关联分析：基本概念和算法

电子信息与通信学院 冯 镔
fengbin@mail.hust.edu.cn

关联规则挖掘

- 给定一系列记录，找到其中隐含的令人感兴趣的联系，用以根据记录中某些项的出现来预测其他项的产生

商场购物篮事务

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

关联规则的例子

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

关联意味着同时出现，而并非因果关系

关联规则挖掘

- 一、从大型事务数据集中发现模式可能在计算上要付出很高的代价
  - 有效挖掘模式的算法
- 二、所发现的某些模式可能是虚假的，因为它们可能是偶然发生的
  - 模式的评估问题

关联规则挖掘

- 二元表示
  - 非对称二元变量
- 项集
  - 包含0个或多个项的集合
  - K-项集
- 支持度计数
  - 包含特定项集的事务个数

$$\sigma(X) = |\{t_i \mid X \subseteq t_i, t_i \in T\}|$$

  - σ({牛奶, 面包,尿布}) = 2

TID	面包	牛奶	尿布	啤酒	鸡蛋	可乐
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

关联规则挖掘

- 关联规则
  - 形如X->Y的蕴含表达式，其中X和Y是不相交的项集
  - 如：{牛奶, 尿布} → {啤酒}

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- 规则评估度量
  - 支持度(s)：规则可以用于给定数据集的频繁程度

$$s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$$

s(牛奶, 尿布->啤酒)=0.4

  - 置信度(c)：Y在包含X的事务中出现的频繁程度

$$c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

c(牛奶, 尿布->啤酒)=0.67

关联规则挖掘

- 问题描述：给定事务的集合T，关联规则发现是指找出支持度大于等于**minsup**并且置信度大于等于**minconf**的所有规则
  - **minsup**是支持度阈值
  - **minconf**是置信度阈值
- 最直接的方法：计算每个可能规则的支持度和置信度，排除不满足条件的规则
  - 可能的规则总数为$R=3^d-2^{d+1}+1$

关联规则挖掘

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Rules:

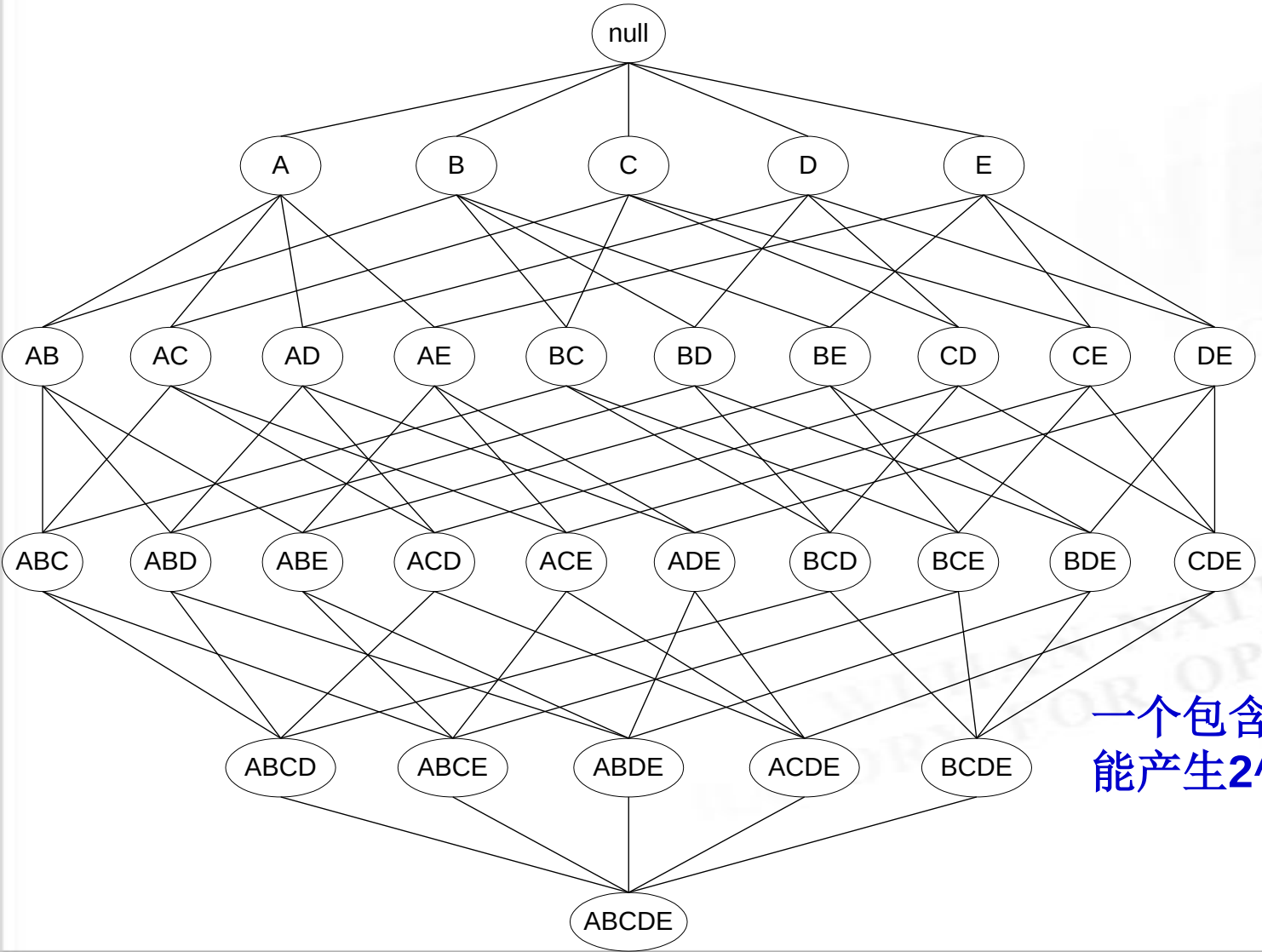
{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

- 上述规则涉及的项来自同一个项集 {牛奶, 尿布, 啤酒}
- 从同一个项集中推导出的规则必然具有相同的支持度，置信度可能不同
- 可将支持度和置信度两个任务分解

关联规则挖掘

- 两步法：
- 1. 频繁项集产生：目标是发现满足最小支持度阈值的所有项集，称作频繁项集
- 2. 规则的产生：从发现的频繁项集中提取所有高置信度的规则，称为强规则
- 频繁项集产生所需的计算开销远大于规则的产生所需的计算开销

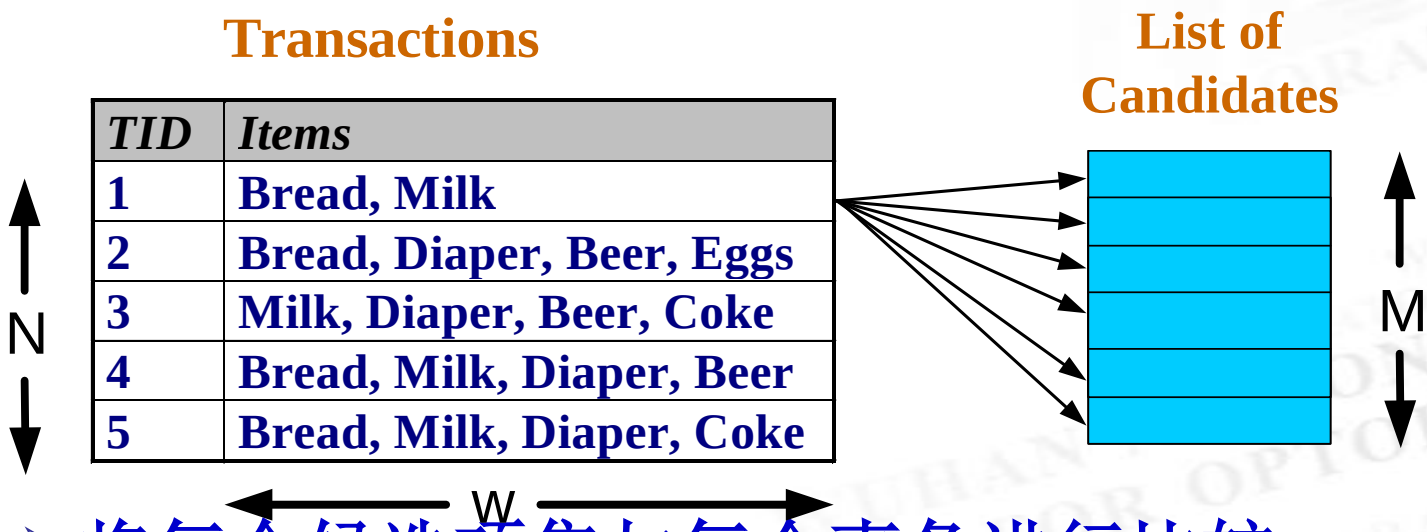
频繁项集产生



一个包含k个项的数据集可能产生2^k个候选项集

频繁项集产生

- **Brute-force法：**
  - 将每个格中的项集作为候选频繁项集
  - 计算每个候选项集的支持度



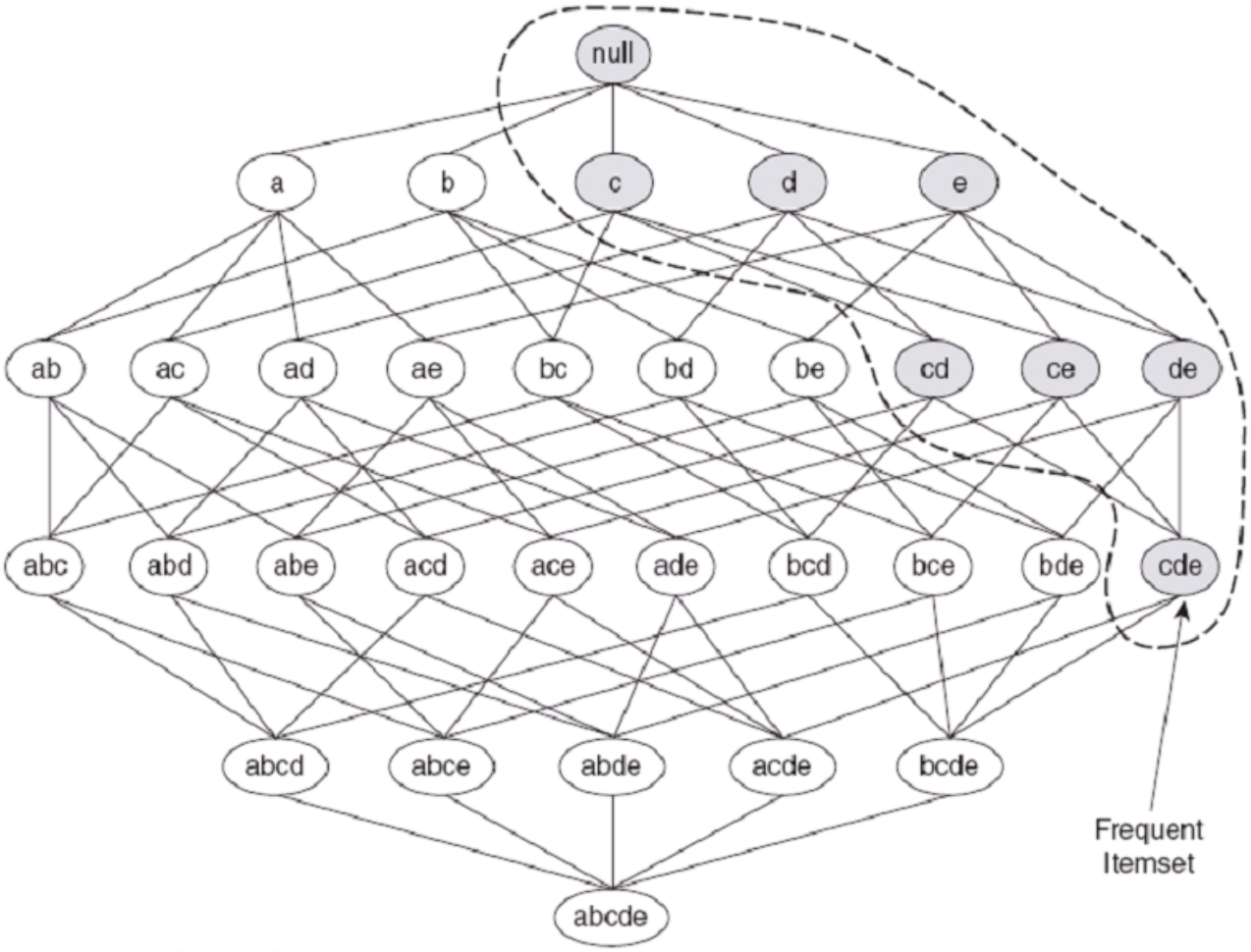
  - 将每个候选项集与每个事务进行比较
  - 复杂度**O(NMw)**

频繁项集产生

- 减少候选项集的数目(M)
  - 全搜索：$M=2^d$
  - 利用剪枝技术减小M
- 减少比较次数(NM)
  - 使用更高级的数据结构存储候选项集或事务集
  - 不需要在每个候选集和事务之间匹配

频繁项集产生

- 先验原理：如果一个项集是频繁的，则它的所有子集一定也是频繁的



频繁项集产生

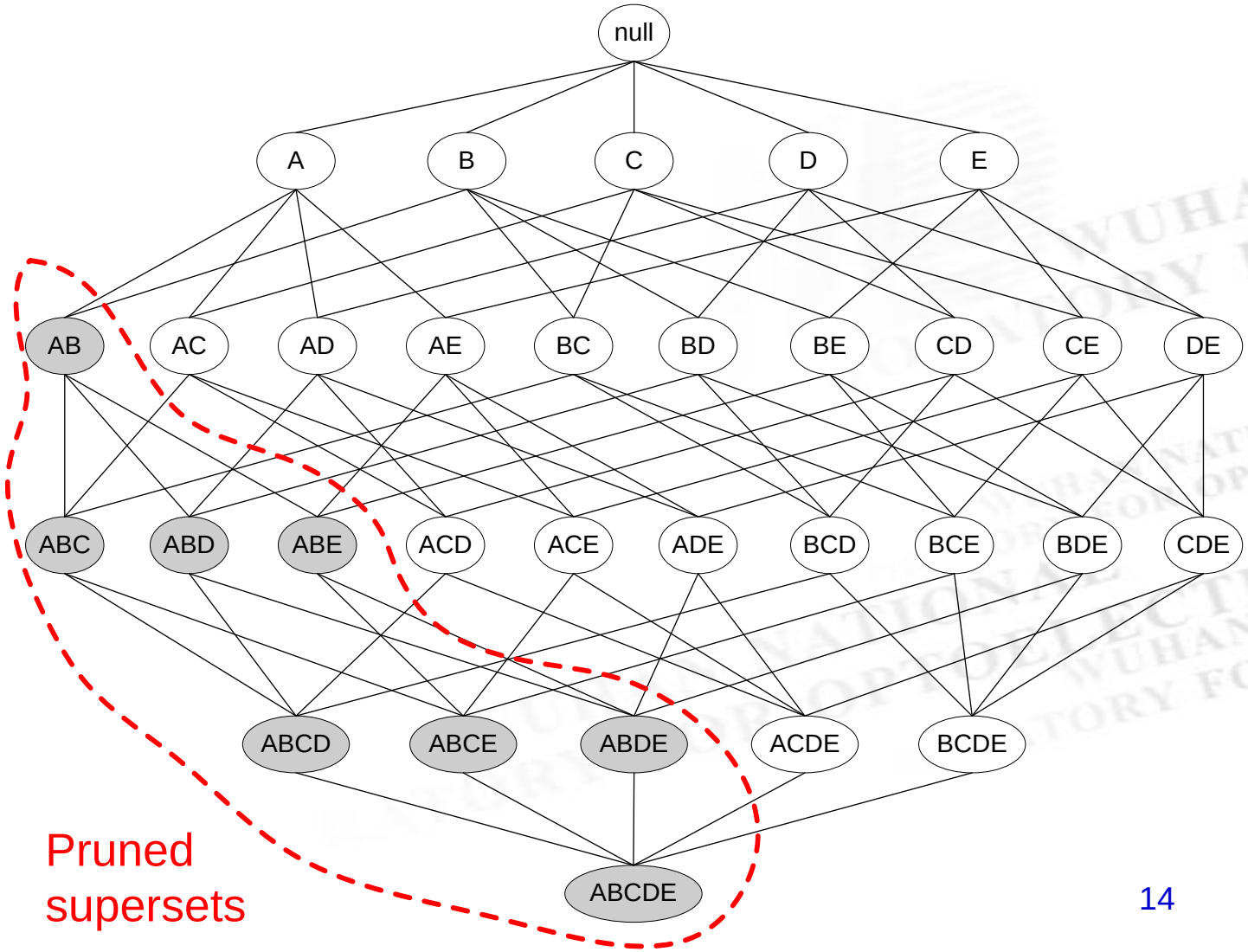
- 相反，如果一个项集是非频繁的，则它的所有超集也一定是非频繁的
- 基于支持度度量的一个关键性质：一个项集的支持度不会超过它的子集的支持度

$$\forall X,Y:(X\subseteq Y)\Rightarrow s(X)\geq s(Y)$$

  - 反单调性

频繁项集产生

- 基于支持度的剪枝



TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Items (1-itemsets)

