HW2

solving a two player deterministic EWN

EWN (Einstein würfelt nicht!)

All assignments in this course will be related to EWN.

Rule Description

EWN is a **two player** and **stochastic** (雙人隨機) game.

But in hw2, we’re going to solve a EWN variant that is

two player and **deterministic** (雙人無隨機).

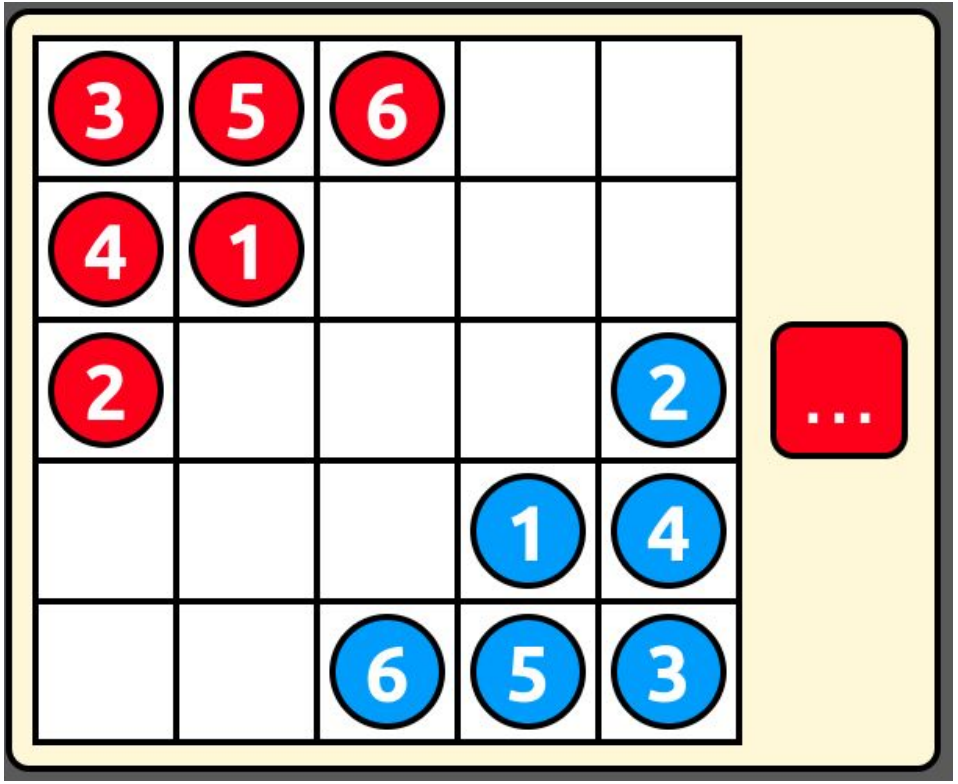
Rules of modified EWN

1. board:

