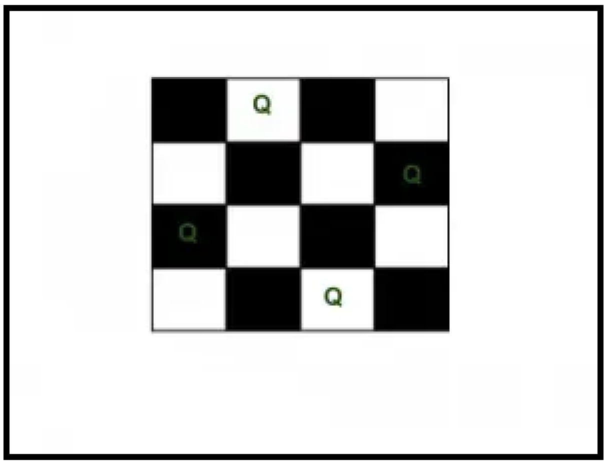
Vashisth Zatakia
60003200163 B3

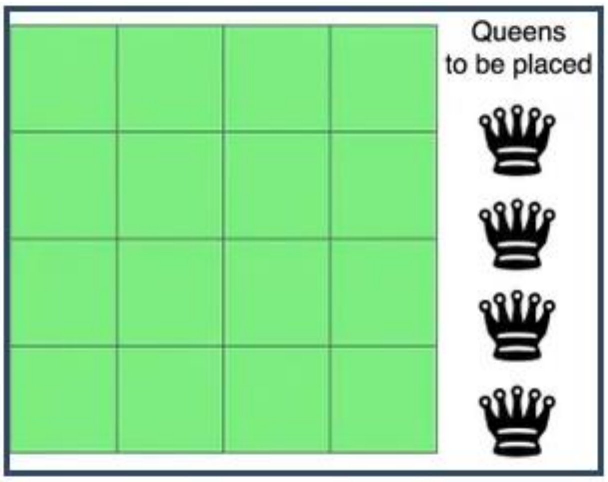
AIM/OBJECTIVE: To implement N-Queens problem using backtracking

Theory:

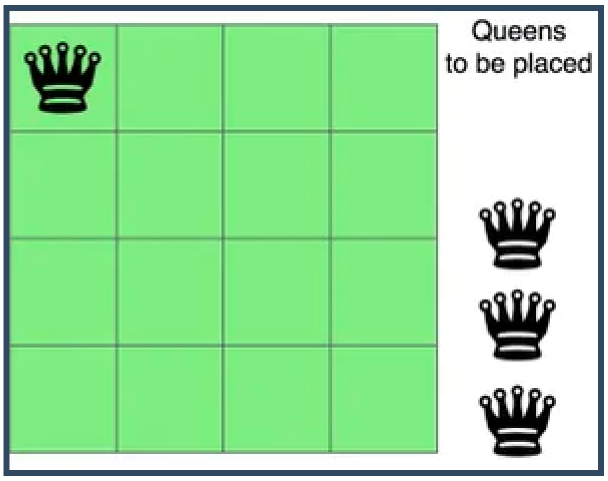
N Queen Problem : N Queens Problem is a famous puzzle in which n-queens are to be placed on a nxn chess board such that no two queens are in the same row, column or diagonal.The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.
For example, following is a solution for 4 Queen problem.



