SE2XB3 OptimizeU V1.0
Design Specification

Final Project for SFWRENG 2XB3:Software Engineering Practice and Experience: Binding Theory to Practice

Group 8
Members:

Lucas Matthew Dutton
Michael Le
Saad Khan
Omar Elemary

McMaster University
Department of Computing and Software

Created and Edited on: April 1, 2018

Contents

1 Revisions

1.1 Revision History

Revision History

```
commit a748c1a6482f27b152745485942452f59440b7ca
Author: Michael Le <lem12@mcmaster.ca>
Date:   Mon Feb 26 20:15:07 2018 -0500

    Initial commit

commit 6c1e97dce4b6d9a60fc9c23689a20e5489e8aa63
Author: Necried <duttonl@mcmaster.ca>
Date:   Mon Feb 26 20:28:02 2018 -0500

    added docs directory

commit 3aa5cffc3768973d9323bdd126545688337c1cdf
Author: Necried <duttonl@mcmaster.ca>
Date:   Mon Feb 26 20:34:06 2018 -0500

    Added helpful readme

commit 7f77082b2d7264cc647b66ce156781e04ccfdb28
Author: Necried <duttonl@mcmaster.ca>
Date:   Tue Feb 27 10:58:30 2018 -0500

    added src folder

commit 8daffe3cf9594f231d7401aca8b03571293476ba
Author: Necried <duttonl@mcmaster.ca>
Date:   Tue Feb 27 11:01:20 2018 -0500

    Testing java commits

commit c677fd1122b34fb2330142c28d5084e5801970a6
Author: Necried <duttonl@mcmaster.ca>
Date:   Tue Feb 27 12:14:13 2018 -0500

    Added meeting minutes folder

commit 33b301c33bf84e9fc01ff55bede6f0e7a42e65bb
Author: Necried <duttonl@mcmaster.ca>
Date:   Tue Feb 27 12:15:05 2018 -0500

    populated readme

commit 90af85614d8b0a86dacb52609f3e867a478a608c
Author: Necried <duttonl@mcmaster.ca>
Date:   Tue Feb 27 12:18:17 2018 -0500
```

```
    added Milestone1

commit 32a141ccf282cf8a5ffd5725ef7ca511883d80d5
Author: Necried <duttonl@mcmaster.ca>
Date:   Sat Mar 3 17:30:26 2018 -0500

    Initial k-means and coordinates ADT

commit 3fec7d578a60192edbfd238bae3d08b989c6432b
Author: Necried <duttonl@mcmaster.ca>
Date:   Sat Mar 3 17:54:15 2018 -0500

    Fully added comments and unit testing in main method

commit adb0a2fadb494e1a88eec340224fb9149f96eea8
Author: Necried <duttonl@mcmaster.ca>
Date:   Sat Mar 3 18:01:19 2018 -0500

    removed test class file

commit fce89457f20fed11d591fba127f6d0035cb7c5b6
Author: Necried <duttonl@mcmaster.ca>
Date:   Mon Mar 5 21:01:33 2018 -0500

    Added hidden dependencies

commit 00e0473b79db337291f0aedf32d47966337b9116
Author: le-michael <lem12@mcmaster.ca>
Date:   Tue Mar 6 00:20:35 2018 -0500

    Cord and Cluster ADT  Basic Display

commit 7d928e70f10d4d908ac61a12ff6d6411f62218ed
Author: Omar Elemary <elemaryo@mcmaster.ca>
Date:   Tue Mar 6 07:32:24 2018 -0500

    Unfunctional Kruskal

commit 4b0a72bce189e4ad8bd75e9dfcaf4343369c1553
Author: le-michael <lem12@mcmaster.ca>
Date:   Tue Mar 6 14:49:04 2018 -0500

    Animation

commit 99990facd2d0f5ec84d86fce5da324dfcbb7cfac
Author: Necried <duttonl@mcmaster.ca>
Date:   Tue Mar 6 11:16:20 2018 -0500

    Updated to ADTs
```

```
commit 8803c1eb1a7aa6ed5fc08ce16b4c7afe4c4bce2c
Author: Necried <duttonl@mcmaster.ca>
Date:   Tue Mar 6 11:22:23 2018 -0500

    im so bad

commit 22251014dc653cf258be560af0fe94ab6ccd2370
Author: Necried <duttonl@mcmaster.ca>
Date:   Tue Mar 6 11:24:10 2018 -0500

    Emptied main method

commit a06041a5d3101e6d0287d311d82c755cabebc6b4
Author: Saad <thesaadfather@gmail.com>
Date:   Tue Mar 6 11:43:39 2018 -0500

    Dijkstra

commit 5cbde859eccedc18a0e20231f8084556309ae178
Author: Saad <thesaadfather@gmail.com>
Date:   Tue Mar 6 12:06:27 2018 -0500

    Dijkstra

commit 6e63dc33d44062f4b5f26e189c3290743e3c384e
Author: le-michael <lem12@mcmaster.ca>
Date:   Tue Mar 6 16:52:11 2018 -0500

    Display K-Means

commit 328f125652a31dd2503641b99c6ff87b3efc6f57
Merge: 6e63dc3 a06041a
Author: le-michael <lem12@mcmaster.ca>
Date:   Tue Mar 6 16:52:44 2018 -0500

    Merge branch 'master' of https://github.com/le-michael/2xb3-optimizeu

commit 6d568fbafebd5f10d557d828710293c8363b143d
Author: le-michael <lem12@mcmaster.ca>
Date:   Tue Mar 6 17:01:54 2018 -0500

    Background colour

commit 33801f1ef00d377107486db96e7fc582070ec152
Author: Necried <duttonl@mcmaster.ca>
Date:   Tue Mar 6 13:17:30 2018 -0500

    Fixed Kruskals

commit 736305b42c9dbd65b0753f0d92c20e2f1e014d6b
Merge: 33801f1 5cbde85
```

```
Author: Necried <duttonl@mcmaster.ca>
Date:   Tue Mar 6 13:18:28 2018 -0500

    Merge branch 'master' of https://github.com/le-michael/2xb3-optimizeu

commit cc8d367a1af48cb22df2d3c153a98fc26315ddbd
Author: Necried <duttonl@mcmaster.ca>
Date:   Thu Mar 8 16:06:35 2018 -0500

    Updated readme

commit 0ba7b8cea753138d1dbf80efe996bc500ee8e2d9
Author: Necried <duttonl@mcmaster.ca>
Date:   Fri Mar 9 11:19:48 2018 -0500

    updated docs

commit f91ad01a1f674e282d9d9e9118fa7354494d6319
Author: Necried <duttonl@mcmaster.ca>
Date:   Tue Mar 13 22:20:22 2018 -0400

    Updated to linked list Cords

commit 51300a97e47ea1405b1c9333929e57da4127400e
Author: le-michael <lem12@mcmaster.ca>
Date:   Tue Mar 13 22:24:33 2018 -0400

    Class for loading data set

commit 5e07ae2f42057d79c28d753e975fae3d04b76c0c
Author: Necried <duttonl@mcmaster.ca>
