3/12/2024

DATE: PAGE:

- Forward chaining, Algo for min max tic tac toe, Algo α-β pruning
Proof: queen.

* Forward chaining:-

Consider the following problem.

As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missibles, and all missibles were sold to it by Robert, who is an American citizen.

Prove Robert is criminal.

Proof:

Consider the sentence

It is a crime for an American to sell weapons to hostile nations.

Let's take three variables x, y, z.

$American(x) \wedge weapon(y) \wedge sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Country A, an enemy of America. and has some missibles.

$Enemy(America, A)$

$Owns(A, x) \wedge Missible(x)$

DATE: PAGE:

- All the ~~missible~~ missiles were sold to country A by Robert

$\forall x \text{Missile}(x) \wedge \text{Owns}(A, x) \Rightarrow \text{Sells}(\text{Robert}, x, A)$

- Missiles are weapons

$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$

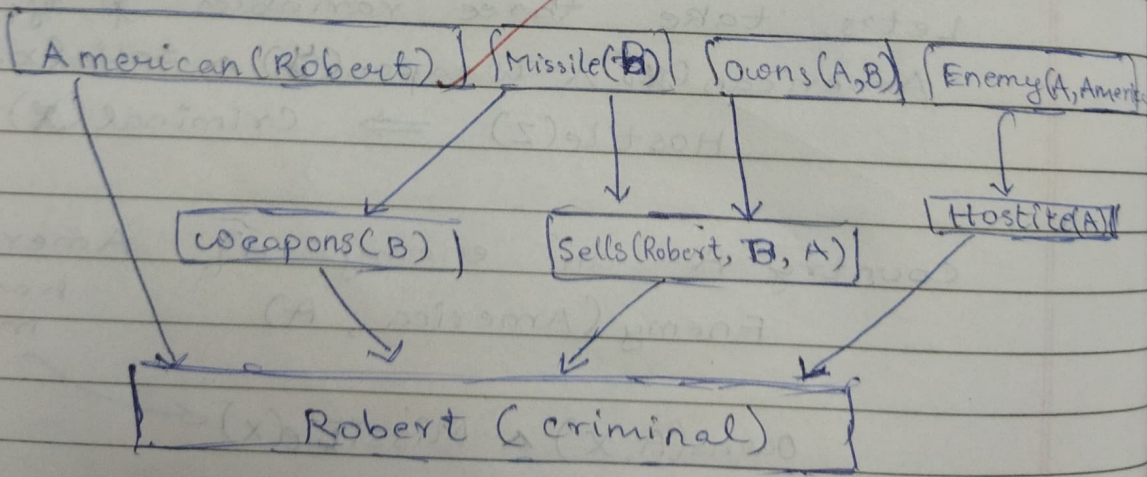
- Enemy of America is known as Hostile

$\forall x \text{ Enemy}(x, \text{America}) \rightarrow \text{Hostile}(x)$

∴ Robert is a criminal

$\text{Criminal}(\text{Robert})$

Forward chaining proof:



DATE: PAGE:

Min-Max Algorithm for Tic-Tac-Toe.

```
board = [[' ',' ',' '],[' ',' ',' '],[' ',' ',' ']]

function print_board(board):
    for row in board:
        print row

function check_winner(board)
    for row in board:
        if row[0] == row[1] == row[2] & row[0] != ' ':
            return row[0]

    for col in range(3):
        if board[0][col] == board[1][col]
            == board[2][col]  and
                board[0][col] != ' ':
            return  board[0][col]

    if board[0][0] == board[1][1] == board[2][2]
                and board[0][0] != ' ':
        return board[0][0]

    if board[0][2] == board[1][1] == b[2][0]
            return b[0][2]

    return None
```

DATE: PAGE:

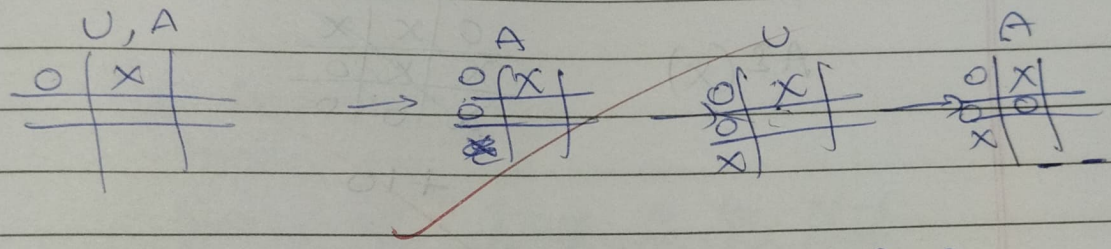
```
def is_full(board):
    for row in board:
        if ' ' in row:
            return False
    return True

def minimax(board, depth, is_maxi):
    win = check_winner(board)
    if win == 'X':
        return 10-depth
    elif win == 'O':
        return depth -10
    elif is_full(board):
        return 0

    if is_maxi:
        best_score = float('-inf')
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = 'X'
                    score = minimax
                        (board,
                        depth+1,
                        False)
                    best_score =
                    max(best_score,
                    score)

        return best_score
    else:
        best_score = float(inf)
        for i in range(3):
            for j in range(3):
```

```
if Board[i][j] == '':
    board[i][j] = 'O'
    score = minimax(board, depth+1, True)
return best_score
```

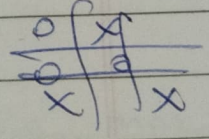
```
find best move ()
    for each empty cell
        if score > best_score
            best_score = score
            best_move = (i, j)
```

Output

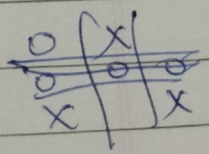
user - O AI - X



now AI can minimize the user's score



but



user win

* Alpha-Beta Pruning for 8-Queens.

