Question 1:

1. We select a random state from the available states = (0 to 24)

2. For each episode record the sequence of states and rewards until it reaches the terminal state.

3. Define the G(s), the discounted reward from the selected random state to the terminal state.

$$G(s) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots + \gamma^{k-1} r_k$$

4. Now we create a value matrix V(s) for each episode. We keep track of the cumulative number of visits and cumulative total discounted reward G(s).

5. For fir first-visit algorithm, Matrices N(s) and S(s) are persistent across episodes, calculated as below, for the first time we enter that state each episode.

Increment counter of visits: N(s) = N(s) +1

Increment total return S(s) = S(s) + G(s)

6. At the end of all the episodes, we will estimate the value matrix as below

V(s) = S(s) / N(s)

7. We run enough number of episodes until all the states were represented in the Value matrix.

8. We selected this method as per the Lecture.

Question 2:

1. We follow the same convergence method except the fact that for Monte-Carlo every-visit, we compute the matrices as below, every time we enter that state each episode.

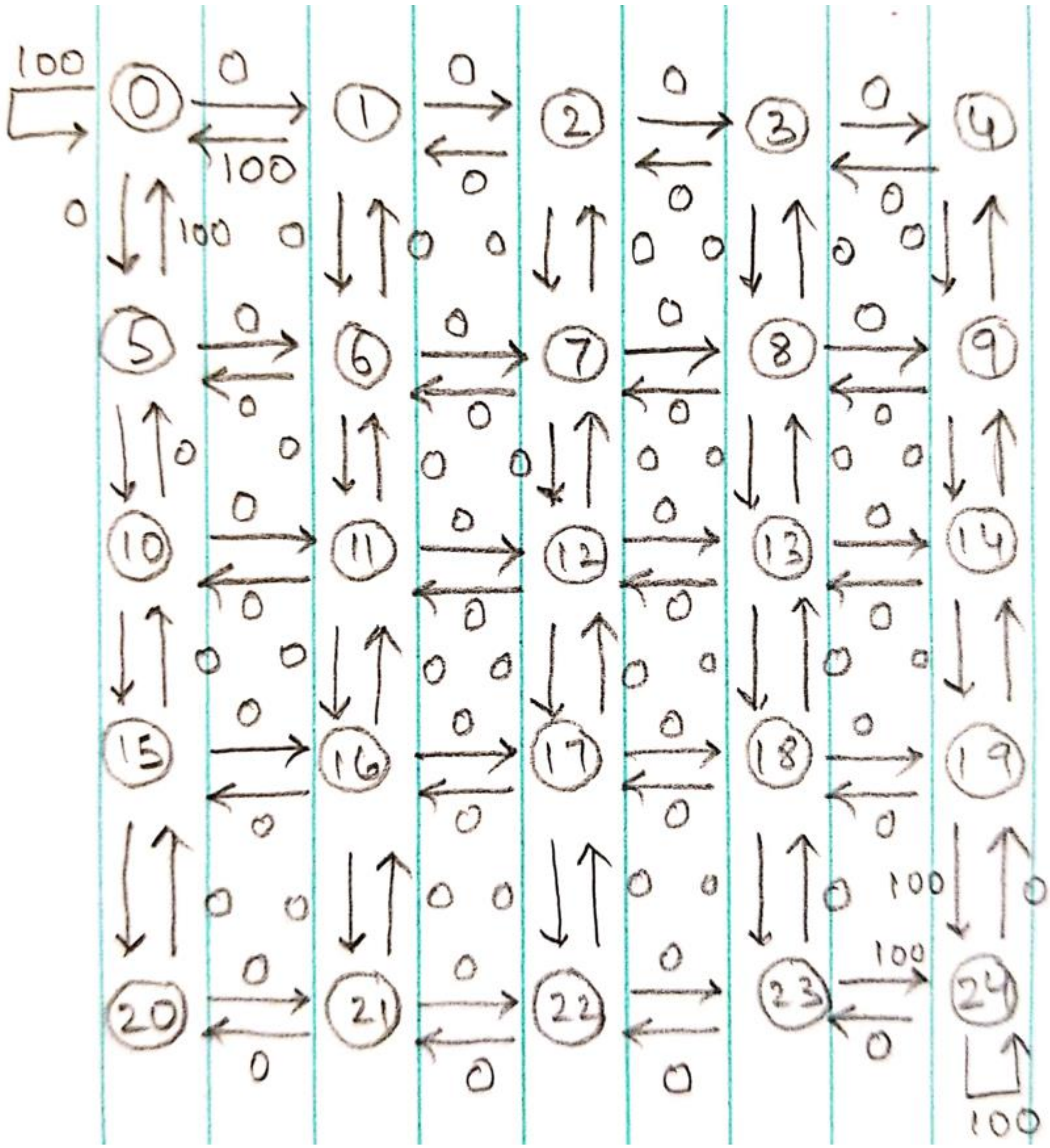
Increment counter of visits: N(s) = N(s) +1

Increment total return S(s) = S(s) + G(s)

2. Part-1 and Part-2 converged in the same number of episodes.

Question 3:

State diagram for Q-Learning



Question 4:

1. We set the gamma = 0.9 and define the rewards matrix.

2. Initialize the Q-matrix.

3. We run a constant number of iterations (say > 500). As we have only 25 states. For each episode,

4. Select a random initial state and perform below states until we reach the terminal state.

5. Randomly select one among all possible actions for the current state

6. Use this possible action and go to the new state.

7. Get maximum Q-value for this next state based on all possible actions and update the Q-value matrix for the selected state.

Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]

8. Algorithm converges after 100 iterations.

Question 5: Below Optimal policies were followed for state = 7

State 7 ⟶ State 2 ⟶ State 1 ⟶ State 0 (terminal state)

State 7 ⟶ State 6 ⟶ State 1 ⟶ State0 (terminal state)

State 7 ⟶ State 6 ⟶ State 5 ⟶ State0 (terminal state)

Question 6:

We set the gamma = 0.9 and define the rewards matrix.

2. Initialize the Q-matrix.

3. We run a constant number of iterations (say >= 100). As we have only 25 states. For each episode,

4. Select a random initial state and perform below states until we reach the terminal state.

5. Select the action with highest value in the Q-matrix.

6. Use this possible action and go to the new state.

7. Get maximum Q-value for this next state based on all possible actions and update the Q-value matrix for the selected state.

Question 7:

State 7 ⟶ State 2 ⟶ State 1 ⟶ State 0 (terminal state)

Question 8:

SARSA converged faster than Q-learning, as it is takes more greedy approach by traversing less paths.