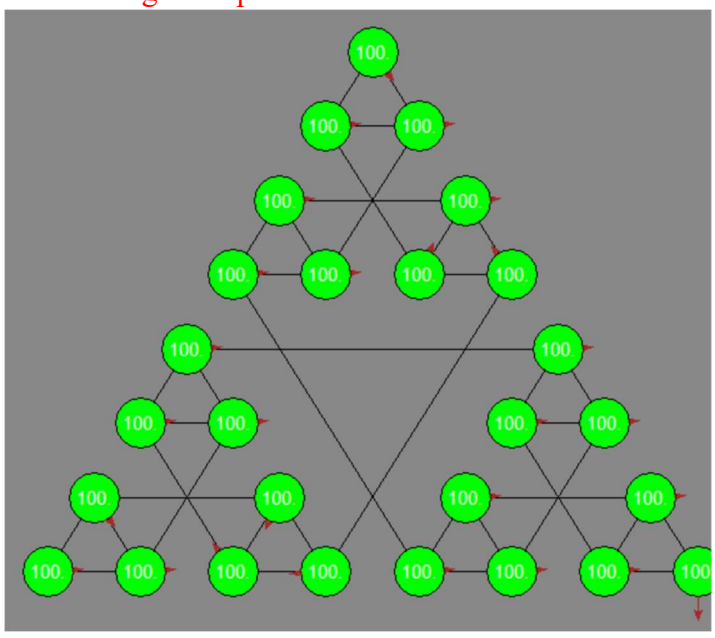
Assignment 5 Report for Part A
Jia-Jia (Jay) Lin
CSE 340 A, Winter 2020
February 15, 2020

## Part A: Value Iteration

**1a. How many iterations of VI are required to turn 1/3 of the states green? (i.e., get their expected utility values to 100).**
For 9 out of 27 states to turn green (value = 100), it took **4 iterations of VI.**

**1b. How many iterations of VI are required to get all the states, including the start state, to 100?**
8 iterations.

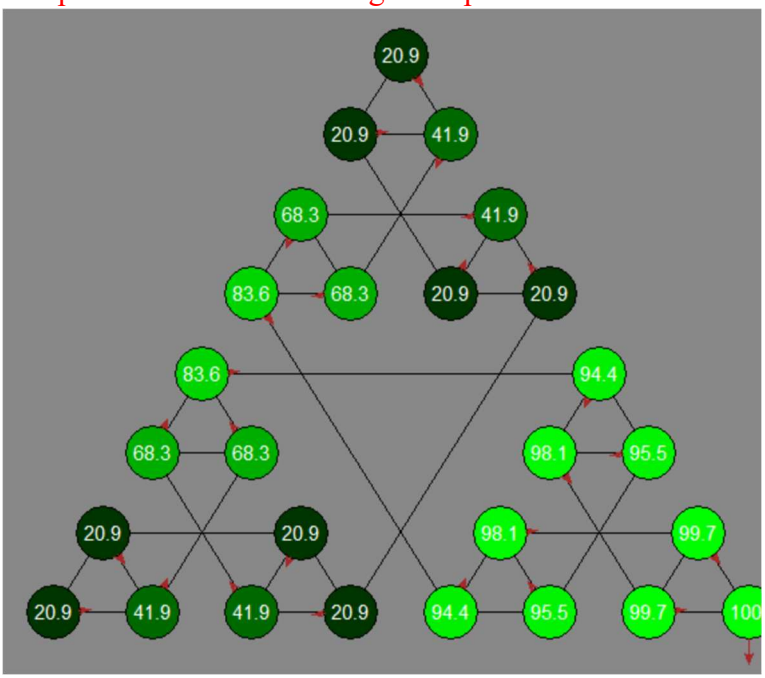
**1c. From the Value Iteration menu, select "Show Policy from VI". (The policy at each state is indicated by the outgoing red arrowhead. If the suggested action is illegal, there could still be a legal state transition due to noise, but the action could also result in no change of state.) Describe this policy. Is it a good policy? Explain.**
This policy is not good enough for the agent because many states point towards an illegal move and do not generally follow the golden path.




**Repeat the above setup except for 20% noise.**
**2a. How many iterations are required for the start state to receive a nonzero value.**
8 iterations.

**2c. At this point, view the policy from VI as before. Is it a good policy? Explain.**

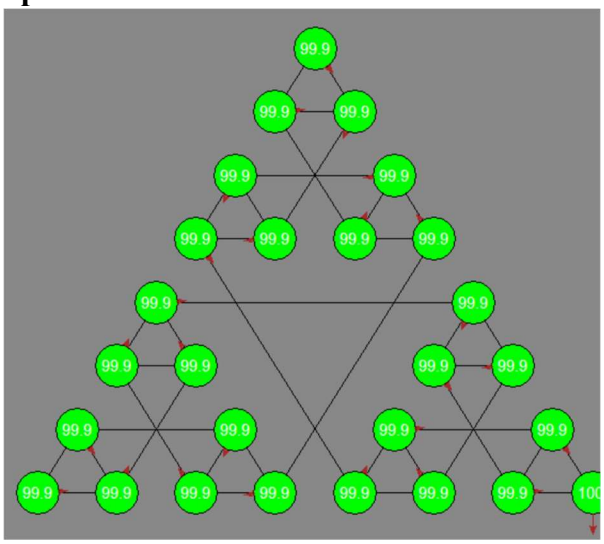
This policy, considering noise, is a good policy because the arrows point toward the goal state in a more straightforward path and is closer to the golden path.




**2d. Run additional VI steps to find out how many iterations are required for VI to converge. How many is it?**

56 iterations total.

