - 例：寻找今年在纽约商店一起销售的产品
- **维度/水平约束**-类似于关系数据库中的投影
 - 例如：与地区、价格、品牌、客户类别相关的情况
- **兴趣度约束**--各种阈值
 - 例如：强规则：min_sup>0.02, min_conf>0.6, min_correlation>0.7
- **规则（或模式）的约束**
 - 例如：小销售（价格<10美元）引发大销售（总和>200美元）。

模式反单调性约束下的模式空间剪枝

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

Item	Price	Profit
a	100	40
b	40	0
c	150	-20
d	35	-15
e	55	-30
f	45	-10
g	80	20
h	10	5

- 一个约束条件 c 是**反单调的**
 - 如果一个项集 S 违反了约束条件 c ，那么它的任何一个超集也会违反。
 - 也就是说，项目集 S 的挖掘可以被终止
- 例1: $c_1: \text{sum}(S.\text{price}) \leq v$ 是**反单调的**。
- 例2: $c_2: \text{range}(S.\text{profit}) \leq 15$ 是**反单调的**。
 - 项目集 ab 违反了 c_2 ($\text{range}(ab) = 40$)。
 - 每一个 ab 的超集也是如此
- 例3: $c_3: \text{sum}(S.\text{Price}) \geq v$ **不是反单调的**
- 例4: $c_4: \text{support}(S) \geq \sigma$ 是反单调的吗?
 - 是的!Apriori修剪本质上是一种反单调约束的修剪!

注: $\text{item.price} > 0$
利润可以是负数

模式单调性及其作用

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

Item	Price	Profit
a	100	40
b	40	0
c	150	-20
d	35	-15
e	55	-30
f	45	-10
g	80	20
h	10	5

■ 一个约束条件 c 是 *单调的*: 如果一个项集 S 满足约束 c , 那么它的任何一个超集也满足约束 c 。

■ 也就是说, 我们不需要在随后的挖掘中检查 c

■ 例1: $c_1: \text{sum}(S.\text{Price}) \geq v$ 是 *单调的*

■ 例2: $c_2: \text{min}(S.\text{Price}) \leq v$ 是 *单调的*。

■ 例3: $c_3: \text{range}(S.\text{profit}) \geq 15$ 是 *单调的*

■ 项目集 ab 满足 c_3

■ 每一个 ab 的超集也是如此

注: $\text{item.price} > 0$
利润可以是负数

带有数据反单调约束的数据空间修剪

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

Item	Price	Profit
a	100	40
b	40	0
c	150	-20
d	35	-15
e	55	-30
f	45	-10
g	80	20
h	10	5

- 一个约束条件 c 是**数据反单调的**：在挖掘过程中，如果一个数据条目 t 不能满足 c 下的模式 p ，那么 t 也不能满足 p 的上集
 - 数据空间修剪：数据输入 t 可以被修剪掉
- 例1: $c_1: \text{sum}(S.\text{Profit}) \geq v$ 是**数据反单调的**
 - 让约束条件 c_1 为: $\text{sum}(S.\text{Profit}) \geq 25$
 - 假设当前 $S=\{c\}$, c 的投影数据库为
 - 10: $\{a, b, d, f, h\}$. 20: $\{b, d, f, g, h\}$, 30: $\{b, d, f, g\}$
 - $T_{30}: \{b, c, d, f, g\}$ 可以被删除，因为它们的组合(加 c)都不能构成一个利润之和 ≥ 25 的 S 。
- 例2: $c_2: \text{min}(S.\text{Price}) \leq v$ 是**数据反单调的**。
 - 考虑 $v=5$ ，但交易中的每件物品，其价格都高于10

注: item.price > 0
利润可以是负数

数据空间修剪应该被反复探索

例子. c_3 : $range(S.Profit) > 25$

- 我们检查b的预测数据库
 - 但 "a" 项不经常出现 ($sup=1$)。
- 从 T_{10} 中删除 "a (40)" 后
 - T_{10} 不能再满足 c_3
 - 由于 "b (0)" 和 "c (-20), d (-15), f (-10), h (5)"
 - 通过删除 T_{10} , 我们也可以修剪 T_{20} 中的 "h"

b's-proj.DB

TID	Transaction
10	a, c, d, f, h
20	c, d, f, g, h
30	c, d, f, g

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	-15
e	-30
f	-10
g	20
h	5

$min_sup = 2$ $price(item) > 0$

约束条件:
 $range\{S.profit\} > 25$

b's-proj.DB	TID	Transaction
	10	a, c, d, f, h
	20	c, d, f, g, h
	30	c, d, f, g

递归
数据
修剪

b's FP-tree

single branch : cdfg: 2

在b的预测数据库中只有一个分支 "cdfg: 2" 需要挖掘

- 注意: 只有在 "a" 在b的预测数据库中被修剪 (通过min-sup) 后, c_3 才能有效地修剪 T_{10}

简洁性：修剪数据空间和模式空间

- 简洁性：约束条件 c 可以通过直接操作数据来执行
- 例1：要找到那些没有第 i 项的模式
 - 从DB中删除 i ，然后挖掘（模式空间修剪）。
- 例2：要找到包含第 i 项的模式
 - 只挖掘 i -projected DB（数据空间剪裁）。
- 例3: $c_3: \min(S.Price) \leq v$ 是简洁的
 - 只从价格 $\leq v$ 的项目开始，只删除高价项目的交易（模式+数据空间修剪）。
- 例4: $c_4: \sum(S.Price) \geq v$ 不简洁
 - 由于项目集 S 的价格之和不断增加，所以无法事先确定。

