ASSIGNMENT 2
AI & ML

Dr. Marwan Torki

Asmaa Gamal
Communication & Electronics

Using Minimax Algorithm to Make Connect 4 AI Agent

The Code

1.The “main.py”:

```
# Asmaa Gamal
#Assignment 2
# Electronics & Communications department
#Using Minimax Algorithm to Make Connect 4 AI Agent

#---------------------------------- libraries & initializations -----------------------------
from   GUI     import  *
import  numpy    as    np
import  pygame
import  timeit                                                          #Measure execution
time of small codes
import  random
import  math
import  sys

# ------------------------------------The 4-connect Game ----------------------------------
print(" ----------------------------------The 4-connect Game using Minimax Algorithms----------
---------------------------- ")
print("-----------------------------------------------------Welcome To Your Game-----------------
-------------------------------")
print_cyan("To start PLZ press on the Algorithm desired type buttons, and press one of the depth
buttons from 1 to 4.. if the depth is greater than this: your CPU, RAMS, and your whole computer
will suffer and start to be slow")
#-------------------------initialization
#the two players:
HUMAN=0
AI  =1

#the circular pieces:
empty     =0
HUMAN_PIECE =1
AI_PIECE    =2

#the dimensions:
total_rows       =6                                                      #length
total_columns =7                                                        #width
focused_lenght=4                                                        #this is the length
of window that i gonna zoom and focus on it later to calculate the scoring utility mechanism

#initializing the sum of the two total scores:
Human_wins=0
AI_wins      =0


#Nodes_expanded:
MinmaxNodes_expanded =0
PruningNodes_expanded=0

total_minmaxNodes =0
total_PruningNodes=0

#total_time_sum
Minmax_time =0
Pruning_time=0

#open_a_file_for_printing_the_tree
file = open('Tree_Output.txt','w')


#playing the game voice:
pygame.mixer.init()
pygame.mixer.music.load("myVoice.mp3")
```

```
pygame.mixer.music.play()

'''
to open any file, but i should still close it manually:
        import os
        os.startfile('myVoice.mp3')
... so, to make it easier i used pygame built-in methods
'''

#-----------------------------------------------------The Game environment-----------------------
-------------------------------------
def CreatingtheBoard():
    board=np.zeros((total_rows,total_columns))
    return board

def print_board(board):
   print("The Board After Updating It:")
   print_cyan(np.flip(board, 0))                                                    #((0 = the x
axis)),so, this np fun aims at flipping the board up side down to reverse the index and make the
0,0 element at the bottom of the board
   print("\n")

#-------some actions we need to do and check in order to drop a piece or knowing the end

def check_terminal_nodes(board):
        return len(get_valid_locations(board)) == 0


def check_valid_locations(board, col):
   return board[total_rows-1][col] == 0                                             #there is 6 rows
from zero to 5 so we need to check starting from the 5th row and rising up so, if it is empty
then it is true and ready to drop a piece in it

def get_valid_locations(board):
   valid_locations = []
   for col in range(total_columns):
      if check_valid_locations(board, col):                                         #if true == empty
         valid_locations.append(col)
   return valid_locations

def TheNextOpenRow(board, col):                                                     #which row is
valid in this specific column
   for r in range(total_rows):
      if board[r][col] == 0:
         return r

def dropping_pieces(board, row, col, piece):                                         #piece = 1 if
human, or 2 if computer, 0r 0 if empty
   board[row][col] = piece                                                         #we could not
return anything here because the board is a tuple so it can't change


#----------------------------preparing for the AI algorithm part ------------------------------
#---------------the count of the wins
def WinningMoves(board, piece):
      #horizontal winning
      for c in range(total_columns-3):
         for r in range(total_rows):
            if board[r][c] == piece and board[r][c+1] == piece and board[r][c+2] == piece and
board[r][c+3] == piece:
               return True


      # vertical win
      for c in range(total_columns):
         for r in range(total_rows-3):
            if board[r][c] == piece and board[r+1][c] == piece and board[r+2][c] == piece and
board[r+3][c] == piece:
```

```
                return True


        # positively sloped diagonals
        for c in range(total_columns-3):
            for r in range(total_rows-3):
                if board[r][c] == piece and board[r+1][c+1] == piece and board[r+2][c+2] == piece and
board[r+3][c+3] == piece:
                return True


        # negatively sloped diagonals
        for c in range(total_columns-3):
            for r in range(3, total_rows):
                if board[r][c] == piece and board[r-1][c+1] == piece and board[r-2][c+2] == piece and
board[r-3][c+3] == piece:
                return True



#-------------the calculating of utilities
def give_utilities(window, piece):                                          #window is an
array that will contain  my focus area of the game board
    score = 0
    opponent_piece = HUMAN_PIECE
    if piece == HUMAN_PIECE:
        opponent_piece = AI_PIECE                                          #this is to switch
turns

    if window.count(piece) == 4:
        score += 1000
    elif window.count(piece) == 3 and window.count(empty) == 1:
        score += 100
    elif window.count(piece) == 2 and window.count(empty) == 2:
        score += 10

    if window.count(opponent_piece) == 3 and window.count(empty) == 1:
        score -= 1500                                                  # this will decrease the
above score by the opponent score, this means the computer will choose to stop u from winning
firstly then to achieve his own win even if the computer had to postpone his own progress in the
game... and i don't prefer to write a program using this concept in the real life.. I will let
other people win and only focus in my own win
                                                                       #it is a big negative value
to make sure that the total score in the future will indicate that u r lossing the game if its
result remains a negative value
    return score




#--------------knowing the utilities positions
def utilities_location(board, piece):                                  # looping the whole
board to know my score locations
    score = 0
    '''
    this below line-------------------------------------> center_array = [int(i) for i in
list(board[:,total_columns//2])]
    is the same as saying:
    ======================
    a            =  board[:,3]          ----------------> this will print a= [0. 0. 0. 0. 0. 0.]
    b            =  list(a)             ----------------> this will print b= [0.0, 0.0, 0.0, 0.0,
0.0, 0.0]
    center_array =  [int(i) for i in b]    ----------------> this will print c= [0, 0, 0, 0, 0, 0]
    '''
    # center column Score
    center_array = [int(i) for i in list(board[:,total_columns//2])]                #int just to
make sure it's an integar this means: loop the whole rows in the 3rd column because it is the
```

```
center column as the seven colums are from 0 to 6
   center_count = center_array.count(piece)                                    #this fun will count
the similar pieces in the center column
   score += center_count * 10

   # Horizontal Score
   for r in range(total_rows):
      row_array = [int(i) for i in list(board[r,:])]                             #all columns
#row_array is for example: [0, 0, 0, 2, 2, 0, 0]
      for c in range(total_columns-3):                                          #until the the 4th
column as mentioned above
         window = row_array[c:c+focused_lenght]                                    # from c to c+4
         score += give_utilities(window, piece)                                  #recursion

   # Vertical Score
   for c in range(total_columns):
      col_array = [int(i) for i in list(board[:,c])]                             #all rows
      for r in range(total_rows-3):                                             #until the the 3rd row
as mentioned above
         window = col_array[r:r+focused_lenght]                                  # from r to r+4
         score += give_utilities(window, piece)                                  #recursion


   # Positively-sloped diagonal score((my focused window will go up ))
   for r in range(total_rows-3):                                                #no winning moves can
start after that row
      for c in range(total_columns-3):                                          #no winning moves can
start after that column
         window = [board[r+i][c+i] for i in range(focused_lenght)]
#focused_lenght=4 so this loop is from 0 to 3
         score += give_utilities(window, piece)                                  #recursion

   # Negatively-sloped diagonal score ((my focused window will go from up to down ))
   for r in range(total_rows-3):                                                #no winning moves can
start after that row
      for c in range(total_columns-3):                                          #no winning moves can
start after that column
         window = [board[r+3-i][c+i] for i in range(focused_lenght)]
#focused_lenght=4 so this loop is from 0 to 3
         score += give_utilities(window, piece)                                  #recursion

   return score                                                               #this is the assuming score
of the board copy 3ala al wd3ya al gdeda de

#-----------------------------------------------the algorithm ---------------------------------
  #----------------------------firstly: •The Minimax without alpha-beta pruning •----------------

def maxmin_WithoutPruning(board, depth, max_min_player):                        #max_min_player is a
true or false value.. for ex: it is true if computer is playing and false if human is playing
   global MinmaxNodes_expanded,file
   MinmaxNodes_expanded+=1

   valid_locations = get_valid_locations(board)
   is_terminal    = check_terminal_nodes(board)


   if is_terminal:
         if WinningMoves(board, AI_PIECE):
            return (None, 100000000000000)                                  #computer wins
         elif WinningMoves(board, HUMAN_PIECE):
            return (None,-10000000000000)                                   #human wins
         else:                                                           # Game is over == no more valid
places for a moves
            return (None, 0)                                              #a tie or a draw
   if depth == 0:
         return (None, utilities_location(board, AI_PIECE))
```

```
   if max_min_player:                                                    #the AI computer is
thinking as a maximizer to evaluate the future game
      value = -math.inf                                                 #value is the maxUtility in
the adversarial search lec algorithm
      column = random.choice(valid_locations)                                  #this fun will
return a random col as an initial choose to begin this algorithm

      for col in valid_locations:
         row =  TheNextOpenRow(board, col)
         board_copy = board.copy()                                      #this copy will allow the
AI to compare between the final states result from it
         dropping_pieces(board_copy, row, col, AI_PIECE)

         new_score = maxmin_WithoutPruning(board_copy, depth-1,  False)[1]          #recursion
from the deepest to the shallowest # and this [1] index is for the second element in the tuple
resulting from the minimax fun to put it in the "new_score"

         # printing_the_utilities_tree
         file.write("The score "                 + str(new_score))
         file.write("\nThe Board State achieving it:\n"  + str(np.flip(board_copy, 0)))
         file.write("\n\n\n\n")



         if new_score > value:                                          #value is the maxUtility
in the adverserial search lec but i prefered to call it value as what wikipedia told me here in
its pseudocode : https://en.wikipedia.org/wiki/Minimax

            value = new_score
            column = col

      return column, value                                              #returning a tuple of only
2 elements

   else:                                                              # the AI computer is thinking
as a minimizier to evaluate the future game
      value = math.inf                                                  #value is the minUtility in
the adversarial search lec algorithm
      column = random.choice(valid_locations)                                  #this fun will
return a random col as an initial choose to begin this algorithm
      for col in valid_locations:
         row = TheNextOpenRow(board, col)
         board_copy = board.copy()                                      #this copy will allow the
AI to compare between the final states result from it
         dropping_pieces(board_copy, row, col, HUMAN_PIECE)
         new_score = maxmin_WithoutPruning(board_copy, depth-1, True)[1]            #recursion
from the deepest to the shallowest # and this [1] index is for the second element in the tuple
resulting from the minimax fun to put it in the "new_score"

         # printing_the_utilities_tree
         file.write("The score="                 + str(new_score))
         file.write("\nThe Board State achieving it:\n"   + str(np.flip(board_copy, 0)))
         file.write("\n\n\n\n")


         if new_score < value:                                          #value is the minUtility
in the adverserial search lec but i prefered to call it value as what wikipedia told me here in
its pseudocode : https://en.wikipedia.org/wiki/Minimax
            value = new_score
            column = col

      return column, value                                              #returning a tuple of
only 2 elements, where value is the minUtility in the adverserial search lec but i prefered to
call it value as what wikipedia told me here in its pseudocode :
https://en.wikipedia.org/wiki/Minimax
```

```
#----------------------------Secondly: •The Minimax with alpha-beta pruning •  ----------------

def max_min(board, depth, alpha, beta, max_min_player):                          #max_min_player
is a true or false value.. for ex: it is true if computer is playing and false if human is
