МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
**"Южно-Уральский государственный университет
(национальный исследовательский университет)"**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ОТЧЕТ

о выполнении практической работы № 1

по дисциплине

«Технологии аналитической обработки информации»

Выполнил:
студент группы КЭ-403
Гольденберг Д.И.

Проверил:
Преподаватель кафедры СП
Гоглачев А.И.

Челябинск, 2025 г.

ОГЛАВЛЕНИЕ

1. ЗАДАНИЕ

1. Разработайте программу, которая выполняет поиск частых наборов объектов в заданном наборе данных с помощью алгоритма Apriori (или одной из его модификаций). Список результирующих наборов должен содержать как наборы, так и значение поддержки для каждого набора. Параметрами программы являются набор, порог поддержки и способ упорядочивания результирующего списка наборов (по убыванию значения поддержки или лексикографическое).

2. Проведите эксперименты на наборе данных baskets.csv (сведения о покупках в супермаркете). В экспериментах варьируйте пороговое значение поддержки (например: 1%, 3%, 5%, 10%, 15%).

3. Выполните визуализацию результатов экспериментов в виде следующих диаграмм:

- сравнение быстродействия на фиксированном наборе данных при изменяемом пороге поддержки;
- количество частых наборов объектов различной длины на фиксированном наборе данных при изменяемом пороге поддержки.

4. Подготовьте отчет о выполнении задания и загрузите отчет в формате PDF в систему. Отчет должен представлять собой связный и структурированный документ со следующими разделами:

- формулировка задания;
- гиперссылка на каталог репозитория с исходными текстами, наборами данных и др. сопутствующими материалами;
- рисунки с результатами визуализации;
- пояснения, раскрывающие смысл полученных результатов.

2. РЕАЛИЗАЦИЯ АЛГОРИТМА

Реализация поиска частых наборов представлена в функции `run_apriori`, которая выполняет очистку данных от `NaN`-значений и их преобразование в список транзакций, а также кодирование данных в формате one-hot encoding, где каждая строка представляет покупку, а столбцы — наличие товаров. Также запускается алгоритм Apriori с заданным порогом минимальной поддержки и выполняется сортировку результатов в зависимости от переданного параметра (`support` или `lexical`). Код метода `run_apriori` представлен в листинге 1.

Листинг 1 – Код метода «run_apriori»

```
def run_apriori(data, min_support, sort_by='support'):
    transactions = data.apply(lambda row: row.dropna().tolist(),
axis=1).tolist()
    unique_items = sorted(set(item for transaction in transactions for item
in transaction))
    encoded_data = pd.DataFrame([[item in transaction for item in
unique_items] for transaction in transactions],
                                columns=unique_items)

    frequent_itemsets = apriori(encoded_data, min_support=min_support,
use_colnames=True)

    if sort_by == 'support':
        frequent_itemsets = frequent_itemsets.sort_values(by='support', as-
cending=False)
    elif sort_by == 'lexical':
        frequent_itemsets = frequent_itemsets.sort_values(by='itemsets')

    return frequent_itemsets
```

Код реализованной программы и всех проведенных экспериментов находится в репозитории по ссылке https://github.com/Goldria/analitycs/blob/main/1_frequent_sets_search/frequent_sets_search.ipynb.

Алгоритм был протестирован на наборе данный, представленном в презентации. Результаты работы совпали с примером из презентации. Код применения реализованного алгоритма представлен в листинге 2, результаты применения реализованного алгоритма представлены на рисунке 1.

Листинг 2 – Код применения реализованного алгоритма

```
data = [
    ['I1', 'I2', 'I5'],
    ['I2', 'I4'],
```

```
    ['I2', 'I3'],
    ['I1', 'I2', 'I4'],
    ['I1', 'I3'],
    ['I2', 'I3'],
    ['I1', 'I3'],
    ['I1', 'I2', 'I3', 'I5'],
    ['I1', 'I2', 'I3'],
    ['I6']
]
```

support	itemsets
0.4	(I2, I1)
0.4	(I1, I3)
0.4	(I2, I3)
0.2	(I5, I1)
0.2	(I4, I2)
0.2	(I5, I2)
0.2	(I2, I1, I3)
0.2	(I2, I5, I1)
0.1	(I4, I1)
0.1	(I5, I3)
0.1	(I2, I1, I4)
0.1	(I5, I1, I3)
0.1	(I5, I2, I3)
0.1	(I2, I5, I1, I3)

Рисунок 1 – Результаты применения реализованного алгоритма

3. ЭКСПЕРИМЕНТЫ

3.1. Загрузка данных и тестовый запуск алгоритма

Данные были загружены с использованием библиотеки «pandas», далее данные были преобразованы в список списков непустых значений строк (список корзин). Тестовый запуск алгоритма производился с порогом в 0.01. Код загрузки данных и тестового запуска алгоритма представлен в листинге 3. Результат запуска алгоритма представлен на рисунке 2.

Листинг 3 – Код загрузки данных и тестового запуска алгоритма

```
filename = 'baskets.csv'
with open(filename, 'rb') as f:
    result = chardet.detect(f.read())

df = pd.read_csv(filename, encoding=result['encoding'])
min_support = 0.01
itemsets = run_apriori(df, min_support, sort_by='support')
```

support	itemsets
0.017200	(макароны, минеральная вода, говяжий фарш)
0.016400	(минеральная вода, шоколад, макароны)
0.016133	(макароны, минеральная вода, молоко)
0.015200	(яйца, минеральная вода, макароны)
0.014000	(минеральная вода, шоколад, молоко)
0.013467	(яйца, минеральная вода, шоколад)
0.013067	(яйца, минеральная вода, молоко)
0.012267	(макароны, минеральная вода, замороженные овощи)
0.011733	(макароны, минеральная вода, блинчики)
0.011200	(макароны, шоколад, молоко)
0.011067	(минеральная вода, говяжий фарш, молоко)
0.011067	(минеральная вода, замороженные овощи, молоко)
0.010933	(минеральная вода, шоколад, говяжий фарш)
0.010933	(яйца, шоколад, макароны)
0.010667	(макароны, минеральная вода, оливковое масло)
0.010400	(макароны, минеральная вода, картофель-фри)
0.010133	(яйца, минеральная вода, говяжий фарш)

Рисунок 2 – Результат запуска алгоритма

3.2. Эксперименты с варьированием порогового значения

Были проведены эксперименты на наборе данных baskets.csv с различными значениями порога поддержки (1%, 3%, 5%, 10%, 15%). Для каждого значения фиксировалось время выполнения алгоритма.

На основании проведенных экспериментов была выполнена визуализация результатов с использованием линейной диаграммы, график зависимости времени выполнения от порога поддержки представлен на рисунке 3.

