CS 5180 Exercise 2
Harin Kumar Nallaguntla
Study Partner: Gopi Sai Nath Mamindlapalli

1. a. State Space (S): The state space consists of (x, y) coordinates. x ranges from 0 to 10, and y ranges from 0 to 10. The state space is defined as: S = {(x, y) | 0 ≤ x ≤ 10, 0 ≤ y ≤ 10}
Action Space (A): The action space includes four feasible actions: LEFT, DOWN, RIGHT, UP. All four actions can be taken at any valid state. The action space is defined as: A = {LEFT, DOWN, RIGHT, UP}

b. In this MDP, there are 104 valid states where the agent can be, and for each state-action pair, there are 4 feasible actions (LEFT, DOWN, RIGHT, UP). Each of these actions leads to 3 possible next states: 1 with a probability of 0.9 (the correct direction) and 2 with a probability of 0.05 each (perpendicular directions). This results in a total of 12 possible state transitions for each state-action pair. Therefore, the conditional probability table for this MDP contains 1248 (104 * 4 * 3) rows, reflecting the 4 available actions and 3 possible next states for each action. And, for every state which are surrounded by a wall on two adjacent sides, we subtract 2 from the total number of rows, because when a state is surrounded by a wall on two of its four sides, two of three of its next state are the same. There are 16 such states. So, there are total 1216 (1248 – 16 * 2) total number of non-zero rows.

2. a. Episodic case: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$, where T is the time step at which the episode end. Then the return is $-\gamma^{T-t-1}$

Continuous case: $G_t = -\gamma^K$, where K is the time step at which the failure happens

b. Sparse Rewards: In this scenario, the reward function provides very infrequent positive feedback, as the agent only receives a reward of +1 upon successfully escaping the maze. This sparsity of rewards can pose a significant challenge during the learning process, as the agent may go through extended periods without any positive reinforcement, making it difficult for the agent to effectively learn and adjust its behavior.

Lack of Intermediate Feedback: The absence of intermediate rewards or penalties means that the agent lacks guidance on its actions while navigating the maze. It doesn't have feedback on the quality of its actions, making it uncertain about which actions are better or worse. Consequently, the agent may require a considerable amount of time to randomly discover the correct sequence of actions that leads to its escape, as it lacks informative cues to help it understand and learn from its experiences within the maze.

3. a. To calculate the discounted returns G0, G1, ..., G5 with a discount factor γ = 0.5, we can work backward using the following formula: G_t = R_(t+1) + γ * G_(t+1)

Starting from time t = T and working backward: **G5 = 0**
For G4: G4 = R5 + γ * G5
G4 = 2

For G3: G3 = R4 + γ * G4
G3 = 3 + 0.5 * 2
G3 = 4

For G2: G2 = R3 + γ * G3
G2 = 6 + 0.5 * 4
G2 = 8

For G1: G1 = R2 + γ * G2
G1 = 2 + 0.5 * 8
G1 = 6

For G0:

$G0 = R1 + \gamma * G1$
$G0 = (-1) + 0.5 * 6$
$G0 = 2$

b. $G0 = 2 + 0.9 *$ *$(7 + 0.9 * (7 + 0.9 * (7 + ...)))$ - (italic terms are in geometric progression)*
$G0 = 2 + 0.9 * (7 / (1 – 0.9))$
$G0 = 65$

$G1 =$ *$7 + 0.9 * (7 + 0.9 * (7 + ...))$*
$G1 = 7 / (1 – 0.9)$
$G1 = 70$

4. "Up" Action:
- When the agent takes the "Up" action, it moves to the first cell of the first row, receiving a reward of +50.
- Subsequently, it enters the first row and collects a -1 reward in each cell.
- The cumulative expected reward for the "Up" action can be expressed as:
$V(up) = 50 + \gamma * (-1 + \gamma * (-1 + \gamma * (-1 + \ldots \text{ upto 100 terms}))) = 50 – (\gamma * (1 – \gamma^{100}) / (1 – \gamma))$

"Down" Action:
- Choosing the "Down" action leads the agent to move to the first cell of the third row, where it receives a reward of -50.
- After that, it enters the third row and accumulates a -1 reward in each cell, except for the last cell, which offers a +1 reward.
- The cumulative expected reward for the "Down" action can be expressed as:
$V(down) = -50 + \gamma * (1 + \gamma * (1 + \gamma * (1 + \ldots \text{ upto 100 terms}))) = -50 + (\gamma * (1 – \gamma^{100}) / (1 – \gamma))$

