112-1 (Fall 2023) Semester

Reinforcement Learning

Assignment #2-1

TA: Shang-Fu Chen (陳尚甫)

Department of Electrical Engineering
National Taiwan University

Outline

- Environment
- Tasks
  - First-visit Monte-Carlo Prediction
  - Temporal-difference Prediction TD(0)
  - Temporal-difference Prediction n-step TD
- Code structure
- Grading
- Submission
- Policy
- Contact

Environment

Grid World

State space

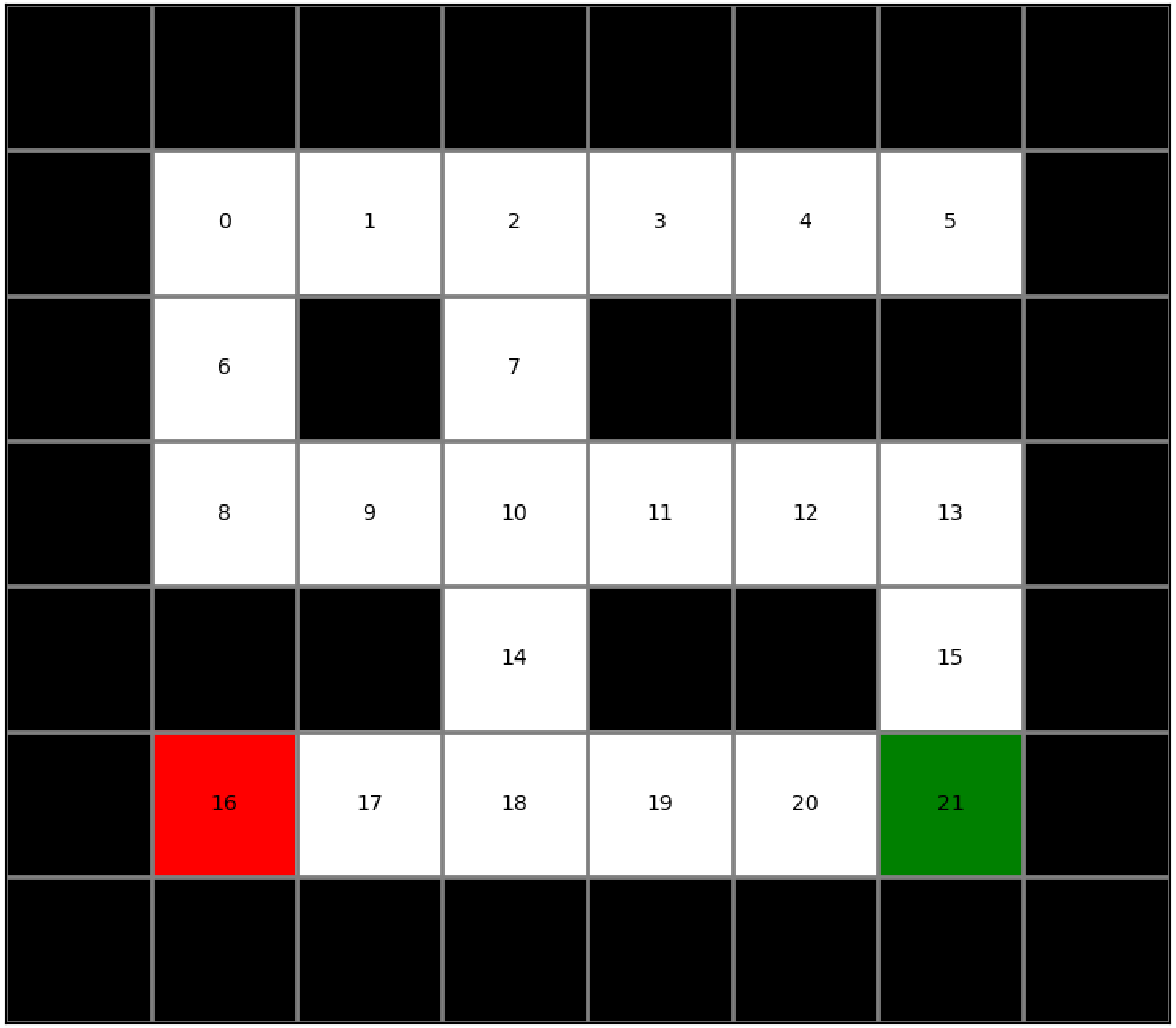
- Nonterminal states: Empty, Wall
- Terminal states: Goal, Trap
- 0-indexed

Action space

- Up, down, left, right
- Hitting the wall will remain at the same state

Reward

- Step reward given at every transition
- Goal reward given after reaching goal state
- Trap reward given after reaching trap state

