CSIS, BITS Pilani K. K. Birla Goa Campus

Artificial Intelligence (CS F407)

Programming Assignment 2
Total Marks: 15
Submission Deadline: 9 PM on 29/10/2025 (Wednesday)

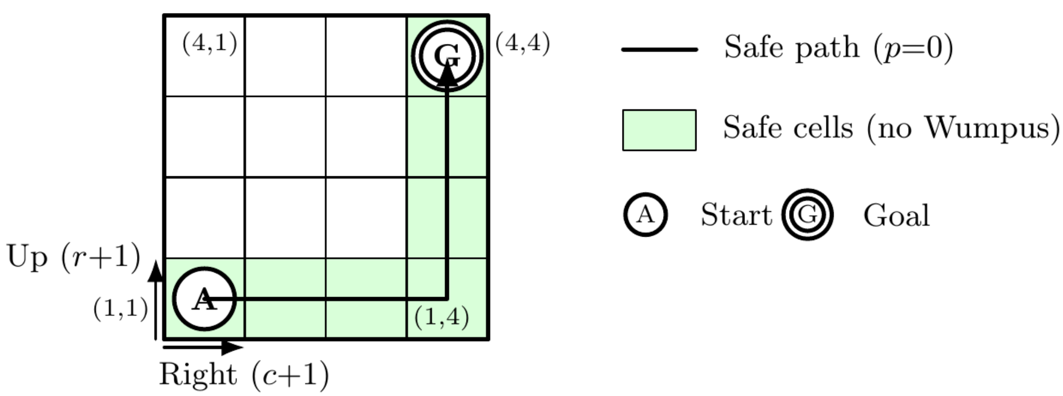
Each student must complete this assignment individually. Your program must be written in Python and should run without errors on Python 3.10.

Plagiarism Warning: Any form of plagiarism will result in **zero marks for everyone involved**. No distinction will be made between minor and major cases.

Late Submission Policy: The deadline is strictly **9 PM**. Late submissions will incur a penalty of 5 marks per day. Submit early to avoid issues such as power or internet failures.

Question 1 (15 marks)

Description of Magic Wumpus World



- **Grid and coordinates.** The world is a fixed 4×4 grid. Coordinates are *1-indexed* as (row, col) with the bottom-left cell $(1, 1)$ denoted **A** (start) and the top-right cell $(4, 4)$ denoted **G** (goal). Moving `Up` increases the row; moving `Right` increases the column.
- **Actions.** At each time step the agent chooses one of four actions {`Up`, `Down`, `Left`, `Right`}. Transitions are deterministic. If an action would move the agent off the grid, the agent stays in place (*invalid move*).
- **Wumpus hazard.** Entering any cell *not* on the designated safe path triggers a Bernoulli trial with probability p. If the Wumpus appears (success), the agent **dies** and the episode ends immediately.

- **Safe path.** Exactly one path from **A** to **G** is marked *safe*: on these cells the Wumpus never appears ($p = 0$). All other cells share the same Wumpus appearance probability p.
- **Rewards and termination.**
  - *Step cost:* -1 on every time step (including invalid moves).
  - *Wumpus:* If the Wumpus appears, reward -100 and the episode terminates.
  - *Goal:* Entering **G** gives reward -1 and terminates the episode.
- **Return (cumulative reward).** The assignment uses undiscounted return ($\gamma = 1$): the episode return is the sum of rewards until termination. The environment prints the cumulative reward at the end of each episode.

Intuition. The optimal behavior is to discover the shortest safe route from **A** to **G** (maximum return), balancing exploration with the high cost of encountering the Wumpus.

Note on `MWW.json`, `MagicWumpusWorld.py`, and `ROLLNO_NAME.py`

- `MWW.json`
  - Located in the current folder. Contains only:
    ```
    {
      "p": 0.20,
      "safe_path": [[1,1],[1,2],[1,3],[1,4],[2,4],[3,4],[4,4]]
    }
    ```
  - `p` is the Wumpus appearance probability used for *all* non-safe cells.
  - `safe_path` is a list of 1-indexed coordinates forming a contiguous 4-neighbour path from **A** $(1, 1)$ to **G** $(4, 4)$ without duplicates.
  - During evaluation, multiple `MWW.json` files will be used. Your program must work for any valid file of this format generated by `generate_mww.py` program.
- `MagicWumpusWorld.py` **(do not modify).**
  - Reads `MWW.json` from the current folder and fixes all other environment details (grid size, start/goal, actions, rewards).
  - Public API used in this assignment:
    * `CurrentState() -> (row, col) or None`
    * `TakeAction(a: str) -> (reward: int, next_state: (row,col) or None)`, where `a` is one of `"Up","Down","Left","Right"`
    * `CumulativeRewardAndSteps() -> (reward, steps)`
    * `reset(seed: Optional[int]) -> (1, 1), i.e State A`
  - Behavior summary: invalid moves keep the agent in place with reward -1; entering **G** ends the episode with reward -1; entering a non-safe cell triggers Wumpus with probability p (if triggered: reward -100 and termination). The environment prints a standardized step line for every action, and on termination prints a short message and the episode's cumulative reward.