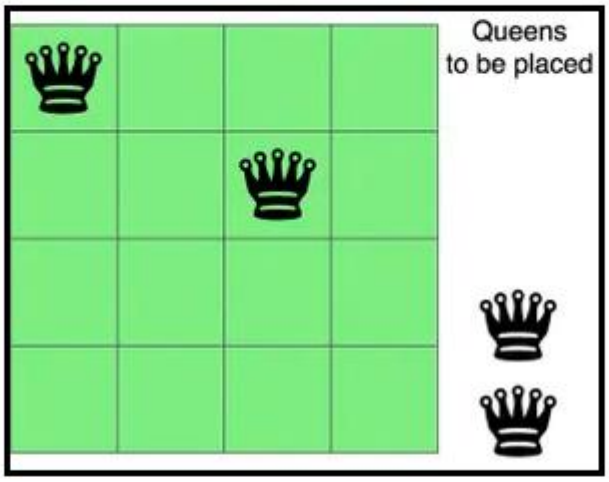
One of the most common examples of the backtracking is to arrange N queens on an NxN chessboard such that no queen can strike down any other queen. A queen can attack horizontally, vertically, or diagonally. The solution to this problem is also attempted in a similar way. We first place the first queen anywhere arbitrarily and then place the next queen in any of the safe places. We continue this process until the number of unplaced queens becomes zero (a solution is found) or no safe place is left. If no safe place is left, then we change the position of the previously placed queen.

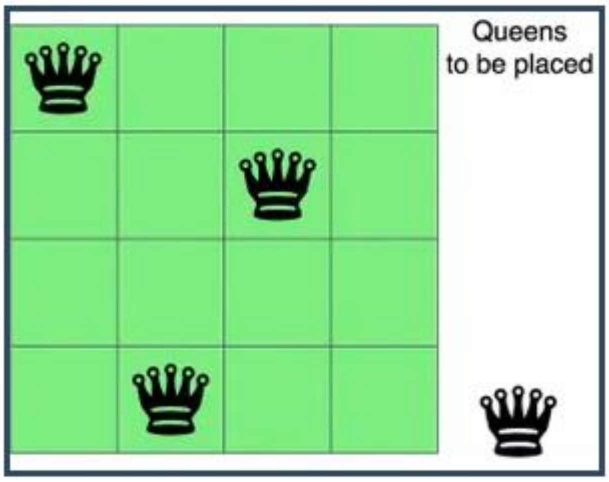


The above picture shows an NxN chessboard and we have to place N queens on it. So, we will start by placing the first queen.



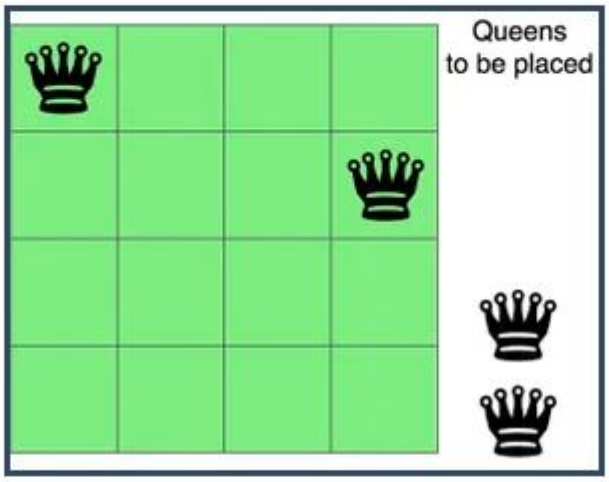
Now, the second step is to place the second queen in a safe position and then the third queen.



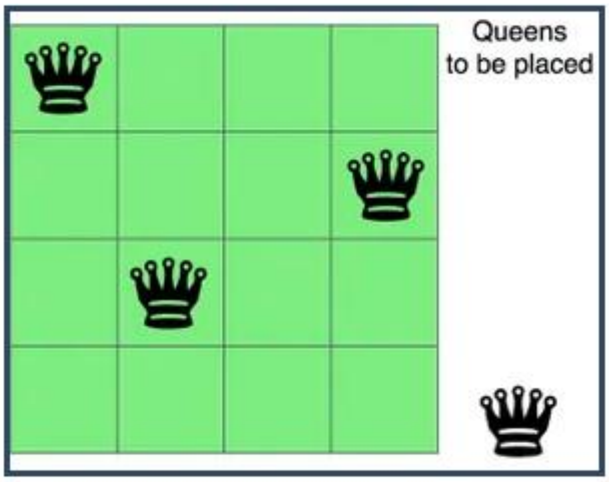


Now, you can see that there is no safe place where we can put the last queen. So, we will just change the position of the previous queen. And this is backtracking.

