关联规则挖掘

丁兆云

关联规则挖掘

丁兆云

主要内容

- 1. 基于概念
- 2. 频繁项挖掘算法
- 3. 关联分析的评估

1.1定义:关联分析（association analysis）

- 关联分析用于发现隐藏在大型数据集中的令人感兴趣的联系，所发现的模式通常用关联规则或频繁项集的形式表示。
- 关联规则反映一个事物与其它事物之间的相互依存性和关联性。如果两个或者多个事物之间存在一定的关联关系，那么，其中一个事物发生就能够预测与它相关联的其它事物的发生。

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

Rules Discovered:
{Diaper} --> {Beer}

1.1定义: 频繁项集（Frequent Itemset）

- 项集（Itemset）
  - 包含0个或多个项的集合

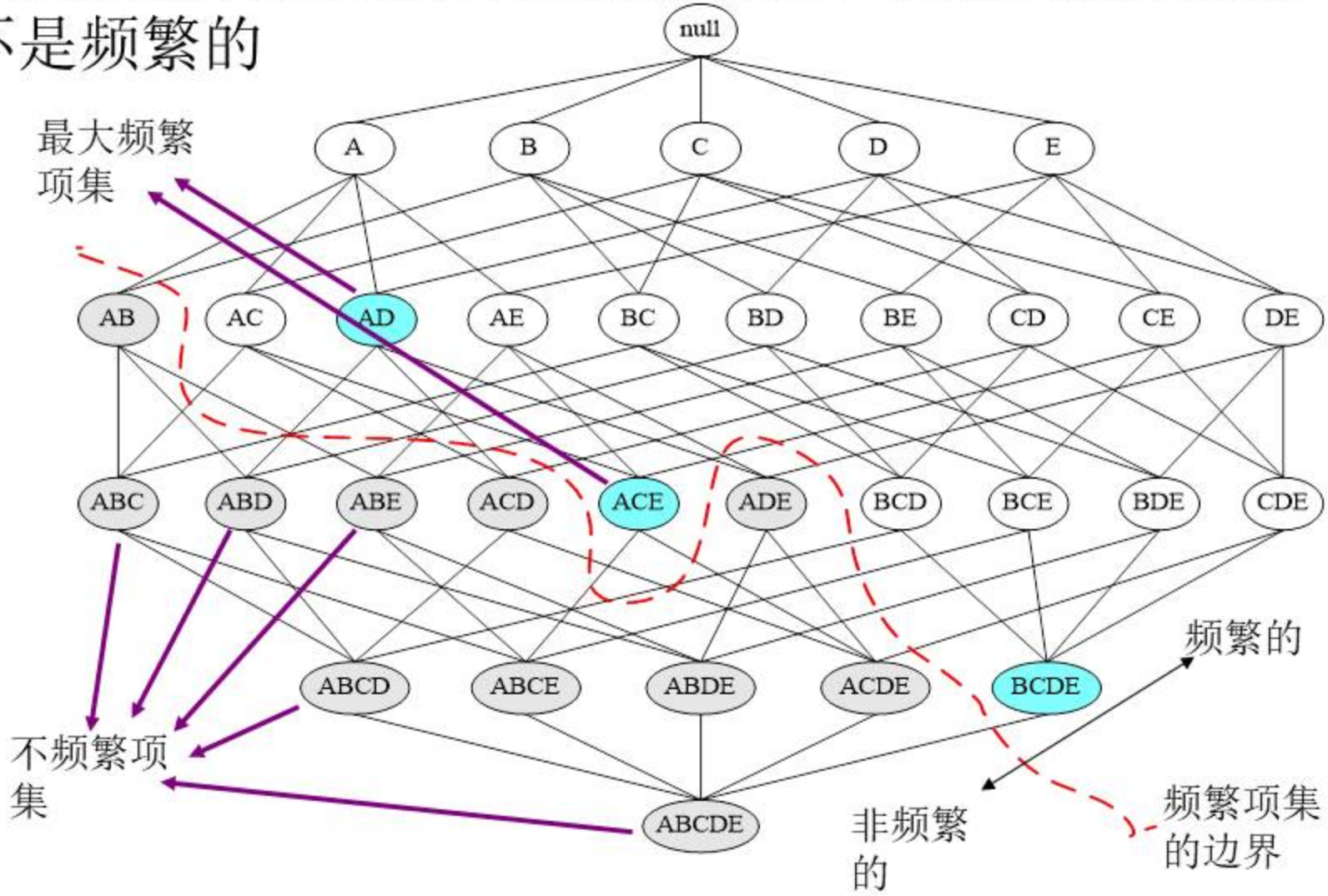
    例子: {Milk, Bread, Diaper}
  - k-项集

    如果一个项集包含k个项
- 支持度计数（Support count）(σ)
  - 包含特定项集的事务个数
  - 例如： σ({Milk, Bread,Diaper}) = 2
- 支持度（Support）
  - 包含项集的事务数与总事务数的比值
  - 例如： s({Milk, Bread, Diaper}) = 2/5
- 频繁项集（Frequent Itemset）
  - 满足最小支持度阈值（ *minsup* ）的所有项集

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

1.2最大频繁项集（Maximal Frequent Itemset）

最大频繁项集是这样的频繁项集，它的直接超集都不是频繁的



1.3定义: 关联规则（Association Rule）

- 关联规则
  - 关联规则是形如 X → Y的蕴含表达式, 其中 X 和 Y 是不相交的项集
  - 例子:
    {Milk, Diaper} → {Beer}

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

- 关联规则的强度
  - 支持度 Support (s)
    - 确定项集的频繁程度
  - 置信度 Confidence (c)
    - 确定Y在包含X的事务中出现的频繁程度

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

1.4关联规则挖掘问题

- 关联规则挖掘问题：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*和*minconf*是对应的支持度和置信度阈值

填空题 2分

设置

事务Id	项集合
10	A, B, C
20	A, C
30	A, D
40	B, E, F

最小支持度50%
最小置信度50%

频繁模式	支持度
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

支持度 = $\text{support}(\{A\} \cup \{C\})$ = [填空1]

置信度 = $\text{support}(\{A\} \cup \{C\}) / \text{support}(\{A\})$ = [填空2]

正常使用填空题需3.0以上版本雨课堂

作答

1.4关联规则挖掘问题

- 关联规则挖掘问题：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*和*minconf*是对应的支持度和置信度阈值

事务Id	项集合
10	A, B, C
20	A, C
30	A, D
40	B, E, F

最小支持度50%
最小置信度50%

频繁模式	支持度
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

规则 $A \Rightarrow C$:

支持度 = support($\{A\} \cup \{C\}$) = 50%

置信度 = support($\{A\} \cup \{C\}$)/support($\{A\}$) = 66.6%

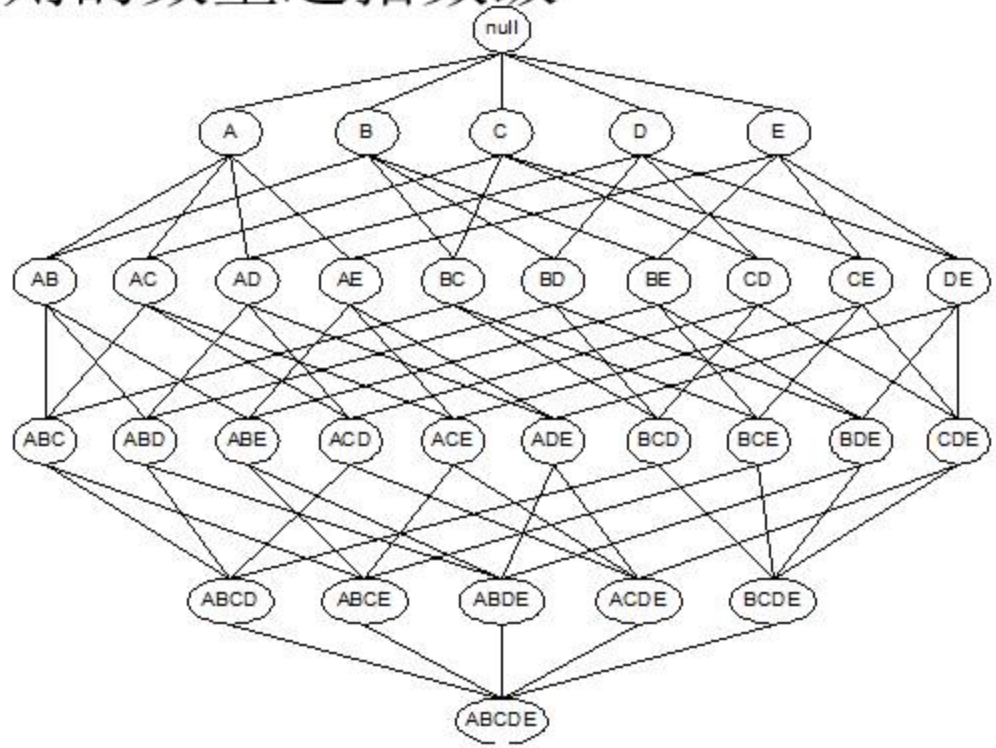
1.4挖掘关联规则（Mining Association Rules）

- 大多数关联规则挖掘算法通常采用的一种策略是，将关联规则挖掘任务分解为如下两个主要的子任务：

1. 频繁项集产生（Frequent Itemset Generation）
   - 其目标是发现满足最小支持度阈值的所有项集，这些项集称作频繁项集。
2. 规则的产生（Rule Generation）
   - 其目标是从上一步发现的频繁项集中提取所有高置信度的规则，这些规则称作强规则（strong rule）。

1.5关联规则原始方法

- 挖掘关联规则的一种原始方法是：Brute-force approach:
  - 计算每个可能规则的支持度和置信度
  - 这种方法计算代价过高，因为可以从数据集提取的规则的数量达指数级

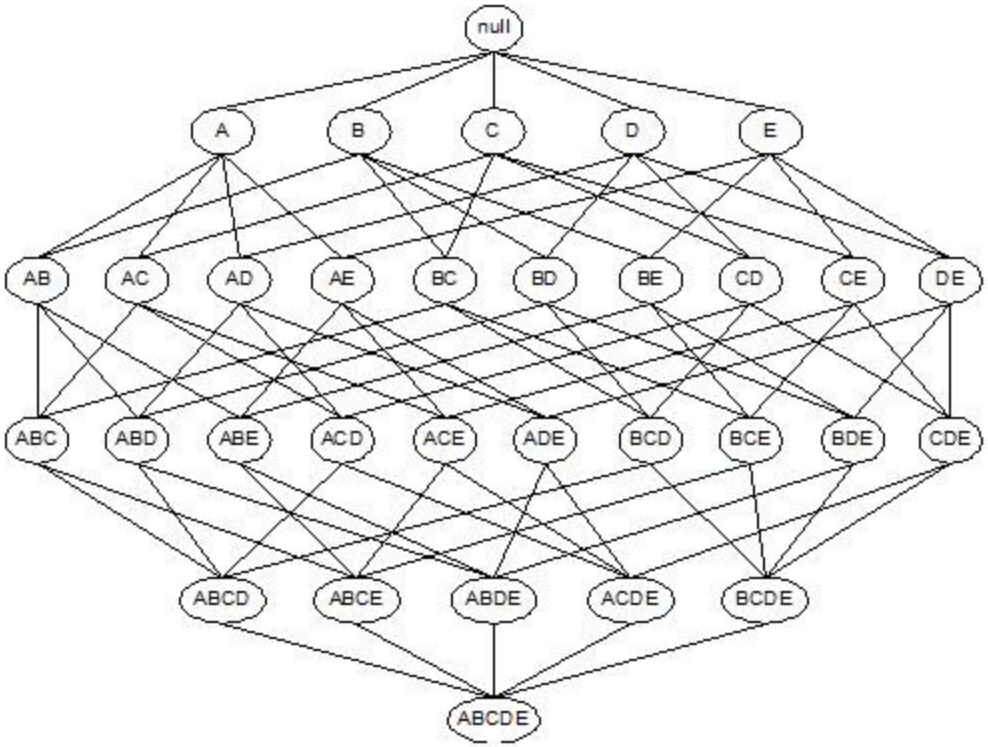


主要内容

- 1. 基于概念
- 2. 频繁项挖掘算法
- 3. 关联分析的评估

2频繁项集产生（Frequent Itemset Generation）

- Brute-force 方法:
  - 计算开销大



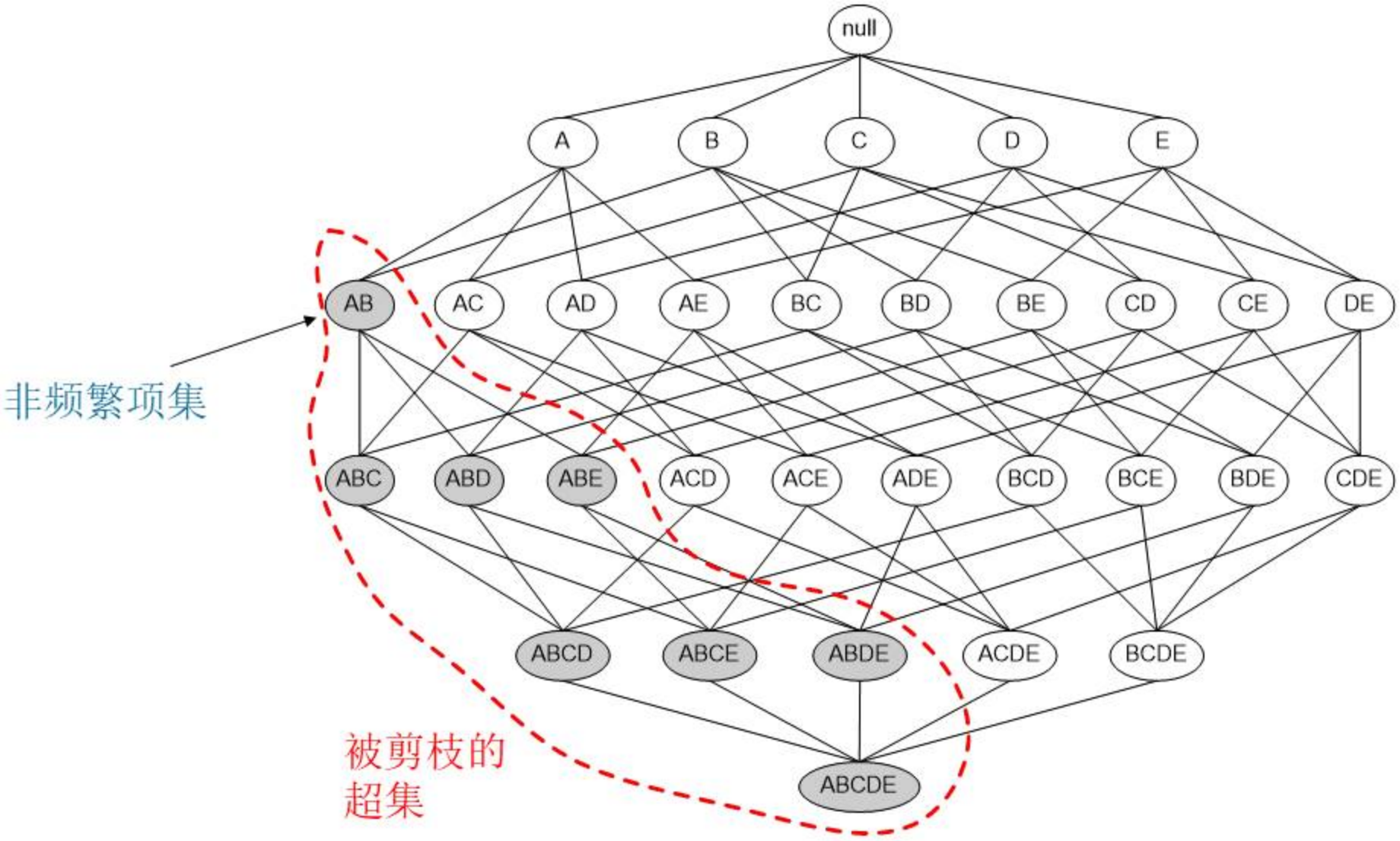
2降低产生频繁项集计算复杂度的方法

- 减少候选项集的数量
  - 先验原理:(Apriori)
- 减少比较的次数
  - 替代将每个候选项集与每个事务相匹配，可以使用更高级的数据结构，或存储候选项集或压缩数据集，来减少比较次数(FPGrowth)

2.1 Apriori

- 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
  - 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:

例子


null
A
B
C
D
E
AB
AC
AD
AE
BC
BD
BE
CD
CE
DE
ABC
ABD
ABE
ACD
ACE
ADE
BCD
BCE
BDE
CDE
ABCD
ABCE
ABDE
ACDE
BCDE
ABCDE
非频繁项集
被剪枝的
超集

