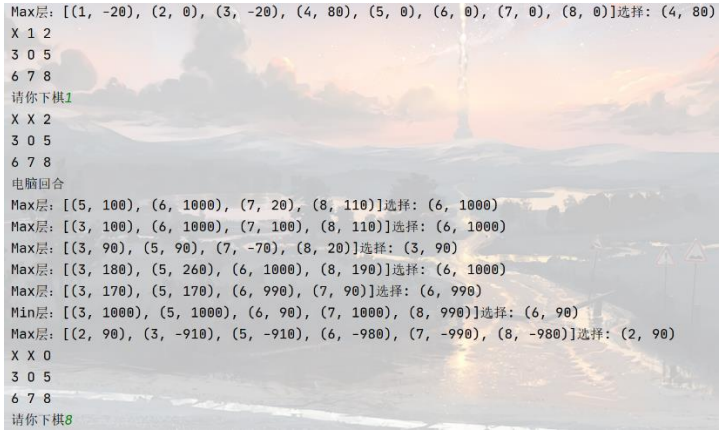
对抗博弈问题报告

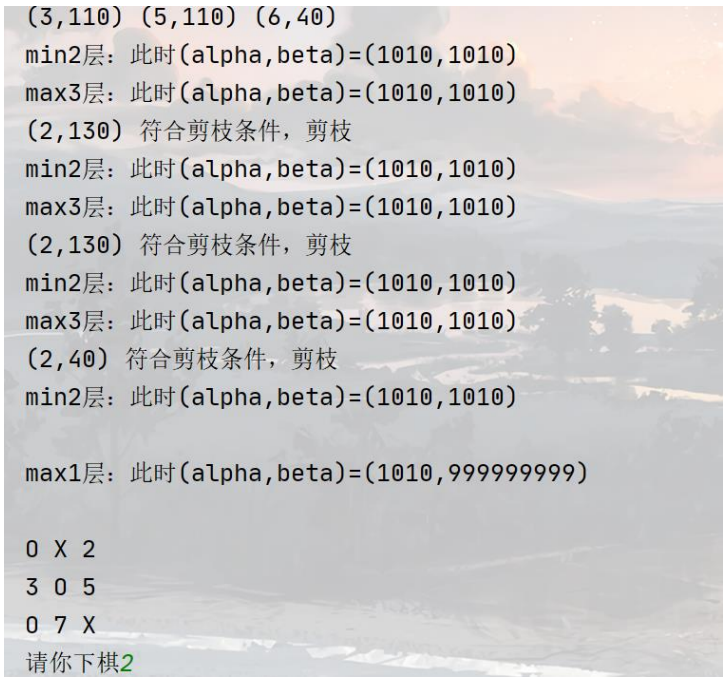
1.项目说明

本次项目是五子棋的人机博弈游戏，基于极大极小值的 alpha_beta 剪枝来完成。

项目一开始是通过写井字棋开始的，因为井字棋的情况比较少，适合发现算法的问题，一开始也没有做界面的优化，最开始也没有进行剪枝，而是采用了单纯的遍历极大极小值来完成，以此来搭建初步的框架，游戏成果如下：



