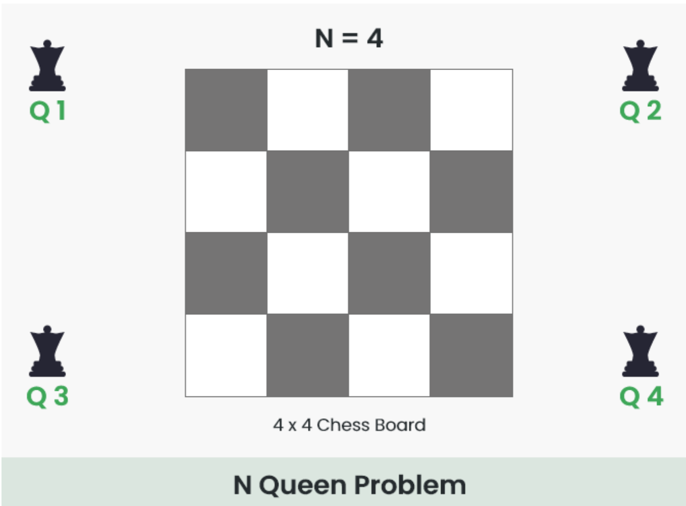
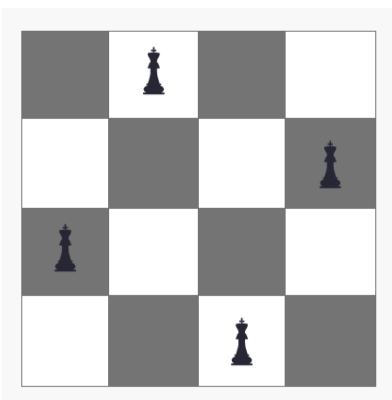
N-QUEENS PROBLEM

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.



For example, the following is a solution for the 4 Queen problem.



The expected output is in the form of a matrix that has 'Q's for the blocks where queens are placed and the empty spaces are represented by '.' . For example, the following is the output matrix for the above 4-Queen solution.

. Q . .

. . . Q

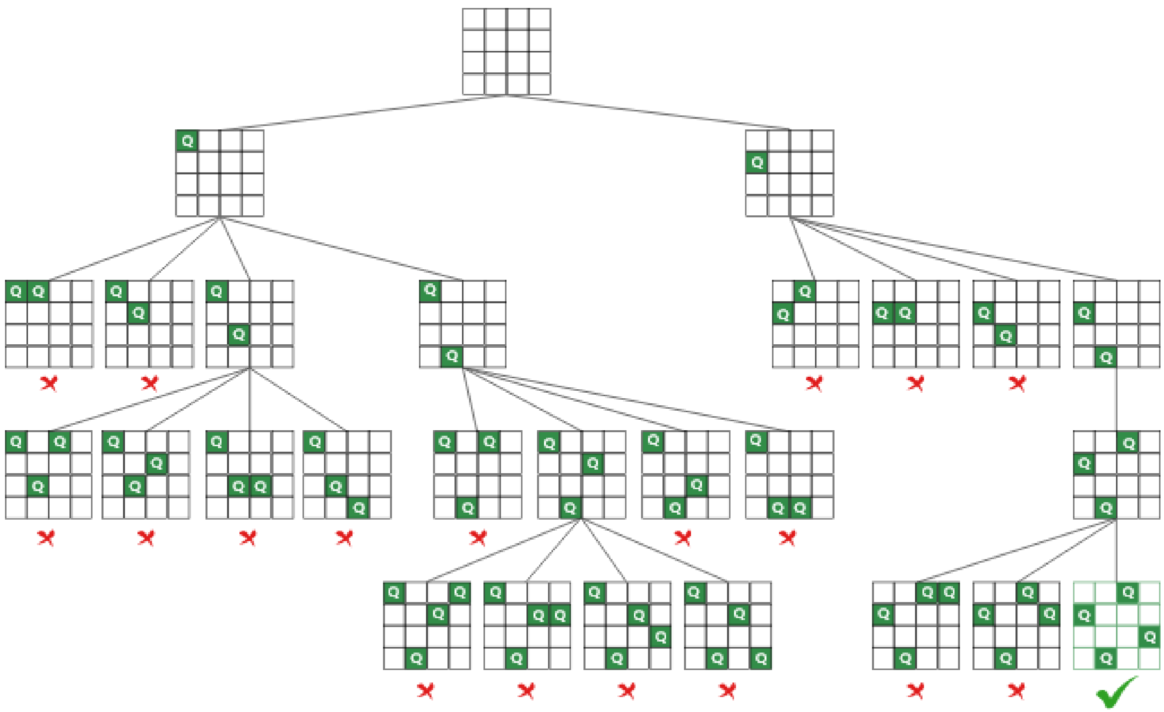
Q . . .

. . Q .

N Queen Problem using Backtracking:

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.

Below is the recursive tree of the above approach:



Follow the steps mentioned below to implement the idea:

- Start in the leftmost column
- If all queens are placed return true
- Try all rows in the current column. Do the following for every row.
    - 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.

    - If placing the queen in [row, column] leads to a solution then return true.
    - If placing queen doesn't lead to a solution then unmark this [row, column] then backtrack and try other rows.
- If all rows have been tried and valid solution is not found return false to trigger backtracking.