- `ROLLNO_NAME.py` **(students *must* modify and submit only this file).**
  - This is your solution file. It should import and use the provided `MagicWumpusWorld` class, run your learning algorithm, and print the required outputs in the specified format.
  - Submit only the `ROLLNO_NAME.py` file. Do **not** submit `MWW.json` or `MagicWumpusWorld.py` files.
  - Ensure your program terminates within the stated time limit and works across multiple `MWW.json` test cases.

Report Generation

Goal. Run controlled experiments in the Magic Wumpus World to understand how TD control algorithms learn a shortest safe path from **A** to **G**.

How to proceed.

1. Execute `ROLLNO_NAME.py`, and understand the output. `ROLLNO_NAME.py` imports the provided `MagicWumpusWorld` class and interacts with the Magic Wumpus World environment. Understand the code given in `ROLLNO_NAME.py`.
2. Implement the baseline TD control algorithms (for analysis and comparison in the report):
   - **SARSA (on-policy):** target $= r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1})$.
   - **Q-learning (off-policy):** target $= r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')$.
   - **Expected SARSA (on-policy expectation):** target $= r_{t+1} + \gamma \sum_{a'} \pi(a' \mid s_{t+1})\, Q(s_{t+1}, a')$.

   The TD error is $\delta = \text{target} - Q(s_t, a_t)$ and the update is $Q \leftarrow Q + \alpha\,\delta$.
3. Compare action-selection strategies:
   - ϵ-**greedy** (use both constant and decaying ϵ),
   - **Optimistic** initial Q-values,
   - **UCB**-style exploration (maintain state–action counts to compute a confidence bonus).
4. Study hyperparameters: vary γ and α (and any exploration parameters) and observe learning speed, stability, and asymptotic performance.

What to plot (label axes, include legends):

- **Regret per episode** (lower is better):

$$\text{regret} = -6 \; - \; (\text{cumulative reward in the episode}).$$

  In this task, -6 is the (hidden) return of the optimal shortest safe path; as learning improves, regret should approach 0.

- **Average TD-error magnitude** per episode, e.g., $\frac{1}{T}\sum_{t=1}^{T}|\delta_t|$ for that episode.
- You can create additional plots to test and compare various ideas.

Notes. Use fixed random seeds when comparing methods, and keep all settings (other than the factor you are studying) identical across runs to ensure fair comparisons.

Requirements

1. **Baseline TD algorithms (for analysis in the report).** Implement *SARSA*, *Q-learning*, and *Expected SARSA* to produce the plots and comparisons requested in *Report Generation*. These baselines help you understand learning behavior on the Magic Wumpus World.
2. **Improved TD algorithm (to submit).** Propose and implement a TD-based variant that learns the safe path from **A** to **G** as quickly and reliably as possible (maximize cumulative return, minimize episodes/steps to near-optimal behavior, and maintain stability).
3. **Program constraints.**
   - You must use the provided `MagicWumpusWorld` class, which reads the safe path given in `MWW.json`.
   - Your program must terminate within **45 seconds** on the evaluation machine. It may stop earlier if it converges to near-optimal behavior.
   - Print the following when your program terminates:

     ```
     Mean cummulative reward in the last 10 episodes :  -XXX
     Time taken by the algorithm : YY Seconds
     ```
   - Do **not** modify `MagicWumpusWorld.py`. Submit only `ROLLNO_NAME.py`.
4. **Report (PDF).** Explain your improved TD algorithm, the design choices (exploration, step-size schedule, initialization, etc.), and why it outperforms the baselines. Include the requested plots with a brief discussion.

How the assignment will be evaluated

- **Algorithm performance (on unseen `MWW.json`).** Higher marks for improved TD methods that reliably approach the optimal policy in less time (must meet the **45 s** limit). Marks for the algorithm will be based on the following two criteria:

  ```
  Mean cummulative reward in the last 10 episodes
  Time taken by the algorithm
  ```
- **Quality of analysis and report.** Clearly present experimental setup, hyperparameters, and comparisons to the three baselines. Include well-labeled plots (regret, average $|\delta|$, etc.) and short, focused takeaways that justify your claims.

- **Code hygiene.** Readable code, and correct output format. Marks will be deducted for not following instructions. Only `ROLLNO_NAME.py` should be changed and submitted.
- **Overall grading.** The improved TD implementation and the report carry **7.5 marks each**. Strong, well-argued negative results (ideas that did not work and why) are valued if they are carefully analyzed.

Instructions for Submission

- Submit exactly **two files**:
  1. Your Python program: `ROLLXYZ_FIRSTNAME.py`
  2. Your report: `ROLLXYZ_FIRSTNAME.pdf`
- Use only capital letters in filenames. Example: `2020H1030999G_ADARSH.py`, `2020H1030999G_ADARSH.pdf`.
- Your program must:
  - Implement only the improved TD Algorithm.
  - Use the provided MagicWumpusWorld class.
  - Terminate within 45 seconds.
  - Produce output in the format shown below:

    ```
    Mean cummulative reward in the last 10 episodes :  -XXX
    Time taken by the algorithm : YY Seconds
    ```
- Multiple `MWW.json` test files are provided; check that your program works correctly on them.
- Submit files directly on Quanta. **Do not zip**.
- Report any bugs in `MajicWumpusWorld.py` to the course IC.