Course Code: CS 5180

Course: Reinforcement Learning and Sequential Decision Making

Name: Pavan Rathnakar Shetty

Please find the entire submission Canvas and in https://github.com/Pavan-r-shetty/Reinforcement-Learning-2023.git as well

EX0 Assignment Submission

1)

Please find the simulate function in Line 42 onwards in the ex0.py file

2)

Please find the manual policy in Line 124 onwards in the ex0.py file

To run this, please uncomment line 295 and comment line 299

```
Line 295: (agent(steps=100, trials=1, policy=manual_policy)  #uncomment this to
run manual policy for Q1, comment this while running Q2, Q3 and Q4)

Line 299: (# main() #uncomment this to run Random, Better and Worse policy for
Q2, Q3 and Q4, comment this while running Q1)
```

3)

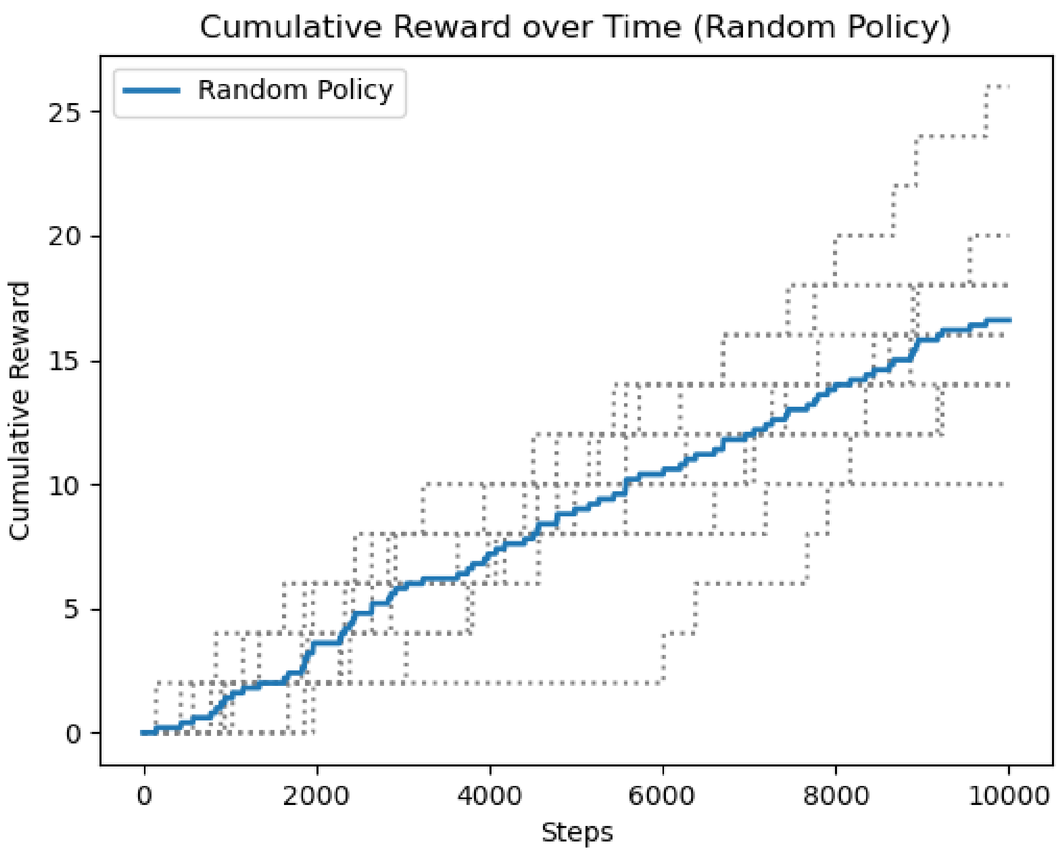
Please find the random policy from Line 197 onwards in the ex0.py file

To run this, please comment line 295 and uncomment line 299

```
Line 295: (agent(steps=100, trials=1, policy=manual_policy)  #uncomment this to
run manual policy for Q1, comment this while running Q2, Q3 and Q4)

Line 299: (# main() #uncomment this to run Random, Better and Worse policy for
Q2, Q3 and Q4, comment this while running Q1)
```

Plot:



Written: How do you think this compares with your manual policy? (You do not have to run your manual

policy for 104 steps!) What are some reasons for the difference in performance?

