114-1 (Fall 2025) Semester

Reinforcement Learning

Assignment #1

TA: Chen-Yu Lin (林辰宇)

Department of Electrical Engineering
National Taiwan University

Outline

- Tasks
  - Iterative policy evaluation
  - Policy iteration
  - Value iteration
  - Async dynamic programming
- Environment
- Code structure
- Report
- Grading
- Submission
- Policy
- Contact
- Appendix

Tasks

Task 1 - Iterative Policy Evaluation

- Problem
  - Evaluate a given non-deterministic policy (probability distribution)
- Solution
  - Iterative application of Bellman expectation backup

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

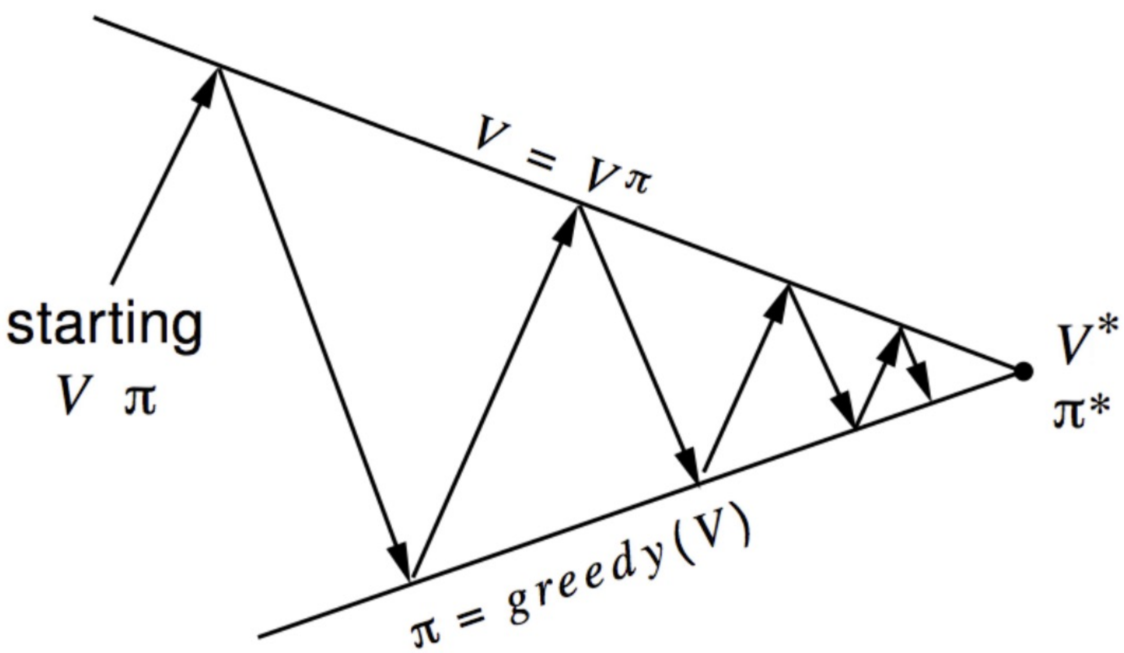
Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$

Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$ Synchronous update
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

[source]

Task 2 - Policy Iteration

- Problem
  - Find the optimal deterministic policy
- Solution
  - Policy evaluation: iterative policy evaluation
  - Policy improvement: greedy policy improvement
  - Eventually converges to optimal policy



[source]

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
       $\Delta \leftarrow 0$
       Loop for each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$ Synchronous update
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
       old-action $\leftarrow \pi(s)$
       $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
       If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Task 3 - Value Iteration

- Problem
    - Find the optimal deterministic policy
- Solution
    - Iterative application of Bellman optimality backup

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$ Synchronous update
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

[source]

Task 4 - Async Dynamic Programming

- Problem
  - Find the optimal deterministic policy with better efficiency
  - Less environment interaction
- Solutions
  - In-place dynamic programming
  - Prioritized sweeping
  - Real-time dynamic programming

