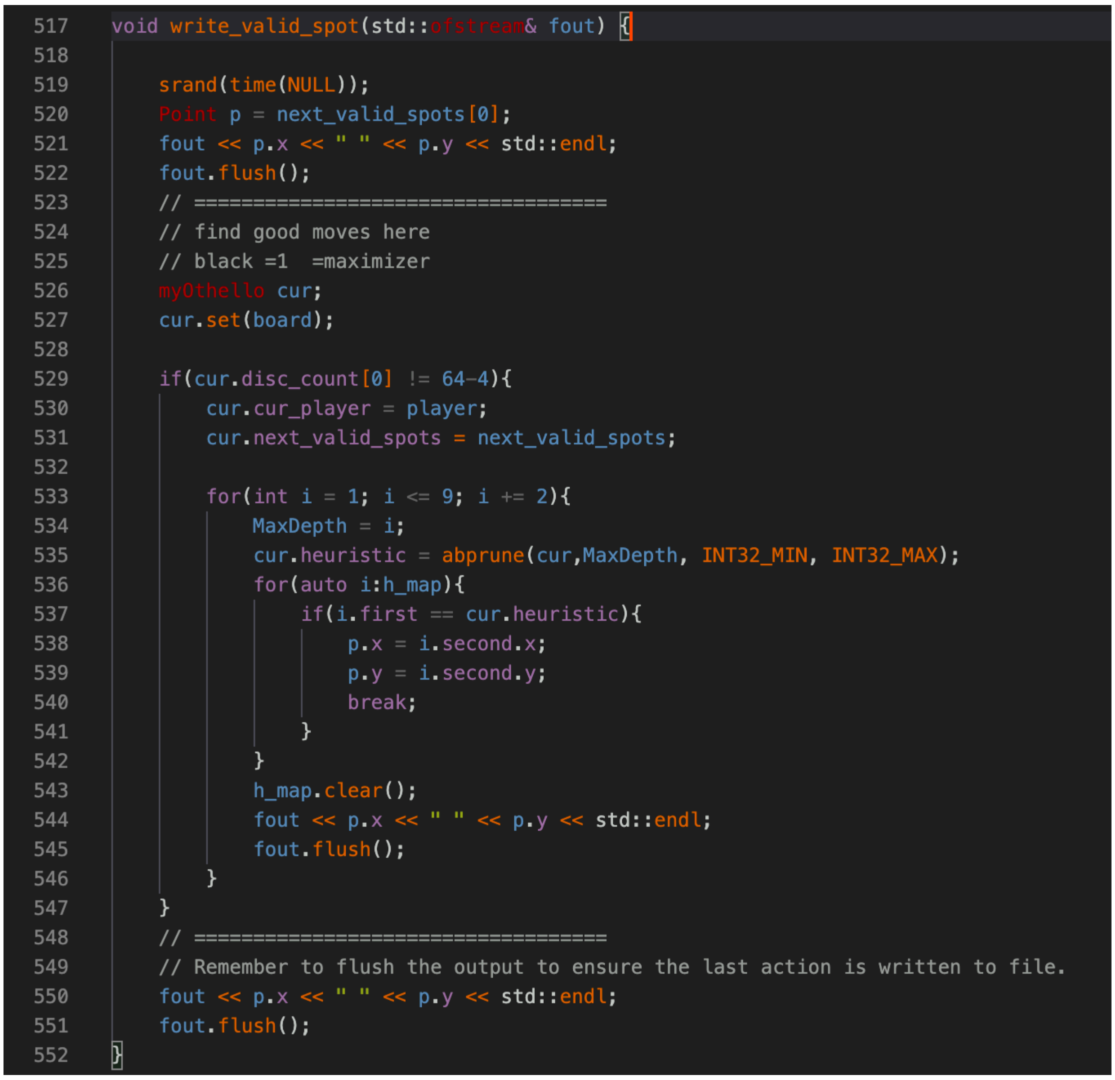
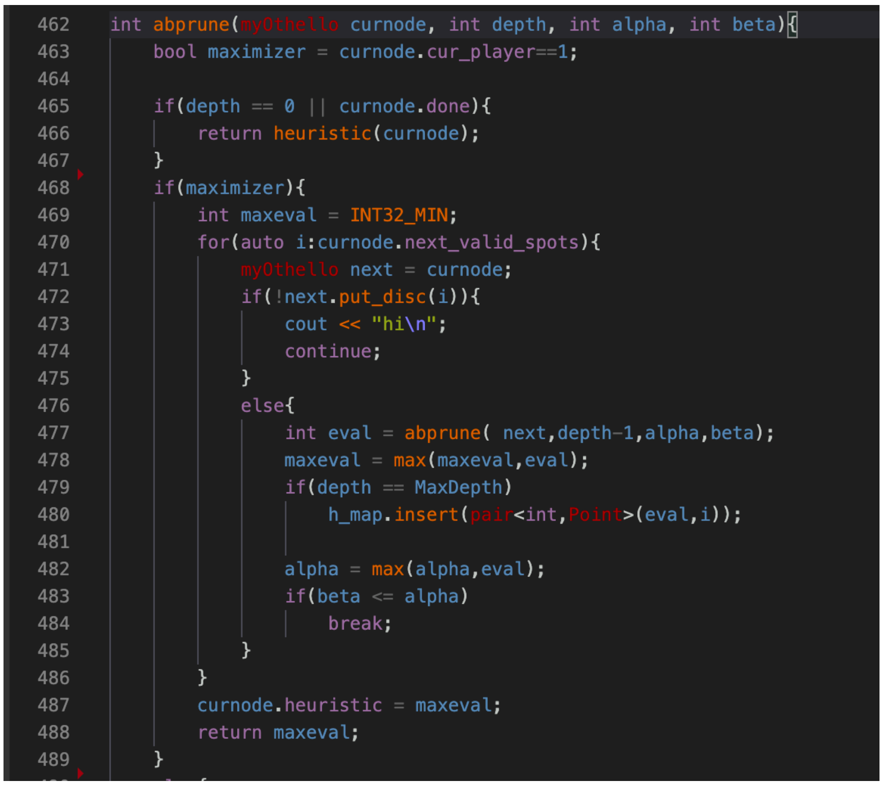
mini project3 report