再然后对程序进行剪枝优化，加入 alpha_beta 剪枝，同时还是在井字棋上面进行，对之中发现的问题进行解决，游戏成果如下：

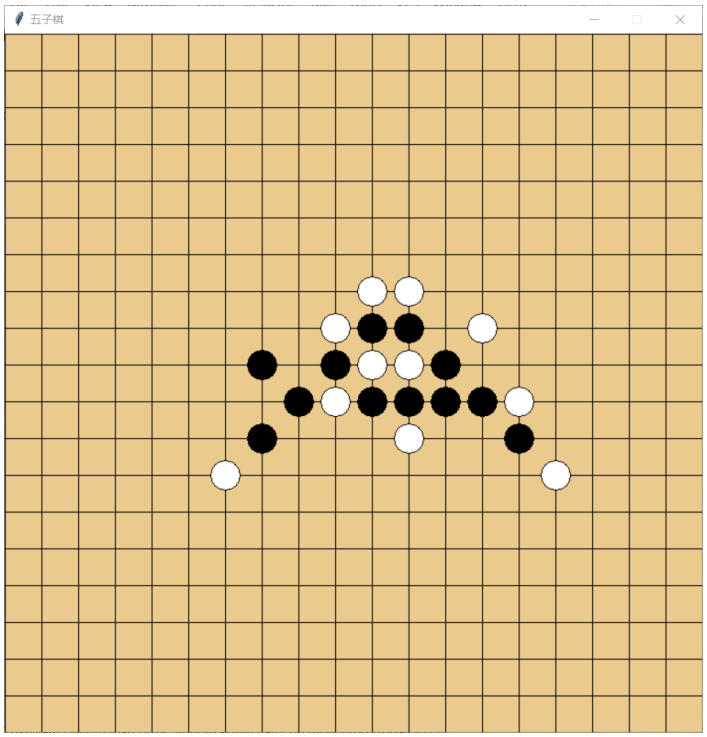


然后就又在此基础上进行五子棋的游戏开发，主要需要替换的是搜索范围的

确定和预测分数的函数还有判断游戏结束的函数以及一些细节，完成之后的结果如下：

```
140 141 142 143 144 145 146 147 148 149 150 151 0   153 154 155 156 157 158 159
160 161 162 163 164 165 166 X   168 169 170 X   172 173 174 175 176 177 178 179
180 181 182 183 184 185 186 0   0   X   X   191 192 193 194 195 196 197 198 199
200 201 202 203 204 205 206 0   X   X   0   0   212 213 214 215 216 217 218 219
220 221 222 223 224 225 226 0   X   229 0   231 232 233 234 235 236 237 238 239
240 241 242 243 244 245 246 0   X   0   0   251 252 253 254 255 256 257 258 259
260 261 262 263 264 265 266 X   0   X   0   271 272 273 274 275 276 277 278 279
280 281 282 283 284 285 286 0   X   289 X   291 292 293 294 295 296 297 298 299
300 301 302 303 304 305 X   307 308 309 310 X   312 313 314 315 316 317 318 319
320 321 322 323 324 325 326 327 328 329 330 331 0   333 334 335 336 337 338 339
340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379
380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399
电脑获胜！
```

然后对游戏的 UI 界面进行了优化，这方面借鉴了网上的一些代码，链接如下：https://github.com/colingogogo/gobang_AI。然后自己对界面进行了改造，增加了开局选择的界面，完成后的界面如下：



再然后就对核心的分数判断函数进行优化，因为一开始使用的用 5*5 的矩阵从左上到右下进行遍历棋局，然后通过分别对横竖还有对角的值进行相加，按照相加和进行判断分类，但是这样子过于粗糙，而且速度也慢，对于 2 层深度的遍历，前期两三秒才能出结果，后期就需要六七秒了，体验很不好。之后对其进行了优化，通过判断行列和对角的列表的值进行判断和空相接的黑子和白子一共有多少进行计数，然后依据结果进行赋值，这样不仅更准确而且速度也更快了，粗略地看来这样子 3 层的时间相当于之前用矩阵的 2 层的时间，而 2 层的遍历基本能在 2 秒内出结果，体验较好。后续还可以对分数进行继续的优化，但限于时间和精力有限就止步于此了。

2.算法说明

极大极小值算法原理:

将搜索数的层数分为 Max 层和 Min 层，Max 层是 ai 下棋，会向下选择对 ai 最好的情况，Min 层是预测人下棋，会尽量选择对 ai 最不利的情况，最后 ai 就是在 Min 中寻找最大值，如果层数够多也就是逐步往下遍历，到达最底层之后，就会对前面下的棋最后的棋局进行评分，进行遍历所有可能的情况，Min 最后选的是最小的值，Max 选的是最大的值，然后就继续往上传递。

alpha beta 剪枝算法原理

极大极小值搜索算法的缺点就是当博弈树的层数变大时，需要搜索的节点数目会指数级增长。比如每一层的节点为 50 时，六层博弈树的节点就是 50 的 6 次方，运算时间会非常漫长，而这在运算过程中有很多是不需要进行预测的，Alpha-Beta 剪枝就是用来将搜索树中不需要搜索的分支裁剪掉，以提高运算速度。基本的原理是：当一个 MIN 层节点的 α 值 ≤ β 值时，剪掉该节点的所有未搜索子节点当一个 MAX 层节点的 α 值 ≥ β 值时，剪掉该节点的所有未搜索子节点其中 α 值是该层节点当前最有利的评分，β 值是父节点当前的 α 值，根节点因为是 MAX 层，所以 β 值 初始化为正无穷大(+∞)。

初始化节点的 α 值，如果是 MAX 层，初始化 α 值为负无穷大(-∞)，这样子节点的评分肯定比这个值大。如果是 MIN 层，初始化 α 值为正无穷大(+∞)，这样子节点的评分肯定比这个值小。

例如一开始进行深度遍历的时候，到达了最底层的 Min 层，这时就需要对其最后的情况进行遍历，得到不同的值，然后选取最小的值，回去一层，将这个值为 α 进行赋值，，代表后面遍历时查找的范围是（α，+∞），因为如果小于 α 的话，因为这里是选取 Max，所以是不会选取比 α 小的数据的，再继续，就又是一个 Min 层，如果对节点的评分进行预测，如果它的值比 α 小的话，那么就说明这一个 Min 节点最后会选择一个不大于 α 的值，那么对于再往上的 Max 层，它就不会选择这个 Min 的值，也就不需要继续往下判断了，这时候就可以进行剪枝。但是如果最后的值都没有小于 α 的话，那么就说明回到 Max 之后，它会选择这一个节点的值，所以这时候就需要把 α 更新为这个节点的值。以此类推，就可以完成这个算法。