   size: 5x5

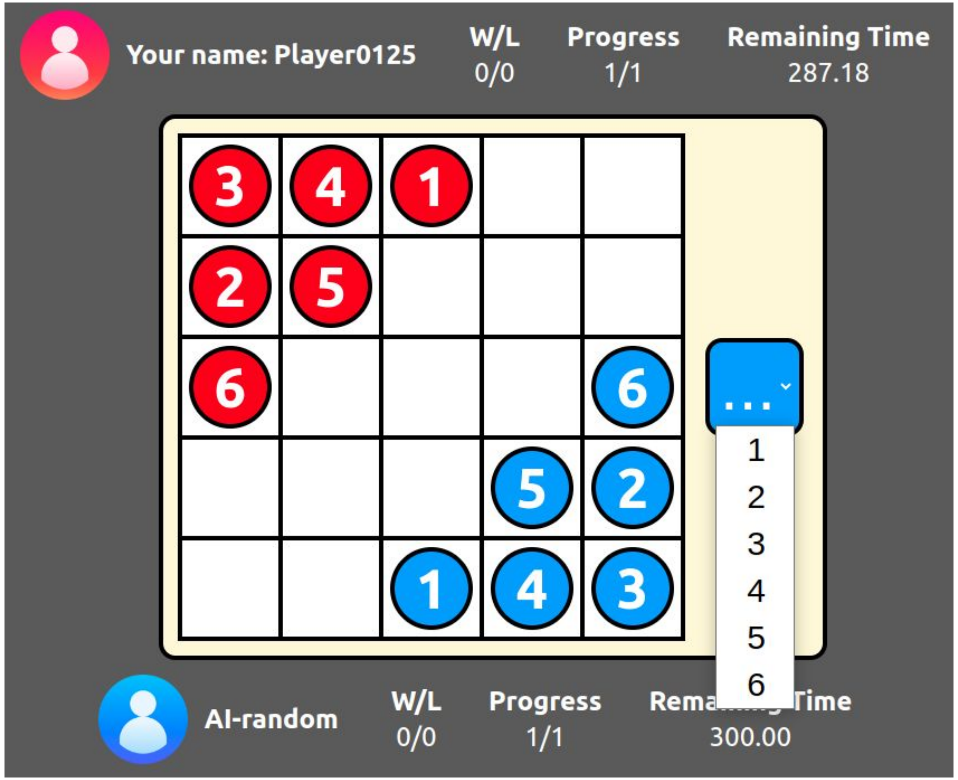
   pieces: 1,2,3,4,5,6

In this EWN variant, dice are not random, you decide the dice value for your opponent.



Rules of modified EWN

The first player can only decide the dice value, because there is no default value for the dice.

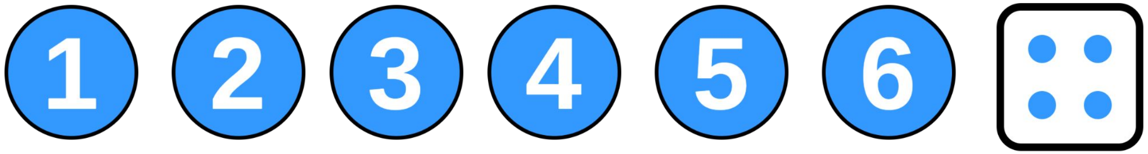


Rules of modified EWN

2. moving procedure

a. determine the moving piece :

if the dice number matches an existing piece, you can only move that matched piece.



In this example, you can only move piece 4.

Rules of modified EWN

2. moving procedure

   b. determine the moving piece :

      if the dice number doesn’t match any existing piece, you can choose to move:

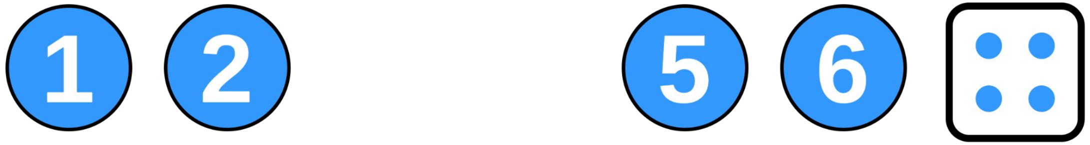
1. The piece that its number is the smallest but bigger than the dice number. (If such piece exists.)
2. The piece that its number is the biggest but smaller than the dice number. (If such piece exists.)

Examples are in the next page.

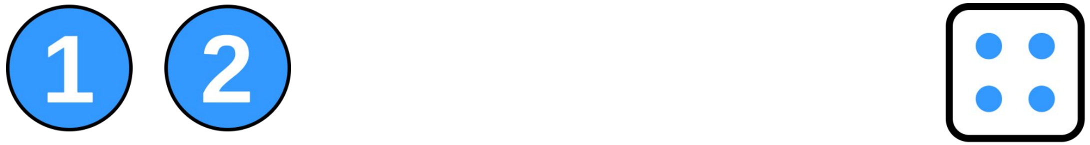
Rules of modified EWN

2. moving procedure

   b. determine the moving piece :



In this example, you can choose to move piece 2 or piece 5.



