Modeling Travel Times to Determine Shortest Path on UTK Campus

Kristina Wilson and Noah Dahle

2022

Contents

1 Brainstorming **3**

2 Notes **5**
2.1 May . . . 5
2.2 June . . . 5

3 Sketch of Timeline **8**

4 Labeled Map of Knoxville **9**

5 Data **10**
5.1 Google Maps Traffic Simulation . . . 10
5.2 GPS . . . 19

6 Modeling **20**
6.1 Program- June 1 . . . 20
6.2 Program- June 10 . . . 38
6.3 Modeling Traffic Conditions . . . 38
6.3.1 Green . . . 38
6.3.2 Orange . . . 38
6.3.3 Red . . . 38
6.3.4 Brown . . . 38
6.3.5 N/A . . . 38

7 Noah's Program **39**

8 Works Cited **43**

9 Latex/Python/etc Notes **44**

10 Time Log **45**
10.1 NOAH 45
10.2 KRISTINA 49

1 Brainstorming

Goal:
Find the fastest route from point A to point B.
Questions:
How does travel time change depending on the time of day or other factors?
Is it faster to walk or to drive?

Noah's Job:

- given travel times between each intersection, develop algorithm to find best route.
- use **Dijkstra's Algorithm** or **Topological Sort**

Kristina's Job:

- Using collected data, find a way to model travel times based on parameters (first parameter- time of day).

How are we going to collect the data?

1. Use Google Maps data (check periodically for travel times)
2. Could physically go and drive roads to time it, then compare with GM Data

Find a way to model the travel times based on parameters:
Basically, total travel time is distance times average speed.

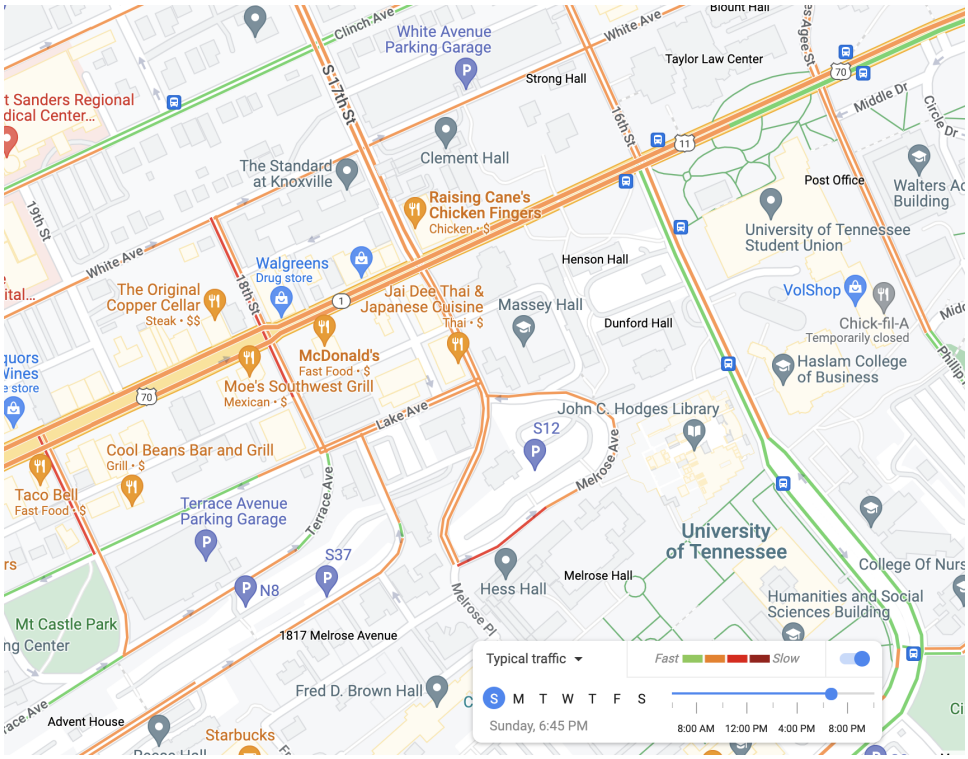
$$T = D * S$$

But adding in other factors like red lights, pedestrians, and traffic, we get something like:

$$T = D * S + R + P + t$$

These factors have different levels. Traffic can be green, yellow, orange, or red on GPS systems. Pedestrian levels change based on class change, time of day, and time of year. Time at a red light may depend on if the light system is on a timer.

Idea. Google Maps has a traffic simulation that could help us model the travel times. This is what traffic typically looks like on UT campus on Sunday at 6:45 pm.



If we design our model using this simulation, then our basic equation could look something like this:

$$\boldsymbol{T} = \boldsymbol{D} * \boldsymbol{S} + \boldsymbol{R}$$

where $\boldsymbol{T}$ is total travel time
$\boldsymbol{D}$ is the total distance
$\boldsymbol{S}$ is the speed limit
$\boldsymbol{R}$ is traffic time according to the simulation

So, now we need to figure out how to model the information given by the simulation!
We need to be able to understand traffic behavior between each node, so we should collect data for each edge.
Make a model for edges(Goal: understand the data from the simulation), then call that function within the main function to output the weights for Noah's program.

2 Notes

2.1 May

May 27- Meeting with Dr. Prosper. She suggested we create an overleaf document with all of our ideas and data. Next meeting will most likely be next Thursday at 9:00 am. Today Kristina worked on creating the project document and Noah worked on learning Python and researching the algorithm.

May 29- Today we worked on making a labeled map for our first path, which is Hodges Library to Neyland Stadium (see section Labeled Map of Knoxville). Then Kristina collected data through Google Maps, and Noah built the graph in his program and started the algorithm. We decided how Kristina should output the modeling function so Noah's algorithm can read it:

Model Output: Number of nodes. Node to, Node from, Weight, Node to, Node from, Weight, ...

The data collected from Google Maps is not very helpful because it rounds to the nearest minute. We need another method to determine travel times. We found a traffic simulator on Google Maps with adjustable days and times. This could be a way to study traffic behavior without having to constantly check the GPS throughout the day.

May 30- Today Noah finished the rough draft of the algorithm. It is mostly finished, but it needs a few touch ups. It's also subject to change in the future depending on how we import the modeled edge weights into the program.
Kristina coded the data page and collected data from the Google Maps simulation for Sunday 6:00-6:15 am.

2.2 June

June 1- Kristina collected data from the Google Maps simulation for Sunday 6:30-10:00 am in 15 minute increments and created the file for the modeling program! Then worked on a way to put the google maps' traffic information in python (learning dictionaries, input) so we can work on a function later. What the program looks like now is under the modeling section. The times will be in 24H time so they will be easy to reference. The times section will be a function so we can control the times of the data that we put input (Later). The edges function reads in a list of the nodes and edge and outputs the edge names that are used in the Data section. The edges function is versatile so we can use it with our next trial with a different set of nodes and edges.
Now we have the traffic conditions for a typical Sunday from 6:00-10:00 am saved in a list. (We also need to account for what these traffic conditions mean to travel times). Noah worked on writing his program in python instead. Due to the conflicting natures of python and c++, this was more difficult than he

thought. For now, his program will stay in c++ unless it needs to be changed to python for some reason. UPDATE: Noah figured out how to call the c++ function within python, so that is how his algorithm will be called

June 3- Kristina created a new notebook with all of the functions for the model (setting up the times list, the edges list, and the entering traffic conditions list) and also stored the data that was already entered (Sun 6-10 am). Then I corrected the formatting of the data by making a loop that looks at every entry and makes sure it is in the correct format. I will save this function and use it when we have all of the data entered (Note: Put in Functions notebook) Next, the goal is to develop a model to determine what green, orange, red means. How much time is added because of each color?

