Date:

Exp: 1

AIM:

To implement N-Queens problem in python using backtracking

ALGORITHM.

Step 1: Start

Step 2: Create an empty N×N chessboard initialised with zeroes

Step 3: Start from the first row (0) and try placing a queen in one of the columns of the current row

Step 4: for each column in the current row, check if placing the queen in that column is safe.

* NO other queens should be placed in the same columns
* No queen should on the left upper diagonal
* No queen should be on the right upper diagonal

Step 5: if placing the queen is safe, place it. move to the next row and repeat the process

Step 6: if no valid position is found placing a queen in the current row, backtracking by removing the queen from the previous row and try placing it in another column.

Step 7: If queens are successfully placed, a solution is found

Step 8: Print the chessboard configuration with queens placed on the board

Step 9: If no solution is found, output NO

Step 10: Stop

Date :

Code:

```
def is_safe(board, row, col, N):
    for i in range(row):
        if board[i][col] == 1:
            return false
    For i,j in zip (range (row, -1, -1), range (col, -1, -1))
        if board [i][j] == 1:
            return false
    for i, j in zip ( range (row, -1, -1), range (col, N)):
        if board [i][j] == 1:
            return false
    return True

def Solve_n queens (board, row, N):
        if row >= N:
            return True
    for col in range (N)
        if is_safe (board, row, col, N):
            board [row] [col] = 1
        if Solve_nqueens (board, row+1, N):
            return True
        board [row] [col] = 0
    return false
def Print_Solution (board, N):
    for row in board:
        Print (" ", join ("Q" if col == 1 else ". " for
                col in row ))
def n-queens (N)
    board = [[0]*N for _ in range[N]]

if Solve_nqueens (boards, 0, N)
    Print (" Solution for {} - Queens :" format (N),
    Print. Solution (board, N)
else
    Print ('NO Solution". format (N))
```

```
N = int ( input ( " Enter the value of N: "))
n_queens (N)
```

Enter no of Queens you want : 8

```
[ Q - - - - - - - ]
[ - - - - Q - - - ]
[ - - - - - - - Q ]
[ - - - - - Q - - ]
[ - - Q - - - - - ]
[ - - - - - - Q - ]
[ - Q - - - - - - ]
[ - - - Q - - - - ]
```

RESULT:

Thus, N-Queens problems using backtracking is implemented