In this example, you can only move piece 2.

Rules of modified EWN

2. moving procedure

c. piece movement and capture

After choosing the moving piece, you can move it in 3 directions.

For the red side, it is down, right, down-right.

For the blue side, it is up, left, up-left.

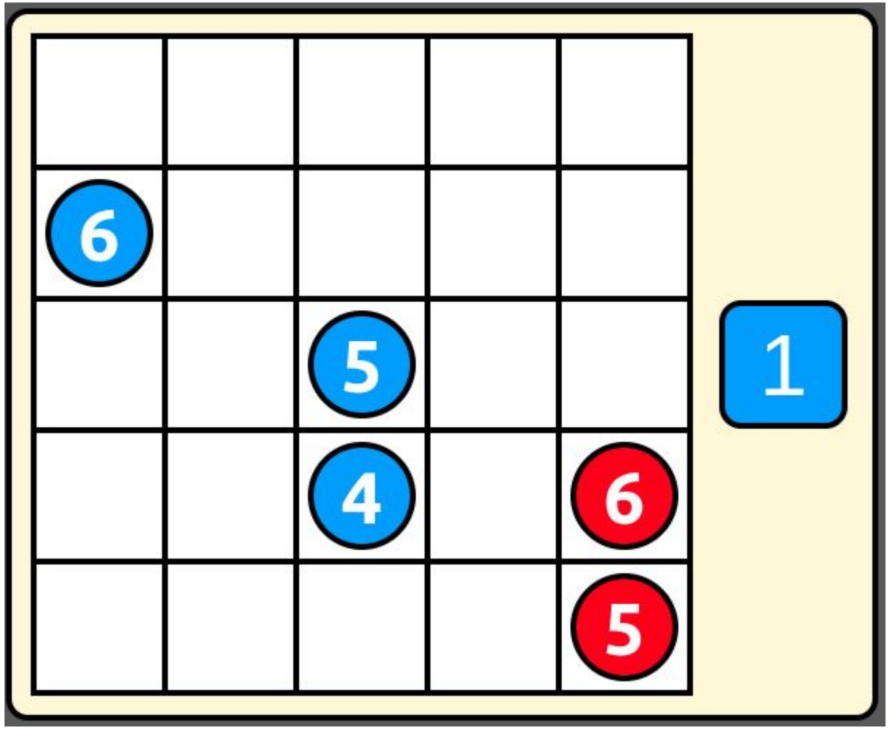
If there is a piece at the destination, the piece at the destination will be removed.

Rules of modified EWN

3. win condition:

Each side has a goal position. For the red side, the goal is the bottom-right corner. For the blue side, the goal is the top-left corner.

For example, the red 5 reaches the bottom-right corner. Red wins the game.