可转换的约束条件:交易中的数据排序

TID	Transaction
10	a, b, c, d, f, h
20	a, b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

Item	Price	Profit
a	100	40
b	40	0
c	150	-20
d	35	-15
e	55	-30
f	45	-5
g	80	30
h	10	5

- 通过对交易中的项目进行适当排序，将强硬的约束条件转换成（反）单调的约束条件
- 检查 c_1 : $\text{avg}(S.\text{profit}) > 20$
 - 按（利润）值降序排列的项目
 - $\langle a, g, h, b, f, d, c, e \rangle$
 - 一个项目集 ab 违反了 c_1 ($\text{avg}(ab) = 20$)。
 - ab^* 也是如此（即 $ab\text{-projected DB}$ ）。
 - C_1 : 如果模式以正确的顺序增长，则是反单调的!
- 项排序法能对Apriori起作用吗?
 - 逐级生成候选需要多向检查!
 - $\text{avg}(agf) = 21.7 > 20$, 但 $\text{avg}(gf) = 12.5 < 20$
 - Apriori不会生成 "agf "作为候选。

不同类型的约束导致了不同的修剪策略

- 综上所述，制约因素可分为以下几类
 - 模式空间的剪枝约束与数据空间的剪枝约束
- 模式空间修剪限制
 - 反单调性：如果违反了约束条件 c ，其进一步的开采可以被终止。
 - 单调性：如果 c 被满足，就不需要再检查 c 了
 - 简洁：如果约束条件 c 可以通过直接操作数据来执行
 - 可转换：如果项目在处理过程中可以适当排序， c 可以转换为单调或反单调的。
- 数据空间修剪限制
 - 数据简洁：数据空间可以在最初的模式挖掘过程中被修剪掉。
 - 数据反单调：如果一个交易 t 不满足 c ，那么 t 可以被修剪以减少数据处理工作量

如何处理多个约束条件？

- 在模式挖掘中使用多种约束是有益的
- 但不同的约束条件可能需要潜在的冲突的项目排序
 - 如果 c_1 和 c_2 之间存在冲突排序
 - 尝试对数据进行排序，先执行一个约束（哪一个？）
 - 然后在挖掘预测数据库时执行其他约束条件
- 例如： c_1 : $\text{avg}(S.\text{profit}) > 20$ ， 和 c_2 : $\text{avg}(S.\text{price}) < 50$
 - 假设 c_1 有更多的修剪能力
 - 按利润降序排序，先用 c_1
 - 对于每个投影数据库，按价格升序对交易进行排序，并在挖掘时使用 c_2




基于约束的序列模式挖掘

- 与基于约束的项集挖掘有许多相似之处
- **反单调性**：如果 S 违反了 c ， S 的超序列也违反了 c 。
 - $\text{sum}(S.\text{price}) < 150; \text{min}(S.\text{value}) > 10$
- **单调性**：如果 S 满足 c ，那么 S 的超级序列也满足 c 。
 - $\text{element_count}(S) > 5; S \{PC, \text{digital_camera}\}$
- **数据反单调**：如果一个关于 S 的序列 s_1 违反了 c_3 ， s_1 可以被删除
 - $c_3: \text{sum}(S.\text{price}) \geq v$
- **简洁**：通过明确地操作数据来执行约束条件 c
 - $S \{i\text{-phone}, \text{MacAir}\}$
- **可转换**。 基于排序值而不是序列顺序的投影
 - $\text{value_avg}(S) < 25; \text{profit_sum}(S) > 160$
 - $\text{max}(S)/\text{avg}(S) < 2; \text{median}(S) - \text{min}(S) > 5$

基于时间约束条件的序列模式挖掘

- **顺序约束**。有些项目必须在其他项目之前发生
 - {代数、几何}→{微积分}。(其中"→"表示排序)
 - 反单调性。修剪违反约束的子模式
- **最小间隙/最大间隙约束**。将两个元素限制在一个模式中
 - 例如, mingap=1, maxgap=4
 - 简洁: 在模式增长期间直接执行
- **最大跨度约束**。模式中的1st和最后一个元素之间的最大允许时间差
 - 例如, 最大跨度 (S) =60 (天)。
 - 简洁: 当1st元素被确定时直接执行
- **窗口大小的约束**。一个元素中的事件不一定要在同一时间发生。强制执行最大允许的时间差
 - 例如, 窗口大小=2: 各种方法将事件合并到元素中去

情节和频繁情节挖掘

- 情节是指事件序列上相互邻近并且满足一定结构关系的事件类型的集合
- 情节和正则表达式。对序列模式的替代
 - 连续情节: AB  一个总的顺序关系: 先A后B
 - 平行情节: $A|B$  是一种部分顺序关系。A和B可以是任何顺序的
 - 正则表达式: $(A|B)C^*(DE)$  (DE)意味着D、E发生在同一时间窗口。
- 例子: 给定一个大型的购物序列数据库, 人们可能想找到
 - 假设模式顺序遵循模板 $(A|B)C^*(D\ E)$, 并且
 - A、B、C*、D和E的价格之和大于100美元, 其中C*表示C出现*次
 - 如何有效地挖掘这种情节模式?