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

If every subset is considered,
$C^6_1 + C^6_2 + C^6_3 = 41$
With support-based pruning,
$6 + 6 + 1 = 13$

关联规则挖掘

- 习题3
- (a)规则 空集→A和A→空集的置信度是多少
- (b)令c1, c2和c3分别是规则{p}→{q}， {p}→{q, r}和{p, r}→{q}的置信度，它们之间可能存在什么关系
- (c)假定(b)中的规则具有相同的支持度，重复(b)的分析
- (d)传递性：假定规则A→B和B→C的置信度都大于阈值minconf，规则A→C可能具有小于minconf的置信度吗

频繁项集产生

Apriori 算法的频繁项集产生

1: k=1

2: $F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup \}$ {发现所有的频繁 1-项集}

3: repeat

4: k=k+1

5: C_k = apriori - gen(F_{k-1}) {产生候选项集}

6: for 每个事务 $t \in T$ do

7: C_t = subset(C_k, t) {识别属于 t 的所有候选}

8: for 每个候选项集 $c \in C_t$ do

9: $\sigma(c) = \sigma(c) + 1$ {支持度计数增值}

10: end for

11: end for

12: $F_k = \{ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup \}$ {提取频繁 k-项集}

13: until F_k = 空集

14: Result = U F_k

频繁项集产生

- Apriori-gen函数通过如下两个操作产生候选项集：
- 1. 候选项集的产生
  - 由前一次迭代发现的频繁(k-1)-项集产生新的候选k-项集
- 2. 候选项集的剪枝
  - 采用基于支持度的剪枝策略，删除一些候选k-项集

频繁项集产生

- **Brute-forth法**

Items

Item
Beer
Bread
Cola
Diapers
Milk
Eggs

→

Candidate Generation

Itemset
{Beer, Bread, Cola}
{Beer, Bread, Diapers}
{Beer, Bread, Milk}
{Beer, Bread, Eggs}
{Beer, Cola, Diapers}
{Beer, Cola, Milk}
{Beer, Cola, Eggs}
{Beer, Diapers, Milk}
{Beer, Diapers, Eggs}
{Beer, Milk, Eggs}
{Bread, Cola, Diapers}
{Bread, Cola, Milk}
{Bread, Cola, Eggs}
{Bread, Diapers, Milk}
{Bread, Diapers, Eggs}
{Bread, Milk, Eggs}
{Cola, Diapers, Milk}
{Cola, Diapers, Eggs}
{Cola, Milk, Eggs}
{Diapers, Milk, Eggs}

→

Candidate Pruning

Itemset
{Bread, Diapers, Milk}

频繁项集产生

- $F_{k-1} \times F_1$法

1. 所有的频繁k-项集是这种方法所产生的候选k-项集的一部分
2. 以字典序存储频繁项集，以避免产生重复候选项集

Frequent 2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

Frequent 1-itemset

Item
Beer
Bread
Diapers
Milk

Candidate Generation

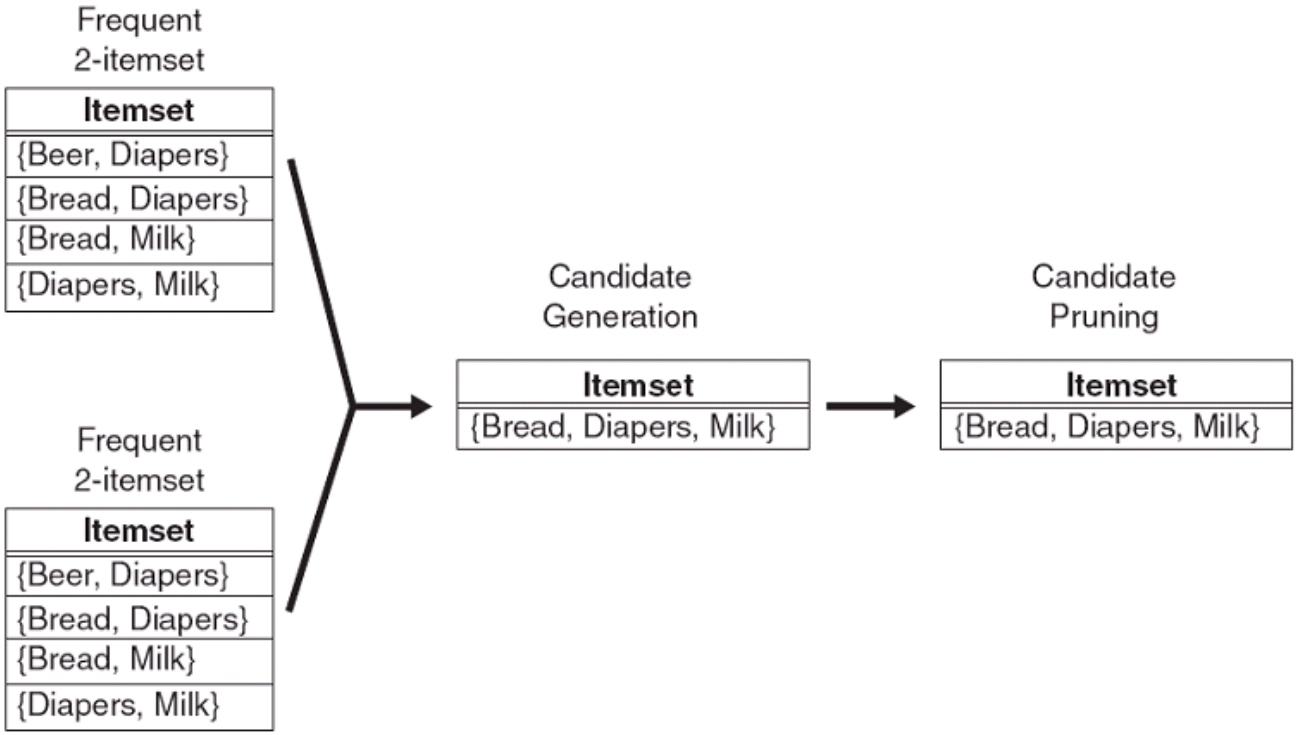
Itemset
{Beer, Diapers, Bread}
{Beer, Diapers, Milk}
{Bread, Diapers, Milk}
{Bread, Milk, Beer}

Candidate Pruning

Itemset
{Bread, Diapers, Milk}

频繁项集产生

- $F_{k-1} \times F_{k-1}$法
  - 合并一对频繁k-1项集，仅当它们的前k-2个项集都相同



频繁项集产生

- 习题7
- 考虑下面的频繁3-项集的集合：

{1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}, {2,3,4}, {2,3,5}, {3,4,5}，假定数据集中只有5个项。

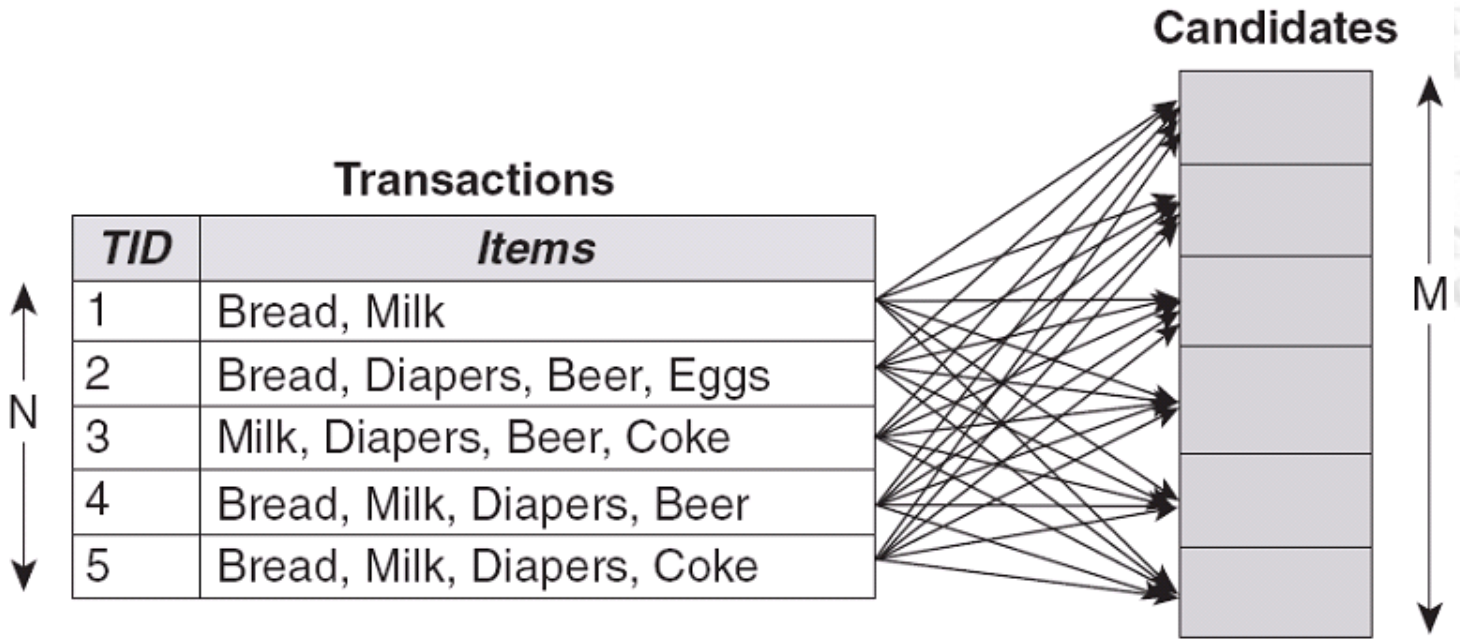
(a)列出采用$F_{k-1} \times F_1$合并策略，由候选产生过程得到的所有候选4-项集

(b)列出由Apriori算法的候选产生过程得到的所有候选4-项集

(c)列出Apriori算法候选剪枝步骤后剩下的所有候选4-项集

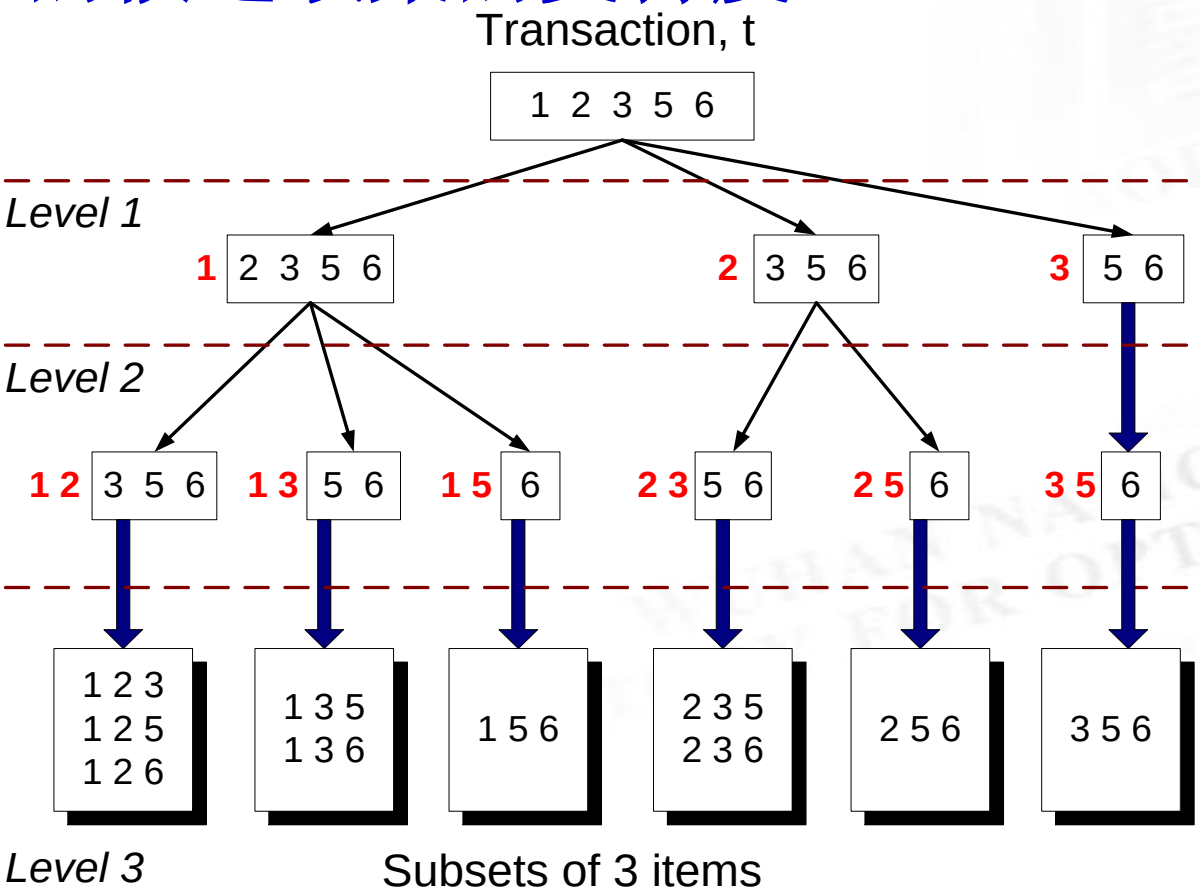
支持度计数

- 确定在候选项剪枝步骤保留下来的每个候选项集出现的频繁程度
- 方法1：将每个事务与所有的候选项集进行比较，并更新包含在事务中的候选项集支持度计数



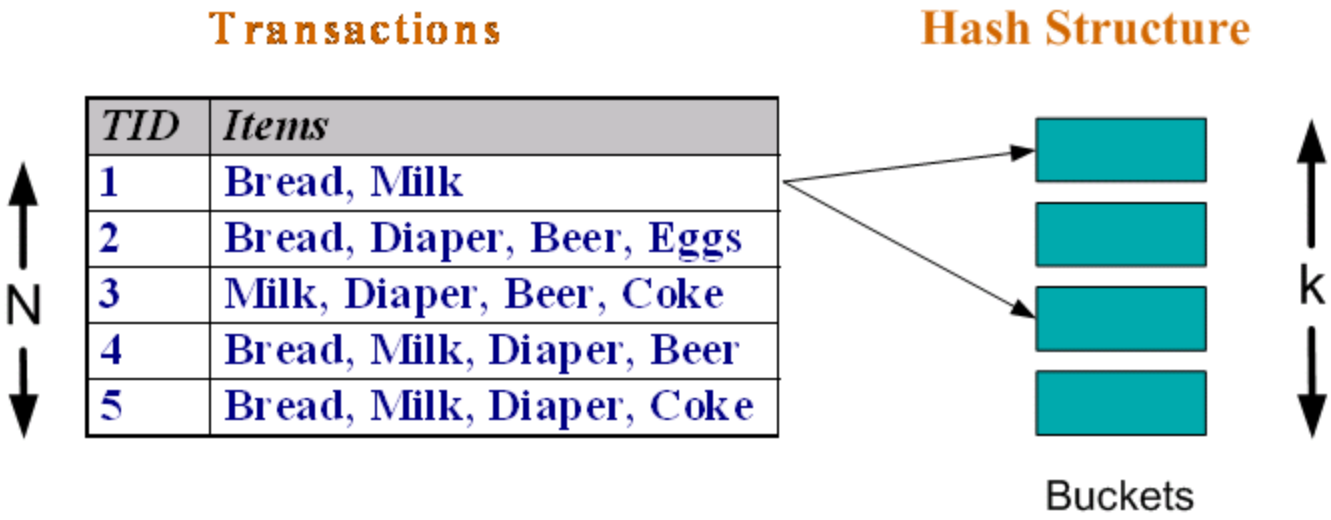
支持度计数

- 方法2：枚举每个事务所包含的项集，并更新对应的候选项集的支持度

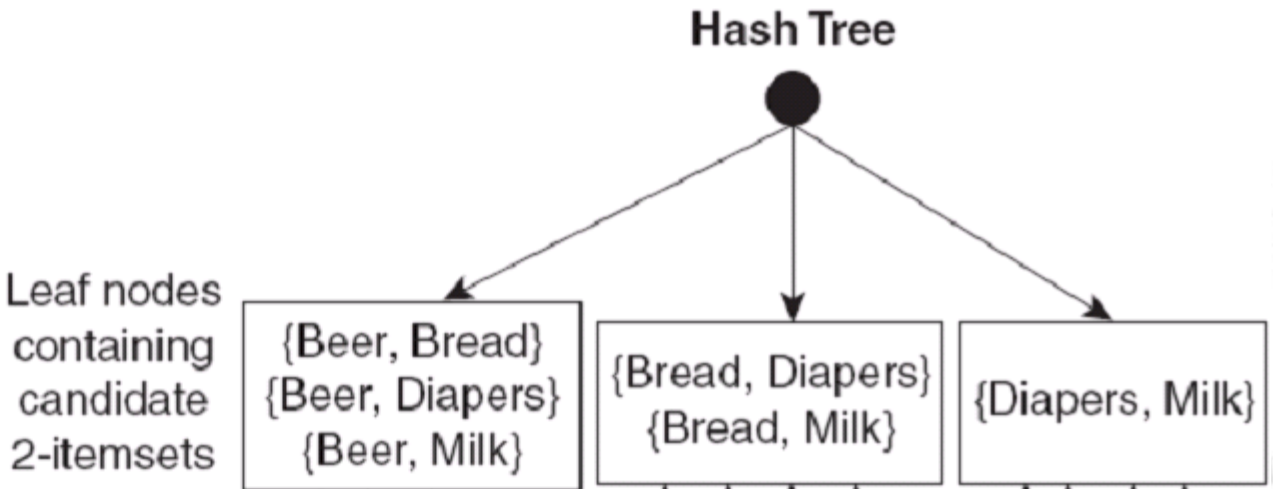


支持度计数

- 将候选项集划分，以hash树结构存储
- 包含在事务中的项集也散列到相应的桶中
- 不是将事务中的每个项集与所有的候选项集进行比较，而是与处于同一桶内的候选项集进行匹配



