1. Introduction

姓名: 蔡懷恩
學號: 112550020
透過給分和扣分來教導如何選擇，實作一個能夠玩connect Four的AI，並比較Minimax Search, Alpha-Beta Pruning的區別，然後製作一個比Alpha-Beta Pruning更強的AI

2. Implementation

2.1. Minimax Search

1. 玩家1先手，在玩家1,2之間輪流，玩家1以分數最大值為目標，玩家2以分數最小值為目標，直到遊戲結束(一方獲勝或平手)。假設對手以最佳化為目標，AI不斷遞迴直到"棋局結束"或"深度4"，用get_heuristic()評估局面，以自己能得到的一樣最佳值的下法之一去下
2. 從有效下法中得到heuristic值之後，以玩家1/玩家2的角度去選出能取得最大值/最小值的方法，回傳(最佳值,能得到最佳值的下法)，持續遞迴直到遊戲結束
3. get_heuristic() 以玩家1的角度給當前局面評分，如果對玩家1有利就高分，反之低分

result:

```
Game 100/100 finished.
execute time 1257064.91 ms
Summary of results:
P1 <function agent_minimax at 0x7fbbfa5a1f80>
P2 <function agent_reflex at 0x7fbbfa5a20c0>
{'Player1': 100, 'Player2': 0, 'Draw': 0}
========================================
        DATE: 2025/04/03
        STUDENT NAME: 蔡懷恩
        STUDENT ID: 112550020
        ========================================
```

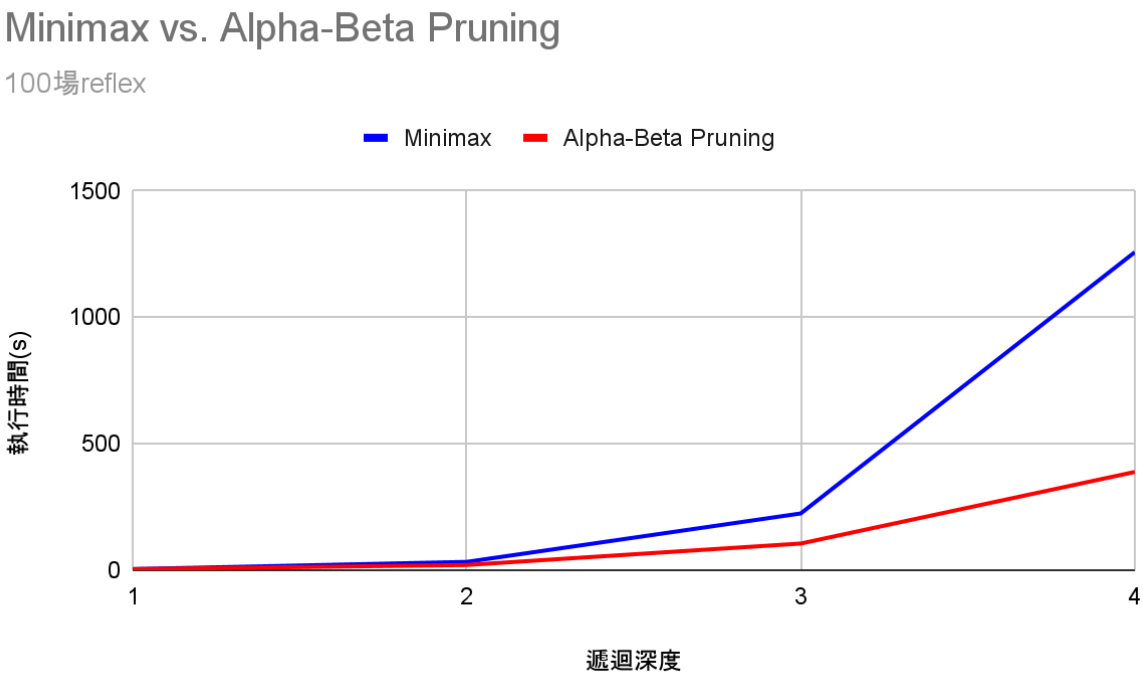
2.2. Alpha-Beta Pruning

1. α: 目前已知「Max player 的最佳選擇」
   β: 目前已知「Min player 的最佳選擇」
   如果玩家1發現玩家2現在可以拿到的min值比先前自己能從玩家2決定後得到的Max值還小時，就捨棄這一條路徑，因為可以大致確定不會選這條路徑
2. β <= α時 剪枝

result:

```
Game 100/100 finished.
execute time 388182.01 ms
Summary of results:
P1 <function agent_alphabeta at 0x7f750459e020>
P2 <function agent_reflex at 0x7f750459e0c0>
{'Player1': 95, 'Player2': 5, 'Draw': 0}
========================================
        DATE: 2025/04/03
        STUDENT NAME: 蔡懷恩
        STUDENT ID: 112550020
        ========================================
```

plot:



當遞迴深度越深，minimax和alpha-beta Pruning會越差越多，但兩者的勝率也會有一點差別

2.3. Stronger AI Agent

我AI的部分沿用alpha-beta Pruning，只修改了heuristic函數
增加了中心控制，越中間權重越高，player1的棋子加分，player2的棋子扣分，因為四子棋這個遊戲越中間越有利，所以能夠勝過alpha-beta Pruning

result:

```
Game 100/100 finished.
execute time 1629048.56 ms
Summary of results:
P1 <function agent_alphabeta at 0x7fb56e7a6020>
P2 <function agent_strong at 0x7fb56e7a6160>
{'Player1': 31, 'Player2': 61, 'Draw': 8}
========================================
        DATE: 2025/04/03
        STUDENT NAME: 蔡懷恩
        STUDENT ID: 112550020
        ========================================
```

勝率有六成

3. Analysis & Discussion

1. difficulties: 要去研究遊戲的勝利手段，讓AI去使用
2. agent_strong() 比起 alpha-beta Pruning 耗的時間更長
3. 可以再加入更多策略，如優先防守對手勝利，
   或是能察覺可以一次創造多個3連

4. Conclusion

我了解了 adversarial search 的核心邏輯，以及如何設計一個能夠評估盤面、預測對手、並做出最有利選擇的AI。
也透過 Stronger AI Agent 理解了 heuristic 的重要性，好的heuristic能夠更有效的幫助AI做出好的判斷

5. References

chatgpt, friends explanation