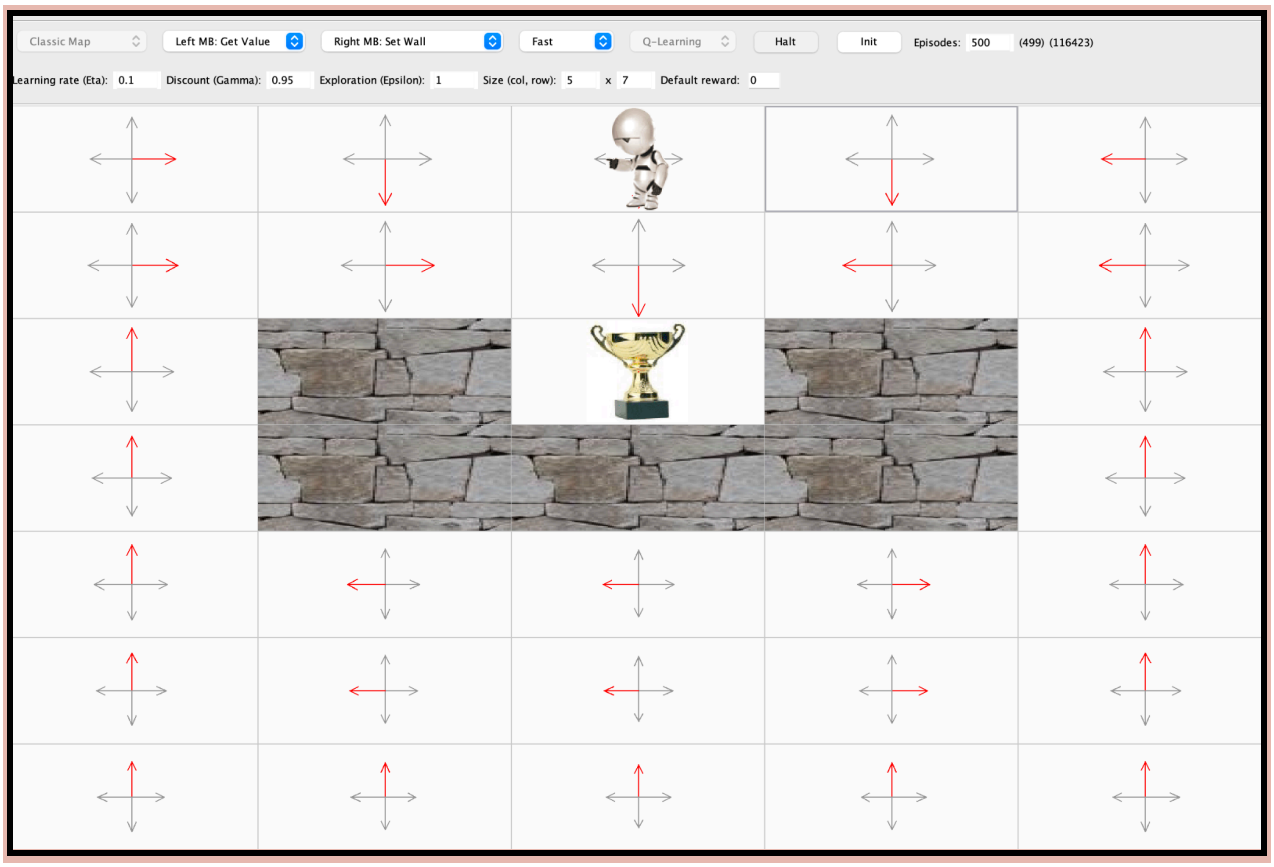


Question 1: After the arrows have converged, some states will have longer arrows than others (i.e., larger Q-values). Why is this so?
<It's because of the gamma factor.>

Closer states have higher values because the agent receives the reward sooner. Some states have longer arrows because the gamma value discounts future rewards. So the impact of the final reward diminishes over time.

Near the goal, the Q-values are higher because the agent expects to reach the reward soon. These high Q-values lead to longer arrows, guiding the agent toward the goal quickly.
As we move away from the goal, the arrows get shorter but still point toward the goal.
The walls (obstacles) force the agent to take longer paths, further reducing Q-values in those areas.

Question 2: In the first few episodes, only the states closest to the goal will have their Q-values increased (even though the agent may start far from the goal). Explain why.

Again, since the gamma diminishes the rewards over time, closer states get higher rewards sooner, and farther states take more episodes to get longer arrows.

Certain arrows (not necessarily in the same state) should converge towards the same lengths. Why is this the case? Give an example of two state-action pairs that get equal Q-values.

