中山大学计算机学院

人工智能

本科生实验报告

（2022学年春季学期）

课程名称：Artificial Intelligence

教学班级	信计系统结构方向	专业（方向）	ICS
学号	20337268	姓名	张文沁

一、 实验题目

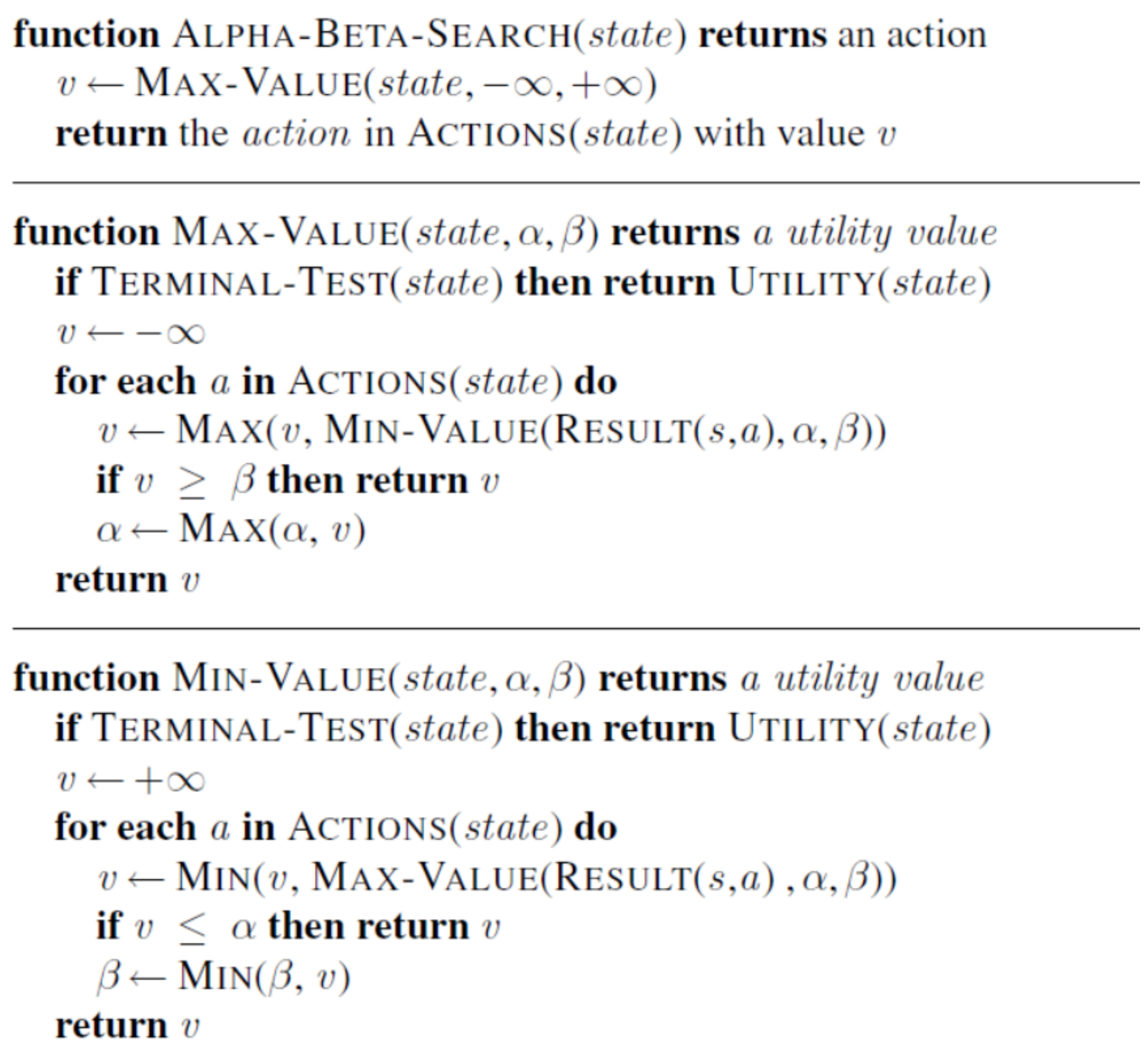
> **实现8*8黑白翻转棋的人机对战**
>
> **要求使用alpha-beta剪枝；**
>
> **1. 不要求实现UI；**
>
> **2. 搜索深度和评价函数不限，自己设计。在报告中说明清楚自己的评价函数及搜索策略。**
>
> **3. 鼓励大家结合高级搜索算法优化评价函数。**
>
> **4. 实验结果要求展示至少连续三个回合（人和机器各落子一次指一回合）的棋局分布情况，并输出每步落子的得分。**

二、 实验内容

1. 算法原理

> **Alpha-beta剪枝算法：**
>
> **通过函数参数的形式递归传递如下内容：**
>
> **1. 祖先Max节点中最大的alpha值，以及祖先Min节点中最小的beta值**
>
> **2. 节点维护“效益值”，Max节点更新上述alpha值，Min节点更新上述beta值**
>
> **3. Max节点的alpha剪枝：效益值≥ 祖先Min节点的最小beta值**

4. Min节点的beta剪枝：效益值≤ 祖先Max节点的最大alpha值Max节点的alpha值≥ 其任一祖先Min节点的beta值Min节点的beta值≤ 其任一祖先Max节点的alpha值



```
function ALPHA-BETA-SEARCH($state$) returns an action
    $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
    return the $action$ in ACTIONS($state$) with value $v$
```

```
function MAX-VALUE($state, \alpha, \beta$) returns $a$ $utility$ $value$
    if TERMINAL-TEST($state$) then return UTILITY($state$)
    $v \leftarrow -\infty$
    for each $a$ in ACTIONS($state$) do
        $v \leftarrow$ MAX($v$, MIN-VALUE(RESULT($s,a$), $\alpha, \beta$))
        if $v \geq \beta$ then return $v$
        $\alpha \leftarrow$ MAX($\alpha, v$)
    return $v$
```

```
function MIN-VALUE($state, \alpha, \beta$) returns $a$ $utility$ $value$
    if TERMINAL-TEST($state$) then return UTILITY($state$)
    $v \leftarrow +\infty$
    for each $a$ in ACTIONS($state$) do
        $v \leftarrow$ MIN($v$, MAX-VALUE(RESULT($s,a$) , $\alpha, \beta$))
        if $v \leq \alpha$ then return $v$
        $\beta \leftarrow$ MIN($\beta, v$)
    return $v$
```

