# 8 Queens problem

## Aim:

To place 8 queens on an 8x8 chessboard such that no two queens threaten each other. This means no two queens share the same row, column, or diagonal.

## Procedure:

1. Start placing queens one by one in different rows.
2. For each queen, check if the position is safe:
   * No other queen is in the same column.
   * No other queen is on the same diagonal.
3. If safe, place the queen and move to the next row.
4. If no position is safe in the current row, backtrack:
   * Remove the previous queen.
   * Try the next position for the previous queen.
5. Repeat until all queens are placed or all possibilities are exhausted.

## Code:

def is\_safe(board, row, col, n):

for i in range(row): # Check column

if board[i][col] == 1:

return False

for i, j in zip(range(row, -1, -1), range(col, -1, -1)): # Check upper left diagonal

if board[i][j] == 1:

return False

for i, j in zip(range(row, -1, -1), range(col, n)): # Check upper right diagonal

if board[i][j] == 1:

return False

return True

def solve\_n\_queens\_util(board, row, n):

if row >= n: # Base case: all queens are placed

print\_board(board, n)

return

for col in range(n):

if is\_safe(board, row, col, n):

board[row][col] = 1 # Place queen

solve\_n\_queens\_util(board, row + 1, n) # Recurse

board[row][col] = 0 # Backtrack

def solve\_n\_queens(n):

board = [[0 for \_ in range(n)] for \_ in range(n)]

solve\_n\_queens\_util(board, 0, n)

def print\_board(board, n):

for row in board:

print(" ".join("Q" if cell == 1 else "." for cell in row))

print()

solve\_n\_queens(8)

## Output:

**Q . . . . . . .**

**. . . . Q . . .**

**. . . . . . Q .**

**. . . Q . . . .**

**. Q . . . . . .**

**. . . . . Q . .**

**. . Q . . . . .**

**. . . . . . . Q**

## Result:

The algorithm will find all valid configurations of 8 queens on the chessboard. If you print the board, you'll see solutions where queens are positioned safely.