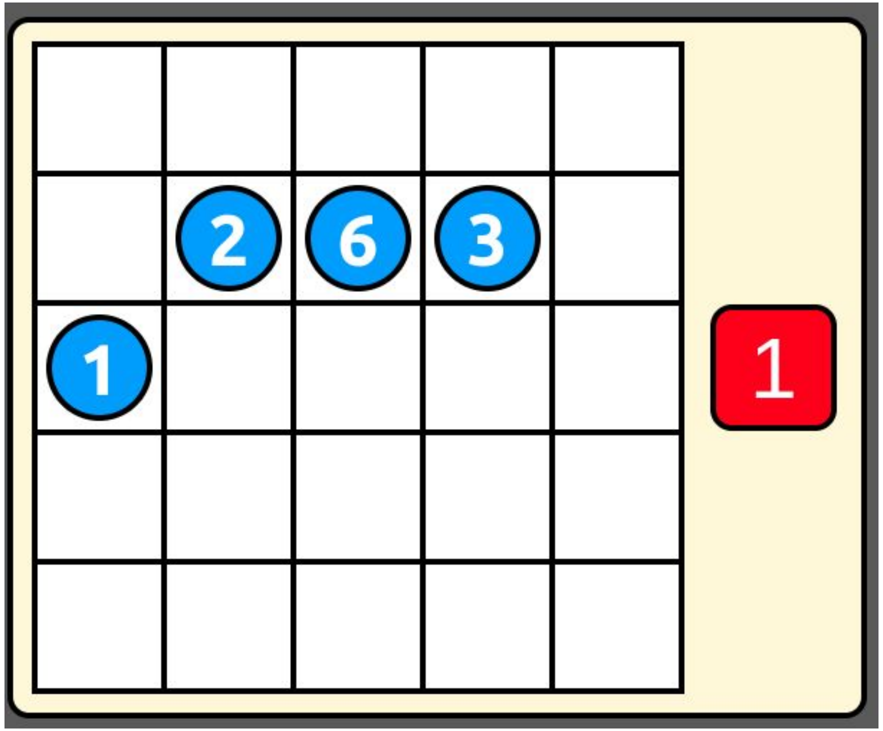


Rules of modified EWN

4. win condition - capture all:

If every piece of the player is captured. The opponent wins.

For example, blue captures every red piece, so blue wins.



Overview of HW2

1. programming 57%:
   a. random baseline: 7%
   b. easy baseline: 30%
   c. medium baseline: 20%

2. code 23%:
   a. MCTS 10%
   b. advance refinements 8%
   c. code readability 5%

3. report 20%:

4. bonus contest 15%:

total grade won't exceed 100%

Programming part

grading criteria

Programming Part - Baselines

For our HW2 and final, we will using this platform written by TA.

https://weilin97462.github.io/EWNPlatform/

You can play the game online, or use the client to play the game by your AI.

The sample code already handles all communication with the client. You can focus on implementing the algorithm.

Programming Part - Baselines

TA is very nice. The sample code is a fully functional MCS-pure program with UCB biased sampling.

You can use C++ in this homework.

The goal of this homework is to **use MCTS** to beat the baselines.

Baselines are hidden in the backend, I will modify the baseline if I think baselines are too hard or too easy.

Programming Part - Baselines

random baseline: **7%**

In our grading program, I will let your program play with random agent for 100 times.

Your score = (min(wins,95)/95) * 7%

Means that if you win 95 times in this 100 times game, you get full score.

Sample code is sufficient to pass this baseline.

Programming Part - Baselines

easy baseline: **30%**

In our grading program, I will let your program play with easy agent for 100 times.

Your score = (min(wins,70)/70) * 30%

Means that if you win 70 times in this 100 times game, you will get full score.

MCTS algorithm is sufficient to pass this baseline.

Programming Part - Baselines

medium baseline: **20%**

In our grading program, I will let your program play with medium agent for 100 times.

Your score = (min(wins,70)/70) * 20%

Means that if you win 70 times in this 100 times game, you will get full score.

MCTS algorithm with refinements is sufficient to pass this baseline.

Programming Part - Submission

```
[any name you like].zip
└── code
    ├── makefile
    └── [other files]
```

Your code must successfully compile an executable named agent after running **make** command.

The submit system will auto rename your zip file, no need to change the name.

submission site:

www.csie.ntu.edu.tw/~tcg/2024/hw2

Programming Part - Grading criteria

1. We will compile your code on csie workstation.

   Make sure your code runs normally on csie workstation.

2. Time limit for a game is **30s**. Recommend using time control introduced in chap 16.
3. **No multithreading or forking.** You will get zero grade in programming if caught.
4. Memory limit: **1GiB**. Exceed the limit will crash your program.

Coding part

grading criteria

Coding Part

To make sure you did implement the MCTS, we will read your code.

1. MCTS 10%

    If you correctly implement MCTS, you will get 10% full marks.

