Assignment 10

Value of Information

(3 pts) Complete the following question of the Berkeley Spring 2021 Project 4:

https://inst.eecs.berkeley.edu/~cs188/sp21/project4/#question-8-4-points-value-of-perfect-information

- The code to be changed for this question is in bayesAgents.py. Note that my solutions to previous parts of this project were posted as Assignment 6 Solution under Assignments at Blackboard. You are welcome to use this code as a starting point for your work on this question.
- Part (a)
    - Should say self.bayesNet, not self.BayesNet
    - Note that nothing is said about the bottom positions for houses. This is because for purposes of this question the houses are always both in top positions. So you do not need to write any code related to bottom positions.
    - You might find it helpful to glance through the body of getExplorationProbsAndOutcomes() for some examples of code that does some of the sorts of things your code will be expected to do.
    - Note that you should include in any assignment passed to getProbability() all of the evidence provided to computeEnterValues().
        - However, your code should not modify the provided evidence. You might want to make a copy of the evidence (e.g., by calling the copy() method on it) as a starting point for creating your assignments.
- Part (b)
    - Note that the `enterEliminationOrder` parameter is the elimination order to be used as the argument in any calls to computeEnterValues().
    - You can, and perhaps should, ignore the Hint for part (b). It seems to be a less-informative version of the directions already given earlier in part (b).
- When both parts (a) and (b) are complete and your code passes the autograder, to see your VPI agent in action you can run

```
python hunters.py -p VPIAgent -v
```

Decision Networks

The following question is adapted from

https://aimacode.github.io/aima-exercises/decision-theory-exercises/ex_22/

For parts 1, 3, 4, 5, and 6 below, add code to the provided file CarBuyingTemplate.py where indicated. Your code should use parameters passed to decision_network() rather than hard-coding the numbers given below. For each part, compute and assign values to the local variables defined just before the YOUR CODE HERE comment. These local variables can then be used along with the parameters and any other local variables you compute to answer later parts of the assignment.

When done, turn in your Python file at Blackboard. For the remaining questions, turn in your answers in a separate document at Blackboard.

A used-car buyer is deciding whether to buy a car. The car's quality can be good or bad. The car costs $5,500. Its market value is $6,000 if it is good quality; if not, it will cost $1,000 to bring it up to good quality. You estimate that the car has 60% chance of being of good quality. You are also considering having a certain test performed on the car before buying it. However, the test passing or failing will not tell you definitively whether the car is good or bad. What you do know about the test is that $P(pass \mid good) = 0.85$ and $P(pass \mid bad) = 0.2$.

1. (1 pt) Taking monetary gains/losses as representing your utilities, calculate the expected utility of purchasing the car without having it tested.
2. (2 pts) Draw the decision network that represents the problem that results if you have decided to perform the test. Based on the information provided in the paragraph above, include appropriate and complete utility and conditional probability tables next to each of the nodes in your diagram except the action node. Assume for the utility table that the test has $0 cost. Hint: The structure of this problem is very similar to the decision problem in the slides regarding taking an umbrella based on the forecast.
3. (2 pts) Produce the values for the conditional probability table
   P(Quality | Test)
   Hint: You might want to use Bayes' rule. If you take that approach, note that a normalize() function is provided in the file. You are allowed to use a different approach if you prefer, although I recommend adding comments so that I can follow your reasoning.
4. (1 pt) Calculate the expected utility of purchasing the car
   a. given that it passes the test
   b. given that it fails the test.

   Since the test is being performed in either case, you should reduce each expected utility by the cost of the test, which will be contained in the parameter test_cost (which defaults to 0).
5. (1 pt) Determine the (unconditional) probability that the test will pass. Hint: This can be done in a similar way to one of the Surprise Candy calculations we performed in class. If your code for this is lengthy, you should add explanatory comments.
6. (1 pt) Determine the VPI (value of information) for the test. Hint: Keep in mind that there are two actions open to the car buyer, purchase or not purchase, and that even the no-purchase action will have negative utility if the test has a non-zero cost.
7. (2 pts) Run the program, which will produce results for tests of cost $0, $100, and $200. Based on these results, is it rational to have the test performed if it costs:
   a. $100?
   b. $200?

