**Software Design Document**

Herugrim

**Team Members:**

|  |  |
| --- | --- |
| **Programmer #1:** | Scott Barrett |
| **Programmer #2:** | David Johnston |
| **Programmer #3:** | Jack Storm |
| **Tester:** | Zigmyal Wangchuk |

**Table of Contents**

[1. Introduction](#h.gjdgxs)

[1.1 Purpose](#h.30j0zll)

1.2 Scope

[1.3 System Overview](#h.tyjcwt)

[1.4 Terminology](#h.3dy6vkm)

[1.5 Overview](#h.1t3h5sf)

[2. Design Considerations](#h.4d34og8)

[2.1 General Constraints](#h.2s8eyo1)

[2.2 Assumptions and Dependencies](#h.17dp8vu)

[2.3 Goals and Guidelines](#h.3rdcrjn)

[3. Data Architecture](#h.26in1rg)

[4. System Architecture](#h.lnxbz9)

[4.1 Subsystem Architecture](#h.35nkun2)

[5. Interface](#h.1ksv4uv)

[6. Gantt Chart](#h.44sinio)

# Introduction

## Purpose

The purpose of this document is to outline the architecture of our implementation of the Apriori algorithm.

## Scope

We will be producing all of the software necessary to analyze a large dataset using the Apriori algorithm. The software will be capable of loading a data set from a file and finding associations according to preset support thresholds. The output will go in the root folder.

## System Overview

The system is being designed exactly to the specification of the assignment. We have decided to prioritize speed over memory efficiency, which has lead us to develop Dynamic 2D array which will store the items in each transaction.

## Terminology

Apriori algorithm: An association mining algorithm commonly used as a baseline by which other data mining algorithms are measured.

## Overview

This document provides a full description of our implementation of the Apriori algorithm, as well as an overview of our timeline for implementing the system. The document is organized according to the table of contents above.

# Design Considerations

## General Constraints

User must have a working computer. User must have Windows 7. User must have a Visual Studios C++ compiler. User should be able to generate datasets in the assumed fashion. User should have a full day for algorithm to run.

## Assumptions and Dependencies

The items in the transactions in the original dataset must be sorted in ascending order, if not then the system can potentially miss associations.

## Goals and Guidelines

We are going to use as much memory as possible and have an emphasis on the speed. We have decided to go this route because Apriori can take an extremely long time and we would like to minimize that as much as possible.

# Data Architecture

## Data Structure

|  |
| --- |
| **Initial Dataset** |
| - mArray: int[][] \*\* |
| + Dataset()  + ~Dataset()  + generateItemSet(nItemSet: int)  + populateArray(fileName: string) |

* + The rationale for choosing this data structure.
    - Relatively simple to implement and it is fast
  + What are the advantages of this data structure?
    - Arrays are fast to traverse through
    - Simple to implement
  + What are the disadvantages of this this data structure?
    - Takes a lot of memory
  + What are the alternative choices:
    - LinkedList

|  |
| --- |
| Queue |
| - mHead: Node<T> \*  - mTail: Node<T> \*  - mCount: int |
| + Queue()  + ~Queue  + getCount(): int  + dequeue(): T  + display(): void  + enqueue(data: T): void  + isEmpty(): bool |

* + The rationale for choosing this data structure.
    - Relatively simple to implement
  + What are the advantages of this data structure?
    - Stores Data in a certain order
    - First in, first out
  + What are the disadvantages of this this data structure?
    - Can only dequeue and enqueue
  + What are the alternative choices:
    - LinkedList
    - Stack

## Data Format

Both the input and output files are expected to done with text files. The input for the original dataset will be read from one file, and the program will output in text files containing the different n-itemset. One for each n-itemsets found.

# System Architecture

The primary idea behind the architecture of our system was to design a 2D Array of integers. The columns in the SD array represents the item in the dataset, and the rows represent a transaction. So a location in the 2D array is a potential item in a transaction. If the location contains 1, then the item is in the transaction, otherwise it will contain a 0.

## Subsystem Architecture

The 2D array will essentially be a table of all the items in the originally dataset. The set up of (transactions, item) will allow us to go through column by column and tally up the number of times the itemset occurs overall and see if it reaches the minimum support threshold. Using a series of for-loops the system will traverse the table finding all the elements that meet the minimum support threshold for each n-itemset. This will go until searching for n-itemset yields zero results, as is the apriori algorithm.

# Interface

The system will prompt the user for the name of a text file containing the data, and the number of possible items in the dataset.

The system will then indicate to the user that it is working through the Apriori algorithm. After each item set is generated, the system will tell the user how long it took to generate the item set. When the task is complete the system will tell the user it is finished and prompt the user to exit the program.

# Gantt Chart