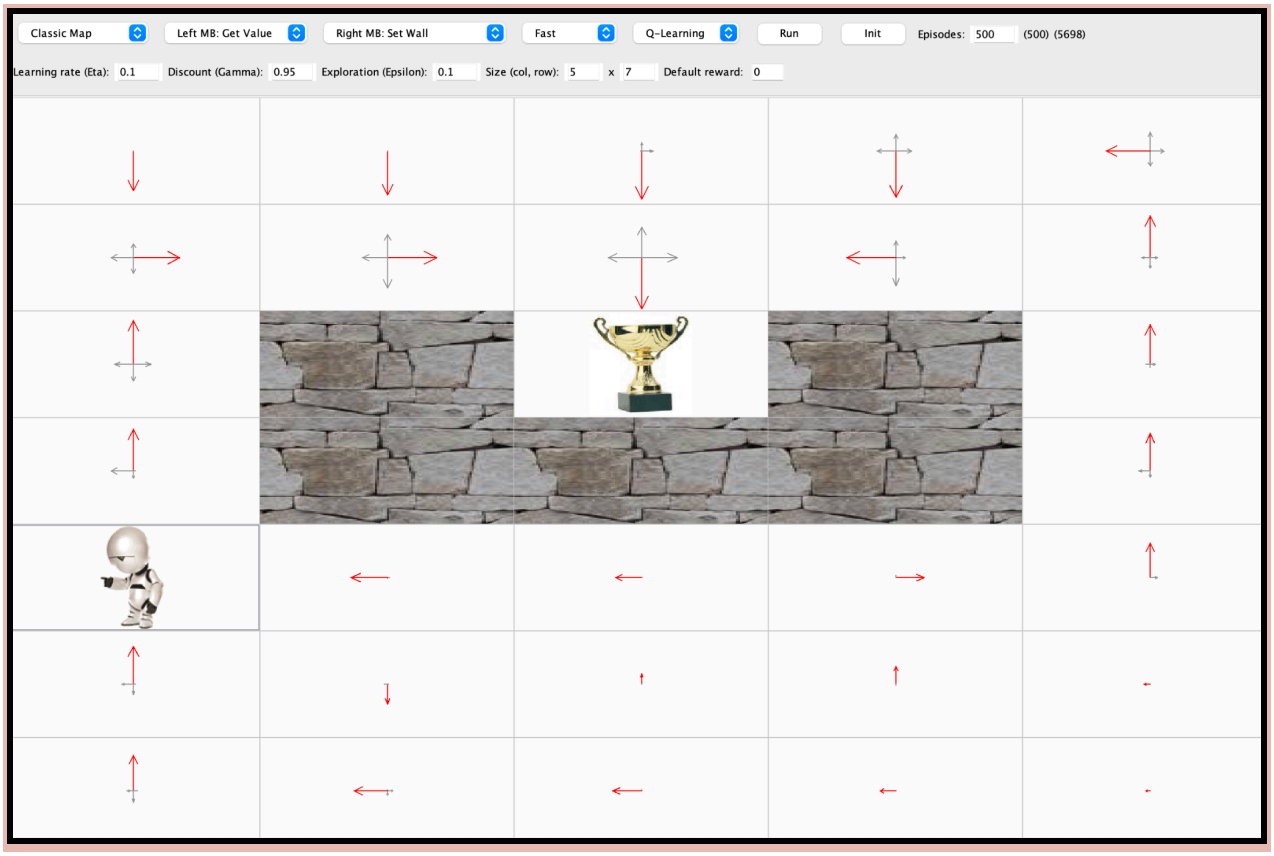
$Q(s1,a) = Q(s1,a)+\alpha[r+\gamma * \max Q(s',a') - Q(s1,a)]$

$Q(s2,a) = Q(s2,a)+\alpha[r+\gamma * \max Q(s',a') - Q(s2,a)]$

Here we can have s1 = 5, s2 = 9, which are **symmetrical** states.
With symmetrical states (same optimal path to the goal), the reward is the same for identical actions. So the arrows will be of the same length.

Question 4 ⋆: If you train the agent long enough, the red arrows will mark the shortest path to the goal. Why does it find the shortest path? (You may have noticed that the problem formulation—how the agent is rewarded—did not define shorter paths as "better".)

As training progresses, exploration decays, and the agent mostly chooses the action with the highest Q-value. So eventually, we form a path built from actions that have the shortest route to the goal. The reward function only provides the reward at the goal, so it doesn't immediately favor shorter paths.
This outcome isn't something we directly enforce but rather a natural consequence of the objective we set. **With a high exploration rate, the agent tries many possible directions, giving the impression that it has discovered the shortest path.** When the agent reaches a state with a higher Q-value, it updates the current state's Q-value accordingly.
As a result, the agent chooses the action with the highest Q-value, which ultimately leads it to follow the shortest path.

