N Queen's problem

Problem Submissions Leaderboard Discussions

Write a Java program to Implement N Queen's problem using Back Tracking.

Input Format

4

Constraints

No Constraints

Output Format

0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0

Sample Input 0

```
4
```

Sample Output 0

```
0  0  1  0
1  0  0  0
0  0  0  1
0  1  0  0
```

Contest ends in 9 days

Submissions: 90
Max Score: 10
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

Java 7

```java
import java.util.*;
public class NQueenBacktracking
{
    int n;
    NQueenBacktracking(int n)
    {
        this.n = n;
    }

    /* Display solution*/
    void displaySolution(int queenBoard[][])
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
                System.out.print(" " + queenBoard[i][j] + " ");
                System.out.println();
        }
    }

    /* isSafe() function check if a queen can be placed on queenBoard[row][col]. */
    boolean isSafe(int queenBoard[][], int row, int col)
    {
        int i, j;
        /* for row on left side */
        for (i = 0; i < col; i++)
            if (queenBoard[row][i] == 1)
                return false;
         /* for upper diagonal on left side */
        for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
            if (queenBoard[i][j] == 1)
                return false;
         /* for lower diagonal on left side */
        for (i = row, j = col; j >= 0 && i < n; i++, j--)
            if (queenBoard[i][j] == 1)
                return false;
        return true;
    }

    /*  Utility function for N Queen problem solution */
    boolean utilityFunctionNQueen(int queenBoard[][], int col)
    {
    /* base case when all queens are placed */
       if (col >= n)
           return true;
        /* for this column try placing this queen in all rows one by one */
        for (int i = 0; i < n; i++)
        {
        /* Check is it safe at queenBoard[i][col] */
            if (isSafe(queenBoard, i, col))
            {
                /* Place this queen in board[i][col] */
                queenBoard[i][col] = 1;

                /* recurence to place rest of the queens */
                if (utilityFunctionNQueen(queenBoard, col + 1) == true)
                    return true;

                /* Backtrack: If solution doesn't achieved then remove queen from queenBoard[i][col] */            queenBoard[i][col] = 0;

            }

        }

        /* If we cannot place queen in any row in this column col, then return false */  return false;
    }

    /* uses solveNQUtil () to solve the problem. Note that there may be more than one
    /* solutions, this function prints one of the feasible solutions.*/
    boolean mainSolutionNQueen()
    {
        int queenBoard[][] = new int[n][n];
        if (utilityFunctionNQueen(queenBoard, 0) == false)
        {
            System.out.print("Solution does not exist");
            return false;
        }
        displaySolution(queenBoard);
        return true;
    }

    // Driver main method
    public static void main(String args[])
    {
        int n;
        //System.out.print("Enter size of queen board i.e. N: ");
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        NQueenBacktracking queen = new NQueenBacktracking(n);
        queen.mainSolutionNQueen();
    }
}

```

Line: 1 Col: 1

Upload Code as File   Test against custom input   Run Code   Submit Code

Testcase 0 ✓

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
4
```

Your Output (stdout)

```
0  0  1  0
1  0  0  0
0  0  0  1
0  1  0  0
```

Expected Output

```
0  0  1  0
1  0  0  0
0  0  0  1
0  1  0  0
```