1. Explore the search space using search strategies and formulate the problem components for the 8-Queens problem using the following information: Place 8 queens on a chessboard such that none of the queen's attack any of the others. A configuration of 8 queens on the board as shown in the figure below, but this does not represent a solution as the queen in the first column is on the same diagonal as the queen in the last column.

Program :

```
def is_safe(board, row, col, n):
    # Check if there is a queen in the same row
    for i in range(col):
        if board[row][i] == 1:
            return False

    # Check if there is a queen in the upper diagonal on the left
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    # Check if there is a queen in the lower diagonal on the left
    for i, j in zip(range(row, n, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def solve_queens(board, col, n):
    if col >= n:
        return True  # All queens are placed successfully
```

```
    for i in range(n):
        if is_safe(board, i, col, n):
            board[i][col] = 1  # Place the queen

            if solve_queens(board, col + 1, n):
                return True  # If placing queen in current position leads to a solution

            board[i][col] = 0  # Backtrack if placing the queen doesn't lead to a solution

    return False  # If no placement in this column leads to a solution

def print_board(board):
    for row in board:
        print(' '.join(['Q' if cell == 1 else '.' for cell in row]))

def main():
    n = 8  # Size of the chessboard
    board = [[0] * n for _ in range(n)]

    if solve_queens(board, 0, n):
        print("Solution found:")
        print_board(board)
    else:
        print("No solution found.")
```

```
if __name__ == "__main__":
    main()
```

Output :

```
Solution found:
Q . . . . . . .
. . . . . . Q .
. . . . Q . . .
. . . . . . . Q
. Q . . . . . .
. . . Q . . . .
. . . . . Q . .
. . Q . . . . .
```