棋局评估算法

这里先后使用了 2 个棋局评估算法，一个是 5*5 的矩阵为单位的评估算法，一个是以一列为单位的评估算法。

矩阵评估

因为棋局是 20*20 的大小，所以需要从（0,0）开始，遍历到（15,15），每次以这个坐标为最左上角的点来形成一个 5*5 的矩阵，然后通过矩阵相乘来提取出 5 行 5 列和 2 个对角线，然后通过计算每个的和，通过和的值来进行分类，对分数进行赋值。算法如下：

```
for i in range(length-5):#行
    for k in range(0,length-5,5):
        board_matri = geta_matri(checkerboard, k*length+i)#得到一个矩阵
        for j in range(5):
            temp = matrix_mul(board_matri, eva_matri[j])
            sc += score_dic.get(temp)
            sc += good_score(board_matri)
            if temp == 5:
                is_win = True
                return sc,is_win
for i in range(0,length-5,5):#列
    for k in range(length-5):
        board_matri = geta_matri(checkerboard, k*length+i)
        for j in range(5,10):
            temp = matrix_mul(board_matri, eva_matri[j])
            sc += score_dic.get(temp)
            sc += good_score(board_matri)
            if temp == 5:
                is_win = True
                return sc,is_win
```

列评估

对于一个 20*20 的棋盘，可以提取出 20 行 20 列以及 31 个从左上到右下、31 个从左上到右下的长度大于等于 5 的数据。然后都转化为一行的列表。对这一个列表进行求和，步骤如下：先是从左往右遍历，找到不为 0 的值，即有棋子，计数+1，然后继续往右看，如果还是同一种棋子，就计数继续+1，如果后面遇到另一种棋子，就说明这里被堵了，所以就结束计数，还需要判断是不是这时是 5 个棋子，如果是 5 个就说明游戏结束了，如果不是 5 个，就对计数-1，因为它的一边已经有了对方的一个棋子了，这时这个棋子的评分就需要降低了，如果后面遇到的是 0，即空，那么就再往后看一个，如果这个还是空或者是对方的棋子，那么就结束计数，并且进行赋值，如果再往后的那个是自己方的棋子，那么就不用结束，反而继续往后看，不过这个空格这里的计数不+1。遍历这一个列表结束之后，依据存储的值进行评分赋值。算法如下：

```
while j < len(checkerboard):
    if checkerboard[ j] == 0:#前面的0
        is_sum = True
        j = j + 1
    else:
        if is_sum:#遇到不是0
            is_sum = False
            temp = checkerboard[j]
            s = 0
            kong=0
            if j+1 <len(checkerboard):#小心j+1超出范围
                while (checkerboard[j] == temp) or (checkerboard[j] == 0 and checkerboard[j+1]==temp and kong==0):
                    if checkerboard[j] == temp:
                        s = s + 1
                    if(checkerboard[j] == 0 and checkerboard[j+1]==temp):#只连续统计一次带空格的，例如：1 0 1 1 1 1
                        kong = 1
                    j = j + 1
                    if j+1>=len(checkerboard):
                        break
            if j <len(checkerboard):
                if checkerboard[j] ==-1*temp and s!=5:#如果连续的旁边有对方的子，数就-1，但是如果有5个了，就不减
```

```python
def evaluate(checkerboard):#对棋局进行评分
    is_fin=False
    score_dic = {0: 0, 1: 10, 2: 100 , 3: 2000  , 4: 30000 ,
                 5: 2000000, -1: -5, -2: -50,
                 -3: -1000, -4: -10000, -5: -1000000}
    score=0
    for i in range(ROW):#一行一行算分
        t=get_row(i,checkerboard)
        t_sum=get_lianxu(t)
        score+=score_dic[t_sum[1]]
        score+=score_dic[-t_sum[-1]]
        if(t_sum[1]==5 or t_sum[-1]==5):#如果5个连起来就算游戏结束
            is_fin=True
            return score,is_fin
    #其余的列和对角线都相同
    for i in range(COLUMN):
        t=get_col(i,checkerboard)
        t_sum=get_lianxu(t)
        score+=score_dic[t_sum[1]]
        score+=score_dic[-t_sum[-1]]
```

3.函数说明

geta_matri(checkerboard, star_n, n=1)
得到一个从 star_n 开始的 5*5 的矩阵

matrix_mul(board_matri, eva_matri)
矩阵对应元素相乘

evaluate2(checkerboard)
通过矩阵评估对分数进行计算

get_lianxu(checkerboard)
得到一行有多少个与 0 接触的连续的 1、-1

get_row(n, checkerboard)
得到第 n 行

get_col(n, checkerboard)
得到第 n 列

get_left_top(n, checkerboard)
得到第 n 个从左上到右下的列表

get_right_top(n, checkerboard)
得到第 n 个从右上到左下

evaluate(checkerboard)
通过列评估对棋局进行评分

get_map(checkerboard)
得到现在有的棋子的旁边一格的所有棋子，当做需要进行判断的棋

calculus(is_ai,checkerboard,flag,depth,alpha=-999999999999,beta=999999999999)
alph_beta 进行剪枝运算

