Expt. No. 5 **Date: - 11/03/2025**

Aim: - To implement game playing algorithm in AI using python prog. Language

Theory: - Game Playing in Artificial Intelligence

Game playing has always been a fascinating domain for artificial intelligence (AI). From the early days of computer science to the current era of advanced deep learning systems, games have served as benchmarks for AI development. They offer structured environments with clear rules, making them ideal for training algorithms to solve complex problems. With AI's ability to learn, adapt, and make strategic decisions, it is now becoming an essential player in various gaming domains, reshaping how we experience and interact with games.

What is Game Playing in Artificial Intelligence?

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. Game playing in AI is an active area of research and has many practical applications, including game development, education, and military training. By simulating game playing scenarios, AI algorithms can be used to develop more effective decision-making systems for real-world applications.

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

The Minimax Search Algorithm

One of the most common search techniques in game playing is the **Minimax algorithm**, which is a depth-first, depth-limited search procedure. Minimax is commonly used for games like chess and tic-tac-toe.

Key Functions in Minimax:

1. **MOVEGEN**: Generates all possible moves from the current position.
2. **STATICEVALUATION**: Returns a value based on the quality of a game state from the perspective of two players.

In a two-player game, one player is referred to as PLAYER1 and the other as PLAYER2. The Minimax algorithm operates by backing up values from child nodes to their parent nodes. PLAYER1 tries to maximize the value of its moves, while PLAYER2 tries to minimize the value of its moves. The algorithm recursively performs this procedure at each level of the game tree.

Example of Minimax:

Figure 1: Before backing up values

(The diagram illustrates the game tree before Minimax values are propagated upward.)

Figure 2: After backing up values

The game starts with PLAYER1. The algorithm generates four levels of the game tree. The values for nodes H, I, J, K, L, M, N, and O are provided by the STATICEVALUATION function. Level 3 is a maximizing level, so each node at this level takes the maximum value of its children. Level 2 is a minimizing level, where each node takes the minimum value of its children. After this process, the value of node A is calculated as 23, meaning that PLAYER1 should choose move C to maximize the chances of winning.