1. Calling tree search

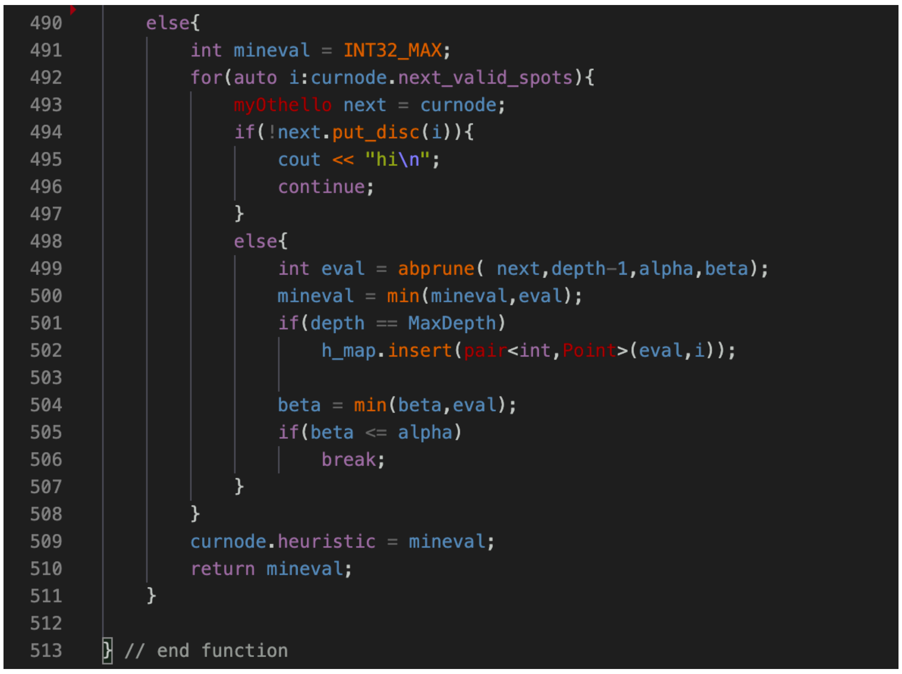


```
void write_valid_spot(std::ofstream& fout) {

    srand(time(NULL));
    Point p = next_valid_spots[0];
    fout << p.x << " " << p.y << std::endl;
    fout.flush();
    // ===================================
    // find good moves here
    // black =1  =maximizer
    myOthello cur;
    cur.set(board);

    if(cur.disc_count[0] != 64-4){
        cur.cur_player = player;
        cur.next_valid_spots = next_valid_spots;

        for(int i = 1; i <= 9; i += 2){
            MaxDepth = i;
            cur.heuristic = abprune(cur,MaxDepth, INT32_MIN, INT32_MAX);
            for(auto i:h_map){
                if(i.first == cur.heuristic){
                    p.x = i.second.x;
                    p.y = i.second.y;
                    break;
                }
            }
            h_map.clear();
            fout << p.x << " " << p.y << std::endl;
            fout.flush();
        }
    }
    // ===================================
    // Remember to flush the output to ensure the last action is written to file.
    fout << p.x << " " << p.y << std::endl;
    fout.flush();
}
```

2. Tree search & abprune

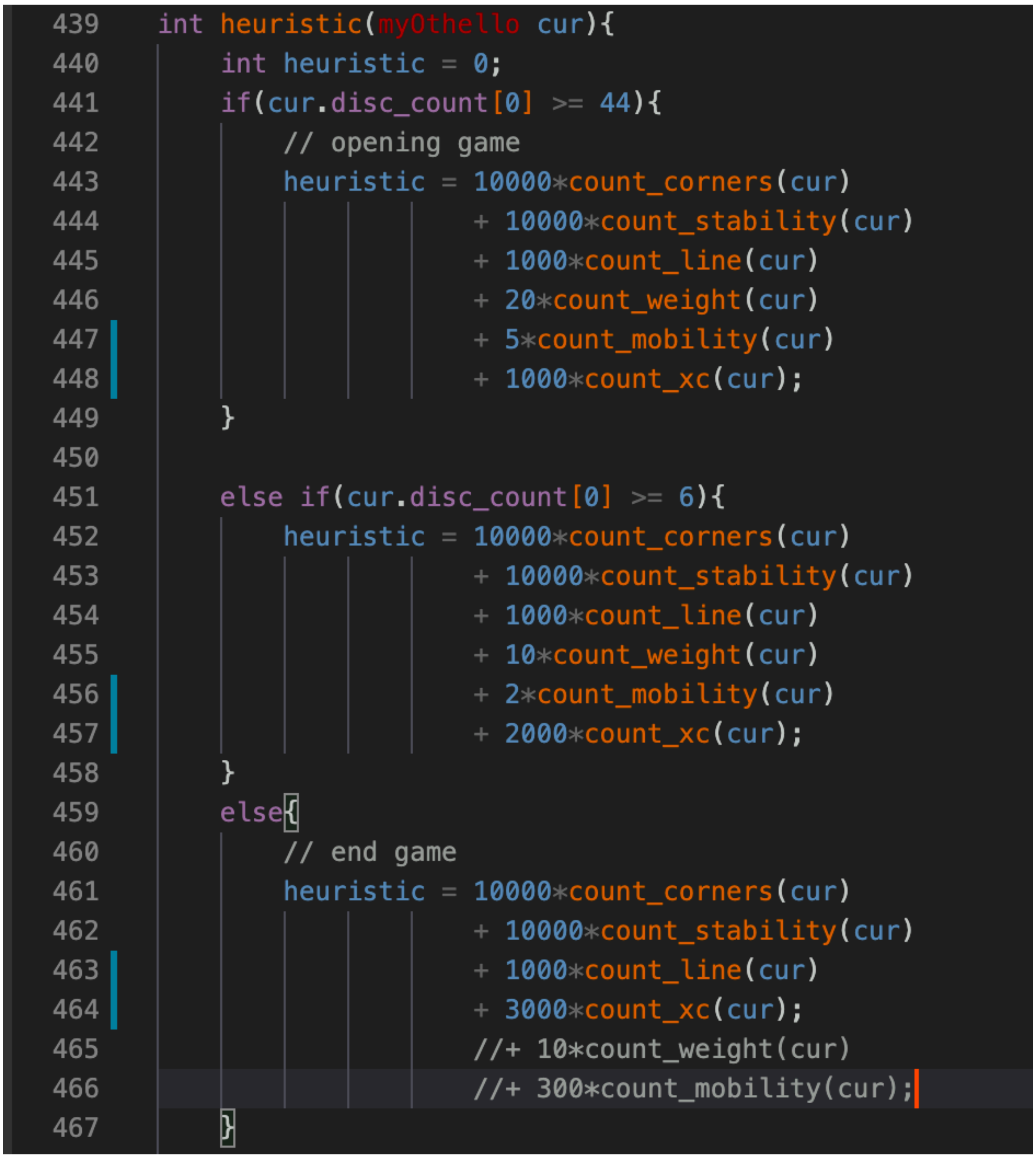


```cpp
int abprune(myOthello curnode, int depth, int alpha, int beta){
    bool maximizer = curnode.cur_player==1;

    if(depth == 0 || curnode.done){
        return heuristic(curnode);
    }
    if(maximizer){
        int maxeval = INT32_MIN;
        for(auto i:curnode.next_valid_spots){
            myOthello next = curnode;
            if(!next.put_disc(i)){
                cout << "hi\n";
                continue;
            }
            else{
                int eval = abprune( next,depth-1,alpha,beta);
                maxeval = max(maxeval,eval);
                if(depth == MaxDepth)
                    h_map.insert(pair<int,Point>(eval,i));

                alpha = max(alpha,eval);
                if(beta <= alpha)
                    break;
            }
        }
        curnode.heuristic = maxeval;
        return maxeval;
    }
```