2.1 Apriori算法过程：最小支持度计数=2

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

$Sup_{min} = 2$

C_1

1st scan

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan

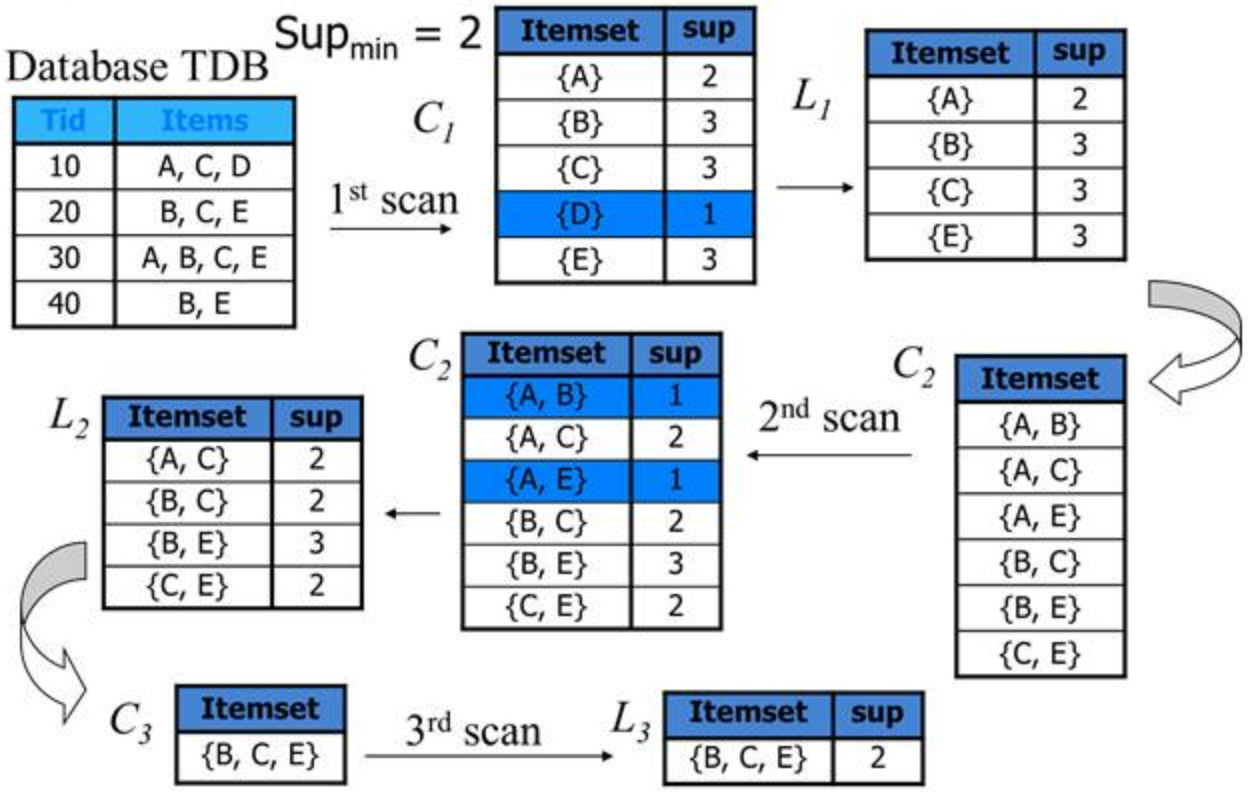
L_3

Itemset	sup
{B, C, E}	2

2.1 Apriori算法注意问题-项的字典序

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

尽管集合具有无序性，但为了方便比较计数，通常对所有商品做一个默认的排序（类似于建立一个字典索引）



2.1 Apriori算法注意问题-项的连接

对于任何2个需要连接的项集

A, B, C

B, C, E

2.1 Apriori算法注意问题-项的连接

对于任何2个需要连接的项集

A, B, C

B, C, E

去掉第1个项集的首项

2.1 Apriori算法注意问题-项的连接

对于任何2个需要连接的项集

A, B, C　去掉第1个项集的首项

B, C, E　去掉第2个项集的尾项

2.1 Apriori算法注意问题-项的连接

对于任何2个需要连接的项集

A, B, C　去掉第1个项集的首项

B, C, E　去掉第2个项集的尾项

若剩下的项一样 → B, C

B, C

2.1 Apriori算法注意问题-项的连接

对于任何2个需要连接的项集

A, B, C　去掉第1个项集的首项

B, C, E　去掉第2个项集的尾项

2.1 Apriori算法注意问题-项的连接

对于任何2个需要连接的项集

A, B, C　去掉第1个项集的首项
B, C, E　去掉第2个项集的尾项

若剩下的项一样 → B, C / B, C → 则可连接 → A, B, C, E

对于任何2个需要连接的项集

A, B, D　去掉第1个项集的首项
B, C, E　去掉第2个项集的尾项

若剩下的项不一样 → B, D / B, C → 则不可连接

2.1 Apriori算法注意问题-项的连接

对于任何2个需要连接的项集

A, B, C　去掉第1个项集的首项
B, C, E　去掉第2个项集的尾项

若剩下的项一样 → B, C / B, C → 则可连接 → A, B, C, E

对于任何2个需要连接的项集

A, B, D　去掉第1个项集的首项
B, C, E　去掉第2个项集的尾项

若剩下的项不一样 → B, D / B, C → 则不可连接

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

$Sup_{min} = 2$

C_1 (1st scan)

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

频繁2项集生成的候选3项集是什么？

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

设置

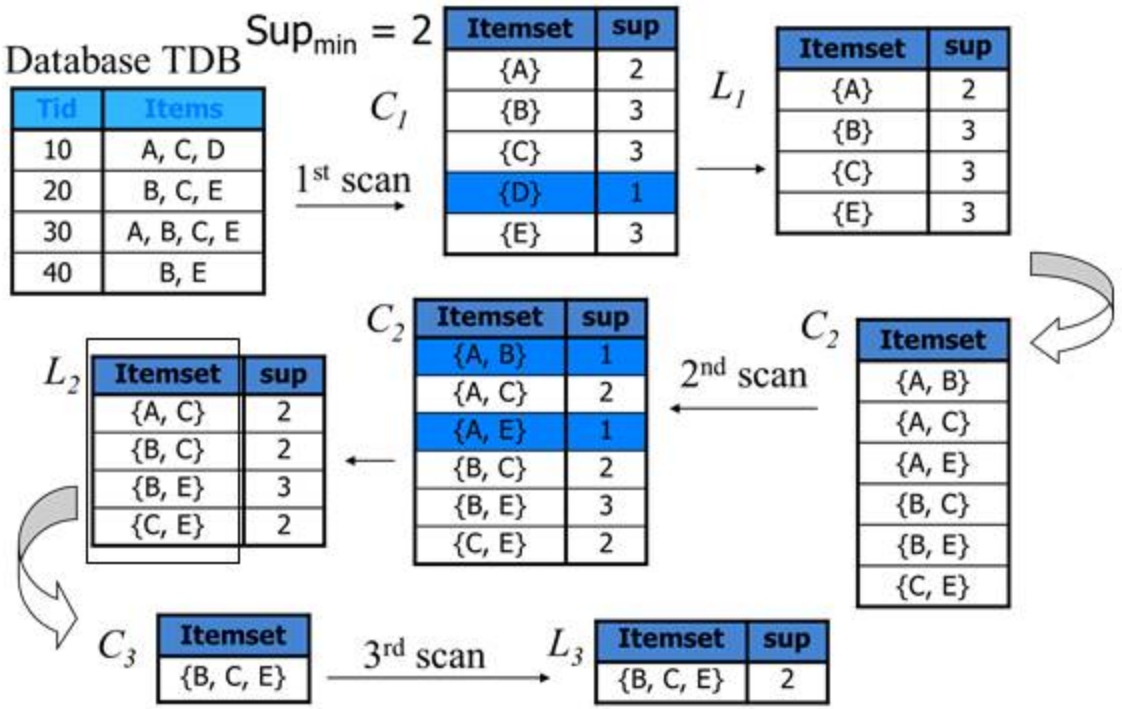
频繁2项集生成的候选3项集是什么？

A A,B,C

B A,C,E

C A,B,E

D B,C,E



提交

2.1 Apriori算法注意问题-项的连接

对于任何2个需要连接的项集

A, B, C　去掉第1个项集的首项
B, C, E　去掉第2个项集的尾项

若剩下的项一样 → B, C / B, C → 则可连接 → A, B, C, E

对于任何2个需要连接的项集

A, B, D　去掉第1个项集的首项
B, C, E　去掉第2个项集的尾项

若剩下的项不一样 → B, D / B, C → 则不可连接

~~A, B, C~~
A, C, E
~~A, B, E~~
B, C, E

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

$Sup_{min} = 2$

C_1 (1st scan)

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

2.1 Apriori算法注意问题-项的连接

对于任何2个需要连接的项集

A, B, C　去掉第1个项集的首项
B, C, E　去掉第2个项集的尾项

若剩下的项一样 → B, C / B, C → 则可连接 → A, B, C, E

对于任何2个需要连接的项集

A, B, D　去掉第1个项集的首项
B, C, E　去掉第2个项集的尾项

若剩下的项不一样 → B, D / B, C → 则不可连接

项的连接准则的优点是降低候选项的生成

~~A, B, C~~
A, C, E
~~A, B, E~~
B, C, E

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

$Sup_{min} = 2$

C_1 (1st scan)

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

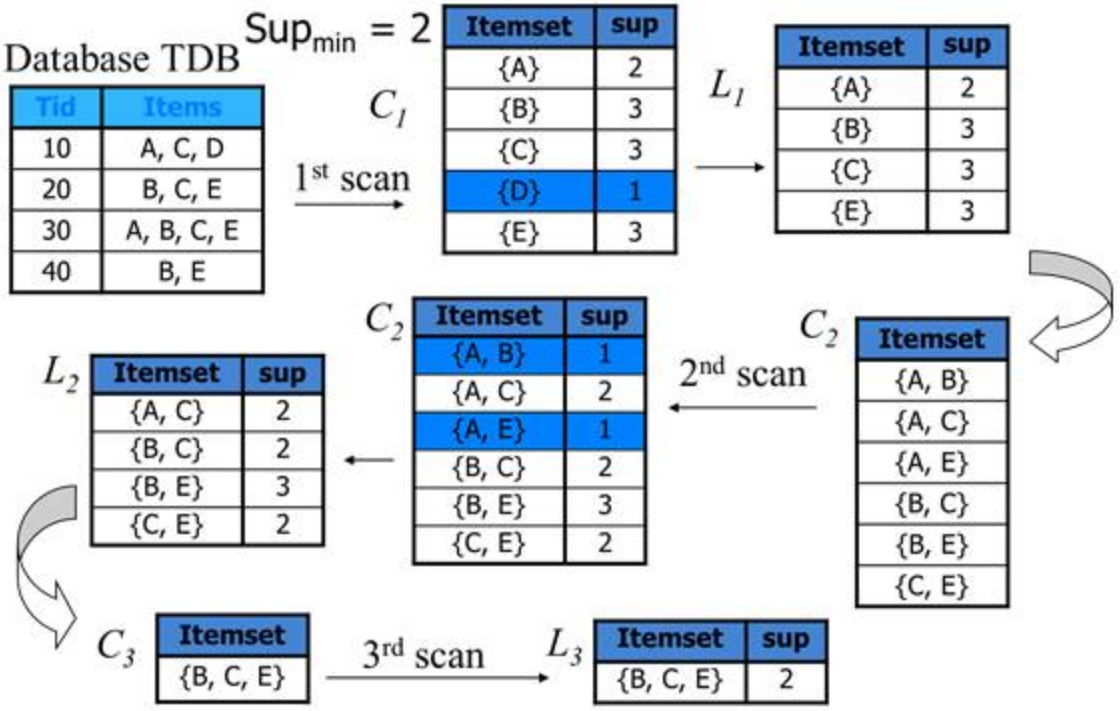
3rd scan

L_3

Itemset	sup
{B, C, E}	2

2.1Apriori算法特点

- 多次扫描数据库
- 候选项规模庞大
- 计算支持度开销大

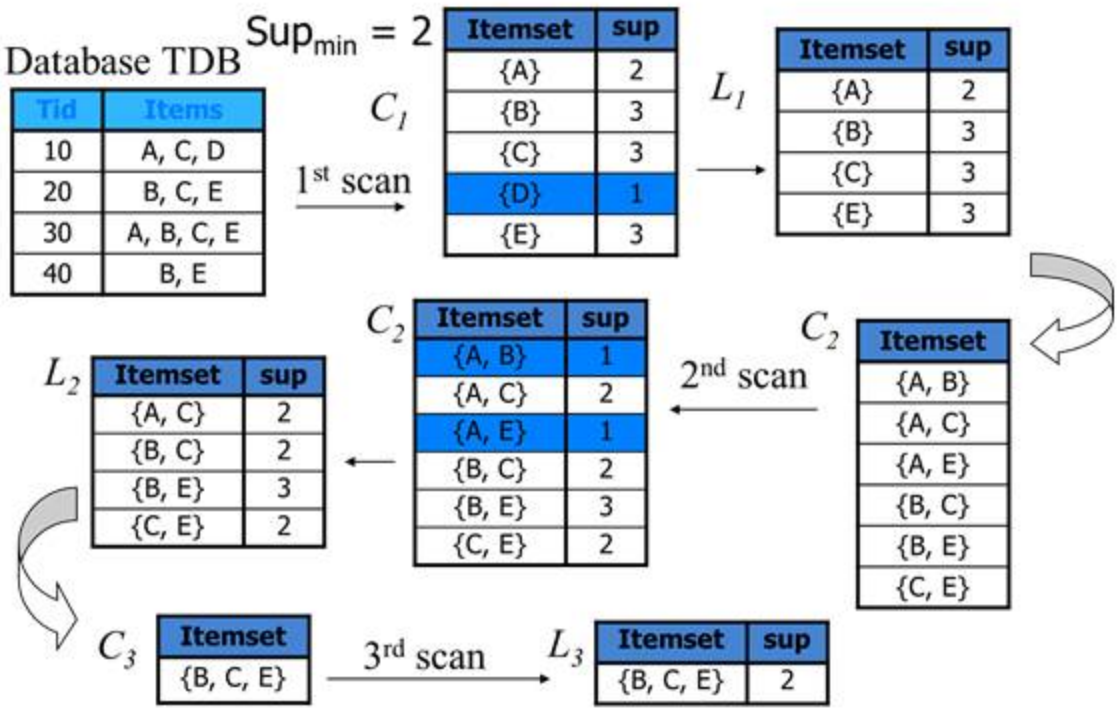


2.1提高Apriori算法性能的方法

- Hash-based itemset counting（散列项集计数）
- Transaction reduction（事务压缩）
- Partitioning（划分）
- Sampling（采样）

2.1Apriori算法缺点

- Apriori算法需要反复的生成候选项，如果项的数目比较大，候选项的数目将达到**组合爆炸式的增长**



2.2 FPGrowth

- 背景
  - 韩家炜等人，2000年
- 基本思想
  - 只扫描数据库两遍，构造频繁模式树（FP-Tree）
  - 自底向上递归产生频繁项集
  - FP树是一种输入数据的压缩表示，它通过逐个读入事务，并把每个事务映射到FP树中的一条路径来构造。

2.2构造FP树：第一遍扫描

原始事物

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

删除支持度小于2的项

Item	Count
A	8
B	7
C	6
D	5
E	3
F	1
G	1

假设事先指定最小支持度计数为2

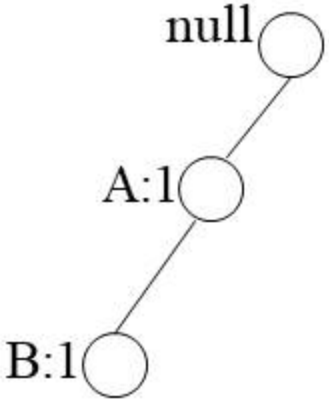
Items按照出现次数降序排列

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	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

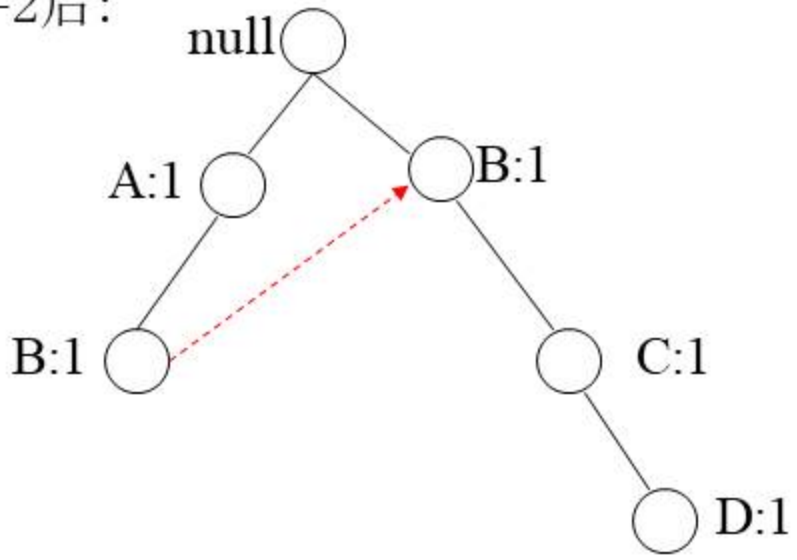
2.2构造FP树：第二遍扫描

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	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

