CS 484

Assignment 3: Reinforcement Learning Algorithms

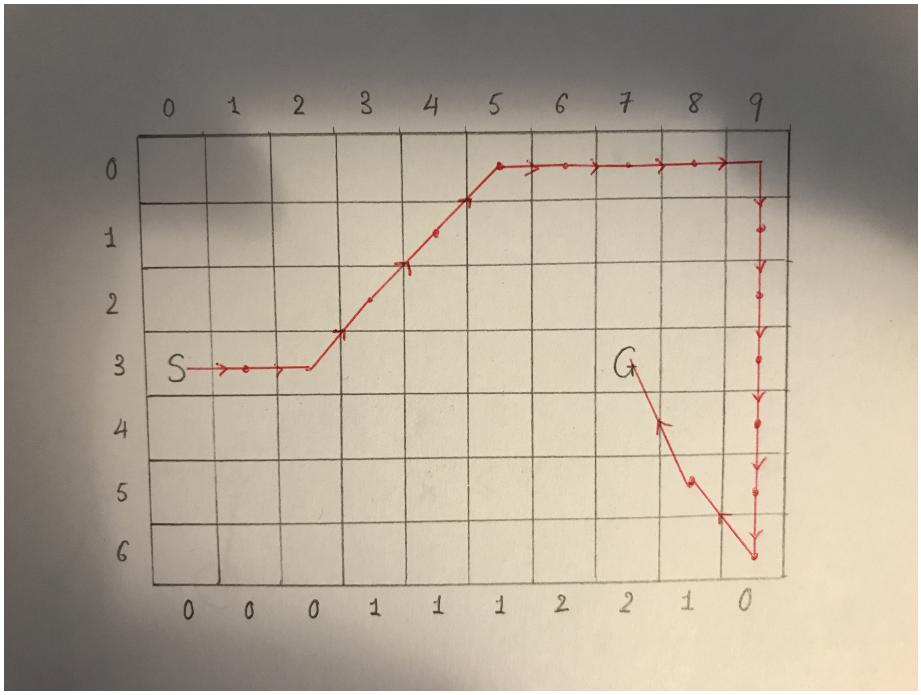
Hieu Le

Problem Formulation

Each cell in the given grid could be regarded as a state. From each state, the agent could adopt four possible actions: move up, down left, or right. This represents a deterministic domain whereby taking an action from a particular state leads to a resultant state with absolute certainty – each move deterministically moves the agent in the chosen direction. If the agent takes an action that would move it off the edge of the grid, it will remain on the last legitimate cell within the grid's boundaries. If the agent moves onto a column with a positive wind force, the agent will be shifted upwards the same number of cells as the magnitude of the wind in that column.

The optimal strategy is defined as the route consisting of the minimum number of actions the agent could take to traverse from a given start cell to a destination cell. The optimal route could be approximated by applying reinforcement learning techniques which allow agents to learn how to take actions in the grid environment so as to maximize the cumulative reward. To facilitate minimizing the number of moves, each taken action will impose a negative reward of `-1` with the exception of an action leading to the goal state which is assigned a reward of `100`. Tweaking reward assignment has a non-trivial effect on how quickly the agent could learn to identify the optimal strategy.

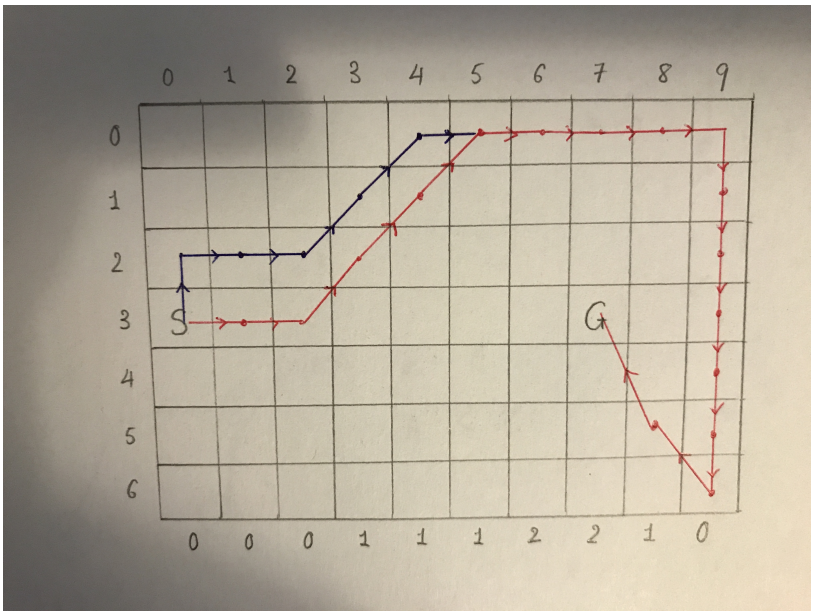
To evaluate the performance of different reinforcement learning algorithms, we could compare their output strategy with the provably optimal route via running Breadth-First Search on the grid. The optimal strategy consisting of 17 steps is shown in red below.



