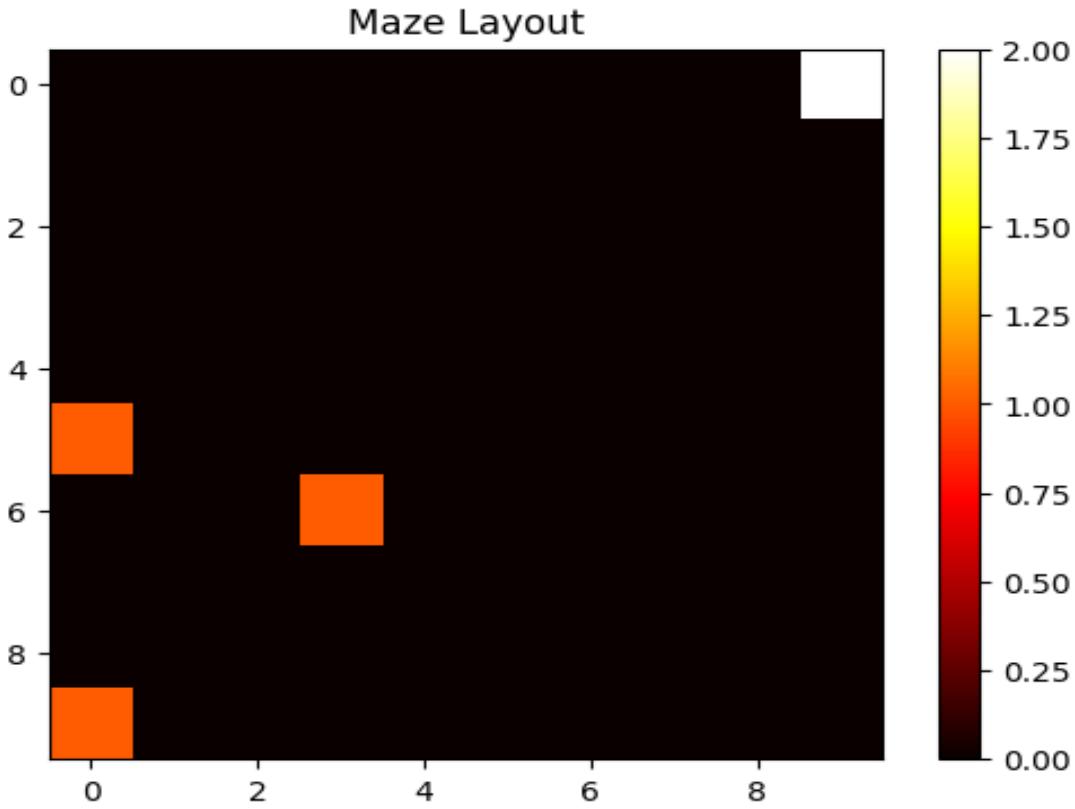
Assignment 2

Swapnil Sanjay Pawar (R11903328)

Priyanka Saxena (R11905469)

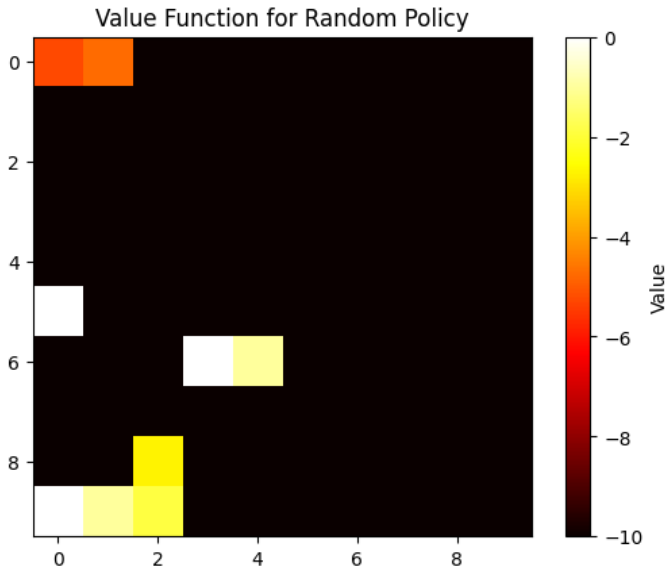
Task 1:

Build your maze with dimensions 10 x 10 and 3 fences, and the goal state (exit) in one of the corners of the maze. Visualize the maze layout on 2D plot.