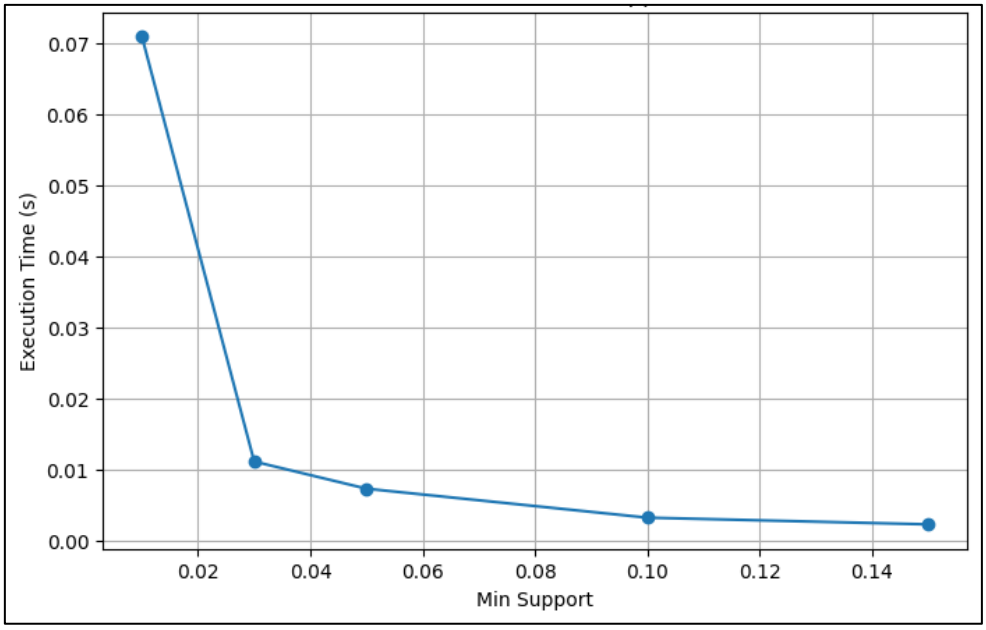

Рисунок 3 – Диаграмма результатов эксперимента

По графику можно сделать следующие выводы.

1. Время выполнения значительно уменьшается при увеличении порога поддержки.

2. С увеличением порога поддержки количество рассматриваемых комбинаций товаров сокращается, что ускоряет выполнение алгоритма.

График зависимости числа частых наборов разной длины от порога поддержки представлен на рисунке 4.

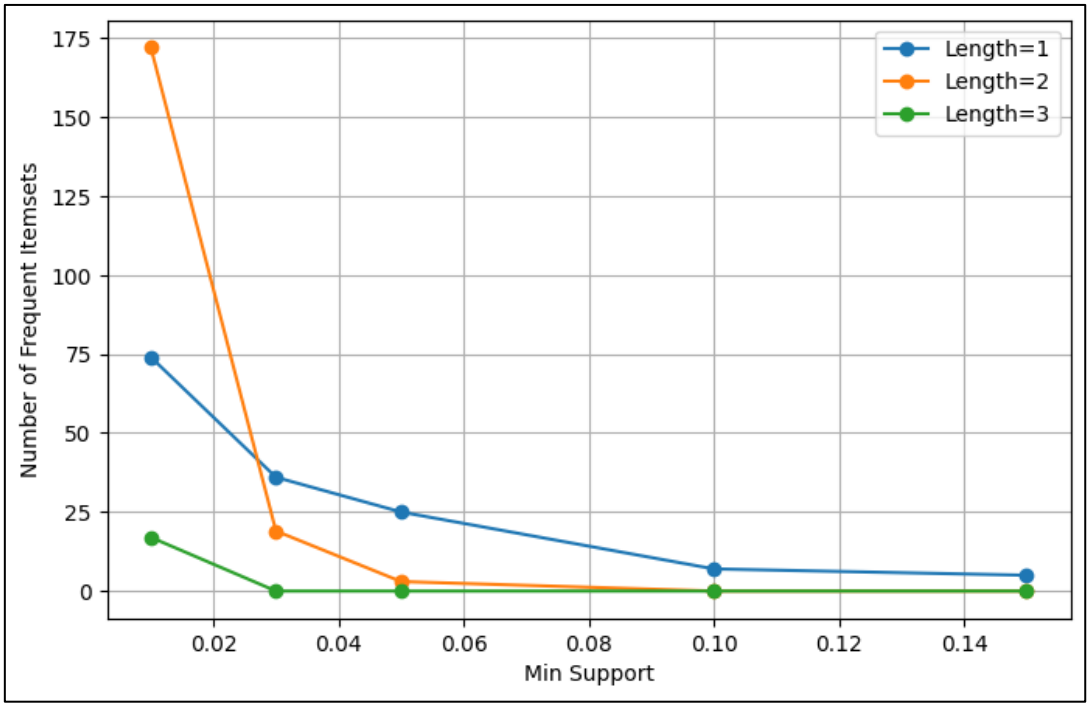

Рисунок 4 – Диаграмма результатов разной длины

1. При увеличении порога поддержки уменьшается общее количество частых наборов.
2. Уменьшается также средняя длина частых наборов.
3. Большие по размеру частые наборы исчезают при высоких значениях порога поддержки.

В ходе работы был реализован алгоритм Apriori для поиска частых наборов товаров. Проведенные эксперименты показали:

- Время выполнения алгоритма уменьшается при увеличении порога поддержки.
- Число частых наборов уменьшается при увеличении порога поддержки.
- Большие по размеру наборы встречаются только при низких значениях порога поддержки.