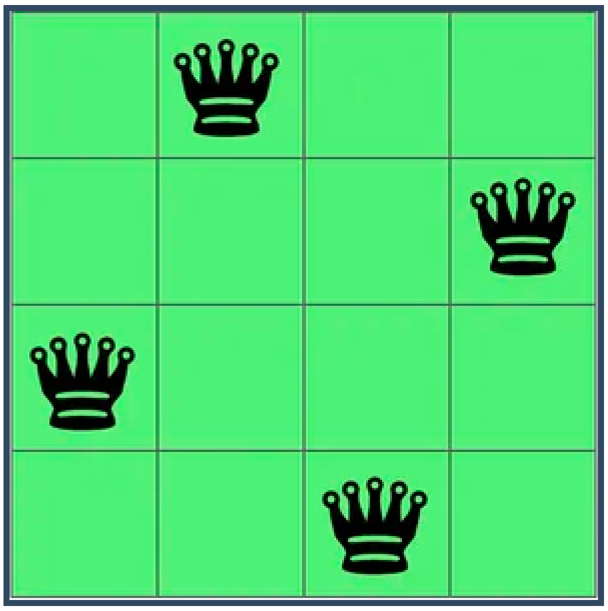
Also, there is no other position where we can place the third queen so we will go back one more step and change the position of the second queen.



And now we will place the third queen again in a safe position until we find a solution.



We will continue this process and finally, we will get the solution as shown below.



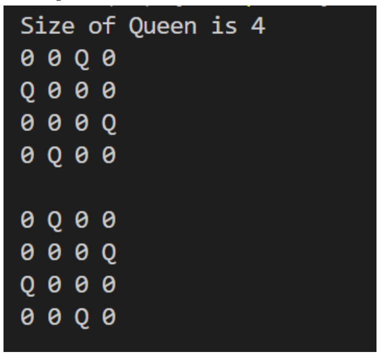
The Time Complexity is O(n^2) because we are selecting if we can put or not put a Queen at that place. The space is the board size times the result.

Vashisth Zatakia
60003200163 B3

Code:

```c
#include <stdio.h>
#include <stdbool.h>
void printSolution(int n, int board[n][n])
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (board[i][j])
            {
                printf("Q ");
            }
            else
            {
                printf("%d ", board[i][j]);
            }
        }
        printf("\n");
    }
    printf("\n");
}
bool isAttacking(int n, int board[n][n], int row, int col)
{
    for (int i = 0; i < col; i++) // if queens placed in same columns
    {
        if (board[row][i])
        {
            return true;
        }
    }
    for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) // if queen placed is conflicting with left upper diagonal
    {
        if (board[i][j])
        {
            return true;
        }
    }
    for (int i = row, j = col; i < n && j >= 0; i++, j--) // if queen placed is conflicting with left lower diagonal
    {
        if (board[i][j])
        {
            return true;
        }
    }
    return false;
}
bool solvenq(int n, int board[n][n], int col)
{
    if (col >= n)
    {
        printSolution(n, board);
    }
    for (int i = 0; i < n; i++)
    {
        if (!isAttacking(n, board, i, col))
        {
            board[i][col] = 1;
            if (solvenq(n, board, col + 1))
            {
                return true;
            }
            board[i][col] = 0;
        }
    }
    return false;
}
int main()
{
    const int n = 4;
    printf("Size of Queen is 4\n");
    int board[4][4] = {{0, 0, 0, 0},
                       {0, 0, 0, 0},
                       {0, 0, 0, 0},
                       {0, 0, 0, 0}};
    solvenq(n, board, 0);
    return 0;
}
```

Output:



```
Size of Queen is 4
0 0 Q 0
Q 0 0 0
0 0 0 Q
0 Q 0 0

0 Q 0 0
0 0 0 Q
Q 0 0 0
0 0 Q 0
```

Conclusion: Thus, we have implemented n-queen problem using backtracking and have found all possible solutions.