Date:   Wed Mar 14 00:03:23 2018 -0400

    Minor changes

commit 67854df60afc24029479fbaa4f9ca17c733dbc60
Author: le-michael <lem12@mcmaster.ca>
Date:   Wed Mar 14 00:19:27 2018 -0400

    Fixed kmeans

commit 56dfb224b1e149bd0564b6e7a15eae0cbca52680
Author: Necried <duttonl@mcmaster.ca>
Date:   Wed Mar 14 20:48:07 2018 -0400

    initial normalize added

commit b115a172c68d12c50315d19d51814b6dfc33fddc
Author: Necried <duttonl@mcmaster.ca>
Date:   Wed Mar 14 20:50:31 2018 -0400
```

```
    normalized to 500

commit 06d1ccb6447265c84e98b083aa38e69dc4e43ed4
Author: Omar Elemary <elemaryo@mcmaster.ca>
Date:   Wed Mar 14 20:54:49 2018 -0400

    Meeting6

commit 765a08b341739f8a817d81107981dcfbc0566358
Author: Necried <duttonl@mcmaster.ca>
Date:   Wed Mar 14 21:49:22 2018 -0400

    Load fix

commit 0249c0463c62c42756c0ba1823ab068c996fa203
Merge: 765a08b 06d1ccb
Author: Necried <duttonl@mcmaster.ca>
Date:   Wed Mar 14 21:49:44 2018 -0400

    Merge branch 'master' of https://github.com/le-michael/2xb3-optimizeu

commit 3a5b1bedd3e61042a6d9c352f8089356f282bc86
Author: Omar Elemary <elemaryo@mcmaster.ca>
Date:   Wed Mar 14 23:10:21 2018 -0400

    cord commented

commit 204b8af524060732303f8d2128dc0498d31ac594
Author: Necried <duttonl@mcmaster.ca>
Date:   Thu Mar 15 11:36:42 2018 -0400

    Added Kruskals with custom ADTs

commit 63777bace85dcdcbc74eb8d16be1b73032387ee8
Merge: 204b8af 3a5b1be
Author: Necried <duttonl@mcmaster.ca>
Date:   Thu Mar 15 11:39:24 2018 -0400

    Merge branch 'master' of https://github.com/le-michael/2xb3-optimizeu

commit 5952707dc9c9506376c190a8f16164f6c00f056a
Author: le-michael <lem12@mcmaster.ca>
Date:   Fri Mar 16 09:18:56 2018 -0400

    Project Proposal

commit b77cdda1ec1ae7064d8d9555d4815e5430d96a3b
Author: Necried <duttonl@mcmaster.ca>
Date:   Fri Mar 16 09:41:38 2018 -0400

    Added meetings
```

```
commit 065793face845b0689bdaa4fbc0f75324754abc1
Author: le-michael <lem12@mcmaster.ca>
Date:   Sun Mar 18 20:23:29 2018 -0400

    Proper coordinate normalization + Data visualization

commit 469a795a3195ab28ec7880dc17128658cae00227
Author: Omar <oelemary@gmail.com>
Date:   Fri Mar 23 10:03:35 2018 -0400

    Meeting8

commit 0fa22584311fe312eab0744fd6762c85ea790cdc
Author: Necried <duttonl@mcmaster.ca>
Date:   Wed Mar 28 17:00:10 2018 -0400

    Updated README

commit d333ae60082742033a6b2c7b89dcd26daaefd95e
Author: Necried <duttonl@mcmaster.ca>
Date:   Wed Mar 28 20:24:48 2018 -0400

    Unit testing added to TODO

commit 55333e51d3d2b61057f63929061d91ff3765f6b1
Author: le-michael <lem12@mcmaster.ca>
Date:   Wed Mar 28 21:10:03 2018 -0400

    readme

commit 1d119826e40b0b745c3906d1e83145bbdc7c1291
Merge: 55333e5 d333ae6
Author: le-michael <lem12@mcmaster.ca>
Date:   Wed Mar 28 21:10:54 2018 -0400

    Merge branch 'master' of https://github.com/le-michael/2xb3-optimizeu

commit 6e50f8d337821d37d99c74a975744fc130575ad5
Author: le-michael <lem12@mcmaster.ca>
Date:   Wed Mar 28 21:13:03 2018 -0400

    readme

commit 5ca76ec84b6dffbc3ea574cc689f610d2df9bb20
Author: Necried <duttonl@mcmaster.ca>
Date:   Wed Mar 28 21:37:03 2018 -0400

    added presentation draft in docs dir

commit 4cb6f90f02bcf7a55a698b46ad3e4f6ff2492e76
```

```
Author: Necried <duttonl@mcmaster.ca>
Date:   Thu Mar 29 19:30:27 2018 -0400

    Updated draft

commit 835b70d28217c4747851d2cce6d2fb09f179e13f
Author: Omar <oelemary@gmail.com>
Date:   Thu Mar 29 20:18:22 2018 -0400

    Meeting9

commit 8ae247773b5d387aca51fc12aff56463aba92463
Author: le-michael <lem12@mcmaster.ca>
Date:   Fri Mar 30 22:50:22 2018 -0400

    Clean slate

commit 81d9cf5a170ad3135f3baba2fba007df7499b6d0
Author: le-michael <lem12@mcmaster.ca>
Date:   Fri Mar 30 23:03:08 2018 -0400

    New cluster and new cord

commit 2c697258561c6c2ebcf3d9867e00bc7621b4b094
Author: Necried <duttonl@mcmaster.ca>
Date:   Fri Mar 30 23:40:30 2018 -0400

    KMeans with ArrayList updated

commit b86d309f9c3749eb242e804085462dd842cf9dc2
Author: Necried <duttonl@mcmaster.ca>
Date:   Fri Mar 30 23:58:53 2018 -0400

    Updated Kruskal and dependencies to ArrayList

commit 11e3a41d66abe191d4a4f08b690a003427515247
Author: le-michael <lem12@mcmaster.ca>
Date:   Sat Mar 31 01:11:29 2018 -0400

    Load and Display conversion

commit 1c0247c32ed68620c708253ae81e00fde4fe6b84
Author: Saad <thesaadfather@gmail.com>
Date:   Sat Mar 31 01:47:31 2018 -0400

    Pushing Slides

commit a532f88a49a11c6d588268abbc0c3466e2e4fa00
Author: le-michael <lem12@mcmaster.ca>
Date:   Sat Mar 31 20:08:53 2018 -0400
```

```
    gui update

commit f27d3ba59e003d402893ac9d73a886d4e8785cf1
Merge: a532f88 1c0247c
Author: le-michael <lem12@mcmaster.ca>
Date:   Sat Mar 31 20:09:20 2018 -0400

    Merge branch 'master' of https://github.com/le-michael/2xb3-optimizeu

commit 7beb0a8650e2b8ce99e831fc330b0d1580b9440c
Author: le-michael <lem12@mcmaster.ca>
Date:   Sat Mar 31 22:13:36 2018 -0400

    Fully finished gui

commit 4f7e746c67c373edcd56e8a5cb49c70d049c8ed4
Author: le-michael <lem12@mcmaster.ca>
Date:   Sat Mar 31 22:19:24 2018 -0400

    Clean up
```