评价函数

•在实际问题中，状态数量可能非常庞大。

•扩展到终止节点的代价是无法容忍的。

•此时，我们需要限制搜索深度。

•在有深度限制时，原来的内部节点会充当“叶子节点”的作用。

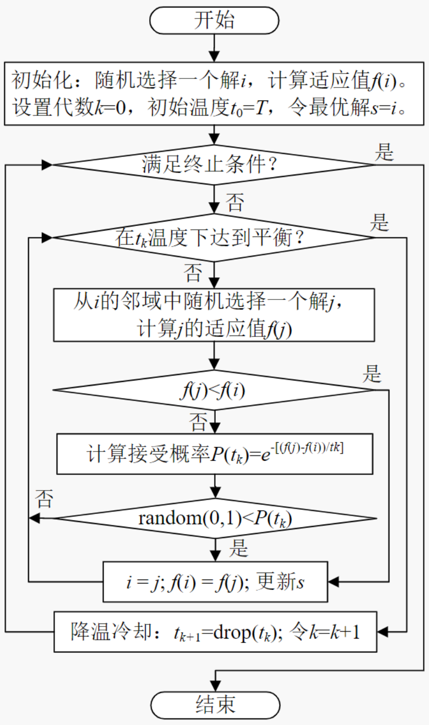
如何得到这些内部节点的“效益值”？

•我们使用“评价函数”，对节点进行优劣评分。

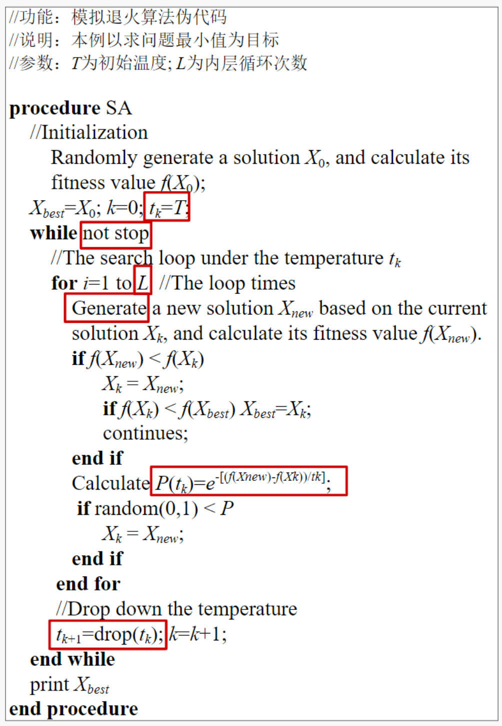
•此时以评价函数值作为节点分值的估计。

模拟退火算法：

1 模拟退火算法



开始
初始化：随机选择一个解i，计算适应值$f(i)$。设置代数k=0，初始温度t_0=T，令最优解s=i。
满足终止条件？
是
否
在t_k温度下达到平衡？
是
否
从i的邻域中随机选择一个解j，计算j的适应值$f(j)$
$f(j)<f(i)$
是
否
计算接受概率$P(t_k)=e^{-[(f(j)-f(i))/tk]}$
否
random(0,1)<$P(t_k)$
是
$i=j$; $f(i)=f(j)$; 更新s
降温冷却：t_{k+1}=drop(t_k); 令k=k+1
结束



```
//功能：模拟退火算法伪代码
//说明：本例以求问题最小值为目标
//参数：T为初始温度; L为内层循环次数

procedure SA
  //Initialization
    Randomly generate a solution $X_0$, and calculate its
    fitness value $f(X_0)$;
  $X_{best}=X_0$; $k=0$; $t_k=T$;
  while not stop
    //The search loop under the temperature $t_k$
    for $i$=1 to $L$  //The loop times
      Generate a new solution $X_{new}$ based on the current
      solution $X_k$, and calculate its fitness value $f(X_{new})$.
      if $f(X_{new}) < f(X_k)$
          $X_k = X_{new}$;
          if $f(X_k) < f(X_{best})$ $X_{best}=X_k$;
          continues;
      end if
      Calculate $P(t_k)=e^{-[(f(Xnew)-f(Xk))/tk]}$;
      if random(0,1) < $P$
          $X_k = X_{new}$;
      end if
    end for
    //Drop down the temperature
    $t_{k+1}$=drop($t_k$); $k$=$k$+1;
  end while
  print $X_{best}$
end procedure
```

7

2. 关键代码展示（带注释）

首先是一些基本函数：

更新棋盘函数：

```
direction = ((-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1,
1))

# to make sure the coord is reasonable
def reasonable(x,y):
    return x>=0 and x<8 and y>=0 and y<8

# to place new chess pieces to the chosen place and update the board
def new_board(board,x,y,color):
    if x < 0 or y < 0:
        return False
    board[x][y] = color
    valid = False
    for d in range(8):
        # search the eight directions one by one
        i = x + direction[d][0]
        j = y + direction[d][1]
        # find the last piece with opposite color
        while reasonable(i,j) and board[i][j] == -color:
            i += direction[d][0]
            j += direction[d][1]
        # reverse the pieces with the opposite color
        if reasonable(i,j) and board[i][j] == color:
            while True:
                i -= direction[d][0]
                j -= direction[d][1]
                if i == x and j == y:
                    break
                valid = True
                board[i][j] = color
    return valid
```

判断是否结束：

```
# if there is 0
def end(board):
    for i in range(8):
        for j in range(8):
            if board[i][j] == 0:
                return True
    return False

# if there are all 1 or all -1
def special(board,color):
    for i in range(8):
        for j in range(8):
            if board[i][j] == -color:
                return False
    return True
```

最终展示函数：

```
from alpha_beta import *
import numpy
from evaluation import execution
from new_board import*

board = numpy.zeros((8,8),dtype=numpy.int)
board[3][4] = board[4][3] = 1 #白
board[3][3] = board[4][4] = -1 #黑
AIColor = 1

print("Begin. Good Luck\n",board)
while(end(board)):
    if(special(board,AIColor)):
        print("No place to lay. You lost")
        break
    elif(special(board,-AIColor)):
        print("AI has no place to lay. You win")
        break
    move = AI_best(board,AIColor)
    x,y = move
    print("The AI chose :",move)
    new_board(board,x,y,1)
    print("After AI, the score is:",sum(sum(board)),"\nand the board is:")
    print(board)
    moves = execution(board,-AIColor)
    print("Accesssible choice:")
    print(moves[0])
    if moves[0] == []:
        input("No place to lay")
        continue
    else:
        i = input("Where to lay")
        # the order can not be changed
        if not i.isdigit() or int(i) >= len(moves[0]) or int(i) < 0:
            print("WARNING! IILEGAL INPUT. YOU LOST AN CHANCE!")
        else:
            yourmove = moves[0][int(i)]
            new_board(board,yourmove[0],yourmove[1],-AIColor)
            print("After Your choice, the score is:",sum(sum(board)),"\nand the 
board is:")
            print(board)

score = sum(sum(board))
if score < 0:
    print("You keep more than AI. You win")
elif score == 0:
    print("DEUCE")
else:
    print("AI keeps more than you. You lost")
```

下面为重点算法部分：

1. 评估函数：

> 评估函数设计：
>
> 1. 权值表：边、角、中心的相对重要程度
> 2. 行动力：棋手合法的可能棋步数量
> 3. 稳定子：绝对不会被翻转的棋子。例如四角
>
> 事实证明稳定子对程序影响不大，于是程序中进行了删减

1. **权值表：**

```
# 1.
# evaluation map, for showing the importance
# I don't know how to get the chart, it comes from web listed in the end. I 
guess it's a kind of experience
Vmap = numpy.array(
    [[500,-25,10,5,5,10,-25,500],
    [-25,-45,1,1,1,1,-45,-25],
    [10,1,3,2,2,3,1,10],
    [5,1,2,1,1,2,1,5],
    [5,1,2,1,1,2,1,5],
    [10,1,3,2,2,3,1,10],
    [-25,-45,1,1,1,1,-45,-25],
    [500,-25,10,5,5,10,-25,500]]
)

# to cauclate the position's weight
def position_weight(board,color):
    return sum(sum(board * Vmap))*color
```

