Open the anaconda prompt

Run following command

```
conda activate base
```

Running the simulation:

To run the program run python SearchAgent.py <config_file_name>

For example python SearchAgent.py dynamic_config1.txt

On the gui your car is the white color car.

The config file contains the starting location of your car and the number and the starting location of other cars on the road.

The car has visibility range of 5 cells, i.e. it can view 5 cells ahead from its current location.

===

We describe the simulator in which you will be writing the code. There are 5 files provided.

1. Simulator.py -contains the code for the gui.
2. Environment.py – contains the code for environment and movement of other cars.
3. searchUtils.py – contains utility functions for search algorithms.
4. randomagent.py – contains the example code for a car which takes a random action at each timestep.
5. SearchAgent.py – contains the code for your car. This is the file you need to modify.

You need to **write the aStarSearch function in the SearchAgent.py** which returns the path to be taken by the car. This function is called at each step.

At every timestep first the computePath function is called which computes the path using aStarSearch and then the update function in SearchAgent class is called which conveys the intended action to the environment based on the computed path. Environment returns the updated state of the car and also updates the state of other cars in the environment.

Environment first updates the position of your car before updating the position of any other car.

Following are the class variables for the agent class:

valid_actions – list of valid actions available for your car.

Env – Instance of the environment class.

Searchutil – Instance of the searchUtils class

state – dictionary representing state of your car. State["location"] – provides the location of your car in grid cell. For eg. If state["location"] = (2,3) it means that car is located in the cell 2,3.

Following are the functions which you can use to write the algorithm for self driving car to choose an action at each timestep.

1. get_hVal(self,start,goal): returns the heuristic cost from start to goal.

2. getNextState(self,s,a): returns the new state obtained on taking a in state s.

3, isThrereAnUnexploredState (selfl): returns true if there is an unexplored state.

4. isGoal(self,current): returns true if current state is goal.

5. getBestStateToMove(self) – returns the best state to move based on current gval+hval.

6. retrievePathFromState(self,s): retrieves the path taken to reach state s

7. updateHeuristicValue(self,nextState,ghval): - updates the heuristic value of nextState as ghval.

Environment class functions – can be accessed by calling self.env.<functionname>

1. getGoalState() – returns the goal state.
2. act(car,a) – takes action a for the car and returns the new state. Updates the position of the car in the environment.
3. applyAction(s,a) – returns the location where the car will move on applying action a in state s based on car's visibility. Does not execute the action and does not update the state of the car in the environment.
4. getAction(s1,s2) – returns the action which takes a car from state s1 to state s2.

Searchutils class functions – Searchutils class contains utility functions for the search algorithm can be accessed by calling self.searchutil.<functionname>

1. retrievePathFromState(s) – retrieve the path taken to reach the give state s.
2. isPresentStateInList(state,searchlist) – returns 1 if the state is present in searchlist
3. isPresentStateInPriorityList(state,searchlist) – returns 1 if the state is present in priority list searchlist.
4. insertStateInPriorityQueue(searchList,state,distanceToGoal) – insert state with cost distancetogoal in the searchList.
5. checkAndUpdateStateInPriorityQueue(searchList,state,distanceToGoal): checks if the state is present in the searchList with a cost higher than distanceToGoal. If the existing cost is higher then state is reinserted with the cost distancetoGoal in the searchList.