Solving a 10×10 Maze with Sub-goal using Reinforcement Learning

Project Overview:

In this project, you will design and implement reinforcement learning (RL) algorithms to solve a 10×10 maze with a sub-goal. The maze consists of walls and open spaces, where the agent (player) starts at a specific point and needs to navigate through the maze to reach the final goal. A sub-goal is introduced in the maze to promote intermediate progress, requiring you to adjust your RL algorithms to first reach this sub-goal before advancing to the final goal.

You are encouraged to select and explore one of the following RL algorithms:

- Value Iteration
- Policy Iteration
- Q-learning
- Policy Gradients
- Actor-Critic

The use of neural networks is not required. You can focus on tabular methods or linear function parameterization. The project is expected to take 25 days (deadline: October 20th 23:59 Amsterdam time), which includes developing, testing, and refining the algorithms, as well as completing a final report of up to 12 A4 pages, with a minimum font size of 10.

This is a group project with a maximum of 4 members per group. You are free to form your own group, but please specify the individual contributions of each group member in the final report.

Deliverables:

You are required to submit your code and report before the deadline.

- **Code**: A complete Python (or other programming languages that you prefer) implementation of the RL algorithm.
- **Report**: You are required to provide a report that includes:
    - The rationale behind your choice of reinforcement learning algorithm (e.g., why you prefer policy iteration over value iteration);
    - How your selected RL algorithm was adapted to include sub-goal.
    - Evaluation on your learned agent (e.g., visualizations showing agent paths, number of steps to reach the goal, convergence speed, etc.).
    - Your findings and what are their implications to the design of RL algorithms to more complex real-world problems.
    - Individual contribution of each group member.

Tools & Libraries:

Suppose you choose Python, you can consider:

- **OpenAI Gym**: To create the maze environment.
- **NumPy**: For maze generation and matrix operations.
- **Matplotlib/Seaborn**: For visualizing results.
- **PyTorch/TensorFlow**: (Optional) For deep learning if using DQN.

Grading Criteria:

- **Implementation** (40%): Correctness of your algorithm (e.g., weather the agent can reach the sub-goal before advancing to the final goal).
- **Analysis** (40%): Justification on your chosen algorithm and an analysis of its performance quality.
- **Presentation** (20%): Clarity of the final report and visualizations.

Maze Layout:

The maze is a 10×10 grid with walls and open paths. The layout of the maze is as follows, which can be generated programmatically using libraries like numpy or gym.

- The maze is a 10x10 grid where:
  - **1** represents a wall (unpassable).
  - **0** represents an open path (passable).
  - **S** marks the starting position of the agent.
  - **G** marks the sub-goal the agent should reach before continuing.
  - **E** marks the end goal.

1	1	1	1	1	1	1	1	1	1
1	S	1	0	0	0	1	0	0	1
1	0	1	0	1	0	1	0	1	1
1	0	0	0	1	0	0	0	0	1
1	1	1	0	1	1	1	1	0	1
1	0	1	G	0	0	0	1	0	1
1	0	0	0	1	1	0	0	0	1
1	1	1	0	1	0	1	1	0	1
1	0	0	0	0	0	1	E	0	1
1	1	1	1	1	1	1	1	1	1