2. **行动力表，用于获取可下子点**

```
# 2.
# The possible directions
def execution(board,color):
    # accessible choices
    moves = []
    # possible moved boards
    ValidBoardList = []
    for i in range(8):
        for j in range(8):
            if board[i][j] == 0:
                newboard = board.copy()
                if new_board(newboard,i,j,color): # if can get some pieces
reversed
                    moves.append((i,j))
                    ValidBoardList.append(newboard)
    return moves, ValidBoardList
```

3. **稳定子表，用于计算稳定子**

```
#3.
# The pieces that never will be reversed
def getstable(board, color): #稳定子
    stable = [0,0,0]
    # 角, 边, 八个方向都无空格
    cind1 = [0,0,7,7]
    cind2 = [0,7,7,0]
    inc1 = [0,1,0,-1]
    inc2 = [1,0,-1,0]
    stop = [0,0,0,0]
    for i in range(4):
        if board[cind1[i]][cind2[i]] == color:
            stop[i] = 1
            stable[0] += 1
            for j in range(1,7):
                if board[cind1[i]+inc1[i]*j][cind2[i]+inc2[i]*j] != color:
                    break
                else:
                    stop[i] = j + 1
                    stable[1] += 1
    for i in range(4):
        if board[cind1[i]][cind2[i]] == color:
            for j in range(1,7-stop[i-1]):
                if board[cind1[i]-inc1[i-1]*j][cind2[i]-inc2[i-1]*j] !=
color:
                    break
                else:
                    stable[1] += 1
    colfull = numpy.zeros((8, 8), dtype=numpy.int)
    colfull[:,numpy.sum(abs(board), axis = 0) == 8] = True
    rowfull = numpy.zeros((8, 8), dtype=numpy.int)
    rowfull[numpy.sum(abs(board), axis = 1) == 8,:] = True
    diag1full = numpy.zeros((8, 8), dtype=numpy.int)
    for i in range(15):
        diagsum = 0
        if i <= 7:
            sind1 = i
            sind2 = 0
            jrange = i+1
        else:
            sind1 = 7
            sind2 = i-7
            jrange = 15-i
        for j in range(jrange):
            diagsum += abs(board[sind1-j][sind2+j])
        if diagsum == jrange:
            for k in range(jrange):
                diag1full[sind1-j][sind2+j] = True
    diag2full = numpy.zeros((8, 8), dtype=numpy.int)
    for i in range(15):
        diagsum = 0
        if i <= 7:
            sind1 = i
            sind2 = 7
            jrange = i+1
        else:
            sind1 = 7
            sind2 = 14-i
            jrange = 15-i
        for j in range(jrange):
            diagsum += abs(board[sind1-j][sind2-j])
        if diagsum == jrange:
            for k in range(jrange):
                diag2full[sind1-j][sind2-j] = True
    stable[2] =
sum(sum(numpy.logical_and(numpy.logical_and(numpy.logical_and(colfull,
rowfull), diag1full), diag2full)))
    return stable
```

4. **计算效益值：**

```
# get evaluation
def evaluation(moves,board,color):
    moves_avaiable, ValidBoardList = execution(board,-color)
    value = position_weight(board,color) + 15* (len(moves)-
len(moves_avaiable))+10*sum(stable)
    return value
```

2. Alpha-Beta剪枝算法

1. **搜索深度：**

```
# to decide the best choice for AI
# pick different depths on different situations
def AI_best(board, mycolor):
    stage = sum(sum(abs(board)))
    if stage <= 9:
        depth = 5
    elif stage >= 49:
        depth = 6
    else:
        depth = 4
    value, bestmove = Alpha_Beta(board, depth, -10000, 10000, mycolor,
mycolor, depth)
    return bestmove
```

2. **Alpha_Beta算法:**

> 说明：A-B算法似乎也可以看作是一个模拟退火的过程，depth为温度，evaluation()为估值函数，score和bestscore比较过程为挑选过程

```
# Alpha_Beta Pruning algorithm
def Alpha_Beta(board,depth,alpha,beta,actcolor,mycolor,maxdepth):
    # to get vaild positions
    moves,ValidBoardList = execution(board, actcolor)
    # nowhere to lay pieces, return an illegal number, and it's rival's turn
    if len(moves) == 0:
        return evaluation(moves,board,mycolor),(-1,-1)
    # run out of depth, end search and return
    if depth == 0:
        return evaluation(moves,board,mycolor),[]
    # beginning
    if depth == maxdepth:
        for i in range(len(moves)):
            # if it is a stable pieces, we have to choose it
            if Vmap[moves[i][0]][moves[i][1]] == Vmap[0][0] and actcolor ==
mycolor:
                return 1000,moves[i]
        if depth >= 4:
            V_value = []
            for i in range(len(ValidBoardList)):
                value, bestmove = Alpha_Beta(ValidBoardList[i], 1, -10000,
10000, -actcolor, mycolor, maxdepth)
            V_value.append(value)
        # sort the data in an ascending order
        ind = numpy.argsort(V_value)
        maxN = 6
        moves = [moves[i] for i in ind[0:maxN]]
        ValidBoardList = [ValidBoardList[i] for i in ind[0:maxN]]

    bestmove = []
    bestscore = -10000
    for i in range(len(ValidBoardList)):
        score, move = Alpha_Beta(ValidBoardList[i], depth-1, -beta, -
max(alpha, bestscore), -actcolor, mycolor, maxdepth)
        score = -score
        if score > bestscore:
            bestscore = score
            bestmove = moves[i]
            # cut off
            if bestscore > beta:
                return bestscore, bestmove
    return bestscore, bestmove
```

