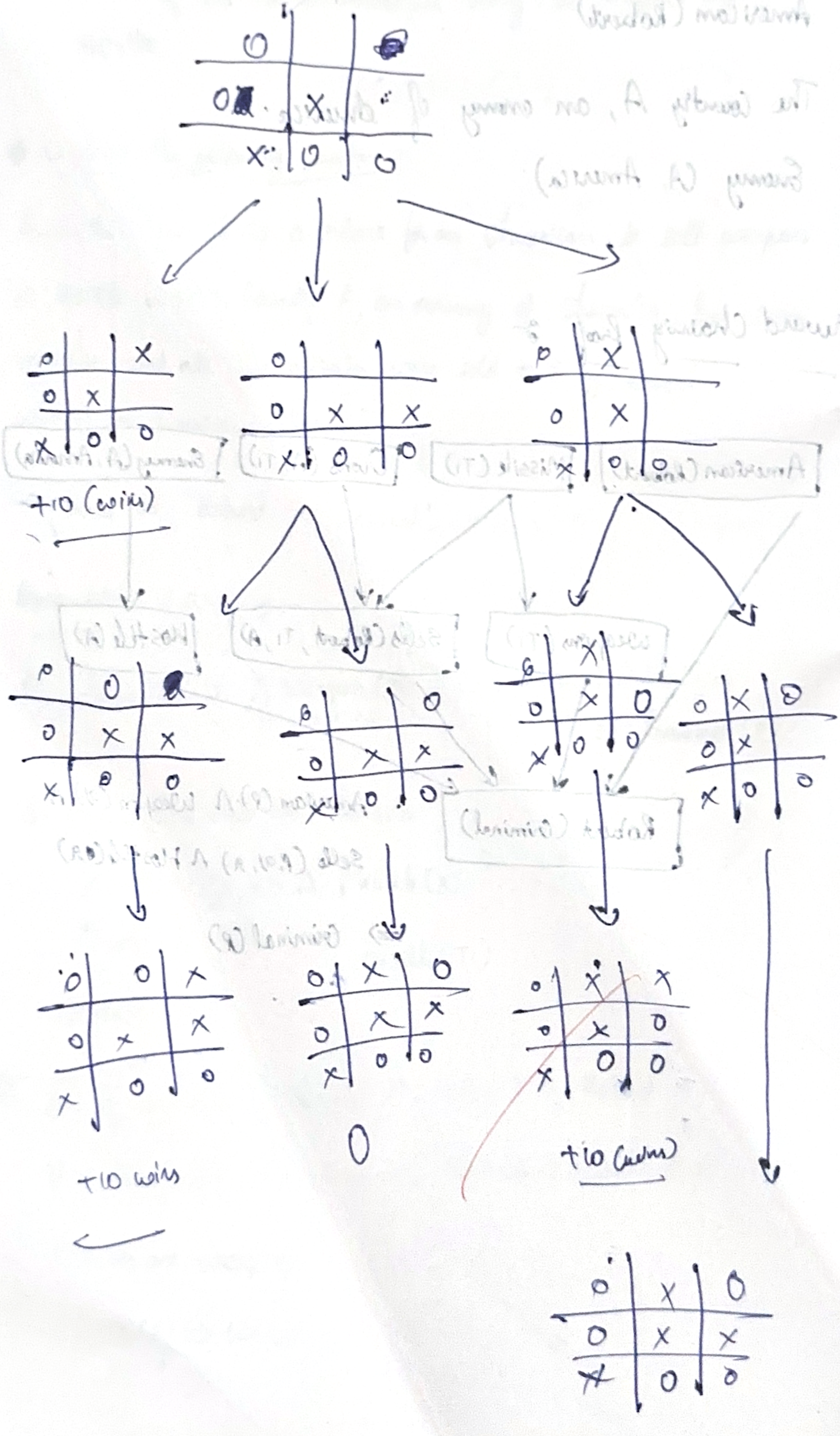
Lab-10

⇒ Tic Tac Toe using Min Max :



Algo :-

```
function minmax (board, depth, isMaximizing):

    if game_over(board):
        return evaluate(board)

    if isMaximizing:
        best_score = -∞
        for each empty cell (row, col) in board:
            simulate_move(board, row, col, 'x')
            score = minmax(board, depth+1, False)
            undo_move(board, row, col)
            best_score = max(best_score, score)
        return best_score

    else:
        best_score = +∞
        for each empty cell (row, col) in board:
            simulate_move(board, row, col, 'o')
            score = minmax(board, depth+1, True)
            undo_move(board, row, col)
            best_score = min(best_score, score)
        return best_score

function find_best_move(board, isMaximizing):
    best_move = (-1, -1)
    best_score = -∞ if isMaximizing else +∞
    for each empty cell (row, col) in board:
        simulate_move(board, row, col, 'x' if isMaximizing else 'o')
        move_score = minmax(board, 0, not isMaximizing)
        undo_move(board, row, col)
        if isMaximizing and move_score > best_score:
            best_score = move_score
            best_move = (row, col)
    return best_move
```

