Journal Report 3
9/16/19 - 9/22/19
Richard Zhan
Computer Systems Research Lab
Period 2, White

Daily Log

Monday September 16

I added a new row in my input file to denote which Vertices could be destinations. I updated my input methods, so they could take the input for which Vertices were potential destinations.

Tuesday September 17

I found a bug in the method ′astar′ that caused incorrect output. Instead of representing the key-value pair for the variable ′parent′ as (Vertex, Edge), I accidentally used (Vertex, Vertex), which caused issues when the destination was found and the route needed to be traced back with the variable ′parent′.

Thursday September 19

I began writing the simulation code and set up global variables such as ′TIME_STEP′ and ′CURRENT_TIME′. I also added variables in the class ′Car′ to track duration of the trips. I started writing the method ′step_simulation′, which does the bulk of the simulation.

Timeline

Date	Goal	Met
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.	No, I corrected a bug in the navigation methods for non-DTD cars. I started but have not finished the simulation code.
9/23/19 - 9/29/19	Finish writing the simulation code and tweak variables to reach realistic settings.	
9/30/19 - 10/6/19	Finish coding a naive (non-optimized) DTD scheme.	

Reflection

This week, I added new input variables that allowed me to control the allowed Car destinations. Then, I tested my program with different input values and found a bug in my navigation system. It took me a while to debug the issue was, but it boiled down to changing a single line (line 44) from $parent[e.end->id] = state.id;$ to $parent[e.end->id] = e.id;$. After fixing this bug, I started writing the simulation code. I plan to use a central method 'step_simulation' that will spawn Cars, update their locations, and record their times.

```
1  /**
2   * Returns the list of Edges that represents the fastest route from 'start'
      to 'end'
3   * Uses the A* algorithm on 'graphVertices' and 'graphEdges'
4   */
5  vector<Edge> astar(Vertex& start, Vertex& end) {
6    unordered_map<int, int> parent; // <VertexIndex, EdgeIndex>
7    unordered_map<int, ld> cost;
8    parent[start.id] = -1;
9    cost[start.id] = 0.0;
10
11   auto compareVector = [&] (Vertex& v1, Vertex& v2) -> bool {
12     ld cur1 = 0.0;
13     ld cur2 = 0.0;
14     if (cost.find(v1.id) != cost.end()) {
15       cur1 = cost[v1.id];
16     }
17     if (cost.find(v2.id) == cost.end()) {
18       cur2 = cost[v2.id];
19     }
20     ld d1 = cur1 + distance(end, v1) / MAX_SPEED;
```

```
    ld d2 = cur2 + distance(end, v2) / MAX_SPEED;
    return d1 > d2;
  };
  priority_queue<Vertex, vector<Vertex>, decltype(compareVector)>
     pq(compareVector);
  pq.push(start);

  while (!pq.empty()) {
    Vertex state = pq.top();
    pq.pop();
    if (state.id == end.id) {
      vector<Edge> route;
      int cur = state.id;
      while (parent[cur] != -1) {
        route.pb(graphEdges[parent[cur]]);
        cur = graphEdges[parent[cur]].start->id;
      }
      reverse(route.begin(), route.end());
      return route;
    }
    for (Edge e: state.edges) {
      ld cur_cost = cost[state.id] + e.length / e.speedLimit;
      if (cost.find(e.end->id) == cost.end() || cur_cost < cost[e.end->id]) {
        cost[e.end->id] = cur_cost;
        parent[e.end->id] = e.id;
        pq.push(*e.end);
      }
    }
  }

  cout << "ERROR: No route found!" << endl;
}
```