AI Lab Experiment 4 - DFS for Tic Tac Toe game

Harshit Aggarwal
RA1911003010782

Source Code-

```
import numpy as np
from math import inf as infinity

game_state = [[' ',' ',' '],
              [' ',' ',' '],
              [' ',' ',' ']]
players = ['X','O']

def play_move(state, player, block_num):
    if state[int((block_num-1)/3)][(block_num-1)%3] == ' ':
        state[int((block_num-1)/3)][(block_num-1)%3] = player
    else:
        block_num = int(input("Block is not empty, Choose again: "))
        play_move(state, player, block_num)

def copy_game_state(state):
    new_state = [[' ',' ',' '],[' ',' ',' '],[' ',' ',' ']]
    for i in range(3):
        for j in range(3):
            new_state[i][j] = state[i][j]
    return new_state

def check_current_state(game_state):
    # Check if draw
    draw_flag = 0
    for i in range(3):
        for j in range(3):
            if game_state[i][j] == ' ':
                draw_flag = 1
    if draw_flag == 0:
        return None, "Draw"

    # Check horizontals
    if (game_state[0][0] == game_state[0][1] and game_state[0][1] ==
game_state[0][2] and game_state[0][0] != ' '):
        return game_state[0][0], "Done"
    if (game_state[1][0] == game_state[1][1] and game_state[1][1] ==
game_state[1][2] and game_state[1][0] != ' '):
```

```
        return game_state[1][0], "Done"
    if (game_state[2][0] == game_state[2][1] and game_state[2][1] ==
game_state[2][2] and game_state[2][0] != ' '):
        return game_state[2][0], "Done"

    # Check verticals
    if (game_state[0][0] == game_state[1][0] and game_state[1][0] ==
game_state[2][0] and game_state[0][0] != ' '):
        return game_state[0][0], "Done"
    if (game_state[0][1] == game_state[1][1] and game_state[1][1] ==
game_state[2][1] and game_state[0][1] != ' '):
        return game_state[0][1], "Done"
    if (game_state[0][2] == game_state[1][2] and game_state[1][2] ==
game_state[2][2] and game_state[0][2] != ' '):
        return game_state[0][2], "Done"

    # Check diagonals
    if (game_state[0][0] == game_state[1][1] and game_state[1][1] ==
game_state[2][2] and game_state[0][0] != ' '):
        return game_state[1][1], "Done"
    if (game_state[2][0] == game_state[1][1] and game_state[1][1] ==
game_state[0][2] and game_state[2][0] != ' '):
        return game_state[1][1], "Done"

    return None, "Not Done"

def print_board(game_state):
    print('----------------')
    print('| ' + str(game_state[0][0]) + ' || ' + str(game_state[0][1]) + '
|| ' + str(game_state[0][2]) + ' |')
    print('----------------')
    print('| ' + str(game_state[1][0]) + ' || ' + str(game_state[1][1]) + '
|| ' + str(game_state[1][2]) + ' |')
    print('----------------')
    print('| ' + str(game_state[2][0]) + ' || ' + str(game_state[2][1]) + '
|| ' + str(game_state[2][2]) + ' |')
    print('----------------')


def getBestMove(state, player):

    winner_loser , done = check_current_state(state)
    if done == "Done" and winner_loser == 'O': # If AI won
        return 1
    elif done == "Done" and winner_loser == 'X': # If Human won
        return -1
    elif done == "Draw":    # Draw condition
        return 0
```

```
    moves = []
    empty_cells = []
    for i in range(3):
        for j in range(3):
            if state[i][j] == ' ':
                empty_cells.append(i*3 + (j+1))

    for empty_cell in empty_cells:
        move = {}
        move['index'] = empty_cell
        new_state = copy_game_state(state)
        play_move(new_state, player, empty_cell)

        if player == 'O':    # If AI
            result = getBestMove(new_state, 'X')    # make more depth tree
for human
            move['score'] = result
        else:
            result = getBestMove(new_state, 'O')    # make more depth tree
for AI
            move['score'] = result

        moves.append(move)

    # Find best move
    best_move = None
    if player == 'O':   # If AI player
        best = -infinity
        for move in moves:
            if move['score'] > best:
                best = move['score']
                best_move = move['index']
    else:
        best = infinity
        for move in moves:
            if move['score'] < best:
                best = move['score']
                best_move = move['index']

    return best_move

# PLaying
play_again = 'Y'
while play_again == 'Y' or play_again == 'y':
    game_state = [[' ',' ',' '],
              [' ',' ',' '],
              [' ',' ',' ']]
```

```
    current_state = "Not Done"
    print("\nNew Game!")
    print_board(game_state)
    player_choice = input("Choose which player goes first - X (You) or
O(AI): ")
    winner = None

    if player_choice == 'X' or player_choice == 'x':
        current_player_idx = 0
    else:
        current_player_idx = 1

    while current_state == "Not Done":
        if current_player_idx == 0: # Human's turn
            block_choice = int(input("Choose where to place (1 to 9): "))
            play_move(game_state ,players[current_player_idx],
block_choice)
        else:   # AI's turn
            block_choice = getBestMove(game_state,
players[current_player_idx])
            play_move(game_state ,players[current_player_idx],
block_choice)
            print("AI plays move: " + str(block_choice))
        print_board(game_state)
        winner, current_state = check_current_state(game_state)
        if winner != None:
            print(str(winner) + " won!")
        else:
            current_player_idx = (current_player_idx + 1)%2

        if current_state == "Draw":
            print("Draw!")

    play_again = input('Want to play again?(Y/N) : ')
    if play_again == 'N':
        exit()
```

Output-

```
New Game!
----------------
|   ||   ||   |
----------------
|   ||   ||   |
----------------
|   ||   ||   |
----------------
Choose which player goes first - X (You) or O(AI): X
Choose where to place (1 to 9): 1
----------------
| X ||   ||   |
----------------
|   ||   ||   |
----------------
|   ||   ||   |
----------------
AI plays move: 2
----------------
| X || O ||   |
----------------
|   ||   ||   |
----------------
|   ||   ||   |
----------------
Choose where to place (1 to 9): 4
----------------
| X || O ||   |
----------------
| X ||   ||   |
----------------
|   ||   ||   |
----------------
AI plays move: 7
```

```
----------------
| X || O ||   |
----------------
| X ||   ||   |
----------------
| O ||   ||   |
----------------
Choose where to place (1 to 9): 5
----------------
| X || O ||   |
----------------
| X || X ||   |
----------------
| O ||   ||   |
----------------
AI plays move: 6
----------------
| X || O ||   |
----------------
| X || X || O |
----------------
| O ||   ||   |
----------------
Choose where to place (1 to 9): 8
----------------
| X || O ||   |
----------------
| X || X || O |
----------------
| O || X ||   |
----------------
AI plays move: 3
```

```
---------------
| X || O || O |
---------------
| X || X || O |
---------------
| O || X ||   |
---------------
Choose where to place (1 to 9): 9
---------------
| X || O || O |
---------------
| X || X || O |
---------------
| O || X || X |
---------------
Draw!
Want to play again?(Y/N) : N
```