June 5- Kristina created a distance list and dictionary with each edge and its distance in feet to the next node. We will decide later which one we should use. Then created a function called timeEachNode that inputs the distance list, the traffic conditions, the time of day, and the day. This function will call the modeling functions for the green, orange, red traffic conditions, and then return a list of the edges and their weights (what Noah's program will read in). Right now I have experimental values for the traffic conditions modeling. Next Goal: Study GM data and model traffic conditions.

June 9- Meeting with Dr. Prosper. Goals/Notes:
1. Error check the traffic input function(If you enter a wrong color, output an error statement!)
2. Make the output to the timeEachNode file a txt file that Noah's program will read.
3. Noah: brainstorm a way to figure how to reduce the amount of edges we deal with.
4. save HodgesNeyland610 data and import to save time and space
5. For collaboration on jupyter notebook: check out google collab or jupyter lab real time collab (experimental)
6. Submit manuscript to EUReCA and/or SIURO? Look at other published papers to get an idea of what we should do.

Today Kristina worked on creating the function output as a text file and saving the already inputted traffic data as a txt file to import to another notebook. Work on making the output as from node, to node, weight.

June 10- Today we fixed the output of the function. Now it is in the correct format for Noah's program. Also, the input traffic conditions function now saves the data as a text file, and error checks as the user inputs the data. We used the function to enter more of the traffic data from Sunday 10:15-4:00 (saved in a file called HodgesNeyland616). I like exporting to a text file much better because if I misenter any data, then it is very easy to fix. New goal: find a way to take the data from python to a chart in this doc.

June 15- (Noah got approved today to work, so he will begin work on the project soon.) First, we tested the output of Kristi's function with Noah's program. We ran into major issues, so we spent a while fixing the function. We got my function working and printing a reasonable path and time. I started to look at the Google Maps' data. The goal for tomorrow is to look at more data from GM and brainstorm the equations for the traffic conditions. (Also, on desktop computer, import functions pdf in modeling section) Noah wrote a function (depth first search) that tests to see if a path exists before running the Dijkstra's algorithm which would just give an error if the path didn't exist.
Interesting finding. We compared our solution to G.M.'s directions and found that they had said edge 1AB was a one-way street when it is not. G.M.'s time was 3 minutes while ours was 1.5 minutes. Then we compared to Apple Maps who chose the same path as us and said it was 2 minutes! We did further research and found that the road used to be a one-way street. So we believe that G.M. had this old information and never updated the system.

June 17- Kristina worked on combining both the python and c++ programs to make one big function. Now the user will input the time, day, and the two nodes, and then the function will output the path. In doing this, we found a lot of bugs and incorrect equations in the first part of the program so we fixed those. We fixed how we defined the indices, and added functions to make the program flow better. Most importantly, Kristina commented the code! The function was working but now as we are testing it we are getting a seg fault. So we need to fix that next. We also need to figure out how to save jupyter notebooks as a pdf. It works on my desktop but not on my laptop.

3 Sketch of Timeline

May 22- May 29

Goals:

- ~~Create project document~~
- ~~Create sketch of timeline for first couple weeks~~
- ~~Create plan to collect data~~ (Google Maps Traffic Simulator)
- ~~Determine project name~~
- ~~Create labeled map of Knoxville~~
- ~~Write program to create map~~

May 30 - June 5

Goals:

- ~~Write program algorithm~~
- ~~Figure out how to call program in Python~~
- Comment program and clean up
- ~~Explore Google Maps' Traffic Simulator, determine a way to record and study the data from it.~~ (Record in table on overleaf, use traffic function on jupyter)

June 6 - June 12

Goals:

- Collect 4 days from Google Maps Simulator
- ~~Error check the traffic input~~ function
- ~~timeEachNode output as a txt file~~
- ~~save HodgesNeyland610 data and import to save time and space~~
- ~~Brainstorm equations to fit data points from traffic conditions~~ (Used placeholder equations for function)

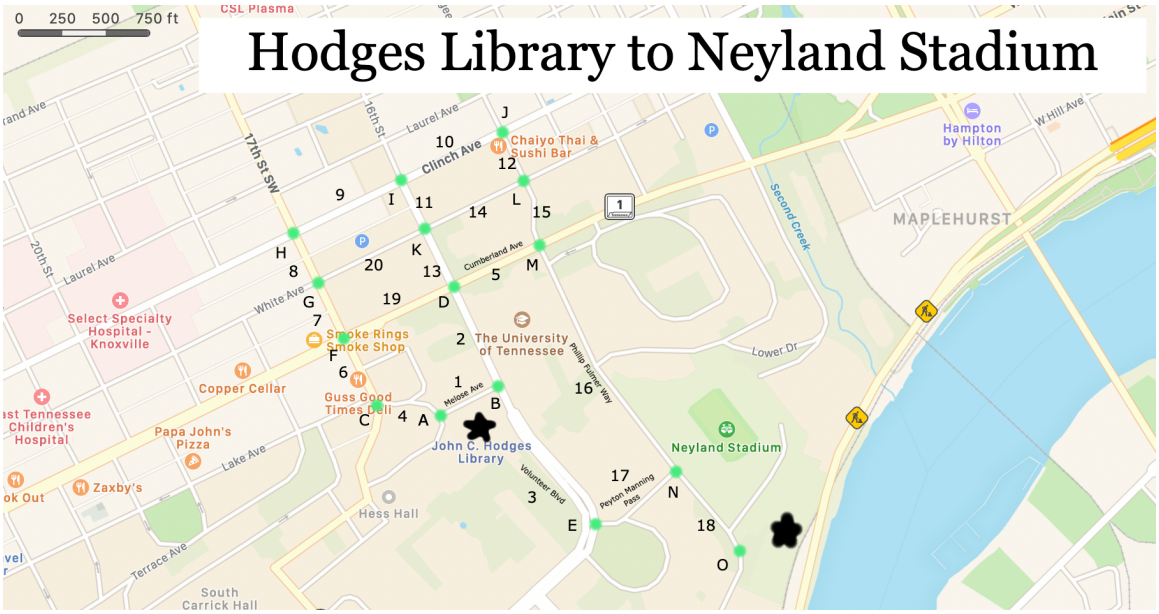
June 13 - June 19

Goals:

- ~~Feed some of these data points into Noah's algorithm and compare to what the GPS says.~~
- Noah: brainstorm a way to figure how to reduce the amount of edges we deal with in program
- G.M. data for traffic conditions equations
- Add days to the function (when we have that data)
- Insert function notebook (6/10?) and combined function from 6/17 into modeling section
- ~~Kristi: Comment your code on functions notebook.~~
- Check out google collab/jupyter lab real time collab
- G.M. data for Sun-Tue.
- Model traffic lights.
- Find a way to take the data from python to a chart in this doc.

4 Labeled Map of Knoxville

We need to figure out a system to label nodes and edges.
Starting Point: Hodges Library
End Point: Neyland Stadium



5 Data

5.1 Google Maps Traffic Simulation

Traffic Levels (increasing): Green, Orange, Red, Brown
N/A means the data is not shown on the simulation.