playing

   global PruningNodes_expanded,file
   PruningNodes_expanded+=1

   valid_locations = get_valid_locations(board)
   is_terminal    = check_terminal_nodes(board)



   if is_terminal:
         if WinningMoves(board, AI_PIECE):
            return (None, 100000000000000)                                  #computer wins
         elif WinningMoves(board, HUMAN_PIECE):
            return (None, -100000000000000)                                 #human wins
         else:                                                           # Game is over == no more valid
places for a moves
            return (None, 0)                                             #a tie or a draw
   if depth == 0:
         return (None, utilities_location(board, AI_PIECE))


   if max_min_player:                                                     #the AI computer is thinking as a
maximizer to evaluate the future game
      value = -math.inf                                                  #value is the maxUtility in the
adversarial search lec algorithm
      column = random.choice(valid_locations)                                  #this fun will return a
random col as an initial choose to begin this algorithm

      for col in valid_locations:
         row =  TheNextOpenRow(board, col)
         board_copy = board.copy()                                        #this copy will allow the AI to
compare between the final states result from it
         dropping_pieces(board_copy, row, col, AI_PIECE)
         new_score = max_min(board_copy, depth-1, alpha, beta, False)[1]  #recursion from the
deepest to the shallowest # and this [1] index is for the second element in the tuple resulting
from the minimax fun to put it in the "new_score"

         # printing_the_utilities_tree
         file.write("The score= "                + str(new_score))
         file.write("\nThe Board State achieving it:\n"  + str(np.flip(board_copy, 0)))
         file.write("\n\n\n\n\n")

         if new_score > value:
            value = new_score
            column = col
         alpha = max(alpha, value)                                        #instead of typing one more if
condition,  this line with the max fun is the same as what the adversarial lec was saying: (( if
value > alpha: alpha= value ))
         if alpha >= beta:                                              #pruning
            break
      return column, value                                               #returning a tuple of only 2
elements

   else:                                                             # the AI computer is thinking as a
minimizier to evaluate the future game
      value = math.inf                                                  #value is the minUtility in the
adversarial search lec algorithm
      column = random.choice(valid_locations)                                  #this fun will return a
random col as an initial choose to begin this algorithm
      for col in valid_locations:
         row = TheNextOpenRow(board, col)
         board_copy = board.copy()                                        #this copy will allow the AI to
compare between the final states result from it
```

```
        dropping_pieces(board_copy, row, col, HUMAN_PIECE)
        new_score = max_min(board_copy, depth-1, alpha, beta, True)[1] #recursion from the
deepest to the shallowest # and this [1] index is for the second element in the tuple resulting
from the minimax fun to put it in the "new_score"

        # printing_the_utilities_tree
        file.write("The score="                 + str(new_score))
        file.write("\nThe Board State achieving it:\n" + str(np.flip(board_copy, 0)))
        file.write("\n\n\n\n" )


        if new_score < value:
           value = new_score
           column = col
        beta = min(beta, value)                                    #instead of typing one more if
condition,  this line with the min fun is the same as what the adversarial lec was saying: (( if
value < beta: beta= value ))
        if alpha >= beta:                                       #pruning
           break
     return column, value                                         #returning a tuple of only 2
elements


#-------------------Calculating The Final Results Between The Two Opponents ---------------------
#---checking if it is a Consecutive pieces sequance
def Consecutive_pieces(array,piece):

   '''
   Horizontal array  len=7 ----------------------> so its enough to loop from 0 to 3------------
-> range(4)--------------> range(7-3)
   vertical   array  len=6 ----------------------> so its enough to loop from 0 to 2------------
-> range(3)--------------> range(6-3)
   the two biggest diagonals array  len=6 --------> so its enough to loop from 0 to 2------------
-> range(3)--------------> range(6-3)
   '''

   # Horizontal, vertical ,and the two biggest diagonals:
   if len(array)==7 or len(array)==6 or len(array)==5:

      for i in range(len(array)-3):
            if array[i] == piece and array[i+1] == piece and array[i+2] == piece and array[i+3]
== piece:
               return True

   #the two small diagonals:
   elif len(array)==4:
      if array[0] == piece and array[1] == piece and array[2] == piece and array[3] == piece:
         return True




#---the two sums of the winning moves
def totalScore(piece ,board):

   global AI_wins,Human_wins
#Golbally declaration
#------Horizontal total Score
   for r in range(total_rows):
      row_array = [int(i) for i in list(board[r, :])]

      if piece== AI_PIECE    and row_array.count(piece) >= 4 and
Consecutive_pieces(row_array,piece):    AI_wins+=1         ;print('AI Horizontal Winning Moves
From Row Num='    ,end=''); print_cyan( r); print('are:'); print_cyan(row_array)
      if piece== HUMAN_PIECE and row_array.count(piece) >= 4 and
Consecutive_pieces(row_array,piece):    Human_wins+=1       ;print('Human Horizontal Winning
Moves From Row Num=',end=''); print_cyan( r); print('are:'); print_cyan(row_array)
```

```
      row_array.clear()
#just to make sure the program will overwrite on it ... if we don't write this line i think it
will do the same thing by itself



#--------vertical total Score
   for c in range(total_columns):
      col_array = [int(i) for i in list(board[:, c])]

      if piece == AI_PIECE   and col_array.count(piece) >= 4 and
Consecutive_pieces(col_array,piece):    AI_wins += 1        ;print('AI vertical Winning Moves
From Column num='   ,end=''); print_cyan(c); print('are:'); print_cyan(col_array)
      if piece == HUMAN_PIECE and col_array.count(piece) >= 4 and
Consecutive_pieces(col_array,piece):   Human_wins += 1     ;print('Human vertical Winning Moves
From Column num=' ,end=''); print_cyan(c); print('are:'); print_cyan(col_array)
      col_array.clear()
#just to make sure the program will overwrite on it ... if we don't write this line i think it
will do the same thing by itself


#--------Positively-slopped diagonal total score((my focused window will go up )):
   #starting from the first row and going upward diagonally:
   for c in range(1,total_columns-3):
#c=0 will be looped in the below and here i will loop c=1,2,3 only because no winning moves can
start after that column
      r=0
      j=c-1
      '''
      j=c-1 ,because:
      ============================
      if c == 1: j = 0
      if c == 2: j = 1
      if c == 3: j = 2
      '''
      window = [board[r+i][c+i] for i in range(total_rows - j)]
#total_rows - j = 6, 5,4 .....as j=0,1,2 so it will loop all valid +ve diagonals

      if piece == AI_PIECE     and window.count(piece) >= 4 and Consecutive_pieces(window,piece):
AI_wins += 1   ;print('AI winning +ve slopped-diagonal starting from the first row and going
upward'); print_cyan(window)
      if piece == HUMAN_PIECE   and window.count(piece) >= 4 and
Consecutive_pieces(window,piece):    Human_wins += 1    ;print('Human winning +ve slopped-
diagonal starting from the first row and going upward'); print_cyan(window)

      window.clear()
# just to make sure the program will overwrite on it ... if we don't write this line i think it
will do the same thing by itself



   # starting from the first column and going upward diagonally:
   for r in range(total_rows-3):
#r=0,1,2  because no winning moves can start after that row
      c=0
      j=r
      '''
      j=r ,because:
      ============================
      if r == 0: j = 0
      if r == 1: j = 1
      if r == 2: j = 2
      '''
      window = [board[r+i][c+i] for i in range(total_rows - j)]
#total_rows - j = 6, 5,4 .....as j=0,1,2 so it will loop all valid +ve diagonals

      if piece == AI_PIECE     and window.count(piece) >= 4 and Consecutive_pieces(window,piece):
AI_wins += 1   ;print('AI winning +ve slopped-diagonal starting from the first column and going
```

```
upward'); print_cyan(window)
      if piece == HUMAN_PIECE   and window.count(piece) >= 4 and
Consecutive_pieces(window,piece):    Human_wins += 1    ;print('Human winning +ve slopped-
diagonal starting from the first column and going upward'); print_cyan(window)

      window.clear()
# just to make sure the program will overwrite on it ... if we don't write this line i think it
will do the same thing by itself



#------------ Negatively-slopped diagonal score ((my focused window will go from up to down ))
   #starting from the most top row and going downward diagonally:
   for c in range(1,total_columns-3):
#c=0 will be looped in the below and here i will loop c=1,2,3 only because no winning moves can
start after that column
      j=c-1
      '''
      j=c-1 ,because:
      ============================
      if c == 1: j = 0
      if c == 2: j = 1
      if c == 3: j = 2
      '''

      window = [board[total_rows-1 -i][c+i] for i in range(total_rows - j)]
#(((total_rows-1 == the most top one))) #total_rows - j = 6, 5,4 .....as j=0,1,2 so it will loop
all valid -ve diagonals

      if piece == AI_PIECE       and window.count(piece) >= 4 and
Consecutive_pieces(window,piece):    AI_wins += 1    ; print('AI winning negatively slopped-
diagonal starting from  most top row and going downward'); print_cyan(window)
      if piece == HUMAN_PIECE    and window.count(piece) >= 4 and
Consecutive_pieces(window,piece):   Human_wins += 1 ; print('Human winning negatively slopped
diagonal starting from  most top row and going downward'); print_cyan(window)

      window.clear()
# just to make sure the program will overwrite on it ... if we don't write this line i think it
will do the same thing by itself



   # starting from the top of the first column and going downward diagonally:
   for r in range(total_rows - 1,total_rows-2,-1):
# r=5,4,3 will be looped  only because no winning moves can start after that column
      if r == 5: j = 0
      if r == 4: j = 1
      if r == 3: j = 2
      c=0
      window = [board[r - i][c + i] for i in range(total_rows - j)]
#total_rows - j = 6, 5,4 .....as j=0,1,2 so it will loop all valid -ve diagonals

      if piece == AI_PIECE        and window.count(piece) >= 4 and
Consecutive_pieces(window,piece):    AI_wins += 1     ;print('AI winning negatively slopped
diagonal starting from the first column and going downward'); print_cyan(window)
      if piece == HUMAN_PIECE       and window.count(piece) >= 4 and
Consecutive_pieces(window,piece):   Human_wins += 1 ;print('Human winning negatively slopped
diagonal starting from the first column and going downward '); print_cyan(window)

      window.clear()
# just to make sure the program will overwrite on it ... if we don't write this line i think it
will do the same thing by itself



   return AI_wins,Human_wins
```

```
#----------------------------------------GUI-------------------------------------------------

#the Dimensions
squareLen  =  100                                #the length of only one side of a square
width      =  total_columns * squareLen
height     =  (total_rows+1) * squareLen

size       =  (width, height)                      #the dataStruct here is a tuple
Radius     =  int((squareLen/2) - 10)               #Radius is half the diameter so it is equal to
squareLen/2 and -10 to make the circle smaller to fit inside the square
screen     =  pygame.display.set_mode(size)              #I found this fun in this link:
https://www.pygame.org/docs/ref/display.html

def DrawBoard(board):

   for c in range(total_columns):
      for r in range(total_rows):
                                                                                          #the
referances of the coming two lines methods and class are in this link:

#https://www.pygame.org/docs/ref/draw.html
                                                                                          #the
first two points are the x y pixels positions and the second two points indicates the dimensions

#((squareLen /2)) is just a little offset from ((c*squareLen)) in order not to overlap shapes
above each other
         pygame.draw.rect(screen, 'sea shell', (c * squareLen, r * squareLen + squareLen,
squareLen, squareLen))
         pygame.draw.circle(screen, 'black', (int(c * squareLen + squareLen /2), int(r *
squareLen + squareLen + squareLen/2)), Radius)

   for c in range(total_columns):
      for r in range(total_rows):

         if board[r][c] == HUMAN_PIECE:
            pygame.draw.circle(screen, 'aqua marine', (int(c * squareLen + squareLen / 2), height
- int(r * squareLen + squareLen/2)), Radius)
         elif board[r][c] == AI_PIECE:
            pygame.draw.circle(screen, 'dark gray', (int(c * squareLen + squareLen / 2), height -
int(r * squareLen + squareLen/2)), Radius)

   pygame.display.update()
#this will only be used when the fun is called but the below ""pygame.display.update()"" is the
main one in my code , I found this in this link:  https://www.pygame.org/docs/ref/display.html


#--------------------the Driver of the Game Entrance GUI
#initialization
pruning_Button= False
minmax_Button =    False

depth_choice  =0
#the defaut depth is =0  if the user didin't press on anything

pushButton_dep1=0
pushButton_dep2=0
pushButton_dep3=0
pushButton_dep4=0
pushButton_dep5=0
pushButton_dep6=0

#the Game EntranceWindow
class introWindow(QtWidgets.QWidget,Ui_Form):
   def __init__(self):
```

```
        QtWidgets.QWidget.__init__(self)
        self.setupUi(self)

        # MyMinmax Button Event
        self.pushButton.clicked.connect(self.MinmaxGui)

        # MinMax-Pruning Button Event
        self.pushButton_2.clicked.connect(self.Pruning_Gui)

        #depth buttons
        self.pushButton_dep1.clicked.connect(self.depth1)
        self.pushButton_dep2.clicked.connect(self.depth2)
        self.pushButton_dep3.clicked.connect(self.depth3)
        self.pushButton_dep4.clicked.connect(self.depth4)
        self.pushButton_dep5.clicked.connect(self.depth5)
