

## 

***N Queen Problem***

*Result File*

**LAB 04: N Queen Problem**

**Introduction**

The **N-Queens problem** is a classic combinatorial optimization problem that involves placing **N queens** on an **N × N chessboard** such that no two queens attack each other. This project implements a **backtracking algorithm** to find a solution and count the number of possible solutions for a given board size.

**Code Explanation**

**1. N-Queens Solution Using Backtracking**

The function N\_Queen(n) initializes an **N × N board** and uses a **backtracking approach** to place queens on the board.

The function safe(board, row, col) checks whether placing a queen at a specific position is valid by ensuring that:

* + No queen exists in the same **row**.
  + No queen exists in the same **upper diagonal**.
  + No queen exists in the same **lower diagonal**.

If a valid placement is found, the algorithm proceeds recursively until all queens are placed.

If a solution exists, it returns the **positions of the queens**.

**2. Displaying the Solution**

The function print\_board(solution, n) prints the board with **'Q' representing a queen** and **'.' representing an empty space**.

If no solution exists, a message is displayed.

**3. Counting the Total Number of Solutions**

The function count\_the\_solution(n) counts the total number of valid solutions using a modified **backtracking approach**.

It ensures that no two queens attack each other by checking column conflicts and diagonal conflicts.

It returns the **total number of valid board arrangements**.

**4. User Input and Execution**

The program prompts the user to enter the board size **(N × N)**.

The user can choose to:

* + **View a solution** (S)
  + **Count the number of solutions** (C)
  + **Do both** (B)

The program then executes the selected option and displays the results.

**Conclusion**

This project successfully implements the **N-Queens problem using backtracking**. The algorithm efficiently finds a valid solution and counts the total number of valid queen placements. The approach demonstrates how backtracking can be used to solve constraint-based optimization problems.

The output of this code is given below:

