Lab 10

Game Playing in Artificial Intelligence

Submission Guidelines:

- Lab must be submitted in the google classroom.
- Submission other than google classroom won't not be accepted.
- You are required to submit a python (version 3 compatible) file and Notebook (Both) named after Your RollNo. e.g **I18-XXXX.ipynb** and **I18-XXXX.py**

Game Playing is an important domain of artificial intelligence. Games don't require much knowledge; the only knowledge we need to provide is the rules, legal moves and the conditions of winning or losing the game.

Both players try to win the game. So, both of them try to make the best move possible at each turn. Searching techniques like BFS(Breadth First Search) are not accurate for this as the branching factor is very high, so searching will take a lot of time. So, we need another search procedures that improve –

- **Generate procedure** so that only good moves are generated.
- **Test procedure** so that the best move can be explored first.

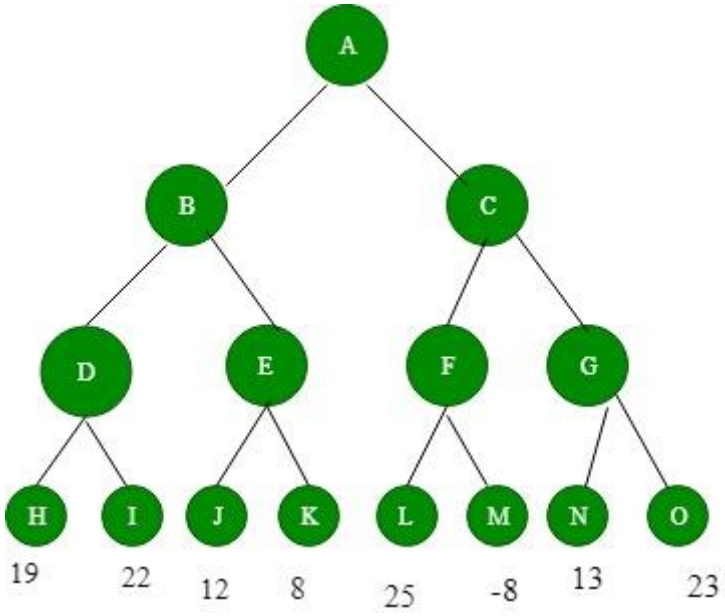
The most common search technique in game playing is **MinMax Search Algorithm**. It is depth-first depth-limited search procedure. It is used for games like **chess and tic-tac-toe**.

Minimax algorithm uses two functions –

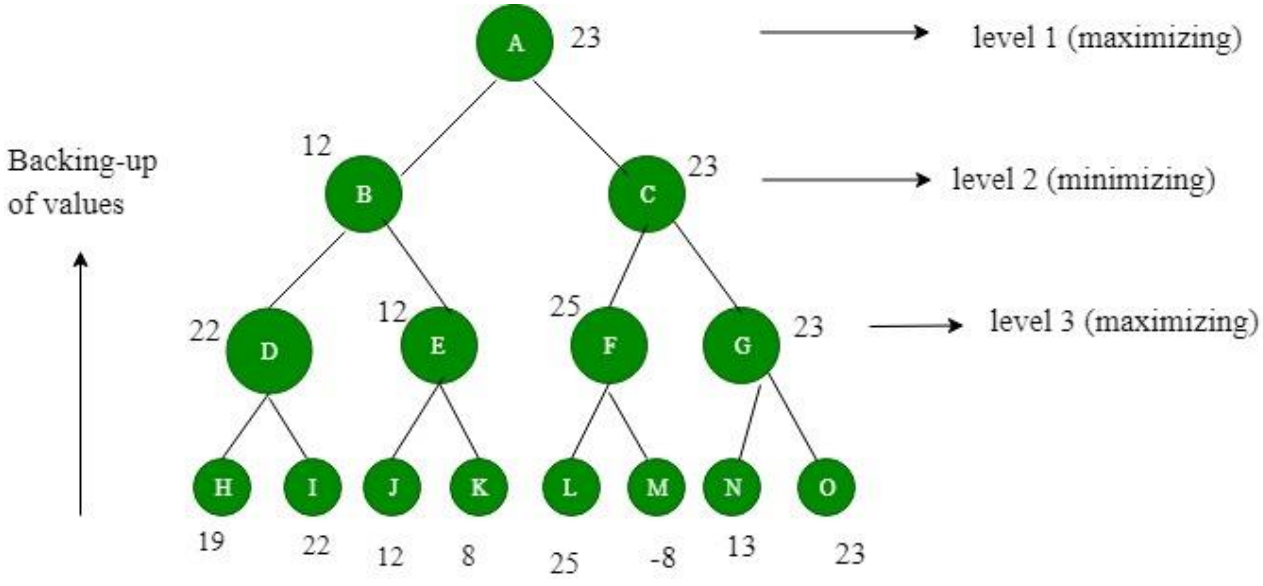
MOVEGEN : It generates all the possible moves that can be generated from the current position.
STATICEVALUATION : It returns a value depending upon the goodness from the view point of wo-player.

This algorithm is a two player game, so we call the first player as PLAYER1 and second player as PLAYER2. The value of each node is backed-up from its children. For PLAYER1 the backed-up value is the maximum value of its children and for PLAYER2 the backed-up value is the minimum value of its children. It provides most promising move to PLAYER1, assuming that the PLAYER2 has make the best move. It is a recursive algorithm, as same procedure occurs at each level.

Before Backing-up Values



After Backing-up values



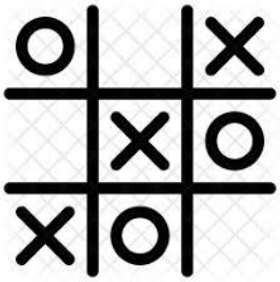
Lab

Task

Artificially Intelligent TIC-TAC-TOE Game

Tic-tac-toe (also known as **noughts** and **crosses** or Xs and Os) for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

The sample tic tac toe board of order 3 by 3 is shown in the figure below. Suppose player 1 had chosen X and player 2 has chosen 0.



Problem Statement

You need to implement **Min-Max Algorithm** to search optimal solution for each player in Tic-Tac-Toe game. (Algorithm is given at the end for your reference)

Rules of Game :

The simple rules to play the game are:

- Two players can play the game. Each player will choose either 0 or X.
- The winning of a player is dependent on the consecutive 0s or Xs (in either row or column or Diagonal).
- Each player will play alternately. There is no biasness towards or against any player.
- The game will stop when either player has won.
- If no player has won and all the board cells are occupied, the game has been drawn.

Code Provided

def initial_state(): Returns starting state of the board.
def player(board): Returns player who has the next turn on a board.
def actions(board): Returns set of all possible actions (i, j) available on the board.
def result(board, action): Returns the board that results from making move (i, j) on the board.
def winner(board): Returns the winner of the game if there is one.
def terminal(board): Returns True if game is over, False otherwise.
def utility(board): Returns 1 if X has won the game, -1 if O has won, 0 otherwise.

Tasks to Perform

def minimax(board): Returns the optimal action for the current player on the board.