1.2 Team Details

Name	Student Number	Roles
Michael Le	400070369	Group Leader, Graphics Implementation
Lucas Dutton	400052930	Log Admin, Algorithms Implementation
Saad Khan	400085498	Documentation Designer, Server Researcher
Omar Elemary	400100169	Algorithms Researcher, Log Assistant

1.3 Attestation and Consent

By virtue of submitting this document we electronically sign and date that the work being submitted by all the individuals in the group is their exclusive work as a group and we consent to make available the application developed through SE2XB3 project, the reports, presentations and assignments (not including my name or student number) for future teaching purposes.

2 Contributions

Note: Member roles are outlines in **Revisions: Team Details**

Name	Contributions	Comments
Lucas Dutton	Implemented Kruskals, K-Means and Associated ADT (Abstract Data Type)	Responsible for implementing and updating algorithms with any project changes
	Project Log admin	Responsible for monitoring and approving Project Logs
	Documentation editor	Responsible for creating and sharing documents such as Requirements and Design Specifications
Michael Le	Implemented graphical representation of outputs	Responsible for presenting data in a graphical environment in Java
	Implemented graphical ADTs	Cluster and Cord ADTs made for interaction between algorithms and graphics
	Source Code documentation	Comment each methods in each module
Saad Khan	Presentation Facilitator	Responsible for organizing presentation slides
	Graphics creator for documents	Responsible for creating diagrams for documents, e.g. UML
Omar Elemary	Meeting Minutes upkeep	Responsible for creating and updating project logs
	Unit tester	Responsible for testing each modules and the application

3 Executive Summary

OptimizeU is intended to be a portable app that addresses the problem of finding an Uber pickup on busy nights, as well as optimizing driving routes of Uber drivers. The project leverages a dataset of twenty million Uber pick ups in New York City to help drivers locate the busiest hotspots in the city, routing more cars to denser locations which allows more users to find access to an Uber vehicle in a faster time. The application uses various algorithms related to machine learning and graphing, as well as pre-processing the dataset with searching and sorting algorithms to generate a pick-up density map of the city. The map will contain heat clusters related to density and distances between each cluster to inform the driver of optimal routes that can be taken. In short, the aim of OptimizeU is decrease wait times of passengers and maximize profits of Uber drivers.

4 Design Description

4.1 Module Description and Decomposition

4.1.1 Module Overview

The table below shows a top-level view of each class in the application. Classes are grouped by their functionality and dependency on one another.

Function	**Class Name**	**Description**
Basic Representation	Cord	2-Dimensional representation of coordinates
Clustering	Cluster	Representation of a cluster produced by K-Means
	KMeans	Clustering algorithm API
Minimum Spanning Tree	KruskalMST	Driver class to calculate Minimum Spanning Tree
	Edge	ADT representation of weighted edges
	Graph	Module to instantiate a graph with directed edges
	UF	Helper module for connecting vertices in a graph
	Heap	Heapsort on edges for Kruskals
Input/Output	Load	Reads dataset, sorts them according to time, and puts each coordinate in a Hash Table
	drawSurface	Graphics implementation of clusters and MST
	demoFrame	Main driver of application