Sun 6:00 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Green	14K	Orange
1B	N/A	7G	Green	14L	N/A
2B	Green	8G	Green	15L	Orange
2D	Orange	8H	Green	15M	N/A
3B	Orange	9H	N/A	16M	Green
3E	Orange	9I	N/A	16N	N/A
4A	N/A	10I	N/A	17E	N/A
4C	N/A	10J	N/A	17N	N/A
5D	Green	11I	Green	18N	Green
5M	Green	11K	Orange	18O	Green
6C	Orange	12J	Orange	19D	Green
6F	Green	12L	N/A	19F	Green
		13D	Orange	20G	Orange
		13K	Green	20K	N/A

Sun 6:15 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Green	14K	Orange
1B	N/A	7G	Green	14L	N/A
2B	Green	8G	Green	15L	Orange
2D	Orange	8H	Green	15M	N/A
3B	Orange	9H	N/A	16M	Green
3E	Green	9I	Green	16N	N/A
4A	N/A	10I	N/A	17E	N/A
4C	N/A	10J	N/A	17N	N/A
5D	Green	11I	Green	18N	Green
5M	Green	11K	Orange	18O	Green
6C	Orange	12J	Orange	19D	Green
6F	Green	12L	N/A	19F	Green
		13D	Orange	20G	Orange
		13K	Green	20K	N/A

Sun 6:30 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Green	14K	Green
1B	N/A	7G	Green	14L	N/A
2B	Green	8G	Green	15L	Orange
2D	Green	8H	Green	15M	N/A
3B	Orange	9H	Green	16M	Green
3E	Green	9I	Green	16N	N/A
4A	N/A	10I	N/A	17E	N/A
4C	N/A	10J	N/A	17N	N/A
5D	Green	11I	Green	18N	Green
5M	Green	11K	Orange	18O	Green
6C	Orange	12J	Red	19D	Green
6F	Green	12L	N/A	19F	Green
		13D	Orange	20G	Green
		13K	Green	20K	N/A

Sun 6:45 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Orange	14K	Green
1B	N/A	7G	Green	14L	N/A
2B	Green	8G	Green	15L	Orange
2D	Green	8H	Green	15M	N/A
3B	Orange	9H	Green	16M	Green
3E	Green	9I	Green	16N	N/A
4A	N/A	10I	Green	17E	N/A
4C	N/A	10J	N/A	17N	N/A
5D	Green	11I	Green	18N	Green
5M	Green	11K	Orange	18O	Green
6C	Orange	12J	Red	19D	Green
6F	Green	12L	N/A	19F	Green
		13D	Orange	20G	Green
		13K	Green	20K	N/A

Sun 7:00 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Orange	14K	N/A
1B	N/A	7G	Green	14L	N/A
2B	Green	8G	Green	15L	Orange
2D	Red	8H	Green	15M	Green
3B	Red	9H	Green	16M	Orange
3E	Green	9I	Green	16N	Green
4A	N/A	10I	Green	17E	N/A
4C	N/A	10J	Green	17N	N/A
5D	Green	11I	Green	18N	Red
5M	Green	11K	Green	18O	N/A
6C	Orange	12J	Red	19D	Green
6F	Green	12L	N/A	19F	Green
		13D	Green	20G	Green
		13K	Green	20K	N/A

Sun 7:15 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Green	14K	N/A
1B	N/A	7G	Green	14L	N/A
2B	Green	8G	Green	15L	Orange
2D	Red	8H	Green	15M	Green
3B	Orange	9H	Green	16M	Orange
3E	Green	9I	Green	16N	Green
4A	N/A	10I	Green	17E	N/A
4C	N/A	10J	Green	17N	N/A
5D	Green	11I	Green	18N	Red
5M	Green	11K	Green	18O	N/A
6C	Orange	12J	Red	19D	Green
6F	Orange	12L	N/A	19F	Green
		13D	Green	20G	Green
		13K	Green	20K	N/A

Sun 7:30 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Orange	14K	N/A
1B	N/A	7G	Green	14L	N/A
2B	Green	8G	Green	15L	Orange
2D	Green	8H	Green	15M	Red
3B	Green	9H	Green	16M	Orange
3E	Green	9I	Green	16N	Green
4A	N/A	10I	N/A	17E	N/A
4C	N/A	10J	Green	17N	Green
5D	Green	11I	Green	18N	Red
5M	Green	11K	Green	18O	Green
6C	Orange	12J	Orange	19D	Green
6F	Green	12L	N/A	19F	Green
		13D	Green	20G	Green
		13K	Green	20K	N/A

Sun 7:45 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Orange	14K	N/A
1B	N/A	7G	Green	14L	N/A
2B	Green	8G	Green	15L	Red
2D	Green	8H	Green	15M	Red
3B	Green	9H	Green	16M	Red
3E	Green	9I	Green	16N	Orange
4A	N/A	10I	N/A	17E	N/A
4C	N/A	10J	Green	17N	Green
5D	Green	11I	Green	18N	Red
5M	Green	11K	Green	18O	Green
6C	Orange	12J	Orange	19D	Green
6F	Green	12L	N/A	19F	Green
		13D	Green	20G	N/A
		13K	Green	20K	N/A

Sun 8:00 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Orange	14K	Green
1B	N/A	7G	Orange	14L	N/A
2B	Orange	8G	Green	15L	Red
2D	Green	8H	Green	15M	Green
3B	Green	9H	Green	16M	Red
3E	Green	9I	Orange	16N	Green
4A	N/A	10I	Green	17E	Green
4C	N/A	10J	Green	17N	Green
5D	Green	11I	Green	18N	Red
5M	Green	11K	Orange	18O	Green
6C	Orange	12J	Red	19D	Green
6F	Green	12L	N/A	19F	Green
		13D	Orange	20G	Green
		13K	Green	20K	N/A

Sun 8:15 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Green	14K	Green
1B	N/A	7G	Orange	14L	N/A
2B	Orange	8G	Green	15L	Red
2D	Green	8H	Green	15M	Green
3B	Green	9H	Green	16M	Red
3E	Orange	9I	Orange	16N	Green
4A	N/A	10I	Green	17E	Green
4C	N/A	10J	Green	17N	Green
5D	Green	11I	Green	18N	Red
5M	Green	11K	Orange	18O	Green
6C	Orange	12J	Red	19D	Green
6F	Green	12L	N/A	19F	Green
		13D	Orange	20G	Green
		13K	Green	20K	N/A

Sun 8:30 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Orange	14K	Green
1B	N/A	7G	Green	14L	N/A
2B	Orange	8G	Green	15L	Green
2D	Green	8H	Green	15M	Green
3B	Green	9H	Orange	16M	Green
3E	Orange	9I	Green	16N	Green
4A	Orange	10I	Orange	17E	N/A
4C	N/A	10J	Green	17N	Green
5D	Green	11I	Green	18N	Orange
5M	Green	11K	Orange	18O	Green
6C	Red	12J	Orange	19D	Green
6F	Green	12L	Green	19F	Green
		13D	Orange	20G	Green
		13K	Green	20K	N/A

Sun 8:45 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Green	14K	Green
1B	N/A	7G	Green	14L	N/A
2B	Red	8G	Green	15L	Green
2D	Green	8H	Green	15M	Green
3B	Green	9H	Orange	16M	Green
3E	Red	9I	Green	16N	Green
4A	Orange	10I	Orange	17E	N/A
4C	N/A	10J	Green	17N	Green
5D	Green	11I	Green	18N	Orange
5M	Green	11K	Orange	18O	Green
6C	Red	12J	Orange	19D	Green
6F	Green	12L	Green	19F	Green
		13D	Orange	20G	Green
		13K	Green	20K	N/A

Sun 9:00 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Orange	14K	Orange
1B	N/A	7G	Green	14L	N/A
2B	Orange	8G	Green	15L	Green
2D	Green	8H	Green	15M	Green
3B	Orange	9H	Orange	16M	Green
3E	Green	9I	Green	16N	Green
4A	Orange	10I	Orange	17E	N/A
4C	N/A	10J	Green	17N	Green
5D	Green	11I	Green	18N	Orange
5M	Green	11K	Orange	18O	Green
6C	Orange	12J	Orange	19D	Green
6F	Green	12L	Green	19F	Green
		13D	Orange	20G	Orange
		13K	Green	20K	N/A