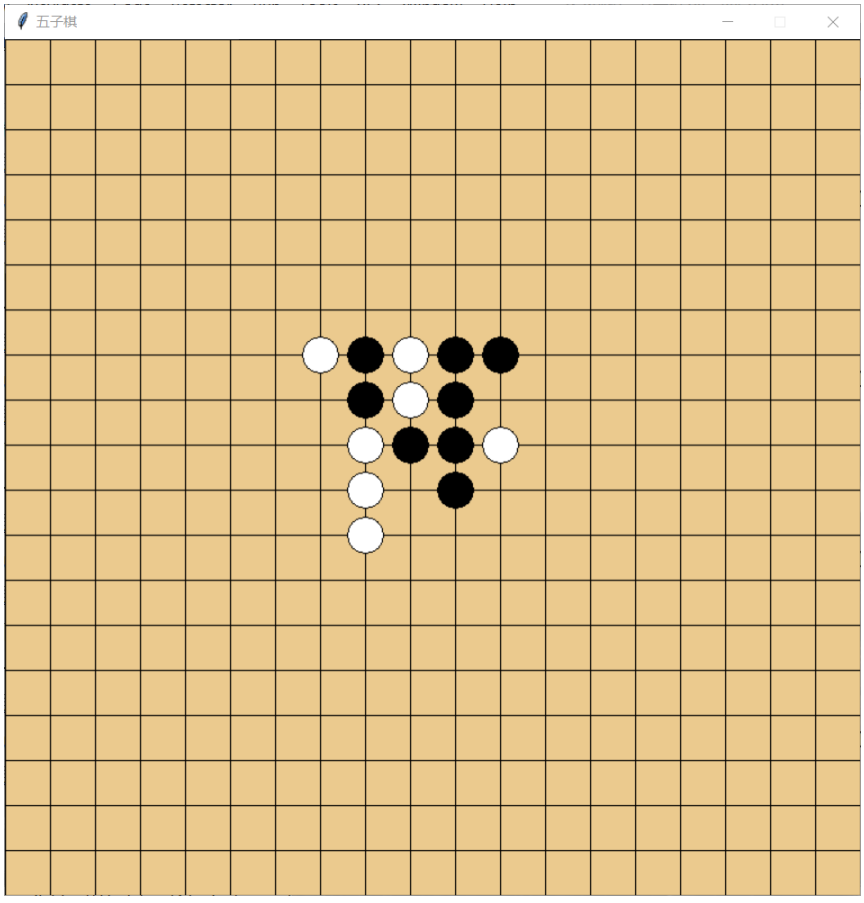
isGameOver(checkerboard)
通过是否有 5 个连在一起的判断棋是否结束

