**GAME SEARCH TREE**

TIC-TAC-TOE

MINMAX

ALPHA-BETA PRUNING

def play\_game(game, strategies: dict, verbose=False):

Minimax-Based Game Search Algorithms

(

We use minimax\_search, which exhaustively searches the game tree to find an optimal move (assuming both players play optimally), and alphabeta\_search, which does the same computation, but prunes parts of the tree that could not possibly have an affect on the optimal move.

A white paper with writing on it

Description automatically generated with medium confidence

In this diagram illustrates it shows the position of all the places

TASK 1:

Tic-tac-toe by implementing the minimax search algorithm

Case 1: -1

Table

Description automatically generated with medium confidence

Case 2: 0

Table

Description automatically generated with low confidence

Table

Description automatically generated

**Tic-tac-toe game and add alpha-beta search algorithm to prune**

Alpha Beta

Case1: -1

Table

Description automatically generated with medium confidence

Text

Description automatically generated

TASK 4:

A picture containing text, whiteboard

Description automatically generatedHere in the above diagram , it shows in the branch of5,9 min is 5 but as 2nd row it takes minimum it chooses less than 3 but we have 5 so the 9 branch is prone i.e cut. But in the first row we want less than 3 so the >1 branch gets proned.

Similarly in 2nd row when comparing 3 and 1 , in the fourth tree, it is 1 as maximum,