In this module of the project , we designed a class describing a Markov decision processes , to help students implement and learn about the reward based learning . To get a basic understanding of the class mdp and learning agents try going through the files:

MDP.py

lAgents.py

ValueAgent.py

Qagent.py

game.py

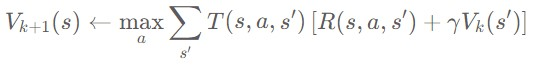
Files to be altered :

ValueAgent.py

Qagent.py

Q1. Value iteration:

Value iteration is a fundamental AI technique used to find optimal policies in Markov Decision Processes by iteratively refining value estimates for states until convergence, enabling intelligent decision-making in dynamic environments. In this question , the equation used for value iteration is given by :



After going through the files MDP.py and ValueAgent.py, you would have noticed some parts of the codes are missing. In this question, you need to implement the runIteration, getValue functions present in ValueAgent.py. runIteration runs the value iteration algorithm provided number of times, and getValue returns the value of the given state. The ValueEstAgent class has values dictionary as its feature , which you can use to implement these.

Q2. Policy evaluation :

After the value iteration has been run for certain number of times, the values of each state will be stored in a dictionary. In this question we will run the iteration 100 times, but we know that policy converges much before values. So in order to understand this, you need to implement getAction function in ValueAgent.py to get the policy associated with each state.

To see the agent under evaluated policy run the following command:

**python dungeon.py -p agent -method valearn -graphics True -world smallmaze.txt -omni T**

Q3. Q-learning:

Q-learning is a prominent reinforcement learning algorithm that helps AI agents learn optimal strategies by updating a table of action values based on experiences, enabling autonomous decision-making and policy optimization in complex, dynamic environments. So in this question , your agent will learn through episodes rather than a predefined mdp. Go thorugh QLearning.py file first to get an understanding. The equation to update Q-values after each episode :



You need to implement update, computeValueFromQValues, computeActionFromQValues, getQValue functions of QLearningAgent class in QlearnAgent.py. You are free to use any data structure to capture Q-values of each state action pair after a particular episode.

The update function takes a transition from a state to nextState through an action getting a reward and uses it to update the current Q-value of that state action pair.

computeValueFromQValues function returns Value of a state with the help of current Q-values.

computeActionFromQValues function returns best Action from a state with the help of current Q-values.

getQValue returns Q-value of a state action pair.

Q4. Policy evaluation of Q-values:

In this question , you need to implement getAction function of QLearningAgent. The Qlearning agent has a learning rate denoted by epsilon. We will use this getAction function to make the agent choose an action at random with the probability of epsilon or choose best action with the help of Q values.

To see the agent under evaluated policy in a unveiled world run the following command:

**python dungeon.py -p agent -method qlearn -graphics True -world world.txt -omni F**