支持度计数



Transactions

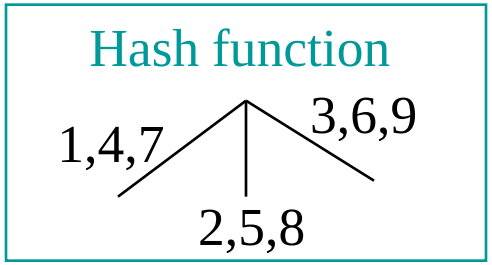
TID	Items
1	Bread, Milk
2	Bread, Diapers, Beer, Eggs
3	Milk, Diapers, Beer, Cola
4	Bread, Milk, Diapers, Beer
5	Bread, Milk, Diapers, Cola

生成hash树

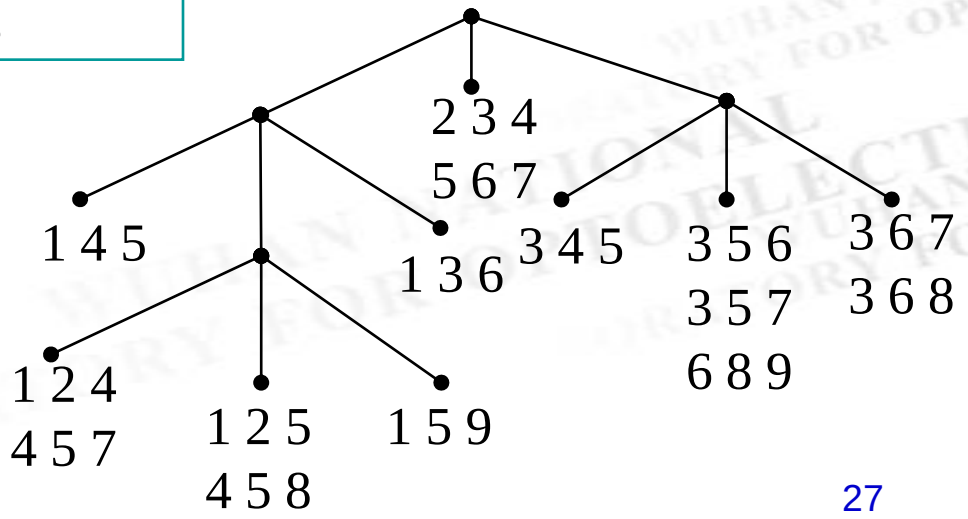
- 假设有15个候选3-项集

{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

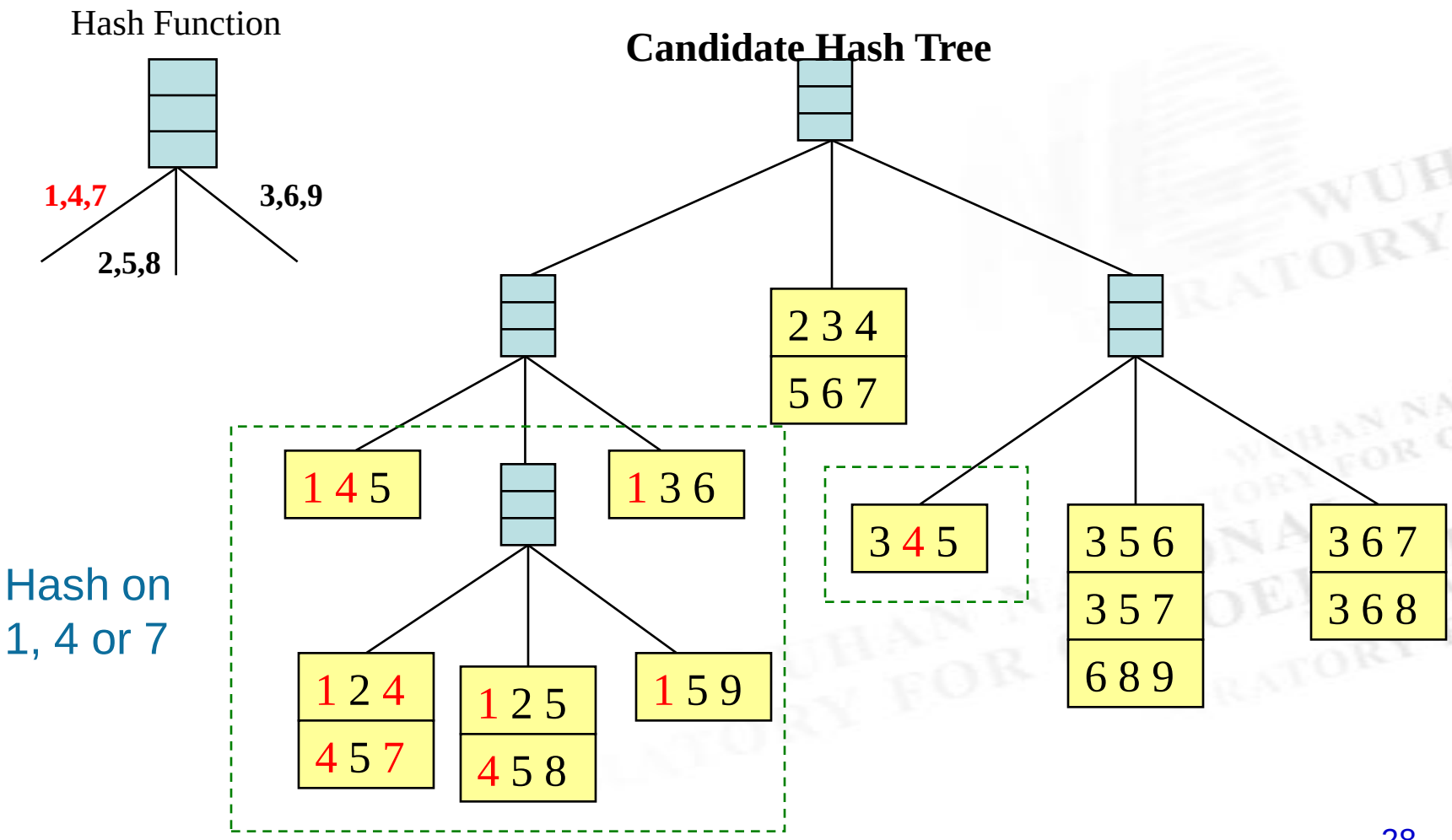
- Hash函数



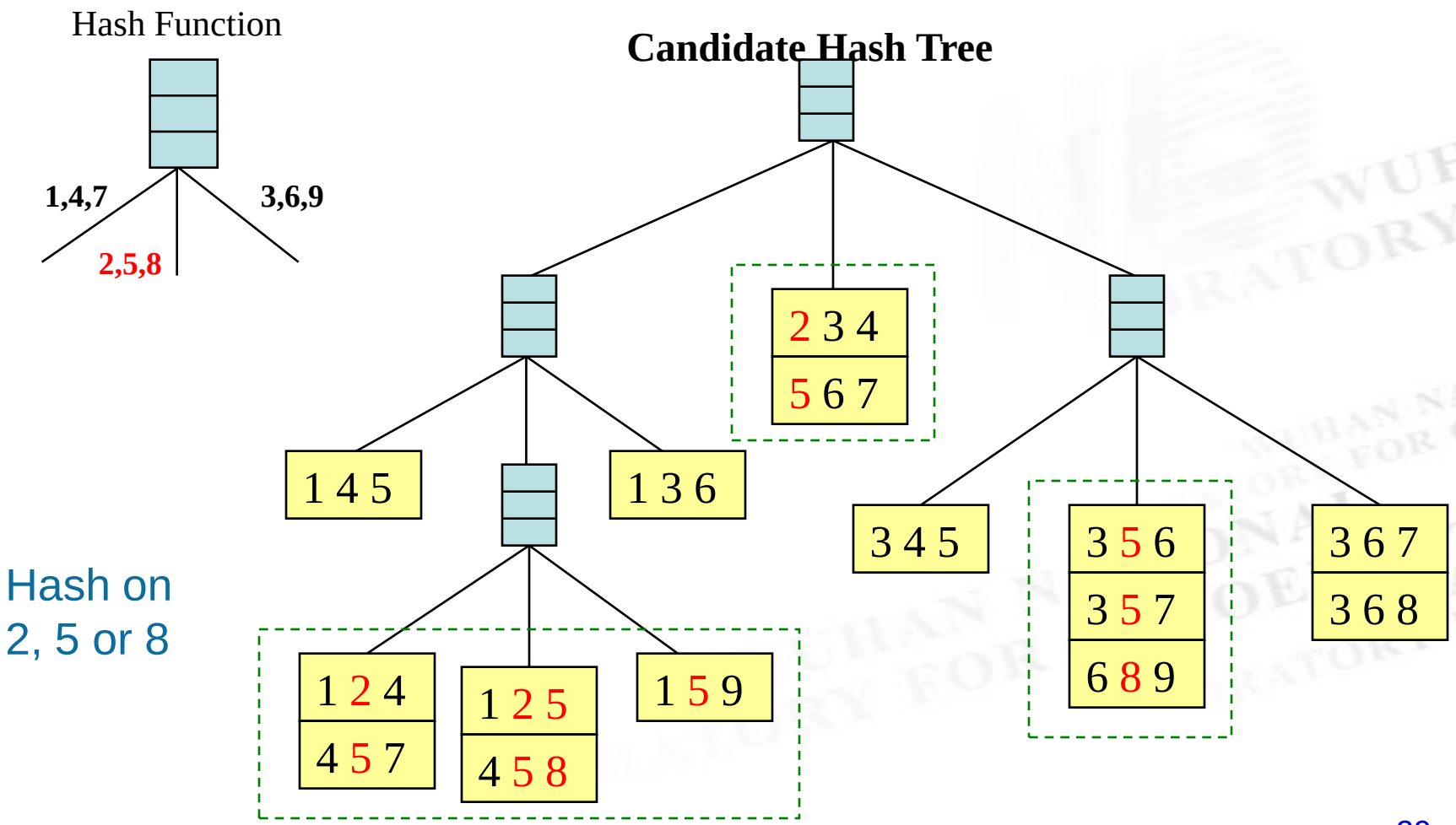
- 最大叶大小
  - 叶结点存储项的最大个数



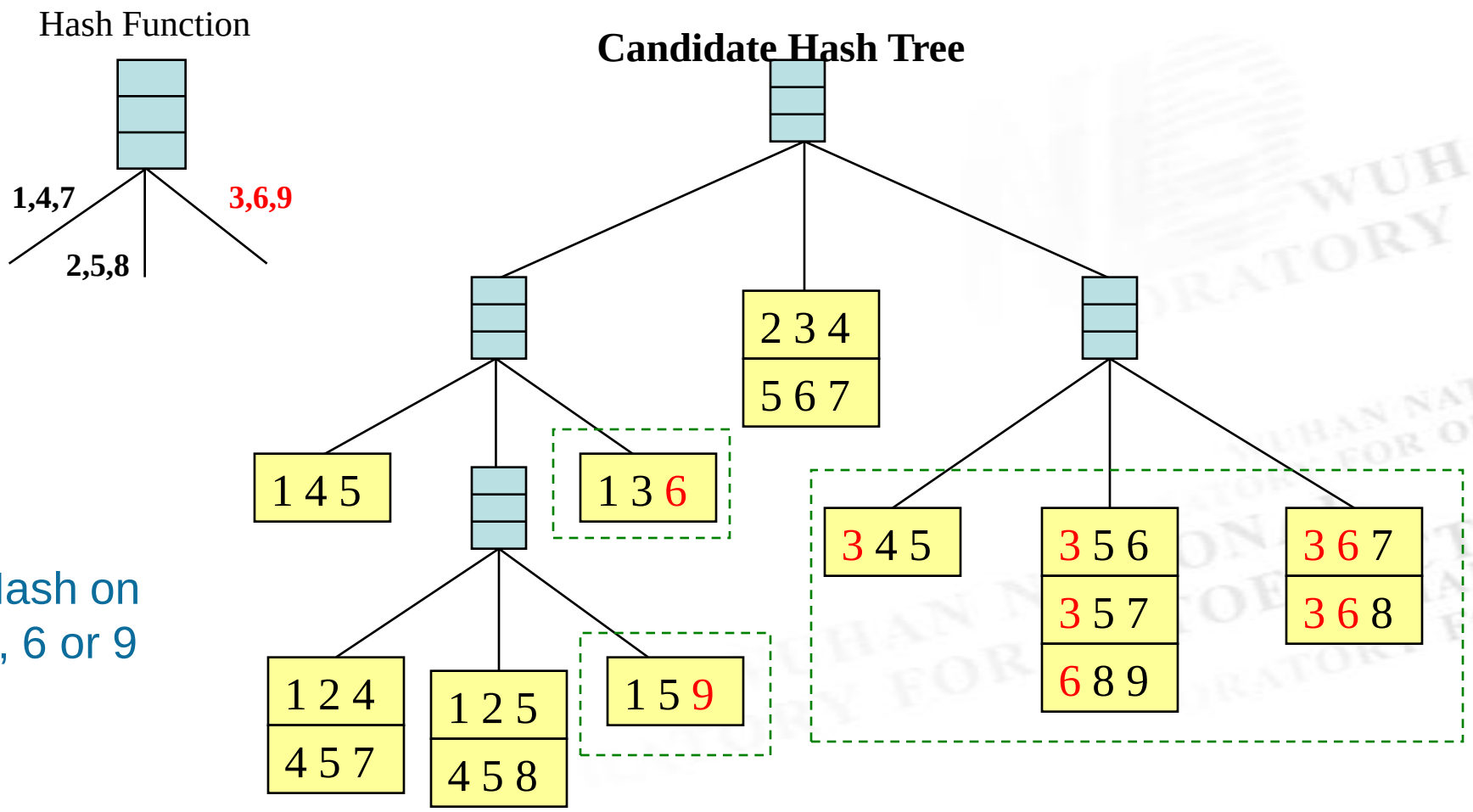
Hash树


Hash Function
1,4,7
2,5,8
3,6,9
Candidate Hash Tree
1 4 5
1 3 6
2 3 4
5 6 7
3 4 5
3 5 6
3 5 7
6 8 9
3 6 7
3 6 8
1 2 4
4 5 7
1 2 5
4 5 8
1 5 9
Hash on
1, 4 or 7

