8-Queens Problem

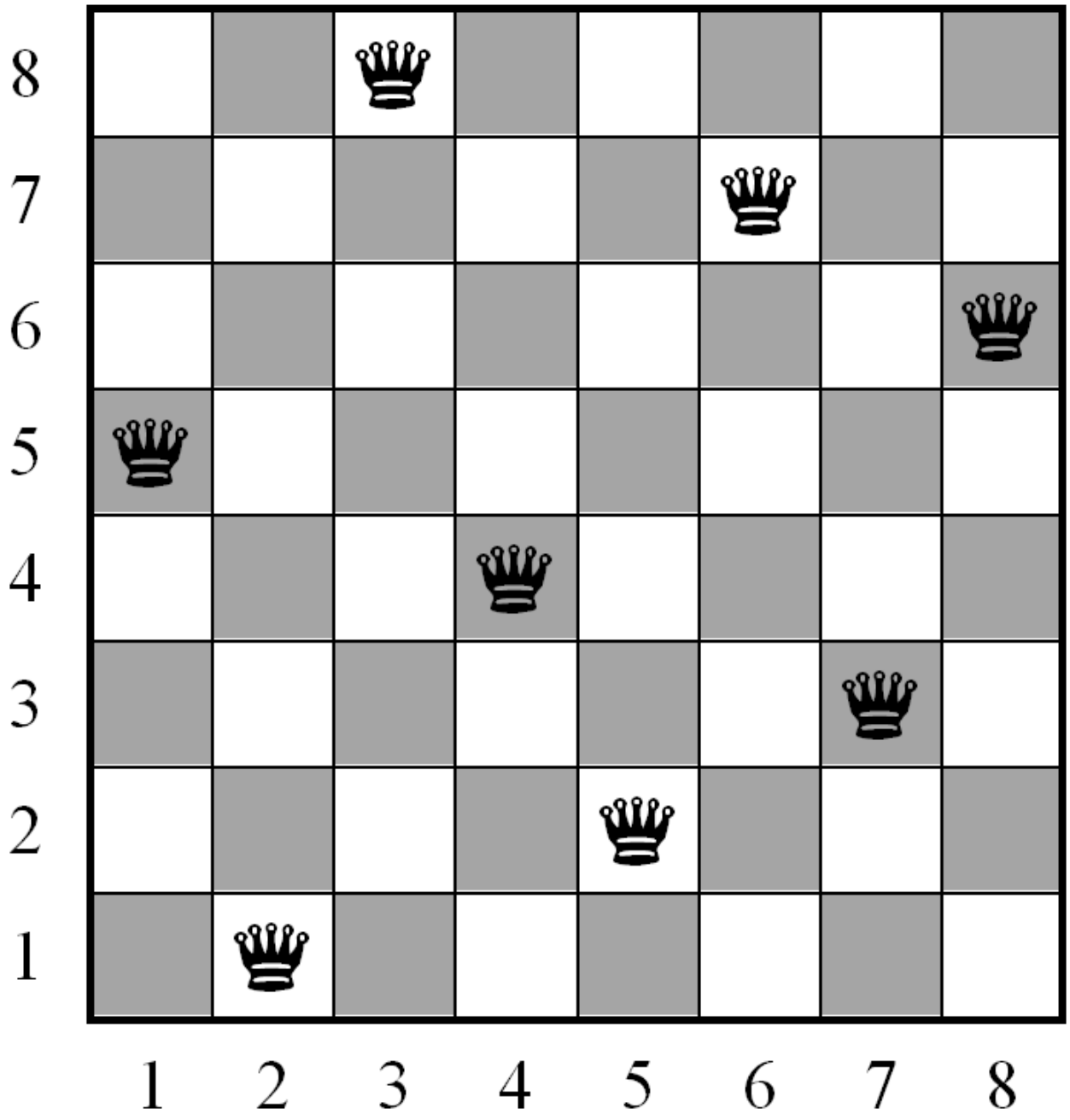
Explanation

The 8-Queens Problem is a classic AI problem where the goal is to place 8 queens on a chessboard (8×8) such that no two queens attack each other.

Queens can attack:

- Horizontally
- Vertically
- Diagonally

So, the goal is to place one queen per row (or column), ensuring that no queen threatens any other.


8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8

Algorithm Used: Backtracking

Backtracking is a depth-first search algorithm that:

1. Places a queen in a row.
2. Moves to the next row if placement is safe.
3. If no position is valid in a row, backtracks to the previous row and tries the next position.

Algorithm Steps

1. Start from the 1st row.
2. Try placing a queen in each column.
3. If placing the queen is safe, place it and move to the next row.
4. If not safe or no column left, backtrack to the previous row.
5. Repeat until all 8 queens are placed.

Python Program Using Backtracking

```
N = 8

def print_solution(board):
    for row in board:
        print(" ".join("Q" if cell else "." for cell in row))
    print()

def is_safe(board, row, col):
    # Check this column on upper side
    for i in range(row):
        if board[i][col]:
            return False

    # Check upper-left diagonal
    for i, j in zip(range(row - 1, -1, -1), range(col - 1, -1, -1)):
        if board[i][j]:
            return False

    # Check upper-right diagonal
    for i, j in zip(range(row - 1, -1, -1), range(col + 1, N)):
        if board[i][j]:
```

```
            return False

    return True

def solve(board, row):
    if row == N:
        print_solution(board)
        return True  # If you want only one solution, return True here

    for col in range(N):
        if is_safe(board, row, col):
            board[row][col] = 1
            if solve(board, row + 1):
                return True  # comment this if you want all solutions
            board[row][col] = 0  # BACKTRACK
    return False

# Initialize board
board = [[0 for _ in range(N)] for _ in range(N)]
solve(board, 0)
```

Example Output

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

AI Applications

1. **Constraint Satisfaction Problems (CSPs)**
    - Timetabling, Scheduling, Resource allocation.
2. **Backtracking and DFS strategies**
    - Used in route finding, puzzles, and logical reasoning.
3. **Game Development**
    - Optimization of movement and placement logic.

4. **AI Planning and Robotics**
    - Task placement and conflict resolution.
5. **Genetic Algorithms & Heuristics**
    - 8-Queens used as a testing problem in heuristic optimization.