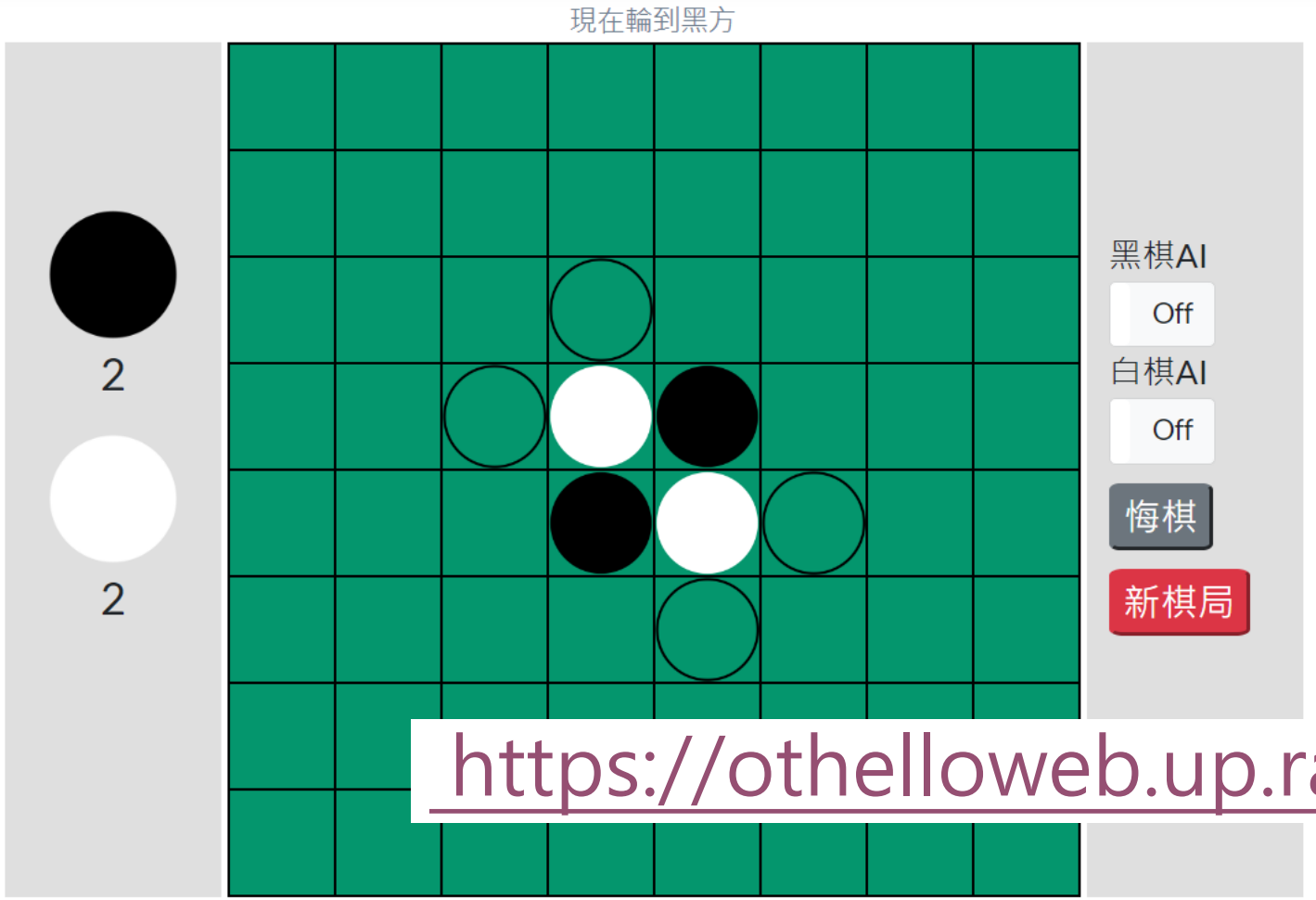
Othello Web

顏少于 C++選修課小專題2023

功能展示

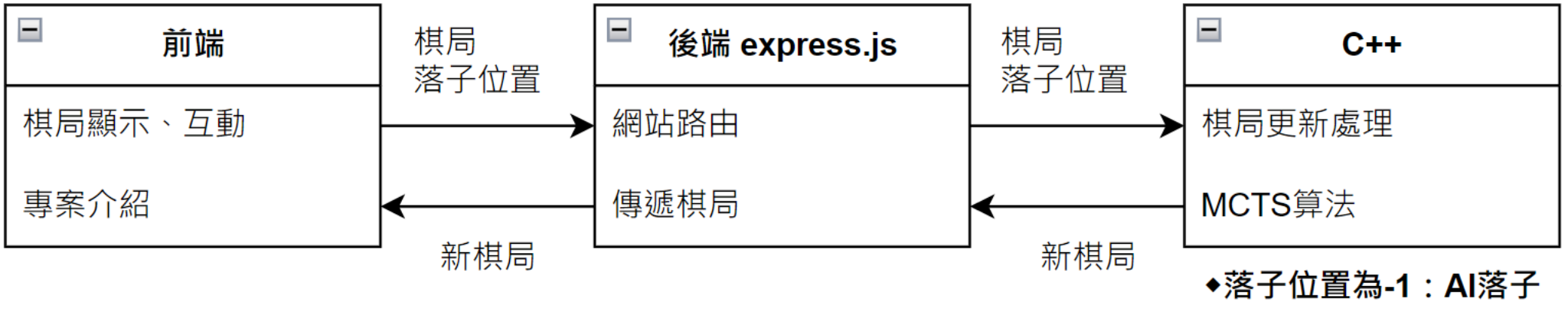
Othello MCTS　首頁　專案介紹　C++終端機版