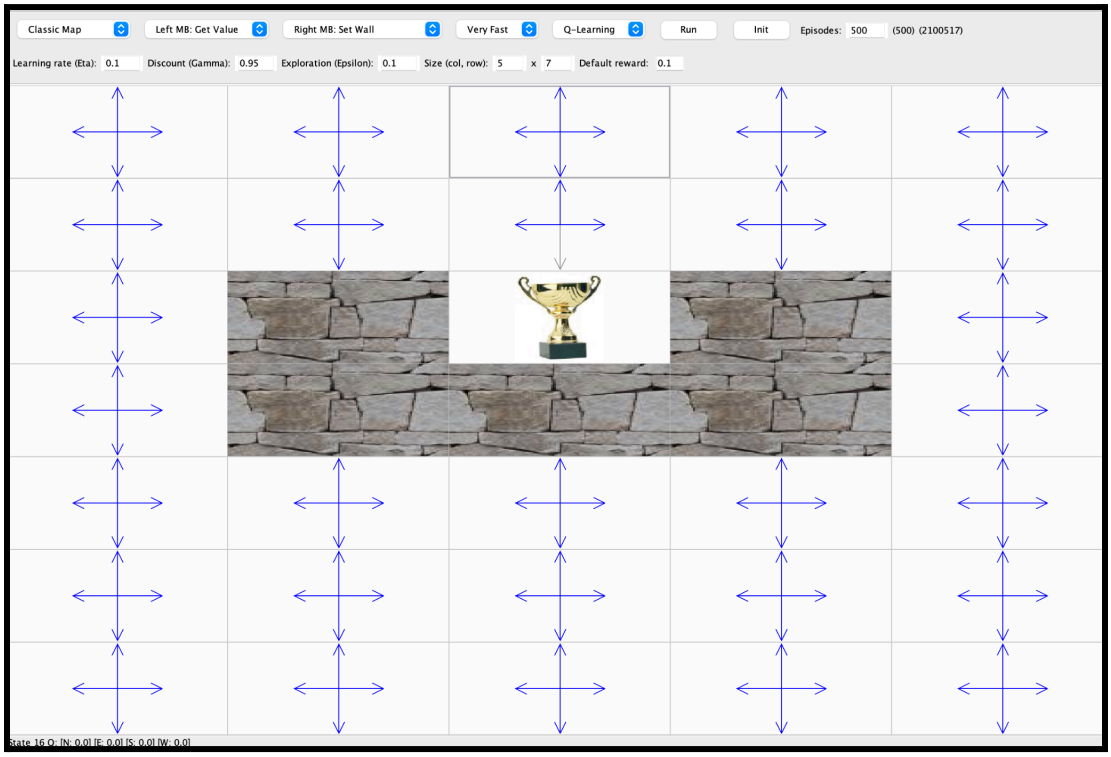


Question 5: Some arrows that previously converged to equal length (with ε = 1.0) will not anymore (unless you train for a very long time). Why?

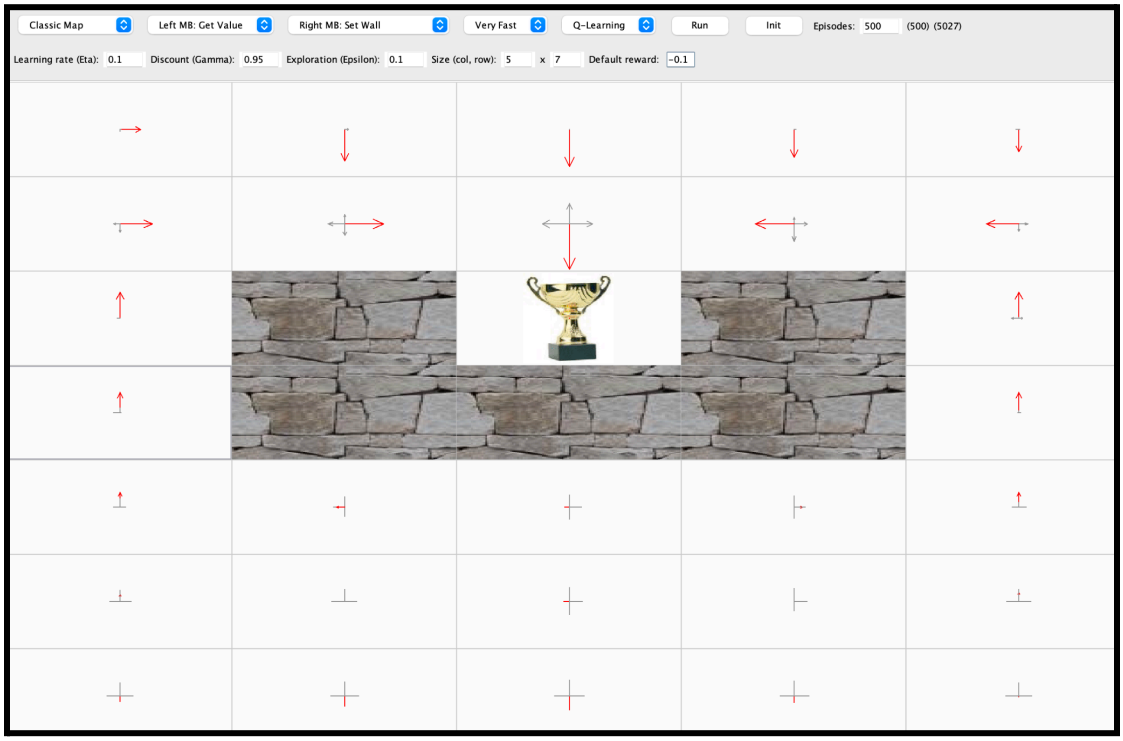
With ε = 1.0, the agent explores randomly, trying all possible actions equally, so Q-values for equally valuable paths even out.

With ε = 0.1, the agent mostly follows the best-known path (greedy choices) and rarely explores alternatives. As a result, less explored actions don't get updated as much, so their Q-values stay lower and don't converge to the same value unless you train for a very long time.

Default reward: 0.1



Default reward: -0.1 (Yellow lines appear when the default reward <= -0.24)



Question 6 : Try setting a default reward of 0.1 and run for enough iterations. Explain what happens and why. Do the same for a default reward of -0.1

Reward = 0.1: The agent is not pushed to reach towards the goal because of a small reward for an optimal action. This encourages the agent to explore/wander more than necessary.

Reward = -0.1: The agent gets penalized for the wrong paths so it instead prefers the shortest route to the goal. Since every step gives a reward of -0.1, the agent will try to minimize the number of steps to reduce the total penalty. The faster it reaches the goal, the fewer negative rewards it accumulates.
Example: If the agent takes 10 steps to reach the goal, it will receive -1.0 total reward. If it takes 15 steps, the total reward will be -1.5, making the longer path less desirable.

