极小极大算法

实验内容

**A，B两名玩家轮流连接一次图中带有数字的点之间的虚线（A先手），每条虚线只能被连接一次，每当一名玩家将一个或多个三角形（指边长为一单位的最小三角形）的所有虚线全部连接时，他连成的三角形个数将计入他的总分，并且，他将被奖励一个回合（换言之对手跳过一次行动）。当所有的18条虚线均被连接时游戏结束，得分高者获胜。**

实验原理

极小极大算法 α-β剪枝

实验代码

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// 保存每条边代表的数字

int edge[11][11] = {

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 1, 3, 5, 0, 0, 0, 0, 0},

{0, 2, 1, 0, 0, 6, 8, 0, 0, 0, 0},

{0, 0, 3, 0, 0, 4, 0, 9, 11,0, 0},

{0, 0, 5, 6, 4, 0, 7, 0, 12,14,0},

{0, 0, 0, 8, 0, 7, 0, 0, 0, 15,17},

{0, 0, 0, 0, 9, 0, 0, 0, 10,0, 0},

{0, 0, 0, 0, 11,12,0, 10,0, 13, 0},

{0, 0, 0, 0, 0, 14,15,0, 13, 0, 16},

{0, 0, 0, 0, 0, 0, 17,0, 0, 16,0},

};

// 保存每个单位三角形代表的数字

int tri[9] = { 7, 56, 98, 448, 3584, 6160, 28672, 49280, 229376 };

int end\_state = (1 << 18) - 1; // 终结状态 2^18 - 1 ，即所有边均被填充

int inf = (1 << 20);

int next\_state(int cur\_state, int edge, int\* cnt)

{

int i;

int new\_state = (cur\_state | edge); // 当前局面并上一条边形成新局面

for (i = 0; i < 9; i++) // 如果新局面能形成一个新的单位三角形，则 cnt++

if (((cur\_state & tri[i]) != tri[i]) && ((new\_state & tri[i]) == tri[i]))

(\*cnt)++;

return new\_state;

}

int alpha\_beta(int player, int cur\_state, int alpha, int beta, int ca, int cb)

{

int remain;

// 如 A 得到 5 分以上则 A 赢

// 如 B 得到 5 分以上则 A 输

if (ca >= 5) return 1;

if (cb >= 5) return -1;

remain = ((~cur\_state) & end\_state); // 计算剩余可走的边

if (player) { // A 走

while (remain) { // 有可走边

int move = (remain & (-remain)); // 选择一条可走边

int ta = ca;

int val;

// A 填了边后形成新的局面

int new\_state = next\_state(cur\_state, move, &ta);

if (ta > ca) // 如果 A 得分了，则 A 继续填一条边

val = alpha\_beta(player, new\_state, alpha, beta, ta, cb);

else // 否则轮到 B 填

val = alpha\_beta(player ^ 1, new\_state, alpha, beta, ca, cb);

if (val > alpha)

alpha = val;

if (alpha >= beta)

return alpha;

remain -= move; // 把边 move 从剩余可选边 remain 中移除

}

return alpha;

}

else { // B 走

while (remain) {

int move = (remain & (-remain));

int tb = cb;

int val;

int new\_state = next\_state(cur\_state, move, &tb);

if (tb > cb)

val = alpha\_beta(player, new\_state, alpha, beta, ca, tb);

else

val = alpha\_beta(player ^ 1, new\_state, alpha, beta, ca, cb);

if (val < beta)

beta = val;

if (alpha >= beta)

return beta;

remain -= move;

}

return beta;

}

}

int main()

{

#if 0

freopen("in.txt", "r", stdin);

#endif

int T, w = 0;

scanf\_s("%d", &T);

while (T--) {

int i;

int n;

int ans;

int cnt = 0; // 偶数轮到 A 走，奇数轮到 B 走

int cur\_state = 0; // 当前局面

int ca = 0; // A 的得分

int cb = 0; // B 的得分

int ta, tb;

int alpha = -inf;

int beta = inf;

scanf\_s("%d", &n);

for (i = 0; i < n; i++) {

int u, v;

ta = ca;

tb = cb;

scanf\_s("%d%d", &u, &v);

cur\_state = next\_state(cur\_state, 1 << edge[u][v], (cnt & 1) ? (&cb) : (&ca));

if (ta == ca && tb == cb) // 不得分，轮到对方走

cnt++;

}

if (cnt & 1)

ans = alpha\_beta(0, cur\_state, alpha, beta, ca, cb);

else

ans = alpha\_beta(1, cur\_state, alpha, beta, ca, cb);

if (ans > 0)

printf("Game %d: A wins.\n", ++w);

else

printf("Game %d: B wins.\n", ++w);

}

return 0;

}

实验结果