        self.pushButton_dep6.clicked.connect(self.depth6)


    # Algorithm-type Buttons to link the Welcoming-Gui with my pygame-Gui
    def MinmaxGui(self):
        global minmax_Button,pruning_Button
        minmax_Button  =True
        pruning_Button  = False
        return minmax_Button



    def Pruning_Gui(self):
        global pruning_Button,minmax_Button
        pruning_Button  = True
        minmax_Button  =False
        return pruning_Button

    # depth buttons
    def depth1(self):
        global depth_choice
        depth_choice = 1
        return depth_choice

    def depth2(self):
        global depth_choice
        depth_choice = 2
        return depth_choice

    def depth3(self):
        global depth_choice
        depth_choice = 3
        return depth_choice

    def depth4(self):
        global depth_choice
        depth_choice = 4
        return depth_choice

    def depth5(self):
        global depth_choice
        depth_choice = 5
        return depth_choice

    def depth6(self):
        global depth_choice
        depth_choice = 6
        return depth_choice


introApp=QtWidgets.QApplication(sys.argv)
EntranceWindow=introWindow()
#Entrance == Welcomming window
```

```
EntranceWindow.show()


minmax_Button  =EntranceWindow.pushButton.clicked.connect(EntranceWindow.MinmaxGui)
pruning_Button =EntranceWindow.pushButton_2.clicked.connect(EntranceWindow.Pruning_Gui)


pushButton_dep1=EntranceWindow.pushButton_dep1.clicked.connect(EntranceWindow.depth1)
pushButton_dep2=EntranceWindow.pushButton_dep2.clicked.connect(EntranceWindow.depth2)
pushButton_dep3=EntranceWindow.pushButton_dep3.clicked.connect(EntranceWindow.depth3)
pushButton_dep4=EntranceWindow.pushButton_dep4.clicked.connect(EntranceWindow.depth4)
pushButton_dep5=EntranceWindow.pushButton_dep5.clicked.connect(EntranceWindow.depth5)
pushButton_dep6=EntranceWindow.pushButton_dep6.clicked.connect(EntranceWindow.depth6)

if      pushButton_dep1!=0:         depth_choice = pushButton_dep1
elif pushButton_dep2!=0:         depth_choice = pushButton_dep2
elif pushButton_dep3!=0:         depth_choice = pushButton_dep3
elif pushButton_dep4!=0:         depth_choice = pushButton_dep4
elif pushButton_dep5!=0:         depth_choice = pushButton_dep5
elif pushButton_dep6!=0:         depth_choice = pushButton_dep6




'''
sys.exit(introApp.exec())
this above line is a comment because the only possible ways to exit my program is:
when the game is over or when the user press the X button to exit the pygame Gui ....
so pressing the X button of the Welcoming Gui Window will only close it and would not exit the
whole game
'''



#---------------------------------------------The Main Program Game ----------------------------

board = CreatingtheBoard()
print_board(board)
game_over = check_terminal_nodes(board)                        #gameover==false in the beginning

pygame.init()                                                 #initialize all imported pygame modules
DrawBoard(board)                                               #calling the GUI board
pygame.display.update()



font = pygame.font.SysFont("segoescript",45 )                  #look here to choose the desired
font: https://www.codegrepper.com/code-examples/python/pygame.font
turn=HUMAN                                                   #I will start with the (human turn) to make
the game beginning more friendly for the user



while not game_over:
   for event in pygame.event.get():                            #Loop to keep the GUI open until the
user press the exit X button
      if event.type == pygame.QUIT:
         sys.exit()
         sys.exit(introApp.exec())                           #so if you press the X button to exit
the pygame the two windows will shut down but if u exit the welcome gui the pygame gui will still
be opened


      if event.type == pygame.MOUSEMOTION:                     # here:
https://www.pygame.org/docs/ref/event.html
         pygame.draw.rect(screen, 'black',(0,0, width,squareLen))    #to make the top row black
when scrolling or moving the mouse on the gui screen
```

```
            x_position = event.pos[0]
            if turn == HUMAN:
                pygame.draw.circle(screen, 'aqua marine', (x_position, int(squareLen/2)), Radius)

        pygame.display.update()

##------------------------------ taking the human input
        if event.type == pygame.MOUSEBUTTONDOWN:
            pygame.draw.rect(screen,'black', (0,0, width, squareLen))      #to return the top row to
be black again
            if turn == HUMAN:
                x_position = event.pos[0]
                col = int(math.floor(x_position/squareLen))              #notice that the x pixels
positions are nums between 0 t0 100 or 100 to 200 so its division by the ((square length =100 ))
will give me the required columns that the user wanted to choose by clicking on it

                if check_valid_locations(board, col):
                    row = TheNextOpenRow(board, col)
                    dropping_pieces(board, row, col, HUMAN_PIECE)

                    #updating & printing the board after each move
                    print_board(board)
                    DrawBoard(board)

                    # switching turns
                    turn += 1
                    turn = turn % 2




##------------------------------ taking the AI input
    if turn == AI and not game_over:
        K=depth_choice                                                                          #depth
=K  #the  method will return a tuple consists of two elements, the first one is the current state
or the current col position , and the second one is its score or utility
        print('your picked depth=', depth_choice)
        #------The Algorithm Button that the user choose to press:
        #Minmax_Without_Pruning:
        if minmax_Button==True and pruning_Button== False:

            #Time
            start = timeit.default_timer()                                                  # This will
return the default time before executing the next line
            col ,minimax_score= maxmin_WithoutPruning(board, K, True)
            stop = timeit.default_timer()                                                   # This will
return the default time after executing the above previous line

            print('Time During Using Minmax Without Pruning Algorithm=',    stop-start,' seconds' )
            Minmax_time +=( stop - start)                                                   #sum = sum + time


            #Nodes_Expanded
            print('Num of Nodes Expanded To Take This Move only=', MinmaxNodes_expanded    ,'
Nodes')
            total_minmaxNodes    = total_minmaxNodes + MinmaxNodes_expanded                      #sum =
sum + current expanded nodes
            MinmaxNodes_expanded=0




        #Minmax_with_Prunning
        elif pruning_Button==True and minmax_Button== False:

            #Time
            start = timeit.default_timer()                                                   # This will
return the default time before executing the next line
```

```
        col ,minimax_score= max_min(board, K, -math.inf, math.inf, True)                #alpha=-
infinity  ,and  beta=+infinity
#alpha=-infinity  ,and  beta=+infinity
        stop = timeit.default_timer()                                                   # This will
return the default time after executing the above previous line

        print('Time During Using Minmax With Alph Beta Pruning Algorithm:', stop-start,'
seconds')
        Pruning_time +=( stop - start)                                                  #sum = sum + time


        #tree_printing
        pass

        #Nodes_Expanded
        print('Num of Nodes Expanded To Take This Move only=', PruningNodes_expanded,' Nodes')
        total_PruningNodes  = total_PruningNodes + PruningNodes_expanded                #sum =
sum + current expanded nodes
        PruningNodes_expanded=0




    elif pruning_Button==False and minmax_Button== False:
        print('You Didn\'t pick any AI algorithm for me so the game will be so easy like a piece
of cake')



    #the user pressed the two options and both are "True":
    else:
        print('Wrong Choice! PLZ Press Only One Button From The Push Buttons On The Welcoming
GUI Window')


    #----------Actions:
    if check_valid_locations(board, col):
        row = TheNextOpenRow(board, col)
        dropping_pieces(board, row, col, AI_PIECE)

        #updating & printing the board after each move
        print_board(board)
        DrawBoard(board)

        #switching turns
        turn += 1
        turn = turn % 2







#----------When the Game is over
  if check_terminal_nodes(board):

    pygame.draw.rect(screen, 'black', (0, 0, width, squareLen))                 # to return the
top row to be black again
    label = font.render("Nice Moves!", 1,'White')                               # one 1 is the first
row this means that the label will be in the top raw of the GUI
    screen.blit(label, (200,15))                                            # .blit not .blits because
blits doesn't work well with me , and #(45,15) is the label position, so its goal is to make the
label appear on my laptop screen as what stack overflow advise me here:
https://stackoverflow.com/questions/37800894/what-is-the-surface-blit-function-in-pygame-what-
does-it-do-how-does-it-work
    DrawBoard(board)
```

```
      pygame.time.wait(750)                                                    # don't automatically
exit until few seconds pass

      print('-----------------------------------------------The program will print (in the few
below lines) the positions that score came from---------------------------------------')

      #-------who wins?
      AI_wins    = 0                                                    #deleting this one line would
not effect on any thing but it is just to clearing it again and making sure it is equal 0 3shan
3'lbtnee fl 2tls3 mn al 4orba b2a :D
      AI_wins       = totalScore(AI_PIECE,  board)[0]
      Human_wins = 0                                                    # this one line will make me
avoid doubling the "Human_Wins" value
      Human_wins = totalScore(HUMAN_PIECE,  board)[1]


      ##-------printing total Num of Nodes Expanded & Time

print('=================================================================================================
=================================================================================================
===')
      if  minmax_Button== True:

         print("-----------------------------------------------Total Nodes Using MinMax WITHOUT α-
β Pruning= ", end='')
         print_cyan_without_NewLine(total_minmaxNodes); print(' Nodes')

         print("-----------------------------------------------Total Time Using MinMax WITHOUT α-β
Pruning = ", end='')
         print_cyan_without_NewLine(Minmax_time)    ; print(' seconds')


      if pruning_Button ==True:

         print("-----------------------------------------------Total Nodes Using MinMax With α-β
Pruning=", end='')
         print_cyan_without_NewLine(total_PruningNodes);print(' Nodes')


         print("-----------------------------------------------Total Time Using MinMax with α-β
Pruning= ", end='')
         print_cyan_without_NewLine(Pruning_time); print(' seconds')


      #-------printing who wins on the Console
      print("-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
---------")
      print("-----------------------------------------------My AI Total Score= ",end='')
      print_cyan(AI_wins)

      print("-----------------------------------------------Human Total score= ",end='')
      print_cyan( Human_wins)


print('=================================================================================================
=================================================================================================
===')
      print('-----------------------------------------------------Your GUI Gonna Shut Down
Automatically After 5 Seconds, Just Wait or Press the X Button of the "PyGame-GUI" to Exit ------
---------------------------------')
      print('-------------------------------------------------------------------------------End
Of The program-------------------------------------------\n------------------
-------------------------------------------------------------bye..bye!----------------------
-----------------------------------------')


      # -------printing who wins on the GUI Screen
```

```
      if AI_wins > Human_wins:

         pygame.draw.rect(screen, 'black', (0, 0, width, squareLen))
# to return the top row to be black again
         font = pygame.font.SysFont("segoescript", 40,bold=False)
# look here to choose the desired font: https://www.codegrepper.com/code-
examples/python/pygame.font
         label = font.render("Game Over! Asmaa's AI Wins", 1,'White')
# one 1 is the first row this means that the label will be in the top raw of the GUI
         screen.blit(label, (50, 15))

      elif AI_wins < Human_wins:

         pygame.draw.rect(screen, 'black', (0, 0, width, squareLen))
# to return the top row to be black again
         label = font.render("Congrats! You Win", 1,'White')
# one 1 is the first row this means that the label will be in the top raw of the GUI
         screen.blit(label, (130, 15))

      else:
         #AI_wins == Human_wins
         pygame.draw.rect(screen, 'black', (0, 0, width, squareLen))
# to return the top row to be black again
         label = font.render("This Is A Draw!", 1,'white')
# one 1 is the first row this means that the label will be in the top raw of the GUI
         screen.blit(label, (130, 15))



      DrawBoard(board)
#updating the board after knowing who wins
      pygame.time.wait(2750)
# don't automatically exit until few seconds pass

      #--------------Finall,shutting down w Al-hamd l Allah^^
      pygame.draw.rect(screen, 'black', (0, 0, width, squareLen))
# to return the top row to be black again
      font = pygame.font.SysFont("segoescript",20, bold=True)
# look here to choose the desired font: https://www.codegrepper.com/code-
examples/python/pygame.font
      label = font.render("Program Gonna Automatically Shut Down After 5 Sec!", 1,'white')
# one 1 is the first row this means that the label will be in the top raw of the GUI
      screen.blit(label, (50,35))
# .blit not .blits because blits doesn't work well with me , and #(45,15) is the label position,
so its goal is to make the label appear on my laptop screen as what stack overflow advise me
here: https://stackoverflow.com/questions/37800894/what-is-the-surface-blit-function-in-pygame-
what-does-it-do-how-does-it-work
      DrawBoard(board)
      pygame.time.wait(5000)
# don't automatically exit until 2.5 seconds pass
      game_over= True

#Yeah Done!.. Thanks to Allah^^
```