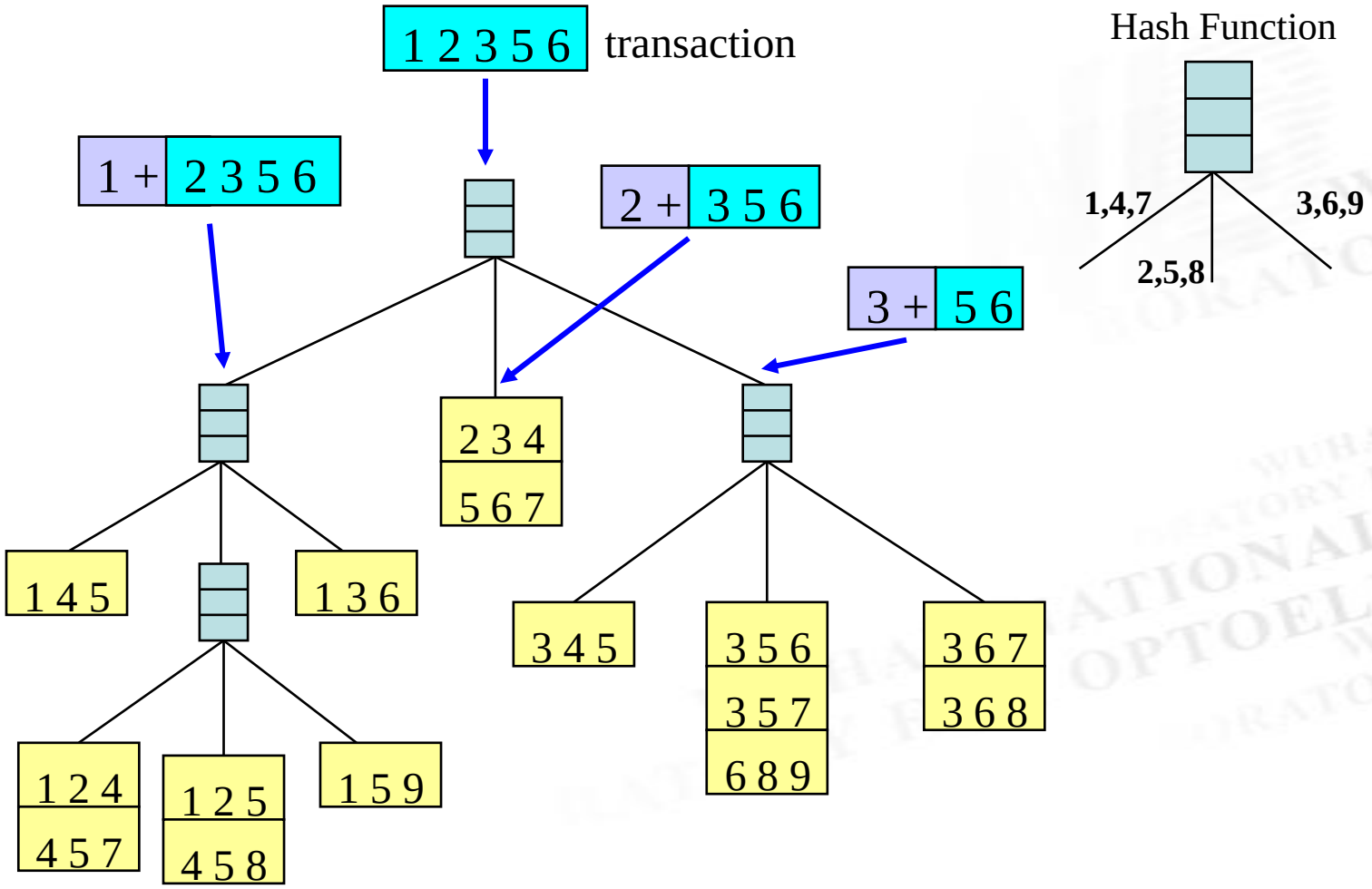
Hash树


Hash Function
1,4,7
2,5,8
3,6,9
Candidate Hash Tree
1 4 5
1 3 6
2 3 4
5 6 7
3 4 5
3 5 6
3 5 7
6 8 9
3 6 7
3 6 8
1 2 4
4 5 7
1 2 5
4 5 8
1 5 9
Hash on
2, 5 or 8

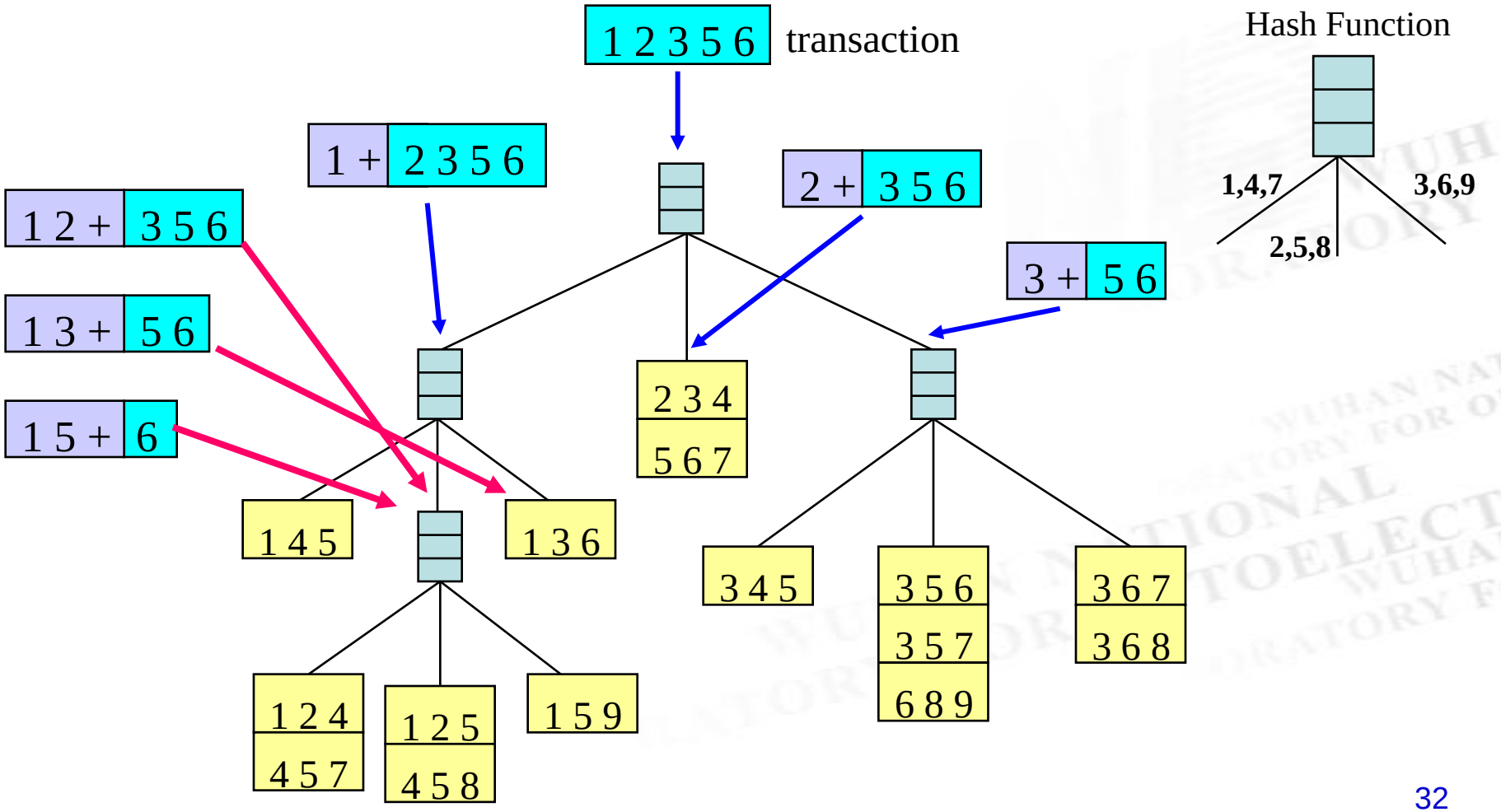
Hash树


Hash Function
1,4,7
2,5,8
3,6,9
Candidate Hash Tree
1 4 5
1 3 6
2 3 4
5 6 7
1 2 4
4 5 7
1 2 5
4 5 8
1 5 9
3 4 5
3 5 6
3 5 7
6 8 9
3 6 7
3 6 8
Hash on
3, 6 or 9

Hash树的子集操作

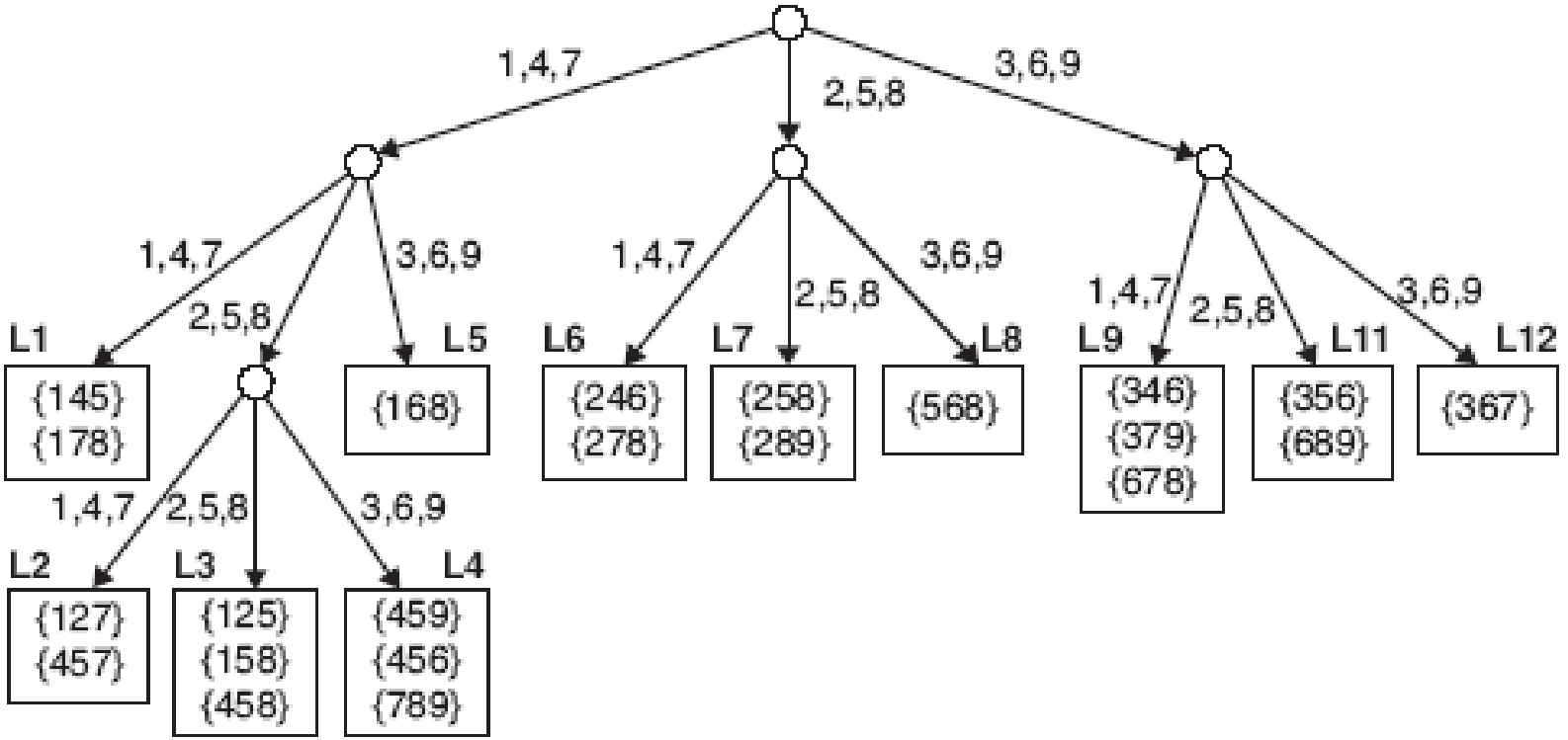

1 2 3 5 6
transaction
1 +
2 3 5 6
2 +
3 5 6
3 +
5 6
2 3 4
5 6 7
1 4 5
1 3 6
3 4 5
3 5 6
3 5 7
6 8 9
3 6 7
3 6 8
1 2 4
4 5 7
1 2 5
4 5 8
1 5 9
Hash Function
1,4,7
2,5,8
3,6,9

Hash树的子集操作


1 2 3 5 6
transaction
Hash Function
1,4,7
2,5,8
3,6,9
1 +
2 3 5 6
2 +
3 5 6
3 +
5 6
1 2 +
3 5 6
1 3 +
5 6
1 5 +
6
2 3 4
5 6 7
1 4 5
1 3 6
3 4 5
3 5 6
3 5 7
6 8 9
3 6 7
3 6 8
1 2 4
4 5 7
1 2 5
4 5 8
1 5 9

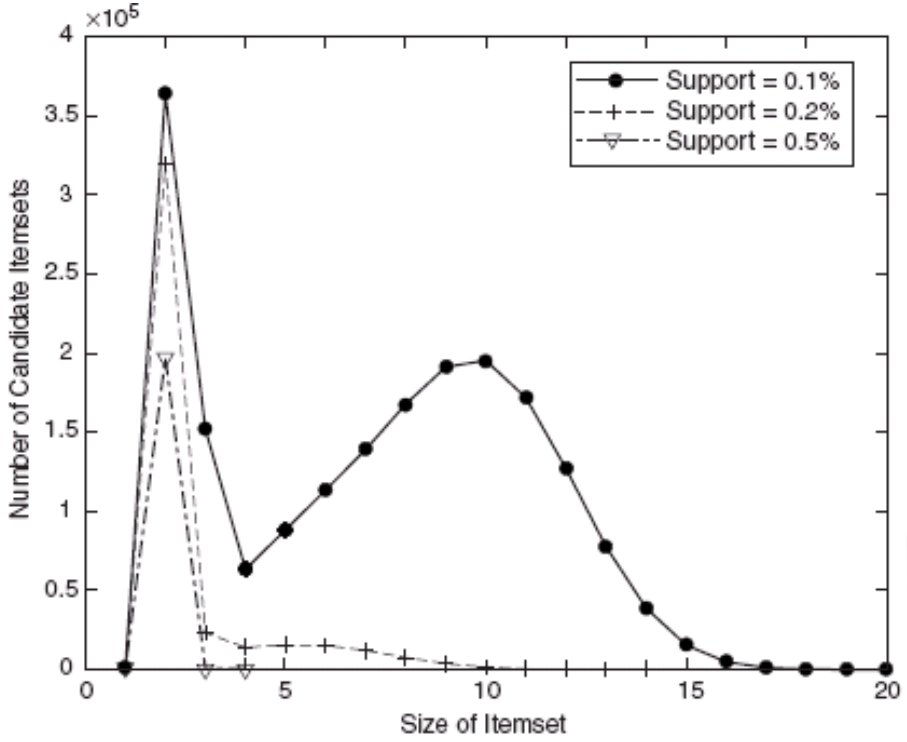
Hash树的子集操作

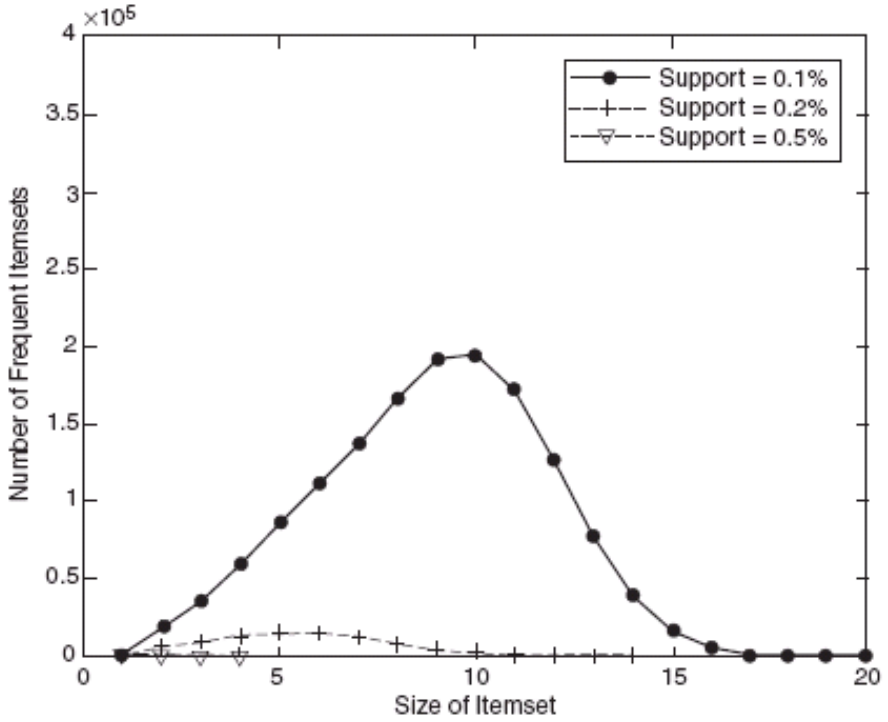
- 习题9
- (a)给定一个包含项{1,3,4,5,8}的事务，在寻找该事务的候选项集时，访问了Hash树的哪些叶结点？
- (b)使用访问的叶结点确定包含的候选项集