    If your program has bug, we will deduce your grade.

Coding Part

To make sure you did implement the MCTS, we will read your code.

2. advance refinements 8%

    There are some advanced techniques introduced in Chap.10, choose 1 or 2 from Chap.10 to implement to get this part of score.

    We will grade this part according to the quality of your refinements, no full mark is guaranteed like MCTS part.

    Deep learning is disallowed for this part!

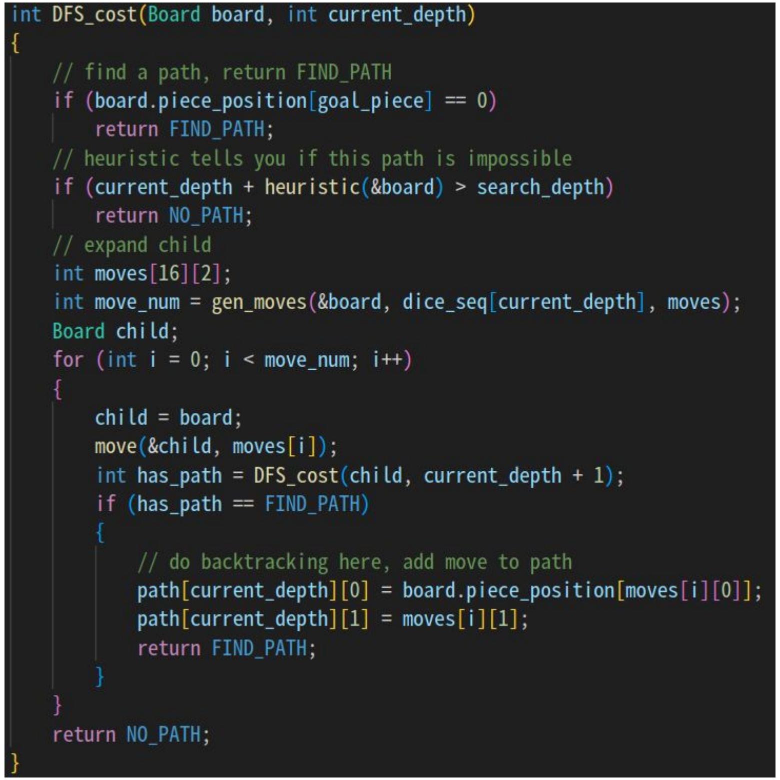
Coding Part

3. readability (5%)

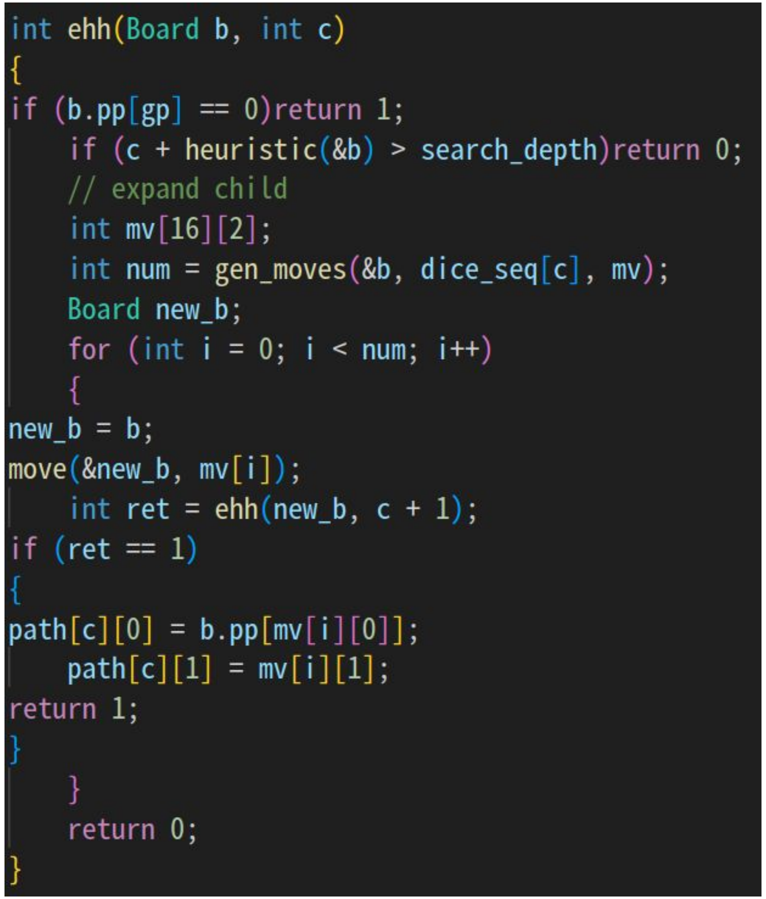
To make sure your code is humanly readable, we encourage you to make your code as readable as possible.