   Explain your answers. Note: You will find that the VPI value produced for at least one of these costs will violate one of the VPI properties given in the slides. This is because the slides assumed that there was no cost to acquire additional evidence, while we are taking such costs into account.

If you test your code with the problem values provided for the original textbook exercise (code for this is included but commented out in the provided Python file), it should produce the following results:

```
Expected utility of purchase w/o test: 290.00

             | P(Quality | Test)
  Test Result |   Good   |  Bad
     Pass     | 0.8421 | 0.1579
     Fail     | 0.4179 | 0.5821

Expected utility of purchase w/ passing test: 389.47
Expected utility of purchase w/ failing test: 92.54

Probability of passing test: 0.6650

VPI for the test: 0.0000
```

You'll notice that the VPI is 0. This is related to the fact that in this example the expected utility of purchasing the car is positive whether the car passes or fails the test. That is, running the test would be unnecessary because the test result would not affect our decision, so there is no value (additional utility) in the information the test result would provide.

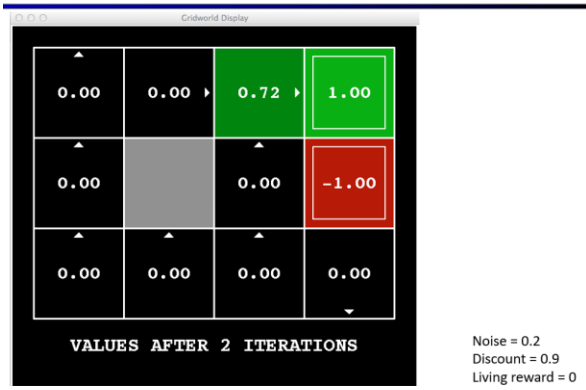
Markov Decision Processes

(2 pts) Recall the 4-by-3 Gridworld that we discussed in class (see Week 10 Markov Decision Processes slides for details). In the transition model for this world, 80% of the time the agent moves in the direction specified by its policy, but the rest of the time it moves at right angles to this direction with equal probability for each of these two directions. If the direction in which it moves would take the agent into a wall, it stays in its current cell (there are walls all around cell (2,2)). Also, once an agent enters one of the two absorbing states, it cannot leave that state.

Another note about the absorbing states is that for k>1, the reward associated such a state is not included in the Bellman equation calculation. That is, at k=1 the rewards for (4,2) and (4,3) were used to initialize the utilities of these states. After that, the states are treated as if they have rewards of 0.

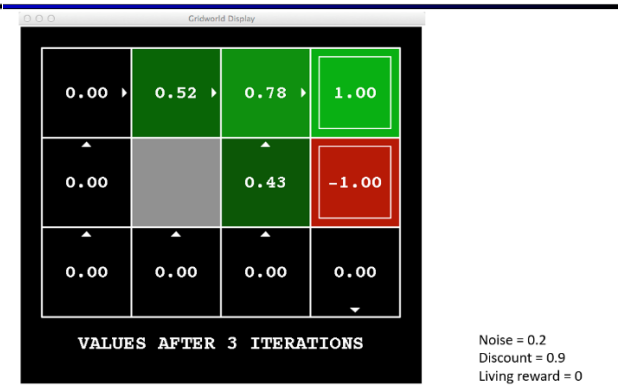
In the Week 10 slides on Value Iteration, at the end of class we saw the following utility values after 2 and 3 steps of iterating the Bellman equations (when the utilities were all initialized to 0):

k=2 $U_{i+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma U_i(s')]$



Noise = 0.2
Discount = 0.9
Living reward = 0

k=3 $U_{i+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma U_i(s')]$



Noise = 0.2
Discount = 0.9
Living reward = 0

Show how the k=3 values of cells (2,3) (the 0.52 value) and (3,2) (the 0.43 value) are produced from the Bellman equation for utility. You can assume that Right is the maximal action for (2,3), but consider both Left and Up actions for (3,2). That is, provide Q values for both the state/action pairs ((3,2),Left) and ((3,2),Up).