Code for N-Queen Algorithm

import java.util.\*;

class EXPT\_08\_NQueens\_2 {

    static int counter = 0;

    private static boolean position(int mat[][], int row, int col) {

        // false if two queens are in the same column

        for (int i = 0; i < row; i++) {

            if (mat[i][col] == 1) {

                return false;

            }

        }

        // false if two queens share the same `/` diagonal

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

            if (mat[i][j] == 1) {

                return false;

            }

        }

        // false if two queens share the same `\` diagonal

        for (int i = row, j = col; i >= 0 && j < mat.length; i--, j++) {

            if (mat[i][j] == 1) {

                return false;

            }

        }

        // true if none of the above conditions occur

        return true;

    }

    private static void print(int mat[][]) {

        int i = 1;

        for (int row = 0; row < mat.length; row++) {

            System.out.print("\n");

            if (row % 8 == 0) {

                counter++;

                System.out.print("\nSolution " + counter + " :\n");

            }

            for (int col = 0; col < mat[i].length; col++) {

                System.out.print(mat[row][col] + "\t");

            }

            System.out.print("\n");

        }

    }

    private static void place\_nqueens(int mat[][], int row) {

        // if n queens are placed successfully, the solution is printed

        if (row == mat.length) {

            print(mat);

            return;

        }

        // placing queens in the chessboard

        for (int i = 0; i < mat.length; i++) {

            // if no two queens threaten each other

            if (position(mat, row, i)) {

                // place the queen on the current square

                mat[row][i] = 1;

                // recur for the next row

                place\_nqueens(mat, row + 1);

                // backtrack and remove the queen from the current square

                mat[row][i] = 0;

            }

        }

    }

    public static void main(String[] args) {

        // `N × N` chessboard

        Scanner in = new Scanner(System.in);

        System.out.print("\nEnter value of N: ");

        int N = in.nextInt();

        // `mat[][]` keeps track of the position of queens in

        // the current configuration

        int[][] mat = new int[N][N];

        // initialize `mat[][]` by `0`

        for (int i = 0; i < N; i++) {

            // Arrays.fill(mat[i], '-');

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

                mat[i][j] = 0;

            }

        }

        long start = System.currentTimeMillis();

        place\_nqueens(mat, 0);

        long end = System.currentTimeMillis();

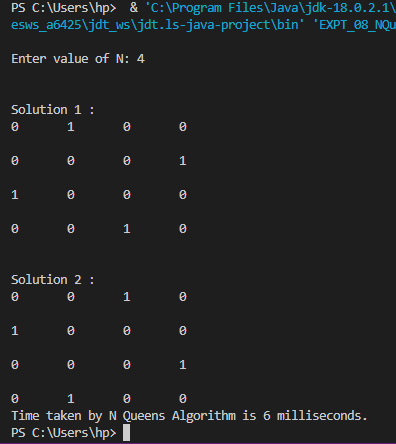
        System.out.println("Time taken by N Queens Algorithm is " + (end - start) + " milliseconds.");

    }

}

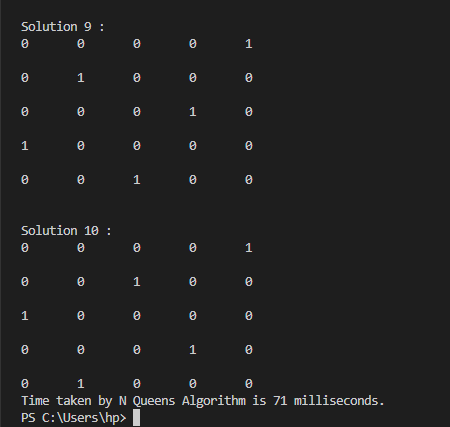
Case 1: When the value of N is 4

Time taken is 6 milliseconds.



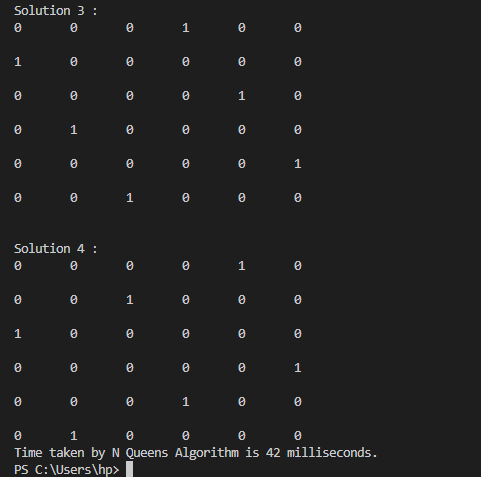
Case 2: When the value of N is 5.

Time taken is 71 milliseconds.

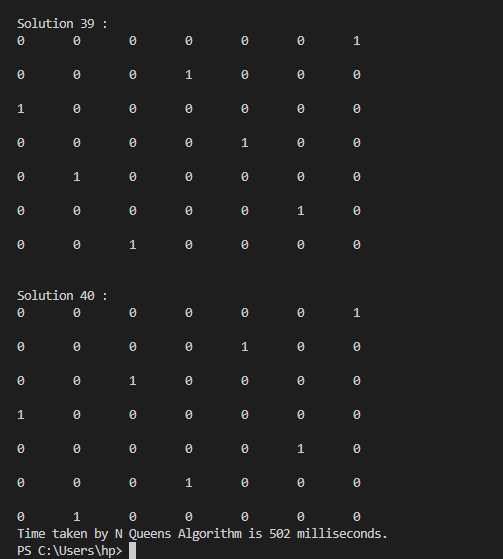


Case 3: When the value of N is 6.

Time taken is 42 milliseconds.

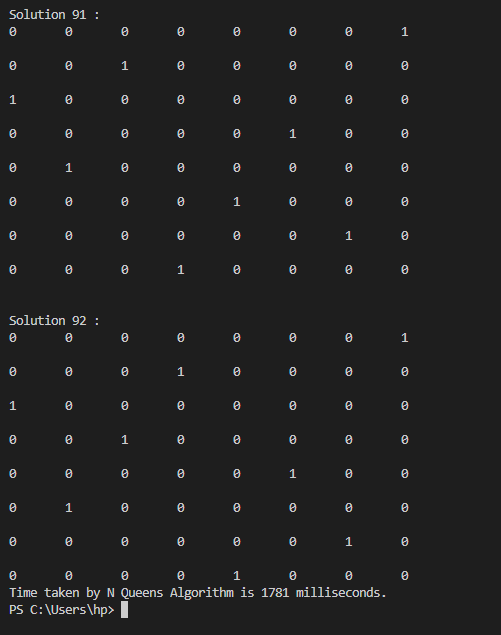


Case 4: When the value of N is 7.



Time taken is 502 milliseconds.

Case 5: When the value of N is 8.



The time taken is 1,781 milliseconds.