摘要：基于约束的模式挖掘

- 为什么是基于约束的挖掘？
- 不同种类的挖掘。不同的修剪策略
- 带图案反单调性的约束性挖掘
- 带有模式单调性的约束性挖掘
- 带有数据反单调性的约束性挖掘
- 带有简洁约束的约束性挖掘
- 可转换约束条件下的约束性挖掘
- 处理多个约束条件
- 基于约束的顺序模式挖掘

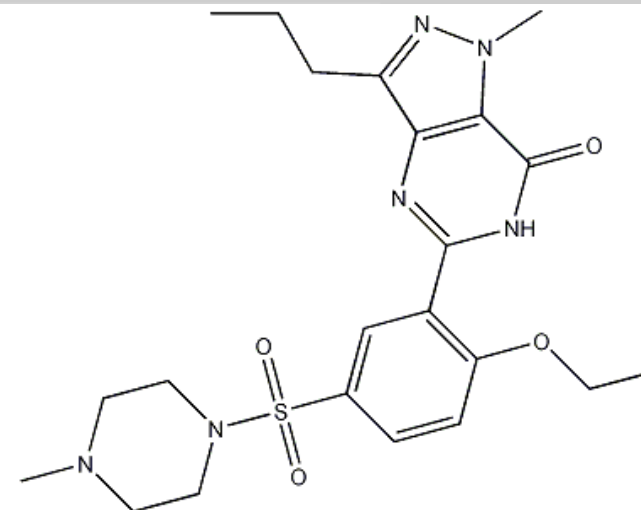
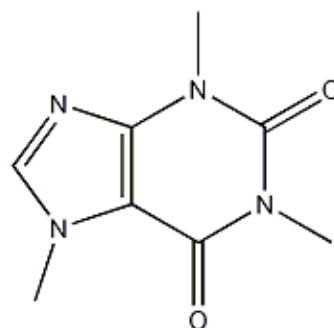
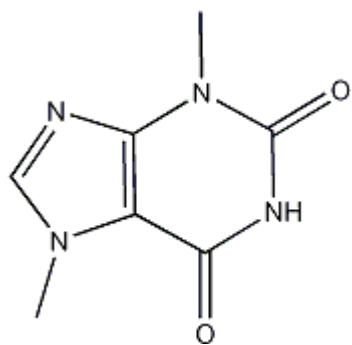
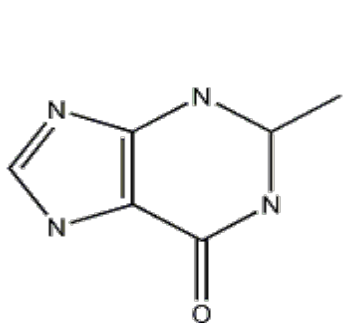
第7章 :高级模式挖掘

- 多层、多维空间中的模式挖掘
- 序列模式挖掘
- 基于约束的频繁模式挖掘
- 图模式挖掘
- 模式挖掘应用
- 摘要

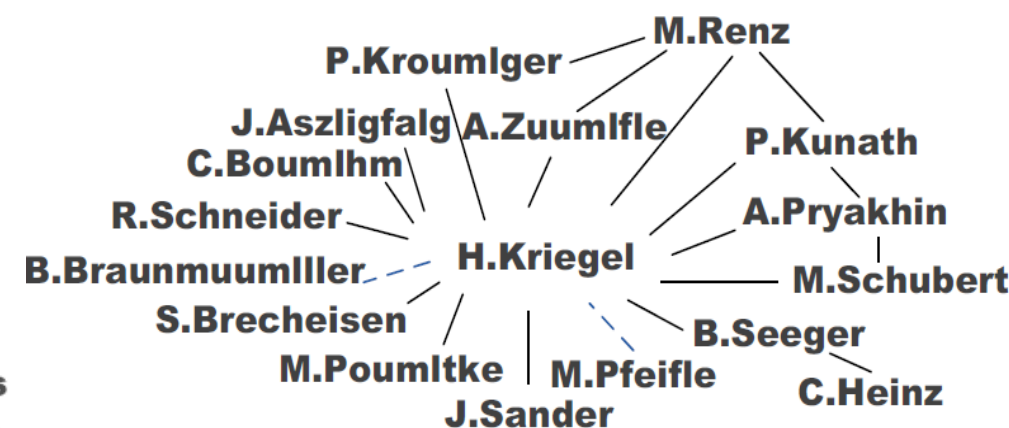
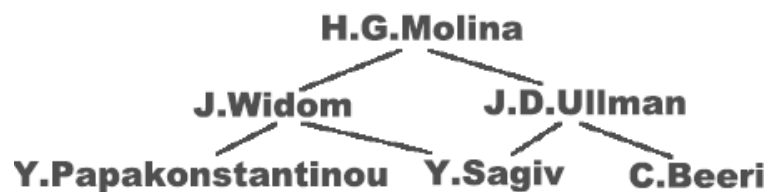


