Artificial Intelligence
Project3

Overview

Goal: Write an agent to locate and catch the ghosts in a maze. You cannot see the ghosts, but can hear them (not very accurate). Each ghost starts at a random positon in the maze and moves around the maze using on of 5 predefined behaviors: does't move, moves aways from the player, moves toward the player, moves left to right and though the passage, and moves clockwise around maze. You need to use probabilistic reason to identify the behavior of each ghost and where they started. The will allow you to determine where they are and where they are going so you can capture them. Ghosts will move every two seconds. You will be able to have your player move as often as once a second; however, if you use more computation time it will move slower.

Files: The files are available in the class git repo. The the course page for instructions on access the git server. The name of this assignment is `project3`. You should only edit MyPlayer.h and MyPlayer.cpp. You should also write a description.txt file.

Grading: Each assignment will be graded as follows:

50%	Performance on test problem instances relative to reference implementations
10%	Accuracy in finding the correct location of the ghosts
20%	Code quality
20%	Description of algorithm used and why it is effective
up to 5% bonus	Performance relative to your classmates

Full Version

Due: 5 pm Monday December 10. Late assignments will not be accepted.

Goal: Write an agent to catch the ghosts as quickly as possible.

Submission: Performat a git commit and a git push to submit your files. Submitting your files will automatically test your code and post results on the contest page. This way you know how you perform relative to your classmates and the reference implementations. Note that the final grading will be using a different set of test instances.

Usefull functions In the Player class, which is the base class for MyPlayer, you have a member variable _maze that has several useful functions for dealing with the maze. (See the header file for all of the functions, I've put the ones that I think you might need here.)

validPositions() return all of the positions in the maze that can be occupied by ghosts and the player.

validMoves(position) return the moves that are possible from a given position

result(position, move) return where the player will be if it makes a specific move

mazeDistance(position1, position2) return the distance in the maze that follows the passages between to positions

directionToward(start, goal) what direction you should head if you are at start and want to reach goal.

noisyDistanceProb(noisyDistance, actualDistance) return how likely it is that a ghost actualDistance away was reported to be noisyDistance away from the player.

Also, the Player class provides the following member functions for your MyPlayer class. Note that ghosts have ids 0–2 and behaviours 0–4. One of the main internal represetations you should be utilizing is a map that stores that probability the each ghost current is in a particular position and has a specified behavior. This is stored as a map from `tuple<Position, int>` to `double`. The tuple stores the ghosts position and behavior and tells you the probability that the ghost is located at the position and has the specified behavior.

clearCalculatedGhostProbabilities() Clear the probabilities of the ghosts position an behavior that you have set.

getCalculatedGhostProbability(ghost, position, behavior) Return the probability that a ghost currently has the specified position and behavior.

setCalculatedGhostProbability(ghost, position, behavior, prob) Set the probability that a ghost currently has the specified position and behavior.

calculatedGhostProbabilities(ghost) Get the entire map of probabilities for a ghost.

timeUsed() how much time (in seconds) elapsed in the players calculations.

Finally, the Observation and Observations classes store the sequence of observations in a given game. Look at the header file. You will probably want to use numObservations, getObservations and ghostAlive in the Observations class and description, data1 and data2 in the Observation class. The header file documents how observations are stored.

Hints: To do well consider the following:

- Submit agents often. Test them, find improvements, document and submit again.
- Try basic algorithms/heuristics first. Perhaps start with stationairy ghosts first.
- Don't examine the entire observation sequence on every move, only examine what has happed since your last move.