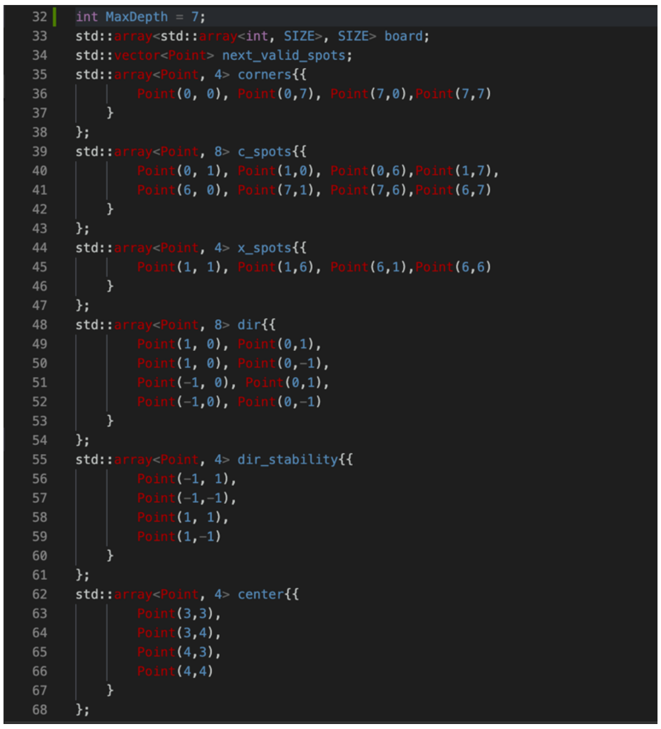


```cpp
    else{
        int mineval = INT32_MAX;
        for(auto i:curnode.next_valid_spots){
            myOthello next = curnode;
            if(!next.put_disc(i)){
                cout << "hi\n";
                continue;
            }
            else{
                int eval = abprune( next,depth-1,alpha,beta);
                mineval = min(mineval,eval);
                if(depth == MaxDepth)
                    h_map.insert(pair<int,Point>(eval,i));

                beta = min(beta,eval);
                if(beta <= alpha)
                    break;
            }
        }
        curnode.heuristic = mineval;
        return mineval;
    }

} // end function
```

