Goals

- Implement value iteration
- Implement policy iteration
- Implement Q-learning with epsilon-greedy policy

Activities

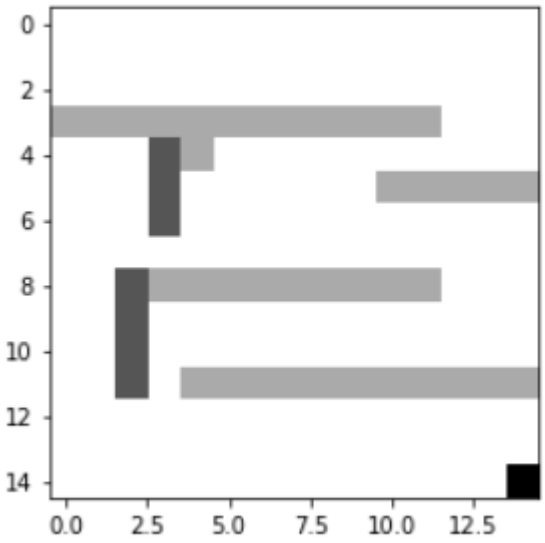
A.Part 1 (use a2p1.ipynb):

1. Add a wall with length of 3 cells at an arbitrary position either vertically or horizontally, or both.

Make sure there is still a path from the START to the GOAL.

-> Added a vertical wall at column 3 from row 4-7,horizontal wall at [4,4], ensuring that path exist form the START to the GOAL.

2. Visualize the maze at the beginning verify your wall cells are where you want.



-> Verified the wall based on the visualization.

3. Implement Policy Iteration.

->Implemented Policy Iteration.

4. Implement Value Iteration.

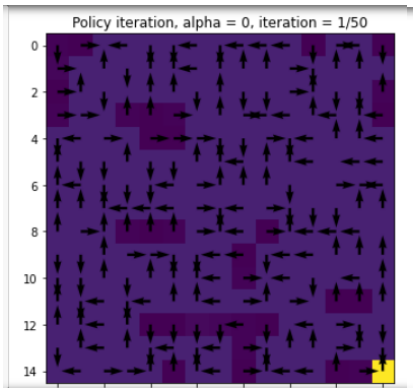
->Implemented Value Iteration.

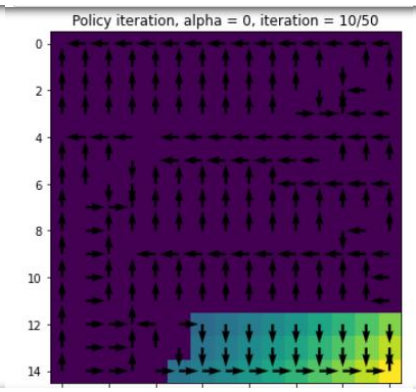
5. Test if your implementations. Policy iteration takes some time to run. Its a good idea to reduce the number of iterations and evaluations to 5-10 to see if your implementation is working.

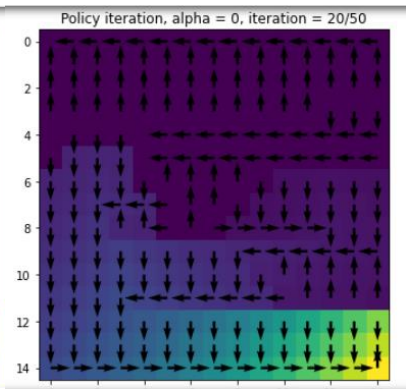
->Implemented Code is working.

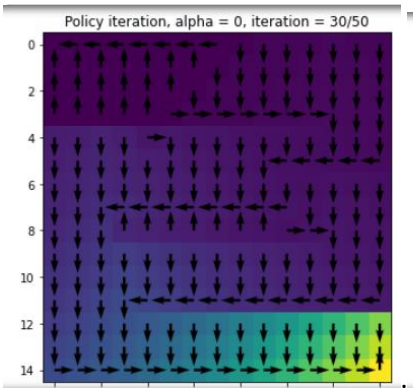
6. Run Policy Iteration and Value Iteration with 0 noise, record your output to the pdf. Visualize 5 iterations of value and policy for both algorithms showing their progress.