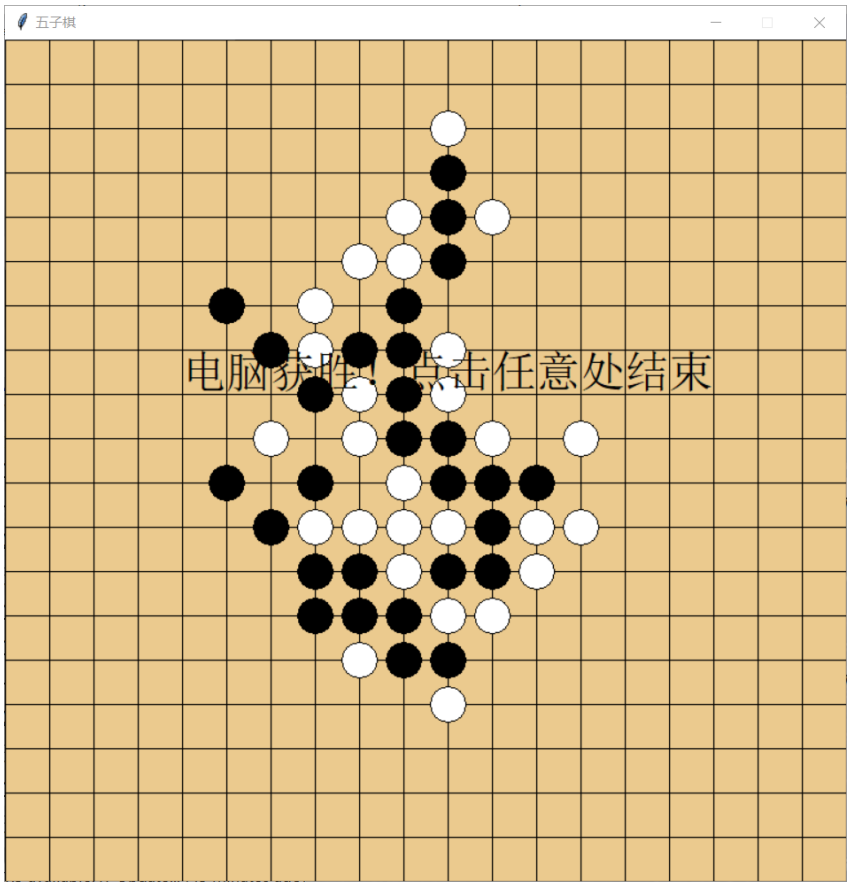
gamewin()
绘制棋盘

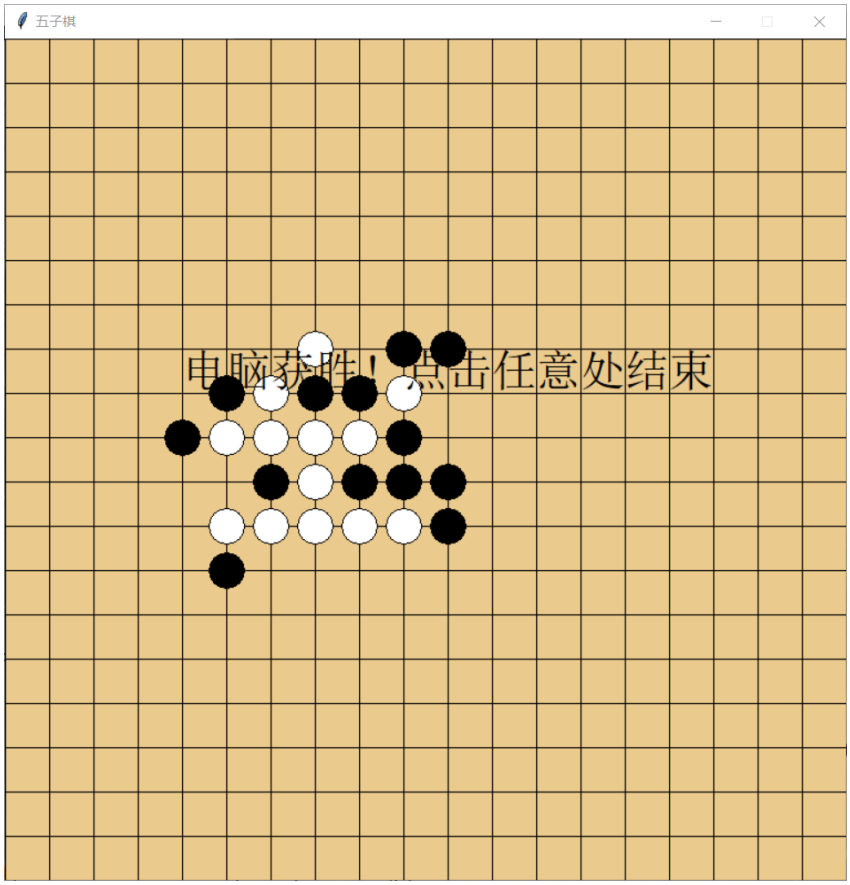
star_win()
绘制开始界面

game()
游戏进行

4.结果展示

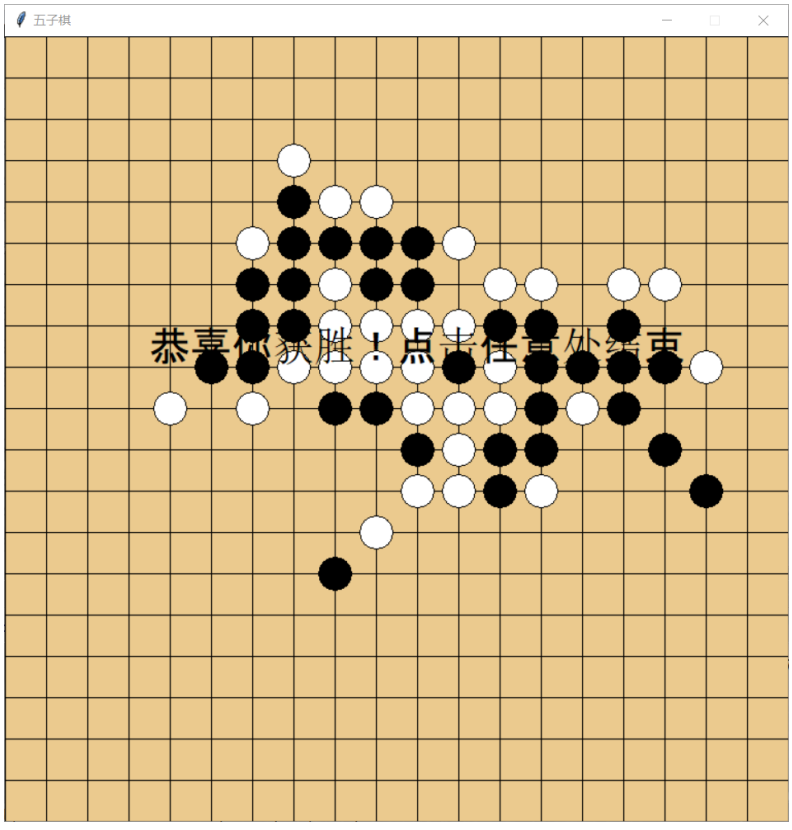


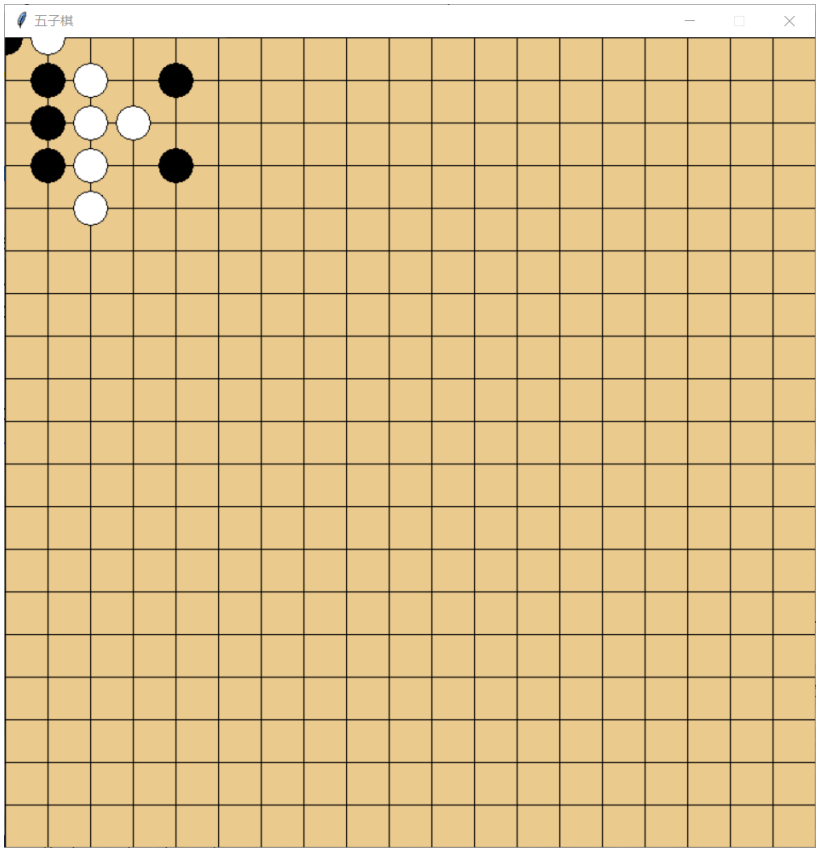

五子棋
电脑获胜！点击任意处结束

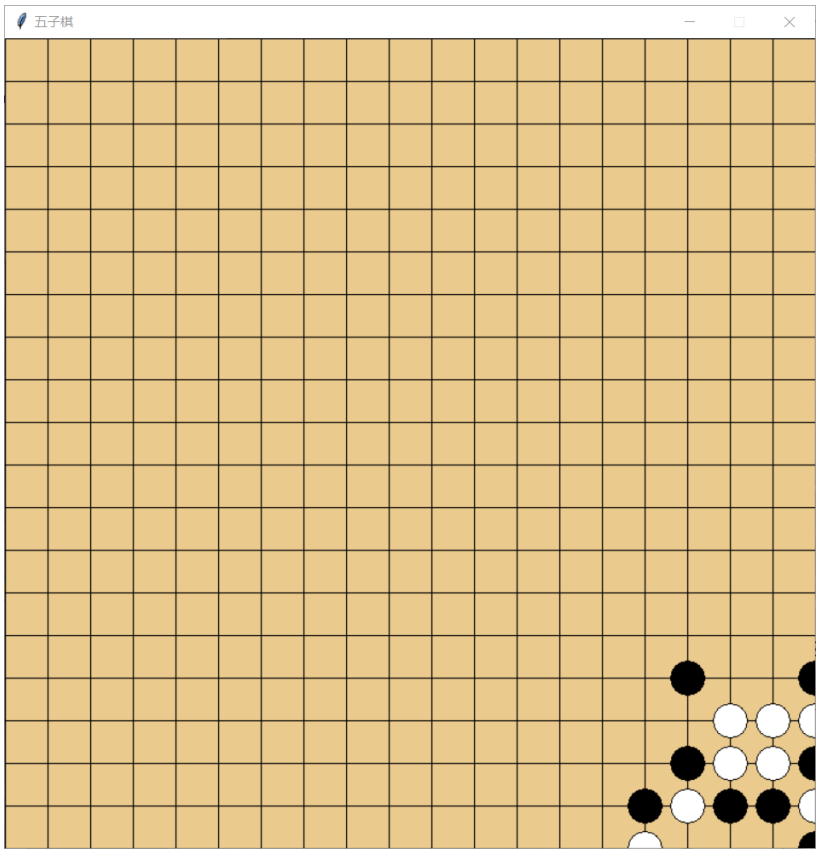

五子棋
电脑获胜！点击任意处结束

```
(下子点：29,得分：2430) 符合条件，更新alpha,beta为 (2280, 2430) (下子点：30,得分：2470) (下子点：49,得分：2390) 符合条件，更新alpha,b
max1层：此时(alpha,beta)=(2280,999999999999),最优下子点为104
max1下子点：124
50 - -21 50 - -20 50 - -1 50 - 1 50 - 20 50 - 19 50 - 21 50 - -19 70 - -21 70 - -20 70 - -1 70 - 1 70 - 20 70 - 19 70 - 2
min2层：此时(alpha,beta)=(2280,999999999999),最优下子点为-1
(下子点：29,得分：2430) 符合条件，更新alpha,beta为 (2280, 2430) (下子点：30,得分：2470) (下子点：49,得分：2390) 符合条件，更新alpha,b
max1层：此时(alpha,beta)=(2280,999999999999),最优下子点为104
max1下子点：126
50 - -21 50 - -20 50 - -1 50 - 1 50 - 20 50 - 19 50 - 21 50 - -19 70 - -21 70 - -20 70 - -1 70 - 1 70 - 20 70 - 19 70 - 2
min2层：此时(alpha,beta)=(2280,999999999999),最优下子点为-1
