Link to the user is backet

support is count of buckets

数值: support confidence interest

below confidence → 指的是 threshold.

$m \to b$ $\frac{m \cup b}{m.}$ $\frac{cm\,b}{cm.}$

$cm \to b$

1. 购物篮模型：描述一种多对多关系，其中一类对象是item，另一类是basket，也叫transaction（交易）

item：super market 所有 goods 的集合，很大。

basket：包含一部分goods，是item的子集，一般数量比较多。

2. 几个概念：

① 支持度 support：I是一个序列，包含I的basket的数目为 support。（通常用分数表示，即 包含I的basket数目 / 总数目。）知识中所有商品的当前的事务比例

② 频繁项集：I的支持度 ≥ support threshold。

③ association rules：$\{i_1, i_2, \cdots, i_k\} \to j$ means：如果包含 $i_1, i_2, \cdots, i_k$，则很可能包含 j。

④ 置信度 conf(I→j)：association rules的可信度

$$conf(I \to j) = \frac{support(I \cup j)}{support(I)}$$

前提成立时，结论也成立的可能性

⑤ 兴趣度 (interest)：association rules 防止热门产品影响表达式的引入。

interest = $conf(I \to j) - Pr[j]$ → 包含j的basket比例。

{a, b, c, d} 是频繁的，那么 {a, b, c} 则一定也是频繁的。

if A, B, C → D 的 conf 好于 X，则 A, B → C, D 一定也好。

⑥ maximal frequent itemsets：{a, b, c} 是frequent，但任意加一个 {a, b, c, x} 都不frequent。

⑦ closed itemsets：{a, b, c} 和任意X的 {a, b, c, x} 有不同的 count。

用 the number of passes an algorithm makes（读取数据的遍数）计算 cost。

⑧ Lift：提升度：$Lift(A \to B) = \frac{support(A \cup B)}{support(A) \times support(B)}$。

>1时：A与B存在正向关联性，买A同时更可能买B。越大表示关联程度越强。

=1时 无关联。

3. find 频繁项：

<1时：负相关，买A不太可能买B，越小表示关联越弱

counting pairs in memory:

① approach 1：用矩阵 若数字用4bytes则 → 一个pair用4byte

② 用三元表：[i, j, c] → 一个pair用12byte

for approch 1：按{1,2}、{1,3}…，{1,n}，{2,3}，{2,4}…{2,n}存储 列优先

i<j，存上三角。{i,j}对应于(i-1)(n-i/2)+j-1的位置。

数据总数 $\frac{n(n-1)}{2}$，总字节数 $2n^2$

当 less than $\frac{1}{3}$ of possible pairs 时 2更好。

上述方法为memory足够多时，但若memory不够呢？ Frequent Pairs.

A-Priori 算法：key-idea: monotonicity - 单调性。

如果集合I出现不少于s次，则I的子集也是。

如果集合I 并未出现在s个basket中，则没有包含I的集合出现在s个basket中。

step1：Read baskets，计算每个item出现的次数，把大于s的item称为频繁item。

这步只需正比于item数目的内存。→ 当然可以根据需要变动。

step2：再读篮子，在主存中只计算2个元素都是频繁item的basket。

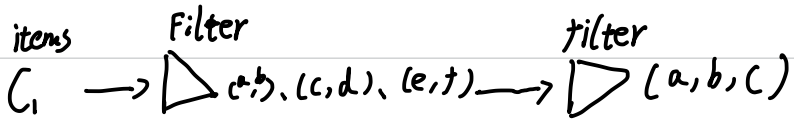
需要与频繁项的平方成正比的内存（计数）

频繁的pairs列表。

→ 用三角矩阵存储的话，n= frequent item 数目。

一般k=2时内存要求最多。即第2步。

目前得到 Pairs (,) count，作为k-1向k传递，每一次都计算该层所有集合的count，删去不需要的再传入下一层.



更大数据集时用 PCY 算法 (Park-Chen-Yu):

来源于 A-Priori 中第一步有大部分内存闲置，用第一步闲置的内存减少第2步所需.

Pass 1 时 PCY: 除项目计数外，维护一个hash表，要有尽可能多bucket → hash冲突时保存内容.
为每一个散列桶保存一个计数，桶只保留计数而不保留实际项目对.

在第一步，不仅要得到每个item的count，还要对每个basket中产生所有项对pairs，
创一张hash表. 桶计数小于s的，它的所有项对都不是频繁的.
含频繁pair的桶一定够格. 但频繁的桶可能内容全是不频繁的.
↓
因为多个不同内容hash到一起.

Pass 2: 只计算hash频繁桶里面的.
用bit图代替hash table. 1-频繁桶. 0-无.