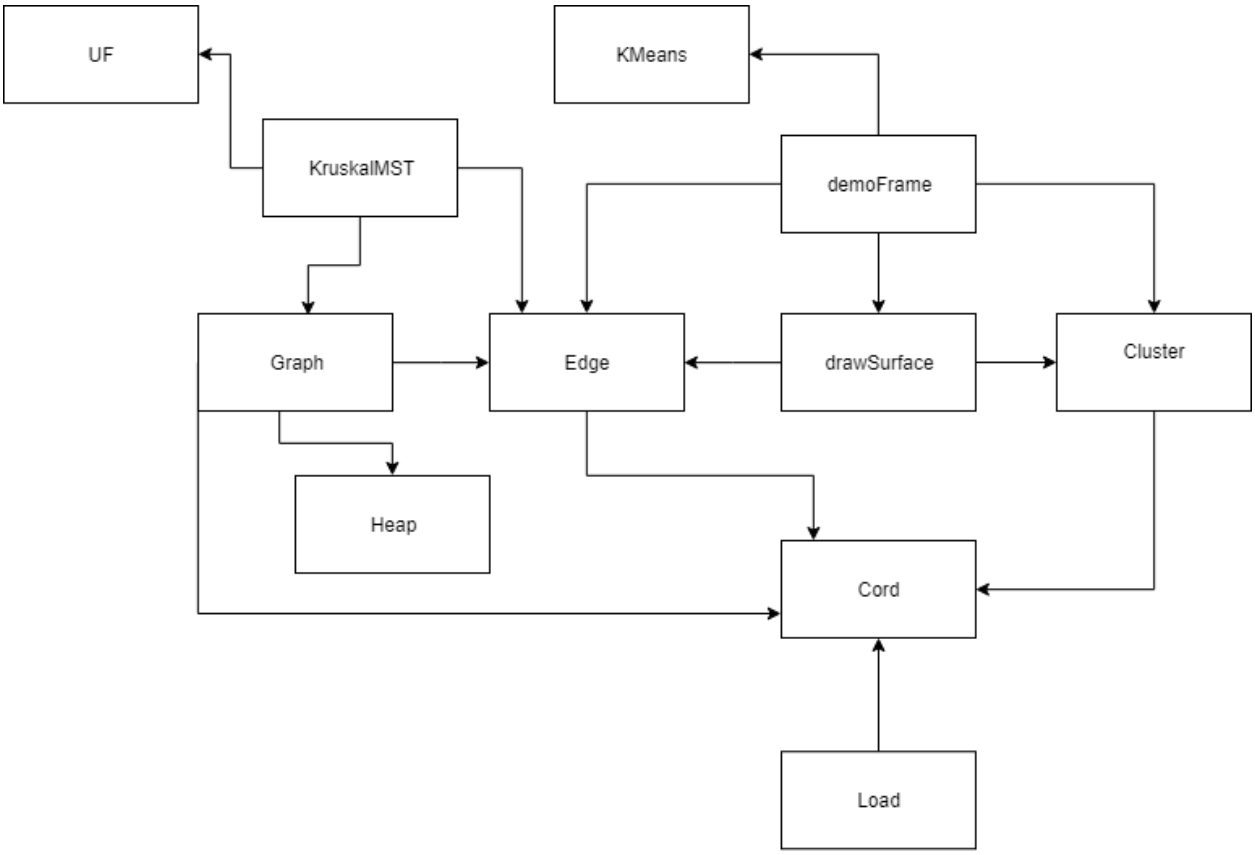
4.1.2 Module Decomposition Semantics

Following the table in the previous section, the following items below provide a more detailed description of the rationale behind the modular decomposition of the design.

- **Basic Representation:** Since the application makes use of coordinates on a two-dimensional plane, the Cord API makes available a convenient coordinate ADT that is used whenever a point in the plane needs to be represented or stored.
- **Clustering:** KMeans and its associated ADT, Clusters, are used to compute clusters from a random number of points. Clusters are stored in the ADT as a point representing the mean of the cluster, called the centroid. All points associated to the centroid are stored in the ADT as well
- **Minimum Spanning Tree:** Kruskal's algorithm is used to generate the minimum spanning tree of the clusters' centroids. All the associated modules are helpers to the algorithm, such as union-find, sorting, and graph/edge representations.
- **Input and Output:** This can be divided by itself into two parts: input and output. Load is the input routine of the application: It leverages countsort to sort input data according to the pickup time which is partitioned by th 24-hour system. The other two modules are used to collect output from the algorithms and display it on a graphic canvas.

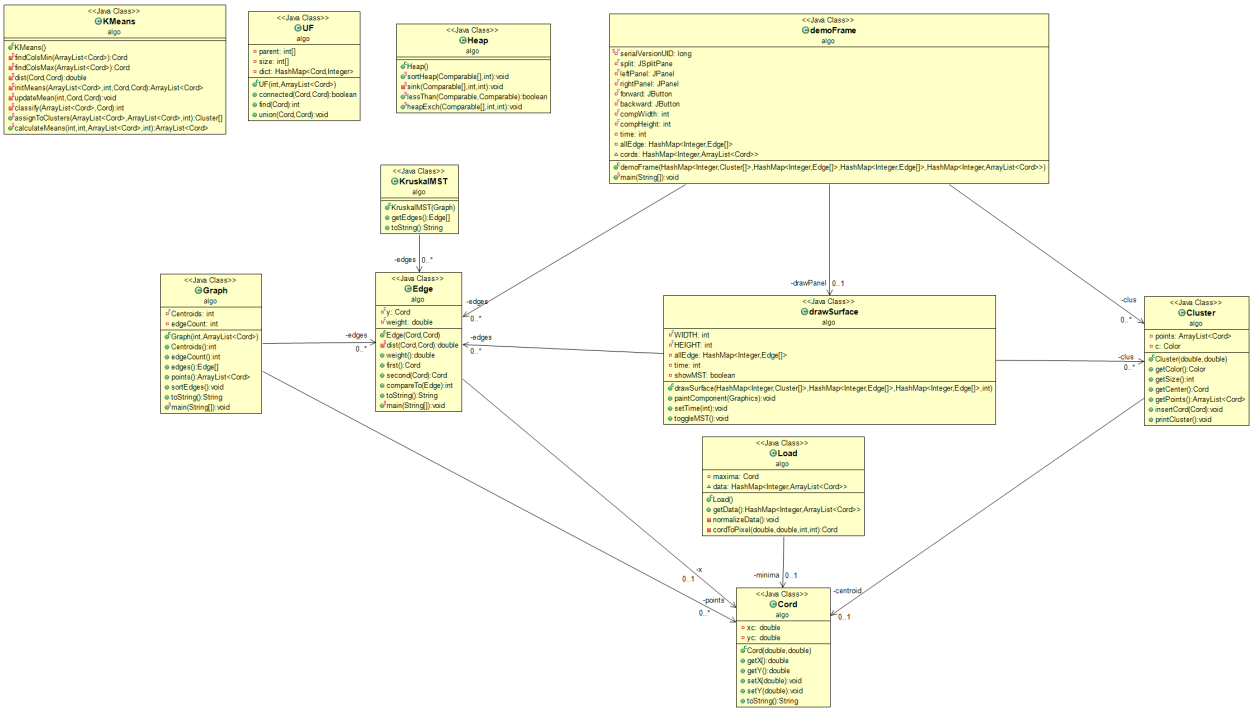
4.1.3 Uses Relation

An overlay of the application can be viewed in the Uses relationship below:



4.1.4 UML Class Diagram

An overlay of the application can be viewed in the UML class diagram below:



4.2 Module Interface Specification

This section will cover:

- A description of the interface of each public entity, as well as its syntax and semantics
- A description of the implementation of private entities of each module, state variables and how the methods maintain these state variables
- Requirements trace back for each module

Several items should be noted about the module specifications:

- Where possible, a more formal, mathematical notation similar to conventions used in SE2AA4 will be used.
- A natural language description will be used for methods that are too complicated or incomprehensible to be expressed with discrete mathematics
- Each module specification will cover both public and private entities. Care is taken to seperate between the two for each module to prevent ambiguity issues.