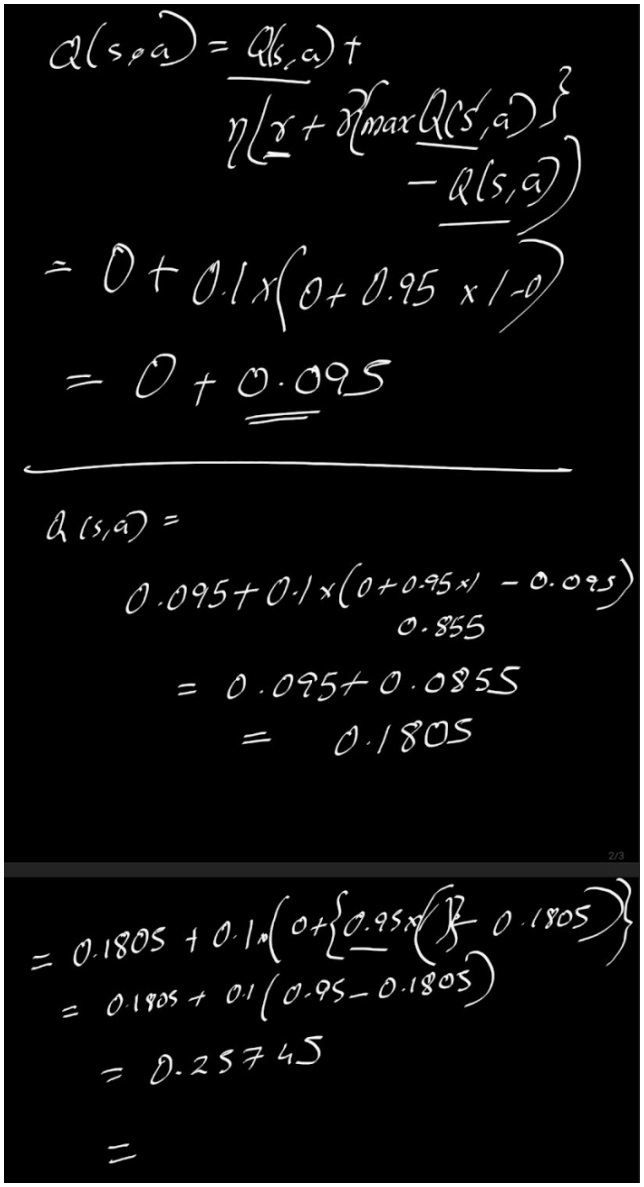
Question 7 : Write down the update rule for Q-learning! Explain the purpose of the different constants and the intuition behind them.

$Q(s, a) := Q(s, a) + \eta\,[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

$Q(s, a)$	Estimated Q-value for taking action a in state s.
r	Immediate reward received after taking action a.
s'	Next state after executing action a.
$\max_{a'} Q(s', a')$	Best Q-value for the next state s'.
γ	Discount factor ($0 \leq \gamma \leq 1$), controlling the importance of future rewards.
α	Learning rate ($0 \leq \alpha \leq 1$), determining how fast Q-values update.

Question 8: What is the true value (the value that Q-learning should converge to Q*) of the action down in state 2 (top row, center square)? Include the complete calculation, not just the answer.

The agent has learned an optimal policy: If Q(s,a) ~ 1, the agent consistently chooses action a in state s. (It's a state very close to the reward and strongly suggests to go South towards the reward.)



Q(s, a) := Q(s, a) + η [r + γ maxa′ Q(s′, a′) − Q(s, a)]

Step-by-Step Updates
Iteration 1:
Q(S, a) = 0 + 0.1 [0 + (0.9 × 1) − 0]
Q(S, a) = 0 + 0.1 × 0.95 = 0.095
Iteration 2:
Q(S, a) = 0.095 + 0.1 [0 + (0.95 × 1) − 0.095]
Q(S, a) = 0.095 + 0.0855 = 0.1805
Iteration 3:
Q(S, a) = 0.1805 + 0.1 [0 + (0.95 × 1) − 0.1805]
Q(S, a) = 0.1805 + 0.1 × (0.95 − 0.1805) = 0.25745