1. don’t name your variable, function name with meaningless names
2. Helpful comments in your code are highly encouraged
3. use #define to define your return value of your function

Coding Part



```
int DFS_cost(Board board, int current_depth)
{
    // find a path, return FIND_PATH
    if (board.piece_position[goal_piece] == 0)
        return FIND_PATH;
    // heuristic tells you if this path is impossible
    if (current_depth + heuristic(&board) > search_depth)
        return NO_PATH;
    // expand child
    int moves[16][2];
    int move_num = gen_moves(&board, dice_seq[current_depth], moves);
    Board child;
    for (int i = 0; i < move_num; i++)
    {
        child = board;
        move(&child, moves[i]);
        int has_path = DFS_cost(child, current_depth + 1);
        if (has_path == FIND_PATH)
        {
            // do backtracking here, add move to path
            path[current_depth][0] = board.piece_position[moves[i][0]];
            path[current_depth][1] = moves[i][1];
            return FIND_PATH;
        }
    }
    return NO_PATH;
}
```

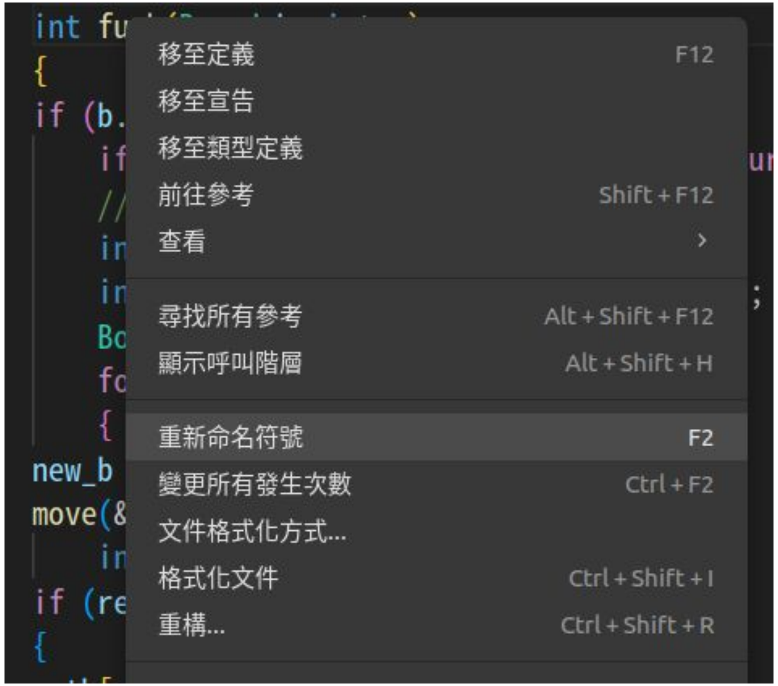


```
int ehh(Board b, int c)
{
if (b.pp[gp] == 0)return 1;
    if (c + heuristic(&b) > search_depth)return 0;
    // expand child
    int mv[16][2];
    int num = gen_moves(&b, dice_seq[c], mv);
    Board new_b;
    for (int i = 0; i < num; i++)
    {
new_b = b;
move(&new_b, mv[i]);
    int ret = ehh(new_b, c + 1);
if (ret == 1)
{
path[c][0] = b.pp[mv[i][0]];
    path[c][1] = mv[i][1];
return 1;
}
    }
    return 0;
}
```

Coding Part

There are some useful tools in vscode to make your code more readable.