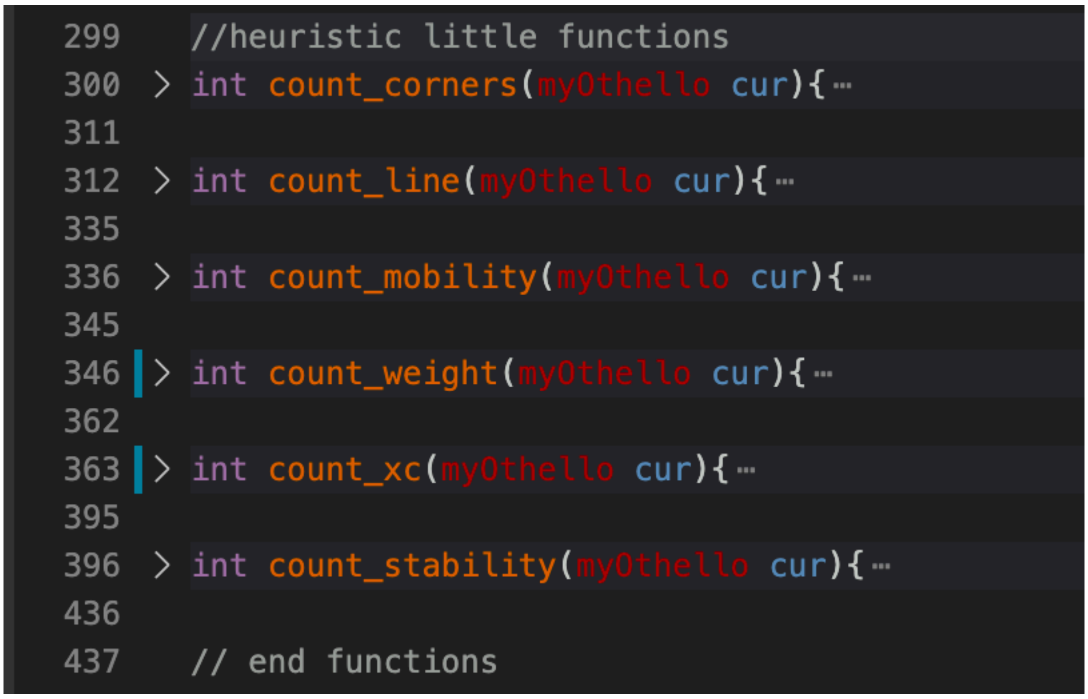
3. State value function design



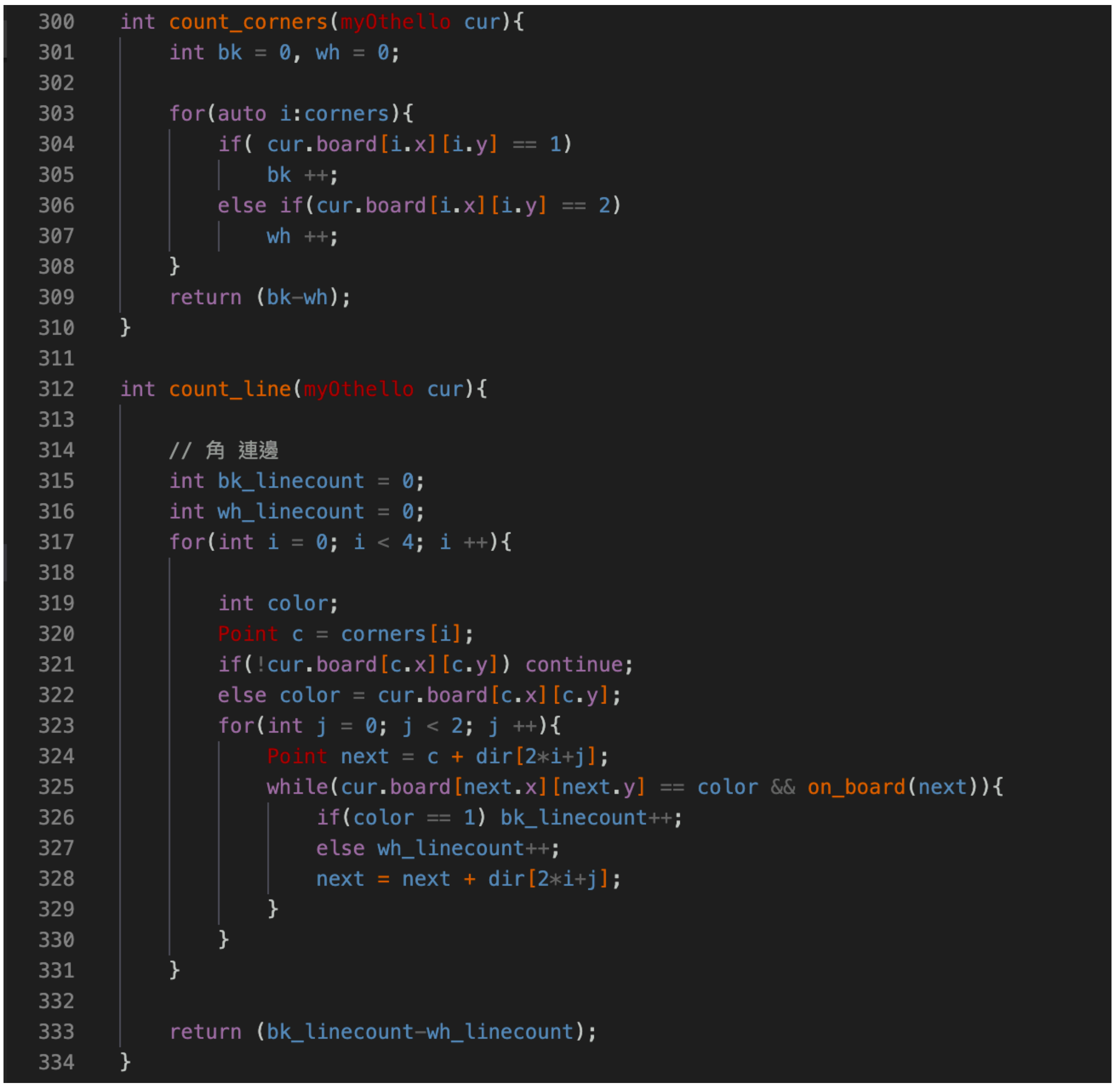
```
int heuristic(myOthello cur){
    int heuristic = 0;
    if(cur.disc_count[0] >= 44){
        // opening game
        heuristic = 10000*count_corners(cur)
                    + 10000*count_stability(cur)
                    + 1000*count_line(cur)
                    + 20*count_weight(cur)
                    + 5*count_mobility(cur)
                    + 1000*count_xc(cur);
    }

    else if(cur.disc_count[0] >= 6){
        heuristic = 10000*count_corners(cur)
                    + 10000*count_stability(cur)
                    + 1000*count_line(cur)
                    + 10*count_weight(cur)
                    + 2*count_mobility(cur)
                    + 2000*count_xc(cur);
    }
    else{
        // end game
        heuristic = 10000*count_corners(cur)
                    + 10000*count_stability(cur)
                    + 1000*count_line(cur)
                    + 3000*count_xc(cur);
                    //+ 10*count_weight(cur)
                    //+ 300*count_mobility(cur);
    }
```



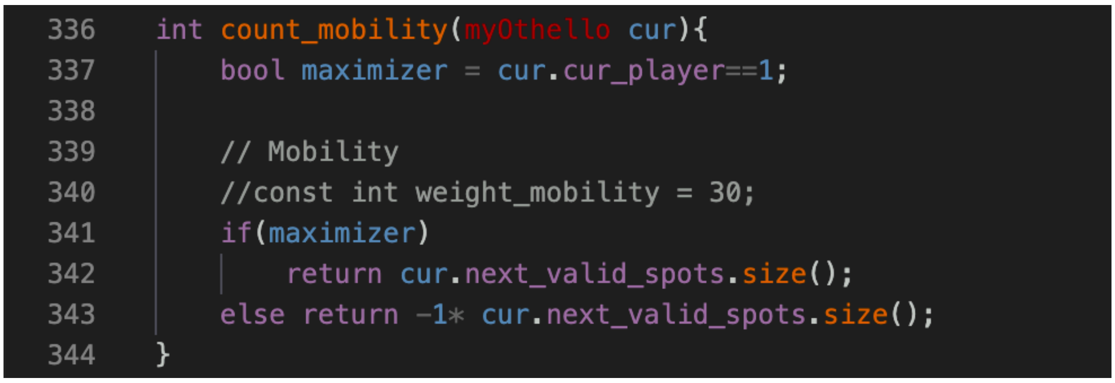
```cpp
int MaxDepth = 7;
std::array<std::array<int, SIZE>, SIZE> board;
std::vector<Point> next_valid_spots;
std::array<Point, 4> corners{{
        Point(0, 0), Point(0,7), Point(7,0),Point(7,7)
    }
};
std::array<Point, 8> c_spots{{
        Point(0, 1), Point(1,0), Point(0,6),Point(1,7),
        Point(6, 0), Point(7,1), Point(7,6),Point(6,7)
    }
};
std::array<Point, 4> x_spots{{
        Point(1, 1), Point(1,6), Point(6,1),Point(6,6)
    }
};
std::array<Point, 8> dir{{
        Point(1, 0), Point(0,1),
        Point(1, 0), Point(0,-1),
        Point(-1, 0), Point(0,1),
        Point(-1,0), Point(0,-1)
    }
};
std::array<Point, 4> dir_stability{{
        Point(-1, 1),
        Point(-1,-1),
        Point(1, 1),
        Point(1,-1)
    }
};
std::array<Point, 4> center{{
        Point(3,3),
        Point(3,4),
        Point(4,3),
        Point(4,4)
    }
};
```