Converged Q-Value
After many iterations, the Q-value converges to: 1

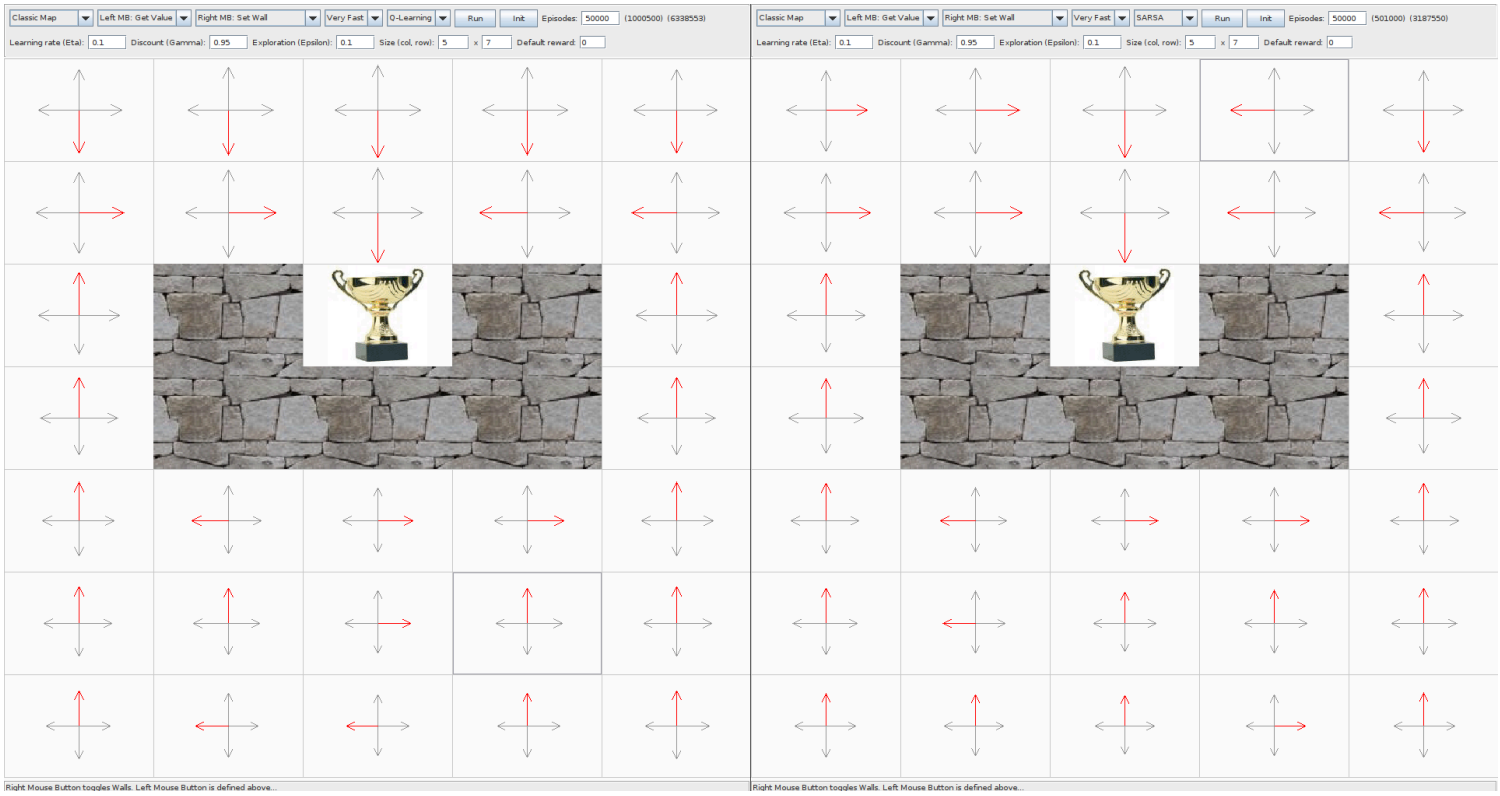
Task 2: SARSA

Question 9: With Q-learning, the lengths of the arrows increased steadily, but using SARSA, they sometimes decreased. Why is that?

With Q-learning, we always consider the maximum of the next state's action (next best possible action) in the update. Whereas in SARSA, a suboptimal action can get penalized with a negative overall score since it depends on the action taken instead of the next best possible action.

Question 10: Do the arrows in the SARSA window converge to the same lengths as the corresponding arrows in the Q-learning window? Motivate your answer

No. The arrow lengths are different, as are the arrow directions.
This is due to SARSA strictly following a particular path once it has found any possible path from the state to the goal. Any other suboptimal paths are henceforth penalized and it keeps picking the first path regardless of optimality.
The final Q-values in SARSA tend to be slightly lower than those in Q-learning because SARSA factors in the possibility of making suboptimal moves.

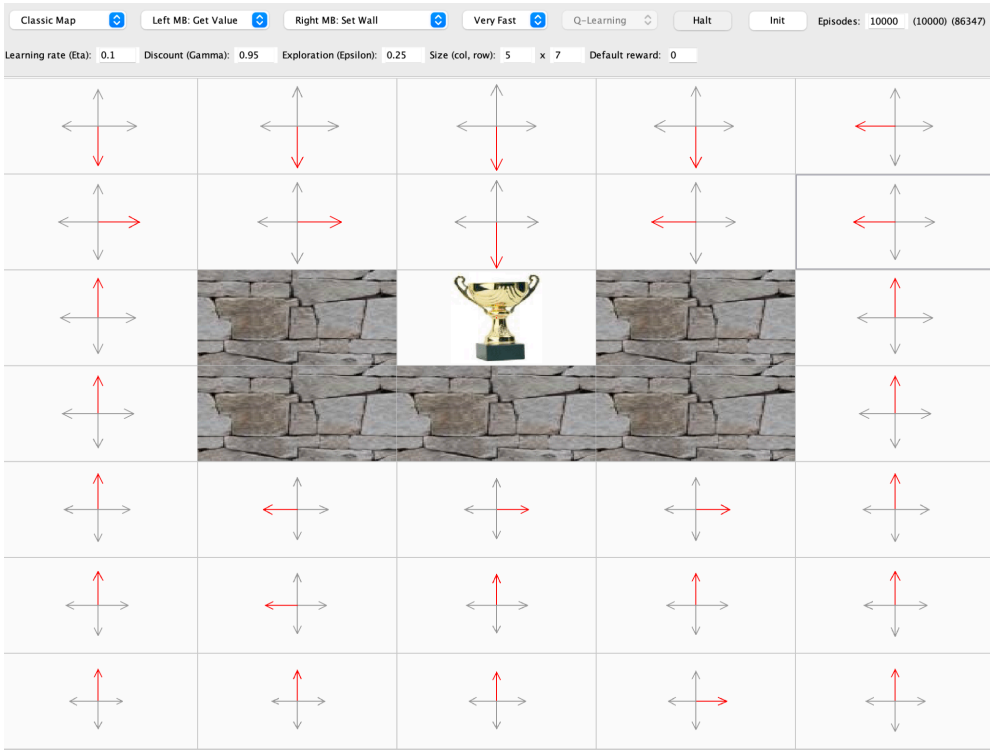


The danger decreases the length because we took a step that decreases the TD error.
Put Q values everywhere (Two rooms) during guidance to decrease exploration.