什么是图模式挖掘？

- 化学信息学。
 - 挖掘频繁的化学合成物结构



- 社会网络、网络社区、推特、...
 - 寻找频繁的研究合作子图

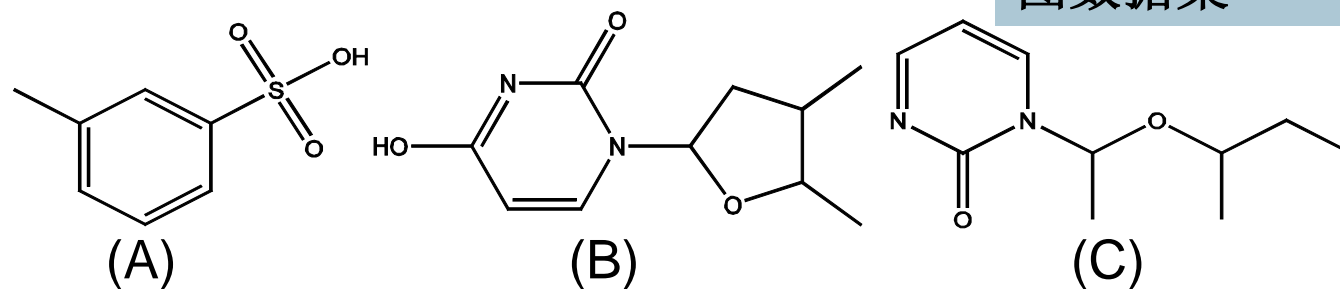


频繁的（子）图模式

- 给定一个标记图数据集 $D = \{G_1, G_2, \dots, G_n\}$ ，一个子图 g 的支持图集为 $D_g = \{G_i \mid g \subseteq G_i, G_i \in D\}$
 - 支持 $(g) = |D_g| / |D|$
 - 如果支持度 $(g) \geq \text{min_sup}$ ，一个(子)图 g 是**频繁的**。
 - 例：化学结构

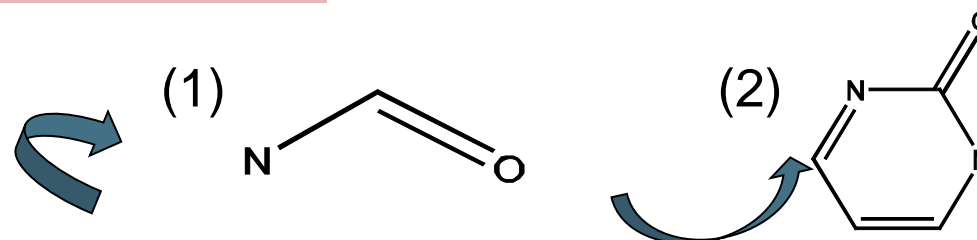
- 替代方案。

- 从单个大型图或网络中挖掘频繁子图模式



$\text{min_sup} = 2$

频繁出现的图模式



support=67%

图模式挖掘的应用

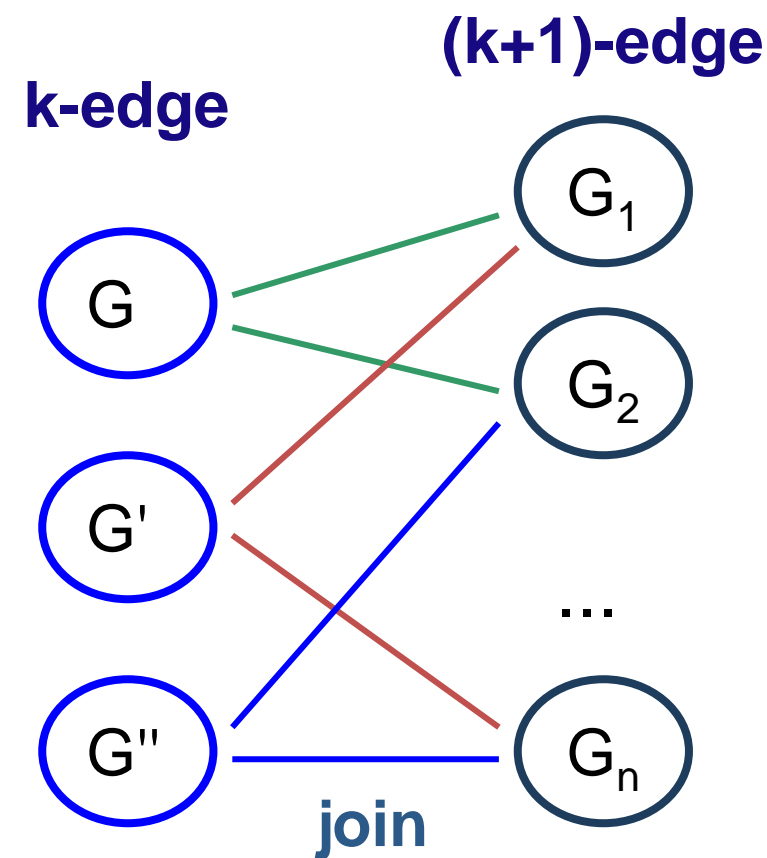
- 生物信息学
 - 基因网络、蛋白质相互作用、代谢途径
- 化学信息学。挖掘化学化合物结构
- 社会网络、网络社区、推特、...
- 手机网络、计算机网络、...
- 网络图、XML结构、语义网、信息网络
- 软件工程。程序执行流程分析
- 图分类、聚类、压缩、比较和相关分析的构建模块
- 图索引和图相似性搜索

图模式挖掘算法:不同的方法

- 候选子图的生成
 - Apriori vs. 模式增长（例如，FSG vs. gSpan）。
- 搜索顺序
 - 广度与深度
- 消除重复的子图
 - 被动与主动（例如，gSpan [Yan & Han, 2002]）。
- 支持度计算
 - 存储嵌入（例如，GASTON [Nijssen & Kok, 2004], FFSM [Huan, Wang, & Prins, 2003], MoFa [Borgelt & Berthold, ICDM'02]
- 发现模式的顺序
 - 路径→树→图（例如，GASTON [Nijssen & Kok, 2004]）。