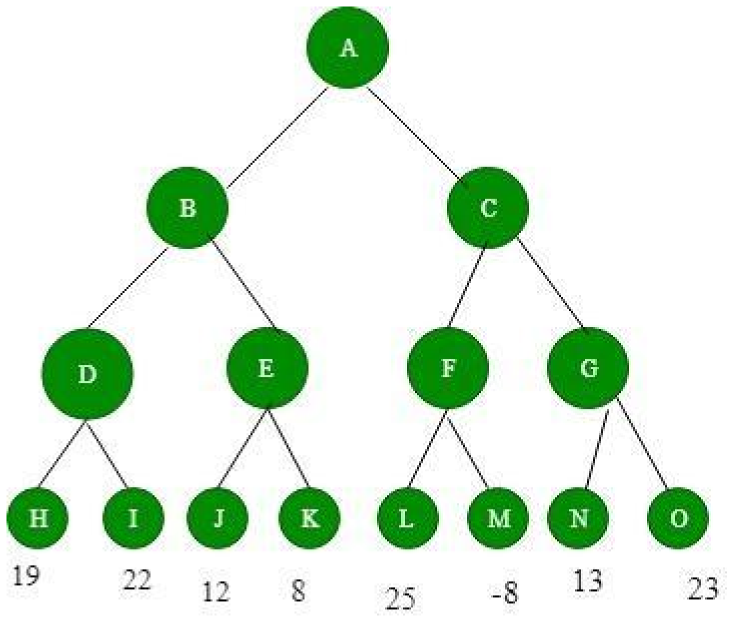


Figure 1: Before backing-up of values

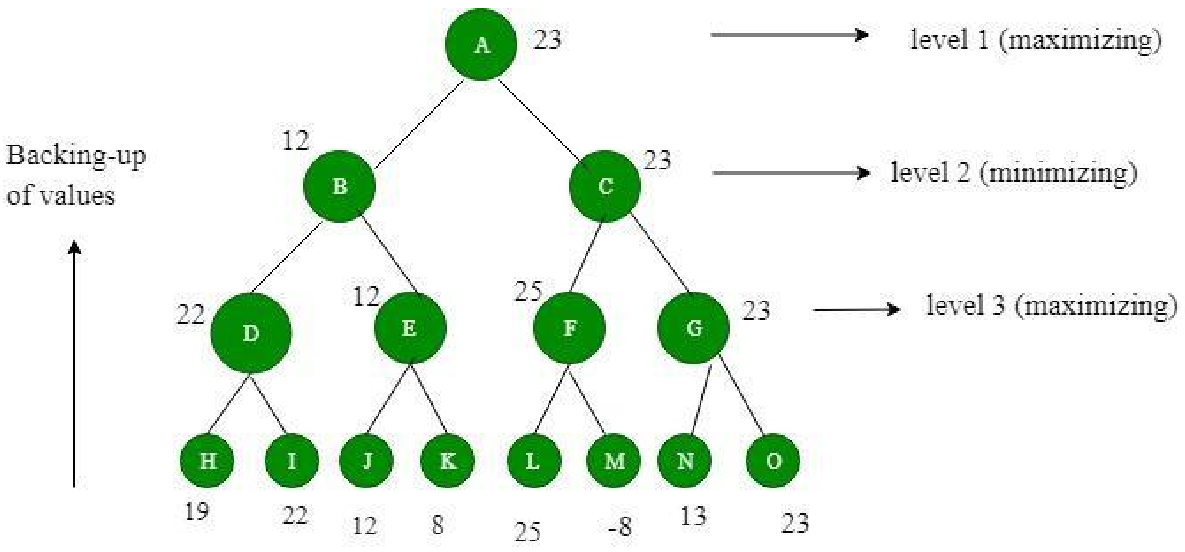


Figure 2: After backing-up of values We assume that PLAYER1 will start the game.

Advantages of Game Playing in Artificial Intelligence

1. **Advancement of AI:** Game playing has been a driving force behind the development of artificial intelligence and has led to the creation of new algorithms and techniques that can be applied to other areas of AI.
2. **Education and training:** Game playing can be used to teach AI techniques and algorithms to students and professionals, as well as to provide training for military and emergency response personnel.
3. **Research:** Game playing is an active area of research in AI and provides an opportunity to study and develop new techniques for decision-making and problem-solving.
4. **Real-world applications:** The techniques and algorithms developed for game playing can be applied to real-world applications, such as robotics, autonomous systems, and decision support systems.

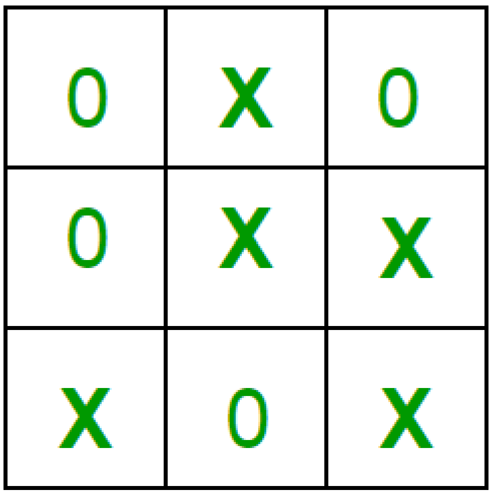
Disadvantages of Game Playing in Artificial Intelligence

1. **Limited scope:** The techniques and algorithms developed for game playing may not be well-suited for other types of applications and may need to be adapted or modified for different domains.
2. **Computational cost:** Game playing can be computationally expensive, especially for complex games such as chess or Go, and may require powerful computers to achieve real-time performance.

<u>Tic-Tac-Toe game: -</u>

Rules of the Game

1. The game is to be played between two people (in this program between HUMAN and COMPUTER).
2. One of the players chooses 'O' and the other 'X' to mark their respective cells.
3. The game starts with one of the players and the game ends when one of the players has one whole row/ column/ diagonal filled with his/her respective character ('O' or 'X').
4. If no one wins, then the game is said to be draw.



5. Implementation In our program the moves taken by the computer and the human are chosen randomly. We use rand() function for this.

Conclusion

AI is shaping the future of gaming in ways that were once thought impossible. From mastering complex games like chess and Go to generating dynamic game worlds and designing entirely new game experiences, AI continues to push the boundaries of what is possible in gaming. As AI evolves, so too will the possibilities for innovation in how games are played, designed, and experienced, ushering in a new era of interactive entertainment.

---END---

Python Code: -

Output: -

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
Board after 1 move
[[0 0 1]
 [0 0 0]
 [0 0 0]]
Board after 2 move
[[0 0 1]
 [0 0 0]
 [0 2 0]]
Board after 3 move
[[0 0 1]
 [1 0 0]
 [0 2 0]]
Board after 4 move
[[0 0 1]
 [1 0 0]
 [0 2 2]]
Board after 5 move
[[0 1 1]
 [1 0 0]
 [0 2 2]]
Board after 6 move
[[0 1 1]
 [1 2 0]
 [0 2 2]]
Board after 7 move
[[1 1 1]
 [1 2 0]
 [0 2 2]]
```

Winner is: 1

--END--