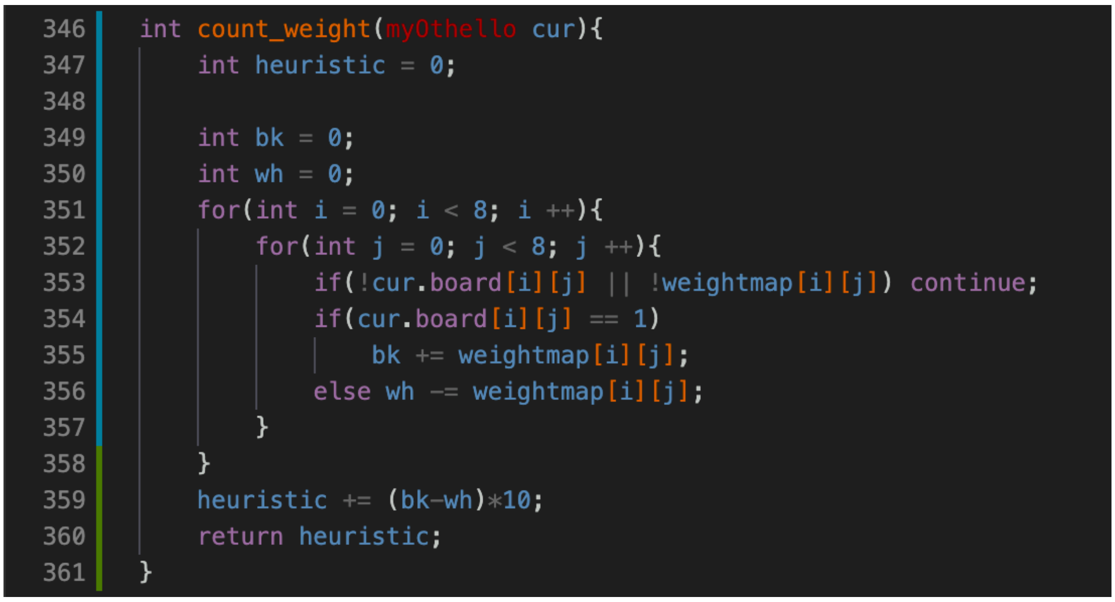
```cpp
//heuristic little functions
int count_corners(myOthello cur){⋯

int count_line(myOthello cur){⋯

int count_mobility(myOthello cur){⋯

int count_weight(myOthello cur){⋯

int count_xc(myOthello cur){⋯

int count_stability(myOthello cur){⋯

// end functions
```



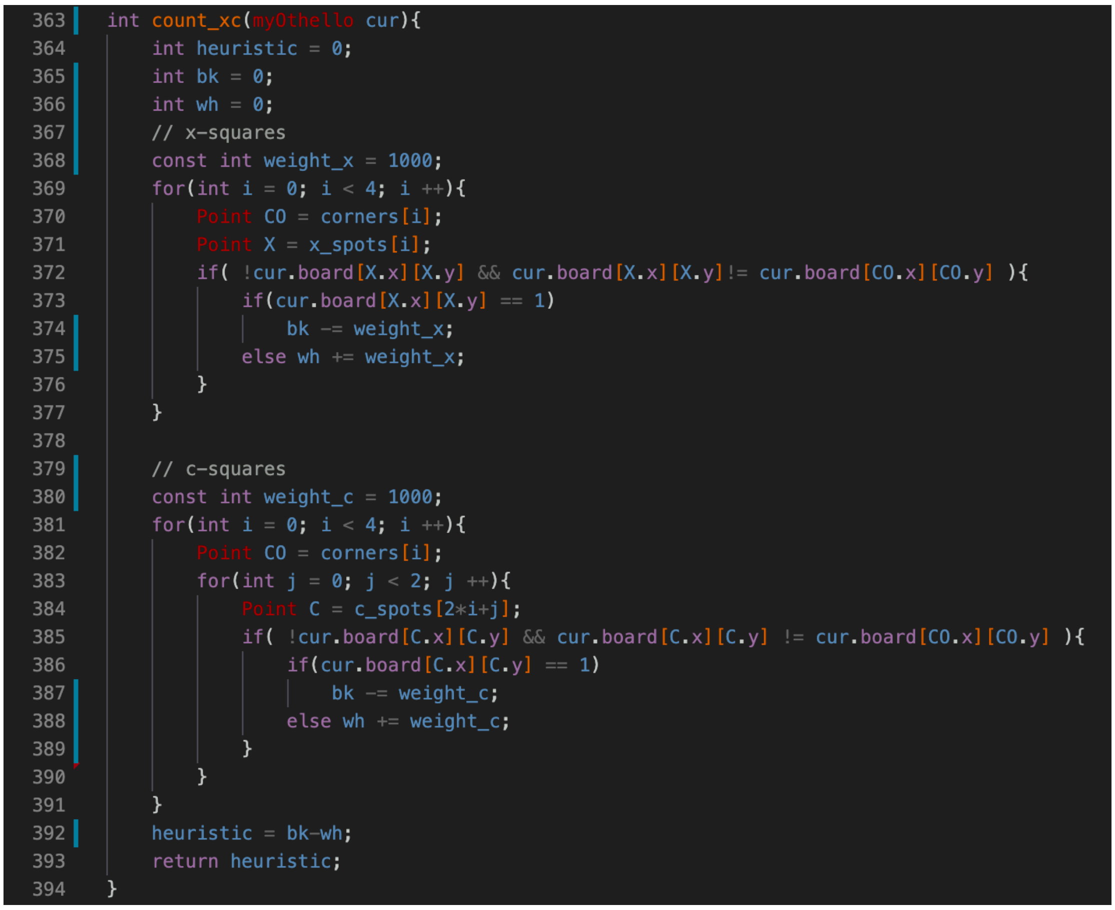
```cpp
int count_corners(myOthello cur){
    int bk = 0, wh = 0;

    for(auto i:corners){
        if( cur.board[i.x][i.y] == 1)
            bk ++;
        else if(cur.board[i.x][i.y] == 2)
            wh ++;
    }
    return (bk-wh);
}

int count_line(myOthello cur){

    // 角 連邊
    int bk_linecount = 0;
    int wh_linecount = 0;
    for(int i = 0; i < 4; i ++){

        int color;
        Point c = corners[i];
        if(!cur.board[c.x][c.y]) continue;
        else color = cur.board[c.x][c.y];
        for(int j = 0; j < 2; j ++){
            Point next = c + dir[2*i+j];
            while(cur.board[next.x][next.y] == color && on_board(next)){
                if(color == 1) bk_linecount++;
                else wh_linecount++;
                next = next + dir[2*i+j];
            }
        }
    }

    return (bk_linecount-wh_linecount);
}
```



