**Practical no.04**

# Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.

def print\_board(board):

for row in board:

print(" ".join("Q" if col else "." for col in row))

print("\n")

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

for i in range(row):

if board[i][col]:

return False

for i, j in zip(range(row, -1, -1), range(col, -1, -1)):

if board[i][j]:

return False

for i, j in zip(range(row, -1, -1), range(col, n)):

if board[i][j]:

return False

return True

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

if row == n:

print\_board(board)

return True

found\_solution = False

for col in range(n):

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

board[row][col] = True

found\_solution = solve\_n\_queens\_backtracking(board, row + 1, n) or found\_solution

board[row][col] = False

return found\_solution

def n\_queens\_backtracking(n):

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

if not solve\_n\_queens\_backtracking(board, 0, n):

print("No solution exists.")

# Branch and Bound Approach

def is\_safe\_branch\_and\_bound(row, col, cols, diags1, diags2, n):

return not (cols[col] or diags1[row + col] or diags2[row - col + (n - 1)])

def solve\_n\_queens\_branch\_and\_bound(row, n, cols, diags1, diags2, board):

if row == n:

print\_board(board)

return True

found\_solution = False

for col in range(n):

if is\_safe\_branch\_and\_bound(row, col, cols, diags1, diags2, n):

board[row][col] = True

cols[col] = True

diags1[row + col] = True

diags2[row - col + (n - 1)] = True

found\_solution = solve\_n\_queens\_branch\_and\_bound(row + 1, n, cols, diags1, diags2, board) or found\_solution

board[row][col] = False

cols[col] = False

diags1[row + col] = False

diags2[row - col + (n - 1)] = False

return found\_solution

def n\_queens\_branch\_and\_bound(n):

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

cols = [False] \* n

diags1 = [False] \* (2 \* n - 1)

diags2 = [False] \* (2 \* n - 1)

if not solve\_n\_queens\_branch\_and\_bound(0, n, cols, diags1, diags2, board):

print("No solution exists.")

# Example usage

print("Backtracking Solution:")

n\_queens\_backtracking(4)

print("Branch and Bound Solution:")

n\_queens\_branch\_and\_bound(4)