**SOFE2715 Data Structure and Algorithms**

**Winter 2017**

**Group 1:** Anna Safonov 100601514

George Zakharov

Justin Kaipada

Riley Stephens

**Tutorial 7 Activity: Project Update**

For this project, our group has chosen to implement Convex Hull and K-means algorithms. Both algorithms are implemented in Python.

**Convex Hull**

*Where We Are*

The main algorithm for convex hull is complete and tested. We are working on improving time complexity for the main method.

*Algorithms and Data Structures Used*

The following algorithms and data structures are used in our implementation of the Convex Hull:

* Arrays are used to store data and as a stack implementation;
* The stack is used to detect and remove concavities along the boundary of data points in the Graham scan method.
* An insertion sort is used to order data based on slopes data points make with the lowest point in the set;
* Graham scan is used as the main algorithm to find the convex hull of a data set.

The following are Python libraries used in our implementation of Convex Hull:

* csv – CSV file reader used to parse CSV data from the test exercises provided for this project
* math – for operations with square roots in the method that determines Euclidean distance between two given points
* matplotlib.pyplot – to plot the convex hull for visual presentation
* timeit – to calculate running time of the main algorithm for time complexity analysis

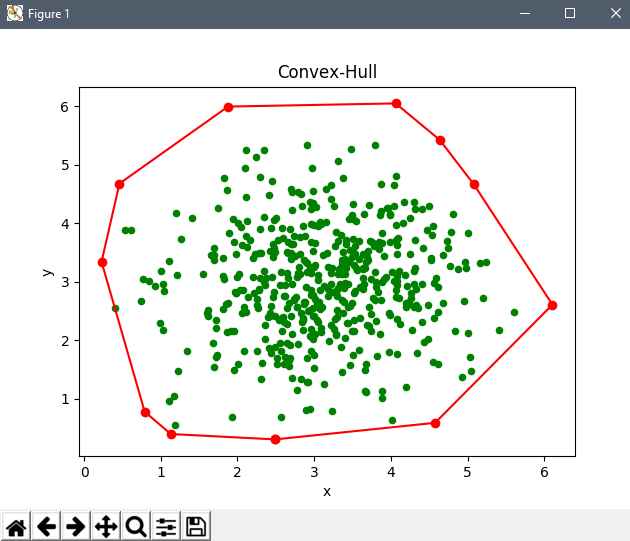
*Challenges Faced and Future Directions*

Currently, the algorithms takes progressively longer time for larger sets of data, and the bottleneck for time complexity is likely in the insertion sort implemented in the main method. An alternative approach is to consider heap sort, and implementing heap sort instead of insertion sort is our next step for this part of the project.

Subsequently, we will also work on calculating and analyzing the time complexity of the algorithm, developing performance matrix, and completing the final report for presentation and evaluation.

Results:

Running the Algorithm for Exercise 3 gives the following output



This what the Convex-Hull main method looks like now:



**K-means**

*Where We Are*

K-means algorithm is implemented and has been tested on the exercises provided. We are working on implementing better data structures to store data – i.e. trying a linked list instead of an array.

*Algorithms and Data Structures Used*

The following algorithms and data structures are used in our implementation of the K-means:

* Arrays are used to store data points and clusters;
* A list is used to store centroids;

The following are Python libraries used in our implementation of K-means:

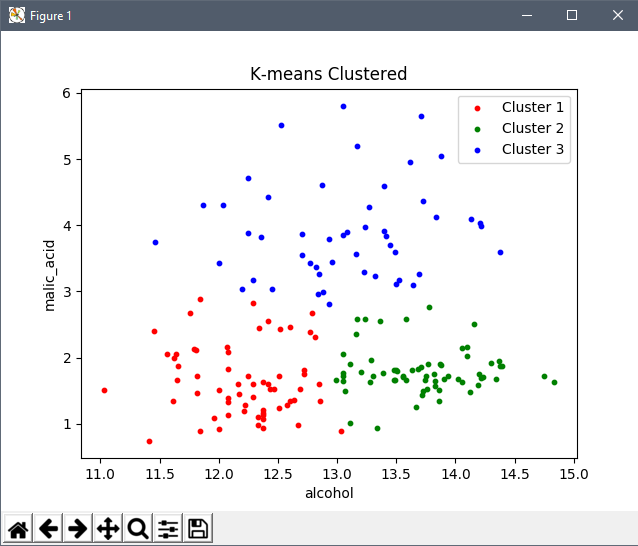
* csv – CSV file reader used to parse CSV data from the test exercises provided for this project
* math – for operations with square roots in the method that determines Euclidean distance between two given points
* matplotlib.pyplot – to plot the K-means clusters for visual presentation
* timeit – to calculate running time of the main algorithm for time complexity analysis
* random – random.choice() function is used to pick random points from the data set to be the centroids at the start of clustering

*Challenges Faced and Future Directions*

* Since we are using a random number generator somehow when executing the **K-means.py** more than 10 times consecutively we get a zero division exception, which we haven’t figured it out why.
* For the future, we would also like to implement some changes so that the number of clusters and the file to be worked with can be passed to the program from command line.

Results

Running the Algorithm for Exercise 3 gives the following output:-



This what the K-means main method looks like now:

