|  |
| --- |
| 电子科技大学  **计算机专业类课程** |
| **实验报告** |
| **课程名称：程序设计基础**  **学院专业：计算机科学与工程学院**  **学生姓名：陈天逸**  **学　　号：2023080901005**  **指导教师：俸志刚** |
| **日　　期：2024年1月4日** |
|  |
| 电子科技大学计算机学院实验中心 |

**电 子 科 技 大 学**

**实 验 报 告**

**实验一**

# 一、实验室名称：

电子科技大学清水河校区品学楼A区305教室

# 二、实验项目名称：

五子棋算法设计实现

# 实验目的：

# 通过编写五子棋AI程序，熟练掌握本学期所学数据结构类型知识并学习博弈树、蒙特卡洛树等新知识

# 实验主要内容：

编写符合以下要求的AI五子棋程序：

·人机对决

·设计出的程序能实现五子棋游戏的所有规则

·编写出的AI算法具有一定的对局能力且落子间隔不过长

·代码语言不过于冗杂、繁复

# 五、实验器材（设备、元器件）：

联想ThinkBook 16 G5+ IRH

处理器 13th Gen Intel(R) Core(TM) i7-13700H 2.40 GHz

机带 RAM 32.0 GB (31.7 GB 可用)

系统类型 64 位操作系统, 基于 x64 的处理器

Windows 11 家庭中文版

CLion2023.2.1 GBK

# 实验步骤：

1.问题描述

编写符合以下要求的AI五子棋程序：

·人机对决

·设计出的程序能实现五子棋游戏的所有规则

·编写出的AI算法具有一定的对局能力且落子间隔不过长

·代码语言不过于冗杂、繁复

1. 算法分析与概要设计

鉴于五子棋游戏本身零和博弈的特性，本次实验需要用到的算法有：蒙特卡洛树；Minmax算法；α-β剪枝算法；启发式搜索算法（优化）

本次设计在思考实验需要实现的功能基础上完成对函数的分装，通过C++语言可用函数与本学期所学结构体、链表等数据结构知识构建出完整的五子棋AI程序代码。在代码运行过程中通过结合蒙特卡洛树与估值函数，从而利用算法达成快速搜索最佳落点的目的，实现AI功能

1. 核心算法的详细设计与实现

零和博弈（zero-sum game），又称零和游戏，与非零和博弈相对，是博弈论的一个概念，属非合作博弈。它是指参与博弈的各方，在严格竞争下，一方的收益必然意味着另一方的损失，博弈各方的收益和损失相加总和永远为“零”，故双方不存在合作的可能。

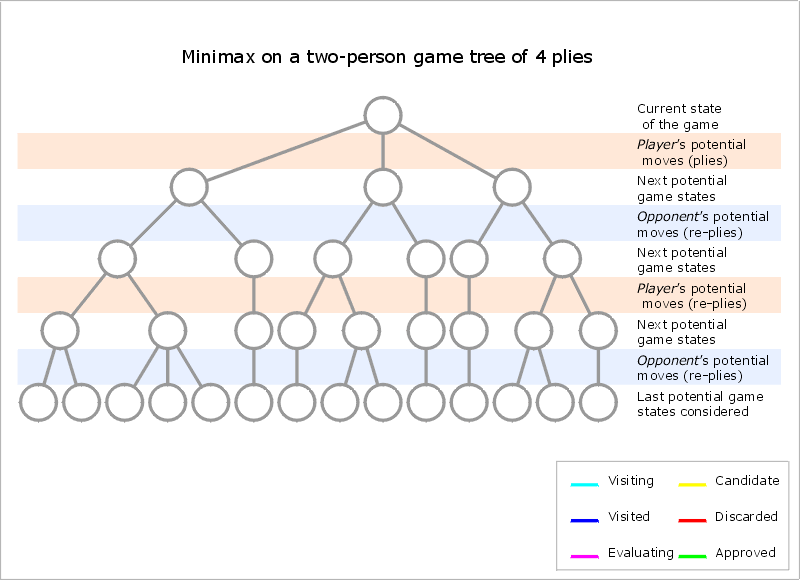
博弈树（game tree）是一种特殊的根树，它是人工智能领域一个重要的研究课题。博弈树可以表示两名游戏参与者之间的一场博弈（游戏），他们交替行棋，试图获胜。

零和情况：双方利益冲突，对一方有利则对另一方不利。一般对节点N取一个估价函数f(N)，一共两类节点：

——叫Max的极大节点追求最大化，有选择时肯定选值最大的；

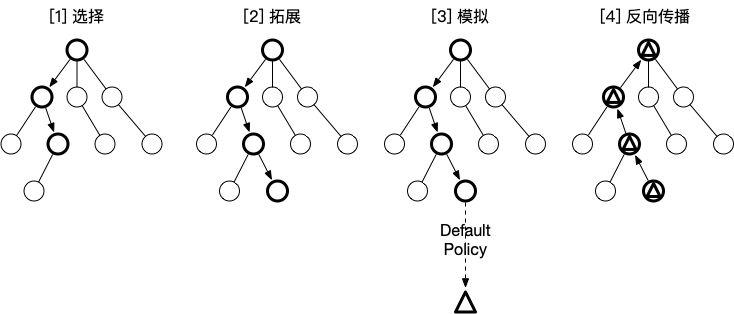
——叫Min的极小节点追求最小化，有选择时肯定选值最小的。

Minmax算法如图所示：

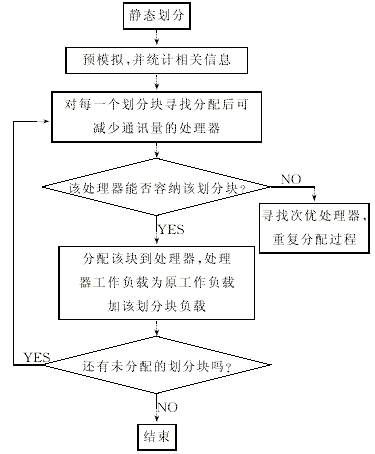


1987年Bruce Abramson在他的博士论文中提出了基于蒙特卡洛方法的树搜索这一想法。这种算法简而言之是用蒙特卡洛方法估算每一种走法的胜率。如果描述的再具体一些，通过不断的模拟每一种走法，直至终局，该走法的模拟总次数N，与胜局次数W，即可推算出该走法的胜率为 W/N。

通过随机的对游戏进行推演来逐渐建立一棵不对称的搜索树的过程。



结合上述三者理论，我们可以通过minmax算法+估值实现在蒙特卡洛树上选取零和博弈对己方而言的最佳位置，从而达成消极博弈中的己方优势。

启发式算法（heuristic algorithm)是相对于最优化算法提出的。一个问题的最优算法求得该问题每个实例的最优解。启发式算法可以这样定义：一个基于直观或经验构造的算法，在可接受的花费（指计算时间和空间）下给出待解决组合优化问题每一个实例的一个可行解，该可行解与最优解的偏离程度一般不能被预计。现阶段，启发式算法以仿自然体算法为主，主要有蚁群算法、模拟退火法、神经网络等。

代码实现：

估值函数代码实现：

long GoBang::score(point p, int name,dirmark& mark)  
{  
 long totalScore = 0;  
 sum result;  
 sum temp;  
 if (mark.find(p) == mark.end())  
 mark.insert(make\_pair(p, vector<bool>(5, false)));  
 for (int i = 1; i <= 4; i++)  
 {  
 if (mark[p][i])  
 continue;  
 direction d = mapping(i);  
 int length = 1;  
 point le, ri, tempPoint;  
 char left[5], right[5];  
 getLinkPiece(length, le, name, p, d, -1);  
 getLinkPiece(length, ri, name, p, d, 1);  
 getBoundary(left, le, right, ri, d, name);  
 temp = situaltionAnalysis(length, name, left, right);  
 setDirMark(p, name, i, d, mark);  
 result += temp;  
 point lePoint=nextPoint(p,d,-1), riPoint=nextPoint(p,d,1);  
 for (int j = 1; j <= 4; j++)  
 {  
 if (chessBoard[lePoint.x][lePoint.y] == name)  
 {  
 if (mark.find(lePoint) != mark.end())  
 mark[lePoint][i] = true;  
 else  
 {  
 mark.insert(make\_pair(lePoint, vector<bool>(5, false)));  
 mark[lePoint][i] = true;  
 }  
 }  
 lePoint = nextPoint(lePoint, d, -1);  
 }  
 for (int j = 1; j <= 4; j++)  
 {  
 if (chessBoard[riPoint.x][riPoint.y] == name)  
 {  
 if (mark.find(riPoint) != mark.end())  
 mark[riPoint][i] = true;  
 else  
 {  
 mark.insert(make\_pair(riPoint, vector<bool>(5, false)));  
 mark[riPoint][i] = true;  
 }  
 }  
 riPoint = nextPoint(riPoint, d, -1);  
 }  
 mark[p][i] = true;  
 }  
 if (result.win5 >= 1 || result.alive4 >= 1 || result.dalive4 >= 2 || (result.dalive4 >= 1 && result.alive3 >= 1) || result.alive3 >= 2)  
 totalScore += 20000000;  
 else  
 totalScore += result.dalive4 \* 10000 + result.die4 \* 5000 + result.alive3 \* 10000 + result.dalive4 \* 1000 + result.die3 \* 500 + result.alive2 \* 1000 + result.dalive2 \* 100 + result.die2 \* 50 + result.alive1 \* 100 + result.dalive1 \* 10 + result.die1 \* 5;  
 return totalScore;  
}