3.The “GUI.py”:

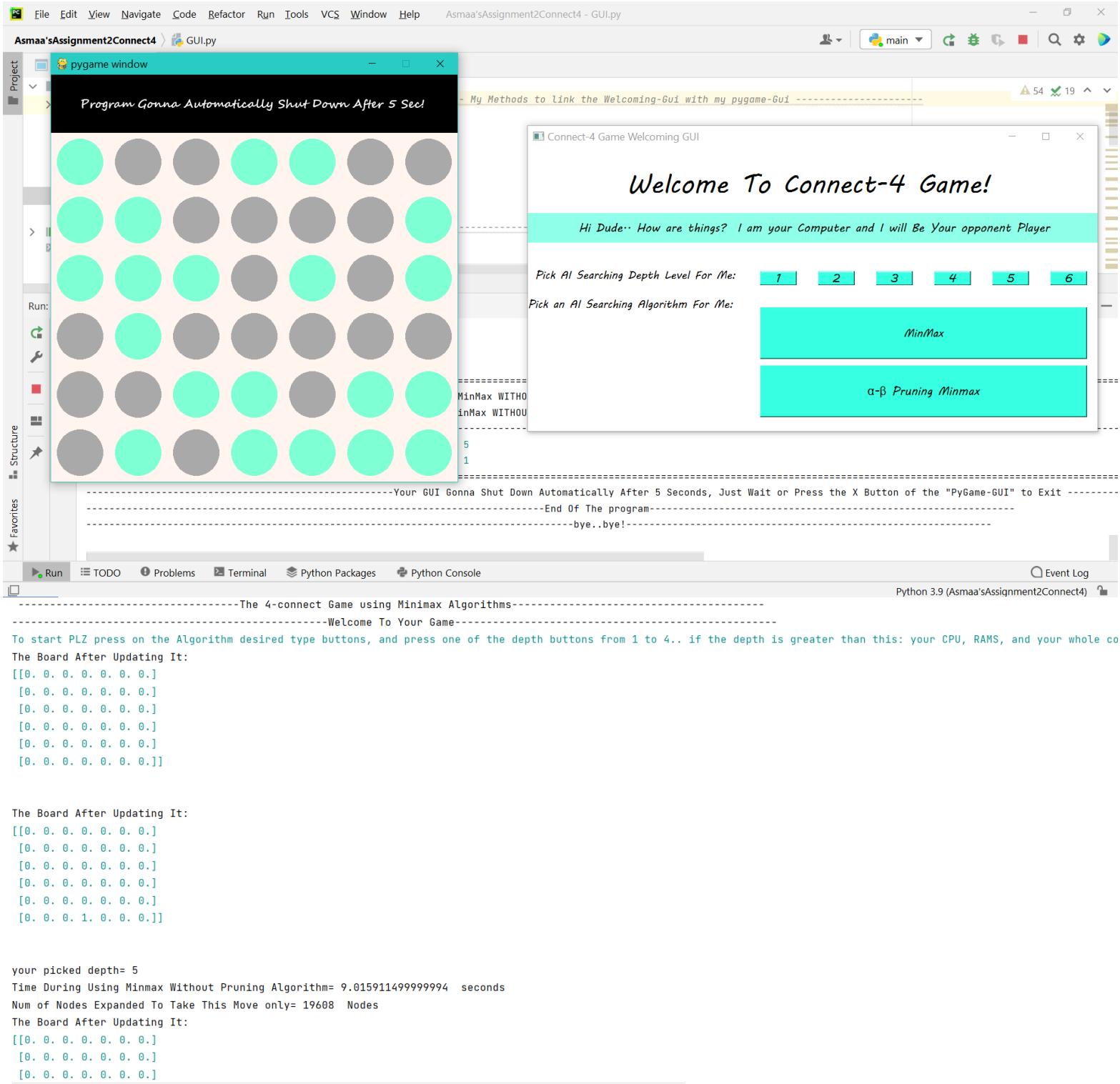
```
# Asmaa Gamal
#Assignment 2
# Electronics & Communications department
#Using Minimax Algorithm to Make Connect 4 AI Agent

#------------------------------ preparing for the GUI stage ----------------------------------

#-----------------------------------------using QT-------------------------------------------

from PyQt6 import QtCore, QtGui, QtWidgets

class Ui_Form(object):
#----------------- My Methods to link the Welcoming-Gui with my pygame-Gui ---------------------
    '''
    def MinmaxGui(self):
       return True

    def Pruning_Gui(self):
        return True
     '''
#-------------------------------------------------------------------------------------------------
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(785, 390)
        Form.setStyleSheet("background-color: rgb(255, 255, 255)\n""")
        self.pushButton = QtWidgets.QPushButton(Form)
        self.pushButton.setGeometry(QtCore.QRect(320, 220, 450, 71))
        font = QtGui.QFont()
        font.setFamily("MV Boli")
        font.setPointSize(11)
        self.pushButton.setFont(font)
        self.pushButton.setStyleSheet("QPushButton{\n""Background:rgb(55, 255, 225)\n""\n""}")
        self.pushButton.setObjectName("pushButton")
        ''''
        # ------------------------------MyMinmax Button Event -----------------------------
        self.pushButton.clicked.connect(self.MinmaxGui)
        # ---------------------------------------------------------------------------------
        '''
        self.pushButton_2 = QtWidgets.QPushButton(Form)
        self.pushButton_2.setGeometry(QtCore.QRect(320, 300, 450, 71))
        font = QtGui.QFont()
        font.setFamily("MV Boli")
        font.setPointSize(12)
        self.pushButton_2.setFont(font)
        self.pushButton_2.setStyleSheet("QPushButton{\n""Background:rgb(55, 255, 225)\n""\n""}")
        self.pushButton_2.setObjectName("pushButton_2")

        '''
        # ------------------------My MinMax-Pruning Button Event -----------------------------
        self.pushButton_2.clicked.connect(self.Pruning_Gui)
        # ---------------------------------------------------------------------------------
        '''

        self.label = QtWidgets.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(0, 10, 800, 81))
        font = QtGui.QFont()
        font.setFamily("MV Boli")
        font.setPointSize(26)
        self.label.setFont(font)
        self.label.setStyleSheet("")
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(Form)
        self.label_2.setGeometry(QtCore.QRect(10, 160, 300, 31))
        font = QtGui.QFont()
```

```
        font.setFamily("MV Boli")
        font.setPointSize(11)
        self.label_2.setFont(font)
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(Form)
        self.label_3.setGeometry(QtCore.QRect(0, 200, 300, 31))
        font = QtGui.QFont()
        font.setFamily("MV Boli")
        font.setPointSize(11)
        self.label_3.setFont(font)
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(Form)
        self.label_4.setGeometry(QtCore.QRect(0, 90, 800, 41))
        font = QtGui.QFont()
        font.setFamily("MV Boli")
        font.setPointSize(12)
        self.label_4.setFont(font)
        self.label_4.setStyleSheet("QLabel{\n""Background:rgb(169, 255, 248)rgb(144, 255,
233)\n""\n""}")
        self.label_4.setObjectName("label_4")



        ##-------pushButtons of depths
        #pushButton_dep1
        self.pushButton_dep1 = QtWidgets.QPushButton(Form)
        self.pushButton_dep1.setGeometry(QtCore.QRect(320, 170, 50, 20))
        font = QtGui.QFont()
        font.setFamily("MV Boli")
        font.setPointSize(12)
        self.pushButton_dep1.setFont(font)
        self.pushButton_dep1.setStyleSheet("QPushButton{\n""Background:rgb(55, 255,
225)\n""\n""}")
        self.pushButton_dep1.setObjectName("pushButton_dep1")


        # pushButton_dep2
        self.pushButton_dep2 = QtWidgets.QPushButton(Form)
        self.pushButton_dep2.setGeometry(QtCore.QRect(400, 170, 50, 20))
        font = QtGui.QFont()
        font.setFamily("MV Boli")
        font.setPointSize(12)
        self.pushButton_dep2.setFont(font)
        self.pushButton_dep2.setStyleSheet("QPushButton{\n""Background:rgb(55, 255,
225)\n""\n""}")
        self.pushButton_dep2.setObjectName("pushButton_dep2")


        ##pushButton_dep3
        self.pushButton_dep3 = QtWidgets.QPushButton(Form)
        self.pushButton_dep3.setGeometry(QtCore.QRect(480, 170, 50, 20))
        font = QtGui.QFont()
        font.setFamily("MV Boli")
        font.setPointSize(12)
        self.pushButton_dep3.setFont(font)
        self.pushButton_dep3.setStyleSheet("QPushButton{\n""Background:rgb(55, 255,
225)\n""\n""}")
        self.pushButton_dep3.setObjectName("pushButton_dep3")


        ##pushButton_dep4
        self.pushButton_dep4 = QtWidgets.QPushButton(Form)
        self.pushButton_dep4.setGeometry(QtCore.QRect(560, 170, 50, 20))
        font = QtGui.QFont()
        font.setFamily("MV Boli")
        font.setPointSize(12)
        self.pushButton_dep4.setFont(font)
        self.pushButton_dep4.setStyleSheet("QPushButton{\n""Background:rgb(55, 255,
```

```
225)\n""\n""}")
        self.pushButton_dep4.setObjectName("pushButton_dep4")



        ##pushButton_dep5
        self.pushButton_dep5 = QtWidgets.QPushButton(Form)
        self.pushButton_dep5.setGeometry(QtCore.QRect(640, 170, 50, 20))
        font = QtGui.QFont()
        font.setFamily("MV Boli")
        font.setPointSize(12)
        self.pushButton_dep5.setFont(font)
        self.pushButton_dep5.setStyleSheet("QPushButton{\n""Background:rgb(55, 255,
225)\n""\n""}")
        self.pushButton_dep5.setObjectName("pushButton_dep5")



        ##pushButton_dep6
        self.pushButton_dep6 = QtWidgets.QPushButton(Form)
        self.pushButton_dep6.setGeometry(QtCore.QRect(720, 170, 50, 20))
        font = QtGui.QFont()
        font.setFamily("MV Boli")
        font.setPointSize(12)
        self.pushButton_dep6.setFont(font)
        self.pushButton_dep6.setStyleSheet("QPushButton{\n""Background:rgb(55, 255,
225)\n""\n""}")
        self.pushButton_dep6.setObjectName("pushButton_dep6")



        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)



    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Connect-4 Game Welcoming GUI"))
        self.pushButton.setText(_translate("Form", "MinMax"))
        self.pushButton_2.setText(_translate("Form", "α-β Pruning Minmax"))
        self.label.setText(_translate("Form", "        Welcome To Connect-4 Game!"))
        self.label_2.setText(_translate("Form", "Pick AI Searching Depth Level For Me:"))
        self.label_3.setText(_translate("Form", "Pick an AI Searching Algorithm For Me:"))
        self.label_4.setText(_translate("Form", "         Hi Dude.. How are things?  I am your
Computer and I will Be Your opponent Player"))

        ##------pushButtons of depths
        self.pushButton_dep1.setText(_translate("Form", "1"))
        self.pushButton_dep2.setText(_translate("Form", "2"))
        self.pushButton_dep3.setText(_translate("Form", "3"))
        self.pushButton_dep4.setText(_translate("Form", "4"))
        self.pushButton_dep5.setText(_translate("Form", "5"))
        self.pushButton_dep6.setText(_translate("Form", "6"))


#------------------------ using colors in printing on the console screen code-------------------


ANSI_RESET = "\u001B[0m"
ANSI_CYAN = "\u001B[36m"

def print_cyan(msg):
    print(f"{ANSI_CYAN}{msg}{ANSI_RESET}")

def print_cyan_without_NewLine(msg):
    print(f"{ANSI_CYAN}{msg}{ANSI_RESET}",end='')
```