Environment

Grid World

State space

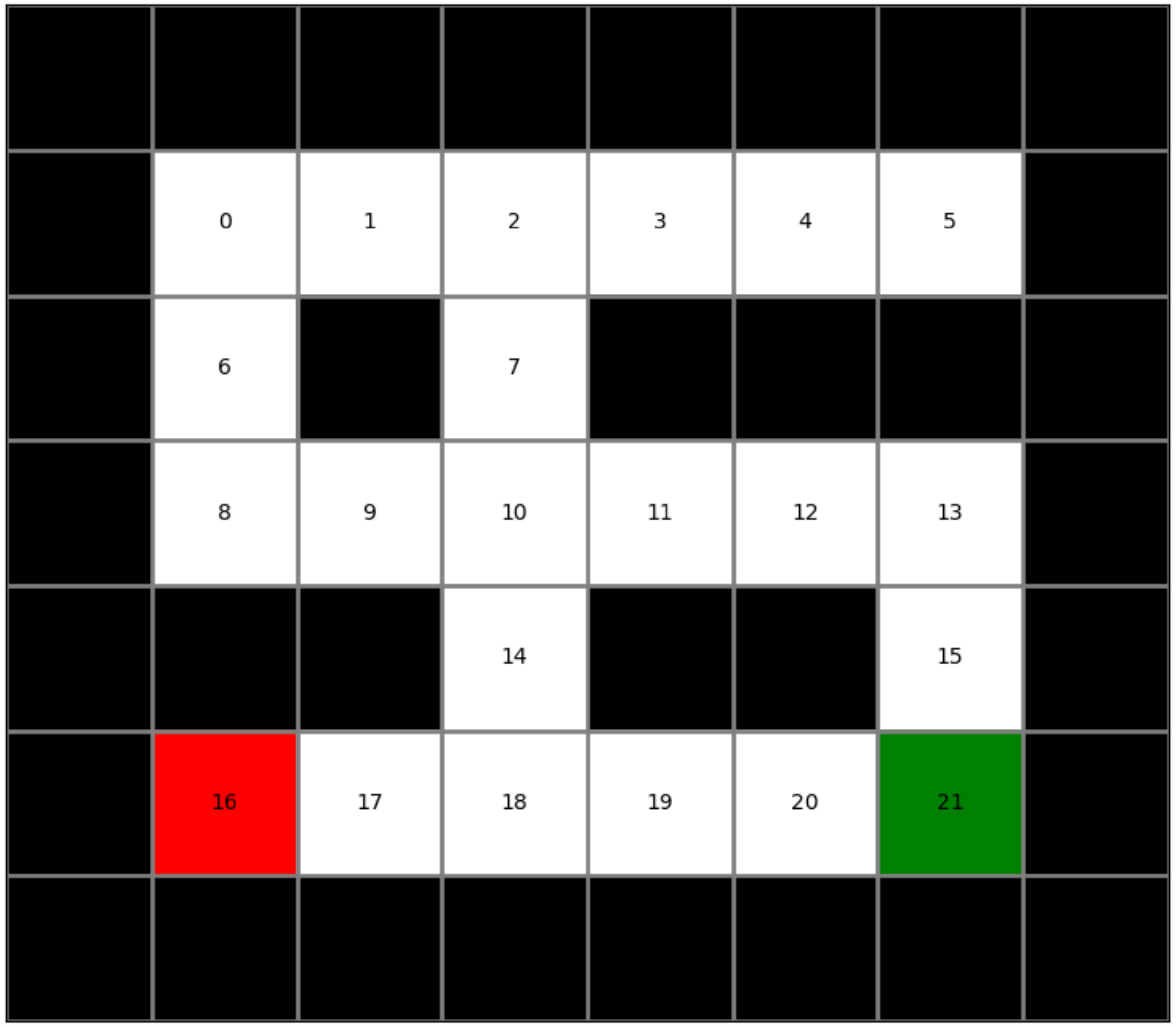
- Nonterminal states: Empty (White)
- Terminal states: Goal (Green), Trap (Red)
- 0-indexed

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



Done Flag

- Separator for episodes
- Return true from step when doing any action at terminal states
- Need to modify the Bellman equation
- Most gym-like environments also use this implementation

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_\pi(s'))$$

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_\pi(s')(1 - Done))$$

Code Structure

requirement.txt

- Conda

```
conda create -n rl_assignment_1 python=3.11
conda activate rl_assignment_1
pip install -r requirement.txt
```

- venv

```
python -m venv venv
source venv/bin/activate
pip install -r requirement.txt
```

DP_solver.py

class **DynamicProgramming**

- Parent class for DP algorithms
- TODO: get_q_value()

class **IterativePolicyEvaluation**

- TODO: get_state_value(), evaluate(), run()

class **PolicyIteration**

- TODO: get_state_value(), policy_evaluation(), policy_improvement(), run()

class **ValueIteration**

- TODO: get_state_value(), policy_evaluation(), policy_improvement(), run()

class **AsyncDynamicProgramming**

- TODO: run()

Feel free to add any function if needed, but you must include at least the run() function so that we can grade your code.

gridworld.py

- Methods:
  - get_action_space(): Get the dimension of action space
  - get_state_space(): Get the dimension of the state space
  - step(): Interact with the environment
  - reset(): Reset the environment
  - visualize(): Draw the maze with policy and values
  - run_policy(): Run the policy from given start state. Output the state history
- Step count:
  - Increment every time when step() method is called
  - Private member. Use get_step_count() to access.
- Step reward might not be constant at every state transition at private test cases
  - **You need to use step() to access the reward function**
- **Don’t try to modify or override any private member (double underscore prefix)**
- **You do not call reset() by yourself. We may rename the function when grading.**

main.py

- Methods:
  - run_policy_evaluation()
  - run_policy_iteration()
  - run_value_iteration()
  - run_async_dynamic_programming()
- These methods will call your written functions in DP_solver.py
- The output will be an image and the step counts for the algorithm

Report

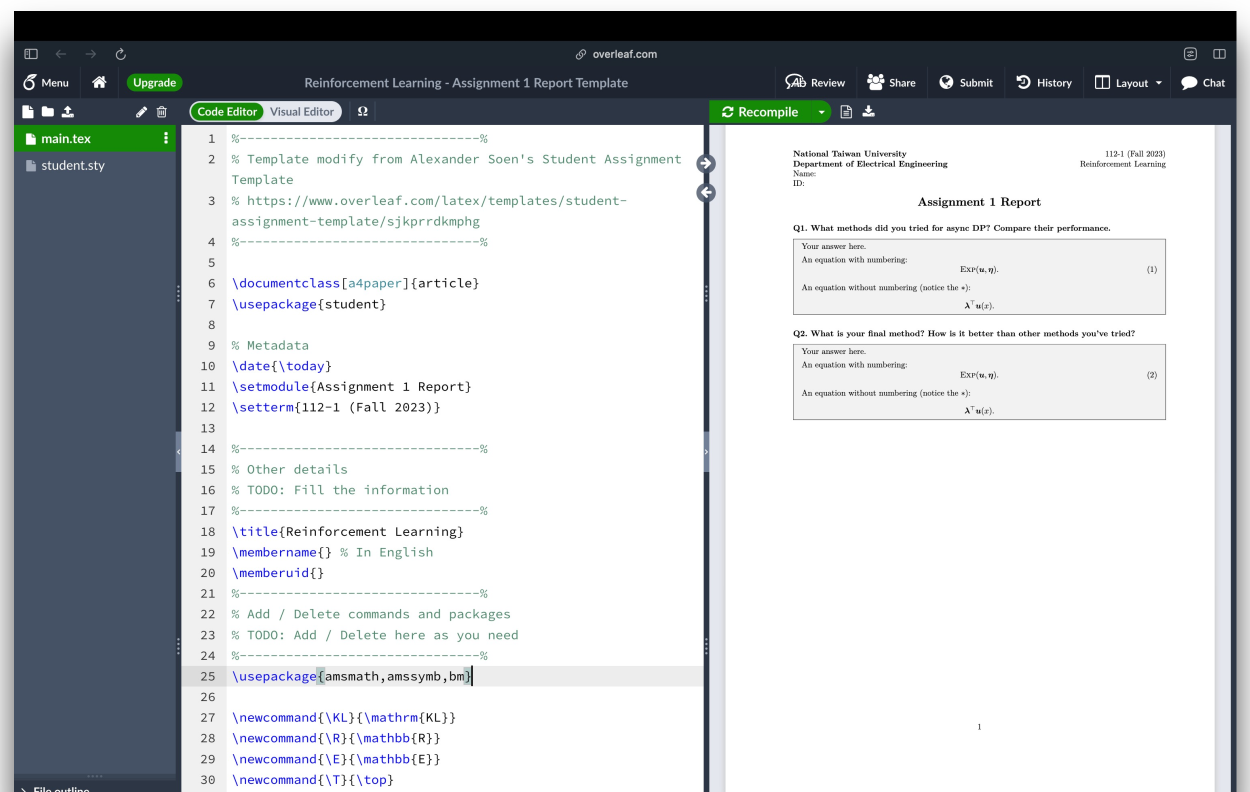
Report

- Q1. What methods have you tried for async DP? Compare their performance. (12%)
  - 4% per method tried with reasonable result and comparison
- Q2. What is your final method? How is it better than other methods you've tried? (8%)
  - 4% for reasonable explanation on how it's better
  - 4% for a novel method that outperforms the three methods mentioned in class