Sun 9:15 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Orange	14K	Orange
1B	N/A	7G	Green	14L	N/A
2B	Orange	8G	Green	15L	Green
2D	Green	8H	Green	15M	Green
3B	Orange	9H	Orange	16M	Green
3E	Green	9I	Green	16N	Green
4A	Orange	10I	Orange	17E	N/A
4C	N/A	10J	Green	17N	Green
5D	Green	11I	Orange	18N	Green
5M	Green	11K	Green	18O	Green
6C	Orange	12J	Orange	19D	Green
6F	Green	12L	Green	19F	Green
		13D	Green	20G	Orange
		13K	Orange	20K	N/A

Sun 9:30 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Orange	14K	Orange
1B	N/A	7G	Orange	14L	N/A
2B	Orange	8G	Green	15L	Orange
2D	Green	8H	Green	15M	Green
3B	Orange	9H	Orange	16M	Green
3E	Orange	9I	Green	16N	Orange
4A	Orange	10I	Orange	17E	Orange
4C	N/A	10J	Green	17N	Orange
5D	Green	11I	Green	18N	Green
5M	Green	11K	Orange	18O	Green
6C	Orange	12J	Orange	19D	Green
6F	Green	12L	Green	19F	Green
		13D	Orange	20G	Orange
		13K	Green	20K	N/A

Sun 9:45 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Orange	14K	Orange
1B	N/A	7G	Orange	14L	N/A
2B	Orange	8G	Green	15L	Orange
2D	Green	8H	Orange	15M	Green
3B	Orange	9H	Orange	16M	Green
3E	Orange	9I	Green	16N	Orange
4A	Orange	10I	Orange	17E	Orange
4C	N/A	10J	Green	17N	Orange
5D	Green	11I	Green	18N	Green
5M	Green	11K	Orange	18O	Green
6C	Orange	12J	Orange	19D	Green
6F	Green	12L	Green	19F	Green
		13D	Orange	20G	Orange
		13K	Green	20K	N/A

Sun 10:00 AM

Edge	Traffic Level	Edge	Traffic Level	Edge	Traffic Level
1A	N/A	7F	Orange	14K	Orange
1B	N/A	7G	Orange	14L	N/A
2B	Orange	8G	Green	15L	Orange
2D	Green	8H	Green	15M	Green
3B	Green	9H	Green	16M	Green
3E	Orange	9I	Green	16N	Orange
4A	Orange	10I	Green	17E	Orange
4C	N/A	10J	Green	17N	Orange
5D	Green	11I	Green	18N	Green
5M	Green	11K	Orange	18O	Green
6C	Orange	12J	Orange	19D	Green
6F	Green	12L	Green	19F	Green
		13D	Orange	20G	Orange
		13K	Green	20K	N/A

5.2 GPS

Date: May 31 2022
Hodges to Neyland Stadium
Google Maps

Time of Day	Travel Time (min)	Distance	Edge
5:45 pm	1	377 feet	1A
	1		1B
	1	0.1 mi	2B
	1		2D
	1	0.2 mi	3B
	1		3E
	1	367 ft	4A
	1	0.1 mi	5D
	1		5M
	1	430 ft	6C
	1		6F
	1	400 ft	7F
	1		7G
	1	302 feet	8G
	1		8H
	1	0.1 mi	9H
	1		9I
	1	0.1 mi	10I
	1		10J
	1	300 ft	11I
	1		11K
	1	318 ft	12J
	1		12L
	1	350 ft	13K
	1		13D
	1	0.1 mi	14K
	1	400 ft	15L
	1		15M
	2	0.4 mi	16M
	2		16N
	1	0.1 mi	17E
	1		17N
	1	0.1 mi	18N
	1		18O
	1	0.1 mi	19D
	1		19F
	1	0.1 mi	20K

6 Modeling

Output format:
NUM (number of nodes)
(if it is a two way) A B (weight) B A (weight)
(if it is a one way) A B (from to) (weight)

6.1 Program- June 1

I am working on developing the actual equation, but I have a way to get the GM data into a list in python, so the program is able to read it (today I called it HodgesNeyland)!
This function asks for the condition at each edge at each time, then adds it to the list.

Modeling Travel Times to Determine Shortest Path - in Doc

June 1, 2022

Kristina Wilson and Noah Dahle

Modeling Travel Times to Determine Shortest Path

Experimenting with a potential model.

Implementing functions that have stored in the google maps traffic simulation data.

This is a model based on google maps traffic conditions. (AKA T = D*S + R)

Hopefully I can develop one with more variables later on!

[1]:
```
import math
```

Goal: make a list defining the times and nodes, then create a list with all info. (Times in 24HR)

0.1 Times List

[36]:
```
# having a for loop will be more helpful when we have a whole day's worth of␣
 ↪data.
# Or if we want to change how much data we are putting in our function.
times = []
val = 600
for i in range(0, 17):
    if (val % 100) == 60:
        val += 40
    times.append(val)
    val += 15
times
```

[36]:
```
[600,
 615,
 630,
 645,
 700,
 715,
 730,
 745,
 800,
 815,
 830,
```

```
 845,
 900,
 915,
 930,
 945,
 1000]
```

We need to account for one way roads in our edge list. So those would not need to be in the list. (Later)

0.2 Edges List

[8]:
```
input1 = [('1', 'A', 'B'), ('2', 'B', 'D'), ('3', 'B', 'E'), ('4', 'A', 'C'),␣
↪('5', 'D', 'M'), ('6', 'C', 'F'), ('7', 'F', 'G'), ('8', 'G', 'H'), ('9',␣
↪'H', 'I'), ('10', 'I', 'J'), ('11', 'I', 'K'), ('12', 'J', 'L'), ('13', 'D',␣
↪'K'), ('14', 'K', 'L'), ('15', 'L', 'M'), ('16', 'M', 'N'), ('17', 'E',␣
↪'N'), ('18', 'N', 'O'), ('19', 'D', 'F'), ('20', 'G', 'K')]
```

[9]:
```
def edgeslist(input):
    output=[]
    length = len(input)
    for i in range(0, 2*length):
        output.append(input[math.floor(i/2)][0]+input[math.floor(i/2)][(i%2) +␣
↪1])
    return output
```

[20]:
```
edges = edgeslist(input1)
```

[31]:
```
def traffic(times, edges):
    output = []
    for j in range(len(times)):
        print('You are entering conditions for', times[j])
        for i in range(len(edges)):
            condition = input("Enter the traffic condition for edge "+␣
↪edges[i]+ ": ")
            element = [times[j], edges[i], condition]
            output.append(element)
    return output
```