Screenshots Of Some Runs

1. Minmax:




```
----------The 4-connect Game using Minimax Algorithms----------
----------Welcome To Your Game----------
To start PLZ press on the Algorithm desired type buttons, and press one of the depth buttons from 1 to 4.. if the depth is greater than this: your CPU, RAMS, and your whole co
The Board After Updating It:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]]


The Board After Updating It:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]]


your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 9.015911499999994  seconds
Num of Nodes Expanded To Take This Move only= 19608  Nodes
The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
```

```
[2, 1, 2, 2, 2, 2, 2]
AI Horizontal Winning Moves From Row Num=4
are:
[1, 1, 2, 2, 2, 2, 1]
AI vertical Winning Moves From Column num=5
are:
[1, 1, 2, 2, 2, 2]
AI winning +ve slopped-diagonal starting from the first column and going upward
[2.0, 2.0, 2.0, 2.0, 2.0, 2.0]
AI winning negatively slopped-diagonal starting from  most top row and going downward
[2.0, 2.0, 2.0, 2.0, 1.0, 1.0]
Human Horizontal Winning Moves From Row Num=0
are:
[2, 1, 2, 1, 1, 1, 1]
================================================================================================================================================================================================
------------------------------------------------Total Nodes Using MinMax WITHOUT α-β Pruning= 222438 Nodes
------------------------------------------------Total Time Using MinMax WITHOUT α-β Pruning = 107.7000546999999 seconds
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------My AI Total Score= 5
------------------------------------------------Human Total score= 1
================================================================================================================================================================================================
----------------------------------------------------Your GUI Gonna Shut Down Automatically After 5 Seconds, Just Wait or Press the X Button of the "PyGame-GUI" to Exit --------
-------------------------------------------------------------------------------End Of The program-----------------------------------------------------------------------
-----------------------------------------------------------------------------------------bye..bye!-----------------------------------------------------------------

Process finished with exit code 0
```

```
Time During Using Minmax Without Pruning Algorithm= 0.0004564999999843167  seconds
Num of Nodes Expanded To Take This Move only= 2  Nodes
The Board After Updating It:
[[1. 2. 2. 1. 1. 2. 2.]
 [1. 1. 2. 2. 2. 2. 1.]
 [1. 1. 1. 2. 1. 2. 1.]
 [2. 1. 2. 2. 2. 2. 2.]
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]


------------------------------------------------The program will print (in the few below lines) the positions that score came from---------------------------------------
AI Horizontal Winning Moves From Row Num=2
are:
[2, 1, 2, 2, 2, 2, 2]
AI Horizontal Winning Moves From Row Num=4
are:
[1, 1, 2, 2, 2, 2, 1]
AI vertical Winning Moves From Column num=5
are:
[1, 1, 2, 2, 2, 2]
AI winning +ve slopped-diagonal starting from the first column and going upward
[2.0, 2.0, 2.0, 2.0, 2.0, 2.0]
AI winning negatively slopped-diagonal starting from  most top row and going downward
[2.0, 2.0, 2.0, 2.0, 1.0, 1.0]
Human Horizontal Winning Moves From Row Num=0
are:
[2, 1, 2, 1, 1, 1, 1]
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

----------The 4-connect Game using Minimax Algorithms----------
----------Welcome To Your Game----------
To start PLZ press on the Algorithm desired type buttons, and press one of the depth buttons from 1 to 4.. if the depth is greater than this: your CPU, RAMS, and your whole computer will suffer and start to be slow
The Board After Updating It:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]]

The Board After Updating It:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 9.015911499999994 seconds
Num of Nodes Expanded To Take This Move only= 19608 Nodes
The Board After Updating It:
[[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. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 2. 1. 0. 0. 0.]]

The Board After Updating It:
[[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. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 2. 1. 0. 0. 0.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 9.183160399999991 seconds
Num of Nodes Expanded To Take This Move only= 19607 Nodes
The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 2. 1. 0. 0. 0.]]

The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 2. 1. 0. 0. 0.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 9.4209058 seconds
Num of Nodes Expanded To Take This Move only= 19575 Nodes
The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 2. 1. 0. 0. 0.]]

The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 1. 2. 1. 0. 0. 0.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 9.481267500000001 seconds
Num of Nodes Expanded To Take This Move only= 19544 Nodes
The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 2. 2. 0. 0. 0.]
[0. 2. 1. 1. 0. 0. 0.]
[0. 1. 2. 1. 0. 0. 0.]]

The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 1. 2. 2. 0. 0. 0.]
[0. 2. 1. 1. 0. 0. 0.]
[0. 1. 2. 1. 0. 0. 0.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 9.840973899999995 seconds
Num of Nodes Expanded To Take This Move only= 19512 Nodes
The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 1. 2. 2. 0. 0. 0.]
 [0. 2. 1. 1. 0. 0. 0.]
 [0. 1. 2. 1. 0. 0. 0.]]

The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 1. 2. 2. 0. 0. 0.]
 [0. 2. 1. 1. 0. 0. 0.]
 [0. 1. 2. 1. 1. 0. 0.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 9.199437200000006 seconds
Num of Nodes Expanded To Take This Move only= 19127 Nodes
The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 1. 2. 2. 0. 0. 0.]
 [0. 2. 1. 1. 2. 0. 0.]
 [0. 1. 2. 1. 1. 0. 0.]]

The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 1. 2. 2. 0. 0. 0.]
 [0. 2. 1. 1. 2. 0. 0.]
 [0. 1. 2. 1. 1. 1. 0.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 9.387496199999987 seconds
Num of Nodes Expanded To Take This Move only= 19126 Nodes
The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 1. 2. 2. 0. 0. 0.]
 [0. 2. 1. 1. 2. 0. 0.]
 [2. 1. 2. 1. 1. 1. 0.]]

The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 1. 2. 2. 0. 0. 0.]
 [0. 2. 1. 1. 2. 0. 0.]
 [2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 9.500249299999979 seconds
Num of Nodes Expanded To Take This Move only= 19126 Nodes
The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 1. 2. 2. 0. 0. 0.]
 [2. 2. 1. 1. 2. 0. 0.]
 [2. 1. 2. 1. 1. 1. 1.]]

The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 1. 2. 0. 0. 0.]
[0. 1. 2. 2. 0. 0. 0.]
[2. 2. 1. 1. 2. 0. 0.]
[2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 9.373958700000003 seconds
Num of Nodes Expanded To Take This Move only= 18740 Nodes
The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 1. 2. 0. 0. 0.]
[0. 1. 2. 2. 0. 0. 0.]
[2. 2. 1. 1. 2. 0. 0.]
[2. 1. 2. 1. 1. 1. 1.]]

The Board After Updating It:
[[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 1. 2. 0. 0. 0.]
[0. 1. 2. 2. 0. 0. 0.]
[2. 2. 1. 1. 2. 0. 0.]
[2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 4.262252499999988 seconds
Num of Nodes Expanded To Take This Move only= 9004 Nodes
The Board After Updating It:
[[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 0. 0. 0.]
[2. 2. 1. 1. 2. 0. 0.]
[2. 1. 2. 1. 1. 1. 1.]]

The Board After Updating It:
[[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[1. 0. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 0. 0. 0.]
[2. 2. 1. 1. 2. 0. 0.]
[2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 4.108310099999983 seconds
Num of Nodes Expanded To Take This Move only= 8707 Nodes
The Board After Updating It:
[[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[1. 0. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 2. 0. 0.]
[2. 2. 1. 1. 2. 0. 0.]
[2. 1. 2. 1. 1. 1. 1.]]

The Board After Updating It:
[[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[1. 1. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 2. 0. 0.]
[2. 2. 1. 1. 2. 0. 0.]
[2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 4.0731997000000035 seconds
Num of Nodes Expanded To Take This Move only= 8410 Nodes
The Board After Updating It:
[[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 2. 2. 0. 0. 0.]
[1. 1. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 2. 0. 0.]

[2. 2. 1. 1. 2. 0. 0.]
[2. 1. 2. 1. 1. 1. 1.]]

The Board After Updating It:
[[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 2. 2. 0. 0. 0.]
[1. 1. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 2. 0. 0.]
[2. 2. 1. 1. 2. 1. 0.]
[2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 3.4467361999999753 seconds
Num of Nodes Expanded To Take This Move only= 7013 Nodes
The Board After Updating It:
[[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 2. 2. 0. 0. 0.]
[1. 1. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 2. 2. 0.]
[2. 2. 1. 1. 2. 1. 0.]
[2. 1. 2. 1. 1. 1. 1.]]

The Board After Updating It:
[[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 2. 2. 0. 0. 0.]
[1. 1. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 2. 2. 0.]
[2. 2. 1. 1. 2. 1. 1.]
[2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 3.3951907999999946 seconds
Num of Nodes Expanded To Take This Move only= 6986 Nodes
The Board After Updating It:
[[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 2. 2. 0. 0. 0.]
[1. 1. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 2. 2. 2.]
[2. 2. 1. 1. 2. 1. 1.]
[2. 1. 2. 1. 1. 1. 1.]]

The Board After Updating It:
[[0. 0. 0. 1. 0. 0. 0.]
[0. 1. 2. 2. 0. 0. 0.]
[1. 1. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 2. 2. 2.]
[2. 2. 1. 1. 2. 1. 1.]
[2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 2.787451699999991 seconds
Num of Nodes Expanded To Take This Move only= 5690 Nodes
The Board After Updating It:
[[0. 2. 0. 1. 0. 0. 0.]
[0. 1. 2. 2. 0. 0. 0.]
[1. 1. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 2. 2. 2.]
[2. 2. 1. 1. 2. 1. 1.]
[2. 1. 2. 1. 1. 1. 1.]]

The Board After Updating It:
[[0. 2. 0. 1. 0. 0. 0.]
[1. 1. 2. 2. 0. 0. 0.]
[1. 1. 1. 2. 0. 0. 0.]
[2. 1. 2. 2. 2. 2. 2.]
[2. 2. 1. 1. 2. 1. 1.]
[2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 0.963306700000004 seconds

Num of Nodes Expanded To Take This Move only= 2060 Nodes
The Board After Updating It:
[[0. 2. 2. 1. 0. 0. 0.]
 [1. 1. 2. 2. 0. 0. 0.]
 [1. 1. 1. 2. 0. 0. 0.]
 [2. 1. 2. 2. 2. 2. 2.]
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]

The Board After Updating It:
[[1. 2. 2. 1. 0. 0. 0.]
 [1. 1. 2. 2. 0. 0. 0.]
 [1. 1. 1. 2. 0. 0. 0.]
 [2. 1. 2. 2. 2. 2. 2.]
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 0.14751119999999673 seconds
Num of Nodes Expanded To Take This Move only= 328 Nodes
The Board After Updating It:
[[1. 2. 2. 1. 0. 0. 0.]
 [1. 1. 2. 2. 0. 0. 0.]
 [1. 1. 1. 2. 0. 2. 0.]
 [2. 1. 2. 2. 2. 2. 2.]
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]

The Board After Updating It:
[[1. 2. 2. 1. 0. 0. 0.]
 [1. 1. 2. 2. 0. 0. 0.]
 [1. 1. 1. 2. 1. 2. 0.]
 [2. 1. 2. 2. 2. 2. 2.]
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 0.0982140000000129 seconds
Num of Nodes Expanded To Take This Move only= 230 Nodes
The Board After Updating It:
[[1. 2. 2. 1. 0. 0. 0.]
 [1. 1. 2. 2. 2. 0. 0.]
 [1. 1. 1. 2. 1. 2. 0.]
 [2. 1. 2. 2. 2. 2. 2.]
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]

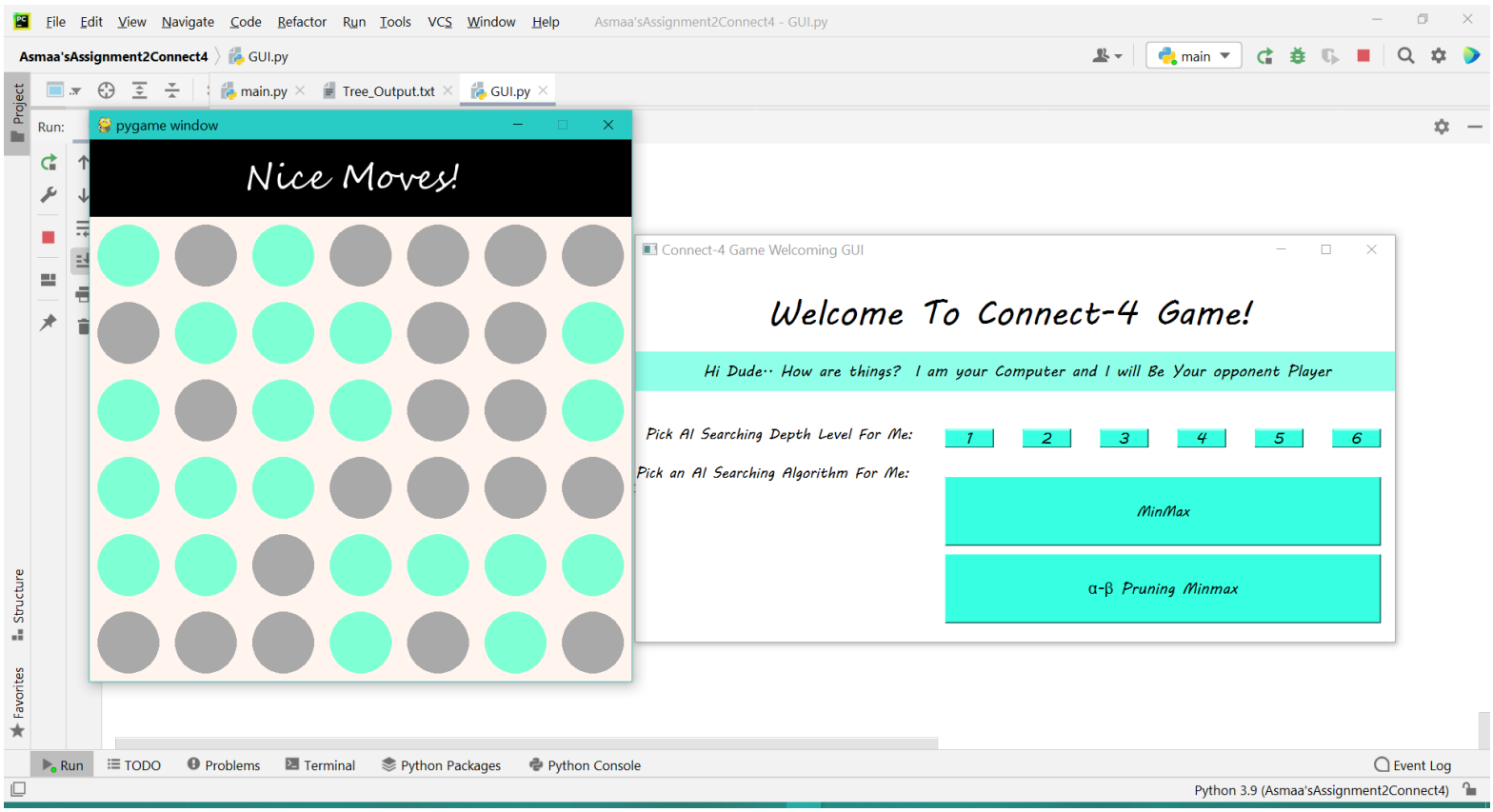
The Board After Updating It:
[[1. 2. 2. 1. 1. 0. 0.]
 [1. 1. 2. 2. 2. 0. 0.]
 [1. 1. 1. 2. 1. 2. 0.]
 [2. 1. 2. 2. 2. 2. 2.]
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]

your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 0.010438900000025342 seconds
Num of Nodes Expanded To Take This Move only= 34 Nodes
The Board After Updating It:
[[1. 2. 2. 1. 1. 0. 0.]
 [1. 1. 2. 2. 2. 2. 0.]
 [1. 1. 1. 2. 1. 2. 0.]
 [2. 1. 2. 2. 2. 2. 2.]
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]

The Board After Updating It:
[[1. 2. 2. 1. 1. 0. 0.]
 [1. 1. 2. 2. 2. 2. 0.]
 [1. 1. 1. 2. 1. 2. 1.]
 [2. 1. 2. 2. 2. 2. 2.]