1. rename variable
   In vscode, you can rename all of your variable in one click, don’t rename them one by one!

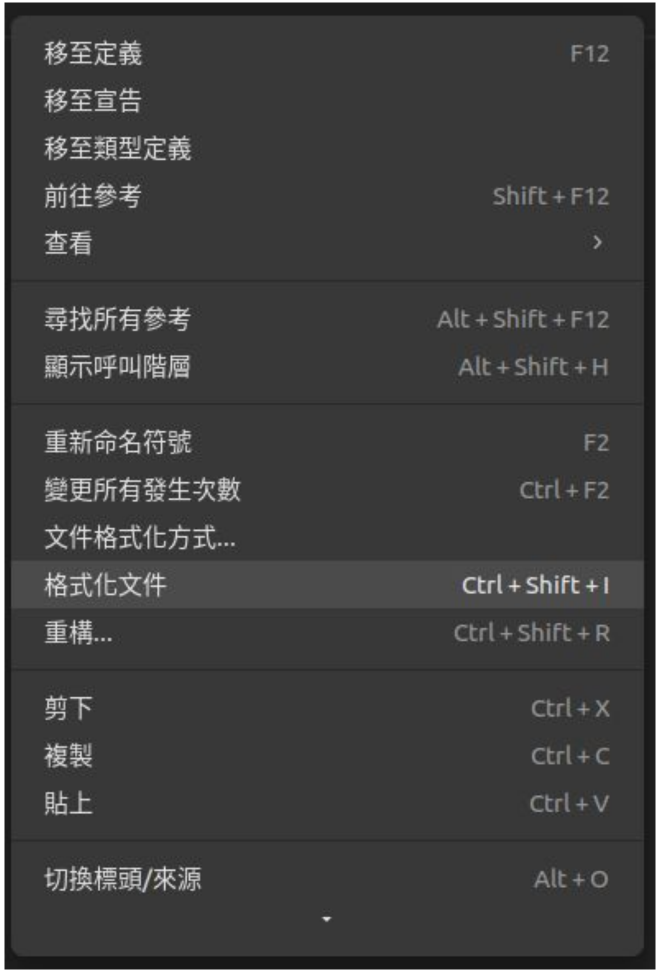


Coding Part

There are some useful tools in vscode to make your code more readable.

2. automatic code formatting:
   In vscode, you can auto indent, auto spacing, auto newline your code in a single click!

   don’t manually indent your code.



Handwritting part

report

Report

Your report should include:

1. Explanation of your code (10%)
2. experiment results (10%)
   (recommend showing the win rate between new and old agent)

Because **report has different deadline**, it has its own submission site:

www.csie.ntu.edu.tw/~tcg/2024/hw2_report

Only pdf format is accepted.

Bonus contest

fight for the glory!!!

Bonus Contest

In HW2, you can only use MCTS in your programming part. I think it is not so fun.

We have a great solution this year! There is a contest available for TCG enthusiasts!

In this contest, **there are no limitations for your program.** You can use any techniques you like such as multithreading, minimax, GPU-acceleration, deep-learning, large databases, ... etc.

Moreover, you can run your program on **any machine** that have access to internet. The only limitation is your creativity!

Hope this contest will be fun!

Bonus Contest

grading:

attendance: 5%

score: 10%

((your_score - min_score)/(max_score - min_score))*10%

score means how many wins you have, win more, earn more!!!

The detail of the contest will be introduced in later days.