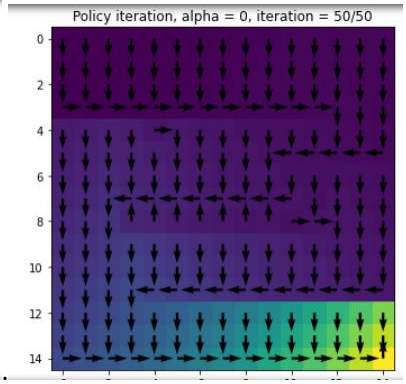
->Implemented Code is working

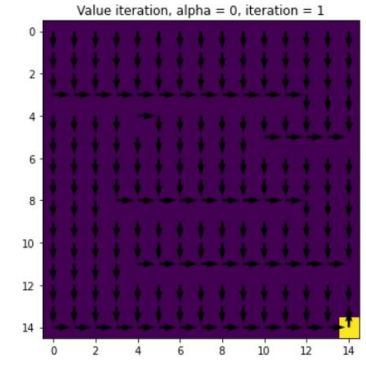


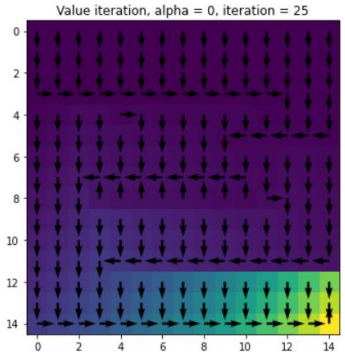


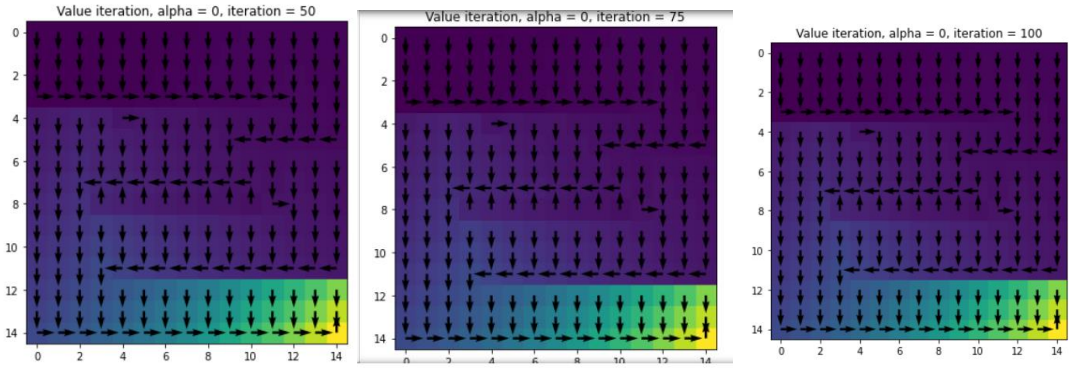












7. For each algorithm, explain the difference between iterations.

Policy Iteration starts with random policy and improves over it every iteration. We can observe the initial iteration of Policy iteration algorithm does not provide good policy.

Over next iterations 10,20,30,50 it improves it improves its performance.

We can also observe the color coded the values of V-values improving as we iteration progressing.

In case value iteration we get better policy very early in the iteration but the value keeps on getting updated for V-values for each state. Both the algorithm reaches same solution. Value iteration might reach optimal policy very early. Policy iteration algorithm performance depends on the initial policy being good or not. Number of calculation involved in policy iteration might be more as well.

8. Run Policy and Value Iteration with noise, record valua and policy from 5 iterations to the pdf.

->Implemented Code is working.

9. Write down your observations regarding the difference between running with noise and without noise.

With Noise 0.2 :

Policy iteration and Value Iteration both takes few more iterations to reach the optimal performance and we can observe they gradually improve over iteration.

With Noise 0.8:

Policy iteration and Value Iteration both do not converge to optimal policy as the moves have significant amount of noise making it difficult for the algorithm to reach optimized goal

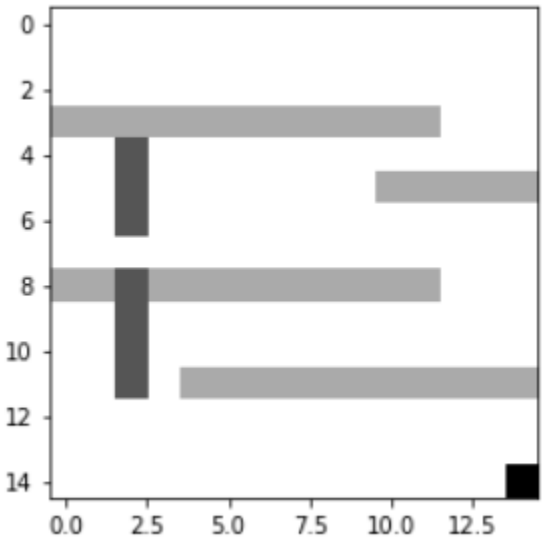
Part 2 (use a2p2.ipynb):

1. Add a wall with length of 3 cells at an arbitrary position either vertically or horizontally, or both.

Make sure there is still a path from the START to the GOAL.