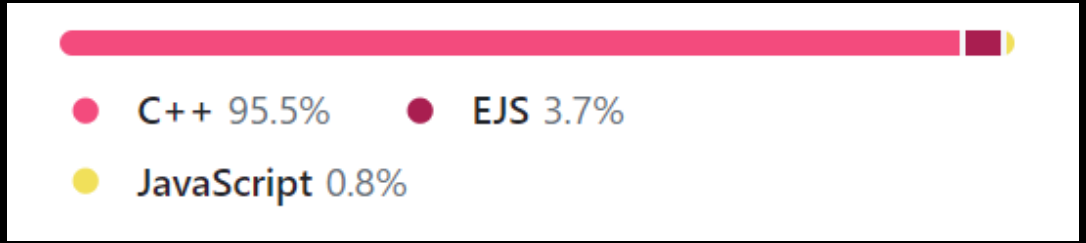


https://othelloweb.up.railway.app/

專案架構

介紹C++內容為主

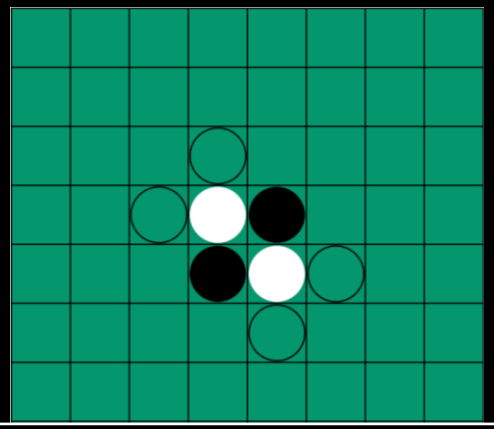




程式運作流程(棋局更新處理)