Task 2:

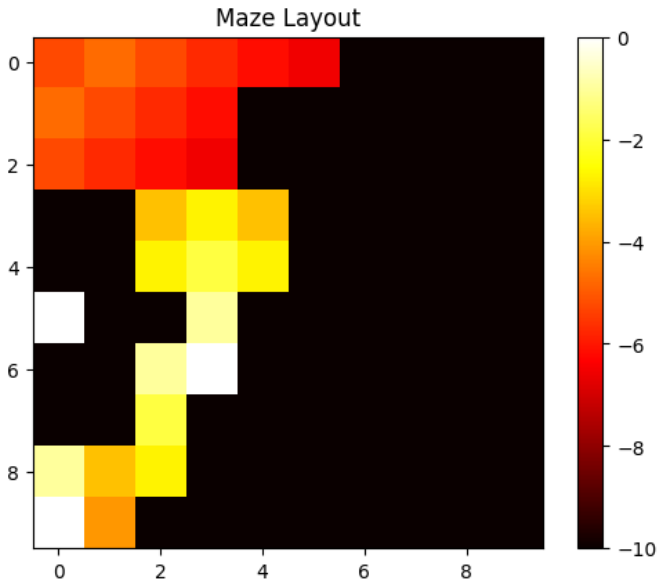
Implement the Policy Evaluation (PE) algorithm for a deterministic policy, π. There is a single possible action at every state. Evaluate a random deterministic policy, π. Plot Value of a random policy on your maze layout.



Task 3:

Repeat Task 2 with manually setting the optimal actions in the radius of 2 states from the goal state. Explain your observations.

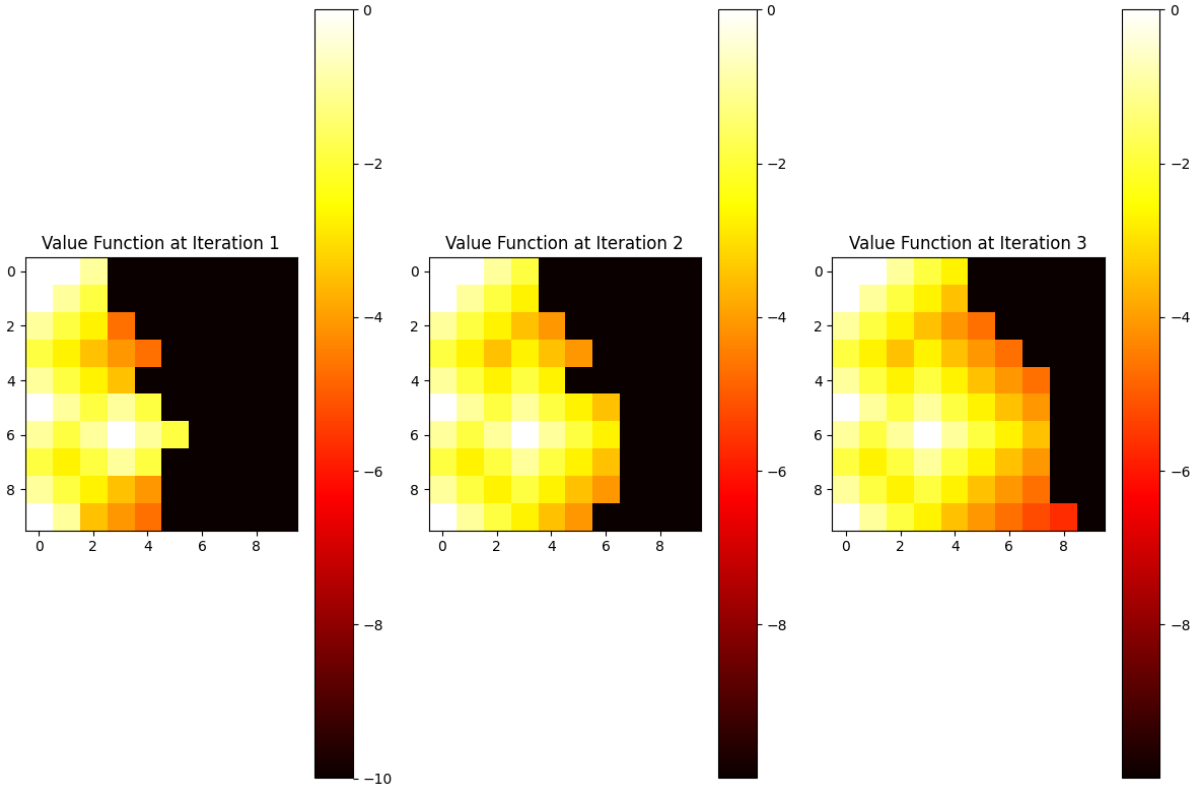
⇨ As observed, by explicitly defining optimal actions in the two-cell vicinity of the goal state, the policy augmented the by far capacity for reaching the goal state with the displayed boost around the regions near the goal state. The value function was higher near the goal, because, having more steps to the goal, there were fewer penalties in the case with the policy with respect to the random policy. Still, the effect was showed to be only localized on the states: the random policy negatively affected the farther states from the goal, which resulted in the suboptimal behaviour of the agent in those regions. This proves that one can improve efficiency near the goal, although a globally efficient policy necessitates improvements around the entire maze.



