Single Dot Reinforcement Learning Assignment

In this assignment, we want to make Pacman learns to reach the dot and eat it. We have two programs:

* singleDotProblem.py and
* singleDot.py.

In singleDotProblem.py, we have two classes:

1. **State class** where it stores the agent position (that is Pacman).
   1. The constructor \_\_init\_\_ method that stores the agent position.
   2. Define special methods:
      1. Define \_\_eq\_\_ method which is a special method that is used to define how objects of a class are compared for equality. It is used to define how instances of a class are compared for equality using the == operator. When you use == to compare two objects, Python will call the \_\_eq\_\_ method to determine if the two objects are considered equal.
      2. Define \_\_hash\_\_ method which is a special method that is used to define how an object is hashed.

These two methods are part of the object’s “magic methods” which allow you to override default behavior in Python. It is used to define how instances of a class are hashed, which is important when objects are used in data structures like sets or as dictionary keys. If you define \_\_eq\_\_ in your class, you should also define \_\_hash\_\_ to maintain consistency between equality and hash values. The \_\_hash\_\_ method returns an integer that represents the hash value of the object. The rule is that if two objects are considered equal (\_\_eq\_\_ returns True), they must have the same hash value.

1. **Problem Class:** Where it stores the following methods:
   1. readMaze: to read the maze stored in a file name, this method defines some variables like wall positions, agent position, dots positions, potential\_moves, xMax (maximum width of maze) and yMax (maximum height of maze).
   2. isWall: returns true if the position is in the wall.
   3. getStartState: returns the start state of the problem that is the start agent position.
   4. isValidMove: that returns true if the agent position is valid, i.e. not inside the wall or outside maze boundaries.
   5. isTermina: returns true if the state is a terminal state, terminal state is when agent position is at the dot position.
   6. Reward: returns the reward for every state, at terminal state it returns a reward of 10 otherwise it returns a reward of 0.
   7. Transition: returns a list of valid new state and action, the format of the list is (state, action)

The file singleDot.py is the learning agent program where it uses the value iteration algorithm. The output of the code is shown in the following image.

A game of a game

Description automatically generated with medium confidence

Your assignment is to complete the part of code that is missing. Follow the instructions comments in the program singleDot.py.

You will need to:

1. Initialize the total return V (V = {})
2. The for episode loop: for episode in range(100):
3. To set the currentState as the problem start state:

currentState = p.getStartState()

1. To write an infinite loop: while True
2. To check if the currentState is a terminal state:

if p.isTerminal(currentState):

V[currentState] = p.reward(currentState)

break

1. To check if currentState is not in V then set V to zero for the currentState:

if currentState not in V: V[currentState] = 0

1. To neighbors: neighbors = p.transition(currentState)
2. To loop over all neighbors:

for s, a in neighbors:

if s in V: v = V[s]

else: v = 0

if v > maxV:

maxV = v

bestState = s

1. When extracting the policy, you need to loop over all states in V:

for s in V:

if p.isTerminall(s):

policy[s] = None

continue

neighbors = p.transition(s)

1. To get the maximum total reward for all actions applicable at state s:

for n, a in neighbors:

if n in V: v = V[n]

else: v = 0

if v > maxV:

maxV = v

bestAction = a

1. Save best action in policy:

policy[currentState] = bestAction

1. Try using the ‘singleDotMedium.txt’ maze and watch the output.

Make sure your code runs with no errors and produce the correct output. Create a github repository and save all files in the github repository.

Submit the link to your github repository in Canvas.