**2e. After convergence, examine the computed best policy once again. Has it changed? If so, how? If not, why not? Explain.**



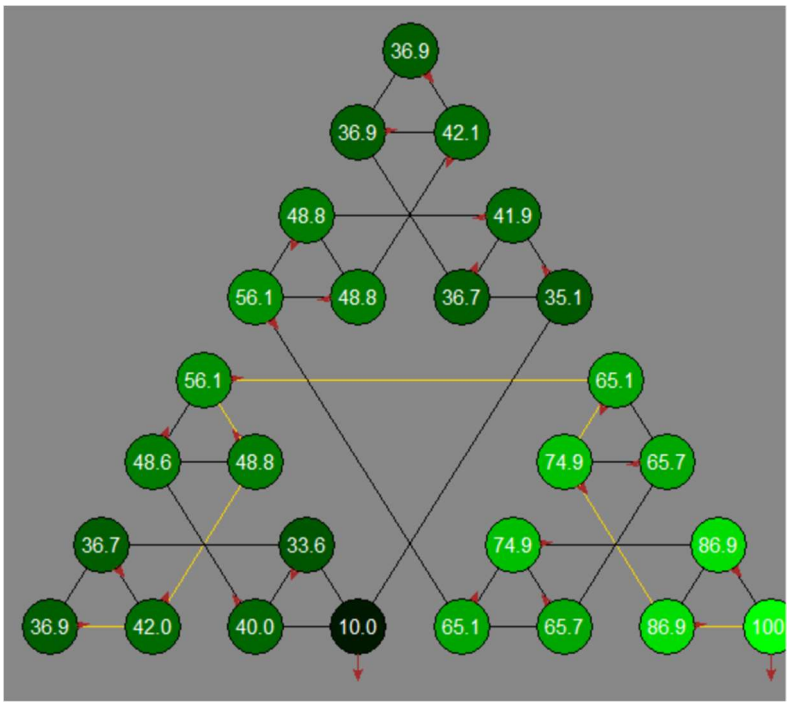

**Repeat the above setup, including 20% noise but with 2 goals and discount = 0.5.**

**3a. Run Value Iteration until convergence. What does the policy indicate? What value does the start state have?**

The value iteration converged after iteration 23. The value of the start state is 0.82. This policy generally DOES NOT point in the same path as the golden path solution. The policy suggests that the agent should move the ring and then “Exit” when reaching the state where each pillar has one ring, which is not optimal.

**3b. Reset the values to 0, change the discount to 0.9 and rerun Value Iteration until convergence. What does the policy indicate now? What value does the start state have?**

The policy now indicates that the agent should follow the golden path. The value of the start state is 36.9.




**Now try simulating the agent following the computed policy. Using the "VI Agent" menu, select "Reset state to s0". Then select "Perform 10 actions". The software should show the motion of the agent taking the actions shown in the policy. Since the current setup has 20% noise, you may see the agent deviate from the implied plan. Run this simulation 10 times, observing the agent closely.**

**4a. In how many of these simulation runs did the agent ever go off the plan?**

1st simulation: Deviated
2nd: Did not deviate, goal
3rd: Deviate, 4 steps away from goal
4th: Deviate, 3 steps away from goal
5th: Did not deviate, goal
6th: Did not deviate, goal
7th: Deviate, 1 step away from goal
8th: Did not deviate, goal
9th: Deviate, 3 steps away from goal

10th: Did not deviate, goal
Out of ten simulations, the agent went off-plan in **5 simulation runs**.

**4b. In how many of these simulation runs did the agent arrive in the goal state (at the end of the golden path)?**
5 simulation runs.

**4c. For each run in which the agent did not make it to the goal in 10 steps, how many steps away from the goal was it?**
See my answer in 4a.

**4d. Are there parts of the state space that seemed never to be visited by the agent? If so, where (roughly)?**
The top triangle portion of the state space seems to have never been traversed by the agent.

**Overall reflections.**
**5a. Since it is having a good policy that is most important to the agent, is it essential that the values of the states have converged?**
Yes, if the values of the states have not converged then the agent would not be able to easily choose an optimal action that leads to the correct solution.

**5b. If the agent were to have to learn the values of states by exploring the space, rather than computing with the Value Iteration algorithm, and if getting accurate values requires re-visiting states a lot, how important would it be that all states be visited a lot?**
It would be important that all states be visited a lot since getting accurate values requires visiting the state many times. This way, the agent can learn the optimal policy.

## Optional Part B: Greedy Q-Learning

Describe the work that you did for Part B. Explain how epsilon-greedy Q-learning compared in its fixed-epsilon vs custom-adjustment versions.

- I implemented handle_transition which fills the Q_VALUES dictionary with the q values for each state and action pair, utilizing the Q-Learning equation from the lecture slides.
- In choose_next_action, I defined what the policy should suggest during each given state s. For example, if the agent is already in a Terminal State, then they should take the action “Exit”. Otherwise, compute a new q-value for the previous state and action. I also account for EPSILON in this function. Epsilon is defined as the increasing probability that random actions are chosen but as training progresses, the agent should favor choices that maximize q values. The randomization also acts like how a real player would act if they don’t know the best plan of actions. Also, the policy won’t get potentially stuck at a state as opposed to if they always choose the direction with optimal q value.

- Epsilon-greedy Q-learning, with the fixed epsilon, means that the agent should take the same probability of choosing a random action or going the optimal path throughout each choosing of an action.
    - Meanwhile, with the change CUSTOM_ALPHA, the threshold for choosing a random action becomes higher and higher (more likely to choose optimal over random over time).