N-Queen Problem Solution Report

M.Junaid 109(4B)

Introduction

This report explains a solution to the N-Queen problem, a classic chess puzzle. The N-Queen problem asks how to place N queens on an N×N chessboard so that no two queens threaten each other. This means no two queens can share the same row, column, or diagonal.

Algorithm Explanation

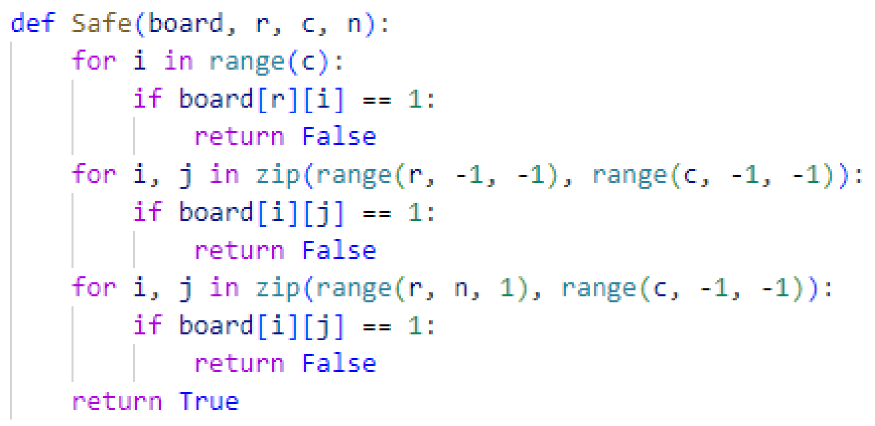
Safety Check Function

The Safe() function checks if a queen can be placed at a specific position without threatening any other queens. It verifies three conditions:

1. No queen in the same row to the left

2. No queen in the upper-left diagonal

3. No queen in the lower-left diagonal



```python
def Safe(board, r, c, n):
    for i in range(c):
        if board[r][i] == 1:
            return False
    for i, j in zip(range(r, -1, -1), range(c, -1, -1)):
        if board[i][j] == 1:
            return False
    for i, j in zip(range(r, n, 1), range(c, -1, -1)):
        if board[i][j] == 1:
            return False
    return True
```

Recursive Solver Function

The solveQ() function is works as follows:

1. If all queens are placed return success

2. Try placing a queen in each row of the current column

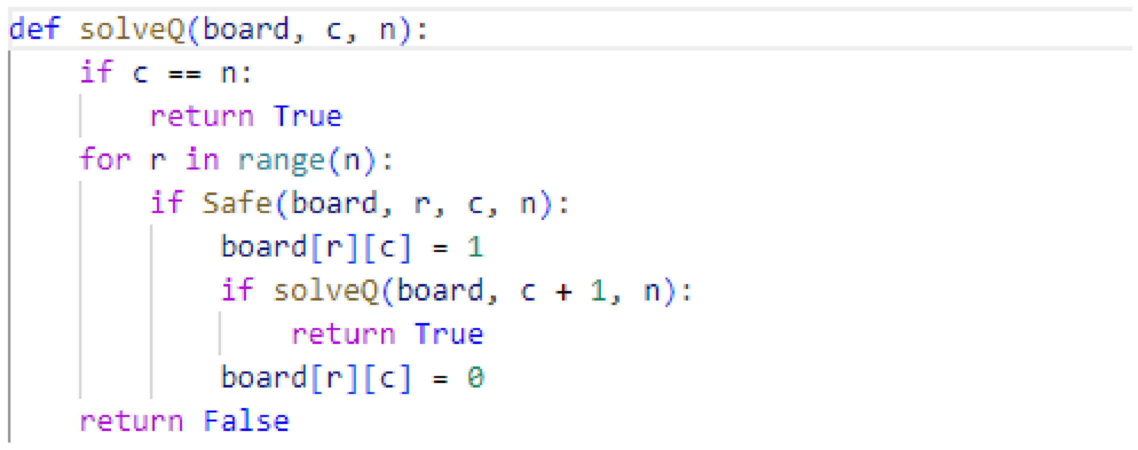
3. For each placement, check if it's safe using the Safe() function

4. If safe, place the queen and recursively try to place the remaining queens

5. If the recursive call succeeds, return success

6. If the recursive call fails, remove the queen (backtrack) and try the next row

7. If no row works, return failure



```python
def solveQ(board, c, n):
    if c == n:
        return True
    for r in range(n):
        if Safe(board, r, c, n):
            board[r][c] = 1
            if solveQ(board, c + 1, n):
                return True
            board[r][c] = 0
    return False
```

