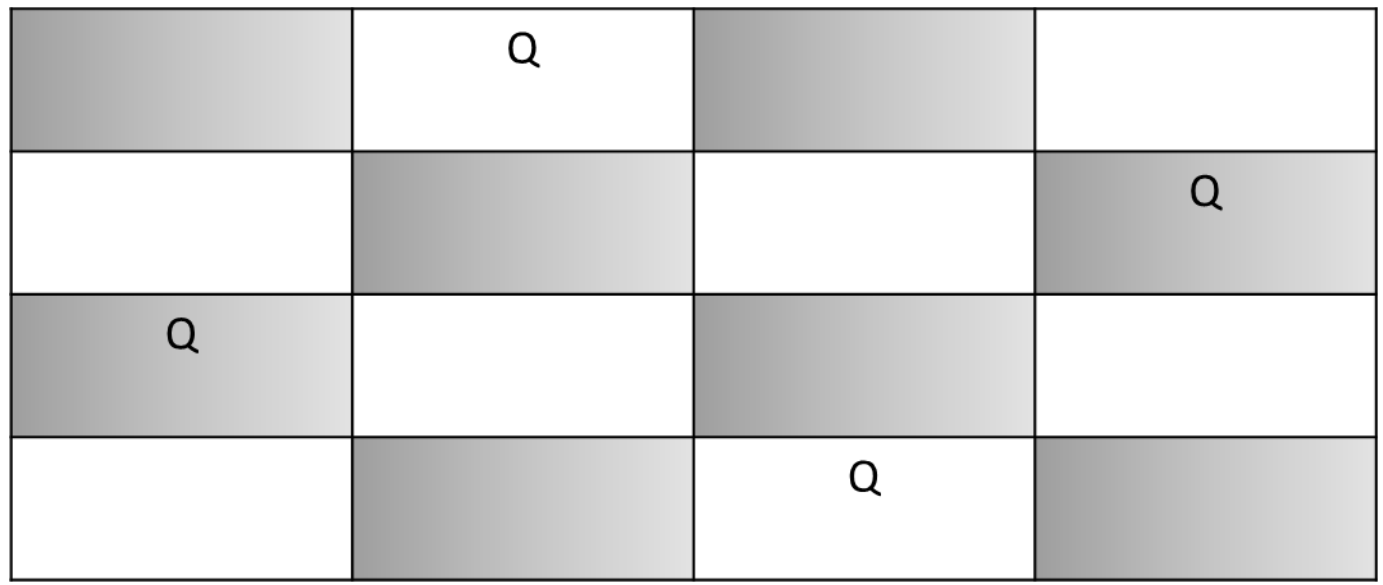
N-Queens Problem

Introduction

- This is a classic example of backtracking
- Problem is to place n queens in a n x n chessboard so that no two queens can attack each other
- No two queens are in same row, same column or having same diagonal
- In any solution to the n-Queens problem, there is exactly one queen in each row.

Introduction

- The idea is to place queens one by one in different columns, starting from the leftmost column.
- When we place a queen in a column, we check for clashes with already placed queens.
- In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution.
- If we do not find such a row due to clashes then we backtrack and return false.


Q
Q
Q
Q

- The expected output is a matrix which has Qs for the blocks where queens are placed. For example, following is the output matrix for above 4 queen solution
- Solution : or Solution

*	Q	*	*	0 1 0 0
*	*	*	Q	0 0 0 1
Q	*	*	*	1 0 0 0
*	*	Q	*	0 0 1 0

N-Queens problem

1) Start in the leftmost column

2) If all queens are placed

    return true

3) Try all rows in the current column.

    Do following for every tried row.

    a) If the queen can be placed safely in this row

    then mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.

b) If placing the queen in [row, column] leads to a solution then return true.

c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.

4) If all rows have been tried and nothing worked,

    return false to trigger backtracking.

- https://www.onlinegdb.com/SyCqpfzhM