**3 PROBLEM DEFINITION**

Connect 4 is a 2-player mathematical board game. Since the game is always played in a completely observable environment, all the players have ‘perfect information’ about the game. The game is all about dropping 4 pieces of the same colored coin together either horizontally, vertically, or diagonally in the available 0 to 42 places in the board, containing a total of 4,531,985,219,092 positions1 to be played.

Since the game is strategical and can be solved using a mathematical solution, the artificial intelligence algorithms can be used to solve the game using a mixture of brute-force methods and sometimes, knowledge-based approach.

There are a few solved conclusions for the game. It is a first player winning game under perfect play. The first person can win when starting in the middle else the second person can get a win if the first person starts around the edges of the board. These conclusions don’t always come true as it is impossible for a person to play a perfect game all the time, the same cannot be said for artificial intelligence agents as it registers and remembers the moves made and also predicts the future moves to a certain degree based on the current move. The amount of look ahead for the artificial intelligence agent depends on the type of algorithm used on the agent.

The problem in focus is to find how individual artificial intelligent agents, in this case a total of three agents based on minimax, expectimax, and monte carlo tree search algorithm, fare against real players, between themselves, and also find the extent to which these aforementioned conclusions hold true.

**3.3 MONTE CARLO TREE SEARCH**

One of the most used heuristic search algorithms for decision making is Monte Carlo tree search2 (MCTS). The Monte Carlo method which used randomness and determines its next move from the previous move was combined with the game-tree search and hence the name MCTS. The MCTS uses the algorithm called the UCT (Upper Confidence bounds applied to Trees) for exploring and exploiting the simulated trees.

The expression for the UCT is given as follows and the highest value of the expression is used in decision-making models.

* *wi* – number of wins for the node after the ith move
* *ni* – number of simulations for the node after the ith move
* *Ni* – total number of simulations after the ith move run by the parent node
* *c* – the exploration parameter, chosen as for the game

The pseudocode for the UCT expression is shown in Figure 1.

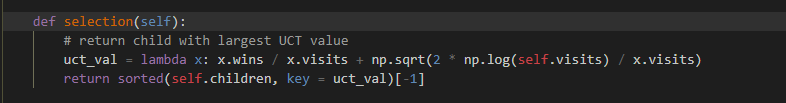
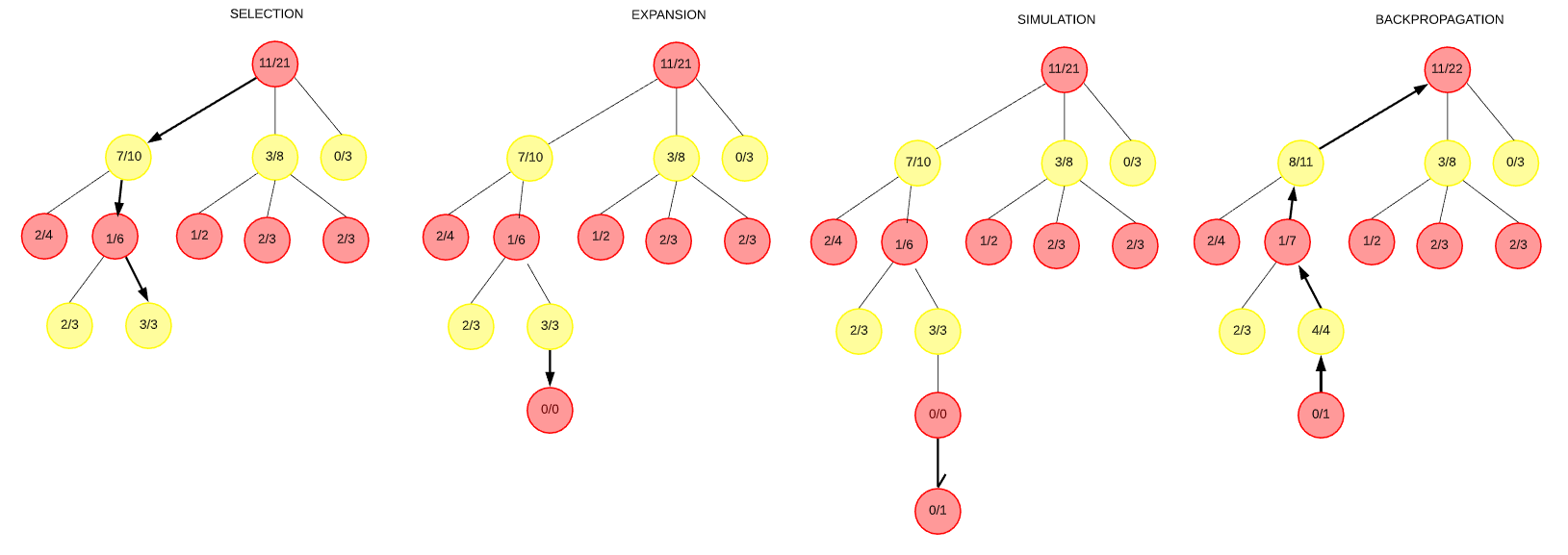