3. 模拟退火算法

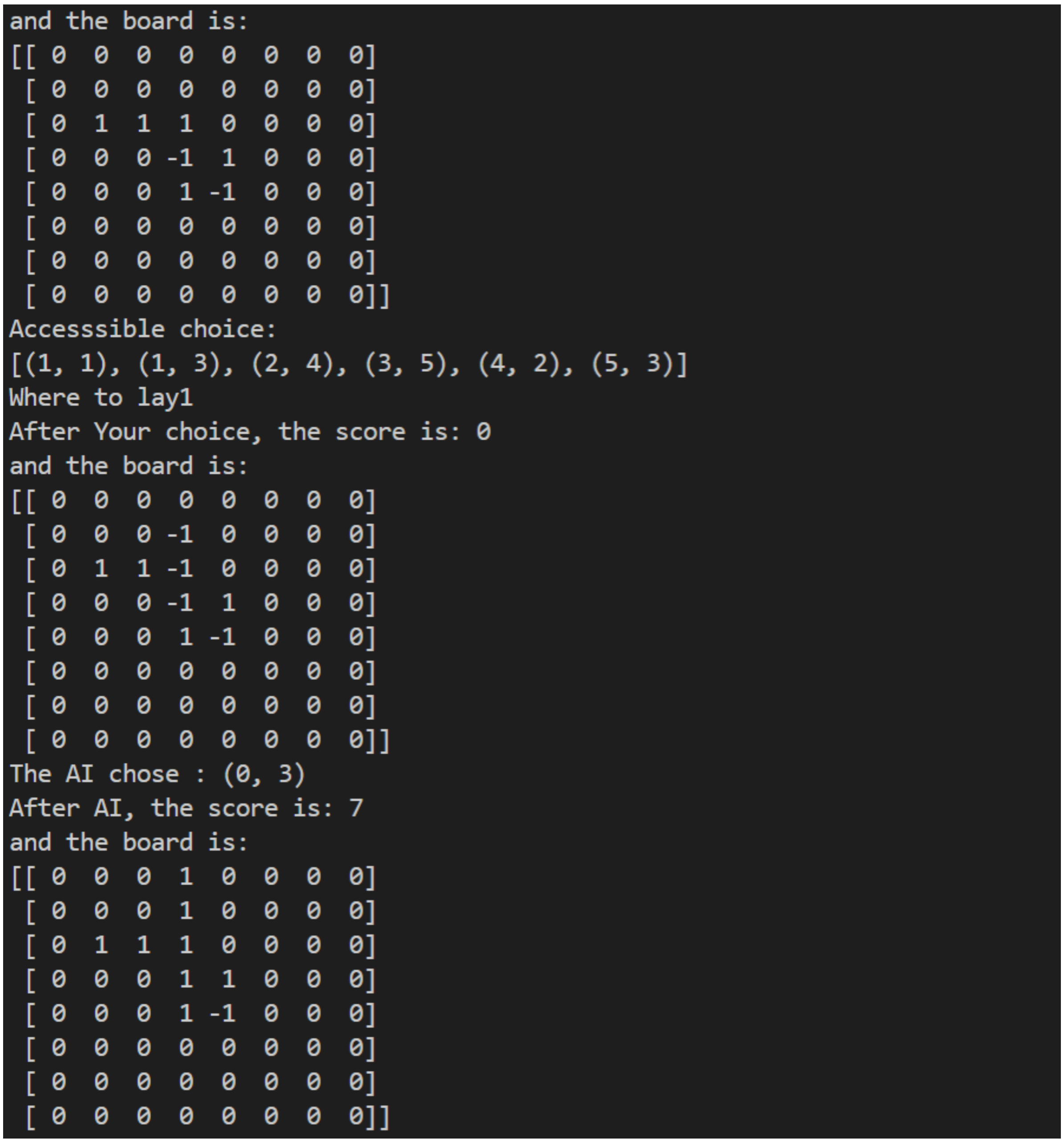
```
# get new x,y
def generate_new(x, y, T):
    x_new = int(x + T * (random() - random())*9)
    y_new = int(y + T * (random() - random())*9)
    return x_new,y_new

# to judge if is's acceptable
def Metrospolis(f, f_new,T):
    if(f_new > f):
        return 1
    else:
        p = math.exp((f-f_new)/T)
        if random() < p:
            return 1
        else:
            return 0

# def run(board,actcolor,mycolor,T=100,Tf=1):
#     bestmove = []
#     while T > Tf:
#         moves,ValidBoardList = execution(board, actcolor)
#         for i in range(len(ValidBoardList)):
#             score = evaluation(moves,ValidBoardList[i],mycolor)
#             x_new,y_new = generate_new(moves[i][0],moves[i][1])
#             score_new = generate_new(x_new,y_new)
#             if Metrospolis(score,score_new,T):
#                 bestmove.append((x_new,y_new))
#                 best_score = score_new
#         T = T * 0.1
#     return best_score,bestmove

def SA_A_B(board,depth,actcolor,T=100,Tf=0):#T=depth
    moves,ValidBoardList = execution(board,actcolor)
    # 2 special examples
    if len(moves) == 0:
        return evaluation(moves,board,actcolor),(-1,-1)
    if depth == 0:
        return evaluation(moves,board,actcolor),[]
    # when it's still have tem
    while(T > Tf):
        x_index,y_index = moves[0][0],moves[0][1]
        board_index = ValidBoardList[0]
        for i in range(len(moves)):
            # to get len(moves) random numbers
            new_board(board_index,x_index,y_index,actcolor)
            score = sum(sum(board * Vmap))*actcolor
            # new value
            x_new,y_new = generate_new(moves[i][0],moves[i][1],T)
            index,next_board = new_board(board_index,x_new,y_new,actcolor)
            if(index):
                score_new = sum(sum(board * Vmap))*actcolor
                # if it's acceptable
                if Metrospolis(score,score_new,T):
                    bestmove = (x_new,y_new)
                    bestscore = score_new
            # update the baseline
            x_index = x_new
            y_index = y_new
            board_index = next_board
        # SA
        T = T * 0.1
    return bestscore,bestmove

def AI_best(board, color):
    stage = sum(sum(abs(board)))
    if stage <= 9:
        depth = 5
    elif stage >= 49:
        depth = 6
    else:
        depth = 4
    _, bestmove = SA_A_B(board, depth, color, color)
    return bestmove
```