```
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]


your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 0.0036258999999745356  seconds
Num of Nodes Expanded To Take This Move only= 9  Nodes
The Board After Updating It:
[[1. 2. 2. 1. 1. 2. 0.]
 [1. 1. 2. 2. 2. 2. 0.]
 [1. 1. 1. 2. 1. 2. 1.]
 [2. 1. 2. 2. 2. 2. 2.]
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]


The Board After Updating It:
[[1. 2. 2. 1. 1. 2. 0.]
 [1. 1. 2. 2. 2. 2. 1.]
 [1. 1. 1. 2. 1. 2. 1.]
 [2. 1. 2. 2. 2. 2. 2.]
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]


your picked depth= 5
Time During Using Minmax Without Pruning Algorithm= 0.0004564999999843167  seconds
Num of Nodes Expanded To Take This Move only= 2  Nodes
The Board After Updating It:
[[1. 2. 2. 1. 1. 2. 2.]
 [1. 1. 2. 2. 2. 2. 1.]
 [1. 1. 1. 2. 1. 2. 1.]
 [2. 1. 2. 2. 2. 2. 2.]
 [2. 2. 1. 1. 2. 1. 1.]
 [2. 1. 2. 1. 1. 1. 1.]]


----------The program will print (in the few below lines) the positions that score came from----------
AI Horizontal Winning Moves From Row Num=2
are:
[2, 1, 2, 2, 2, 2, 2]
AI Horizontal Winning Moves From Row Num=4
are:
[1, 1, 2, 2, 2, 2, 1]
AI vertical Winning Moves From Column num=5
are:
[1, 1, 2, 2, 2, 2]
AI winning +ve slopped-diagonal starting from the first column and going upward
[2.0, 2.0, 2.0, 2.0, 2.0, 2.0]
AI winning negatively slopped-diagonal starting from  most top row and going downward
[2.0, 2.0, 2.0, 2.0, 1.0, 1.0]
Human Horizontal Winning Moves From Row Num=0
are:
[2, 1, 2, 1, 1, 1, 1]
================================================================================================================================================
================================================================================
----------Total Nodes Using MinMax WITHOUT a-β Pruning= 222438 Nodes
----------Total Time Using MinMax WITHOUT a-β Pruning = 107.7000546999999 seconds
--------------------------------------------------------------------------------------------------------------------------------------------------------
----------My AI Total Score= 5
----------Human Total score= 1
================================================================================================================================================
================================================================================
----------Your GUI Gonna Shut Down Automatically After 5 Seconds, Just Wait or Press the X Button of the "PyGame-GUI" to Exit ----------
----------End Of The program----------
----------bye..bye!----------

Process finished with exit code 0
```

2. Minmax with α-β pruning:




```
 ----------------------------------The 4-connect Game using Minimax Algorithms-------------------------------------
-----------------------------------------------------Welcome To Your Game-----------------------------------------------------
To start PLZ press on the Algorithm desired type buttons, and press one of the depth buttons from 1 to 4.. if the depth is greater than this: your CPU, RAMS, and your whole com
The Board After Updating It:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]]


The Board After Updating It:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]]


your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 1.4070797000000006  seconds
Num of Nodes Expanded To Take This Move only= 2827  Nodes
The Board After Updating It:
```

```
your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.000598199999998883  seconds
Num of Nodes Expanded To Take This Move only= 2  Nodes
The Board After Updating It:
[[1. 2. 1. 2. 2. 2. 2.]
 [2. 1. 1. 1. 2. 2. 1.]
 [1. 2. 1. 1. 2. 2. 1.]
 [1. 1. 1. 2. 2. 2. 2.]
 [1. 1. 2. 1. 1. 1. 1.]
 [2. 2. 2. 1. 2. 1. 2.]]


-----------------------------------------------The program will print (in the few below lines) the positions that score came from---------------------------------------
AI Horizontal Winning Moves From Row Num=2
are:
[1, 1, 1, 2, 2, 2, 2]
AI Horizontal Winning Moves From Row Num=5
are:
[1, 2, 1, 2, 2, 2, 2]
AI vertical Winning Moves From Column num=4
are:
[2, 1, 2, 2, 2, 2]
AI vertical Winning Moves From Column num=5
are:
[1, 1, 2, 2, 2, 2]
AI winning +ve slopped-diagonal starting from the first row and going upward
[2.0, 2.0, 2.0, 2.0, 2.0, 2.0]
AI winning negatively slopped-diagonal starting from  most top row and going downward
AI vertical Winning Moves From Column num=5
are:
[1, 1, 2, 2, 2, 2]
AI winning +ve slopped-diagonal starting from the first row and going upward
[2.0, 2.0, 2.0, 2.0, 2.0, 2.0]
AI winning negatively slopped-diagonal starting from  most top row and going downward
[2.0, 2.0, 2.0, 2.0]
Human Horizontal Winning Moves From Row Num=1
are:
[1, 1, 2, 1, 1, 1, 1]
Human vertical Winning Moves From Column num=2
are:
[2, 2, 1, 1, 1, 1]
Human winning +ve slopped-diagonal starting from the first column and going upward
[1.0, 1.0, 1.0, 1.0, 2.0]
==========================================================================================================================================================================
-----------------------------------------------Total Nodes Using MinMax With α-β Pruning=26863 Nodes
-----------------------------------------------Total Time Using MinMax with α-β Pruning= 12.604137199999972 seconds
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------My AI Total Score= 6
-----------------------------------------------Human Total score= 3
==========================================================================================================================================================================
-----------------------------------------------------Your GUI Gonna Shut Down Automatically After 5 Seconds, Just Wait or Press the X Button of the "PyGame-GUI" to Exit -------
-------------------------------------------------------------------------------End Of The program-------------------------------------------------------------------
-----------------------------------------------------------------------------------bye..bye!-------------------------------------------------------------------

Process finished with exit code 0
```

----------The 4-connect Game using Minimax Algorithms----------

----------Welcome To Your Game----------

To start PLZ press on the Algorithm desired type buttons, and press one of the depth buttons from 1 to 4.. if the depth is greater than this: your CPU, RAMS, and your whole computer will suffer and start to be slow

The Board After Updating It:

[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]]

The Board After Updating It:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 1.4070797000000006 seconds
Num of Nodes Expanded To Take This Move only= 2827 Nodes
The Board After Updating It:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]]

The Board After Updating It:
[[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. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 1.225373900000001 seconds
Num of Nodes Expanded To Take This Move only= 2471 Nodes
The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 2. 1. 0. 0. 0.]]

The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 2. 1. 0. 0. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 1.199402599999999 seconds
Num of Nodes Expanded To Take This Move only= 2659 Nodes
The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 2. 1. 2. 0. 0.]]

The Board After Updating It:
[[0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 2. 1. 2. 0. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 1.1788497000000007 seconds

Num of Nodes Expanded To Take This Move only= 2630 Nodes
The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 1. 2. 0. 0.]]

The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 1. 2. 1. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 1.0309688999999977 seconds
Num of Nodes Expanded To Take This Move only= 2145 Nodes
The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]
 [0. 0. 2. 1. 2. 1. 0.]]

The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 1. 0.]
 [0. 0. 2. 1. 2. 1. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 1.7053671000000037 seconds
Num of Nodes Expanded To Take This Move only= 3686 Nodes
The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 2. 0.]
 [0. 0. 2. 1. 0. 1. 0.]
 [0. 0. 2. 1. 2. 1. 0.]]

The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 2. 0.]
 [0. 0. 2. 1. 1. 1. 0.]
 [0. 0. 2. 1. 2. 1. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 1.5051804000000004 seconds
Num of Nodes Expanded To Take This Move only= 3163 Nodes
The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 2. 2. 0.]
 [0. 0. 2. 1. 1. 1. 0.]
 [0. 0. 2. 1. 2. 1. 0.]]

The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]

[0. 0. 1. 2. 2. 2. 0.]
[0. 0. 2. 1. 1. 1. 0.]
[0. 0. 2. 1. 2. 1. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 1.1708138999999989 seconds
Num of Nodes Expanded To Take This Move only= 2573 Nodes
The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 1. 2. 0. 0.]
[0. 0. 1. 2. 2. 2. 0.]
[0. 0. 2. 1. 1. 1. 0.]
[0. 0. 2. 1. 2. 1. 0.]]

The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0.]
[0. 0. 1. 1. 2. 0. 0.]
[0. 0. 1. 2. 2. 2. 0.]
[0. 0. 2. 1. 1. 1. 0.]
[0. 0. 2. 1. 2. 1. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.7766537999999983 seconds
Num of Nodes Expanded To Take This Move only= 1604 Nodes
The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 0. 1. 2. 0. 0.]
[0. 0. 1. 1. 2. 0. 0.]
[0. 0. 1. 2. 2. 2. 0.]
[0. 0. 2. 1. 1. 1. 0.]
[0. 0. 2. 1. 2. 1. 0.]]

The Board After Updating It:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 2. 2. 2. 0.]
 [0. 0. 2. 1. 1. 1. 0.]
 [0. 0. 2. 1. 2. 1. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.4372329999999991 seconds
Num of Nodes Expanded To Take This Move only= 901 Nodes
The Board After Updating It:
[[0. 0. 0. 2. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 2. 2. 2. 0.]
 [0. 0. 2. 1. 1. 1. 0.]
 [0. 0. 2. 1. 2. 1. 0.]]

The Board After Updating It:
[[0. 0. 1. 2. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 2. 2. 2. 0.]
 [0. 0. 2. 1. 1. 1. 0.]
 [0. 0. 2. 1. 2. 1. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.12088959999999815 seconds
Num of Nodes Expanded To Take This Move only= 246 Nodes
The Board After Updating It:
[[0. 0. 1. 2. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 2. 2. 2. 0.]

