Experiment Number: 7

Problem Statement:

WAP and perform Time complexity analysis of N queen problem using Backtracking strategy.

Also show on paper how backtracking is done with an example. i.e. show nested recursive calls

NAME: Harshwardhan Patil ROLLNO: 50

CLASS: IT-B BATCH: 01

DATE OF PERFORMANCE: 24 – March – 2025

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code –**

class NQueens {

    static final int N = 5;

    static int solutionCount = 1;

    static void printSolution(int board[][]) {

        System.out.println("Solution " + solutionCount + ":");

        solutionCount++;

        int[] solution = new int[N];

        for (int i = 0; i < N; i++) {

            for (int j = 0; j < N; j++) {

                if (board[i][j] == 1) {

                    solution[i] = j;

                    break;

                }

            }

        }

        System.out.print("[ ");

        for (int i = 0; i < N; i++) {

            System.out.print(solution[i] + " ");

        }

        System.out.println("]");

        for (int i = 0; i < N; i++) {

            for (int j = 0; j < N; j++) {

                if (board[i][j] == 1) {

                    System.out.print("Q\t");

                } else {

                    System.out.print("-1\t");

                }

            }

            System.out.println();

        }

        System.out.println();

    }

    static boolean isSafe(int board[][], int row, int col) {

        int i, j;

        // Check this column (above rows)

        for (i = 0; i < row; i++)

            if (board[i][col] == 1)

                return false;

        // Check upper-left diagonal

        for (i = row, j = col; i >= 0 && j >= 0; i--, j--)

            if (board[i][j] == 1)

                return false;

        // Check upper-right diagonal

        for (i = row, j = col; i >= 0 && j < N; i--, j++)

            if (board[i][j] == 1)

                return false;

        return true;

    }

    static void solveNQueens(int board[][], int row) {

        if (row >= N) {

            printSolution(board);

            return;

        }

        for (int col = 0; col < N; col++) {

            if (isSafe(board, row, col)) {

                board[row][col] = 1;

                solveNQueens(board, row + 1);

                board[row][col] = 0; // Backtrack

            }

        }

    }

    public static void main(String args[]) {

        int board[][] = new int[N][N];

        solveNQueens(board, 0);

        if (solutionCount == 0) {

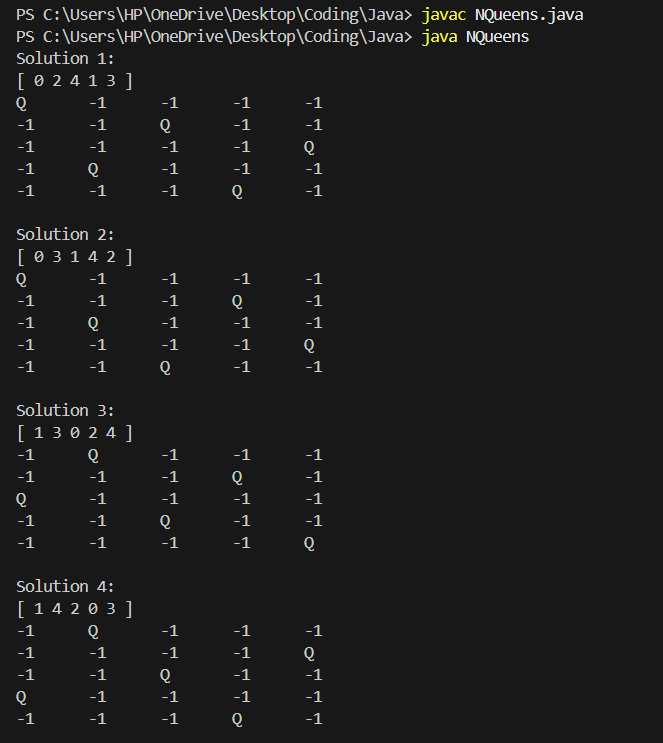
            System.out.println("No solution exists.");

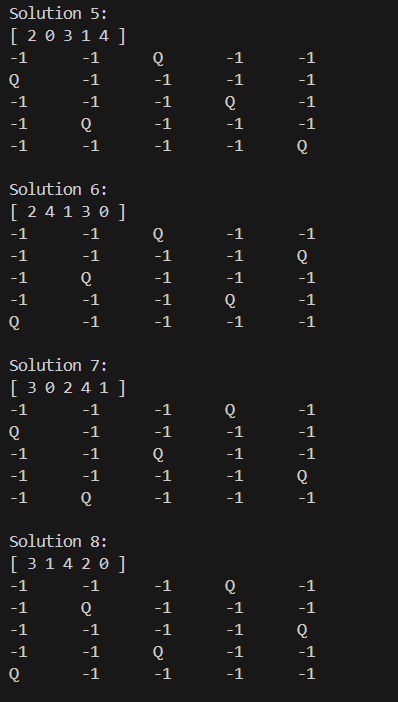
        }

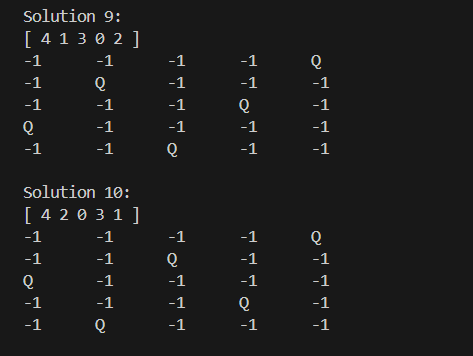
    }

}

**Output –**

****

****

****

**Write up –**