Cord Module

Top Level Description

An abstract data type to represent a point in a two-dimensional space

Module

Cord

Uses

N/A

Syntax

Exported Types

Cord = ?

Exported Access Programs

Routine name	In	Out
Cord	$\mathbb{R}$, $\mathbb{R}$	Cord
getX		$\mathbb{R}$
getY		$\mathbb{R}$
setX	$\mathbb{R}$	
setY	$\mathbb{R}$	
toString		String

Semantics

State Variables

xc: $\mathbb{R}$
yc: $\mathbb{R}$

State Invariant

None

Access Routine Semantics

Cord(x,y):
Constructor for a coordinate point in a 2D-plane

- transition: $xc, yc := x, y$
- output: $out := self$

getX():
Accessor for the x-coordinate

- output: $out := xc$

getY():
Accessor for the y-coordinate

- output: $out := yc$

setX(x):
Mutator for the x-coordinate

- transition: $xc := x$
- output: None

setY(y):
Mutator for the y-coordinate

- transition: $yc := y$
- output: None

toString():
Override for Java method toString. Represents a coordinate in the form of a string "(x,y)" where x and y are xc and yc respectively.

Cluster Module

Top Level Description

Stores the centroid of a clusters and points associated with the cluster.

Module

Cluster

Uses

Cord, Color, ArrayList

Syntax

Exported Types

Cluster = ?

Exported Access Programs

Routine name	In	Out
Cluster	$\mathbb{R}$, $\mathbb{R}$	Cluster
getColor		*Color*
getSize		$\mathbb{N}$
getCenter		Cord
getPoints		sequence of Cord
insertCord	Cord	
printCluster		

Semantics

State Variables

centroid: Cord
points: seq of Cord
c: Color

State Invariant

None

Assumptions

- c : Color represents the color of the cluster when drawn with Java's graphics library
- centroid represents a Cluster's centroid
- points represents the Cord points associated with the centroid

Access Routine Semantics

Cluster(x,y):
Constructor for Cluster Object

- transition: points, centroid, c :=<>, Cord(x, y), random Color
- output: *out* := *self*

getColor():
Return the color of the cluster

- output: *out* := *c*

getSize():
Return the number of points associated with the cluster

- output: *out* := |points|

getCenter():
Returns the Cord representing the centroid

- output: *out* := centroid

getPoints():
Returns the sequence of points associated to the cluster

- output: *out* := points

insertCord(n):
Insert a point into the sequence of Cord, associating the inserted point with the cluster

- transition: points := points||n
- output: None

printCluster():
Prints each Cord object in the sequence of coordinates

KMeans Library Module

Top Level Description

A module that clusters points using K-Means classification

Module

KMeans

Uses

Cord, Cluster

Syntax

Exported Types

KMeans = ?

Exported Access Programs

Routine name	In	Out
calculateMeans	$\mathbb{N}, \mathbb{N}, \mathbb{N}$, seq of Cord	seq of Cord
assignToClusters	seq of Cord, seq of Cord, $\mathbb{N}$	seq of Cluster

Private Access Programs

Routine name	In	Out
findColsMin	seq of Cord	Cord
findColsMax	seq of Cord	Cord
dist	Cord, Cord	$\mathbb{R}$
updateMean	$\mathbb{N}$, Cord, Cord	
classify	seq of Cord, Cord	$\mathbb{N}$

Semantics

State Variables

None

State Invariant

None

Assumptions

- calculateMeans is called to obtain cluster centroids before a call to assignToClusters.
- Cluster sizes remain consistent between the two method calls.

Access Routine Semantics

calculateMeans(k, datasize, maxIterations, items :: seq of Cord):
KMeans Clustering heuristic. Takes as input:

- k: The amount of clusters to generate
- datasize: The size of the input data
- maxIterations: Loop termination count, in the event that the heuristic is non-deterministic.
- items: The input data of Cords as a sequence

Returns as output a set of cluster centroids as calculated from the input parameters.

assignToClusters(means, items, k : $\mathbb{N}$):
Associate each input point with centroids calculated from calculateMeans. Takes as input:

- k: The amount of clusters to generate
- means: The centroids generated from calculateMeans
- items: The input points

Returns as output a sequence of Cluster objects, each representing points associated to a centroid.

Private Routine Semantics

findColsMin(items):
Output the point denoting the bottom left of the map used to calculate clusters

- output: $out := \exists(c : \text{Cord} | c \in \text{items} \wedge \forall(n : \mathbb{N} | n \in [0..|\text{items}| - 1] : c \leq \text{items}[n]) : c)$

findColsMan(items): *Output the point denoting the bottom left of the map used to calculate clusters*

- output: $out := \exists(c : \text{Cord} | c \in \text{items} \wedge \forall(n : \mathbb{N} | n \in [0..|\text{items}| - 1] : c \geq \text{items}[n]) : c)$

dist(a, b): *Compute the Euclidean distance between two coordinates*

- output: $out := \sqrt{(b.xc - a.xc)^2 + (b.yc - a.yc)^2}$

updateMean(size : $\mathbb{N}$, mean, item):
Updates the mean centroid after adding associating an additonal point to it

- transition: $mean := (\text{mean} * (\text{size} - 1) + \text{item})/\text{size}$
- output: None

classify(means, item : Cord):
Classifies a point to the nearest centroid, returns the index in which the point is associated to in the sequence of centroids

Edge Module

Top Level Description

An object representing weighted edges in a two-dimensional graph

Module

Edge

Uses

Cord

Syntax

Exported Types

Cord = ?

Exported Access Programs

Routine name	In	Out
Edge	Cord, Cord	Edge
weight		$\mathbb{R}$
first		Cord
second	Cord	Cord
compareTo	Edge	$\mathbb{Z}$
toString		String

Private Access Programs

Routine name	In	Out
dist	Cord, Cord	$\mathbb{R}$

Semantics

State Variables

x: Cord
y: Cord
weight: $\mathbb{R}$

State Invariant

None

Access Routine Semantics

Edge(X,Y):
Constructor of an Edge object

- transition: $x, y, weight := X, Y, dist(X, Y)$
- output: $out := self$

weight():
Accessor for the weight of the edge

- output: $out := weight$

first():
Returns the point stored in the state variable x

- output: $out := x$

second(point):
Returns the point that connects from the input point

- output: $out := point = \text{point} = y \Rightarrow x | \text{point} = x \Rightarrow y$
- exception: $exc := point \neq x \vee y \Rightarrow IllegalArgumentException$

compareTo(that):
Implemens compareTo for Java's comparable interface. Used for sorting edges by weight.