基于Apriori的方法

- Apriori属性（反单调性）。一个大小为 k 的子图是频繁的，当且仅当它的所有子图都是频繁的。
- 如果一个候选的 $(k+1)$ 边/顶点子图对应的两个 k 边/顶点子图是频繁的，那么这个子图就会被生成。
- 迭代式挖掘过程。
 - 候选生成 → 候选修剪 → 支持计数 → 候选淘汰

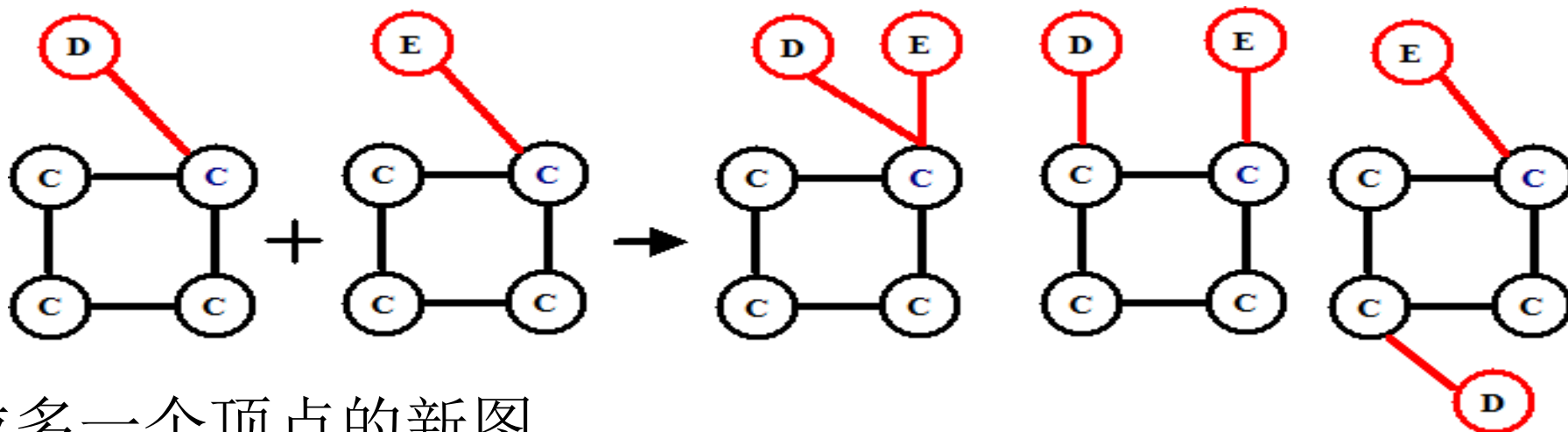


候选者生成。

顶点生长与边缘生长

□ 方法：广度搜索，Apriori连接两个 k 大小的图形

□ 产生大小 $(k+1)$ 候选图的多种可能性



- 生成多一个顶点的新图
 - AGM (Inokuchi, Washio, & Motoda, PKDD'00)
- 生成多一条边的新图
 - FSG (Kuramochi & Karypis, ICDM'01)
- 性能显示，*通过边生长的效率更高*

