1. **Apriori**

Apriori算法过程如下：

* 扫描一次数据库，计算出各个1项集的支持度，得 到频繁1项集的集合。
* 连接步：将(k - 1)项集broadcast出去，通过self-join计算出候选k项集。
* 剪枝步：候选k项集的任意子集必须是频繁项集。
* 扫描一次数据库，计算候选k项集的支持度，过滤出频繁k项集，并且collect到driver。

通过迭代循环，重复步骤2-4，直到没有候选项集。

1. **FPGrowth**

FPGrowth算法过程如下：

* 扫描一次数据库，找出频繁1项集F-List，按降序排序。
* 将F-List中的项（item）分为Q个组（group），每一个组都有一个唯一的group-id，我们将所有项以及其所对应的group-id记为G-List。
* 对每个transaction，过滤不频繁集合，剩下的频繁项集按F-List顺序排序。从右向左扫描每个transaction，根据G-List生成conditional transaction。
* 将group-id相同的conditional transaction传输到同一个节点上。将这些conditional transaction插入到FP-tree，并调用单机FP-Growth挖掘频繁项集。

1. **FIN**

FIN算法过程如下：

* 扫描一次数据库，找出频繁1项集F-List，按降序排序。
* 将F-List中的项（item）分为Q个组（group），每一个组都有一个唯一的group-id，我们将所有项以及其所对应的group-id记为G-List。
* 对每个transaction，过滤不频繁集合，剩下的频繁项集按F-List顺序排序。从右向左扫描每个transaction，根据G-List生成conditional transaction。
* 将group-id相同的conditional transaction传输到同一个节点上。将这些conditional transaction插入到PPC-Tree。
* 先序遍历PPC-Tree，生成当前group中的所有2-Nodeset。
* 根据Nodeset，建立set-enumeration tree，同时挖掘所有频繁项集。

1. **PrePost**

PrePost算法过程如下：

* 扫描一次数据库，找出频繁1项集F-List，按降序排序。
* 将F-List中的项（item）分为Q个组（group），每一个组都有一个唯一的group-id，我们将所有项以及其所对应的group-id记为G-List。
* 对每个transaction，过滤不频繁集合，剩下的频繁项集按F-List顺序排序。从右向左扫描每个transaction，根据G-List生成conditional transaction。
* 将group-id相同的conditional transaction传输到同一个节点上。将这些conditional transaction插入到PPC-Tree。
* 先序和后序遍历PPC-Tree，生成当前group中的所有NList。
* 根据NList，建立set-enumeration tree，同时挖掘所有频繁项集。