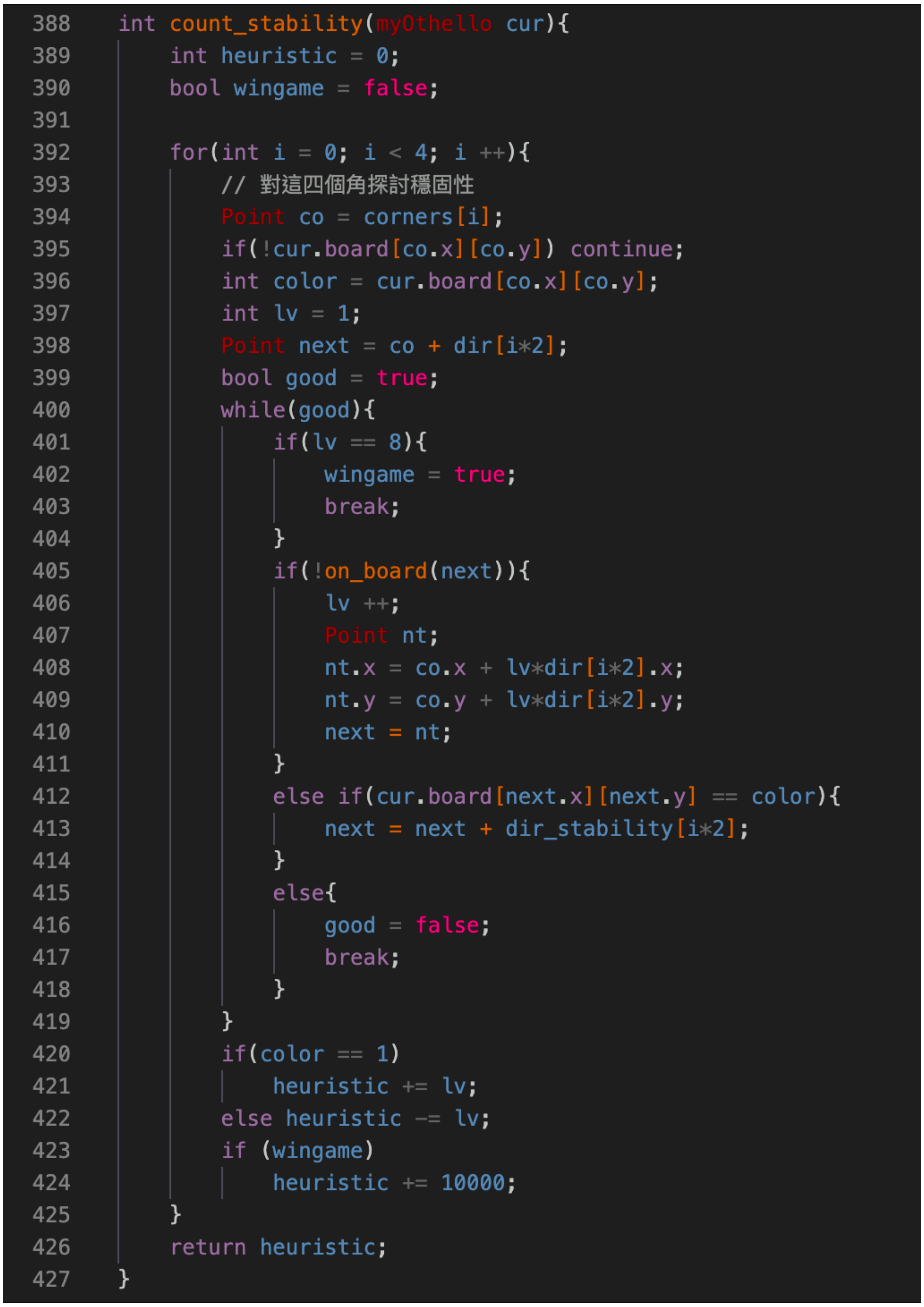
```cpp
int count_mobility(myOthello cur){
    bool maximizer = cur.cur_player==1;

    // Mobility
    //const int weight_mobility = 30;
    if(maximizer)
        return cur.next_valid_spots.size();
    else return -1* cur.next_valid_spots.size();
}
```



```cpp
int count_weight(myOthello cur){
    int heuristic = 0;

    int bk = 0;
    int wh = 0;
    for(int i = 0; i < 8; i ++){
        for(int j = 0; j < 8; j ++){
            if(!cur.board[i][j] || !weightmap[i][j]) continue;
            if(cur.board[i][j] == 1)
                bk += weightmap[i][j];
            else wh -= weightmap[i][j];
        }
    }
    heuristic += (bk-wh)*10;
    return heuristic;
}
```

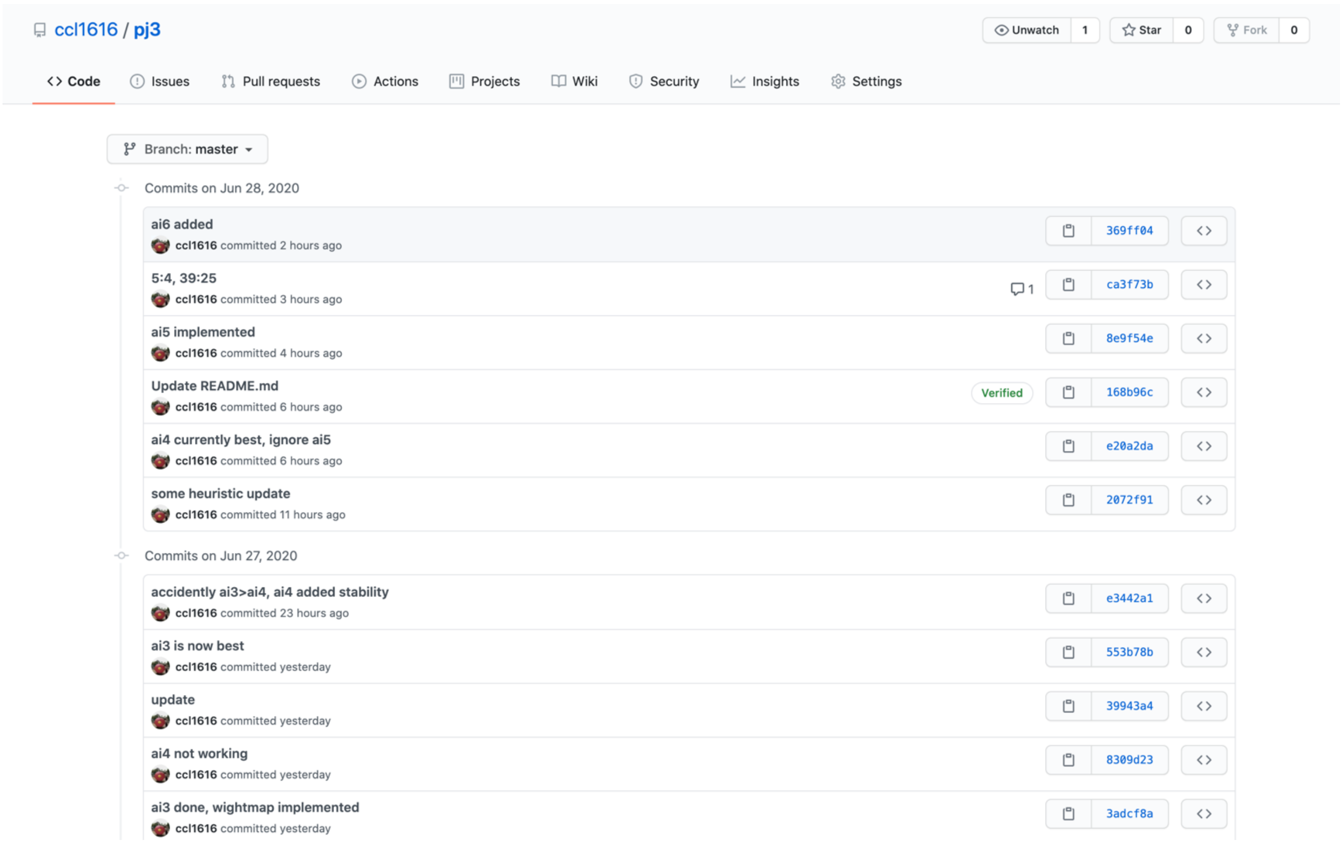


```cpp
int count_xc(myOthello cur){
    int heuristic = 0;
    int bk = 0;
    int wh = 0;
    // x-squares
    const int weight_x = 1000;
    for(int i = 0; i < 4; i ++){
        Point CO = corners[i];
        Point X = x_spots[i];
        if( !cur.board[X.x][X.y] && cur.board[X.x][X.y]!= cur.board[CO.x][CO.y] ){
            if(cur.board[X.x][X.y] == 1)
                bk -= weight_x;
            else wh += weight_x;
        }
    }

    // c-squares
    const int weight_c = 1000;
    for(int i = 0; i < 4; i ++){
        Point CO = corners[i];
        for(int j = 0; j < 2; j ++){
            Point C = c_spots[2*i+j];
            if( !cur.board[C.x][C.y] && cur.board[C.x][C.y] != cur.board[CO.x][CO.y] ){
                if(cur.board[C.x][C.y] == 1)
                    bk -= weight_c;
                else wh += weight_c;
            }
        }
    }
    heuristic = bk-wh;
    return heuristic;
}
```



```cpp
int count_stability(myOthello cur){
    int heuristic = 0;
    bool wingame = false;

    for(int i = 0; i < 4; i ++){
        // 對這四個角探討穩固性
        Point co = corners[i];
        if(!cur.board[co.x][co.y]) continue;
        int color = cur.board[co.x][co.y];
        int lv = 1;
        Point next = co + dir[i*2];
        bool good = true;
        while(good){
            if(lv == 8){
                wingame = true;
                break;
            }
            if(!on_board(next)){
                lv ++;
                Point nt;
                nt.x = co.x + lv*dir[i*2].x;
                nt.y = co.y + lv*dir[i*2].y;
                next = nt;
            }
            else if(cur.board[next.x][next.y] == color){
                next = next + dir_stability[i*2];
            }
            else{
                good = false;
                break;
            }
        }
        if(color == 1)
            heuristic += lv;
        else heuristic -= lv;
        if (wingame)
            heuristic += 10000;
    }
    return heuristic;
}
```