To choose **up**: V(up) > V(down) -> **$50 – (\gamma * (1 – \gamma^{100}) / (1 – \gamma)) > -50 + (\gamma * (1 – \gamma^{100}) / (1 – \gamma))$**
To choose **down**: V(down) > V(up) -> **$-50 + (\gamma * (1 – \gamma^{100}) / (1 – \gamma)) > 50 – (\gamma * (1 – \gamma^{100}) / (1 – \gamma))$**

5. a. The absolute values of rewards are not important, only the intervals or differences between them matter. This is because the values of states and actions are learned relative to each other and depend on the relative differences in expected returns. The sign and magnitude of individual rewards do not affect the relative values. Let's prove that adding a constant c to all rewards adds a constant v_c to the values of all states using
Equation 3.8: $G_t = R_{t+1} + \gamma * R_{t+2} + \gamma^2 * R_{t+3} + \ldots$

We can rewrite this equation for the state-value function as follows:
$V(s_t) = E[G_t | s_t]$

Now, let's consider the effect of adding a constant c to all rewards:
- Original sequence of rewards: $R_{t+1}, R_{t+2}, R_{t+3}, \ldots$
- New sequence of rewards with constant c added: $R_{t+1} + c, R_{t+2} + c, R_{t+3} + c, \ldots$

Now, calculate the new state-value function using the updated reward sequence:
$V'(s_t) = E[G_t | s_t]$ (with the new rewards)
Using Equation 3.8 with the new rewards:
$V'(s_t) = (R_{t+1} + c) + \gamma * (R_{t+2} + c) + \gamma^2 * (R_{t+3} + c) + \ldots$
$V'(s_t) = (R_{t+1} + \gamma * R_{t+2} + \gamma^2 * R_{t+3} + \ldots) + c + \gamma * c + \gamma^2 * c + \ldots$
$V'(s_t) = V(s_t) + (c / (1 - \gamma))$

So, adding a constant c to all the rewards adds a constant $v_c = (c / (1 - \gamma))$ to the values of all states. This constant v_c is independent of the specific states and policies and depends only on the constant c and the discount factor γ. It does not affect the relative values of any states under any policies, as the additional constant v_c is the same for all states.

b. Adding a constant, c, to all rewards in an episodic task, like maze running, can significantly impact the task, unlike in continuing tasks where relative values remain the same. Episodic tasks have a natural ending point, such as completing an episode with a specific objective. Rewards in episodic tasks are designed to guide the agent's progress toward this objective. Adding c uniformly shifts the entire reward scale, altering the task's success criteria and the agent's perception of success. This shift doesn't change relative reward differences within an episode but shifts the absolute reward values, which can profoundly affect the agent's behavior and task perception.

Here's an example:
Consider a maze-running task where the original rewards are as follows:

- Finding the exit: +10
- Hitting walls: -1
- All other states: 0

In this scenario, the agent's goal is to maximize its total reward within each episode.
Now, if you add a constant c = 5 to all rewards in this episodic task, the new reward structure becomes:

- Finding the exit: +15
- Hitting walls: +4
- All other states: +5

This changes the agent's perception of the task. The agent might now prioritize hitting walls over finding the exit because hitting walls yields higher rewards. The agent may not focus on the original task's objective (finding the exit) as it did before the reward modification.

6. a. $V_\pi(C)$: Value function of center state
$V_\pi(N)$: Value function of north state
$V_\pi(S)$: Value function of south state
$V_\pi(E)$: Value function of east state
$V_\pi(W)$: Value function of west state

$V_\pi(C) = 0.25 * (V_\pi(N) + V_\pi(S) + V_\pi(E) + V_\pi(W))$
$V_\pi(C) = 0.25 * (2.3 + 0.4 + 0.7 – 0.4)$
$V_\pi(C) = 0.75$

So, numerically calculating the right-hand side of the Bellman equation for the center state yields $V_\pi(C) = 0.75$, which matches the value in the figure, $V_\pi(C) = 0.7$, to within one decimal place. This verifies that the Bellman equation holds for the center state with respect to its neighboring states.

b. $V_\pi(C) = 0.25 * (V_\pi(N) + V_\pi(S) + V_\pi(E) + V_\pi(W))$
$V_\pi(C) = 0.25 * (19.8 + 16 + 16 + 19.8)$
$V_\pi(C) = 17.9$

numerically calculating the right-hand side of the Bellman equation for the center state yields $V_\pi(C) = 17.9$, which closely matches the value in the figure, $V_\pi(C) = 17.8$. This verifies that the Bellman equation holds for the center state with respect to its neighboring states.