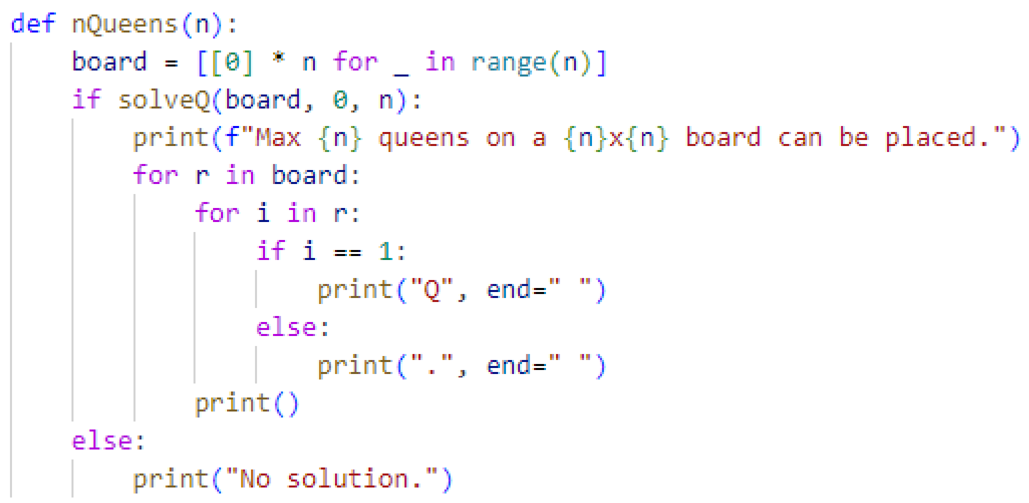
Main Function

The nQueens() function initializes the board by taking input from user and calls the function solveQ(). If a solution is found, it prints the result.

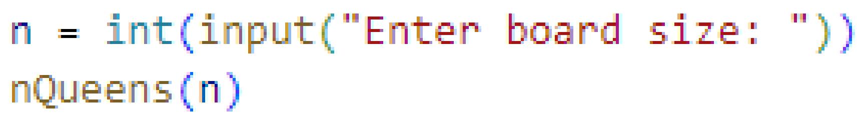
1. Creates an empty N×N board filled with zeros

2. Calls the function solveQ() starting from column 0

3. Prints the solution if one exists, or "No solution" otherwise

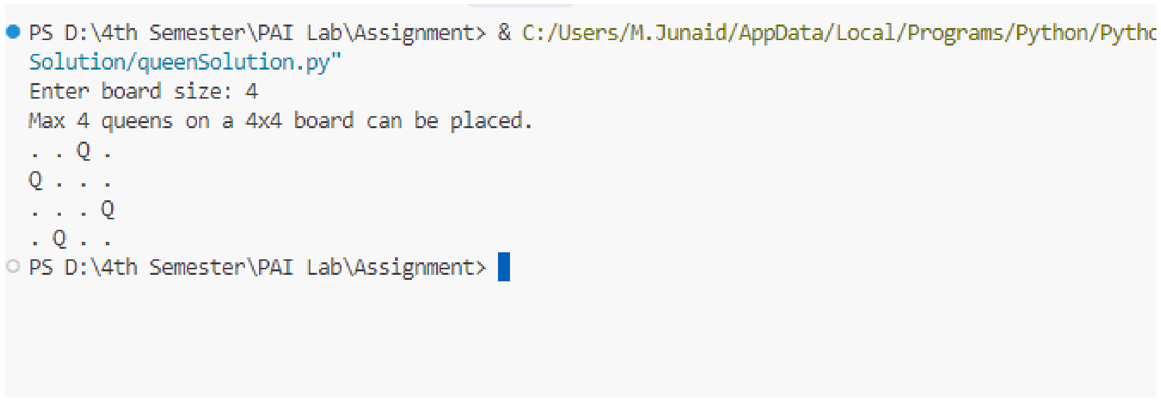


```python
def nQueens(n):
    board = [[0] * n for _ in range(n)]
    if solveQ(board, 0, n):
        print(f"Max {n} queens on a {n}x{n} board can be placed.")
        for r in board:
            for i in r:
                if i == 1:
                    print("Q", end=" ")
                else:
                    print(".", end=" ")
            print()
    else:
        print("No solution.")
```



```python
n = int(input("Enter board size: "))
nQueens(n)
```

Results



```
PS D:\4th Semester\PAI Lab\Assignment> & C:/Users/M.Junaid/AppData/Local/Programs/Python/Pytho
Solution/queenSolution.py"
Enter board size: 4
Max 4 queens on a 4x4 board can be placed.
. . Q .
Q . . .
. . . Q
. Q . .
PS D:\4th Semester\PAI Lab\Assignment>
```