人工智能大作业报告——五子棋

计算机 1802 班 寇凯淇 20184446

一、目标功能：

实现一个五子棋博弈程序 ai(List_me, List_enemy,List_all)，返回落子坐标元组。单步限制 6 分钟以内。

二、实现过程：

1、准备阶段：

首先学习了解了五子棋的规则，学习棋谱。学习五子棋博弈程序所需博弈树、极小极大值算法、AlphaBeta 剪枝算法。

2、实现阶段

基于规则设定所需函数并编写完整实现最基础的落子博弈算法。在 runChess 架构里运行检查程序的正确性，保证能够正常运行。为算法补充决策树、极小极大值算法，提高决策时分析的合理性、广泛性。为博弈程序增加 AlphaBeta 剪枝算法，删除多余决策分支，提高决策效率。为各种棋局局势制定初步的权值，提高博弈程序对棋局的分析。

三、测试过程：

将编写好的五子棋博弈程序 ai 在 runChess 架构上运行，并于预置博弈函数和之前版本的博弈程序进行五子棋的博弈，分析博弈结果、花费时间并复盘整个棋局，分析博弈程序决策情况，特别是关键棋局局势的情况下，博弈程序做出的落子决策。并依据这些信息对五子棋博弈程序进行优化修改。

四、核心算法：

1、极小极大之算法（配合 AlphaBeta 剪枝）

(1) 算法思想：

通过递归的形式，建立决策树，在叶子节点或棋局结束时返回对该子节点的棋局评估值。通过 who 值判断落子方，通过 Alpha、Beta 判断剪枝选择，满足条件时执行剪枝操作，舍去其子节点及更深的递归查找。记录各个子节点的棋局评估值，在极小极大结点分别返回最小值或最大值。最终返回时选择综合情况对己方最优的选择。

(2) 伪代码:

```
def  AlphaBeta(dep, who, Alpha, Beta, MyList, EnList, HasList, AlList):
    If dep ==  最深深度  ||  游戏结束：
        return envalution(who, MyList, EnList, HasList, AlList)
    else:
        寻找落子点并保存至 lists 列表
        遍历列表 lists 中的结点
            If  己方落子：
                将落子点保存至己方落子信息的列表中
                Temp = AlphaBeta(dep+1, 2, Alpha, Beta, MyList, EnList, HasList, AlList)
```

恢复现场
If temp > Alpha:
Alpha = temp
if Alpha >= Beta:
剪枝 返回 Beta
else：
将落子点保存至敌方落子信息的列表中
Temp = AlphaBeta(dep+1, 1, Alpha, Beta, MyList, EnList, HasList, AlList)
恢复现场
If temp < Beta:
Beta = temp
if Alpha >= Beta:
剪枝 返回 Alpha
If 己方：
返回 Alpha
else：
返回 Beta

2、评估函数

(1) 算法思想：

扫描棋局，先判断整个棋局是否结束，即一方达成五子相连。若达成则直接返回权值最大的连五。若棋局仍在继续，则查找双方的落子信息，查找双方构成局势，依据一定的棋形计算双方棋局得分，并通过 who 判断下一子的落子方，计算进攻防守权值，计算当前棋局的权值得分，并返回。

(2) 伪代码：

def envalution(who, MyList, EnList, HasList, AlList):
If 己方获胜：
返回正权值
elif 敌方获胜：
返回负权值
else：
遍历己方落子信息
判断棋形 加分
遍历敌方落子信息
判断棋形 加分
当前棋局总分 = 己方分数 - 敌方分数
返回当前棋局权值

(3) 棋形权值：

连五：10000000；活四：180000；死四：40000；活三：30000；死三：5000；活二：10000；死二：10；其余：0。

五、重要优化报告：

v1 版本：博弈程序实现极小极大值算法，添加评估函数对棋局进行简单的胜负判断。

v2 版本：将查找可落子函数进行优化，对结点进行简单排序，优先查找靠近已有落子

的坐标位置。

v3 版本：增加 AlphaBeta 剪枝算法。

v4 版本：修改可落子函数检测循序，将原本靠近即可修改为由里及外，并限制查找范围为已落子坐标附近距离为 2 的坐标。

v5.0 版本：增加棋局棋形的权重，修改计算当前棋局得分方式，各个棋形得分累加。

v5.1 版本：修改评价函数计算当前棋局得分的方式，在计算己方得分的同时计算敌方棋形得分，当前棋局的分等于双方的分之差。

v5.2 版本：添加进攻防守权重 1.5。

v6 版本：修改评估函数查找顺序 从最新落子开始查找。

对局结果：先手胜，后手负（敌方活三棋形未发现形成活四）。

v7.1 版本：增加了时间计算功能并非限制；修改胜负判断函数，胜负判断只分析最后一手；更改部分权重：提高活三权重 降低死四权重 提高进攻防守权重为 2。

对局结果：先手胜 Time<248.01，后手负 Time<285.92（敌方活四未发现）。

v7.2 版本：更新了权重 提高了连五的权值 将其余棋局权重降低 提高活四权重占比。

对局结果：先手胜 Time:126.99，后手负 Time：291.78。

v7.2 版本：调整了活四的棋谱。

对局结果：先手负 Time:超时，后手负 Time：58.03。

v7.2 版本：调整了权重调高了活四、死四，降低活三。

对局结果：先手胜 Time:142.12，后手负 Time：超时。

v7.2 版本：调整了权重调高了活四、活三。

对局结果：先手胜 Time:162.26，后手负 Time：229.11。

v7.2 版本：提高了落子范围为 3 降低探索深度为 2 。

对局结果：先手胜，后手胜。

六、附件：

源程序

```
#20184446.py
#import time
size = 15 #棋盘大小
Serch_Depth = 2 #最大搜索深度
goals= [] #保存子节点的得分
LianWU = 10000000      #连五 *****
HuoSI = 180000       #活四 _****_
SiSI = 40000         #死四 #****_     *_***    **_**
HuoSAN = 30000       #活三 _***_      _*_**_
SiSAN = 5000         #死三 _***#      _*_**#
                      #死三 _**_*#     *__**
                      #死三 *_*_*      #_***_#
HuoER =  10000       #活二 __**__     _*_*_    _*__*_
SiER = 10            #死二 ___**#     __*_*#
                      #死二 _*__*#     *___*
El = 0                #其他
def ai(List_me, List_enemy, List_all):    #ai 博弈函数
    #time_start = time.time()
```

```
    global goals
    if len(List_enemy) == 0:    #若先手 7，7
        #print(7,7)
        #time_end = time.time()
        #print("Time:"+str(time_end-time_start))
        return (7,7)
    #保护现场
    MyList = List_me    #我方落子序列
    EnList = List_enemy #敌方落子序列
    AlList = List_all   #棋盘棋子序列
    HasList = MyList + EnList #已落子序列
    Alpha = float('-inf')
    Beta = float('inf')
    next_step = generator(HasList, List_all) #搜索可落子点
    dep =0  #第 0 层
    who = 1# 1 为 ai 2 为敌手
    Max = AlphaBeta(dep, 1, Alpha, Beta, MyList, EnList, HasList, AlLis
t) #调用算法
    i = goals.index(Max)
    #print(next_step[i])
    #time_end = time.time()
    #print("Time:"+str(time_end-time_start))
    return next_step[i]
#极大极小算法 有剪枝
def AlphaBeta(dep, who, Alpha, Beta, MyList, EnList, HasList, AlList):
    #返回当前结点权重
    global goals
    if if_win(MyList, EnList)!=0 or dep == Serch_Depth: #决策树叶子结点或
游戏结束
        return envalution(who, MyList, EnList, HasList, AlList) #返回权
重
    else:
        G = []  #保存子节点权重值
        G.clear()
        lists = generator(HasList, AlList)  #寻找子节点 即可落子的点
        for pos in lists:   #遍历可落子点
            if who == 1: #ai 落子     极大值点
                MyList.append(pos)  #ai 落子--->子节点
                HasList.append(pos)
                temp = AlphaBeta(dep+1, 2, Alpha, Beta, MyList, EnList,
 HasList, AlList)    #深度增加，敌手落子,返回子节点权值
                G.append(temp) #保存子节点权重为列表
                MyList.pop()    #恢复现场
                HasList.pop()
```

```
                if temp > Alpha:
                    Alpha = temp
                    if Alpha >= Beta:
                        return Beta     #剪枝
            else:   #敌手落子    极小值点
                EnList.append(pos)
                HasList.append(pos)
                temp = AlphaBeta(dep+1, 1, Alpha, Beta, MyList, EnList,
 HasList, AlList)    #深度增加，ai 落子，  返回子节点权值
                G.append(temp) #保存子节点权重为列表
                EnList.pop()
                HasList.pop()
                if temp < Beta:
                    Beta = temp
