

**课 程 实 验 报 告**

**课程名称： 人工智能导论**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期：**

**计算机科学与技术学院**

# 一、α-β剪枝算法介绍

α−β剪枝是一种搜索算法，用以减少极小化极大算法搜索树的节点数。这是一种对抗性搜索算法，主要应用于机器游玩的二人游戏（如井字棋、象棋、围棋）。当算法评估出某策略的后续走法比之前策略的还差时，就会停止计算该策略的后续发展。该算法和极小化极大算法所得结论相同，但剪去了不影响最终决定的分枝。

α−β剪枝的名称来自计算过程中传递的两个边界，这些边界基于已经看到的搜索树部分来限制可能的解决方案集。其中，α表示目前所有可能解中的最大下界，β表示目前所有可能解中的最小上界。

因此，如果搜索树上的一个节点被考虑作为最优解的路上的节点（或者说是这个节点被认为是有必要进行搜索的节点），那么它一定满足以下条件（N是当前节点的估价值）：α≤N≤β

  在我们进行求解的过程中，α和β会逐渐逼近。如果对于某一个节点，出现了α > β的情况，那么说明这个点一定不会产生最优解了，所以，我们就不再对其进行扩展（也就是不再生成子节点），这样就完成了对博弈树的剪枝。

# 二、算法分析

## 1 全局变量说明

1. ttt[3][3]，用于记录tic-toc-toe的棋盘
2. e，用于记录电脑胜的可能情况数；p，用于记录玩家胜的可能情况数
3. cnt，用于记录棋盘上棋子的个数

## 2 函数说明

### void Print()

功能：输出棋盘的情况。

如果数组中该位置的值为1，表示为电脑下的子，输出“O”。如果为-1，为玩家下的子，输出“X”。若为0，为空，输出“-”。

### int Win()

功能：判断赢家

当棋盘中有三个1连成一线，表示电脑赢，返回1；当棋盘中有三个-1连成一线，表示玩家赢了，返回-1。否则返回0。

### int Back\_up()

功能：计算电脑或玩家胜分别有几个机会胜，返回倒推值

1. 首先考虑电脑的情况。将棋盘上的空位都标为1，检查是否存在三个1连成一线的机会，e赋值为机会的个数。
2. 再考虑玩家的情况。将棋盘上的空位都标为-1，检查是否存在三个-1连成一线的机会，p赋值为机会的个数的相反数。
3. 返回e+p的值(可以理解为电脑的分数)。

### void Input()

功能：玩家输入下子位置，记录进棋盘中

1. 输入行列数，棋盘上相应的格子赋值为-1
2. cnt++

### int AlphaBeta(int &value,int d,bool first)

功能：模拟下棋步骤，进行α-β剪枝，返回倒推值

令先下的一方为α结点，后下的一方为为β结点。

参数说明：value为节点的评估值，d为搜索深度， first标记是否为先下的一方。

变量说明：bool类型prone记录是否剪枝，tmp为当前搜索节点的倒推值，ab为最终倒推值(子结点的最小β或最大α)。

1. 当模拟先下的一方下棋时，ab初始化为最大评估值不会超过的数(代码中选取10)。for循环遍历搜索树，模拟每一种基于当前棋盘的可能：若模拟下一步后对方胜了，直接结束搜索，进行剪枝；否则进行下一深度搜索，换一方模拟。
2. 当d=6时，即模拟6步后(选择模拟6步是因为在模拟5步的情况下，有些情况电脑仍然会输给玩家；在模拟6步的情况下，电脑不可能输给玩家。如果玩家想赢电脑，可以将最大搜索深度改为小于6的正整数)，或d+cnt=9时，即棋盘下满时，不再继续模拟，直接调用Back\_up返回当前模拟棋盘上电脑的分数。
3. 首先考虑先下的情况。一次模拟结束后，ab更新为当前最小的β。如果ab<value，即后下一方的最小分小于先下一方的最大分，β<α，α剪枝，退出当前循环。
4. 若该节点所有情况模拟结束后ab>value，后下一方的最小分大于先下一方的最大分，即β>α，可以作为下子位置，value更新为ab，作为节点的评估值。模拟完后复原棋盘。
5. 先考虑后下的情况基本相同。不同在于：刚开始ab赋值为-10(最小评估值不会小于的值)。每次模拟后ab更新为最大α，如果ab>value，即α>β，进行β剪枝。若ab<value，即该节点模拟结束后α<β，value更新为ab，作为节点的评估值。