7. a. Guessing: As we are using an equiprobable random policy, we can say the agent ends up at L half the time and at R the other half of the time, so V(A) is an average of the rewards at both L and R, which is 0.5

Verifying: Bellman equation is:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

Here, we are using an equiprobable random policy

V(L) = 0 and V(R) = 1
Now,

$$V(A) = \pi(L|A)(0 + V(L)) + \pi(R|A)(0 + V(R))$$

V(A) = 0.5 * (0 + 0) + 0.5 * (0 + 1)
V(A) = 0.5

Hence verified.

b. Guessing: The value function of states drop as they move away from the Right terminal state, in geometric progression. So, if V(E) = 0.5, then V(D) = 0.25, V(C) = 0.5^3, V(B) = 0.5^4, V(A) = 0.5^5

Verifying: using Bellman equation,
V(E) = 0.5 * (V(D) + V(R)) = 0.5 (0 + 1) = 0.5
similarly,
V(D) = 0.5 * (0 + 0.5) = 0.5^2
V(C) = 0.5 * (0 + 0.5^2) = 0.5^3
V(B) = 0.5 * (0 + 0.5^3) = 0.5^4
V(A) = 0.5 * (0 + 0.5^4) = 0.5^5

c. $V(S_n) = 0.5^n$, where S_n is nth state from R state

8. a.

$$V(high) = \pi(search|high)[\alpha(r_{search}+\gamma V(high))+(1-\alpha)(r_{search}+\gamma V(low))]+\pi(wait|high)(r_{wait}+\gamma V(high))$$

$$\begin{aligned}\mathrm{V(low)} = {} & \pi(search|low)[(1-\beta)(-3+\gamma V(high)) + \beta(r_{search} + \gamma V(low))] + \\ & \pi(wait|low)(r_{wait} + \gamma V(low)) + \pi(recharge|low)(0 + \gamma V(high))\end{aligned}$$

b. By substituting the given values, we get two equations:
0.685 * V(high) = 0.135 * V(low) + 5
0.55 * V(low) = 0.45 * V(high) + 1.5

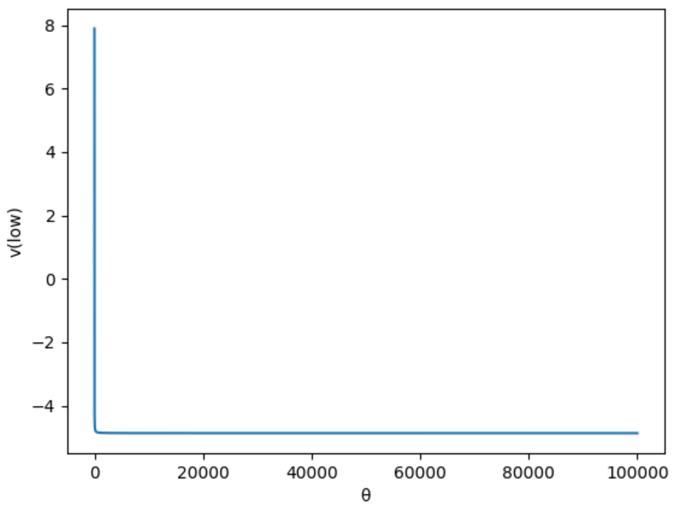
Solving, we get **V(high) = 9.33 and V(low) = 10.36**

c. By substituting the given values, we get two equations:
V(high) = (0.135 * V(low) + 5) / 0.685 **(1)**
V(low) = θ * (3+ 0.9 * V(low)) + (1 - θ) * (0.9 * V(high)) **(2)**

By substituting **(2)** in **(1)**, we get
V(low) = (6.55 - 3.55 * θ) / (0.83 + 0.73 * θ)

Choosing the value of θ which maximizes the value of V(low) maximizes V(high) as well

The graphs of V(low) vs θ is:



The maximum value of V(low) occurs at θ = 0 as we can see from the graph, **at θ = 0**, **V(low) = 7.89**
Substituting, **V(high) = 8.85**

9. a.

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$

b.

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

c.

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a')]$$