Muhammad Junaid
0253-BSCS-17
E1

Artificial Intelligence(Ai)

Final Report:Tic tac toe(Minimax)

Language:JAVA(GUI Android app)

Government college University (GCU) Lahore

Tic Tac Toe using Minimax Algorithm

Tic Tac Toe : Tic tac toe is X's and O's is a paper and pencil game for two players *X* and *O*, who take turns marking the spaces in a 3×3 grid.

X		
O	**O**	**O**
X		

Winner O's

Win condition: In order to win the game ,a player(computer or human) must place three of their marks in horizontal or vertical or diagonal.

Draw Condition:If the both players (computer and human) do not meet any condition of win like 3 place horizontal , vertical,diagonal.

Minimax:

Minimax can be implemented using a recursive or backtracking algorithm which traverses a game tree consisting of possible moves and their outcomes. The root node of the tree represents the current game state. Each branch from a node represents a possible move a player could make. Each child node represents the resulting game state of a possible move.

Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that the opponent is also playing optimally.

The minimax algorithm performs a **depth-first search** algorithm for the exploration of the complete game tree.

The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

At the terminal node, the terminal values are given so we will compare those values and backtrack the tree until the initial state occurs. The algo generates the whole game-tree and apply utility function to get the values for the terminal states.

A **utility function** is used to judge terminal states and assign a numerical value to terminal states based on the rules of the game. If the terminal state represents a winning outcome for the player performing the search then it will be assigned a high value.If at terminal point losing outcome then it is assigned a negative value.Assigning value from terminal to root.This is Depth First Search and terminal values propagated up to the tree one level(depth+1).This is terminal to root process like reverse (depth -1).In every move check the condition of win or lose or draw.Once the values have been propagated up the tree, the child nodes of the root node are evaluated and the node with the highest value is selected as the best move to make.

Properties of Minimax algorithm:

- Complete- Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- Optimal- Min-Max algorithm is optimal if both opponents are playing optimally.
- Time complexity- As it performs DFS for the game-tree, so the time complexity of the Min-Max algorithm is O(b^m), where b is the branching factor of the game-tree, and m is the maximum depth of the tree.
- Space Complexity- Space complexity of Minimax algorithm is also similar to DFS which is O(bm).

Drawback :

It is slow for complex games such as chuss , because the branching factor is very large.This limitation of the minimax algorithm can be improved from **alpha-beta pruning**

Pseudocode for Minimax Algorithm:

```
function  minimax( int depth,int Player)    //player_ai= 1,player_human=2

                          int min=Integer.MAX_VALUE;
                         int max=Integer.MIN_VALUE;
                                //initial stage we assume min value is very large   because we can
                                easily find the minimum value .vice versa for max value.

                  If turn==player_ai /computer player or maximaxing player
                  {
                          placeAMove(location,player_ai);
```

```
                        int currentScore=minmax(depth+1,player_human);
                        max=Math.max(currentScore, max);

                       **if** depth == 0 **or** node is a terminal node **then**
                System.out.println("Computer Score for position"+point+"="+currentScore);
        Return max;
                            }
            else if  turn==player_human     //minimizing player
              {
                        placeAMove(locationt,player_human);
                        int currentScore=minmax(depth+1,player_ai);
            Return    min =Math.min(currentScore,min);


              }
```

Project Description:

How many classes,method we use in our project?

Class:**Location**

constructor(location x,location y)

Class:**Board**

Method:

isGameOver();

hasPlayerwon(int player);

getAvailable_cell();

placeAMove(location,player);

DisplayBoardvalue();

minmax(int depth,int player);

Class:**Tic tac Toe(Main Activity)**

Location:Location (constructor) is a class that stores the 2d array element in the form of x and y.

isGameOver();That returns the True or false value on the bases of player_Ai win or player_human win or any available location.

hasPlayerwon(int player): This method accepts a player and returns True or False value if the player is winning or not.

Winning possible is total 8,3time horizontal,3 time vertical,2time diagonal.

