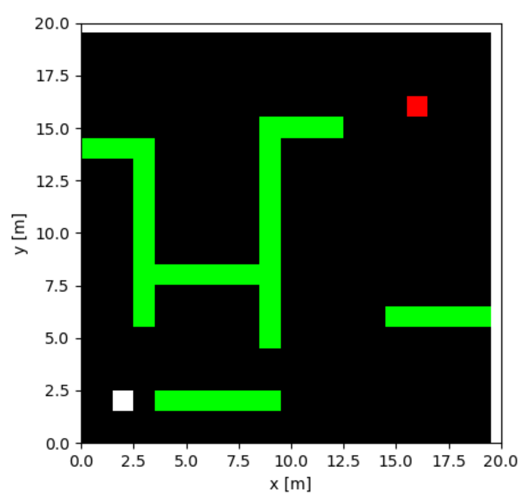
Homework 5 - Reinforcement Learning

-- Course: *Intelligent Robotics – Professor: Qi Hao*

Coding Homeworks. Most of coding assignments will be done by Python(>=3.5) under a simple robotics simulator. You can follow the Coding instruction to use this simulator to complete the coding part in question1-3. Your final submission should be a compressed package with extension .zip, which includes your codes and explanations (you need to know how to write the manuscript with Markdown or LATEX). Your code should be run step-by-step without any error. Real-time animation is also recommended.

Grid Map Environment

- white: the start position
- red: the goal position
- green: the obstacle
- black: ground
- obstacle reward: -1
- goal reward: 10
- other reward: 0
- over the bound: -5



Question1

Please simulate the Monte Carlo Reinforcemenet learning with Exploring Starts under the given grid map environment.

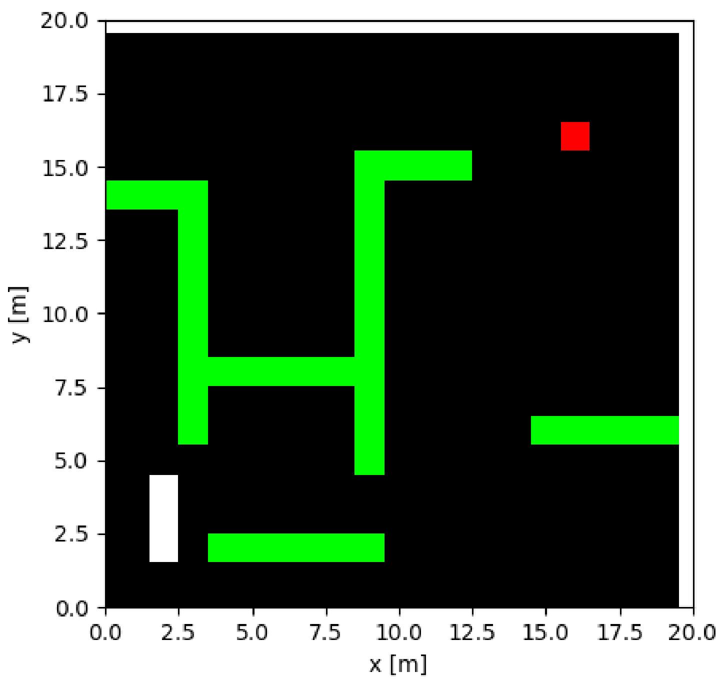
Pseudocode:

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:
  $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
  Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
  Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow$ average$(Returns(S_t, A_t))$
      $\pi(S_t) \leftarrow \mathrm{argmax}_a\, Q(S_t, a)$

Experimental Demonstration:



Question2

Please simulate the Sarsa (on-policy TD control)) algorithm under the given grid map environment.