### void PvE()

功能：模拟人机对战，判断胜负

变量说明：row、col：电脑下子的位置，value：根节点的评估值，tmp当前子结点的倒推值，d为搜索深度。

1. 棋盘初始化为0，value、tmp初始化为-10(最小倒推值不会小于的值)，d初始化为1，cnt初始化为0。
2. 让玩家选择哪方先下。若玩家先下，玩家输入下子位置，记录进棋盘。如果玩家下完这步后赢了，结束对战。否则for循环遍历搜索树，调用剪枝函数确定最优的下子位置。如果电脑下完后胜了，结束对战。否则，若剪枝结束后tmp>value，即后下一方最小分大于先下一方最大分，可以作为下子位置，value更新为tmp，row，col更新为当前位置。
3. 调用Back\_up，若p和e都为0即双方都不可能胜，平局，结束游戏。
4. 若电脑先下，情况基本相同。先让电脑进行剪枝，若没有找到能落子的位置，即不输给对玩家的位置已经全部被占，不管下在哪个空位玩家都会赢，结束游戏。否则让玩家下子。模拟结束后还原棋盘，初始化变量。

### int main()

功能：主函数

While循环，调用PvE函数进行人机对战，对战结束后进行继续或退出操作。

# 三、实验结果及分析

例1.电脑先下，双方和局。第一步，电脑搜索到胜的情况最多的位置(2,2)，一共有8种。电脑能够找到胜的几率最大的位置，并阻止玩家三子一线。

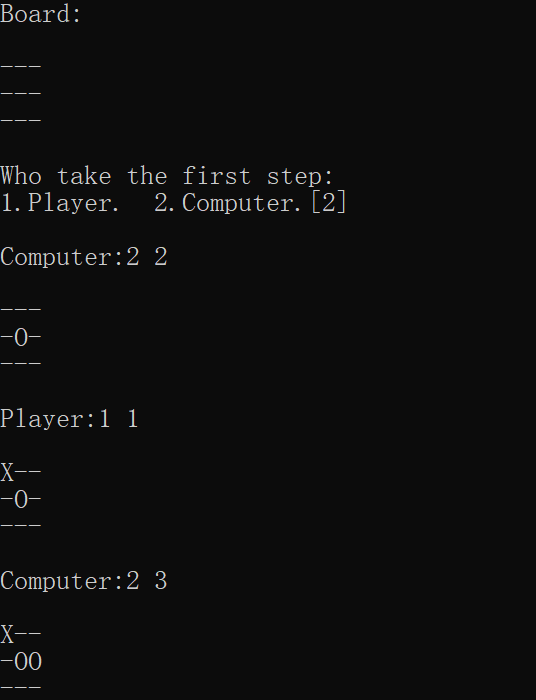


图1 例1截图1

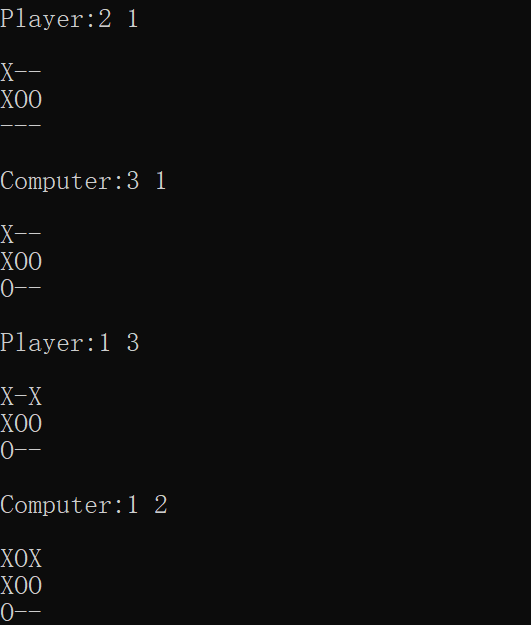


图2 例1截图2

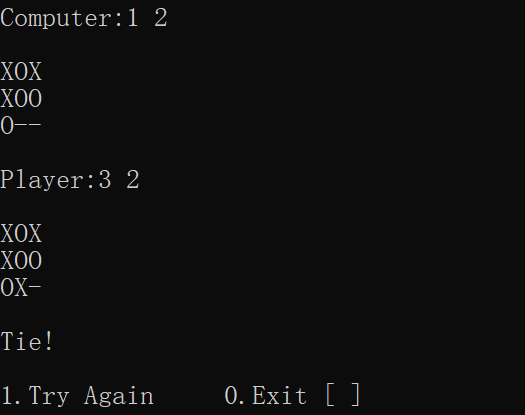


图3 例1截图3

例2.玩家先下，双方和局

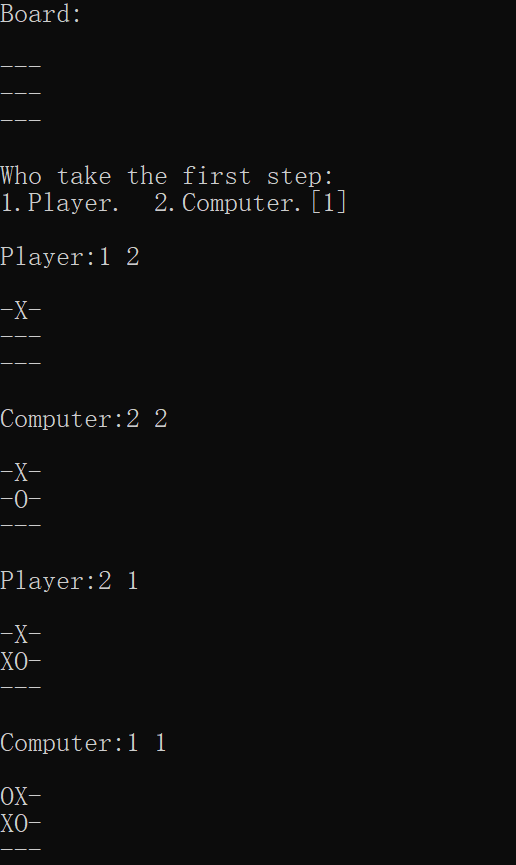


图4 例2截图1

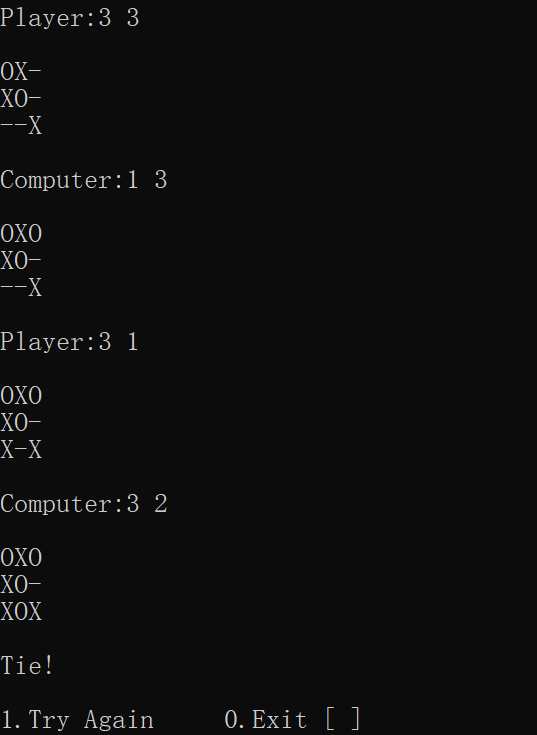


图5 例2截图2

例3.电脑先下，电脑胜

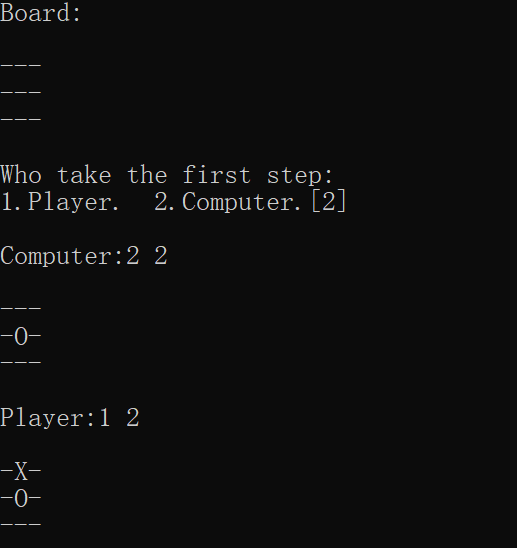


图6 例3截图1

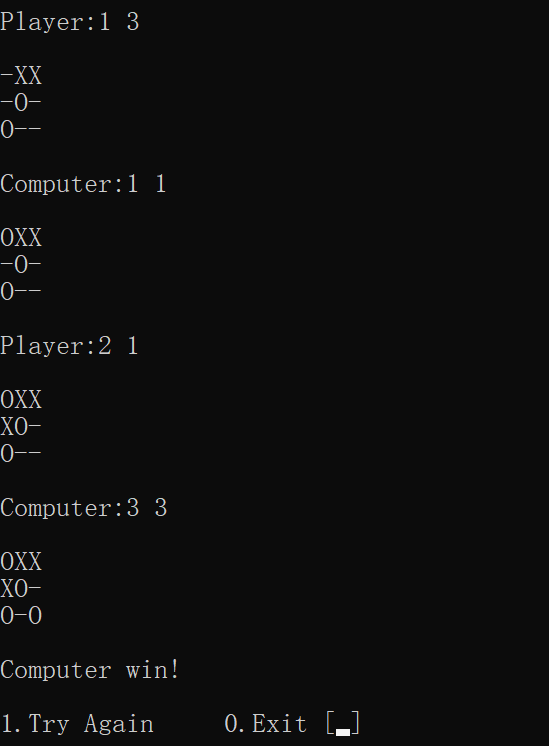


图7 例3截图2

例4.玩家先下，玩家胜。由于当最大搜索深度为6时，玩家不能赢电脑，因此将搜索深度改成5，模拟玩家胜的情况。

