# Homework 3

## By Vladimir Hardy

### **Requirements:**

You are to write a program that finds the driving direction of a specified initial and end location much like MapQuest or Google Maps. You are to read the map information from a text file and create a graph. Each intersection and point of interest in the map will be given a unique name and be a vertex in the graph. Each road connecting 2 intersections is an edge in the graph. The edge names do not need to be unique. Use a directional graph since some streets may be one-ways.

The text file will look like:

| Intersection Name | Street Name | Intersection Name | Direction | Distance | Speed Limit |
|---|---|---|---|---|---|
| Alafaya&GeminiN | Gemini | Gemini&GreekParkCt | East | .3 | 35 |
| Gemini&GreekParkCt | Gemini | Alafaya&GeminiN | West | .3 | 35 |
| Gemini&GreekParkCt | Gemini | Gemini&KnightCtE | East | .5 | 35 |
| Gemini&KnightCtE | Gemini | Gemini&GreekParkCt | West | .5 | 35 |
| Gemini&KnightCtE | KnightCt | Arena | North | .1 | 20 |
| Arena | KnightCt | Gemini&KnightCtW | South | .1 | 20 |

Steps:

1. Write a function to perform a Quick Sort or Merge Sort on a list of alphanumeric data.
2. Read the text file and add each intersection name to the list. Only read the first column since all intersection names will be in there.
3. Sort the list using the Quick Sort or Merge Sort algorithm
4. Copy the names from the sorted list into the graph. Omit duplicates.
5. Create the graph by reading the original map file and adding an edge for each street. Use a binary search algorithm to find the proper vertex in the graph.
6. In the main program ask the user to input a start and end intersection.
7. Find the shortest and quickest path from the start to the end intersection. The output should indicate the name of the street and the distance.

The output should look like:

From Alafaya&GeminiN

Take Gemini East to Gemini&GreekParkCt .3

Take Gemini East to Gemini&KnightCtE .5

Take KnightCt North to Arena .1

The shortest path is the path with the least mileage while the fastest path is the path with the least time. The time is the distance multiplied by the speed in MPH.

## Description:

I began working this problem by practicing with my understanding with merge sort and quick sort to understand how the algorithms work (and which would be easier). I decided to start working with merge sort since it seemed the easiest to implement, and I already had some working code from class. This is when I noticed that arrays with C++ are different than arrays with Java, such that you cannot use methods to get the size of the array like .size() and the sizeof() method

doesn’t seem to work either. This meant that you would need to know the size of the array ahead of time in order to sort this data structure (which made me wonder whether I should switch to vectors, but by this point I was committed with arrays). The next step was to gather information from a file, and when I attempted to use my already existing function with linked lists, I noticed that the delimiter with one space would not work with this filetype, so I would need to create a new function for different file types. By this point I kind of lost hope in finishing the assignment on time, so that I could focus on other classes and exams, so I left the code as it is.

```
int test_array[] = {30, 29, 11, 38, 42, 19, 8};
mq.merge_sort(test_array,  n: 7);
```

```
30
0
8
11
19
29
38
```

Here is an array to test the merge sort, however some numbers appear out of nowhere.