Assignment5：实现Apriori算法的频繁项集挖掘

作业要求

1. 现Apriori算法，在给定的GroceryStore购买记录数据集（存储在Groceries.csv），完成算法测试。
2. 使用Apriori算法找出支持度不低于support threshold的频繁项集（support threshold分别设为3、4、5），并比较不同的support threshold挖掘得到的结果（可以从不同的角度进行分析）。

作业过程

算法原理

Apriori算法是一种逐层搜索的迭代算法，通过不断筛选具有k个项的候选集C_k(由L_{k-1}与L_1连接后满足非空子集也是频繁项集的项的集和)和频繁集L_k（满足最小支持度的C_k）,直到不再生成新的频繁项集。将迭代过程中的频繁项集收集可产生强关联规则。

代码实现

在代码实现的过程中，首先尝试了L_{k-1}与L_1连接后去除非频繁项的方法，来生成新的候选项集，但经运行尝试结果，发现改方法的运行耗时过长，特别是当阈值设置为3时很久无法运行出结果，分析主要消耗在于每次L_{k-1}和L_1连接都会产生大量明显不满足条件的候选项集再来筛选，会多出许多比必要的计算。经查阅资料改进，将产生新候选集的方法改为直接使用L_{k-1}项集的项两两合并，具体方法为：对L_{k-1}的每个频繁项中的元素进行排序，遍历对比两个频繁项时，为了使合并后具有k个项，当频繁项满足前$k-2$项相等时，即可合并，合并后的项再检验其所有非空子集是否存在于L_{k-1}中，存在则加入候选集。

主函数

```
if __name__ == "__main__":
    threshold = 3
    # 读取Groceries.json文件
    with open('Groceries.json', 'r') as file:
        data = json.load(file)
    # 创建0-1矩阵
    all_items = set()
    for items_str in data.values():
        items = items_str.split(',')
        all_items.update(items)
    all_items = sorted(list(all_items))
    transactions = []
    for items_str in data.values():
        items = items_str.split(',')
        transaction_vector = [1 if item in items else 0 for item in all_items]
        transactions.append(transaction_vector)
    df = pd.DataFrame(transactions, columns=all_items)  # 使用panda存储便于操作
    transactions = [items.split(',') for items in data.values()]
    start_time = time.time()
    # 初始化首轮候选项集
    support_series = df.sum(axis=0)
    candidates = list(support_series[support_series > threshold].index)
    k = 0
```

```
    while candidates != []:
        k = k + 1
        write_candidates(candidates,threshold,k)
        # 连接产生新的候选项集
        candidates =generate_next_candidates(candidates)
        # 根据阈值过滤产生频繁项集
        candidates = count_prune(candidates,transactions,threshold)
        print(f'set {k} --nums of candidates: {len(candidates)}')
    end_time = time.time()
    time = end_time - start_time
    print(f"threshold {threshold} time taken: {time:.4f} s")
```

产生$k+1$项候选项集

```
def generate_next_candidates(candidates, ms):
    candidates = list(map(lambda i: sorted(i.split(ms)), candidates))
    k = len(candidates[0])
    C = []
    for i in range(len(candidates)):
        for j in range(i, len(candidates)):
            if candidates[i][:k - 1] == candidates[j][:k - 1] and candidates[i]
[k - 1] != candidates[j][k - 1]:
                tmp = candidates[i][:k - 1] + sorted([candidates[j][k - 1],
candidates[i][k - 1]])
                # 判断所有k项的子集是否在频繁项集中
                subsets = []
                for q in range(len(tmp)):
                    t = [tmp[q]]
                    tt = list(set(tmp) - set(t))
                    subsets.append(tt)
                is_fre = True
                for w in subsets:
                    if w not in candidates:
                        is_fre = False
                        break
                if is_fre:
                    C.append(tmp)
    return C
```

筛选出支持度大于`threshold`的频繁项集

```
def count_prune(candidates,df,threshold):
    if candidates == []:return []
    nums = [0] * len(candidates)
    for transaction in transactions:
        for i,candidate in enumerate(candidates):
            is_support = True
            for item in candidate:
                if item not in transaction:
                    is_support = False
            if is_support:
                nums[i]+=1
    positions = [i for i, num in enumerate(nums) if num > threshold]
    can = [candidates[i] for i in positions]
```

```
    candidates = []
    for i in can:
        candidates.append(','.join(i))
    return candidates
```

二、 运行测试

设置support threshold阈值分别为3、4和5，运行比较

阈值3：

```
set 1 --nums of candidates: 5425
set 2 --nums of candidates: 25985
set 3 --nums of candidates: 31620
set 4 --nums of candidates: 13940
set 5 --nums of candidates: 2896
set 6 --nums of candidates: 335
set 7 --nums of candidates: 20
set 8 --nums of candidates: 1
set 9 --nums of candidates: 0
threshold 3 time taken: 2376.8372 s
```

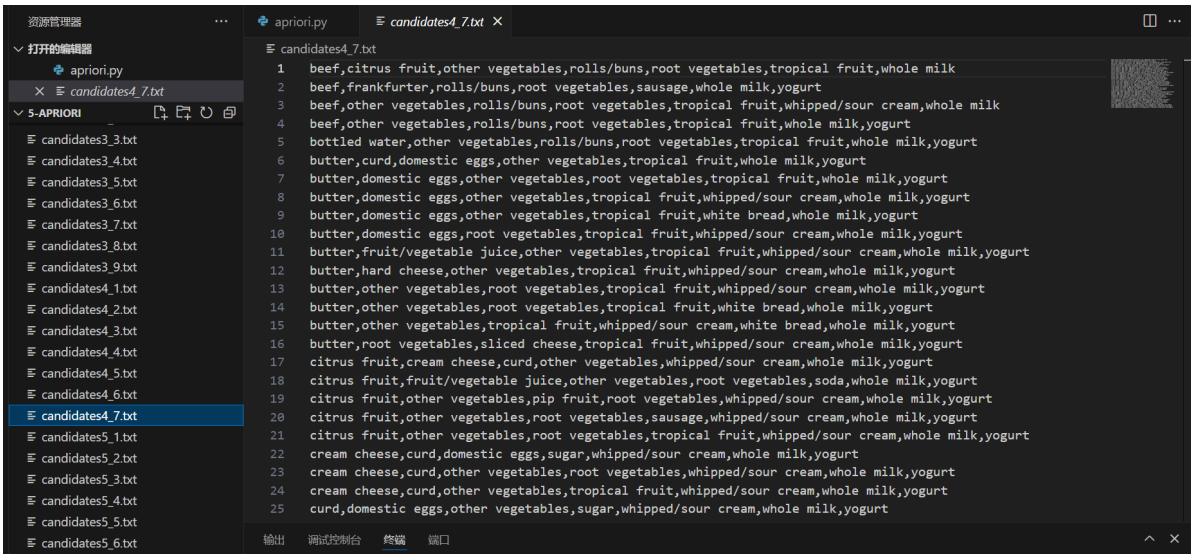
阈值4：

```
set 1 --nums of candidates: 4782
set 2 --nums of candidates: 19226
set 3 --nums of candidates: 18597
set 4 --nums of candidates: 5988
set 5 --nums of candidates: 785
set 6 --nums of candidates: 38
set 7 --nums of candidates: 0
threshold 4 time taken: 1272.6152 s
```

阈值5：

```
set 1 --nums of candidates: 4282
set 2 --nums of candidates: 14805
set 3 --nums of candidates: 11864
set 4 --nums of candidates: 2952
set 5 --nums of candidates: 281
set 6 --nums of candidates: 5
set 7 --nums of candidates: 0
threshold 5 time taken: 792.3931 s
```

收集过程的频繁项集结果：



总结

随着支持度阈值的增大，每次生成的频繁项集数量都有所减少，降低了运行时间，且迭代的次数也逐步降低，越低的支持度阈值能够达到k的值越大，过程中候选项集的产生数量都会经历一段从上升到下降为0的过程，处理上升过程面临的内存问题是主要难点。