[60]:
```
HodgesNeyland
```

[60]:
```
[[600, '1A', 'NA'],
 [600, '1B', 'NA'],
 [600, '2B', 'GREEN'],
 [600, '2D', 'ORANGE'],
 [600, '3B', 'ORANGE'],
 [600, '3E', 'ORANGE'],
 [600, '4A', 'NA'],
```

```
[600, '4C', 'NA'],
[600, '5D', 'GREEN'],
[600, '5M', 'GREEN'],
[600, '6C', 'ORANGE'],
[600, '6F', 'GREEN'],
[600, '7F', 'GREEN'],
[600, '7G', 'GREEN'],
[600, '8G', 'GREEN'],
[600, '8H', 'GREEN'],
[600, '9H', 'NA'],
[600, '9I', 'NA'],
[600, '10I', 'NA'],
[600, '10J', 'NA'],
[600, '11I', 'GREEN'],
[600, '11K', 'ORANGE'],
[600, '12J', 'ORANGE'],
[600, '12L', 'NA'],
[600, '13D', 'ORANGE'],
[600, '13K', 'GREEN'],
[600, '14K', 'ORANGE'],
[600, '14L', 'NA'],
[600, '15L', 'ORANGE'],
[600, '15M', 'NA'],
[600, '16M', 'GREEN'],
[600, '16N', 'NA'],
[600, '17E', 'NA'],
[600, '17N', 'NA'],
[600, '18N', 'GREEN'],
[600, '18O', 'GREEN'],
[600, '19D', 'GREEN'],
[600, '19F', 'GREEN'],
[600, '20G', 'ORANGE'],
[600, '20K', 'NA'],
[615, '1A', 'NA'],
[615, '1B', 'NA'],
[615, '2B', 'GREEN'],
[615, '2D', 'ORANGE'],
[615, '3B', 'ORANGE'],
[615, '3E', 'GREEN'],
[615, '4A', 'NA'],
[615, '4C', 'NA'],
[615, '5D', 'GREEN'],
[615, '5M', 'GREEN'],
[615, '6C', 'ORANGE'],
[615, '6F', 'GREEN'],
[615, '7F', 'GREEN'],
[615, '7G', 'GREEN'],
```

```
[615, '8G', 'GREEN'],
[615, '8H', 'GREEN'],
[615, '9H', 'NA'],
[615, '9I', 'GREEN'],
[615, '10I', 'NA'],
[615, '10J', 'NA'],
[615, '11I', 'GREEN'],
[615, '11K', 'ORANGE'],
[615, '12J', 'ORANGE'],
[615, '12L', 'NA'],
[615, '13D', 'ORANGE'],
[615, '13K', 'GREEN'],
[615, '14K', 'ORANGE'],
[615, '14L', 'NA'],
[615, '15L', 'ORANGE'],
[615, '15M', 'NA'],
[615, '16M', 'GREEN'],
[615, '16N', 'NA'],
[615, '17E', 'NA'],
[615, '17N', 'NA'],
[615, '18N', 'GREEN'],
[615, '18O', 'GREEN'],
[615, '19D', 'GREEN'],
[615, '19F', 'GREEN'],
[615, '20G', 'ORANGE'],
[615, '20K', 'NA'],
[630, '1A', 'NA'],
[630, '1B', 'NA'],
[630, '2B', 'GREEN'],
[630, '2D', 'GREEN'],
[630, '3B', 'ORANGE'],
[630, '3E', 'GREEN'],
[630, '4A', 'NA'],
[630, '4C', 'NA'],
[630, '5D', 'GREEN'],
[630, '5M', 'GREEN'],
[630, '6C', 'ORANGE'],
[630, '6F', 'GREEN'],
[630, '7F', 'GREEN'],
[630, '7G', 'GREEN'],
[630, '8G', 'GREEN'],
[630, '8H', 'GREEN'],
[630, '9H', 'GREEN'],
[630, '9I', 'G'],
[630, '10I', 'NA '],
[630, '10J', 'NA'],
[630, '11I', 'GREEN'],
```

```
[630, '11K', 'ORANGE'],
[630, '12J', 'RED'],
[630, '12L', 'NA'],
[630, '13D', 'ORANGE'],
[630, '13K', 'GREEN'],
[630, '14K', 'GREEN'],
[630, '14L', 'NA'],
[630, '15L', 'ORANGE'],
[630, '15M', 'NA'],
[630, '16M', 'GREEN'],
[630, '16N', 'NA'],
[630, '17E', 'NA'],
[630, '17N', 'NA'],
[630, '18N', 'GREEN'],
[630, '18O', 'GREEN'],
[630, '19D', 'GREEN'],
[630, '19F', 'GREEN'],
[630, '20G', 'GREEN'],
[630, '20K', 'NA'],
[645, '1A', 'NA'],
[645, '1B', 'NA'],
[645, '2B', 'GREEN'],
[645, '2D', 'GREEN'],
[645, '3B', 'ORANGE'],
[645, '3E', 'GREEN'],
[645, '4A', 'NA'],
[645, '4C', 'NA'],
[645, '5D', 'GREEN'],
[645, '5M', 'GREEN'],
[645, '6C', 'ORANGE'],
[645, '6F', 'GREEN'],
[645, '7F', 'ORANGE'],
[645, '7G', 'GREEN'],
[645, '8G', 'GREEN'],
[645, '8H', 'GREEN'],
[645, '9H', 'GREEN'],
[645, '9I', 'GREEN'],
[645, '10I', 'GREEN'],
[645, '10J', 'NA'],
[645, '11I', 'GREEN'],
[645, '11K', 'ORANGE'],
[645, '12J', 'RED'],
[645, '12L', 'NA'],
[645, '13D', 'ORANGE'],
[645, '13K', 'GREEN'],
[645, '14K', 'GREEN'],
[645, '14L', 'NA'],
```

```
[645, '15L', 'ORANGE'],
[645, '15M', 'NA'],
[645, '16M', 'GREEN'],
[645, '16N', 'NA'],
[645, '17E', 'NA'],
[645, '17N', 'NA'],
[645, '18N', 'GREEN'],
[645, '18O', 'GREEN'],
[645, '19D', 'GREEN'],
[645, '19F', 'GREEN'],
[645, '20G', 'GREEN'],
[645, '20K', 'NA'],
[700, '1A', 'NA'],
[700, '1B', 'NA'],
[700, '2B', 'GREEN'],
[700, '2D', 'RED'],
[700, '3B', 'RED'],
[700, '3E', 'GREEN'],
[700, '4A', 'NA'],
[700, '4C', 'NA'],
[700, '5D', 'GREEN'],
[700, '5M', 'GREEN'],
[700, '6C', 'ORANGE'],
[700, '6F', 'G'],
[700, '7F', 'O'],
[700, '7G', 'G'],
[700, '8G', 'G'],
[700, '8H', 'G'],
[700, '9H', 'G'],
[700, '9I', 'G'],
[700, '10I', 'G'],
[700, '10J', 'G'],
[700, '11I', 'G'],
[700, '11K', 'G'],
[700, '12J', 'R'],
[700, '12L', 'NA'],
[700, '13D', 'G'],
[700, '13K', 'G'],
[700, '14K', 'NA'],
[700, '14L', 'NA'],
[700, '15L', 'O'],
[700, '15M', 'G'],
[700, '16M', 'O'],
[700, '16N', 'G'],
[700, '17E', 'NA'],
[700, '17N', 'NA'],
[700, '18N', 'R'],
```

```
[700, '18O', 'NA'],
[700, '19D', 'G'],
[700, '19F', 'G'],
[700, '20G', 'G'],
[700, '20K', 'NA'],
[715, '1A', 'NA'],
[715, '1B', 'NA'],
[715, '2B', 'G'],
[715, '2D', 'R'],
[715, '3B', 'O'],
[715, '3E', 'G'],
[715, '4A', 'NA'],
[715, '4C', 'NA'],
[715, '5D', 'G'],
