311505026 鄭采玲

i. [30 pts total] Explain how you implement Minimax algorithm [15 pts] and MCTS [15 pts] for this game in detail?

minimax:

基本結構:

- minmax 函數接受六個參數：depth（搜索深度）、isMaximizingPlayer（是否為最大化玩家）、alpha 和 beta（Alpha-Beta 剪枝）、color（當前玩家顏色）、hexagon_board（六角棋盤狀態）。
- 如果深度為 0 或已經沒有剩餘可選擇的六邊形（由 display_remaining_hexes() 函數決定），那麼就計算並返回當前玩家的連通區域數量作為評估值。

- 函數會根據六邊形的標籤組織所有未被選擇（selected）且未被預訂（booked）的六邊形。
- hexes_by_label 是一個字典，將每個標籤對應到一組六邊形位置及其信息。
- available_labels 是一個過濾後的字典，只包含至少一個未被選擇的六邊形的標籤。

最大化和最小化步驟:

- 若當前是最大化玩家，則遍歷所有可用的標籤並嘗試所有可能的六邊形組合。
- 每個組合都會被標記為已選擇、已預訂並設置為當前玩家的顏色。
- 然後切換顏色並遞歸調用 minmax 函數，減少深度且變為最小化玩家。
- 根據遞歸調用的返回值來更新當前的最佳評估值（bestVal）和最佳移動（best_move）。
- 使用 Alpha-Beta 剪枝來剪去不必要的搜索分支。
- 遍歷完所有組合後，返回當前的最佳評估值和最佳移動。
- 最小化玩家的邏輯類似，只是評估值和剪枝的方向相反。

Alpha-Beta 剪枝:

- alpha 代表最大化玩家的最佳已知下界。
- beta 代表最小化玩家的最佳已知上界。
- 當 beta 小於等於 alpha 時，剪去後續的分支，因為這些分支不會影響最終決策。

MCST:

TreeNode 類別表示 MCTS 中的樹節點。每個節點包含以下屬性：

- state：當前節點的遊戲狀態。
- parent：父節點。
- children：子節點列表。
- visits：該節點被訪問的次數。
- reward：從該節點得到的累計獎勵。

方法：

- is_fully_expanded()：判斷節點是否完全展開（所有可能的子節點都已創建）。
- best_child(exploration_weight=1.4)：根據 UCT（Upper Confidence Bound for Trees）公式選擇最優的子節點。
- most_visited_child()：返回訪問次數最多的子節點。

MCTS 類別實現了蒙特卡洛樹搜索演算法。

方法：

- __init__(self, iterations)：初始化 MCTS 物件，設置迭代次數。
- search(self, initial_state)：開始 MCTS 搜索，從初始狀態開始，進行給定次數的迭代，返回最優子節點的狀態。
- _select(self, node)：選擇節點以進行展開或模擬。沿著 UCT 最優路徑向下選擇，直到找到未完全展開的節點。
- _expand(self, node)：展開節點，創建新的子節點。
- _simulate(self, state)：從當前狀態進行隨機模擬，直到達到終局狀態，返回模擬結果的獎勵值。
- _backpropagate(self, node, reward)：回傳獎勵值，更新節點的訪問次數和獎勵值。

HexagonBoardState 類別表示六角棋盤的狀態。

屬性：

- board：棋盤狀態，是一個字典，鍵為位置，值為六邊形的信息。
- color：當前玩家的顏色。
- depth：當前狀態的深度。

方法：

- get_possible_moves()：返回所有可能的下一步狀態。
    - 遍歷所有未被預訂（booked）且未被選擇（selected）的六邊形，按標籤分組。
    - 對每個標籤生成所有可能的組合，對每個組合創建新狀態，並加入可能的移動列表。
- is_terminal()：判斷當前狀態是否為終局狀態。
- get_reward()：返回當前狀態的獎勵值。

遊戲流程

1. 初始化 MCTS 和根節點：
    - 使用初始狀態創建 MCTS 和根節點。
2. 迭代搜索：
    - 重複指定次數的搜索過程：
        - _select：從根節點開始，選擇最優子節點。
        - _expand：展開選擇的節點，創建新的子節點。
        - _simulate：對新狀態進行隨機模擬，獲得獎勵值。
        - _backpropagate：回傳獎勵值，更新節點的訪問次數和獎勵值。
3. 返回最優行動：
    - 搜索結束後，返回訪問次數最多的子節點的狀態作為最優行動。

ii. [10 pts total] Explain the most diHicult thing you faced when implementing the code. Why? [5 pts] And how you solved it? [5 pts]
我遇到最大的困難是界定下每個棋子的分數為多少，因為這個影響前期不具至關重要。我覺得最好的解決方法是 GAN，但我能力不足不會寫，只好慢慢嘗試，嘗試防禦為幾分，攻擊為幾分，分布為幾分，慢慢嘗試去解決問題。

iii. [10 pts total] Compare Minimax algorithm and MCTS, what is the diHerence between them? [5 pts] Which one do you think is better for “Strands”? [5pts]
Minmax 可以很好的找到最佳解，但是他能計算的量有限，只能到範圍縮到很小的時候再使用。蒙地卡羅就比較 free，他可以探索的方式慢慢嘗試最佳解在哪，可以探索的範圍會比 minimax 還廣，但是就不保證結果，一切都是機率問題了。我覺得 minimax 比較好，因為 minimax 前期不能使用的地方還能用策略 cover