Figure 1. Pseudocode for UCT expression

The MCTS consists of four basic steps based on the random sampling of the search space.

1. *Selection* – The tree selects the node with the highest UCT value i.e., the tree selects the node which has the highest chance of winning after iterating from current state node to the selected node.
2. *Expansion* – The selected node is then expanded further creating multiple child nodes by randomly selecting one of the possible moves and expanding further until the leaf node is reached.
3. *Rollout* – The simulations, sometimes called playout or rollout, are run on the finally expanded node to check the status of the game – win, loss, or tie.
4. *Backpropagation* – The result from the rollout is updated for the nodes selected along the path i.e., the number of wins for the selected node, the total number of simulations ran on the selected node, and the total number of simulations on the parent node.

The working example of the steps in mentioned in Figure 2.



The pseudocode for the MCTS steps is shown in Figure 3.

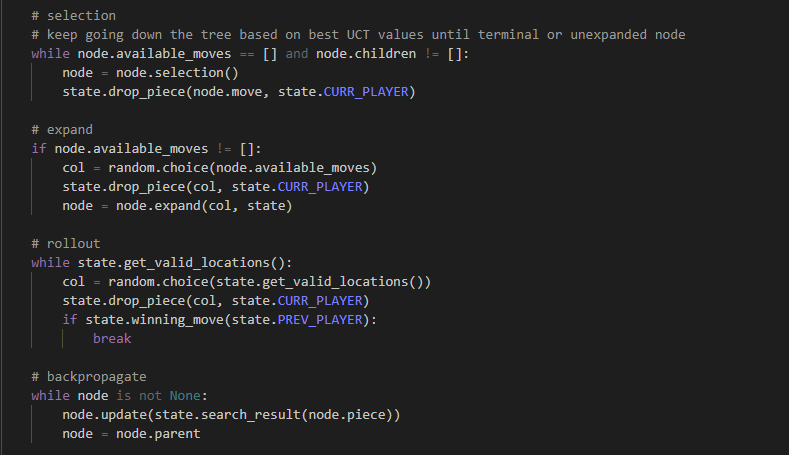


Figure 3. Pseudocode for MCTS steps

MUKESH TO PROVIDE since he is gonna write for the other two algorithms

Possible parameters – same for all the three algorithms and a default structure can be provided

IMPLEMENTATION –

writing it roughly here…

* Based on the number of iterations (if needed, write simulations) the model can perform better or worse than the other two
* But increasing the number of simulations would increase the computation power of the model resulting in a longer game time.

OTHER MODELS – minimax and expectimax

similar…

* increasing the depth of the model increases the chances of a sure win when pitting against the other models.

guess that’s it for implementation…

REFERENCES

[1] [The On-Line Encyclopedia of Integer Sequence (OEIS)](https://oeis.org/A212693)

[2] [Wiki. Monte Carlo tree search](https://en.wikipedia.org/wiki/Monte_Carlo_tree_search)