



# 人工智能实验报告

学 院 计算机学院

专 业 计算机科学与技术

班 级 计科5班

学 号 3216004669

姓 名 林文敏

指导教师 张伯泉

（2018年 9月）

实验课程名称：\_\_人工智能\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验项目名称** | α-β 剪支博弈树算法实现一字棋 | | | **实验成绩** |  |
| **实 验 者** | **林文敏** | **专业班级** | **计科5班** | **学 号** | **3216004669** |
| **实验日期** | **2018.09.27** | | |  |  |
| 一、实验内容 α-β 剪支博弈树算法实现一字棋 二、实验设计（原理分析及流程）1 游戏规则 "一字棋"游戏（又叫"三子棋"或"井字棋"），是一款十分经典的益智小游戏。"井字棋"  的棋盘很简单，是一个 3×3 的格子，很像中国文字中的"井"字，所以得名"井字棋"。一方首先三子连成一线就胜利。 2 α-β剪支算法 α -β 剪支技术的基本思想或算法是，边生成博弈树边计算评估各节点的倒推值，并且根据评估出的倒推值范围，及时停止扩展那些已无必要再扩展的子节点，即相当于剪去了博弈树上的一些分支，从而节约了机器开销，提高了搜索效率。  具体方法如下：  (1) 对于一个与节点 MIN，若能估计出其倒推值的上确界 β，并且这个 β 值不大于 MIN 的父节点(一定是或节点)的估计倒推值的下确界 α，即 α≥β，则就不必再扩展该MIN 节点的其余子节点了(因为这些节点的估值对 MIN 父节点的倒推值已无任何影响了)。这一过程称为 α 剪枝。  (2) 对于一个或节点 MAX，若能估计出其倒推值的下确界 α，并且这个 α 值不小于 MAX 的父节点(一定是与节点)的估计倒推值的上确界 β，即 α≥β，则就不必再扩展该 MAX 节点的其余子节点了(因为这些节点的估值对 MAX 父节点的倒推值已无任何影响 了)。这一过程称为 β 剪枝。 3 输赢判断算法设计 因为每次导致输赢的只会是当前放置的棋子,输赢算法中只需从当前点开始扫描判断是否已经形成三子。对于这个子的八个方向判断是否已经形成三子。如果有，则说明有一方胜利，如果没有则继续搜索，直到有一方胜利或者搜索完整个棋盘。 | | | | | |
| **三、实验代码及数据记录**  **1.代码**  #include <bits/stdc++.h>  using namespace std;  typedef long long ll;  typedef pair<int, int> P;  const int MAXN = 5;  const int INF = 0x3f3f3f3f;  char maze[MAXN][MAXN];  void initMaze(); // 初始化数组  void printMaze(); // 画出当前棋盘  int checkWin(char x); // 检查是否赢了  int myTurn(); // 使用者下棋  int aiTurn(); // 机器人下棋  P findBest(int depth, int result); // 找到最佳落子点  int calculate(); // 计算 e(p)  int main()  {  initMaze();  printf("如果想让机器先走请在下方输入：0 \n ");  int op;  scanf("%d", &op);  int isOver = 0;  if( op ) {  while( !isOver ) {  isOver = myTurn();  if( isOver ) break;  system("cls");  isOver = aiTurn();  }  }  else {  while( !isOver ) {  isOver = aiTurn();  if( isOver ) break;  isOver = myTurn();  system("cls");  }  }  if( isOver == 1 ) puts("恭喜你，获得胜利~");  else if( isOver == 2 ) puts("承让承认，我们的机器人胜利啦~");  else puts("你和我们机器人一样聪明哟~");  system("pause");  return 0;  }  void initMaze()  {  for( int i = 0; i < MAXN; ++ i ) {  for( int j = 0; j < MAXN; ++ j ) {  maze[i][j] = ' ';  }  }  }  void printMaze()  {  puts("目前棋况如下:\n");  for( int i = 0; i < MAXN; ++ i ) {  for( int j = 0; j < MAXN; ++ j ) {  if( j < 2 ) {  printf(" %c%c |", maze[i][j], " \n"[j == MAXN - 1]);  }  else {  printf(" %c%c", maze[i][j], " \n"[j == MAXN - 1]);  }  }  if( i < 2 ) {  printf(" ---------------\n");  }  puts("");  }  }  int checkWin(char x)  {  int diaL = 0, diaR = 0, equ = 0;  for( int i = 0; i < MAXN; ++ i ) {  int raw = 0, col = 0;  for( int j = 0; j < MAXN; ++ j ) {  if( maze[i][j] == x ) raw++;  if( maze[j][i] == x ) col++;  if( maze[i][j] == ' ' ) equ++;  }  if( maze[i][i] == x ) diaL++;  if( maze[i][MAXN - i - 1] == x ) diaR++;  if( raw == MAXN || col == MAXN || diaL == MAXN || diaR == MAXN ) return 1;  }  if( !equ ) return 2;  return 0;  }  int myTurn()  {  puts("输入你想要放置的位置，例如放置在第一行第一个位置则输入：0 0");  int x, y;  scanf("%d %d", &x, &y);  while( maze[x][y] != ' ' ) {  printf("啊呀，这个位置已经被放过啦,再试一次？