56 Silver

新モンゴル高専
Bilguuntushig Amarsaikhan
Dulguun Zolzaya
Shur-Erdene Buyannemekh（教員）

1. Problem:
The problem was a board game that takes points by seizing territories by building walls, playing between two players. Because of that, we explored a few ways that are written down below.

2. Exploration:
We tried Brute force methods like minimax and encountered issues in large-state space games due to the exponential growth of possibilities. These methods exhaustively search through all potential moves, becoming impractical for games with complex branching and deep states. So it's not compatible in our case. After that, we studied AI.

3. Why Reinforcement Learning from all other AI learning methods?
Reinforcement Learning (RL) is well–suited for board games due to its ability to learn strategies through interactions. Unlike supervised learning, which lacks feasible labeled data, and unsupervised learning, which lacks feedback, RL navigates game challenges.

4. Why Monte Carlo Tree Search for Board Games like ours (from all other RL algorithms)?
Monte Carlo Tree Search (MCTS) is favored for board games due to its balanced exploration-exploitation strategy, adaptability to partial information, simulation-driven decision-making, iterative improvement, and suitability for sequential gameplay. Unlike other RL algorithms, MCTS excels in complex game scenarios where long-term strategy, uncertainty, and dynamic decision sequences are pivotal, making it a powerful tool for mastering various board games.

5.Main learning algorithm: Self-Play + Monte Carlo Tree Search + Deep Neural Network
In self-play with MCTS and deep neural networks, agents learn by competing against each other. MCTS guides decision-making based on simulations, and deep neural networks estimate values and policies. Through iterations of self-play, agents improve their strategies, making this approach effective for mastering complex board games.

6. Tools and environment
Language: Python, C++,
Deep Learning Frameworks: TensorFlow, PyTorch
RL Libraries: OpenAI Gym, Stable Baseline
Simulators and Environments: We made our game in Python using pygame library, and Jupyter notebook