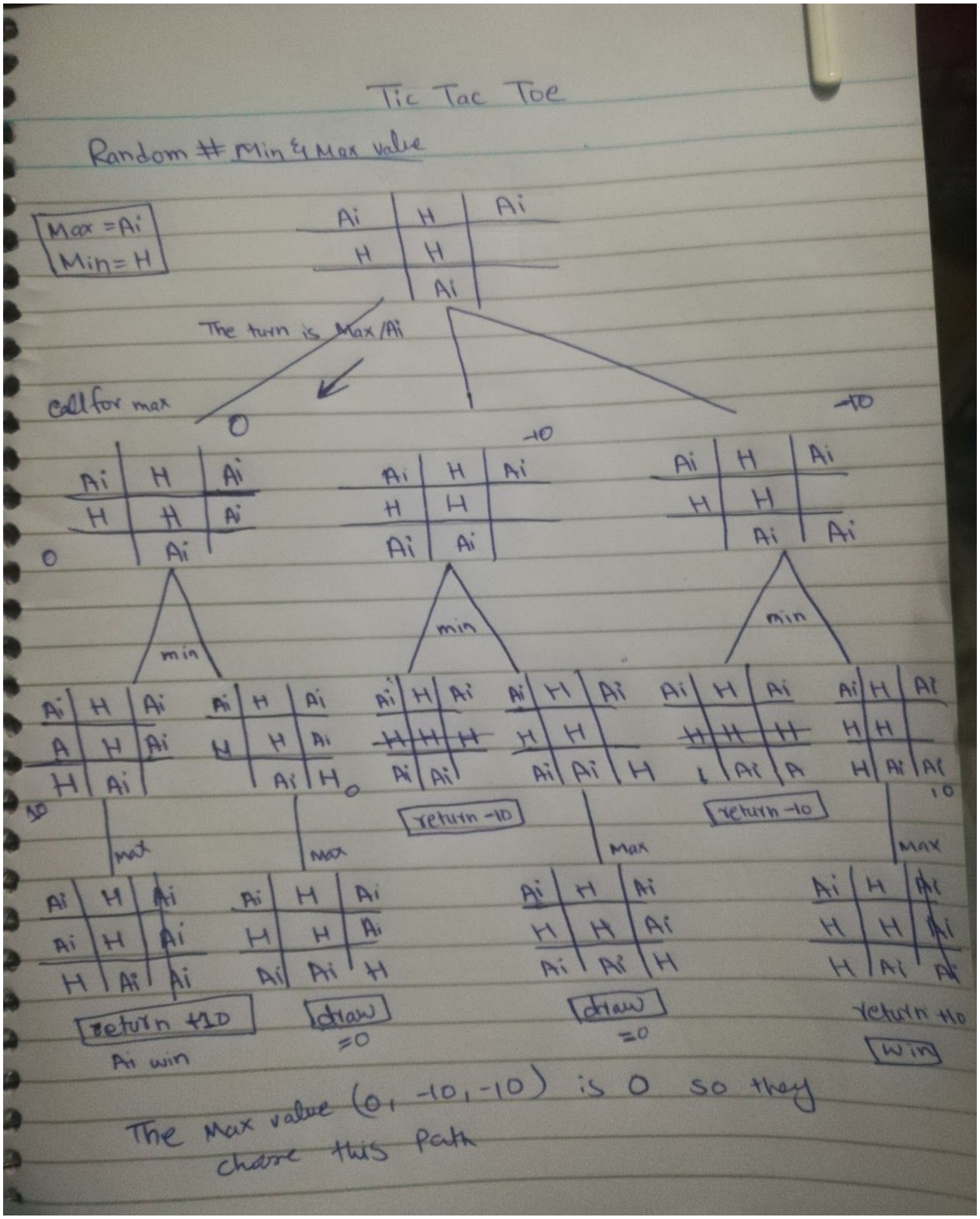
displayboard(): This method shows the Tic tac toe board and shows human and Ai input.

getAvaliablecell():This method returns the arraylist of Location that are available on board.

placeAMove(Location location,int player): This return true or false value if location is empty or not.If location is already filled by another player or Ai_player then return false.other than they place a location on board.

minimax(int depth, int turn): That return the min or max value .Minimax is a recursive function and calling for maximizer for max player then plays a minimizer for min player.Base of max player as Ai_player find the max value in the current score.In min player as human_player find the min value this method call recursively(repeated) until they reached at terminal stage.This is also DFS and generate a tree from depth 0 to n.When the reached at terminal level(leaves) than they assigned a value +10 if player_ai win and if player_human win assigned -10 value.Then they traverse to upward.

Tree of Tic Tac Toe:



Tic tac toe is based on Depth first Search ,minimax first travel from depth 0 to n at every level they check the condition of win or lose or draw.When they reached at terminal stage of any node if the player_ai win they assign a positive value or +10 or lose -10.child of node is compared if it is max player turn than they find the max value in child node and assign to parent node.If it the turn of min player than find the min value and assigned to parent node this is recursive process and call repeatedly.

Reference:

https://en.wikipedia.org/wiki/Minimax

https://www.tutorialspoint.com/listtutorial/MiniMax-Game-Trees/2316