- output:

Condition	*out*
weight < that.weight	-1
weight > that.weight	1
weight ≡ that.weight	0

toString():
Java override of toString method. Prints the details of the Edge object

Private Routine Semantics

dist(a, b): *Compute the Euclidean distance between two coordinates. Used for calculating the weight of the edge*

- output: $out := \sqrt{(b.xc - a.xc)^2 + (b.yc - a.yc)^2}$

Heap Library Module

Top Level Description

Heapsorts an array of Comparable Objects. Used in KruskalMST to sort an array of Edges.

Module

Heap

Uses

None

Syntax

Exported Types

Heap = ?

Exported Access Programs

Routine name	In	Out
sortHeap	seq of Comparable, $\mathbb{N}$	

Private Access Programs

Routine name	In	Out
sink	seq of Comparable, $\mathbb{N}, \mathbb{N}$	
lessThan	Comparable, Comparable,	$\mathbb{B}$
heapExch	seq of Comparable, $\mathbb{N}, \mathbb{N}$	

Semantics

State Variables

None

State Invariant

None

Access Routine Semantics

sortHeap(x : seq of Comparable, n : $\mathbb{N}$):

Calls heapsort on the array x of size n. The process is twofold:

1. Builds the heap from the array. The array will obey the max heap property after this process terminates
2. Sorts the array using n extractMaxes

Formally, sortHeap produces an output array which satisfies the following property:

$$\forall(n : \mathbb{N} | n \in [1..|x| - 1] : x[n] \geq x[n - 1])$$

sortHeap happens in-place, which is important in this application as processing a large dataset may constrain available memory.

Private Routine Semantics

sink(x : seq of Comparable, n : $\mathbb{N}$, length : $\mathbb{N}$):
A helper function used to sink elements in a heap. Takes as input:

- x: an array of Comparable objects
- n: The index of the element to sink
- length: The size of x

This function is used to fix any violations in the max heap property.

lessThan(a, b):
Compares a and b using Java's compareTo implementation

heapExch(arr, i ,j):
Swaps elements at index i and j. This accounts for heaps having 1-index starts

- transition: $arr[i-1], arr[j-1] := arr[j-1], arr[i-1]$

Graph Module

Top Level Description

Representation of a directed graph, in the specific application domain of representing the output of clustering with K-Means using cluster centroids as nodes and edges as the connection between each point.

Module

Graph

Uses

Edge, Cord, ArrayList

Syntax

Exported Types

Graph = ?

Exported Access Programs

Routine name	In	Out
Graph	$\mathbb{N}$, seq of Cord	Graph
centroids		$\mathbb{N}$
edgeCount		$\mathbb{N}$
edges		seq of Edge
points		seq of Cord
sortEdges		
toString		

Semantics

State Variables

Centroids: $\mathbb{N}$
edgeCount: $\mathbb{N}$
edges: seq of Edge
points: seq of Cord

State Invariant

None

Access Routine Semantics

Graph(k, means):
Creates a graph representation of cluster centroids

- transition: Centroids, edgeCount, points, edges := k, $(k*(k-1))/2$, means, $\forall(n : \mathbb{N} | n \in [0..|means|-2] \wedge \forall(p : \mathbb{N} | p \in [n+1..|means|-1] \wedge n \neq p) : \text{Edge}(\text{edges.get}(n), \text{edges.get}(p)))$
- output: *out* := *self*

centroids():
Accessor for number of centroids

- output: *out* := Centroids

edgeCount():
Returns the number of edges in the graph

- output: *out* := edgeCount

edges():
Returns all edges objects that are in the graph as a sequence

- output: *out* := edges

points():
Returns all points (vertices) represention the graph as a sequence

- output: *out* := points

sortEdges():
Sorts the edges that are in the state variable edges

toString():
Java override of representing an object as a String. Prints all edges present in the graph.

UF Module

Top Level Description

Stores connected points in a graph. Essential for Kruskal's Algorithm to work.

Module

UF

Uses

Cord, ArrayList, HashMap

Syntax

Exported Types

UF = ?

Exported Access Programs

Routine name	In	Out
UF	$\mathbb{N}$, seq of Cord	UF
connected	Cord, Cord	$\mathbb{B}$
find	Cord	$\mathbb{N}$
union	Cord, Cord	

Semantics

State Variables

parent: seq of $\mathbb{N}$
size: seq of $\mathbb{N}$
dict: HashMap of <Cord, $\mathbb{N}$ >

State Invariant

None

Access Program Semantics

UF(k, means):
Constructor of a Union Find data structure. Creates a bijection between input centroids and enumerable natural numbers in the HashMap construction. This allows UF operations to

operate on natural numbers instead of abstract types.
Transition of state variables on initialization:

- parent: Seq of integers with element equals the index
- size: Seq of integers with element equals one
- dict: Bijection relation between Cord input and an enumerated integer

connected(p, q):
Answers the question: Are p and q in the same set?

- output: *out* := find(p) ≡ find(q)

find(p):
Finds the parent index of the node

- output: *out* := dict.get(p) = parent[p] ⇒ p|find(parent[p])

union(p, q):
Takes two disjoint sets and performs a union on both sets. Internally, this will:

- Make both sets belong to the same parent in the parent sequence
- Adjust the size of sets in the size sequence

KruskalMST Module

Top Level Description

Builds the minimum spanning tree of a given graph

Module

KruskalMST

Uses

Edges, UF, Graph

Syntax

Exported Types

KruskalMST = ?

Exported Access Programs

Routine name	In	Out
KruskalMST	Graph	
getEdges		seq of Edge
toString		String

Semantics

State Variables

edges: seq of Edges

State Invariant

None

Access Program Semantics

KruskalMST(G):
Constructs a minimum spanning tree from the given graph. Stores the edges that makes up the MST graph in a sequence of edges.
The procedure and the corresponding use relations are characterized as follows:

- Calls sortEdges from Graph module to sort each edge according to their weight

- Calls a UF instance to keep track of sets of edges
- For each shortest edge, add it to the edges state variable if it was not previouly included. UF instance keeps track of checking for disjointness

getEdges():
Returns the edges included in the minimum spanning tree

- output: *out* := edges

toString():
Java's toString override, shows edges that are in the minimum spanning tree as a string representation.