                    if Alpha >= Beta:
                        return Alpha    #剪枝
        #print(G)
        goals = G   #另存  最后退出保存的应是第一层的权值
        if who == 1: #ai 落子     极大值点
            return Alpha
        else:  #敌手落子     极小值点
            return Beta

def generator(HasList, AlList):     #搜索可以落子的点  当前为棋子邻近的点
    blank = list(set(AlList).difference(set(HasList)))
    lists = []
    for pos in blank:
        if is_neighbour(pos, HasList, 3):
            lists.append(pos)   #添加
    return lists    #返回可落子点列表

def is_neighbour(pos, HasList, dis):    #判断点是否为已落子的邻近点  dis 距
离
    for i in range(1, dis+1):
        if (pos[0],pos[1]-1*i) in HasList or (pos[0],pos[1]+1*i) in Has
List or (pos[0]-1*i,pos[1]) in HasList or (pos[0]+1*i,pos[1]) in HasLis
t or (pos[0]-1*i,pos[1]-1*i) in HasList or (pos[0]-1*i,pos[1]+1*i) in H
asList or (pos[0]+1*i,pos[1]-1*i) in HasList or (pos[0]+1*i,pos[1]+1*i)
 in HasList:
            return True     #是邻近点
        else:
            return False    #不是

def if_win(MyList, EnList):#判断胜负  1 ai 胜  2  敌手胜  0  继续
```

```
    for dis in range(8):
        if len(MyList) != 0:
            my_pos = MyList[-1]
            if getPoint(my_pos, dis, 1) in MyList and getPoint(my_pos,
dis, 2) in MyList and getPoint(my_pos, dis, 3) in MyList and getPoint(m
y_pos, dis, 4) in MyList:
                return 1
        if len(EnList) != 0:
            en_pos = EnList[-1]
            if getPoint(en_pos, dis, 1) in EnList and getPoint(en_pos,
dis, 2) in EnList and getPoint(en_pos, dis, 3) in EnList and getPoint(e
n_pos, dis, 4) in EnList:
                return 2
    return 0   #暂无胜负 棋局继续

def envalution(who, MyList, EnList, HasList, AlList):   #评价函数 这里的
who 指的是该出手的一方 最新一步是由对方下的
    Weight_AD = 2
    value_ai = 0   #ai 权值初始化
    value_enermy = 0    #敌手权值初始化
    blank = list(set(AlList).difference(set(HasList)))
    if if_win(MyList,EnList) == 1:    #ai 胜
        return LianWU
    elif if_win(MyList,EnList) == 2:  #敌手胜
        return -LianWU
    else:   #未结束
        for pos in reversed(MyList):  #计算 ai 权值
            for dis in range(8):
                t = 1
                if who == 1:#ai 先手
                    t = t * Weight_AD
                value_ai += Get(pos, MyList, EnList, AlList, blank, dis)
 * t

        for pos in reversed(EnList):  #计算敌手权值
            for dis in range(8):
                t = 1
                if who == 2:#敌手先手
                    t = t * Weight_AD
                value_enermy += Get(pos, EnList, MyList, AlList, blank,
 dis) * t
        return value_ai - value_enermy

m_offset = [(0,-1),(1,-1),(1,0),(1,1),(0,1),(-1,1),(-1,0),(-1,-1)]
```

```
def getPoint(pos, dis, offset):
    r = pos[0]
    c = pos[1]
    r = r + offset * m_offset[dis][1]
    c = c + offset * m_offset[dis][0]
    return (r,c)

def Get(pos, my_list, enermy_list, all_list, blank, dis):
    # 0 1 1110    活四 1
    if getPoint(pos, dis, -1) in blank and getPoint(pos, dis, 1) in my_
list and getPoint(pos, dis, 2) in my_list and getPoint(pos, dis, 3) in
my_list and getPoint(pos, dis, 4) in blank:
        return HuoSI/2
    #2 1 1110   死四 1
    if getPoint(pos, dis, -1) not in (my_list + blank) and getPoint(pos,
 dis, 1) in my_list and getPoint(pos, dis, 2) in my_list and getPoint(p
os, dis, 3) in my_list and getPoint(pos, dis, 4) in blank:
        return SiSI
    #1 0111     死四 2
    if getPoint(pos, dis, 1) in blank and getPoint(pos, dis, 2) in my_l
ist and getPoint(pos, dis, 3) in my_list and getPoint(pos, dis, 4) in m
y_list:
        return SiSI
    #1 1011     死四 3
    if getPoint(pos, dis, 1) in my_list and getPoint(pos, dis, 2) in bl