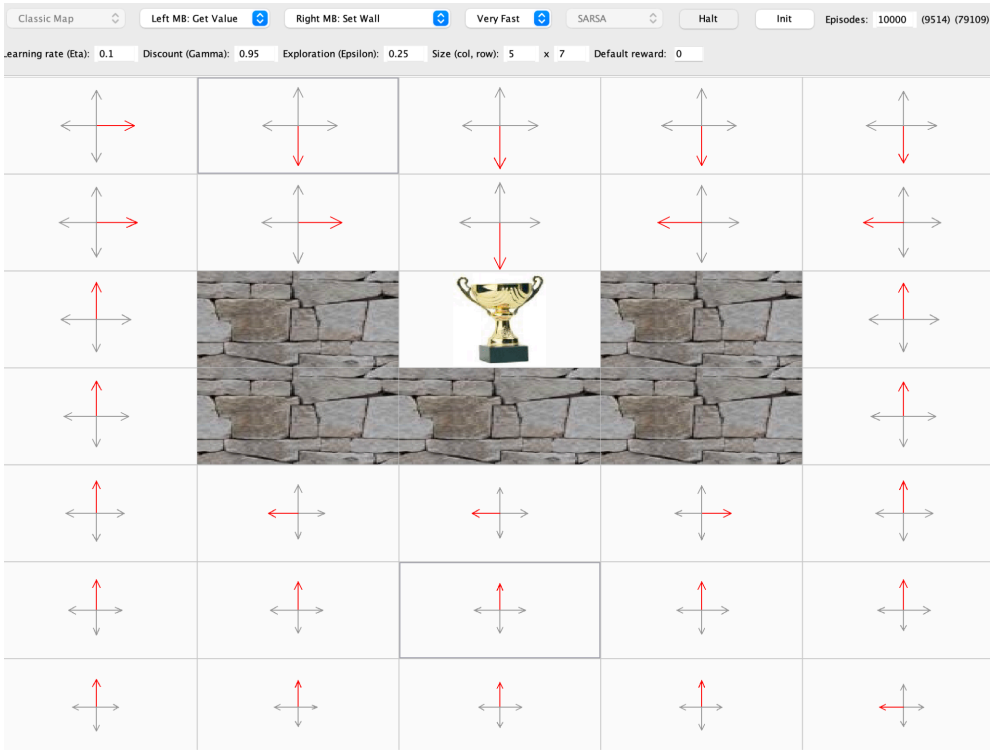
Question 11: The arrows look different from the result of running with Q-learning (as done at the start of the lab). Explain why.

With more conservative value estimates (taking suboptimal values by not exploring as much as the Q learning) in some states, SARSA takes into account the exploration policy when updating values, so if there's a chance of taking a suboptimal action during exploration, SARSA will factor that risk into its value estimates.

Question 12: Run the experiments a few times in Q-learing and study the resulting Q-values. Are they what you expected? Explain.



Question 13: Run the experiments a few times in SARSA and study the resulting Q-values. Are they what you expected? Explain.



With 10,000 episodes, both algorithms have had sufficient time to converge toward optimal policies.
The exploration rate is relatively low, meaning both algorithms spend most of their time exploiting their learned knowledge and converge to similar Q-values.
Yes, it behaved as expected. Q learning gave more or less almost the same value irrespective of exploration rate. Sarsa, with more iterations, was able to reach the q value of q learning.

Question 14: The Q-learning update rule and SARSA update rule behaved differently in this scenario. Explain why.

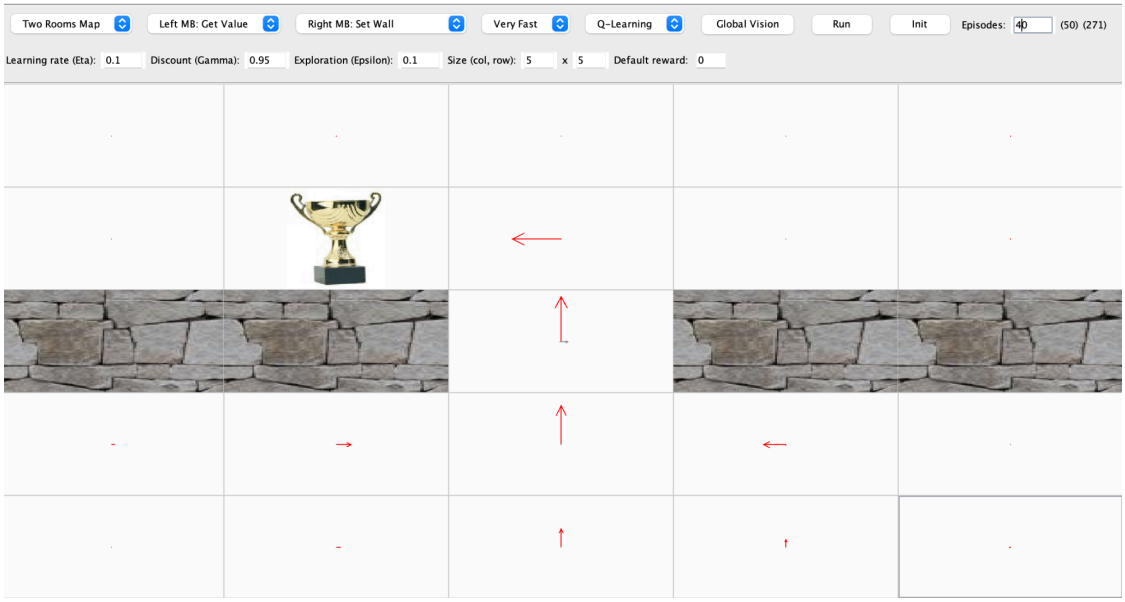
The Q-values for SARSA will always be <= Q-values for Q-learning at any state due to suboptimal updates done in SARSA.

The Q values for Sarsa will always be less than or equal to those of Q learning because it will consider suboptimal values. But as we have 10,000 episodes with a decent amount set for the exploration/epsilon, it reaches Q values very close to that of Q learning.

Question 15: What can you say about the average action count in the experiment? Compare guiding the robot to not guiding the robot.

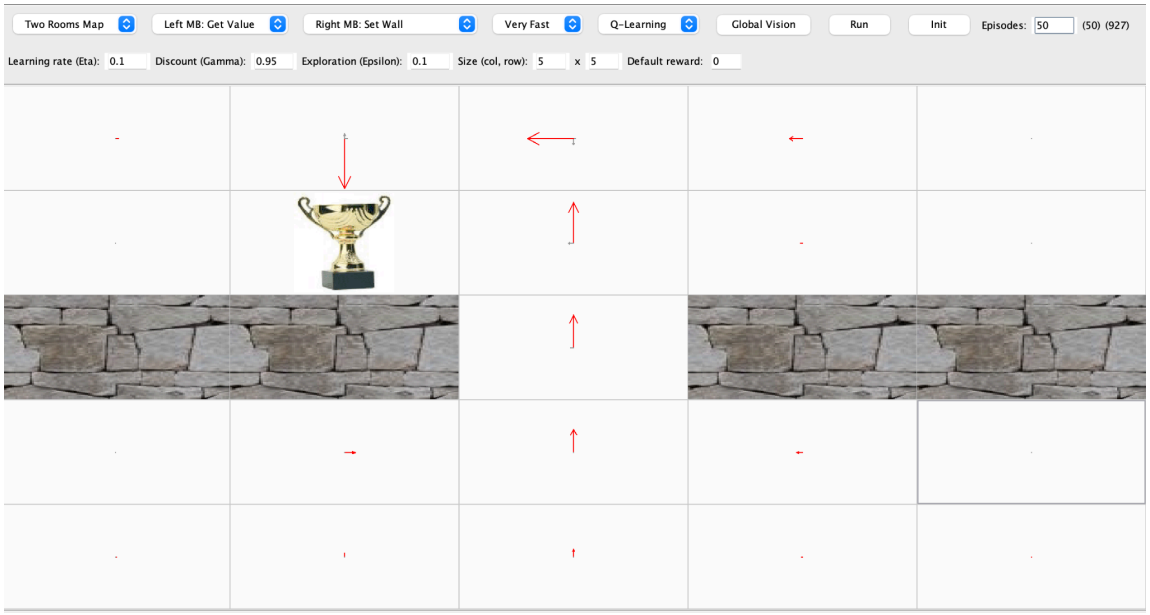
With Guidance -

The agent took 271 actions over 50 episodes. This comes to 5.42 average actions per episode.



Without Guidance -

The agent took 927 actions over 50 episodes. This comes to 18.54 average actions per episode.



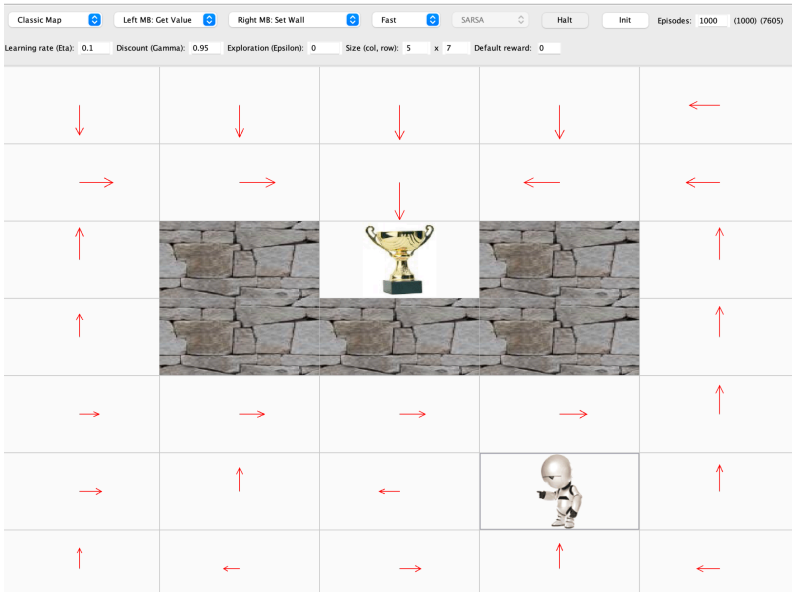
Additional experiments - Setting LR to 0.001 makes the Q-value updates very small and the arrows are extremely short

Question 16: What source did you mainly use for preparing for this lab? Did you also use other sources? Examples include lectures, books, and websites.