    (You need to point out how your method is different from them)
- LaTeX PDF format. Handwriting is forbidden.
  - [Overleaf template]
  - Write clearly and concisely in a few sentences
  - Practice using LaTeX for the final report

Overleaf

- Online LaTeX editor
- LaTeX
  - Good for math equations, tables and indexes
  - Widely used in paper writing and math writing
- Traditional Chinese will cause some compile problems
- [Official guide]



Grading

Grading

- Iterative policy evaluation (25%)
    - Test cases (5% x 5 test cases)

- Policy iteration (20%)
    - Test cases (4% x 5 test cases)

- Value iteration (25%)
    - Test cases (5% x 5 test cases)

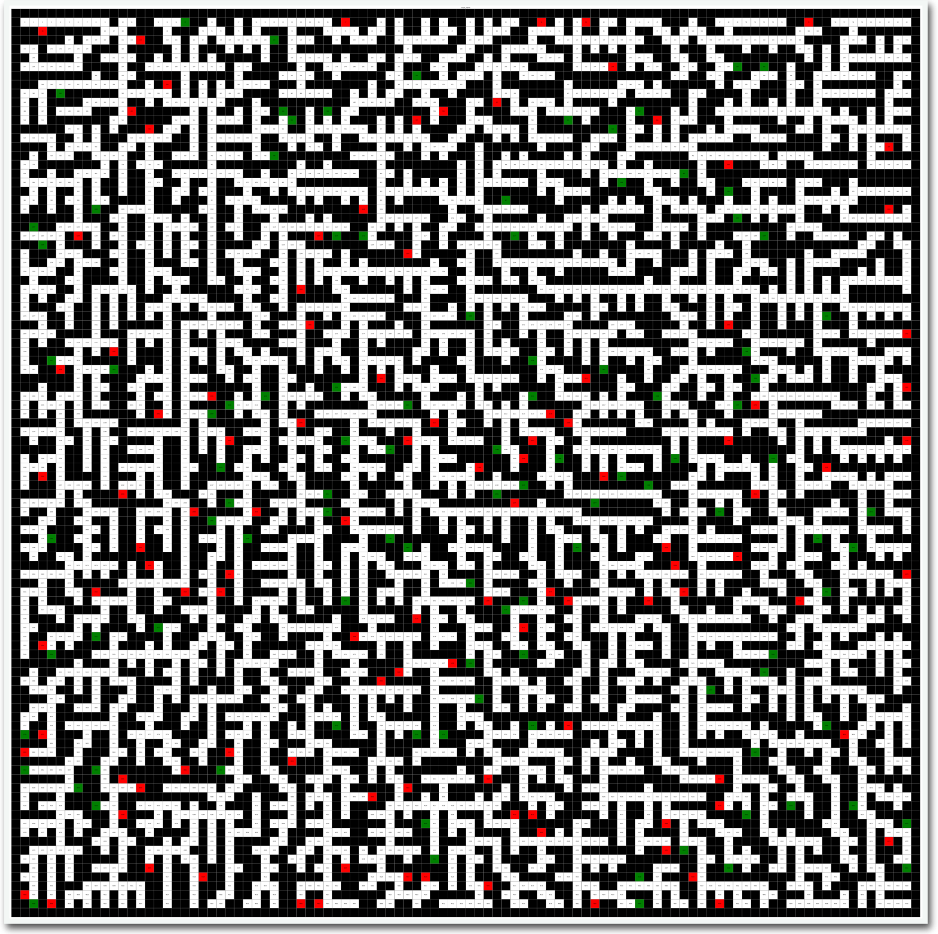
- Async dynamic programming (30%)
    - Better than your sync DP (5% x 2 test cases) (Both policy iteration and value iteration)

      Note 1: Your method must use fewer steps than your synchronous DP, and the policy must be optimal.