[715, '5M', 'G'],
[715, '6C', 'O'],
[715, '6F', 'O'],
[715, '7F', 'G'],
[715, '7G', 'G'],
[715, '8G', 'G'],
[715, '8H', 'G'],
[715, '9H', 'G'],
[715, '9I', 'G'],
[715, '10I', 'G'],
[715, '10J', 'G'],
[715, '11I', 'G'],
[715, '11K', 'G'],
[715, '12J', 'R'],
[715, '12L', 'NA'],
[715, '13D', 'G'],
[715, '13K', 'G'],
[715, '14K', 'NA'],
[715, '14L', 'NA'],
[715, '15L', 'O'],
[715, '15M', 'G'],
[715, '16M', 'O'],
[715, '16N', 'G'],
[715, '17E', 'NA'],
[715, '17N', 'NA'],
[715, '18N', 'R'],
[715, '18O', 'NA'],
[715, '19D', 'G'],
[715, '19F', 'G'],
[715, '20G', 'G'],
[715, '20K', 'NA'],
[730, '1A', 'NA'],
[730, '1B', 'NA'],
```

```
[730, '2B', 'G'],
[730, '2D', 'G'],
[730, '3B', 'G'],
[730, '3E', 'G'],
[730, '4A', 'NA'],
[730, '4C', 'NA'],
[730, '5D', 'G'],
[730, '5M', 'G'],
[730, '6C', 'O'],
[730, '6F', 'G'],
[730, '7F', 'O'],
[730, '7G', 'G'],
[730, '8G', 'G'],
[730, '8H', 'G'],
[730, '9H', 'G'],
[730, '9I', 'G'],
[730, '10I', 'NA'],
[730, '10J', 'G'],
[730, '11I', 'G'],
[730, '11K', 'G'],
[730, '12J', 'O'],
[730, '12L', 'NA'],
[730, '13D', 'G'],
[730, '13K', 'G'],
[730, '14K', 'NA'],
[730, '14L', 'NA'],
[730, '15L', 'O'],
[730, '15M', 'R'],
[730, '16M', 'O'],
[730, '16N', 'G'],
[730, '17E', 'NA'],
[730, '17N', 'G'],
[730, '18N', 'R'],
[730, '18O', 'G'],
[730, '19D', 'G'],
[730, '19F', 'G'],
[730, '20G', 'G'],
[730, '20K', 'NA'],
[745, '1A', 'NA'],
[745, '1B', 'NA'],
[745, '2B', 'GREEN'],
[745, '2D', 'GREEN'],
[745, '3B', 'GREEN'],
[745, '3E', 'GREEN'],
[745, '4A', 'NA'],
[745, '4C', 'NA'],
[745, '5D', 'GREEN'],
```

```
[745, '5M', 'GREEN'],
[745, '6C', 'O'],
[745, '6F', 'G'],
[745, '7F', 'O'],
[745, '7G', 'G'],
[745, '8G', 'G'],
[745, '8H', 'G'],
[745, '9H', 'G'],
[745, '9I', 'G'],
[745, '10I', 'NA'],
[745, '10J', 'G'],
[745, '11I', 'G'],
[745, '11K', 'G'],
[745, '12J', 'O'],
[745, '12L', 'NA'],
[745, '13D', 'G'],
[745, '13K', 'G'],
[745, '14K', 'NA'],
[745, '14L', 'NA'],
[745, '15L', 'R'],
[745, '15M', 'R'],
[745, '16M', 'R'],
[745, '16N', 'O'],
[745, '17E', 'NA'],
[745, '17N', 'G'],
[745, '18N', 'R'],
[745, '18O', 'G'],
[745, '19D', 'G'],
[745, '19F', 'G'],
[745, '20G', 'NA'],
[745, '20K', 'NA'],
[800, '1A', 'NA'],
[800, '1B', 'NA'],
[800, '2B', 'O'],
[800, '2D', 'G'],
[800, '3B', 'G'],
[800, '3E', 'G'],
[800, '4A', 'NA'],
[800, '4C', 'NA'],
[800, '5D', 'G'],
[800, '5M', 'G'],
[800, '6C', 'O'],
[800, '6F', 'G'],
[800, '7F', 'O'],
[800, '7G', 'O'],
[800, '8G', 'G'],
[800, '8H', 'G'],
```

```
[800, '9H', 'G'],
[800, '9I', 'O'],
[800, '10I', 'G'],
[800, '10J', 'G'],
[800, '11I', 'G'],
[800, '11K', 'O'],
[800, '12J', 'R'],
[800, '12L', 'NA'],
[800, '13D', 'O'],
[800, '13K', 'G'],
[800, '14K', 'G'],
[800, '14L', 'NA'],
[800, '15L', 'R'],
[800, '15M', 'G'],
[800, '16M', 'R'],
[800, '16N', 'G'],
[800, '17E', 'G'],
[800, '17N', 'G'],
[800, '18N', 'R'],
[800, '18O', 'G'],
[800, '19D', 'G'],
[800, '19F', 'G'],
[800, '20G', 'G'],
[800, '20K', 'NA'],
[815, '1A', 'NA'],
[815, '1B', 'NA'],
[815, '2B', 'O'],
[815, '2D', 'G'],
[815, '3B', 'G'],
[815, '3E', 'O'],
[815, '4A', 'NA'],
[815, '4C', 'NA'],
[815, '5D', 'G'],
[815, '5M', 'G'],
[815, '6C', 'O'],
[815, '6F', 'G'],
[815, '7F', 'G'],
[815, '7G', 'O'],
[815, '8G', 'G'],
[815, '8H', 'G'],
[815, '9H', 'G'],
[815, '9I', 'O'],
[815, '10I', 'G'],
[815, '10J', 'G'],
[815, '11I', 'G'],
[815, '11K', 'O'],
[815, '12J', 'R'],
```

```
[815, '12L', 'NA'],
[815, '13D', 'O'],
[815, '13K', 'G'],
[815, '14K', 'G'],
[815, '14L', 'NA'],
[815, '15L', 'R'],
[815, '15M', 'G'],
[815, '16M', 'R'],
[815, '16N', 'G'],
[815, '17E', 'G'],
[815, '17N', 'G'],
[815, '18N', 'R'],
[815, '18O', 'G'],
[815, '19D', 'G'],
[815, '19F', 'G'],
[815, '20G', 'G'],
[815, '20K', 'NA'],
[830, '1A', 'NA'],
[830, '1B', 'NA'],
[830, '2B', 'O'],
[830, '2D', 'G'],
[830, '3B', 'G'],
[830, '3E', 'O'],
[830, '4A', 'O'],
[830, '4C', 'NA'],
[830, '5D', 'G'],
[830, '5M', 'G'],
[830, '6C', 'R'],
[830, '6F', 'G'],
[830, '7F', 'O'],
[830, '7G', 'G'],
[830, '8G', 'G'],
[830, '8H', 'G'],
[830, '9H', 'O'],
[830, '9I', 'G'],
[830, '10I', 'O'],
[830, '10J', 'G'],
[830, '11I', 'G'],
[830, '11K', 'O'],
[830, '12J', 'O'],
[830, '12L', 'G'],
[830, '13D', 'O'],
[830, '13K', 'G'],
[830, '14K', 'G'],
[830, '14L', 'NA'],
[830, '15L', 'G'],
[830, '15M', 'G'],
```

```
[830, '16M', 'G'],
[830, '16N', 'G'],
[830, '17E', 'NA'],
[830, '17N', 'G'],
[830, '18N', 'O'],
[830, '18O', 'G'],
[830, '19D', 'G'],
[830, '19F', 'G'],
[830, '20G', 'G'],
[830, '20K', 'NA'],
[845, '1A', 'NA'],
[845, '1B', 'NA'],
[845, '2B', 'R'],
[845, '2D', 'G'],
