

E03 Othello Game (min-max, $\alpha - \beta$ pruning and evaluation function)

19335016 HaoRan Chen

October 11, 2021

Contents

1	Othello	2
2	Tasks	2
3	Codes	3
4	Results	5

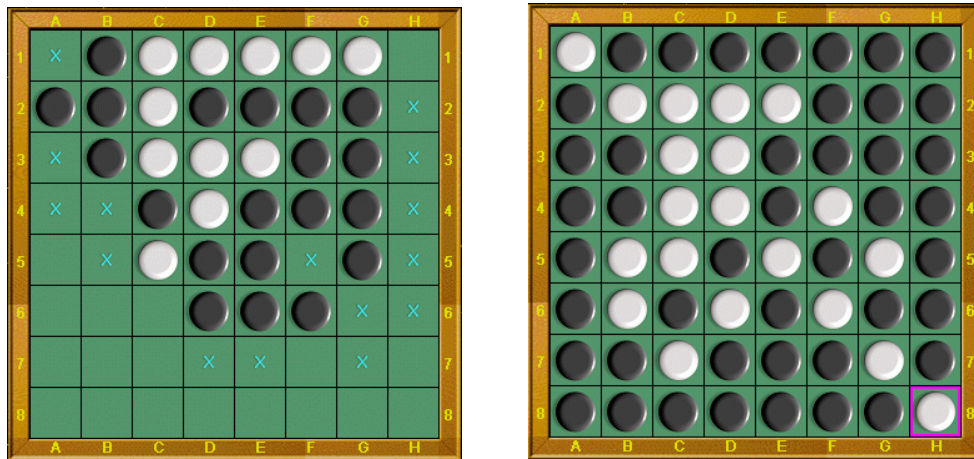


Figure 1: Othello Game

1 Othello

Othello (or Reversi) is a strategy board game for two players, played on an 8×8 uncheckered board. There are sixty-four identical game pieces called disks (often spelled "discs"), which are light on one side and dark on the other. Please see figure 1.

Players take turns placing disks on the board with their assigned color facing up. During a play, any disks of the opponent's color that are in a straight line and bounded by the disk just placed and another disk of the current player's color are turned over to the current player's color.

The object of the game is to have the majority of disks turned to display your color when the last playable empty square is filled.

You can refer to http://www.tothello.com/html/guideline_of_reversed_othello.html for more information of guideline, meanwhile, you can download the software to have a try from <http://www.tothello.com/html/download.html>. The game installer `tothello_trial_setup.exe` can also be found in the current folder.

2 Tasks

1. In order to reduce the complexity of the game, we think the board is 6×6 .
2. There are several evaluation functions that involve many aspects, you can turn to http://blog.sina.com.cn/s/blog_53ebdba00100cpy2.html for help. In order to reduce the difficulty of the task, I have given you some hints of evaluation function in the file `Heuristic Function for Reversi (Othello).cpp`.

3. Please choose an appropriate evaluation function and use min-max and $\alpha - \beta$ pruning to implement the Othello game. The framework file you can refer to is `Othello.cpp`. Of course, I wish your program can beat the computer.
4. Write the related codes and take a screenshot of the running results in the file named `E06_YourNumber.pdf`, and send it to `ai_201901@formail.comhh`

3 Codes

```

int Othello::Judge(Othello *board, enum Option player){
    int value=0;
    Stable(board);

    int x = 0, y = 0, i = 0, j = 0, k = 0,
    my_tiles = 0, opp_tiles = 0,
    my_front_tiles = 0, opp_front_tiles = 0;

    double p = 0, m = 0, f = 0, d = 0;

    int X1[] = {-1, -1, 0, 1, 1, 1, 0, -1};
    int Y1[] = {0, 1, 1, 1, 0, -1, -1, -1};
    int V[6][6] = {
        {15, 2, 6, 6, 3, 15},
        {2, -3, 0, 0, -3, 2},
        {6, 0, -3, -3, 0, 6},
        {6, 0, -3, -3, 0, 6},
        {2, -3, 0, 0, -3, 2},
        {15, 2, 6, 6, 3, 15},
    };

    for(i = 0; i < 6; i++)
        for(j = 0; j < 6; j++) {
            if(board->cell[i][j].color == player) {
                d += V[i][j];
            }
        }
}

```

```

        my_tiles++;
    }
    else if(board->cell[i][j].color == (enum Option) - player){
        d -= V[i][j];
        opp_tiles++;
    }
    if(board->cell[i][j].color != SPACE)    {
        for(k=0; k<8; k++)    {
            x = i + X1[k]; y = j + Y1[k];
            if(x >= 0 && x < 6 && y >= 0 && y < 6 &&
            board->cell[x][y].color == SPACE)    {
                if(board->cell[i][j].color == player)    my_front_tiles++;
                else opp_front_tiles++;
                break;
            }
        }
    }
}

if(my_tiles > opp_tiles)
    p = (100.0 * my_tiles) / (my_tiles + opp_tiles);
else if(my_tiles < opp_tiles)
    p = -(100.0 * opp_tiles) / (my_tiles + opp_tiles);
else p = 0;

if(my_front_tiles > opp_front_tiles)
    f = -(100.0 * my_front_tiles) / (my_front_tiles + opp_front_tiles);
else if(my_front_tiles < opp_front_tiles)
    f = (100.0 * opp_front_tiles) / (my_front_tiles + opp_front_tiles);
else f = 0;

my_tiles = Rule(board, player);
opp_tiles = Rule(board, (enum Option) - player);
if(my_tiles > opp_tiles)

```

```

        m = (1.0 * 75 * my_tiles)/(my_tiles + opp_tiles); //75=100*6/8
    else if(my_tiles < opp_tiles)
        m = -(1.0 * 75 * opp_tiles)/(my_tiles + opp_tiles);
    else m = 0;

    value = (10 * p) + (78.922 * m) + (74.396 * f) + (10 * d);
    return (int)value;
}

```

4 Results

[illegible]

Among them, white chess is the transformed AI and black chess is the original AI.