棋局State

下棋方=黑
合法步=[19, 26, 37, 44]
棋子數=[白2, 黑2]
棋盤=

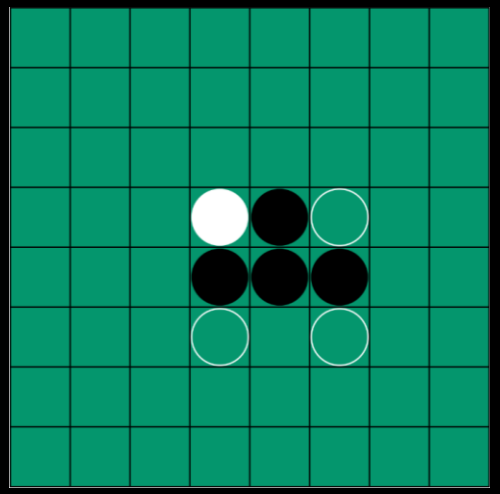


新步=37

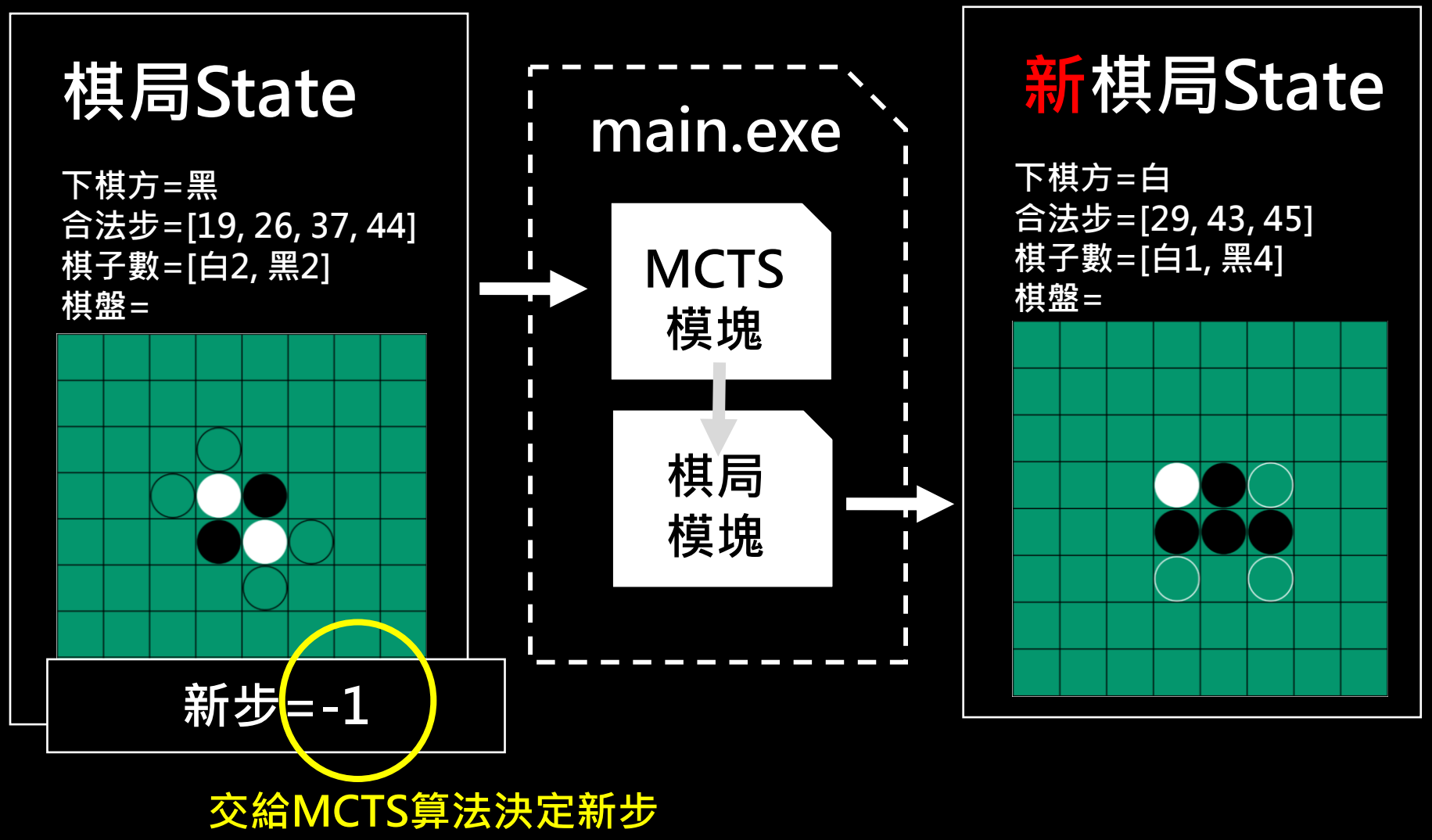
Main
.exe

新棋局State

下棋方=白
合法步=[29, 43, 45]
棋子數=[白1, 黑4]
棋盤=



程式運作流程(棋局更新處理+MCTS算法)