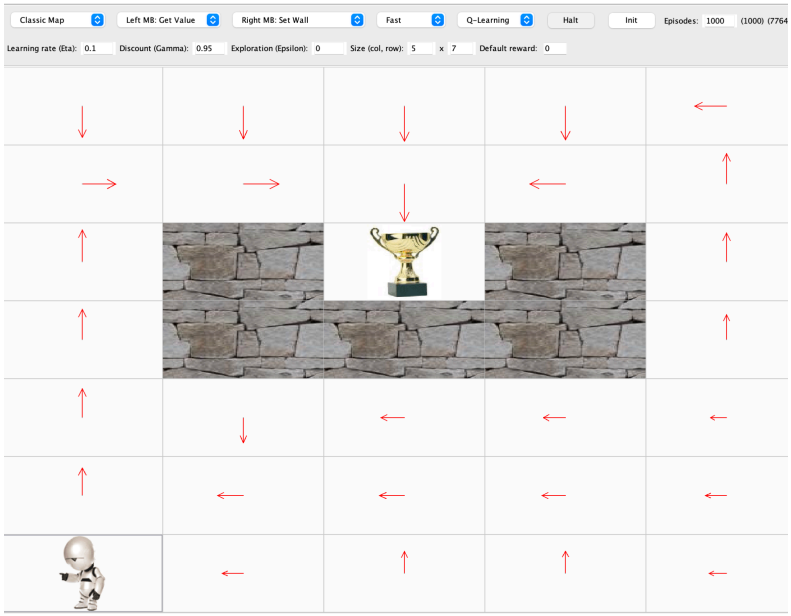
Studium lecture slides

https://www.youtube.com/watch?v=0iqz4tcKN58

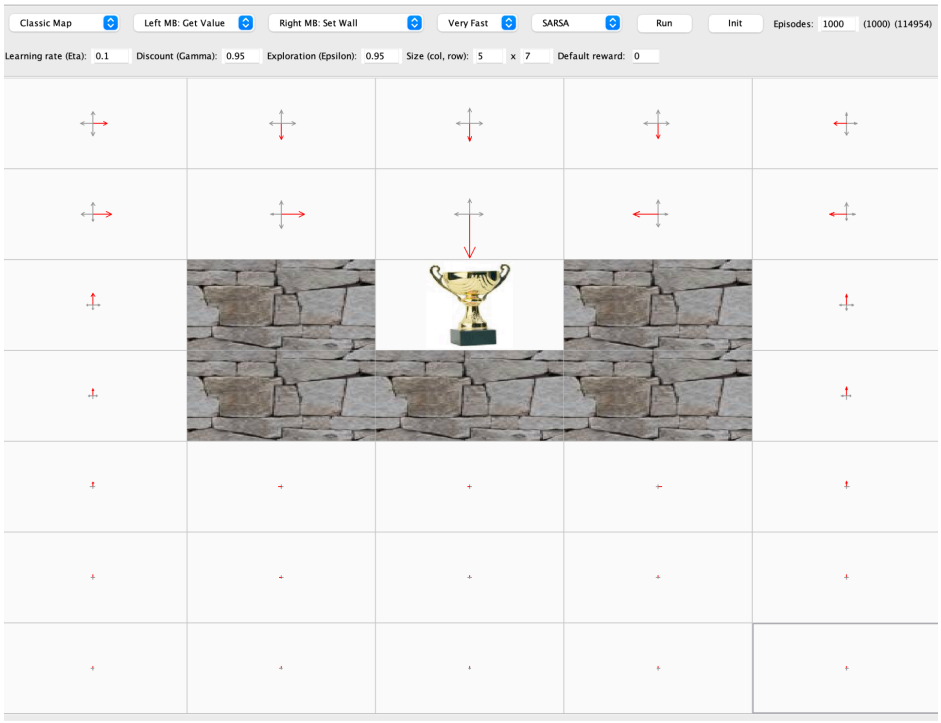
Sarsa: Epsilon:0



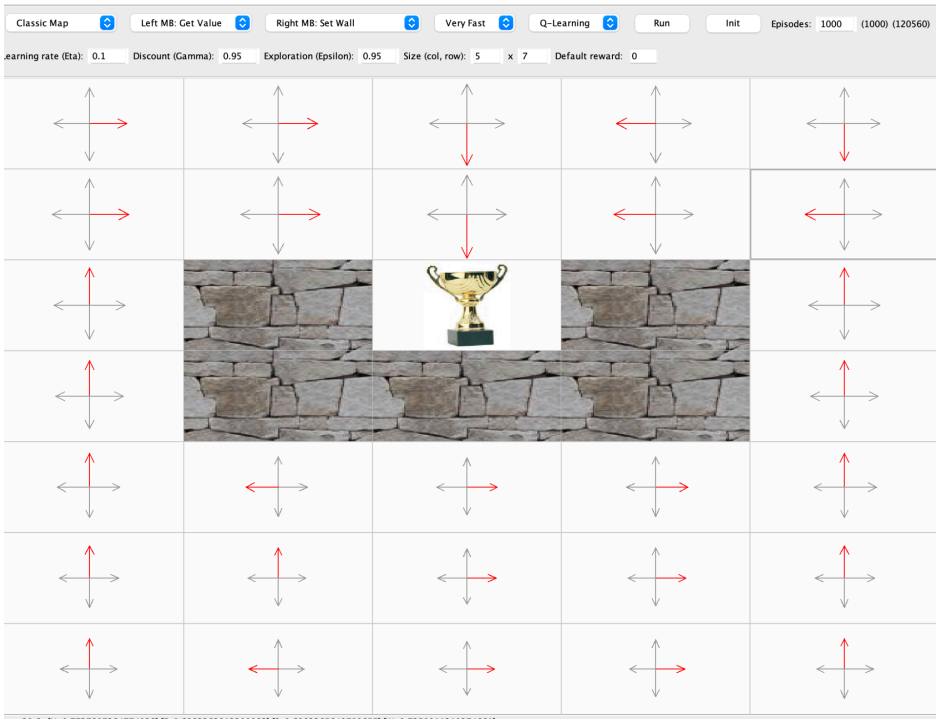
QLearning: Epsilon:0



Sarsa: Epsilon:0.95



QLearning: Epsilon:0.95



Guide a bit out of the optimal path. To give the agent an idea. So, try to get to the reward from every step.

Sarsa sticks to the first optimal path so high chances of getting stuck in the local minima.