

Competitive programming

Assignment-Backtracking

Date:16/02/2026

Backtracking : N-Queens Problem

Code:

```
import java.util.Scanner;

public class Main

{
    static int N;

    static char[][] board;

    static boolean isSafe(int row, int col)

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

            if (board[i][col] == 'Q')

                return false;

        for (int i = row - 1, j = col - 1; i >= 0 && j >= 0; i--, j--)

            if (board[i][j] == 'Q')

                return false;

        for (int i = row - 1, j = col + 1; i >= 0 && j < N; i--, j++)

            if (board[i][j] == 'Q')

                return false;

        return true;
    }

    static boolean solve(int row)

    {
        if (row == N)
```

```
    return true;

for (int col = 0; col < N; col++)
{
    if (isSafe(row, col))

        board[row][col] = 'Q';

    if (solve(row + 1))

        return true;

    board[row][col] = ':';

}

return false;
}

public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);

    N = sc.nextInt();

    board = new char[N][N];

    for (int i = 0; i < N; i++)

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

            board[i][j] = ':';

    if (solve(0))

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

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

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

                System.out.println();
            }
    }
}
```

```

    }

}

else

{

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

}

sc.close();

}

}

```

The screenshot shows a web-based Java IDE interface from OnlineGDB. The code editor displays a Java file named Main.java. The code implements a backtracking algorithm to solve the N-Queens problem. It includes methods to check if a queen can be placed safely in a given row and column, and a recursive solve method that tries to place queens in all rows. The code uses static variables for the board and a boolean array to track column safety.

```

1+ import java.util.Scanner;
2+ public class Main
3+
4+     static int N;
5+     static char[][] board;
6+     static boolean isSafe(int row, int col)
7+     {
8+         for (int i = 0; i < row; i++)
9+             if (board[i][col] == 'Q')
10+                return false;
11+         for (int i = row + 1, j = col - 1; i >= 0 && j >= 0; i--, j--)
12+             if (board[i][j] == 'Q')
13+                 return false;
14+         for (int i = row + 1, j = col + 1; i >= 0 && j < N; i--, j++)
15+             if (board[i][j] == 'Q')
16+                 return false;
17+         return true;
18+     }
19+     static boolean solve(int row)
20+     {
21+         if (row == N)
22+             return true;
23+         for (int col = 0; col < N; col++)
24+         {
25+             if (isSafe(row, col))
26+             {
27+                 board[row][col] = 'Q';
28+                 if (solve(row + 1))
29+                     return true;
30+                 board[row][col] = '.';
31+             }
32+         }
33+         return false;
34+     }
35+     public static void main(String[] args)
36+     {
37+         Scanner sc = new Scanner(System.in);

```

The screenshot shows the OnlineGDB Java IDE interface. The main window displays a Java code editor with the file name 'Main.java'. The code implements a N-Queens solver. The input section contains a 4x4 board configuration:

```
4
. Q .
. . . Q
Q . . .
. . Q .
```

The output section shows the solved board state:

```
. Q .
. . . Q
Q . . .
. . Q .
```

The status bar at the bottom right indicates the date and time as 16-02-2026 11:43.

```
36 public static void main(String[] args)
37 {
38     Scanner sc = new Scanner(System.in);
39     int N = sc.nextInt();
40     char board[][] = new char[N][N];
41     for (int i = 0; i < N; i++)
42         for (int j = 0; j < N; j++)
43             board[i][j] = '.';
44     if (solve())
45     {
46         for (int i = 0; i < N; i++)
47             for (int j = 0; j < N; j++)
48                 System.out.print(board[i][j] + " ");
49                 System.out.println();
50     }
51     else
52     {
53         System.out.println("No solution exists");
54     }
55     sc.close();
56 }
57
58
59 }
```