DATE: PAGE:

```
function is_safe(board, row, col):
    for i from 0 to row-1:
        if board[i] == col or abs(board[i] - col) == abs(i-row):
            return False
    return True

function minimax(board, row, alpha, beta, is_maxi):
    if row == N
        return 1

    if is_maxi:
        max_score = 0
        for col from 0 to N-1:
            if is_safe(board, row, col):
                board[row] = col
                score = minimax(board, row+1, alpha, beta, False)
                max_score += score
                board[row] = -1
                alpha = max(alpha, max_score)
                if beta <= alpha:
                    break
        return max_score

    else:
        min_score = infinity
```

DATE:

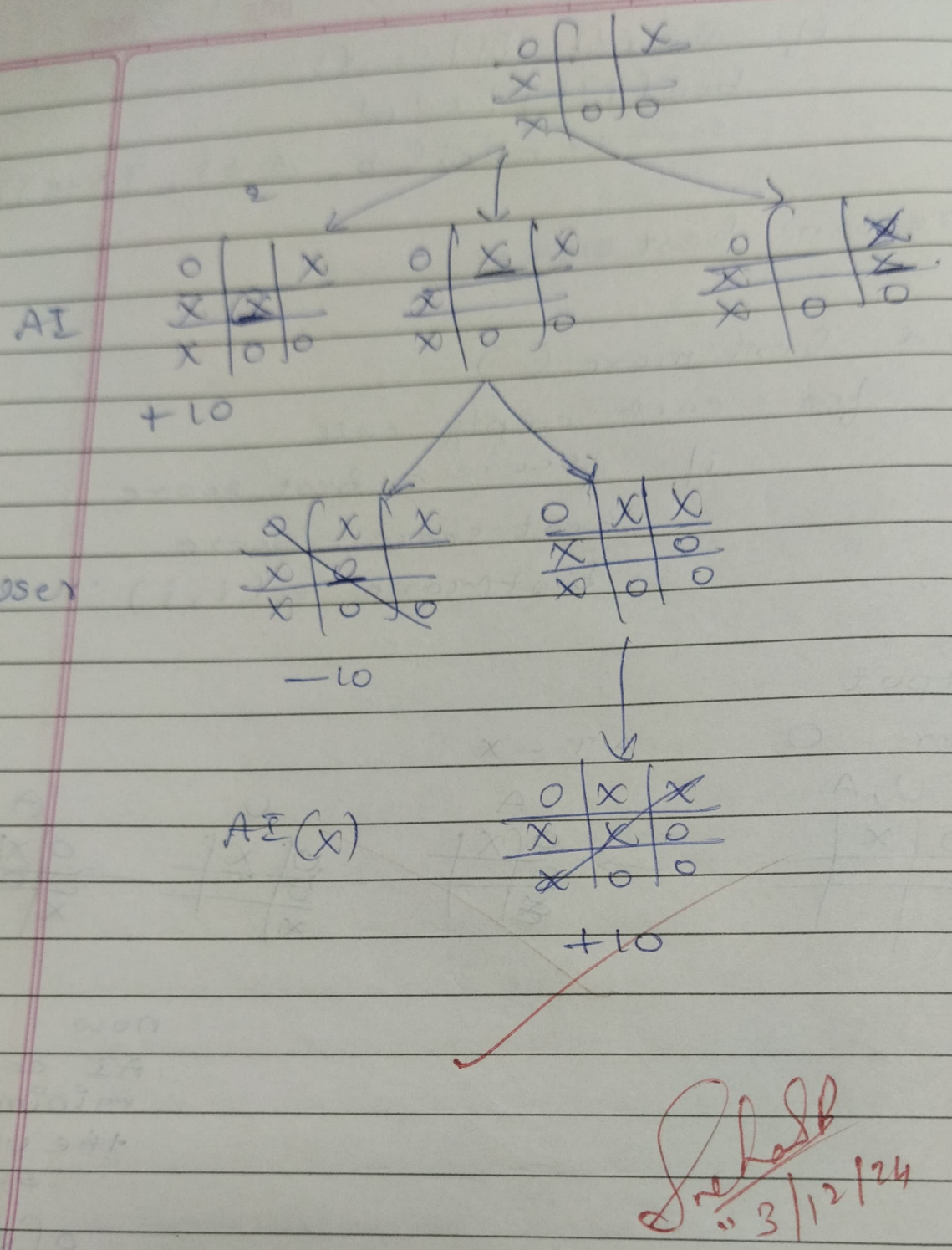
PAGE:

```
for col from 0 to N-1:
    if is_safe(board, row, col):
        board[row] = col
        score = minimax(board, row+1,
                alpha, beta, True)
        min_score = min(min_score,
                score)
        board[row] = -1
        beta = min(beta, min_
                score)
        if beta <= alpha:
            break

return min_score
```

output:-

```
_ _ Q _ _ _ _ _
_ _ _ _ _ Q _ _
_ _ _ _ _ _ _ Q
Q _ _ _ _ _ _ _
_ _ _ Q _ _ _ _
_ _ _ _ _ _ Q _
_ _ _ _ Q _ _ _
_ Q _ _ _ _ _ _
```

DATE:



3/12/24