ank and getPoint(pos, dis, 3) in my_list and getPoint(pos, dis, 4) in m
y_list:
        return SiSI/2
    #0 1 110    活三 1
    if getPoint(pos, dis, -1) in blank  and getPoint(pos, dis, 1) in my
_list and getPoint(pos, dis, 2) in my_list and getPoint(pos, dis, 3) in
 blank:
        return HuoSAN/2
    #0 1 0110   活三 2
    if getPoint(pos, dis, -1) in blank  and getPoint(pos, dis, 1) in bl
ank and getPoint(pos, dis, 2) in my_list and getPoint(pos, dis, 3) in m
y_list and getPoint(pos, dis, 4) in blank:
        return HuoSAN
    #0 1 112    死三 1
    if getPoint(pos, dis, -1) in blank  and getPoint(pos, dis, 1) in my
_list and getPoint(pos, dis, 2) in my_list and getPoint(pos, dis, 3) no
t in (my_list + blank):
        return SiSAN
    #0 1 0112   死三 2
```

```
    if getPoint(pos, dis, -1) in blank  and getPoint(pos, dis, 1) in bl
ank and getPoint(pos, dis, 2) in my_list and getPoint(pos, dis, 3) in m
y_list and getPoint(pos, dis, 4) not in (my_list + blank):
        return SiSAN
    #0 1 1012   死三 3
    if getPoint(pos, dis, -1) in blank  and getPoint(pos, dis, 1) in my
_list and getPoint(pos, dis, 2) in blank and getPoint(pos, dis, 3) in m
y_list and getPoint(pos, dis, 4) not in (my_list + blank):
        return SiSAN
    # 1 0011    死三 4
    if getPoint(pos, dis, 1) in blank and getPoint(pos, dis, 2) in blan
k and getPoint(pos, dis, 3) in my_list and getPoint(pos, dis, 4) in my_
list:
        return SiSAN
    # 1 0101    死三 5
    if getPoint(pos, dis, 1) in blank and getPoint(pos, dis, 2) in my_l
ist and getPoint(pos, dis, 3) in blank and getPoint(pos, dis, 4) in my_
list:
        return SiSAN/2
    #20 1 1102  死三 6
    if getPoint(pos, dis, -2) not in (my_list + blank) and getPoint(pos,
 dis, -1) in blank and getPoint(pos, dis, 1) in my_list and getPoint(po
s, dis, 2) in my_list and getPoint(pos, dis, 3) in blank and getPoint(p
os, dis, 4) not in (my_list + blank):
        return SiSAN/2
    #00 1 100   活二 1
    if getPoint(pos, dis, -2) in blank and getPoint(pos, dis, -1) in bl
ank and getPoint(pos, dis, 1) in my_list and getPoint(pos, dis, 2) in b
lank and getPoint(pos, dis, 3) in blank:
        return HuoER/2
    #0 1 010    活二 2
    if getPoint(pos, dis, -1) in blank  and getPoint(pos, dis, 1) in bl
ank and getPoint(pos, dis, 2) in my_list and getPoint(pos, dis, 3) in b
lank:
        return HuoER/2
    #0 1 0010   活二 3
    if getPoint(pos, dis, -1) in blank  and getPoint(pos, dis, 1) in bl
ank and getPoint(pos, dis, 2) in blank and getPoint(pos, dis, 3) in my_
list and getPoint(pos, dis, 4) in blank:
        return HuoER/2
    #2 1 1000   死二 1
    if getPoint(pos, dis, -1) not in (my_list + blank)  and getPoint(po
s, dis, 1) in my_list and getPoint(pos, dis, 2) in blank and getPoint(p
os, dis, 3) in blank and getPoint(pos, dis, 4) in blank:
```

```
        return SiER
    #2 1 0100   死二 2
    if getPoint(pos, dis, -1) not in (my_list + blank)  and getPoint(po
s, dis, 1) in blank and getPoint(pos, dis, 2) in my_list and getPoint(p
os, dis, 3) in blank and getPoint(pos, dis, 4) in blank:
        return SiER
    #2 1 0010   死二 3
    if getPoint(pos, dis, -1) not in (my_list + blank)  and getPoint(po
s, dis, 1) in blank and getPoint(pos, dis, 2) in blank and getPoint(pos,
 dis, 3) in my_list and getPoint(pos, dis, 4) in blank:
        return SiER
    # 1 0001    死二 4
    if getPoint(pos, dis, 1) in blank and getPoint(pos, dis, 2) in blan
k and getPoint(pos, dis, 3) in blank and getPoint(pos, dis, 4) in my_li
st:
        return SiER/2
    return 0
```