读入事务 TID=1后：



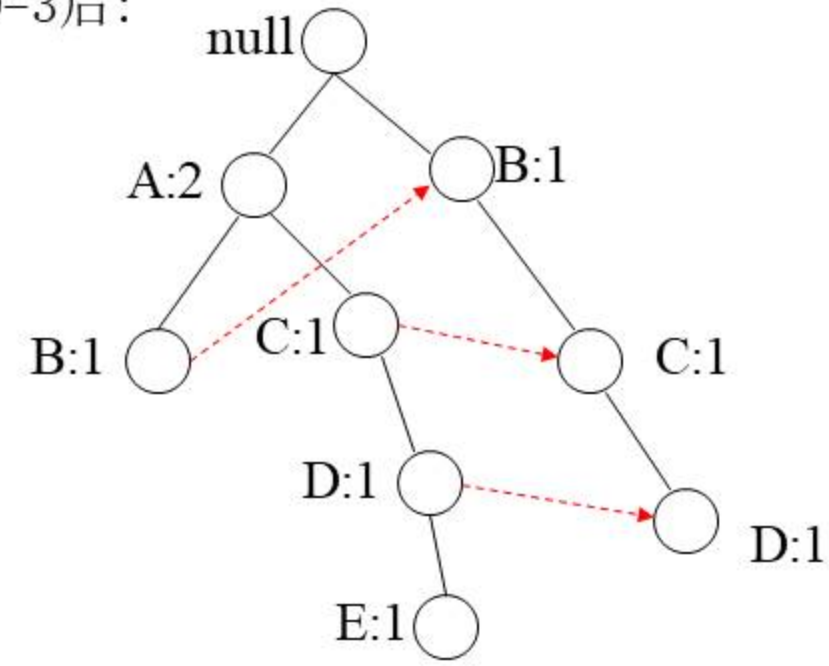
读入事务 TID=2后：



2.2构造FP树：第二遍扫描

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	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

读入事务 TID=3后：

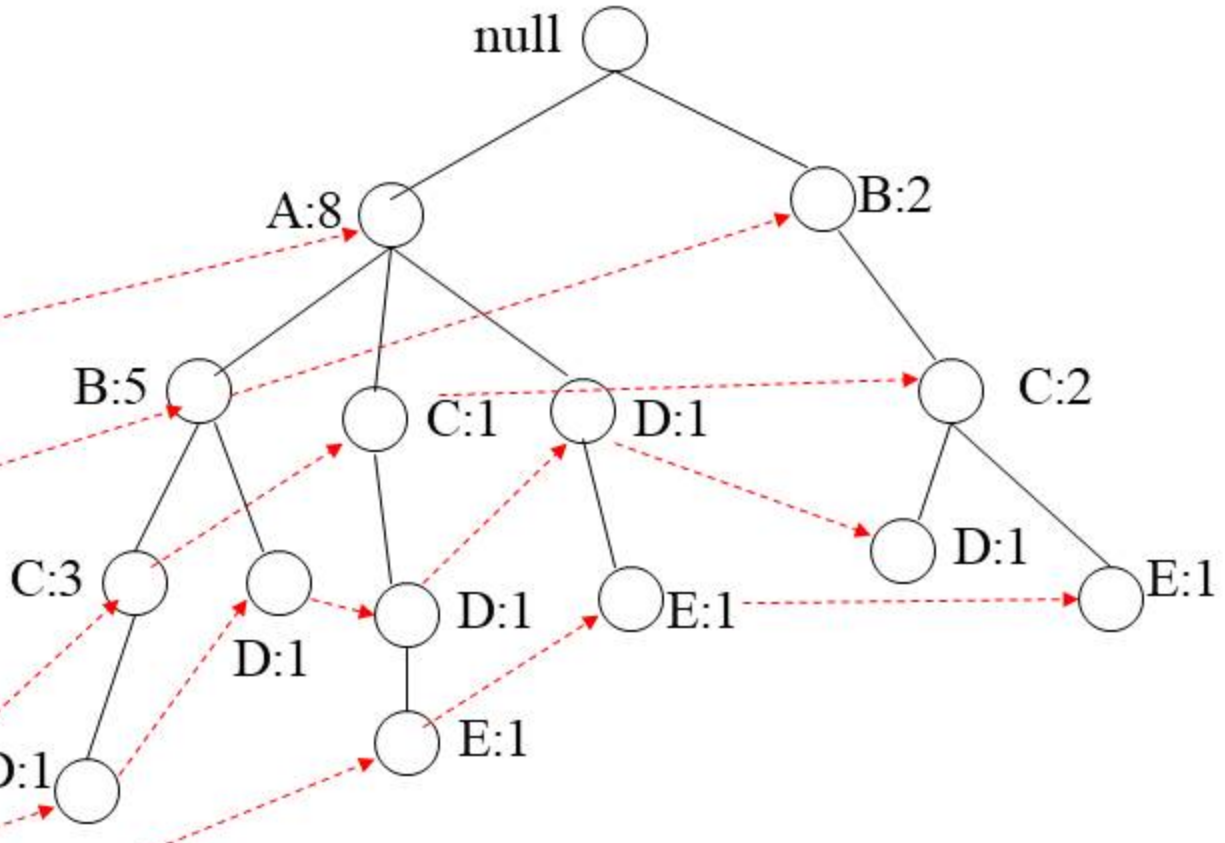


2.2构造FP树：第二遍扫描

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	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Header table

Item	Pointer
A	
B	
C	
D	
E	



2.2构造FP树：用FP-tree挖掘频繁集

- 基本思想 (分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的条件模式基, 然后生成它的条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空, 或只含唯一的一个路径 (此路径的每个子路径对应的项集都是频繁集)

练习：构造FP树

TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

最小支持度= 0.5

步骤:

1. 扫描数据库一次，得到频繁1-项集
2. 把项按支持度递减排序
3. 再一次扫描数据库，建立FP-tree

头表

Item	frequency	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

练习：构造FP树

TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

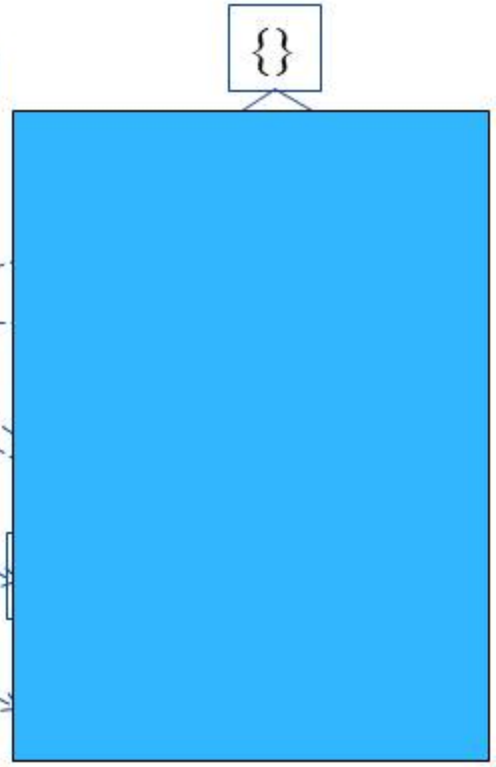
最小支持度= 0.5

步骤:

1. 扫描数据库一次，得到频繁1-项集
2. 把项按支持度递减排序
3. 再一次扫描数据库，建立FP-tree

头表

Item	frequency	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



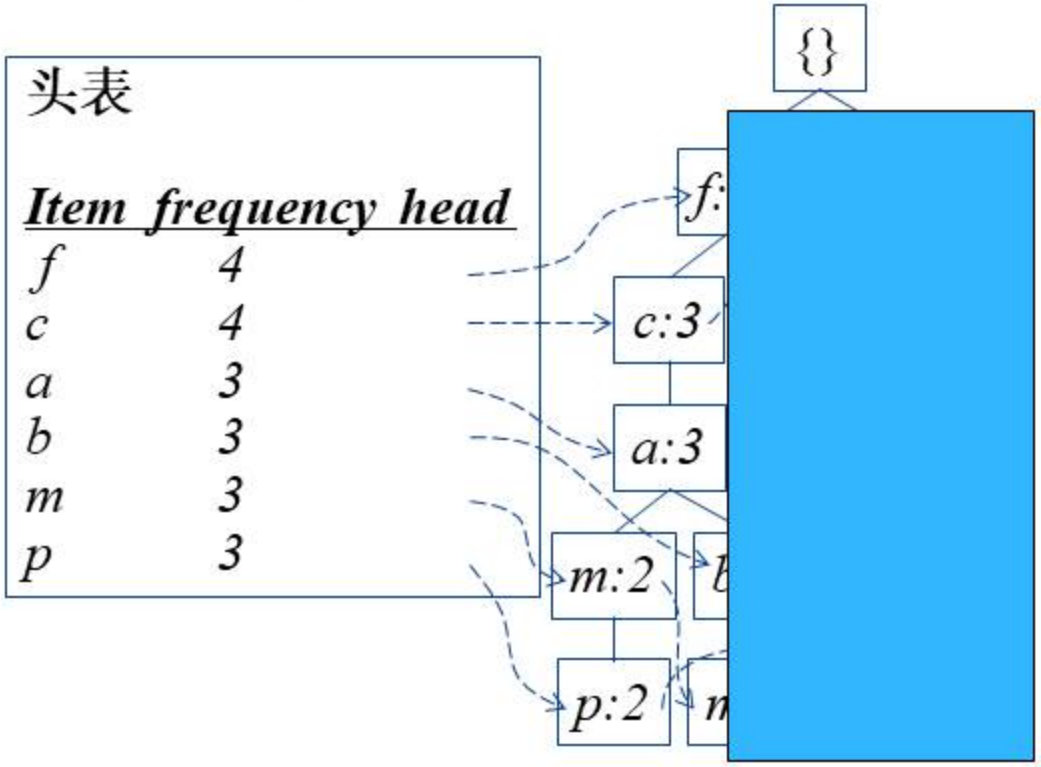
练习：构造FP树

TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

最小支持度= 0.5

步骤:

1. 扫描数据库一次，得到频繁1-项集
2. 把项按支持度递减排序
3. 再一次扫描数据库，建立FP-tree



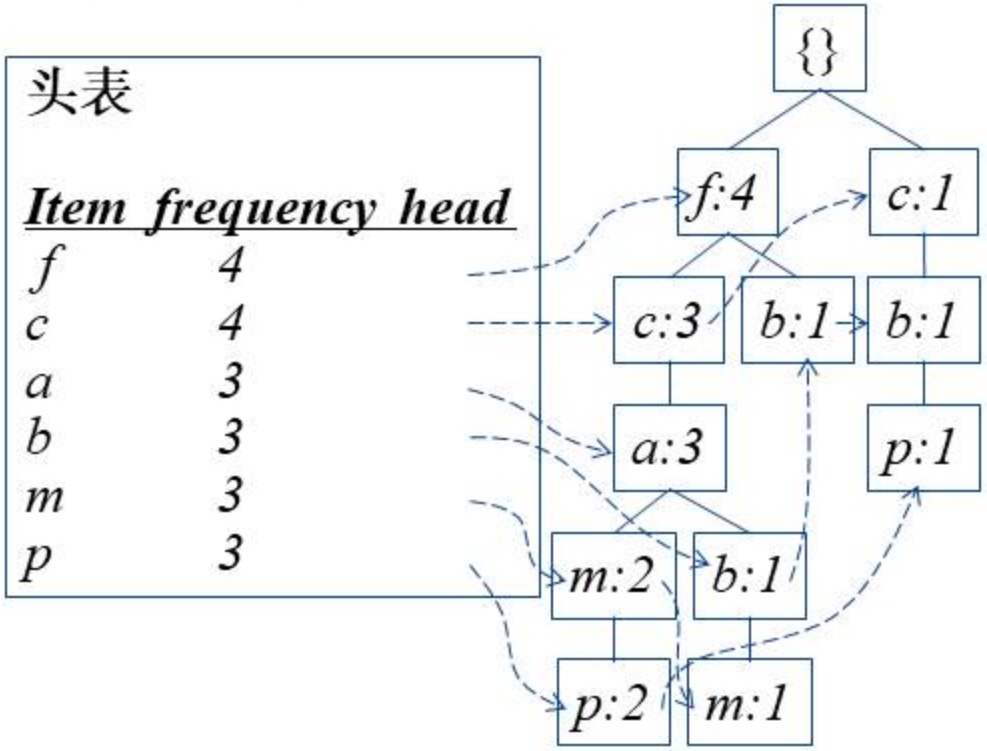
练习：构造FP树

TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

最小支持度= 0.5

步骤:

1. 扫描数据库一次，得到频繁1-项集
2. 把项按支持度递减排序
3. 再一次扫描数据库，建立FP-tree

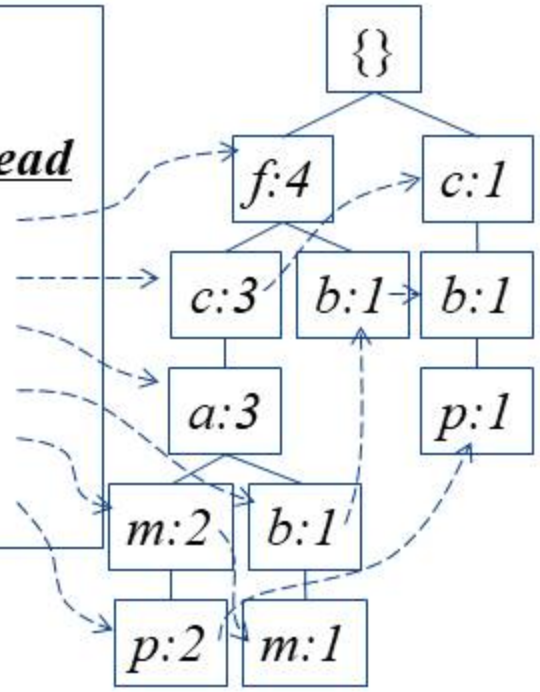


练习：生成条件模式

- 从FP-tree的头表开始
- 按照每个频繁项的连接遍历 FP-tree
- 列出能够到达此项的所有前缀路径，得到条件模式基

头表

Item	frequency	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



条件模式库

item	cond. pattern base
c	f:3
a	fc:3
b	fca:1, f:1, c:1
m	fca:2, fcab:1

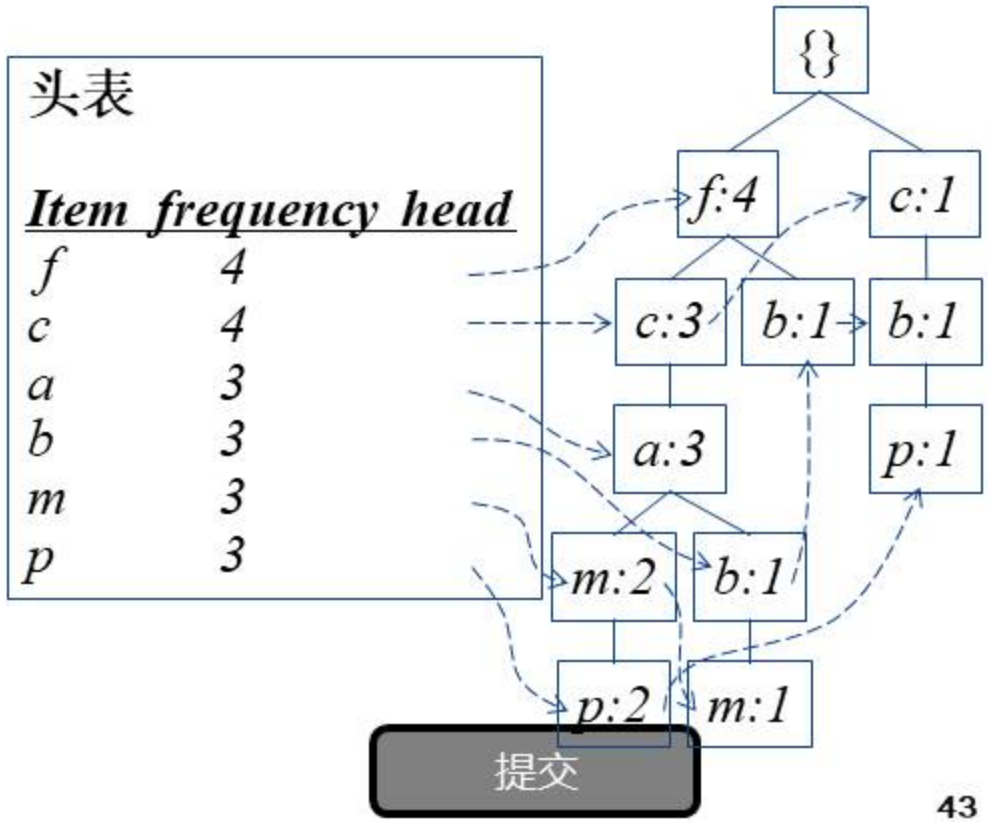
P的条件模式基为：

A f:2,c:2,a:2,m:2,p :2

B f:2,c:1,a:1,m:1,p :1

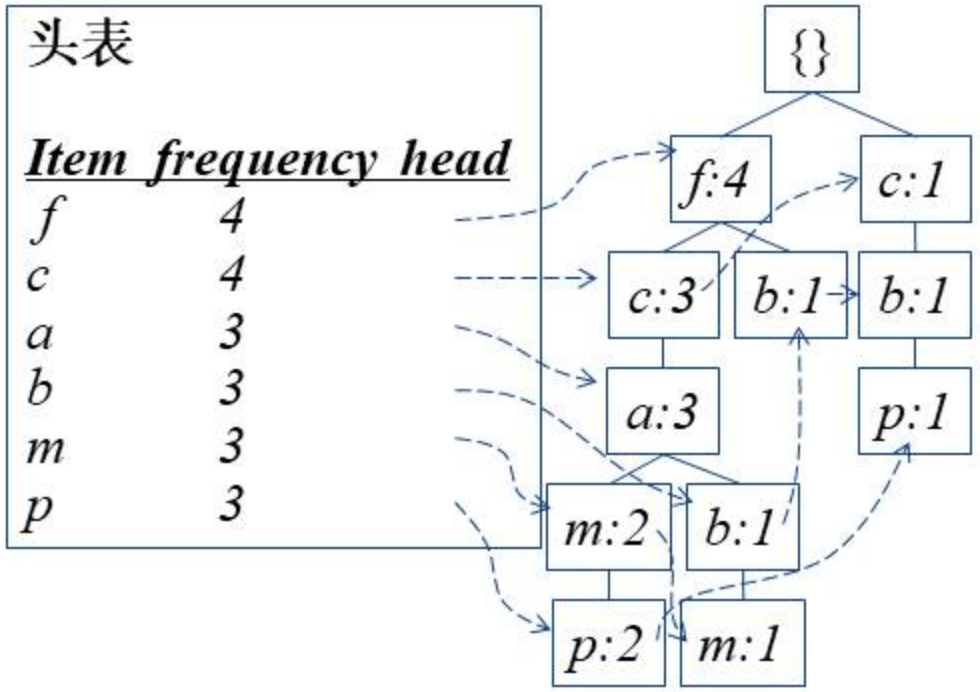
C f:1,c:1,a:1,m:1,p :1

D c:1,b:1,p :1



练习：生成条件模式

- 从FP-tree的头表开始
- 按照每个频繁项的连接遍历 FP-tree
- 列出能够到达此项的所有前缀路径，得到条件模式基



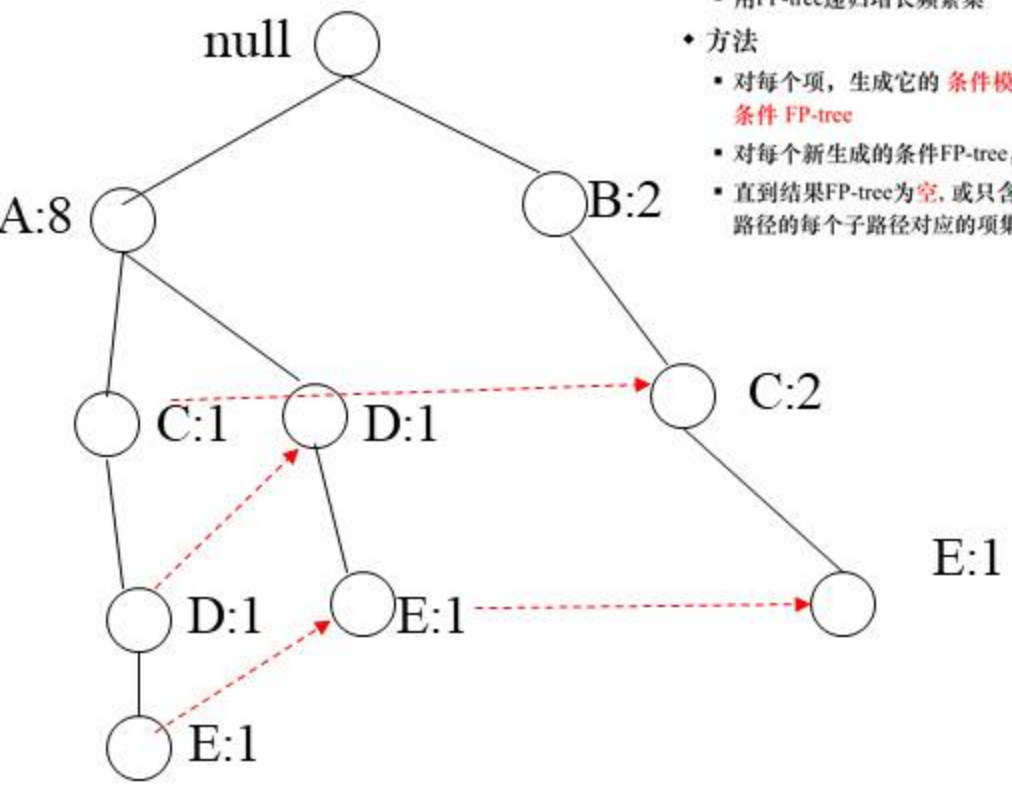
条件模式库

item	*cond. pattern base*
c	*f:3*
a	*fc:3*
b	*fca:1, f:1, c:1*
m	*fca:2, fcab:1*
p	*fcam:2, cb:1*

2.2构造FP树：用FP-tree挖掘频繁集

- 基本思想 (分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的条件模式基, 然后生成它的条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空, 或只含唯一的一个路径 (此路径的每个子路径对应的项集都是频繁集)

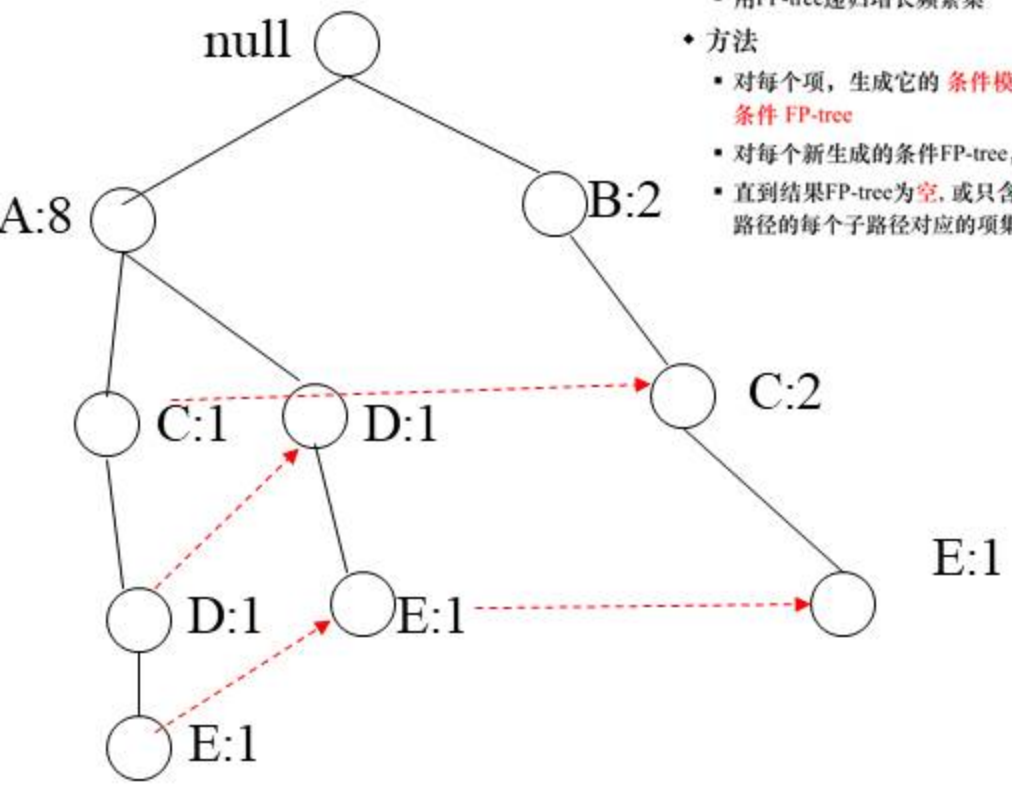
举例：节点E（1/3）



- 基本思想(分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的 条件模式基，然后生成它的 条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空，或只含唯一的一个路径（此路径的每个子路径对应的项集都是频繁集）

后缀模式 (PostModel)
E

举例：节点E（1/3）



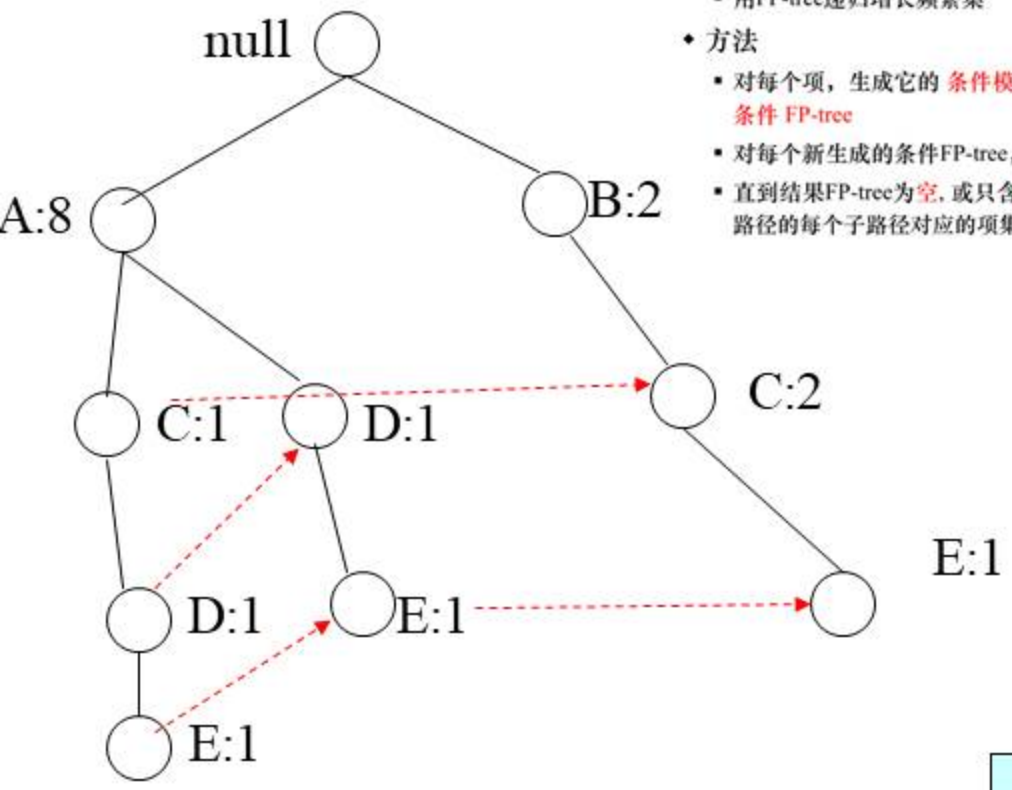
- 基本思想(分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的 条件模式基，然后生成它的 条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空，或只含唯一的一个路径（此路径的每个子路径对应的项集都是频繁集）

Items
{A:8,C:1,D:1,E:1}
{A:8,D:1,E:1}
{B:2,C:2,E:1}

后缀模式 (PostModel)
E

举例：节点E（1/3）

- 基本思想(分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的 条件模式基，然后生成它的 条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空，或只含唯一的一个路径(此路径的每个子路径对应的项集都是频繁集)



Items
{A:8,C:1,D:1,E:1}
{A:8,D:1,E:1}
{B:2,C:2,E:1}

Items
{A:1,C:1,D:1,E:1}
{A:1,D:1,E:1}
{B:1,C:1,E:1}

后缀模式 (PostModel)
E

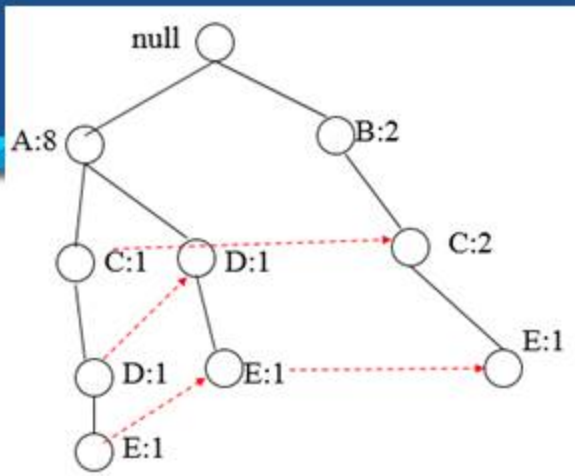
条件模式基 (Conditional Pattern Base,CPB)
{A:1,C:1,D:1}
{A:1,D:1}
{B:1,C:1}

举例：节点E（2/3）

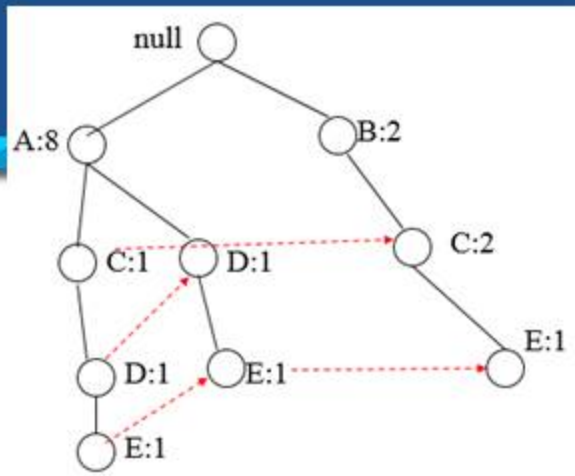
- 基本思想(分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的 条件模式基，然后生成它的 条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空，或只含唯一的一个路径 (此路径的每个子路径对应的项集都是频繁集)

条件模式基
{A:1,C:1,D:1}
{A:1,D:1}
{B:1,C:1}

表头项
A:2
C:2
D:2



举例：节点E（2/3）



- 基本思想(分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的 条件模式基，然后生成它的 条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空，或只含唯一的一个路径 (此路径的每个子路径对应的项集都是频繁集)

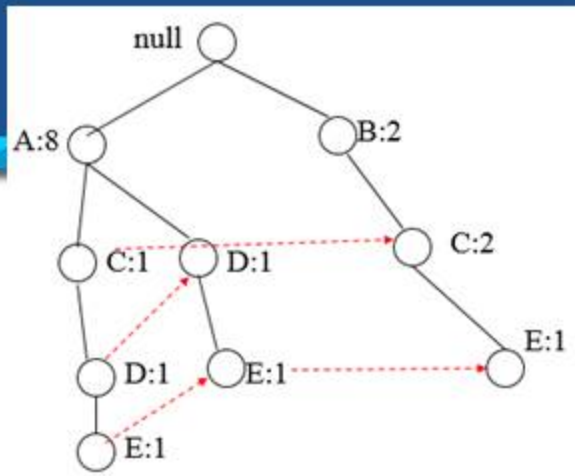
条件模式基
{A:1,C:1,D:1}
{A:1,D:1}
{B:1,C:1}

表头项
A:2
C:2
D:2

后缀模式 (PostModel)
E

频繁项集
AE:2
CE:2
DE:2
E:3

举例：节点E（2/3）



- 基本思想(分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的 条件模式基，然后生成它的 条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空，或只含唯一的一个路径 (此路径的每个子路径对应的项集都是频繁集)

条件模式基
{A:1,C:1,D:1}
{A:1,D:1}
{B:1,C:1}

表头项
A:2
C:2
D:2

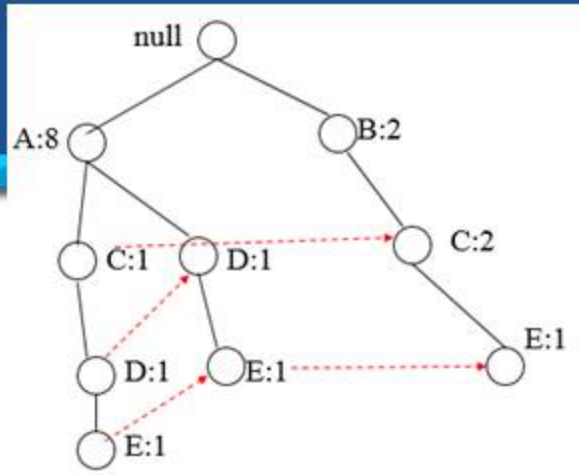
后缀模式 (PostModel)
E

频繁项集
AE:2
CE:2
DE:2
E:3

新的后缀模式
AE
CE
DE

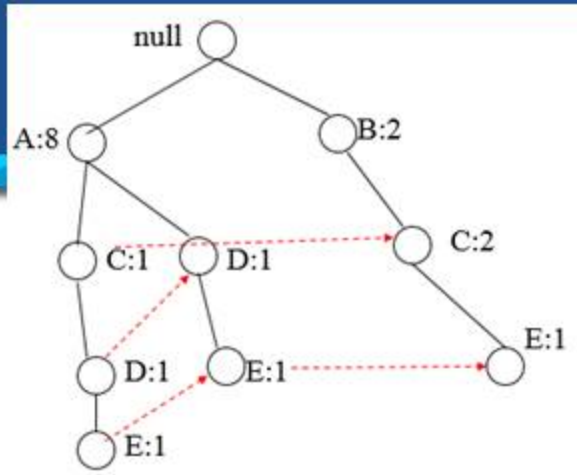
举例：节点E（3/3）

新的后缀模式
AE
CE
DE



- 基本思想(分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的 条件模式基，然后生成它的 条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空，或只含唯一的一个路径 (此路径的每个子路径对应的项集都是频繁集)

举例：节点E（3/3）



新的后缀模式
AE
CE
DE

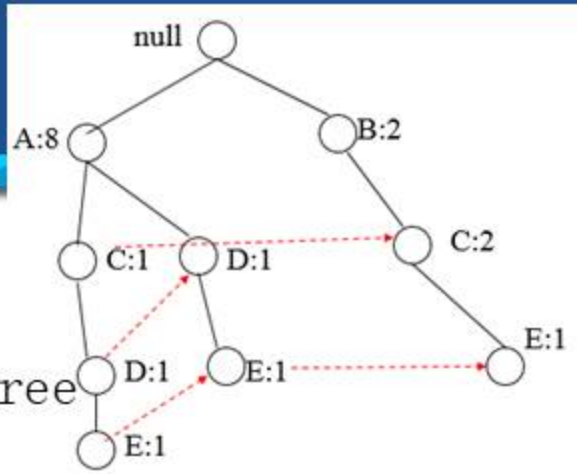
Items
{A:1,C:1,D:1,E:1}
{A:1,D:1,E:1}
{B:1,C:1,E:1}

- 基本思想(分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的 条件模式基，然后生成它的 条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空，或只含唯一的一个路径 (此路径的每个子路径对应的项集都是频繁集)

举例：节点E（3/3）

新的后缀模式
AE
CE
DE

Items
{A:1,C:1,D:1,E:1}
{A:1,D:1,E:1}
{B:1,C:1,E:1}

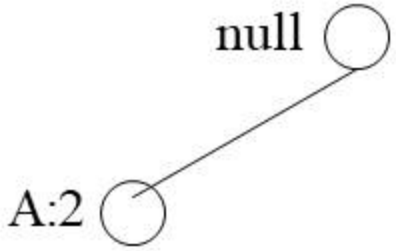


null

后缀模式为AE的FP-Tree

null

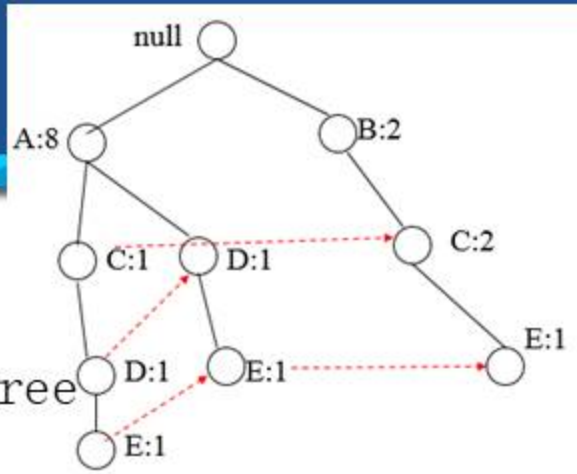
后缀模式为CE的FP-Tree



后缀模式为DE的FP-Tree

- 基本思想(分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的 条件模式基，然后生成它的 条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空，或只含唯一的一个路径 (此路径的每个子路径对应的项集都是频繁集)

举例：节点E（3/3）



新的后缀模式
AE
CE
DE

Items
{A:1,C:1,D:1,E:1}
{A:1,D:1,E:1}
{B:1,C:1,E:1}

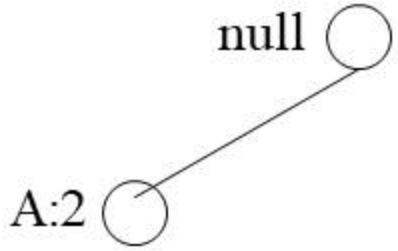
频繁项集
ADE:2
AE:2
CE:2
DE:2
E:3

null

后缀模式为AE的FP-Tree

null

后缀模式为CE的FP-Tree



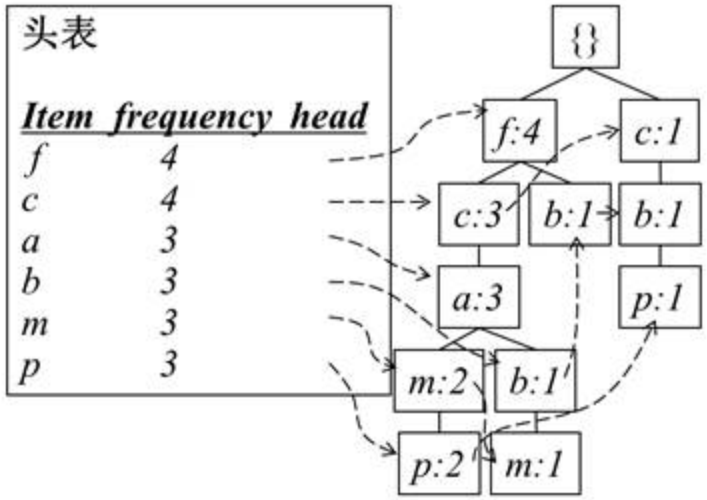
后缀模式为DE的FP-Tree

- 基本思想(分治)
  - 用FP-tree递归增长频繁集
- 方法
  - 对每个项，生成它的 条件模式基，然后生成它的 条件 FP-tree
  - 对每个新生成的条件FP-tree，重复这个步骤
  - 直到结果FP-tree为空，或只含唯一的一个路径 (此路径的每个子路径对应的项集都是频繁集)

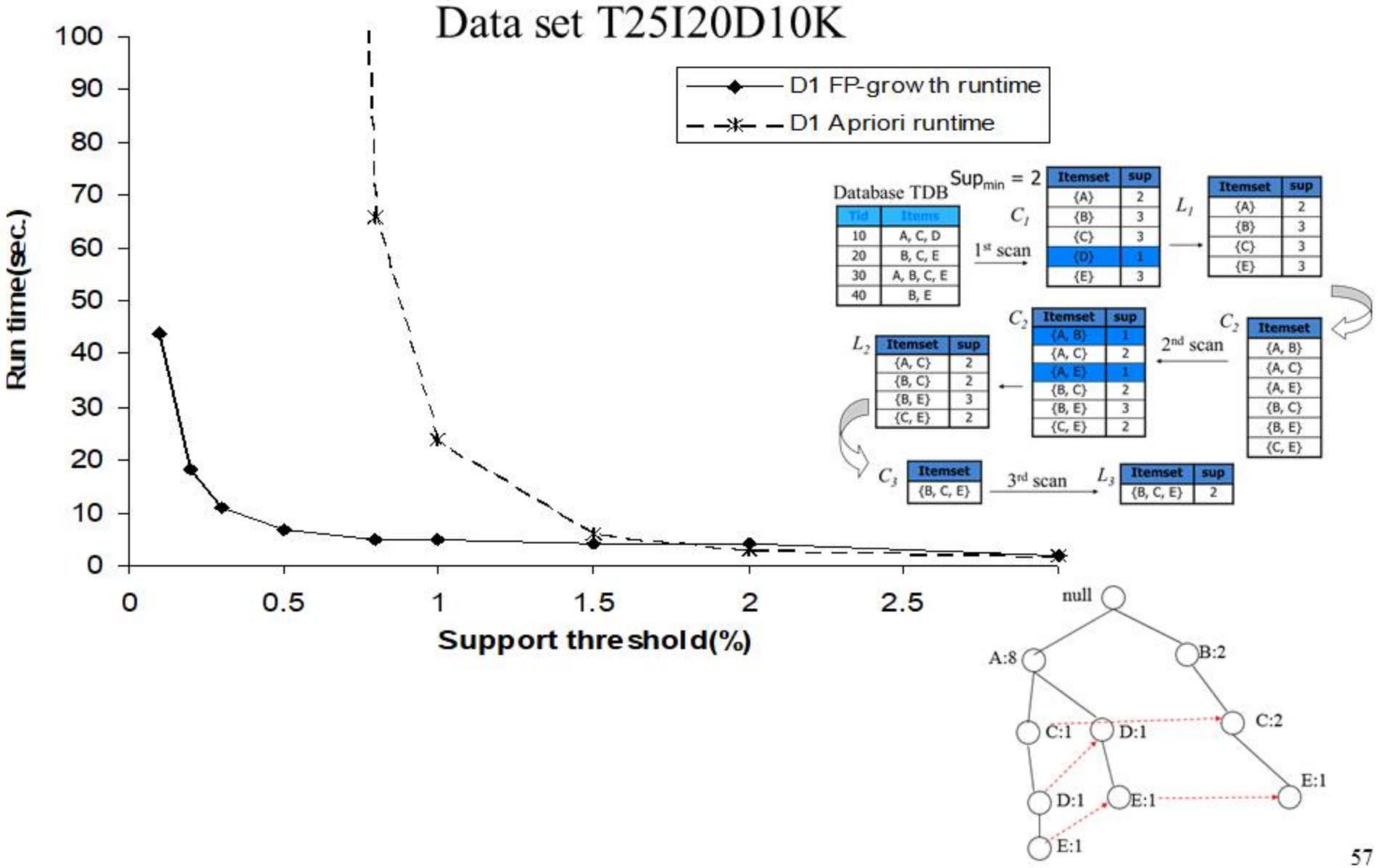
2.2FP-tree 结构的优点

- 完备:
  - 不会打破交易中的任何模式
  - 包含了频繁模式挖掘所需的全部信息
- 紧密
  - 支持度降序排列: 支持度高的项在FP-tree中共享的机会也高
  - 决不会比原数据库大

https://www.cnblogs.com/itbuyixiaogong/p/9077428.html



2.2FP-tree 结构的优点-性能对比



2.3挖掘关联规则（Mining Association Rules）

- 大多数关联规则挖掘算法通常采用的一种策略是，将关联规则挖掘任务分解为如下两个主要的子任务：

1. 频繁项集产生（Frequent Itemset Generation）
   - 其目标是发现满足最小支持度阈值的所有项集，这些项集称作频繁项集。
2. 规则的产生（Rule Generation）
   - 其目标是从上一步发现的频繁项集中提取所有高置信度的规则，这些规则称作强规则（strong rule）。

2.3产生关联规则

- 任务描述：给定频繁项集Y，查找Y的所有非空真子集$X \subset Y$，使得 $X \rightarrow Y - X$ 的置信度超过最小置信度阈值*minconf*
  - 例子：If {A,B,C} is a frequent itemset, 候选规则如下：

$$AB \rightarrow C, \quad AC \rightarrow B, \quad BC \rightarrow A$$
$$A \rightarrow BC, \quad B \rightarrow AC, \quad C \rightarrow AB$$

- 如果 $|Y| = k$, 那么会有 $2^k - 2$ 个候选关联规则（不包括 $Y \rightarrow \varnothing$ and $\varnothing \rightarrow Y$）

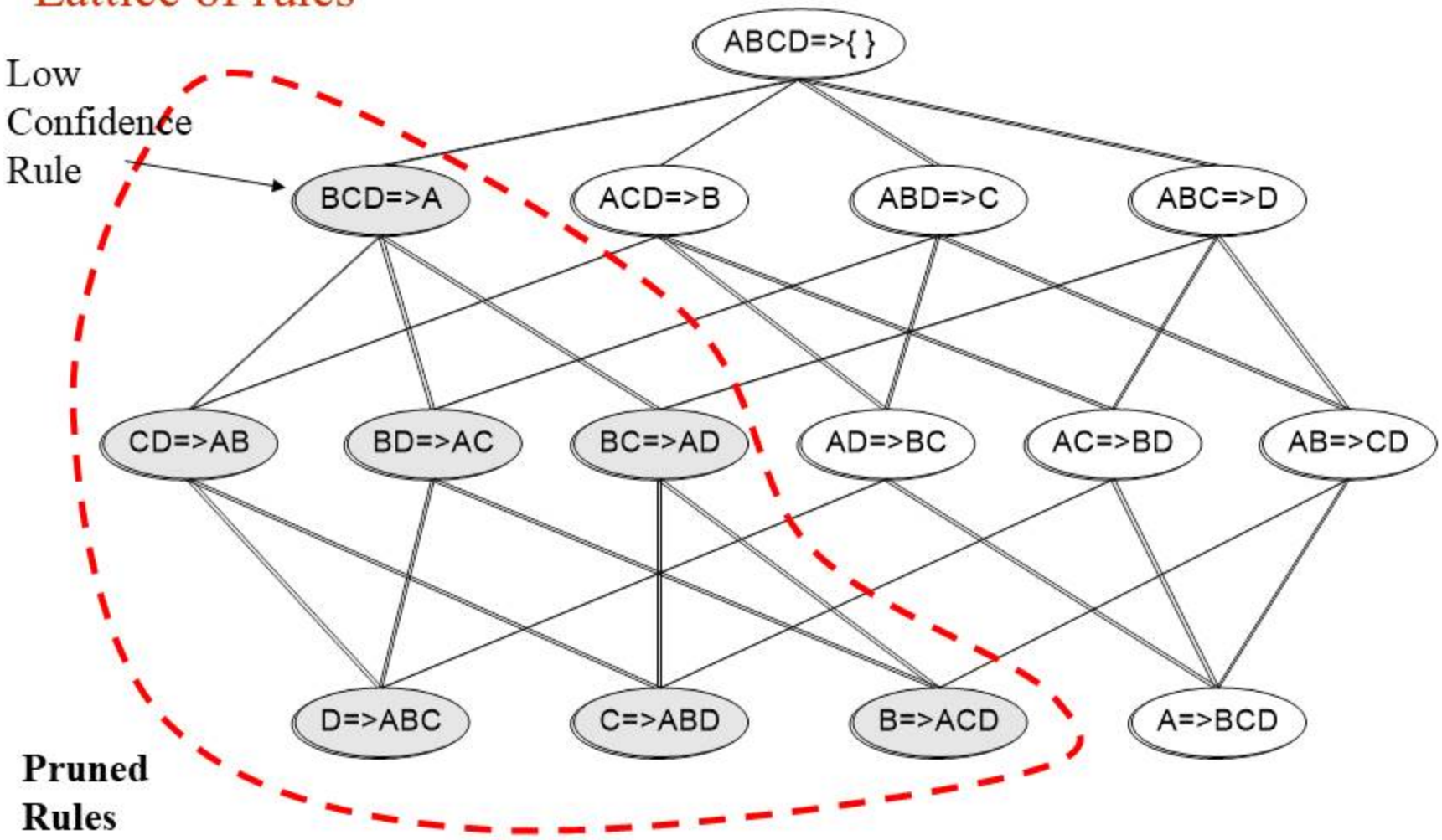
2.3产生关联规则

- How to efficiently generate rules from frequent itemsets?
  - 通常，置信度不满足反单调性（anti-monotone property），例如：

    c(ABC →D) 可能大于也可能小于 c(AB →D)
  - 但是，针对同一个频繁项集的关联规则，如果规则的后件满足子集关系，那么这些规则的置信度间满足反单调性
  - e.g., Y= {A,B,C,D}:

    $c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$

2.3Rule Generation for Apriori Algorithm



主要内容

- 1. 基于概念
- 2. 频繁项挖掘算法
- 3. 关联分析的评估

3. 关联分析的评估（Pattern Evaluation）

- *play basketball* ⇒ *eat cereal* [支持度=, 置信度=]

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

3. 关联分析的评估（Pattern Evaluation）

- *play basketball* ⇒ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%.
- *play basketball* ⇒ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: lift

