**Question 1**

**Introduction:**

|  |
| --- |
| The problem statement asked for the implementation of a value iteration agent which runs a specified number of iterations to assign values to each state. It should update all the states in each iteration and at the end of the iteration, we have a value for each state. |

**Analysis & Explanation:**

|  |
| --- |
| The function runs in the initialization of the class - a sort of offline planning - so that when agent initializes, each state already has a value assigned to it and the agent can follow the action that gives it the maximum value.  In each iteration, states are iterated in the order of which they are received by the function getStates(). For each state, all possible actions are iterated to get a Q-value using the equation    The discount, probability and reward are obtained using function calls that returns previously stored values. The max q-value is taken from the array which stores value of each state and is 0 initially.  The maximum q value obtained at each state is then taken as the value of the state. Later, when an agent is at a state, it can look up the action to take by searching all the available actions and match which action produces the max q value stored for that state. In this way, policy is determined for each state. |

**Interesting Findings:**

|  |
| --- |
|  |

**Challenges:**

|  |
| --- |
|  |

**Behavior of code for different hyper parameters:**

|  |
| --- |
| (number of iteration ig) |

**Question 2**

**Introduction:**

|  |
| --- |
| The problem statement asked for assigning appropriate values of discount and noise that would allow an agent to cross a bridge safely, while avoiding the fire on either side. |

**Analysis & Explanation:**

|  |
| --- |
| In this problem, the optimal policy is already defined. The road to the goal is very risky since any mistake that causes agent to take any move other than the optimal move will cause it to fall into fire.    Discount value was thus given 0.9, since a higher value will give a higher importance to future state values and will assign higher q values to the actions that leads the agent to the states with higher q values. This essentially means the agent is less likely to take risks and will always choose the states giving the highest q value. The noise is assigned value of 0, because the noise would introduce uncertainty, which we cannot afford since even a small noise value might cause the agent to end up in an unwanted state after taking the optimal action. |

**Interesting Findings:**

|  |
| --- |
|  |

**Challenges:**

|  |
| --- |
|  |

**Behavior of code for different hyper parameters:**

|  |
| --- |
|  |

**Question 3**

**Introduction:**

|  |
| --- |
| The problem statement asked for assigning appropriate values of living reward, discount and noise that would allow an agent to get to the specified goal (either one of the two) using a specified path (either the safe long path or short riskier path). |

**Analysis & Explanation:**

|  |
| --- |
| The living reward causes the agent to place more importance on taking the longer road to the goal. This is because by taking the long road it will be able to collect more reward at the end since each state gives a reward, essentially rewarding the agent for exploration.  On other hand, higher discount would place more importance on reaching goals that are further away with higher value assigned to it. The future rewards are multiplied by this value thus the discount represents the amount of importance attached to the values of the states that can be reached in the future.  Finally, noise represents the uncertainty i.e. there is a probability that even after taking the optimal action, the agent may end up in a different state than the one specified in the policy.  With these factors in mind, the following cases were solved.    This asks the agent to take a risky path to a more closer goal. With the knowledge of the influence of these factors, the living reward was given a value of 0 so agent is not encouraged to keep on exploring. A discount value of 0.2 was given to encourage agents to reach the goal but not a higher value so agent does not move towards the higher valued goal (+10). Since the path is risky, a noise value could cause agent to fall into the negatively valued states so the noise value is assigned 0. This ensures the agent ends up where the action should take it to.    Since we still want to prefer the closer goal, the discount value is kept unchanged to 0.2. The living reward is increased to 0.5 to encourage the agent to take a longer path. However a noise of 0.1 is also introduced so that agent ends up moving towards the closer goal instead of moving towards the further exit because of living reward.    Previously, the lower values of discount made the agent prioritize immediate rewards over rewards in states further away. Thus value of discount is increased to 0.7 which is a balance between the agent wanting to reach the goal state further away with a higher value but not so high that it is not willing to take the riskier path. The noise value is kept at 0 but a value of 0.1 also works to make sure the agent does not end up in the negatively valued states by accident. Again, the living reward is kept low at a value of 0.2 so that agent does not explore.    The same values for the previous solution was used but living reward was increased so agent would explore the longer path to reach the goal. As before, the high discount causes agent to prioritize the higher value goal at a distance and low noise ensures agent reaches goal with little thrashing around. If noise is increased, the agent thrashes around but eventually reaches the +10 through the long way.    The living reward is increased to 1 while discount value is assigned at 0 so that all the states are assigned value 0.9 equally. The discount value 0 also ensures that agent does not place any importance in future goals which is why it does not move towards the closer or further away goal state. The noise value does not have much importance here as lower noise value means it stays in one place and higher noise value means it thrashes around but in the end, the episode does not terminate. |

**Interesting Findings:**

|  |
| --- |
| (living reward 1 diye noise dile shorter path ney but no noise dile longer path to +10) for 3c  3d te noise baraile thrash around kore eventually goal e jay |

**Challenges:**

|  |
| --- |
|  |

**Behavior of code for different hyper parameters:**

|  |
| --- |
|  |

**Question 4**

**Introduction:**

|  |
| --- |
| The problem statement asked for modification of the solution to the first problem by updating one state per iteration as opposed to updating all states in each iteration. The question instructs to ignore terminal state and update state in the order provided by getStates() function. |

**Analysis & Explanation:**

|  |
| --- |
| Iteration is done as usual but before iterating the a current state index is initialized at 0 and number of states is determined using the length of the state array returned by the getStates() function. Then in each iteration, using the current state index, the state is accessed and updated accordingly. Each time the current index is incremented by 1 and mod by the number of states to ensure cyclic value iteration. This is repeated until the number of iterations complete. |

**Interesting Findings:**

|  |
| --- |
|  |

**Challenges:**

|  |
| --- |
|  |

**Behavior of code for different hyper parameters:**

|  |
| --- |
|  |

**Question 5**

**Introduction:**

|  |
| --- |
| The problem statement asked for the modification of the first solution to implement a prioritized sweeping algorithm |

**Analysis & Explanation:**

|  |
| --- |
|  |

**Interesting Findings:**

|  |
| --- |
|  |

**Challenges:**

|  |
| --- |
|  |

**Behavior of code for different hyper parameters:**

|  |
| --- |
|  |