**电子科技大学计算机科学与工程学院**

**标 准 实 验 报 告**

**（实验）课程名称 数据结构与算法**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：陶浩轩 学 号：2023080902011 指导教师：陈端兵**

1. **实验室名称：**

学知三组团 110139

**二、实验项目名称：**

频繁模式与关联规则挖掘

1. **实验原理：**

FP树通过将数据集中的频繁项集压缩成树形结构，仅存储满足最小支持度的项，从而能够高效地挖掘频繁项，因为它减少了冗余数据，并允许通过树结构快速计算项集的支持度。

**四、实验目的：**

 通过构建频繁模式树（FP-tree），挖掘数据集中的频繁项集和关联规则，以了解项目之间的关联性。

**五、实验内容：**

1. 构建FP-tree

2. 使用FP-Growth挖掘频繁项集

**六、实验器材（设备、元器件）：**

Windows 10，Python 3.12.6

**七、实验步骤：**

1. **读取数据集**：

调用 readcsv 函数，传入事务数据库的文件路径，打开CSV文件，逐行读取内容，将每个项目转换为整数类型，并将每个事务表示为一个元组（包含支持度计数和项目集列表），最后将所有事务存储在一个列表中并返回。

1. **初始化FP树**：

 创建 FPTree 类的一个实例，初始化根节点（其nid设为-1，支持度为0），同时初始化一个空的事务数据库和头表，为构建FP树做准备。

1. **构建FP树**：

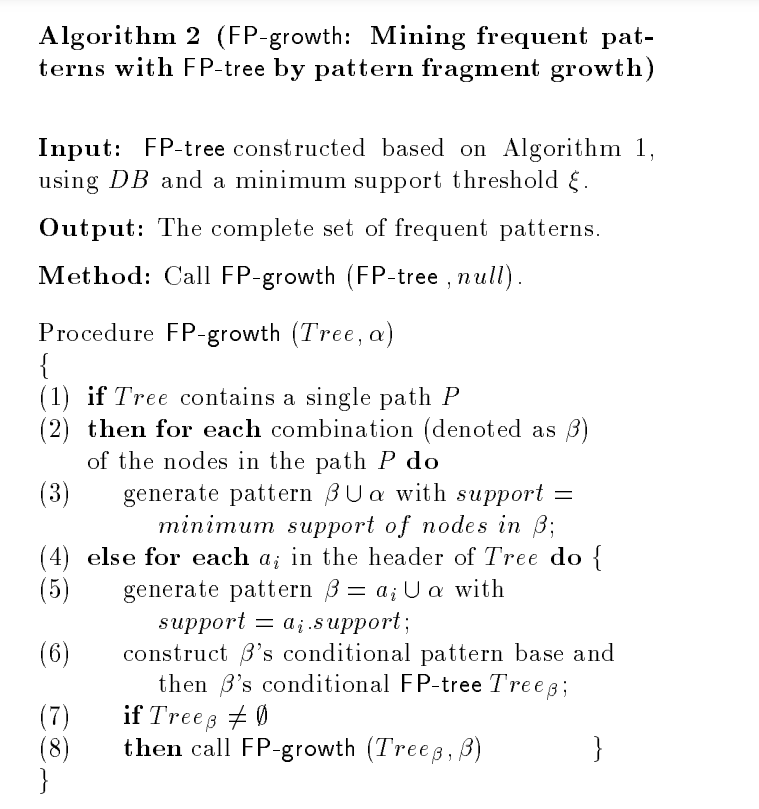
调用 build 方法，传入事务数据库文件路径或已经读取的事务列表，遍历每个事务，按支持度降序处理项目集，更新FP树和头表，对于每个项目，如果在当前节点的子节点中已存在，则增加其支持度，否则创建新节点，并将其添加到子节点和头表中。

1. **递归增长FP树以挖掘频繁项集**：

 调用 growth 方法，传入最小支持度阈值，若最小支持度为小数，则转换为事务总数的整数比例，遍历头表中的每个项目，若其支持度大于等于最小支持度，则递归构建条件模式树，并挖掘出频繁项集。

1. **递归挖掘频繁模式**：

 fp\_growth 方法用于递归挖掘频繁模式，若树为单路径，则直接提取路径中的所有组合作为频繁项集；否则，遍历头表中的每个项目，构建条件模式，并递归挖掘。



1. **获取条件模式**：

 cond\_pattern 方法用于获取给定节点的条件模式，通过从当前节点向上遍历至根节点，记录路径上的节点ID，形成条件模式。

1. **打印FP树和头表**：

 print 方法用于打印FP树的结构，展示树中每个节点的ID和支持度；print\_header\_table 方法用于打印头表，显示每个项目的支持度和链接的节点。

