**EXPERIMENT NO : 07 DATE : 26/03/24**

**Aim**: To write code to play tic-tac-toe against a player using the minimax algorithm.

**Theory**:

Tic Tac Toe is one of the most played games and is the best time killer game that

you can play anywhere with just a pen and paper. If you don’t know how to play this game don’t worry let us first understand that. The game is played by two individuals and first, we draw a board with a 3×3 square grid. The first player chooses ‘X’ and draws it on any of the square grid, then it’s the chance of the second player to draw ‘O’ on the available spaces. Like this, the players draw ‘X’ and ‘O’ alternatively on the empty spaces until a player succeeds in drawing 3 consecutive marks either in the horizontal,

vertical or diagonal way. Then the player wins the game otherwise the game draws when all spots are

filled.

**Rules:**

• Traditionally the first player plays with "X". So, you can decide who wants to go

with "X" and who wants to go with "O".

• Only one player can play at a time.

• If any of the players have filled a square then the other player and the same

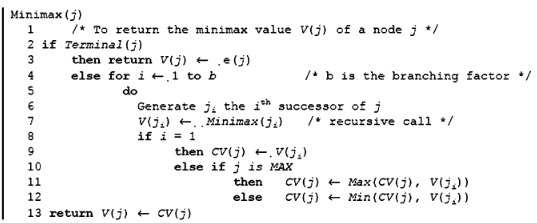
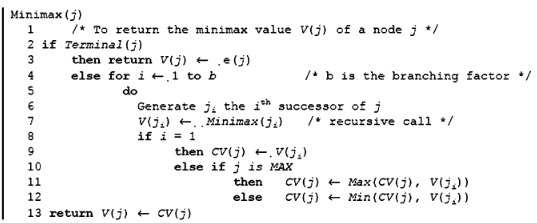
player cannot override that square.

• There are only two conditions that may match will be draw or may win.

• The player that succeeds in placing three respective marks (X or O) in a

horizontal, vertical, or diagonal row wins the game.

**Algorithm:**



**Code:**

from copy import deepcopy

def heuristic(state):

for row in range(3):

if state[row][0] == state[row][1] and state[row][1] == state[row][2]:

if state[row][0] == "X":

return 100

elif state[row] == "O":

return -100

for column in range(3):

if (

state[0][column] == state[1][column]

and state[1][column] == state[2][column]

):

if state[0][column] == "X":

return 100

elif state[0][column] == "O":

return -100

if (state[0][0] == state[1][1] and state[1][1] == state[2][2]) or (

state[0][2] == state[1][1] and state[1][1] == state[2][0]

):

if state[1][1] == "X":

return 100

elif state[1][1] == "O":

return -100

return 0

def genChildren(state, maxPlayer):

children = []

for i in range(3):

for j in range(3):

tempState = deepcopy(state)

if tempState[i][j] == "\_":

if maxPlayer:

tempState[i][j] = "X"

else:

tempState[i][j] = "O"

temptempState = deepcopy(tempState) children.append(temptempState)

tempState[i][j] = "\_"

return children

def minimax(state, maxPlayer):

heuristicValue = heuristic(state)

if abs(heuristicValue) == 100:

return heuristicValue

children = []

heuristics = []

if maxPlayer:

children = genChildren(state, maxPlayer)

if len(children) == 0:

return heuristic(state)

for child in children:

heuristicValue = minimax(child, False)

heuristics.append(heuristicValue)

max = -99999

for i in range(len(children)):

if heuristics[i] > max:

max = heuristics[i]

return max

else:

# find the best move to make here:

children = genChildren(state, maxPlayer)

if len(children) == 0:

return heuristic(state)

for child in children:

heuristicValue = minimax(child, True)

heuristics.append(heuristicValue)

min = 99999

for i in range(len(children)):

if heuristics[i] < min:

min = heuristics[i]

return min

def movesLeft(state):

for i in range(3):

for j in range(3):

if state[i][j] == "\_":

return True

def playGame():

print("You will play as 'X', against the AI. Good luck.....\n")

state = [["\_" for i in range(3)] for j in range(3)]

print(\*state, sep="\n")

global drawFlag

while movesLeft(state):

print("enter the matrix coordinates of where you want to play:")

x, y = input().split()

x = int(x)

y = int(y)

if x not in range(3) and y not in range(3):

print("Invalid coordinates, enter again")

print()

continue

if state[x][y] != "\_":

print("Oi, no cheating, you cant overwrite a move.... enter again")

print()

continue

state[x][y] = "X"

print("your move: ")

print(\*state, sep="\n")

if heuristic(state) == 100:

print("YAYY, you won and beat the ai...")

print("exiting game....")

drawFlag = False

return

moves = []

heuristics = []

moves = genChildren(state, False)

for move in moves:

heuristics.append(minimax(move, True))

# choose the best move

min = 99999

for i in range(len(moves)):

if heuristics[i] < min:

min = heuristics[i]

state = moves[i]

print("ai move: ")

print(\*state, sep="\n")

# check if the ai won

if heuristic(state) == -100:

print("Oh no, the ai beat you...")

print("exiting game....")

drawFlag = False

return

drawFlag = True

if \_\_name\_\_ == "\_\_main\_\_":

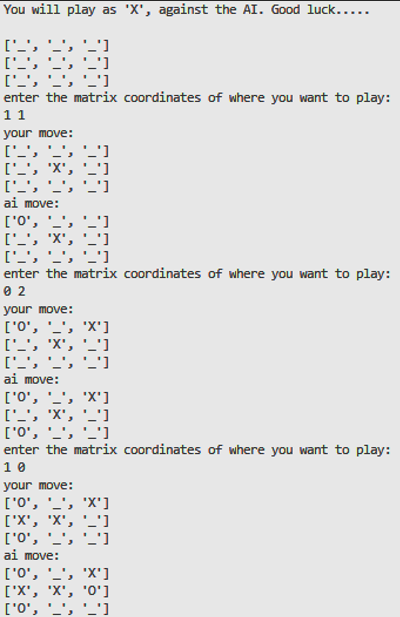
playGame()

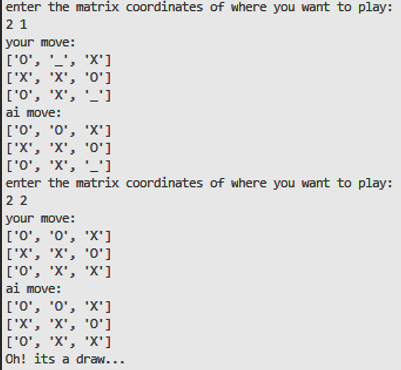
if drawFlag:

print("Oh! its a draw...")

print()

**Output:**

****

****

**Conclusion:**

The code for playing tic-tac-toe was successfully executed using the minimax algorithm.