ALPHA & BETA PRUNING ALGORITHM :

import time

def max\_alpha\_beta(self, alpha, beta):

maxv = -2

px = None

py = None

result = self.is\_end()

if result == 'X':

return (-1, 0, 0)

elif result == 'O':

return (1, 0, 0)

elif result == '.':

return (0, 0, 0)

for i in range(0, 3):

for j in range(0, 3):

if self.current\_state[i][j] == '.':

self.current\_state[i][j] = 'O'

(m, min\_i, in\_j) = self.min\_alpha\_beta(alpha, beta)

if m > maxv:

maxv = m

px = i

py = j

self.current\_state[i][j] = '.'

if maxv >= beta:

return (maxv, px, py)

if maxv > alpha:

alpha = maxv

return (maxv, px, py)

def min\_alpha\_beta(self, alpha, beta):

minv = 2

qx = None

qy = None

result = self.is\_end()

if result == 'X':

return (-1, 0, 0)

elif result == 'O':

return (1, 0, 0)

elif result == '.':

return (0, 0, 0)

for i in range(0, 3):

for j in range(0, 3):

if self.current\_state[i][j] == '.':

self.current\_state[i][j] = 'X'

(m, max\_i, max\_j) = self.max\_alpha\_beta(alpha, beta)

if m < minv:

minv = m

qx = i

qy = j

self.current\_state[i][j] = '.'

if minv <= alpha:

return (minv, qx, qy)

if minv < beta:

beta = minv

return (minv, qx, qy)

def play\_alpha\_beta(self):

while True:

self.draw\_board()

self.result = self.is\_end()

if self.result != None:

if self.result == 'X':

print('The winner is X!')

elif self.result == 'O':

print('The winner is O!')

elif self.result == '.':

print("It's a tie!")

self.initialize\_game()

return

if self.player\_turn == 'X':

while True:

start = time.time()

(m, qx, qy) = self.min\_alpha\_beta(-2, 2)

end = time.time()

print('Evaluation time: {}s'.format(round(end - start, 7)))

print('Recommended move: X = {}, Y = {}'.format(qx, qy))

px = int(input('Insert the X coordinate: '))

py = int(input('Insert the Y coordinate: '))

qx = px

qy = py

if self.is\_valid(px, py):

self.current\_state[px][py] = 'X'

self.player\_turn = 'O'

break

else:

print('The move is not valid! Try again.')

else:

(m, px, py) = self.max\_alpha\_beta(-2, 2)

self.current\_state[px][py] = 'O'

self.player\_turn = 'X'