Apriori计算复杂度

- 支持度阈值
  - 降低支持度阈值将导致更多的项集是频繁的
  - 会增加频繁项集的最大长度



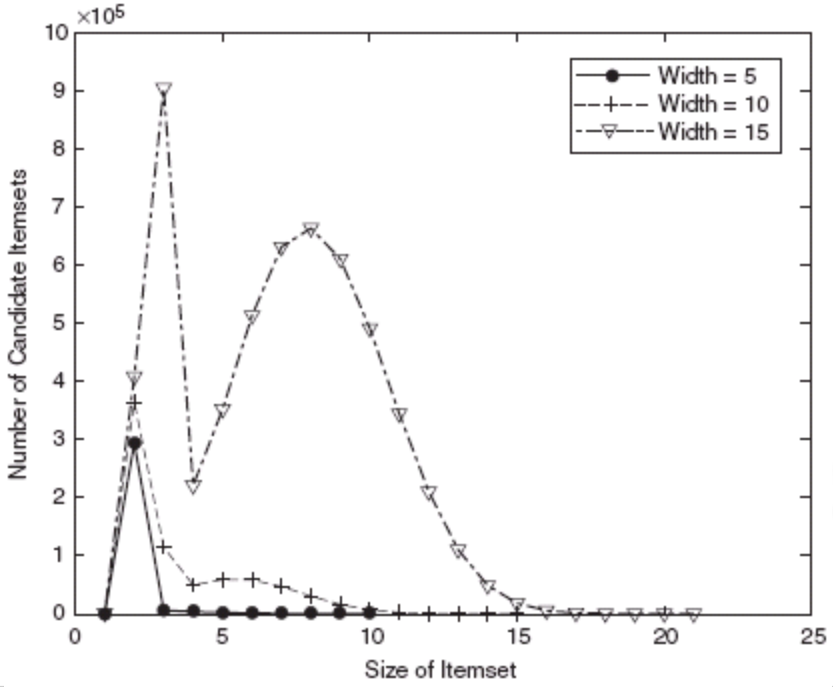


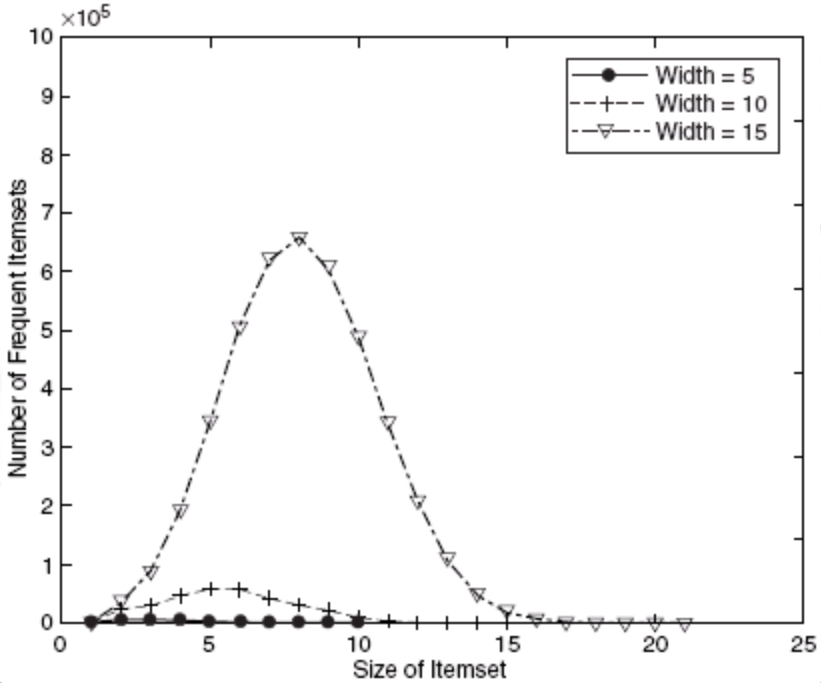
Apriori计算复杂度

- 数据集的维度
  - 需要更多的空间来存储项的支持度计数
  - 如果频繁项集的数目也增长，计算量和IO开销将增加
- 事务数
  - 由于需要反复扫描数据集，因此其运行时间随着事务数增加而增加

Apriori计算复杂度

- 事务的平均宽度
  - 频繁项集的最大长度随事务平均宽度增加而增加
  - 事务中包含的项集也随之增加，同时将增加支持度计数时hash树的遍历次数





关联规则挖掘

- 两步法：
- 1. 频繁项集产生：目标是发现满足最小支持度阈值的所有项集，称作频繁项集
- 2. 规则的产生：从发现的频繁项集中提取所有高置信度的规则，称为强规则

规则产生

- 每个频繁k-项集能产生最多2^k-2个关联规则
  - 将项集Y划分成两个非空的子集X和Y-X，使得X->Y-X满足置信度阈值

$$\sum_{i=1}^{K-1} C_i^K = 2^K - 1 - 1 = 2^K - 2$$

- 从频繁项集提取的规则必然已经满足支持度阈值

规则产生

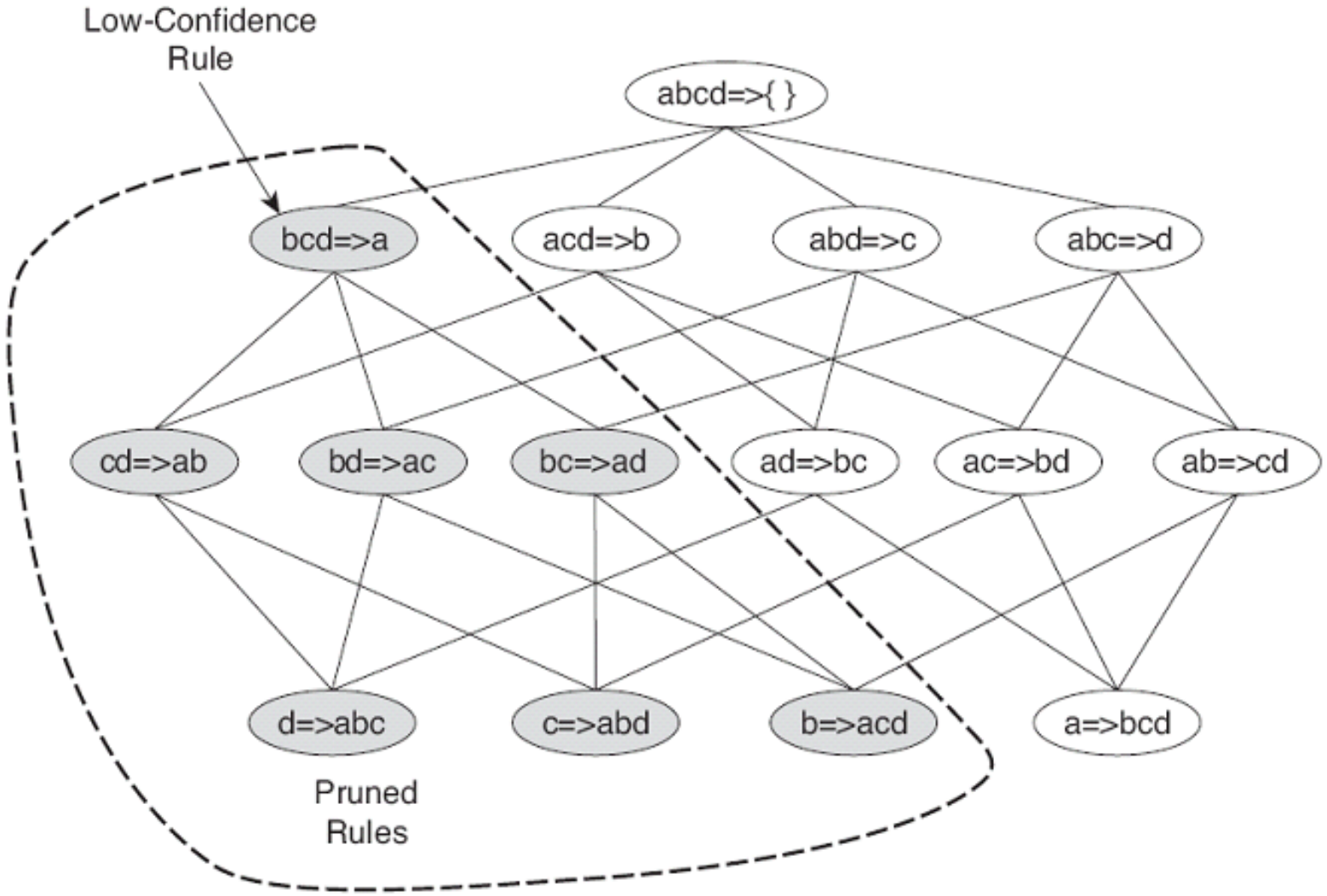
- 置信度不具有任何单调性
  - X->Y的置信度可能大于，小于或等于规则X*->Y*的置信度，其中X*∈X且Y*∈Y

- 定理：如果规则X->Y-X不满足置信度阈值，则X'->Y-X'的规则一定也不满足置信度阈值，其中X'是X的子集
  - σ(Y)/σ(X) >= σ(Y)/σ(X')

规则产生

- **Apriori**算法采用逐层算法来产生关联规则，初始提取规则后件只含一个项的所有高置信度规则；然后使用这些规则来产生新的候选规则
  - 每层对应规则后件中的项数
  - 由**{acd}->{b}**和**{abd}->{c}**合并成**{ad}->{bc}**
- 如果任意结点具有低置信度，则可以立即剪掉该结点生成的整个子图

规则产生


Low-Confidence Rule
abcd=>{ }
bcd=>a
acd=>b
abd=>c
abc=>d
cd=>ab
bd=>ac
bc=>ad
ad=>bc
ac=>bd
ab=>cd
d=>abc
c=>abd
b=>acd
a=>bcd
Pruned Rules

规则产生

Apriori 算法的规则产生

1: for 每一个频繁 k-项集 f_k, k≥2 do
2: $H_1=\{i \mid i \in f_k\}$ {规则的 1-项后件}
3: call ap-genrules(f_k, H_1)
4: end for

过程 ap-genrules(f_k, H_m)
1: k=|f_k| {频繁项集的大小}
2: m=|H_m| {规则后件的大小}
3: if k>m+1 then
4: H_{m+1} = apriori-gen(H_m)
5: for 每个 $h_{m+1} \in H_{m+1}$ do
6: conf=σ(f_k)/ σ(f_k - h_{m+1})
7: if conf ≥ minconf then
8: output: 规则 (f_k - h_{m+1}) →h_{m+1}
9: else
10: 从 H_{m+1} delete h_{m+1}
11: end if
12: end for
13: call ap-genrules(f_k, H_{m+1})
14: end if

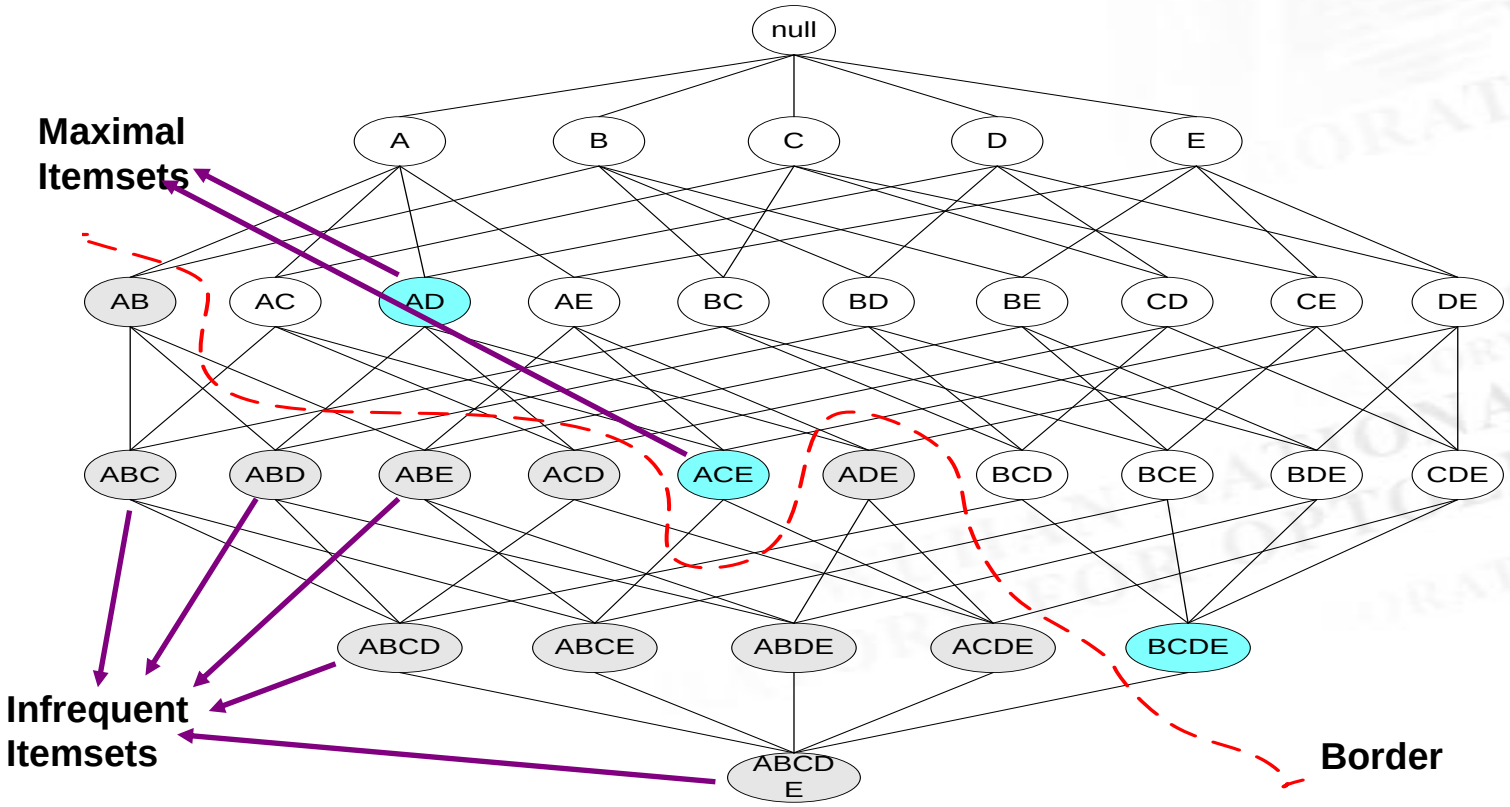
频繁项集的紧凑表示

- 由事务数据集产生的频繁项集的数量可能非常大，需要从中识别出可推导出其他所有的频繁项集的，较小的，具有代表性的项集

- 最大频繁项集和频繁闭项集

频繁项集的紧凑表示

- 最大频繁项集(Maxmial Frequent Itemset)
  - 其直接超集都不是频繁的频繁项集

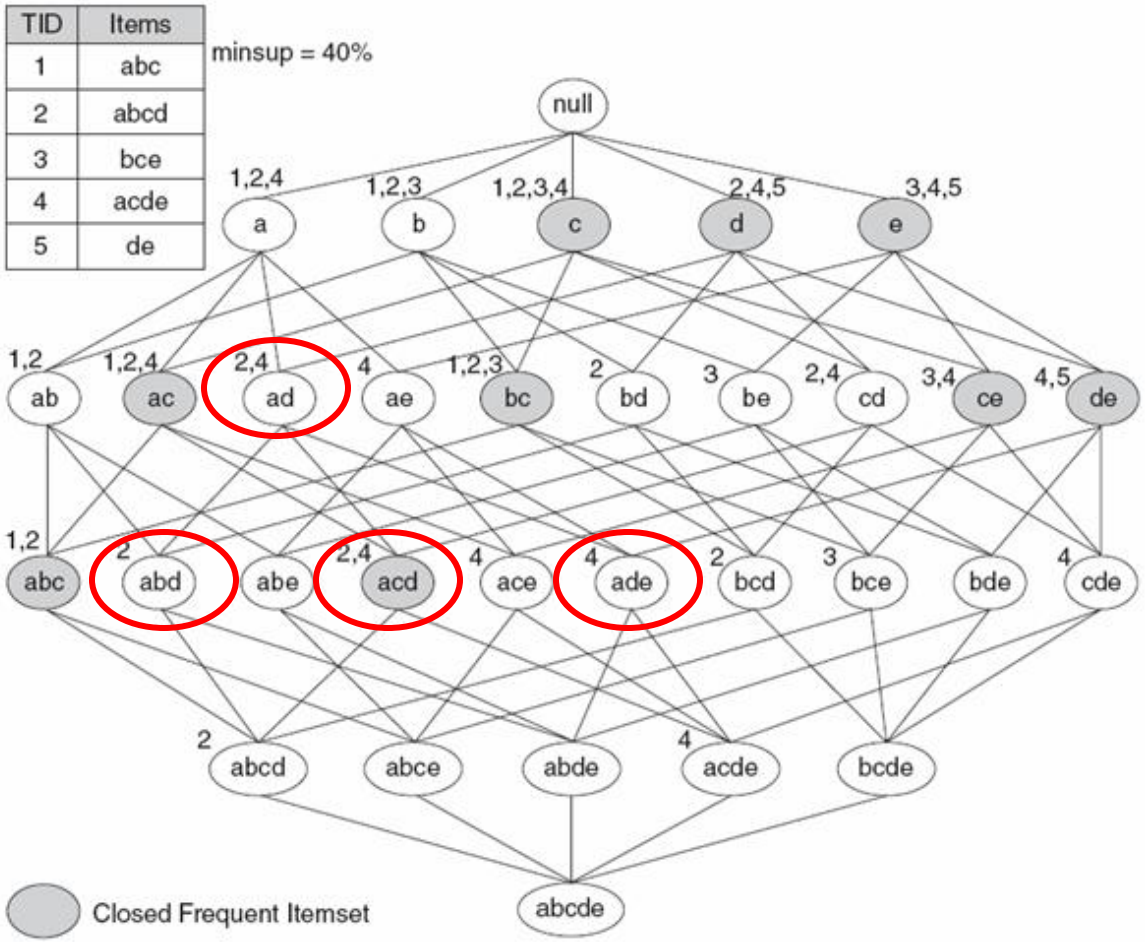


频繁项集的紧凑表示

- 闭项集(Closed Itemset)
  - 项集X是闭的，如果它的直接超集都不具有和它相同的支持度计数
  - 项集的一种最小表示，同时不丢失支持度信息
  - 如果至少存在一个X的直接超集，其支持度计数与X相同，X就不是闭的