模式-增长的方法

□ 从 k 边到 $(k+1)$ 边的子图的深度优先增长，然后是 $(k+2)$ 边的子图

□ 主要挑战

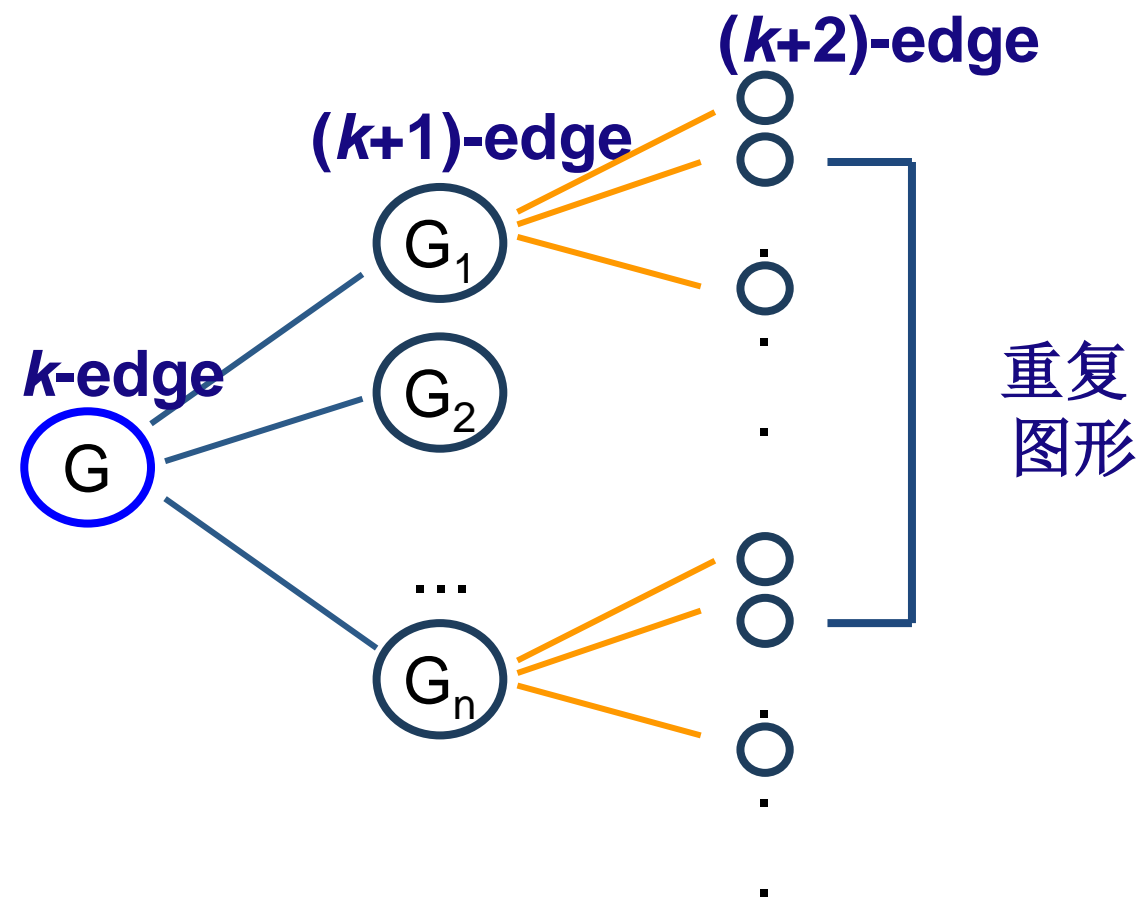
□ 生成许多重复的子图

□ 解决问题的主要想法

□ 定义一个生成子图的顺序

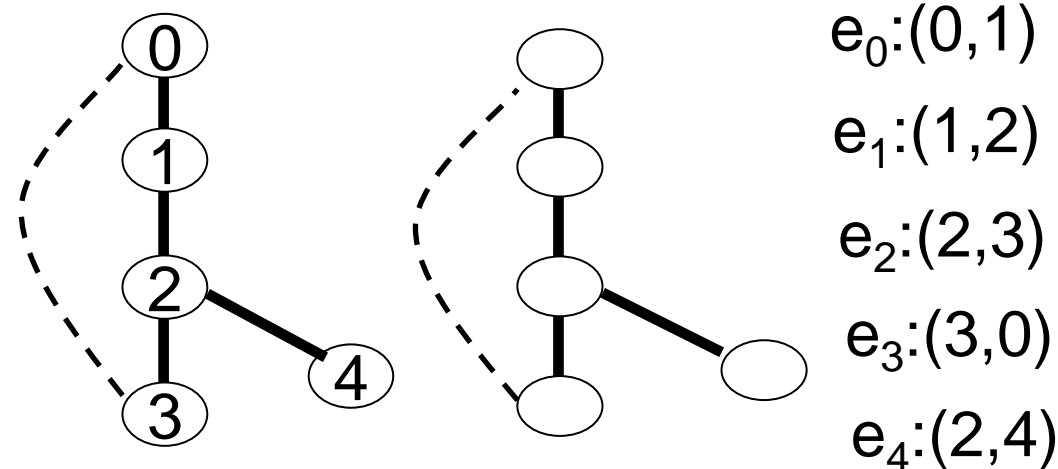
□ DFS生成树。使用深度优先搜索将一个图扁平化为一个序列

□ gSpan (Yan & Han, ICDM'02)



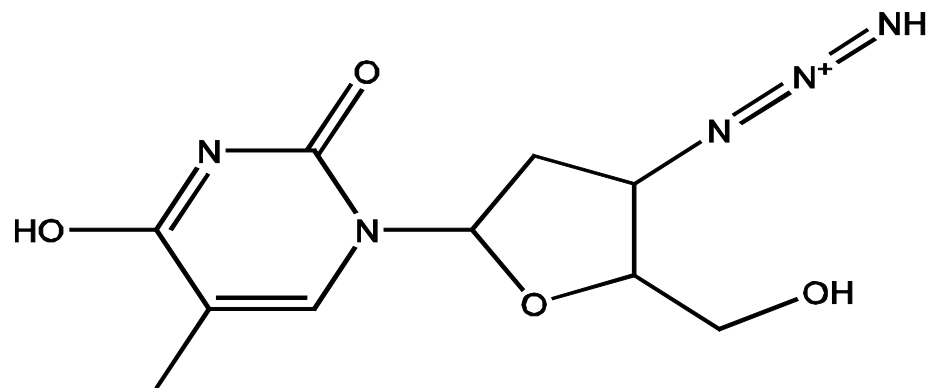
gSPAN: 图形模式按顺序增长

- 总的思想是，先生成频繁树，再在频繁树的基础上，生成频繁子图，满足最小支持度，满足最小DFS编码的所有频繁子图。
- 子图模式增长中的**最右路径扩展**
- 所有节点根据发现时间排序，最后发现的节点叫做**最右节点**。从第一个节点到最右节点的直线路径叫做**最右路径**
 - 最右边的路径。从根到最右边的叶子的路径（在每一步选择索引最小的顶点）
 - 减少重复子图的生成
- 完备性**。使用最右路径扩展的图的枚举是完整的
- DFS编码**：使用深度优先搜索将一个图扁平化为一个序列



为什么要挖掘封闭图式？

- 挑战：一个有 n 个边的频繁图可能有 2^n 个子图
- 动机：探索*封闭的*频繁子图以处理图型爆炸问题
- 如果不存在与 G 相同支持度的 G 的超图，那么一个频繁图 G 就是*封闭的*。

