五子棋AI思路

第一种遍历，横的是否有5颗，纵的，主对角线，副对角线

这种直接统计黑色有多少颗，白色有多少颗，这种比较蠢，不符合人的思维。

第二种，直接遍历棋盘，计算开放度加棋子的连续，这种比较厉害

void AI() {

int maxScore = Integer.MIN\_VALUE;

int max\_x = -1, max\_y = -1;

// 遍历整个棋盘

for (int x = 0; x < board.length; x++) {

for (int y = 0; y < board[0].length; y++) {

// 只考虑空位置

if (board[x][y] == ' ') {

int score = 0;

// 考虑所有可能的方向

for (int dx = -1; dx <= 1; dx++) {

for (int dy = -1; dy <= 1; dy++) {

if (dx != 0 || dy != 0) {

score += score(board, x, y, dx, dy);

}

}

}

// 如果这个位置的分数比当前最大的分数还要大，那么更新最大分数和位置

if (score > maxScore) {

maxScore = score;

max\_x = x;

max\_y = y;

}

}

}

}

// 在最好的位置上放置一个棋子

if (max\_x != -1 && max\_y != -1) {

board[max\_x][max\_y] = 'B'; // 假设AI是黑棋

}

}

int getopenness(char[][] board, int x, int y, int dx, int dy) {

int openness = 0;

for (int i = 1; i <= 4; i++) {

if (x + i \* dx < 0 || x + i \* dx >= board.length || y + i \* dy < 0 || y + i \* dy >= board[0].length || board[x + i \* dx][y + i \* dy] != ' ') {

break;

}

openness++;

}

for (int i = 1; i <= 4; i++) {

if (x - i \* dx < 0 || x - i \* dx >= board.length || y - i \* dy < 0 || y - i \* dy >= board[0].length || board[x - i \* dx][y - i \* dy] != ' ') {

break;

}

openness++;

}

return openness;

}

int score(char[][] board, int x, int y, int dx, int dy) {

int blackcount = 0;

int whitecount = 0;

for (int i = 0; i < 5; i++) {

if (x + i \* dx < 0 || x + i \* dx >= board.length || y + i \* dy < 0 || y + i \* dy >= board[0].length) {

return 0;

}

if (board[x + i \* dx][y + i \* dy] == 'B') {

blackcount++;

} else if (board[x + i \* dx][y + i \* dy] == 'W') {

whitecount++;

}

}

if (blackcount > 0 && whitecount > 0) {

return 0;

}

int openness = getopenness(board, x, y, dx, dy);

if (blackcount == 4 && openness >= 2) {

return 10000; // 活四

} else if (blackcount == 4 && openness == 1) {

return 1000; // 冲四

} else if (blackcount == 3 && openness >= 2) {

return 1000; // 活三

} else if (blackcount == 3 && openness == 1) {

return 100; // 眠三

} else if (blackcount == 2 && openness >= 2) {

return 100; // 活二

} else if (blackcount == 2 && openness == 1) {

return 10; // 眠二

} else {

return 1; // 其他

}

}