棋局模塊

```
 0  1  2  3  4  5  6  7
 8  9 10 11 12 13 14 15
16 17 18 19 20 21 22 23
24 25 26  ●  ○ 29 30 31
32 33 34  ○  ● 37 38 39
40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63
legalStepList: 20 29 34 43
numberOfPieces: 2 2
color:1
gameOver:0
```

```
class State{
    friend class MCTS; //讓MCTS可以使用State的私人變數

    private:
        int table[8][8];
        vector<int> legalStepList;
        int numberOfPieces[2];
        int color;
        bool gameOver;

    public:
        State();
        void move(int position);
        void updateNumberOfPieces();
        void updateLegalStepList();
        int findLegalStep(int position,int direction);
        bool flip(int position,int direction);
        bool isLegal(int position);
        bool isGameOver();
        void print();

        State(json inputData);
        json toJson();
};
```

棋局變數

棋局更新函式

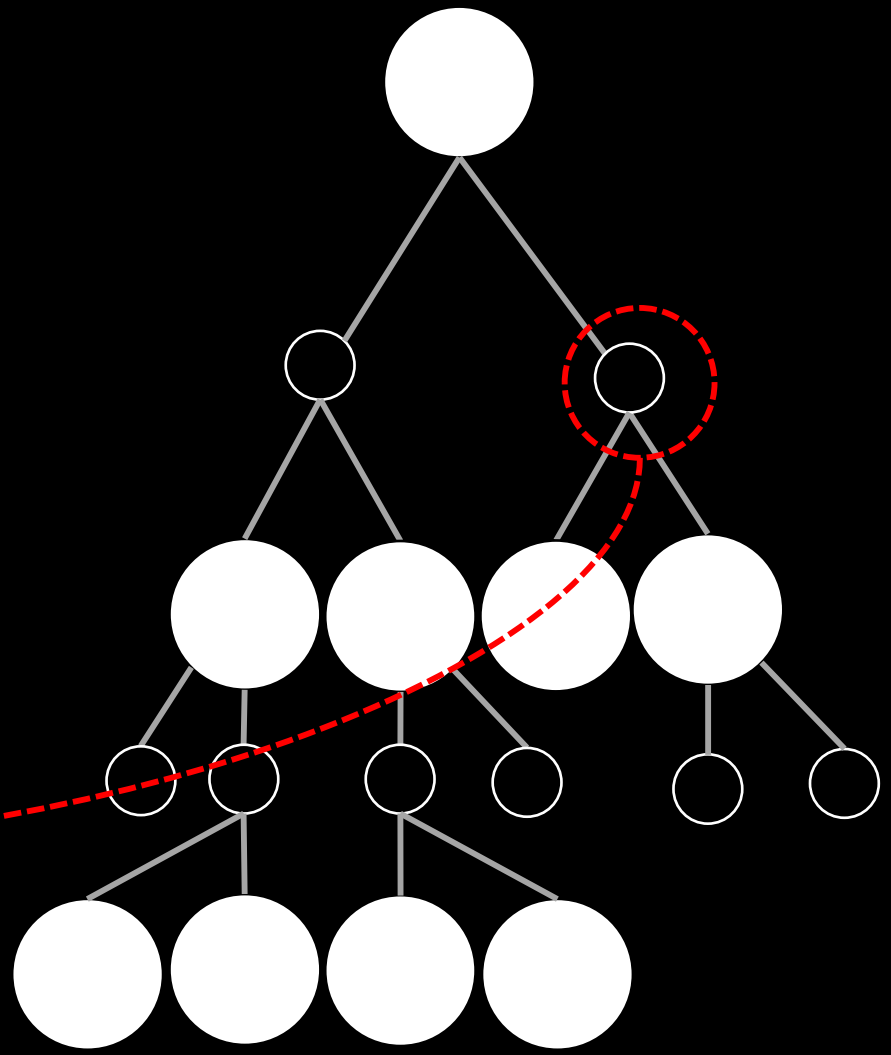
蒙特卡洛樹搜索

Monte Carlo tree search

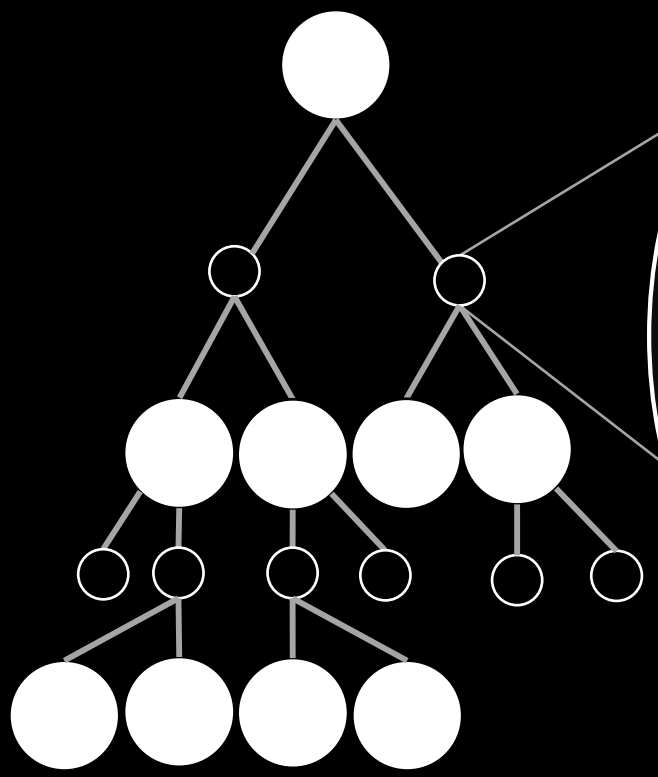
蒙特卡洛樹搜索(MCTS)

