CS 4300: Artificial Intelligence

Assignment: Model Search Agent

Create an agent to perform in the Taxi environment, (taxi source code).

The performance measure used by this assignment to assess the quality of your agent will be the episode total reward, averaged over at least 100 episodes. An agent that doesn't complete some episodes (times out / runs out of memory / crashes), will be given an average score of -200.

After your report and code are reviewed, assignment grades will be assigned. The maximum *possible* score will be controlled by the agent's performance measure. See the table below.

Use the GitHub repository available for this course to store your solutions. Make a directory named `taxi-model-search`, and store your agent in `taxi-model-search/agent1.py`. This file must contain a class `Agent1` with the methods:

- `def __init__(self)`: initializes the agent, including creating a model instance
- `def reset(self)`: resets the agent's data and model instance
- `def agent_function(self, state)`: receives `state` an observation from the gymnasium environment, returns an action for the gymnasium environment.

Note that you are to implement an agent that has a *model* of the environment and uses one of the uninformed *search* algorithms we have discussed. DO NOT make a reinforcement learning agent, or use some other algorithms for these agents.

The model must contain at least these methods:

- ACTIONS(s) -> list of actions allowed in state s
- RESULT(s, a) -> state that results from action a in state s
- GOAL_TEST(s) -> true or false, depending on the state s

Where `s` is the state received from the gymnasium environment, and `a` is an action number used by the gymnasium environment.

We discussed in class that some form of depth limited search would probably be most successful for this problem, with iterative deepening search probably being the best.

Create a short report, containing these elements:

- An explanation of which search algorithm you chose, and why. It's fine if you use our discussion in class to provide the explanation of why you chose it.
- The average performance of your agent over 100 (or more) episodes.

Required Submissions

- Code submitted to github.
- A PDF file containing your report submitted to Canvas.

Performance Measure Expectations

Average Score	Maximum Possible Grade
a < -100	50%
-100 <= a < 0	65%
0 <= a < 1	75%
1 <= a < 2	80%
2 <= a < 3	85%
3 <= a < 5	90%
5 <= a < 7	95%
a >= 7	100%

Hints

Any solution to this problem requires that you drive to the passenger, pick them up, drive to the destination,

drop them off.

You might want to consider solving the problem in two stages:

- Solve the problem of driving to the passenger and picking them up.
- Solve the problem of driving to the destination and dropping off the passenger.

This will make the depth of goal states about $^1/_2$ of goal states that would be found solving the full problem. Since the depth of solution is the exponent in our time complexity, this is a *huge* time savings.