α-β剪枝算法代码实现（部分）：

value = alpha; bestMove = get<0>(temp[0]);  
if (deep == endDeep - 1)  
 preValue = wholeValue();  
for (auto p : temp)  
{  
 if (value >= beta)   
 {  
 isbreak = true;  
 break;  
 }  
 if (isEmpty(get<0>(p).x, get<0>(p).y))  
 {  
 if (get<1>(p) >= 10000000)  
 {  
 bestMove = get<0>(p);  
 return { 10000000,false,false };  
 }  
 int nextEndDeep;  
 if (get<2>(p)&&endDeep<=10&&extension)   
 nextEndDeep = endDeep+2;  
 else  
 nextEndDeep = endDeep;  
 place(get<0>(p).x, get<0>(p).y, **COMPTER**);  
 if (zobristMap.find(zobristKey) != zobristMap.end())  
 {  
 responseValue = zobristMap[zobristKey];  
 number++;  
 }  
 else  
 {  
 tuple<long,bool,bool> reTuple;   
 if (deep != nextEndDeep - 1)  
 reTuple = findHumanMove(dc, deep + 1, nextEndDeep, value, beta, get<0>(p),preHumMove);  
 else  
 reTuple = { wholeValue(preValue,get<0>(p)),false,false };  
 if (get<2>(reTuple))  
 {  
 unPlace(get<0>(p).x, get<0>(p).y);  
 return reTuple;  
 }  
 responseValue = get<0>(reTuple);  
 if(!get<1>(reTuple))  
 zobristMap.insert(make\_pair(zobristKey, responseValue));  
 }  
 unPlace(get<0>(p).x, get<0>(p).y);   
 if (responseValue > value)   
 {  
 value = responseValue;  
 bestMove = { get<0>(p).x,get<0>(p).y };  
 }  
 if (responseValue >= 10000000)  
 return { value,false,false };  
 }  
}

# 实验数据及结果分析：

|  |  |  |
| --- | --- | --- |
| AI先手（程序内与人对战） | 与其他AI对局后手（网址：http://gobang2.light7.cn/  ）（普通） | 与其他AI对局后手（简单） |
|  |  |  |
| 中心区域  与人在中心区域的对决没有出现问题，先后手的落子速度相较α-β剪枝与启发式算法优化前提速明显，优化后前几步落子也十分迅速，棋力能战胜绝大部分没有系统钻研过五子棋的人（缺少系统学习过五子棋的五子棋高手样本） | 边界区域  程序内与人对战和在网页上对战AI（网页AI后手落子，人根据AI落子在程序内落子，模拟AI间的对局）程序都正常运行，估分函数能给出对应位置的分数 | 接近边界区域  模拟AI对决过程中程序正常运转，结合大量实战过程后可知，4层深度（程序输出的深度，真实深度与搜索时间相关）的AI棋力大概与网页AI简单难度相当  （PS：网页AI截图内容更为美观，故此处选择网页AI截图） |