三、 实验结果及分析

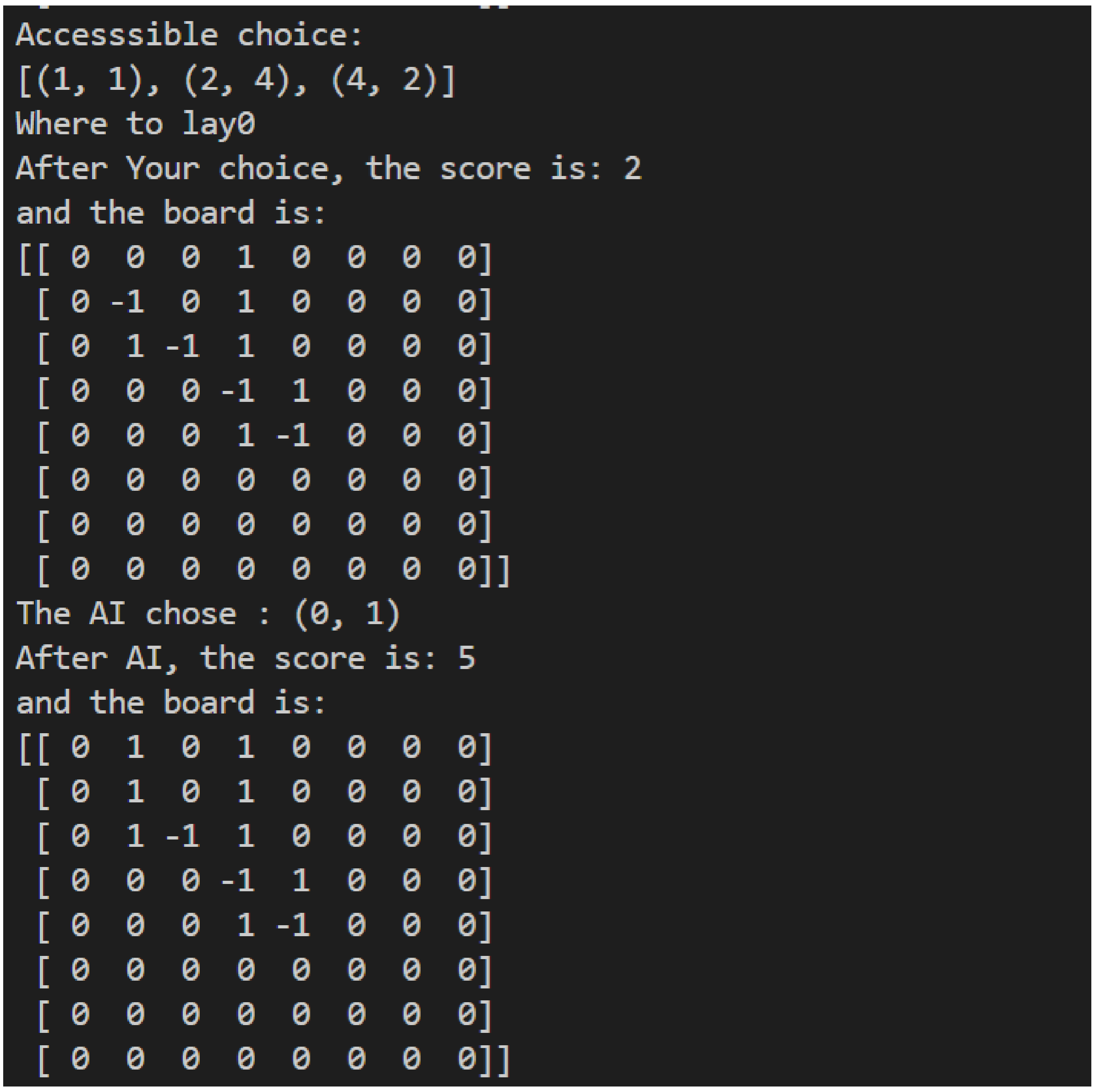
实验结果展示示例（可图可表可文字，尽量可视化）

正常进行：

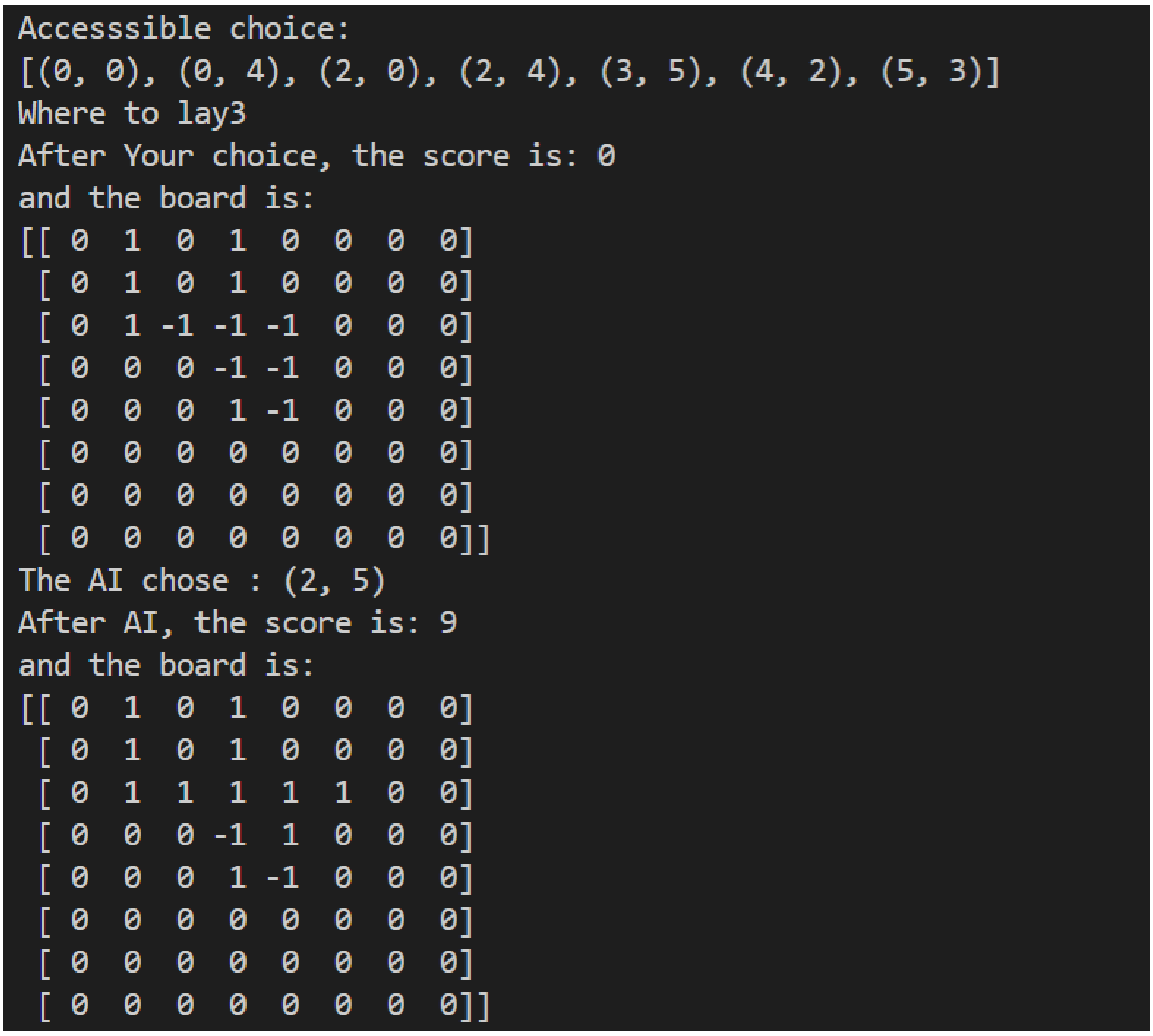
> **Accessible choice 是可以落子的位置，输入下标来落子**
>
> **the score 是总分数，计算方式为8个维度的和**