-> Added a vertical wall at column 3 from row 4-7,horizontal wall at row8 from column 1-3, ensuring that path exist form the START to the GOAL.

2. Visualize the maze at the beginning verify your wall cells are where you want.



3. Finish the step function.

->Implemented Code is working.

4. Finish the choose_action_epsilon function.

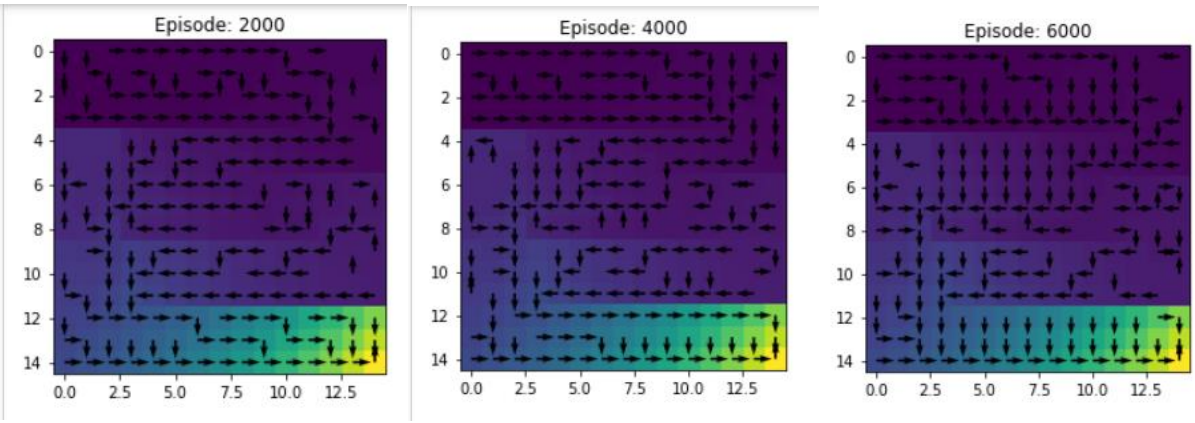
->Implemented Code is working.

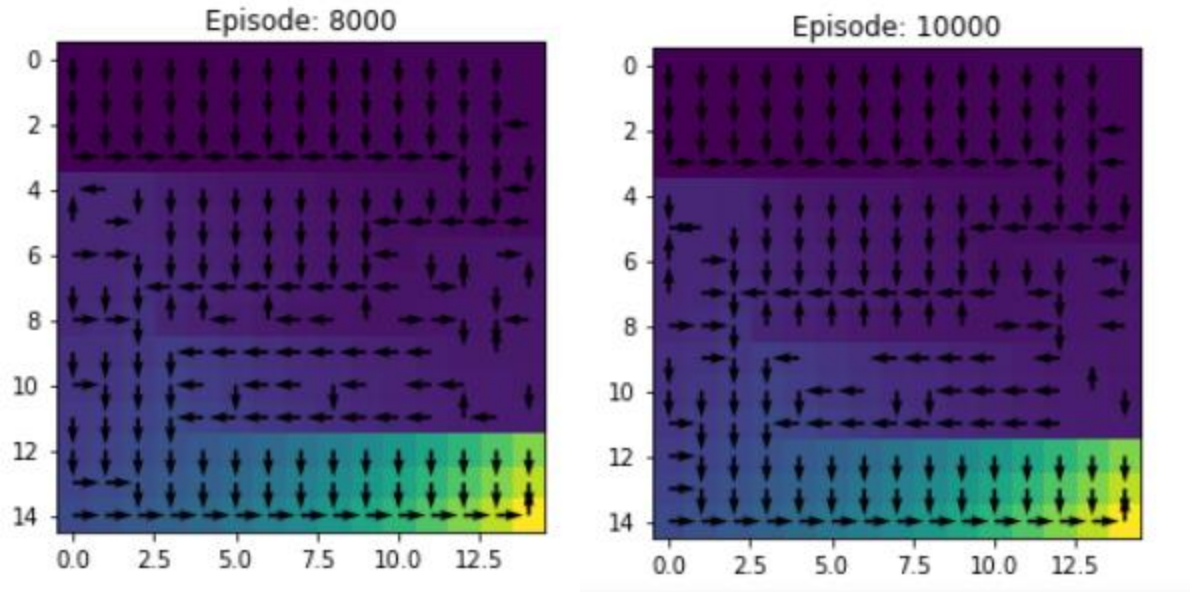
5. Implement the Q-learning update.

->Implemented Code is working.

6. Save five images showing the training progress of the

q-learning agent. Discuss the differences between the images.





Q learning improves over multiple iteration and episode reward is better with more iteration. Initially the policy based on q -learning is not optimized. As the iteration increases. It converges to better output.