Answer:

- Decision Making Process:
    - Random Policy: The policy makes decisions out of random calls and doesn't account for environment's state, previous actions. This will lead to wandering around without clear direction, hitting walls and moving away from the goal.
    - Manual Policy: When user moves manually, the decision of the action is based on current state, previous state memory and goal state. Users can avoid mistakes as the goal is visualized along with directional decisions.
- Efficiency:

    - Random Policy: Due to luck, it may perform better, but over several iterations, it is inefficient since there is no strategy involved in taking the right direction.
    - Manual Policy: A person can strategize and often makes decisions to approach the goal. While users can make mistakes or take suboptimal paths, directional decision-making usually results in a better performance compared to a random approach.
- Learning & Adaptability:
    - Random Policy: It doesn't adapt or learn. It will repeat the same actions irrespective of the outcome.
    - Manual Policy: Users can adapt and learn from past mistakes. If a particular path leads to a dead-end or takes longer, a person might choose a different path in the subsequent attempts.
- Exploration vs. Exploitation:
    - Random Policy: It continuously explores without any sense of exploiting known good paths. While this continuous exploration might be useful in some scenarios to discover unknown rewards, it's not efficient when the goal is known.
    - Manual Policy: Users tend to balance exploration and exploitation based on their knowledge. Once a good path is known, they are more likely to exploit it rather than continue exploring. Approach towards safe choices biases the decision.

In summary, while the random policy can be seen as a baseline or naive approach, the manual policy, driven by user intuition and adaptability, usually performs better, especially in well-defined environments like the Four Rooms problem. The primary reasons for the difference in performance are user adaptability, strategic planning, and the ability to learn from past mistakes. However, it's worth noting that for a human, manually deciding actions for a very long time (like 10^4 steps) can be exhausting, and performance may deteriorate over time due to fatigue or loss of attention/focus.

4)

Please find the worse policy from Line 212 onwards in the ex0.py file

Please find the better policy from Line 227 onwards in the ex0.py file

To run this, please comment line 295 and uncomment line 299

```
Line 295: (agent(steps=100, trials=1, policy=manual_policy)  #uncomment this to
run manual policy for Q1, comment this while running Q2, Q3 and Q4)

Line 299: (# main() #uncomment this to run Random, Better and Worse policy for
Q2, Q3 and Q4, comment this while running Q1)
```

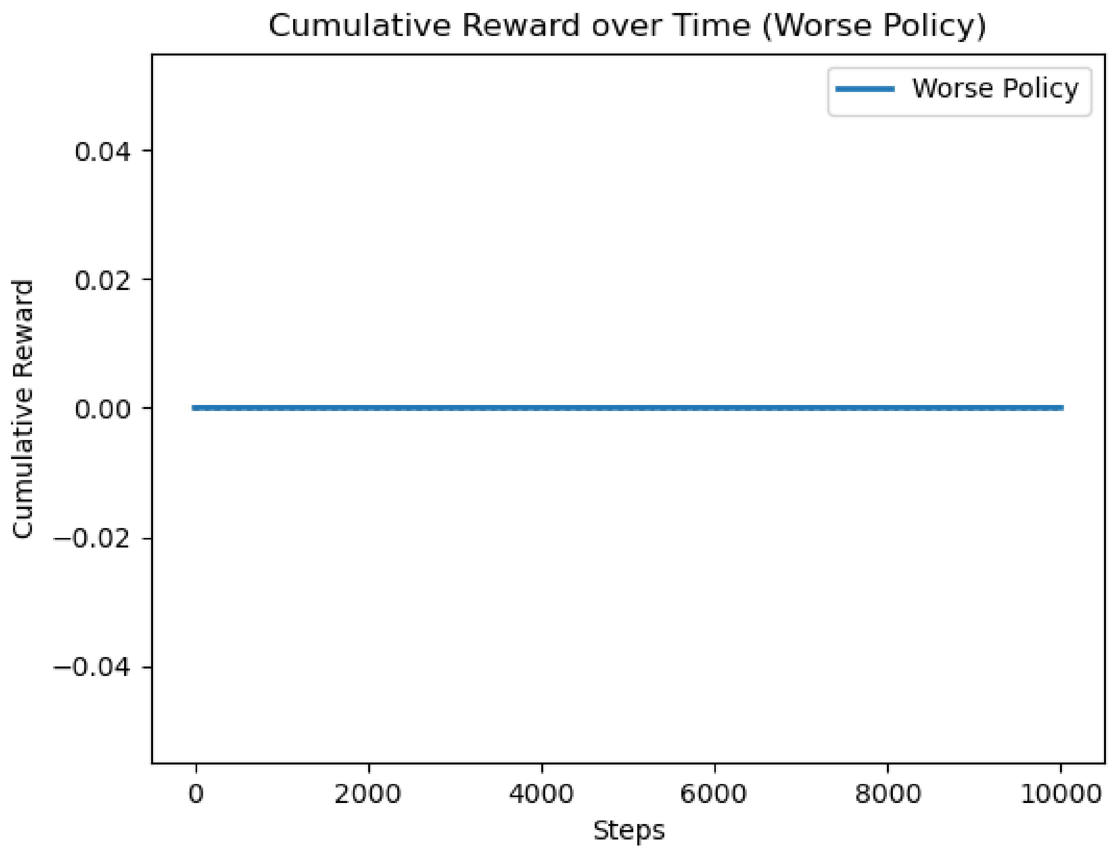
Written: Describe the strategy each policy uses, and why that leads to generally worse/better performance.