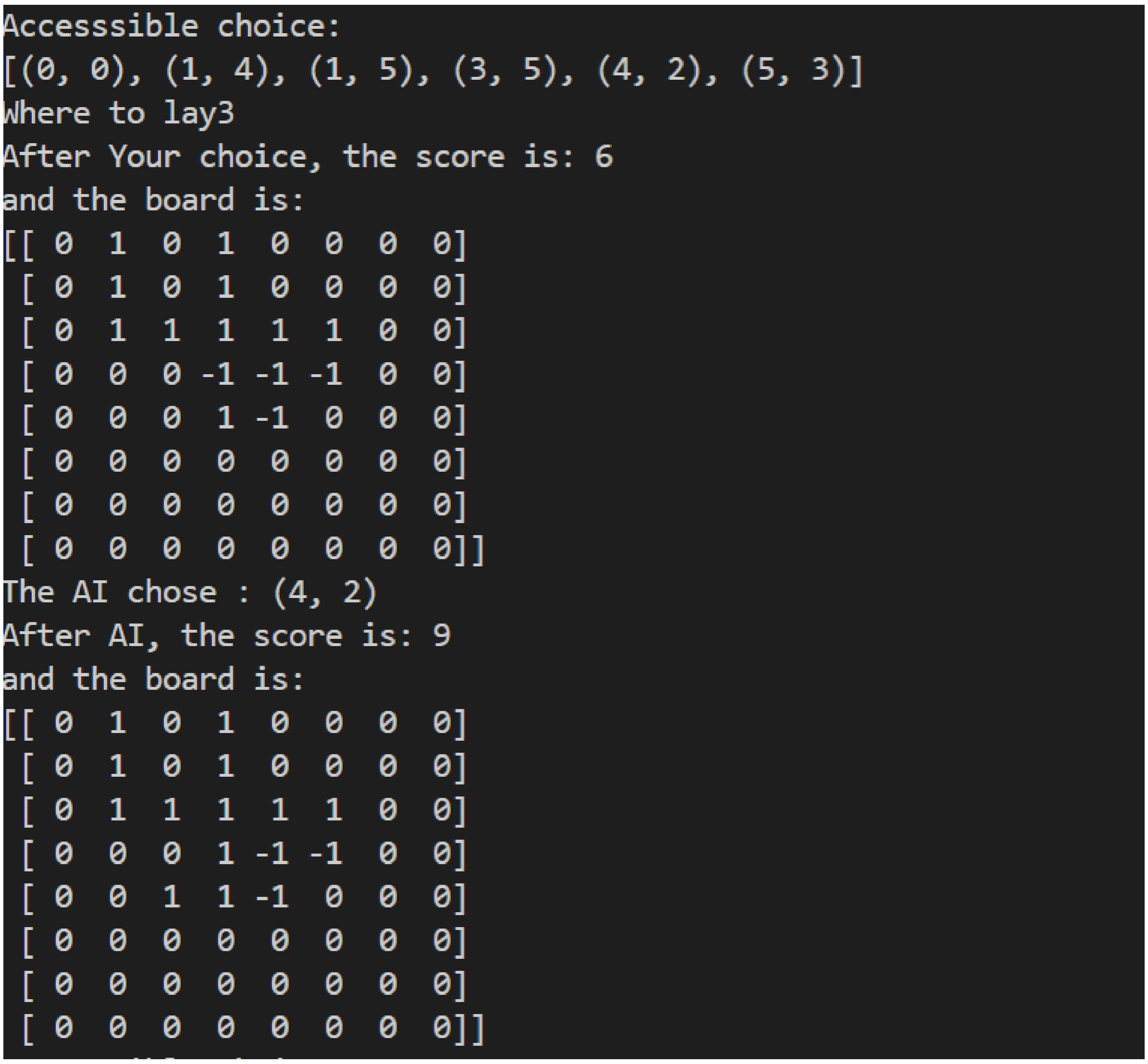
```
and the board is:
[[ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  1  1  1  0  0  0  0]
 [ 0  0  0 -1  1  0  0  0]
 [ 0  0  0  1 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
Accesssible choice:
[(1, 1), (1, 3), (2, 4), (3, 5), (4, 2), (5, 3)]
Where to lay1
After Your choice, the score is: 0
and the board is:
[[ 0  0  0  0  0  0  0  0]
 [ 0  0  0 -1  0  0  0  0]
 [ 0  1  1 -1  0  0  0  0]
 [ 0  0  0 -1  1  0  0  0]
 [ 0  0  0  1 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
The AI chose : (0, 3)
After AI, the score is: 7
and the board is:
[[ 0  0  0  1  0  0  0  0]
 [ 0  0  0  1  0  0  0  0]
 [ 0  1  1  1  0  0  0  0]
 [ 0  0  0  1  1  0  0  0]
 [ 0  0  0  1 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
```



```
Accesssible choice:
[(1, 1), (2, 4), (4, 2)]
Where to lay0
After Your choice, the score is: 2
and the board is:
[[ 0  0  0  1  0  0  0  0]
 [ 0 -1  0  1  0  0  0  0]
 [ 0  1 -1  1  0  0  0  0]
 [ 0  0  0 -1  1  0  0  0]
 [ 0  0  0  1 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
The AI chose : (0, 1)
After AI, the score is: 5
and the board is:
[[ 0  1  0  1  0  0  0  0]
 [ 0  1  0  1  0  0  0  0]
 [ 0  1 -1  1  0  0  0  0]
 [ 0  0  0 -1  1  0  0  0]
 [ 0  0  0  1 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
```



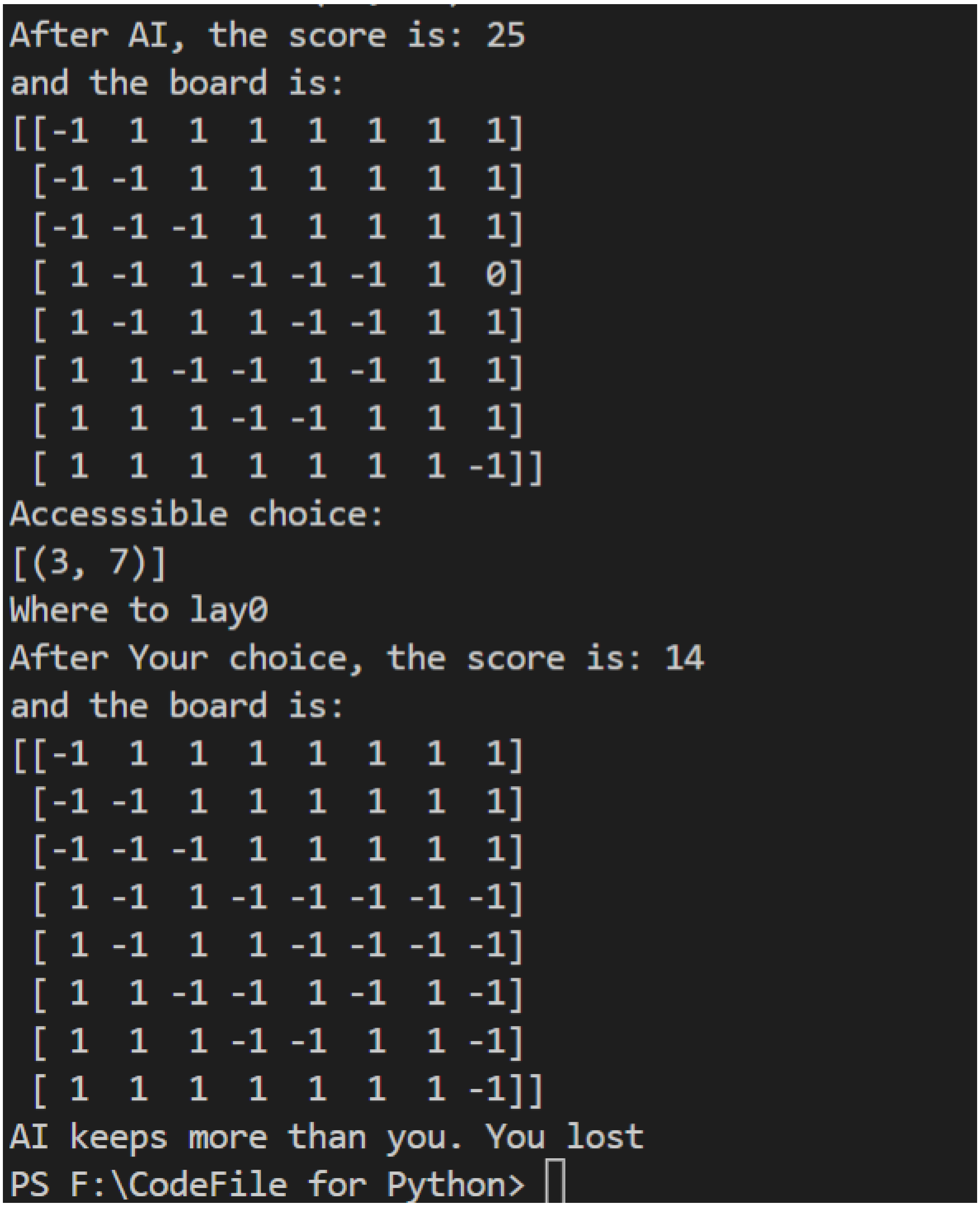
```
Accesssible choice:
[(0, 0), (0, 4), (2, 0), (2, 4), (3, 5), (4, 2), (5, 3)]
Where to lay3
After Your choice, the score is: 0
and the board is:
[[ 0  1  0  1  0  0  0  0]
 [ 0  1  0  1  0  0  0  0]
 [ 0  1 -1 -1 -1  0  0  0]
 [ 0  0  0 -1 -1  0  0  0]
 [ 0  0  0  1 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
The AI chose : (2, 5)
After AI, the score is: 9
and the board is:
[[ 0  1  0  1  0  0  0  0]
 [ 0  1  0  1  0  0  0  0]
 [ 0  1  1  1  1  1  0  0]
 [ 0  0  0 -1  1  0  0  0]
 [ 0  0  0  1 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
```



```
Accesssible choice:
[(0, 0), (1, 4), (1, 5), (3, 5), (4, 2), (5, 3)]
Where to lay3
After Your choice, the score is: 6
and the board is:
[[ 0  1  0  1  0  0  0  0]
 [ 0  1  0  1  0  0  0  0]
 [ 0  1  1  1  1  1  0  0]
 [ 0  0  0 -1 -1 -1  0  0]
 [ 0  0  0  1 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
The AI chose : (4, 2)
After AI, the score is: 9
and the board is:
[[ 0  1  0  1  0  0  0  0]
 [ 0  1  0  1  0  0  0  0]
 [ 0  1  1  1  1  1  0  0]
 [ 0  0  0  1 -1 -1  0  0]
 [ 0  0  1  1 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
```