Task 4:

Implement the Policy Improvement (PI) Algorithm, and find the optimal policy π*. Visualize the optimal value function, V_i, on a 2D plot (maze layout) at 3 different iterations, i, of PI. Explain your observations.

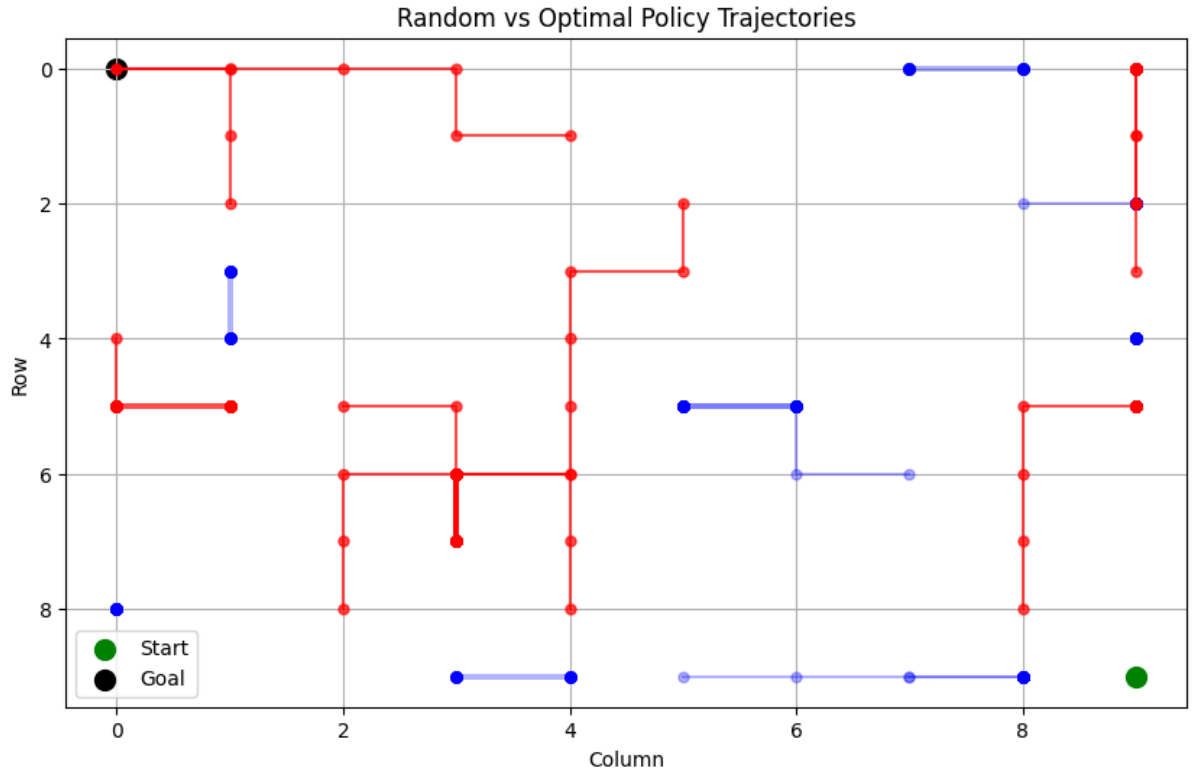
- ⇨ Regarding the Policy Improvement (PI) algorithm, the value function became gradually better with iterations as the policy was updated progressively. First, the value function depicted low value across the maze denoting a stochastic plan that was not helpful in identifying what an efficient agent would do. In the following iterations however, the calculated values in the value function increased toward the end, which means it offers better guidance towards the goal. The policy converging in a third iteration and the value function starts sharpening in the vicinity of the Goal shown that it has converged to the Optimal Policy. This affirms with how the PI algorithm improves or updates the policy time after time by using the current value function in its evaluation to obtain the best result.



