Assignment #4

Sudhanshu Pathak

Q1:

NOTE: I have used Java for programming. Please run program in following format:

```
To run policy:
java PolicySimulator

To run Qlearning agent:
java QLearning

Please put code and mdp.txt under same folder.

Results:
Results are included in the submission
Result1.txt for MDP1
Result2.txt for MDP2
```

MDPSimulator:

Simulator is Designed in such a way that it accepts the input MDP in the format, state, action and reward and generates the reward for given input, such as :

mdp.simulateMDP(1000, "P1", **null**)

Here 1000 is number of trials, P1 is the policy and third parameter is for QFunction.

Agent starts from A,10,T,F or A,10,F,F randomly
Based on the input policy, the action is taken agent transition transitions to nextState and reward is calculated.

For Parking MDP I have considered following assumption:

Implementation is divided into following parts:

```
.populateDriveMatrix();
.populateParkMatrix();
.populateExitMatrix();
.calculateReward();
```

Firstly, the no of states are considered as 81 x 81 due to Location {A,B}, O {T,F}, P{T,F} combinations. Each states is represented as “A1,T,F” similarly there would be 80 states and one last for EXIT state.

The objective of the implementation is to calculate three matrices mainly Drive, Park and Exit.

Drive:

Assumptions: Towards A1 or B1 the probability increases. Towards Bn probability decreases.
The cyclic order is considered to generate matrix as given in question. There is a specific set of transition from A1 to B1 and B10 to A10.

Park:

The action Park on incoming state if Park is false then probability is set to 1.

Exit:

The action Exit on incoming state if Park is true then Probability is set to 1.

Reward:

Reward is calculated in the following rule as mentioned in question:

MDP1 and MDP2 varies in terms of the rewards taken:

MDP1:
If park is false then add reward -1
If park is true and occupied is also true then it would mean agent would collide hence add a negative reward of -100.
For the handicapped location, give a negative reward of -50.
For successful parking in any other location give reward with respect to its location's closeness to the store. (1000/no) where no is the location of the agent.

Reward Function:

```
if (park.equals("F")){
      reward.add(-1.0);
}else if (park.equals("T")){

      if(ocup.equals("T")){
            reward.add(-100.0);
      } else if (no == 1){
            reward.add(-50.0);
      } else{
            reward.add(1000.0/no);
      }
```

MDP2:

If park is false then add reward -1
If park is true and occupied is also true then it would mean agent would collide hence add a negative reward of -10.
For the handicapped location, give a negative reward of -5.
For successful parking in any other location give reward with respect to its location's closeness to the store. (100/no) where no is the location of the agent.

Reward Function:

```
                if (park.equals("F")){
                reward.add(-1.0);
          }else if (park.equals("T")){

                if(ocup.equals("T")){
                      reward.add(-10.0);
                } else if (no == 1){
                      reward.add(-5.0);
                } else{
                      reward.add(100.0/no);
                }
          }
```

Q2:
Performance comparison of Policies:

	Policy1	Policy2	Policy3	Policy4
MDP1	40.90511	87.84458	108.93582	109.4053
MDP2	12.7565	16.6784	19.49388	19.5769

Policy3:

For policy3, I used simple value of occupied parameter, if Ocupied is true then simpy return 0 else 1.This will help agent gain more reward.

```
          if (ocup.equals("T")){
                return 0;
          } return 1;
```

Policy4:
For policy 4, I used Ocupied and Park parameters. If Ocupied is true and Park is false then

return 0 else return 1. This will help agent gain slightly more reward than policy 3.

```
if (ocup.equals("T") & (park.equals("F"))){
                        return 0;
                } return 1;
```

Q3:

QLearning Algorithm:

The QLEarning algorithm will calculate the Q(s,a) of all the states for given number of times. The Q(S,A) matrix space will be of 81x3 since for parking lot there are only three possible actions. (S is state and A is action).
For this algorithm, normally I have run trials for 1000 times.

Algorithm:
In order to achieve the Exploration and Exploitation, I have considered following conditions. If state's park status if True then it would mean vehicle is parked and returned action is Exit which 2. Otherwise, calculate a mathematical value using expression 1- (((1-0.1)/10000)*trials). Normally this expression would return value between 0 to 1.
Explore:
If the number is less than a random number than chose action between Drive and Park.
Exploit:
Get the action which gives the maximum reward.

Exploration/Exploitation logic:

```
if(park.equals("T"))
      action = 2;
else{
      double d = greedyPolicyCalc(i);

      //Explore
      if(Math.random() <= d){
            action = rand.nextInt(noAction-1);
      }else {   //Exploit
            action = maxQAction(state);
      }
}
```

Now, calculate the nextState usning random function for the input **action**.
Using random calculate the probability and chose the state having probability higher than the generated probability.

For example if random = 0.3 and State = [0.1 0.1 0.3 0.7] then chose the state between 0.3 and 0.7 randomly. This provides better distribution.

Apply qFuncton formula:
qFunction[state][action] = qFunction[state][action] + alpha*(rewardMatrix[0][state] + beta*maxQValue(nextState) - qFunction[state][action]);

Finally set the nextState to currentState.

Learned behavior:
It learns more during exploration
During Exploitation agent gains more knowledge

Graphs:
MDP1.txt

MDP2.txt