结束:

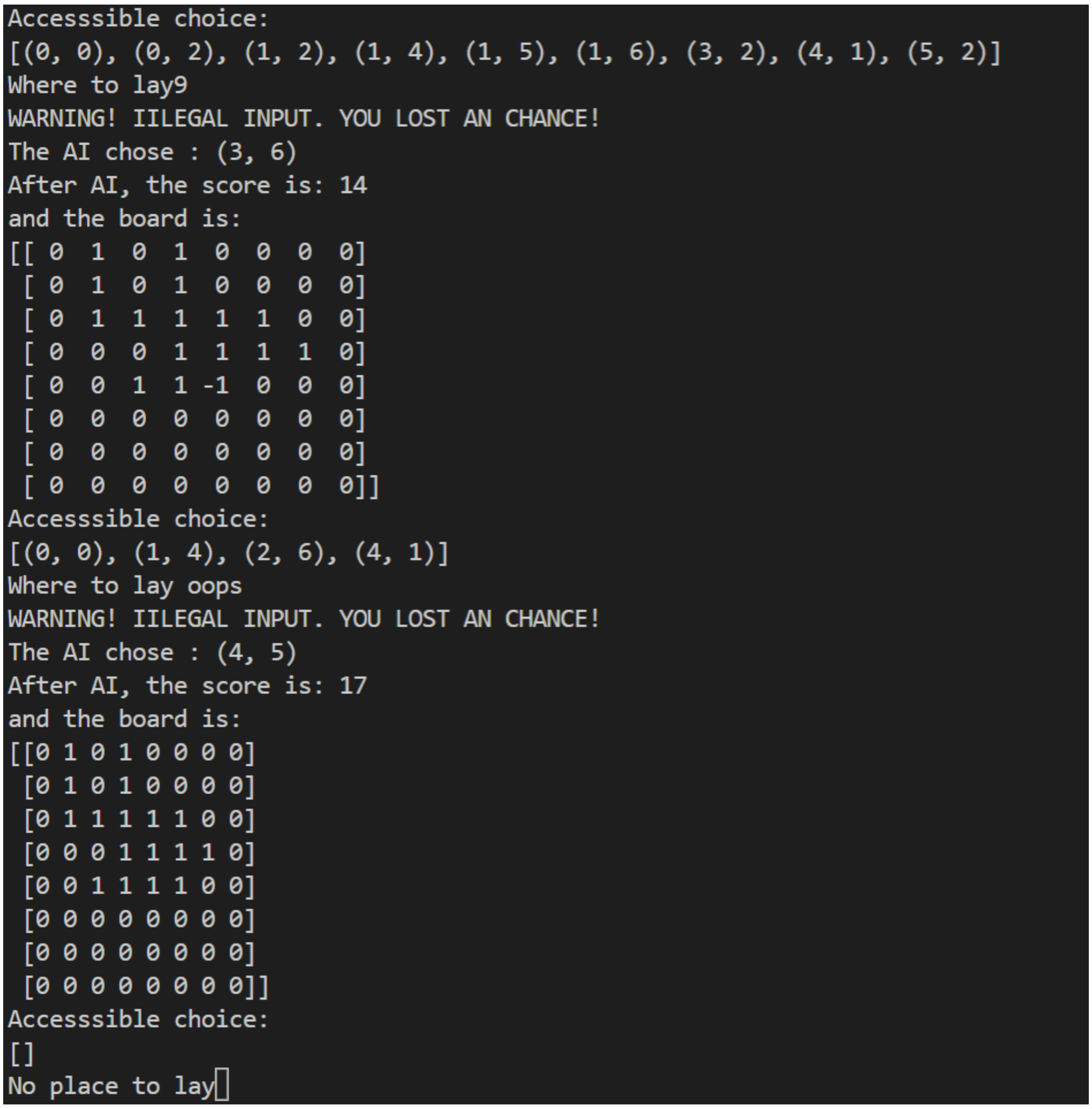
全部占据:



```
After AI, the score is: 25
and the board is:
[[-1  1  1  1  1  1  1  1]
 [-1 -1  1  1  1  1  1  1]
 [-1 -1 -1  1  1  1  1  1]
 [ 1 -1  1 -1 -1 -1  1  0]
 [ 1 -1  1  1 -1 -1  1  1]
 [ 1  1 -1 -1  1 -1  1  1]
 [ 1  1  1 -1 -1  1  1  1]
 [ 1  1  1  1  1  1  1 -1]]
Accesssible choice:
[(3, 7)]
Where to lay0
After Your choice, the score is: 14
and the board is:
[[-1  1  1  1  1  1  1  1]
 [-1 -1  1  1  1  1  1  1]
 [-1 -1 -1  1  1  1  1  1]
 [ 1 -1  1 -1 -1 -1 -1 -1]
 [ 1 -1  1  1 -1 -1 -1 -1]
 [ 1  1 -1 -1  1 -1  1 -1]
 [ 1  1  1 -1 -1  1  1 -1]
 [ 1  1  1  1  1  1  1 -1]]
AI keeps more than you. You lost
PS F:\CodeFile for Python>
```