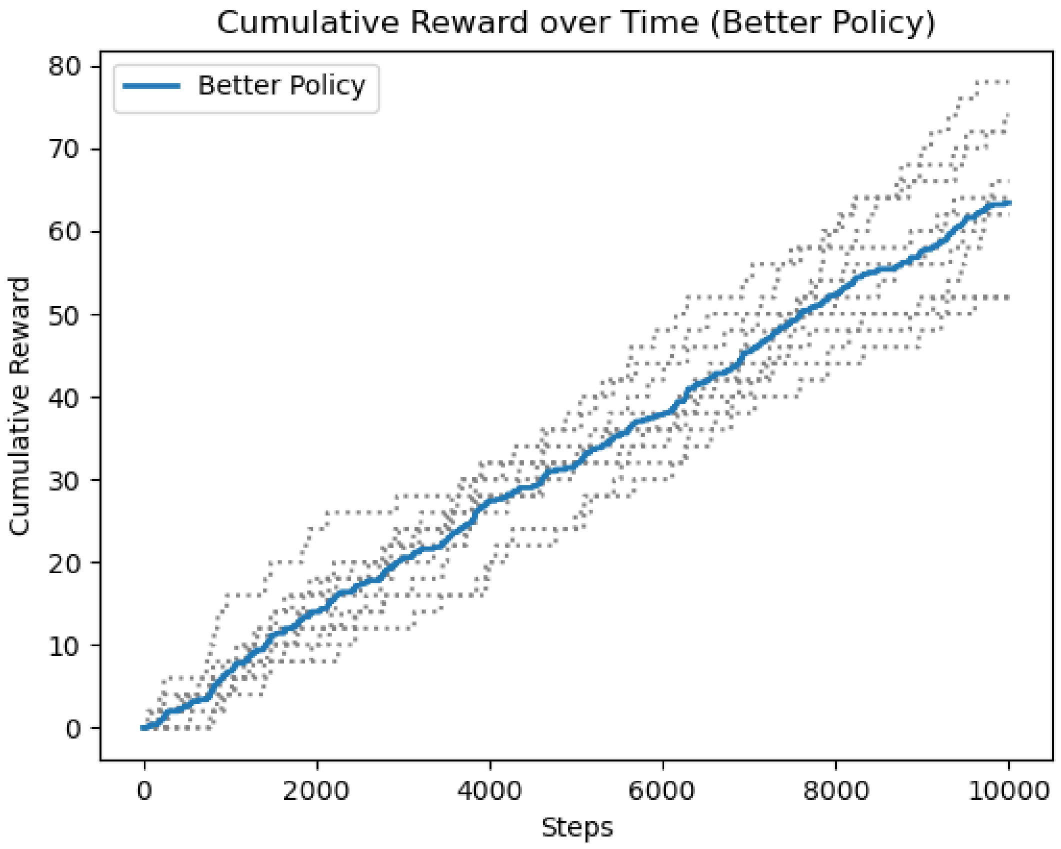
Answer:

- worse_policy:
  - Strategy: This policy always selects the UP action regardless of the state in which it is in.
  - Description: Since it's always taking the UP action, it lacks any form of exploration. It can get stuck easily in scenarios where the UP action doesn't lead to a new state or towards the goal. This will especially be detrimental in cases where the agent starts or gets stuck under a wall and keeps trying to move up. The agent will not be able to explore other parts of the environment, and hence, it will miss the opportunity to receive rewards from other reachable states. (we can hardcode the action to any direction, results in a similar worse policy)
  - Performance Implications: Due to its lack of adaptability and exploration, worse_policy will generally exhibit poor performance, especially when the optimal action isn't continuously moving up. It is designed to take bad decision just to show random and better policy is better.
- better_policy:
  - Strategy: This policy prioritizes the UP and RIGHT actions 80% of the time and the LEFT and DOWN actions 20% of the time. (you can change UP and RIGHT priority to a number more than 60% to result in similar reward)
  - Description: The motivation behind this policy is to balance exploration with a slight bias towards moving in the general direction of the goal (assuming the goal is towards the upper right side). By favoring the UP and RIGHT actions, the agent is more likely to head towards the goal. However, by occasionally selecting the LEFT and DOWN actions, the agent can also explore other parts of the environment and possibly find shortcuts or avoid getting stuck around walls and other non-goal states.
  - Performance Implications: The better_policy is designed to be a balance between exploration and exploitation. By leaning more towards the actions that move the agent closer to the goal, it's likely to achieve higher rewards over time compared to purely random actions. However, it still retains a degree of randomness to ensure the agent doesn't get stuck and can adapt to various states. Priority action can be played with numbers 60% - 90%, so that we can tune down the bias and increase randomness or exploration.