```
 [0. 0. 2. 1. 1. 1. 0.]
 [0. 2. 2. 1. 2. 1. 0.]]


The Board After Updating It:
[[0. 0. 1. 2. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 2. 2. 2. 0.]
 [0. 1. 2. 1. 1. 1. 0.]
 [0. 2. 2. 1. 2. 1. 0.]]


your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.16876039999999648
seconds
Num of Nodes Expanded To Take This Move only= 382  Nodes
The Board After Updating It:
[[0. 0. 1. 2. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 2. 2. 2. 0.]
 [0. 1. 2. 1. 1. 1. 0.]
 [2. 2. 2. 1. 2. 1. 0.]]


The Board After Updating It:
[[0. 0. 1. 2. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 2. 2. 2. 0.]
 [1. 1. 2. 1. 1. 1. 0.]
 [2. 2. 2. 1. 2. 1. 0.]]


your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.13683219999999352
seconds
Num of Nodes Expanded To Take This Move only= 309  Nodes
The Board After Updating It:
```

[[0. 0. 1. 2. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 1. 2. 2. 0.]
 [0. 0. 1. 2. 2. 2. 0.]
 [1. 1. 2. 1. 1. 1. 0.]
 [2. 2. 2. 1. 2. 1. 0.]]

The Board After Updating It:
[[0. 0. 1. 2. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 1. 2. 2. 0.]
 [0. 1. 1. 2. 2. 2. 0.]
 [1. 1. 2. 1. 1. 1. 0.]
 [2. 2. 2. 1. 2. 1. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.18990060000000142 seconds
Num of Nodes Expanded To Take This Move only= 444 Nodes
The Board After Updating It:
[[0. 0. 1. 2. 2. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 2. 1. 1. 2. 2. 0.]
 [0. 1. 1. 2. 2. 2. 0.]
 [1. 1. 2. 1. 1. 1. 0.]
 [2. 2. 2. 1. 2. 1. 0.]]

The Board After Updating It:
[[0. 0. 1. 2. 2. 0. 0.]
 [0. 1. 1. 1. 2. 0. 0.]
 [0. 2. 1. 1. 2. 2. 0.]
 [0. 1. 1. 2. 2. 2. 0.]
 [1. 1. 2. 1. 1. 1. 0.]
 [2. 2. 2. 1. 2. 1. 0.]]

your picked depth= 5

Time During Using Minmax With Alph Beta Pruning Algorithm: 0.14421899999999965 seconds
Num of Nodes Expanded To Take This Move only= 327 Nodes
The Board After Updating It:
[[0. 0. 1. 2. 2. 0. 0.]
 [0. 1. 1. 1. 2. 2. 0.]
 [0. 2. 1. 1. 2. 2. 0.]
 [0. 1. 1. 2. 2. 2. 0.]
 [1. 1. 2. 1. 1. 1. 0.]
 [2. 2. 2. 1. 2. 1. 0.]]

The Board After Updating It:
[[0. 0. 1. 2. 2. 0. 0.]
 [0. 1. 1. 1. 2. 2. 0.]
 [0. 2. 1. 1. 2. 2. 0.]
 [1. 1. 1. 2. 2. 2. 0.]
 [1. 1. 2. 1. 1. 1. 0.]
 [2. 2. 2. 1. 2. 1. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.10102899999999693 seconds
Num of Nodes Expanded To Take This Move only= 230 Nodes
The Board After Updating It:
[[0. 0. 1. 2. 2. 2. 0.]
 [0. 1. 1. 1. 2. 2. 0.]
 [0. 2. 1. 1. 2. 2. 0.]
 [1. 1. 1. 2. 2. 2. 0.]
 [1. 1. 2. 1. 1. 1. 0.]
 [2. 2. 2. 1. 2. 1. 0.]]

The Board After Updating It:
[[0. 0. 1. 2. 2. 2. 0.]
 [0. 1. 1. 1. 2. 2. 0.]
 [1. 2. 1. 1. 2. 2. 0.]
 [1. 1. 1. 2. 2. 2. 0.]
 [1. 1. 2. 1. 1. 1. 0.]

[2. 2. 2. 1. 2. 1. 0.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.051618300000000123 seconds
Num of Nodes Expanded To Take This Move only= 119 Nodes
The Board After Updating It:
[[0. 0. 1. 2. 2. 2. 0.]
 [0. 1. 1. 1. 2. 2. 0.]
 [1. 2. 1. 1. 2. 2. 0.]
 [1. 1. 1. 2. 2. 2. 0.]
 [1. 1. 2. 1. 1. 1. 0.]
 [2. 2. 2. 1. 2. 1. 2.]]

The Board After Updating It:
[[0. 0. 1. 2. 2. 2. 0.]
 [0. 1. 1. 1. 2. 2. 0.]
 [1. 2. 1. 1. 2. 2. 0.]
 [1. 1. 1. 2. 2. 2. 0.]
 [1. 1. 2. 1. 1. 1. 1.]
 [2. 2. 2. 1. 2. 1. 2.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.03960529999999096 seconds
Num of Nodes Expanded To Take This Move only= 100 Nodes
The Board After Updating It:
[[0. 0. 1. 2. 2. 2. 0.]
 [0. 1. 1. 1. 2. 2. 0.]
 [1. 2. 1. 1. 2. 2. 0.]
 [1. 1. 1. 2. 2. 2. 2.]
 [1. 1. 2. 1. 1. 1. 1.]
 [2. 2. 2. 1. 2. 1. 2.]]

The Board After Updating It:
[[0. 0. 1. 2. 2. 2. 0.]

[0. 1. 1. 1. 2. 2. 0.]
[1. 2. 1. 1. 2. 2. 1.]
[1. 1. 1. 2. 2. 2. 2.]
[1. 1. 2. 1. 1. 1. 1.]
[2. 2. 2. 1. 2. 1. 2.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.0116110999999961 seconds
Num of Nodes Expanded To Take This Move only= 38 Nodes
The Board After Updating It:
[[0. 0. 1. 2. 2. 2. 0.]
[2. 1. 1. 1. 2. 2. 0.]
[1. 2. 1. 1. 2. 2. 1.]
[1. 1. 1. 2. 2. 2. 2.]
[1. 1. 2. 1. 1. 1. 1.]
[2. 2. 2. 1. 2. 1. 2.]]

The Board After Updating It:
[[1. 0. 1. 2. 2. 2. 0.]
[2. 1. 1. 1. 2. 2. 0.]
[1. 2. 1. 1. 2. 2. 1.]
[1. 1. 1. 2. 2. 2. 2.]
[1. 1. 2. 1. 1. 1. 1.]
[2. 2. 2. 1. 2. 1. 2.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.0021504999999999056 seconds
Num of Nodes Expanded To Take This Move only= 7 Nodes
The Board After Updating It:
[[1. 2. 1. 2. 2. 2. 0.]
[2. 1. 1. 1. 2. 2. 0.]
[1. 2. 1. 1. 2. 2. 1.]
[1. 1. 1. 2. 2. 2. 2.]
[1. 1. 2. 1. 1. 1. 1.]
[2. 2. 2. 1. 2. 1. 2.]]

The Board After Updating It:
[[1. 2. 1. 2. 2. 2. 0.]
 [2. 1. 1. 1. 2. 2. 1.]
 [1. 2. 1. 1. 2. 2. 1.]
 [1. 1. 1. 2. 2. 2. 2.]
 [1. 1. 2. 1. 1. 1. 1.]
 [2. 2. 2. 1. 2. 1. 2.]]

your picked depth= 5
Time During Using Minmax With Alph Beta Pruning Algorithm: 0.000598199999998883 seconds
Num of Nodes Expanded To Take This Move only= 2 Nodes
The Board After Updating It:
[[1. 2. 1. 2. 2. 2. 2.]
 [2. 1. 1. 1. 2. 2. 1.]
 [1. 2. 1. 1. 2. 2. 1.]
 [1. 1. 1. 2. 2. 2. 2.]
 [1. 1. 2. 1. 1. 1. 1.]
 [2. 2. 2. 1. 2. 1. 2.]]

----------The program will print (in the few below lines) the positions that score came from----------
AI Horizontal Winning Moves From Row Num=2
are:
[1, 1, 1, 2, 2, 2, 2]
AI Horizontal Winning Moves From Row Num=5
are:
[1, 2, 1, 2, 2, 2, 2]
AI vertical Winning Moves From Column num=4
are:
[2, 1, 2, 2, 2, 2]
AI vertical Winning Moves From Column num=5
are:
[1, 1, 2, 2, 2, 2]
AI winning +ve slopped-diagonal starting from the first row and going upward
[2.0, 2.0, 2.0, 2.0, 2.0, 2.0]

AI winning negatively slopped-diagonal starting from most top row and going downward
[2.0, 2.0, 2.0, 2.0]
Human Horizontal Winning Moves From Row Num=1
are:
[1, 1, 2, 1, 1, 1, 1]
Human vertical Winning Moves From Column num=2
are:
[2, 2, 1, 1, 1, 1]
Human winning +ve slopped-diagonal starting from the first column and going upward
[1.0, 1.0, 1.0, 1.0, 2.0]
==========
----------Total Nodes Using MinMax With α-β Pruning=26863 Nodes
----------Total Time Using MinMax with α-β Pruning= 12.604137199999972 seconds
----
----------My AI Total Score= 6
----------Human Total score= 3
==========
----------Your GUI Gonna Shut Down Automatically After 5 Seconds, Just Wait or Press the X Button of the "PyGame-GUI" to Exit ----------
----------End Of The program----------
----------bye..bye!----------
Process finished with exit code 0

From these two comparisons between our two Methods in time and space we can conclude that alpha-beta pruning is the best

Sample Printing Of The Tree_Output.txt file and Its Utilities Score in Each Move with Depth=1 Using Minmax with α-β pruning

The score= 0
The Board State achieving it:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [2. 0. 0. 1. 0. 0. 0.]]

The score= 0
The Board State achieving it:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 2. 0. 1. 0. 0. 0.]]

The score= 0
The Board State achieving it:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]]

The score= 10
The Board State achieving it:
[[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. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]]

The score= 0
The Board State achieving it:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 2. 0. 0.]]

The score= 0
The Board State achieving it:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 2. 0.]]

The score= 0
The Board State achieving it:
[[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. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 2.]]

The score= 10
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [2. 0. 0. 1. 0. 0. 0.]]

The score= 10
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 2. 0. 1. 0. 0. 0.]]

The score= 20
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]]

The score= 20
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]]

The score= 20
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 2. 0. 0.]]

The score= 10
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 2. 0.]]

The score= 10
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 2.]]

The score= 20
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [2. 0. 2. 1. 0. 0. 0.]]

The score= 20
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 2. 2. 1. 0. 0. 0.]]

The score= 60
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]]

The score= 30
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]]

The score= 30
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 2. 1. 2. 0. 0.]]

The score= 20
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 2. 0.]]

The score= 20
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 2.]]

The score= -1440
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [2. 0. 2. 1. 0. 0. 0.]]

The score= -1440
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 2. 2. 1. 0. 0. 0.]]

The score= -1320
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]]

The score= 70
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]]

The score= -1430