(下子点：29,得分：2420) 符合条件，更新alpha,beta为 (2280, 2420) (下子点：30,得分：2460) (下子点：49,得分：2380) 符合条件，更新alpha,b
max1层：此时(alpha,beta)=(2280,999999999999),最优下子点为104
max1下子点：145
50 - -21 50 - -20 50 - -1 50 - 1 50 - 20 50 - 19 50 - 21 50 - -19 70 - -21 70 - -20 70 - -1 70 - 1 70 - 20 70 - 19 70 - 2
min2层：此时(alpha,beta)=(2280,999999999999),最优下子点为-1
(下子点：29,得分：2430) 符合条件，更新alpha,beta为 (2280, 2430) (下子点：30,得分：2470) (下子点：49,得分：2390) 符合条件，更新alpha,b
max1层：此时(alpha,beta)=(2280,999999999999),最优下子点为104
max1下子点：144
50 - -21 50 - -20 50 - -1 50 - 1 50 - 20 50 - 19 50 - 21 50 - -19 70 - -21 70 - -20 70 - -1 70 - 1 70 - 20 70 - 19 70 - 2
min2层：此时(alpha,beta)=(2280,999999999999),最优下子点为-1
```

```
max1下子点：171
50 - -21 50 - -20 50 - -1 50 - 1 50 - 20 50 - 19 50 - 21 50 - -19 70 - -21 70 - -20 70 - -1 70 - 1 70 - 20 70 - 19 70 - 2
min2层：此时(alpha,beta)=(2280,999999999999),最优下子点为-1
(下子点：29,得分：2320) 符合条件，更新alpha,beta为 (2280, 2320) (下子点：30,得分：2360) (下子点：49,得分：2280) 符合条件，更新alpha,b
max1层：此时(alpha,beta)=(2280,999999999999),最优下子点为104
max1下子点：187