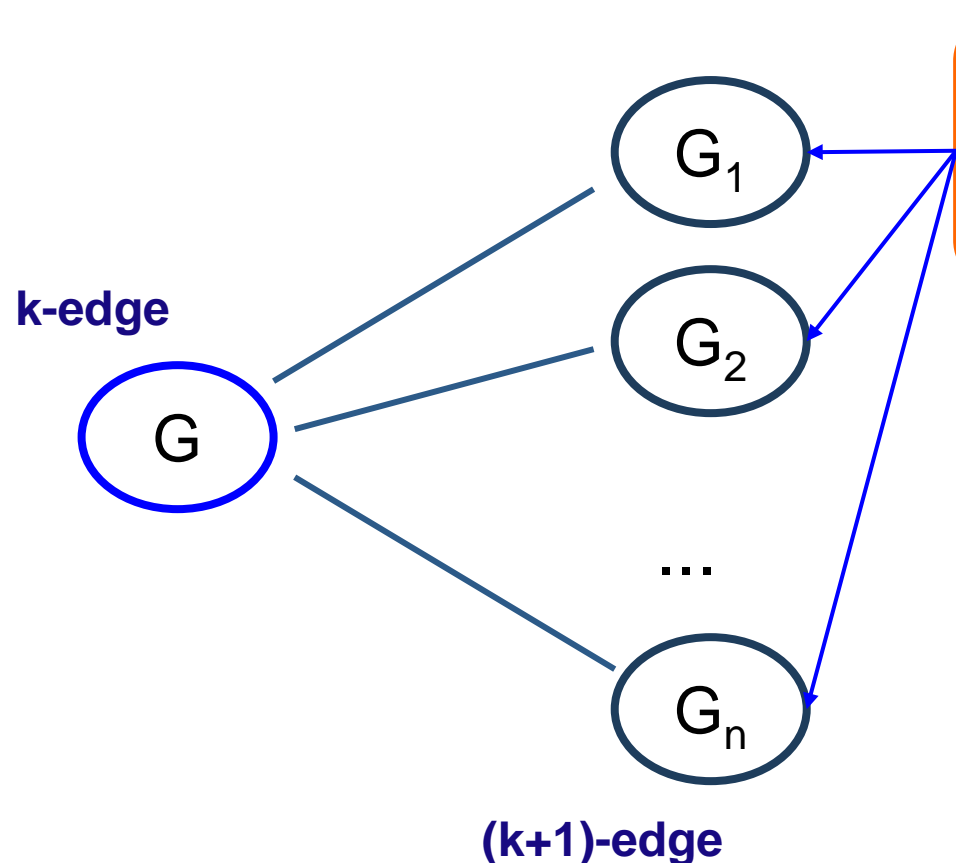


如果这个子图在图形数据集中是*封闭的*，这意味着它的频繁超图中没有一个具有相同的支持度。

- *无损压缩*。不包含非封闭图，但仍能确保挖掘结果是完整的
- 算法**CloseGraph**。直接挖掘封闭图的模式

CloseGraph:直接挖掘封闭图的模式

- CloseGraph:通过扩展gSpan来挖掘封闭图模式（Yan & Han, KDD'03）。



在什么条件下我们可以
停止搜查他们的子图。
即：提前终止？

- 假设 G 和 G_1 是频繁的，并且 G 是 G_1 的一个子图
- 如果在数据集的图中中 G 出现的任何部分， G_1 也出现了，那么我们就需要增长 G （除了一些特殊的、微妙的情况），因为除了 G_1 的后代， G 的其他后代都不会是闭得。

第7章 :高级模式挖掘

- 多层、多维空间中的模式挖掘
- 序列模式挖掘
- 基于约束的频繁模式挖掘
- 图模式挖掘
- 模式挖掘应用
- 摘要



模式挖掘应用： 软件bug检测

- 从源代码中挖掘规则
 - Bug是异常的行为（例如，通过统计分析）。
 - 挖掘编程规则（例如，通过频繁项集挖掘）。
 - 挖掘函数优先级协议（例如，通过频繁子序列挖掘）。
 - 揭示被忽视的条件（例如，通过频繁项集/子图挖掘）。
- 从修订历史中挖掘规则
 - 通过频繁项集挖掘
- 从源代码中挖掘复制-粘贴模式
 - 寻找复制粘贴的错误（例如CP-Miner [Li et al., OSDI'04]）
 - 参考文献。Z. Li, S. Lu, S. Myagmar, Y. Zhou, "CP-Miner: A Tool for Finding Copy-paste and Related Bugs in Operating System Code", OSDI'04

应用实例：挖掘复制和粘贴的错误

- ❑ 复制粘贴很常见
 - ❑ 在Linux文件系统中占12%。
 - ❑ 19%在windows系统中
- ❑ 复制粘贴的代码很容易出错
- ❑ 通过顺序模式挖掘来挖掘 "忘记改变" 的错误
 - ❑ 从源代码建立一个序列数据库
 - ❑ 挖掘顺序模式
 - ❑ 查找不匹配的标识符名称和错误

```
void __init prom_meminit(void)
{
    .....
    for (i=0; i<n; i++) {
        total[i].adr = list[i].addr。
        total[i].bytes = list[i].size。
        total[i].more = & total[i+1];
    }
    .....
```

```
for (i=0; i<n; i++) {
    taken[i].adr = list[i].addr。
    taken[i].byte = list[i].size。
    taken[i].more = & total[i+1];
}
```

代码复制和粘贴，
但忘记改变 "id"!