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

$$lift(B, C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89$$

$$lift(B, \neg C) = \frac{1000/5000}{3000/5000 * 1250/5000} = 1.33$$

主要内容

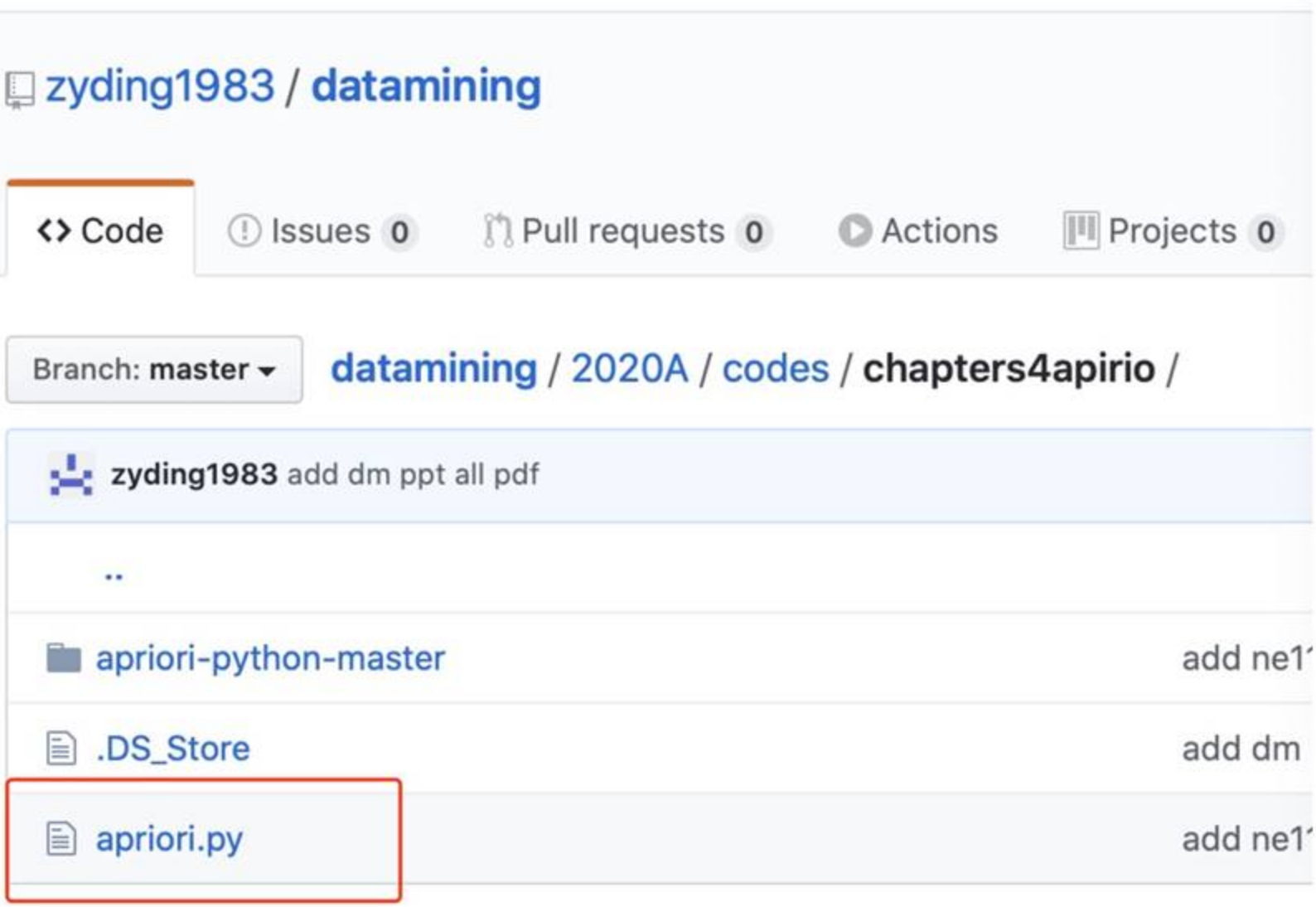
- 1. 基于概念
- 2. 频繁项挖掘算法
- 3. 关联分析的评估

问题？

第5次课后作业

- 第五次课后作业-在educoder平台上完成作业
- https://www.educoder.net/shixuns/egbxla2s/challenges
- https://www.educoder.net/shixuns/7ctykufv/challenges

提交作业截至时间：2020年3月05日


zyding1983 / datamining
Code
Issues 0
Pull requests 0
Actions
Projects 0
Branch: master
datamining / 2020A / codes / chapters4apirio /
zyding1983 add dm ppt all pdf
..
apriori-python-master
add ne1
.DS_Store
add dm
apriori.py
add ne1

```
69
70          U = []
71          for f in F:
72              for x in f:
73                  U.append(x)
74          return U
75
76
77      T = [['A','C','D'],['B','C','E'],['A','B','C','E'],['B','E']]
78
79      Z= apriori(T,3)
80      print(Z)
```

apriori() › for f in D[0] › if C[f]>=minSupport

un: apriori

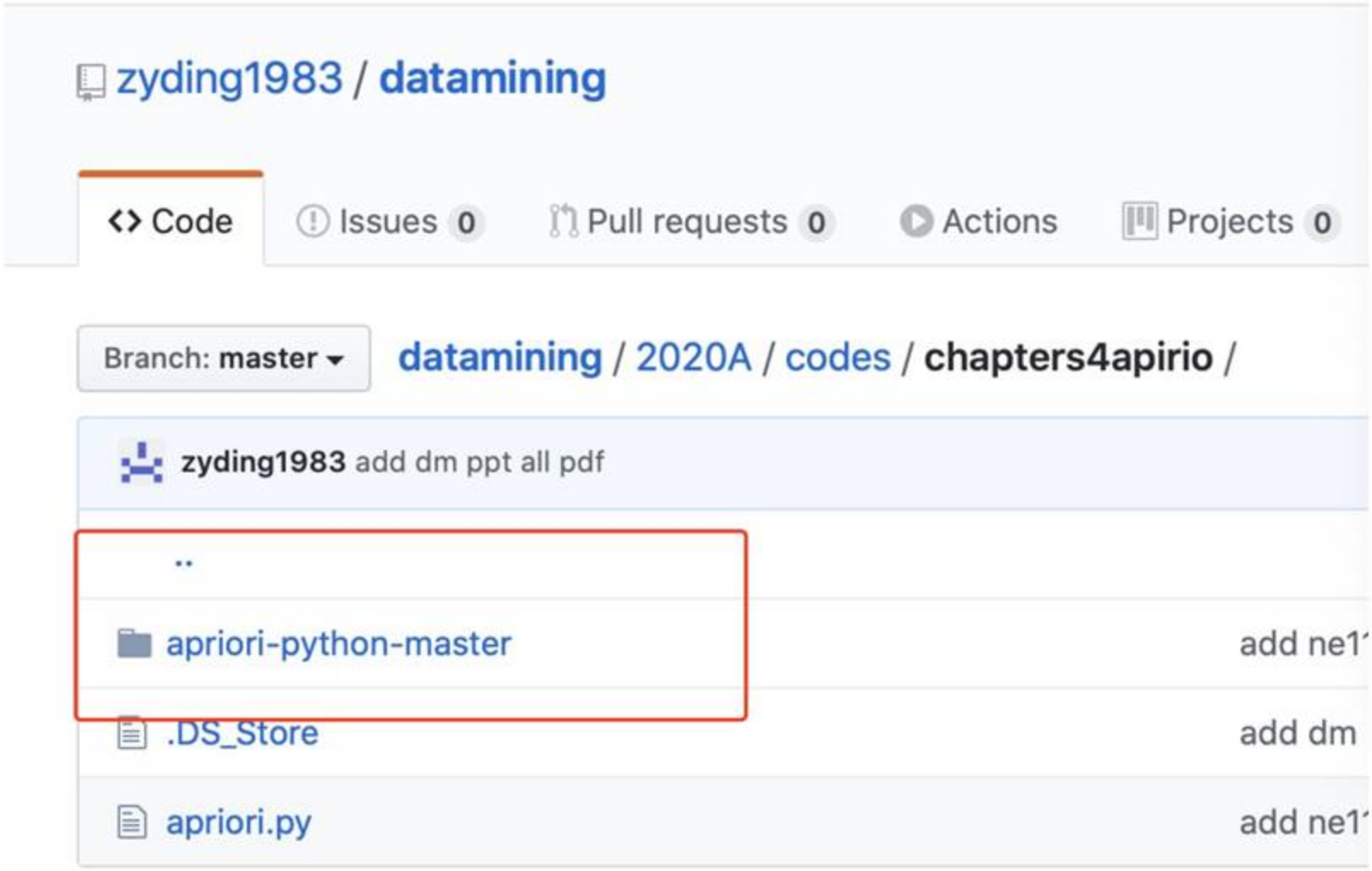
```
/Users/zyding/venv/bin/python /Users/zyding/IdeaProjects/python1/untitled/apr
[['B'], ['C'], ['E'], ['B', 'E']]

Process finished with exit code 0
```

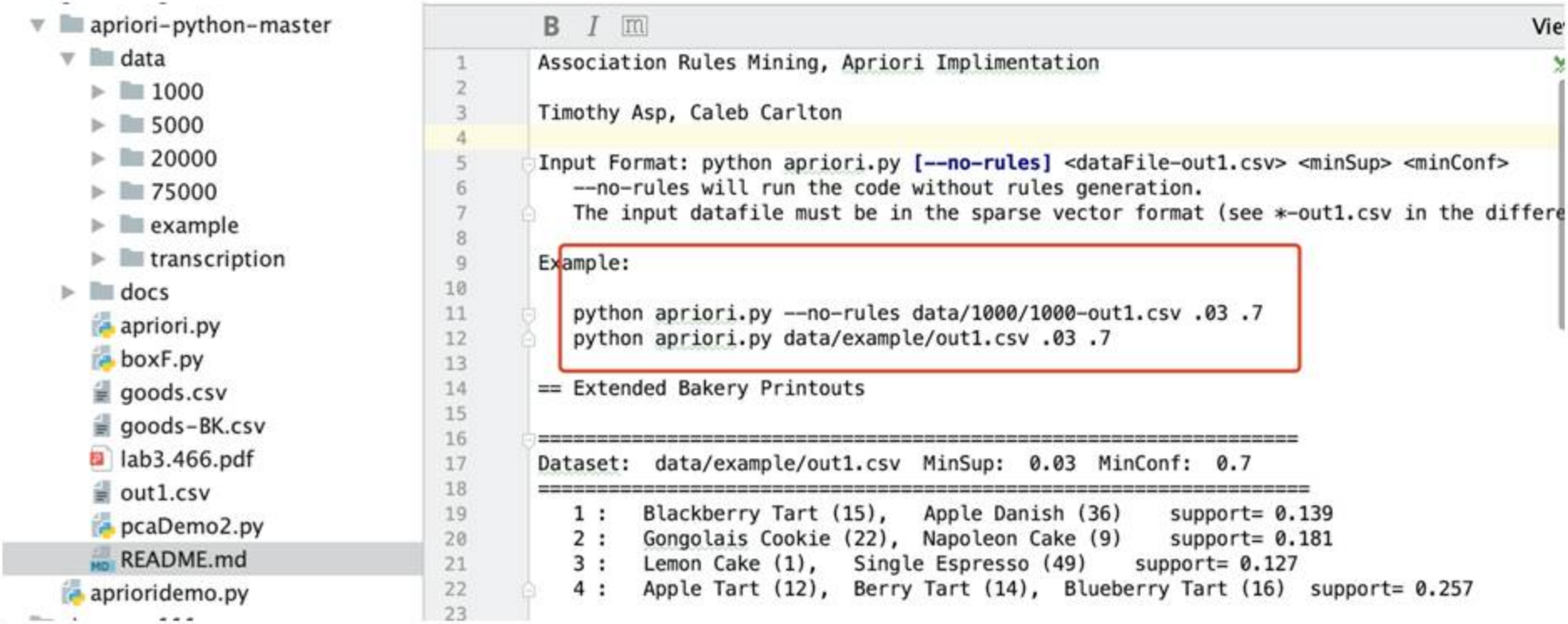
```
            F.append([])
            for c in D[k]:
                count = 0;
                for t in T:
                    if compareList(c,t):
                        count += 1
                if count>= minSupport:
                    F[k].append(c)
            k += 1

        U = []
        for f in F:
            for x in f:
                U.append(x)
        return U


T = [['A','C','D'],['B','C','E'],['A','B','C','E'],['B','E']]

Z= apriori(T,2)
print(Z)
```

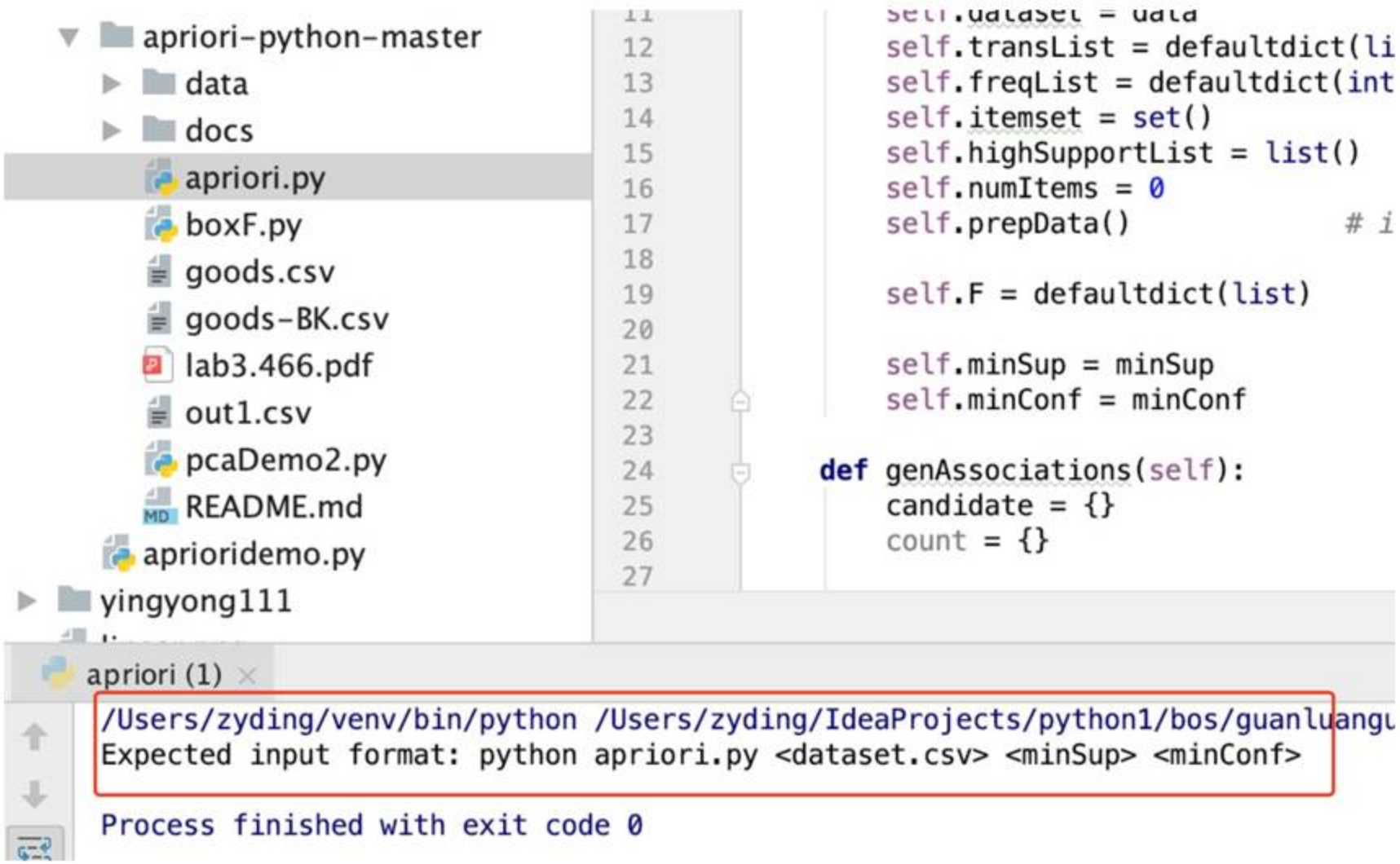
apriori()

