Project 2 Part 1 (Multiagent Pacman)

David Hoang, 913611912

CSC 665 Spring 2018

Overview

The objective of this assignment was to do questions 1-2, filling in the ReflexAgent evaluationFunction and MinimaxAgent getAction function so that they would work in accordance to the provided auto-grader.

ReflexAgent evaluationFunction

The goal of this part was to fill in the evaluationFunction inside the ReflexAgent class. The premise of the evaluation is simply the distance of the closest food and the closest ghost, along with the number of pellets left on the board. Some adjustments to the weight of each component was done, however. This implementation completes the autograder with a record of 10/10 wins with an average score of about 1245.

MinimaxAgent getAction

The goal of this part was to create a multiagent Minimax path search function, with pacman as agent 0, and with a variable number of ghosts.

The gist of this solution was in creating recursive ghostMin and pacmanMax functions. The ghostMin function starts from the bottom-up, first by recursively calling itself and the pacmanMax functions to get to the bottom depth where it then evaluates the states at that depth. Essentially, it is finding minimax evaluation values of the whole gameState (that is, considering all other ghosts' actions as well as pacman's action) for each of that ghost's action, then returning minimum value. The minimum value refers to pacman's expected worse positions of ghosts.

Then, pacmanMax function calls the ghostMin functions and finds the max values of the minimum ghost values returned from the ghosts' recursive ghostMin minimax function.

This implementation passes all the autograder's tests, however, many of the games are lost.

Thoughts

The ReflexAgent evaluationFunction was mostly straightforward, just tracking some reasonable components. The most difficult part was to figure out a good ratio of the three components that I used to evaluate the board.

MinimaxAgent getAction was much more difficult than any of the previous problems thus far. The concept of minimax itself was actually simple, but implementing it turned out to be a lot of work. Especially since I am not used to recursive thinking, usually opting instead for an iterative approach. However, any iterative solutions that I tried to implement ended up being very messy and did not work properly. It was impossible trying to think about all the steps required without writing it in a tree. It took a small bit of effort to also generalize it to accept any depth and any amount of agents.