4. Version control

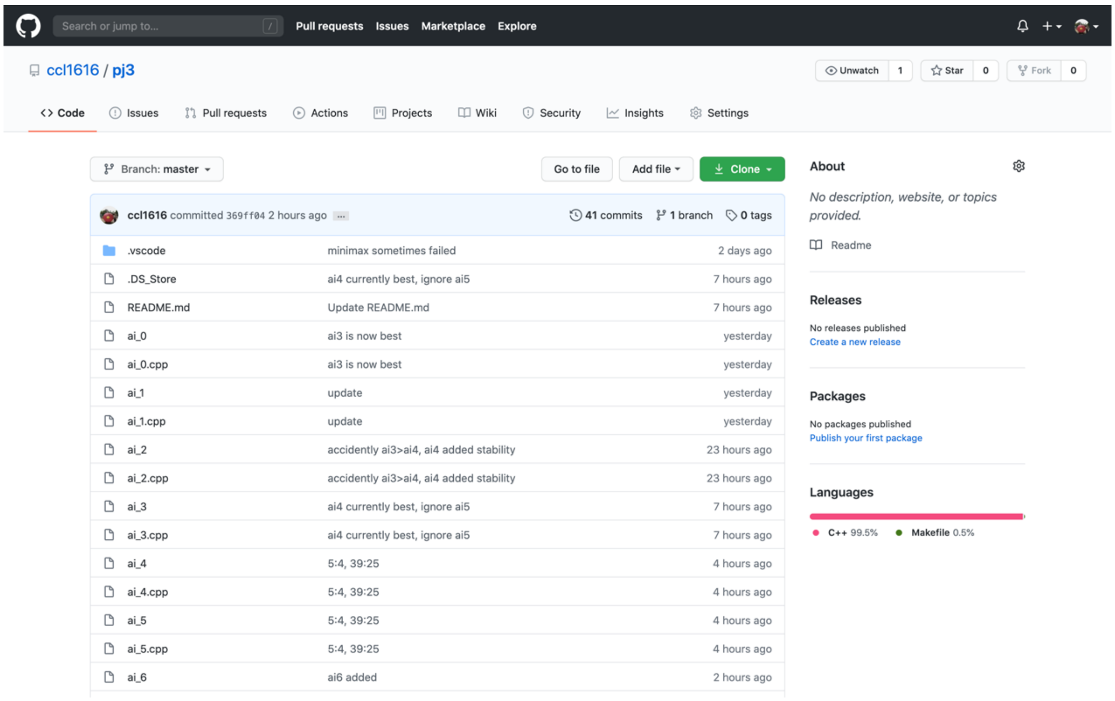


Commits on Jun 28, 2020

Commit	Author / time		SHA
ai6 added	ccl1616 committed 2 hours ago		369ff04
5:4, 39:25	ccl1616 committed 3 hours ago	1	ca3f73b
ai5 implemented	ccl1616 committed 4 hours ago		8e9f54e
Update README.md	ccl1616 committed 6 hours ago	Verified	168b96c
ai4 currently best, ignore ai5	ccl1616 committed 6 hours ago		e20a2da
some heuristic update	ccl1616 committed 11 hours ago		2072f91

Commits on Jun 27, 2020

Commit	Author / time	SHA
accidently ai3>ai4, ai4 added stability	ccl1616 committed 23 hours ago	e3442a1
ai3 is now best	ccl1616 committed yesterday	553b78b
update	ccl1616 committed yesterday	39943a4
ai4 not working	ccl1616 committed yesterday	8309d23
ai3 done, wightmap implemented	ccl1616 committed yesterday	3adcf8a



ccl1616 committed 369ff04 2 hours ago · 41 commits · 1 branch · 0 tags

.vscode	minimax sometimes failed	2 days ago
.DS_Store	ai4 currently best, ignore ai5	7 hours ago
README.md	Update README.md	7 hours ago
ai_0	ai3 is now best	yesterday
ai_0.cpp	ai3 is now best	yesterday
ai_1	update	yesterday
ai_1.cpp	update	yesterday
ai_2	accidently ai3>ai4, ai4 added stability	23 hours ago
ai_2.cpp	accidently ai3>ai4, ai4 added stability	23 hours ago
ai_3	ai4 currently best, ignore ai5	7 hours ago
ai_3.cpp	ai4 currently best, ignore ai5	7 hours ago
ai_4	5:4, 39:25	4 hours ago
ai_4.cpp	5:4, 39:25	4 hours ago
ai_5	5:4, 39:25	4 hours ago
ai_5.cpp	5:4, 39:25	4 hours ago
ai_6	ai6 added	2 hours ago

About

No description, website, or topics provided.