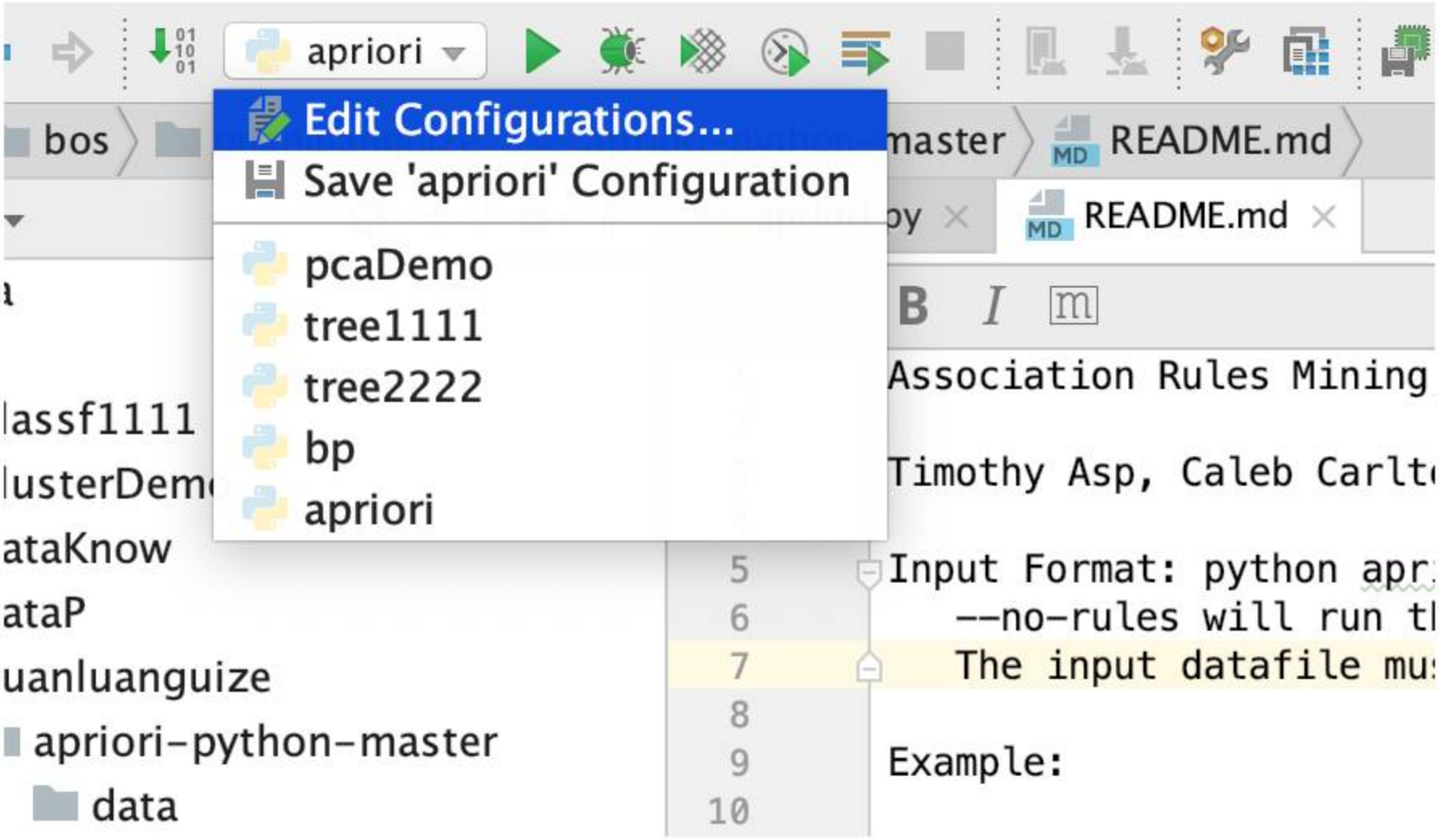
un: apriori

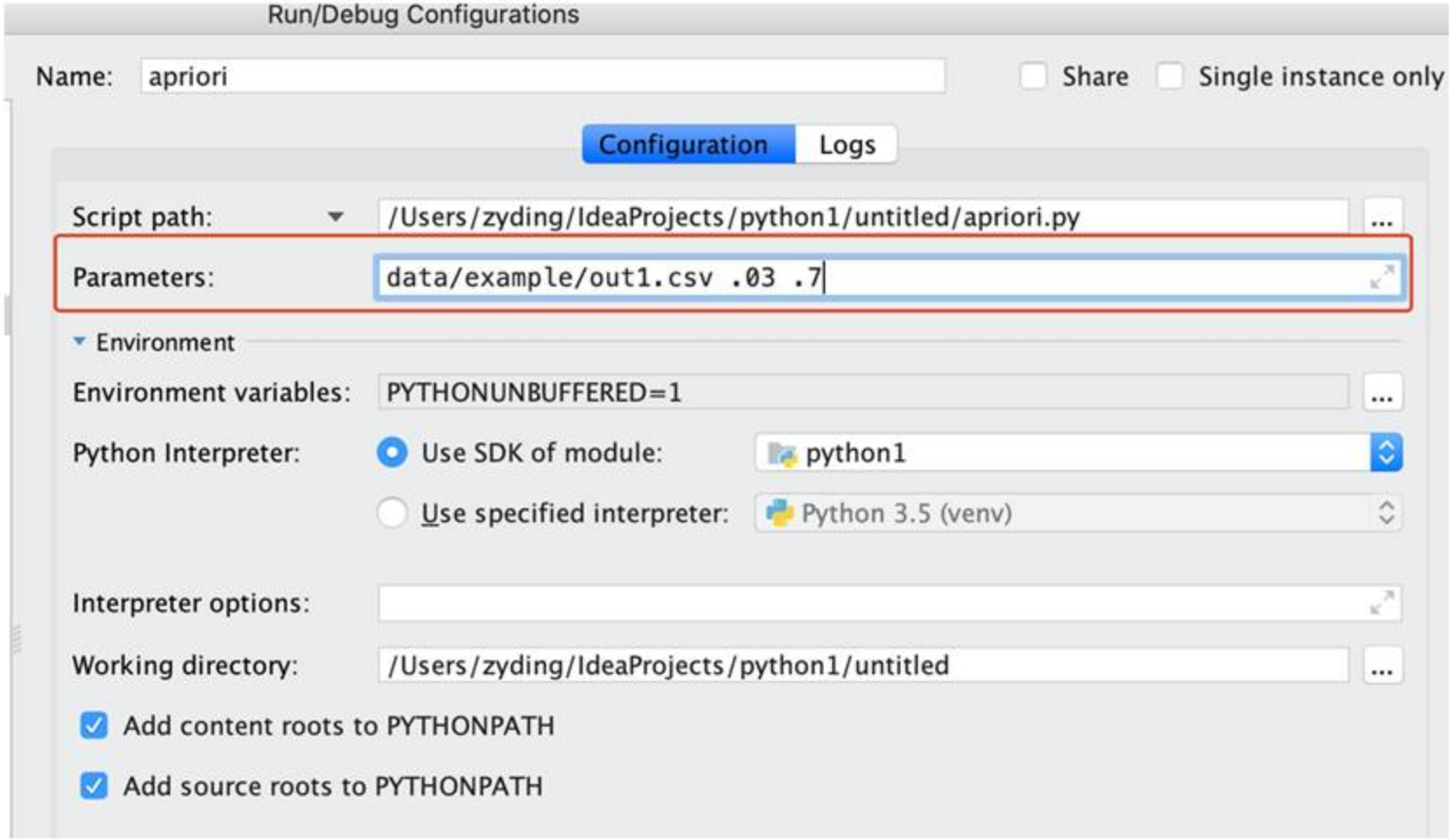
```
/Users/zyding/venv/bin/python /Users/zyding/IdeaProjects/python1/untitled/apriori.py
[['A'], ['B'], ['C'], ['E'], ['A', 'C'], ['B', 'C'], ['B', 'E'], ['C', 'E'], ['B', 'C', 'E']]
```

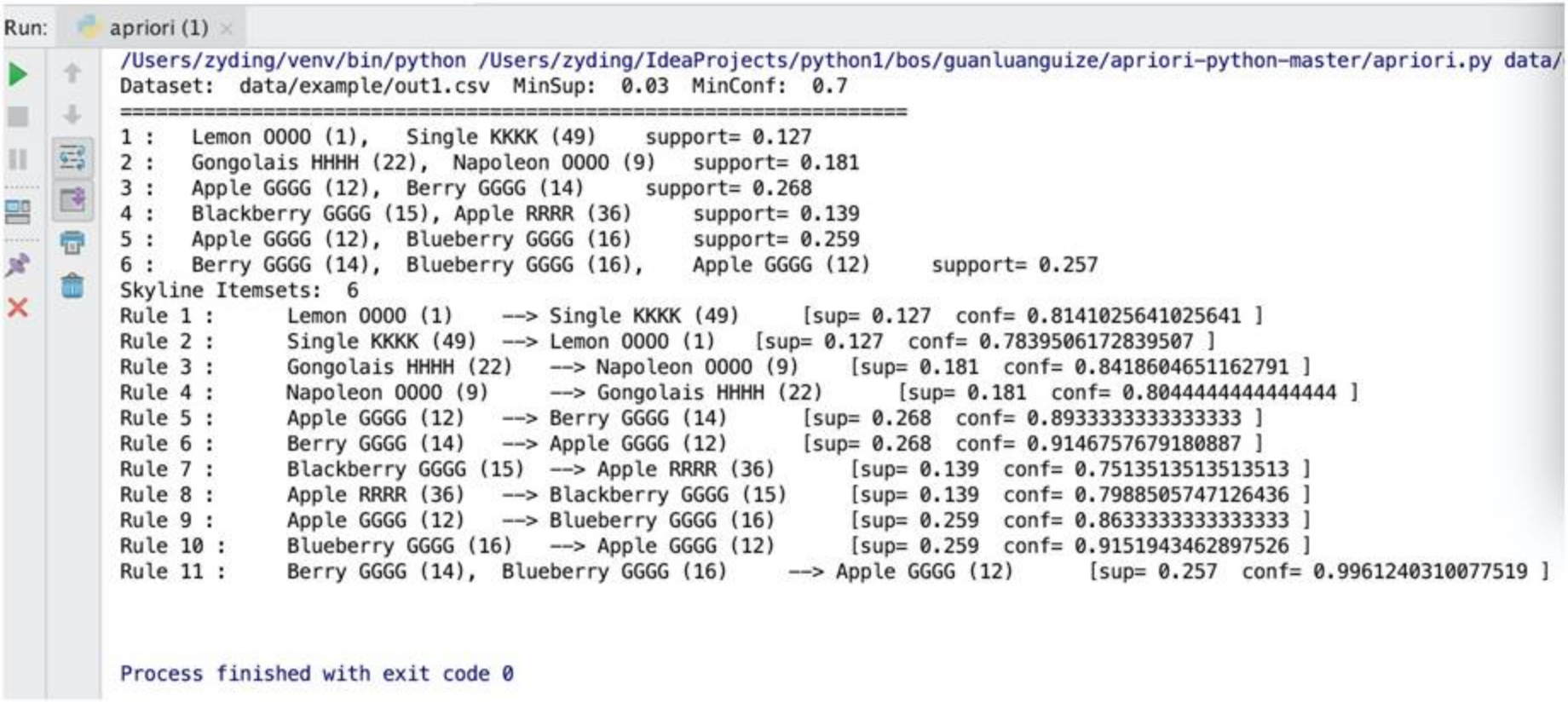

zyding1983 / datamining
Code
Issues 0
Pull requests 0
Actions
Projects 0
Branch: master
datamining / 2020A / codes / chapters4apirio /
zyding1983 add dm ppt all pdf
..
apriori-python-master
add ne1
.DS_Store
add dm
apriori.py
add ne1

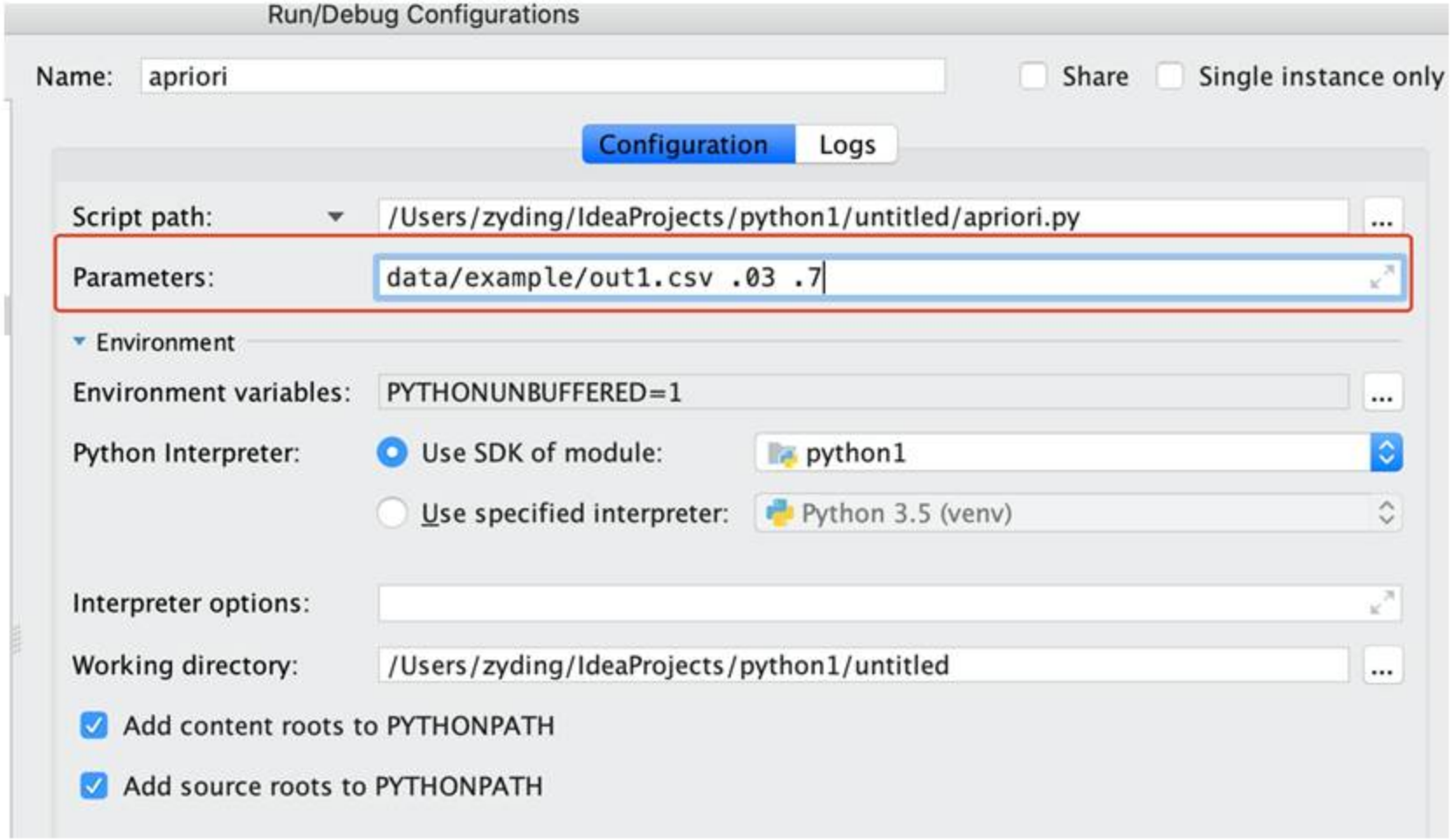

apriori-python-master
data
1000
5000
20000
75000
example
transcription
docs
apriori.py
boxF.py
goods.csv
goods-BK.csv
lab3.466.pdf
out1.csv
pcaDemo2.py
README.md
aprioridemo.py
B I
Vie
Association Rules Mining, Apriori Implimentation
Timothy Asp, Caleb Carlton
Input Format: python apriori.py [--no-rules] <dataFile-out1.csv> <minSup> <minConf>
--no-rules will run the code without rules generation.
The input datafile must be in the sparse vector format (see *-out1.csv in the differe
Example:
python apriori.py --no-rules data/1000/1000-out1.csv .03 .7
python apriori.py data/example/out1.csv .03 .7
== Extended Bakery Printouts
Dataset: data/example/out1.csv MinSup: 0.03 MinConf: 0.7
1 : Blackberry Tart (15), Apple Danish (36) support= 0.139
2 : Gongolais Cookie (22), Napoleon Cake (9) support= 0.181
3 : Lemon Cake (1), Single Espresso (49) support= 0.127
4 : Apple Tart (12), Berry Tart (14), Blueberry Tart (16) support= 0.257

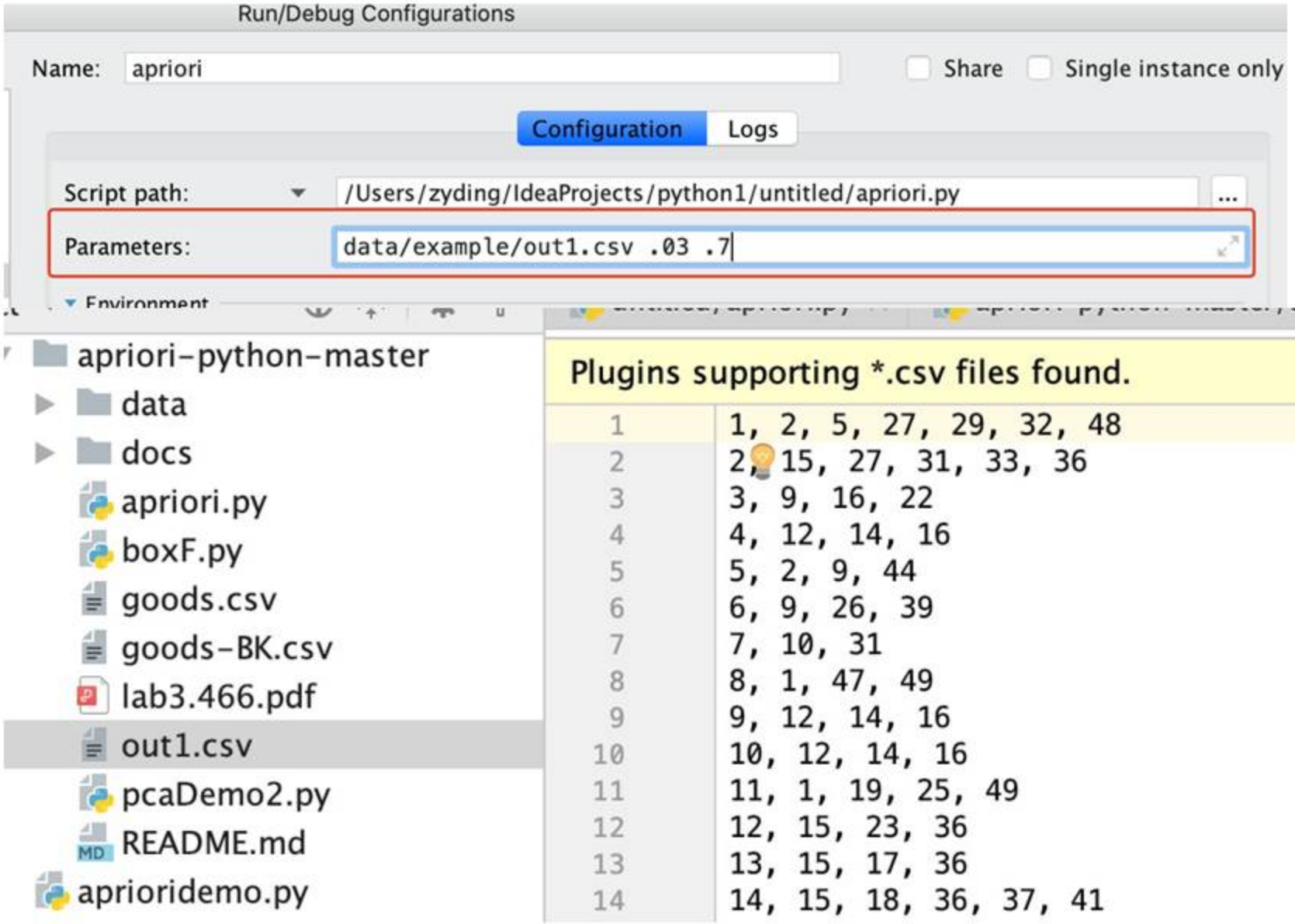

apriori-python-master
data
docs
apriori.py
boxF.py
goods.csv
goods-BK.csv
lab3.466.pdf
out1.csv
pcaDemo2.py
README.md
aprioridemo.py
yingyong111
self.dataset = data
self.transList = defaultdict(li
self.freqList = defaultdict(int
self.itemset = set()
self.highSupportList = list()
self.numItems = 0
self.prepData() # i
self.F = defaultdict(list)
self.minSup = minSup
self.minConf = minConf
def genAssociations(self):
candidate = {}
count = {}
apriori (1)
/Users/zyding/venv/bin/python /Users/zyding/IdeaProjects/python1/bos/guanluangu
Expected input format: python apriori.py <dataset.csv> <minSup> <minConf>
Process finished with exit code 0

