Artificial Intelligence Homework 2
Multiagent

2014 / 11 / 12

Question 1 – Reflex Agent

- Given a game state, a reflex agent chooses the action that leads to the highest value of evaluation function.
- getAction(self, gameState)

```
# Collect legal moves and successor states
legalMoves = gameState.getLegalActions()
# Choose one of the best actions
scores = [self.evaluationFunction(gameState, action) for action in legalMoves]
bestScore = max(scores)
bestIndices = [index for index in range(len(scores)) if scores[index] == bestScore]
chosenIndex = random.choice(bestIndices) # Pick randomly among the best
return legalMoves[chosenIndex]
```

- evaluationFunction(self, currentGameState, action)

Question 1 – Reflex Agent

- Given a game state, a reflex agent chooses the action that leads to the highest value of evaluation function.
- getAction(self, gameState)
- evaluationFunction(self, currentGameState, action)

```
successorGameState = currentGameState.generatePacmanSuccessor(action)
newPos = successorGameState.getPacmanPosition()
oldFood = currentGameState.getFood()
newGhostStates = successorGameState.getGhostStates()
newScaredTimes = [ghostState.scaredTimer for ghostState in newGhostStates]
return successorGameState.getScore()
```

  - Hints: Go to the nearest food, eat the capsule then go chasing the ghosts.

Question 2 – Minimax Agent

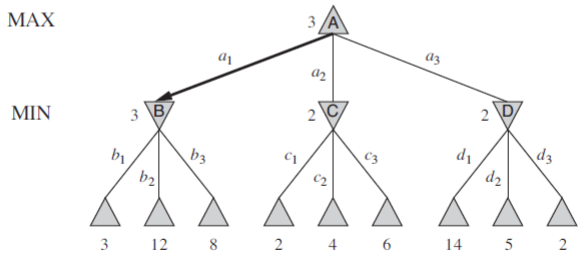
- Given a game state, a reflex agent chooses the action that leads to the highest value of evaluation function.
- getAction(self, gameState)

```
legalMoves = gameState.getLegalActions()  only consider the legal moves of the Pacman
_____                                     remove action "stop" from legal moves
numOfAgents = gameState.getNumAgents()    the total number of agents in the game
_____                                     new game states result from every legal moves of Pacman
scores = [self.evaluationFunction(gameState, action) for action in legalMoves]  the "game value" of  new game states evaluated by self.minimax
bestScore = max(scores)
bestIndices = [index for index in range(len(scores)) if scores[index] == bestScore]
chosenIndex = random.choice(bestIndices) # Pick randomly among the best
return legalMoves[chosenIndex]
```

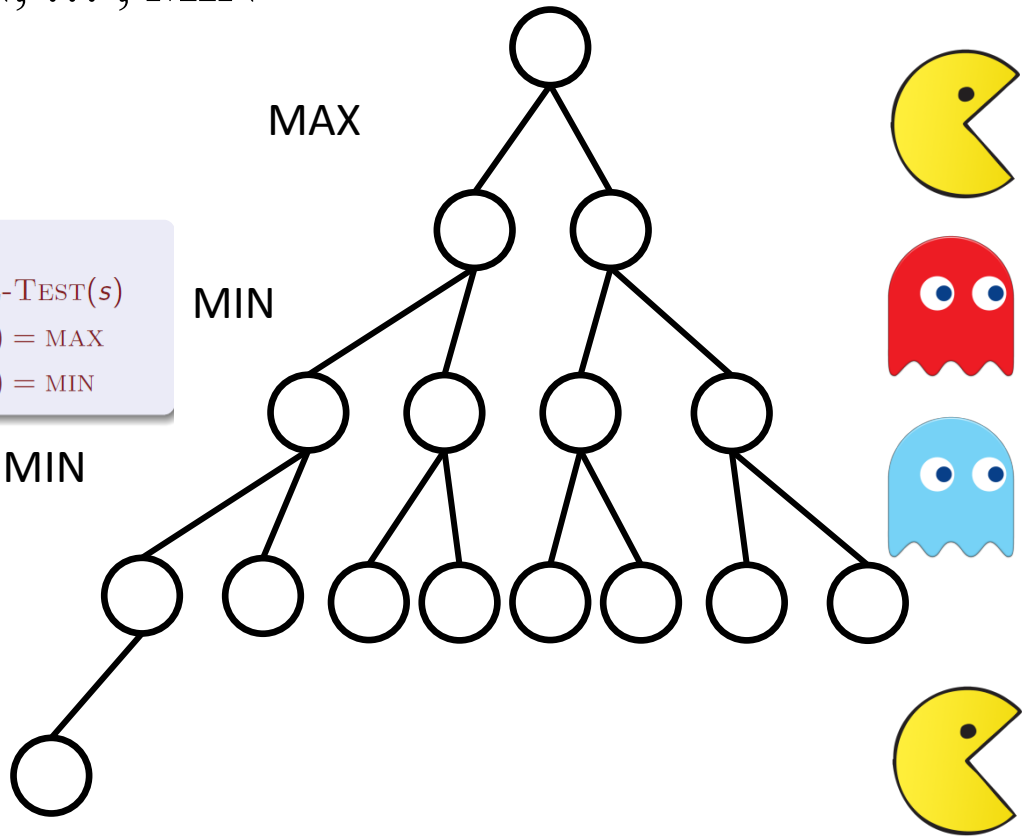
Question 2 – Minimax Agent

- minimax(self, gameState, depth, agentIndex)
  - 1 depth: MAX, MIN, MIN, … , MIN
  - recursive call
  - 10 lines only

$$\textsc{Minimax}(s) = \begin{cases} \textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\ \max_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{min} \end{cases}$$



Minimax Search



Question 3 – Alpha Beta Agent

- getAction(self, gameState)
  - nearly the same as minimax agent
- alphabeta(self, gameState, depth, agentIndex, alpha, beta)

Question 4 – Expected Minimax Agent

- getAction(self, gameState)
  - nearly the same as minimax agent
- expectiMinimax(self, gameState, depth, agentIndex)
  - nearly the same as minimax agent, except…
  - The ghosts don't return the min of all game values, they return the average of them.

Question 5 – Better Evaluation Function

- Originally, the evaluation is based on the score of the given state.
- Try to “eat” the ghost four times in the game!
- This evaluation function is different from the one of Question 1 in that here the evaluation is only a function of game state, where as in Question 1 the evaluation is a function of game state and action.