50 - -21 50 - -20 50 - -1 50 - 1 50 - 20 50 - 19 50 - 21 50 - -19 70 - -21 70 - -20 70 - -1 70 - 1 70 - 20 70 - 19 70 - 2
min2层：此时(alpha,beta)=(2280,999999999999),最优下子点为-1
(下子点：29,得分：2320) 符合条件，更新alpha,beta为 (2280, 2320) (下子点：30,得分：2360) (下子点：49,得分：2280) 符合条件，更新alpha,b
max1层：此时(alpha,beta)=(2280,999999999999),最优下子点为104
max1下子点：185
50 - -21 50 - -20 50 - -1 50 - 1 50 - 20 50 - 19 50 - 21 50 - -19 70 - -21 70 - -20 70 - -1 70 - 1 70 - 20 70 - 19 70 - 2
min2层：此时(alpha,beta)=(2280,999999999999),最优下子点为-1
(下子点：29,得分：2320) 符合条件，更新alpha,beta为 (2280, 2320) (下子点：30,得分：2360) (下子点：49,得分：2280) 符合条件，更新alpha,b
max1层：此时(alpha,beta)=(2280,999999999999),最优下子点为104
max1下子点：206
50 - -21 50 - -20 50 - -1 50 - 1 50 - 20 50 - 19 50 - 21 50 - -19 70 - -21 70 - -20 70 - -1 70 - 1 70 - 20 70 - 19 70 - 2
min2层：此时(alpha,beta)=(2280,999999999999),最优下子点为-1
(下子点：29,得分：2325) 符合条件，更新alpha,beta为 (2280, 2325) (下子点：30,得分：2365) (下子点：49,得分：2285) 符合条件，更新alpha,b
max1层：此时(alpha,beta)=(2280,999999999999),最优下子点为104
(下子点：205,得分：2000260) 此时游戏结束，不往下预测
```




