## A Star Algorithm - Tic Tac Toe game

Code:

In [6]:

```
player, opponent = 'x', 'o'

def isMovesLeft(board) :
    for i in range(3) :
        for j in range(3) :
            if (board[i][j] == '_') :
                return True
    return False

def evaluate(b) :
    for row in range(3) :
        if (b[row][0] == b[row][1] and b[row][1] == b[row][2]) :
            if (b[row][0] == player) :
                return 10
            elif (b[row][0] == opponent) :
                return -10

    for col in range(3) :
        if (b[0][col] == b[1][col] and b[1][col] == b[2][col]) :
            if (b[0][col] == player) :
                return 10
            elif (b[0][col] == opponent) :
                return -10

    if (b[0][0] == b[1][1] and b[1][1] == b[2][2]) :
        if (b[0][0] == player) :
            return 10
        elif (b[0][0] == opponent) :
            return -10

    if (b[0][2] == b[1][1] and b[1][1] == b[2][0]) :
        if (b[0][2] == player) :
            return 10
        elif (b[0][2] == opponent) :
            return -10

    return 0

def minimax(board, depth, isMax) :
    score = evaluate(board)
    if (score == 10) :
        return score

    if (score == -10) :
        return score

    if (isMovesLeft(board) == False) :
        return 0

    if (isMax) :
        best = -1000
        for i in range(3) :
            for j in range(3) :
                if (board[i][j]=='_') :
                    board[i][j] = player
                    best = max( best, minimax(board,
                                            depth + 1,
                                            not isMax) )
                    board[i][j] = '_'
        return best

    else :
        best = 1000
        for i in range(3) :
            for j in range(3) :
                if (board[i][j] == '_') :
                    board[i][j] = opponent
                    best = min(best, minimax(board, depth + 1, not isMax))
                    board[i][j] = '_'
        return best

def findBestMove(board) :
    bestVal = -1000
    bestMove = (-1, -1)
    for i in range(3) :
        for j in range(3) :
            if (board[i][j] == '_') :
                board[i][j] = player
                moveVal = minimax(board, 0, False)
                board[i][j] = '_'
                if (moveVal > bestVal) :
                    bestMove = (i, j)
                    bestVal = moveVal

    print("The value of the best Move is :", bestVal)
    print()
    return bestMove

board = [
    [ 'x', 'o', 'x' ],
    [ 'o', 'o', 'x' ],
    [ '_', '_', '_' ]
]

bestMove = findBestMove(board)
print("The Optimal Move is :")
print("ROW:", bestMove[0]+1, " COL:", bestMove[1]+1)


#Output
```

```
The value of the best Move is : 10

The Optimal Move is :
ROW: 3  COL: 3
```