CS 520 Final

16:198:520

> You cannot be graded on what you don't write down. This problem is to be completed individually, without any coordination with others. Complete each question to the best of your ability. If you write code to solve one of the problems, you must submit that code along with your final answers. Explain your process and algorithms. Be explicit. Answers, without evidence of the thought and process that led to them, will not earn credit.

Q1 - Sheep Dog Bot

You are taking part in the 2020 Sheep Dog Bot competition. A sheep is released into an 7x7 cell grid, and you have two sheep dog bots under your control. Your goal is to pin the sheep in the lower right corner of the field (position (6,6)):

```
Example Initial State               Sheep is Cornered
[_][_][_][_][_][_][D]     ->     [_][_][_][_][_][_][_]
[_][_][_][_][_][_][_]     ->     [_][_][_][_][_][_][_]
[_][_][_][_][_][_][_]     ->     [_][_][_][_][_][_][_]
[_][_][_][_][_][_][_]     ->     [_][_][_][_][_][_][_]
[S][_][_][_][_][_][_]     ->     [_][_][_][_][_][_][_]
[_][_][_][_][_][_][_]     ->     [_][_][_][_][_][_][D]
[_][_][_][D][_][_][_]     ->     [_][_][_][_][_][D][S]
```

The rules are these:

- You see the entire field, and have complete information to use when directing your dog bots.
- Each round, you may issue commands to your dog bots in the form of where to move, and then the sheep has a chance to move in response. The commands given to each dog bot may differ, but you command them simultaneously.
- Valid commands to a dog bot are 'up', 'down', 'left', 'right', 'hold', and the dog bot will execute the command it is given as long as:
  - the command does not move the dog bot out of the field
  - the dog bot does not end up in the same cell as the sheep (that cell is blocked by the sheep)
  - after the dog bots execute their commands, they will not be in the same cell.
- Each dog can move at most one cell per round (you can issue commands to both dogs each round).
- After the dogs move, the sheep moves according to the following rules:
  - If the sheep has neighbors in the up/down/left/right directions, it will choose uniformly at random among the available neighbors.
  - If the sheep has no free directions to move, it will stay in place.
  - If the sheep is pinned on (6,6) and has no free directions to move, the game is over.
  - The sheep can move at most one square per round, and always moves one square unless it is cornered.
- Your final score is the number of rounds it took to corner the sheep.

Questions:

1) How many possible sheep/dog configurations might the field be in?
2) For a given field configuration C (position of dogs, position of sheep), formulate a mathematical description for $T(C)$, the minimal number expected rounds needed to corner the sheep. *Hint: For what configurations C is $T(C)$ easy to compute?*
3) If you could calculate $T(C)$ for any configuration C, show that you could construct an optimal dog-controlling algorithm for cornering the sheep.
4) Write an algorithm and code to solve for the function T. Is T always finite? What does it mean when T is infinite?
5) Given the initial configuration in the above example, how many rounds do you need (on average) to corner the sheep?
6) What is the worst possible initial state (position of dog bots, sheep), and why? Justify mathematically.
7) You are allowed to place your dog bots anywhere in the field at the start, then the sheep will be placed uniformly at random in one of the remaining unoccupied cells. Where should you place your dog bots initially, and why? Justify mathematically.
8) Do you think better strategies exist than the one you came up with? Justify.

Bonus: Answer the same questions with the following modification: the dog bots win if they can corner the sheep in any of the four corners, but when the sheep moves it is allowed to strategize about which cell to move to, and wants to keep the dog bots from cornering it as long as possible.

Question 2 - Bot Negotiations

You are a robot. One of your parts is a machine that can be in any one of ten possible states:

$$\textbf{New}, \textbf{Used}_1, \textbf{Used}_2, \textbf{Used}_3, \textbf{Used}_4, \textbf{Used}_5, \textbf{Used}_6, \textbf{Used}_7, \textbf{Used}_8, \textbf{Dead}$$

- In states **New** and $\textbf{Used}_1, \ldots, \textbf{Used}_8$, you can **USE** the machine. In states $\textbf{Used}_1, \ldots, \textbf{Used}_8$, **Dead**, you can **REPLACE** the machine.
- In any state, the action **REPLACE** sends the machine to state **New** with probability 1, at cost 255.
- In state **New**, **USE** gives a reward of 101, and transitions to state $\textbf{Used}_1$ with probability 1. For $i = 1, \ldots, 7$, in state $\textbf{Used}_i$, taking action **USE** yields a reward of $100 - 10 * i$, and with probability $0.1 * i$ the machine transitions to state $\textbf{Used}_{i+1}$, otherwise stays in state $\textbf{Used}_i$. In state $\textbf{Used}_8$, action **USE** yields a reward of 20, and transitions to state **Dead** with probability 0.8, otherwise stays in $\textbf{Used}_8$ with probability 0.2.
- The only action available in state **Dead** is **REPLACE**.
- At every time step, all future rewards (and costs) are discounted at a factor of $\beta = 0.9$.

a) For each of the 10 states, what is the optimal utility (long term expected discounted value) available in that state (i.e., $U^*(\textbf{state})$)?

b) What is the optimal policy that gives you this optimal utility - i.e., in each state, what is the best action to take in that state?

c) Instead of buying a new machine, a MachineSellingBot offers you the following option: you could buy a used machine, which had an equal chance of being in $\textbf{Used}_1$ and $\textbf{Used}_2$. Intuitively:

   – If the MachineSellingBot were offering you this option for free, you would never buy a new machine.
   – If the MachineSellingBot were offering you this option at a cost of 255, you would never take this option over buying a new machine.

   What is the highest price for which this used machine option would be the rational choice? i.e., what price should MachineSellingBot be selling this option at?

d) For different values of β (such that $0 < \beta < 1$), the utility or value of being in certain states will change. However, the **optimal policy** may not. Compare the optimal policy for $\beta = 0.1, 0.3, 0.5, 0.7, 0.9, 0.99$, etc. Is there a policy that is optimal for all sufficiently large β? Does this policy make sense? Explain.

Bonus: The cost of a new machine is 255*. What is the (long term discounted) value of a new machine? Determine the break-even cost of a new machine - the price above which you are operating at a net loss, and below which you are operating at a net gain, i.e., when does a new machine become so expensive this game isn't even worth playing?*