对结果的分析：对战人和其他AI时AI程序都能正常运行，说明程序内估分函数的分数设置较为合理。

区分人与其他AI作为对战对手的必要性：

在之前的报告内容中提过，五子棋本质上是一种零和博弈，所以我们采用数值化并利用博弈树的方式找到最佳落子位置，但采取minmax的一大前提是程序符合消极博弈的要求，即对手总能找到于对手而言的最佳落子位置，那么对于并不总是落子于最佳位置的人，将其与其他AI区分对待进行比较是有必要的，实验结果说明对于以人为对手的对战，AI程序也能发挥实力。

# 八、总结及心得体会：

这次的实验结果不算尽善尽美，但也足以自慰，毕竟最终完成后可以与网页上大佬们的人机五子棋较量一二，我已经很欣慰了。

在A班的学习过程使人深深感觉到自己的不足，主要是身边的大佬太多了，在第三节课的时候就已经有大佬做完了所有的作业在学博弈树了，果然是比我厉害的人比我更努力，感觉自己只要稍稍松懈就会找不着北（事实上由于自身基础薄弱，有些课上的内容并不能好好消化，必须课后再去学习）。我在之前学的Java和C#的简陋知识完全派不上用场。

这学期的A班学习过程弥足珍贵，很多次我都对能和这么多编程大佬以及专业而有趣的老师坐在一起而感到惭愧不已，进而认识到自己的幸运。年少时期看到这么多耀眼的同龄人，他们不藏私，不骄傲，让彼此的思想碰撞出璀璨的焰火，进而使我这样的人心生向往，向前迈步。

学习新知识的阶段算比较惬意轻松的了，毕竟在实际复现进行代码实践之前，大可以自信地认为自己已经学会了（然后被代码实操狠狠打脸），这次实验中学到的剪枝算法和蒙特卡洛树搜索似乎与深度搜素和广度搜索有关，这些知识基础无疑会为我之后学习相关知识提供便利。

编写代码的经历痛苦居多，光是debug程序中不知何时产生的野指针就耗费了大量时间，甚至让人怀疑起选择计算机专业的决定是否正确，好在最后代码能跑（否则连实验数据和结果分析都做不了）。

这次完成AI五子棋代码的过程相当艰难，而且由于本人基础薄弱，不得不去网上观摩大佬们的开源代码，经常是读懂了代码后尝试复现结果完全不如源代码简洁明了，所以不少部分与我以往的代码风格完全不同（高中好友说的），我倒是觉得我的水平还谈不上什么代码风格之类的东西。

总的来说，这次实验设计过程使我学到了相当多的知识，也打磨了我的代码编写能力（实际仍有待提高）。这次完成实验内容的过程中多次烦扰各位大佬（在此感谢陈筑江大佬和陈彦儒大佬的无私帮助以及舍友的支援）希望下一次的实验设计能更独立地完成，希望我的代码能力能在之后的努力中不断提高。

# 对本实验过程及方法、手段的改进建议及展望：

对于实验过程，我觉得没有可以挑刺的部分，老师给足了学习的时间，各位大佬自我提升的效果也很显著，不过我的基础过差，在做完level1的作业后突然面对AI五子棋这种庞然大物还是觉得过于突然，难免有些准备不足，写代码的过程较为狼狈。

AI五子棋的程序编写感觉与课程内容的学习存在一部分割裂，毕竟学习编写AI五子棋的过程需要掌握的绘图函数使用、系统函数使用以及甚至要学习C++的部分内容等诸多前提知识与数据结构与算法没有过多联系，对于没接触过的人来说上手难度较大。

实验对于算法的运用相当充分，使我受益颇丰。

本次实验对已有他人函数的复现过多，自己缺少熟练运用各种系统函数和绘图函数实现功能的能力，深深感到自己有迫切的学习相关知识的需求，毕竟代码能力落后于他人就只能靠努力赶上。

过去锻炼自己代码编写能力一直是在力扣上做题，现在看来这种行为还是太过片面，有失偏颇。确实，一直做力扣题库里的题能让代码能力提高，但力扣题库内的题偏向学术与思考，而一个合格的编程者还需要用丰富的代码储存武装自己，缺少对代码的了解，只凭思考还是巧妇难为无米之炊，这次编写程序就多次在绘图函数和系统函数的使用上卡壳掉链子，希望以后能补上短板，提高自己的编程能力。

**报告评分：**

**指导教师签字：**