**Name :** Atharva Arjun Goral

**Roll No :** 3101033

**Batch :** B  **Div :** A

## **AI**

# **Assignment No. 3**

**CODE :**

import time

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

for i in range(col):

if board[row][i]:

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, n, 1), range(col, -1, -1)):

if board[i][j]:

return False

return True

def solve\_nqueens\_backtracking(board, col, n):

if col >= n:

return True

for i in range(n):

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

board[i][col] = 1

if solve\_nqueens\_backtracking(board, col + 1, n):

return True

board[i][col] = 0

return False

def solve\_nqueens\_branch\_and\_bound(n):

def solve(col, left\_diag, right\_diag, rows):

if col == n:

return True

for i in range(n):

if not rows[i] and not left\_diag[i + col] and not right\_diag[i - col + n - 1]:

board[i][col] = 1

rows[i] = left\_diag[i + col] = right\_diag[i - col + n - 1] = True

if solve(col + 1, left\_diag, right\_diag, rows):

return True

board[i][col] = 0

rows[i] = left\_diag[i + col] = right\_diag[i - col + n - 1] = False

return False

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

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

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

rows = [False] \* n

if solve(0, left\_diag, right\_diag, rows):

return board

return None

def print\_board(board):

if board:

for row in board:

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

else:

print("No solution found.")

if \_\_name\_\_ == "\_\_main\_\_":

n = int(input("Enter the number of queens: "))

print("Solving using Backtracking...")

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

start\_time = time.time()

if solve\_nqueens\_backtracking(board, 0, n):

print\_board(board)

else:

print("No solution found.")

print("Time taken (Backtracking):", time.time() - start\_time, "seconds")

print("\nSolving using Branch and Bound...")

start\_time = time.time()

board = solve\_nqueens\_branch\_and\_bound(n)

print\_board(board)

print("Time taken (Branch and Bound):", time.time() - start\_time, "seconds")

**OUTPUT :**

Enter the number of queens: 5

Solving using Backtracking...

Q . . . .

. . . Q .

. Q . . .

. . . . Q

. . Q . .

Time taken (Backtracking): 0.0004515647888183594 seconds

Solving using Branch and Bound...

Q . . . .

. . . Q .

. Q . . .

. . . . Q

. . Q . .

Time taken (Branch and Bound): 0.0004980564117431641 seconds