1. **挖掘并打印最大频繁项集**：

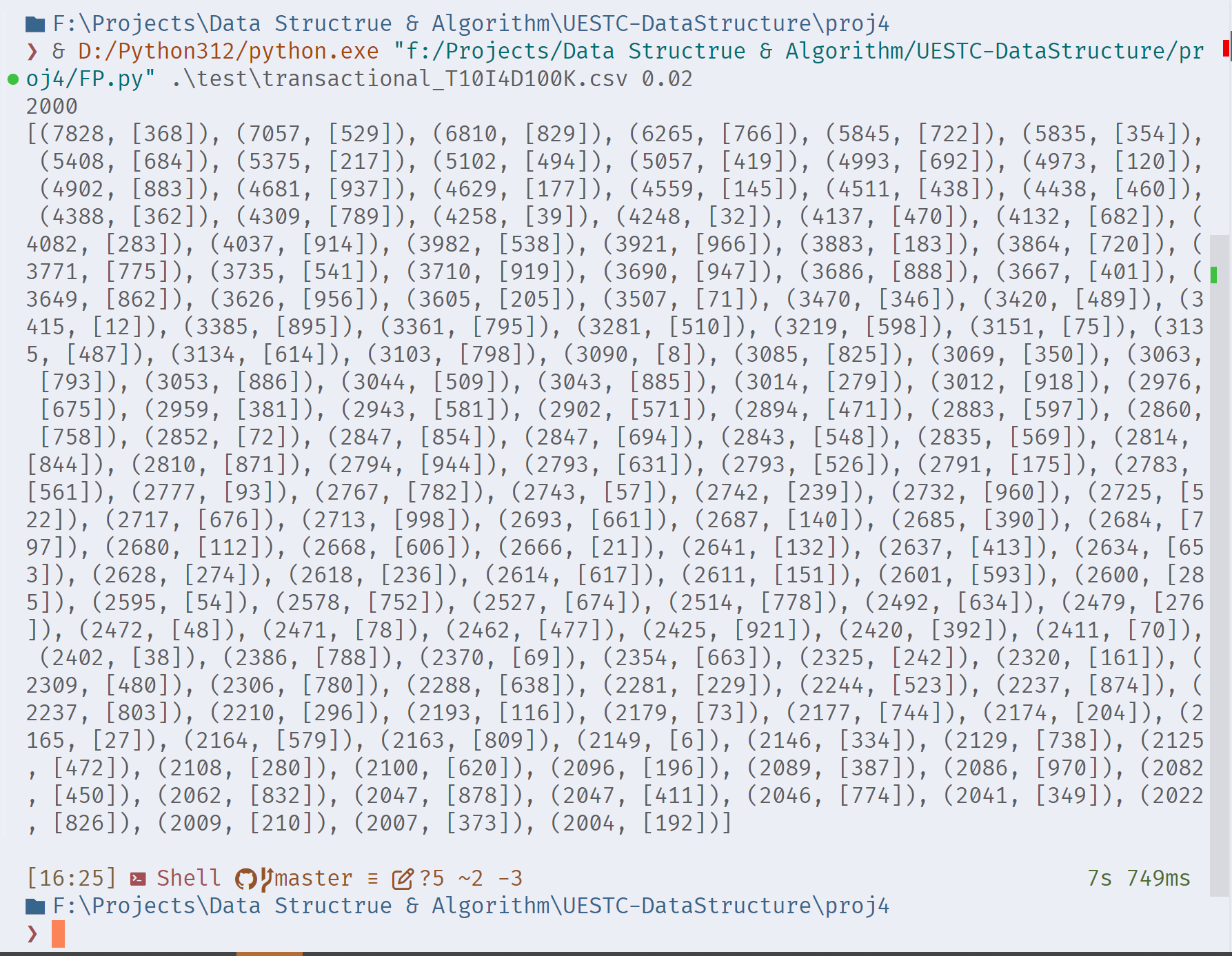
调用 mine 函数，传入事务数据库文件路径和最小支持度，构建FP树，挖掘频繁项集，并按支持度降序排序结果；print\_maximal 函数用于打印前n个最大频繁项集。

1. **主程序执行**：

 在 if \_\_name\_\_ == '\_\_main\_\_': 块中，解析命令行参数以获取文件路径和最小支持度，调用 mine 函数执行频繁模式挖掘，并打印挖掘出的频繁项集。

**八、实验数据及结果分析：**

使用数据集：transactional\_T10I4D100K.csv，共有100000笔交易



可以看到，相较于实验一提供的 Apriori 算法，FP-Tree 可以在更短的时间之内完成更加巨量的数据的挖掘。

**九、实验结论：**

通过本次实验，我们验证了FP-Tree算法在频繁模式与关联规则挖掘中的有效性和高效性。实验结果表明，FP-Tree算法不仅能够快速挖掘出频繁项集，而且能够有效地生成关联规则，为数据分析和决策提供了有力的支持

**十、总结及心得体会：**

在本次实验中，我深入理解了FP-Tree算法的原理和实现过程，并通过实践操作掌握了如何使用该算法进行频繁模式挖掘。实验过程中，我体会到了数据结构算法在优化方面的重要性。高效的数据结构是提升处理效率的一大关键。

**十一、对本实验过程及方法、手段的改进建议：**

1. 可以进一步针对数据集进行数据结构方面的优化，比如针对数据的范围设定特定大小的数据类型
2. 在读取数据集和构建FP树的过程中，可以利用多线程或分布式计算来并行处理数据，提高构建速度

**报告评分：**

**指导教师签字：**