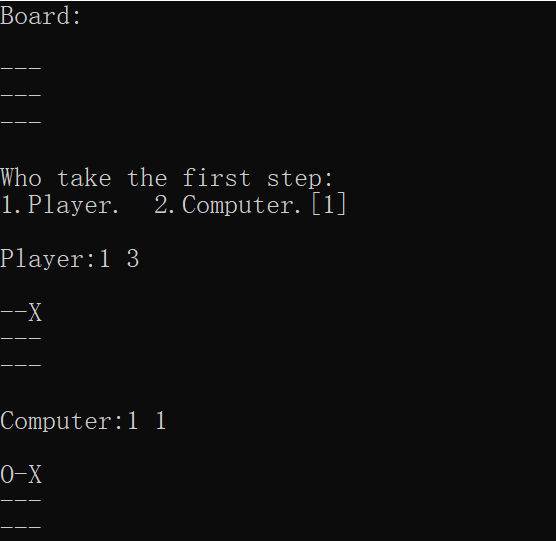


图8 例4截图1

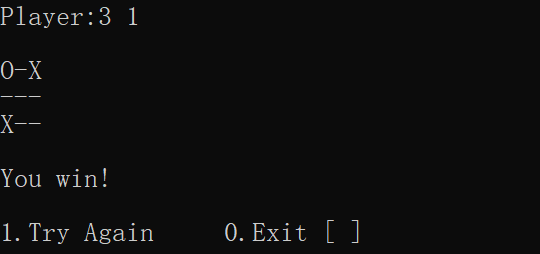


图9 例4截图2

# 四、α-β剪枝算法优缺点分析

优点：相比minimax算法减少了搜索树的分枝，将搜索时间用在更有希望的子树上，继而提升搜索深度。

缺点：严重依赖于算法的寻找顺序。如果总是先去搜索最坏的分支，β剪枝就不会发生，最终会搜索遍整个搜索树，和minimax算法没有区别。

# 五、心得体会

写完这次结课作业，我对α-β剪枝算法原理有了比较清晰的理解，体会到了它较minimax算法的优点，并且能够写井字棋的代码进行应用。井字棋的棋盘小，下子情况不多，搜索树不大，所以井字棋的α-β剪枝并不复杂，搜索深度只要达到4就可以不输给人类，搜索时间很短。但如果是五子棋或其他更复杂的棋，棋盘更大，搜索深度也大大增加，分支也会指数倍增加。如果程序能挑选最好的分支先进行模拟，那么有效分支的数量就接近于实际分支数量的平方根，搜索需要的时间会大大减少，这是算法可能达到的最好的情况，可能需要用到反馈和奖励方面的知识，希望之后能够学会使用。