[845, '3B', 'G'],
[845, '3E', 'R'],
[845, '4A', 'O'],
[845, '4C', 'NA'],
[845, '5D', 'G'],
[845, '5M', 'G'],
[845, '6C', 'R'],
[845, '6F', 'G'],
[845, '7F', 'G'],
[845, '7G', 'G'],
[845, '8G', 'G'],
[845, '8H', 'G'],
[845, '9H', 'O'],
[845, '9I', 'G'],
[845, '10I', 'O'],
[845, '10J', 'G'],
[845, '11I', 'G'],
[845, '11K', 'O'],
[845, '12J', 'O'],
[845, '12L', 'G'],
[845, '13D', 'O'],
[845, '13K', 'G'],
[845, '14K', 'G'],
[845, '14L', 'NA'],
[845, '15L', 'G'],
[845, '15M', 'G'],
[845, '16M', 'G'],
[845, '16N', 'G'],
[845, '17E', 'NA'],
[845, '17N', 'G'],
[845, '18N', 'O'],
[845, '18O', 'G'],
[845, '19D', 'G'],
```

```
[845, '19F', 'G'],
[845, '20G', 'G'],
[845, '20K', 'NA'],
[900, '1A', 'NA'],
[900, '1B', 'NA'],
[900, '2B', 'O'],
[900, '2D', 'G'],
[900, '3B', 'O'],
[900, '3E', 'G'],
[900, '4A', 'O'],
[900, '4C', 'NA'],
[900, '5D', 'G'],
[900, '5M', 'G'],
[900, '6C', 'O'],
[900, '6F', 'G'],
[900, '7F', 'O'],
[900, '7G', 'G'],
[900, '8G', 'G'],
[900, '8H', 'G'],
[900, '9H', 'O'],
[900, '9I', 'G'],
[900, '10I', 'O'],
[900, '10J', 'G'],
[900, '11I', 'G'],
[900, '11K', 'O'],
[900, '12J', 'O'],
[900, '12L', 'G'],
[900, '13D', 'O'],
[900, '13K', 'G'],
[900, '14K', 'O'],
[900, '14L', 'NA'],
[900, '15L', 'G'],
[900, '15M', 'G'],
[900, '16M', 'G'],
[900, '16N', 'G'],
[900, '17E', 'NA'],
[900, '17N', 'G'],
[900, '18N', 'O'],
[900, '18O', 'G'],
[900, '19D', 'G'],
[900, '19F', 'G'],
[900, '20G', 'O'],
[900, '20K', 'NA'],
[915, '1A', 'NA'],
[915, '1B', 'NA'],
[915, '2B', 'O'],
[915, '2D', 'G'],
```

```
[915, '3B', 'O'],
[915, '3E', 'G'],
[915, '4A', 'O'],
[915, '4C', 'NA'],
[915, '5D', 'G'],
[915, '5M', 'G'],
[915, '6C', 'O'],
[915, '6F', 'G'],
[915, '7F', 'O'],
[915, '7G', 'G'],
[915, '8G', 'G'],
[915, '8H', 'G'],
[915, '9H', 'O'],
[915, '9I', 'G'],
[915, '10I', 'O'],
[915, '10J', 'G'],
[915, '11I', 'O'],
[915, '11K', 'G'],
[915, '12J', 'O'],
[915, '12L', 'G'],
[915, '13D', 'G'],
[915, '13K', 'O'],
[915, '14K', 'O'],
[915, '14L', 'NA'],
[915, '15L', 'G'],
[915, '15M', 'G'],
[915, '16M', 'G'],
[915, '16N', 'G'],
[915, '17E', 'NA'],
[915, '17N', 'G'],
[915, '18N', 'G'],
[915, '18O', 'G'],
[915, '19D', 'G'],
[915, '19F', 'G'],
[915, '20G', 'O'],
[915, '20K', 'NA'],
[930, '1A', 'NA'],
[930, '1B', 'NA'],
[930, '2B', 'O'],
[930, '2D', 'G'],
[930, '3B', 'O'],
[930, '3E', 'O'],
[930, '4A', 'O'],
[930, '4C', 'NA'],
[930, '5D', 'G'],
[930, '5M', 'G'],
[930, '6C', 'O'],
```

```
[930, '6F', 'G'],
[930, '7F', 'O'],
[930, '7G', 'O'],
[930, '8G', 'G'],
[930, '8H', 'G'],
[930, '9H', 'O'],
[930, '9I', 'G'],
[930, '10I', 'O'],
[930, '10J', 'G'],
[930, '11I', 'G'],
[930, '11K', 'O'],
[930, '12J', 'O'],
[930, '12L', 'G'],
[930, '13D', 'O'],
[930, '13K', 'G'],
[930, '14K', 'O'],
[930, '14L', 'NA'],
[930, '15L', 'O'],
[930, '15M', 'G'],
[930, '16M', 'G'],
[930, '16N', 'O'],
[930, '17E', 'O'],
[930, '17N', 'O'],
[930, '18N', 'G'],
[930, '18O', 'G'],
[930, '19D', 'G'],
[930, '19F', 'G'],
[930, '20G', 'O'],
[930, '20K', 'NA'],
[945, '1A', 'NA'],
[945, '1B', 'NA'],
[945, '2B', 'O'],
[945, '2D', 'G'],
[945, '3B', 'O'],
[945, '3E', 'O'],
[945, '4A', 'O'],
[945, '4C', 'NA'],
[945, '5D', 'G'],
[945, '5M', 'G'],
[945, '6C', 'O'],
[945, '6F', 'G'],
[945, '7F', 'O'],
[945, '7G', 'O'],
[945, '8G', 'G'],
[945, '8H', 'O'],
[945, '9H', 'O'],
[945, '9I', 'G'],
```

```
[945, '10I', 'O'],
[945, '10J', 'G'],
[945, '11I', 'G'],
[945, '11K', 'O'],
[945, '12J', 'O'],
[945, '12L', 'G'],
[945, '13D', 'O'],
[945, '13K', 'G'],
[945, '14K', 'O'],
[945, '14L', 'NA'],
[945, '15L', 'O'],
[945, '15M', 'G'],
[945, '16M', 'G'],
[945, '16N', 'O'],
[945, '17E', 'O'],
[945, '17N', 'O'],
[945, '18N', 'G'],
[945, '18O', 'G'],
[945, '19D', 'G'],
[945, '19F', 'G'],
[945, '20G', 'O'],
[945, '20K', 'NA'],
[1000, '1A', 'NA'],
[1000, '1B', 'NA'],
[1000, '2B', 'O'],
[1000, '2D', 'G'],
[1000, '3B', 'G'],
[1000, '3E', 'O'],
[1000, '4A', 'O'],
[1000, '4C', 'NA'],
[1000, '5D', 'G'],
[1000, '5M', 'G'],
[1000, '6C', 'O'],
[1000, '6F', 'G'],
[1000, '7F', 'O'],
[1000, '7G', 'O'],
[1000, '8G', 'G'],
[1000, '8H', 'G'],
[1000, '9H', 'G'],
[1000, '9I', 'G'],
[1000, '10I', 'G'],
[1000, '10J', 'G'],
[1000, '11I', 'G'],
[1000, '11K', 'O'],
[1000, '12J', 'O'],
[1000, '12L', 'G'],
[1000, '13D', 'O'],
```

```
[1000, '13K', 'G'],
[1000, '14K', 'O'],
[1000, '14L', 'NA'],
[1000, '15L', 'O'],
[1000, '15M', 'G'],
[1000, '16M', 'G'],
[1000, '16N', 'O'],
[1000, '17E', 'O'],
[1000, '17N', 'O'],
[1000, '18N', 'G'],
[1000, '18O', 'G'],
[1000, '19D', 'G'],
[1000, '19F', 'G'],
[1000, '20G', 'O'],
[1000, '20K', 'NA']]
```