频繁项集的紧凑表示

- 频繁闭项集(Closed Frequent Itemset)
  - 一个项集是频繁闭项集，如果它是闭的，且它的支持度大于或等于最小支持度阈值



频繁项集的紧凑表示

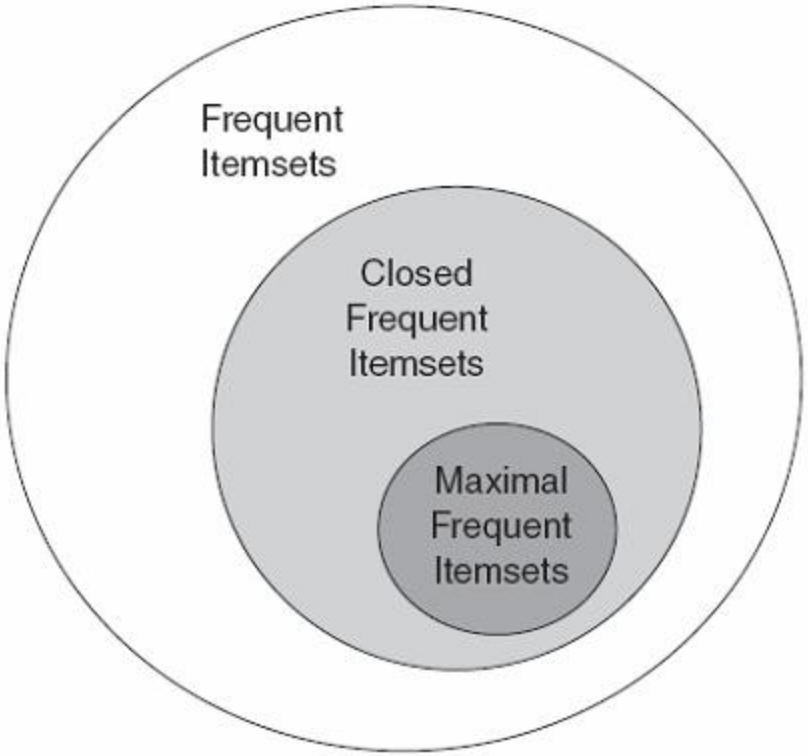
- 使用频繁闭项集的支持度确定那些非闭的频繁项集的支持度
- 非闭项集的支持度一定与它的某个直接超集中的最大支持度相同

使用频繁闭项集进行支持度计数

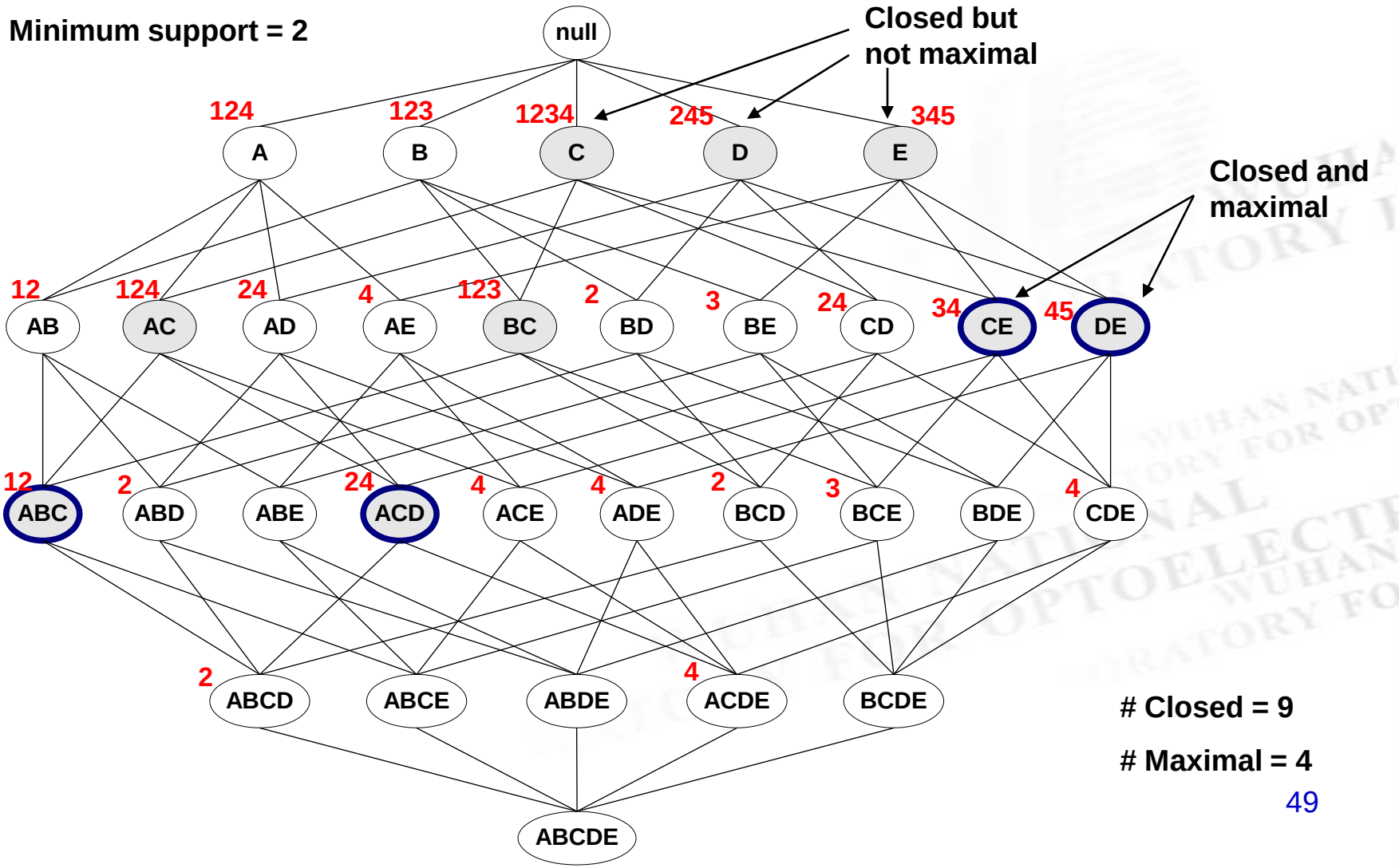
1: 设 C 是频繁闭项集的集合
2: 设 k_{max} 是频繁闭项集的最大长度
3: $F_{kmax}=\{f|f\in C, |f|=k_{max}\}$ {找出长度为 k_{max} 的所有频繁项集}
4: for $k=k_{max}-1$ downto 1 do
5: $F_k=\{f|f\subset F_{k+1}, |f|=k\}$ {找出长度为 k 的所有频繁项集}
6: for 每个 $f\in F_k$ do
7: if $f\notin C$ then
8: f.support = max{f'.support|$f'\in F_{k+1}, f\subset f'$}
9: end if
10: end for
11: end for

频繁项集的紧凑表示

- 最大频繁项都是闭的，因为任何最大频繁项集都不可能与它的直接超集具有相同的支持度计数



频繁项集的紧凑表示

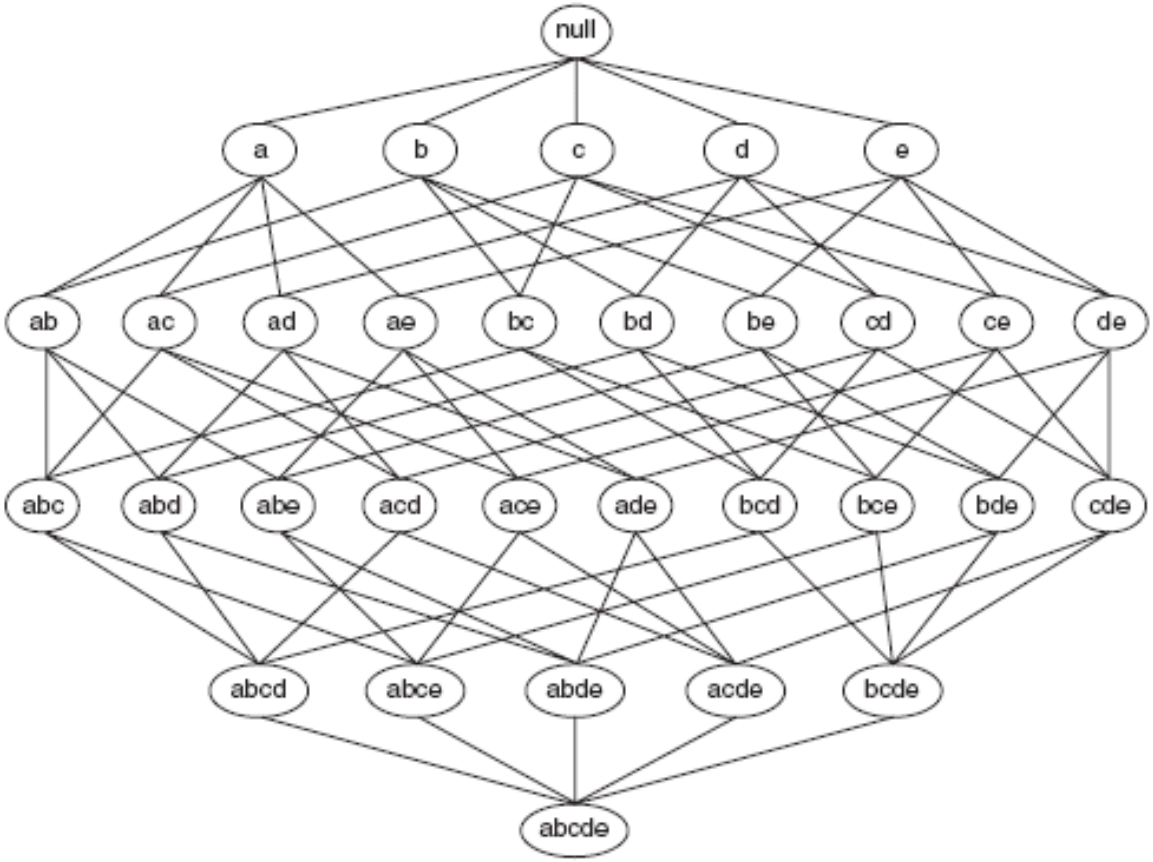


Closed = 9

Maximal = 4

习题11

- 给定如下格结构和事务，标记其中每个结点

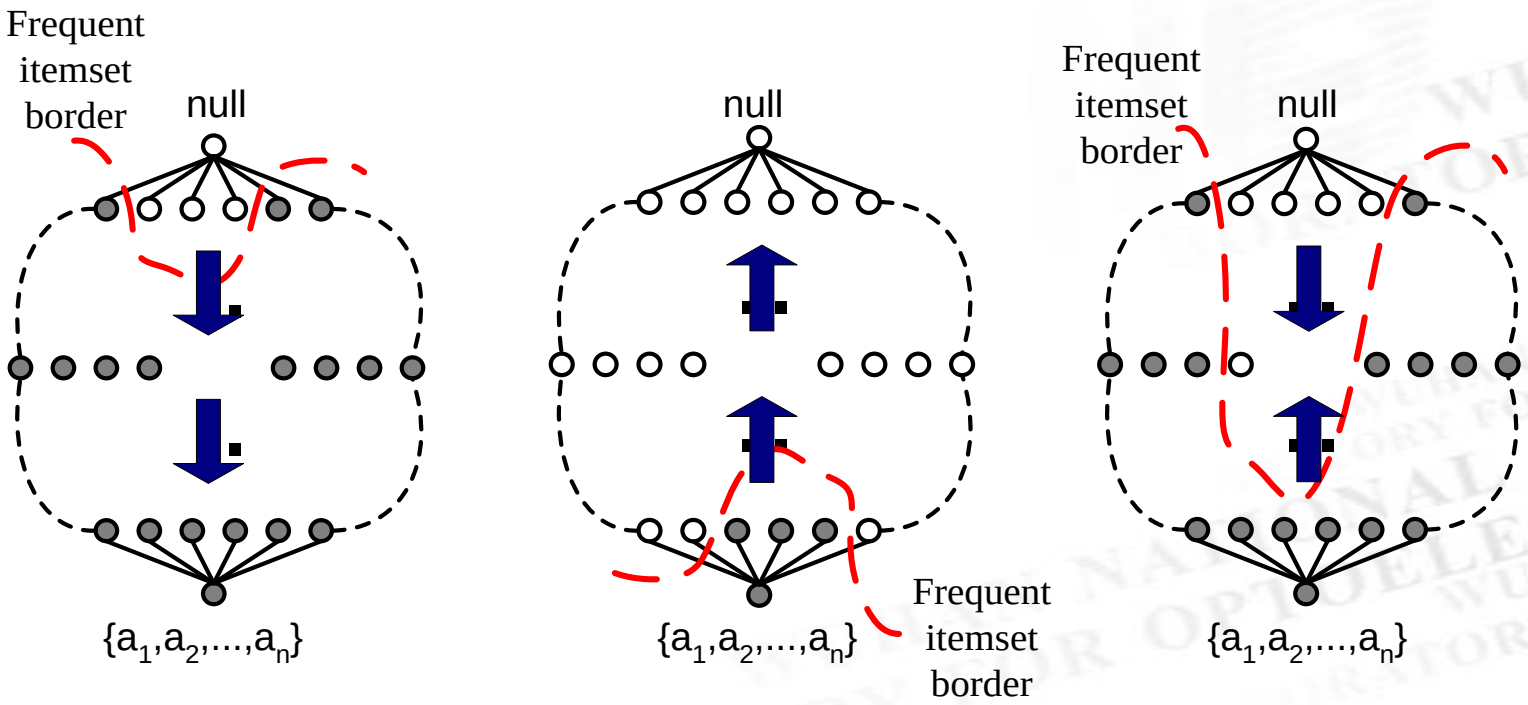


Transaction ID	Items Bought
1	$\{a, b, d, e\}$
2	$\{b, c, d\}$
3	$\{a, b, d, e\}$
4	$\{a, c, d, e\}$
5	$\{b, c, d, e\}$
6	$\{b, d, e\}$
7	$\{c, d\}$
8	$\{a, b, c\}$
9	$\{a, d, e\}$
10	$\{b, d\}$