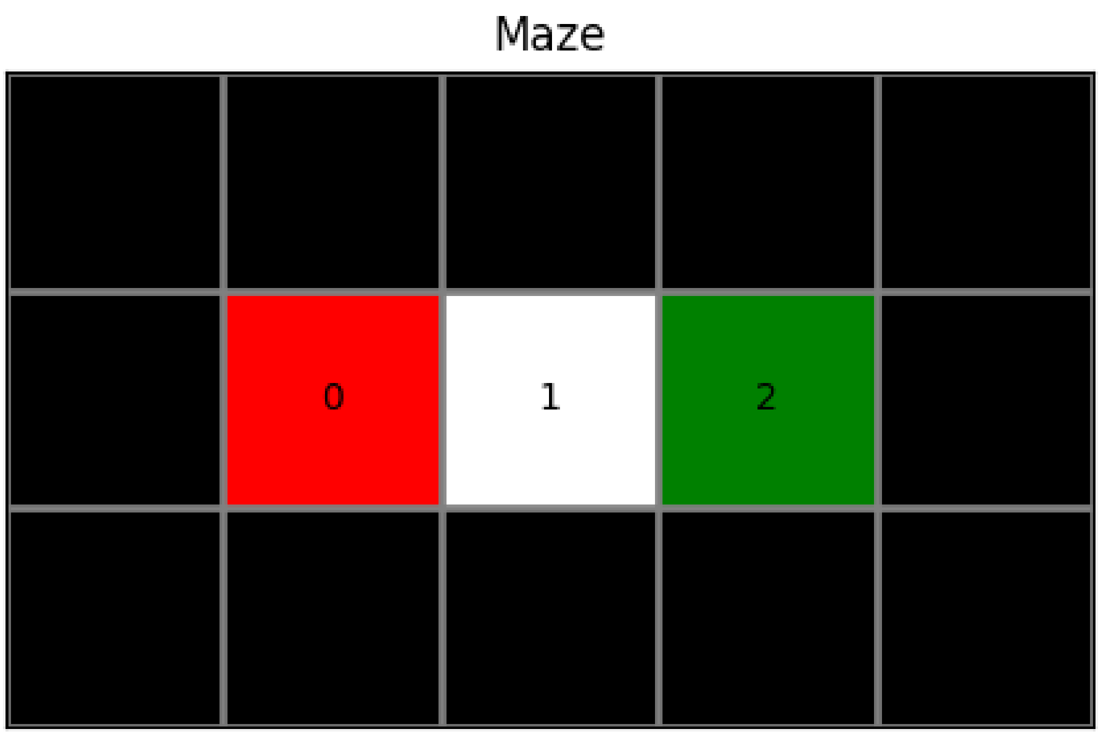


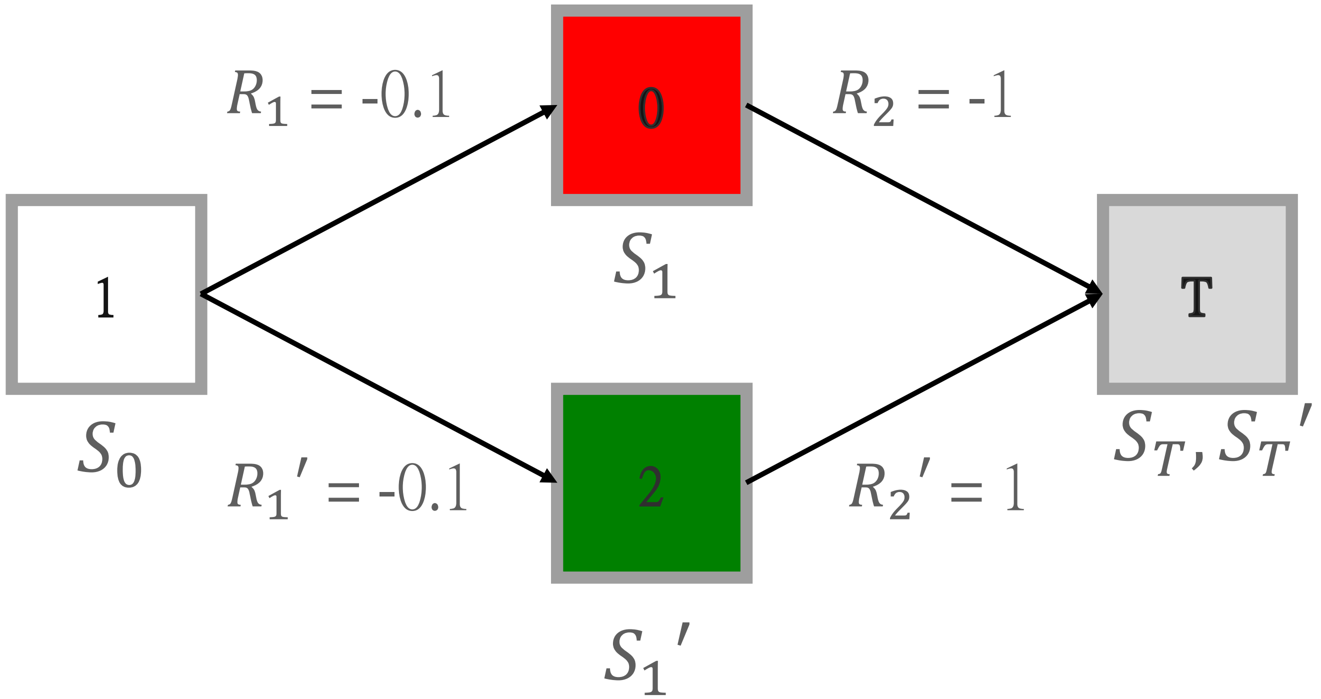
Interaction with Environments

- Learn to interact with a OpenAI gym-like environment
- Grid World in this assignment is a MRP (defined by maze.txt and traj.json, do not modified)
- Grid World functions:
  - step(): Interact with the environment
  - check(): Check if the exploration process end (explore 300 episodes)
  - reset(): Reset the environment to the initial state
  - Update the values function with states, rewards and done flags

```python
def run(self) -> None:
    """Run the algorithm until self.grid_world.check() == False"""
    # TODO: Update self.values with first-visit Monte-Carlo method
    current_state = self.grid_world.reset()
    while self.grid_world.check():
        next_state, reward, done = self.grid_world.step()
        continue
```

Terminal State





- The value of the terminal state is not considered in this assignment
- Size of self.values = 3 in the example

Tasks

Task 1 - First-Visit Monte-Carlo Prediction

- Evaluate a policy by predicting the value function for each state
- Update the value function with First-visit Monte-Carlo method using **state, reward, and done from the step() function.**
- Update the value function per episode
- Calculate the state value with the same order **returned by the step() function**

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Be care of the index of S and R !

(S_{T-1} is goal or trap in our case)

Loop forever (for each episode):

Generate an episode following π: $S_0, \cancel{A_0}, R_1, S_1, \cancel{A_1}, R_2, \ldots, S_{T-1}, \cancel{A_{T-1}}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Task 2 - TD(0)

- Evaluate a policy by predicting the value function for each state
- Update the value function with TD(0) method using **the step() function**
- Update the value function per step

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    ~~$A \leftarrow$ action given by π for S~~
    ~~Take action A,~~ observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

Task 3 - N-step TD

- Evaluate a policy by predicting the value function for each state
- Update the value function with n-step TD method using **the step() function**
- Update the value function **per step** expect steps that out of range of the n-step TD

n-step TD for estimating $V \approx v_\pi$

Input: a policy π
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer n