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 0. 2. 1. 2. 0. 0.]]

The score= -1440
The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 2. 0.]]

The score= -1440

The Board State achieving it:
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 2.]]

The score= 60
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [2. 0. 2. 1. 0. 0. 0.]]

The score= 60
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 2. 2. 1. 0. 0. 0.]]

The score= 60
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]]

The score= 60
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 0. 2. 1. 2. 0. 0.]]

The score= 60
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 2. 0.]]

The score= 60
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 2.]]

The score= 160
The Board State achieving it:

[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [2. 0. 2. 2. 0. 0. 0.]
 [2. 0. 2. 1. 0. 0. 0.]]

The score= 60
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [2. 2. 2. 1. 0. 0. 0.]]

The score= 70
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 2. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [2. 0. 2. 1. 0. 0. 0.]]

The score= 60
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [2. 0. 2. 1. 2. 0. 0.]]

The score= 60
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [2. 0. 2. 1. 0. 2. 0.]]

The score= 60
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 2. 2. 0. 0. 0.]
 [2. 0. 2. 1. 0. 0. 2.]]

The score= -1240
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0. 0.]
 [2. 0. 1. 1. 0. 0. 0.]
 [2. 0. 2. 2. 0. 0. 0.]
 [2. 0. 2. 1. 0. 0. 0.]]

The score= -1340
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]

[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 0. 2. 2. 0. 0. 0.]
[2. 2. 2. 1. 0. 0. 0.]]

The score= 190
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 0. 2. 2. 0. 0. 0.]
[2. 0. 2. 1. 0. 0. 0.]]

The score= -1340
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 0. 2. 2. 0. 0. 0.]
[2. 0. 2. 1. 2. 0. 0.]]

The score= -1340
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 0. 2. 2. 0. 0. 0.]
[2. 0. 2. 1. 0. 2. 0.]]

The score= -1340
The Board State achieving it:
[[0. 0. 0. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 0. 2. 2. 0. 0. 0.]
[2. 0. 2. 1. 0. 0. 2.]]

The score= 290
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 0. 1. 1. 0. 0. 0.]
[2. 0. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 0.]]

The score= 1180
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 0.]]

The score= 190
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]

[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 0. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 2. 0. 0.]]

The score= 190
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 0. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 2. 0.]]

The score= 190
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 0. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 2.]]

The score= -1720
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 0.]]

The score= -3320
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 2. 1. 1. 0. 0. 0.]
[0. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 0.]]

The score= -3320
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 2. 0. 0.]]

The score= -3320
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 2. 0.]]

The score= -3320
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]

[0. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 2.]]

The score= -2220
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 0.]]

The score= -4720
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 2. 1. 1. 0. 0. 0.]
[0. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 0.]]

The score= -4720
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 2. 0. 0.]]

The score= -4720
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 2. 0.]]

The score= -4720
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 0. 1. 1. 0. 0. 0.]
[0. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 2.]]

The score= -7230
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 0.]]

The score= -8040
The Board State achieving it:
[[0. 2. 2. 2. 0. 0. 0.]
[0. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]

[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 0.]]

The score= -9720
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 2. 0. 0.]]

The score= -9720
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 2. 0.]]

The score= -9720
The Board State achieving it:
[[0. 0. 2. 2. 0. 0. 0.]
[0. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 2.]]

The score= -4250
The Board State achieving it:
[[1. 2. 2. 2. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 0. 0.]]

The score= -5840
The Board State achieving it:
[[1. 0. 2. 2. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 2. 0. 0.]]

The score= -5840
The Board State achieving it:
[[1. 0. 2. 2. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]
[2. 1. 2. 1. 0. 2. 0.]]

The score= -5840
The Board State achieving it:
[[1. 0. 2. 2. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 1. 1. 1. 0. 0. 0.]
[2. 2. 2. 2. 0. 0. 0.]

[2. 1. 2. 1. 0. 0. 2.]]

The score= -5750
The Board State achieving it:
[[1. 2. 2. 2. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 2. 2. 2. 0. 0. 0.]
 [2. 1. 2. 1. 2. 1. 0.]]

The score= -5650
The Board State achieving it:
[[1. 2. 2. 2. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 2. 2. 2. 0. 2. 0.]
 [2. 1. 2. 1. 0. 1. 0.]]

The score= -5750
The Board State achieving it:
[[1. 2. 2. 2. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 2. 2. 2. 0. 0. 0.]
 [2. 1. 2. 1. 0. 1. 2.]]

The score= -7150
The Board State achieving it:
[[1. 2. 2. 2. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 2. 2. 2. 0. 2. 0.]
 [2. 1. 2. 1. 2. 1. 0.]]

The score= -7040
The Board State achieving it:
[[1. 2. 2. 2. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 1. 1. 1. 0. 2. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 2. 2. 2. 0. 2. 0.]
 [2. 1. 2. 1. 0. 1. 0.]]

The score= -7150
The Board State achieving it:
[[1. 2. 2. 2. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 1. 1. 1. 0. 0. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 2. 2. 2. 0. 2. 0.]
 [2. 1. 2. 1. 0. 1. 2.]]

The score= -8540
The Board State achieving it:
[[1. 2. 2. 2. 0. 0. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 1. 1. 1. 0. 2. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 2. 2. 2. 0. 2. 0.]
 [2. 1. 2. 1. 2. 1. 0.]]

The score= -8440
The Board State achieving it:
[[1. 2. 2. 2. 0. 2. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 1. 1. 1. 0. 2. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 2. 2. 2. 0. 2. 0.]
 [2. 1. 2. 1. 0. 1. 0.]]

The score= -8540
The Board State achieving it:
[[1. 2. 2. 2. 0. 0. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 1. 1. 1. 0. 2. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 2. 2. 2. 0. 2. 0.]
 [2. 1. 2. 1. 0. 1. 2.]]

The score= -5050
The Board State achieving it:
[[1. 2. 2. 2. 0. 2. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 1. 1. 1. 0. 2. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 2. 2. 2. 2. 2. 0.]
 [2. 1. 2. 1. 1. 1. 0.]]

The score= -8440
The Board State achieving it:
[[1. 2. 2. 2. 0. 2. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 1. 1. 1. 0. 2. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 2. 2. 2. 0. 2. 0.]
 [2. 1. 2. 1. 1. 1. 2.]]

The score= -2160
The Board State achieving it:
[[1. 2. 2. 2. 0. 2. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 1. 1. 1. 2. 2. 0.]
 [2. 1. 1. 1. 1. 1. 0.]
 [2. 2. 2. 2. 2. 2. 0.]
 [2. 1. 2. 1. 1. 1. 0.]]

The score= -2160
The Board State achieving it:
[[1. 2. 2. 2. 0. 2. 0.]
 [2. 1. 1. 1. 0. 1. 0.]
 [2. 1. 1. 1. 0. 2. 0.]
 [2. 1. 1. 1. 1. 1. 0.]
 [2. 2. 2. 2. 2. 2. 0.]
 [2. 1. 2. 1. 1. 1. 2.]]

The score= 2720
The Board State achieving it:
[[1. 2. 2. 2. 2. 2. 0.]
 [2. 1. 1. 1. 1. 1. 0.]
 [2. 1. 1. 1. 2. 2. 0.]
 [2. 1. 1. 1. 1. 1. 0.]
 [2. 2. 2. 2. 2. 2. 0.]
 [2. 1. 2. 1. 1. 1. 0.]]

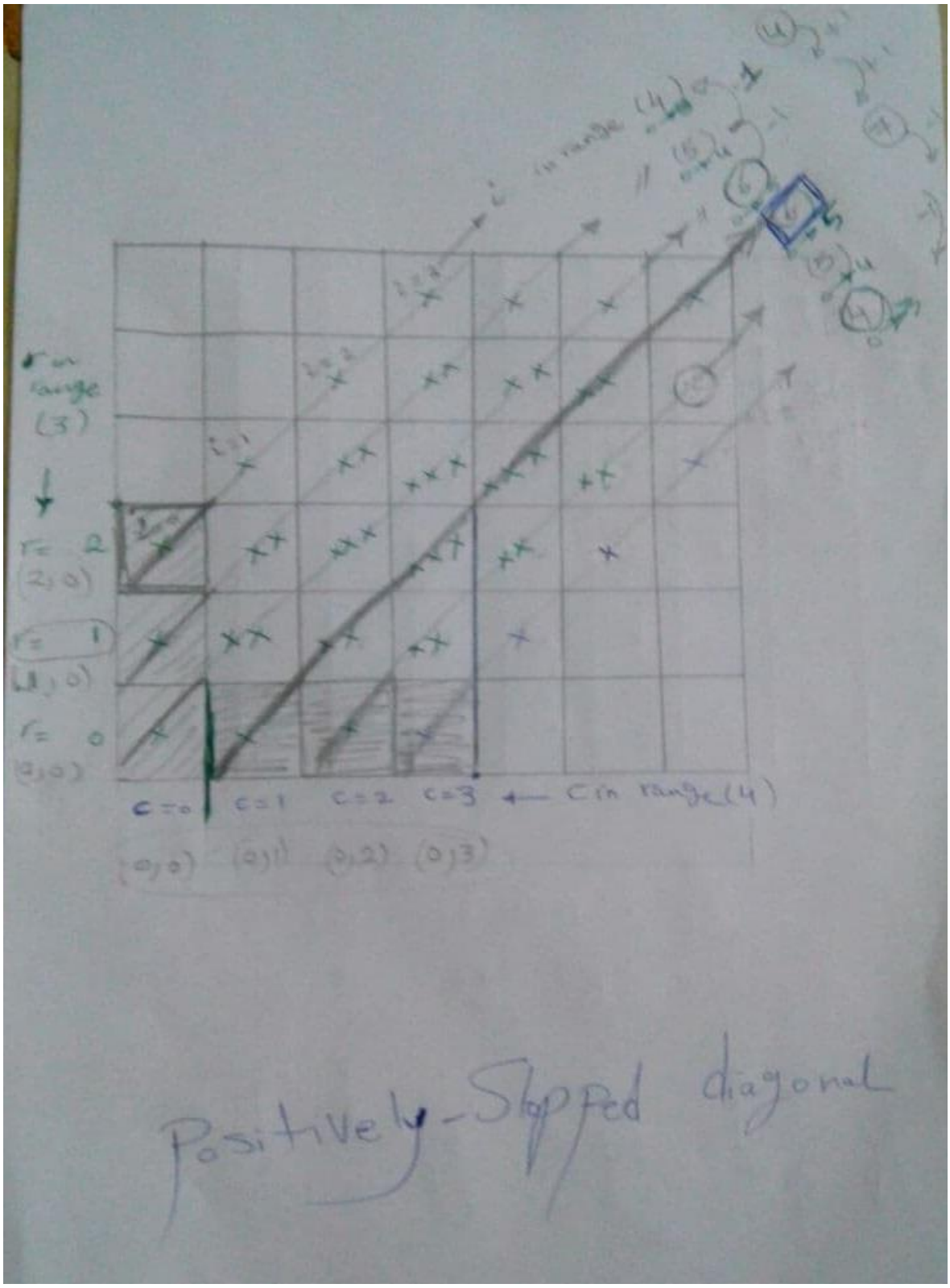
The score= 830
The Board State achieving it:
[[1. 2. 2. 2. 0. 2. 0.]
 [2. 1. 1. 1. 1. 1. 0.]
 [2. 1. 1. 1. 2. 2. 0.]
 [2. 1. 1. 1. 1. 1. 0.]
 [2. 2. 2. 2. 2. 2. 0.]
 [2. 1. 2. 1. 1. 1. 2.]]

The score= 5120
The Board State achieving it:
[[1. 2. 2. 2. 2. 2. 0.]
 [2. 1. 1. 1. 1. 1. 0.]
 [2. 1. 1. 1. 2. 2. 0.]
 [2. 1. 1. 1. 1. 1. 0.]
 [2. 2. 2. 2. 2. 2. 2.]
 [2. 1. 2. 1. 1. 1. 1.]]

The score= 6620
The Board State achieving it:
[[1. 2. 2. 2. 2. 2. 0.]
 [2. 1. 1. 1. 1. 1. 0.]
 [2. 1. 1. 1. 2. 2. 2.]
 [2. 1. 1. 1. 1. 1. 1.]
 [2. 2. 2. 2. 2. 2. 2.]
 [2. 1. 2. 1. 1. 1. 1.]]

The score= 100000000000000
The Board State achieving it:
[[1. 2. 2. 2. 2. 2. 2.]
 [2. 1. 1. 1. 1. 1. 1.]
 [2. 1. 1. 1. 2. 2. 2.]
 [2. 1. 1. 1. 1. 1. 1.]
 [2. 2. 2. 2. 2. 2. 2.]
 [2. 1. 2. 1. 1. 1. 1.]]

My Diagonals Algorithm Ideas



My Diagonals Algorithm Ideas