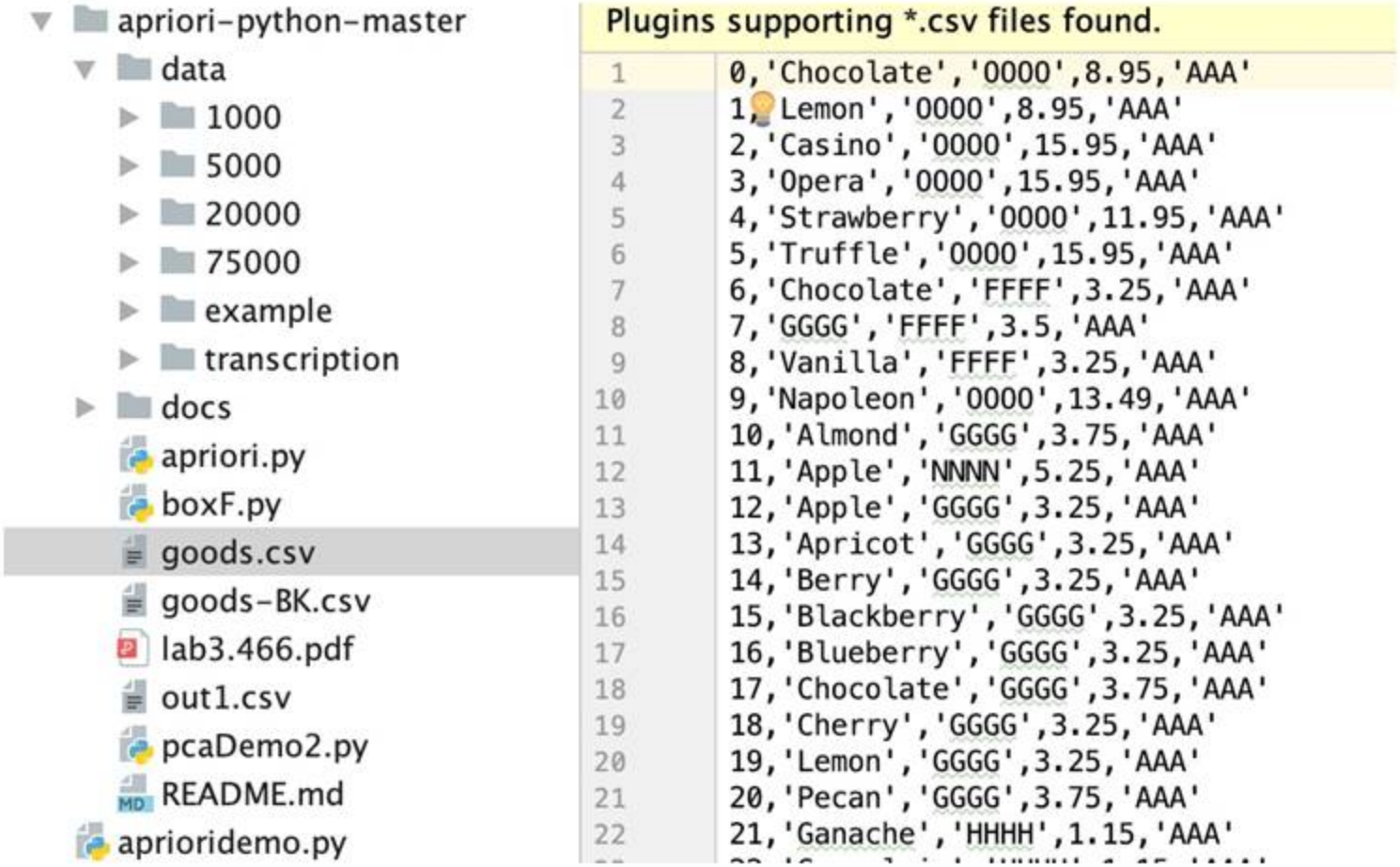

apriori
Edit Configurations...
Save 'apriori' Configuration
pcaDemo
tree1111
tree2222
bp
apriori
README.md
README.md
Association Rules Mining
Timothy Asp, Caleb Carlt
Input Format: python apr
--no-rules will run t
The input datafile mu
Example:
apriori-python-master
data

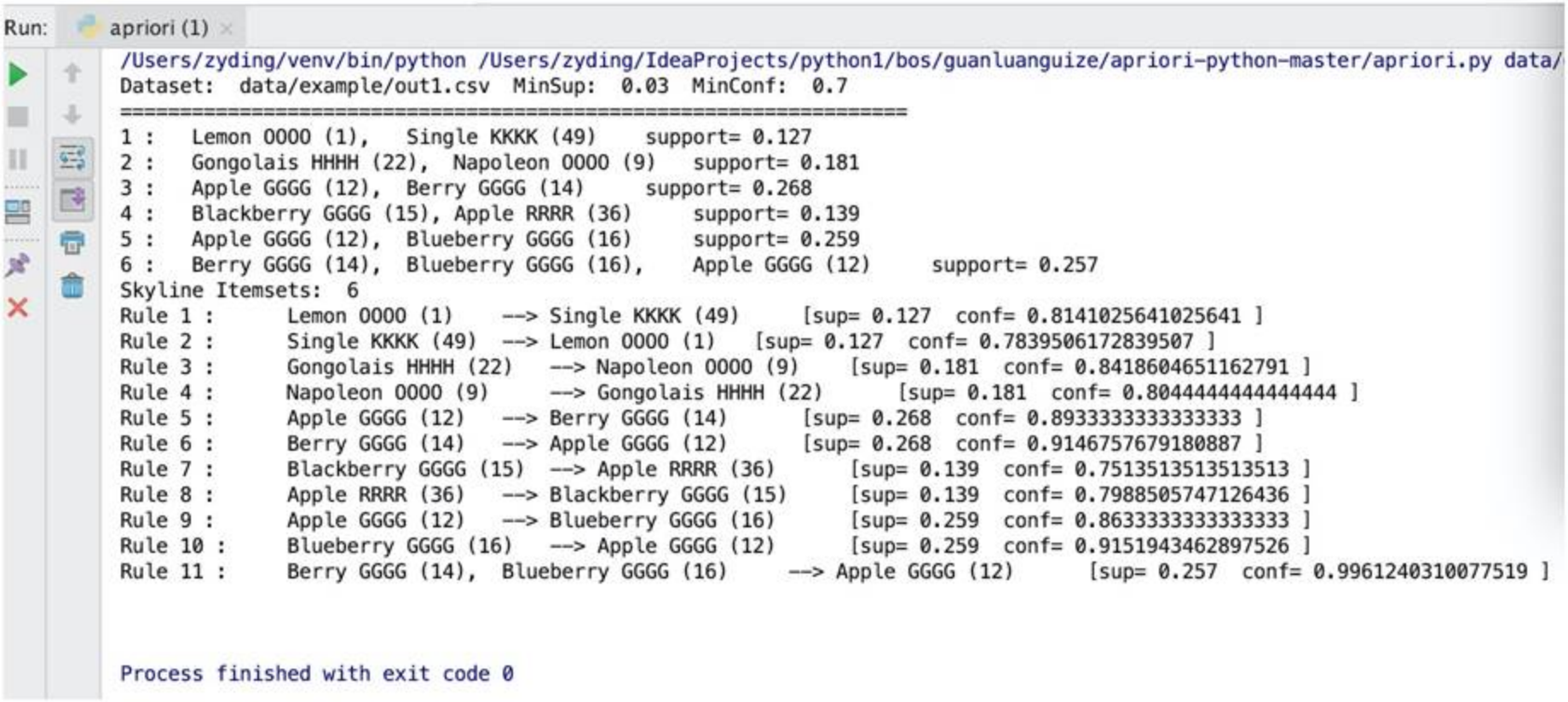

Run/Debug Configurations
Name: apriori
Share
Single instance only
Configuration
Logs
Script path: /Users/zyding/IdeaProjects/python1/untitled/apriori.py
Parameters: data/example/out1.csv .03 .7
Environment
Environment variables: PYTHONUNBUFFERED=1
Python Interpreter: Use SDK of module: python1
Use specified interpreter: Python 3.5 (venv)
Interpreter options:
Working directory: /Users/zyding/IdeaProjects/python1/untitled
Add content roots to PYTHONPATH
Add source roots to PYTHONPATH


Run:
apriori (1)
/Users/zyding/venv/bin/python /Users/zyding/IdeaProjects/python1/bos/guanluanguize/apriori-python-master/apriori.py data/
Dataset: data/example/out1.csv MinSup: 0.03 MinConf: 0.7
1 : Lemon OOOO (1), Single KKKK (49) support= 0.127
2 : Gongolais HHHH (22), Napoleon OOOO (9) support= 0.181
3 : Apple GGGG (12), Berry GGGG (14) support= 0.268
4 : Blackberry GGGG (15), Apple RRRR (36) support= 0.139
5 : Apple GGGG (12), Blueberry GGGG (16) support= 0.259
6 : Berry GGGG (14), Blueberry GGGG (16), Apple GGGG (12) support= 0.257
Skyline Itemsets: 6
Rule 1 : Lemon OOOO (1) --> Single KKKK (49) [sup= 0.127 conf= 0.8141025641025641]
Rule 2 : Single KKKK (49) --> Lemon OOOO (1) [sup= 0.127 conf= 0.7839506172839507]
Rule 3 : Gongolais HHHH (22) --> Napoleon OOOO (9) [sup= 0.181 conf= 0.8418604651162791]
Rule 4 : Napoleon OOOO (9) --> Gongolais HHHH (22) [sup= 0.181 conf= 0.8044444444444444]
Rule 5 : Apple GGGG (12) --> Berry GGGG (14) [sup= 0.268 conf= 0.8933333333333333]
Rule 6 : Berry GGGG (14) --> Apple GGGG (12) [sup= 0.268 conf= 0.9146757679180887]
Rule 7 : Blackberry GGGG (15) --> Apple RRRR (36) [sup= 0.139 conf= 0.7513513513513513]
Rule 8 : Apple RRRR (36) --> Blackberry GGGG (15) [sup= 0.139 conf= 0.7988505747126436]
Rule 9 : Apple GGGG (12) --> Blueberry GGGG (16) [sup= 0.259 conf= 0.8633333333333333]
Rule 10 : Blueberry GGGG (16) --> Apple GGGG (12) [sup= 0.259 conf= 0.9151943462897526]
Rule 11 : Berry GGGG (14), Blueberry GGGG (16) --> Apple GGGG (12) [sup= 0.257 conf= 0.9961240310077519]
Process finished with exit code 0


Run/Debug Configurations
Name: apriori
Share
Single instance only
Configuration
Logs
Script path: /Users/zyding/IdeaProjects/python1/untitled/apriori.py
Parameters: data/example/out1.csv .03 .7
Environment
Environment variables: PYTHONUNBUFFERED=1
Python Interpreter: Use SDK of module: python1
Use specified interpreter: Python 3.5 (venv)
Interpreter options:
Working directory: /Users/zyding/IdeaProjects/python1/untitled
Add content roots to PYTHONPATH
Add source roots to PYTHONPATH


Run/Debug Configurations
Name: apriori
Share
Single instance only
Configuration
Logs
Script path: /Users/zyding/IdeaProjects/python1/untitled/apriori.py
Parameters: data/example/out1.csv .03 .7
Environment
apriori-python-master
data
docs
apriori.py
boxF.py
goods.csv
goods-BK.csv
lab3.466.pdf
out1.csv
pcaDemo2.py
README.md
aprioridemo.py
Plugins supporting *.csv files found.
1 1, 2, 5, 27, 29, 32, 48
2 2, 15, 27, 31, 33, 36
3 3, 9, 16, 22
4 4, 12, 14, 16
5 5, 2, 9, 44
6 6, 9, 26, 39
7 7, 10, 31
8 8, 1, 47, 49
9 9, 12, 14, 16
10 10, 12, 14, 16
11 11, 1, 19, 25, 49
12 12, 15, 23, 36
13 13, 15, 17, 36
14 14, 15, 18, 36, 37, 41


apriori-python-master
data
1000
5000
20000
75000
example
transcription
docs
apriori.py
boxF.py
goods.csv
goods-BK.csv
lab3.466.pdf
out1.csv
pcaDemo2.py
README.md
aprioridemo.py
Plugins supporting *.csv files found.
0,'Chocolate','0000',8.95,'AAA'
1,'Lemon','0000',8.95,'AAA'
2,'Casino','0000',15.95,'AAA'
3,'Opera','0000',15.95,'AAA'
4,'Strawberry','0000',11.95,'AAA'
5,'Truffle','0000',15.95,'AAA'
6,'Chocolate','FFFF',3.25,'AAA'
7,'GGGG','FFFF',3.5,'AAA'
8,'Vanilla','FFFF',3.25,'AAA'
9,'Napoleon','0000',13.49,'AAA'
10,'Almond','GGGG',3.75,'AAA'
11,'Apple','NNNN',5.25,'AAA'
12,'Apple','GGGG',3.25,'AAA'
13,'Apricot','GGGG',3.25,'AAA'
14,'Berry','GGGG',3.25,'AAA'
15,'Blackberry','GGGG',3.25,'AAA'
16,'Blueberry','GGGG',3.25,'AAA'
17,'Chocolate','GGGG',3.75,'AAA'
18,'Cherry','GGGG',3.25,'AAA'
19,'Lemon','GGGG',3.25,'AAA'
20,'Pecan','GGGG',3.75,'AAA'
21,'Ganache','HHHH',1.15,'AAA'


Run:
apriori (1)
/Users/zyding/venv/bin/python /Users/zyding/IdeaProjects/python1/bos/guanluanguize/apriori-python-master/apriori.py data/
Dataset: data/example/out1.csv MinSup: 0.03 MinConf: 0.7
1 : Lemon OOOO (1), Single KKKK (49) support= 0.127
2 : Gongolais HHHH (22), Napoleon OOOO (9) support= 0.181
3 : Apple GGGG (12), Berry GGGG (14) support= 0.268
4 : Blackberry GGGG (15), Apple RRRR (36) support= 0.139
5 : Apple GGGG (12), Blueberry GGGG (16) support= 0.259
6 : Berry GGGG (14), Blueberry GGGG (16), Apple GGGG (12) support= 0.257
Skyline Itemsets: 6
Rule 1 : Lemon OOOO (1) --> Single KKKK (49) [sup= 0.127 conf= 0.8141025641025641]
Rule 2 : Single KKKK (49) --> Lemon OOOO (1) [sup= 0.127 conf= 0.7839506172839507]
Rule 3 : Gongolais HHHH (22) --> Napoleon OOOO (9) [sup= 0.181 conf= 0.8418604651162791]
Rule 4 : Napoleon OOOO (9) --> Gongolais HHHH (22) [sup= 0.181 conf= 0.8044444444444444]
Rule 5 : Apple GGGG (12) --> Berry GGGG (14) [sup= 0.268 conf= 0.8933333333333333]
Rule 6 : Berry GGGG (14) --> Apple GGGG (12) [sup= 0.268 conf= 0.9146757679180887]
Rule 7 : Blackberry GGGG (15) --> Apple RRRR (36) [sup= 0.139 conf= 0.7513513513513513]
Rule 8 : Apple RRRR (36) --> Blackberry GGGG (15) [sup= 0.139 conf= 0.7988505747126436]
Rule 9 : Apple GGGG (12) --> Blueberry GGGG (16) [sup= 0.259 conf= 0.8633333333333333]
Rule 10 : Blueberry GGGG (16) --> Apple GGGG (12) [sup= 0.259 conf= 0.9151943462897526]
Rule 11 : Berry GGGG (14), Blueberry GGGG (16) --> Apple GGGG (12) [sup= 0.257 conf= 0.9961240310077519]
Process finished with exit code 0