6.2 Program- June 10

Here is my notebook of functions so far. The purpose of the explain function is so a quick reference prints every time I import the functions notebook.

***

6.3 Modeling Traffic Conditions

Note: Speed limit on UTK campus is 20mph on campus streets and 35 mph on city street.

6.3.1 Green

Traffic Level	Speed Limit (mph)	Distance (ft)	Travel Time

6.3.2 Orange

6.3.3 Red

6.3.4 Brown

6.3.5 N/A

7 Noah's Program

Here is Noah's program as of June 1.

This is a very rough, but working, draft of my program. I will need to clean this up. First, I need to get rid of memory leaks by deleting all memory I allocated, then I will go through and comment what my program is doing. After that, I will try to make it more efficient.

```
#include <iostream>
#include <vector>
#include <map>
#include <list>
#include <algorithm>
#include <deque>
#include <fstream>

using namespace std;
class Edge {
    public:
    class Node *to;
    class Node *from;
    double weight;
};

//node class
class Node {
    public:
        Node();
        char id;                          //name of node
        multimap <double, Edge*> edges;  //edge list
        list <Edge*> ledges;
        Edge *backlink;  //sets all backlinks to null
        double distance;      //sets all distances to -1
};

Node::Node() {
    backlink = NULL;
    distance = -1;
}


//graph class
class Graph {
    public:
        map <char, Node *> graph;  //graph vector
        vector <Node *> v;
        //~Graph();               //destructor to delete all nodes
        multimap <double, Node*> Dmap;
        double Dijkstra(Node *start, Node *end);
        deque <Node*> path;
};


double Graph::Dijkstra(Node *start, Node *end) {
    list <Edge*>::const_iterator lit;
    Dmap.insert(make_pair(0, start));
    start->distance = 0;
    double d;
    Node *n;
    Edge *e;
    double dist;

    while (!Dmap.empty()) {
        n = Dmap.begin()->second;
        dist = Dmap.begin()->first;
        Dmap.erase(Dmap.begin()->first);

        for (lit = n->ledges.begin(); lit != n->ledges.end(); lit++) {
            e = *lit;
            d = e->weight + dist;

            if (e->to->distance == -1 || d < e->to->distance) {
                if (Dmap.find(e->to->distance) != Dmap.end())
                    Dmap.erase(e->to->id);

                e->to->distance = d;
                e->to->backlink = e;
                Dmap.insert(make_pair(e->to->distance, e->to));
            }
        }
    }
```

```
   n = end;
    while (n != start) {
        path.push_front(n);
        n = n->backlink->from;
    }

    return 0;
}


int main(int argc, char *argv[]) {

    Graph g;
    map <char, Node*>::const_iterator mit;
    map <double, Edge*>::const_iterator eit;
    int total = 0;
    char from, to, start, end;
    double weight;
    int i;
    Node *s;
    Node *e;
    ifstream fin;

    if (argc != 2) return -1;

    fin.open(argv[1]);

    //get total number of nodes
    fin >> total;

    for (i = 0; i < total; i++) {
        Node *n = new Node;
        n->id = 'A' + i;
        g.graph.insert(make_pair(n->id, n));
        g.v.push_back(n);
    }


    while (fin >> from >> to >> weight) {
            Node *n1 = g.graph.find(from)->second;
            Node *n2 = g.graph.find(to)->second;
            Edge *e = new Edge;
            e->weight = weight;
            e->from = n1;
            e->to = n2;
            n1->edges.insert(make_pair(weight, e));
            n1->ledges.push_back(e);

    }

    for (mit = g.graph.begin(); mit != g.graph.end(); mit++) {
        Node *n = mit->second;

        for (eit = n->edges.begin(); eit != n->edges.end(); eit++) {
        }
    }

    printf("Starting node: ");
    cin >> start;
    printf("Ending node: ");
    cin >> end;

    s = g.graph.find(start)->second;
    e = g.graph.find(end)->second;

    g.Dijkstra(s, e);

    printf("Path: ");

    printf("%c ", g.graph.find(start)->first);

    for (i = 0; (size_t) i < g.path.size(); i++) {
        printf("%c ", g.path[i]->id);
```

```
    }
    printf("\nDistance: %.2lf\n", g.path[g.path.size()-1]->distance);


    return 0;
}
```

8 Works Cited

Labeled Map of Knoxville Map is from Apple Maps
Data collected from Google Maps

9 Latex/Python/etc Notes

- To comment out code: command-/
- Online Jupyter Notebook Viewer: https://htmtopdf.herokuapp.com/ipynbviewer/

10 Time Log

10.1 NOAH

MAY

	5/23	5/24	5/25	5/26	5/27	5/28	5/29	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
					9:00		3:30	
					10:00		7:00	
					8:00			
					11:00			
Total:	0:00	0:00	0:00	0:00	4:00	0:00	3:30	**7:30**

JUNE

	5/30	5/31	6/1	6/2	6/3	6/4	6/5	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
	12:30		5:30					
	2:30		8:30					
Total:	2:00	0:00	3:00	0:00	0:00	0:00	0:00	**5:00**

	6/6	6/7	6/8	6/9	6/10	6/11	6/12	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
				10:00				
				10:30				
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 1	6/13	6/14	6/15	6/16	6/17	6/18	6/19	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
			6:00					
			10:00					
Total:	0:00	0:00	4:00	0:00	0:00	0:00	0:00	**4:00**

Week 2	6/20	6/21	6/22	6/23	6/24	6/25	6/26	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 3	6/27	6/28	6/29	6/30	7/1	7/2	7/3	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

JULY

Week 4	7/4	7/5	7/6	7/7	7/8	7/9	7/10	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 5	7/11	7/12	7/13	7/14	7/15	7/16	7/17	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 6	7/18	7/19	7/20	7/21	7/22	7/23	7/24	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 7	7/25	7/26	7/27	7/28	7/29	7/30	7/31	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

AUGUST

Week 8	8/1	8/2	8/3	8/4	8/5	8/6	8/7	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 9	8/8	8/9	8/10	8/11	8/12	8/13	8/14	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 10	8/15	8/16	8/17	8/18	8/19	8/20	8/21	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

10.2 KRISTINA

MAY

	5/23	5/24	5/25	5/26	5/27	5/28	5/29	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
					9:00		3:30	
					11:00		7:00	
					6:00			
					6:30			
Total:	0:00	0:00	0:00	0:00	2:30	0:00	3:30	**6:00**

JUNE

	5/30	5/31	6/1	6/2	6/3	6/4	6/5	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
	5:00		2:00		9:30		9:00	
	7:00		7:00		11:30		10:00	
Total:	2:00	0:00	5:00	0:00	2:00	0:00	1:00	**10:00**

	6/6	6/7	6/8	6/9	6/10	6/11	6/12	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
				10:00	10:00			
				10:30	12:00			
				9:00	2:00			
				10:30	5:00			
					7:00			
					10:00			
Total:	0:00	0:00	0:00	2:00	8:00	0:00	0:00	**10:00**

Week 1	6/13	6/14	6/15	6/16	6/17	6/18	6/19	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
			6:00		10:00			
			10:00		1:00			
Total:	0:00	0:00	4:00	0:00	3:00	0:00	0:00	**7:00**

Week 2	6/20	6/21	6/22	6/23	6/24	6/25	6/26	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 3	6/27	6/28	6/29	6/30	7/1	7/2	7/3	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

JULY

Week 4	7/4	7/5	7/6	7/7	7/8	7/9	7/10	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 5	7/11	7/12	7/13	7/14	7/15	7/16	7/17	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 6	7/18	7/19	7/20	7/21	7/22	7/23	7/24	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 7	7/25	7/26	7/27	7/28	7/29	7/30	7/31	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

AUGUST

Week 8	8/1	8/2	8/3	8/4	8/5	8/6	8/7	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 9	8/8	8/9	8/10	8/11	8/12	8/13	8/14	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**

Week 10	8/15	8/16	8/17	8/18	8/19	8/20	8/21	
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
Total:	0:00	0:00	0:00	0:00	0:00	0:00	0:00	**0:00**