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 лабораторной работы № 1

по дисциплине  
«Технологии аналитической обработки информации»

|  |  |
| --- | --- |
|  | Выполнил: студент группы КЭ-403  Разуев Г.А.  Проверил:  Преподаватель кафедры СП  Гоглачев А.И. |

Челябинск-2025

**ОГЛАВЛЕНИЕ**

[1. ЗАДАНИЕ 2](#_Toc191724011)

[2. РЕАЛИЗАЦИЯ АЛГОРИТМА 3](#_Toc191724012)

[3. ЭКСПЕРИМЕНТЫ 5](#_Toc191724013)

[3.1. Загрузка данных и тестовый запуск алгоритма 5](#_Toc191724014)

[3.2. Эксперименты с варьированием порогового значения 5](#_Toc191724015)

# 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:
   1. сравнение быстродействия на фиксированном наборе данных при изменяемом пороге поддержки;
   2. 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:
   1. формулировка задания;
   2. гиперссылка на каталог репозитория с исходными текстами, наборами данных и др. сопутствующими материалами;
   3. рисунки с результатами визуализации;
   4. пояснения, раскрывающие смысл полученных результатов.

# РЕАЛИЗАЦИЯ АЛГОРИТМА

Реализация алгоритма представлена в классе «AprioriSolver», который инкапсулирует логику по выполнению поиска частых наборов объектов в заданном наборе данных с помощью алгоритма Apriori. В результате метод «solve» реализованного класса возвращает список результирующих наборов должен, который содержит 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). Код метода «solve» реализованного класса представлен в листинге 1.

Листинг 1 – Код метода «solve»

def solve(self, data: List[List], min\_support: float, sort\_method: str = 'support', verbose: bool = False) -> pd.DataFrame:

"""Метод для запуска алгоритма"""

# Исходные частые наборы

frequent\_itemsets = self.\_get\_frequent\_itemsets(data, min\_support, verbose)

# Преобразуем результаты в DataFrame

results = [(tuple(sorted(itemset)), count) for itemset, count in frequent\_itemsets.items()]

results\_df = pd.DataFrame(results, columns=['items\_set', 'support'])

# Рассчитываем поддержку

results\_df['support'] = results\_df['support'] / len(data)

# Упорядочиваем результаты

if sort\_method == 'support':

results\_df = results\_df.sort\_values(by='support', ascending=False)

elif sort\_method == 'lexical':

results\_df = results\_df.sort\_values(by='items\_set')

else:

raise ValueError('Invalid sort method')

return results\_df.reset\_index(drop=True)

Код реализованной программы и всех проведенных экспериментов находится в репозитории по ссылке https://github.com/ke103rga/analytics.

Алгоритм был протестирован на наборе данный, представленном в презентации. Результаты работы реализованного алгоритма 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']

]

apriori\_solver = AprioriSolver()

min\_support = 0.  *# Порог поддержки*

result = apriori\_solver.solve(data, min\_support, sort\_method='lexical', verbose=False)

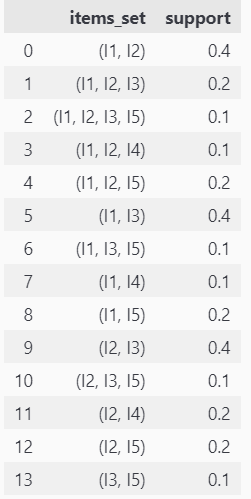


Рисунок – Результаты применения реализованного алгоритма

# ЭКСПЕРИМЕНТЫ

## 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

baskets = pd.read\_csv(os.path.join(DATA\_PATH, 'baskets.csv'), encoding='windows-1251', header=None)

baskets\_transactions = [list(row.dropna()) for index, row in baskets.iterrows()]

apriori\_solver.solve(baskets\_transactions, 0.01)



Рисунок – Результат запуска алгоритма

## Эксперименты с варьированием порогового значения

Были проведены эксперименты на наборе данных baskets.csv (сведения о покупках в супермаркете). В экспериментах варьировалось пороговое значение поддержки (0.01, 0.015, 0.02, 0.025, 0.03, 0.05, 0.10).

На основании проведенных экспериментов была выполнена визуализация результатов с использованием линейной диаграммы, на которой ось X – значение порога поддержки, первая ось Y – время выполнения в секундах, вторая ось Y – количество частых наборов. Результат представлен на рисунке 3.

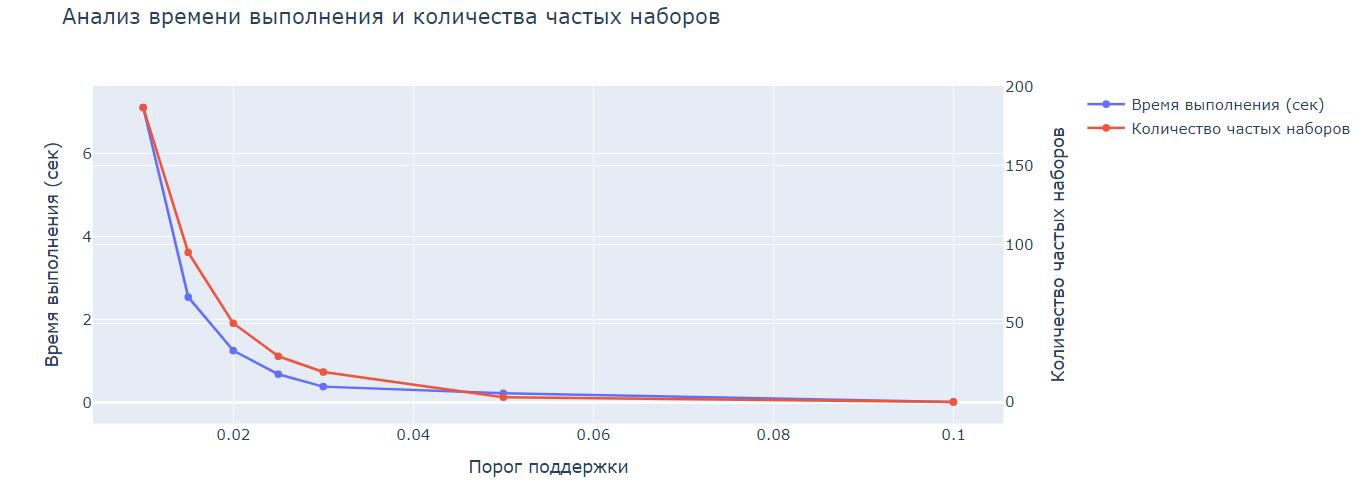


Рисунок – Диаграмма результатов эксперимента

Из анализа данного графика можно сделать ряд выводов:

1. Метрики времени выполнения и количества полученных наборов имеют около единичную корреляцию, что говорит об оптимизированности алгоритма, которая позволяет не проверять заведомо не обладающие или наоборот обладающие необходимым значением поддержки наборы;
2. И время выполнения и количество результирующих наборов экспоненциально убывают с ростом порогового значения поддержки;
3. В данном блоке данных нет ни одного набора с уровнем поддержи 10% или более;
4. В данном блоке данных лишь три набора с уровнем поддержи 5% или более.