Load Module

Top Level Description

Loads the dataset into the program. Implicitly sorts and hashes each input row as well from the input CSV

Module

Load

Uses

HashMap, ArrayList, Cord, Java IO

Syntax

Exported Types

Load = ?

Exported Access Programs

Routine name	In	Out
Load		
getData		HashMap of $< \mathbb{N}, \text{seq of Cord} >$

Private Access Programs

Routine name	In	Out
normalizeData		
cordToPixel	$\mathbb{R}, \mathbb{R}, \mathbb{N}, \mathbb{N}$	Cord

Semantics

State Variables

minima: Cord
maxima: Cord
data: HashMap of $< \mathbb{N}$, seq of Cord$>$

State Invariant

None

Access Program Semantics

Load():
Reads the dataset of Uber pickups from the dataset. For each row, the following occurs:

- The item is classified according to the pickup hour. In other words, Countsort occurs as the data is being processed.
- The pickup coordinates, which are in latitude and longitude, are converted to fit a graphical interface. In other words, pixels are used to represent pickup locations and the geographical map.
- The data is put into the HashMap data state variable with the pickup time as key and pickup coordinates as value.

Note that maxima (top right) and minima (bottom right) of the frame of the graphical interface is recorded as well to obtain the boundaries of the input data.
Lastly, each data point is normalized to fit the boundaries of the graphical interface.

getData():
Returns the result of Load initialization

- output: *out* := data

Private Program Semantics

normalizeData():
Normalizes the data to fit into the display frame, using minima and maxima to obtain the normalization.

cordToPixel(longitude, latitude, mapWidth, mapHeight):
Converts from latitude-longitude to a pixel representation

drawSurface Module

Top Level Description

Graphics package, using JFrame graphics library. This module is used to convert ADTs into a graphical context, such as points and lines. **The class interface, syntax and semantics will not be specified as this is out of the project's scope.**

Module

drawSurface

Uses

HashMap, Cord, Edge, Cluster

demoFrame Module

Top Level Description

Main function for the application. As the module deconstruction is too lengthy, instead the main function's procedure will be listed:

1. Invokes Load to get input data, sorted by hours
2. Set required amount of clusters. This has to be in accordance to requirements, such that a number chosen can generate clusters consistently.
3. Once data input is complete, use KMeans public functions, calculateMeans and assignToClusters to obtain Cluster objects.
4. Cluster centroids are passed into Graph and KruskalMST to get information about centroid connectivity and minimum spanning tree respectively.
5. The results are passed into the drawSurface module to generate the graphical interface.

