

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 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
```

f t in

Contest ends in 2 months

Submissions: 55

Max Score: 10

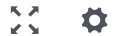
Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

[More](#)

Java 7



```
1 //224G1A0553
2 import java.util.*;
3 public class NQueenBacktracking {
4     int n;
5     NQueenBacktracking(int n) {
6         this.n = n;
7     }
8     /* Display solution*/
9     void displaySolution(int queenBoard[][]) {
10     for (int i = 0; i < n; i++) {
11         for (int j = 0; j < n; j++)
12             System.out.print(" " + queenBoard[i][j] + " ");
13         System.out.println();
14     }
15 }
16 }
17 /* isSafe() function check if a queen can
18 be placed on queenBoard[row][col]. */
19 boolean isSafe(int queenBoard[][], int row, int col) {
```

```
20  int i, j;
21  /* for row on left side */
22  for (i = 0; i < col; i++)
23  if (queenBoard[row][i] == 1)
24  return false;
25
26  /* for upper diagonal on left side */
27  for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
28  if (queenBoard[i][j] == 1)
29  return false;
30
31  /* for lower diagonal on left side */
32  for (i = row, j = col; j >= 0 && i < n; i++, j--)
33  if (queenBoard[i][j] == 1)
34  return false;
35  return true;
36  }
37  /* Utility function for N Queen problem solution */
38  boolean utilityFunctionNQueen(int queenBoard[][], int col) {
39  /* base case when all queens are placed */
40
41  if (col >= n)
42  return true;
43
44  /* for this column try placing this queen in all rows one by one */
45  for (int i = 0; i < n; i++) {
46
47  /* Check is it safe at queenBoard[i][col] */
48  if (isSafe(queenBoard, i, col)) {
49  /* Place this queen in board[i][col] */
50  queenBoard[i][col] = 1;
51
52  /* recurrence to place rest of the queens */
53  if (utilityFunctionNQueen(queenBoard, col + 1) == true)
54  return true;
55
56  /* Backtrack: If solution doesn't achieved then remove queen from queenBoard[i][col] */
57  queenBoard[i][col] = 0;
58
59  }
60
61  }
62
63  /* If we cannot place queen in any row in this column col, then return false */
64  return false;
65  }
66  /* uses solveNQueenUtil () to solve the problem. Note that there may be more than one
67  /* solutions, this function prints one of the feasible solutions.*/
68  boolean mainSolutionNQueen() {
69  int queenBoard[][] = new int[n][n];
70  if (utilityFunctionNQueen(queenBoard, 0) == false) {
71  System.out.print("Solution does not exist");
72  return false;
73
74  }
75  displaySolution(queenBoard);
76  return true;
77  }
78  // Driver main method
79  public static void main(String args[]) {
80  int n;
81  //System.out.print("Enter size of queen board i.e. N: ");
82  Scanner sc = new Scanner(System.in);
83  n = sc.nextInt();
84  NQueenBacktracking queen = new NQueenBacktracking(n);
85  queen.mainSolutionNQueen();
```

```
86  }  
87  }
```

Line: 1 Col: 13

 [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
```