(简化的例子来自 *linux-2.6.6/arch/sparc/prom/memory.c*)

第7章 :高级模式挖掘

- 多层、多维空间中的模式挖掘
- 序列模式挖掘
- 基于约束的频繁模式挖掘
- 图模式挖掘
- 模式挖掘应用
- 摘要



摘要：高级频繁模式挖掘

- 挖掘多样化的模式
 - 挖掘多层次的关联
 - 挖掘多维关联
 - 挖掘定量关联
 - 挖掘负相关关系
 - 挖掘压缩和冗余意识的模式
- 序列模式挖掘
 - 序列图案和序列图案挖掘
 - **GSP**: 基于**Apriori**的顺序模式挖掘
 - **SPADE**: 垂直数据格式的顺序模式挖掘
 - **PrefixSpan**: 通过模式增长的顺序模式挖掘
 - **CloSpan**: 挖掘封闭的序列模式
- 基于约束的频繁模式挖掘
 - 为什么需要基于约束的挖掘?
 - 带有模式反单调性的约束性挖掘
 - 带有模式单调性的约束性挖掘
 - 带有数据反单调性的约束性挖掘
 - 带有简洁约束的约束性挖掘
 - 可转换约束条件下的约束性挖掘
 - 处理多个约束条件
 - 基于约束的顺序模式挖掘
- 图形模式挖掘
 - 图模式和图模式挖掘
 - 基于**Apriori**的图模式挖掘方法
 - **gSpan**: 一种基于模式增长的方法
 - **CloseGraph**: 挖掘封闭图的模式
- 模式挖掘应用：挖掘软件复制和粘贴的错误

References: Mining Diverse Patterns

- R. Srikant and R. Agrawal, "Mining generalized association rules", VLDB'95
- Y. Aumann and Y. Lindell, "A Statistical Theory for Quantitative Association Rules", KDD'99
- K. Wang, Y. He, J. Han, "Pushing Support Constraints Into Association Rules Mining", IEEE Trans. Knowledge and Data Eng. 15(3): 642-658, 2003
- D. Xin, J. Han, X. Yan and H. Cheng, "On Compressing Frequent Patterns", Knowledge and Data Engineering, 60(1): 5-29, 2007
- D. Xin, H. Cheng, X. Yan, and J. Han, "Extracting Redundancy-Aware Top-K Patterns", KDD'06
- J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent Pattern Mining: Current Status and Future Directions", Data Mining and Knowledge Discovery, 15(1): 55-86, 2007
- F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng, "Mining Colossal Frequent Patterns by Core Pattern Fusion", ICDE'07

References: Sequential Pattern Mining

- R. Srikant and R. Agrawal, “Mining sequential patterns: Generalizations and performance improvements”, EDBT’96
- M. Zaki, “SPADE: An Efficient Algorithm for Mining Frequent Sequences”, Machine Learning, 2001
- J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, “Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach”, IEEE TKDE, 16(10), 2004
- X. Yan, J. Han, and R. Afshar, “CloSpan: Mining Closed Sequential Patterns in Large Datasets”, SDM’03
- J. Pei, J. Han, and W. Wang, “Constraint-based sequential pattern mining: the pattern-growth methods”, J. Int. Inf. Sys., 28(2), 2007
- M. N. Garofalakis, R. Rastogi, K. Shim: Mining Sequential Patterns with Regular Expression Constraints. IEEE Trans. Knowl. Data Eng. 14(3), 2002
- H. Mannila, H. Toivonen, and A. I. Verkamo, “Discovery of frequent episodes in event sequences”, Data Mining and Knowledge Discovery, 1997

References: Constraint-Based Frequent Pattern Mining

- R. Srikant, Q. Vu, and R. Agrawal, “Mining association rules with item constraints”, KDD'97
- R. Ng, L.V.S. Lakshmanan, J. Han & A. Pang, “Exploratory mining and pruning optimizations of constrained association rules”, SIGMOD'98
- G. Grahne, L. Lakshmanan, and X. Wang, “Efficient mining of constrained correlated sets”, ICDE'00
- J. Pei, J. Han, and L. V. S. Lakshmanan, “Mining Frequent Itemsets with Convertible Constraints”, ICDE'01
- J. Pei, J. Han, and W. Wang, “Mining Sequential Patterns with Constraints in Large Databases”, CIKM'02
- F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi, “ExAnte: Anticipated Data Reduction in Constrained Pattern Mining”, PKDD'03
- F. Zhu, X. Yan, J. Han, and P. S. Yu, “gPrune: A Constraint Pushing Framework for Graph Pattern Mining”, PAKDD'07

References: Graph Pattern Mining

- C. Borgelt and M. R. Berthold, Mining molecular fragments: Finding relevant substructures of molecules, ICDM'02
- J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism, ICDM'03
- A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data, PKDD'00
- M. Kuramochi and G. Karypis. Frequent subgraph discovery, ICDM'01
- S. Nijssen and J. Kok. A Quickstart in Frequent Structure Mining can Make a Difference. KDD'04
- N. Vanetik, E. Gudes, and S. E. Shimony. Computing frequent graph patterns from semistructured data, ICDM'02
- X. Yan and J. Han, gSpan: Graph-Based Substructure Pattern Mining, ICDM'02
- X. Yan and J. Han, CloseGraph: Mining Closed Frequent Graph Patterns, KDD'03
- X. Yan, P. S. Yu, J. Han, Graph Indexing: A Frequent Structure-based Approach, SIGMOD'04
- X. Yan, P. S. Yu, and J. Han, Substructure Similarity Search in Graph Databases, SIGMOD'05