Task 5:

Write a function, s', r = step(s, a), that receives the current state, s, and the current action, a, and returns the next state, s', and reward, r. Generate 10 trajectories from 10 different initial states, using a random policy. Generate 10 trajectories from 10 different initial states, using the optimal policy from Task 4 above. Explain your observations.

⇨ In the visualization of the trajectories generated using the random and optimal policies, we observe clear differences in the efficiency and path taken by the agent. The random policy trajectories are erratic, with the agent wandering in different directions without a clear path to the goal, often taking unnecessary steps and sometimes failing to reach the goal within the maximum number of steps. These trajectories are represented by blue lines and appear scattered and inefficient. In contrast, the optimal policy trajectories are more direct and efficient, as shown by the red lines. The agent following the optimal policy consistently reaches the goal in fewer steps, taking a clear path with minimal detours. The difference highlights the effectiveness of the optimal policy in guiding the agent toward the goal more quickly, while the random policy results in a less predictable and slower performance. The green marker indicates the start position, and the black marker shows the goal, emphasizing the difference in how quickly each policy leads the agent to the goal.



Task 6:

Implement Q-learning algorithm for the tabular case, where Q function is given by a table. Plot an accumulated reward as a function of the iteration number of Q-learning algorithm for 5 runs of Q-learning from scratch. Plot an average curve of the 4 runs of Q-learning, and the variance (use fill_between). Explain your observations Repeat Task 5 with the SARSA Algorithm. Explain your observations.

⇨ In the Q-learning algorithm, the total sum of the reward received raises over time from the policy learning via balance between exploration and exploitation. From the epsilon-greedy strategy, we are sure the agent goes through enough exploration but when exploitation of best-known actions begins the average reward rises to depict the learning. The variances are observed to reduce when the agent is closer to the optimal policy, and this shows that the probability distribution of the agent gets closer to the one with the lowest entropy when the agent becomes more efficient in the environment. On the other hand, the accumulated reward in the SARSA algorithm increases with iterations but its convergence rate is slightly slower than that of the Q-learning algorithm. This is so because SARSA is an on-policy method that uses the actual taken steps to update the value function unlike the off-policy Q-learning which uses the optimal future steps to update it, this makes SARSA explore the state space more carefully. Consequently, SARSA has higher variance, and its learning rate is slower, because the agent can spend a lot of time on sub-optimal paths, which is why its learning depends crucially on the initial exploration phase.