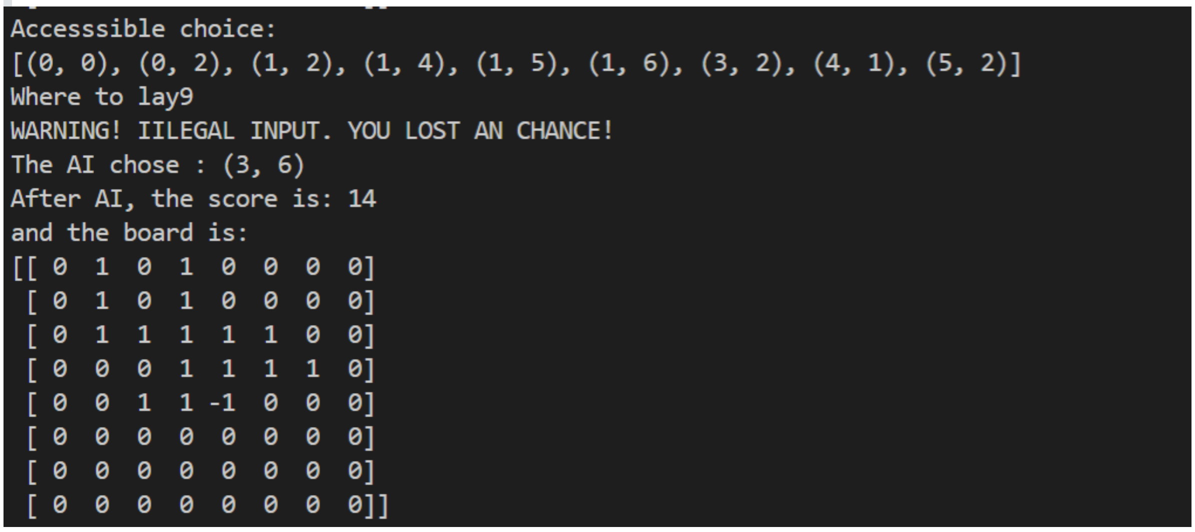
无子可下：

> **因为输入大于可选择的范围或者不合规范的字符串丧失一次落子的机会**



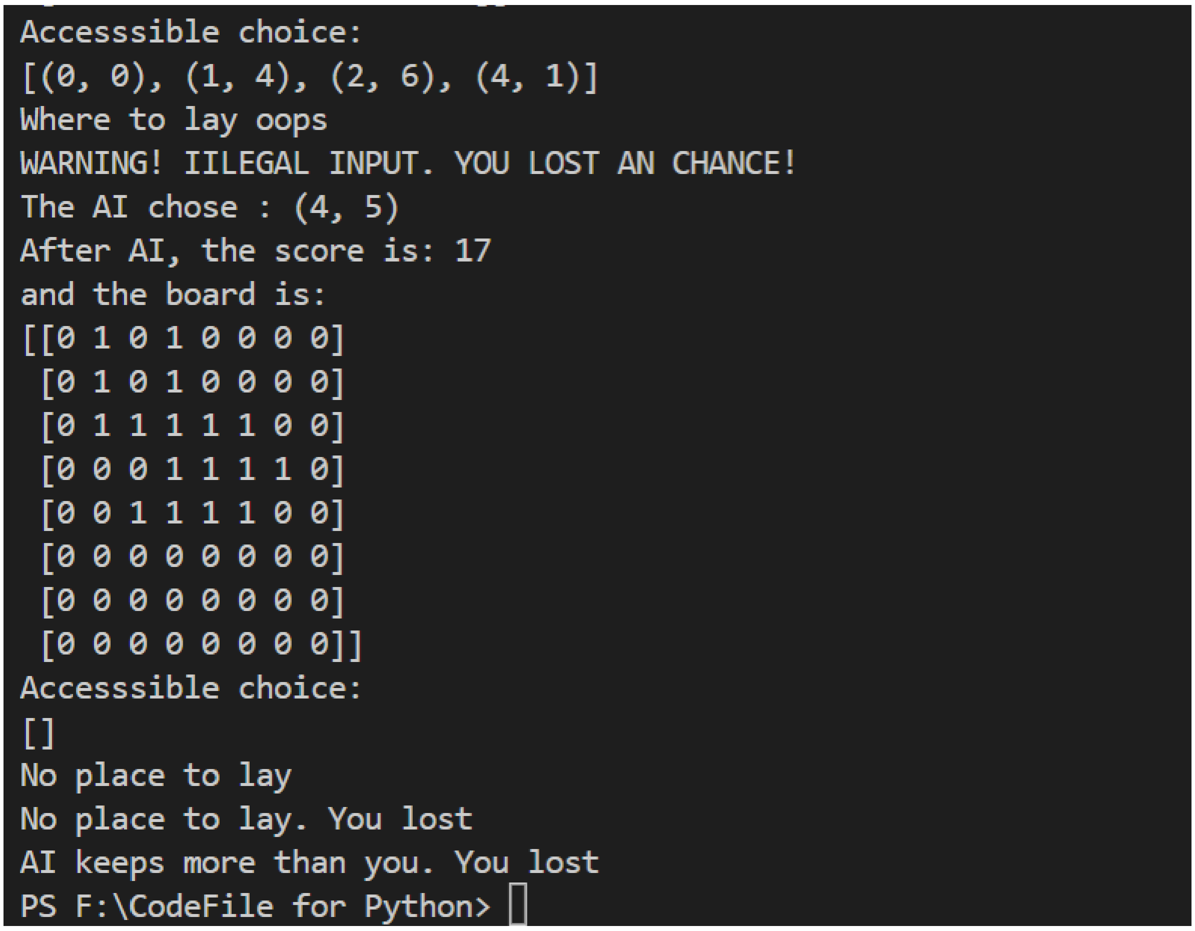
```
Accesssible choice:
[(0, 0), (0, 2), (1, 2), (1, 4), (1, 5), (1, 6), (3, 2), (4, 1), (5, 2)]
Where to lay9
WARNING! IILEGAL INPUT. YOU LOST AN CHANCE!
The AI chose : (3, 6)
After AI, the score is: 14
and the board is:
[[ 0  1  0  1  0  0  0  0]
 [ 0  1  0  1  0  0  0  0]
 [ 0  1  1  1  1  1  0  0]
 [ 0  0  0  1  1  1  1  0]
 [ 0  0  1  1 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
Accesssible choice:
[(0, 0), (1, 4), (2, 6), (4, 1)]
Where to lay oops
WARNING! IILEGAL INPUT. YOU LOST AN CHANCE!
The AI chose : (4, 5)
After AI, the score is: 17
and the board is:
[[0 1 0 1 0 0 0 0]
 [0 1 0 1 0 0 0 0]
 [0 1 1 1 1 1 0 0]
 [0 0 0 1 1 1 1 0]
 [0 0 1 1 1 1 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
Accesssible choice:
[]
No place to lay
```

> **输入大于可选择的范围：**



```
Accesssible choice:
[(0, 0), (0, 2), (1, 2), (1, 4), (1, 5), (1, 6), (3, 2), (4, 1), (5, 2)]
Where to lay9
WARNING! IILEGAL INPUT. YOU LOST AN CHANCE!
The AI chose : (3, 6)
After AI, the score is: 14
and the board is:
[[ 0  1  0  1  0  0  0  0]
 [ 0  1  0  1  0  0  0  0]
 [ 0  1  1  1  1  1  0  0]
 [ 0  0  0  1  1  1  1  0]
 [ 0  0  1  1 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
```

> **输入不是数字**



```
Accesssible choice:
[(0, 0), (1, 4), (2, 6), (4, 1)]
Where to lay oops
WARNING! IILEGAL INPUT. YOU LOST AN CHANCE!
The AI chose : (4, 5)
After AI, the score is: 17
and the board is:
[[0 1 0 1 0 0 0 0]
 [0 1 0 1 0 0 0 0]
 [0 1 1 1 1 1 0 0]
 [0 0 0 1 1 1 1 0]
 [0 0 1 1 1 1 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
Accesssible choice:
[]
No place to lay
No place to lay. You lost
AI keeps more than you. You lost
PS F:\CodeFile for Python> 
```

五、 参考资料

> https://zhuanlan.zhihu.com/p/35121997

> https://blog.csdn.net/weixin_48241292/article/details/109468947