**Design and Analysis of Algorithm**

**Experiment No. : 12**

**Write a program to implement N Queen problem using backtracking approach**

Experiment No. 12

1. **Aim:** Write a program to implement N Queen problem using backtracking approach.
2. **Algorithm**

This problem is to find an arrangement of N queens on a chess board, such that no queen can attack any other queens on the board.The chess queens can attack in any direction as horizontal, vertical, horizontal and diagonal way.A binary matrix is used to display the positions of N Queens, where no queens can attack other queens.

**Input and Output**

**Input:**

The size of a chess board. Generally, it is 8. as (8 x 8 is the size of a normal chess board.)

**Output:**

The matrix that represents in which row and column the N Queens can be placed.If the solution does not exist, it will return false.

1 0 0 0 0 0 0 0

0 0 0 0 0 0 1 0

0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 1

0 1 0 0 0 0 0 0

0 0 0 1 0 0 0 0

0 0 0 0 0 1 0 0

0 0 1 0 0 0 0 0

In this output, the value 1 indicates the correct place for the queens.

The 0 denotes the blank spaces on the chess board.

**Algorithm**

**isValid(board, row, col)**

**Input: The chess board, row and the column of the board.**

**Output −** True when placing a queen in row and place position is a valid or not.

Begin

if there is a queen at the left of current col, then

return false

if there is a queen at the left upper diagonal, then

return false

if there is a queen at the left lower diagonal, then

return false;

return true //otherwise it is valid place

End

**solveNQueen(board, col)**

**Input −** The chess board, the col where the queen is trying to be placed.

**Output −** The position matrix where queens are placed.

Begin

if all columns are filled, then

return true

for each row of the board, do

if isValid(board, i, col), then

set queen at place (i, col) in the board

if solveNQueen(board, col+1) = true, then

return true

otherwise remove queen from place (i, col) from board.

done

return false

End

**Conclusion and Discussion**: For finding a single solution where the first queen ‘Q’ has been assigned the first column and can be put on N positions, the second queen has been assigned the second column and would choose from N-1 possible positions and so on; the time complexity is **O ( N ) \* ( N - 1 ) \* ( N - 2 ) \* … 1 )**. i.e The worst-case time complexity is **O ( N! )**. Thus, for finding all the solutions to the N Queens problem the time complexity runs in **polynomial time**.