(2)

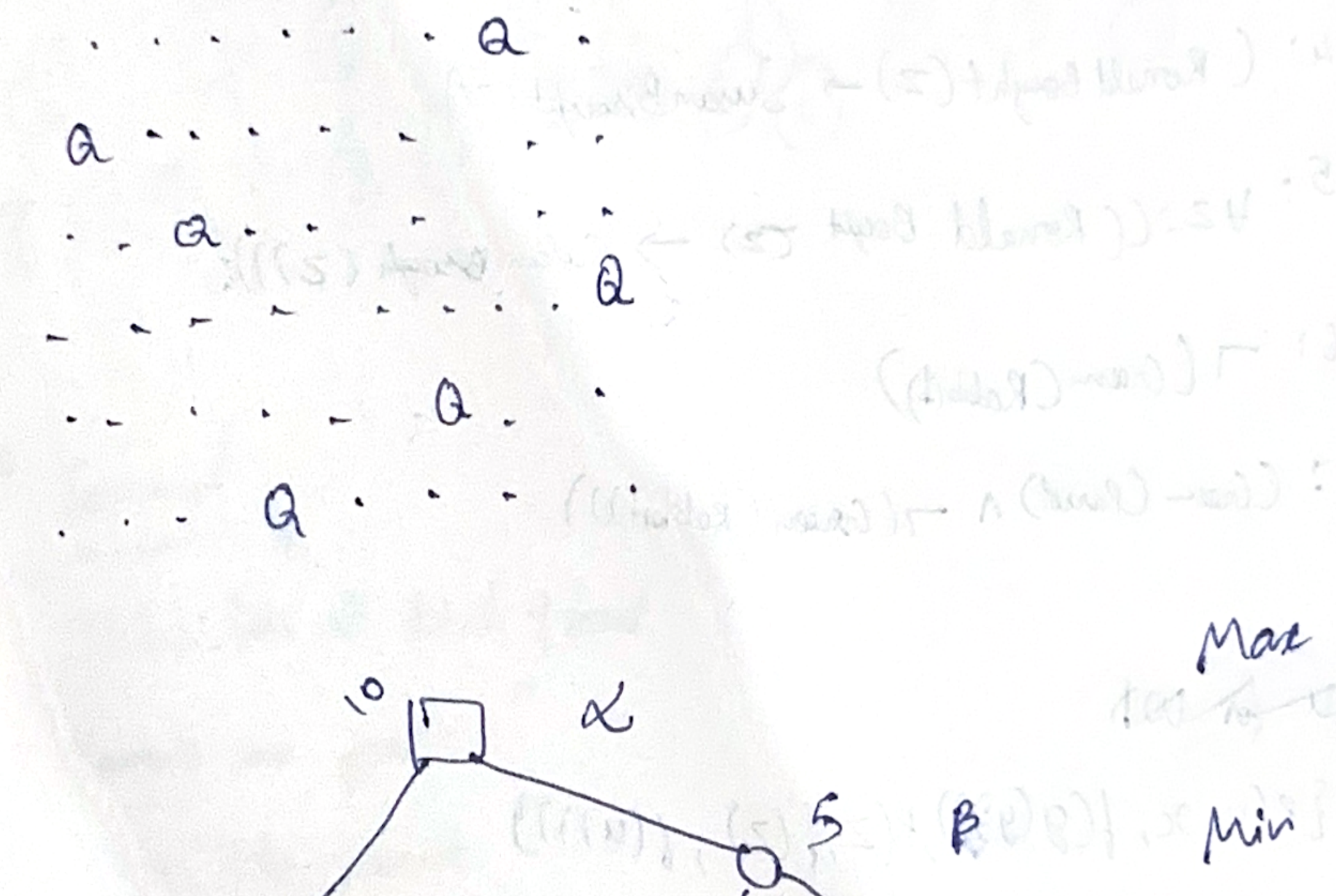
Solve 8-queens using alpha-beta. :-

```
def alpha_beta (self, board, col, alpha, beta, max_player):

    if col >= self.size:
        return 0, [row[:] for row in board]

    if max_player:
        max_eval = float('-inf')
        best_board = None
        for row in range (self.size):
            if self.is_safe (board, row, col):
                board[row][col] = 1

                eval_score, potential_board = self.alpha_beta_search
                                    (board, col+1, alpha, beta, False)
                board[row][col] = 0

                if eval_score > max_eval:
                    max_eval = eval_score
                    best_board = potential_board

                alpha = max (alpha, eval_score)
                if beta <= alpha
                    break;
        return max_eval, best_board.

    Else:
```

```
beta = min(beta, eval_score)
if beta <= alpha
    break
return max_eval, best_board
```

Output:

```
. Q . . . . . .
. . . . Q . . .
. . . . . . Q .
Q . . . . . . .
. . Q . . . . .
. . . . . . . Q
. . . . . Q . .
. . Q . . . . .
```



Sm
3/12/24