Algorithm for N-Queen:-

* Make isSafe Function

1) First of all we check that if its safe to place a queen

2) Check these condtions:

- No Queen in the same column above this position
- No Queen in the upper left diagonal
- No Queen in the upper right diagonal.

3) Returns true if all conditions met otherwise false.

4) We make Function of nQueen

Base Case:

~~(if (all Queens are placed) || (x == n))~~

- if All Queens are placed ($x >= n$) return true.

5) For each coulmn in the current row try placing the Queen.

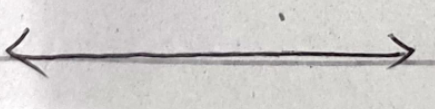
- if 'isSafe' returns 'true'
  - place the queen at(x, col).
  - Recue to place the rest of Queens on the next row ('x+1').
  - If the Recursion retuns 'true' then solution is found, return ('true')

- if the recursion returns 'false' backtrack by removing the queen from (x, col)

- if no suitable coulmn is found return 'false'

6) In Main function we create Chessboard [4][4] and initilize to 0.

- Call 'nQueen' function
  - if 'nQueen' retunns 'true'
    Cout << Solution Found.
  - if 'nQueen' returns 'false'
    cout << NoSolution exists



In order to solve N Queens Problem we must know about what is Backtracking:-

Back Tracking:-

In backtracking, we start with one possible move out of many available moves, we then try to solve the problem.

If we are able to solve the problem with the selected move then we will print the solution

Else we will backtrack and select some other move and try to solve it.

If none of the moves works out we claim that there is no solution for the problem.

What is N Queens Problem?

The N- Queens Problem is a chess puzzle in which N chess queens must be placed on N×N chessboard so that no two queens attack each other.