8 queens using alpha beta Pruning search algorithm.

Algorithm

1) Define a function init with self and size.
2) Define a another function is_safe with parameters with self, board, row and column.
3) Using a zip combine a row and column
4) Define a function a alpha beta search with parameters self, board, col, alpha, beta, maxi_player.

if col >= self.size.

return 0, [row[i] for row in board]

5) check the condition maximizing_player. in that check the is_safe call a function alpha_beta_search function.

DATE: PAGE:

if eval_score > max_eval:
    max_eval = eval_score
    best_board = potential_board.

6) define a function solve to call the function
7) print the board:
8) it will print the solution if found or else print no solution found.

Passed

min max algorithm for tic tac toe

1) Import a math & library.
2) Take a AI='X', HUMAN='O', EMPTY='-'.
3) define a function printboard. checks the winner player X or O has won.
4) Check the board is full and no winner exists.
5) Define a function minmax. with parameter board. depth.
6) AI → positive score (10-depth)
   Human = negative score (depth-10)
   .. draw: Return 0
7) AI attempts to maximize its score by choosing moves that lead to better outcomes.
   simulates all possible moves and recursively evaluates them using minmax
Human: player minimizes the score, assuming they play optimally.
8) loops through all empty cells. places the AI's move in an empty cell & evaluates it using the minimax function.

DATE: PAGE:

tracks the moves with the highest score
a) Print the solutions.

AI = 10
Human = -10.

→ Output

Current board:

X O X

O X O

_ _ _.

→ Output:

Solution found

. Q . . . . . .

. . . Q . . . .

. . . . . Q . .

Q . . . . . .

. . Q . . . . .

. . . . Q . . .

. Q . . . . . .

→ Output:

→ Robert is a criminal

Proved
FOL
9/12/24