Module

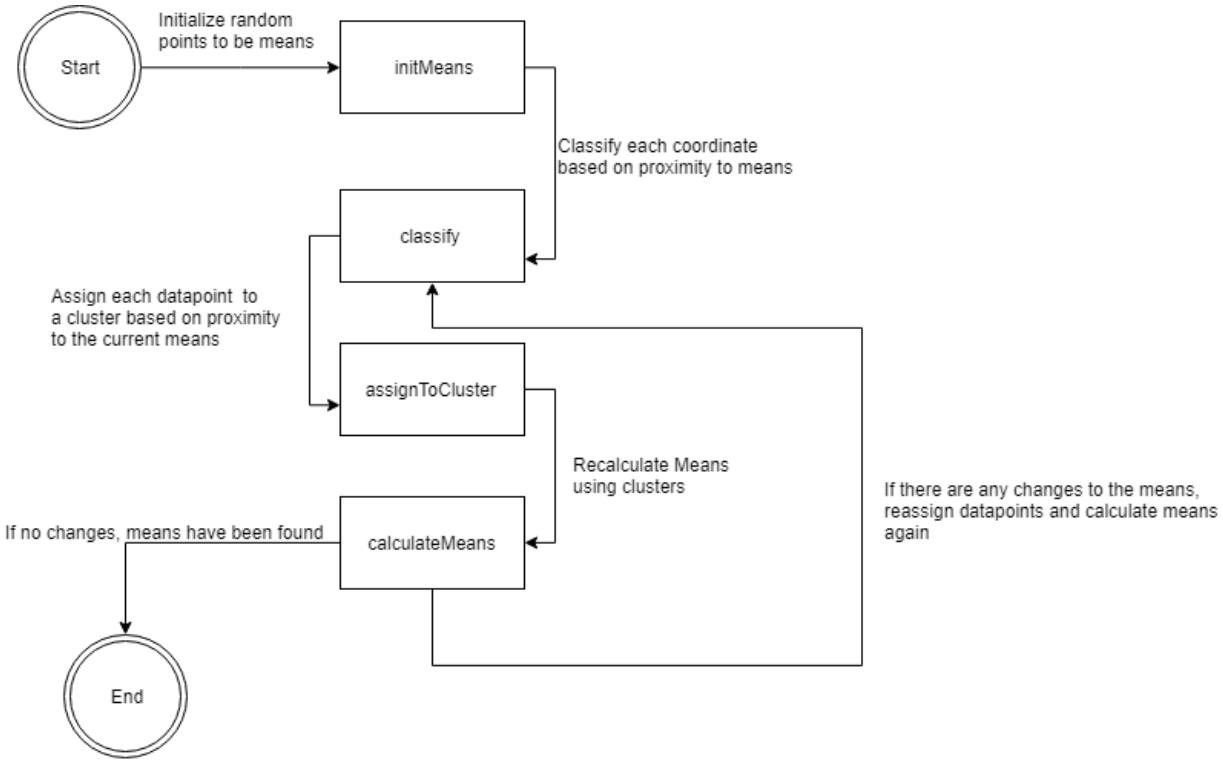
demoFrame

Uses

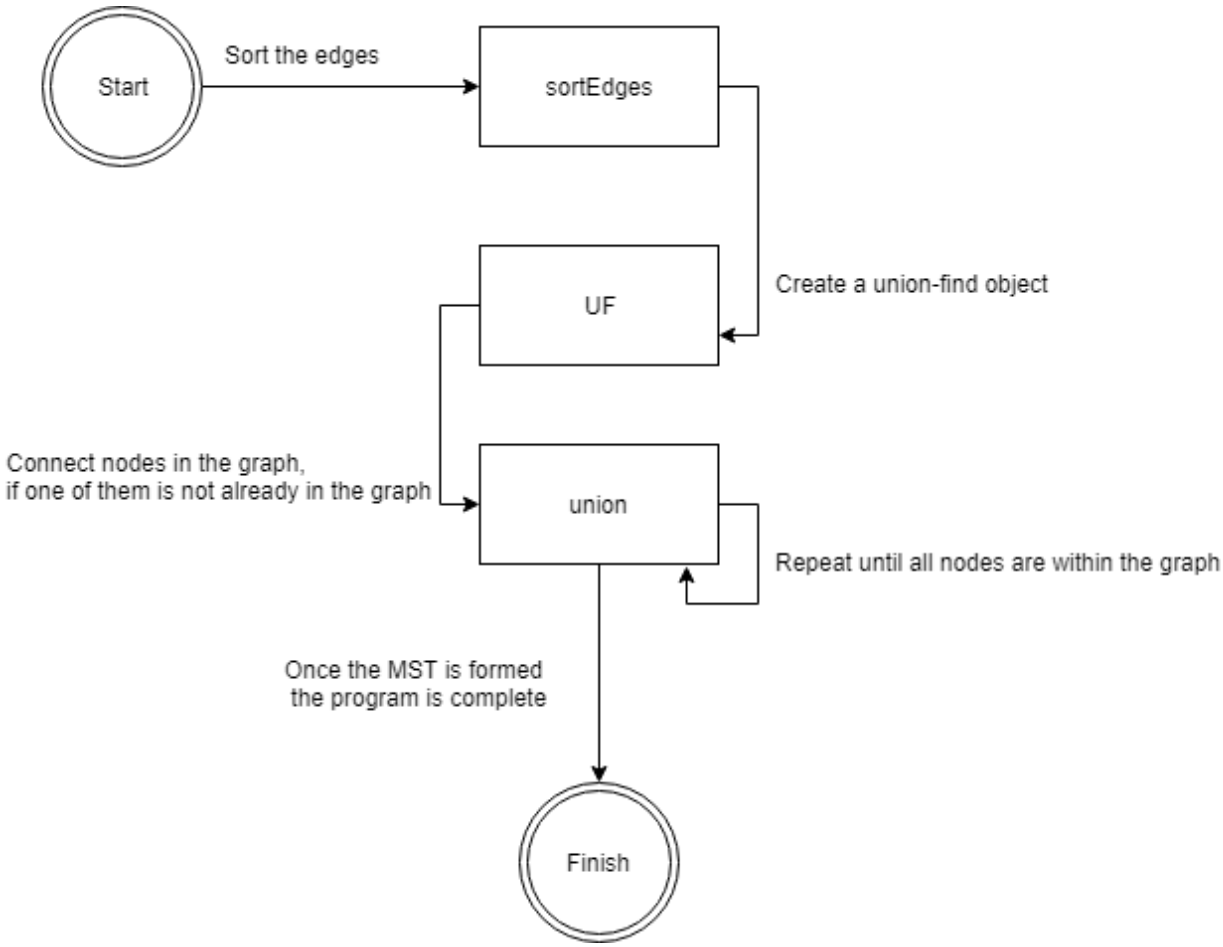
HashMap, ArrayList, Cord, Edge, Cluster, Graph, KruskalMST, KMeans, drawSurface, Load

4.3 UML State Machine Diagrams

The diagram below is the UML State Machine Diagram for KMeans.



The diagram below is the UML State Machine Diagram for KruskalMST.



5 Internal Review and Evaluation

The internal review and evaluation process addresses two important factors of any software product: Verification and Validation. We evaluate based on code testing to verify that the product is working right and review that the right product is being built by tracing back to specifications, particularly non-functional specifications, as functional specifications are covered in the modular decomposition section in more detail.

5.1 Verification

5.1.1 White Box Testing

For each module, a white box testing procedure is carried out to verify that each individual components are functioning as expected. Test cases such as boundary cases, exception cases and abnormal input are taken into consideration. Each test case will also obey module interface specification limits and assumptions: For example, a negative number will not be tested on a function that only accepts $\mathbb{N}$.

- Cord - Emphasis on constructor, accessor and mutator methods testing.
- Cluster - Emphasis on constructor, accessor and mutator methods testing.
- KMeans - Relatively black-boxed approach to testing; Only cluster size verified.
- Edge - Emphasis on constructor, accessor and mutator methods, as well as comparison of weights between edges.
- Graph - Test of properly initialized edges based on points passed into the ADT
- Heap - Test of proper helper functions, and verification of sorting.
- UF - Verify union-find methods are working as expected.
- KruskalMST - Verify that the constructed MST is optimal.
- Load - Test that Load functions, and countsort/hash tables are initiated.

5.1.2 Black Box Testing

The modules not included for white box testing in the above section, drawSurface and demoFrame, are used for black box testing. The purpose is to verify the output given a valid input into the program. Testing is mainly done on demoFrame as it is the driver of the entire program. Testing parameters taken into consideration are:

- Cluster consistency: Since K-Means is a heuristic that randomly selects start points of centroids, the goal of such a test is to check whether generated cluster hotspots are consistent on each run.
- Cluster amounts: Since K-Means allows the amount of clusters to be generated to be modified, several cluster amounts are tested to generate the most consistent and user-friendly output. Too many clusters may hamper the ability of users to consult the application for its uses, while too few clusters will provide an inaccurate representation of pickups in the city. Consistency issues are addressed by testing consistent output for the same amount of clusters.
- Program runtime: Since the application is intended to be used in real-time, the algorithm and graphical generation should not exceed a certain threshold, or it may become unusable by its users. This ties to the requirement that the application should not take more than 3 seconds to complete.

5.2 Validation

The goal of validation is to ensure a correct product, i.e. a product that satisfies requirements, is built. We evaluate based on non-functional specifications of the product, i.e. software qualities of the application, and qualify how the project has met each requirement.

- Reliability: As consistency is constantly emphasized as an important attribute, care is taken to ensure the algorithm generates similar or equivalent outputs for each run. Steps such as choosing optimal cluster size is carried out to ensure that consistency is not compromised.
- Security, Safety, Portability: Out of scope of this project. However, if the project scale were to increase, it should be of main concern that the platform of this application is secure.