Backtracking is a general algorithmic technique used to solve problems by incrementally building a solution and then undoing the choices when they are found to be incorrect. It is often used for solving problems with constraints or searching for all possible solutions.

The N-Queens problem is a classic puzzle that involves placing N queens on an N×N chessboard such that no two queens threaten each other. In other words, no two queens should be in the same row, column, or diagonal.

Here's an implementation of the N-Queens problem using the backtracking technique in Python:def is\_safe(board, row, col, N):

# Check if it's safe to place a queen at board[row][col]

# Check this row on the left side

for i in range(col):

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

return False

# Check upper diagonal on the left side

i = row

j = col

while i >= 0 and j >= 0:

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

return False

i -= 1

j -= 1

# Check lower diagonal on the left side

i = row

j = col

while j >= 0 and i < N:

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

return False

i += 1

j -= 1

return True

def solve\_n\_queens\_util(board, col, N):

# Base case: If all queens are placed, return True

if col >= N:

return True

# Consider this column and try placing a queen in all rows one by one

for i in range(N):

if is\_safe(board, i, col, N):

# Place the queen

board[i][col] = 1

# Recur to place the rest of the queens

if solve\_n\_queens\_util(board, col + 1, N):

return True

# If placing the queen in board[i][col] doesn't lead to a solution, backtrack and remove the queen

board[i][col] = 0

# If the queen can't be placed in any row of this column, return False

return False

def solve\_n\_queens(N):

# Create an empty N x N chessboard

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

if not solve\_n\_queens\_util(board, 0, N):

print("No solution exists.")

else:

# Print the solution

for i in range(N):

for j in range(N):

print(board[i][j], end=" ")

print()

# Example usage

N = 4

solve\_n\_queens(N)