M：最大频繁项集
C：频繁闭项集
N：频繁的，非最大或闭项集
I：非频繁的
Minsup=30%

产生频繁项集的其他方法

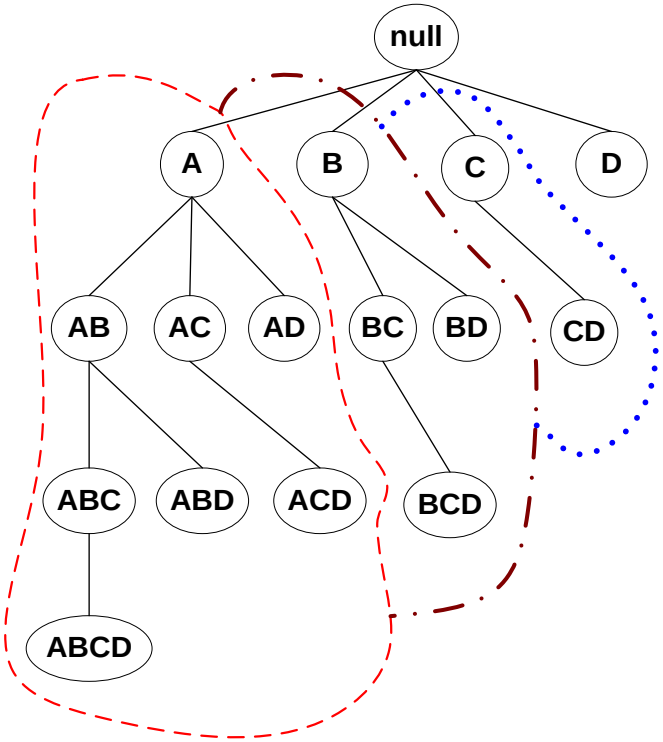
- 项集格遍历



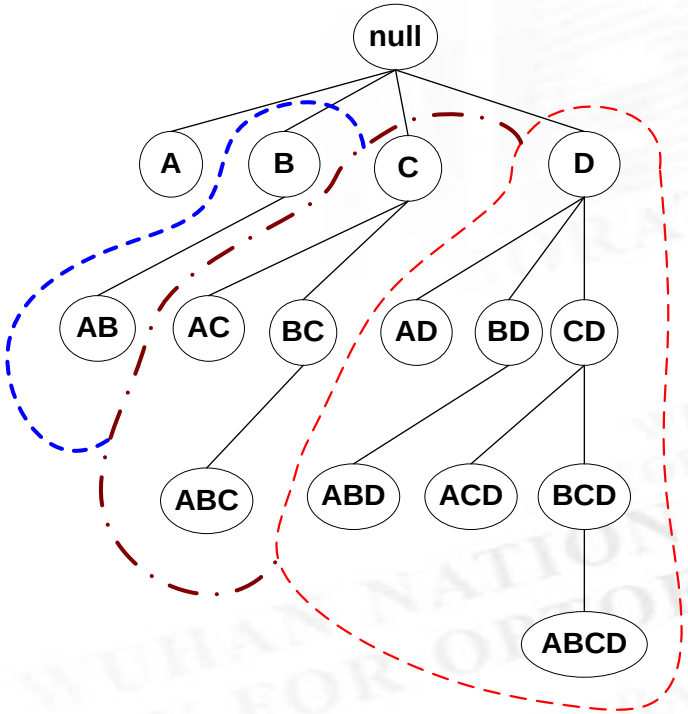
(a) General-to-specific (b) Specific-to-general (c) Bidirectional

产生频繁项集的其他方法

- 等价类



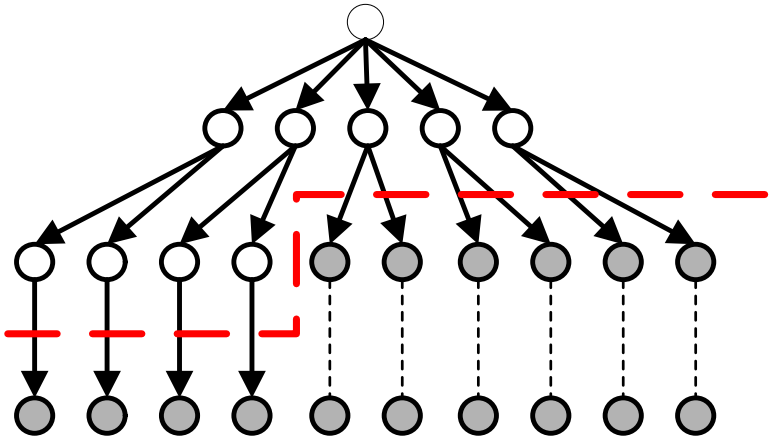
(a) Prefix tree



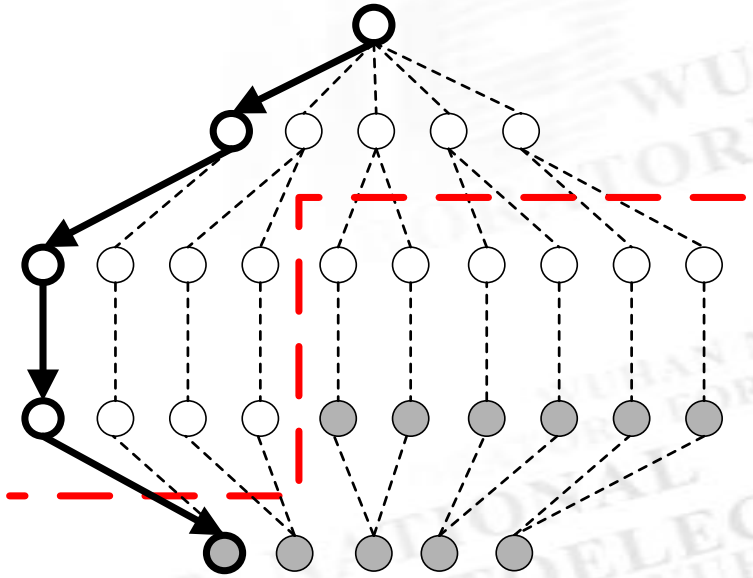
(b) Suffix tree

产生频繁项集的其他方法

- 宽度优先与深度优先

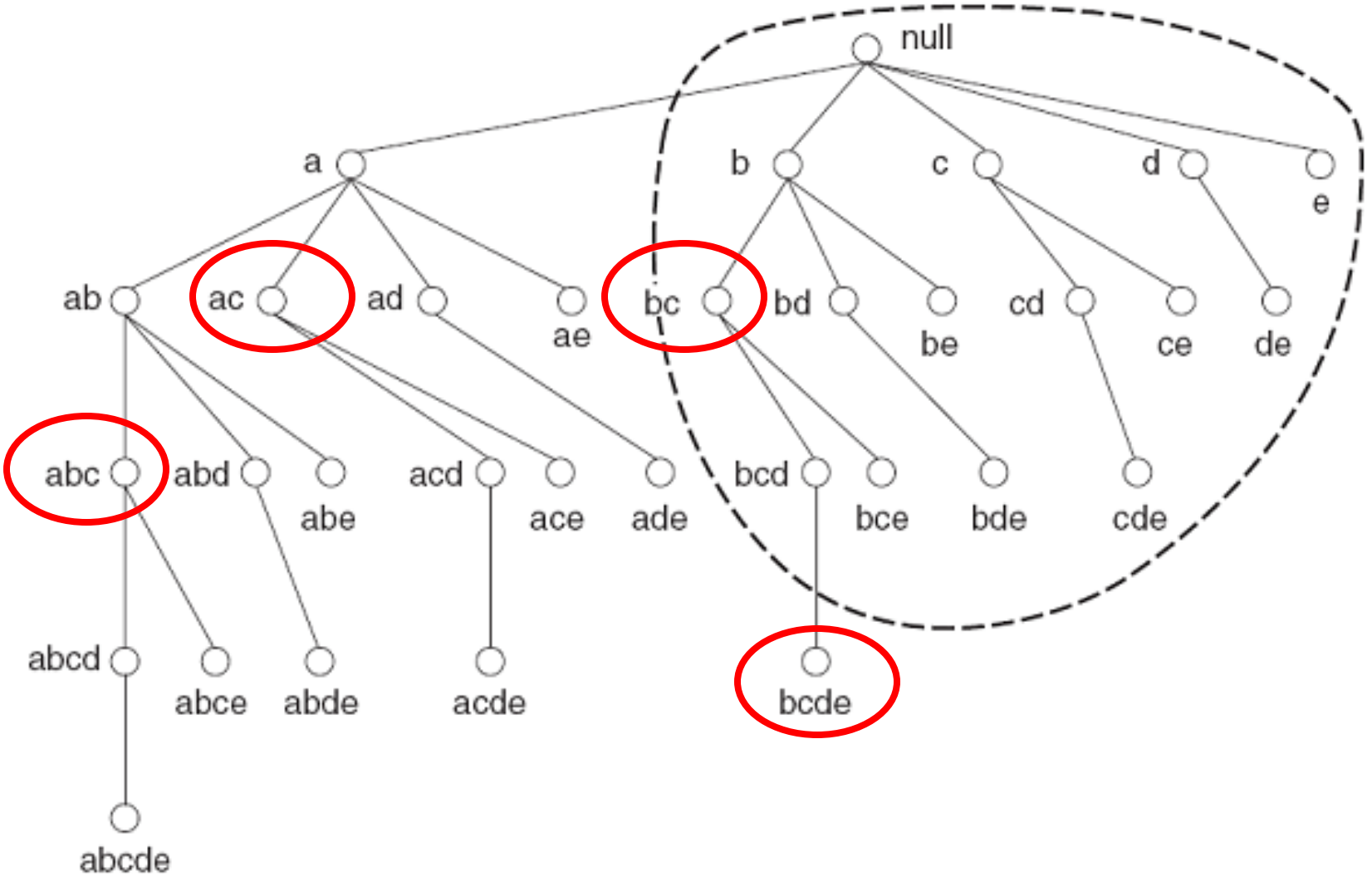


(a) Breadth first



(b) Depth first

产生频繁项集的其他方法



产生频繁项集的其他方法

- 事务数据集的表示
  - 水平和垂直数据形式

Horizontal Data Layout

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B

Vertical Data Layout

A	B	C	D	E
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

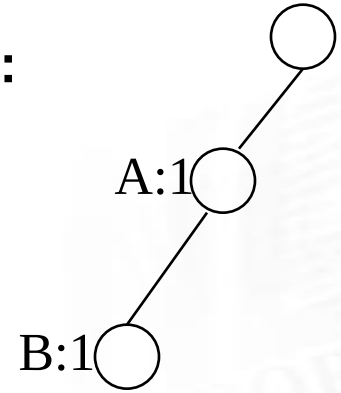
FP增长算法

- FP树是一种输入数据的压缩表示方法，通过逐个读入事务，并把每个事务映射到FP树中的一条路径来构造
  - 事务中相同的项意味着路径的重叠
- 一旦构建了FP树，可直接从中提取频繁项集，而不必重复扫描数据

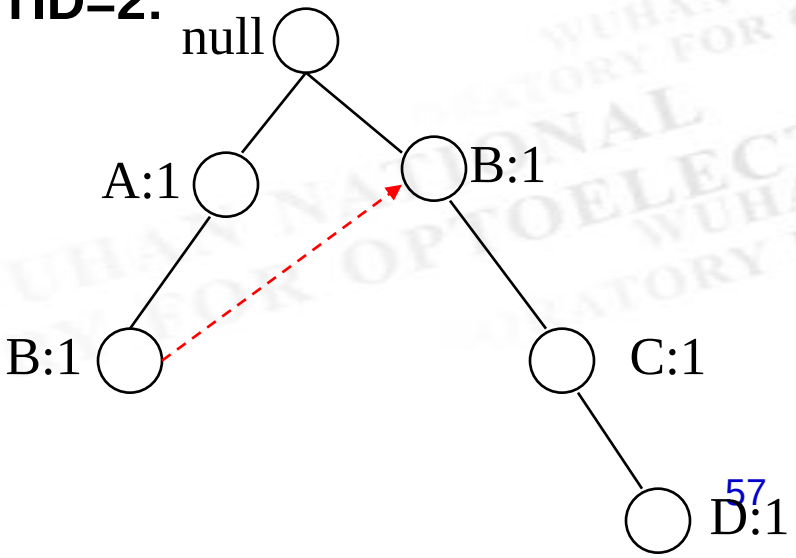
FP增长算法

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

After reading TID=1:



After reading TID=2:



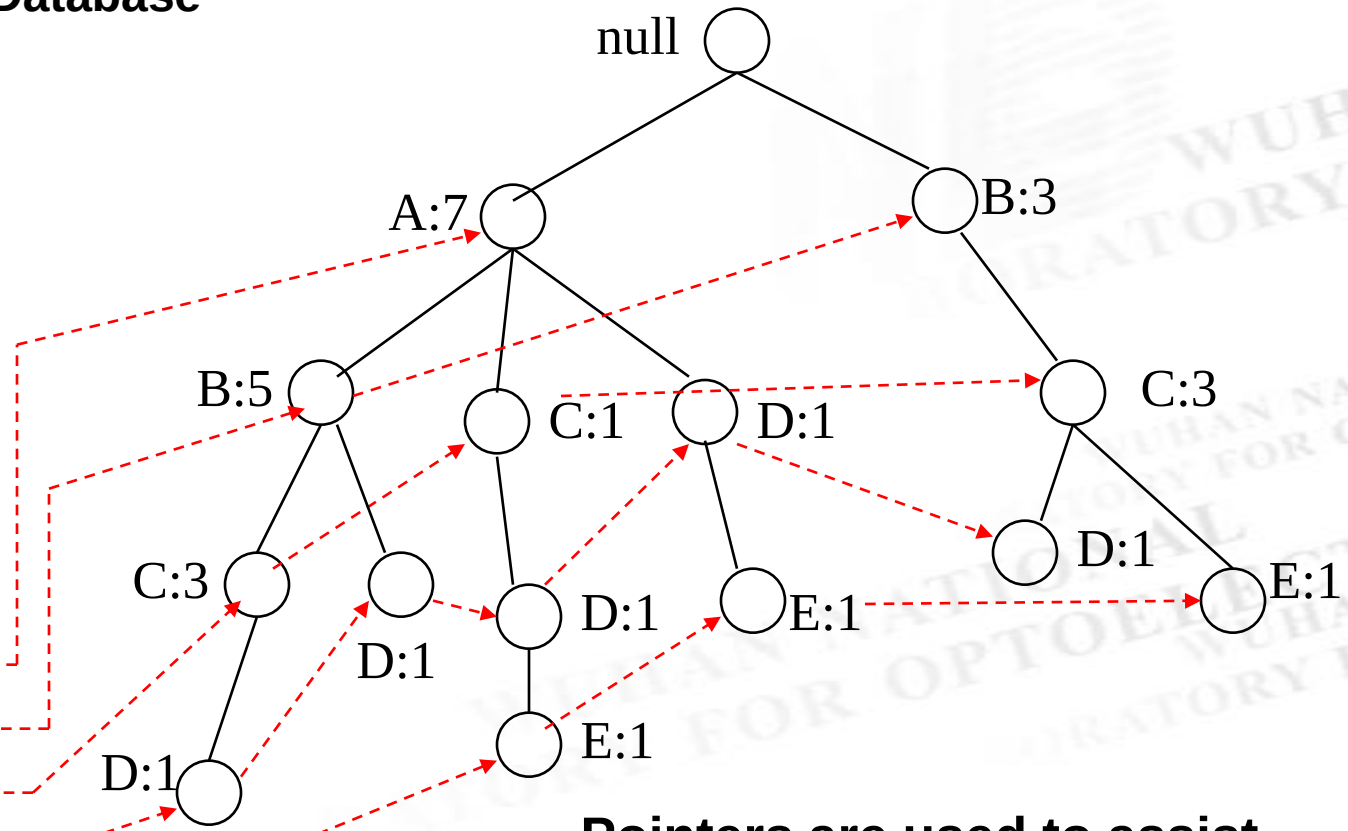
FP增长算法

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Transaction Database

Header table

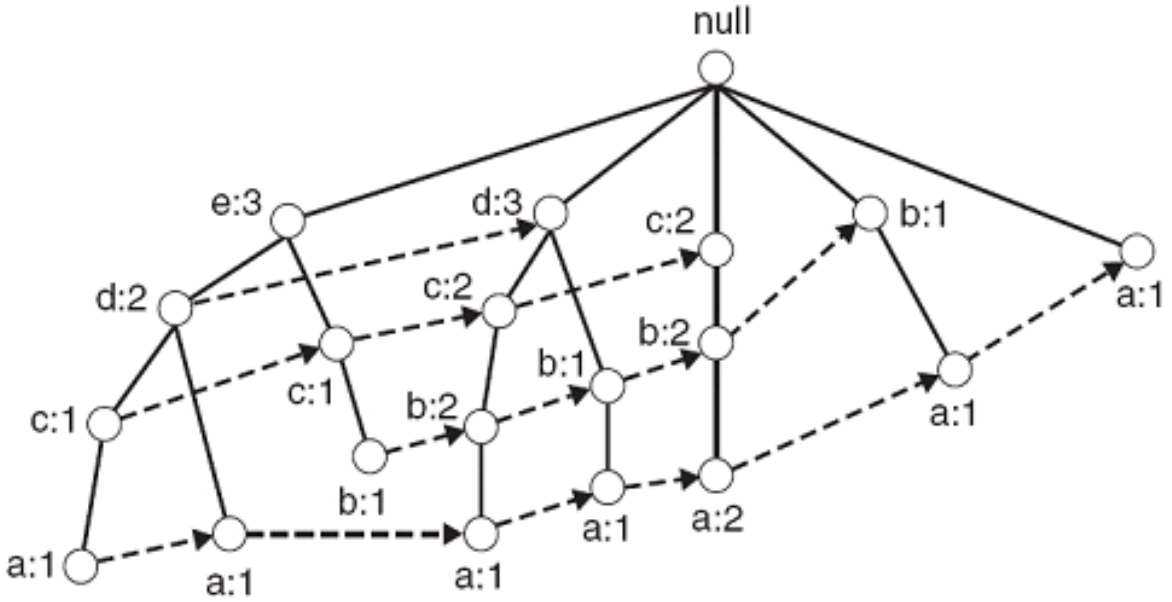
Item	Pointer
A	
B	
C	
D	
E	



Pointers are used to assist frequent itemset generation

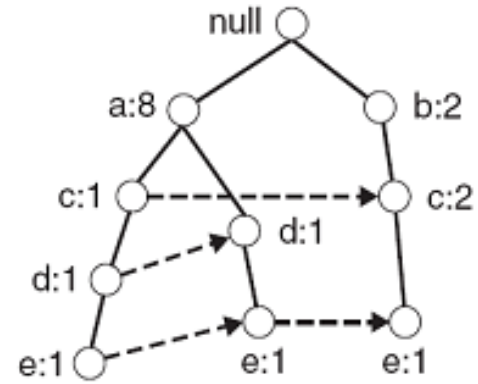
FP增长算法

- FP树的大小通常比未压缩的数据小
- 最坏情况下，需要附加的空间为每个项存放结点间的指针和计数，存储空间更大
- FP树的大小也依赖于如何对项排序

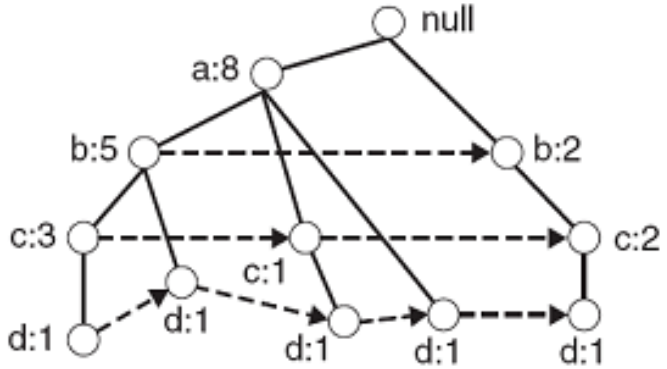


FP增长算法

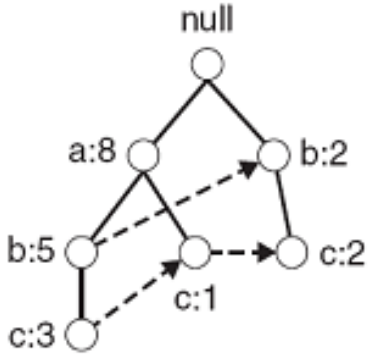
- FP增长是一种自底向上方式探索树，产生频繁项集的算法



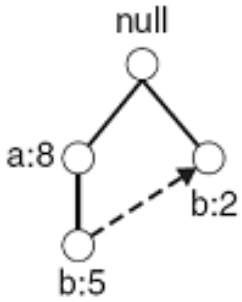
(a) Paths containing node e



(b) Paths containing node d



(c) Paths containing node c



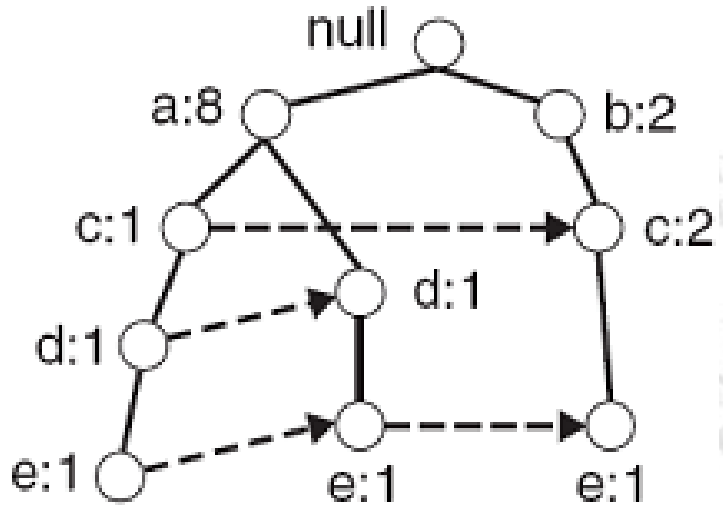
(d) Paths containing node b

(e) Paths containing node a

FP增长算法

- 考虑发现所有以e结尾的频繁项集任务
- 1. 收集包含e结点的所有路径，称为前缀路径
- 2. 若最小支持度为2，则

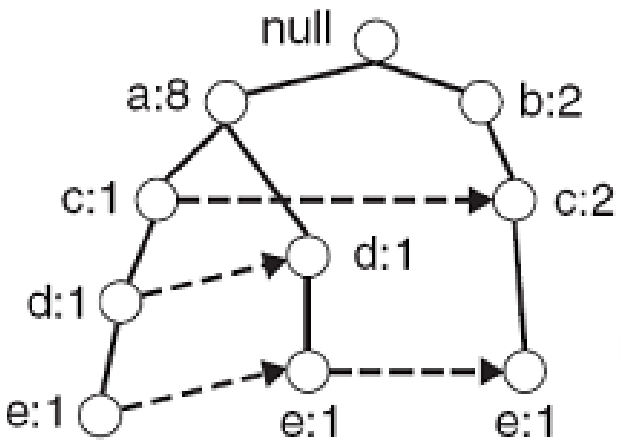
通过统计与结点e相关联的支持度计数，可知{e}是频繁项集



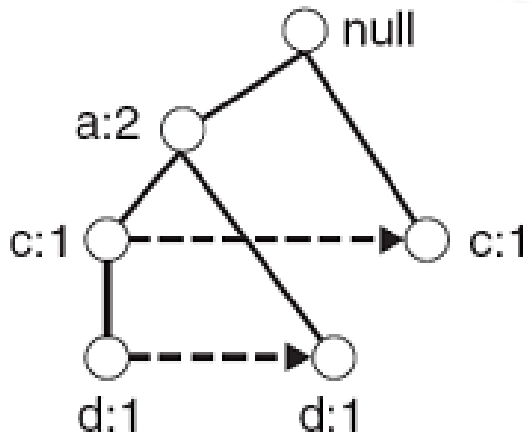
(a) Prefix paths ending in e

FP增长算法

- 3. 以de，ce，be，ae结尾的项集是否是频繁项集？
  - 将前缀路径转化成条件FP树
  - 用于发现以某特定后缀结尾的频繁项集

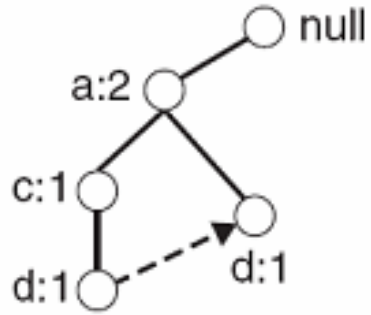


(a) Prefix paths ending in e



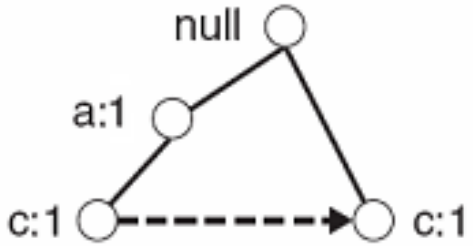
(b) Conditional FP-tree for e

FP增长算法



(c) Prefix paths ending in de

(d) Conditional FP-tree for de



(e) Prefix paths ending in ce

(f) Prefix paths ending in ae

关联模式的评估

- 关联分析算法具有产生大量模式的潜在能力
  - 其中许多是不感兴趣的和冗余的
  - 如果{A,B,C}->{D}和{A,B}->{D}具有同样的支持度和置信度，则产生了冗余
- 在原先的模式评估中，仅采用支持度和置信度作为评价准则
- 感兴趣度量可以用来删除和排序模式

关联模式的评估

- 客观兴趣度度量
  - 通过统计论据建立
- 主观兴趣度度量
  - 需要领域专家的大量先验知识
  - {黄油}->{面包}不是有趣的，即使它具有很高的支持度和置信度

关联模式的评估

- 兴趣度的客观度量
  - 是一种评估关联模式质量的数据驱动的方法，不依赖于领域知识，只需最小限度用户的输入信息
- 相依表

Contingency table for $X \rightarrow Y$

	Y	/Y	
X	f_{11}	f_{10}	f_{1+}
/X	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	|T|

f_{11}: support of X and Y
f_{10}: support of X and $\overline{Y}$
f_{01}: support of $\overline{X}$ and Y
f_{00}: support of $\overline{X}$ and $\overline{Y}$

用来定义多种度量

- support, confidence, lift, Gini, J-measure, etc.

关联模式的评估

➢置信度的局限性

	Coffee	/Coffee	
Tea	150	50	200
/Tea	650	150	800
	800	200	1000

Association Rule: Tea → Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.8

⇒ 虽然置信度很高，但却是误导

⇒ P(Coffee|$\overline{\text{Tea}}$) = 0.8125

置信度的缺陷在于忽略了规则后件中项集的支持度

关联模式的评估

- 提升度(lift)
  - 计算规则置信度和规则后件中项集的支持度之间的比率

$$lift(A \rightarrow B) = \frac{c(A \rightarrow B)}{b(B)} \qquad Lift = \frac{P(Y \mid X)}{P(Y)}$$

- 对于二元变量，提升度等效为兴趣因子

$$I(A,B) = \frac{s(A,B)}{s(A) \times s(B)} = \frac{Nf_{11}}{f_{1+} f_{+1}}$$

关联模式的评估

- 兴趣因子比较模式的频率与统计独立假设下计算的基准频率
- 基准频率 $\frac{f_{1+}}{N} \times \frac{f_{+1}}{N}$
- 模式频率 $\frac{f_{11}}{N}$

	Coffee	/Coffee	
Tea	150	50	200
/Tea	650	150	800
	800	200	1000

1000 × 150/(200×800) = 0.9375

- I(A,B) = 1 => 统计独立
- I(A,B) > 1 => 正相关
- I(A,B) < 1 => 负相关

关联模式的评估

- 兴趣因子的局限性

	p	/p	
q	880	50	930
/q	50	20	70
	930	70	1000

I(p,q) = 1.02

	r	/r	
s	20	50	70
/s	50	880	930
	70	930	1000

I(r,s) = 4.08

此时置信度是更好的选择

关联模式的评估

- 相关分析

$$\phi = \frac{f_{11}f_{00} - f_{01}f_{10}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}$$

  - 相关度的值从-1到+1，0表示统计独立
- 相关分析的局限性
  - 把项在事务中同时出现和同时不出现视为同等重要
  - 更适合分析对称的二元变量

关联模式的评估

- IS是一种用于处理非对称二元变量的度量

$$IS(A,B)=\sqrt{I(A,B)\times s(A,B)}=\frac{s(A,B)}{\sqrt{s(A)s(B)}}$$

  - 当模式的兴趣因子和支持度都很大时，IS也很大
  - 等价于二元变量的余弦度量
  - 也可以表示为从一对二元变量中提取出的关联规则的置信度的几何均值

$$IS(A,B)=\sqrt{\frac{s(A,B)}{s(A)}\times\frac{s(A,B)}{s(B)}}=\sqrt{c(A\rightarrow B)\times c(B\rightarrow A)}$$

关联模式的评估

- 客观度量的一致性
- 客观度量的性质
- 好的度量M必须满足三点性质：
- 当A和B统计独立时，M(A,B) = 0
- 当P(A)和P(B)保持不变时，M(A,B)随P(A,B)单调递增
- 当P(A,B)和P(B)(或P(A))保持不变时，M(A,B)随P(A)(或P(B))单调递减

关联模式的评估

10个相依表

Example	f_{11}	f_{10}	f_{01}	f_{00}
E1	8123	83	424	1370
E2	8330	2	622	1046
E3	9481	94	127	298
E4	3954	3080	5	2961
E5	2886	1363	1320	4431
E6	1500	2000	500	6000
E7	4000	2000	1000	3000
E8	4000	2000	2000	2000
E9	1720	7121	5	1154
E10	61	2483	4	7452

采用不同的度量对相依表排序

#	ϕ	λ	α	Q	Y	κ	M	J	G	s	c	L	V	I	IS	PS	F	AV	S	ζ	K
E1	1	1	3	3	3	1	2	2	1	3	5	5	4	6	2	2	4	6	1	2	5
E2	2	2	1	1	1	2	1	3	2	2	1	1	1	8	3	5	1	8	2	3	6
E3	3	3	4	4	4	3	3	8	7	1	4	4	6	10	1	8	6	10	3	1	10
E4	4	7	2	2	2	5	4	1	3	6	2	2	2	4	4	1	2	3	4	5	1
E5	5	4	8	8	8	4	7	5	4	7	9	9	9	3	6	3	9	4	5	6	3
E6	6	6	7	7	7	7	6	4	6	9	8	8	7	2	8	6	7	2	7	8	2
E7	7	5	9	9	9	6	8	6	5	4	7	7	8	5	5	4	8	5	6	4	4
E8	8	9	10	10	10	8	10	10	8	4	10	10	10	9	7	7	10	9	8	7	9
E9	9	9	5	5	5	9	9	7	9	8	3	3	3	7	9	9	3	7	9	9	8
E10	10	8	6	6	6	10	5	9	10	10	6	6	5	1	10	10	5	1	10	10	7

关联模式的评估

- 反演性

A	B
1	0
0	0
0	0
0	0
0	1
0	0
0	0
0	0
0	0
1	0

(a)

C	D
0	1
1	1
1	1
1	1
1	0
1	1
1	1
1	1
1	1
0	1

(b)

E	F
0	0
1	0
1	0
1	0
1	1
1	0
1	0
1	0
1	0
0	0

(c)

如果度量在反演操作下是不变的，则向量对(C,D)的度量值和向量对(A,B)的度量值应当向等

关联模式的评估

- 定义：如果交换频度计数f_{11}和f_{00}，f_{10}和f_{01}后，它的值不变，则称客观度量M在反演操作下是不变的

- 反演操作下保持不变的度量，不适合分析非对称的二元数据

关联模式的评估

- 零加性
- 向数据集中添加不相关数据的过程就是所谓的“零加”操作
- 如果增加f_{oo}而保持相依表中所有其他的频度不变并不影响M的值，则度量M在零加操作下是不变的
- 对文档分析或购物篮分析时，期望度量在零加操作下保持不变

关联模式的评估

- 缩放不变性

	男	女	
高	30	20	50
低	40	10	50
	70	30	100

	男	女	
高	60	60	120
低	80	30	110
	140	90	230

- 客观度量M在行/列缩放操作下是不变的，如果M(T)=M(T')，其中T是频度计数为[f_{11};f_{10}; f_{01};f_{00}]的相依表，T'是频度计数为[$k_1k_3f_{11}$; $k_2k_3f_{10}$; $k_1k_4f_{01}$; $k_2k_4f_{00}$]的相依表
  - k_1,k_2,k_3,k_4是正常量

关联模式的评估

- 多个二元变量的度量

c	b	/b	
a	f_{111}	f_{101}	f_{1+1}
/a	f_{011}	f_{001}	f_{0+1}
	f_{+11}	f_{+01}	f_{++1}

/c	b	/b	
a	f_{110}	f_{100}	f_{1+0}
/a	f_{010}	f_{000}	f_{0+0}
	f_{+10}	f_{+00}	f_{++0}

关联模式的评估

- 辛普森悖论
  - 隐藏的变量可能会导致观察到的一对变量之间的联系消失或逆转方向
  - c(HDTV->健身器)=99/180=55%
  - c(/HDTV->健身器)=54/120=45%
  - 购买了HDTV的顾客

更有可能购买健身器？

买HDTV	买健身器		
	是	否	
是	99	81	180
否	54	66	120
	153	147	300

关联模式的评估

- 对于大学生
  - c(HDTV->健身器)=1/10=10%
  - c(/HDTV->健身器)=4/34=11.8%
- 对于在职人员
  - 98/170=57.7%
  - 50/86=58.1%

顾客组	买HDTV	买健身器		总数
		是	否	
大学生	是	1	9	10
	否	4	30	34
在职人员	是	98	72	170
	否	50	36	86

关联模式的评估

- 在组合数据下，**HDTV**与健身器呈正相关
- 在分层数据下，**HDTV**与健身器呈负相关
- 需要适当的分层才能避免因辛普森悖论产生虚假的模式
  - **a/b < c/d** 并且 **p/q < r/s**
  - 当**(a+p)/(b+q) > (c+r)/(d+s)**时，出现悖论