五子棋

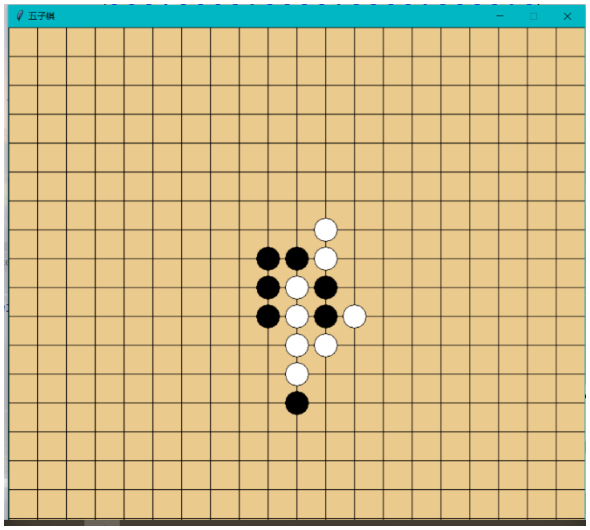

五子棋

5.遇到的问题及解决

1.

问题：

在我用矩阵评分判断的时候有时会遇到如下的情况，右下角的白色是我已经形成了一个 3 连子，但是 ai 却没去去堵我，而是选择自己在左边再去形成一个 3 个连子的矩阵。



问题原因：

后面在 debug 之后发现的问题是，因为是 5*5 的矩阵一个一个地去遍历，所以这时往下预测的时候，如果它在左边形成了一个三连子，那么这个 3 连子的值就会算 5*3 遍，因为有这么多个矩阵会套进它，然后白子最好的做法是形成右下角的 4 连子，这个 4 连子就之后被统计到 2 次，而我当时的评分是自己 3 连子 1000，对方 4 连子-10000，这样算出的分数是-5000，而如果黑子去堵白子的三连子，白子继续形成 4 连子，那么只会被统计到一次，形成的分数是-10000，所以最后在 Max 层，它会选择-5000，也就是自己去形成一个 3 连子。

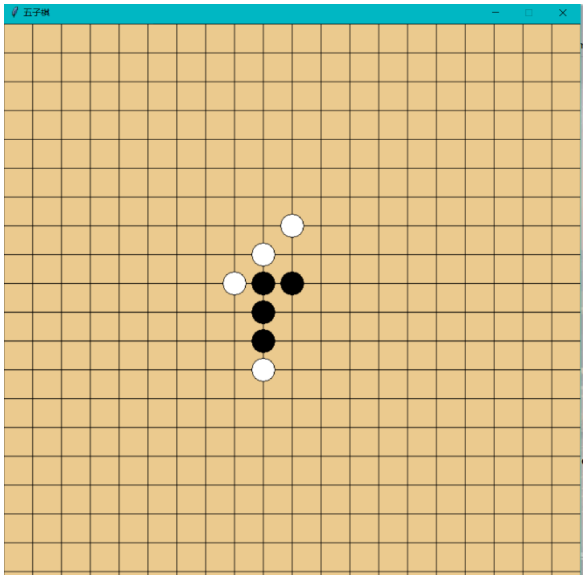
解决方法：

矩阵法对于横竖没什么区别，但是对于对角的情况会严重误判，所以还是太粗糙了，我们需要让横竖、对角都只能算一遍，这样才能够公平地计分。于是后面就想到了一列一列地计数，这样就不会出现重复计分的情况了，也就是前面提到的列评分方法。

2.

问题：

白子是 ai，在自己在左上角能形成 4 连时，ai 却选择去阻拦对方黑子在下面形成 4 连子



问题原因：如果 ai 的白子去连成 4 个，那么黑子就去给自己连成 4 个，这样子 2 个 4 个是有区别的，但是因为当时是单纯的计算连起来的子的个数所以 2 者不能区分开来。反而是靠各种其他的细节进行判断。

解决方法：对于这种连子如果后面接的是其他子就对它的计数-1，这样与两边都是空的区分开来。

解决方法：对于这种连子如果后面接的是其他子就对它的计数-1，这样与两边都是空的区分开来。

还有许多问题不再一一列举了。

6.心得

本实验整体来说还是有一点难度的，主要问题在于极大极小值算法、alpha_beta 剪枝算法的实现，但是如果从简单的井字棋开始设计，就会比较好对其进行 debug，找到算法的问题所在。而后续转化为五子棋的时候就会比较简单。还有一个难点是评估函数，评估函数是可变性最大的，也是决定算法的速度还有能力的一个重要影响，这种还需要对五子棋有一定的理解才比较好写出。我写的是比较粗糙的，后续还可以针对棋局的特殊布局来进行优化。