\n");  scanf("%d %d", &x, &y);  }  maze[x][y] = 'O';  printMaze();  int result = checkWin('O');  if( result == 1 ) return 1;  else if( result == 2 ) return 3;  return 0;  }  int aiTurn()  {  puts("智能机器人下好棋啦\n");  pair<int, int> loc = findBest(3, -INF);  maze[loc.second / MAXN][loc.second % MAXN] = 'X';  printMaze();  int result = checkWin('X');  if( result == 1 ) return 2;  else if( result == 2 ) return 3;  return 0;  }  P findBest(int depth, int result)  {  if( !depth ) return make\_pair(calculate(), -1);  if( depth & 1 ) {  P Max = make\_pair(-INF, -1);  for( int i = 0; i < MAXN; ++ i ) {  for( int j = 0; j < MAXN; ++ j ) {  if( maze[i][j] == ' ' ) {  maze[i][j] = 'X';  if( calculate() == INF ) {  maze[i][j] = ' ';  return Max = make\_pair(INF, i \* MAXN + j);  }  else if( checkWin('X') == 2 ) {  maze[i][j] = ' ';  return Max = make\_pair(0, i \* MAXN + j);  }  P tmp = findBest(depth - 1, Max.first);  if( Max.first <= tmp.first ) {  Max = tmp;  Max.second = i \* MAXN + j;  }  maze[i][j] = ' ';  if( depth == 1 && Max.first >= result ) return Max;  }  }  }  return Max;  }  else {  P Min = make\_pair(INF, -1);  for( int i = 0; i < MAXN; ++ i ) {  for( int j = 0; j < MAXN; ++ j ) {  if( maze[i][j] == ' ' ) {  maze[i][j] = 'O';  if( calculate() == -INF ) {  maze[i][j] = ' ';  return Min = make\_pair(-INF, i \* MAXN + j);  }  else if( checkWin('O') == 2 ) {  maze[i][j] = ' ';  return Min = make\_pair(0, i \* MAXN + j);  }  P tmp = findBest(depth - 1, Min.first);  if( Min.first >= tmp.first ) {  Min = tmp;  Min.second = i \* MAXN + j;  }  maze[i][j] = ' ';  if( result > Min.first ) return Min;  }  }  }  return Min;  }  }  int calculate()  {  if( checkWin('X') == 1 ) return INF;  else if( checkWin('O') == 1 ) return -INF;  int resX = 0, resO = 0;  int diaLX = 0, diaRX = 0, diaLO = 0, diaRO = 0;  for( int i = 0; i < MAXN; ++ i ) {  int rawX = 0, colX = 0, rawO = 0, colO = 0;  for( int j = 0; j < MAXN; ++ j ) {  if( maze[i][j] == 'X' || maze[i][j] == ' ' ) rawX++;  if( maze[j][i] == 'X' || maze[j][i] == ' ' ) colX++;  if( maze[i][j] == 'O' || maze[i][j] == ' ' ) rawO++;  if( maze[j][i] == 'O' || maze[j][i] == ' ' ) colO++;  }  if( maze[i][i] == 'X' || maze[i][i] == ' ' ) diaLX++;  if( maze[i][MAXN - i - 1] == 'X' || maze[i][MAXN - i - 1] == ' ' ) diaRX++;  resX += (rawX == MAXN) + (colX == MAXN);  if( maze[i][i] == 'O' || maze[i][i] == '.' ) diaLO++;  if( maze[i][MAXN - i - 1] == 'O' || maze[i][MAXN - i - 1] == ' ' ) diaRO++;  resO += (rawO == MAXN) + (colO == MAXN);  }  resX += (diaLX == MAXN) + (diaRX == MAXN);  resO += (diaLO == MAXN) + (diaRO == MAXN);  return resX - resO;  }  maze[3][3] : 存储棋盘现在落子情况  **2. 结果截图**  选择先走后走情况    落子提示      机器赢提示：    平局提示：    由于真的没有赢过机器 人赢提示并无截图 | | | | | |