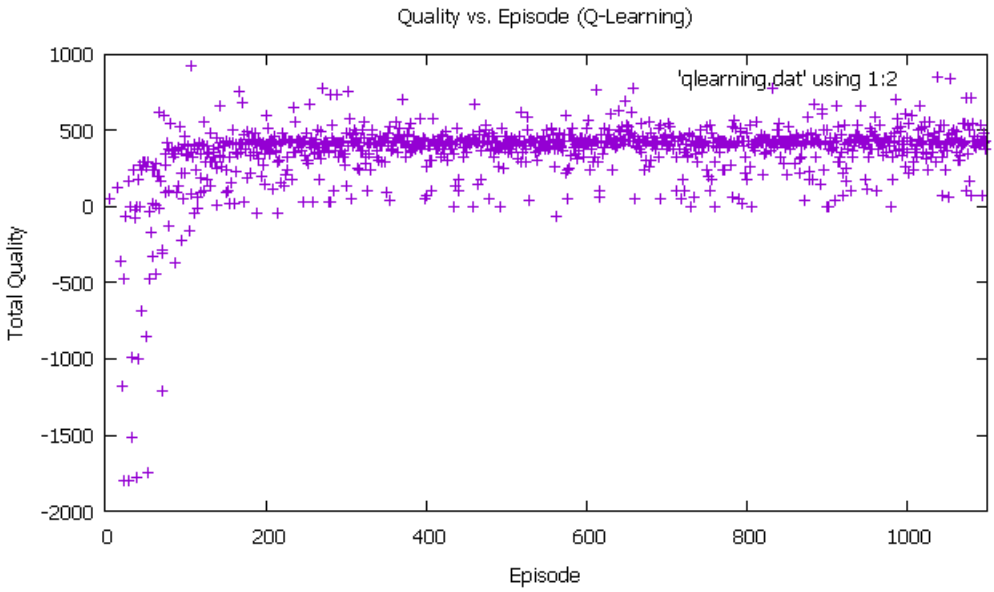
For both implementations of Q-Learning and SARSA algorithms in this project, the learning rate α is initialized to `0.10` and the discount factor γ is initialized to `0.80`. The training state is initialized by picking an arbitrary state from the search space. Actions are selected from each state using the ϵ-greedy policy, with the value of ϵ initialized to `0.1`. All of these functionalities are encapsulated in the base `TemporalDifferenceLearning` class, which also provides initialization logic and strategy generation. Q-Learning and SARSA therefore only differ from each other in how they train and update the Q-value tables.

Q-Learning Algorithm

Up until the `300` episodes, Q-Learning does not report a strategy that reaches the goal state. A complete route is generated when training the algorithm over `300` episodes which is only outperformed by the optimal strategy by two extra steps. Training the algorithm over `600` episodes moves it within one step from the optimal strategy. An output strategy consisting of 18 steps is shown in blue below.



The optimal strategy is eventually discovered after training the algorithm over `1100` episodes. It yields the exact same route as obtained from running Breadth-First Search on the grid. For each training episode, the total quality scores of the path from a randomly chosen state to the goal state is also recorded. The quality scores for the first `1000` episodes is as follows.



SARSA Algorithm

SARSA does not report a strategy to reach the goal state until the 2600th episode. The algorithm obtains the optimal solution after the 5000th episode. The quality scores for the first 1000 episodes of training SARSA algorithm is as follows.