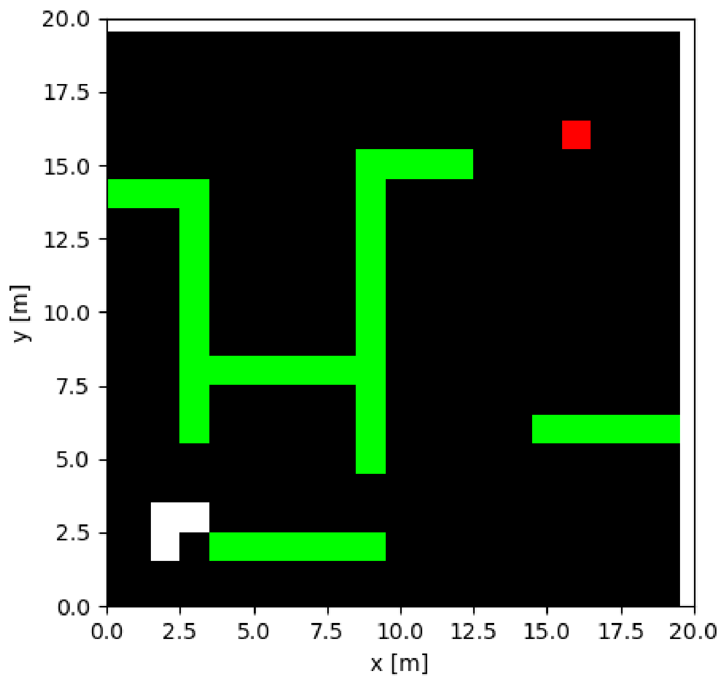
Pseudocode:

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma Q(S', A') - Q(S, A)\right]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Experimental Demonstration:



Question3

Please simulate the Q_learning (Off-policy TD Control) algorithm under the given grid map environment.

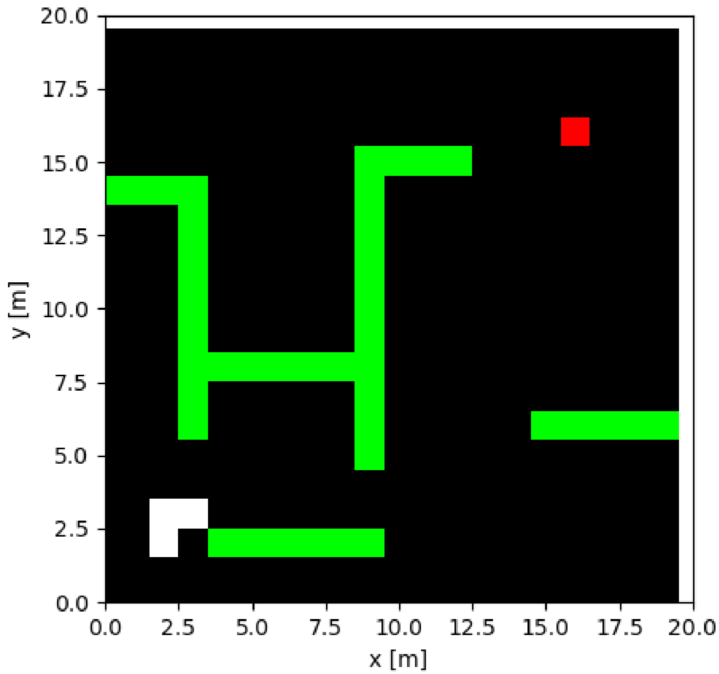
Pseudocode:

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Experimental Demonstration:



Note: The above demonstrations are only parts of the output policy. Normally, your solution should be different but moving to the goal is necessary.

Question4 - Extra Credit

Please add the heuristic reward on the grid map, such as the DWA reward, A star reward, or distance-to-goal based reward learned form the previous lectures, to achieve a regular policy as you expected.

Coding instruction

Install the intelligent robotics simulator

```
git clone -b edu https://github.com/hanruihua/intelligent-robot-simulator.git
cd intelligent-robot-simulator
pip install -e .
```

Note1: Please confirm that this repository is under the *edu* branch. You can use **git branch** to check current branch. If it is not under the *edu* branch, you can use **git checkout edu** to change current branch to *edu* branch.

Note2: The pycharm reduces the functionality of Matplotlib, which may lead to the failure of saving the gif animation. You can follow this link to solve this problem

Note3: If you have installed this simulator, you can use *git pull* to fetch the code update.

Code for questions

There are multiple files for these questions in the source folder.

- *question1_run.py*: is the main program you should run for question1
- *question2_run.py*: is the main program you should run for question2
- *question3_run.py*: is the main program you should run for question3
- *reinforcement_learning.py*: is the library to implement three reinforcement learning algorithms: Monte Carlo Exploring Starts, Sarsa, and Q-learning. You should fill in this file to complete these three algorithms for the questions.
- *grid_map.py*: is the file that defines the class about the grid map for you to use. You can add heuristic reward here for extra question4.
- *map_matrix.npy* and *reward_matrix.npy*: define the map and the reward in each grid.

You should complement the parts between ---- in the file **reinforcement_learning.py** for question1-3, and the file **grid_map.py** for extra question4. You can set the parameter *animation = True* in *question_run.py* to generate the animation.