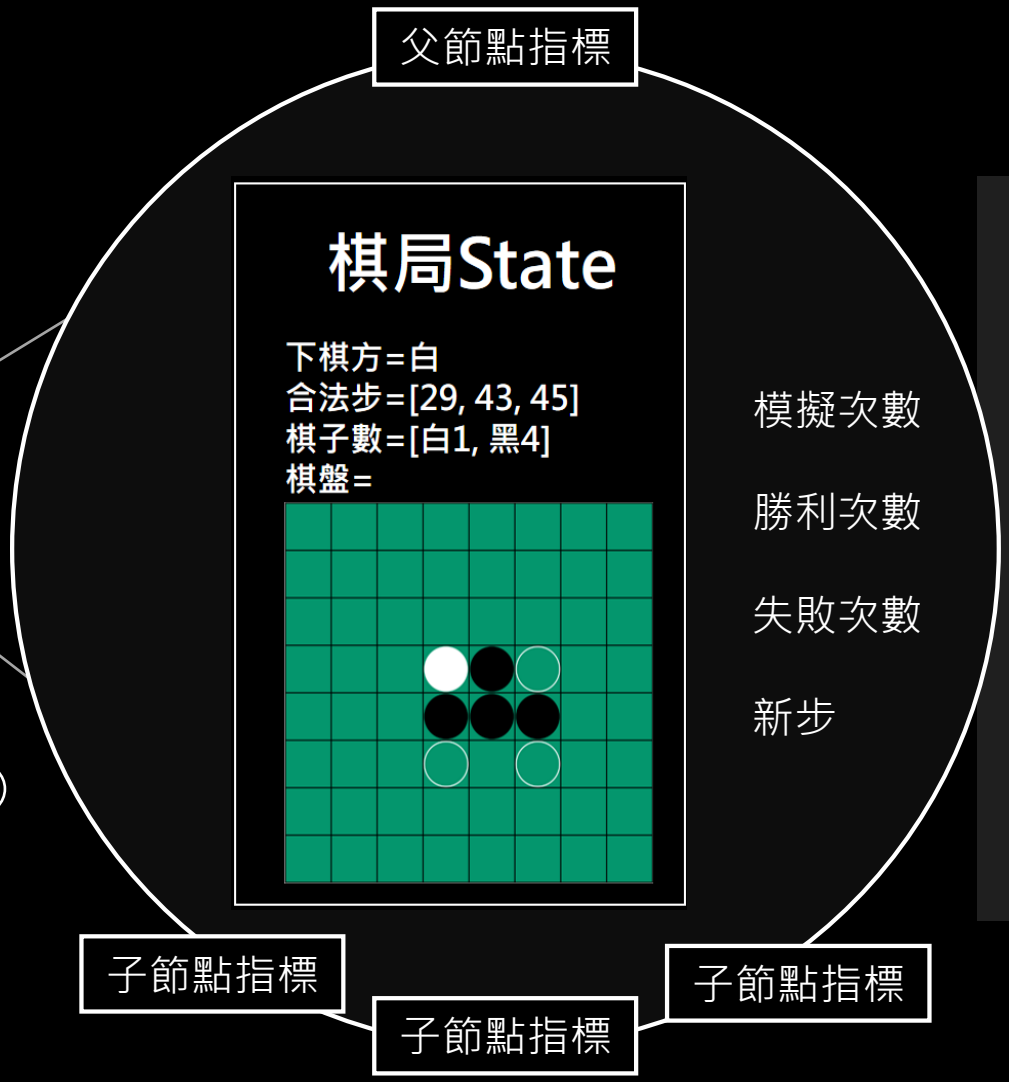
步驟一 根據演算法建構棋局樹

步驟二 選擇最佳落子位置



棋局樹節點



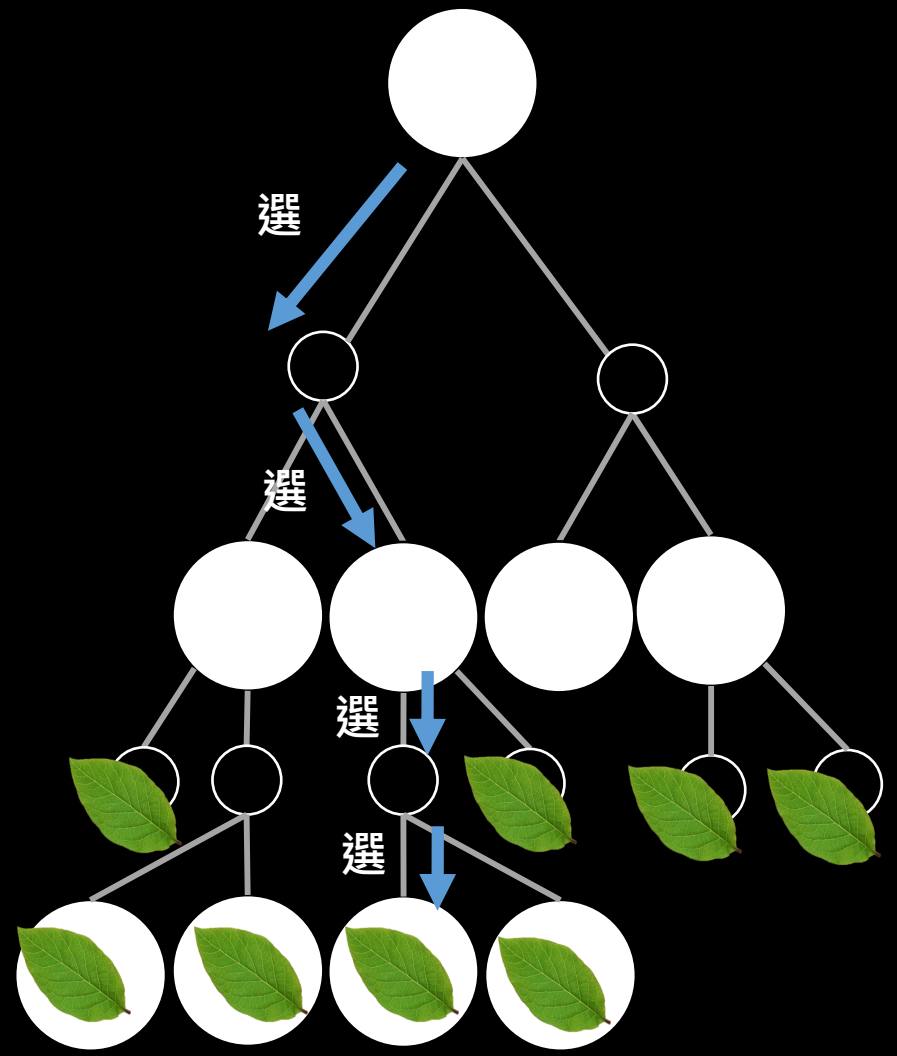



```
class Node{
    public:
        State state;
        int wins;
        int loses;
        int simulations
        int newPosition;
        Node *parent;
        vector<Node*> children;
        Node(State state);
        void print();
};
```

選擇 selection

用UCB公式不斷選擇新節點
直到葉節點

$$UCB(\text{節點}) = \text{勝率} + \sqrt{2}\sqrt{\frac{log\,\text{父節點模擬次數}}{\text{模擬次數}}}$$



若此節點未被模擬過則....