Readme

Releases

No releases published
Create a new release

Packages

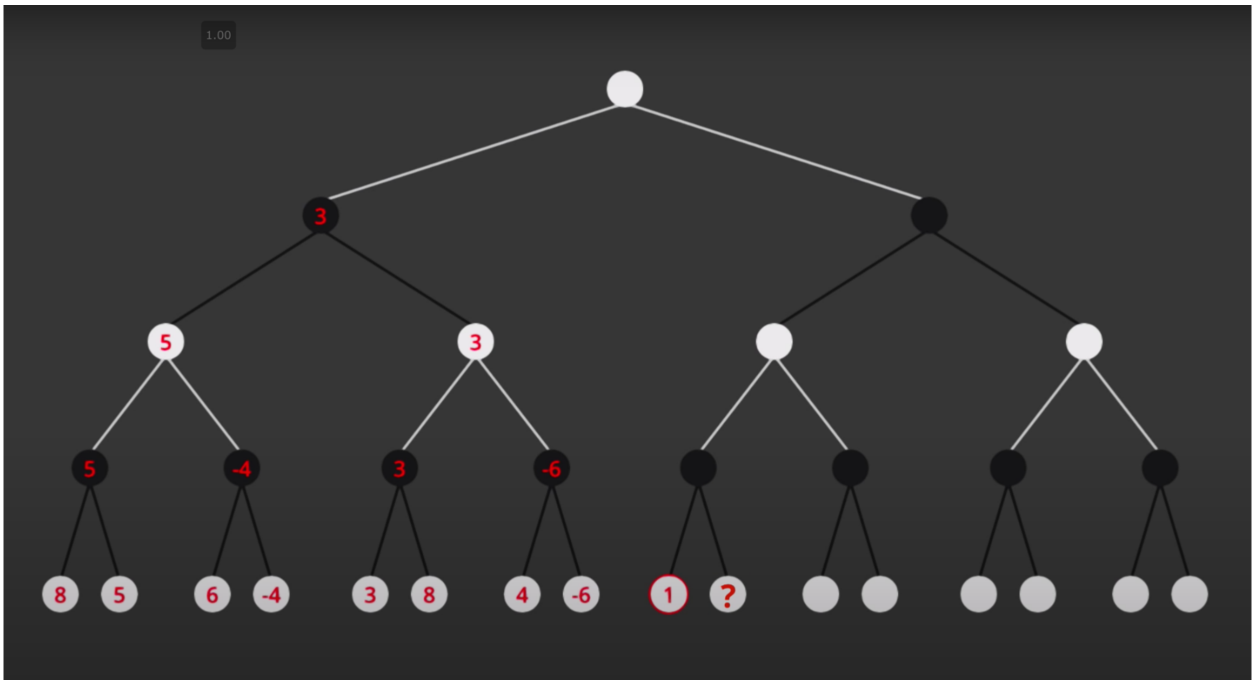
No packages published
Publish your first package

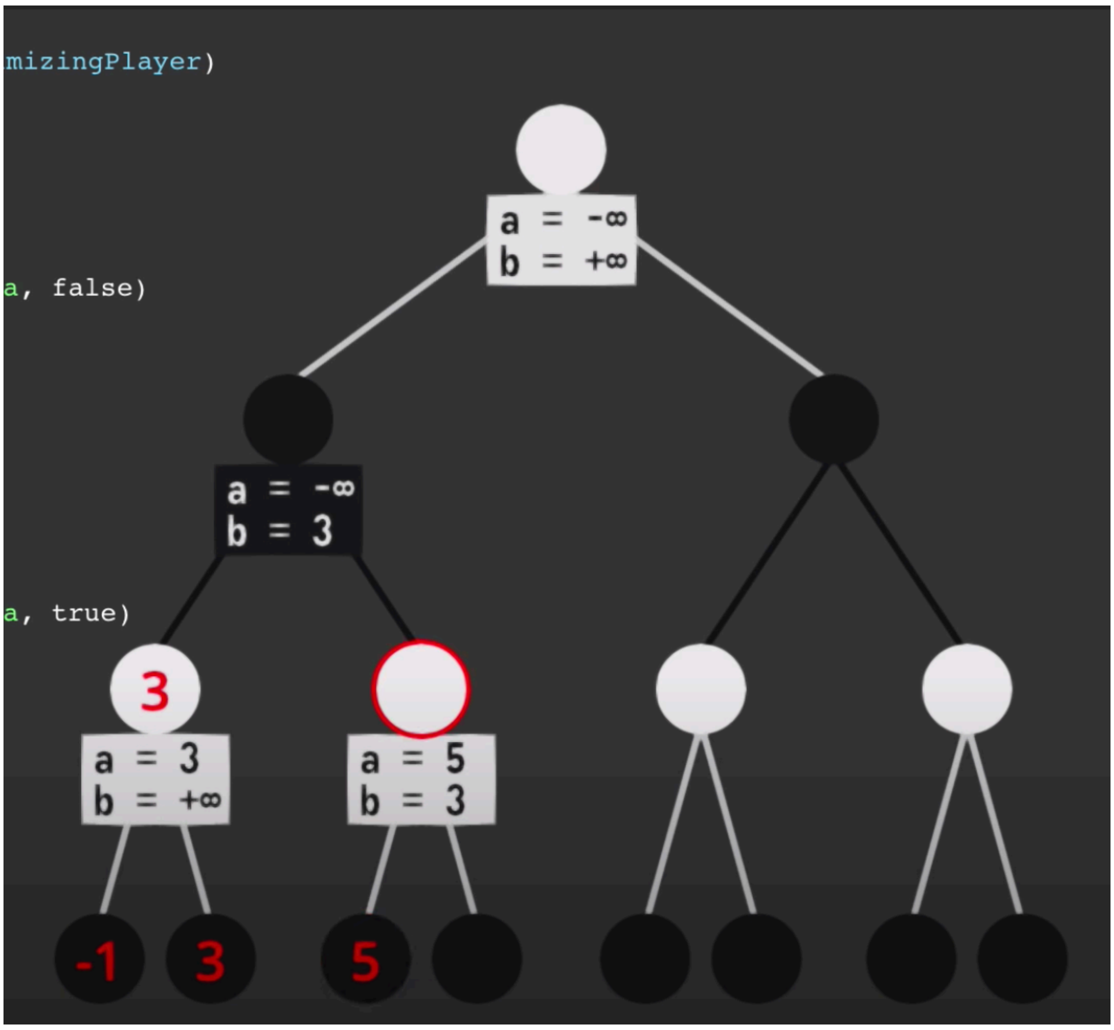
Languages

● C++ 99.5% ● Makefile 0.5%

5. Abprune

https://www.youtube.com/watch?v=l-hh51ncgDI





6. Reference
   https://github.com/eigenfoo/otto-othello