Initialize $V(s)$ arbitrarily, for all $s \in \mathcal{S}$
All store and access operations (for S_t and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | ~~Take an action according to $\pi(\cdot|S_t)$~~
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then $T \leftarrow t+1$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose state's estimate is being updated)
  | If $\tau \geq 0$: Skip n-1 step
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$ Be care of the index R !
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ Skip n-1 step $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

Text book p.144

Code Structure

DP_solver.py

class **DynamicProgramming**

- Parent class for DP algorithms

class **MonteCarloPrediction**

- TODO: run()

class **TDZeroPrediction**

- TODO: run()

class **TDNstepPrediction**

- TODO: run()

Feel free to add any function if needed

Grading

Grading

- First-visit Monte-Carlo prediction (10%)
  - Test cases (2% x 5 cases)
- TD(0) prediction (10%)
  - Test cases (2% x 5 cases)
- N-step TD prediction (10%)
  - Test cases (2% x 5 cases)

Criteria

- Test cases:
  - Call run() and check the final output
  - Check the state values after evaluation
  - Run time limit **3 minute** for each case to avoid infinite loops

- Sample solutions are provided for reference
  - Floating-point errors may occur due to the python version
  - State values should be exactly the same to the sample solutions if Python == 3.10.13
  - Mean error of state values < 0.005 (May be adjusted)

Submission

Submission & Report

- Deadline: 2023/10/19 Thu 09:30am
- No late submission is allowed
- Submission format and report format will be declared in Assignment #2-2

Policy

Policy

Package

- You can use any Python standard library (e.g., heap, queue···)
- System level packages are prohibited (e.g., sys, os, multiprocess, subprocess···) for security concern

Collaboration

- Discussions are encouraged
- Write your own codes

Plagiarism & cheating

- All assignment submissions will be subject to duplication checking (e.g., MOSS)
- Cheater will receive an **F** grade for this course

Grade appeal

- Assignment grades are considered finalized two weeks after release

Contact

Questions?

- General questions
  - Use channel <u>#assignment 2</u> in slack as first option
  - Reply in thread to avoid spamming other people
- Personal questions
  - DM us on Slack: <u>TA 劉冠廷 Guan-Ting Liu</u>

    <u>TA 陳尚甫 Shang-Fu Chen</u>