Problem Descriptions

In this problem, we consider a maze shown on the second page. This maze consists of several walls that the agent cannot enter, and bumps and oils that moving to them have negative rewards. For simplicity, consider an 18×18 matrix, where each element is associated with one of the following:

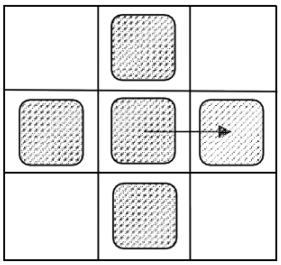
- *Empty*
- *Full (Wall)*
- *Bump*
- *Oil*

State Space (S): The state-space contains all cells in the maze except the walls, where the agent can possibly be there $(18 \times 18 - 76(walls) = 248)$.

Action Space (A): The agent can take one of the four possible actions at any given state: *up (U), down (D), right (R), and left (L).*

Transition Probabilities: After choosing an action, the agent will either move to one of the neighborhood cells or stay in its current cell. After taking any action, with a probability of 1-p, the agent moves the anticipated state and, with an equal probability of p/3, will move to one of the other neighboring cells.

Consider the following example:

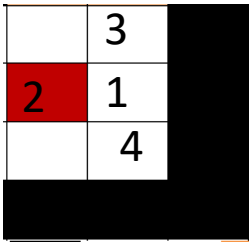


If action *right (R)* is selected, the agent: $\begin{cases} \text{moves to the state in right with probability } 1-\text{p} \\ \text{moves to the state in left with probability p}/3 \\ \text{moves to the the state in up with probability p}/3 \\ \text{moves to the state in down with probability p}/3 \end{cases}$

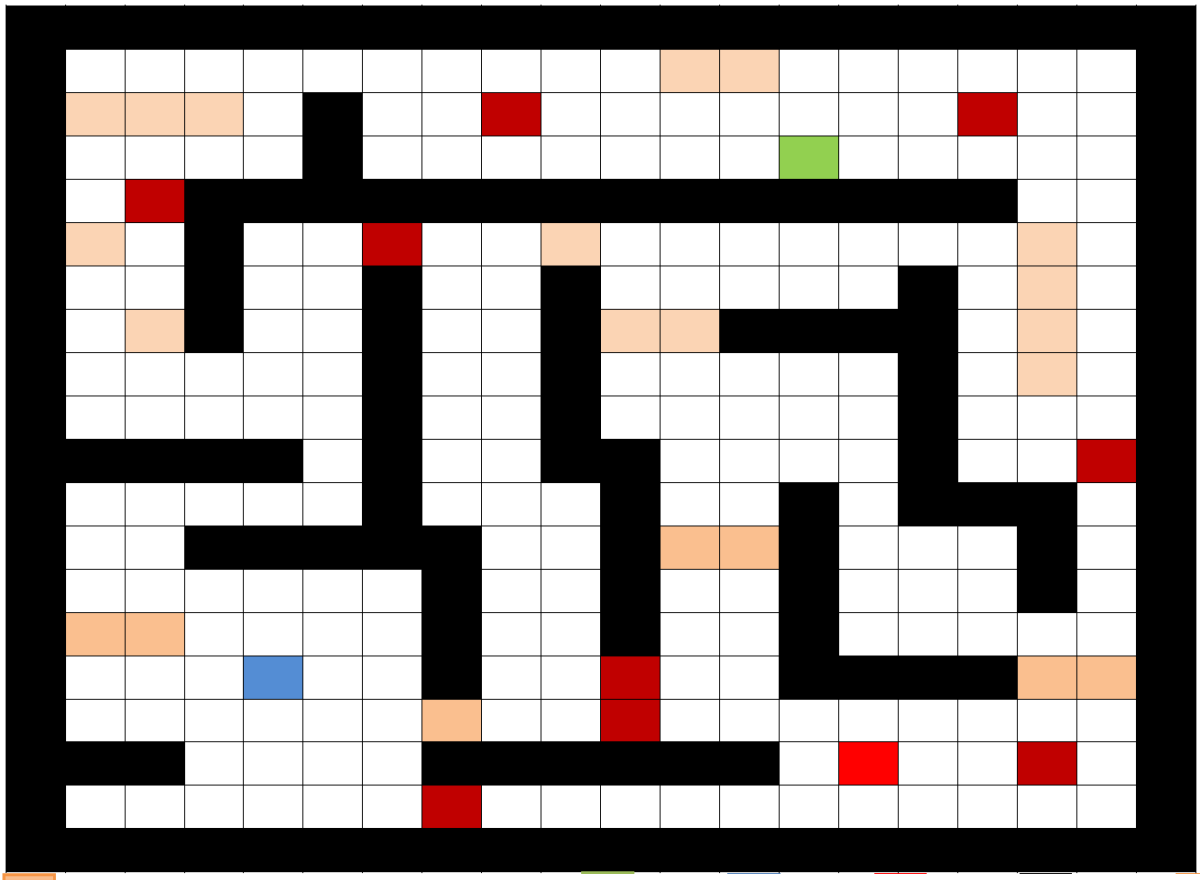
Notice that if any of the neighboring cells are wall, the agent stays in the current cell.

