Journal Report 2
9/9/19 - 9/15/19
Richard Zhan
Computer Systems Research Lab
Period 2, White

---

## Daily Log

### Monday September 9

I fixed a bug which caused my imported Edges to have the same Vertices. When I instantiated them, I was passing in the Vertices by value instead of reference. I modified all my methods to pass the Car, Edge, and Vertex classes by reference instead of value.

### Tuesday September 10

I implemented the method 'findReachable', which is a simple BFS. It will be used to find the potential car destinations from each Vertex. This will be used in car generation when we assign a destination to a Car. The method excludes itself from the output list.

### Thursday September 12

I coded the A* algorithm for navigation. I spent a lot of time playing around with the C++ priority_queue structure in order to use a clean implementation of it. I plan to continue testing the A* algorithm to make sure it functions correctly before moving on.

## Timeline

| Date | Goal | Met |
|---|---|---|
| 8/26/19 - 9/1/19 | N/A | N/A |
| 9/2/19 - 9/8/19 | Begin to setup framework for the program. | Yes, I have created the classes (Car, Edge, Vertex) which will represent my road network. |
| 9/9/19 - 9/15/19 | Finish coding the basic A* navigation system and collect data on the average amount of time for each trip. | No, I have coded the non-DTD navigation system, but have not written the necessary simulation code to collect data. |
| 9/16/19 - 9/22/19 | Finish writing the simulation code and collect data on the average amount of time for each trip. | |
| 9/23/19 - 9/29/19 | Finish coding a naive (non-optimized) DTD scheme. | |

## Reflection

Last week, I was too ambitious in setting goals. I wanted to code the navigation system as well as collect data, which would require also writing code to simulate a run. In reality, I have coded the basic navigation system for my program. I also coded the method *findReachable*, which will be run on each Vertex at the start of the program. It generates all of the *possibleCarDestinations* for each Vertex. This will prevent the creation of impossible journeys. However, for the proof-of-concept model, I need additional restrictions on the destinations. Next week, I plan to update some of my input files and input file reading methods, so I can control which Vertices can be used as destinations. This function will only be used temporarily for my proof-of-concept model, since I only want to two Vertices to be possible destinations. Next week, I would like to continue testing my A* algorithm to ensure that it is working properly. I would also like to begin writing the simulation code, and my optimistic goal would be to finish it by the end of the week.

---

```
/**
  * Fills variable 'start.edges' with Vertices that can be reached from
     variable 'start'
  * variable 'start.edges' will not include variable 'start'
  */
void findReachable(Vertex& start) {
  vector<Vertex> queue;
  queue.pb(start);
  int index = 0;
  unordered_set<int> visited;
  visited.insert(start.id);
  while (index < queue.size()) {
    Vertex state = queue[index];
    if (index > 0) {
      start.possibleCarDestinations.pb(state);
    }
    for (Edge e: state.edges) {
      if (visited.find(e.end->id) == visited.end()) {
```

```
          visited.insert(e.end->id);
          queue.pb(*e.end);
        }
      }
      index++;
    }
}
```