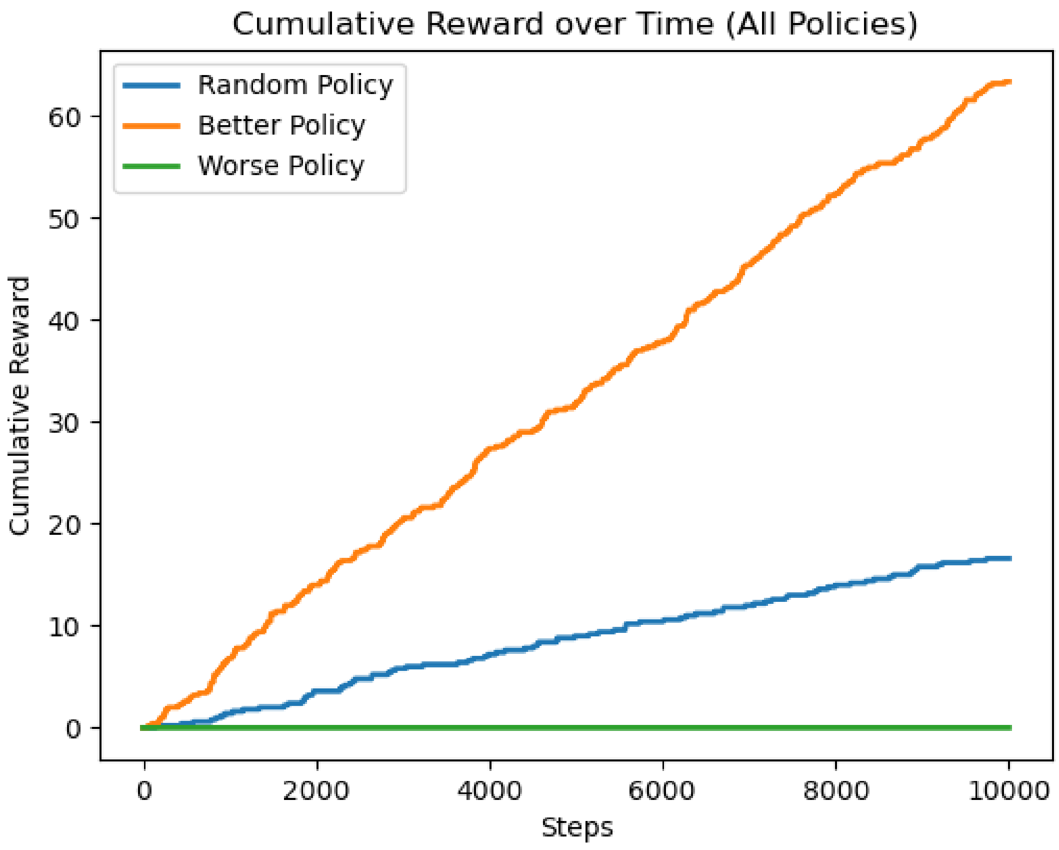
In summary, the key difference between the two policies is the balance of exploration vs. exploitation. The worse_policy exploits a single action without any exploration, leading to suboptimal performance. In contrast, the better_policy attempts to find a balance of exploration vs. exploitation, leading to generally better performance.

Plot:


Cumulative Reward over Time (Worse Policy)
Worse Policy
0.04
0.02
0.00
−0.02
−0.04
Cumulative Reward
0
2000
4000
6000
8000
10000
Steps



Cumulative Reward over Time (Better Policy)
Better Policy
Cumulative Reward
Steps
0
2000
4000
6000
8000
10000
0
10
20
30
40
50
60
70
80


Cumulative Reward over Time (All Policies)
Random Policy
Better Policy
Worse Policy
Cumulative Reward
0
10
20
30
40
50
60
Steps
0
2000
4000
6000
8000
10000