Reward Function:

The primary objective is to find the optimal policy which leads the agent to the goal state with the maximum accumulated reward. Therefore, we define the reward of taking any action -1, the reward for ending up to the oil states -5, the reward for ending up the bump states -10, and the reward for the goal state +200. For instance, this can be expressed for the following example as:



$$\begin{cases} R(1, L, 2) = -6 & -5 \text{ for oil and } -1 \text{ for taking an action} \\ R(1, L, 3) = -1 & \text{for taking an action} \\ R(1, L, 4) = -1 & \text{for taking an action} \\ R(1, L, 1) = -1 & \text{for taking an action} \end{cases}$$



Bump

Wall

Oil

Start

End

The goal is to analyze the performance of the temporal difference learning algorithms. Set $p = 0.02$, $\gamma = 0.95$, $\alpha = 0.3$, $\epsilon = 0.1$, the total number of episodes 1,000, the maximum number of steps if the agent does not reach the goal state 1,000 (i.e., the maximum length of each episode). Any parameter that is not specified can be set/tuned by you, and you need to include them in your final report.

Implement Q-Learning, SARSA and actor-critic algorithm ($\beta = 0.05$) algorithms. For each algorithm, you need to run the algorithm for 10 independent runs. Report your results in terms of:

a) How many times among 10 independent runs, upon the termination of learning, a path from start to goal has been obtained?

b) Show the optimal policy and the optimal path from start to goal for one of the 10 independent runs (use the results of a run that the path from start to goal is found; if the path is not found in all runs, show the results from a random run).
c) Show the average accumulated reward (in 10 independent runs) with respect to the episode number.
d) Plot the average accumulated reward with respect to the episode number for all algorithms in a single plot. Describe your findings.

*For any of the algorithms that in all runs the path from start to goal is not found, you need to change the parameters (e.g., α, β, ϵ, episode number, length of the episode) in a trial and error and report the best solutions found.

SARSA Algorithm

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
. Initialize s
. Choose a from s using policy derived from Q
. . . (e.g., ϵ -greedy)
. Repeat (for each step of episode):
. . Take action a, observe r, s'
. . Choose a' from s' using policy derived from Q
. . . (e.g., ϵ -greedy)
. . $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
. . $s \leftarrow s'$; $a \leftarrow a'$;
. until s is terminal

Q-Learning Algorithm

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
. Initialize s
. Repeat (for each step of episode):
. . Choose a from s using policy derived from Q
. . . (e.g., ϵ -greedy)
. . Take action a, observe r, s'
. . $Q(s, a) \leftarrow Q(s, a) + \alpha\left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$
. . $s \leftarrow s'$;
. until s is terminal

Tabular Actor-Critic Algorithm

$V(s)=0$, $H(s,a)=0$, for all $s \in \mathcal{S}$, $a \in \mathcal{A}$.

Repeat for N episodes

- Start from a random state $s_0 \in \mathcal{S}$, $t=0$

  While $t<T$ (episode length):

  - Select action: $a_t \sim \pi(\cdot|s_t)$: $\pi(a|s)=\frac{e^{H(s,a)}}{\sum_{a' \in A} e^{H(s,a')}}$
  - Take action a_t, move to state s_{t+1} and observe R_{t+1}.
  - $\delta_t = R_{t+1} + \gamma V(s_{t+1}) - V(s_t)$
  - $V(s_t) = V(s_t) + \alpha \delta_t$
  - $H(s_t, a_t) = H(s_t, a_t) + \beta \delta_t (1 - \pi(a_t|s_t))$
  - $t = t+1$

Please attach your report and codes in a zip file and email it to me at
f.ghoreishi@northeastern.edu