      Note 2: You only need to include the best-performing version of your method in the code you submit.
    - Report (20%)

Criteria

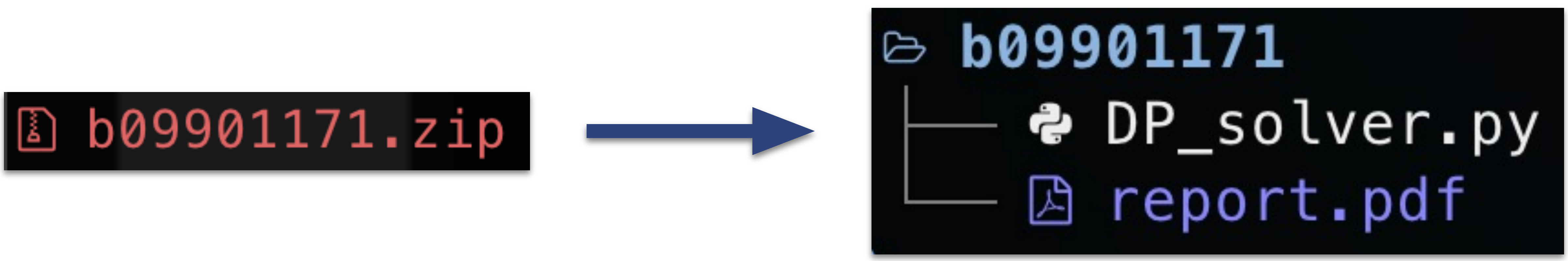
- Test cases:
  - Call run() and check the final output
  - Task 1: Check the values after evaluation
  - Task 2, 3, 4: Only check if the output policy is optimal
  - Run time limit **3 minute** for each case to avoid infinite loops
  - Up to 1500 states in private test cases
  - Only task 4 considers step count
- Sample solutions are provided for reference
  - Optimal policy might not be unique
  - In the sample solutions, the policy is derived by following the traversal order of the action index.



Submission

Submission

- Submit on NTU COOL with following zip file structure
  - Get rid of pycache, DS_Store, __MACOSX, etc.
  - Student ID with **lower case**
  - **10-point deduction** for wrong format (See appendix for common mistakes)



- **Deadline: 2025/10/01 Wed 11:59 PM**
- **No late submission is allowed**

Policy

Policy

Package

- You can use any Python standard library (e.g., heap, queue…)
- However, system level packages are prohibited (e.g., sys, os, multiprocess, subprocess, shutil, pathlib, …) for security concern

  Importing any one of them will result in 0 points for the whole assignment (even if you didn't call it)

Collaboration

- Discussions are encouraged
- Write your own codes

Plagiarism & cheating

- All assignment submissions will be subject to duplication checking (e.g., MOSS)
- Any student who cheats will receive an F grade in this course.

Policy

Policy on AI-Generated Code

- Use of AI tools is permitted, but you are responsible for any code they produce
- Code that match others will be considered plagiarism, regardless of source

Grade appeal

- Assignment grades are considered finalized two weeks after release

Contact

Questions?

- General questions
  - Use channel **#assignment** in slack as first option
  - Reply in thread to avoid spamming other people
- Personal questions
  - DM me on Slack: **TA 林辰宇 D14942012**
- To ensure everyone has equal access to the same clarifications, only questions before 2025/09/30 Tue 11:59 PM will be answered

Appendix

Appendix: Common Submission Mistakes

Several mistakes that may lead to score deductions, including but not limited to:

- Forgetting to remove your debugging codes and system level packages
  - Example: pdb.set_trace() → Get 0 points for all test cases
  - Importing system level packages (os, sys…) → Get 0 points for the whole assignment
- Did not check your submission files thoroughly:
  - Incorrect file names: report.pdf.pdf
  - Redundant files: .DS_Store, __MACOSX
  - Missing files
  - Format mistakes will cost 10 points from the assignment

Appendix: Zip

- We recommend zipping using `zip -r {your_id}.zip {your_id} -x  ‘**/.*’`
- Check your zip file with `unzip -l archive.zip`

```
❯ unzip -l b06901150.zip
Archive:  b06901150.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  09-09-2025 21:19   b06901150/
    15569  09-26-2024 03:58   b06901150/DP_solver.py
    53300  09-26-2024 04:26   b06901150/report.pdf
---------                     -------
    68869                     3 files
```

Correct

Incorrect

```
❯ unzip -l b06901150.zip
Archive:  b06901150.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  09-09-2025 21:19   b06901150/
      220  09-09-2025 21:19   __MACOSX/._b06901150
     6148  09-09-2025 21:19   b06901150/.DS_Store
      120  09-09-2025 21:19   __MACOSX/b06901150/._.DS_Store
    15569  09-26-2024 03:58   b06901150/DP_solver.py
      176  09-26-2024 03:58   __MACOSX/b06901150/._DP_solver.py
    53300  09-26-2024 04:26   b06901150/report.pdf
---------                     -------
    75533                     7 files
```

(Note: There will be an extra __MACOSX folder if you are zipping the files using macOS’s UI)