模擬 rollout

隨機落子直到遊戲結束，
並回傳勝利與否，更新節點變數。

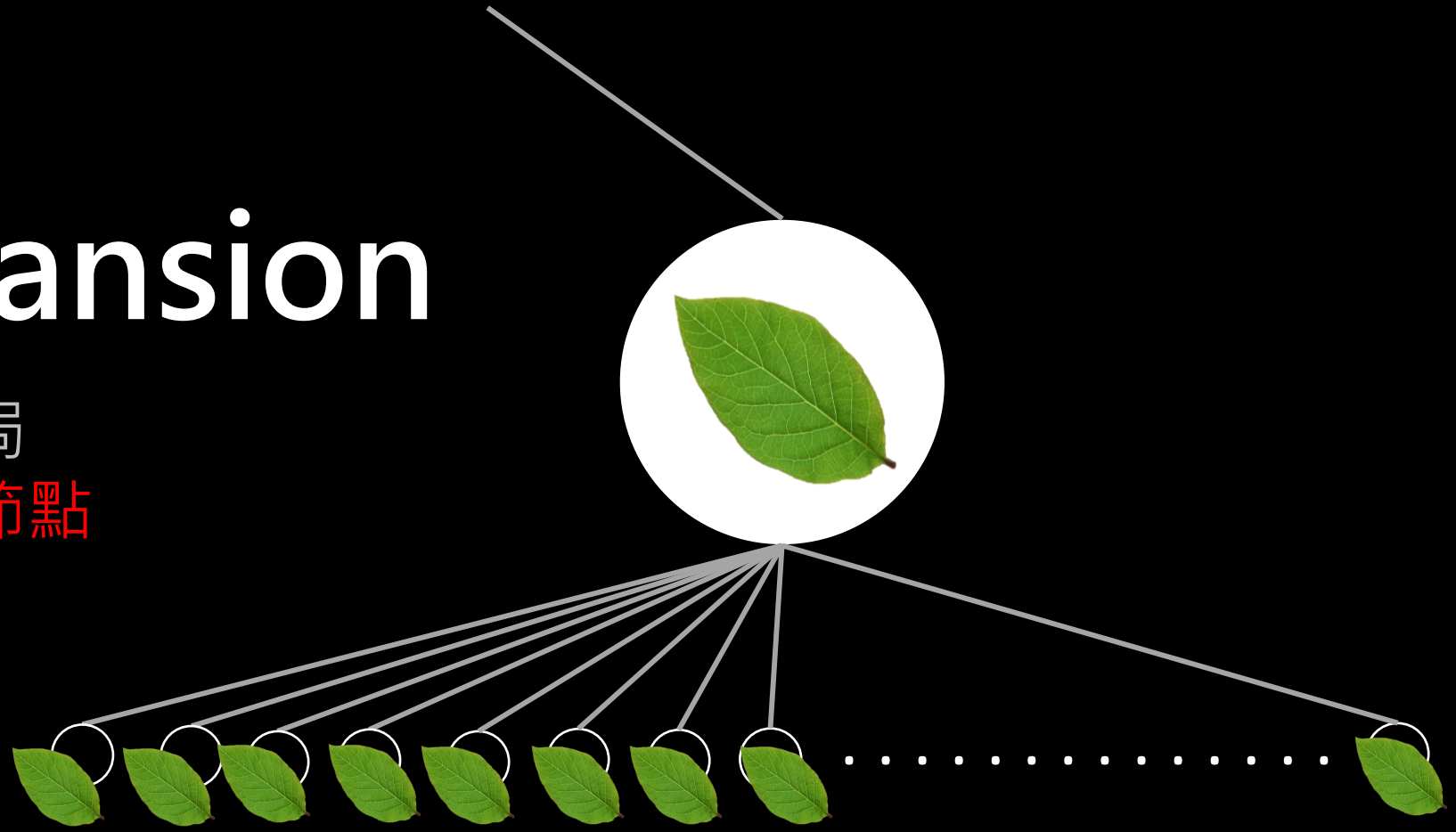
模擬次數、勝利次數
、失敗次數、新步

(重複模擬50次)

若此節點被模擬過則....

