**Part 1: Implement a basic driving agent**

I’ve saved this basic implementation as agent1.py which made just one simple change to the original file:

Line 28 was originally: action = None

And has been changed to action = random.choice(Environment.valid\_actions)

Which chooses an action randomly (None, left, right, forward.)

The agent will move in accordance to the random action chosen as long as the action is allowed by the current state of the traffic lights.

Result: The agent eventually reaches the destination through trial and error but not after considerable time as it moves in random directions.

**Part 2: Identify and update state**

Every state will be determined by a 2-tuple: (Waypoint, Light)

Due to the # of cars in the simulation, it’s very rare that we run into another car on the left, right, or oncoming positions and even then, there are no rewards or penalties for running into other cars. So to simplify the learning, I’ve left those variable out of the state.

I’ve implemented this intermediate step as agent2.py.

**Part 3: Implement Q-Learning**

I implemented Q-Learning with the learning \_rate = 0.5 and discount\_factor = 0.5. The Q dictionary is initialized with all values set to 0 by default. At every step, the agent checks the Q-values for all possible actions (None, left, right, and forward) and picks the one that yields the largest Q-value. If all possible actions lead to a Q-value of 0 (which means those Q-values have never been calculated yet), then the agent will just pick one of the 4 actions randomly.

I also started to enforce deadlines and changed the # of trials to 100.

With the above implementation, I found that my agent would quickly learn not to disobey the traffic signals after several trials. However, it would get stuck in a local optima where it ends up staying in the same place (action = None) and collecting a reward of 1 every time. Since the traffic lights seem to cycle every 5 rounds, I’ve modified my agent to keep track of how many rounds in a row did it choose action = None and if it’s greater than 6, then it chooses an action at random regardless of Q values. With this change, the agent doesn’t get stuck in one place and can continue learning.