Project 2: Multi-agent Search

Introduction:

For this project, you will need to design a Malmo agent that will perform optimally while avoiding the enemy and attempting to arrive at all of the destination blocks. Along the way, you will need to implement an evaluation function, an agent action selection function and an agent movement function.

A capable agent should be able to consider both destination block locations and enemy location when choosing an optimal action. Meanwhile the enemy will pick the next action to take from a list of available options randomly.

Files you'll edit:

reflex.py

Files you can ignore:

multiagent.py
testClassic.lay
smallClassic.lay
openClassic.lay

Files to submit:

reflex.py

Academic Dishonesty:

The University of Virginia's Honor Code: students pledge never to lie, cheat, or steal, and accept that the consequence for breaking this pledge is permanent dismissal from the University.
In general, we expect that you will be using code, examples, and ideas from many different websites and resources for your projects. This is allowed within reason. Wholesale copying of a

project is not allowed and will result in an honor violation. In ALL cases, you need to cite all sources at the top of the file where the code or algorithm was used. Failure to properly attribute your sources will result in a 0 for the project at a minimum.

Getting Help:

You are not alone! If you find yourself stuck on something, contact the TAs for help. If you can't make our office hours, let us know and we will schedule more. We want these projects to be rewarding and instructional, not frustrating and demoralizing. But, we don't know when or how to help unless you ask.

Reflex Agent:

In this project you will implement a reflex agent that will be able to visit all the destination blocks while avoiding the enemy. To make things easier for you, we've provided some helper functions in reflex.py. You will find instructions for how to use these helper functions in the description of three questions.

To test your code, simply run

```
python multiAgent.py -m LayoutFileName
```

to see how your agent behaves.

Important: Since we are using multiple agents, you will need to launch 2 seperate Minecraft clients in order to run the files we provided (each agent must run in it's own separate window). To do this simply run the *launchClient* script twice in 2 separate command line windows. Once two Minecraft clients are fully loaded, click on the mods tab in each client, scroll down and find the Malmo mod and click configure. Here you will see the port that is assigned to each client. Make sure that your 2 ports are 10000, 10001. Your code will not run if your ports are not configured correctly!

Question 1 (10 Points): Evaluation Function

Implement the evaluation function *evalfuncReflex()* in reflex.py. This function should be able to evaluate state-action pairs and return a score for each action. In other words, the evaluation

function should assess how far the player is from the nearest destination block, how many destination blocks have not been visited and how far the player is from the enemy at any given time. To make things easier for you, we provided you with a *manhattan_distance*() method to quickly calculate distances. Your evaluation function should take in all the provided parameters and return a score, which you'll use in Q2.
Basically, your agent's intelligence would depend on your evaluation function of a given possible action. The better your evaluation function is, the faster your agent will be at completing the task. Your agent should easily and reliably clear the layout files which are provided by us.

Hint 1: As features, try the reciprocal of important values (such as distance to food) rather than just the values themselves.

Hint 2: When doing the evaluation, update the remaining blocks which have not been visited to take the current action's influence into consideration.

Question 2 (3 Points): Agent's Action Selection

In this question you will implement the agent's action selection function *chooseAction()* in the reflex.py. Decide which action is the best for the agent to take at its current. The best available action is determined by the evaluation function, you implemented in Q1.

To help you, we have provided you with an *illegalMoves()* method, which determines what actions would result in the agent walking into a wall. The method returns a list of actions the agent cannot take.

Your function should return a string specifying the direction of the next move e.g. "left".

Question 3 (2 Points): Agent's Movement Function

Implement the movement function *reflexAgentMove(),* it will update agent's location based on its movements. Your agent should move according to the action selected in Q2.

To help you, we also provided you with helper movement functions (similarly to project 1), so you can call these functions to send movement commands to the agent.

Alternatively you can just use the standard Malmo movement commands if it makes you implementation simpler.

Remember that after you move the agent, the world_state value that was originally declared at the start of the main mission loop and that was used to move the agent is not updated automatically. In other words, when you move the agent, the `world_state` value is still the world state of the agent before it moved. Thus you must update the world state and get a new observation after you moved the agent. To better understand how this works, look at the code we provided for running the enemy agent.

Grading:

We will run your code on a number of different terrains to ensure that your code can arrive at all of the destination blocks safely for each map. You will be awarded 10 points for a correct implementation of evaluation function, 3 points for a correct implementation of agent's selection of action, and 2 points for a correct implementation of agent's movement logic for a total of 15 points.