擴展 expansion

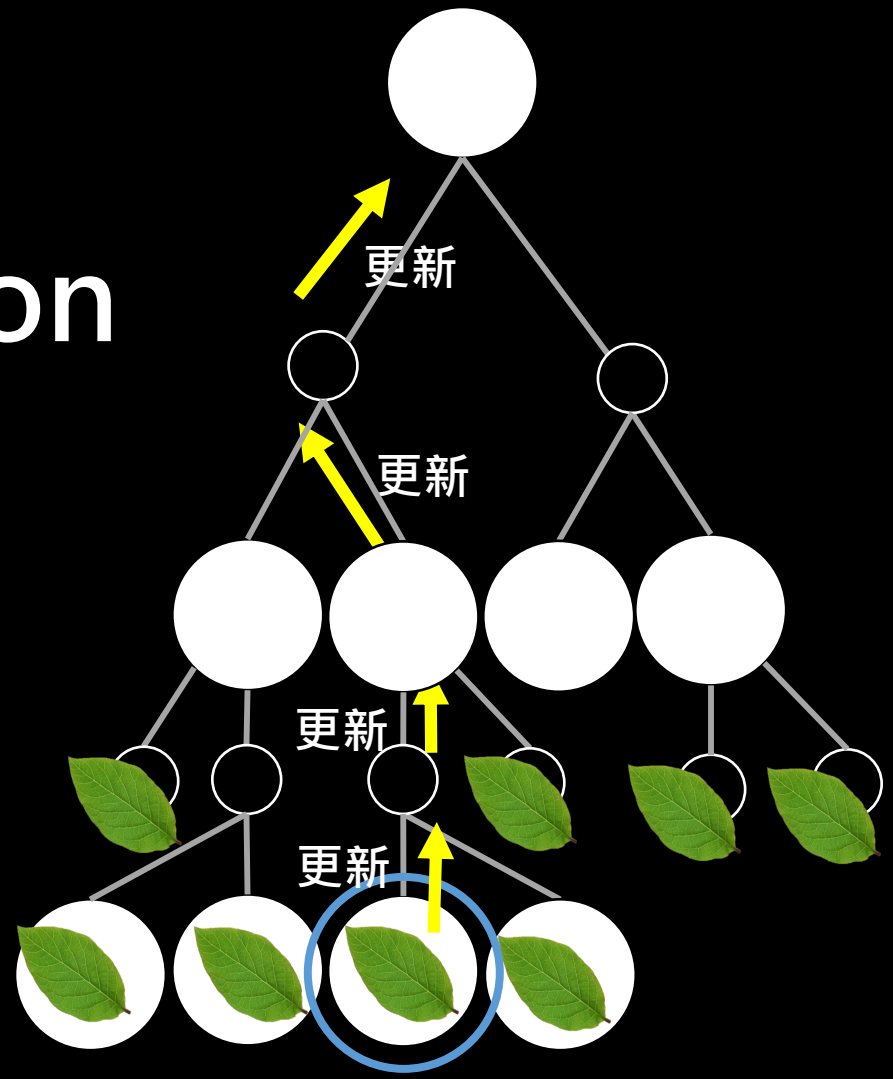
展開所有可能新棋局
此節點則不再是葉節點



更新 back propagation

從葉節點往根節點更新變數

模擬次數、勝利次數
、失敗次數、新步



MCTS模塊


```
class MCTS{
    Node *root;
    public:
        MCTS(State state);
        ~MCTS();
        void deleteNode(Node* node);
        double UCB(Node* node);

        Node* selection(Node* node);
        void expansion(Node* node);
        int rollout(State state);
        void backpropagation(Node *node,int addWins,int addLoses,int addSimulations);

        int predict();

};
```

MCTS核心流程

詳細程式碼

- GitHub
[https://github.com/ShyShyFaceElephant/OthelloWeb]

- 專題網站
[https://othelloweb.up.railway.app/]

參考資料

- Roy Hung (2019). *A Reversi Playing Agent and the Monte Carlo Tree Search Algorithm*

[https://royhung.com/reversi]

- David Foster (2017). *AlphaGo Zero Explained In One Diagram*

[https://medium.com/applied-data-science/alphago-zero-explained-in-one-diagram-365f5abf67e0]